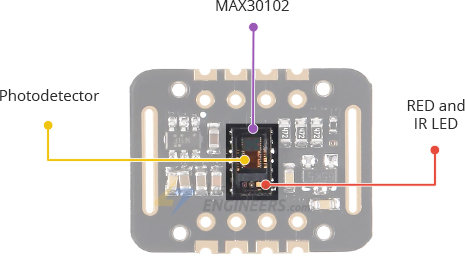
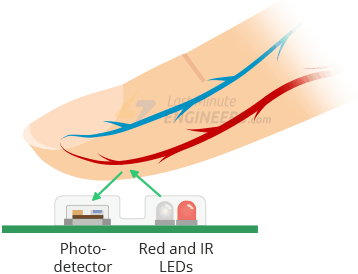
**1 MAX30102** **sensor**

**1.1 Introduction:**

The MAX30102 can be used as an oximeter by measuring the oxygen concentration in the blood (SpO2 percentage) and also can be used as heart rate sensor (beats per minute). The device has two LEDs, one emitting red light, another emitting infrared light. For pulse rate, only the infrared light is needed. Both the red light and infrared light is used to measure oxygen levels in the blood.



*How does the max 30102 pulse oximeter work?*

Graphical user interface, text, application

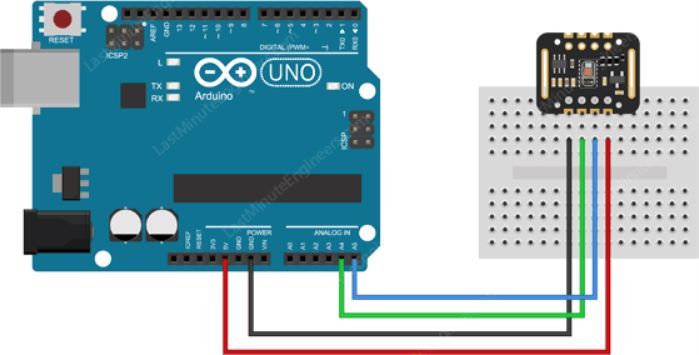
Description automatically generated When the heart pumps blood, there is an increase in oxygenated blood as a result of having more blood. As the heart relaxes, the volume of oxygenated blood also decreases. By knowing the time between the increase and decrease of oxygenated blood, the pulse rate is determined.

**1.2 Circuit Diagram:**

**Diagram, schematic

Description automatically generated**

A picture containing text

Description automatically generated

***1.3* SpO2**

Oxygen saturation is an essential element in the management and understanding of patient care. Oxygen is tightly regulated within the body because hypoxemia can lead to many acute adverse effects on individual organ systems. These include the brain, heart, and kidneys

**1.3.1 SpO2 Code in Arduino:**

#include <Wire.h>

#include "MAX30105.h"

#include "spo2\_algorithm.h"

MAX30105 particleSensor;

#define MAX\_BRIGHTNESS 255

#if defined(\_AVR\_ATmega328P) || defined(AVR\_ATmega168\_)

//Arduino Uno doesn't have enough SRAM to store 100 samples of IR led data and red led data in 32-bit format

//To solve this problem, 16-bit MSB of the sampled data will be truncated. Samples become 16-bit data.

uint16\_t irBuffer[100]; //infrared LED sensor data

uint16\_t redBuffer[100]; //red LED sensor data

#else

uint32\_t irBuffer[100]; //infrared LED sensor data

uint32\_t redBuffer[100]; //red LED sensor data

#endif

int32\_t bufferLength; //data length

int32\_t spo2; //SPO2 value

int8\_t validSPO2; //indicator to show if the SPO2 calculation is valid

int32\_t heartRate; //heart rate value

int8\_t validHeartRate; //indicator to show if the heart rate calculation is valid

byte pulseLED = 11; //Must be on PWM pin

byte readLED = 13; //Blinks with each data read

void setup()

{

Serial.begin(115200); // initialize serial communication at 115200 bits per second:

pinMode(pulseLED, OUTPUT);

pinMode(readLED, OUTPUT);

// Initialize sensor

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

{

Serial.println(F("MAX30105 was not found. Please check wiring/power."));

while (1);

}

Serial.println(F("Attach sensor to finger with rubber band. Press any key to start conversion"));

while (Serial.available() == 0) ; //wait until user presses a key

Serial.read();

byte ledBrightness = 60; //Options: 0=Off to 255=50mA

byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32

byte ledMode = 2; //Options: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green

byte sampleRate = 100; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200

int pulseWidth = 411; //Options: 69, 118, 215, 411

int adcRange = 4096; //Options: 2048, 4096, 8192, 16384

particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth, adcRange); //Configure sensor with these settings

}

void loop()

{

bufferLength = 100; //buffer length of 100 stores 4 seconds of samples running at 25sps

//read the first 100 samples, and determine the signal range

for (byte i = 0 ; i < bufferLength ; i++)

{

while (particleSensor.available() == false) //do we have new data?

particleSensor.check(); //Check the sensor for new data

redBuffer[i] = particleSensor.getRed();

irBuffer[i] = particleSensor.getIR();

particleSensor.nextSample(); //We're finished with this sample so move to next sample

Serial.print(F("red="));

Serial.print(redBuffer[i], DEC);

Serial.print(F(", ir="));

Serial.println(irBuffer[i], DEC);

}

//calculate heart rate and SpO2 after first 100 samples (first 4 seconds of samples)

maxim\_heart\_rate\_and\_oxygen\_saturation(irBuffer, bufferLength, redBuffer, &spo2, &validSPO2, &heartRate, &validHeartRate);

//Continuously taking samples from MAX30102. Heart rate and SpO2 are calculated every 1 second

while (1)

{

//dumping the first 25 sets of samples in the memory and shift the last 75 sets of samples to the top

for (byte i = 25; i < 100; i++)

{

redBuffer[i - 25] = redBuffer[i];

irBuffer[i - 25] = irBuffer[i];

}

//take 25 sets of samples before calculating the heart rate.

for (byte i = 75; i < 100; i++)

{

while (particleSensor.available() == false) //do we have new data?

particleSensor.check(); //Check the sensor for new data

digitalWrite(readLED, !digitalRead(readLED)); //Blink onboard LED with every data read

redBuffer[i] = particleSensor.getRed();

irBuffer[i] = particleSensor.getIR();

particleSensor.nextSample(); //We're finished with this sample so move to next sample

//send samples and calculation result to terminal program through UART

Serial.print(F("red="));

Serial.print(redBuffer[i], DEC);

Serial.print(F(", ir="));

Serial.print(irBuffer[i], DEC);

Serial.print(F(", HR="));

Serial.print(heartRate, DEC);

Serial.print(F(", HRvalid="));

Serial.print(validHeartRate, DEC);

Serial.print(F(", SPO2="));

Serial.print(spo2, DEC);

Serial.print(F(", SPO2Valid="));

Serial.println(validSPO2, DEC);

}

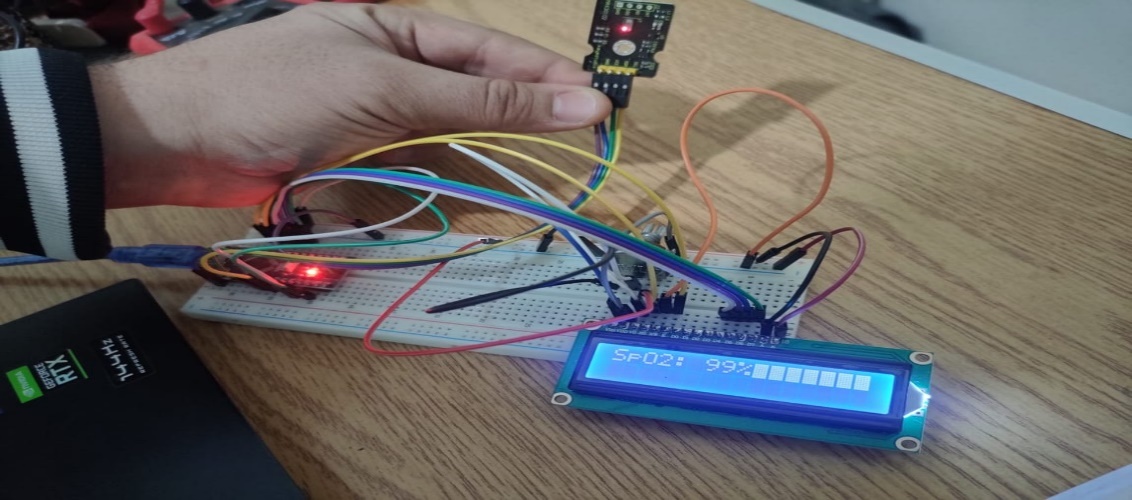
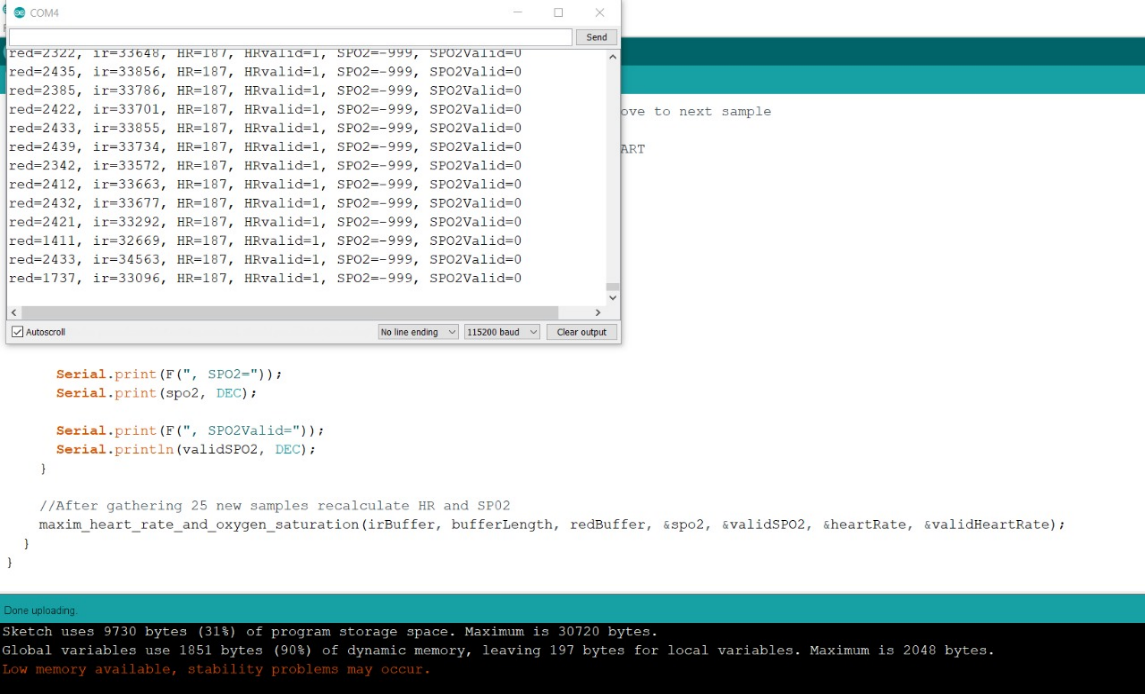
//After gathering 25 new samples recalculate HR and SP02

maxim\_heart\_rate\_and\_oxygen\_saturation(irBuffer, bufferLength, redBuffer, &spo2, &validSPO2, &heartRate, &validHeartRate);

}

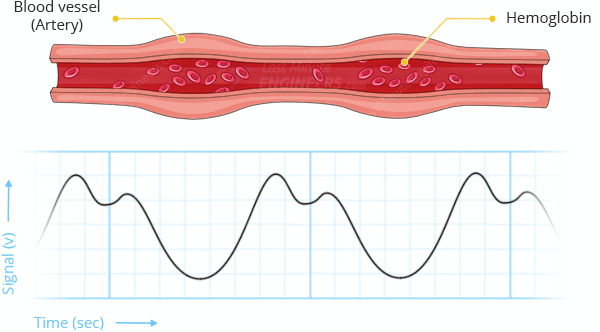
}

**1.3.2 Real Life Reading of SpO2:**



**1.4 Heart Rate Measurement**

The oxygenated hemoglobin (HbO2) in the arterial blood has the characteristic of absorbing IR light. The redder the blood (the higher the hemoglobin), the more IR light is absorbed. As the blood is pumped through the finger with each heartbeat, the amount of reflected light changes, creating a changing waveform at the output of the photodetector. As you continue to shine light and take photodetector readings, you quickly start to get a heart-beat (HR) pulse reading.



**1.4.1 Heart Rate Code in Arduino:**

#include <Wire.h>

#include "MAX30105.h"

#include "heartRate.h"

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;

void setup()

{

Serial.begin(115200);

Serial.println("Initializing...");

// Initialize sensor

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

{

Serial.println("MAX30105 was not found. Please check wiring/power. ");

while (1);

}

Serial.println("Place your index finger on the sensor with steady pressure.");

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

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

particleSensor.setPulseAmplitudeGreen(0); //Turn off Green LED

}

void loop()

{

long irValue = particleSensor.getIR();

if (checkForBeat(irValue) == true)

{

//We sensed a beat!

long delta = millis() - lastBeat;

lastBeat = millis();

beatsPerMinute = 60 / (delta / 1000.0);

if (beatsPerMinute < 255 && beatsPerMinute > 20)

{

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;

}

}

Serial.print("IR=");

Serial.print(irValue);

Serial.print(", BPM=");

Serial.print(beatsPerMinute);

Serial.print(", Avg BPM=");

Serial.print(beatAvg);

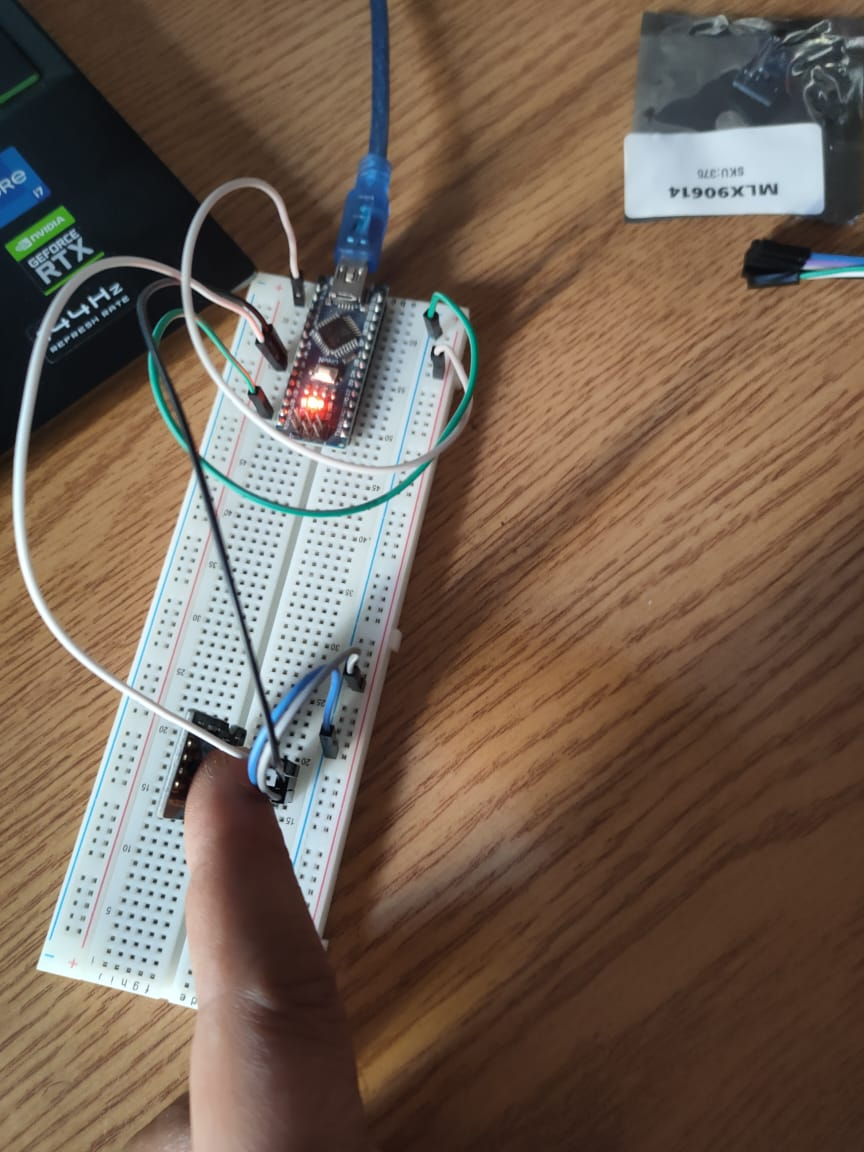
if (irValue < 50000)

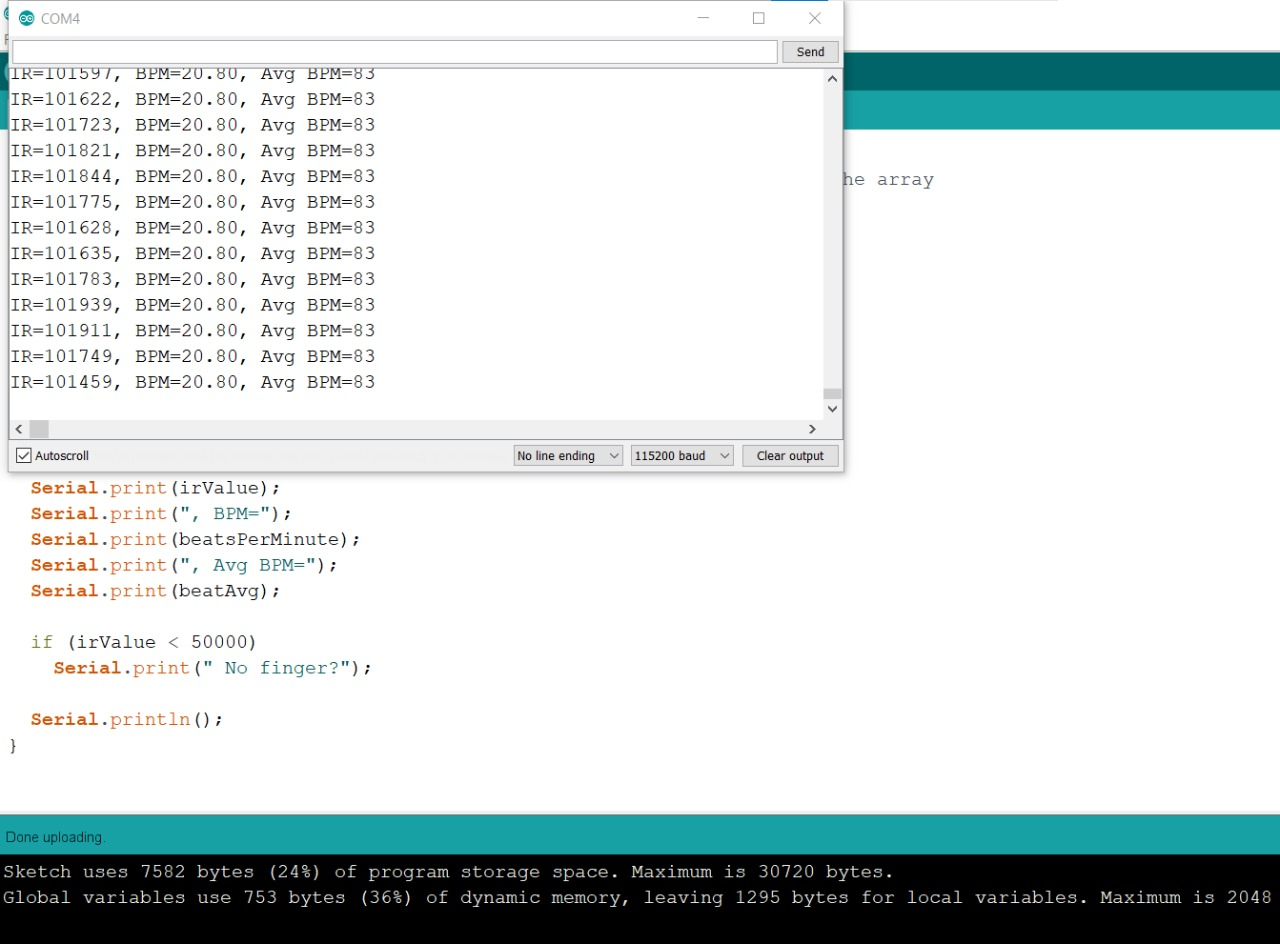
Serial.print(" No finger?");

Serial.println();

}

**1.3.2 Real Life Reading of SpO2:**





**1.5 Reading Temperature (Spare using)**

**1.5.1 Arduino Code of Reading Temperature:**

#include "MAX30105.h"

MAX30105 particleSensor;

void setup() {

Serial.begin(9600);

Serial.println("Initializing...");

// Initialize sensor

if (particleSensor.begin(Wire, I2C\_SPEED\_FAST) == false) { //Use default I2C port, 400kHz speed

Serial.println("MAX30102 was not found. Please check wiring/power. ");

while (1);

}

//The LEDs are very low power and won't affect the temp reading much but

//you may want to turn off the LEDs to avoid any local heating

particleSensor.setup(0); //Configure sensor. Turn off LEDs

particleSensor.enableDIETEMPRDY(); //Enable the temp ready interrupt. This is required.

}

void loop() {

float temperature = particleSensor.readTemperature();

Serial.print("temperatureC=");

Serial.print(temperature, 4);

float temperatureF = particleSensor.readTemperatureF();

Serial.print(" temperatureF=");

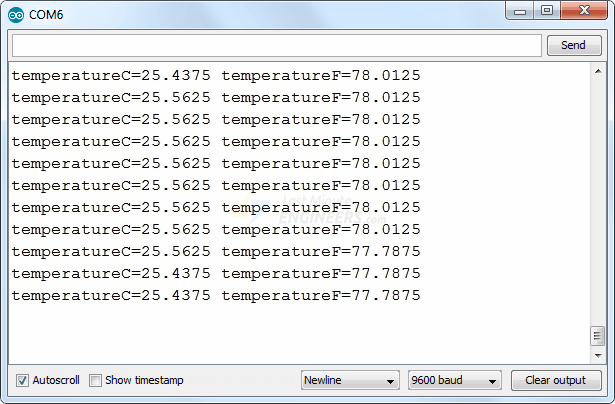
Serial.print(temperatureF, 4);

Serial.println(); }

**1.3.2 Real Life Reading of Reading Temperature:**

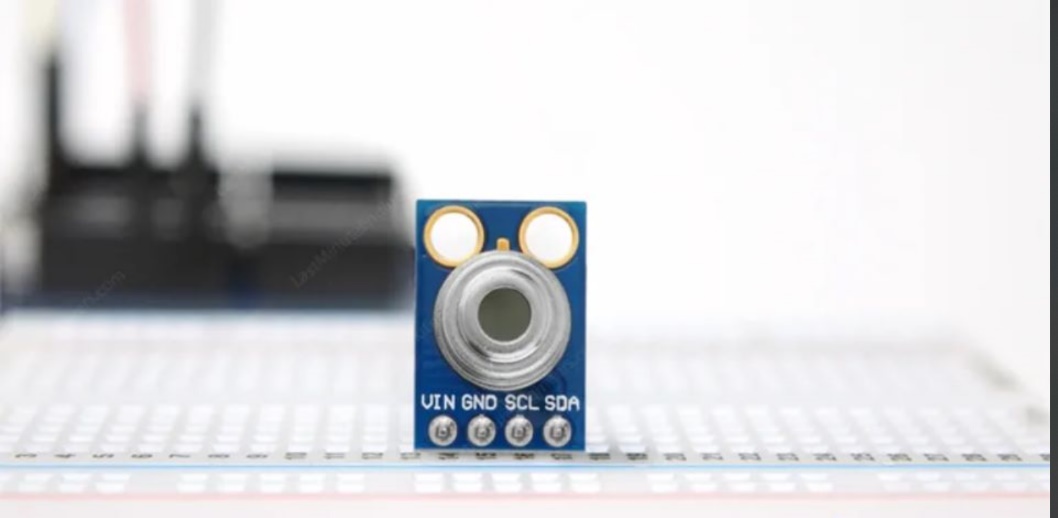
A picture containing text

Description automatically generated



**2 mlx90614 sensor:**

**2.1 Introduction:**

At the heart of the module is a high precision non-contact infrared temperature sensor from melexis – MLX90614. Unlike most temperature sensors, this sensor measures temperature without being physically touched. This can be very useful for monitoring the temperature of something moving like a spinning motor shaft or objects on a conveyor belt for example. Simply point the sensor at what you want to measure and it will detect the temperature by absorbing the emitted IR waves.

A screenshot of a computer

Description automatically generated with medium confidence

**2.2 Connections:**

A picture containing text, electronics, screenshot

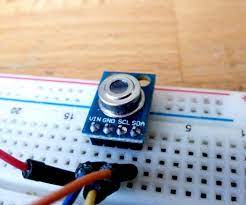
Description automatically generated

VCC is the power pin. You can connect it to 3.3V or 5V output from your Arduino.

GND is the ground.

SCL is the I2C clock pin, connect to your Arduino’s I2C clock line.

SDA is the I2C data pin, connect to your Arduino’s I2C data line.

****

**1.3 Circuit Diagram:**

Diagram, schematic

Description automatically generated

A close-up of a computer

Description automatically generated with low confidence

**2.4 mlx90614 sensor Temperature code in Arduino:**

#include <Adafruit\_MLX90614.h>

Adafruit\_MLX90614 mlx = Adafruit\_MLX90614();

void setup() {

Serial.begin(9600);

while (!Serial);

if (!mlx.begin()) {

Serial.println("Error connecting to MLX sensor. Check wiring.");

while (1);

};

}

void loop() {

Serial.print("Ambient = "); Serial.print(mlx.readAmbientTempC());

Serial.print("\*C\tObject = "); Serial.print(mlx.readObjectTempC()); Serial.println("\*C");

Serial.print("Ambient = "); Serial.print(mlx.readAmbientTempF());

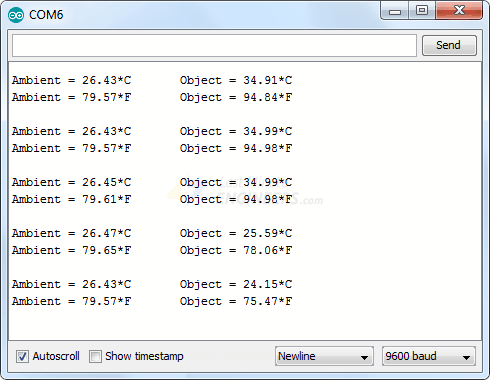
Serial.print("\*F\tObject = "); Serial.print(mlx.readObjectTempF()); Serial.println("\*F");

Serial.println();

delay(500);

}

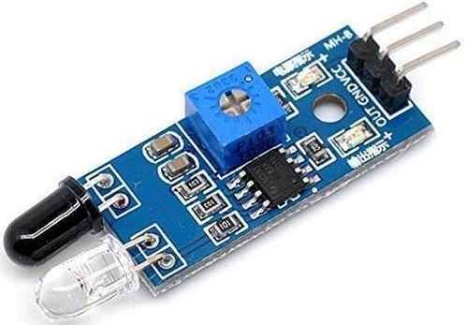
**2.5 Real Life Reading of SpO2:**



**3 IR sensor**

**3.1 Introduction:**

An infrared (IR) sensor is an electronic device that measures and detects infrared radiation in its surrounding environment. Infrared radiation was accidentally discovered by an astronomer named William Herchel in 1800. While measuring the temperature of each color of light (separated by a prism), he noticed that the temperature just beyond the red light was highest. IR is invisible to the human eye, as its wavelength is longer than that of visible light (though it is still on the same electromagnetic spectrum). Anything that emits heat (everything that has a temperature above around five degrees Kelvin) gives off infrared radiation.

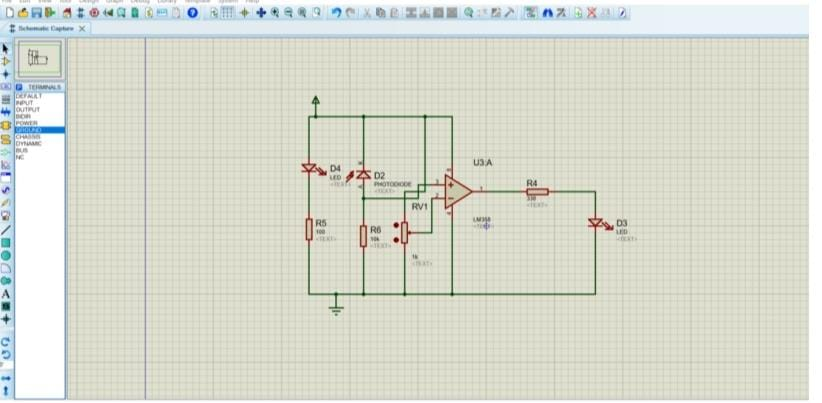


**3.2 Connections:**

Diagram, schematic

Description automatically generated

**3.3 The circuit**



**3.4 IR Sensor Code in Arduino:**

int IRSensor = 9; // connect IR sensor module to Arduino pin D9

int LED = 13; // connect LED to Arduino pin 13

void setup(){

Serial.begin(115200); // Init Serial at 115200 Baud Rate.

Serial.println("Serial Working"); // Test to check if serial is working or not

pinMode(IRSensor, INPUT); // IR Sensor pin INPUT

pinMode(LED, OUTPUT); // LED Pin Output

}

void loop(){

int sensorStatus = digitalRead(IRSensor); // Set the GPIO as Input

if (sensorStatus == 1) // Check if the pin high or not

{

// if the pin is high turn off the onboard Led

digitalWrite(LED, LOW); // LED LOW

Serial.println("Motion Detected!"); // print Motion Detected! on the serial monitor window

}

else {

//else turn on the onboard LED

digitalWrite(LED, HIGH); // LED High

Serial.println("Motion Ended!"); // print Motion Ended! on the serial monitor window

}

}

**3.5 IR Real Life**

A picture containing electronics

Description automatically generated