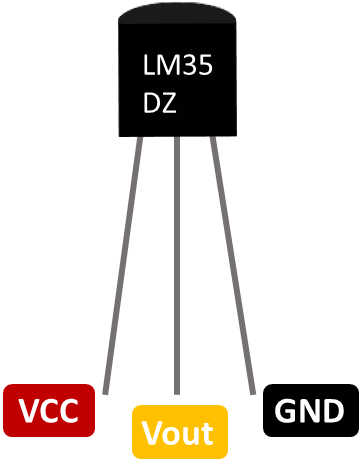
**EMBEDDED**

**TEMPERATURE SENSOR:-**

**LM 35**



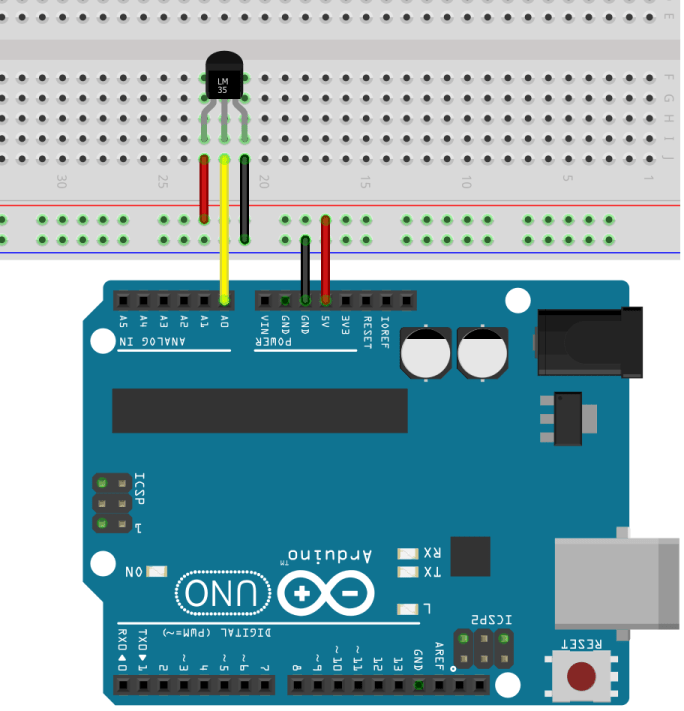
### **Pin Configuration:**

|  |  |  |
| --- | --- | --- |
| **Pin Number** | **Pin Name** | **Description** |
| 1 | Vcc | Input voltage is +5V for typical applications |
| 2 | Analog Out | There will be increase in 10mV for raise of every 1°C. Can range from -1V(-55°C) to 6V(150°C) |
| 3 | Ground | Connected to ground of circuit |

**Features**

* Calibrated Directly in Celsius (Centigrade)
* Linear + 10-mV/°C Scale Factor
* 0.5°C Ensured Accuracy (at 25°C)
* Rated for Full −55°C to 150°C Range
* Suitable for Remote Applications
* Low-Cost Due to Wafer-Level Trimming
* Operates From 4 V to 30 V
* Less Than 60-μA Current Drain
* Low Self-Heating, 0.08°C in Still Air
* Non-Linearity Only ±¼°C Typical
* Low-Impedance Output, 0.1 Ω for 1-mA Load

### LM35 with Arduino:-



### **LM35 Temperature Sensor Applications:-**

* Measuring temperature of a particular environment
* Providing thermal shutdown for a circuit/component
* Monitoring Battery Temperature
* Measuring Temperatures for HVAC applications.

Why LM35 would be a suitable sensor for our project as compared to other sensors?

* The LM35 is linear temperature sensors that output a voltage proportional to the temperature value.
* It requires very little current to operate, about 60uA. This results in very low self-heating (around 0.08ºC in still air), which means that the temperature measurements won’t be affected by the sensor itself.
* It has a large range when compared to DHT11, DHT22.
* To read the temperature from these sensors we just need to read the sensor’s output voltage using an analog pin.
* For eg: On connecting it with Arduino, all one needs to do to get temp readings to 2 decimal points is to use the analogRead()

**CODE:-int val;**

int tempPin = 1;

void setup()

{

Serial.begin(9600);

}

void loop()

{

val = analogRead(tempPin);

float mv = ( val/1024.0)\*5000;

float cel = mv/10;

float farh = (cel\*9)/5 + 32;

Serial.print("TEMPRATURE =");

Serial.print(cel);

Serial.print("\*C");

Serial.println();

delay(2000);

if (cel<37)

Serial.print ("Temperature is okay \n");

else

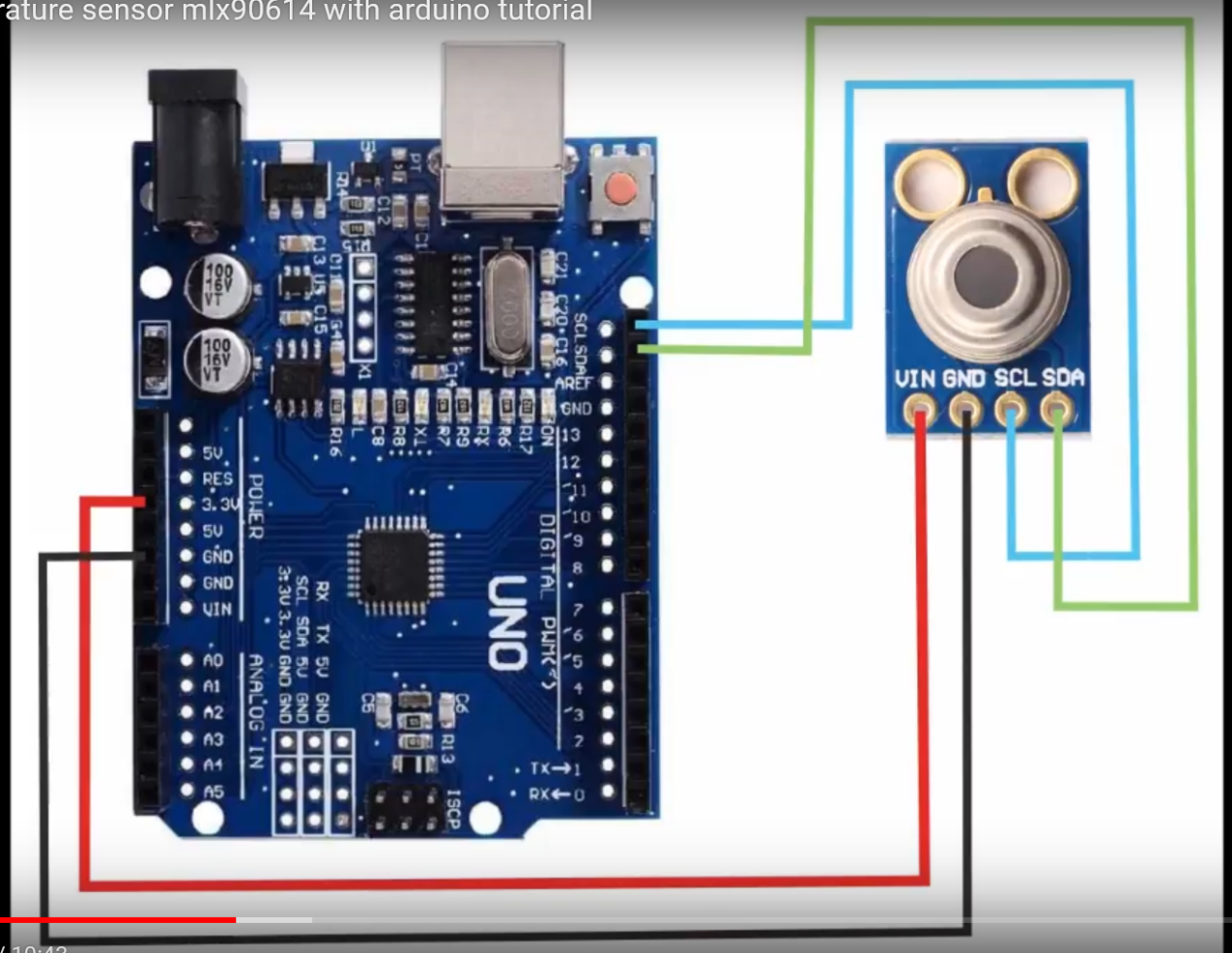
Serial.print("Temperature is high might have fever \n");

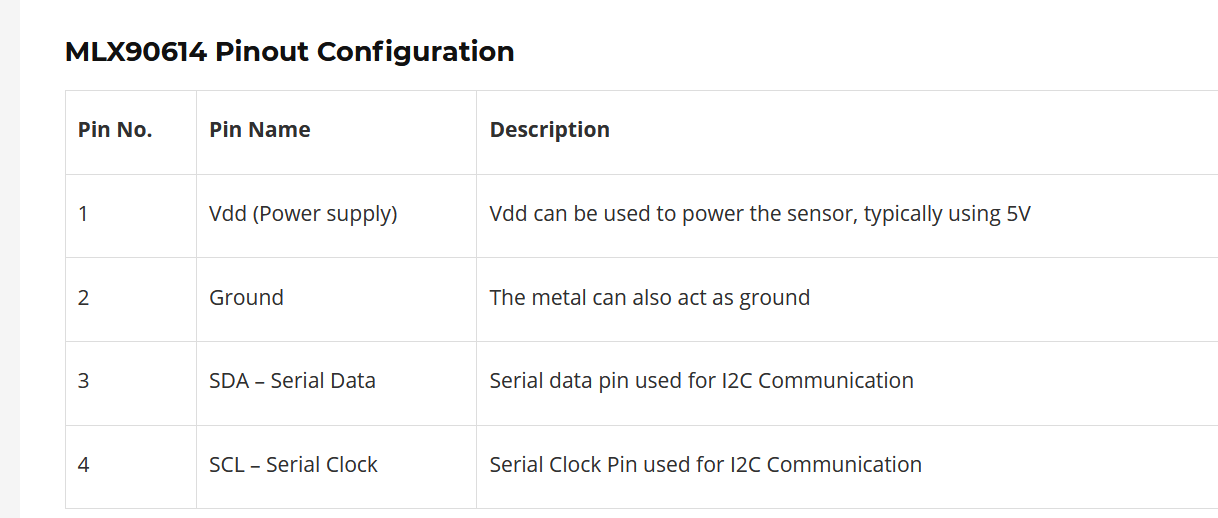
delay (2000);

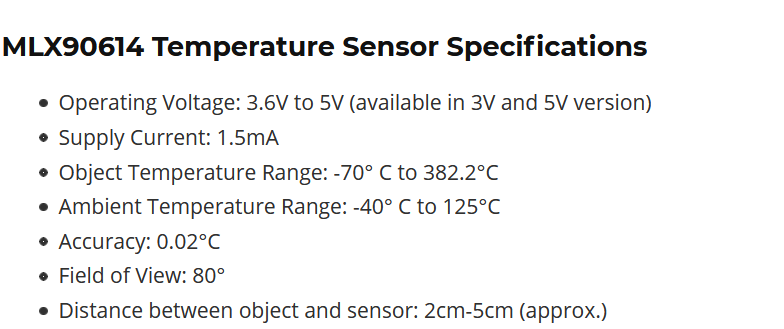
}

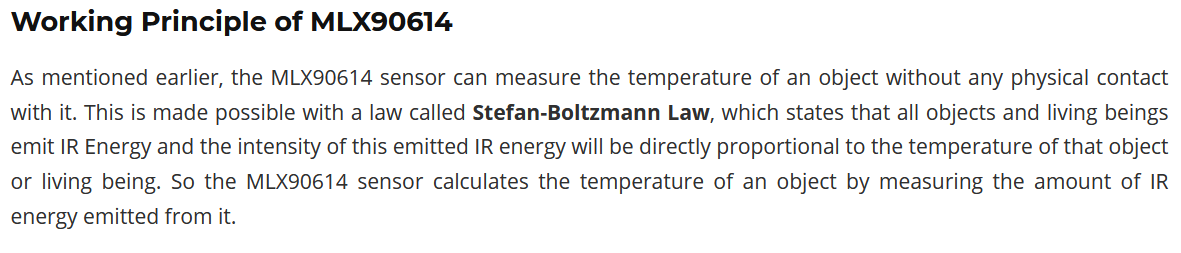
**OR,**

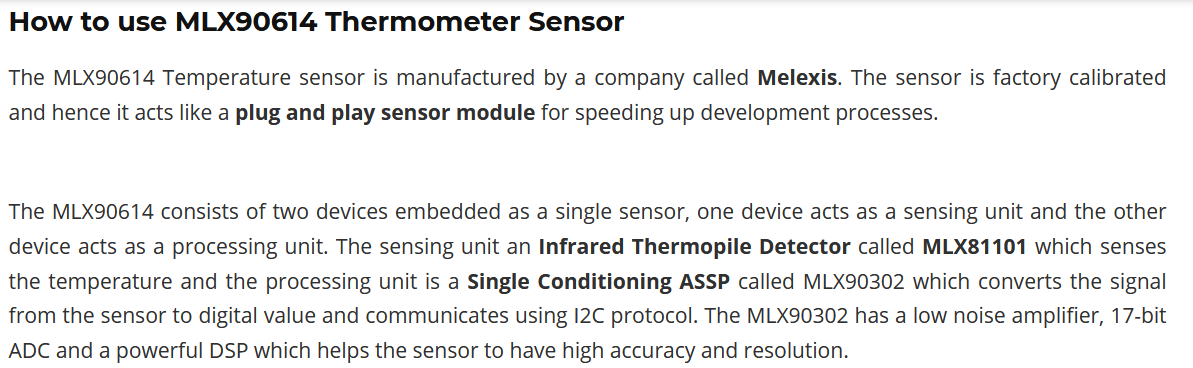
**MLX-90614**(More preferable)

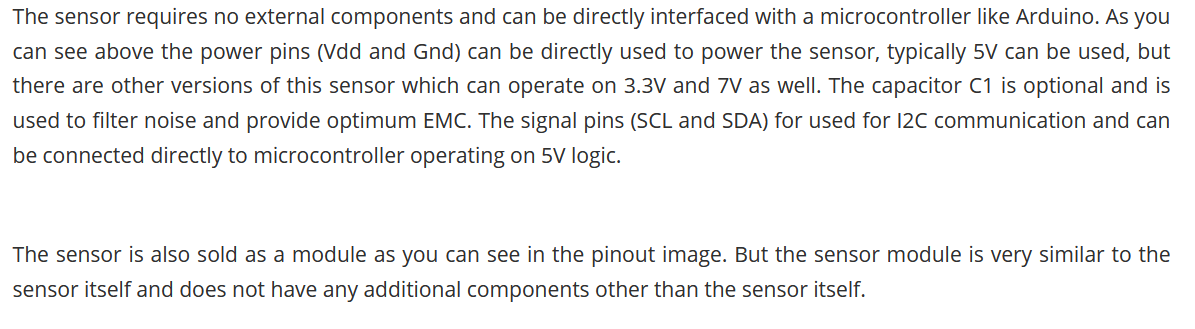
****

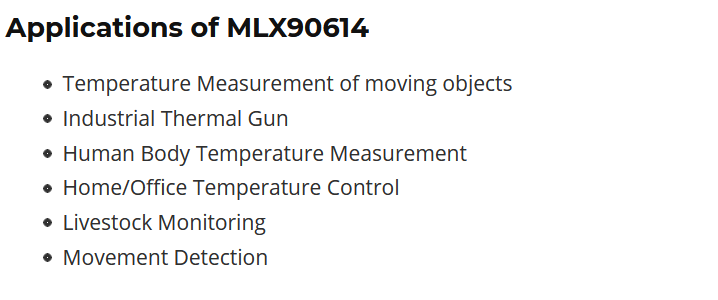
****

****

****

****

****

****

**<https://youtu.be/NzMncrfZAmU>**

**Code:-**

**#include <LiquidCrystal\_I2C.h>**

**#include <Wire.h>**

**#include <Adafruit\_MLX90614.h>**

**#define I2C\_ADDR 0x27 //I2C adress, you should use the code to scan the adress first (0x27) here**

**#define BACKLIGHT\_PIN 3 // Declaring LCD Pins**

**#define En\_pin 2**

**#define Rw\_pin 1**

**#define Rs\_pin 0**

**#define D4\_pin 4**

**#define D5\_pin 5**

**#define D6\_pin 6**

**#define D7\_pin 7**

**LiquidCrystal\_I2C lcd(I2C\_ADDR,En\_pin,Rw\_pin,Rs\_pin,D4\_pin,D5\_pin,D6\_pin,D7\_pin);**

**Adafruit\_MLX90614 mlx = Adafruit\_MLX90614();**

**void setup() {**

**mlx.begin();**

**lcd.begin (16,2);**

**lcd.setBacklightPin(BACKLIGHT\_PIN,POSITIVE);**

**lcd.setBacklight(HIGH); //Lighting backlight**

**lcd.home ();**

**}**

**PULSE RATE SENSOR:-**

**MAX30102:-**

**Benefits and Features**

● Heart-Rate Monitor and Pulse Oximeter Sensor in LED Reflective Solution

● Tiny 5.6mm x 3.3mm x 1.55mm 14-Pin Optical Module

• Integrated Cover Glass for Optimal, Robust

Performance

● Ultra-Low Power Operation for Mobile Devices

• Programmable Sample Rate and LED Current for

Power Savings

• Low-Power Heart-Rate Monitor (< 1mW)

• Ultra-Low Shutdown Current (0.7µA, typ)

● Fast Data Output Capability

• High Sample Rates

● Robust Motion Artifact Resilience

• High SNR

● -40°C to +85°C Operating Temperature Range

**Applications**

● Wearable Devices

● Fitness Assistant Devices

● Smartphones

● Tablets

**Pin description:**

**VIN**: main power supply input, 1.8V~5V  
**3-bit pad:** selects the pull-up level of the bus, depending on the pin master voltage, optional 1.8V or 3.3V end (this end contains 3.3V and above)  
**SCL:** the clock connected to the I2C bus  
**SDA:** data connected to the I2C bus  
**INT:** interrupt pin of the MAX30102 chip  
**RD:** RED LED ground terminal of MAX30102 chip, generally not connected  
**IRD:** the IR LED ground of the MAX30102 chip is generally not connected  
**GND:** ground wire

**Principle description:**

**Photodissolution method:**

The measurement of pulse and blood oxygen saturation is performed by using human tissue to cause different light transmittance when the blood vessel beats.  
Light source:  
A specific wavelength of light-emitting diode selective for oxyhemoglobin (HbO2) and hemoglobin (Hb) in arterial blood.

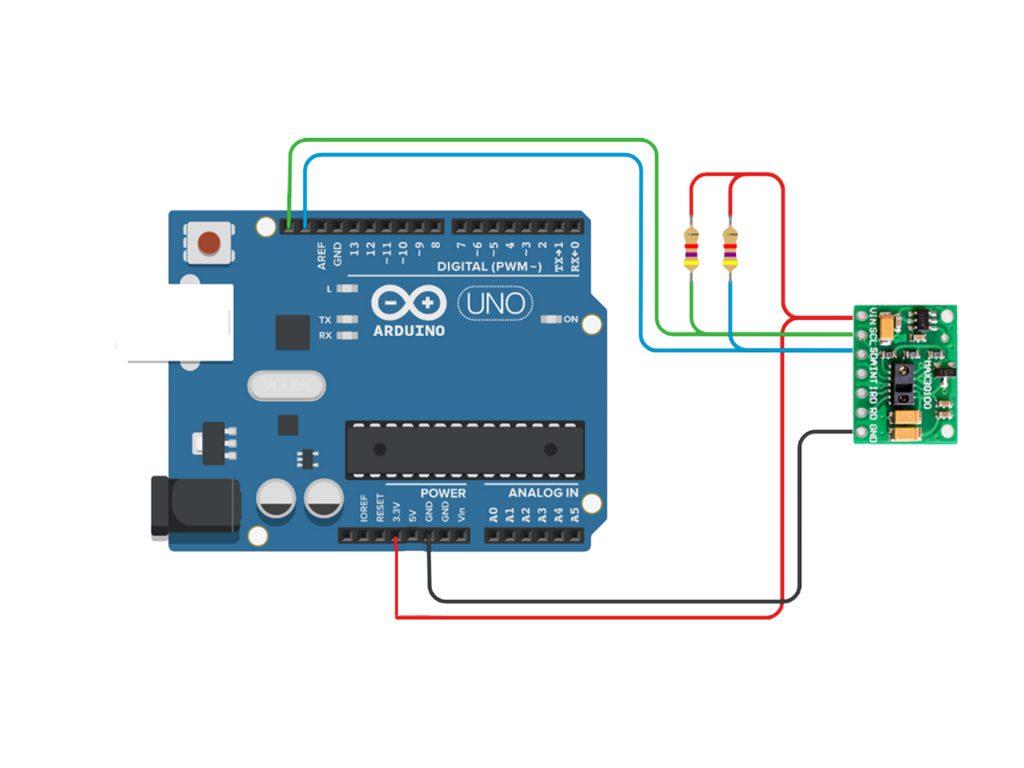
**Light transmittance is converted into an electrical signal:**

The change in the volume of the arterial pulsation causes the light transmittance of the light to change. At this time the light reflected by the human tissue is received by the photoelectric transducer, converted into an electrical signal, and amplified and output.

Why MAX30102 would be suitable as compared to other sensors?

* The MAX30102 integrates red and IR LEDs to modulate LED pulses for oxygen saturation (SpO2) and heart rate measurements.
* Space savings: Maintains a very small solution size without sacrificing optical or electrical performance; Integrates internal LEDs, photodetectors, optical elements, and low-noise electronics with ambient light rejection

**MAX30102 WITH ARDUINO:-**



**CODE:-**

#include <Adafruit\_GFX.h> //OLED libraries

#include <Adafruit\_SSD1306.h>

#include <Wire.h>

#include "MAX30105.h" //MAX3010x library

#include "heartRate.h" //Heart rate calculating algorithm

MAX30105 particleSensor;

const byte RATE\_SIZE = 4; //Increase this for more averaging. 4 is good.

byte rates[RATE\_SIZE]; //Array of heart rates

byte rateSpot = 0;

long lastBeat = 0; //Time at which the last beat occurred

float beatsPerMinute;

int beatAvg;

#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)

static const unsigned char PROGMEM logo2\_bmp[] =

{ 0x03, 0xC0, 0xF0, 0x06, 0x71, 0x8C, 0x0C, 0x1B, 0x06, 0x18, 0x0E, 0x02, 0x10, 0x0C, 0x03, 0x10, //Logo2 and Logo3 are two bmp pictures that display on the OLED if called

0x04, 0x01, 0x10, 0x04, 0x01, 0x10, 0x40, 0x01, 0x10, 0x40, 0x01, 0x10, 0xC0, 0x03, 0x08, 0x88,

0x02, 0x08, 0xB8, 0x04, 0xFF, 0x37, 0x08, 0x01, 0x30, 0x18, 0x01, 0x90, 0x30, 0x00, 0xC0, 0x60,

0x00, 0x60, 0xC0, 0x00, 0x31, 0x80, 0x00, 0x1B, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x04, 0x00, };

static const unsigned char PROGMEM logo3\_bmp[] =

{ 0x01, 0xF0, 0x0F, 0x80, 0x06, 0x1C, 0x38, 0x60, 0x18, 0x06, 0x60, 0x18, 0x10, 0x01, 0x80, 0x08,

0x20, 0x01, 0x80, 0x04, 0x40, 0x00, 0x00, 0x02, 0x40, 0x00, 0x00, 0x02, 0xC0, 0x00, 0x08, 0x03,

0x80, 0x00, 0x08, 0x01, 0x80, 0x00, 0x18, 0x01, 0x80, 0x00, 0x1C, 0x01, 0x80, 0x00, 0x14, 0x00,

0x80, 0x00, 0x14, 0x00, 0x80, 0x00, 0x14, 0x00, 0x40, 0x10, 0x12, 0x00, 0x40, 0x10, 0x12, 0x00,

0x7E, 0x1F, 0x23, 0xFE, 0x03, 0x31, 0xA0, 0x04, 0x01, 0xA0, 0xA0, 0x0C, 0x00, 0xA0, 0xA0, 0x08,

0x00, 0x60, 0xE0, 0x10, 0x00, 0x20, 0x60, 0x20, 0x06, 0x00, 0x40, 0x60, 0x03, 0x00, 0x40, 0xC0,

0x01, 0x80, 0x01, 0x80, 0x00, 0xC0, 0x03, 0x00, 0x00, 0x60, 0x06, 0x00, 0x00, 0x30, 0x0C, 0x00,

0x00, 0x08, 0x10, 0x00, 0x00, 0x06, 0x60, 0x00, 0x00, 0x03, 0xC0, 0x00, 0x00, 0x01, 0x80, 0x00 };

void setup() {

display.begin(SSD1306\_SWITCHCAPVCC, 0x3C); //Start the OLED display

display.display();

delay(3000);

// Initialize sensor

particleSensor.begin(Wire, I2C\_SPEED\_FAST); //Use default I2C port, 400kHz speed

particleSensor.setup(); //Configure sensor with default settings

particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low to indicate sensor is running

}

void loop() {

long irValue = particleSensor.getIR(); //Reading the IR value it will permit us to know if there's a finger on the sensor or not

//Also detecting a heartbeat

if(irValue > 7000){ //If a finger is detected

display.clearDisplay(); //Clear the display

display.drawBitmap(5, 5, logo2\_bmp, 24, 21, WHITE); //Draw the first bmp picture (little heart)

display.setTextSize(2); //Near it display the average BPM you can display the BPM if you want

display.setTextColor(WHITE);

display.setCursor(50,0);

display.println("BPM");

display.setCursor(50,18);

display.println(beatAvg);

display.display();

if (checkForBeat(irValue) == true) //If a heart beat is detected

{

display.clearDisplay(); //Clear the display

display.drawBitmap(0, 0, logo3\_bmp, 32, 32, WHITE); //Draw the second picture (bigger heart)

display.setTextSize(2); //And still displays the average BPM

display.setTextColor(WHITE);

display.setCursor(50,0);

display.println("BPM");

display.setCursor(50,18);

display.println(beatAvg);

display.display();

tone(3,0); //And tone the buzzer for a 100ms you can reduce it it will be better

delay(100);

noTone(3); //Deactivate the buzzer to have the effect of a "bip"

//We sensed a beat!

long delta = millis() - lastBeat; //Measure duration between two beats

lastBeat = millis();

beatsPerMinute = 60 / (delta / 1000.0); //Calculating the BPM

if (beatsPerMinute < 255 && beatsPerMinute > 20) //To calculate the average we strore some values (4) then do some math to calculate the average

{

rates[rateSpot++] = (byte)beatsPerMinute; //Store this reading in the array

rateSpot %= RATE\_SIZE; //Wrap variable

//Take average of readings

beatAvg = 0;

for (byte x = 0 ; x < RATE\_SIZE ; x++)

beatAvg += rates[x];

beatAvg /= RATE\_SIZE;

}

}

}

if (irValue < 7000){ //If no finger is detected it inform the user and put the average BPM to 0 or it will be stored for the next measure

beatAvg=0;

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(30,5);

display.println("Please Place ");

display.setCursor(30,15);

display.println("your finger ");

display.display();

noTone(3);

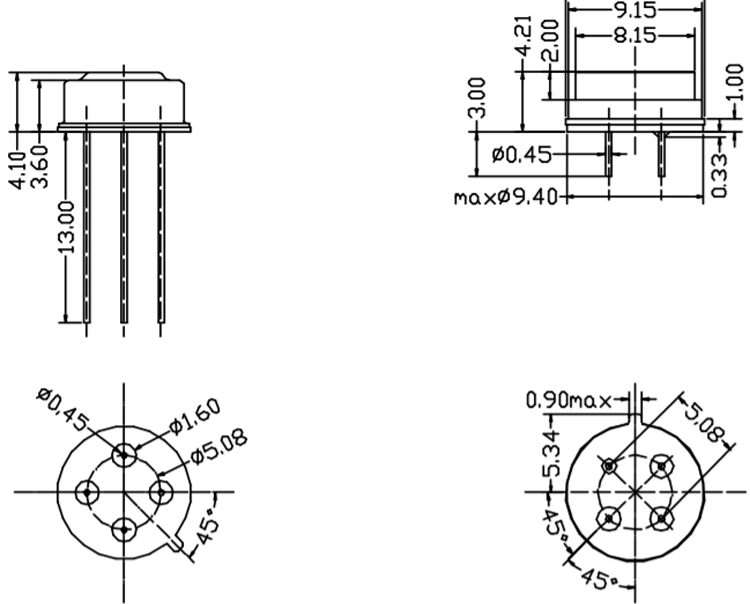
}

}

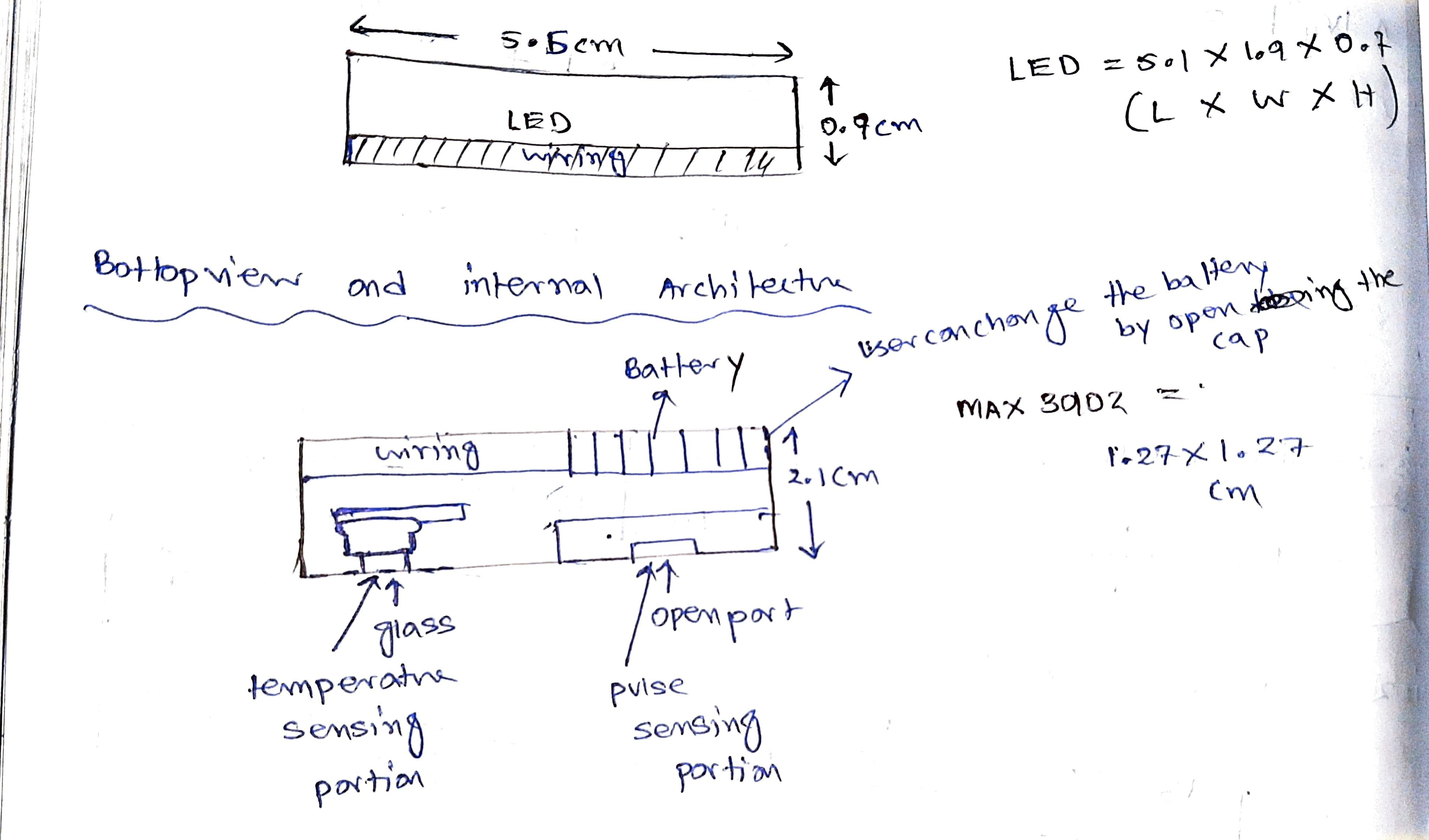
**GADGET:-**

**MLX90614 DIMENSION:-**



**Top View:-**





**COMPONENT:-**

1. Strap
2. Battery case
3. MLX 90614
4. MAX 30102
5. Wires
6. 4-Bit 7-Segment 0.56" Red LED Digital Display Tube For DIY or Arduino
7. Casings.
8. Panasonic CR2032 3V Lithium Coin Battery,(5 nos)