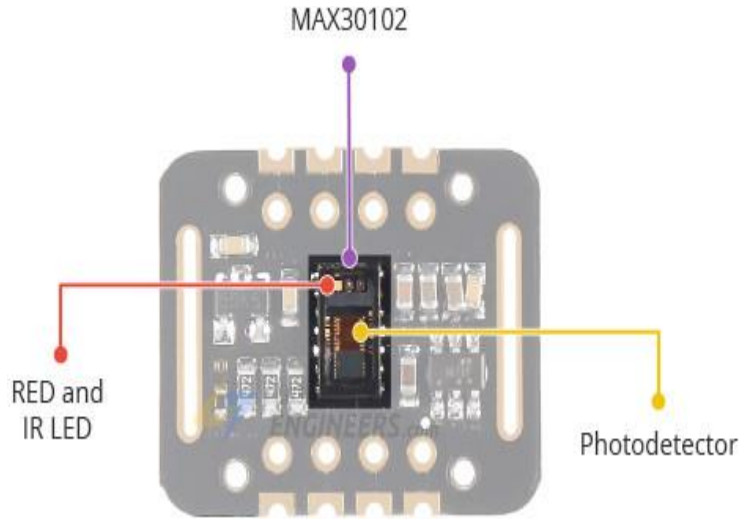
Interfacing MAX30102 Pulse Oximeter and Heart Rate Sensor with Arduino

The MAX30102 pulse oximeter and heart rate sensor is an I2C-based low-power plug-and-play biometric sensor. It can be used by students, hobbyists, engineers, manufacturers, and game & mobile developers who want to incorporate live heart-rate data into their projects.

MAX30102 Module Hardware Overview

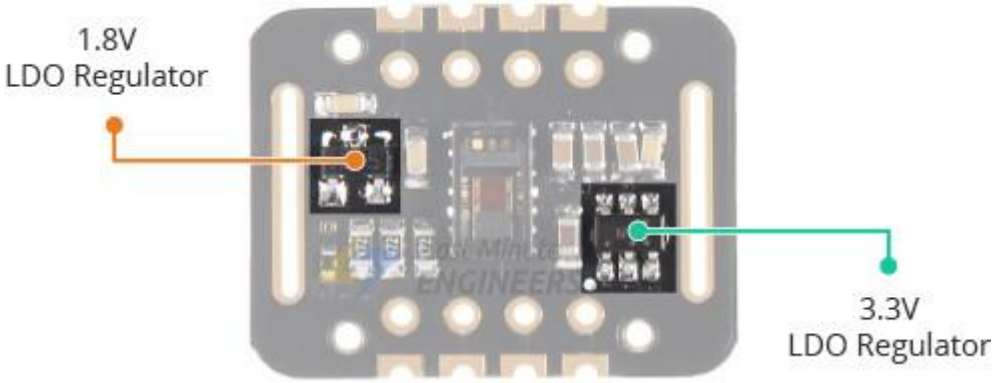
The module features the MAX30102 – a modern (the successor to the MAX30100), integrated pulse oximeter and heart rate sensor IC, from Analog Devices. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse oximetry (SpO2) and heart rate (HR) signals.



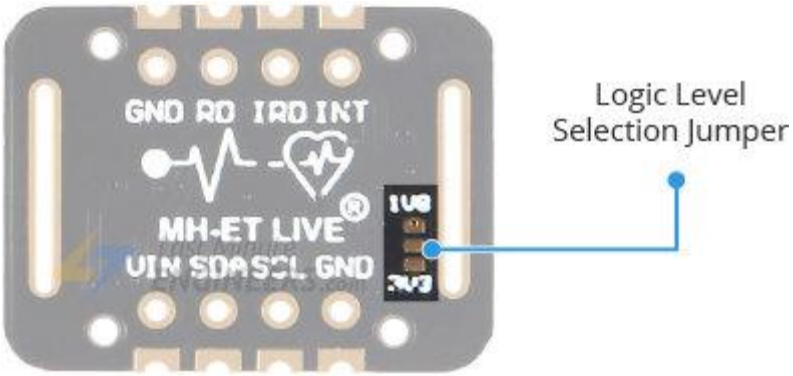
Behind the window on one side, the MAX30102 has two LEDs – a RED and an IR LED. On the other side is a very sensitive photodetector. The idea is that you shine a single LED at a time, detecting the amount of light shining back at the detector, and, based on the signature, you can measure blood oxygen level and heart rate.

Power Requirement

The MAX30102 chip requires two different supply voltages: 1.8V for the IC and 3.3V for the RED and IR LEDs. So the module comes with 3.3V and 1.8V regulators.



On the back of the PCB you'll find a solder jumper that can be used to select between 3.3V and 1.8V logic level. By default 3.3V logic level is selected which is compatible with logic levels for Arduino. But you can also select 1.8V logic level as per your requirement. This allows you to connect the module to any microcontroller with 5V, 3.3V, even 1.8V level I/O.



One of the most important features of the MAX30102 is its low power consumption: the MAX30102 consumes less than 600µA during measurement. Also it is possible to put the MAX30102 in standby mode, where it consumes only 0.7µA. This low power consumption allows implementation in battery powered devices such as handsets, wearables or smart watches.

On-Chip Temperature Sensor

The MAX30102 has an on-chip temperature sensor that can be used to compensate for the changes in the environment and to calibrate the measurements.

This is a reasonably precise temperature sensor that measures the 'die temperature' in the range of -40˚C to +85˚C with an accuracy of ±1˚C.

I2C Interface

The module uses a simple two-wire I2C interface for communication with the microcontroller. It has a fixed I2C address: $0xAE_{HEX}$ (for write operation) and $0xAF_{HEX}$ (for read operation).

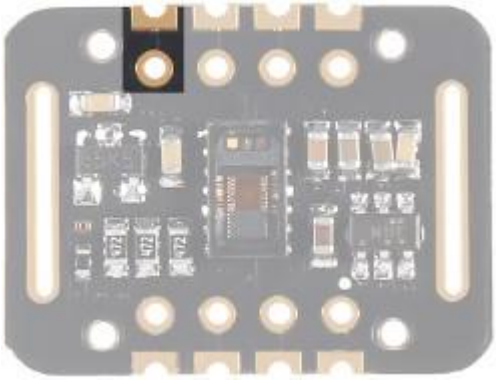
FIFO Buffer

The MAX30102 embeds a FIFO buffer for storing data samples. The FIFO has a 32-sample memory bank, which means it can hold up to 32 SpO2 and heart rate samples. The FIFO buffer can offload the microcontroller from reading each new data sample from the sensor, thereby saving system power.

Interrupts

The MAX30102 can be programmed to generate an interrupt, allowing the host microcontroller to perform other tasks while the data is collected by the sensor. The interrupt can be enabled for 5 different sources:

- **Power Ready**: triggers on power-up or after a brownout condition.
- **New Data Ready**: triggers after every SpO2 and HR data sample is collected.
- **Ambient Light Cancellation**: triggers when the ambient light cancellation function of the SpO2/HR photodiode reaches its maximum limit, affecting the output of the ADC.
- **Temperature Ready**: triggers when an internal die temperature conversion is finished.
- **FIFO Almost Full**: triggers when the FIFO becomes full and future data is about to be lost.



The INT line is an open-drain, so it is pulled HIGH by the onboard resistor. When an interrupt occurs the INT pin goes LOW and stays LOW until the interrupt is cleared.

Technical Specifications

Here are the technical specifications:

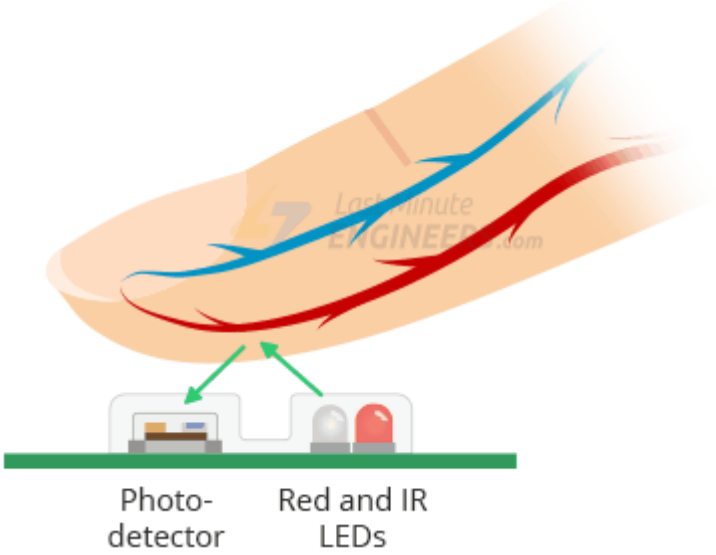
Power supply	3.3V to 5.5V
Current draw	~600µA (during measurements) ~0.7µA (during standby mode)
Red LED Wavelength	660nm
IR LED Wavelength	880nm
Temperature Range	-40˚C to +85˚C
Temperature Accuracy	±1˚C

You can find extensive information for the MAX30102 sensor from the datasheet.

MAX30102 Datasheet

How MAX30102 Pulse Oximeter and Heart Rate Sensor Works?

The MAX30102, or any optical pulse oximeter and heart-rate sensor for that matter, consists of a pair of high-intensity LEDs (RED and IR, both of different wavelengths) and a photodetector. The wavelengths of these LEDs are 660nm and 880nm, respectively.



The MAX30102 works by shining both lights onto the finger or earlobe (or essentially anywhere where the skin isn't too thick, so both lights can easily penetrate the tissue) and

measuring the amount of reflected light using a photodetector. This method of pulse detection through light is called Photoplethysmogram.

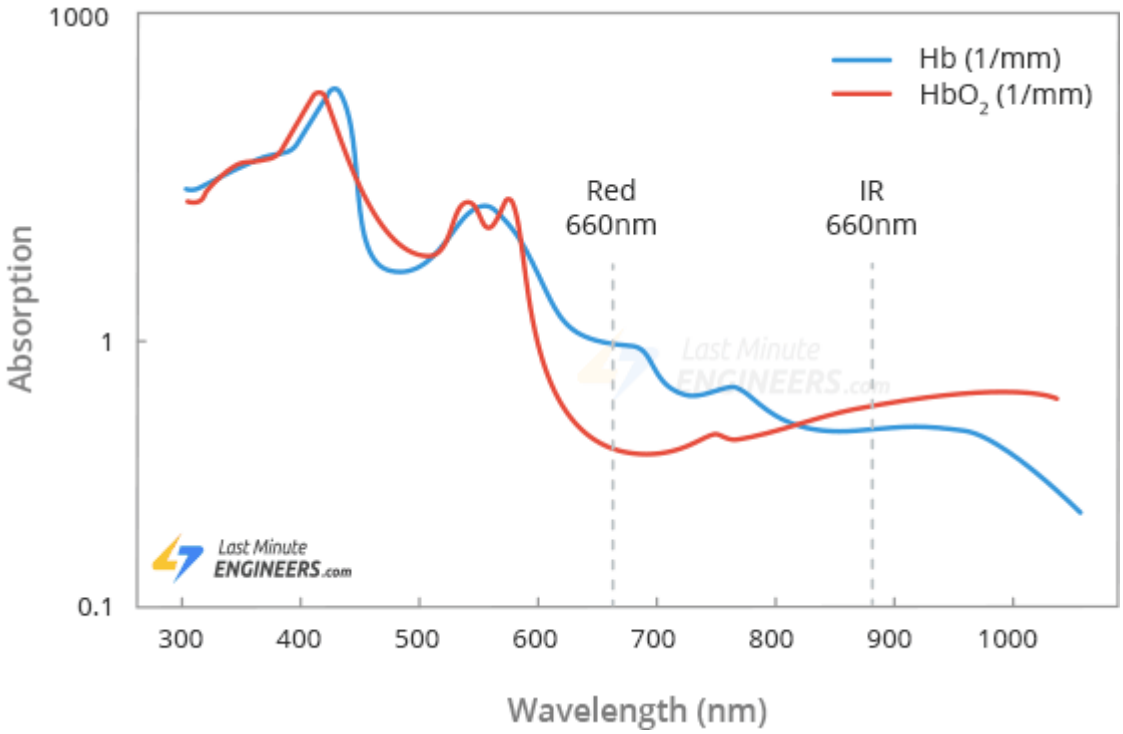
The working of MAX30102 can be divided into two parts: Heart Rate Measurement and Pulse Oximetry (measuring the oxygen level of the blood).

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.

Pulse Oximetry

Pulse oximetry is based on the principle that the amount of RED and IR light absorbed varies depending on the amount of oxygen in your blood. The following graph is the absorption-spectrum of oxygenated hemoglobin (HbO2) and deoxygenated hemoglobin (Hb).



As you can see from the graph, deoxygenated blood absorbs more RED light (660nm), while oxygenated blood absorbs more IR light (880nm). By measuring the ratio of IR and RED light received by the photodetector, the oxygen level (SpO2) in the blood is calculated.

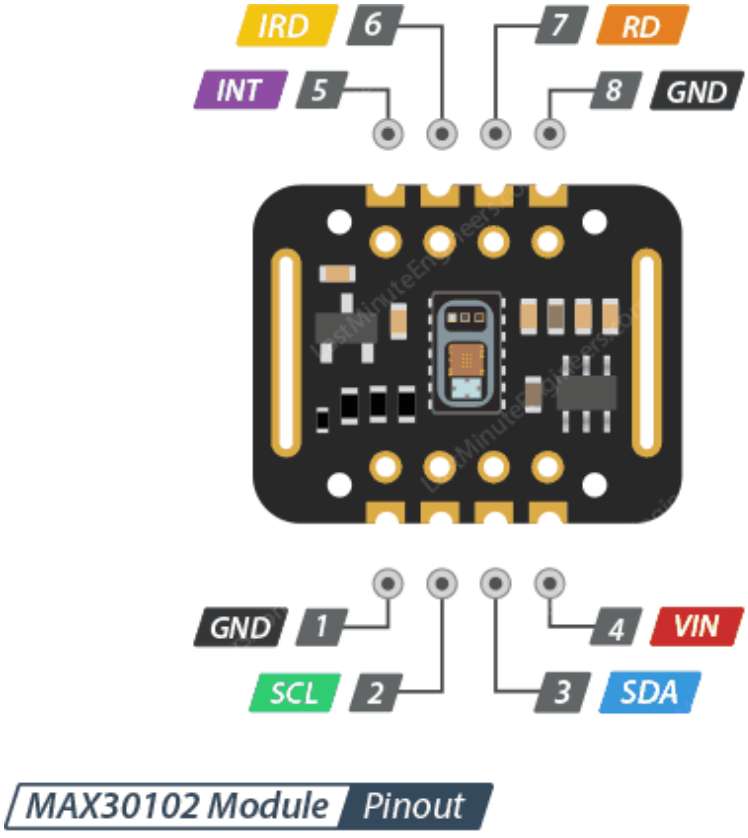
Did you know?

The measurement of hemoglobin oxygen saturation (HbO2) by measuring the absorbance of red and IR light was introduced in 1935 by a German physician, Karl Matthes.

At first, there were no good photodetectors so instead of the IR band, the green band of the light spectrum was used. As technology advanced, more reliable methods were developed, and green light was replaced by IR light.

MAX30102 Module Pinout

The MAX30102 module brings out the following connections.



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

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.

INT - The MAX30102 can be programmed to generate an interrupt for each pulse. This line is open-drain, so it is pulled HIGH by the on-board resistor. When an interrupt occurs the INT pin goes LOW and stays LOW until the interrupt is cleared.

IRD - The MAX30102 integrates an LED driver to drive LED pulses for SpO2 and HR measurements. Use this if you want to drive the IR LED yourself, otherwise leave it unconnected.

RD - pin is similar to the IRD pin, but is used to drive the Red LED. If you don't want to drive the red LED yourself, leave it unconnected.

GND is the ground.

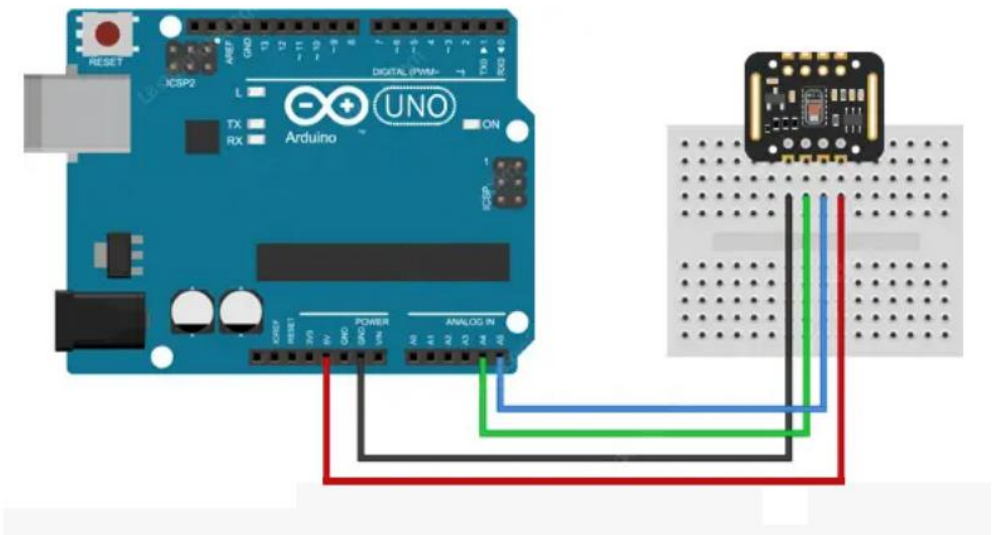
Wiring up a MAX30102 Module to an Arduino

Now that we know everything about the module, we can begin hooking it up to our Arduino!

Start by connecting the VCC pin to the power supply, 3V-5V is fine. Use the same voltage that your microcontroller logic is based off of. For most Arduinos, that is 5V. For 3.3V logic devices, use 3.3V. Now connect GND to common ground.

Connect the SCL pin to the I2C clock pin and the SDA pin to the I2C data pin on your Arduino. Note that each Arduino Board has different I2C pins which should be connected accordingly. On the Arduino boards with the R3 layout, the SDA (data line) and SCL (clock line) are on the pin headers close to the AREF pin. They are also known as A5 (SCL) and A4 (SDA).

The following illustration shows the wiring.

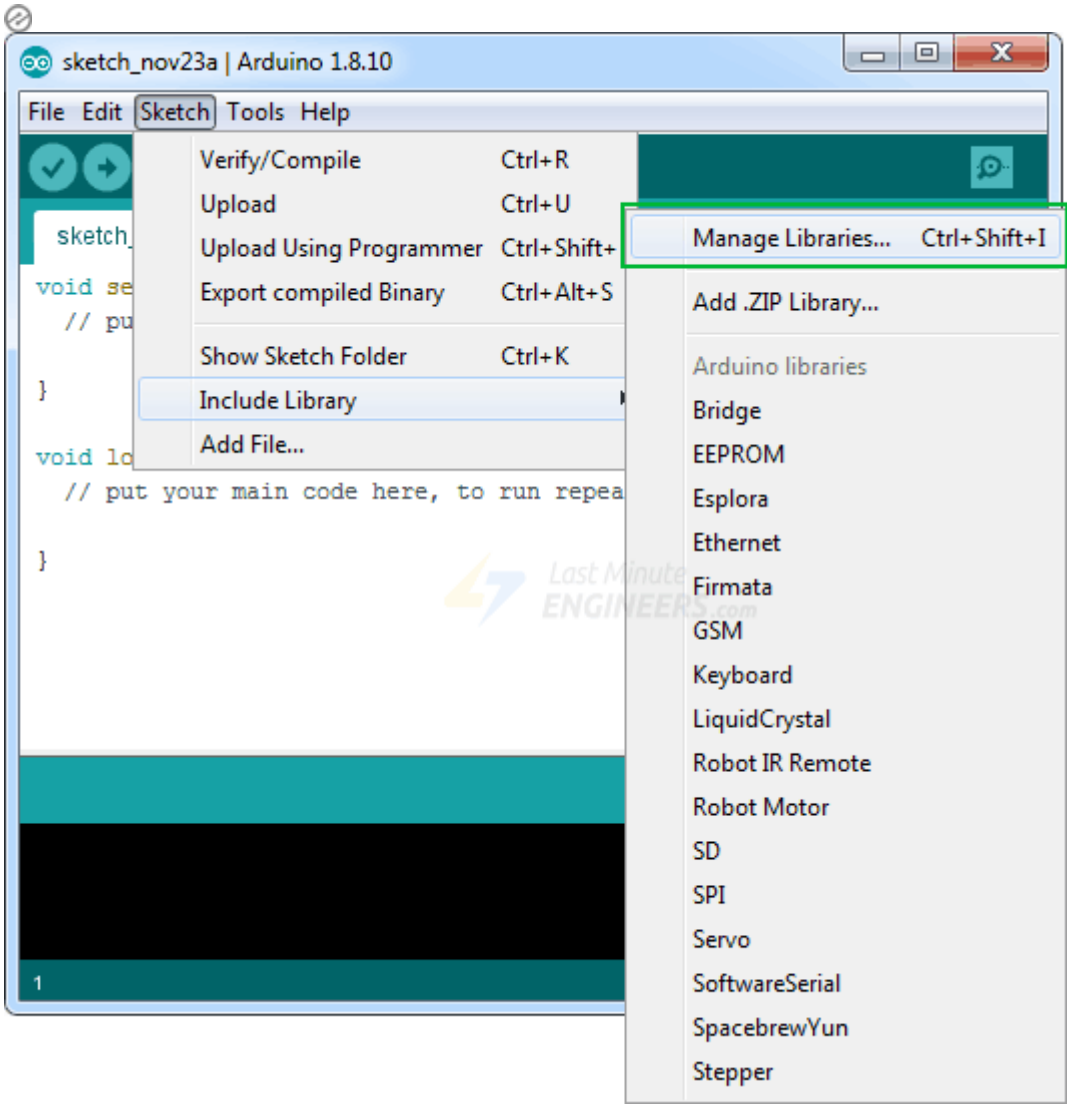


Library Installation

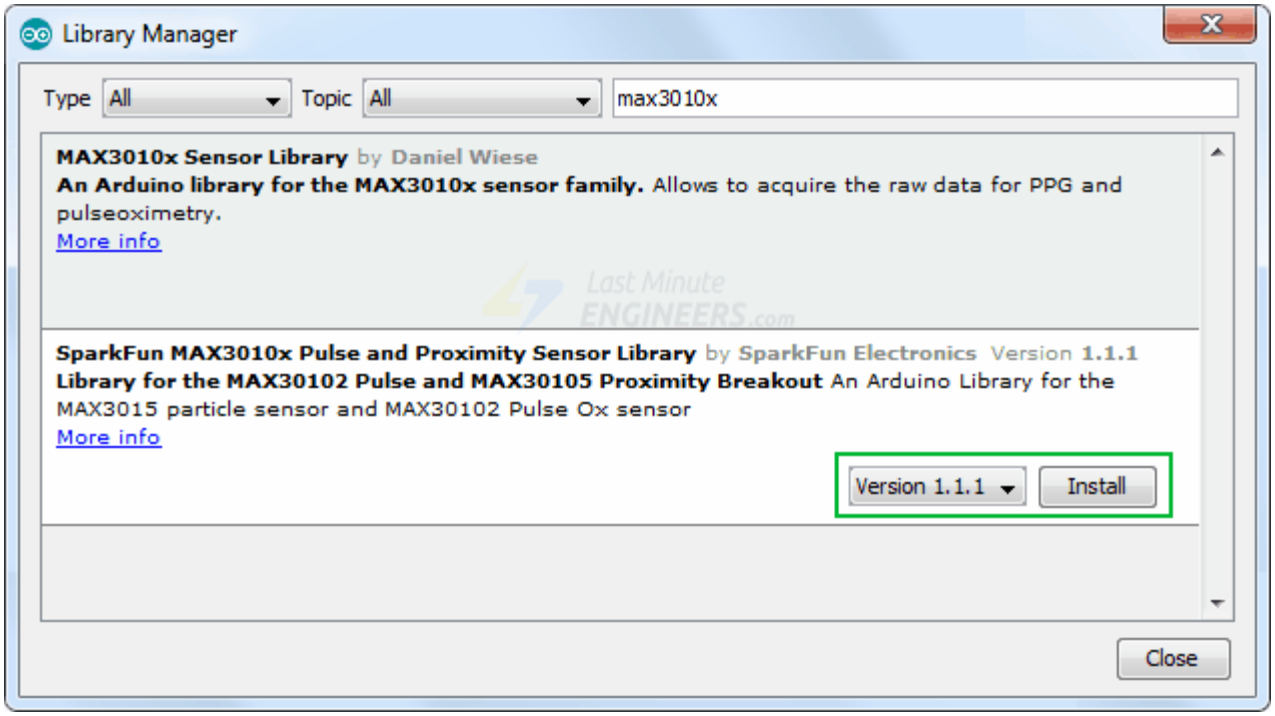
There are several libraries available for the MAX30102 sensor. However in our example, we are using the one by SparkFun Electronics. This library exposes most of the features of the

MAX30102 and offers simple and easy to use functions to calculate pulse rate and SpO2. You can download this library from within the Arduino IDE Library Manager.

To install the library navigate to the Sketch > Include Library > Manage Libraries… Wait for Library Manager to download libraries index and update list of installed libraries.



Filter your search by typing **MAX3010x**. Look for **SparkFun MAX3010x Pulse and Proximity Sensor** Library. Click on that entry, and then select Install.



MAX30102 Example Sketches

The SparkFun_MAX3010x library has a number of example sketches. You can use these example sketches as a basis for developing your own code.

To access the example sketches, navigate to the **File > Examples > SparkFun MAX3010x Pulse and Proximity Sensor Library**. You will see a selection of example sketches.

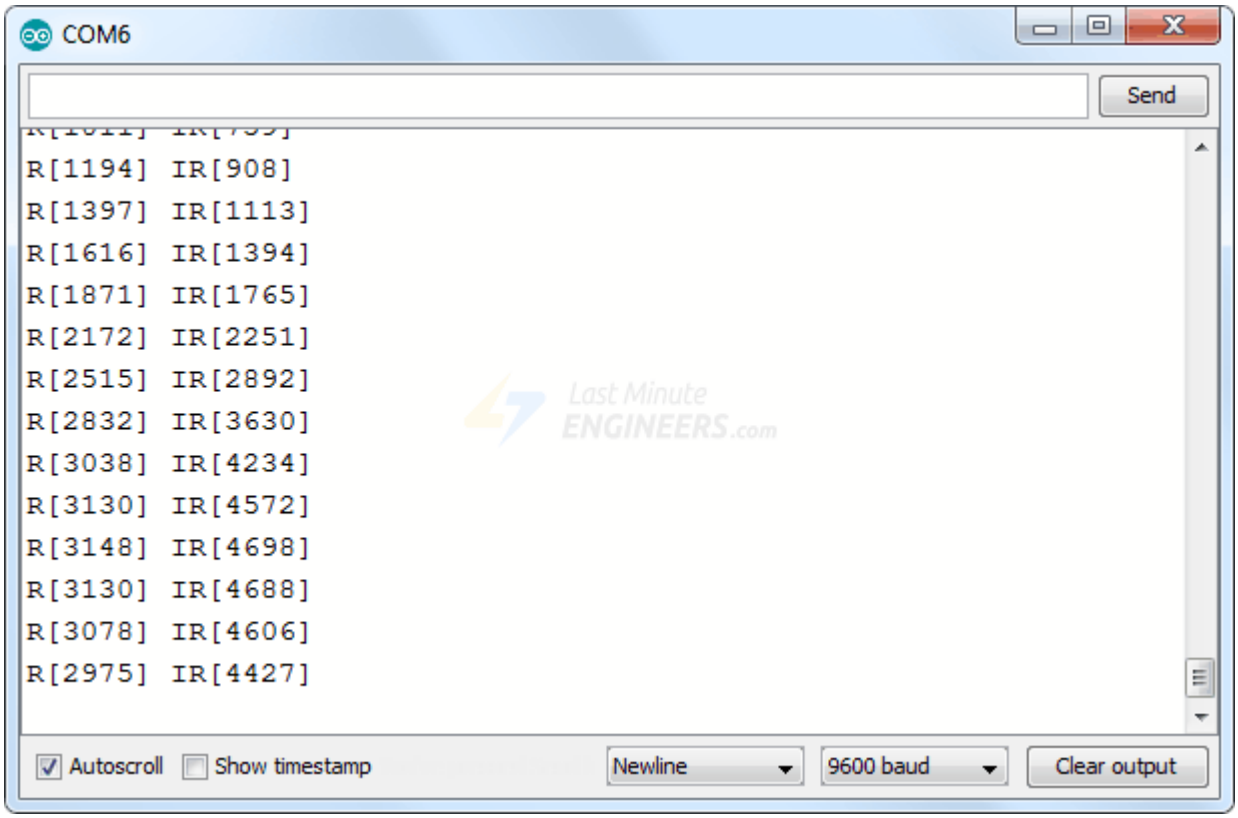
Example 1 – Reading Red & IR

The first example outputs the raw values (IR and Red readings) read by the sensor. Load it onto your Arduino and open the Serial Terminal to see the printed values.

```
#include <Wire.h>
#include "MAX30105.h"

MAX30105 particleSensor;

void setup() {
        Serial.begin(9600);

        // Initialize sensor
        if (particleSensor.begin() == false) {
                Serial.println("MAX30102 was not found. Please check
wiring/power.");
                while (1);
        }

        particleSensor.setup(); //Configure sensor. Use 6.4mA for LED drive
}
```

```
void loop() {
        Serial.print(" R[");
        Serial.print(particleSensor.getRed());
        Serial.print("] IR[");
        Serial.print(particleSensor.getIR());
        Serial.println("]");
}
```

With the sensor pointing up, swipe your hand over the sensor. You should see a change in values as your hand reflects different amounts of light.

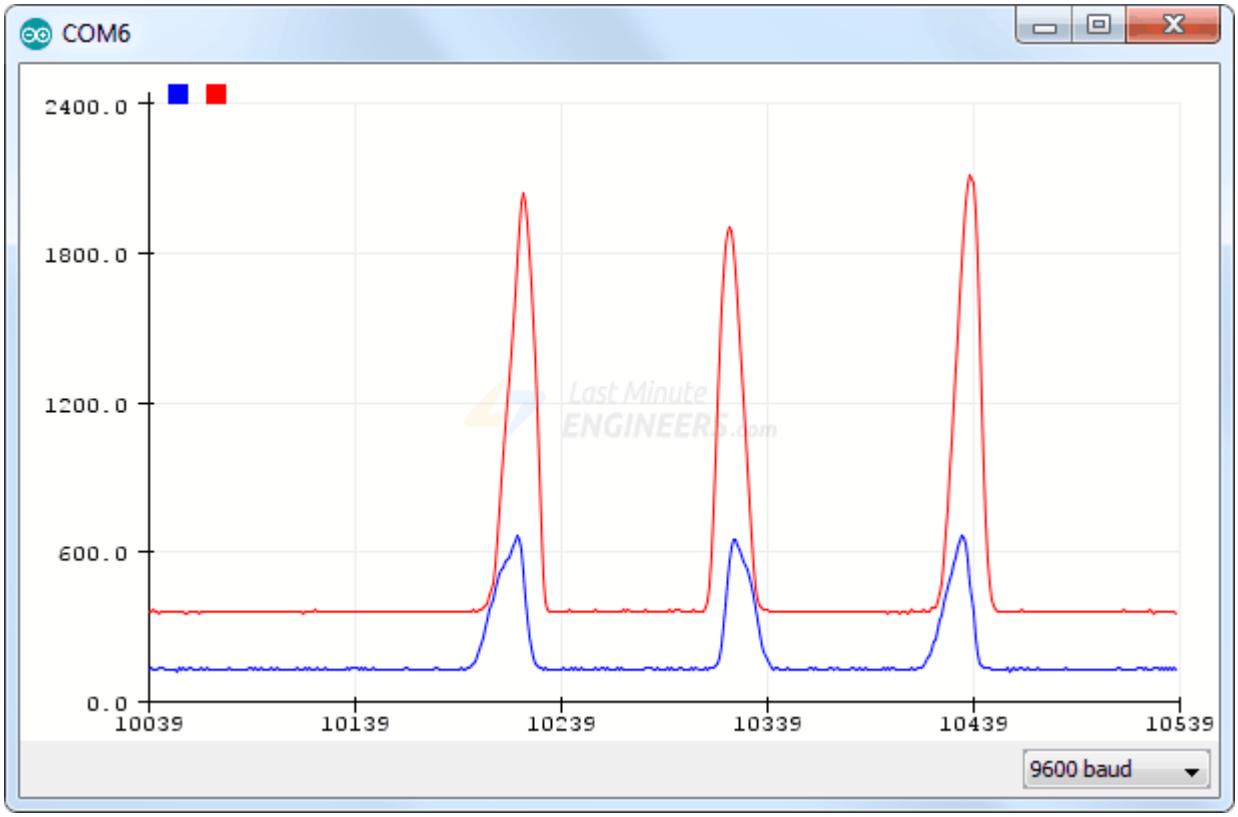


Serial data can be hard to visualize if you're only looking at values. If you are using the Arduino IDE v1.6.6+, there is an option to view the data on a graph using the **Arduino Serial Plotter**.

To start, replace the loop() in the code above with this code snippet:

```
void loop() {
        Serial.print(particleSensor.getRed());
        Serial.print(", ");
        Serial.println(particleSensor.getIR());
}
```

In the Arduino IDE, choose Tools > Serial Plotter. You should see a wave similar to the image below, when you swipe your hand over the sensor.



Example 2 – Presence Sensing

Our next experiment shows how to use the MAX30102 sensor as a general purpose proximity or a reflectance sensor and can serve as the basis for more practical experiments and projects.

This example works by taking a handful of readings during setup and averaging them together. It then uses this average as a baseline. If the sensor detects a significant change from the average, “Something is there!” is printed. Go ahead and try the sketch out.

```
#include <Wire.h>
#include "MAX30105.h"

MAX30105 particleSensor;

long samplesTaken = 0; //Counter for calculating the Hz or read rate
long unblockedValue; //Average IR at power up
long startTime; //Used to calculate measurement rate

void setup() {
  Serial.begin(9600);

  // 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);
  }
```

```
  //Setup to sense up to 18 inches, max LED brightness
  byte ledBrightness = 0xFF; //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
  int sampleRate = 400; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200
  int pulseWidth = 411; //Options: 69, 118, 215, 411
  int adcRange = 2048; //Options: 2048, 4096, 8192, 16384

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

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

  //Take an average of IR readings at power up
  unblockedValue = 0;
  for (byte x = 0 ; x < 32 ; x++) {
    unblockedValue += particleSensor.getIR(); //Read the IR value
  }
  unblockedValue /= 32;

  startTime = millis();
}

void loop() {
  samplesTaken++;

  Serial.print("IR[");
  Serial.print(particleSensor.getIR());
  Serial.print("] Hz[");
  Serial.print((float)samplesTaken / ((millis() - startTime) / 1000.0), 2);
  Serial.print("]");

  long currentDelta = particleSensor.getIR() - unblockedValue;

  Serial.print(" delta[");
  Serial.print(currentDelta);
  Serial.print("]");

  if (currentDelta > (long)100) {
    Serial.print(" Something is there!");
  }

  Serial.println();
}
```

Swipe your hand over the sensor again and look for the message “Something is there!” printed on the serial terminal. Try testing the range at which the sensor can detect something.

Note that the MAX30102 is capable of reading up to 18-bits or values up to 262,144. An extremely small movement can be detected!

Example 3 – Reading Temperature

Our next example outputs readings from the on-board temperature sensor in both Celsius and Fahrenheit. Although the temperature reading should be used to calibrate HR and SpO2 measurements, it can be useful if you need a sensitive and fast-responding temp sensor.

```
#include <Wire.h>

#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();
}
```

Now try heating the sensor with your finger or blowing a light breath on the sensor. You should see something like the output below.

Example 4 – Measuring Heart-Rate (BPM)

This is where the fun begins! In this example, we'll measure heart rate (Beats Per Minute or BPM) of the person we're monitoring.

Warning:

This sketch detects heart-rate optically. This method is tricky and prone to give false readings. So please DO NOT use it for actual medical diagnosis.

```
#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)) {
    Serial.println("MAX30102 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();
}
```

After uploading the sketch, keep your finger on the sensor as steady as possible and wait a few seconds for the readings to make sense. You will see a result like this.

Trouble Seeing a Heartbeat?

If you're having trouble seeing a heartbeat, here's what to do.

- If you hold the sensor too hard, you will squeeze all the blood from your fingers and there will be no sign! If you hold it too lightly, you will invite noise from movement and ambient light. Sweatspot pressure (Not too hard, not too soft) on the pulse sensor will give a good clean signal.
- A varying pressure can cause blood to flow differently in your finger, causing the sensor readings to go wonky. Try to apply constant pressure by attaching the sensor to your finger using a rubber band or other tightening device.
- Try the sensor on different parts of your body that have capillary tissue (such as earlobe or lower lip).

Example 5 – Measuring Oxygen Saturation (SpO2)

In our last example, we'll measure blood oxygen level (SpO2) of the person we're monitoring. Go ahead and try the sketch out.

```
#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);
  }
}
```

After uploading the sketch, keep your finger on the sensor as steady as possible and wait a few seconds for the readings to make sense. You will see a result like this.