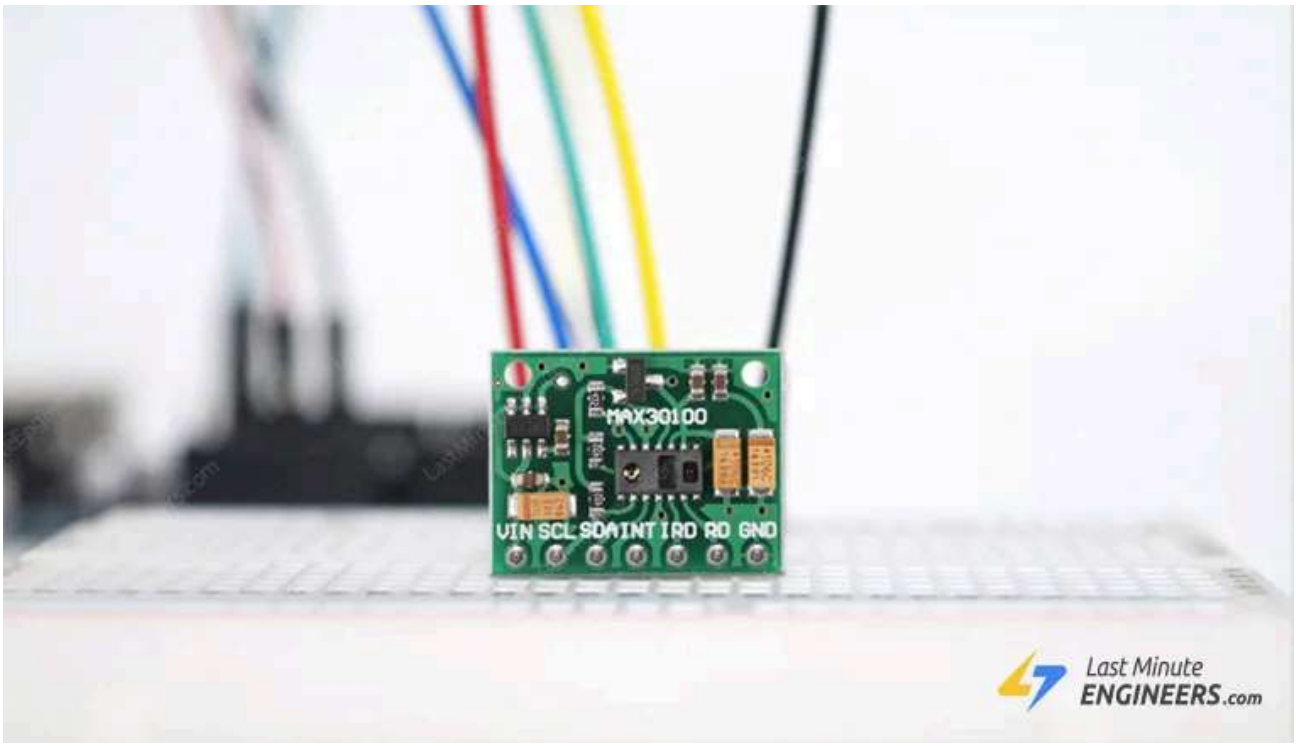
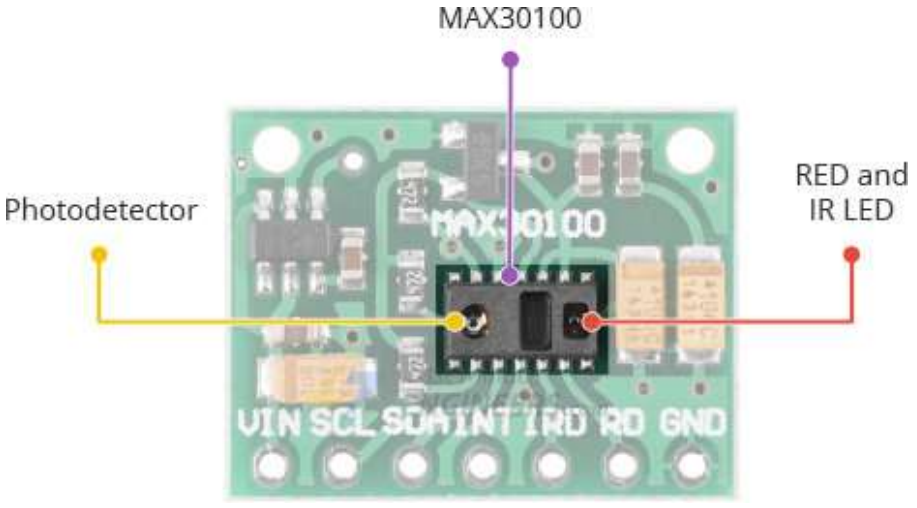
Interfacing MAX30100 Pulse Oximeter and Heart Rate Sensor with Arduino



The MAX30100 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.

MAX30100 Module Hardware Overview

The module features the MAX30100 – a modern, 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.

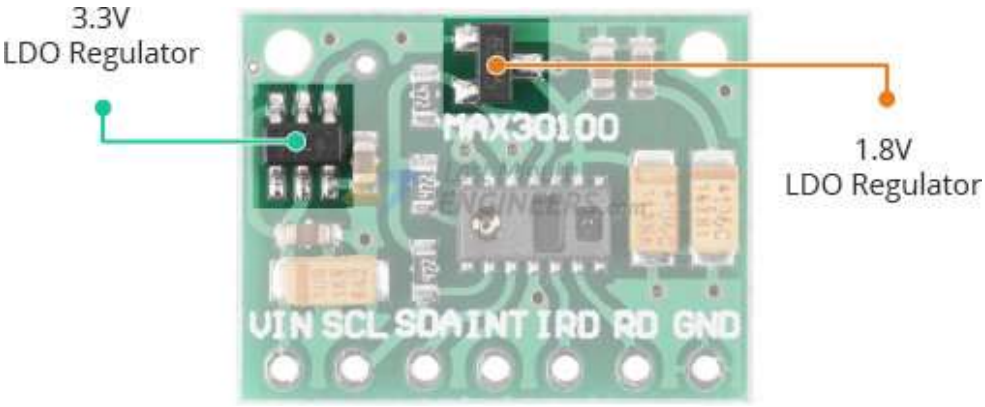


On the right, the MAX30100 has two LEDs – a RED and an IR LED. And on the left is a very sensitive photodetector. The idea is that you shine a single LED at a time, detecting the amount of light shining

back at the detector, and, based on the signature, you can measure blood oxygen level and heart rate.

Power Requirement

The MAX30100 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. 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 MAX30100 is its low power consumption: the MAX30100 consumes less than 600µA during measurement. Also it is possible to put the MAX30100 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 MAX30100 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 MAX30100 embeds a FIFO buffer for storing data samples. The FIFO has a 16-sample memory bank, which means it can hold up to 16 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 MAX30100 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.
- **SpO2 Data Ready** : triggers after every SpO2 data sample is collected.
- **Heart Rate Data Ready** : triggers after every heart rate data sample is collected.
- **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 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 MAX30100 sensor from the datasheet.
MAX30100 Datasheet

MAX30100 vs. MAX30102

Analog Devices has stepped up their game with their new MAX30102 sensor. The MAX30102 beats the MAX30100 in these areas:

FIFO – The MAX30102 has a 32-sample memory bank, which means it can hold up to 32 SpO2 and heart rate samples. Whereas only 16 samples can be kept in MAX30100.

ADC Resolution – The MAX30102 is capable of reading up to 18-bits or values up to 262,144. Whereas the MAX30100 only has 16-bit ADC resolution. Because of that the MAX30102 can detect extremely small movements.

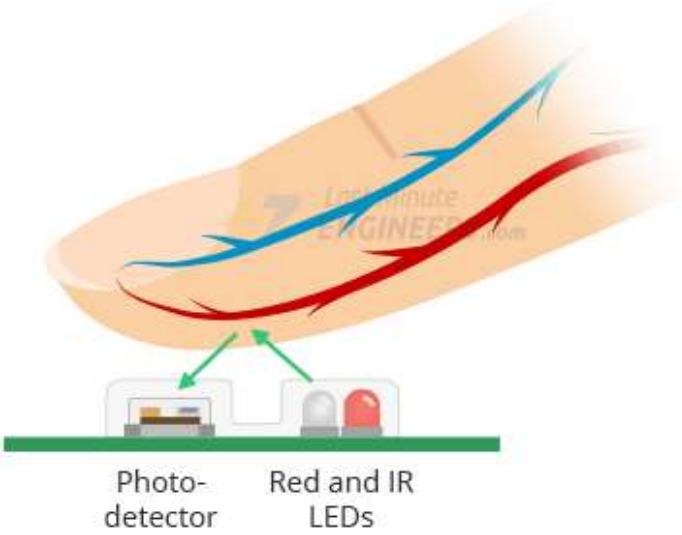
LED Pulse Width – The MAX30102 has a narrower LED pulse width than the MAX30100, resulting in lower power consumption.

To get started with the MAX30102 module, you can refer to the tutorial below

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…

How MAX30100 Pulse Oximeter and Heart Rate Sensor Works?

The MAX30100, 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.

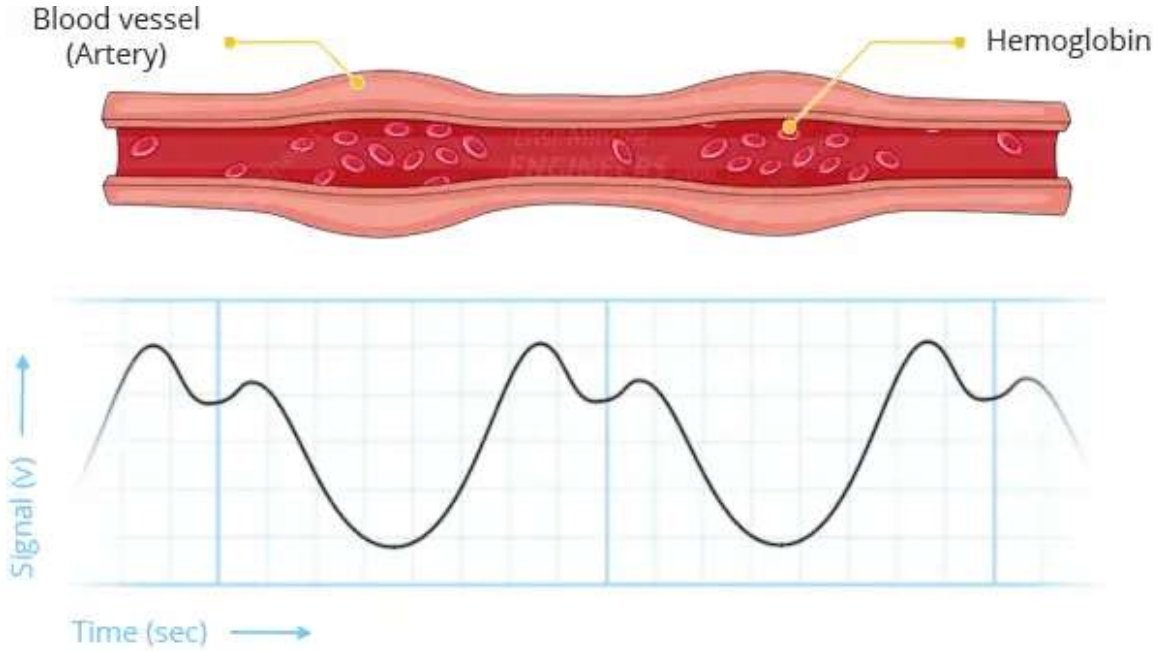


The MAX30100 works by shining both lights onto the finger or earlobe (or essentially anywhere where the skin isn't too thick, so both lights can easily penetrate the tissue) and measuring the amount of reflected light using a photodetector. This method of pulse detection through light is called Photoplethysmogram.

The working of MAX30100 can be divided into two parts: Heart Rate Measurement and Pulse Oximetry (measuring the oxygen level of the blood).

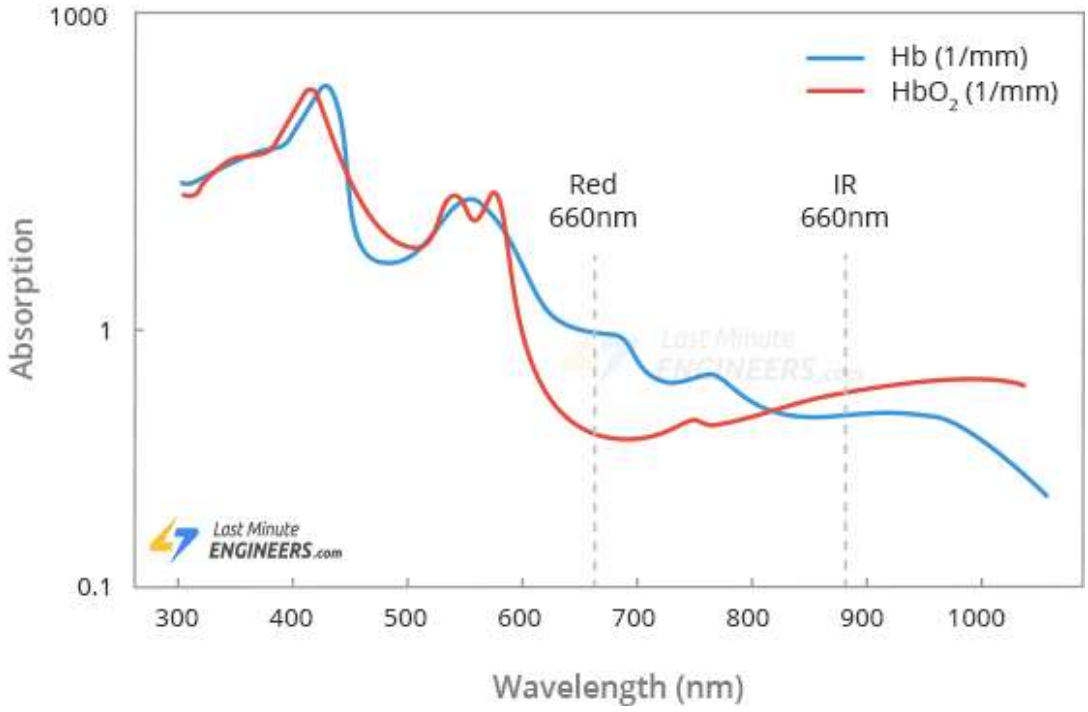
Heart Rate Measurement

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.

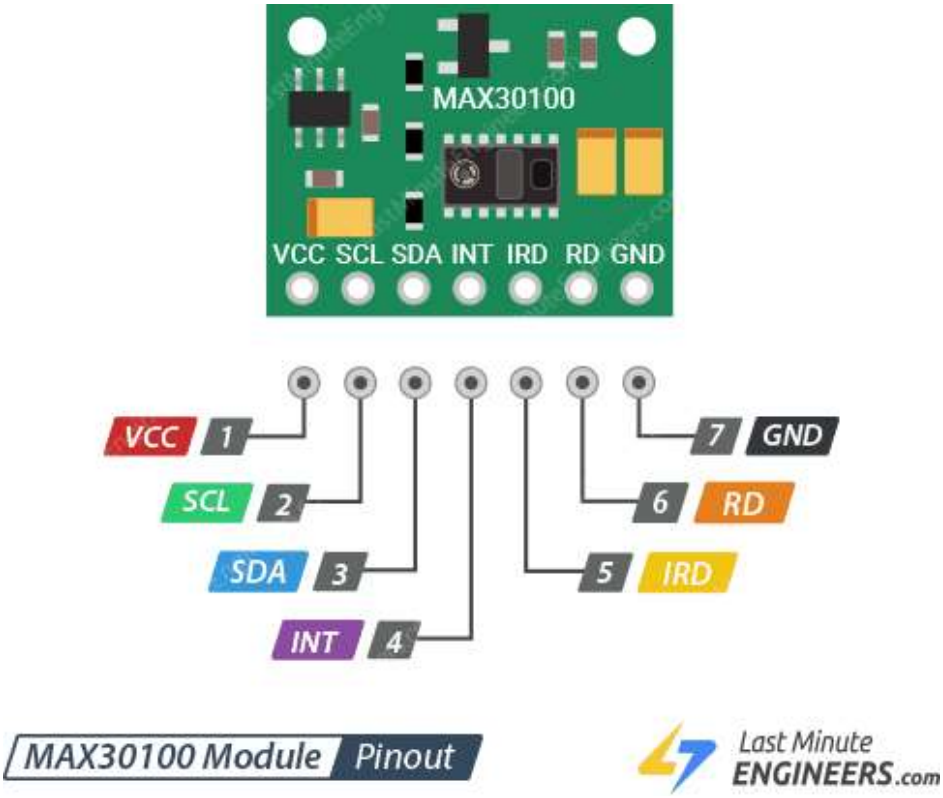
Did you know?

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.

MAX30100 Module Pinout

The MAX30100 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 MAX30100 can be programmed to generate an interrupt for each pulse. This line is 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.

IRD The MAX30100 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.

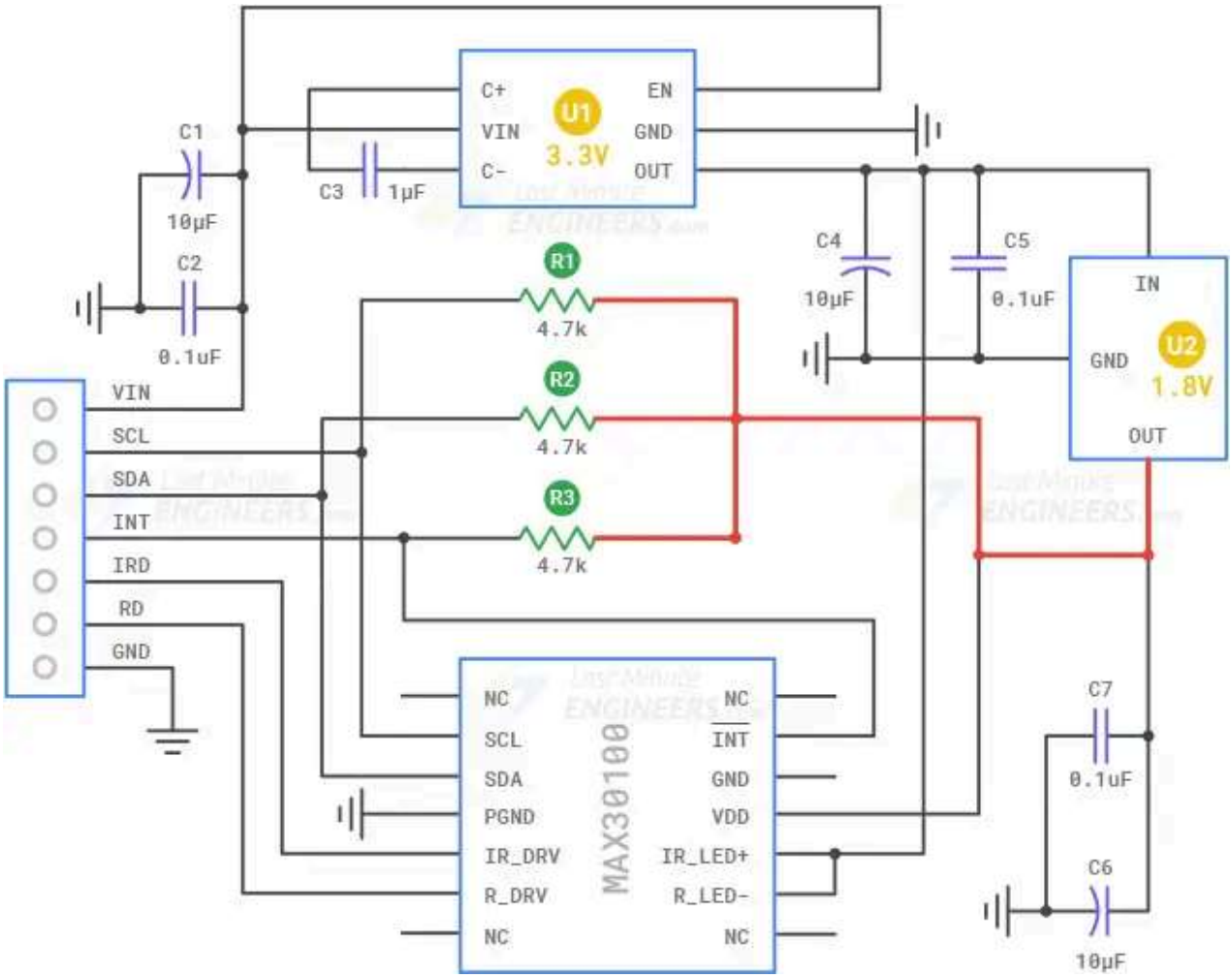
MAX30100 Not Working – Problem and Solutions to Fix

There is no doubt that the MAX30100 module is cheap and very popular among hobbyists, but unfortunately the module (sold in the thousands) has a serious design problem. Well, let's fix it.

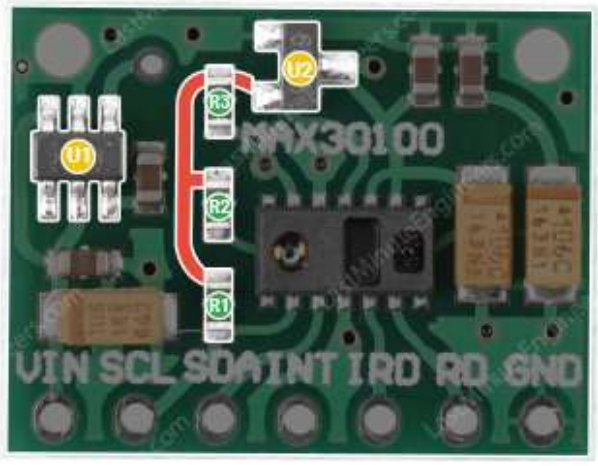
The Problem

The MAX30100 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 two linear voltage regulators – U1 and U2. The first generates 5V to 3.3V. The second regulator is connected to the output of the first and generates 1.8V.

Now take a closer look at the 4.7kΩ pull-up resistors (R1, R2 and R3) for the SCL, SDA and INT signal lines. They are connected to the 1.8V supply (highlighted with a thick red line)!



The two regulators U1, U2 and the trace that connects the 4.7kΩ pull-up resistors to the 1.8V supply are highlighted on the module as well.



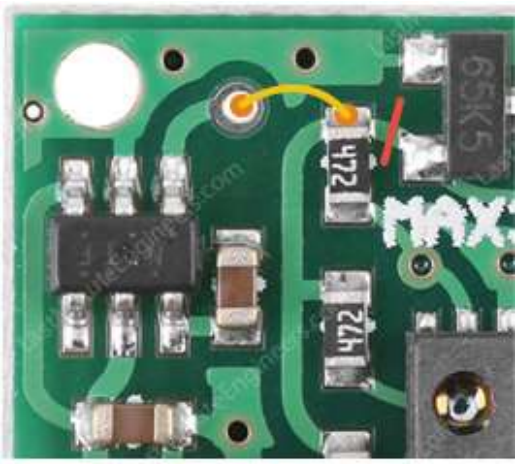
If you connect such module to the Arduino's 5V logic – it will not show up on the I2C bus because the logic level is too low (Arduino reports HIGH if the voltage exceeds 3.0V for 5V-boards and 2.0V for 3.3V-boards). Even with a 3.3V logic board, it will not work.

Don't worry! We have two solutions to fix this problem.

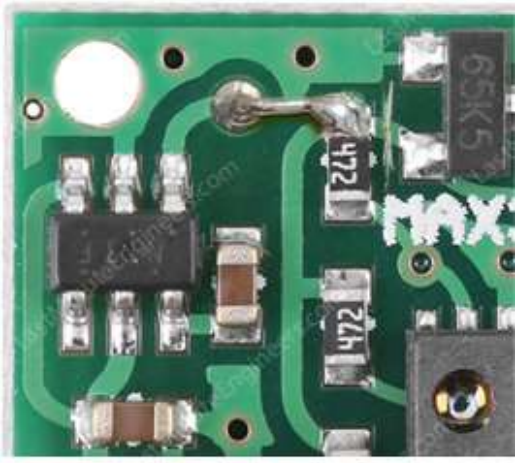
Solution 1 (tested):

1. Cut the trace in the place of the red line. This will disconnect all 4.7kΩ pull-up resistors from the 1.8V supply voltage.
2. Now make a jumper as shown by the yellow line with a piece of wire or a solder blob. This will pull all the 4.7kΩ resistors up to 3.3V.

Here's the board with and without the modifications:

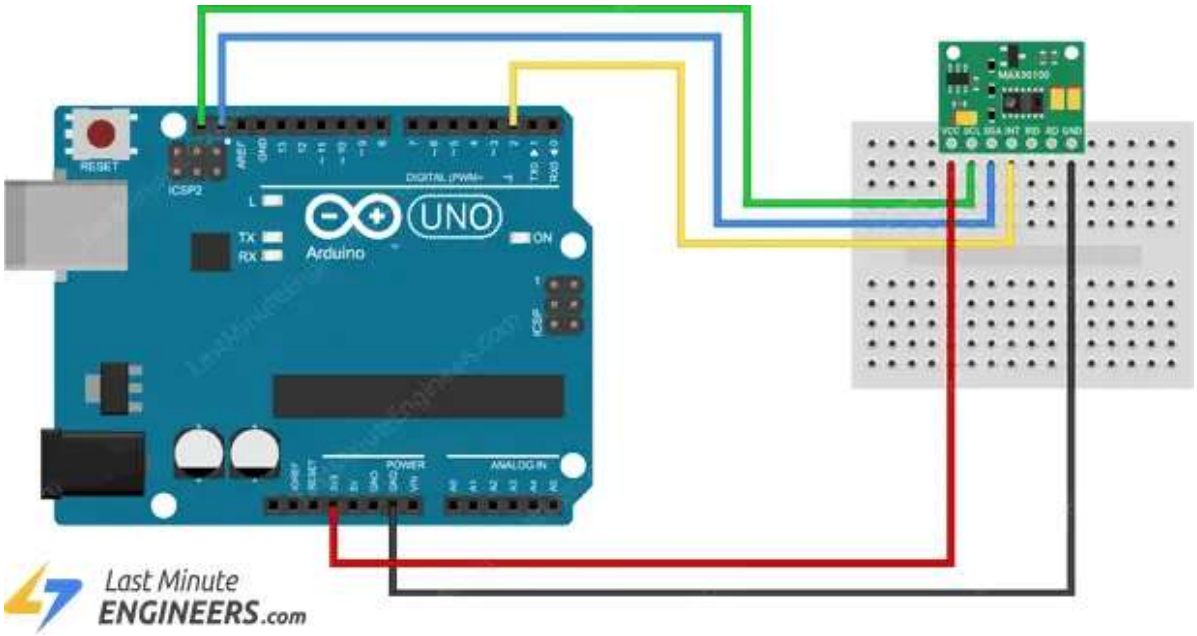


Before



After

After following these steps, connect the module to the Arduino UNO using the wiring diagram below:



The wiring is fairly easy. Simply connect the VIN pin to Arduino's 3.3V and GND to ground. Connect the SCL pin to the I2C clock pin and the SDA pin to the I2C data pin on your Arduino. Finally connect the INT pin to digital pin 2.

Solution 2 (tested):

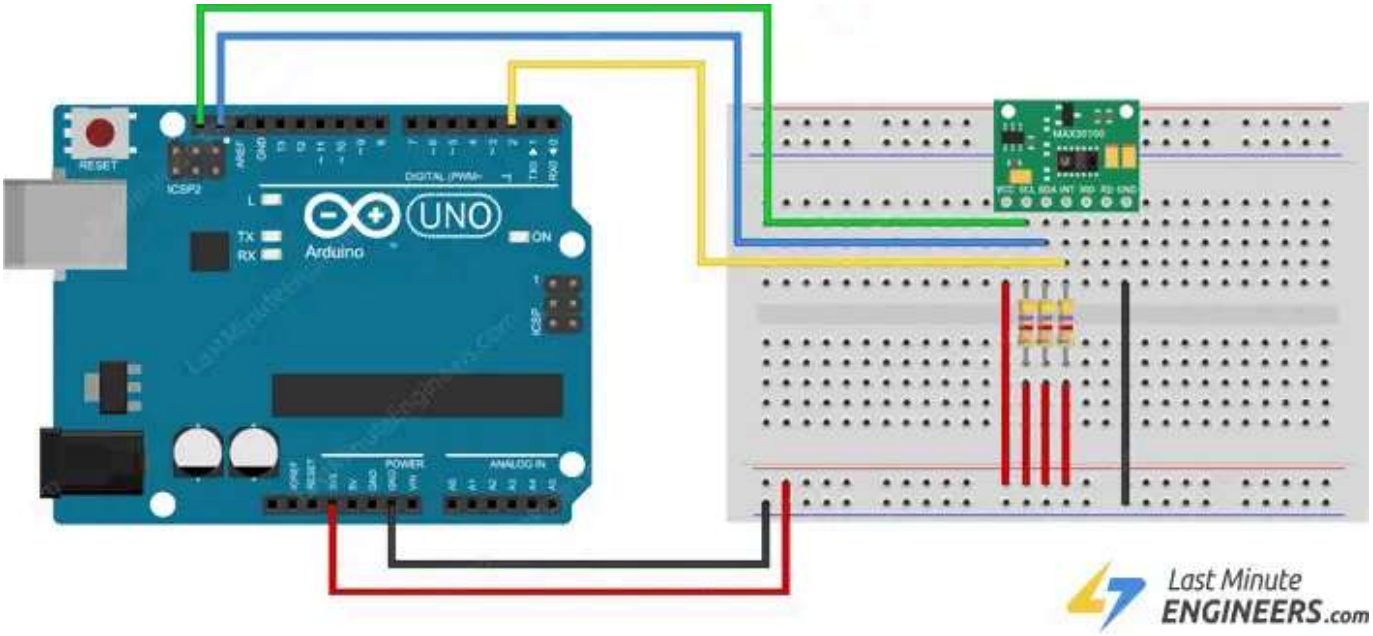
The second solution is to remove all 4.7kΩ pullup resistors from the board. We will use external pullup resistors on the SCL, SDA and INT signal lines later.

Here's the board with and without the resistors removed:

Before

After

After removing the resistors, connect the module to the Arduino UNO using the wiring diagram below:



The wiring is the same as the first one; you just need to add external 4.7kΩ pullup resistors on the SCL, SDA and INT signal lines.

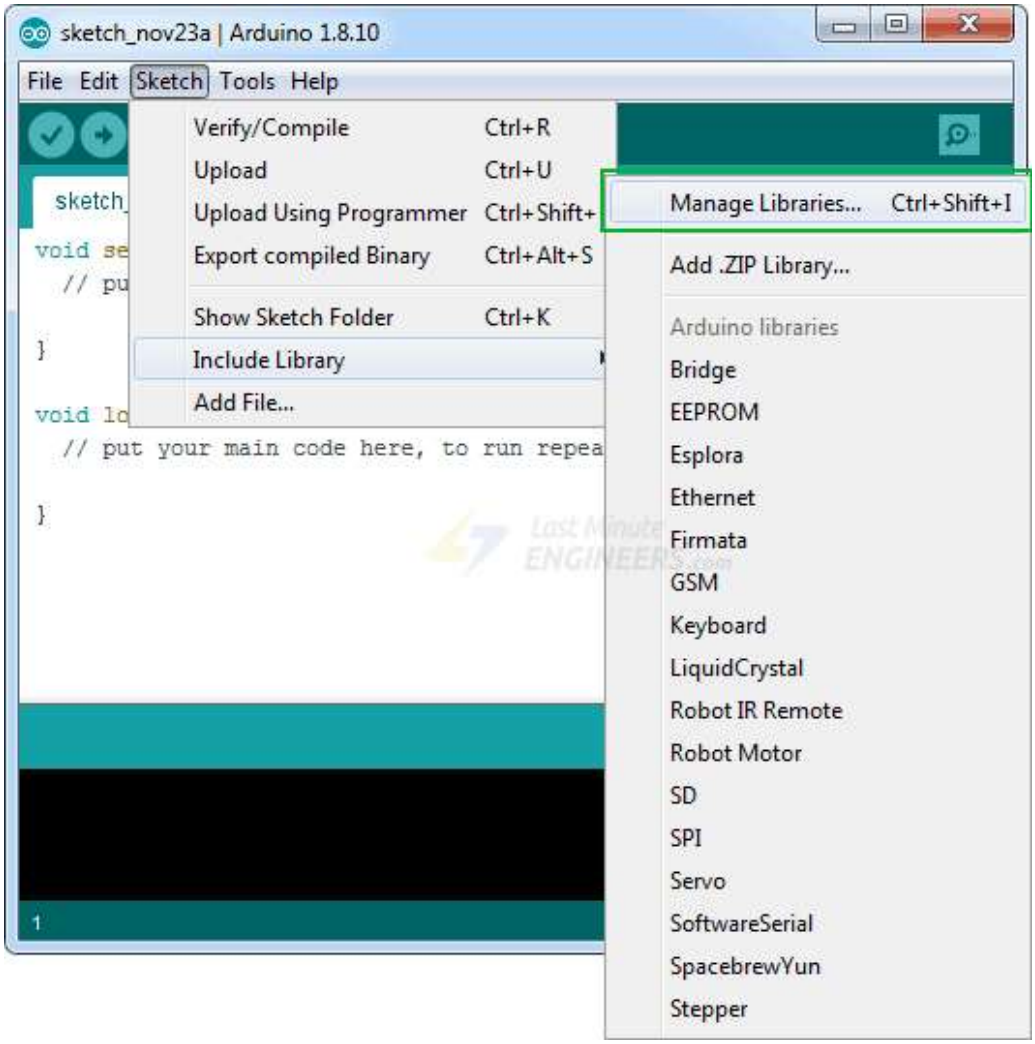
Our Evaluation:

We have tested both solutions and they work great. So try one which is easier for you to implement.

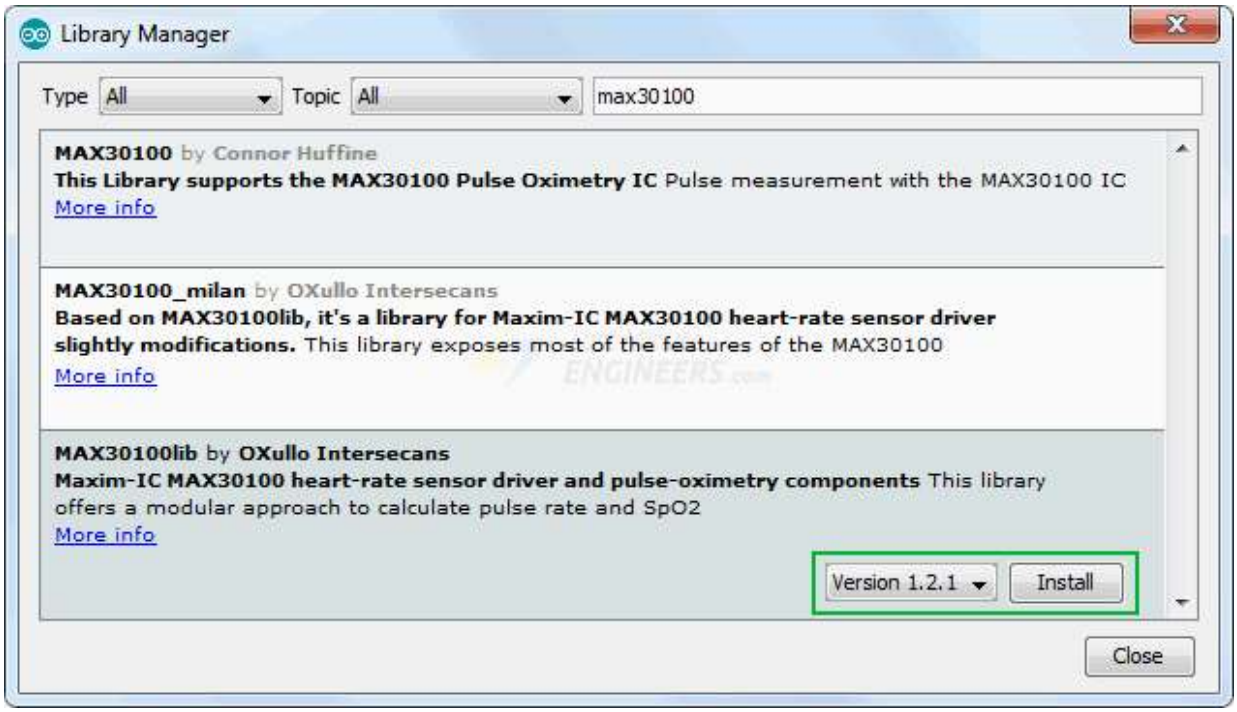
Library Installation

There are several libraries available for the MAX30100 sensor. However in our example, we are using the one by OXullo Intersecans. This library exposes most of the features of the MAX30100 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 **max30100**. There should be a couple entries. Look for **MAX30100lib** Library by **OXullo Intersecans**. Click on the entry, and then select Install.



Example 1 – Measuring Heart-Rate (BPM) and Oxygen Saturation (SpO2)

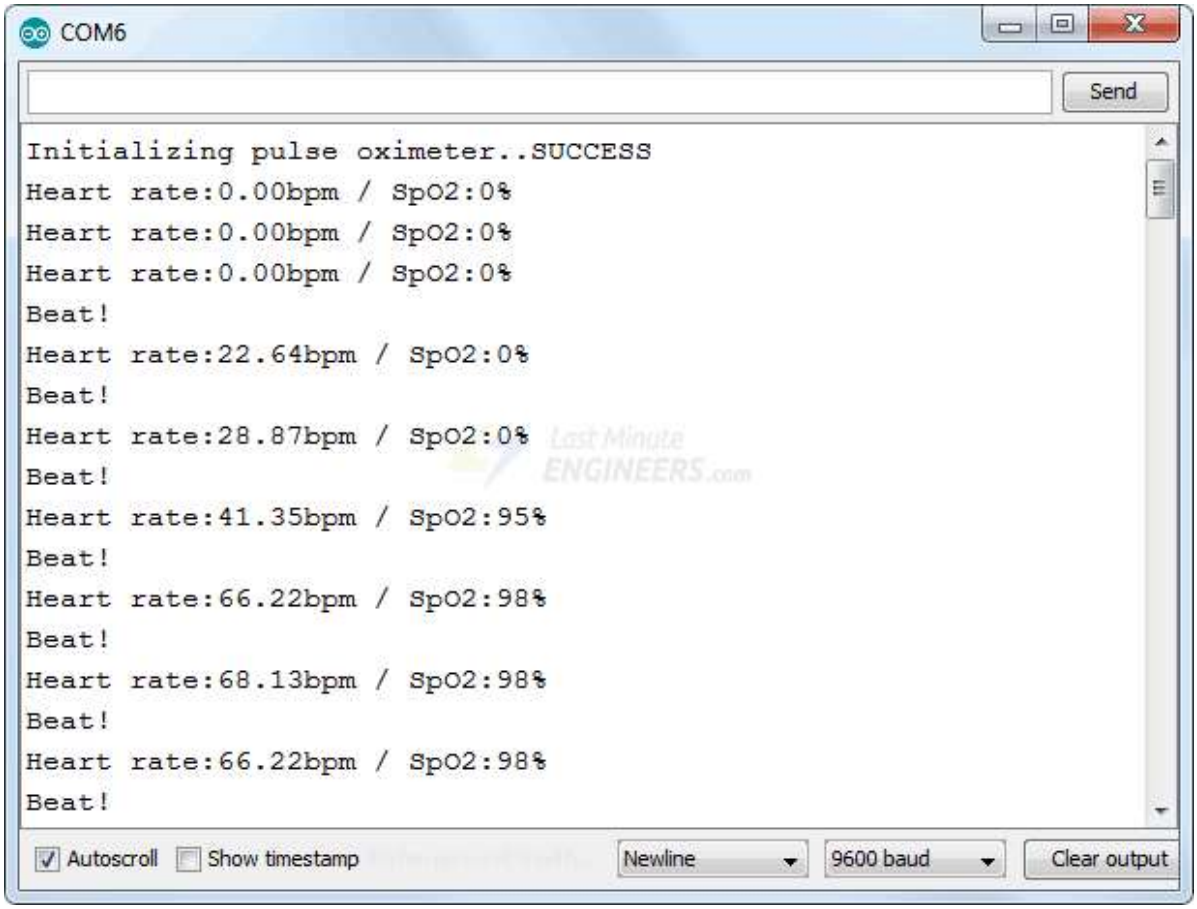
This is where the fun really begins! In this example, we'll measure heart rate (Beats Per Minute or BPM) and blood oxygen level (SpO2) of the person we're monitoring.

Warning:

This sketch detects heart-rate and oxygen saturation 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 "MAX30100_PulseOximeter.h"

#define REPORTING_PERIOD_MS     1000

// Create a PulseOximeter object
PulseOximeter pox;

// Time at which the last beat occurred
uint32_t tsLastReport = 0;

// Callback routine is executed when a pulse is detected
void onBeatDetected() {
    Serial.println("Beat!");
}

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

    Serial.print("Initializing pulse oximeter..");

    // Initialize sensor
    if (!pox.begin()) {
        Serial.println("FAILED");
        for(;;);
    } else {
        Serial.println("SUCCESS");
    }

        // Configure sensor to use 7.6mA for LED drive
        pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);

    // Register a callback routine
    pox.setOnBeatDetectedCallback(onBeatDetected);
}

void loop() {
    // Read from the sensor
    pox.update();

    // Grab the updated heart rate and SpO2 levels
    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
        Serial.print("Heart rate:");
        Serial.print(pox.getHeartRate());
        Serial.print("bpm / SpO2:");
        Serial.print(pox.getSpO2());
        Serial.println("%");

        tsLastReport = millis();
    }
}
```

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.



Code Explanation:

Let's start at the top of the example. Two libraries viz. MAX30100_PulseOximeter.h and Wire.h are included, a constant called `REPORTING_PERIOD_MS` is defined to delay the measurements, `PulseOximeter` object is created and a variable called `tsLastReport` is created to store the time at which the last beat occurred.

```
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"

#define REPORTING_PERIOD_MS     1000

PulseOximeter pox;

uint32_t tsLastReport = 0;
```

Next, a callback routine is defined which is executed when a pulse is detected. You can write some interesting code in it, for example, to watch the built-in LED blink with your heartbeat!

```
void onBeatDetected() {
    Serial.println("Beat!");
}
```

Next let's look at the setup. There are three functions to point out. First, the `pox.begin()` function call makes sure that we can communicate with the sensor. `begin()` function returns 1 on success (or 0 if it fails). It also configures the MAX30100 to begin collecting data.

```
if (!pox.begin()) {
        Serial.println("FAILED");
        for(;;);
} else {
        Serial.println("SUCCESS");
}
```

Secondly and equally as important the `setIRLedCurrent()` function sets the current through the IR LED. The following line sets the current of 7.6mA through the IR LED.

```
pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);
```

By default the library sets the IR LED current to 50mA which can sometimes cause problems like the **LED not turning on** or getting the **Initializing pulse oximeter.. FAILED** message on the serial monitor. In that case, set the LED current to a lower value. Here are the other possible values you can use on the `setIRLedCurrent()` function.

- MAX30100_LED_CURR_0MA
- MAX30100_LED_CURR_4_4MA
- MAX30100_LED_CURR_7_6MA
- MAX30100_LED_CURR_11MA
- MAX30100_LED_CURR_14_2MA
- MAX30100_LED_CURR_17_4MA
- MAX30100_LED_CURR_20_8MA
- MAX30100_LED_CURR_24MA
- MAX30100_LED_CURR_27_1MA
- MAX30100_LED_CURR_30_6MA
- MAX30100_LED_CURR_33_8MA
- MAX30100_LED_CURR_37MA
- MAX30100_LED_CURR_40_2MA
- MAX30100_LED_CURR_43_6MA
- MAX30100_LED_CURR_46_8MA
- MAX30100_LED_CURR_50MA

Remember! The higher the current, the brighter the LED and the deeper it reaches your skin. So, you can play with it while you figure it out.

Finally `setOnBeatDetectedCallback()` registers the callback function.

```
pox.setOnBeatDetectedCallback(onBeatDetected);
```

In the main loop, the biometric data is collected from the MAX30100 sensor with the `update()` function. It actually pulls data in from sensor FIFO.

```
pox.update();
```

We then print the biometric data on the serial monitor once every second (`REPORTING_PERIOD_MS` is set to 1000; change it as per your requirement). For this we use `millis()` instead of `delay()`. Since `delay()` pauses the entire code, we cannot get the updated measurement.

```
if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
        Serial.print("Heart rate:");
        Serial.print(pox.getHeartRate());
        Serial.print("bpm / SpO2:");
        Serial.print(pox.getSpO2());
        Serial.println("%");

        tsLastReport = millis();
}
```

Trouble Seeing a Heartbeat?

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

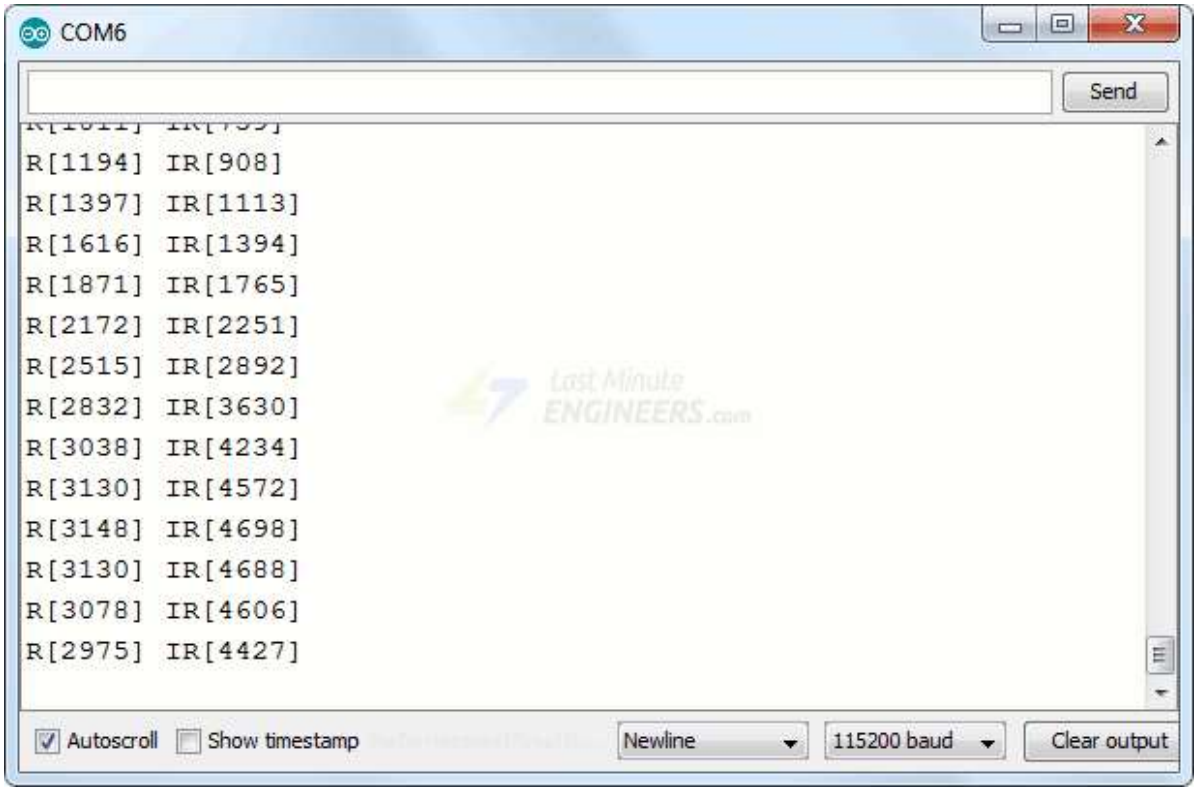
1. If you hold the sensor too hard, you will squeeze all the blood from your fingers and there will be no sign! If you hold it too lightly, you will invite noise from movement and ambient light. Sweatspot pressure (Not too hard, not too soft) on the pulse sensor will give a good clean signal.
2. A varying 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.
3. Try the sensor on different parts of your body that have capillary tissue (such as earlobe or lower lip).

Example 2 – Reading Red & IR

The second 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 "MAX30100_PulseOximeter.h"

// Create a MAX30100 object
MAX30100 sensor;

void setup() {
    Serial.begin(115200);

    Serial.print("Initializing MAX30100..");

    // Initialize sensor
    if (!sensor.begin()) {
        Serial.println("FAILED");
        for(;;);
    } else {
        Serial.println("SUCCESS");
    }

    // Configure sensor
    configureMax30100();
}

void loop() {
    uint16_t ir, red;

    sensor.update();

    while (sensor.getRawValues(&ir, &red)) {
                Serial.print("R[");
                Serial.print(red);
                Serial.print("] IR[");
                Serial.print(ir);
                Serial.println("]");
    }
}

void configureMax30100() {
  sensor.setMode(MAX30100_MODE_SPO2_HR);
  sensor.setLedsCurrent(MAX30100_LED_CURR_50MA, MAX30100_LED_CURR_27_1MA);
  sensor.setLedsPulseWidth(MAX30100_SPC_PW_1600US_16BITS);
  sensor.setSamplingRate(MAX30100_SAMPRATE_100HZ);
  sensor.setHighresModeEnabled(true);
}
```

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.

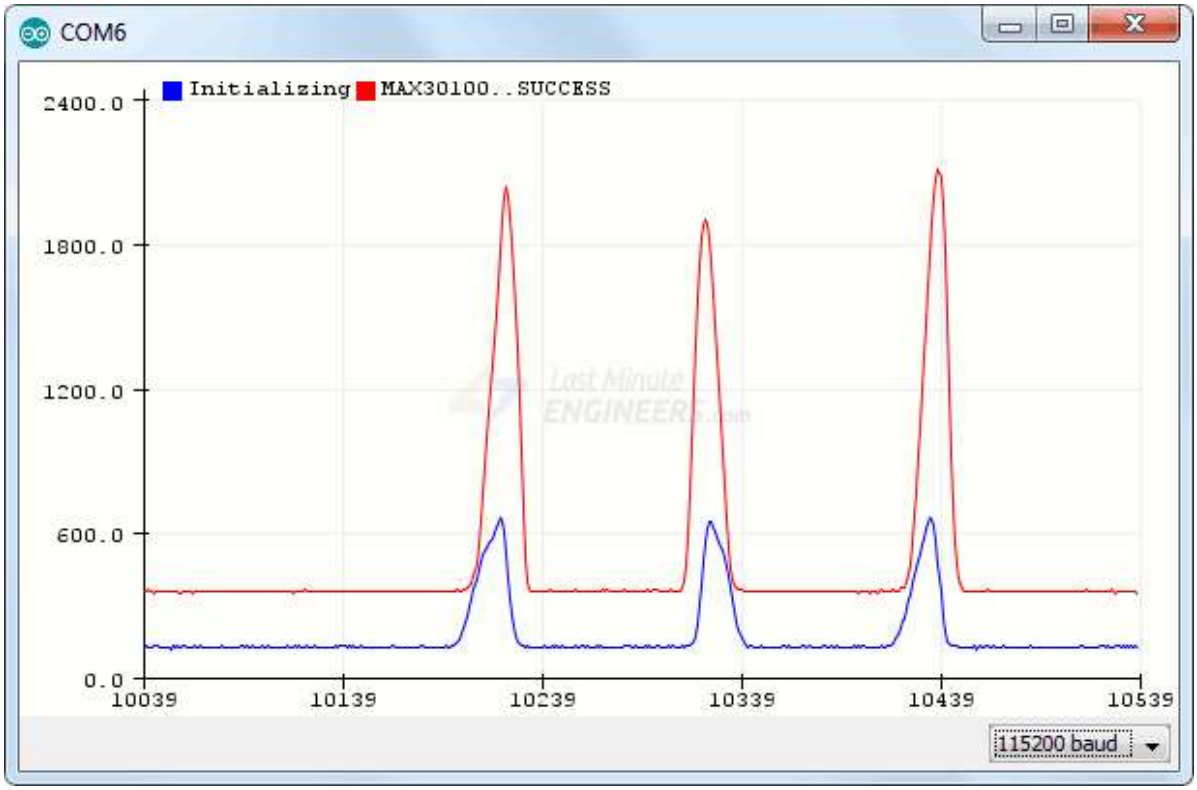


Serial data can be hard to visualize if you're only looking at values. If you are using the Arduino IDE v1.6.6+, there is an option to view the data on a graph using the **Arduino Serial Plotter**.

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

```
while (sensor.getRawValues(&ir, &red)) {
        Serial.print(red);
        Serial.print(", ");
        Serial.println(ir);
}
```

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 3 – Presence Sensing

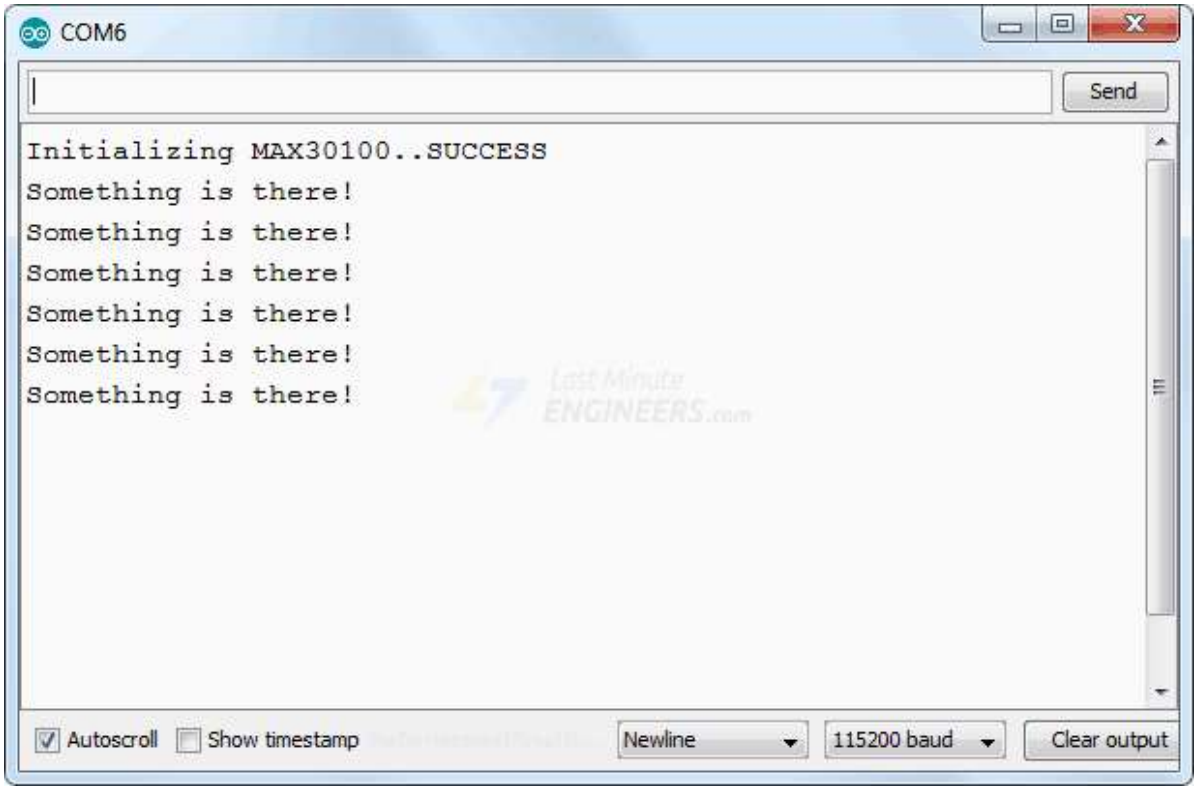
Our next experiment shows how to use the MAX30100 sensor as a general purpose proximity or a reflectance sensor and can serve as the basis for more practical experiments and projects.

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 "MAX30100_PulseOximeter.h"

#define REPORTING_PERIOD_MS     1000

int lastOccurrence = LOW;

// Create a MAX30100 object
MAX30100 sensor;

uint16_t ir, red;
uint16_t avg_ir = 0, avg_red = 0;

void setup() {
    Serial.begin(115200);

    Serial.print("Initializing MAX30100..");

    // Initialize sensor
    if (!sensor.begin()) {
        Serial.println("FAILED");
        for(;;);
    } else {
        Serial.println("SUCCESS");
    }

    configureMax30100();

    takeSampleReadings();
}

void loop() {
  sensor.update();

  while (sensor.getRawValues(&ir, &red)) {
    if (ir > 10*avg_ir && red > 10*avg_red) {
      if (lastOccurrence == LOW) {
        Serial.println("Something is there!");
        lastOccurrence = HIGH;
      }
    }
    else{
      lastOccurrence = LOW;
    }
  }
}

void configureMax30100() {
  sensor.setMode(MAX30100_MODE_SPO2_HR);
  sensor.setLedsCurrent(MAX30100_LED_CURR_50MA, MAX30100_LED_CURR_27_1MA);
  sensor.setLedsPulseWidth(MAX30100_SPC_PW_1600US_16BITS);
  sensor.setSamplingRate(MAX30100_SAMPRATE_100HZ);
  sensor.setHighresModeEnabled(true);
}

void takeSampleReadings() {
  delay(50);
  for (int i = 0; i <= 9; i++) {
    sensor.update();
    sensor.getRawValues(&ir, &red);
    avg_ir += ir;
    avg_red += red;
```

```
    delay(50);
  }
  avg_ir /= 10;
  avg_red /= 10;
}
```

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 MAX30100 is capable of reading up to 16-bits or values up to 65,536. An extremely small movement can be detected!

Example 4 – Reading Temperature

Our last 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 "MAX30100_PulseOximeter.h"

#define REPORTING_PERIOD_MS     1000

// Create a MAX30100 object
MAX30100 sensor;

// Time when the last reading was taken
uint32_t tsLastReading = 0;

void setup() {
    Serial.begin(115200);

    Serial.print("Initializing MAX30100..");

    // Initialize sensor
    if (!sensor.begin()) {
        Serial.println("FAILED");
        for(;;);
    } else {
        Serial.println("SUCCESS");
    }
```

```
    // Configure sensor
    configureMax30100();
}

void loop() {
  sensor.update();

  if (millis() - tsLastReading > REPORTING_PERIOD_MS) {
    sensor.startTemperatureSampling();
    if (sensor.isTemperatureReady()) {
      float temp = sensor.retrieveTemperature();
      Serial.print("Temperature = ");
      Serial.print(temp);
      Serial.print("*C | ");
      Serial.print((temp * 9.0) / 5.0 + 32.0);//print the temperature in Fahrenheit
      Serial.println("*F");
    }
    tsLastReading = millis();
  }
}

void configureMax30100() {
  sensor.setMode(MAX30100_MODE_SPO2_HR);
  sensor.setLedsCurrent(MAX30100_LED_CURR_50MA, MAX30100_LED_CURR_27_1MA);
  sensor.setLedsPulseWidth(MAX30100_SPC_PW_1600US_16BITS);
  sensor.setSamplingRate(MAX30100_SAMPRATE_100HZ);
  sensor.setHighresModeEnabled(true);
}
```

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