BUDDY : The Talking Alarm Clock

Electronics Design Workshop

ECECC09

RISHABH JAIN : 2020UEC2561

SAKSHAM KAUSHIK : 2020UEC2563

ANISH MITTAL : 2020UEC2570

Under Supervision of : **Prof. D.V. GADRE**

January, 2022 - April, 2022

INDEX

Project Documentation

Rishabh Jain, Saksham Kaushik, Anish Mittal

April 20, 2022

Abstract

We have learned many electronics concepts throughout the ECE Lab section and applied almost all of them directly or indirectly in our EDW lab project. Additionally, having taken away a lot from this course, we aimed to make a project that would challenge us academically and intellectually. This documentation contains a detailed explanation of our project "BUDDY - The Talking Alarm Clock", an alarm clock that tells the time verbally so that knowing time for the visually impaired is as easy as it is for us.

1 Introduction

1.1 Description

"BUDDY - The Talking Alarm Clock" as the name suggests, is an alarm clock that speaks out time loud for you whenever the "Tell Time" Button is pressed. Along with speaking out time, one can set an alarm at the desired time using a combination of buttons. According to the 6-block model, every electronic circuit consists of 6 electronic device types demonstrated in the image below.

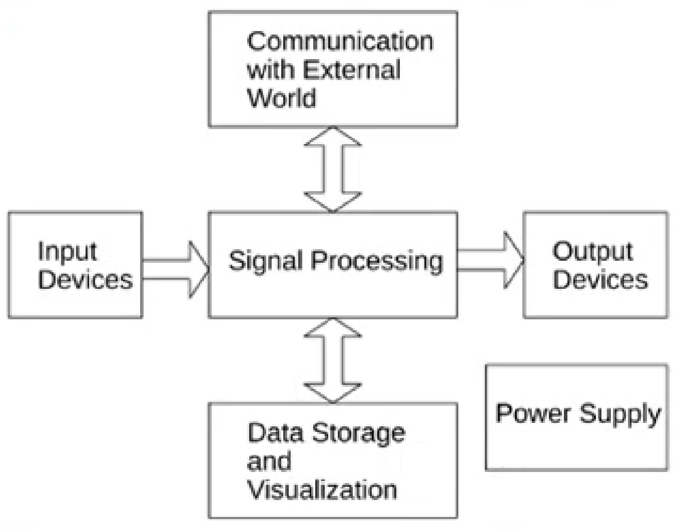

Communication with External World
Input Devices
Signal Processing
Output Devices
Data Storage and Visualization
Power Supply

Figure 1: 6 Block Model

For the signal processing unit, ESP32 is used which is connected to the Real Time Clock Module (RTC) which is the input block. LED Module and speaker ,are used to display and speak the time respectively, are the output block.

1.2 Motivation

The idea for this project was conceived during an illustrative class held by our respected Professor D.V. Gadre in which he demonstrated various projects developed by him and his students. The center of attraction for us was the projects developed to help the visually impaired students. Taking the inspiration from that, we decided to develop BUDDY The Talking Alarm Clock.

1.3 Objectives

We wanted to develop a project that makes the functions of a general alarm clock a seamless task for visually impaired people too. So, we included the following functionality in our clock:

1. **Tell Time** : BUDDY speaks out time for you.
2. **Set Alarm**: One can set an alarm at desired time of the day.

2 Components and Analysis

2.1 RTC Module

The DS3231 is a low-cost, extremely accurate I2C real-time clock (RTC) with an integrated Temperature Compensated Crystal Oscillator (TCXO) and crystal. The device incorporates a battery input and maintains accurate timekeeping when the main power to the device is interrupted.

This module is used to provide the clock signal required for us to display the time. We will be using a DS3231 real-time clock signal which will be connected to the ESP32.

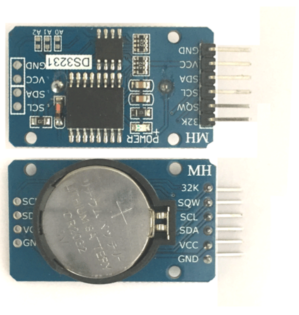


Figure 2: Image showing DS3231 real time clock

2.2 ESP32

ESP32 is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip . ESP32 has ample storage capacity with a flash memory of 4 MB. It is designed to achieve the best performance, showing robustness, versatility and reliability in a wide variety of applications and power scenarios.

The entire project centers around the ESP32 microcontroller. It is used here for less noise and better performance. We can easily connect and control electronic components for useful computing. We are using Arduino IDE to code the ESP32. Also, we are storing audio files in the flash memory.

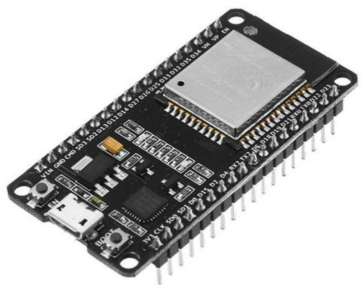


Figure 3: ESP32 microcontroller

2.3 Audio Board

We are using the PAM8403 IC, all parts of which are SMD components making it small in size and compatible. It is to amplify the audio signal given by the ESP32 DAC pin IO 25. So, the PAM8403 provides a very good sound and also minimizes part size.

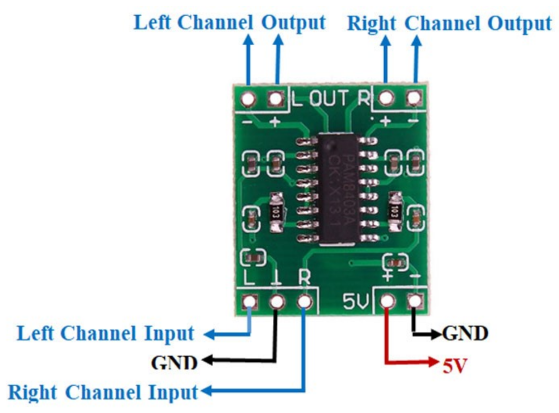


Figure 4: PAM8403 audio board

2.4 Speaker

We are using a 4ohm speaker which will play the music/audio file through audio board. The music/audio files are stored in the flash memory of ESP32.

Figure 5: Speaker

2.5 LED Display Module

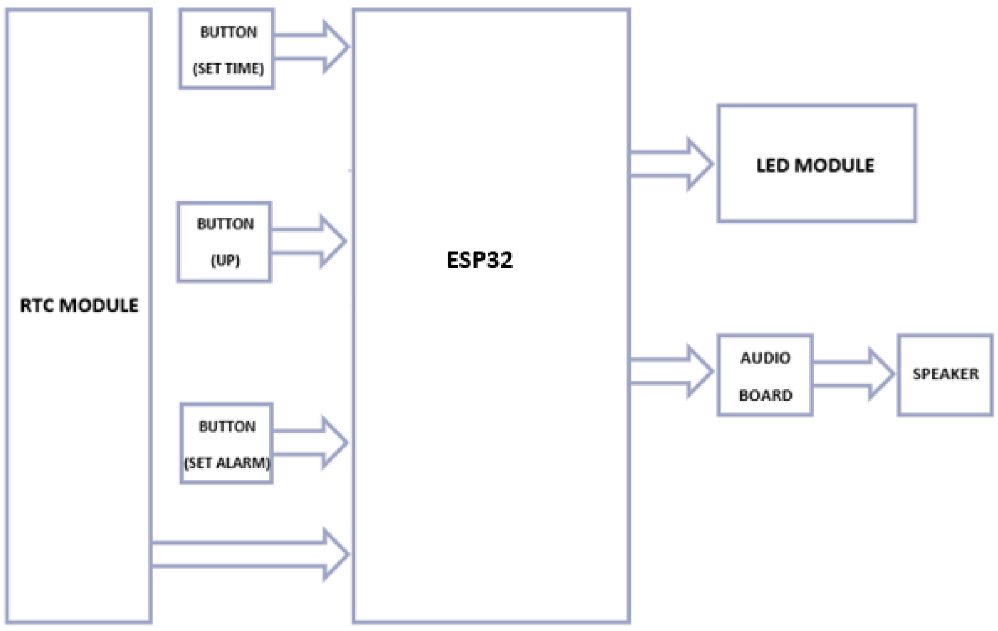
TM1637 DISPLAY module is used for displaying numbers. The module consists of four 7- segment displays working together.

Along with speaking the time, we are using a 4-digit led display to display the time and alarm information so that it can also work as a normal alarm clock.

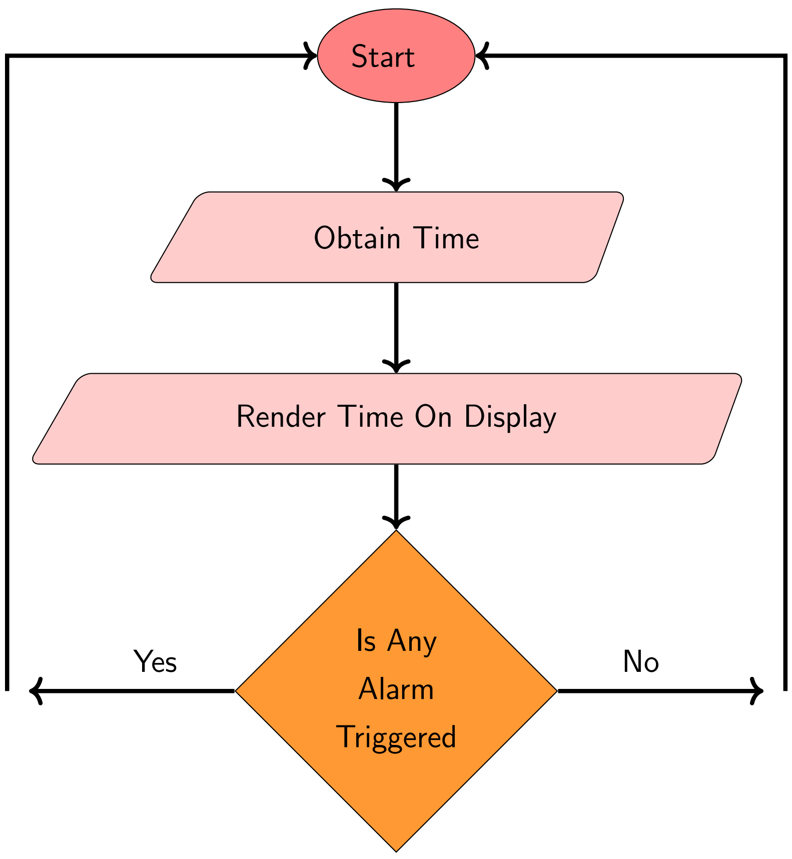
Figure 6: TM1637 LED Module

3 Working

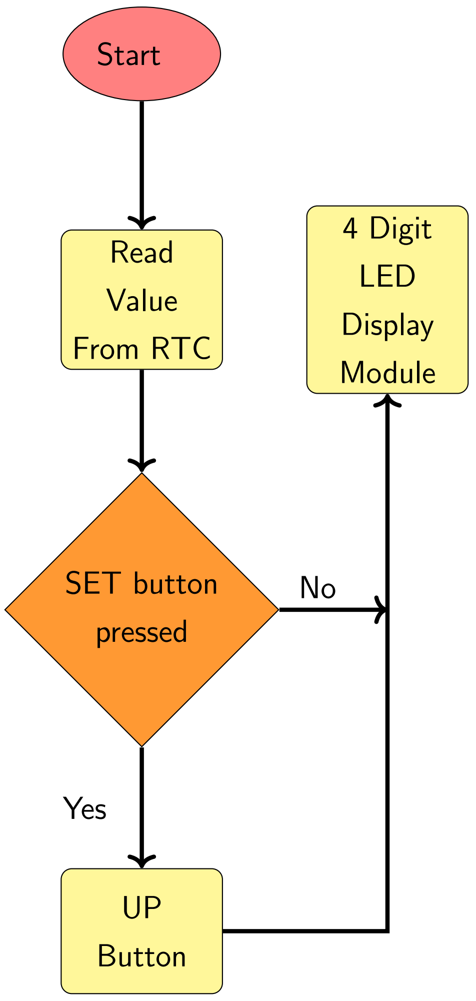
3.1 Block diagram



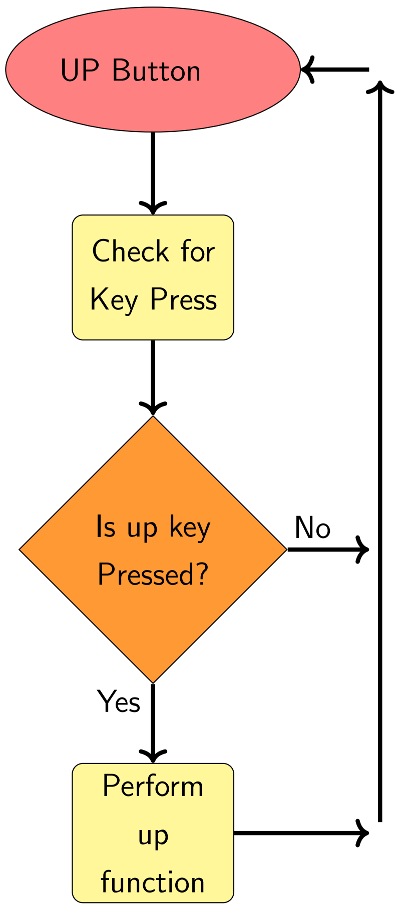
3.2 Flowcharts



Flowchart demonstration of Alarm Set



Flowchart demonstration of Alarm Set



Flowchart demonstration of UP operation

4 Designing

Moving to designing the circuit and PCB layout. We have used EAGLE software for designing the schematic and board layout file of the ciruit.

1. Hardware Design

A schematic was designed for PCB fabrication and a better explanation of connections. Following is a completely labelled circuit diagram for the project- BUDDY The Talking Alarm Clock.

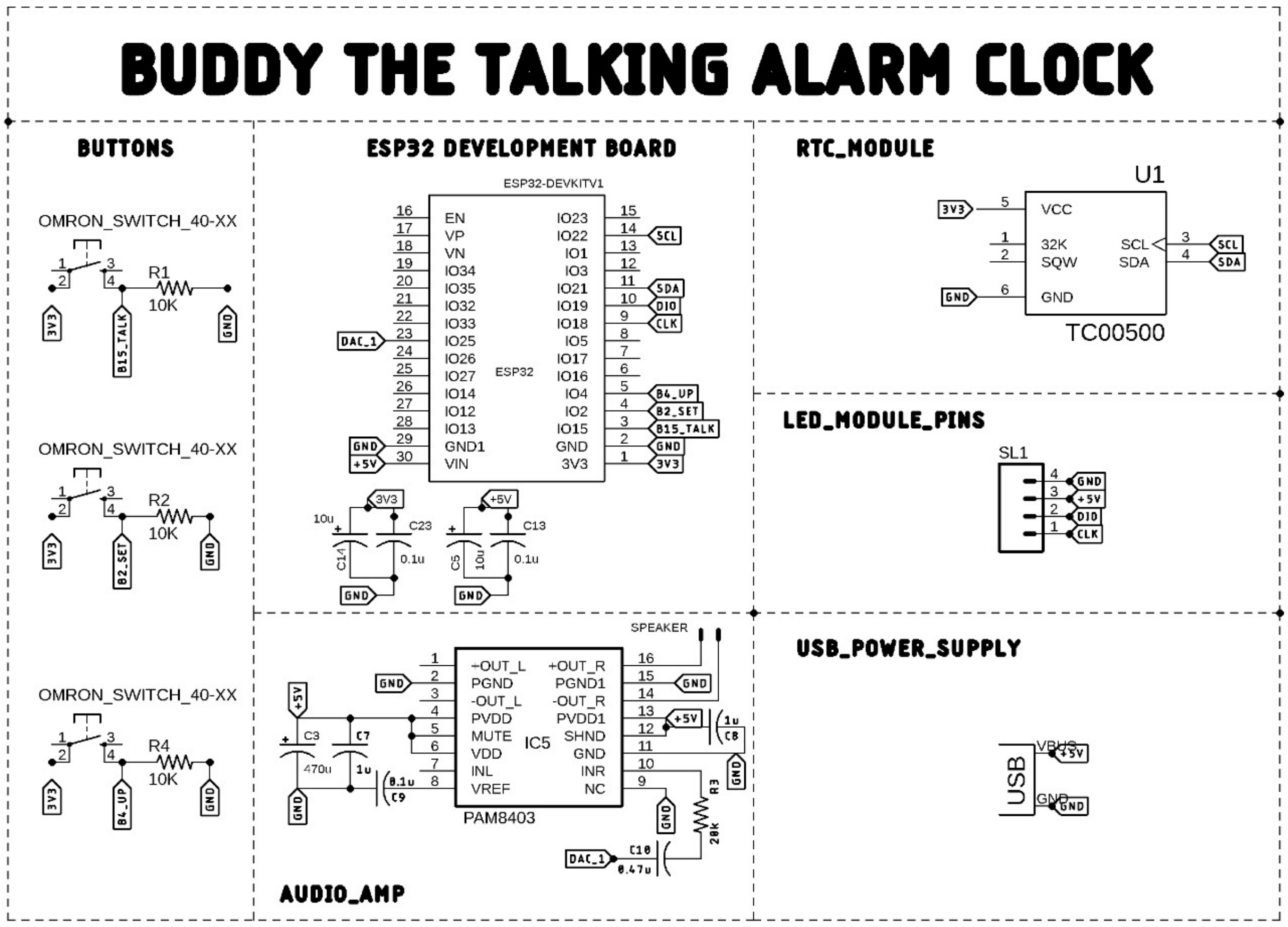


Figure 7: Schematic

As shown in the schematic above the PAM8403 is connected to DAC i.e. IO25 of ESP32.The Audio signal is being provided through this DAC IO25 only.The RTC module pins SDA is connected to IO21 and SCL is connected to IO22 of ESP32.The DIO and CLK of LED Module are connected to pin IO19 and IO18 of ESP32 respectively.The IO15 pin of ESP32 is connected to TELL TIME button and IO2 and IO4 pins of ESP32 are connected to SET and UP buttons respectively.All the Buttons and RTC are connected to 3V3 pin of ESP32 and ground and LED Module,Vin of ESP32 and PAM8403 are connected to +5V of USB mini and all GND pin are connected to ground of USB mini.

2. **Software Design**

Using Arduino Genuino IDE we have coded our ESP32 for different functionalities.

```
//BUDDY THE TALKING ALARM CLOCK
//Rishabh Jain 2020UEC2561
//Saksham Kaushik 2020UEC2563
//Anish Mittal 2020UEC2570
//BUDDY is an alarm clock that tells time when the
//TellTime Button is pressed.
//It uses RTC module DS3231 and TM1637 led module for display
//Alarm can be set using the SetButton and UpButton

//libraries
#include "SoundData.h"
#include "XT_DAC_Audio.h"
#include "MusicDefinitions.h"
#include <RTClib.h>
#include <SPI.h>
#include <Wire.h>
RTC_DS1307 rtc;



//classes
int8_t PROGMEM TwinkleTwinkle[] = {
  NOTE_C5,NOTE_C5,NOTE_G5,NOTE_G5,NOTE_A5,NOTE_A5,NOTE_G5,BEAT_2,
  NOTE_F5,NOTE_F5,NOTE_E5,NOTE_E5,NOTE_D5,NOTE_D5,NOTE_C5,BEAT_2,
  NOTE_G5,NOTE_G5,NOTE_F5,NOTE_F5,NOTE_E5,NOTE_E5,NOTE_D5,BEAT_2,
  NOTE_G5,NOTE_G5,NOTE_F5,NOTE_F5,NOTE_E5,NOTE_E5,NOTE_D5,BEAT_2,
  NOTE_C5,NOTE_C5,NOTE_G5,NOTE_G5,NOTE_A5,NOTE_A5,NOTE_G5,BEAT_2,
  NOTE_F5,NOTE_F5,NOTE_E5,NOTE_E5,NOTE_D5,NOTE_D5,NOTE_C5,BEAT_4,
  NOTE_SILENCE,BEAT_5,SCORE_END
};
XT_MusicScore_Class Music(TwinkleTwinkle,TEMPO_ALLEGRO,INSTRUMENT_PIANO);
// Create the main player class object.
//Use GPIO 25, one of the 2 DAC pins and timer 0
XT_DAC_Audio_Class DacAudio(25, 0);
XT_Wav_Class Am(AM);
XT_Wav_Class Pm(PM);
XT_Wav_Class TheTimeIs(THETIMEIS);
XT_Wav_Class TheAlarmIsSetTo(THEALARMISETO);
XT_Wav_Class Zero(ZERO);
XT_Wav_Class OClock(OCLOCK);
XT_Wav_Class One(ONE);
XT_Wav_Class Two(TWO);
XT_Wav_Class Three(THREE);
XT_Wav_Class Four(FOUR);
XT_Wav_Class Five(FIVE);
XT_Wav_Class Six(SIX);
XT_Wav_Class Seven(SEVEN);
XT_Wav_Class Eight(EIGHT);
XT_Wav_Class Nine(NINE);
XT_Wav_Class Ten(TEN);
XT_Wav_Class Eleven(ELEVEN);
XT_Wav_Class Twelve(TWELVE);
XT_Wav_Class Thirteen(THIRTEEN);
XT_Wav_Class Fourteen(FOURTEEN);
XT_Wav_Class Fifteen(FIFTEEN);
XT_Wav_Class Sixteen(SIXTEEN);
XT_Wav_Class Seventeen(SEVENTEEN);
XT_Wav_Class Eighteen(EIGHTEEN);
XT_Wav_Class Nineteen(NINTEEN);
XT_Wav_Class Twenty(TWENTY);
XT_Wav_Class Thirty(THIRTY);
XT_Wav_Class Forty(FOURTY);
XT_Wav_Class Fifty(FIFTY);
XT_Sequence_Class Sequence;
// The sequence object is one in which the audios are added
//sequentially in the manner they need to be played.

//pins and declaration
// the alarm is by default set to 6 AM
int alarmh=6;
int alarmm=00;
int alarm_am_pm = 0;
long x=millis();
const int clc = 18;  //ForTM1637 LED display(CLK)
const int data = 19; //ForTM1637 LED display(DIO)
uint8_t digits[] = { 0x3f, 0x06, 0x5b, 0x4f,
0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f };
int n=0; //for setting hours, minutes and am/pm

//program start
void setup() {
pinMode(clc, OUTPUT);
pinMode(data, OUTPUT);
pinMode(15,INPUT); // TellTimeButton
pinMode(2,INPUT); // SetButton
pinMode(4,INPUT ); //UpButton
//pinMode(26,OUTPUT); Initally used for buzzer for testing purpose
Serial.begin(9600);
//Initally were used to set the buzzer for testing purpose
//digitalWrite( 26,LOW );
//digitalWrite( 26,HIGH );

start();
```

```
writeValue(0x8c);
stop();
write(0x00, 0x00, 0x00, 0x00);

  //Serial.begin(9600);
 if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);
  }

  if (! rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
// used to adjust time of rtc with time of the computer
//    rtc.adjust(DateTime(F(DATE), F(TIME)));

  }
}

void loop() {

  DateTime now = rtc.now(); // used to get time from RTC(DS3231)
  int v=0;//AM PM
  int h=now.hour();
  int m=now.minute();
  int s=now.second();

  if (h>11){
    v=1;
  }
  if (h>12) {
   h=h-12;
  }

//Serial.print(h);
//Serial.print(":");
//Serial.print(m);
//Serial.print(":");
//Serial.print(s);
//Serial.println();

  int lz=(h / 10);
  //used to check if hours is double digit or not
  if (lz==0){
write(0x00, digits[h % 10] | ((s& 0x01) << 7), digits[m/ 10], digits[m % 10])
  }
  else {
write(digits[h / 10], digits[h % 10]
| ((s & 0x01) << 7), digits[m / 10], digits[m % 10]);
```

```
  }

  //used to play alarm on the set alarm time
  if (h==alarmh && m==alarmm && v==alarm_am_pm) {
     if(Music.Playing==false)    {     // if not playing
       DacAudio.Play(&Music);
     }
  }
  DacAudio.FillBuffer();

  if(digitalRead(15)==HIGH){ // TellTime Button
  // Clear out any previous playlist
    Sequence.RemoveAllPlayItems();
    delay(200);
    while(digitalRead(15));
    delay(200);
    is_Pressed(15,h,m,v);
    DacAudio.Play(&Sequence);
    Serial.println(digitalRead(15));
  }

  if(digitalRead(2)==HIGH){ //SetButton
    delay(200);
    while(digitalRead(2));
    delay(200);
    n=(n+1)%3;
    Serial.println(n);
  }

  if(digitalRead(4)==HIGH){ // Up Button
    Sequence.RemoveAllPlayItems();
    delay(200);
    while(digitalRead(4));
    delay(200);
    setAlarm(4);
    DacAudio.Play(&Sequence);
//      Serial.print(alarmh);  used for testing purpose
//      Serial.print(':');
//      Serial.print(alarmm);
//      Serial.print(':');
//      Serial.print(alarm_am_pm);
//      Serial.println();
  }

  while(millis()-x<500){
  // colon will be off for half a second
    DacAudio.FillBuffer();
    if (lz==0){
      write(0x00, digits[h % 10]
      | ((0x00) << 7), digits[m/ 10], digits[m % 10]);
    }
    else {
        write(digits[h / 10], digits[h % 10]
        | (( 0x00) << 7), digits[m / 10], digits[m % 10]);
    }
  }

  while(millis()-x<500){
  // colon will be on for half a second
    DacAudio.FillBuffer();
    if (lz==0){
      write(0x00, digits[h % 10]
      | ((0x01) << 7), digits[m/ 10], digits[m % 10]);
    }
    else {
        write(digits[h / 10], digits[h % 10]
        | (( 0x01) << 7), digits[m / 10], digits[m % 10]);
    }
  }
}



//functions

void setAlarm(int pinUp){
// used for incrementing the hour/minute/am_pm when
//the Up Button is pressed.
PinUp is the pin for the Up Button.
  if(n==0){
      ++alarmh;
      if(alarmh>12){
        alarmh=1;
      }
    }

    else if(n==1){
       ++alarmm;
       if(alarmm>=60){
        alarmm=0;
        }
    }

      else if(n==2){
        alarm_am_pm=(alarm_am_pm+1)%2;
      }
      is_Alarm_Pressed(pinUp, alarmh, alarmm, alarm_am_pm);
 }

void is_Pressed(int pinNumber, int h,int m,int v)
{ //Creates the sequence when the Tell Time Button
  //is pressed using if-else conditions

    Sequence.AddPlayItem(&TheTimeIs);
     if (h==1){
        Sequence.AddPlayItem(&One);
     }
     else if (h==2){
      Sequence.AddPlayItem(&Two);
     }
     else if (h==3){
      Sequence.AddPlayItem(&Three);
     }
     else if (h==4){
      Sequence.AddPlayItem(&Four);
     }
     else if (h==5){
      Sequence.AddPlayItem(&Five);
     }
     else if (h==6){
      Sequence.AddPlayItem(&Six);
     }
     else if (h==7){
      Sequence.AddPlayItem(&Seven);
     }
     else if (h==8){
      Sequence.AddPlayItem(&Eight);
     }
     else if (h==9){
       Sequence.AddPlayItem(&Nine);
     }
     else if (h==10){
      Sequence.AddPlayItem(&Ten);
     }
     else if (h==11){
       Sequence.AddPlayItem(&Eleven);
     }
     else if (h==12){
       Sequence.AddPlayItem(&Twelve);
     }

    if (m==0){
       Sequence.AddPlayItem(&OClock);
    }
```

```
else if (m==1){
 Sequence.AddPlayItem(&Zero);
  Sequence.AddPlayItem(&One);
}
else if (m==2){
  Sequence.AddPlayItem(&Zero);
  Sequence.AddPlayItem(&Two);
}
else if (m==3){
  Sequence.AddPlayItem(&Zero);
  Sequence.AddPlayItem(&Three);
}
else if (m==4){
  Sequence.AddPlayItem(&Zero);
  Sequence.AddPlayItem(&Four);
}
else if (m==5){
  Sequence.AddPlayItem(&Zero);
  Sequence.AddPlayItem(&Five);
}
else if (m==6){
  Sequence.AddPlayItem(&Zero);
  Sequence.AddPlayItem(&Six);
}
else if (m==7){
  Sequence.AddPlayItem(&Zero);
  Sequence.AddPlayItem(&Seven);
}
else if (m==8){
  Sequence.AddPlayItem(&Zero);
  Sequence.AddPlayItem(&Eight);
}
else if (m==9){
  Sequence.AddPlayItem(&Zero);
  Sequence.AddPlayItem(&Nine);
}
else if (m==10){
 Sequence.AddPlayItem(&Ten);
}
else if (m==11){
 Sequence.AddPlayItem(&Eleven);
}
else if (m==12){
 Sequence.AddPlayItem(&Twelve);
}
else if (m==13){
 Sequence.AddPlayItem(&Thirteen);
}
```

```
else if (m==14){
 Sequence.AddPlayItem(&Fourteen);
}
else if (m==15){
 Sequence.AddPlayItem(&Fifteen);
}
else if (m==16){
 Sequence.AddPlayItem(&Sixteen);
}
else if (m==17){
 Sequence.AddPlayItem(&Seventeen);
}
else if (m==18){
 Sequence.AddPlayItem(&Eighteen);
}
else if (m==19){
 Sequence.AddPlayItem(&Nineteen);
}
else if (m==20){
 Sequence.AddPlayItem(&Twenty);
}
else if (m==21){
     Sequence.AddPlayItem(&Twenty);
     Sequence.AddPlayItem(&One);
}
else if (m==22){
    Sequence.AddPlayItem(&Twenty);
     Sequence.AddPlayItem(&Two);
}
else if (m==23){
    Sequence.AddPlayItem(&Twenty);
     Sequence.AddPlayItem(&Three);
}
else if (m==24){
    Sequence.AddPlayItem(&Twenty);
    Sequence.AddPlayItem(&Four);
}
else if (m==25){
    Sequence.AddPlayItem(&Twenty);
     Sequence.AddPlayItem(&Five);
}
else if (m==26){
    Sequence.AddPlayItem(&Twenty);
     Sequence.AddPlayItem(&Six);
}
else if (m==27){
    Sequence.AddPlayItem(&Twenty);
     Sequence.AddPlayItem(&Seven);
}
else if (m==28){
    Sequence.AddPlayItem(&Twenty);
      Sequence.AddPlayItem(&Eight);
}
else if (m==29){
    Sequence.AddPlayItem(&Twenty);
    Sequence.AddPlayItem(&Nine);
}
else if (m==30){
  Sequence.AddPlayItem(&Thirty);
}
else if (m==31){
    Sequence.AddPlayItem(&Thirty);
    Sequence.AddPlayItem(&One);
}
else if (m==32){
    Sequence.AddPlayItem(&Thirty);
    Sequence.AddPlayItem(&Two);
}
else if (m==33){
    Sequence.AddPlayItem(&Thirty);
    Sequence.AddPlayItem(&Three);
}
else if (m==34){
    Sequence.AddPlayItem(&Thirty);
    Sequence.AddPlayItem(&Four);
}
else if (m==35){
    Sequence.AddPlayItem(&Thirty);
    Sequence.AddPlayItem(&Five);
}
else if (m==36){
    Sequence.AddPlayItem(&Thirty);
    Sequence.AddPlayItem(&Six);
}
else if (m==37){
    Sequence.AddPlayItem(&Thirty);
    Sequence.AddPlayItem(&Seven);
}
else if (m==38){
    Sequence.AddPlayItem(&Thirty);
    Sequence.AddPlayItem(&Eight);
}
else if (m==39){
    Sequence.AddPlayItem(&Thirty);
    Sequence.AddPlayItem(&Nine);
}
```

```
else if (m==40){
    Sequence.AddPlayItem(&Forty);
}
else if (m==41){
    Sequence.AddPlayItem(&Forty);
    Sequence.AddPlayItem(&One);
}
else if (m==42){
    Sequence.AddPlayItem(&Forty);
    Sequence.AddPlayItem(&Two);
}
else if (m==43){
    Sequence.AddPlayItem(&Forty);
    Sequence.AddPlayItem(&Three);
}
else if (m==44){
    Sequence.AddPlayItem(&Forty);
    Sequence.AddPlayItem(&Four);
}
else if (m==45){
    Sequence.AddPlayItem(&Forty);
    Sequence.AddPlayItem(&Five);
}
else if (m==46){
    Sequence.AddPlayItem(&Forty);
    Sequence.AddPlayItem(&Six);
}
else if (m==47){
    Sequence.AddPlayItem(&Forty);
    Sequence.AddPlayItem(&Seven);
}
else if (m==48){
    Sequence.AddPlayItem(&Forty);
    Sequence.AddPlayItem(&Eight);
}
else if (m==49){
  Sequence.AddPlayItem(&Forty);
  Sequence.AddPlayItem(&Nine);
}
else if (m==50){
  Sequence.AddPlayItem(&Fifty);
}
else if (m==51){
  Sequence.AddPlayItem(&Fifty);
  Sequence.AddPlayItem(&One);
}
else if (m==52){
  Sequence.AddPlayItem(&Fifty);
```

```
     Sequence.AddPlayItem(&Two);
    }
    else if (m==53){
     Sequence.AddPlayItem(&Fifty);
     Sequence.AddPlayItem(&Three);
    }
    else if (m==54){
     Sequence.AddPlayItem(&Fifty);
     Sequence.AddPlayItem(&Four);
    }
    else if (m==55){
       Sequence.AddPlayItem(&Fifty);
     Sequence.AddPlayItem(&Five);
    }
    else if (m==56){
       Sequence.AddPlayItem(&Fifty);
     Sequence.AddPlayItem(&Six);
    }
    else if (m==57){
       Sequence.AddPlayItem(&Fifty);
     Sequence.AddPlayItem(&Seven);
    }
    else if (m==58){
       Sequence.AddPlayItem(&Fifty);
     Sequence.AddPlayItem(&Eight);
    }
    else if (m==59){
      Sequence.AddPlayItem(&Fifty);
      Sequence.AddPlayItem(&Nine);
    }
    if (v==0){
      Sequence.AddPlayItem(&Am);
    }
    else if (v==1){
      Sequence.AddPlayItem(&Pm);
    }
}

void is_Alarm_Pressed(int pinNumber, int h,int m,int v)
{ // used to create sequence when the Set Button is pressed

    Sequence.AddPlayItem(&TheAlarmIsSetTo);
     if (h==1){
        Sequence.AddPlayItem(&One);
     }
     else if (h==2){
      Sequence.AddPlayItem(&Two);
     }
     else if (h==3){
      Sequence.AddPlayItem(&Three);
     }
     else if (h==4){
      Sequence.AddPlayItem(&Four);
     }
     else if (h==5){
      Sequence.AddPlayItem(&Five);
     }
     else if (h==6){
      Sequence.AddPlayItem(&Six);
     }
     else if (h==7){
      Sequence.AddPlayItem(&Seven);
     }
     else if (h==8){
      Sequence.AddPlayItem(&Eight);
     }
     else if (h==9){
       Sequence.AddPlayItem(&Nine);
     }
     else if (h==10){
      Sequence.AddPlayItem(&Ten);
     }
     else if (h==11){
       Sequence.AddPlayItem(&Eleven);
     }
     else if (h==12){
       Sequence.AddPlayItem(&Twelve);
     }

    if (m==0){
       Sequence.AddPlayItem(&OClock);
    }
    else if (m==1){
     Sequence.AddPlayItem(&Zero);
      Sequence.AddPlayItem(&One);
    }
    else if (m==2){
      Sequence.AddPlayItem(&Zero);
      Sequence.AddPlayItem(&Two);
    }
    else if (m==3){
      Sequence.AddPlayItem(&Zero);
      Sequence.AddPlayItem(&Three);
    }
    else if (m==4){
      Sequence.AddPlayItem(&Zero);
      Sequence.AddPlayItem(&Four);
    }
    else if (m==5){
      Sequence.AddPlayItem(&Zero);
      Sequence.AddPlayItem(&Five);
    }
    else if (m==6){
      Sequence.AddPlayItem(&Zero);
      Sequence.AddPlayItem(&Six);
    }
    else if (m==7){
      Sequence.AddPlayItem(&Zero);
      Sequence.AddPlayItem(&Seven);
    }
    else if (m==8){
      Sequence.AddPlayItem(&Zero);
      Sequence.AddPlayItem(&Eight);
    }
    else if (m==9){
      Sequence.AddPlayItem(&Zero);
      Sequence.AddPlayItem(&Nine);
    }
    else if (m==10){
     Sequence.AddPlayItem(&Ten);
    }
    else if (m==11){
     Sequence.AddPlayItem(&Eleven);
    }
    else if (m==12){
     Sequence.AddPlayItem(&Twelve);
    }
    else if (m==13){
     Sequence.AddPlayItem(&Thirteen);
    }
    else if (m==14){
     Sequence.AddPlayItem(&Fourteen);
    }
    else if (m==15){
     Sequence.AddPlayItem(&Fifteen);
    }
    else if (m==16){
     Sequence.AddPlayItem(&Sixteen);
    }
    else if (m==17){
     Sequence.AddPlayItem(&Seventeen);
    }
    else if (m==18){
     Sequence.AddPlayItem(&Eighteen);
```

```
}
else if (m==19){
 Sequence.AddPlayItem(&Nineteen);
}
else if (m==20){
 Sequence.AddPlayItem(&Twenty);
}
else if (m==21){
    Sequence.AddPlayItem(&Twenty);
    Sequence.AddPlayItem(&One);
}
else if (m==22){
   Sequence.AddPlayItem(&Twenty);
    Sequence.AddPlayItem(&Two);
}
else if (m==23){
   Sequence.AddPlayItem(&Twenty);
    Sequence.AddPlayItem(&Three);
}
else if (m==24){
   Sequence.AddPlayItem(&Twenty);
   Sequence.AddPlayItem(&Four);
}
else if (m==25){
   Sequence.AddPlayItem(&Twenty);
    Sequence.AddPlayItem(&Five);
}
else if (m==26){
   Sequence.AddPlayItem(&Twenty);
    Sequence.AddPlayItem(&Six);
}
else if (m==27){
   Sequence.AddPlayItem(&Twenty);
    Sequence.AddPlayItem(&Seven);
}
else if (m==28){
   Sequence.AddPlayItem(&Twenty);
    Sequence.AddPlayItem(&Eight);
}
else if (m==29){
   Sequence.AddPlayItem(&Twenty);
   Sequence.AddPlayItem(&Nine);
}
else if (m==30){
 Sequence.AddPlayItem(&Thirty);
}
else if (m==31){
   Sequence.AddPlayItem(&Thirty);
```

```
        Sequence.AddPlayItem(&One);
    }
    else if (m==32){
        Sequence.AddPlayItem(&Thirty);
        Sequence.AddPlayItem(&Two);
    }
    else if (m==33){
        Sequence.AddPlayItem(&Thirty);
        Sequence.AddPlayItem(&Three);
    }
    else if (m==34){
        Sequence.AddPlayItem(&Thirty);
        Sequence.AddPlayItem(&Four);
    }
    else if (m==35){
        Sequence.AddPlayItem(&Thirty);
        Sequence.AddPlayItem(&Five);
    }
    else if (m==36){
        Sequence.AddPlayItem(&Thirty);
        Sequence.AddPlayItem(&Six);
    }
    else if (m==37){
        Sequence.AddPlayItem(&Thirty);
        Sequence.AddPlayItem(&Seven);
    }
    else if (m==38){
        Sequence.AddPlayItem(&Thirty);
        Sequence.AddPlayItem(&Eight);
    }
    else if (m==39){
        Sequence.AddPlayItem(&Thirty);
        Sequence.AddPlayItem(&Nine);
    }
    else if (m==40){
        Sequence.AddPlayItem(&Forty);
    }
    else if (m==41){
        Sequence.AddPlayItem(&Forty);
        Sequence.AddPlayItem(&One);
    }
    else if (m==42){
        Sequence.AddPlayItem(&Forty);
        Sequence.AddPlayItem(&Two);
    }
    else if (m==43){
        Sequence.AddPlayItem(&Forty);
        Sequence.AddPlayItem(&Three);
```

```
}
else if (m==44){
   Sequence.AddPlayItem(&Forty);
   Sequence.AddPlayItem(&Four);
}
else if (m==45){
   Sequence.AddPlayItem(&Forty);
   Sequence.AddPlayItem(&Five);
}
else if (m==46){
   Sequence.AddPlayItem(&Forty);
   Sequence.AddPlayItem(&Six);
}
else if (m==47){
   Sequence.AddPlayItem(&Forty);
   Sequence.AddPlayItem(&Seven);
}
else if (m==48){
   Sequence.AddPlayItem(&Forty);
   Sequence.AddPlayItem(&Eight);
}
else if (m==49){
 Sequence.AddPlayItem(&Forty);
 Sequence.AddPlayItem(&Nine);
}
else if (m==50){
 Sequence.AddPlayItem(&Fifty);
}
else if (m==51){
 Sequence.AddPlayItem(&Fifty);
 Sequence.AddPlayItem(&One);
}
else if (m==52){
 Sequence.AddPlayItem(&Fifty);
 Sequence.AddPlayItem(&Two);
}
else if (m==53){
 Sequence.AddPlayItem(&Fifty);
 Sequence.AddPlayItem(&Three);
}
else if (m==54){
 Sequence.AddPlayItem(&Fifty);
 Sequence.AddPlayItem(&Four);
}
else if (m==55){
   Sequence.AddPlayItem(&Fifty);
 Sequence.AddPlayItem(&Five);
}
```

```
    else if (m==56){
       Sequence.AddPlayItem(&Fifty);
     Sequence.AddPlayItem(&Six);
    }
    else if (m==57){
       Sequence.AddPlayItem(&Fifty);
     Sequence.AddPlayItem(&Seven);
    }
    else if (m==58){
       Sequence.AddPlayItem(&Fifty);
     Sequence.AddPlayItem(&Eight);
    }
    else if (m==59){
      Sequence.AddPlayItem(&Fifty);
      Sequence.AddPlayItem(&Nine);
    }
    if (v==0){
      Sequence.AddPlayItem(&Am);
    }
    else if (v==1){
      Sequence.AddPlayItem(&Pm);
    }
}

// The below functions are used to display the
time on the TM1637 Module
void write(uint8_t first, uint8_t second,
uint8_t third, uint8_t fourth){
  start();
  writeValue(0x40);
  stop();
  start();
  writeValue(0xc0);

     writeValue(first);

  //writeValue(first);
  writeValue(second);
  writeValue(third);
  writeValue(fourth);
  stop();
}

void start(void){
  digitalWrite(clc,HIGH);
  digitalWrite(data,HIGH);
  delayMicroseconds(5);
  digitalWrite(data,LOW);
  digitalWrite(clc,LOW);
  delayMicroseconds(5);
}

void stop(void){
  digitalWrite(clc,LOW);
  digitalWrite(data,LOW);
  delayMicroseconds(5);

  digitalWrite(clc,HIGH);
  digitalWrite(data,HIGH);
  delayMicroseconds(5);
}

bool writeValue(uint8_t value){
  for(uint8_t i = 0; i < 8; i++)
  {
    digitalWrite(clc, LOW);
    delayMicroseconds(5);
   // Serial.print(data, (value & (1 << i)) >> i);
    digitalWrite(data, (value & (1 << i)) >> i);
    delayMicroseconds(5);
    digitalWrite(clc, HIGH);
    delayMicroseconds(5);
  }
  digitalWrite(clc,LOW);
  delayMicroseconds(5);
  pinMode(data,INPUT);
  digitalWrite(clc,HIGH);
  delayMicroseconds(5);
  bool ack = digitalRead(data) == 0;
  pinMode(data,OUTPUT);

  return ack;
}
```

5 Setup and Fabrication

5.1 Board Layout Design

Autodesk EAGLE software has been used to design PCB for the project. Board layout for the same is as follows:

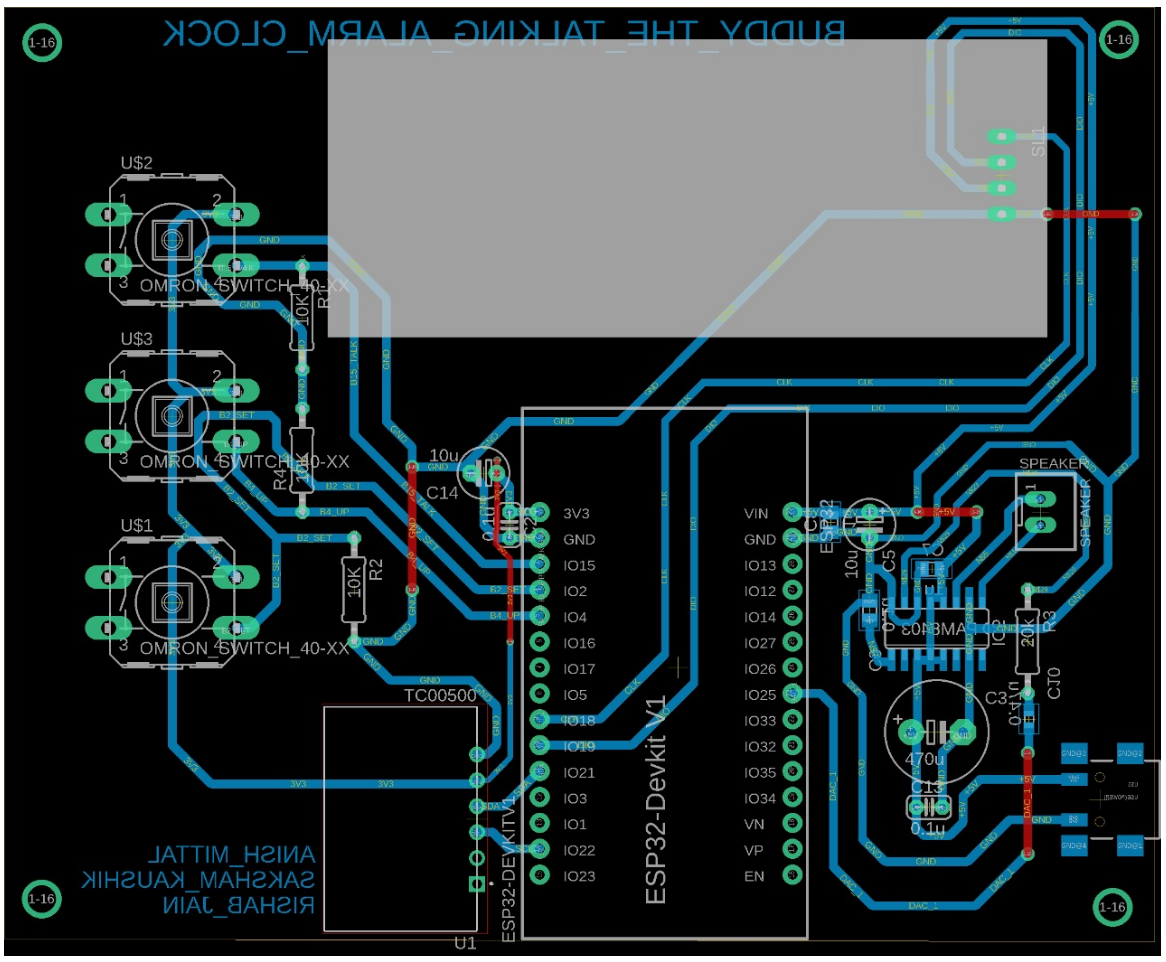


Figure 8: Board Layout for PCB Design

5.2 PCB Fabrication :

Follow the below steps for PCB fabrication:

1. Copper Clad Cutting Copper Clad of desired size, as made in software design , was cut using a CNC(Computer Numeric Control) machine.

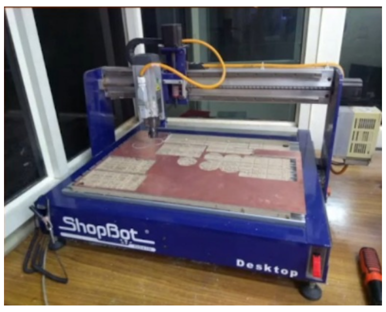


Figure 9: Copper Clad cutting

2. Sanding : The copper clad cut so far may have attracted fingerprints and scratches which can cause problems in PCB fabrication. Hence, sanding is done on copper clad .

Figure 10: Sanding

3. Printing and Pressing Board layout on Copper : The Board layout made above is printed and using iron press , it is aligned and pasted on the PCB. Then , after waiting for it to cool down , we use water to remove excess paper off the copper.

Figure 11: Pressing

4. Etching : FeCl3 solution is used for etching, This solution dissolves all the copper on the clad but the connections as the connections are covered with a layer of ink.

Figure 12: Etching

5. Scrubbing : Then the layer on connections is removed by scrubbing.

6. Conformal Coating Spray : This is coated with conformal coating spray which provides protection from moisture, shock etc to the PCB.

7. Holes Drilling : Now holes are drilled for the stands and at the specified postions where through-hole components are needed to be placed .

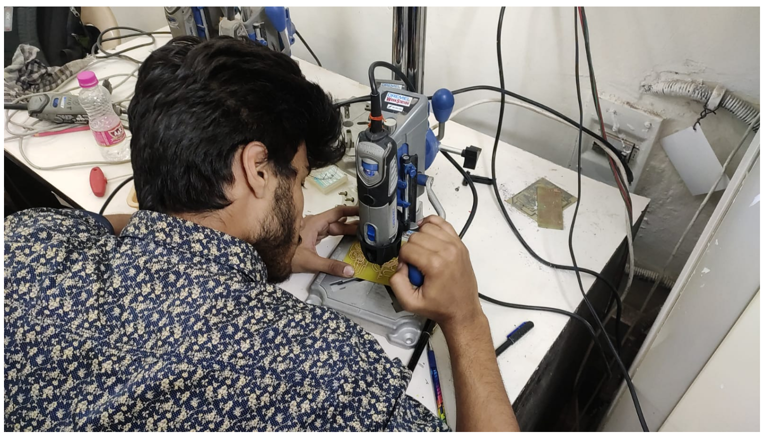


Figure 13: Drilling holes

5.3 Connecting the Components

All the components are soldered at their specific positions after programming ESP32 with the above code.

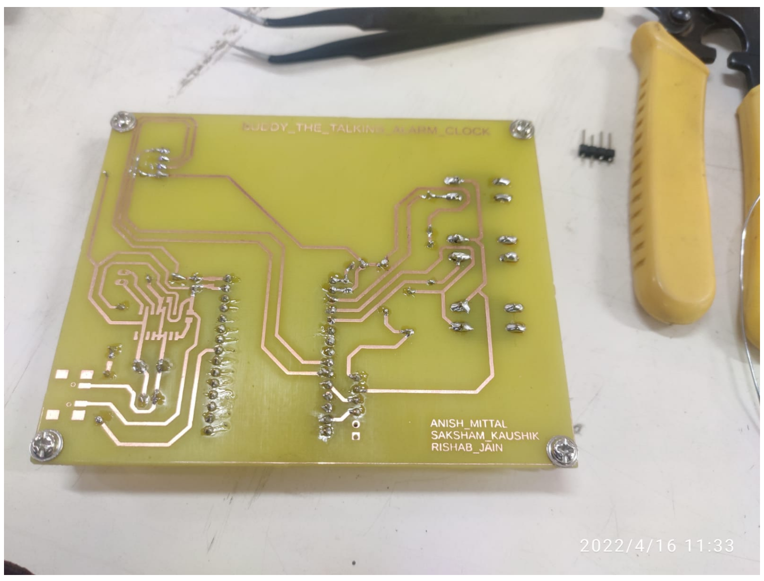


Figure 14: Soldering

5.4 Outer Case

We have used acrylic to make the outer case for this clock.First we designed cutouts for speaker, Display and buttons by using PartWorks software and then used CNC machine to cut it into desired size .

Then this is screwed on both sides which acts as a case for our setup.

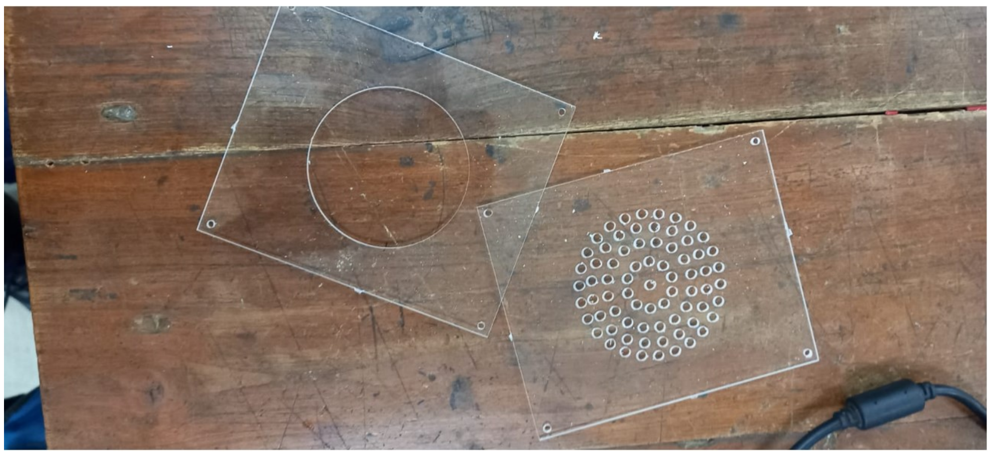


Figure 15: Acrylic Cutting

6 Conclusion

The fully Functioning Model of our Project BUDDY The Talking Alarm Clock was obtained with all the functions working properly including Tell Time and Set Alarm. A glimpse of final prototype is shown below.