

Digital Alarm Clock

1) Introduction

The Arduino code presented here demonstrates the creation of a digital clock with an alarm function. This project combines both hardware components and programming logic to achieve its goal. By utilizing an Arduino board, a Real-Time Clock (RTC) module, a buzzer, potentiometer and an LED, a versatile timekeeping device with an added wake-up alarm feature is created.

To facilitate communication between the Arduino board and the connected peripherals, the code incorporates libraries such as Wire, LiquidCrystal, DS1307RTC, and TimeLib. To transfer the date and time information on the computer to the DS1307RTC module, we ran the code opened by clicking files -> examples -> DS1307RTC -> set time from the Arduino ide, in the circuit consisting only of the Arduino ide and our RTC module. In this way, the information was transferred successfully. Our RTC module has become suitable for use in our project. The LiquidCrystal library specifically enables control of a standard 16x2 LCD display, providing a clear and user-friendly interface for setting the time and alarm.

Upon starting the project, the LCD screen displays the words "Digital Clock" and "0420 Embedded". The reset button allows the user to turn off the alarm and return to the main screen. Pressing the Set button once brings up the time setting screen, where the hour and minute can be adjusted using the other two buttons. Pressing the Set button again sets the alarm.

One notable feature of this project is the real-time display of the current time and date. This is made possible through the DS1307RTC library, which communicates with the RTC module. The LCD screen continuously updates to provide accurate time and date information.

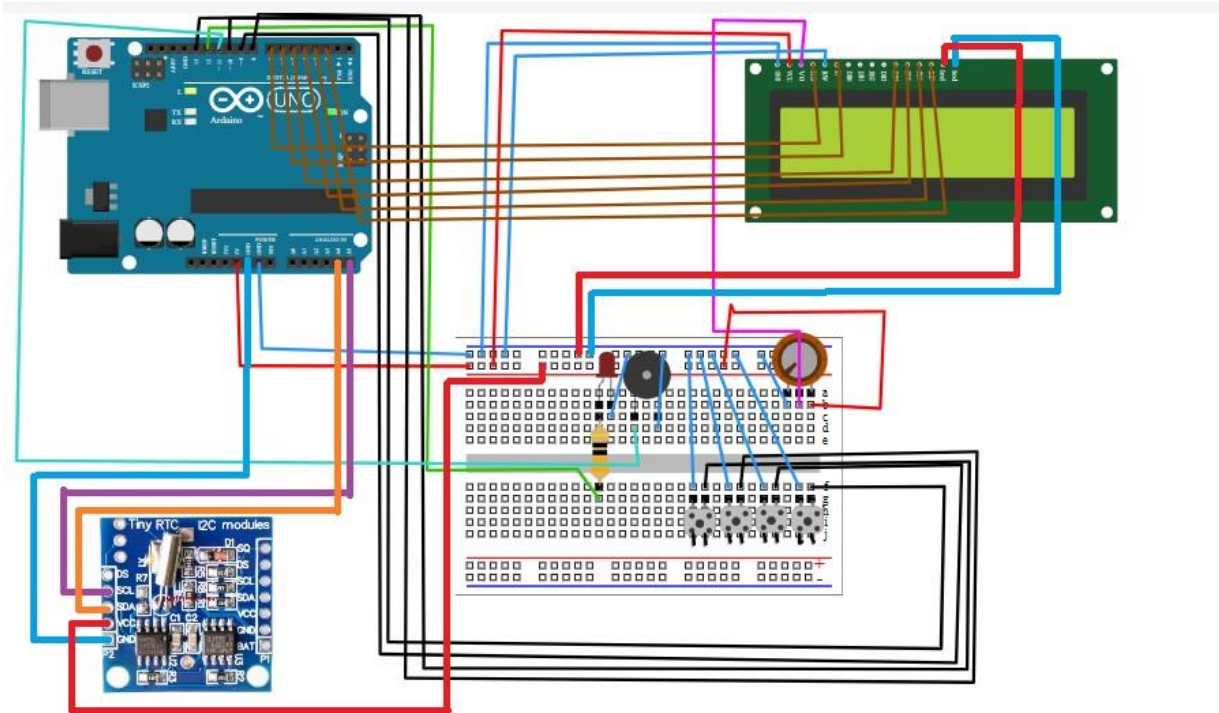
When the alarm is set, the Arduino board constantly monitors the current time. Once the specified alarm time is reached, a wake-up message is displayed on the LCD screen, and both a buzzer and a red LED are activated.

This project not only demonstrates the practical application of Arduino programming but also showcases the integration of various components to achieve a specific goal. Whether used as a bedside clock or a reliable wake-up companion, it highlights the versatility and creativity that can be achieved with Arduino-based solutions.

2) Components

Arduino UNO	1
2*16 LCD Screen	1
Potentiometer	1
DS1307RTC	1
Buzzer	1
LED	1
Button	4
Resistor	1
M-F Jumper Cable	2
M-M Jumper Cable	33

3) Design



LCD

LCD's VSS and RW are connected to the ground.

VDD is connected to the 5 Volt.

VEE is connected to the potentiometer (middle pin).

RS is connected to the Arduino's 7. Pin

E is connected to the Arduino's 6. Pin

D4 is connected to the Arduino's 5. Pin

D5 is connected to the Arduino's 4. Pin

D6 is connected to the Arduino's 3. Pin

D7 is connected to the Arduino's 2. Pin

BLK is connected to the ground

BLA is connected to the 5 Volt

BUTTONS

First button's left side (set alarm) is connected to the Arduino's 10. Pin and the right side is connected to the ground.

Second button's left side (set hour) is connected to the Arduino's 9. Pin and the right side is connected to the ground.

Third button's left side (set minutes) is connected to the Arduino's 8. Pin and the right side is connected to the ground.

Last button's left side (reset alarm) is connected to the Arduino's 7. Pin and the right side is connected to the ground.

RED LED

Red led's long leg is connected to the resistor and the resistor is connected to the Arduino's 12. Pin and the other leg of red led is connected to the ground.

BUZZER

Buzzer's long leg is connected Arduino's 11. Pin and the short leg is connected to the ground.

POTENTIOMETER

Potentiometer's middle pin is connected to the LCD Screen (VEE). Left pin is connected to the ground and right pin is connected to the 5 Volt.

DS1307RTC

DS1307RTC's SDA pin is connected to the Arduino's Analog Pin (A4) and SCL pin is connected to the Arduino's Analog Pin (A5) and VCC pin in connected to the 5 Volt and lastly GND pin is connected to the ground.

4) CODE

For set time and hour in DS1307RTC, we use files->examples->DS1307RTC->set time code :

```
#include <Wire.h>
```

```
#include <TimeLib.h>
```

```
#include <DS1307RTC.h>
```

```
const char monthName[12] = {
```

```
  "Jan", "Feb", "Mar", "Apr", "May", "Jun",
```

```
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
```

```
};
```

```
tmElements_t tm;
```

```
void setup() {
```

```
  bool parse=false;
```

```
  bool config=false;
```

```
  // get the date and time the compiler was run
```

```
  if (getDate(DATE) && getTime(TIME)) {
```

```
    parse = true;
```

```
    // and configure the RTC with this info
```

```
    if (RTC.write(tm)) {
```

```
      config = true;
```

```
    }
```

```
  }
```

```
  Serial.begin(9600);
```

```
  while (!Serial) ; // wait for Arduino Serial Monitor
```

```

delay(200);

if (parse && config) {
    Serial.print("DS1307 configured Time=");
    Serial.print(TIME);
    Serial.print(", Date=");
    Serial.println(DATE);
} else if (parse) {
    Serial.println("DS1307 Communication Error :-{");
    Serial.println("Please check your circuitry");
} else {
    Serial.print("Could not parse info from the compiler, Time=");
    Serial.print(TIME);
    Serial.print("", Date="");
    Serial.print(DATE);
    Serial.println("");
}
}

void loop() {

bool getTime(const charstr)
{
    int Hour, Min, Sec;

    if (sscanf(str, "%d:%d:%d", &Hour, &Min, &Sec) != 3) return false;
    tm.Hour = Hour;
    tm.Minute = Min;
    tm.Second = Sec;
    return true;
}

```

```
}
```

```
bool getDate(const char *str)
```

```
{
```

```
    char Month[12];
```

```
    int Day, Year;
```

```
    uint8_t monthIndex;
```

```
    if (sscanf(str, "%s %d %d", Month, &Day, &Year) != 3) return false;
```

```
    for (monthIndex = 0; monthIndex < 12; monthIndex++) {
```

```
        if (strcmp(Month, monthName[monthIndex]) == 0) break;
```

```
    }
```

```
    if (monthIndex >= 12) return false;
```

```
    tm.Day = Day;
```

```
    tm.Month = monthIndex + 1;
```

```
    tm.Year = CalendarYrToTm(Year);
```

```
    return true;
```

```
}
```

Our Project code :

```
#include <Wire.h>
#include <LiquidCrystal.h>
#include <DS1307RTC.h>
#include <TimeLib.h>

tmElements_t tm;

LiquidCrystal LCD(7, 6, 5, 4, 3, 2);

void setalarms(int, int, int );

#define set_save 10
#define incrhour 9
#define incrminute 8
#define buzzer 11
#define redLED 12
#define resetButton 13

int hours = 5, minutes = 0;
void setup()
{
    // Set up button and pin configurations
    pinMode(resetButton, INPUT_PULLUP);
    pinMode(buzzer, OUTPUT);
    pinMode(redLED, OUTPUT);
    pinMode(set_save, INPUT_PULLUP);
    pinMode(incrhour, INPUT_PULLUP);
    pinMode(incrminute, INPUT_PULLUP);
    digitalWrite(set_save, HIGH);
    digitalWrite(incrhour, HIGH);
    digitalWrite(incrminute, HIGH);
    digitalWrite(redLED, LOW);
    // Initialize the LCD display
    LCD.begin(16,2);
    LCD.clear();
    delay(1000);
    LCD.setCursor(0,0);
    LCD.print("Digital Clock");
    delay(1000);
    LCD.setCursor(0,1);
    LCD.print("0420 Embedded");
    delay(2000);
}

void loop()
{
    // Update the current time and date
```



```

    time_date();
// Check if the reset button is pressed
    if (digitalRead(resetButton) == 0)
    {
        hours = 5;
        minutes = 0;
        LCD.clear();
        LCD.setCursor(0,0);
        LCD.print("Digital Clock");
        LCD.setCursor(0,1);
        LCD.print("0420 Embedded");
        delay(1000);
    }
    // Check if the set/save button is pressed
    if(digitalRead(set_save) == 0 )
    {
        delay(170);
        setalarms(tm.Hour, tm.Minute, tm.Second);
    }
// Check if it's time for the alarm
    else if(hours == tm.Hour && minutes == tm.Minute)
    {
        LCD.clear();
        LCD.setCursor(0,0);
        LCD.print("    Wake UP !");
        LCD.setCursor(0,1);
        LCD.print("    Wake UP !");
        beep();
    }

    else
    {
        digitalWrite(redLED, LOW);
    }
}
// Function to display current time and date on the LCD
void time_date()
{
    if (RTC.read(tm))
    {
        LCD.setCursor(0,0);
        LCD.print("Time= " + (String)tm.Hour + ":" + (String)tm.Minute + ":" +
(String)tm.Second + "   ");
        LCD.setCursor(0,1);
        LCD.print("Date= " + (String)tm.Day + "/" + (String)tm.Month + "/" +
(String)tm.YearToCalendar(tm.Year) + "   ");
    }
}

```

```

    delay(170);
}
// Function to set alarms
void setalarms(int HOURS, int MINUTES, int SECONDS)
{
    // Clear the LCD display and set the cursor to the starting position
    LCD.clear();
    LCD.setCursor(0,0);
    LCD.print("    SET ALARM    ");
    LCD.setCursor(0,1);

    // Infinite loop for setting the alarm time
    while (1)
    {
        // Check if the hour increment button is pressed
        if (digitalRead(incrhour) == 0 && digitalRead(incrminute) == 1 &&
digitalRead(set_save) == 1 )
        {
            HOURS = HOURS + 1;
            delay(170);
            if (HOURS > 23)
                HOURS = HOURS - 24;
        }
        // Check if both hour and minute increment buttons are pressed, and set the
alarm
        else if (digitalRead(incrhour) == 1 && digitalRead(incrminute) == 1 &&
digitalRead(set_save) == 0)
        {
            hours = HOURS;
            minutes = MINUTES;
            LCD.clear();
            LCD.setCursor(0,0);
            LCD.print("ALARM TIME    ");
            LCD.setCursor(0,1);
            LCD.print("HAS BEEN SENT    ");
            delay(2000);
            break; // Exit the loop
        }
        // Check if the minute increment button is pressed
        else if (digitalRead(incrhour) == 1 && digitalRead(incrminute) == 0 &&
digitalRead(set_save) == 1)
        {
            MINUTES = MINUTES + 1;
            delay(170);
            if (MINUTES > 60)
                MINUTES = MINUTES - 60;
        }
        // Display the current alarm time on the LCD
        LCD.setCursor(4,1);

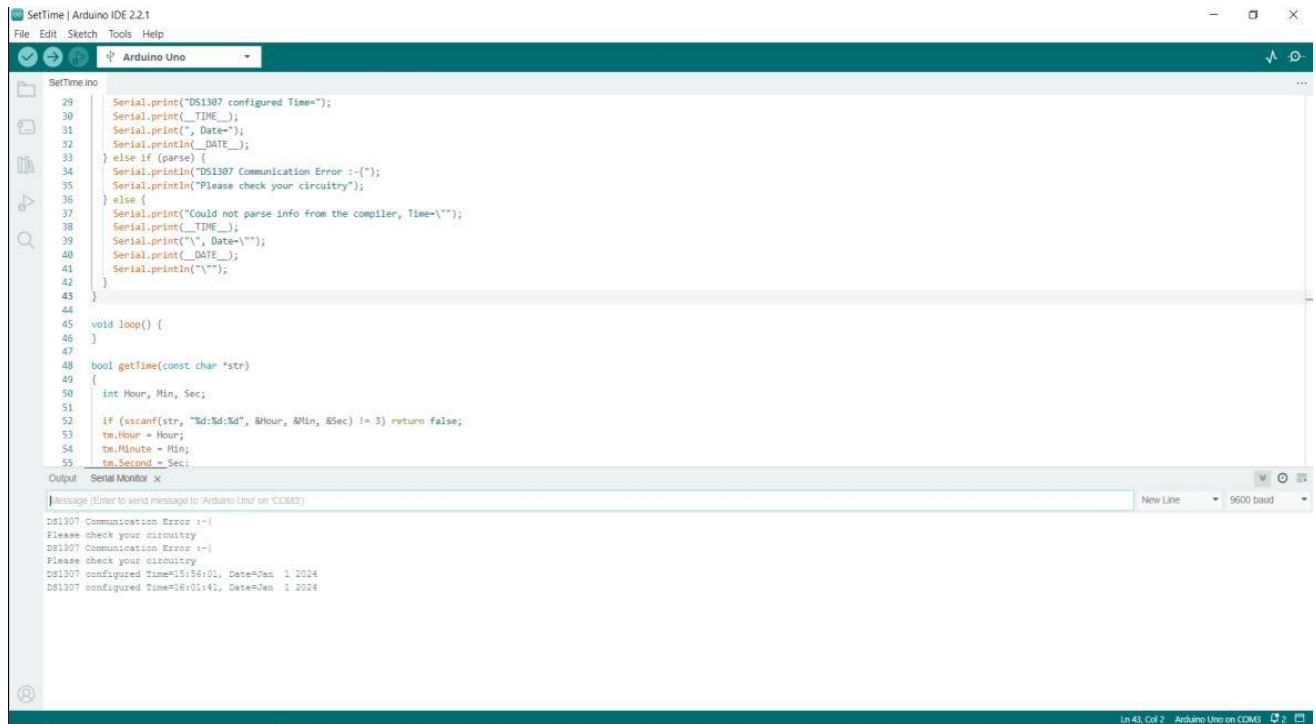
```

```
    LCD.print((String)HOURS + ":" + (String)MINUTES + ":" + (String)SECONDS +  
    "    ");  
  }  
}  
// Function to produce a beep and flash the red LED  
void beep()  
{  
  digitalWrite(buzzer, HIGH);  
  delay(100);  
  digitalWrite(buzzer, LOW);  
  delay(100);  
  digitalWrite(redLED, HIGH);  
  delay(100);  
  digitalWrite(redLED, LOW);  
  delay(100);  
}
```

5)

DEMO

We set the time and date of DS1307RTC by transferring the timeset code to our RTC circuit that we connected to Arduino.

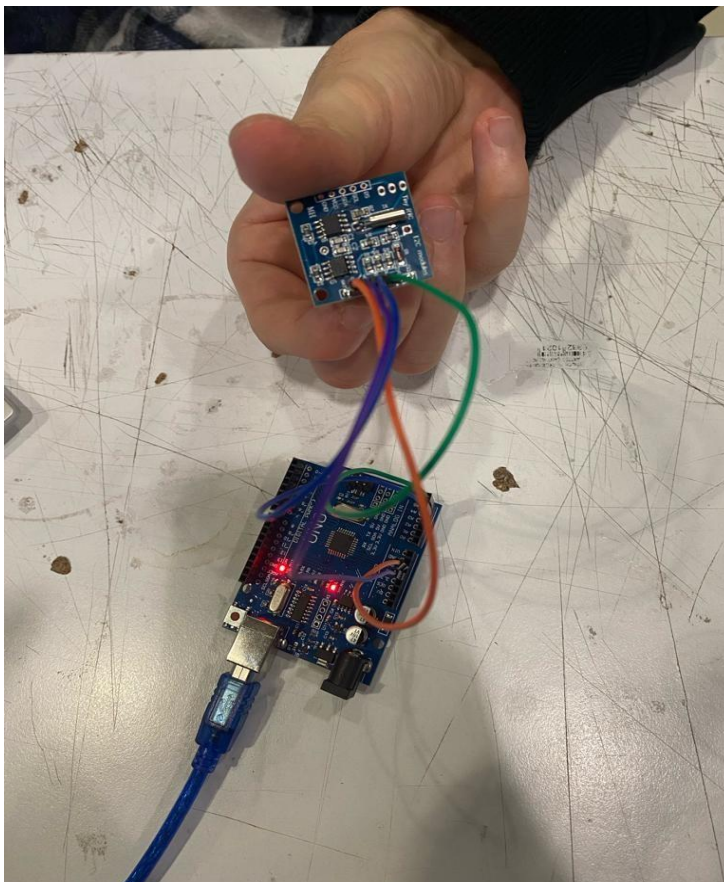


The screenshot shows the Arduino IDE interface with the 'SetTime.ino' file open. The code is as follows:

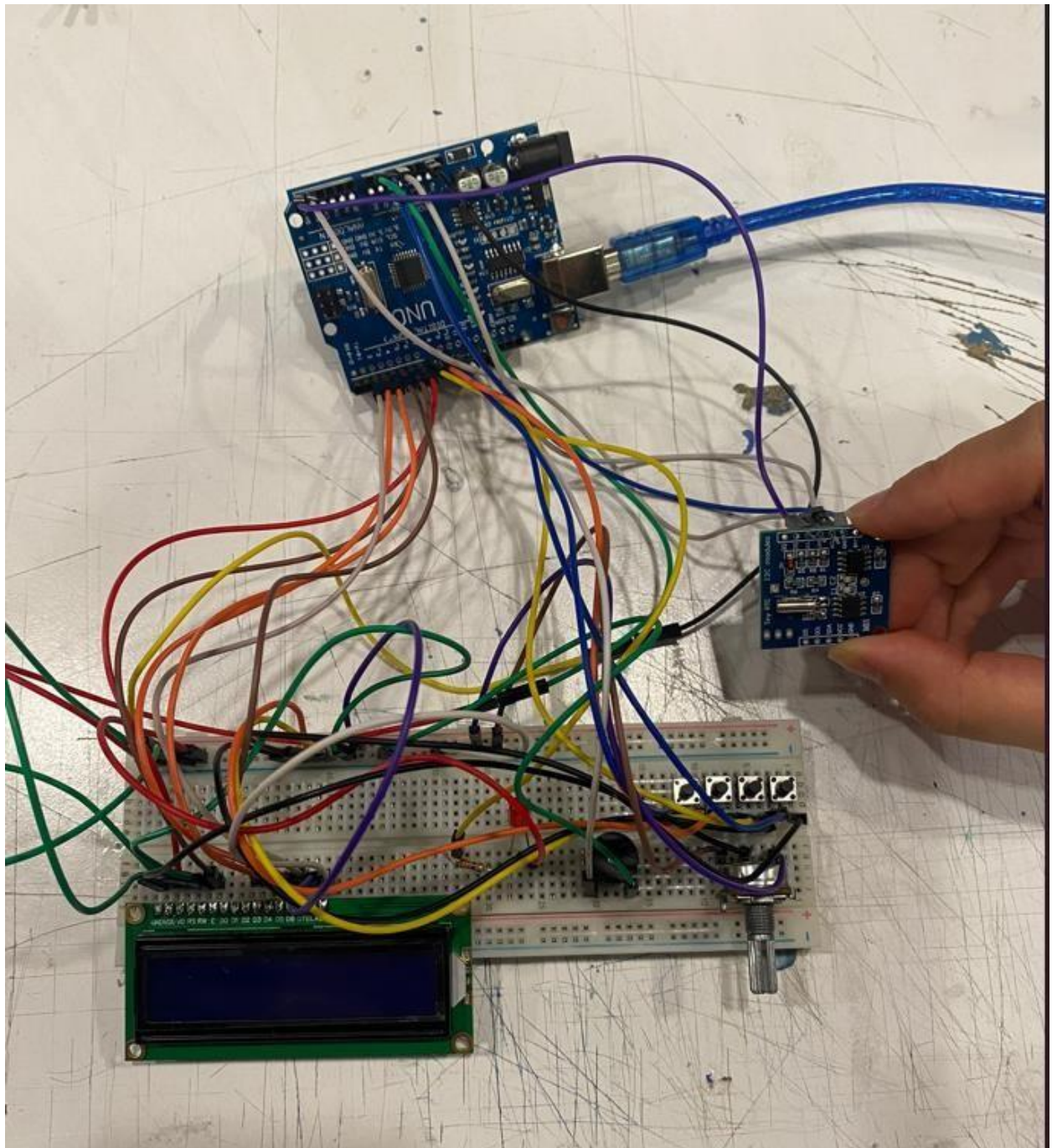
```
29 Serial.print("DS1307 configured Time=");
30 Serial.print(_TIME__);
31 Serial.print(", Date=");
32 Serial.println(_DATE__);
33 } else if (parse) {
34   Serial.println("DS1307 Communication Error :-(");
35   Serial.println("Please check your circuitry");
36 } else {
37   Serial.print("Could not parse info from the compiler, Time=\"");
38   Serial.print(_TIME__);
39   Serial.print("\", Date=\"");
40   Serial.print(_DATE__);
41   Serial.println("\");
42 }
43 }
44
45 void loop() {
46 }
47
48 bool getTime(const char *str)
49 {
50   int Hour, Min, Sec;
51
52   if (sscanf(str, "%d:%d:%d", &Hour, &Min, &Sec) != 3) return false;
53   tm.Hour = Hour;
54   tm.Minute = Min;
55   tm.Second = Sec;
```

The Serial Monitor at the bottom shows the following output:

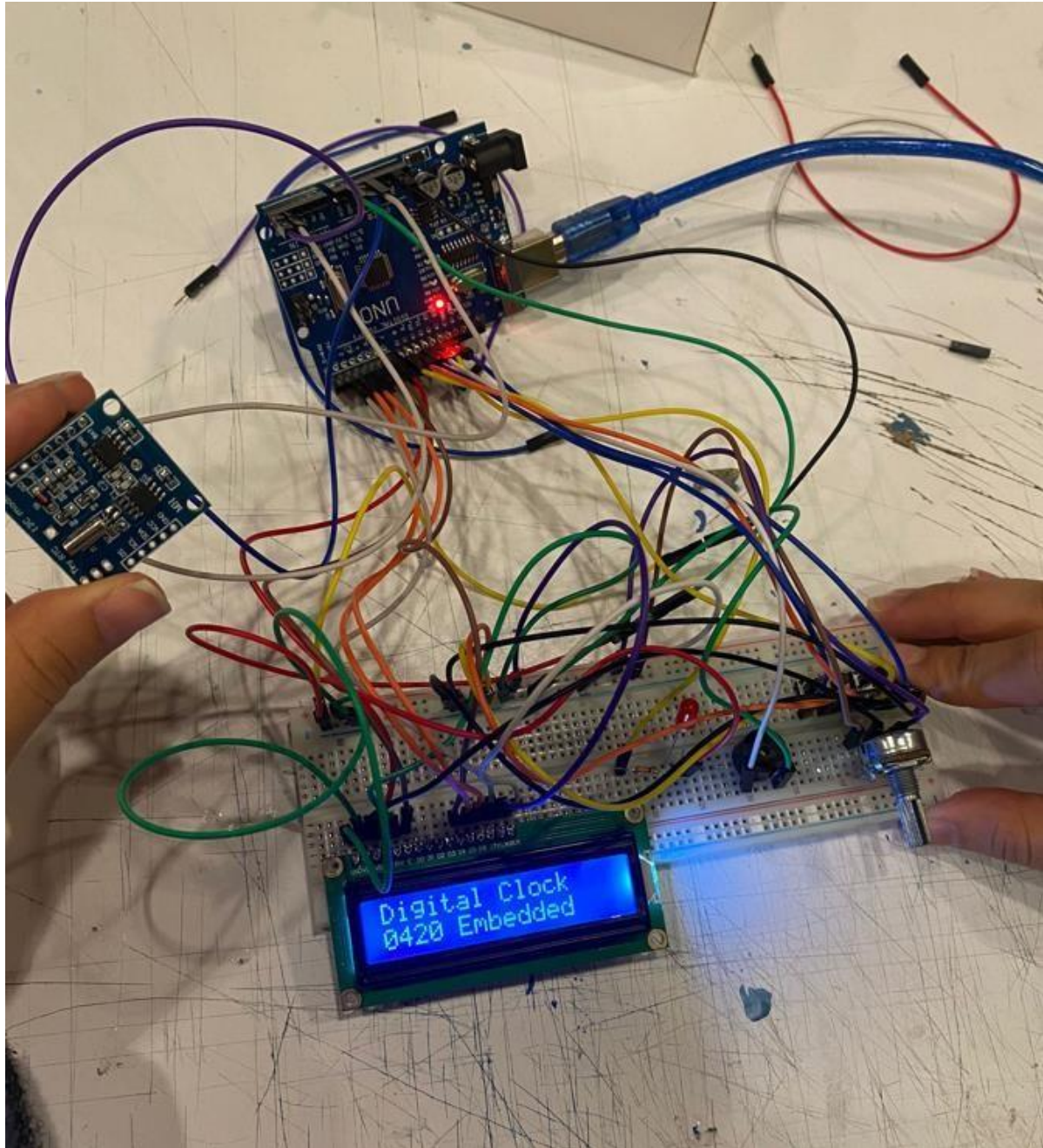
```
DS1307 Communication Error :-[
Please check your circuitry
DS1307 Communication Error :-[
Please check your circuitry
DS1307 configured Time=15:56:01, Date=Jan 1 2024
DS1307 configured Time=16:01:41, Date=Jan 1 2024
```



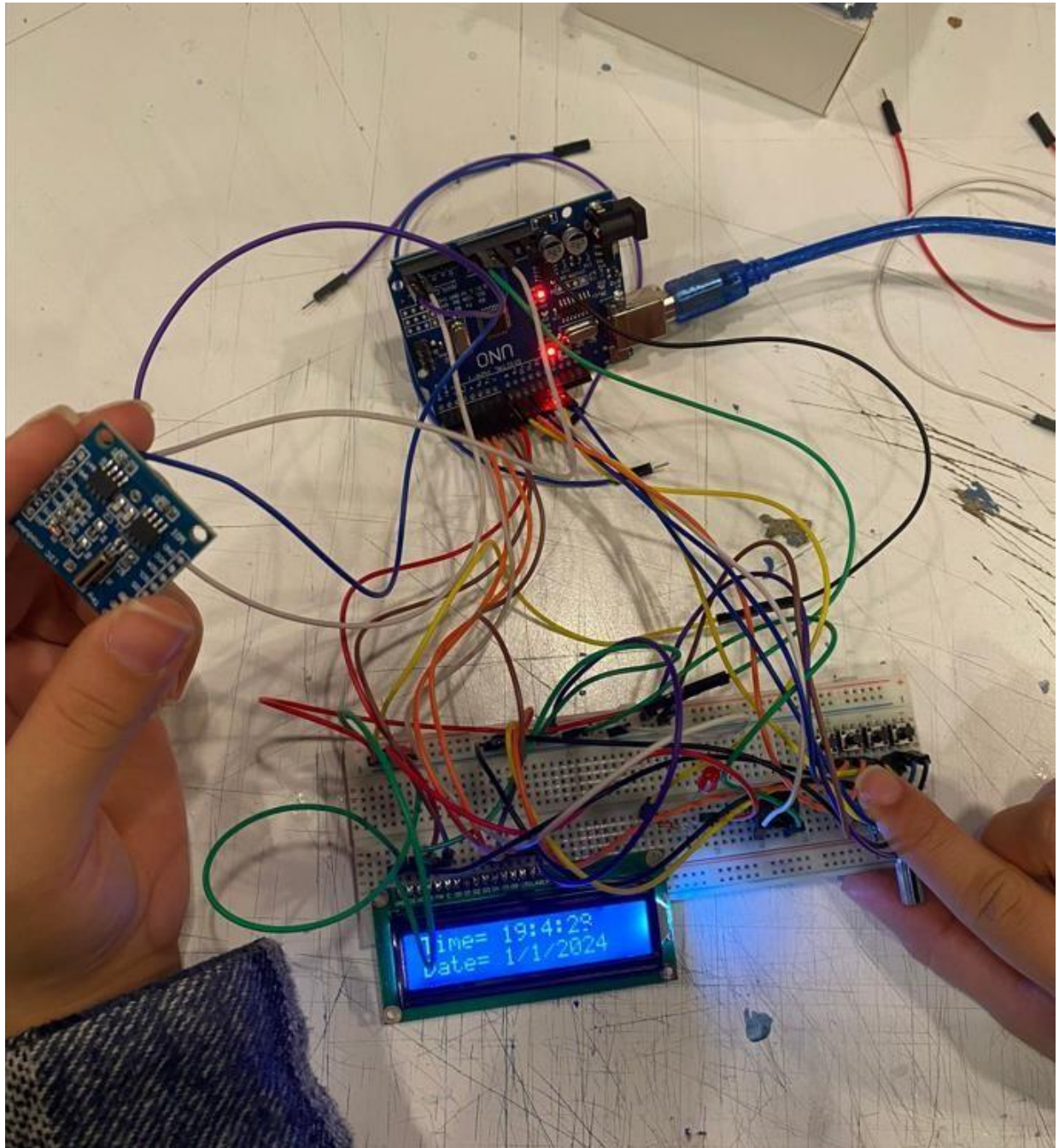
Closed state of our circuit.



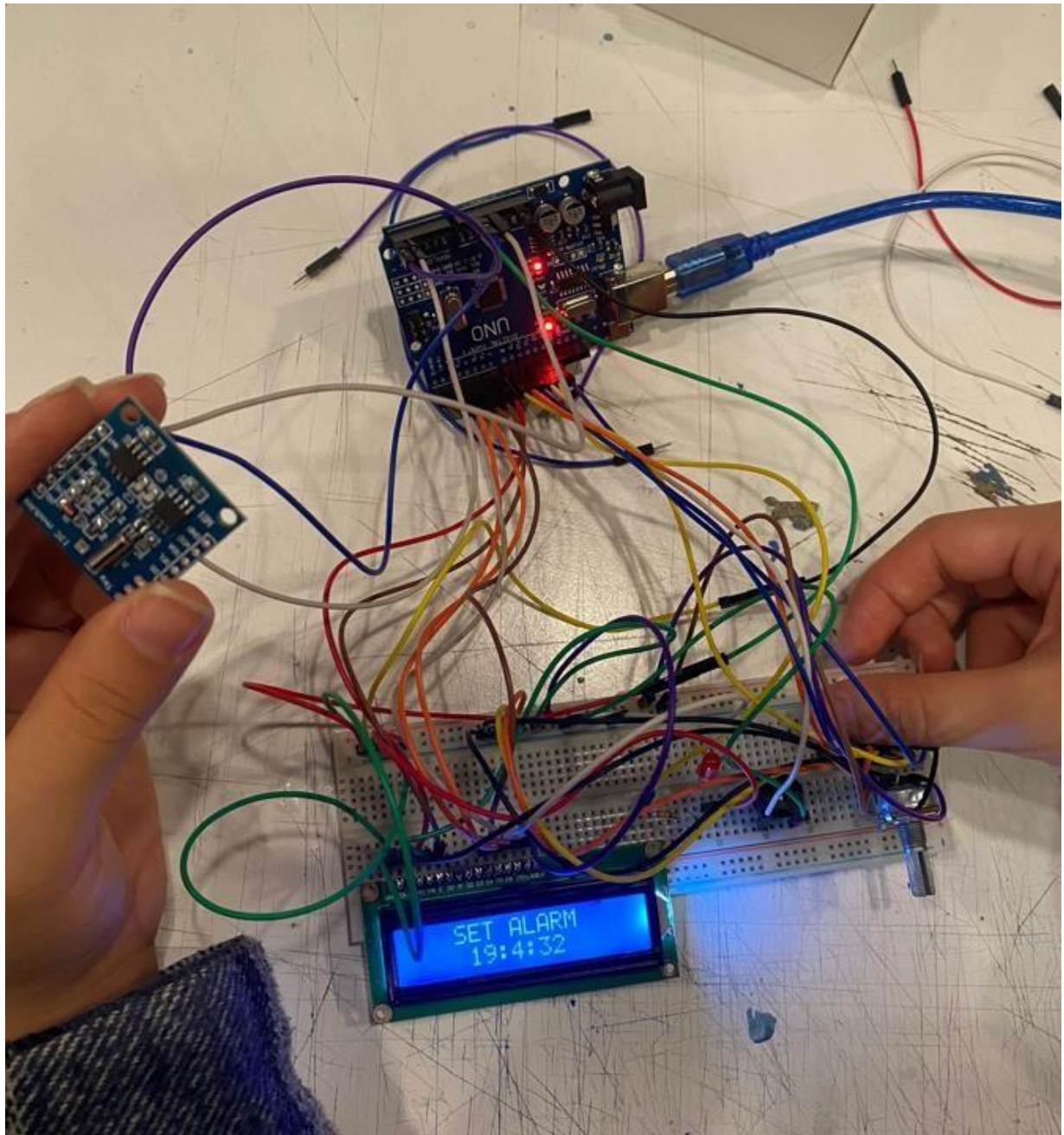
Initial State of our Project.



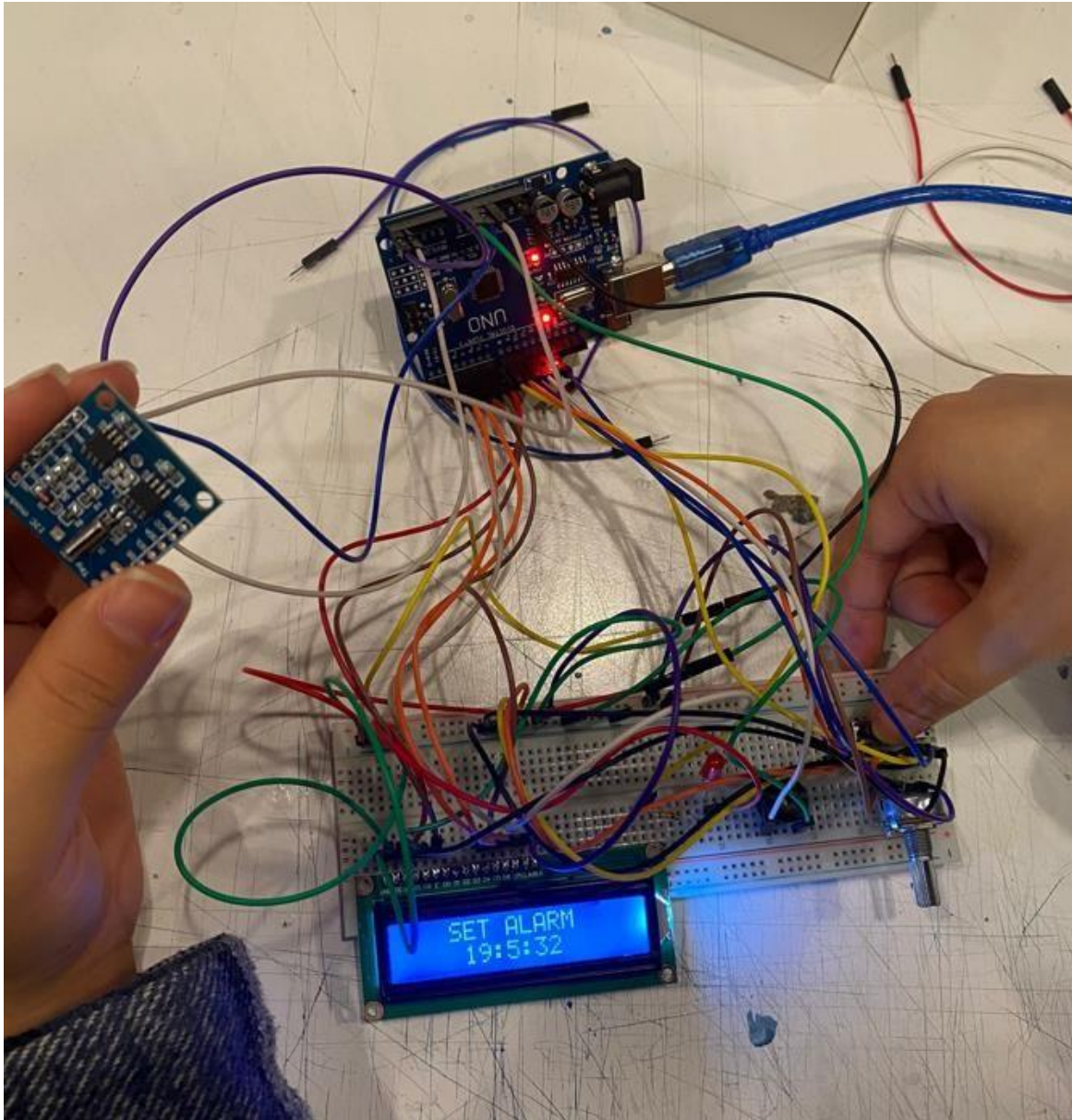
After the initial state, date and time information in RTC shows.



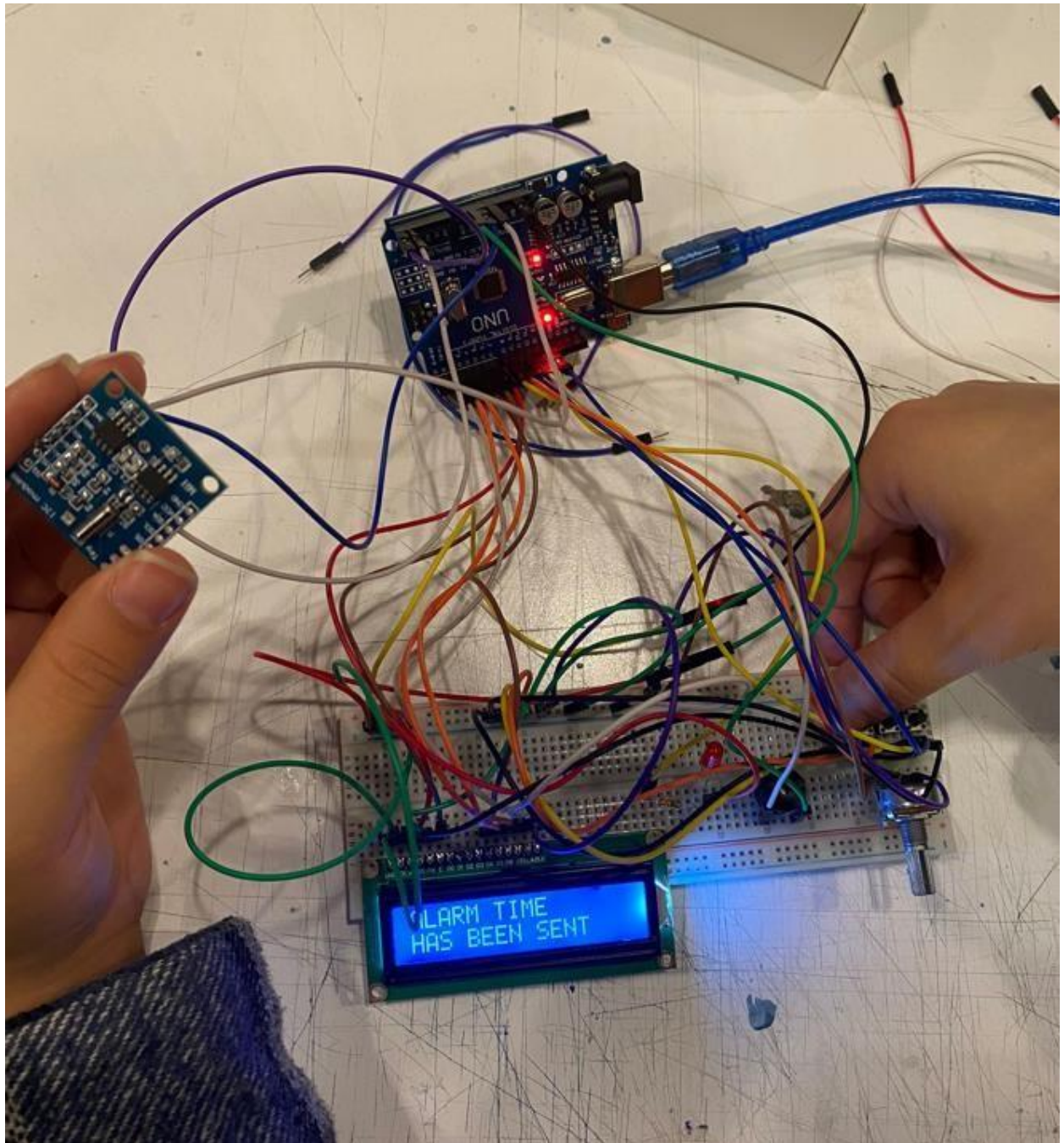
For setting the alarm, we pressed the first button and set alarm screen showed.



We set the alarm in time 19:05 (we increase the minutes with third button.)

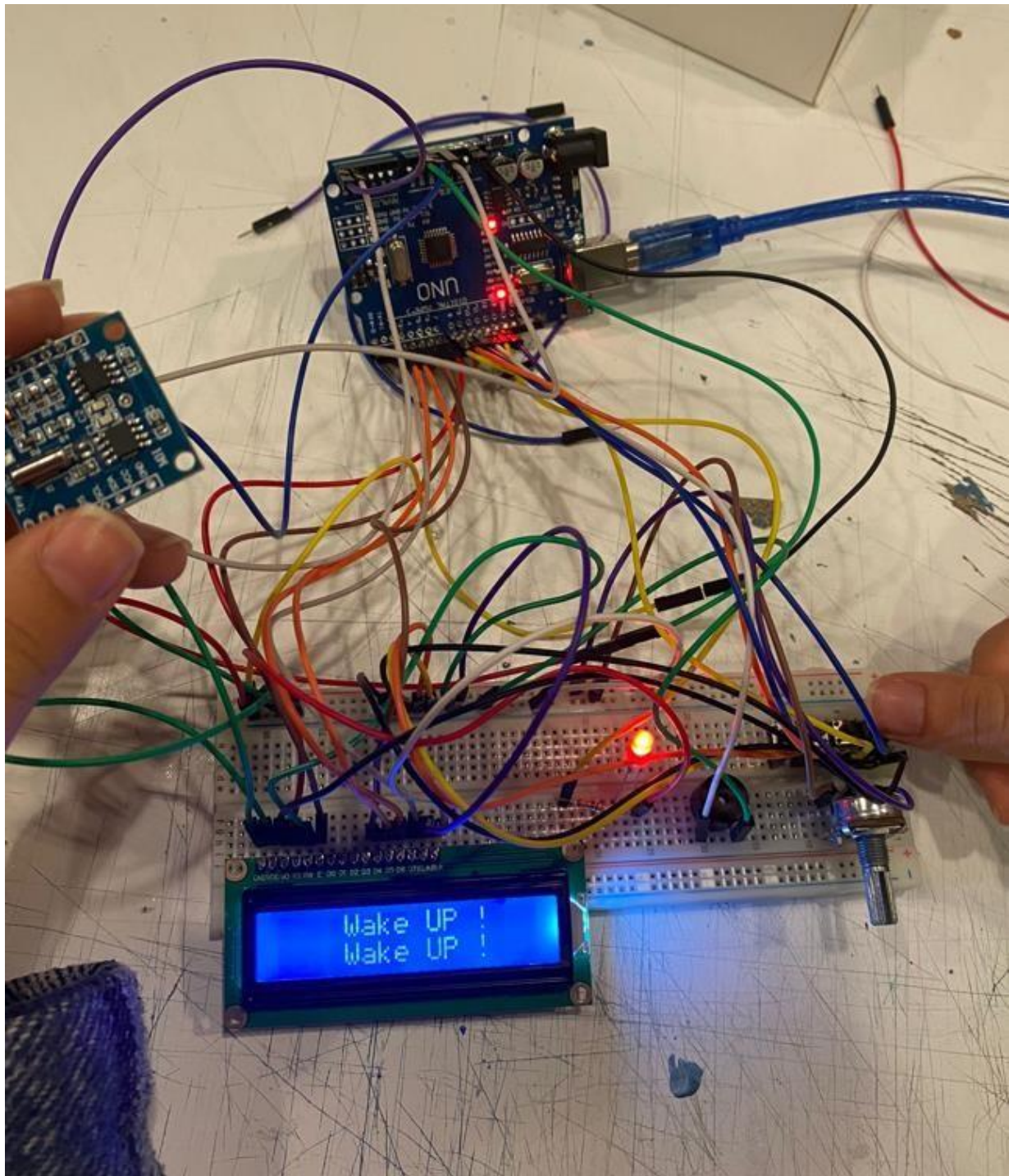


After the adjust the time, for setting the alarm we pressed the first button again.

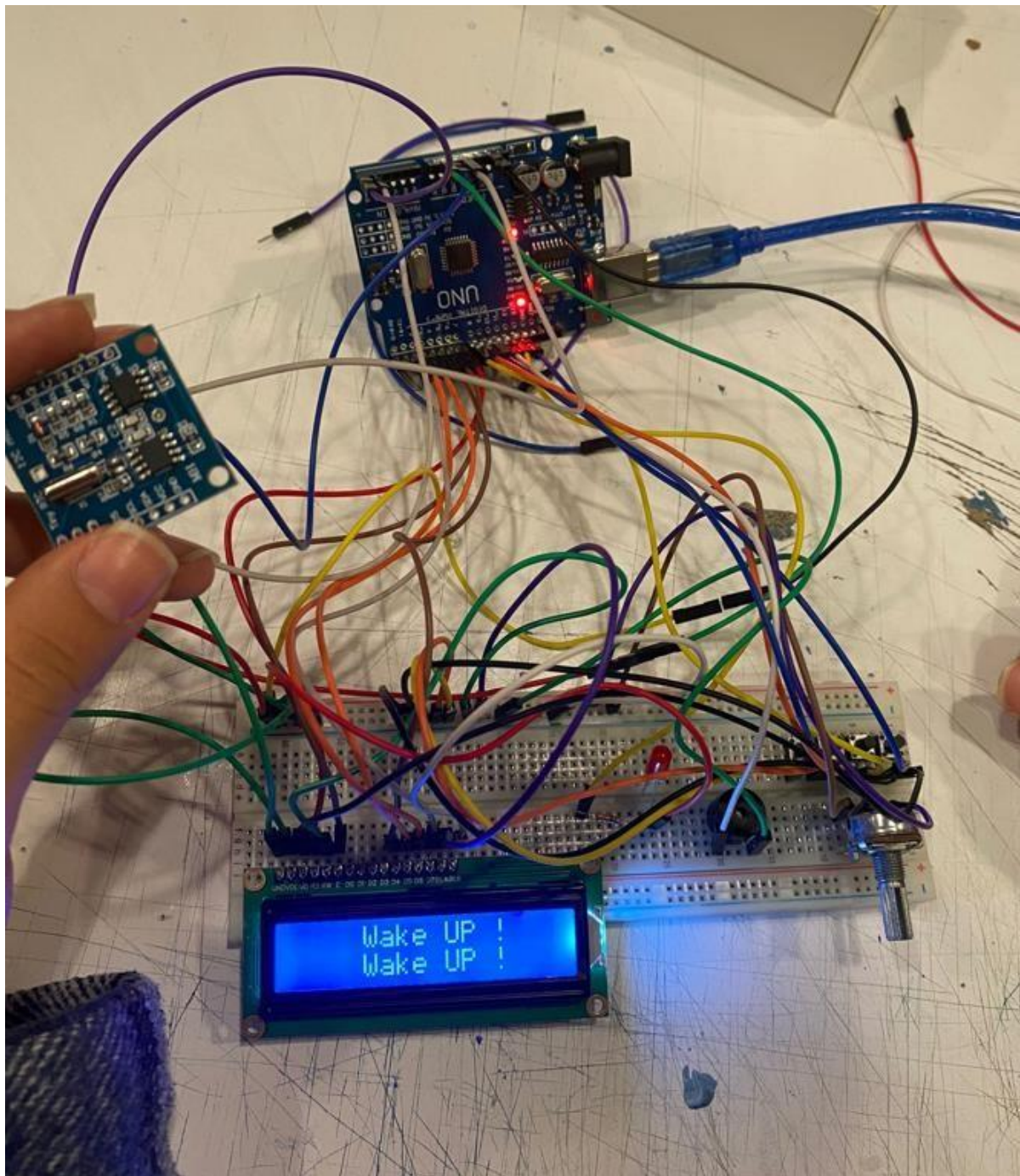


When the time we adjust become equal to the RTC time the red led flashes and buzzer rings. (It is in a loop 0.1 seconds on, 01. Seconds off)

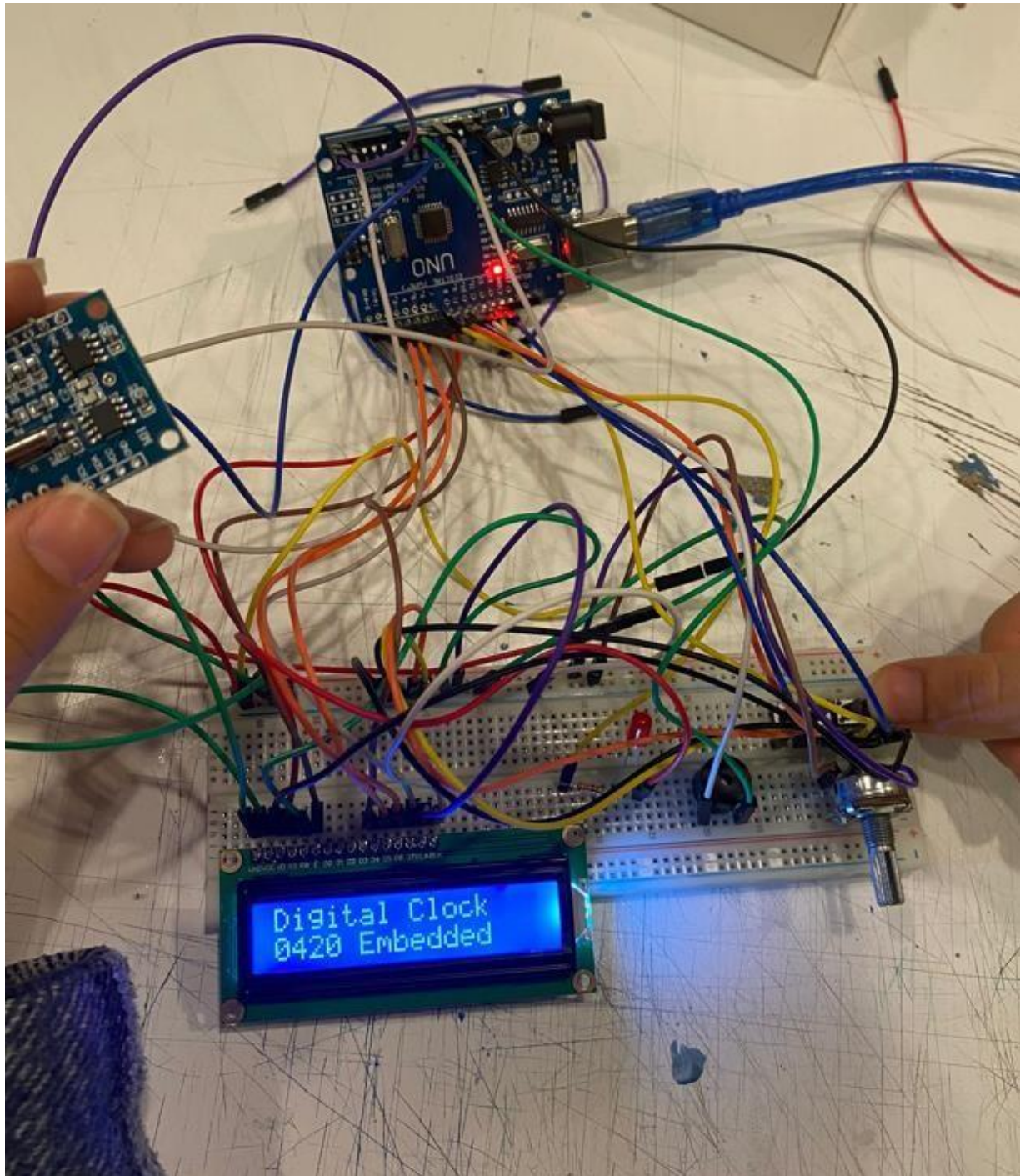
Red led flashes and buzzer rings.



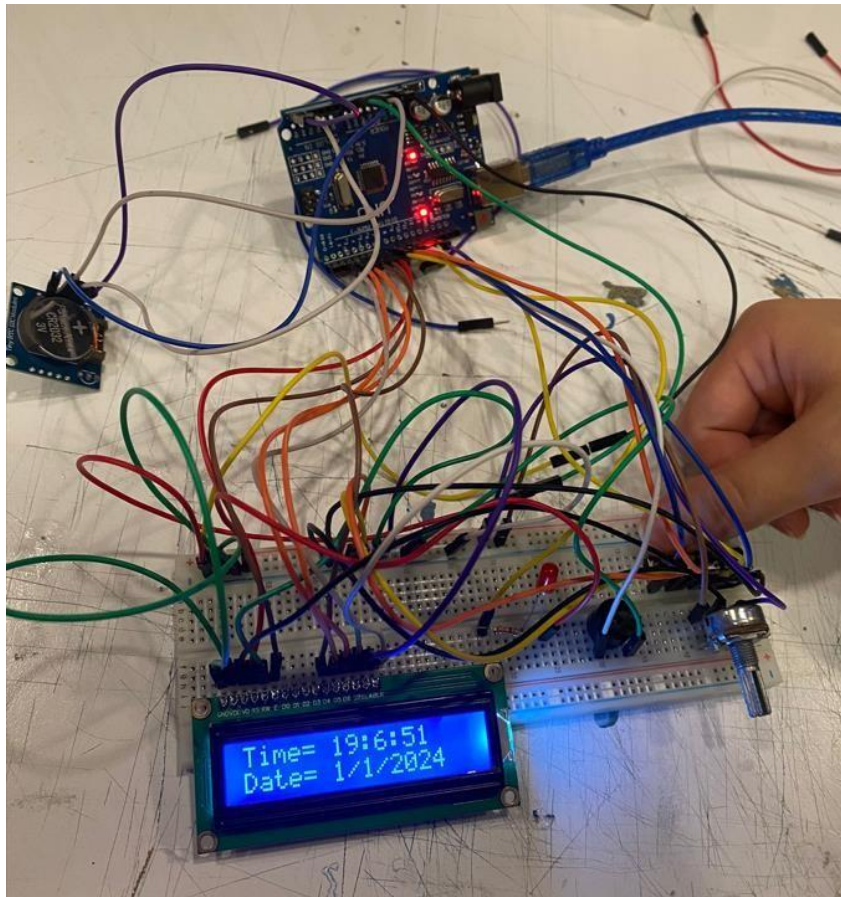
Red Led doesn't flashes, buzzer doesn't rings.

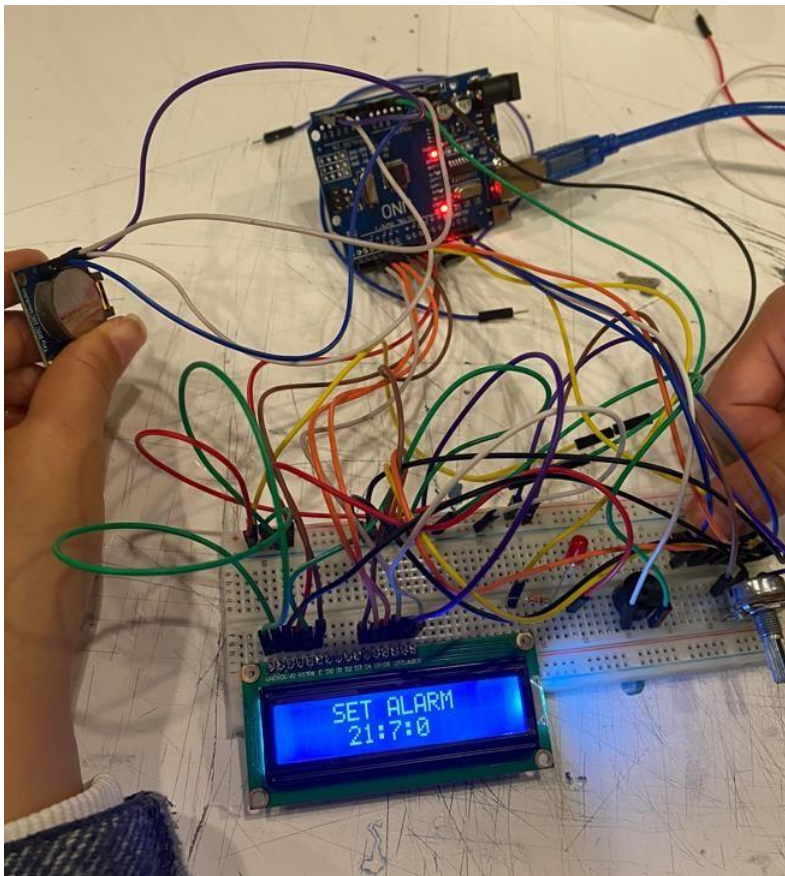
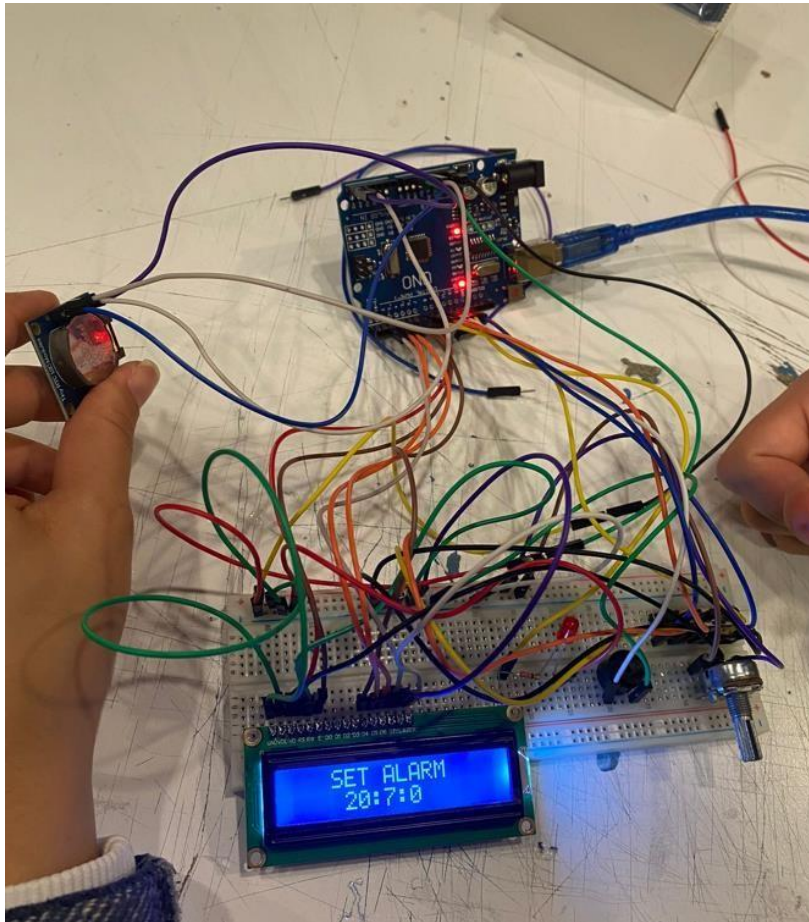


And when we pressed the fourth button (reset button) , we went back to the initial position.



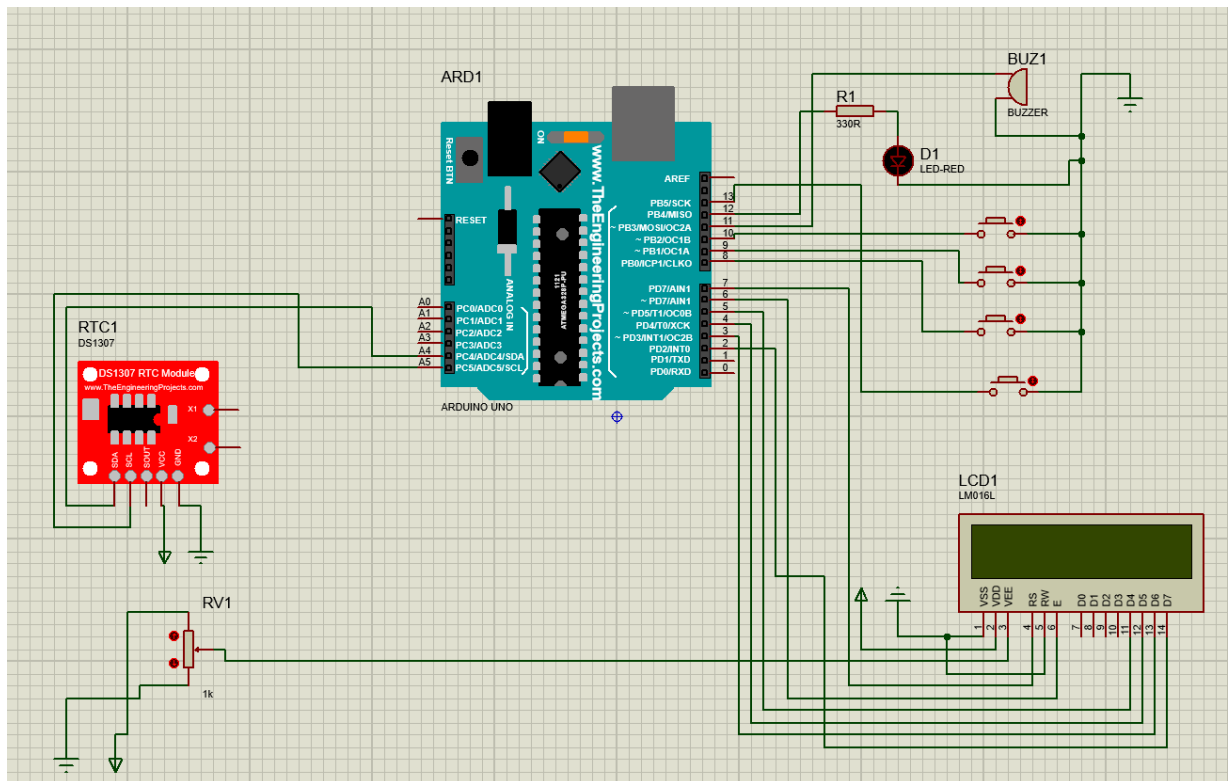
This is just shown the increment of time:



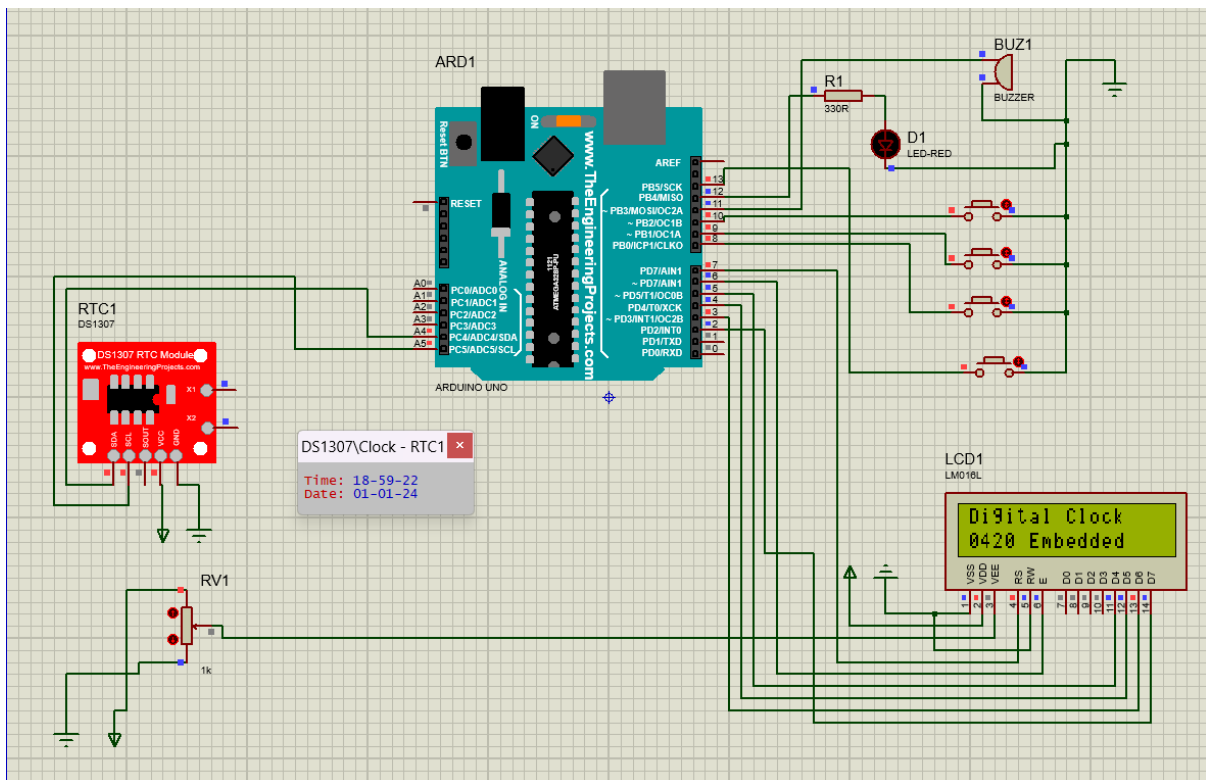


6) SIMULATION

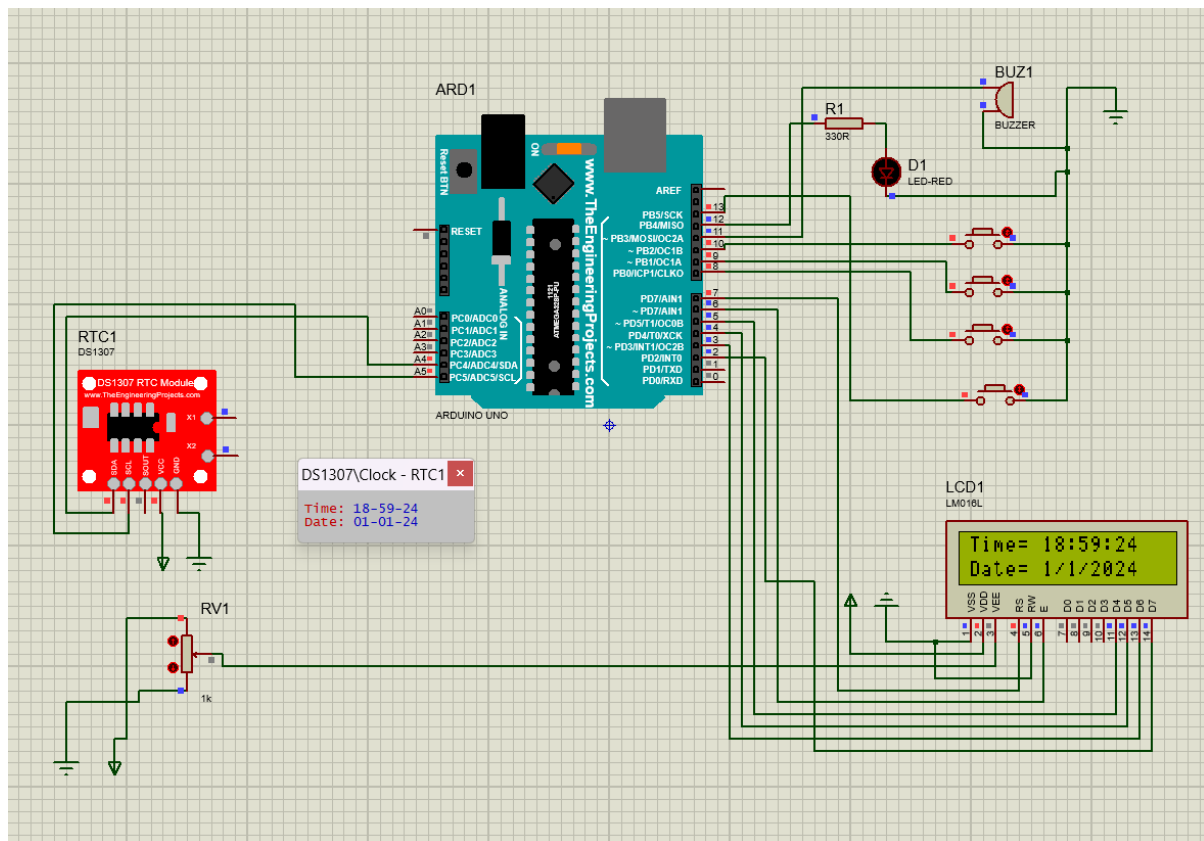
Before the simulation is start



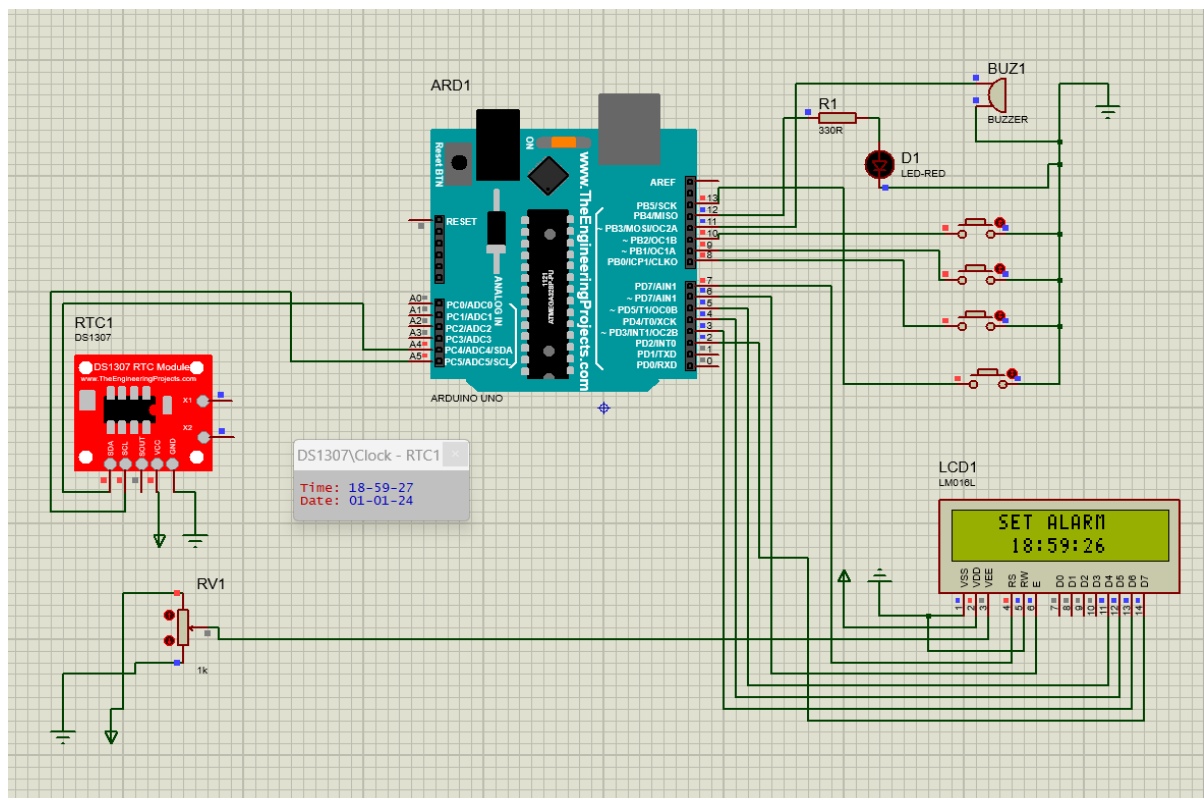
After we start the simulation, Digital Clock and 0420 Embedded texts appears.



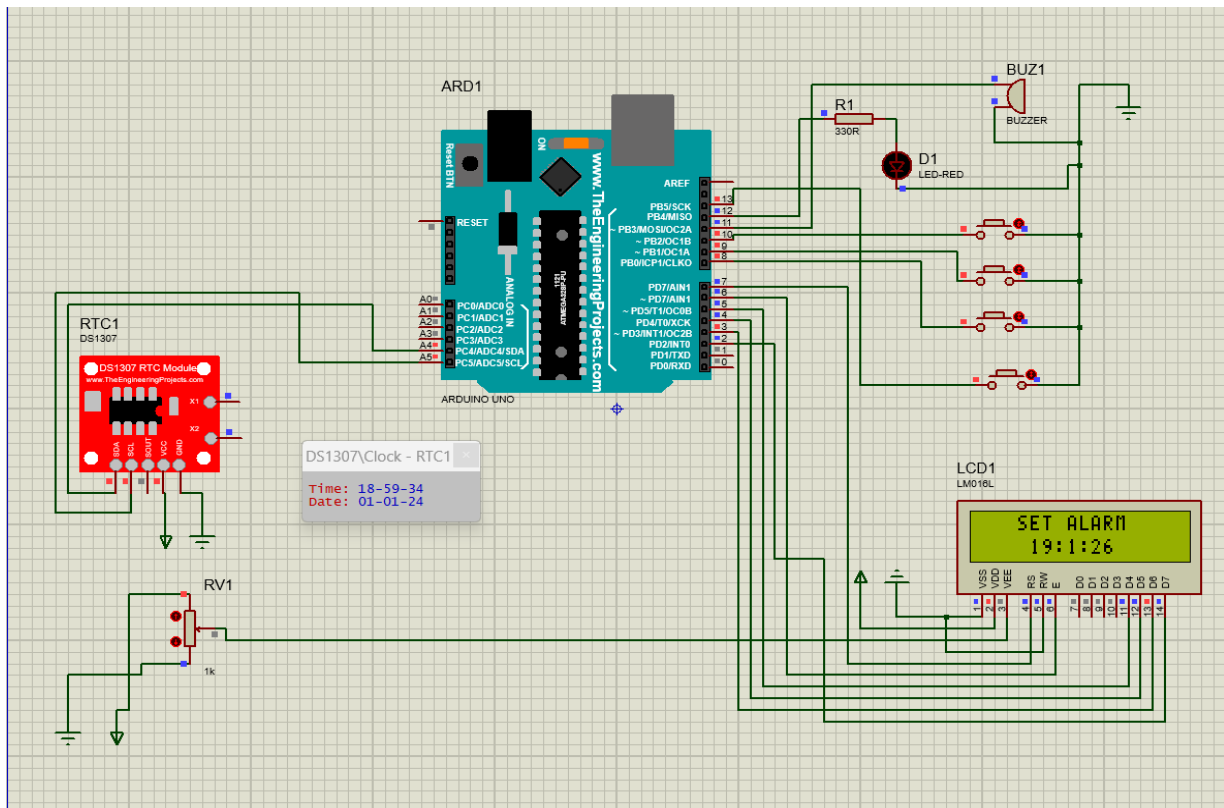
Then, time and date information in the RTC will shown in the LED Screen.



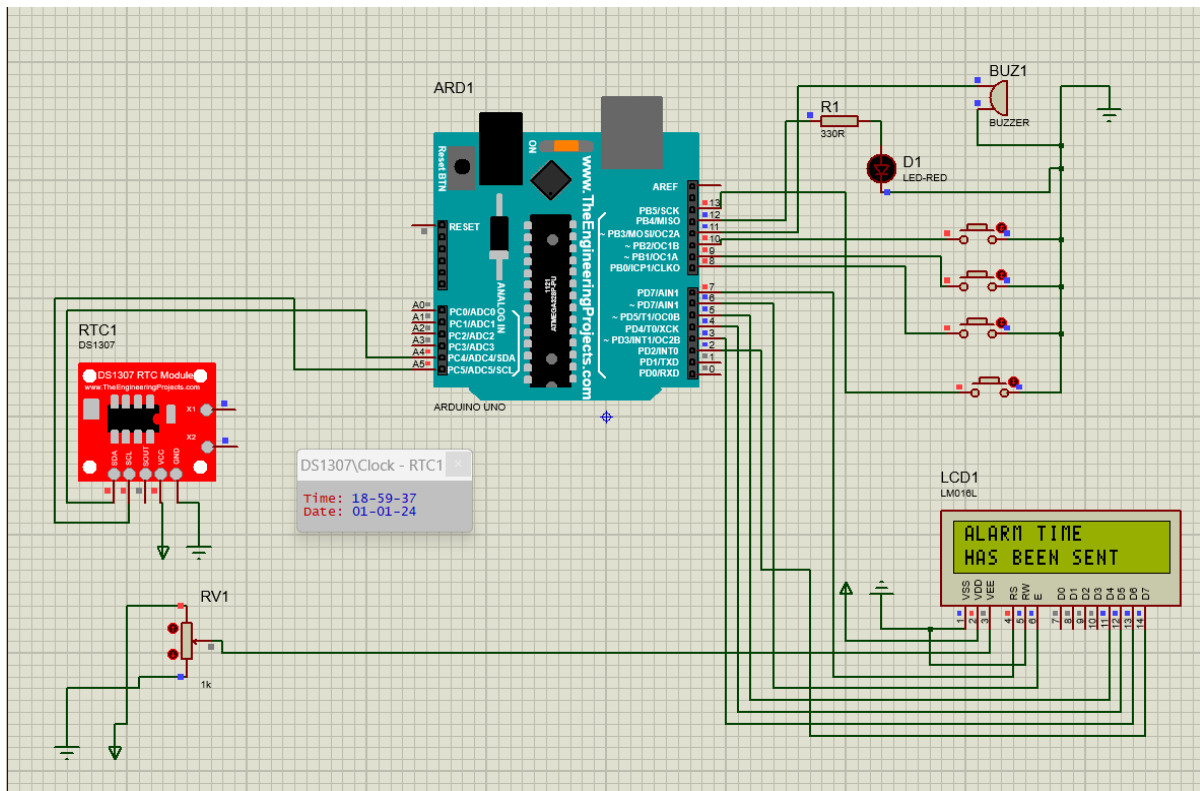
When we want to set an alarm, we click the first button. And Set Alarm current time text will be shown.



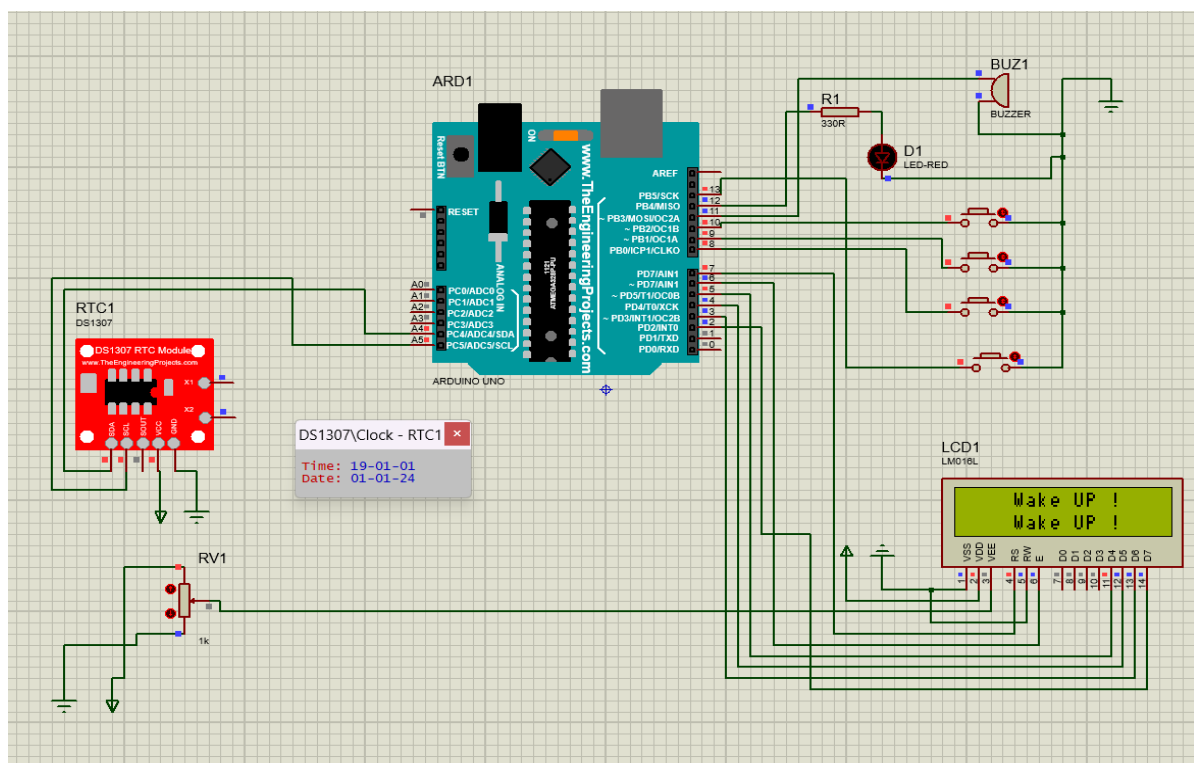
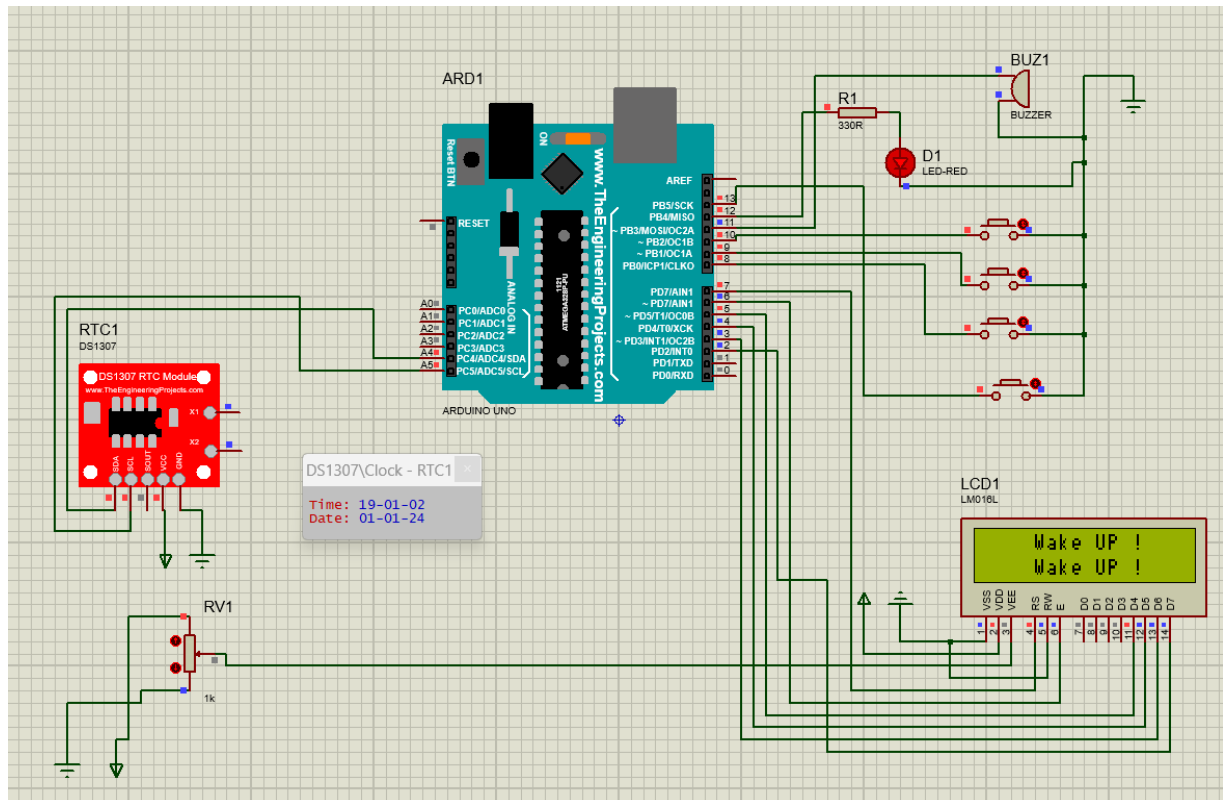
Using the second and third button, we can adjust the hour and minute. (hour with second button and minutes with the third button)



After adjust the time , for setting the alarm we should click the first button again and Alarm Time Has Been Send text will shown in the LCD Screen



And our time is continues. When the RTC time is equal to the time we set. Wake Up ! Wake Up ! text comes, red led flashes and buzzer rings for a 0.1 second and doesn't flashes , doesn't rings for a 0.1 second. (in a loop during the 1 minutes, In order to exit the loop, it must exceed the set minute, that is, when the hour and minute we set are not equal to RTC, the loop is exited.)



If we want to exit out of the loop (closed the alarm), we must pressed the fourth button(reset button) . Then, Clock will back to the initial position. And also we can use these feature for back to the initial position.

