Department of Electronic & Telecommunication Engineering
University of Moratuwa

Digital Alarm Clock

Ayodya W.K.H	190065K
Bandara D.P.S.D	190070V
Bandara D.R.K.W.M.S.D	190071B
Bandara E.M.D.A	190072E

Supervisor: Mr.Pramitha Muthukudaarachchi

This report is submitted as partial fulfillment of module EN1093

August 2021

Table of Content

Digital Alarm Clock

Group 4

Dept of Electronic and Telecommunication Engineering, University of Moratuwa

ABSTRACT

Time plays an important role in our daily activities, where there is need for accurate time management. This paper focus on the development and implementation of an improved digital alarm clock using Atmega328P Microcontroller and other electronics components such as 16x2 LCD display, RTC IC, 4x3 keypad, piezo buzzer as an I/O device. This digital alarm clock is a standard alarm clock with all necessary functionalities.

1. INTRODUCTION

Digital clock is a type of clock that displays time manually. This alarm clock is a fully functional programmable digital alarm clock with piezo buzzer as sound output.

DS3231 RTC module is used to keep track of the current time. ATmega328P handles storage, processing, and input/output tasks. The time, date, alarms, and all other UI components such as menus and tones are displayed using the 16x2 LCD display. 4x3 keypad is used for user inputs and UI controls. Keys on the keypad have been defined and allow the user to interface with a menu display on LCD module. Menu options range from core clock functionality like setting the current time as well as alarm timing. There is a factory reset button located inside the enclosure. A separate button is added for the alarm interruptions.

RTC module, LCD display, Piezo buzzer, keypad, and buttons are directly connected to the Atmega328P Microcontroller.

2. METHODOLOGY

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

2.1.Specifications

Dimensions :
Microcontroller : ATmega328P
RTC Module : DS3231

2.2.RTC Module

The DS3231 RTC Module is working with the I2C protocol with an inbuilt battery chamber so even when the power is not available. The module is temperature dependable and so keeping it separated from main PCB is the solution for heating. The chip maintains seconds, minutes, hours, date, month, and year information. The date at the end of the month is automatically adjusted for months fewer than 31 days, including corrections for leap year.

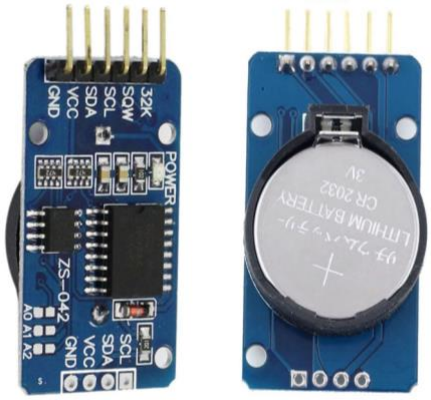


Figure 2.2.DS3231 RTC Module

Features

- 400kHz I2C Interface
- 3V CR2032 Battery Backup
- 32 Bytes AT24C32 EEPROM

The module operates in either the 24-hour or 12-hour format with an AM/PM indicator.

2.3.Buzzer

The active piezo buzzer is the sound output device in the digital alarm clock. It uses square waves to generate tones. The notes and time ratios will be used to generate tone sequences. There are five songs and these will be used to trigger the alarm alerts. This data is stored in program memory to utilize storage properly. With use of PC0 pin the buzzer is connected to the Atmega328P microcontroller.

Figure 2.3.Active Piezo Buzzer

Features

- Active Buzzer
- Amplifier created with BJT for better sound

Since this system is working as the alarm output it is needed to implement an interrupt to the system.

2.4.LCD Display

The 16x2 LCD display achieves a large viewing area in a compact package. The LCD can work in two different modes, namely the 4-bit mode and 8-bit mode.

For utilization of pins effectively 4-bit configuration is used. Total 6 pins are connected with the microcontroller.

The main functionalities of the LCD display is to display

- Time/date
- Control Menu
- Alarm Selection and Tone Selection

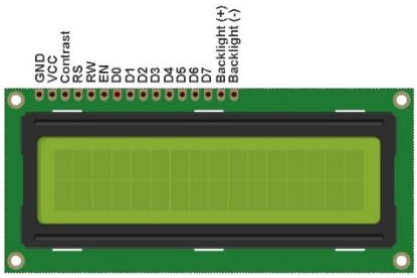


Figure 2.4.16x2 LCD Display

Features

- Available in Green and Blue Backlight
- Alphanumerical LCD display module
- Always ON display

All the menu display will be done by algorithms.

2.5.Keypad Input

The 4x3 matrix keypad is used as the input device. The number keys are used as basic number inputs.

Other than the keypad, four buttons are used for

- UP
- DOWN
- SELECT
- DESELECT

Seven pins are connected with the microcontroller for communication.

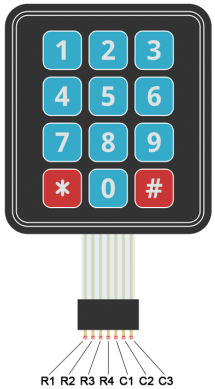


Figure 2.5.4x3 Matrix Keypad

Features

- Multiple buttons sharing less ports
- Good debouncing

Reducing debouncing effect helps to improve the user experience.

2.6.Main Circuit

2.6.1. Micro controller

The ATmega328P which is a single-chip microcontroller created by Atmel in the megaAVR family is used as the microcontroller in this device. It has a modified Harvard architecture 8-bit RISC processor core.

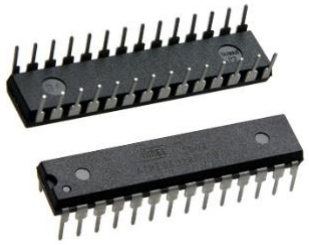


Figure 2.6.1.1. ATmega328P Microcontroller

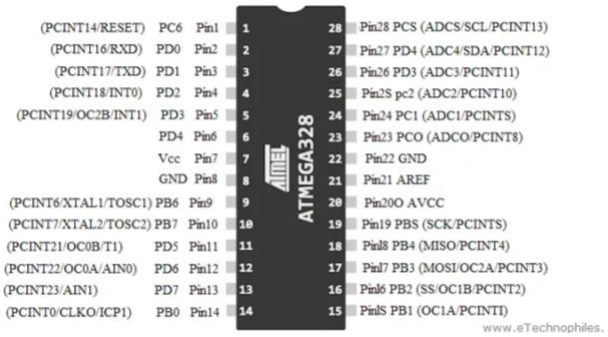


Figure 2.6.1. 2.ATmega238P Pin Configuration

Features

- Maximum CPU speed : 20MHz
- Flash Memory : 32kB
- SRAM : 2kB
- EEPROM : 1kB
- Pin Count : 32
- Maximum I/O pins : 23
- External Interrupts : 2

2.6.2. PCB

Components mounted on PCB

- 16MHz Crystal Oscillator
- LM7805 Voltage Regulator
- Capacitors
  - 1µF Capacitors
  - 22nF Capacitors
  - 20nF Capacitors
  - 100nF Capacitors
  - 330nF Capacitors
- Resistors
  - 750Ω Resistors
  - 1kΩ Resistors
  - 10kΩ Resistors
- Push Buttons
- NPN Transistor

Designing the PCB is a very important step, because all the other factors depend on the reliability of the PCB.

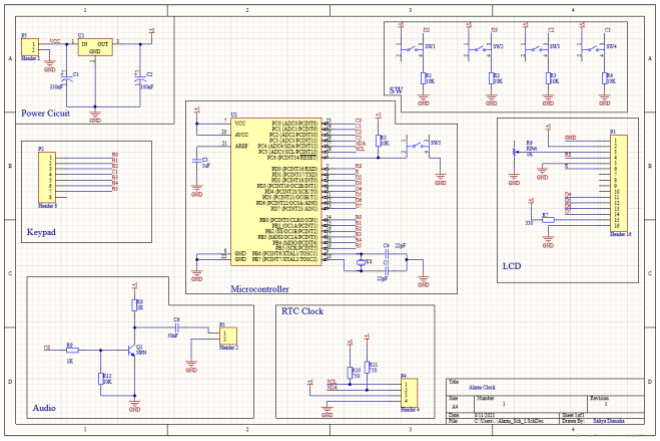


Figure 2.6.2.1. Schematic Design

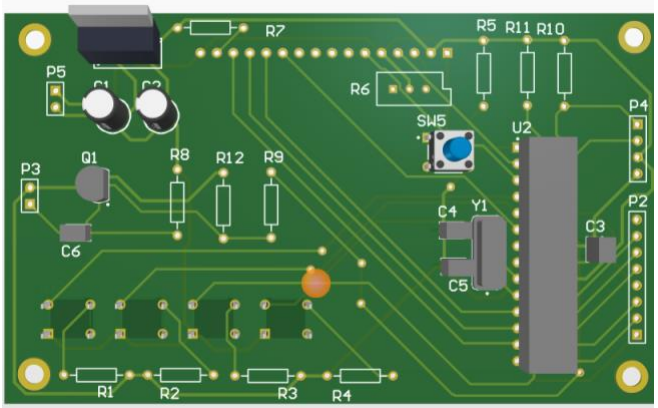


Figure 2.6.2.2. PCB Design

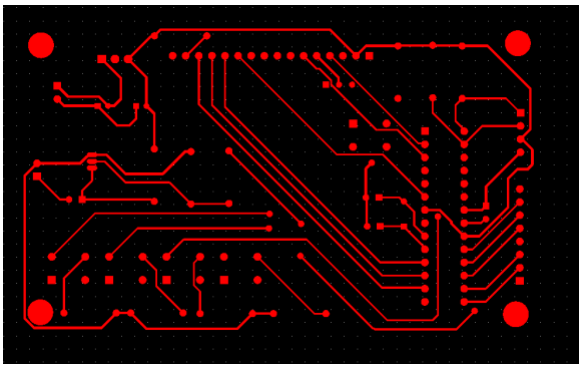


Figure 2.6.2.3. Top layer of PCB Design

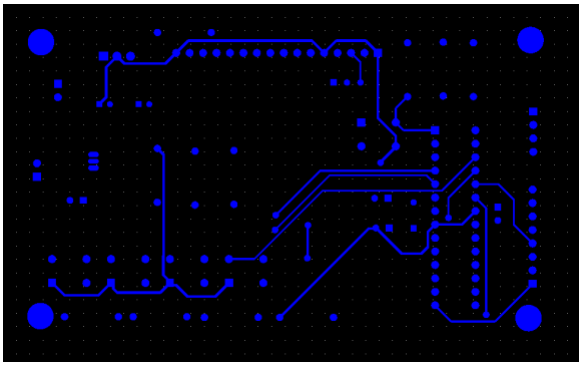


Figure 2.6.2.4. Bottom layer of PCB Design

Size of the PCB : 6cm x 10cm

2.7.Battery

18650 Lithium-ion rechargeable Batteries are used.

- Capacity : 1200mAh – 3600mAh
- Voltage : 3.7V
- Charge Density : 1.5Wh – 11.5Wh
- Shelf life : 36+ Months
- Cell count : 2
- Dimensions : 18mm x 65mm cylindrical size (diameter x height)

2.8.Enclosure Design

The enclosure is designed with a glossy finish to get a futuristic appearance.
Material : Acrylonitrile Butadiene Styrene (ABS) Plastics

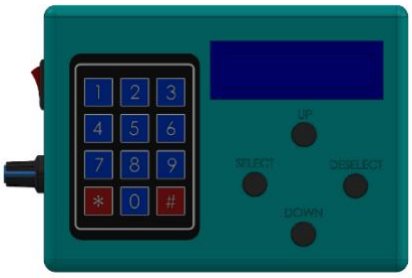


Figure 2.8.1. Front view of the Enclosure

Figure 2.8.2. Back View of the Enclosure

Figure 2.8.3. Side view of the Enclosure

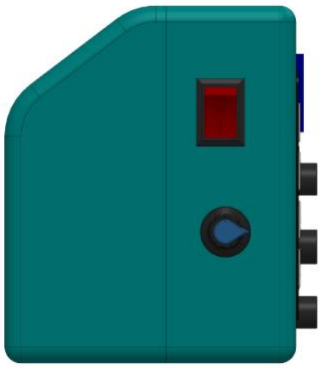
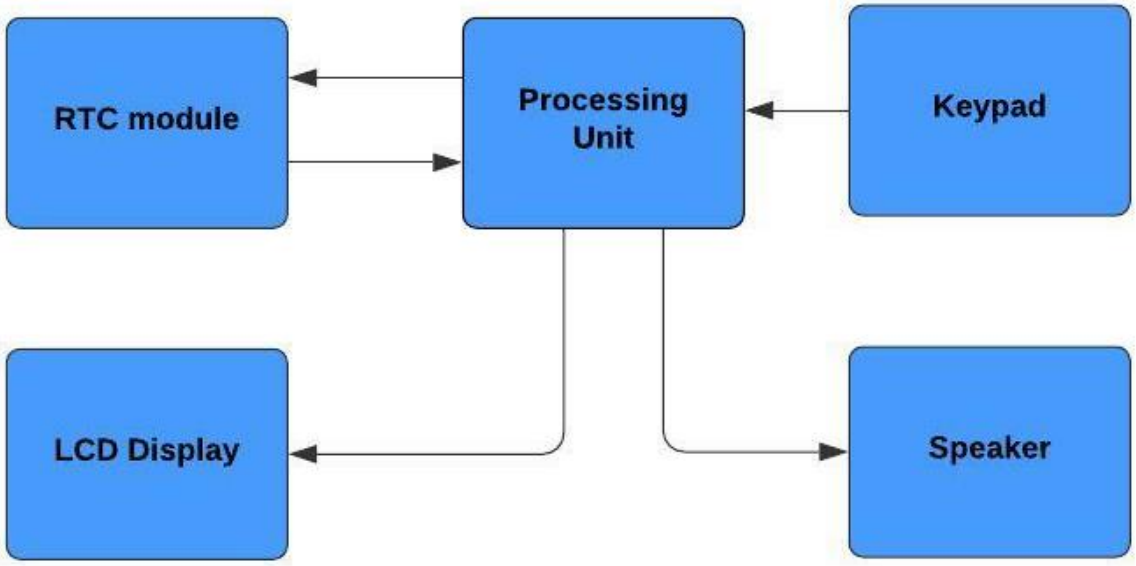


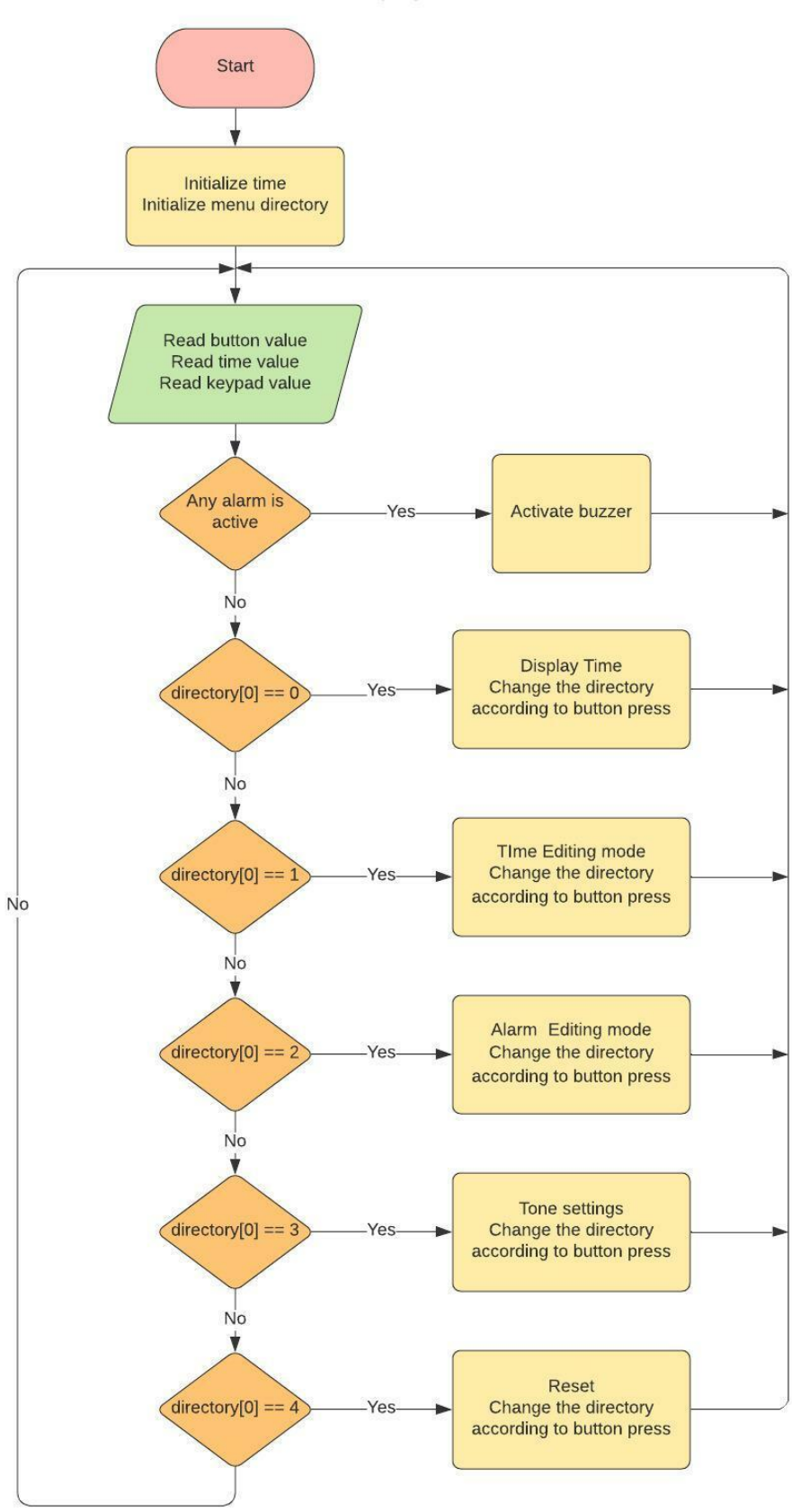
Figure 2.8.4. Side view of the Enclosure

2.9.Block Diagram

The following block diagram illustrates data flow between components in the circuit.



2.10.Algorithm



3. RESULTS

Algorithm was thoroughly checked for correctness and execution. After several iterations we were able to build smooth transitions between displays. Alarming was executed properly at previously assigned times. External interrupt was able to stop it.

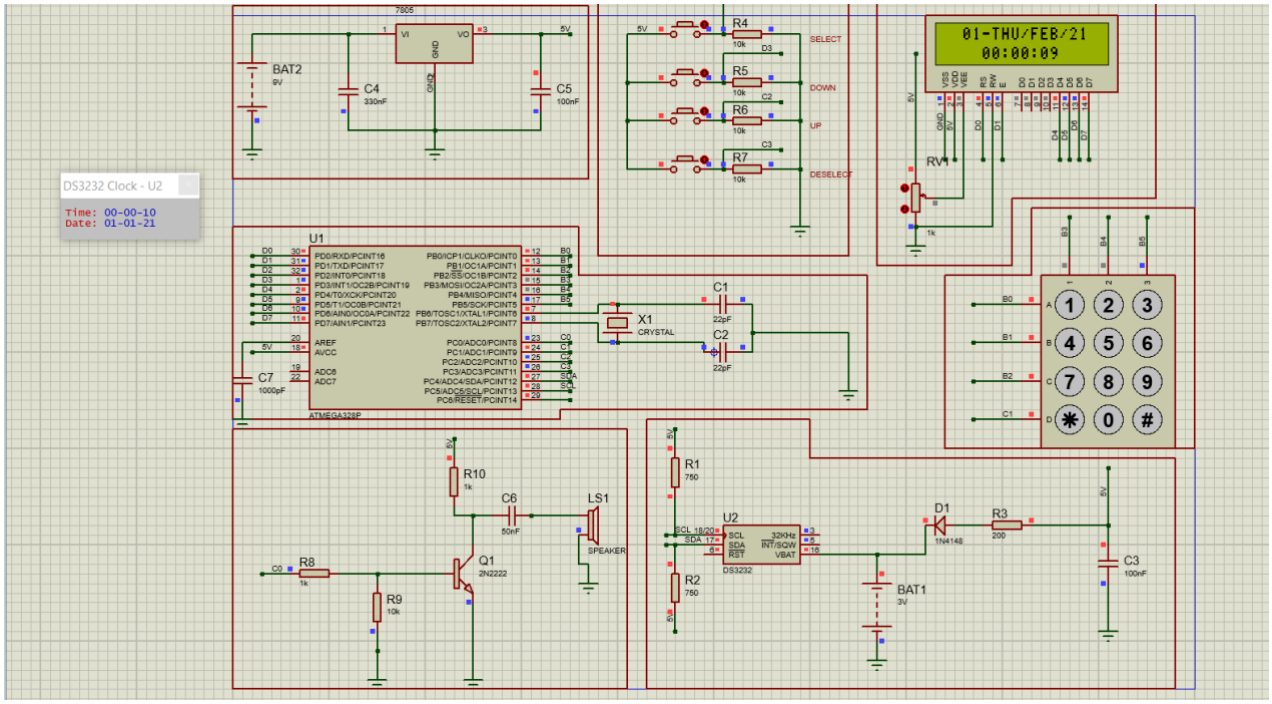


Figure 3.1. Stimulation

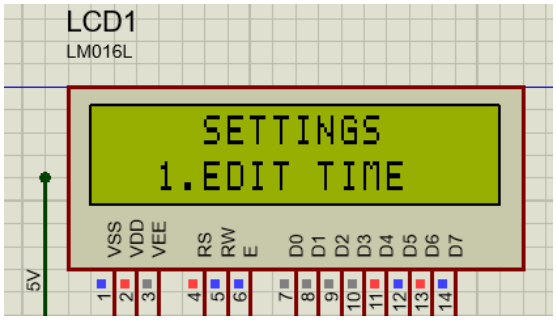


Figure 3.2.Edit Time

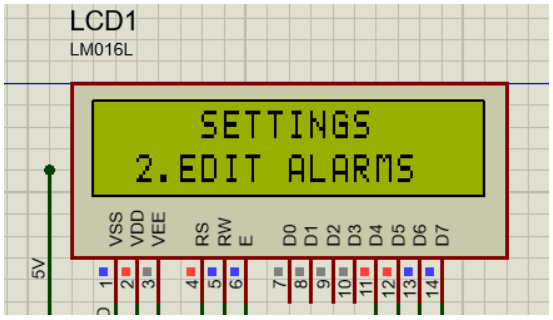


Figure 3.3.Edit Alarms

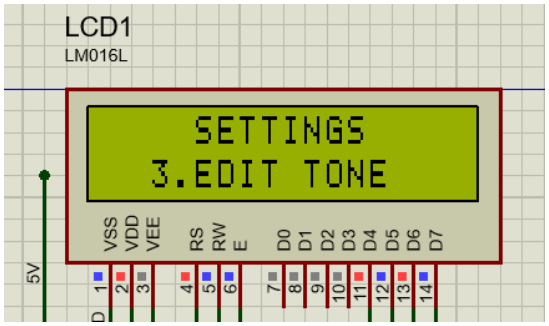


Figure 3.4.Edit Tone

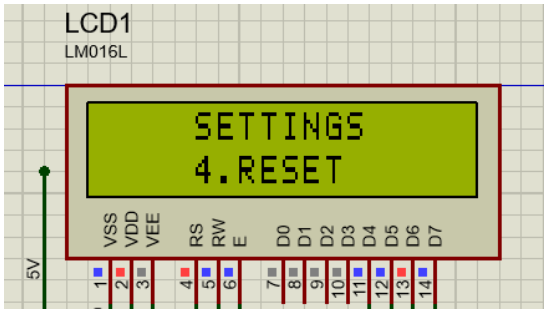


Figure 3.5.Reset

4. DISCUSSION

We had to face a lot of challenges in designing and implementing the digital alarm clock.

Problems faced while carrying on the project

- Insufficient memory due to data of long songs
- Difficulty in dividing ports among microproducts
- Insufficient space usage in first iteration of PCB design
- Large footprint in the first iteration of solidworks design

Solutions for these problems

- Taking short music snippets in the code to preserve memory
- First analyzing requirement of each sub product. Some equipment demand specific ports of the microprocessor. Therefor we have carefully divided ports in a way such the effect between sub-products are minimal
- To make the density higher. Rethink the PCB layout and carryout several iterations of PCB design to create an efficient layout
- We analyze the space requirement of each component. Then arrange them in a way to save space while delivering the futuristic look.

5. ACKNOWLEDGEMENTS

The completion of digital alarm clock was not an easy task for us we faced a lot of challenges since this is our first project. There are many personnel behind the accomplishment of this project.

First, we would like to pay our gratitude to our supervisor Ms. Pramitha Muthukudaarachchi and Ms. Pasan Dissanayake who always encouraged self-studying and guided us whenever we needed help. The weekly meetings held by our supervisor were very helpful to clear our doubts and discuss issues regarding the project.

In addition, we would like to convey our sincere gratitude to all the lecturers and instructors who were always willing to share their knowledge with us.
It is our duty to thank our ENTC family, without whom we could not have accomplished this project. Everyone in our batch helped each other and learnt everything together.

We developed team spirit working together in this project for about 2 months and all our group members gave their maximum contribution to succeed the given project.

Further, we would like to thank all the people who are not mentioned here for the great support provided.

6. REFERENCES

https://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf

https://datasheets.maximintegrated.com/en/ds/DS3231.pdf

https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

https://www.pololu.com/file/download/piezo-buzzer-datasheet.pdf?file_id=0J226

7.APPENDICES

7.1. Header File initializePorts.h

```
/*
 * initializePorts.h
 *
 * Created: 6/22/2021 10:45:35 AM
 *  Author: USER
 */



#ifndef INITIALIZEPORTS_H_
#define INITIALIZEPORTS_H_

//defining busses, ports and pins
#define BUS_B DDRB
#define BUS_C DDRC
#define BUS_D DDRD
#define PORT_B PORTB
#define PORT_C PORTC
#define PORT_D PORTD
#define PIN_B PINB
#define PIN_C PINC
#define PIN_D PIND

//defining pins required for lcd
#define RS PORTD0
#define EN PORTD1
#define LCD_DATA0 PORTD4
#define LCD_DATA1 PORTD5
#define LCD_DATA2 PORTD6
#define LCD_DATA3 PORTD7

//defining pins required for keypad
#define KEYPAD_R1 0
#define KEYPAD_R2 1
#define KEYPAD_R3 2
#define KEYPAD_R4 3
#define KEYPAD_C1 4
#define KEYPAD_C2 5
#define KEYPAD_C3 1

//defining pins required for buttons
#define BUTTON_SELECT  2
#define BUTTON_DOWN     3
#define BUTTON_UP       2
#define BUTTON_DESELCT 3

//defining pins required for audio-output
#define AUDIO_OUT PORTC0

//defining pins required for rtc
#define RTC_0  PORTC0
#define RTC_1  PORTC1

#include <avr/io.h>
```

```
void initializePorts(){

	//initialize data direction of port **Check for direction correction
	BUS_B = 0b11110000;
	BUS_C = 0b0000000;
	BUS_D = 0b11110011;


}

#define SET 0
#define CLEAR 1
#define VIEW 2

#endif /* INITIALIZEPORTS_H_ */
```

7.2.Header File lcd.h

```
/*
 * lcd.h
 *
 * Created: 6/21/2021 10:03:18 PM
 *  Author: USER
 */


#ifndef LCD_H_
#define LCD_H_

#ifndef F_CPU
#define F_CPU 16000000UL
#endif

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

#define LCD_DIRECTORY  DDRD /*Define LCD port*/
#define LCD_Port PORTD            /* Define LCD data port */
#define RS PORTD0           /* Define Register Select pin */
#define EN PORTD1                 /* Define Enable Signal pin */

void sendCommand( unsigned char command );
void sendCharacter( unsigned char data );
void initializeLCD (void);
void sendString (char *str);
void sendString_XY (char row, char pos, char *str);
void clearDisplay();


void sendCommand( unsigned char command )
{
	LCD_Port = (LCD_Port & 0b00001111) | (command & 0b11110000); /* sending first four
bits */
	LCD_Port &= ~ (1<<RS);
	 /* RS=0 to enable command registry */
```

```
	LCD_Port |= (1<<EN);
/* Sending enabling pulse */
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);

	_delay_us(200);

	LCD_Port = (LCD_Port & 0b00001111) | (command << 4);		/* sending last four
bits */
	LCD_Port |= (1<<EN);
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);
	_delay_ms(2);
}


void sendCharacter( unsigned char data )
{
	LCD_Port = (LCD_Port & 0b00001111) | (data & 0b11110000);	/* sending first
four bits */
	LCD_Port |= (1<<RS);
	/* RS=1 to enable data registry */
	LCD_Port|= (1<<EN);
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);

	_delay_us(200);

	LCD_Port = (LCD_Port & 0b00001111) | (data << 4);			/* sending
last four bits */
	LCD_Port |= (1<<EN);
	_delay_us(1);
	LCD_Port &= ~ (1<<EN);
	_delay_ms(2);
}

void initializeLCD (void)		/*Initializing the LCD*/
{
	LCD_DIRECTORY = 0b11111111;	/*Select LCD port as output */
	_delay_ms(20);				/* Accommodating power on delay */

	sendCommand(0b00000010);		/* Initialize LCD  */
	sendCommand(0b00101000);		/* 2 line, 5*7 matrix in 4-bit
mode configurations*/
	sendCommand(0b00001100);		/* Display on cursor off*/
	sendCommand(0b00000110);		/* Increment cursor (shift cursor
to right)*/
	sendCommand(0b00000001);		/* Clear display screen*/
	_delay_ms(2);
}


void sendString (char *str)	/* Send string to LCD function */
{
	int i;
	for(i=0;str[i]!=0;i++)		/* Send each letter to LCD */
	{
		sendCharacter(str[i]);
```

```
        }
}

void sendString_XY (char row, char pos, char *str)      /* Send string to LCD with x,y
position */
{
        if (row == 0 && pos<16)
        sendCommand((pos & 0b00001111)|0b10000000);             /* Command of first row and
required position<16 */
        else if (row == 1 && pos<16)
        sendCommand((pos & 0b00001111)|0b11000000);             /* Command of first row and
required position<16 */
        sendString(str);                                                        /* Call LCD
string function */
}

void clearDisplay()
{
        sendCommand(0b00000001);           /* Command to clear the display */
        _delay_ms(2);
        sendCommand(0b10000000);           /* Set cursor to home */

}


#endif /* LCD_H_ */
```

7.3.Header File tone.h

```
/*
 * tone.h
 *
 * Created: 8/11/2021 6:15:40 PM
 *  Author: USER
 */


#ifndef TONE_H_
#define TONE_H_

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
```

```
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
```

```
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
#define REST 0

extern int notes[];
extern int durations[];
extern int notes_G[];
extern int duration_G[];
extern int melody[];
extern int noteDurations[];
extern int StarWars[];
extern int GodFather[];

int notes[] = {
       NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,
       NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
       NOTE_C5, NOTE_D5, NOTE_B4, NOTE_B4, 0,
       NOTE_A4, NOTE_G4, NOTE_A4, 0,

       NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,
       NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
       NOTE_C5, NOTE_D5, NOTE_B4, NOTE_B4, 0,
       NOTE_A4, NOTE_G4, NOTE_A4, 0,

       NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,
       NOTE_A4, NOTE_C5, NOTE_D5, NOTE_D5, 0,
       NOTE_D5, NOTE_E5, NOTE_F5, NOTE_F5, 0,
       NOTE_E5, NOTE_D5, NOTE_E5, NOTE_A4, 0,

       NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
       NOTE_D5, NOTE_E5, NOTE_A4, 0,
       NOTE_A4, NOTE_C5, NOTE_B4, NOTE_B4, 0,
       NOTE_C5, NOTE_A4, NOTE_B4, 0,

       NOTE_A4, NOTE_A4};


int durations[] = {
       125, 125, 250, 125, 125,
       125, 125, 250, 125, 125,
       125, 125, 250, 125, 125,
       125, 125, 375, 125,

       125, 125, 250, 125, 125,
       125, 125, 250, 125, 125,
       125, 125, 250, 125, 125,
```

```
        125, 125, 375, 125,

        125, 125, 250, 125, 125,
        125, 125, 250, 125, 125,
        125, 125, 250, 125, 125,
        125, 125, 125, 250, 125,

        125, 125, 250, 125, 125,
        250, 125, 250, 125,
        125, 125, 250, 125, 125,
        125, 125, 375, 375,

        250, 125};


int notes_G[]={

        NOTE_G4,      NOTE_C4,      NOTE_DS4,     NOTE_F4,
        NOTE_G4,      NOTE_C4,      NOTE_E4,      NOTE_F4,
        NOTE_G4,      NOTE_C4,      NOTE_E4,      NOTE_F4,
        NOTE_G4,      NOTE_C4,      NOTE_DS4,     NOTE_F4,
        NOTE_D4,      NOTE_G3,      NOTE_AS3,     NOTE_C4,
        NOTE_D4,      NOTE_F4,      NOTE_AS3,     NOTE_DS4,
        NOTE_D4,      NOTE_F4,      NOTE_AS3,     NOTE_DS4,
        NOTE_D4,      NOTE_C4,      NOTE_GS3,     NOTE_AS3,
        NOTE_C4,      NOTE_F3,      NOTE_G4,      NOTE_C4,
        NOTE_DS4,     NOTE_F4,      NOTE_G4,      NOTE_C4,
        NOTE_DS4,     NOTE_F4,      NOTE_D4,      NOTE_G3,
        NOTE_AS3,     NOTE_C4,      NOTE_D4,      NOTE_G4,
        NOTE_C4,      NOTE_DS4,     NOTE_F4,      NOTE_G4,
        NOTE_C4,      NOTE_E4,      NOTE_F4,      NOTE_G4,
        NOTE_C4,      NOTE_E4,      NOTE_F4,      NOTE_G4,
        NOTE_C4,      NOTE_DS4,     NOTE_F4,      NOTE_D4,
        NOTE_G3,      NOTE_AS3,     NOTE_C4,      NOTE_D4,
        NOTE_F4,      NOTE_AS3,     NOTE_DS4,     NOTE_D4,
        NOTE_F4,      NOTE_AS3,     NOTE_DS4,     NOTE_D4,
        NOTE_C4,      NOTE_GS3,     NOTE_AS3,     NOTE_C4,
        NOTE_F3,      NOTE_G4,      NOTE_C4,      NOTE_DS4,
        NOTE_F4,      NOTE_G4,      NOTE_C4,      NOTE_DS4,
        NOTE_F4,      NOTE_D4,      NOTE_G3,      NOTE_AS3,
        NOTE_C4,      NOTE_D4
        };


int duration_G[]={

        500, 500, 250, 250, 500, 500, 250, 250, 1500, 1500, 250, 250, 1000,
        1000, 250, 250, 500, 500, 250, 250, 1500, 1500, 1000, 250, 250, 1000,
        1000, 250, 250, 500, 250, 250, 500, 500, 1000, 1000, 250, 250, 1000,
        1000, 250, 250, 500, 500, 250, 250, 500, 500, 500, 250, 250, 500, 500,
        250, 250, 1500, 1500, 250, 250, 1000, 1000, 250, 250, 500, 500, 250,
        250, 1500, 1500, 1000, 250, 250, 1000, 1000, 250, 250, 500, 250, 250,
        500, 500, 1000, 1000, 250, 250, 1000, 1000, 250, 250, 500, 500, 250,
        250, 500

        };
```

```
int melody[] = {
	NOTE_D5, NOTE_CS5, NOTE_B4, NOTE_FS4,
	NOTE_FS4, NOTE_FS4, NOTE_FS4, NOTE_FS4,
	NOTE_B4, NOTE_B4, NOTE_B4, NOTE_B4,
	NOTE_B4, NOTE_A4, NOTE_B4, NOTE_G4,
	NOTE_G4, NOTE_G4, NOTE_G4, NOTE_G4,
	NOTE_B4, NOTE_B4, NOTE_B4, NOTE_B4,
	NOTE_B4, NOTE_CS4, NOTE_D5, NOTE_A4,
	NOTE_A4, NOTE_A4, NOTE_A4, NOTE_A4,
	NOTE_D4, NOTE_D4, NOTE_D4, NOTE_D4,
	NOTE_D4, NOTE_E4, NOTE_E4, NOTE_CS4,
	NOTE_G4, NOTE_G4, NOTE_G4, NOTE_G4,
	NOTE_B4, NOTE_B4, NOTE_B4, NOTE_B4,
	NOTE_B4, NOTE_CS5, NOTE_D5, NOTE_A4,
	NOTE_A4, NOTE_A4, NOTE_A4, NOTE_A4,
	NOTE_D4, NOTE_D4, NOTE_D4, NOTE_D4,
	NOTE_D4, NOTE_E4, NOTE_E4, NOTE_CS4,
	0,
	};


int noteDurations[] = {
		8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 4, 4,
		8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 4, 4,
		8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 4, 2,
		8,
	};



int StarWars[] = {
	NOTE_B4,-4, NOTE_E5,-4, NOTE_B4,-4, NOTE_E5,-4,
	NOTE_B4,8,  NOTE_E5,-4, NOTE_B4,8, REST,8,  NOTE_AS4,8, NOTE_B4,8,
	NOTE_B4,8,  NOTE_AS4,8, NOTE_B4,8, NOTE_A4,8, REST,8, NOTE_GS4,8, NOTE_A4,8,
NOTE_G4,8,
	NOTE_G4,4,  NOTE_E4,-2,
	NOTE_B4,-4, NOTE_E5,-4, NOTE_B4,-4, NOTE_E5,-4,
	NOTE_B4,8,  NOTE_E5,-4, NOTE_B4,8, REST,8,  NOTE_AS4,8, NOTE_B4,8,

	NOTE_A4,-4, NOTE_A4,-4, NOTE_GS4,8, NOTE_A4,-4,
	NOTE_D5,8,  NOTE_C5,-4, NOTE_B4,-4, NOTE_A4,-4,
	NOTE_B4,-4, NOTE_E5,-4, NOTE_B4,-4, NOTE_E5,-4,
	NOTE_B4,8,  NOTE_E5,-4, NOTE_B4,8, REST,8,  NOTE_AS4,8, NOTE_B4,8,
	NOTE_D5,4, NOTE_D5,-4, NOTE_B4,8, NOTE_A4,-4,
	NOTE_G4,-4, NOTE_E4,-2,
	NOTE_E4, 2, NOTE_G4,2,
	NOTE_B4, 2, NOTE_D5,2,

	NOTE_F5, -4, NOTE_E5,-4, NOTE_AS4,8, NOTE_AS4,8, NOTE_B4,4, NOTE_G4,4
	};


int GodFather[]={
	REST, 4, REST, 8, REST, 8, REST, 8, NOTE_E4, 8, NOTE_A4, 8, NOTE_C5, 8, //1
	NOTE_B4, 8, NOTE_A4, 8, NOTE_C5, 8, NOTE_A4, 8, NOTE_B4, 8, NOTE_A4, 8, NOTE_F4,
8, NOTE_G4, 8,
```

```
	NOTE_E4, 2, NOTE_E4, 8, NOTE_A4, 8, NOTE_C5, 8,
	NOTE_B4, 8, NOTE_A4, 8, NOTE_C5, 8, NOTE_A4, 8, NOTE_C5, 8, NOTE_A4, 8, NOTE_E4,
8, NOTE_DS4, 8,

	NOTE_D4, 2, NOTE_D4, 8, NOTE_F4, 8, NOTE_GS4, 8, //5
	NOTE_B4, 2, NOTE_D4, 8, NOTE_F4, 8, NOTE_GS4, 8,
	NOTE_A4, 2, NOTE_C4, 8, NOTE_C4, 8, NOTE_G4, 8,
	NOTE_F4, 8, NOTE_E4, 8, NOTE_G4, 8, NOTE_F4, 8, NOTE_F4, 8, NOTE_E4, 8, NOTE_E4,
8, NOTE_GS4, 8,

	NOTE_A4, 2, REST,8, NOTE_A4, 8, NOTE_A4, 8, NOTE_GS4, 8, //9
	NOTE_G4, 2, NOTE_B4, 8, NOTE_A4, 8, NOTE_F4, 8,
	NOTE_E4, 2, NOTE_E4, 8, NOTE_G4, 8, NOTE_E4, 8,
	NOTE_D4, 2, NOTE_D4, 8, NOTE_D4, 8, NOTE_F4, 8, NOTE_DS4, 8,

	NOTE_E4, 2, REST, 8, NOTE_E4, 8, NOTE_A4, 8, NOTE_C5, 8, //13
	};

void play_tone(uint16_t delay, uint8_t duration) {
	// delay is half-period in microseconds
	// duration is in 10ms increments

	// example: 440Hz --> delay=1136


	// duration = 2*delay * cycles (all in same units)
	// cycles = 10000 * duration / delay / 2
	// cycles = 100 * duration / (delay/50)
	uint16_t tmp = 100 * duration;
	uint16_t delaysm = delay / 50;
	uint16_t cycles = tmp / delaysm;

	while(cycles > 0) {
		if (delay==NOTE_DS8){
			PORTC |= (1<<PC0);
			_delay_us(31);
			PORTC &= ~(1<<PC0);
		_delay_us(31);}


		if (delay==NOTE_D8){
			PORTC |= (1<<PC0);
			_delay_us(33);
			PORTC &= ~(1<<PC0);
		_delay_us(33);}


		if (delay==NOTE_CS8){
			PORTC |= (1<<PC0);
			_delay_us(35);
			PORTC &= ~(1<<PC0);
		_delay_us(35);}


		if (delay==NOTE_C8){
			PORTC |= (1<<PC0);
			_delay_us(37);
			PORTC &= ~(1<<PC0);
```

```
_delay_us(37);}


if (delay==NOTE_B7){
       PORTC |= (1<<PC0);
       _delay_us(39);
       PORTC &= ~(1<<PC0);
_delay_us(39);}


if (delay==NOTE_AS7){
       PORTC |= (1<<PC0);
       _delay_us(41);
       PORTC &= ~(1<<PC0);
_delay_us(41);}


if (delay==NOTE_A7){
       PORTC |= (1<<PC0);
       _delay_us(44);
       PORTC &= ~(1<<PC0);
_delay_us(44);}


if (delay==NOTE_GS7){
       PORTC |= (1<<PC0);
       _delay_us(46);
       PORTC &= ~(1<<PC0);
_delay_us(46);}


if (delay==NOTE_G7){
       PORTC |= (1<<PC0);
       _delay_us(49);
       PORTC &= ~(1<<PC0);
_delay_us(49);}


if (delay==NOTE_FS7){
       PORTC |= (1<<PC0);
       _delay_us(52);
       PORTC &= ~(1<<PC0);
_delay_us(52);}


if (delay==NOTE_F7){
       PORTC |= (1<<PC0);
       _delay_us(55);
       PORTC &= ~(1<<PC0);
_delay_us(55);}


if (delay==NOTE_E7){
       PORTC |= (1<<PC0);
       _delay_us(58);
       PORTC &= ~(1<<PC0);
_delay_us(58);}
```

```
if (delay==NOTE_DS7){
       PORTC |= (1<<PC0);
       _delay_us(62);
       PORTC &= ~(1<<PC0);
_delay_us(62);}


if (delay==NOTE_D7){
       PORTC |= (1<<PC0);
       _delay_us(65);
       PORTC &= ~(1<<PC0);
_delay_us(65);}


if (delay==NOTE_CS7){
       PORTC |= (1<<PC0);
       _delay_us(69);
       PORTC &= ~(1<<PC0);
_delay_us(69);}


if (delay==NOTE_C7){
       PORTC |= (1<<PC0);
       _delay_us(73);
       PORTC &= ~(1<<PC0);
_delay_us(73);}


if (delay==NOTE_B6){
       PORTC |= (1<<PC0);
       _delay_us(78);
       PORTC &= ~(1<<PC0);
_delay_us(78);}


if (delay==NOTE_AS6){
       PORTC |= (1<<PC0);
       _delay_us(82);
       PORTC &= ~(1<<PC0);
_delay_us(82);}


if (delay==NOTE_A6){
       PORTC |= (1<<PC0);
       _delay_us(87);
       PORTC &= ~(1<<PC0);
_delay_us(87);}


if (delay==NOTE_GS6){
       PORTC |= (1<<PC0);
       _delay_us(93);
       PORTC &= ~(1<<PC0);
_delay_us(93);}


if (delay==NOTE_G6){
```

```
        PORTC |= (1<<PC0);
        _delay_us(98);
        PORTC &= ~(1<<PC0);
_delay_us(98);}


if (delay==NOTE_FS6){
        PORTC |= (1<<PC0);
        _delay_us(104);
        PORTC &= ~(1<<PC0);
_delay_us(104);}


if (delay==NOTE_F6){
        PORTC |= (1<<PC0);
        _delay_us(110);
        PORTC &= ~(1<<PC0);
_delay_us(110);}


if (delay==NOTE_E6){
        PORTC |= (1<<PC0);
        _delay_us(117);
        PORTC &= ~(1<<PC0);
_delay_us(117);}


if (delay==NOTE_DS6){
        PORTC |= (1<<PC0);
        _delay_us(123);
        PORTC &= ~(1<<PC0);
_delay_us(123);}


if (delay==NOTE_D6){
        PORTC |= (1<<PC0);
        _delay_us(131);
        PORTC &= ~(1<<PC0);
_delay_us(131);}


if (delay==NOTE_CS6){
        PORTC |= (1<<PC0);
        _delay_us(139);
        PORTC &= ~(1<<PC0);
_delay_us(139);}


if (delay==NOTE_C6){
        PORTC |= (1<<PC0);
        _delay_us(147);
        PORTC &= ~(1<<PC0);
_delay_us(147);}


if (delay==NOTE_B5){
        PORTC |= (1<<PC0);
        _delay_us(156);
```

```
        PORTC &= ~(1<<PC0);
_delay_us(156);}


if (delay==NOTE_AS5){
        PORTC |= (1<<PC0);
        _delay_us(165);
        PORTC &= ~(1<<PC0);
_delay_us(165);}


if (delay==NOTE_A5){
        PORTC |= (1<<PC0);
        _delay_us(175);
        PORTC &= ~(1<<PC0);
_delay_us(175);}


if (delay==NOTE_GS5){
        PORTC |= (1<<PC0);
        _delay_us(185);
        PORTC &= ~(1<<PC0);
_delay_us(185);}


if (delay==NOTE_G5){
        PORTC |= (1<<PC0);
        _delay_us(196);
        PORTC &= ~(1<<PC0);
_delay_us(196);}


if (delay==NOTE_FS5){
        PORTC |= (1<<PC0);
        _delay_us(208);
        PORTC &= ~(1<<PC0);
_delay_us(208);}


if (delay==NOTE_F5){
        PORTC |= (1<<PC0);
        _delay_us(220);
        PORTC &= ~(1<<PC0);
_delay_us(220);}


if (delay==NOTE_E5){
        PORTC |= (1<<PC0);
        _delay_us(233);
        PORTC &= ~(1<<PC0);
_delay_us(233);}


if (delay==NOTE_DS5){
        PORTC |= (1<<PC0);
        _delay_us(247);
        PORTC &= ~(1<<PC0);
_delay_us(247);}
```

```
if (delay==NOTE_D5){
       PORTC |= (1<<PC0);
       _delay_us(262);
       PORTC &= ~(1<<PC0);
_delay_us(262);}


if (delay==NOTE_CS5){
       PORTC |= (1<<PC0);
       _delay_us(277);
       PORTC &= ~(1<<PC0);
_delay_us(277);}


if (delay==NOTE_C5){
       PORTC |= (1<<PC0);
       _delay_us(294);
       PORTC &= ~(1<<PC0);
_delay_us(294);}


if (delay==NOTE_B4){
       PORTC |= (1<<PC0);
       _delay_us(311);
       PORTC &= ~(1<<PC0);
_delay_us(311);}


if (delay==NOTE_AS4){
       PORTC |= (1<<PC0);
       _delay_us(330);
       PORTC &= ~(1<<PC0);
_delay_us(330);}


if (delay==NOTE_A4){
       PORTC |= (1<<PC0);
       _delay_us(349);
       PORTC &= ~(1<<PC0);
_delay_us(349);}


if (delay==NOTE_GS4){
       PORTC |= (1<<PC0);
       _delay_us(370);
       PORTC &= ~(1<<PC0);
_delay_us(370);}


if (delay==NOTE_G4){
       PORTC |= (1<<PC0);
       _delay_us(392);
       PORTC &= ~(1<<PC0);
_delay_us(392);}
```

```
if (delay==NOTE_FS4){
       PORTC |= (1<<PC0);
       _delay_us(415);
       PORTC &= ~(1<<PC0);
_delay_us(415);}


if (delay==NOTE_F4){
       PORTC |= (1<<PC0);
       _delay_us(440);
       PORTC &= ~(1<<PC0);
_delay_us(440);}


if (delay==NOTE_E4){
       PORTC |= (1<<PC0);
       _delay_us(466);
       PORTC &= ~(1<<PC0);
_delay_us(466);}


if (delay==NOTE_DS4){
       PORTC |= (1<<PC0);
       _delay_us(494);
       PORTC &= ~(1<<PC0);
_delay_us(494);}


if (delay==NOTE_D4){
       PORTC |= (1<<PC0);
       _delay_us(523);
       PORTC &= ~(1<<PC0);
_delay_us(523);}


if (delay==NOTE_CS4){
       PORTC |= (1<<PC0);
       _delay_us(554);
       PORTC &= ~(1<<PC0);
_delay_us(554);}


if (delay==NOTE_C4){
       PORTC |= (1<<PC0);
       _delay_us(587);
       PORTC &= ~(1<<PC0);
_delay_us(587);}


if (delay==NOTE_B3){
       PORTC |= (1<<PC0);
       _delay_us(622);
       PORTC &= ~(1<<PC0);
_delay_us(622);}


if (delay==NOTE_AS3){
       PORTC |= (1<<PC0);
```

```
        _delay_us(659);
        PORTC &= ~(1<<PC0);
_delay_us(659);}


if (delay==NOTE_A3){
        PORTC |= (1<<PC0);
        _delay_us(698);
        PORTC &= ~(1<<PC0);
_delay_us(698);}


if (delay==NOTE_GS3){
        PORTC |= (1<<PC0);
        _delay_us(740);
        PORTC &= ~(1<<PC0);
_delay_us(740);}


if (delay==NOTE_G3){
        PORTC |= (1<<PC0);
        _delay_us(784);
        PORTC &= ~(1<<PC0);
_delay_us(784);}


if (delay==NOTE_FS3){
        PORTC |= (1<<PC0);
        _delay_us(831);
        PORTC &= ~(1<<PC0);
_delay_us(831);}


if (delay==NOTE_F3){
        PORTC |= (1<<PC0);
        _delay_us(880);
        PORTC &= ~(1<<PC0);
_delay_us(880);}


if (delay==NOTE_E3){
        PORTC |= (1<<PC0);
        _delay_us(932);
        PORTC &= ~(1<<PC0);
_delay_us(932);}


if (delay==NOTE_DS3){
        PORTC |= (1<<PC0);
        _delay_us(988);
        PORTC &= ~(1<<PC0);
_delay_us(988);}


if (delay==NOTE_D3){
        PORTC |= (1<<PC0);
        _delay_us(1047);
        PORTC &= ~(1<<PC0);
```

```
_delay_us(1047);}


if (delay==NOTE_CS3){
       PORTC |= (1<<PC0);
       _delay_us(1109);
       PORTC &= ~(1<<PC0);
_delay_us(1109);}


if (delay==NOTE_C3){
       PORTC |= (1<<PC0);
       _delay_us(1175);
       PORTC &= ~(1<<PC0);
_delay_us(1175);}


if (delay==NOTE_B2){
       PORTC |= (1<<PC0);
       _delay_us(1245);
       PORTC &= ~(1<<PC0);
_delay_us(1245);}


if (delay==NOTE_AS2){
       PORTC |= (1<<PC0);
       _delay_us(1319);
       PORTC &= ~(1<<PC0);
_delay_us(1319);}


if (delay==NOTE_A2){
       PORTC |= (1<<PC0);
       _delay_us(1397);
       PORTC &= ~(1<<PC0);
_delay_us(1397);}


if (delay==NOTE_GS2){
       PORTC |= (1<<PC0);
       _delay_us(1480);
       PORTC &= ~(1<<PC0);
_delay_us(1480);}


if (delay==NOTE_G2){
       PORTC |= (1<<PC0);
       _delay_us(1568);
       PORTC &= ~(1<<PC0);
_delay_us(1568);}


if (delay==NOTE_FS2){
       PORTC |= (1<<PC0);
       _delay_us(1661);
       PORTC &= ~(1<<PC0);
_delay_us(1661);}
```

```
if (delay==NOTE_F2){
       PORTC |= (1<<PC0);
       _delay_us(1760);
       PORTC &= ~(1<<PC0);
_delay_us(1760);}


if (delay==NOTE_E2){
       PORTC |= (1<<PC0);
       _delay_us(1865);
       PORTC &= ~(1<<PC0);
_delay_us(1865);}


if (delay==NOTE_DS2){
       PORTC |= (1<<PC0);
       _delay_us(1976);
       PORTC &= ~(1<<PC0);
_delay_us(1976);}


if (delay==NOTE_D2){
       PORTC |= (1<<PC0);
       _delay_us(2093);
       PORTC &= ~(1<<PC0);
_delay_us(2093);}


if (delay==NOTE_CS2){
       PORTC |= (1<<PC0);
       _delay_us(2217);
       PORTC &= ~(1<<PC0);
_delay_us(2217);}


if (delay==NOTE_C2){
       PORTC |= (1<<PC0);
       _delay_us(2349);
       PORTC &= ~(1<<PC0);
_delay_us(2349);}


if (delay==NOTE_B1){
       PORTC |= (1<<PC0);
       _delay_us(2489);
       PORTC &= ~(1<<PC0);
_delay_us(2489);}


if (delay==NOTE_AS1){
       PORTC |= (1<<PC0);
       _delay_us(2637);
       PORTC &= ~(1<<PC0);
_delay_us(2637);}


if (delay==NOTE_A1){
```

```
        PORTC |= (1<<PC0);
        _delay_us(2794);
        PORTC &= ~(1<<PC0);
_delay_us(2794);}


if (delay==NOTE_GS1){
        PORTC |= (1<<PC0);
        _delay_us(2960);
        PORTC &= ~(1<<PC0);
_delay_us(2960);}


if (delay==NOTE_G1){
        PORTC |= (1<<PC0);
        _delay_us(3136);
        PORTC &= ~(1<<PC0);
_delay_us(3136);}


if (delay==NOTE_FS1){
        PORTC |= (1<<PC0);
        _delay_us(3322);
        PORTC &= ~(1<<PC0);
_delay_us(3322);}


if (delay==NOTE_F1){
        PORTC |= (1<<PC0);
        _delay_us(3520);
        PORTC &= ~(1<<PC0);
_delay_us(3520);}


if (delay==NOTE_E1){
        PORTC |= (1<<PC0);
        _delay_us(3729);
        PORTC &= ~(1<<PC0);
_delay_us(3729);}


if (delay==NOTE_DS1){
        PORTC |= (1<<PC0);
        _delay_us(3951);
        PORTC &= ~(1<<PC0);
_delay_us(3951);}


if (delay==NOTE_D1){
        PORTC |= (1<<PC0);
        _delay_us(4186);
        PORTC &= ~(1<<PC0);
_delay_us(4186);}


if (delay==NOTE_CS1){
        PORTC |= (1<<PC0);
        _delay_us(4435);
```

```
                PORTC &= ~(1<<PC0);
            _delay_us(4435);}


            if (delay==NOTE_C1){
                PORTC |= (1<<PC0);
                _delay_us(4699);
                PORTC &= ~(1<<PC0);
            _delay_us(4699);}


            if (delay==NOTE_B0){
                PORTC |= (1<<PC0);
                _delay_us(4978);
                PORTC &= ~(1<<PC0);
            _delay_us(4978);}

            cycles--;

        }
}



#endif /* TONE_H_ */
```

7.4.Source code file rtc.cpp

```
/*
 * rtc.cpp
 *
 * Created: 6/21/2021 10:05:21 PM
 *  Author: USER
 */
#ifndef F_CPU
#define  F_CPU 16000000UL
#endif

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

class RTC{
        public:
        #define RTC_WADDR 0b11010000//*
        #define RTC_RADDR 0b11010001//*
        #define rs PB0//*
        #define en PB1//*
        #define direction DDRD//*?
        #define port PORTD//*?
        RTC(int b);
        void init();
        void i2c_init();
        void i2c_start();
        void i2c_stop();
        void i2c_write(uint8_t data);
```

```
        unsigned char i2c_read();
        unsigned char i2c_lastread();
        unsigned char binTobcd(unsigned char data);
        unsigned char bcdTobin(unsigned char data);
        void clock_init();
        void ReadTime(int *sec, int *min, int *hour, int *day, int *wday, int *month, int
*year);
        void setTime(int sec, int min, int hour,int day, int mon,int wday, int year);
        void setDate(int day, int mon,int wday, int year);

};

#endif

/*
 * rtc.cpp
 *
 * Created: 6/21/2021 10:05:11 PM
 *  Author: USER
 */
#ifndef F_CPU
#define  F_CPU 16000000UL
#endif
#include <avr/io.h>
#include <util/delay.h>
#include "RTC.h"
#include <stdio.h>



RTC::RTC(int b)
{
        int c = b;
        c = c/3;
}

void RTC::i2c_init()
{
    TWBR = 0xFF;
}

void RTC::i2c_start()
{
    TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
    while ((TWCR & (1<<TWINT)) == 0);
}

void RTC::i2c_stop()
{
    TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);
}

void RTC::i2c_write(unsigned char data)
{
    TWDR = data;
    TWCR = (1<<TWINT)|(1<<TWEN);
    while ((TWCR & (1<<TWINT)) == 0);
```

```
}
unsigned char RTC::i2c_read()
{
    TWCR = (1 << TWINT) | (1 << TWEN)| (1 << TWEA);
    while ((TWCR & (1 << TWINT)) == 0);
    return TWDR;

}

unsigned char RTC::i2c_lastread()
{
       TWCR = (1 << TWINT) | (1 << TWEN);
       while ((TWCR & (1 << TWINT)) == 0);
       return TWDR;

}

unsigned char RTC::binTobcd(unsigned char data)
{


       return( (data/10*16) + (data%10) );
}

unsigned char RTC::bcdTobin(unsigned char data)
{

    return( (data/16*10) + (data%16) );
}


void RTC::clock_init()
{
    i2c_start();
    i2c_write(RTC_WADDR);
       i2c_write(0x0E);
       i2c_write(0x20);
       i2c_write(0x08);
    i2c_stop();

}

void RTC::setTime(int sec, int min, int hour,int day, int mon,int wday, int year)
{
    i2c_start();
    i2c_write(RTC_WADDR);
    i2c_write(0x00);
    i2c_write(binTobcd(sec));
    i2c_write(binTobcd(min));
    i2c_write(binTobcd(hour));

       i2c_write(binTobcd(wday));
       i2c_write(binTobcd(day));
       i2c_write(binTobcd(mon));
       i2c_write(binTobcd(year));
       i2c_stop();
}
```

```
void RTC::ReadTime(int *sec, int *min, int *hour, int *day, int *wday, int *month, int
*year)
{
    i2c_start();
    i2c_write(RTC_WADDR);
    i2c_write(0x00);
    i2c_stop();

    i2c_start();
    i2c_write(RTC_RADDR);
    *sec = bcdTobin(i2c_read());
    *min = bcdTobin(i2c_read());
    *hour = bcdTobin(i2c_read());

    *wday = bcdTobin(i2c_read());
    *day = bcdTobin(i2c_read());
    *month = bcdTobin(i2c_read());
    *year = bcdTobin(i2c_lastread());
    i2c_stop();

}

void RTC::init()
{
    i2c_init();
       clock_init();
}
```

7.5.Source code file AlarmClockFinal.cpp

```
/*
 * AlarmClockFinal.cpp
 *
 * Created: 6/21/2021 9:56:27 PM
 * Author : USER
 */

//defining clock frequency as 16MHz
#define F_CPU 16000000UL

//including library files
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <time.h>
#include <avr/pgmspace.h>
#include <inttypes.h>

//including user-built libraries
#include "lcd.h"
#include "userInputs.h"
#include "initializePorts.h"
#include "RTC.h"
#include "tone.h"
```

```
//initializing variable needed for song playback
int initial = 0;
const float songSpeed = 1.0;
const float songSpeed_G = 1.0;
int activeAlarm = -1;
int var=0;
int notone[]={9,10,11,13};
char* tones[5] = {"1.PIRATES ", "2.GOT THEME", "3.DESPACITO","4.STAR WARS","5.GOD
FATHER"};

//initializing variables related to RTC timing
int sec, min, hour, day, wday, month, year;
int temp_sec, temp_min, temp_hour, temp_day, temp_wday, temp_month, temp_year;
char* daysOfWeek[7] = {"SUN", "MON", "TUE", "WED", "THU", "FRI", "SAT"};
char* monthsOfYear[12] = {"JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP",
"OCT", "NOV", "DEC"};
int prevSecond;
int directory[3];

//initializing variables related to tone selection
int tone = 0;
int tempTone = tone;
int key;
int button;
int pos=0;
int mode=-1;
char* modeString;
void updateTime(int* tempSec, int* tempMin, int* tempHour, int* tempDay, int* tempWday,
int* tempMonth, int* tempYear, int *pos, int key);//Prototype function to change values
based on key press
void blinkDisplay(int position);
//initializing variables related to alarm set, change and selection
bool toDisplay;
int alarmNo = -1;
bool alarms[5] = {false, false, false, false, false};
int alarmTime[5][3] = {0};
bool reset = false;
bool isAlarmActive = false;

//temperory string for line 1 and line 2 of LCD display.
char tempString1[17];
char tempString2[17];

//RTC class active
RTC rtc(1);


//Functions for alarm sound playing
void play_tone(uint16_t delay, uint8_t duration);
void pla();
void sto();
void del(int ms);
void play_();

//External interrupt for alarm stop
ISR (INT0_vect)
{
```

```
        sto();


}

//Main function
int main(){

        //Intializing LCD, RTC
        initializeLCD();
        rtc.init();
        rtc.setTime(0,0,0,1,1,4,21);
        toDisplay = true;

        //Pieze as the output
        DDRC |= (1<<PC0);

        //set trigger of external interrut INT0 on rising-edge
        EICRA |= (1 << ISC01)|(1 << ISC00);
        EIMSK |= (1 << INT0);
        sei();



        while(1){

                _delay_ms(200);

                //reading button, rtc, and keypad values
                button = buttonInput();
                rtc.ReadTime(&sec, &min, &hour, &day, &wday, &month, &year);
                key = keyPadInput();

                //search whether alarm time has arrived
                for (int i=0; i<5; i++){
                        if((alarmTime[i][0] == hour) && (alarmTime[i][1] == min )&&
(alarms[i]) ) {
                                isAlarmActive = true;
                                activeAlarm = i;
                        }

                }

                //play tone if the alarm is active
                if (isAlarmActive){

                        pla();
                        play_();

                }

                //Directory wise search for identify the current display
                switch(directory[0]){

                        case 0://timeDisplay

                                if (toDisplay){
                                        clearDisplay();
```

```
                                        prevSecond = sec;
                                        sprintf(tempString1,"%02d-%s/%s/%02d", day,
daysOfWeek[wday], monthsOfYear[month], year);
                                        sprintf(tempString2,"%02d:%02d:%02d", hour, min, sec);
                                        sendString_XY(0,2,tempString1);
                                        sendString_XY(1,4,tempString2);
                                        toDisplay = false;
                                }

                                //write the display for every new second
                                else if (sec!=prevSecond){
                                        prevSecond = sec;
                                        sprintf(tempString1,"%02d-%s/%s/%02d", day,
daysOfWeek[wday], monthsOfYear[month], year);
                                        sprintf(tempString2,"%02d:%02d:%02d", hour, min, sec);
                                        sendString_XY(0,2,tempString1);
                                        sendString_XY(1,4,tempString2);
                                }

                                //Change the display for button DOWN
                                if (button == DOWN){
                                        directory[0] = 1;
                                        toDisplay = true;
                                }

                                break;

                        case 1://setTime

                                switch(directory[1]){

                                        case 0://Displaying the EDIT TIME
                                                if (toDisplay){
                                                        clearDisplay();
                                                        sendString_XY(0,0,"    SETTINGS");
                                                        sendString_XY(1,0,"  1.EDIT TIME");
                                                        toDisplay = false;
                                                }

                                                //Logic for button input
                                                if (button == UP) {
                                                        directory[0] = 0;
                                                        toDisplay = true;
                                                }
                                                else if (button == DOWN) {
                                                        directory[0]=2;
                                                        toDisplay = true;
                                                }
                                                else if (button == SELECT) {
                                                        directory[1]=1;
                                                        toDisplay = true;
                                                        clearDisplay();
                                                }
                                                break;

                                        case 1://Editing mode screen

                                                if (toDisplay){
```

```
                                        clearDisplay();
                                        temp_sec  = sec;
                                        temp_min  = min;
                                        temp_hour = hour;
                                        temp_day  = day;
                                        temp_wday = wday;
                                        temp_month = month;
                                        temp_year = year;

                                        sprintf(tempString1, "%02d-
%s/%s/%02d",temp_day, daysOfWeek[temp_wday], monthsOfYear[temp_month],year);
                                        sprintf(tempString2,"%02d:%02d:%02d",
temp_hour, temp_min, temp_sec);
                                        sendString_XY(0,2,tempString1);
                                        sendString_XY(1,4, tempString2);
                                        toDisplay = false;

                                }

                                //Logic for button input
                                if (button == DESELECT){
                                        toDisplay = true;
                                        directory[1] = 0;

                                }

                                //Change the values depend on keypad input
                                else if (key!=255){

                                        updateTime(&temp_sec, &temp_min,
&temp_hour, &temp_day, &temp_wday, &temp_month, &temp_year, &pos, key);

                                        if (pos==13){
                                                pos = 0;
                                                rtc.setTime(temp_sec, temp_min,
temp_hour, temp_day, temp_month, temp_wday, temp_year);
                                                directory[1]=0;
                                                toDisplay = true;
                                        }
                                }

                                blinkDisplay(pos);
                                sprintf(tempString1, "%02d-%s/%s/%02d",temp_day,
daysOfWeek[temp_wday], monthsOfYear[temp_month],temp_year);
                                sprintf(tempString2,"%02d:%02d:%02d", temp_hour,
temp_min, temp_sec);
                                sendString_XY(0,2,tempString1);
                                sendString_XY(1,4, tempString2);

                                break;
                        }

                        break;


                case 2://setAlarm

                        switch(directory[1]){
```

```
                        case 0://Displaying EDIT ALARMS
                                if (toDisplay){
                                        clearDisplay();
                                        sendString_XY(0,0,"    SETTINGS");
                                        sendString_XY(1,0," 2.EDIT ALARMS");
                                        toDisplay = false;
                                }

                                //Logic for button inputs
                                if (button == UP) {
                                        directory[0] = 1;
                                        toDisplay = true;
                                }
                                else if (button == DOWN) {
                                        directory[0]=3;
                                        toDisplay = true;
                                }
                                else if (button == SELECT) {
                                        directory[1]=1;
                                        toDisplay = true;
                                        clearDisplay();
                                }
                                break;


                        case 1://Selection of alarm slot

                                if (toDisplay){

                                        sendString_XY(0,0,"ENTER ALARM:");
                                        sendString_XY(0,13,"_");
                                        toDisplay = false;
                                }

                                if (button == UP) alarmNo = (alarmNo+1)%5;

                                else if (button == DOWN){
                                        alarmNo = alarmNo-1;
                                        if (alarmNo<0) alarmNo =0;
                                        alarmNo = alarmNo%5;
                                }

                                else if (button == SELECT){
                                        if (alarmNo!=-1){
                                                toDisplay = true;
                                                directory[1]=2;
                                        }
                                }

                                else if (button == DESELECT){
                                        alarmNo = -1;
                                        directory[1]=0;
                                        toDisplay = true;
                                }

                                sprintf(tempString1, "ENTER ALARM: %d ",
alarmNo+1);
```

```
                sendString_XY(0,0,tempString1);

                break;

            case 2://Selection of alarm mode

                if (toDisplay){
                    sendString_XY(1,0,"ENTER MODE:");
                    sendString_XY(1,13,"_");
                    toDisplay = false;
                }

                if (button == UP) mode = (mode+1)%3;

                else if (button == DOWN){
                    mode = mode-1;
                    if (mode<0) mode =4;
                    mode = mode%3;
                }

                else if (button == SELECT){
                    if (mode!=-1){
                        toDisplay = true;
                        directory[1]=3;
                        if (mode==SET) pos = 7;
                    }
                }

                else if (button == DESELECT){
                    mode = -1;
                    alarmNo = -1;

                    directory[1]=0;
                    toDisplay = true;


                }
                if (mode==SET) modeString = "SET";
                else if (mode==CLEAR) modeString = "CLR";
                else if (mode==VIEW) modeString ="VIEW";

                sprintf(tempString1, "ENTER MODE: %s ",
modeString);
                sendString_XY(1,0,tempString1);


                break;

            case 3://Clearing, Setting or Viewing the alarm

                //Clearing the alarm
                if (mode == CLEAR){

                    alarmTime[alarmNo][0] = 0;
                    alarmTime[alarmNo][1] = 0;
                    alarmTime[alarmNo][2] = 0;
                    alarms[alarmNo] = false;
                    toDisplay = true;
```

```
                                        directory[1]=0;
                                }

                                //Setting the alarm
                                else if (mode == SET){

                                        if (toDisplay) {
                                                clearDisplay();
                                                toDisplay = false;
                                        }
                                        if (key!=255){
                                                updateTime(&alarmTime[alarmNo][2],
&alarmTime[alarmNo][1], &alarmTime[alarmNo][0], &temp_day, &temp_wday, &temp_month,
&temp_year, &pos, key);
                                                if (pos==13){
                                                        alarms[alarmNo] = true;
                                                        pos = 0;
                                                        directory[1]=0;
                                                        toDisplay = true;
                                                }
                                        }


        sprintf(tempString1,"%02d:%02d:%02d", alarmTime[alarmNo][0],
alarmTime[alarmNo][1], alarmTime[alarmNo][2]);
                                                sendString_XY(0,0,tempString1);

                                                _delay_ms(200);
                                                sendString_XY(0,((pos-
7)/2)*2+(pos-6)/2,"_");

                                                if (button == DESELECT) {
                                                        mode = CLEAR;
                                                        pos = 0;
                                                        }

                                }

                                //Viewing the alarm
                                else if (mode==VIEW){

                                        if (toDisplay){
                                                clearDisplay();
                                                toDisplay = false;
                                        }

                                        sprintf(tempString1,"%02d:%02d:%02d",
alarmTime[alarmNo][0], alarmTime[alarmNo][1], alarmTime[alarmNo][2]);
                                        sendString_XY(0,0,tempString1);

                                        if (alarms[alarmNo])
sprintf(tempString2,"ACTIVE");
                                        else sprintf(tempString2,"INACTIVE");
                                        sendString_XY(1,0,tempString2);

                                        if (button==DESELECT) {
                                                toDisplay = true;
                                                directory[1]=0;
```

```
                                        }
                                }

                                break;
                        }

                        break;

                case 3://setTone

                        switch(directory[1]){
                                case 0:
                                        if (toDisplay){
                                                clearDisplay();
                                                sendString_XY(0,0,"    SETTINGS");
                                                sendString_XY(1,0,"  3.EDIT TONE");
                                                toDisplay = false;
                                        }

                                        if (button == UP) {
                                                directory[0] = 2;
                                                toDisplay = true;
                                        }
                                        else if (button == DOWN) {
                                                directory[0] = 4;
                                                toDisplay = true;
                                        }
                                        else if (button == SELECT) {
                                                directory[1]=1;
                                                toDisplay = true;
                                        }
                                        break;


                                case 1:
                                        if (toDisplay){
                                                clearDisplay();
                                                sendString_XY(0,0,tones[tempTone]);
                                                toDisplay = false;
                                        }

                                        if (button == UP) {
                                                if (tempTone!=0) tempTone = (tempTone-
1)%5;
                                                else tempTone = 4;
                                                toDisplay = true;
                                        }
                                        else if (button == DOWN) {
                                                tempTone = (tempTone+1)%5;
                                                toDisplay = true;
                                        }
                                        else if (button == SELECT) {
                                                tone = tempTone;
                                                directory[1] = 0;
                                                toDisplay = true;
                                        }

                                        else if (button == DESELECT) {
```

```
                                tempTone = tone;
                                directory[1] = 0;
                                toDisplay = true;
                        }
                        break;
                }
                break;

        case 4://Reset

                switch(directory[1]){
                        case 0:
                                if (toDisplay){
                                        clearDisplay();
                                        sendString_XY(0,0,"    SETTINGS");
                                        sendString_XY(1,0,"    4.RESET");
                                        toDisplay = false;
                                }

                                if (button == UP) {
                                        directory[0] = 3;
                                        toDisplay = true;
                                }
                                else if (button == SELECT) {
                                        directory[1]=1;
                                        toDisplay = true;
                                }
                                break;


                        case 1:
                                if (toDisplay){
                                        clearDisplay();
                                        sendString_XY(0,0,"CONFIRM RESET");
                                        toDisplay = false;
                                }

                                if (button == UP || button == DOWN) {
                                        reset = !reset;
                                        if (reset) sendString_XY(1,0,"Y");
                                        else sendString_XY(1,0,"n");
                                }

                                else if (button == SELECT) {
                                        if (reset){//
                                                rtc.setTime(0,0,0,1,1,4,21);
                                                reset = false;
                                                activeAlarm = -1;
                                                var=0;
                                                tone = 0;
                                                tempTone = tone;
                                                pos=0;
                                                mode=-1;
                                                alarmNo = -1;

                                                for (int x=0; x<5; x++){
                                                        alarms[x] = false;
                                                }
```

```
                                                        for (int row=0; row<5; row++){
                                                               for (int column = 0;
column<3; column++){

                alarmTime[row][column] = 0;
                                                               }
                                                        }
                                                        isAlarmActive = false;
                                                        directory[1] = 0;
                                                        toDisplay = true;
                                                        }
                                                }

                                        else if (button == DESELECT) {
                                                reset = false;
                                                directory[1] = 0;
                                                toDisplay = true;
                                        }

                                        break;
                                }
                        default:
                                break;




                }//directory_0_Switch

        }
}

void blinkDisplay(int position){
        if (position==0) sendString_XY(0,13,"_");
        else if (position==1) sendString_XY(0,14,"_");
        else if (position == 2) sendString_XY(0,9,"___");
        else if (position == 3) sendString_XY(0,9,"___");
        else if (position == 4) sendString_XY(0,2,"_");
        else if (position == 5) sendString_XY(0,3,"_");
        else if (position == 6) sendString_XY(0,5,"___");
        else if (position == 7) sendString_XY(1,4,"_");
        else if (position == 8) sendString_XY(1,5,"_");
        else if (position == 9) sendString_XY(1,7,"_");
        else if (position == 10) sendString_XY(1,8,"_");
        else if (position == 11) sendString_XY(1,10,"_");
        else if (position == 12) sendString_XY(1,11,"_");

        _delay_ms(100);
}
//Function to update time
void updateTime(int* tempSec, int* tempMin, int* tempHour, int* tempDay, int* tempWday,
int* tempMonth, int* tempYear, int *pos, int key){
        switch(*pos){

                case 0:
                        *tempYear = key*10+*tempYear%10;
```

```
                        _delay_ms(500);
                        break;

                case 1:
                        *tempYear = 10*(*tempYear/10)+key;
                        _delay_ms(500);
                        break;

                case 2:
                        if (key<2) *tempMonth = key*10+*tempMonth%10;
                        _delay_ms(500);
                        break;

                case 3:
                        *tempMonth = 10*(*tempMonth/10)+key-1;
                        if (*tempMonth<0) *tempMonth=0;
                        else if (*tempMonth>=12) *tempMonth = 11;
                        _delay_ms(500);
                        break;
                //Day calculation is need very hard to code this
                case 4:
                        if (*tempMonth!=1 && key<=3) *tempDay = key*10+*tempDay%10;
                        else if (*tempDay==1 && key<=2) *tempDay = key*10+*tempDay%10;
                        _delay_ms(500);
                        break;


                case 5:
                        *tempDay = 10*(*tempDay/10)+key;
                        if ((*tempMonth==0 || *tempMonth==2 || *tempMonth==4 ||
*tempMonth==6 || *tempMonth==7 || *tempMonth==9 || *tempMonth==11)&& *tempDay>31)
*tempDay =31;
                        else if ((*tempMonth==3 || *tempMonth==5 || *tempMonth==8 ||
*tempMonth==10) && *tempDay>30) *tempDay =30;
                        else if (*tempMonth==1 && *tempYear==0 && *tempDay>28) *tempDay =
28;
                        else if (*tempMonth==1 && *tempYear%4==0 && *tempDay>29) *tempDay
=29;
                        _delay_ms(500);
                        break;

                case 6:
                        if (key<=7 && key>=1) *tempWday = key-1;
                        _delay_ms(500);
                        break;

                case 7:
                        if (key<=2) *tempHour = key*10+*tempHour%10;
                        _delay_ms(500);
                        break;

                case 8:
                        if ((*tempHour>=20 && key<=3) ||(*tempHour<20)) *tempHour =
10*(*tempHour/10)+key;
                        _delay_ms(500);
                        break;

                case 9:
```

```
                        if (key<=5) *tempMin = 10*key + *tempMin%10;
                        _delay_ms(500);
                        break;

                case  10:
                        *tempMin = 10*(*tempMin/10)+key;
                        _delay_ms(500);
                        break;

                case 11:
                        if (key<=5) *tempSec = 10*key + *tempSec%10;
                        _delay_ms(500);
                        break;

                case 12:
                        *tempSec = 10*(*tempSec/10)+key;
                        _delay_ms(500);
                        break;

        }

                *pos = *pos+1;


}



void pla(){

var=1;}

void sto(){
        var = 0;
        isAlarmActive = false;
        alarms[activeAlarm] = false;
}

void del(int ms){
        while(ms--){_delay_ms(1);}}

        int Pirates()
        {
                const int totalNotes = sizeof(notes) / sizeof(int);
                // Loop through each note
                for (int i = 0; i < totalNotes; i++)
                {if (var==0){
                return 0;}


                const int currentNote = notes[i];
                int wait = durations[i] / songSpeed;
                // Play tone if currentNote is not 0 frequency, otherwise pause (noTone)
                if (currentNote !=0){
                play_tone( notes[i], wait/10); }

                // delay is used to wait for tone to finish playing before moving to next
loop
```

```
                else{del(wait);}

        }
}
int GameOfThrones(){

        const int totalNotes = sizeof(notes_G) / sizeof(int);
        // Loop through each note
        for (int i = 0; i < totalNotes; i++)
        {if (var==0){
        return 0;}

        const int currentNote = notes_G[i];
        int wait = duration_G[i] / songSpeed_G;
        // Play tone if currentNote is not 0 frequency, otherwise pause (noTone)
        if (currentNote != 0)
        {
                play_tone( notes_G[i], wait/10);// tone(pin, frequency, duration)
        }
        else
        {
                del(wait);
        }

        // delay is used to wait for tone to finish playing before moving to next loop

}


}

int despacito(){

        const int totalNotes = sizeof(melody) / sizeof(int);
        // Loop through each note
        for (int i = 0; i < totalNotes; i++)
        {if (var==0){
        return 0;}

        const int currentNote = melody[i];
        int wait = noteDurations[i] / songSpeed_G;
        // Play tone if currentNote is not 0 frequency, otherwise pause (noTone)
        if (currentNote != 0)
        {
                play_tone( melody[i], ((2000/wait)/10));// tone(pin, frequency, duration)
        }
        else
        {
                del(wait);
        }

        // delay is used to wait for tone to finish playing before moving to next loop

}


}
```

```
int starwars(){

	const int totalNotes = sizeof(StarWars) / sizeof(int)/2;
	// Loop through each note
	for (int i = 0; i < totalNotes; i++)
	{if (var==0){
	return 0;}

	const int currentNote = StarWars[i];
	int wait = abs(StarWars[i+1]) / songSpeed_G;
	// Play tone if currentNote is not 0 frequency, otherwise pause (noTone)
	if (currentNote != 0)
	{
		play_tone( StarWars[i], ((2000/wait)/10));// tone(pin, frequency, duration)
	}
	else
	{
		del(wait);
	}

	// delay is used to wait for tone to finish playing before moving to next loop

}


}


int godfather(){

	const int totalNotes = sizeof(GodFather) / sizeof(int)/2;
	// Loop through each note
	for (int i = 0; i < totalNotes; i++)
	{if (var==0){
	return 0;}

	const int currentNote = GodFather[i];
	int wait = abs(GodFather[i+1]) / songSpeed_G;
	// Play tone if currentNote is not 0 frequency, otherwise pause (noTone)
	if (currentNote != 0)
	{
		play_tone( GodFather[i], ((2000/wait)/10));// tone(pin, frequency,
duration)
	}
	else
	{
		del(wait);
	}

	// delay is used to wait for tone to finish playing before moving to next loop

}


}
void play_(){
```

```
        if(tone ==0){
                Pirates();
        }
        else{
                if(tone==1){
                        GameOfThrones();
                }
                else{
                        if(tone ==2){
                                despacito();
                        }
                        else{
                                if (tone==3){
                                        starwars();
                                }
                                else{
                                        if(tone==4){
                                                godfather();
                                        }
                                }
                        }
                }
        }

}
```