**REPORT**

**Group F3 IT IFE, 4th semester**

Monday 10:00 lab

Date: 25.05.2020

Project title: Smart Home

Description: Controlling sprinkles, buzzer through Wi-Fi. Motion detection.

**Team:**

Zuzanna Grabowska 224088 - Team Leader

Justyna Bebak-Szczucka 224084

Patrycja Klejszmit 224091

**Devices used:** Moduł Arduino UNO R3 CH340 klon

**Hardware used:** LED Display, expander PCF8574, ESP8266 module, Active Buzzer, RTC DS1307, Diodes LED, Temperature sensor TMP36, Motion Sensor PIR HC-SR501, Stepper Motor 28BYJ-48 5V + ULN2003

**Contents**

1. Task division and realization. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3

2. Project description. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3

2.1. General Description. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3

2.2. User Guide. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4

3. Peripherals and interface configuration. . . . . . . . . . . . . . . . . . . .8

3.1 ESP8266 and Internet connection . . . . . . . . . . . . . . . . . . . . . . . .8

3.2 LCD Display via PCF8674 expander module . . . . . . . . . . . . . . .10

3.3 I2C bus. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .11

3.4 Passive Buzzer. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .11

3.5 Real Time Clock and Temperature Sensor . . . . . . . . . . . . . . . . . .12

3.6 Motion Sensor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .13

3.7 Stepper Motor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .13

3.8 LED diodes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .14

4. Failure Mode and Effect Analysis. . . . . . . . . . . . . . . . . . . . . . . .15

5. Other information. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

6. References. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .17

1.Task division and realization

Due to critical, pandemic world situation, we were forced to cooperate online. That’s why only one of us - Zuzia, was responsible for physically connecting all cables and devices. However, we shared the rest of work as follows.

|  |  |
| --- | --- |
| Zuzanna | * WiFi mode * LCD display |
| Justyna | * Buzzer song * Clock and temperature |
| Patrycja | * Motion detector * Stepper motor |

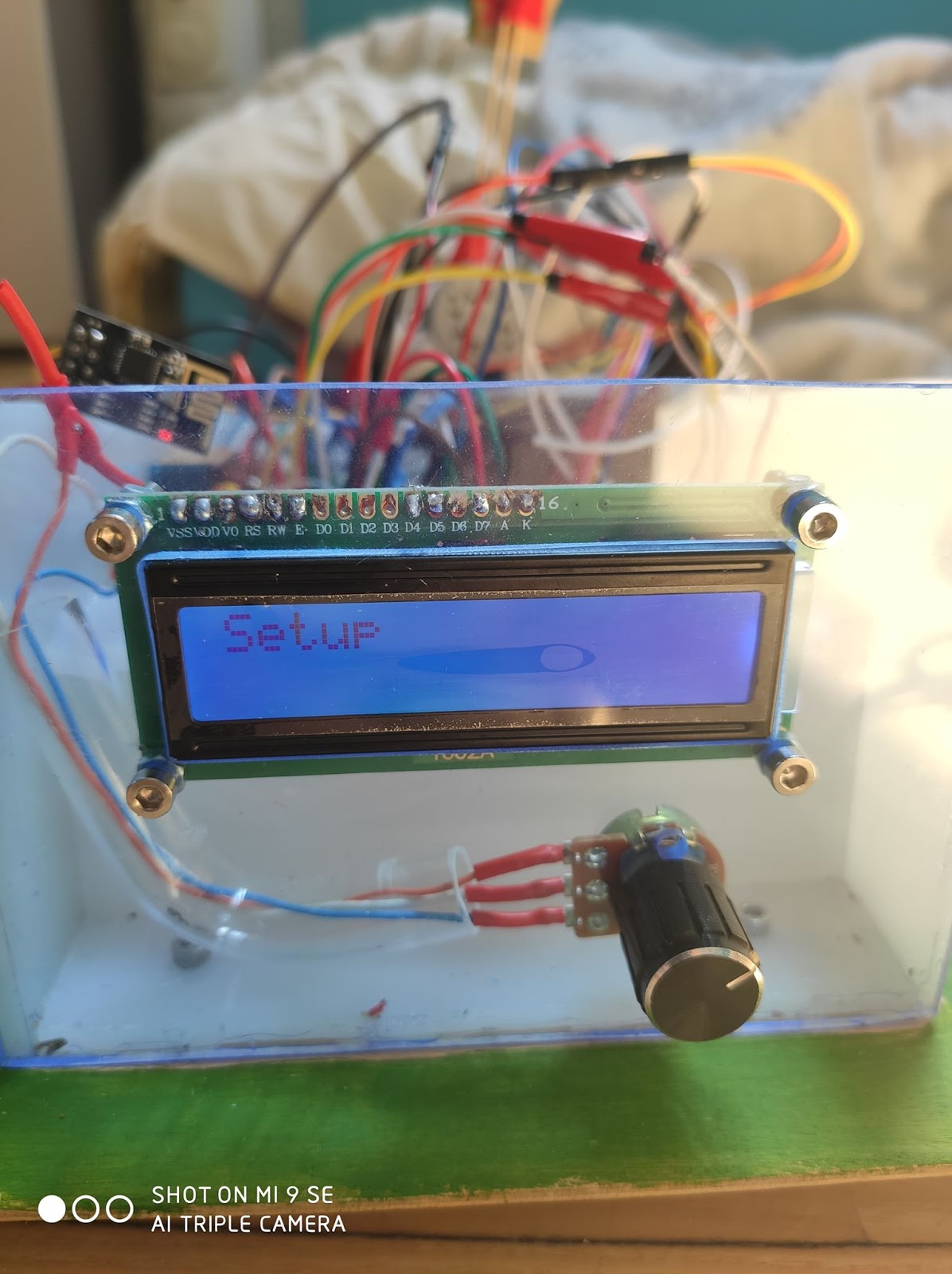
Fig 1. Work division

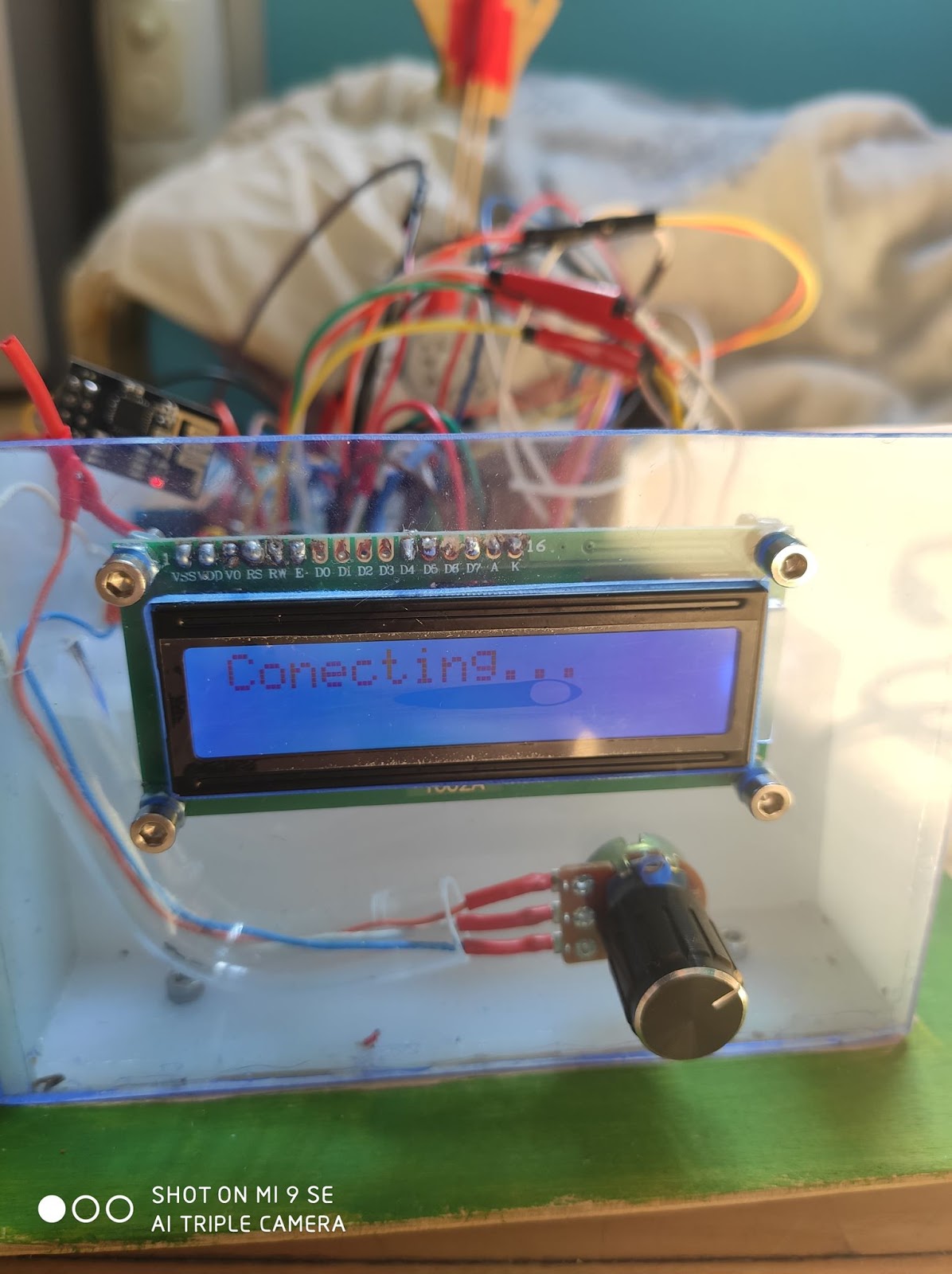
2. Project description

2.1 General description

The idea of this project came from the need of controlling sprinkles remotely in our team leader’s house. That’s why we decided to create the smart home in order to make life easier. The project is an embedded system which enables user to manage various devices, such as diodes or buzzer, using WiFi. It also has a fun functionality which is welcoming the user by waving its hand after the motion detection.

2.2 User guide

1. The first step is connecting the device to the power supply. The system makes a setup automatically and connects to WiFi through AT commands. After those configurations, there should be a temperature and time displayed on a screen, which means that the device is ready for use.  
  
Fig 2. Setup of the device

  
Fig 3. System connecting to WiFi

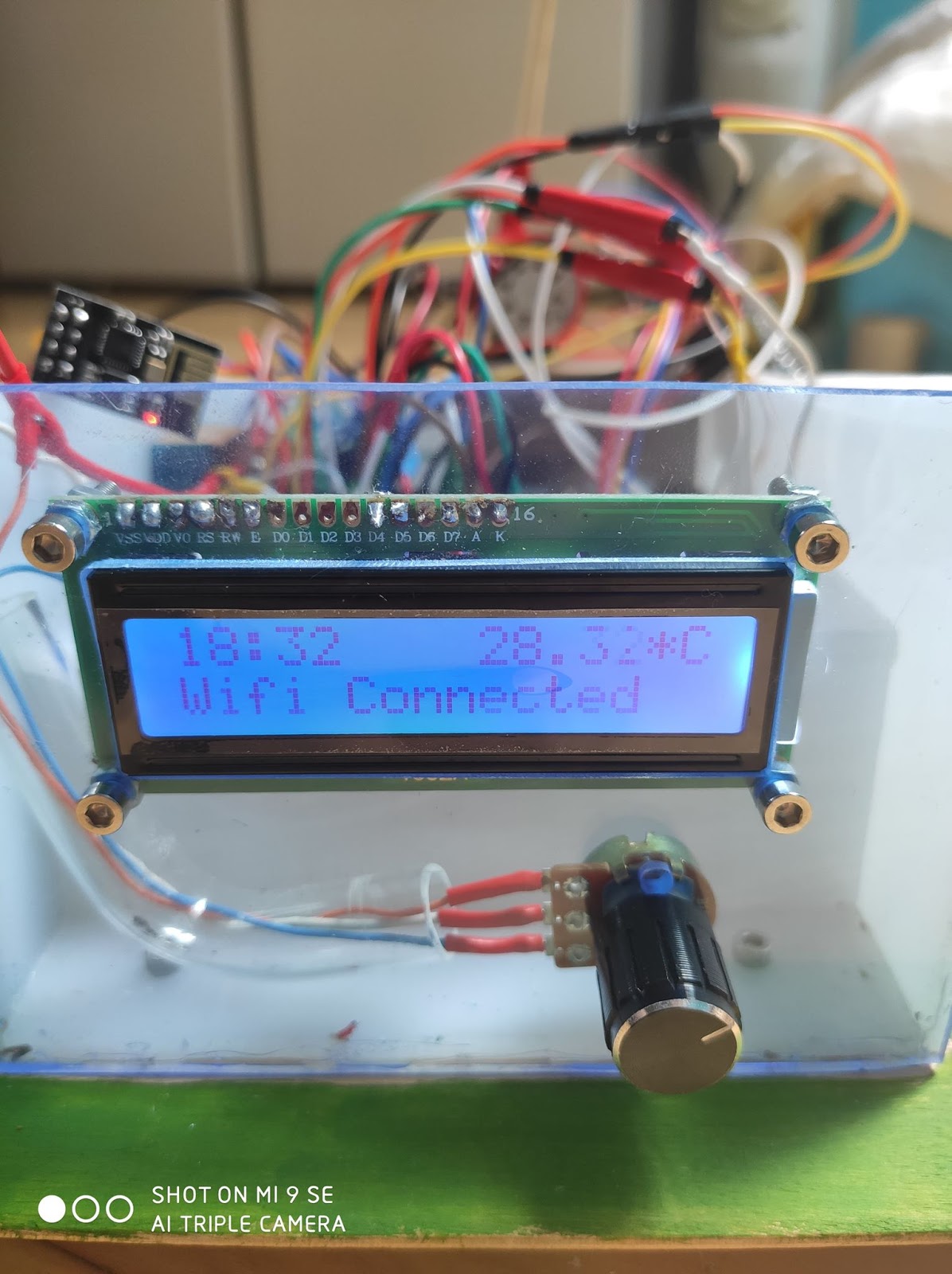
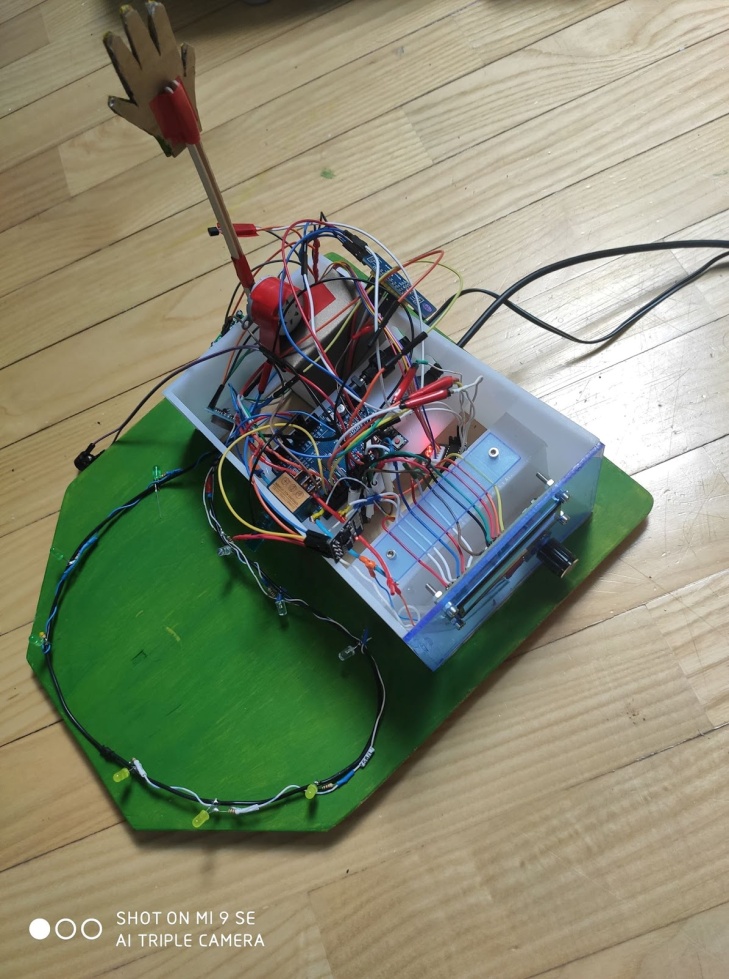
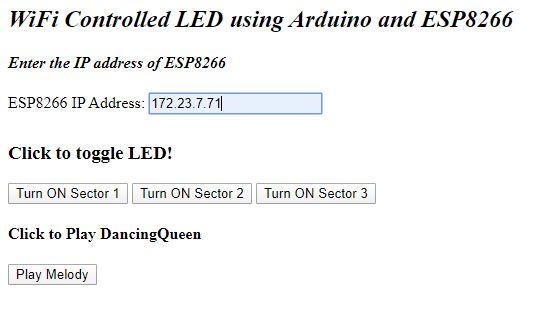
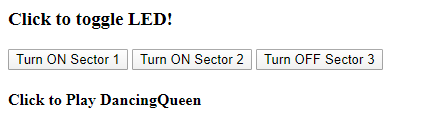
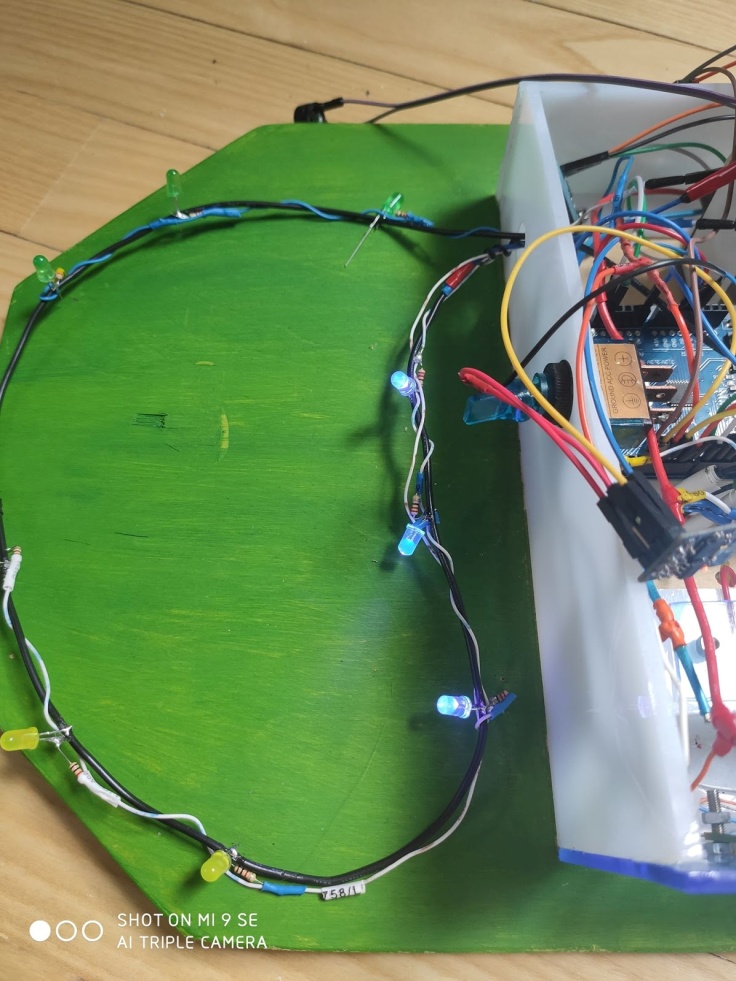
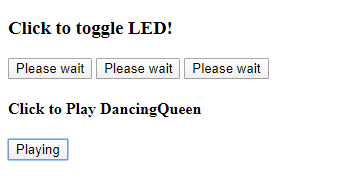


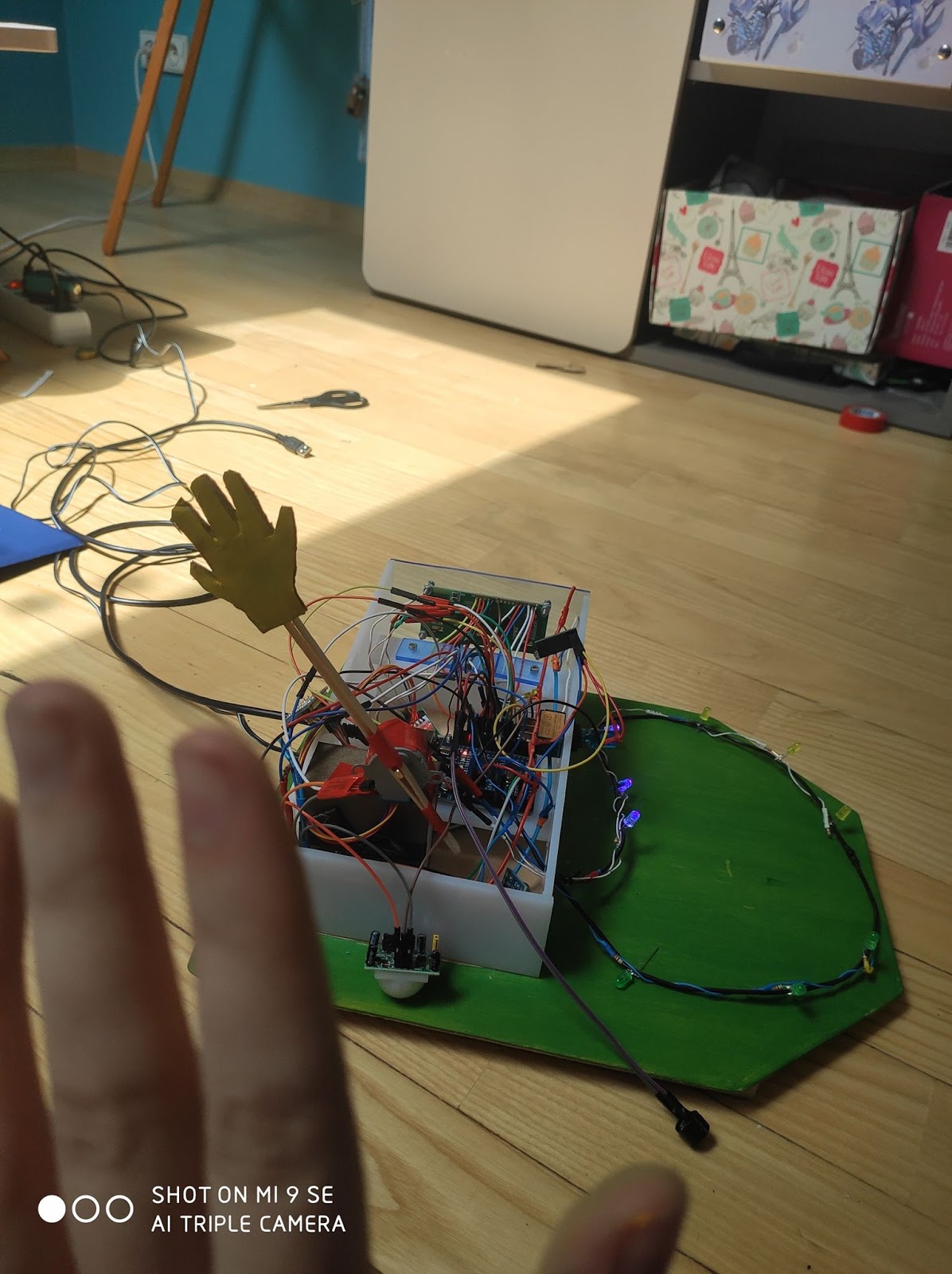
Fig 4. Configured device - ready for work

2. Now we can start the management of functionalities through the WiFi. On the html page there are four buttons. Three of them switch the diodes from different sections, whereas the fourth one is responsible for playing the song on buzzer. When you decide to listen to the song, you have to wait until it ends and the proper statement is displayed on the page.

  
Fig 5. Our whole device  
  
  
Fig 6. Screenshot from html page, each functionality is turned off  
  
  
Fig 7. Screenshot from html page, diodes from sector 3 turned on

  
Fig 8. Blue diodes (sector 3) turned on  
  
  
Fig 9. Screenshot from html page, buzzer playing the song

3. This smart home has a fun addition. When it sees you’re moving, it will wave to you with a huge hand. It works by detecting the motion with motion sensor that activates the stepper motor which moves the hand.

  
Fig 10. Waving the hand

3.Peripherals and interface configuration

3.1 ESP8266 and Internet connection

For our communication with the user via internet we use ESP8266 module.

According to datasheet[4] the ESP module maximum working voltage is 3.6V, in order to work with this module safely it is very important to connect it correctly using appropriate level conversion method. We used a circuit consisting of a 1 KΩ Resistor and two 1KΩ Resistors connected in serial before connecting the TX and RX pins of our module to Arduino.

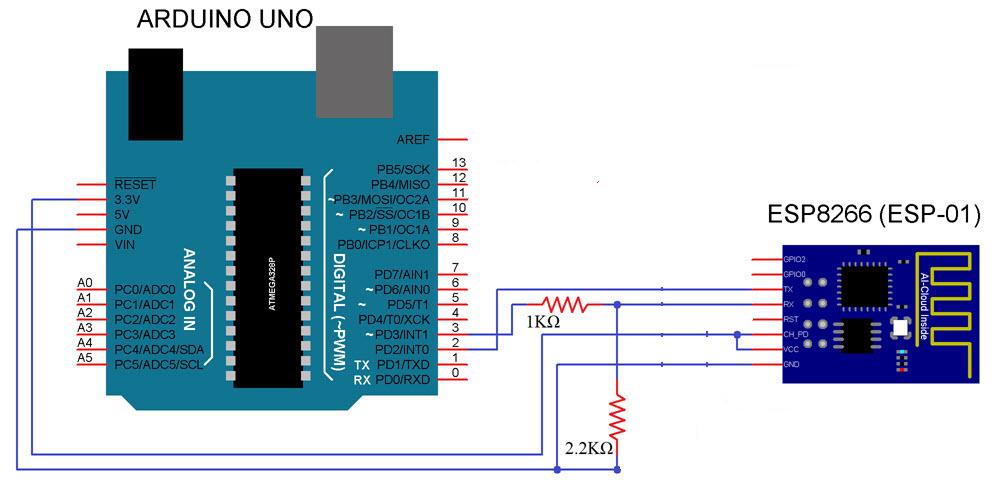


Fig 11. Connection of ESP8266 module with appropriate level conversion.[5]

To communicate with our module we use SoftwareSerial Library, which is a built-in support for serial communication, using UART [6].We provide this library with number of pins we want to use as TX (Transmitting) and RX (Receiving) pins.

SoftwareSerial esp8266(3, 4);

We begin our communication by setting a baud rate. ESP8266 runs on the following possible baud rates:

9600/115200: Usually runs on these two baud rates where you can issue AT commands to it

74880: This baud rate is usually used by the boot loader.

In our code we use 115200, because we use 9600 for Debug serial communication and we try to avoid mistakes this way.

Additionally we use a function created by us to easily send messages and receiving answers from AT commands that are used to establish Internet connection

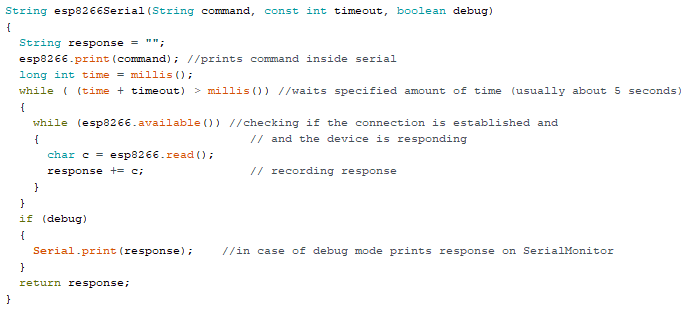


Fig 12. Fragment of code containing function used to communicate with ESP8266

This function is responsible for sending messages via SoftwareSerial and returning the response.We use this function to send AT commands

|  |  |
| --- | --- |
| AT command | Description |
| AT+RST | restart the module |
| AT+CWMODE=<mode> | Wifi mode with possible options:  **1** = Station mode (client)  **2** = AP mode (host)  **3** = AP + Station mode |
| AT+CWJAP=<ssid>,< pwd > | join the AP where  ssid - name of Wifi  pwd - password |
| AT+CIFSR | Get local IP address. |
| AT+CIPMUX=<mode> | Enable / disable multiplex mode  0 - single connection  1 - multiple connection |
| AT+CIPSERVER=<mode>,<port> | Configure ESP8266 as server  mode 0 - close server mode  mode 1- open  port - port(we use 80) |

Fig 13. AT commands used by us in order of usage [7]

In order to obtain massage from server we search our stream of information for “?” and then proceed to divide our information into smaller parts containing commands for the rest of our application.

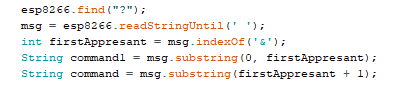


Fig.14 Fragment of code responsible for division of messages.

Exemplary original message looks like this:

*led1=ON&led2=ON&led3=OFF&buzzer=ON*

After division we proceed to if statements in order to switch on or off our diodes (sprinkles sectors).

3.2 LCD Display via PCF8674 expander module

In our project LCD Display is used to display time, temperature and information about connection to Internet.

Considering the fact that the Arduino board has limited number of pins, we decided to use PCF8674 expander module and I2C interface. To accomplish this task we use LiquidCrystal\_I2C Library which allows us to communicate with LCD display using 19th and 18th pin of our Arduino board for SCL and SDA data transfer lines in I2C bus.Exemplary circuit can be viewed here. [8]

We provide the library with information about our expander address which can be calculated from the three address bits A0, A1 and A2.First 7-bits combine to form the slave address. The last bit of the slave address defines the operation (read or write) to be performed. When it is high (1), a read is selected, while a low (0) selects a write operation. It can be also checked using I2C Scanner[2].In our case, because we don’t have any additional expander devices we decided to leave the default address 0x20.

We specify the address and information about our LCD display in this line of code LiquidCrystal\_I2C lcd(0x20, 16, 2);

An LCD [3] is an electronic display module which uses liquid crystal to produce a visible image. We use the 16×2 LCD display which is a very basic module commonly used in circuits. The 16×2 displays 16 characters per line in 2 such lines. In this LCD each character is displayed in a 5×7-pixel matrix.

3.3 I2C bus

The I2C bus is used by two devices: PCF8674 expander and RTC DS1307.Each uses a different I2C address.

The two wires, or lines are called Serial Clock (or SCL) and Serial Data (or SDA). The SCL line is the clock signal which synchronize the data transfer between the devices on the I2C bus and it’s generated by the master device. The other line is the SDA line which carries the data. The data signal is transferred in sequences of 8 bits, after start condition occurs the first 8 bits sequence which contains the address of the slave. After each sequence follows a bit called Acknowledge and then another addressing sequence for internal registers of slave device, and next the data sequences. After the addressing, the data transfer sequences begin either from the master or the slave depending of the selected mode at the R/W bit.

Arduino IDE has built-in *Wire*library which is used for making communication with I2C devices. If we study both RTC and LiquidCristal\_I2C include Wire.h header and invoke

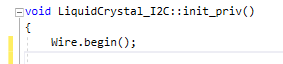
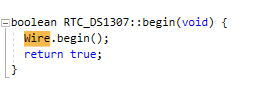


Fig 15. and Fig 16. Fragments from .cpp files from both libraries.

Wire.begin() function this initiates the Wire library and joins the I2C bus in master mode. [9]

3.4 Passive Buzzer

To achieve changes in volume we use toneAC Library and connect both speaker wires to Arduino pins, which are specific for different boards, in case of Arduino Uno they are pins 9 and 10.[1]

Due to ESP8266 Wifi Module that establishes WiFi connection (which is explicitly described in the paragraph 3.1), when “Play Melody” button in the index.html file is pressed, a fragment of “Dancing Queen” chorus by ABBA is played. The melody is played in a loop until the button is pressed again. It finishes current loop before stopping. The function responsible for playing the melody (Play\_Dancing\_Queen) uses toneAC library and three headers. The library allows to control the intensity of the sound by producing an alternating push/pull between two pins.It's not really alternating current as in-wall electrical wiring because it's a square wave and never produces a negative voltage. However, the effect of the alternating push/pull creates an effective double voltage differential which produces the higher volume level. and provides toneAC() function that takes four parameters - frequency, duration, intensity of a note and background (allowing note to play in the background). Header “pitches.h” assigns frequencies to the musical notes names, header “durations.h” assigns numerical durations (specifically denominators by which the constant of 10500 milliseconds is divided and assigned as integer noteDuration variable in Play\_Dancing\_Queen function) to names of note values and “melody.h” consists of three arrays with notes, respective duration and respective numerical intensity of every note.

3.5 Real Time Clock and Temperature Sensor

In our project we use DS1307 RTC module for keeping track of current time and then displaying it on LCD Display. This module is low-current device that runs on a single lithium cell battery (CR2032). Our device uses I2C communication to transfer data. Arduino always acts as a master and RTC as a slave. We connect SDA nad SLD pins of our clock to analog pins A4 and A5 of Arduino accordingly. To control RTC we use *RTClib* [10]and specifying the model of our device to enable accurate addressing and functioning of this library. Time and date can be also updated manually.



Fig 17. Fragment of code with adjustment of time and date.

Then we can easily display it using *ourtime.hour()* and other functions available in this library.

In order to display temperature on our LCD we use Temperature Sensor TMP36 this sensor uses the property of diodes; as a diode changes temperature the voltage changes with it at a known rate. The sensor measures the change with accuracy to 0.1°C and outputs an analog voltage between 0 and 1.75VDC based on it. To get the temperature we need to measure the output voltage and calculate the outcome.

Firstly we need to convert the number 0-1023 from the ADC into 0-5000 mV (= 5V)

float voltage = reading \* 5.0;

voltage /= 1024.0;

Then, to convert millivolts into temperature,we use this code:

float temperature = (voltage - 0.5) \* 100 ;

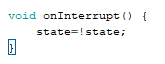
Then we can display this value on our LCD.

3.6 Motion Sensor

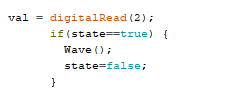
In our project we use Motion Sensor PIR HC-SR501 to detect motion in order to wave to user using stepper motor. This device can detect motion based on changes in infrared light in the environment.The sensor has two built-in potentiometers to adjust the delay time (the potentiometer at the left) and the sensitivity (the potentiometer at the right). In order to obtain the effect we desire we change the input from motion sensor into interrupt.In case of Arduino Uno the pins which can be translated to interrupt pins are pins number 2 and 3 [11].

attachInterrupt(digitalPinToInterrupt(2), onInterrupt,RISING);

The line of code above transforms the second pin of   
Arduino into an interruptor and when value on this pin starts to rise, the board automatically calls function *onInterrupt().*

**

This function changes value of boolean state into opposite value,



so in next loop this part of code will be executed. We can’t attach function *Wave()* directly to the interrupt because it changes outputs of pins and won't work appropriately and may cause various number of mistakes during execution of our code.

3.7 Stepper Motor

We use Stepper Motor 28BYJ-48 5V and driver motor ULN2003 to wave back at users.

The motor has a 4 coil unipolar arrangement and each coil is rated for +5V so it is relatively easy to control with any basic microcontrollers. This motor has a stride angle of 5.625°/64, this means that the motor will have to make 64 steps to complete one rotation and for every step it will cover a 5.625° .In the full step control mode always two windings are energized at the same time.

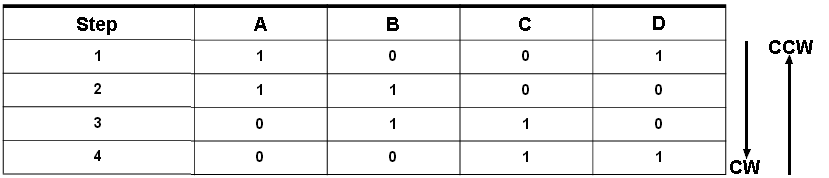


Fig 18. Table showing directions where 1 means the coil is energized and 0 means not energized

In our code we use built-in Arduino library called Stepper, which enables easy and fast control over Stepper motor, by providing information about used pins, number of steps and speed of rotating.

3.8 LED diodes

In order to properly represent sectors of sprinkles we devised and created a circuit consisting of diodes where one cable is connected to GND and other cables are connected with output pins on arduino. Anodes of diodes are connected via resistors to pins and cathodes to ground.

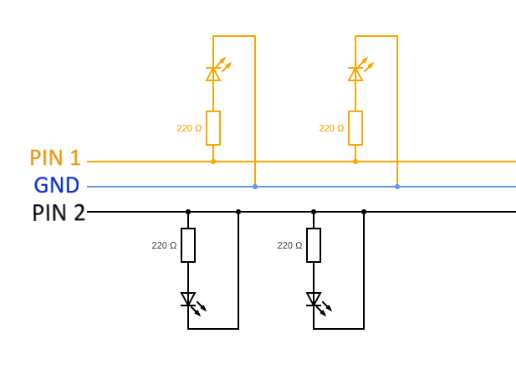


Fig 19. Shows adjusted by us circuit.

Diodes are controlled via internet using buttons on our html side.

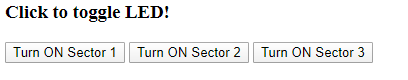


Fig 20. Screenshot of our html site.

4. Failure Mode and Effect Analysis

Our project consists of a number of different devices working in relation with one another. Some of the functionalities are independent, and some are connected with each other in the way that failure of one means failure of the other.

|  |  |
| --- | --- |
| **Functionality** | **Consequences of failure** |
| Microcontroller | Critical |
| ESP82766 module | Critical |
| Power Supply | Critical |
| LED Diodes | Medium |
| LED Display | Medium |
| Buzzer | Negligible |
| RTC | Negligible |
| Temperature Sensor | Negligible |
| Motion Sensor | Negligible |
| Stepper Motor | Negligible |

Fig 20. Table of functionalities and exemplary consequences of its failures.

In the table above we can see the analysis of devices and functionalities, rating the consequence of their failure from critical to negligible. This project mainly depends on Microcontroller, Wifi module and Power Supply and failures of these elements would cause greatest damage in our project. LED Display and diodes are crucial for contact with the user but any error in its functioning would not be critical for the whole system. Failure of the rest of the elements would not influence the system besides taking away their own functionalities.

Detection of the problem depends on the functionality in which it would be detected:

Failure of Microcontroller would be the easiest to detect as it would cause immediate malfunctioning or even a shut down of the whole system.

Power source problems could be detected by observing the behaviour of the diodes or a piezo buzzer, any blinking of lights or abrupt lowering of the volume would indicate a possible power-related problem.

Most of these problems, because of creation of our project which involved a lot of soldering, can be caused by breakings in connection. We try to use as much heat shrink tubing to prevent this situations.

5. Other Information

Due to the fact that this device is supposed to work in long term as sprinkles control device, we tried to implement an off switch to LCD Display in order to save energy and improve longevity of our project.

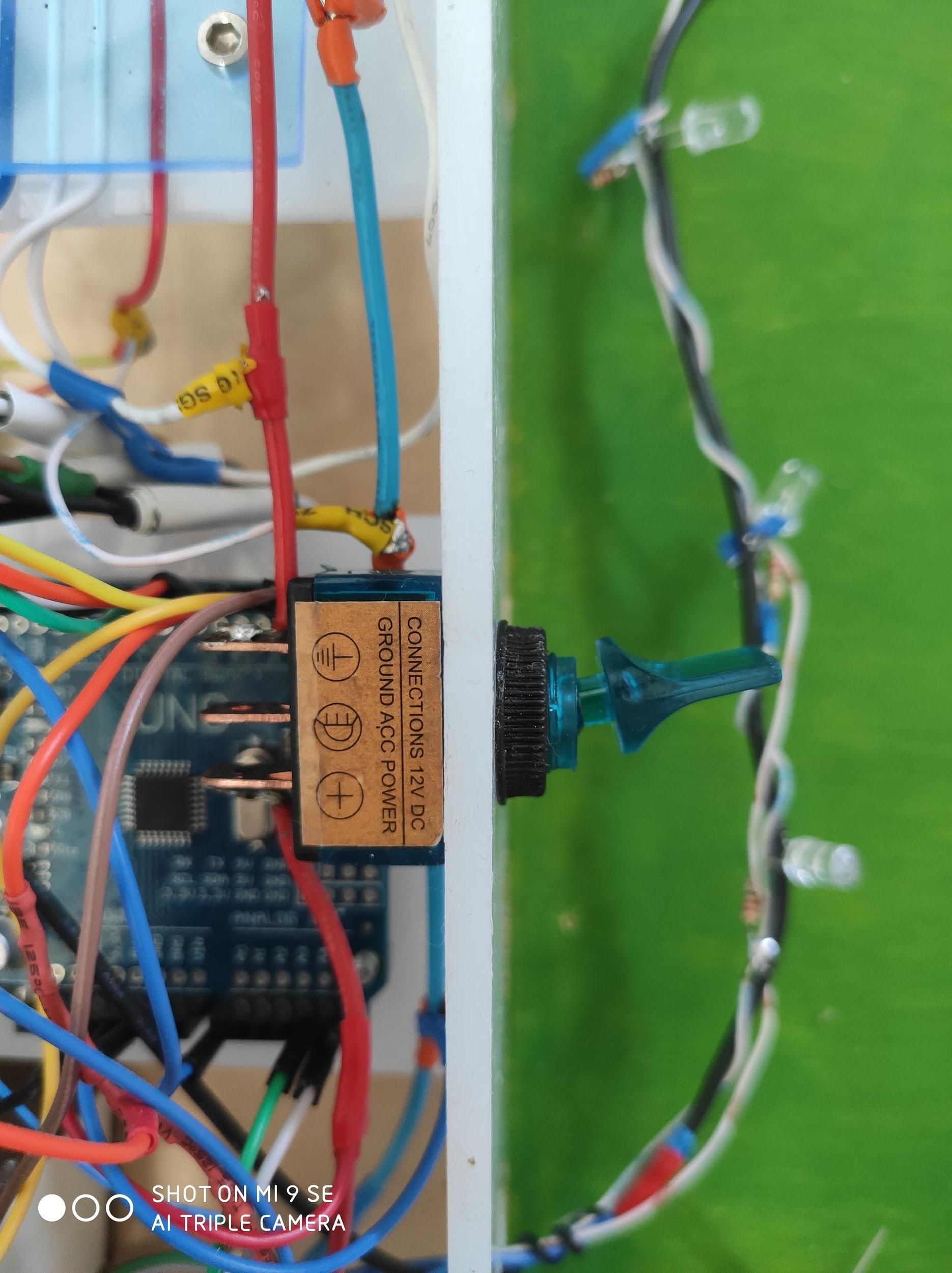


Fig 21. Implementation of our switch.

Unfortunately our idea was not properly thought through as switching off power from LCD causes reset of whole screen and does not work correctly.

6.REFERENCES

[1]<https://bitbucket.org/teckel12/arduino-toneac/wiki/Home>

[2]<https://playground.arduino.cc/Main/I2cScanner/>

[3]<https://components101.com/16x2-lcd-pinout-datasheet>

[4][ESP-01/07/12 Series Modules User's Manual](http://wiki.ai-thinker.com/_media/esp8266/esp8266_series_modules_user_manual_v1.1.pdf)

[5]<https://www.electronicshub.org/wifi-controlled-led-using-esp8266-and-arduino/>

[6]<https://www.arduino.cc/en/Reference/softwareSerial>

[7]<https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf>

[8]<http://1.bp.blogspot.com/-3rGvx1WiEHQ/VRJQyqSIqTI/AAAAAAAASAc/Gs9DN97CM8Q/s1600/PCF8574-LCD-HD44780-schemat.png>

[9]<https://circuitdigest.com/microcontroller-projects/arduino-i2c-tutorial-communication-between-two-arduino>

[10]<https://www.arduinolibraries.info/libraries/rt-clib>

[11]<https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/>