Wroclaw University Of Science And Technology

Caner Olcay (276715)

Piotr Kędzia (275540)

Mikołaj Hucz (275546)

Optoelectronics Final Project

Pulse Oximeter Design and Analysis

1. Table Of Contents

2. Introduction

The objective of this project was to design and develop a pulse oximeter capable of non-invasive measurement of blood oxygen saturation (SpO_2) and heart rate (HR). The report outlines the theoretical principles, hardware and software design, and testing methodology of the device.

3. Theoretical Background

The MAX30102 sensor has several literature-based applications showcasing its health monitoring versatility. One of its primary applications is pulse oximetry, where it measures blood oxygen saturation (SpO_2) levels in real-time. This is crucial in medical devices for monitoring patients with respiratory or cardiac conditions, as it ensures adequate oxygen delivery in clinical and wearable settings. Studies on noninvasive pulse oximetry highlight its accuracy, even under challenging conditions such as motion artifacts.

Another significant application of the MAX30102 is in heart rate monitoring using photoplethysmography (PPG). The sensor detects variations in blood volume during each cardiac cycle, providing a reliable measurement of heart rate. This is particularly useful in fitness tracking and stress monitoring devices, which benefit from the sensor's ability to capture real-time cardiovascular data during both physical activity and rest.

Additionally, the MAX30102 is employed in sleep apnea detection. Analyzing irregularities in oxygen saturation and heart rate patterns during sleep helps identify conditions such as sleep apnea, characterized by disrupted breathing. This application is commonly found in wearable devices designed for at-home sleep studies, reducing the reliance on expensive clinical polysomnography.

Finally, the sensor plays a role in stress and emotional monitoring. Variations in blood oxygen levels and heart rate variability (HRV) indicate stress and emotional states. Devices using the MAX30102 leverage this data to provide insights into mental health, suggest relaxation techniques, or monitor chronic stress conditions. Research has shown strong correlations between PPG-derived metrics and psychological states.

The operation of the MAX30102 relies on the physical phenomenon of photoplethysmography (PPG), a non-invasive optical method. This technique involves the emission of light, typically infrared and red, into the skin. Oxygenated hemoglobin in the blood absorbs more infrared light, while deoxygenated hemoglobin absorbs more red light. The sensor detects either the transmitted or reflected light, and variations in the detected light correspond to changes in blood volume within the capillaries. These periodic changes are processed to extract cardiovascular metrics such as heart rate and oxygen saturation. The PPG waveform, modulated by the cardiac cycle, provides precise data while advanced algorithms account for potential interferences like motion artifacts or ambient light.

4. Assumptions

Functional Assumptions

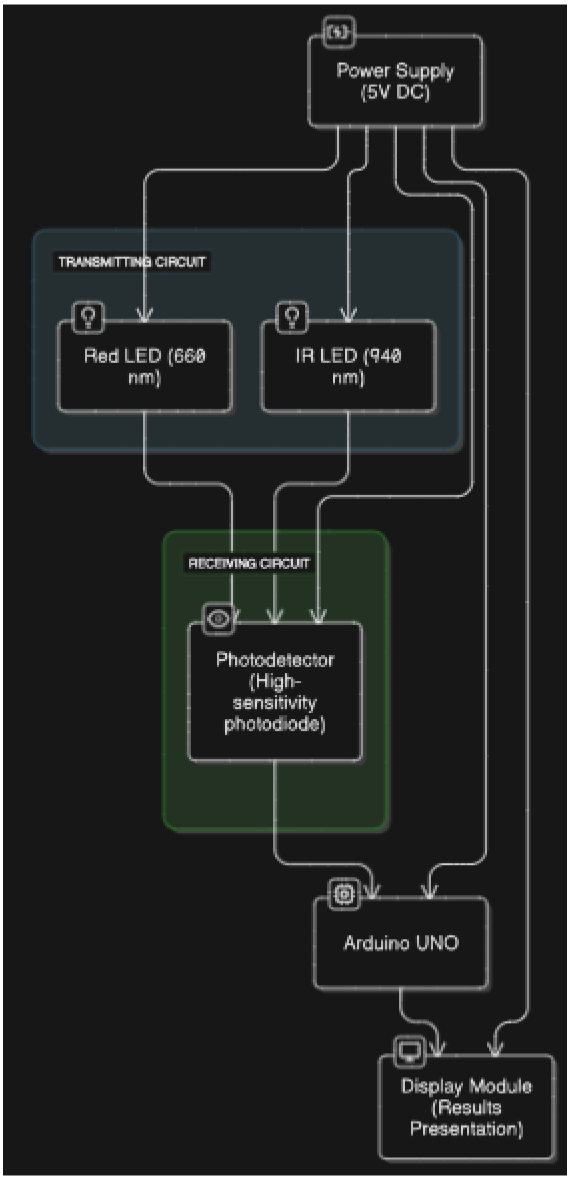
- The device measures blood oxygen saturation (SpO_2) and heart rate (HR) in real-time.
- Both transmittance and reflectance measurement modes are supported.

Design Assumptions

- The system uses LEDs at 660 nm and 940 nm for light emission.
- A photodetector captures the intensity of transmitted or reflected light.
- Signal processing algorithms filter noise and calculate physiological parameters.

5. Description of the Hardware

Block Diagram:



Block Diagram of the Pulse Oximeter

1. Red and IR LEDs (Light Sources)
2. Photodetector (Light Receiver)
3. Microcontroller (Signal Processing)
4. Display Module (Results Presentation)

Mechanical and Electrical Components

- LEDs: 660 nm (Red) and 940 nm (IR)
- Photodetector: High-sensitivity photodiode
- Microcontroller: STM32
- Power Supply: 5V DC

PCB Design and Assembly

The PCB includes LED drivers, photodetector signal amplifiers, and connections to the microcontroller.

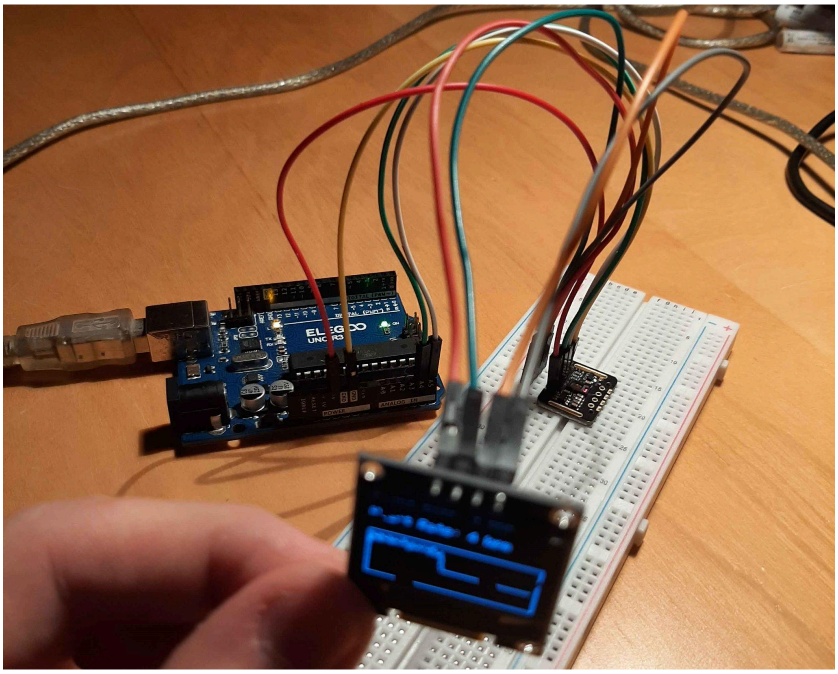


Fig. 1. Breadboard Assembly with MAX30102 Sensor and Oled Display 0,96 SSD1306

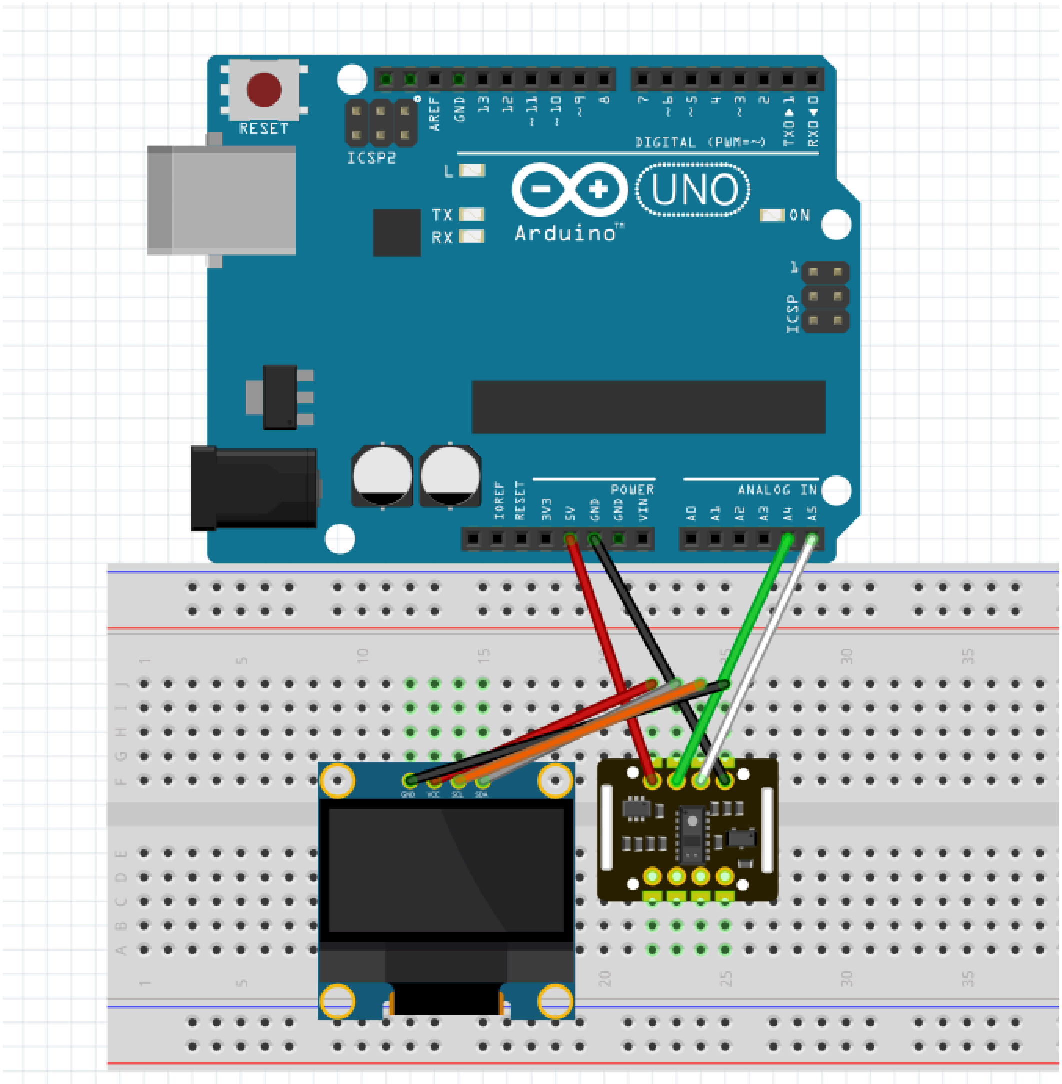


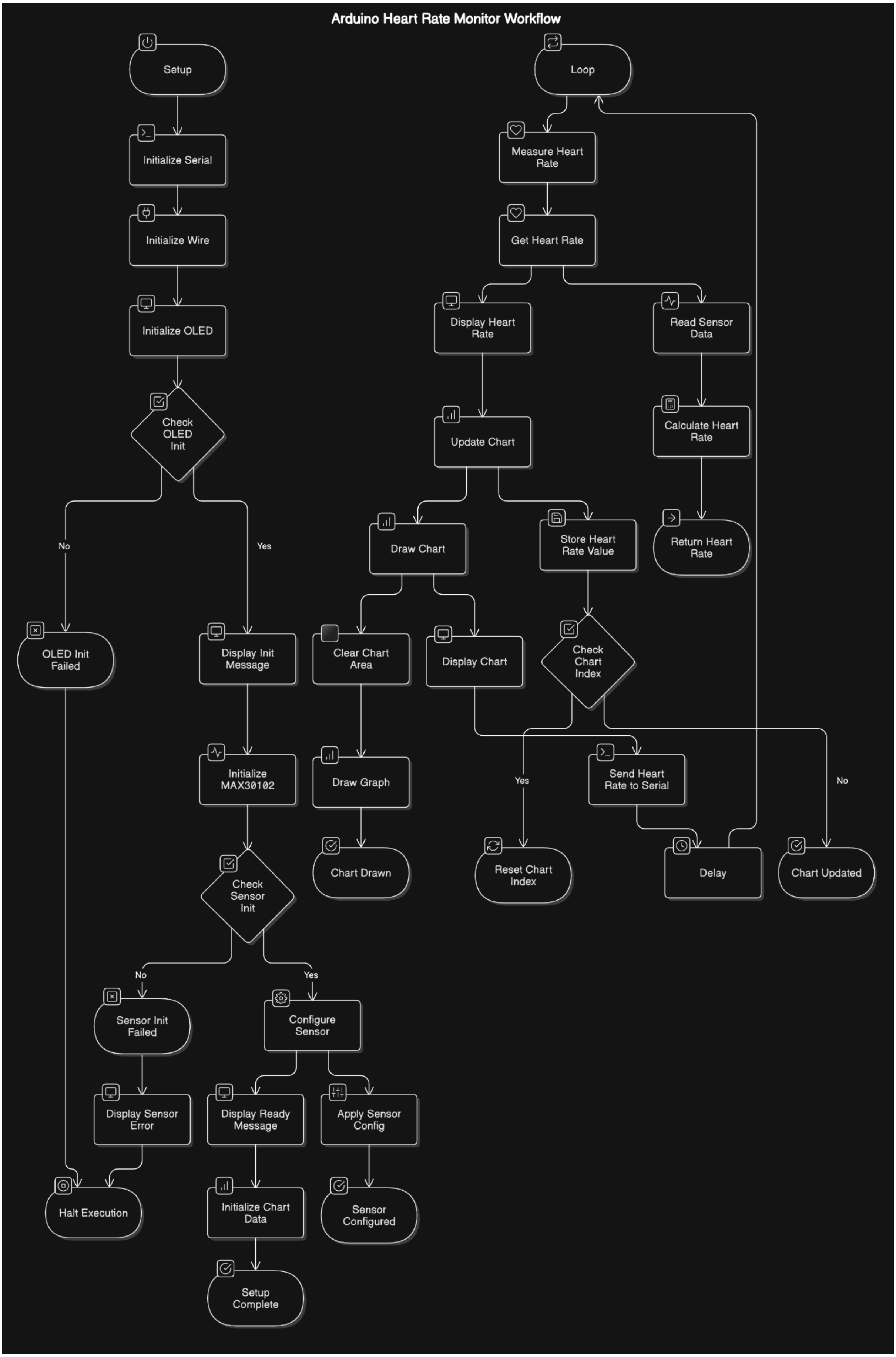
Fig 2. All connections between the components, made using fritzing software

6. Description of the Software

Main Algorithm

The algorithm processes photodetector signals and calculates SpO_2 and HR. Key steps:

1. Signal acquisition from photodetector.
2. Calculation of the red/IR absorption ratio.
3. SpO_2 and HR computation using empirical formulas.


Arduino Heart Rate Monitor Workflow
Setup
Initialize Serial
Initialize Wire
Initialize OLED
Check OLED Init
No
Yes
OLED Init Failed
Display Init Message
Initialize MAX30102
Check Sensor Init
No
Yes
Sensor Init Failed
Configure Sensor
Display Sensor Error
Display Ready Message
Apply Sensor Config
Halt Execution
Initialize Chart Data
Sensor Configured
Setup Complete
Loop
Measure Heart Rate
Get Heart Rate
Display Heart Rate
Read Sensor Data
Update Chart
Calculate Heart Rate
Draw Chart
Store Heart Rate Value
Return Heart Rate
Clear Chart Area
Display Chart
Check Chart Index
Draw Graph
Yes
Send Heart Rate to Serial
No
Chart Drawn
Reset Chart Index
Delay
Chart Updated

Key Functions and Code

Listing 1. Signal Processing Code Example:

This function collects sensor data, processes it, and stores results.

```
void loop()
{
  bufferLength = 100; // buffer length of 100 stores 4 seconds of samples

  // Read first 100 samples
  for (byte i = 0; i < bufferLength; i++)
  {
    while (particleSensor.available() == false)
      particleSensor.check(); // Wait for new data

    redBuffer[i] = particleSensor.getRed();
    irBuffer[i] = particleSensor.getIR();
    particleSensor.nextSample();
  }

  // Calculate SpO2 and Heart Rate
  maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer, &spo2,
&validSPO2, &heartRate, &validHeartRate);
}
```

Key Variables:

bufferLength - set to 100, representing the number of samples to read from the MAX30102 sensor. Stores approximately 4 seconds of data based on the sampling rate.

redBuffer[] - array to store red light readings from the MAX30102 sensor. Used for SpO2 calculations.

irBuffer[] - array to store infrared (IR) light readings from the MAX30102 sensor. Used for heart rate and SpO2 calculations.

spo2 - stores the calculated oxygen saturation percentage (SpO2).

validSPO2 - a flag indicating whether the calculated SpO2 value is valid (1 for valid, 0 for invalid).

heartRate - stores the calculated heart rate in beats per minute (BPM).

validHeartRate - a flag indicating whether the calculated heart rate is valid (1 for valid, 0 for invalid).

***This code is given in the official examples provided by the author of the library designed for our sensor, here is also more detailed info about formulas, etc. ***

Link to official repository/documentation:
https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library

Transmission Protocols

Data is transmitted via I2C to the display module.

We made simple I2C scanner to confirm at which address, our devices are working:

This function is used to detect and identify all I2C devices connected to the system. It helps verify the correct wiring and confirms the availability of devices.

```
void setup() {
  Serial.begin(115200);
  Wire.begin();
  Serial.println("\nI2C Scanner");

  for (byte i = 8; i < 120; i++) {
    Wire.beginTransmission(i);
    if (Wire.endTransmission() == 0) {
      Serial.print("Found I2C device at address 0x");
      if (i < 16) {
        Serial.print("0");
      }
      Serial.println(i, HEX);
    }
  }
}
```

```
18:31:43.017 -> I2C Scanner
18:31:43.017 -> Found I2C device at address 0x3C
18:31:43.017 -> Found I2C device at address 0x57
```

this part was additional troubleshooting, which totally aligned with our expected results, as given by the author of library made to handle the display -

```
#define SCREEN_ADDRESS 0x3D ///< See datasheet for Address; 0x3D for
128x64, 0x3C for 128x32
```

Final Code

However in the final code in which oled displayed functionalities, required some changes. We approached unanticipated errors mainly related to exceeding usable memory:

Sketch uses 20484 bytes (63%) of program storage space. Maximum is 32256 bytes. Global variables use 2204 bytes (107%) of dynamic memory, leaving -156 bytes for local variables. Maximum is 2048 bytes.

We observed the bizarre output, even while optimizing the memory used in the code, the display refused cooperating. This was more than strange to us, as the example given by the author worked completely alright(of course we didn't change anything in the connections):
**15:56:27.054 -> Initializing OLED display…*
*15:56:27.054 -> SSD1306 allocation failed - Check OLED connections and address.**

Due to this unexpected error we slightly changed(reducing memory used) our approach and skipped SPO2 measurement and focused mainly on the BMP part:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "MAX30105.h"

// OLED definitions
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET    -1  // Set to -1 for automatic reset
#define SCREEN_ADDRESS 0x3C

// Create an instance for the OLED and MAX30102 sensor
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);
MAX30105 particleSensor;

// Heart rate variables
int32_t heartRate = 0;         // Heart rate value
int8_t validHeartRate = 0;     // Heart rate validity flag

// Chart data array (to store heart rate values)
#define MAX_POINTS 123
int chartData[MAX_POINTS];
```

```
int chartIndex = 0;

void setup() {
  Serial.begin(115200);  // Start serial communication for debugging
  Wire.begin();

  // Initialize OLED
  if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    while (1);  // Halt execution if OLED initialization fails
  }

  // Initialize the OLED display
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println(F("Initializing MAX30102..."));
  display.display();

  // Initialize MAX30102 sensor
  if (!particleSensor.begin(Wire, I2C_SPEED_STANDARD)) {
    Serial.println(F("MAX30102 not detected. Check wiring."));
    display.clearDisplay();
    display.setCursor(0, 0);
    display.println(F("MAX30102 error!"));
    display.display();
    while (1);  // Halt execution if the sensor initialization fails
  }

  // Configure the MAX30102 sensor
  configureSensor();

  // Display ready message on OLED
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println(F("MAX30102 Ready!"));
  display.display();

  Serial.println(F("Setup complete!"));

  // Initialize chart data
  for (int i = 0; i < MAX_POINTS; i++) {
    chartData[i] = 0;
  }
}

void loop() {
  // Measure heart rate
  heartRate = getHeartRate();

  // Display the heart rate on the OLED
  display.clearDisplay();
  display.setCursor(0, 0);
  display.print(F("Heart Rate: "));
  display.print(heartRate);
  display.println(F(" bpm"));
```

```
  // Update chart with heart rate value
  updateChart(heartRate);

  // Display the chart
  drawChart();

  display.display();

  // Send the heart rate to the Serial Plotter
  Serial.print("HeartRate: ");
  Serial.println(heartRate);

  // Shorten the delay to update more frequently
  delay(200);  // Adjust delay to 200 ms for more frequent updates
}

// Function to configure MAX30102 sensor
void configureSensor() {
  byte ledBrightness = 60; // Adjust LED brightness (0-255)
  byte sampleAverage = 4;   // Average samples (1, 2, 4, 8, 16, 32)
  byte ledMode = 2;         // LED mode: 2 = Red + IR
  byte sampleRate = 100;    // Sampling rate in Hz (50, 100, 200, etc.)
  int pulseWidth = 411;     // Pulse width (69, 118, 215, 411)
  int adcRange = 4096;      // ADC range (2048, 4096, 8192, 16384)

  // Apply the sensor configuration settings
  particleSensor.setup(ledBrightness, sampleAverage, ledMode,
sampleRate, pulseWidth, adcRange);
  Serial.println(F("Sensor configured successfully"));
}

// Function to get heart rate from MAX30102
int32_t getHeartRate() {
  uint16_t irBuffer[100];   // Infrared data buffer
  uint16_t redBuffer[100];  // Red data buffer
  int bufferLength = 100;   // Data length

  // Read the first 100 samples
  for (byte i = 0; i < bufferLength; i++) {
    while (particleSensor.available() == false) { // Check for new data
      particleSensor.check();
    }
    redBuffer[i] = particleSensor.getRed();
    irBuffer[i] = particleSensor.getIR();
    particleSensor.nextSample(); // Move to the next sample
  }

  // DEBUG: Print the raw sensor readings to the serial monitor
  Serial.print("Red: ");
  Serial.print(redBuffer[0]);
  Serial.print("  IR: ");
  Serial.println(irBuffer[0]);

  // Now we will calculate the heart rate from the raw values
  // Let's use a basic method here for debugging:
  int32_t calculatedHeartRate = calculateHeartRate(redBuffer, irBuffer,
bufferLength);
```

```
  // Return the heart rate or 0 if invalid
  return calculatedHeartRate;
}

// Simple calculation method for heart rate (based on signal
fluctuations)
int32_t calculateHeartRate(uint16_t* redBuffer, uint16_t* irBuffer, int
length) {
  // In a real implementation, you would use algorithms to find peaks
in the pulse signal
  int32_t heartRate = 0;

  // Basic peak detection algorithm (you can replace this with more
sophisticated ones)
  for (int i = 1; i < length - 1; i++) {
    if (irBuffer[i] > irBuffer[i - 1] && irBuffer[i] > irBuffer[i + 1])
{
      heartRate++; // Count peaks
    }
  }

  // Estimate heart rate based on peak count
  heartRate = heartRate * 60 / (length / 2);  // Simple estimate of BPM

  return heartRate;
}

// Function to update chart with heart rate value
void updateChart(int value) {
  chartData[chartIndex] = value;  // Store new heart rate value
  chartIndex++;
  if (chartIndex >= MAX_POINTS) {
    chartIndex = 0;  // Reset chart index when full
  }
}

// Function to draw the chart
void drawChart() {
  // Draw the chart area (clear the chart area)
  display.drawRect(0, 20, 128, 44, SSD1306_WHITE);  // Chart border

  // Draw the graph by connecting the points
  for (int i = 1; i < MAX_POINTS; i++) {
    int x1 = i - 1;
    int y1 = map(chartData[x1], 0, 100, 0, 44); // Map value to chart
height
    int x2 = i;
    int y2 = map(chartData[x2], 0, 100, 0, 44); // Map value to chart
height
    display.drawLine(x1, 64 - y1 - 20, x2, 64 - y2 - 20,
SSD1306_WHITE);  // Draw line
  }
}
```

With this code, we made quite a readable display, showing measurement data in real-time as well on the simple chart updating around 10 seconds indefinitely

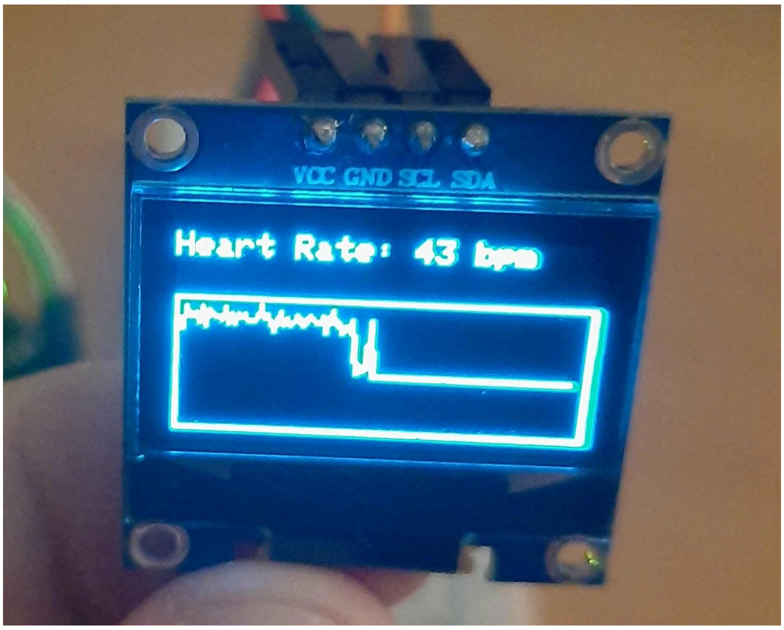


Fig. 2. Output after sometime

Here the only issue might be adjusting the LED brightness to our current environment, aside from that, we expected more accurate measurement. The cause of that might be from not-so-stable connections of the whole circuit or not positioning our finger properly.

7. Start-Up and Calibration

First Start-Up

1. Connect the device to a 5V power source.
2. Ensure LEDs and photodetectors are operational.

Calibration

1. Use a standard calibration solution or a controlled test subject.
2. Adjust signal amplification and threshold levels for optimal readings.

8. Test Measurements

Data Analysis

Measurement conditions include all parameters that might have an effect on the resulting data. In our case, we tested those:

-The placement of the sensor e.g finger
-Gentle but firm contact between the sensor and the skin
-Keep the sensor and the body part being measured as still as possible during readings.
-The MAX30102 is sensitive to light leakage, and external light can affect its ability to detect IR and red signals accurately.
-Ensure a quiet environment with minimal mechanical vibrations, which can affect stability and accuracy.
-Adjust the LED brightness to balance signal strength and power consumption.

Table 1. Sample Data Collected

IR	BPM	Avg BPM
123867	55.71	56
123048	60.12	61
122352	71.68	64

Table 2. Logged Measurement Data

	Red	Ir	HR	HRvalid	SPO2	SPO2Valid
13:59:39	65535	45169	166	1	76	1
13:59:40	65535	45708	166	1	76	1
13:59:41	65535	45403	166	1	76	1

9. Technical Specification

Power Supply: 5V DC

LEDs: 660 nm (Red) and 940 nm (IR)

Photodetector: High-sensitivity photodiode

Current Consumption:

Entire system: 120 mA

Arduino Uno: 50mA

MAX30102 sensor: 50 mA

Display: 20mA

10. User Manual

Steps to Operate:

1. Place the sensor on the desired body part (e.g., fingertip or wrist).
2. Power on the device.
3. Wait for the results to display.

11. Summary

The pulse oximeter successfully measured SpO_2 and HR with alright accuracy. Future improvements could focus on enhancing signal processing algorithms and ergonomic sensor designs.

12. Bibliography

[1] Principles of Pulse Oximetry. J. Clin. Monit. Comput., 2018.
[2] Photoplethysmography in Biomedical Engineering, Springer, 2017.

13. Appendix

Sources:

https://lastminuteengineers.com/max30102-pulse-oximeter-heart-rate-sensor-arduino-tutorial/?utm_content=cmp-true

https://www.instructables.com/Arduino-and-the-SSD1306-OLED-I2C-128x64-Display/

https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library