Co-funded by the
Erasmus+ Programme
of the European Union

MODULE ON MICROCONTROLLER PROGRAMING

Title

Using temperature sensor for Arduino applied for COVID19

Required elements

- 1x Arduino Uno
  https://lcsc.com/product-detail/Development-Boards-Development-Kits_Arduino-Arduino-UNO-Rev3_C123746.html
- 1x Breadboard
  https://www.newark.com/multicomp/mcbb400/breadboard-solderless-abs/dp/99W1759?COM=ref_hackster
- 1x Set of jumper wires
  https://www.newark.com/adafruit/759/wire-gauge-28awg/dp/88W2571?COM=ref_hackster
- 1x Temperature Sensor IC DS18B20+
  https://www.newark.com/maxim-integrated-products/ds18b20/temp-sensor-0-5deg-c-to-92-3/dp/73Y1389?COM=ref_hackster
- 1x JLCPCB customized PCB
  https://cart.jlcpcb.com/quote?orderType=1&stencilWidth=100&stencilLength=100&stencilCounts=5&stencilLayer=2&stencilPly=1.6&steelmeshSellingPriceRecordNum=A8256537-5522-491C-965C-646F5842AEC9&purchaseNumber=
- 1x Resistor 1k ohm
  https://www.newark.com/multicomp/mccfr0w4j0102a50/carbon-film-resistor-1kohm-250mw/dp/58K5001?COM=ref_hackster
- 1x Bitzcraze Micro SD card deck
  https://store.bitcraze.io/collections/decks/products/sd-card-deck
- 3x SparkFun Pushbutton switch 12mm
  https://www.newark.com/omron-electronic-components/b3f-1000/switch-tactile-spst-no-50ma-though/dp/36K7138?COM=ref_hackster
- RGB Blacklight LCD 16x2
  https://www.newark.com/adafruit-industries/398/rgb-backlight-positive-lcd-16x2/dp/53W5919?COM=ref_hackster
- Arduino Nano R3
  https://www.newark.com/arduino/a000005/dev-board-atmega328-arduino-nano/dp/13T9275?COM=ref_hackster
- JLCPCB Printed Circuit Board
  https://jlcpcb.com/

Apps

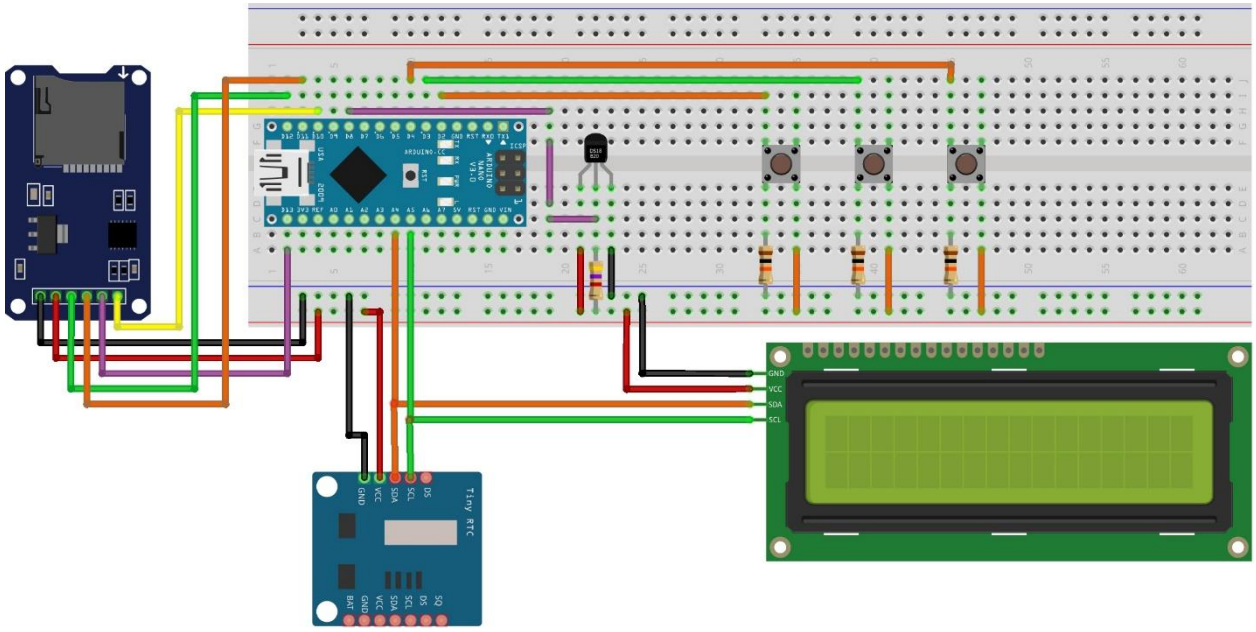
- Arduino IDE

  or
- Arduino Web Editor

Construction of the Datalogger with Temperatur Sensor with Arduino

As previously mentioned, the project consists of creating a Datalogger with Temperature Sensor with Arduino and, through this data, we can monitor the temperature of the patient being treated.

Thus, the circuit will be developed.



Therefore, as you can see, this circuit has a DS18B20 temperature sensor with Arduino, which is responsible for measuring the patient's temperature reading.

In addition, the Arduino Nano will be responsible for collecting this data and storing it on the SD Card Module's memory card.

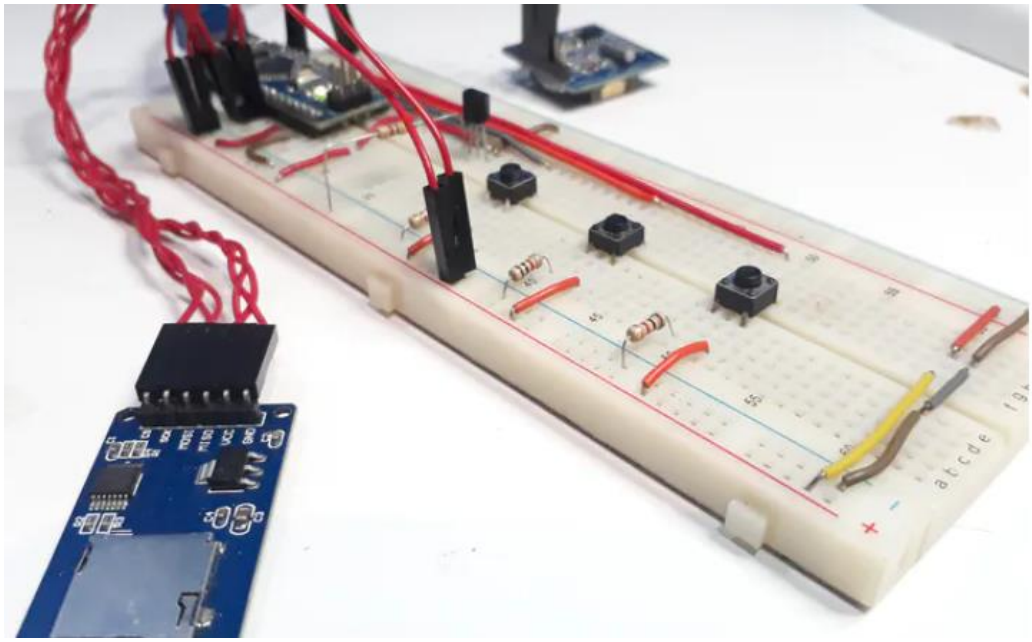
Each information will be saved with its respective time, which will be read from the RTC Module DS1307.

Thus, for the data of the temperature sensor with Arduino to be saved, the user must perform the process through the Control Menu with the 16x2 LCD.

Each button is responsible for controlling an option, as shown on the LCD screen 16x2

- Option M is responsible for starting the measurement and recording of data on the Memory Card.
- Option H is responsible for adjusting the system hours.
- Option O/P is used to confirm data entry in the system or to pause writing data to the memory card.

In addition to updating the hours, the user can select one of the three buttons to monitor the patient with a temperature sensor with Arduino. The circuit is shown below.

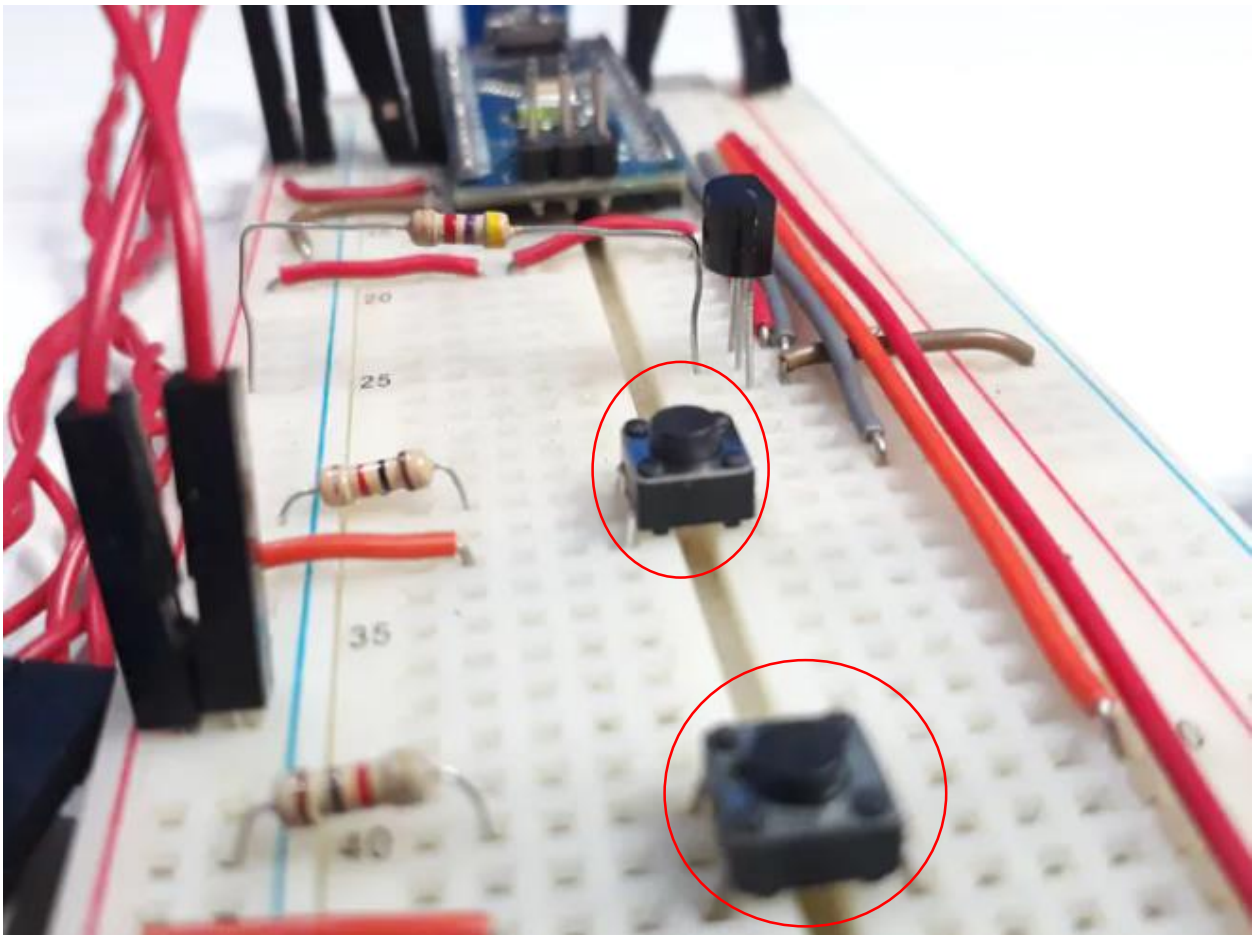


Datalogger control menu

First, the user must check and adjust the system hours. This process is performed when the second button is pressed.

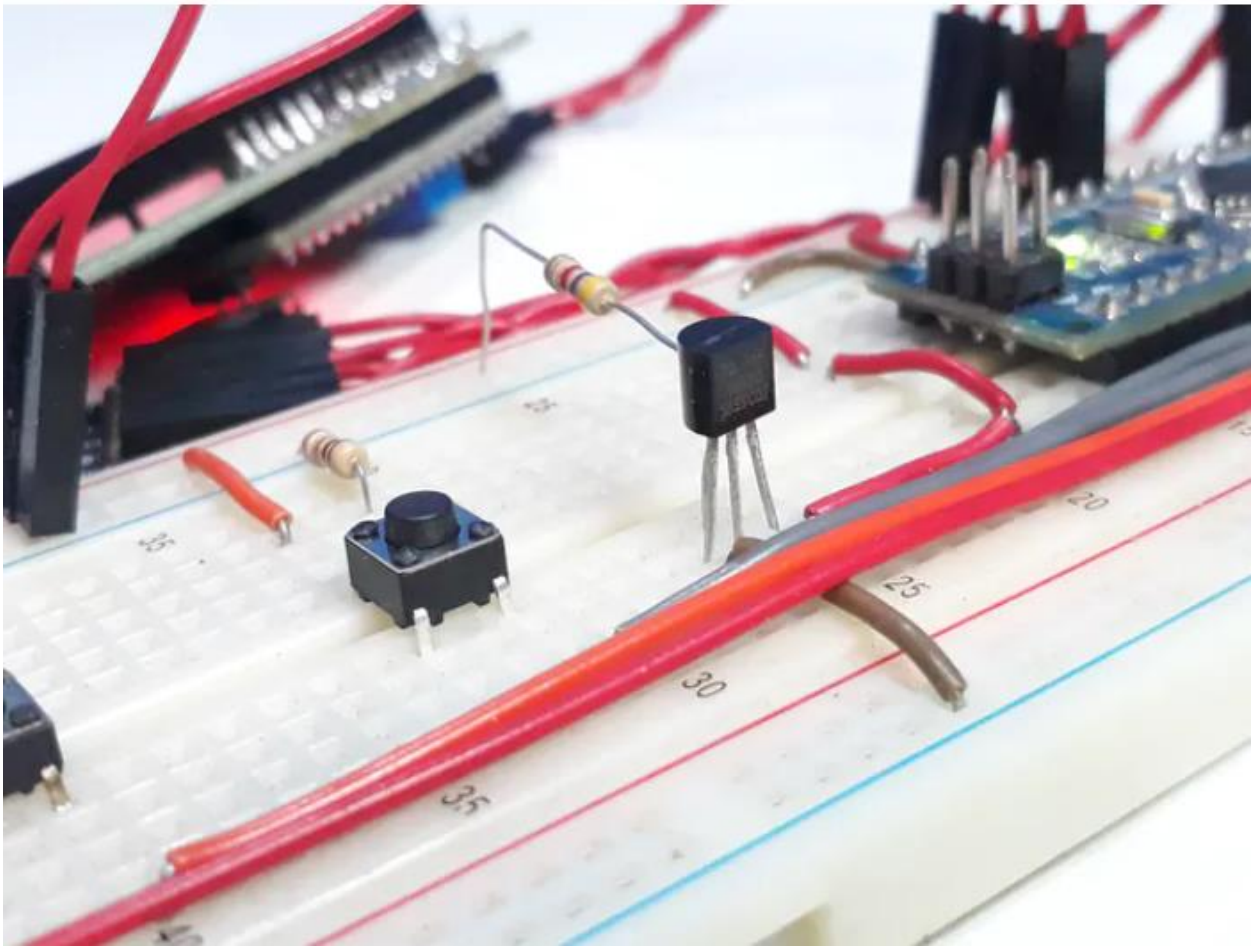
When the button is pressed, the following screen should appear

From this screen, the user will be able to enter the hour and minute values from the buttons connected to digital pins 2 and 3 of the Arduino.



Finally, the user must start the patient monitoring process through the temperature sensor with Arduino Datalogger.

To do this, the user must press the measurement button, which is connected to digital pin 2. Then, the system will perform the reading with the temperature sensor for Arduino and save it on the memory card.



After that, it will create two columns: one for the hours and one for the temperature inside the text file. After that, it will display the hours and temperature on the LCD screen

Coding steps

First, we define all the libraries for controlling the modules and declaring variables used when programming the Datalogger with a temperature sensor for Arduino. The code block is shown below.

```
#include <DallasTemperature.h> //Library with all function of DS18B20 Sens
or
#include <DS1307.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#include <OneWire.h> //OneWire Library for DS18B20 Sensor
#include <SD.h>
#include <SPI.h>

LiquidCrystal_I2C lcd(0x27,16,2);  // Configuration of the LCD 16x2

#define ONE_WIRE_BUS 8 //Digital Pin to connect the DS18B20 Sensor

//Define one instance, oneWire to communicate with the sensor
OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature sensors(&oneWire);
DeviceAddress sensor1;

File myFile;

#define Buttonmeasure 2
#define Buttonadjusthour 3
#define Buttonok 4

bool measure = 0, adjusthour = 0, ok = 0;
bool measure_state = 0, adjusthour_state = 0, ok_state = 0;
bool measure_process = 0, adjust_process = 0;

byte actualMin = 0, previousMin = 0;
byte actualHour = 0, previousHour = 0;
byte minUpdate = 0;

int pinoSS = 10;

int DataTime[7];
```

Hereafter, we have the void setup function. This function is used to configure the pins and device initialization, as shown below.

```
void setup()
{

Serial.begin(9600);
DS1307.begin();
sensors.begin();

pinMode(pinoSS, OUTPUT);

Wire.begin();
lcd.init();
lcd.backlight();

lcd.setCursor(3,0);
lcd.print("Temp System");
lcd.setCursor(3,1);
lcd.print("Datalogger");
delay(2000);

// Localiza e mostra enderecos dos sensores
Serial.println("Locating DS18B20 sensors ...");
Serial.print("Sensor Localization successfully!");
Serial.print(sensors.getDeviceCount(), DEC);
Serial.println(" Sensor");

if(SD.begin())
{
//Initialize the SD Card
Serial.println("SD Card ready to use.");
}

else

{
Serial.println("SD Card initialization failed.");
return;
}

  DS1307.getDate(DataTime);

  lcd.clear();

  sprintf(times, "%02d:%02d", DataTime[4], DataTime[5]);

  lcd.setCursor(5,0);
  lcd.print(times);
  lcd.setCursor(0,1);
  lcd.print("1-M   2-H  3-O/P");
}
```

First, the serial communication, the real-time clock and the temperature sensor for Arduino DS18B20 were started.

After initializing and testing the devices, the message with the menu options was printed on the 16x2 LCD screen.

After that, the system reads the hours and updates the value by calling the updateHour function. Thus, this function has the purpose of presenting the hourly value every minute. The function code block is shown below.

```
void updateHour()
{
      DS1307.getDate(DataTime);

      if(DataTime[5] != minUpdate)
      {
          sprintf(times, "%02d:%02d", DataTime[4], DataTime[5]);

          lcd.setCursor(0,0);
          lcd.print("                ");
          lcd.setCursor(5,0);
          lcd.print(times);

          minUpdate = DataTime[5];
      }

}
```

The code portion for controlling the hours is shown below

```
if(adjusthour == 0 && adjusthour_state == 1)
{
  adjusthour_state = 0;
}

if(adjusthour == 1 && adjusthour_state == 0 && measure_process == 0)
{
  adjust_process = 1;
}
```

When the hours' button is pressed and the measure_process variable is set to 0, the condition will be true and the adjust_process variable will be set to 1.

The measure_process variable is used to signal that the system is monitoring temperature. When its value is 0, the system will allow the user to enter the time setting menu.

Therefore, after the adjust_process variable receives a value of 1, the system will enter the time adjustment condition. This code block is shown below.

```
//Adjust Hour
  if(adjust_process == 1)
  {
      lcd.clear();
      DS1307.getDate(DataTime);
      lcd.setCursor(0,0);
      lcd.print("Adjust Hour:");

      sprintf(times, "%02d:%02d", DataTime[4], DataTime[5]);

      lcd.setCursor(5,1);
      lcd.print(times);

    //Hour Adjust
    do
    {
      measure = digitalRead(Buttonmeasure);
      adjusthour = digitalRead(Buttonadjusthour);
      ok = digitalRead(Buttonok);

      if(measure == 0 && measure_state == 1)
      {
        measure_state = 0;
      }

      if(measure == 1 && measure_state == 0)
      {
        DataTime[4]++;

        if(DataTime[4] > 23)
        {
          DataTime[4] = 0;
        }

        measure_state = 1;

        sprintf(times, "%02d:%02d", DataTime[4], DataTime[5]);

        lcd.setCursor(5,1);
        lcd.print(times);
        DS1307.setDate(DataTime[0],DataTime[1],DataTime[2],DataTime[3],Dat
aTime[4],DataTime[5],00);
      }

      if(adjusthour == 0 && adjusthour_state == 1)
      {
        adjusthour_state = 0;
      }

      if(adjusthour == 1 && adjusthour_state == 0)
      {
        DataTime[5]++;

        if(DataTime[5] > 59)
        {
          DataTime[5] = 0;
        }
```

```
        sprintf(times, "%02d:%02d", DataTime[4], DataTime[5]);

        lcd.setCursor(5,1);
        lcd.print(times);

        DS1307.setDate(DataTime[0],DataTime[1],DataTime[2],DataTime[3],Dat
aTime[4],DataTime[5],00);

        adjusthour_state = 1;
      }

      if(ok == 1)
      {
        lcd.clear();

        DS1307.getDate(DataTime);
        sprintf(times, "%02d:%02d", DataTime[4], DataTime[5]);

        lcd.setCursor(0,0);
        lcd.print(times);
        lcd.setCursor(0,1);
        lcd.print("1-M    2-H    3-O");
        adjust_process = 0;
      }

    }while(ok != 1);
  }
```

When adjusting the hours, these buttons have their functions changed, that is, they are multifunction. This allows you to use a button for more than one function and reduce the complexity of the system.

In this way, the user will adjust the value of the hours and minutes and then save the data in the system when the Ok button is pressed.

As you can see, the system will read the 3 buttons, as shown below.

```
measure = digitalRead(Buttonmeasure);
adjusthour = digitalRead(Buttonadjusthour);
ok = digitalRead(Buttonok);
```

Note that the measure button (Buttonmeasure) has changed its function. It will now be used to adjust the hour values, as shown below.

The following two conditions are similar and are used to adjust the hours and minutes, as shown above.

```
if(measure == 0 && measure state == 1)
      {
        measure_state = 0;
```

```
      }

      if(measure == 1 && measure_state == 0)
      {
        DataTime[4]++;

        if(DataTime[4] > 23)
        {
          DataTime[4] = 0;
        }

        measure_state = 1;

        sprintf(times, "%02d:%02d", DataTime[4], DataTime[5]);

        lcd.setCursor(5,1);
        lcd.print(times);
        DS1307.setDate(DataTime[0],DataTime[1],DataTime[2],DataTime[3],Dat
aTime[4],DataTime[5],00);
      }

      if(adjusthour == 0 && adjusthour_state == 1)
      {
        adjusthour_state = 0;
      }

      if(adjusthour == 1 && adjusthour_state == 0)
      {
        DataTime[5]++;

        if(DataTime[5] > 59)
        {
          DataTime[5] = 0;
        }

        sprintf(times, "%02d:%02d", DataTime[4], DataTime[5]);

        lcd.setCursor(5,1);
        lcd.print(times);

        DS1307.setDate(DataTime[0],DataTime[1],DataTime[2],DataTime[3],Dat
aTime[4],DataTime[5],00);

        adjusthour_state = 1;
      }
```

Therefore, each time one of the two buttons is pressed, the value of positions 4 and 5 of the DataTime vector will be changed and secondly, these values will be saved in the DS1307 memory.

After the adjustments, the user must click on the Ok button, to finish the process. When this event occurs, the system will execute the following lines of code.

```
if(ok == 1)
      {
```

```
        lcd.clear();

        DS1307.getDate(DataTime);
        sprintf(times, "%02d:%02d", DataTime[4], DataTime[5]);

        lcd.setCursor(0,0);
        lcd.print(times);
        lcd.setCursor(0,1);
        lcd.print("1-M    2-H    3-O");
        adjust_process = 0;
      }
```

After that, when the button is pressed, the following portion of code will be executed.

```
if(measure == 0 && measure_state == 1)
  {
    measure_state = 0;
  }

  if(measure == 1 && measure_state == 0 && measure_process == 0)
  {
    measure_process = 1;
    measure_state = 1;

    if (SD.exists("temp.txt"))
    {
      Serial.println("Deleted the previous file!");
      SD.remove("temp.txt");
      myFile = SD.open("temp.txt", FILE_WRITE);
      Serial.println("Created the file!");
    }

    else

    {
      Serial.println("Created the file!");
      myFile = SD.open("temp.txt", FILE_WRITE);
      myFile.close();
    }

    delay(500);

    myFile.print("Hour: ");
    myFile.println("Temperature");

    DS1307.getDate(DataTime);
    actualMin = previousMin = DataTime[5];

    sprintf(times, "%02d:%02d", DataTime[4], DataTime[5]);

    lcd.clear();
    lcd.setCursor(5,0);
    lcd.print(times);

    lcd.setCursor(0,1);
    lcd.print("Temperature: ");
```

```
    lcd.setCursor(14,1);

    sensors.requestTemperatures();
    float TempSensor = sensors.getTempCByIndex(0);

    lcd.print(TempSensor);
  }
```

In the code portion above, the system will assign a value of 1 to the measure_process variable. It is responsible for allowing the data to be saved on the SD Card.

In addition, the system will check whether a text file with a data log exists or not. If there is a file, the system will delete and create a new one to store the data.

After that, the code flow will execute the following program block.

```
if(measure_process == 1)
  {
    updateTemp();

    byte contMin = 0, contHour = 0;
    DS1307.getDate(DataTime);
    actualMin = DataTime[5];

      if(actualMin != previousMin)
      {
        contMin++;
        previousMin = actualMin;
      }

      if(contMin == 5)
      {
        sprintf(times, "%02d:%02d ", DataTime[4], DataTime[5]);

        sensors.requestTemperatures();
        float TempSensor = sensors.getTempCByIndex(0);

        myFile.print(times);
        myFile.println(TempSensor);
        contMin = 0;
      }
      if(actualHour != previousHour)
      {
        contHour++;
        previousHour = actualHour;
      }

      if(contHour == 5)
      {
        myFile.close();
        lcd.clear();
        lcd.setCursor(5,0);
        lcd.print("Finished");
        lcd.setCursor(5,1);
        lcd.print("Process");
```

```
        measure_process = 0;
        contHour = 0;
      }
      if(ok == 1)
      {
        myFile.close();

        lcd.clear();
        lcd.setCursor(6,0);
        lcd.print("Stoped");
        lcd.setCursor(5,1);
        lcd.print("Process");

        measure_process = 0;
        delay(2000);

        lcd.clear();

        DS1307.getDate(DataTime);

        sprintf(times, "%02d:%02d", DataTime[4], DataTime[5]);

        lcd.setCursor(5,0);
        lcd.print(times);
        lcd.setCursor(0,1);
        lcd.print("1-M   2-H  3-O/P");
      }

  }
```

First, the updateTemp() function will be executed. It is similar to the updateHour() function; however, it displays the temperature every 1 minute.

After that, the system will collect the time data from the Real-Time Clock and store the current minute value in the currentMin variable. Then, it will check if the min variable has been changed, according to the condition presented below.

```
if(actualMin != previousMin)
{
   contMin++;
   previousMin = actualMin;
}
```

Therefore, if the current minute variable is different from the previous value, it means that a change in the value has occurred.

This way, the condition will be true and the value of the minute count will be increased (contMin) and the current value will be assigned to the variable previousMin, to store its previous value.

Therefore, when the value of this count is equal to 5, it means that 5 minutes have passed and the system must perform a new temperature reading and save the hour and temperature value in the SD Card log file.

```
if(contMin == 5)
{
   sprintf(times, "%02d:%02d ", DataTime[4], DataTime[5]);

   sensors.requestTemperatures();
   float TempSensor = sensors.getTempCByIndex(0);

   myFile.print(times);
   myFile.println(TempSensor);
   contMin = 0;
}
```

In this way, this process will be repeated until reaching the value of 5 hours of monitoring the patient's temperature with the temperature sensor with Arduino.

The code portion is shown below and is similar to the minute count, which was presented above.

```
      if(actualHour != previousHour)
      {
        contHour++;
        previousHour = actualHour;
      }

      if(contHour == 5)
      {
        myFile.close();
        lcd.clear();
        lcd.setCursor(5,0);
        lcd.print("Finished");
        lcd.setCursor(5,1);
        lcd.print("Process");
        measure_process = 0;
        contHour = 0;
      }
```

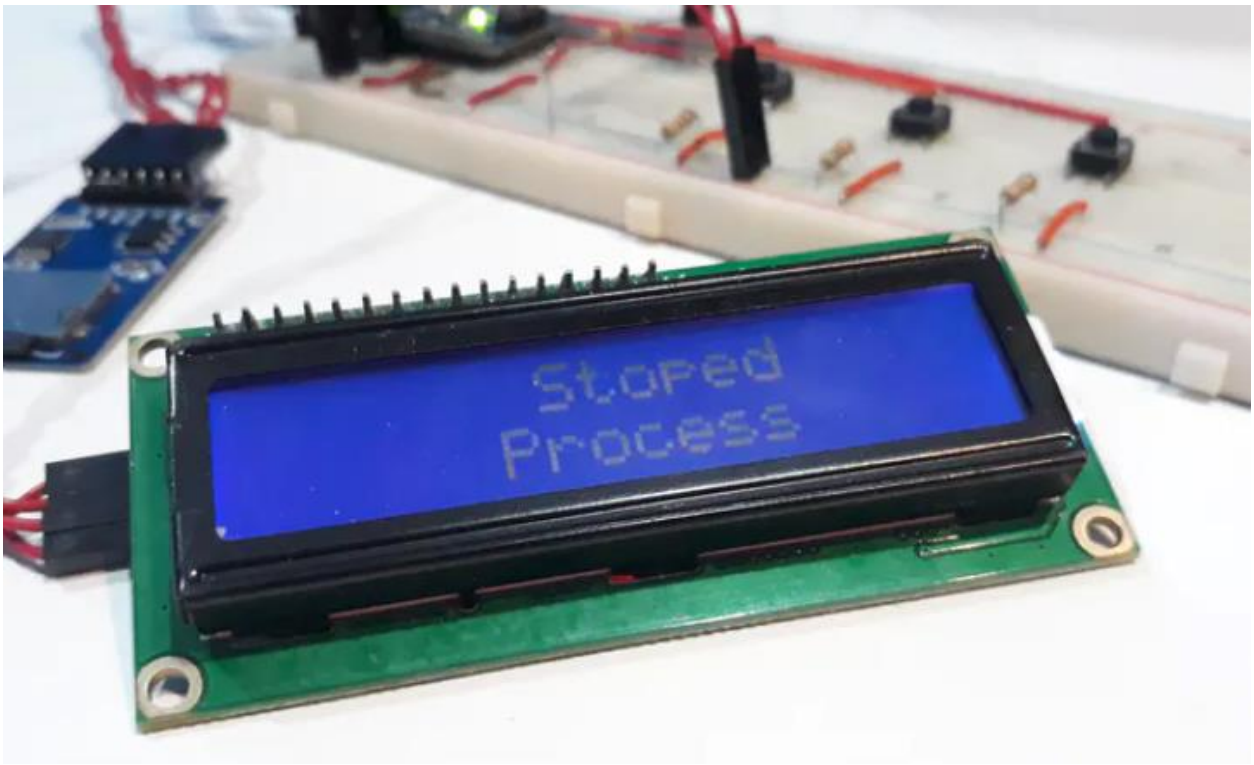
After reaching 5 hours of monitoring, the system will close the log file and present the message "Finished Process" to the user.

In addition, the user can press the Ok/Pause button in order to stop recording data. When this occurs, the following code block will be executed.

```
      if(ok == 1)
      {
        myFile.close();
```

```
        lcd.clear();
        lcd.setCursor(6,0);
        lcd.print("Stoped");
        lcd.setCursor(5,1);
        lcd.print("Process");

        measure_process = 0;
        delay(2000);

        lcd.clear();

        DS1307.getDate(DataTime);

        sprintf(times, "%02d:%02d", DataTime[4], DataTime[5]);

        lcd.setCursor(5,0);
        lcd.print(times);
        lcd.setCursor(0,1);
        lcd.print("1-M   2-H  3-O/P");
      }
```

Then, the system will close the file and present the message "Stoped Process", as shown below.



Accesing the SD Card Module data with Arduino

After the process of monitoring the Datalogger with the temperature sensor with Arduino, it is necessary to remove the memory card and access the data on the computer.

To view and analyze the data with better quality, export / copy all information of the text file to Excel.

After that, you can plot graphs and analyze the results obtained.

Conclusion

The Datalogger with a temperature sensor with Arduino allows us, in addition to measuring the temperature, to record information on the patient's temperature behavior over a period of time.

With these stored data, it is possible to analyze and understand how the temperature of the patient infected by COVID 19 behaves. In addition, it is possible to evaluate the temperature level and associate its value with the application of some type of medication.

Therefore, through these data, the Datalogger with temperature sensor for Arduino aims to assist doctors and nurses in the study of patients' behavior.

Price List

JLCPCB Customized PCB - 4.00$

JLCPCB Arduino Uno - 14.24 $

JLCPCB Printed Circuit Board - Project - $2.00/5 pieces

Maxim Integrated DS18B20 Programmable Resolution 1-Wire Digital Thermometer - $4.24

Arduino Nano R3 - $20.46

Jumper wires (generic) - $3.95

Adafruit RGB Backlight LCD - 16x2 - $12.95

SparkFun Pushbutton switch 12mm / 3 pieces - $0.81

Resistor 1k ohm - $0.067

Bitcraze Micro SD card deck - $8.00

Breadboard (generic) - $2.67

Total price - 73.387 $ = 311.20 Ron

Useful Links

- https://arduinomodules.info/
- https://learn.sparkfun.com/tutorials/what-is-an-arduino/all
- https://www.youtube.com/watch?v=nL34zDTPkcs
- https://www.youtube.com/watch?v=QO_Jlz1qpDw
- https://randomnerdtutorials.com/9-arduino-compatible-temperature-sensors-for-your-electronics-projects/