Class: ECE 342
Name: Yuhao Su
Mentor: Karthik
Date: 4/17/2021

Introduction

The project my group chose is *Nn-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 Figure1 and the interface definition is shown in Table 1. The first I chose is the Prox-sensor block. The Prox_sensor block only includes a **proximity sensor**. The proximity sensor can detect the proximity or presence of nearby objects, so it does not require a physical contact device. When the scanner approaches the user, the proximity sensor is triggered and tells the system that it is time to work through a set program. Therefore, the proximity sensor is the basis for the realization of this system.

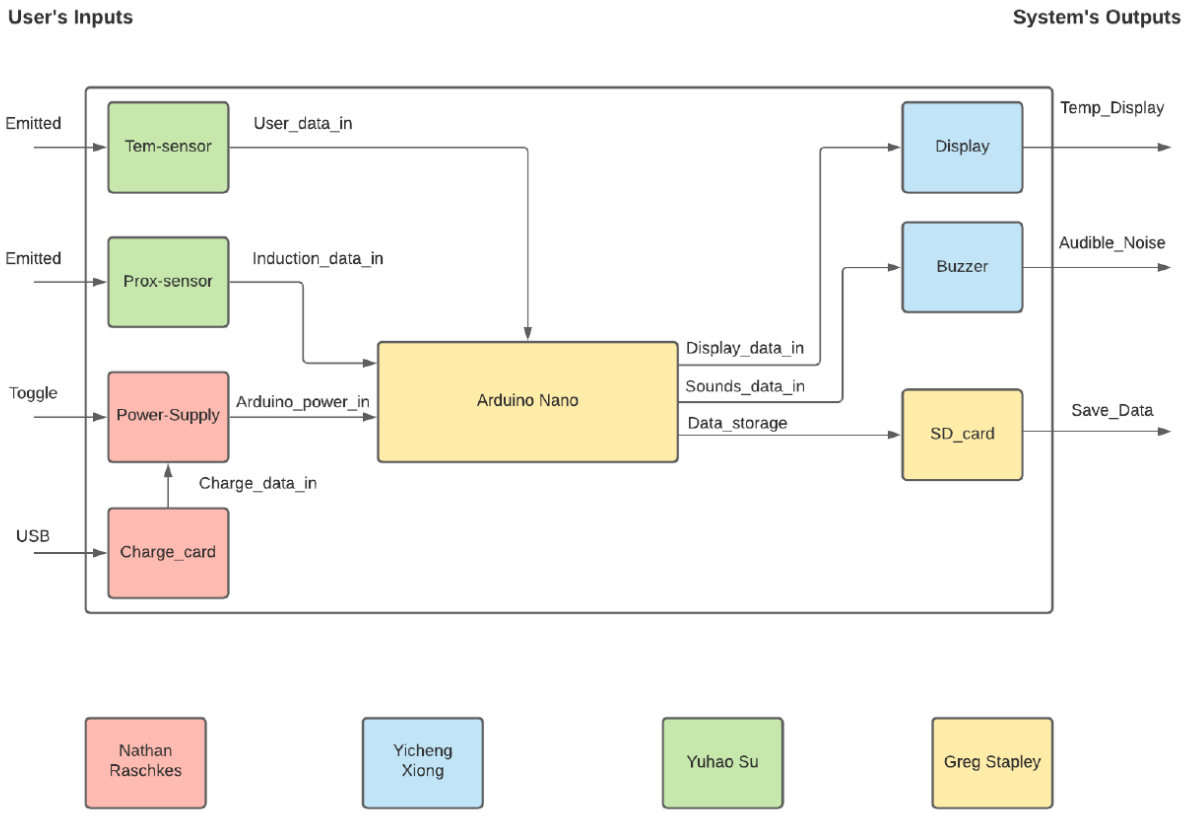

Figure 1: Block diagram

Table 1

Interface Name	Type	Interface Descriptions
User_data_in	Digital	Voltage: 4.5V~5.5V standard SPI interface Current: 0.2~200mA
Induction_data_in	Digital	Maximum Voltage: 3.8V Maximum Current: 200mA Detect the user's gesture and perform the tasks specified by the microcontroller.
Arduino_power_in	Electrical	Output Voltage: 5V Maximum Current: 20mA
Charge_data_in	Digital	Input voltage: 5V Charging cut-off voltage: 4.2V
Display_data_in	Digital	Interface Type: IIC interface VCC: Power + (DC 3.3 ~5v)
Sounds_data_in	Electrical/Sound	Input Voltage: 3-5V Max Current : 30mA
Data_storage	Digital/Electrical	Acquiring data from the Arduino Nano by transmitting the data to the SD card.
Temp_Display	Digital	This is where the temperature readings of Fahrenheit and Celsius will be illustrated.
Audible_Noise	Electrical/Audio	~80 dB continuous sound
Save_Data	Coding	This is the .txt file with recorded data.

Documentation

The proximity sensor I use is HiLetgo APDS-9960 (Figure 3). By viewing the datasheet, the maximum operating voltage and current are 3.8V and 200mA. The HiLetgo sensor board has 6 pins and the pin labels and corresponding description are shown in Table 2. The Arduino Nano will provide a 3.3V voltage to the proximity sensor, pin A4 and pin A5 will be connected to pin SDA and pin SCL on the sensor. The physical map connection is shown in Figure 2. The schematic of the sensor is shown in Figure 4, the custom PCB layout is shown in Figure 5 and the mechanical drawing is shown in Figure 6.

Material list

- Arduino Nano
- APDS-9960
- LED

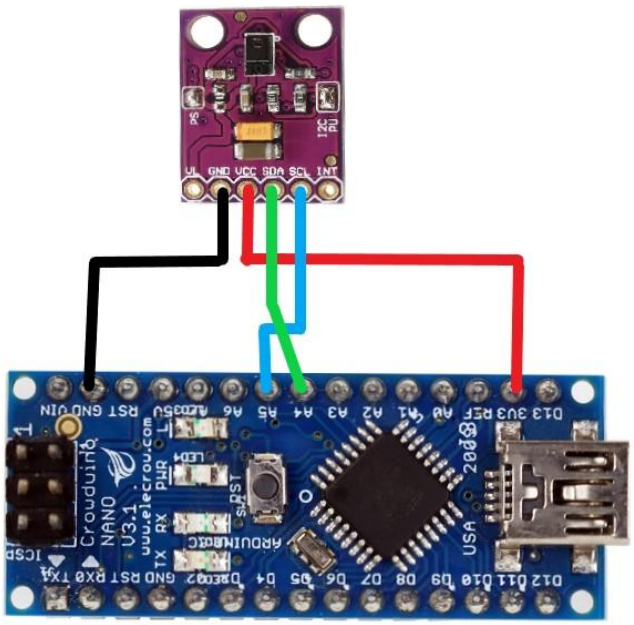

Figure 2: Physical map connection

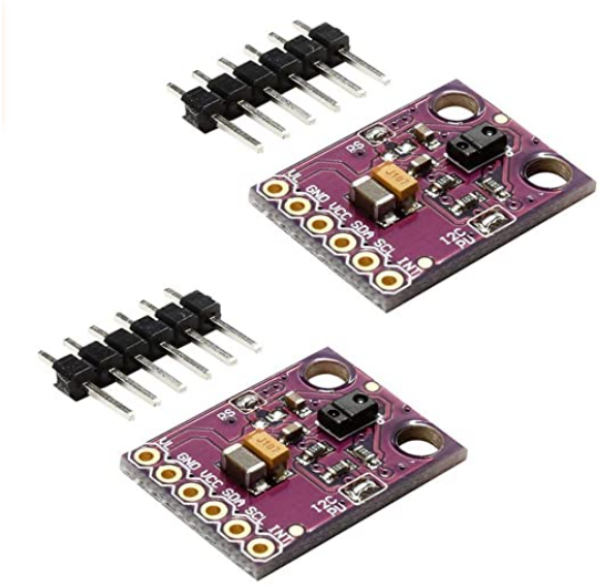

Figure 3: HiLetgo APDS-9960

Table 2

Pin Label	Description
VL	Optional power to the IR LED if PS jumper is disconnected. Must be 3.0 - 4.5V.
GND	Connect to ground.
VCC	Used to power the sensor. Must be 2.4-3.8V
SDA	I^2C data.
SCL	I^2C clock.
INT	External interrupt pin. Active LOW on interrupt event.

I^2C is the communication protocol used by APDS-9960. 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.
The operating range of the proximity sensor is from 4 inch to 8 inch (10-20cm). The size of the APDS-9960 gesture sensor module is 5.8*3.9*0.3 inches and weigh 0.18 Ounces.

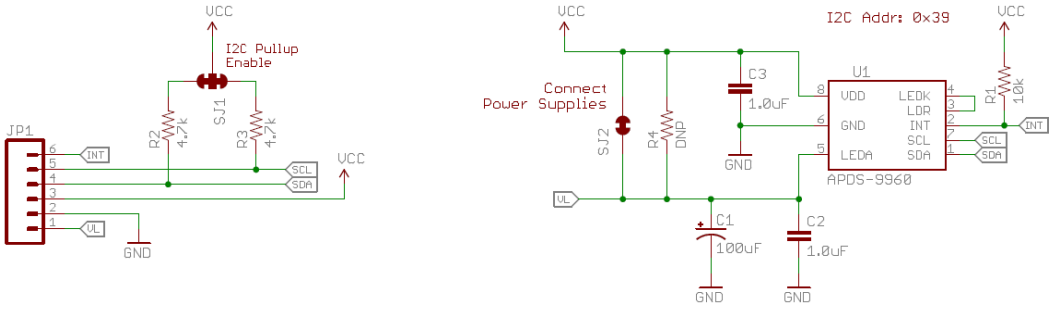

Figure 4: Schematic diagram of APDS-9960

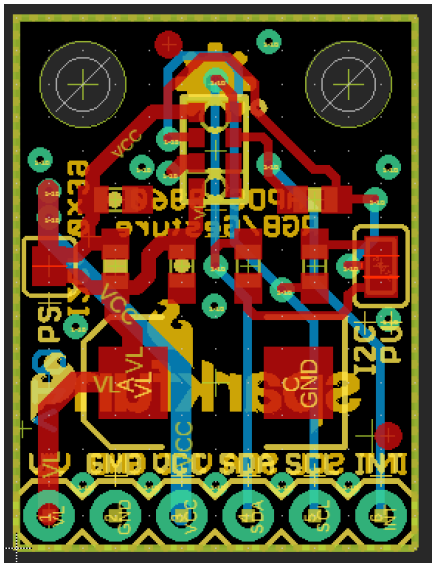

Figure 5: PCB layout of APDS-9960

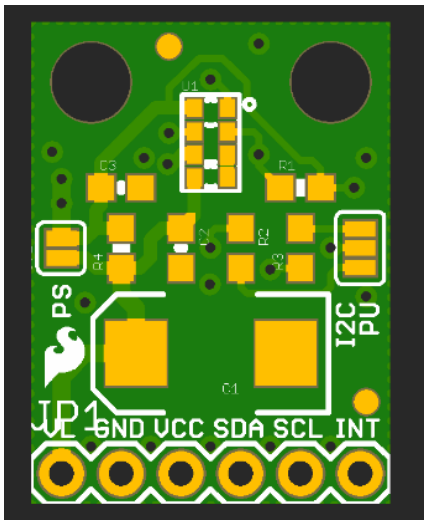

Figure 6: Mechanical drawing

Examination

Induction_data_in	Digital	Maximum Voltage: 3.8V Maximum Current: 200mA Detect the user's gesture and perform the tasks specified by the microcontroller.

To make sure the sensor can work approximately, hookup the sensor and connect the physical circuit as shown in Figure 7. Since the maximum operating voltage is 3.8V, connect the VCC pin to the 3.3v supply on the arduino. Use a multimeter to test the current when the circuit is working, and get the current 0.29mA (Figure 8). Therefore, the maximum current value and maximum voltage value of the interface definition have been proved.

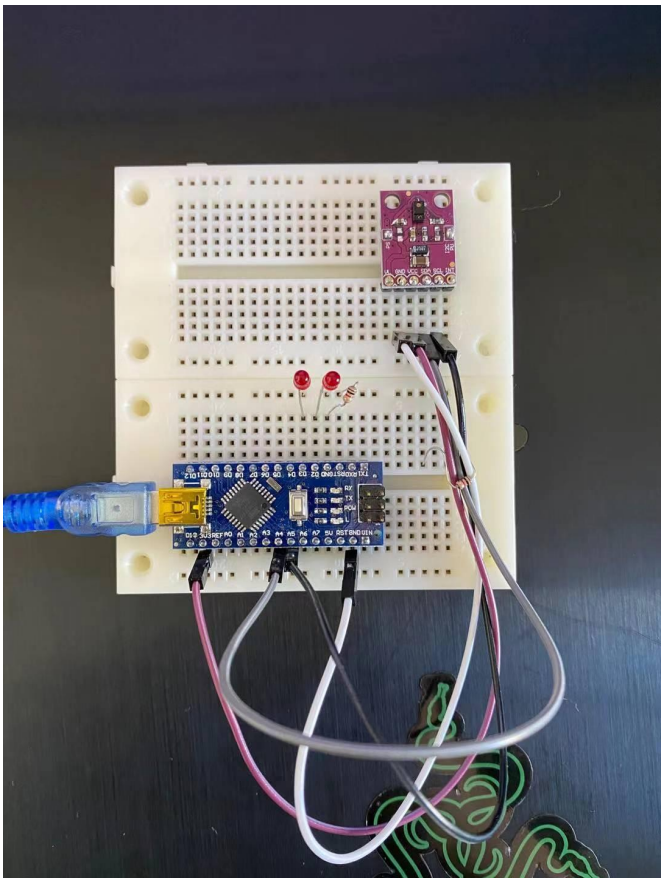

Figure 7: Physical circuit

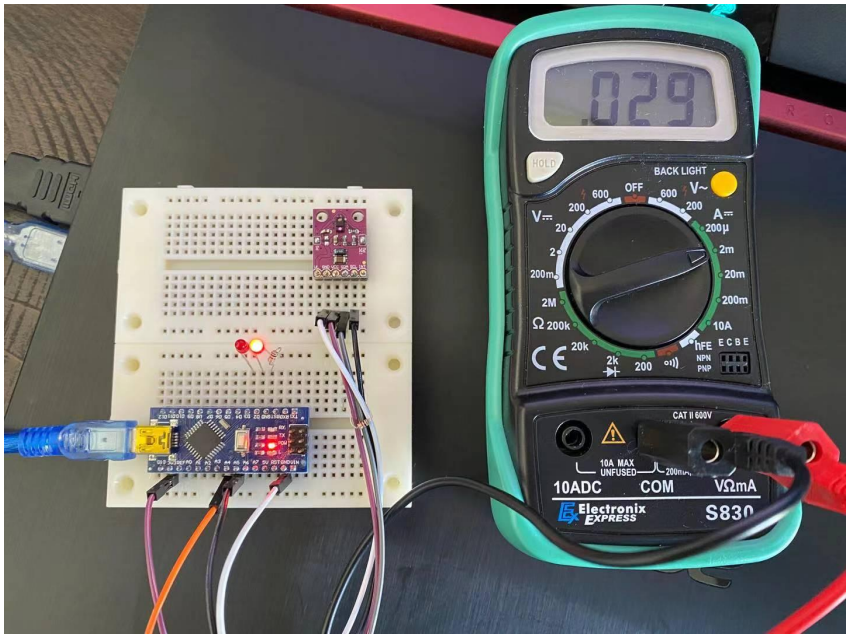

Figure 8: Operating current

To detect the user's gesture, connect the sensor with the microcontroller (Arduino Nano). Write a test code to make sure the sensor works as wished. Connect two LEDs with digital pin D2 and D3 as output. By coding, when the gesture goes to the right, the right LED is turned on, when the gesture goes to the left, the right LED is turned off. When the gesture goes up, the left LED is turned on, when the gesture goes down, the left LED is turned off. The example code is shown in Figure 9.

```
// Include the header files
#include "SoftwareSerial.h"
#include "Adafruit_APDS9960.h"

Adafruit_APDS9960 apds;

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

void setup()
{
  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);
}
void loop()
{
    uint8_t gesture = apds.readGesture();

          if(gesture == APDS9960_UP) // When the gesture goes up
          {
            digitalWrite(3, HIGH); // The LED conncted with D3 would be turned on
          }

          if(gesture == APDS9960_DOWN)// When the gesture goes down
          {
            digitalWrite(3, LOW); // The LED conncted with D3 would be turned off
          }

          if(gesture == APDS9960_LEFT)// When the gesture goes left
          {
            digitalWrite(2, LOW); // The LED conncted with D2 would be turned on
          }

          if(gesture == APDS9960_RIGHT)// When the gesture goes right
          {
            digitalWrite(2, HIGH); // The LED conncted with D3 would be turned off
          }
```

Figure 9