Department of Electronic and Telecommunication Engineering

University of Moratuwa

EN3023 - Electronic Design Realization

UV-C Disinfector

By TheSquad

SHYAMAL M.A.L.	- 180601T
PATHIRANA R.P.U.A.	- 190432J
ABEYSINGHE W.A.M.S.Y.	- 190014F
RATHNAYAKE R.M.K.L.	- 190520D

This Document is Submitted as Partial Fulfillment

For The Module EN3023

Electronics Design Realization

21st January 2023

Table of Contents

1. Introduction

Due to the COVID-19 Pandemic, wearing masks and using sanitiser constantly became the new normal. Today people have adjusted their lifestyles to make peace with the virus. If COVID warned us of one thing, it is to take precautions before action. These viruses can quickly spread with the physical engagement of frequently used items like mobile phones, cash, car keys, masks, etc.

Using sanitisers is not suitable for most objects like money and mobile phone. So, there is a requirement for a convenient and easily reusable mechanism to sanitise these objects frequently and constantly.

We propose an intelligent electronics system powered by a microcontroller to disinfect items. It will be a compact **360-degree disinfecting box using ultraviolet sterilisation**. The disinfector will use UV tubes to perform the task.

It is a very time-consuming task to sanitise everything regularly. Using too many sanitisers is not eco-friendly since its production involves using chemicals. Also, there is a recurring cost of buying sanitisers. Research has also proven that UV-C light is effective in killing viruses in a short period. The most effective wavelength for disinfection purposes is 257nm.

Our solution will be less time-consuming, more environmental-friendly, and cost-effective. The current situation in the world suggests that there will be a niche market for this kind of product. Our primary target market will be the people who are constantly on travel. This device will be battery-operatable and compact. So, people who want to sanitise their keys and mobile phones when they enter their vehicles or come to their homes will be our main target crowd.

Since this problem is a global problem, finding a target audience outside the country is possible. The main goal of the device is to act as a user-friendly portable end product that anyone can use conveniently.

2. Design Approach

During the component selection, a DC power-operated UV-C LED is selected so that the DC power is enough to operate the device. Hence a battery is used in the final product, which makes it possible for the product to be more portable. Also, going with a DC UV light is very cost-effective. Hence a UV-C

A buzzer is included to notify the user that the object is disinfected.

The main goal of the project was to make a portable device. So, the enclosure is designed with 12cmX15cmX20cm dimensions. Since the user should not see the electronic components, two compartments are available in the device; one compartment is for all the electronic components. The other is for the user to place the object to be sanitised. The sanitised object should be exposed to UV light at a 360-degree angle. So, a net is used to hold the things. A battery compartment with a detachable lid is designed at the bottom for the user to detach the battery and recharge it. The hinge is designed using the side panels of the compartments for elegance.

The amount of time taken by the UV-C light to sanitise a surface depends very much on the intensity of the UV-C light and the type of material the surface is made up of. Objects like paper and plastic can be sanitised easily, while the things like glass and metal take much time. So, the user interface is designed to have the option to select three options, 6mins, 8 mins and 10 mins.

3. Code Explanation

After generalising the concept product with its functions and features, we directly moved into the coding part. Since we decided to use Atmega328p as our main microcontroller, coding was done with AVR/ C++ using Atmel Microchip Studio.

Our first task was to assign pins to the microcontroller. The pins were assigned as follows.

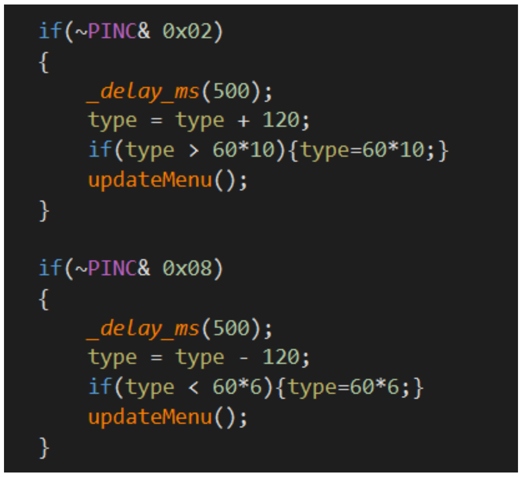
- PD3 to give a command signal to the UVC lights.
- PD5 & PD6 for two LED s to indicate the status of the UVC light.
- SCA & SCL pins for the OLED display
- PB2 for the buzzer
- PC0 /1/2/3 for the push buttons (Start, Force Stop, UP, Down)
- PD1 for the contact switch

The next task was to count the time through the code. Since the time was not much critical issue for the final output, delays were used to calculate the time rather than using the AVR Timers. The time was calculated through loops with an inbuilt `_delay_ms()` function using **100ms** delay as the minimum time step. At each 100ms, the code will check for the **Contact Switch** and the **STOP** pins. If the STOP pin is LOW (STOP button is pressed), the UVC light will be turned off and go back to the main menu. If the Contact Switch is open (LOW), UVC lights will be turned off, and the time countdown will be paused. The process will not be stopped.

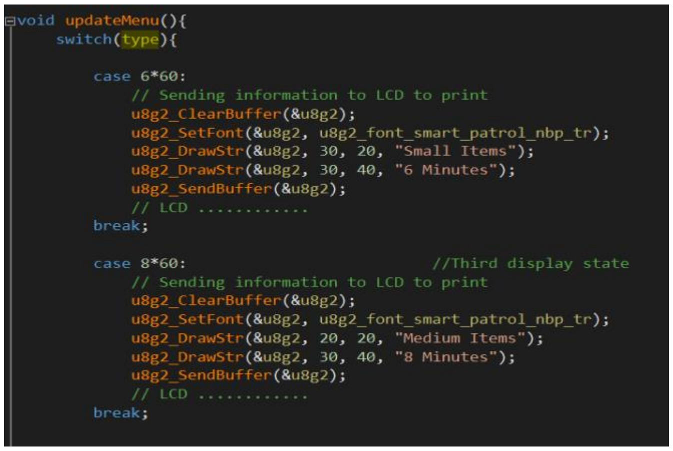
Since our product has three configurations, it can switch to three-time limits. Six minutes, 8 minutes, and 10 minutes. One integer variable called `type was` assigned for this. It is initially set for 8 minutes. Since the code operates by counting the seconds, the variable will be 8*60, 6*60 or 10*60.

The main code consists of three functions. `updateMenu(), UV_light() and play_melody()`

The first function is to change through the different disinfection configuration types, as mentioned above, by pressing the UP and DOWN buttons. At the same time, the menu code will call the function and change the class accordingly. UpdateMenu function will change the Display according to the selected type.



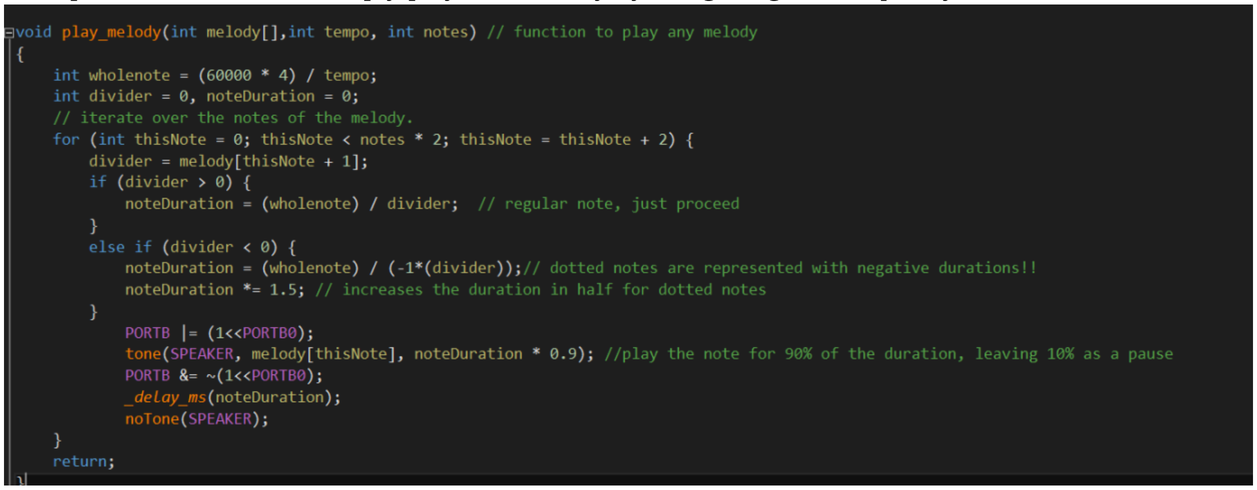
```
if(~PINC& 0x02)
{
    _delay_ms(500);
    type = type + 120;
    if(type > 60*10){type=60*10;}
    updateMenu();
}

if(~PINC& 0x08)
{
    _delay_ms(500);
    type = type - 120;
    if(type < 60*6){type=60*6;}
    updateMenu();
}
```



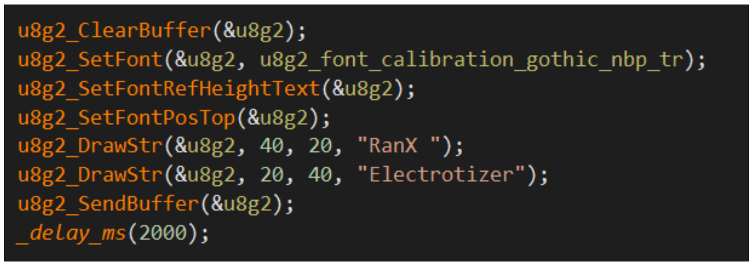
```
void updateMenu(){
    switch(type){

        case 6*60:
            // Sending information to LCD to print
            u8g2_ClearBuffer(&u8g2);
            u8g2_SetFont(&u8g2, u8g2_font_smart_patrol_nbp_tr);
            u8g2_DrawStr(&u8g2, 30, 20, "Small Items");
            u8g2_DrawStr(&u8g2, 30, 40, "6 Minutes");
            u8g2_SendBuffer(&u8g2);
            // LCD ............
        break;

        case 8*60:                          //Third display state
            // Sending information to LCD to print
            u8g2_ClearBuffer(&u8g2);
            u8g2_SetFont(&u8g2, u8g2_font_smart_patrol_nbp_tr);
            u8g2_DrawStr(&u8g2, 20, 20, "Medium Items");
            u8g2_DrawStr(&u8g2, 30, 40, "8 Minutes");
            u8g2_SendBuffer(&u8g2);
            // LCD ............
        break;
```

`UV_light()` function is the primary function which operates to turn on the UV light. This function keeps the light on while the Stop button is not pressed or Lid is not open. And count the time by seconds until it equals the given variable (type).

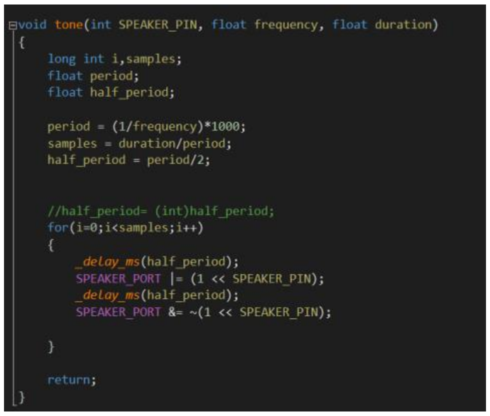
`play_melody()` function will be called at the end of each disinfection process to indicate that the task is completed. This function simply plays the melody by using the given frequency and durations.



```
void play_melody(int melody[],int tempo, int notes) // function to play any melody
{
    int wholenote = (60000 * 4) / tempo;
    int divider = 0, noteDuration = 0;
    // iterate over the notes of the melody.
    for (int thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {
        divider = melody[thisNote + 1];
        if (divider > 0) {
            noteDuration = (wholenote) / divider;  // regular note, just proceed
        }
        else if (divider < 0) {
            noteDuration = (wholenote) / (-1*(divider));// dotted notes are represented with negative durations!!
            noteDuration *= 1.5; // increases the duration in half for dotted notes
        }
            PORTB |= (1<<PORTB0);
            tone(SPEAKER, melody[thisNote], noteDuration * 0.9); //play the note for 90% of the duration, leaving 10% as a pause
            PORTB &= ~(1<<PORTB0);
            _delay_ms(noteDuration);
            noTone(SPEAKER);
    }
    return;
}
```



```
u8g2_ClearBuffer(&u8g2);
u8g2_SetFont(&u8g2, u8g2_font_calibration_gothic_nbp_tr);
u8g2_SetFontRefHeightText(&u8g2);
u8g2_SetFontPosTop(&u8g2);
u8g2_DrawStr(&u8g2, 40, 20, "RanX ");
u8g2_DrawStr(&u8g2, 20, 40, "Electrotizer");
u8g2_SendBuffer(&u8g2);
_delay_ms(2000);
```



```
void tone(int SPEAKER_PIN, float frequency, float duration)
{
    long int i,samples;
    float period;
    float half_period;

    period = (1/frequency)*1000;
    samples = duration/period;
    half_period = period/2;


    //half_period= (int)half_period;
    for(i=0;i<samples;i++)
    {
        _delay_ms(half_period);
        SPEAKER_PORT |= (1 << SPEAKER_PIN);
        _delay_ms(half_period);
        SPEAKER_PORT &= ~(1 << SPEAKER_PIN);

    }

    return;
}
```

It takes the inputs as the melody array, the tempo speed that the piece needs to be played and the musical notations. Then generate the tones and output to a previously assigned Speaker pin.

For the OLED, we used a custom-created AVR library. The library allows us to customise the text position, font, and size. This custom-created U8g2 library is much similar to the Arduino Adafruit OLED library.

We generated tone according to the frequency of the buzzer tone by creating 50% duty cycles. The following code snippet shows the function for generating techniques for frequency. Then we found a musical notation library with frequency notation durations for a popular melody. Using that melody, we create music at the end of each Disinfection process.

4. Simulation

Before going for an end product, the electronics components were simulated using proteus to ensure that the components were functioning as intended and debug the code. For this purpose, Proteus 8.13 is used.

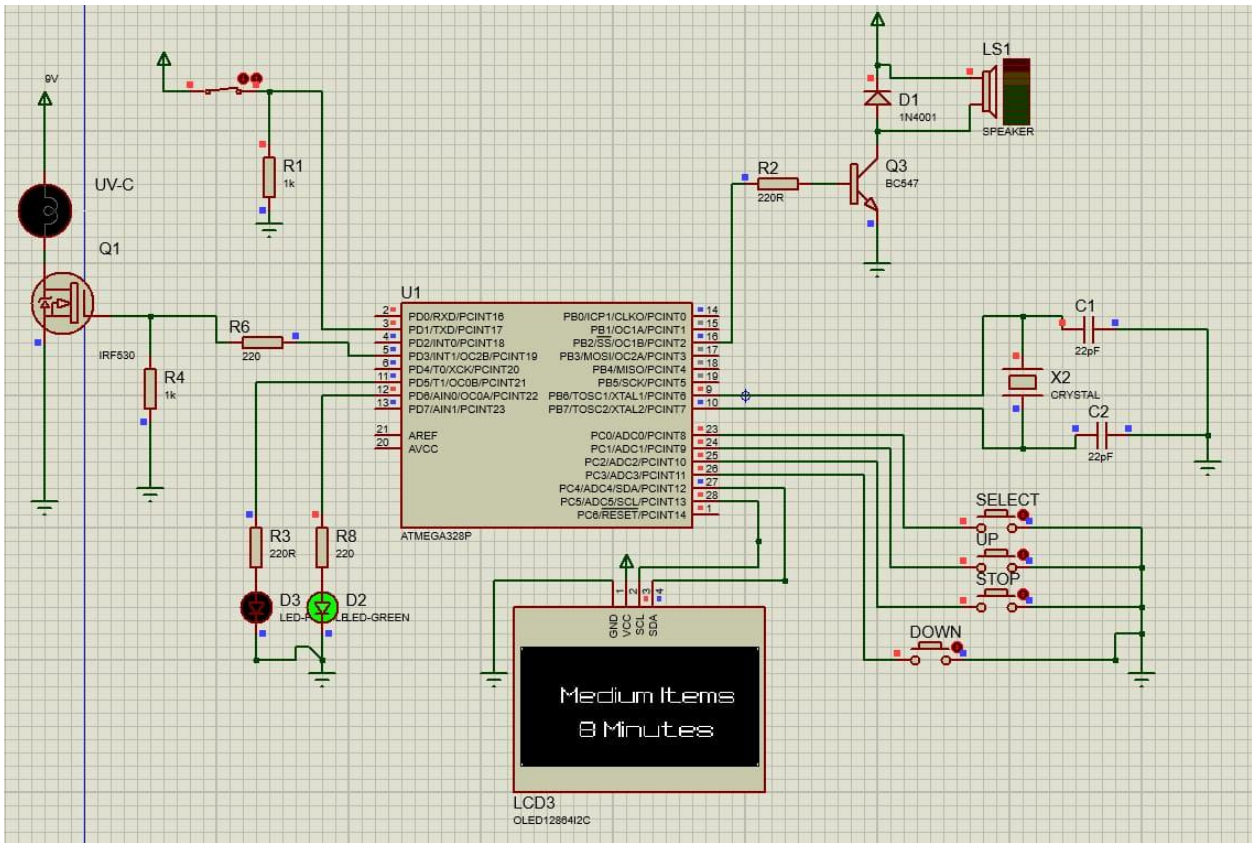


Figure 1: Main Menu of the Product Interface

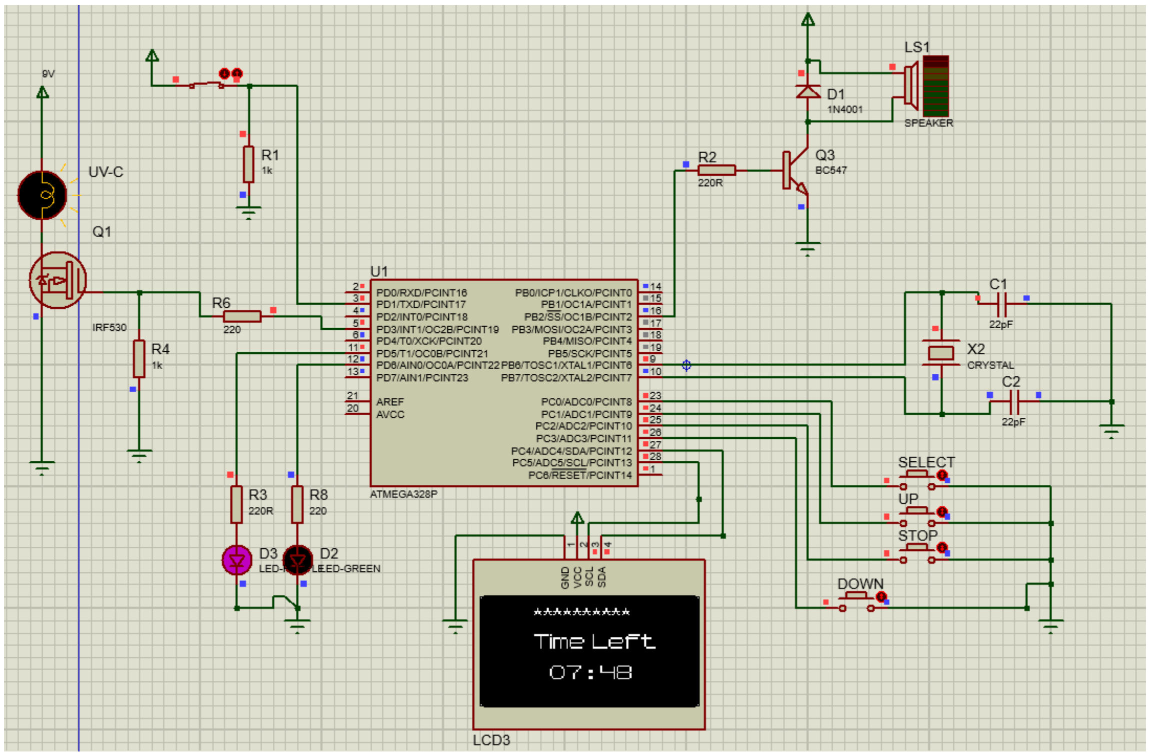


Figure 2: During the Sanitization Process

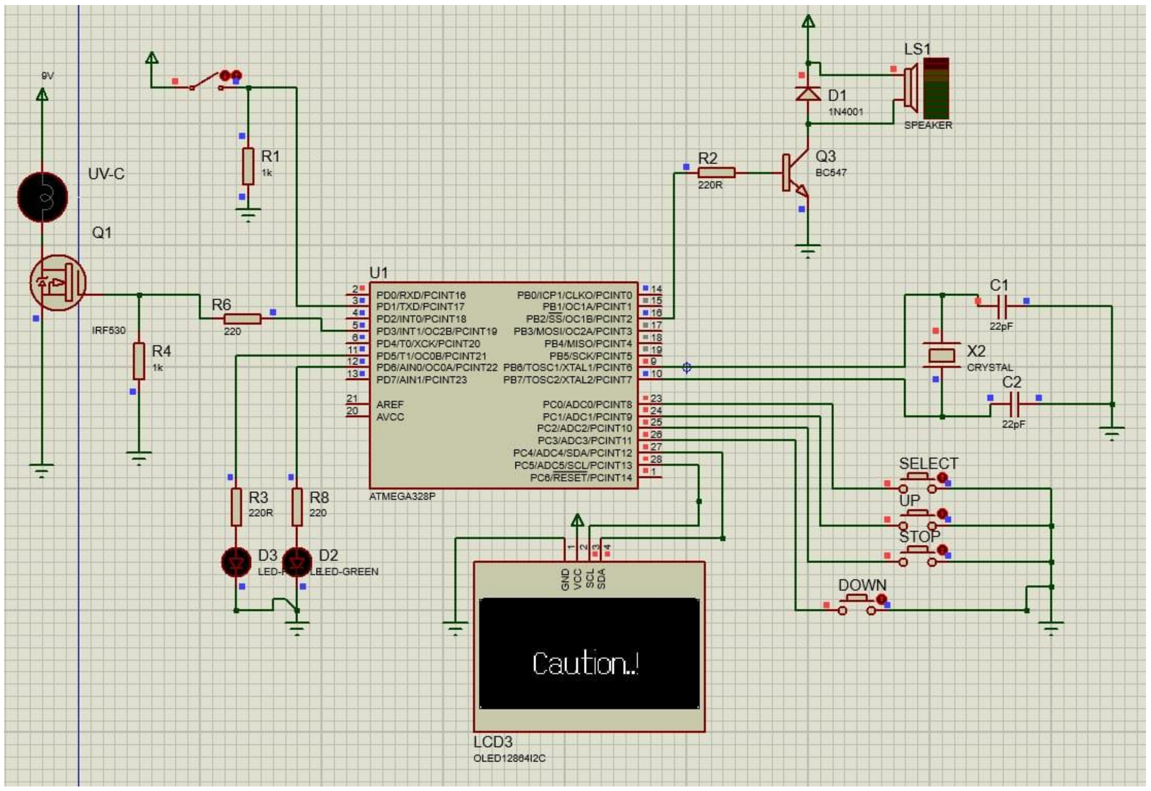


Figure 3: When the Lid Gets Open Accidentally During the Process

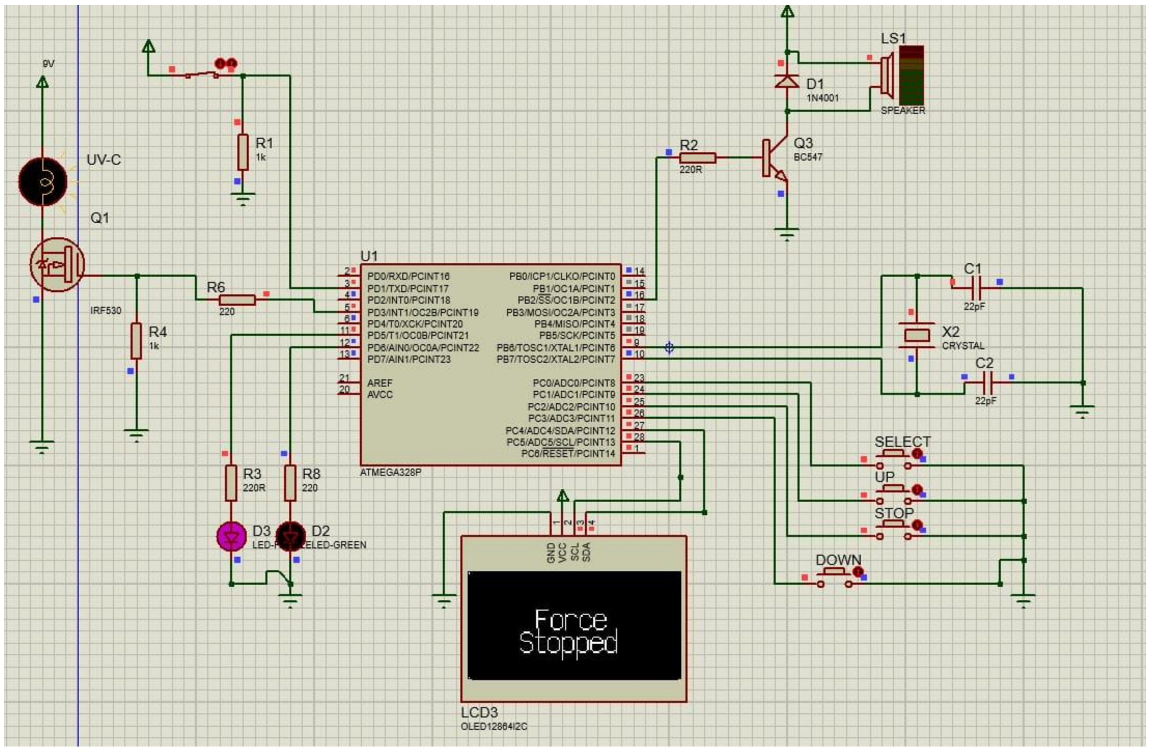
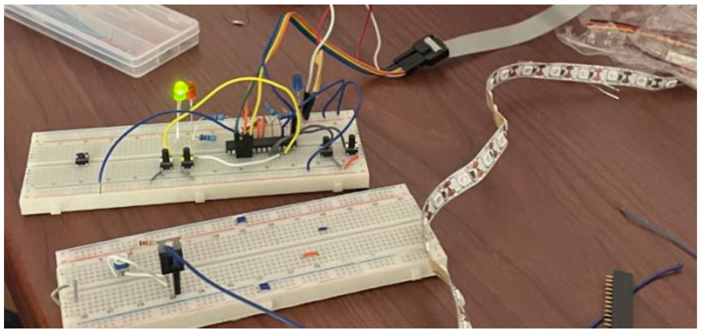
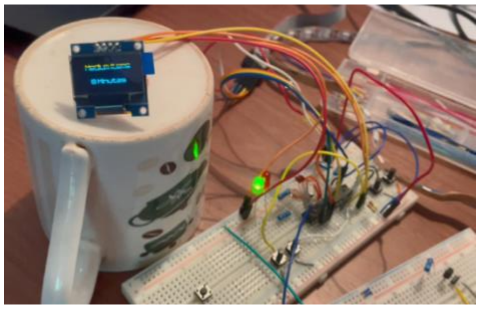


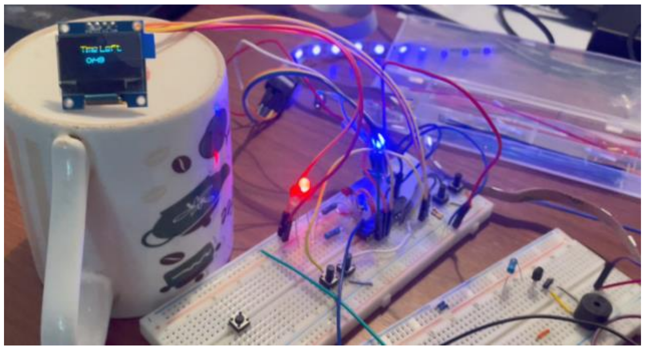
Figure 4: When the Stop Button is Pressed

5. Prototype Testing

After doing the simulation, breadboard testing was started. First, a sample code was sent to the microcontroller and checked. After that, components were purchased and implemented the whole circuit in a breadboard.





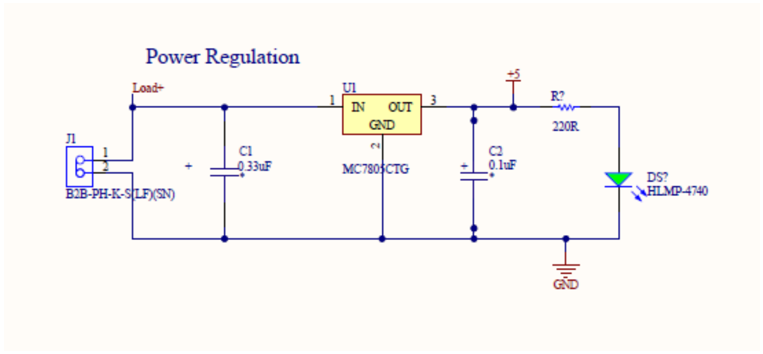


6. PCB

After a successful breadboard testing, PCB designing was started. It was decided to have two PCBs. The main PCB contains all the main components and a separate PCB for the keypad. The PCBs were designed using Altium software. It is essential to decide on the manufacturer before the designing stage since the design must meet the manufacturer's requirements. Due to time and cost constraints, we decided to go with a local manufacturer. Both PCBs are single layers. However, several jumpers were added to the main PCB.

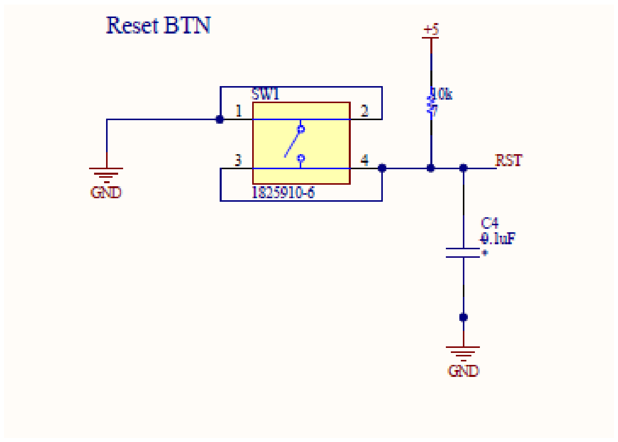
First, the schematic design was done. The schematic design was divided into several parts: power regulation, microcontroller unit, load driving unit, buzzer unit, and header connections. This division became very helpful in explaining the design to other group members.

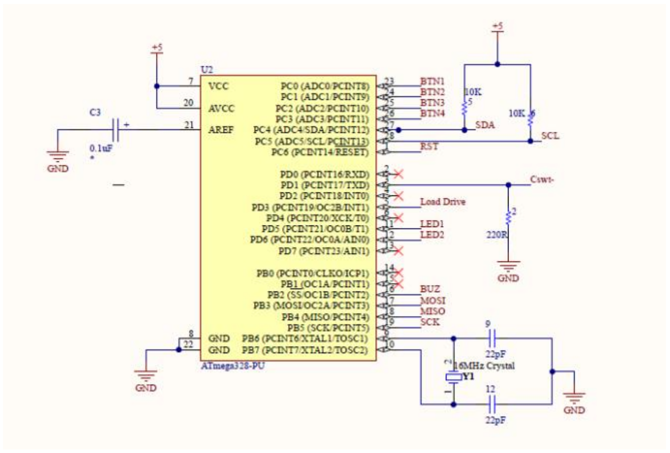
Power Regulation



Since the operating voltage of the ATmega 328p microcontroller is 5V, a 7805 regulator is used to obtain that requirement. 0.33uF and 0.1uF capacitors are added. However, the positive voltage for the load was provided before the regulation.

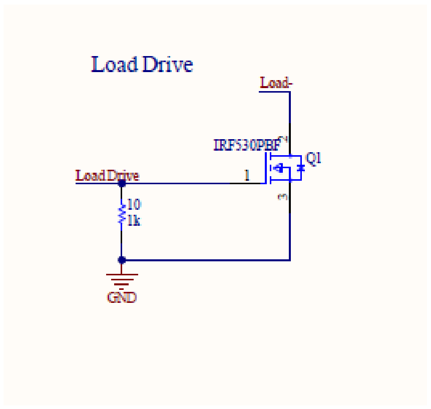
Microcontroller unit





Considering the requirements, ATmega 328P was taken as the microcontroller. The reset button was also added to the PCB to make the test process easier.

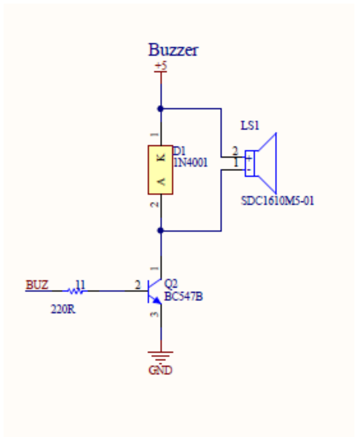
Load driving



The UV-led strip was used as the main load. Even if the operating voltage of that is 5V and the microcontroller can provide that voltage, it draws a comparatively high current. Drawing a high current through the microcontroller heats the circuit. Therefore, we are using IRF530 MOSFET to drive the load.

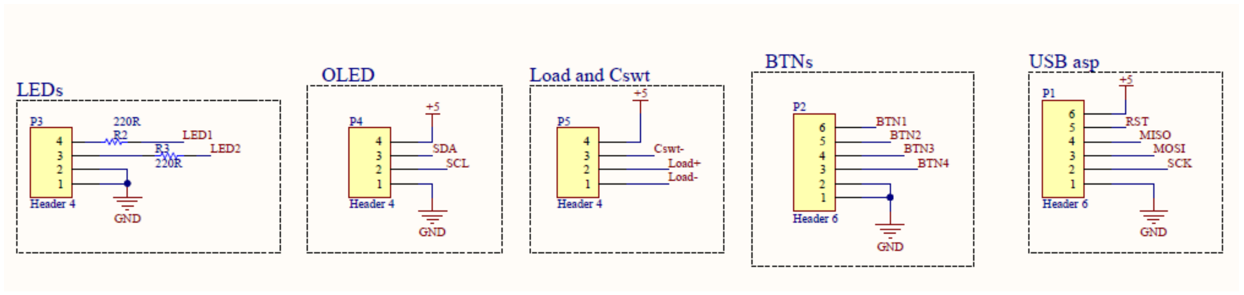
Buzzer

Since the current drawn through the microcontroller does not give enough sound, a BC547 transistor was used to drive the buzzer.

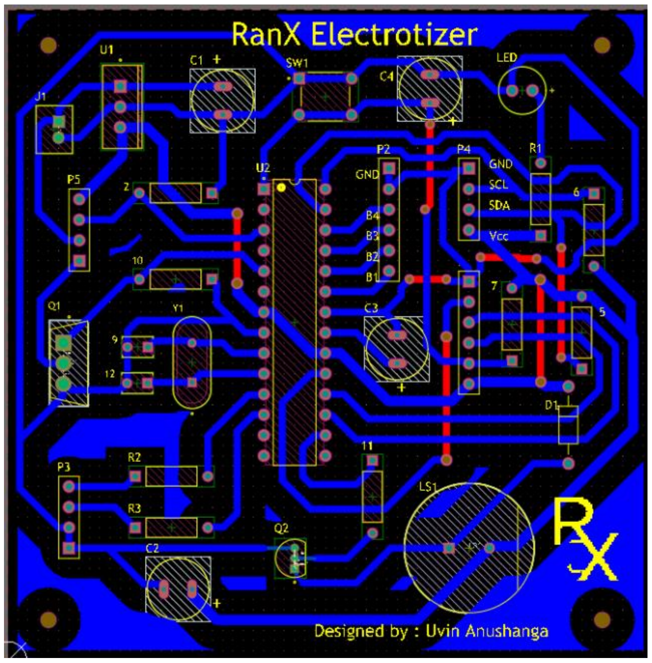


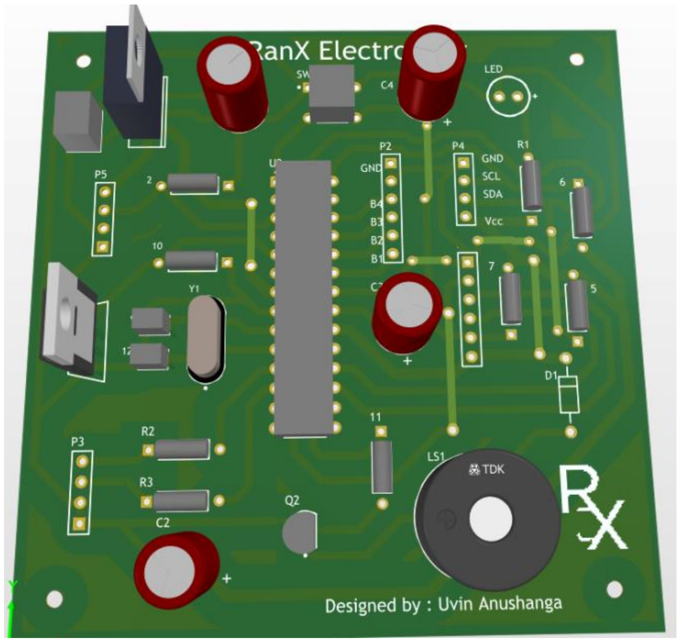
Header connections

Five separate headers were used for LEDs, OLED display, contact switch, load(UV strip), keypad, and the USBasp interface.

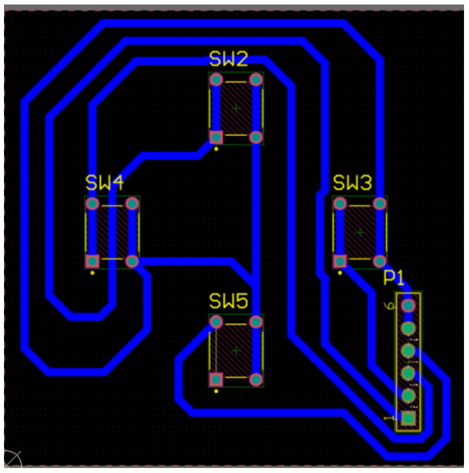


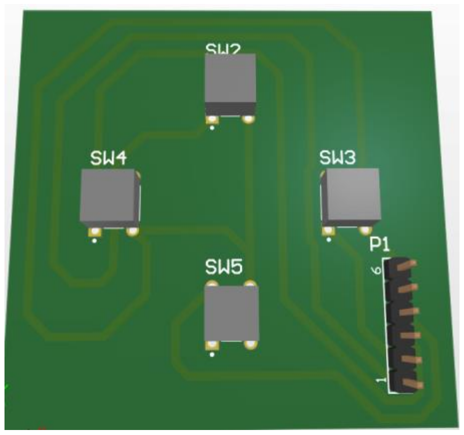
Layout design (main PCB)





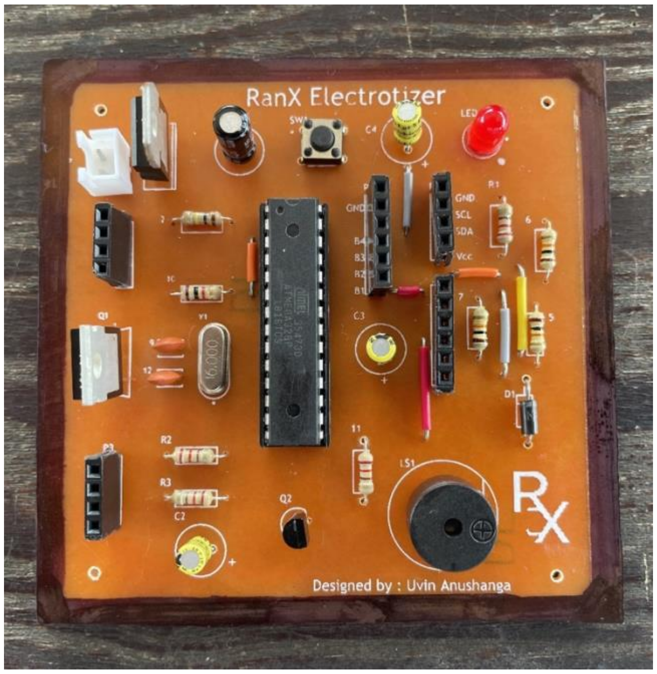
Layout design (Button PCB)

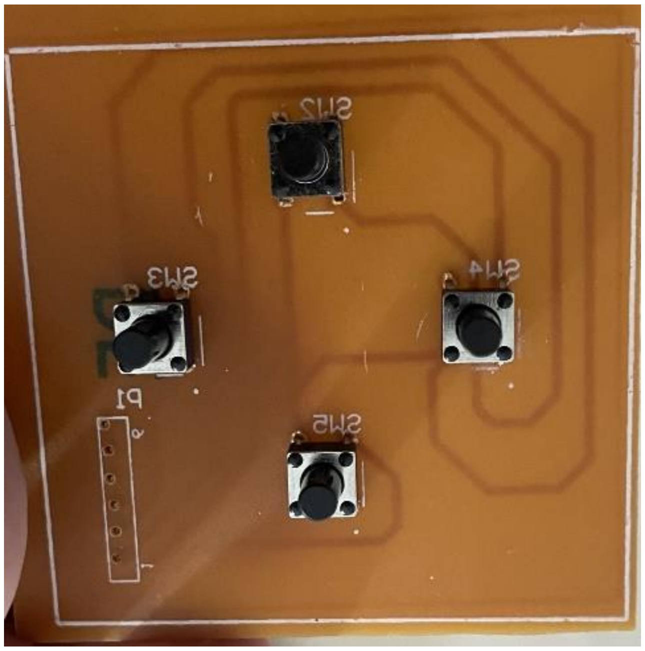




Since It was decided to go with local manufacturing, it is critically important that the manufacturing requirements are met in the layout design. The track width and the clearance had to be increased as much as possible. Otherwise, tracks may get removed in the etching process. 1.3mm width is used for power traces, and 1mm width is used for other traces. Both PCBs are single layers, and the routing is done on the bottom layer. Few jumpers are added in the main PCB. 1mm clearance was maintained between the traces, and 2mm clearance was kept between the atoms and the copper pour.

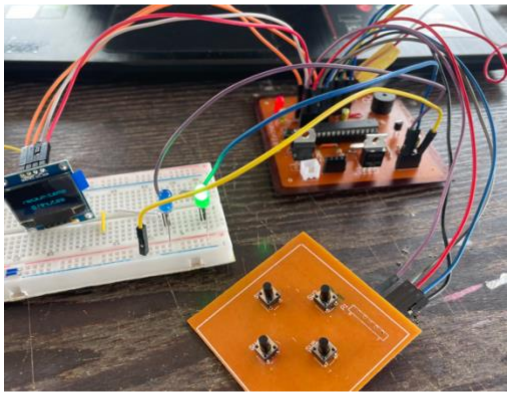
Soldered PCBs

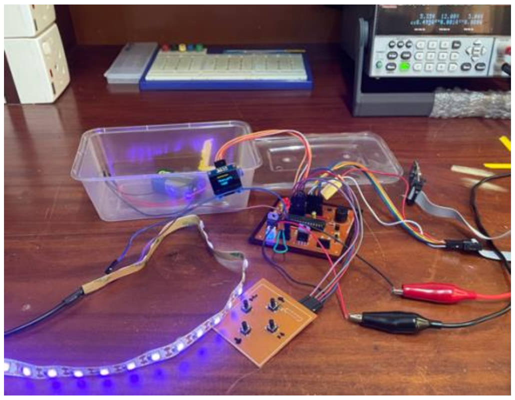




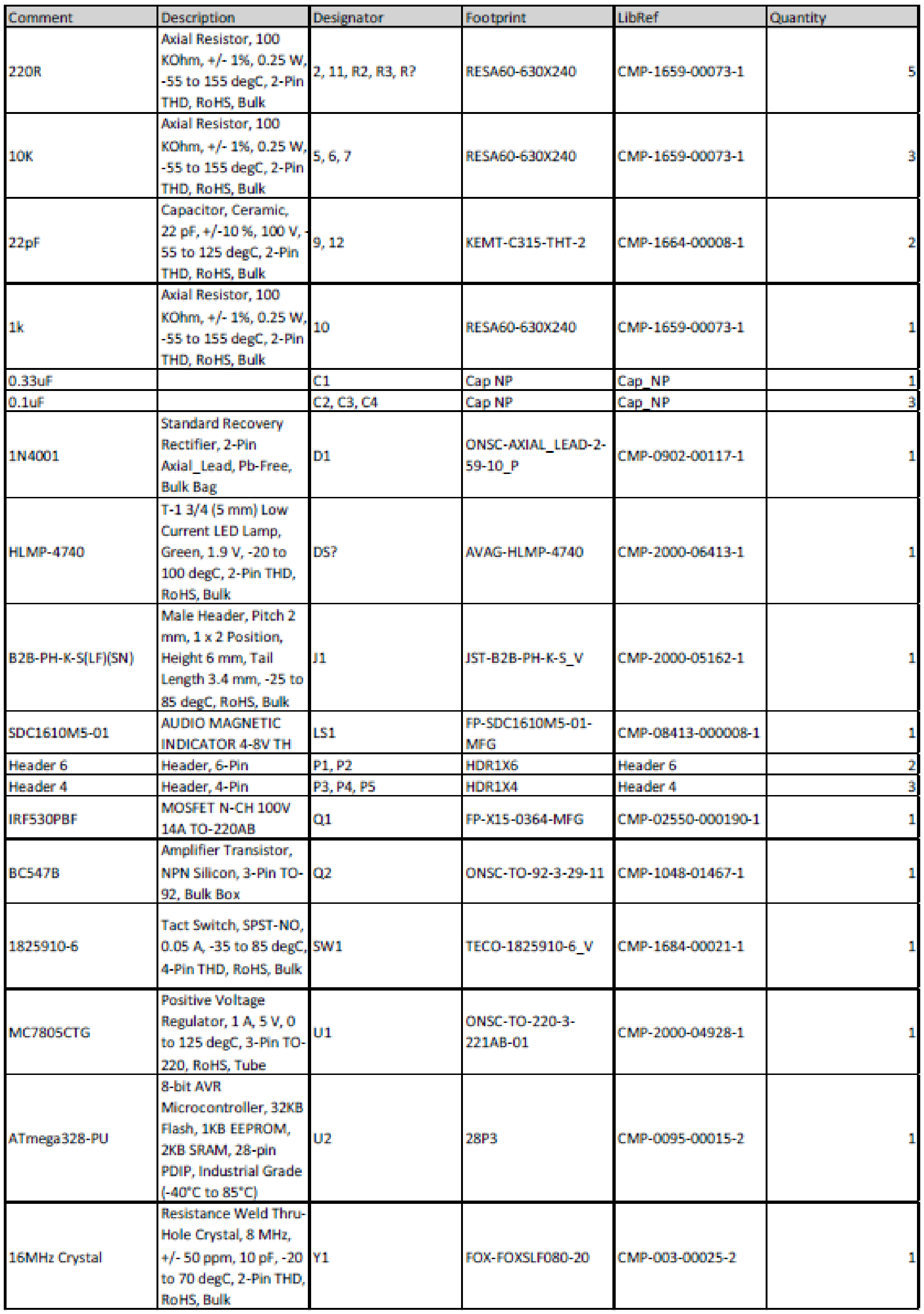
a. PCB Testing

After the soldering of the PCB, the product didn't work on the first try. There were some issues and loose connections. A few more rounds of testing were done to find and correct those errors.



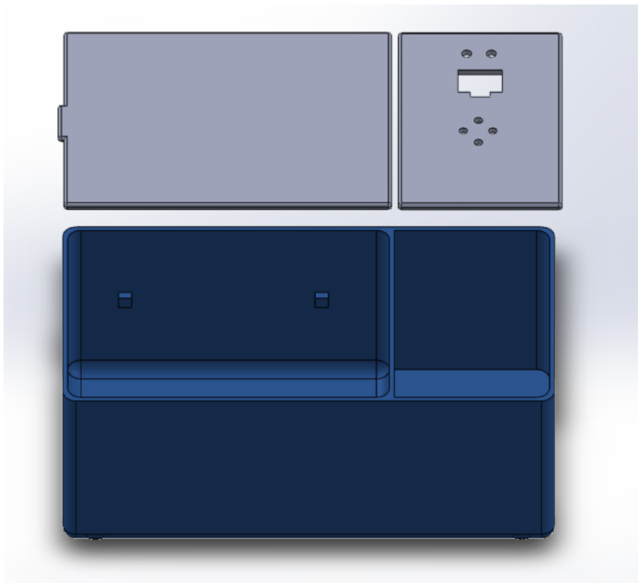


b. Bill of Materials



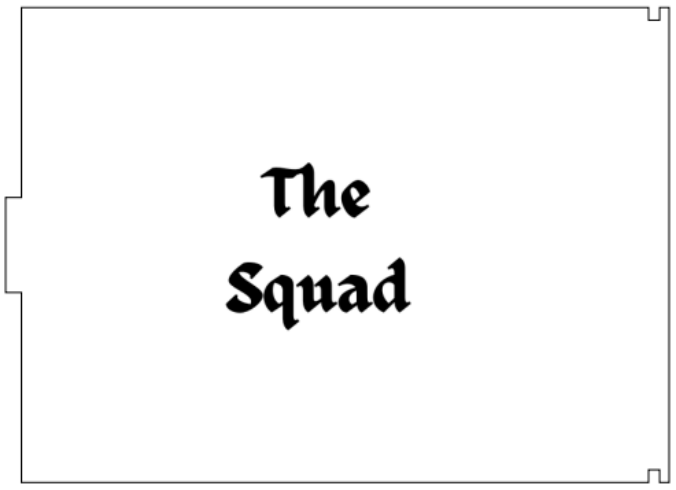
Comment	Description	Designator	Footprint	LibRef	Quantity
220R	Axial Resistor, 100 KOhm, +/- 1%, 0.25 W, -55 to 155 degC, 2-Pin THD, RoHS, Bulk	2, 11, R2, R3, R?	RESA60-630X240	CMP-1659-00073-1	5
10K	Axial Resistor, 100 KOhm, +/- 1%, 0.25 W, -55 to 155 degC, 2-Pin THD, RoHS, Bulk	5, 6, 7	RESA60-630X240	CMP-1659-00073-1	3
22pF	Capacitor, Ceramic, 22 pF, +/-10 %, 100 V, -55 to 125 degC, 2-Pin THD, RoHS, Bulk	9, 12	KEMT-C315-THT-2	CMP-1664-00008-1	2
1k	Axial Resistor, 100 KOhm, +/- 1%, 0.25 W, -55 to 155 degC, 2-Pin THD, RoHS, Bulk	10	RESA60-630X240	CMP-1659-00073-1	1
0.33uF		C1	Cap NP	Cap_NP	1
0.1uF		C2, C3, C4	Cap NP	Cap_NP	3
1N4001	Standard Recovery Rectifier, 2-Pin Axial_Lead, Pb-Free, Bulk Bag	D1	ONSC-AXIAL_LEAD-2-59-10_P	CMP-0902-00117-1	1
HLMP-4740	T-1 3/4 (5 mm) Low Current LED Lamp, Green, 1.9 V, -20 to 100 degC, 2-Pin THD, RoHS, Bulk	DS?	AVAG-HLMP-4740	CMP-2000-06413-1	1
B2B-PH-K-S(LF)(SN)	Male Header, Pitch 2 mm, 1 x 2 Position, Height 6 mm, Tail Length 3.4 mm, -25 to 85 degC, RoHS, Bulk	J1	JST-B2B-PH-K-S_V	CMP-2000-05162-1	1
SDC1610M5-01	AUDIO MAGNETIC INDICATOR 4-8V TH	LS1	FP-SDC1610M5-01-MFG	CMP-08413-000008-1	1
Header 6	Header, 6-Pin	P1, P2	HDR1X6	Header 6	2
Header 4	Header, 4-Pin	P3, P4, P5	HDR1X4	Header 4	3
IRF530PBF	MOSFET N-CH 100V 14A TO-220AB	Q1	FP-X15-0364-MFG	CMP-02550-000190-1	1
BC547B	Amplifier Transistor, NPN Silicon, 3-Pin TO-92, Bulk Box	Q2	ONSC-TO-92-3-29-11	CMP-1048-01467-1	1
1825910-6	Tact Switch, SPST-NO, 0.05 A, -35 to 85 degC, 4-Pin THD, RoHS, Bulk	SW1	TECO-1825910-6_V	CMP-1684-00021-1	1
MC7805CTG	Positive Voltage Regulator, 1 A, 5 V, 0 to 125 degC, 3-Pin TO-220, RoHS, Tube	U1	ONSC-TO-220-3-221AB-01	CMP-2000-04928-1	1
ATmega328-PU	8-bit AVR Microcontroller, 32KB Flash, 1KB EEPROM, 2KB SRAM, 28-pin PDIP, Industrial Grade (-40°C to 85°C)	U2	28P3	CMP-0095-00015-2	1
16MHz Crystal	Resistance Weld Thru-Hole Crystal, 8 MHz, +/- 50 ppm, 10 pF, -20 to 70 degC, 2-Pin THD, RoHS, Bulk	Y1	FOX-FOXSLF080-20	CMP-003-00025-2	1

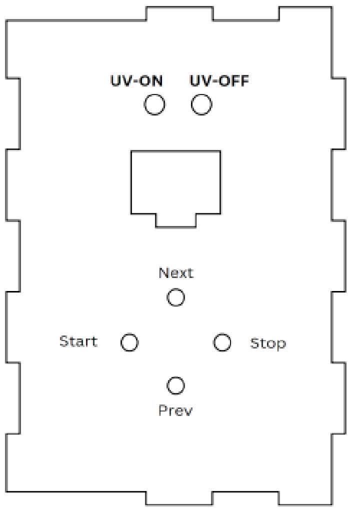
7. Enclosure



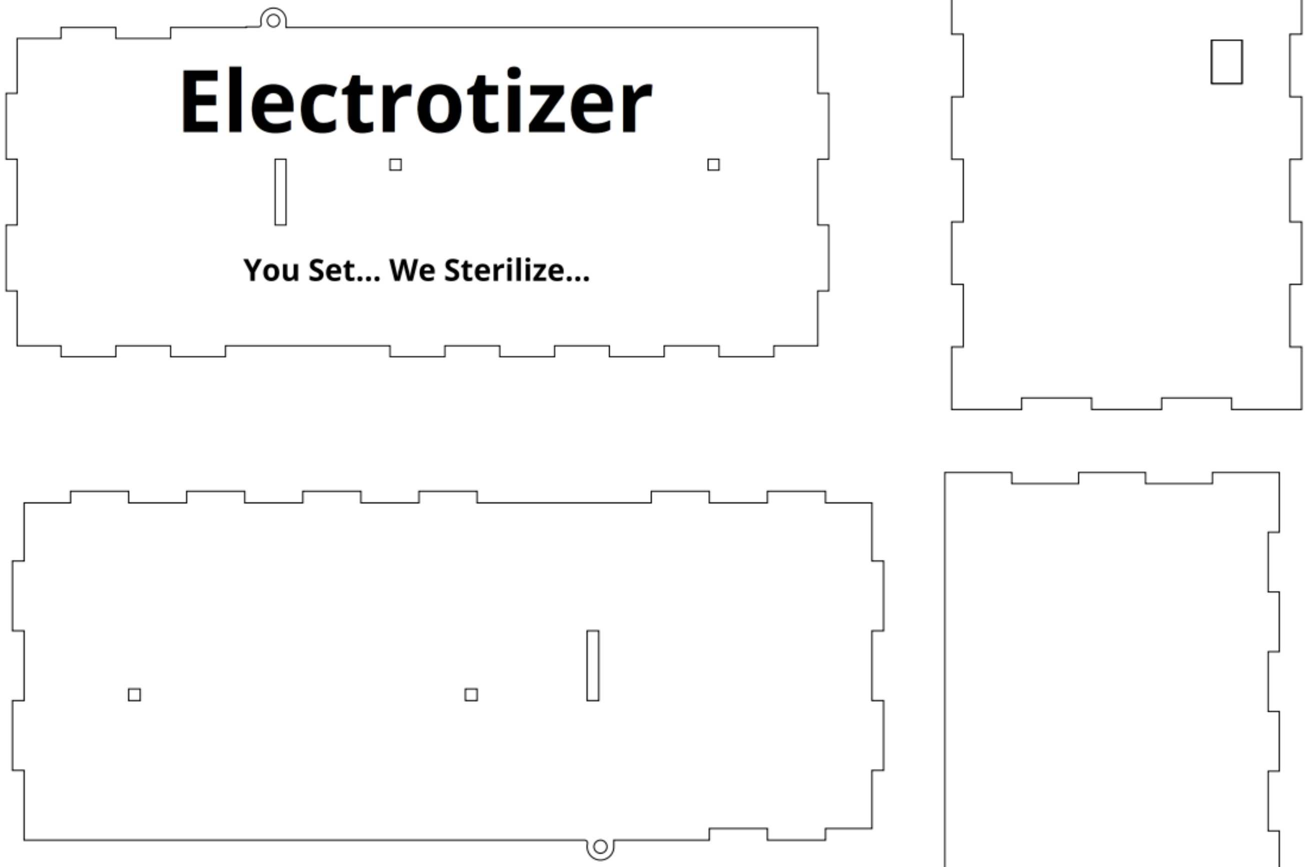
The enclosure design has 30x15x10 cm^3 dimensions. Therefore, the Laser-cut method with wood material was used due to the high cost of the 3D printing method, but, Manufacture requested an Illustrator design file for Laser cutting. Thus, the final enclosure design was done using Illustrator software.

Top parts

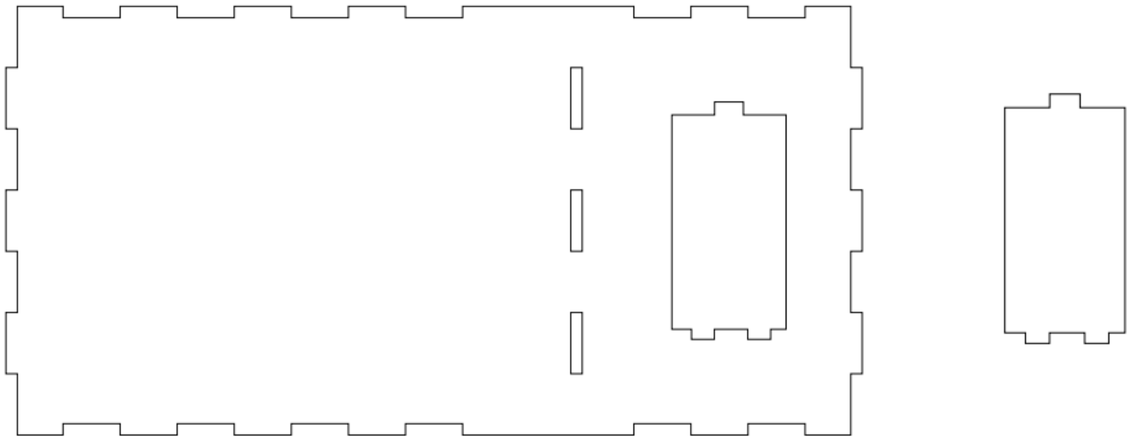




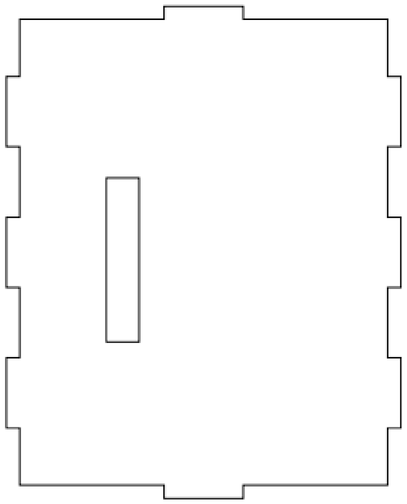
Side parts



Bottom parts

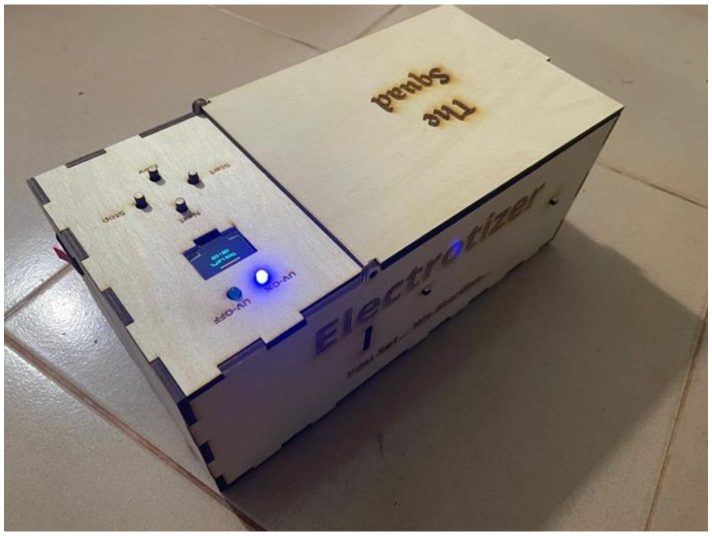


Compartment separation part



Printed enclosure

The opening – The closing lid is connected using a hinge, allowing it to rotate 180 degrees. This enclosure design has two compartments. One is for placing the device that needs to be sterilised, and the second compartment includes PCs, batteries, displays, etc. The top user interface panel is removable for the maintenance of the circuits. The battery compartment is mounted at the bottom of the second main compartment. Therefore batteries can be replaced easily.



8. Marketing and Branding

a. Marketing Strategy

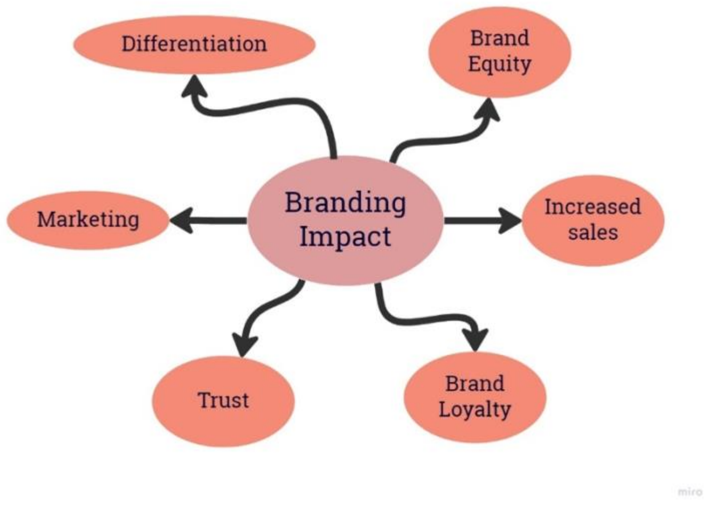
Creating a need for using UV sterilisers in everyday activities and marketing a related product can be done through a variety of strategies, such as:

1. **Highlighting the benefits of UV sterilisation**: Emphasizing the benefits of UV sterilisation, such as its ability to kill germs and bacteria, can help to create a sense of need for the product, especially for people who are concerned about maintaining a clean and healthy environment.
2. **Focusing on specific scenarios**: Identifying specific methods where UV sterilisation can be beneficial, such as in the kitchen, bathroom, or for personal items like phones, can help to create a sense of need for the product in those specific areas.
3. **Creating awareness of potential hazards**: Educating consumers about the dangers of germs and bacteria and how UV sterilisation can help mitigate those hazards can develop a sense of need for the product.
4. **Promoting UV sterilisation as a proactive solution**: Positioning UV sterilisation as a proactive solution to maintaining a clean and healthy environment, rather than just a reactive one, can help to create a sense of need for the product.
5. **Creating a sense of urgency**: Highlighting the fact that germs and bacteria can spread quickly and that UV sterilisation can be an effective way to mitigate that spread can create a sense of urgency and need for the product.
6. **Creating a Personalized Experience**: Creating a personalised experience for the customer, for example, creating a UV Sterilizer with a phone holder to sterilise the phone as well, could be a unique selling point and build a sense of need for the customer.

We believe the above marketing strategies would create the necessary market for our product.

b. Product Branding

Branding is an essential aspect of any electronic product because it helps to differentiate the product from its competitors, build trust with customers, and create a lasting impression in the minds of consumers.



We are deploying our product under the brand **RanX**. It's a brand that we created. Instead of calling the product a UV steriliser, we have called it "**The Electrotizer**". The name implies that it is an electronic sanitiser. Since the name "sanitiser" is more familiar to the public, we have decided to go with that name.

The Brand

RanX

You decide.. We create..

Product Name

ELECTROTIZER

YOU SET.. WE STERILIZE..

Appendix

a. The User Manuel

Introduction

Electrotizer is a compact and versatile UV-C LED disinfection case. It is perfect for quickly disinfecting high-touch items in the home, office, retail environment, or on the go. Feel safe with Electrotizer knowing you use innovative disinfection technology to protect against COVID-19 and other harmful viruses and bacteria.

Safety Precautions

- Unintended use of the device, or damage to the housing, may result in exposure to ultraviolet radiation.
- Avoid getting your eyes or skin exposed to UV light in case of damage to housing since it can cause irritations. (The product is designed to switch off the UV light when the lid is open)
- Do not open the device while the sanitisation process is happening since ozone created due to UV light is harmful to breathe.

User Instructions

- Turn on the device using the on-off button located on the side of the device.
- Wait until the OLED display is powered up and the green LED is turned on.
- The Next, Previous, Start and Stop buttons can be used to navigate the user interface shown in the OLED display.
- Select the required type of disinfection from the following list shown in the display.
  - Small items (6 mins) - Plastics, Paper, Wood etc.
  - Medium items (8mins) - Metallic items
  - Large items (10 mins) - When using several things at once.
- Open the lid of the sanitisation compartment and place the device on the net.
- Make sure to close the lid first.
- Use the start button to turn the process on.
- Use the stop button to stop the sanitisation in an emergency case and start the process from the beginning.

Maintenance and Troubleshooting

- The battery compartment available on the bottom side of the device can be used to access the battery. If the battery is low (the green light is dim), remove the two batteries and recharge them.
- Turn on the device using the on-off button if you are not using it.

Technical Specifications

• Dimensions	- 20cm X 15cm X 12cm
• Wight	- 520g
• UV-C light wavelength	- 260nm-280nm
• Voltage	- 7.4VDC
• Battery life	- 10 hours of sanitisation per charge
• LED lifespan	- up to 10K hours

Applications

- Reception
- Office
- Laboratories
- Hospitals
- Vehicles
- Domestic use

b. Code

```
/*
 * GccApplication1.cpp
 *
 * Created: 11/8/2022 9:04:55 PM
 * Author : Lahiru Shyamal
 */

#define F_CPU 16000000UL

#include <avr/io.h>
#define __DELAY_BACKWARD_COMPATIBLE__
#include <util/delay.h>

#include <u8g2.h>
#include <stdio.h>
#include <u8x8_avr.h>
#include <string.h>


#define SSD1306_ADDR  0x78

#include "noteFreq.h"
#include "tone.h"

u8g2_t u8g2;

int type = 8*60;
uint8_t minute ;
uint8_t second ;

void updateMenu();
void UV_light();
char res[100];

void play_melody(int melody[],int tempo, int notes );


#define SPEAKER PORTB2



int main (void)
{
    DDRC = 0x00;
    DDRD = 0xfe;


    DDRB = 0x05;
  PORTC = 0xff;
    PORTD = 0x01;


     u8g2_Setup_ssd1306_i2c_128x64_noname_f(&u8g2, U8G2_R0, u8x8_byte_avr_hw_i2c,
u8x8_avr_delay);
    u8g2_SetI2CAddress(&u8g2, SSD1306_ADDR);
    u8g2_InitDisplay(&u8g2);
    u8g2_SetPowerSave(&u8g2, 0);

    u8g2_ClearBuffer(&u8g2);
    u8g2_SetFont(&u8g2, u8g2_font_calibration_gothic_nbp_tr);
    u8g2_SetFontRefHeightText(&u8g2);
    u8g2_SetFontPosTop(&u8g2);
    u8g2_DrawStr(&u8g2, 40, 20, "RanX ");
    u8g2_DrawStr(&u8g2, 20, 40, "Electrotizer");
    u8g2_SendBuffer(&u8g2);
    _delay_ms(2000);

    u8g2_ClearBuffer(&u8g2);
    u8g2_SetFont(&u8g2, u8g2_font_smart_patrol_nbp_tr);
    u8g2_SetFontRefHeightText(&u8g2);
    u8g2_SetFontPosTop(&u8g2);
    u8g2_DrawStr(&u8g2, 10, 15, "Designed by");
    u8g2_DrawStr(&u8g2, 10, 40, "...The Squad...");
    u8g2_SendBuffer(&u8g2);
    _delay_ms(2000);

while (1)
    {
        PORTD |= (1<<PORTD6);
        PORTD &= ~(1<<PORTD3);
        PORTD &= ~(1<<PORTD5);

    if(~PINC& 0x01){
        PORTD &= ~(1<<PORTD6);
        //play_melody(got_melody, got_tempo,got_notes);
        UV_light();
         }
    if(~PINC& 0x02)
    {
        _delay_ms(500);
        type = type + 120;
        if(type > 60*10){type=60*10;}
        updateMenu();

    }

    if(~PINC& 0x08)
    {
        _delay_ms(500);
        type = type - 120;
        if(type < 60*6){type=60*6;}
        updateMenu();
    }

    updateMenu();


    }
}


void UV_light(){
    int temp = 0;

    while(temp <= type)
    {
        int i = 0;
        while (i<10){

            if ((~PIND & (1<<PIND1))) // Opening the LID
            {
                PORTD &= ~(1<<PORTD3);
                PORTD &= ~(1<<PORTD5);

                    // Sending information to LCD to print
                    u8g2_ClearBuffer(&u8g2);
                    u8g2_SetFont(&u8g2, u8g2_font_calibration_gothic_nbp_tr);
                    u8g2_DrawStr(&u8g2, 30, 30, "Caution..!");
                    u8g2_SendBuffer(&u8g2);
                    // LCD ...........
            }
            else if(~PINC& (1<<PINC2)) // Forced Stop button
            {

                    // Sending information to LCD to print
                    u8g2_ClearBuffer(&u8g2);
                    u8g2_SetFont(&u8g2, u8g2_font_calibration_gothic_nbp_tr);
                    u8g2_DrawStr(&u8g2, 40, 20, "Force ");
                    u8g2_DrawStr(&u8g2, 30, 35, "Stopped");
                    u8g2_SendBuffer(&u8g2);
                    // LCD ...........
                    _delay_ms(1000);
                return;
            }
            else   // running through the given time
            {
                PORTD |= (1<<PORTD3);
                PORTD |= (1<<PORTD5);
                _delay_ms(100);
                i=i+1;
            }
        }

        temp = temp+1;
        minute = (type-temp)/60;
        second = (type-temp)%60;
        if (second<10){sprintf(res, "0%d : 0%d", minute,second);}

        else{sprintf(res, "0%d : %d", minute,second);}

        // Sending information to LCD to print
            u8g2_ClearBuffer(&u8g2);
            u8g2_SetFont(&u8g2, u8g2_font_smart_patrol_nbp_tr);
            u8g2_DrawStr(&u8g2, 30, 5, "**********");
            u8g2_DrawStr(&u8g2, 30, 20, "Time Left");
            u8g2_DrawStr(&u8g2, 40, 40, res);
            u8g2_SendBuffer(&u8g2);
       // LCD ...........

    }

    PORTD &= ~(1<<PORTD3);
    PORTD &= ~(1<<PORTD5);
    if (type+1==temp){

        // Sending information to LCD to print
```

```
        u8g2_ClearBuffer(&u8g2);
        u8g2_SetFont(&u8g2, u8g2_font_smart_patrol_nbp_tr);
        u8g2_DrawStr(&u8g2, 50, 20, "O");
        u8g2_DrawStr(&u8g2, 40, 35, "/ | )");
        u8g2_DrawStr(&u8g2, 41, 45, "/  |..........");
        u8g2_SendBuffer(&u8g2);
        // LCD ............

        play_melody(got_melody, got_tempo,got_notes);
        }
    return;

}




void updateMenu(){
    switch(type){

        case 6*60:
            // Sending information to LCD to print
            u8g2_ClearBuffer(&u8g2);
            u8g2_SetFont(&u8g2, u8g2_font_smart_patrol_nbp_tr);
            u8g2_DrawStr(&u8g2, 30, 20, "Small Items");
            u8g2_DrawStr(&u8g2, 30, 40, "6 Minutes");
            u8g2_SendBuffer(&u8g2);
            // LCD ............
        break;

        case 8*60:                          //Third display state
            // Sending information to LCD to print
            u8g2_ClearBuffer(&u8g2);
            u8g2_SetFont(&u8g2, u8g2_font_smart_patrol_nbp_tr);
            u8g2_DrawStr(&u8g2, 20, 20, "Medium Items");
            u8g2_DrawStr(&u8g2, 30, 40, "8 Minutes");
            u8g2_SendBuffer(&u8g2);
            // LCD ............
        break;


        case 10*60:
            // Sending information to LCD to print
            u8g2_ClearBuffer(&u8g2);
            u8g2_SetFont(&u8g2, u8g2_font_smart_patrol_nbp_tr);
            u8g2_DrawStr(&u8g2, 20, 20, "Large Items");
            u8g2_DrawStr(&u8g2, 30, 40, "10 Minutes");
            u8g2_SendBuffer(&u8g2);
            // LCD ...........;
        break;

    }
    return;
}



void play_melody(int melody[],int tempo, int notes) // function to play any melody
{



    int wholenote = (60000 * 4) / tempo;

    int divider = 0, noteDuration = 0;
    // iterate over the notes of the melody.
    for (int thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {
        divider = melody[thisNote + 1];
        if (divider > 0) {
            noteDuration = (wholenote) / divider;  // regular note, just proceed
        }
        else if (divider < 0) {
               noteDuration = (wholenote) / (-1*(divider));// dotted notes are
represented with negative durations!!
           noteDuration *= 1.5; // increases the duration in half for dotted notes
        }


            PORTB |= (1<<PORTB0);
             tone(SPEAKER, melody[thisNote], noteDuration * 0.9); //play the note
for 90% of the duration, leaving 10% as a pause
            PORTB &= ~(1<<PORTB0);
            _delay_ms(noteDuration);

            noTone(SPEAKER);

    }
    return;
}
```