DESIGN AND IMPLEMENTATION OF CLOUD PLATFORM FOR SMART HOME AUTOMATION

Abstract

The design and implementation of a cloud platform for IOT devices is presented in this report. This platform is specifically developed to provide the user with a portal to control the physical devices anywhere from the geo or globe. The system utilizes a 5V relay module typically consists of a relay, driver circuitry, and supporting components. The NodeMCU which is an open-source development board based on the ESP8266 Wi-Fi module and microcontroller unit (MCU) with built-in Wi-Fi connectivity. This methodology consists of Ac supply which is been connected to the bulb and the relay parallelly and web application which gives the user to take input and the following inputs are being converted to actions. This platform for IOT devices is an innovative and sustainable solution for controlling the status of the electrical equipment's overcoming the challenges such as manual switching and failure in data analysis. This platform which also supports the application from smartphones makes it much more flexible in usage. The successful implementation of this projecthas a great capability for improving the IoT application and has much more future prospect on the same.

Keywords: This platform for IOT devices is an innovative and sustainable solution for controlling.

Authors

Gayathri R
Assistance Professor
MVJ College of Engineering
Balachandra Layout
Babusabpalya
Kalyan Nagar post
Bangalore, Karnataka, India.

I. INTRODUCTION

The idea of a network of smart devices had been previously discussed as far back as 1982, but it was Peter T. Lewis who coined the term "Internet of things" in a speech. The Internet of things (IoT) refers to objects that are equipped with sensors, processing capability, software, and other technologies. These features allow them to establish connections and exchange data with other devices and systems through the Internet or other communication networks. As home automation continues to become a reality with the proliferation of IoT, numerous players such as Apple, Amazon, Google, and Samsung are converging in this space to provide platforms and solutions for smart homes. With this in mind, the current study seeks to explore IoT concepts through a systematic review of academic research papers, company white papers,expert discussions with specialists, and online databases.

In [1], the author reviews the fundamental concepts of IoT. Further, the paper presents an infrastructure for the healthcare domain, which consists of various technologies: IOIO microcontroller, signal processing, communication protocols, secure and efficient mechanisms for large file transfer, data base management system, and the centralized cloud. The paper emphasizes on the system and software architecture and design which is essential to overall IoT and cloud based medical applications. The infrastructure presented in the paper can also be applied to other healthcare domains. It concludes with recommendations and extensibilities found for the solution in the healthcare domain. In [2] the authors present a comprehensive and thorough survey of ICT-enabled waste management models. Specifically, we focus on the adoption of smart devices as a key enabling technology in contemporary waste management. We report on the strengths and weaknesses of various models to reveal their characteristics. This survey sets up the basis for delivering new models in the domain as it reveals the needs for defining novel frameworks for waste management. In [3-6] The longitudinal learning system was able to provide a self-control mechanism for better operation of the devices in monitoring stage. The framework of the monitoring system is based on a combination of pervasive distributed sensing units, information system for data aggregation, and reasoning and context awareness. Results are encouraging as the reliability of sensing information transmission through the proposed integrated network architecture is very good. In [7-10], SCADA based system is used for monitoring. Even though it is an effective way of monitoring, the cost involved discourages the users to adapt these systems.

From the literature survey, we found that, with the rise in the number of connected devices, there is a growing need to create a cloud platform powered by Python to manage IoT devices. IoT and cloud computing have emerged as a solution to address issues such as management, aggregation, and storage for large-scale IoT platforms. However, one potential drawback of this rapid advancement is the unregulated proliferation of heterogeneous sensor networks that lack interoperability features, even when deployed within the same context. Ensuring the security and privacy of IoT projects is crucial. However, protecting the data collected and transmitted by IoT devices can be a challenging task, especially as their use continues to grow and evolve. Simplification and increased understanding are necessary to overcome these challenges. The current IoT environment is highly dependent on power and integration, which can result in increased costs in terms of time and money.

To address these drawbacks, there is a strong motivation to develop an environment that

can overcome these limitations. The primary goal of this IoT platform is to eliminate the barriers among various layers and integrate them to achieve effective and smooth collaboration. The IoT platform facilitates a flexible connection between hardware and the cloud while ensuring security measures and comprehensive data processing capabilities. The IoT platform is solely used for following features: Authentication, Authorization, Accessibility Develop Web application for accessing the cloud, Interface between IOT device and cloud Users need to provide their username and password for authorization in order to retrieve data from the server. The registration process of IoT devices includes authorization and verification to ensure security. Authorization is important to restrict access to the IoT platform to authorized personnel only. Verification of IoT devices includes validating their model, IP address, and MAC address to ensure their authenticity. The proposed system also enables the discovery and sharing of registered IoT devices. A web-based interface is provided by the IoT platform for registered users to add and manage their IoT devices. To access the current status of the devices, a web application has been developed that connects to the server where the device data is stored.

II. SYSTEM DESCRIPTION

An IoT platform refers to a network that facilitates the interconnection, administration, and communication between sensors and devices via the internet. Its functionalities usually encompass device management, data aggregation and analysis, as well as the capacity to create and execute rules and actions based on the collated data. Furthermore, an IoT platform should be able to accommodate growth, maintain high security standards, and support interoperability with other devices and systems. It should be capable of handling substantial amounts of data and integrating with various data sources, including cloud storage, to offer real-time analytics and insights.

Key Components

- **Device Management: -**The registration, provisioning, and management of devices and sensors connected to the IoT platform is facilitated by this component. It enables remote device management, over-the-air updates, and monitoring of device health and status.

- **Data Collection and Analysis:** The data management component of an IoT platform facilitates the collection and storage of data generated by connected devices and sensors. This component also includes the necessary tools for data analysis and visualization, such as real-time data streaming, data storage, and data visualization tools.

- **Security:** Security is an essential aspect of an IoT platform, and it must have strong measures in place to prevent security breaches, unauthorized access, and other security risks. Such measures may include secure communication protocols, device authentication, and data encryption.

- **Scalability and Interoperability:** The ability to handle many devices and data is a crucial aspect of an IoT platform. Moreover, it should be capable of integrating with

various systems and devices and connecting to different data sources such as cloud storage. The platform should support various devices and protocols for efficient data exchange.

- **User Interface and Management:** The interface of the platform should be intuitive and easy to use, providing users with convenient access to manage devices, sensors, and data.

1. **Hardware Description:** The hardware description of an IoT platform encompasses the physical components and infrastructure required to support the platform's functionality. While the specific hardware setup may vary depending on the scale and specific use cases of the IoT platform, there are several common elements to consider. Here is a general hardware description for an IoT platform:

- **IoT Devices:** These are the physical devices that connect to the IoT platform and collect or transmit data. They can include sensors, actuators, smart appliances, industrial machinery, or any other device capable of interacting with the physical environment. IoT devices may vary in their communication protocols, power requirements, and form factors, depending on the specific application and industry.

- **Gateway Devices:** In some IoT deployments, gateway devices are used to aggregate and preprocess data from multiple IoT devices before transmitting it to the IoT platform. Gateways act as intermediaries, facilitating communication between devices and the cloud infrastructure. They may perform data filtering, protocol translation, or local data processing tasks. Gateway devices can be hardware-based or software-based, depending on the requirements of the IoT platform.

- **Network Infrastructure:** The IoT platform relies on a robust and reliable network infrastructure to facilitate communication between devices, gateways, and the central cloud-based platform. This infrastructure can include wired connections (Ethernet, Powerline Communication) or wireless technologies (Wi-Fi, Bluetooth, Zigbee, LoRaWAN, cellular networks) depending on factors such as device proximity, data transmission requirements, and the environment in which the IoT platform operates.

- **Cloud Infrastructure:** The IoT platform typically leverages cloud-based infrastructure for storing, processing, and analyzing data generated by IoT devices. This infrastructure may consist of servers, databases, and scalable computing resources hosted on public or private cloud platforms. Cloud infrastructure provides the necessary computational power, storage capacity, and scalability to handle large volumes of data and perform real-time analytics.

- **Storage Systems:** To store and manage the data generated by IoT devices, the IoT platform may utilize various storage systems. This can include databases such as SQL (e.g., MySQL, PostgreSQL) or NoSQL (e.g., MongoDB, Cassandra) databases, as well as distributed file systems like Hadoop Distributed File System (HDFS) or Amazon S3. The choice of storage systems depends on factors such as data structure, volume, access patterns, and the analytical requirements of the IoT platform.

- **Security Systems:** Security is a critical aspect of an IoT platform, and the hardware setup should include measures to protect data integrity, confidentiality, and availability. This may involve hardware security modules (HSMs) for cryptographic operations, secure elements or trusted platform modules (TPMs) for device authentication and encryption key management, firewalls, intrusion detection systems, and secure communication protocols (TLS/SSL) to protect data transmission between devices and the cloud infrastructure.

- **Power Management:** IoT devices often operate on limited power sources or rely on battery power. The IoT platform may include power management mechanisms to optimize energy consumption and extend device battery life. This can involve power management ICs, energy harvesting techniques, or low-power design considerations for devices and gateways.

It's important to note that the hardware description of an IoT platform can vary significantly based on specific use cases, industry requirements, and scalability needs. The described components provide a general overview of the hardware infrastructure needed to support an IoT platform, but the actual implementation may involve additional components or customization based on the unique requirements of the platform.

2. **Software Description:** The software description of an IoT platform encompasses the various software components and applications that enable the functionality, management, and integration of IoT devices and data. Here is a comprehensive software description for an IoT platform:

    - **Operating System:** The IoT platform may require an operating system that provides the necessary infrastructure for device connectivity, data management, and application execution. Depending on the specific requirements, the platform can leverage lightweight operating systems such as Linux-based distributions (e.g., Raspbian, Ubuntu Core) or real-time operating systems (RTOS) tailored for embedded systems.

    - **Device Connectivity and Protocols:** The IoT platform relies on software protocols and libraries to establish seamless communication between devices and the platform infrastructure. This includes protocols such as MQTT (Message Queuing Telemetry Transport), CoAP (Constrained Application Protocol), or HTTPS (Hypertext Transfer Protocol) for device-to-platform communication. Software libraries and frameworks enable device discovery, provisioning, and secure authentication (e.g., OAuth, JWT) to ensure the integrity and confidentiality of data transmission.

    - **Data Ingestion and Processing:** The IoT platform requires software components to ingest, process, and store data generated by IoT devices. This includes data ingestion systems that can handle real-time data streams and batch processing frameworks (e.g., Apache Kafka, Apache Spark) for processing large volumes of data. Additionally, software components for data transformation, filtering, and aggregation are essential to extract meaningful insights from raw sensor data.

- **Data Storage and Databases:** The IoT platform requires software systems and databases to store and manage the collected IoT data. This may involve the utilization of relational databases (e.g., MySQL, PostgreSQL) or NoSQL databases (e.g., MongoDB, Cassandra) depending on the data structure, volume, and access patterns. Additionally, distributed file systems (e.g., Hadoop Distributed File System, Amazon S3) or time-series databases (e.g., InfluxDB, Prometheus) may be employed for efficient storage and retrieval of time-stamped sensor data.

- **Analytics and Machine Learning:** The IoT platform can leverage software components for advanced analytics and machine learning to derive insights, detect patterns, and make predictions from the collected data. This includes statistical analysis libraries (e.g., NumPy, SciPy), machine learning frameworks (e.g., scikit-learn, TensorFlow), and data visualization tools (e.g., Matplotlib, Plotly) to enable descriptive, diagnostic, predictive, and prescriptive analytics on IoT data.

- **Application Development and Integration:** The IoT platform provides software development tools, frameworks, and APIs that enable the creation of custom applications and integration with existing systems. This includes web development frameworks (e.g., Django, Flask) for building user interfaces and APIs, software development kits (SDKs) or APIs for device integration, and development environments that facilitate rapid prototyping and deployment of IoT applications.

- **Security and Authentication:** Security is a critical aspect of an IoT platform, and software components are necessary to ensure secure device connectivity, data transmission, and user authentication. This includes software libraries for implementing secure communication protocols (e.g., TLS/SSL), encryption algorithms, and secure authentication mechanisms (e.g., OAuth, JWT). Intrusion detection systems, firewalls, and vulnerability assessment tools can also be employed to enhance the security of the platform.

- **Scalability and Orchestration:** The IoT platform should be designed to scale seamlessly as the number of devices and data volume increases. Software components for distributed computing, containerization technologies (e.g., Docker, Kubernetes), and orchestration frameworks (e.g., Apache Mesos, Apache ZooKeeper) enable efficient resource allocation, load balancing, and horizontal scaling to handle the growing demands of the IoT ecosystem.

3. **Front-End Web Development:** Front-end web development is an essential aspect of creating engaging and interactive websites. This comprehensive report explores the core technologies used in front-end development, namely HTML, CSS, and JavaScript. It provides an in-depth analysis of each language, their functionalities, and how they work together to build modern web interfaces. The report discusses the role of HTML in structuring web content, CSS in styling and layout, and JavaScript in adding interactivity and dynamic behavior. Additionally, it highlights best practices, emerging trends, and the future outlook of front-end web development. Front-end web development involves the creation of user interfaces for websites and web applications. It encompasses HTML,

CSS, and JavaScript, the three fundamental languages that work together to deliver a seamless user experience.

- **HTML (Hyper Text Markup Language):** HTML is the foundation of every web page. It provides the structure and semantics for web content, defining the elements and their relationships. Key points about HTML include:

  - **Structure and Tags:** HTML uses a tag-based structure to define elements such as headings,paragraphs, lists, images, and links.
  - **Document Structure:** HTML provides a hierarchical structure with tags like <html>,
  - <head>, and <body>, ensuring proper organization and readability of web content.
  - **Semantics:** HTML5 introduced semantic elements such as <header>, <nav>, <article>, and
  - <footer>, which provide more meaningful structure and improve accessibility.
  - **Forms and Input:** HTML includes form elements like <input>, <select>, and <textarea> foruser input and data submission.

- **CSS (Cascading Style Sheets):** CSS is responsible for the visual presentation and layout of web content. It allows developers to style HTML elements, define colors, fonts, layouts, and animations. Key points about CSS include:

  - **Selectors and Rules:** CSS uses selectors to target specific HTML elements and apply stylingrules to them.
  - **Box Model:** CSS employs the box model to determine how elements are sized and spaced,consisting of content, padding, borders, and margins.
  - **Flexbox and Grid:** CSS introduces flexible box layouts (flexbox) and grid layouts (grid),enabling responsive and dynamic designs.
  - **Media Queries:** CSS media queries allow developers to apply different styles based on thedevice's screen size, enabling responsive design for various devices.

- **JavaScript:**JavaScript is a powerful scripting language that adds interactivity and dynamic behavior to web pages. It enables tasks such as form validation, event handling, DOM manipulation, and asynchronous operations. Key points about JavaScript include:

  - **DOM Manipulation:** JavaScript interacts with the Document Object Model (DOM), allowing developers to modify HTML elements, add or remove content dynamically, and respond to user actions.
  - **Event Handling:** JavaScript handles user interactions through event listeners, responding toactions like button clicks, form submissions, or mouse movements.
  - **AJAX and Fetch:** JavaScript facilitates asynchronous communication with servers using techniques like AJAX and the Fetch API, enabling dynamic content loading without page refresh.
  - **Libraries and Frameworks:** JavaScript has a vast ecosystem of libraries and frameworks like React, Angular, and Vue.js that simplify complex web development tasks and enhance productivity.

4. **Back-End Web Development:** Django 4.2.1, a high-level Python web framework, coupled with Python 3.1.0, offers a robust and efficient solution for back-end web development. This comprehensive report explores the key features, advantages, and functionalities of Django and Python in building scalable and secure web applications. It delves into Django's architecture, its powerful ORM (Object-Relational Mapping) capabilities, and the versatility of Python for implementing business logic. The report also highlights the benefits of using Django and Python together, along with best practices, community support, and emerging trends in back-end web development. Back-end web development involves building the server-side components of web applications. Django, a Python web framework, in combination with Python programming language, provides a comprehensive solution for developing efficient and scalable back-end systems.

   - **Django Framework:** Django is a high-level, open-source web framework that follows the model-view- controller (MVC) architectural pattern. Key points about Django include

     - **MVC Architecture:** Django's MVC architecture separates the application logic into models (data representation), views (user interface), and controllers (business logic).
     - **Object-Relational Mapping (ORM):** Django's built-in ORM simplifies database management by mapping Python objects to database tables, making it easier to perform CRUD(Create, Read, Update, Delete) operations.
     - **URL Routing:** Django's URL routing mechanism directs incoming requests to the appropriate views, allowing developers to define clean and maintainable URL structures.
     - **Template Engine:** Django includes a powerful template engine that facilitates the renderingof dynamic content, separating presentation from logic.
     - **Authentication and Authorization:** Django provides robust authentication and authorization mechanisms, allowing developers to secure their web applications with ease.
     - **Administration Interface:** Django offers an automatic admin interface that enables developers to manage and manipulate data models without writing additional code.
     - **Scalability and Performance:** Django's design principles, such as caching, database query optimization, and middleware support, contribute to building scalable and performant web applications.

   - **Python Programming Language:** Python, a versatile and powerful programming language, serves as the foundation for Django. Key points about Python include:

     - **Readability and Simplicity:** Python's clean syntax and code readability make it easy to writeand maintain code, reducing development time.
     - **Large Standard Library:** Python's extensive standard library provides a wide range of pre- built modules and functions, enabling rapid development without reinventing the wheel.
     - **Integration Capabilities:** Python seamlessly integrates with other technologies

and frameworks, making it suitable for building complex back-end systems that incorporateexternal services and APIs.

- **Robust Ecosystem:** Python has a thriving ecosystem with a vast collection of third-party libraries and frameworks, providing additional functionality and facilitating development across different domains.

Benefits of Django and Python in Back-End Web Development

- **Rapid Development:** Django's code organization, built-in features, and Python's expressivesyntax allow for faster development, reducing time to market.
- **Scalability and Flexibility:** Django's scalability features, combined with Python's versatility,support the development of scalable and adaptable back-end systems.
- **Security:** Django's built-in security features, along with Python's emphasis on writing securecode, help protect web applications against common vulnerabilities.
- **Community Support and Documentation:** Django and Python have large and active communities that offer support, share knowledge, and contribute to the continuous improvement of the frameworks.
- **Testing and Debugging:** Django's testing framework and Python's extensive testing librariesenable efficient testing and debugging of web applications.

III. IMPLEMENTATION

1. **Log-In:** The login form/sign-in form is crucial for accessing websites or web applications and ensuring data security. Upon successful login, the control panel form is displayed, enabling users to modify the switch status.

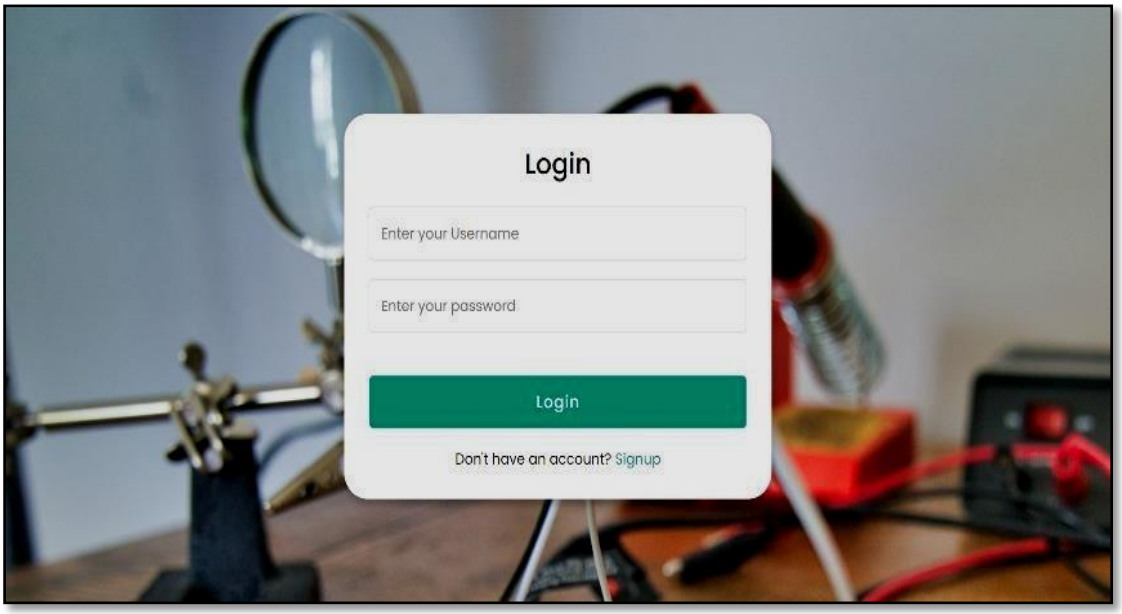


Figure 1: Login Page

The purpose of a login page is to grant users access to an application or website by providing their login credentials, such as a username and password. Additionally, some

login pages may offer the option to authenticate using a social media account, such as Google. The Login page requires the user to enter their credentials, which are then verified by the application. After successful validation, the user gains access to the secure part of the application.

2. **Sign-Up:** This feature offers users a convenient and streamlined registration process without the need to create a new account and fill in all the required information. By simply clicking on the sign-up button, users can easily connect with the application, which makes them feel more at ease. The password should be at least 8 characters long, but having more characters can provide better security. Using a combination of uppercase letters, lowercase letters, numbers, and symbols can make the password more complex and secure.

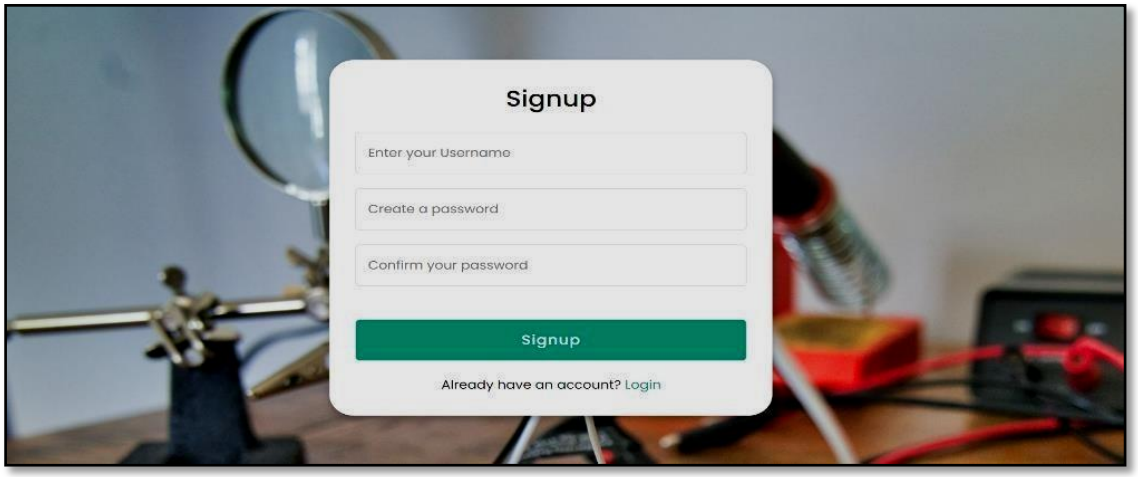


Figure 2: Sign Up Page

Avoid using words found in a dictionary. The new password must be significantly different from the previous passwords to ensure better security. Choose a username that is easy to remember but difficult to guess. To create a secure username, avoid using easily guessable numbers such as your address or date of birth, and do not use your social security or ID number. Upon submitting the signup form, a confirmation email will be sent to the email addressprovided.

3. **Home Page:** The first impression of your load control web application is crucial as it can influence users' perception of it, either positively or negatively. The application's interface has become the primary means through which users interact with it. A website with a well-thought-out design can create a positive impression on potential clients or customers, increase lead generation, and drive conversions.

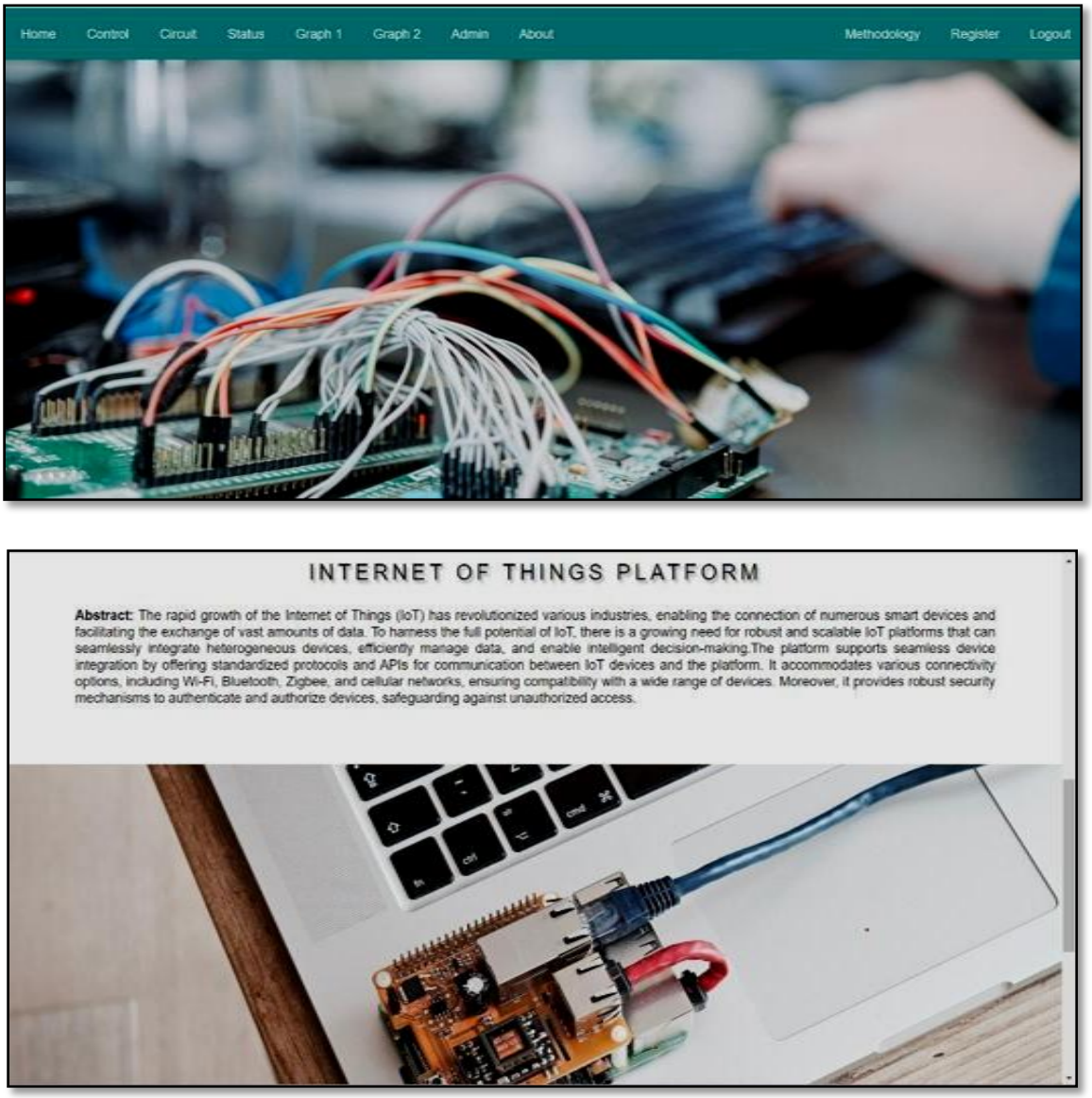


Figure 3: Home Page

Additionally, it provides a great user experience, making it easy for visitors to access and navigate the site. The home page includes links to the sign-up and login forms, as well as information about the team. These links allow the user to easily access the corresponding pages or information. The home page is a crucial starting point for any website as it is the first impression that visitors get of your brand. It serves multiple purposes including attracting and engaging visitors, educating them about your brand, and encouraging them to explore other pages of the site.

- To educate visitors on your brand.
- To encourage visitors to view other pages.
- To attract and engage visitors.

4. **About the Owner Page:** The about the owner page serves as a representation of the application's purpose and the team behind it. It can also include contact details or location

information. Essentially, the about us page can be viewed as a textual self-portrait or brief biography of the business. The about the owner page is a common feature on business websites, providing customers with information about the people and purpose behind the company. This page is also commonly found on newly developed applications/projects. Typically, the about us page includes details about the application's history and the people involved, often presented through short articles and accompanied by photographs.

Figure 4: About Owners

The about the owner page not only includes information about the goals and attitude of a website but also reflects its purpose and personality as well as its team members. Additionally,the page may contain contact details and location information.

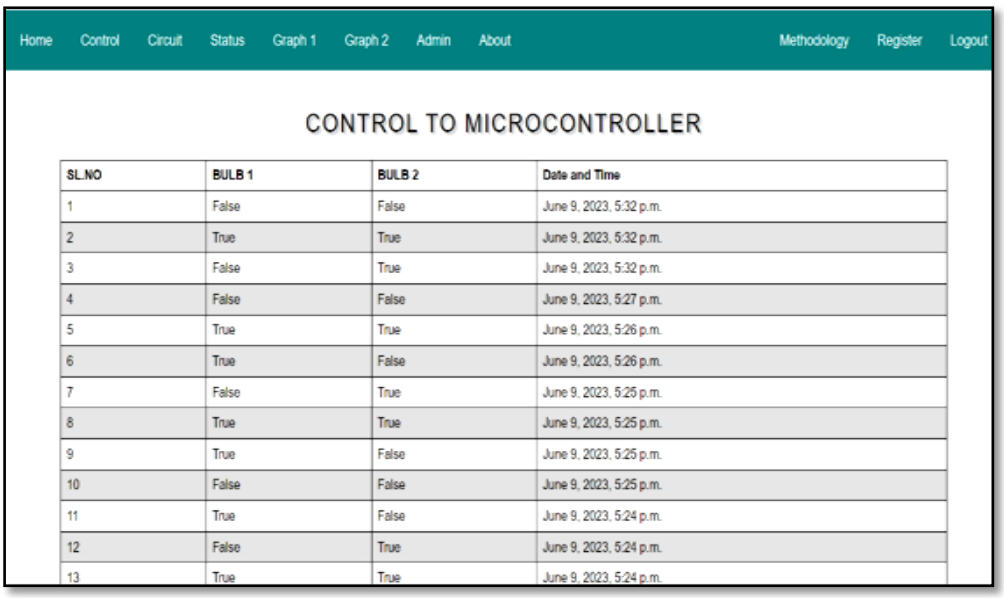
5. **Switch Panel:** The control panel serves as an intermediary between the host computer

and the peripheral devices, managing their functions and ensuring effective communication between them. One of the primary roles of the control panel is to consolidate all connections to the peripheral devices, enabling the user to easily access and manage them from a single location. Another important function of the control panel is to provide power to the peripheral devices, as required.

Figure 5: Switch Panel

The control switches present in the panel play a crucial role in sending data to the microcontroller, which processes the information and executes the required actions. The switches are designed to convert the user's input into a text file format that can be easily processed by the microcontroller. Moreover, the submit button is conveniently placed under the switches, allowing the user to submit their desired changes with ease. Once the submit icon is clicked, the status of the load is changed to the required state of the switch, enabling the userto effectively control the connected peripheral devices.

6. **Data Sent to Microcontroller:** The Node MCU ESP8266 is a popular IoT (Internet of Things) development board that can be programmed to send and receive data wirelessly. It features built-in Wi-Fi capabilities, making it easy to connect to the internet and exchange information with other devices or cloud services. To establish a connection between the NodeMCU and a server using protocols like HTTPS or MQTT. With HTTPS, we can send data to the NodeMCU by making HTTPS POST requests with the desired payload.

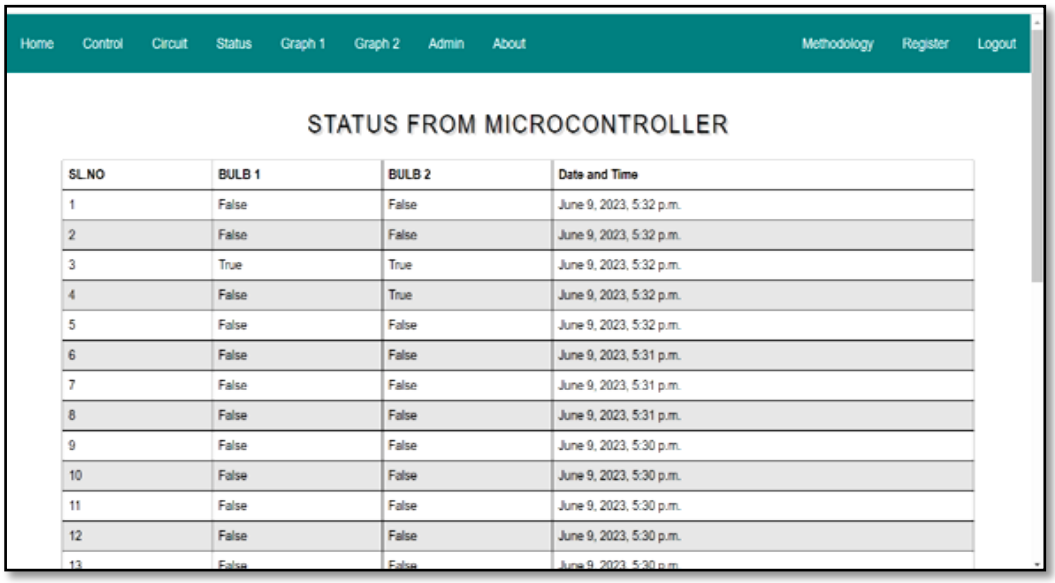


SL.NO	BULB 1	BULB 2	Date and Time
1	False	False	June 9, 2023, 5:32 p.m.
2	True	True	June 9, 2023, 5:32 p.m.
3	False	True	June 9, 2023, 5:32 p.m.
4	False	False	June 9, 2023, 5:27 p.m.
5	True	True	June 9, 2023, 5:26 p.m.
6	True	False	June 9, 2023, 5:26 p.m.
7	False	True	June 9, 2023, 5:25 p.m.
8	True	True	June 9, 2023, 5:25 p.m.
9	True	False	June 9, 2023, 5:25 p.m.
10	False	False	June 9, 2023, 5:25 p.m.
11	True	False	June 9, 2023, 5:24 p.m.
12	False	True	June 9, 2023, 5:24 p.m.
13	True	True	June 9, 2023, 5:24 p.m.

Figure 6: Control to Microcontroller

To send data to the NodeMCU, we used programming language Python. This language provides libraries and APIs specifically designed for working with the NodeMCU ESP8266, simplifying the process of sending and receiving data.

7. **Data Received from Microcontroller:** To receive data on the Node MCU ESP8266, you can implement various communication protocols such as HTTPS or MQTT. With HTTPS, we can set up a web server on the Node MCU and handle incoming requests. This allows us to receive data sent as part of HTTPS POST or GET requests. Once received, we can process and utilize the data according to our application's requirements. By using python as the programming language, the language provides libraries and APIs that simplify the implementation of communication protocols and data processing on the board.



SL.NO	BULB 1	BULB 2	Date and Time
1	False	False	June 9, 2023, 5:32 p.m.
2	False	False	June 9, 2023, 5:32 p.m.
3	True	True	June 9, 2023, 5:32 p.m.
4	False	True	June 9, 2023, 5:32 p.m.
5	False	False	June 9, 2023, 5:32 p.m.
6	False	False	June 9, 2023, 5:31 p.m.
7	False	False	June 9, 2023, 5:31 p.m.
8	False	False	June 9, 2023, 5:31 p.m.
9	False	False	June 9, 2023, 5:30 p.m.
10	False	False	June 9, 2023, 5:30 p.m.
11	False	False	June 9, 2023, 5:30 p.m.
12	False	False	June 9, 2023, 5:30 p.m.
13	False	False	June 9, 2023, 5:30 p.m.

Figure 7: Data Received from Microcontroller

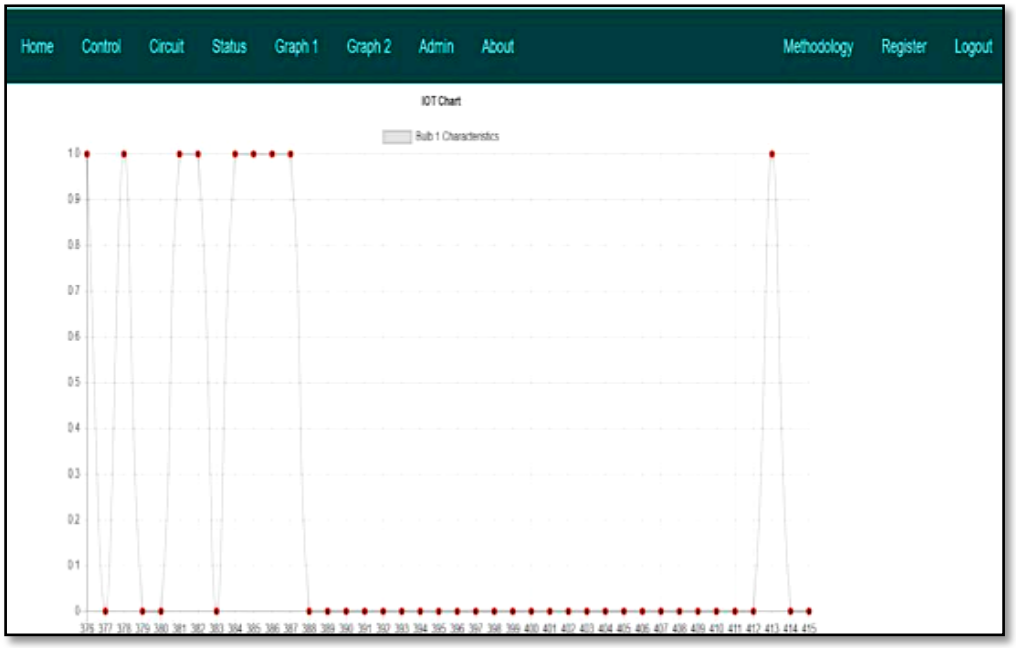


Figure 8: Graph of Load Switching

8. **Register:** This register interface on the authentication modes allows the new user to signup when he/she is a new user and the existing user can login by using the credentials which were generated when the user had created an account.

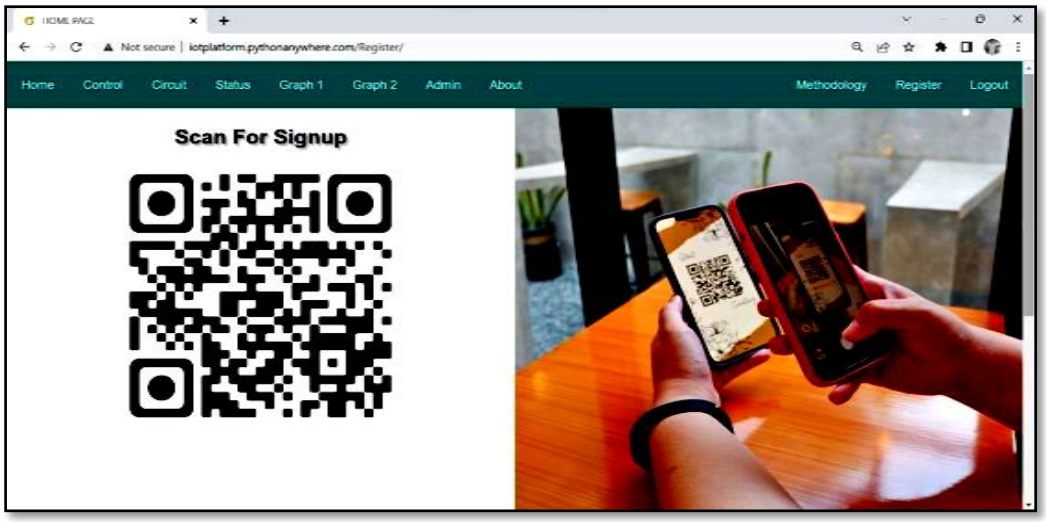


Figure 9: m Registration QR

9. **Admin View Interface:** A built in admin module is been provided by Django which can also be used to perform (CRUD) operations on the models. The meta data is been read by the model which dispenses a quick interface where the content of the application is been managed by the user. The admin related tasks can be performed by the user using the built-in modules that is admin interface ofDjango.

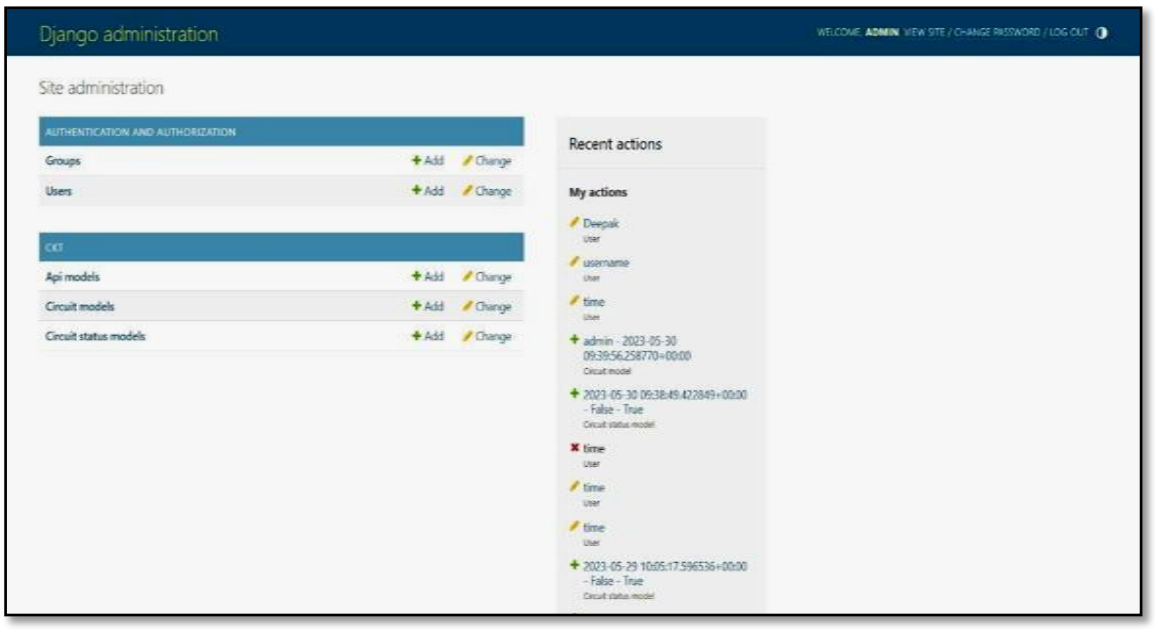


Figure 10: Admin View Interface

10. **Hardware Connection:** This methodology consists of Ac supply which is been connected to the bulb and the relay parallelly. This is been connected to micro controller which is been connected to the host where this acts as the mediator between the cloud and the Node MCU. The web application gives the user to take input and the following inputs are been converted to actions. Output, resulting in higher efficiency and improved performance.

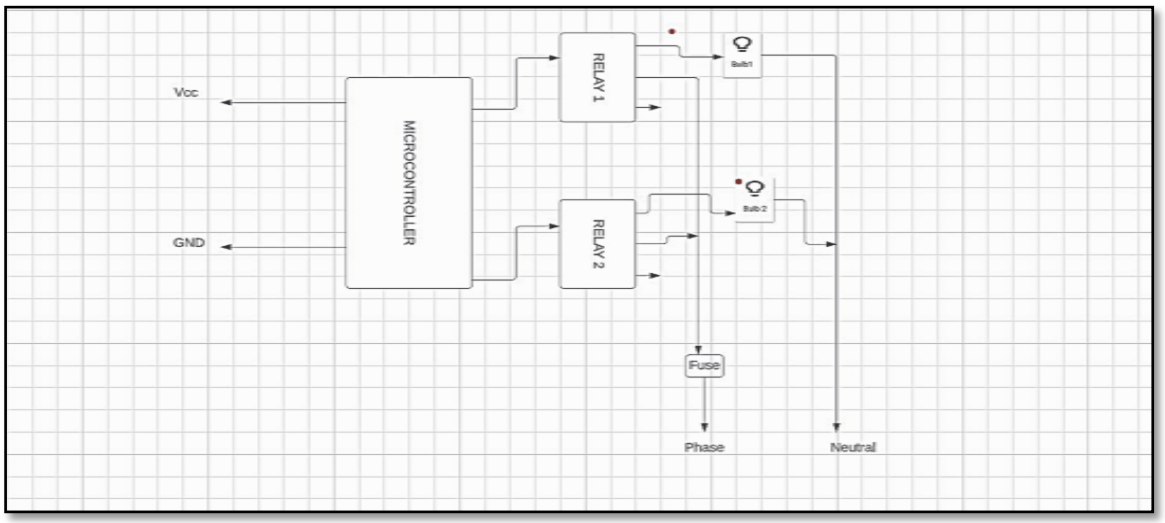


Figure 11: Hardware Block Diagram

A 5V relay module typically consists of a relay, driver circuitry, and supporting components. The relay itself has a coil and contacts. When a 5V signal is applied to the coil, it generates a magnetic field that activates the relay, causing the contacts to change

their state. The contacts of the relay can be either normally open (NO) or normally closed (NC). In the normally open state, there is no connection between the relay's common terminal and the NO terminal. When the relay is activated, the connection is established. In the normally closed state, the connection between the common terminal and the NC terminal is present by default, and it is broken when the relay is activated. A 5V relay module is commonly used in various applications, such as home automation, robotics, IoT projects, and industrial control systems. It allows low-power microcontrollers or digital circuits to control devices that operate at higher voltages or currents, such as motors, lights, solenoids, and heaters.

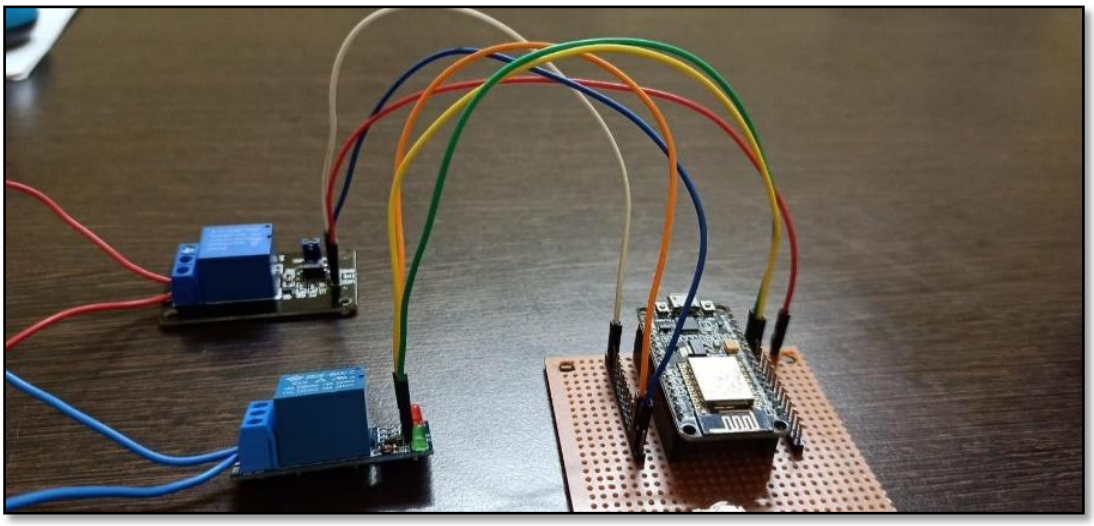


Figure 12: Hardware Connection

This circuit consists of various devices such as relay module, LED bulb, various resistors, Fuse, Node MCU etc. Here the switching of the bulb is been automatically controlled from various geographical locations as per the above shown diagram.

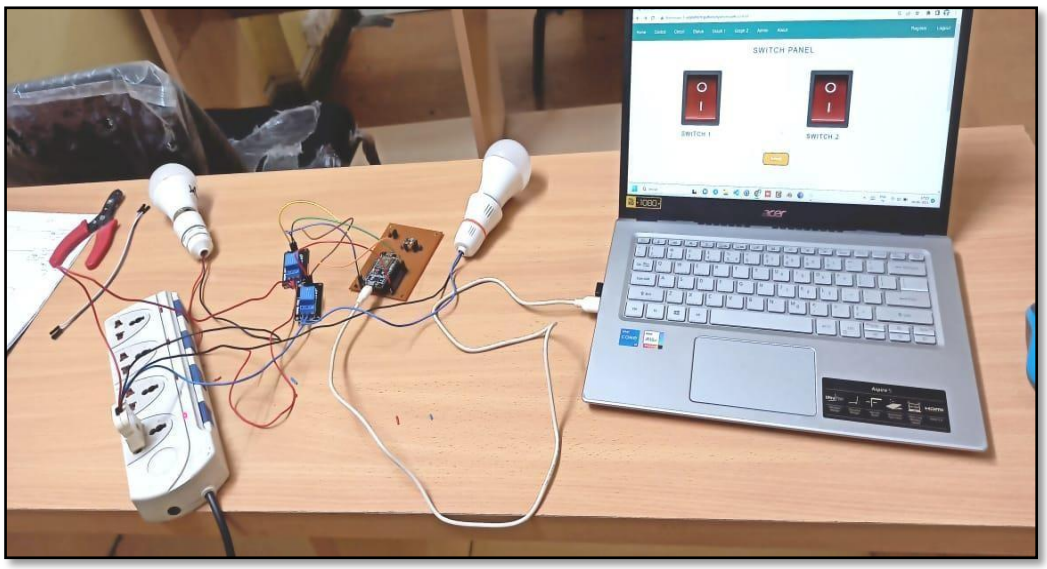


Figure 13: Project Implementation

IV. CONCLUSION

Developing an IoT platform using Python is to create a robust and scalable system that enables seamless communication, control, and data exchange between interconnected devices in a network. The platform facilitates the collection, analysis, and utilization of data from various IoT devices, allowing control. Python, with its versatility and extensive libraries, provides a flexible and powerful programming language to build the IoT platform, ensuring efficient device management, data processing, and integration with other technologies. We achieved real-time data streaming, device monitoring and control, data analytics, secure communication, and the ability to scale the platform as the IoT ecosystem expands.

REFERENCES

[1] J. Mohammed, C. -H. Lung, A. Ocneanu, A. Thakral, C. Jones and A. Adler, "Internet of Things: Remote Patient Monitoring Using Web Services and Cloud Computing," 2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom), Taipei, 2014, pp. 256-263, doi: 10.1109/iThings.2014.45.Nicholas D., Darrell B., Somsak S., “Home Automation using Cloud Network and Mobile Devices”, IEEE Southeastcon 2012, Proceedings of IEEE.

[2] T. Anagnostopoulos et al., "Challenges and Opportunities of Waste Management in IoT-Enabled Smart Cities: A Survey," in IEEE Transactions on Sustainable Computing, vol. 2, no. 3, pp. 275-289, 1 July-Sept. 2017, doi: 10.1109/TSUSC.2017.2691049.

[3] S. D. T. Kelly, N. K. Suryadevara and S. C. Mukhopadhyay, "Towards the Implementation of IoT for Environmental Condition Monitoring in Homes," in IEEE Sensors Journal, vol. 13, no. 10, pp. 3846-3853, Oct. 2013, doi: 10.1109/JSEN.2013.2263379.

[4] M. Soliman, T. Abiodun, T. Hamouda, J. Zhou and C. -H. Lung, "Smart Home: Integrating Internet of Things with Web Services and Cloud Computing," 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, Bristol, UK, 2013, pp. 317-320, doi: 10.1109/CloudCom.2013.155.

[5] M. Soliman, T. Abiodun, T. Hamouda, J. Zhou and C. -H. Lung, "Smart Home: Integrating Internet of Things with Web Services and Cloud Computing," 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, Bristol, UK, 2013, pp. 317-320, doi: 10.1109/CloudCom.2013.155.

[6] C. Wang, M. Daneshmand, M. Dohler, X. Mao, R. Q. Hu and H. Wang, "Guest Editorial - Special Issue on Internet of Things (IoT): Architecture, Protocols and Services," in IEEE Sensors Journal, vol. 13, no. 10, pp. 3505-3510, Oct. 2013, doi: 10.1109/JSEN.2013.2274906.

[7] R. Kumar and M. P. Rajasekaran, "An IoT based patient monitoring system using raspberry Pi," 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16), Kovilpatti, India, 2016, pp. 1-4, doi: 10.1109/ICCTIDE.2016.7725378.

[8] L. O. Aghenta and M. T. Iqbal, "Development of an IoT Based Open Source SCADA System for PV System Monitoring," 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), Edmonton, AB, Canada, 2019, pp. 1-4, doi: 10.1109/CCECE.2019.8861827.

[9] M. B. Yassein, M. Q. Shatnawi, S. Aljwarneh and R. Al-Hatmi, "Internet of Things: Survey and open issues of MQTT protocol," 2017 International Conference on Engineering & MIS (ICEMIS), Monastir, Tunisia, 2017, pp. 1-6, doi: 10.1109/ICEMIS.2017.8273112.

[10] N. Kumar, J. Madhuri and M. ChanneGowda, "Review on security and privacy concerns in Internet of Things," 2017 International Conference on IoT and Application (ICIOT), Nagapattinam, India, 2017, pp. 1-5, doi: 10.1109/ICIOTA.2017.8073640.

