BASIC ELECTRIC VEHICLE (EV) MONITORING AND OPERATION TESTING SYSTEM

“Enhancing Efficiency and Sustainability through Smart Data Analytics."

PROJECT CREATED BY: LINGESHWAR.G, KISHOREKUMAR.J, LAKSHMANAN.M, NARENDIRAN.V

PROJECT REVIEWED BY: PRABHAKARAN.S

PROJECT CREATED DATE: 25/MAY/2024

PROJECT CODE: IOT 004

COLLEGE CODE: 5115

TEAM NAME: IOT 1518

EXECUTIVE SUMMARY

THIS PROJECT INVOLVES CREATING A REAL-TIME ELECTRIC VEHICLE (EV) MONITORING SYSTEM USING AN ESP32 MICROCONTROLLER, DHT22 SENSOR, PIR SENSOR, POTENTIOMETER, AND BLYNK IOT PLATFORM. THE SYSTEM TRACKS TEMPERATURE, VIBRATION, AND BATTERY HEALTH, ALLOWING FOR REMOTE MONITORING VIA THE BLYNK APP. KEY FINDINGS INCLUDE SUCCESSFUL INTEGRATION OF SENSORS WITH ESP32, REAL-TIME DATA TRANSMISSION TO BLYNK, AND A FUNCTIONING CONTROL MECHANISM TO ENABLE OR DISABLE MONITORING.

TABLE OF CONTENT

CONTENT

1. Project Title..........................1
2. Executive Summary......................2
3. Table of Contents......................3
4. Project Objective......................4
5. Scope:.................................5
6. Methodology:...........................6
7. Artifacts used.........................8
8. Technical coverage:...................11
9. Results:..............................15
10. Challenges and Resolutions: 17
11. Conclusion:..........................19
12. References: 20

PROJECT OBJECTIVE

The objective of this project is to develop an EV monitoring system that collects and transmits real-time data on temperature, vibration, and battery health to a remote server using the Blynk IoT platform. The project addresses the need for an accessible and efficient monitoring solution for EVs.

SCOPE

The project scope includes:

- ➢ Setting up the ESP32 microcontroller with necessary sensors.
- ➢ Integrating the system with the Blynk platform.
- ➢ Developing code for data collection and transmission.
- ➢ Implementing a control mechanism to enable or disable monitoring.

Assumptions include stable Wi-Fi connectivity and proper functioning of sensors. The project is limited to the specified sensors and the Blynk platform.

METHODOLOGY

The project follows these steps:

Setup and Configuration: Configure ESP32 with Wi-Fi credentials and Blynk authentication.

Sensor Integration: Connect DHT22 sensor, PIR sensor, and potentiometer to ESP32.

Code Development: Write Arduino code to read sensor data and send it to Blynk.

Testing and Validation: Test the system to ensure accurate data transmission and control functionality.

Each step is justified to ensure a systematic approach to developing a reliable monitoring system.

Requirement Analysis: Identifying the requirements for IoT integration and remote monitoring.

System Design: Designing the architecture for integrating the EV Monitoring system with the Blynk web console.

Implementation: Updating the hardware and software components, configuring the Blynk dashboard, and programming communication protocols.

Testing and Validation: Conducting testing to ensure the reliability and functionality of the system.

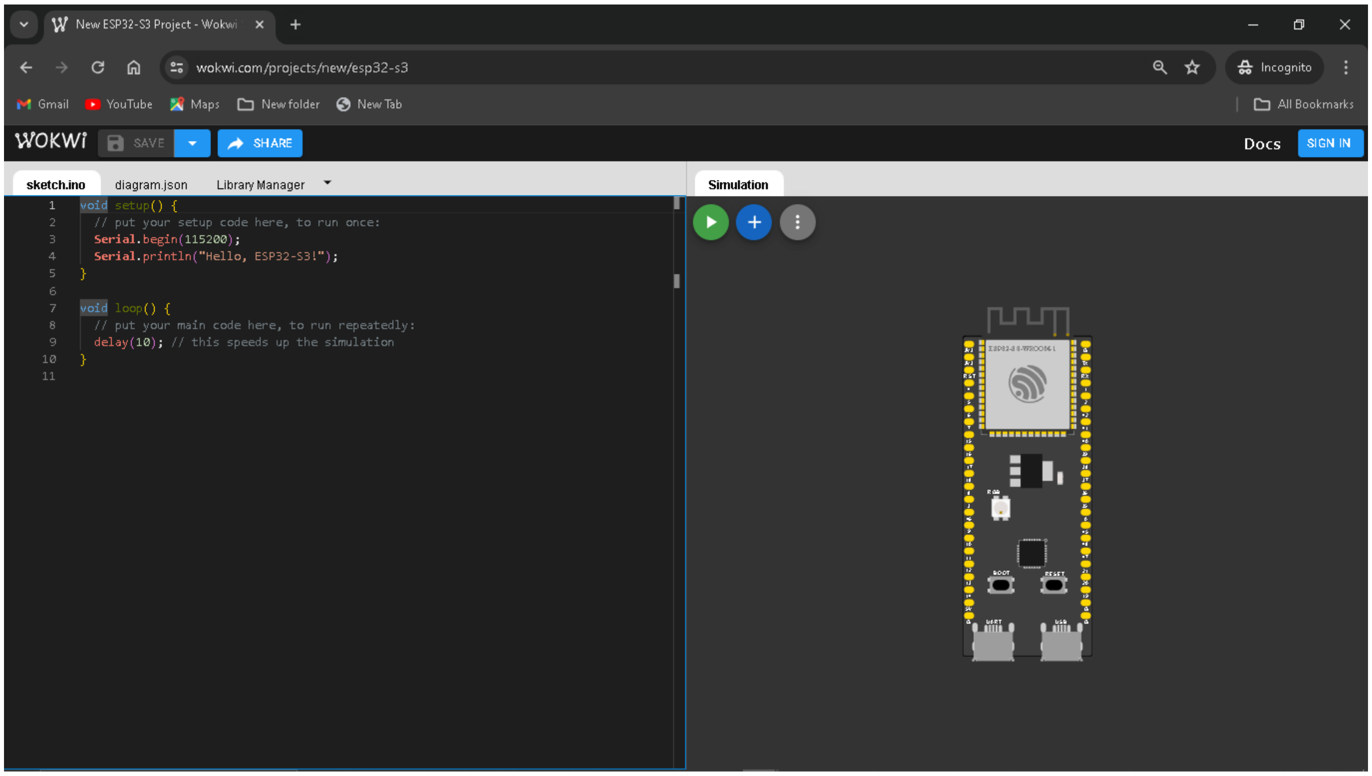
Deployment: Deploying the system and conducting real-world testing to verify performance.

ARTIFACT USED

The following artifacts were utilized throughout the project:

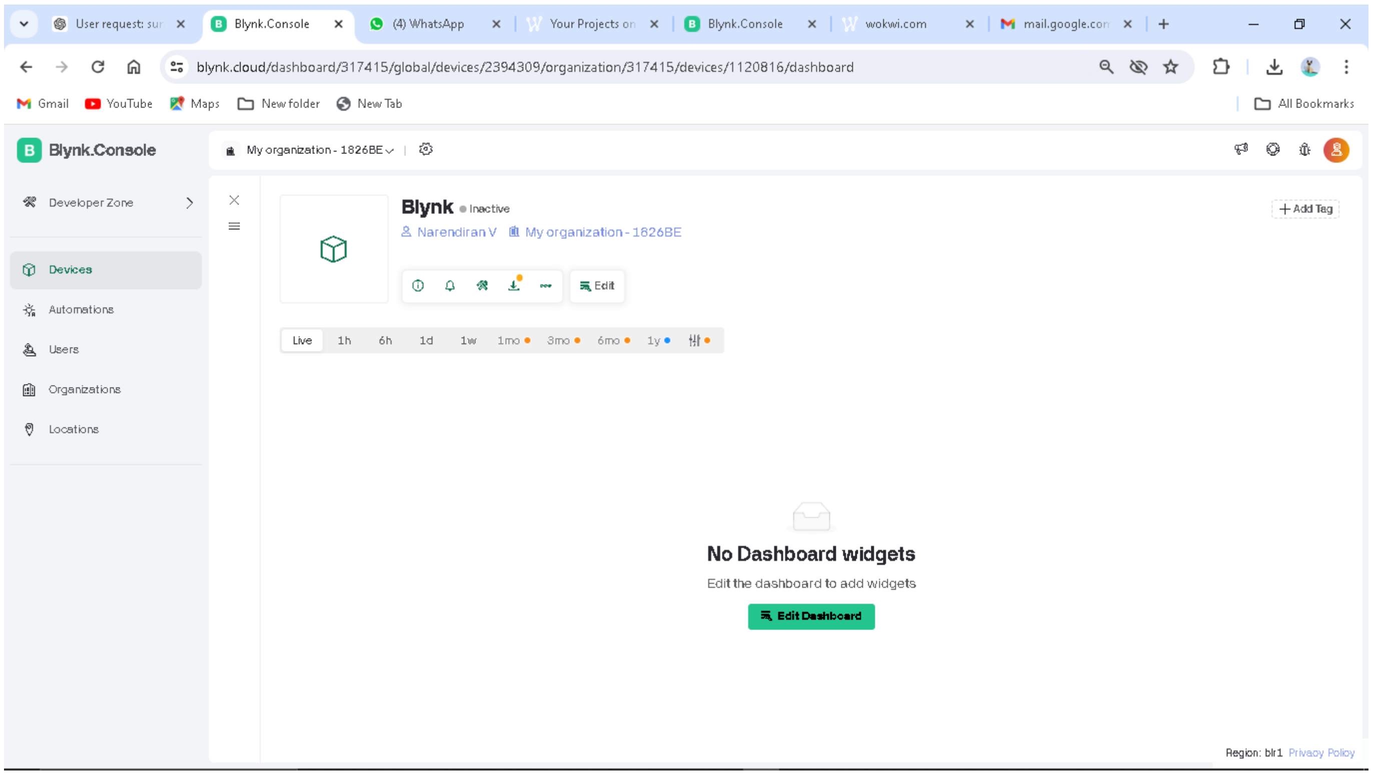
- **EV Data:** Real-time data on temperature, vibration, and battery health collected from various intersections.
- **Research Articles:** Explored literature on IoT-based monitoring systems and ESP32 capabilities.
- **Blynk Library:** Arduino library for interfacing with the Blynk platform and sending/receiving data.

➢ **Wokwi Online Simulator Tool:** Used for testing and debugging Arduino code.



```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println("Hello, ESP32-S3!");
}

void loop() {
  // put your main code here, to run repeatedly:
  delay(10); // this speeds up the simulation
}
```

- **Blynk IoT Platform:** Web console and mobile app for IoT device control and data visualization.


blynk.cloud/dashboard/317415/global/devices/2394309/organization/317415/devices/1120816/dashboard

No Dashboard widgets

Edit the dashboard to add widgets

- **ESP32 Wi-Fi Module:** Hardware platform for enabling Wi-Fi connectivity and IoT capabilities.

TECHNICAL COVERAGE

The EV Monitoring and Operation Testing System provides the following technical functionalities:

Adaptive Signal Control: Dynamic adjustment of EV signal timings based on its flow.

Remote Monitoring: Real-time monitoring of EV and system status via the Blynk dashboard.

Remote Control: Ability to remotely adjust configurations using the Blynk web or mobile interface.

Data Logging: Logging of system performance metrics for analysis and optimization.

8.1 Function Description

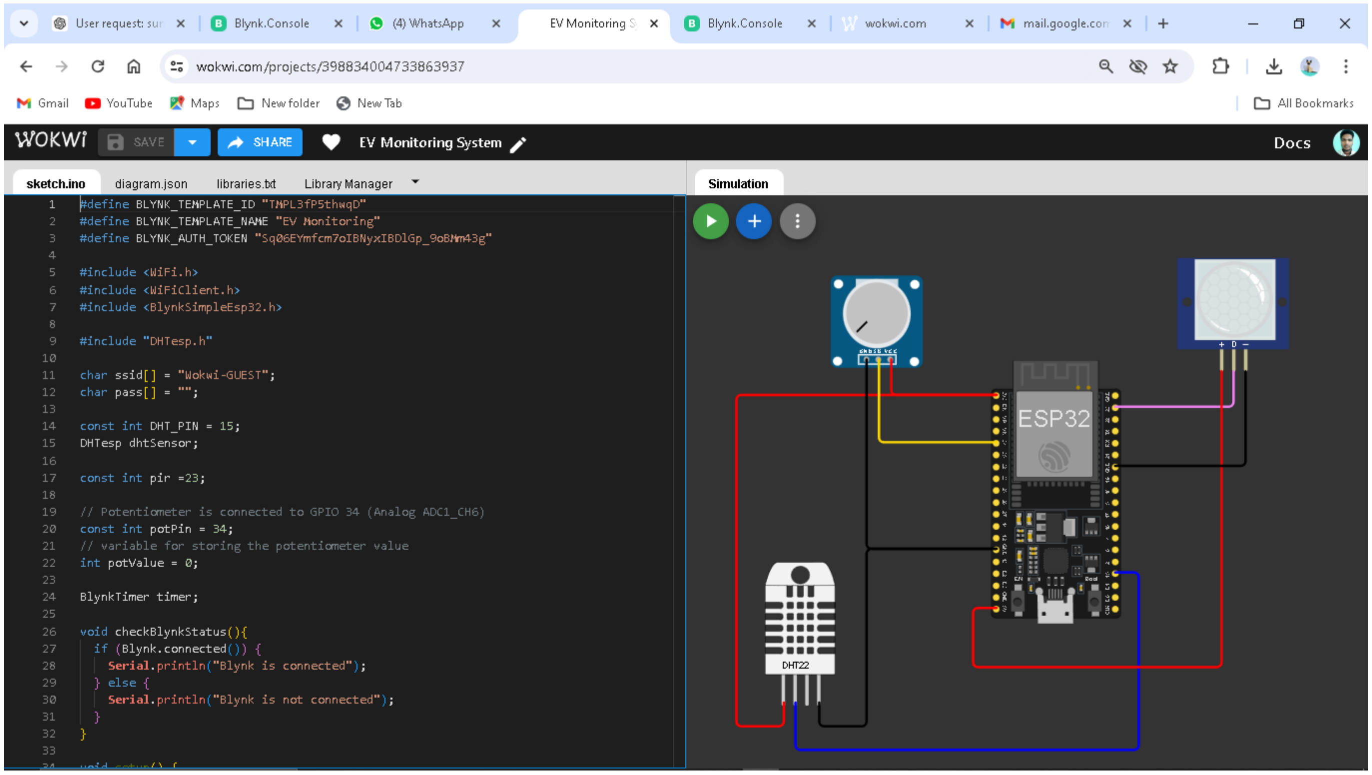
checkBlynkStatus (): Checks and prints the Blynk connection status every 5 seconds.

Setup (): Initializes serial communication, connects to Wi-Fi and Blynk, sets up the DHT sensor and PIR sensor, and schedules periodic data sending.

Send Data (): Reads temperature from the DHT sensor, vibration from the PIR sensor, and battery health (via a potentiometer), then sends this data to Blynk virtual pins.

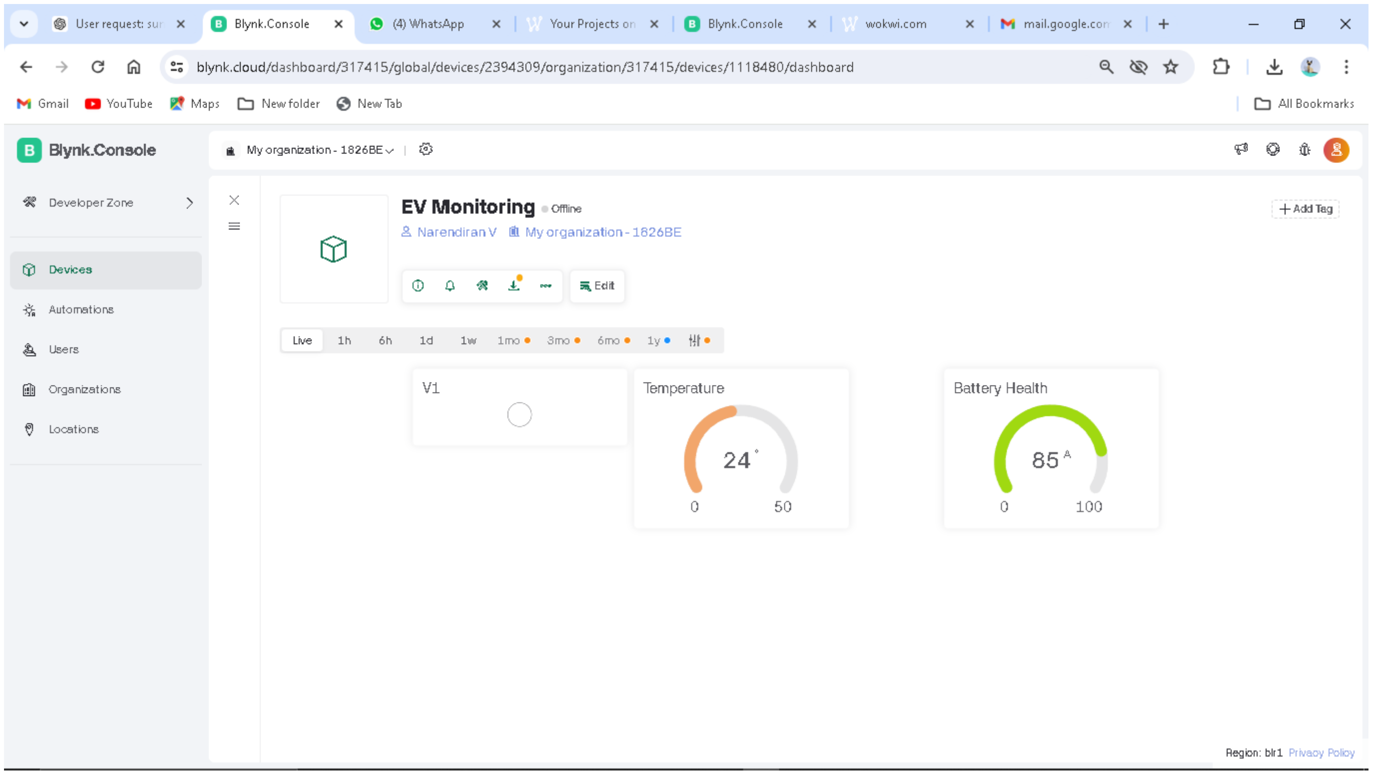
Loop (): Continuously runs Blynk and timer tasks.

1. Circuit Diagram:



```
#define BLYNK_TEMPLATE_ID "TMPL3fP5thwqD"
#define BLYNK_TEMPLATE_NAME "EV Monitoring"
#define BLYNK_AUTH_TOKEN "Sq06EYmfcm7oIBNyxIBDlGp_9oBMm43g"

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

#include "DHTesp.h"

char ssid[] = "Wokwi-GUEST";
char pass[] = "";

const int DHT_PIN = 15;
DHTesp dhtSensor;

const int pir =23;

// Potentiometer is connected to GPIO 34 (Analog ADC1_CH6)
const int potPin = 34;
// variable for storing the potentiometer value
int potValue = 0;

BlynkTimer timer;

void checkBlynkStatus(){
  if (Blynk.connected()) {
    Serial.println("Blynk is connected");
  } else {
    Serial.println("Blynk is not connected");
  }
}

```

2. Blynk Web Console:



RESULT

Integration with Blynk Platform

Successful integration with the Blynk platform, enabling remote monitoring and control.

Real-time accessibility via the Blynk mobile/web application from anywhere with an internet connection.

Provision of critical real-time data on EV conditions for informed decision-making.

9.2 Remote Monitoring and Control Capabilities

Customizable user interface on the Blynk platform, tailored to specific monitoring needs.

Instant notifications on mobile devices for prompt response to system abnormalities or emergencies.

Analysis of sensor data and dynamic adjustment of monitoring parameters for optimized EV performance.

9.3 Data Logging and Analysis

Logging of historical EV data for tracking performance trends and optimizing monitoring strategies.

Robust user authentication mechanisms on the Blynk platform for secure access and control.

CHALLENGES AND RESOLUTION

10.1 Sensor Accuracy

Continuous refinement of sensor calibration through iterative testing and feedback loops.

Collaboration with sensor experts and incorporation of advanced sensor fusion techniques.

10.2 User Engagement and Retention

Implementation of user-centric features such as personalized notifications and interactive community forums.

Targeted marketing campaigns and incentivized user referrals for user acquisition and retention.

10.3 Legal and Copyright Compliance

Development of robust content moderation processes and copyright detection algorithms.

Establishment of partnerships with copyright agencies and provision of clear guidelines for users.

10.4 Scalability and Performance

Employment of scalable cloud infrastructure and database optimization techniques.

Implementation of caching mechanisms, load balancing, and performance monitoring tools.

10.5 Data Security and Privacy

Implementation of robust data encryption techniques and access control mechanisms.

Regular security audits to protect user data from unauthorized access and data breaches.

CONCLUSION

The successful integration of the EV Monitoring and Operation Testing System with the Blynk platform marks a significant advancement in EV monitoring technology. By leveraging IoT capabilities, the system offers enhanced flexibility, efficiency, and convenience in monitoring EV conditions.

Moving forward, it is recommended to:

- Continue refining algorithms for improved data accuracy and reliability.
- Further enhance user engagement strategies to foster a vibrant user community.
- Stay vigilant in maintaining legal compliance and data privacy standards.
- Monitor system performance and scalability to accommodate future growth and demand.

In conclusion, the IoT-based integration with the Blynk platform lays a solid foundation for smarter and more connected EV monitoring, paving the way for more efficient, sustainable, and reliable vehicle management.

REFERENCES

1) Wokwi Simulator, Available at: www.wokwi.com/simulator, Accessed on May 10, 2024.
2) Arduino Official Website, Available at: www.arduino.cc, Accessed on May 10, 2024.
3) Blynk IoT Platform, Available at: www.blynk.io, Accessed on May 10, 2024.

PROGRAM CODE

```
#define BLYNK_TEMPLATE_ID "TMPL3fP5thwqD"

#define BLYNK_TEMPLATE_NAME "EV Monitoring"

#define BLYNK_AUTH_TOKEN
"Sq06EYmfcm7oIBNyxIBDlGp_9oBMm43g"


#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>

#include "DHTesp.h"

char ssid[] = "Wokwi-GUEST";

char pass[] = "";

const int DHT_PIN = 15;

DHTesp dhtSensor;

const int pir =23;

// Potentiometer is connected to GPIO 34 (Analog
ADC1_CH6)

const int potPin = 34;

// variable for storing the potentiometer value

int potValue = 0;

BlynkTimer timer;
```

```
void checkBlynkStatus(){
  if (Blynk.connected()) {
    Serial.println("Blynk is connected");
  } else {
    Serial.println("Blynk is not connected");
  }
}
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  timer.setInterval(5000L, checkBlynkStatus);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  pinMode(pir,INPUT);
  timer.setInterval(1000L, sendData); // Send DHT data
every 2 seconds
  Serial.println("Hello, ESP32!");
}
void sendData() {
  float t = dhtSensor.getTemperature();
  if (isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" °C");
  const int IP=digitalRead(pir);
  Serial.print("Vibration: ");
  Serial.println(IP);
  potValue = analogRead(potPin);
  float volts = 100* (potValue / 4096.0);
  Serial.println("Battery Health: " + String(volts));
  Blynk.virtualWrite(V2, t);
  Blynk.virtualWrite(V1, IP);
  Blynk.virtualWrite(V0, volts);
}
void loop() {
  // put your main code here, to run repeatedly:
  Blynk.run();
  timer.run();
}
```