Established – 1961 Subject: IOT

SEVA SADAN'S

R. K. TALREJA COLLEGE

OF

ARTS, SCIENCE & COMMERCE

ULHASNAGAR – 421 003

CERTIFICATE

This is to certify that Mr./Ms. Akash Ashok Balghare **of S.Y. Computer Science (SYCS) Roll No.** 2524002 **has satisfactorily completed The Internet Of Thing Mini Project entitled** Smart Heart Rate Monitor System **during the academic year 2025 – 2026, as a part of the practical requirement. The project work is found to be satisfactory and is approved for submission.**

PROF. INCHARGE **HEAD OF DEPT**

INDEX

Sr. No.	Chapters	Page No.
1	Introduction	3- 7
2	Requirement Specification	8-9
3	System Design	10-27
4	Implementation	28-37
5	System Testing and Result	38-43
6	Future Scope and Conclusion	44-45
7	References	46-47
8	Glossary	48-49

INTRODUCTION

IoT Based Heart Rate Monitoring System Introduction:

The **IoT Based Heart Rate Monitoring System** is a simple and practical project that focuses on measuring and monitoring the human heart rate using IoT technology. The system is designed to calculate the heart rate in terms of **Beats Per Minute (BPM)** and display the result on an OLED screen.

The project is built using an **ESP8266 microcontroller** as the main controller, which interfaces with a **pulse sensor** to detect heartbeats and an **OLED display** to show the output. The system also uses **WiFi connectivity** to upload the heart rate data to the cloud platform for remote monitoring.

This project provides a basic understanding of how IoT can be used in healthcare applications by combining sensors, microcontrollers, and cloud services to perform real-time monitoring.

Motivation :

The motivation behind developing the **IoT Based Heart Rate Monitoring System** is the growing need for simple and affordable health monitoring solutions. Monitoring heart rate regularly can help in understanding a person's health condition and detecting abnormalities at an early stage.

This project provides an educational and practical way for students to learn about **IoT concepts**, **sensor interfacing**, and **wireless communication**. It also demonstrates real-world applications of IoT in healthcare, remote patient monitoring, and fitness tracking systems.

Problem Definition :

The main problem addressed by this project is the need for a **basic and low-cost system** that can measure heart rate and make the data accessible remotely. Traditional heart rate monitoring devices can be expensive and not easily accessible to everyone.

This project aims to solve the problem by designing a system that can **detect heartbeats using a pulse sensor**, **calculate BPM accurately**, **display the result locally**, and **upload the data to the cloud** using IoT technology. The system helps in understanding how real-time health data can be monitored and stored efficiently.

How it Works :

The **IoT Based Heart Rate Monitoring System** works using an ESP8266 microcontroller to interface with a pulse sensor and other components.
The pulse sensor detects changes in blood flow through the finger and generates an analog signal corresponding to heartbeats.

The ESP8266 reads this signal, processes it, and calculates the heart rate in terms of **Beats Per Minute (BPM)**. The calculated BPM is displayed on an OLED screen for local monitoring. Using built-in WiFi connectivity, the system uploads the heart rate data to the ThingSpeak cloud platform, allowing users to monitor the data remotely.

This combination of sensor input, data processing, and cloud connectivity enables real-time heart rate monitoring in a simple and effective manner.

Key Features :

Heart Rate Measurement :

Uses a pulse sensor to accurately detect heartbeats and calculate BPM.

OLED Display Output :

Displays real-time measurement status and final heart rate value on an OLED screen for easy viewing.

IoT Cloud Integration :

Uploads heart rate data to the ThingSpeak cloud platform using WiFi for remote monitoring and data analysis.

Real-Time Monitoring :

Provides continuous and timely heart rate updates, making it useful for basic health tracking.

User-Friendly Operation :

Simple design and easy-to-understand working make the system suitable for beginners and students.

Educational Value :

Helps students learn about IoT, sensor interfacing, data processing, and wireless communication through a practical healthcare application.

_Scope of the Project :

The **IoT Based Heart Rate Monitoring System** has wide scope in the field of healthcare and IoT applications. The system provides a simple and cost-effective way to monitor heart rate in real time. It can be used for basic health monitoring, fitness tracking, and educational purposes.

In academic environments, this project helps students understand the integration of sensors, microcontrollers, and cloud platforms. The system also demonstrates real-world applications of IoT in healthcare, such as remote patient monitoring and health data analysis.

The project can be further enhanced by adding more sensors, mobile application support, alert systems, or advanced data analytics, increasing its future potential.

_Academic Institutions :

The **IoT Based Heart Rate Monitoring System** can be used as a learning tool in academic institutions. Students can gain hands-on experience with IoT, sensor interfacing, and wireless communication. It helps in understanding practical implementation of healthcare monitoring systems.

_Businesses and Healthcare Services :

Healthcare centers, gyms, and wellness programs can use this system for basic heart rate monitoring. The data collected can help in fitness tracking and health analysis. Its low-cost design makes it suitable for small clinics and health awareness programs.

_Data Security and Health Monitoring :

The system can be improved to store and manage health data securely on cloud platforms. This helps in maintaining records and ensuring safe access to heart rate information for future reference and analysis.

Online Learning Platforms :

Online learning platforms can use this project to teach IoT and healthcare monitoring concepts. Students can learn remotely by building and testing the system at home, making learning more interactive and practical.

Objective :

The objective of the **IoT Based Heart Rate Monitoring System** project is to design a simple and reliable system capable of measuring and monitoring human heart rate using IoT technology. The project aims to provide an easy-to-use solution for basic health monitoring while helping students understand IoT concepts and sensor integration.

Key goals include :

Promote Learning in IoT and Healthcare :

Encourage students to learn about IoT, microcontrollers, and healthcare monitoring through hands-on practical experience.

Accurate Heart Rate Measurement :

Develop a system that can detect heartbeats and calculate heart rate in terms of BPM using a pulse sensor.

Real-Time Monitoring :

Enable real-time display of heart rate on an OLED screen and upload data to the cloud for remote access.

User-Friendly Design :

Create a simple and easy-to-operate system that can be used by beginners without complex setup.

Cloud Data Storage :

Use IoT cloud platforms to store and visualize heart rate data for future reference and analysis.

Support Future Enhancements :

Provide a foundation for adding advanced features such as alerts, mobile applications, or additional health sensors.

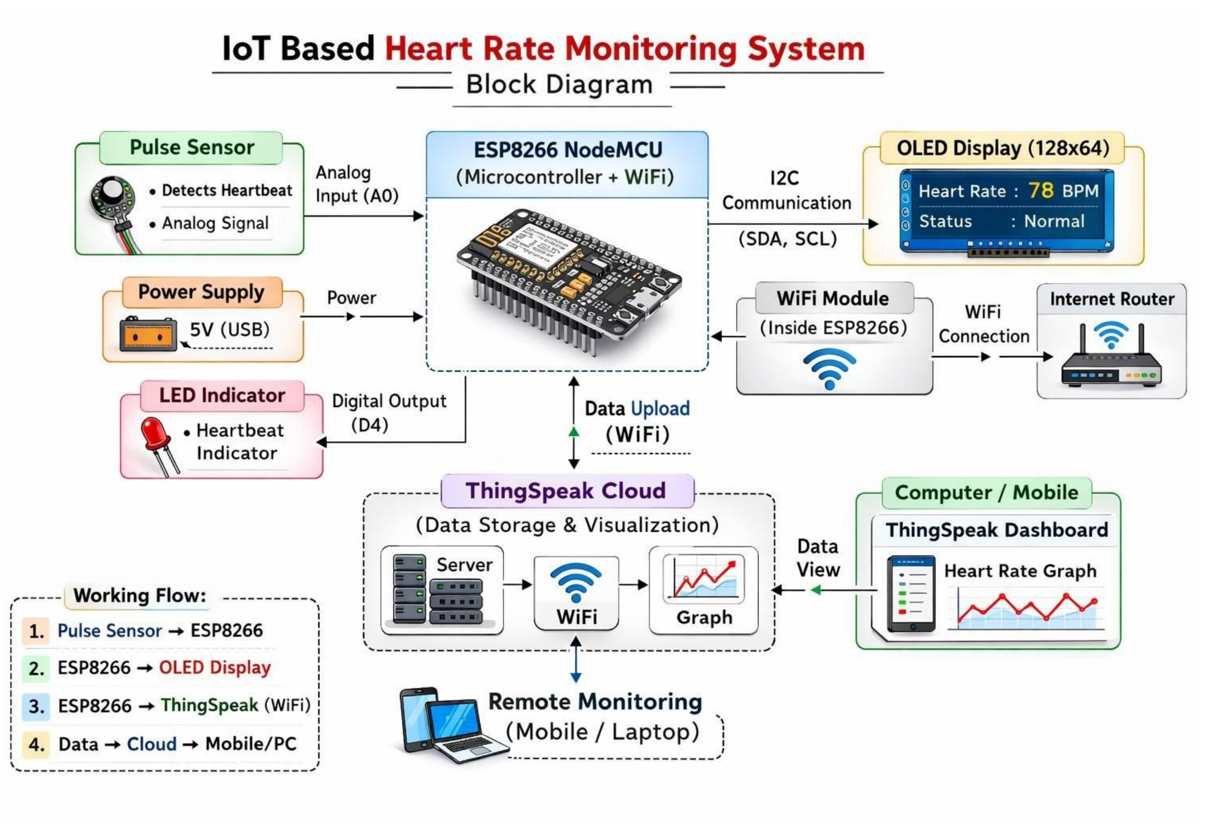
REQUIREMENT SPECIFICATION

Component	**Specification**
Microcontroller	ESP8266 NodeMCU
Heart Rate Sensor	Pulse Sensor for detecting heartbeats
Display	OLED Display (128×64)
Indicator	LED for heartbeat indication
Connecting Wires	Jumper wires for circuit connections
Breadboard	For mounting and testing components
Power Supply	USB power or 5V external supply

Component	**Specification**
Arduino IDE	Software for programming the ESP8266
ESP8266 Board Package	Required to program NodeMCU
Display Libraries	Adafruit SSD1306 and Adafruit GFX
IoT Platform	ThingSpeak for cloud data storage
WiFi Network	Internet connection for data upload

SYSTEM DESIGN

<u>Block Diagram</u> :



The block diagram represents the overall structure and interaction between the main components of the IoT Based Heart Rate Monitoring System. It explains how the sensor, microcontroller, display, and cloud platform work together to measure and monitor heart rate.

ESP8266 NodeMCU :

Acts as the main controller of the system. It processes the signal received from the pulse sensor, calculates the heart rate, controls the OLED display, and uploads data to the cloud using WiFi.

Pulse Sensor :

Detects heartbeats by sensing changes in blood flow through the finger and sends analog signals to the ESP8266.

OLED Display (128×64) :

Displays the measuring status, progress, and final BPM value for local monitoring.

WiFi Connectivity :

Allows the ESP8266 to connect to the internet and send heart rate data to the ThingSpeak cloud platform.

ThingSpeak Cloud Platform :

Stores and visualizes the heart rate data in graphical form for remote monitoring and analysis.

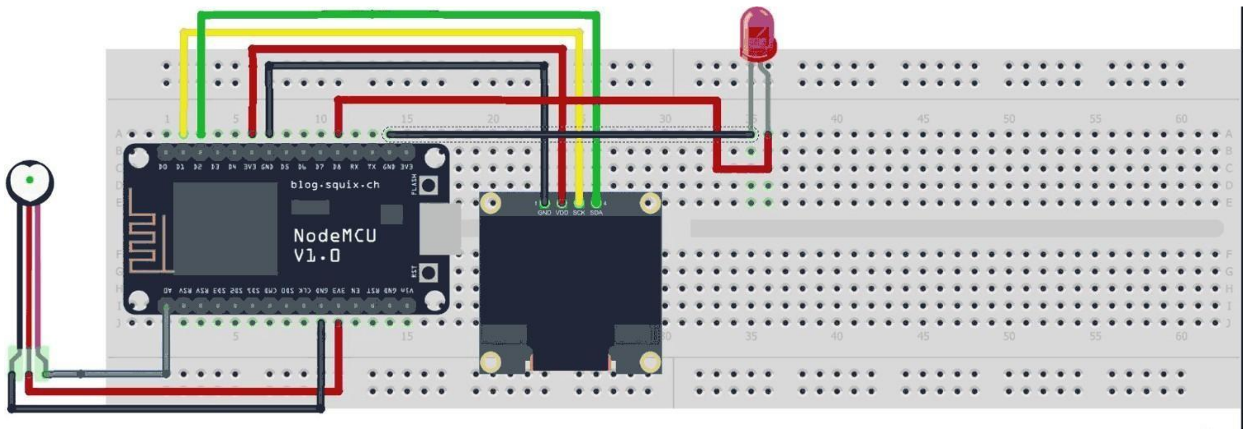
LED Indicator :

Blinks whenever a heartbeat is detected, providing visual confirmation of pulse detection.

Power Supply :

Provides required power to the ESP8266 and connected components through USB or external source.

Circuit Diagram :



Description :

The block diagram demonstrates the interaction between the key components of the **IoT Based Heart Rate Monitoring System**, such as the ESP8266 microcontroller, pulse sensor, OLED display, WiFi module, and cloud platform. The ESP8266 receives analog signals from the pulse sensor to detect heartbeats and calculate the heart rate in BPM.

The calculated heart rate is displayed on the OLED screen for local monitoring, while the WiFi connectivity allows the system to send heart rate data to the ThingSpeak cloud platform for remote access. An LED indicator provides visual feedback for each detected heartbeat. Power is supplied to all components through a USB or external power source.

Key Steps of Circuit Diagram :

ESP8266 Setup :

Connect the ESP8266 NodeMCU as the main controller of the system. All components such as the pulse sensor, OLED display, LED, and WiFi communication are controlled through this board.

Pulse Sensor :

Connect the pulse sensor output pin to the analog pin (A0) of the ESP8266. This sensor detects heartbeats by sensing changes in blood flow through the finger.

OLED Display (128×64) :

Connect the OLED display to the ESP8266 using I2C pins (SDA and SCL). The display is used to show the measuring status and final BPM value.

LED Indicator :

Connect an LED to a digital pin of the ESP8266 with a resistor. The LED blinks whenever a heartbeat is detected, providing visual feedback.

WiFi Connectivity :

Use the built-in WiFi module of the ESP8266 to connect to the internet. This allows heart rate data to be uploaded to the ThingSpeak cloud platform.

Power Supply :

Power the ESP8266 and connected components using a USB cable or a regulated 5V power supply to ensure stable operation.

These steps ensure proper integration of all components for accurate heart rate monitoring.

Benefits of Circuit Diagram :

Clear Visualization :

Provides a clear view of how all components are connected, helping to understand the flow of signals in the system.

Error Detection :

Helps in identifying wiring or connection errors before implementing the actual hardware.

Documentation :

Acts as an important reference for future modifications, troubleshooting, or project replication.

Ease of Assembly :

Makes the hardware assembly process easier by clearly showing pin connections and component placement.

This is important for projects like the **IoT Based Heart Rate Monitoring System** to ensure safe and correct operation.

When to Use a Circuit Diagram ?

Design Phase :

Used while planning the project to visualize component connections and system structure.

Troubleshooting :

Helps in diagnosing problems by checking connections and signal flow.

Documentation :

Essential for documenting the project for reports, submissions, or sharing with others.

Education :

Helps students understand electronic components, circuit connections, and working principles clearly.

Assembly Guidance :

- The assembly guidance provides a clear roadmap for assembling the **IoT Based Heart Rate Monitoring System**. It helps ensure that all components such as the ESP8266, pulse sensor, OLED display, and LED are connected correctly. Proper pin connections and correct placement of components reduce the chances of errors and ensure smooth functioning of the system.

Circuit Model Testing Methods :

Simulation Testing :

Simulation tools like **Proteus** or **Tinkercad** can be used to understand the circuit design before physical assembly. This helps in identifying possible design issues at an early stage.

Continuity Testing :

After assembling the circuit, a multimeter can be used to check continuity. This ensures that all connections are properly made and there are no short circuits.

Functional Testing :

Power the circuit and verify that each component such as the pulse sensor, OLED display, LED indicator, and WiFi connection works correctly as expected.

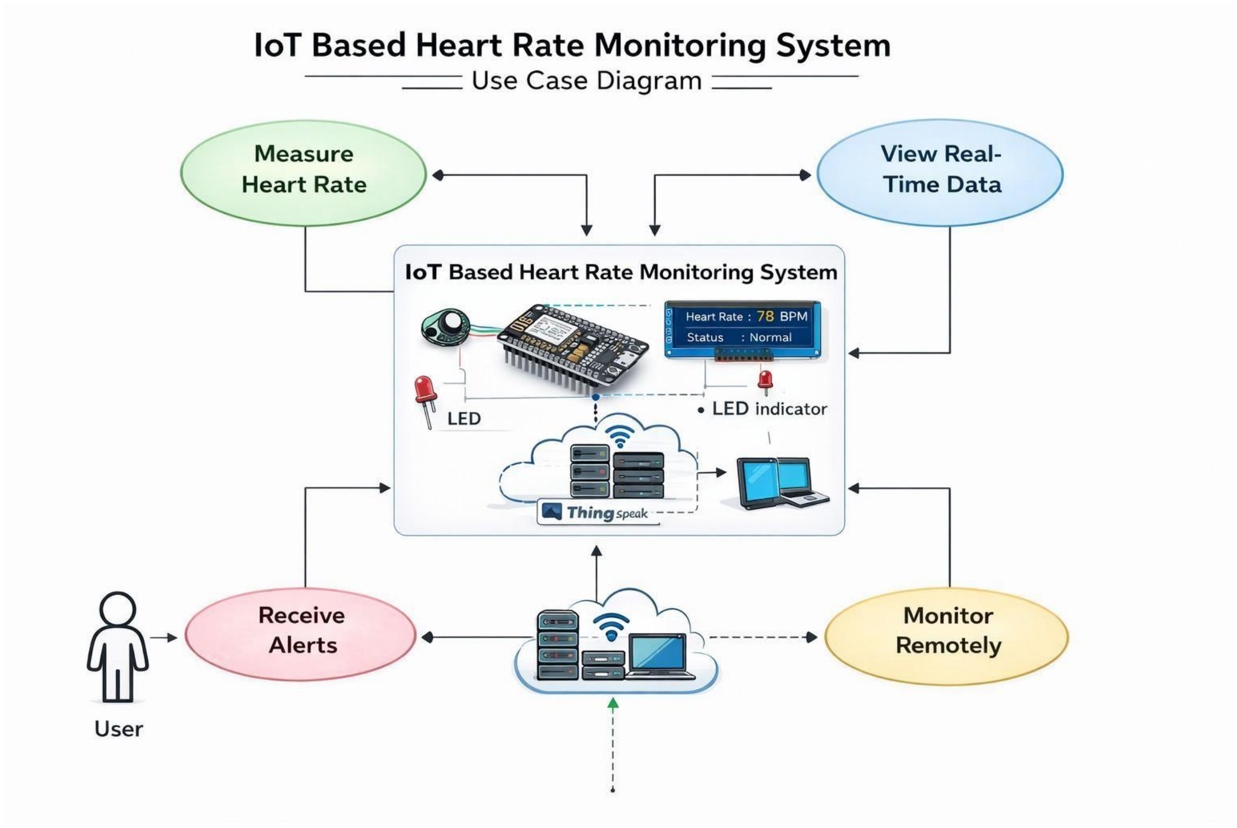
Load Testing :

Test the system under normal operating conditions to ensure that it remains stable while measuring heart rate and uploading data to the cloud.

Debugging :

If any issue or unexpected behavior occurs, debugging is done by checking sensor readings, pin connections, and code logic to identify and fix the problem.

Use Case Diagram :



Description :

1. The **Use Case Diagram for the IoT Based Heart Rate Monitoring System** illustrates the interactions between the user and the system's functionalities. It provides a visual representation of the key actions performed by the user and how the system responds to those actions.

2. The main actors identified in the system are **User**, **Pulse Sensor**, and **Cloud Platform (ThingSpeak)**. Below is a detailed explanation of the use cases included in the diagram.

Measure Heart Rate :

Actor: User, Pulse Sensor
Description:
This use case allows the user to measure their heart rate by placing a finger on the pulse sensor. The pulse sensor detects heartbeats and sends the signal to the ESP8266 microcontroller for processing.

Calculate BPM :
Actor: IoT Based Heart Rate Monitoring System
Description:
The system processes the signals received from the pulse sensor and calculates the heart rate in terms of Beats Per Minute (BPM). This

ensures accurate and meaningful heart rate measurement.

- **Display Heart Rate :**
- **Actor:** IoT Based Heart Rate Monitoring System
  Description:
  Once the BPM is calculated, the system displays the heart rate on the OLED display. This allows the user to view the result instantly on the device.

- **Upload Data to Cloud :**
- **Actor:** Cloud Platform (ThingSpeak)
  Description:
  The system uploads the heart rate data to the ThingSpeak cloud platform using WiFi connectivity. This enables remote monitoring and storage of heart rate data for future reference.

- **Provide Visual Feedback :**
- **Actor:** IoT Based Heart Rate Monitoring System
  Description:
  The system provides visual feedback through an LED indicator, which blinks whenever a heartbeat is detected. This confirms that the sensor is working properly.

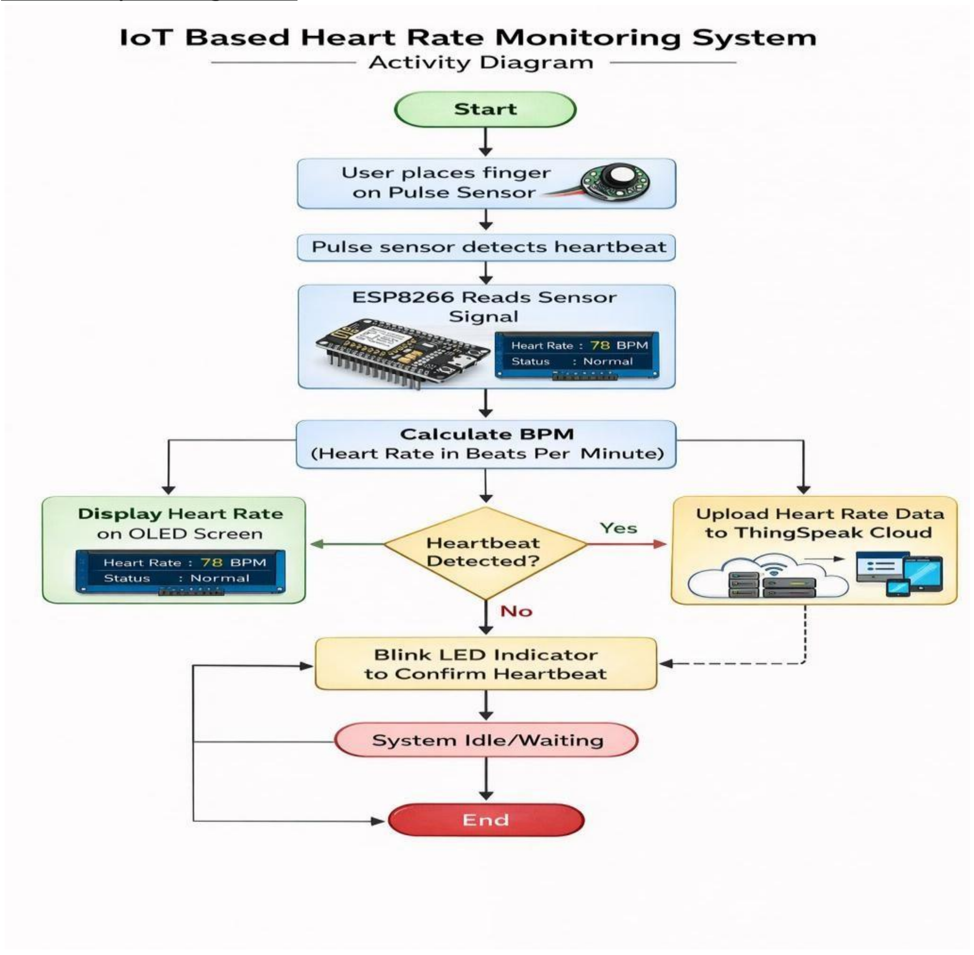
Purpose of Use Case Diagrams :

- **Visualizing System Interactions :**
- Use Case Diagrams provide a clear visual representation of how the user interacts with the heart rate monitoring system, making system functionality easy to understand.
- **Defining System Requirements :**

- They help identify and define the functional requirements of the system, ensuring that all features such as measurement, display, and cloud upload are included.
- **Facilitating Communication :**

  Use Case Diagrams act as an effective communication tool between students, teachers, and developers by presenting system functionality in a simple visual form.
- **Guiding System Design :**
- These diagrams help guide system design by highlighting important functionalities that need to be implemented in the project.
- **Documenting Functional Requirements :**
- They serve as documentation for the functional behavior of the system, which is useful for future upgrades or maintenance.
- **Supporting Test Case Development :**
- Each use case can be converted into test cases during testing to verify that the system performs correctly.
- **Identifying User Needs :**
- By focusing on user interaction, Use Case Diagrams help understand user needs and expectations for effective heart rate monitoring.
- **Simplifying Complex Systems :**
- For IoT-based systems, Use Case Diagrams simplify complexity by breaking the system into manageable and understandable interactions.

Activity Diagram :


IoT Based Heart Rate Monitoring System
Activity Diagram
Start
User places finger on Pulse Sensor
Pulse sensor detects heartbeat
ESP8266 Reads Sensor Signal
Heart Rate : 78 BPM
Status : Normal
Calculate BPM (Heart Rate in Beats Per Minute)
Display Heart Rate on OLED Screen
Heart Rate : 78 BPM
Status : Normal
Heartbeat Detected?
Yes
No
Upload Heart Rate Data to ThingSpeak Cloud
Blink LED Indicator to Confirm Heartbeat
System Idle/Waiting
End

Description:

Activity diagrams are a type of UML (Unified Modeling Language) diagram that represent the flow of activities within a system. They are useful for showing how different actions are performed step by step. In the **IoT Based Heart Rate Monitoring System**, the activity diagram explains the complete workflow starting from sensor initialization to displaying and uploading the heart rate data. It helps in understanding how the system behaves during real-time operation.

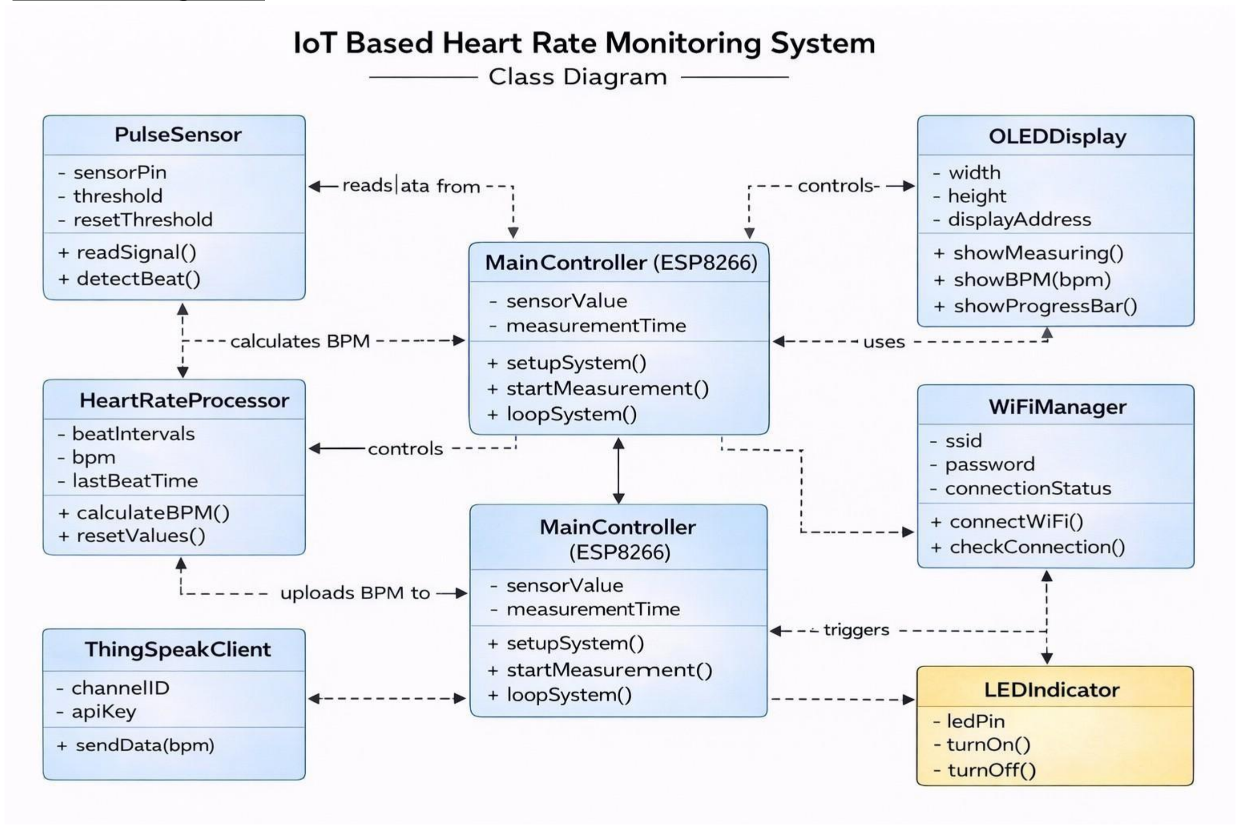
Purpose of Activity Diagrams :

Visualize Workflow :
Activity diagrams provide a clear visual representation of the workflow of the

heart rate monitoring system. They show the sequence of activities such as system start, heartbeat detection, BPM calculation, display output, and data upload to the cloud, making the process easy to understand.

- **Modeling Processes :**
- These diagrams help in modeling both high-level and detailed processes involved in heart rate monitoring. They describe how the system reads sensor data, processes it, and generates meaningful output, which is useful during system design and development.
- **Identify Parallel Activities :**
- Activity diagrams can represent parallel activities such as displaying BPM on the OLED screen while simultaneously sending data to the ThingSpeak cloud platform. This helps in understanding concurrent operations within the system.
- **Clarifying Responsibilities :**
- The diagram clearly shows which component performs which task. For example, the pulse sensor detects heartbeats, the ESP8266 processes the data, and the OLED display shows the results. This clarity helps in proper system understanding and implementation.
- **Supporting Communication :**
- Activity diagrams act as an effective communication tool between students, instructors, and developers. They present the system workflow in a visual format that is easy to explain and understand, even for non-technical users.
- **Facilitating Documentation :**
- These diagrams serve as an important part of project documentation. They provide a reference for future improvements, maintenance, or explanation of the system's working.
- **Enhancing System Design :**
- During the design phase, activity diagrams help identify possible issues or inefficiencies in the workflow. This allows improvements to be made before actual implementation of the system.
- **Testing and Validation :**
- Activity diagrams assist in testing and validation by clearly showing expected system behavior. Test cases can be created based on each activity to ensure that the system works correctly and reliably.

Class Diagram :



Class Diagram : Description

Class diagrams are a fundamental part of object-oriented design, providing a visual representation of a system’s classes and their relationships. They serve as a blueprint for the structure of a software application, illustrating how different classes interact with each other.

In the context of the **IoT Based Heart Rate Monitoring System**, class diagrams play an important role in organizing and defining the system components such as the microcontroller, pulse sensor, display module, and cloud platform. The diagram helps in understanding how data flows from the sensor to the display and cloud service, ensuring clarity among developers and users.

Purpose of Class Diagrams :

Modeling System Structure :

The class diagram visualizes the main components of the heart rate monitoring system, including the ESP8266 microcontroller, pulse sensor, OLED display, and ThingSpeak cloud service. It provides a clear structural overview of the system.

Understanding Relationships :

It shows how different classes such as Sensor, Display, WiFiModule, and CloudService interact with each other. This helps in understanding dependencies and data communication within the system.

Design Blueprint :

The class diagram acts as a blueprint for implementing the system logic in code. It defines how sensor data is processed, displayed on the OLED screen, and uploaded to the cloud platform.

Documentation :

It serves as an important reference for future maintenance, debugging, and enhancement of the system. Developers can easily understand the system architecture and extend functionality when required.

Sequence Diagram :

```
 User       Pulse Sensor       ESP8266(NodeMCU)       OLED Display
ThingSpeak Cloud
  |          |            |            |            |
  |----Power ON-->|            |            |            |
  |          |            |            |            |
  |          |----Heartbeat Data-->|            |            |
  |          |            |            |            |
  |          |            |--Calculate BPM---- >|            |
  |          |            |            |            |
  |          |            |--Display BPM ------>|            |
  |          |            |            |            |
  |          |            |--Send BPM via WiFi-------------------- >|
  |          |            |            |            |
  |          |            |<--Upload Success------------------------|
  |          |            |            |            |
  |          |----Next Reading---->|            |            |
```

Description :

A Sequence Diagram is a type of interaction diagram in Unified Modeling Language (UML) that illustrates how different components of a system interact with each other in a specific sequence over time. It focuses on the order in which messages or data are exchanged between objects to complete a particular process.

In the **IoT Based Heart Rate Monitoring System**, the sequence diagram helps in understanding how heart rate data flows from the pulse sensor to the microcontroller, then to the display and cloud platform, showing the dynamic behavior of the system.

Purpose of Sequence Diagrams :

Visual Representation of Interactions :

Sequence diagrams provide a clear visual representation of interactions between system components such as the pulse sensor, ESP8266 microcontroller, OLED display, and ThingSpeak cloud server in a time-based sequence.

Clarification of Message Flow :

They help clarify how heart rate data is sensed, processed, displayed, and uploaded to the cloud. This makes it easier to understand the flow of control during heart rate measurement and data transmission.

Facilitation of Communication :

The diagram acts as a communication tool among developers and students, helping them discuss and understand the system workflow and data exchange clearly.

Guidance for Development :

It serves as a guide for implementing interactions between hardware and software components. Developers can follow the sequence to program sensor reading, BPM calculation, display output, and cloud upload correctly.

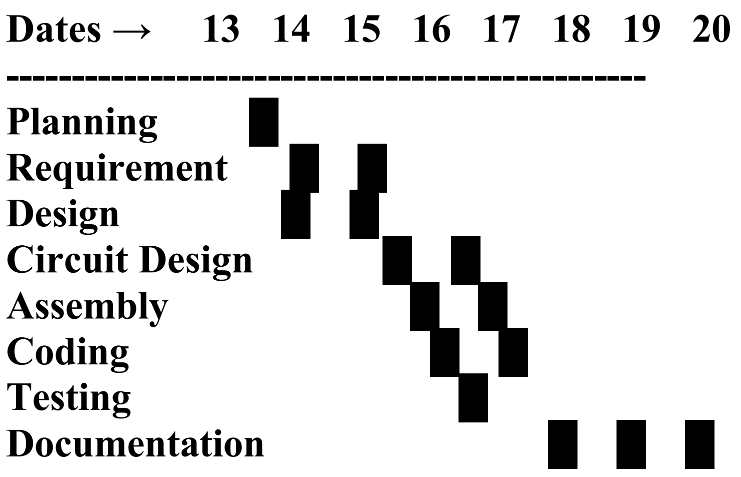
Documentation :

Sequence diagrams provide clear documentation of system operation, making future maintenance, debugging, and enhancements easier.

Identification of Issues :

By visualizing the interaction flow, sequence diagrams help identify delays, incorrect data transfer, or communication issues between components, improving system reliability.

Gantt Chart :



outlines the tasks involved in a project along a timeline, showing when each task starts, how long it will take, and when it should be completed. Gantt charts are widely used in project management to plan, coordinate, and track specific tasks or activities, providing a clear overview of the project's progress and deadlines.

1. The above Gantt chart provides a visual representation of the timeline for the **IoT Based Heart Rate Monitoring System** project. The project tasks are scheduled between **13 February 2026 and 20 February 2026**. The chart breaks the project into key phases such as planning, design, hardware assembly, coding, testing, and documentation, clearly displaying each phase's start and end dates.

2. This representation helps in understanding how the project was completed within a short time frame while ensuring proper planning and timely execution of all activities.

3.

Advantages of Using Gantt Charts :

Clear Visualization :

Gantt charts provide a clear and simple visual representation of the project timeline, making it easy to understand task sequences and deadlines.

Effective Planning :

They help in organizing tasks efficiently by assigning time durations to each activity, ensuring systematic project execution.

Tracking Progress :
Gantt charts allow continuous monitoring of project progress, helping identify completed, ongoing, and pending tasks.

Collaboration :

They improve coordination among team members by clearly defining responsibilities and timelines for each phase of the project.

outlines the tasks involved in a project along a timeline, showing when each task starts, how long it will take, and when it should be completed. Gantt charts are widely used in project management to plan, coordinate, and track specific tasks or activities, providing a clear overview of the project's progress and deadlines.

4. The above Gantt chart provides a visual representation of the timeline for the **IoT Based Heart Rate Monitoring System** project. The project tasks are scheduled between **13 February 2026 and 20 February 2026**. The chart breaks the project into key phases such as planning, design, hardware assembly, coding, testing, and documentation, clearly displaying each phase's start and end dates.

5. This representation helps in understanding how the project was completed within a short time frame while ensuring proper planning and timely execution of all activities.

outlines the tasks involved in a project along a timeline, showing when each task starts, how long it will take, and when it should be completed. Gantt charts are widely used in project management to plan, coordinate, and track specific tasks or activities, providing a clear overview of the project's progress and deadlines.

6. The above Gantt chart provides a visual representation of the timeline for the **IoT Based Heart Rate Monitoring System** project. The project tasks are scheduled between **13 February 2026 and 20 February 2026**. The chart breaks the project into key phases such as planning, design, hardware assembly, coding, testing, and documentation, clearly displaying each phase's start and end dates.

7. This representation helps in understanding how the project was completed within a short time frame while ensuring proper planning and timely execution of all activities.

Advantages of Using Gantt Charts :

Clear Visualization :

Gantt charts provide a clear and simple visual representation of the project timeline, making it easy to understand task sequences and deadlines.

Effective Planning :

They help in organizing tasks efficiently by assigning time durations to each activity, ensuring systematic project execution.

Tracking Progress :

Gantt charts allow continuous monitoring of project progress, helping identify completed, ongoing, and pending tasks.

Collaboration :

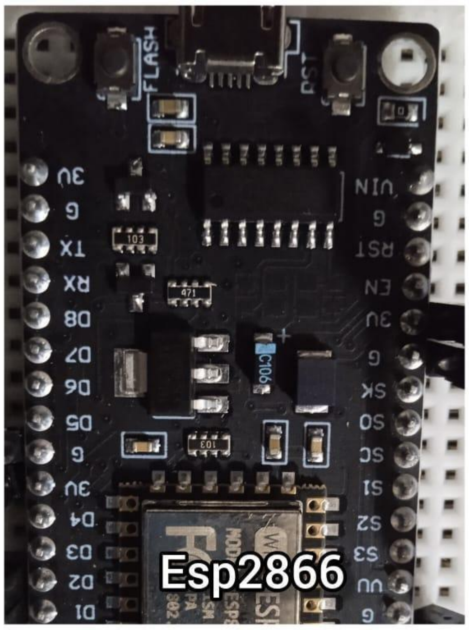
They improve coordination among team members by clearly defining responsibilities and timelines for each phase of the project.

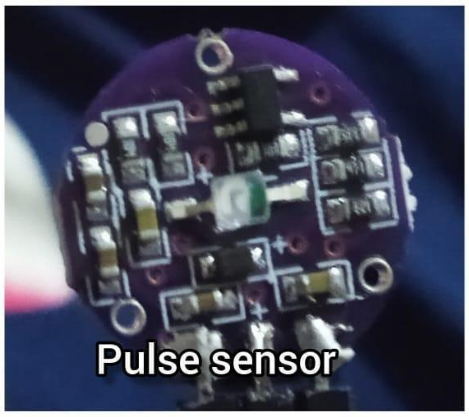
SYSTEM IMPLEMENTATION

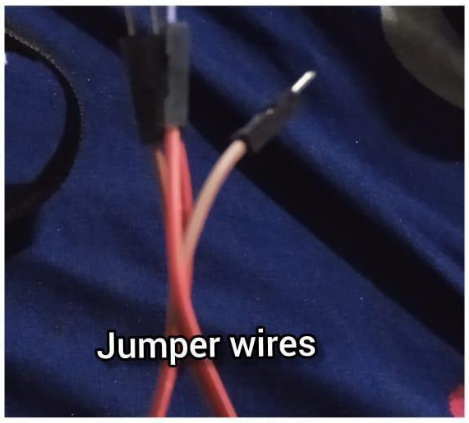
Step-by-Step Assembly:

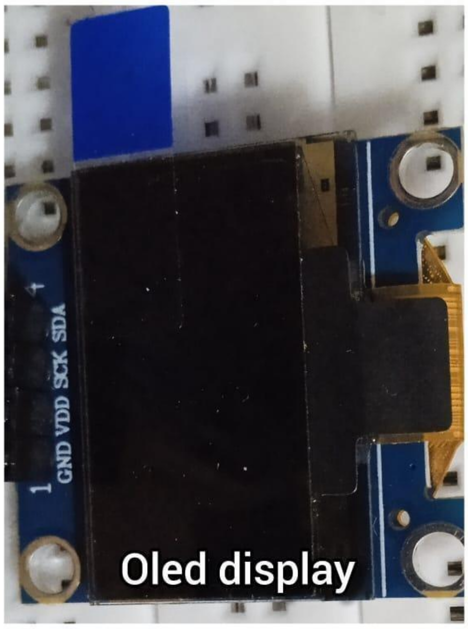
Step 1 :

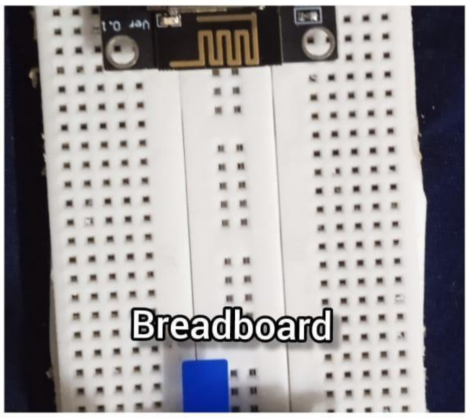
Firstly, identify these components.





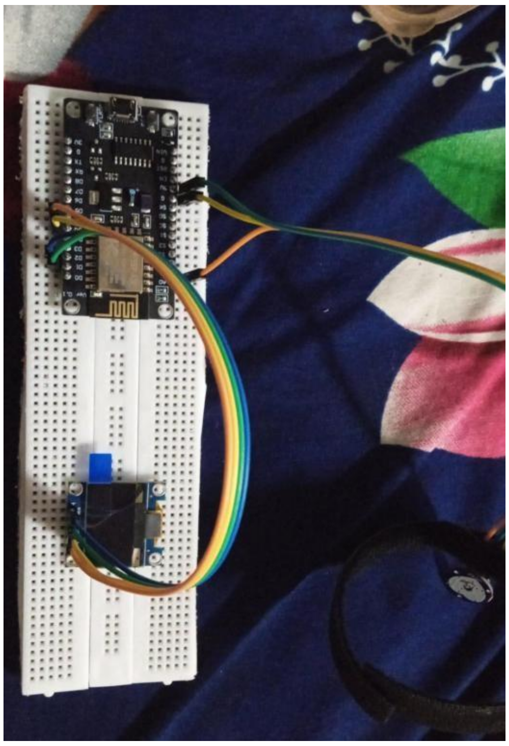




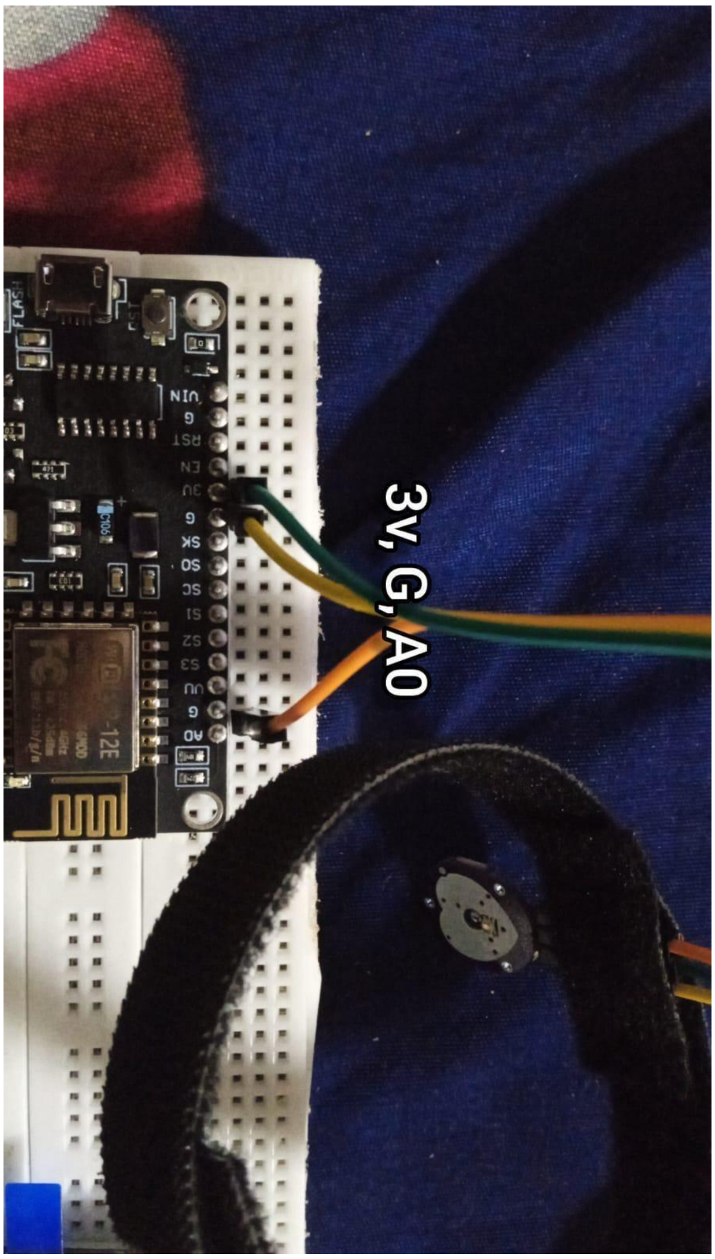


Step 2 :

Place all components on Breadboard :

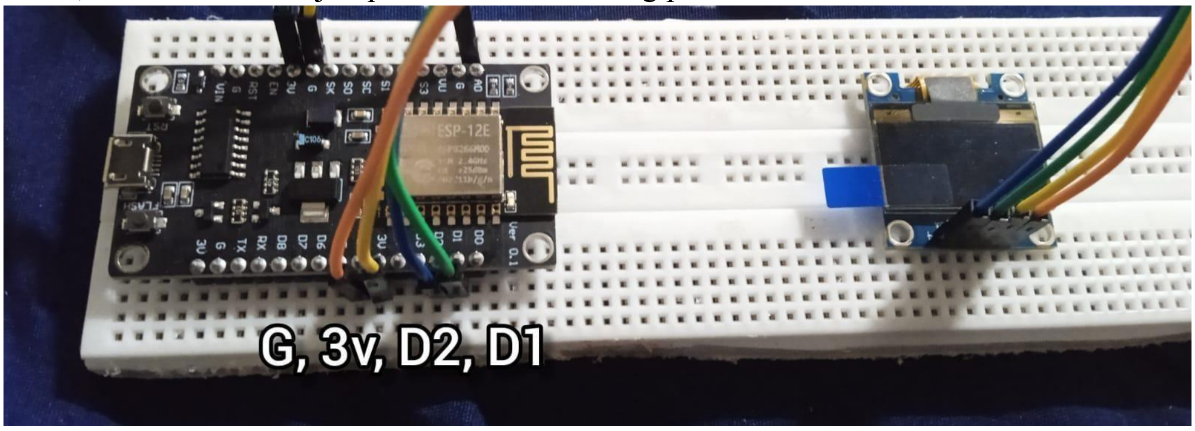


Step 3 : Connect Pulse sensor jumper wires on following pins.



Step 4 :

Then, connect the Oled jumper wires on following pins



Step 5 :

So, let’s create the program for this project. This program includes all three functions. We can run these separately. It is as follows.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <ESP8266WiFi.h>
#include <ThingSpeak.h>

Adafruit_SSD1306 display(128, 64, &Wire);

// -------- WiFi --------
const char* ssid = "ESP_TEST";
const char* password = "123456789";

// -------- ThingSpeak --------
WiFiClient client;
unsigned long channelID = 3263806;
const char* writeAPIKey = "76P4WTT1W9CR9Y9K";

// -------- Sensor --------
const int sensorPin = A0;
const int ledPin = D8;

// -------- Thresholds --------
int threshold = 580;
int resetThreshold = 550;

// -------- Timing --------
unsigned long lastSampleTime = 0;
unsigned long measureStartTime = 0;
unsigned long lastBeatTime = 0;
unsigned long restartTimer = 0;

const unsigned long sampleInterval = 50;
const unsigned long measureDuration = 20000;   // 20 sec
const unsigned long restartDelay = 30000;      //  30 seconds
const unsigned long minBeatInterval = 300;

// -------- Variables --------
int sensorValue;
bool pulseDetected = false;
bool measurementDone = false;

int beatIntervals[5];
int beatIndex = 0;
int bpm = 0;
```

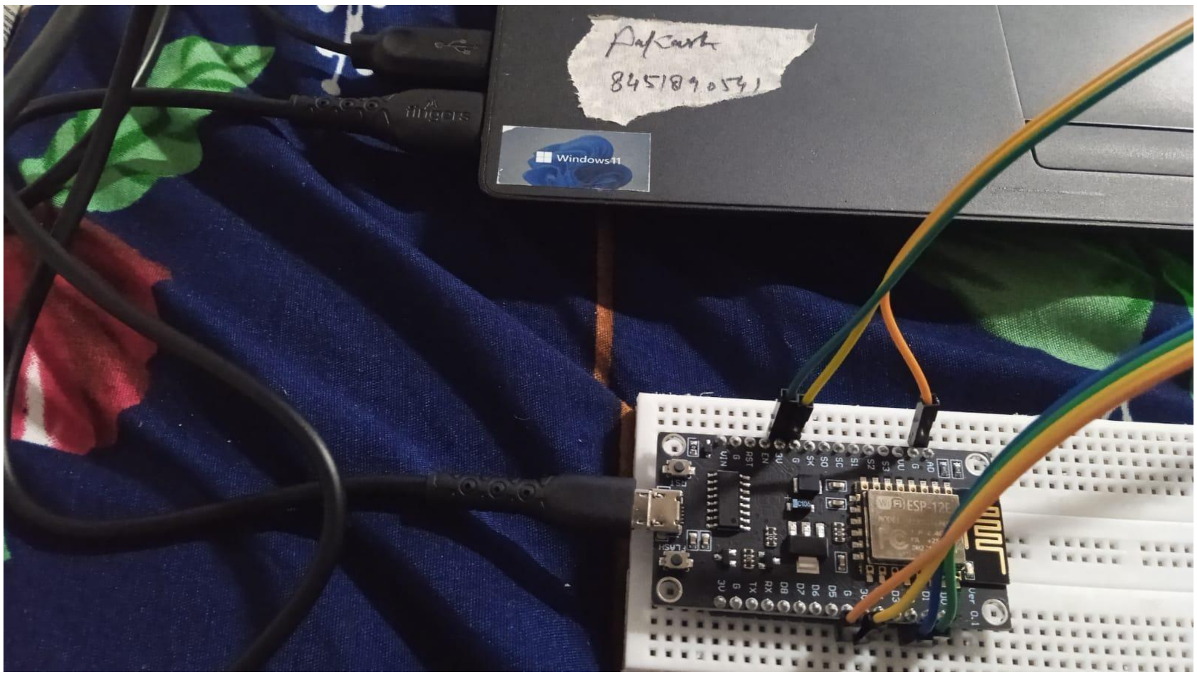
```
void setup() {
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);

  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.clearDisplay();
  display.setTextColor(WHITE);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) delay(300);

  ThingSpeak.begin(client);
  startMeasurement();
}

void startMeasurement() {
  measureStartTime = millis();
  lastBeatTime = 0;
  beatIndex = 0;
  bpm = 0;
  measurementDone = false;

  display.clearDisplay();
  display.setTextSize(2);
  display.setCursor(0, 20);
  display.println("Measuring");
  display.display();
}

void loop() {
  unsigned long currentMillis = millis();

  // -------- AUTO RESTART AFTER 30 SECONDS --------
  if (measurementDone && (currentMillis - restartTimer >= restartDelay)) {
    startMeasurement();
  }

  if (measurementDone) return;

  // -------- SENSOR SAMPLING --------
  if (currentMillis - lastSampleTime >= sampleInterval) {
    lastSampleTime = currentMillis;
    sensorValue = analogRead(sensorPin);

    if (sensorValue > threshold && !pulseDetected &&
        (currentMillis - lastBeatTime > minBeatInterval)) {

      pulseDetected = true;
      digitalWrite(ledPin, HIGH);

      if (lastBeatTime > 0 && beatIndex < 5) {
```

```
      beatIntervals[beatIndex++] = currentMillis - lastBeatTime;
     }
     lastBeatTime = currentMillis;
    }

    if (sensorValue < resetThreshold) {
     pulseDetected = false;
     digitalWrite(ledPin, LOW);
    }
   }

  // -------- PROGRESS BAR --------
  int progress = map(currentMillis - measureStartTime, 0, measureDuration, 0, 128);
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.println("Measuring...");
  display.drawRect(0, 20, 128, 10, WHITE);
  display.fillRect(0, 20, progress, 10, WHITE);
  display.display();

  // -------- BPM CALCULATION --------
  if (currentMillis - measureStartTime >= measureDuration) {

    long sum = 0;
    for (int i = 0; i < beatIndex; i++) sum += beatIntervals[i];

    if (beatIndex > 0) {
     long avgInterval = sum / beatIndex;
     bpm = 60000 / avgInterval;
    }

    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(0, 0);
    display.println("Heart Rate");
    display.setCursor(0, 30);
    display.print("BPM: ");
    display.print(bpm);
    display.display();

    ThingSpeak.writeField(channelID, 1, bpm, writeAPIKey);

    Serial.print("Stable BPM: ");
    Serial.println(bpm);

    measurementDone = true;
    restartTimer = currentMillis;
  }
}
```

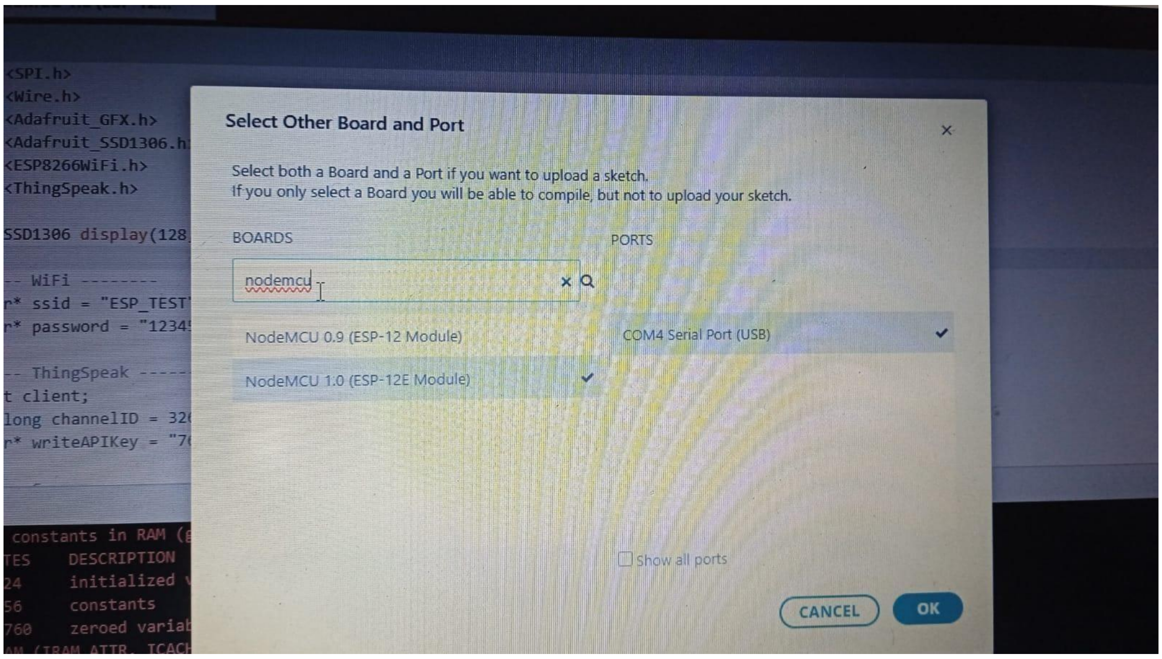
SYSTEM TESTING AND RESULT

Step 6 :

Connect Datacable from Esp8266 to laptop for code uploading.



Step 7: Open Arduino IDE,and select Board and Ports,after that type the code on Arduino IDE



```
<SPI.h>
<Wire.h>
<Adafruit_GFX.h>
<Adafruit_SSD1306.h
<ESP8266WiFi.h>
<ThingSpeak.h>

SSD1306 display(128

-- WiFi --------
r* ssid = "ESP_TEST
r* password = "1234

-- ThingSpeak -----
t client;
long channelID = 32
r* writeAPIKey = "7
```

Select Other Board and Port

Select both a Board and a Port if you want to upload a sketch.
If you only select a Board you will be able to compile, but not to upload your sketch.

BOARDS: nodemcu

- NodeMCU 0.9 (ESP-12 Module)
- NodeMCU 1.0 (ESP-12E Module) ✓

PORTS

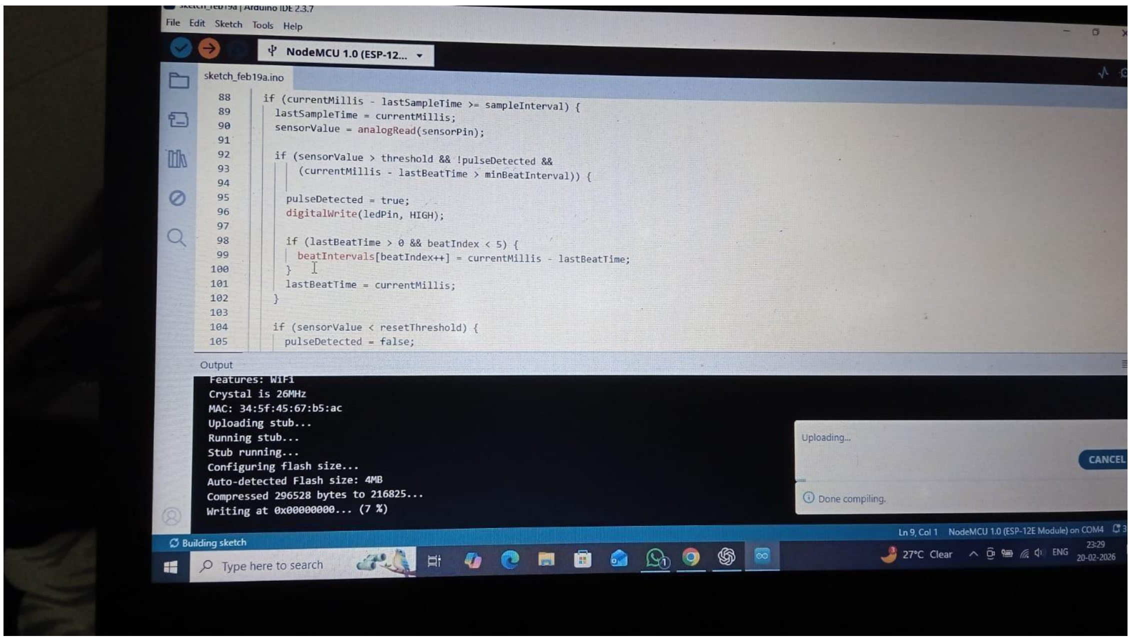
- COM4 Serial Port (USB) ✓

☐ Show all ports

CANCEL OK

```
constants in RAM (
TES    DESCRIPTION
24     initialized
56     constants
760    zeroed varia
```

Step 8: Upload the code



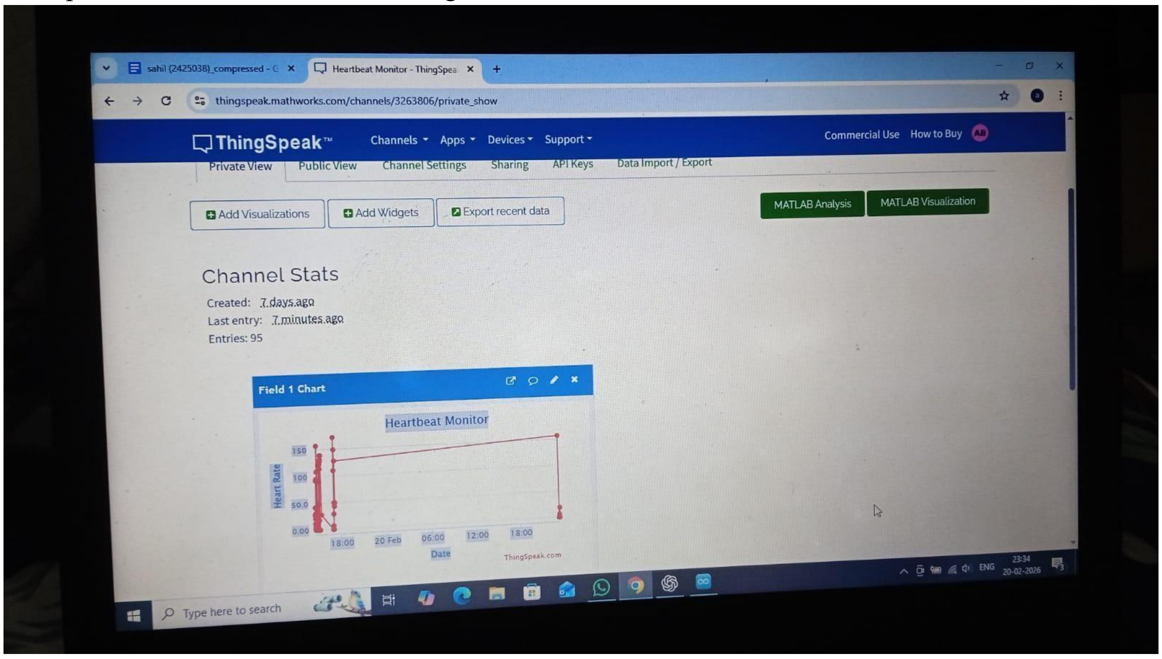
```
  if (currentMillis - lastSampleTime >= sampleInterval) {
    lastSampleTime = currentMillis;
    sensorValue = analogRead(sensorPin);

    if (sensorValue > threshold && !pulseDetected &&
        (currentMillis - lastBeatTime > minBeatInterval)) {

      pulseDetected = true;
      digitalWrite(ledPin, HIGH);

      if (lastBeatTime > 0 && beatIndex < 5) {
        beatIntervals[beatIndex++] = currentMillis - lastBeatTime;
      }
      lastBeatTime = currentMillis;
    }

    if (sensorValue < resetThreshold) {
      pulseDetected = false;
```

```
Features: WiFi
Crystal is 26MHz
MAC: 34:5f:45:67:b5:ac
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 296528 bytes to 216825...
Writing at 0x00000000... (7 %)
```

sketch_feb19a | Arduino IDE 2.3.7
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12...
sketch_feb19a.ino
Output
Uploading...
CANCEL
Done compiling.
Ln 9, Col 1 NodeMCU 1.0 (ESP-12E Module) on COM4
Building sketch

Step 9:

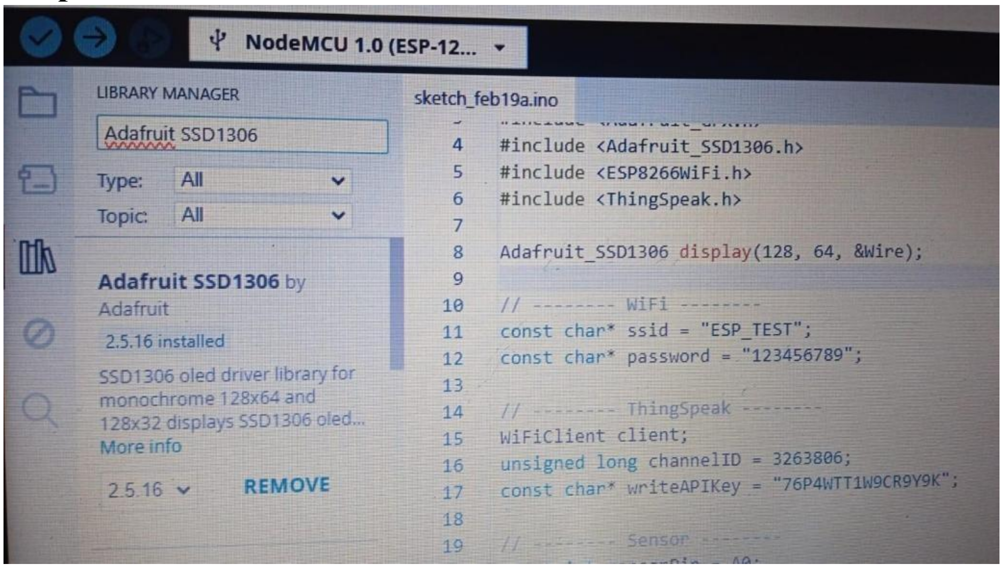
Create an channel on THINKSPEAK ,ThingSpeak is an **IoT cloud platform** used to collect, store, and visualize sensor data in real time. In the **IoT Based Heart Rate Monitoring System**, ThingSpeak plays an important role in remote monitoring.



Channel Stats
Created: 7 days ago
Last entry: 7 minutes ago
Entries: 95

Heartbeat Monitor - ThingSpeak
thingspeak.mathworks.com/channels/3263806/private_show
ThingSpeak
Channels Apps Devices Support
Commercial Use How to Buy
Private View Public View Channel Settings Sharing API Keys Data Import / Export
Add Visualizations
Add Widgets
Export recent data
MATLAB Analysis
MATLAB Visualization
Field 1 Chart
Heartbeat Monitor
Heart Rate
150
100
50.0
0.00
18:00
20 Feb
06:00
12:00
18:00
Date
ThingSpeak.com

Step 9:Install Libraies in Arduino IDE



Adafruit SSD1306

Type: All

Topic: All

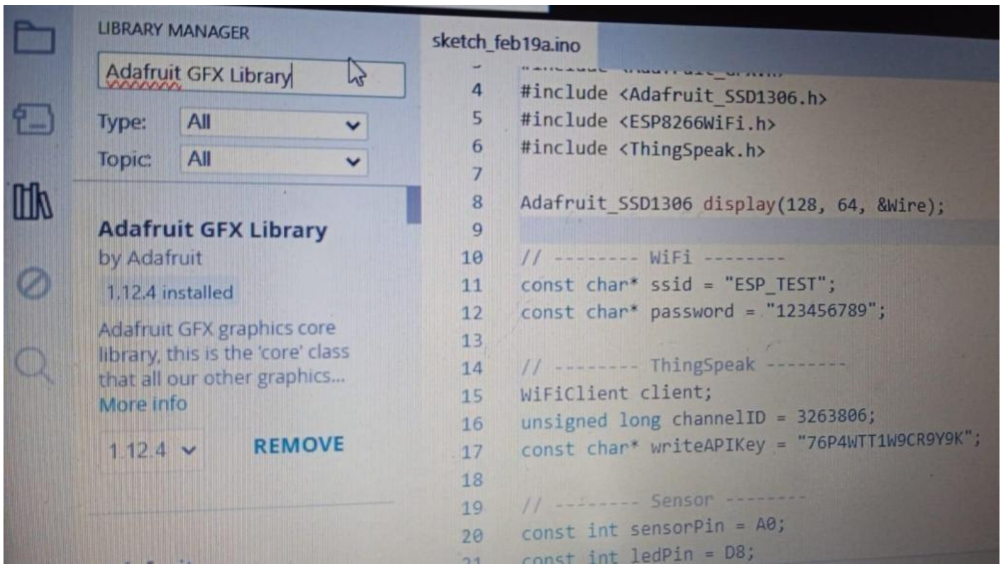
Adafruit SSD1306 by Adafruit

2.5.16 installed

SSD1306 oled driver library for monochrome 128x64 and 128x32 displays SSD1306 oled...
More info

2.5.16 REMOVE

```
#include <Adafruit_SSD1306.h>
#include <ESP8266WiFi.h>
#include <ThingSpeak.h>

Adafruit_SSD1306 display(128, 64, &Wire);

// -------- WiFi --------
const char* ssid = "ESP_TEST";
const char* password = "123456789";

// -------- ThingSpeak --------
WiFiClient client;
unsigned long channelID = 3263806;
const char* writeAPIKey = "76P4WTT1W9CR9Y9K";

// -------- Sensor --------
```



Adafruit GFX Library

Type: All

Topic: All

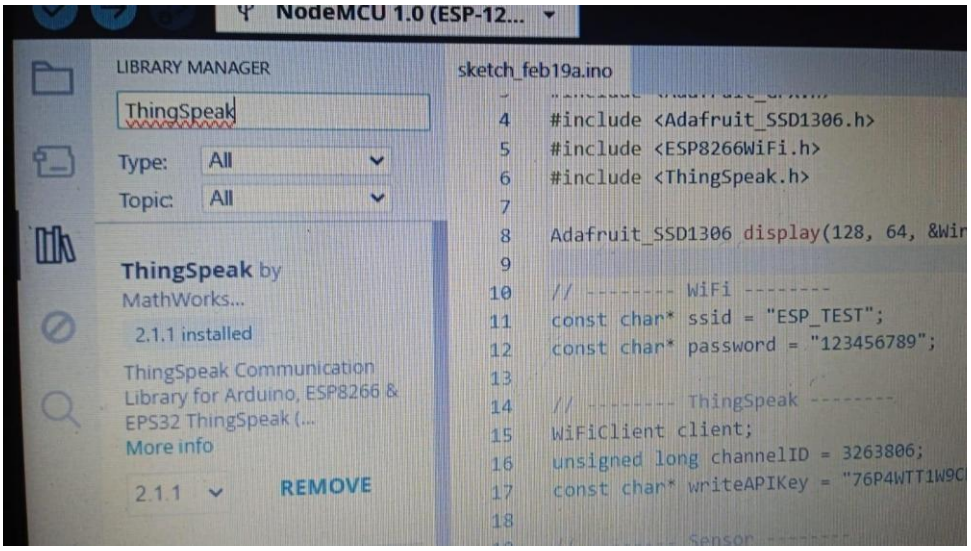
Adafruit GFX Library by Adafruit

1.12.4 installed

Adafruit GFX graphics core library, this is the 'core' class that all our other graphics...
More info

1.12.4 REMOVE

```
#include <Adafruit_SSD1306.h>
#include <ESP8266WiFi.h>
#include <ThingSpeak.h>

Adafruit_SSD1306 display(128, 64, &Wire);

// -------- WiFi --------
const char* ssid = "ESP_TEST";
const char* password = "123456789";

// -------- ThingSpeak --------
WiFiClient client;
unsigned long channelID = 3263806;
const char* writeAPIKey = "76P4WTT1W9CR9Y9K";

// -------- Sensor --------
const int sensorPin = A0;
const int ledPin = D8;
```



ThingSpeak

Type: All

Topic: All

ThingSpeak by MathWorks...

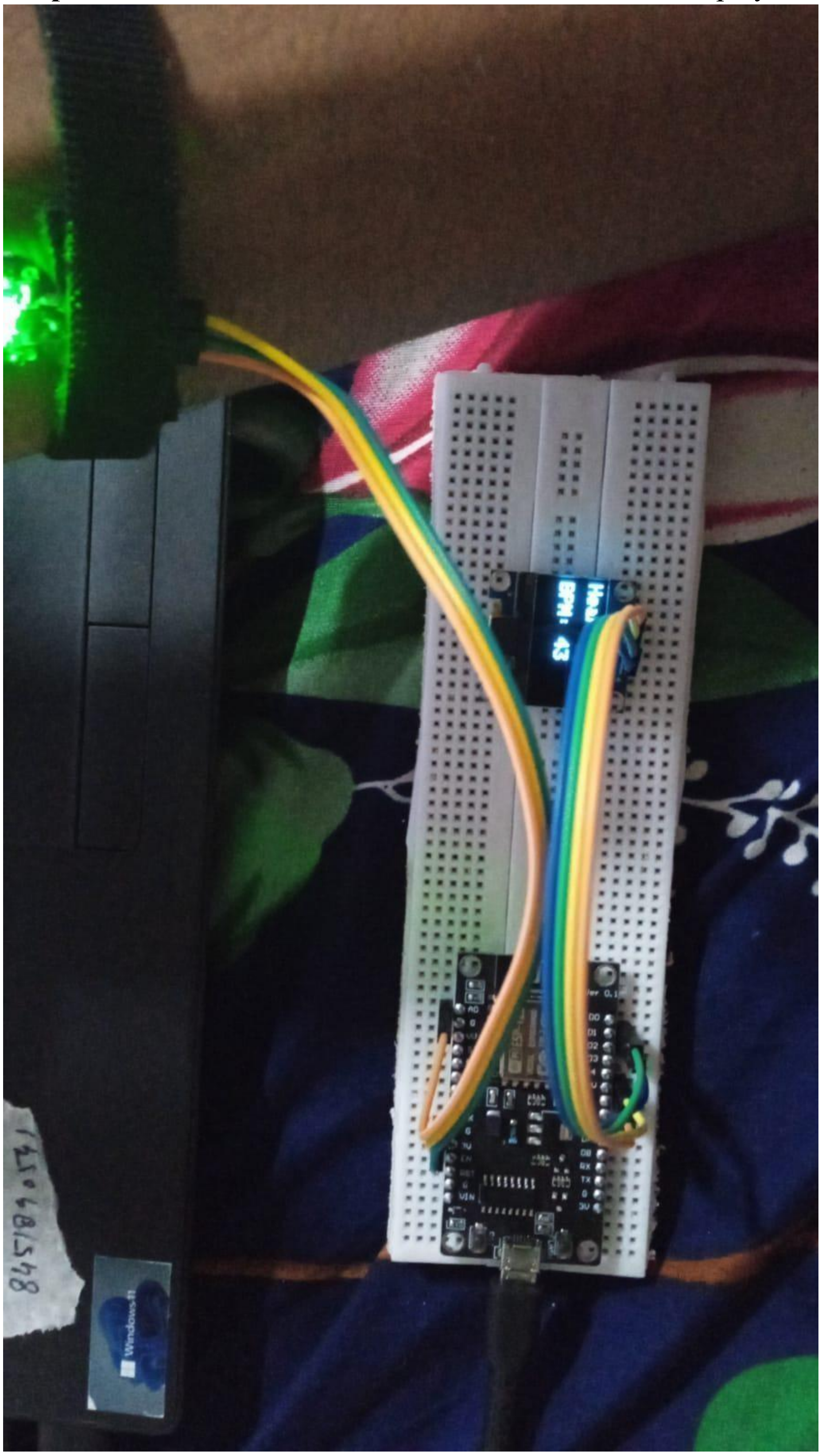
2.1.1 installed

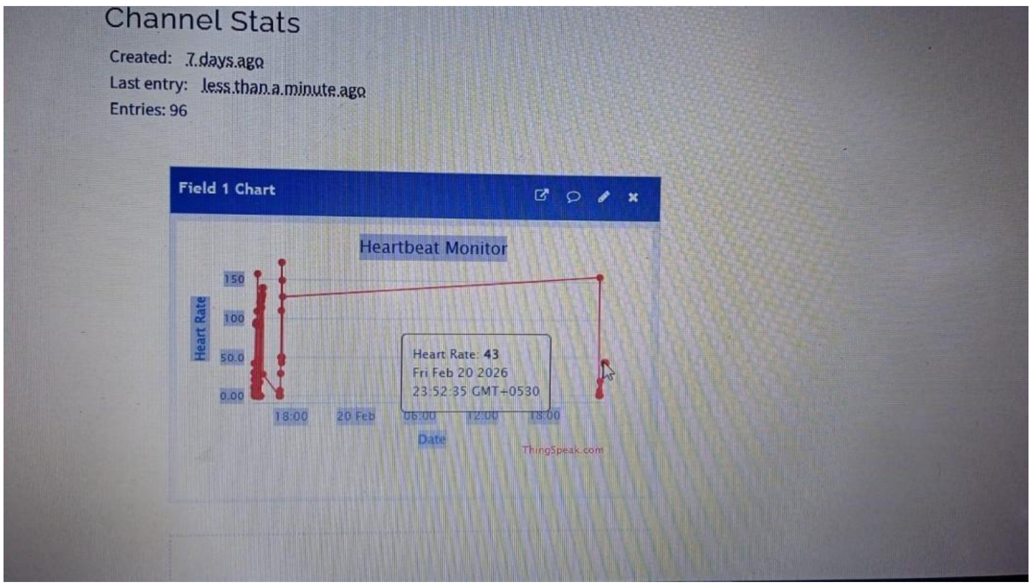
ThingSpeak Communication Library for Arduino, ESP8266 & EPS32 ThingSpeak (...
More info

2.1.1 REMOVE

```
#include <Adafruit_SSD1306.h>
#include <ESP8266WiFi.h>
#include <ThingSpeak.h>

Adafruit_SSD1306 display(128, 64, &Wir

// -------- WiFi --------
const char* ssid = "ESP_TEST";
const char* password = "123456789";

// -------- ThingSpeak --------
WiFiClient client;
unsigned long channelID = 3263806;
const char* writeAPIKey = "76P4WTT1W9C

// -------- Sensor --------
```

Step 10: Now run the code and see the result on OLED display and THINKSPEAK





Channel Stats

Created: 7 days ago

Last entry: less than a minute ago

Entries: 96

FUTURE SCOPE AND CONCLUSION

Future Scope :

The **IoT Based Heart Rate Monitoring System** provides a strong base for future improvements and advanced health monitoring applications. Several enhancements can be implemented to improve accuracy, usability, and functionality.

Future improvements could include :

Advanced Sensors :
More accurate biomedical sensors such as SpO_2 sensors, temperature sensors, or ECG sensors can be added to monitor multiple health parameters along with heart rate.

Mobile Application Integration :
A dedicated mobile application can be developed to display real-time heart rate data, alerts, and historical records for better user experience.

AI and Data Analytics :
Machine learning algorithms can be used to analyze heart rate patterns and detect abnormalities such as irregular heartbeat or stress conditions.

Real-Time Alerts :
The system can be enhanced to send instant alerts via SMS, email, or app notifications in case of abnormal BPM values.

Wearable Design :
The system can be converted into a wearable device such as a smart band or wristwatch for continuous health monitoring.

Healthcare Integration :
The collected data can be shared directly with hospitals or doctors, enabling remote patient monitoring and telemedicine applications.

Conclusion :

The **IoT Based Heart Rate Monitoring System** successfully achieves its objective of measuring and monitoring heart rate in real time. The system detects heartbeats using a pulse sensor, displays BPM values on an OLED screen, and uploads the data to the ThingSpeak cloud platform for remote monitoring.

This project demonstrates the effective use of IoT technology in healthcare-related applications and provides valuable hands-on experience in embedded systems, sensors, and cloud-based data monitoring. It is especially useful for **students** to understand the working of biomedical sensors, IoT communication, and real-time data visualization.

It is important to note that this project is developed for educational, testing, and demonstration purposes only. The system is not intended for professional medical diagnosis or real-life clinical use, as it does not meet medical-grade accuracy and certification standards.

Overall, due to its simplicity, low cost, and expandability, this project serves as an excellent learning platform for students and beginners. With further improvements and medical-grade components, it can be extended into a more advanced health monitoring system.

REFERENCES

References :

1. **YouTube Reference** –
   Project reference and implementation guidance taken from a YouTube tutorial. URL: https://youtu.be/u-gq7wABn9Q?si=jdlq0EduSs7s5yBf
2. **Arduino IDE** –
   Software used for writing, compiling, and uploading the program to the ESP8266 board. URL: https://www.arduino.cc/en/software
3. **Pulse Sensor and Circuit Connections** –
   Information related to the pulse sensor and hardware connections was referred from online resources and Google search.
4. **Mentor Guidance** –
   Guidance and support were provided by **Sahil Sir** during the design, development, and testing phases of the project.
5. **Components Purchased** –
   ESP8266 (NodeMCU), pulse sensor, OLED display, LEDs, and connecting wires were purchased from local electronics stores.

GLOSSARY

Glossary :

ESP8266 (NodeMCU):
A low-cost WiFi-enabled microcontroller used as the main controller in this project. It reads sensor data, processes heart rate values, and uploads data to the cloud.

Pulse Sensor:
A biomedical sensor used to detect heartbeats by measuring changes in blood flow. It provides analog signals that are used to calculate heart rate in beats per minute (BPM).

OLED Display (SSD1306):
A small organic light-emitting diode display used to show real-time information such as measuring status and heart rate values.

ThingSpeak:
An IoT cloud platform used to store, visualize, and analyze heart rate data uploaded through the internet.

WiFi:
A wireless networking technology that allows the ESP8266 to connect to the internet and send data to the cloud platform.

LED Indicator:
A light-emitting diode used to indicate heartbeat detection by blinking whenever a pulse is detected.

Arduino IDE:
An Integrated Development Environment used to write, compile, and upload programs to the ESP8266 microcontroller.

Analog Pin (A0):
A pin on the ESP8266 used to read analog signals from the pulse sensor.

BPM (Beats Per Minute):
A unit used to measure heart rate, representing the number of heartbeats in one minute.

IoT (Internet of Things):
A technology that enables devices to connect to the internet and exchange data without human intervention.

Threshold Value:
A predefined sensor value used to detect valid heartbeats and avoid false readings.

Cloud Platform:
An online service used to store and access sensor data remotely, allowing real-time monitoring and analysis.

Additional Reference
6. GitHub Repository – Smart Heart Rate Monitor System
Source code and project documentation available at:
https://github.com/akashbalghare4321-art/Smart-heart-rate-monitor-system