Smart Health Monitoring System For Cyclist Using IoT And Variable Sensor

SUMMER INTERNSHIP PROJECT REPORT

(V-SRI 2025)

Submitted in partial fulfillment summer internship

by

Shaik Sameer

INDIAN INSTITUTE OF TECHNOLOGY PATNA

Under the Guidance of

Prof Dr. Swati Shukla Mam

VIT-AP UNIVERSITY

SCHOOL OF ELECTRONICS ENGINEERING

VIT-AP UNIVERSITY

AMARAVATI-522237

JULY 2025

CERTIFICATE

This is certify that the internship Project work titled **"Smart Health Monitoring System For Cyclist Using IoT And Variable Sensor"** that is being submitted **by Shaik Sameer** is in partial fulfilment of the requirements for the summer internship work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Internship Mentor

Project work is satisfactory / unsatisfactory

Approved by

DEAN

School of Electronics Engineering

Contents

Objective:

To design and develop a low-cost, portable, and wireless health monitoring system capable of measuring heart rate and blood pressure, with real-time data transmission to a mobile phone using Wi-Fi.

Introduction:

This project focuses on building an IoT-based health monitoring device using an ESP32 microcontroller and the MAX30102 sensor module. The system is designed to non-invasively measure heart rate and approximate blood pressure values by analysing photoplethysmography (PPG) signals. It is powered by three 3.7V lithium-ion cells connected in series, delivering a total of 11.1V. This voltage is regulated to 5V using an LM2596 buck converter to safely power the ESP32 and sensor.

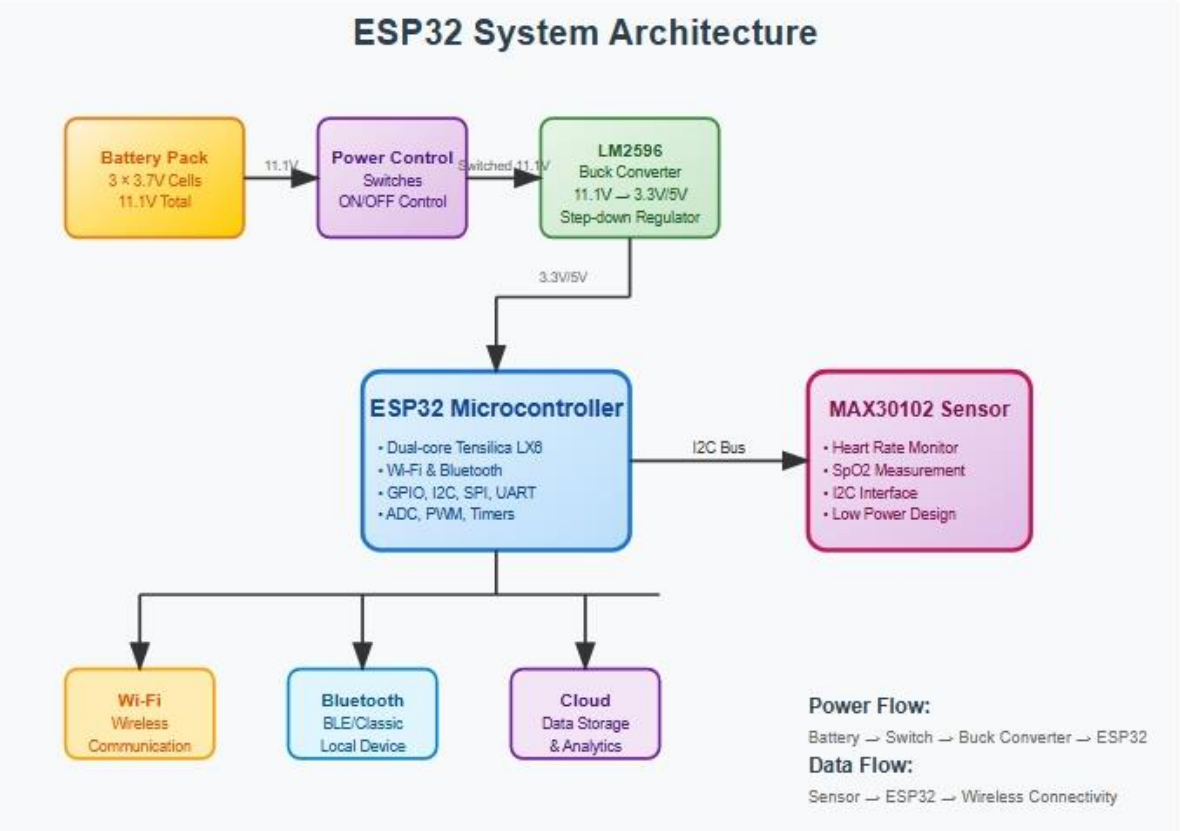
A pair of switches are used to control the power supply from the battery to the system, allowing efficient energy management. The ESP32 reads data from the MAX30102 sensor over the I2C interface and processes it. The processed heart rate data is then transmitted wirelessly via Wi-Fi and displayed in real-time on a mobile phone browser, enabling remote monitoring without the need for a traditional display.

This smart health device can be useful in home-based patient care, fitness tracking, rural health setups, and telemedicine applications. The project demonstrates the practical integration of embedded systems, wireless communication, and biomedical sensing in a compact, battery-operated device.

Theme

- To build a low-cost, wireless, portable blood pressure monitor
- Real-time pulse and oxygen data acquisition
- Battery powered for mobility

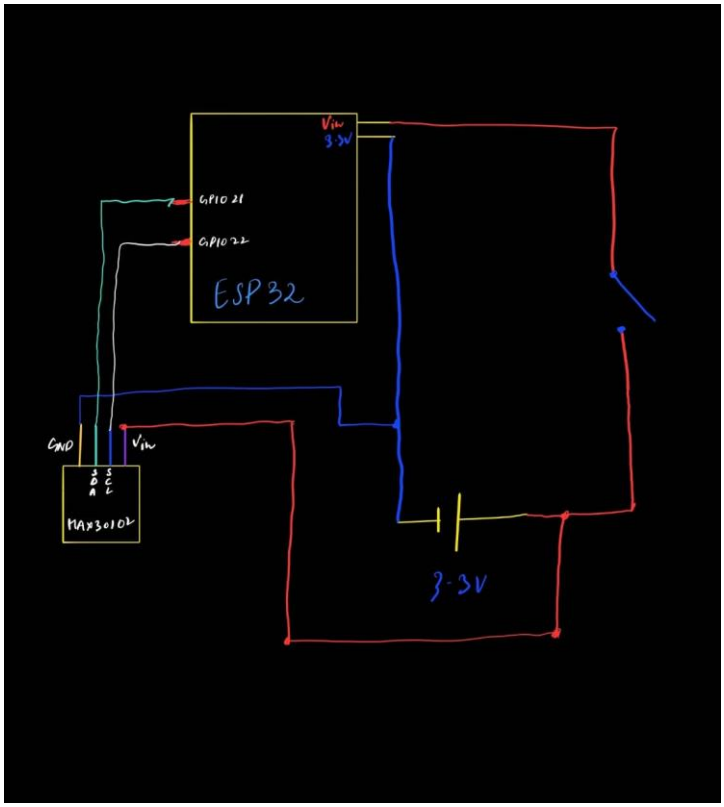
System Architecture



Component List

- ESP32
- MAX30102
- LM2596 Buck Converter
- 3x 3.7V Lithium-ion cells
- Switches
- Wires and case/enclosure

Circuit Diagram



Working Principle Overview

1. MAX30102 Sensor (Pulse & SpO_2 Monitoring):
    a. Uses Infrared (IR) and Red LEDs to measure heart rate and blood oxygen (SpO_2) levels by detecting changes in light absorption through the fingertip.
2. ESP32 Microcontroller:
    a. Reads data from the MAX30102 sensor via I^2C communication.
    b. Processes the raw signals to calculate BPM (Beats Per Minute) and SpO_2.
    c. Sends live data wirelessly via Wi-Fi (to a mobile app, server, or dashboard).
3. LM2596 Buck Converter:
    a. Converts higher battery voltage (e.g., 7.4V–12V) down to a safe 5V to power the ESP32 and sensor modules efficiently.
4. Power Switches:
    a. Allow manual control of power to various components—improving safety, testing, and battery management.

Power Management

Why We Used 3 × 3.7V Batteries

The MAX30102 sensor and ESP32 microcontroller require a regulated 5V power supply to function reliably. Since each Li-ion battery provides 3.7V, we connected three cells in series to get a total of approximately:

3.7V × 3 = 11.1V (nominal)
Max voltage when fully charged = ~12.6V

Using three cells in series offers:

- 🔋 High enough input voltage for the buck converter to regulate down to 5V
- 🔁 Better battery life and longer operation time (increased capacity and discharge time)
- ✅ Ensures stable 5V output even as battery charge drops slightly

Role of LM2596 Buck Converter in Safe Operation

The LM2596 is a step-down (buck) voltage regulator module. It plays a crucial role in:

Voltage Regulation
- Converts 11.1V input from the battery pack to a steady 5V output
- Protects sensitive components like ESP32 and MAX30102 from overvoltage damage

Heat and Power Efficiency
- More efficient than linear regulators like the 7805, especially when input voltage is high
- Minimal heat loss even under continuous operation

Output Current Handling
- Can supply up to 2A of current, which is sufficient for ESP32 and the MAX30102 combined
- Ensures no brownout or instability during Wi-Fi or Bluetooth transmission

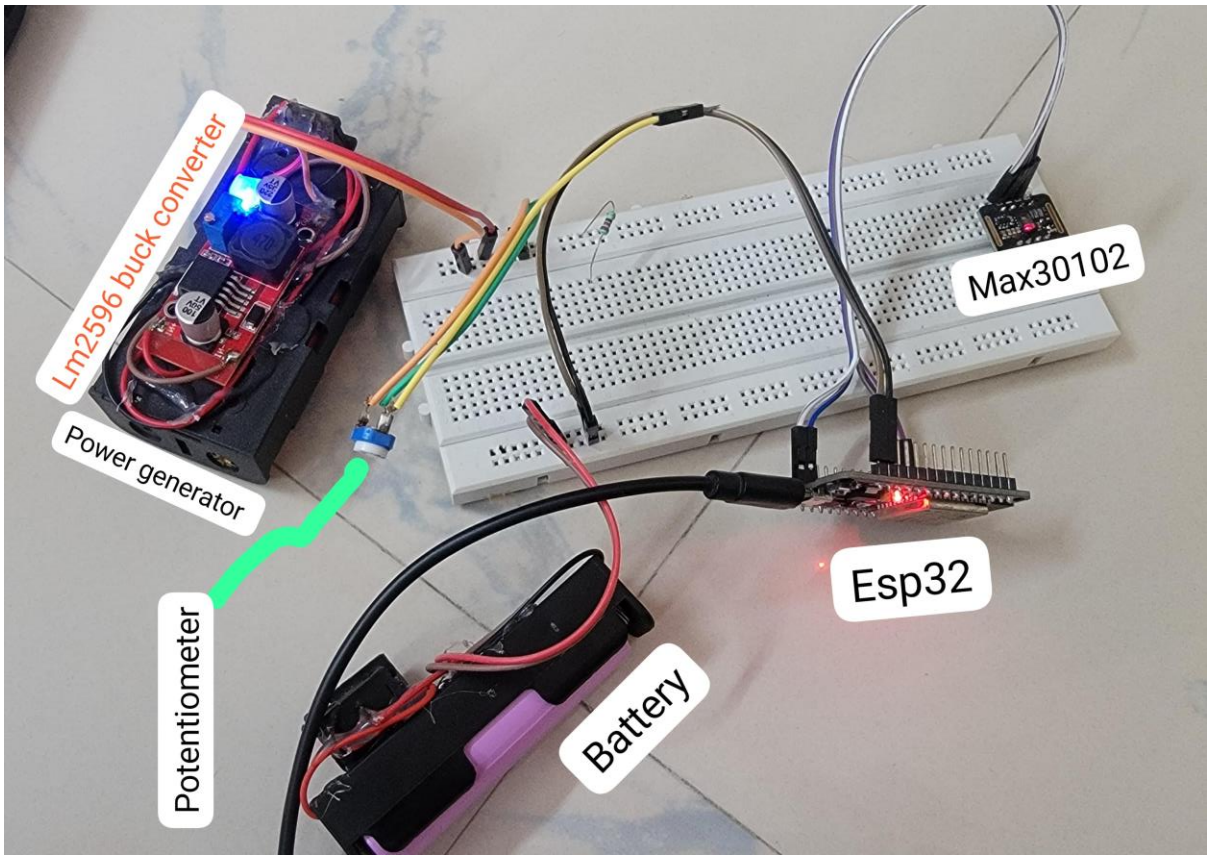
 Safe and Adjustable
- Includes a potentiometer to precisely set output voltage
- Some LM2596 modules come with overcurrent and thermal protection

Current Consumption Details

Component	Typical Current Draw	Description
ESP32 (Idle)	~70–100 mA	Can spike to 250–300 mA during Wi-Fi/Bluetooth activity
MAX30102	~1–5 mA	Very low power, especially with green LED disabled
Bluetooth TX	~100–150 mA	While transmitting data
Total	~150–350 mA avg	Within safe limits of LM2596 (up to 2A)

Battery Life Estimate
Assuming 350mA current draw from a ~2200mAh × 3 battery pack:
Battery life ≈ (2200 mAh) / (350 mA) = ~6 hours (approx.)

Hardware Setup



- Real images of project wiring, power supply, and mounted components

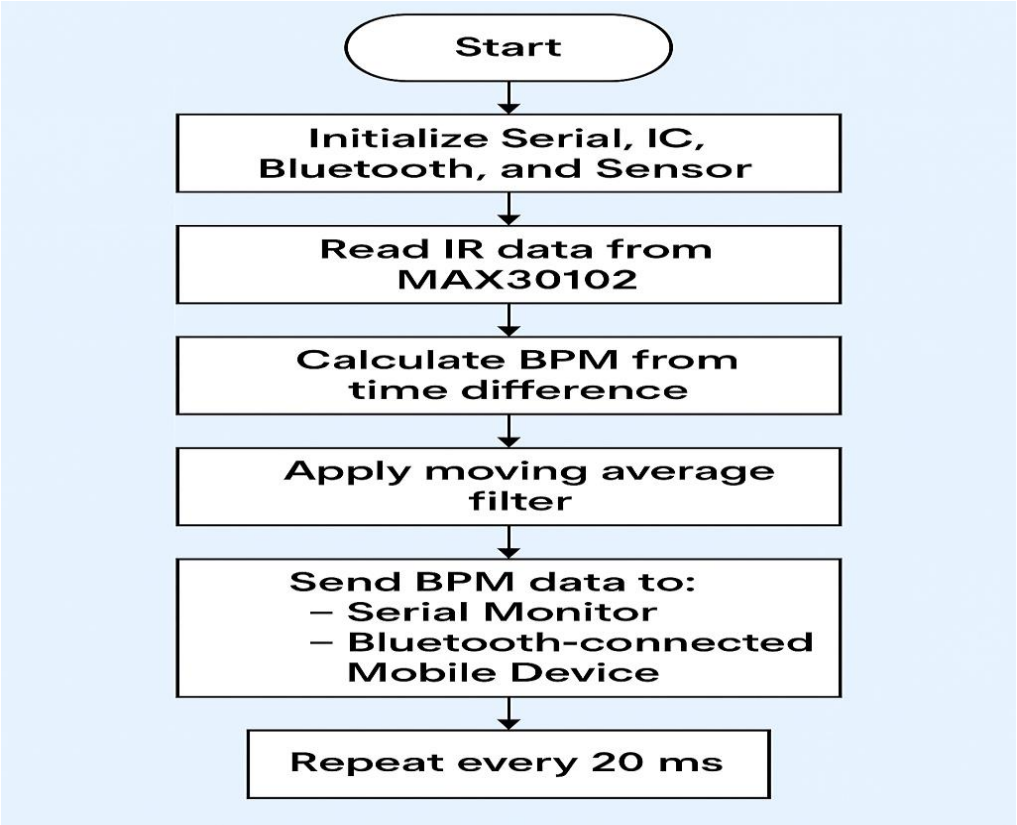
Software Overview

The software for this project is written in Arduino C++ and is deployed to the ESP32 microcontroller. It interfaces with the MAX30102 pulse sensor via I2C and uses Bluetooth Serial communication to transmit live heart rate data to a mobile device.

◇ Key Functional Modules:

1. Sensor Initialization
    a. The `MAX30105` library is used to configure the MAX30102 sensor.
    b. The sensor is initialized with default settings, and the green LED is disabled to reduce power consumption.
2. I2C Communication Setup
    a. The ESP32 communicates with the MAX30102 using I2C on pins GPIO21 (SDA) and GPIO22 (SCL).
3. Heartbeat Detection & BPM Calculation
    a. The sensor provides infrared (IR) values which are used to detect peaks (heartbeats).
    b. The time difference between two heartbeats (`delta`) is used to calculate the Beats Per Minute (BPM).
    c. A moving average filter smooths the BPM output.
4. Bluetooth Communication
    a. The ESP32 acts as a Bluetooth server named `ESP32-BPM-Logger`.
    b. Processed data (timestamp and BPM) is sent over Bluetooth Serial (`SerialBT`) to a paired mobile device.
    c. The same data is also logged to the serial monitor for debugging.
5. Data Format
    a. The output format is CSV:
    Timestamp(ms),BPM
    e.g., 54325,75

Logic Flow Diagram:



Overview of code

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

📌 Explanation:

- Wire.h: Enables I2C communication (used to talk to the MAX30102 sensor).
- MAX30105.h: Library to configure and read data from the MAX30102 sensor.
- heartRate.h: Contains functions like checkForBeat() to detect heartbeats from IR data.
- BluetoothSerial.h: Enables ESP32 to communicate via Bluetooth (Classic, not BLE).

```
MAX30105 particleSensor;
BluetoothSerial SerialBT;
```

```
long lastBeat = 0;
float beatsPerMinute = 0;
int beatAvg = 0;
```

- particleSensor: Object to access sensor functions.
- SerialBT: Bluetooth Serial object for sending data to a phone.
- lastBeat: Timestamp (ms) of the last detected beat.
- beatsPerMinute: Instantaneous heart rate value.
- beatAvg: Moving average of BPM to smooth output.

```
void setup() {
  Serial.begin(115200);        // For debug if needed
  SerialBT.begin("ESP32-BPM-Logger");  // Appears on phone
```

- Serial.begin(115200): Opens the USB serial port for debugging.
- SerialBT.begin(...): Starts Bluetooth with a device name ESP32-BPM-Logger. This name appears in the phone's Bluetooth pairing list.

```
  Wire.begin(21, 22);  // SDA = D21, SCL = D22
```

- Initializes I2C communication on ESP32 using GPIO21 as SDA and GPIO22 as SCL.

```
if (!particleSensor.begin(Wire, I2C_SPEED_STANDARD)) { Serial.println("MAX30102 not found. Check wiring."); while (1); }
```

- Checks if the sensor is connected and working.
- If it fails, it prints an error and stops the program using while(1);.

```
  particleSensor.setup(); // Default settings
  particleSensor.setPulseAmplitudeRed(0x0A);   // Weak red
```

```
  particleSensor.setPulseAmplitudeGreen(0);    // Disable green
```

- Initializes the sensor with default LED pulse settings.
- Weak red LED (intensity = 0x0A) is used to detect pulse.
- Green LED is turned off to save power.

```
  SerialBT.println("Timestamp(ms),BPM");
  Serial.println("Timestamp(ms),BPM");
}
```

- Sends a header line (CSV format) to both the serial monitor and Bluetooth.
- Useful for logging in Excel or terminal apps.

```
void loop() {
  long irValue = particleSensor.getIR();
```

- Reads IR light intensity from the sensor. It reflects blood flow under the skin, which is used to detect heartbeats.

```
  if (checkForBeat(irValue)) {
    long now = millis();
    long delta = now - lastBeat;
    lastBeat = now;

    beatsPerMinute = 60000.0 / delta;
    beatAvg = (beatAvg * 3 + beatsPerMinute) / 4;
```

- checkForBeat(irValue): Returns true when a pulse peak is detected.
- delta = now - lastBeat: Time between current and last beat.
- 60000 / delta: Converts time interval to BPM.
- beatAvg: Smoothes out BPM with a simple moving average (weight of 75% past value, 25% current).

```
    String line = String(now) + "," + String(beatAvg);
    SerialBT.println(line);
    Serial.println(line);
  }

  delay(20);
}
```

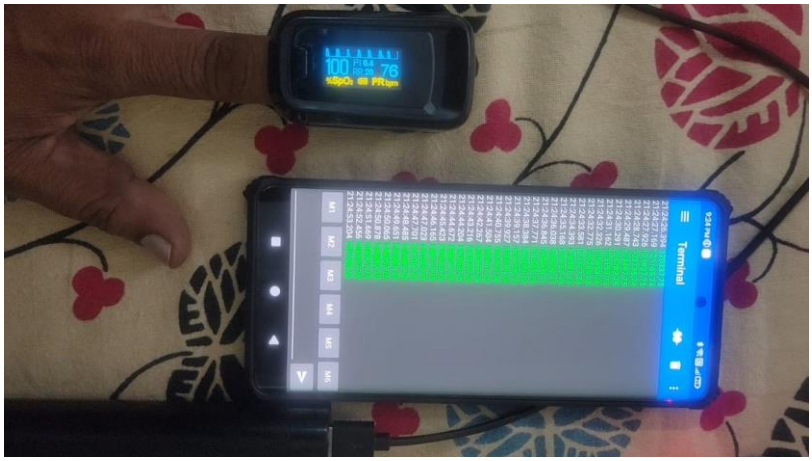
- Creates a string of timestamp,BPM format.
- Sends this data to both the phone (Bluetooth) and the serial monitor.
- Small delay(20) to prevent flooding the processor and allow stable signal processing.

Key Features Summary

Feature	Function
💓 Heartbeat Detection	Via MAX30102 IR values using checkForBeat()
🔄 BPM Calculation	Based on time between beats (60000 / delta)
📶 Bluetooth Output	Real-time data to mobile using SerialBT
⚡ Power Optimization	Green LED off, low pulse amplitude
📄 CSV Format	Easy logging with timestamp and BPM
📱 Mobile Integration	Can be viewed with any serial Bluetooth terminal app

Result

The user can connect a mobile phone to the ESP32 over Bluetooth and receive real-time heart rate data, which can be displayed in a terminal app or logged externally.



We compared the data with real time bp machine and we got accurate results for every second when we compared but the first 10 seconds the data is coming not accurately as it starting values from 0 and it is taking 10 to 15 seconds to give accurate data

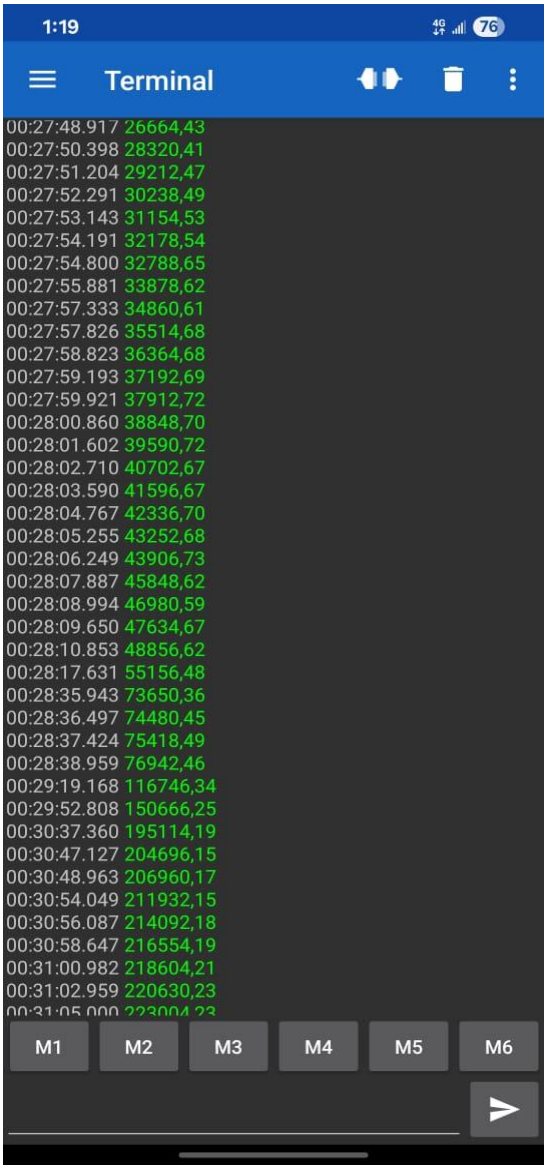
Applications

- Home health monitoring
- Elderly care
- Remote clinics / rural healthcare
- Fitness and wellness tracking

Advantages

- Portable and wireless
- Low-cost and battery operated
- Real-time monitoring capability
- Easy to build and use

Mobile Interface



- Screenshots or demo of live heart rate on mobile app

Future Improvements

- Integration with mobile app or cloud
- Adding OLED/LCD display
- Using AI/ML for better BP estimation from PPG

Github link

https://github.com/sameernabeen/Esp32-live-heart-rate

Conclusion

The project titled “Smart Wireless Blood Pressure and Heart Rate Monitoring System using ESP32 and MAX30102” successfully demonstrates the integration of embedded systems, biomedical sensing, and wireless communication for real-time health monitoring.

Using the MAX30102 pulse sensor, the system is able to accurately detect heartbeats and calculate the user’s heart rate. The ESP32 microcontroller efficiently processes this data and transmits it wirelessly to a mobile phone using Bluetooth communication, allowing live visualization without the need for external displays. The device is powered by three 3.7V Li-ion batteries regulated through an LM2596 buck converter, ensuring stable 5V operation for both the ESP32 and the sensor.

Additional power control through dual switches enhances safety and energy efficiency. The heart rate data is displayed in a clean CSV format (Timestamp, BPM) and can be monitored through any Bluetooth terminal app on mobile devices.

This portable, low-cost, and battery-operated solution holds strong potential for applications in home healthcare, remote patient monitoring, fitness tracking, and rural medical setups.