Introduction

The project my group chose is Non-contact Temperature Scanner. The scanner is required to show the accurate temperature, to alert the user when they have a fever, no contact to function, to log user information, be intuitive and show the temperature in Fahrenheit and Celsius. To implement this system, a couple of sensors are used. The block diagram is shown in Figure 1 and the interface definition is shown in Table 1. The second block I chose is the Tem-sensor block. The Tem-sensor block only includes a temperature sensor. The temperature sensor I used is the MLX90614 Thermal Sensor. MLX90614 uses laser or infrared to measure the temperature of an object. Its operating voltage is 3.6V to 5V. The temperature range that can be used for detection is $-40^\circ C$ to $85^\circ C$. 

![Figure 1: Block diagram](image-url)
<table>
<thead>
<tr>
<th>Interface Name</th>
<th>Type</th>
<th>Interface Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>User_data_in</td>
<td>Digital</td>
<td>Voltage: 3.6V~5V standard SPI interface Current: Maximum 2mA Detect the object’s temperature from $-40^\circ C$ ($-40^\circ F$) to $85^\circ C$($185^\circ F$).</td>
</tr>
<tr>
<td>Induction_data_in</td>
<td>Digital</td>
<td>Maximum Voltage: 3.8V Maximum Current: 2mA Detect the user's gesture and perform the tasks specified by the microcontroller.</td>
</tr>
<tr>
<td>Arduino_power_in</td>
<td>Electrical</td>
<td>Output Voltage: 5V Maximum Current: 20mA</td>
</tr>
<tr>
<td>Charge_data_in</td>
<td>Digital</td>
<td>Input voltage: 5V Charging cut-off voltage: 4.2V</td>
</tr>
<tr>
<td>Display_data_in</td>
<td>Digital</td>
<td>Interface Type: I2C interface VCC: Power + (DC 3.3 ~5v)</td>
</tr>
<tr>
<td>Sounds_data_in</td>
<td>Electrical/Sound</td>
<td>Input Voltage: 3-5V Max Current : 30mA</td>
</tr>
<tr>
<td>Data_storage</td>
<td>Digital/Electrical</td>
<td>Acquiring data from the Arduino Nano by transmitting the data to the SD card.</td>
</tr>
<tr>
<td>Temp_Display</td>
<td>Digital</td>
<td>This is where the temperature readings of Fahrenheit and Celsius will be illustrated.</td>
</tr>
<tr>
<td>Audible_Noise</td>
<td>Electrical/Audio</td>
<td>~80 dB continuous sound</td>
</tr>
<tr>
<td>Save_Data</td>
<td>Coding</td>
<td>This is the .txt file with recorded data.</td>
</tr>
</tbody>
</table>
Documentation

The temperature sensor I use is HiLetgo GY-906 MLX90614ESF Non-contact Infrared Temperature Sensor Module (Figure 3). By viewing the datasheet, the operating voltage is from 3.6V to 5V and the operating current is from 0.2mA to 200mA. The HiLetgo sensor board has 4 pins and the pin labels and corresponding description are shown in Table 2. The Arduino Nano will provide a 5V voltage to the temperature sensor, pin A4 and pin A5 will be connected to pin SDA and pin SCL on the sensor. To help testing the temperature sensor, I added a proximity sensor and an OLED display. The proximity sensor would capture the user’s gesture, once the proximity sensor was emitted, the temperature sensor would capture the user’s temperature and show the temperature on the OLED display in Fahrenheit and Celsius. The physical map connection is shown in Figure 2. The schematic of the temperature sensor is shown in Figure 4.

Bill of Materials

<table>
<thead>
<tr>
<th>Component</th>
<th>Bill</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino Nano</td>
<td>$13.99 3PCS</td>
</tr>
<tr>
<td>GY-906 MLX90614</td>
<td>$14.99</td>
</tr>
<tr>
<td>APDS-9960</td>
<td>$9.39 2PCS</td>
</tr>
<tr>
<td>OLED Display</td>
<td>$12.99</td>
</tr>
<tr>
<td>LED</td>
<td>$0.35 each</td>
</tr>
</tbody>
</table>

Figure 2: Physical map connection
Table 2

<table>
<thead>
<tr>
<th>Pin Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>Connect to ground.</td>
</tr>
<tr>
<td>VIN</td>
<td>Used to power the sensor. Must be 2.4-3.8V</td>
</tr>
<tr>
<td>SDA</td>
<td>( i^2C ) data.</td>
</tr>
<tr>
<td>SCL</td>
<td>( i^2C ) clock.</td>
</tr>
</tbody>
</table>

\( i^2C \) is the communication protocol used by MLX90614. The \( i^2C \) protocol allows multiple peripheral digital integrated circuits to communicate with one or more controller chips. The \( i^2C \) protocol enables asynchronous serial ports to obtain the same data transfer rate and clock rate, thereby supporting multi-controller systems.
## Examination

| User_data_in | Digital | Voltage: 3.6V~5V standard SPI interface  
|             |        | Current: Maximum 2mA  
|             |        | Detect the object’s temperature from $-40^\circ C$ ($-40^\circ F$) to $85^\circ C$ ($185^\circ F$). |

To make sure the sensor can work approximately, hookup the sensor and connect the physical circuit as shown in Figure 2. Since the maximum operating voltage is from 3.6V to 5V, connect the VIN pin to the 5v supply on the arduino. Use a multimeter to test the current when the circuit is working, and get the current 0.05mA (Figure 5). Therefore, the operating current value and operating voltage value of the interface definition have been proved.

![Figure 5: Operating current](image)
When the proximity sensor is emitted, which means there is a user in front of the system, the temperature begins to work and detect the user's temperature and the OLED display shows the temperature in Fahrenheit and Celsius. The testing code is shown in Figure 6 and Figure 7.

```c
#include "SoftwareSerial.h"
#include "Adafruit_APDS9960.h"
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Wire.h>
#include <Adafruit_MLX90614.h>

#define SCREEN_WIDTH 128  // OLED display width, in pixels
#define SCREEN_HEIGHT 32  // OLED display height, in pixels
#define OLED_RESET -1     // Reset pin # (or -1 if sharing Arduino reset pin)

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);  //Declaring the display name (display)
Adafruit_MLX90614 mlx = Adafruit_MLX90614();

Adafruit_APDS9960 apds;

SoftwareSerial mySoftwareSerial(10, 11);  // RX, TX

void setup() {
  mlx.begin();
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  //Start the OLED display
  display.clearDisplay();
  display.display();

  Serial.begin(9600);
  Wire.begin();
  // Configures pin D2 and D3 as outputs
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);

  if(!apds.begin())
  {
    Serial.println("Falha ao inicializar o dispositivo. Verifique as conexões!");
  }
  else
  {
    Serial.println("Dispositivo inicializado!");
  }
  apds.enableProximity(true);
  apds.enableGesture(true);
}
```

Figure 6: Testing code (1)
void loop()
{
  uint8_t gesture = apds.readGesture();
  display.clearDisplay();
  digitalWrite(3, LOW);
  digitalWrite(2, LOW);
  // If a user's gesture is captured
  if (gesture == APDS9960_UP || gesture == APDS9960_DOWN || gesture == APDS9960_LEFT || gesture == APDS9960_RIGHT)
  {
    digitalWrite(2, HIGH);
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0, 4);
    display.println("Target");
    display.setTextSize(2);
    display.setCursor(50, 0);
    display.println(mlx.readObjectTempC(), 1);
    display.setCursor(110, 0);
    display.println("C");
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0, 20);
    display.println("Target");
    display.setTextSize(2);
    display.setCursor(50, 17);
    display.println(mlx.readObjectTempF(), 1);
    display.setCursor(110, 17);
    display.println("F");
    display.display();
  } // If the detected temperature is higher than the setting temperature
  if (mlx.readObjectTempC() > 38 || mlx.readObjectTempF() > 100)
  {
    digitalWrite(3, HIGH);
    delay(1000);
  }
}

Figure 7: Testing code (2)

*Updated Demo Video link: [https://youtu.be/r9PMwFnEB8E](https://youtu.be/r9PMwFnEB8E)