



Project Learning for better Establishment on Labor market

DOCUMENTATIONS OF STUDENTS' PROJECTS



Co-funded by the
Erasmus+ Programme
of the European Union

Attendance system with RFID (CZ-HU) _____	2
Egg painting robot (SK-PL) _____	19
Electronic Price Tag (SK-SI) _____	38
Greenhouse (CZ-SI) _____	61
Intelligent mirror (SI-PL) _____	80
Key borrowing system (SK-HU) _____	92
Lixie Clock (SK-PL) _____	121
RFID lock (CZ-PL) _____	135
Solar tracker (PL-SI) _____	150
Sorting machine documentation (SK-CZ) _____	171
Temperature regulation (CZ-PL) _____	199
Tesla Coil (HU-SI) _____	255
Voltage meter with voice output (SK-HU) _____	270
Voltmeter web (CZ-SK) _____	310
Voting for school lunches (SK-SI) _____	338
Weather station (CZ-PL) _____	356
Weather station Arduino (CZ-HU) _____	372
WordClock (SK-PL) _____	390



Co-funded by the
Erasmus+ Programme
of the European Union



Attendance system with RFID

Hardware and software documentation

*This project was made under the cooperation of Erasmus+ KA2 P.L.E.L. program
between the Hungarian and Czech partners*

2019.

Table of contents

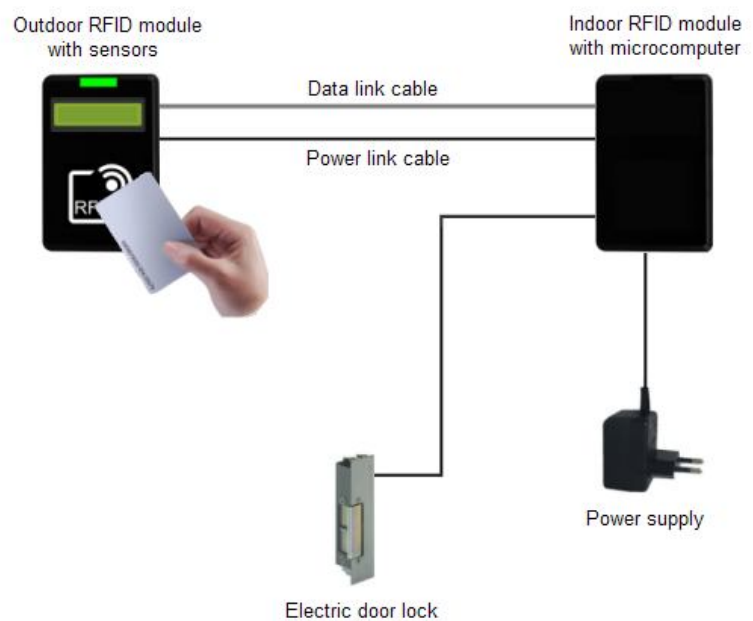
1. Implementation of the hardware	3
1.1. Prior description	3
1.2. Concept of the system.....	4
2. Maps of the hardware	4
3. Components	8
3.1. List of the components.....	8
3.2. Pictures of the components	9
4. Implementation of the hardware	10
4.1. Assembled final design	10
4.2. Inside views of the units	11
5. Implementation of the software	12
5.1. Prior description	12
5.2. Working process of the system	12
5.3. Format of the database file.....	13
5.4. Working process of the software.....	14
5.5. Flowchart of the program.....	15
5.6. Flowchart of the administrator's menu	16
6. Appendix.....	17
7. Final statement	17

1. Implementation of the hardware

1.1. Prior description

The task of the RFID access control system is providing the one-way passage through a door with a magnetic card, if the passing person identifies and authenticates himself/herself with an RFID tag. The person can open the magnetic lock from one direction if he/she is registered in the system.

This can be seen on the figure bellow.



1.2. Concept of the system

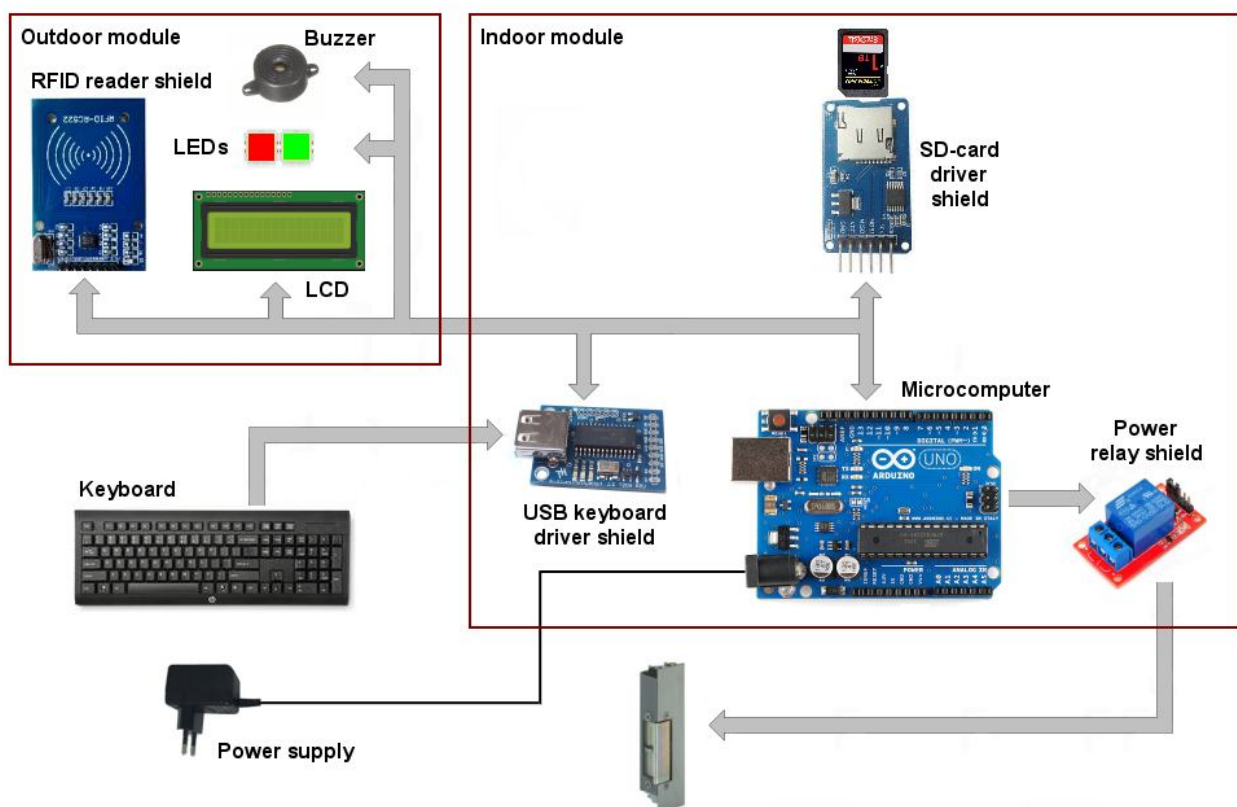
Beside the door, on one side of the door, an RFID reading device is placed. On the reading device can be found an LCD display and an authorization light.

To open the door the RFID tag has to keep close to the reader. If the person is registered, the registered person's name appears on the display of the reader, the enable light goes on, and an audible beep sounds, and the magnetic lock of the door opens. In the case of an unregistered person, an inhibitory text goes on the display, an inhibitory light goes on the display, and an inhibitory sound signal is heard and the door will remain closed.

The system consists of an outdoor and an indoor unit.

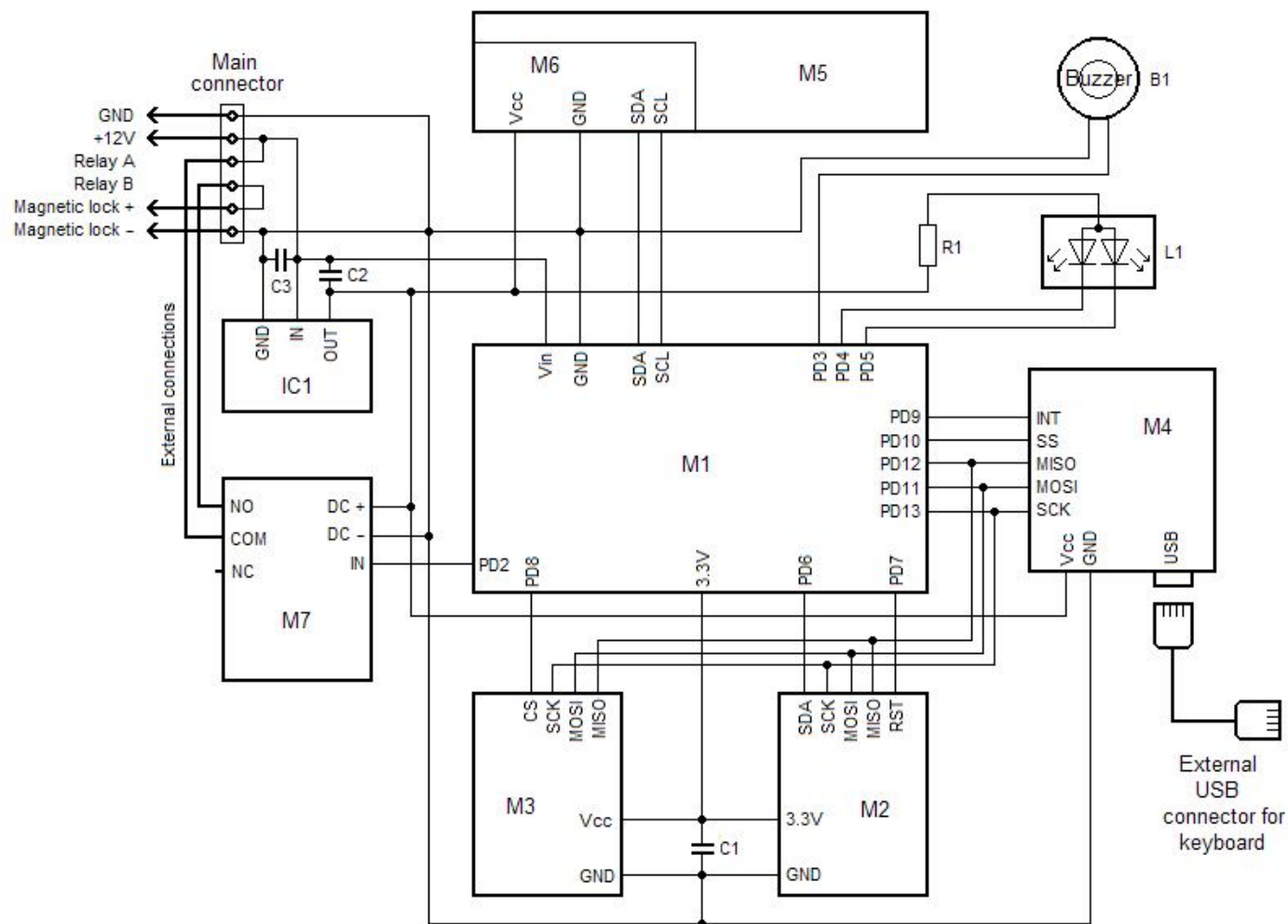
The outdoor unit contains only the RFID reader and the display. The indoor unit consists of the microcomputer and the other components.

This can be seen on the figure bellow.



2. Maps of the hardware

The maps of the hardware can be seen on the following pages. It contains the hardware wiring circuit diagram, and the maps of the printed circuit boards of the indoor and outdoor units.



Company:



MSzC Kandó Kálmán
Informatikai Szakgimnáziuma
3525 Miskolc, Palóczy u. 3.

Title: Attendance system with RFID

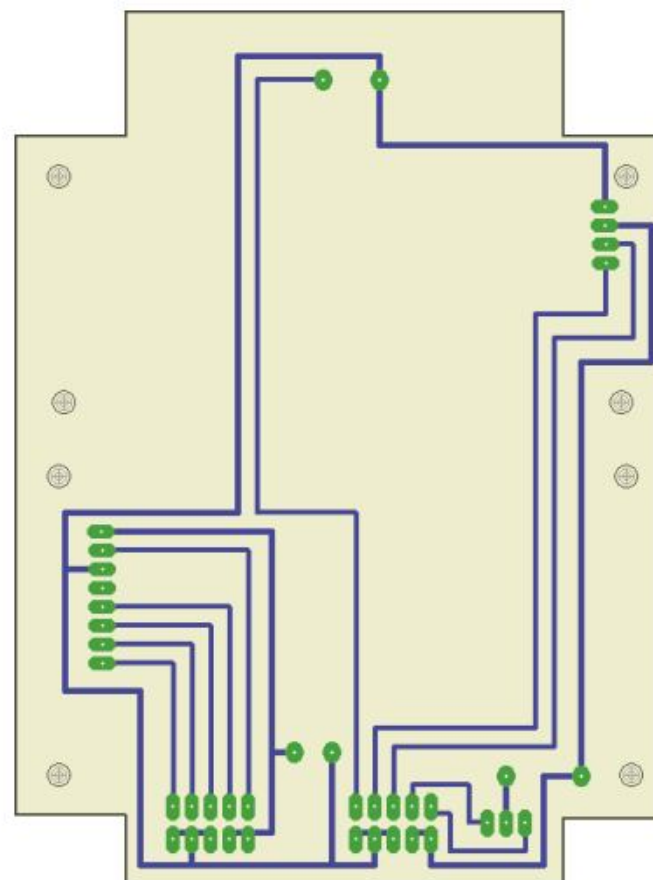
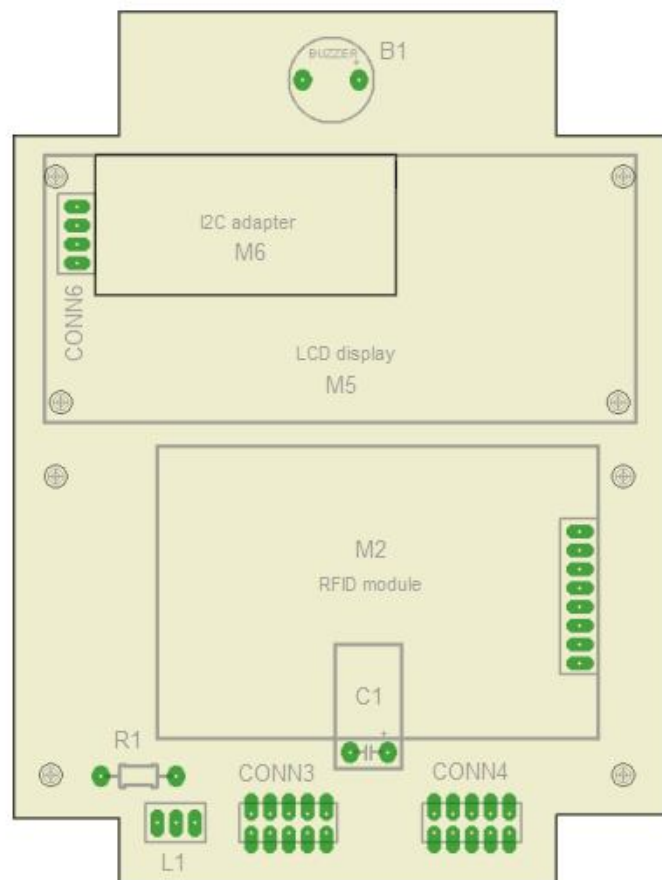
Subtitle: Wiring schematic

Sheet: 1/3

Rev: 1.0

Date: 2018.11.04.

Drawn by: Szedlák Botond



Company:



MSzC Kandó Kálmán
Informatikai Szakgimnáziuma
3525 Miskolc, Palóczy u. 3.

Title: Attendance system with RFID

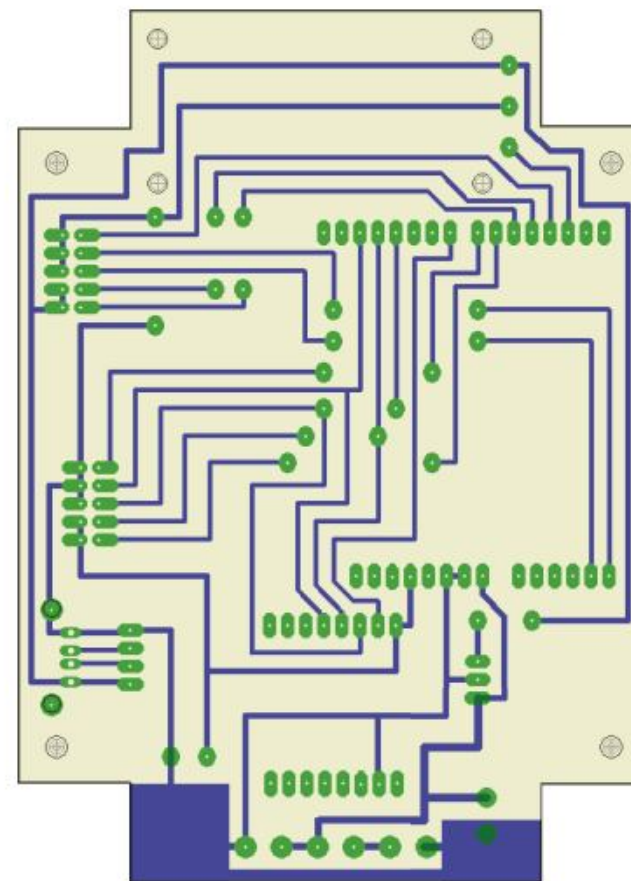
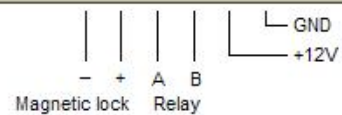
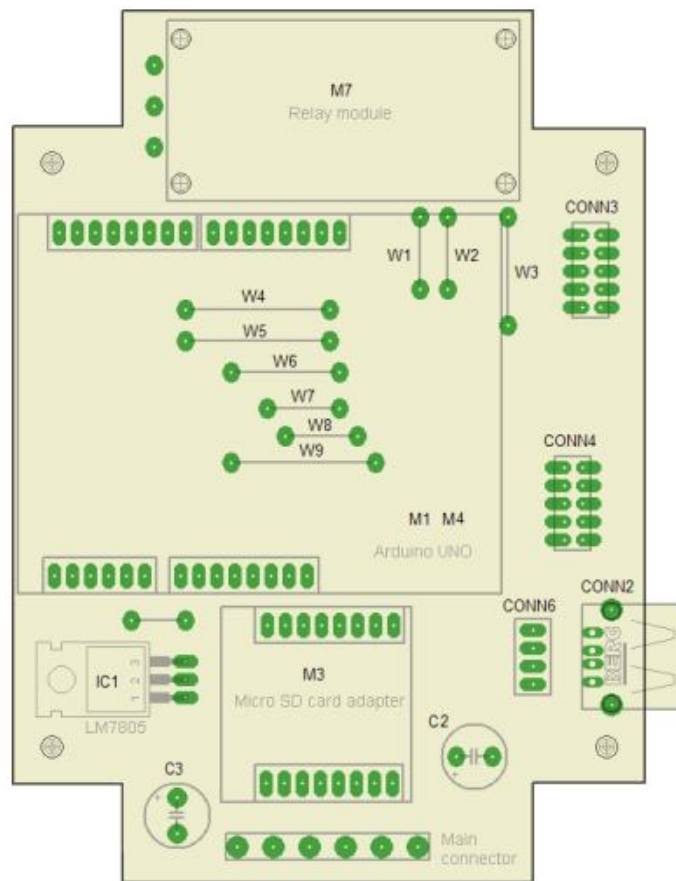
Subtitle: PCB layout of outdoor module
LCD display and RFID reader

Sheet: 2/3

Rev: 1.0

Date: 2018.11.04.

Drawn by: Szedlák Botond



Company:



MSzC Kandó Kálmán
Informatikai Szakgimnáziuma
3525 Miskolc, Palóczy u. 3.

Title: Weather forecast station with Arduino

Subtitle: PCB layout of indoor module

Sheet: 3/3

Rev: 1.0

Date: 2018.11.04.

Drawn by: Szedlák Botond

3. Components

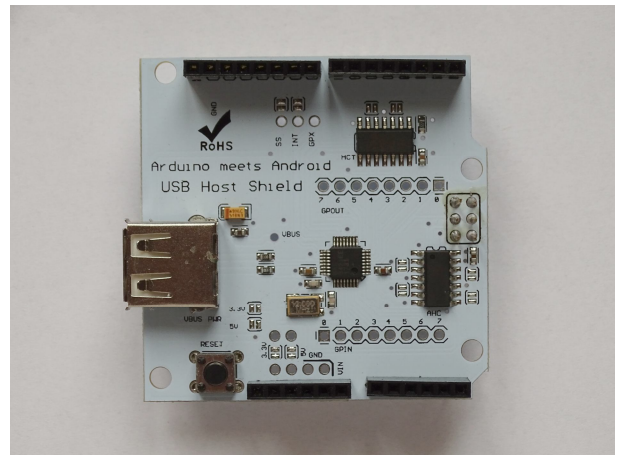
3.1. List of the components

No.	Device name	Type / Value
1.	M1	Arduino UNO R3 microcomputer
2.	M2	MFRC 522 RFID module
3.	M3	SD card adapter module
4.	M4	USB host shield module
5.	M5	LCD display module, 1602
6.	M6	I ² C adapter module
7.	M7	Relay module, 5V
8.	B1	DC buzzer, 5V
9.	L1	Red/Green duo-LED
10.	R1	220 Ω
11.	C1	100 μ F
12.	C2	100 μ F
13.	C3	2200 μ F
14.	IC1	LM7805
15.	CONN1	PCB terminal block connector
16.	CONN2	External USB A connector
17.	CONN3	Pinheader connector, 2x5 pins
18.	CONN4	Pinheader connector, 2x5 pins
19.	CONN5	Pinheader connector, 1x4 pins
20.	CONN6	Pinheader connector, 1x4 pins

3.2. Pictures of the components



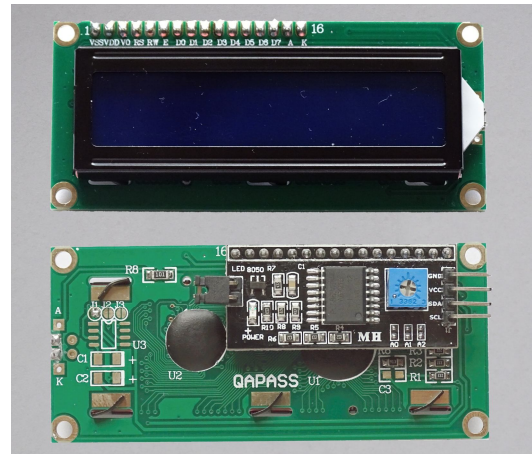
Arduino Uno microcomputer



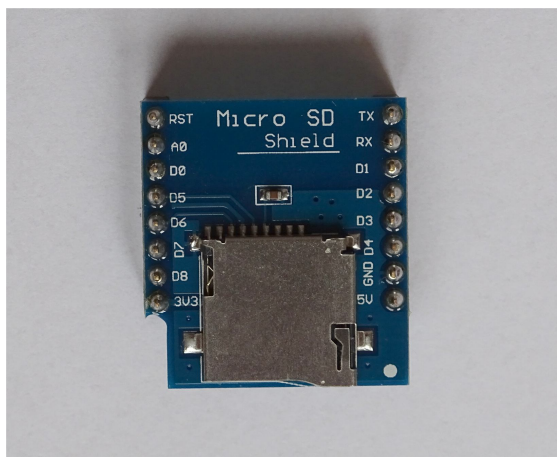
USB host shield



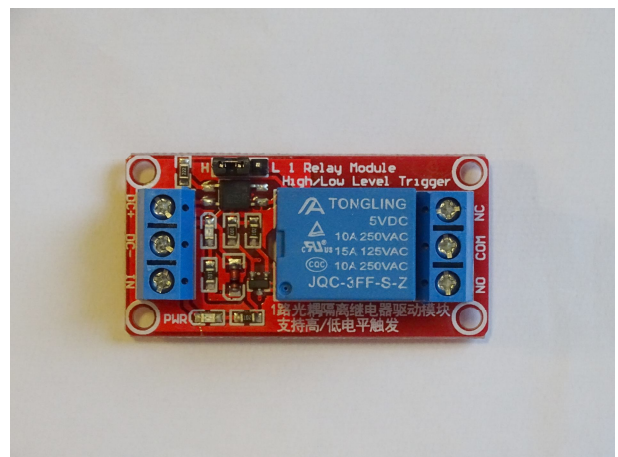
RFID MFRC-522 shield



LCD display with I²C bus driver



Micro SD card slot



Relay shield

4. Implementation of the hardware

4.1. Final assembly design

The final design can be seen the figures bellow. The magnetic lock and the attendance system units are assembled on the model of the door. The system gets electric power from a built-in power supply which is also assembled in the same model. It uses 230 V(AC) power.



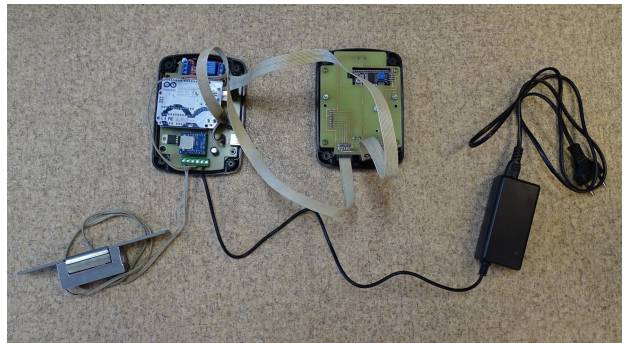
Outdoor view



Indoor view

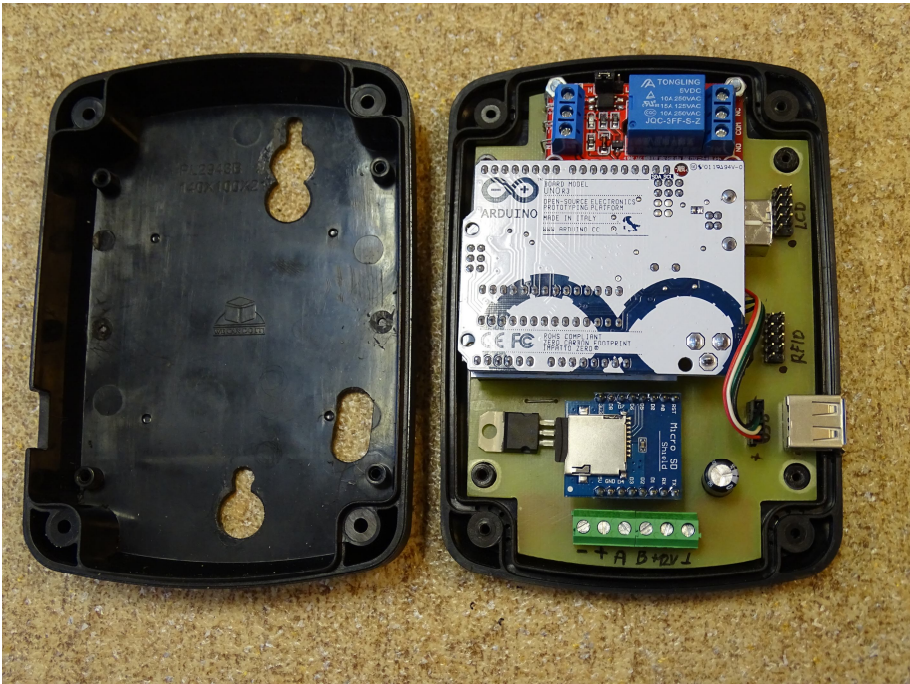


Keyboard connection for "administrator mode"

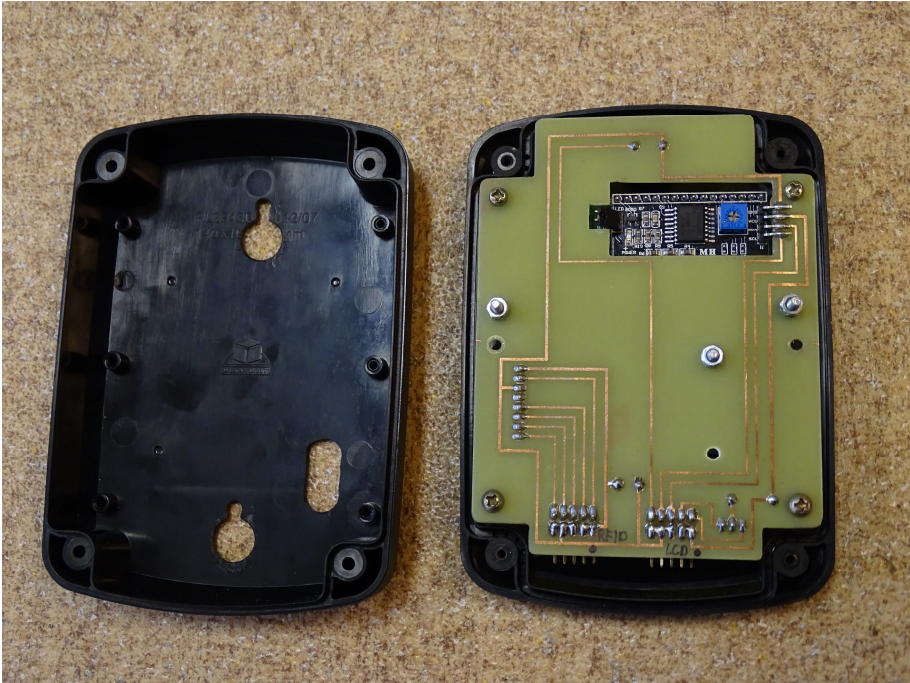


Connection of the units and other components

4.2. Inside views of the units



Inside view of the indoor unit



Inside view of the outdoor unit

5. Implementation of the software

5.1. Prior description

The implemented system with the supported features by the following services:

- Handling a passive RFID tag: card, key-ring, etc.
- Registering RFID tags on the system:

The registration data should be connected to the controller of the microcomputer on a SD-card. In this database the names of the people can also be entered. The database should have a simple structure (*nuid.text* file).

- The treatment of the keyboard:

If you want to register a new person, you can have a keyboard connection to the indoor unit (through an external USB port), thus we can give the name of this person and the registration can be done easily with some different keystrokes.

- Display the name of the entering person on the LCD display.

Visualization of the permission of entering, and the visualization of the light of prohibition (green and red LED), and make a tone sound.

5.2. Working process of the system

After switching on the power supply, the Arduino microcomputer boots up, and goes to stand-by state. At this time the system is ready to work, and the “Ready” message appears on the display.

Operational states:

a) “RFID mode”

If we hold an RFID tag (e.g. card) close to the RFID reader, then the software reads the identifier of the tag (NUID), and searches this ID between the IDs stored on the SD card.

If the ID exists, then it also reads the user’s name, which is stored next to the ID. After that the magnetic lock opens, the “Welcome” message and the user’s name appear on the display, the green enable light goes on, and an audible beep sounds.

If the ID of the tag does not exist between the IDs, the software does not open the door, the “Unknow user” “Access denied!” appears on the display, a red disable light goes on, and an audible beep sounds. After two seconds the system goes back to stand-by state.

b) “Administrator mode”

This mode allows us to register a new user and delete an existing user.

To do so, a PC keyboard must be connected to the USB port of the indoor unit.

After pressing the [F1] key, the system administrator prompts you to log in. After entering the admin password, the main menu appears.

The [F2] key allows us to start registering a new user. To do this, first enter the user name and then hold the RFID tag close to the reader. Successful registration is indicated by a short green light and beep, and the “Successful registration” message appears on the display.

The [F3] key allows us to start deleting an existing user. To do this, just enter the user name. Successful deletion is indicated by a short green light and beep, and “Successful deletion” message appears on the display.

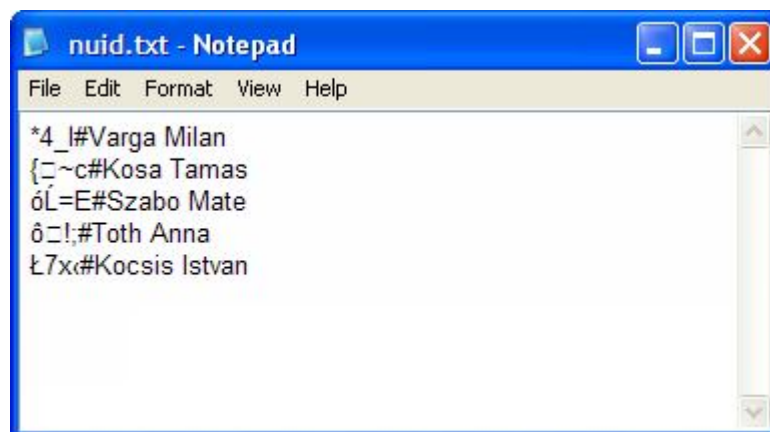
Unsuccessful registration or deletion are indicated by a short red light and beep, and the display shows the cause of the failure (e.g. ID of the tag is already registered, username does not exist, etc.)

The [ESC] key allows us to exit any of the menu states. If we exit from the main menu, the “Menu closed.” message appears on the display, and the system goes back to stand-by state.

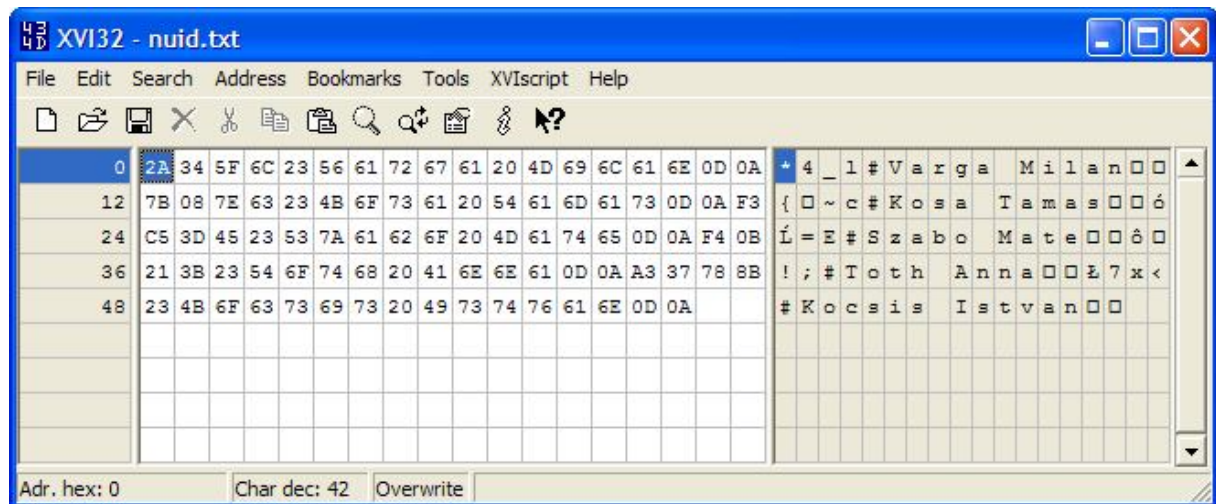
5.3. Format of the database file

To store the user data and the NUID data we find out a format. This is a simple textfile format. In it we store each coherent data in separated lines. First place there is the NUID, then a separator character '#' follows, and finally the user's name goes. The username has to be less than 16 character. At the end of the line there should be a Return character (CR and LF characters: 0x0D and 0x0A).

This can be seen in the following two figures:



File has been opened in Notepad.



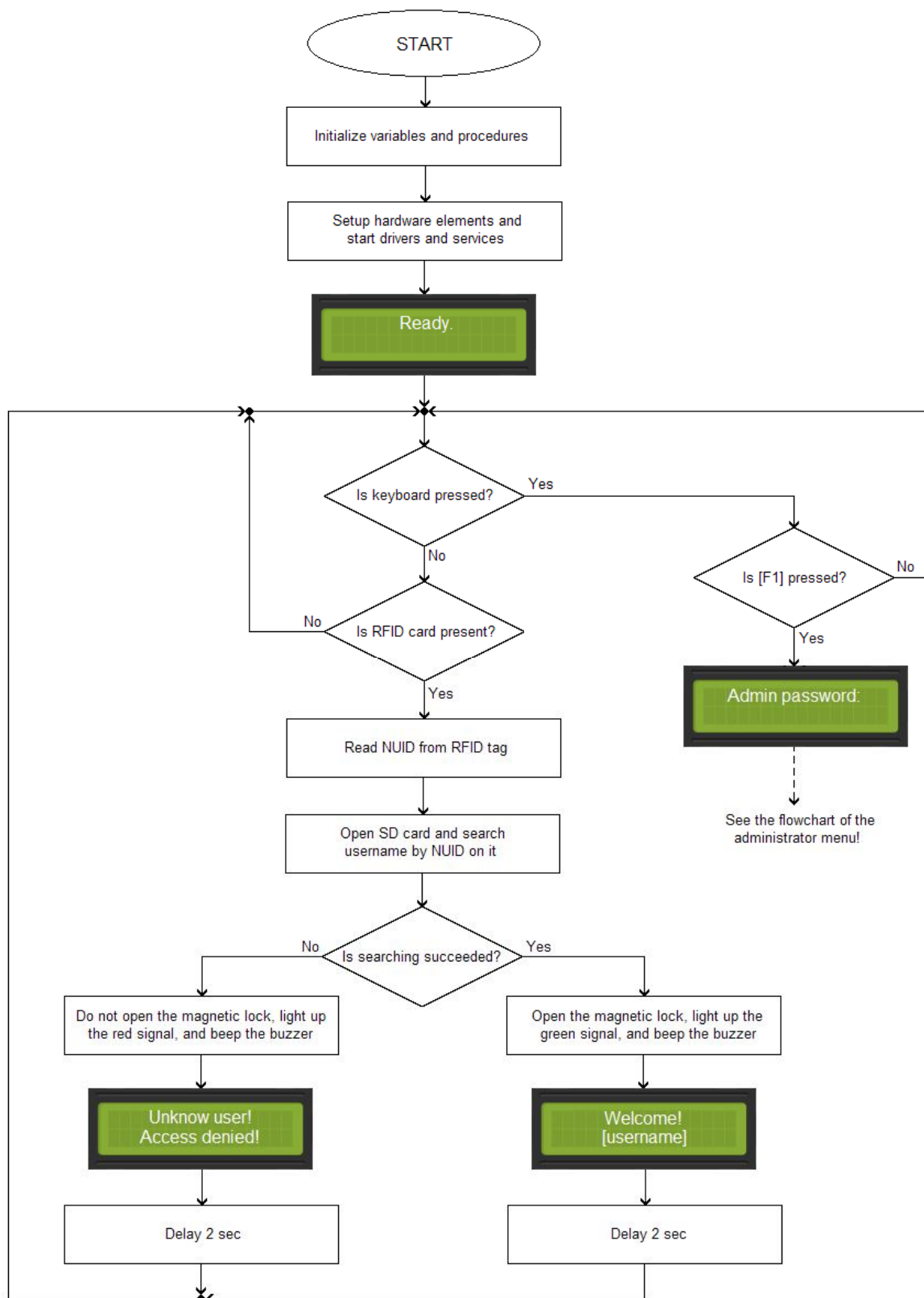
File has been opened in HexEditor.

5.4. Operating process of the software

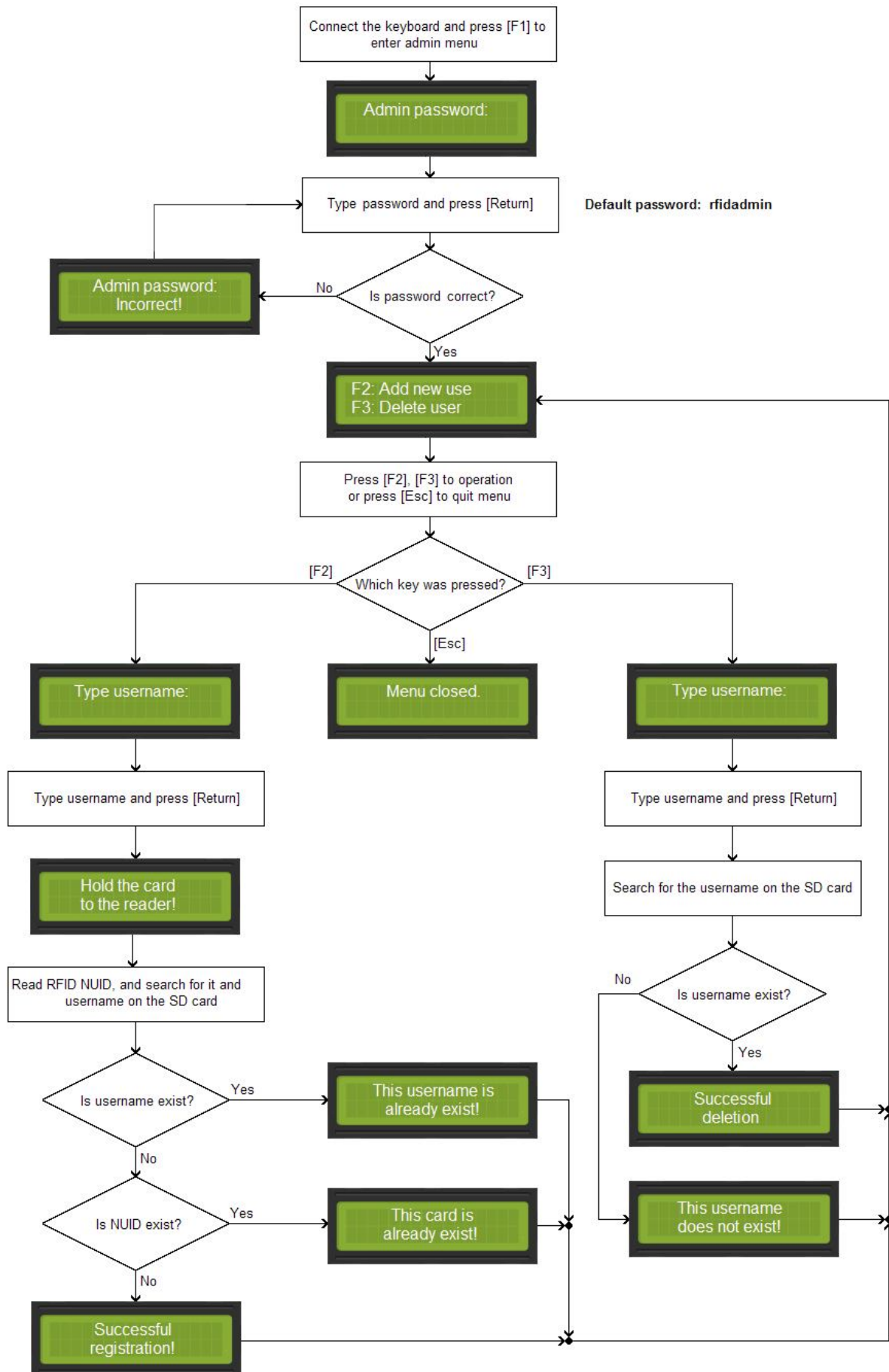
Once the device is turned on, the “setup” part of the software runs, which is responsible for initializing and launching each software components. Then, the “loop” section is executed continuously. Meanwhile doing it, the program monitors the events of the USB keyboard, which are key pressures and RFID events, this is, the activity of the RFID reader. If any event occurs, the procedure will handle and execute it. The structure of the software has been designed in the way to overview and understand it easily. Each of the individual functions is implemented by one method or function. All of these are in the decoding section of the program code. During the programming, we tried to use as few global variables as possible, because the Arduino Uno microcomputer has only 2 kB of dynamic memory. So, the program mainly use local variables. Constant values are placed in the program memory so the entire program occupies only the 68% of the dynamic memory.

The flowchart of the program, the flowchart of the administrator’s menu can be found in the diagrams on the following pages.

5.5. Flowchart of the program



5.6. Flowchart of the administrator menu



6. Appendix

The following directories can be found on the attached CD disk:

\Sketch: directory of the software for Arduino, and the used Arduino libraries

\NUID: directory of the user registration file for SD card, and the description of the *nuid.txt* file format

\Documents: directory of the documents

\Pictures: directory of the pictures of the working of the system

7. Final statement

This project was made under the cooperation of Erasmus+ KA2 P.L.E.L. program between the Hungarian and the Czech partners.

The final hardware and the test software were designed and created in the vocational school: Miskolci SZC Kandó Kálmán Informatikai Szakgimnáziuma, Miskolc, Hungary.

Hungarian team:

Student: Szedlák Botond

Mentor: Kósa Tamás

The proposal hardware and the final software was designed and created in the vocational school: Střední průmyslová škola elektrotechnická, Havířov, Czech Republic.

Czech team:

Student: Hynek Tuleja

Mentor: Ing. Petr Madecki

This document consists of 17 pages.



Zespół szkół technicznych, Mikołów, POLAND
Stredná priemyselná škola elektrotechnická, Košice, SLOVAKIA

Robot for painting Easter eggs



Co-funded by the
Erasmus+ Programme
of the European Union

Student: Mariusz Wróbel
Teacher: Artur Opoka

Student: Adam Škrip
Teacher: Jozef Gmiter

Content

1. Introduction	3
2. Theoretical part	4
2.1 Market analysis	4
2.2 Technical analysis	5
2.3 Used technologies	8
2.4 Financial analysis	10
3 Practical part	10
3.1 Electronics	10
3.2 Programs	12
3.3 Mechanical	13
.....	15
3.4 Testing	15
4 Conclusion	18

1. Introduction

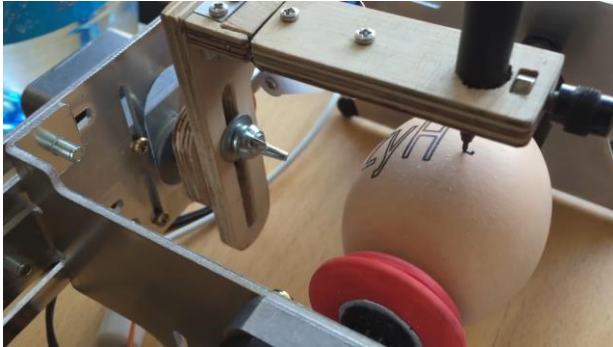
Our project is a robot which paints eggs. This robot can paint eggs, but also anything that has the shape of a sphere for example Christmas baubles, ping pong balls and others. This is not a project that will revolutionize human life. it is a project that will make people's lives more interesting and will help people get used to ubiquitous robots. The industrial revolution, in which people in various positions are gradually replaced by robots, is inevitable That is why it is worth getting the young generation used to the ubiquity of robots through the smallest activity, such as painting eggs for Easter. people who have lived among various robots and machines will be interested in them. this will cause more people to go to study in the direction of robotics and in the end we will have more technicians and robotics engineers who can easily find a job in a company that has robots that need to be repaired, programmed, exchange. I chose this project because I am very interested in technology and robots. I'm fascinated by the world where robots do the smallest activities, that's why I'm learning a profession that will give me a good life in this world that gets more and more like science fiction every year.



<https://iautomatyka.pl/zautomatyzowana-fabryka-samochodow-tesli-seria-filmow/>

2. Theoretical part

Before starting any activities we browsed the internet in search of eggbots. these robots were made of various materials such as wood, metal but our favorite was the one made of plastic, printed in a 3d printer.



Wooden eggbot

https://www.youtube.com/watch?v=ler2DgN_rzA



Metal eggbot

<https://www.youtube.com/watch?v=TrDg2YItQSo>

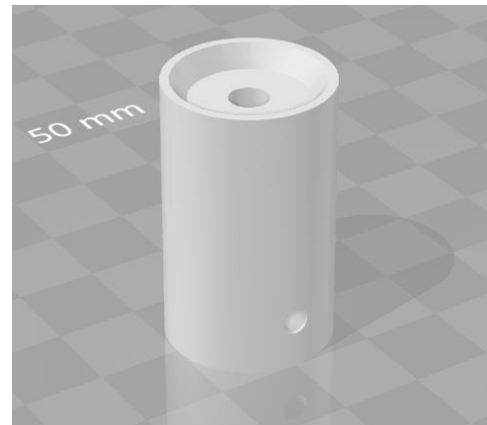
More information on the technologies used can be found in section 2.3

2.1 Market analysis

There are no similar solutions on the market because it is not profitable for companies to create such an expensive device that will be used once a year. Instead of limiting ourselves to painting eggs, we should focus on extending the capabilities of the robot, for example, painting something bigger than eggs or the ability to convert a robot to a plotter. Then people will be more likely to buy such robots because they will be more useful for them than for Easter. for example, our robot has 2 different overlays that make it possible for a robot to draw on Christmas baubles or ping-pong balls.



Christmas bauble overlay



Ping-pong ball overlay

2.2 Technical analysis

The principle of operation is very simple. First of all, you need to draw a pattern in the graphics program that is to be drawn on the egg. It must be a program with the ability to create paths. we chose inkscape. Any fillings must be done line after the line because the robot does not detect the shapes, only the path. Then we have to put the egg in our robot. You should make sure that it is not crooked. If everything is set correctly, turn on the power supply and upload our project to Arduino and the robot automatically begins to draw. After completing the drawing process, turn the power supply off and pull the egg out gently.

Our project consists of Arduino Leonardo which is the "brain" of our robot, 2 stepper motors that are responsible for the movement of x and y and servo which is responsible for the movement of z.

Here are photos that show how to connect individual parts:

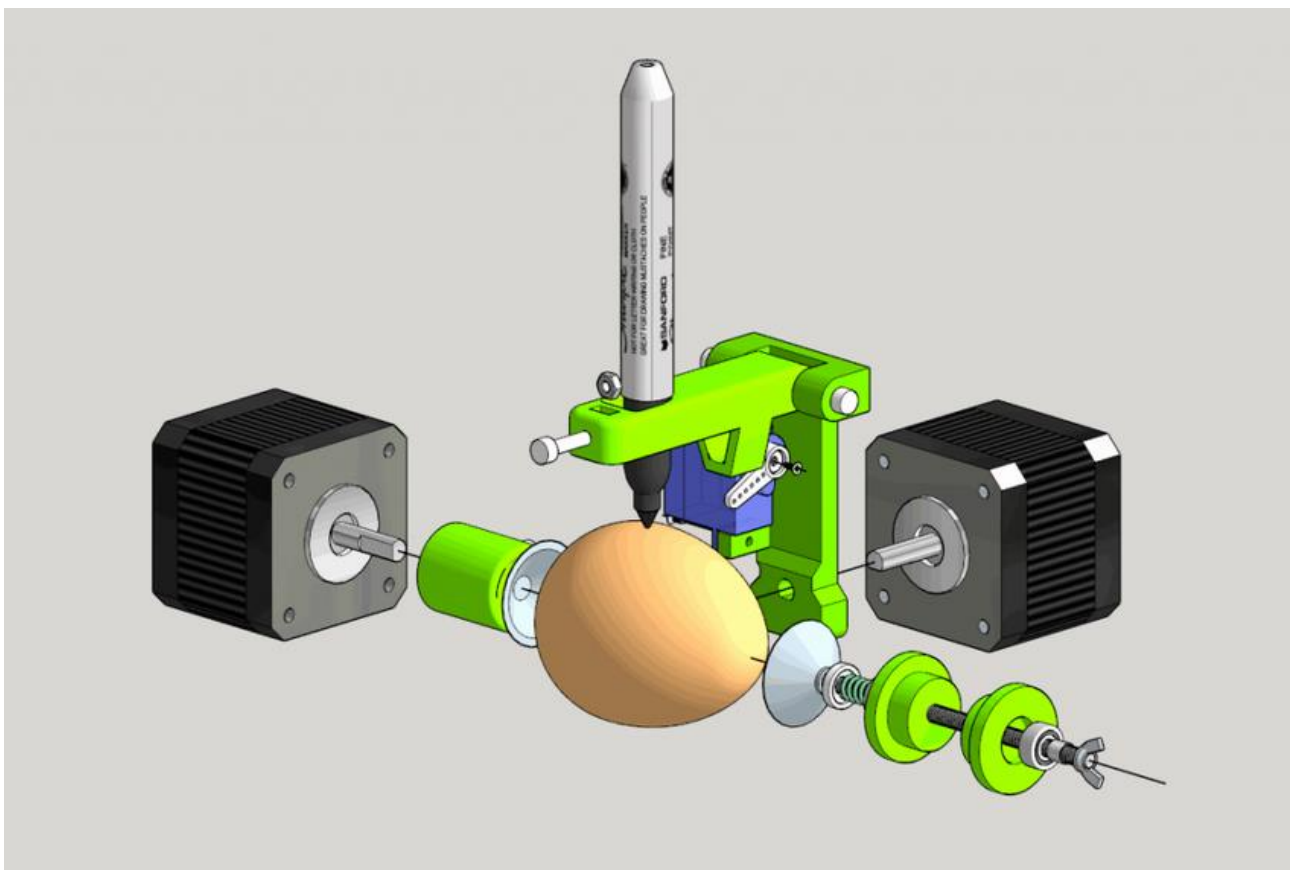


Diagram of elements connection

<https://www.jjrobots.com/sphere-o-bot-assembly-and-user-guide/>

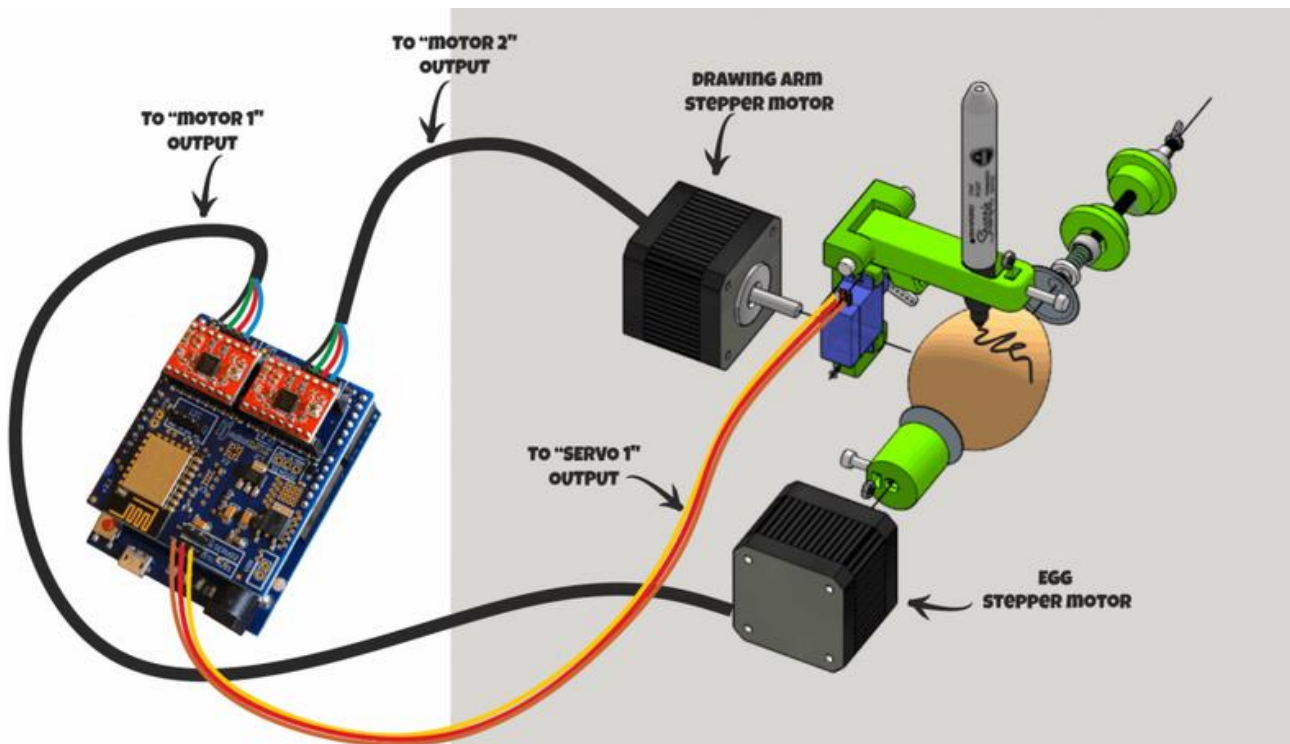
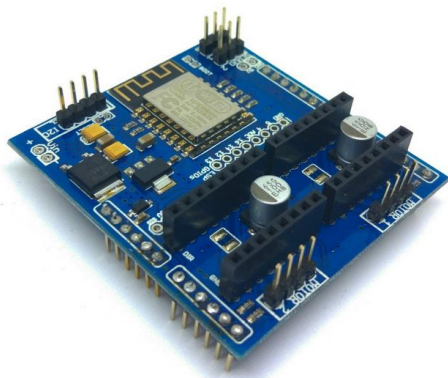


Diagram of connecting the wires

<https://www.jjrobots.com/sphere-o-bot-assembly-and-user-guide/>

They communicate through Arduino Leonardo. For Arduino to be able to send data to stepper motors, it needs an overlay called "brain shield" and drivers.



Brain shield

<https://www.jjrobots.com/product/b-robot-electronic-brain-shield/>



Stepper motor driver

<http://www.alselectro.com/stepper-driver-a4988.html>

To make our robot look better, we have designed special parts, thanks to which the robot has its own unique look



The front beam



Logo of the project

2.3 Used technologies

Parts

High quality NEMA 17 Stepper motors

Step Angle 1.8°
Step Accuracy <5%
Holding Torque 45 N·cm (62oz.in)
Rated Current/phase 1.68A
Phase Resistance 1.65ohms
Voltage 12-24 V
Inductance 3.6mH±20%(1KHz)
Weight 280g

SG90 servo

Parameters for 4.8 V voltage:

Moment: 1.8 kg * cm (0.18 Nm)
Speed: 0.1 s / 60 °

Dimensions: 22 x 11.5 x 27 mm
Weight: 9 g

Arduino Leonardo

Power supply: from 7 V to 12 V
Model: Arduino Leonardo - A000057
Microcontroller: ATmega32u4
The maximum clock frequency is 16 MHz
SRAM memory: 2.5 kB
Flash memory: 32 kB (4 kB reserved for the bootloader)
EEPROM memory: 1 kB
I / O ports: 20
PWM outputs: 7
Number of analog inputs: 12 (A / C converter channels with a resolution of 10 bits)
Serial interfaces: UART, SPI, I2C, USB
External interruptions
LED diode connected to pin 13
MicroUSB socket for programming
DC connection 5.5 x 2.1 mm for power supply

B-robot Brain Shield

Two stepper motor outputs
two servo outputs
I2C communications
Push button (customisable)

sensor port (sonar, IR...)

A4988 Stepper motor drivers

Five different step resolutions: full-step, half-step, quarter-step, eighth-step, and sixteenth-step

Adjustable current control lets you set the maximum current output with a potentiometer, which lets you use voltages above your stepper motor's rated voltage to achieve higher step rates

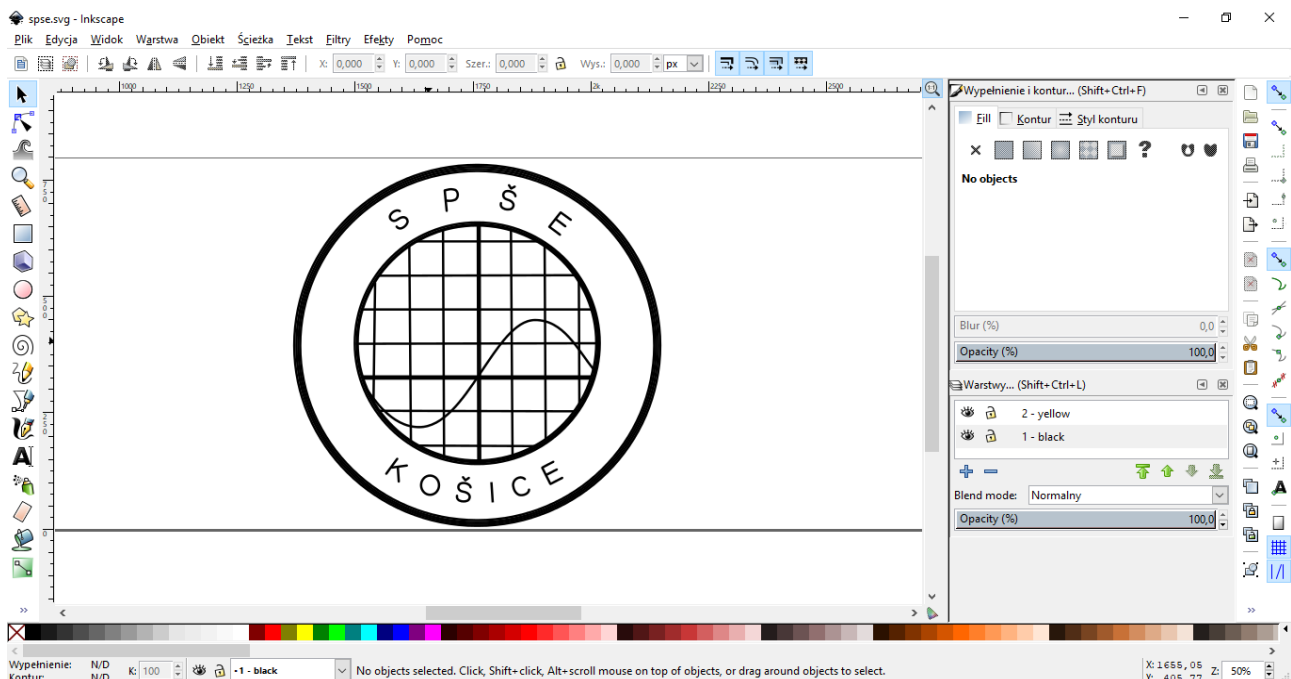
Intelligent chopping control that automatically selects the correct current decay mode (fast decay or slow decay)

Over-temperature thermal shutdown, under-voltage lockout, and crossover-current protection

Short-to-ground and shorted-load protection

Power supply 12v/2A

To print all the parts, a 3D printer was used in Zespół szkół technicznych in Mikołów. we decided to control using Arduino because we both know how to connect wires and how to program in Arduino IDE. We used Arduino Leonardo because it has more inputs than the more popular Arduino Uno and has more SRAM memory that is needed for the robot to work efficiently. The shield used has special pins designed specifically for controllers used in plotter projects. We used NEMA 17 Stepper motors and servo motor for movement. The robot writes on the egg with a 0.5 mm marker. For better effect, use thinner markers (0.25 mm). the program that we used in this project was written in Arduino IDE, but for the correct operation we also used the inkscape plugin. To create patterns we used inkscape program because in order for the robot to draw graphics on the egg you need a program that has the ability to create paths.



One of the graphic designs created by Mariusz Wróbel

2.4 Financial analysis

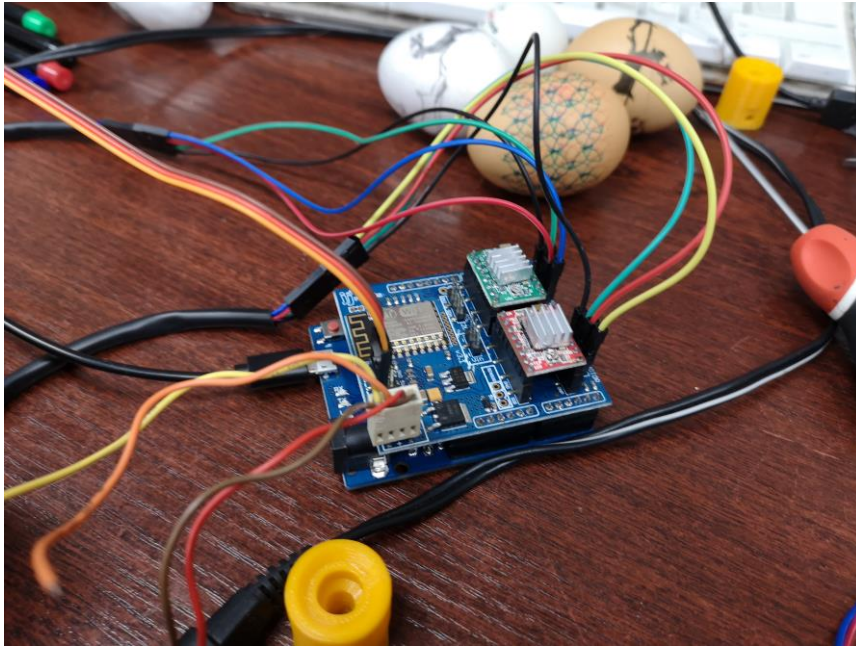
S	W	O	T
<ul style="list-style-type: none">- Easy construction- A good start with Arduino- Versatility- Parts can be reused	<ul style="list-style-type: none">- High price- It can easily be destroyed (base need to by upgrade)- A computer is required for the operation	<ul style="list-style-type: none">- There are no such robots on the market- creating robots with larger sizes	<ul style="list-style-type: none">- It can easily be counterfeited by large companies- Too much price for the consumer

The whole project cost something about 168.63 euros. We did not include screws, cables and other minor things, because these can be found in the workshop. There are no ready solutions for our project on the market. The only thing you can find are prepared kits for self-creation.

3 Practical part

3.1 Electronics

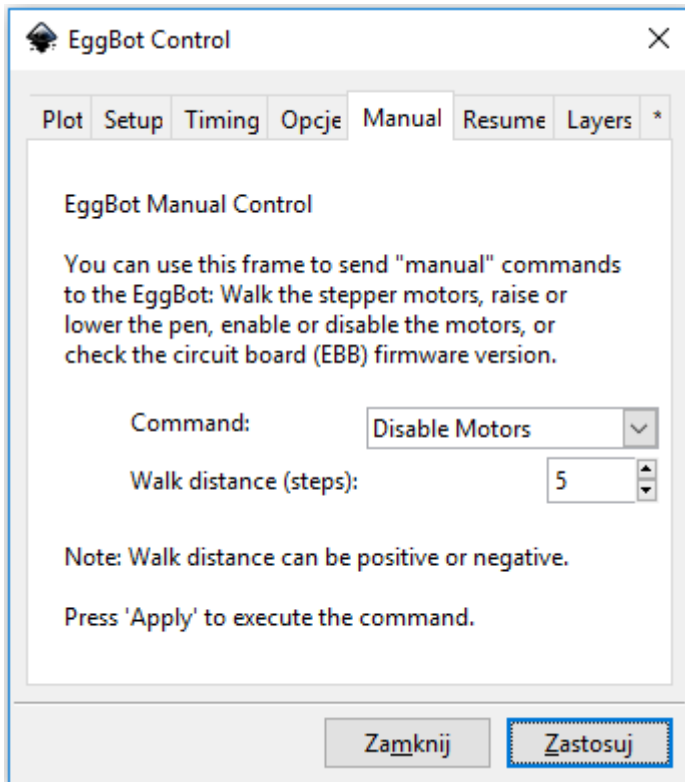
Our robot transforms the input signal (being a path in the inkscape program) through the plugin and Arduino to the data of the Cartesian system, which are our output signal.



Stepper motors are responsible for X and Y movement. We used servo for the z movement, because it is lighter and smaller, which fits perfectly into the robot's arm



All data is displayed on the screen on the fly thanks to the plug-in used. The screen shows errors, the current state of the robot, statistics etc.



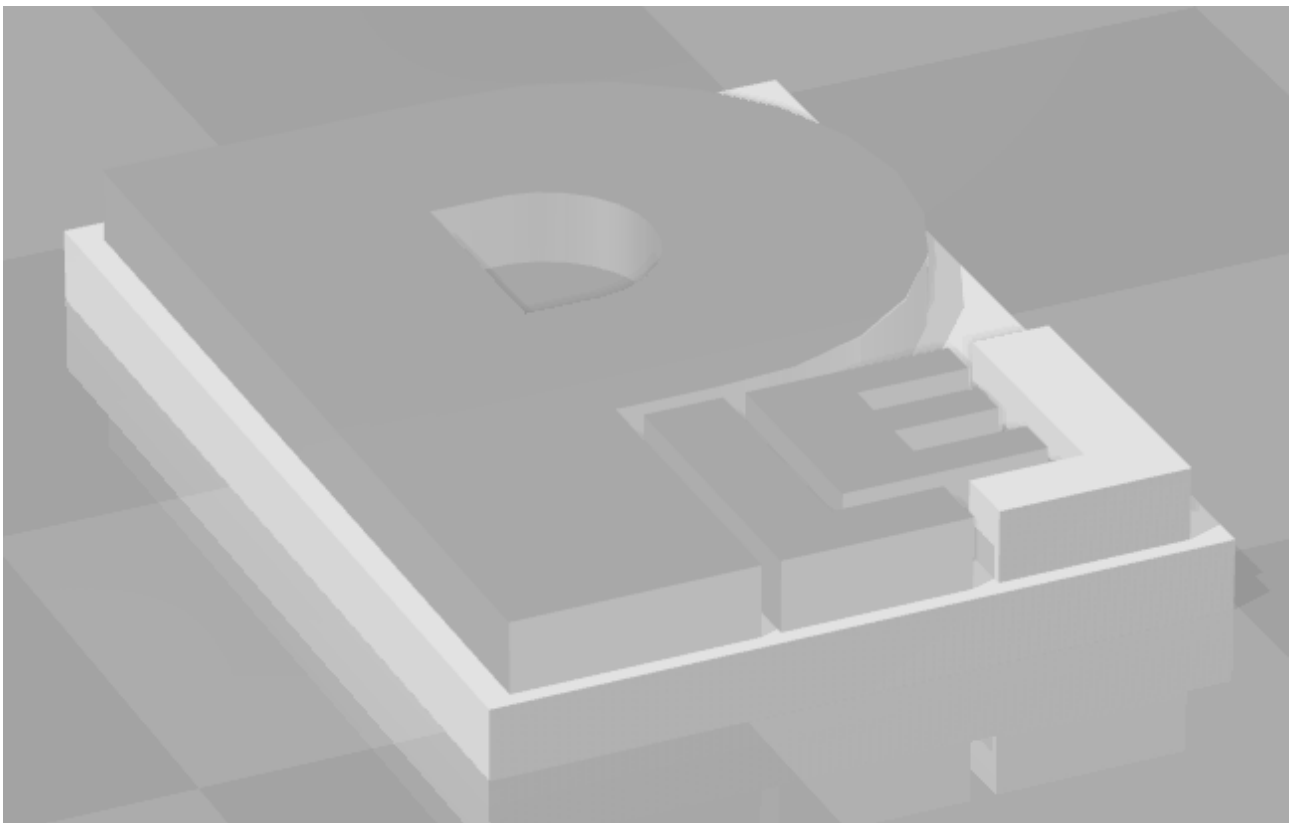
3.2 Programs

In our project, we used the language used in Arduino programming and plugins for the inkscape graphics program. on the computer, we draw a pattern using the path. then the plug exchanges these data for x y z values from which are sent to Arduino Leonardo who uses egg stepper to apply the egg pattern. The only data we have for storage are individual patters, so we store them on the computer

EggbotHolidays	19.11.2018 15:43	Folder plików	
General	19.11.2018 15:43	Folder plików	
Geometric Eggs	19.11.2018 15:43	Folder plików	
Getting Started	19.11.2018 15:43	Folder plików	
cocacola	20.03.2019 11:04	Dokument SVG	13 KB
EggBotTemplate	31.01.2015 07:38	Dokument SVG	2 KB
gountlet	09.03.2019 17:11	Dokument SVG	366 KB
PLEL	07.03.2019 21:17	Dokument SVG	16 KB
spse	21.03.2019 10:42	Dokument SVG	16 KB
thanos an guntlet	10.03.2019 17:48	Dokument SVG	1 033 KB
thanos	18.03.2019 10:22	Dokument SVG	932 KB

3.3 Mechanical

To make our project look more like a school project, Mariusz Wróbel has designed 2 parts that make our robot is different. At the beginning it was to be only the logo of the project.



The first prototype had convex letters because I wanted to paint them with paints. after printing I was not satisfied because the printout was too small. I told you that I would do another part – a beam with the names of our cities.

This is the first beam prototype

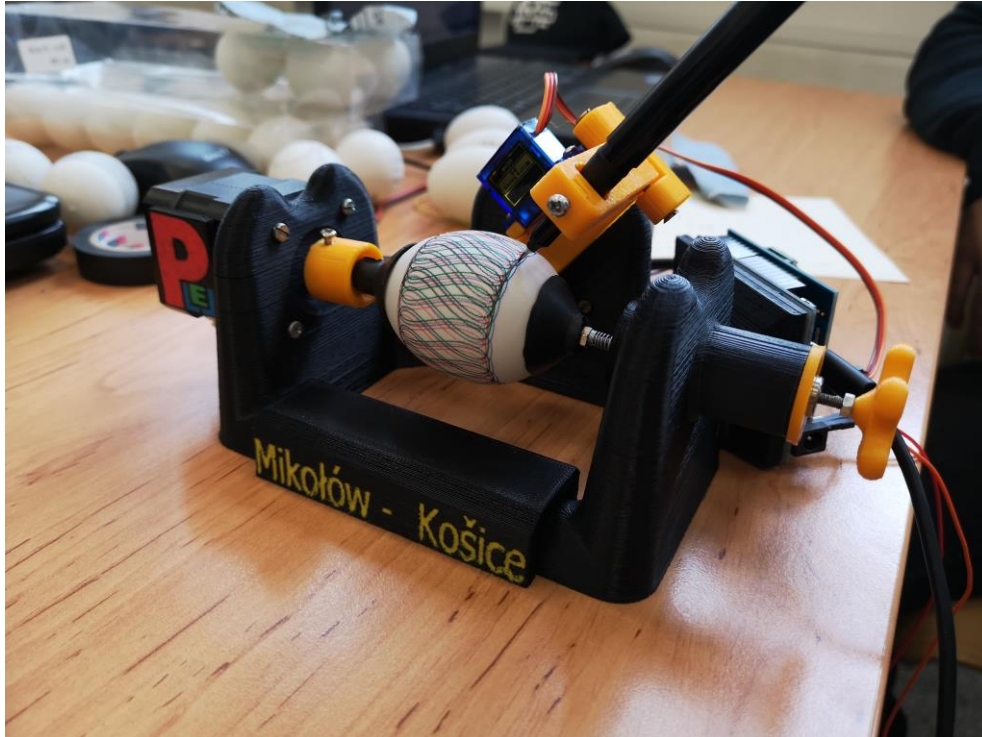


I decided that the printer might not print it, that's why I put a letter in the model. I did the same with the logo.



At the beginning I wanted to fill these models with paint but plasticine turned out to be a better idea.

That's how it looks with the rest of the parts.



3.4 Testing

After submitting our parts into one project which went without a problem, we immediately started painting our first egg. Unfortunately, we wrongly plugged the power supply because instead of attaching it to the shield, we propped it directly to the Arduino. This caused one of the driver to burn out. Thankfully our teacher had a substitute otherwise our project would end on the first day. We have properly plugged in the power supply and we started printing our first pattern which was the inscription "Hello world".



Then we wanted to print a photo. during printing we noticed that the pen was stuck in a styrofoam egg that caused the changes and the image came out crookedly.



The next day we brought ordinary eggs on which it was better.



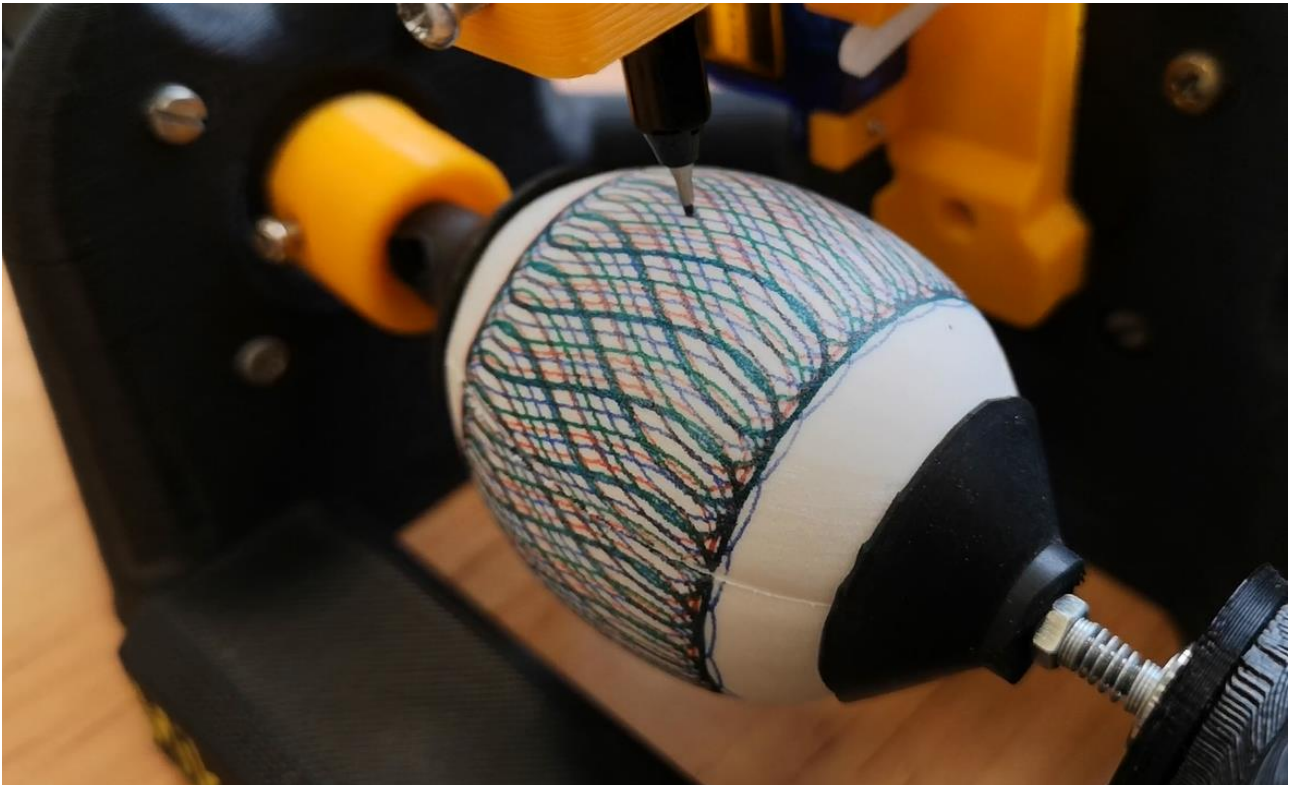
Everything worked, so we thought we would hide all the wires inside the model. during this accidentally we interrupted one of the wires of the stepper motor. When we gave the next painting egg, the robot did not work properly and it took us some time to find the source of the problem.

Here we can see eggs prepared for the show at the end of their stay in Poland.



We continued printing in Slovakia, but on plastic eggs. it was a bad idea because the material from which the eggs were made did not absorb the ink, which made the eggs smudge. It took more time for you to touch new eggs.

Here you see the process of creating a colored egg.



4 Conclusion

Our project was successful because our robot fulfills its function. When we made this project, we learned to draw in 3d, draw vector drawings and collaborate.

We did not have big problems creating this robot. The only problems we came across were easy to fix. As of today, we do not intend to start a business related to our project but maybe one day there will be a moment.

Other applications of our project are, as we said earlier, painting of Christmas balls.

As for international cooperation, We are satisfied with it. we managed to do the majority of the project before the first meeting, thanks to which we had a lot of time to test our robot.



Going to the pros and cons of the project, We are able to name only the pros, because if there were any cons, they were overwhelmed by the pros.

Examples of pros:

- Great accommodation
- Kind people
- Good studio equipment
- Helpful teachers



Stredná priemyselná škola elektrotechnická, Košice, SLOVAKIA
Srednja poklicna in tehniška šola, Murska Sobota, SLOVENIA

ELECTRONIC PRICE TAG



Co-funded by the
Erasmus+ Programme
of the European Union

Contents

1	Introduction	2
2	Theoretical part.....	3
2.1	Market analysis	3
2.2	Technical analysis.....	4
2.3	Used Technologies	5
2.4	Financial Analysis.....	7
3	Practical part	8
3.1	Electronics	8
3.2	Programs	12
3.3	Testing.....	14
4	Conclusion	15
5	References.....	16
6	Attachments	I

1 Introduction

After listing the projects in the international project Erasmus +, in which our school was also involved, I thought that I might be involved as well. Although I wrote this project as the second in the application, but after I got it I found out that I was most interested in it. I got a foreign partner, a student of a Slovenian partner school in Murska Sobota Aleksa Marinic.

We built our project on the ESP-12E microcontroller, which is a copy of the ESP8266 microcontroller and on the HELTEC e-ink 2.9. I have devoted myself to the hardware, design and communication via wireless technology. While Aleks Marinič has devoted himself to designing the format of the data to be sent, the communication protocol and the creation of a website through which data can be changed on the electronic price list.

We used foreign online stores to buy, especially the best known E-bay or AlieExpress.

Initially, we had minor problems with starting a microcomputer, but everything was solved, so we started working on price tag design, communications and later on the website.

We would like to thank p. Ing. Michal Copka, as well as foreign teachers, who acted as a consultant for our project and many times supplied us with many great ideas and motivation.

2 Theoretical part

Perhaps the most common is nowadays used paper price tags, which have to be constantly printed and rewritten, or overlapped with something. But the time goes forward and in this way some trades can be upgraded for a relatively small amount. The electronic price tag is cheap, economical, energy-free, so unless data changes, requiring some minimal energy. Price tag maintenance is very easy and the easiest way is to charge the batteries or replace them.

2.1 Market analysis

Our electronic price tag is similar to electronic reader kindle. The electronic price tag needs energy only to overwrite the display. The display is energy independent. The entire circuit is powered from the lipo battery.

Display is black and white, works only on 2 colors. E Ink is the inventor of several novel types of electrophoretic ink, often called electronic ink. When laminated to a plastic film, and then adhered to electronics, it creates an Electronic Paper Display (EPD). Although futuristic-sounding, electronic ink is actually a straightforward fusion of chemistry, physics and electronics. It's so much like paper, it utilizes the same pigments used in the printing industry today.

ESP8266 is a 3V WiFi module very popular for its Internet of Things applications. ESP8266 maximum working Voltage is 3.6V and its very important to note. You must know how to power it, how to serial-connect it with Arduino safely, how to ping and many other things.

2.2 Technical analysis

The connection between esp8266 and the display is as follows:

3V3 – VCC

GND – GND

D7 – SDI

D5 – CLK

D4 – CS

D3 – DC

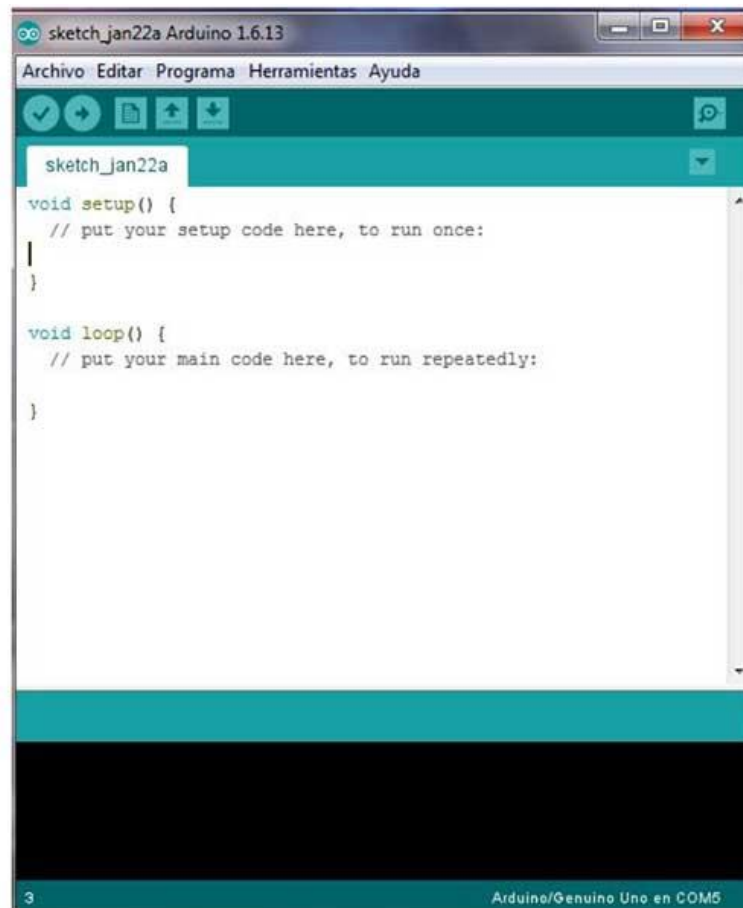
D1 – BUSY

RST – D0

We need to connect RST pin with pin D0, because we want to save energy. With this connection we can take our esp8266 to the deep sleep mode. We may also connect the battery to pin Vin and pin GND.

2.3 Used Technologies

The Arduino IDE is an integrated development environment (IDE). It is composed of tools that help us to program the Arduino. The most striking feature is a white text box with a text editor. There are also various menus such as Arduino selection, serial links that are attached, or a list of sample examples. A common help is a window for displaying serial line messages. Arduino IDE supports various Ardiono boards such as Uno, Nano, Mega, Esplora, Ethernet, Fio, Pro, Mini ...



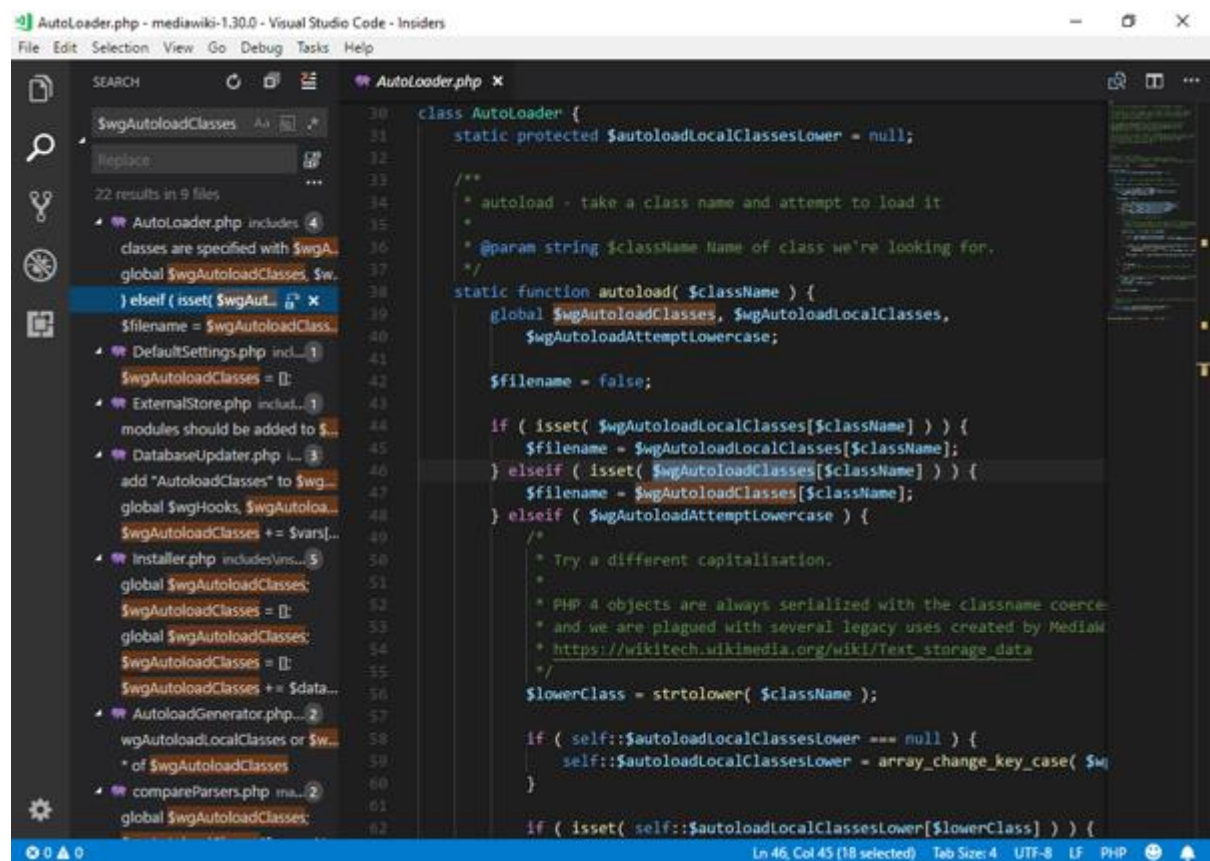
Pic. 2.3-1. (Arduino IDE)

Universal programming languages are C and C ++. The program also includes some built-in libraries such as EEPROM, Firmata, GSM, Servo, TFT and WiFi.

Microsoft Studio Code is an editor created by Microsoft for Windows, Linux and macOS. It includes debugging, syntax highlighting, intelligent code completion ... It can also be set to be as user friendly as possible.

advantages:

- is poor
- open source and free license
- simplicity
- freedom
- community
- potential



Pic. 2.3-2. (Microsoft Studio Code)

2.4 Financial Analysis

The electronic price tag can be purchased from China for some € 10 approximately. But our market does not offer such a price and there is a problem. Is it better to buy large amounts of such price tags or make them?

For the first time it is a little more expensive to make such a project because one does not know which parts are best for him. We decided to use esp8266 microchip, which costs about 15 €. We also used an eink display whose price depends on quality. If one chooses a cheaper one, the colors will be weaker. The more expensive picture is the better picture. The price is around 20 €. The battery is the cheapest of all. It cost about 5 €.

3 Practical part

3.1 Electronics

NodeMCU is an open-source platform that allows you to create interactive projects, especially in the field of automation, or other projects to connect to the Internet / other devices.

parameters:

- Processor model: Xtensa LX106
- Processor frequency: 80 MHz
- Number of processor cores: 1x
- Memory capacity: 4MB

- 802.11n Wireless LAN

outputs:

- micro USB
- 11 GPIO pins

The heart of the NodeMcu includes a single-core 32-bit Xtensa LX106 processor with RISC instruction set and 80MHz. It can be overclocked to 160MHz.

Up to 1MB of compiled code can be uploaded to the board and it is equipped with MB flash memory. Offers up to 11 GPIO pins. While all but GPIO16 can be PWM modulated. The output current of the GPIO pins is 12 mA at 3.3V. With a long load of 5V, the ESP8266 chip overheats and is destroyed. There is also one analog pin on the board. By default, ESP8266 can use a pin between 0-1V. Thanks to the NodeMCU components it operates in the range of 0-3.3V. ESP8266 implements 802.11 wifi at 2.4 GHz in b / g / n bands, with the best gain of 25dB in b.

NodeMCU supports interfaces such as OneWire, I2C, SPI, UART, ADC. The board can create HTTP, HTTPS request POST and GET method and has a fully integrated MQTT protocol. It also supports WEP, WPA / WPA2 PSK encryption. The board can also be programmed in Micropython, Javascript, Lispe and others. It can also be programmed via OTA (On the air), so wirelessly, without having to connect to the computer.



Pic. 3.1-1. (Esp 8266)



Pic. 3.1-2. (Esp 8266)

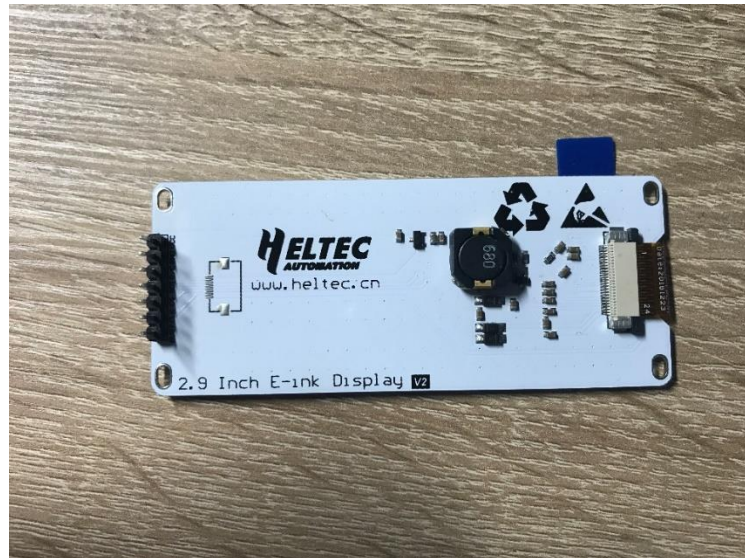
We chose the HELTEC E-ink 2.9-inch screen. E-ink or even electronic paper is a flat display unit that reflects light as normal paper, capable of storing text and images permanently without power consumption.

parameters:

- Size 296x128
- Black and white
- VCC, GND, D / C, SDI, CS, CLK, BUSY inputs



Pic. 3.1-3. (Display)



Pic. 3.1-4. (Display)

We also used a CELLEVIA BATTERIES LP443450 3.7V 750mAh battery. And to it a charging board that can be charged through a micro USB charger. The LIPO battery is a lithium-polymer battery that is used in almost any electronics. They have features such as:

- Lightweight
- Relatively high capacity
- Minimal self-discharge
- Great performance
- It is relatively small



Pic. 3.1-5. (Battery)

3.2 Programs

The website is currently on free hosting. The website is still only for functionality. In the future, we would like to make the website so that each user has their rights. To be one main user who could create accounts and those accounts to be able to change data on the electronic price list. It will all work by signing up.

Here we have some examples from code:

```
unsigned char image[1024];
Paint paint(image, 0, 0);
Paint paint1(image, 0, 0); // width shor
Paint paint2(image, 0, 0);
Paint paint3(image, 0, 0);
Paint paint4(image, 0, 0);
Paint paint5(image, 0, 0);
Paint paint6(image, 0, 0);
Paint paint7(image, 0, 0);
Paint paint8(image, 0, 0);
Paint paint9(image, 0, 0);
Paint paint10(image, 0, 0);
Paint paint11(image, 0, 0);
Paint paint12(image, 0, 0);
Paint paint13(image, 0, 0);
Paint paint14(image, 0, 0);
Epd epd;

// Initialisation for variables
char price[100];
char date[100];
char tax[100];
char price_without_tax[100];
char quantity[100];
char price_quantity[100];
char name_good[100];
char id_number[100];
char priceWithoutTax[100];
```

In this section, we define individual code variables.

```

client.print(String("GET ") + url + " HTTP/1.1\r\n" +
              "Host: " + host + "\r\n" +
              "Connection: close\r\n\r\n");
unsigned long timeout = millis();
while (client.available() == 0) {
  if (millis() - timeout > 5000) {
    Serial.println(">>> Client Timeout !");
    client.stop();
    return;
  }
}
char x[100];
// Read all the lines of the reply from server and pr:
while (client.available()) {
  String line = client.readStringUntil('\r');
  Serial.print(line);
  line.toCharArray(x, 100);
  for(int i = 0; i < 99; i++) {
    x[i] = x[i + 1];
  }
}
}

```

This is communication with server and answer from server.

3.3 Testing

Since we are both from different countries and schools. So we initially worked on the project separately. First, we started to create the design and layout of the elements on the electronic price list. Since we initially had a problem linking ESP8266 with our paper screen, we created the code through Arduino Uno. Over time, we found out how to connect ESP8266 with a paper screen. It was enough to define the ports in the code. The connection to the screen is as follows: Pin 3V3 is connected to VCC pin, pin GND is connected to GND, pin D7 to pin SDI, pin D5 to pin CLK, pin D4 to pin CS, pin D3 to pin DC and pin D1 to pin BUSY . After the success we decided to continue working on WiFi connection. We started by creating a simple website that we created with Microsoft Studio Code. After creation, we focused on communicating this website with the ESP8266 microcomputer. We initialized individual variables in the code for the ESP8266 microcomputer. We have created a program in PHP where we have guided individual variables from the website into the memory of our microcomputer ESP8266. We have created a code that records the individual variables and of course the connection of the ESP8266 microcomputer to available WiFi. All these aspects are being watched by the monitor series in the Arduino IDE development environment. In the beginning we had a problem with the fact that the individual data was recorded too long. We later fixed this issue. We have solved this by sending individual variables at once and then splitting them in the program. We also resolved the automatic recalculation without VAT. After entering the amount and VAT, the price excluding VAT is recalculated. All these PHP programs are uploaded to the servers on which the website is connected and to which the microcomputer is connected by WiFi after the data change.

4 Conclusion

By implementing our project we have brought an interesting solution for the modernization of shops. We have shown that this is a relatively simple and cheap way to do it. This solution is almost energy-independent, mobile, easy to use. To operate, it needs to have its own WiFi network, which is used almost everywhere today. We think this or similar solution will be commonly used in stores in the future.

We can see the potential for improvement in a color screen from a better manufacturer. Maybe in speeding up or possibly some interesting characters on the price tag. The website could also be able to award prizes to authorized employees.

Perhaps the biggest problem was the constant redrawing of the screen, which was solved by the condition that the screen should only be redrawn if a value is changed. If no value is changed, the screen will not be redrawn. We also had a problem getting involved. But this problem has been solved over time. There was also a problem with the missing € sign in the paper screen library. However, we see the biggest "pluses" that we have experience in designing and implementing a project, solving problems and trying to remove them.

5 References

[1 Description ESP8266]

<https://www.espressif.com/en/products/hardware/esp8266ex/overview>

[2 Description e-ink obrazovky] <https://www.eink.com/>

https://www.waveshare.com/wiki/2.9inch_e-Paper_Module

[3 How to connect esp8266 with e-ink]

<https://github.com/soonuse/epd-library-arduino>

[4 Arduino IDE]

https://en.wikipedia.org/wiki/Arduino_IDE

[5 Microsoft Studio Code]

https://en.wikipedia.org/wiki/Visual_Studio_Code

6 Attachments

Edit data on price tag

Choose price tag ▾

Name	<input type="text" value="Enter name of product"/>
Price	<input type="text" value="Enter price of product"/>
Tax	<input type="text" value="Enter tax of product"/>
Quantity	<input type="text" value="Enter quantity of product"/>
Price of quantity	<input type="text" value="Enter price of quantity of product"/>
Date	<input type="text" value="dd.mm.rrrr"/>
Barcode	<input type="text" value="Enter barcode of product"/>

Submit

```
#include <SPI.h>
#include <epd2in9.h>
#include <epdpaint.h>
#include "imagedata.h"
#define COLORED      0
#define UNCOLORED    1
#define BUSY_PIN      5 // D1
#define RST_PIN       4 // D2
#define DC_PIN        0 // D3
#define CS_PIN        2 // D4

// Include library for WIFI
#include <ESP8266WiFi.h>

#include <EEPROM.h>

// SSID and password for WIFI connection
const char* ssid      = "Roman-WiFi";
const char* password = "sroman84";

const char* host = "www.eptag.co.nf";
```

```

unsigned char image[1024];
Paint paint(image, 0, 0);
Paint paint1(image, 0, 0); // width shou
Paint paint2(image, 0, 0);
Paint paint3(image, 0, 0);
Paint paint4(image, 0, 0);
Paint paint5(image, 0, 0);
Paint paint6(image, 0, 0);
Paint paint7(image, 0, 0);
Paint paint8(image, 0, 0);
Paint paint9(image, 0, 0);
Paint paint10(image, 0, 0);
Paint paint11(image, 0, 0);
Paint paint12(image, 0, 0);
Paint paint13(image, 0, 0);
Paint paint14(image, 0, 0);
Epd epd;

// Initialisation for variables
char price[100];
char date[100];
char tax[100];
char price_without_tax[100];
char quantity[100];
char price_quantity[100];
char name_good[100];
char id_number[100];
char priceWithoutTax[100];

```

```

char ID[1];
char ID_old[1];

int counter = 0;
int counter2 = 0;
int ending = 0;

void setup() {
    Serial.begin(9600);
    EEPROM.begin(512);
    delay(10);
    ID_old[0] = '#';
}

void loop() {
    // We start by connecting to a WiFi network
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    /* Explicitly set the ESP8266 to be a WiFi-client
       would try to act as both a client and an access point
       network-issues with your other WiFi-devices (e.g.
       Arduino WiFi Shield)
    */
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    // we try to connect to WiFi
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());

    delay(5000);

    Serial.println(" ");
    Serial.print("connecting to ");
    Serial.println(host);
    // Use WiFiClient class to create TCP connections
    WiFiClient client;
    const int httpPort = 80;
    if (!client.connect(host, httpPort)) {
        Serial.println("connection failed");
        return;
    }

    // We now create a URI for the request
    String url = "/eptag/eptag1.txt";

    Serial.print("Requesting URL: ");
    Serial.println(url);

```

```

client.print(String("GET ") + url + " HTTP/1.1\r\n" +
              "Host: " + host + "\r\n" +
              "Connection: close\r\n\r\n");
unsigned long timeout = millis();
while (client.available() == 0) {
  if (millis() - timeout > 5000) {
    Serial.println(">>> Client Timeout !");
    client.stop();
    return;
  }
}
char x[100];
// Read all the lines of the reply from server and pr:
while (client.available()) {
  String line = client.readStringUntil('\r');
  Serial.print(line);
  line.toCharArray(x, 100);
  for(int i = 0; i < 99; i++) {
    x[i] = x[i + 1];
  }
}

counter = 0;
counter2 = 0;
ending = 0;

ID[0] = x[0];
ID_old[0] = EEPROM.read(0);
counter = (counter + 2);

for(int y = 0; y < 100; y++) {
  if(x[counter] == '#') {
    counter++;
    break;
  }
  tax[y] = x[counter];
  counter++;
}

for(int y = 0; y < 100; y++) {
  if(x[counter] == '#') {
    counter++;
    break;
  }
  quantity[y] = x[counter];
  counter++;
}

for(int y = 0; y < 100; y++) {
  if(x[counter] == '#') {
    counter++;
    break;
  }
}

```

```

Serial.println();
Serial.println("closing connection");

if (epd.Init(lut_full_update) != 0) {
    Serial.print("e-Paper init failed");
    return;
}

// We checked if new ID is equal to the old ID (ID_old)
if(ID[0] != ID_old[0]) {
    Serial.println("Refreshing screen.");
    //ID_old[0] = ID[0];

    EEPROM.write(0, ID[0]);
    EEPROM.commit();

    epd.ClearFrameMemory(0xFF); // bit set = white, bit reset = black  resetuje obrazovl
    epd.DisplayFrame();
    epd.ClearFrameMemory(0xFF); // bit set = white, bit reset = black
    epd.DisplayFrame();

    //////////////////////////////////////

    paint.SetRotate(ROTATE_90);
    paint.SetWidth(16);
    paint.SetHeight(120);

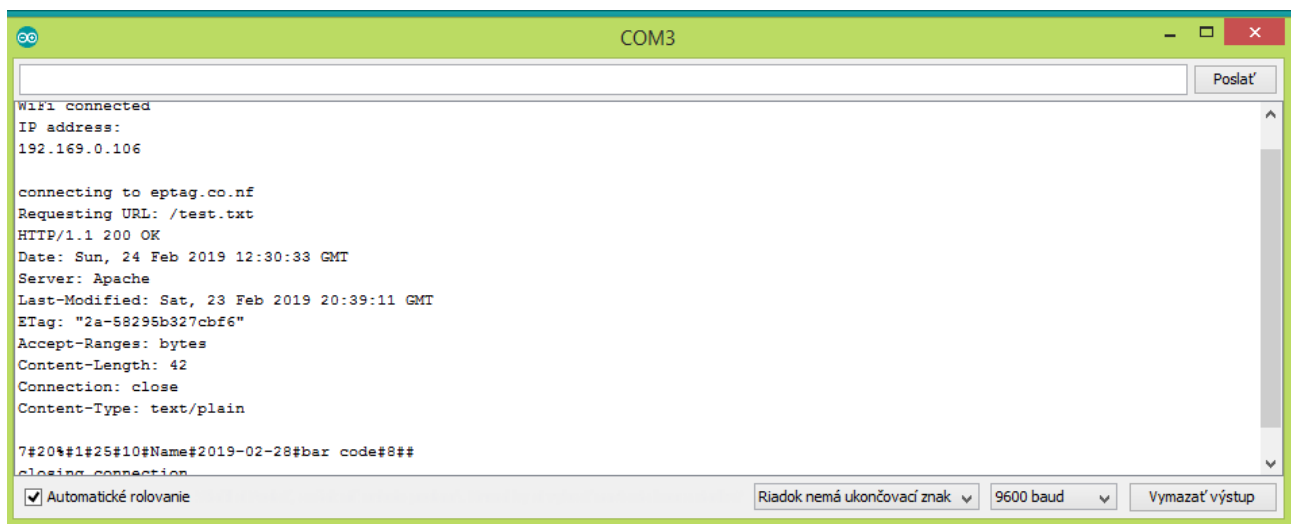
    paint.Clear(COLORED);
    paint.DrawStringAt(0, 4, "Price without tax", sFont12, UNCOLORED);
    epd.SetFrameMemory(paint.GetImage(), 60, 90, paint.GetWidth(), paint.GetHeight());
C

    else {
        Serial.println("Screen is not refreshing.");
    }

    for(int i = 0; i < 100; i++) {
        price[i] = ' ';
        date[i] = ' ';
        tax[i] = ' ';
        price_without_tax[i] = ' ';
        quantity[i] = ' ';
        price_quantity[i] = ' ';
        name_good[i] = ' ';
        id_number[i] = ' ';
        priceWithoutTax[i] = ' ';
    }

    Serial.println("DeepSleep - START");
    ESP.deepSleep(30e6);
    Serial.println("DeepSleep - END");
}

```





Střední průmyslová škola elektrotechnická, Havířov, CZECHIA
Srednja poklicna in tehniška šola, Murska Sobota, SLOVENIA

GREENHOUSE



Co-funded by the
Erasmus+ Programme
of the European Union

Student: Alex Kolebacz
Tutor: Ing. Václav Sedlák

Student: Marko Pintarič
Tutor: Darko Oskomič

Contents

1. Introduction	1
2. Theoretical Part.....	2
2.1 Market Analysis.....	2
2.2 Technical Analysis	2
2.3 Used Technologies	3
2.4 Financial Analysis	8
3. Practical Part.....	10
3.1 Electronics	11
3.2 Programs	13
3.3 Mechanical.....	15
3.4 Testing	15
4. Conclusion	17

1 Introduction

The purpose of the project is for students to get to know what's happening in the real world, which means, they can transfer things that are happening in large complexes to a model that they produce.

Throughout the whole project we tried to not buy everything, but we used materials that were already in use and we re-used them through the project. In the whole project, we consider that waste is a major problem on.

I have chosen this project because I have greenhouse in the garden, so I know which work is behind growing the plants. If it's dry season, the greenhouse must be watered every day, at low temperatures the greenhouse would have to be heated to make your plants flourish and for vegetables to enjoy them.

We tried to solve regulating the basic requirements needed for plants maintenance like plants irrigation, ventilation, heating or cooling the air in the greenhouse. To have enough water in the water tank the water level there will be controlled and for the lack of water it will be pumped from the external tank. Only with these changes the customer's work will be decrease a lot.

Next thing to regulate is light amount there, when is necessary to shine during the day, when it isn't and when to turn lights in greenhouse off to have a rest in the night.

To save energy, solar panels can be added to charge the battery from which the control units, sensors, etc. are powered. At the same time, for the protection of the environment, the plants will be watered mostly with rainwater, which will not only reduce the rare drinking water usage, but will also be better for them because, unlike drinking water, it contains no chlorine.

A fertilizer dispenser may be added for customers who will use greenhouse mostly for growing vegetables. This could be triggered either by pressing a button, or triggered by a timer set by the customer as needed.

2 Theoretical part

Our task was to create a functional greenhouse model. We undertook these tasks by first visiting a normal greenhouse in our area. There we got useful information about what information is needed for the operation of such a system.

After talking to the owner of the greenhouse, we found that everything could not be done in our model. Therefore, in agreement with the Czechs, we decided to use the following data on our model:

- measurement of room temperature and humidity
- measurement of soil moisture
- watering and measuring the height of the water in the reservoir
- heating - opening and closing windows
- alarm when value is exceeded
- Wi-Fi connection

2.1 Market analysis

My school is located in the agricultural area of Slovenia. Therefore, there are quite a few greenhouses in our area that deal with commercial vegetable production.

In the Moravian-Silesian Region, greenhouses are only private in gardens or smaller horticultural firms. We had greenhouses in Havířov, where flowers were grown at the Havířov event in flowers, but today they are replaced by the Globus hypermarket.

2.2 Technical analysis

This project is divided into five main parts: measurement, control of measured values, regulation, listing and settings for changing intervals for given quantities, which are soil moisture, temperature and air humidity.

Since this program is designed for a reduced model of a greenhouse, we have solved it by averaging the values of the quantity obtained from the sensors and then comparing them with the interval for the given output, by means of which we control its regulation. Here are the outputs, the quantity that the given output controls, and then its interval, which will be used as the default, with the variable names for its limits to control it:

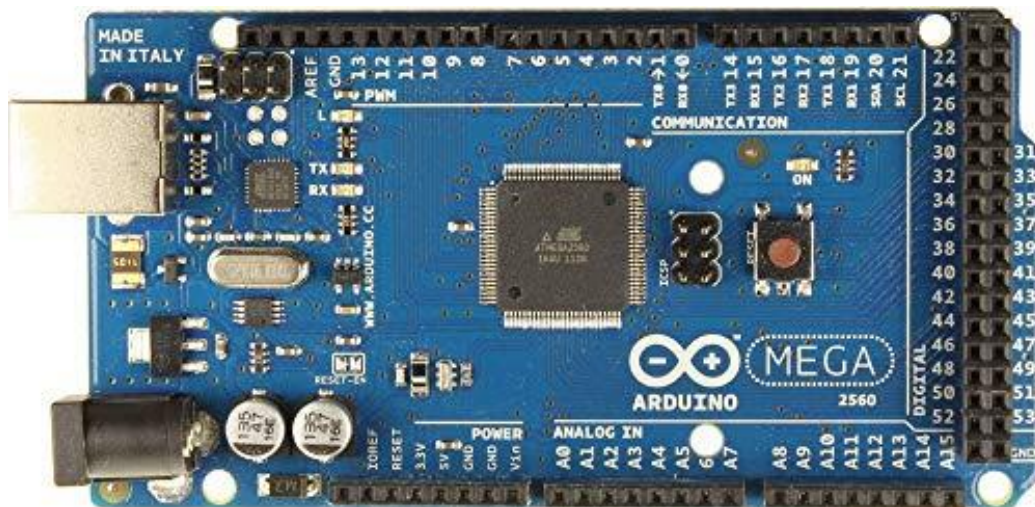
Values regulations							
Value		Soil humidity		Air humidity		Air temperature	
Interval	Extreme	Min.	Max.	Min.	Max.	Min.	Max.
	Set value	20%	40%	40%	50%	27°C	28°C
Output		irrigation	-	-	ventilation	heating	cooling

Regulations

A specific description of the individual regulations can be found in the *Used Technology* section.

2.3 Used Technologies

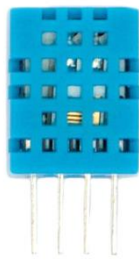
I used a single board computer Arduino Mega 2560, which I programmed in Arduino IDE environment, to put the whole project into operation.



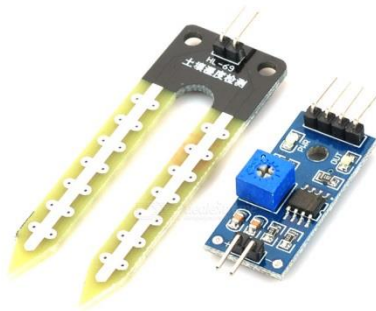
Arduino Mega 2560

I used the following types of sensors to measure the quantities:

- 2pcs DHT11 - temperature and humidity sensor
- 4pcs Soil sensor with LM393 chip - for soil moisture measurement
- 1pc HY-SRF05 - ultrasonic sensor for monitoring the amount of water in the tank by scanning



DHT sensor



Soil sensor with LM393 chip



Ultrasonic sensor

The DHT sensor is used to measure air temperature and humidity. It has one wire for sending temperature and humidity data to the Arduino a two pins to power the sensor. The fourth is not used. The data pin is connected to the Arduino as a digital input. If one of the pins is not connected to the Arduino, it returns *nan*, which is useful for checking if it is connected, or not.

Soil sensor only measures soil moisture. In addition to power, it has two pins for data. One sends data in analogue form, while the other sends only logical 1 and 0. With this sensor, it is not possible to determine whether it is connected or not using the measurement function, since analog inputs always have some environment and voltage-dependent value, while in digital it will always show that it is dry.

The last type of sensor is designed to measure distances from 2 to 450 centimeters. It is shown in two versions: with or without output pin. Both types have Echo and Trig pin. By sending a $5\mu\text{s}$ pulse to the Trig pin, the sensor sends an ultrasonic signal and measures how long it returns. The measured time is sent to Arduino using Echo Pin. Subsequently, it must be converted to centimeters by multiplying by 0,01715. This constant is obtained by dividing the sound velocity at 20°C in $\text{cm} / \mu\text{s}$ (which is 0.0343) by two, since the measured pulse also includes the backward path and we would have twice the distance.

I used the following parts to make the controls:

- 1pc 8-channel relay module - for triggering control components
- 2pcs 28BYJ-48 - stepper motors for opening and closing windows



Eight-channel relay module

A relay is a component that switches a high current circuit on a low current circuit. Thereby the circuits are separated and there is no risk of destruction of components in a small circuit.



Stepper motor

A stepper motor synchronous motor that rotates in steps, depending on the number of pole pairs inside the motor. For its proper function we use driver with ULN2003 circuit.

These components serve as a display and control system:

1pc HD44780 - LCD display for displaying values or errors

1pc Membrane keyboard 3x4 - for controlling the greenhouse and setting the interval limits



LCD display 20x4



Numeric keypad

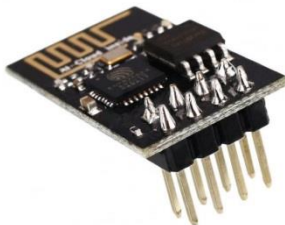
This display has four lines of 20 locations. An I2C interface is soldered from behind, reducing the number of wires to just four, thus making it easier to connect to the Arduino - two pins for 5V power and control signals SDA for data and SCL as a clock signal.

The mechanical numeric keypad we use consists of four rows and three columns. It has seven pins, each of which means a given line or column that serves as the coordinate of a button pressed. When pressed, two pins (one column and the other line in which the button is located) send a signal to the Arduino that a button is pressed, and pressed on the display device based on a given row / column combination number or character.

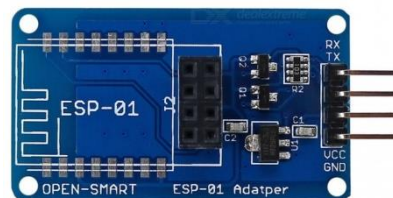
Here are the components needed to connect to Wifi:

1pc ESP8266 - 01 - Wifi module

1pc HW-580 - adapter for WiFi module ESP8266 – 01



Wifi module ESP8266-01



adapter for Wifi module ESP8266-01

This Wifi module is one of the range of ESP8266 modules. When purchasing this module, it already has an AT command program installed, so it can work when connected to Arduino. However, if it is to work independently, a special USB adapter must load the

program, but this will erase the AT command program so that it will no longer be usable. It works in 3.3V logic.

The adapter shown on the right serves to convert the voltage from 3.3V to 5V and vice versa directly for ESP 01, so no circuits with resistors or transistors can be solved.

Here's some information on how we'd like to implement regulation:

1. Soil moisture regulation

It will be controlled by drip irrigation. When the soil moisture reaches a minimum, the solenoid valve opens and the soil begins to irrigate, until it reaches the middle of the interval. At this point, the valve closes.

2. Humidity control

In this case, I use the opening and closing windows that we will solve by two stepper motors rotation, each one attached to one window. To open the window, the motor rotates fifty times 360 ° angle and the same for closing in the opposite direction.

The windows open when the air humidity rises above the upper limit. The windows will close when the humidity drops below to the middle of interval.

3. Air temperature control

Here I use a heating light bulb and a computer fan for cooling. If the temperature exceeds the middle of the interval, the bulb is switched off. When the upper limit is exceeded, the fan starts, which switches off again in the middle of the interval when the temperature drops. If the temperature drops to the lower limit, the bulb turns on and heats the greenhouse again.

2.4 Financial Analysis

SWOT analysis:

- Strong:
- makes the job much easier for the customer
 - there is no need for frequent visits to the greenhouse
 - ecological
 - possibility to monitor current and older measured data on the Internet in graphic presentation
- Weak:
- no fertilizer dispenser
 - our model is powered from the mains, so it must be near a power outlet
 - for real size applications, control types need to be changed for more efficient operation
- Opportunities:
- can be improved, for example, by supplying fertilizer
 - in the case of the use of controls without the need for mains power, it is possible to use a battery that can be charged with solar panels, and the location of the greenhouse will not depend on anything
- Threats:
- if one or more soil sensors are suddenly disconnected, the average soil moisture can be affected and irrigation may be poorly controlled.
 - in the event of disconnection of one or more soil sensors, the greenhouse does not notify the customer in any way, this problem can be detected during continuous monitoring of the values via the Internet or by occasional checking of the sensors by either inserting the sensor into the water or wrapping it in a dry cloth

Costs:

Here is a list of used components for making the model:

	Name	Characteristic	Price (euro)
1	Plexiglas	3mm 1,5m ²	90,00
2	water pump	1000l/h; Claro AC220-240V, 50Hz, 22W	35,00
3	bulb	60W	4,00
4	water tank	1,5l	9,00
5	heating pipes	different	15,00
6	grass container	43x30 cm	11,00
7	watering material	Pipe, steel strip, ...	20,00
8	fan from the computer	DC 24V, 1,7 W	0,00
9	adhesives	Hot, silicon, moment	18,00
10	Wood construction	70x60x15 cm	0,00
11	Consumables	Screws, spacers	20,00
12	electrical clips		0,00
13	Switching power supply	Input: AC 230V, 5A Output: DC 3,5V; 5V; +/- 12V	0,00

Costs from the Slovenia side (Components with 0,00 price were not bought)

Here is a list of the components used to implement the regulation and the correct functionality of the program:

Component	Quantity	Price per piece	Total price in €
Arduino	1	29,61	29,61
DHT 11	2	2,49	4,98
Soil humidity sensor	4	2,45	9,81
Alarm	1	3,46	3,46
Stepper motor + driver	2	5,10	10,19
Ultrasonic sensor	1	3,66	3,66
Keyboard 3x4	1	3,11	3,11
Wi-Fi adapter	1	7,24	7,24
LCD display HD44780	1	13,00	13,00
Wi-Fi module ESP 01	1	3,00	3,00
Relay module 8 channels	1	13,46	13,46
In total		2609,00	101,52

Costs from the Czech side (price converted from CZK by the exchange 1 CZK = 25.7EUR)

3 Practical part

For the greenhouse we used Arduino MEGA as the central part. After discussing with our colleague, we decided to carry out the following measurements in our glasshouse, namely the measurement of moisture in the atmosphere and in the soil (FC-28 soil sensor), the air temperature, and the measurement of the water level with the ultrasonic sensor (HC-SR04). To perform the actions we used 2 DC motors to open or close the windows. We also used an aquarium pump to water the soil as needed. We also included the fan and bulb system with which we will heat the greenhouse.

We also had to use relays in our system since Arduino can deliver up to 5V voltage, but we also needed 12V motor and alarm outputs, and 230V outlet for fan, pump and bulb

The program was written in the C language, in the Arduino program, the program is running the line for the line, and calls certain commands from the sub-libraries, Arduino is programmed so that if it detects that one of the sensors is not connected, or immediately notifies it via siren, and reports an error via the LCD display (20x4 characters). Arduino has a Wi-Fi module connected to it, which can connect to the local Wi-Fi, and, consequently, can provide information about the state of the greenhouse to another server, otherwise we can observe the state of the greenhouse through the LCD display, and through it we perform, and change the parameters set in the greenhouse, all this is done via a keyboard (3x4).

3.1 Electronics

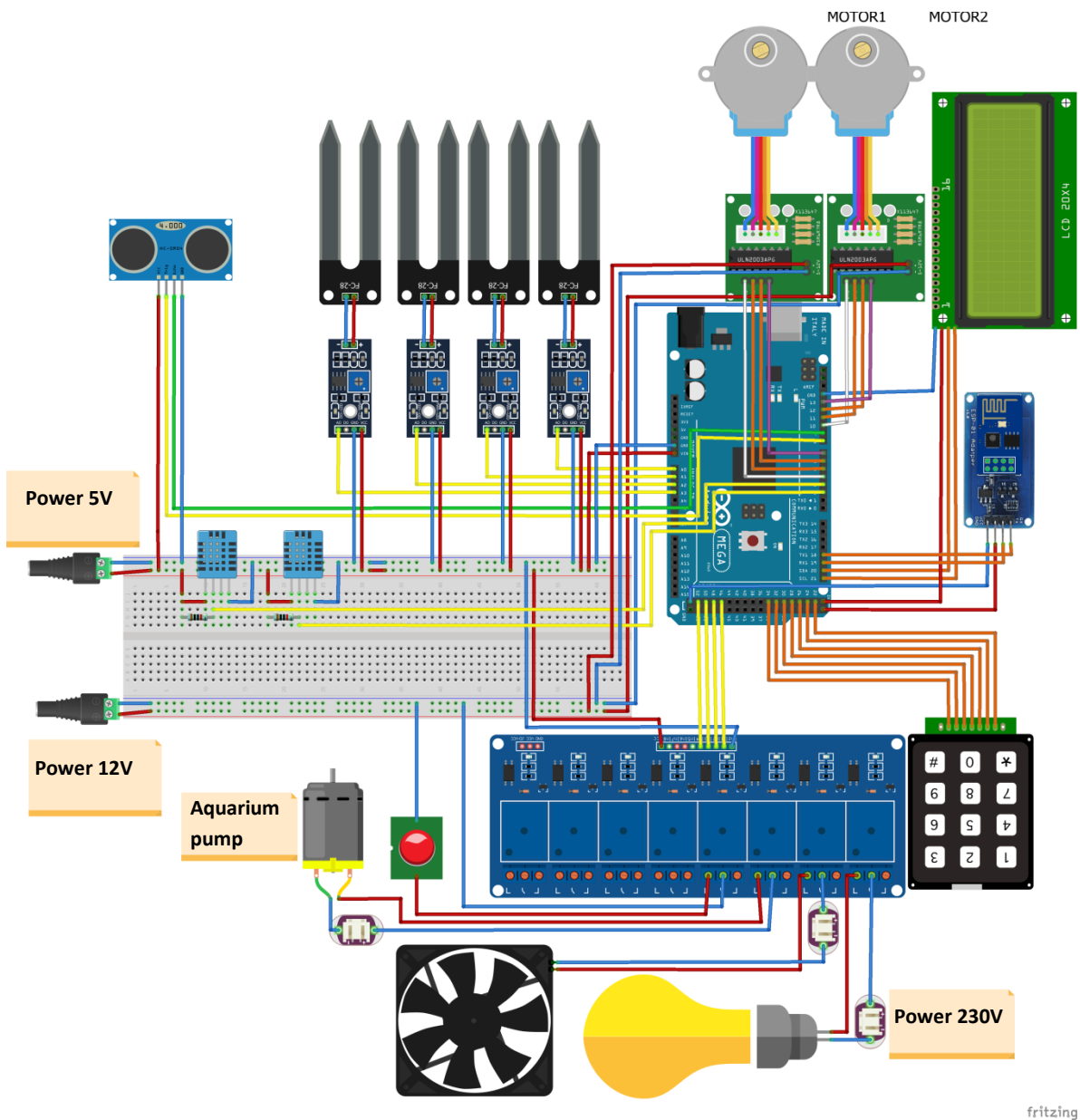


Diagram of the electrical part of the greenhouse, created in the Fritzing program

First after running Arduino and displaying Hello on the LCD display, the greenhouse asks whether you want to run it with internet connection and sending measured values to the internet or not. If you agree, it will require your Wi-Fi name and password.

Entering this data works on the principle of old button phones, so it will not be particularly difficult for older generations (see Table for entering Wi-Fi data). After selecting the correct symbol, you can press the button on which the next symbol is located to confirm it and the program will automatically save the previous one. If the next character is on the same button as the current one, the current one must be confirmed by pressing * once to confirm the current symbol and the cursor moved one position to the right. The # button is

available to delete one or more characters. The confirmation is then executed by pressing * twice and moving to the password entry stage, which works just the same except that when the cursor is moved to the next position, the current symbol changes to * to avoid any abuse. Once again, the password is confirmed by double pressing * and the Wi-Fi login starts. If it takes more than 10 seconds to log in, you probably have one of the wrong entries and must restart the entire Arduino using the Reset button. If WIFI CONNECTED appears on the LCD, it means that it has successfully connected to the Internet.

count	button	0	1	2	3	4	5	6	7	8	9
0		space	.	a	d	g	j	m	p	t	w
1		0	-	b	e	h	k	n	q	u	x
2			_	c	f	i	l	o	r	v	y
3			/	A	D	G	J	M	s	T	z
4			&	B	E	H	K	N	P	U	W
5			1	C	F	I	L	O	Q	V	X
6				2	3	4	5	6	R	8	Y
7									S		Z
8									7		9

Table for entering Wi-Fi data

After a successful login, the sensors are measured and checked to see if they have been measured. If there is a bad contact between the sensor and the Arduino, or if there is a complete disconnection, the display will show *Error* and number 1-3 (except for soil sensors) and the buzzer will be triggered and then switched off by pressing *. Here is a list of errors and the sensors that belong to them:

Errors	Type of sensor	Pin on Arduino
Error 1	DHT11	2
Error 2	DHT11	3
Error 3	Ultrasonic	Echo pin – 8 Trig pin – 9

List of the errors and their meaning

If the values pass the check, some of them must be transferred or averaged. Then comes another check, this time whether or not a regulation should start or stop. If so, it will show up on the LCD. Then, the adjusted measured data is sent to the display and then to the Internet.

Finally, there is some time to wait for the * button to be pressed to turn on the setting mode. In it, the limits of temperature and humidity control intervals can be changed.

This mode consists of three phases: the selection of the quantity, the choice of the interval of the interval (the lower = the minimum, the upper = the maximum) and the input of the new value itself. Again, the buttons * (confirmation) and # (deletion of the last number, return to the previous phase, when the button is pressed in the first phase with the empty entry field ending the setting mode) are used, but the 0-9 buttons are already typing numerically. Here are the preferences in the stages:

Number	Phase 1	Phase 2	Phase 3
1	Temperature	Minimum	Entering new values in the range 10-99
2	Air humidity	Maximum	
3	Soil humidity	-	

Phase of the setting mode

If the * button is not pressed at the time to start the mode, it will return to the beginning and the sensors will start to measure again. The whole process, from measurement to waiting for setting mode to run, takes about one minute.

3.2 Programs

We use the Thingspeak website to view the measured data on the Internet via Wi-Fi. We've created an account on Google and Thingspeak to use it. Here are the data used to create them:

First name Green	Last name House
Username greenhouse.erasmus@gmail.com	
You can use letters, numbers & periods	
Password Greenhouse2019	Confirm Greenhouse2019

Gmail account data

User ID

GreenhouseKA2



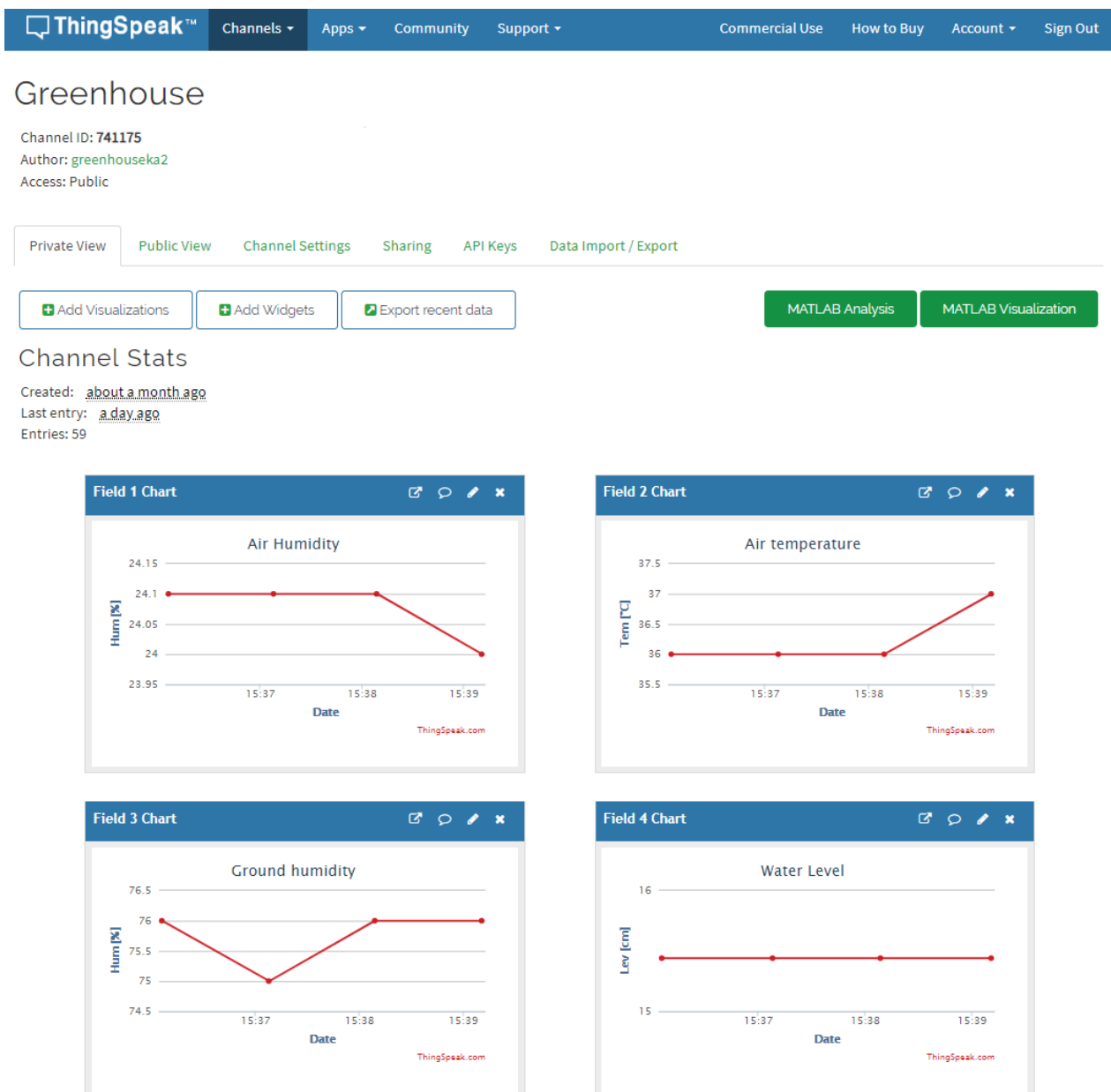
Password

Greenhouse2019



Thingspeak account data

Here is Thingspeak page with graphs of individual values:



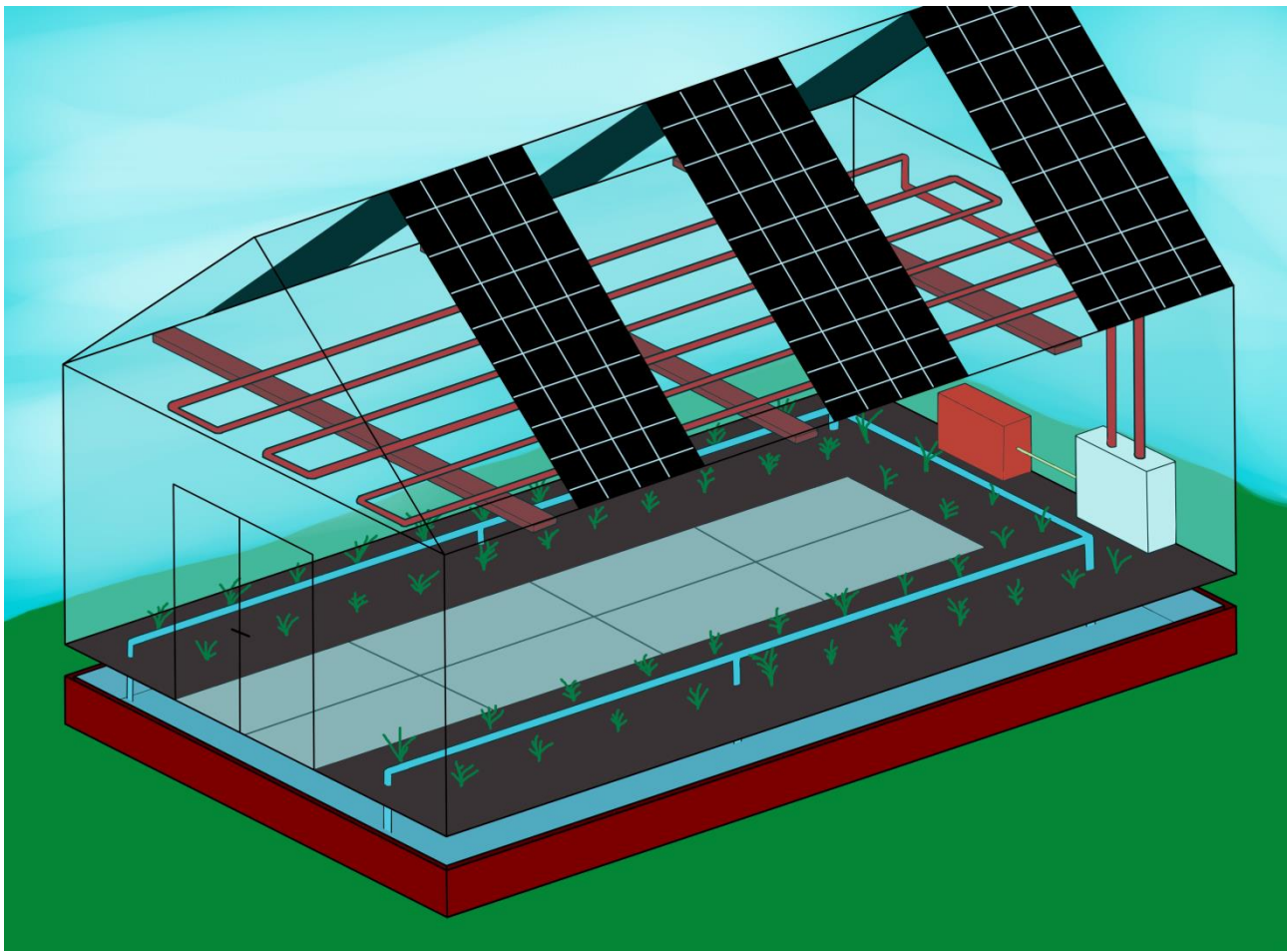
Thingspeak website

3.3 Mechanical

We have incorporated or assembled a shaft connected to the electric motor in our greenhouse and are connected to the window and through this shaft to the nuts attached to the greenhouse window that opened or closed the greenhouse window in relation to the temperature or humidity in the greenhouse if it is too big or too low.

We also used a PVC tube with a larger diameter in which there is a 230V bulb with a power of about 70W, which will be located on one end of the tube through the fan, the heater will heat.

3.4 Testing



Example of a smart greenhouse in practice

Since the greenhouse is not yet completely finished, there was no possibility to test it as a whole. As far as the software part is concerned, we have the whole done and tested.

First I tested and familiarized with the components to see if they were defective. I found either a trusted program for that component on the Internet, or a sample program obtained by installing some of the libraries, and after uploading the program tried to get some feedback. After verifying the individual components, I gradually added all the test programs to one program and always checked for problems. Then I slowly edited it and adjusted it to the specifications. When this was working, I started creating my sensor control functions, entering values, and the entire setup mode. In the end, I tried to extend it with a Wi-Fi module, send data to the Thingspeak site, and enter data to sign in to Wi-Fi so that you can sign in anywhere.

When it comes to input and control testing, whether it's setting mode or Wi-Fi data, I tried to try out all the options a person could do, who didn't know how it worked and when I found an untreated option, I tried to fix it and try it until the problem is resolved.

In the control testing, I changed the environment of each sensor and checked whether the LED on the relay was on or off.

4 Conclusion

The task was to create a functional model of a greenhouse with given specifications, what it should be able to do. Since we haven't finished it yet, I can't judge whether the job was successful or not, but I hope it was successful. So far, both the program and the model looked very promising, now to work together.

Since we were to create a mere model, it was not possible to implement some of the regulations or to implement them even for a greenhouse in real terms, so we were partially limited. In addition, a greenhouse with real dimensions should have sensors of temperature and humidity from outside to use, for example, air conditioning to replace our incandescent lamp and fan, under unfavorable control conditions (heat, frost or even high humidity). To charge the battery, solar panels can be put on the roof of the greenhouse for more economical operation.

On my programming side, I had the biggest problem with entering and deleting the Wi-Fi login to make it like button cell phones. Sometimes when I erased characters, they were deleted more than I had, or they were displayed correctly on the display, but it was saved differently in the variable. Finally, I put it into operation and it should work normally.

This project opened the door to the world of Arduino programming and the peripherals I can connect to. At the beginning of this project, I had little experience with Arduino Ethernet, DHT11 sensors, relay modules, and the Thingspeak site, so it wasn't a complete stranger to me. Now I have added new sensors and Wi-Fi module.

Furthermore, I had the opportunity to train the English language, get to know the new region and the people living in it and try to work with someone who lives outside this state. I have to say that working with someone remotely is quite challenging because we can't discuss each other and have to wait for the answer from the other and at the same time not knowing when to read it slows down the whole job. I prefer working with a person I can meet one afternoon or weekend and work together. I think it is faster and more effective than social networking. Yet it is a good experience, because something like this can happen at work, so I will know how to work in this situation.



Zespół szkół technicznych, Mikolow, POLAND
Srednja poklicna in tehniška šola, Murska Sobota, SLOVENIA

SMART MIRROR



Co-funded by the
Erasmus+ Programme
of the European Union

Contents

Table of contents

Contents	2
1 Introduction.....	1
2 Theoretical part	2
2.1 Market analysis.....	2
2.2 Technical analysis	2
2.3 Used Technologies.....	2
2.4 Financial Analysis.....	3
3 Practical part.....	4
3.1 Electronics	4
3.2 Programs.....	4
3.3 Mechanical	4
3.4 Testing	4
4 Conclusion	5
5 References	8
6 Attachments	9

Table of pictures

Picture 1: Example of a smart mirror	1
Picture 2: A few Smart Mirror examples on etsy.com	2
Picture 3: Raspbian Operating System	3
Picture 4: HTML markup language	3
Picture 5: JavaScript coding language	3
Picture 6: Python programing language	3
Picture 7: Finished project_1.....	6
Picture 8: Finished project_2.....	7
Picture 9: Raspberry Pi Touch Display	9
Picture 10: PIR HC-SR501 sensor	9
Picture 11: Raspberry Pi 3 Model B+	10
Picture 12: JSON example.....	10

1 Introduction

Our project is a smart mirror which is able to detect motion and then show the data everybody needs in the morning when stepping in front of a mirror. Specifically it shows the time and date, and also the current outside temperature in the city that you want. It will help the mankind by not having to check their phone or television for the weather, or not having to check the clock for the time and date. Smart mirror can be pretty helpful, for example when you are dressing up in the morning in front of the mirror it shows you the weather and temperature outside so you can choose the appropriate outfit.

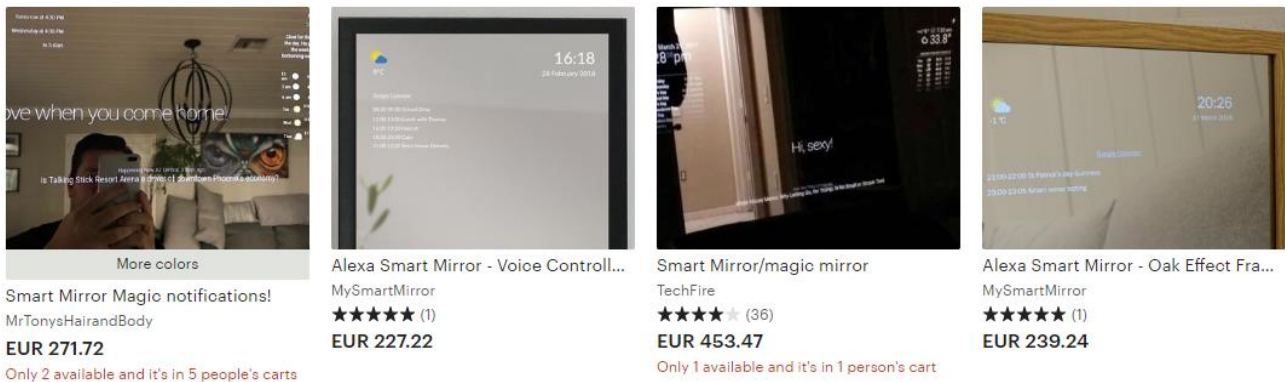


Picture 1: Example of a smart mirror

2 Theoretical part

2.1 Market analysis

On the market you can see similar solutions to our mirror, with a bit higher price and maybe quality. Going from 200 € – 350 €, sounds a bit of a high price, and with that you don't get the flexibility of doing things different, the way you want it. And it also limits you so it's hard to do future upgrades.



Picture 2: A few Smart Mirror examples on etsy.com

2.2 Technical analysis

The smart mirror is a two-way mirror with a screen behind it. The screen shows the temperature and date, and is connected to a Raspberry Pi 3 Model B+. We added a PIR HC-SR501 sensor, so the screen turns on only when motion is detected and stays turned off when no motion is detected.

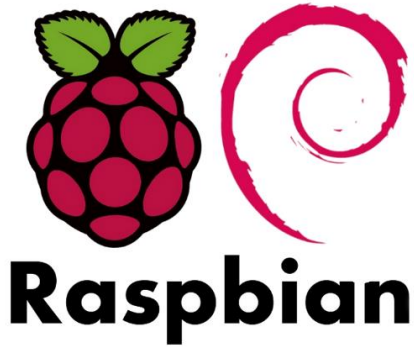
2.3 Used Technologies

For the smart mirror we used a Raspberry Pi Touch Display to show the data behind the mirror. PIR HC-SR501 sensor was used to detect motion. (Picture 9: Raspberry Pi Touch Display, Picture 10: PIR HC-SR501 sensor)

The brains of the project is the Raspberry Pi 3 Model B+. (Picture 11: Raspberry Pi 3 Model B+)

For making the frame of the mirror, a circular saw was used and some other woodworking tools.

Raspberry Pi is running a Raspbian operating system. The site for showing all data was programmed in HTML and JavaScript. The program for the motion sensor detection was written in python.



Picture 3: Raspbian Operating System



Picture 4: HTML markup language



Picture 5: JavaScript coding language



Picture 6: Python programming language

2.4 Financial Analysis

Strengths of our project are: the requirement of only one cable (power), the thing that you can change the data that you are seeing on the mirror by the raspberry behind it, it looks modern and futuristic.

Weaknesses: The display which is used on the mirror isn't as big as it could be, but for the same price we could get a bigger screen moreover it's not bright enough. The location has to be set manually because of the lack of a GPS module.

Opportunities: No mainstream smart mirror companies.

Threats: No threats.

3 Practical part

3.1 Electronics

From the power outlet goes the electricity needed to power the Raspberry Pi. The Raspberry Pi has a ribbon connector going to the Display, which is showing all of the data. The Display also requires power, so that is supplied from the Raspberry Pi. The PIR sensor is connected to the Raspberry Pi with three wires, two for power and ground and one for the detecting signal, which sends 3.3V.

3.2 Programs

The HTML site showing all of the information gets its weather data from <https://openweathermap.org/> which is a great site because it has information from all over the globe. The data is in JSON form so it is easy to read (Picture 12: JSON example). The site also shows the current time and it refreshes every 5 minutes to get new information. The PIR sensor is operated with a python code which checks the signal transferred to the Raspberry Pi every few seconds. If it detects no signal (no motion), the display turn off after some set seconds. But if it detects a signal it turn the screen on.

3.3 Mechanical

For the Mirror a frame was designed so it can be hung, but also so all the electronics are in place. The wooden frame has a hole at the bottom where the PIR sensor sits. The Display and Raspberry Pi just slide in the back, so the screen is flush with the mirror.

3.4 Testing

First testing was just the website opening and then opening at full-screen on Raspberry pi startup. Till then it was all going smooth. When the PIR arrived we plugged it on the Raspberry Pi like a diagram on the internet site showed, because our sensor doesn't have any markings for what either pins are. So for a long time we had it wired wrong, we had the power and the ground switched and to our relief it didn't fry anything. Then when we switched the wires to the correct connectors it started working correctly.

4 Conclusion

In conclusion our teamwork to make the best we can out of what we have was very good. We have successfully created a Smart Mirror that is still upgradable and able to be modified. The things that we would change/modify are: -buy a bigger screen (a normal computer monitor) to fit more information on it; -get a GPS module or find some way to automatically get the location of the mirror and with that show the proper weather; -probably make a software for showing the data rather than a website.

Overall we are really happy with our product and are ready to improve it as stated above.

About the expanding and creating a business out of it doesn't sound bad at all, could be profitable, but not in the state that it is in right now.

The international cooperation was successful, it forced us to use the English language, but we also tried to communicate in our own languages, which wasn't so effective. We came to a common solution every time, there was no arguing.

We liked the project and will encourage younger students to participate. It makes you more flexible with teamwork, unique and tough situations and basically expand your imagination of what is achievable.



Picture 7: Finished project_1



Picture 8: Finished project_2

5 References

<https://www.instructables.com/id/Smart-Mirror-by-Raspberry-Pi/> (accessible on 22.3.2019)

https://www.etsy.com/market/smart_mirror (accessible on 22.3.2019)

<https://pi-hole.net/raspbian-logo/> (accessible on 22.3.2019)

<https://en.wikipedia.org/wiki/HTML> (accessible on 22.3.2019)

<https://www.javatpoint.com/javascript-tutorial> (accessible on 22.3.2019)

<https://www.python.org/> (accessible on 22.3.2019)

<https://www.amazon.com/Raspberry-Pi-7-Touchscreen-Display/dp/B0153R2A9I> (accessible on 22.3.2019)

<https://www.addicore.com/PIR-Infrared-Motion-Sensor-HC-SR501-p/168.htm> (accessible on 22.3.2019)

<https://shop.pimoroni.com/products/raspberry-pi-3-b-plus> (accessible on 22.3.2019)

6 Attachments



Picture 9: Raspberry Pi Touch Display



Picture 10: PIR HC-SR501 sensor



Picture 11: Raspberry Pi 3 Model B+

```
{"coord":{"lon":16.17,"lat":46.66},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01d"}],"base":"stations","main":{"temp":284.87,"pressure":1033,"humidity":57,"temp_min":282.15,"temp_max":287.04},"visibility":10000,"wind":{"speed":1.5,"deg":190},"clouds":{"all":0},"dt":1553247346,"sys":{"type":1,"id":6825,"message":0.0041,"country":"SI","sunrise":1553230520,"sunset":1553274555},"id":3194648,"name":"Murska Sobota","cod":200}
```

Picture 12: JSON example



Stredná priemyselná škola elektrotechnická, Košice, SLOVAKIA
Miskolci Szakképzési Centrum Kandó Kálmán Informatikai
Szakgimnáziuma, HUNGARY

RFID SYSTEM FOR BORROWING KEYS FROM SCHOOL RECEPTION



Co-funded by the
Erasmus+ Programme
of the European Union

Contents

Table of Contents

Contents	2
1 Introduction.....	1
2 Theoretical part	3
2.1 Market analysis.....	4
2.2 Technical analysis	4
2.3 Used Technologies.....	5
2.3.1 Server-side script language - PHP	5
2.3.2 Database system: MySQL	5
2.3.3 Hypertext Markup Language: HTML.....	5
2.3.4 Client-side scripting language - JavaScript	6
2.3.5 Styling sheets: CSS	6
2.3.6 Server Distribution Apache.....	6
2.3.7 Developing platform Arduino Uno R3	6
2.3.8 Programming language Wiring (C++)	7
2.3.9 Programming language C++.....	7
2.3.10 RFID Technology	7
2.4 Financial Analysis.....	8
3 Practical part.....	10
3.1 Electronics	11
3.2 Programs.....	11
3.3 Mechanical	13
3.4 Testing	18
4 Conclusion	19
5 What can be improved	20
6 References	21
7 Attachments	22

1 Introduction

RFID technology is widely used in this time period. However, it is also unknown term to many people nowadays. Population nowadays is meeting this technology mostly every day without recognizing it. Many people could meet it in a work place or in cantina of the school or the company. Also, this technology is used to check and control all the attendance of workers. In the school lunchrooms, cookers can easily manage whether student has some diet or some food allergy and depending on it serve exact meal to everyone.

During the time, when we were thinking which project we will take, we've been also thinking about the real establishment of the project in a real life and the way, how to change something from something used in those days to something needed and much better than it was in the past.

Each project was given to us with a proper specification and demand because of the international collaboration of European countries in a project called PLEL under the wings of Erasmus+ organization of each country.

When we were choosing the best project and the way of the creation it, we were mostly oriented in a way, that this project should be time efficient in a school reception. Our RFID based borrowing system should change the way of borrowing keys in a much more modern way. This project might be used in our schools but in the other hand, it could be used in many organizations too, where is important to track the borrowing of the keys or other things and manage the processes in such a better way. We can also, say that there are many other ways and plenty similar projects are used, but they are much more expensive, and they are also not the best solutions for everyone.

Our project is based on more platforms such as programming language PHP, server Apache or development on the Arduino platform. Because of that, we have divided the project into two parts: server and client. The rule of the Arduino programming and the case was Alex's task and the processing the data on the server side and the frontend for the receptionist in a web development was Erik's part.

For the buying thing we mostly used the official shops with electronics and some internet shopping platforms.

The biggest problem with this type of project was communication with each other. It is so hard to test these things in such a big distance. However, we had to opportunity to test it together because we had included a scholarship in this project. On one week we were in Hungary and the other we were in Slovakia. There we had the opportunity to test it and made the finalized the project.

Every week, our work was continuing and a despite of small problems with communication and misunderstandings of the specialization, we were moving with that to the final goal. But everything would not be possible without the help of our school tutors and we would like to thank our tutor and whole project coordinator Ing. Michal Copko for the technical help and support with solving cases. The same acknowledgements belong to Ing. Rolland Kiss from Hungary, too.

The key borrowing system that we implemented is a cost and time effective, modern alternative to todays standard, paper tracked key borrowing, where human labor and tedious tracking of the keys are required. With our system one can simply set it up once, and expect it to work with little to no maintenance. Our system is also intuitive for the end user(s), with an everywhere accessible web app.

2 Theoretical part

We didn't have to make any analysis for our project because we had a very well-defined documentation detailing what we had to do. Also, our project coordinators were helpful explaining some specifics about what we had to do.

We had to implement an automatic, logged key borrowing system. For this we decided to go with a client-server architecture where the client is a terminal only sending input and receiving output from the server. This means that much of the work is delegated to the server. For this we had to establish a secure communication between the client and the server. After we could communicate safely over the wire, we established how our data will look like on the client and how the server will make sense of it. When all was good and done the only thing that remained was testing it out.

Technologies used by our project are widespread, mature and mostly free technologies. Technologies that cost money have a free alternative in parentheses. Our software stack consists of:

- An Operating System capable of running the Apache webserver, in our case ©Microsoft Windows was used (Linux is a free, widespread Server Operating System alternative)
- The after mentioned Apache webserver, part of the XAMPP stack
- The PHP scripting language, part of the XAMPP stack
- MySQL as a database system, part of the XAMPP stack
- Arduino's AVR GNU Compiler Collection stack, which contains the Ethernet library used in this project for TCP/IP based communication
- The MFRC522 RFID library

Electronics used were:

- A computer, capable of handling our software stack
- The Arduino single-board microcontroller
- Arduino ethernet shield
- MFRC522 rfid reader

- RJ45 Ethernet cables
- Relay (a door lock opening or closing upon voltage)
- Micro switches
- HP 4067 multiplexer

The only **non-electronic** parts required for this project are the material of the casing, which in our case was **wood**.

The **Tools** needed for this project, are the ones required to build the casing which in our case was simple tools available in most homes/workplaces being:

- A hammer
- Nails

Detailed description of technologies, tools and softwares used to make this project is discussed in Section *Chyba! Nenašiel sa žiaden zdroj odkazov.Chyba! Nenašiel sa žiaden zdroj odkazov..*

2.1 Market analysis

Our project does not have an easily googleable counterpart, which means we targeted a specific problem with our solution. Some other solutions that draw similarities to ours are *library book tracking systems*, which are similar for tracking, but totally different by usage. For example our solution is fully automatic once setup, while most library book tracking solutions need human labor to register a lease or a return.

2.2 Technical analysis

Our project should be able to hold and track keys in a special key holder enclosure, protected by an RFID authenticator system, tracking every borrowing and return on a server, being able to secure and easily retrieve said servers data through a web interface.

The projects backbone is the well-known client-server connection setup, with the client being the key holder and the server running our server software. The system communicates over the Internet or LAN. The servers web interface is also accessible from the Internet or LAN depending on the configuration.

2.3 Used Technologies

In this section a detailed list of technologies used to make the web application (2.3.1-2.3.6) and the client application (2.3.7-2.3.10) is shown.

2.3.1 Server-side script language - PHP

From many server programming languages, we had chosen the PHP. The biggest reason of this choice was, that this one is widely spread all over the world and it is used by many people for creation systems like our one. This language is also used for making the page content much more dynamic. It has also very well support of mostly each web server, so the solution could be implemented mostly everywhere. Most of its attributes were taken from C and Perl. PHP can cooperate with database and consists of object programming support too. Despite of all this, PHP is using simple syntax so many users could understand it and simply correct the mistakes. It is running on a web server and as an input it is taking the PHP scripts and as a result is giving the web pages, but it is not the rule every time. PHP scripts usually has suffix .php, but the suffix could be changed in the server settings. Official page with a manual is on the address *www.php.net*. It is spread as „PHP License v7.1.23”, which enables to use freely this language.

2.3.2 Database system: MySQL

MySQL is a database system based on a SQL language which is used for storing data from web applications. It is mostly used with a PHP programming language because of the good cooperation between these two languages. This system is open source, so its community is wide, and it has really well support for users too. This database system has also well-made documentation and users can find it on: www.dev.mysql.com. Database system MySQL is cooperating with almost each unix and Windows systems. It uses GNU/GPL license for free usage.

2.3.3 Hypertext Markup Language: HTML

The Hyper Text Markup Language is the standard language for creating web sites and web applications first released in 1993. It is the first building block of every website. HTML files are opened through a web browser, which will parse and show the website following the standards pre-defined rules. We choose the latest version of the language HTML5 which is well supported across all major browsers. HTML is responsible for the content of a website.

2.3.4 Client-side scripting language - JavaScript

JavaScript is an interpreted, object oriented dynamic scripting language parsed and run by the web browser. It's first release was in 1995. Alongside HTML and CSS JavaScript is one of the core technologies required to build a website or web application. JavaScript enables simple HTML web pages to be more dynamic. Paired with an appropriately designed backend, one can make a single site application, which only updates parts of the page to show new content.

2.3.5 Styling sheets: CSS

Cascading style sheet is as styling language. It helps us to make the design of the website. It means that it makes all data representation on a client side for nice view for the users and hosts on the website. This document includes all the style of the page and how should elements be seen. This division helps the site make the unique view.

2.3.6 Server Distribution Apache

Apache is a free webserver released in 1995. Apache is popular and is estimated to be running on 39% of all web servers. Most of its users run it on the Linux Operating System but Apache supports Windows and a wide range of Unix like systems. Earlier versions supported a lot more Operating Systems, but support was cut for them. The Apache webserver is designed to be extended through modules, which cleanly separates its core purpose from other tasks it may be required to complete. We specifically used the PHP and MySQL apache modules to get PHP scripting and database connection capabilities in PHP.

2.3.7 Developing platform Arduino Uno R3

Arduino Uno is a single board microcontroller based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again. The Arduino family of single board microcontrollers are an LGPL licensed open-source hardware and software stack developed by Arduino LLC. The boards are manufactured by Arduino SRL. The premise of this company was to make cheap and open

microcontrollers. The hardware reference designs are available only, subject to the Creative Commons Attribution Share-Alike license.

2.3.8 Programming language Wiring (C++)

Wiring is an open-source programming framework for microcontrollers. Wiring allows writing cross-platform software to control devices attached to a wide range of microcontroller boards to create all kinds of creative coding, interactive objects, spaces or physical experiences. The framework is thoughtfully created with designers and artists in mind to encourage a community where beginners through experts from around the world share ideas, knowledge and their collective experience. There are thousands of students, artists, designers, researchers, and hobbyists who use Wiring for learning, prototyping, and finished professional work production.

2.3.9 Programming language C++

C++ is a low-level multi paradigm programming language designed in 1985. Despite its rather old age the language still sees a lot of use cases. C++ is a compiled language, which means that in contrast to JavaScript for example, the code is not understood by the machine line by line, as it is running, but it is transformed into a machine runnable binary file. This has the advantage of blazing fast speeds but brings some disadvantages with it too. For example, a C++ program compiled for a 64bit processor won't run on a 32bit processor, while an interpreted language will work without changes. This is the same with CPU architectures. C++ was used with the Arduinos AVR GCC stack, and the chosen libraries, but the application code is C style (procedural).

2.3.10 RFID Technology

Radio Frequency Identification (RFID) is an automated identification technology that uses tags to transmit data upon RFID reader queries. In another word, RFID is the wireless non-contact use of radio-frequency electromagnetic fields to transfer data for the purposes of automatically identifying and tracking tags attached to assets (Jin Li, Cheng Tao, 2006). There exist RFID technologies of different categories based on their capacity. Even if they are different in system or categories, they all are based on same fundamental principle where the attribution of unique physical goods can be easily transposed to the computer system (Fabian Ropraz, 2008).

2.4 Financial Analysis

A quick SWOT analysis of our project yields the following important financial attributes

<i>Strengths</i>	<i>Weaknesses</i>	<i>Opportunities</i>	<i>Threats</i>
Cheap, well implemented solution	Lack of prebuilt units of our solution.	Well-defined market niche	Convincing large consultancies to use our solution may be hard

The minimum budget of this solution is around 35€ for the client, 10€ for the wooden casing of the client and 32€ for the cheapest server, being a Raspberry Pi 3, including a 13€ shipment cost on the parts totals to about **90€** as of the time of writing. Because of the DIY nature of our solution, owning any of the parts brings the cost down.

As a server, we could also use a hosting service which costs from 20€ per month, but for this project it could run just as a virtual server on the school server. In the table above you could see the prices and the conclusion of them.

Item	Price
Arduino uno R3	20,-€
Ethernet shield 2	21,50,-€
RFID reader	6,-€
Arduino adapter 9V	3,50,-€
Network UTP cabling RJ45	0,50,-€
Breadboard	8,-€
Wood casing	10,-€
Final price	69,50,-€

The solutions on the market are mostly much more expensive because they are made as a complete solution with all room tracking system where the administrator could exactly see

everything. From the door opened, up to person who used code, RFID tag or card to open it. But this solution needs more than 100€ for applying it as a real system in the school and much is more unpractical is needed. In the school, there is no need to see which room has been opened and you don't need to see who entered it. All those data could be taken as an GDPR thread for people and the freedom of move. Yes, sometimes it could be good to use the electromagnetic locking for doors for easier opening, but it cost a lot. When we take the count of rooms in the school and number of doors and locks which we will need for project like this, we will consider, that is priceless and very expensive for school. It could be implemented in a future, so the teacher won't need then going for a key and he or she will use a simple car to open each room with approval of the system. But this is not real for now because of the price.

3 Practical part

For logging and the security of the keys, we used RFID cards which when registered with the server, will open the door and append metadata after registration to the borrowing or return such as the name of the card owner, the date and time when the action happened, which key was taken or put back. If an unauthenticated user tries to open the door the server will simply deny access, but no logging of such an event will take place.

For this to work out we had to make our Arduino a viable terminal for simple, secure communication with the server. This means that even if the communication between the terminal and the server is understood by a third party, it should not be possible to open the door without the server's consent.

But this only delegate the problem, not fixes it. Even though the Arduino is safe from tempering this way, the server still has to be secured, because now the server is the sole authenticator of the system. For this we prepared a sessional user login system for the web application and all operations on the server side require this authenticated session. To further mitigate threats, we also prepared the web app against SQL injection vulnerabilities with the `mysqli_escape` family of PHP functions. This is a pretty common and widely used technique to secure a database.

Now that our modules for the project was done and secured against potential threats, we had to make it functional as a unit. To do this we had to make the Arduino communicate with the server. For this we choose to communicate over ethernet cables and the standard HTTP 1.1 protocol. This came with a lot of pros that outweighed the cons, for example

1. The server already was designed for this, and was fast to speak this protocol
2. The HTTP is a well-established protocol used all over the web and beyond
3. The PHP programming language has built in language constructs to help parse HTTP data (POST and GET data) easily

So, we had to implement a simple version of the HTTP protocol on the Arduino, these mostly consisted of data sending and reading responses back. After we did this, we had to decide the data, the order of sending said data and what communication should happen between the server and the terminal. We settled on the following steps:

1. The Arduino gets an RFID card, reads it, makes a connection to the server and sends the cards ID to the server
2. The server checks the card's ID against its database of authenticated IDs and if it's found it responses 1 – open the door - or 0 – to keep the door closed – If the response is 0 the connection is closed and we are back to waiting for cards
3. If the response was a 1 the Arduino opens the door and waits for a key to be taken or put back to place. The user has an indefinite time for this which can be changed through the server's timeout configuration. When a key is put back a positive integer, representing its place in the closet from left to right, top to bottom is sent to the server. If it's taken the same number is sent with a negative sign.
4. The server gets its number representing the key and the action, closes the connection with the Arduino and does it's bookkeeping of the keys. It also updates every open instance of the web interface.

3.1 Electronics

The PCB we used was the Arduino Uno R3 microcontroller which has a 16MHz 8bit CPU which was equipped with a W1500 Ethernet-Micro SD card reader combo shield. We omitted the use of the SD card and disable it while running our client software. Using a multiplexer, we also added an RFID reader to the Arduino, but this came with a downside: at a time, we could only use the ethernet shield for communication or the RFID reader for getting cards. This didn't turn into a problem because of our one time use for the RFID reader which happens at the beginning of our transaction process.

The design of the PCB for the Arduino can be seen on Attachment 1 and 2 on page 13.

3.2 Programs

We developed and connected 5 modules for this project

For the arduino these were:

- A module to set up the ethernet shield for HTTP protocol style connection over the wire
- A module to read the ID's of RFID keys

On the server side we developed:

- A module to make Session based logins and made all server side functionality check for this session before any action.
- A module to put into and retrieve data from the MySQL database
- A module that can query the database and show the results in real-time

These modules were connected through Ethernet cables on the hardware, and the HTTP 1.1 protocol on the software layer.

One clever line of code that came out of a miscommunication was the determination of the key's number. We made our Proof Of Concept to have keys numbered in a 4X4 row like the following:

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

This is called a Column Major matrix, but on the software side, we agreed on a Row Major matrix which looks like the following:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

To overcome this limitation without having to take our closet apart, we come up with the following clever line of array indexing code:

```
int table[] { 0, 4, 8, 12, 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11,
15};

int result = table[changedHolder];
```

changedHolder here is the number of the key holder that had a key taken or returned, and the table array is a lookup table for converting this 4X4 column major matrix into a 4x4 row major matrix.

3.2.1 Web user interface and processing data

First and one of the most important things of this part was to create an UI and the layout of the application, then we have to decide which backend will we use. We were thinking about the way how and by which language should we create all the stuff. We managed, that the best way for the project would be combination of PHP and the MySQL database. PHP was the only option because Arduino could understand and communicate just with this one language. The main part of the UI was created by HTML, CSS and partly by JavaScript and jQuery library.

3.2.1.1 HTML, CSS and JavaScript

The main output of the project data is on the website. For this type of project, we haven't used any of CMS (Content Management System) but we have decided to use our knowledge of creating and managing websites. For the design and all this stuff, we have used two column model with a side menu bar and the layering messages. We have styled our own buttons and the color schema of the page. As the main output we are using a simple table.

Id (count)	Room	Teacher	Borrow time	Return time
1	28b	Janko Hráško	21.2.2019 12:32:02	21.2.2019 13:23:13

Table 1 Example for web output - porter

Id (count)	Room	Tag	Teacher	Tag	Borrow time	Return time
1	28b	124594561	Janko Hráško	12ade1586	21.2.2019 12:32:02	21.2.2019 13:23:13

Table 2 Example for web output - Administrator

For a dynamic point of the website we have used several jQuery and JavaScript functions. One of the is used for showing the layer messages, the second one is used for opening the menu. Other functions are used for interactivity and the intuitiveness of the website.

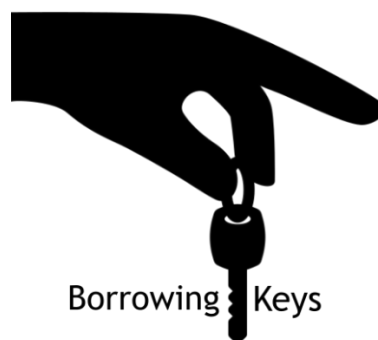
We have also a special function for displaying the names as a search engine. It is connected with PHP which is responsible for searching data match in a database and then display it on the screen if there is a match.

3.2.1.2 Color, shapes and logo

Our main idea was simple, we wanted just easy and simply made output. But during the time we considered to change the point of view and we decided to make it more complex and unique. This was also the reason of the project future use as a real system as we planned.

We have made our custom color scheme via online tool www.color.adobe.com where we made a color triad.

The second important thing for making a unique system was to have our own logo so we made one. The biggest inspiration comes by a website with a lot of free icons www.freepik.com. Then we considered to make our own with an open source software Inkscape. For the descriptive logo we have added a word phrase: "borrowing keys" to make the sense of the project purpose.



Picture 1 System logo

3.2.2 Creating program part of the software

3.2.2.1 Algorithmization

When we were writing the code, we tried to follow all applicable standards and rules for writing semantic and syntactically correct code. In the case of testing, we first focused on various errors and different ways of capturing data. We followed the unwritten rule "Errors first", which should provide the system with faster interaction and error checking. To prevent MySQL injection attack, we used the best available solutions for our code writing method.

3.2.2.2 Web server side

All data that which are the output from a web application was previously processed on the server side by the PHP programming language. It takes care of the entire backend and received tags when attached to an RFID reader. PHP can select data from a database based on certain requests that are later written to the screen by the echo () command. The most common function for linking and retrieving data from a database was just the mysqli_query() function, which uploaded the necessary data from the database.

In the SQL language, we mainly worked with the selection of specific data (table rows) through SELECT, where we often determined specific parameters using WHERE, AND, OR, IS, LIMIT.

3.2.2.3 Database structure

The structure of the database, ie individual tables, was divided according to the type of stored data. We have created several tables for functionality and complexity of the solution. In borrowing, we chose to link relational databases of teachers and keys to the structure of m: n, which helped us to simplify saving and pairing of individual data.

Variable name	Id	Name	Surname	Tag
Variable type	Integer	Varchar	Varchar	Varchar

Tabuľka 1 Entity-relational diagram - teachers

Variable name	Id	Name	Tag
Variable type	Integer	Varchar	Varchar

Tabuľka 5 Entity-relational diagram – rooms

Variable name	Id_Teacher	Id_Room	Borrowing time	Returning time
Variable type	Integer	Varchar	Varchar	Varchar

Tabuľka 2 Entity-relational diagram - Borrowings

Variable name	Id	Name	Surname	Username	Mail	Password	Rights	Last Login
Variable type	Integer	Varchar	Varchar	Varchar	Varchar	Varchar	Integer	Bigint

Tabulka 3 *Entity-relational diagram* - Login data

The most important part was to determine which data would be served as unique identifiers. In the key and teacher table, there are individual "id_name" columns that may not be repeated in the table. In the borrowed key table, in addition to the teacher id and key, the borrowing time is also given as the primary key. This is so that the key can be return again after new borrowing.

3.2.2.4 *System Login*

The basis of the whole system is the possibility of its use only after login. The system itself is built on the levels of user rights, that is, depending on what settings the user is allowed to. We have chosen a simple but clear look for the login form.

In the program side, we are using storing the data about users into Sessions. There we store username, id, and user rights to the SESSION variable, which helps us display specific data for different levels.

Also, the hash () function is used for increased security, which creates a digital fingerprint from the specified password, and even if an attack was created on the database, the attacker would not get access to the passwords.

The functionality is based on the isset() function, which examines the status of the button. If Log in has been pushed, the script will test if the fields have been filled in, if not, it will go back to the blank login form. In the case of incorrect logins, the program will also return an attempt to log in to the login page.

3.2.2.5 User levels

The Key Loan Application offers 2 user levels. The first is Administrator. In addition to the standard features available to the user, the user can add and delete records from the system if something has gone wrong with the previous chip or has been lost.

In the table there is a "rights" column for the user, which defines what data the user has access to. If it contains "1" in it, then it has access to everything, if it is "2", the user does not have any additional functionality.

A standard user (doorman) can view records that have been created based on name, time, or date that have not yet been returned. It can also create manual borrowing records. returning the key to the gatehouse but does not have other options.

3.2.2.6 Manual records

Our program also offers creating manual records. In practice, this means that in the event of a reader failure, the application can still be used and there is no need to wait for the error to be corrected. The doorman, a common user of this system, simply enters the name and the number of the room to which the key is borrowed, respectively. only the room number is returned and only the system itself will take care of the rest.

After the programming page, it is solved through PHP as most applications. When borrowing the system checks whether the key is no longer borrowed, which may become inconsistency of the doorman, that the given return and the system still records it as borrowed even if the key is already physically at the gatehouse. If it finds that the key has not yet been returned, a window will pop up informing the concierge about who borrowed it and, if it needs to be registered in the system, click the link that directs it to return.

The system works similarly for returning keys. It also checks if the key has been returned, if it was, the message pops up and the problem is only on the side of the doorman, who should check the keys in the closet.

3.3 Mechanical

The casing was built out of wood. It has dedicated space for the Arduino inside it, with holes cut out for the I/O ports. Unlike the Arduino the RFID reader is exposed on the outside of the casing. The microswitches reside inside the wooden case in a 4 by 4 table.

You can see a photo of the casing from behind in Attachment 3 on page 14 and built up on Attachment 4 on page 15.

3.4 Testing

Testing the project was pretty simple in our case. After we built our Proof of Concept software and hardware implementations, we simply did a one-time setup. Parts of the Proof of Concept were debugged with logs and printing.

Our setup consisted of the following steps:

1. Grabbed a pair of keys and RFID cards
2. Added one user to the server software
3. Added one of the RFID cards as Authenticated to the server software
4. Added the two keys with a random room number to be tracked
5. Logged in with the newly created user

After we set up our solution we could easily test right away. We checked the authentication of the cards by checking if the door opened for the right RFID card and stayed closed for the other. We checked the tracking of keys by borrowing or putting back keys and checking if the web interface showed our changes or not.

The only way we limit the usage of our solution is that after every successful door opening only one borrowing or return can be made and the closet has to be closed, otherwise all subsequent borrowing or returns will be made by the name of the opener. This is of course completely optional and may not even be needed, in which case our project doesn't tie the hand of its users with any rule.

4 Conclusion

Our project was successful despite all the communication, technical and implementation difficulties we faced. We plan on cleaning up our code together sometime in the future.

My opinion on this project is not the most positive one. There were many communication issues over the Internet communication. I can say the same, that my partner Alex was prepared, and he has all the needed knowledge for both parts of the project but then comes many problems during testing. We had to change many things and lines of code to work as a unit. My opinion on Alex is therefore positive, he knew all the things and helped me with such issues in my code which was not working properly, but in the conclusion we made it. And now, I can say that the project itself is fully functional and has much more abilities than it was told to have. (Erik Takáč)

My opinion on Erik Takáč was really positive. Despite having some communication problems and misunderstandings with each other thorough written mediums, working with him in person was a really smooth experience. He was well prepared, open-minded for the project, he helped with a lot of misunderstandings related to our project. Cooperating with someone over such a great distance wasn't without cons tough, we couldn't really test how our projects play together, so the limited time we got to work together was all we had for making it or breaking it. I have to acknowledge my partners preparation, hard work and personality which was probably the biggest factor in finishing this project and making it look as polished as it is. (Alex Sárkány)

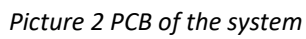
5 What can be improved

- We could have used another ethernet shield that doesn't have a micro-SD card reader or we could find a use case for the reader on the ethernet shield as for now it is wasted.
- In case of a bigger Arduino board, we could use a TLS based HTTPS instead of HTTP but this only makes the sent data readable, because the board only connects to the server, it won't receive and process data from any other host.
- As pointed out in 3.2 we miswired the microswitches and had to use a Look up table to use the original layout we talked out in advance.
- We can improve security issues in the software part such as using PDO instead of `mysqli_prepared_statements` to provide much more secure system with all the vulnerabilities for the bigger sectors than just schools.
- There is also possibility to change the UX and add the teacher login side where teachers could see which keys do they have to give back

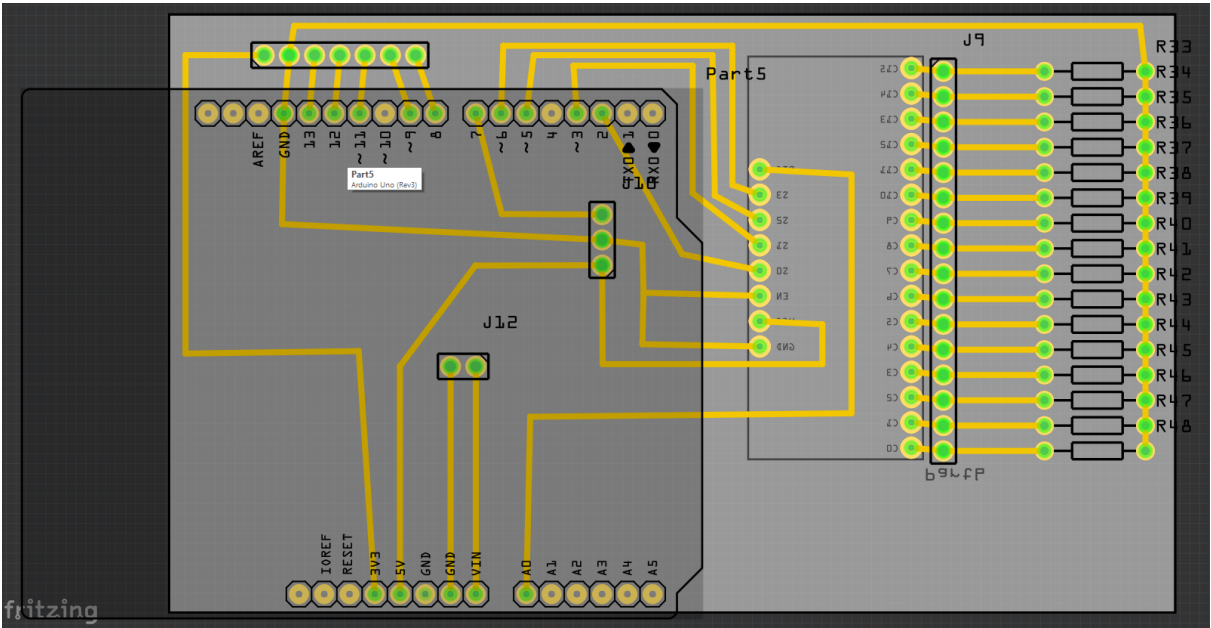
6 References

- [1] Jin Li, Cheng Tao, 2006. "Analysis and Simulation of UHF RFID System", in proceedings of the 8th International Conference on Signal Processing, 2006.
- [2] Ropraz Fabian. 2008, Project Work: Using RFID for Supply Chain Management, University of Fribourg, Switzerland

Attachment A - PCB of the project

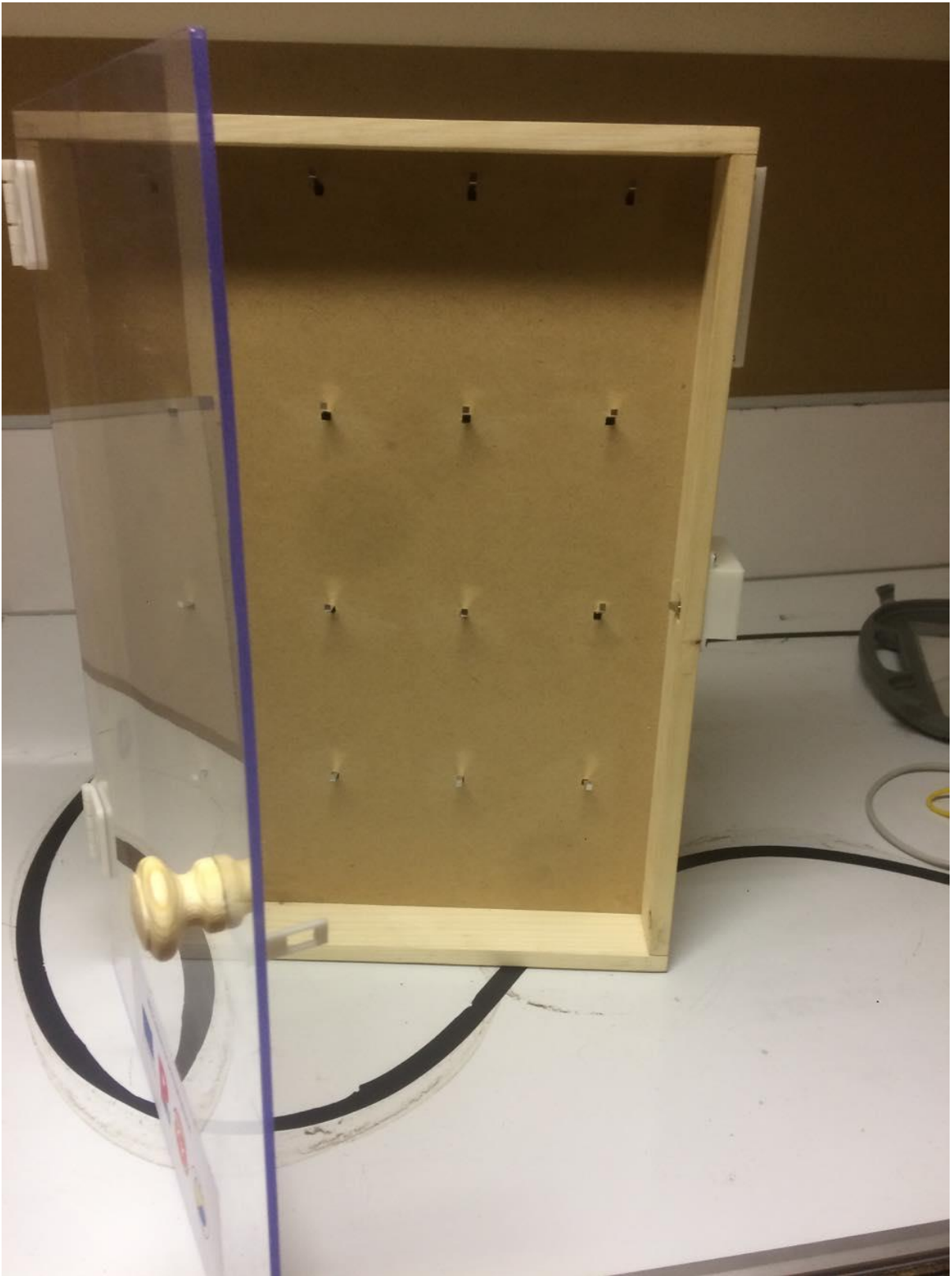


Attachment B - PCB unit of the project



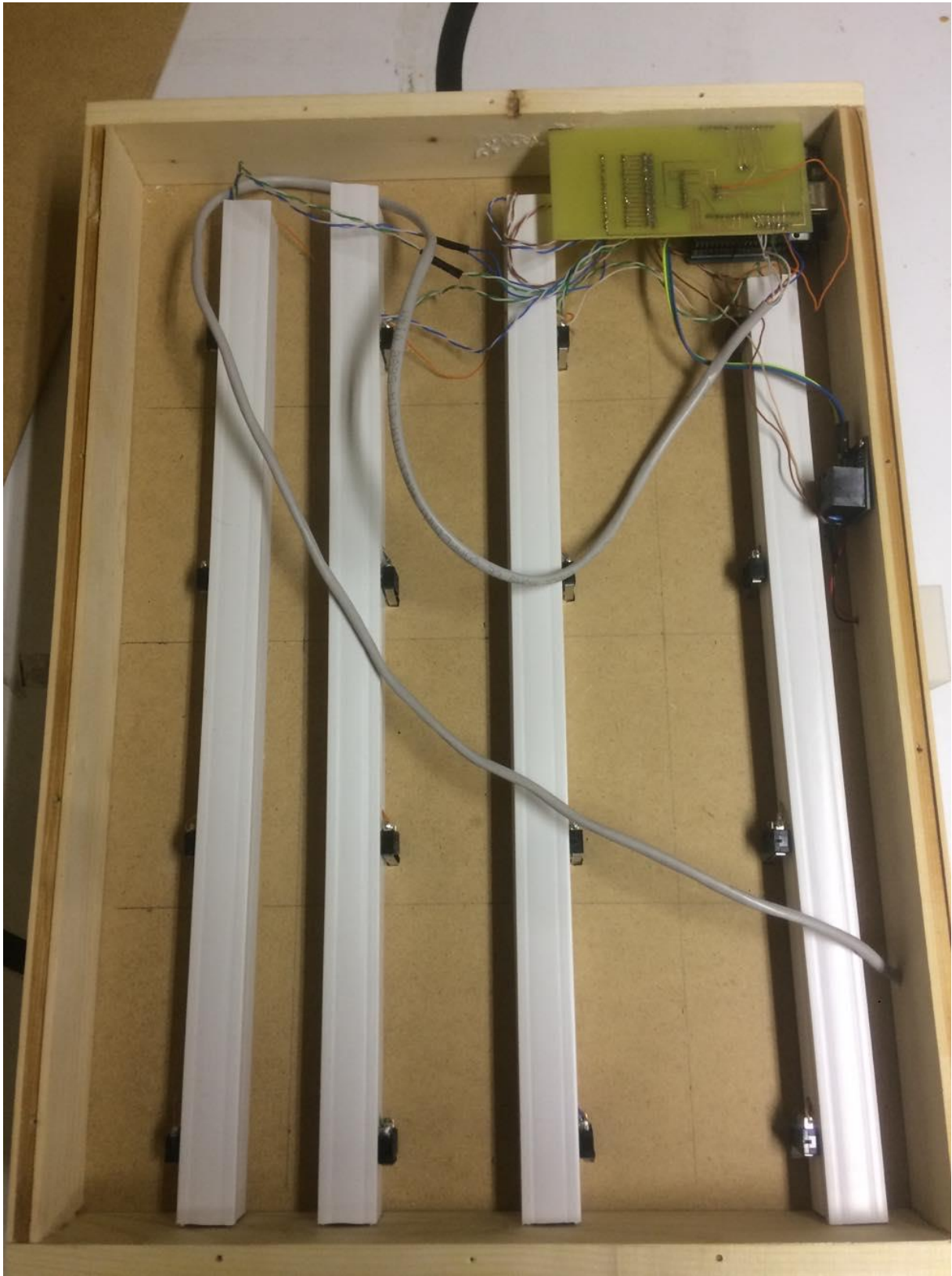
Picture 3 PCB unit

Attachment C - Closet of the project



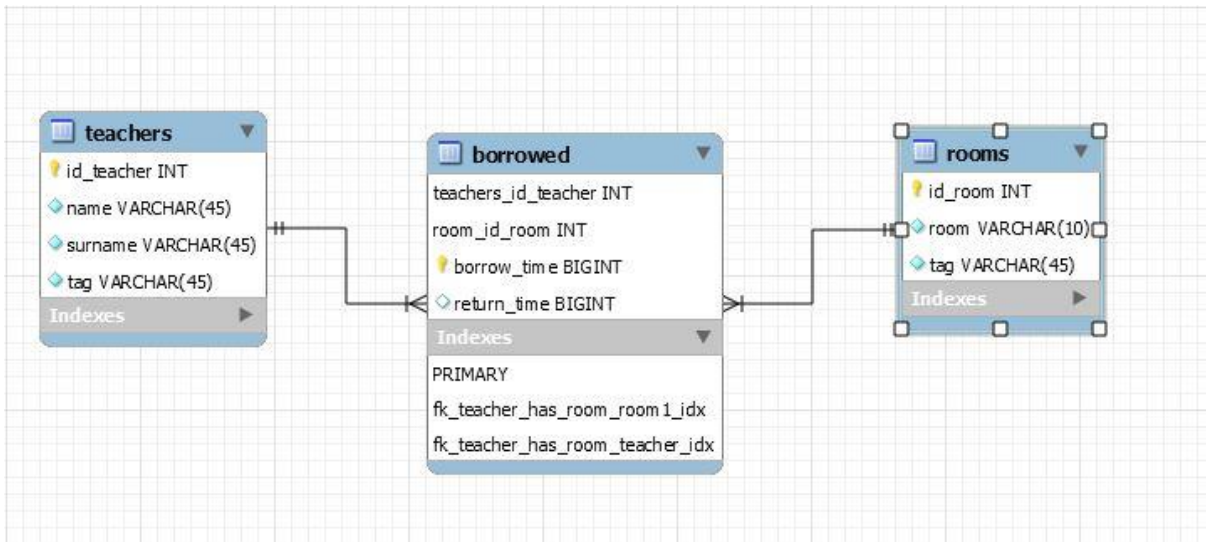
Picture 4 Closet of the project for storing the keys

Attachment D - Inside of the project closet




Picture 5 Wiring of the system


Attachment E - Logical model of the MySQL database



Picture 6 Logical database model

Attachment F- Main website view

**KEY BORROWING SYSTEM**You are logged in as **testuser**



All borrowings

New recordNew return

ID	Room	Teacher	Borrowed	Returned
1	30	Repovsky Radovan	04.03.2019 14:55:03	Not yet
2	30	Repovsky Radovan	28.02.2019 13:20:00	28.02.2019 13:20:08
3	64a	Repovsky Radovan	28.02.2019 12:45:24	28.02.2019 12:45:36
4	7	Repovsky Radovan	28.02.2019 12:34:26	28.02.2019 12:34:42
5	29	Copko Michal	28.02.2019 11:22:03	28.02.2019 12:42:16
6	7	Hrivnakova Maria	28.02.2019 10:01:52	28.02.2019 12:32:43
7	64a	Dolinska Iveta	28.02.2019 09:59:28	28.02.2019 12:36:20
8	30	Repovsky Radovan	28.02.2019 09:29:37	28.02.2019 13:19:25

Picture 7 Main Web Output screen

Attachment G - Main file structure



Picture 8 Main file structure



Stredná priemyselná škola elektrotechnická, Košice, SLOVAKIA
Zespół Szkół Technicznych, Mikołowie, POLAND

LIXIE CLOCK



Co-funded by the
Erasmus+ Programme
of the European Union

1 Introduction

My project is about Lixie clock. This clock is modern version of Nixie clock. They can show you time, date or temperature. But our clock show only time. When you want to have a modern home, you must have this clock. And why I chose this project? I wanted meet new people, visit new place, learn something new. And I like things like this clock – which are lighting and which are beautiful.

2 Theoretical part

Lixie works by shining light through the edge of a clear acrylic. When the light hits an engraving or edge, it scatters in a different direction, and lands in your eye! This allows the flat surfaces of the acrylic to stay clear, and have only edges light up! Now put ten pieces in a stack, each marked with a different number 0-9, and you can show each number individually, on a clear surface!

2.1 Market analysis

You can buy lixie clock on the internet (cca 40\$). They have done, so you don't need to do anything. But in my opinion it is too expensive. When you do it yourself, you spend some time with work on this, but it is cheaper. And when you do it yourself, you must understand what you do, so if they don't work you can repair it. My project is very different from existing clocks on the internet. The lixie clock on the internet look like this picture:



Our clock is on other picture:



Our clocks are bigger and they don't have seconds. PLEL in the middle is lighting like seconds. The box under numbers is wooden. When you buy lixie clock from internet shop, your clock doesn't have wooden box. The box is plastic.

2.2 Technical analysis

Lixie works by shining light through the edge of a clear acrylic. When the light hits an engraving or edge, it scatters in a different direction, and lands in your eye! This allows the flat surfaces of the acrylic to stay clear, and have only edges light up! Now put ten pieces in a stack, each marked with a different number 0-9, and you can show each number individually, on a clear surface!

To control the Lixie Clock we used a rather common ATmega32 microprocessor.

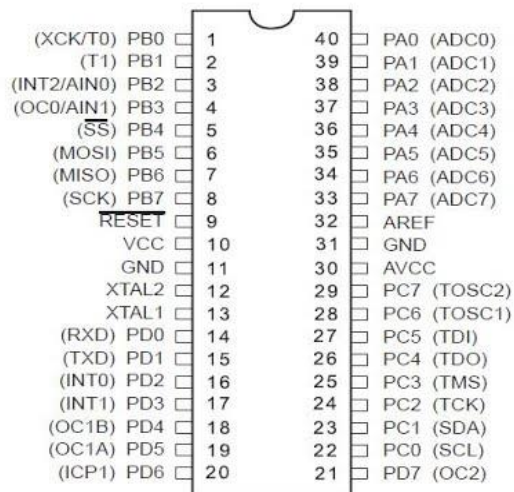
It sends a signal in the A, B, C, D code to then separate coders make it signals number 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 which causes the given plate to be illuminated.

A clipping from the Atmega32 Atmel system catalog, which was chosen for the performance of our task due to its parameters.

2. Configuration Summary

Features	ATmega32A
Pin count	32
Flash (KB)	32
SRAM (KB)	2
EEPROM (KB)	1
General Purpose I/O pins	23
SPI	1
TWI (I ² C)	1
USART	1
ADC	10-bit, up to 76.9ksps (15ksps at max resolution)
ADC channels	8
AC propagation delay	Typ 400ns
8-bit Timer/Counters	2
16-bit Timer/Counters	1
PWM channels	4
RC Oscillator	+/-3%
VREF Bandgap	
Operating voltage	2.7 - 5.5V
Max operating frequency	16MHz
Temperature range	-55°C to +125°C
JTAG	Yes

Drawing showing the microprocessor lead layout. We connected the BCD decoders to 1 out of 10 to the PA, PB and PC ports. The control panel and the RTC module communicating via the I²C bus were connected to the PD port.



A drawing showing the microprocessor layout. It has been programmed in an external programmer with the code written and compiled in the BascomAVR software.

The outputs of the ABCD coder are connected to the outputs PB, PA, and then to the appropriate leds.

The outputs of PC 0-3 are set to: min, h, days, months.

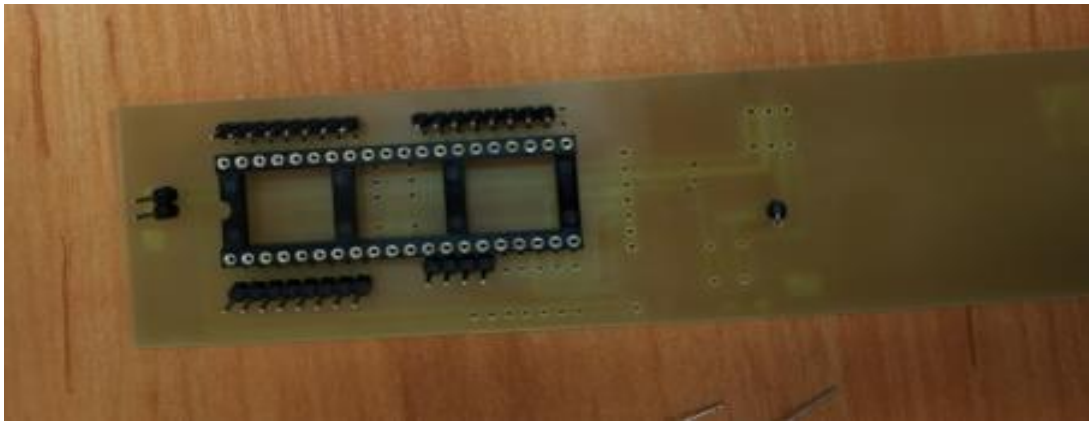
PD6- PD7 outputs are the ability to connect thermometers.

with PD4-PD5 outputs, the time module (DS1307US2) is connected.

outputs from PC4 to PC7 are shown next : . , - , °c

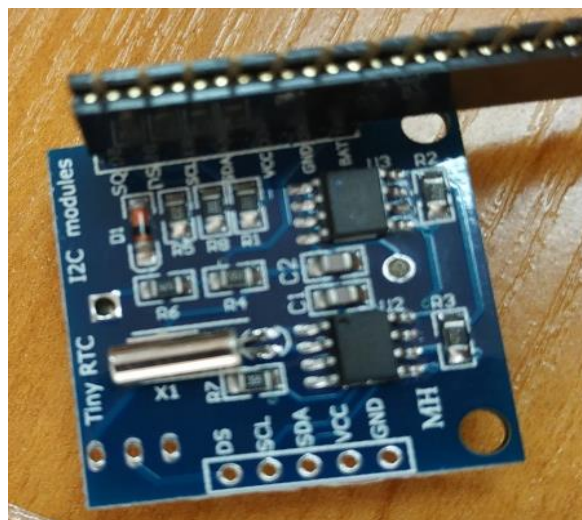
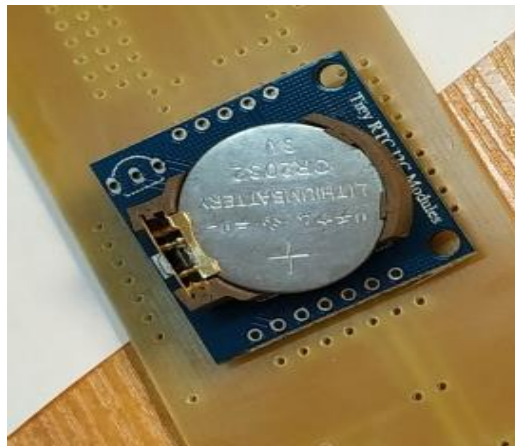


The appearance of the printed circuit board before assembly. The plate was milled and drilled on a CNC milling machine.

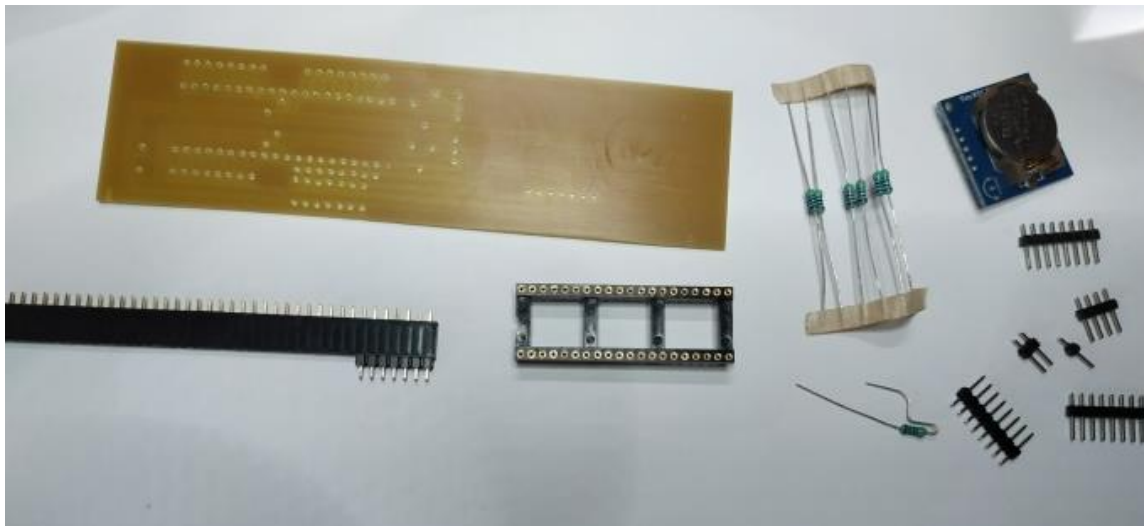


System RTC (Real-time clock)

The appearance of the module with visible battery and layout side. it is necessary to carry out information about time, date, etc. during power supply decay. According to the producer, it is enough for a few years of work of our clock. In addition to the RTC chip, AT24xx chip is available for storing data during main power brag. This gives you another opportunity to expand our construction.

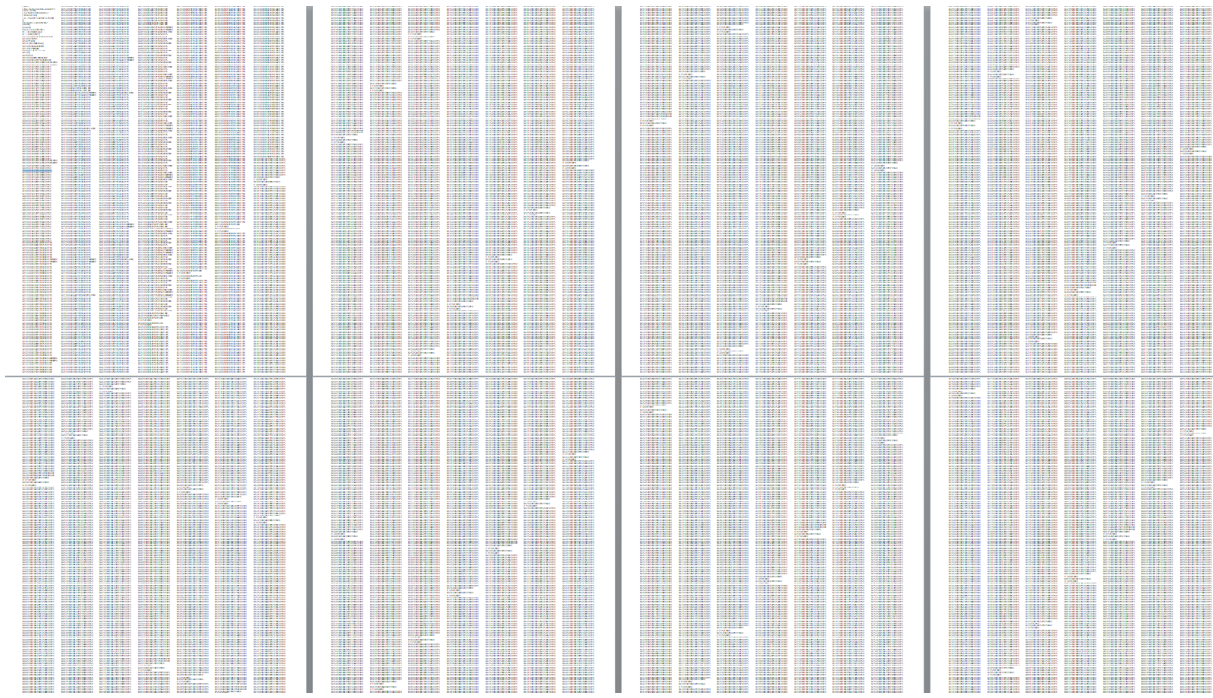


Real-Time Clock RTC - an element of computer systems used for counting down time regardless of the state of the machine (work, blocking, switching off), it is mounted on almost all personal computers, servers and many embedded systems, PLC controllers.

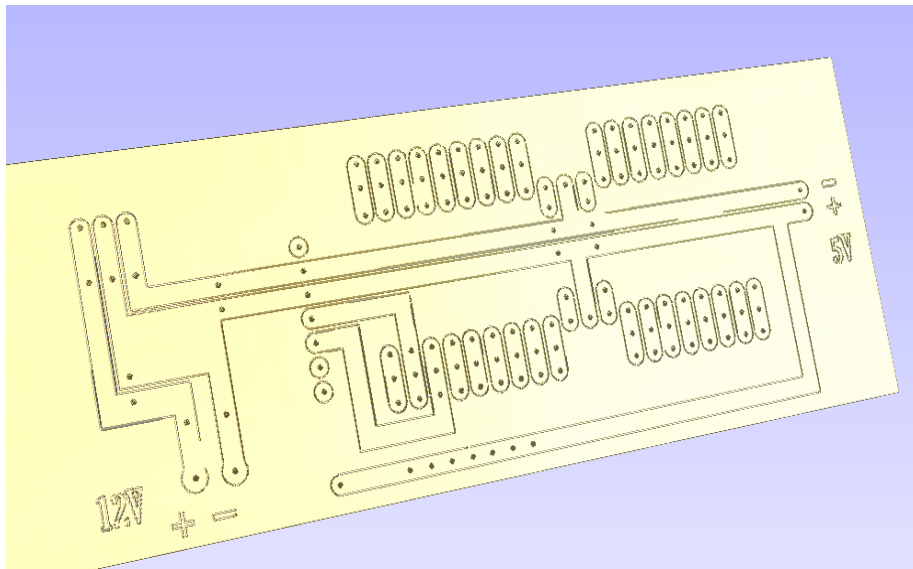


2.3 Used Technologies

Fragment of pcb-gcode.docx



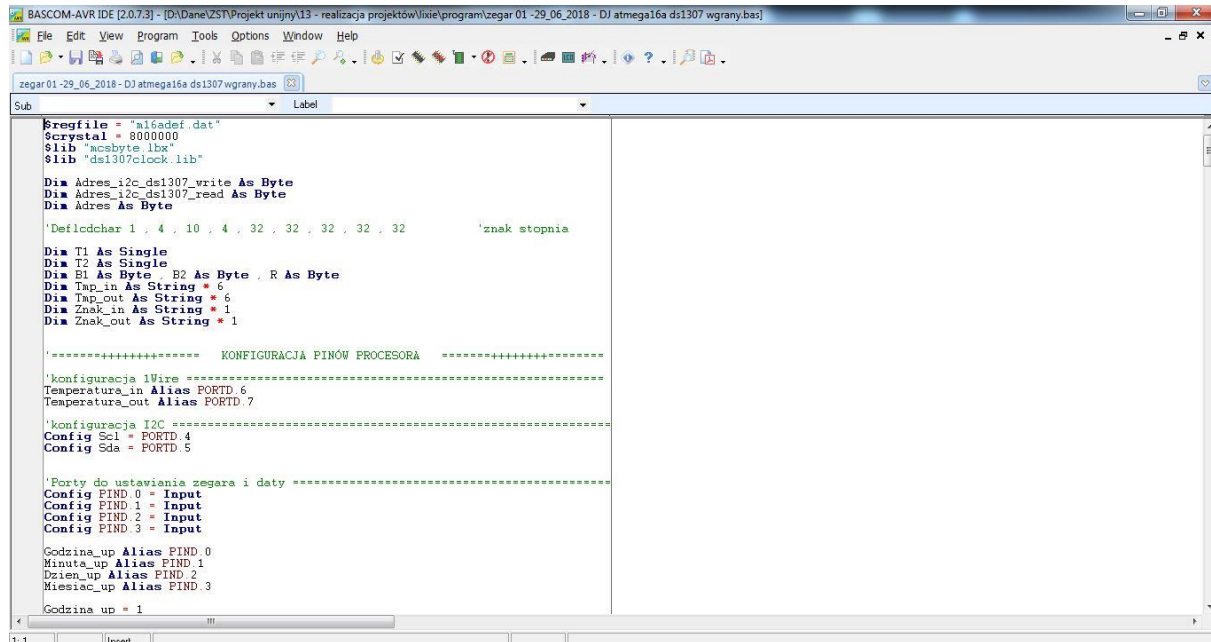
pcb visualization



The board was milled on a CNC machine using a 30 degree milling cutter, the whole milling took about 3 hours and was repeated to improve the quality of the cutter.

After milling the paths, the milling cutter was replaced with a drill bit 0.8mm in diameter, with all holes drilled.

A fragment of the bascom code to control the clock



```
Regfile = "atmega16a.def"
$crystal = 8000000
$lib "acsbtype.lib"
$lib "ds1307clock.lib"

Dim Adres_i2c_ds1307_write As Byte
Dim Adres_i2c_ds1307_read As Byte
Dim Adres As Byte

'Definidchar 1, 4, 10, 4, 32, 32, 32, 32, 32 'znak stopnia

Dim T1 As Single
Dim T2 As Single
Dim B1 As Byte, B2 As Byte, R As Byte
Dim Tap_in As String * 6
Dim Tap_out As String * 6
Dim Znak_in As String * 1
Dim Znak_out As String * 1

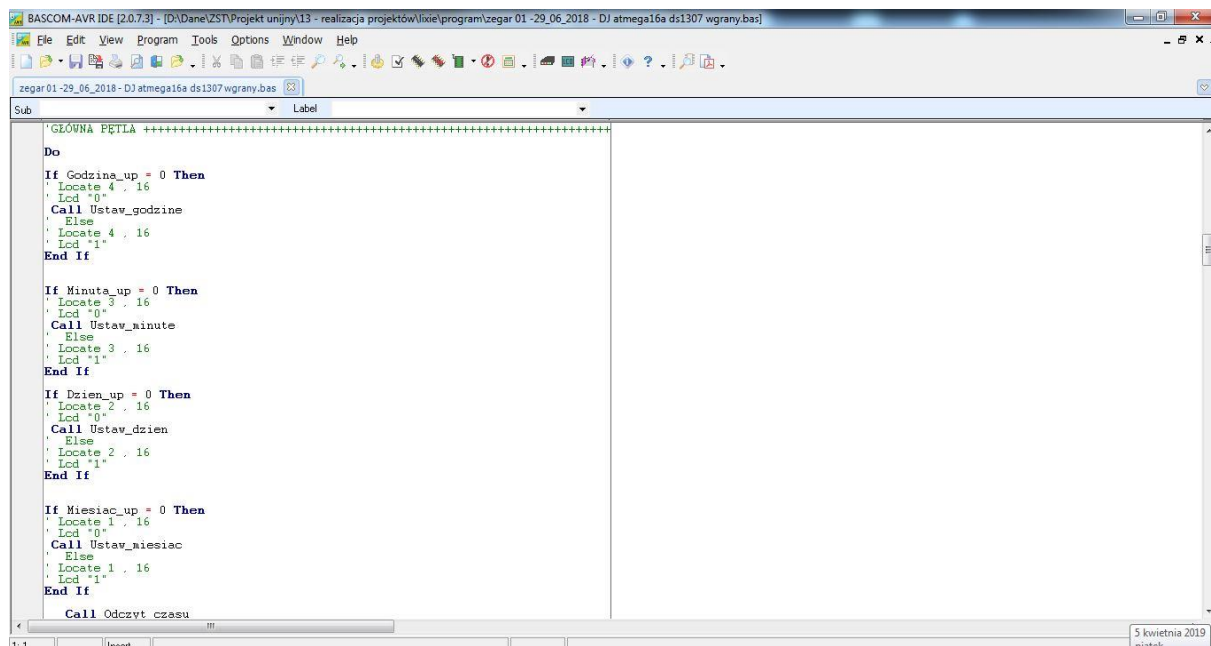
'===== KONFIGURACJA PINÓW PROCESORA =====
'konfiguracja I2Wire =====
Temperature_in Alias PORTD.6
Temperature_out Alias PORTD.7

'konfiguracja I2C =====
Config Scl = PORTD.4
Config Sda = PORTD.5

'Porty do ustawienia zegara i daty =====
Config PIND.0 = Input
Config PIND.1 = Input
Config PIND.2 = Input
Config PIND.3 = Input

Godzina_up Alias PIND.0
Minuta_up Alias PIND.1
Dzien_up Alias PIND.2
Miesiac_up Alias PIND.3

Godzina_up = 1
```



```
'GŁÓWNA PETLA ++++++
Do
If Godzina_up = 0 Then
  Locate 4, 16
  Lcd "0"
  Call Ustaw_godzine
Else
  Locate 4, 16
  Lcd "1"
End If

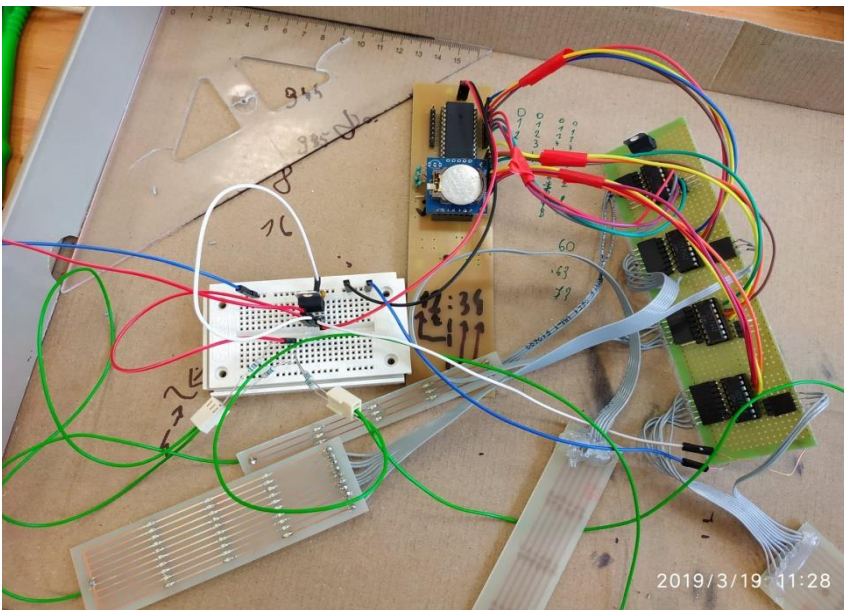
If Minuta_up = 0 Then
  Locate 3, 16
  Lcd "0"
  Call Ustaw_minute
Else
  Locate 3, 16
  Lcd "1"
End If

If Dzien_up = 0 Then
  Locate 2, 16
  Lcd "0"
  Call Ustaw_dzien
Else
  Locate 2, 16
  Lcd "1"
End If

If Miesiac_up = 0 Then
  Locate 1, 16
  Lcd "0"
  Call Ustaw_miesiac
Else
  Locate 1, 16
  Lcd "1"
End If

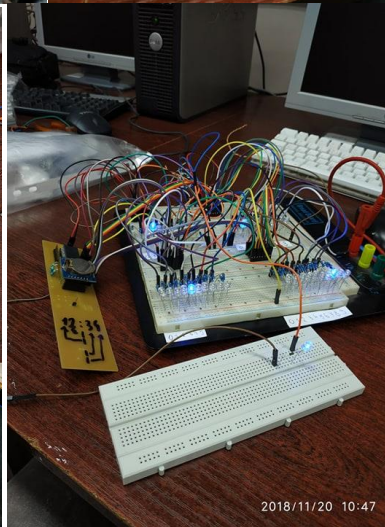
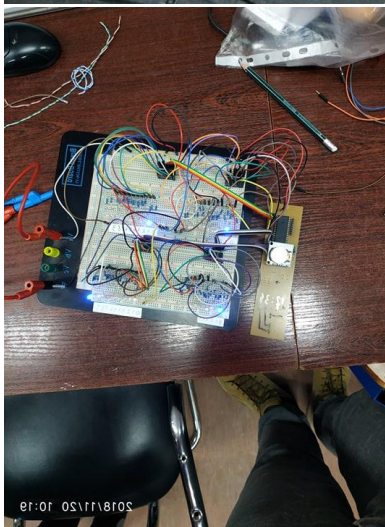
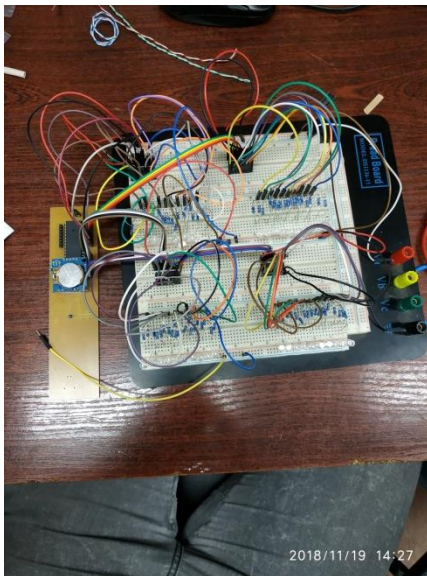
Call Odczyt_czasu
```

After installing, assembling and connecting elements, we connected the system to the LCD display to check the correctness of connections and program operation.



in the picture next you can see the connected control boards and encoders along with the LEDs illuminating the plates with numbers

At the joint meeting, together with my project participant, I joined and put a stop to the arrangement on the LEDs



the program worked correctly and allows setting hours and minutes.

then all the encoders from the large plate were transferred to one small plate

2.4 Financial Analysis

I was writing about this in 2.1. In my opinion this clock is more expensive, when you buy it on internet. The most expensive were the numbers, but if you can do the numbers yourself, it is cost only your time. The other things are very cheap. The LEDs and the resistors are cost a few cents for example.

3 Practical part

3.1 Electronics

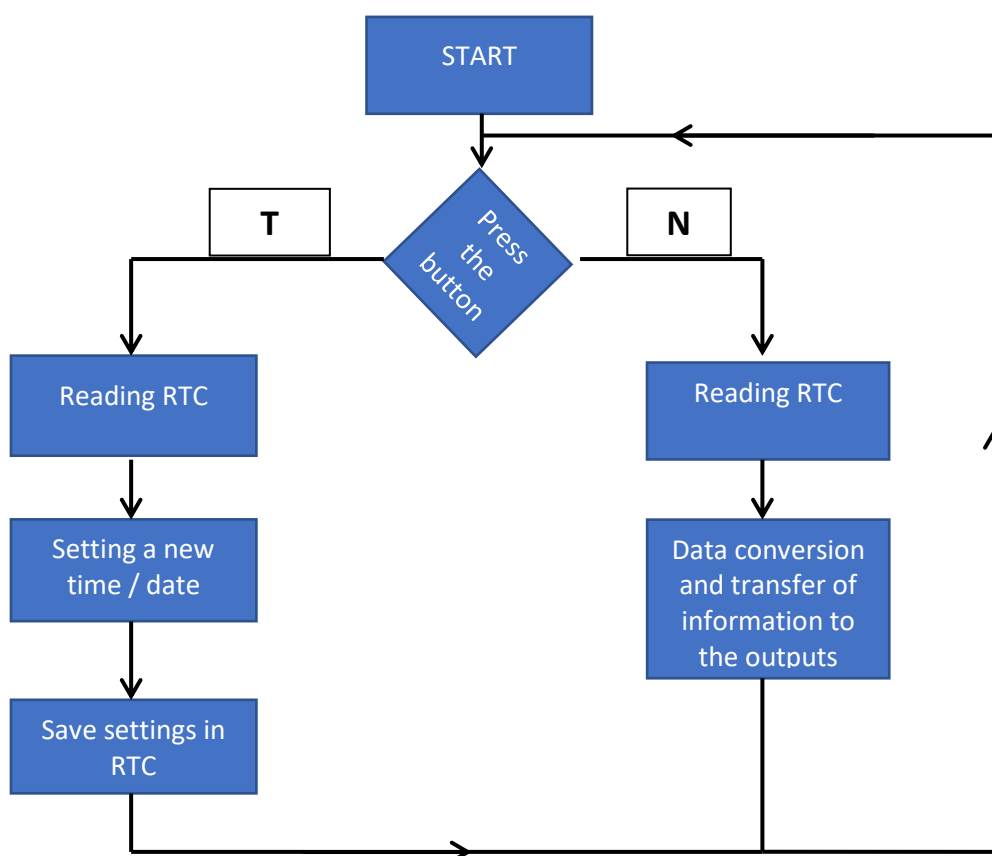
We used 7 PCB – 5 numbers and 2 for control time. First is for ATmega and battery. The second is for encoders, which connect the numbers with ATmega. The others PCB are for the numbers. There are LEDs, which lighting under numbers. Every number have 3 LEDs.

Our clock don't sense values from environment, because we have only time. But if we program it for temperature too, they will by sense temperature from enviroment.

Our clock is powered with 230V, but ATmega need battery too.

3.2 Programs

The program code works as follows:



3.3 Mechanical

The numbers did for us the company from Košice. But the box, where are the numbers and other things did we. We used white wooden desk. We cut it on 2 pieces. We put the smaller desk on bigger desk. In bigger desk is one big hole, where are PCBs with LEDs. In smaller desk are 5 holes for plate with numbers. In every hole are 10 plates. The holes are enough big for these 10 plates and they don't fall. The other PCBs are behind it in small black box.

3.4 Testing

When we was testing our clock for first time (pictures in 2.3 in the end), every LEDs was lighting and LEDs, which showed time wasn't lighting. So we change polarity of LEDs and then it was lighting correctly. When we tested it in last week, when we worked on this, we have more problems. For example, the clock didn't work correctly and the reason was the something was bad connected. So we did must detect where is problem and repair it. Then the seconds was too fast and minutes was changes every cca 15 seconds. So we did must repair it too.

4 Conclusion

This project is successful, I think. We did work on it to the end. In the last morning I was thinking we didn't have enough time for finish it but we do it. So its successful for me. This project learned me a lot of things. For first time I was working on something with somebody, who isn't from my country. I am not very good in English and my partner too, so sometimes it was hard to understood each other. But it was very good experience. I met new very good people. I learned don't give up.

I am planning continue working on this project. I want to edit it because I want to make it more beautiful and better.

5 References

<https://www.tindie.com/products/connornishijima/lixie-an-led-alternative-to-the-nixie-tube/>

https://www.mcselec.com/index.php?id=14&option=com_content&task=view

<http://www.alldatasheet.com>



Střední průmyslová škola elektrotechnická, Havířov, CZECH REPUBLIC
Zespół Szkół Technicznych, Mikołów, POLAND

RFID LOCK



Co-funded by the
Erasmus+ Programme
of the European Union

Contents

1	Introduction	1
2	Theoretical part	2
2.1	Market analysis	2
2.2	Technical analysis	3
2.3	Used Technologies	5
2.4	Financial Analysis	9
3	Practical part	6
3.1	Electronics	6
3.2	Programs	8
3.3	Mechanical	9
3.4	Testing	9
4	Conclusion	10
5	References	11
6	Attachments	12

1 Introduction

In this project we had to create locking mechanism using RFID technology, which uses electromagnetic field to identify objects. This way we can uniquely identify users and grant access ex. to room or not basing on user.

RFID lock is simple mechanism that releases electromagnet in electromagnetic lock which allows to open doors after RFID tag gets close to RFID reader, if the tag is entered into system.

That means, that everyone can implement this technique of accessing rooms or houses in cheap and easy way if one knows electronics and programming.

We've chosen this project, because it is practical – we can use it in our lives with experience we gain in the project and because it uses quite new technology involving touchless access to different places.

2 Theoretical part

2.1 Market analysis

In the market you can find many similar solutions. It is because the technology to open locks by cards or tags is so popular. Most of the entry doors to blocks of flats use that technology.

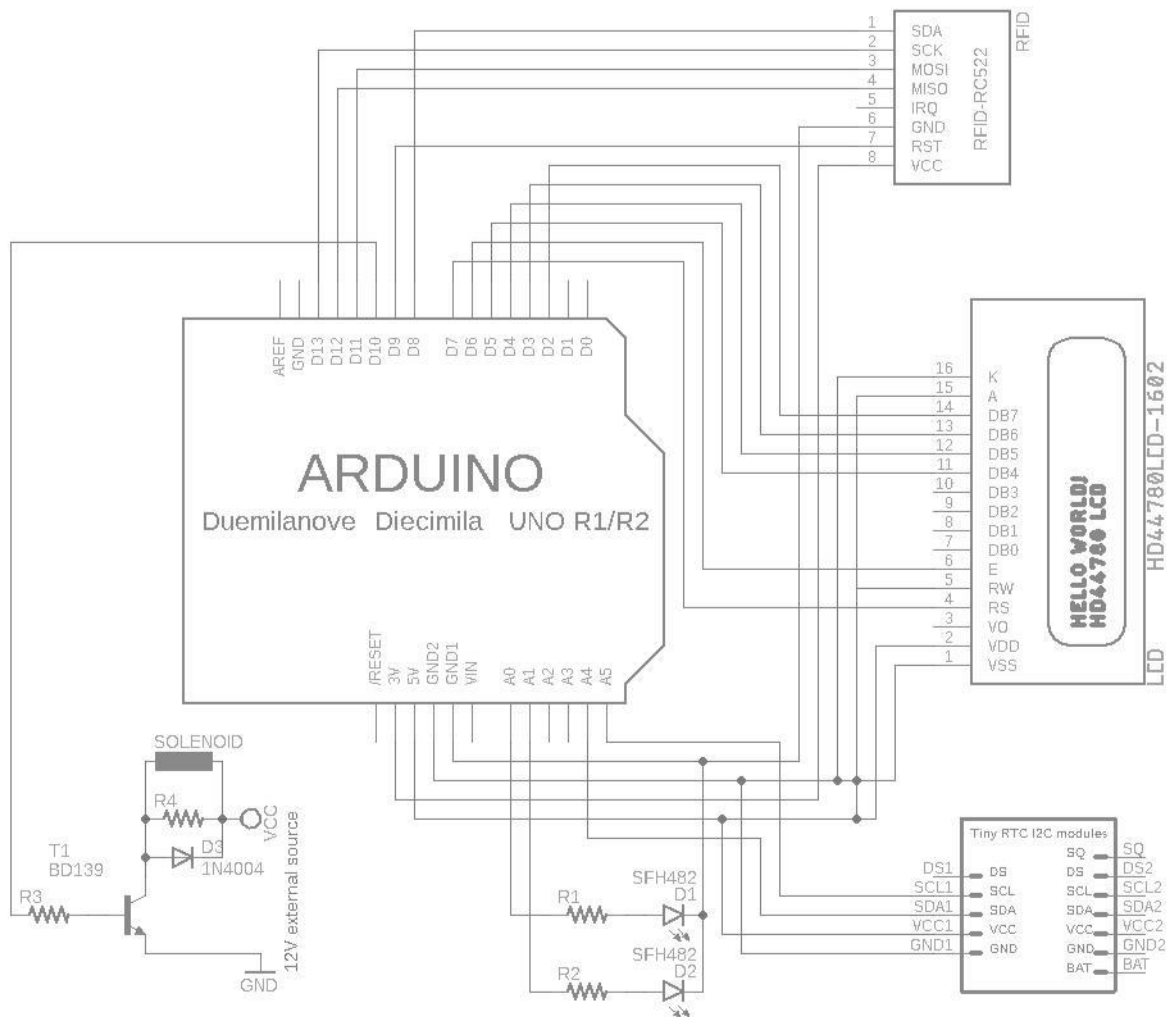
The disadvantages of our project are of course the price, and size. As we haven't got any special machines to mass produce the electronic parts and assembly machines our solution is on the bigger side. The price also comes through the quantity of essential components. As a single component the price is much higher than a mass-producing company would buy.

The advantage is that you can add a new tag or card with only using a "Master card" (special card only for adding or removing tags or cards). Another advantage is that we include LCD display and you can see what you are supposed to do.

2.2 Technical analysis

The project shut work as normal lock that you can find in entrance doors to block of flats or offices. The function is you use your card or tag and the door will open. We improve this with the possibility of selecting one card and by that card you can easily manage other cards (adding/removing). This system we designed for few users like from 1 – 20 users. That is because adding like 200 users will be uncomfortable.

Our project consists of μ processor Arduino Uno, LCD display 20*4 characters, RFID reader (RFID-RC522), real time clock (Tiny RTC I2C module), solenoid (12V), external power supply (12V), bipolar transistor NPN (BD139) and LED diode indication.



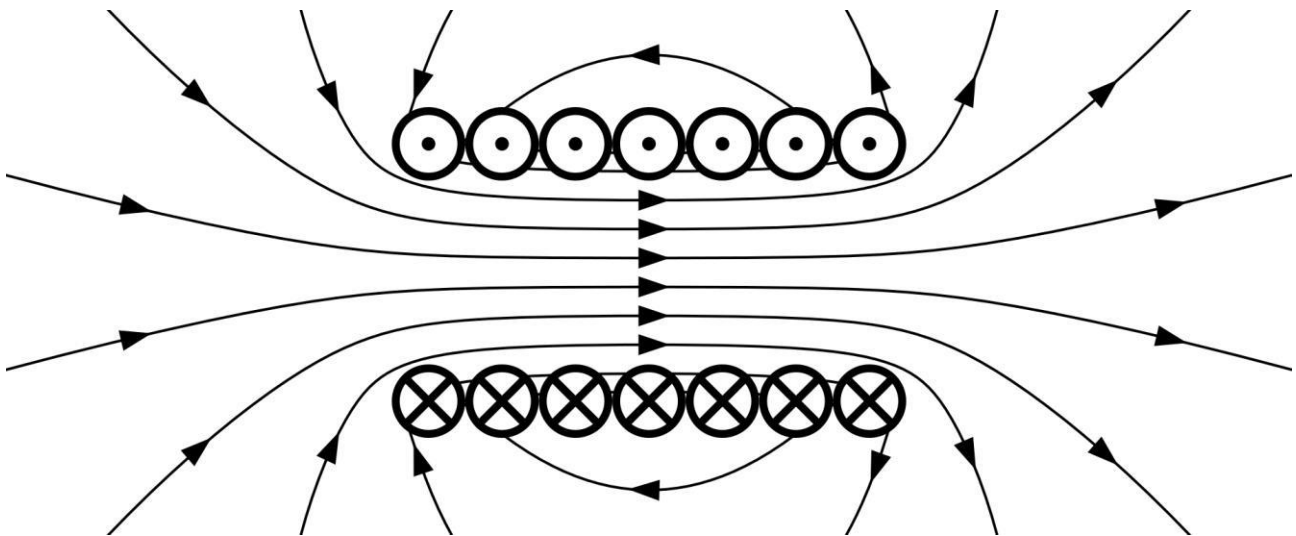
Solenoid is the generic term for a coil of wire used as an electromagnet. It also refers to any device that converts electrical energy to mechanical energy using a solenoid. The device creates a magnetic field from electric current and uses the magnetic field to create linear motion. Common applications of solenoids are to power a switch, like the starter in an automobile, or a valve, such as in a sprinkler system.

How a Solenoid Works

A solenoid is a coil of wire in a corkscrew shape wrapped around a piston, often made of iron. As in all electromagnets, a magnetic field is created when an electric current passes through the wire.

Electromagnets have an advantage over permanent magnets in that they can be switched on and off by the application or removal of the electric current, which is what makes them useful as switches and valves and allows them to be entirely automated.

Like all magnets, the magnetic field of an activated solenoid has positive and negative poles that will attract or repel material sensitive to magnets. In a solenoid, the electromagnetic field causes the piston to either move backward or forward, which is how motion is created by a solenoid coil.



Reference 1

2.3 Used Technologies

We've used Arduino Uno with RFID reader and real time clock. On the programming side we had to use C language used by Arduino and a few libraries: RTC library for real time clock, SPI (Serial Peripheral Interface) library for communication, MFRC522 library for RFID reader and Liquid Crystal for handling LCD. Because of the type of Arduino, we used, it was challenging to create compact and fully functioning program, but after two meetings software and hardware part worked just fine.

2.4 Financial Analysis

SWOT Analysis

Strengths

- Our project is very compact
- Simple but effective solution
- Unique adding/removing by "Master card "

Weaknesses

- Cost per unit
- Possible low production rate

Opportunities

- Future improvement
- Expansion

Threats

- Bigger cost than companies from China
- Finger print scanner instead of RFID

Comparison

I will compare our project to **AGPtEK RFID Home Security Kit** that I find on Amazon (reference no. 3).

Advantages of our solution:

- Cheaper
- You can have more than 10 tags
- More compact

Disadvantages of our solution:

- Less secure
- More time to prepare the lock (building and programing process)

3 Practical part

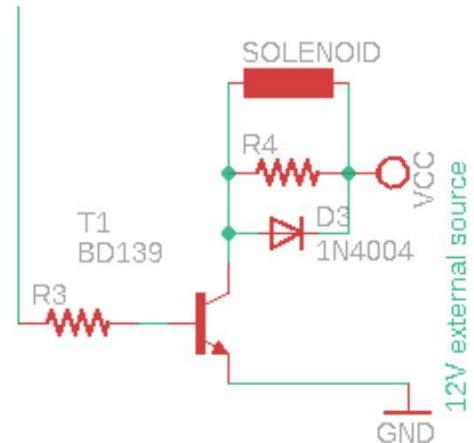
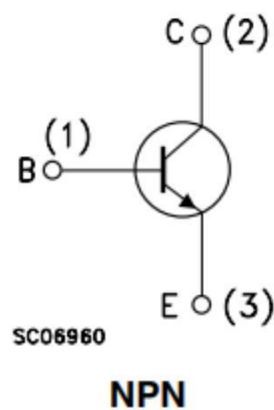
3.1 Electronics

Transistor as a Switch

The transistor is used because the solenoid is 12V and the Arduino can't have 12V output. In our project we use transistor to switch on/off the solenoid. For this purpose, we use BD193. It is bipolar transistor type NPN. When logical 1 is the output of the Arduino the transistor is saturated, and the lock can be opened. When logical 0 is the output of the Arduino the transistor is cut-off and the lock can't be opened.



Reference 4



LCD display

For communication with user we decided to use basic liquid-crystal display.

DB4-DB7: Those are data bus line.

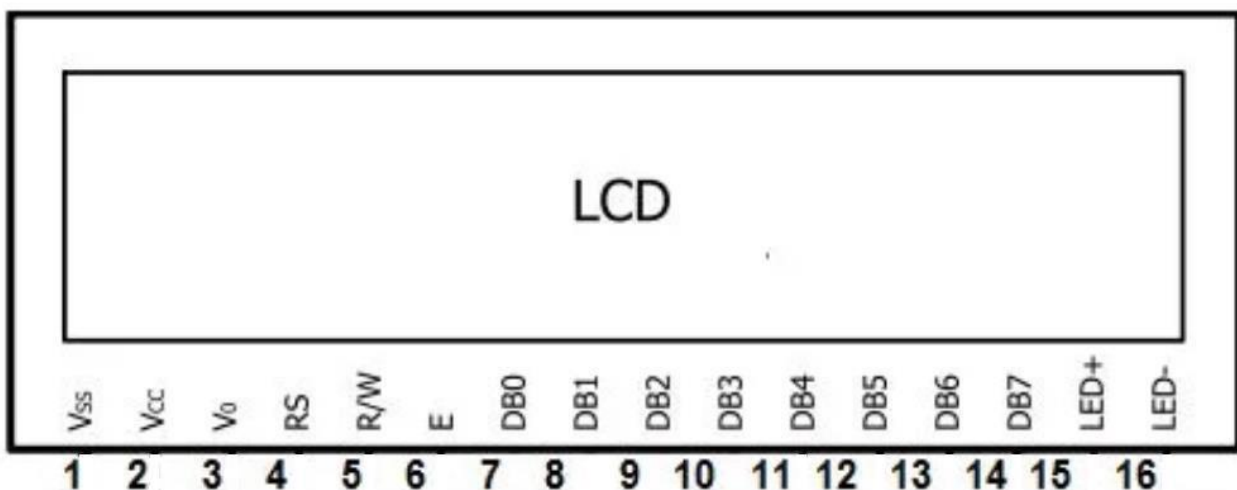
Pin LED+ and LED: Those are used for lighting the display up.

E: Chip enable signal

6 EH H->L Chip enable

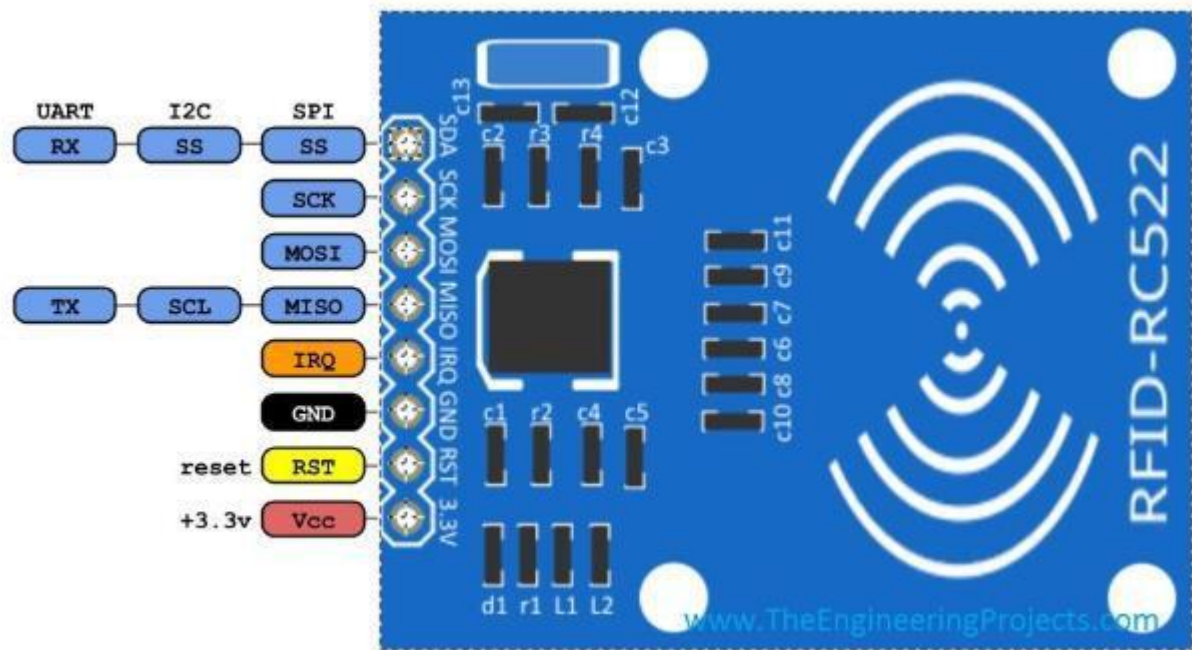
signal RS: Instruction code

R/W: Read/Write mode



RFID RC 552

Main purpose of our project is to open lock with RFID cards/tags, so we must have some RFID reader. Radio-Frequency Identification (RFID) is normally used to read information stored on special cards/tags. Tags can be read over some thin objects. That is useful for us because we can hide it in our case with all electronic parts. Our reader Reads / Writes at the clock 13.56MHz. It uses SPI protocol to communicate with Arduino. The benefits of this reader are low cost and low voltage. It only uses 3.3V from Arduino.



Reference 5

3.2 Programs

After turning the system on, we are asked to add a Master Tag – RFID tag, which will be administrating all other tags. When it gets close to reader, system switches to programming mode and expects new tag to get close to reader. After that happens, newly registered tag will be granted access after it gets close to reader. If the system is in programming mode and already registered tag gets close to reader, it will be unregistered and won't be able to access what's behind doors anymore, unless administrator registers it again.

Out of programming mode, program listens for new tags getting close to reader and when that happens, it looks for tag's unique identifier in an array and if it gets found, access is granted and green LED lights up, else it is denied and red LED lights up then. This way, we don't store any data outside the Arduino's internal memory as it is hard to develop program which won't overflow available program memory of Arduino Uno.

We also used Real Time Clock, which allows us to show actual date and time on the display, when the system listens to any incoming tags and nothing happens. If the project gets developed further, after switching to bigger Arduino, it is possible to store information about users and history of grants and denies of access with timestamp.

Flowchart (Attachment 2)

3.3 Mechanical

We've created simple door model using wooden plank and wooden post (simple blueprint can be found in attachment 1). Inside jamb created from post there is place, where electromagnet is mounted and jammed. Latch has been made of spring, small screw and small steel washer keeping it all together.

3.4 Testing

At the beginning, we were testing only if the program works correctly – we weren't using electromagnetic lock, just LCD, LEDs and of course RFID reader. When we started with the code, there were some small mistakes, that we had to find, so we tested everything line after line. After hours of testing, debugging and correcting the code, all started to work as we wanted since the beginning.

In the next meeting session, we focused on making the circuit for electromagnet and testing it. We had to power electromagnet up with 12 V power supply, so we had to use transistors and relay the opening signal from Arduino to electromagnet as 12 V. After that was working fine, all we had to do was recreate the circuit, as using contact plate is not very durable, and mount everything together – put the electromagnet and box with electronics in place.

4 Conclusion

We find the project successful and we have learned a lot working together.

Patryk Bojczuk:

I've learned more about electronics and got to know C language, which is used to program Arduino Uno. I think, in the future project can be extended, using Arduino with more memory, we could create access logging (ex. if someone gets access to the door or not, it will be logged in file with current timestamp and RFID tag's unique identifier) and possibility to open the door via phone application or web page.

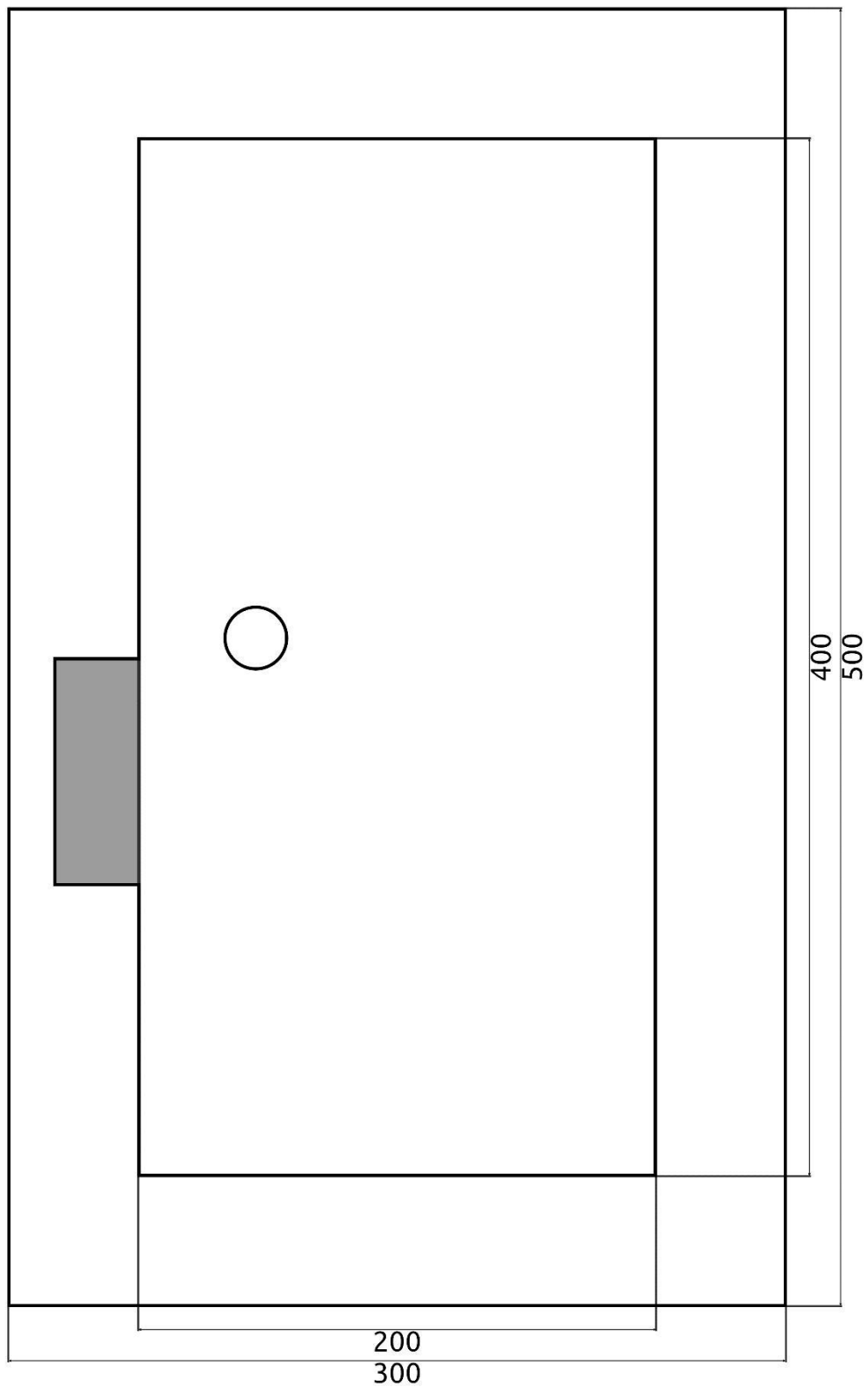
I think that our teamwork has been successful, we split a work between us both, so we had been working most of the time. The big limitation was a time. We must have some changes in design because of the time. I think

5 References

1. <https://en.wikipedia.org/wiki/Solenoid>
2. <https://sciencing.com/a-solenoid-work-4567178.html>
3. https://www.amazon.com/Access-Control-Security-Electromagnetic-Proximity/dp/B071RRHQR3/ref=sr_1_8?keywords=rfid%2Bdoor%2Block&qid=1554133998&s=gate&sr=8-8&th=1
4. <https://www.st.com/resource/en/datasheet/cd00001225.pdf>
5. <https://www.theengineeringprojects.com/2015/08/interfacing-rfid-rc522-arduino.html>

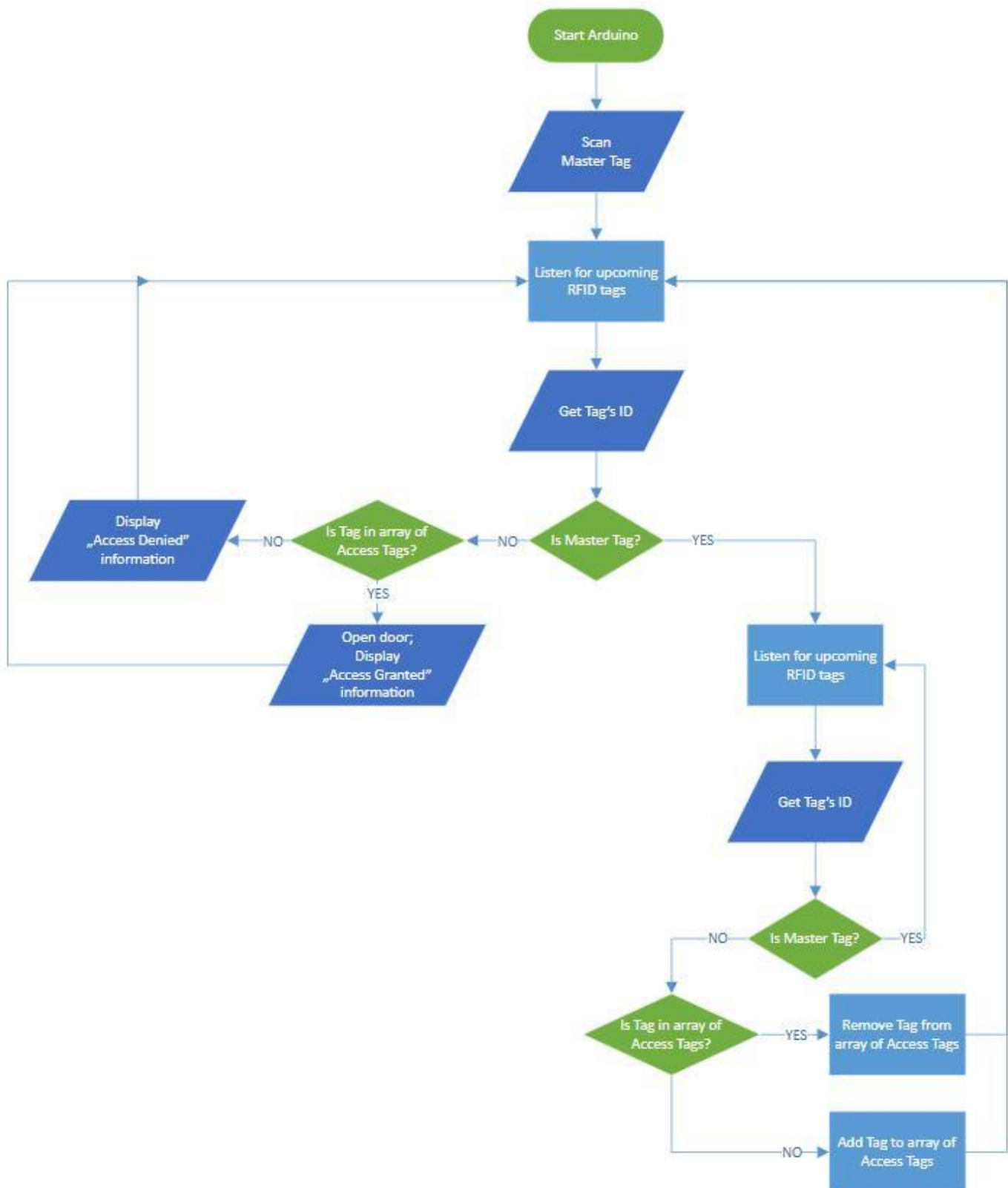
6 Attachments

6.1 Attachment 1



Grey area is where electromagnet is placed, and circle is a simple handle.

6.2 Attachment 2





Zespół szkół technicznych, Mikolow, POLAND
Srednja poklicna in tehniška šola, Murska Sobota, SLOVENIA

SOLAR TRACKER



Co-funded by the
Erasmus+ Programme
of the European Union

Contents

Contents	2
1 Introduction	1
2 Theoretical part	2
2.1 Market analysis	2
2.2 Technical analysis	3
2.3 Used Technologies	7
2.4 Financial Analysis	8
3 Practical part	10
3.1 Electronics	10
3.2 Program	14
3.3 Testing	15
4 Conclusion	17
5 Sources	19
5.1 Pictures	19

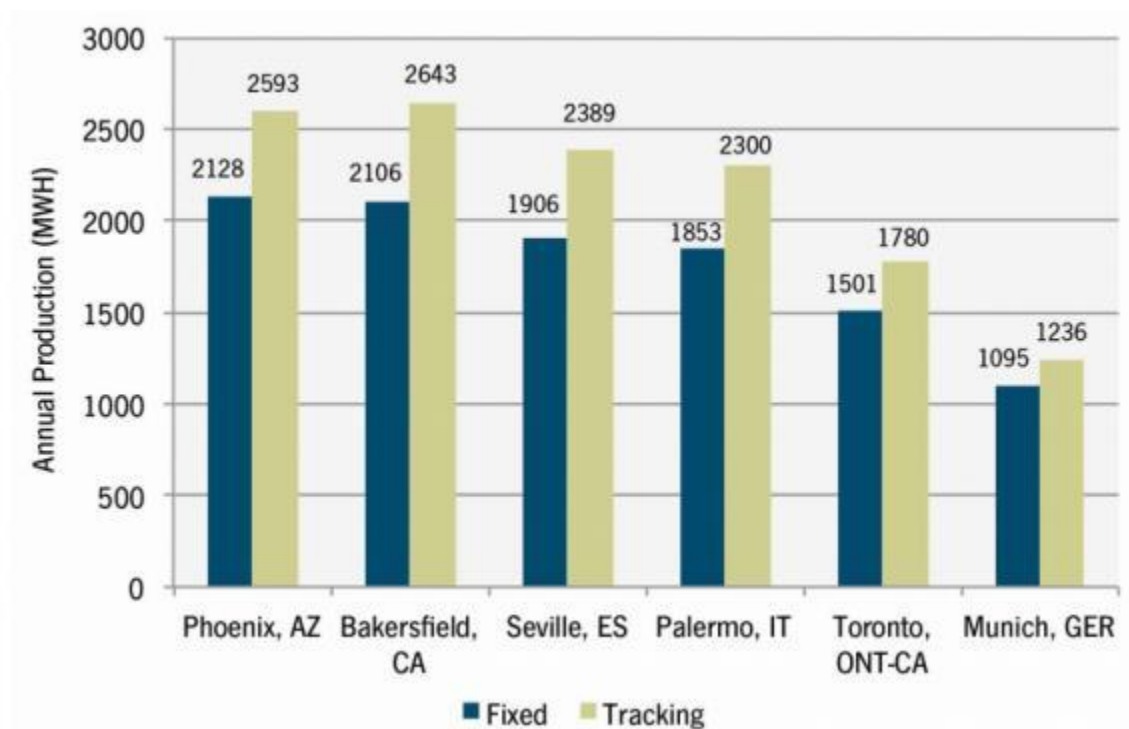
Pictures

Picture 1: Solar tracker efficiency graph	1
Picture 2: Solar trackers	2
Picture 3: 2 Axis Solar tracker	2
Picture 4: Arduino Mega	3
Picture 5: SG90 Servo motor	4
Picture 6: Photoresistor	4
Picture 7: MLT-BT05 Bluetooth module	5
Picture 9: Blynk app page 1	5
Picture 8: Blynk app page 2	5
Picture 10: Arduino Mega shield	6
Picture 11: Solar tracker in progress	7
Picture 12: Printing the main base	8
Picture 13: Photoresistors in the tracker	10
Picture 14: LDR schematic	10
Picture 16: Bluetooth module on the shield	10
Picture 15: Bluetooth module schematic	10
Picture 17: Servos and gears from the side	11
Picture 18: Servos and gear from the front	11
Picture 19: Servos and gears from the top	11
Picture 20: Servo schematic	11
Picture 21: Our shield for all the connections	12
Picture 22: Shield schematic	12
Picture 23: Case from the front side	13
Picture 24: Case from the back side	13
Picture 25: The main part of the program	14
Picture 26: Final product	16

1 Introduction

The project that we chose is a solar tracker. We both chose the project for the same reason and that reason is the fact that we are both interested in renewable sources of energy. The main benefit of solar trackers is significantly higher performance of solar panels. In a world that gets more polluted day by day this can be very important. More efficient solar panels mean that we can fit more solar panels on a smaller surface or fit the same amount of solar panels on the same surface but instead generate more power. The world is moving more and more towards renewable sources of energy and making all sources of renewable energy more efficient is a crucial step towards a cleaner, pollution free future.

As we can see on the graph below, a solar panel that is mounted on some kind of a solar tracker (in this case biaxial servo system) generates a notably higher amount of electricity, thus making it more efficient.



Picture 1: Solar tracker efficiency graph

2 Theoretical part

2.1 Market analysis

There are a lot of different versions of this project on the current market. They are mostly different in dimensions, amount of produced energy, ways to follow the sun, and finally of course the cost. Solar panels with solar tracking capabilities are definitely better than solar panels without any kind of tracking, because the ability to track the sun maximizes its potential of produced energy. It can, and it is normally used in renewable ways to produce energy.



Picture 2: Solar trackers

For example, let's look at a tracker that can be bought, Solar Tracker 2-axis ST44M2V3P.



Picture 3: 2 Axis Solar tracker

It has an astronomic tracking algorithm with precision of 0,5 degrees; all types of CPV usage including round parabolic, but mainly attended for 3 flat solar panels; real time clock & date; setting, monitoring, upgrading, controlling & driving via PC and more.

2.2 Technical analysis

Our project should be 3D-printed in as good quality as possible, and then be put up all together with the rest of parts, which are:

- an Arduino (we used Arduino Mega),



Picture 4: Arduino Mega

Microcontroller	Atmega2560	Flash Memory	256 KB of which 8 KB used by bootloader
Operating Voltage	5V	SRAM	8 KB
Input Voltage (recommended)	7-12V	EEPROM	4 KB
Input Voltage (limit)	6-20V	Clock Speed	16 MHz
Digital I/O Pins	54 (of which 15 provide PWM output)	LED_BUILTIN	13
Analog Input Pins	16	Lenght	101.52 mm
DC Current per I/O Pin	20mA	Width	53.3 mm
DC Current for 3.3V Pin	50mA	Weight	37g

- 2 servo motors (SG90),



Height	32 mm
Width	12 mm
Length	32 mm
Speed (sec)	0.1
Torque (kg-cm)	2.5
Weight (g)	14.7
Voltage	4.8 - 6

Picture 5: SG90 Servo motor

- 4 LDRs,



Number of Pins	2
Light Resistance	10-20k Ohm
Dark Resistance	>1M Ohm
Lead Spacing	0.2"

Picture 6: Photoresistor

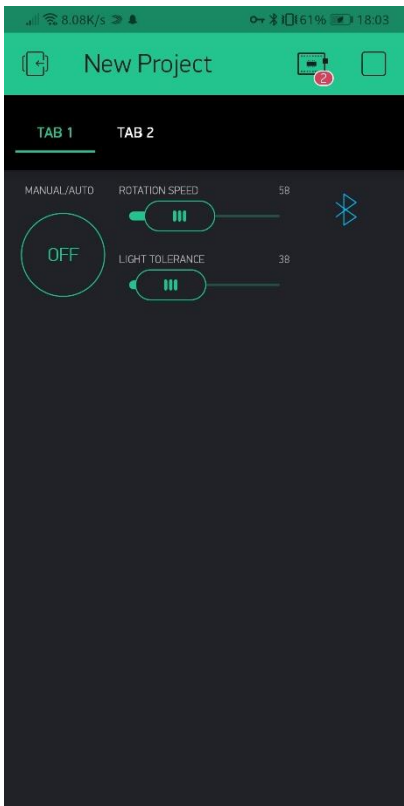
- 4 x 10 kΩ resistor,

- bluetooth module paired with a phone app,

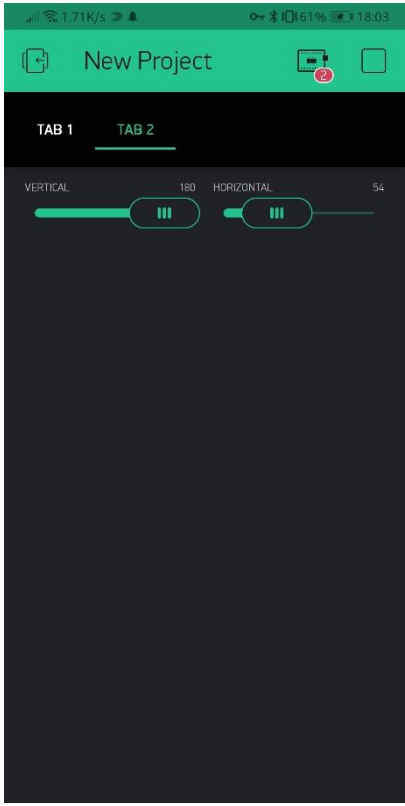


Standby power consumption	90uA-400uA
Power input	3.6V – 6V
Size	43mm x 15mm

Picture 7: MLT-BT05 Bluetooth module

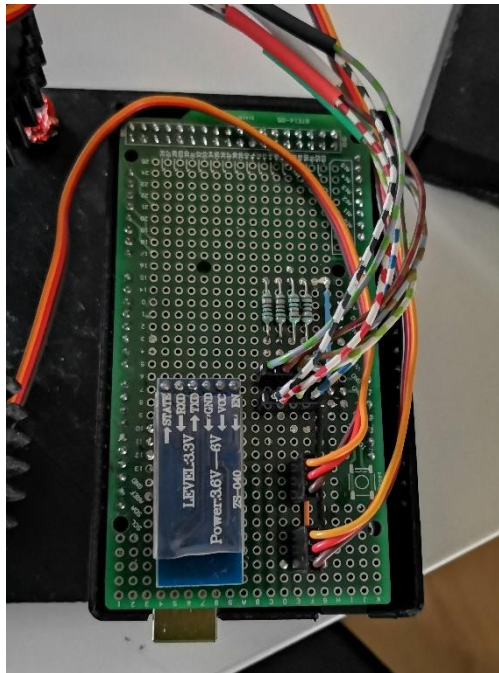


Picture 8: Blynk app page 1



Picture 9: Blynk app page 2

- a custom made Arduino shield for easier and nicer connections.



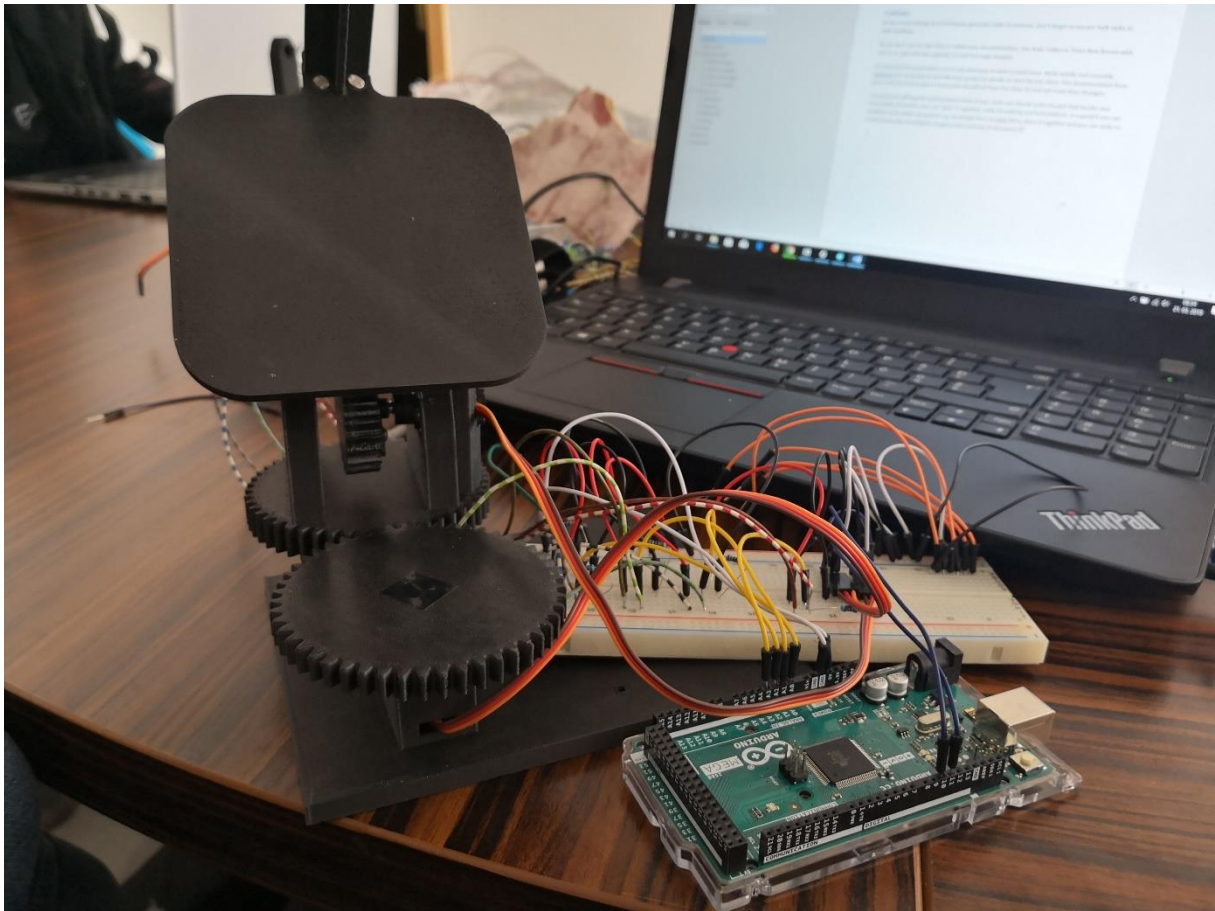
Picture 10: Arduino Mega shield

When it is finished, it should calculate from which side the light shines the brightest and automatically rotate the platform towards it. Also, there is a phone app to control the whole sun tracker via Bluetooth.

However, in comparison to normal sun trackers that are currently out on the market, our project is just a small model. Because of its small size it doesn't even generate enough power to power itself. This is all because of the small surface area of the solar panel and that's why there is no need for the solar panel to even be attached to it.

The project serves as a demonstration piece, a small model/toy that shows how the real things work. Bigger and more expensive solar trackers usually also have higher quality parts so they can be more accurate.

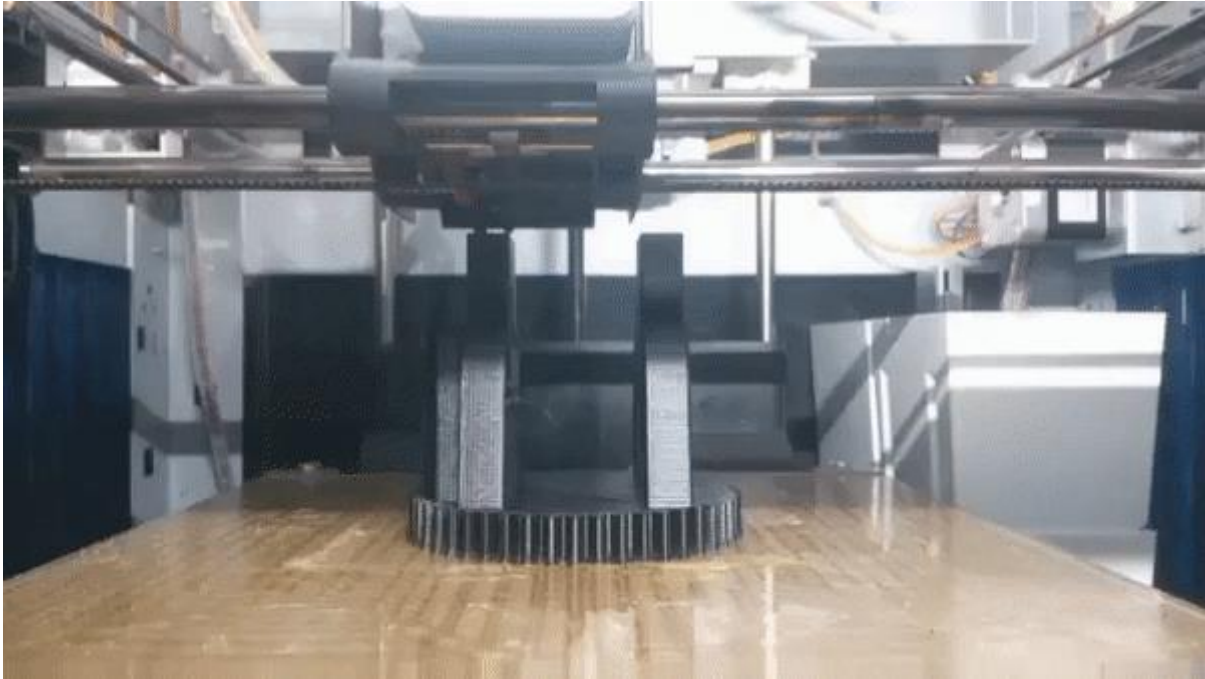
Because we used servo motors in this project, the tracker was originally limited to 180 degree horizontal motion, but that was quickly fixed with some software tricks. The cables that attach the 4 LDRs to the Arduino also needed to be really soft, because otherwise they could put too much strain on the servo motors, which are quite cheap.



Picture 11: Solar tracker in progress

2.3 Used Technologies

- 3D printer: We printed most of this solar tracker. Most of the parts that we needed were printed on an XYZ da Vinci 1.0 A. This process was really long since obviously 3D printers take some time to print things so every part needed some hours to be fully 3D printed. The entire thing was printing for almost a whole week.



Picture 12: Printing the main base

- 4 LDRs: We are using 4 LDRs; they are used to measure the light. In the program we calculate 4 averages (top, bottom, left, right) and then move the servo motors according to the highest average.
- Servo motors: We are using 2 servo motors (one for vertical rotation and the other for horizontal rotation).
- Bluetooth module: This is used for the communication with the Blynk phone app which can be used to manually control the solar tracker and can also be used to adjust some settings for the solar tracker.
- Arduino Mega: This is the brain of the project. Everything depends on it. The program is uploaded onto it. It calculates where the tracker has to rotate.
- Programming language: The program language we are using is C (in the default Arduino IDE).

2.4 Financial Analysis

Our project costed about 80€. Sadly it cannot really be made any cheaper, because we already used some of the cheapest parts available on the market. Of course it could always be made more expensive. For example we could use a better 3D printer with a bit more expensive filament so the 3D printed parts would turn out a little bit nicer and smoother. We could also

use higher quality motors or maybe some other modules to increase the accuracy of the final product. Theoretically, there is no limit on how expensive the project could be.

We think that we can't really compare our project to the solar trackers that are out on the market today, because it's tiny compared to the real things. The ones that are sold on the market are usually really expensive, big and are made to generate lots of energy, while our project is just a miniature model to show how solar trackers work.

To summarize things up, our solar tracker cannot really be compared to anything else on the market today because of its small size and low power generating capabilities.

Strengths: This solar tracker is really cheap (in comparison to other big solutions on the market today) and can be placed anywhere. It can also be used as a model to show and explain to people how some of the 2 axis solar trackers work.

Weaknesses: It is really small and can generate only small amounts of power (not enough to fully power itself due to its small platform for solar panels).

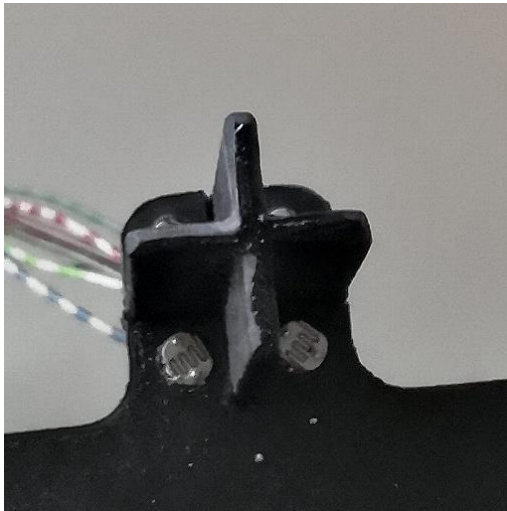


Picture 6: Solar Sun Tracker for 600€

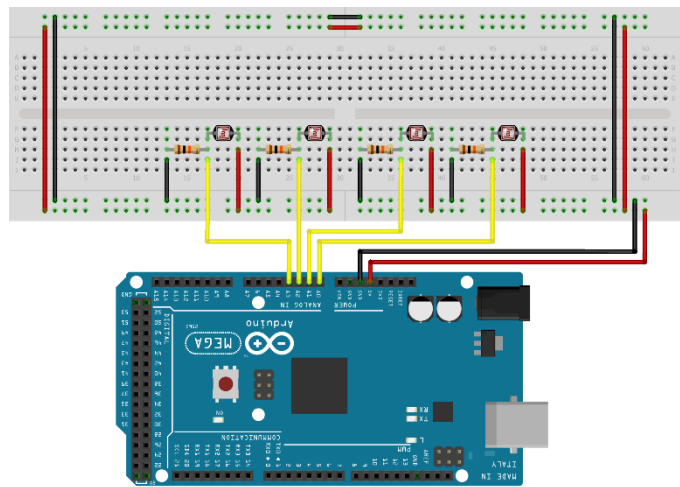
3 Practical part

3.1 Electronics

For sensing the light from the environment we used 4 photoresistors, which are light sensitive resistors whose resistance decreases as the intensity of light they are exposed to increases. They are plugged into the first four analog inputs (A0, A1, A2, A3) on the Arduino Mega. One leg of the LDR is connected to VCC (5V) and the other leg is connected to the analog input pin on Arduino. A 10K Ohm resistor is also connected to the same leg and then grounded.



Picture 13: Photoresistors in the tracker



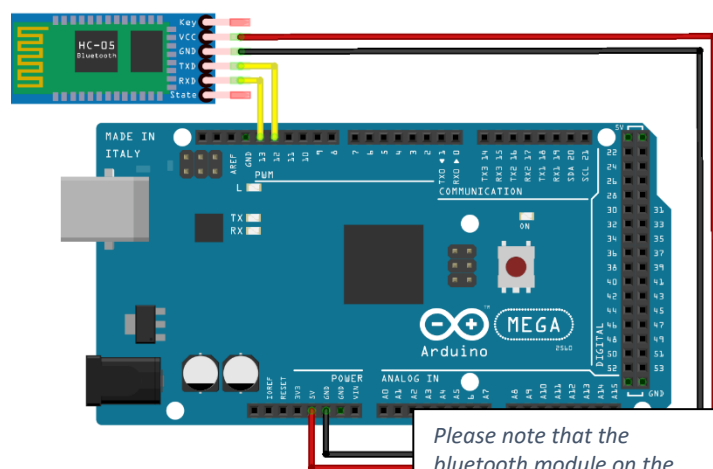
fritzing

Picture 14: LDR schematic

We also used a MLT-BT05 Bluetooth module paired with the Blynk IoT platform. The module is also connected to the 5v Arduino power output and off course also grounded. The RX and TX pins of the module are connected to the 12 and 13 digital output pins on our Arduino.



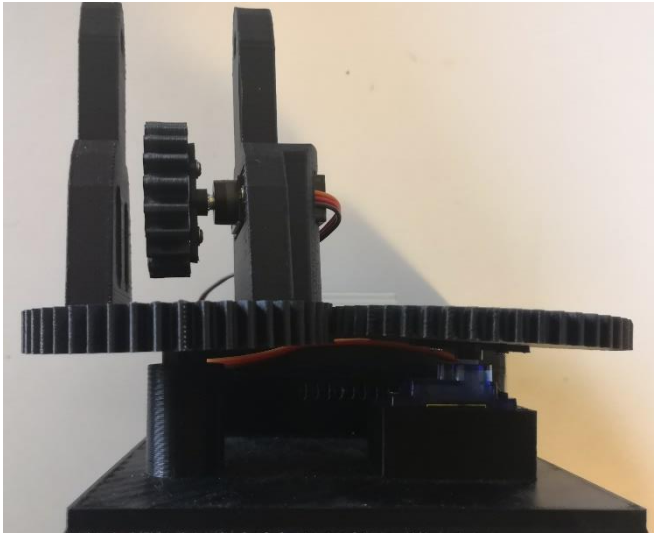
Picture 15: Bluetooth module on the shield



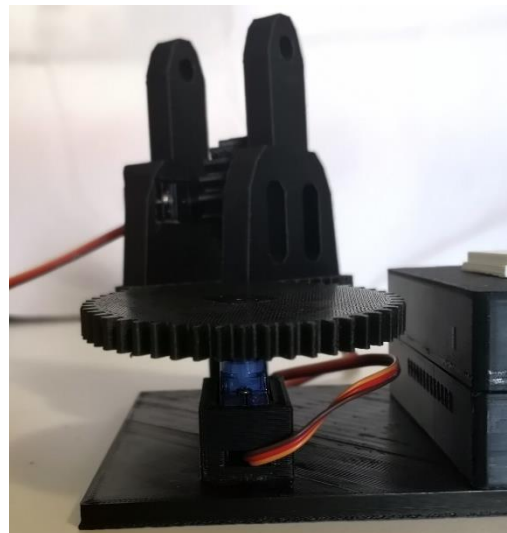
Please note that the bluetooth module on the schematic is HC-05, while we are using MLT-BT05 bluetooth module on our project

Picture 16: Bluetooth module schematic

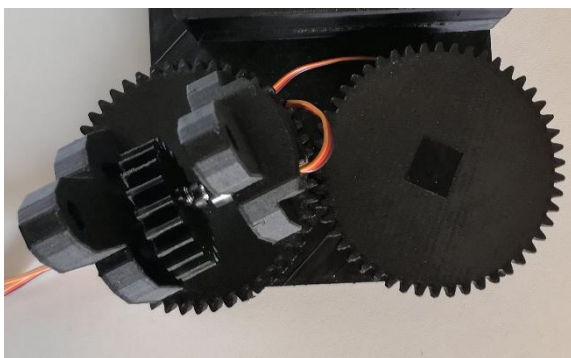
The parts that make everything move are the two servo motors. They are rotating the platform with some gears, which are of course 3D printed. They are connected to the VCC (5V) on the Arduino. The data cables are connected to the 10 and 11 PWD digital output pins.



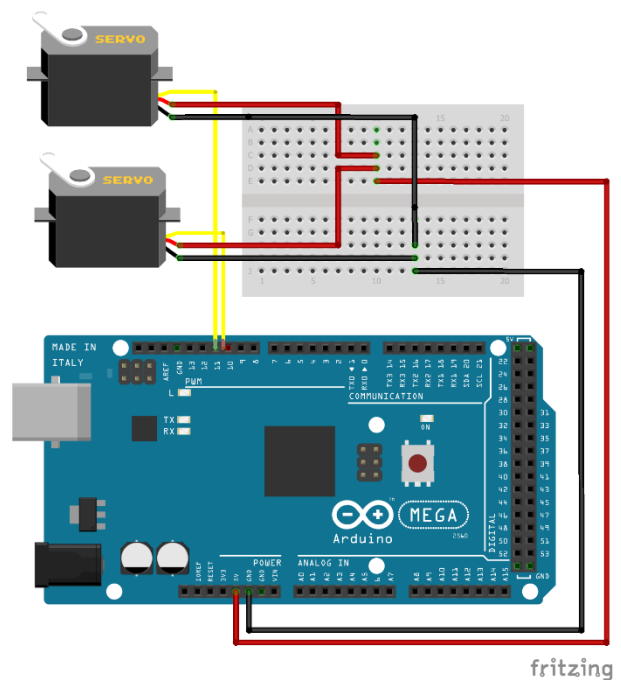
Picture 17: Servos and gears from the side



Picture 18: Servos and gear from the front

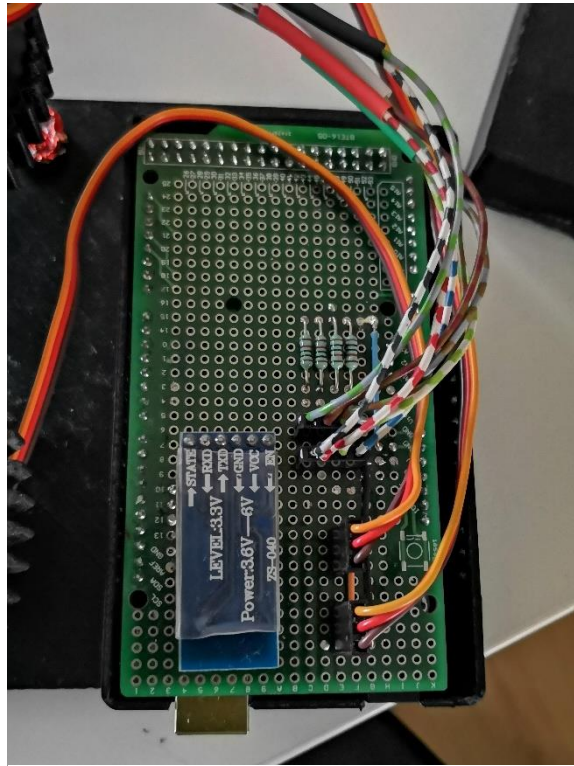


Picture 19: Servos and gears from the top

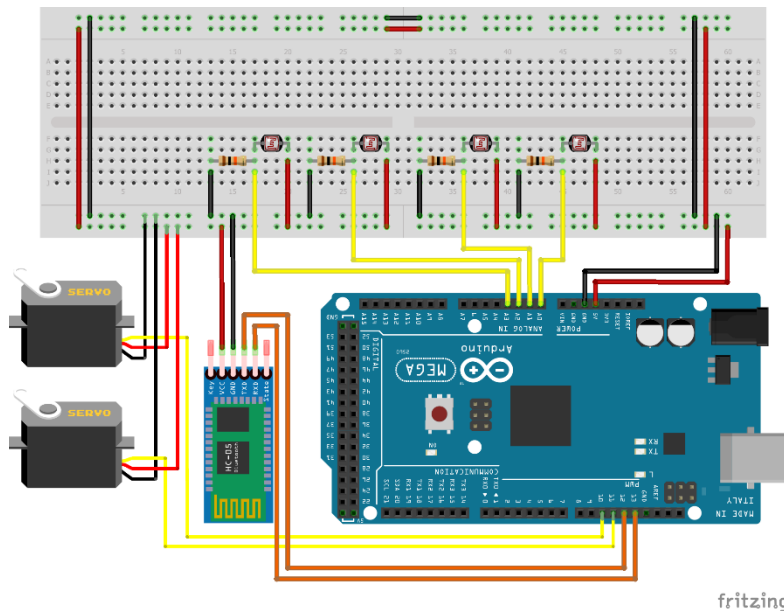


Picture 20: Servo schematic

We made a custom shield to which all of the input and output devices are connected to so everything looks cleaner and more organized. It is also worth mentioning that the cables can be plugged out in case if any of the parts fail.

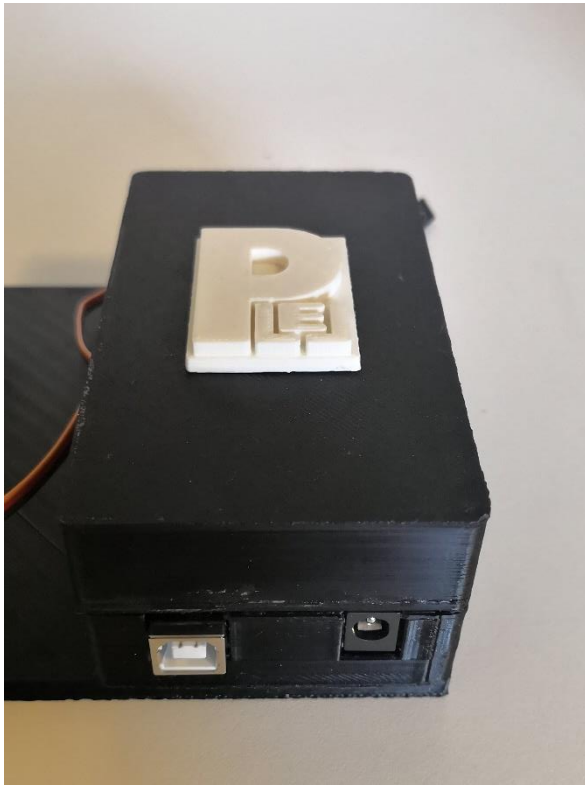


Picture 21: Our shield for all the connections



Picture 22: Shield schematic

We also 3D printed a case for the Arduino Mega and the shield to be housed in, mainly for aesthetic purposes.



Picture 23: Case from the front side



Picture 24: Case from the back side

3.2 Program

The logic behind the tracker is pretty simple. The Arduino takes the values from the 4 photoresistors and calculates the top, bottom, left and right averages and then compares them.

```
//VOID FOR SOLAR PANEL AND LIGHT SOURCE ALIGNMENT
void action(){
  configuration(); //CALLS THE VOID FOR CALCULATING AVERAGES
  if(vertical.read() >= 0){
    servov=servov-1*inverter; //IF VERTICAL SERVO MORE THAN 90 DEGREES IT ADDS 1 OTHERWISE IT SUBTRACTS 1
    //THIS VARIABLE IS LATER WRITTEN TO THE VERTICAL SERVO
  }
  else{changev=0;}
  if(LAVG>(RAVG + tolerance)){ //CHECKS IF THE LIGH IS STRONGER FROM THE BOTTOM
    if(horizontal.read() < 180){
      servoh=servoh+1*inverter; //IF VERTICAL SERVO MORE THAN 90 DEGREES IT SUBTRACTS 1 OTHERWISE IT ADDS 1
      //THIS VARIABLE IS LATER WRITTEN TO THE HORIZONTAL SERVO
    }

    //SECTION FOR CHECKING AND REACTING IF HORIZONTAL SERVO AT 180 OR 0 DEGREES
    else{
      if(BAVG > (TAVG+tolerance)){
        servoh= 20; servov= 20;
      }
      else if((BAVG+tolerance) < TAVG){
        servoh= 20; servov= 160;
      }
    }
  }
  else if((LAVG+tolerance)<RAVG){ //CHECKS IF THE LIGH IS STRONGER FROM THE RIGHT
    if(horizontal.read() > 0){
      servoh=servoh-1*inverter; //IF VERTICAL SERVO MORE THAN 90 DEGREES IT SUBTRACTS 1 OTHERWISE IT ADDS 1
      //THIS VARIABLE IS LATER WRITTEN TO THE HORIZONTAL SERVO
    }

    //SECTION FOR CHECKING AND REACTING IF HORIZONTAL SERVO AT 180 OR 0 DEGREES
    else{
      if(BAVG > (TAVG+tolerance)){
        servoh= 160; servov= 20;
      }
      else if((BAVG+tolerance) < TAVG){
        servoh= 160; servov= 160;
      }
    }
  }
}
```

Picture 25: The main part of the program

One of the more annoying things here was the fact that the servos only rotate 180 degrees but we needed the full 360 degree motion. To achieve that we basically had to check if the horizontal servo is either at 180 or 0 degrees and then flip the both vertical and horizontal servo values if it was.

We also implemented some variables into the program, with which we can control the speed (frequency of comparing the 4 above mentioned averages) and the tolerance of the light sensitivity via the Bluetooth Blynk app.

Speaking of the Bluetooth Blynk app, we can also turn off the automatic tracking and manually control both of the servos.

3.3 Testing

When it comes to solar trackers, there isn't much that needs to be tested. We tested it without the solar panel, because the main point of the project is that it has to follow the brightest light source. Also, there is no point in testing it with a solar panel, because the project itself is so small that it would most certainly use up more power than it would actually produce.

The only bigger and more annoying problem that we encountered while testing was a problem with the HC-05 Bluetooth module that we intended to use in the first place. After a lot of wasted time trying to get it to work, we decided that it would be easier to test some other Bluetooth modules. In the end we tried the HM-10 Bluetooth module and got good results from it. But because that Bluetooth module wasn't available for use (we borrowed it for testing purposes), we ordered a similar module named MLT-BT05, which ended up working perfectly.

When it comes to expectations, we pretty much expected what we got. The tracker works (it follows the brightest light source just fine), but can sometimes twitch a little bit. It can also get a little bit confused when the light comes directly from the 180 or 0 degrees of the horizontal servo, because it has to flip both servo values in order to get to the other side (the other 180 degrees) and when the light stays at 180/0 degrees, the solar tracker turns back and forth. This could be fixed in the case of normal daytime tracking with some programming tricks, but we ran out of time. Maybe we could divide the functionality of the tracker on two modes in the blynk app (demo mode and daytime sun following mode, which would only be able to flip the values of both of the servos once per day, so the servos wouldn't keep rotating when the sun would be shining from the 180 or 0 degrees).



Picture 26: Final product

4 Conclusion

The project does what we planned for it to do from the beginning. We planned to make a solar tracker that rotates towards the brightest source of light and it does just that. It would be nice if it could generate electricity, but sadly we could not achieve that on a project of such a small scale. Given enough resources we could potentially build a tracker that would not include any cheap 3D printed parts or the cheapest servo motors. Of course Arduino isn't an ideal microcontroller for a higher grade device that has to be placed outside; that's why if we would decide to make something more robust, we would go with an industrial grade microcontroller. The photoresistors aren't the perfect solution for tracking the sun either. To track the sun more accurately and without any problems with different weather conditions we would have to use some kind of GPS technology instead of the photoresistors.

In the end, sure it might not generate any electricity at all and it might be made from really cheap parts, but it can still always be used as a miniature model, to show how solar trackers work and to educate people on the importance of renewable sources of energy. It is also really fun to play around with and it usually quickly gets attention of people that are passing by.

We both think that we couldn't profit from this device in any possible way, that's why there is no point in developing it further without drastically changing the fundamentals of the tracker. If we changed out the parts for some more appropriate ones, then we could make this idea profitable. There is always so much more that can be done. We could always make a web application to see the statistics, maybe control the tracker over the web app from far away, have a camera on it, to see how it is positioned when we wouldn't be there to see the device itself... But the current market already has a lot of good trackers for all the different price points, so we think that no matter what we would do, it would not be worth the time and energy that would be needed to make this idea profitable.

The cross-border cooperation was fun in our opinion. We had no trouble with communicating with each other. It would be nice if we could spend more time in each country, so we could do more work on the project together. To be honest, we were running low on time, even if the project is quite a simple one. However this taught us how to work relatively quickly under pressure. It showed us the importance of time.

We definitely gained some experience with this project, especially when it comes to cooperation with people from different nations that you don't personally know. This experience itself is really useful, since the world is becoming more and more connected and a lot of people these days are deciding to go work abroad.

5 Sources

5.1 Pictures

- Solar tracker efficiency graph
 - o <https://www.kiewit.com/plant-insider/current-issue/fixed-tilt-vs-axis-tracker-solar-panels/>
- Solar trackers
 - o <https://www.solarpowerworldonline.com/2015/01/solarsense-completes-development-financing-2-15-mw-allearth-solar-tracker-projects/>
- 2 Axis Solar tracker
 - o <https://www.ase-energy.com/tracker-suiveur-solaire-2axes,fr,4,TRACKER3PAN2A.cfm>
- Arduino Mega
 - o <https://store.arduino.cc/mega-2560-r3>
- Servo motors (SG90)
 - o <https://ixen-robotics.com/product/sg90-micro-servo-motor>
- Photoresistor
 - o <https://www.pcboard.ca/gl5528-light-dependent-resistor>
- MLT-BT05 Bluetooth module
 - o <https://blog.yavilevich.com/2017/03/mlt-bt05-ble-module-a-clone-of-a-clone/>



Stredná priemyselná škola elektrotechnická, Košice, SLOVAKIA
Střední průmyslová škola elektrotechnická, Havířov, CZECHIA

CONVEYOR BELT WITH POSSIBILITY OF SORTING PRODUCTS



Co-funded by the
Erasmus+ Programme
of the European Union

Introduction

Technological revolution has been here for years now and nowadays, automation is the key to a successful business. Today, one person is enough where three men were needed in the past. Robotic arms, CNC machines and conveyor belts are used very often in various industries. We were fascinated and curious about how an ordinary sorting machine works and the idea of building one seemed to be getting stronger and stronger every day.

We liked the idea of a conveyor belt that would sort objects into containers based on their physical properties. We also decided that using a programmable logic controller would be our best choice.

We needed to make our expectations realistic and think about some properties we need to agree to before starting to work on it. We decided that we want the machine to be compact, visually elegant and fully automatic. It has to be connected to the PLC by a connector and controlling should be user friendly. Also, parameters of the sorting process are adjustable and although, the machine will be complicated, it will look simple to an ordinary user.

We also decided to print most of the parts on the 3D printer using PLA material. This is supposed to increase the accuracy and precision of all the home-made parts, lower the cost of some parts (why would we buy them when we can print them?) and maybe save us some time. Also, it's not that easy to get a lathe and similar big or rare machines in our hands.

Everything started with designing the sorting machine visually in 3D – on the paper at first, but then we transferred the data to Autodesk AutoCAD so it can also be printed by the 3D printer. We then went on a shopping spree and bought majority of needed electronic components and materials like wooden plates, transistors, buttons and so on. We then started to print parts on the 3D printer one by one and we slowly began to construct the sorting machine. The last step was to program the PLC and after some tests, we were good to go.

All that work resulted in an automatic PLC sorting machine that sorts the object based on their weight, color, height and its ferromagnetic properties.

Theoretical part

Market analysis

As we stated earlier, there are some sorting machines being used today, but most of them are pretty simple and you are not able to change the sorting properties – you'd have to rebuild the machine. Most commonly sorted are fruits and vegetables like apples or potatoes, which are sorted by size.

More complex are garbage sorting machines, which are fairly new to the market and can sort out iron materials, plastic waste, heavy materials like bricks or glass and secondary heavy materials like wet paper or hard plastic. These machines are, indeed, huge and really expensive.

Though, we could not find any smaller sorting machines that could, for example, sort out batteries based on their dimensions or voltage.

Our machine will be compact, cheap and really easy to operate. It is able to sort out products based on four different and independent features of the objects.

Technical analysis

There will be several wooden plates used when building the sorting machine. One as a platform, four at the front to hold all the electronics and one side of the conveyor belt, and two at the rear, to hold the other side of the conveyor belt.

To move and deliver the objects we decided to use a conveyor belt similar to those you can see in stores (though, smaller, of course). This conveyor belt will be powered by a DC brushed motor with reduced RPM but high torque, so we won't have to adjust it ourselves, though, it would be possible. Since the motor runs on 12V, we will use a relay to control it. The conveyor belt will be placed on two cylinders and both of these cylinders will be rotating on bearings. To make the cylinder spin, however, we will design teeth on the front cylinder and also on the motor shaft. They will both act like two gears and the cylinder will rotate along with the motor.

There is a big amount of servomotors available on the market nowadays. Well, that's because they are so handy. Regular person doesn't really notice that, but they are also used in cars

and airplanes. The smaller ones, which we will talk about, are mainly used in variety of RC models. The most used servo is a 9g one. It's tiny but its torque is huge, so we decided to use them to push the objects onto the scale and conveyor belt, to drop objects from the conveyor belt into the containers and as a gate so the object doesn't slide onto the conveyor belt right away after its properties are detected. For servo to be able to push the objects linearly, we will design a part called "linear servo" which consists of a gear and "pusher" with integrated teeth. You will see how it's done and how it works in the next chapter. On those two container servos that will make the objects drop from the belt into the containers, as well as on the gate servo, we'll just design a shaft extender, as you'll see in the next chapter. Servos use PWM signal to be controlled and they operate on 5 volts, so we will use transistors to drop the voltage of the signal and we'll need to make the PLC generate PWM signal, which might not be easy.

To detect presence of an object, we are going to use several infrared optical barriers and Arduino to convert diode's analog output to a digital one. Optical barrier (optobarrier) works like a gate made by a light beam. One optobarrier consists of a LED (light source) and a photodiode (receiver). As light is being emitted from LED and received by photodiode, the photodiode's resistance is changing based on how much light it receives. Since the LEDs and therefore, the light emitted by them, is infrared, naked eye can't see any of that happening. But if there's something in between the source and the receiver, beam is cut off and this change is what we're after. Though, the photodiode's output is analog, and we don't really need to know how much light there is (we only need to know "yes or no"), we will convert the analog signal to a digital one. This can be done by using an operational amplifier and a few resistors, but we thought it would be easier to let Arduino do this conversion. Arduino will compare photodiode's value to the preprogrammed one and turn the output on or off based on that. Though, the output voltage will only be as high as 5 volts, so we will also use a transistor to "make" it 24 volts.

Since the PLC operates on 24 volts, motor operates on 12 volts and other things like servo and Arduino operate on 5 volts, we need to use a relay and several NPN transistors (common ones like 2N2222 or BC547). To provide these voltages, we'll also use 12V and 5V BEC (Battery Elimination Circuit). These circuits are just tiny boxes or PCBs and are most widely used in RC toys. We could use a computer power supply instead of regulating the voltage with another electronic components, but this way we saved a lot of space, few bucks and few pins of the connector.

We checked the market to see prices and products available to detect object's weight, but we found out that there is huge lack of simple scales and tensiometers available on the market.

Designing a scale is quite tricky, but we found that the best solution is to put a spring in the middle and a countersunk screw right next to it, so when the spring is pressed, its thread will touch the screw's head and this will act like a switch. You'll see what we're talking about in the next chapter as well.

Since the scale and the first servo (feeder) are pretty big, we will have to lift it, but as we do that, there will be big vertical leap between the scale and the belt. We don't want the objects falling from that height so we will also design a "slide" part. Its name says it all – the objects will slide, but while sliding, their ferromagnetic properties, as well as their color, will be detected.

We will also use optical barriers as a height sensor and as a color sensor (we can detect the color of an object when emitting light on them and measuring amount of the reflected light). To detect the ferromagnetic properties we'll use an induction sensor IE 5295.

We will print the objects on the 3D printer as well – tall, heavy, normal, white and light one. We will wrap the normal one into an aluminum foil to make it appear like metal.

Then, we will use two Centronics connectors to be able to disconnect the machine from the PLC.

Used Technologies

As we wanted to make this sorting machine perfect, we needed to be precise. There are parts, where one tenth of a millimeter is a big deal and many other ones, where one millimeter could cause malfunction. To let ourselves be creative, we decided to print as many parts as possible on a 3D printer. If we would make all the parts from wood, for example, it would take forever in case we wanted to be precise enough. With the 3D printer, it barely took a week to sketch all the parts on a paper and another two or three weeks to convert these sketches to a 3D environment. We used Autodesk AutoCAD to design the sorting machine, later exported each part to STL format, used software called Cura to properly prepare the printing process and finally, we just watched it getting printed. Whole sorting machine was planned to be grey so we also used grey filament. Also, since ABS is pretty tough to print and too hard for these parts (although, it's the most used material in 3D printing world), we decided to use PLA. PLA's disadvantages are low

melting temperature and the fact that it decomposes when it's soaked in water. None of those mattered to us, though.

To make the program we used PLC AMAP 99/12 from AMIT company (*Attachment 1, page 17*). This PLC is capable of 21 digital and 15 analog inputs. The output may be analog (6), relay digital (19) or transistor digital (4). PLC communicates with computer via RS232. That usually makes programming complicated because nowadays laptops don't have a COM port. You have to have USB to COM port converter or work with an old computer. AMAP also has a possibility to work with displays with integrated keyboard. Memory of AMAP is quite poor, 256kB RAM and 512kB Flash, but it's still enough to work with. To connect our model to PLC we use 25 and 8 core cables ended with 24 pin centronics connectors (*Attachment 2, page 17*). PLC, same as the sorting machine, is powered by 24V DC from a power supply.

To write software for the sorting machine we used program DetStudio made by AMIT itself. There are used mostly two languages, C and RS (relay schematic). RS language is a kind of graphic language. Software is made of blocks connected together (*Attachment 3, page 18*). Every block has its unique function like time delay or logic function. In this program we work with inputs and outputs as aliases. One alias can be only one input or output signal. We also use constants for values that are used only in software for system logic – these aren't inputs and neither outputs. In the program we used about 22 aliases and even more constants.

Financial Analysis

Well, since there is no machine similar to ours, we can't compare the prices no matter how hard we try.

Anyways, the first thing to do before buying all the needed parts was to collect their prices and check the budget. First, we listed all the parts we need and might need. Then, we sorted the items based on their importance. As we saw that the price was much higher than 100 euros, we decided not to buy several parts that are not necessary or can be replicated. For example, we didn't have to buy a color sensor because we constructed one ourselves (the one that was supposed to be provided by our school, didn't work, so at first, we wanted to buy another one). Since we had stuff like heat shrinks or some "leftover" metal rods, we used them and didn't have to buy them, therefore include them in the budget.

As we compressed the price as much as possible, the final price of all the parts was just a little bit more than 100€. Well, it was 97€ without the wooden plates. It was hard to predict their price in the budget.

Here's a little SWOT analysis of our sorting machine:

<p><u>Strengths</u></p> <p>Reliability Precision Design Being one of a kind</p>	<p><u>Weaknesses</u></p> <p>Complicated to build Sorting machine can only sort round objects with the same diameter</p>
<p><u>Opportunities</u></p> <p>To design a sorting machine being able to sort much different objects, maybe a variety of them</p>	<p><u>Threats</u></p> <p>Arduino-based boards may be able to replace PLCs</p>

Practical part

Electronics

First things first, the machine could not work without the motor-powered conveyor belt. As we stated earlier, the conveyor belt (60mm wide and 1400mm long) is placed on two 3D printed cylinders and one of them has teeth on its side. There's also a motor with teeth on its shaft and we designed the "motor holder" part in a way that these teeth touch and all the energy from the motor is delivered to the conveyor belt. The motor is a brushed one and works with 12V direct current, so we had to adjust the voltage for it and put a relay before the motor. Also, these brushed motors always have high RPM (we're talking about thousands), so we needed to make sure that a gearbox is included, which converts it to, in our case, 100RPM. The motor requires an average of 0.5A and no more than 1A when operating. To see how those gears look like, check out *Attachment 4 on page 18*.

To control the sorting machine, there are four buttons. Each of them will have its own function:

- Green: Start – Turning on all the sensors, initialization of the program and variables. In case of the dock not being empty, motor starts turning and the sorting process starts.
- Yellow: Pause – Turning off the motor. Since there is no timer used in the program (well, there are few minor ones), it's sufficient to turn the motor off and the sorting process is temporarily stopped. Program, variables and sensors still run, only the motor does not. Pushing this button again simply turns the motor back on.
- Blue: Finish – Finishing the sorting process and turning the machine off. Objects, which are already being sorted, will be sorted but after this is done, the system shuts down. This is the right way to turn the sorting machine off.
- Red: Stop – Immediately stops every single process, resets all variables and shuts everything down. This button is supposed to be used in case of emergency.

Typical servo has three wires: 5V, ground and signal. We connected 5V and ground, but we couldn't connect the signal cable because the PLC only provides 24 volts and that's too much, so we decided to use 2N2222 transistors, hook their base on the PLC's output and their output on the servo. We were hoping for it to work, but we have got it working only as we switched the

transistors to BC547 after hours of wondering what's wrong. We still don't know the reason why didn't it work with 2N2222 ones, though. See the scheme in *Attachment 5, page 18*.

The first servo is called feeder - it kind of feeds the sorting machine with objects. Since the servo only moves radially and we need a linear movement there, we designed a part called linear servo, which consists of two parts: linear pusher with teeth underneath and rotating gear with teeth all around. The gear is hooked on the servo and as it rotates and delivers the energy to the pusher, the pusher moves linearly and that's how it's able to move the objects. The pusher can be in four different states:

- Idle state – the pusher is way in the back and doesn't affect any object
- Position 1 – moving an object onto the scale
- Position 2 – moving the pusher back just a little bit so it doesn't touch the object
- Position 3 – moving the object onto the slide

You can see how the linear servo looks like in *Attachment 6, page 19*.

The two container servos are much simpler – we just printed a part to lengthen their shaft and put them in their place. They can be in three different states:

- Idle state – the extended shaft is parallel to the belt and doesn't affect any object
- Position to sort the object into container 1 or 3 (about 45°)
- Position to sort the object into container 2 or 4 (about 135°)

You can see those two servos in *Attachment 7, page 20*.

There is a shaft extender on the fourth – gate servo - as well, though, this one is shorter and has a different shape. It also has three positions:

- Idle state – the extended shaft is parallel to the belt and doesn't affect any object
- Position to stop the object near the induction sensor for us to make multiple detections
- Position to stop the object near the two-way optical barrier to detect the objects color

Since we want the beam of the optical barriers to be as thin as the LEDs, we designed and printed a housing for them. It's also nicer, as seen in *Attachment 8, page 21*.

Scheme of the optical barriers and how is the voltage converted is shown in *Attachment 9, page 22*.

Designing a scale was really tricky, but we somehow managed to do so and you can see the scale in *Attachment 10, page 22*.

When a heavy object is placed on the scale, the spring bends enough to touch the countersunk screw so the current can flow and that's how we achieved a digital scale. Although, it might be sensitive to shaking and calibrating might be necessary, we could not think of anything better. With it being designed as it is, we can adjust the sensitivity and calibrate it in three different ways.

Since the part containing the dock, feeder and the scale is much higher than the conveyor belt, we designed a "slide", which we later printed. We put the color sensor and induction sensor on that part to detect additional properties of an object while sliding. Slide is pictured in *Attachment 11, page 23*.

Overall, we have these sensors:

- Optical barrier detecting if there is an object to sort
- Optical barrier that is triggered when a tall object is being sorted
- Scale with digital output
- Induction sensor
- Two-way optical barrier acting as a color sensor
- Five optical barriers, each for one container, confirming that the object was sorted successfully

You can see the overall scheme in *Attachment 20, page 28*.

Programs

The main program is created in AMAP 99/12 PLC by AMIT company. We chose this PLC because it has enough digital inputs (we need at least 10) and digital outputs to be able to make PPM or PWM signal. The PLC is powered by 24 volts and it communicates with PC via COM port. To make program itself, we used program DetStudio where we use RS (relay schematic) language. That means that the program is made by many different blocks connected together. It's way more clearly visible but in more complex programs you may get lost.

To get the sorter running we need input and output signals. All inputs are digital (0 or 1), also outputs are digital but they have to control servos, so the relay digital outputs can't be used because they are too slow for the PPM and PWM signals. There is only one exception for belt conveyor motor.

The program works like this: when we push the start button and the infrared signal of an optical barrier in dock is interrupted, the clock starts running. Counter which is connected to the clock slowly changes PWM values to let the servo slowly change its position. We chose to do it that way because when we tried to change the PWM value instantly, it pushed the object so fast that in the most cases it fell on the ground. The first position is before infrared optobarrier, the second position is on the scale where we measure if the object is heavy or not and the last motion pushes the object on the slide and the belt. There, two sorting servos wait for the object. We programmed them all by testing various lengths of pulses. When we push the pause button, belt stops running but all sorting stuff is still active. Pushing the same button again ends the pause. When we push button stop or finish (same as stop but stops after the end of the sorting process), all inputs like optobarriers will be active but servos will ignore them so it looks like servos are off (no signal in servos). Also, when the dock is empty, the machine stops since there is therefore nothing to sort. When machine is off you can touch everything, it won't make a mess because everything is reset in the next sorting process.

For Arduino, we programmed it in its default programming environment – Arduino IDE. The programming language used here is widely used C. Quick description of function of our program: Defining and declaring all needed variables; declaring if a pin will be used as output or input; defining a constant for each optical barrier – this constant was measured before; reading a value on the photodiode → comparing this value to our constant → if it's lower, turn the output pin ON

and if it's higher than the constant, turn the output pin OFF → do that with every single photodiode and repeat.

You can see the source code in *Attachment 12, page 24*.

Mechanical

We truly took our time and designed whole sorting machine in the 3D environment of AutoCAD. Totally, around 50 different parts were designed and 3D printed to make our sorting machine work and we have used almost 2 kilograms of material. It was all printed on cheapish GEEETECH i3 3D printer not costing more than 150€. Since this 3D printer is not a property of our school, we could be printing even at nights.

Starting with a dock (tube), it's a two-part hollow cylinder in which the objects are stored before being sorted. These two parts are joint together by three M3 screws and whole tube sits on four legs using four long M3 screws as in the photo in *Attachment 12, page 25*.

This way we are able to put the dock down for easier carrying.

The front two legs, along with holding the tube, also hold housings of the optical barriers. The rear ones have holes in them and that's where two 6mm rods – parts of a linear servo – go. You can check out how it looks in *Attachment 14, page 25*.

On the side, there are buttons, so we put them in a box. Every single cable from that side goes through this box and to the other side where we have put all the electronics. See the picture in *Attachment 15, page 25*.

We decided not to put the electronics into one case but rather design few layers that altogether form a case. This makes it much easier to construct a put the sorting machine together. In that box, we have put all the transistors, relay, BECs, connectors and so on. You can see how does the electronics box look in *Attachment 16, page 26*.

For the wooden plates, we joint them together using countersunk wood screws and the rear ones (triangles) are connected to the front ones by M8 threaded rod. Beneath the triangles we created something like rails and by screwing the nuts on the rod we can tighten the conveyor belt, which is pretty important. For the triangles, there is one screw coming from underneath

somewhere in the middle of the triangle and we also used 3D printed angles to hold the triangles down as seen on the photo in *Attachment 17, page 27*.

As we stated, the cylinders have to rotate so we have put them on bearings. You can see that in *Attachment 17, page 27 as well*.

With the containers, first plan was to put them very close to the conveyor belt, but we realized this is not a good option. We needed more space for the servos, both M8 rods and all the photodiodes. We adjusted the plans and to get the objects from the belt to the container, we, again, used slides. There are two slides on each side and both pairs are connected by a 6mm metal rod to keep them at the right angle. These slides are places and screwed on the M8 rods and also hold the optical barriers. On one side they also hold both servos. You can see that in *Attachment 7, page 20*.

We have also printed many anchors to help our cable management, angles to hold the wooden plates in 90° angle and other little things that were not necessary, but we added them to make the machine more stable.

Check out the design of the sorting machine in *Attachment 18, page 27*.

Testing

Everybody wants working program and hardware without any issues or hazard. This is the main reason why we test everything several times to be sure that everything is all right.

First, we tested all the sensors before they were mounted into model. Most of them worked properly except the color sensor. This sensor had to give signal only when bright color is nearby. The problem was that this sensor gave us the same signal regardless of color of the object that was next to it even though we measured them from the same distance. So, we changed it with another sensor made by us – a two-way optical barrier.

After making a program when the model was almost done, we discovered plenty of hazards connected with the first part of sorter. By this we mean problems with feeder, dock, slide and metal detector.

First problem is connected with the feeder and the dock. When the feeder loads and pushes the object, another one drops on it. Then it moves with the servo and meanwhile second object drops next to it. This is the hazard - there cannot be two objects at same time. Also, these objects may drop down from the feeder. We invented barriers around the dock to minimize the risk of falling but this wasn't enough. The thing that helped was pretty simple: we increased height of the pusher and also adjusted the rear legs accordingly. Though, it took hours to figure out.

Second problem was too fast feeder pushing objects to the air and on the floor. Problem was solved by changing program. Instead of doing one big step (change) in PWM signal, we did tens of little steps. That upgrade significantly slows the feeder.

Last problem was slider and metal detector. First, we thought that slider will work perfectly. We were wrong. Sometimes when we sorted high object, it started to roll like a ball. We needed to slow it down and do something that stops it rolling. Meanwhile we discovered, when metal object is going through, it's too fast. Detector detects it but PLC is too slow to notice. We decided to merge these problems to one. We decided to add another - fourth servo to stop the objects on the slider so we will have just enough time for PLC to get the signal and slow the objects down before falling on the conveyor belt.

We tested model not only for program or model hazards but also for servo values. To have good position of the servo, we had to have good and precise signal input. We let servos running but we changed its signal values to find the best one. When the servo was in good place, we saved the last values. We did the same process for all of them.

The only problem we didn't fix (because it isn't possible) doesn't seem to be so big. While sorting is working, all measuring sensors are on. That means, when someone swings his hand through the optobarrier, PLC will think that it's usual object and gives signal to PLC. This situation will ruin sorting process. To fix process someone have to press STOP button, take all objects that were in motion and then press ON button to start process again. ON button resets values. Though, only trained personnel should be granted the permission to work around this kind of machine.

Conclusion

In this project we had to make belt conveyor with possibility of sorting high, heavy, differently colored and metal objects. When the project started, we were both quite excited but also nervous, because it was the first our project like this (working with stranger from other country). It all started by receiving an email with information about the project and what's our job. Later we received another one with an email address of the other student that we should be working with. Initiating contact wasn't easy and from the beginning, there was not much of a communication, but when we first met, it wasn't bad at all. We were working (virtually) together almost for a year when the Czechs arrived in Slovakia. By the time, we made a lot, but it was mostly just design, though, many parts were already printed. It's funny how those 6 hours in school mostly appear as a lifetime but as we worked together, it flew so fast that we didn't even notice and we were saying goodbye. In Slovakia, we focused on the hardware and only ran a few tests with the PLC. Well, non-existent machine is not a programmable one. By the time that Slovaks arrived in Czechia, all the hardware was supposed to be finished (until we found out that it's not working and had to repair it) and the main goal in the second mobility was to program the machine. Looking back, most of the work was done within the two weeks we worked together in one room. Those 10 days were the only ones when we could work with both the sorting machine and the PLC controlling it. We can say that our project is successful even through obstacles like the impossibility of testing the program because PLC was in Czech Republic and the machine was in Slovakia. By now, we have made few minor improvements, so to have the project fully done, we need just little time together to make last debug and test. Anyways, photo of the working sorting machine is in *Attachment 19, page 28*.

It was strange, yet encouraging to see ourselves putting our heads together. What one of us could not have known, the other did. It took a day or two to get used to each other, though, then we worked together splendidly. We think it's very important for us in the future to be able to work well with anyone.

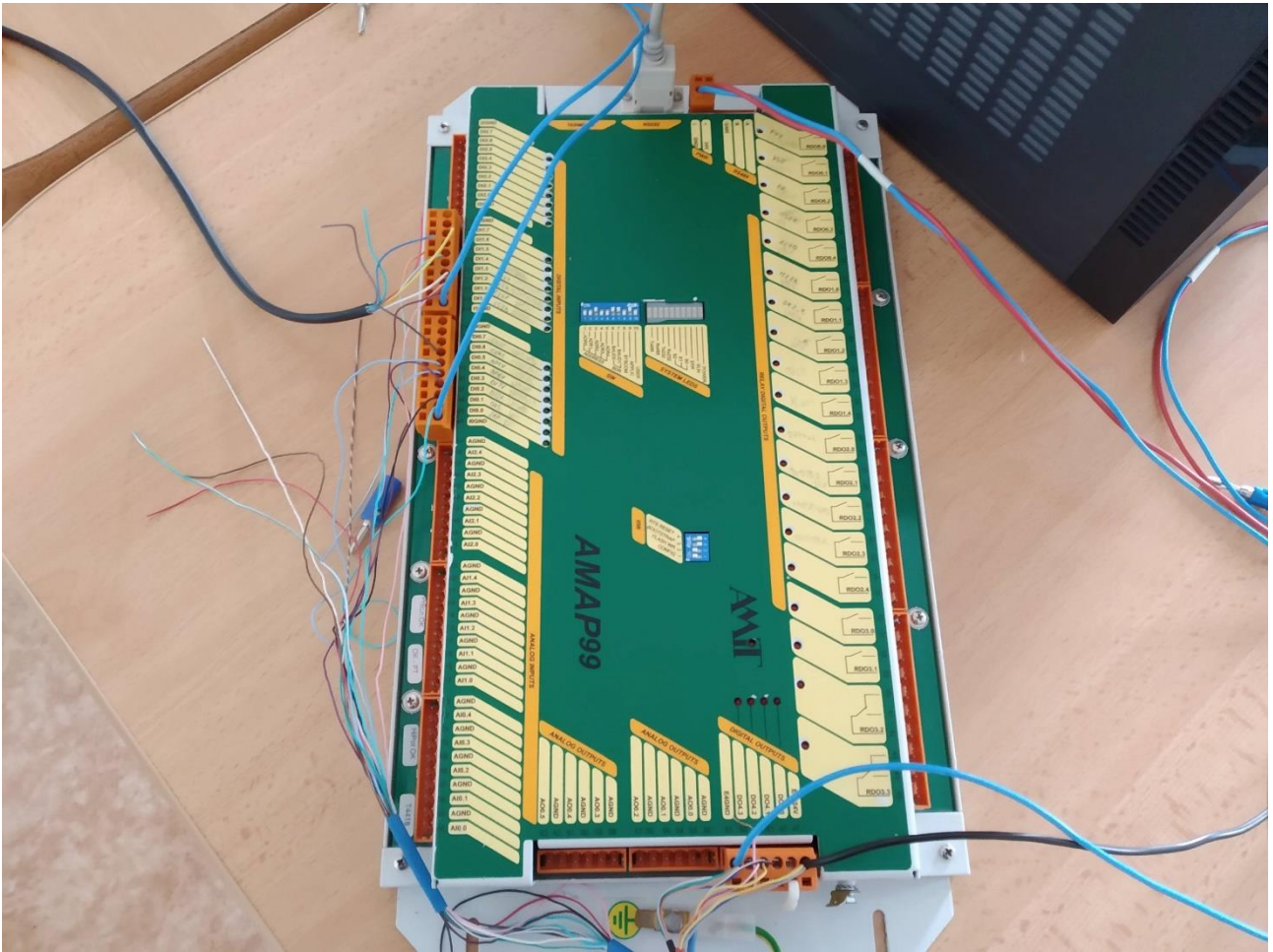
It was also good to learn to manage our time, since we only had limited – very limited – amount of time. Working effectively was not an option, it was a necessity.

It's unbelievable how many unexpected problems we encountered during our work, but we are proud to say that we solved every single one of them.

After finishing the Erasmus+ project, we are not completely sure about what is next with our prototype. We will probably upgrade it and try to win some competitions. We'll see some other – preferably also dark – sides of our project as other people will tell us what they think about it. Anyways, there is a low probability that we will still work together because we are still students and we live really far away from each other. However, if one was to travel to a city near the other one, we would probably get a beer together anyways.

The Erasmus+ project was very useful to each of us. We did something new in life and we learned a whole lot. Not only about creating a working machine, but also about other students, schools and their pros and cons. One of most important things is that we experienced international cooperation. That's pretty unusual for high school. People usually meet like that on university or at work. We think it was a great possibility and would recommend it to anyone.

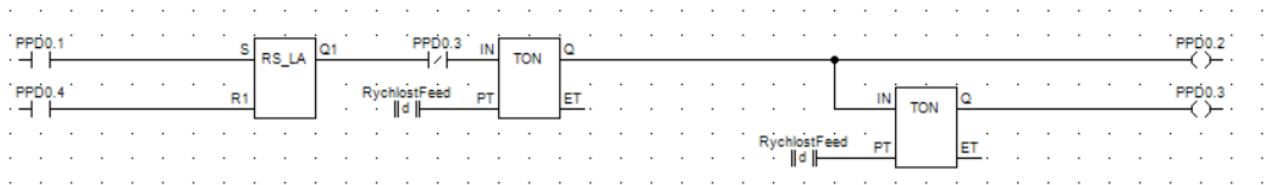
Attachments



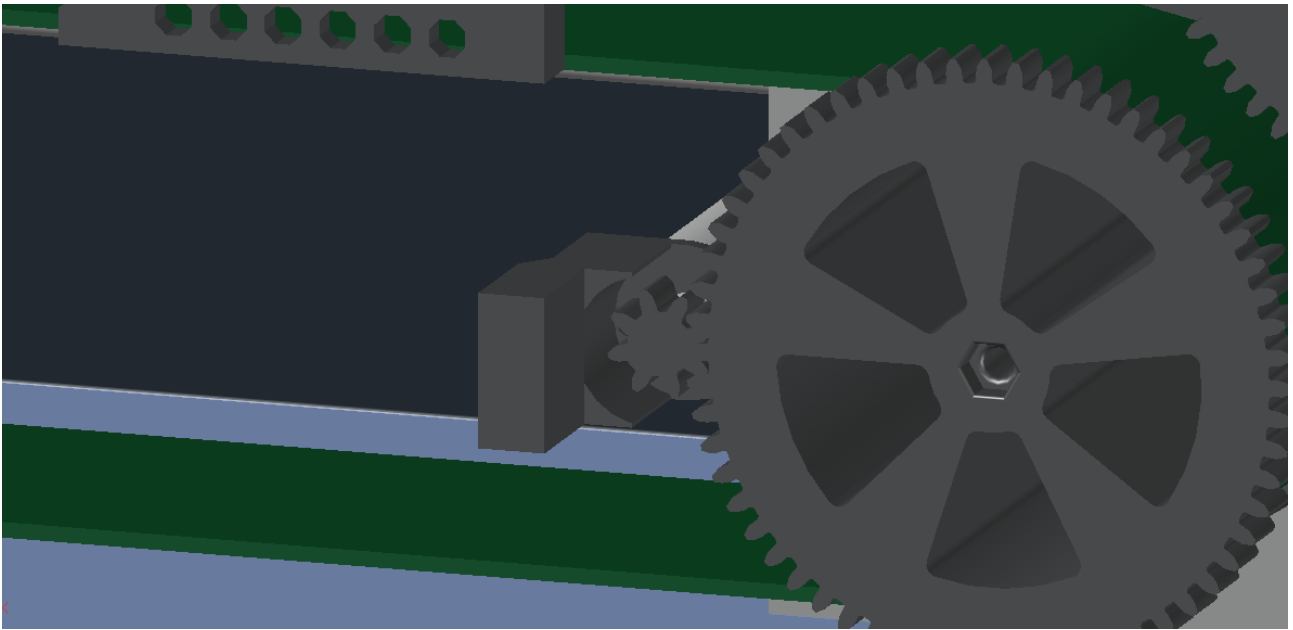
Attachment 1 – AMIT AMAP99



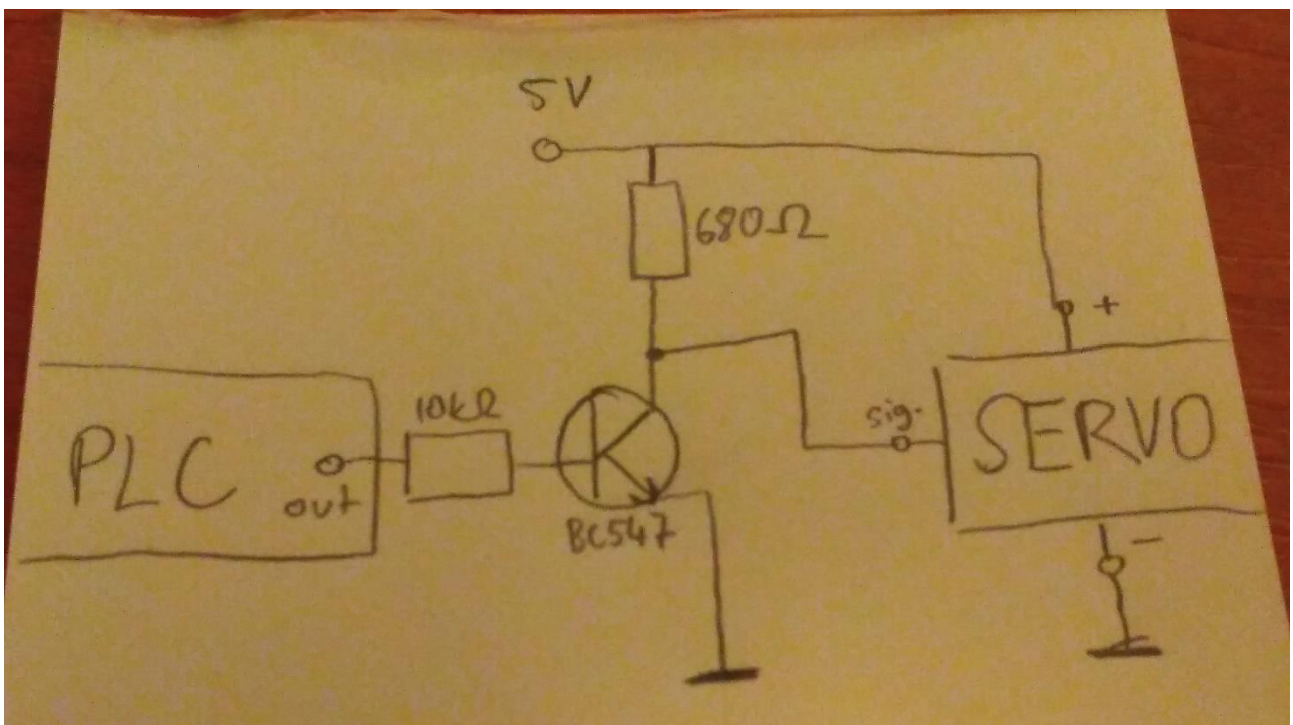
Attachment 2 – Centronics connector



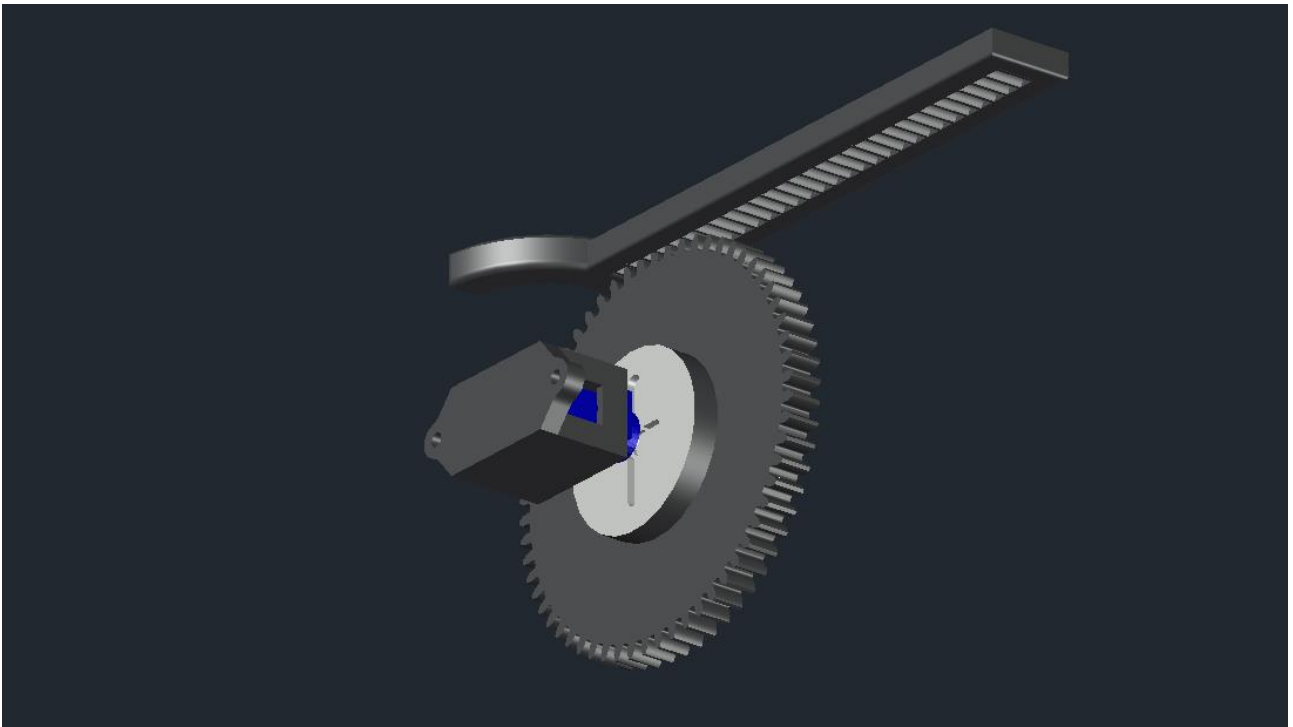
Attachment 3 – Small sample of a code



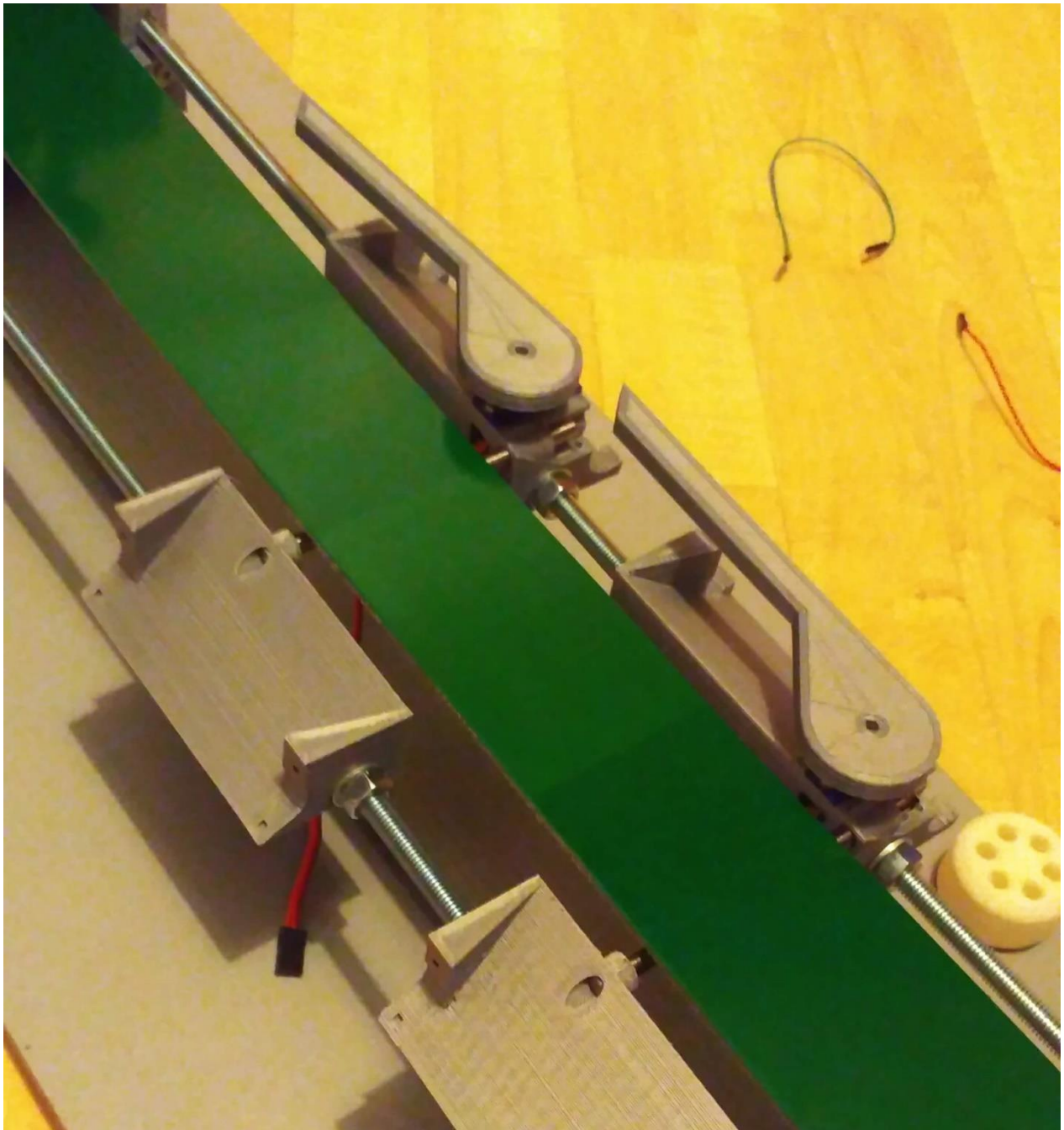
Attachment 4 – Gears



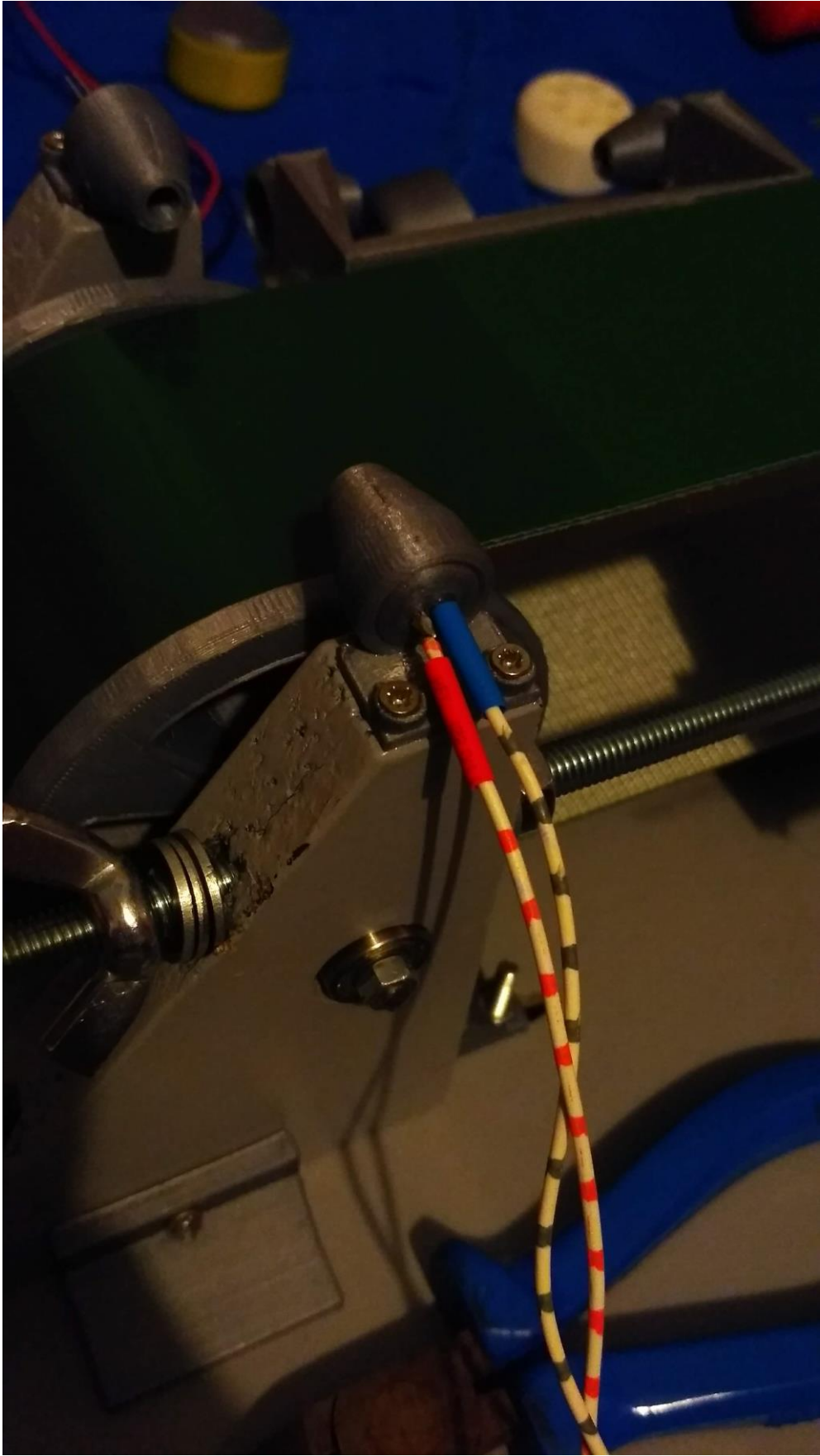
Attachment 5 – Scheme of a servo



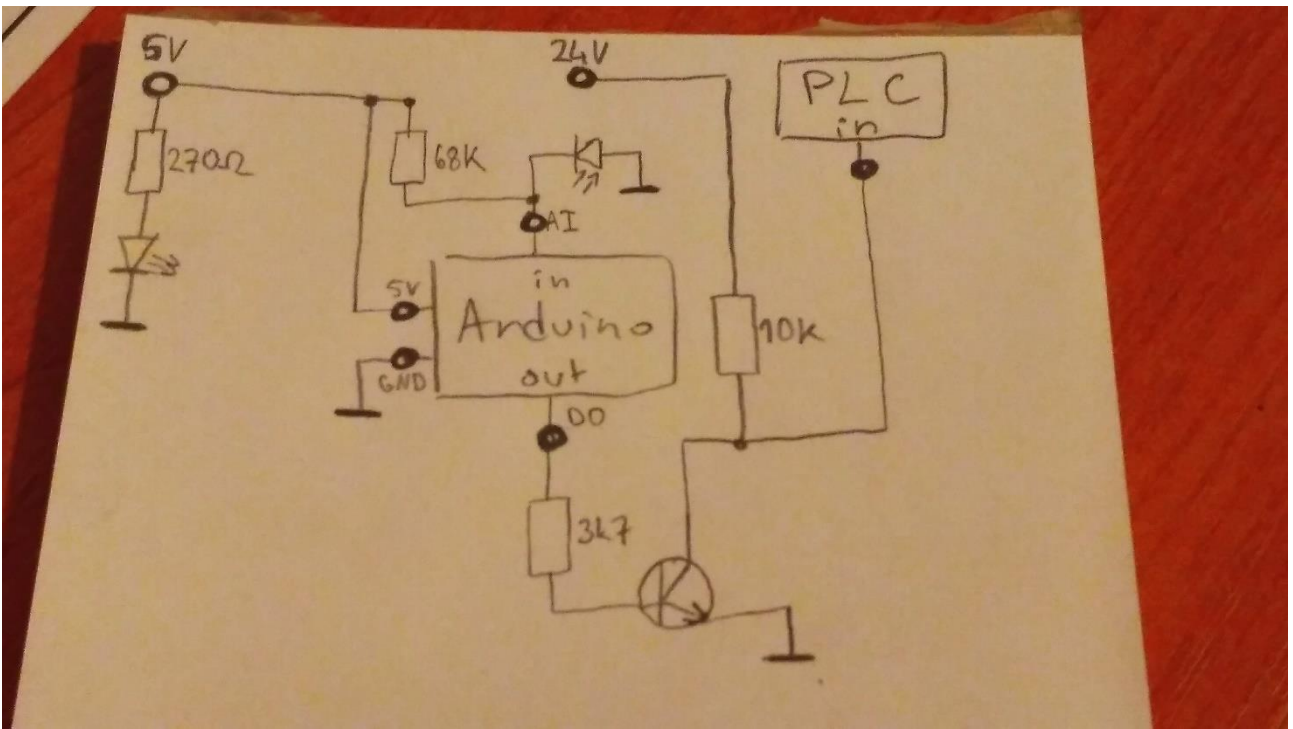
Attachment 6 – Linear servo



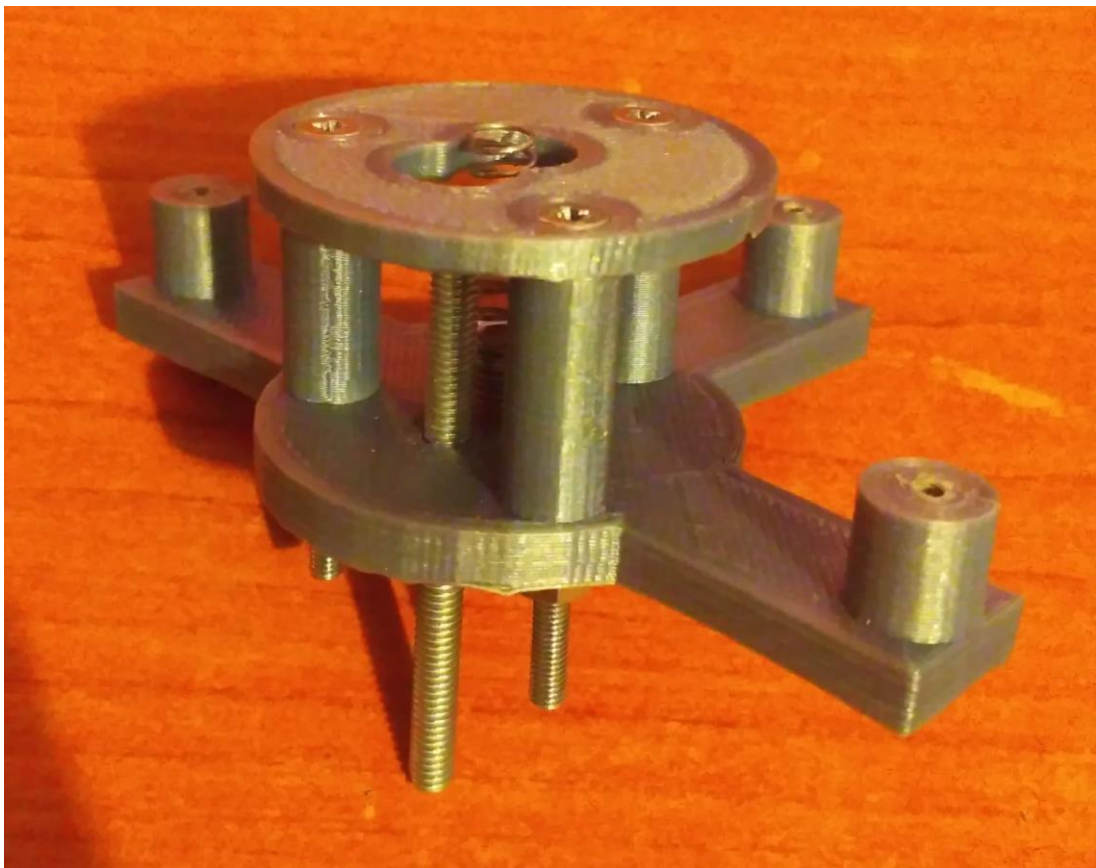
Attachment 7 – Servos and container slides



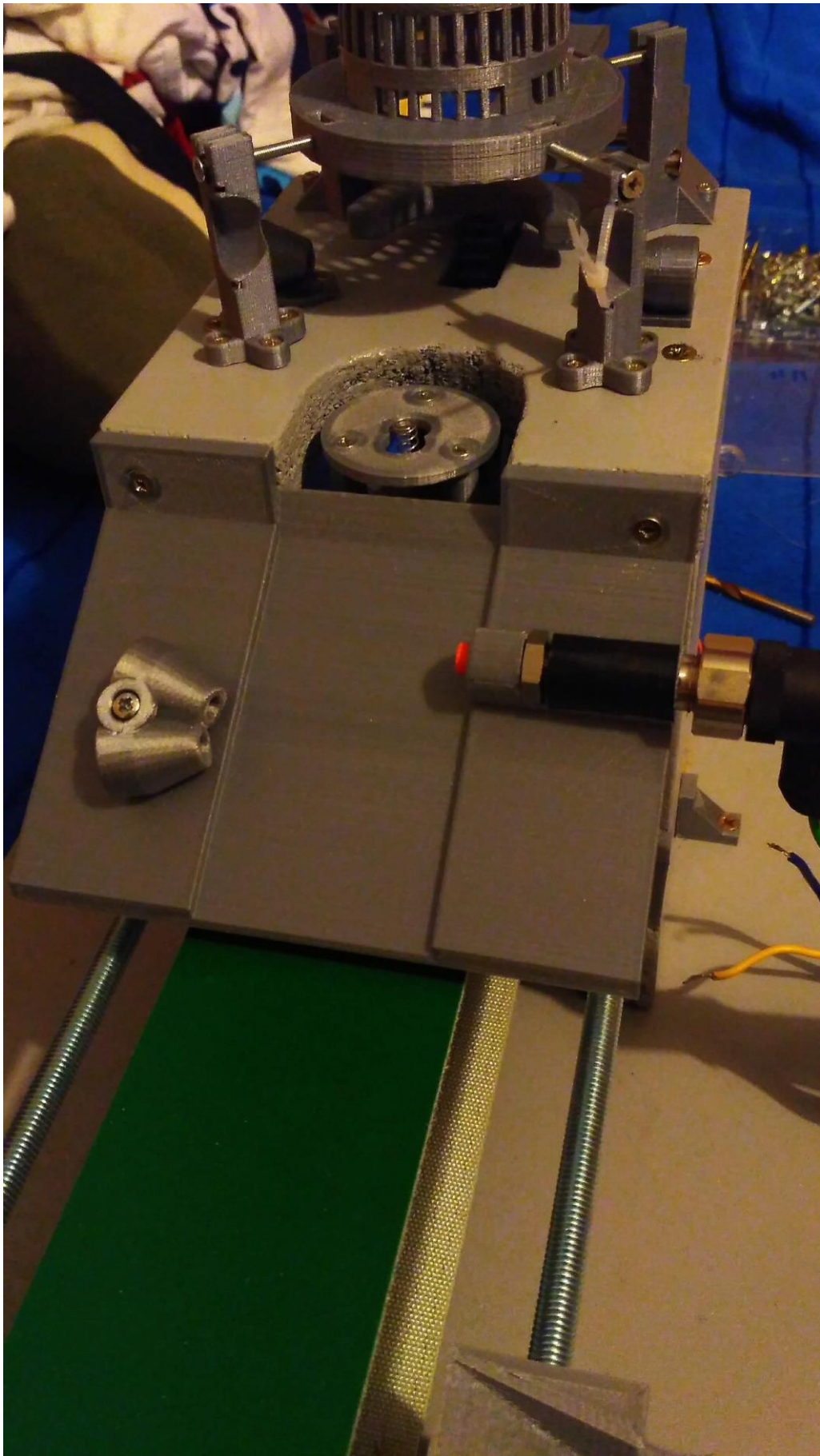
Attachment 8 – Last optical barrier




Attachment 9 – Scheme of an optical barrier



Attachment 10 - Scale



Attachment 11 - Slide

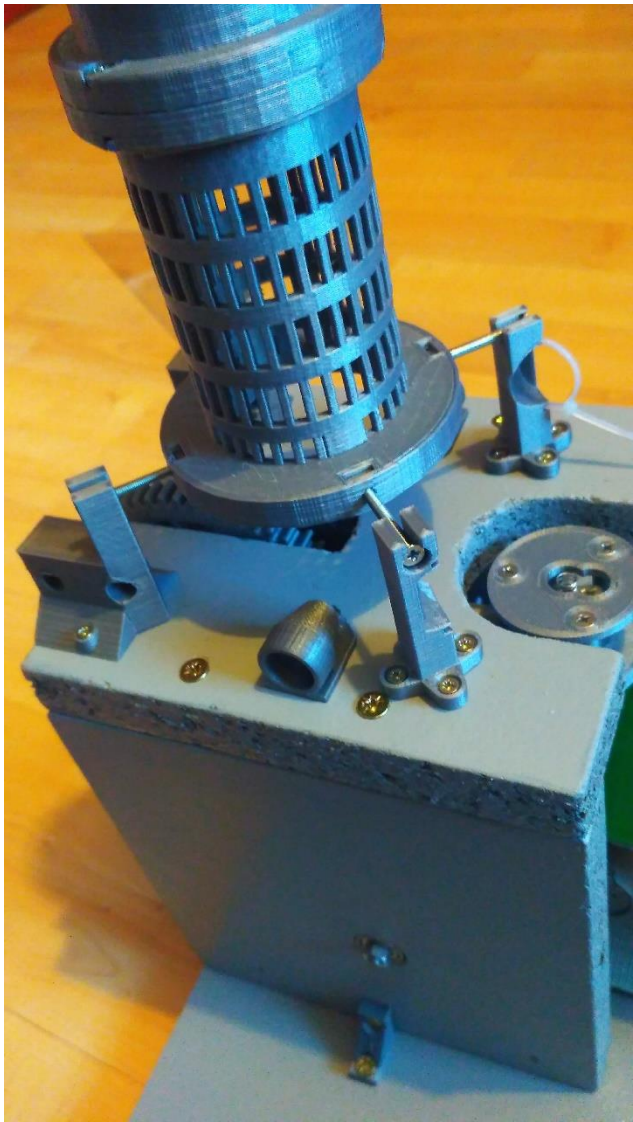


The screenshot shows the Arduino IDE interface. The title bar reads "Arduino_Final | Arduino 1.8.5". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for checking, running, serial monitor, and uploading/downloading. The main editor area shows the following code:

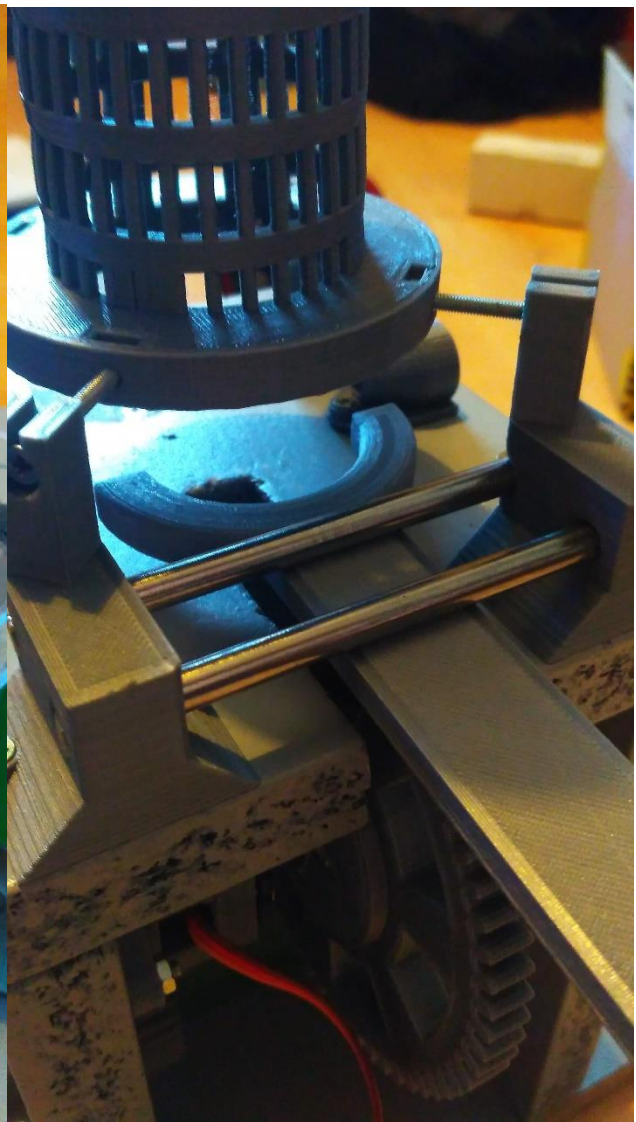
```
void dock() {  
  dockRead = analogRead(inDock);  
  dockVal = dockRead * (5.0 / 1023.0);  
  
  if (dockVal < dockComp) digitalWrite(outDock, HIGH);  
  else digitalWrite(outDock, LOW);  
}  
  
void height() {  
  heightRead = analogRead(inHeight);  
  heightVal = heightRead * (5.0 / 1023.0);  
  
  if (heightVal < heightComp) digitalWrite(outHeight, HIGH);  
  else digitalWrite(outHeight, LOW);  
}  
  
void cont() {  
  contRead = analogRead(inCont);  
  contVal = contRead * (5.0 / 1023.0);  
  
  if (contVal < contComp) digitalWrite(outCont, HIGH);  
  else digitalWrite(outCont, LOW);  
}  
  
void color() {  
  colorRead = analogRead(inColor);
```

Below the code editor is a status bar that says "Done Saving." and a black area for the serial monitor. At the very bottom, a footer bar displays "49" on the left and "Arduino Pro or Pro Mini, ATmega328P (5V, 16 MHz) on COM4" on the right.

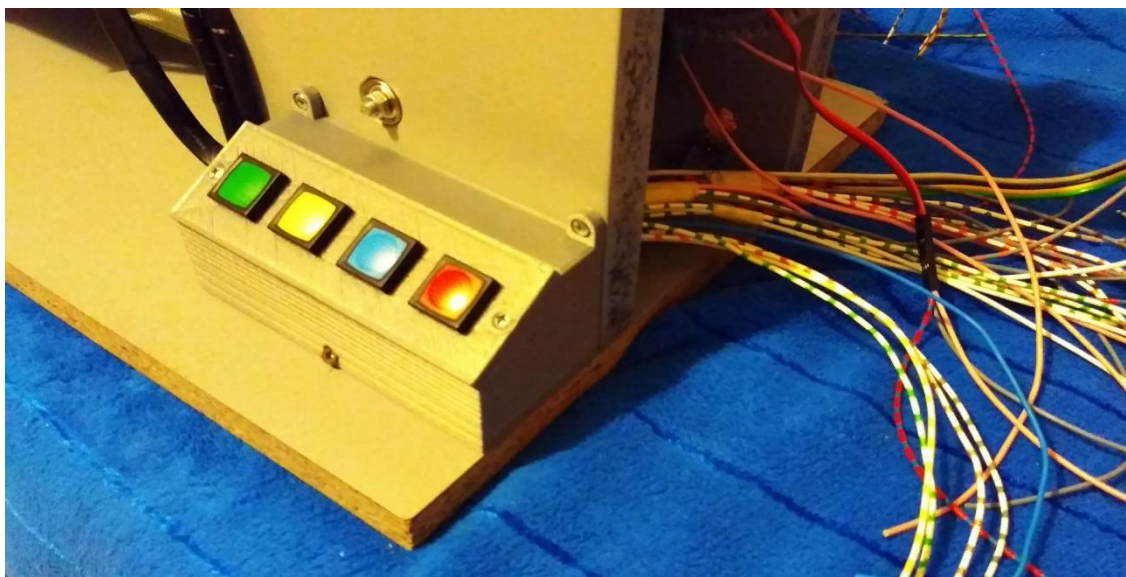
Attachment 12 – Arduino code



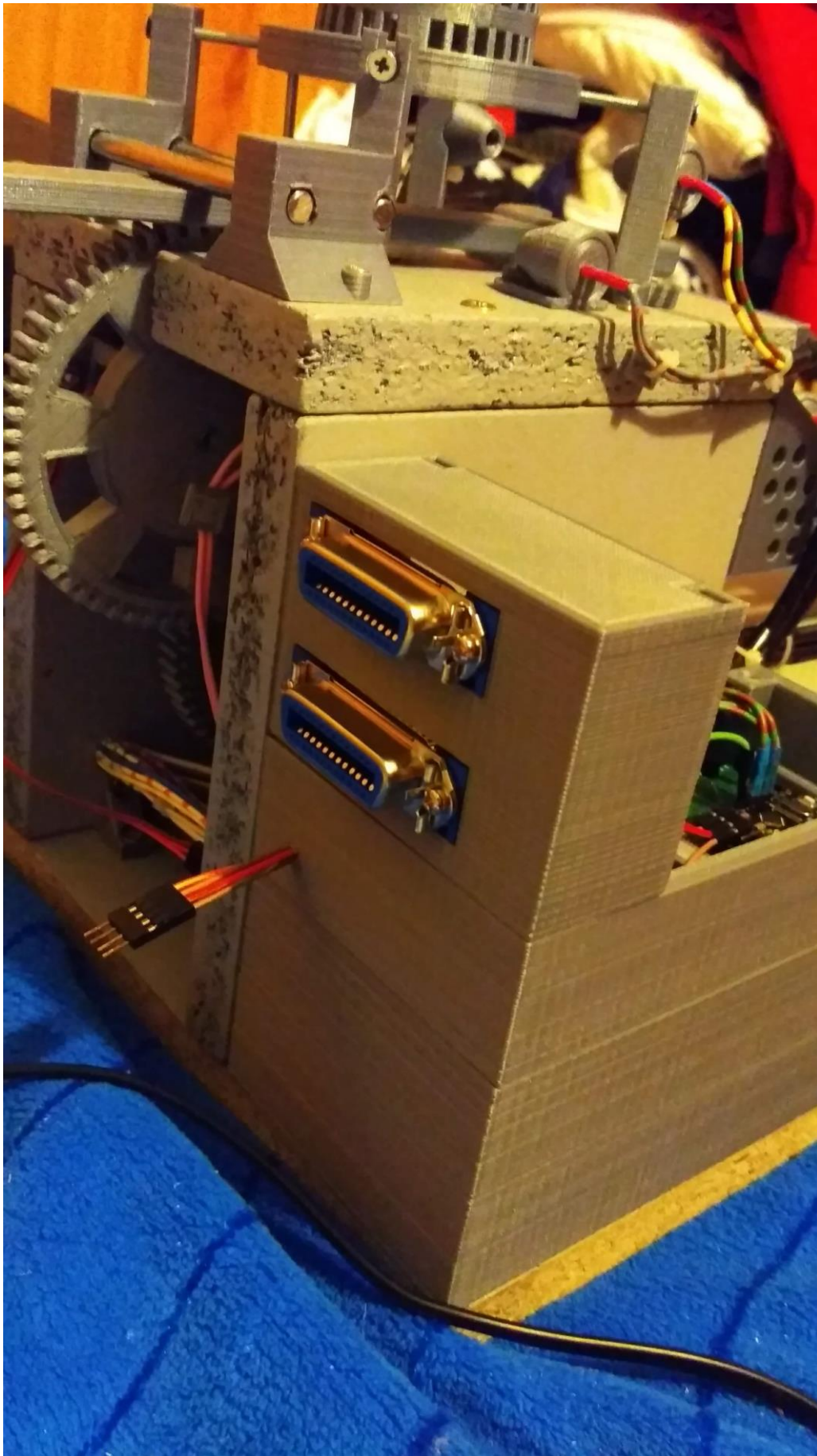
Attachment 13 – Tube and dock



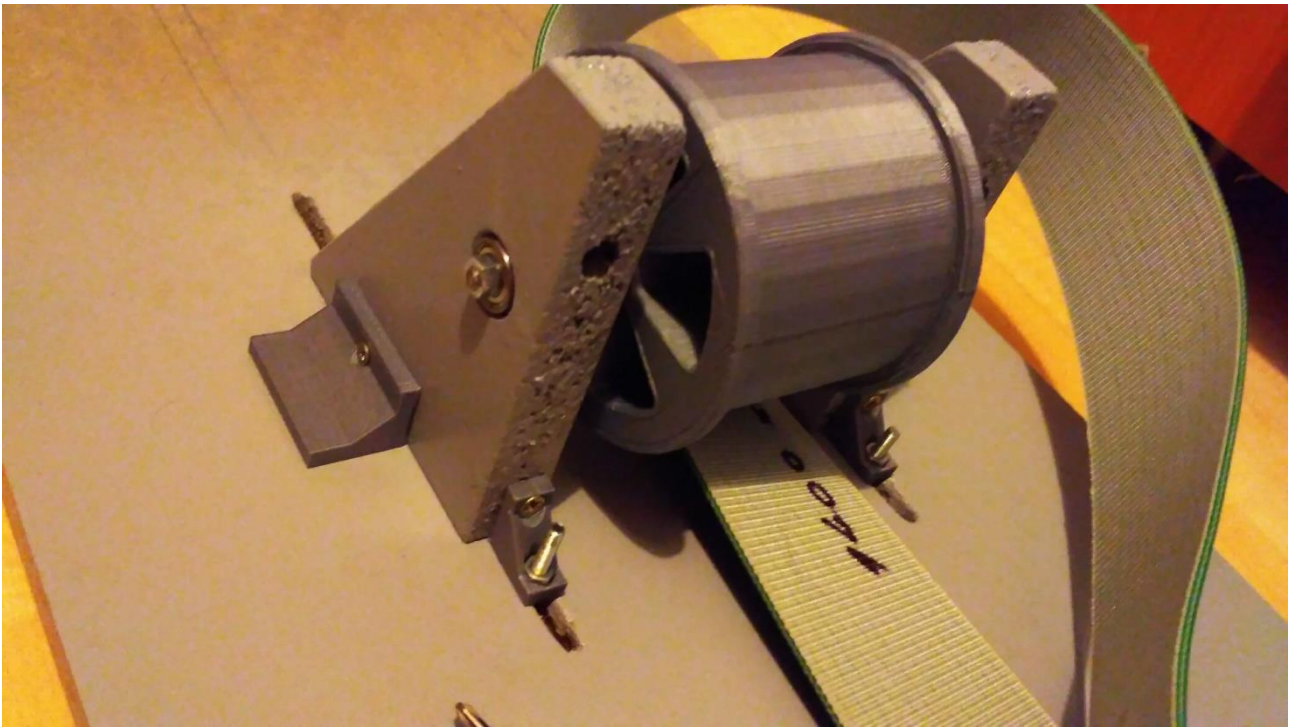
Attachment 14 – Rear legs



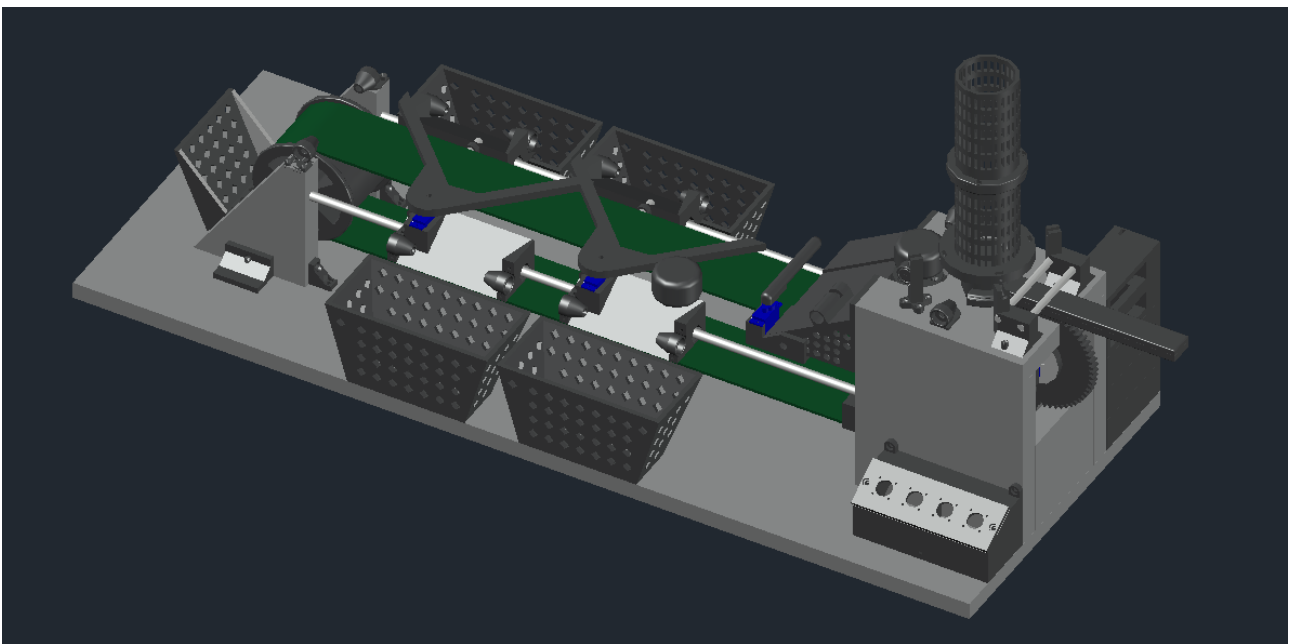
Attachment 15 - Buttons



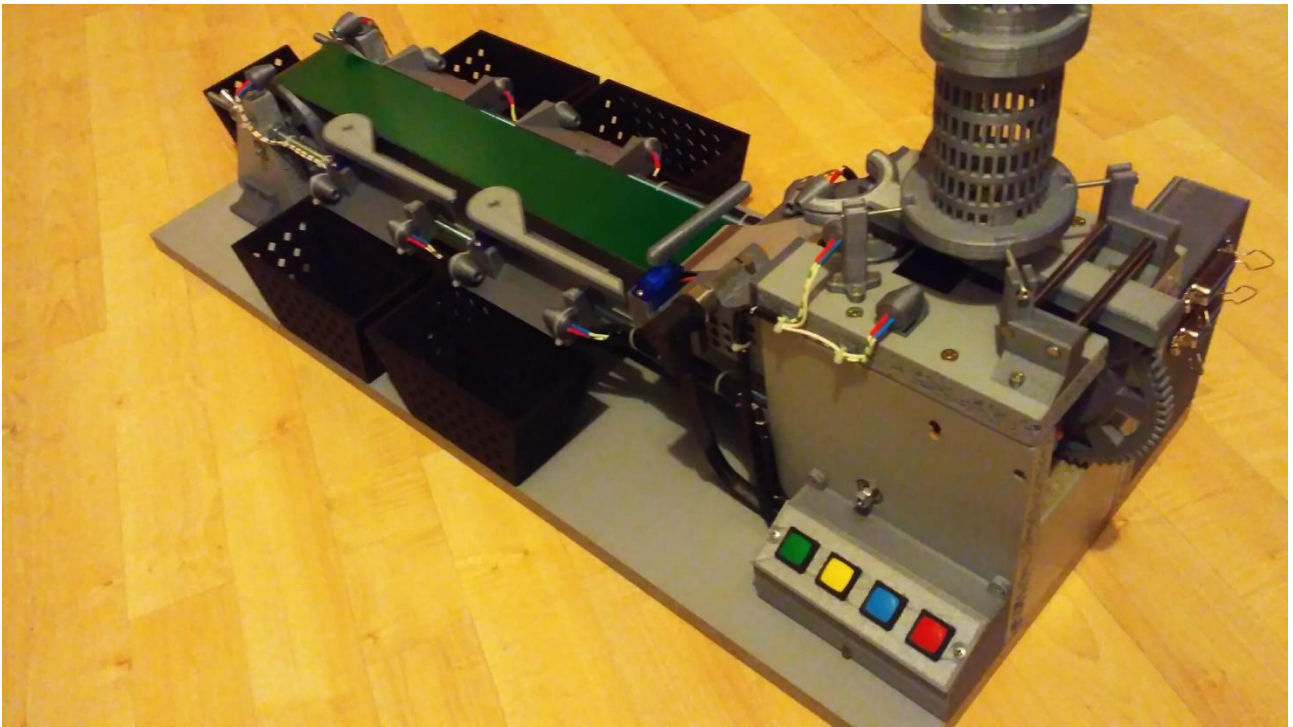
Attachment 16 – Electronics box



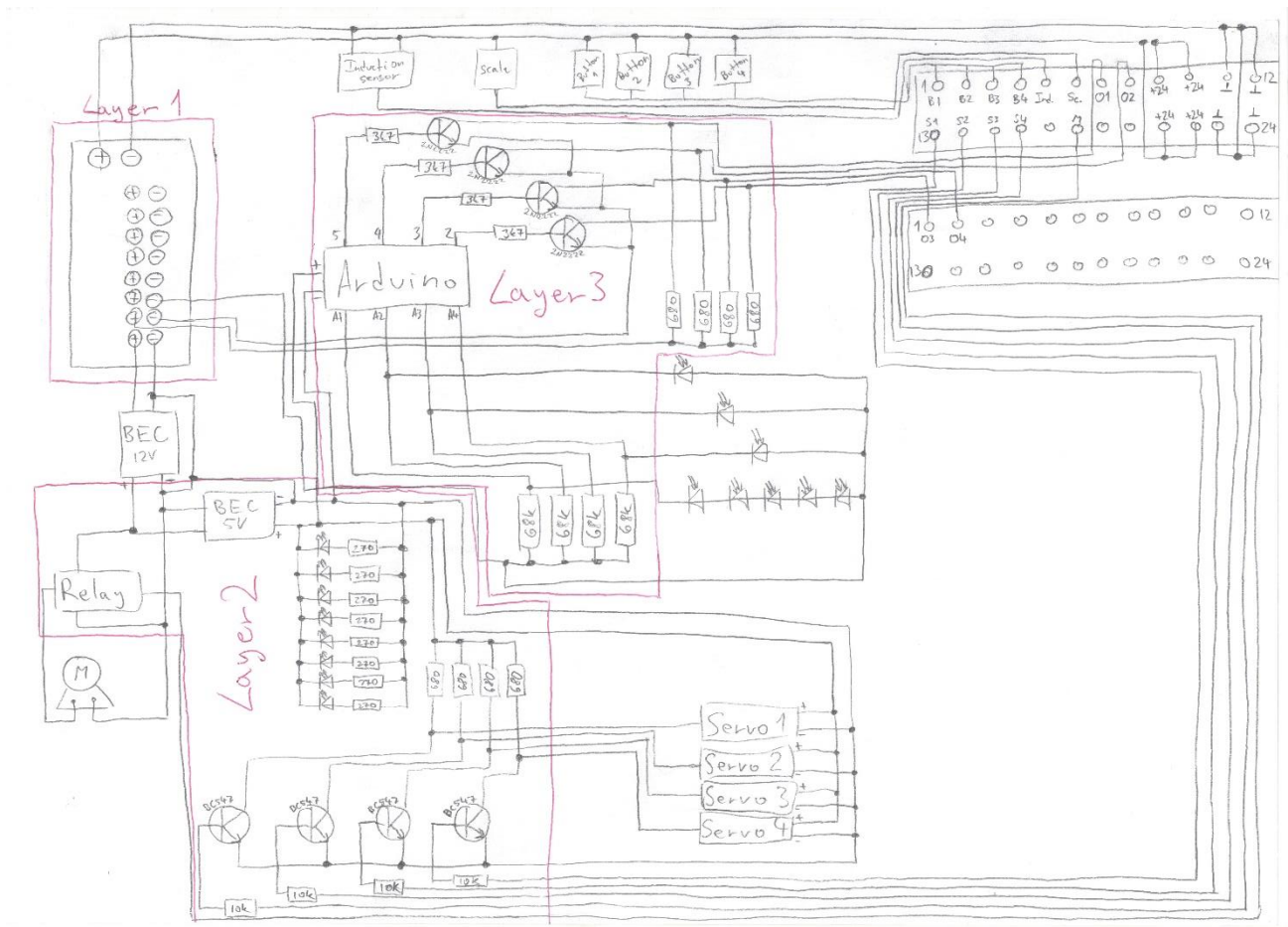
Attachment 17 – Rear rails and cylinder



Attachment 18 – 3D design of the sorting machine



Attachment 19 – Final version of the sorting machine



Attachment 20 – Brief scheme of the sorting machine



Střední průmyslová škola elektrotechnická Havířov,
Makarenkova 1, Havířov, CZECH REPUBLIC
Zespół szkół technicznych, Rybnicka 44, Mikołów, POLAND

TEMPERATURE REGULATOR



Co-funded by the
Erasmus+ Programme
of the European Union

Table of content

1 Introduction	4
2 Theoretical part	5
2.1 Market analysis	5
2.2 Temperature regulation.....	6
2.2.1 Two-point regulation	6
2.2.2 Proportional regulation	8
2.2.3 Proportional integrating regulation.....	8
2.3 Temperature sensor.....	9
2.4 Pulse width modulation	9
2.5 Heating element.....	10
2.5.1 Power transistor	10
2.5.2 Resistance wire	11
2.5.3 Power resistor	11
2.5.4 Group of power resistors	12
2.6 Transistor	13
2.7 Chamber	15
2.8 Cooler.....	16
2.9 Rotary encoder	16
3 Financial analysis.....	17
4 Solving hardware and software.....	18
4.1 Programming language	18
4.2 Code	18
4.2.1 Main menu.....	18
4.2.1.1 Function setval	19
4.2.1.2 Menu function.....	19
4.2.2 Two-point regulation	21
4.2.3 Proportional regulation.....	23
4.2.4 Proportional integrating regulation.....	25
4.3 Interface.....	26
4.3.1 Interface on Arduino	27
4.3.2 Setting network.....	27
4.3.3 Web interface	28
4.3.4 Output file.....	28
4.3.5 Transferring measuring files	29
4.4 Hardware design.....	29
4.4.1 Microcontroller.....	30
4.4.1.1 Arduino pinout	30
4.4.1.2 Front panel connector	30
4.4.1.3 A heating control panel connector	31
4.4.1.4 PCB.....	31
4.4.1.5 Realisation	32
4.4.2 Heating and cooling circuit.....	33
4.4.2.1 Power supply.....	33
4.4.2.2 Heating control PCB	34
4.4.2.3 Heater PCB	35
4.4.2.4 Realisation	36
4.4.3 Front panel.....	37
4.4.3.1 PCB.....	38

4.4.3.2 Realisation	39
5 Chamber and case for controller	40
5.1 Case for microcontroller.....	40
5.2 The case for the heating element.....	41
5.3 The case for the power supply.....	43
6 Control measures.....	44
6.1 Two point.....	44
6.2 Proportional regulator	45
6.3 Proportional integrating regulator	46
7 Tuning the regulator	48
7.1 Check accuracy of constants	50
7.1.1 Proportional regulation	50
7.1.2 Proportional integrating regulation.....	51
8 Final model	52
8.1 Controller of regulator.....	52
8.2 Chamber with the heating element.....	53
9 Conclusion	55
10 References	56

1 Introduction

This project arises in cooperation with Polish and Czech student and tutors. Our purpose is to build a temperature regulator, which can demonstrate three basic types of regulation. These types are two-point regulation, proportional regulation and proportional integrating regulation.

During the project, we use a popular platform, Arduino Mega. This platform includes a processor ATmega2560, 54 input/output pins thereof 15 can be used with PWM.

Enlargement of this platform could be shields. We use two shields simultaneously. One shield will be ethernet card, to communicate via the internet or host some server. Next shield will be TFT LCD, to display information about the program. Both of these shields have an SD card slot with the driver.

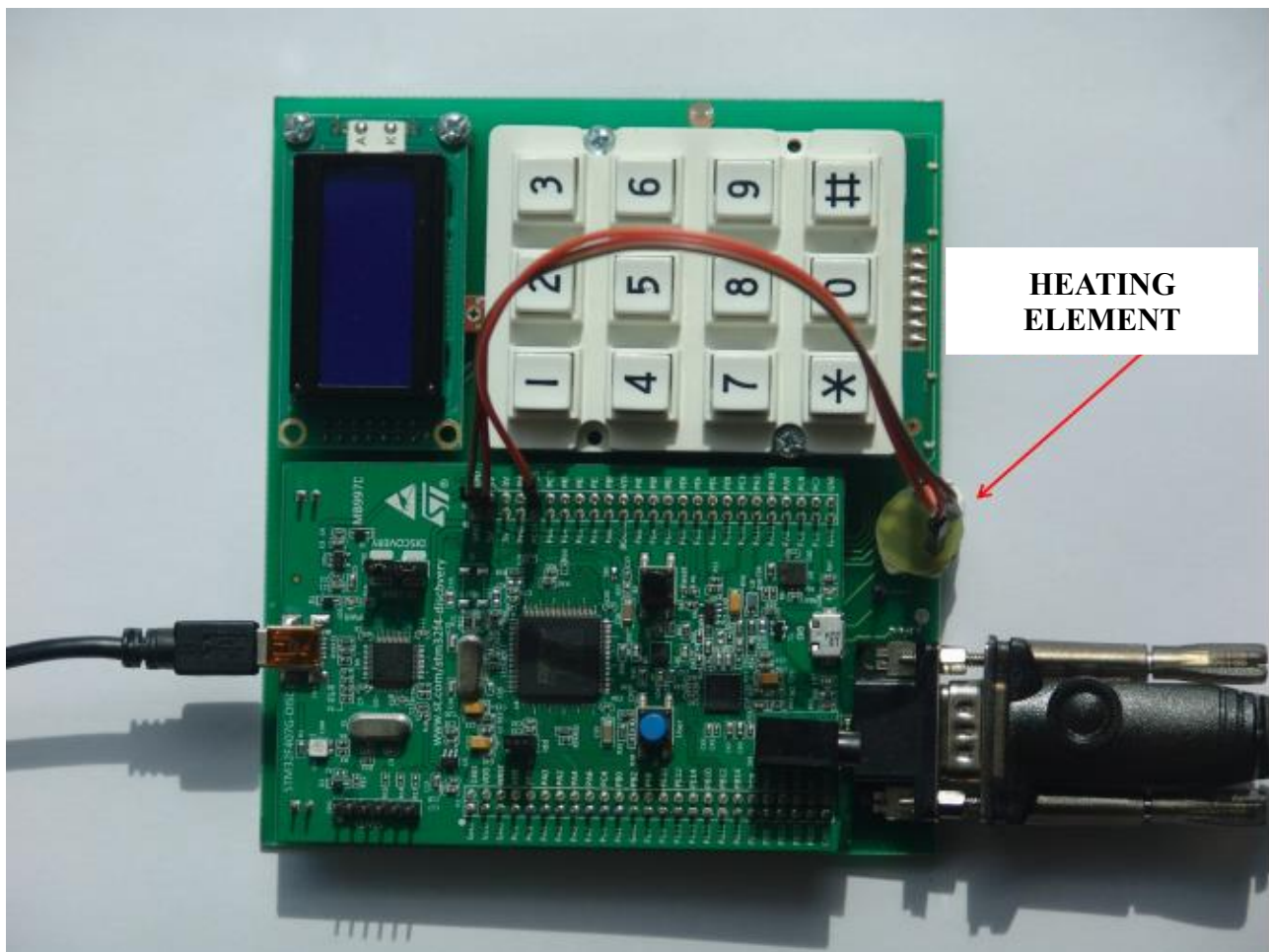
Because this model should be used as demonstrating model, it must be small, light and easy to use. This is the reason why we won't use some big shields or enlargements. These things include a lot of for us unnecessary things, and it will be unused.

2 Theoretical part

To make a school model we must pay attention to some factors. At first, the model of the regulator should be small. In another case, there should be a chamber with a heater. Next factor is the speed of regulation. Regulation shouldn't be fast, but also shouldn't be very slow. We must prototype the system given to these factors.

2.1 Market analysis

When we tried to search some same models on the internet, we must end with negative results. We couldn't find anything like this. We found the same model at Czech school but build on ARM-based discovery board. This solution has many disadvantages; for example, the big measuring error caused by the absence of chamber, no control interface (only serial port) and small heating power.



Picture 1 - School model of regulator

2.2 Temperature regulation

The main idea of temperature regulation is to keep the temperature in room/chamber constant. We have many types of regulation, but our work analyses only three types. For most types of regulations, we have defined the necessary variables:

Table 2.1 -List of variables using in the regulations

Name of variable	Symbol	Calculation
Set temperature	w	Defined by user
Measured temperature	y	Measured by sensor
Control error	e	$e=w-y$
Actuating variable	u	Dependant on the type of regulator



Picture 2 - Block diagram of temperature regulation

2.2.1 Two-point regulation

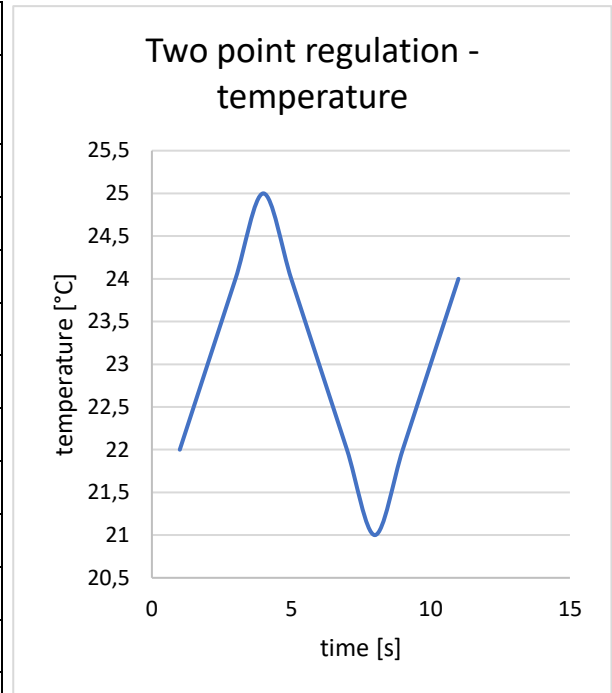
Two-point regulation is the simplest type of regulation. This system has only two states – on and off. For this type of regulation, we need to know only two parameters – set temperature and hysteresis. The principle of regulation is, that heater is on to while when the temperature is higher than the set temperature. When temperature decrease (measured temperature is lower than the set temperature minus hysteresis), the regulator starts heating. For demonstration, we made a simple graph.

On this type of regulation applies:

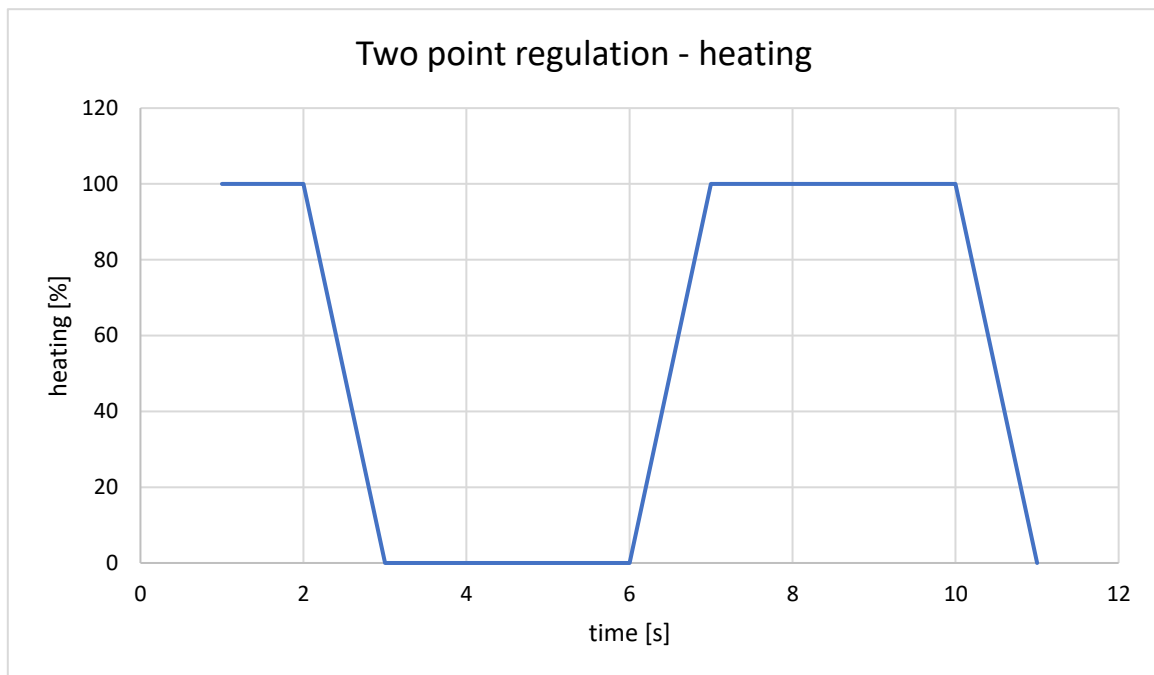
$$u = \begin{cases} u_{max} & \text{for } e > 0 \\ u_{min} & \text{for } e < 0 \end{cases} \quad (1)$$

Table 2.2 - Two-point regulation

Set temperature = 24°C		Hysteresis = 2°C	
Time [s]	Actual temperature [°C]	Control error [°C]	Heating [%]
1	22	2	100
2	23	1	100
3	24	0	0
4	25	-1	0
5	24	0	0
6	23	1	0
7	22	2	100
8	21	3	100
9	22	2	100
10	23	1	100
11	24	0	0



Picture 3 - Time dependence of temperature in two-point regulation (theoretical)



Picture 5 - time dependence of heating in two-point regulation (theoretical)

2.2.2 Proportional regulation

The proportional regulation (P regulation) amplify control error. This type of regulation is specific; therefore, we afford a table of used variables/constants.

Table 2.3 - list of variables using in the proportional regulation

Name of variable	Symbol	Calculation
Set temperature	w	Defined by user
Measured temperature	y	Measured by sensor
Control error	e	$e=w-y$
Actuating variable	u	Dependant on the type of regulator
Reinforcement of regulator	K	Set by engineer
-	u_b	$u_b=(u_{min}+u_{max})/2$

When we increment K, the control error is smaller, but we must pay attention to the stability of the regulator. The number u_b can add to the calculation of actuating variable what cause zero control error. When the control error is so big, the regulator works like a two-point regulator.

$$u = \begin{cases} u_{max} & \text{for } Ke + u_b > u_{max} \\ Ke & \text{for } Ke + u_b \in \langle u_{min}, u_{max} \rangle \\ u_{min} & \text{for } Ke + u_b < u_{min} \end{cases} \quad (2)$$

2.2.3 Proportional integrating regulation

The proportional integrating regulation (PI regulation) is a combination of proportional and integrating regulation. The advantage of this regulation is that control error is zero automatically

$$u(t) = K \left[e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau \right] \quad (3)$$

Because PI regulation use integration – this operation we couldn't do at any microprocessor, we must convert the equation to discrete form.

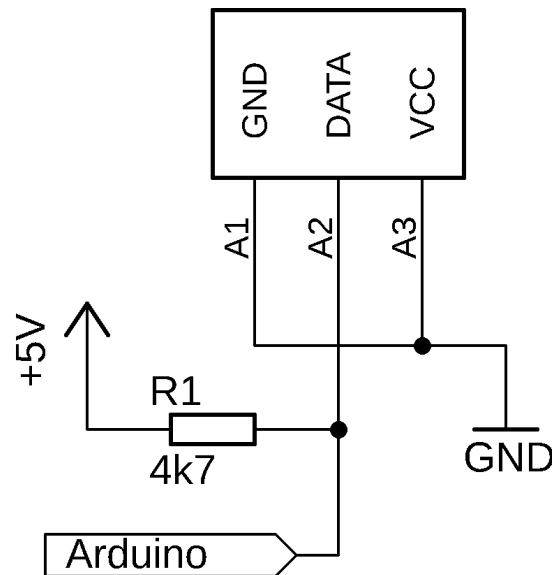
$$u = u_0 - Ke_0 + (T\tau + K)e \quad (4)$$

Table 2.4 - list of added variables in PI regulation

Name of variable	Symbol	Calculation
Old actuating variable	u_0	Save old actuating variable
Old control error	e_0	Save old control error
Integrating time constant	τ	Set by engineer
Time	T	The time between 2 measures

2.3 Temperature sensor

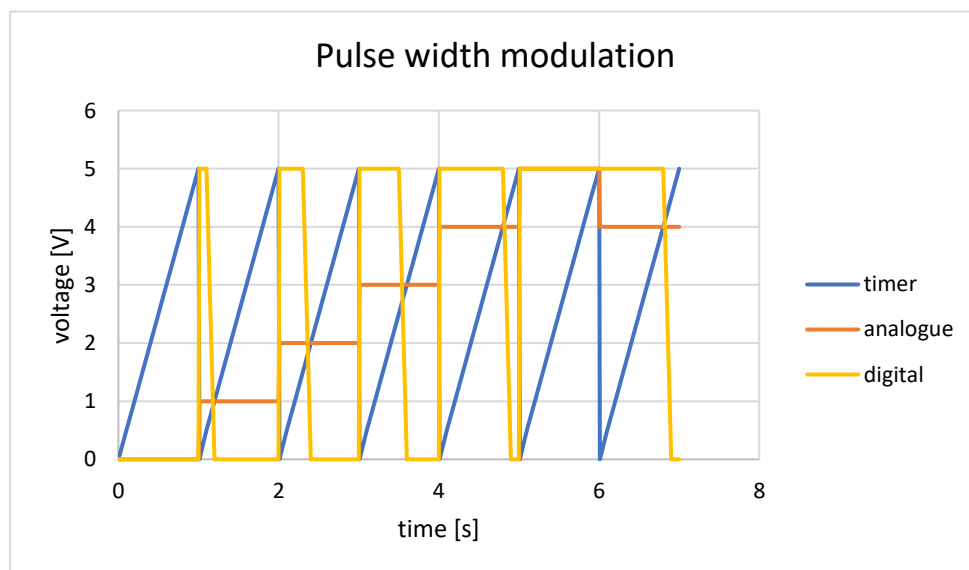
For measuring temperature, we choose from the offer of digital sensor DS18B20. This sensor could communicate via one wire, use 5V as a power supply and doesn't distort measures when the supply voltage is lower than 5V. Every measure sensor sends 16 bits of information. This type of temperature sensor could measure from -55°C to 125°C . For the right function, we must connect pull up resistor.



Picture 7 - Simple circuit with DS18B20

2.4 Pulse width modulation

Shortly PWM. It is a simple modulation, where the level of input voltage represents the time of logic value 1. The PWM signal at Arduino has a default frequency approximately 500 Hz. The timer generates saw wave, and the PWM is logical 1 from the time when saw wave is lower than the analogue signal. When the analogue signal is higher the value of the output signal is logical 0.



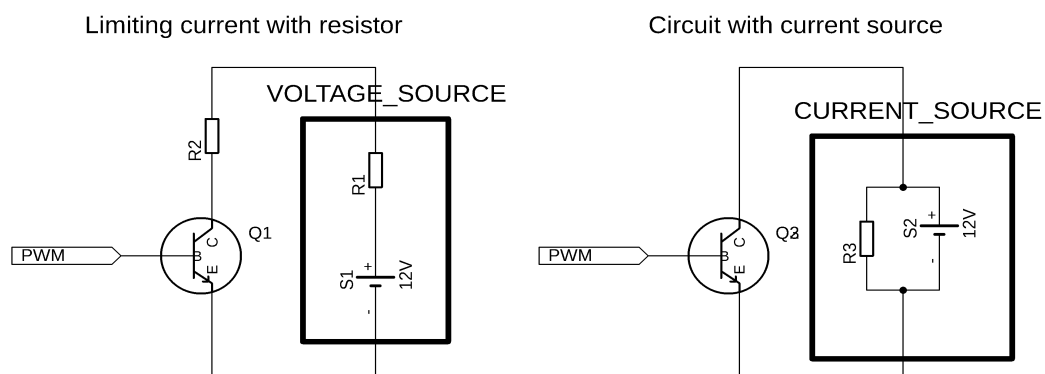
Picture 8 - Pulse width modulation

2.5 Heating element

The heater element should be heated to 50°C within 5 minutes, and the maximum temperature at 100% heating power should be 70°C. From practice we know, that most of the electronic components in dependence on current makes heat. In ordinary life we usually solve heat makes by semiconductors, that's the reason, why we started with transistors. The second idea is to use small resistance to limit current and connect a power resistor.

2.5.1 Power transistor

Using power transistor was effective in heating, but when we need to cool the circuit it wasn't fast enough. Another problem with this solution is that we need to limit the current going through the transistor. We had two methods: first is to connect the resistor to circuit what makes the voltage drop and cause heating of resistor, the second solution is to use the current source, but we weren't sure about using it in combination with pulse width modulation.



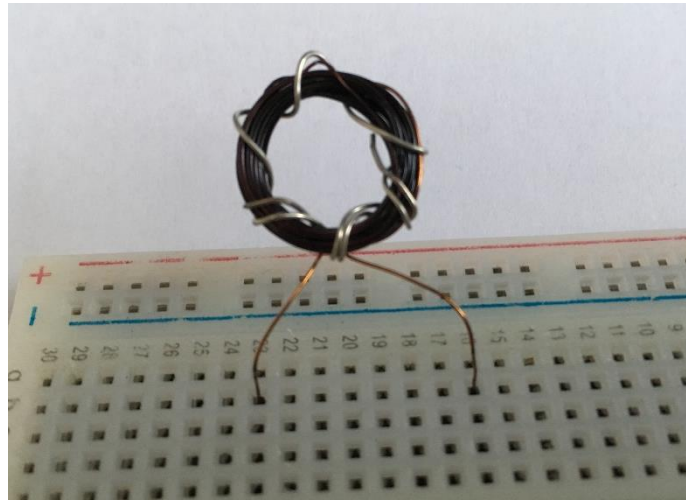
Picture 9 - Power transistor as a heater



Picture 10 - Power transistor with mounted heatsink

2.5.2 Resistance wire

The second variant of the heating element was resistance wire. This method is useful in cooling, but heating wasn't as fast as we need. We could make a solenoid, but when wire starts heating the dielectric coat at wire starts to burn and makes a short circuit.



Picture 11 - Solenoid from resistance wire

2.5.3 Power resistor

The idea of a power resistor is simple. We know that on every element in circuit occurs voltage drop. These voltage drops cause losses, and these losses cause heating of the component.

To describe it, we used two resistors:

- 15Ω , 10W
- 6Ω , 50W

The current going through resistors we could calculate using Ohm's law - equation 5,6,7.

$$I = \frac{U}{R} \quad (5)$$

$$I_{15} = \frac{12}{10} = 1,2[A] \quad (6)$$

$$I_6 = \frac{12}{6} = 2[A] \quad (7)$$

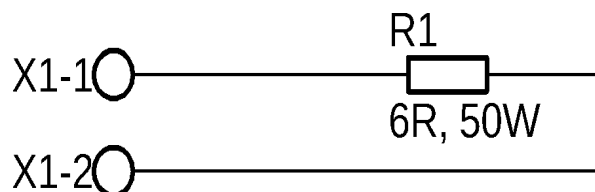
Calculating power – equation number 8,9,10.

$$P = U * I \quad (8)$$

$$P_{15} = 12 * 1,2 = 14,4[W] \quad (9)$$

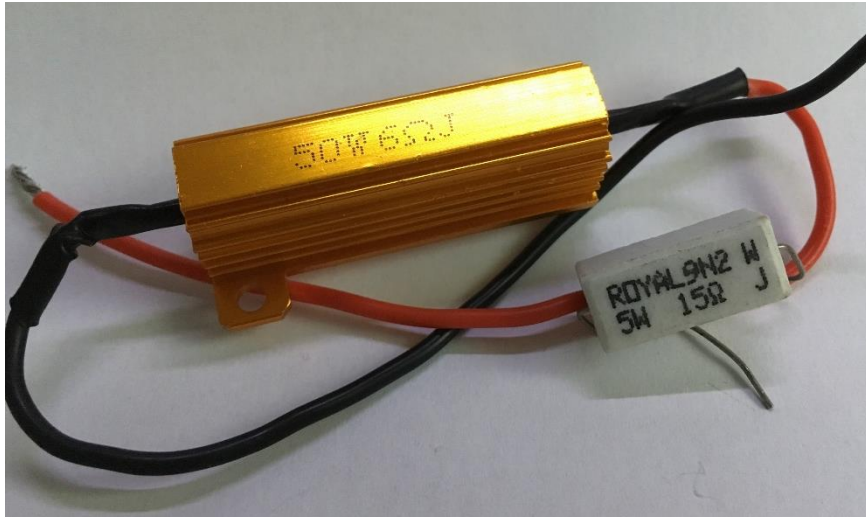
$$P_6 = 12 * 2 = 24[W] \quad (10)$$

From these calculations, we could see, that in some cases with these types of resistors we could have power for heating, without overloading resistors.



Picture 12 – Power resistor as a heater

After some tests, we conclude, that these power resistors are made in a ceramic package or heatsinks. This cause a very long cooling of them. This problem excludes this variant.



Picture 13 - Demonstrating of packages of power resistors, the orange one 50W, white one 5W

2.5.4 Group of power resistors

The first idea was, that when we need some time for cooling one power resistor heated to some temperature when we use four resistors connected parallel, the maximum temperature will be divided to these resistors and time to cooling will be shorter. We could try four 47Ohm resistors. The total resistance of the circuit we could calculate – equation 11.

$$R = \left(\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4} \right)^{-1} = \left(\frac{4}{47} \right)^{-1} = 11,75[R] \quad (11)$$

Via this resistance will go current I.

$$I = \frac{U}{R} = \frac{12}{11,75} \doteq 1[A] \quad (12)$$

Because all values of the resistor are the same, the current will be divided by some resistors.

$$I_1 = I_2 = I_3 = I_4 = \frac{I}{4} = \frac{1}{4} = 0,25[A] \quad (13)$$

The parallel connection should cause, that every one of the resistors connected to the circuit will be heated less, and cooling of them should be faster.

This idea is the truth, but we need to change resistors because cooling is also too long. This is caused by the ceramic package, which is quickly heated and cooled in an extended amount of time. Also, we decided that we could have a 12V, DC power supply with minimally current 2A, which will allow us to heat chamber faster.

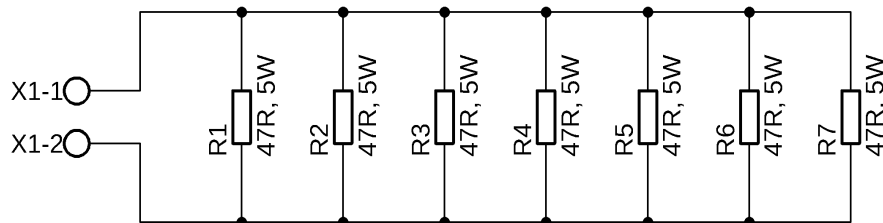
After problems with the ceramic package of power resistor, we decided to try another material of package. We choose the carbon package. After some tests with one resistor, we know that cooling is beneficial, and heating is also very fast.

Following our decision, we could have a more significant current. That's the reason, why we connect seven resistors to decrease total resistance of circuit – equation 14.

$$R = \left(\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4} + \frac{1}{R_5} + \frac{1}{R_6} + \frac{1}{R_7} \right)^{-1} = \left(\frac{7}{47} \right)^{-1} \doteq 6,7[R] \quad (14)$$

$$I = \frac{U}{R} = \frac{12}{6,7} \doteq 1,79[A] \quad (15)$$

$$P = U * I = 12 * 1,79 \doteq 21,48[W] \quad (16)$$



Picture 14 - Group of power resistors as a heater

After hardware tests, we have a perfect heater, which could heat a chamber for more than 70°C within less than 5 minutes.



Picture 17 - First model of heater for testing heating in chamber

2.6 Transistor

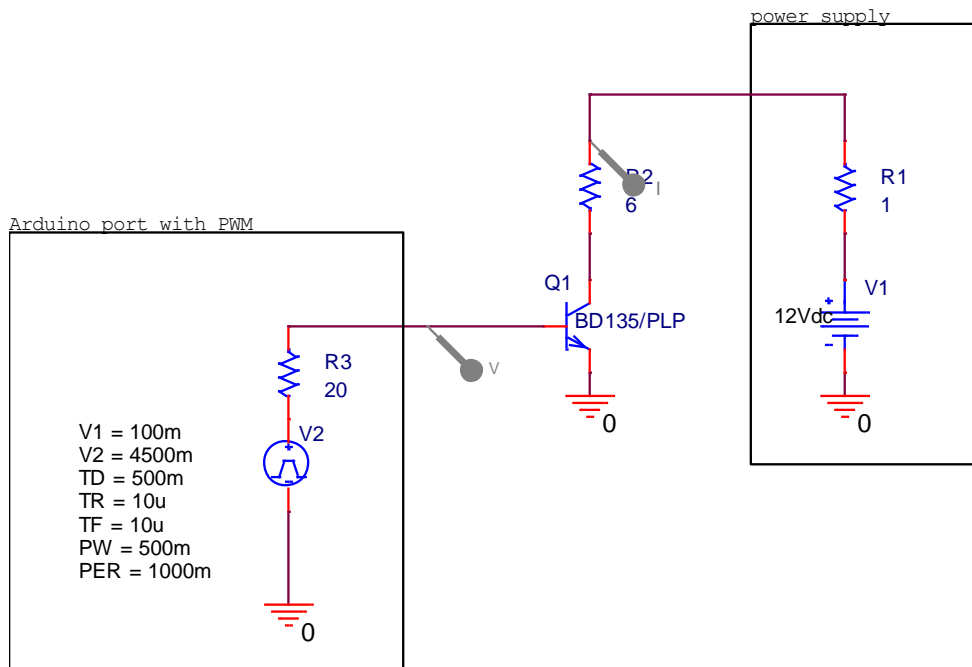
We decided to regulate power heating with pulse width modulation. Because we need to heat, we must switch the big power. We could choose between a relay or transistor. The relay has a limited number of switches to the minute. From Arduino data-sheet we know that the frequency of PWM is approximately 500 Hz. For the standard relay, this frequency will be problematic. The transistor is also in some points of view problematic, but for our use is ideal. The frequency 500 Hz isn't problematic, and we could limit the current going via PN junction. We found transistor BD 139, which have collector current is 1,5A and in pulse mode collector current is 3A.

Table 2.5 - Parameters of the transistor BD 139

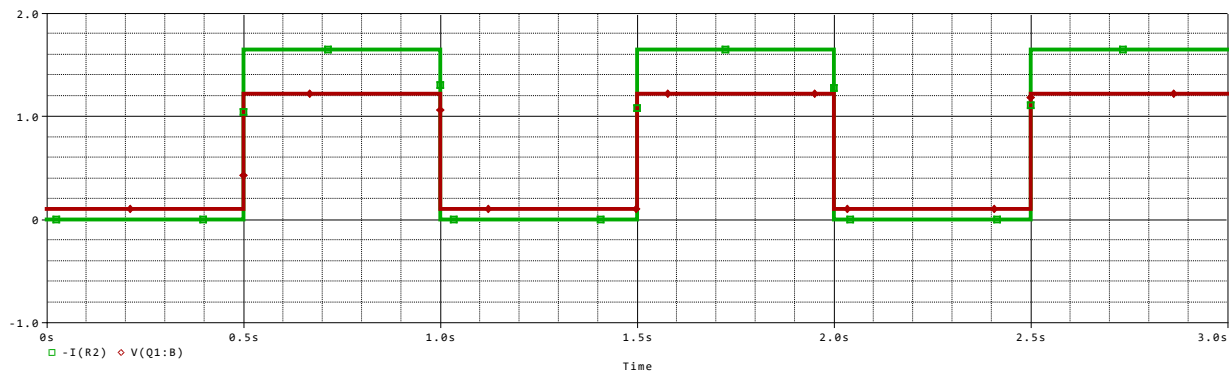
I_C	1,5A
I_{CM}	3A
P_{TOT}	12,5W
V_{CEO}	80V
V_{CBO}	80V
h_{21E}	100-250

Source: official datasheet, <https://www.st.com/resource/en/datasheet/cd00001225.pdf>

Now we use equation 15, where we calculated, that maximally we have in circuit 1,79A. The transistor could switch in pulse mode (we use PWM), maximally 3A. That means that this transistor will be ideal for switching as a heater as a fan (maximal current is many times lower than current going through heater).



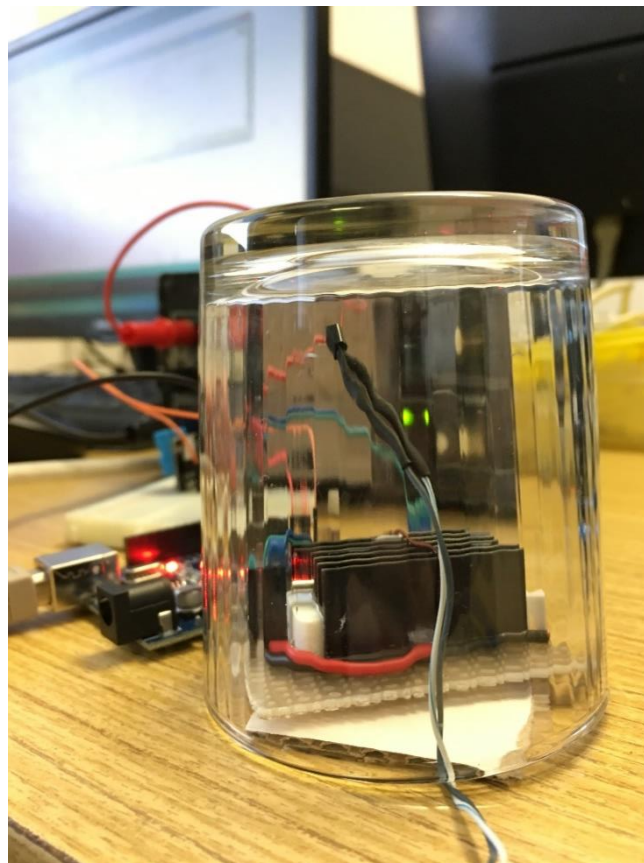
Picture 15 - Circuit with switching transistor



Picture 16 - Simulation of circuit with switching transistor

2.7 Chamber

Because it should be a school model, it should be safe. That is the reason, why we build a closed chamber. Regulation could be slower in comparison with natural cooling. From the first hardware tests, we know that the chamber shouldn't be smaller than 200 ml but shouldn't be too big. In case, where the chamber is too big regulation works like with natural cooling – what is a correct function, but after some measures are impossible to cool it. On the other side, when the chamber is smaller than 200 ml, the chamber is heated in seconds and cools tens of minutes.

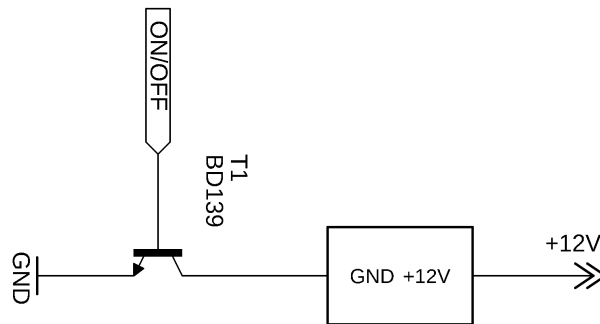


Picture 17 - First hardware test with group of power resistors with heatsink (chamber - 200ml glass)

Because the temperature in the chamber could be over 100°C, we must choose thermal resistive material. This is the reason, why we couldn't print it on the 3D printer.

2.8 Cooler

To make measuring faster and more truthful, we add a cooling circuit. It consists of one transistor for switching cooler, small cooler and indicating LED. After the end of every measuring the cooler is switched on and cooling while user set parameters of next measuring or downloading measuring files.

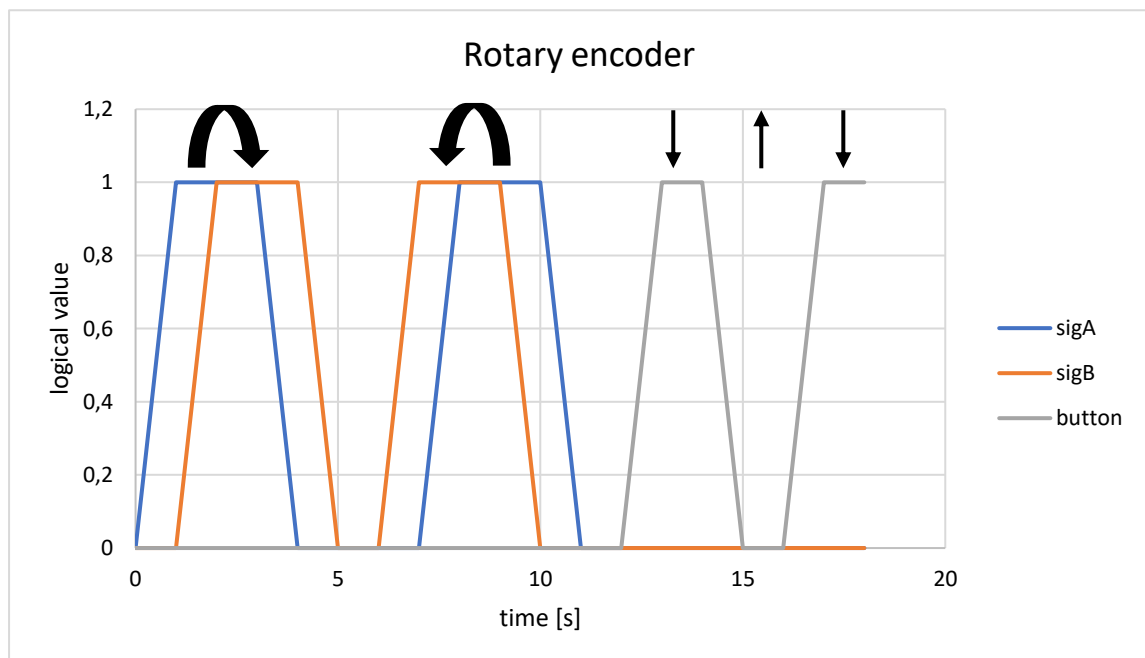


Picture 18 - Circuit with fan

2.9 Rotary encoder

The primary requirement of the project is to demonstrate regulations. For demonstration is needed to change some parameters quick and easily. We have more options for example keyboard, buttons, but we decide that the optimal will be a rotary encoder. With one hardware element, we could easily control the menu as setting variables.

For reading data from the encoder, we need 3 pins. First two pins are used to signalise the direction of rotating. The third pin is used to signalise the pressing of the button integrated under the shaft of the encoder.



Picture 19 - Time dependencies of rotary encoder

3 Financial analysis

Price recalculated with exchange rate from 5.3.2019

1 CZK = 0,039 €

1 PLN = 0,23 €

Table 3.1 - Budget for the project					
Part name	Quantity	Shop	Code	Price/psc	Total
Arduino mega precise clone	1	aruino-shop.cz	1423598706	29,70 €	29,70 €
TFT display	1	laskarduino.cz	LA141016	15,30 €	15,30 €
Arduino ethernet shield	1	laskarduino.cz	LA110002	11,24 €	11,24 €
Rotary encoder	1	gme.cz	775-029	2,70 €	2,70 €
Resistor 47R, 5W	7	elfax.cz	A0283T	0,24 €	1,68 €
Resistors 330R	4	gme.cz	119-416	0,04 €	0,16 €
Green LED diode	2	farnell.com	2112108	0,15 €	0,30 €
Red LED diode	1	farnell.com	2843622	0,22 €	0,22 €
Yellow LED diode	1	farnell.com	2112109	0,24 €	0,24 €
PCB connector 8x2	4	elfax.cz	F07010	0,31 €	1,24 €
Cable connector 8x2	4	gme.cz	800-009	0,43 €	1,72 €
Pin header 18x2	1	gme.cz	832-035	0,24 €	0,24 €
Transistor BD139	2	farnell.com	2453823	0,33 €	0,66 €
Screw clamp	2	gme.cz	841-154	0,16 €	0,32 €
Fan	1	gme.cz	625-273	2,30 €	2,30 €
Power supply 12V, 3A	1	farnell.com	2815984	19,50 €	19,50 €
Photo resistive PCB	1	farnell.com	1267744	4,77 €	4,77 €
Box for microcontroller	1	bns-sp-z-oo.business.site	G3122	11,66 €	11,66 €
Box for heater	1	bns-sp-z-oo.business.site	G3634	16,33 €	16,33 €
Flat cable	2	farnell.com	301930	1,68 €	3,36 €
Power cable	1	farnell.com	1124386	7,21 €	7,21 €
DS18B20	1	farnell.com	2515553	2,34 €	2,34 €
Total price					133,19 €

4 Solving hardware and software

Hardware for our project is major. Despite this, we started from software, little by little adding the following components and expanding the program. From the theoretical part, we have selected sensor and parts, and next peripherals are not specific for correct function.

4.1 Programming language

For programming Arduino, we used a simple language called wiring. This language based on C language. The advantage is, that for this language are many of libraries, which save our time. Most of the libraries are public, without no limitations to use.

For our program we used these libraries:

- OneWire.h – to communicate with sensor DS18B20
- DallasTemperature.h – to read data from sensor DS18B20
- Adafruit_GFX.h – for redraw bitmaps to TFT display
- Adafruit_TFTLCD.h – essential functions to work with TFT display
- SPI.h – library to communicate via SPI – SD card
- SD.h – essential functions to work with SD card
- Ethernet.h – essential functions to work with ethernet shield

4.2 Code

The whole program has many functions, but we have chosen some prime functions for this regulator. Because is the source code added to this documentation, we will show only the logic of these functions.

4.2.1 Main menu

The primary menu function is very simple function, using main commands from the adafruit library for TFT display and my functions created to simplification of the program. First, let us describe our functions used in the paragraphs below.

4.2.1.1 Function setval

This function communicates with a rotary encoder, and count pulses.

```
float setval(float value, float mn, float mx, float inc){
    s = digitalRead(enc_a);           //read value from sigA
    if ((e_l == LOW) && (s == HIGH)) { /*testing if sigA_last==0 and sigA==1 -
exclude incrementing and decrementing more times than one*/
        if (digitalRead(enc_b) == LOW) { //test if sigB==0-rot. left
            value=value-inc;             //decrement value with inc
        }
        else {
            value=value+inc;             //increment value with inc
        }
        if(value<mn) value=mn;           //limit value to minimum
        if(value>mx) value=mx;           //limit value to maximum
    }
    e_l = s;                           //save sigA to auxiliary variable
    return value;                       //return set value
}
```

Source code 1 - setval (float value, float mn, float mx, float inc) function

4.2.1.2 Menu function

In our program, we have defined one function for print menu and the second function is a combination of functions to make a full menu with selecting in the menu.

```
void menu(int pos, String val [6], int pos1, float val1, int pos2, float val2,
int pos3, float val3, int pos_o){
    int i=0;
    int place [6] = {64,112,160,208,256}; //x positions (lines)
    lcd.setTextColor(white);
    lcd.setTextSize(4);
    if(pos_o!=pos){ //test if menu wasn't printed
        for(i=0;i<5;i++){
            if(pos==i){ //if i==pos draw white rect on this line
                lcd.fillRect(0,place[i]-8,480,48,white);
                lcd.setTextColor(black); //set contrast text colour
            }
            else{
                if(pos_o==i){ //if i==last_pos draw black rect on this line
                    lcd.fillRect(0,place[i]-8,480,48,black);
                }
                lcd.setTextColor(white); //set contrast text colour
            }
            lcd.setCursor(1,place[i]); //set cursor position
            lcd.print(val[i]); //print text from string val
            if(pos1==i){ //position of val1==i ... (disable with number > 5)
                lcd.setCursor(310,place[i]);
                lcd.print(val1); //print val1
            }
            if(pos2==i){ //position of val2==i ... (disable with number > 5)
                lcd.setCursor(310,place[i]);
                lcd.print(val2); //print val2
            }
        }
    }
}
```

```

        if(pos3==i){           //position of val3==i ... (disable with number > 5)
            lcd.setCursor(310,place[i]);
            lcd.print(val3);           //print val3
        }
    }
}

```

Source code 2 - Draw a menu function

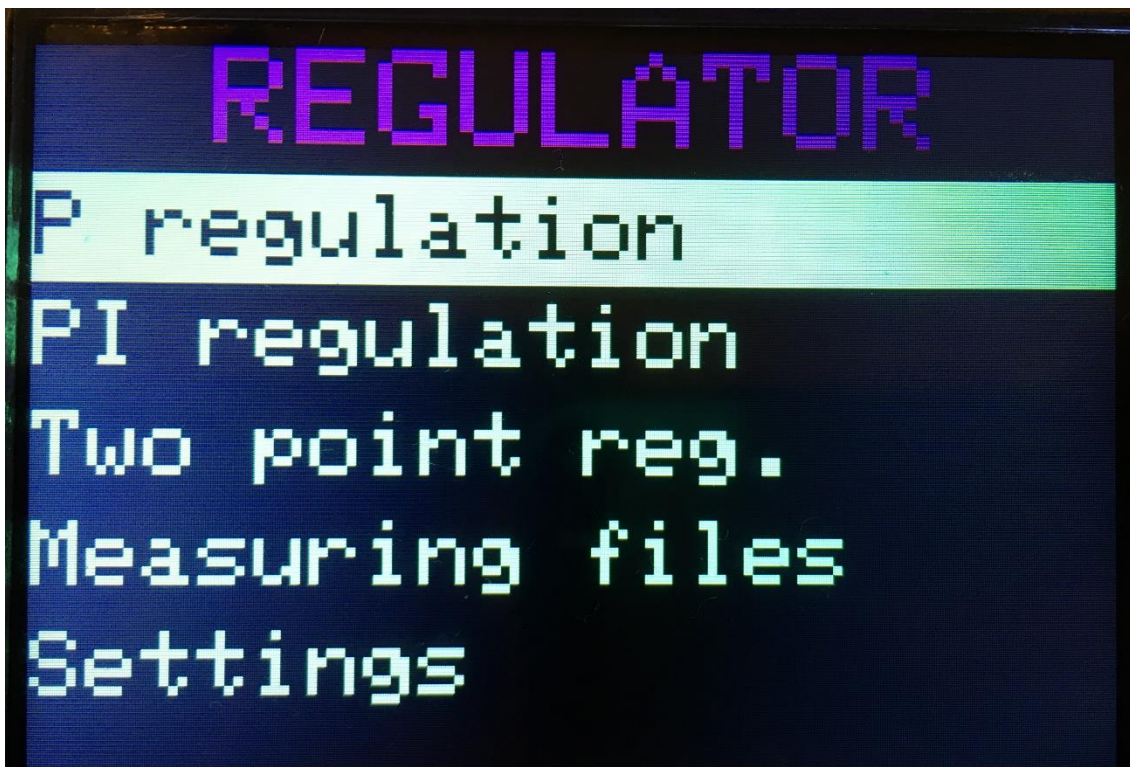
Combination of functions:

```

int menu_w_s(int pos, String val [6], int pos1, float val1, int pos2, float
val2, int pos3, float val3){
    int pos_o=150;           //set last_pos >5 for activate function
    while(1){
        menu(pos, val, pos1, val1, pos2, val2, pos3, val3, pos_o); //call menu
        pos_o=pos;
        pos=setval(pos,0,4,1);           //change position in menu
        if(digitalRead(conf)==0) break; //interrupt by user of selecting
    }
    return pos;           //return last position
}

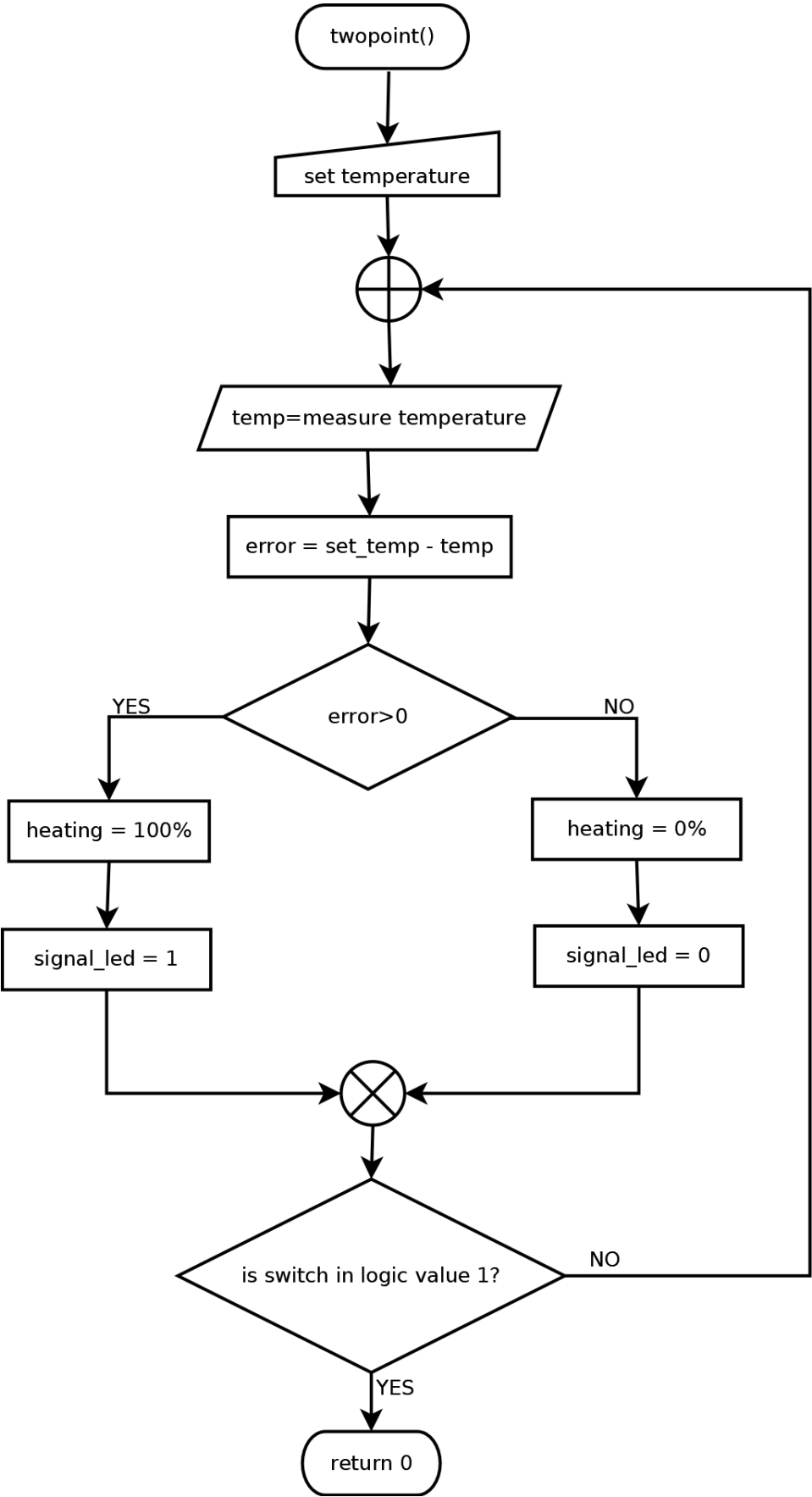
```

Source code 3 - Drawing a menu with selecting function



Picture 20 - Menu function at TFT display

4.2.2 Two-point regulation



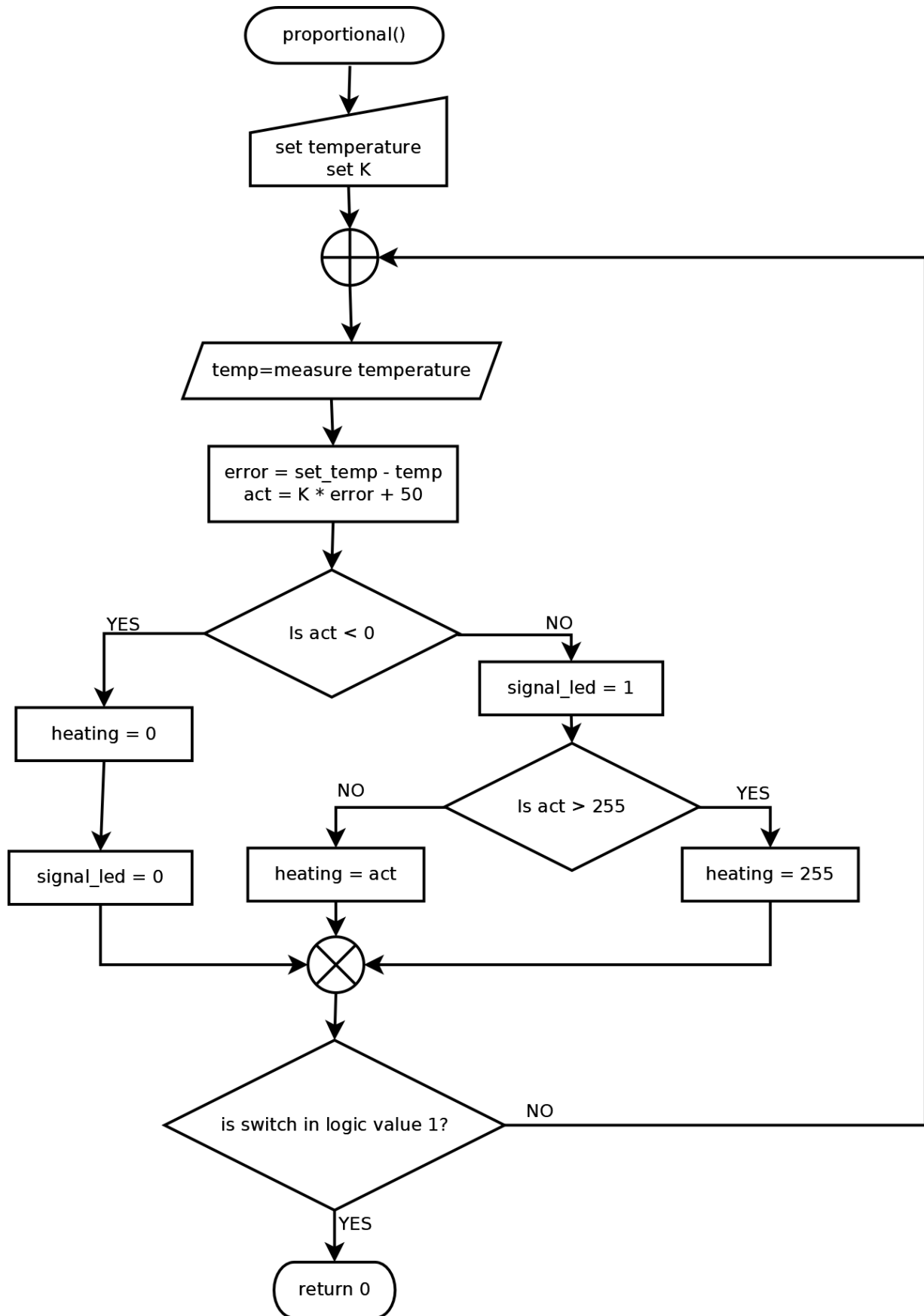
Picture 21 - Logic of two-point regulation

This function uses many functions. Therefore, there is only the major of function.

```
while(digitalRead(conf)==1){
    temperature=measure();           //measure a temperature
    e=set-temperature;               //calculate control error
    x=set-konst;                     //calculate lower limit
    if(temperature<x) action=255;    //if temp is lower than lower limit
    if(temperature>set) action=0;    //if temp is higher than set temp.
    if(temperature_o!=temperature){
        /*rewrite temperature at TFT display*/
        temperature_o=temperature;
        analogWrite(heat, action);   //send PWM to heater
        if(action>0) digitalWrite(led_h, 1); //set led diode
        else digitalWrite(led_h,0);
        /*save data to temporary files*/
    }
    web("TWO POINT REGULATION", konst, 0, set, 3); //open web server
    for(i=0; i<20; i++){             //active waiting
        if(digitalRead(conf)==0){
            digitalWrite(led_h, 0);   //switch off led
            analogWrite(heat, 0);      //switch off heater
            break;
        }
        delay(250);
    }
}
//create measuring file
createoutputfile(newfilename(), "TWO POINT REG", set, konst, 0, 0, 3);
```

Source code 4 - Two-point regulation (shortened)

4.2.3 Proportional regulation



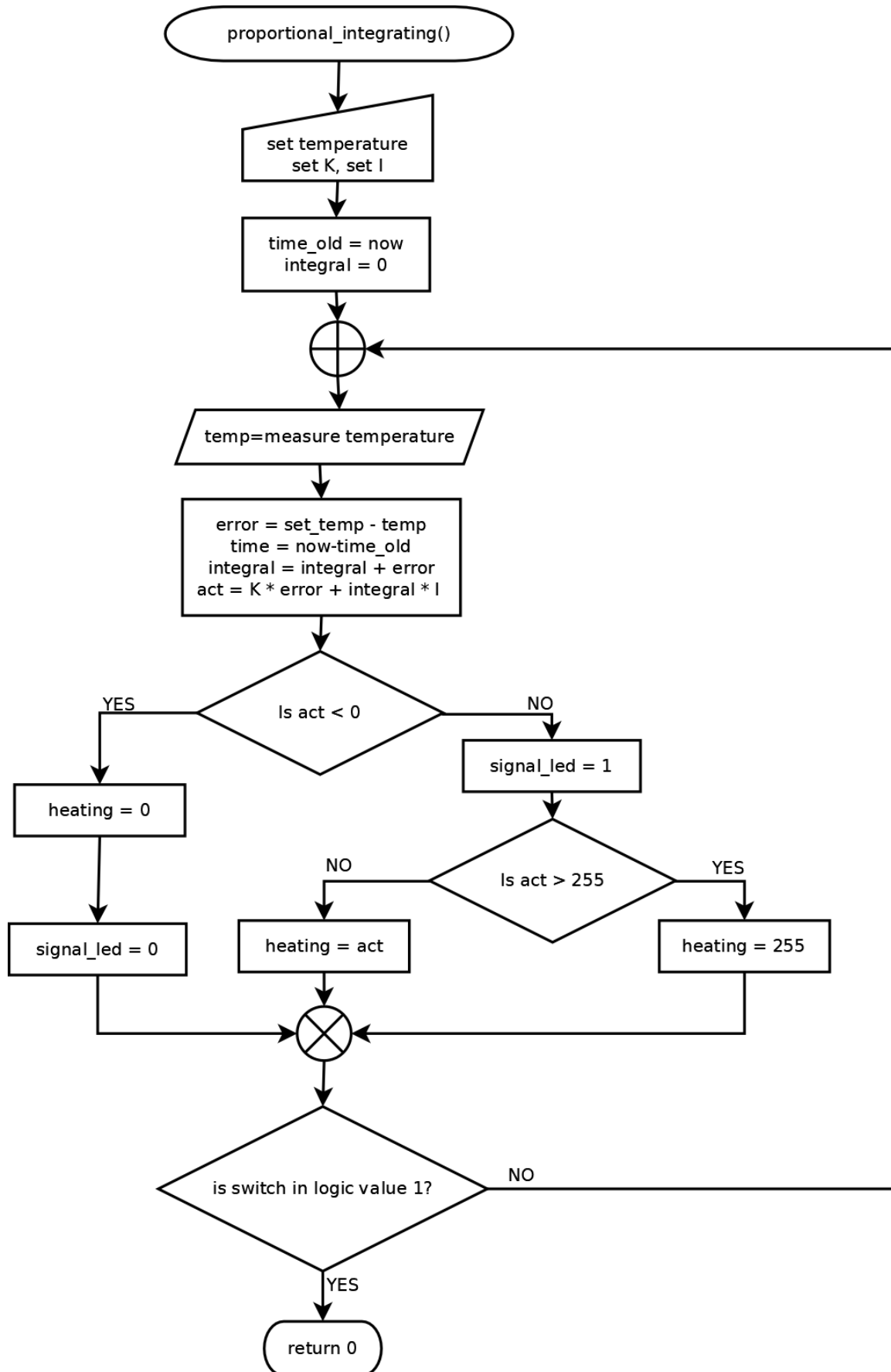
Picture 22 - Logic of proportional regulation

This function uses many functions. Therefore, there is only the major of function.

```
while(digitalRead(conf)==1){
    temperature=measure();           //measure temperature
    e=set-temperature;               //calculate control error
    act=konst*e;                     //calculate actuating variable
    action=(int)act+50;               //add ub (50 - from HW tests)
    if(action>255) action=255;       //limitations
    if(action<0) action=0;
    if(temperature_o!=temperature){
        /*rewrite temperature at TFT display*/
        temperature_o=temperature;
        analogWrite(heat, action);   //send PWM to heater
        if(action>0) digitalWrite(led_h, 1); //set led diode
        else digitalWrite(led_h, 0);
        /*save data to temporary files*/
    }
    web("P REGULATION",konst,0,set,1); //open web server
    for(i=0; i<20; i++){             //active waiting
        if(digitalRead(conf)==0){
            digitalWrite(led_h, 0);   //switch off led
            analogWrite(heat, 0);      //switch off heater
            break;
        }
        delay(250);
    }
}
//create measuring file
createoutputfile(newfilename(), "P regulation", set, konst, 0, 24, 1);
```

Source code 5 - Proportional regulation (shortened)

4.2.4 Proportional integrating regulation



Picture 23 - Logic of proportional integrating regulation

This function uses many functions. Therefore, there is only the major of function.

```

T_o=millis()/1000; //time of start measuring
while(digitalRead(conf)==1){
    temperature=measure(); //measure temperature
    e=set-temperature; //calculate control error
    Tm=millis()/1000; //calculate current time
    T=Tm-T_o; //calculate time between 2 measures
    integral=integral+e; //calculate integrating time constant
    act=konst*e+integral*konst1; //calculate actuating variable
    action=(int)act; //convert to int
    if(action>255) action=255; //limitations
    if(action<0) action=0;
    if(temperature_o!=temperature){
        /*rewrite temperature at TFT display*/
        temperature_o=temperature;
        if(action>0) digitalWrite(led_h, 1); //set led diode
        else digitalWrite(led_h, 0);
        analogWrite(heat, action); //send PWM to heater
        T_o=T; //save time of last measure
        /*save data to temporary files*/
    }
    web("PI REGULATION",konst,konst1,set,2); //open web server
    for(i=0; i<20; i++){ //active waiting
        if(digitalRead(conf)==0){
            digitalWrite(led_h, 0); //switch off led
            analogWrite(heat, 0); //switch off heater
            break;
        }
        delay(250);
    }
}
//create measuring file
createoutputfile(newfilename(), "PI regulation", set, konst, konst1, 8, 2);

```

Picture 24 - Proportional integrating regulation (shortened)

4.3 Interface

Because the final model should be a model for demonstrating types of regulations, we made a web interface for control measure and interface on Arduino for setting measure, or network. For more comfort at measuring at the front of the box are 4 LEDs.

Table 4.1 - Pins intended for signalling connection to the front panel

Function	Port	Colour
Power ON	29	Green
Fan ON	31	Yellow
Heater ON	33	RED
Additional pin	35	Without LED
External power supply ON	Hardware solution	Green

4.3.1 Interface on Arduino

Arduino interface is a simple menu, where the user could choose via rotary encoder between regulation. Setting IP network and transferring measuring files. Everything is showed on TFT display.

To make a user-friendly interface, we decided to add logos of our school. These logos are shown when we are starting Arduino.



Picture 25 - Logos of our schools

To show these logos, we need to convert it to bitmap, which display could redraw.

4.3.2 Setting network

Because our project should be as a demonstrating model, we should have an interface to show time dependencies on site. For this, we need to set IP addresses. Our project isn't built for a network with dynamic host configuration protocol. This is the reason, why you must set: IP address, subnet mask, gateway, and DNS server address. After setting this address, all data are stored to configure file (config.txt) at the SD card. At the start of the program, this file is read and configure the network card. That means that the restart of the Arduino is required.

193.168.1.20. 255.255.255.0. 192.168.1.255. 8.8.8.8.

External file 1 - The structure of config.txt

The addresses in the configuration file are written in order:

- IP address
- Subnet mask
- Gateway address
- DNS IP address

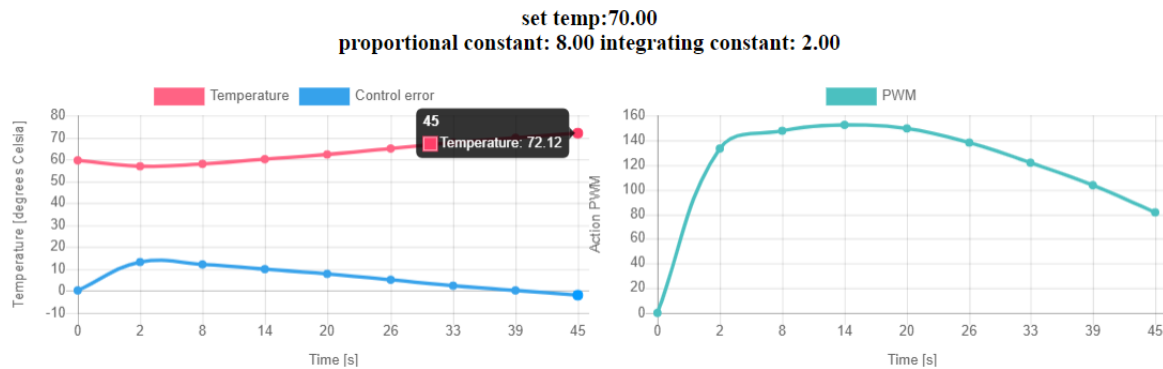
Every address ending with a dot, because, the file doesn't have to be created by Arduino, but could be created by the user. The user could use an operating system like Windows or Linux based system. There could be a problem with ending of a line, and therefore there is a dot.

4.3.3 Web interface

When users measure, at the same time they can control measure data via a web server. This control should be intuitive and without any difficulties. Via this interface user could only show time dependencies of actuating variable, actual temperature and value of PWM.

For plotting graphs on site, we use an Open source tool called chart.js. This tool is created to plot the data on site, to interactive graphs. More about this toll at website www.chartjs.org

PI REGULATION



Picture 26 - Web interface while measuring

This solution has only one problem – chart.js needs quite a big library. If we send this library from Arduino to the browser, it will take much time. That means that to see measuring graphs the controller must be connected to the internet.

4.3.4 Output file

After the end of measuring, program make a measuring file automatically, which contains time from start measuring, actuating variable, control error, and temperature.

```
PI regulation
Set temperature: 70.00 Proportional constant: 8.00 Integral constant: 2.00
| time | actuating variable | control error | measured temperature |
| 0.00 | 0.00 | 0.00 | 23.50 |
| 2.00 | 255.00 | 46.44 | 23.56 |
| 8.00 | 255.00 | 44.81 | 25.19 |
```

External file 2 - An example of measuring file

4.3.5 Transferring measuring files

Transferring measuring file is a perfect tool to make the output of measure. Via simply HTML page, you could copy measured data. Because these data are formatted as a chart, copy to excel is very simple and creating graph also.

MEASURING FILE 0041.TXT

PI regulation

Set temperature: 70,00 Proportional constant: 8,00 Integral constant: 2,00

time	measured temperature	control error	actuating variable
0,00	0,00	0,00	23,37
2,00	255,00	46,63	23,37
8,00	255,00	45,00	25,00
14,00	255,00	40,38	29,62

[GO BACK](#)

Picture 27 - Downloading measuring file

When we need download this file we could use this address:

<http://XXX.XXX.XXX.XXX/?YYYY.TXT>

where X - numbers in IP address, and YYYY is a number of measuring file (shown at the end of measure).

4.4 Hardware design

The hardware part of the project is the central part. First is needed to choose sensors and actuators. As we write the best version of the actuator is a combination of 7 power resistors. As a temperature sensor, we choose very reliable sensor DS18B20. For clarity, we made some printed circuit boards (PCB) via the photoresist method.

This PCB s we partitioned to 4 parts:

- PCB as output panel from the microcontroller
- PCB for LEDs and with pins for the encoder
- PCB for heater's and fan's electronic
- PCB for heater

For connect these PCB s between themselves, we use standard connectors for 16 wire flat cable.

4.4.1 Microcontroller

As a microcontroller, we use only Arduino Mega. Because this is a primary microcontroller for developing and learning, there is no place for soldering; there is only a pin header. We decided that wires at the pin header are not reliable and friendly, and we made a PCB with two connectors for other peripherals (PCB s). On top of these, there are two pins connected to the external power supply to indicate the state ON/OFF.

4.4.1.1 Arduino pinout

This table shows, Arduino pin header with pins 22-53. On pins 0-22 and all analogue pins, we have mounted shield. Therefore, we don't make a PCB for them.

Table 4.2 - Arduino Mega pinout (pins 22- 53)

	5V	5V	
temperature sensor	22	23	encoder – button
	24	25	encoder – sigA
	26	27	encoder – sigB
	28	29	LED – power
	30	31	LED - fan
	32	33	LED – heater
	34	35	LED – additional pin
	36	37	spare pin 37
	38	39	spare pin 39
	40	41	spare pin 41
	42	43	spare pin 43
fan - PWM	44	45	
heater - PWM	46	47	
	48	49	spare pin 49
	50	51	spare pin 51
	52	53	spare pin 53
	GND	GND	

4.4.1.2 Front panel connector

The connector from the microcontroller to front panel PCB. We are using a 16-core cable. On another side of the connection is the same connection of connector.

Table 4.3 – Front panel connector

5V	1	2	GND
encoder – button	3	4	GND
encoder – sigA	5	6	GND
encoder – sigB	7	8	GND
LED – power	9	10	spare pin 43
LED - fan	11	12	spare pin 41
LED – heater	13	14	spare pin 39
LED – additional pin	15	16	spare pin 37

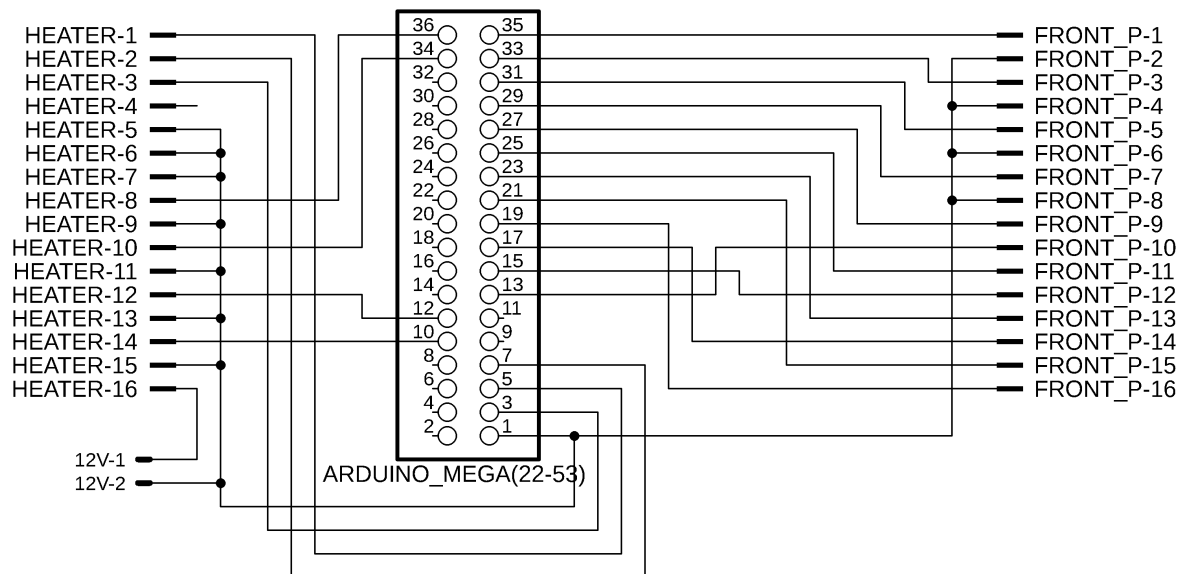
4.4.1.3 A heating control panel connector

The connector from the microcontroller to front panel PCB. We are using a 16-core cable. On another side of the connection is the same connection of connector.

Table 4.4 – Heating control panel connector

spare pin 51	1	2	spare pin 49
spare pin 53	3	4	
GND	5	6	GND
GND	7	8	5V
GND	9	10	temperature sensor
GND	11	12	fan
GND	13	14	heater
GND	15	16	12V - power supply

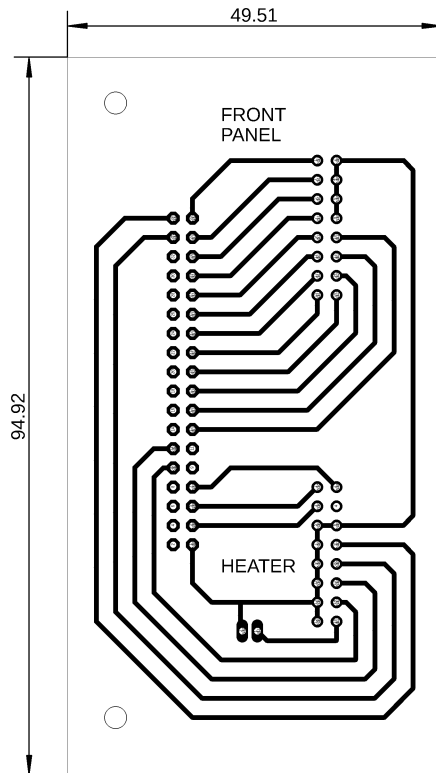
4.4.1.4 PCB



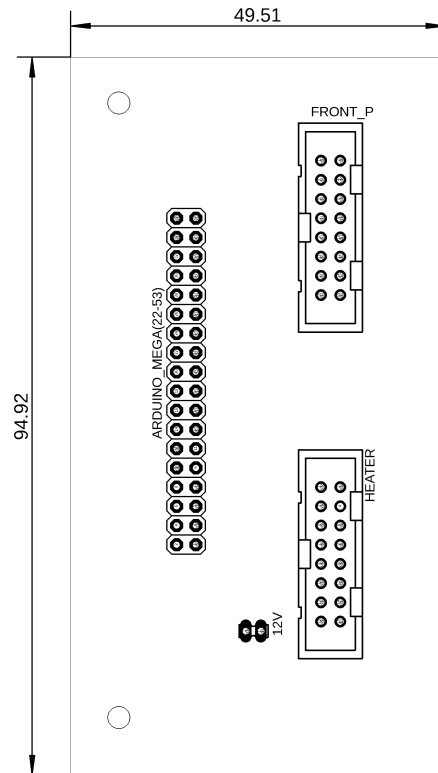
Picture 28 - Schematic of microcontroller PCB

Table 4.5 - List of parts to microcontroller PCB

Part	Value	Device	Note
FRONT_P	-	Connector 8x2	
HEATER	-	Connector 8x2	
ARDUINO_MEGA(22-53)	-	Pinhead 18x2	Output from Arduino

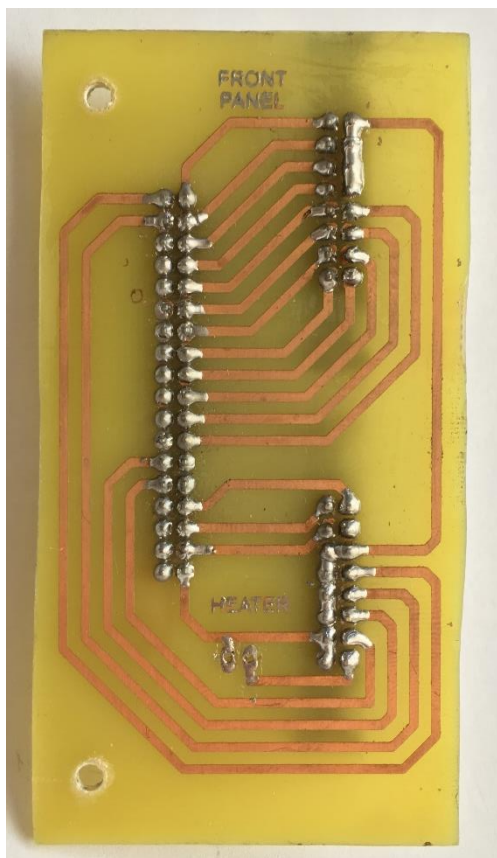


Picture 30 - Microcontroller PCB
(scale 1:1)

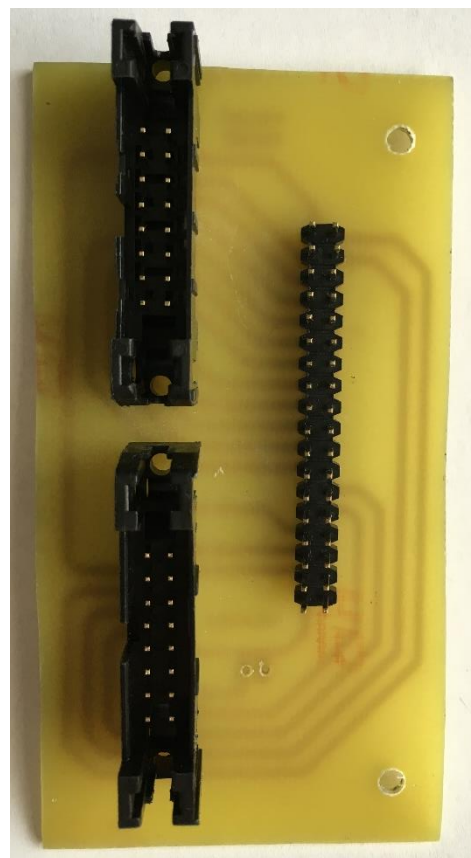


Picture 29 - Microcontroller PCB
(component placement)

4.4.1.5 Realisation



Picture 31 - Microcontroller PCB (top)



Picture 32 - Microcontroller PCB
(bottom)

4.4.2 Heating and cooling circuit

Because we use transistors for switching cooling and heating, we must connect the external power supply and thermometer. For realising this we could use universal PCB, but when we solder connector there are so many paths to connect, and it will be messy. Thought heater is a simple circuit; we also make a simple PCB, with coarser paths for higher current.

4.4.2.1 Power supply

From demonstrating model, we expect low weight for mobility. Using an external power supply will be very complicated. We need to have some laboratory supply with output voltage 12V and minimally with 2A. Many laboratories in our schools are equipped with standard supply with heavy coils. To avoid this situation, we add a switching power supply. Moreover, the switching supply will ensure a constant output voltage. As a power supply, we have chosen to supply Mean Well LRS-35-12.

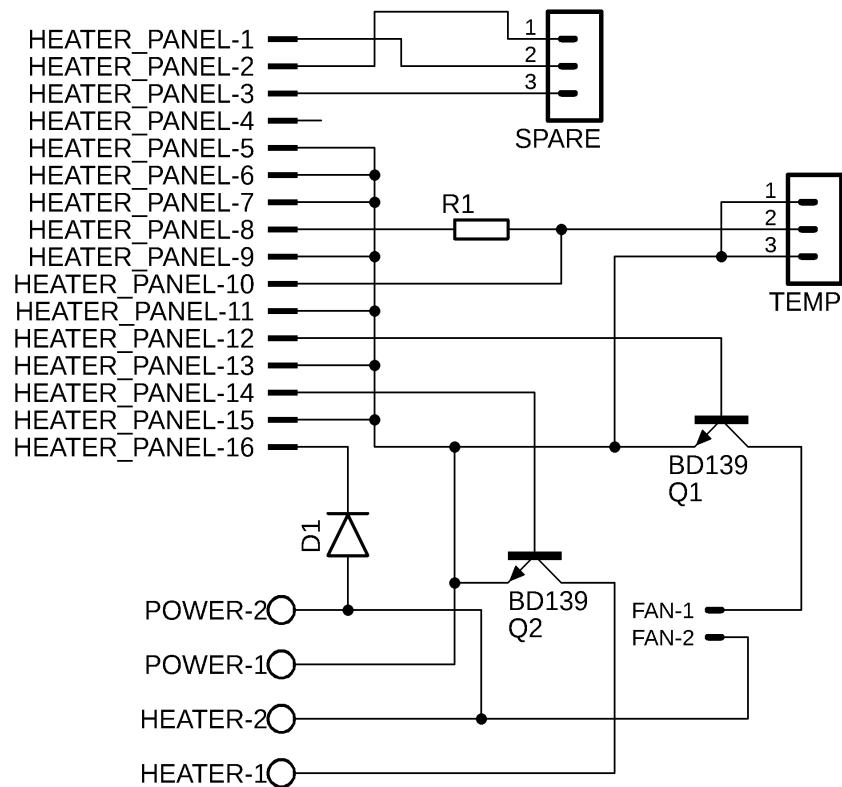
Table 4.6 - LRS-35-12

The minimal input AC voltage	85V
The maximal input AC voltage	264V
Output voltage	12V DC
Output current	3A
Output max power	36W

Source: official datasheet,

http://www.farnell.com/datasheets/2547981.pdf?_ga=2.189240107.1092266099.1551970860-455427088.1537292678

4.4.2.2 Heating control PCB

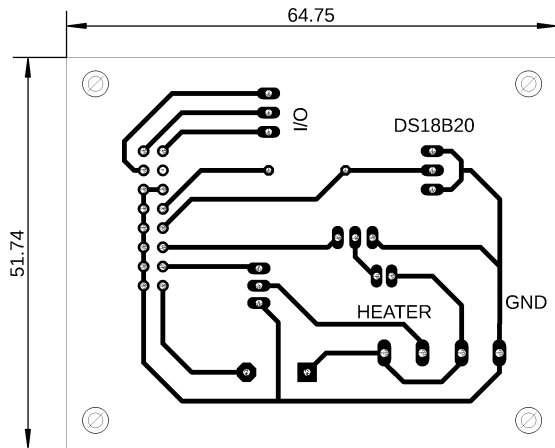


Picture 33 - Schematic of heating control PCB

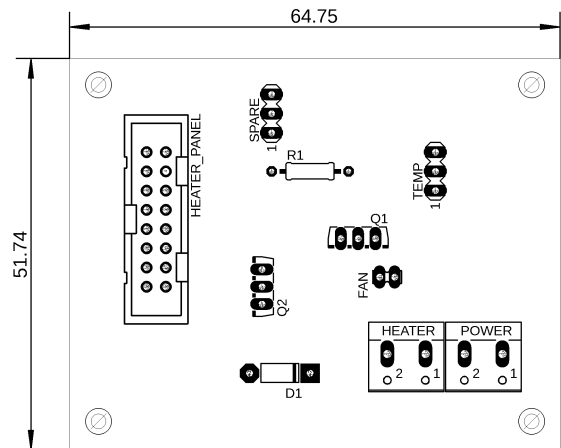
Table 4.7 - List of parts for heating control PCB

Part	Value	Device	Note
HEATER_PANEL	-	Connector 8x2	
POWER	-	Screw Clamp	For power supply
HEATER	-	Screw Clamp	For heater
FAN	-	Pinhead 2x1	For fan
SPARE	-	Pinhead 3x1	For spare pins
TEMP	-	Pinhead 3x1	For DS18B20
Q1	-	BD139	For fan
Q2	-	BD139	For heater
D1	-	-	Additional protection
R1	4k7	Resistor	For DS18B20

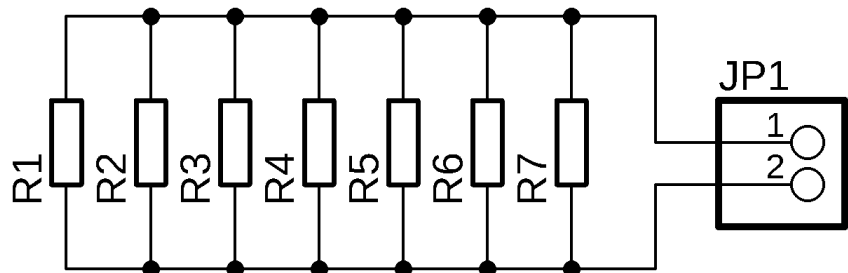
4.4.2.3 Heater PCB



Picture 35 - Heating control PCB (scale 1:1)



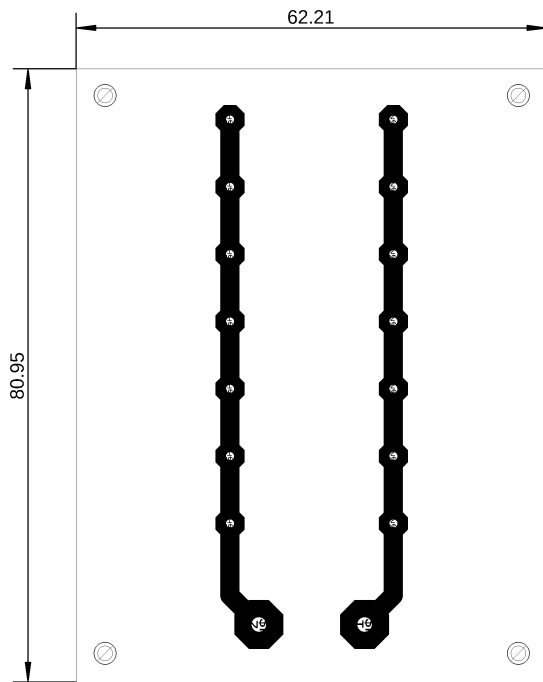
Picture 36 – Heater control PCB (component placement)



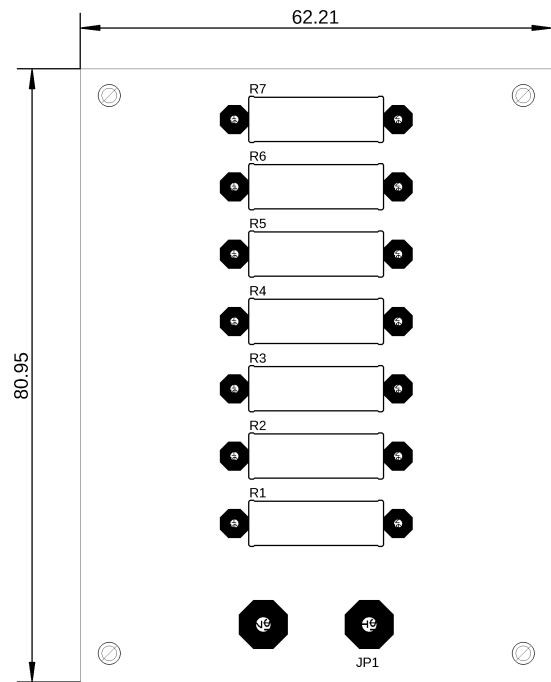
Picture 34 - Schematic of heater PCB

Table 4.8 - List of parts for heater PCB

Part	Value	Device	Note
R1-R7	47R, 5W	resistor	Carbon package
JP1	-	-	Only soldering pad

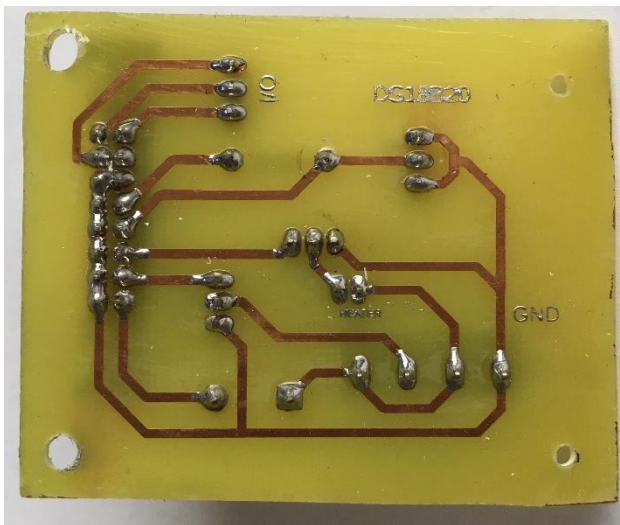


Picture 37 - Heater PCB (scale 1:1)

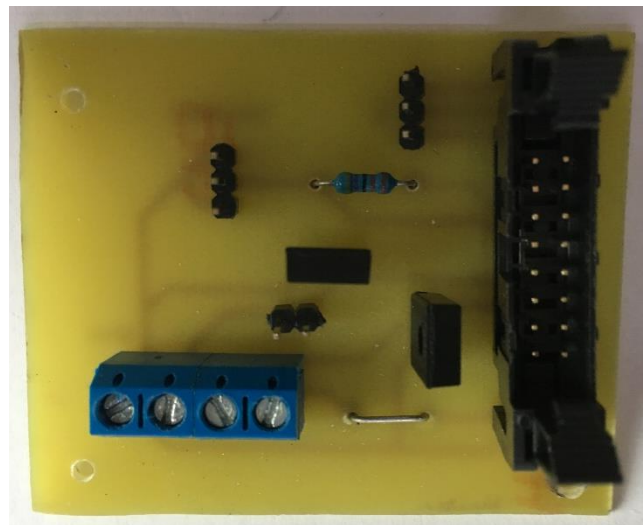


Picture 38 - Heater PCB (component placement)

4.4.2.4 Realisation



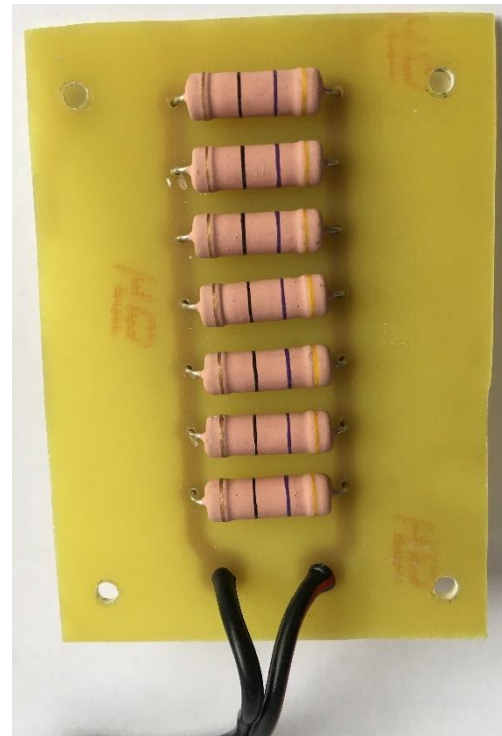
Picture 39 - Heating control PCB (top)



Picture 40 - Heating control PCB (bottom)



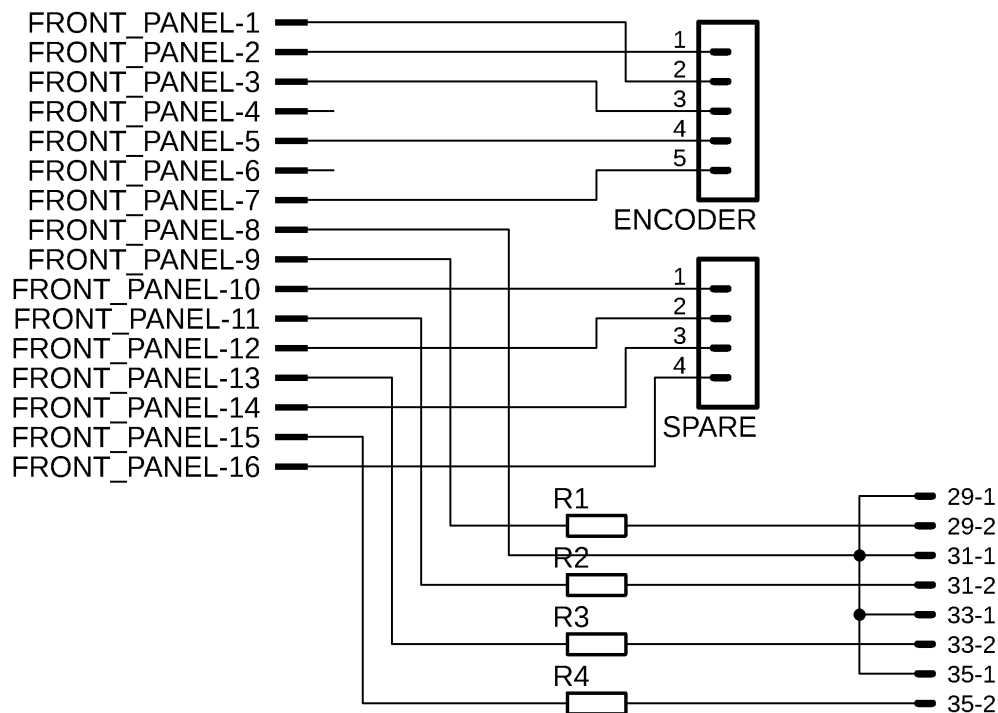
Picture 41 - Heater PCB (top)



Picture 42 - Heater PCB (bottom)

4.4.3 Front panel

The front panel functionally has only one function – displacement LEDs and encoder from microcontroller pin header. This PCB consists of 4 resistors, 4 pins for diodes and 3 additional pins for possibly new peripherals.

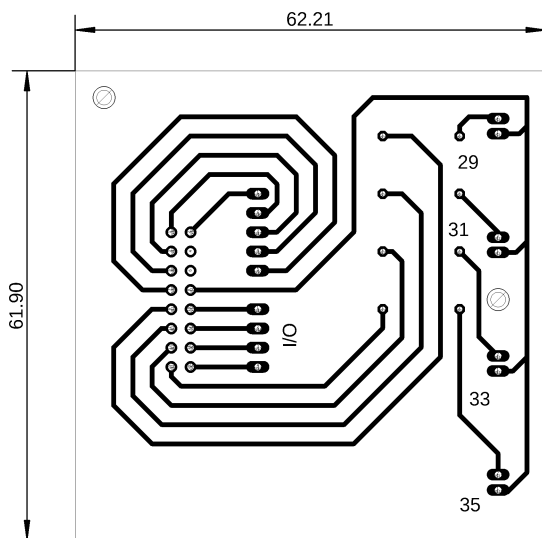


Picture 43 - Schematic of front panel PCB

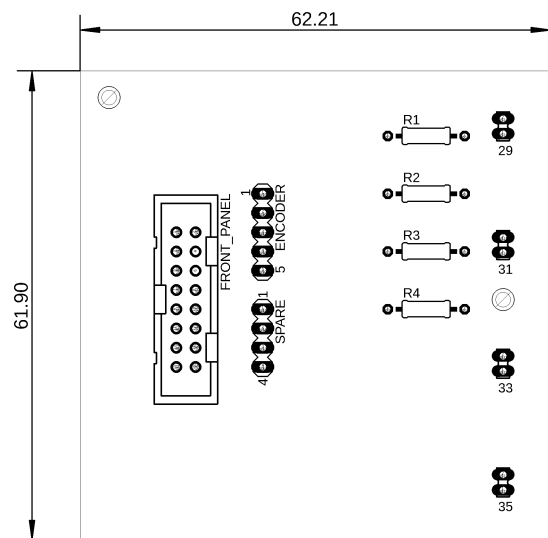
Table 4.9 - List of parts for front panel PCB

Part	Value	Device	Note
FRONT_PANEL	-	Connector 8x2	
ENCODER	-	Pinhead 5x1	For rotary encoder
SPARE	-	Pinhead 3x1	For spare pins
R1-R4	330R	resistor	
29	-	Pinhead 2x1	For LED diode
31	-	Pinhead 2x1	For LED diode
33	-	Pinhead 2x1	For LED diode
35	-	Pinhead 2x1	For LED diode

4.4.3.1 PCB

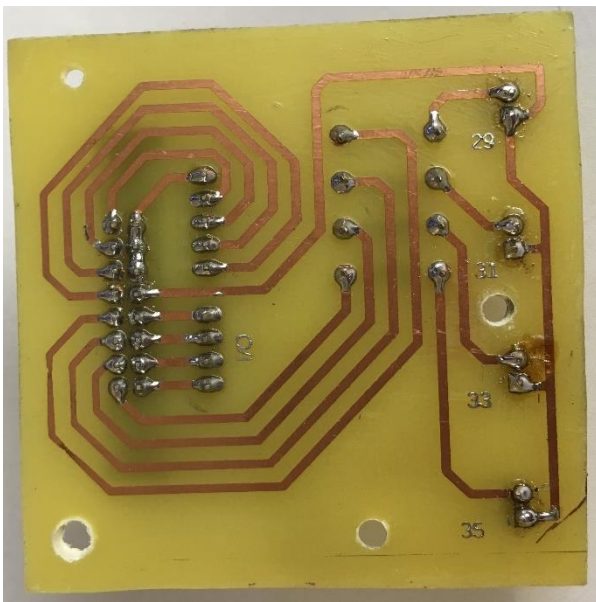


Picture 45 - Front panel PCB (scale 1:1)

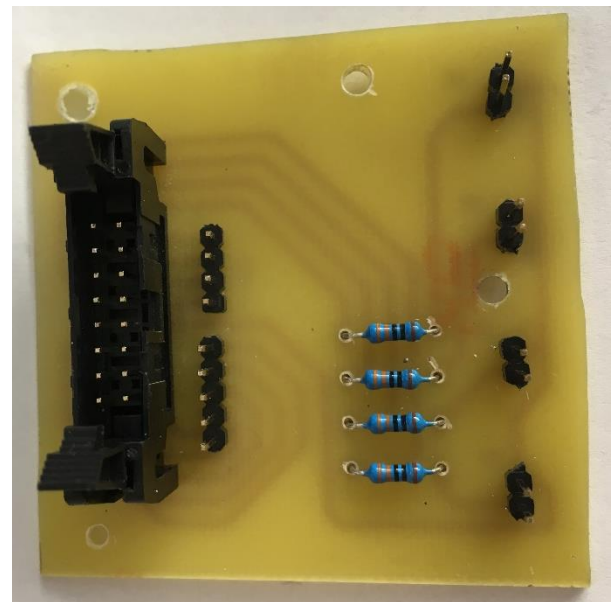


Picture 44 - Front panel PCB (component placement)

4.4.3.2 Realisation



Picture 47 - Front panel PCB (top)



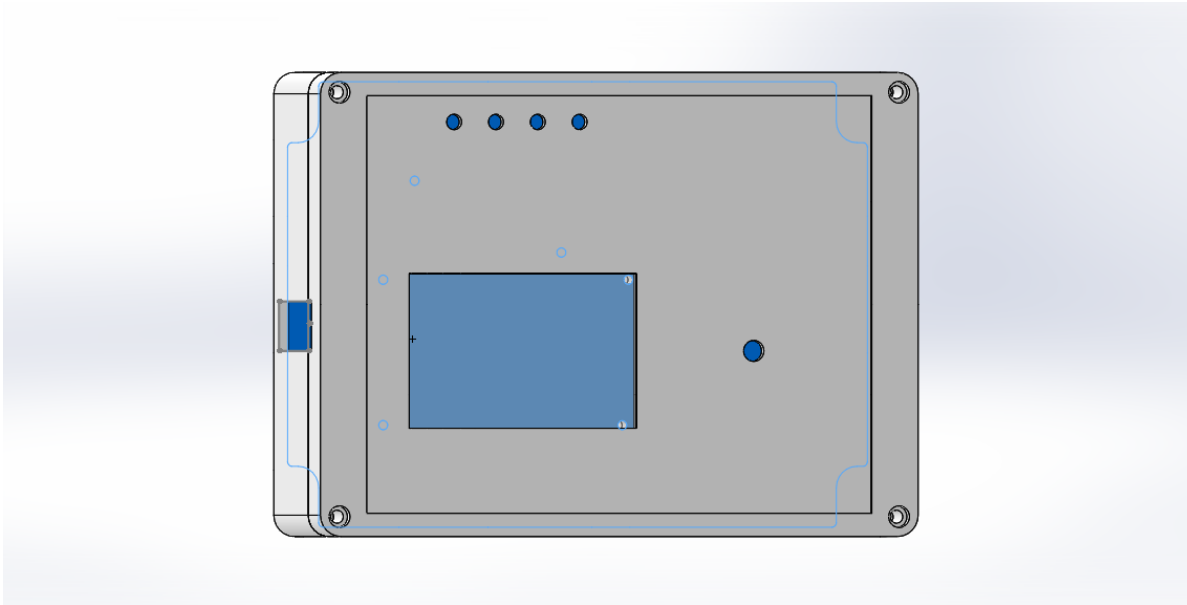
Picture 46 - Front panel PCB (bottom)

5 Chamber and case for controller

For an excellent presentation of the model and for extending a lifetime of microcontroller and used peripherals, we need to cover whole electronic into the cases.

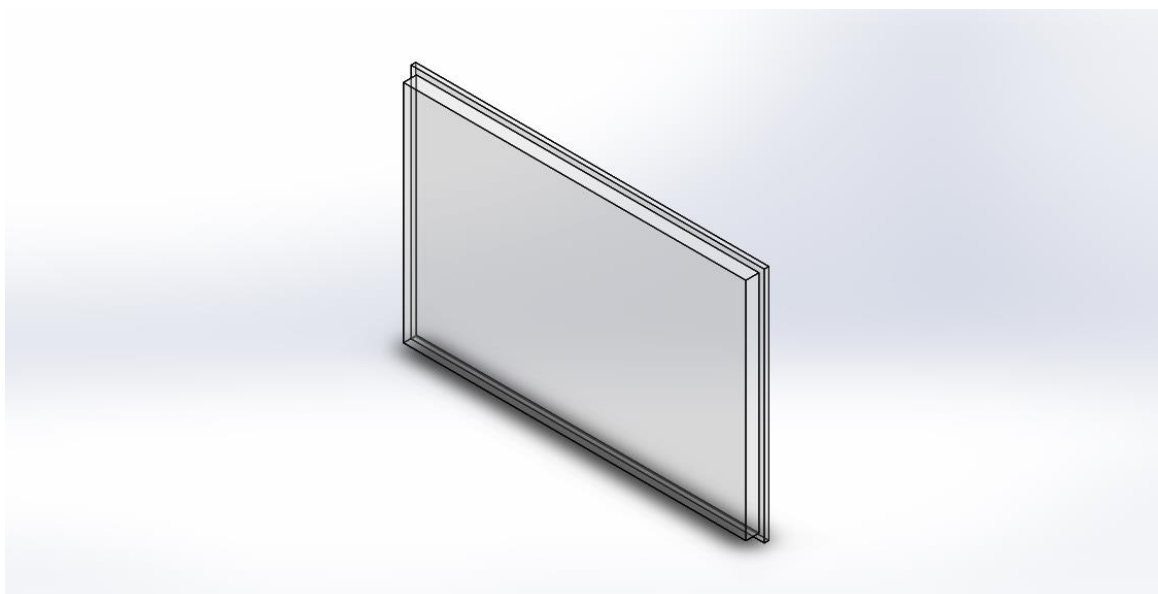
5.1 Case for microcontroller

This case is the hardest to do. We need to cut a hole for display. This is impossible without a Computer numeric control machine (CNC). For this, we need to write a G code, which this machine uses for precisely placing of the milling cutter.

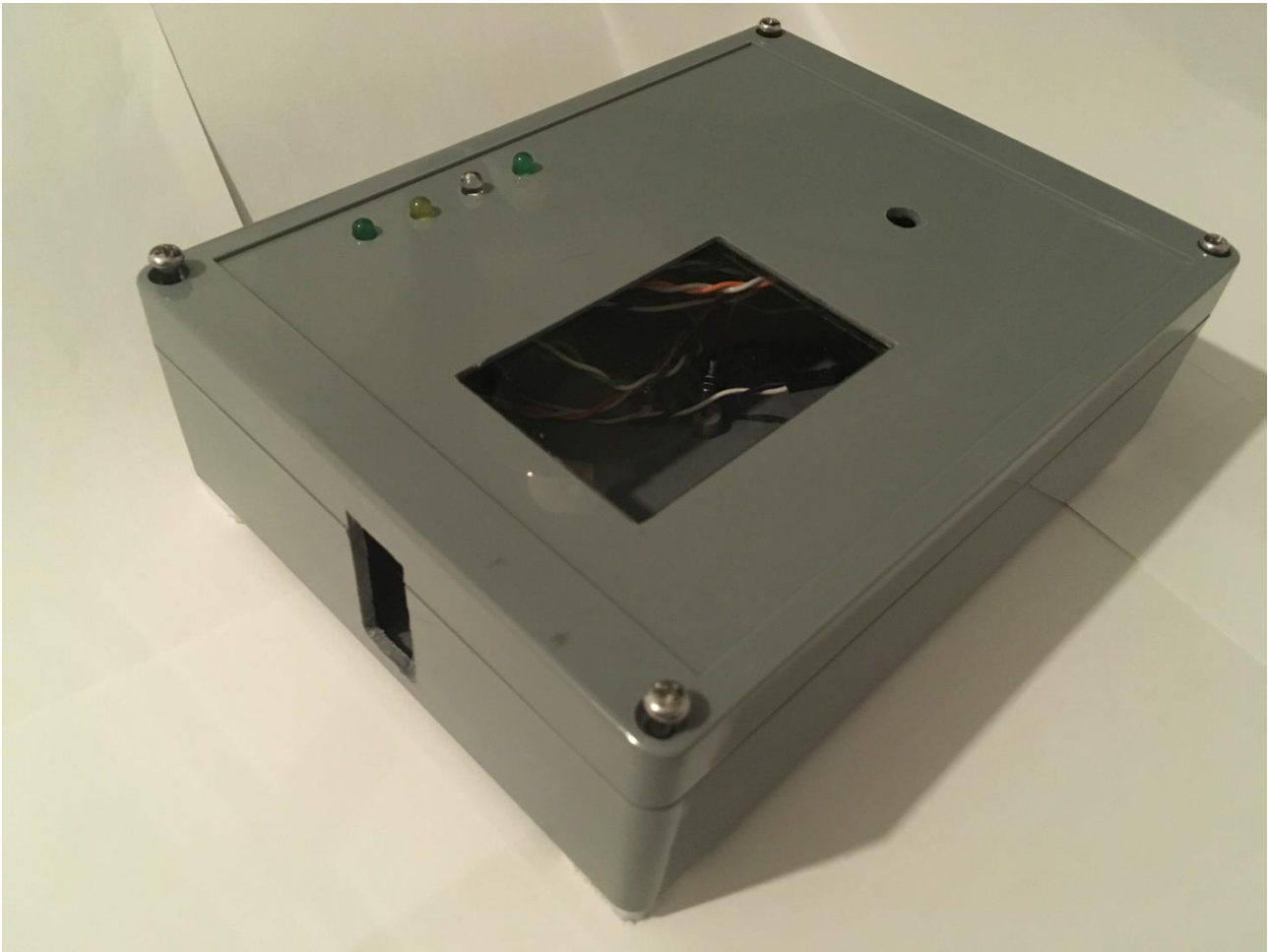


Picture 48 - SOLIDWORKS model of microcontroller case

We have the option to cut acrylic glass as a cover of the display. This glass is glued to case same as LEDs. When we place all components into the case, we found, that made separately PCB s for microcontroller and front panel was unnecessary.



Picture 49 - SOLIDWORKS model of acrylic protective glass

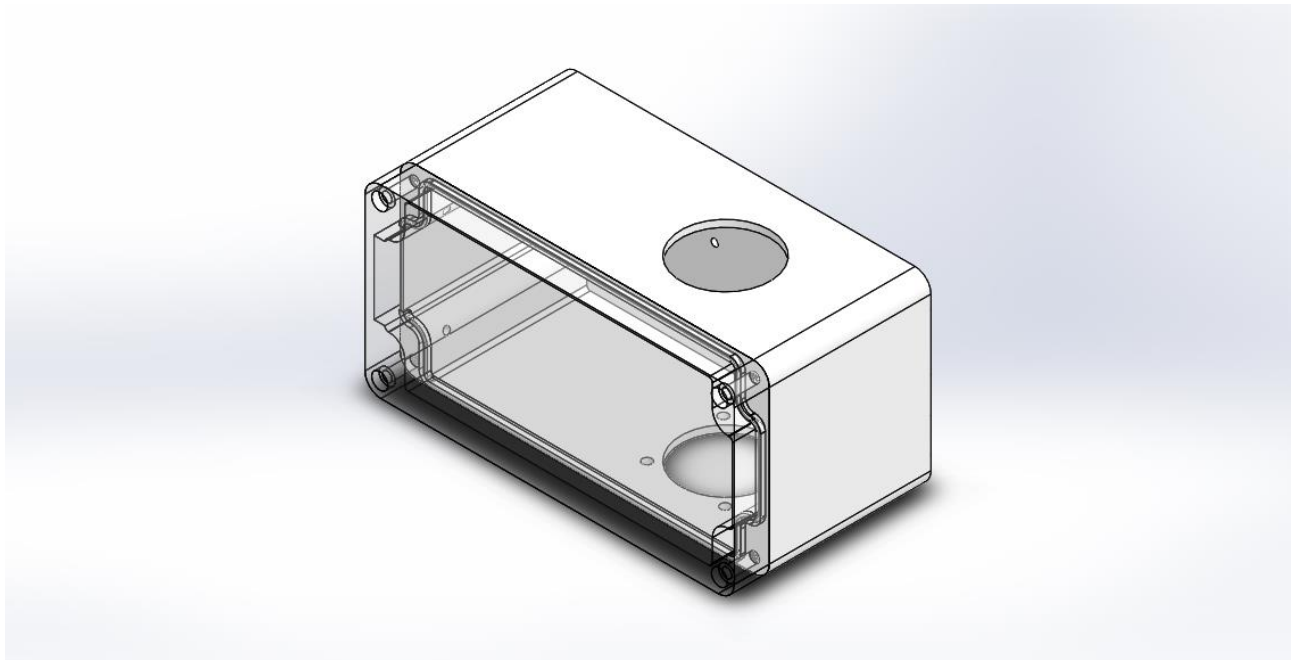


Picture 50 - Final realization of case for microcontroller (with glued LED diodes and transparent protective glass)

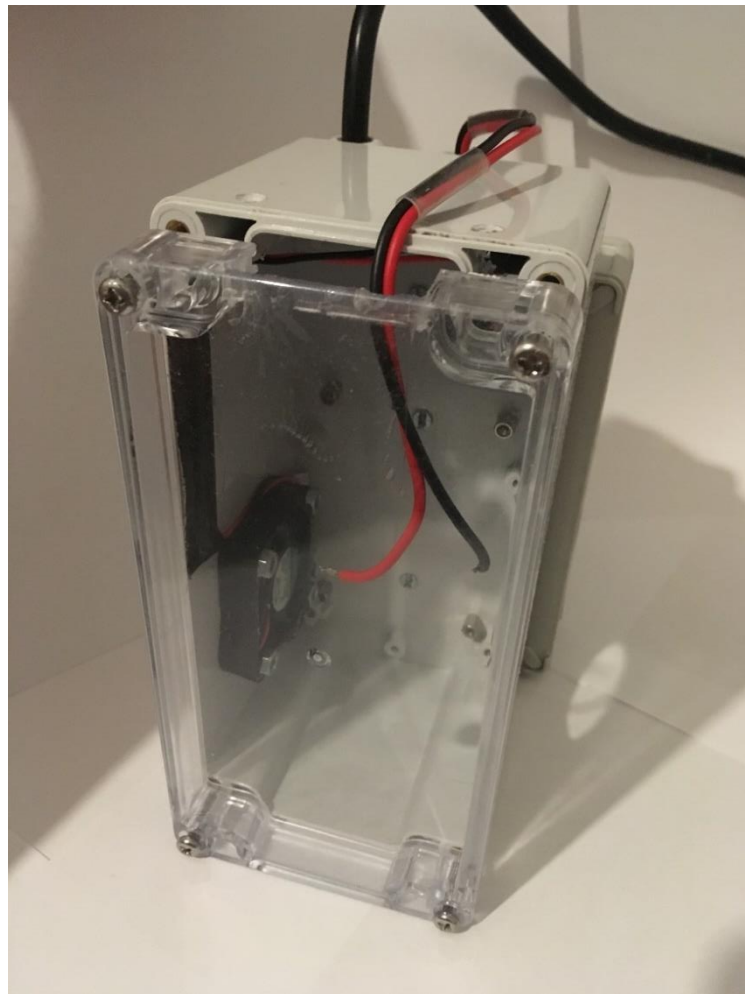
5.2 The case for the heating element

The design of this case was for us uppermost. The PCB s should be placed so as the heater element doesn't cause overheating of transistors, the wires from screw clamps to the heater weren't so long, and the hole for fan must be next to the heater. We decided that a hole for cooling should be on the longer side of the box, and the hole should be twice. One for a blow out hot air and second for entry colder air.

Next hole what we need is for flat cable and power supply. We decided that the hole will be at the side of the transparent part.



Picture 52 - SOLIDWORKS model of chamber



Picture 51 - Final realization of chamber (astern is mounted power supply)

5.3 The case for the power supply

This case has only one function to protect users from hazardous live-part of supply. The supply is screwed to the heating chamber. We use grommets prepared by the producer.



Picture 54 - Side view of chamber



Picture 53 - Back view of chamber

6 Control measures

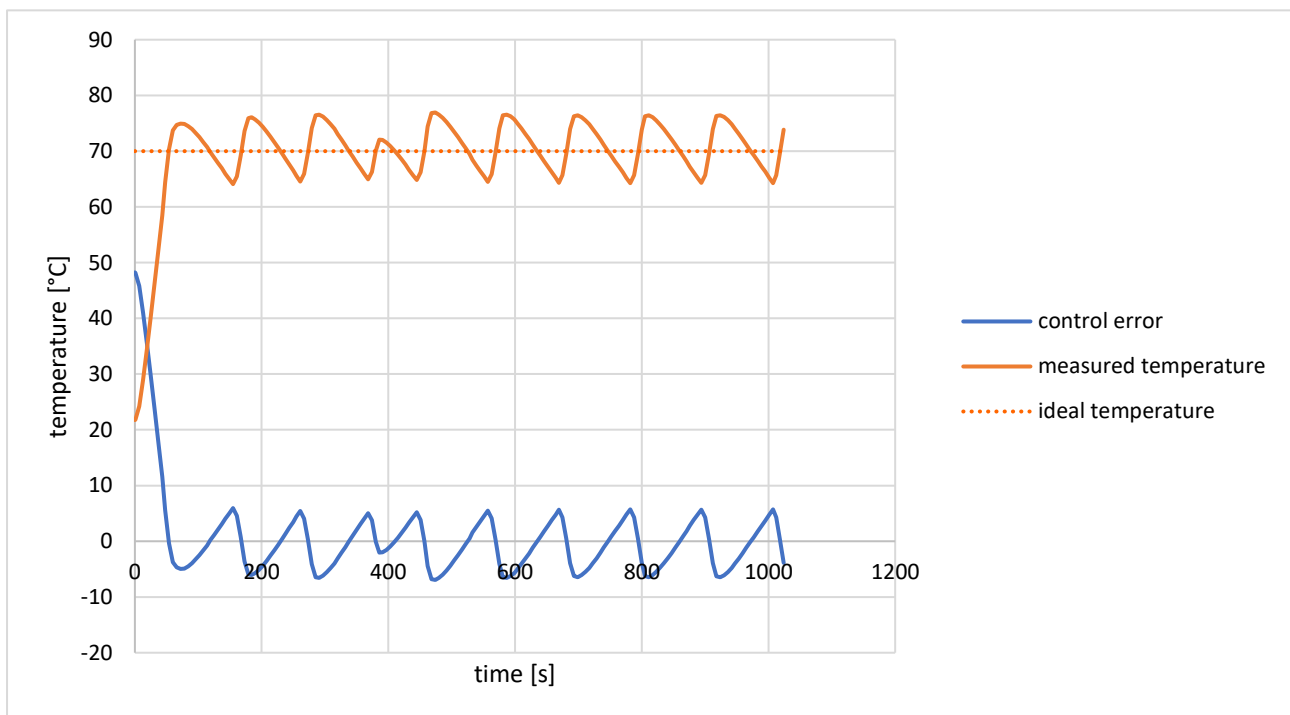
This control measures were done only for testing purposes. In this point, we put only graphs and result of measuring. These measures are used as hardware tests of heater and chamber. After these tests, we could make a final model of the regulator.

6.1 Two point

For control this algorithm we perform necessary measure with these parameters:

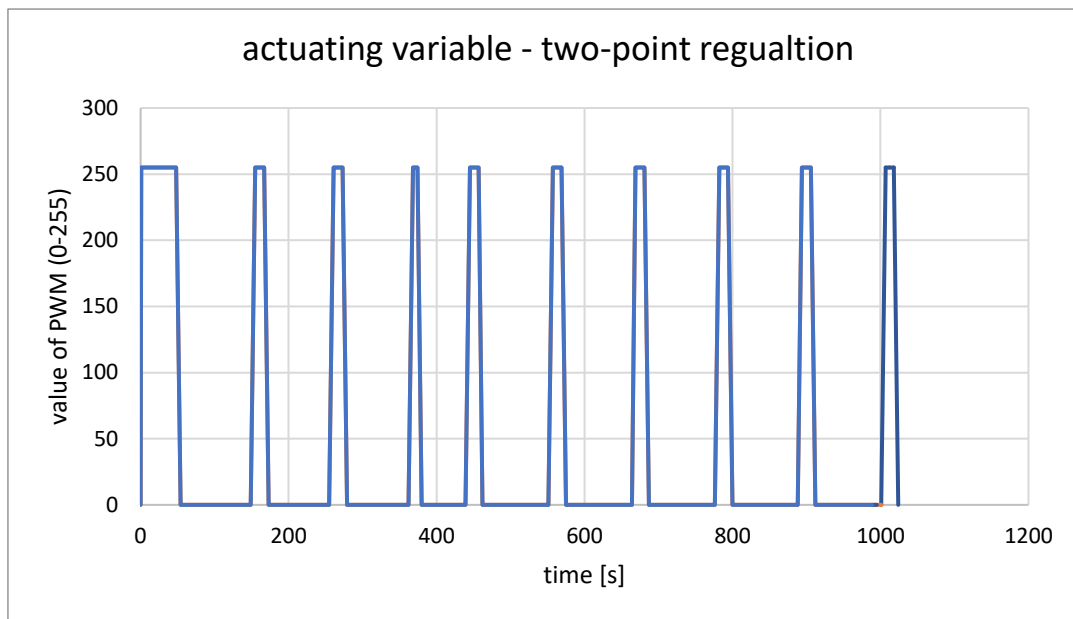
Table 6.1 - Parameters for performing HW tests of two-point regulation

Hysteresis	5°C
Set temperature	70°C



Picture 55 - HW test, two-point regulation - temperature, control error

Regulation of the temperature is complicated. The result temperature oscillate between 65°C and 74°C.



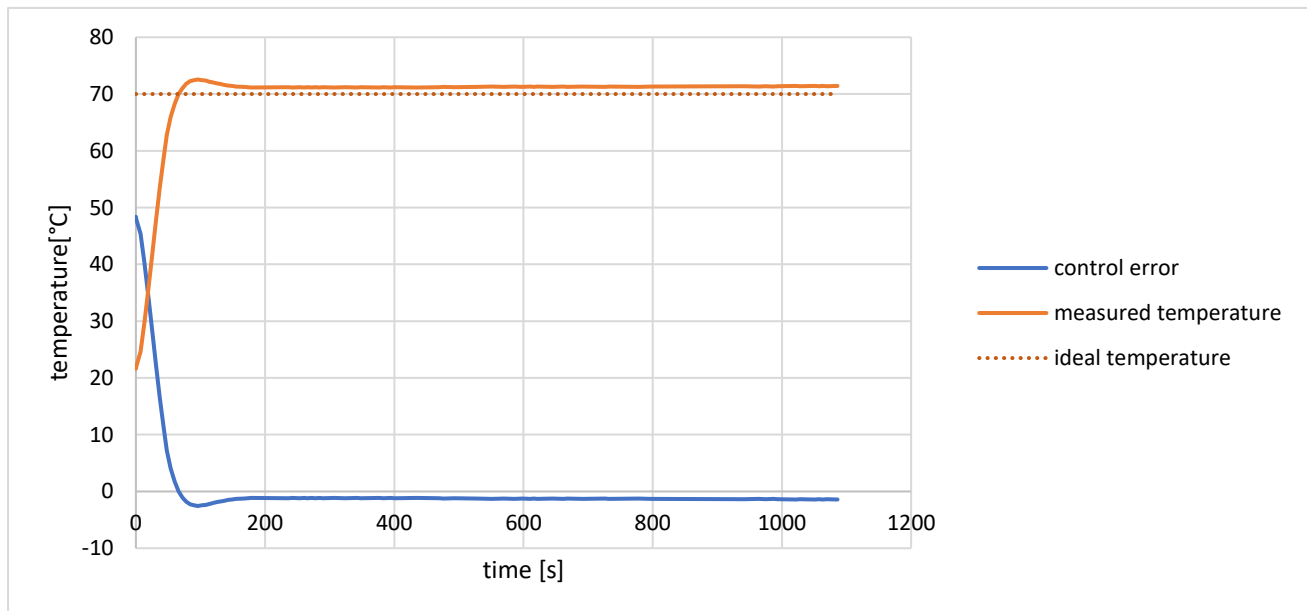
Picture 56 - HW test, two-point regulation, - actuating variable

6.2 Proportional regulator

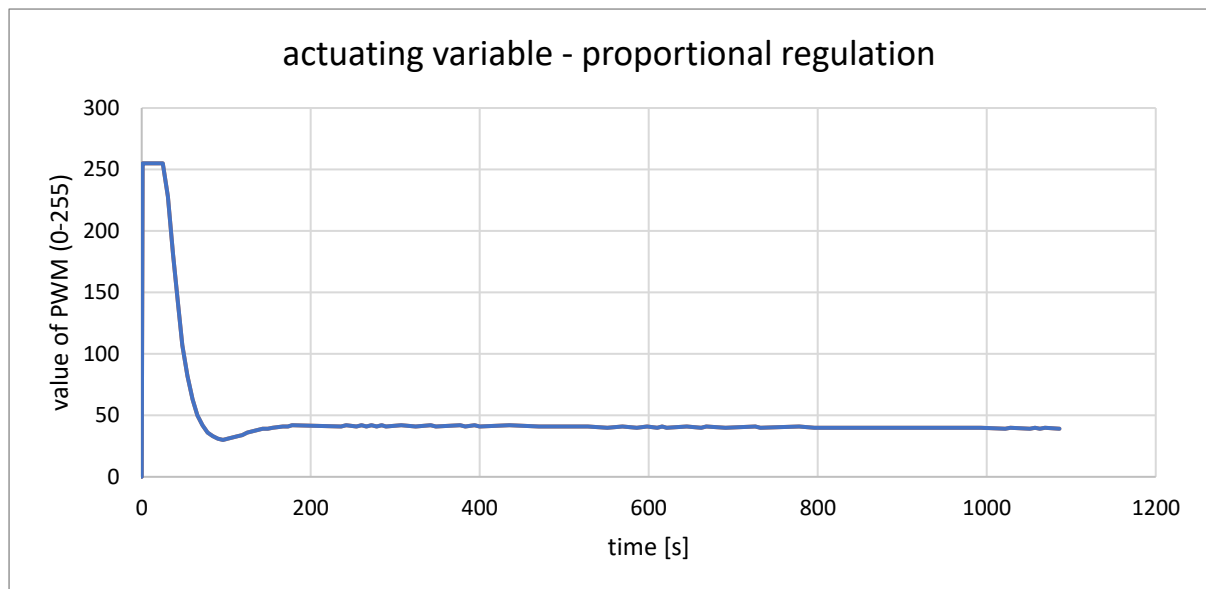
For control this algorithm we perform necessary measure with these parameters:

Table 6.2 - Parameters for performing HW tests of P regulation

Reinforcement of regulator	8
Set temperature	70°C



Picture 57 - HW test, P regulation - temperature, control error



Picture 58 - HW test, P regulation, - actuating variable

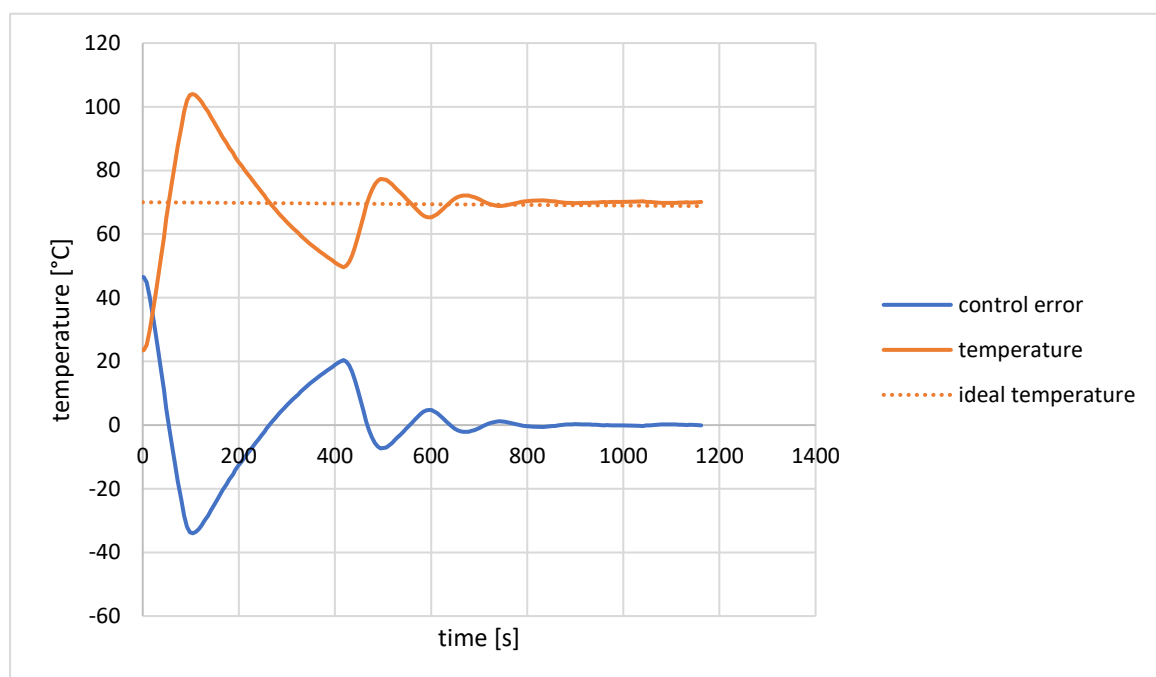
From this plot, we brightly see value u_b . It means, that if the regulator has required temperature, still heats a bit.

6.3 Proportional integrating regulator

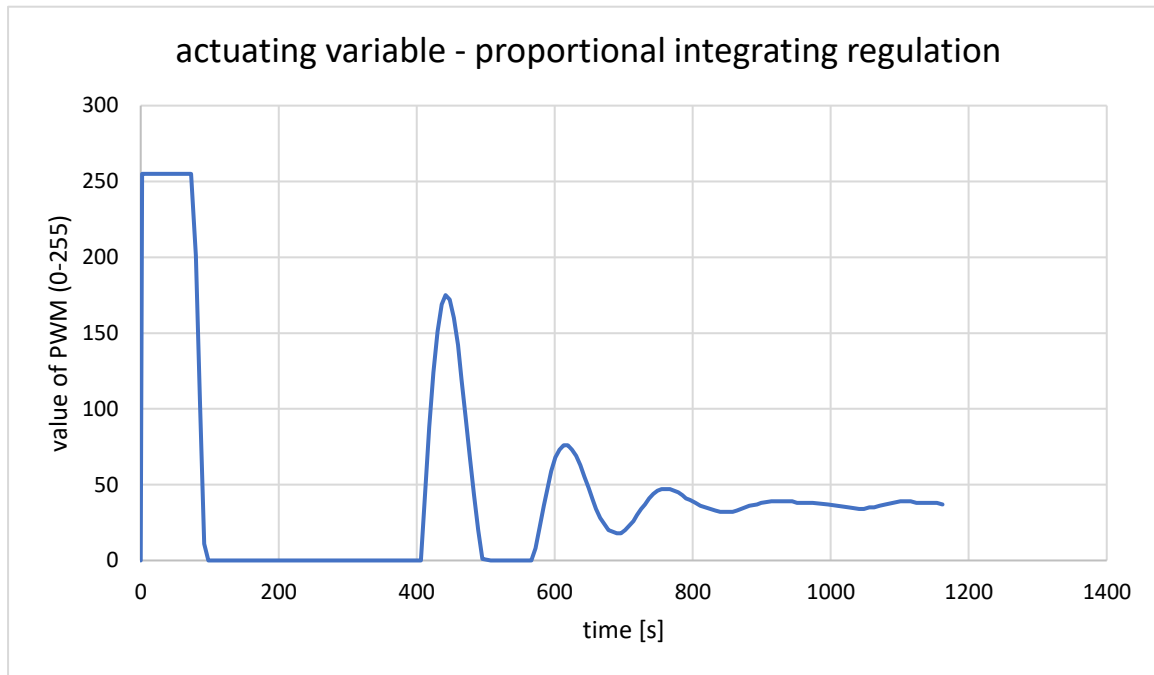
For control this algorithm we perform necessary measure with these parameters:

Table 6.3 - Parameters for performing HW tests of PI regulation

Reinforcement of regulator	8
Integrating time constant	2
Set temperature	70°C



Picture 59 - HW test, PI regulation - temperature, control error



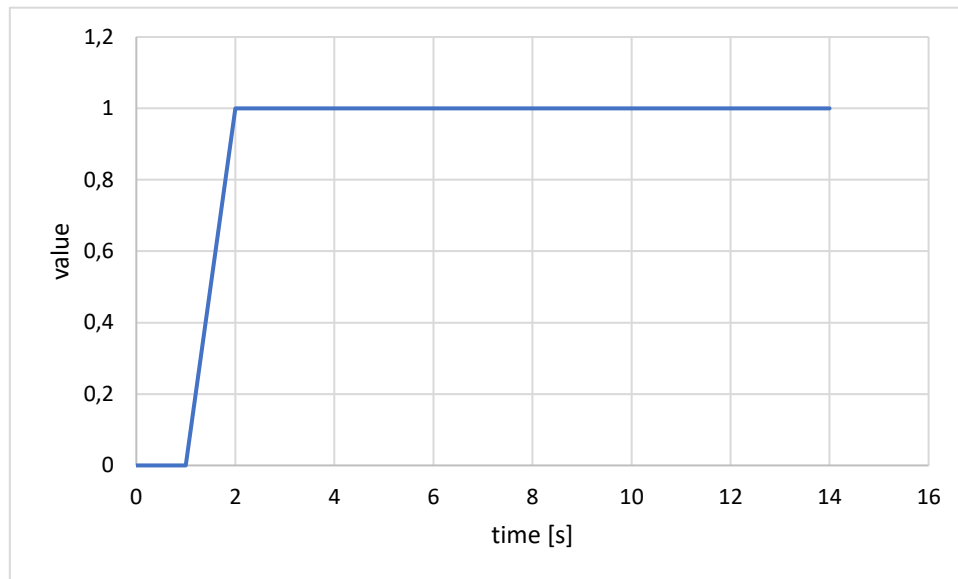
Picture 60 - - HW test, PI regulation, - actuating variable

From these graphs we can see, that regulation works, as we except, but at the end of regulation, the control error starts to oscillate. It means that control error is not zero (oscillate between -0,6 and 0,6) — this problem we could remove by tuning all constants.

As we write above, when the control error is too big, regulation looks like two-point regulations, but when is control error smaller, characteristic is typical for a PI regulator.

7 Tuning the regulator

For tuning the regulator, we need to measure the transient characteristic. For measure this characteristic we need to send unit jump as actuating variable.



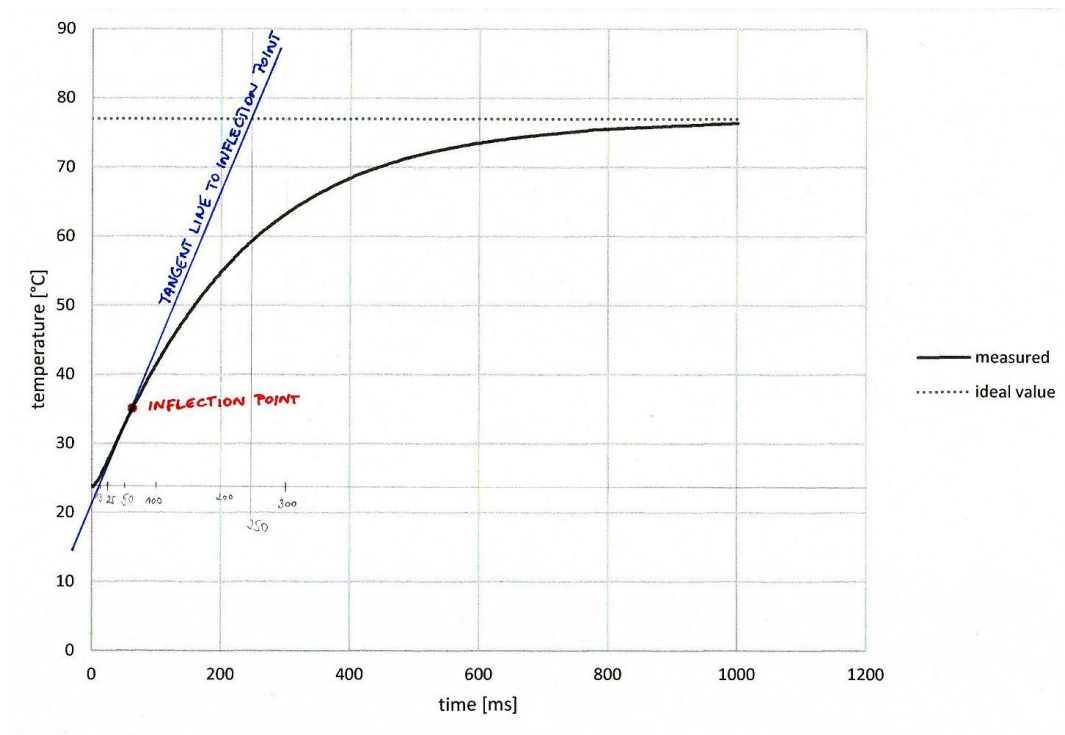
Picture 61 – Unit jump

In measured characteristic, we need to find an inflexion point (the point where curve is going from concave to convex shape) and make a tangent line to this point. In point, where tangent crosses the parallel line to x-axis going through the start of characteristic, we have point L. After that, we should make a sei OL. The tangent should also cross second parallel line going through the maximum value of the transient characteristic. Now we should make a perpendicular line to a maximum value which goes through crossing point. Where perpendicular line crosses the parallel line to the x-axis going through the start of the characteristic, we will have a point T. Second sei will be sei between points LT.

From lengths of this seis, we can calculate using the empirically acquired equations, constants to the regulator. We will use a method called Zingler-Nichols method.

Table 7.1 – Equations for constants using Zingler-Nicholas method

Regulation	K_P	K_I
P	$\frac{T}{L}$	0
PI	$0,9 * \frac{T}{L}$	$0,27 * \frac{T}{L^2}$



Picture 62 – Transient characteristic, inflex point, tangent to inflex point

From this plot we could read these numbers:

- Length of sei 0L: 13 = L
- Length of sei LT: 237 = T

Table 7.2 – Calculate constants Zingler-Nicholas method

Regulation	K_P	K_I
P	$\frac{T}{L} = \frac{237}{13} = 18,2$	0
PI	$0,9 * \frac{T}{L} = 0,9 * \frac{237}{13} = 16,4$	$0,27 * \frac{T}{L^2} = 0,27 * \frac{237}{13^2} = 0,4$

7.1 Check accuracy of constants

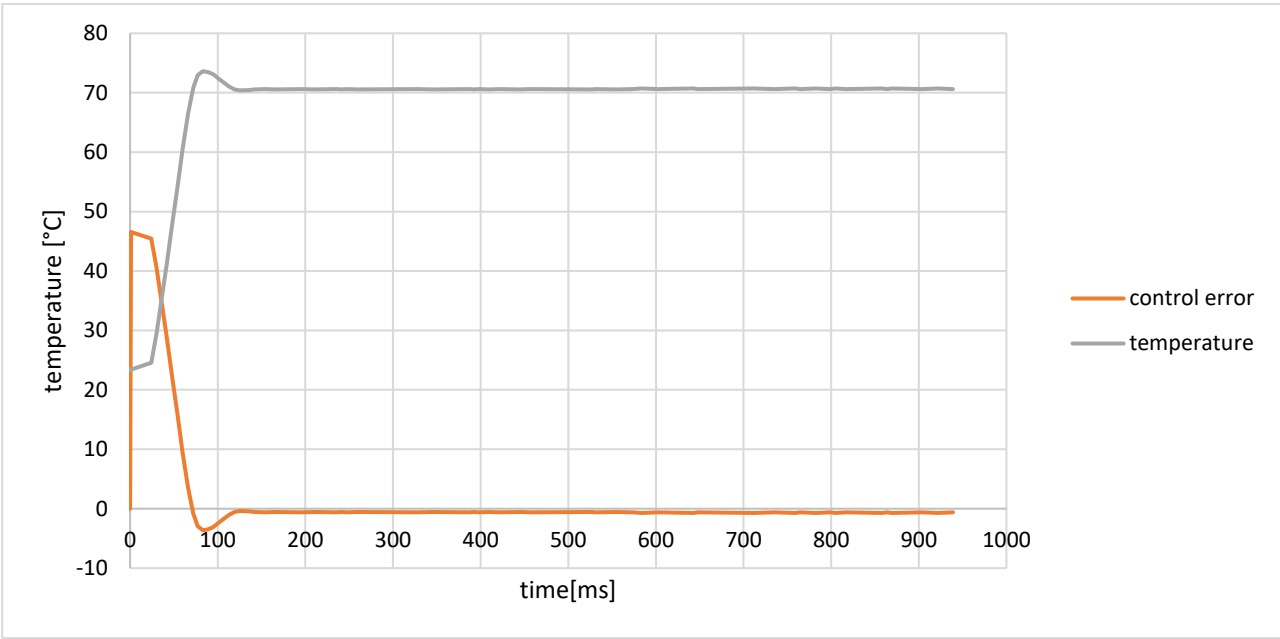
This check we perform with the next measure.

7.1.1 Proportional regulation

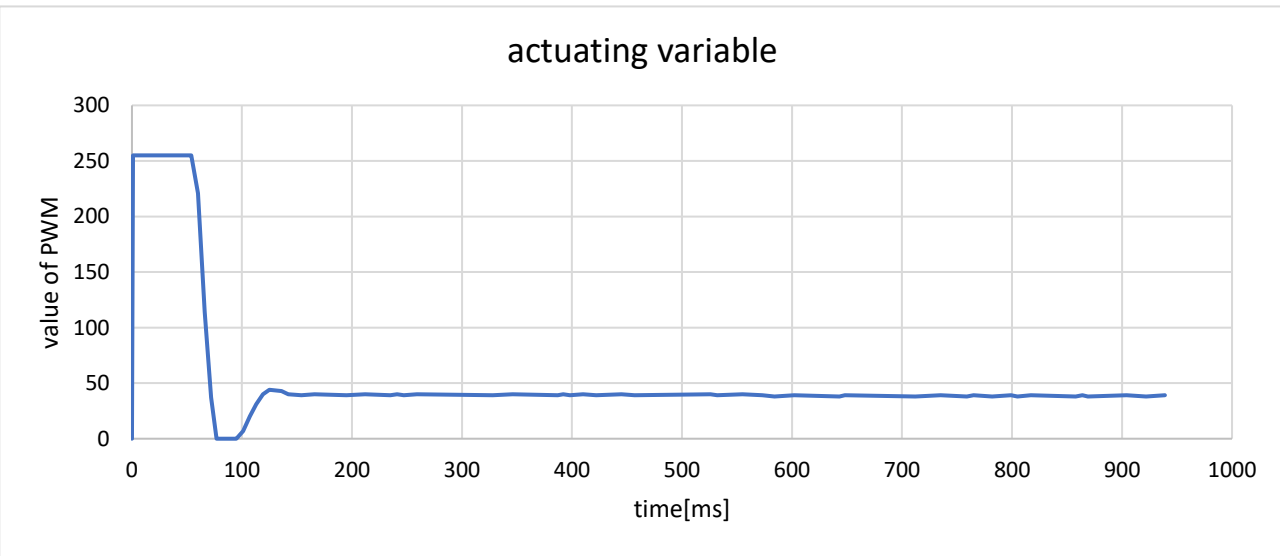
We did measure with these parameters:

Table 7.3 – Parameters used to test tuned P regulator

Reinforcement of regulator	18,2
Set temperature	70°C



Picture 64 – Temperature, control error, tuned P regulator



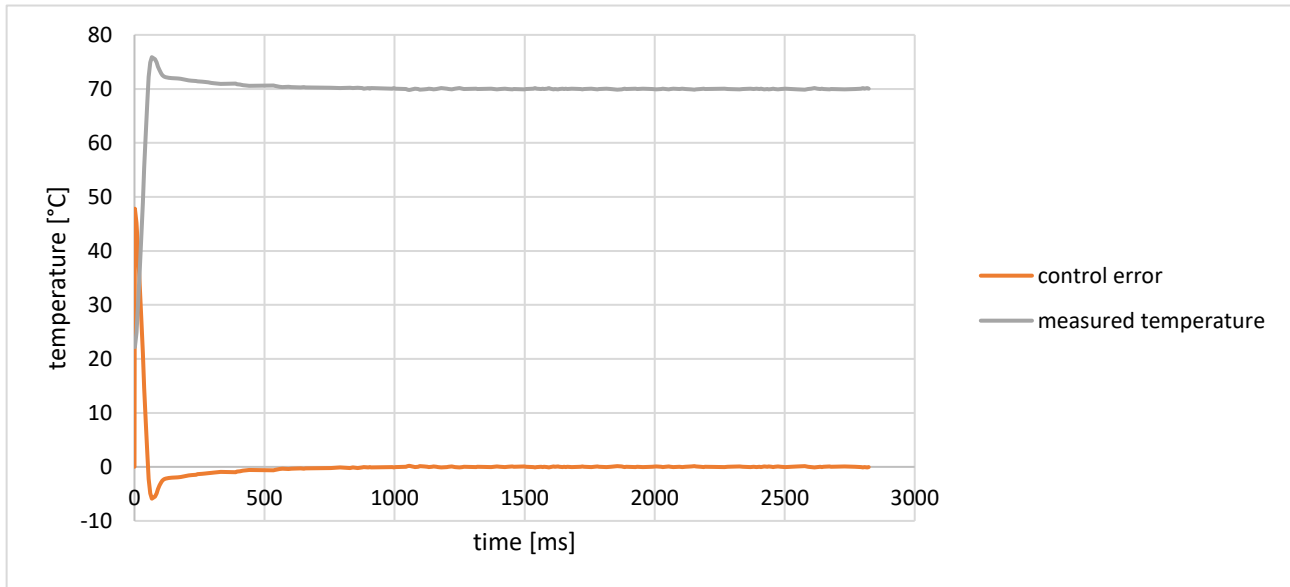
Picture 63 – Actuating variable, tuned P regulator

7.1.2 Proportional integrating regulation

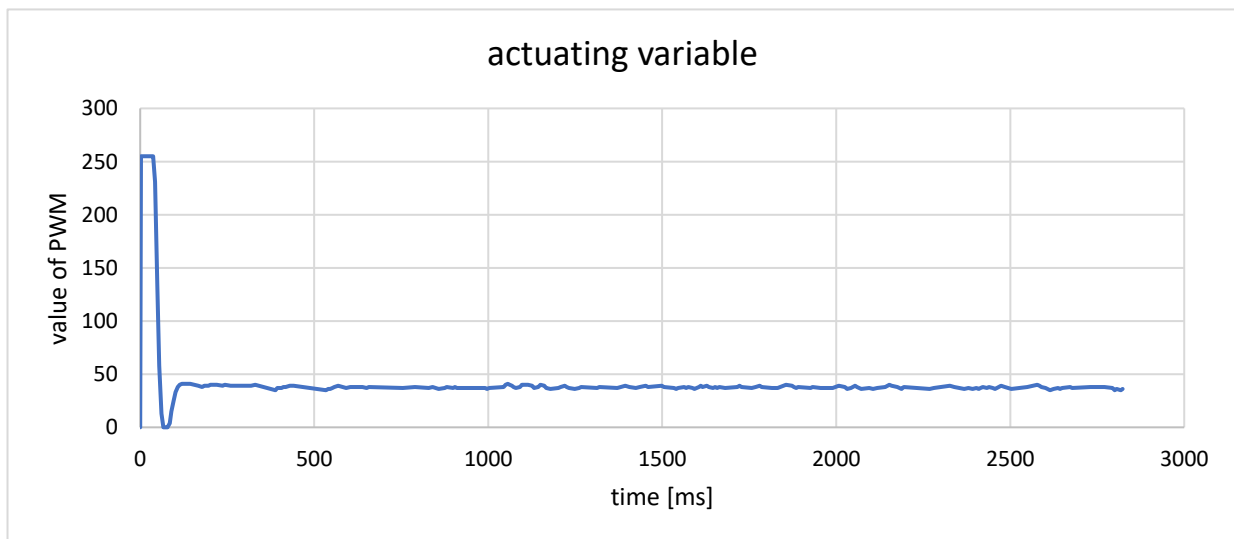
We did measure with these parameters:

Table 7.4 - Parameters used to test the tuned PI regulator

Reinforcement of regulator	16,4
Integrating time constant	0,4
Set temperature	70°C



Picture 65 - Temperature, control error, tuned P regulator



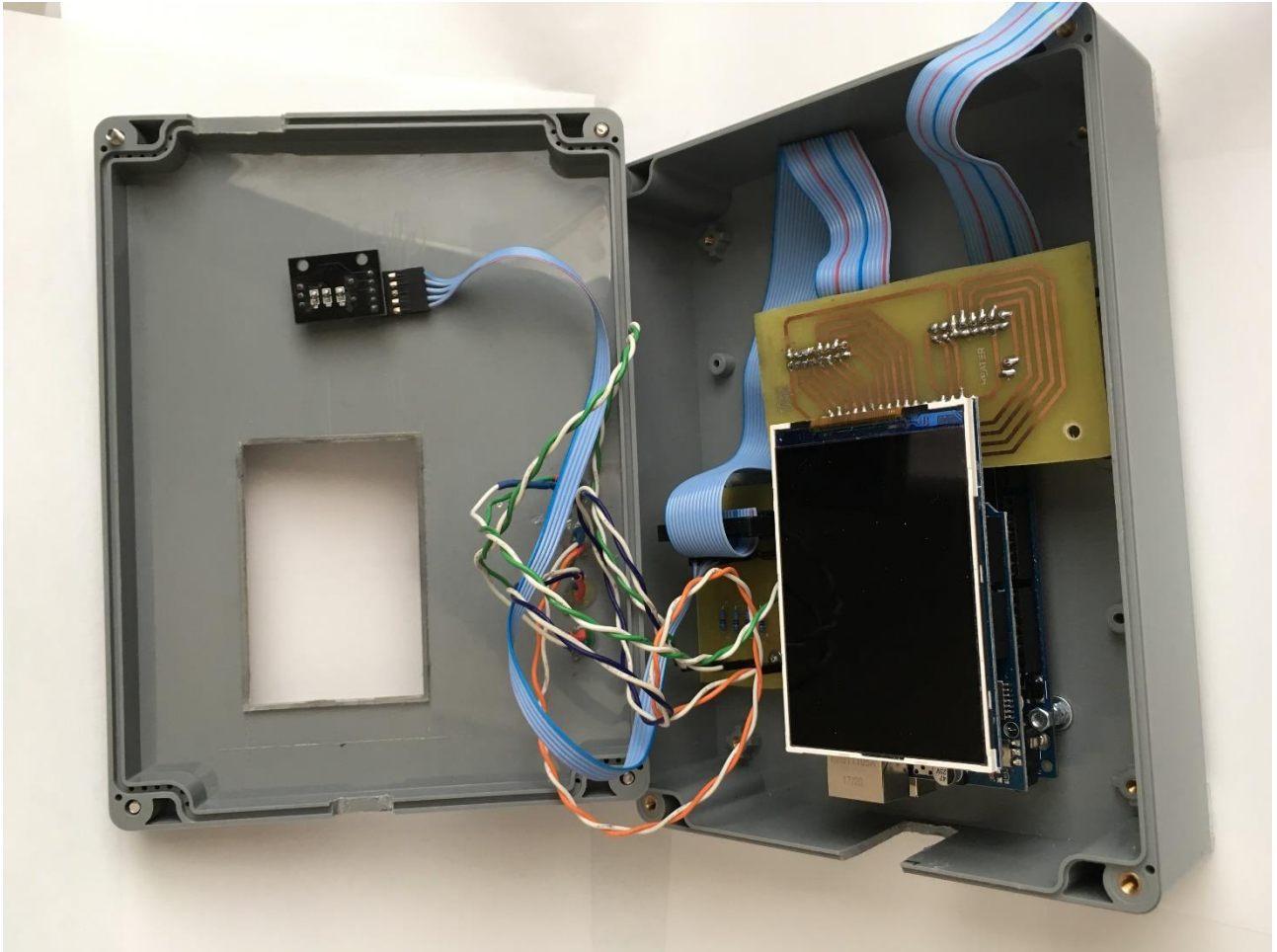
Picture 66 – Actuating variable, tuned P regulator

As we can see, the bouncing isn't periodically. That means that bouncing can be from opened window or draft. We could eliminate the option, that we have something wrong in the code.

8 Final model

The last part is to connect all the components mentioned above to one model. We screwed all PCB s to cases; we connect all peripherals to Arduino.

8.1 Controller of regulator

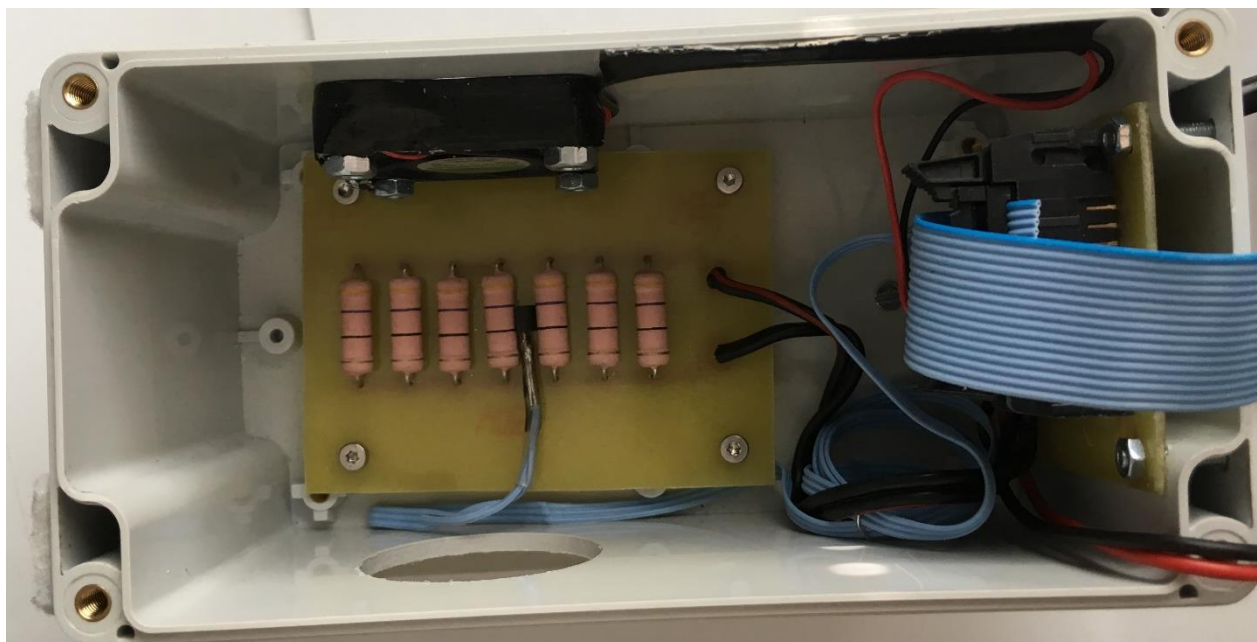


Picture 67 - Assembling case of controller



Picture 69 - Assembled controller

8.2 Chamber with the heating element



Picture 68 - Assembling chamber (open)



Picture 70 - Assembled chamber

9 Conclusion

In the end, we must say, that conception of the regulator is fulfilled. We build a model for demonstrating temperature regulations, web interface with graphs. During realisation and testing, we found many problems, but mostly we figured out.

Some problems we couldn't solve. These problems, we think are not caused by wrong programming of a microcontroller or by wrong wiring. The microcontroller causes these problems — for example, we are trying to solve the downloading of measuring files. We sent from web browser requirement via a URL address. This method is called GET. The problem is, that if we don't refresh the browser, the server run at Arduino couldn't redirect us. We could refresh the page, but this will be disturbing for the user. We could solve it via javascript, but this solution, we think it is too complicated to the size of the regulator.

10 References

1. **Mgr. Tomáš Zárybnický et al.** Elektronická učebnice. [Online] [Cited: 7 March 2019.] <https://eluc.kr-olomoucky.cz/verejne/lekce/957>.
2. **SPŠE Havířov.** Číslicové řídicí systémy. [Online] 05 11 2017. [Cited: 7 March 2019.] published at school intranet.
3. **ST Microelectronic.** [Online] May 2008. [Cited: 7 March 2019.] <https://www.st.com/resource/en/datasheet/cd00001225.pdf>.
4. **Mean Well.** farnell.com. [Online] 11 3 2016. [Cited: 7 March 2019.] http://www.farnell.com/datasheets/2547981.pdf?_ga=2.189240107.1092266099.1551970860-455427088.1537292678.
5. **Integrated, Maxim.** maximintegrated.com. [Online] - - 2018. [Cited: 7 March 2019.] <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>.
6. **Skalický, Jiří.** *Teorie řízení I.* Brno: Vysoké učení technické, 2002. ISBN 8021421126.



MSZC Kandó Kálmán Informatikai Szakgimnáziuma, Miskolc, Hungary
Srednja poklicna in tehniška šola, Murska Sobota, SLOVENIA

TESLA COIL DOCUMENTATION



Co-funded by the
Erasmus+ Programme
of the European Union

Student: Jan Slavic

Tutor: Darko Oskomič

Student: Bácsalmási Gergő

Tutor: Sándor Péter

Table of Contents

1. Intrudocion	1
2. Theoretical part	2
2.1 Market analysis	2
2.2 Technical analysis	2
2.2.1 Transformers	2
2.2.2 Resonance circuits	4
2.3 Used Technologies	7
2.3.1 Slayer exciter	7
2.4 Financial Analysis	8
3. Practical part	8
3.1 Electronics	9
3.2 Mechanical	10
3.2.1 The secondary coil construction	10
3.2.2 The primary coil construction	10
3.2.3 The toroid	11
3.3 Testing	11
4. Conclusion	12
a. The Project	12
b. The Tesla coil	12
c. Teamwork	12

Table of Figures

Figure 1: Tesla coil	1
Figure 2: Nikola Tesla	1
Figure 3: A circuit of a transformer	2
Figure 4: Transformer ratio equation.....	3
Figure 5:Transformer power equation.....	3
Figure 6: Real transformer	3
Figure 7: Resonant frequency formula.....	4
Figure 8: Slayer exciter diagram.....	7
Figure 9: 2n2222 transistor	9
Figure 10: Slayer exciter circuit	9
Figure 11: The secondary coil.....	10
Figure 12: The primary coil.....	10
Figure 13: The toroid	11
Figure 14: Flourescent tube light bulb	11

Table of tables

Table 1: SWOT table	8
---------------------------	---

1. Intrudocion

In the time of wireless charging and radio waves filling our space, let us take a look at the device that started it all, the famous tesla coil. It was originally invented by Nikola Tesla, a Serbian inventor. He was born in 1856 in the Austrian Empire. Tesla received an advanced education in engineering and physics and gained practical experience in the early 1880s working in telephony and at Continental Edison in the new electric power industry. In the summer of 1889, Tesla traveled to the 1889 Exposition Universelle in Paris and learned of Heinrich Hertz' experiments that proved the existence of electromagnetic radiation, including radio waves. Tesla decided to explore these experiments

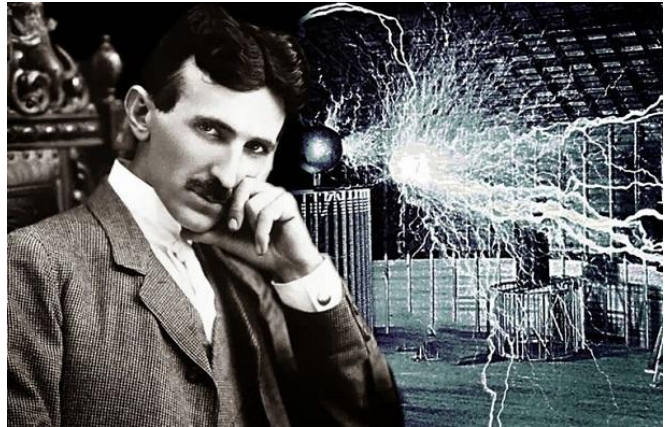


Figure 2: Nikola Tesla

more fully. In repeating, and then expanding on, these experiments, Tesla tried powering a Ruhmkorff coil with a high speed alternator he had been developing as part of an improved

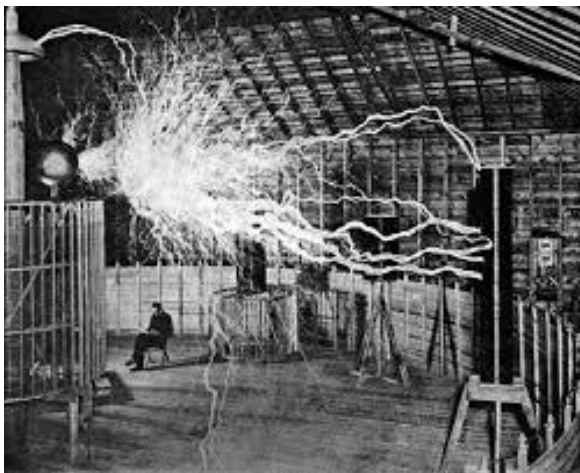


Figure 1: Tesla coil

arc lighting system but found that the high-frequency current overheated the iron core and melted the insulation between the primary and secondary windings in the coil. To fix this problem Tesla came up with his Tesla coil with an air gap instead of insulating material between the primary and secondary windings and an iron core that could be moved to different positions in or out of the coil.

A Tesla coil is an electrical resonant transformer. Connected to the right circuit it can produce very high voltage. It was commercially used in sparkgap radio transmitters and in medical equipment such as electrotherapy and violet ray devices. Today, their main usage is for entertainment, educational displays and experiments. While the voltage that the transformer puts out is very high, the current on this small of a scale is very tiny that's why it is not lethal. Tesla's dream was to bring wireless power to every home in the world, and he proved that it is possible. If we continue his work, we will be able to make his dream come true.

2. Theoretical part

The Tesla coil is a resonant air gap transformer. That means that at the core of it are two wire windings. One of the windings is referred to as the primary and the second one as the secondary. The secondary winding is actually the main tower of the Tesla coil and the primary winding are the few turns of wire on the bottom of the coil. The primary is also the coil that gets driven by the drive circuitry. Most Tesla coil transformers are air gap transformers, which means that between the primary and secondary the galvanic isolative material is air.

2.1 Market analysis

Tesla coils are mostly used for entertainment and education purposes and they are quite dangerous and complicated to build, so the consumer version of a Tesla coil is not the smartest idea, but as people started to put videos of Tesla coils on the internet the public started to get interested in the phenomenon of the Tesla coil. And so a company of the name OneTesla was born. The company started with a kickstarter campaign to raise the money for their company. There they were selling a Tesla coil kit and gathered feedback from hundreds of builders of the original kit to determine what could be improved.

2.2 Technical analysis

A Tesla coil looks like a complicated device but broken down in its core components it is quite simple. In this section we will see a couple of the technologies used in a tesla coil.

2.2.1 Transformers

Let us take a look at the core device which a Tesla coil is, basically a transformer. Transformers are devices which take an alternating voltage on the primary or the input and transfers it to the secondary or the output. A varying current in one coil of the transformer produces a varying magnetic flux, which, in turn, induces a varying electromotive force across a second coil wound around the same core.

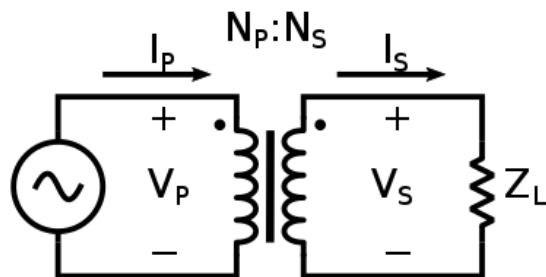


Figure 3: A circuit of a transformer

Depending on the ratio of primary to secondary turns the voltage can be either lowered or risen or it can stay the same if the ratio of turns is 1 : 1. The higher the amount of secondary turns in proportion to the primary the voltage on the output will be higher, also the opposite is true. We can calculate the voltage on the output of the transformer with the following equation:

$$\text{Turns ratio} = \frac{V_P}{V_S} = \frac{N_P}{N_S} = a$$

Figure 4: Transformer ratio equation

The equation states, that the turns ratio is equal to the number of primary turns (N_P) divided by the number of secondary turns (N_S) which also equals to the voltage on the primary (V_P) divided by the voltage on the secondary (V_S).

But this voltage increase does not come without a cost, because as the voltage increases the current lowers. By the law of conservation of energy, apparent, real and reactive power are each conserved in the input and output:

$$S = I_P V_P = I_S V_S$$

Figure 5: Transformer power equation

Where **S** is conserved power and **I** is current.

As this seems to be quite easy math, the problem is that these equations are true only for an ideal transformer. In case of a real transformer we have to encounter other problems like:

- Core losses
- Power losses (because of wire resistance)
- Different capacitances

But it gives us an idea of what the output voltage should be. The development of an electrical transformer is a complicated task, but the basic theory behind it is quite simple.

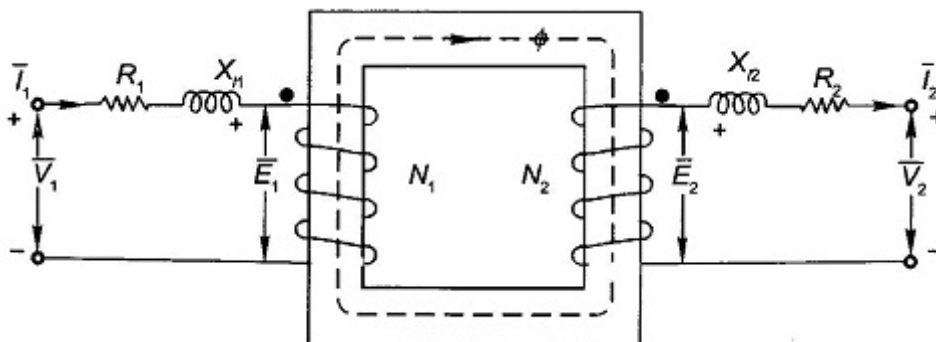


Figure 6: Real transformer

2.2.2 Resonance circuits

The main part of a Tesla coil is a Transformer, but as the secondary of a Tesla coil is not connected to any load, because that would collapse the output voltage, but it has stray parasitic capacitance. This is the capacitance between one side of the secondary winding and the air or space around it, because of that the secondary winding and the parasitic capacitance makes an LC circuit. If we want the transformer to work, we have to drive the primary at the resonant frequency of the secondary coil. So first we have to calculate the resonant frequency of the secondary coil, we do that with the next equation:

$$F = \frac{1}{2\pi\sqrt{LC}}$$

Figure 7: Resonant frequency formula

After we have the resonant frequency of the secondary coil, then we know with what frequency to drive the primary at and we can plan our drive circuitry accordingly. But these calculations do not encompass all the physical properties. That is why it smart to put some way of frequency adjustment to the circuit. The resonance frequency of the secondary coil can be tuned by adding a capacitor to it. In Tesla coils that capacitor is a toroid mounted to the top of the coil.

2.2.3 Driving or powering a Tesla coil

There are many ways you can drive a Tesla coil. Here we are going to take a look at some ways:

Spark gap circuit

The circuit operates in a rapid, repeating cycle in which the supply transformer (T) charges the primary capacitor (C1) up, which then discharges in a spark through the spark gap, creating a brief pulse of oscillating current in the primary circuit which excites a high oscillating voltage across the secondary

Current from the supply transformer (T) charges the capacitor (C1) to a high voltage.

When the voltage across the capacitor reaches the breakdown voltage of the spark gap (SG) a spark starts, reducing the spark gap resistance to a very low value. This completes the primary circuit and current from the capacitor flows through the primary coil (L1). The current flows rapidly back and forth between the plates of the capacitor through the coil, generating radio frequency oscillating current in the primary circuit at the circuit's resonant frequency.

The oscillating magnetic field of the primary winding induces an oscillating current in the secondary winding (L2), by Faraday's law of induction. Over a number of cycles, the energy in the primary circuit is transferred to the secondary. The total energy in the tuned circuits is limited to the energy originally stored in the capacitor C1, so as the oscillating voltage in the secondary increases in amplitude ("ring up") the oscillations in the primary decrease to zero ("ring down"). Although the ends of the secondary coil are open, it also acts as a tuned circuit due to the capacitance (C2), the sum of the parasitic capacitance between the turns of the coil plus the capacitance of the toroid electrode E. Current flows rapidly back and forth through the secondary coil between its ends. Because of the small capacitance, the oscillating voltage across the secondary coil which appears on the output terminal is much larger than the primary voltage.

The secondary current creates a magnetic field that induces voltage back in the primary coil, and over a number of additional cycles the energy is transferred back to the primary. This process repeats, the energy shifting rapidly back and forth between the primary and secondary tuned circuits. The oscillating currents in the primary and secondary gradually die out ("ring down") due to energy dissipated as heat in the spark gap and resistance of the coil.

When the current through the spark gap is no longer sufficient to keep the air in the gap ionized, the spark stops ("quenches"), terminating the current in the primary circuit. The oscillating current in the secondary may continue for some time.

The current from the supply transformer begins charging the capacitor C1 again and the cycle repeats. This entire cycle takes place very rapidly, the oscillations dying out in a time of the order of a millisecond. Each spark across the spark gap produces a pulse of damped sinusoidal high voltage at the output terminal of the coil. Each pulse dies out before the next spark occurs, so the coil generates a string of damped waves, not a continuous sinusoidal voltage.[18] The high voltage from the supply transformer that charges the capacitor is a 50 or 60 Hz sine wave. Depending on how the spark gap is set, usually one or two sparks occur at the peak of each half-cycle of the mains current, so there are more than a hundred sparks per second. Thus the spark at the spark gap appears continuous, as do the high voltage streamers from the top of the coil.

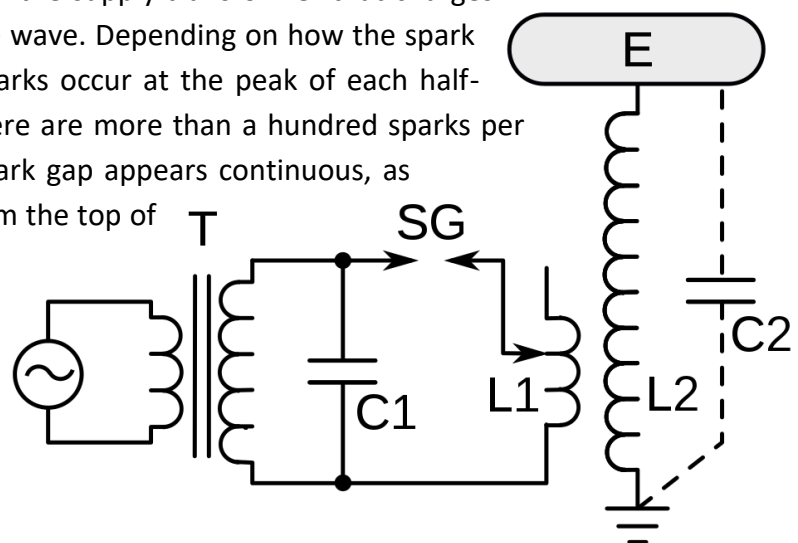


Figure 8: Spark gap Tesla coil circuit

Solid state driver circuits

Modern transistor or vacuum tube Tesla coils do not use a primary spark gap. Instead, the transistor(s) or vacuum tube(s) provide the switching or amplifying function necessary to generate RF power for the primary circuit. Solid-state Tesla coils use the lowest primary operating voltage, typically between 155 and 800 volts, and drive the primary winding using either a single, half-bridge, or full-bridge arrangement of bipolar transistors, MOSFETs or IGBTs to switch the primary current. Vacuum tube coils typically operate with plate voltages between 1500 and 6000 volts, while most spark gap coils operate with primary voltages of 6,000 to 25,000 volts. The primary winding of a traditional transistor Tesla coil is wound around only the bottom portion of the secondary coil. This configuration illustrates operation of the secondary as a pumped resonator. The primary 'induces' alternating voltage into the bottom-most portion of the secondary, providing regular 'pushes' (similar to providing properly timed pushes to a playground swing). Additional energy is transferred from the primary to the secondary inductance and top-load capacitance during each "push", and secondary output voltage builds (called 'ring-up'). An electronic feedback circuit is usually used to adaptively synchronize the primary oscillator to the growing resonance in the secondary, and this is the only tuning consideration beyond the initial choice of a reasonable top-load.

In a dual resonant solid-state Tesla coil (DRSSTC), the electronic switching of the solid-state Tesla coil is combined with the resonant primary circuit of a spark-gap Tesla coil. The resonant primary circuit is formed by connecting a capacitor in series with the primary winding of the coil, so that the combination forms a series tank circuit with a resonant frequency near that of the secondary circuit. Because of the additional resonant circuit, one manual and one adaptive tuning adjustment are necessary. Also, an interrupter is usually used to reduce the duty cycle of the switching bridge, to improve peak power capabilities; similarly, IGBTs are more popular in this application than bipolar transistors or MOSFETs, due to their superior power handling characteristics. A current-limiting circuit is usually used to limit maximum primary tank current (which must be switched by the IGBT's) to a safe level. Performance of a DRSSTC can be comparable to a medium-power spark-gap Tesla coil, and efficiency (as measured by spark length versus input power) can be significantly greater than a spark-gap Tesla coil operating at the same input power.

2.3 Used Technologies

2.3.1 Slayer exciter

The slayer exciter circuit is probably the simplest thing that can generate such high voltage.

This is a basic representation of a slayer exciter and how it is connected to the Tesla coil. The capacitor connected with the striped line is the stray or parasitic capacitance. The other parts on the diagram are as follows: Part Q is the transistor, T is the transformer or the physical Tesla coil, R is a resistor and D is the diode.

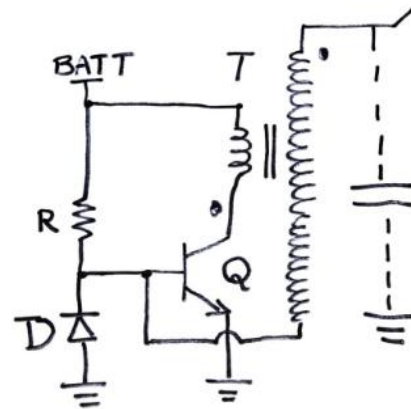


Figure 9: Slayer exciter diagram

The slayer exciter works like this:

1. Turning the circuit on, R drives the base of the transistor Q
2. Q turns on and drives current into the primary of the transformer. The current is limited because of the limited available base current.
3. The created magnetic field drives the secondary of the transformer.
4. The secondary voltage wants to grow large, but the tiny stray capacitance on the output resists the change, although very small, against the rise of the output end and so in return the voltage on the other end of the transformer goes down, pulling the base of the transistor low.
5. Diode D prevents the base voltage to fall more than 0.7V below ground, which in return pushes the output end of the secondary high.
6. The transistor turns off and so the magnetic field starts to reduce.
7. The base voltage rises again, and Q turns on and the cycle repeats.

The beauty of this circuit is that it tunes itself, because of the electromagnetic radiation the coil puts out. With other options like a function generator or a custom circuit is that you have to tune the coil to the exact resonant frequency of the secondary. The slayer exciter circuit is the simplest yet least powerful of the options.

2.4 Financial Analysis

SWOT Analysis	
Strengths:	Cheap, easy to understand how it works, safe,
Weaknesses:	The transistor is overheating, because of the power the coil requires, complicated mechanical construction, the coil is weak
Opportunities:	With a better circuit it can output more power,
Threats:	_____

Table 1: SWOT table

3. Practical part

In our case the problem was building a Tesla coil. We knew going in to this project that it would not be a simple task to accomplish. We started of by looking at some pictures of Tesla coils on the internet, we could then decide on the size of the secondary coil. When we got the wire we wrapped the secondary coil, the exact dimensions will be described in the Mechanical section. Then we started work on the primary coil construction. We had to watch out that it would end up bigger in diameter then the secondary. After that we made toroid on top of the tower. Sadly because of the low power drive circuitry we could only use it as a show piece. With all the parts mounted to a piece of wood, we could design the drive circuitry. Thankfully we found a circuit called a slayer exciter, which consists only of a couple components. The circuit was build on a breadboard for easy parts replacing and adjustment and connected with aligator clips to the primary for adjusting the number of primary turns and with that in theory rising the voltage. With all the components connected we turned it on and were astounded by the fact that it actually worked.

3.1 Electronics

The slayer exciter is quite a simple circuit, it only consists from a transistor, a resistor, a diode and a switch. But in our case we found out that the diode is not necessary. We did not use it because it limits the voltage on the base of the transistor and with that it also limits the power output of the transistor, which in turn limits the power of the Tesla coil. The problem appears when we run the coil for too long, because the transistor can burn out. But we are willing to sacrifice the possible destruction of the transistor for some more power on the coil. Not many transistors could be used in this circuit,

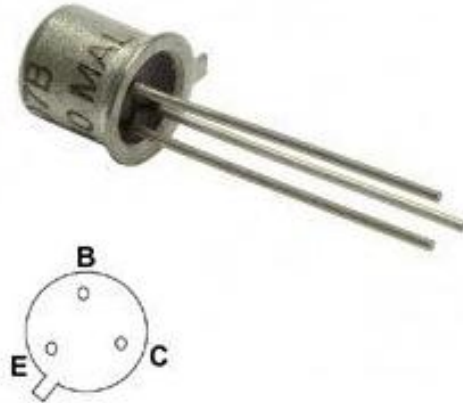


Figure 10: 2n2222 transistor

because for the specific values that the transistor has. We used a 2N2222 transistor in a metal can package. As for the switch we used a snap action momentary switch, the reason being so the Tesla coil is not turned on for too long and in turn burn out the transistor. On the circuit we can also see the number of turns on the primary which is 10 and the secondary number of turns 700.

The specific slayer exciter circuit used in our project is shown below.

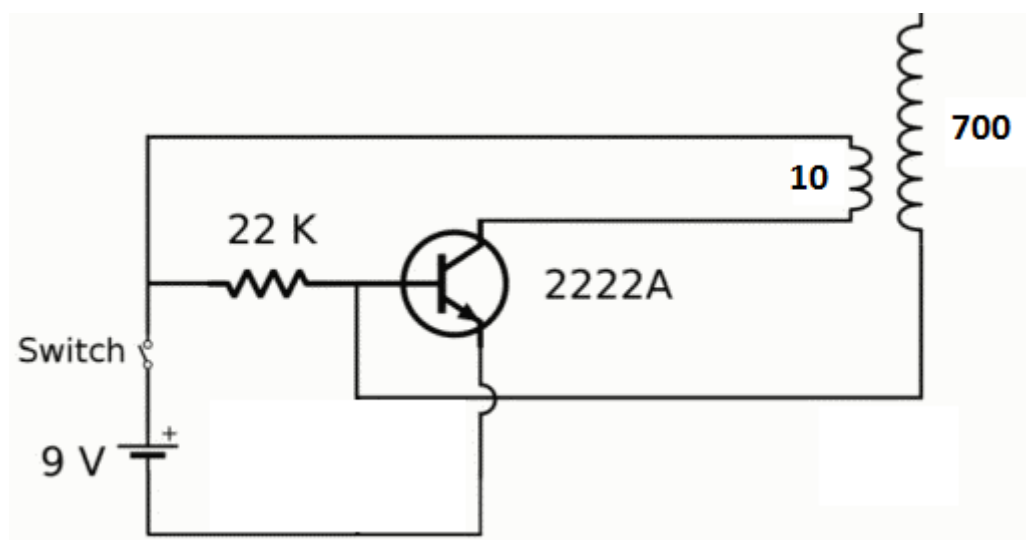


Figure 11: Slayer exciter circuit

3.2 Mechanical

3.2.1 The secondary coil construction

For the physical build we started of by winding secondary coil on a piece of PVC pipe. As we wanted the secondary coil to have around 700 turns and we had 0,3 mm diameter enameled copper wire, we calculated that we need the pieace of pipe that is a little bit longer than 21 cm. We calculated the length of the pipe by multiplying the number of turns by the diameter the wire and we got the length of the pipe in milimeters. Then we added a little to that number what for mounting purposes. We choose to get a pipe with a diameter of 5 cm. After we got the pipe we wound the coil by hand and secured it on both ends with hot glue. The coil got mounted to pieace of wood.



Figure 12: The secondary coil

3.2.2 The primary coil construction

We started the primary of by finding a plastic circle that was a little bigger in diameter then the secondary. Then we build plastic posts which would hold the wire. The posts were made from plastic stryips cut from 2 mm acrylic. And then 10 holes were drilled into these posts on 6 mm spacing between the holes. After that we used 2 mm wire and made a coil from it. Then we fished it through the holes so the posts were evenly spaced. The post were attached to the plastic circle with super glue, screws and nuts. And that is how we got a primary winding for our Tesla coil.



Figure 13: The primary coil

3.2.3 The toroid

The final part of the build was to make the toroid. A toroid is used for tuning the resonance frequency of the Tesla coil. The toroid was made from 1,5 mm steel wire. First the wire was wound in a coil and then that coil was soldered to a wire circle, so we got a toroidal shape. We finished the toroid off by adding a cross in the middle so the toroid can be screwed to the top of the secondary coil. Sadly the toroid could not be used and therefore it is only a

showpiece, but the coil still works without the toroid.



Figure 14: The toroid

3.3 Testing

The testing was not complicated. We just turned it on by holding the momentary switch and waving a Fluorescent tube light bulb around the Tesla coil to see if it turns on. We found out that if we wanted the bulb to turn on, the bulb would have to touch the open side of the secondary winding first and then it would light up and stay light if not directly touching the secondary winding.



Figure 15: Fluorescent tube light bulb

4. Conclusion

The Project

In the end the project was very successful, we have made a device which is in the electronics world known as a big and complicated task. And we carried it out in a simple yet working manner. We learned a lot about Tesla and his invention and gained a lot of knowledge about electronics and physics.

The Tesla coil

In the future we will continue to improve the Tesla coil by adding and designing a whole new circuit. With the new circuit it will be able to put more power through the coil and with that make bigger sparks and light up neon lights from a greater distance.



Teamwork

We gained happy memories. The teamwork was pretty hard first, but as the time passed we got to know each other, we spoke more. We got to know another countries culture and lifestyle. The living conditions were also different because coming from a city that is almost nine times smaller than the other city according to the population count and vice versa was a big change.



Stredná priemyselná škola elektrotechnická, Košice, SLOVAKIA
Miskolci SZC Kandó Kálmán informatikai szakgimnáziuma, Miskolci, HUNGARY

VOLTAGE METER WITH VOICE OUTPUT



Co-funded by the
Erasmus+ Programme
of the European Union

Content

List of abbreviations and symbols	4
Introduction	5
Theoretical part	6
Market Analysis	6
Technical analysis.....	6
Microcontroller	7
Analog to digital converter	7
Voltage reference.....	7
Circuit of measuring negative voltage	7
Measuring ranges	7
Power supply	8
Used technologies	8
Arduino	8
Voltage measuring	10
Measuring negative voltage	11
Measuring ranges	12
Financial analysis	12
Practical part.....	14
Electronics	14
Controlling	14
Voltage measuring	14
Measuring negative voltage	15
Measuring ranges	17
Power supply	18
Printed circuit board	18
Programs.....	19
Measuring part	19
Mechanical part.....	20
Testing.....	21
Conclusion	22
Reference.....	23
Attachments	24
Attachment no. 1	25
Images of device	25
Attachment no. 2	29

Schematics of device	29
Attachment no. 3.....	30
Simulation of absolute value converter.....	30
Attachment no. 4.....	31
Calibration	31
Attachment no. 5.....	33
Definition of information byte “statusByte”	33
Attachment no. 6.....	34
Program codes.....	34
Measuring part	34
Part of output peripherals	37

List of abbreviations and symbols

- **ADC** – analog to digital converter
- **DAC** – digital to analog converter
- **IDE** – integrated development environment
- **MSB** –most significant bit
- **LSB** –least significant bit
- **LCD** – liquid crystal display
- **ppm/°C** – parts per million per degree Celsius
- **PCB** – printed circuit board
- **op-amp** – operational amplifier

Introduction

On today's market is a big number of digital multimeters which are the basic meter for electricians. They are ideal for use outdoor and indoor, for electronic installation and microelectronics. Sometimes, when an electronic circuit is realized on printed circuit board (PCB) it's difficult to hold measuring probes on right place and don't make a short circuit during reading the value from display of multimeter. This is the reason of higher risk of destroying electronic device during measuring.

One solution of this problem is multimeter with voice output so it isn't necessary to look at the display when measuring because the value is spoken by the multimeter. Voice output is useful mostly during repairing and starting electronic devices made on PCB when is often used measuring voltage so for these applications voltage meter with voice output is enough. The current market provides only a few measuring instruments with the voice output what's the reason of realization voltage meter with the voice output in English, Hungarian and Slovak in this project.

Except measurements in electronics the voltage meter with voice output is probably the only way how visually impaired persons can measure the voltage.

Theoretical part

Market Analysis

Only a few digital multimeters with voice output are available on the market. We found Winhy 890S, Borbede BD-19A and NKTECH NK-51F. These multimeters can say the values in English and NKTECH NK-51F in Chinese too. Their advantage is the price from \$20 to \$28 but they can't say values in other languages for example in Slovak or Hungarian. The tolerance of DC voltage from manufacturer is 0.5% when measuring up to 200V and 0.8% when measuring up to 2000V.



Figure 1 - NKTECH NK-51F.

The only one digital multimeter with voice output in different languages than English and Chinese is possible to buy by special order in eshop Special Needs Computers as Talking multimeter. The packing contains digital multimeter digital caliper and device for talking measured values from both measuring instruments. The



Figure 2 - Multimeter and caliper with voice output.

maximum number of speaking languages is only two but a customer can choose from English, German, Italian, upon request from French and Spanish and from other languages with delivery delay up to 90 days. Multimeter measures on automatic ranges and the highest measured voltage is 600V. The tolerance isn't known. The disadvantage is the high price. Whole packing with multimeter, caliper and talking device costs \$1656 what isn't affordable price.

Voltage meter with voice output in more than two languages isn't currently on the market as well as affordable talking voltmeter in Hungarian or Slovak. Our voltage meter has voice output in three languages: English, Slovak and Hungarian.

Technical analysis

The goal of this project was to build a voltage meter with voice output in three languages – Slovak, Hungarian and English. Voltage meter measure voltage on two ranges 0-4V and 0-40V, tolerance of measurement is 0,01V and value of voltage is visible on display.

The way the project works is that we touch negative with negative and positive with positive and then the device will measure the voltage and it will state the value through the speaker. If it's on automatic-range setting then it will decide whether to use the 0-4V or the 0-40 range.

The two Arduinos communicate through i2c, the Arduino Nano will send the measured value in binary to Arduino Uno and with the help of that value it will display the correct numbers on the LCD and state it through the speaker.

Microcontroller

The basic element of voltage meter with voice output is microcontroller. In our voltage meter two microcontrollers are used. The first one process data from ADC, sets input ranges, perform the appropriate mathematical operations, calculate the value of voltage and communicates with the second microcontroller. The second manages output peripherals of voltage meter. Receives data from the first microcontroller, process them, displays relevant information on LCD, controls the speech setting and says the value of voltage.

Analog to digital converter

The component which measures voltage is called analog to digital converter (ADC). It provides conversion from analogue signal to discrete digital, which is further send to microcontroller which processes it and recalculates the received value to voltage. The ADC needs the reference voltage which is used to measure input voltage. The resolution of voltage meter is 0,01V at both ranges and the base range is up to 4V what means that minimal resolution of ADC has to be 12 bits.

Voltage reference

Reference voltage is needed for ADC and the accuracy of analogue to digital conversion depends from its stability. 12 bits ADC is used in this voltage meter what means that ideal voltage reference is 4,096V. With this value we ensure that one step of ADC represents voltage 1mV.

Circuit of measuring negative voltage

The voltage meter contains absolute value converter to be able to measure negative voltage. This circuit ensures that both negative and positive value of voltages is converted to a positive value which does not damage the ADC and can be measured. To determine whether the input voltage is positive or negative accuracy comparator is used and the output is read by microcontroller.

Measuring ranges

Our voltage meter measures input voltage on two ranges: 0-4V and 0-40V. The output voltage from input voltage divider has to be less than 4,096V because we used voltage reference 4,096V. For the range 0-4V voltage divider ratio is 1:1 and for range 0-40V the ratio is 1:10. Input voltage divider is controlled digitally by a multiplexor to be able to make automatic switching between ranges. The voltage divider circuit also implements the measured voltage protection circuit, which ensures that the voltage after the divider is not higher than 4V even when the maximum measuring voltage of 40V is exceeded.

Power supply

Although this voltage meter is designed for wide use so it is battery powered device. The power supply must provide stabilized 5V for microcontrollers and voltage reference. Circuit of measuring negative voltage needs symmetrical 9V. This voltage has to be generated from battery voltage. The circuit of input ranges also use symmetrical power voltage.

Used technologies

Arduino

Arduino is an open-source electronic platform based on easy-to-use software and hardware. Arduino developing boards consists of microcontroller which is programmed in programming language Arduino in software Arduino IDE. Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments.

Arduino was born in Ivrea Interaction Design Institute as an easy tool for prototyping aimed at students and electronics enthusiasts. The worldwide community of students, programmers, professionals and artists contributed to the development and improvement of Arduino. When Arduino gained its place on market, a large number of shields were created to extend the application of this platform in a wide range.

Arduino developing platforms

Arduino developing boards include microcontroller, converter for the communication via USB, USB connector, connector of power and pins to connect peripherals making them ideal tools for prototyping and for embedded applications. The base element of each board is an 8-bit AVR microcontroller from Atmel or a 32-bit microcontroller. There is a large number of Arduino boards and shields on the market from basic to advanced. For our application following boards were considered: Arduino Uno, Nano, Mini and micro.

ARDUINO UNO

Arduino Uno is currently probably the most used board. Uno is direct continuation of the main development line, which began with the first Arduino with a serial port instead of USB. The board is based on Atmega328P microcontroller with 32kB flash memory, 2kB SRAM and 1kB EEPROM. The microcontroller is clocked with 16MHz crystal. The board communicates with the surrounding environment using 14 digital and 6 analog pins.



Figure 3 - Arduino Uno.

ARDUINO MINI

Arduino Mini is the smallest board with dimensions only 18 x 30mm. But there is an absence of USB communication converter, so the microcontroller must be programmed by an external programmer. Used microcontroller is ATmega328 with 14 digital and 8 analog pins. However, this board is gradually retreating from the market and is no longer produced officially.

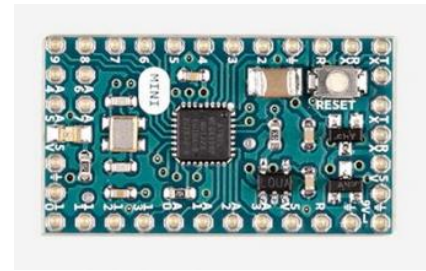


Figure 4 - Arduino Mini.

ARDUINO MICRO

Arduino Micro is similar in size to Arduino Mini but in this case the USB communication converter is implemented in used microcontroller ATmega32U4. This is the reason why is this board the best for applications with computer where can be used for example like mouse or keyboard. Board has 20 input-output pins.



Figure 5 - Arduino Micro.

ARDUINO NANO

Arduino Nano is the smaller version of Arduino Uno. Dimensions are 18 x 45mm. However, it does not include power connector and the ATmega328 is SMD. On board are 14 digital and 8 analog pins. Mini USB connector and converter is used for the communication with computer.

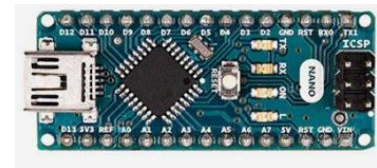


Figure 6 - Arduino Nano.

We chose Arduino Nano for our voltage meter – measuring part. Device does not need to communicate with computer, but small dimensions are an advantage.

DFPlayer

The DFPlayer Mini is a small and low-cost MP3 module with a simplified output directly to the speaker. The module can be used as a standalone module with attached battery, speaker and push buttons or used in combination with an Arduino UNO or any other with RX/TX capabilities.

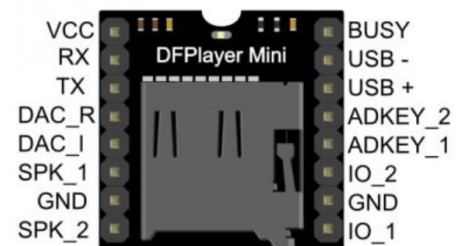


Figure 7 - DFPlayer

The DFPlayer perfectly integrates hard decoding module, which supports common audio formats such as MP3, WAV and WMA. Besides, it also supports TF card with FAT16, FAT32 file system. Through a simple serial port, you can play the designated music without any other tedious underlying operations.

Software Arduino

Arduino is programmed by the Arduino Programming language in Arduino IDE. IDE means that programming software include all necessary parts for writing and compiling code and programming microcontroller, so

there is no need for additional software equipment. Arduino IDE is an open-source application which works with Windows, Mac OS X and Linux. Arduino Programming language is based on hardware programming language Wiring, which is similar to programming language C++. This programming language we used for programing voltage meter with voice output. Details about the program code is in the chapter Programs.

Voltage measuring

Analog to digital converter (ADC) is an electronic component used to convert analog signal to digital. It is the core of measuring instruments which measure input voltage and communicate with microcontroller which further access received data. ADC compare measuring voltage with reference voltage which is divided by the resolution of ADC. The output is an integer expressing input voltage to the reference voltage. Nowadays there are a large number of ADCs that use different methods, output communication and accuracy. The most common principles measuring voltage by ADC are: parallel comparator method, integrating method and successive approximation.

Parallel ADC

The first group is parallel ADC. Converters using this conversion principle utilize a process of comparing the measured voltage to the threshold voltages at the resistive divider. In order to determine the value that the input voltage has already exceeded a comparator is attached to each step of the step transmission characteristic. Outputs of comparators are processed by decoder to binary code. Benefit of this method is the smallest conversion time (1 cycle). If the reference voltages at the comparator inputs are distributed unevenly based on the desired function dependency, it is possible to create a nonlinear ADC with the desired function without a more serious structural change.

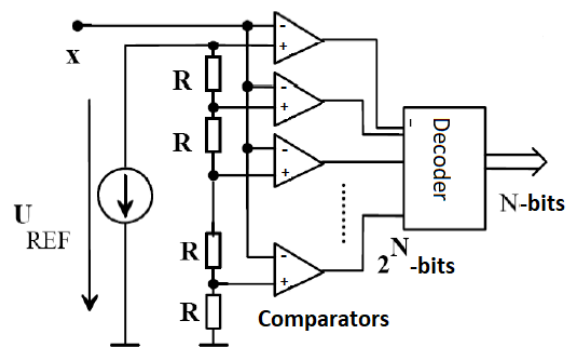


Figure 8 - Structural schematic of parallel ADC.

Integrating ADC

The second group consists of ADCs with inter-conversion to time interval. They consist of two parts, where in the first part the input voltage is converted into a pulse whose duration is proportional to its size. In

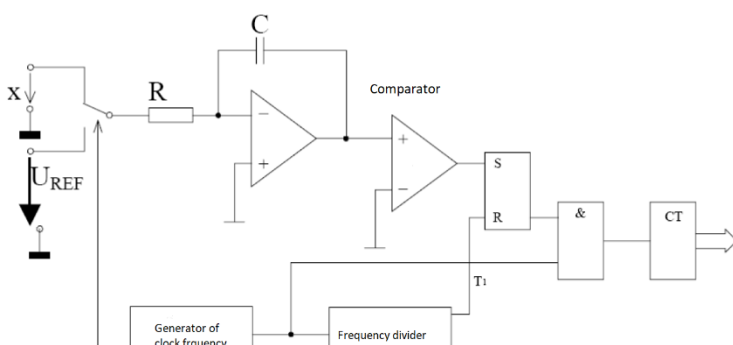


Figure 9 - Structural schematic of integrating ADC.

the second part, the duration of this interval is converted to a number by a simple timer.

The most accurate converter of this group is the double integrating ADC. Its activity is spread over two time periods. In the first one, the measured voltage x is

connected to the input integrator. The duration of this stage is constant, determined by the time T . At the end of this first phase of conversion, the output voltage of the integrator is proportional to the value of the input voltage U_Y . $U_Y = \frac{1}{RC} \int_0^T x \cdot dt = \frac{xT}{RC}$ During the second phase, the output voltage of the integrator is discharged by a constant current from the U_{REF} reference voltage source. The TX time at which the integrator voltage exceeds zero depends on its initial value. $T_x = \frac{xT}{U_{REF}}$ The resulting digital data is obtained by counting the number of periods of frequency over the duration of the time interval T_x .

The biggest advantage of these converters is the suppression of the periodic jamming signal due to the integration principle. This ADCs are very accuracy but on the other hand the conversion time is relatively large (2^N cycles). The non-linear transmission is achieved by a differently arranged integration circuit, using different capacitor discharge methods. This results in a reduced number of available conversion functions.

Approximation ADC

Approximation ADCs are characterized by a DAC in feedback. The output of the DAC is compared to the input voltage x and the control logic sets DAC according to the output of the comparator.

The most common are successive approximation ADCs. The DAC setting algorithm is based on setting MSB to 1. If the measured voltage is greater, this bit remains set and the next bit is set to 1. If the measured voltage is smaller MSB is set to 0 and next bit is set to 1. This procedure is repeated after LSB, when the DAC content is the output value. These ADCs have a relatively short conversion

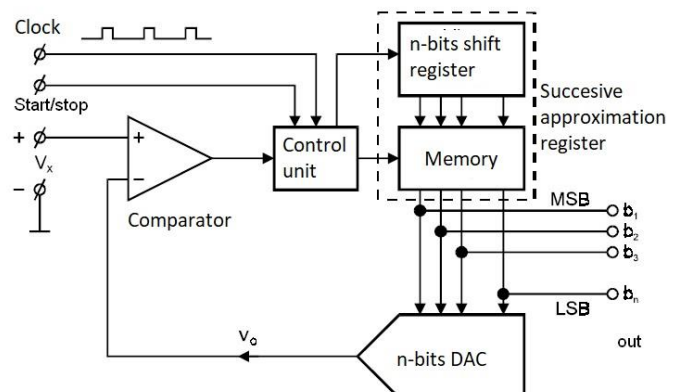


Figure 10 - Structural schematic of successive approximation ADC.

time (N cycles) and represents the midpoint in the achievable measurement accuracy and conversion time. This type of transducer is sensitive to poor grounding arrangements. At the time of the transducer decision, voltage peaks arise on the inductance of poorly arranged ground connections, which are added to the input voltage and degrade the weighing of the respective bit. This is the reason why we use ADC using this method.

Measuring negative voltage

A feature that every digital voltmeter should have is the ability to measure negative voltage. However, the analog-to-digital converter can only measure the voltage within the reference voltage range, which is mostly positive, and thus the negative voltage would damage the converter. This problem has two solutions: to use a floating measuring ground or an absolute value converter.

FLOATING MEASURING GROUND

In this method, the negative terminal of the meter is not coupled to the zero potential of the device but has the potential at the center of the reference voltage of the ADC. This connection allows to measure both positive and negative voltage to the negative terminal of the meter. However, the measured voltage must not be higher or lower than half the reference voltage. This method has the advantage of simplicity of connection, but it is sensitive to shifting the potential of the measuring ground. The disadvantage is the accuracy of the measurement. If it is desired to measure the voltage at the device terminals with x resolution, a double resolution ADC is needed because the measuring range is only half the reference voltage and thus half the resolution.

ABSOLUTE VALUE KONVERTER

Using this method, the output voltage of the ABS converter is always positive and equal to the absolute value of the input voltage. Such a circuit is realized by operating amplifiers. The advantage is that by connecting the negative terminal and the zero potential, an ADC with the resolution required for measurement is sufficient. However, it is necessary to power the operational amplifiers with negative voltage, which requires higher power requirements. In a voltmeter with a voice output, we used an absolute to value converter to improve measurement stability.

Measuring ranges

In order to measure higher voltages than the ADC reference voltage, it is necessary to add a voltage divider to the input of voltage meter. The voltmeter is equipped with two dividing ratios that are switched manually or automatically based on the measured voltage. The first of the range switching options is toggle the switch. This solution is simple, but it is not possible to switch ranges automatically and multiple input voltage dividers reduce the total internal voltmeter resistance. The second method is based on changing the split ratio of the same voltage divider. Another resistor is connected to one of the resistors of the base divider what change the split ratio of a divider. This solution provides the ability to automatically switch ranges.

Financial analysis

STRENGTHS

- International cooperation
- Clear multilanguage voice output
- Versatility of usage of voltage meter

WEAKNESSES

- Limited time to work together on the project
- Higher price than multimeters on current market

OPPORTUNITIES

- Opportunity to offer measuring instrument to customers on the Slovak and Hungarian market

THREATS

- Misunderstanding or failure to communicate in a foreign language

The project budget was used for this voltmeter with voice output. The financial cost of our voltmeter is greater than the price of multimeter with voice output like a Winhy 890S but it is the only one of its kind. In fact, there is no similar voltmeter on the market that can speak in Slovak, Hungarian and English.

Practical part

Electronics

All electronic schematics are in the chapter Schematics of device.

Controlling

In our voltage meter with voice output we used two microcontrollers on developing platforms Arduino Nano and Arduino Uno. These two microcontrollers communicate with each other via I²C.

Microcontroller – measuring part

In the measuring part of the voltmeter is used Arduino Nano, which with its dimensions fit exactly on the printed circuit board. The I2C protocol is used to communicate with the ADC. The clock wire is connected to pin A5 and the data wire to A4. The output from the negative voltage detector is applied to pin A0 so voltage can be measured by the built-in ADC. This achieves higher accuracy because even a low logic zero voltage can be detected as a positive comparator output, which would not be possible with the Arduino digital pin. A digital pin 3 is used to control the input range circuit. A button is attached to the microcontroller to set individual measuring ranges. In order for the button to be read reliably, a capacitor C17 is attached to it to prevent unwanted glitches when the button is pressed and released.

Microcontroller – part of output peripherals

Voltage measuring

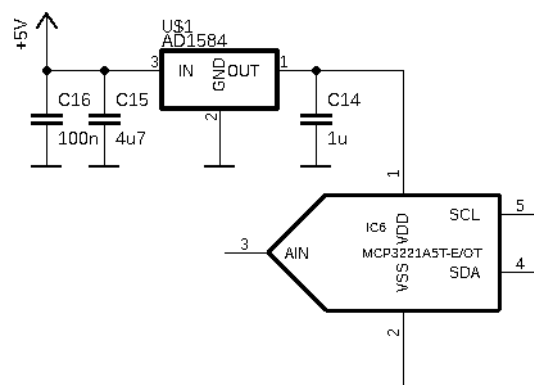


Figure 11 - Schematic of ADC and voltage reference.

Analog to digital converter

In our voltmeter we used the MCP3221 ADC from Microchip. The reference to the datasheet [2] is in the References chapter. It is a 12-bit successive approximation converter that communicates over an I2C protocol with a fixed address. The converter is equipped with sample and hold technology, which means that the input voltage is held constant throughout the conversion. The small power consumption and the SOT-23 case make this converter suitable for use in portable battery-powered devices. The maximum power current during conversion is 250μA and the typical standby current is 5nA. Powering the entire converter is directly

from the reference voltage, which can be from 2.7 to 5.5V. Pull-up resistors R14 and R17 are connected to the I2C wires according to the transmission frequency. We used 400kHz transmission frequency, according to which the value of resistors is 2k Ω .

Voltage reference

The AD1584 [3] voltage reference used for the ADC is from manufacturer Analog Devices. The reference voltage is 4.096V with a maximum tolerance of 1% and a temperature shift of voltage up to 50ppm / $^{\circ}\text{C}$. For proper operation, the input voltage must be lower than 12V and 600mV greater than the output voltage. The maximum output current is 5mA, which is several times more than the consumption of the ADC. The reference is made in the SOT23 case. Output C14 capacitors, C15 and C16, are added for reference voltage stability. In order to reduce the power loss to the reference, the stabilized 5V is used for powering.

Measuring negative voltage

The negative voltage measurement block consists of an absolute value converter and a negative voltage detector.

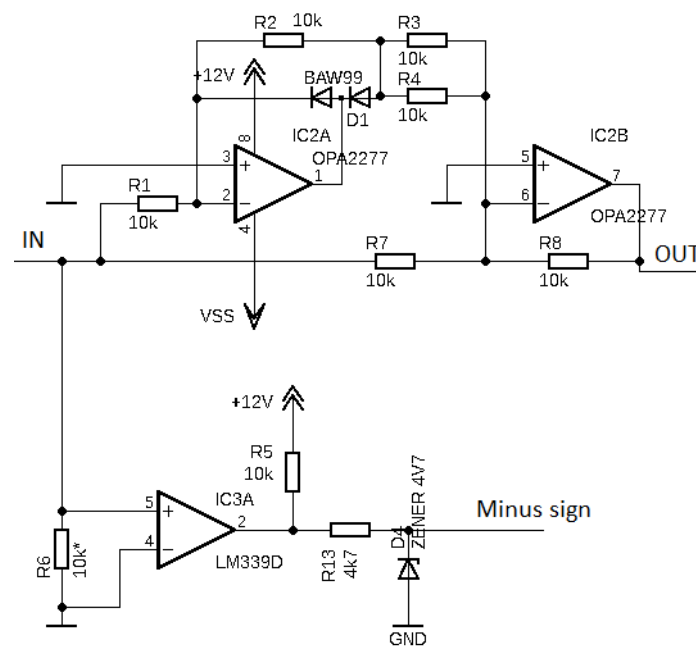


Figure 12 - Schematic of absolute value converter and negative voltage detector.

Absolute value convertor

An absolute voltage converter is a circuit that always provides a positive output voltage that depends only on the input voltage but not on the polarity. Any converter made only from diodes is not satisfactory because some voltage is needed to open the diodes, which the voltage meter does not measure. Therefore, special converters using operational amplifiers are used for absolute value converters. In our voltage meter we have used verified circuit published also in the magazine *Konstrukční elektronika A Rádio* 3/1996 [4] in chapter Rectifiers and converters to absolute value.

The used circuit includes two operational amplifiers with a signal applied to the inverting inputs. At negative input voltage, op-amp1 is positive, D1 generates feedback for op-amp1, diode D2 is closed, and therefore op-amp2 operates as an input voltage inverter. When the input voltage is positive, the op-amp1 output is negative and the feedback is closed via D2 and the resistor. At anode D2 is the inverted input voltage. The direct signal and the signal from the one-way rectifier are summed at the inverting input op-amp2. Since the signal from the one-way rectifier is doubly amplified, the op-amp2 output is positive again. For proper functionality, the converter requires a signal source with a small internal resistance, so one op-amp is connected before the converter.

Before we make the prototype of this converter, the circuit was simulated in the NI Multisim simulation program, where its satisfactory properties were confirmed. The schematic diagram of the simulation circuit (Figure 23) and the simulated graph of the output voltage versus input voltage (Figure 24) are given in Attachment no. 1. The graph shows high linearity in the working area from -4.1V to 4.1V. We used high precision OPA2277 operating amplifiers from Burr-Brown [5]. They have a very low offset voltage, only 10 μ V and a temperature shift of 0.1 μ V / °C. C3, C8 - C10 capacitors are added to make the converter output as clean as possible. The circuit uses two diodes. The solution is, for example, used diode BAW99 [623], which in one SOT23 case contains two identical diodes in series and thereby saves PCB space. The resistors used are with a small tolerance to ensure the same transducer characteristics at any input voltage polarity. When resistors were installed, they were measured on a precision multimeter and only resistors with the same resistance were selected.

Negative voltage detector

The absolute value converter ensures the measurement of negative voltage, but for the microcontroller it is necessary to indicate whether the input voltage is negative or positive. A negative voltage detector is used for this purpose. It consists of the voltage comparator LM339 [7] in the SO14 case, which compares the input voltage with the negative measuring clamp. The comparator is designed to sensitively compare low voltages, what is desirable in our application. Datasheet specifies 2mV. The output of this comparator is open collector, which means that if the voltage at the negative input is higher than the positive, the output is switched through the internal transistor to negative supply voltage. For this reason, an R5 resistor is added to provide a positive output voltage when the internal transistor is open. The zener diode 4.7V is added to the output to measure the comparator output by the microcontroller. The R6 resistor was only used in the prototype and is not soldered in the voltage meter.

Measuring ranges

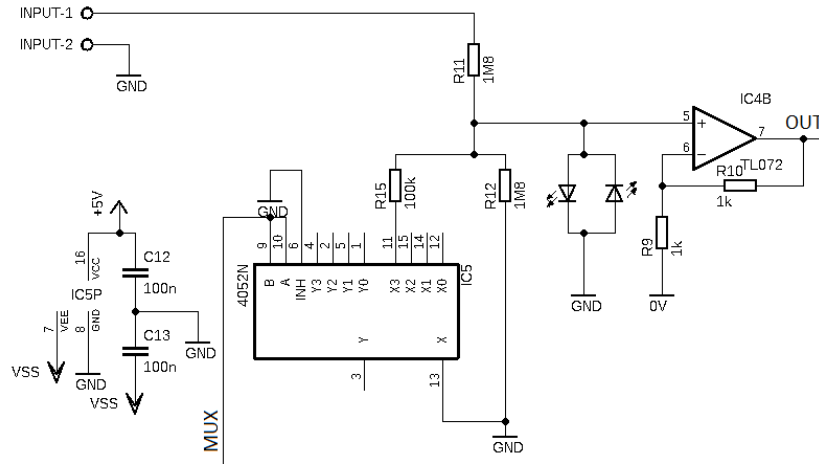


Figure 13 - Schematic of input ranges.

We used a voltage divider with variable dividing ratio to switch measuring ranges. The required voltage ratio of the input range circuit for the range up to 4V is 1: 1 and for the range up to 40V is 1:10. To use a voltage divider, these ratios are adjusted to 1: 2 and 1:20, and the output voltage is then amplified twice. The resistors R11 and R12 create a 1: 2 base divider. Resistor R15 is connected to this divider by multiplexer, which changes the dividing ratio to 1:20. The ratio of R15 to R11 and R12 resistors is calculated by adjusting the equation where R_x is the value of the resistors R11 and R12 and R_y the value of the resistor R15:

$$\frac{\frac{R_x \cdot R_y}{R_x + R_y}}{R_x + \frac{R_x \cdot R_y}{R_x + R_y}} = \frac{1}{20}$$

$$\frac{R_y}{R_x + 2R_y} = \frac{1}{20}$$

$$R_y = \frac{R_x}{18}$$

The used multiplexer is a double four-channel analog multiplexer CD4052B from Texas Instruments [8]. Only one channel is used for switching. Channel switching control is digitally using two pins 9 and 10, which we have connected, so the multiplexer switches the output with two channels. Typically, the resistance of the closed channel is below 7Ω at 25°C , which is a negligible value to the tolerance of the resistors. The multiplexer is powered positively by 5V and negatively 9V, which ensures switching of the divider even when measuring negative voltage. For stability, the circuit contains capacitors C12 and C13.

The variable voltage divider is followed by a device protection against a higher input voltage than the A / D converter reference voltage. Thus, the maximum voltage for input to the converter is 4V, which corresponds to 2V at the output of the divider. The protection is realized by antiparallel connection of two blue LEDs between the divider output and ground.

Power supply

An integrated circuit by Intersil is used as a voltage inverter [9]. The ICL7660S can be used at voltages from 1.5V to 12V and requires only two external components without a critical value. The circuit efficiency is about 97% under the used conditions and the internal resistance of the negative voltage source is about 100Ω. An external capacitor C1 according to the data sheet and C4 -C7 capacitors for stabilizing and filtering the voltage are connected.

The device is realized on two PCBs. One is a measuring part and the other is a shield for Arduino Uno as a board with output peripherals.

Used PCB is one-sided FR4 with five jumpers. PCB was produced by photo-way technology and etched in ferric chloride. The PCB was designed in EAGLE (Easy Applicable Graphical Layout Editor). The program does not include the Arduino Nano component, so it was necessary to download a library with this component. The analog wires are in most cases separated from the digital wires by the third wire, so that the digital signal is not induced to the measured voltage circuits. Capacitors are located as close to the power pins of the integrated circuits as possible. The PCB dimensions are 90 x 63mm, and contain all the parts of the measuring part, except the button and switch. They are connected by wires. The output peripheral part is connected to the measuring part via a four-pin connector. Two pins are used to power (9V, GND) and two to I2C (SCL, SDA). The connector is connected with a counterpart with

a cable soldered to the PCB with output peripherals. In Attachment no. 1 is a photo of the printed circuit board from component's side (Figure 20) and from connections' side (Figure 21).

Programs

The functionality of the entire device is ensured by programmed ATmega microcontrollers on Arduino development boards in the Arduino IDE environment. The device features are: range change, language change, voice output volume, displaying on LCD and voice output. Whole program code is in chapter Program codes.

Measuring part

The microcontroller of the measuring part is supposed to receive data from the ADC, calculate the voltage, set the input divider according to the selected range and send data to the other microcontroller. First, the button is read and the range is changed if necessary, then the data is received from the ADC and the voltage is calculated and finally the voltage is sent to the other microcontroller.

Change measuring range

Changing the measuring range is performed after pressing the button. The pin connected to the button is set as input with a pull up resistor and the pin connected to the input range circuit as output. Two functions were created in the program: *read_button()* and *range_update()*. Both functions are void.

The "*read_button()*" function checks if the button is pressed. If so, it will set the flag of the pressed button, and 100ms will not respond to the button change state, which is an anti-oscillation protection when the button is pressed. Subsequently, it changes the variable "range" determining the measuring range depending on its actual value. The button press flag ensures that if the button is still pressed, the measurement range is not changed multiple times and the program continues. If the button is no longer pressed, the 100ms program does not respond to the button change and cancels the button press flag.

The *range_update()* function sets the circuit of the input ranges based on the value of variable "range". The first command in the function is to call the *read_button()* function to update the variable range. If the 0-4V measurement range is selected, the microcontroller will set the input range circuit for a 1: 1 voltage transmission. If the 0-40V range is selected, it will set the input range circuit to a 1:10 voltage transmission. If the selected measuring range is automatic, the control of the input range circuit takes place according to the actual value of the measured voltage. If the measured voltage is higher than 4V, the range is 0-40V, if the voltage is less than 3.9V, the range is 0-4V. In order to keep the automatic range stable, hysteresis needs to be created when switching ranges. Therefore, the voltage limits are 4V and 3.9V.

Calculation of voltage

Another function in the program is *read_adc()*, which contains data receiving from the ADC and calculates the voltage. The microcontroller uses the I2C protocol to communicate with the ADC, so the

Wire.h library is added to the program to ensure communication through this protocol. The microcontroller of the measuring part triggers the I2C line at 400kHz as the master.

At the beginning of the function, a value of 0 is inserted into the variable “*voltage*”. The converter will send two bytes. The bytes are inserted into the variable “*adc*” and the first is shifted by 8 bits to the MSB because it is the upper byte of the 12-bit converter. The voltage is calculated according to the formula: $\frac{\text{hodnota adc} * \text{referenčné napätie}}{\text{rozlíšenie prevodníka}}$ and added to the variable *voltage*. This cycle runs ten times, and then the voltage variable is divided by ten to obtain the average value of the measured voltage. Furthermore, it is necessary to determine whether the measured voltage is negative or not. For this is used the boolean *read_comparator()* function. In this function, the pin voltage is read by the internal 10-bit ADC. If the measured voltage is greater than 0.25V, the function returns logic 1 and if it is lower returns a logic 0. If the function returns log. 0, the voltage value of the variable “*voltage*” is subtracted from 0, thereby becoming negative. Then, calibration constants are applied to the current range.

Sending data

Sending data to the other microcontroller also takes place on the I2C line 5 times per second. The transmitted data carries the voltage value, the polarity of the measured voltage, and the current measuring range. The *send_voltage()* function sends 3 bytes of data. The first is the so-called *statusByte* providing information about voltage polarity and measurement range. The second is the value before the decimal point and the third is the value after the decimal point. The description of individual bits of *statusByte* byte is in Attachment no. 5. After the byte values are calculated, they are sent to the microcontroller of output peripherals with address 8.

Main program

Once the device is turned on, 1s wait is executed and then the program starts. This waiting is important so that the microcontroller of the measuring part starts the communication via the I2C interface after initializing the second microcontroller as a slave. Without this waiting, there were various problems with communication between microcontrollers. Microcontroller inputs and outputs and I2C communication are initialized. The sequence of the program is as follows. First, the *range_update()* function is called, then *read_adc()*, and if the time from program execution is greater than or equal to “*timeToSend*” variable, the *send_voltage()* function is called and 200ms is added to the “*timeToSend*”. There is no waiting in the main program, which provides an immediate response to the push of a button, making the range switching reliable.

Mechanical part

The voltmeter is built in a universal plastic box with dimensions of 150 x 70 x 180mm. It is suitable for use in the laboratory as well as outdoors. Voltmeter view is (Figure 15) in attachment. Printed circuit boards are placed in the center of the box and are fixed in place by three screws (Figure 16). 15mm distance spacers are inserted between the measuring PCB and the Arduino Uno. (Figure 17) Output peripherals and controls are located on the front panel of the box. The front panel design was in the program EAGLE, on which a 1:1

scale was printed on the paper. The individual holes were marked, drilled and well-filed. All parts are fastened with nuts except LCD and speaker, which is fixed in place by hot-melt glue. The speaker is located on the front of the side wall where several holes are drilled for good sound propagation.

The front panel description is printed on sticky paper that is stuck to the panel (Figure 18). The finish is matt lacquer protecting the paper against wetting and rubbing. Anti-slip rubber feet are also glued on the bottom side for stability when pressing buttons. Front view of device shows (Figure 19) in attachment no. 1.

Testing

Voltmeter testing with voice output took place throughout the development of the device. The individual segments were first connected to the contact field, where their development took place, and then prototype circuit boards containing individual segments were constructed. These units have been measured and refined several times to achieve the best results. The most critical part of the voltmeter is the converter to absolute value. We had multiple OPA2277 operational amplifiers, so this circuit was measured with each piece. The operational amplifier at which the deviation was the smallest was subsequently used in the final version as well.

After successful completion of work on this device, it was necessary to calibrate the voltmeter with the voice output. Calibration was done by writing the calibration constants to the microcontroller program of the measurement part. An accurate 5-digit UNI-T UT71E multimeter was used for measurements. The reference voltage of the ADC was measured and then the voltmeter and the precision multimeter were compared. After writing the calculated calibration constants for both ranges, the measurement uncertainty was about 5% on the 0-4V range and 7% on the 0-40V range. To increase accuracy, a program for calculating the voltage has been modified.

After calibration, the accuracy of the voltmeter was compared to a precision UNI-T UT71E multimeter. All ranges from minimum to maximum were tested. The test circuit contained our voltmeter connected in parallel with the precision multimeter, which simultaneously measured the voltage on two power supplies connected in series to reach 40V. The expected deviation of the measurement was about 1% in both ranges. The measured values are in (Table 1) on page 31 in the Attachments chapter. The measured deviation was maximum 0.01V. At voltages greater than 1V, the deviation is between 0.02% and 1% at both ranges. At voltages below 1V, the measurement uncertainty at 0-4V ranges from 1% to 5%.

The rest of the device also went through testing: automatic range, range and language switching, volume control, voice output in all languages, and displaying information on LCD. The automatic range works reliably, switching ranges and language is fully functional without bugs.

Conclusion

This project impressed us right from the start because it is a very specific device. The implementation of a voltmeter with voice output required knowledge of both electronics (measurement) and programming (voice output). Two of the different countries worked on the project, each on their part, which tested our communication skills. We have been working on our own parts, testing and improving them. During the first week of the internship, we tested the communication between our prototypes and the overall functionality of the device. We produced the final printed circuit boards that we tested during the second week, put them in a box, and created a measuring instrument suitable for both the laboratory and the home. We took care of intuitive operation and ease of use. The use of the measuring instrument is wide and will find its application, for example, in the household of the visually impaired people to measure batteries and similar safe measurements.

Matej Grega:

During this project, I learned a lot of new information about electronics, especially about operational amplifiers and microcontrollers. The implementation of the measuring part verified my skills in designing and manufacturing printed circuit boards. International communication was in English, so we developed our language skills. Collaboration has not been smooth, but we have solved all the misunderstandings in two working weeks when we were working side by side on the device. I appreciate the implementation of the project during the weekly internships, but also the time to work was limited, which led to more intensive work at home and internet communication.

I consider the whole project to be successful because the equipment is fully functional and the collaboration has been without major problems. I intend to improve my voice output voltmeter a little more for laboratory use. This project also inspired me to think about devices for the visually impaired people. I plan to create other devices for the blind and partially sighted.

Mate Budai:

During the making of this project I learnt new things, my partner knows what to do and it was very good working with him. We communicated in English with minor problems but we understood each other which was the goal. These 2 weeks that we spent in a classroom in Hungary and Slovakia wasn't much, however, it was very efficient, since my partner was hardworking and we tried to solve things together if one of us didn't understand something. I spent one week in Slovakia and came back with new experiences. The project wasn't just a task for me but also a journey because I had the chance to talk with foreign people and made new friends. Our project was a success and it's working. I hope I will have more opportunities like this one in the future.

Reference

1. Diplomová práce, Eduard Cabuk, Prevodníky A/D a D/A realizované technikou SI, Žilinská univerzita v Žiline, Elektrotechnická fakulta, 2007
2. <http://ww1.microchip.com/downloads/en/devicedoc/20001732e.pdf>
3. https://www.analog.com/media/en/technical-documentation/data-sheets/ad1582_1583_1584_1585.pdf
4. Konstrukční elektronika A Rádio, 3/1996, ISSN 1211-3557
5. <http://www.ti.com/lit/ds/symlink/opa2277-ep.pdf>
6. <https://pdf1.alldatasheet.com/datasheet-pdf/view/1005385/HDSEMI/BAW99.html>
7. <https://www.onsemi.com/pub/Collateral/LM339-D.PDF>
8. <http://www.ti.com/lit/ds/symlink/cd4052b.pdf>
9. <http://pdf1.alldatasheet.com/datasheet-pdf/view/67437/INTERSIL/ICL7660S.html>
10. <https://www.arduino.cc/en/Main/Products>
11. <https://www.robot-r-us.com/vmchk/sound-voice/dfplayer-mini-mp3-player-module-for-arduino.html>

Attachments

Attachment no. 1: Images of device

Attachment no. 2: Schematics of device

Attachment no. 3: Simulation of absolute value converter

Attachment no. 4: Calibration

Attachment no. 5: Definition of information byte "statusByte"

Attachment no. 6: Program codes

Attachment no. 1

Images of device



Figure 15 - View of the device.

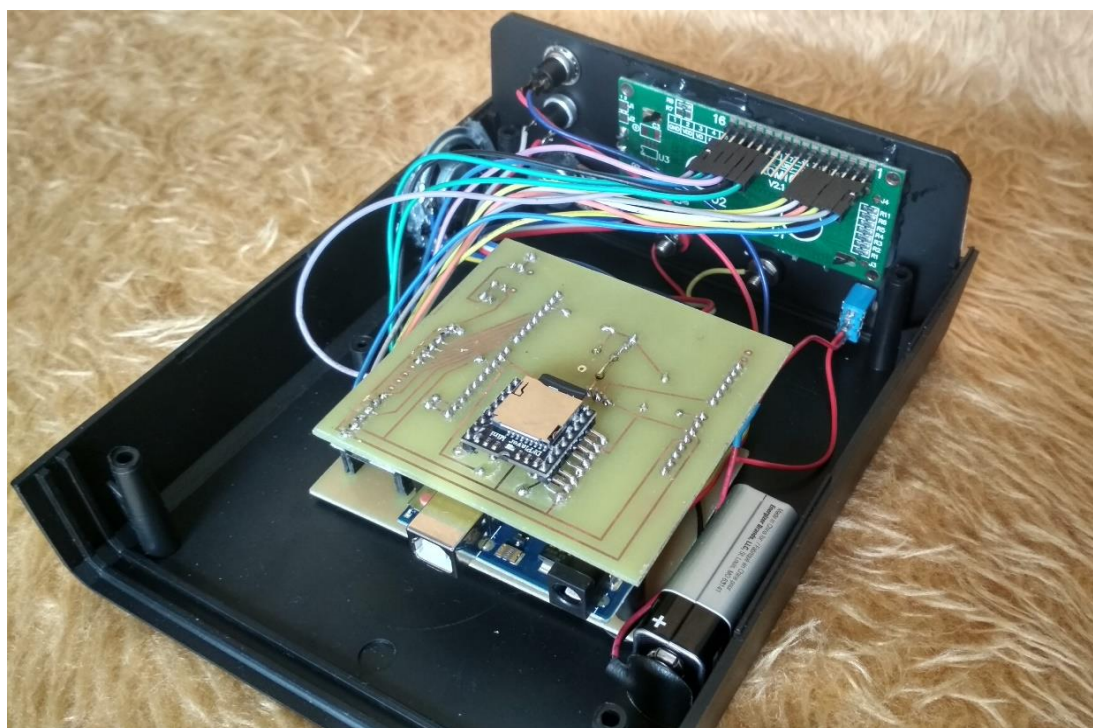


Figure 16 - Internal wiring of voltage meter.

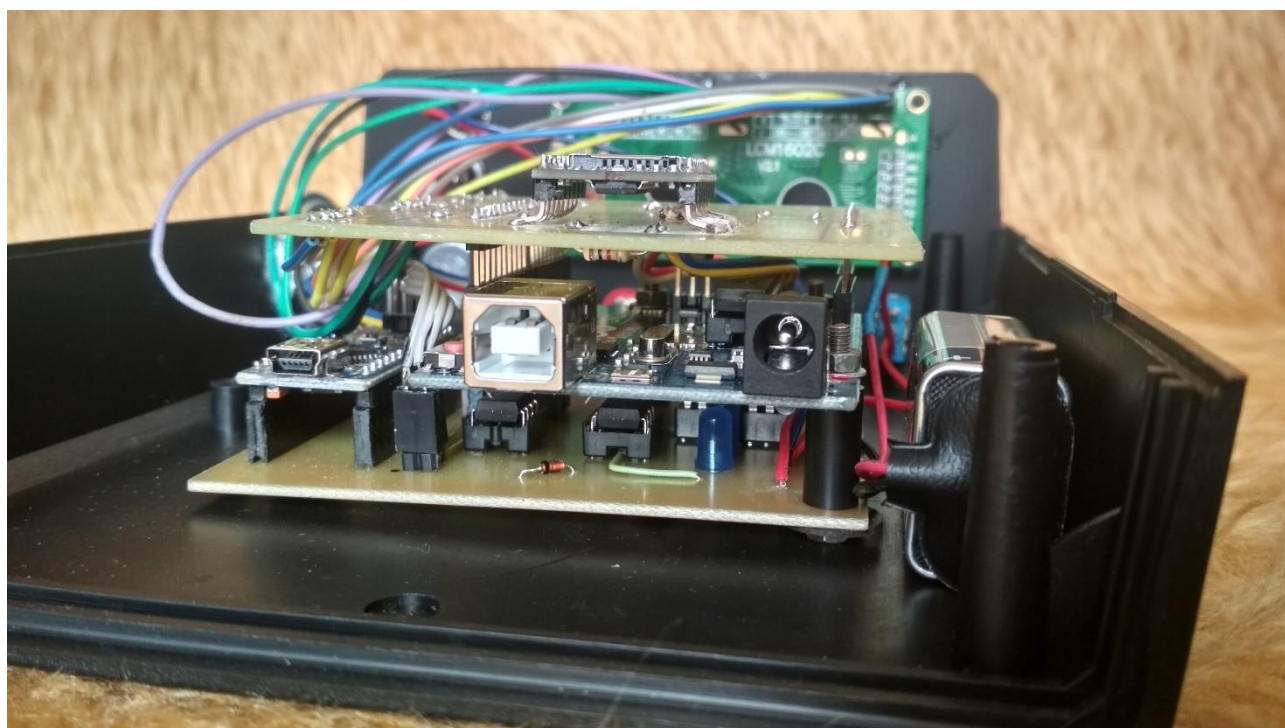


Figure 17 -Back view.

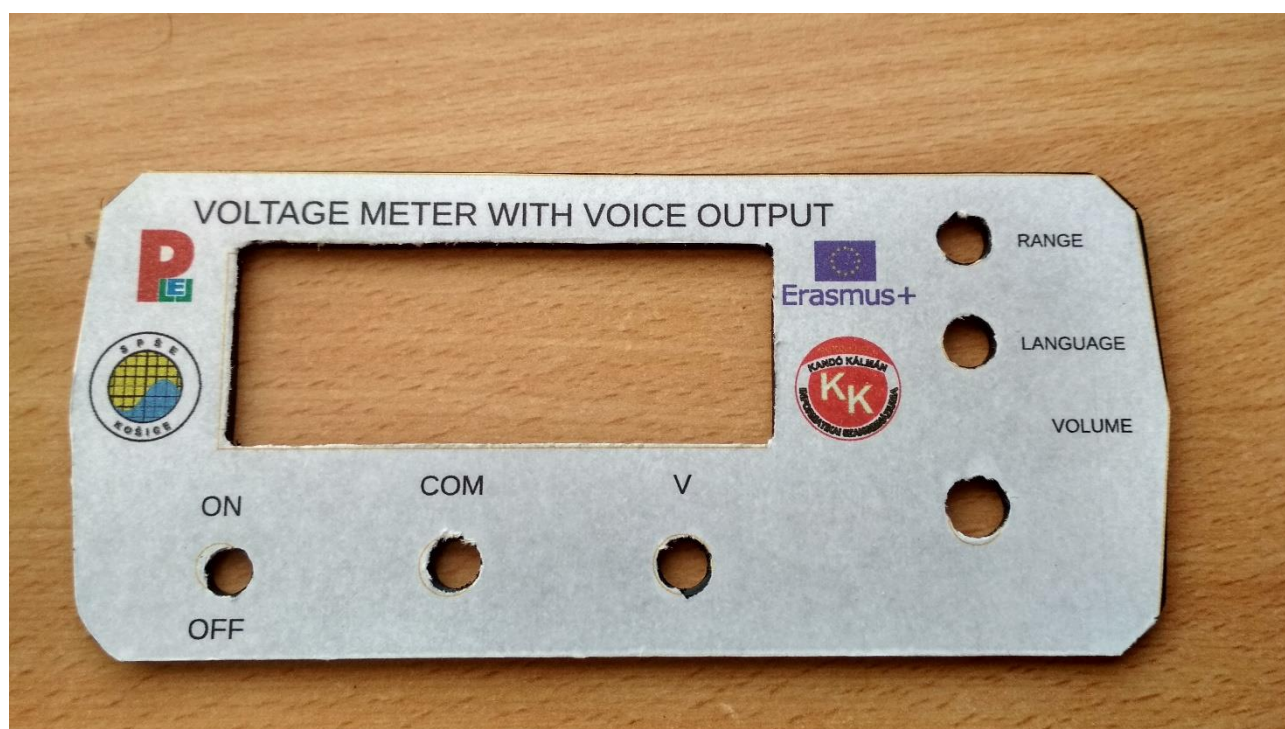


Figure 18 -Front panel.



Figure 19 - Front view of finished device.

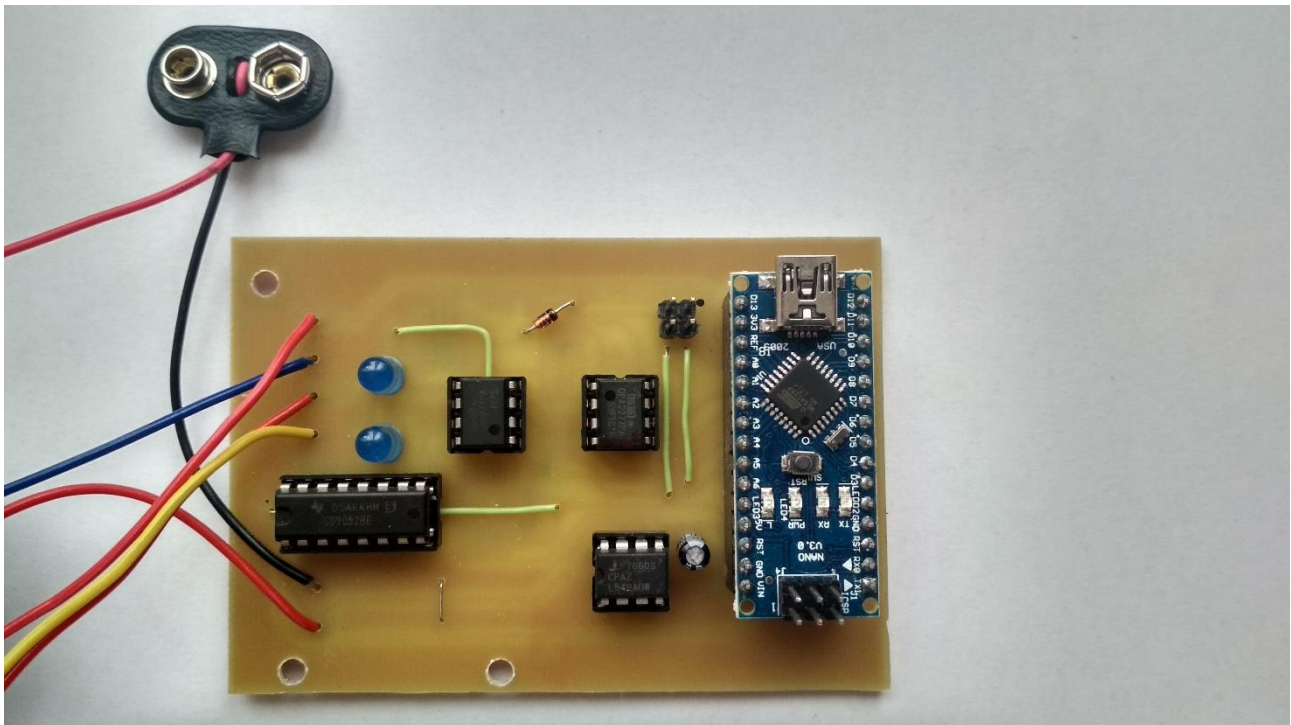


Figure 20 - Printed circuit board, measuring part, component side.

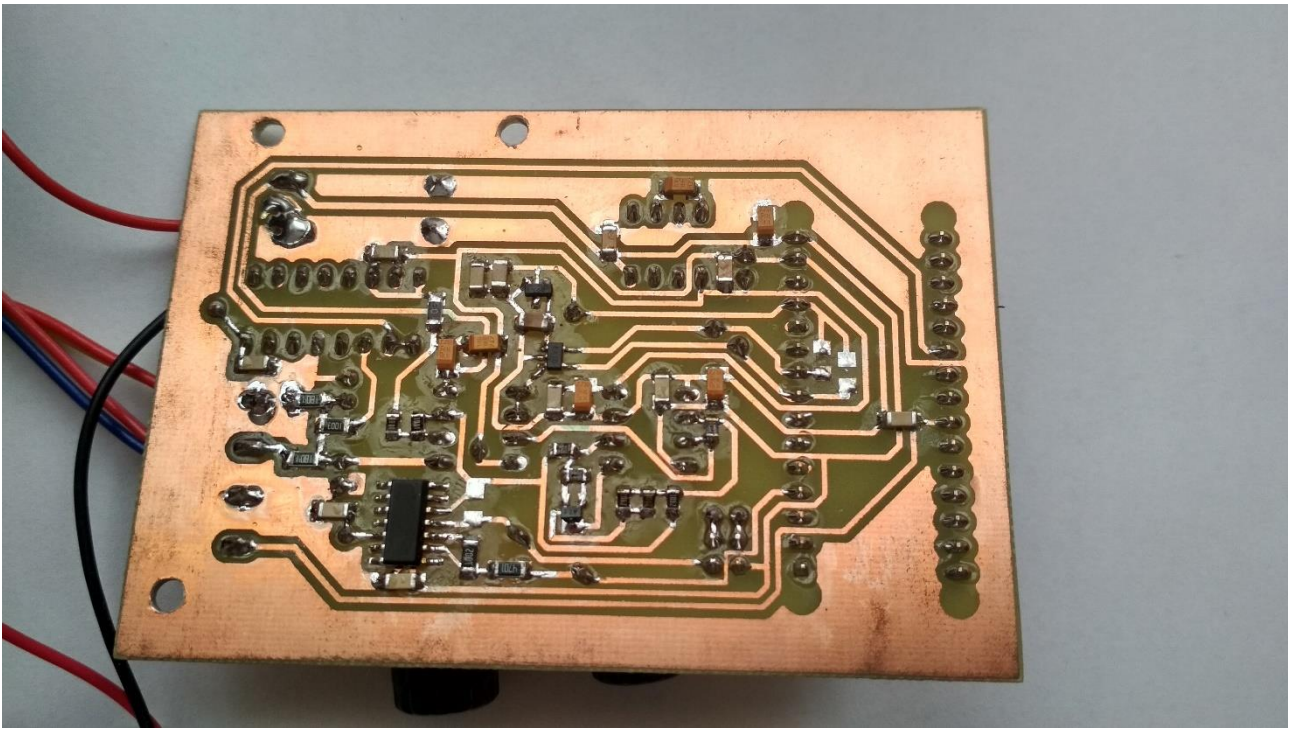


Figure 21 - Printed circuit board, measuring part, connections side.

Attachment no. 2

Schematics of device

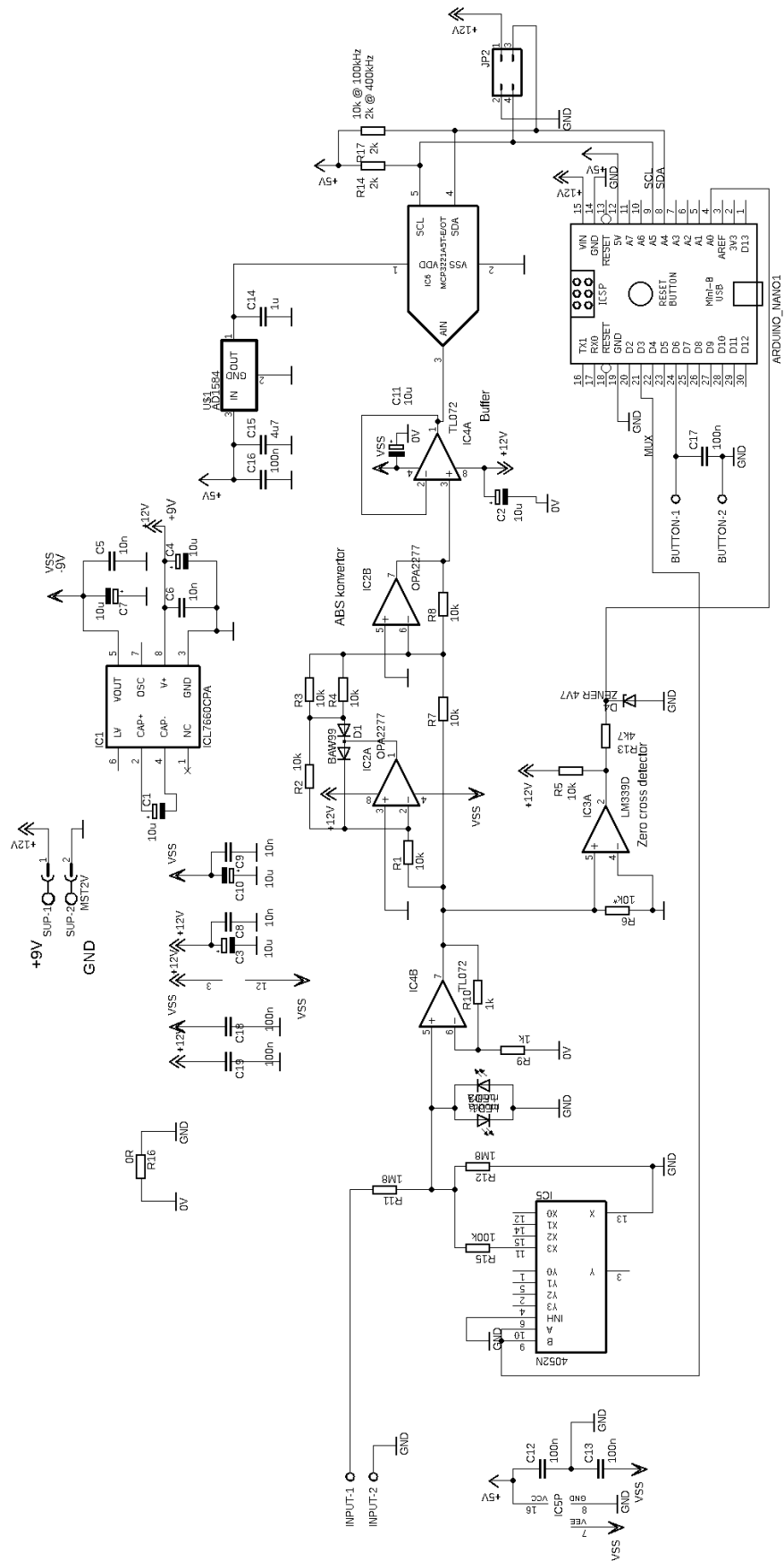


Figure 22 - Schematic of measuring part of voltage meter.

Attachment no. 3

Simulation of absolute value converter

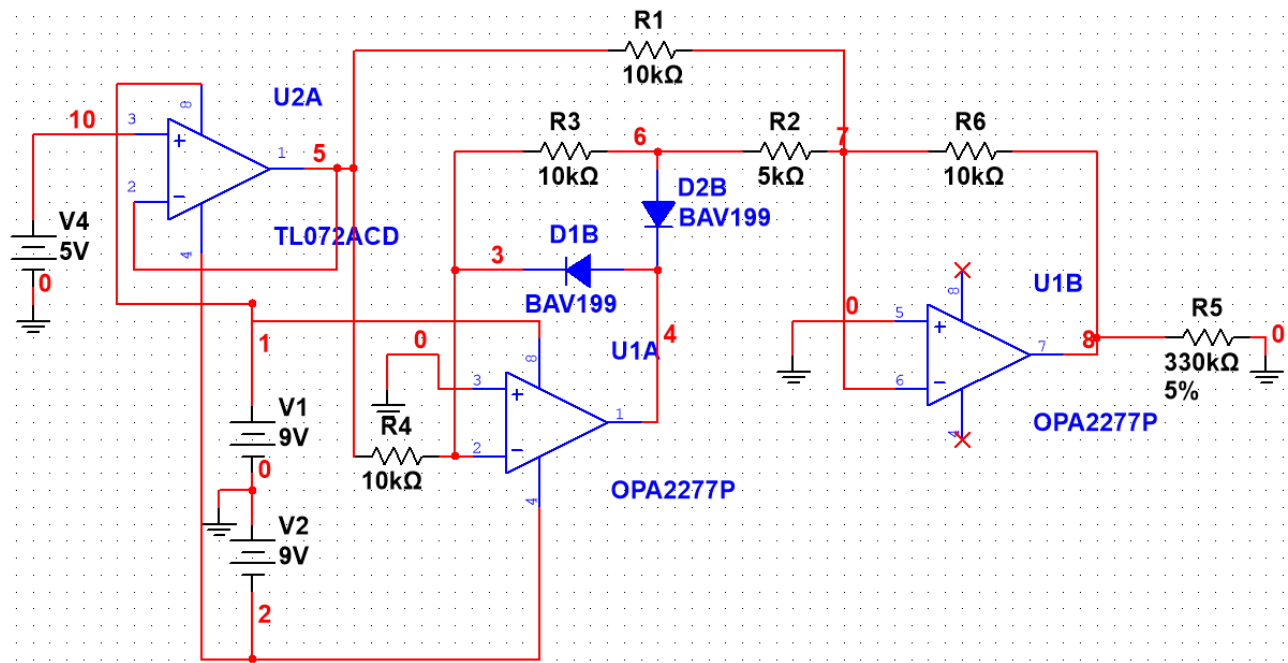


Figure 23 - Schematic of simulated circuit of absolute value converter.

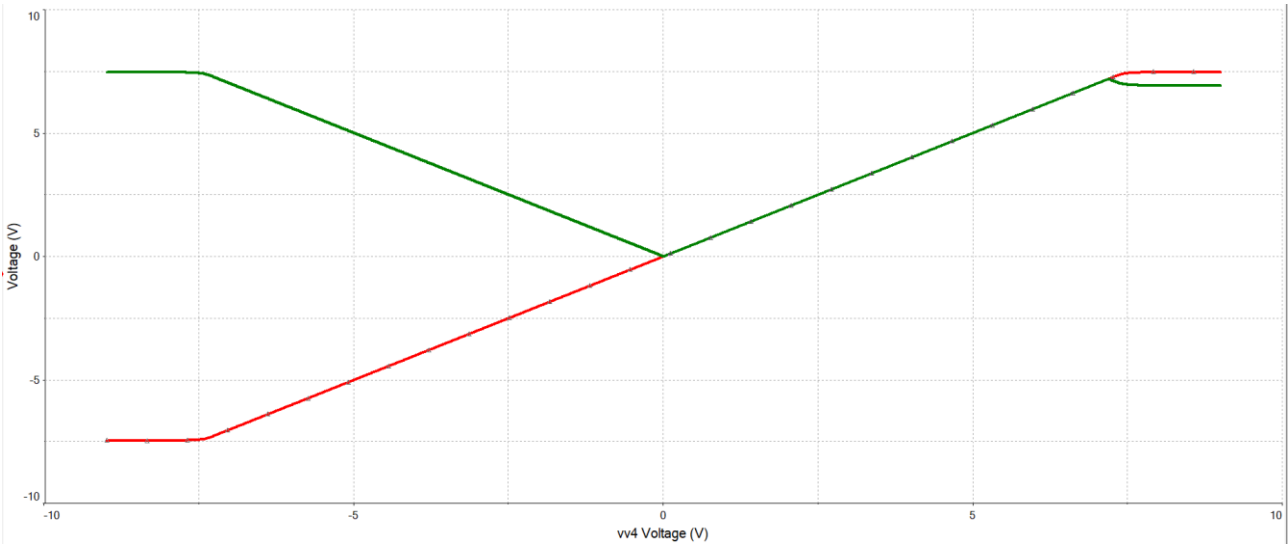


Figure 24 - Graph of simulated converter, output voltage vs input voltage.

Attachment no. 4

Calibration

Č.M.	rozsah 0-40V		
	UT71E	naš voltmeter	odchýlka [%]
1	0,0000	0,00	0,000
2	0,9996	1,01	-1,040
3	2,0008	2,01	-0,460
4	3,0007	3,01	-0,310
5	4,006	4,01	-0,100
6	5,003	5,01	-0,140
7	5,998	6,01	-0,200
8	7,004	7,02	-0,228
9	8,006	8,01	-0,050
10	9,000	9,02	-0,222
11	10,000	10,01	-0,100
12	11,008	11,01	-0,018
13	12,009	12,02	-0,092
14	13,001	13,00	0,008
15	13,999	14,00	-0,007
16	15,004	15,00	0,027
17	16,002	16,00	0,012
18	17,002	17,00	0,012
19	18,001	18,00	0,006
20	19,003	19,00	0,016
21	20,001	20,00	0,005
22	21,003	21,00	0,014
23	22,002	22,00	0,009
24	23,004	23,00	0,017
25	24,000	24,00	0,000
26	25,003	25,00	0,012
27	26,005	26,00	0,019
28	27,007	27,00	0,026
29	27,998	28,00	-0,007
30	29,001	29,01	-0,031
31	30,002	30,01	-0,027
32	31,007	31,01	-0,010
33	32,002	32,01	-0,025
34	33,004	33,01	-0,018
35	34,003	34,01	-0,021
36	35,004	35,01	-0,017
37	36,000	36,01	-0,028
38	37,001	37,01	-0,024
39	38,002	38,01	-0,021
40	39,002	39,01	-0,021
41	40,00	40,05	-0,125

Č.M.	rozsah 0-4V		
	UT71E	naš voltmeter	odchýlka [%]
1	0,0000	0,00	0,000
2	0,1005	0,09	10,448
3	0,2012	0,19	5,567
4	0,3036	0,29	4,480
5	0,4025	0,39	3,106
6	0,5030	0,49	2,584
7	0,6005	0,59	1,749
8	0,7004	0,69	1,485
9	0,8036	0,79	1,692
10	0,9038	0,89	1,527
11	1,0012	0,99	1,119
12	1,2504	1,24	0,832
13	1,5000	1,49	0,667
14	1,7515	1,74	0,657
15	2,0018	2,00	0,090
16	2,2506	2,25	0,027
17	2,5056	2,50	0,223
18	2,7532	2,75	0,116
19	3,0026	3,00	0,087
20	3,2531	3,25	0,095
21	3,5026	3,50	0,074
22	3,7502	3,75	0,005
23	4,003	4,00	0,075

Table 1 - Accuracy of voltage meter compared with UNI-T UT71E multimeter.

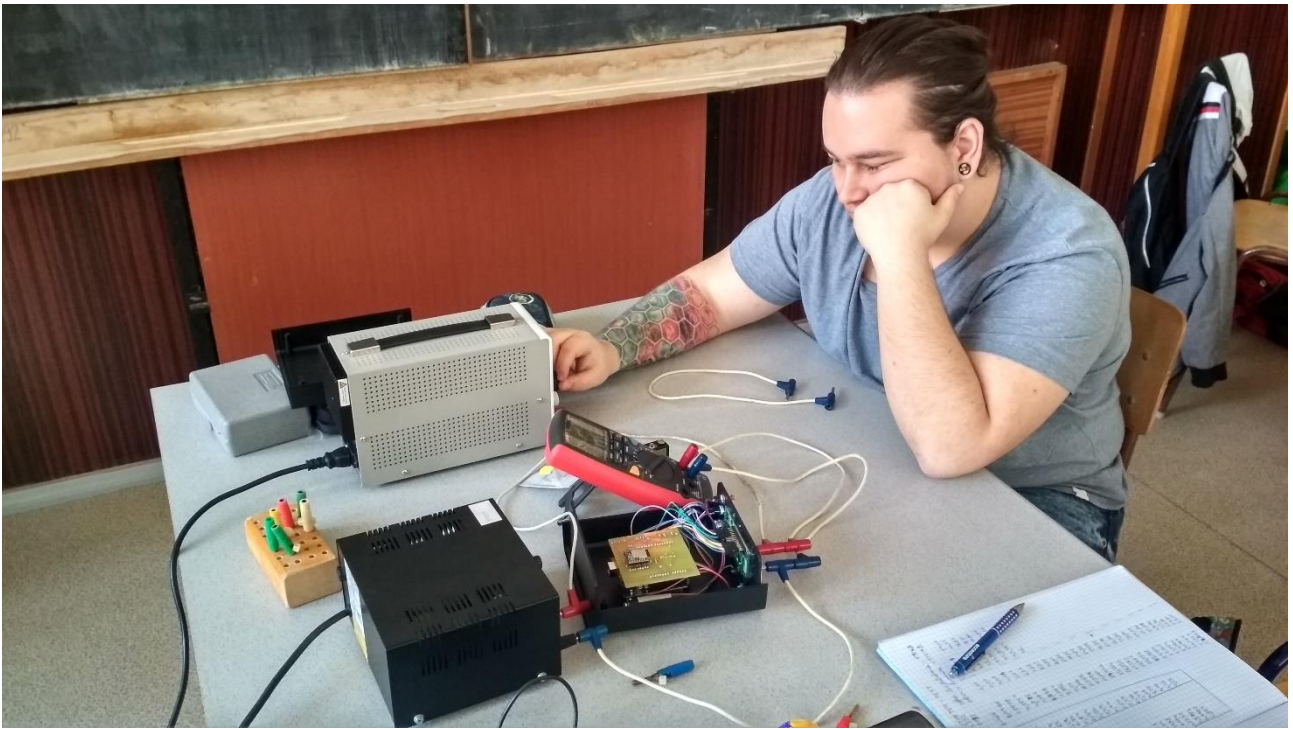


Figure 25 - Process of measuring accuracy with multimeter UNI-T UT71E.

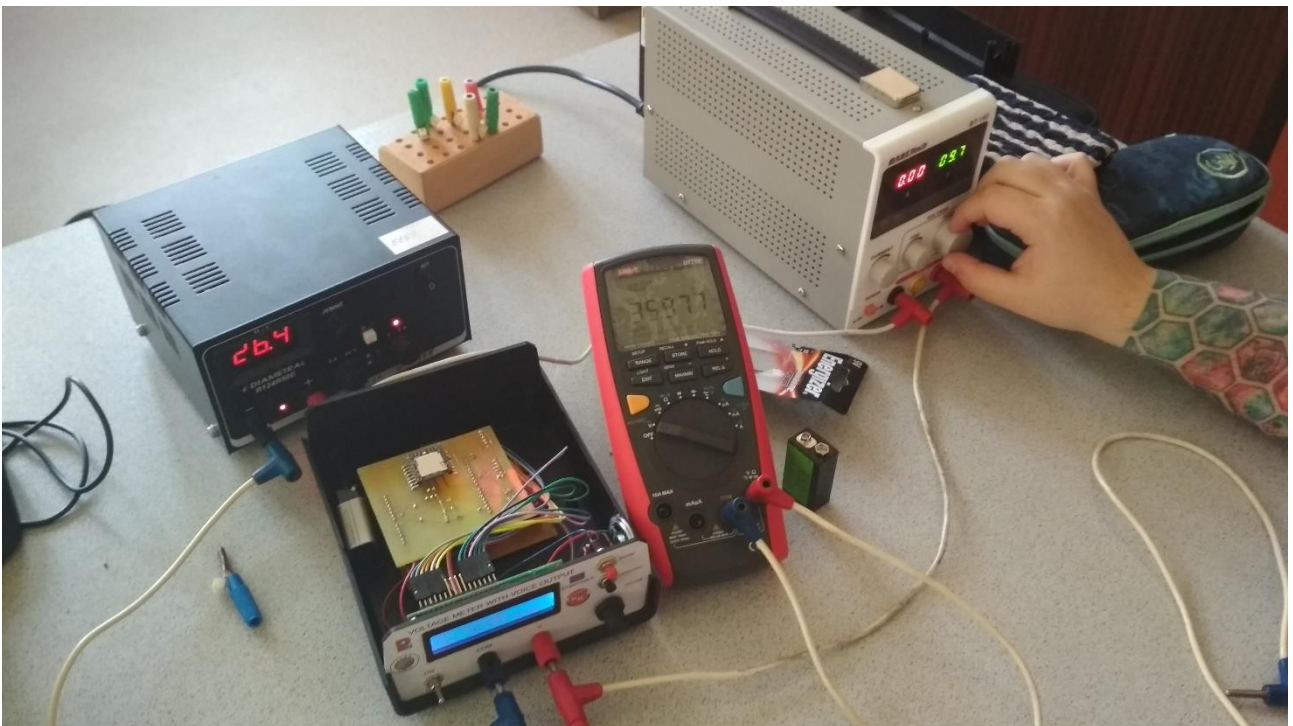


Figure 26 - Measuring accuracy of voltage meter with voice output.

Attachment no. 5

Definition of information byte “statusByte”

Table 2 shows meaning and function of each bit in byte “statusByte”.

minus	minus	minus	minus	-	auto range	measuring 0-40V	measuring 0-4V
-------	-------	-------	-------	---	------------	-----------------	----------------

Table 2 - Detail of bits of byte statusByte.

Examples:

- Positive voltage, manual range 0-4V: 11110001
- Negative voltage, manual range 0-4V: 00000001
- Positive voltage, manual range 0-40V: 11110010
- Negative voltage lower than 4V, automatic range: 00000101
- Positive voltage higher than 4V, automatic range: 11110110

Attachment no. 6

Program codes

Measuring part

```
#include <Wire.h>
#define comparator_input_pin A1          //pin for reading comparator (positive or negative measured voltage)
#define range_output_pin 3              //output pin for switching ranges
#define button_pin 6                    //pin connected to button
#define output_address 8                //address of the second arduino in I2C
#define button_time_delay 100           //time in milliseconds in which arduino doesn't read the button after
the first press
#define TimeToSendConstant 200         //delay between two sendings of voltage data via I2C to the second
arduino
#define number_measurements 10          //number of adc measurements to one voltage value
#define adc_resolution 4096            //resolution of ADC
#define reference_voltage 4.088         //value of voltage reference for adc
#define gain4V 1.005                   //gain in range 0-4V
#define offset4V 0.0                   //offset in range 0-4V
#define gain40V 10.05295               //gain in range 0-40V measured @ 40V
#define offset40V 0.0                  //offset in range 0-40V
#define auto_range_to_down_voltage 3.9 //if voltage is lower than this constant auto range select range 0-4V
#define auto_range_to_up_voltage 4.0   //if voltage is higher than this constant auto range select range 0-4V

boolean button_pressed = 0;
float voltage;
byte range = 0;                //0-40V = 0; 0-4V = 1; auto = 2;
boolean range_output;          //0-40V = 0; 0-4V = 1;
long timeToSend = 0;
long button_time = 0;

/*-----*/
//Function for reading ADC more times and computing the value of voltage.

void read_adc(){
    float calibration_variable;
    voltage = 0;
    for(byte i = 1; i <= number_measurements; i++){
        float adc;
        Wire.requestFrom(B1001101, 2); //requests value (two bytes) from ADC via I2C
        while(Wire.available() < 2){}; //after receiving them
        adc = (Wire.read() << 8);      //read them
        adc += Wire.read();
        voltage += (adc * reference_voltage) / adc_resolution; //add current value of measured voltage
        delay(2);
    }
    voltage /= number_measurements;    //compute the average voltage
    if(read_comparator() == 0){        //If measured voltage is negative
        voltage = 0 - voltage;        //do it negative.
    }
    if(range_output){                  //apply calibration constants and variables
        voltage = (voltage * gain40V) + offset40V;
        if(voltage > 0.0){
            calibration_variable = map(voltage, 0, 40, 97, 0);
            voltage += calibration_variable / 1000;
        }
        else if(voltage < 0.0){
            calibration_variable = map(voltage, -40, 0, 0, 75);
            voltage -= calibration_variable / 1000;
        }
    }
    else{
        voltage = (voltage * gain4V) + offset4V;
    }
}

/*-----*/
//return 0 if measured voltage is negative, return 1 if positive

boolean read_comparator(){
    boolean comparator_positive;
    if(analogRead(comparator_input_pin) > 50){
        comparator_positive = 1;
    }
    else{
```

```

    comparator_positive = 0;
}
return comparator_positive;
}
/*-----*/
//Function for reading button and changing ranges 0-40, 0-4, auto

void read_button(){
    if(((digitalRead(button_pin) == 0) && (button_pressed == 0)) && (millis() >= button_time)){
        button_pressed = 1;
        button_time = millis() + button_time_delay;
        switch(range){
            case 0:
                range = 1;
                break;

            case 1:
                range = 2;
                break;

            case 2:
                range = 0;
        }
    }
    if((digitalRead(button_pin) && button_pressed) && (millis() >= button_time)){
        button_pressed = 0;
        button_time = millis() + button_time_delay;
    }
}
/*-----*/

void range_update(){
    read_button();
    if(range <= 1){                                     //if manual range is selected
        range_output = !range;
    }
    else{                                                //if automatic range is selected
        if((abs(voltage) > auto_range_to_up_voltage) && (range_output == 0)){ //and voltage is higher than
            auto_range_to_up_voltage
            range_output = 1;                             //set range 0-40V
        }
        if((abs(voltage) < auto_range_to_down_voltage) && range_output){ //if voltage is lower than
            auto_range_to_down_voltage
            range_output = 0;                             //set range 0-4V
        }
    }
    digitalWrite(range_output_pin, range_output);      //update range
}

/*-----*/
//Function for sending the value of measured voltage and status byte

void send_voltage(){
    byte statusByte = 0;
    byte voltageByte = 0;
    byte decimalByte = 0;
    float abs_voltage = abs(voltage);
    if(range_output){                                   //if voltage range is 0-40V
        statusByte = B00000001;                       //set range in status byte
    }
    else{                                                //if voltage range is 0-4V
        statusByte = B00000010;                       //set range in status byte
    }
    if(range == 2){                                     //if voltage range is auto
        statusByte |= B00000100;                       //set range in status byte
    }
    if(voltage >= 0){                                   //if the measured voltage is positive
        statusByte |= 0xf0;                             //set sign in status byte
    }
    else{                                                //if the measured voltage is negative
        statusByte &= ~(0xf0);                         //clear sign in status byte
    }
    voltageByte = byte(abs_voltage);                    //voltagebyte is absolute value of measured voltage
    before the decimal point
    decimalByte = byte((abs_voltage - voltageByte) * 100); //decimalbyte is absolute value of measured voltage
    after the decimal point

```

```

    Wire.beginTransmission(output_address);                //start of I2C communicatoin withe the second
arduino
    Wire.write(statusByte);                                //send status byte, voltage byte and decimal byte
    Wire.write(voltageByte);
    Wire.write(decimalByte);
    Wire.endTransmission();                                //end of I2C communication
}
/*-----*/

void setup() {
    delay(1000);
    Wire.begin();
    Wire.setClock(400000);
    pinMode(comparator_input_pin, INPUT);
    pinMode(range_output_pin, OUTPUT);
    pinMode(button_pin, INPUT_PULLUP);
}

void loop(){
    range_update();
    read_adc();
    if(millis() >= timeToSend){
        send_voltage();
        timeToSend = millis() + TimeToSendConstant;
    }
}

```


Part of output peripherals

```
#include <Wire.h>
#include "Arduino.h"
#include "SoftwareSerial.h"
#include <DFRobotDFPlayerMini.h>
#include <LiquidCrystal.h>

#define interrupt_time_constant 250

LiquidCrystal lcd(8, 7, 6, 5, 4, 3);
SoftwareSerial mySoftwareSerial(9, 10); // RX, TX
DFRobotDFPlayerMini myDFPlayer;

int language=-1;//0 EN, 1 HU, 2 SK
const byte interruptPinL = 2;
byte statusbyte, voltagebyte, decimalbyte;
long time_to_say = 0;
long interrupt_time = 0;
int vol;

byte speaker[8]={
  B00001,
  B00011,
  B01101,
  B01001,
  B01101,
  B00011,
  B00001,
  B00000
};
byte speakervolume1[8]={
  B00000,
  B00000,
  B00000,
  B10000,
  B00000,
  B00000,
  B00000,
  B00000
};
byte speakervolume2[8]={
  B00000,
  B01000,
  B00100,
  B10100,
  B00100,
  B01000,
  B00000,
  B00000
};
byte speakervolume3[8]={
  B00010,
  B01001,
  B00101,
  B10101,
  B00101,
  B01001,
  B00010,
  B00000
};

void isrL() {
  if(interrupt_time <= millis()){
    language++;
    interrupt_time = millis() + interrupt_time_constant;
  }
  if(language>2){
    language=0;
  }
  lcd.setCursor(12, 1);
  if(language==0){
    lcd.clear();
    lcd.print("EN");
    lcd.setCursor(3, 0);
    lcd.print("Voltage");
  }
  if(language==1){
```

```

    lcd.clear();
    lcd.print("SK");
    lcd.setCursor(3, 0);
    lcd.print("Napatie");
}
if(language==2){
    lcd.clear();
    lcd.print("HU");
    lcd.setCursor(3, 0);
    lcd.print("Feszultseg");
}
}

void volume_update(){
    vol = map(analogRead(A0),0,1024,0,30);
    myDFPlayer.volume(vol);
    lcd.setCursor(0, 1);
    lcd.write(byte(0));
    if(vol == 0){
        lcd.print("x");
    }
    if((vol<10) && (vol>0)){
        lcd.write(byte(1));
    }
    if((vol<20) && (vol>10)){
        lcd.write(byte(2));
    }
    if((vol<31) && (vol>20)){
        lcd.write(byte(3));
    }
}

void lcd_update(){
    lcd.setCursor(3, 1);
    if(statusbyte & B11110000){
        lcd.print(" ");
    }
    else{
        lcd.print("-");
    }
    lcd.print(voltagebyte);
    lcd.print(".");
    if ( decimalbyte < 10){
        lcd.print("0");
    }
    lcd.print(decimalbyte);
    if(voltagebyte < 10){
        lcd.setCursor(8,1);
    }
    else{
        lcd.setCursor(9,1);
    }
    lcd.print("V ");
    lcd.setCursor(11, 1);
    if (statusbyte & B00000100) {
        if(statusbyte & B00000001){
            lcd.print("auto2");
        }
        if(statusbyte & B00000010){
            lcd.print("auto1");
        }
    }
    else{
        if (statusbyte & B00000001){
            lcd.print("0-40 ");
        }
        else if(statusbyte & B00000010){
            lcd.print("0-4 ");
        }
    }
}

void sayNumber(int n, int offset, int b) {
    if (n==0) {
        myDFPlayer.play(offset+2);//null
        delay(800);
    } else {

```

```

if (n>=1000) {
    int thousands = n / 1000;
    sayNumber(thousands,offset,b);
    myDFPlayer.play(offset+34);delay(500);
    n %= 1000;
    if ((n > 0) && (n<100)) {myDFPlayer.play(offset+31);delay(500);};//and
}
if (n>=100) {
    int hundreds = n / 100;
    sayNumber(hundreds,offset,b);
    myDFPlayer.play(offset+30);delay(500);
    n %= 100;
    if (n > 0) {myDFPlayer.play(offset+31);delay(500);};//and
}
if (n>19) {
    int tens = n / 10;
    b=0;
    switch (tens) {
        case 2:
if(offset==64&&n==20){myDFPlayer.play(offset+13);delay(700);break;}myDFPlayer.play(offset+22);delay(600);
break;
        case 3: if(offset==64){myDFPlayer.play(offset+14);delay(700);} else
{myDFPlayer.play(offset+23);delay(700);} break;
        case 4: if(offset==64){myDFPlayer.play(offset+15);delay(700);} else
if(offset==34){myDFPlayer.play(offset+24);delay(900);} else {myDFPlayer.play(offset+24);delay(500);} break;
        case 5: if(offset==64){myDFPlayer.play(offset+16);delay(700);} else
if(offset==34){myDFPlayer.play(offset+25);delay(900);} else {myDFPlayer.play(offset+25);delay(500);} break;
        case 6: if(offset==64){myDFPlayer.play(offset+17);delay(700);} else
if(offset==34){myDFPlayer.play(offset+26);delay(1000);} else {myDFPlayer.play(offset+26);delay(600);} break;
        case 7: if(offset==64){myDFPlayer.play(offset+18);delay(700);} else
if(offset==34){myDFPlayer.play(offset+27);delay(1000);} else {myDFPlayer.play(offset+27);delay(700);} break;
        case 8: if(offset==64){myDFPlayer.play(offset+19);delay(700);} else
if(offset==34){myDFPlayer.play(offset+28);delay(900);} else {myDFPlayer.play(offset+28);delay(450);} break;
        case 9: if(offset==64){myDFPlayer.play(offset+20);delay(800);} else
if(offset==34){myDFPlayer.play(offset+29);delay(1000);} else {myDFPlayer.play(offset+29);delay(500);} break;
    }
    n %= 10;
}
switch(n) {
    case 1: if(b==1){ myDFPlayer.play(offset+2);delay(600);} myDFPlayer.play(offset+3);delay(600); break;
    case 2: if(b==1){ myDFPlayer.play(offset+2);delay(600);} myDFPlayer.play(offset+4);delay(500); break;
    case 3: if(b==1){ myDFPlayer.play(offset+2);delay(600);} myDFPlayer.play(offset+5);delay(500); break;
    case 4: if(b==1){ myDFPlayer.play(offset+2);delay(600);} myDFPlayer.play(offset+6);delay(500); break;
    case 5: if(b==1){ myDFPlayer.play(offset+2);delay(600);} myDFPlayer.play(offset+7);delay(500); break;
    case 6: if(b==1){ myDFPlayer.play(offset+2);delay(600);} myDFPlayer.play(offset+8);delay(500); break;
    case 7: if(b==1){ myDFPlayer.play(offset+2);delay(600);} myDFPlayer.play(offset+9);delay(800); break;
    case 8: if(b==1){ myDFPlayer.play(offset+2);delay(600);} myDFPlayer.play(offset+10);delay(500); break;
    case 9: if(b==1){ myDFPlayer.play(offset+2);delay(600);} myDFPlayer.play(offset+11);delay(500); break;
    case 10: myDFPlayer.play(offset+12);delay(800); break;
    case 11: if(offset==64){myDFPlayer.play(offset+24);delay(500);myDFPlayer.play(offset+3);delay(500);}
else {myDFPlayer.play(offset+13);delay(900);} break;
    case 12: if(offset==64){myDFPlayer.play(offset+24);delay(500);myDFPlayer.play(offset+4);delay(500);}
else {myDFPlayer.play(offset+14);delay(900);} break;
    case 13: if(offset==64){myDFPlayer.play(offset+24);delay(500);myDFPlayer.play(offset+5);delay(500);}
else {myDFPlayer.play(offset+15);delay(900);} break;
    case 14: if(offset==64){myDFPlayer.play(offset+24);delay(500);myDFPlayer.play(offset+6);delay(500);}
else {myDFPlayer.play(offset+16);delay(900);}break;
    case 15: if(offset==64){myDFPlayer.play(offset+24);delay(500);myDFPlayer.play(offset+7);delay(500);}
else {myDFPlayer.play(offset+17);delay(900);} break;
    case 16: if(offset==64){myDFPlayer.play(offset+24);delay(500);myDFPlayer.play(offset+8);delay(500);}
else {myDFPlayer.play(offset+18);delay(900);} break;
    case 17: if(offset==64){myDFPlayer.play(offset+24);delay(500);myDFPlayer.play(offset+9);delay(500);}
else { myDFPlayer.play(offset+19);delay(900);} break;
    case 18: if(offset==64){myDFPlayer.play(offset+24);delay(500);myDFPlayer.play(offset+10);delay(500);}
else { myDFPlayer.play(offset+20);delay(900);} break;
    case 19: if(offset==64){myDFPlayer.play(offset+24);delay(500);myDFPlayer.play(offset+11);delay(500);}
else { myDFPlayer.play(offset+21);delay(900);} break;
}
}
}

void say_voltage(byte statusbyte_say, byte voltagebyte_say, byte decimalbyte_say){
    int a,b=0;
    if(language==0)a=0; //English
    if(language==1)a=34; //Slovak
    if(language==2)a=64; //Hungarian
    if((statusbyte_say & B11110000) == false){

```

```

    switch(a){
    case 0:myDFPlayer.play(32);delay(800);break;//minus
    case 34:myDFPlayer.play(90);delay(1000);break;//minus
    case 64:myDFPlayer.play(a+26);delay(1000);break;//minus
    }
}
sayNumber(voltagebyte_say,a,b);
myDFPlayer.play(1+a);delay(700);//dot
b=1;
sayNumber(decimalbyte_say,a,b);
switch(a){
    case 0:myDFPlayer.play(33);break;//voltage
    case 34:
        if(voltagebyte < 2){
            myDFPlayer.play(91);break;//voltage 91volt
        }
        else{
            myDFPlayer.play(92);break;//voltage 92volts
        }
    case 64:myDFPlayer.play(a+25);break;//voltage
}
}

void setup(){
    Wire.begin(8);                // join i2c bus with address #8
    Wire.onReceive(receiveEvent); // register event
    mySoftwareSerial.begin(9600);
    lcd.createChar(0,speaker);
    lcd.createChar(1,speakervolume1);
    lcd.createChar(2,speakervolume2);
    lcd.createChar(3,speakervolume3);
    pinMode(interruptPinL, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(interruptPinL), isrL, RISING);
    lcd.begin(16, 2);
    myDFPlayer.begin(mySoftwareSerial);
    myDFPlayer.setTimeout(500); //Set serial communicaiton time out 500ms
    myDFPlayer.outputDevice(DFPLAYER_DEVICE_SD);
    isrL();
}

void loop()
{
    volume_update();
    lcd_update();
    if((time_to_say <= millis())&&(vol != 0)){
        say_voltage(statusbyte, voltagebyte, decimalbyte);
        time_to_say = millis() + 4500;
    }
    delay(200);
}

void receiveEvent(int howMany) {
    statusbyte = Wire.read();
    voltagebyte = Wire.read();
    decimalbyte = Wire.read();
}

```



Stredná priemyselná škola elektrotechnická, Košice, SLOVAKIA
Střední průmyslová škola elektrotechnická, Havířov, CZECHIA

VOLTMETER WITH WEB OUTPUT



Co-funded by the
Erasmus+ Programme
of the European Union

Contents

1	Introduction.....	1
2	Theoretical part	2
2.1	Market analysis.....	2
2.2	Technical analysis	2
2.2.1	A/D Conversion.....	2
2.2.2	Channel ranges	2
2.2.3	File storage	2
2.2.4	VoltBoard Circuit	2
2.2.5	Power.....	4
2.3	Used Technologies.....	4
2.4	Financial Analysis.....	5
2.4.1	Hardware	5
2.4.2	Software	6
2.4.3	Our work.....	6
3	Practical part.....	8
3.1	Electronics	8
3.1.1	Summary Circuit	8
3.1.2	VoltBoard Circuit	8
3.2	Programs.....	9
3.2.1	Program for Arduino.....	9
3.2.2	Memory	10
3.2.3	Communication	10
3.2.4	Example JSON we send to the remote server	10
3.2.5	Debugging.....	10
3.2.6	Running modes.....	11
3.2.7	LCD.....	11
3.2.8	Configuration.....	12
3.2.9	Sampling	12
3.2.10	Flowchart.....	12
3.3	Mechanical	14
3.3.1	Case.....	14
3.3.2	Channel connectors.....	15
3.3.3	Range selecting.....	16
3.3.4	Switches and Buttons	16

3.4	Testing	17
3.4.1	Measurement precision.....	17
3.4.2	Issues	17
4	Conclusion	18
5	References	19
6	Attachments	20

1 Introduction

Voltmeter is an electrical instrument used to measure voltage between two places in an electrical circuit. There are many kinds of voltmeters these days but not many or even none of them is designed to simultaneously work on more channels. In our project, we made a physical device – the voltmeter that has 4 channel input and an Ethernet connector. We also wrote a website which receives data from a remote device and stores them.

User creates measurement logs which can be later send to a server and displayed in various charts.

Our product can be used, for example, by technical school students who have to measure voltage at many parts of some device at once very often in their classes. Additional simplification is that students can use our webpage to display several types of charts without creating them themselves.

We have chosen this project because the task seemed like it could be extended in many ways, and so did we. Also writing the documentation in English is beneficial for our language skills.

2 Theoretical part

2.1 Market analysis

On today's market, there aren't such solutions like ours. You can find there all types of multimeters, oscilloscopes and voltmeters (although mainly for indoor use in cars), because nobody cares about simplifying the life of technical school students in their classes. Companies don't focus on features like web output. Therefore, if our product was publicly available, only students might be interested.

2.2 Technical analysis

Our voltmeter consists of an Arduino with Ethernet Shield, our custom circuit on PCB and several mechanical components inside a case. We appreciated using Arduino as a microcontroller because it has a compact design, and it's easy to debug over USB. Libraries which are supplied with Arduino IDE are also an advantage for us, because they are open-source, they're designed for good code readability and stability and are meant to be used with official Arduino hardware and shields. Shields are "modules" that have same pinout as Arduino so one of them can be stacked on top of another.

2.2.1 A/D Conversion

Arduino has 10-bit A/D converters (their output is 0 - 1023). Their range is controllable using the onboard voltage reference sources – 1.1V, 2.56V, 5V and external (max 5V). If we wanted the Voltmeter to be applicable for students, we needed range about 0-30V. To solve that, we designed a circuit on PCB that reduces the input voltage and gave it the name "VoltBoard". It includes voltage dividers for reducing the input voltage and an operational amplifier that amplifies it back. For more information, see the section *2.2.4 VoltBoard Circuit*. The reference voltage for A/D converters was set to 2.56V.

2.2.2 Channel ranges

Not all 4 channels support same ranges. Channels 1 + 2 support 0-300mV, 0-3V and 0-30V voltage ranges and in addition we added option to connect a 1 Ohm resistor to the circuit so Voltmeter measures current through that 1 Ohm resistor and becomes an amperemeter. This can be done only on 300mV range so the current range is 0-300mA. Channels 3 and 4 can be set only to ranges 0-3V and 0-30V. All channels can be additionally completely disabled so user doesn't see them in log.

2.2.3 File storage

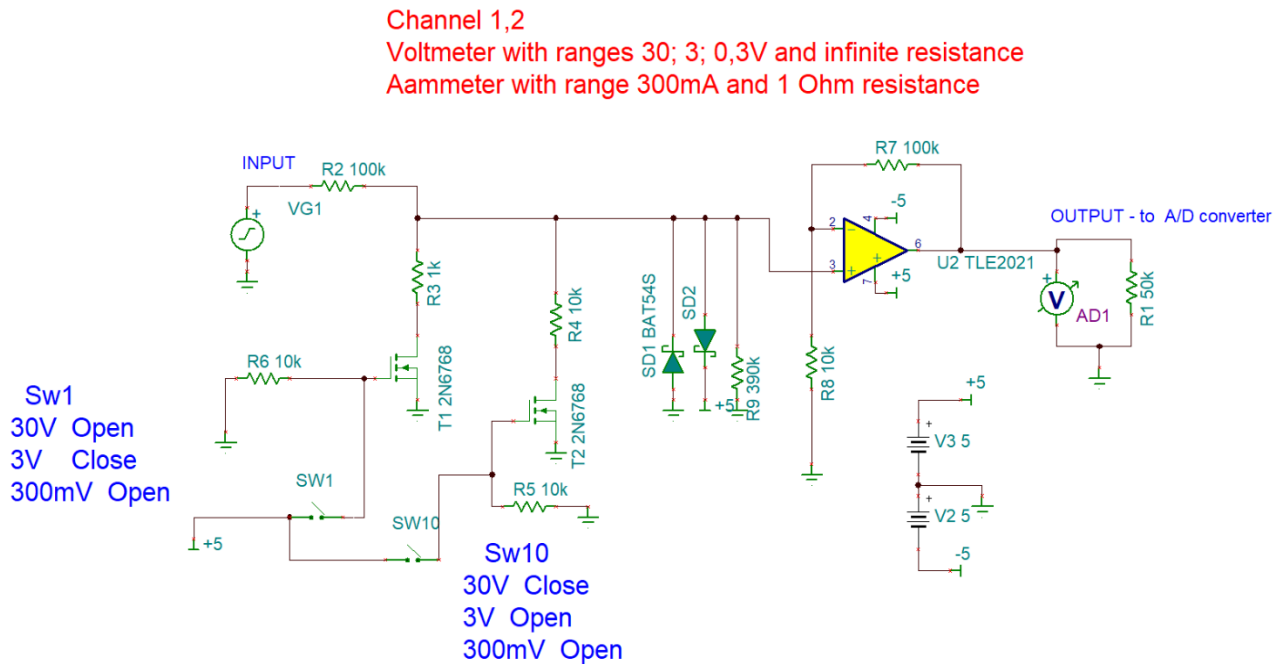
We needed a way how to configure the whole device without adding unnecessary controls like a keyboard. Luckily, Ethernet Shield includes a micro SD Card slot which we used for storing configuration data. SD Cards are a non-volatile media used to transfer files between portable devices. Our purpose wasn't to transfer the files. It would be quite complicated because you will need to disassemble whole device to get the card out of it. We instead used it as a storage for our configuration and log files.

2.2.4 VoltBoard Circuit

The core of the circuit is a 4-channel operational amplifier (for details about specific electronic parts, see the section *2.3 Used Technologies*). Before an input channel is connected to the amplifier, it must pass through controllable voltage dividers. They are made of one static resistor and one (on channels 3,4) or two (on channels 1,2) "dynamic" ones that are connected through MOSFET transistors so we can switch them on and off. To make the circuits safer, two Schottky diodes are connected to each input and each output of the amplifier. If there will be too high voltage connected, they're going to let the current flow through them – to ground and not to the rest of circuit. (This is called [voltage clamping](#))

2.2.4.1 Simulations

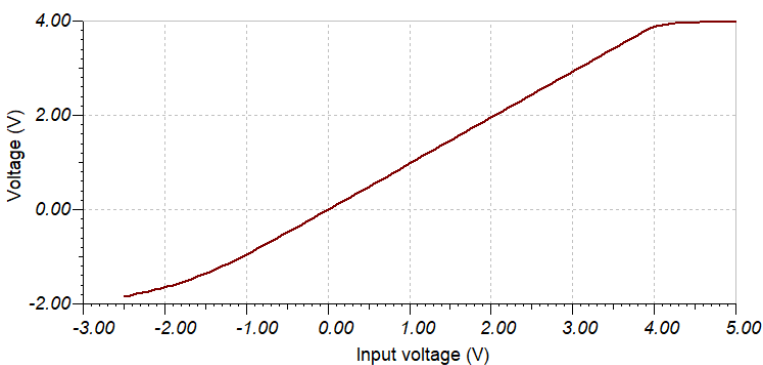
To ensure circuit stability, we performed several simulations of the range-switching circuits in the TINA



application. This simplified schematic of one channel was used.

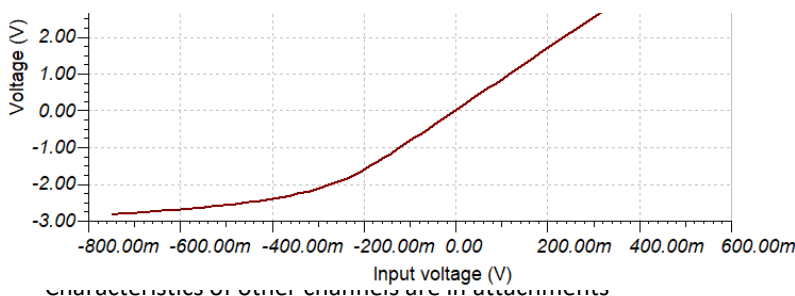
Transfer characteristic should be linear independently on selected range. Only when voltage exceeds range that operational amplifier accepts (or is negative) it's cut off.

Figure 1 - 300mV Range Transfer Characteristic

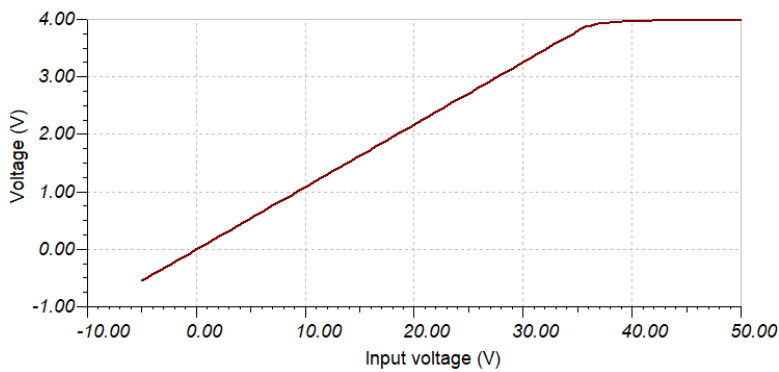


When 3V range is selected, the upper cut-off is more aggressive, and the negative cut-off is slower. This applies only to channels 1 and 2.¹

Figure 2 - 3V Range Transfer Characteristic – Channels 1,2



Characteristics of other channels are in attachments



Problem happens with the 30V range. Here it's visible that negative input produces fairly linear negative output. For that reason, in the real circuit there are also Schottky diodes on the op-amp output.

Figure 3 - 30V Range Transfer Characteristic – Channels 1,2

2.2.5 Power

The whole appliance can be powered by the USB or the DC power plug which are included on the Arduino. Because we had the resources, we decided to add an option to power it by battery. For that reason, we added a Power Switch (Battery/External) as can be seen in *Figure 4- Block diagram*. The operational amplifier requires symmetric power supply; therefore, we used an inverting voltage converter to power it.

2.3 Used Technologies

- [Arduino IDE](#)^I was used for programming the hardware
 - Version 1.6.8
 - Libraries: LiquidCrystal, Ethernet
- The code is written in C++ with use of [AVR Libc](#)^{II} and open-source Arduino libraries
- A 3D printer was used to print the case for our hardware (ANYCUBIC KOSSEL Linear)
 - Delta "Fused Deposition Modelling" technology
 - Nozzle diameter: 0.4 mm
 - The case was separated to 4 parts for it could be printed by this printer
 - Material: PLA
- 3D model was created in SketchUp
- Arduino MEGA had to be used, because demands on the memory size and pin count was huge
 - ATmega2560
 - 54 Digital pins
 - 16 Analog input pins
 - 256 KB flash memory
 - 8 KB SRAM
 - Clock Speed 16 MHz
- Arduino components
 - 16×2 char LCD display
 - Arduino Ethernet Shield 2 with microSD Card slot
- [TI TINA](#)^{III} was used as simulation platform for VoltBoard circuit
- [Eagle](#)^{IV} was used to design the VoltBoard
- All electronical parts on VoltBoard are SMD
- For adjusting voltage range, we used voltage dividers and operational amplifier (TLE2024CDW) that was powered by an inverting voltage converter (MAX1044/ICL7660)
- Switching MOSFET transistors in voltage dividers (BSS123W)
- Dual Schottky diodes for voltage clamping (BAT54S)

- Rotaryswitches(LORLIN CK1060)
- 4 AA Battery container
- Physical parts are connected using jumpers and pin headers
- Communication with server is serialized in JSONfor better manipulation on server side
- HTTP has been selected as the communication protocol
- HTML (Hyper Text Mark-up Language) is used to build web pages and structures in a web browser. We used this hypertext mark-up language to create web application
- CSS (Cascading Style sheets) is used to create structured documents. Cascading style sheets separate the structure and the design of the web page. In another words, we used CSS to create design of our web application
- PHP (Hypertext Preprocessor) is open source programming language originally designed for web development. It is executed on the server's side and we used PHP to fetch data from the database stored on the server.
- JavaScript is a high-level, interpreted programming language that is characterized as dynamic, weakly typed, prototype-based and multi-paradigm. We are using JavaScript to process data from measurements in our web application.
- Chart.js is open source JavaScript's library, which enables us to display graphs on a canvas
- SQL (Structured Query Language) is domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). We used SQL in order to create a database and to control updates and deletes over the database
- MySQL is open source relational database management system. It is multiuser and multithread database system supported on many platforms like Windows, Linux, Solaris ... The database consists of tables which have rows and columns. Table's cells are used to store data of measurements
- JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

For information about how are these parts connected, refer to *Figure 4- Block diagram*

2.4 Financial Analysis

	Helpful	Harmful
Internal origin	Strengths	Weaknesses
	Amount of simplification it provides	Low precision
External origin	Opportunities	Threats
	Demand from schools	Schools will keep their verified solutions

2.4.1 Hardware

- Mechanical parts: 367.25 CZK (14.27 € ²)
- Arduino + components (shield, display): 2063 CZK (80.16 € ²)
- Electronic components: 251.591 CZK (9.77 € ²)

² EUR = 25.73 CZK

- 3D printing: approx. 100 CZK (3.87 € 2)

Total: 2681.841 (104.21 € 2)

An Auto Range TRMS multimeter could be bought for the same price.³

2.4.2 Software

Within software part, we had to pay for a server, where measurements and web application will be stored. We chose a virtual private web server LAMP with operating system Linux, distribution CentOS 6.x 64bit. This server has storage of 1GB RAM, 20GB of disk storage and one virtual processor. It features also one network adapter Ethernet 01 with IP address 85.255.10.223 and MAC address 00:50:56:a0:b8:1d. The total cost of the server is 2,79€ for a month.

We were also considering to buy our own domain, although we have not bought it yet, because it is not necessary. The average cost of the second level domain is 9,95€ for a year.

2.4.3 Our work

We imagine our work would take approx. 3 months if skilled professionals did it.

Monthly salary

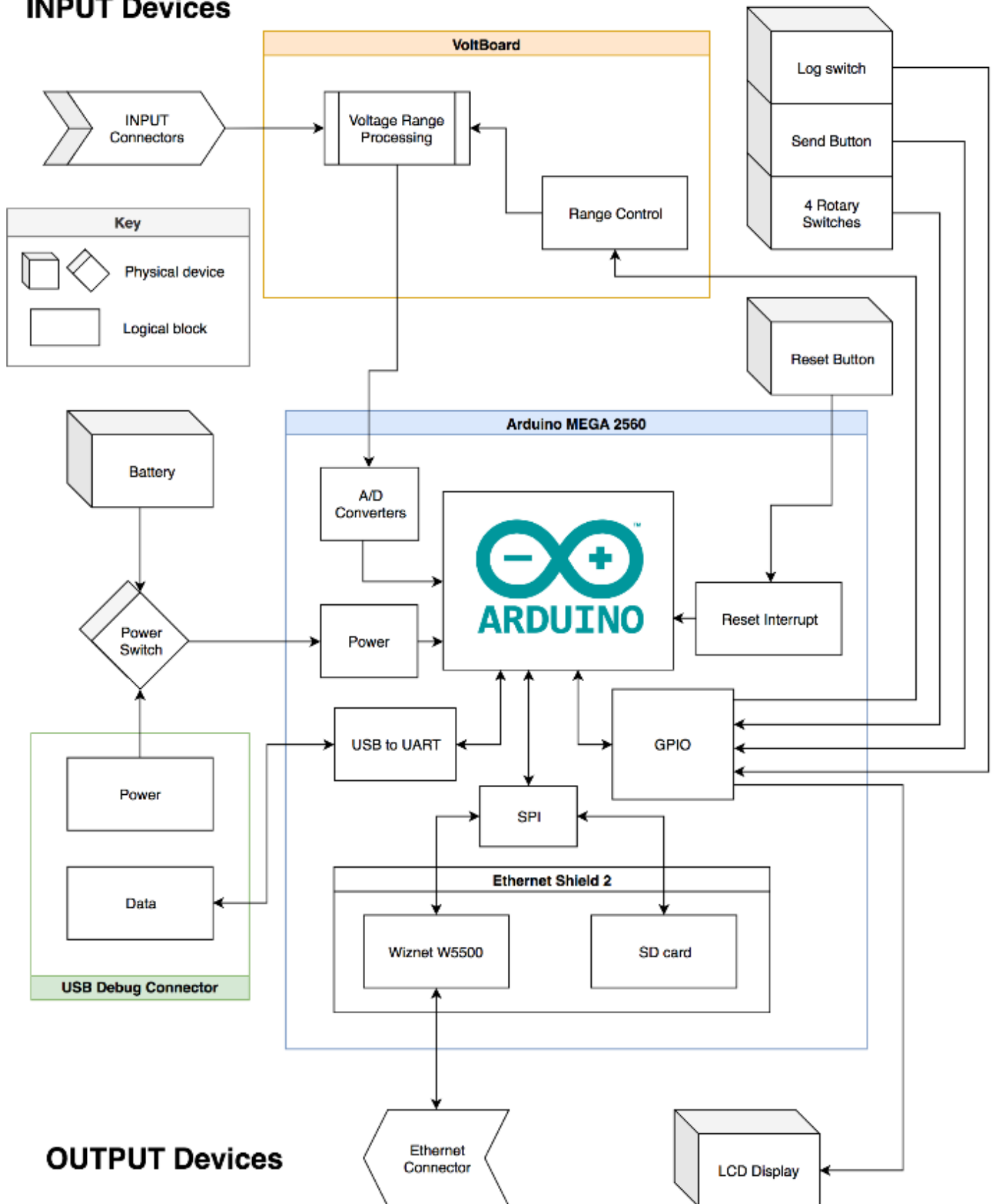
- for a microprocessor programmer in Czech Republic is 52 307 CZK⁴ (2 026.53 €3) in average.
- for a PHP programmer in Slovakia is 1 607 €⁵

³ Source: google.com

⁴ Source: platy.cz

⁵ Source: platy.sk

INPUT Devices



Ondfej Sabela 2019

Figure 4- Block diagram

3 Practical part

3.1 Electronics

3.1.1 Summary Circuit

- You can see it in attachments - *Figure 18 - Summary Schematic*

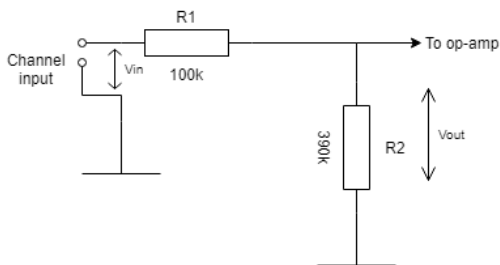
3.1.2 VoltBoard Circuit

- You can see it in attachments - *Figure 15 - VoltBoard Schematic*

For the range control, we have used controllable voltage dividers and operational amplifier. To ensure security of voltage inputs, Schottky diodes were added to each channel as voltage clampers.

3.1.2.1 Values of Controllable Voltage Dividers

For each channel: The op-amp has 100k (Rf) and 10k (Rs) resistors in its feedback circuit, so its gain is constantly 11 (it's $1 + R_f/R_s$). An advantage is that it also provides infinite input impedance⁶, so our

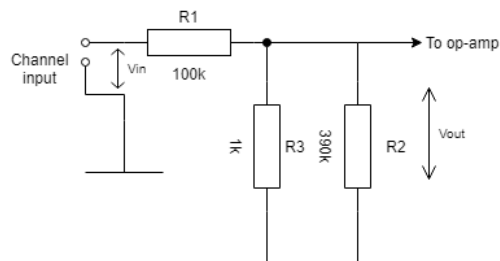


Voltmeter will act like an ideal voltmeter⁷. Before op-amp there is a voltage divider, that looks like this by default:

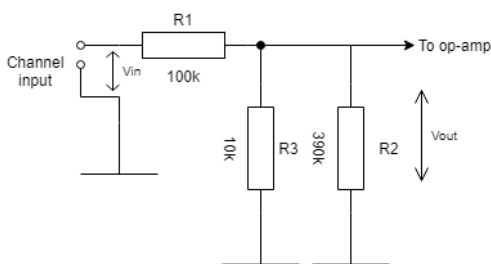
Here the R_1/R_2 ratio is 0.256, so V_{out}/V_{in} ratio is 0.796.⁷

If we connect 300mV to input, we get 238.78mV at output.⁸ Operational amplifier will amplify it 11x to 2.62 V and we can convert it in A/D easily. But we cannot connect 30 volts

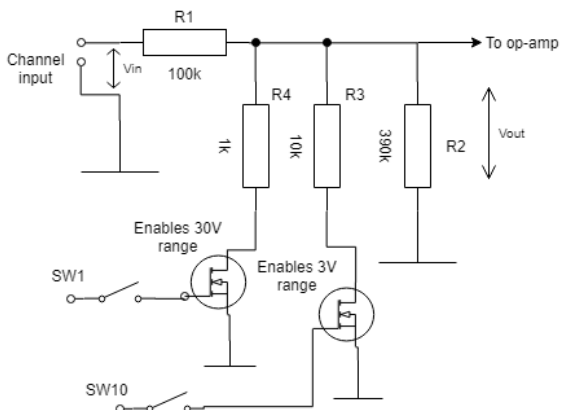
there.



However, if we add a 1k parallel resistor, together with the 390k R2 they have the resistance of 997 Ohms. That makes the dividing ratio 100.301 and V_{out}/V_{in} is 0.01. Now we can connect a 30V input, the divider will divide it to 0.296V and the amplifier will increase it to 3.256V. That voltage is applicable for us.



It works the same way for 3V range, except for R3 value, which is here 10kOhms. $R_3 + R_2$ in parallel are 9.75k \Rightarrow 3V will decrease to 0.266 V and after amplification, it will be 2.926V.

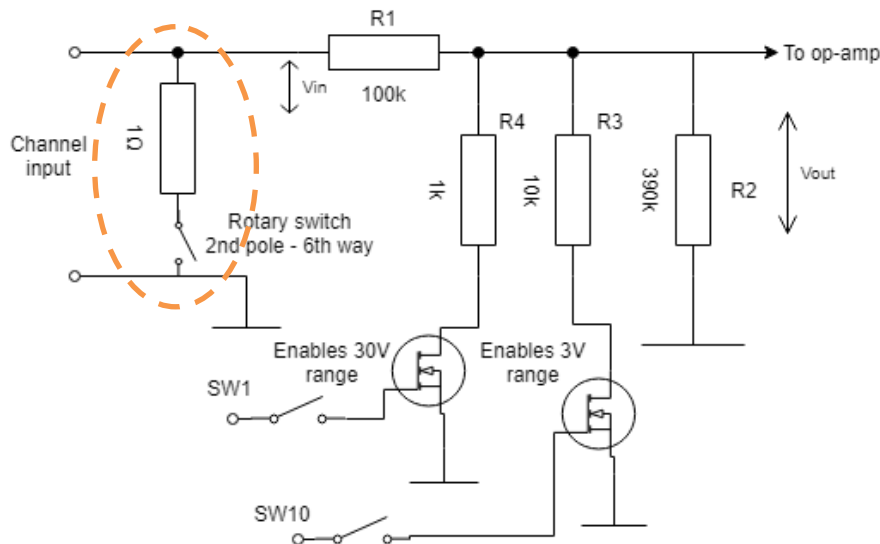


The result circuit will consist of all these three variants merged into one, with use of transistors. Note: real VoltBoard includes

/ the voltage dividers. They cause 200mV voltage drop on they own. See

additional circuit parts. See the next section for more information.

3.1.2.2 Pseudo-Ammeter



This easy trick enables us to measure not only voltage but also current. By attaching 1 Ohm resistor between input pins voltmeter will then measure voltage drop on that resistor⁹, which is equal to current. The resistor is attached mechanically through the second pole of the rotary switches. More information in the section 3.3.3 Range selecting.

3.2 Programs

3.2.1 Program for Arduino

Arduino program is written in C++ with use of Arduino IDE and separated into

- MultiChannelVoltmeter.ino – main program, main loop, communication with user and with server
- TakeSample.h – functions for taking different kind of samples from different channels
- Debug.ino – accepting debug commands from UART
- LCD.h – LCD configuration and displaying of values
- MultiJSON.h – custom library for JSON creation

Program is further described using flowchart at the page 13.

⁹ $V_{\text{drop}} = I * R$

3.2.2 Memory

Arduino MEGA has only 8192 bytes of RAM so we heavily use the F() macro for storing strings in ROM. The ROM size is 253 952 bytes. This makes 4420 bytes of RAM available. Compiled program size is 62 356 bytes.

3.2.3 Communication

Communication with server is established using HTTP which is a text protocol. For easier manipulation at server side, we store measurement logs as JSON. It has this format:

3.2.4 Example JSON we send to the remote server

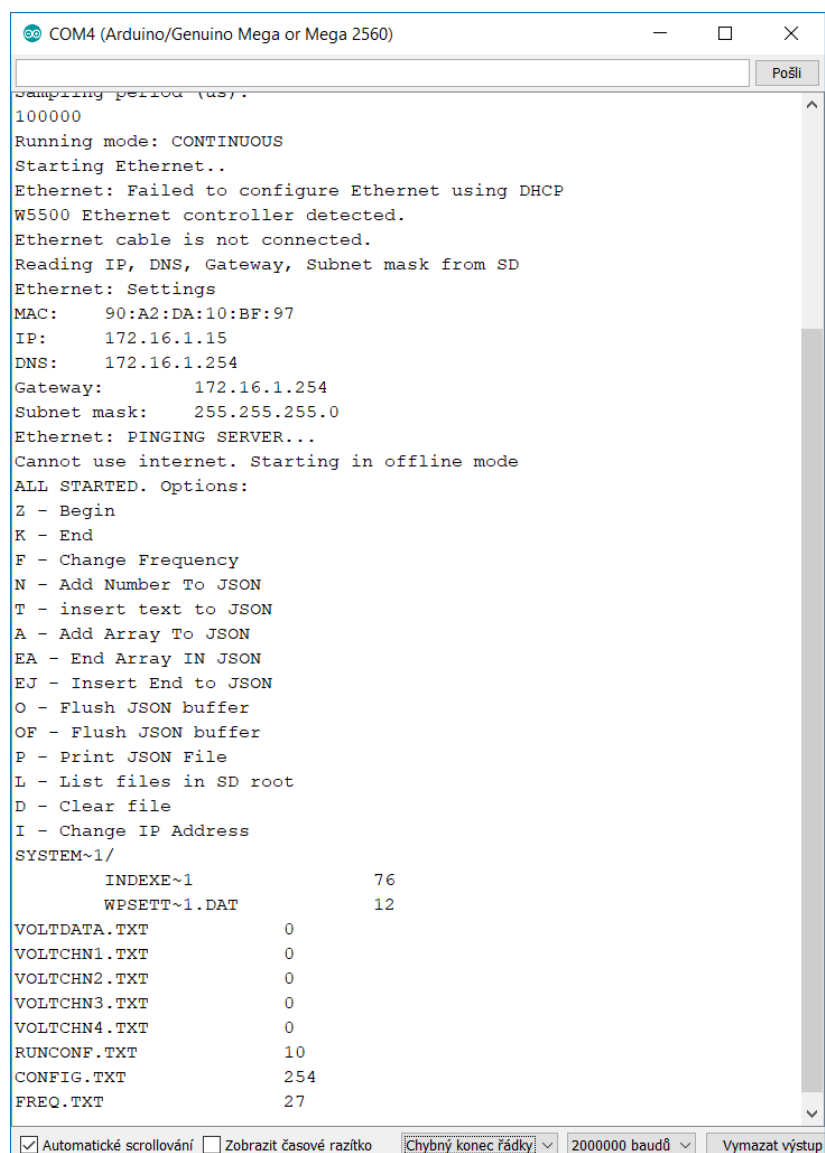
```
{
  "freq": 20000, //in Hz or "manual" if MANUAL SAMPLE mode has been used
  "duration_s": 0,
  "duration_us": 150, //in  $\mu$ S. (total duration = duration_s + duration_us/10^6)
  "averaging_count": 256, //count of samples which have been averaged to take 1 sample
  "range": [
    "30V",
    "3V",
    "300mV",
    "300mA"//30V, 3V, 300mV, 300mA, AUTO, OFF
  ],
  "channels": [
    [
      20,
      21,
      19
    ],
    [
      1.5,
      1.6,
      1.5
    ],
    [
      100,
      120,
      125
    ],
    [
      2.5,
      1.05,
      1.0
    ]
  ]
}
```

3.2.5 Debugging

Program accepts several debug commands that user can send to Arduino through USB Serial. Program is not case-sensitive. The baudrate must be set to 2000000 (2M) and line ending has to be turned off as you can see in *Figure5 - USB Debugging*.

You can see list of specific debug commands in *Figure14 - Flowchart 2 in Attachments*.

Figure5 - USB Debugging



3.2.5.1 Debug measurement

When user types "Z" into serial port, a continuous unlogged measurement will start with the sampling frequency configured in config files. It can be stopped by the letter "K". Some difficulties may occur if the sampling frequency is too high, because Arduino might not be able to receive input from serial or not even send any output. In that case, only the RESET button will work. Apart from taking samples, Arduino will also print A/D converter values, selected ranges etc. into serial.

3.2.6 Running modes

The core part of our Voltmeter application is creating measurement log files. We designed Voltmeter to have 2 different modes of operation (how values are written to the logs)

- Continuous sampling
When user toggles "Log switch (the blue one)" to "START" position, the Voltmeters begins to take samples at a defined sampling frequency and stores them to SD card until user switches back to "STOP" position
 - This is useful for making oscilloscope-like measurement
- Manual sampling
When "Log switch" is at "START", Voltmeter waits until "SEND" button is pressed. When user presses it, it displays current real-time values on display. When user releases the button, Voltmeter takes a sample and stores it on SD card
 - This is useful for making for example various characteristics charts - when you need to change the input parameters and then take next sample

Running mode can be changed when user presses the SEND button at the phase of SD Card initialization at start-up.

3.2.7 LCD

At the start of program, user sees information about "booting up" the Voltmeter and about some of settings (Running mode, Ethernet...). Then if the CONTINUOUS running mode is active, user immediately sees a screen with all channel's values. There is a small V1, V2... etc. symbol on the left side of each channel's value. (These are custom chars defined in LCD.h) Information about selected ranges is not displayed on LCD, user must see them on labels on the voltmeter's chassis. Also, on the rightest side of the display, you can see 2 special symbols – one for SD Card, one for Ethernet and an X or • symbol next to them. The X indicates that the card or ethernet connection is not working. None logs can be made when SD Card is inaccessible.

When the device is waiting for an input from serial port via USB, there is a message displayed on the display. That happens when some debug commands are executed.

3.2.8 Database

The MySQL database is created on our web server using SQL language. This data consists of one table with a name "merania ". "Merania" has five columns:

- Id_merania - this colmun is configured as a primary key of the table. It is used as a specific identification of each row in the table. The data type is set to unsigned int, which enables us to store 65,535 different measurements
- Nazov_merania – this column is configured to store name of the measurement.

3.2.9 Configuration

There are several files stored in SD Card's root folder (their names can be changed in program, but must be compliant with the [8.3 convention](#)). Specific settings which we store on SD card:

- Sampling frequency
- IP Address, MAC Address, Default Gateway, Subnet Mask (or whether to use DHCP)
- Running mode

We included an 1GB card formatted as FAT16 in our product.

3.2.10 Sampling

We have done analysis of Arduino A/D converters and decided, that program will take several samples and work further with their average. The amount of these samples is controllable using the `AVERAGING_COEF` constant in MultiChannelVoltmeter.ino and will show up as `averaging_count` in JSON.

3.2.11 Flowchart

The program begins in the file MultiChannelVoltmeter.ino by the function setup(). Here's the main program.

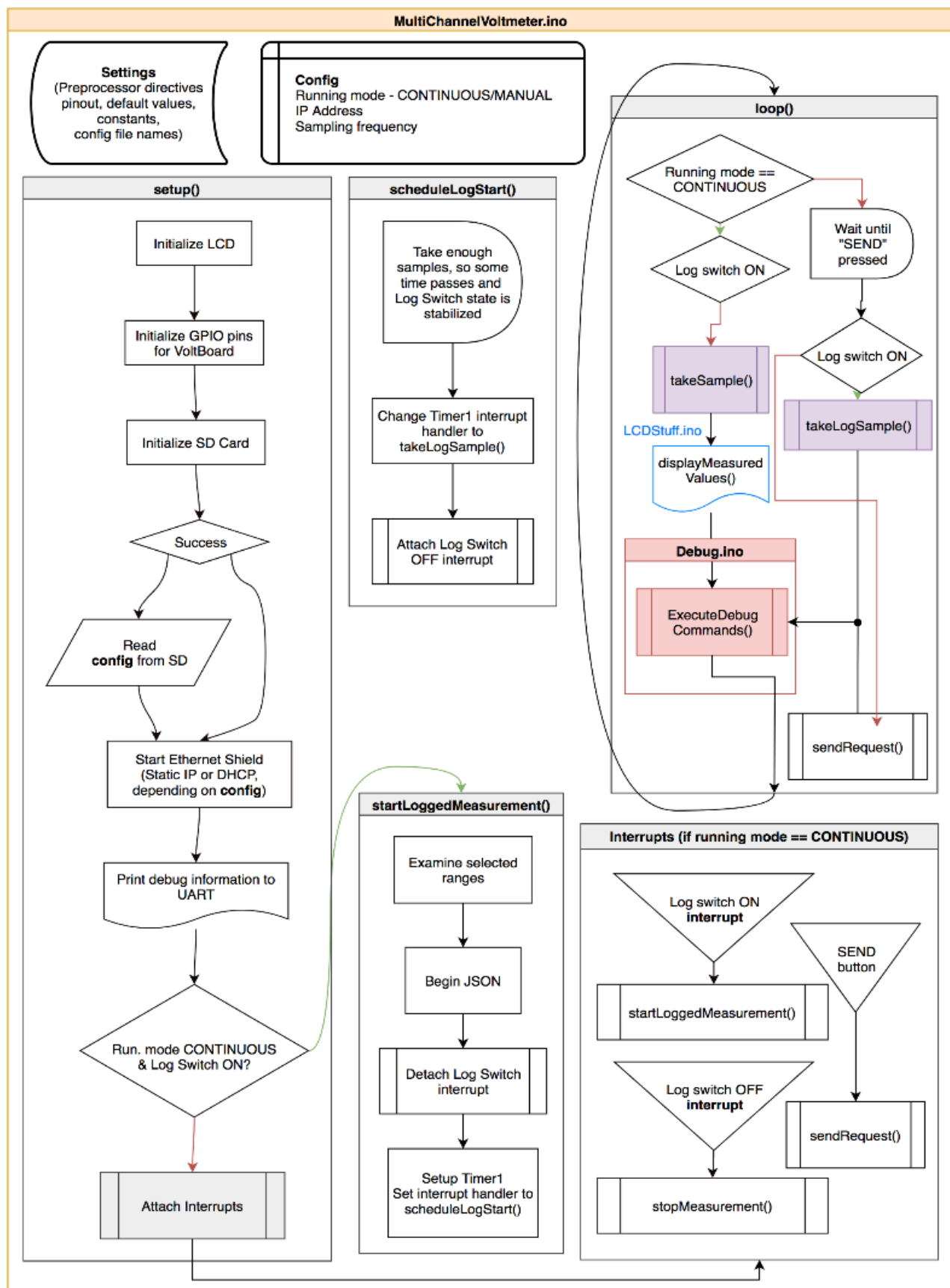


Figure6 - Flowchart 1

For more detailed flowchart of other parts of program, see the *Figure14 - Flowchart 2 in Attachments*.

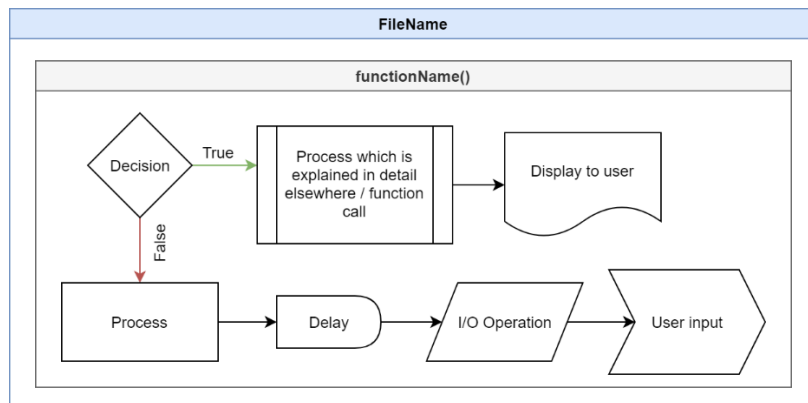


Figure7 - Meaning of diagram symbols

3.3 Mechanical

3.3.1 Case

The physical Voltmeter is enclosed in case with these parameters:

- Width: 120mm
- Height: 50mm (60mm with all mechanical parts attached)
- Length: 200mm
- Weight: 150g (370g including all hardware)

Mechanical controls are glued to the case by a glue gun. In the real market application (if we had more time to experiment with 3D printer to determine right dimensions of holes in surface) all parts would hold together without use of glue.

Figure 9 - Case with components inside visible



Figure 8 - Case with components without top part

Figure 10 - Case

As you can see the bottom part has a cable container at the right. Users can store the connectors for all the channels there.

3.3.2 Channel connectors

JST SM connectors have been used for providing a way to attach Voltmeter to a circuit. Their pros include

- The female connector can hold in well sized hole without need of glue.
- They are cheap
- A lock that prevents them from separating
- Polarized shape

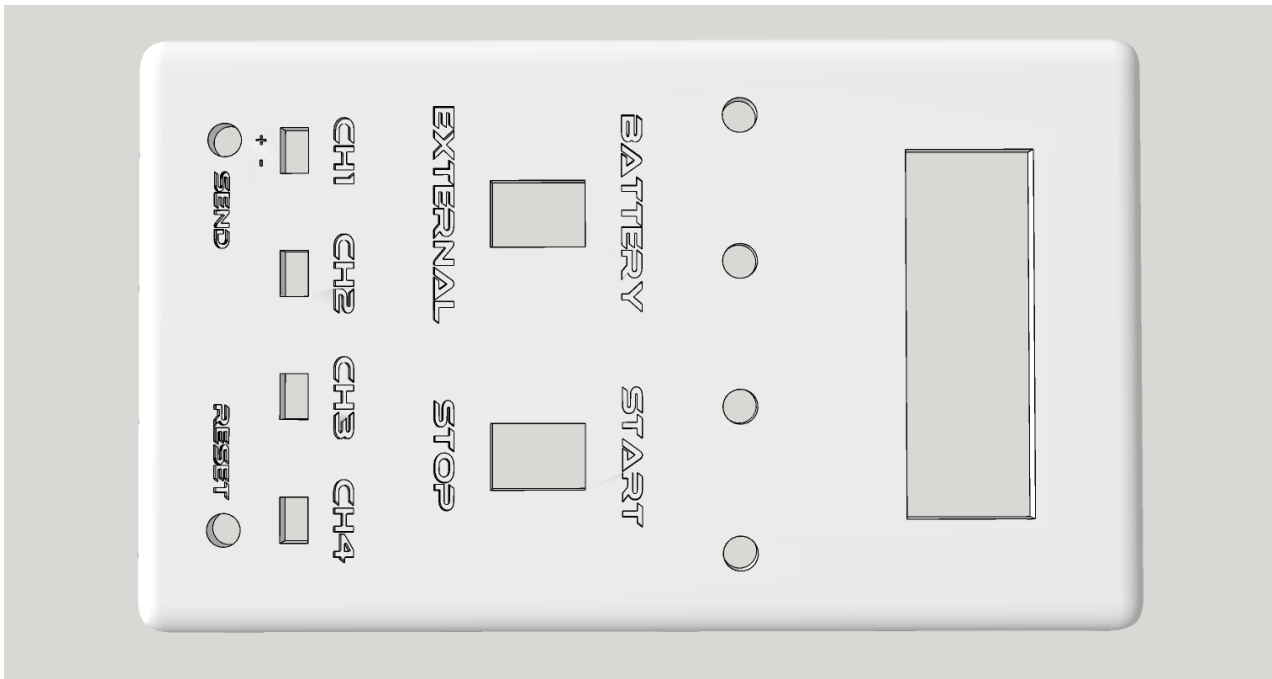


Figure 11 - Case - Top part

Here is the top part of the case. There are holes for buttons (SEND, RESET), switches (POWER – BATTERY/EXTERNAL, LOG – START/STOP), channel connectors, rotary switches and LCD.

3.3.3 Range selecting

Rotary switches (LORLIN CK1060) are used to select range of each channel.

They are 2 pole 6 position switches. First pole is used as range select. Positions are labelled as

Channels 1,2	Channels 3,4
1. OFF	1. OFF
2. AUTO	2. AUTO
3. 30V	3. 30V
4. 3V	4. 3V
5. 300mV	5. OFF
6. 300mA	6. OFF

On first 2 channels the second pole's position 6 is used to attach a 1 Ohm resistor to circuit and transform the voltmeter into ammeter.¹⁰

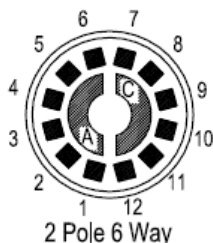


Figure 12 - Rotary switch
pins, image from
datasheet

3.3.4 Switches and Buttons

All switches and buttons have one pole attached to ground and the second to an Arduino's digital pin with enabled internal pull-up resistor.

¹⁰ More information on page 2, for complete explanation look into complete schematic

3.4 Testing

3.4.1 Measurement precision

This table shows random measurement values compared with ordinary multimeter (TB 313) with automatic range and $\pm 0.5\%$ accuracy.

Range	Our value	Multimeter
300mV	93.95	95.5
300mV	18.33	15.7
300mV	213.76	219
3V	0.961	0.960
3V	0.9	0.920
3V	1.17	1.243
30V	29.56	28.8
30V	0.52	0.415
30V	12.28	12.58
300mA	18.36	20.1
300mA		
300mA		

3.4.2 Issues

Sometimes SD Card isn't loaded correctly. When this happens, usually disconnecting power source and connecting again is sufficient.

Using commands 'Z' and 'K' to start and stop debug measurement works only with sampling frequencies below 40 Hz. When the debug measurement starts, Arduino isn't fast enough to read from serial line, so it can be stopped only using the "SEND" button. If the frequency is as high as 40 Hz, only reset can stop it. Arduino might or might not output some random data to UART.

Our voltmeter has very bad impact on the measured circuit (about 200mV – 2V voltage drop, depending on range and actual voltage). That is about 20x worse than if you just connected one wire to Arduino's analogue pin and another to ground.

Output value is highly dependent on calibration constants in TakeSample.h. Values may be even negatives if calibration is inaccurate.

4 Conclusion

Simulations are nice but... not everything that you learn at school is accurate. The technology is interesting, but when you connect all channels to same circuit, you get a voltage drop of 800mV. You can't even measure anything on 300mV ranges. But we learned something.

Ondřej Sabela's opinion: Our teamwork was not definitely a teamwork, because each of us was working on totally different platforms – we didn't even need to see each other. However, when there was the opportunity, we were working together and helping each other, for example with the calibration or with JavaScript errors.

When people from our schools met each other, it was a good time, despite it was full of hard work. When looking at other students' projects, sometimes I got inspired.

I devoted a lot of time to this project. Most valuable skills that I got are probably solving problems with Arduino and electronics, designing a printable case and working with not-so-working 3D printer.

Problems that we needed to solve:

- Arduino reads string badly from UART
- Too slow output to UART
- Small buffer for converting between string and decimal numbers
- Bad school's 3D printer, used own one
- Difficulties with pin header tightness
- LCD connector positions upside down
- SD Card's SPI pins 50-53 collide with others
- Interrupts not working as expected
- Too fast button response
- Fitting text into small LCD display
- Too high rotary switch poles

What about doing a business with our project? Maybe in Japan, where people are massively interested in simplifying even the smallest parts of their lives. Definitely not in other countries on our planet.

5 References

- I. <https://www.arduino.cc/en/main/software>
- II. <https://www.nongnu.org/avr-libc/>
- III. <http://www.ti.com/tool/TINA-TI>
- IV. <https://www.autodesk.com/products/eagle/overview>
- V. <https://www.allaboutcircuits.com/textbook/direct-current/chpt-8/voltmeter-impact-measured-circuit/>
- VI. https://sk.wikipedia.org/wiki/Hypertextov%C3%BD_zna%C4%8Dkov%C3%BD_jazyk
- VII. <https://sk.wikipedia.org/wiki/JavaScript>
- VIII. [https://sk.wikipedia.org/wiki/PHP_\(skriptovac%C3%AD_jazyk\)](https://sk.wikipedia.org/wiki/PHP_(skriptovac%C3%AD_jazyk))
- IX. https://sk.wikipedia.org/wiki/Structured_Query_Language
- X. <https://sk.wikipedia.org/wiki/MySQL>
- XI. <https://sk.wikipedia.org/wiki/CSS>
- XII. <https://www.json.org/>
- XIII. <https://sk.wikipedia.org/wiki/JQuery>

Project's website is available at the address <http://85.255.10.223>

Project's issues were often solved using the biggest developers community at the address <https://stackoverflow.com/>

Official web page of Chart.js library is available at the address <https://www.chartjs.org/>

Project's troubles were also solved at the address <https://www.w3schools.com/>

6 Attachments

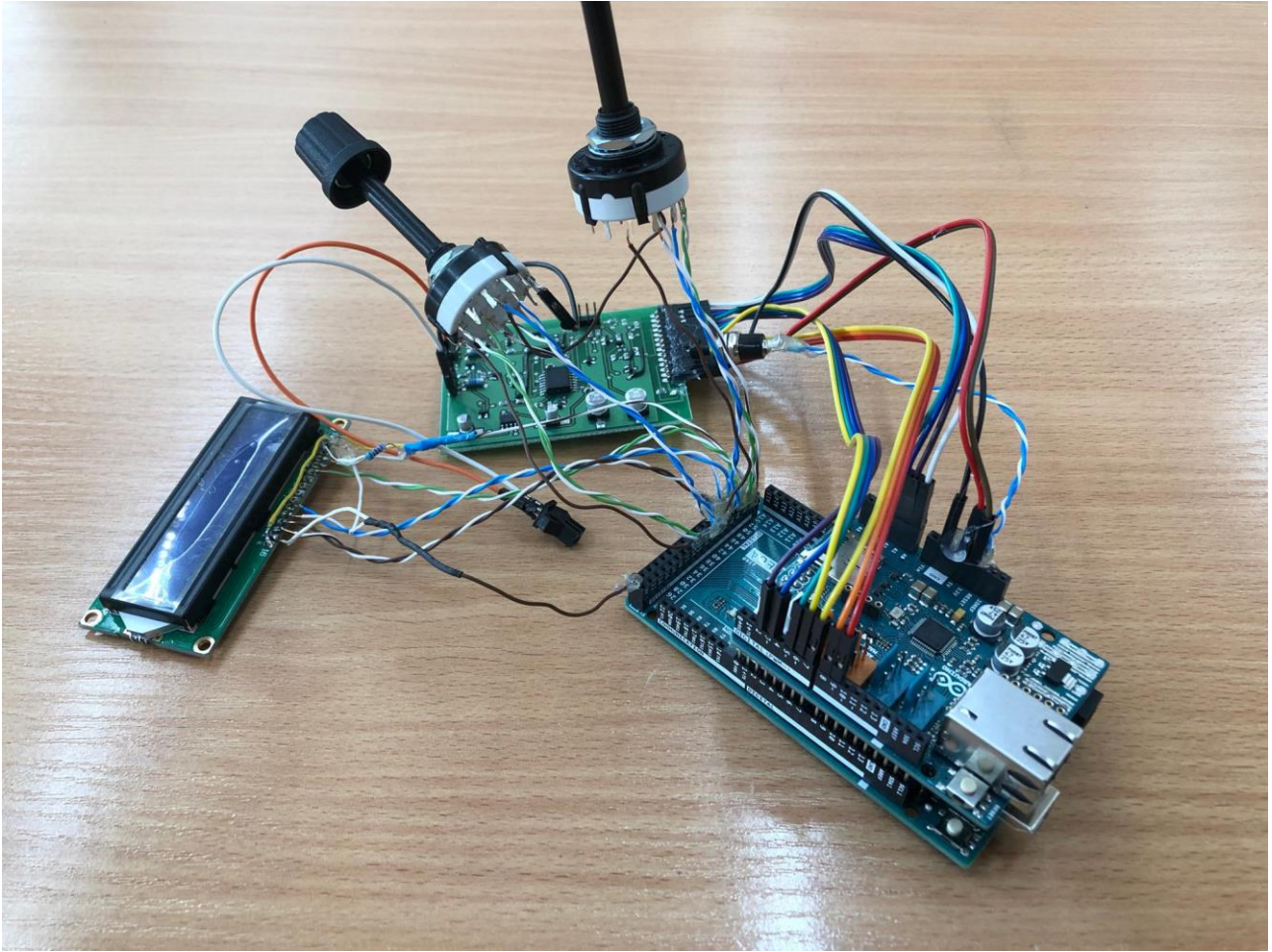


Figure 13– Prototype

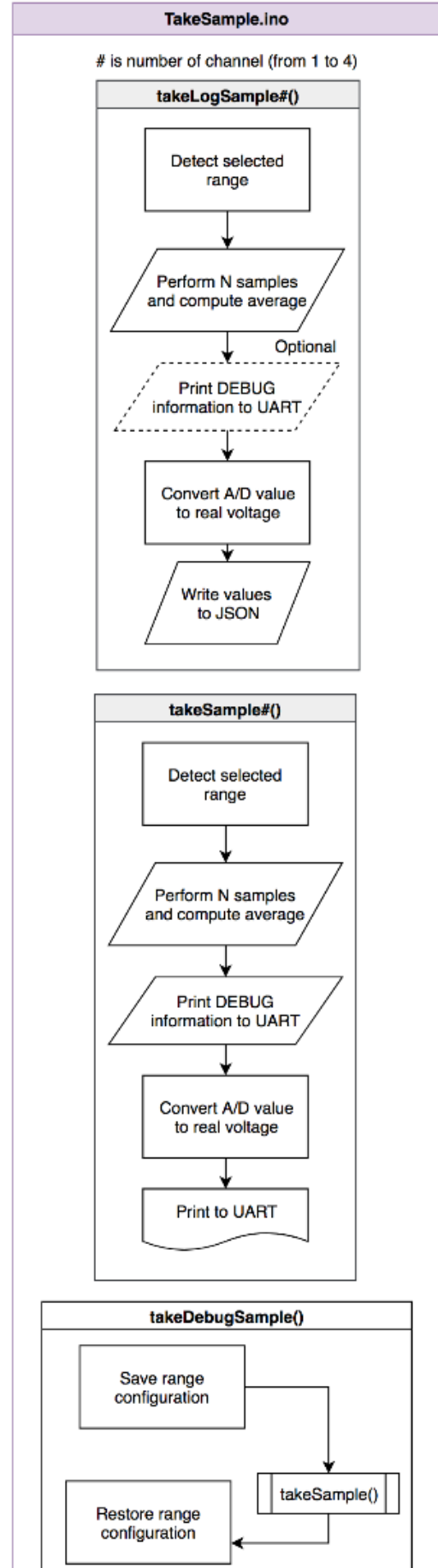
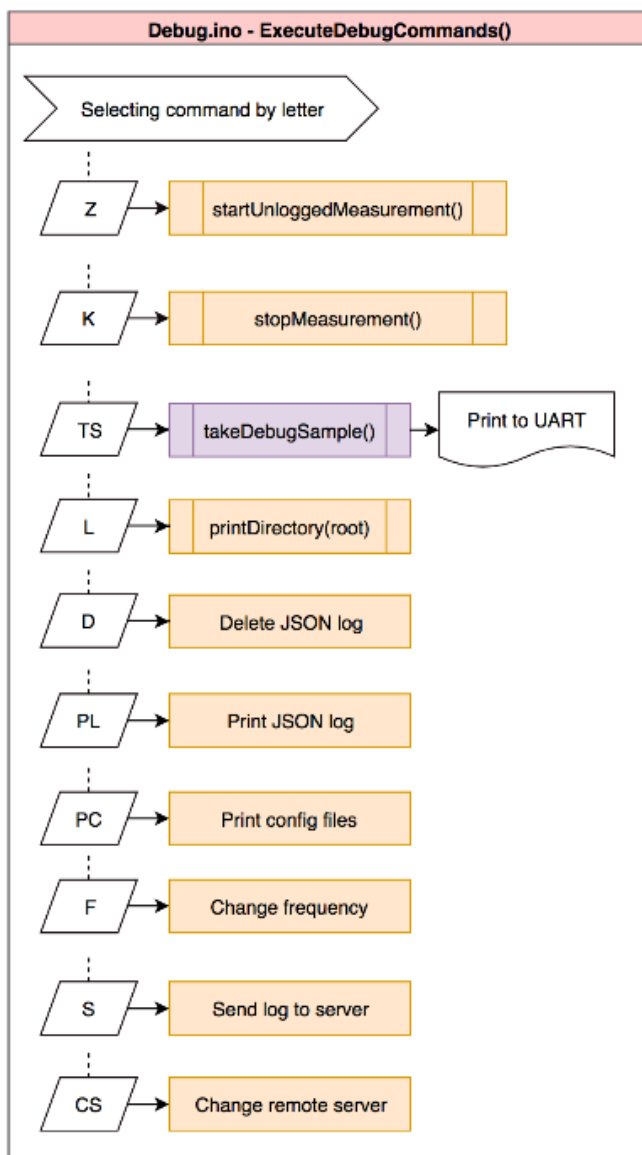
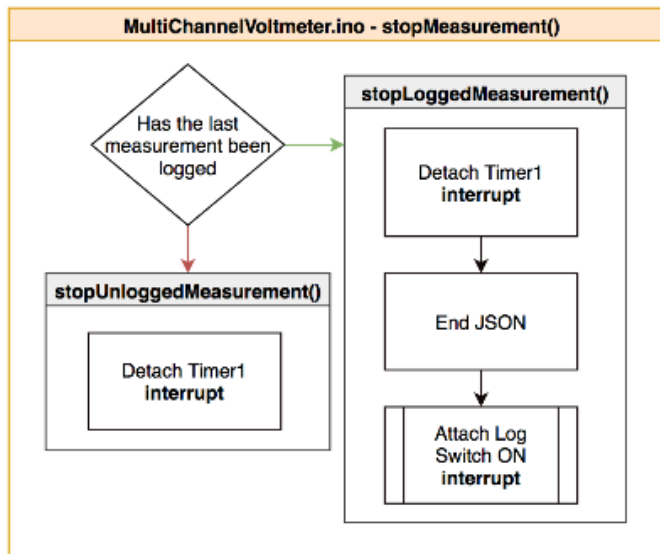
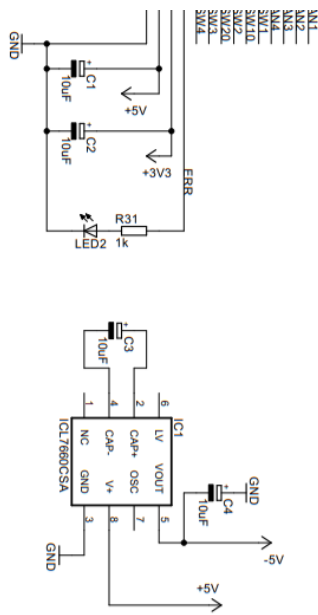
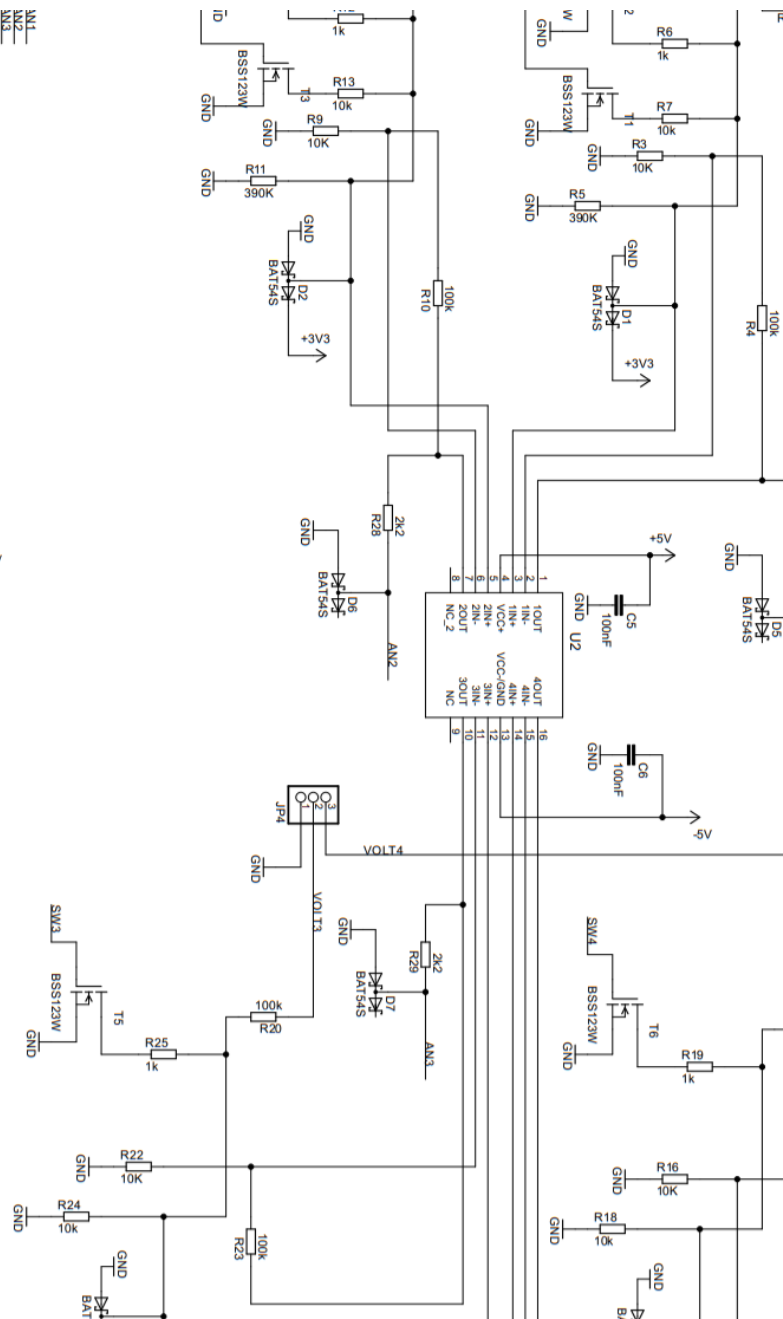


Figure14 - Flowchart 2



TITLE: volt3
Document Number:
Date: 06.11.2018 20:02

Figure 15 - VoltBoard Schematic

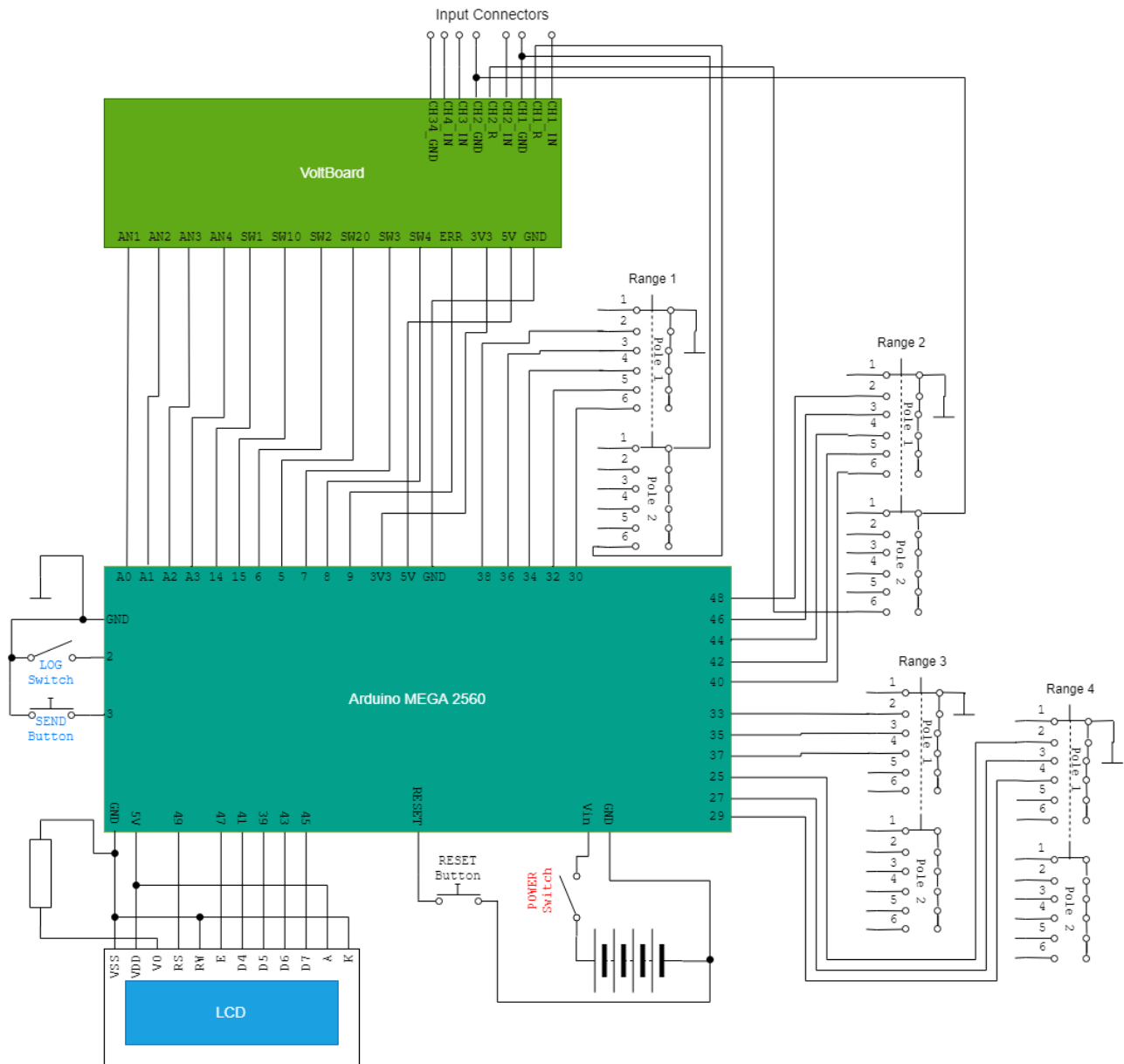


Figure 18 - Summary Schematic

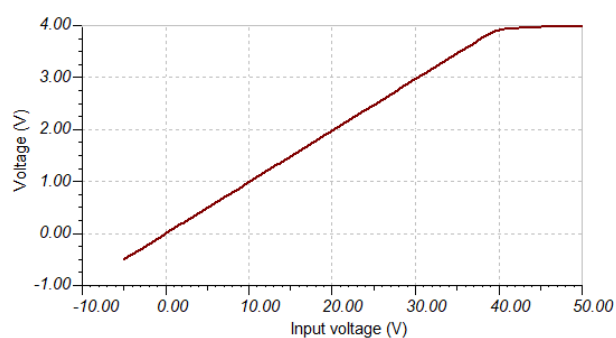


Figure 17 - Channels 3,4 Simulated Transfer Characteristic - range 30V

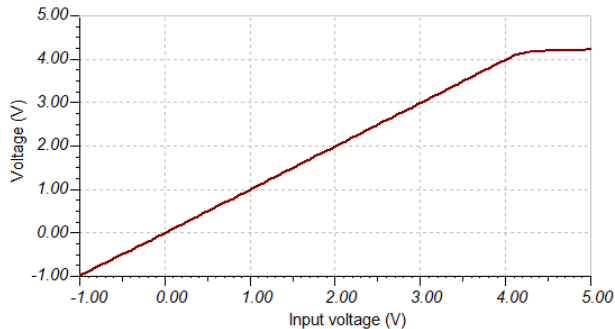


Figure 16 - Channels 3,4 Simulated Transfer Characteristic - range 3V

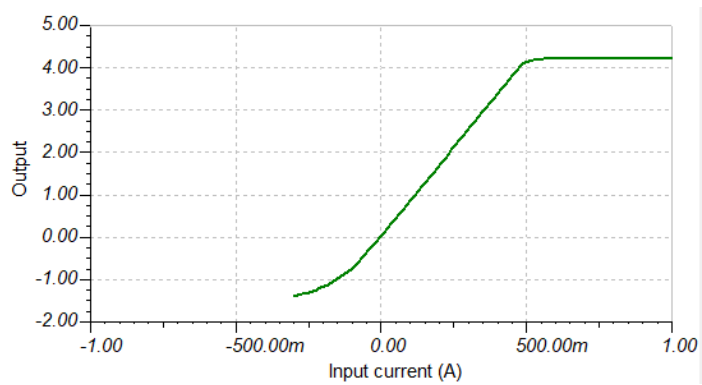


Figure 19 - Channel 1,2 Simulated Transfer Characteristic - range 300mA



Stredná priemyselná škola elektrotechnická, Košice, SLOVAKIA
Srednja poklicna in tehniška šola, Murska Sobota, SLOVENIA

SYSTEM FOR FEEDBACK OF SCHOOL LUNCHES



Co-funded by the
Erasmus+ Programme
of the European Union

Table of Contents

1	INTRODUCTION	1
2	THEORETICAL PART.....	3
2.1	MARKET ANALYSIS	3
2.2	TECHNICAL ANALYSIS	4
2.3	USED TECHNOLOGIES	4
2.3.1	<i>Software</i>	4
2.3.2	<i>Hardware</i>	7
2.4	FINANCIAL ANALYSIS	8
2.4.1	<i>Cost of software and hardware</i>	8
2.4.2	<i>Cost of labor</i>	9
2.4.3	<i>Retail price</i>	9
3	PRACTICAL PART.....	10
3.1	SOFTWARE	10
3.1.1	<i>Server (back-end)</i>	10
3.1.2	<i>Web (front-end) interface</i>	11
3.1.3	<i>Device interface</i>	12
3.2	HARDWARE	13
3.3	MECHANICAL.....	14
3.4	TESTING	14
4	CONCLUSION	15
5	REFERENCES	16

1 Introduction

Having a good diet is very important for everyone and even more so for students in the stage of growing up. They need to eat quality healthy food and that is what school canteens are for. School canteens are part of almost all school institutions, whether it is a primary/secondary school or even a university.

The main problem, however, is that people do not really enjoy eating at school canteens. They prefer to eat in buffets, fast foods or they just buy easy processed food in the nearest shop. Those options, on the other hand, are not as good in nutritional value and are usually far more expensive. The only real alternative is to eat at home, but that is generally not possible because most people do not have the time nor ability to leave school. School canteens, on the other hand, check all the boxes. They should provide a good nutritional value, are cheap because they are funded by the government (they are even free in some countries) and are usually part of the school complex, so you can go get lunch anytime you have 10 minutes of free time.

So why are so many students not using the services of school canteens? It's probably safe to say, that it is a majority of them.

They might not like the canteen environment. They might think it is not cool to eat budget food. But the main reason is probably that they do not enjoy the food served there. And we cannot blame them. They generally cannot pick the food they want and have to eat what is on the list for that day. If they do not like that food, they will go hungry that day unless they choose another option.

That is the problem we personally faced and we wanted to do something about it. How could we improve this? How could we make the food taste good? A voting system!

What is the easiest way to improve the quality of your food? To get proper feedback. Yes, of course, the cooks may ask a few students whether they liked the food, but that is not a good way to go about it. A few students do not represent the entire school. We need to engage all of them. A properly setup voting system that makes it easy to give your feedback on the food you have eaten is the way to go about it. We are pretty sure, that with our system, it would be possible to get the engagement rate higher than 70% on every meal. Why? Because we have made it user-friendly and quick.

Our voting system provides the cooks with accurate feedback data. They will be able to see, which meal was successful and which one was not. Having that data is very important because they can see it right away and have far more space to experiment with the food. They will also improve their cooking skills this way. Everyone wins here.

2 Theoretical part

2.1 Market analysis

Our project idea is nothing ground-breaking. It is simple. Yet, after doing some market research we found out, that if we really wanted to push this project more, it would definitely have potential. Would it, however, be a profitable venture in its current form? No. The main customers are school canteens and school canteens do not have a budget to afford a solution like this. Our project would only work as a non-profit.

There is a possibility for making a profit though - pivoting our project towards restaurants. Restaurants are not run by the government but as a business. That means, that they can afford to spend a good amount of money even on things that are not a necessity. We would have to do market research first to find out if it would make sense to go into this area. If we decided to, we would have to change a few things and focus on commercial usage.

We would have to shape the project into something like a template. That way it could be easily customized into the needs of customers without spending too much time on it. The main part of the application would be the same, but each business would have a different graphics, colors and other easily customizable stuff like the scale of rating on which you can rate a meal on.

Another thing that would have to be done is to make it extensible, so we can add some additional functionality based on the needs of individual customers. So, if someone wants to measure the quality of customer service, cleanliness of the place or overall impression of a restaurant, that functionality could be included without much work.

Do services like this already exist? While there are multiple companies providing voting systems, we have not found one specializing in this exact use case. They are usually specializing in other areas and not in our market niche which is voting on meals. You definitely can hire a company and they can create such system for you, but that would be definitely very costly. That is why the biggest advantage of this project would be its low price of implementation as we already have the backbone figured out and we could spit out the implementation for individual businesses in about a week on average, all while improving our main codebase with each new iteration.

2.2 Technical analysis

We know, that our project will contain web and device interfaces. They also need a way to store the data. Because the data are not exclusive to one user, we cannot store them in a cache or in device memory. We have to have a server with a database responsible for data handling. This server needs a public IP address so those interfaces will not have to be on the same network to communicate with it. It also should have at least 1GB of RAM to handle the Java application. Communication will be handled with HTTP calls (REST API).

Web interface will be a single page application (SPA) and the best framework for our purpose is Angular. It is very important, that its design is very user-friendly and provides a quick way for the user to do the task he wants to do. That is the reason why we decided for SPA – it provides both. It is also a technology that is growing in popularity rapidly. The idea for the web interface is, that it should provide a way for customers to vote from wherever they want. They should be able to use their computer or phone and just log into their account which was assigned to them. This is particularly useful if you were in a hurry or forgot to vote when you were in a canteen.

Device interface should provide a way for customers to vote right on the spot in the canteen. The time it takes to vote here also has to be really short, because there is only one device and a queue can develop. If that happens, it is no longer quick and people will just skip the voting.

Authentication should be realized using RFID technology and every customer will have his own RFID tag in addition to his login information used in the web interface. Those two credentials will be linked to one account, so every customer has only one vote. It should be made of a microcomputer, touch screen, RFID reader and it should be all packaged in a case made specifically for this device.

2.3 Used Technologies

2.3.1 Software

When choosing our software stack, we wanted to pick as few technologies as we could, but still get the job done efficiently. We went for mature but modern technologies that are also used in commercial enterprise applications. That is mainly because we wanted to make sure we finish the project and did not want to get stuck because of some limitation imposed upon us by a new hot programming framework. There are also far better resources and documentations to confront in case of a problem

Java

Object-oriented programming language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere", meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to "bytecode" that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. (1)

We have chosen Java, because we already had experience with it and because it is a mature language with many libraries and great documentation. It also gave us an option to use a great framework - Spring and its own platform JavaFX.

Spring

Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform. A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments. (2)

Spring framework is another mature technology ideal not only for big enterprise applications but also for small projects like this. It is the most popular Java framework – even more popular than official Java EE solution developed by Oracle. It contains a lot of modules that can be included in the application in case of need. We used 3 of them.

Spring MVC – web module that allows our application to communicate through the internet using HTTP protocol and REST endpoints.

Spring Security – module responsible for authentication and authorization. It is very important, that every customer has his own account, but also that he will not have admin privileges such as the ability to start and stop voting.

Spring Data – module that simplifies database communication and access. It is packed with object-relational mapping framework Hibernate and on top of it provides another layer of abstraction customized towards the rest of Spring framework.

Java FX

JavaFX is a software platform for creating and delivering desktop applications, that can run across a wide variety of devices and it is the standard GUI library of Java platform. (3)

We have used JavaFX because we needed a GUI application for our touchscreen and it satisfied our need just fine.

Typescript

TypeScript is an open-source programming language for application-scale JavaScript. TypeScript adds optional types to JavaScript that support tools for large-scale JavaScript applications for any browser, for any host, on any OS. TypeScript compiles to readable, standards-based JavaScript. (4)

Typescript is the default language used by Angular. It is very similar to Java because it is object-oriented. That means, that all code we wrote throughout the entire project has almost the same structure.

Angular

Angular is a TypeScript-based open-source web application framework that makes it easy to build applications with the web. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. (5)

We picked this solution because we wanted to create a web application similar to a native Android application. We were thinking of using React instead, but we chose Angular because it is more similar to our backend solution (Spring). Applications created in Angular are called „single page application“ and they refresh themselves in the background using HTTP requests. While this can be done using native JavaScript, it is incomparably harder.

REST API

Representational State Transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services. RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations. When HTTP is used, as is most common, the operations (HTTP methods) available are GET, HEAD, POST, PUT, PATCH, DELETE, CONNECT, OPTIONS and TRACE. By using a stateless protocol and standard operations, RESTful systems aim for fast

performance, reliability, and the ability to grow, by re-using components that can be managed and updated without affecting the system as a whole, even while it is running. (6)

MySQL

MySQL is an open-source relational database management system (RDBMS). With its proven performance, reliability and ease-of-use, MySQL has become the leading database choice for web-based applications, used by high profile web properties including Facebook, Twitter, YouTube, Yahoo! and many more. (7)

We picked MySQL mainly because it is the most popular choice for projects like ours and has great documentation.

2.3.2 Hardware

Raspberry Pi

The Raspberry Pi is a low-cost credit-card sized single-board computer. The Raspberry Pi was created in the UK by the Raspberry Pi Foundation. The Raspberry Pi Foundation's goal is to "advance the education of adults and children, particularly in the field of computers, computer science and related subjects." (8)

It is the exact type of computer we needed for our project. We could not imagine anything better. Arduino was not an option, because it is only a microcontroller and not a microcomputer. It is not possible to run a full-fledged operating system on it. Raspberry provides everything we could wish for – 4 USB ports, Wi-Fi, Ethernet port, incredible processing power for its size and most importantly the ability to run Linux on it.

Raspberry Pi display

Display of our choice is official 7-inch Raspberry Pi display with a capacitive touch technology and a resolution of 800x480.

RFID

Radio-frequency identification (RFID) uses electromagnetic fields to automatically identify and track tags attached to objects. The tags contain electronically stored information. Passive tags collect energy from a nearby RFID reader's interrogating radio waves. Active tags have a local power source (such as a battery) and may operate hundreds of meters from the RFID reader. (9)

RFID is the standard choice for use cases such as ours. We are using a reader that operates on a frequency of 125KHz. Every customer will have his own passive tag that he will use for authentication purpose.

2.4 Financial Analysis

2.4.1 Cost of software and hardware

The only expense really necessary for this project is a server. Basic web hosting usually provides an option to serve static web pages using Apache HTTP server or NGINX. This would be enough to serve our static web interface, but it cannot be used to serve an entire Java application. The best solution would be to just rent out a private server. There are many companies that provide this service (Digital Ocean, Vultr, Linode, ...). The cheapest plan would suffice as there will not be many users using the app at once. And if it for some reason will not be enough, it is easy to upgrade to the next server plan.

The estimated cost of a server:

Digital Ocean – Standard Droplet = **5€/month**

(Optional) Domain – **10€/year**

We could further decrease the cost by getting only one server and use it to host all of our customers.

The voting device is an **optional** extension, that our customers could order in the case of interest. It is only a one-time purchase – though a little more expensive one. We have built a prototype, but it would have to be further refined and improved on before we would be able to offer it as a commercial solution.

The estimated cost of a physical device we have made is:

Raspberry Pi 3 Model B+ = **30€**

Official Raspberry Pi 7-inch display = **80€**

RFID Reader (125KHz) RDM-6300 = **10€**

Case = **10€**

MicroSD card = **5€**

Other items = **5€**

Which gives us a combined cost of **140€**. This could be cut down by at least another 40€ would we decide to manufacture the device in a bulk. The device would also provide much more power and features than someone might need. To further cut down the cost, we could find a cheaper alternative to Raspberry Pi and also use a smaller resistive touch display instead of the official capacitive 7-inch Raspberry one. That way we should be able to fit within the budget of **70€** per device, but at the expense of its quality.

2.4.2 Cost of labor

Of course, we have to take into account the time it takes to customize the project towards the customer requirements, because time is money. It should take about 15 hours for the basic implementation of the system into a restaurant or canteen. If we take into account an average salary of 15€/hour, it would give us about **225€** in a labor cost per project. This, however, is by no means an accurate cost, it is just an estimate. We could also provide support on the 15€/hour basis.

2.4.3 Retail price

The implementation price of the project would be 100€ for a device and 225€ for manual labor, which is **335€** combined (not including the server cost, because it will be paid by a customer). That is also the price we would be offering the solution for institutions like school canteens if we decided for a non-profit.

If on the other we wanted to be profitable, we would also have to take into account the value of the whole project – everything we did up to this point. If we had to put a price tag on basic implementation, it would range somewhere between **1000€ - 1200€**. Of course, we would have to make our project suited for commercial use and market it before we can ask for money.

3 Practical part

3.1 Software

3.1.1 Server (back-end)

Server part is the core of our project. It is responsible for all of the storable data our project uses. It serves as a point with which every interface communicates. We only have two of them now – web and device interface, but we could add as many as we need. If we decided, that we want to extend our project with Android and iOS applications, it could be done without much trouble. Why is that? Because those interfaces are not responsible for data handling, they only act as a middleman through which you communicate with a server side. The codebase of the back-end is hosted on a server running OS Ubuntu and is accessible through a public IP address. Code is written in **Java** and our framework of choice is **Spring**. Application server Apache Tomcat, on which our back-end app is running, is also responsible for hosting static files of the web interface.

We chose **MySQL** as a database system. To model the database (Figure 1), we have used the object-relational mapping (ORM) framework **Hibernate**. It gave us the ability to use Java objects as savable database entities and vice versa. It simplified our workflow by a lot.

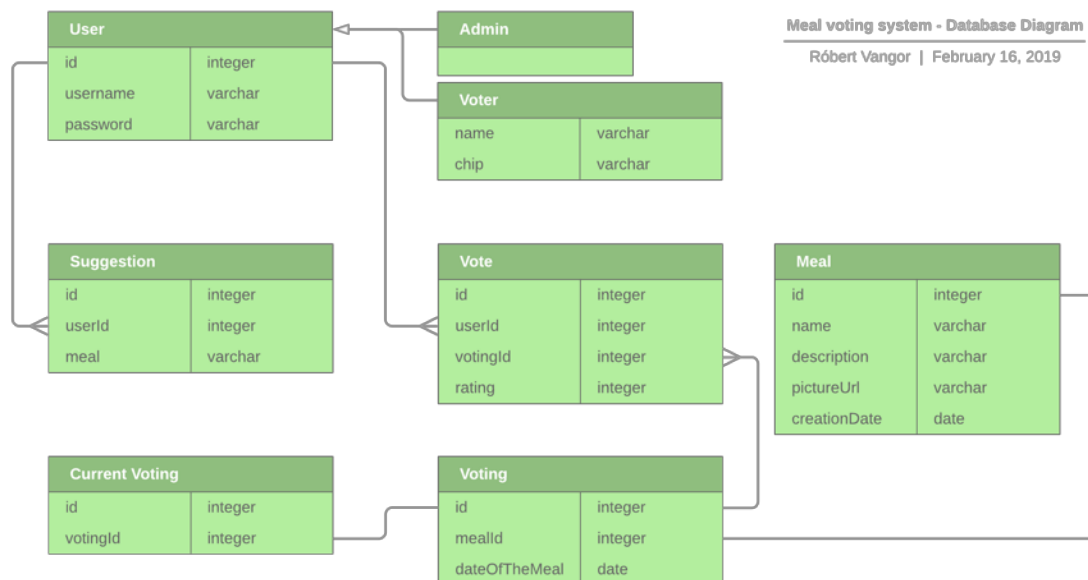


Figure 1 Database diagram

3.1.2 Web (front-end) interface

Web interface is a single page application developed using **Angular** and its functionality can be compared to a native Android app. It is very similar, but it is a web application and can be run on almost all devices. When designing the app interface, we tried to follow material design guidelines as well. The web application consists of admin and voting components. After you log in, you will get redirected to one of those based on your credentials.

Admin component provides a way for administrators to manage meals, past votings, review suggestions but most importantly to start or stop voting.

The screenshot displays the 'MANAGE VOTING' interface for an administrator. It includes a 'CURRENT VOTING' section for 'FRUIT SALAD WITH GRILLED CHICKEN' (3.67 / 3 VOTES) with an 'END' button. Below is a 'PAST VOTINGS' table listing previous meals, their scores, and vote counts. The 'MEALS' section features a 'CREATE MEAL' button and a list of existing meals with their scores. Finally, the 'SUGGESTIONS' section shows user-submitted meal ideas.

Date	Meal	Score	Votes	Action
18.2 .2018	Pork Chops in Lemon-Butter Sauce	2.5	2 votes	🗑️
16.2 .2018	Fruit Salad with Grilled Chicken	4.67	3 votes	🗑️
14.2 .2018	Smoked Salmon Carbonara with Lemon and Dill	2.0	2 votes	🗑️
13.2 .2018	Tofu Kabobs with Barbecue Sauce	1.5	2 votes	🗑️
12.2 .2018	Pork Chops in Lemon-Butter Sauce	2.5	2 votes	🗑️

Meal	Score	Action
Fruit Salad with Grilled Chicken	4.17	🗑️
Pork Chops in Lemon-Butter Sauce	2.5	🗑️
Smoked Salmon Carbonara with Lemon and Dill	2.0	🗑️
Tofu Kabobs with Barbecue Sauce	1.5	🗑️

User	Suggestion	Action
ROBERT	Grilled chicken with chilli & sesame seeds	🗑️
ROBERT	Healthy Chicken Burgers with Spinach Basil Pesto & Mozzarella	🗑️
MICHAL	Golden Beet & Beet Greens Pasta W/ Ricotta and Feta Cheese	🗑️

Figure 2 Admin component

Voter component provides a way for customers to cast their vote, see the results of past votings, see the rating of meals and give their suggestion for new meals.

The screenshot displays the 'Voter component' interface. At the top, it says 'CURRENT VOTING' with a 'Sign out' link. The current meal being voted on is 'FRUIT SALAD WITH GRILLED CHICKEN' with a date of '22.3.2019' and '3.67 / 3 VOTES'. Below this is a row of five circular buttons numbered 1 to 5, where button 4 is highlighted in blue. The 'PAST VOTINGS' section shows a table of previous meals and their ratings. The 'MEALS' section displays a list of meals with their ratings. At the bottom, there is a text input field for 'Give us a meal suggestion' and a 'SUBMIT' button.

Date	Meal	Rating	Votes
18.2.2018	Pork Chops in Lemon-Butter Sauce	2.5	2 votes
16.2.2018	Fruit Salad with Grilled Chicken	4.67	3 votes
14.2.2018	Smoked Salmon Carbonara with Lemon and Dill	2.0	2 votes
13.2.2018	Tofu Kabobs with Barbecue Sauce	1.5	2 votes
12.2.2018	Pork Chops in Lemon-Butter Sauce	2.5	2 votes

Meal	Rating
Fruit Salad with Grilled Chicken	4.17
Pork Chops in Lemon-Butter Sauce	2.5
Smoked Salmon Carbonara with Lemon and Dill	2.0
Tofu Kabobs with Barbecue Sauce	1.5

Figure 3 Voter component

3.1.3 Device interface

In the previous section, we focused on the hardware, now we will talk about what goes on inside of the Raspberry Pi. It is running a **Raspbian OS** which is just a distribution of Linux customized for our micro-computer. Because we already used Java on the server side, we have decided to be consistent and use it on the device as well. Specifically, **Java FX**, which is a graphic user interface (GUI) framework and a part of the Java platform. It fits our requirements perfectly. We also tried to imitate the design of the web interface. It displays current vote results and information about a meal that is being voted on, specifically its name, description and a picture (if there is one assigned to that meal). It is all being refreshed every 30 seconds or after a successful voting. The device can be in 3 states: unauthenticated, authenticated and no voting in progress.

The **unauthenticated state** occurs, when there is a voting in progress and the device is waiting for someone to authenticate with their RFID tag.

The **authenticated state** occurs after customer authenticates with his valid RFID tag and it holds either until he votes or until 15 seconds pass. After that, the device switches back to unauthenticated state.

The **no voting in progress state** is self-explanatory.

3.2 Hardware

The only part of the project focused on hardware is a voting device. All behavior is handled by **Raspberry Pi**. There are two additional parts connected to it.

The first part is an official 7-inch **Raspberry Pi display**. It is connected to Raspberry Pi using GPIO pins to provide power (5V and Ground) and using Digital Serial Interface (DSI) to provide a means of communication between them. Everything works perfectly, mainly because they are made to work with each other. The screen uses capacitive touch, so it is very responsive and quick. It also has a resolution of 800x480, hence it is capable of displaying pictures. Our first decision was to use a character LCD display and the buttons instead, but we would sacrifice a lot of functionality, so we chose a touch display even though the final product will be a lot more expensive.

The second part is an **RFID reader** (RDM-6300), which operates on a frequency of 125KHz. It is connected to Raspberry Pi using GPIO pins (Figure 2). It uses power pins (5V and Ground) and serial pins (RX/TX). And because RFID communication depends on electromagnetism, it needs to have a coil connected to it. We are using a UART protocol as a means of communication between a reader and Raspberry.

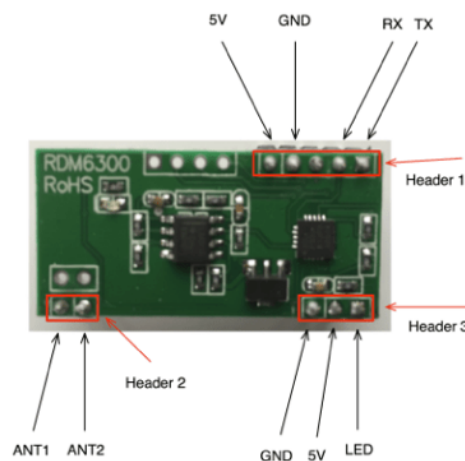


Figure 4 RDM-6300 wiring

The device needs to have a power supply with at least 2.5A. For the application to work, it needs to be connected to the internet whether it is through the ethernet cable or a Wi-Fi – our device provides both options. Another necessity is a keyboard in a case we need to configure something.

3.3 Mechanical

After we build the device, we had to create a case for it. First, we had to figure out the organization of parts. There was a requirement that needed to be satisfied – putting RFID reader on the front of the device for an easy access, so we put it side by side with a display. We also added a power connector and USB port to the back of the case, so it will not have to be opened up all the time. The material used for our case is plexiglass, which will not interfere with RFID's ability to read tags.

3.4 Testing

Overall, our project is working as it should. As of the writing, we are not aware of any major issue. Of course, nothing is perfect and there surely are some very specific bugs, but we still tried to minimize them. A lot of testing was done and we hopefully eliminated all problematic scenarios that could occur. A real-world usage would surely give us an answer to this.

4 Conclusion

It is safe to say that our project was a success. Not so much because of the final product, but because we have learned a lot on the way. We had a chance to improve in many areas of our lives, gain new experiences and meet a lot of new people. The most important thing was probably the fact, that we had to get out of our comfort zone and communicate in English – even on a professional level. We also had to learn how to collaborate with others and get the job done efficiently because we did not have much time to work together in person. It was not really a problem though. We were aware of that fact and we adapted to it. We have also got a chance to get to know another (even though very similar) country.

There were not any big problems that we have faced, because we planned ahead. Yes, we managed to blow out a display, but that happened mostly because the screen was faulty to begin with. Another thing learned – only buy electronic parts from trustworthy reputable brands. But even then, it was not a big setback, because we had a plan and we stuck with it.

Regarding the future of our project, we are probably not going to continue working on it. There are other more important things for us to do next. But the skills we gained while working on this project will help us tremendously should we decide to start working on something new in the future. If someone, however, wants to continue working on this project, we will be happy to further provide our knowledge of this field and all of our code.

We are very glad that we were part of this project. It was a great experience and we would certainly participate again. Our gratitude goes to everyone who made this project possible.

5 References

1. Wikipedia: the free encyclopedia. (2019). Java (programming language). [online] Available at: [https://en.wikipedia.org/wiki/Java_\(programming_language\)](https://en.wikipedia.org/wiki/Java_(programming_language)) [Accessed 19 Apr. 2019].
2. Wikipedia: the free encyclopedia. (2019). Spring Framework. [online] Available at: https://en.wikipedia.org/wiki/Spring_Framework [Accessed 19 Apr. 2019].
3. Wikipedia: the free encyclopedia. (2019). JavaFX. [online] Available at: <https://en.wikipedia.org/wiki/JavaFX> [Accessed 19 Apr. 2019].
4. Wikipedia: the free encyclopedia. (2019). TypeScript. [online] Available at: <https://en.wikipedia.org/wiki/TypeScript> [Accessed 19 Apr. 2019].
5. Wikipedia: the free encyclopedia. (2019). Angular (web framework). [online] Available at: [https://en.wikipedia.org/wiki/Angular_\(web_framework\)](https://en.wikipedia.org/wiki/Angular_(web_framework)) [Accessed 19 Apr. 2019].
6. Wikipedia: the free encyclopedia. (2019). MySQL. [online] Available at: <https://en.wikipedia.org/wiki/MySQL> [Accessed 19 Apr. 2019].
7. Wikipedia: the free encyclopedia. (2019). Raspberry Pi. [online] Available at: https://en.wikipedia.org/wiki/Raspberry_Pi [Accessed 19 Apr. 2019].
8. Wikipedia: the free encyclopedia. (2019). Representational state transfer. [online] Available at: https://en.wikipedia.org/wiki/Representational_state_transfer [Accessed 19 Apr. 2019].
9. Wikipedia: the free encyclopedia. (2019). Radio-frequency identification. [online] Available at: https://en.wikipedia.org/wiki/Radio-frequency_identification [Accessed 19 Apr. 2019].



Střední průmyslová škola elektrotechnická, Havířov, CZECH REPUBLIC
Zespół Szkół Technicznych, Mikołów, POLAND

WEATHER STATION



Co-funded by the
Erasmus+ Programme
of the European Union

Contents

1	Introduction.....	1
2	Theoretical part.....	2
2.1	Market analysis	2
2.2	Technical analysis.....	2
2.2.1	Measuring data about weather	2
2.2.2	Measuring the speed of the wind	2
2.2.3	Measuring the direction of the wind	3
2.2.4	Displaying the data.....	3
2.2.5	How everything communicates.....	3
2.3	Used Technologies	3
2.3.1	The Real Time Operating System.....	3
2.4	Financial Analysis	5
2.4.1	Used budget	5
2.5	SWOT analysis	5
3	Practical part.....	6
3.1	Electronics	6
3.1.1	Arduino Mega2560.....	6
3.1.2	BME280	7
3.1.3	SS411P	7
3.1.4	Arduino LCD 20x4.....	8
3.2	Programs.....	8
3.2.1	States of the program.....	8
3.2.2	The real time operating system.....	9
3.2.3	Tasks.....	10
3.3	Mechanical	10
3.3.1	The box with the PCB.....	10
3.3.2	The box with the Hall sensors.....	11
3.4	Testing.....	11
4	Conclusion.....	12
5	References	13
6	Attachments.....	14

1 Introduction

The goal of our project was to build a weather station that is able to measure temperature, pressure, humidity and the speed and direction of the wind. We use 2 types of sensors to read the needed data and a LCD display to show it.

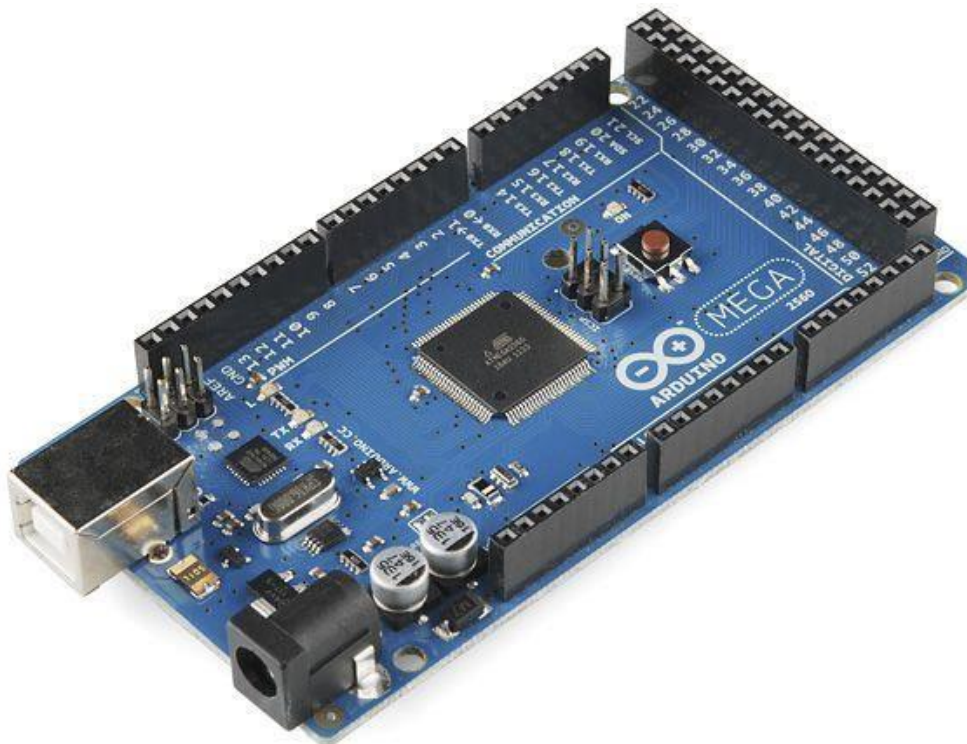
The weather station can be used by people to get information about the quantities mentioned above. It should be placed somewhere outside because there is no wind inside buildings but it's your decision.

Bořek:

I have chosen this project because I have never worked on something similar before. On the other hand you can easily check if the project works as it should be by using other already working gadgets to read the information about current weather. However during the development I faced many problems which I successfully solved and got new experiences.

Marek:

I picked it because it looked interesting and i wanted to learn something about Arduino, measuring weather, programming in C language and I like to potter a little.



2 Theoretical part

2.1 Market analysis

Most of the other weather station you can buy on the internet provide more functions and display even time or other data.

However not every product measures the speed or the direction of the wind. That's one of the advantages of our weather station. Also because we used a real time operating system every change like of temperature, humidity and so it display immediately without any delays.

There are huge differences between the costs of the weather stations. Small sized ones cost about 20 Euros and the price of the bigger ones moves even around 200 Euros. The bigger weather stations are able to measure the data about wind as well, but they cost much more than our product would cost.

Some disadvantages of our weather station is its size, use of two boxes connected through a cable that can be cut easily and that not all data are displayed at once.

I have also found weather station that provide internet connection via Wi-Fi.

To sum everything up I think that our station is perfect for outdoors measurements for a fair price and displays you enough important data.

2.2 Technical analysis

Our goal was to create a useful weather station but keep it simple for us as well. So we picked sensors that are able to measure more information at once.

2.2.1 Measuring data about weather

To measure all the needed data about weather, we use a BME280 sensor. It's very simple to use because it gives instant result in a digital form. All we had to do here was to connect it to our microcontroller and we were ready to read the data.

2.2.2 Measuring the speed of the wind

One of the tasks we were supposed to do is to calculate the speed of the wind and display it to the user. To calculate such thing we decided to use a hall sensor. This sensor generates either logic "1" or "0".

Generated value is affected by magnets. Hall sensor will be described later. So when the wind rotates the arrow on our weather station, the arrow also rotates a magnet hidden below. As the magnet rotates it affects the output of the sensor. We register the moment when the sensor starts to generate logic "0" (as time value). Then the wind affects the arrow and the magnets starts to rotate. When it switches the output of the sensor to logic "1" we register this moment as well and now we also know that when the sensor generates logic "0" again we can end the calculation and start a new one. So after the calculation is ended we know how long it takes the magnet to turn around once. Now I had to use a bit of physics.

You can use this equation to measure speed if you know time and distance:

$$v = \frac{s}{t}$$

v...speed(m/s)

s...distance(m)

t... time (s)

However we were having only time at this point. So we just simply measured the radius of the area around which the magnets turns. With this value you can calculate the circuit of the area – in our case it is equal to the needed distance.

$$O = 2\pi \cdot r$$

O ... circuit (m)

r ... radius (m)

So now we had all the needed value to calculate the speed of the wind. By the way the final time we got was in milliseconds so we had to convert it to seconds.

Anyway, this solution can be a bit tricky because you have to check for the output value of the sensor all the time. That wasn't possible for us because we were using delay function in the program. Delay functions just stop the program for some time and wait on this instruction. If we let it like this it would have insane impact for your calculations because it we would be calculating with a huge accuracy error. To counter this problem I decided to write my own real time operating system which I will describe later.

2.2.3 Measuring the direction of the wind

To calculate the direction of the wind we use a similar method as we used before. We have 8 hall sensors that work same as the sensor used to calculate the speed of the wind. Each of those 8 sensors represents one of the points of the compass.

Direction	
↑	North
↗	Northeast
→	East
↘	Southeast
↓	South
↙	Southwest
←	West
↖	Northwest

When the wind blows it rotates 3 magnets close to each other. Those magnets affect the output value of the 8 sensors. The magnet in the middle has an opposite pole than the other 2 magnets. So as the 3 magnets rotate the one in the middle switches the output values to logic "0" and the other magnets switch the previous and next sensor to logic "1". The magnet which generates logic "0" signalizes the direction of the wind.

2.2.4 Displaying the data

To display the calculated data we use a simple LCD display (20x4). After we measure some data we just send it to the LCD display.

2.2.5 How everything communicates

All the parts are able to communicate with each other because they are connected to the microcontroller.

2.3 Used Technologies

2.3.1 The Real Time Operating System

As I said above, we had to achieve a multitasking effect to successfully calculate the speed of the wind. The first thing that came on my mind when I was thinking how to solve this problem was to use a real time operating system. Unfortunately, there is no official multitasking system for Arduino microcontrollers so I was forced to program it on my own.

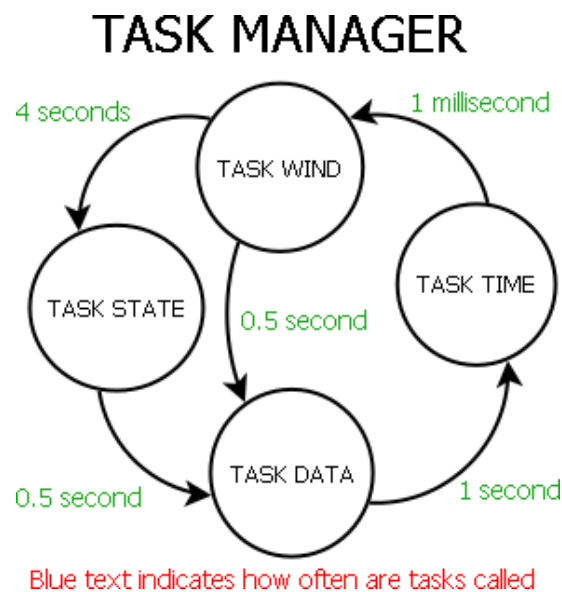
Real multitasking systems use so called process control blocks to manipulate with the running processes, execute them on processor and so. There is no way to program such things when using Arduino so I created a “fake” solution.

My system allows you to create your own tasks and then execute them. Each task contains 3 entries.

Entry	Data type	Description
Function	Pointer to a function	What function the task executes
DelayTime	Unsigned long	How often is the task executed
LastRepeatTime	Unsigned long	The last the task was executed

Then there is a manager object. It has a front of waiting tasks that are ready to be executed. The manager object allows you to add a task to its front or remove it or free it from the memory. It also repeatedly loops through all tasks in the front and asks each of them if they should be executed. Task is executed if the current time – task’s LastRepeatTime is equal or bigger than task’s DelayTime. This means that tasks are executed in a time interval.

Well this is not a real multitasking system at all because all tasks are being executed consecutively and the program has to wait until the entire front is checked before executing the first tasks in the front again. But it was enough for us because we are no longer talking about time inaccuracy in seconds but in microseconds! That means calculations are now very close to reality.



The current version of the system is still very basic and supports only 8 tasks in the front per manager. In our program we are using 4 tasks to make everything work.

1. Task state – this task switches the state of the program. It basically means that the program shows different data on the LCD display.
2. Task data – this task asks for new data from the BME280 sensor.
3. Task time – task to calculate the elapsed time since to beginning of the program.
4. Task wind – this task is used to calculate the speed of the wind.

2.4 Financial Analysis

2.4.1 Used budget

Here is a table of all the parts we purchased on the internet and its price in Euro.

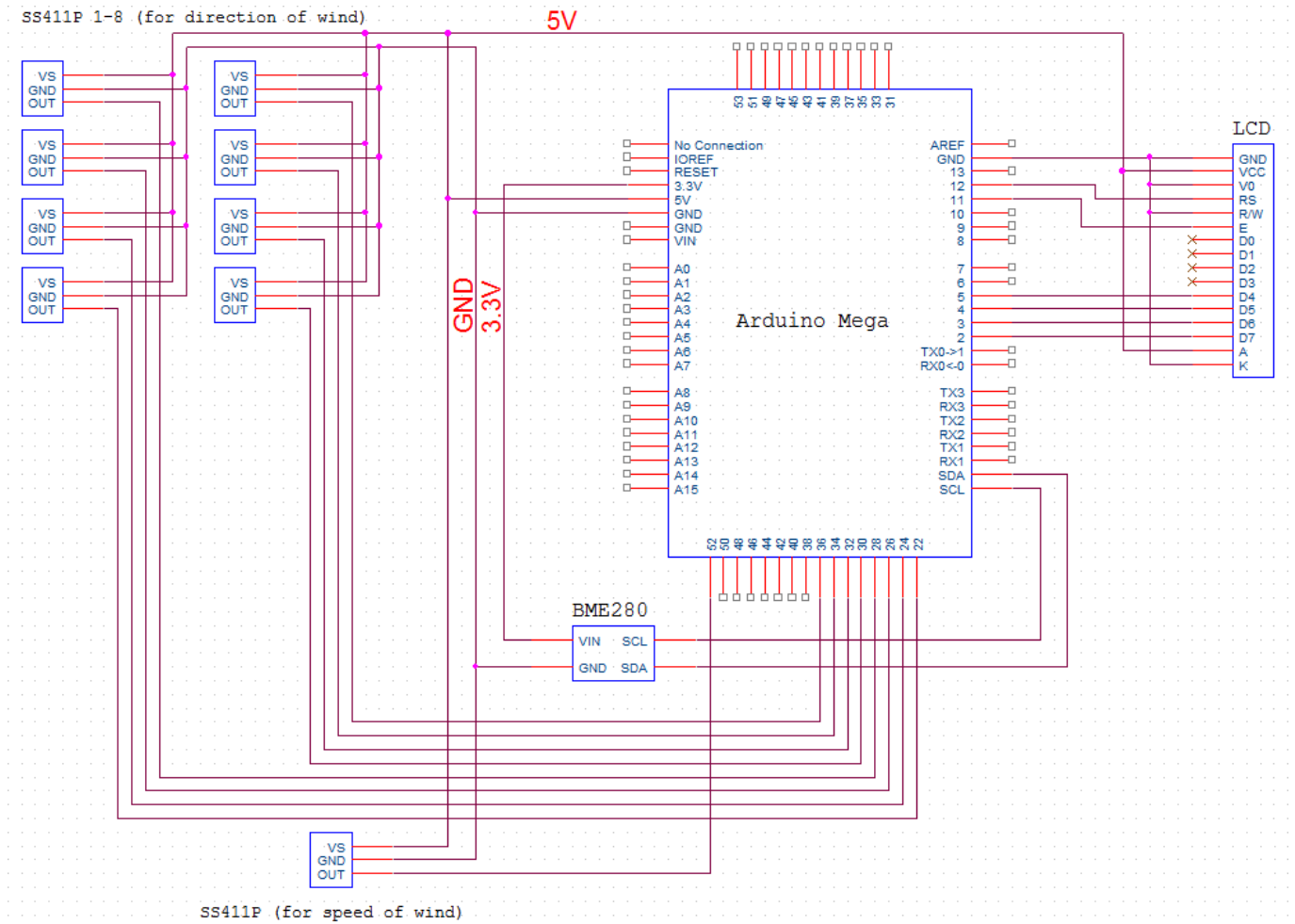
Name	Price
Arduino Mega2560 Clone	15 €
BME280	11 €
9x SS411P	5,75 €
LCD	5,72 €
Wires	2 €
All together	39,47 €

2.5 SWOT analysis

Strenghts	Weaknesses
High precision Fast response Good looking RTOS	Boxes are large Weather station is made of 2 boxes
Opportunities	Threats
Updating the station's sensors and processor Adding more parts Connecting to the internet Saving measured data to a database	Creation of new better sensors

3 Practical part

This is the scheme of our project. It shows how all electronic parts are connected with each other.



3.1 Electronics

Firstly I decided to write a table of all used electronic parts, peripherals and sensors here and then describe each of them.

Name	Type
Arduino Mega2560	Microcontroller
BME280	Sensor
SS411P	Sensor
Arduino LCD 20x4	Display

3.1.1 Arduino Mega2560

This is the microcontroller we used to program and connect all the parts together. We had to use Arduino Mega because it has a lot of digital pins and also much more memory for the code.

Arduino microcontrollers can be programmed in the C/C++ programming language. The creators of this microcontroller have also written a special library that allows you to program the board with the processor. For programming we also used an official IDE for Arduino that helped us with some debugging.

Arduino Mega2560 generates voltage of 5 V and 3.3 V. We use this power other components. Those components have to be grounded also.

Digital pins 22 – 36 and 52 are used to read the output from the hall sensors SS411P.

3.1.2 BME280

BME280 is a digital sensor that is able to read temperature, humidity and pressure. It was created by a company called Bosch. This sensor allows you to communicate with it via the I2C interface or SPI. The speed of the communication can be even 3,4 MHz if you use I2C or 10 MHz when using SPI. Operating range is from -40 °C to +85 °C for temperature, 0 – 100% humidity and 300 – 1100 hPa. Measurement inaccuracy is +/- 1 °C for temperature, +/- 3% for humidity and +/- 1 Pa for pressure. Supply voltage is 1,71 – 3.6 V.

It has got 4 pins. Supply voltage (VIN) which we connected to the 3.3 V pin on the microcontroller board. Then ground (GND). SDA is a Serial Data In / Master Out Slave In pin that the sensor uses to send the data. SCK is a pin for SPI clock.

The sensor achieves high performance, needs very little supply voltage and is optimized to give the best accuracy results. It has extremely fast response time (12 ms in average).

We used a fan-made library to communicate with the sensor and it provided us very simple function to read the data.



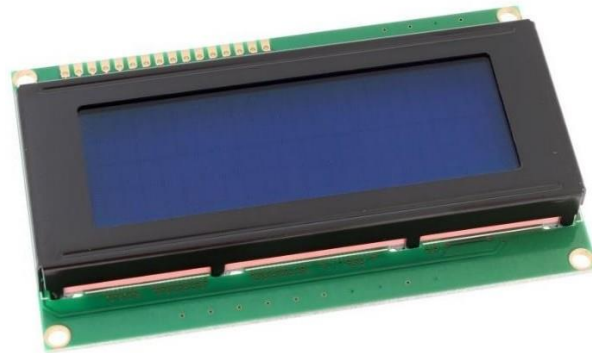
3.1.3 SS411P

Small, versatile digital device, operating on a magnetic field from a magnet. The sensor receives signal, every time when it meets with magnets. They are bipolar and have enhanced sensitivity, so they are compatible to use with smaller and cheaper magnets. Small, leaded, flat TO-92 package (SS411P) often allows for more space on the PCB. Sensitive, bipolar magnetics respond to alternating North and South poles, making these products suited for speed sensing and RPM measurement; Built-in pull-up resistor can easily interface with common electronic circuits without adding external components, helping to reduce total system cost. RoHS-compliant materials meet Directive 002/95/EC.



3.1.4 Arduino LCD 20x4

The LCD we have picked has blue backlight, 4 rows and 20 columns. Its supply voltage is 4 – 5.5 V so our only option was to connect it to the 5 V pin. The LCD is able to print basic ASCII chars which are saved in the memory as well as Chinese chars. It has got 16 pins and 8 of them are for the data bus. That means it can show up to 256 different characters (28). All the functions the LCD is able to do take only a few microseconds per function so it's response time is very fast. It's made of glass, organic sealant, organic fluid and polymer based polarizers. Other pins the LCD has are RS and R/W to tell the LCD what instruction it should be executing now. Then E which is the enable pin and A and K to supply and ground the backlight.



Pin	Level	Function
RS	log. „0“	Enables instruction mode
RS	log. „1“	Enables character mode
R/W	log. „0“	LCD is in the write mode
R/W	log. „1“	LCD is in the read mode

The difference between the instruction and character mode is that when the instruction mode is selected the output of the LCD is interpreted as instructions. That means „clear the display“ or „move the cursor“ and so. In the character mode all output is interpreted as a 8 bit character.

3.2 Programs

To program the weather station, we used the C++ programming language, official and 3rd-party libraries. C++ is a very fast, object-oriented programming language used to program processors word wide.

Because the program contains a lot of values that are represented as number to set something I defined constants for all this values so when I look at the code I know what I'm setting by the value. This is also one of the reasons why we used Arduino Mega2560. Because it gives us much more memory so we can defines constants, enums and so for the needed values.

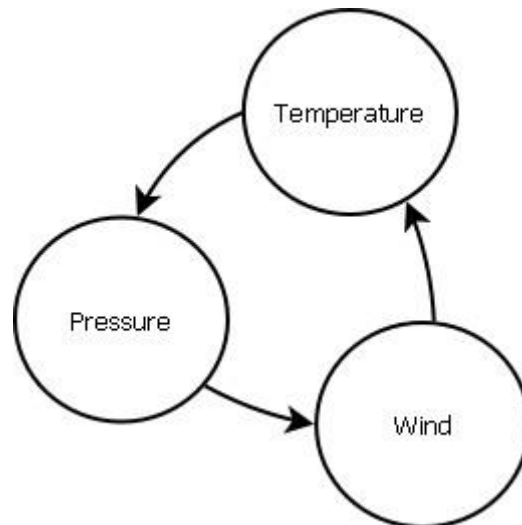
3.2.1 States of the program

Beside tasks I divided the program into states. The program is made of 3 states and in each one of them different data is displayed on the LCD. Here is the table of the states and what is shown:

State	What is shown on the LCD
Temperature	Temperature and humidity
Pressure	Pressure
Wind	The speed and the direction of the wind

I made an enum for the states to prevent using just numbers and make it more readable for me.

The default state of the program is "Temperature". The state of the program changes its value every time the state task is being executed. I use the switch structure to read the state and then switch its value. Another diagram shows the order how the states changes:



Because the state task is called every 4 seconds it means that also the state of the program changes every 4 seconds.

3.2.2 The real time operating system

To program the Real Time Operating System I decided to manage everything via a so-called "Manager" object. It's a C++ class that allows you to manage tasks and do further operations on them. Here is a table of all functions that you can call on an instance of the manager class:

Function	Description
CreateTask	Allocates a new task in memory and returns a pointer to it
AddTask	Adds a tasks to the front of tasks of the manager object
RemoveTask	Removes a task from the front
FreeTask	Frees a task from memory
Run	Runs all the tasks in the front of tasks of the manager object

Manager also contains a front of tasks. Each created task can be added to this front and that run. If the tasks is no longer needed for some time it can be removed from the front or completely freed from the memory. The front is an array of pointers to the task structure. Because is a statically allocated array it has a constant size of 8 - that's the maximal amount of tasks in 1 front.

To represent tasks in the program I used a structure because there are no functions need - only variables. The variables each task has have already been described above.

The system also declares a new data type – a pointer to a function. This is useful for telling a task which function is associated with it.

When the manager executes all the tasks in its front they are actually not called „parallelly“ nor the processor switches between them. They are all executed sequentially but there are no delays at all so it creates an illusion of multitasking. The biggest advantage of this solution is the fact that no critical sections are made through the executing so there are no mutexes, semaphores and other solutions needed.

3.2.3 Tasks

State task – This task has already been described above. It is called every 4 seconds to change the state of the program. It uses the switch structure to determinate in which state the program is and that change it. It also clears entire LCD display.

Data task – This is the task responsible for writing data on the LCD. It's called every 0.5 seconds to make sure that the user gets the most accurate data often but also not to call it all the time. It also works with the state of the program to determinate what data should be print on the LCD. The data it shows on the LCD are being read from the BME280 sensor when the task is active.

Time task – In this task the lasted time from the program's start is calculated. It is called every second because the lowest value it shows on the LCD is seconds. It does some mathematical solutions and calculation to convert millisecond to the hours-minutes-seconds format and then prints the result on the LCD.

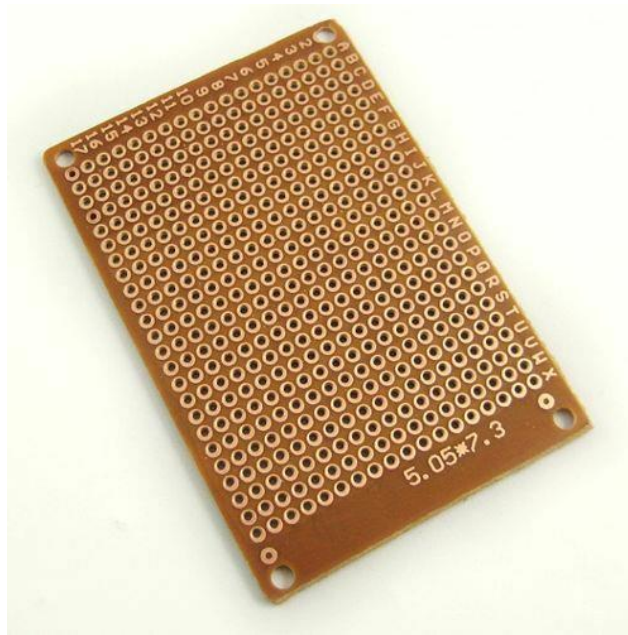
Wind task – This is probably the most important tasks because it is responsible for calculating the speed of the wind. The method I described above is executed here and because the speed of the wind has to be very accurate it's called every 10 milliseconds.

3.3 Mechanical

The weather station consists of two boxes. One box contains the PCB, processor, LCD and sensors and the other second box is used to measure the speed and the direction of the wind and contains the hall sensors. The box with the processor was made by the Czech side and the box to calculated information about wind by the Polish group.

3.3.1 The box with the PCB

To put all the needed parts into the box we have measured the dimensions of it and place the parts so everything works. We had to place the LCD on the microcontroller because there is a hole for the LCD on the roof of the box. We have got a green plate suitable for soldering and we fixed the microcontroller to it. To get the data from the other box we used a huge cable with 12 wires. 9 wires were used for the output data and the rest 2 wires were used as 5V and ground. To use only 1 wire for 5V and ground we had to solder all the ground and supply voltage wires of the hall sensors together and then solder it with the cable. Because the BME280 sensor is connected directly to the microcontroller no special work was needed here. But we had to connect the 5V supply wire and the ground wire from the cable with the LCD and microcontroller. So we soldered a joint on the green plate. The data wires from the box are connected directly to the microcontroller as well. When all the soldering was done we just placed the green plate to the box and fixed it.



3.3.2 The box with the Hall sensors

First I looked for no longer used box for cables. Then I've created the project, how the box will look after editing. I've cut of two holes to set the spinning engines, and one more to lead out the cables, also I have cut out one rectangled hole for the display. To create the arrow that points direction of the wind and those orbal shape parts that are used to measure wind speed I've used (with my teacher help) milling machine to cut them out from plastic slab. Many parts are re-used parts from other things, for example: magnets, spinning engines and box.

3.4 Testing

We have done several tests to know if everything works as it should.

We started as simply as possible – with the BME280 sensor. This sensor is very simple to use but paradoxically we spent very long time with it because we used an Arduino Mega board. This microcontroller is a bit different from the Arduino Uno which the sensor natively supports and all the schemes are made for it. Mega has the registers we needed on different pins and I spent at least 20 minutes solving the problem why we can't get any date. When I connected the sensor correctly everything worked.

Then we decided to connect the LCD. There were no problems here because when the LCD is connected right the backlight starts to glow. We just had to solder it and then print some characters.

The last thing we tested out is if we receive signal from all the hall sensors. I wrote a small program that configures all the pins connected to the hall sensor's output and then it reads values on these pins and displays it on the LDC. Everything worked well and we could be move to further programming.

When everything was connected right and together I just wrote the program and we moved to testing of the final product. Again no problems appeared here.

4 Conclusion

At the end I think our project was successful. We have programmed all the needed behavior, built the model and tested if everything works.

Bořek:

In my opinion our weather station was a success mainly because all my calculations work and also because I was forced to create my own RTOS which improved my programming skills a lot. But now I see that the project could be much better if I added more parts. One of the things I would like to add is a time sensor that allows you to read the current time or even date. Another thing that could be changed is the LCD. I wanted to use the mobile-like screen to make the project look much more professional and the display text would also look really nice with colors, fonts and different text size. The last thing I thought would be nice to implement is a Wi-Fi sensor. Because I know computer languages like PHP, HTML, CSS and a bit of JavaScript we could literally make a website and display the measured data there. Also some kind of a database could be set up to be able to visit previous measurements.

In the future I would I could add this features to the weather station but everything depends on the fact if there is enough place in the box. Also I don't exactly know how the Wi-Fi on the Arduino works so we might need a keyboard to control it.

5 References

Information about the LCD display:

https://www.hwkitchen.cz/user/related_files/lcd-displej-20x4-modry-s-podsvetlenim-ds-c2004a3-btsswe-naa.pdf

Information about the hall sensors:

https://sensing.honeywell.com/index.php?ci_id=45288

Information about the BME280:

<https://navody.arduino-shop.cz/navody-k-produktum/senzor-bme280-mereni-teploty-relativni-vlhkosti-a-barometrickeho-tlaku.html>

RTOS:

https://en.wikipedia.org/wiki/Real-time_operating_system

Everything about Arduino, pins, versions:

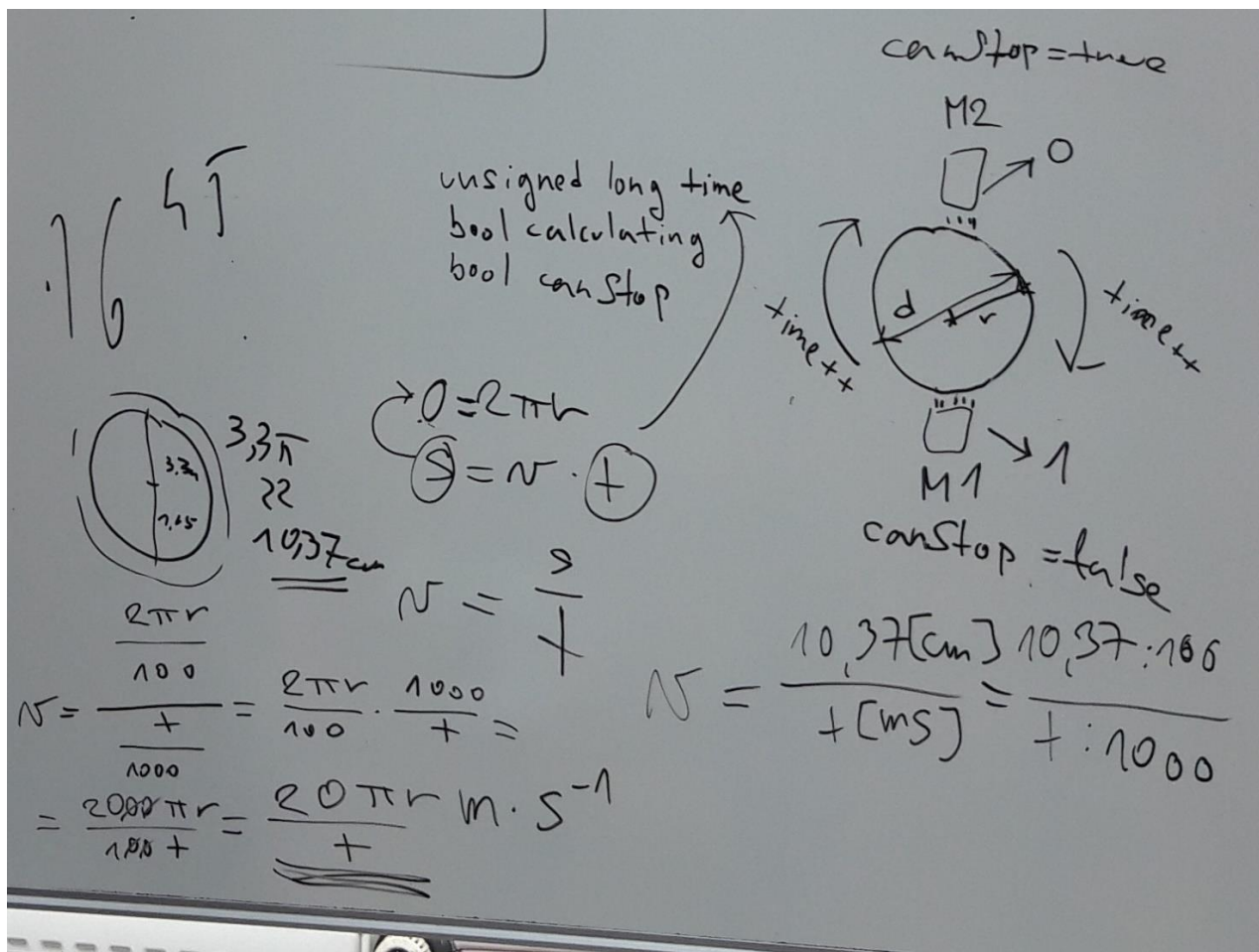
<https://www.arduino.cc/>

Info about other weather stations in the business:

https://www.google.com/search?q=weather+station&client=firefox-b-d&source=lnms&tbm=shop&sa=X&ved=0ahUKEwirkDz_7HhAhUbTRUIHeiJD1EQ_AUIDygC&biw=946&bih=958

6 Attachments

This is my idea which I used to calculate the speed of the wind:





Střední průmyslová škola elektrotechnická, Havířov, CZECH REPUBLIC
Miskolci SZC Kandó Kálmán informatikai szakgimnáziuma, Miskolci, HUNGARY

WEATHER FORECAST STATION WITH ARDUINO



Co-funded by the
Erasmus+ Programme
of the European Union



Co-funded by the
Erasmus+ Programme
of the European Union



Weather forecast station with Arduino

Hardware and software documentation

*This project was made under the cooperation of Erasmus+ KA2 P.L.E.L.
program between the Hungarian and Czech partners*

2019.

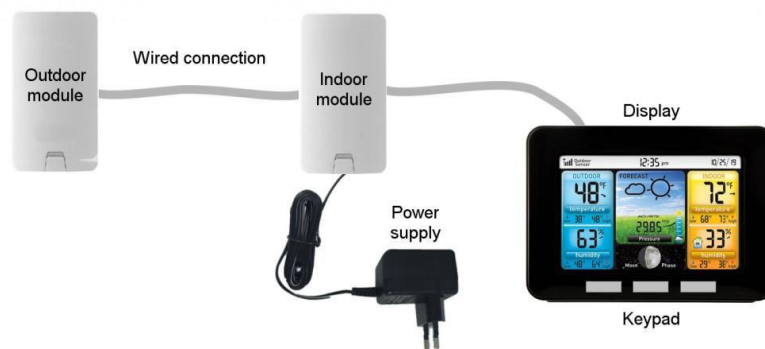
Table of contents

1. Implementation of the hardware.....	3
1.1. Prior description	3
1.2. Concept of the system	3
2. Maps of the hardware	4
3. Components	9
3.1. List of the components	9
3.2. Pictures of the components	10
4. Implementation of the hardware	11
4.1. Final assembly design.....	11
5. Implementation of the software	12
5.1. Prior description	12
5.2. Working process of the system	12
6. Operating process of the programs	13
6.1. Flowchart of the programs	13
7. User interface of the indoor unit	15
8. Appendix	17
9. Final statement	17

1. Implementation of the hardware

1.1. Prior description

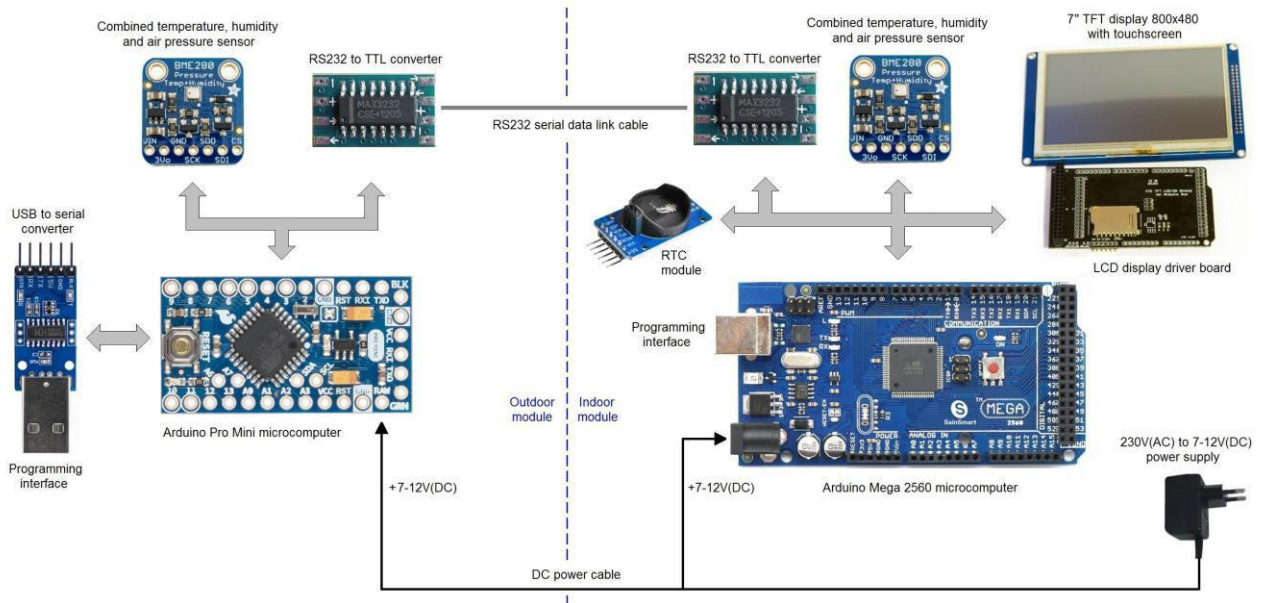
Weather station for measuring and displaying indoor and outdoor climate conditions. The task of the weather station is to measure and display the outside and indoor climatic factors of a building. The station measures the ambient temperature, the air pressure and the humidity outside the building, as well as the room temperature and the humidity inside the room. The measure values are displayed on a single graphical interface on a monitor screen. In addition, it stores the sampled values at specified times, based on it creates retrospective change graphs and weather forecasts.



1.2. Concept of the system

The format of execution is based on a microcomputer-supported, external and internal meeting unit, which is equipped with a display device and a control panel (it can be conveniently integrated). The outdoor unit, situated outside the building, measures climatic data using sensors with the help of microcomputer. The data is transmitted through an interface to the indoor unit, situated inside the building. The indoor unit processes and displays the received data together with the climatic data, measured by itself. The display mode can be modified by the help of pushbuttons. We suggest wired connection to the communication through the interface: in this case the outdoor unit sends data to the indoor unit through RS232C interface.

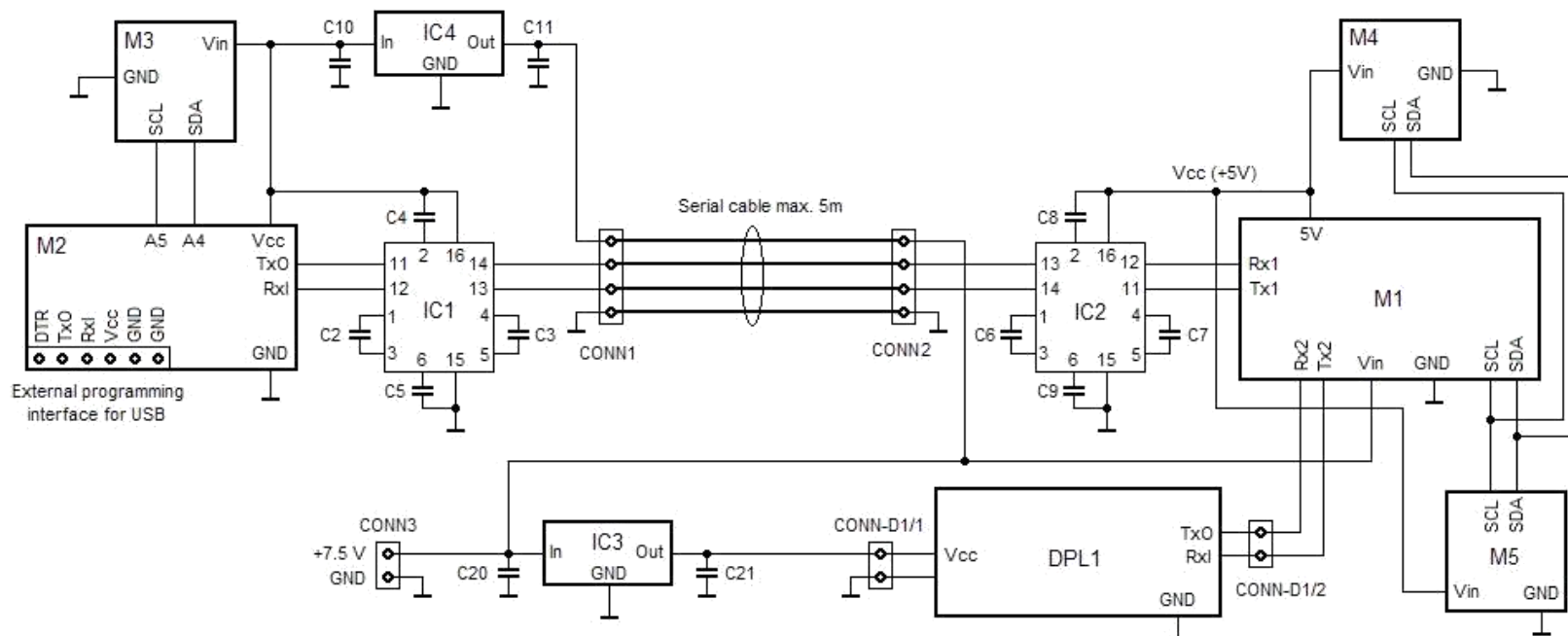
The following diagram illustrates the structure of the system.




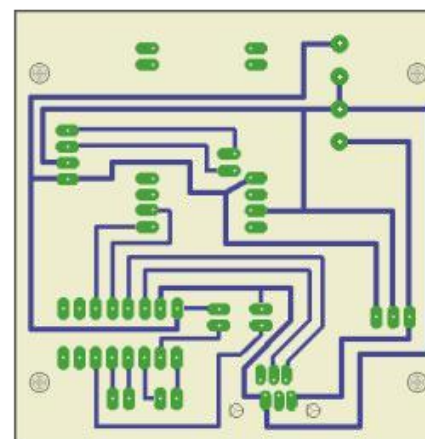
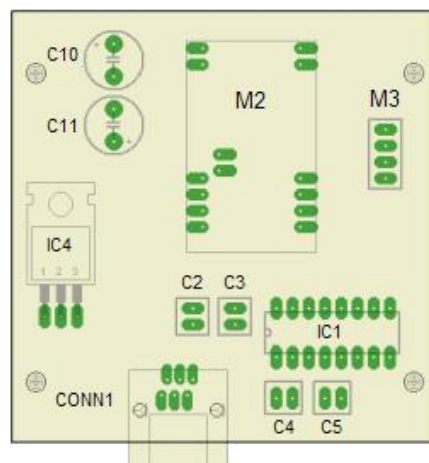
2. Maps of the hardware

The maps of the hardware can be seen on the following pages.

It contains the hardware wiring circuit diagram, and the maps of the printed circuit boards of the indoor and outdoor units.



Company:			Title: Weather forecast station with Arduino		
 <p>MSzC Kandó Kálmán Informatikai Szakgimnáziuma 3525 Miskolc, Palóczy u. 3.</p>			Subtitle: Wiring schematic		
			Sheet: 1/4	Rev: 1.0	Date: 2018.11.04.
			Drawn by: Vigh Gergő		



Company:



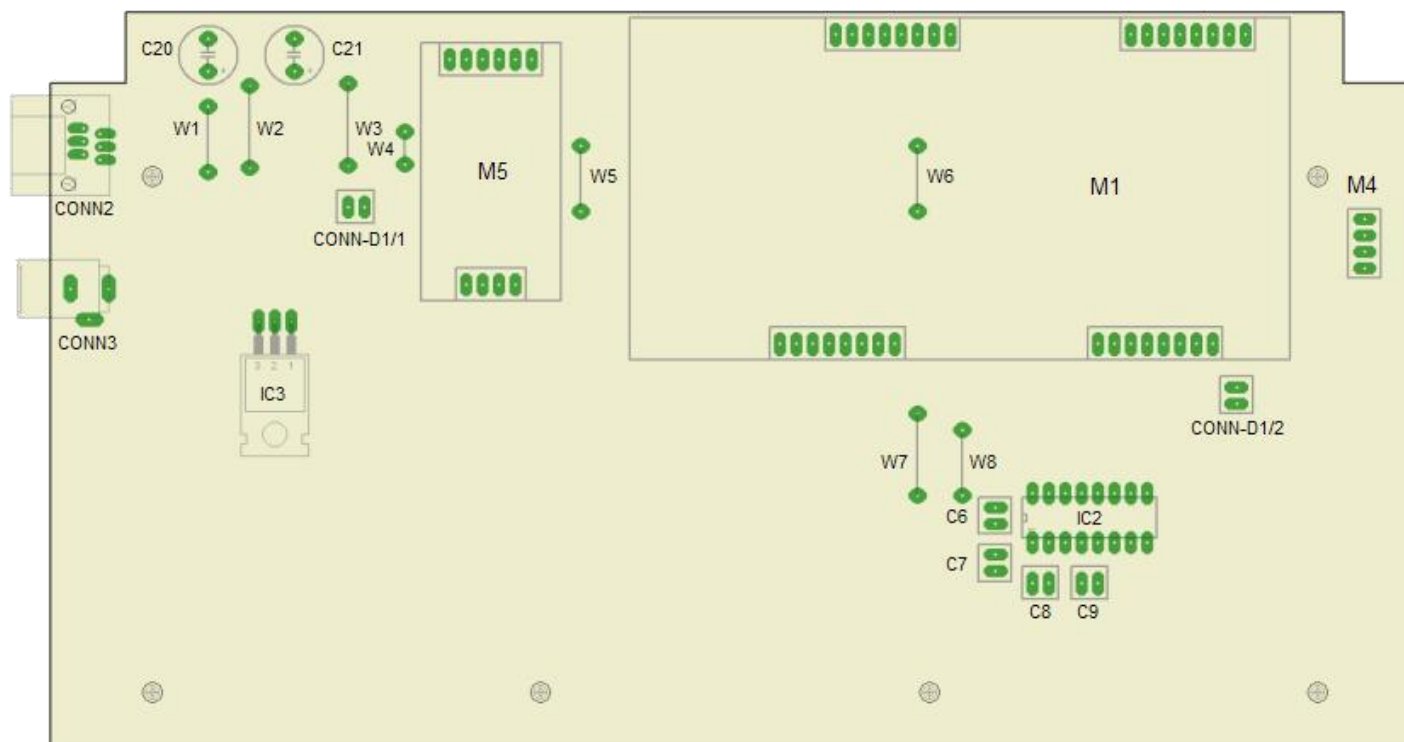
MSzC Kandó Kálmán
Informatikai Szakgimnáziuma
3525 Miskolc, Palóczy u. 3.

Title: Weather forecast station with Arduino

Subtitle: PCB layout of outdoor module

Sheet: 2/4 Rev: 1.0 Date: 2018.11.04.

Drawn by: Vigh Gergő



Company:



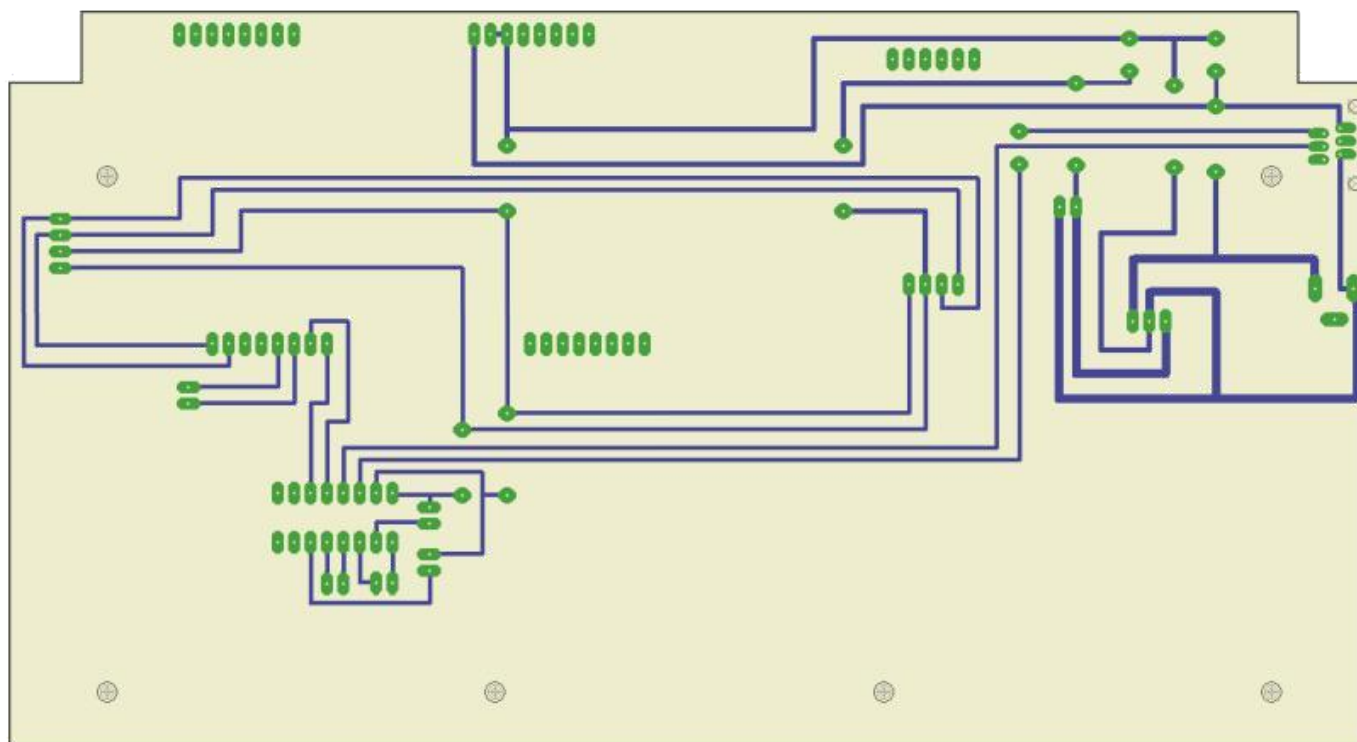
MSzC Kandó Kálmán
Informatikai Szakgimnáziuma
3525 Miskolc, Palóczy u. 3.

Title: Weather forecast station with Arduino

Subtitle: PCB layout of indoor module
Component layout

Sheet: 3/4 Rev: 1.0 Date: 2018.11.04.

Drawn by: Vigh Gergő



Company:



MSzC Kandó Kálmán
Informatikai Szakgimnáziuma
3525 Miskolc, Palóczy u. 3.

Title: Weather forecast station with Arduino

Subtitle: PCB layout of indoor module
Routing layout

Sheet: 4/4 Rev: 1.0 Date: 2018.11.04.

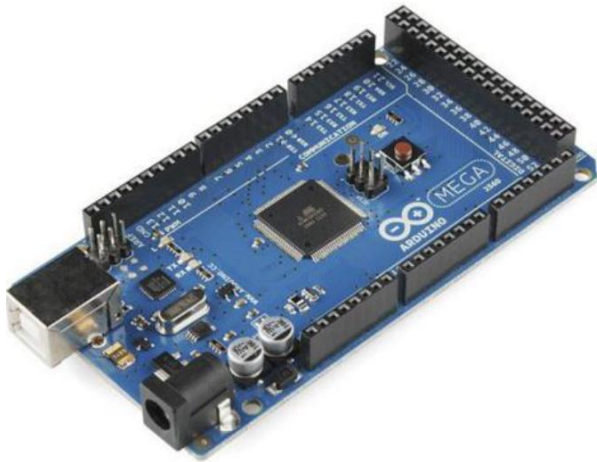
Drawn by: Vígh Gergő

3. Components

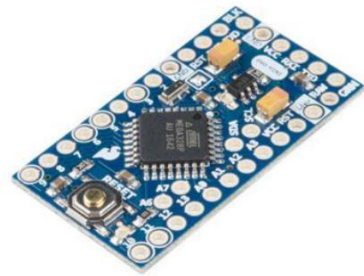
3.1. List of the components

No.	Device name	Type / Value
1.	M1	Arduino Mega 2560 microcomputer
2.	M2	Arduino Pro Mini microcomputer
3.	M3, M4	BME 280 combined sensor module
4.	M5	DS 3231 real time clock module
5.	IC1, IC2	MAX 232 (TTL-to-RS232 interface)
6.	IC3	L78S05 (5V voltage regulator)
7.	IC4	LM7805 (5V voltage regulator)
8.	DPL1	Nextion 8048T070 touchscreen
9.	C10, C11	100 μ F/10V capacitor
10.	C20, C21	1000 μ F/10V capacitor
11.	C2, C3, C4, C5, C6, C7, C8 C9	1 μ F/10V capacitor
12.	CONN1, CONN2	RJ12 connector
13.	CONN3	DC Jack 2,1mm PCB connector
14.	CONN-D1/1	Pinheader connector, 1x2 pins
15.	CONN-D1/2	Pinheader connector, 1x2 pins
16.		
17.		
18.		
19.		
20.		

3.2. Pictures of the components



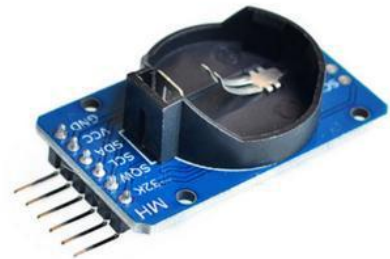
Arduino Mega 2560 microcomputer



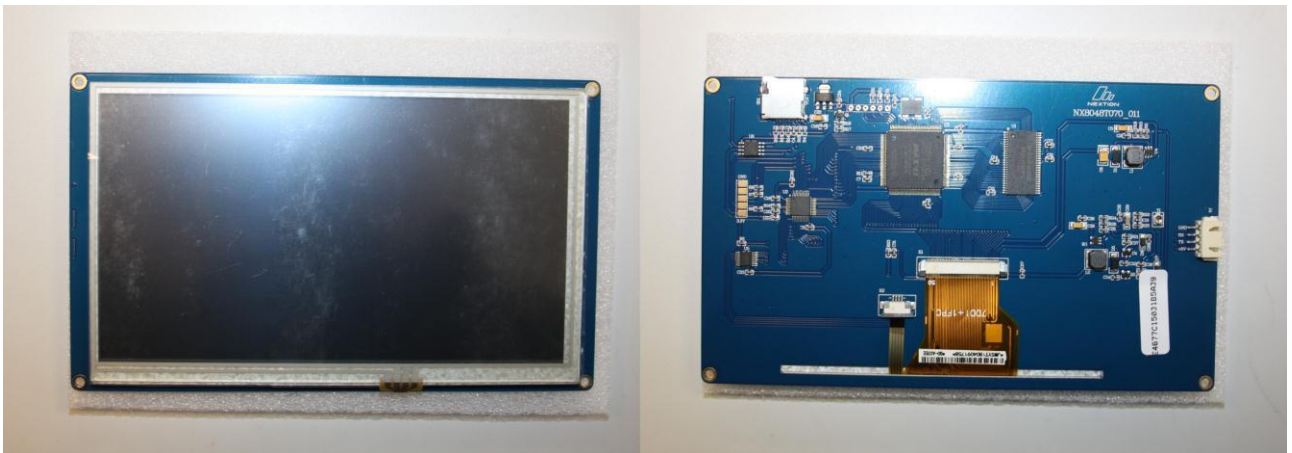
Arduino Pro Mini microcomputer



BME 280 combined sensor module



DS 3231 real time clock module



Nextion 8048T070 touchscreen



MAX 232

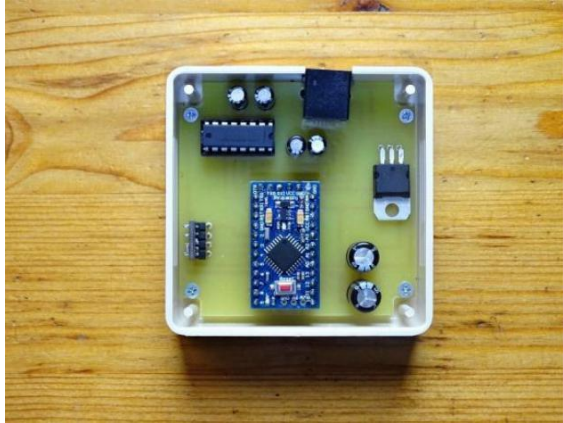


LM 7805

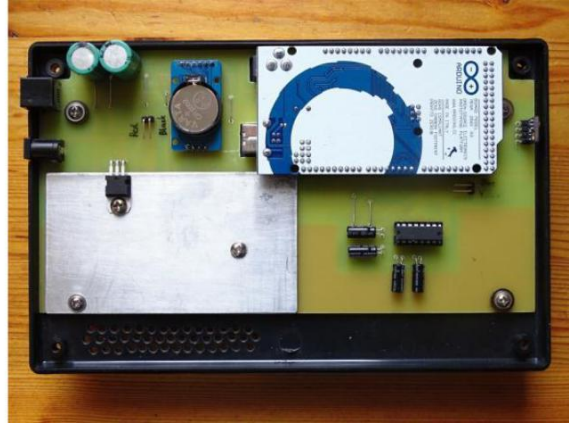
4. Implementation of the hardware

4.1. Final assembly design

The final design can be seen on the pictures bellow.



Inside view of the outdoor unit



Inside view of the indoor unit



The assembled entire system

5. Implementation of the software

5.1. Prior description

The implemented system with the supported features by the following services:

- Display indoor and outdoor climate conditions, such as temperature, air pressure and humidity
- Store climate data in EEPROM memory of the microcomputer
- Create weather forecast based on the changing of the air pressure
- Display statistic of the weather changing based on stored data

5.2. Working process of the system

After switching on the power supply, the Arduino microcomputers boot up, and they go to stand-by state. The Nextion touchscreen also boots up, and displays the starting screen. At this time the system is ready to work. The system has three operational modes: “measuring mode”, “setting mode” and “statistic mode”.

a) “Measuring mode”

In this mode the microcomputer of the outdoor and indoor unit measures the three climate condition data. The outdoor microcomputer transmits data to the indoor microcomputer, and it displays both of data on the screen continuously in time. The indoor microcomputer also creates weather forecast using the air pressure, and displays it on the screen by a gauge indicator. The actual time displays too.

b) “Setting mode”

In this mode the indoor microcomputer allows to set some settings, such as the date/time or the value of the default normal local air pressure. The default air pressure is required to calculate the weather forecast. It depends on altitude of then location place above sea level.

c) “Statistic mode”

In this mode the indoor microcomputer allows to display climate condition data 14 days back. We can choose the information that we want to display: temperature, air pressure or humidity.

6. Operating process of the programs

Once the devices are turned on, the “setup” part of the software runs, which is responsible for initializing and launching each software components. Then, the “loop” section is executed continuously.

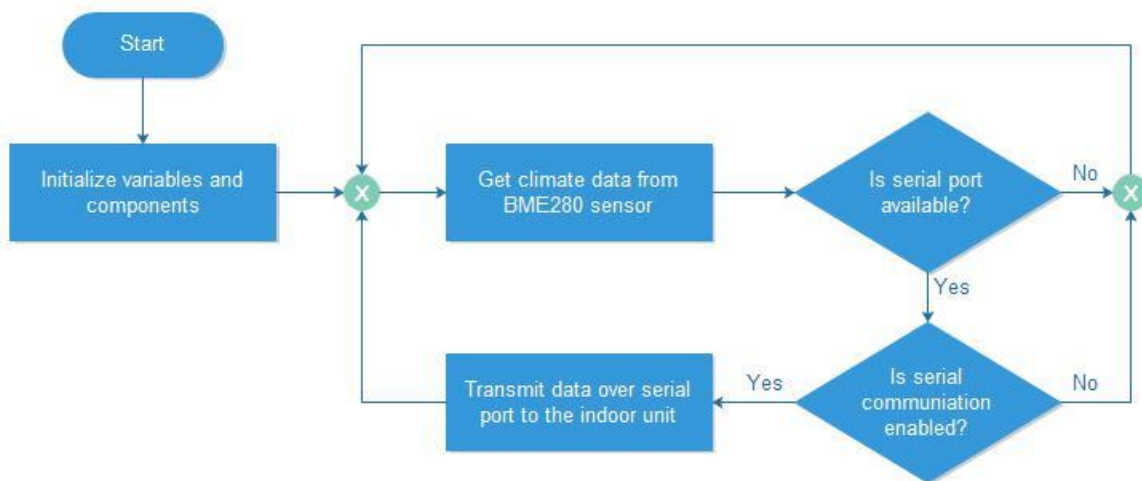
Meanwhile, the program of the outdoor unit monitors the BME280 sensor continuously and sends data to the indoor unit every second through the RS232C interface.

The program of the indoor unit also monitors its own BME280 sensor continuously and receives data from the outdoor unit, and sends them to the Nextion touchscreen to display. At the same time it stores these data in EEPROM memory, too. The touchscreen has an own microcomputer, which creates the user interface screen, and displays the climate data on it.

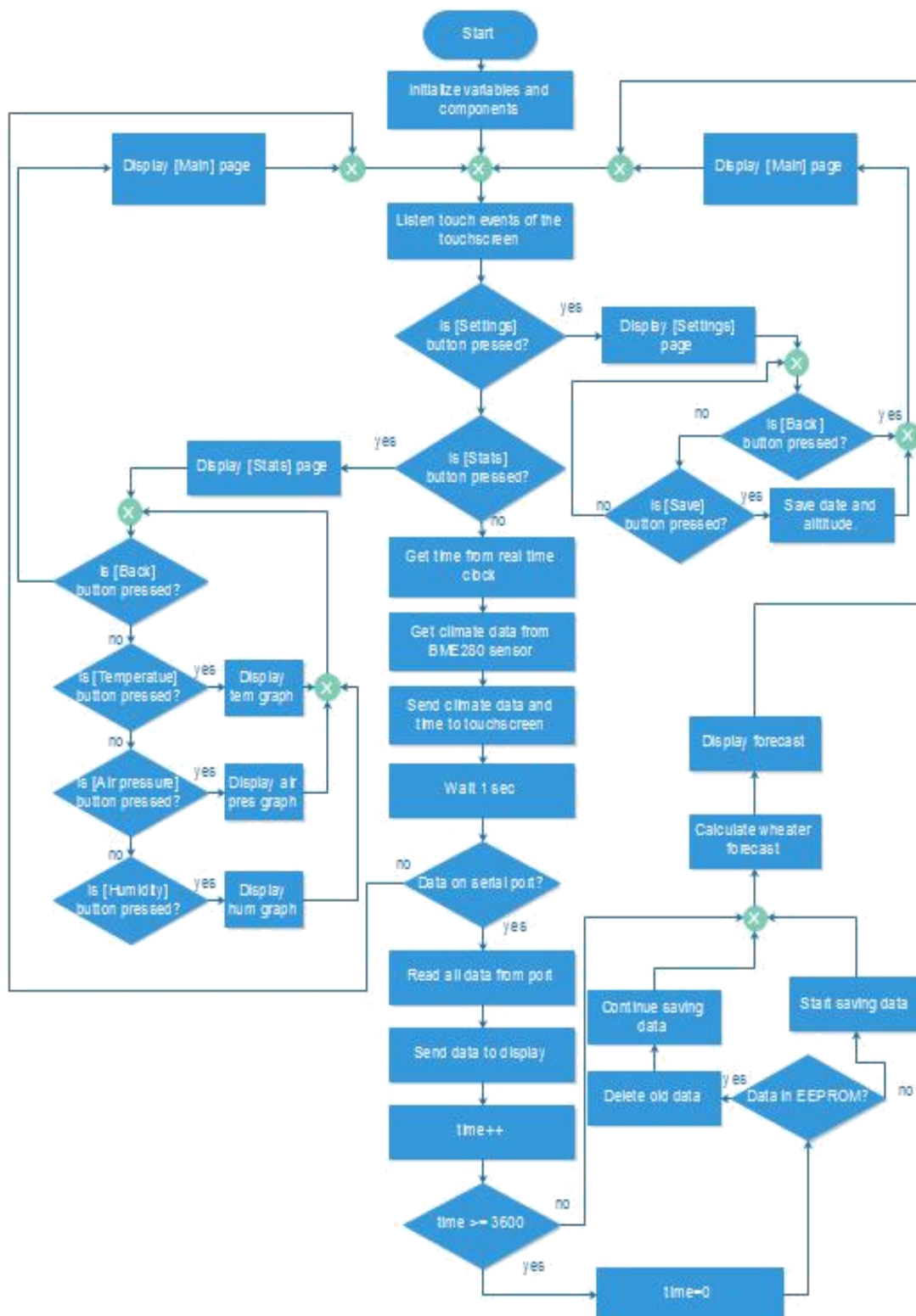
If the system runs in “Setting mode” the microcomputer of indoor unit stops the receiving data from the outdoor unit, and turns the touchscreen into the setting screen. At this time, the Nextion microcomputer allows to set time, date and value of the default normal local air pressure.

If the system runs in “Statistic mode” the microcomputer of indoor unit stops the receiving data from the outdoor unit, and turns the touchscreen into statistics screen. At this time, the microcomputer of indoor unit is reading data, which is stored in EEPROM memory, and sends them to the Nextion touchscreen to display on a graph. In this case we can choose, which data to display: temperature, air pressure or humidity.

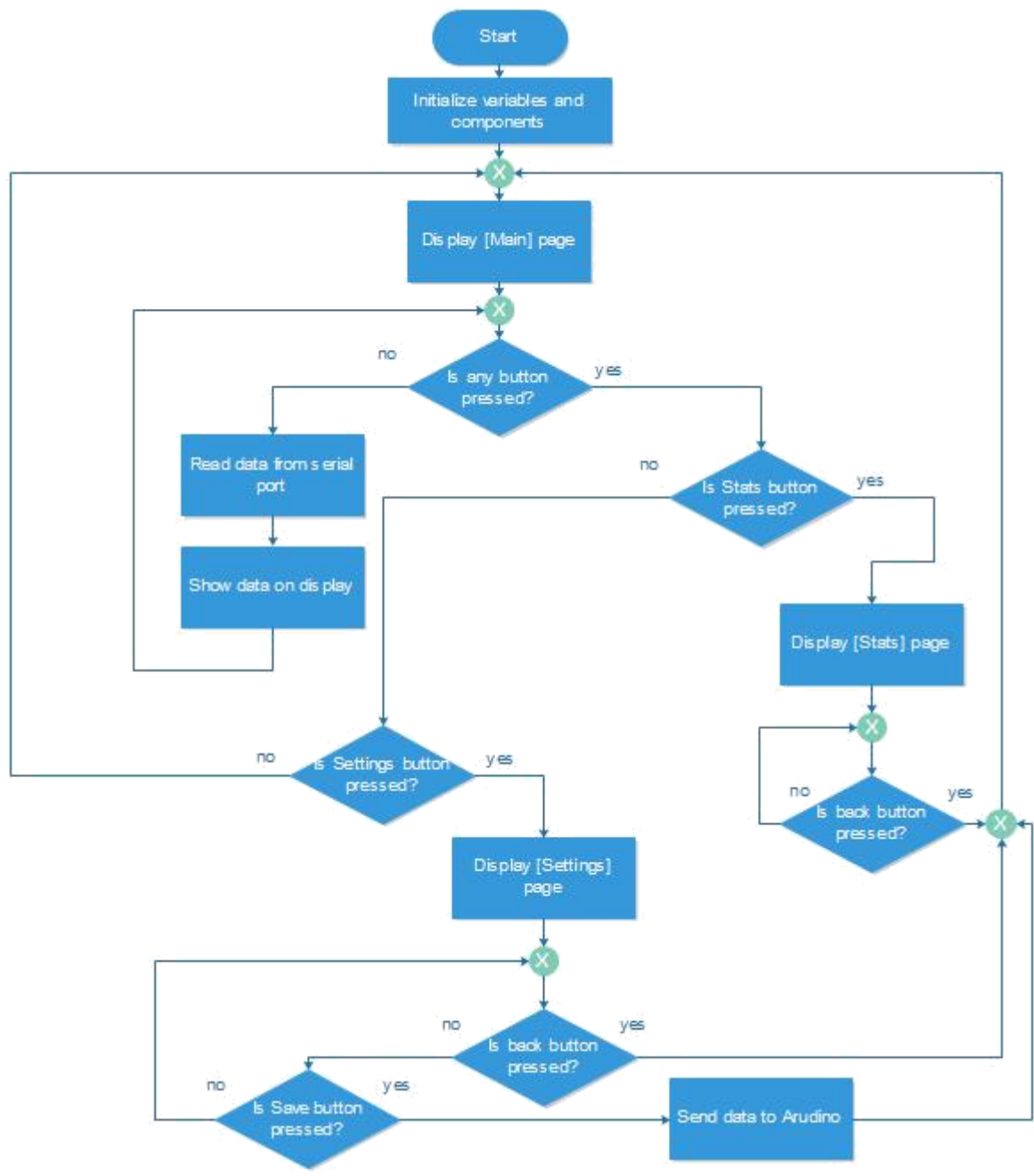
6.1. Flowchart of the programs



Flowchart of the software of the outdoor unit (Arduino Pro Mini)



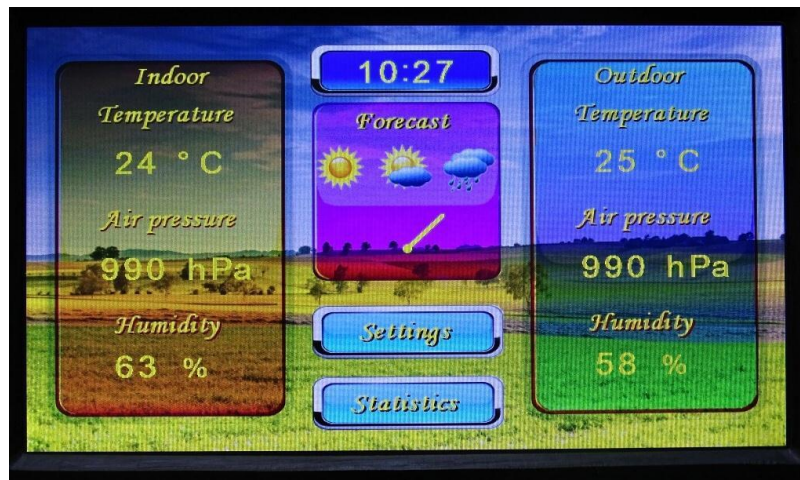
Flowchart of the software of the indoor unit (Arduino Mega 2560)



Flowchart of the software of the touchscreen (Nextion 8048T070)

7. User interface of the indoor unit

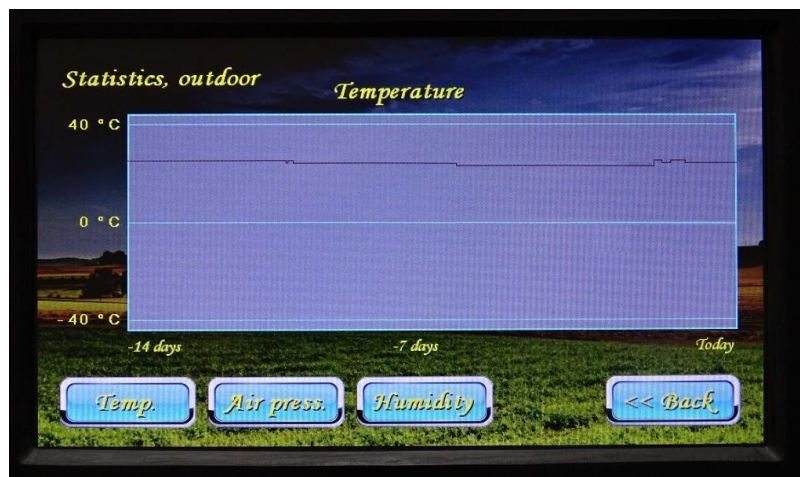
We designed a GUI, which handles the functions of the system, such as settings, and display statistics. This interface has three different screens, which allows to handling the different functions. These screens can be seen on the following pictures.



Main screen



Settings screen



Statistics screen

8. Appendix

The following directories can be found on the attached CD disk:

\Sketch: directory of the software for Arduino and Nextion, and the used Arduino libraries

\Documents: directory of the documents

\Pictures: directory of the pictures of the working system

9. Final statement

This project was made under the cooperation of Erasmus+ KA2 P.L.E.L. program between the Hungarian and Czech partners.

The final hardware and the test software were designed and created in the vocational school:
Miskolci SZC Kandó Kálmán Informatikai Szakgimnáziuma, Miskolc, Hungary.

Hungarian team:

Student: Vigh Gergő

Mentor: Kósa Tamás

The proposal hardware and the final software was designed and created in the vocational school:
Střední průmyslová škola elektrotechnická, Havířov, Czech Republic.

Czech team:

Student: Timotej Plachta

Mentor: Ing. Zuzana Paučková

This document consists of 17 pages.



Zespół Szkół Technicznych, Mikołów, POLAND
Stredná priemyselná škola elektrotechnická, Košice, SLOVAKIA

WORD CLOCK



Co-funded by the
Erasmus+ Programme
of the European Union

Table of contents

Motivation	4
1 Introduction.....	1
2 Theoretical part	2
2.1 Development platforms.....	2
2.1.1 Arduino	2
2.1.2 RTC DS3231.....	4
2.1.3 Eterneth shield ESP2866.....	4
2.2 Market analysis.....	5
2.3 Technical analysis	5
2.4 Used Technologies.....	5
3 Practical part.....	6
3.1 Electronics	8
3.2 Programs.....	10
3.3 Testing	12
4 Conclusion	13
5 References	14
6 Attachments	15

Motivation

In this short part I will explain to you why did I chose this project. Thanks to this project, I could improve my technical and language experts as well as electronic skills. At the same time, I wanted to try working on an international project by not doing any project with a partner from another country. I chose Word clock from the list of projects we were offered. what immediately caught my attention was the name of the project. Of course, these clocks are controlled by the Arduino, with whom I have not yet met. As the Arduino is taught in our fourth year at our school, I think it is too short for those who want to make a living. So I decided on this project to get in touch with the Arduino a little earlier. this way I would like to thank Mr Michal Copko for his help and useful advice on the project.

1 Introduction

Word clock is type of clock, which shows time by words located on the boards. The task of these hours is to show the real time in words spelled out on both boards. Boards consist of two main parts: „Hardware” and „Software”.

Software part, which was assigned for slovakians, consists of an Arduino Mega2560, Real Time Clock (RTC) DS3231, Ethernet shield ESP8266, IDE cable and several Jumper wires for the Arduino. Hardware part was the task for polish student and in this part is incorporated making design for boards in program Corel. The word board has 3 layers - first one is from wood, second is transparent glass and the last one is black using only for showing words . On the back-side are LEDs connected with 510Ω resistors, which are using one common ground connected to Arduino Mega2560. To solve the problem with bunch of jumper wires between the boards we changed them for one IDE cable divided into several parts and connected with single word on the board.

The reason why we have choosen this project is that it sounds like the best to do. We were so interested about it and we had a plan how to do it from the begining.

2 Theoretical part

At the beginning we had to look up, which technological parts we will use in our project (some modules, microprocessors etc.). We searched a lot of websites to find out true informations about each parts.

2.1 Development platforms

2.1.1 Arduino

We started with a microprocessor Arduino, with whom we have not come into contact yet. On the internet, we came across an e-book about what an Arduino is and how does it works. The book was named Getting started with Arduino. Here we found all we wanted to know about it and then we could start with programming. However, nobody of us two did not have a practice in Arduino programme, we asked Mr. Copko to explain and show us, how to write programme for our Word Clock.

Arduino is an open source physical computing platform based on a simple I / O board and development environment that implements the processing language. Arduino can be used to develop stand-alone interactive objects, or can be attached to software on your computer (such as Flash, Processing, Max/MPS). The boards may be assembled manually or purchased pre-assembled; Open source IDEs can be downloaded free of charge.

From wide range of Arduino models, firstly we have chosen Uno. Because of few pins on Arduino Uno we had to change it to Arduino Mega2560, which is equipped with more free pins where we are able to connect all words from the clocks, one similar ground and modules we needed.

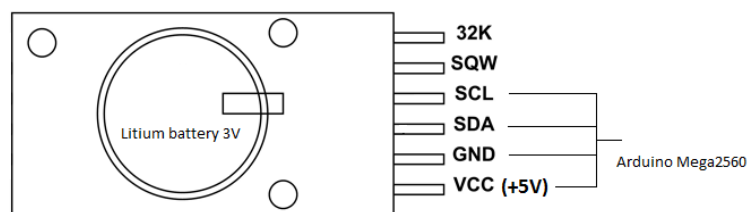
Arduino Uno

Arduino Uno is a direct successor to the major developmental sector, which began with the first Arduino serial port instead of USB, continuing through Arduino Extreme through NG to today's Uno. On the Arduino board is located ATmega328 processor and classic USB. Two other special boards have evolved from this line. Arduino Ethernet, as well as Uno, except that the Ethernet port for the network connection is located on the board instead of the USB port. The second special board is Arduino Bluetooth. As its name suggests, a Bluetooth module for wireless communication is built into the board instead of USB.

2.1.2 RTC DS3231

The DS3231 is a low-cost, extremely accurate I2C realtime clock (RTC) with an integrated temperature-compensated crystal oscillator (TCXO) and crystal. The device incorporates a battery input, and maintains accurate timekeeping when main power to the device is interrupted. The integration of the crystal resonator enhances the long-term accuracy of the device as well as reduces the piece-part count in a manufacturing line. The DS3231 is available in commercial and industrial temperature ranges, and is offered in a 16-pin, 300-mil SO package. The RTC maintains seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. (Dallas Semiconductor Corporation, 2006)

We used RTC DS3231 for determining real time to our boards. It works on pins SCL,SDA,VCC,GND which is connected with an Arduino. However pins 32K and 32K are not connected with anything, they are not needed (free pins).



0.3 RTC DS3231

2.1.3 Eterneth shield ESP2866

The NodeMCU is an interesting platform - it integrates the ESP2866 Wi-Fi chip, which takes care of the computing and wifi part of the device. The board went through three hardware versions with two versions of the ESP 2866 chip during development. ESP 2866 can also be called System on a Chip (Soc). It offers everything a common developer needs from a wide range of programming options through hardware to CPU overclocking. . (Editors Root.cz, 2017)



2.2 Market analysis

On the market there are verbal words that determine the real time by illuminating the words on the display or by illuminating the individual words on the board made up of LED strips as is the case. This solution is not just for fun, but the word clock is also a design accessory. Therefore, we think that our solution to the clock could find its candidates, for whom this idea is certainly interesting and which might even improve it.

Although, Americans are the most often involved in the word-lesson project, and they mainly create it in the English language, we decided to do it in two languages and to make nothing more original, we made word clock in the unconventional combination of Slovak and Polish. We see similarities in our project with others that we have also decided to choose a 12-hour time format in order to avoid double project work in terms of programming, creating larger board sizes etc.

2.3 Technical analysis

Although our project does not contain many parts or modules, it works better than we expected. Parts, which we applied, can be seen in the simple schematic created by us two.

All modules are interconnected to Arduino Mega as well as polish and slovak boards. For saving wires we connected with Arduino only slovak board, but these two boards are linked to each other. 23+23 word and one similar GND communicate with Arduino bycable. The principle of this circuit is that the arduino acquires real-time data from the RTC module. They are processed in the arduino and based on the program I send a signal to the Arduino pins, which light up the words or 4 LEDs in the corners. ESP 8266 module is connected to internet by Nodemcu and two pins from this module are interconnected, specifically pins TX and RX. Using these pins arduino gets time from Nodemcu and the arduino processes it.

2.4 Used Technologies

HARDWARE	USAGE
IDE cable	We used this cable for saving wires
soldering iron	soldering LEDs with resistors, wires
hot glue stick	fasten boards together with hot glue and 4 threaded rods
CNC machine	holes for LEDs in boards, small holes for words on black plastic
double-sided tape	fasten wood, black plastic and matt plaxiglass
other tools	for details on our project

SOFTWARE	USAGE
Arduino IDE	Project programmes
COREL Draw	first sketch of boards
AutoCAD	3D scetch of boards

3 Practical part

At the beginning we had a problem, when we did not know anything about working with an Arduino and writing programmes in Arduino IDE. Fortunately we found method how to solve this problem. We asked our teachers, who gave us so many helpful and useful informations. By the time we found out that our Arduino Uno is not equipped with enough pins, so we had to reach for bigger type – Arduino Mega2560. However Uno is equipped with 14 digital pins together with pin number 0, Arduino Mega2560 dominates with 53 free digital pins.

We continued at work with making design of boards in Corel. But before we started doing a sketch in Corel, we had had to make a plan of those boards (how many words do we need, where it will be on the boards, how to solve problem with minutes from 1 to 4 and so on). As you can see in the picture, in the first “part” of the boards are words which show hours from one to twelve. In second “part” are minutes, but not from one to fifty nine. Marcin got an idea, how to make it easier. On the boards are only “big” minutes like five, ten, fifteen, twenty, thirty, fourty, fifty and 4 LEDs in every corner (they show minutes from one to four), while if the real time is for example 10:38, on the boards will light up words ten (hours), thirty, five and 3 LEDs in the corners. Sequentially we made 3D sketch in AutoCAD due to holes in boards for LEDs. By the LEDs have about 2 centimeters we had to use a board at least 5 centimeters thick or more.



Actually we used AutoCad to make sure how thick will the boards be. Marcin made another sketch of black plastic plate with holes for words as you can see in the picture. After we had all sketches from Corel and AutoCad, we would be able to create boards using CNC machine. However only one of us have CNC machine at school, boards were made by Marcin in Poland. And that was their main assignment. As soon as we had finished making boards we started with electronics. Specifically with soldering LEDs with resistors. We used blue LEDs with opening voltage 0,6V and resistors 510Ω. After we had created all “LED stripes”, we checked everyone word if it is correct. That was all we needed to do from technical part.

Now we could start with informatics part (connecting Arduino with all modules, write a programme for this project and so on). Firstly we made easier connection of Arduino and

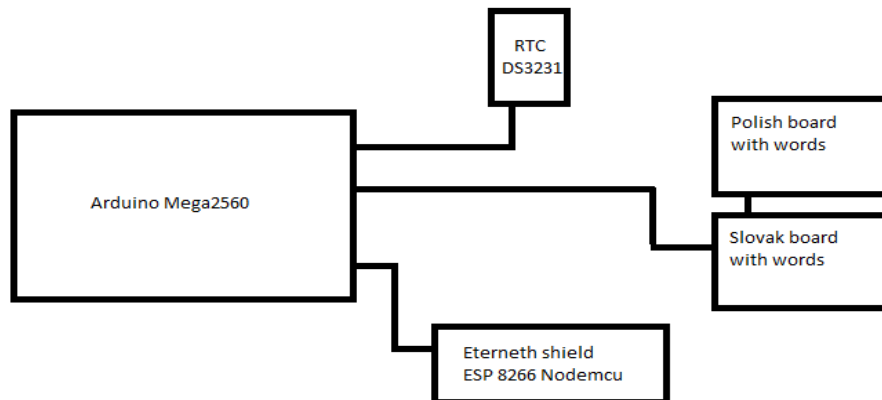
all LED “stripes” – everyone word. We used here IDE cable. On the Slovaks board we made “paths” of this IDE cable, than we split it and made “groups” of wires which are interconnected with every single word. For example, at the top of the boards are words one, two, three and two LEDs. They are linked with “group” of 5 wires from this cable to Arduino. All pluses from LED stripes and one similar GND are attached with Arduino by these way. Lately we pasted this divided cable to wooden boards with hot glue due to not moving with this cable. Thereby wires are not sensitive to hot glue and do not interfere with the electrical connection, we could use hot glue above the wires at the board. For making sure we have checked all words again if there is not anything bad. Meanwhile we were checking this cable connections, one of us cut jumpers, which were linked between these boards. We used jumpers of approx 15 centimeters long, because when we will cut the jumpers too short the connection could break up and than we soldered them – every polish word (only one common plus from this stripe) with slovak.

Subsequently we had to decide how to make a box of these two boards. More precisely what material to use between the boards to hold the whole project together. Here we used mentioned threaded rods, from one 1 meter long we made 4 pieces 15 centimeters long with radius 6. After that we had had these rods, we made 4 holes in every corner but somewhere where it is not located any word or one of the LEDs in the corners. The holes were made by CNC machine and we made it because we had put here the rods and stick by hot glue because of stability. If we put the rods on the board and than stick with hot glue, it will fall down. Later we made a “test of stability” when we tried if the boards are too strong same as the rods. The last thing we had to do was to find something to hide all wires inside the box, more precisely between the boards. Here we used lino, due to its elasticity. This lino is pinned down to the box by small nails. Also we made small hole on one side of this lino, with which we can get to the whole electronics.

3.1 Electronics

In the theoretical part you could see a simple diagram of how the individual electronic components are linked to the boards. in our case, we didn't use any PCBs because we didn't want to but we didn't need it. by connecting the boards to the drivers and also using the IDE cable to connect the "LED strips" to the Arduino. Our clocks work on +12V, what is power supply for an Arduino. These 12 Volts are transformed to output by lighting up “LED stripes”. But that is not all. These clocks are getting data about time from the internet by Nodemcu. Pins from ESP8266, more precisely TX and rx are connected to serial1 TX and rx on Arduino by voltage divider 1:2, where one applies 5V and next one applies 3V3 logic.

Subsequently the data are sent to the Arduino, which processed these data and sends them to the RTC module. It turns them into an output signals by lighting up the “LED stripes”.



The main task of the first program is to load data from RTC module DS3231 and subsequently light up the LEDs located under the black plastic plates and matt plexiglass.

Brief algorithm:

1. Load time from RTC module and convert it from text format (string) to integer format (int) for better work with it.
2. Determining if a second microcomputer connected to Wi-Fi has sent Arduino time data from the internet.
3. If the exact time data has been received from the internet, it will be converted from text format to numbered and then sent to the RTC module where it will be set.
4. Time is converted from 24-hour to 12-hour format for display.
5. It is calculated in which five minute we are now (0,5,10,15,20,25,30,35,40,45,50,55 minutes) and which minute(s) of that five minute (0,1,2,3,4).
6. The appropriate combination of hours, five minutes, and minutes in five minutes will light up as a combination of LEDs on the light panel.
7. Waiting until 1000ms (1 second) expires and proceeds from step no. 1.

Second program - for ESP8266 Nodemcu:

This program is designed to retrieve time from a school server that is synchronized with the exact Internet time and send it in the appropriate format to Arduino. ESP8266 and Arduino are linked to each other by a serial link.

Brief algorithm:

1. Connect to a WiFi network.
2. Open the website on the school server (<https://spseke.sk/web/servertime.php>), where the time of the server is located.
3. Save the loaded time and send the time in the "# time #: HH: MM: SS" format.
4. Waiting 12 hours to synchronize time again.

3.2 Programs

However we needed RTC DS3231 library, which is not filed in library offer, we had to download it to make this clocks fully work. Here can be seen the programs which we have written and then used in our project. Firstly we wrote program only with RTC DS3231 module, lately with Mr. Copko's help we have done another one but here is used ESP8266 Nodemcu too.



```
hodiny_s_RTC

#include <DS3231.h>

DS3231 rtc(SDA, SCL);

unsigned long cas;
int minuty;
int i;
int patMinut;
int maleMinuty;
int hodiny;

/*
  Zapojenie:
  23 - 29 patminutove LEDky
  36 - 40 minutove LEDky
  42 - 53 hodinove LEDky

  Pouzite funkcie:
  pinMode - nastavi pin ako vstupn (na tlacidlo) alebo vystupny (na LED)
  millis - vrati pocet milisekund od zaciatku vykonavania programu
  digitalWrite - zapne/vypne LEDku
*/

void setup() {
  Serial.begin(9600);
  for (i = 22; i <= 53; i++) { // prejdí všetky LEDky od pinu 22 do pinu 35
    pinMode(i, OUTPUT);      // nastav ich ako výstup
  }
```

```

    digitalWrite(i, LOW);      // na zaciatku ich vsetky zhasni
}
cas = millis();               // uloz zaciatok vykonavania programu do premennej cas
Serial.println(cas);          // vypis zaciatok vykonavania programu
rtc.begin();
}

void loop() {
    String rtcCas;
    rtcCas = rtc.getTimeStr(FORMAT_SHORT);
    String h = rtcCas.substring(0, 2);
    String m = rtcCas.substring(3, 5);
    minuty = m.toInt();
    hodiny = h.toInt();

    if (hodiny > 12) hodiny = hodiny - 12; //dvanasthodinovy cas
    if (hodiny == 0) hodiny = 12; //ak je po polnoci, tak nie je 0 hodin ale 12 hodin

    Serial.println(rtcCas);
    Serial.print(hodiny);
    Serial.println(minuty);

    patMinut = minuty / 5; //modre ledky, ktore signalizuju patminutovky (0,5,10,15,20,25,30,35,40,45,50,55)
    maleMinuty = minuty % 5; //zelene ledky, ktore signalizuju cas medzi patminutovkami (1,2,3,4)

    for (i = 22; i <= 53; i++) { //prejdi vsetky ledky pripojene na piny 22 az 53
        digitalWrite(i, LOW);      //zhasni kazdu ledku
    }

    digitalWrite(41 + hodiny, HIGH); //zapni prislusnu cervenu LED

    switch (patMinut) {
        case 0: break; // 0-4 minuta

        case 1: digitalWrite(23, HIGH); // 5-9 minuta slovo pat
            break;

        case 2: digitalWrite(24, HIGH); // 10-14 minuta slovo desat
            break;

        case 3: digitalWrite(25, HIGH); // 15-19 minuta slovo patnast
            break;

        case 4: digitalWrite(26, HIGH); // 20-24 minuta slovo dvadsat
            break;

        case 5: digitalWrite(26, HIGH); // 25-29 minuta slova dvadsat a pat
            digitalWrite(23, HIGH);
            break;

        case 6: digitalWrite(27, HIGH); // 20-34 minuta slovo tridsat
            break;

        case 7: digitalWrite(27, HIGH); // 35-39 minuta slova tridsat a pat
            digitalWrite(23, HIGH);
            break;

        case 8: digitalWrite(28, HIGH); // 40-44 minuta slovo styridsat

```

```

    break;

case 9: digitalWrite(28, HIGH); // 45-49 minuta slova styridsat a pat
    digitalWrite(23, HIGH);
    break;

case 10: digitalWrite(29, HIGH); // 50-54 minuta slovo patdesiat
    break;

case 11: digitalWrite(29, HIGH); // 55-59 minuta slova patdesiat a pat
    digitalWrite(23, HIGH);
    break;
}

if (maleMinuty > 0) { //ak nie je 0, 5, 10, 15, ..... tak:
    digitalWrite(36 + maleMinuty - 1, HIGH); //zapni prislusnu zelenu led
} //inak (pri 0,5,10,15,...) nechaj zelene vypnute

while (millis() - cas < 1000) {} //potom do casu 1000ms nerob nic
cas = millis(); //uloz si novy zaciatok, odkedy pocitas cas
//Serial.println(cas); //vypis novy zaciatok

```

Programme of clocks with RTC, without Nodemcu

3.3 Testing

Because we wanted to know if our project is well-working, we made small tests during our work so many times. As I meant before, we made test of stability, when we wanted to get sure about if it is static. Later we made test of the connection of IDE cable, where we checked if it is conducting. At the end of our work we made the last one test, where we wanted to know if it works as we wanted. Fortunately it works so good without any problems. RTC did not have a lag, program was written very well and clear if we want to change something, we did not have any problem with ESP8266 Nodemcu either and all of the LEDs light up really soft.

4 Conclusion

Even though we went through many problems, we almost lost the hope of successful ending of our project. Though we had solved all the problems and our work was successful. During this project I have learned many new things about Arduino, whether what it is or how does it works and got knowledge how to make programmes in Arduino IDE too. Also I have confirmed my knowledge from electrotechnic, when we were soldering LEDs with resistors or LEDs together to make a stripe of them. What concerns the problems during our work, they were not a few, but we advised each of them to manage the job successfully. Perhaps the biggest problem was, when we could not find anything to connect the IDE cable to Arduino.

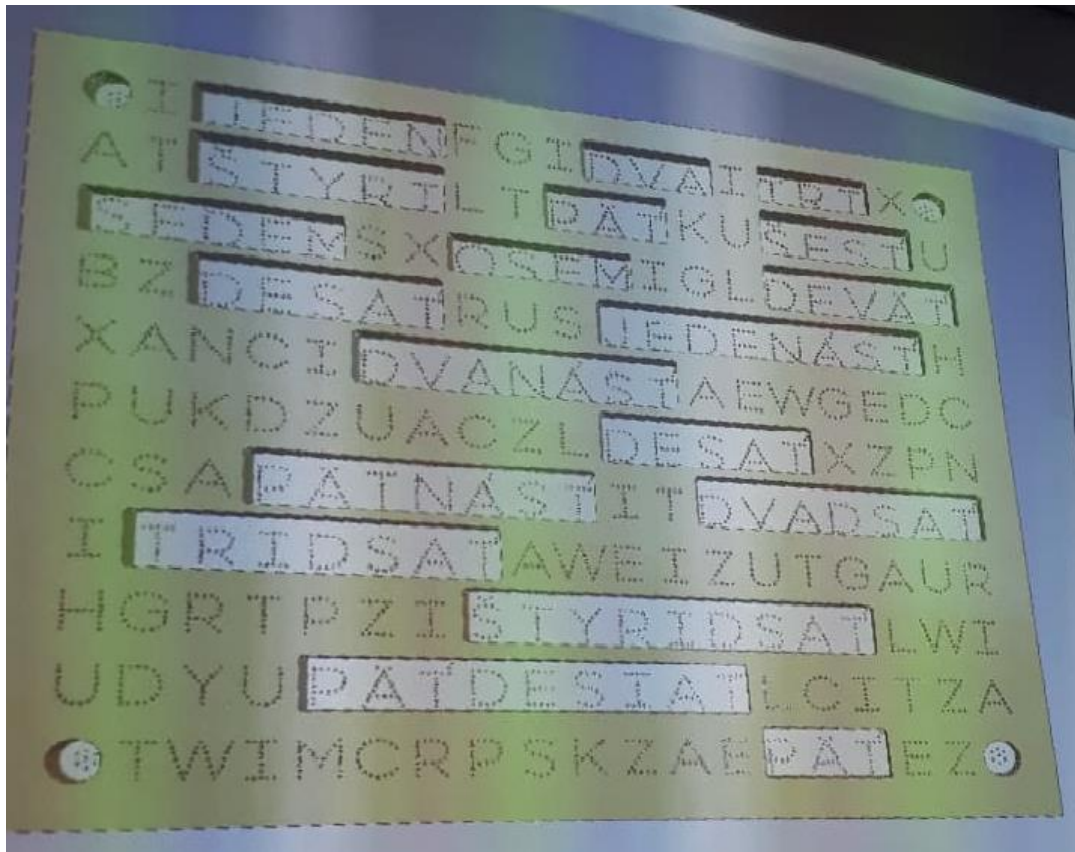
We would definitely like to continue working on the project, fine-tune the details and by having the opportunity for one of us to use this project to practical school leaving exams, I would like to take this opportunity.

I have to say for myself, that I am really pleased with how we worked with Marcin on this project and that we managed to finish this project. And I also liked the division of tasks by being equally same divided between two us.

5 References

- 1) Banzi, Massimo: Getting started with Arduino. In: www.store.arduino.cc [cit. 2019-01-02]
Dostupné na: <https://store.arduino.cc/getting-started-with-arduino-3rd-edition>
- 2) Dallas Semiconductor Corporation: DS3231, In: web.wpi.edu [2019-01-03] Dostupné na: <https://web.wpi.edu/Pubs/E-project/Available/E-project-010908-124414/unrestricted/DS3231-DS3231S.pdf>
- 3) Programing Arduino: Šášik, R: Programovanie Arduina. In: www.arduino-poslovensky.sk, 2016 (translated from slovak language)
- 4) Redaction Root.cz: NodeMCU a jeho verzie: doska s Wi-Fi čipom ESP 2866. In: www.root.cz [cit. 2019-01-02] Dostupné na: https://www.root.cz/clanky/nodemcu-a-jeho-verzie-doska-s-wi-fi-cipom-esp8266/?fbclid=IwAR0o_AzmLJybr4dKfrzyfHp0-Z3drtwhY2aR49yfFeeEjVMYXU8XYWb0WMo (translated from czech language)
- 5) Voda, Zbyšek – tým HW KITCHEN: Průvodce světem Arduina: Seznámení s Arduinem – Typy desek, 2018. s.14-15. (translated from czech language)

6 Attachments



AutoCAD sketch of Slovak board



Polish board with milled holes for LEDs and black plastic plate with milled words



Slovak board with divided IDE cable and connected to LED stripes