A

Project Report

On

Heart-Rate Monitoring System

Submitted in partial fulfillment of the requirement for the VI semester

Bachelor of Technology

In

Electronics and Communication Engineering

By

Mansi Joshi 2118766

Under the Guidance of

Mr. Abhijit Bhakuni

Assistant Professor

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI,

DISTRICT- NAINITAL-263132

2023-2024

STUDENT’S DECLARATION

I, **Mansi Joshi** hereby declare the work, which is being presented in the project, entitled ‘ **Heart-Rate Monitoring System**’ in partial fulfillment of the requirement for the **VI semester** of the degree **Bachelor of Technology (B.Tech.)** in the session **2023-2024**, is an authentic record of my work carried out under the supervision of **Mr. Abhijit Bhakuni.**
The matter embodied in this project has not been submitted by me for the award of any other degree.

Date: Mansi Joshi

CERTIFICATE

The project report entitled "**Heart Rate Monitoring System**" being submitted by **Mansi Joshi** D/o **Bhuwan Chandra Joshi**, **2118766** of B.Tech.(ECE) to Graphic Era Hill University Bhimtal Campus for the award of bonafide work carried out by her. She have worked under my guidance and supervision and fulfilled the requirement for the submission of a report.

Mr. Abhijit Bhakuni
(Project Guide)

Dr. Sandeep Sunori
(HOD, ECE)

ACKNOWLEDGEMENT

We take immense pleasure in thanking the Honorable Director **'Prof. (Col.) Anil Nair (Retd.)',** GEHU Bhimtal Campus to permit me and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance, and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me with everything that I need. I again want to extend thanks to our president **'Prof. (Dr.) Kamal Ghanshala'** for providing us with all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to **'Dr. Sandeep Sunori'** (Head, Electronics and Communication Engineering, GEHU Bhimtal Campus), our project guide **'Mr. Abhijit Bhakuni'** (Assistant Professor, Department of Electronics and Communication Engineering, GEHU Bhimtal Campus) and other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this thesis.

Finally, yet importantly, I would like to express my heartiest thanks to my beloved parents, for their moral support, affection, and blessings. I would also like to pay my sincere thanks to all my friends and well-wishers for their help and wishes for the successful completion of this research.

Mansi Joshi 2118766

ABSTRACT

The project explores the significance of heart rate monitoring as a critical component of maintaining overall health and fitness. It underscores the importance of tracking heart rate to monitor fitness progress, detect health concerns, and manage stress levels.

1. Fitness Progress Tracking: Monitoring heart rate during exercise helps individuals gauge workout intensity and adjust routines to achieve fitness goals efficiently.

2. Detection of Health Concerns: Regular heart rate monitoring can aid in early detection of cardiac issues like arrhythmias, allowing timely medical intervention.

3. Stress Management: Observing heart rate changes in response to stress can help individuals employ relaxation techniques to manage stress effectively.

The project delves into various heart rate measurement systems, including ECG, PPG, and SCG, discussing their principles, methodologies, accuracy, and limitations. It highlights future advancements aimed at enhancing accuracy, continuous monitoring, and remote monitoring capabilities.

A comprehensive overview of the Blynk IoT platform is provided, emphasizing its user-friendly interface, versatile hardware compatibility, data visualization, and security features. The integration of the MAX30102 sensor and ESP8266 microcontroller is detailed, outlining their functionalities, operating principles, and applications in health monitoring.

Lastly, the project presents a system architecture for a heart rate monitoring system using Blynk IoT, detailing hardware components, software components, communication flow, benefits, and limitations. It includes a step-by-step guide for setting up Blynk and configuring the app for heart rate measurement with an ESP8266, ensuring users can monitor their heart rate data remotely.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Monitoring heart rate is a crucial aspect of maintaining overall health and fitness. The heart rate, defined as the number of times the heart beats per minute (bpm), serves as a valuable indicator of various physiological factors. This report delves into the significance of monitoring heart rate and highlights its relevance in tracking fitness progress, identifying health concerns, and managing stress levels.

1.1 Fitness Progress Tracking

One of the primary reasons to monitor heart rate is to track fitness progress during exercise. By observing changes in heart rate during physical activity, individuals can gauge the intensity of their workouts and adjust accordingly. For instance, a higher heart rate during exercise indicates increased effort and intensity, while a lower heart rate may suggest improved cardiovascular fitness over time. By monitoring heart rate during workouts, individuals can optimise their training routines to achieve desired fitness goals efficiently.

1.2 Detection of Health Concerns

Monitoring heart rate can also aid in the early detection of potential health issues, such as arrhythmias (irregular heartbeat). An irregular heart rate, either too fast (tachycardia) or too slow (bradycardia), may signify underlying cardiac conditions that require medical attention. Regular monitoring of heart rate patterns can help individuals identify abnormalities and seek timely medical intervention, thus preventing the progression of serious cardiac ailments.

1.3 Stress Management

Furthermore, monitoring heart rate can serve as an effective tool for managing stress levels. Stress exerts a significant influence on heart rate, with heightened stress often leading to an increase in heart rate. By monitoring changes in heart rate in response to stressors, individuals can gain insight into their stress levels and employ appropriate relaxation techniques to mitigate stress-induced physiological responses. Practising mindfulness, deep breathing exercises, and other stress-reducing techniques can help regulate heart rate and promote overall well-being.

CHAPTER 2

LITERATURE REVIEW

2.1 HEART RATE MEASUREMENT ALGORITHM

Measuring heart rate is a crucial aspect of health monitoring. Here's a detailed breakdown of the methodology behind various heart rate measurement systems:

2.1.1. Electrical Activity

- Principle

ECG measures the electrical activity of the heart muscle as it contracts and relaxes. Electrodes placed on specific locations on the chest, limbs, or back detect these tiny electrical signals.

- Methodology

Electrodes pick up the electrical impulses generated by the heart.

An ECG machine amplifies these weak signals and displays them as a waveform (ECG trace) on a screen or prints it on paper.

Trained professionals can interpret the ECG trace to identify heart rhythm abnormalities, heart attacks, and other heart-related conditions.

- Accuracy

Highly accurate and considered the gold standard for heart rhythm assessment.

- Limitations

Requires medical-grade equipment and trained personnel for interpretation. Not suitable for continuous monitoring due to the number of electrodes needed.

2.1.2. Peripheral Blood Flow Variations

- Principle

PPG measures the tiny changes in blood volume that occur with each heartbeat in your arteries. A light source (LED) and a light detector (photodiode) are used to detect these variations.

- Methodology

A sensor clips onto your finger, earlobe, or forehead.

The LED emits light that penetrates your skin and tissues.

As blood volume in the underlying artery changes with each heartbeat, the amount of light absorbed or reflected by the tissues changes.

The photodiode detects these fluctuations in light intensity, which are then processed to estimate heart rate.

- Accuracy

Reasonably accurate for resting heart rate but can be affected by movement or sensor placement.

- Limitations

Less precise than ECG, especially during exercise. Sensor placement and external factors can influence readings.

2.1.3. Acoustic Techniques

- Principle

SCG detects the subtle sounds produced by the heart's contractions and blood flow. A microphone placed on the chest picks up these sounds.

- Methodology

A microphone is placed on your chest near your heart.

The microphone captures the sounds generated by your heartbeat.

These sounds are then filtered and analyzed to extract heart rate information.

- Accuracy

Reasonably accurate for resting heart rate but can be affected by breathing, movement, and background noise.

- Limitations

Sensitive to external noise and may not be suitable for use in noisy environments.

Choosing the Right Method

- The most suitable heart rate measurement system depends on the application and desired level of accuracy.
- Medical diagnosis: ECG is the gold standard for diagnosing heart conditions.
- Fitness monitoring: PPG-based wearables like smartwatches offer convenient heart rate tracking during exercise.
- General health monitoring: PPG-based sensors or combination approaches might be suitable for home use or basic health monitoring.

Future Advancements

The field of heart rate measurement is constantly evolving. Researchers are exploring new techniques and improving existing ones for:

- Enhanced Accuracy: Minimizing the impact of noise and movement on readings.
- Continuous Monitoring: Developing comfortable and reliable wearables for long-term heart rate tracking.
- Remote Monitoring: Enabling healthcare professionals to monitor patients' heart health remotely.

2.2 DATA PROCESSING TECHNIQUES IN HEARTBEAT MEASUREMENT SYSTEMS

2.2.1 Filtering

Noise Reduction: Raw sensor data can be contaminated with noise from various sources like electrical interference, movement artifacts, and sensor imperfections. Filtering techniques such as averaging filters, median filters, and bandpass filters can help remove unwanted noise and isolate the signal related to the heartbeat.

Bandpass Filtering: The human heart rate typically falls within a specific range (around 60 to 100 beats per minute). Bandpass filters allow only frequencies within this range to pass through, further eliminating high-frequency noise and low-frequency baseline drift.

2.2.2 Signal Amplification

The electrical signals from the heart, especially in PPG (photoplethysmography) systems, are very weak. Amplification techniques increase the signal strength to improve the signal-to-noise ratio and facilitate further processing.

2.2.3 Peak Detection

Identifying heartbeats in the processed signal is crucial for calculating heart rate. Peak detection algorithms locate the local maxima in the signal, which correspond to the peaks of the pulsatile waveform caused by each heartbeat.

2.2.4 Artifact Removal

Movement artifacts caused by user activity can significantly affect the PPG signal. Techniques like peak amplitude thresholding, pattern recognition, and machine learning algorithms can help identify and remove these artifacts to improve data quality.

2.2.5 Baseline Correction

The baseline of the PPG signal can drift over time due to factors like breathing or sensor placement. Baseline correction techniques aim to remove this drift and ensure accurate peak detection.

2.2.6 Heart Rate Calculation

Once the R-peaks (representing heartbeats) are identified in the processed signal, the heart rate can be calculated. This typically involves measuring the time interval between consecutive R-peaks and converting it to beats per minute (bpm).

By applying these data processing techniques, heartbeat measurement systems can extract reliable heart rate information from raw sensor data, enabling a wide range of applications in health monitoring and fitness tracking.

CHAPTER 3

OVERVIEW OF BLYNK IOT PLATFORM

Blynk is an intuitive and versatile IoT platform that empowers users to easily build and deploy connected projects without extensive coding knowledge. At its core, Blynk provides a comprehensive suite of tools and services designed to streamline the development and management of IoT applications.

3.1User-Friendly Interface

Blynk offers a user-friendly interface accessible through both web and mobile applications, making it convenient for users to monitor and control their IoT devices from anywhere with internet access.

3.2 Drag-and-Drop Interface

One of Blynk's standout features is its drag-and-drop interface, which allows users to quickly design custom dashboards and user interfaces for their IoT projects without writing any code.

3.3 Versatile Hardware Compatibility

Blynk supports a wide range of popular hardware platforms, including Arduino, Raspberry Pi, ESP8266, ESP32, and more, enabling users to leverage their preferred hardware for IoT development.

3.4 Comprehensive Widget Library

Blynk provides a rich library of pre-built widgets, such as buttons, sliders, graphs, and gauges, that users can easily incorporate into their projects to create interactive and visually appealing user interfaces.

3.5 Cloud Connectivity

Blynk's cloud infrastructure facilitates seamless communication between IoT devices and the Blynk app, allowing users to remotely monitor and control their devices in real-time.

3.6 Data Visualization and Logging

Blynk enables users to visualise sensor data in real-time through dynamic graphs and charts, as well as log historical data for later analysis and insights.

3.7 Event and Notification Management

Blynk supports event-driven programming, allowing users to define triggers and automate actions based on predefined conditions. Additionally, users can receive notifications via email, SMS, or push notifications directly to their mobile devices.

3.8 Secure Communication

Security is a top priority for Blynk, with built-in features such as secure communication protocols (SSL/TLS), token-based authentication, and data encryption to ensure the integrity and confidentiality of IoT data.

3.9 Scalability and Flexibility

Whether you're building a simple home automation project or a complex industrial monitoring system, Blynk offers scalability and flexibility to accommodate projects of any size and complexity.

3.10 Community and Support

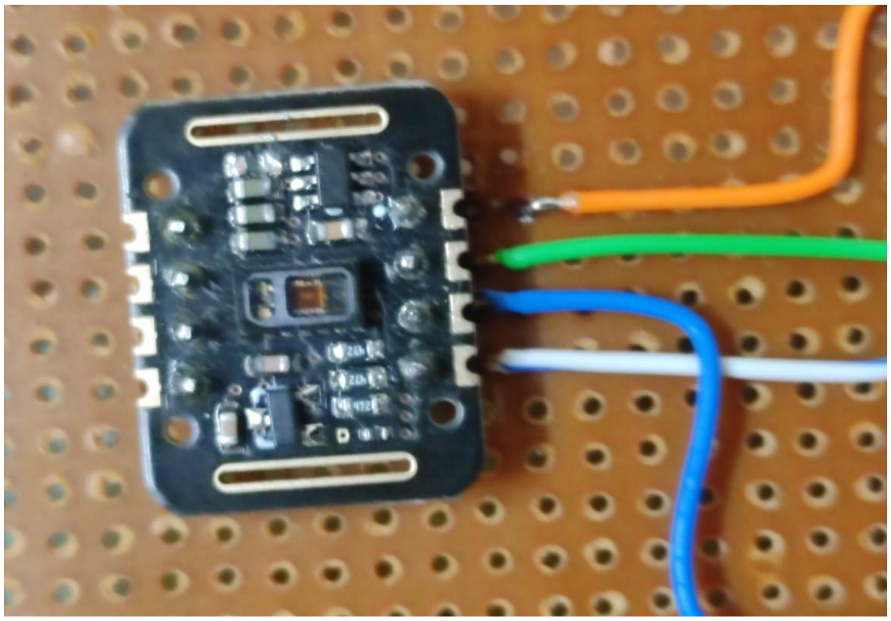
Blynk boasts a vibrant and active community of developers and enthusiasts who contribute to the platform's ecosystem by sharing projects, tutorials, and troubleshooting tips. Additionally, Blynk provides comprehensive documentation, tutorials, and customer support to assist users throughout their IoT journey.

CHAPTER 4

HEART RATE MONITORING SYSTEM

Creating a heart rate monitoring system involves several key components to accurately measure and monitor heart rate data. Here's a detailed overview of the components required:

4.1Max30102 Sensor



4.1.1 Functionality

The MAX30102 sensor combines two key functionalities:

- Photoplethysmography (PPG): Measures the changes in blood volume by shining light into the skin and detecting the amount of light absorbed or reflected by the blood vessels.
- Pulse oximetry: Calculates the oxygen saturation level in the blood by analyzing the absorption of light at different wavelengths.
- Integrating both PPG and pulse oximetry capabilities into a single module, the MAX30102 sensor provides a comprehensive solution for monitoring heart rate and SpO2 levels.

4.1.2 Operating Principle

- The MAX30102 sensor utilises red and infrared (IR) LEDs to illuminate the skin and a photodetector to measure the light intensity transmitted or reflected by the blood vessels.
- Changes in blood volume during each heartbeat result in fluctuations in the detected light intensity, which are then processed to extract heart rate and SpO2 information.

4.1.3 Features

- Integrated red and IR LEDs: The MAX30102 sensor features built-in red and IR LEDs, eliminating the need for external light sources.
- High sensitivity: The sensor's high sensitivity enables accurate detection of small changes in blood volume, ensuring precise heart rate and SpO2 measurements.
- Low power consumption: Optimized power management features allow the sensor to operate efficiently, making it suitable for battery-powered applications.
- Digital interface: The sensor communicates with the microcontroller or host device via an I2C interface, simplifying integration into existing systems.
- Ambient light rejection: Advanced algorithms and signal processing techniques minimize the effects of ambient light interference, enhancing measurement accuracy.
- Small form factor: The compact size of the MAX30102 sensor makes it suitable for wearable devices and portable healthcare applications.

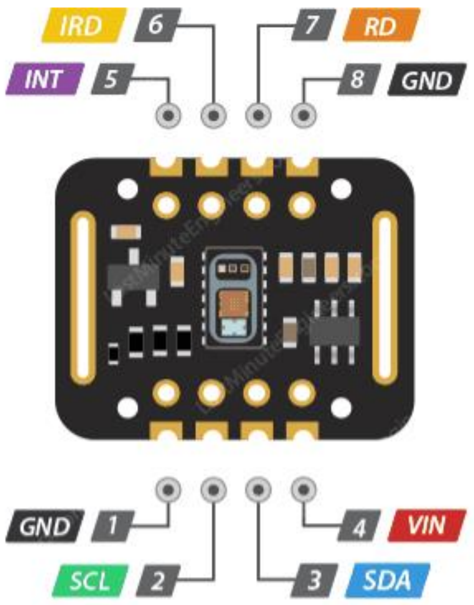
4.1.4 Applications

- Health monitoring devices: The MAX30102 sensor is commonly used in wearable fitness trackers, smartwatches, and health monitoring devices to measure heart rate and SpO2 levels.
- Medical instruments: The sensor finds applications in medical instruments such as pulse oximeters, patient monitors, and medical diagnostic devices for non-invasive monitoring of vital signs.
- Sports and fitness: Athletes and fitness enthusiasts use devices equipped with the MAX30102 sensor to track their heart rate during exercise and optimize training routines.
- Wellness and lifestyle: The sensor enables individuals to monitor their heart rate and SpO2 levels in real-time, promoting overall health and well-being.

4.1.5 Integration

The MAX30102 sensor is typically interfaced with a microcontroller or development board, such as Arduino or Raspberry Pi, using the I2C communication protocol.
Maxim Integrated provides a comprehensive software library and example code to facilitate the integration of the MAX30102 sensor into various applications.
Additionally, breakout boards and modules featuring the MAX30102 sensor are available from third-party manufacturers, simplifying the prototyping and development process.

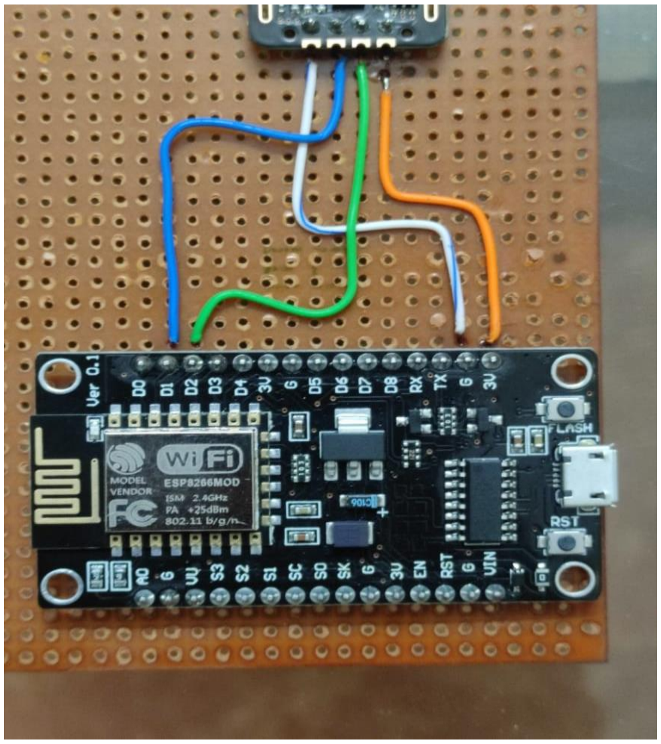
4.1.6 Pin Diagram



Overall, the MAX30102 sensor offers a reliable and convenient solution for measuring heart rate and SpO2 levels in a wide range of applications, from wearable fitness devices to medical instruments, enabling non-invasive monitoring of vital signs with high accuracy and efficiency.

4.2 <u>ESP8266</u>

The ESP8266, from Espressif Systems, is a popular and versatile Wi-Fi Microcontroller Unit (MCU) that has revolutionized the Internet of Things (IoT) landscape. Its affordability, ease of use, and feature set make it a compelling choice for a wide range of applications. Let's delve deeper into its technical specifications, functionalities, and explore the reasons behind its success.


Ver 0.1
D0 D1 D2 D3 D4 3V G D5 D6 D7 D8 RX TX G 3V
FLASH
RST
ESP8266MOD
A0 G VU S3 S2 S1 SC SO SK G 3V EN RST G VIN

4.2.1 Memory

- On-board RAM: Typically 32 KB (limited and a crucial factor for project complexity).
- Flash Memory: Usually 80 KB for code storage (expandable with external memory options).

4.2.2 General purpose input / output interface (gpio)

- ESP8266EX has 17 GPIO pins which can be assigned to various functions by programming the appropriate registers.
- Each GPIO can be configured with internal pull-up or pull-down, or set to high impedance,and when configured as an input, the data are stored in software registers; the input can also be set to edge-trigger or level trigger CPU interrupts. In short, the IO pads are bi-directional, non-inverting and tristate, which includes input and output buffer with tristate control inputs.
- These pins can be multiplexed with other functions such as I2C, I2S, UART, PWM, IR Remote Control, etc.

4.2.3 Serial Peripheral Interface (SPI/HSPI)

- ESP8266EX has 3 SPIs.

- One general Slave/Master SPI
- One Slave SDIO/SPI
- One general Slave/Master HSPI

4.2.4 Networking

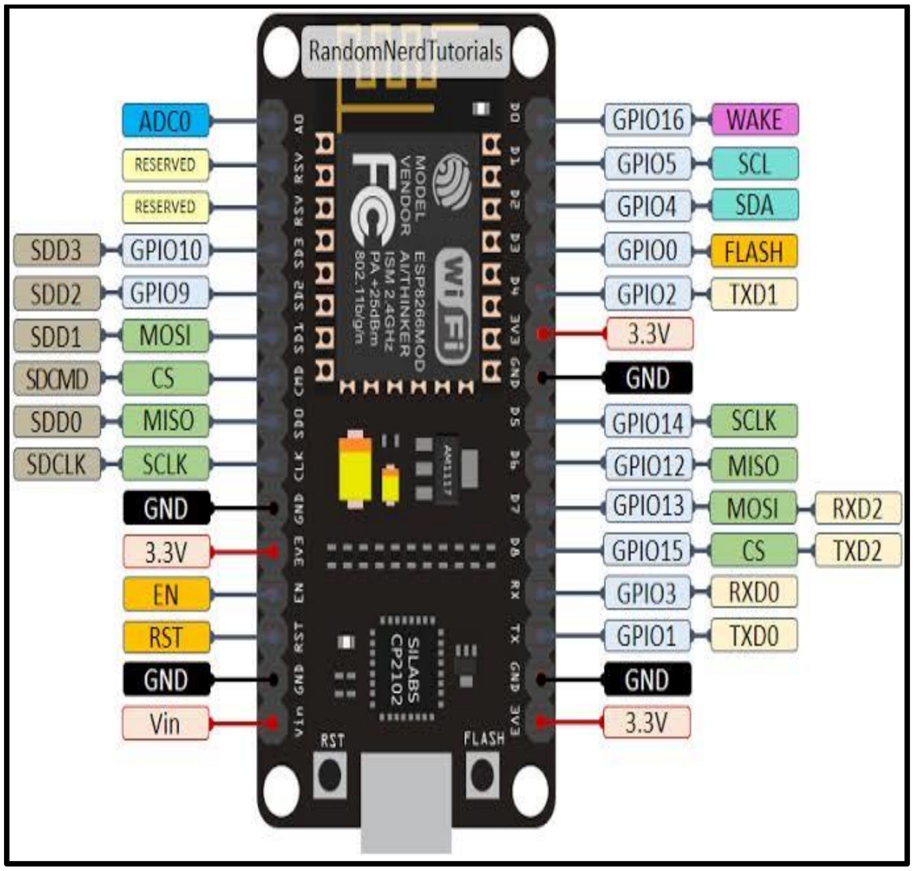
- Integrated Wi-Fi transceiver supporting 802.11 b/g/n standards for Wi-Fi connectivity.

4.2.5 Power Consumption

- Designed for low-power operation with deep sleep modes that significantly reduce power draw during idle periods, making it suitable for battery-powered applications.

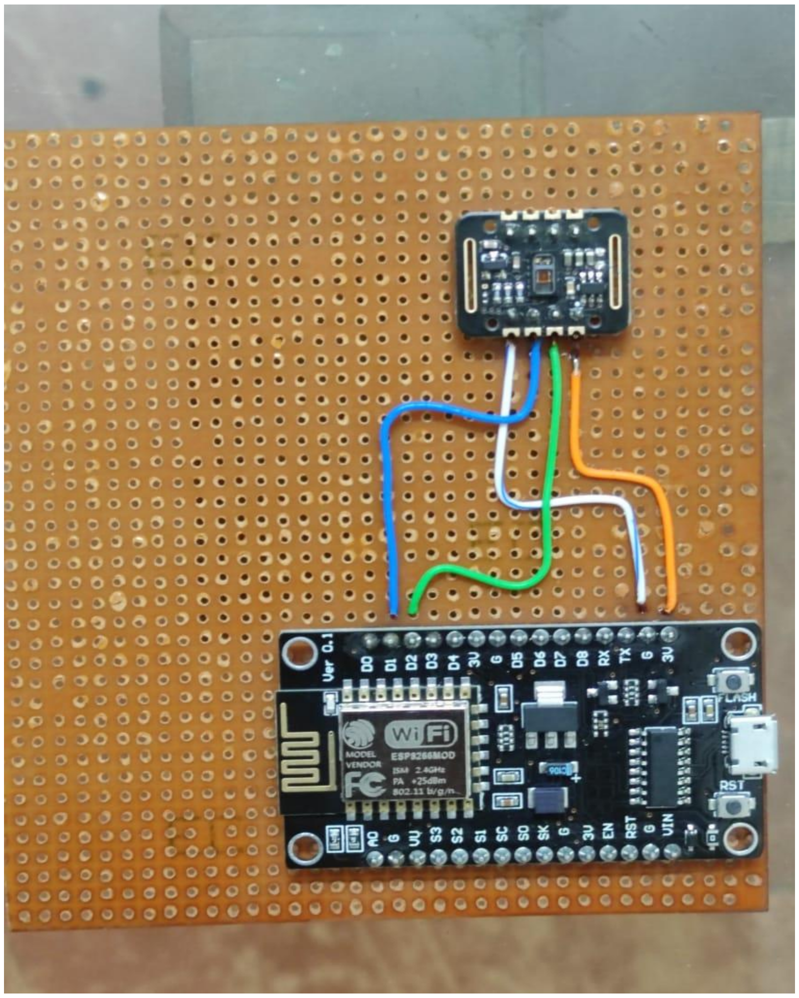
4.2.6 Development Environment

- Supports various open-source Integrated Development Environments (IDEs) like Arduino IDE and ESP-IDF, making programming accessible to beginners and experienced developers alike. Popular frameworks like Arduino core for ESP8266 simplify development further.



CHAPTER 5
SYSTEM ARCHITECTURE

System Architecture



Extracting accurate heart rate information from raw sensor data in a heartbeat measurement system measures your heart rate using a sensor and displays it on your smartphone through the Blynk IoT platform. Here's a breakdown of the architecture:

5.1 Hardware Components

5.1.1 Microcontroller Board

● The brain of the system, typically an Arduino Uno in this case.exclamation It reads sensor data, processes it, and communicates with Blynk.

5.1.2 Heart Rate Sensor

- Detects changes in blood flow using a technique called photoplethysmography (PPG). We'll clip this sensor onto your finger.

5.1.3 Jumper Wires

- Connect the microcontroller board to the heart rate sensor, following the circuit diagram

5.2 Software Components

5.2.1 Arduino IDE

The development environment where you write code for the Arduino board.

5.2.2 Blynk App

The mobile application for creating the user interface and interacting with the system.

5.2.3 Heart Rate Sensor Library

Depending on the specific sensor used, a library might be needed to interpret its raw data.

5.3 System Communication Flow

- The heart rate sensor outputs an analog signal based on the detected blood flow variations.
- The Arduino reads the analog signal from the sensor through a specific analog pin (e.g., A0).expand_more
- The code written in Arduino IDE processes the raw sensor data. This might involve filtering noise and converting the signal into a meaningful heart rate value (beats per minute).

- The Arduino code utilizes the Blynk library to send the heart rate data to a specific virtual pin on the Blynk cloud server. Virtual pins act as channels for data exchange between the hardware and the app.
- The Blynk app on your smartphone connects to the Blynk cloud server using your authentication token.
- The app receives the heart rate data from the designated virtual pin on the server.
- The app displays the heart rate data on a user-friendly interface, typically a "Value Display" widget.

5.4Benefits of this Architecture

5.4.1 Modular Design

- Each component has a specific function, making the system easier to understand, maintain, and modify.

5.4.2 Simplicity

- Uses readily available components and open-source software libraries, making it accessible for beginners.

5.4.3 Scalability

- The system can be extended to incorporate additional sensors or functionalities in the future.

5.4.4 Remote Monitoring

You can view your heart rate data from anywhere with an internet connection through the Blynk app.

5.5Limitations

5.5.1 Accuracy

- The system provides an estimate of your heart rate and may not be as precise as medical-grade equipment. Factors like sensor placement and movement can affect accuracy.

5.5.2 Security

- Ensure secure communication between the hardware and the Blynk cloud server, especially if handling sensitive health data.

5.5.3 Processing Power

- The chosen microcontroller's processing power might limit the complexity of calculations or data processing within the code.

This system architecture offers a simple and effective way to build a basic heart rate monitoring system using Blynk IoT. By understanding the components, communication flow, and limitations, you can build upon this foundation to create more advanced health monitoring applications.

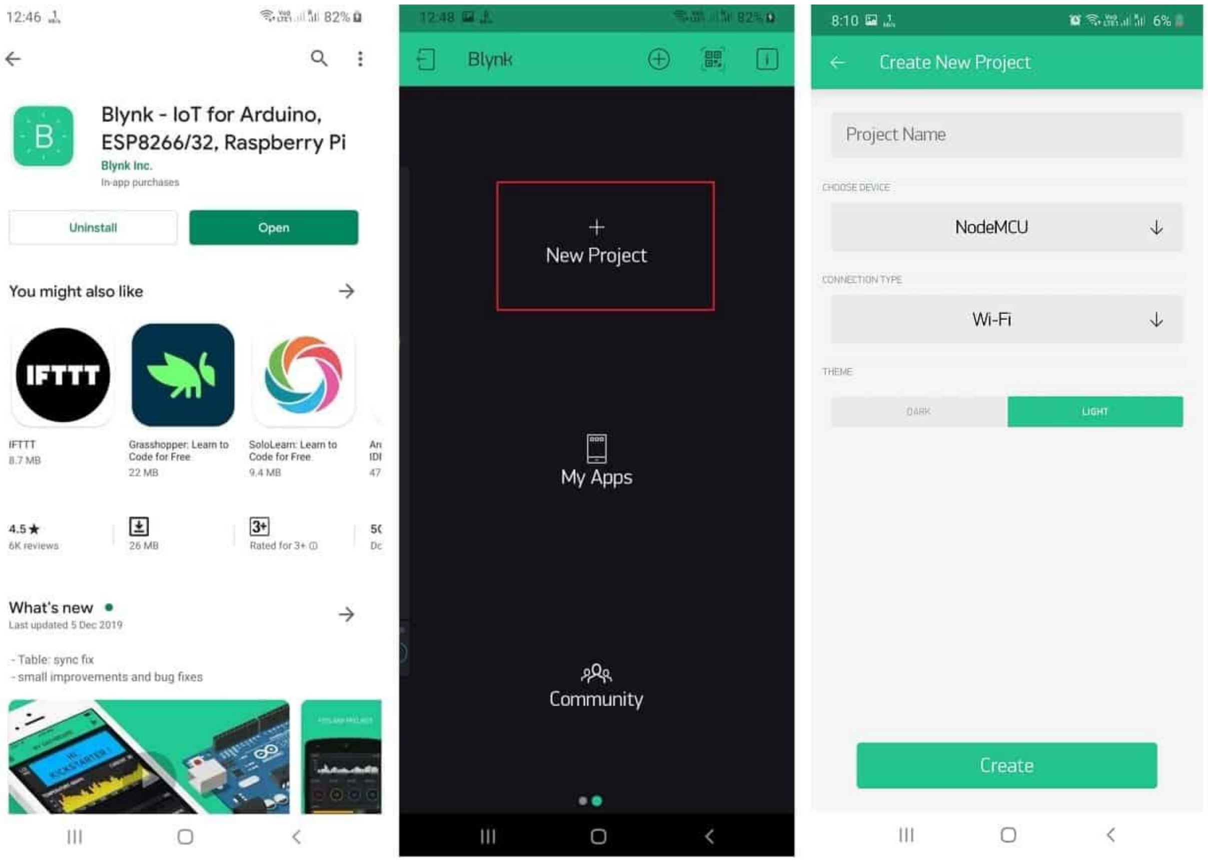
CHAPTER 6

SETTING UP BLYNK & UBIDOTS

SETTING UP BLYNK AND UBIDOTS FOR HEART RATE MONITORING

6.1 Creating Blynk Account

- Download the Blynk App
  - Search for "Blynk" in the App Store (iOS) or Google Play Store (Android) and install the application.
- Open the App
  - Launch the Blynk app on your smartphone or tablet.
- Sign Up Option
  - You'll be presented with two primary options: "Log In" and "Create New Account". Select "Create New Account" to proceed.



CHOOSE DEVICE: NodeMCU

CONNECTION TYPE: Wi-Fi

THEME: DARK [✓ LIGHT]

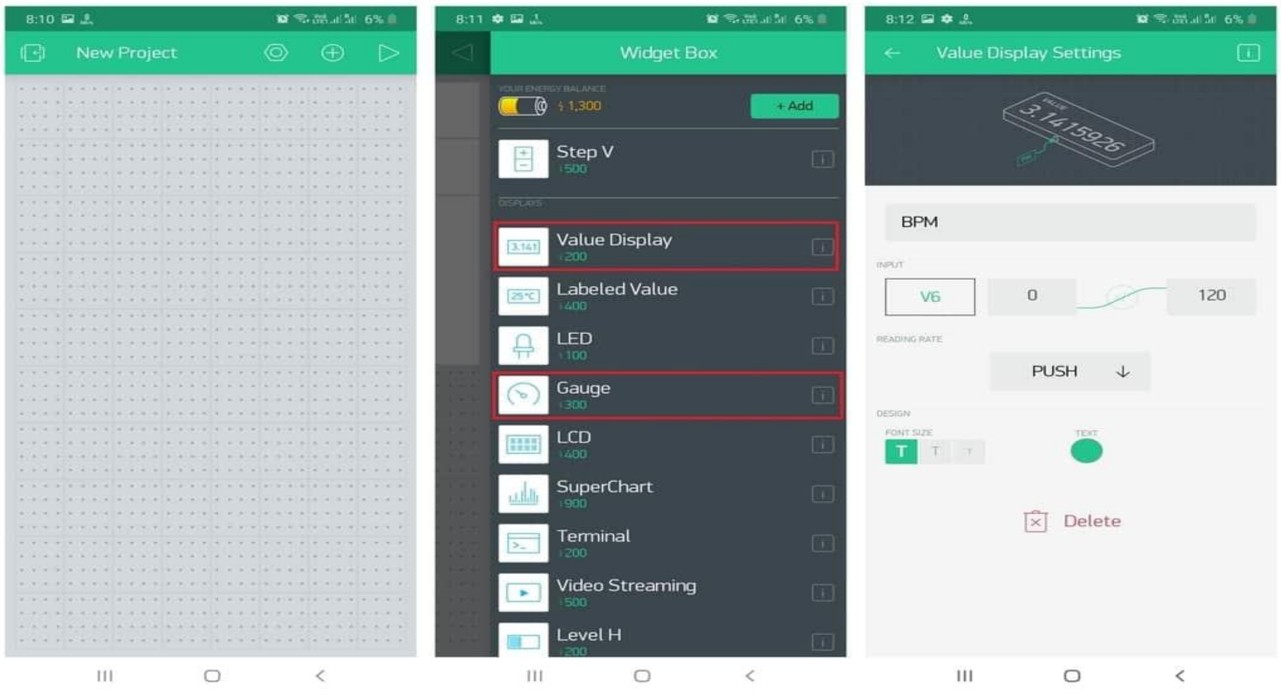
- Email and Password
    - Enter a valid email address that you'll use as your username for Blynk. Create a strong password following recommended password security practices.
- Sign Up
    - Once you've entered your email and password, tap the "Sign Up" button.

6.2 Configuring Blynk App

Here's a detailed guide on configuring the Blynk app for heart rate measurement using an ESP8266:

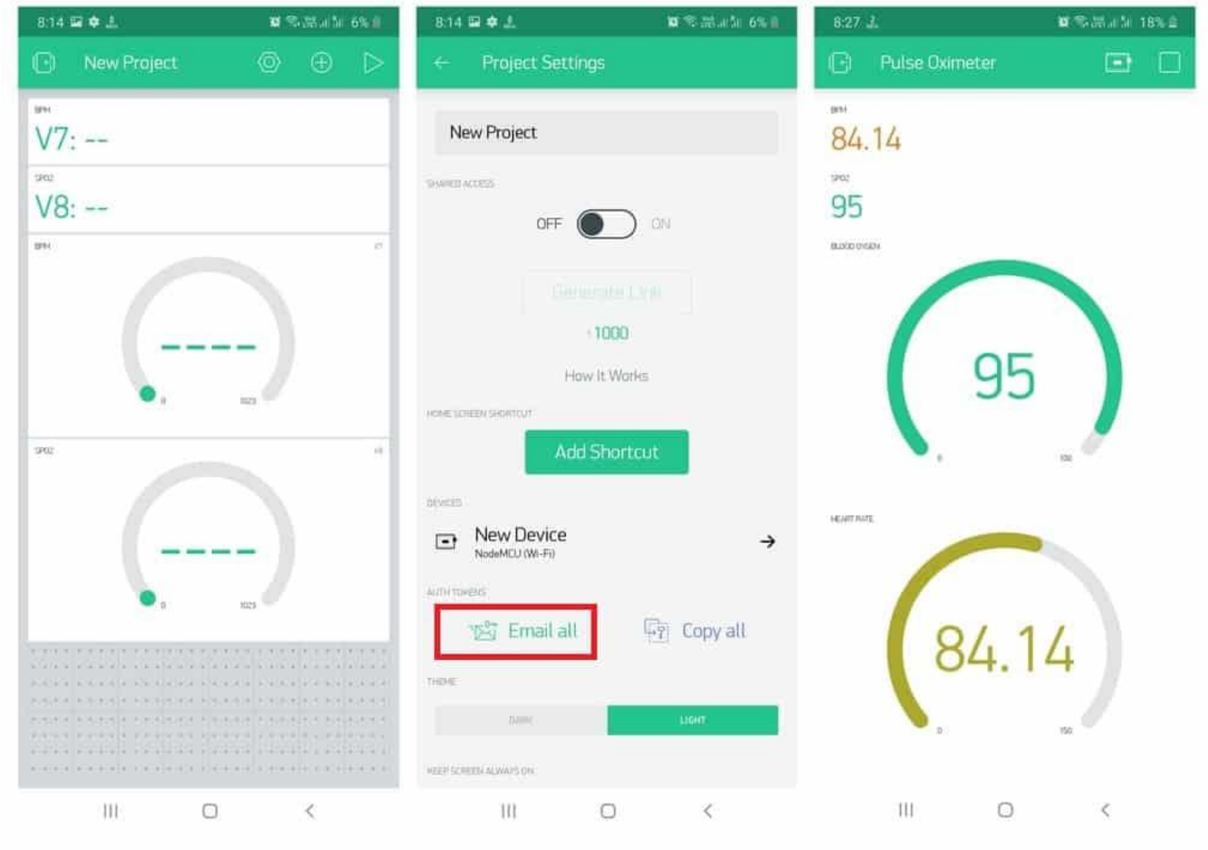
6.2.1 Create a New Project

- New Project Button: Click the "+" button (usually located at the bottom right corner) to create a new project.
- Project Name: Enter a descriptive name for your project, like "Heart Rate Monitor with ESP8266".
- Connection Type: Choose "Wi-Fi" as the connection type for your ESP8266.
- Board Selection: Select "ESP8266" from the list of available boards.



6.2.2 Adding a Virtual Pin for Heart Rate Data

- Hardware Tab: Click on the "Hardware" tab located at the bottom of the screen.
- Create New Pin: Tap the "+" button in the top right corner to create a new virtual pin.
- Pin Type: Choose "Value" as the pin type since you'll be displaying a numerical heart rate value.
- Pin Name: Give your virtual pin a descriptive name, like "HeartRate".
- Pin Number (Optional): Blynk assigns a default pin number. You can keep this or choose a custom number (between V0 and V9) for better organization.



6.2.3 Building the App Interface

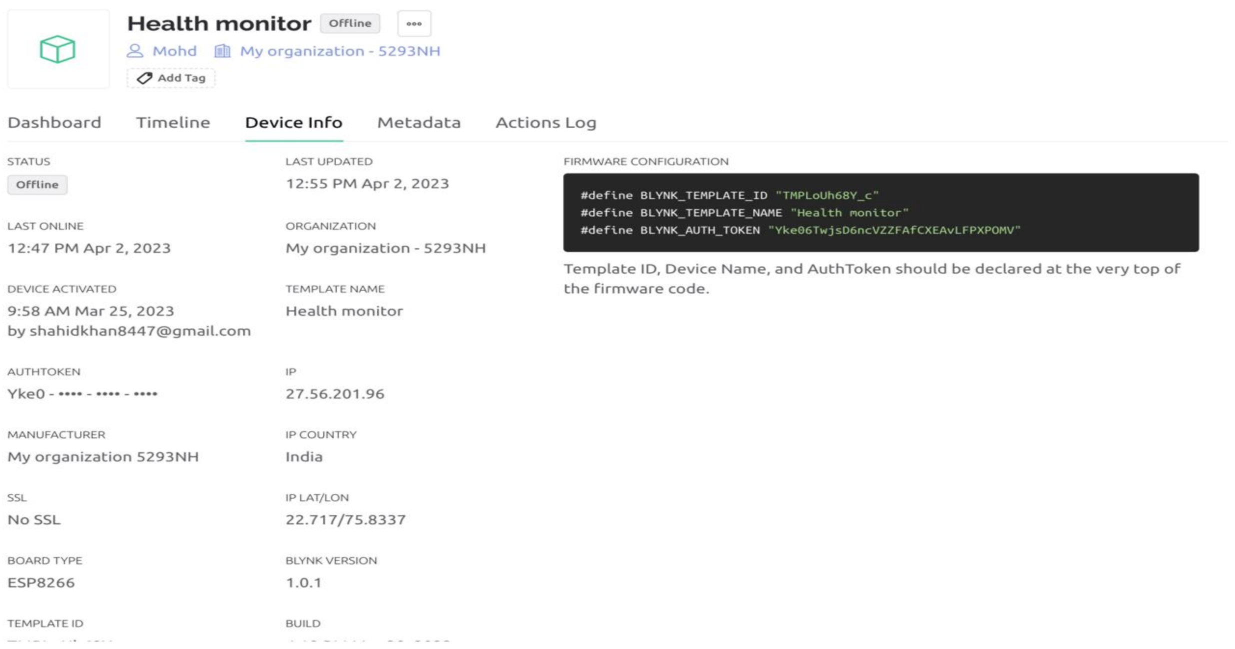
- Widgets Tab: Click on the "Widgets" tab located at the bottom of the screen.
- Drag and Drop Widget: Drag and drop a "Value Display" widget onto the workspace in the app editor.
- Configure Widget: Tap on the "Value Display" widget to configure its properties.
- Pin Selection: Select the virtual pin you created earlier (e.g., "HeartRate") to display the heart rate data on this widget.
- Customization: You can further customize the appearance of the widget by changing its color, font size, etc.

6.2.4 Saving the Project

- Once you've configured the virtual pin and added the widget, click the ">" button located at the top right corner to save your Blynk project.

6.2.5 Note the Auth Token

- In the project settings or under your profile information, locate your Blynk Auth Token. You'll need this token in your ESP8266 code to connect to the Blynk cloud server.



STATUS
Offline

LAST UPDATED
12:55 PM Apr 2, 2023

LAST ONLINE
12:47 PM Apr 2, 2023

ORGANIZATION
My organization - 5293NH

DEVICE ACTIVATED
9:58 AM Mar 25, 2023
by shahidkhan8447@gmail.com

TEMPLATE NAME
Health monitor

AUTHTOKEN
Yke0 - •••• - •••• - ••••

IP
27.56.201.96

MANUFACTURER
My organization 5293NH

IP COUNTRY
India

SSL
No SSL

IP LAT/LON
22.717/75.8337

BOARD TYPE
ESP8266

BLYNK VERSION
1.0.1

TEMPLATE ID

BUILD

FIRMWARE CONFIGURATION

```
#define BLYNK_TEMPLATE_ID "TMPLoUh68Y_c"
#define BLYNK_TEMPLATE_NAME "Health monitor"
#define BLYNK_AUTH_TOKEN "Yke06TwjsD6ncVZZFAfCXEAvLFPXPOMV"
```

Template ID, Device Name, and AuthToken should be declared at the very top of the firmware code.

6.3 Connecting Hardware to Blynk

6.3.1 Install Blynk Library

- Open the Arduino IDE.
- Go to "Sketch" -> "Include Library" -> "Manage Libraries".
- Search for "Blynk" in the search bar and install the "Blynk by Blynk" library.

6.3.2 Install Heart Rate Sensor Library

- Your heart rate sensor requires a specific library for interpretation, search for and install the relevant library following the same steps as above.

6.3.3 Connect Hardware (refer to a circuit diagram for specific pin connections)

- Connect the ESP8266 board to a power source (e.g., USB cable to your computer).
- Connect the heart rate sensor to the ESP8266 board according to the sensor's specifications. This typically involves connecting the sensor's positive wire to a power pin (e.g., 3.3V), negative wire to ground (GND), and output wire to an analog input pin (e.g., A0).

6.3.4 Configure Blynk App:

- Follow the steps outlined previously (in response "Detailed steps for configuring the blynk app for heart rate measurement using esp 8266") to create a new project in the Blynk app, add a "Value" virtual pin for heart rate data, and optionally build your app interface with a "Value Display" widget.
- Note down your Blynk Auth Token from the app settings or profile information.

6.3.5 Write and Upload Arduino Code

- Open a new sketch in Arduino IDE.
- Include the necessary libraries.
- Define your Blynk Auth Token and virtual pin number in the code.
- Set up Wi-Fi connection on the ESP8266 with your Wi-Fi credentials (SSID and password).

- Initialize communication with Blynk using your Auth Token.
- Read the raw analog data from the heart rate sensor.
- Process the sensor data (filtering noise if needed) and convert it to a meaningful heart rate value (beats per minute). This might involve calculations specific to your sensor.
- Send the heart rate data to the designated virtual pin (e.g., V0) on the Blynk cloud server using the Blynk library functions.

6.3.6 Upload Code and Test

- Connect the ESP8266 board to your computer using a USB cable.
- In Arduino IDE, select the correct ESP8266 board type and programmer from the "Tools" menu.
- Select the COM port your ESP8266 is connected to.
- Click the "Upload" button to compile and upload the code to your ESP8266 board.

6.3.7 Test Functionality

- Open the Blynk app on your smartphone/tablet.
- Ensure your ESP8266 and smartphone are connected to the same Wi-Fi network.
- If everything is connected and programmed correctly, you should see the heart rate data being displayed on the "Value Display".

6.4 Connecting Ubidots to blynk

6.4.1 Setting Up Ubidots:

- Create a free Ubidots account if you don't have one already (https://ubidots.com/).
- Once logged in, navigate to the "Devices" section and click "Add Device."
- Choose a name for your device (e.g., "Heart Rate Monitor").

6.4.2 Modifying the Blynk Code:

- Locate the section in your code where Ubidots credentials are defined (around lines 40-50).

- Replace the placeholder values with your actual Ubidots information:
- "BBUS-VW4Fjz8eEFs4CvBVQQeS0VDjmclK5G" with your Ubidots Token.
- "heart-beat-monitoring" with your chosen Device Label.
- Update "heartrate" and "SPo2" variable labels to match your Ubidots variables (if applicable).

6.4.3 Uploading the Code:

- Ensure you have the necessary libraries installed in your Arduino IDE for Blynk, WiFi, MQTT, MAX30105 sensor, and any custom libraries used in your code.
- Upload the modified code to your ESP8266 board using the Arduino IDE.

6.4.4 Visualizing Data in Ubidots:

- In your Ubidots dashboard, navigate to your created device.
- You should see your heart rate data plotted over time in a graph.
- You can customize the graph (timeframe, units, etc.) using the options provided.
- If your code publishes average heart rate data as well, you can add another chart to visualize it.

6.5 Source code

```
#define BLYNK_TEMPLATE_ID "TMPL378Y_UgCS"
#define BLYNK_TEMPLATE_NAME "Heart Rate Monitor"
#include <Wire.h>
#include "MAX30105.h"
#include "heartRate.h"
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <BlynkSimpleEsp8266.h>

MAX30105 particleSensor;
```

```
const byte RATE_SIZE = 4;
byte rates[RATE_SIZE];
byte rateSpot = 0;
long lastBeat = 0;
float beatsPerMinute;
int beatAvg;

// WiFi credentials
char auth[] = "8wkGbDYRqrB8B4cOLVqovi0zFKyjTOgE";
char ssid1[] = "PO”;
char pass2[] = "123";
const char* ssid = "PO";
const char* password = "123";

// Ubidots credentials
const        char*        ubidotsToken        =        "BBUS-
VW4Fjz8eEFs4CvBVQQeS0VDjmclK5G";
const char* deviceLabel = "heart-beat-monitoring";
const char* variableLabel1 = "heartrate";
const char* variableLabel2 = "SPo2";
const char* mqttBroker = "industrial.api.ubidots.com";
const int mqttPort = 1883;

WiFiClient wifiClient;
PubSubClient client(wifiClient);
#define REPORTING_PERIOD_MS 350
#define VIRTUAL_PIN_BPM V7
#define VIRTUAL_PIN_SPO2 V6
BlynkTimer timer;
bool fingerDetected = false;
```

```
#define MOVING_AVERAGE_SIZE 5
float bpmBuffer[MOVING_AVERAGE_SIZE];
float avgBuffer[MOVING_AVERAGE_SIZE];
int bufferIndex = 0;

void setup() {
  Serial.begin(115200);
  delay(1000); // Delay to ensure Serial Monitor is ready

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected");

  client.setServer(mqttBroker, mqttPort);

  pinMode(16, OUTPUT);
  Blynk.begin(auth, ssid1, pass2);

  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) {
    Serial.println("MAX30105 was not found. Please check
wiring/power.");
    while (1);
  }
  Serial.println("Place your index finger on the sensor with
steady pressure.");
  particleSensor.setup();
  particleSensor.setPulseAmplitudeRed(0x0A);
  particleSensor.setPulseAmplitudeGreen(0);
  timer.setInterval(REPORTING_PERIOD_MS, sendSensorData);
  // Ubidots MQTT connection
  reconnect();
}

void loop() {
  Blynk.run();
  timer.run();
  long irValue = particleSensor.getIR();
  if (irValue >= 50000 && !fingerDetected) {
    Serial.println("Finger detected");
    fingerDetected = true;
    // Start readings or take any other action as needed
  } else if (irValue < 50000 && fingerDetected) {
    Serial.println("Finger not detected, stopping readings");
    fingerDetected = false;
    // Clear the display or take any other action as needed
  }

}

void sendSensorData() {
  if (!fingerDetected) {
    // Do not send or print readings if finger is not detected
    return;
  }

  long irValue = particleSensor.getIR();

  if (checkForBeat(irValue)) {
    long delta = millis() - lastBeat; // Define delta here
    lastBeat = millis();
    beatsPerMinute = 60 / (delta / 1000.0);

    if (beatsPerMinute < 120 && beatsPerMinute > 20) {
      rates[rateSpot++] = (byte)beatsPerMinute;
      rateSpot %= RATE_SIZE;
      beatAvg = 0;
      for (byte x = 0; x < RATE_SIZE; x++)
        beatAvg += rates[x];
      beatAvg /= RATE_SIZE;
    }

    // Publish data to Ubidots
    char payload[150];
    sprintf(payload,                "{\"%s\":%d,\"%s\":%d}",
variableLabel1,     (int)beatsPerMinute,     variableLabel2,
beatAvg);
    client.publish((String("/v1.6/devices/")                +
deviceLabel).c_str(), payload);
    Serial.println("Data published to Ubidots");
  }
  float temperatureC = particleSensor.readTemperature();
  float temperatureF = particleSensor.readTemperatureF();

  // Update moving average buffers
  bpmBuffer[bufferIndex] = beatsPerMinute;
  avgBuffer[bufferIndex] = beatAvg;
  bufferIndex = (bufferIndex + 1) % MOVING_AVERAGE_SIZE;

  // Calculate moving average
  float bpmSum = 0, avgSum = 0;
  for (int i = 0; i < MOVING_AVERAGE_SIZE; i++) {
    bpmSum += bpmBuffer[i];
    avgSum += avgBuffer[i];
  }
  beatsPerMinute = bpmSum / MOVING_AVERAGE_SIZE;
  beatAvg = avgSum / MOVING_AVERAGE_SIZE;

  // Publish data to Blynk
  if (beatsPerMinute != 0.00) {
    Blynk.virtualWrite(VIRTUAL_PIN_BPM, beatsPerMinute);
    Serial.print(" BPM=");
    Serial.print(beatsPerMinute);
  }
  if (beatAvg != 0.00) {
    Blynk.virtualWrite(VIRTUAL_PIN_SPO2, beatAvg);
    Serial.print(", Avg BPM=");
    Serial.print(beatAvg);
  }
  Serial.println();

  delay(1); // Adjust as needed for sampling rate
}

bool checkForBeat(long irValue) {
  static long previousValue = 0;
  static long currentValue = 0;

  // Update the current value
  currentValue = irValue;
  // Check if there is a beat by comparing current and
previous values
  if (currentValue > previousValue && currentValue > 50000)
{
    previousValue = currentValue;
    return false; // No beat detected
  } else {
    previousValue = currentValue;
    return true; // Beat detected
  }
}

void reconnect() {
  while (!client.connected()) {
    Serial.println("Attempting MQTT connection...");
    if (client.connect("ESP8266Client", ubidotsToken, "")) {
      Serial.println("Connected to Ubidots");
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" retrying in 5 seconds");
      delay(5000);
    }
  }
}
```

This code implements a heart rate monitor using the MAX30105 sensor, Blynk for visualisation, and Ubidots for data storage. Here's a breakdown of the code:

6.5.1 Libraries and Definitions

Includes libraries for:

- I2C communication (Wire.h)

- MAX30105 sensor (MAX30105.h) - likely a custom library for the sensor
- Heart rate calculation (heartRate.h) - likely a custom library for processing sensor data
- WiFi connection (ESP8266WiFi.h)
- MQTT communication (PubSubClient.h)
- Blynk integration (BlynkSimpleEsp8266.h)

Defines constants for:

- Blynk template ID and name (for online project management)
- WiFi credentials (SSID and password)
- Ubidots credentials (token, device label, variable labels, broker address, and port)
- Reporting period (how often to send data - 350ms)
- Blynk virtual pin numbers for heart rate and SpO2 (assumed to be calculated later)
- Moving average buffer size (for smoothing data)

6.5.2 Global Variables

- particleSensor: An instance of the MAX30105 sensor class.
- rates: Array to store recent heart rate readings for averaging.
- rateSpot: Index for the rates array.
- lastBeat: Millisecond timestamp of the last detected heartbeat.
- beatsPerMinute: Current calculated heart rate.
- beatAvg: Average heart rate based on recent readings.
- fingerDetected: Flag to indicate if a finger is on the sensor.
- bpmBuffer and avgBuffer: Buffers for storing moving average data for heart rate and average heart rate.
- bufferIndex: Index for the moving average buffers.

6.5.3 Setup Function

- Initializes serial communication for debugging.
- Connects to the WiFi network specified by credentials.
- Sets up the Ubidots MQTT client.

- Sets pin 16 as output (possibly for an LED indicator).
- Initializes Blynk with credentials.
- Begins communication with the MAX30105 sensor and configures pulse amplitude settings.
- Starts a Blynk timer to send sensor data periodically (every 350ms).
- Connects to the Ubidots MQTT broker.

6.5.4 Loop Function

- Runs Blynk tasks to maintain connection and handle incoming messages.
- Executes the Blynk timer callback (sendSensorData) periodically.
- Checks for finger detection based on IR sensor value:
- If finger is detected, starts readings.
- If finger is removed, stops readings.

6.5.5 sendSensorData Function

- Skips sending data if a finger is not detected.
- Reads the IR value from the sensor.
- Calls checkForBeat to detect a heartbeat based on changes in IR value.
- If a beat is detected:
- Calculates heart rate based on the time between beats.
- Filters out heart rates outside a reasonable range (20-120 bpm).
- Updates the rates array for averaging and calculates the moving average heart rate.
- Publishes data (heart rate and average heart rate) to Ubidots using MQTT.
- Reads temperature data from the sensor (not currently used).
- Updates moving average buffers for heart rate and average heart rate.
- Calculates overall moving average heart rate and average heart rate.
- Sends heart rate and average heart rate data to Blynk virtual pins if values are non-zero.
- Prints heart rate and average heart rate data to the serial monitor (for debugging).

- Adds a small delay (adjustable) to control sampling rate.

6.5.6 CheckForBeat Function

- Compares current and previous IR sensor readings to detect a potential peak (heartbeat).
- Returns false if no beat is detected, true if a beat is likely.

6.5.7 Reconnect Function

- Attempts to connect to the Ubidots MQTT broker until successful.

6.5.8 Overall Functionality

This code demonstrates a basic heart rate monitoring system using an Arduino board (ESP8266), a MAX30105 sensor, Blynk for real-time data visualization, and Ubidots for data storage. The code:

- Read heart rate data from the sensor.
- Implements finger detection to avoid erroneous readings when no finger is present.
- Filters out unreasonable heart rate values.
- Calculates a moving average for heart rate smoothing.
- Sends heart rate and average heart rate data to Blynk.

CHAPTER 7

LIMITATIONS AND CHALLENGES

While Blynk IoT offers a convenient platform for building heart rate monitoring systems, there are limitations and challenges to consider:

7.1 Accuracy

7.1.1 Sensor Limitations

The accuracy of heart rate measurement heavily relies on the chosen sensor and its placement. PPG (photoplethysmography) sensors used in most Blynk projects are susceptible to:

- Movement Artifacts: Movement during measurement can introduce noise into the signal, leading to inaccurate readings.
- Sensor Placement: Improper placement on the finger, earlobe, or forehead can affect signal quality and accuracy.
- Skin Tone and Perfusion: Variations in skin tone and blood flow can impact sensor performance.
- Data Processing: Basic filtering techniques used in Blynk projects might not be sufficient to remove all noise from the sensor signal, especially during exercise or with significant movement. Advanced algorithms might be required for improved accuracy.

7.2 Blynk Platform Limitations

7.2.1 Limited Data Storage

The free tier of Blynk offers limited data storage for historical heart rate data. Upgrading to a paid plan might be necessary for extended data storage and analysis.

7.2.2 Security Considerations

While Blynk offers basic security features, transmitting sensitive health data like heart rate requires additional measures to ensure data privacy and security. Implementing encryption and secure communication protocols is crucial.

7.3 Overall System Challenges

7.3.1 Calibration

Most Blynk projects lack built-in calibration routines. Regularly comparing readings with a medical-grade heart rate monitor can help identify and adjust for potential biases.

7.3.2 User Dependence

Accurate results rely on proper sensor placement and minimal movement during measurement. User training and clear instructions are essential to ensure reliable data collection.

7.3.3 Limited Functionality

Basic Blynk projects typically display heart rate data but lack advanced features like heart rate zone monitoring, historical data analysis, or integration with fitness applications.

7.3.4 Not a Medical Device

Blynk-based heart rate monitoring systems are not replacements for medical-grade equipment. They cannot diagnose heart conditions and should not be used for critical medical decision-making.

CHAPTER 8
FUTURE REFRENCES FOR HEART RATE MEASUREMENT

Future References for Heart Rate Measurement using Blynk IoT

The field of heart rate monitoring with Blynk IoT is constantly evolving. Here's a breakdown of potential future references you might encounter:

8.1Advanced Sensor Integration

Biometric Sensors: Explore the integration of advanced biometric sensors like electrocardiogram (ECG) sensors for more accurate heart rate measurements and insights into heart rhythm. References might include research papers or project examples using ECG sensors with Blynk.

Multi-Sensor Fusion: Look for resources on combining data from Blynk-compatible pulse oximeters, temperature sensors, and accelerometers for a more holistic view of health metrics. This might involve references on sensor fusion techniques and Blynk libraries supporting multiple sensor data streams.

8.2 Improved Data Processing and Analysis

Machine Learning Integration: Explore the use of machine learning algorithms on Blynk platforms (if supported) to analyze heart rate data for anomaly detection, arrhythmia identification, or activity recognition. References could include research papers or tutorials on implementing machine learning for heart rate analysis on Blynk-compatible devices.

Advanced Signal Processing Techniques: Look for resources on advanced signal processing techniques like noise filtering and artifact removal to improve the accuracy and reliability of heart rate data collected by Blynk-compatible sensors. References could include research papers or tutorials focusing on signal processing for PPG (photoplethysmography) signals used in heart rate sensors.

CONCLUSION

Heart rate monitoring is essential for maintaining health, tracking fitness progress, detecting potential health issues, and managing stress. Various methods such as ECG, PPG, and SCG provide different advantages and limitations in heart rate measurement. Future advancements aim to improve accuracy, enable continuous monitoring, and facilitate remote health management.

The integration of IoT platforms like Blynk, along with sensors like MAX30102 and microcontrollers such as ESP8266, allows for the development of efficient and user-friendly heart rate monitoring systems. These systems enable users to track their heart rate in real-time, optimize fitness routines, and detect health anomalies early, contributing to overall well-being. The Blynk IoT platform's ease of use and versatile features make it an ideal choice for creating scalable and flexible health monitoring applications.

REFERENCES

[1] HEART RATE MONITORING SYSTEM, 2018
JETIR May 2018, Volume 5, Issue 5,
www.jetir.org (ISSN-2349-5162)

[2] Heartbeat Rate Monitoring System by Pulse
Technique Using HB Sensor ISBN No.978-1-
4799-3834-6/14/ 2014 IEEE.

[3] IoT based System for Heart Rate Monitoring,
International Journal of Engineering Research &
Technology (IJERT) http://www.ijert.org ISSN:
2278-0181 IJERTV9IS070673 www.ijert.org
Vol. 9 Issue 07, July-2020

[4] Heart Rate Monitoring Applications and
Limitations Juul Achten and Asker E.
Jeukendrup, Sports Med 2003; 33 (7): 517-538
REVIEW ARTICLE 0112-1642/03/0007-
0517/0

[5] Heart Rate Monitoring System using Heart Rate
Sensor and Arduino Uno with Web Application,
International Journal of Engineering and
Advanced Technology (IJEAT) ISSN: 2249 –
8958, Volume-8 Issue-4, April, 2019