

# Department of Electronic & Telecommunication Engineering

UNIVERSITY OF MORATUWA



EN1093 : LABORATORY PRACTICE I

---

## Digital Alarm Clock

---

Group 25

*Team Members*

190622R O.K.D. Tharindu  
190626H P.D. Tharundi  
190631T T.A. Thenuwara  
190636M A.G.N. Udara

*Course Instructor*

Randima Senanayake

## **Abstract**

As the final assessment of the EN1093 module, we were given the task of designing a Digital alarm clock. Our Digital alarm clock is a standard Alarm clock with all necessary functionalities and including a simple Timer functionality. We were instructed to use AVR C++ for firmware development and, Proteus for simulations. Also, to use Atmega328P Microcontroller, 16x2 LCD display and for the RTC IC, DS1307 or DS3231. For sound output, we used a simple active piezo buzzer. This document contains the process of creating our Alarm clock, issues we faced, challenges, functionalities, firmware details, and technical specifications.

We also attached a 3D object of our Enclosure design, be sure to use Adobe reader DC for a good experience.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Goal . . . . .	1
<b>2</b>	<b>Method</b>	<b>2</b>
2.1	Steps of the Project . . . . .	2
2.2	Components and their Algorithms . . . . .	2
2.2.1	The RTC Module . . . . .	2
2.2.2	Buzzer . . . . .	3
2.2.3	LCD Display . . . . .	4
2.2.4	Keypad input . . . . .	6
2.3	Features and Overall working Methodology . . . . .	7
2.3.1	Displaying Time and date . . . . .	7
2.3.2	Selecting sub menus and features. . . . .	7
2.3.2.1	SET ALARM . . . . .	7
2.3.2.2	SET TIME . . . . .	8
2.3.2.3	SEE ALARMS . . . . .	9
2.3.2.4	TIMER . . . . .	9
2.3.3	Alarm interrupting . . . . .	10
2.3.4	Factory resetting . . . . .	10
2.3.5	Microcontroller . . . . .	10
2.3.6	Programmer . . . . .	10
<b>3</b>	<b>Results</b>	<b>11</b>
<b>4</b>	<b>Discussion</b>	<b>13</b>
4.1	Problems and challenges we faced during this project . . . . .	13
4.2	Overall impression and possible improvements . . . . .	13
<b>5</b>	<b>Acknowledgement</b>	<b>14</b>
<b>6</b>	<b>Appendix</b>	<b>15</b>
6.1	The PCB . . . . .	15
6.2	Proteus schematics . . . . .	18
6.3	Enclosure . . . . .	19
6.4	AVR Codes . . . . .	21
<b>Bibliography</b>		<b>51</b>

# List of Figures

2.1	Steps . . . . .	2
2.2	Time and date . . . . .	7
2.3	Setting An Alarm LCD Menus . . . . .	8
2.4	Changing LCD Menus . . . . .	8
2.5	Available Alarm list LCD Menus . . . . .	9
2.6	Timer LCD Menus . . . . .	9
6.1	Schematics Design . . . . .	15
6.2	PCB Design . . . . .	16
6.3	BOM . . . . .	17
6.4	Proteus Simulation . . . . .	18
6.5	3D Enclosure View . . . . .	19
6.6	Transparent Enclosure views . . . . .	20

# List of Algorithms

6.1	Main.cpp . . . . .	21
6.2	Alarm.h . . . . .	22
6.3	Alarm.cpp . . . . .	22
6.4	Buzzer.h . . . . .	27
6.5	Buzzer.cpp . . . . .	28
6.6	pitches.h . . . . .	30
6.7	melodies.h . . . . .	31
6.8	Display.h . . . . .	35
6.9	Display.cpp . . . . .	35
6.10	Keypad.h . . . . .	40
6.11	Keypad.cpp . . . . .	41
6.12	ds1307.h . . . . .	42
6.13	ds1307.cpp . . . . .	42
6.14	i2cmaster.h . . . . .	43
6.15	i2cmaster.cpp . . . . .	46

# 1 Introduction

To utilize what we learned in semester 2 and also with the self-accumulated knowledge throughout the semester, we designed this Digital Alarm clock as a fulfilment for the final assignment of the EN1093 module. It is a fully functional and programmable digital alarm clock with Buzzer as sound output. We are using a DS1307 RTC module with a battery backup power input to store and manage time. We are using *I<sup>2</sup>C* protocol to manage the RTC module. Apart from that Atmega 328P will handle all other storage, processing and input/output tasks. 16 × 2 LCD display is used to display time, dates, alarms and other UI components such as menus and tones. A 4 × 3 keypad is used for the user inputs and UI controls. A separate button is added for the alarm interruptions. There is a Factory reset button located inside the enclosure which requires a pin to click it. RTC module, LCD display, Piezo Buzzer, keypad, reset and Stop buttons are directly connected to the Atmega 328P Micro Controller. For reprogramming the microcontroller, there is a serial input reserved in the PCB.

## 1.1 Goal

Designing a Digital Alarm clock, with all the conventional features and minimalist design to reduce the cost but keeping a good user experience.

## 2 Method

### 2.1 Steps of the Project

Using teamwork and step by step process, we were able to finish this project. These are the main steps we followed

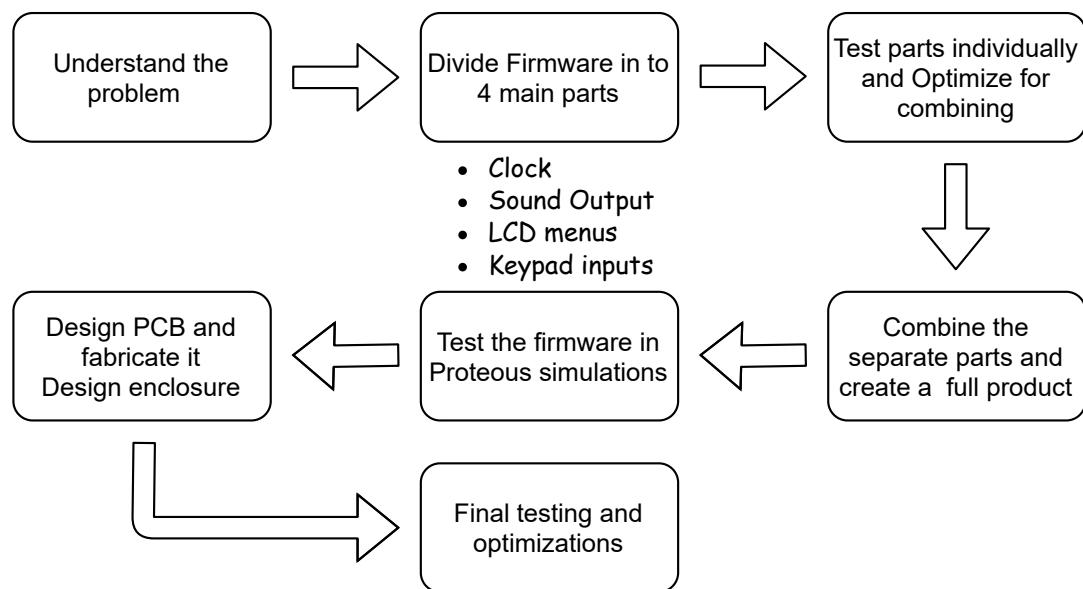


Figure 2.1: Steps

After printing the PCB, the final stage of testings were started. All the components and PCB details is available in the Appendix

### 2.2 Components and their Algorithms

Here we will look at the main components and how the algorithms and codes work.

#### 2.2.1 The RTC Module

The DS1307 RTC module is working with the I2C protocol with an inbuilt battery chamber so even when the power is not available, The date and time data will be preserved. Using SCL and SDA pins to connect to the microcontroller. The module is temperature dependable so keeping it separated from the main PCB was a solution for the heating. The I2C libraries that we implemented were created by Peter Fleury. The chip 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.

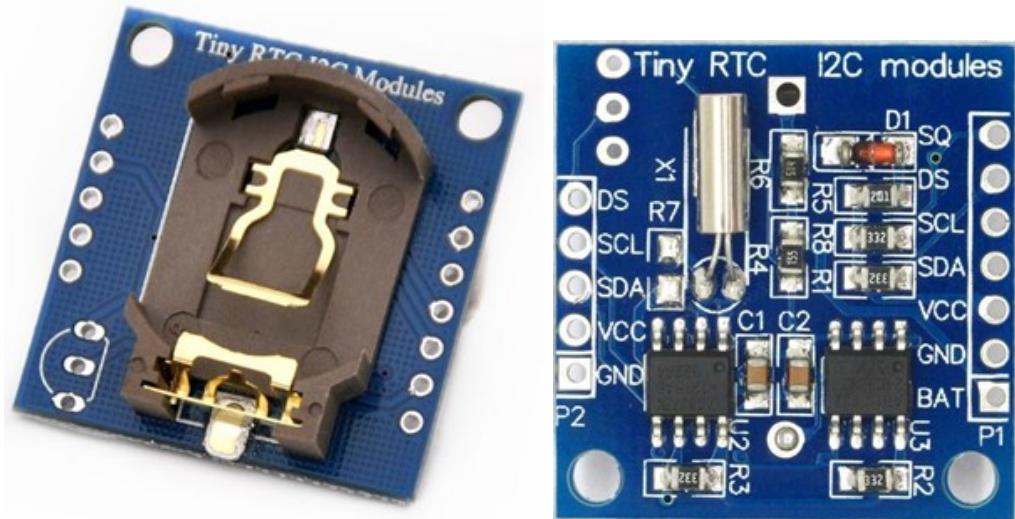
## Features

- 32KHz Crystal oscillator
- 3V CR2032 battery backup
- 32 bytes 24C32 EEPROM

## Algorithm

With I2C address 0xD0. The module is taking and stores data in 24-hour format.

- ds1307\_setdate() Function takes year, month, day, day of week, hour, minute and second in decimal form and send the data to RTC module to rewrite current information.
- ds1307\_getdate() Function is used to pull the above information back to the microcontroller.



### 2.2.2 Buzzer

The Active piezo buzzer is the Sound output device in the Alarm Clock and it used Square waves to generate tones. The pitches header file contains the keys and their designated frequencies. Using this pitch data, time ratios and provided tempo data given in the melodies header file will be used to generate tone sequences. The negative time ratio sequences indicate extended tones. Positive ones are shorter. There are 5 songs and these will be used to trigger the alarm alerts. This data is stored in Program memory to utilize the storage properly. We will use the PD2 pin to connect The buzzer to Atmega 328P. The melodies are adapted from the works of Robson Couto.

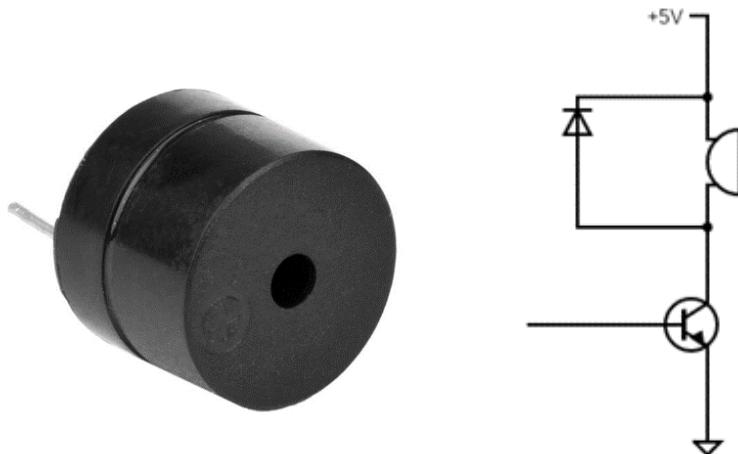
## Features

- Active Buzzer
- Using amplifier for better sound volume created with a BJT
- The diode across the buzzer prevent sudden voltage spikes and protect the BJT

## Algorithm

To generate a tone out of the buzzer, it needs a signal with that specific frequency. Using simple delays and toggling the output pin we can archive this while generating 50% duty cycle square waves. Since this part is also working as the Alarm output, to implement an interrupt to the system we will use external interrupt pin PD3 to utilize this. Upon the rising edge high, the tone playing function will stop.

- Play\_Note() Function is used to generate 50% duty square waves when the relevant frequency and the time duration is given. First, generate the time duration per cycle and then calculate the number of cycles. For each cycle turn 50% of time output high and then turn output low for the remaining 50% cycle. Repeat this for all cycles. If the input frequency was 0, then the output will be 0 for the given duration.
- Play() function will take single integer input which is the song number. After that reading values from the melodies header, it will take notes and duration and trigger Play\_Note() function with the predetermined length and tempo will be used to generate the tone duration. The external interrupt button can be used to stop this function and stop the sound from playing.



### 2.2.3 LCD Display

This 16-character, 2-line parallel liquid crystal display achieves a large viewing area in a compact package. The DDRAM address 0x00 corresponds to the first character of the top line, address 0x0F corresponds to the last character of the top line, address 0x40 corresponds to the first character of the second line, and address 0x4F corresponds to the last character of the second line.

The main functionality of the LCD is to display

- Time/Date
- Control menus
- Tone selections

We are utilizing maximum contrast by grounding the  $V_0, V_{EE}$  pin and 4bit configuration mode. Total 6 pins are connected with the microcontroller using a male-female connector. The datasheet will be provided.

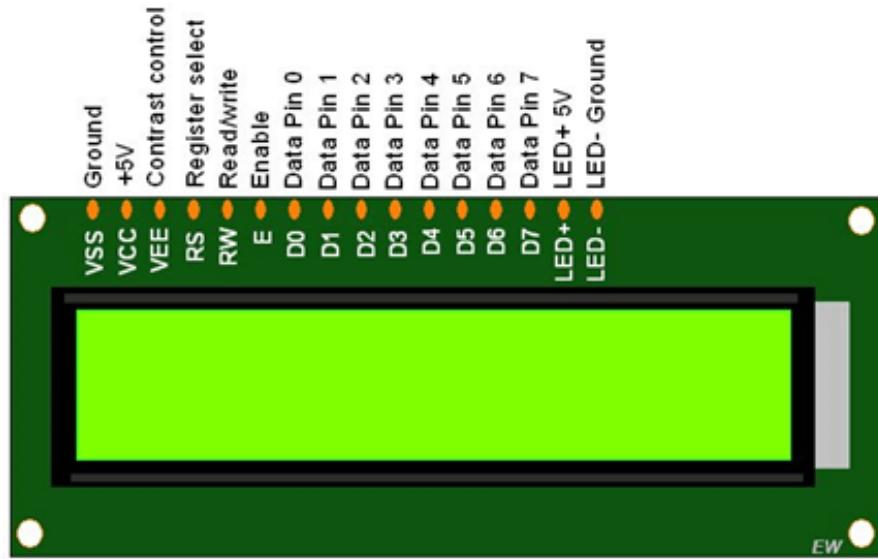
## Features

- Using 4Bit mode to utilize pins efficiently
- Blue backlight
- Looks Stylish
- Always-on display

## Algorithm

All the menus and content displaying will be done by this algorithm.

- `displayTime()` will be used to display the time and date on the main screen. The data for this display option will be taken from the RTC module..
- `LCD_SetAlarm()` Function is used to take user inputs from the 4x4 keypad and set one alarm at a time. To reach this point user will have to choose from “Set Alarm” from the listed menu.
- `LCD_Tone()` Function will display currently available Alarm tones. The alarm tone names are stored In an array named `tone_List`. You can select a tone and this function will give an integer output for the selected song index.
- `LCD_SetDate()` This function will grant the user to update system tie and date with the help of keypad inputs.



## 2.2.4 Keypad input

We are utilizing a  $4 \times 3$  matrix keypad for our input purposes. Using number keys as basic number inputs as well as for menu navigation

- 8 :- UP
- 2 :- DOWN
- #:-RETURN/ CANCEL
- \* :- ENTER/ SELECT

## Features

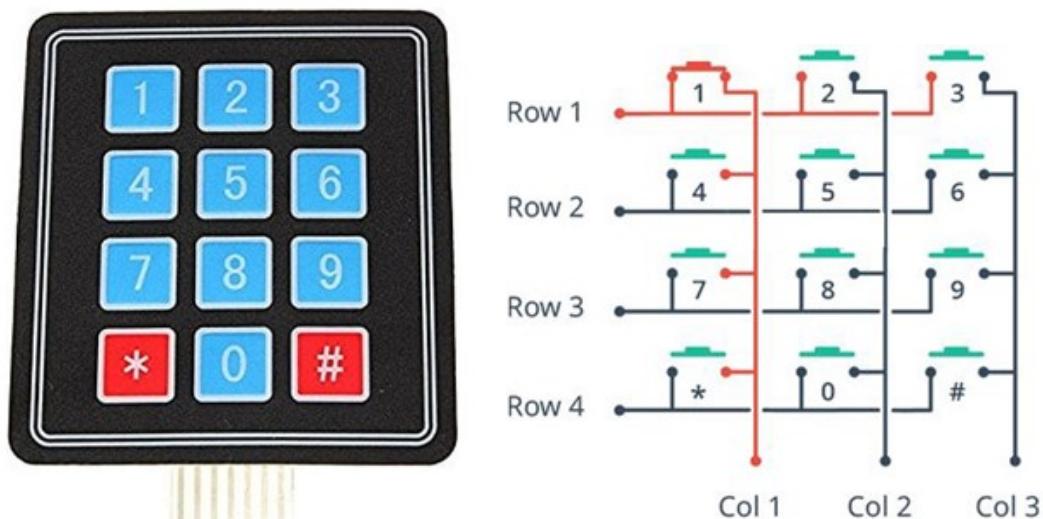
- Active Buzzer
- Using amplifier for better sound volume created with a BJT
- The diode across the buzzer prevent sudden voltage spikes and protect the BJT

use 7 pins for communication with the Microcontroller.

## Algorithm

In a nutshell, keypad works as a matrix of push buttons. There are 4 output pins (PC0 to PC3), and the inputs will be toggled in a sequence from top raw to bottom raw. In a instance if a button is pressed we can recognize that row with the toggled on pin. At instance of the button is pressed, that input pin will toggle 5V accordingly. There are 3 input pins (PB0 to PB2). They are connected using a pull down resistor. By the column and raw number, system can recognize the pressed button.

- btnPress() function will check the pins and ports and return the character given in the location reading the matrix data. We will utilize this to get properly control the inputs.



## 2.3 Features and Overall working Methodology

Here we will look at the features and how the overall algorithm work.

### 2.3.1 Displaying Time and date

After startup and after the Welcome screen, the date and time data will be constantly be pulling out of the RTC module. And with the LCD display, it will keep updating accordingly. This process doesn't have any specified delays. It will constantly do this process till a key input is given.

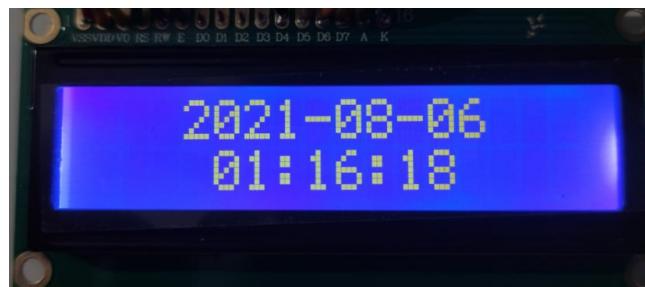


Figure 2.2: Time and date

After a key input, It will navigate to the First list containing

1. SET ALARM
2. SET TIME
3. SEE ALARMS
4. TIMER

### 2.3.2 Selecting sub menus and features.

Here we will check each scenario.

#### 2.3.2.1 SET ALARM

If you select *SET ALARM*, it will navigate to a new screen where you need to enter the alarm details and you can use the keypad numbers to enter the alarm time. The Alarm should be inserted in hh:mm order. After that pressing “ \* ” will take to the next stage to select an alarm tone. To navigate the list up and down, the 2 and 8 buttons are reserved. After choosing the alarm, alarm is set. The algorithm will update a list of arrays with the Alarm Time and Chosen Tone data. These data will be linked by an index.



Figure 2.3: Setting An Alarm LCD Menus

#### 2.3.2.2 SET TIME

If you select *SET TIME*, it will open a new screen to set alarm and time. Using keypad inputs date should be added in *yyyy/mm/dd* format and time in *hh : mm : ss* format. After setting time this data will be sent to RTC module with the `ds1307_setdate()` function.



Figure 2.4: Changing LCD Menus

### 2.3.2.3 SEE ALARMS

If you select *SEE ALARMS*, you can see all the currently available Alarms. If there is Non, A No alarms message will be displayed. From here you can delete alarms as you please. “5” Key is reserved for that purpose. User can remove any and all alarms, if the list is empty, it will say that.



Figure 2.5: Available Alarm list LCD Menus

### 2.3.2.4 TIMER

*TIMER* will give you a countdown timer. You can set how long of a timer you need by setting the time in *hh : mm : ss* format. After the timer reach 00:00:00, Star Wars Imperial march theme will be played.

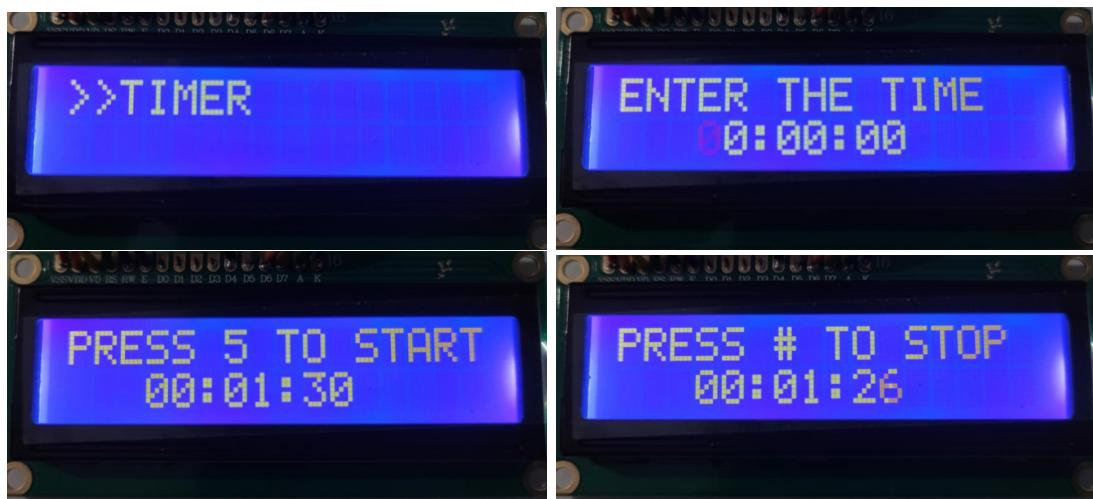


Figure 2.6: Timer LCD Menus

### 2.3.3 Alarm interrupting

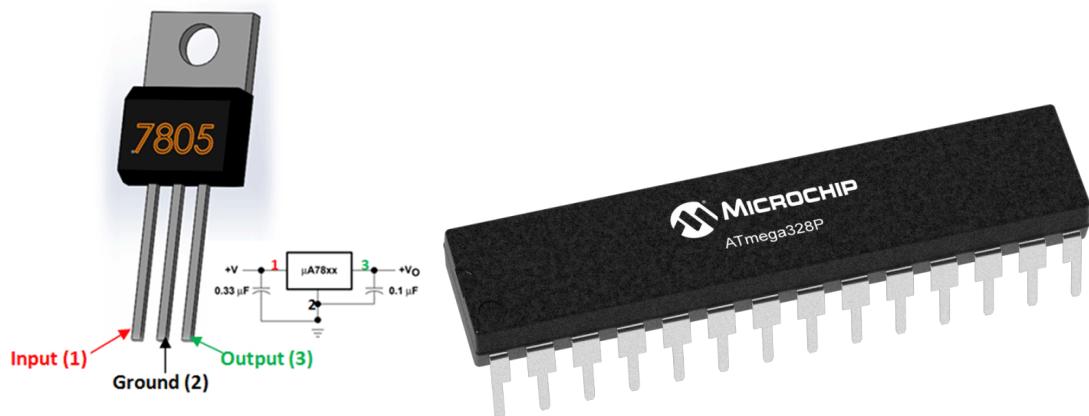
Since Alarms usually get abused when stopping alarm, we added a separate, durable and big button for this purpose. By using external interrupt pins, the “sp” variable will be changed to 0. Using this even when the microcontroller is busy with the for loop, checking this variable can be used to stop the buzzer music from playing.

### 2.3.4 Factory resetting

Using the inbuilt reset pin, which is also shared with the programmer, the factory resetting Will be done. The push button will reset everything except the RTC module because it has a separate memory inside. In the enclosure a pin cutout is provided for the user to push the hidden button with a pin. This will stop accidental button presses.

### 2.3.5 Microcontroller

ATmega328P is a simple, low-cost, low-powered 28 pin all-rounder. 16Mhz crystal oscillator is connected to provide the clock signals. We are utilising all 28 pins in our product and, for power input a power regulator is available in the PCB. A L7805CV power regulator ic accompanied by  $0.33\mu\text{F}$  and  $0.1\mu\text{F}$  capacitors provide stable 5V input.

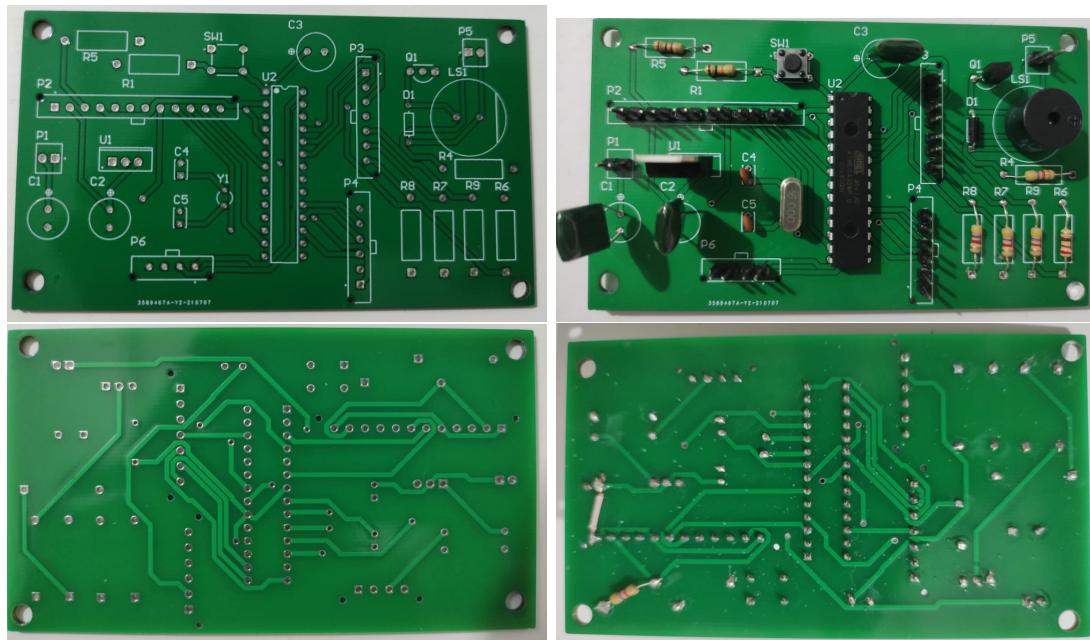


### 2.3.6 Programmer

Since our goal was to create a re-programmable digital alarm clock, there is an inbuilt programmer pin reserved header. Using either a USB to serial/TTL programmer or an Arduino device, user can flash firmware. The source code is available in this GitHub Repository.

### 3 Results

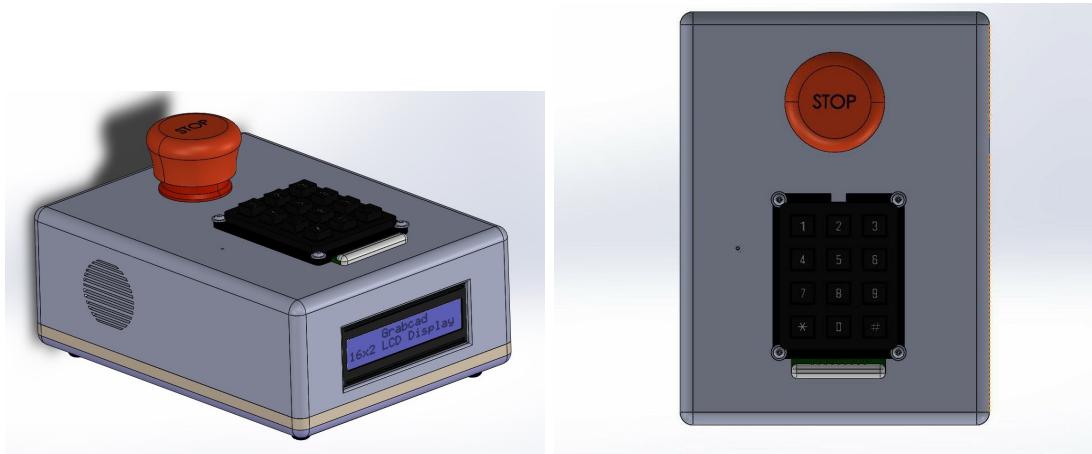
Our double layer PCB design was working as intended but one pin in LCD header was not grounded. The Buzzer had a good volume. The LCD had a contrast issue but was able to fix it by placing a resistor in the pin 3. The current usage was good. The keypad had good responses, the debouncing effect was in an acceptable range. After continuous run over a week, the Time in the RTC module was not affected.



**Dimensions**

Length	104 mm
Width	60 mm

As for our enclosure design, we were not able to 3D print it due to quarantine and other issues. After a 3D print, the keypad will be relabeled using durable sticker.



### Dimensions

Length	160 mm
Width	120.39 mm
Height(without button)	60 mm
Height(with button)	90.06 mm

## 4 Discussion

As beginners for the embedded designing, we had to learn all the software such as,

- Proteus
- Altium designer
- Solidworks

from the fundamentals.

Also, this was the first time we were introduced to AVR C++ and, we did not have much background in C language. We had to learn AVR C++ from the basics.

### 4.1 Problems and challenges we faced during this project

- Since the start of the semester, due to COVID 19 pandemic, our team was not able to collaborate in person.
- We had problems with acquiring few parts we needed for our product.
- The deliveries were delayed. So we had to wait for the ordered parts to come for a long time.
- Due to a manufacturing error, our PCB had an unrouted pin.
- Proteus simulations have frequency issues. So we had to alter codes many times for the buzzer. Due to this problem we are using  $1MHz$  internal frequency.
- LCD display had too high a contrast level, we had to connect a resistor in order to get proper readability.

### 4.2 Overall impression and possible improvements

Even though we were not able to 3D print our enclosure design and test it, our design has a good durability and the practicality. The basic box design is suitable for keeping internal components safe from hard impacts. The external button for stopping the alarm is really useful when the user is in a hurry and if the impact is high. The buzzer volume is load and the alarm tones are in a wide range of styles for user to choose.

Since there is no mechanism to change the LCD brightness level, this might become a problem for some scenarios. We will fix this issue and add a option to change screen timeout time.

Overall, the project was a tremendous success, both in its operations and in the lessons obtained from taking on such an involved project. It was a really good experience and gave us the push we needed to start our own projects on embedded systems. We had issues when printing the PCB and finally decided to order the PCB from JLC-PCB. Since from the beginning, when we were writing firmware for our micro products, the codes we wrote were easily combinable and was able to communicate with each parts easily.

There are few features we would like to add to this design. Adding a snooze option could improve user experience. This can be implemented by using the same interrupt button. We can add a option for this by providing a new settings menu too. Giving access to our 9V battery without removing the whole case is also a good feature to adept.

## 5 Acknowledgement

First of all, we would like to pay our gratitude toward our supervisor/instructor/evaluator Mr. Randima Senanayaka for his guidance and encouragement to overcome this challenge through self study and teamwork. Also, without ENTC family this project would not reach success. We would like to thank all the staff members who helped us throughout this project and semester 2. Our sincere gratitude for all the parties behind providing us with free Solidworks and Altium licenses.

The help, encouragement, tutorials we received from all the guest lecturers from *EN1070* module is invaluable and we would like to pay our gratitude.

# 6 Appendix

All the relevant files and progress data is available in this Github repository.

## 6.1 The PCB

Schematics, PCB and table of content. Used Altium Designer to design these models

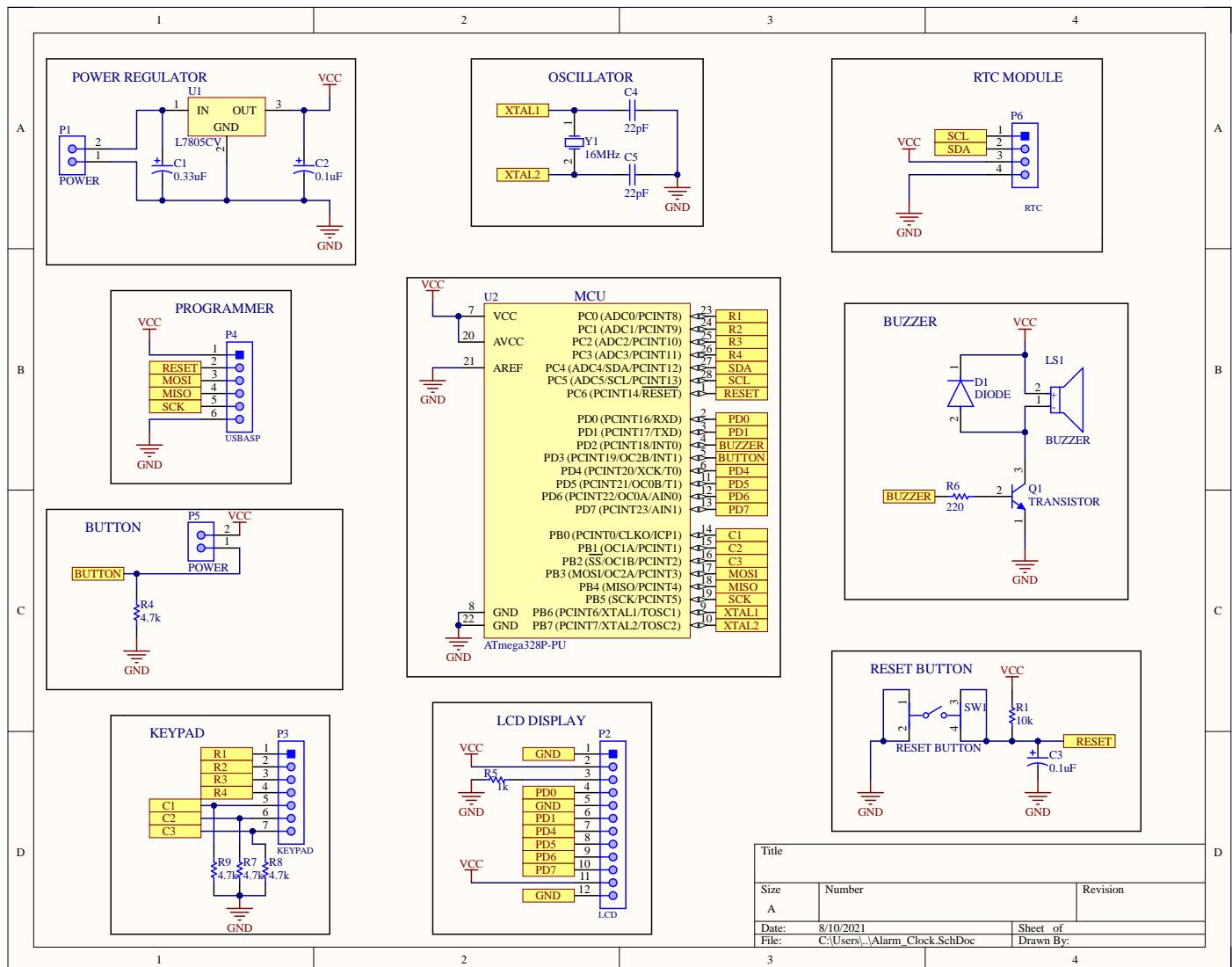


Figure 6.1: Schematics Design

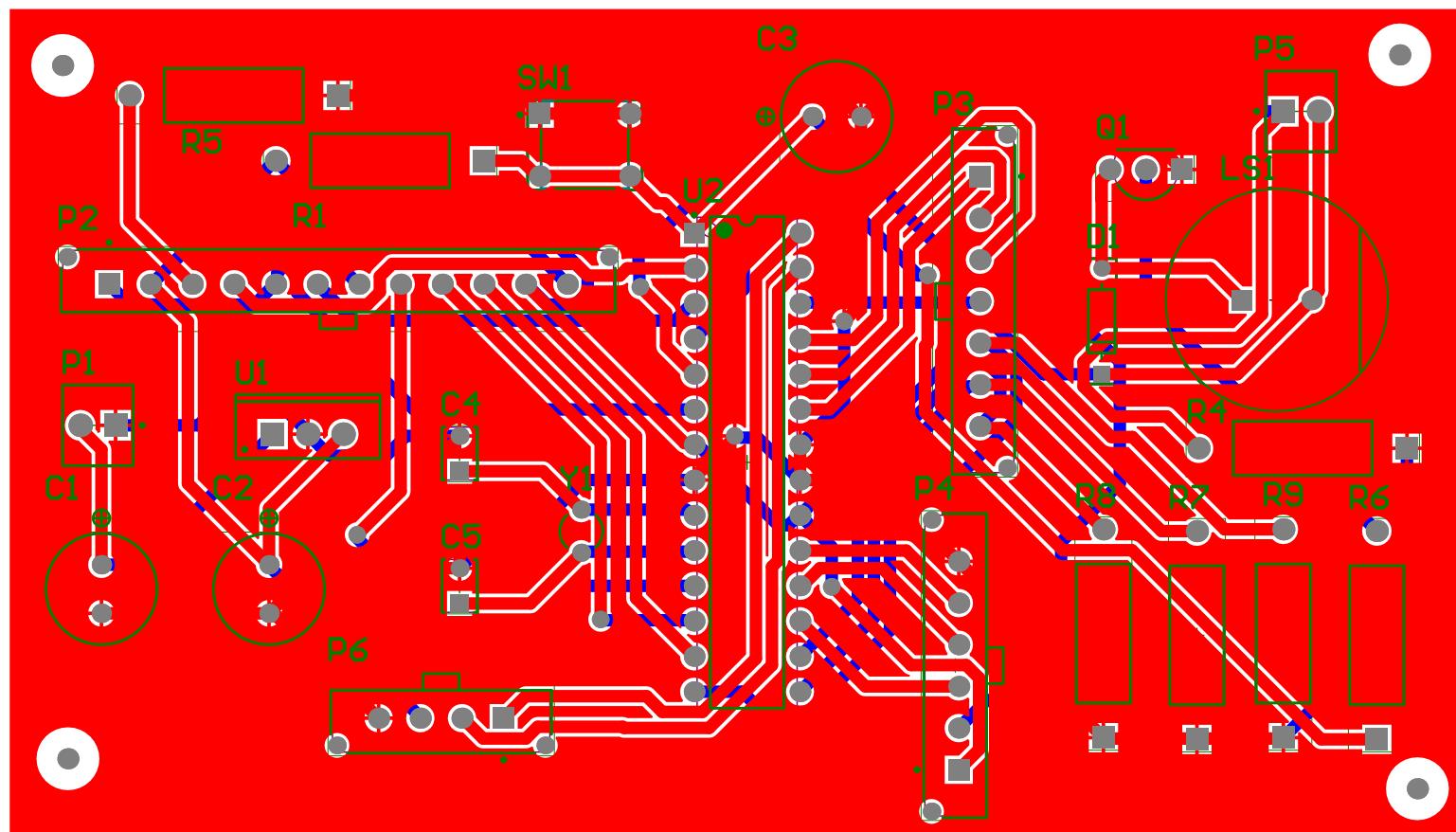


Figure 6.2: PCB Design

Bill of Materials for Project [Alarm_Clock.PrjPcb] (No PCB Document Selected)								
Source Data From:		<u>Alarm_Clock.PrjPcb</u>						
Project:		<u>Alarm_Clock.PrjPcb</u>						
Variant:		<u>None</u>						
Creation Date:	8/10/2021	1:01 PM						
Print Date:	10-Aug-21	1:01:17 PM						
Footprint	Comment	LibRef	Designator	Description		Quantity		
PNSC-FR-A-	0.33uF	CMP-0744-	C1	Aluminum Electrolytic Capacitor, 1000 uF		1		
PNSC-FR-A-	0.1uF	CMP-0744-	C2, C3	Aluminum Electrolytic Capacitor, 1000 uF		2		
KEMT-C315-	22pF	CMP-1664-	C4, C5	Capacitor, Ceramic, 0.01 uF, +/- 10%, 100		2		
FP-017AG-	DIODE	CMP-07163-	D1	DIODE GEN PURP 15V 150MA DO35		1		
FP-	BUZZER	CMP-08413-	LS1	AUDIO MAGNETIC INDICATOR 4-8V TH		1		
MOLX-22-27-	POWER	CMP-2000-	P1, P5	Male Header, Pitch 2.54 mm, 1 x 2 Position,		2		
66201211122	LCD	CMP-1502-	P2	THT Male Header WR-MPC3, Vertical,		1		
66200711122	KEYPAD	CMP-1502-	P3	THT Male Header WR-MPC3, Vertical,		1		
66200611122	USBASP	CMP-1502-	P4	THT Male Header WR-MPC3, Vertical,		1		
66200411122	RTC	CMP-1502-	P6	THT Male Header WR-MPC3, Vertical,		1		
FAIR-TO-92-	TRANSISTOR	CMP-2000-	Q1	NPN General-Purpose Amplifier, 40 V, -55		1		
VISH-A1-	10k	CMP-1240-	R1	Power Metal Film Leaded Resistor, 62		1		
VISH-A1-	4.7k	CMP-1240-	R4, R7, R8, R9	Power Metal Film Leaded Resistor, 62		4		
VISH-A1-	1k	CMP-1240-	R5	Power Metal Film Leaded Resistor, 62		1		
VISH-A1-	220	CMP-1240-	R6	Power Metal Film Leaded Resistor, 62		1		
FP-1825910-	RESET BUTTON	CMP-03407-	SW1	SWITCH TACTILE SPST-NO 0.05A 24V		1		
TO220	L7805CV	CMP-0244-	U1	Positive Voltage Regulator, 12V, 3-Pin TO-		1		
28P3	ATmega328P-	CMP-0095-	U2	8-bit AVR Microcontroller, 32KB Flash, 1KB		1		
R38	16MHz	XTAL	Y1	Crystal Oscillator		1		
						25		
Approved	Notes							

Figure 6.3: BOM

## 6.2 Proteus schematics

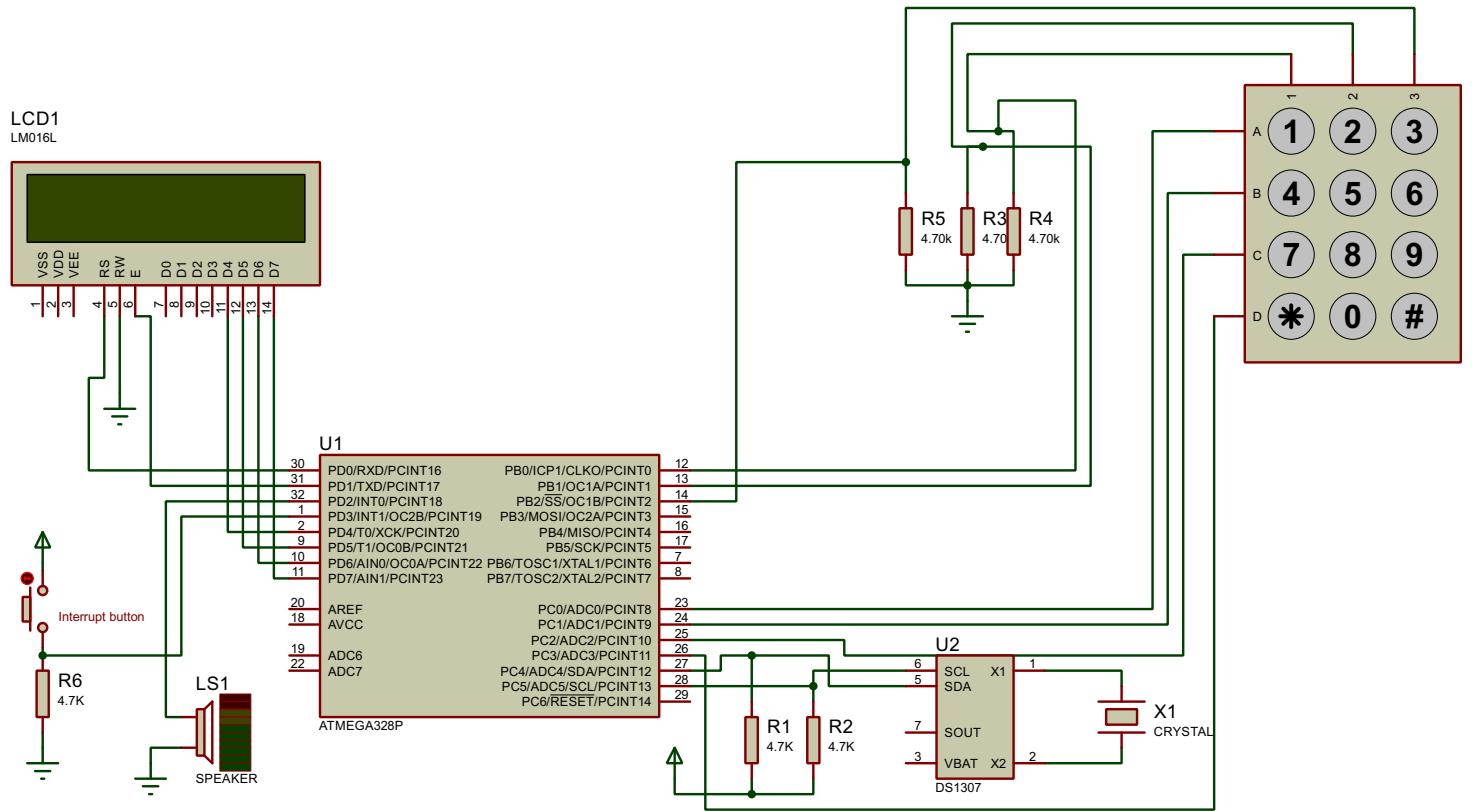


Figure 6.4: Proteus Simulation

### **6.3 Enclosure**

Please use Adobe reader DC to view this. Also for better view, open in full screen by right clicking.

Figure 6.5: 3D Enclosure View

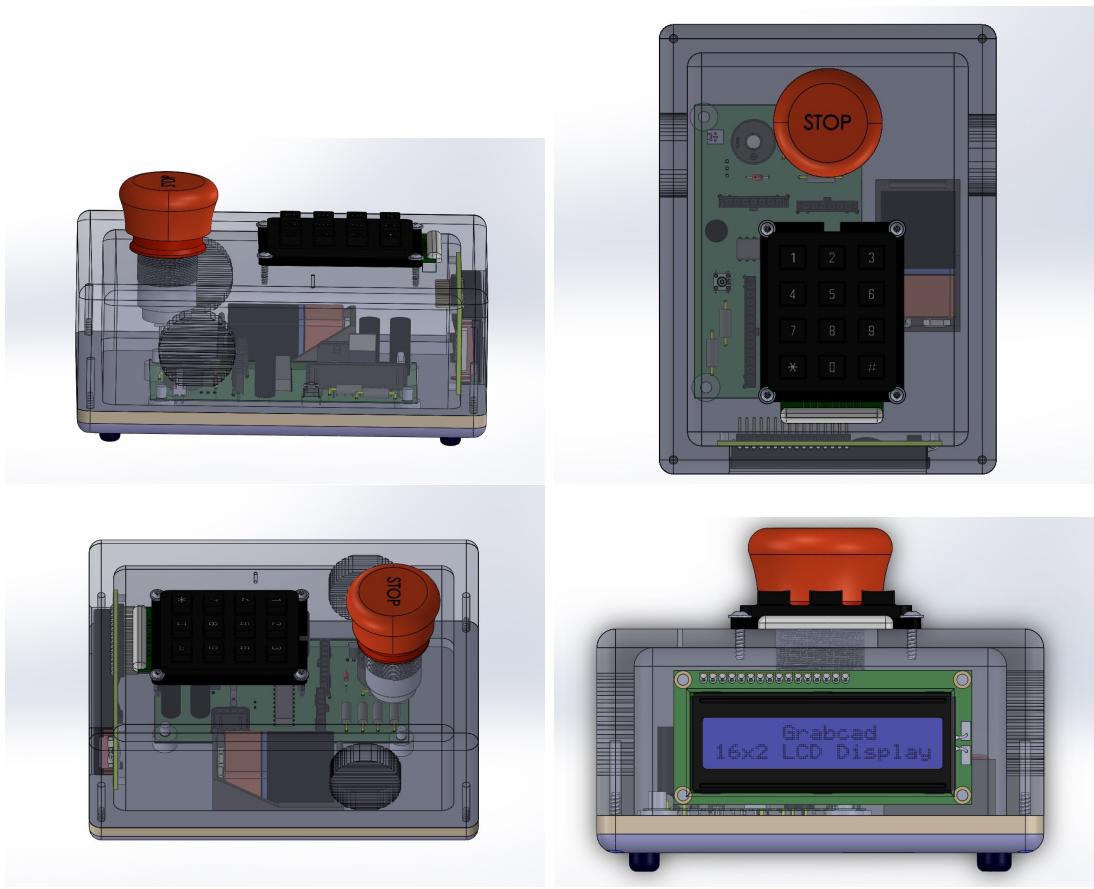


Figure 6.6: Transparent Enclosure views

## 6.4 AVR Codes

```
1 #include <avr/io.h>      /* Include AVR std. library file */
2 #include <avr/interrupt.h>
3 #include "Display.h"
4 #include "ds1307.h"
5 #include "Alarm.h"
6 #include "Keypad.h"
7 #include "util/delay.h"
8 #include "Buzzer.h"
9 #define BUTTON_DELAY 500
10
11 #ifndef F_CPU
12 #define F_CPU 1000000UL
13 #endif
14 int main()
15 {
16     init_buzzer(); // Initialize pins used for buzzer and Interrupt
17     button
18     LCD_Init(); /* Initialization of LCD*/
19     LCD_String(" WELCOME!!!");
20     _delay_ms(3000);
21     ds1307_init();
22     //int clocktime[6] = {21,05,4,12,8,35};
23     //setClockTime(clocktime);
24     //setAlarm(1209);setAlarm(1223);
25     int menu = 0;
26     char key;
27     while(1){
28         checkAlarm();
29         key = btnPress();
30         if (menu==0){
31             displayTime();
32         }
33         if (key=='*'){
34             _delay_ms(BUTTON_DELAY);
35             menu++;
36             LCD_Clear();
37         }
38         if (menu>0){
39             LCD_Menu(key,menu);
40         }
41         if (menu==3 || menu==-1){
42             menu = 0;
43             LCD_Clear();
44         }
45         if (key=='#'){
46             _delay_ms(BUTTON_DELAY);
47             menu--;
48             LCD_Clear();
49         }
50     }
51 }
```

Algorithm 6.1: Main.cpp

```

1  /*
2  *  Alarm.h
3  *
4  *  Created: 5/13/2021 8:39:09 PM
5  *  Author: Dakshina Tharidndu
6  */
7
8
9 #ifndef __ALARM_H__
10#define __ALARM_H__
11#include <avr/io.h>
12class Alarm
13{
14//variables
15public:
16protected:
17private:
18
19//functions
20public:
21    Alarm();
22    ~Alarm();
23
24protected:
25private:
26    Alarm( const Alarm &c );
27    Alarm& operator=( const Alarm &c );
28
29}; //Alarm
30extern bool isPress(uint8_t prt);
31extern int powerOf(int base, int power);
32//extern void getAlarmTime();
33extern void ringAlarm(int tone);
34extern void setAlarm(int alarm_time);
35extern void checkAlarm();
36extern void updateAlarmArray(int removeAlarm);
37extern void setClockTime(int clockTime[6]);
38extern int getDay(int y, int m, int d);
39extern void displayTime();
40extern void displayZero(uint8_t digit);
41extern void setTone(int t);
42extern void timer(char key,int state);
43extern void timerCountDown();
44extern void resetAlarmVariables();
45#endif //__ALARM_H__

```

Algorithm 6.2: Alarm.h

```

1
2
3 using namespace std;
4 #include "stdlib.h"
5 #include "Alarm.h"
6 #include "Display.h"
7 #include "util/delay.h"
8 #include <avr/io.h>
9 #include "ds1307.h"
10#include "Buzzer.h" //This is used to play the tone with the speaker
11#define BUTTON_DELAY 500
12#ifndef F_CPU
13#define F_CPU 1000000UL
14#endif
15

```

```

16 // default constructor
17 Alarm::Alarm()
18 {
19 } //Alarm
20
21 // default destructor
22 Alarm::~Alarm()
23 {
24 } //~Alarm
25 int numberOfAlarms = 0;
26 int alarmArray[10];
27 int toneArray[10];
28 int monthsDays[12]={31,28,31,30,31,30,31,31,30,31,30,31};
29 uint8_t year = 0;
30 uint8_t month = 0;
31 uint8_t day = 0;
32 uint8_t dayofweek = 0;
33 uint8_t hour = 0;
34 uint8_t minute = 0;
35 uint8_t second = 0;
36 int powerOf(int base, int power){
37     int result = 1;
38     while(power!=0){
39         result*=base;
40         power--;
41     }
42     return result;
43 }
44 bool isPress(uint8_t prt){
45     if (PIND == 1<< prt){
46         return true;
47     }
48     else{
49         return false;
50     }
51 }
52 void ringAlarm(int tone){
53     LCD_Clear();
54     LCD_String("RING!!");
55     play(tone);          //Trigger buzzer. Can be interrupted by the int1
                           //button.
56     //_delay_ms(2000);
57 }
58 void setAlarm(int alarm_time){
59     if (alarm_time<2400){
60         alarmArray [numberOfAlarms] = alarm_time;
61         numberOfAlarms++;
62     }
63 }
64 void setTone(int t){
65     toneArray [numberOfAlarms] = t;
66 }
67 void updateAlarmArray(int removeAlarm){
68     for (int i=0;i<numberOfAlarms;i++){
69         if (i<removeAlarm){
70             alarmArray [i] = alarmArray [i];
71             toneArray [i] = toneArray [i];
72         }
73         else{
74             alarmArray [i] = alarmArray [i+1];
75             toneArray [i] = toneArray [i+1];
76         }

```

```

77     }
78 }
79 void checkAlarm(){
80     if (numberOfAlarms != 0){
81         ds1307_getdate(&year, &month, &day, &dayofweek, &hour, &minute, &
82         second);
83         for (int i = 0; i <= numberOfAlarms;i++){
84             if (alarmArray[i]/100 == hour && alarmArray[i]%100 == minute){
85                 ringAlarm(toneArray[i]);
86                 updateAlarmArray(i);
87                 numberOfAlarms--;
88             }
89         }
90     }
91 }
92 void setClockTime(int clockTime[6]){
93     int y = clockTime[0];
94     int m = clockTime[1];
95     int d = clockTime[2];
96     int D = getDay(y,m,d);
97     int h = clockTime[3];
98     int M = clockTime[4];
99     int s = clockTime[5];
100    if ((0<y) & (y<99) & (0<m)&(m<13) & (0<d)&(d<32) & (0<=h)&(h<24) & (0<=
101        M)&(M<60) & (0<=s)&(s<60)){
102        ds1307_setdate(y, m, d, D, h, M, s);
103    }
104 int getDay(int y, int m, int d){
105     int days = 0;
106     for (int i = 1;i<=y;i++){
107         if (i%4 == 0){
108             days+= 366;
109         }
110         else{
111             days+= 365;
112         }
113     }
114     for (int j = 0;j<m-1;j++){
115         if ((y%4==0) & (j == 1)){
116             days+=monthsDays[j];
117             days+=1;
118         }
119         else{
120             days +=monthsDays[j];
121         }
122     }
123     return (days+d+5)%7;
124 }
125 void displayValue(int val){
126     char vals[10];
127     itoa(val,vals,10);
128     LCD_String(vals);
129 }
130 void displayZero(uint8_t digit){
131     if (digit<10){
132         LCD_String("0");
133     }
134 }
135 void displayTime(){

```

```

137     ds1307_getdate(&year, &month, &day, &dayofweek, &hour, &minute, &second
138     );
139 //    char secondS[10];char minuteS[10];char hourS[10];char yearS[10];char
140 //    monthS[10];char dayS[10];
141 //    itoa(second,secondS,10);itoa(minute,minuteS,10);itoa(hour,hours,10);
142 //    itoa(year,yearS,10);itoa(month,monthS,10);itoa(day,dayS,10);
143 LCD_Home(0);LCD_String(" 20");
144 displayZero(year);displayValue(year);LCD_String("-");
145 displayZero(month);displayValue(month);LCD_String("-");
146 displayZero(day);displayValue(day);
147 LCD_Home(1);LCD_String("      ");
148 displayZero(hour);
149 displayValue(hour);
150 LCD_String(":");
151 displayZero(minute);
152 displayValue(minute);
153 LCD_String(":");
154 displayZero(second);
155 displayValue(second);
156 //_delay_ms(1);
157 }
158
159 int alarm_var = 0;
160 char alarms[10];
161 void showAlarms(char key,int state){
162 LCD_Home(0);
163 if ((key=='8') & (state==2)){
164     alarm_var++;
165     //LCD_Clear();
166     _delay_ms(BUTTON_DELAY);
167 }
168 if ((key=='2') & (state ==2)){
169     alarm_var--;
170     //LCD_Clear();
171     _delay_ms(BUTTON_DELAY);
172 }
173 if ((key=='5') & (numberOfAlarms!=0)){
174     LCD_Clear();
175     displayTyping(alarmArray[alarm_var]/100,3);LCD_String(":");
176     displayTyping(alarmArray[alarm_var]%100,3);
177     LCD_String(" DELETED");LCD_Home(1);LCD_String(" SUCCESSFULLY");
178     _delay_ms(3000);
179     LCD_Clear();
180     updateAlarmArray(alarm_var);
181     numberOfAlarms--;
182     _delay_ms(BUTTON_DELAY);
183 }
184 if (alarm_var>=numberOfAlarms){
185     alarm_var=0;
186 }
187 if (alarm_var<0){
188     alarm_var=numberOfAlarms-1;
189 }
190 if (state==2){
191     LCD_Home(0);
192     if (numberOfAlarms==0){
193         LCD_String(" NO ALARMS");
194         LCD_Home(1);
195         LCD_String(" ARE SET");

```

```

195     }
196     else{
197         LCD_String("PRESS 5 TO DEL");
198         LCD_Home(1);
199         LCD_String(">>");
200         displayTyping(alarmArray[alarm_var]/100,3);LCD_String(":");
201         displayTyping(alarmArray[alarm_var]%100,3);
202     //     displayTyping(alarmArray[(alarm_var+1)%numberOfAlarms]/100,3);
203     //     LCD_String(":");
204     }
205 }
206 }
207 unsigned int timerTime[3];
208 int timerState = 0;
209 int temp1 = 0;
210 int data1 = 0;
211 int start=0;
212 void timer(char key, int state){
213
214     LCD_Home(1);
215     LCD_String("    ");
216     for (int i=0;i<3;i++){
217         if ((i!=3) & (i!=0)){
218             LCD_String(":");
219         }
220         if (i==timerState){
221             displayTyping(timerTime[i],temp1);
222         }
223         else{
224             displayTyping(timerTime[i],3);
225         }
226     }
227
228     if (timerState==3){
229         LCD_Home(0);
230
231         if (key=='5'){
232             start=1;
233         }
234         if (start==1){
235             LCD_String("PRESS # TO STOP ");
236             timerCountDown();
237         }
238         if (start==0){
239             LCD_String("PRESS 5 TO START");
240         }
241         if ((timerTime[2]==0) & (timerTime[1]==0) & (timerTime[0]==0) ){
242             start=0;
243             timerState=0;
244             LCD_Clear();
245             LCD_String("TIMES UP!!! ");
246             play(1);
247         }
248     }
249     else{
250         LCD_Home(0);
251         LCD_String("ENTER THE TIME");
252     }
253     if ((int(key)>47) & (int(key)<58) & (state ==2) & (timerState!=3)){
254         data1+=(int(key)-48);

```

```

255     _delay_ms(BUTTON_DELAY);
256     if (temp1 == 0){
257         data1*=10;
258         temp1 = 1;
259         timerTime[timerState] = data1;
260     }
261     else{
262         timerTime[timerState] = data1;
263         timerState++;
264         data1=0;
265         temp1 = 0;
266     }
267 }
268
269 }
270
271 int tempSec=60;
272 void timerCountDown(){
273     ds1307_getdate(&year, &month, &day, &dayofweek, &hour, &minute, &second
274     );
275
276     if (tempSec!=second){
277         timerTime [2] -=1;
278         tempSec=second;
279     }
280     if (timerTime [2]==-1){
281         if (timerTime [1]==0){
282             if (timerTime [0]!=0){
283                 timerTime [1]=59;timerTime [0]-=1;
284             }
285         }
286         else{
287             timerTime [2]=59;timerTime [1] -=1;
288         }
289     }
290     void resetAlarmVariables(){
291         tempSec=60;
292         timerTime [0]=0;timerTime [1]=0;timerTime [2]=0;
293         timerState = 0;
294         temp1 = 0;
295         data1 = 0;
296         start=0;
297     }

```

Algorithm 6.3: Alarm.cpp

```

1 /*
2  * Buzzer.h
3  *
4  * Created: 14/06/2021 16:34:31
5  * Author: Nuwan Udara
6 */
7
8
9 #ifndef BUZZER_H_
10 #define BUZZER_H_
11
12 /**
13 When the duration and note frequency are given this will generate 50%
14 duty
15 cycle square wave with given frequency for the given duration.
16 */

```

```

16 extern void Play_Note(float freq , float dur);
17 /**
18 Get the tempo value for the selected Alarm
19 Input will take the song number and return the assigned tempo
20 */
21 extern int tempochoose(int number);
22 /**
23 play function
24 This will take one input. The song number.
25 It will take tempo and the chords and duration data from melodies.h
26 accordingly
27 Using Play_Note it will generate sound
28 */
29 extern void play(int number);
30 /**
31 Declare input,output pins and interrupt pin methods
32 */
33 extern void init_buzzer();
34 /**
35 #endif /* BUZZER_H_ */

```

Algorithm 6.4: Buzzer.h

```

/*
 * Buzzer.cpp
 *
 * Created: 14/06/2021 16:32:48
 * Author: Nuwan Udara
 */
#define F_CPU 1000000UL //problem with Proteus simulation, so currently
                     //using 1MHz F_CPU //define F_CPU 16000000UL
#include <avr/io.h>
#define __DELAY_BACKWARD_COMPATIBLE__ // This is requaired to make
                                 _delay_ms() take vairables
#include <util/delay.h>           // for delay functions
#include <avr/interrupt.h>         // for clocks and interrupts
#include <avr/pgmspace.h>          //for PROGMEM
#include "pitches.h"              // Pitch data is stored in this
#include "melodies.h"              // Chords and duration data and tempo data
                                 // are stored in here
#include "Buzzer.h"               // The Include File for the main
//Buzzer declaration
#define buzzer_DDR DDRD
#define buzzer_PORT PORTD
#define buzzer_PIN PORTD2
// interrupt button declaration
#define interruptDDR DDRD
// volatile int sp = 1;      // This is a reference to stop the alarm with a
                           // button press
void init_buzzer(){
  // pins ports declaration
  buzzer_DDR |= _BV(buzzer_PIN) ;      // make buzzer pin output
  EICRA |= _BV(ISC11) | _BV(ISC10);    // set rising edge method
  EIMSK |= _BV(INT1);                 // enable pin PD1 External interrupt
}

```

```

34 sei(); // this Enables the Global Interrupt Enable (I-bit) in the Status Register (SREG)
35 }
36
37
38 void Play_Note(float freq, float dur){
39 // variables
40 long int i,cycles;
41 float half_period,period;
42
43 if (freq != 0){
44 period=(1/freq)*1000;
45 cycles=dur/period;
46 half_period = period/2;
47
48 for (i=0;i<cycles;i++)
49 {
50 //50% duty cycle
51 _delay_ms(half_period);
52 buzzer_PORT |= _BV(buzzer_PIN);
53 _delay_ms(half_period);
54 buzzer_PORT &= ~_BV(buzzer_PIN);
55 }
56 }
57 else {
58 _delay_ms(dur); // rest
59 }
60 return;
61 }
62
63 void play(int number){
64 sp=1;
65 int tempo=tempochoose(number); // choose the right tempo from the list
// using the function
66 int notes = sizeof(melody0[number])/sizeof(melody0[number][0])/2;
67 int wholenote = (60000 * 2.5) / tempo;
68 int divider = 0;
69 int noteDuration = 0;
70
71 for (int thisNote = 0; thisNote < notes *2 ; thisNote = thisNote + 2)
72 {
73 if (sp==1){
74 divider = pgm_read_word_near(melody0[number]+thisNote + 1);
75
76 if (divider > 0) { //positive divider ( positive duration)
// it is a regular note
77 noteDuration = (wholenote) / divider;}
78
79 else if (divider<0){
80 noteDuration=(wholenote)/(divider); // negative durations need to
be increased and made positive
81 noteDuration=noteDuration*1.5;
82 noteDuration-=noteDuration;
83 }
84 else if(divider==0){return;} // this will be used to stop playing
85 Play_Note((pgm_read_word_near(melody0[number]+thisNote)),
noteDuration*0.9 ); // play the selected pitch
86 _delay_ms(noteDuration*0.5);
87 }
88 else{
89 sp=0;
90 break;

```

```

91     }
92   }
93 }
94
95 /** External interrupt for INT0 (PD2) pin ISR function
96 We will use this to stop currently playing melody ***/
97
98 ISR (INT1_vect)           //External interrupt Stop button ISR
99 {
100   sp=0;
101   _delay_ms(100);
102 }
103
104 int tempochoose(int number){
105   if (number==0){return tempos[0];}
106   else if (number==1){return tempos[1];}
107   else if (number==2){return tempos[2];}
108   else{return tempos[3];}
109 }

```

Algorithm 6.5: Buzzer.ccp

```

1 #define NOTE_B0 31
2 #define NOTE_C1 33
3 #define NOTE_CS1 35
4 #define NOTE_D1 37
5 #define NOTE_DS1 39
6 #define NOTE_E1 41
7 #define NOTE_F1 44
8 #define NOTE_FS1 46
9 #define NOTE_G1 49
10 #define NOTE_GS1 52
11 #define NOTE_A1 55
12 #define NOTE_AS1 58
13 #define NOTE_B1 62
14 #define NOTE_C2 65
15 #define NOTE_CS2 69
16 #define NOTE_D2 73
17 #define NOTE_DS2 78
18 #define NOTE_E2 82
19 #define NOTE_F2 87
20 #define NOTE_FS2 93
21 #define NOTE_G2 98
22 #define NOTE_GS2 104
23 #define NOTE_A2 110
24 #define NOTE_AS2 117
25 #define NOTE_B2 123
26 #define NOTE_C3 131
27 #define NOTE_CS3 139
28 #define NOTE_D3 147
29 #define NOTE_DS3 156
30 #define NOTE_E3 165
31 #define NOTE_F3 175
32 #define NOTE_FS3 185
33 #define NOTE_G3 196
34 #define NOTE_GS3 208
35 #define NOTE_A3 220
36 #define NOTE_AS3 233
37 #define NOTE_B3 247
38 #define NOTE_C4 262
39 #define NOTE_CS4 277
40 #define NOTE_D4 294
41 #define NOTE_DS4 311

```

```

42 #define NOTE_E4 330
43 #define NOTE_F4 349
44 #define NOTE_FS4 370
45 #define NOTE_G4 392
46 #define NOTE_GS4 415
47 #define NOTE_A4 440
48 #define NOTE_AS4 466
49 #define NOTE_B4 494
50 #define NOTE_C5 523
51 #define NOTE_CS5 554
52 #define NOTE_D5 587
53 #define NOTE_DS5 622
54 #define NOTE_E5 659
55 #define NOTE_F5 698
56 #define NOTE_FS5 740
57 #define NOTE_G5 784
58 #define NOTE_GS5 831
59 #define NOTE_A5 880
60 #define NOTE_AS5 932
61 #define NOTE_B5 988
62 #define NOTE_C6 1047
63 #define NOTE_CS6 1109
64 #define NOTE_D6 1175
65 #define NOTE_DS6 1245
66 #define NOTE_E6 1319
67 #define NOTE_F6 1397
68 #define NOTE_FS6 1480
69 #define NOTE_G6 1568
70 #define NOTE_GS6 1661
71 #define NOTE_A6 1760
72 #define NOTE_AS6 1865
73 #define NOTE_B6 1976
74 #define NOTE_C7 2093
75 #define NOTE_CS7 2217
76 #define NOTE_D7 2349
77 #define NOTE_DS7 2489
78 #define NOTE_E7 2637
79 #define NOTE_F7 2794
80 #define NOTE_FS7 2960
81 #define NOTE_G7 3136
82 #define NOTE_GS7 3322
83 #define NOTE_A7 3520
84 #define NOTE_AS7 3729
85 #define NOTE_B7 3951
86 #define NOTE_C8 4186
87 #define NOTE_CS8 4435
88 #define NOTE_D8 4699
89 #define NOTE_DS8 4978
90 #define REST 0

```

Algorithm 6.6: pitches.h

```

1 #include <avr/pgmspace.h>
2
3 const int tempos[] PROGMEM={225,120,85,120,80};
4
5 const int melody0[5][400] PROGMEM= {{           //DOOM music
6     NOTE_E2, 8, NOTE_E2, 8, NOTE_E3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_D3, 8,
7     NOTE_E2, 8, NOTE_E2, 8, //1
8     NOTE_C3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_AS2, 8, NOTE_E2, 8, NOTE_E2,
9     8, NOTE_B2, 8, NOTE_C3, 8,
10    NOTE_E2, 8, NOTE_E2, 8, NOTE_E3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_D3, 8,
11    NOTE_E2, 8, NOTE_E2, 8,

```

```

9   NOTE_C3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_AS2, -2,
10  NOTE_E2, 8, NOTE_E2, 8, NOTE_E3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_D3, 8,
11    NOTE_E2, 8, NOTE_E2, 8, //5
12  NOTE_C3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_AS2, 8, NOTE_E2, 8, NOTE_E2,
13    8, NOTE_B2, 8, NOTE_C3, 8,
14  NOTE_E2, 8, NOTE_E2, 8, NOTE_E3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_D3, 8,
15    NOTE_E2, 8, NOTE_E2, 8,
16  NOTE_C3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_AS2, -2,
17  NOTE_E2, 8, NOTE_E2, 8, NOTE_E3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_D3, 8,
18    NOTE_E2, 8, NOTE_E2, 8, NOTE_E3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_D3, 8,
19    NOTE_E2, 8, NOTE_E2, 8,
20  NOTE_C3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_AS2, -2,
21  NOTE_E2, 8, NOTE_E2, 8, NOTE_E3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_D3, 8,
22    NOTE_E2, 8, NOTE_E2, 8, //13
23  NOTE_C3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_AS2, 8, NOTE_E2, 8, NOTE_E2,
24    8, NOTE_B2, 8, NOTE_C3, 8,
25  NOTE_E2, 8, NOTE_E2, 8, NOTE_E3, 8, NOTE_E2, 8, NOTE_E2, 8, NOTE_D3, 8,
26    NOTE_E2, 8, NOTE_E2, 8,
27 },
28 { // imperail march
29   NOTE_A4,4, NOTE_A4,4, NOTE_A4,4, NOTE_F4,-8, NOTE_C5,16,
30   NOTE_A4,4, NOTE_F4,-8, NOTE_C5,16, NOTE_A4,2,//4
31   NOTE_E5,4, NOTE_E5,4, NOTE_E5,4, NOTE_F5,-8, NOTE_C5,16,
32   NOTE_A4,4, NOTE_F4,-8, NOTE_C5,16, NOTE_A4,2,
33   NOTE_A5,4, NOTE_A4,-8, NOTE_A4,16, NOTE_A5,4, NOTE_GS5,-8, NOTE_G5,16,
34     //7
35   NOTE_DS5,16, NOTE_D5,16, NOTE_DS5,8, REST,8, NOTE_A4,8, NOTE_DS5,4,
36     NOTE_D5,-8, NOTE_CS5,16,
37   NOTE_C5,16, NOTE_B4,16, NOTE_C5,16, REST,8, NOTE_F4,8, NOTE_GS4,4,
38     NOTE_F4,-8, NOTE_A4,-16,//9
39   NOTE_C5,4, NOTE_A4,-8, NOTE_C5,16, NOTE_E5,2,
40   NOTE_A5,4, NOTE_A4,-8, NOTE_A4,16, NOTE_A5,4, NOTE_GS5,-8, NOTE_G5,16,
41     //7
42   NOTE_DS5,16, NOTE_D5,16, NOTE_DS5,8, REST,8, NOTE_A4,8, NOTE_DS5,4,
43     NOTE_D5,-8, NOTE_CS5,16,
44   NOTE_C5,16, NOTE_B4,16, NOTE_C5,16, REST,8, NOTE_F4,8, NOTE_GS4,4,
45     NOTE_F4,-8, NOTE_A4,-16,//9
46   NOTE_A4,4, NOTE_F4,-8, NOTE_C5,16, NOTE_A4,2,
47 },
48 { // Game of throne
49   NOTE_G4,8, NOTE_C4,8, NOTE_DS4,16, NOTE_F4,16, NOTE_G4,8, NOTE_C4,8,
50     NOTE_DS4,16, NOTE_F4,16, //1
51   NOTE_G4,8, NOTE_C4,8, NOTE_DS4,16, NOTE_F4,16, NOTE_G4,8, NOTE_C4,8,
52     NOTE_DS4,16, NOTE_F4,16,
53   NOTE_G4,8, NOTE_C4,8, NOTE_E4,16, NOTE_F4,16, NOTE_G4,8, NOTE_C4,8,
54     NOTE_E4,16, NOTE_F4,16,
```

```

51   NOTE_G4,8, NOTE_C4,8, NOTE_E4,16, NOTE_F4,16, NOTE_G4,8, NOTE_C4,8,
      NOTE_E4,16, NOTE_F4,16,
52   NOTE_G4,-4, NOTE_C4,-4, //5
53
54   NOTE_DS4,16, NOTE_F4,16, NOTE_G4,4, NOTE_C4,4, NOTE_DS4,16, NOTE_F4,16,
      //6
55   NOTE_D4,-1, //7 and 8
56   NOTE_F4,-4, NOTE_AS3,-4,
57   NOTE_DS4,16, NOTE_D4,16, NOTE_F4,4, NOTE_AS3,-4,
58   NOTE_DS4,16, NOTE_D4,16, NOTE_C4,-1, //11 and 12
59
60   //repeats from 5
61   NOTE_G4,-4, NOTE_C4,-4, //5
62
63   NOTE_DS4,16, NOTE_F4,16, NOTE_G4,4, NOTE_C4,4, NOTE_DS4,16, NOTE_F4,16,
      //6
64   NOTE_D4,-1, //7 and 8
65   NOTE_F4,-4, NOTE_AS3,-4,
66   NOTE_DS4,16, NOTE_D4,16, NOTE_F4,4, NOTE_AS3,-4,
67   NOTE_DS4,16, NOTE_D4,16, NOTE_C4,-1, //11 and 12
68   NOTE_G4,-4, NOTE_C4,-4,
69   NOTE_DS4,16, NOTE_F4,16, NOTE_G4,4, NOTE_C4,4, NOTE_DS4,16, NOTE_F4
      ,16,
70
71   NOTE_D4,-2, //15
72   NOTE_F4,-4, NOTE_AS3,-4,
73   NOTE_D4,-8, NOTE_DS4,-8, NOTE_D4,-8, NOTE_AS3,-8,
74   NOTE_C4,-1,
75   NOTE_C5,-2,
76   NOTE_AS4,-2,
77   NOTE_C4,-2,
78   NOTE_G4,-2,
79   NOTE_DS4,-2,
80   NOTE_DS4,-4, NOTE_F4,-4,
81   NOTE_G4,-1,
82
83   NOTE_C5,-2, //28
84   NOTE_AS4,-2,
85   NOTE_C4,-2,
86   NOTE_G4,-2,
87   NOTE_DS4,-2,
88   NOTE_DS4,-4, NOTE_D4,-4,
89   NOTE_C5,8, NOTE_G4,8, NOTE_GS4,16, NOTE_AS4,16, NOTE_C5,8, NOTE_G4,8,
      NOTE_GS4,16, NOTE_AS4,16,
90   NOTE_C5,8, NOTE_G4,8, NOTE_GS4,16, NOTE_AS4,16, NOTE_C5,8, NOTE_G4,8,
      NOTE_GS4,16, NOTE_AS4,16,
91   REST,4, NOTE_GS5,16, NOTE_AS5,16, NOTE_C6,8, NOTE_G5,8, NOTE_GS5,16,
      NOTE_AS5,16,
92   NOTE_C6,8, NOTE_G5,16, NOTE_GS5,16, NOTE_AS5,16, NOTE_C6,8, NOTE_G5,8,
      NOTE_GS5,16, NOTE_AS5,16,
93 },
94 { //pink panther
95   NOTE_E4,-4, REST,8, NOTE_FS4,8, NOTE_G4,-4, REST,8, NOTE_DS4,8,
96   NOTE_E4,-8, NOTE_FS4,8, NOTE_G4,-8, NOTE_C5,8, NOTE_B4,-8, NOTE_E4,8,
      NOTE_G4,-8, NOTE_B4,8,
97   NOTE_AS4,2, NOTE_A4,-16, NOTE_G4,-16, NOTE_E4,-16, NOTE_D4,-16,
98   NOTE_E4,2, REST,4, REST,8, NOTE_DS4,4,
99
100  NOTE_E4,-4, REST,8, NOTE_FS4,8, NOTE_G4,-4, REST,8, NOTE_DS4,8,
101  NOTE_E4,-8, NOTE_FS4,8, NOTE_G4,-8, NOTE_C5,8, NOTE_B4,-8, NOTE_G4,8,
      NOTE_B4,-8, NOTE_E5,8,
102  NOTE_DS5,1,

```

```

103     NOTE_D5, 2, REST, 4, REST, 8, NOTE_DS4, 8,
104     NOTE_E4, -4, REST, 8, NOTE_FS4, 8, NOTE_G4, -4, REST, 8, NOTE_DS4, 8,
105     NOTE_E4, -8, NOTE_FS4, 8, NOTE_G4, -8, NOTE_C5, 8, NOTE_B4, -8, NOTE_E4, 8,
106     NOTE_G4, -8, NOTE_B4, 8,
107
108     NOTE_AS4, 2, NOTE_A4, -16, NOTE_G4, -16, NOTE_E4, -16, NOTE_D4, -16,
109     NOTE_E4, -4, REST, 4,
110     REST, 4, NOTE_E5, -8, NOTE_D5, 8, NOTE_B4, -8, NOTE_A4, 8, NOTE_G4, -8,
111     NOTE_E4, -8,
112     NOTE_AS4, 16, NOTE_A4, -8, NOTE_AS4, 16, NOTE_A4, -8, NOTE_AS4, 16, NOTE_A4
113     , -8, NOTE_AS4, 16, NOTE_A4, -8,
114     NOTE_G4, -16, NOTE_E4, -16, NOTE_D4, -16, NOTE_E4, 16, NOTE_E4, 16, NOTE_E4
115     , 2,
116
117 },
118 { // GODFATHER theme
119     REST, 4, REST, 8, REST, 8, REST, 8, NOTE_E4, 8, NOTE_A4, 8, NOTE_C5,
120     8, //1
121     NOTE_B4, 8, NOTE_A4, 8, NOTE_C5, 8, NOTE_A4, 8, NOTE_B4, 8, NOTE_A4,
122     8, NOTE_F4, 8, NOTE_G4, 8,
123     NOTE_E4, 2, NOTE_E4, 8, NOTE_A4, 8, NOTE_C5, 8,
124     NOTE_B4, 8, NOTE_A4, 8, NOTE_C5, 8, NOTE_A4, 8, NOTE_C5, 8, NOTE_A4,
125     8, NOTE_E4, 8, NOTE_DS4, 8,
126
127     NOTE_D4, 2, NOTE_D4, 8, NOTE_F4, 8, NOTE_GS4, 8, //5
128     NOTE_B4, 2, NOTE_D4, 8, NOTE_F4, 8, NOTE_GS4, 8,
129     NOTE_A4, 2, NOTE_C4, 8, NOTE_C4, 8, NOTE_G4, 8,
130     NOTE_F4, 8, NOTE_E4, 8, NOTE_G4, 8, NOTE_F4, 8, NOTE_F4, 8, NOTE_E4,
131     8, NOTE_E4, 8, NOTE_GS4, 8,
132
133     NOTE_A4, 2, REST, 8, NOTE_A4, 8, NOTE_A4, 8, NOTE_GS4, 8, //9
134     NOTE_G4, 2, NOTE_B4, 8, NOTE_A4, 8, NOTE_F4, 8,
135     NOTE_E4, 2, NOTE_E4, 8, NOTE_G4, 8, NOTE_E4, 8,
136     NOTE_D4, 2, NOTE_D4, 8, NOTE_D4, 8, NOTE_F4, 8, NOTE_DS4, 8,
137
138     NOTE_E4, 2, REST, 8, NOTE_E4, 8, NOTE_A4, 8, NOTE_C5, 8, //13
139
140     //repeats from 2
141     NOTE_B4, 8, NOTE_A4, 8, NOTE_C5, 8, NOTE_A4, 8, NOTE_B4, 8, NOTE_A4,
142     8, NOTE_F4, 8, NOTE_G4, 8, //2
143     NOTE_E4, 2, NOTE_E4, 8, NOTE_A4, 8, NOTE_C5, 8,
144     NOTE_B4, 8, NOTE_A4, 8, NOTE_C5, 8, NOTE_A4, 8, NOTE_C5, 8, NOTE_A4,
145     8, NOTE_E4, 8, NOTE_DS4, 8,
146
147     NOTE_D4, 2, NOTE_D4, 8, NOTE_F4, 8, NOTE_GS4, 8, //5
148     NOTE_B4, 2, NOTE_D4, 8, NOTE_F4, 8, NOTE_GS4, 8,
149     NOTE_A4, 2, NOTE_C4, 8, NOTE_C4, 8, NOTE_G4, 8,
150     NOTE_F4, 8, NOTE_E4, 8, NOTE_G4, 8, NOTE_F4, 8, NOTE_F4, 8, NOTE_E4,
151     8, NOTE_E4, 8, NOTE_GS4, 8,
152
153     NOTE_A4, 2, REST, 8, NOTE_A4, 8, NOTE_A4, 8, NOTE_GS4, 8, //9
154     NOTE_G4, 2, NOTE_B4, 8, NOTE_A4, 8, NOTE_F4, 8,
155     NOTE_E4, 2, NOTE_E4, 8, NOTE_G4, 8, NOTE_E4, 8,
156     NOTE_D4, 2, NOTE_D4, 8, NOTE_D4, 8, NOTE_F4, 8, NOTE_DS4, 8,
157
158     NOTE_E4, 2 //13
159 }
160 };

```

Algorithm 6.7: melodies.h

```

1  /*
2  *  Display.h
3  *
4  *  Created: 6/11/2021 6:58:37 PM
5  *  Author: Dakshina Tharidndu
6  */
7
8
9 #ifndef __DISPLAY_H__
10#define __DISPLAY_H__
11
12
13 class Display
14 {
15 //variables
16 public:
17 protected:
18 private:
19
20 //functions
21 public:
22     Display();
23     ~Display();
24 protected:
25 private:
26     Display( const Display &c );
27     Display& operator=( const Display &c );
28
29 }; //Display
30 extern void LCD_Command( unsigned char cmnd );
31 extern void LCD_Char( unsigned char data );
32 extern void LCD_Init (void);
33 extern void LCD_String (char *str);
34 extern void LCD_String_xy (char row, char pos, char *str);
35 extern void LCD_Clear();
36 extern void LCD_Home(int r);
37 extern void LCD_Menu(char key,int state);
38 extern void LCD_SetAlarm(char key,int state);
39 extern void LCD_SetDate(int key, int state);
40 extern void LCD_Tone(char key,int state);
41 extern void resetVariables();
42 extern void showAlarms(char key,int state);
43 extern void displayTyping(int val,int pos);
44 extern void displayValue(int val);
45#endif //__DISPLAY_H__

```

Algorithm 6.8: Display.h

```

1
2 #ifndef F_CPU
3 #define F_CPU 1000000UL
4 #endif
5 #include <avr/io.h>      /* Include AVR std. library file */
6 #include <util/delay.h>
7 #define BUTTON_DELAY 500
8 #include "Display.h"
9 #include "Keypad.h"
10 #include "stdlib.h"
11 #include "Alarm.h"
12 #define LCD_Dir  DDRD      /* Define LCD data port direction */
13 #define LCD_Port PORTD      /* Define LCD data port */
14 #define RS PBO          /* Define Register Select pin */
15 #define EN PB1          /* Define Enable signal pin */

```

```

16 // default constructor
17 Display::Display()
18 {
19 } //Display
20
21 // default destructor
22 Display::~Display()
23 {
24 } //~Display
25 void LCD_Command( unsigned char cmnd )
26 {
27     LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0); /* sending upper nibble
28     */
29     LCD_Port &= ~ (1<<RS);      /* RS=0, command reg. */
30     LCD_Port |= (1<<EN);       /* Enable pulse */
31     _delay_us(1);
32     LCD_Port &= ~ (1<<EN);
33
34     _delay_us(200);
35
36     LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4); /* sending lower nibble */
37     LCD_Port |= (1<<EN);
38     _delay_us(1);
39     LCD_Port &= ~ (1<<EN);
40     _delay_ms(2);
41 }
42
43 void LCD_Char( unsigned char data )
44 {
45     LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0); /* sending upper nibble
46     */
47     LCD_Port |= (1<<RS);      /* RS=1, data reg. */
48     LCD_Port |= (1<<EN);
49     _delay_us(1);
50     LCD_Port &= ~ (1<<EN);
51
52     _delay_us(200);
53
54     LCD_Port = (LCD_Port & 0x0F) | (data << 4); /* sending lower nibble */
55     LCD_Port |= (1<<EN);
56     _delay_us(1);
57     LCD_Port &= ~ (1<<EN);
58     _delay_ms(2);
59 }
60 void LCD_Init (void)      /* LCD Initialize function */
61 {
62     LCD_Dir = 0xFF;        /* Make LCD port direction as o/p */
63     _delay_ms(20);         /* LCD Power ON delay always >15ms */
64
65     LCD_Command(0x02);    /* send for 4 bit initialization of LCD */
66     LCD_Command(0x28);    /* 2 line, 5*7 matrix in 4-bit mode */
67     LCD_Command(0x0c);    /* Display on cursor off*/
68     LCD_Command(0x06);    /* Increment cursor (shift cursor to
69     right)*/
70     LCD_Command(0x01);    /* Clear display screen*/
71     _delay_ms(2);
72 }
73 //char temp1='/' ;
74 void LCD_String (char *str) /* Send string to LCD function */

```

```

75 {
76     int i;
77     for(i=0;str[i]!=0;i++) /* Send each char of string till the NULL
    */
78     {
79         LCD_Char (str[i]);
80     }
81 }
82
83 void LCD_String_xy (char row, char pos, char *str) /* Send string to LCD
   with xy position */
84 {
85     if (row == 0 && pos<16)
86         LCD_Command((pos & 0x0F)|0x80); /* Command of first row and required
   position<16 */
87     else if (row == 1 && pos<16)
88         LCD_Command((pos & 0x0F)|0xC0); /* Command of first row and required
   position<16 */
89     LCD_String(str); /* Call LCD string function */
90 }
91
92 void LCD_Clear()
93 {
94     LCD_Command (0x01); /* Clear display */
95     _delay_ms(2);
96     LCD_Command (0x80); /* Cursor at home position */
97 }
98 void LCD_Home(int r){
99     if(r==0){
100         LCD_Command (0x80);
101     }
102     else{
103         LCD_Command (0xC0);
104     }
105 }
106 int blink_var=0;
107 void LCD_Blink(char ch[10]){
108     if (blink_var==0){
109         LCD_String(ch);
110         blink_var=1;
111         _delay_ms(100);
112     }
113     else{
114         LCD_String(" ");
115         blink_var=0;
116         _delay_ms(100);
117     }
118 }
119 void displayTyping(int val,int pos){
120     char val0S[2];char val1S[2];
121     itoa(val/10,val0S,10);itoa(val%10,val1S,10);
122     if (pos==0){
123         LCD_Blink(val0S);
124         LCD_String(val1S);
125     }
126     else if (pos==1){
127         LCD_String(val0S);
128         LCD_Blink(val1S);
129     }
130     else{
131         LCD_String(val0S);LCD_String(val1S);
132     }

```

```

133 }
134 int clkTime = 0;
135 int num = 0;
136 signed int p = 3;
137 void LCD_SetAlarm(char key, int state){
138     if (p != -1){
139         LCD_Home(0);
140         LCD_String("      ");
141         displayTyping(clkTime/100,3-p);
142         LCD_String(":");
143         displayTyping(clkTime%100,1-p);
144     }
145 //LCD_String("0000");
146 if ((int(key)>47) & (int(key)<58) & (state ==2) & (p != -1)){
147     clkTime += (int(key)-48)*powerOf(10,p);
148     p--;
149     _delay_ms(BUTTON_DELAY);
150     if (p == -1){
151         LCD_Home(0);
152         LCD_String("      ");
153         displayTyping(clkTime/100,3);LCD_String(":");displayTyping(clkTime
154 %100,3);
155         _delay_ms(1000);
156         LCD_Home(0);
157         LCD_String("SELECT THE TONE");
158         _delay_ms(2000);
159         LCD_Clear();
160     }
161     if (p == -1){
162         LCD_Tone(key,state);
163     }
164     if (state==3){
165         setAlarm(clkTime);
166         LCD_Home(0);
167         LCD_String("ALARM IS SET TO");
168         LCD_Home(1);
169         displayTyping(clkTime/100,3);LCD_String(":");displayTyping(clkTime
170 %100,3);
171         clkTime=0;num=0;p=3;
172         _delay_ms(2000);
173         LCD_Clear();
174     }
175     int clockTime[6];
176     int ClockState = 0;
177     int temp = 0;
178     int data = 0;
179     char dataS[10];
180     void LCD_SetDate(int key, int state){
181         LCD_Home(0);
182         LCD_String("    20");
183         for (int i=0;i<6;i++){
184             if (i==3){
185                 LCD_Home(1);
186                 LCD_String("      ");
187             }
188             if ((i!=3) & (i!=0)){
189                 LCD_String(":");
190             }
191             if (i==ClockState){
192                 displayTyping(clockTime[i],temp);

```

```

193     }
194     else{
195         displayTyping(clockTime[i],3);
196     }
197 }
198 if ((int(key)>47) & (int(key)<58) & (state ==2)){
199     data+=(int(key)-48);
200     _delay_ms(BUTTON_DELAY);
201     if (temp == 0){
202         data*=10;
203         temp = 1;
204         clockTime[ClockState] = data;
205     }
206     else{
207         clockTime[ClockState] = data;
208         ClockState++;
209         data=0;
210         temp = 0;
211     }
212 }
213 if ((ClockState == 6) & (state==3)){
214     setClockTime(clockTime);
215     ClockState = 0;
216     LCD_Home(0);LCD_Clear();
217     LCD_String("      DONE!");
218     _delay_ms(2000);
219     LCD_Clear();
220 }
221 }
222 char tone_List[5][10] = {"DOOM      ", "STAR WARS", "GOT      ", "PANTHER  ",
223 "GODFATHER"};
224 int tone_Var = 0;
225 void LCD_Tone(char key,int state){
226     LCD_Home(0);
227     if ((key=='8') & (state==2)){
228         tone_Var++;
229         //LCD_Clear();
230         _delay_ms(BUTTON_DELAY);
231     }
232     if ((key=='2') & (state ==2)){
233         tone_Var--;
234         //LCD_Clear();
235         _delay_ms(BUTTON_DELAY);
236     }
237     if (tone_Var>4){
238         tone_Var=0;
239     }
240     if (tone_Var<0){
241         tone_Var=4;
242     }
243     if (state==2){
244         LCD_Home(0);LCD_String(">>");
245         LCD_String(tone_List[tone_Var%5]);
246         // LCD_Home(1);
247         // LCD_String(menu_List[(menu_Var+1)%3]);
248     }
249     if (state==3){
250         setTone(tone_Var);
251     }
252 }
253 char menu_List[4][11] = {"SET ALARM ", "SET TIME   ", "SEE ALARMS", "TIMER

```

```

        "};

254 int menu_Var = 0;
255 void LCD_Menu(char key, int state){
256     LCD_Home(0);
257     if ((key=='8') & (state==1)){
258         menu_Var++;
259         //LCD_Clear();
260         _delay_ms(BUTTON_DELAY);
261     }
262     if ((key=='2') & (state ==1)){
263         menu_Var--;
264         //LCD_Clear();
265         _delay_ms(BUTTON_DELAY);
266     }
267     if (menu_Var>3){
268         menu_Var=0;
269     }
270     if (menu_Var<0){
271         menu_Var=3;
272     }
273     if (state==1){
274         LCD_Home(0);LCD_String(">>");
275         LCD_String(menu_List[menu_Var%4]);
276         resetVariables();
277         // LCD_Home(1);
278         // LCD_String(menu_List[(menu_Var+1)%3]);
279     }
280     if ((state >1) & (menu_Var==0)){
281         LCD_SetAlarm(key,state);
282     }
283     if ((state >1) & (menu_Var==1)){
284         LCD_SetDate(key,state);
285     }
286     if ((state >1) & (menu_Var==2)){
287         showAlarms(key,state);
288     }
289     if ((state >1) & (menu_Var==3)){
290         timer(key,state);
291     }
292 }
293 void resetVariables(){
294     resetAlarmVariables();
295     tone_Var=0;
296     clockTime[0]=0;clockTime[1]=0;clockTime[2]=0;clockTime[3]=0;clockTime
297         [4]=0;clockTime[5]=0;
298     ClockState = 0;
299     temp = 0;
300     data = 0;
301     clkTime = 0;
302     num = 0;
303     p = 3;
304 }
```

Algorithm 6.9: Display.cpp

```

1 /*
2 * Keypad.h
3 *
4 * Created: 6/13/2021 11:40:03 AM
5 * Author: Dakshina Tharidndu
6 */
7
8
```

```

9 #ifndef __KEYPAD_H__
10 #define __KEYPAD_H__
11
12
13 class Keypad
14 {
15 //variables
16 public:
17 protected:
18 private:
19
20 //functions
21 public:
22     Keypad();
23     ~Keypad();
24 protected:
25 private:
26     Keypad( const Keypad &c );
27     Keypad& operator=( const Keypad &c );
28
29 }; //Keypad
30 extern char btnPress();
31 #endif //__KEYPAD_H__

```

Algorithm 6.10: Keypad.h

```

1
2 #include "Keypad.h"
3 #include <avr/io.h>
4 #include <util/delay.h>
5 #define keyDir DDRC
6 #define keyPort PORTC
7 #define keyPin PINB
8 char keys[4][3] = {{'1','2','3'},{'4','5','6'},{'7','8','9'}, {'*','0','#'}};
9 uint8_t row[4] = {PORTC0,PORTC1,PORTC2,PORTC3};
10 uint8_t column[3] = {PINB0,PINB1,PINB2};
11 // default constructor
12 Keypad::Keypad()
13 {
14 } //Keypad
15
16 // default destructor
17 Keypad::~Keypad()
18 {
19 } //~Keypad
20
21
22 char btnPress(){
23     keyDir = 0x0F;
24     for (int i=0;i<4;i++){
25         keyPort = 1<<row[i];
26         for (int j = 0;j<3;j++){
27             if (keyPin==(1<<column[j])){
28                 //_delay_ms(100);
29                 return keys[i][j];
30             }
31         }
32     }
33     return '&';
34 }

```

Algorithm 6.11: Keypad.cpp

```

1 #include <avr/io.h>
2 extern uint8_t ds1307_dec2bcd(uint8_t val);
3 extern uint8_t ds1307_bcd2dec(uint8_t val);
4 extern void ds1307_init();
5 extern void ds1307_setdate(uint8_t year, uint8_t month, uint8_t day,
6     uint8_t dayofweek, uint8_t hour, uint8_t minute, uint8_t second);
7 extern void ds1307_getdate(uint8_t *year, uint8_t *month, uint8_t *day,
8     uint8_t *dayofweek, uint8_t *hour, uint8_t *minute, uint8_t *second);

```

Algorithm 6.12: ds1307.h

```

1 #ifndef DS1307_H
2 #define DS1307_H
3
4 #include <avr/io.h>
5 #include "ds1307.h"
6 #include "i2cmaster.h"
7 #include "util/delay.h"
8 #define DS1307_ADDRESS 0xDO
9
10 #ifndef F_CPU
11 #define F_CPU 1000000UL
12 #endif
13
14
15 uint8_t ds1307_dec2bcd(uint8_t val) {
16     return val + 6 * (val / 10);
17 }
18 uint8_t ds1307_bcd2dec(uint8_t val) {
19     return val - 6 * (val >> 4);
20 }
21 void ds1307_init() {
22     i2c_init();
23     _delay_us(10);
24 }
25 void ds1307_setdate(uint8_t year, uint8_t month, uint8_t day, uint8_t
26     dayofweek, uint8_t hour, uint8_t minute, uint8_t second) {
27
28     i2c_start_wait(DS1307_ADDRESS | I2C_WRITE);
29     i2c_write(0x00);
30     i2c_write(ds1307_dec2bcd(second));
31     i2c_write(ds1307_dec2bcd(minute));
32     i2c_write(ds1307_dec2bcd(hour));
33     i2c_write(ds1307_dec2bcd(dayofweek));
34     i2c_write(ds1307_dec2bcd(day));
35     i2c_write(ds1307_dec2bcd(month));
36     i2c_write(ds1307_dec2bcd(year));
37     i2c_write(0x00);
38     i2c_stop();
39 }
40 void ds1307_getdate(uint8_t *year, uint8_t *month, uint8_t *day, uint8_t *
41     dayofweek, uint8_t *hour, uint8_t *minute, uint8_t *second) {
42     i2c_start_wait(DS1307_ADDRESS | I2C_WRITE);
43     i2c_write(0x00);
44     i2c_stop();
45
46     i2c_rep_start(DS1307_ADDRESS | I2C_READ);
47     *second = ds1307_bcd2dec(i2c_readAck() & 0x7F);
48     *minute = ds1307_bcd2dec(i2c_readAck());
49     *hour = ds1307_bcd2dec(i2c_readAck());
50     *dayofweek = ds1307_bcd2dec(i2c_readAck());
51     *day = ds1307_bcd2dec(i2c_readAck());

```

```

51     *month = ds1307_bcd2dec(i2c_readAck());
52     *year = ds1307_bcd2dec(i2c_readNak());
53     i2c_stop();
54 }
55 #endif

```

Algorithm 6.13: ds1307.cpp

```

1 #ifndef _I2CMASTER_H
2 #define _I2CMASTER_H     1
3 /*
4  ****
5 * Title:      C include file for the I2C master interface
6 *             (i2cmaster.S or twimaster.c)
7 * Author:     Peter Fleury <pfleury@gmx.ch> http://jump.to/fleury
8 * File:       $Id: i2cmaster.h,v 1.10 2005/03/06 22:39:57 Peter Exp $
9 * Software:   AVR-GCC 3.4.3 / avr-libc 1.2.3
10 * Target:    any AVR device
11 * Usage:     see Doxygen manual
12 ****
13 */
14 #ifdef DOXYGEN
15 /**
16  @defgroup pfleury_ic2master I2C Master library
17  @code #include <i2cmaster.h> @endcode
18
19  @brief I2C (TWI) Master Software Library
20
21  Basic routines for communicating with I2C slave devices. This single
22  master
23  implementation is limited to one bus master on the I2C bus.
24
25  This I2c library is implemented as a compact assembler software
26  implementation of the I2C protocol
27  which runs on any AVR (i2cmaster.S) and as a TWI hardware interface for
28  all AVR with built-in TWI hardware (twimaster.c).
29  Since the API for these two implementations is exactly the same, an
30  application can be linked either against the
31  software I2C implementation or the hardware I2C implementation.
32
33  Use 4.7k pull-up resistor on the SDA and SCL pin.
34
35  Adapt the SCL and SDA port and pin definitions and eventually the delay
36  routine in the module
37  i2cmaster.S to your target when using the software I2C implementation !
38
39  Adjust the CPU clock frequency F_CPU in twimaster.c or in the Makfile
40  when using the TWI hardware implementaion.
41
42  @note
43  The module i2cmaster.S is based on the Atmel Application Note AVR300,
44  corrected and adapted
45  to GNU assembler and AVR-GCC C call interface.
46  Replaced the incorrect quarter period delays found in AVR300 with
47  half period delays.
48
49  @author Peter Fleury pfleury@gmx.ch http://jump.to/fleury
50
51  @par API Usage Example
52  The following code shows typical usage of this library, see example
53  test_i2cmaster.c

```

```

45
46 @code
47
48 #include <i2cmaster.h>
49
50
51 #define Dev24C02 0xA2      // device address of EEPROM 24C02, see
52   datasheet
53
54 int main(void)
55 {
56     unsigned char ret;
57
58     i2c_init();           // initialize I2C library
59
60     // write 0x75 to EEPROM address 5 (Byte Write)
61     i2c_start_wait(Dev24C02+I2C_WRITE);    // set device address and
62     write mode
63     i2c_write(0x05);                  // write address = 5
64     i2c_write(0x75);                  // write value 0x75 to
65     EEPROM
66     i2c_stop();                     // set stop conditon =
67     release bus
68
69
70     // read previously written value back from EEPROM address 5
71     i2c_start_wait(Dev24C02+I2C_WRITE);    // set device address and
72     write mode
73
74     i2c_write(0x05);                  // write address = 5
75     i2c_rep_start(Dev24C02+I2C_READ);    // set device address and
76     read mode
77
78     ret = i2c_readNak();             // read one byte from EEPROM
79     i2c_stop();
80
81     for(;;);
82 }
83 @endcode
84 */
85 #endif /* DOXYGEN */
86
87 /**
88 #if (_GNUC_ * 100 + _GNUC_MINOR_) < 304
89 #error "This library requires AVR-GCC 3.4 or later, update to newer AVR-
90   GCC compiler !"
91 #endif
92
93 /**
94 #include <avr/io.h>
95
96 /**
97 ** defines the data direction (reading from I2C device) in i2c_start(),
98   i2c_rep_start() */
99 #define I2C_READ    1
100
101 /**
102 ** defines the data direction (writing to I2C device) in i2c_start(),
103   i2c_rep_start() */
104 #define I2C_WRITE   0
105
106 /**
107 /**

```

```

98  *@brief initialize the I2C master interface. Need to be called only once
99  *@param void
100  *@return none
101  */
102  extern void i2c_init(void);
103
104
105 /**
106  *@brief Terminates the data transfer and releases the I2C bus
107  *@param void
108  *@return none
109  */
110  extern void i2c_stop(void);
111
112
113 /**
114  *@brief Issues a start condition and sends address and transfer direction
115
116  *@param     addr address and transfer direction of I2C device
117  *@retval    0   device accessible
118  *@retval    1   failed to access device
119  */
120  extern unsigned char i2c_start(unsigned char addr);
121
122
123 /**
124  *@brief Issues a repeated start condition and sends address and transfer
125  *       direction
126
127  *@param     addr address and transfer direction of I2C device
128  *@retval    0   device accessible
129  *@retval    1   failed to access device
130  */
131  extern unsigned char i2c_rep_start(unsigned char addr);
132
133 /**
134  *@brief Issues a start condition and sends address and transfer direction
135
136  If device is busy, use ack polling to wait until device ready
137  *@param     addr address and transfer direction of I2C device
138  *@return    none
139  */
140  extern void i2c_start_wait(unsigned char addr);
141
142
143 /**
144  *@brief Send one byte to I2C device
145  *@param     data byte to be transferred
146  *@retval    0   write successful
147  *@retval    1   write failed
148  */
149  extern unsigned char i2c_write(unsigned char data);
150
151
152 /**
153  *@brief      read one byte from the I2C device, request more data from
154  *            device
155  *@return    byte read from I2C device
156  */
157  extern unsigned char i2c_readAck(void);

```

```

158 /**
159  * @brief      read one byte from the I2C device, read is followed by a stop
160  *             condition
161  * @return     byte read from I2C device
162 */
163 extern unsigned char i2c_readNak(void);
164 /**
165  * @brief      read one byte from the I2C device
166
167 Implemented as a macro, which calls either i2c_readAck or i2c_readNak
168
169 @param      ack 1 send ack, request more data from device<br>
170             0 send nak, read is followed by a stop condition
171 @return     byte read from I2C device
172 */
173 extern unsigned char i2c_read(unsigned char ack);
174 #define i2c_read(ack) (ack) ? i2c_readAck() : i2c_readNak();
175
176
177 /**@}*/
178 #endif

```

Algorithm 6.14: i2cmaster.h

```

1 /*
2 ****
3 * Title:      I2C master library using hardware TWI interface
4 * Author:    Peter Fleury <pfleury@gmx.ch> http://jump.to/fleury
5 * File:      $Id: twimaster.c,v 1.3 2005/07/02 11:14:21 Peter Exp $
6 * Software:  AVR-GCC 3.4.3 / avr-libc 1.2.3
7 * Target:    any AVR device with hardware TWI
8 * Usage:     API compatible with I2C Software Library i2cmaster.h
9 ****
10 */
11 #include <inttypes.h>
12 #include <compat/twi.h>
13
14
15 /* define CPU frequency in Mhz here if not defined in Makefile */
16 #ifndef F_CPU
17 #define F_CPU 1000000UL
18 #endif
19
20 /* I2C clock in Hz */
21 #define SCL_CLOCK 100000L
22
23
24 /*
25 ****
26
27 Initialization of the I2C bus interface. Need to be called only once
28 ****
29 */
30 void i2c_init(void)
31 {
32     /* initialize TWI clock: 100 kHz clock, TWPS = 0 => prescaler = 1 */
33     TWSR = 0;                                /* no prescaler */

```

```

32     TWBR = ((F_CPU/SCL_CLOCK)-16)/2; /* must be > 10 for stable operation
33     */
34 }/* i2c_init */
35
36
37 /*
38 * Issues a start condition and sends address and transfer direction.
39 * return 0 = device accessible, 1= failed to access device
40 ****
41 */
42 unsigned char i2c_start(unsigned char address)
43 {
44     uint8_t    twst;
45
46     // send START condition
47     TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
48
49     // wait until transmission completed
50     while(!(TWCR & (1<<TWINT)));
51
52     // check value of TWI Status Register. Mask prescaler bits.
53     twst = TW_STATUS & 0xF8;
54     if ( (twst != TW_START) && (twst != TW_REP_START)) return 1;
55
56     // send device address
57     TWDR = address;
58     TWCR = (1<<TWINT) | (1<<TWEN);
59
60     // wait until transmission completed and ACK/NACK has been received
61     while(!(TWCR & (1<<TWINT)));
62
63     // check value of TWI Status Register. Mask prescaler bits.
64     twst = TW_STATUS & 0xF8;
65     if ( (twst != TW_MT_SLA_ACK) && (twst != TW_MR_SLA_ACK) ) return 1;
66
67     return 0;
68 }/* i2c_start */
69
70 /*
71 * Issues a start condition and sends address and transfer direction.
72 * If device is busy, use ack polling to wait until device is ready
73
74 * Input: address and transfer direction of I2C device
75 ****
76 */
77 void i2c_start_wait(unsigned char address)
78 {
79     uint8_t    twst;
80
81     while ( 1 )
82     {
83         // send START condition
84         TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);

```

```

87     // wait until transmission completed
88     while(!(TWCR & (1<<TWINT)));
89
90     // check value of TWI Status Register. Mask prescaler bits.
91     twst = TW_STATUS & 0xF8;
92     if ( (twst != TW_START) && (twst != TW_REP_START)) continue;
93
94     // send device address
95     TWDR = address;
96     TWCR = (1<<TWINT) | (1<<TWEN);
97
98     // wait until transmission completed
99     while(!(TWCR & (1<<TWINT)));
100
101    // check value of TWI Status Register. Mask prescaler bits.
102    twst = TW_STATUS & 0xF8;
103    if ( (twst == TW_MT_SLA_NACK )||(twst ==TW_MR_DATA_NACK) )
104    {
105        /* device busy, send stop condition to terminate write
106        operation */
107        TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);
108
109        // wait until stop condition is executed and bus released
110        while(TWCR & (1<<TWSTO));
111
112        continue;
113    }
114    //if( twst != TW_MT_SLA_ACK) return 1;
115    break;
116 }
117 /* i2c_start_wait */
118
119 /*
120 ****
121 Issues a repeated start condition and sends address and transfer
122 direction
123 Input:   address and transfer direction of I2C device
124
125 Return:  0 device accessible
126          1 failed to access device
127 ****
128 */
129 unsigned char i2c_rep_start(unsigned char address)
130 {
131     return i2c_start( address );
132 }
133 /* i2c_rep_start */
134
135 /*
136 ****
137 Terminates the data transfer and releases the I2C bus
138 ****
139 */
140 void i2c_stop(void)
141 {
142     /* send stop condition */

```

```

141 TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);
142
143 // wait until stop condition is executed and bus released
144 while(TWCR & (1<<TWSTO));
145
146 /* i2c_stop */
147
148 /*
149 */
150
151
152 Send one byte to I2C device
153
154 Input: byte to be transferred
155 Return: 0 write successful
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175 */
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190 */
191
192
193
194

```

```
    */
195 unsigned char i2c_readNak(void)
196 {
197     TWCR = (1<<TWINT) | (1<<TWEN);
198     while (!(TWCR & (1<<TWINT)));
199
200     return TWDR;
201
202 }/* i2c_readNak */
```

Algorithm 6.15: i2cmaster.cpp

# Bibliography

- [1] P. Fleury, *Interfacing a hd44780 based lcd to an avr*. [Online]. Available: <http://www.peterfleury.epizy.com/avr-lcd44780.html> (visited on 2021).
- [2] R. Couto, *Arduino songs*. [Online]. Available: <https://github.com/robsoncouto/arduino-songs> (visited on 2021).
- [3] *Electrical waveforms and electrical signals*. [Online]. Available: <https://www.electronics-tutorials.ws/waveforms/waveforms.html> (visited on 2021).
- [4] *I/o port operations in avr*. [Online]. Available: <https://maxembedded.com/2011/06/port-operations-in-avr/> (visited on 2021).
- [5] *4x4 keypad interfacing with atmega32 microcontroller*. [Online]. Available: <https://www.electronics-tutorials.ws/waveforms/waveforms.html> (visited on 2021).
- [6] ATMEL, *Atmega 328p data sheet*. [Online]. Available: [https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P\\_Datasheet.pdf](https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf) (visited on 2021).
- [7] XIAMEN, *16x2 lcd module*. [Online]. Available: [https://components101.com/sites/default/files/component\\_datasheet/16x2%5C%20LCD%5C%20Datasheet.pdf](https://components101.com/sites/default/files/component_datasheet/16x2%5C%20LCD%5C%20Datasheet.pdf) (visited on 2021).