PROJECT OVERVIEW

1. Objective

Develop a wireless pulse monitoring system using two Raspberry Pi (RPi) units. The system captures pulse signals via a sensor, transmits the data over Bluetooth, and displays the pulse data along with key health metrics on a Graphical User Interface (GUI).

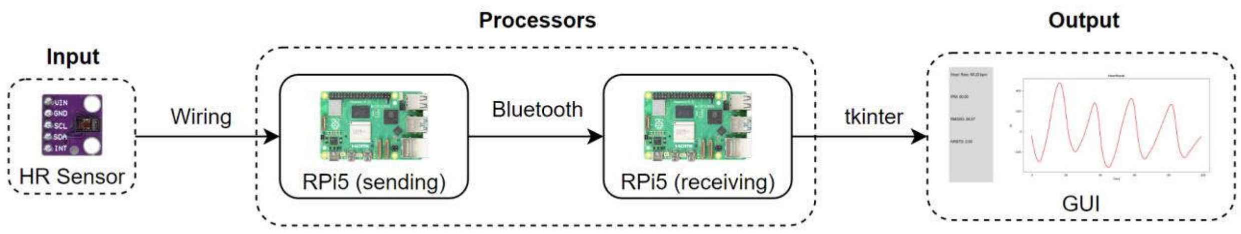


Figure 1. System block diagram

2. Project Phases and Breakdown

i. Setup and sensor integration

- Understand the pulse sensor module's working.
- Configure the first RPi to connect with the pulse sensor module.

ii. Data transmission

- Set up a Bluetooth connection between the two RPi units.
- Program the first RPi to transmit the collected pulse data wirelessly to the second RPi via Bluetooth.
- Calculate and present health metrics:
    - BPM (Beats Per Minute)
    - IPM (Impulses Per Minute)

- HRSTD (Heart Rate Standard Deviation)
- RMSSD (Root Mean Square of Successive Differences

iii. GUI development

- Develop a GUI on the second RPi to display the transmitted pulse data.

SETUP AND SENSOR INTEGRATION

1. Equipment List

Items	Model	Quantity	Power Supply
Raspberry Pi	RPi 5	2	5 VDC (27W)
Heart Rate Sensor	GY-MAX30102	1	3.3 to 5 VDC

2. Understanding MAX 30102 Sensor

i. Brief overview

The MAX30102 is a pulse oximetry and heart rate biosensor module based on PPG (i.e., Photo Plethysmo Graphy). It incorporates the advanced MAX30102 IC, an upgraded version of the MAX30100, designed for accurate Heart Rate (HR) and Oxygen Saturation (SpO2) measurements (i.e., SpO2 measurements are not required for this project).

ii. Key components

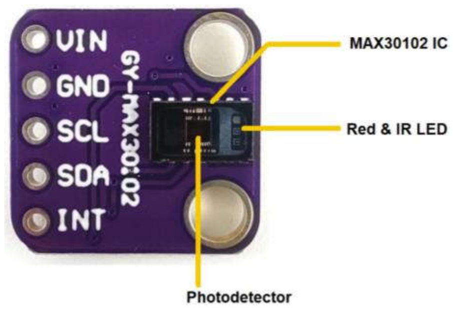


Figure 2. Key components of the MAX30102 sensor

The module features two LEDs, including a red LED and an infrared (IR) LED, a photodetector, optimized optics, and low-noise analog signal processing. It communicates via the I2C protocol, making it compatible with microcontrollers such as Arduino, ESP32, STM32, and Raspberry Pi.

iii. Working principles

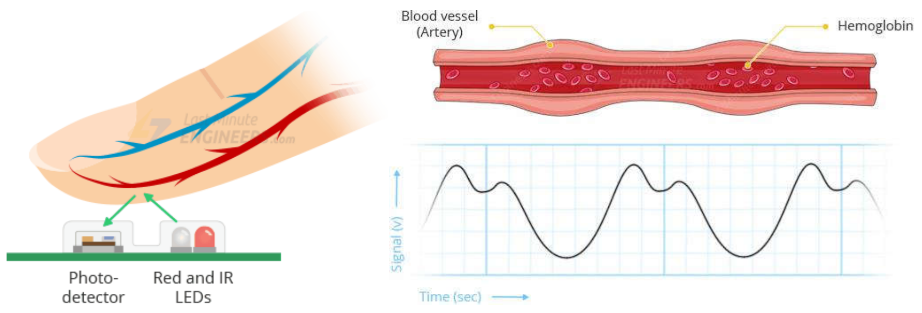


Figure 3. Illustrations of working principles (lastminuteengineers.com)

The MAX30102 heart rate sensor measures heart rate by detecting changes in light absorption caused by blood flow in tissues. It uses IR and red LEDs to emit light that penetrates the skin and reflects off blood vessels.

Photodiodes capture the reflected light, with the IR LED reaching deeper layers and the red LED targeting surface-level blood flow. The sensor converts these light signals into analog data, which is amplified and digitized using an integrated analog-to-digital converter (ADC) for processing.

iv. Wiring Schematic

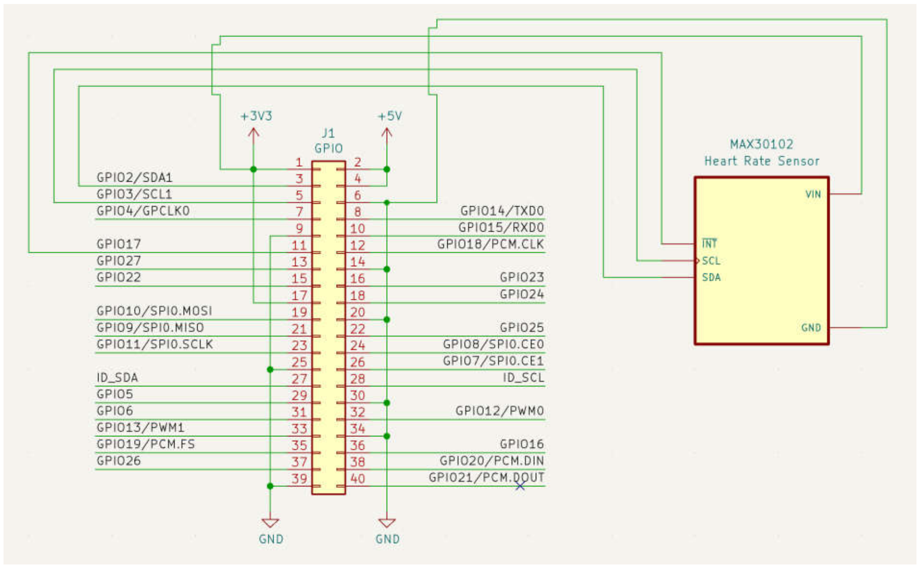


Figure 4. Schematic for the first RPi 5 and the MAX30102 sensor

- RPi 5 → MAX30102 sensor module:
  - 3.3V → Vin
  - GPIO2/SDA → SDA
  - GPIO3/SCL → SCL
  - GPIO17 → INT

DATA PROCESSING

1. Data Acquisition

Infrared (IR) data is read from the MAX30102 sensor continuously in batches of 100 samples. The sampling frequency (f_s) is set to **25 Hz**, which matches the sensor's capabilities. A sliding window buffer with a size of 100 samples (equivalent to 4 seconds of data at 25 Hz) is implemented using a deque to store the most recent data for analysis.

Every 500 milliseconds, the buffer is updated with new data, and the metrics are recalculated.

2. Bandpass Filtering

The raw IR data is filtered using a Butterworth bandpass filter to eliminate noise and retain the physiological signal. The filter has the following characteristics:

- Low cutoff frequency (f_{low}): 0.5 Hz
- High cutoff frequency (f_{high}): 3.0 Hz
- Order: 1

The bandpass filter is applied using the forward-backward filtering technique to ensure zero phase distortion.

3. Feature Extraction

i. Peak Detection

Peaks in the filtered IR signal represent heartbeat occurrences. The peaks are detected using the `scipy.signal.find_peaks` function, with a minimum distance between peaks set to $\frac{f_s}{2.5}$.

This ensures the detection of peaks corresponding to heart rates up to 150 beats per minute (bpm).

ii. Inter-Beat Interval (IBI) Calculation

The Inter-Beat Interval (IBI) is the time difference between consecutive peaks, calculated as:

$$\text{IBI (ms)} = \frac{\text{Difference in peak indices}}{f_s} \times 1000$$

4. Metrics Computation

1. **Heart Rate (HR):** The average heart rate in beats per minute (bpm) is computed as:

$$\text{HR (bpm)} = \frac{60}{\text{Average IBI (s)}}$$

2. **Impulses Per Minute (IPM):** The number of peaks (heartbeats) per minute is calculated using the duration of the sliding window:

$$\text{IPM} = \frac{\text{Number of peaks}}{\text{Window duration (minutes)}}$$

3. **Heart Rate Standard Deviation (HRSTD):** The standard deviation of heart rate values, derived from the inverse of IBIs:

$$\text{HRSTD} = \sqrt{\frac{\sum_{i=1}^{N} \left(\frac{60}{\text{IBI}_i} - \overline{\text{HR}} \right)^2}{N}}$$

4. **Root Mean Square of Successive Differences (RMSSD):** RMSSD is a measure of HRV and is computed as:

$$\text{RMSSD} = \sqrt{\frac{\sum_{i=1}^{N-1} (\text{IBI}_{i+1} - \text{IBI}_i)^2}{N-1}}$$

DATA TRANSMISSION

1. Establishing Bluetooth Pairing Between Raspberry Pi Units

Before running the program, a Bluetooth pairing is established between the two Raspberry Pi's.

i. Steps for Receiving Raspberry Pi

The Receiver Pi is configured to act as a server, making it discoverable and pairable. The following steps are performed:

1. Start the Bluetooth Service:

```
sudo systemctl start bluetooth
sudo hciconfig hci0 up
```

2. Configure Bluetooth Settings by launching the Bluetooth command-line interface:

```
bluetoothctl
```

3. Make the device discoverable and allow pairing:

```
power on            # Turn on Bluetooth
agent on            # Enable the agent to manage pairing
default-agent       # Set the current agent as the default
discoverable on     # Make the server visible to clients
pairable on         # Allow the server to be paired with
```

ii. Steps for Sending Raspberry Pi

The Sender Pi is configured to act as a client, searching for and pairing with the Receiver Pi.

1. Start the Bluetooth Service:

```
sudo systemctl start bluetooth
sudo hciconfig hci0 up
```

2. Configure Bluetooth Settings by launching the Bluetooth command-line interface:

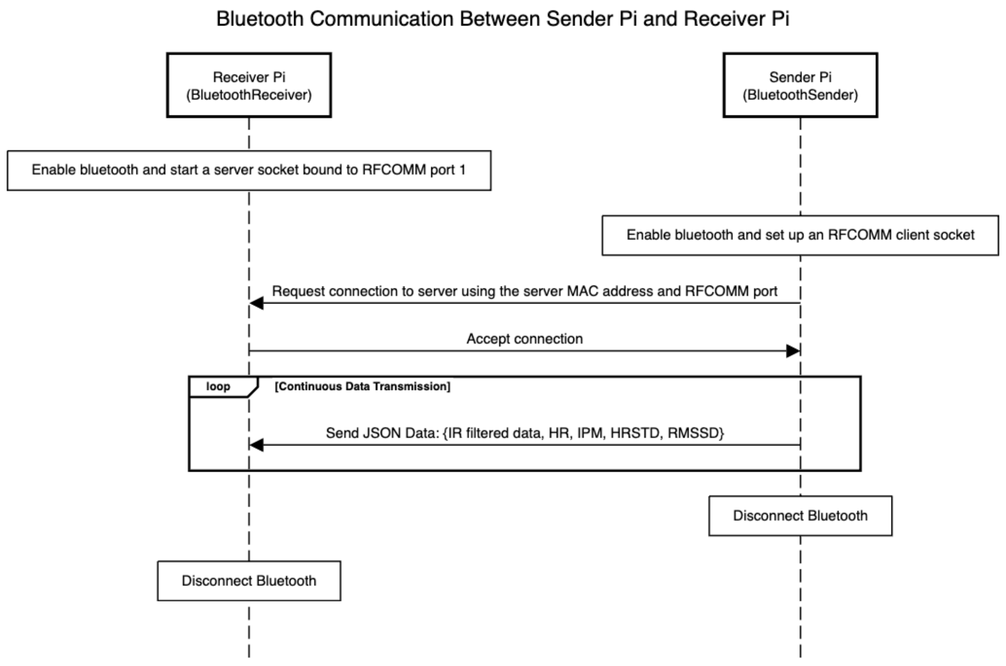
```
bluetoothctl
```

3. Scan for devices and the pair, trust and connect to receiver raspberry pi's MAC address:

```
power on              # Turn on Bluetooth
agent on              # Enable the agent to manage pairing
default-agent         # Set the current agent as the default
scan on              # Scan for devices
pair XX:XX:XX:XX:XX:XX # Pair to the MAC address of receiver
trust XX:XX:XX:XX:XX:XX # Trust the MAC address of receiver
connect XX:XX:XX:XX:XX:XX # Connect to the MAC address of receiver
```

2. Transmitting Bluetooth Messages

Once the two Raspberry Pi's are paired, the messages are transmitted through the python program using the `pybluez` library. The figure below shows the sequence of steps performed for Bluetooth communication between the Pi's.



Bluetooth Communication Between Sender Pi and Receiver Pi
Receiver Pi (BluetoothReceiver)
Sender Pi (BluetoothSender)
Enable bluetooth and start a server socket bound to RFCOMM port 1
Enable bluetooth and set up an RFCOMM client socket
Request connection to server using the server MAC address and RFCOMM port
Accept connection
loop
[Continuous Data Transmission]
Send JSON Data: {IR filtered data, HR, IPM, HRSTD, RMSSD}
Disconnect Bluetooth
Disconnect Bluetooth

Figure 5. Bluetooth communication flowchart between Sender and Receiver

iii. Initial Setup

- **Receiver Pi**:
  - The Receiver Pi initializes its Bluetooth adapter and starts a server socket bound to RFCOMM port 1. This step enables the Receiver Pi to accept incoming connections from the Sender Pi.
- **Sender Pi**:
  - The Sender Pi activates its Bluetooth adapter and sets up a client socket configured to connect to the Receiver Pi's MAC address and RFCOMM port.

iv. Connection Establishment

- The Sender Pi sends a connection request to the Receiver Pi using its Bluetooth MAC address and RFCOMM port as identifiers.
- The Receiver Pi accepts the connection, establishing a secure communication channel.

v. Data Transmission

- Once the connection is established, the Sender Pi begins transmitting JSON-encoded health metrics and sensor data to the Receiver Pi in real time.
- The transmitted data includes:
  - **IR Filtered Data**: Processed raw sensor values.
  - **Health Metrics**:
    - Heart Rate (HR)
    - Impulses Per Minute (IPM)
    - Heart Rate Standard Deviation (HRSTD)
    - Root Mean Square of Successive Differences (RMSSD)

vi. Disconnection and Cleanup

- After the data transmission is complete, both devices disconnect their Bluetooth connections to release resources.
- The Receiver Pi and Sender Pi disable their respective Bluetooth adapters to conserve power and ensure clean shutdown.

GUI DEVELOPMENT

1. Implementation

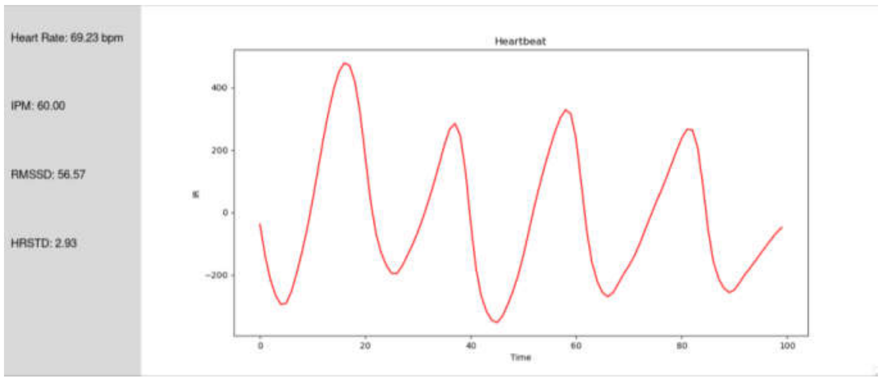
- Uses tkinter, matplotlib
- Update GUI every second

2. Received Data Format (JSON)

The GUI receives data in the following JSON format:

- **BPM**: Beats Per Minute.
- **IPM**: Impulses Per Minute.
- **RMSSD**: Root Mean Square of the Successive Differences.
- **HRSTD**: Heart Rate Standard Deviation.
- **raw_values**: Array of raw sensor readings.

3. GUI Configuration


Heart Rate: 69.23 bpm

IPM: 60.00

RMSSD: 56.57

HRSTD: 2.93

Figure 6. Graphical user interface of the system

- **Graph**: Displays IR data for real-time visualization.
- **Labels**: Four data points are shown on the left side:
    - **Heart Rate**: Value in BPM.
    - **IPM**: Impulses per minute.
    - **RMSSD**: Statistical measure of heart rate variability.
    - **HRSTD**: Heart rate standard deviation