Institute: Engineering and Technology

Department: Computer Science and Engineering

Course Code: EE1222

Course Title: Internet of Things and Automation

Project Title: Smart University Campus

Project Report

Group 11 Team Members:

Aryan Lunawat (2022BTECH021)

Naman Jain (2022BTECH061)

Divanshu Kachhawa (2022BTECH033)

Faculty Guides:

Er. Divanshu Jain

Mr. Mahesh Saini

November 2024

CERTIFICATE

This certifies that the project titled **"Smart University Campus"**, submitted by **Aryan Lunawat**, **Naman Jain**, and **Divanshu Kachhawa**, in partial fulfillment of the requirements for the Bachelor of Technology degree in Computer Science Engineering at JK Lakshmipat University, Jaipur, is a testament to their work under our supervision and guidance for the course IoT and Automation. We affirm that their work meets the necessary standards for final project submission.

Er. Divanshu Jain

Assistant Professor

Department of Computer Science Engineering

Institute of Engineering and Technology

JK Lakshmipat University, Jaipur

Date of Submission: 30/11/24

ACKNOWLEDGEMENTS

We extend our heartfelt gratitude to our faculty mentors, **Er. Divanshu Jain**, Assistant Professor at IET, JK Lakshmipat University, and **Mr. Mahesh Saini**, Lab Assistant at IET, JK Lakshmipat University, for their invaluable guidance, expertise, and continuous support throughout this project. Their encouragement and cooperation were crucial to its successful completion.

We also wish to express our sincere appreciation to **Dr. Dheeraj Sanghi**, Vice Chancellor of JK Lakshmipat University, and **Dr. Renu Jain**, Director of the Institute of Engineering and Technology, for their consistent encouragement and support during the course of our work.

Finally, we are deeply thankful to our family and friends for their unwavering moral and emotional support, which inspired us to aim for excellence in every step of this journey.

Thank you all for your contributions to the success of our project.

Sincerely Yours,

Aryan Lunawat

Naman Jain

Divanshu Kachhawa

ABSTRACT

The “Smart University Campus” project aims to demonstrate how IoT and Automation can be seamlessly and easily integrated into a university to enhance security, improve resource management, increase efficiency and overall comfort. The project involves various smart applications: a Smart Gate using RFID for access control, Smart Parking with IR sensors for real-time occupancy monitoring, a Smart Classroom with PIR sensors for automated lighting and ventilation based on occupancy, Smart Cafeteria equipped with a flamed detection and alert system and a Smart Garden utilizing a soil moisture sensor to manage automated irrigation.

The system leverages the various sensors and NodeMCU ESP8266 microcontroller for real-time data transmission to the Blynk application, enabling remote monitoring, automation and manual control of essential campus functions. Data gathered from soil moisture sensor is also uploaded to ThingSpeak server for visualization and further analysis using machine learning techniques are deployed to predict future trends, classify soil and detect anomalies.

This comprehensive project showcases the integration of multiple IoT solutions into a single, cohesive system, highlighting the potential for scalable, adaptable smart campus environments. The innovative use of AI/ML and cloud platforms demonstrates the power of IoT in creating intelligent, efficient and sustainable spaces. This project showcases an innovative integration of IoT solutions tailored specifically for educational institutions. This initiative helped us build essential technical skills in IoT hardware integration, programming, and project management. It also encouraged teamwork and improved our problem-solving abilities as we collaborated to overcome challenges.

LIST OF FIGURES

CONTENTS

CHAPTER 1: PROBLEM STATEMENT

Traditional university campuses face various challenges, including security breaches, inefficient resource management, energy wastage and the need for enhanced comfort and convenience for students and staff. Manual monitoring and controlling of various campus operations are time-consuming and prone to errors. There is a need for a smart, automated solution that integrates different IoT-enabled devices to enhance campus security, resource management and overall efficiency.

This project aims to create a cohesive Internet of Things (IoT) solution that automates and monitors essential campus functions, thereby improving overall operational efficiency and user experience.

CHAPTER 2: LITERATURE SURVEY

The research paper titled **"**Internet of Things and Its Applications to Smart Campus: A Systematic Literature Review**"**, published on ResearchGate in December 2022, examined the geographical and demographic distribution of IoT research. It found that the majority of IoT research papers were authored by researchers in Asia, followed by those in Europe and South America. Together, Asia and Europe accounted for 85% of all publications. In contrast, North America and Africa each contributed only 4% of the research, indicating significant research opportunities in these regions.

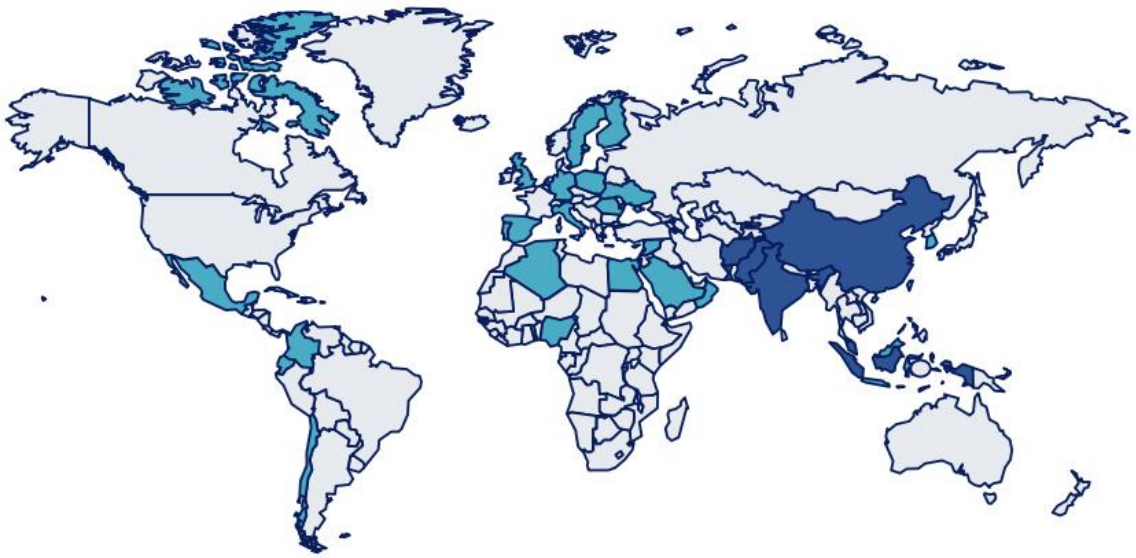


Figure: Geographical Distribution of Authors in IoT Research

This study highlights that most IoT-based smart campus deployments have been conducted as experimental projects, leaving significant potential for real-world applications. Researchers widely recognize smart campuses as ideal testbeds for implementing and refining IoT technologies.

The research paper also outlined various technologies essential for implementing a smart campus using IoT and automation, as illustrated below.

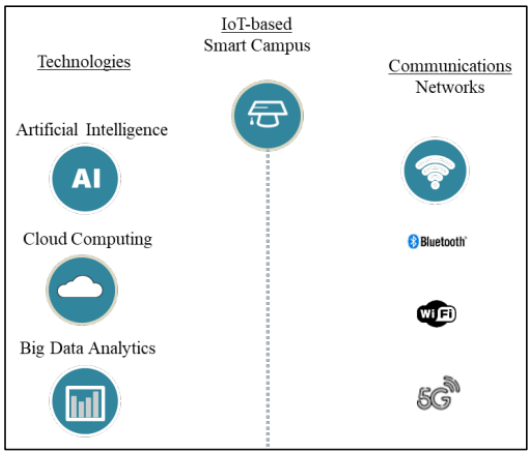


Figure: The technologies required for the smart campus.

The implementation of Internet of Things (IoT) technology on university campuses is a growing trend in the field of campus management. Studies have explored how IoT technology can be used to create a "Smart Campus" that is capable of providing a safe and secure environment for students and faculty.

The research paper titled **"**Design and Implementation of a Smart Campus Flexible Internet of Things Architecture at a Brazilian University**"**, published on IEEE on August 16, 2024, offers significant insights into defining a layered IoT architecture for creating a smart campus. It also outlines the overall operations flow, which is depicted in the accompanying flowchart.

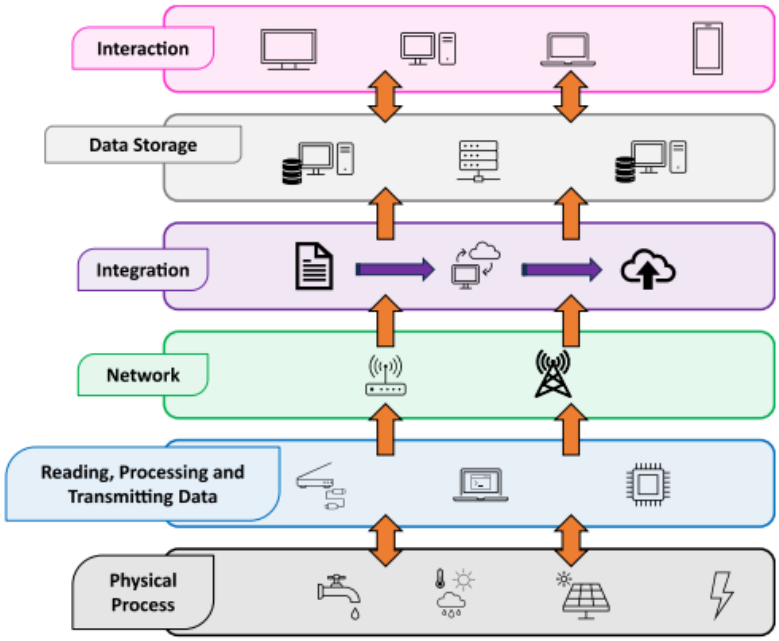


Figure: IoT layered architecture defined for implementing the smart campus.

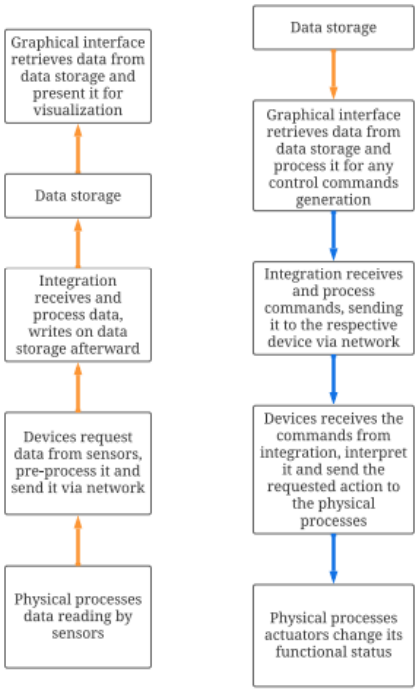


Figure: Architecture general operation flowcharts.

1. IoT in Smart Campus Solutions:

- IoT-based automation systems have been widely implemented for smart homes and industries but are still evolving for university campuses.
- Existing solutions focus on individual components like RFID-based access control or smart energy systems but lack integration for holistic campus management.

2. Technologies Used:

- **RFID**: Secure access control using unique identification.
- **IR Sensors**: Accurate detection for counting and monitoring parking slots.
- **PIR Sensors**: Detect motion for energy-efficient room management.
- **Soil Moisture Sensors**: Automate irrigation.
- **Flame Sensors**: Ensure safety by detecting gas leaks and fire.

3. Blynk IoT Platform:

- Blynk provides a centralized dashboard for real-time monitoring and controlling IoT devices. It offers flexibility in visualization and control, suitable for diverse automation needs.

Recent studies have extensively explored the adoption of IoT technologies in enhancing smart campus solutions. IoT enables the interconnection of devices and sensors, allowing for real-time data collection and analysis. Key theories underpinning this project include:

- **Smart Automation**: Utilizing sensors and actuators to automate tasks such as lighting control, access management, and environmental monitoring.
- **Data Analytics**: Employing machine learning techniques to analyze collected data for predictive maintenance and resource optimization.
- **User-Centric Design**: Ensuring that the system is intuitive and accessible through platforms like Blynk, facilitating user interaction with smart devices.

The implementation of IoT in various sectors has shown significant improvements in operational efficiency and security. IoT-enabled smart campuses leverage advanced sensors, microcontrollers, and automation to create a cohesive and adaptive environment. Previous studies have highlighted the benefits of using RFID for access control, IR sensors for parking management, PIR sensors for presence detection, and soil moisture sensors for environmental monitoring. The use of cloud platforms like ThingSpeak and mobile applications like Blynk for real-time data monitoring and control further enhances the system's efficiency and usability.

CHAPTER 3: PROJECT OBJECTIVES

- **Enhance Security:** Implement a smart gate lock system controlled by RFID technology to restrict access based on authorized personnel.
- **Parking Assistance:** Design a smart parking system to display slot availability in real-time to users using IR sensors.
- **Optimize Resource Management and Improve Comfort:** Create automated classroom environments that adjust lighting and ventilation based on occupancy using PIR sensors to automate energy usage.
- **Ensure Safety:** Integrate fire detection systems in cafeteria using flame sensor to alert users of potential hazards using notifications and alarms.
- **Environmental Monitoring:** Deploy soil moisture sensors in gardens to automate irrigation based on moisture levels to optimize water usage.
- **Centralized Control:** Integrate all components with the Blynk app for seamless monitoring and control in real-time.
- Send soil moisture sensor data to ThingSpeak for visualization and to Integrate AI/ML techniques on the collected data for predictive analysis, soil classification and anomaly detection.
- Develop a scalable and customizable IoT solution for university campuses.

CHAPTER 4: INNOVATIONS AND DISTINCTIONS

- **Holistic Integration:** Unlike existing isolated IoT solutions, this project combines security, parking, energy management, and safety into a single centralized and cohesive system.
- **Real-Time Monitoring:** Ensures up-to-date data visualization on the Blynk app providing remote monitoring and control.
- **Machine Learning Applications:** The project incorporated AI/ML techniques for predictive analysis of soil moisture levels, soil classification and anomaly detection, enhancing agricultural practices on campus.
- **Scalability and Customizability:** Can be expanded to include additional features or components allowing future expansions.
- **Cost-Effectiveness:** Uses affordable components without compromising functionality.
- **User-Friendly Interface:** The use of Blynk allows users to control various systems remotely through a mobile application.

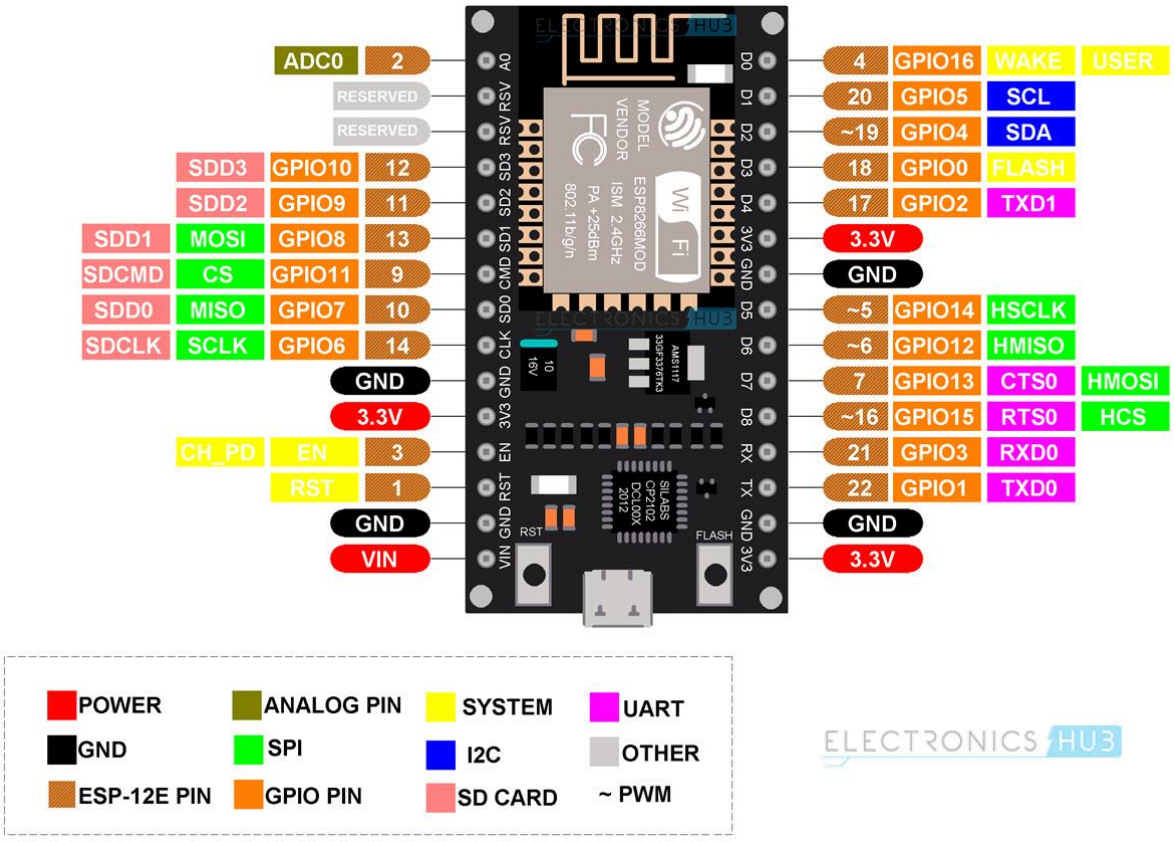
CHAPTER 5: COMPONENTS USED

1. NodeMCU ESP8266:

The NodeMCU (Node MicroController Unit) is an open-source platform tailored for developing Internet of Things (IoT) applications. It integrates both software and hardware tools built around the ESP8266 System-on-a-Chip (SoC) from Espressif Systems. NodeMCU's offering includes firmware that runs on the ESP8266 Wi-Fi module, coupled with a development board based on the ESP-12 module. Its firmware, designed for ease of use, employs the Lua scripting language, which makes it perfect for rapid prototyping and IoT project development.

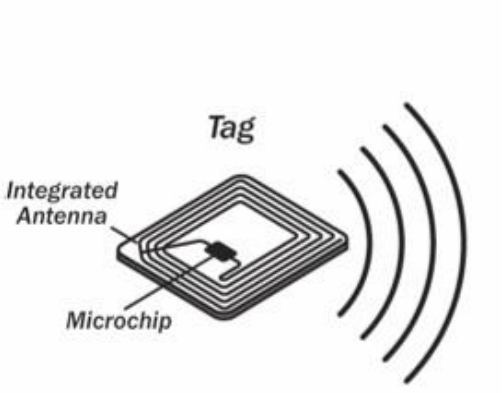
NodeMCU Pinout:

The image below illustrates the pinout of a NodeMCU board. A standard NodeMCU board, typically following the original NodeMCU Devkit design, features 30 pins in total. Of these, 8 pins are designated for power, and 2 are reserved for specific purposes. The other 20 pins are directly linked to the pins of the ESP-12E module.



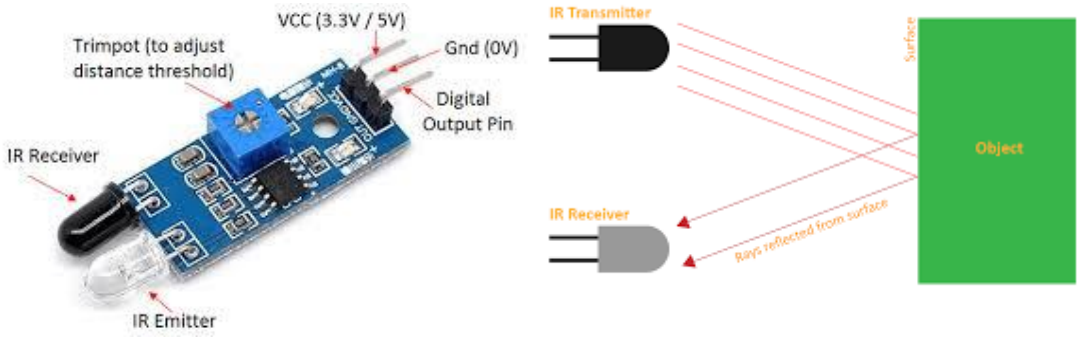
2. **RFID Reader Module (MFRC522):**
   The RFID Reader Module (MFRC522) is a small, affordable, and efficient device often used in RFID-based projects. It operates at a frequency of 13.56 MHz, enabling wireless communication with RFID tags. This module can both read and write data to compatible tags, making it versatile for different applications. With a communication range of 3-5 cm and low power requirements of 2.5V to 3.3V, it's ideal for battery-powered setups. Designed for easy integration, it works smoothly with microcontrollers like Arduino or Raspberry Pi and supports multiple communication protocols, such as SPI, I2C, and UART, making it flexible for a variety of uses.

3. **RFID Tags:**
   RFID Tags are essential components of an RFID system, used to store and transmit data wirelessly to an RFID reader. Each tag contains a small integrated circuit for processing and memory, along with an antenna to communicate with the reader. Passive RFID tags, used in this project, operate at a frequency of 13.56 MHz and store a unique identifier (UID). These tags rely on the RFID reader's signal for power, making them lightweight, cost-effective, and suitable for short-range applications like access control and asset tracking. When within range of the reader, they securely transmit stored data for identification.



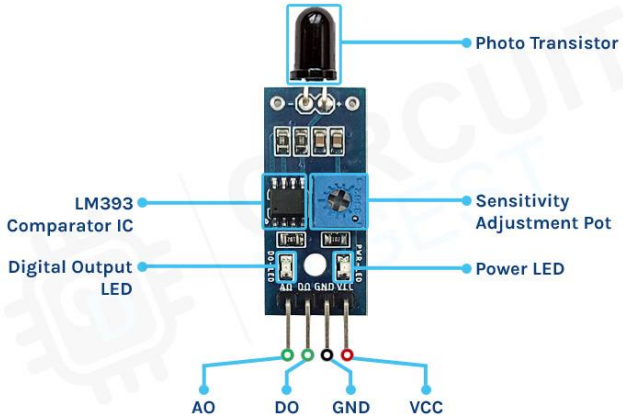
4. **IR Sensor:**

An IR sensor is an electronic device that uses infrared light to detect nearby objects by sensing the reflected infrared radiation. Commonly used for proximity sensing and object detection, an IR sensor comprises an IR emitter, like an LED, and an IR detector, such as a photodiode or phototransistor. The emitter projects infrared light, which reflects off objects, and the reflected light is then picked up by the detector. Depending on this detection, the sensor generates a digital output: a LOW signal (0) indicates the presence of an object, whereas a HIGH signal (1) denotes no object detected.



5. **Flame Sensor:**

A flame sensor is designed to detect flames or fire by identifying the infrared light emitted by the flame. It typically consists of an infrared receiver and related circuitry to process the signal. The sensor is sensitive to flame wavelengths and provides a digital output: LOW (0) when a flame is detected and HIGH (1) when no flame is present. It is commonly used in fire detection systems and can be integrated with microcontrollers like Arduino and Raspberry Pi for various safety applications.

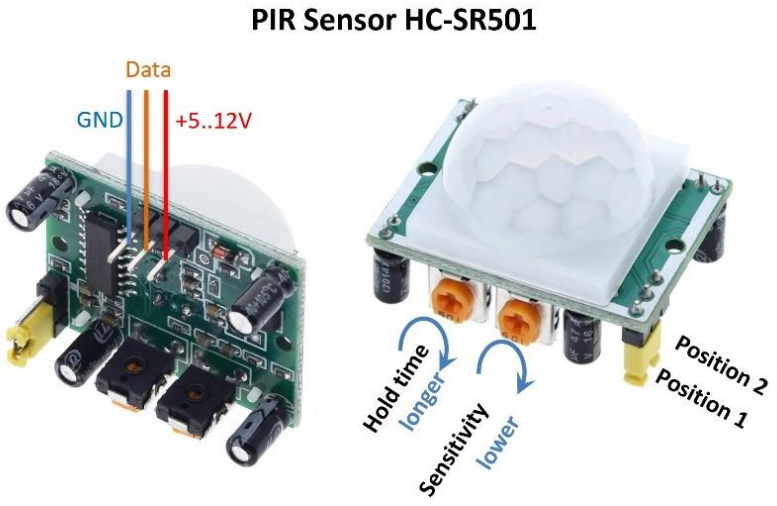


6. **Buzzer:**

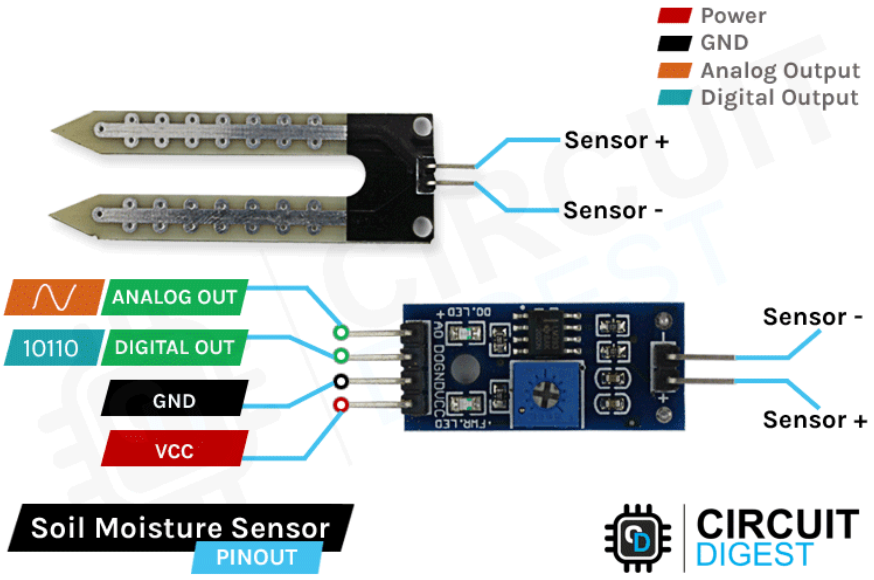
A buzzer is an electronic device used to generate sound or an audible alert, typically used in alarm/alert systems or as a signalling device. It works by converting electrical energy into sound using an electromagnetic coil or piezoelectric element. Its compact design and minimal power usage make it perfect for use in projects with limited space and battery power.

7. **PIR Sensor (HCSR501):**

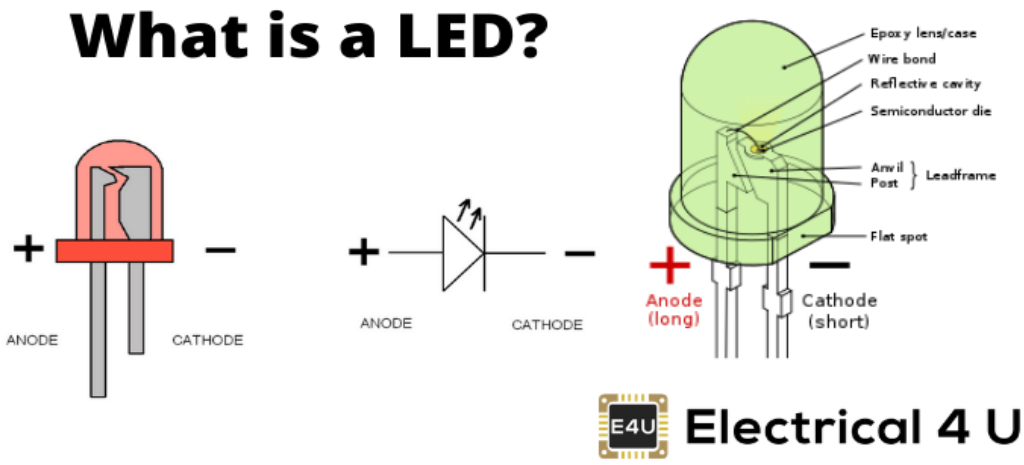
A Passive Infrared (PIR) sensor detects movement by monitoring variations in infrared radiation from objects. PIR sensors are widely used in applications such as security systems, automatic lighting, and energy management due to their efficiency and cost-effectiveness. When a warm object, like a human, moves, the sensor's special elements detect this change and generate a signal. Key components include a material that senses infrared changes, a lens that focuses the light, and a circuit that processes the signal. These sensors are known for their low power consumption and wide detection range but may be less accurate in extreme temperatures and primarily detect warm objects.



8. **Soil Moisture Sensor (YL-69):**
   A soil moisture sensor operates using two conductive plates that serve as probes and function as a variable resistor. The sensor measures the water content in soil by detecting changes in resistance between the probes. When placed in moist soil, the resistance decreases, enabling better electrical conductivity between the plates. In contrast, dry soil exhibits higher resistance and lower conductivity, indicating reduced moisture levels. This variation in resistance is used to assess soil moisture, making the sensor highly suitable for applications such as automated irrigation systems and agricultural soil monitoring.

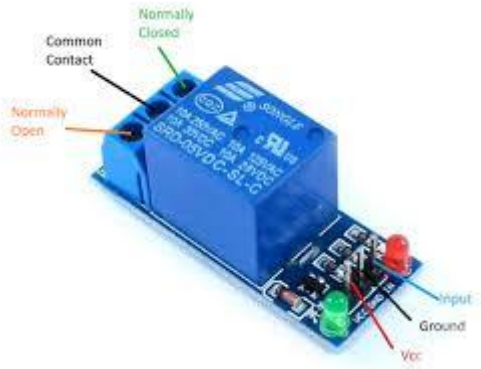


9. **Small LED Lights:**
   Light-Emitting Diode that lights up when electricity passes through it in the correct direction.



10. 5V Single Channel Relay Module:

The 5V single-channel relay module is an electrically operated switch that enables low-voltage digital signals from microcontrollers like NodeMCU or Arduino to control high-voltage AC or DC devices, such as appliances or lights. It operates on a 5V power supply and can handle loads up to 250V AC or 30V DC at 10A. The relay module has two states: Normally Open (NO), where the circuit is open and the connected device remains off, and Normally Closed (NC), where the circuit is closed and the device is powered on. It includes an optocoupler for electrical isolation, ensuring safe control of high-voltage devices.



11. 5V Small DC Motor:

A small DC motor is a compact electric motor that converts electrical energy into mechanical motion. Operating on low voltages (e.g., 3V or 5V), it's commonly used in small projects like robotics and automation. The motor consists of a stator with a magnet and a rotor with windings. When powered, the rotor rotates due to the magnetic field. Speed can be controlled by adjusting the voltage, and direction can be reversed by changing the polarity. These motors are ideal for applications requiring precise movement.

12. Mini Submersible Pump:

A mini submersible pump is a small, water-resistant pump designed to move liquids, typically water, from one location to another. It operates by using an electric motor to drive an impeller, which creates pressure to push water through a tube or pipe. These pumps are submersible, meaning they can be fully submerged in water without damage. Mini submersible pumps typically operate on low voltage (e.g., 5V or 12V) and are compact, making them ideal for tasks that require space-saving solutions.

13. Wires:

These are thin metal strips that create an electrical connection between two components in a circuit. They offer minimal resistance to current flow and are typically coated with an insulating layer.

14. Power bank with USB type-b cable:

To directly supply power to circuit model using USB type-b cable.

15. Laptop:

Need laptop to upload code and also to use Blynk and ThingSpeak.

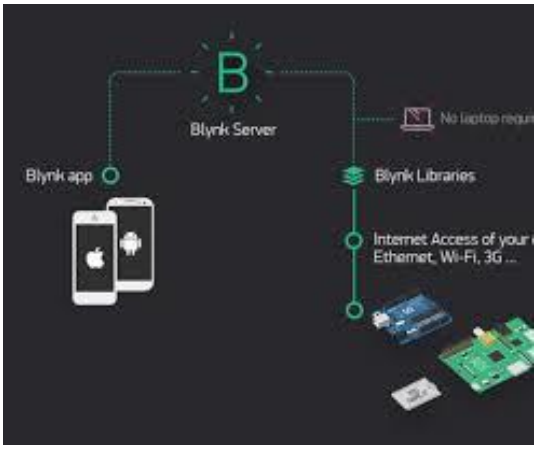
CHAPTER 6: SOFTWARE USED

1. **Arduino IDE**

   The Arduino IDE (Integrated Development Environment) is an open-source software platform designed for writing, compiling, and uploading code to microcontrollers such as the NodeMCU ESP8266. It offers an intuitive interface, enabling developers to write programs in the Arduino programming language, which is derived from C/C++. In this project, the Arduino IDE was used to write the necessary code for the NodeMCU ESP8266, which controls various IoT devices and communicates with external platforms like Blynk and ThingSpeak. The IDE supports easy integration with libraries and provides features such as code auto-completion, error checking, and serial monitoring to facilitate debugging. Once the code is written, it is compiled and uploaded directly to the NodeMCU ESP8266 using a USB connection, enabling the microcontroller to execute the programmed tasks.

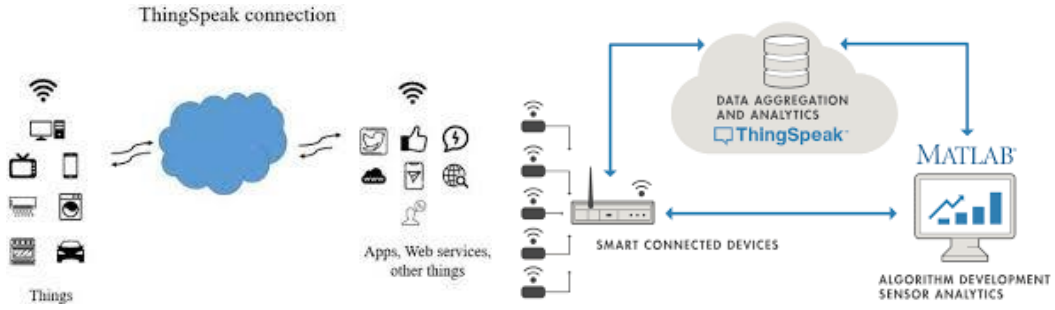
2. **Blynk:**

   Blynk 2.0 is a cloud-based IoT platform that enables the remote control and monitoring of hardware using a mobile app. It allows users to control electronic appliances and monitor sensor data in real time, with features like secure cloud storage, alerts, and notifications. Blynk provides a simple and intuitive interface for managing IoT projects, while NodeMCU, an open-source firmware and development board, facilitates the creation of IoT devices with Wi-Fi connectivity. Blynk's use of Virtual Pins creates an abstraction between the hardware and the app, simplifying data exchange and enabling seamless control and monitoring of the system. This platform offers real-time data visualization, making it easier to interact with and manage hardware remotely.



Blynk

3. **ThingSpeak:**
   ThingSpeak is an open-source IoT platform that uses Ruby and provides MATLAB-based analytics. It allows for the aggregation, visualization, and analysis of real-time data streams in the cloud. It supports data communication over HTTP, allowing for remote device control. We utilized private data channels to store and monitor sensor data, ensuring privacy and security. ThingSpeak's integration with MATLAB enables advanced data visualization and analysis, and its inbuilt tools help create charts to plot real-time data. The REST API facilitates seamless communication between the NodeMCU and ThingSpeak for efficient data upload and retrieval.



ThingSpeak™

CHAPTER 7: SCHEMATIC DIAGRAMS

Board 1: Smart Entrance and Smart Parking:

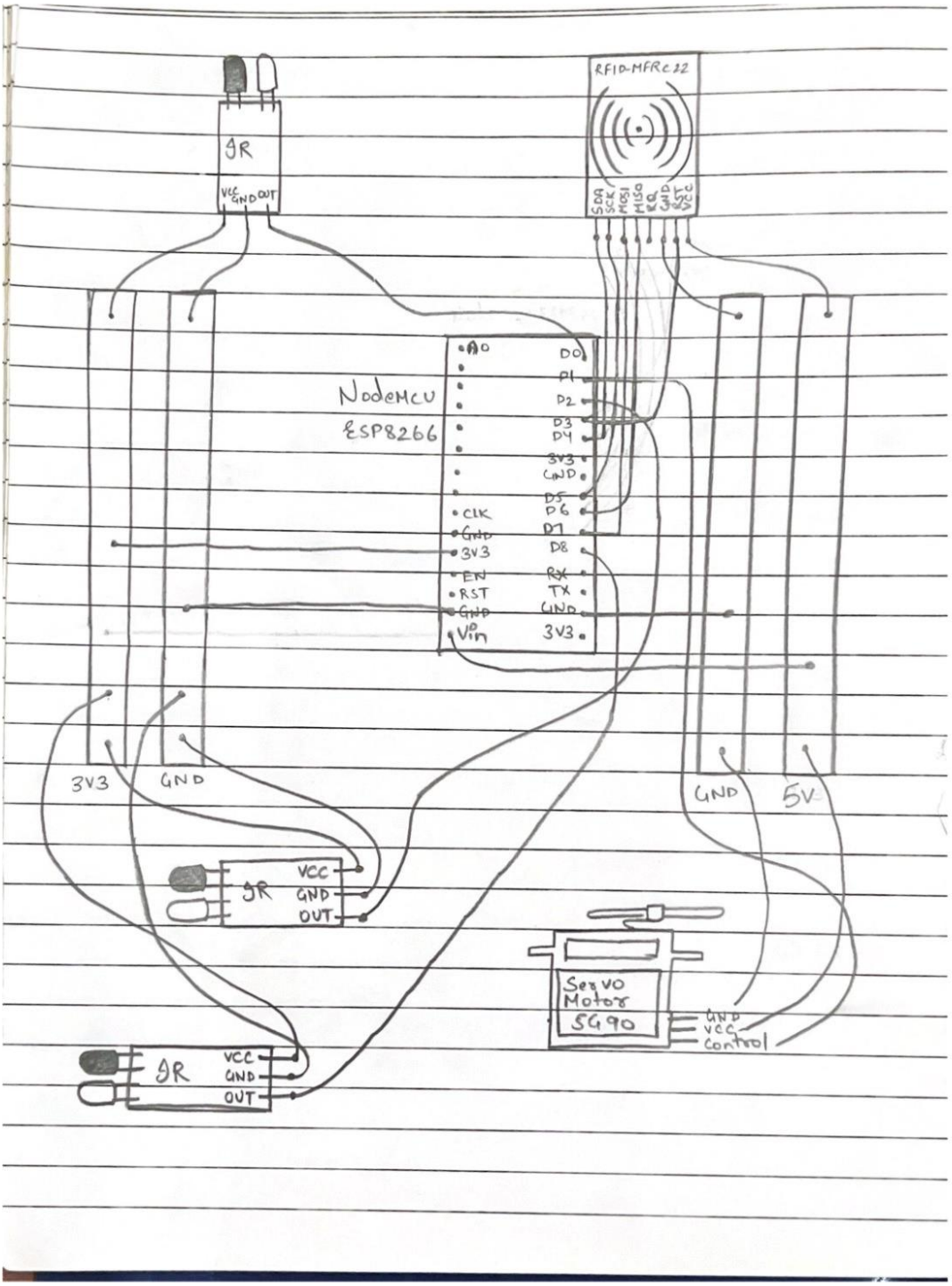


Figure: Handmade Circuit Diagram of Board 1

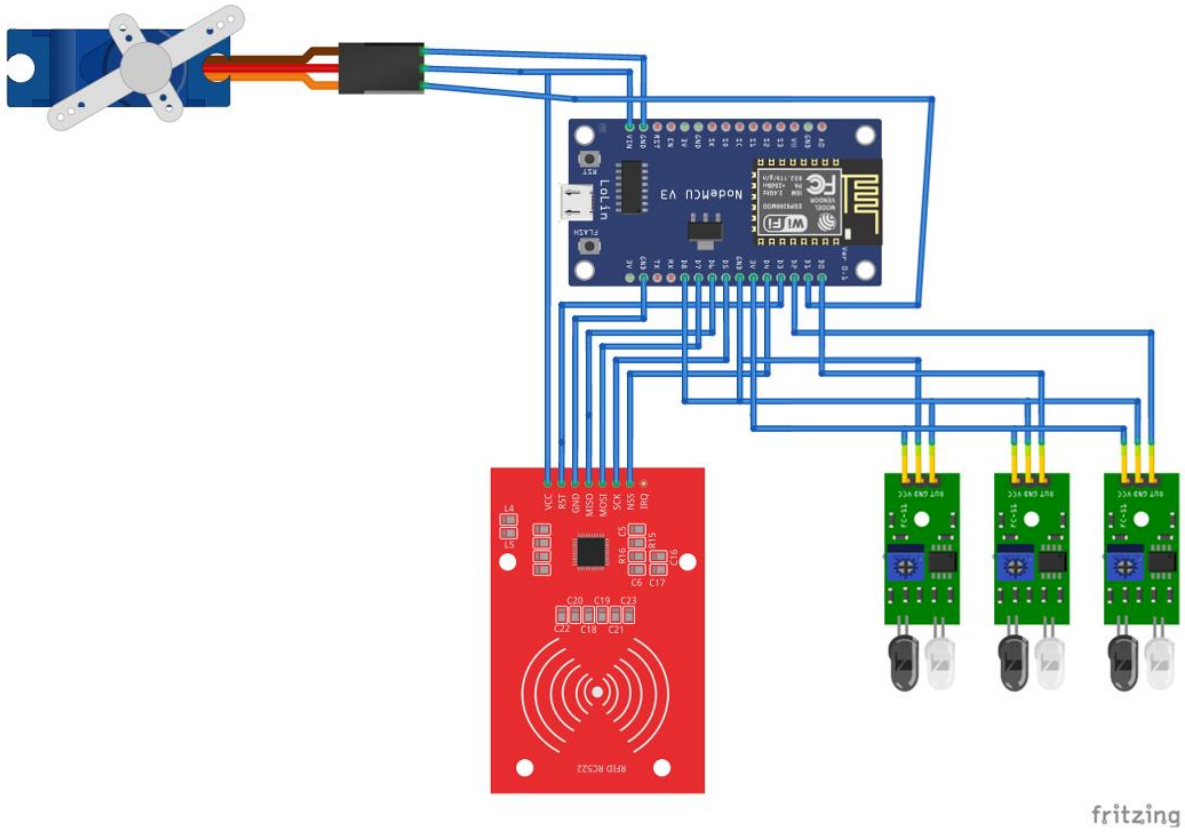


Figure: Circuit Diagram of Board 1 made on Fritzing software

Board 2: Smart Classroom, Smart Cafeteria and Smart Garden.

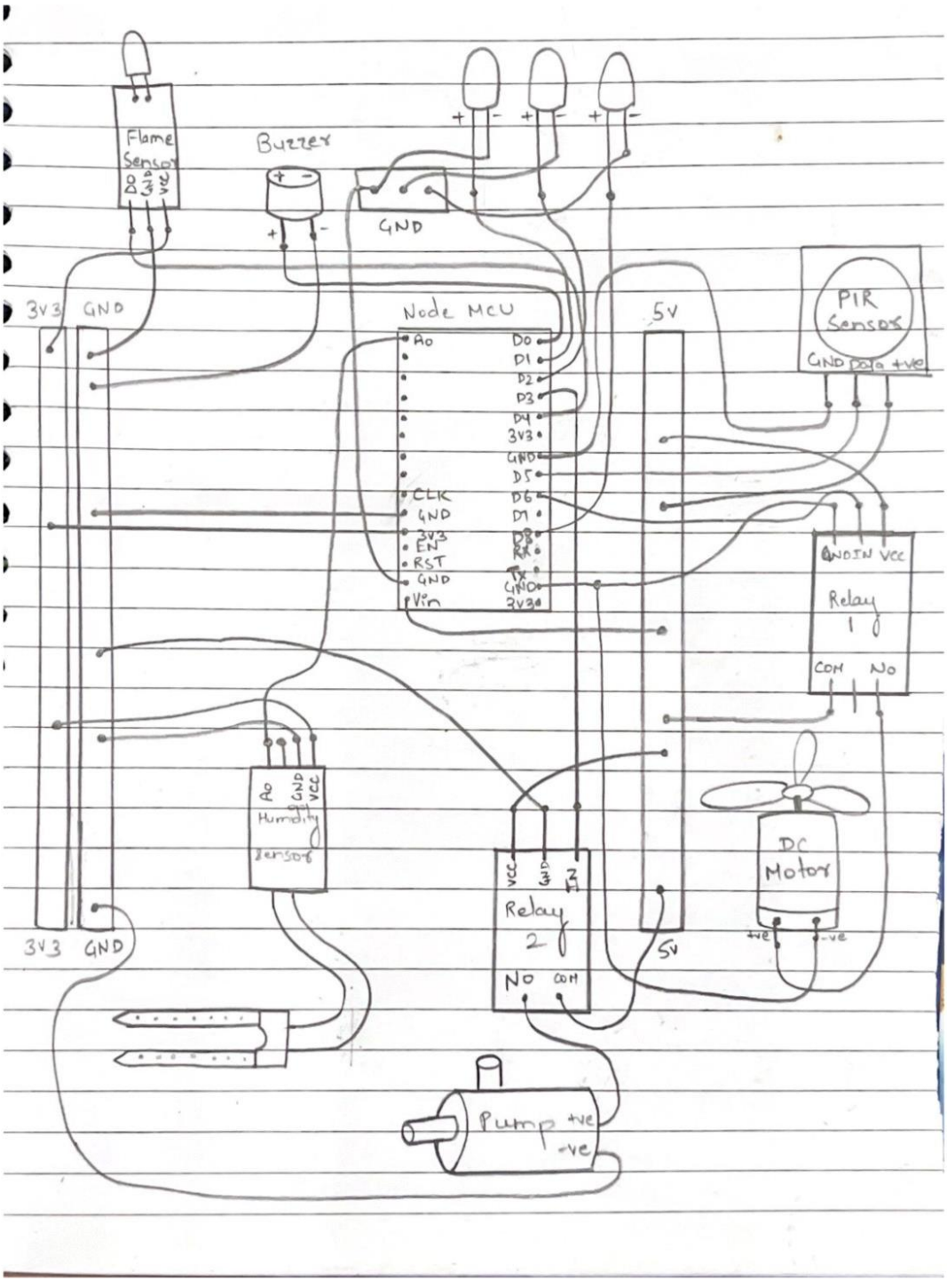


Figure: Handmade Circuit Diagram of Board 2

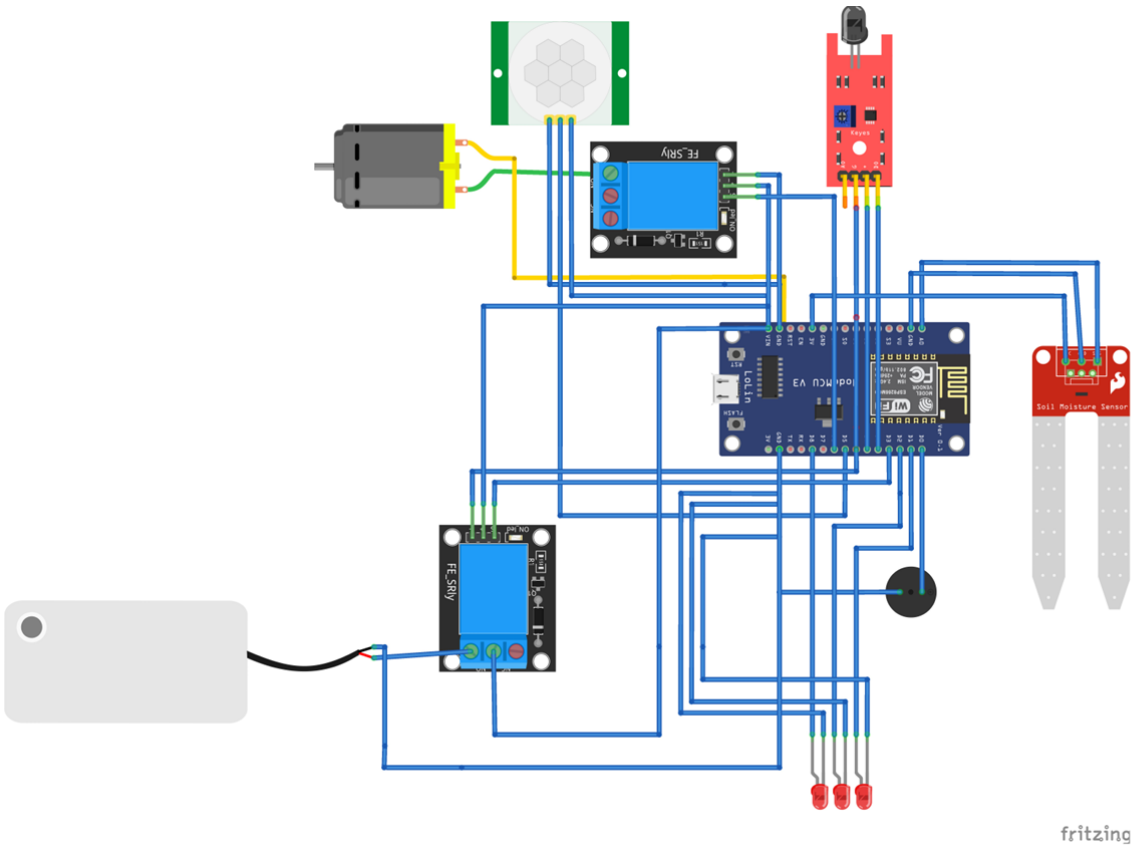


Figure: Circuit Diagram of Board 2 made on Fritzing software

CHAPTER 8: METHODOLOGY

1. **System Design:** Design the architecture of smart university campus, including sensor placements and communication protocol as shown in the circuit diagram.

2. **Blynk Setup:**
   - i. **Create a Blynk Account:**
     - Open the Blynk website or download the Blynk app.
     - Create an account using your email and log in.
   - ii. **Create a New Project:**
     - Click on create New Project
     - Select NodeMCU as the device and specify the connection type being Wi-Fi.
   - iii. **Obtain Credentials:**
     - After project creation, note down the Auth Token, Template ID and Template Name.
     - These credentials will be used in the Arduino IDE code for connecting the device.
   - iv. **Define Datastreams:**
     - In the project dashboard, navigate to Datastreams.
     - Create a datastream for each sensor or actuator you want to use (e.g., PIR sensor, Pump, etc.).
     - Set the data type (e.g., integer, float) and connection pin (virtual or physical) along with min/max and default values.
   - v. **Event Notifications (Optional):**
     - Set up Event in Blynk to trigger actions or send notifications based on specific conditions.
     - Notifications can also be sent via email or push notifications.
   - vi. **Add Widgets:**
     - Open the Blynk project dashboard & add widgets (e.g., Switch, Gauge, Label, Button) for controlling or monitoring.
     - For each widget, link the corresponding Datastream.
   - vii. **Configure and Save:**
     - Save changes for each widget and project settings.
     - Ensure all datastreams and widgets are configured correctly for real-time interaction.

All this can also be setup in the same way in Blynk Mobile App and can be controlled and monitored easily.

3. ThingSpeak Setup:

- **Sign up for ThingSpeak:**
  - Visit ThingSpeak
  - Create a free account or log in if you have one.
- **Create a new Channel:**
  - After logging in, go to Channels on the top menu and click New Channel.
  - Name your channel (e.g., "Smart Garden") and add a Field 1 to store the soil moisture data. Field 1 will store soil moisture reading.
- **Get your API Key:**
  - Go to the API Keys tab of your channel to get the Write API Key.
  - You'll use this API key to write data to ThingSpeak from your ESP8266.
- **Export Data from ThingSpeak:**
  - Go to channel and click on Data tab.
  - Click on Export CSV option.

4. Configuring Arduino IDE and Libraries for ESP8266 Projects

i. Adding ESP8266 Core to Arduino IDE

- Launch the Arduino IDE and navigate to File > Preferences.
- In the Additional Board Manager URLs field, paste the following link:
- http://arduino.esp8266.com/stable/package_esp8266com_index.json
- Go to Tools > Board > Boards Manager.
- Search for "ESP8266" in the Boards Manager.
- Install the latest version of the esp8266 package by the ESP8266 Community.
- Under Tools > Board, choose your board (e.g., NodeMCU 1.0).
- Adjust any additional settings, like Baud Rate, as required

ii. Install Blynk Libraries

- Download the latest release of the Blynk libraries from the official Blynk GitHub repository.
- Unpack the downloaded .zip file.
- Move the extracted library folders (e.g., Blynk and BlynkESP8266) to the Arduino libraries directory: C:/Users/<YourUsername>/Documents/Arduino/libraries
- Restart Arduino IDE to ensure the libraries are recognized.

5. Setup Arduino IDE:

For programming and uploading code to microcontroller: NodeMCU ESP8266.

i. Open the Arduino IDE, select Tools->Board->NodeMCU and then select the appropriate COM port.
ii. Go to Sketch>Include Library>Manage Libraries and install these libraries and include them in code:
    - 'BlynkSimpleEsp8266.h' for Blynk integration.
    - 'MFRC522.h' for RFID functionality.
    - 'ESP8266WiFi.h' for Wi-Fi connectivity.
    - 'ThingSpeak.h' for data logging.
    - 'Servo.h': Control the servo motor for smart door lock.
    - 'SPI.h': Facilitate SPI communication.
iii. Write required code for working of the project.
iv. Paste the Auth Token, Template Name & Template ID in the code.
v. Connect the NodeMCU ESP8266 to the Wi-Fi network.
vi. Verify and upload the sketch to NodeMCU.

6. **Component Integration:**
   Choose appropriate hardware components and design circuits for RFID, IR, PIR, Soil Moisture and Flame sensors. Connect each component with NodeMCU ESP8266 for communication.

   Board 1 Setup:

   - **Smart Gate:**
     - Components: RFID Reader MFRC522, Servo Motor SG90, RFID Tags, NodeMCU ESP8266.
     - Functionality: Automates gate access control using RFID and Blynk for manual control.
     - Blynk Setup: 1 Control Switch widget to lock/unlock gate.
     - Make connections and upload code.
   - **Smart Parking:**
     - Components: 3 IR Sensors, NodeMCU ESP8266.
     - Functionality: Monitors and displays parking slot occupancy on Blynk.
     - Blynk Setup: 3 Display LED widgets to show occupancy of each of the 3 parking spots and 1 Display Value widget to show the total available parking slots.
     - Make connections and upload code.

   Board 2 Setup:

   - **Smart Classroom:**
     - Components: PIR Sensor, 5V DC Motor (with plastic fan attached), small LED Light, Relay, NodeMCU ESP8266.
     - Functionality: Automates fan and light control based on room occupancy.
     - Blynk Setup: 2 Control Switch widgets to turn Light and Fan ON/OFF
     - Make connections and upload code.
   - **Smart Cafeteria:**
     - Components: Flame Sensor, Buzzer, NodeMCU ESP8266.
     - Functionality: Automates watering based on soil moisture levels and displays sensor data on Blynk.
     - Blynk Setup: Enable Event Handling for this such that when flame detected, push notification and email send to user.
     - Make connections and upload code.

- **Smart Garden:**
    - Components: Small Submersible Pump, Soil Moisture Sensor YL-69, LEDs (one red and one green), Relay, NodeMCU ESP8266.
    - Functionality: Automates watering based on soil moisture levels and displays sensor data on Blynk.
    - Blynk Setup: 1 Control Switch widget to turn pump ON/OFF and 1 Gauge widget to display soil dryness value.
    - ThingSpeak Setup: Create a new channel and add a Field 1 to store the soil moisture data.
    - Machine Learning Setup: Upload CSV data file obtained from ThingSpeak and implement future prediction, soil classification and anomaly detection. Processed data CSV file can be obtained by downloading it.
    - Make connections and upload code.

7. **Machine Learning Setup:**
    1. **Upload csv file obtained from ThingSpeak.**
    2. **Install required libraries for Machine Learning**
    3. **Data Preprocessing:**
        - Convert created_at to datetime and set it as index
        - Create lag features
        - Split data to training and testing sets
    4. **Future Predictions:**
        - Train a Random Forest Regressor
        - Iteratively predict future values
        - Update lagged values for the next iteration
        - Print predicted future values and plot it
    5. **Soil Condition Classifier:**
        - Define condition for classification and apply it
        - Train a Decision Tree Classifier
        - Predict Soil Condition of old and new input data
    6. **Anomaly Detection:**
        - Visualize data distribution
        - Train isolation forest model
        - Filter anomalies and visualize them
    7. **Download processed soil data**

8. **Testing & Validation:** Connect the ESP8266 board (NodeMCU) to a power bank using a USB cable and upload the code. Conduct tests to ensure each system component functions correctly with the integrated environment.

Workflow:

1. **System Setup:**
    - Design architecture, place sensors (RFID, PIR, IR, etc.), and integrate with NodeMCU ESP8266.
2. **Blynk Setup:**
    - Create a project, configure Datastreams, and add widgets for monitoring and control.
3. **ThingSpeak Setup:**
    - Create a channel, configure fields, and obtain the API key for data logging.
4. **Arduino IDE Configuration:**
    - Install libraries, write and upload code to NodeMCU for sensor processing and control.
5. **Data Communication:**
    - Sensors send data to NodeMCU for processing, uploaded to Blynk for monitoring and ThingSpeak for logging.
6. **Visualization & Analysis:**
    - Use Blynk for real-time monitoring, ThingSpeak for trends, and export data for ML analysis (predictions, classifications, anomaly detection).
7. **Validation & Actuation:**
    - Test the system; NodeMCU triggers actuators based on sensor inputs and Blynk commands

CHAPTER 9: RESULTS

Smart University Campus:

Figure: Complete Model of Smart University Campus

Smart Entrance:

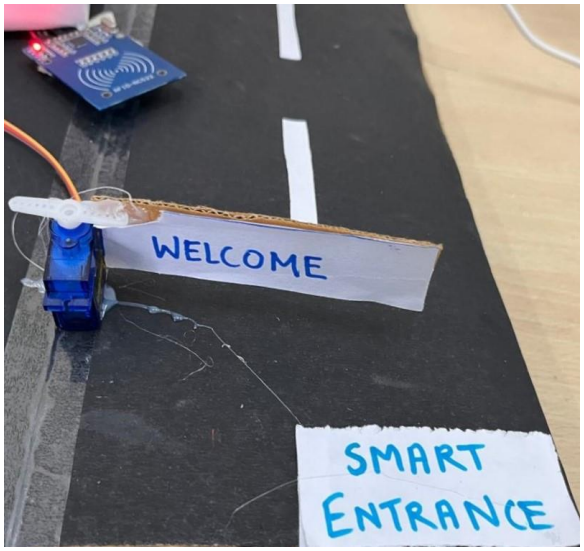


Figure: Smart Gate Locked

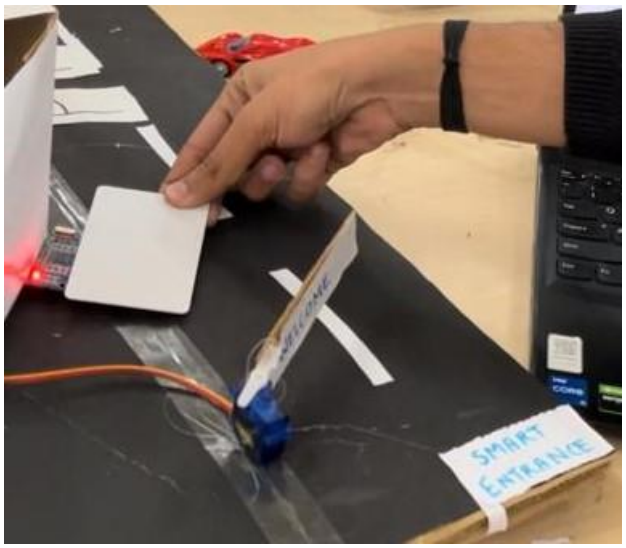


Figure: Smart Gate Unlocked

Smart Parking:

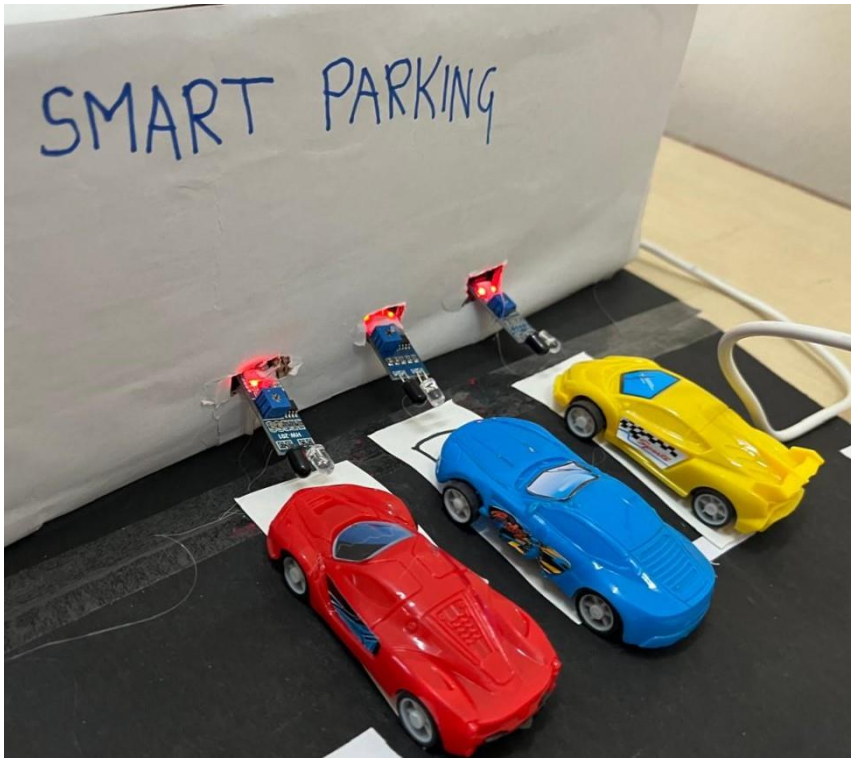


Figure: Smart Parking of University Campus

Smart Gate and Smart Parking Integration on Blynk:

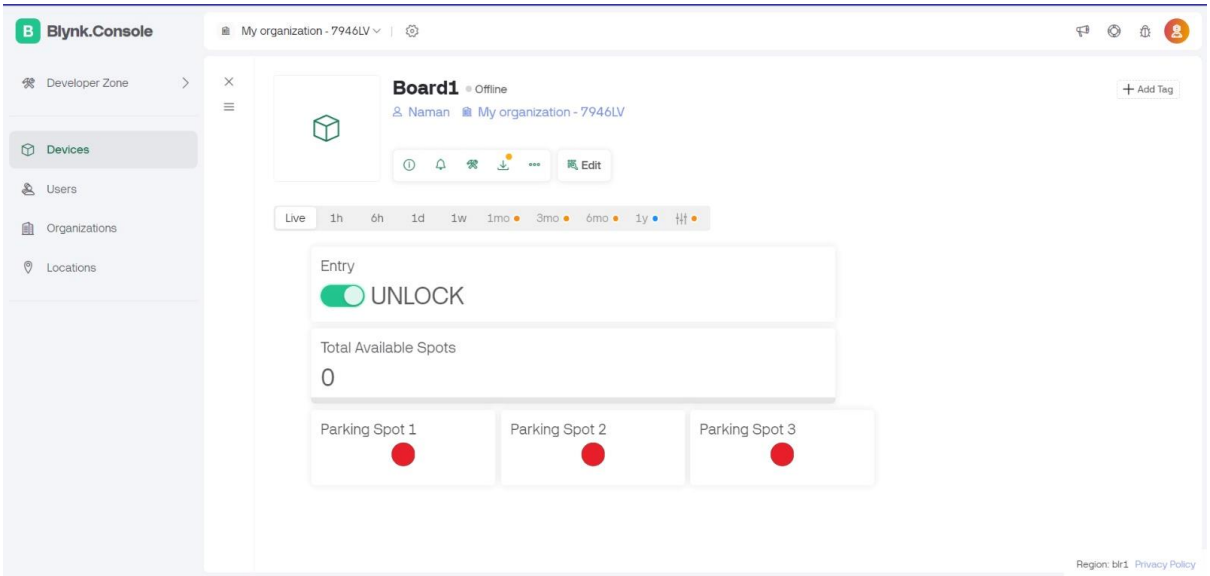
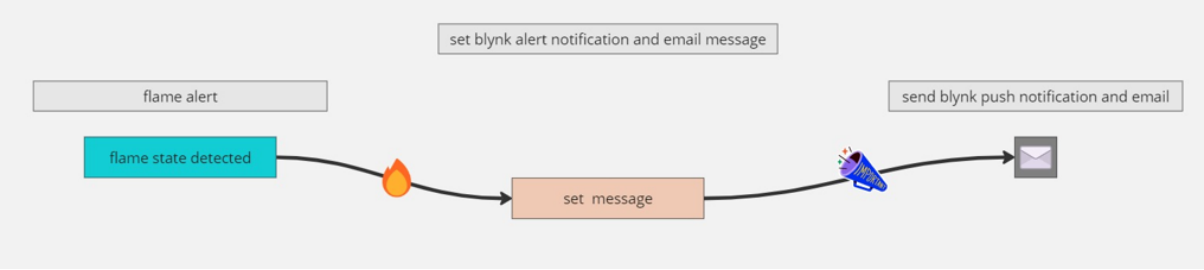


Figure: Dashboard of Smart Gate and Smart Parking on Blynk

Smart Cafeteria:

Figure: Smart Cafeteria of University Campus

Alert System Flowchart:



Fire Alert on Smartphone:

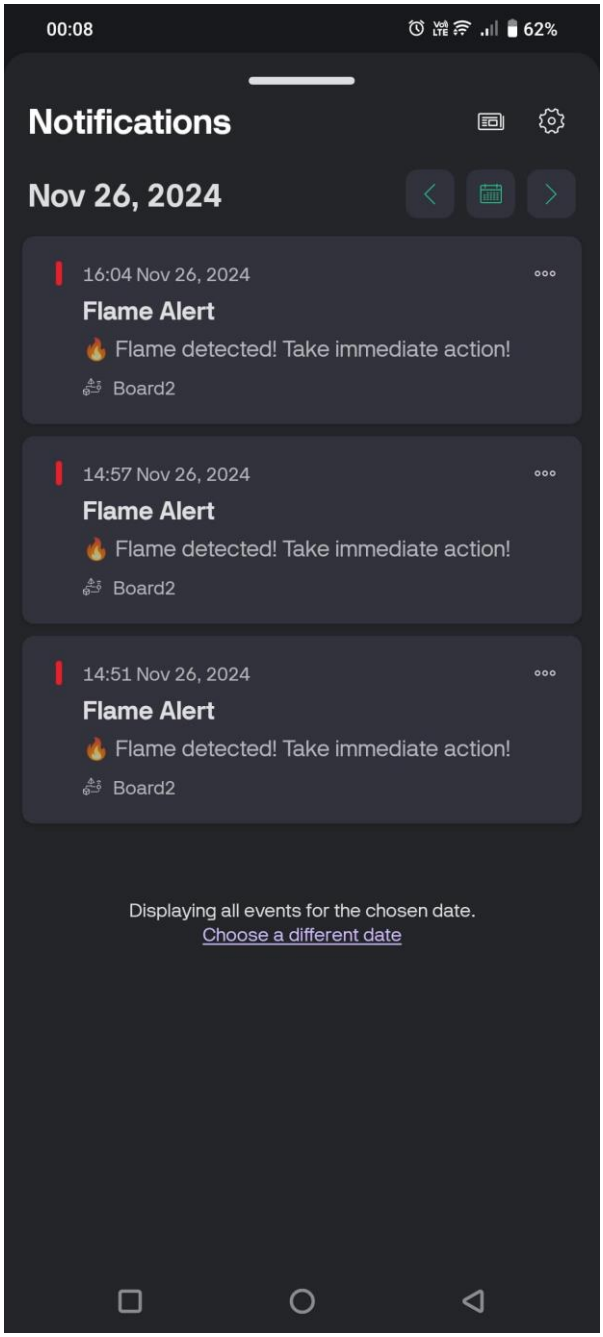


Figure: Alert Notification on Phone

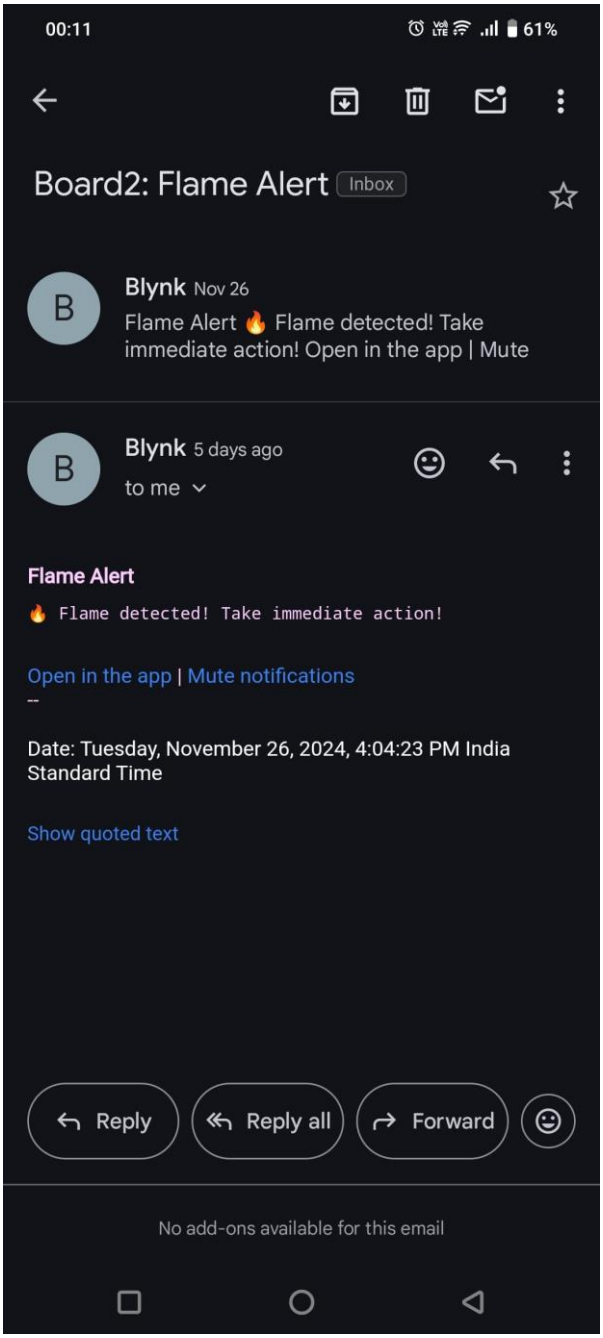


Figure: Fire Alert Message on Gmail.

Kitchen Fire Alert Notification on Blynk:

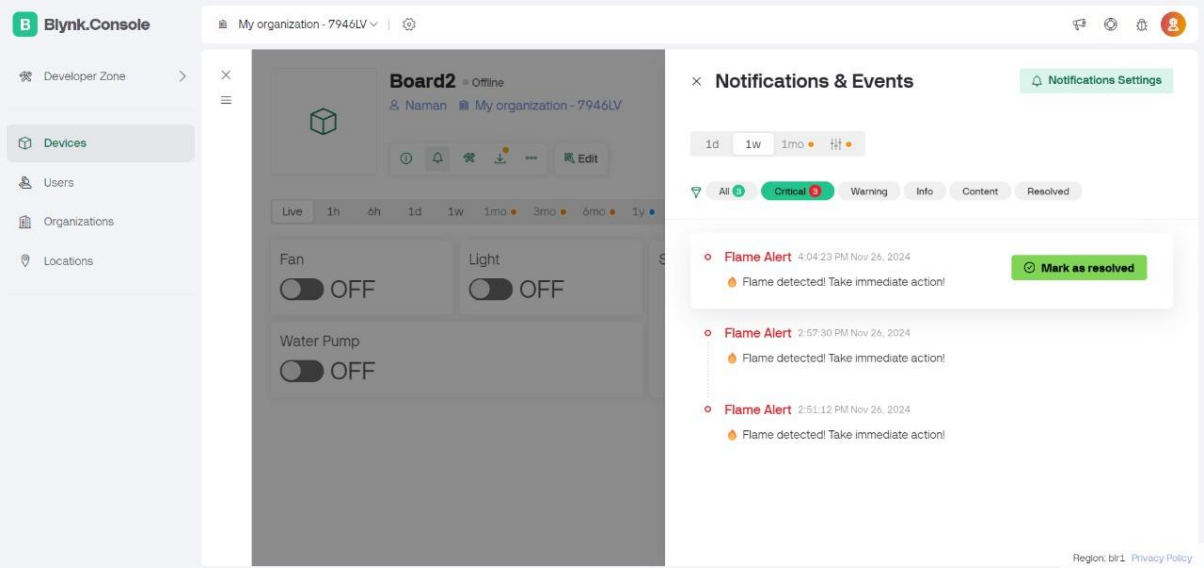


Figure: Flame Alert Notification on Blynk

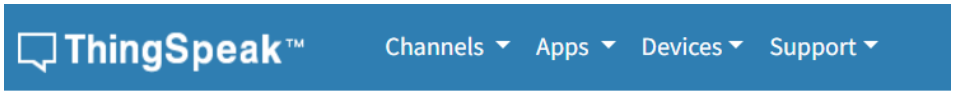
Smart Classroom:

Figure: Smart Classroom of University Campus

Smart Garden:

Figure: Smart Garden of University Campus

Real-Time Soil Moisture Data Value on ThingSpeak:



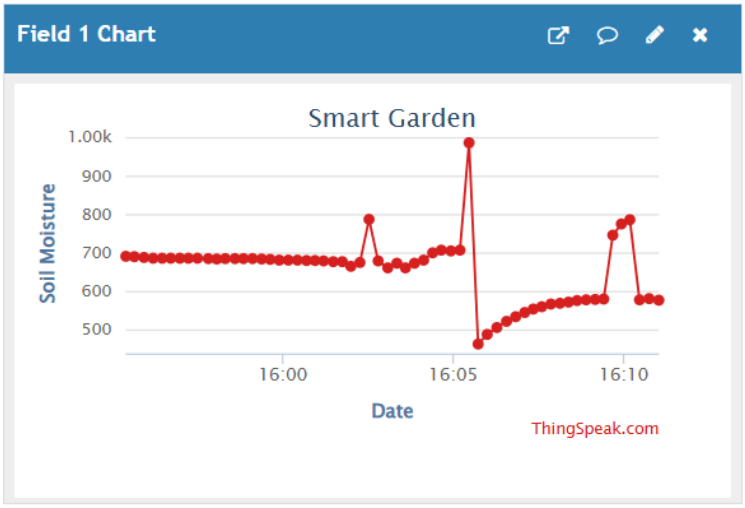


Figure: Data Chart of Soil-Moisture Value in Real-Time.

Smart Classroom and Smart Garden Integration on Blynk:

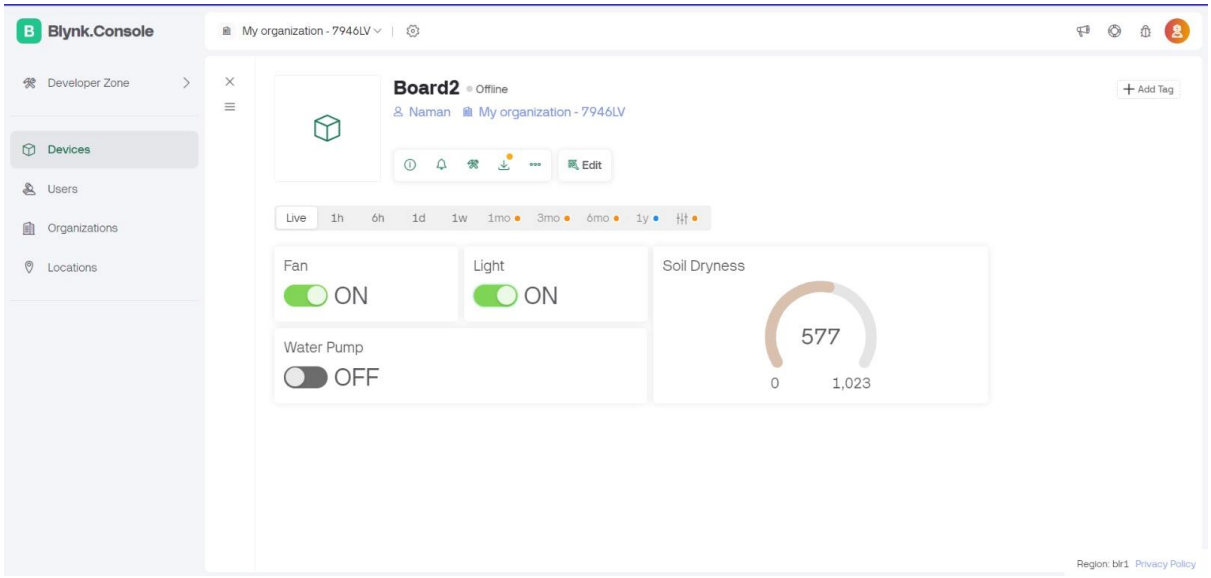
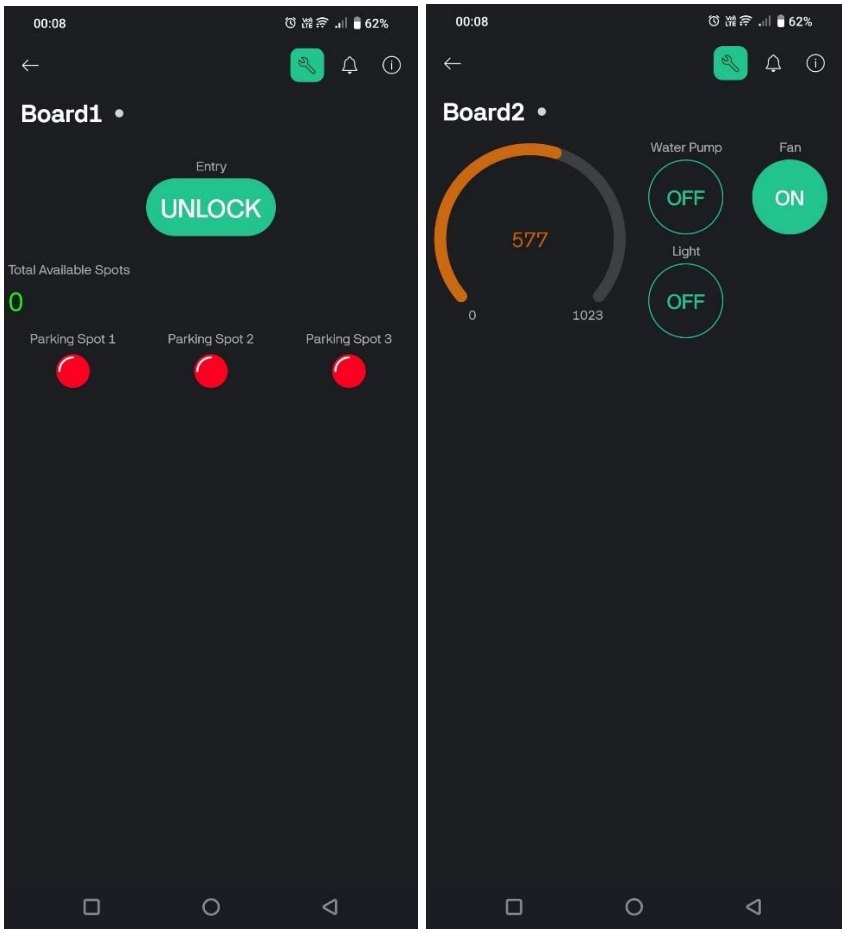


Figure: Dashboard of Smart Classroom and Smart Garden on Blynk

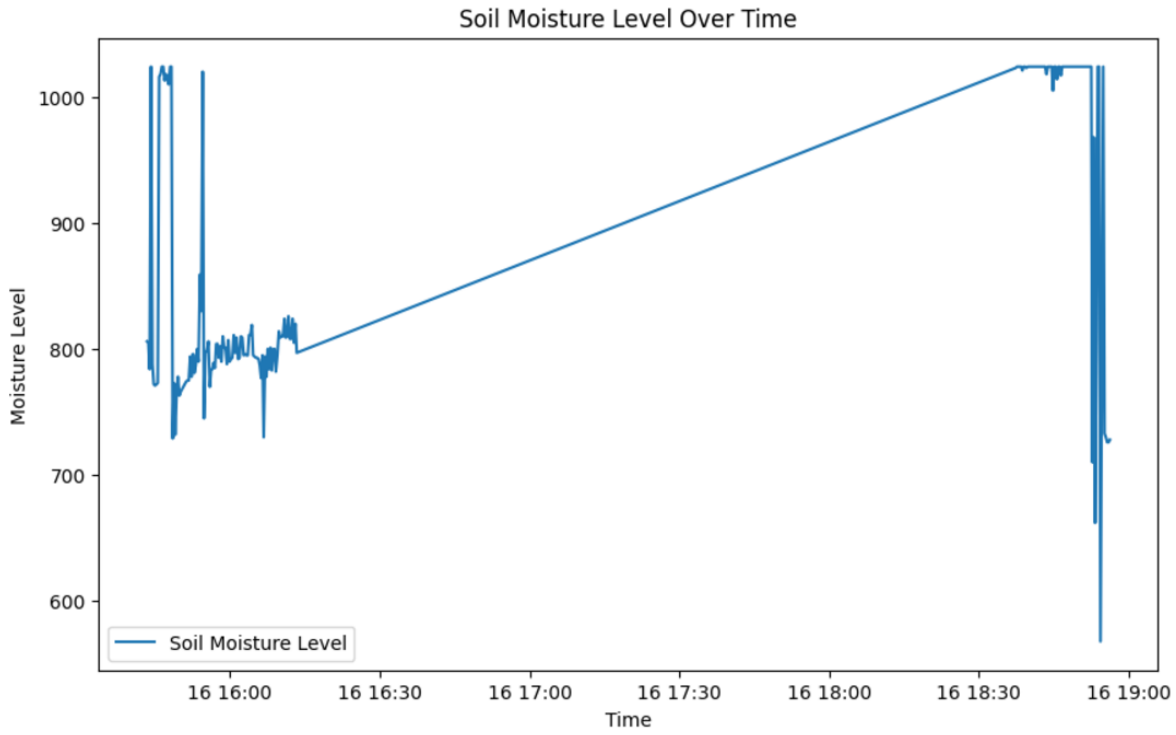
Smart University Campus Blynk Implementation on Smartphone:



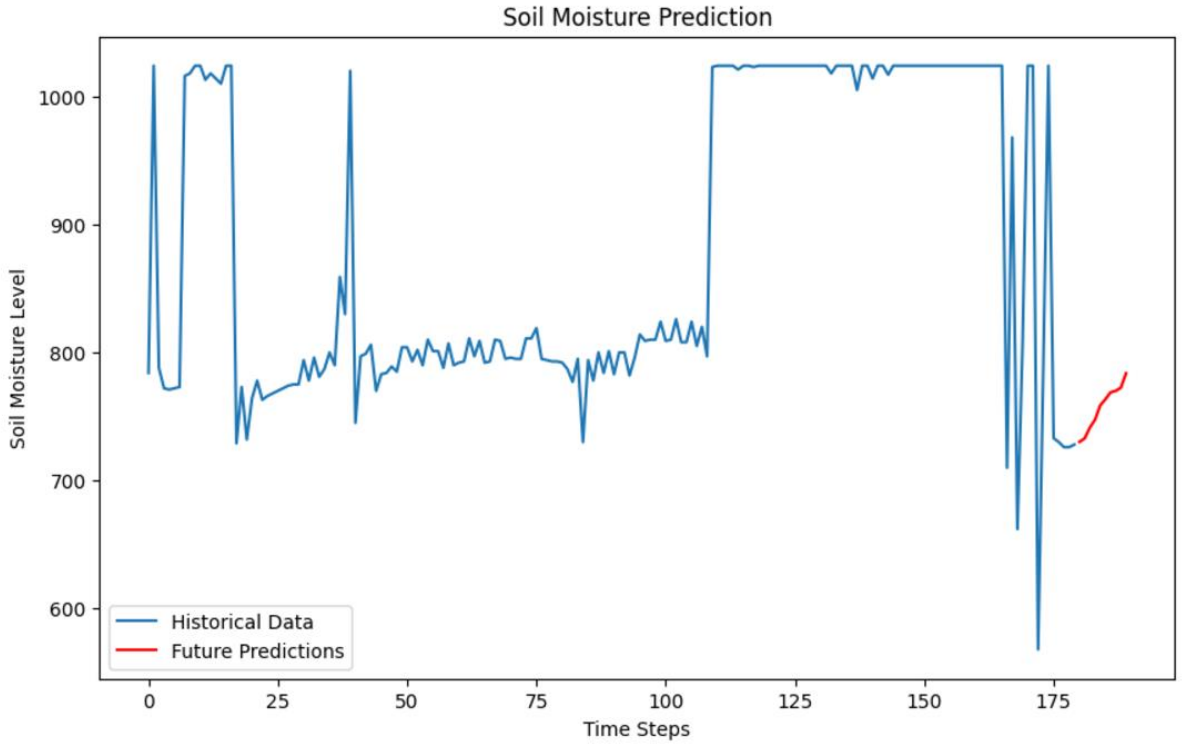
Figures: Blynk Monitoring and Control on Smartphone

Results After Applying Machine Learning to Soil Moisture Data Collected:

Obtained Soil Moisture Data Visualization:



Future Soil Moisture Value Prediction Using Random Forest Regressor:




```
Predicted Future Soil Moisture Levels: [730.16, 732.84, 741.28, 747.41, 758.59, 763.41, 768.83, 769.98, 772.43, 783.71]
```

Soil Condition Classification Using Decision Tree Classifier:

Classification Applied on Existing Data:

created_at	field1	condition
2024-11-16 15:43:55+05:30	784	Wet
2024-11-16 15:44:12+05:30	1024	Dry
2024-11-16 15:44:29+05:30	788	Wet
2024-11-16 15:44:45+05:30	772	Wet
2024-11-16 15:45:00+05:30	771	Wet

Soil Classification on Input Values:

```
# Example moisture level prediction
input_value = 1000  # Example moisture level
predicted_condition = le.inverse_transform(classifier.predict([[input_value]]))
print("Predicted Soil Condition:", predicted_condition)
```

```
Predicted Soil Condition: ['Dry']
```

```
# Example moisture level prediction
input_value = 800  # Example moisture level
predicted_condition = le.inverse_transform(classifier.predict([[input_value]]))
print("Predicted Soil Condition:", predicted_condition)
```

```
Predicted Soil Condition: ['Moderate']
```

```
# Example moisture level prediction
input_value = 500  # Example moisture level
predicted_condition = le.inverse_transform(classifier.predict([[input_value]]))
print("Predicted Soil Condition:", predicted_condition)
```

```
Predicted Soil Condition: ['Wet']
```

Evaluating the Accuracy of Classifier:

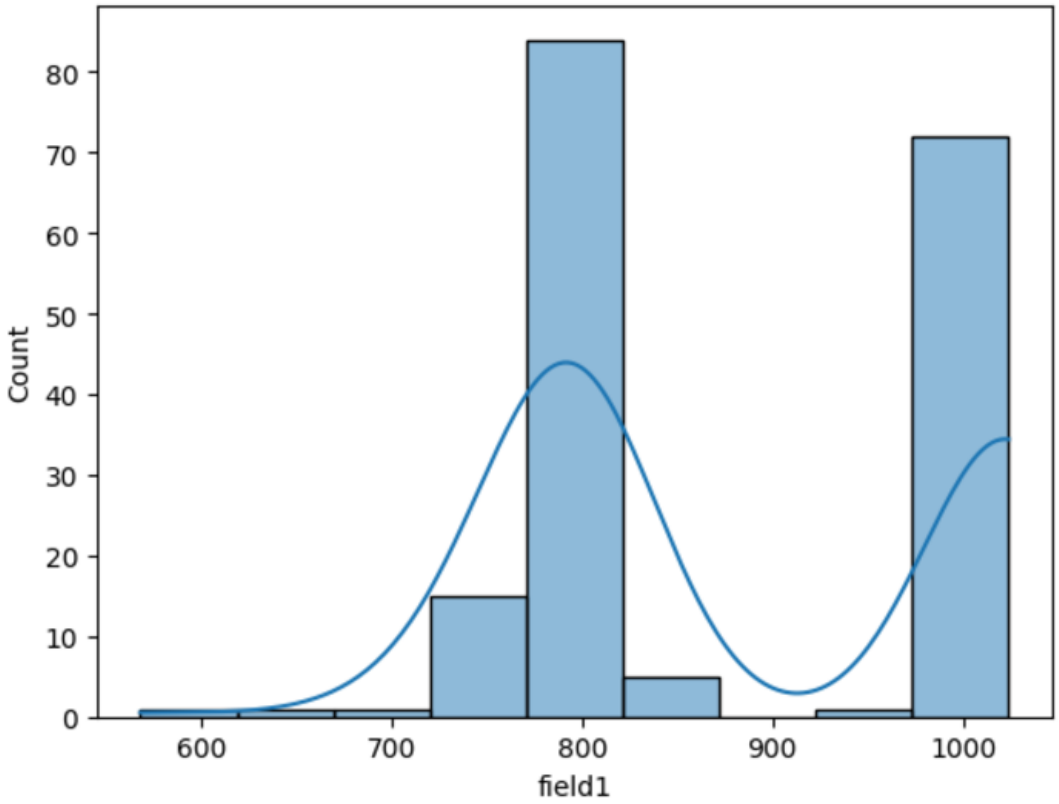
```
# Predict and evaluate
y_pred = classifier.predict(X_test)
from sklearn.metrics import accuracy_score
print("Classification Accuracy:", accuracy_score(y_test, y_pred))
```

```
Classification Accuracy: 0.9722222222222222
```

Soil classifier accuracy is very high as seen here leading to high reliability.

Anomaly Detection Using Isolation Forest Model:

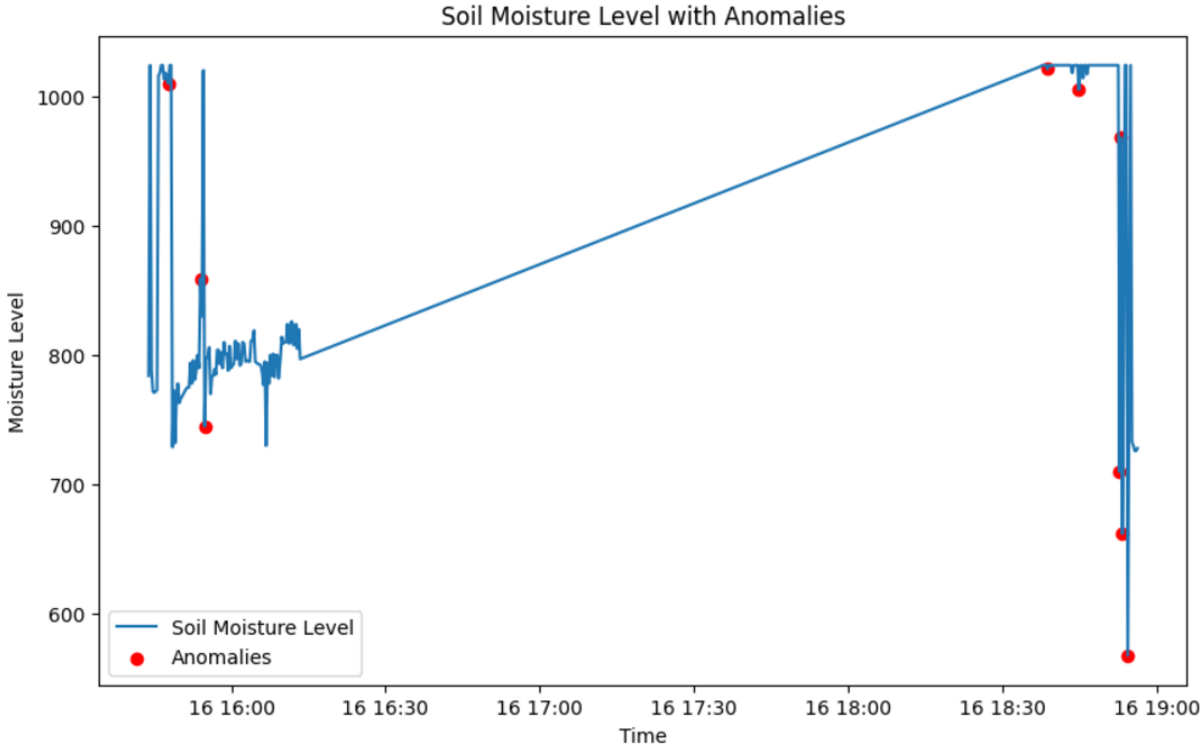
Data Distribution Visualizing:



Filtering and Printing Anomalies:

created_at	field1	lag_1	lag_2	condition	condition_encoded	anomaly
2024-11-16 15:47:44+05:30	1010	1014.0	1018.0	Dry	0	Anomaly
2024-11-16 15:54:00+05:30	859	790.0	800.0	Moderate	1	Anomaly
2024-11-16 15:54:49+05:30	745	1020.0	830.0	Wet	2	Anomaly
2024-11-16 18:38:41+05:30	1021	1024.0	1024.0	Dry	0	Anomaly
2024-11-16 18:44:48+05:30	1005	1024.0	1024.0	Dry	0	Anomaly
2024-11-16 18:52:44+05:30	710	1024.0	1024.0	Wet	2	Anomaly
2024-11-16 18:52:59+05:30	968	710.0	1024.0	Dry	0	Anomaly
2024-11-16 18:53:15+05:30	662	968.0	710.0	Wet	2	Anomaly
2024-11-16 18:54:19+05:30	568	1024.0	1024.0	Wet	2	Anomaly

Anomalies Visualized Over Time:



Processed Soil Data:

created_at	field1	lag_1	lag_2	condition	condition_encoded	anomaly
2024-11-16 18:52:13+05:30	1024	1024	1024	Dry	0	Normal
2024-11-16 18:52:28+05:30	1024	1024	1024	Dry	0	Normal
2024-11-16 18:52:44+05:30	710	1024	1024	Wet	2	Anomaly
2024-11-16 18:52:59+05:30	968	710	1024	Dry	0	Anomaly
2024-11-16 18:53:15+05:30	662	968	710	Wet	2	Anomaly
2024-11-16 18:53:31+05:30	807	662	968	Moderate	1	Normal
2024-11-16 18:53:46+05:30	1024	807	662	Dry	0	Normal
2024-11-16 18:54:02+05:30	1024	1024	807	Dry	0	Normal
2024-11-16 18:54:19+05:30	568	1024	1024	Wet	2	Anomaly
2024-11-16 18:54:35+05:30	788	568	1024	Wet	2	Normal
2024-11-16 18:54:52+05:30	1024	788	568	Dry	0	Normal
2024-11-16 18:55:08+05:30	733	1024	788	Wet	2	Normal
2024-11-16 18:55:24+05:30	730	733	1024	Wet	2	Normal
2024-11-16 18:55:40+05:30	726	730	733	Wet	2	Normal
2024-11-16 18:55:56+05:30	726	726	730	Wet	2	Normal
2024-11-16 18:56:12+05:30	728	726	726	Wet	2	Normal

The processed data has information of time created, soil moisture value, soil condition and anomaly if any.

The project successfully demonstrated the following outcomes:

- **Smart Gate Functionality**: The gate opens only for authorized RFID tags preventing unauthorized entry while providing manual control via Blynk.
- **Parking Management System**: Users can view real-time parking slots availability through an intuitive interface.
- **Classroom Environment Automation**: Lights and fans operate automatically based on occupancy detected by PIR sensors.
- **Cafeteria Fire Safety Alerts**: The system sends alerts by push notifications and email via Blynk when flames are detected in the cafeteria area.
- **Automated Garden Irrigation System**: Soil moisture levels are monitored, triggering irrigation only when necessary.
- Real-time monitoring & control achieved through Blynk & ThingSpeak.
- Predictive analysis and classification of soil moisture data using AI/ML techniques.

CHAPTER 10: LEARNING OUTCOMES

- **IoT System Design:** Gained practical hands-on experience in IoT and automation by integrating sensors and actuators with microcontrollers.
- **Technical Skills Development**: Gained proficiency in IoT hardware integration, programming with Arduino IDE, and using cloud platforms like ThingSpeak for real-world applications.
- **Programming Skills:** Developed proficiency in C for IoT applications.
- **Blynk Integration:** Learned to interface hardware with IoT dashboards for real-time control and monitoring.
- **ThingSpeak Integration:** Enhanced understanding of cloud platforms for data storage and monitoring.
- **Machine Learning Integration:** Developed skills in data analysis and machine learning using Python.
- **Problem-Solving:** Tackled issues with component compatibility, communication, and system optimization by applying critical thinking to devise innovative solutions for real-world challenges.**Project Management Experience**: Learned to manage a comprehensive project from conception through implementation while coordinating multiple components.
- **Team Collaboration:** Coordinated and collaborated effectively with peers to design and implement the project within deadlines leading to enhanced teamwork capabilities.

REFERENCES

1. MathWorks. *ThingSpeak IoT Analytics Platform.* Retrieved from https://thingspeak.mathworks.com/
2. Blynk IoT Platform. *Blynk for IoT Projects.* Retrieved from https://blynk.io/
3. IEEE Xplore. *IoT-Based Smart Campus: A Framework for Automation and Monitoring.* DOI: 10.1109/10637672
4. ScienceDirect. *Smart Campus Solutions Using IoT: A Review and Future Directions.* Retrieved from https://www.sciencedirect.com/science/article/pii/S2542660524000416
5. ResearchGate. *Internet of Things and Its Applications to Smart Campus: A Systematic Literature Review.* Retrieved from https://www.researchgate.net/publication/366153276_Internet_of_Things_and_Its_Applications_to_Smart_Campus_A_Systematic_Literature_Review
6. ResearchGate. *A Systematic Review of IoT in Smart University Model and Contribution.* Retrieved from https://www.researchgate.net/publication/364648243_A_Systematic_Review_of_the_IoT_in_Smart_University_Model_and_Contribution
7. Electronics Hub. *Getting Started with NodeMCU.* Retrieved from https://www.electronicshub.org/getting-started-with-nodemcu/
8. Circuits4You. *RFID Reader RC522 Interface with NodeMCU Using Arduino IDE.* Retrieved from https://circuits4you.com/2018/10/03/rfid-reader-rc522-interface-with-nodemcu-using-arduino-ide/
9. IT Law Wiki. *RFID Tag.* Retrieved from https://itlaw.fandom.com/wiki/RFID_tag
10. STEMpedia AI. *Interfacing IR Sensor with Arduino.* Retrieved from https://ai.thestempedia.com/example/interfacing-ir-sensor-with-arduino/
11. SolArduino. *Interfacing Flame Sensor with Arduino.* Retrieved from https://solarduino.com/cgi-sys/suspendedpage.cgi
12. Circuit Digest. *Interfacing Flame Sensor with Arduino.* Retrieved from https://circuitdigest.com/microcontroller-projects/interfacing-flame-sensor-with-arduino

13. Robu.in. *5V Passive Buzzer.* Retrieved from https://robu.in/product/5v-passive-buzzer/

14. WLED Knowledge Base. *PIR Sensors.* Retrieved from https://kno.wled.ge/advanced/pir-sensors/

15. Circuit Digest. *Interfacing Soil Moisture Sensor with Arduino UNO.* Retrieved from https://circuitdigest.com/microcontroller-projects/interfacing-soil-moisture-sensor-with-arduino-uno

16. Electrical4U. *LED (Light Emitting Diode).* Retrieved from https://www.electrical4u.com/led-or-light-emitting-diode/

17. Components101. *5V Single Channel Relay Module.* Retrieved from https://components101.com/switches/5v-single-channel-relay-module-pinout-features-applications-working-datasheet

18. IndiaMart. *DC Toy Motors Micro Coreless Motor.* Retrieved from https://www.indiamart.com/proddetail/dc-toy-motors-micro-coreless-motor-11613135862.html

19. Sharvielectronics. *12V High-Quality DC Mini Submersible Pump.* Retrieved from https://sharvielectronics.com/product/12v-high-quality-dc-mini-submersible-pump/

20. Fritzing. *Fritzing Software.* Retrieved from https://fritzing.org/

21. Arduino. *Arduino Software (IDE).* Retrieved from https://www.arduino.cc/en/software

APPENDIX

Board1 Code (Smart Gate and Smart Parking):

```
#define BLYNK_TEMPLATE_ID "TMPL376McIYI9"
#define BLYNK_TEMPLATE_NAME "Board1"
#define BLYNK_AUTH_TOKEN "gO8N55BcwPtfmexfu2e1-jDYoQcIXpap"

#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#define RST_PIN         D3
#define SS_PIN          D4
#define SERVO_PIN       D1

#define IR_SENSOR1_PIN  D0
#define IR_SENSOR2_PIN  D2
#define IR_SENSOR3_PIN  D8

MFRC522 rfid(SS_PIN, RST_PIN);
Servo myServo;

byte validUID[4] = {0xA1, 0x94, 0x1C, 0x1B};

char auth[] = BLYNK_AUTH_TOKEN;
```

```
char ssid[] = "vivo Y35";
char pass[] = "24262426";

bool parkingSpot1Occupied = false;
bool parkingSpot2Occupied = false;
bool parkingSpot3Occupied = false;
int availableSpots = 3;

bool doorLocked = true;
unsigned long unlockTime = 0;
const unsigned long AUTO_LOCK_DELAY = 2000;
bool manualUnlock = false;

void connectToWiFi() {
  WiFi.begin(ssid, pass);
  Serial.print("Connecting to Wi-Fi");
  int retryCount = 0;
  while (WiFi.status() != WL_CONNECTED && retryCount < 20) {
    delay(500);
    Serial.print(".");
    retryCount++;
  }
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("Connected to Wi-Fi!");
  } else {
    Serial.println("Failed to connect to Wi-Fi.");
  }
```

```
}

void connectToBlynk() {
  Blynk.config(auth);
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("Connecting to Blynk...");
    Blynk.connect();
    int retryCount = 0;
    while (!Blynk.connected() && retryCount < 20) {
      delay(500);
      Serial.print(".");
      retryCount++;
    }
    if (Blynk.connected()) {
      Serial.println("Connected to Blynk!");
    } else {
      Serial.println("Failed to connect to Blynk.");
    }
  }
}

void setup() {
  Serial.begin(115200);

  SPI.begin();
  rfid.PCD_Init();
  myServo.attach(SERVO_PIN);
```

```
  myServo.write(0);

  pinMode(IR_SENSOR1_PIN, INPUT);
  pinMode(IR_SENSOR2_PIN, INPUT);
  pinMode(IR_SENSOR3_PIN, INPUT);

  Serial.println("Initializing...");

  connectToWiFi();
  connectToBlynk();

  Blynk.virtualWrite(V6, "Locked");
  Serial.println("Place RFID card and monitor parking spots.");
}

void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    connectToWiFi();
  }
  if (!Blynk.connected()) {
    connectToBlynk();
  }

  Blynk.run();
  checkParkingSpots();

  if (!doorLocked && !manualUnlock && (millis() - unlockTime >=
AUTO_LOCK_DELAY)) {
```

```
    lockDoor();
  }
  if (doorLocked && rfid.PICC_IsNewCardPresent() &&
rfid.PICC_ReadCardSerial()) {
    if (checkUID()) {
      unlockDoor();
      Blynk.virtualWrite(V1, 1);
    }
    rfid.PICC_HaltA();
  }
}

void checkParkingSpots() {
  parkingSpot1Occupied = digitalRead(IR_SENSOR1_PIN) == LOW;
  parkingSpot2Occupied = digitalRead(IR_SENSOR2_PIN) == LOW;
  parkingSpot3Occupied = digitalRead(IR_SENSOR3_PIN) == LOW;
  availableSpots = 3 - (parkingSpot1Occupied + parkingSpot2Occupied +
parkingSpot3Occupied);

  Blynk.virtualWrite(V2, parkingSpot1Occupied ? 1 : 0);
  Blynk.virtualWrite(V3, parkingSpot2Occupied ? 1 : 0);
  Blynk.virtualWrite(V4, parkingSpot3Occupied ? 1 : 0);
  Blynk.virtualWrite(V5, availableSpots);
}

bool checkUID() {
  for (byte i = 0; i < 4; i++) {
    if (rfid.uid.uidByte[i] != validUID[i]) return false;
```

```
  }
  return true;
}

void unlockDoor() {
  myServo.write(90);
  doorLocked = false;
  unlockTime = millis();
  Blynk.virtualWrite(V6, "Unlocked");
}

void lockDoor() {
  myServo.write(0);
  doorLocked = true;
  Blynk.virtualWrite(V6, "Locked");
  Blynk.virtualWrite(V1, 0);
}

BLYNK_WRITE(V1) {
  manualUnlock = param.asInt();
  if (manualUnlock) {
    unlockDoor();
  } else {
    lockDoor();
  }
}
```

Board2 Code (Smart Classroom, Smart Kitchen and Smart Garden):

```
#define BLYNK_TEMPLATE_ID "TMPL3o34QEZZf"
#define BLYNK_TEMPLATE_NAME "Board2"
#define BLYNK_AUTH_TOKEN "qSk0ysoLGAyQEWk7ouVQVvFEZZndlUNl"

#include <BlynkSimpleEsp8266.h>
#include <ESP8266WiFi.h>
#include <ThingSpeak.h>

#define BLYNK_PRINT Serial

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "vivo Y35";
char pass[] = "24262426";

WiFiClient client;
unsigned long myChannelNumber = 2748636;
const char *myWriteAPIKey = "GTO1VLJK5UO7AB6V";

int soilMoistureSensor = A0;
int pump_relay = D3;
int WET = D1;
int DRY = D2;
int PIRSensor = D5;
int Motor_Relay = D6;
int led = D8;
int flameSensor = D4;
```

```
int buzzer = D0;

bool pumpControl = false;
bool motorControl = false;
bool lightControl = false;
bool flameAlertSent = false;

void setup() {
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
  ThingSpeak.begin(client);

  pinMode(soilMoistureSensor, INPUT);
  pinMode(pump_relay, OUTPUT);
  pinMode(WET, OUTPUT);
  pinMode(DRY, OUTPUT);
  pinMode(PIRSensor, INPUT);
  pinMode(Motor_Relay, OUTPUT);
  pinMode(led, OUTPUT);
  pinMode(flameSensor, INPUT);
  pinMode(buzzer, OUTPUT);

  digitalWrite(pump_relay, HIGH);
  digitalWrite(Motor_Relay, HIGH);
  digitalWrite(WET, LOW);
  digitalWrite(DRY, LOW);
  digitalWrite(led, LOW);
```

```
  digitalWrite(buzzer, LOW);
}

BLYNK_CONNECTED() {
  Blynk.syncVirtual(V1, V2, V3);
}

BLYNK_WRITE(V1) {
  pumpControl = param.asInt();
  digitalWrite(pump_relay, pumpControl ? LOW : HIGH);
}

BLYNK_WRITE(V2) {
  motorControl = param.asInt();
  digitalWrite(Motor_Relay, motorControl ? LOW : HIGH);
}

BLYNK_WRITE(V3) {
  lightControl = param.asInt();
  digitalWrite(led, lightControl ? HIGH : LOW);
}

void loop() {
  Blynk.run();

  int flameDetected = digitalRead(flameSensor);
  digitalWrite(buzzer, flameDetected == HIGH ? LOW : HIGH);
```

```
  if (flameDetected == LOW && !flameAlertSent) {
    Blynk.logEvent("flame_alert", "🔥 Flame detected! Take immediate action!");
    flameAlertSent = true;
  } else if (flameDetected == HIGH) {
    flameAlertSent = false;
  }

  int moistureLevel = analogRead(soilMoistureSensor);
  Blynk.virtualWrite(V0, moistureLevel);

  ThingSpeak.setField(1, moistureLevel);
  ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

  if (!pumpControl) {
    if (moistureLevel < 900) {
      digitalWrite(pump_relay, HIGH);
      digitalWrite(DRY, LOW);
      Blynk.virtualWrite(V1, 0);
    } else {
      digitalWrite(pump_relay, LOW);
      digitalWrite(WET, LOW);
      digitalWrite(DRY, HIGH);
      Blynk.virtualWrite(V1, 1);
    }
  }
```

```
  bool motionDetected = digitalRead(PIRSensor);
  if (!lightControl) {
    if (motionDetected) {
      digitalWrite(led, HIGH);
      Blynk.virtualWrite(V3, 1);
    } else {
      digitalWrite(led, LOW);
      Blynk.virtualWrite(V3, 0);
    }
  }

  if (!motorControl) {
    if (motionDetected) {
      digitalWrite(Motor_Relay, LOW);
      Blynk.virtualWrite(V2, 1);
    } else {
      digitalWrite(Motor_Relay, HIGH);
      Blynk.virtualWrite(V2, 0);
    }
  }

  delay(500);
}
```

Code of Machine Learning:

https://colab.research.google.com/drive/1DoQGLcnTJzpIey1PPJsZTX4wzKMFozXe?usp=sharing