

PREDICTIVE MAINTENANCE SCHEDULING AND ENVIRONMENT MONITORING

**GE19612 - PROFESSIONAL READINESS FOR INNOVATION,
EMPLOYABILITY AND ENTREPRENEURSHIP PROJECT REPORT**

Submitted by

AJITHA B (2116220701019)

ALDRIN ROGER S (2116220701023)

HAREESH S (2116220701079)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI

MAY 2025

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Project titled “**PREDICTIVE MAINTENANCE SCHEDULING AND ENVIRONMENT MONITORING**” is the bonafide work of **AJITHA B (220701019)**, **ALDRIN ROGER (220701023)**, **HAREESH S (220701031)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. P. Kumar., M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Professor

Department of Computer Science and
Engineering,
Rajalakshmi Engineering College,
Chennai - 602 105.

SIGNATURE

Dr. M. Ayyadurai,

SUPERVISOR

Assistant Professor

Department of Computer
Science and Engineering,
Rajalakshmi Engineering
College, Chennai - 602 105.

Submitted to Mini Project Viva-Voce Examination held on _____

Internal Examiner

External Examiner

ABSTRACT

In modern computer laboratories and working environments, computers and laptops could be running all the time, even during idle states, resulting in unnecessary power consumption, a hefty energy bill, with the risk of hardware damage due to overheating. In this paper, we propose an intelligent scheduling system that uses IoT-based smart plugs to monitor and manage the power states of electrical devices. IoT sensors and cloud automation will be integrated into the system to record usage patterns of devices and detect inactivity, after which it may apply scheduled shutdowns or power-saving modes during non-usage hours. This will help conserve energy and extend the life of devices while also minimizing downing time due to thermal stress or overuse. However, the system was validated in a control lab environment that provided substantial energy savings while providing efficient management of assets.



ACKNOWLEDGMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.**, our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.**, and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D.**, for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.**, our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR, M.E., Ph.D.**, Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guides **Dr. M. AYYADURAI** and **Dr. G.M. SASIKALA**. We are very glad to thank our Project Coordinator, **Dr. M. AYYADURAI** Assistant Professor Department of Computer Science and Engineering for his useful tips during our review to build our project.

| | |
|-----------------------|----------------------|
| AJITHA B | 2116220701019 |
| ALDRIN ROGER S | 2116220701023 |
| HAREESH S | 2116220701079 |



TABLE OF CONTENTS

| CHAPTER NO. | TITLE | PAGE NO. |
|--------------------|---------------------------------|-----------------|
| | ABSTRACT | iii |
| | ACKNOWLEDGMENT | iv |
| | LIST OF TABLES | vii |
| | LIST OF FIGURES | viii |
| 1. | INTRODUCTION | 1 |
| | 1.1 GENERAL | 1 |
| | 1.2 OBJECTIVES | 2 |
| | 1.3 EXISTING SYSTEM | 3 |
| 2. | LITERATURE SURVEY | 4 |
| 3. | PROPOSED SYSTEM | 7 |
| | 3.1 GENERAL | 7 |
| | 3.2 SYSTEM ARCHITECTURE DIAGRAM | 7 |
| | 3.3 DEVELOPMENT ENVIRONMENT | 8 |
| | 3.3.1 HARDWARE REQUIREMENTS | 8 |
| | 3.3.2 SOFTWARE REQUIREMENTS | 9 |
| | 3.4 DESIGN THE ENTIRE SYSTEM | 9 |
| | 3.4.1 ACTIVITY DIAGRAM | 9 |
| | 3.4.2 WORK FLOW DIAGRAM | 11 |
| 4. | MODULE DESCRIPTION | 13 |
| | 4.1 SYSTEM ARCHITECTURE | 13 |



| | | |
|----|---|----|
| | 4.2 SYSTEM WORKFLOW | 15 |
| 5. | IMPLEMENTATIONS AND RESULTS | 17 |
| | 5.1 IMPLEMENTATION | 17 |
| | 5.2 OUTPUT SCREENSHOTS | 17 |
| 6. | CONCLUSION AND FUTURE ENHANCEMENTS | 21 |
| | 6.1 CONCLUSION | 21 |
| | 6.2 FUTURE ENHANCEMENTS | 22 |
| | REFERENCES | 24 |



LIST OF TABLES

| TABLE NO | TITLE | PAGE NO |
|----------|-----------------------|---------|
| 3.1 | HARDWARE REQUIREMENTS | 9 |
| 3.2 | SOFTWARE REQUIREMENTS | 9 |



LIST OF FIGURES

| FIGURE NO | TITLE | PAGE NO |
|-----------|---------------------|---------|
| 3.1 | SYSTEM ARCHITECTURE | 8 |
| 3.2 | ACTIVITY DIAGRAM | 10 |
| 3.3 | WORKFLOW DIAGRAM | 12 |
| 4.1 | SEQUENCE DIAGRAM | 20 |

CHAPTER 1

INTRODUCTION

1.1 GENERAL

The rapid digital transformation of educational institutions, corporate offices, and research labs has almost made it mandatory for the increase in substantial computing devices like desktops, laptops, and workstations in the premises. These systems became a prerequisite for everyday activities such as programming, simulation, documentation, and data processing. One of the commonly shared problems related to all these environments is leaving computers literally switched on-an example can be night, weekends, or extended periods of no use. This, apart from consuming unnecessary energy, causes overheating, faster degradation of hardware, and increased maintenance costs.

Yet most of the institutions rely on users to even carry out this protocol in power shutdown management manually or expect them to remember turning off their systems, which people often do not do, mostly due to human error or because of expectations of short-term reuse. In addition, the usual methods for saving power like sleep mode or screensavers do not suffice to address relatively longer inactivity or misconfigurations of the system. Collectively, these issues lead to energy inefficiency, environmental impact, and operational costs when scaled across dozens or hundreds of machines, however.

The birth of the Internet of Things (IoT), however, presented entirely new possibilities for automating and optimizing usage of devices much more intelligently. Smart plugs connected to IoT can also be integrated with existing infrastructure for real-time monitoring of power consumption and determine the nonuse periods of operation, all setting up an automated scheduling policy without user intervention. Smart systems come with an added advantage of providing data analytics dashboards to administrators for more informed decision-making and proactive maintenance.

Introducing schedule management systems developed particularly for computer labs and such environments feature the combination of the IoT-based smart plugs; cloud connectivity; intelligent scheduling algorithm. This aims to minimize wasteful power consumption and extend the useful lifespan of devices while on the other hand ensuring that the systems are running only when they are actually required. Testing was done on the proposed solution in a controlled lab setting, and evaluation criteria were based on energy savings, cost-effectiveness, and ease of deployment.

1.2 OBJECTIVE

And the purpose of IoT-based SMSSMS is to optimize the efficiency and the reliability of maintenance operations utilizing RM and AM technologies. The system captures data from IoT devices and continuously monitoring power consumption, CPU temperature and air quality parameters to check the equipment performance. By incorporating predictive analytics, the system is able to predict at potential failure before it happens, enabling for upcoming maintenance, and stopping cost downtime. Automatic alerts inform maintenance personnel when readings go over predetermined thresholds, plus automatic-of-action enable auto-shutoff of devices, say to protect themselves. All data is safely secured in a combined cloud database for easy access and evaluation. In addition, the system gives a customizable user-friendly web dashboard to the users also wherein users can track data in real time, manage the schedule of the maintenance and get alerts in efficient manner. In the end, the system intends to minimize operational expenses, enlarge product life, and optimize revolutions, leading to improved reliability and pieces per hour in sector associated lines. The IoT-based Smart Maintenance Scheduling and Monitoring System as well, aims for improving the decision-making by providing the historical data analysis along with predictive analytics. With cloud-based architecture, the system is able to offer scalability and flexibility, adjusting to the Industrial changes of the requirements. The automation of decision-making ensures a smaller need for human intervention and therefore it significantly reduces the occurature of errors and also speed up and improve the accuracy of maintenance activities. This system essentially works to

optimize workflow, offers cost efficient solutions as a result of stopping unnecessary repairs and keeps up a more sustainable practice for Industrial apparatus management. The system, which features latest IoT technology enabled, is efficient to Industry 4.0, it triggers smarter and the more linkages industrial operation.

1.3 EXISTING SYSTEM

Current maintenance tools are mostly manual or semi-automated, based on scheduled and fixed dates instead of real-time information. These platforms do not have the capability to integrate with IoT technologies and as a result delay fault detection and inefficient use of resources, causing increase downtime. Maintenance records are usually stored in spreadsheets or separate programs, making it difficult to track and analyze data. Without the predictive abilities or the central control, the current systems fail in optimising activities of repairs and, consequently, additionally enhancing operational expenses as well as decreasing the equipment reliability.

CHAPTER 2

LITERATURE

SURVEY

An analytical model-based maintenance scheme is presented for an IoT-enabled hybrid flow shop (HFS). The model evaluates machine health and identifies hidden risks through performance metrics such as real-time cycle time, work in progress (WIP), and squared coefficient of variance (SCV) in service time. Maintenance action is initiated by comparing these metrics against threshold values so that machine failure could be prevented. The case study of an IoT-enabled HFS validates the scheme optimized using interval type-2 fuzzy logic systems (IT2FLSs).[1]

In this paper, we present a simple technique for online monitoring and predictive maintenance in connected manufacturing systems of the future via the Internet of Things. While IoT platforms support predictive maintenance using data from a variety of machines, they all still suffer from communication incompatibility with industrial protocols. The proposed method solves this problem by enabling real-time process monitoring for quality assurance as well as condition monitoring to avoid unplanned downtimes. Provisions are made for a case study to demonstrate the feasibility.[2]

This study proposes an availability-oriented maintenance scheduling policy for vibrating-grate biomass boilers with the rationale of reducing costs in achieving the intended availability of the system. From constant failure rates, we use Weibull regression for time-dependent hazard functions and validate them with Kolmogorov–Smirnov tests. Ans maintenance impact is one of the most critical factors influencing the prioritization of the components relative to their availability impact. Based on fault tree analysis, the system availability estimation shows that the proposed predictive maintenance model can reduce 32% of the maintenance requirements.[2], [3]

This paper focuses on a comparative study of the various meta-heuristic algorithms, namely Genetic Algorithm, Particle Swarm Optimization, Artificial Bee Colony, and Grey Wolf Optimizer deployed in multi-state system maintenance optimization under

time and availability constraints. The aim is to minimize total costs of preventive maintenance by optimizing the inspection and maintenance intervals. The performance measures of the algorithms were assessed based on solution quality, stability, and convergence speed, with an intent to identify the most efficient approach.[4]

In this paper ,the dissertation explores the development of easy-to-use mobile applications that focus on mechanical device maintenance including bicycles along with automobiles and air conditioners for predictive maintenance needs. Predictions related to device failure and health status and service notices are achieved through algorithm fine-tuning of an external calibration device within the application system. The application uses data from pressure controller sensors together with calibration data along with age and component failure information for optimal maintenance scheduling which reduces equipment downtime. The proposed solution increases reliability alongside efficiency through the execution of prompt preventive actions relying on acquired data.[5]

This paper presents ReinFog, a deep reinforcer learning (DRL)-based distributed framework for effectively managing resources between an edge and a cloud under a unified dynamic scheduling for an application of IoT in scalable and adaptable systems that are energy-efficient and demonstrate significant reductions in response time and cost of systems. [6]

In this paper,the DRLIS algorithm is presented; it implements deep reinforcement learning to improve the load distribution and response times for IoT applications running in edge-fog-cloud systems. The scheduler can adapt to changing environments with superiority over conventional metaheuristic and RL methods.[7]

In this paper DRLMONT, which is a deep Q-learning based task scheduling technique that efficiently allocates tasks between fog and cloud layers. The technique optimizes energy consumption, latency of processing, and cost under heterogeneous dynamic workloads in a cloud-fog environment.[8]

Application of deep reinforcement learning schedules multi-component jobs in

edge-cum-cloud computing environments. In modeling inter-task dependencies and server constraints, improvement in throughput is achieved along with a reduction in job completion time under varying loads.[9]

In this paper, the establishment of a DRLF, which refers to a deep reinforcement learning framework designed to schedule dynamic tasks in edge-cloud infrastructures. The RNN architecture with deep reinforcement learning will efficiently capture the temporally inclined patterns of IoT workloads. The simulation results conducted on CloudSim and iFogSim were compared with conventional scheduling techniques and proved that the proposed reinforcement learning-based dynamic scheduling (RLbDS) algorithm achieved better adaptability and efficiency through all experimental results.[10]

CHAPTER 3

PROPOSED SYSTEM

3.1 GENERAL

The IoT Based Smart Maintenance Scheduling and Monitoring System improves the reliability of equipment by incorporating real time monitoring and automated scheduling. The sensors in the IoT collect temperature, vibration, and usage data and send it to a cloud server for processing. Anomalies are detected using predictive algorithms, and maintenance tasks are scheduled accordingly. A web-based portal for equipment statuses, alerts, and maintenance history for easy visibility. SMS and email alerts are used for rapid communication with maintenance crews. Groundbreaking modular, scalable design enables easy integration with existing platform, minimize unplanned downtime, and advance the transition to smart, automated industrial operation.

3.2 SYSTEM ARCHITECTURE DIAGRAM

The system proposed the three-layer structure for real-time monitoring and intelligent maintenance. The sensor/controller layer is implemented by an ESP32 microcontroller equipped with power consumption, CPU temperature, and quality sensor. This information is processed on-the-edge and sent over Wi-Fi to Blynk IoT cloud/communication layer. Using blynk mobile app as a gateway to graphical show (and push) real-time sensor data into web app/backend. Here's where the custom server comes in: it listens for the data, compares it to some user-set threshold levels, and takes action when needed. For example, if any one parameter is out of normal range, then the server is able to remotely power off the device against damage. At the same time, all sensor readings, events and actions are time-stamped and synchronized with a central, safe database for logging and future analysis. The system provides a web dashboard for users to monitor device status, view historical data, and manage maintenance schedules efficiently, ensuring a robust and intelligent maintenance solution.

The HW architecture is divided into 3 layers: ESP32 + the sensors, Blynk IoT which manages the communication and the visualization and a back-end server which processes the readings. The system also can automatically turn off devices when they exceed thresholds. Everything is finally stored in a secure way, available on a website dashboard for monitoring and scheduling maintenance.

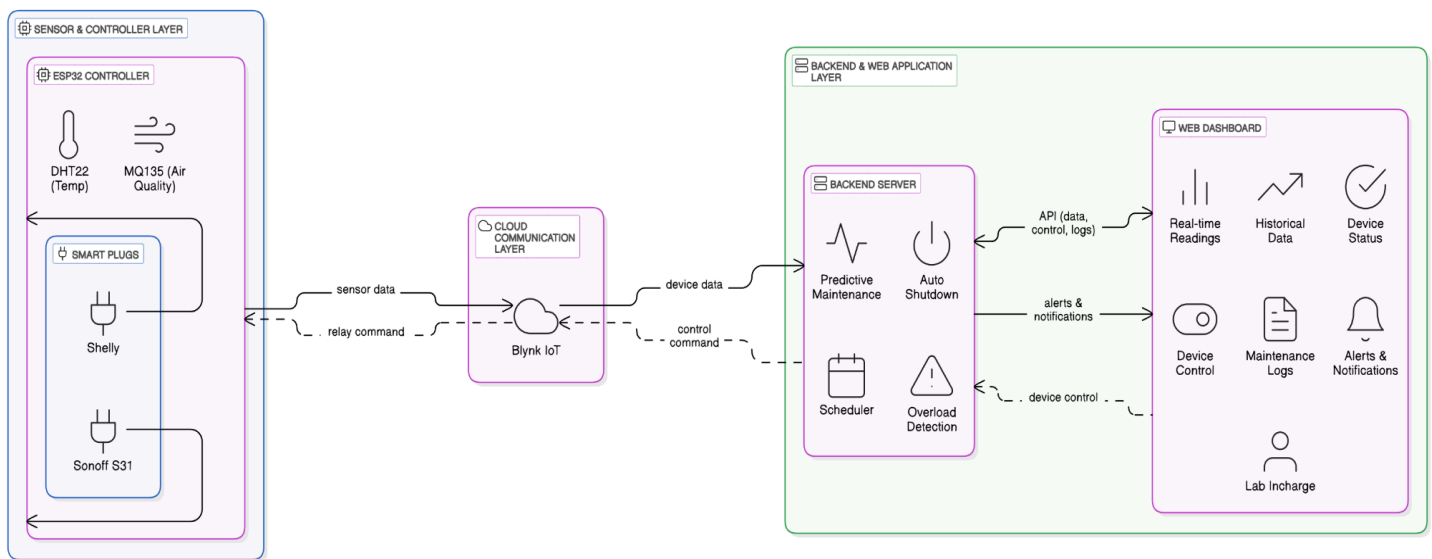


Fig 3.1: System Architecture

3.3 DEVELOPMENTAL ENVIRONMENT

3.3.1 HARDWARE REQUIREMENTS

The hardware specifications could be used as a basis for a contract for the implementation of the system. This therefore should be a full, full description of the whole system. It is mostly used as a basis for system design by the software engineers.

Table 3.1 Hardware Requirements

| COMPONENTS | SPECIFICATION |
|--------------|------------------|
| PROCESSOR | Intel Core i5 |
| RAM | 8 GB RAM |
| POWER SUPPLY | +5V power supply |

3.3.2 SOFTWARE REQUIREMENTS

The things that this system needs are the Blynk IoT system for real time data visualization, Arduino IDE ESP32 for ESP32 programming, a backend server ofcourse. js or Python Flask), a cloud storage system (such as Firebase or MySQL) and a web application framework (such as a React or Angular framework) for the dashboard. It Should have good internet and api connections.

Table 3.2 Software Requirements

| COMPONENTS | SPECIFICATION |
|------------------|---------------------|
| Operating System | Windows 7 or higher |
| Frontend | HTML, CSS,React js |
| Backend | Node js |
| Database | Postgresql |

3.4 DESIGN OF THE ENTIRE SYSTEM

3.4.1 ENTITY-RELATIONSHIP DIAGRAM

In the fig.3.2 the Entity-Relationship Diagram of the IOT Maintenance System. This is a diagram that shows all types of relationships between users, devices, sensors, readings, events, or maintenance logs that will be possible for secure and normalized data representation.

To enhance the performance of queries, indexing into timestamp and sensor_id fields in the readings table has been deployed to enable real-time data retrieval for purposes of graph plotting and threshold comparison. Similarly, device_id and event_type fields will be indexed into the events table for rapid log access and pattern recognition.

Moreover, the schema is modular enough to horizontally scale the system. This means that adding any new lab or device would only require simple modifications in the devices and sensors tables without any effect on core logic or user interface. It also includes proper provisions for audit logging as well as compliance reporting, which are essential for the needs of institutional IT environments.

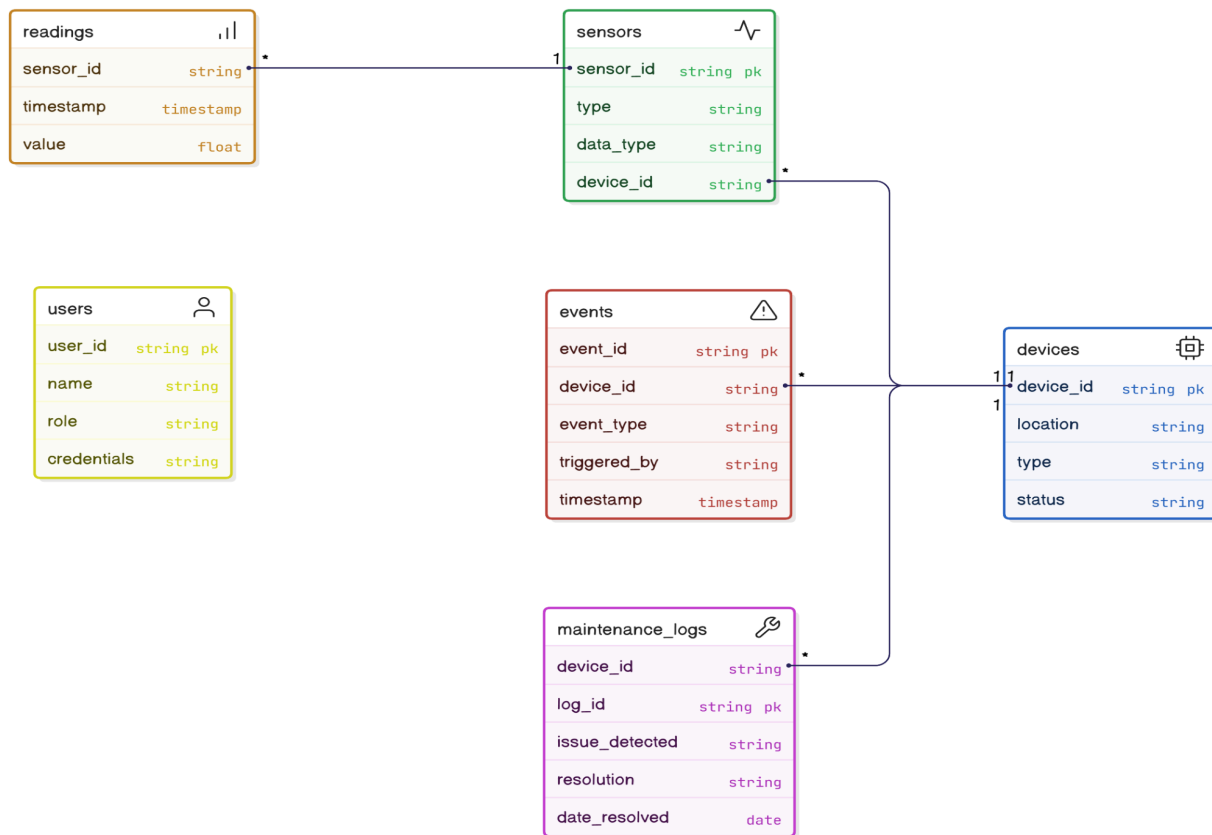


Fig 3.2: Entity-Relationship Diagram

3.4.2. WORKFLOW DIAGRAM

The workflow of the IoT-based Smart Maintenance Scheduling and Monitoring System is initiated by ESP32 microcontroller, that gathers real-time information from the sensors attached base upon parameters that includes the power consumption, CPU temperature and the quality of the air. The ESP32 processes information at its local level before sending the data through Wi-Fi to Blynk IoT cloud platform for real-time visualization. Blynk simultaneously sends this sensor data to the backend server which then performs threshold checkups of the recorded values. If any parameter of a device exceeds safe limits, the corresponding control actions are automatically performed by the backend, for example, shutdown of the device to avert damage. All sensor data, events, and actions are stored in a single database to record and analyze information. The web interface provides customers with access to current and past data, notification, management of maintenance schedules and management support to aid a easier decision and management remotely.

The method of operation of the IoT-based Smart Maintenance Scheduling and Monitoring System starts with the ESP32 microcontroller, the ESP32 microcontroller collects in real time data from sensors for monitoring consumption of power, CPU temperature and air quality. This data is treated locally and relayed to the Blynk IoT System via Wi-Fi, in which it is shown live and quickly for simple visual checking. Blynk also a data sends to that backend server, where it performs evaluations based on threshold. If a reading is found to be beyond safe value, automated control action like shutdown of the device is implemented to avoid damage. The platform captures all data and events in a secured centralized database where they are traceable and tamper-proof. Users have the access to live data, schedule status alert, live pair data by using web dashboard, remote monitoring, alarms , receipt as well as scheduled maintenance knowledge. It has role-based activities for various users, remote device control capabilities, and all-in-one solution for facility management that sends SMS or Email alerts with serious diligence of notification services for faster remediation and complete system control in industrial plants.

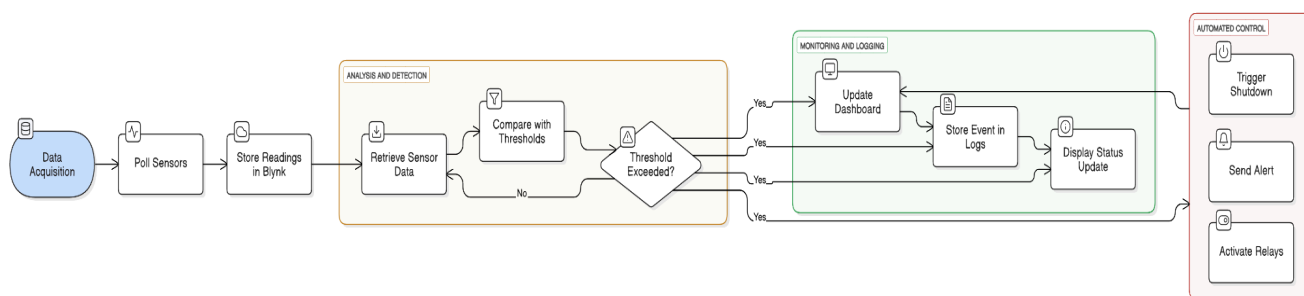


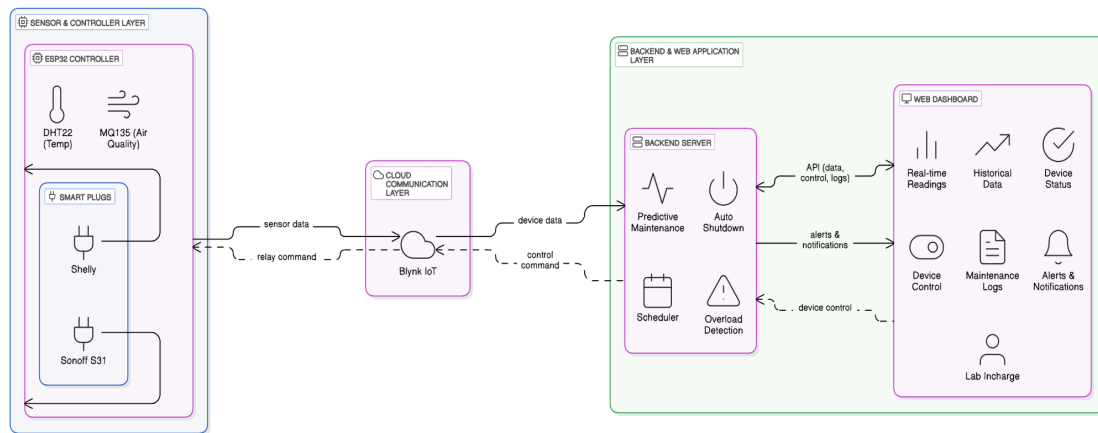
Fig 3.3 Work Flow Diagram

CHAPTER 4

MODULE DESCRIPTION

The workflow for the proposed Placement Tracker system is designed to ensure a structured and efficient process for managing campus recruitment activities. It consists of the following sequential steps:

4.1 SYSTEM ARCHITECTURE



4.1.1 USER INTERFACE DESIGN

The Fig 4.1 illustrates the workflow of the *IoT-based Smart Maintenance Scheduling and Monitoring System*, starting with the ESP32 microcontroller collecting real-time data from connected sensors that monitor parameters like power consumption, CPU temperature, and air quality. The ESP32 locally processes this data and transmits it over Wi-Fi to the Blynk IoT Cloud Platform for real-time display and analysis.

Upon receiving the data, Blynk forwards it to the backend server, which performs threshold validation. If any parameter exceeds the defined safe limits, the backend system automatically triggers appropriate control actions, such as device shutdown or warning signals, to prevent potential damage.

Simultaneously, all sensor readings, triggered events, and system actions are logged in a centralized database, enabling traceability and historical analysis. The web dashboard gives users secure access to live data, system alerts, and maintenance schedules.

Role-based access ensures that different stakeholders—such as facility managers, maintenance staff, or administrators—can perform relevant actions like viewing alerts, updating maintenance logs, or remotely controlling devices. Critical alerts are sent via SMS or Email, ensuring swift intervention. This sequence ensures an automated, transparent, and proactive approach to equipment health and industrial maintenance.

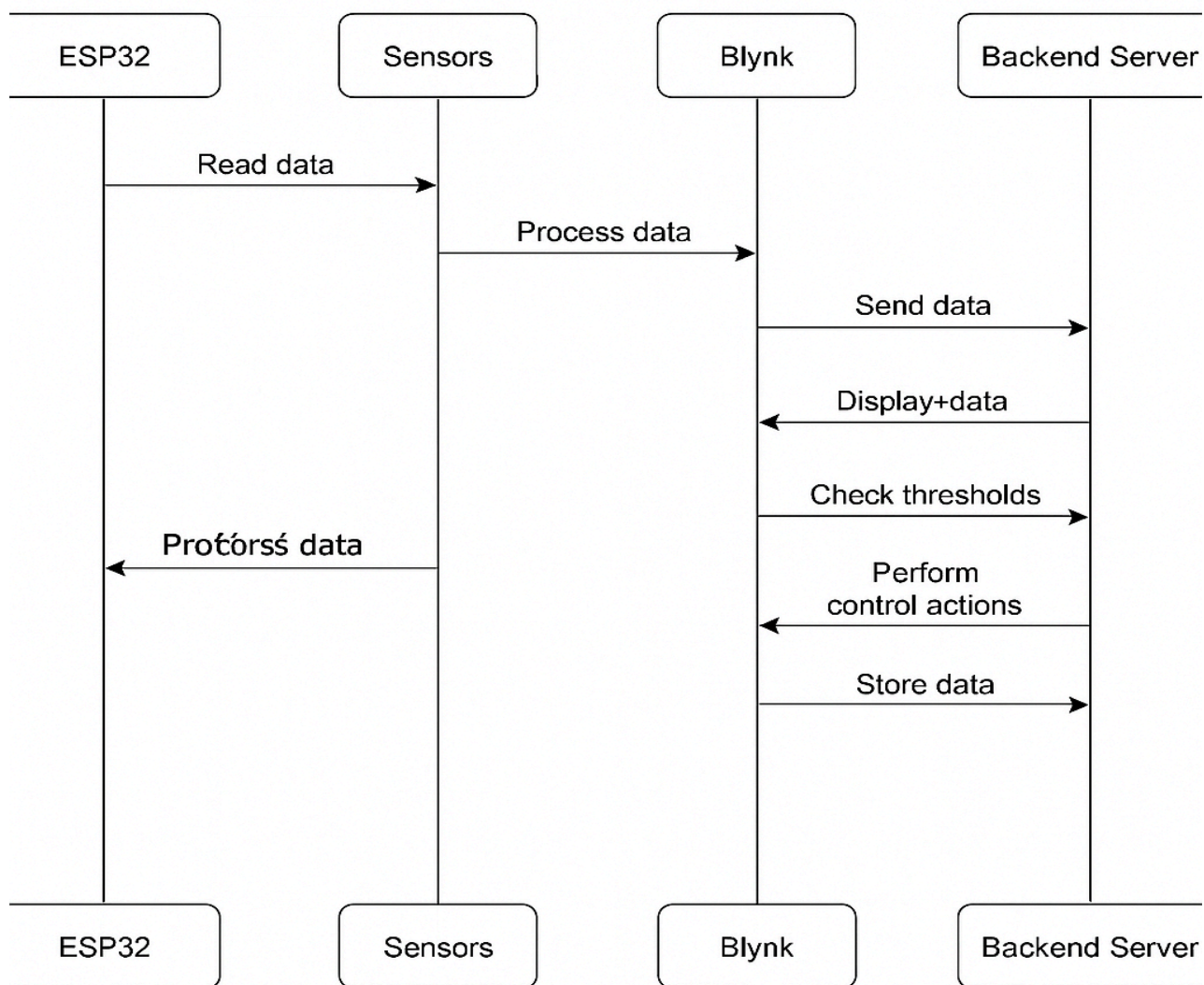


Fig 4.1. SEQUENCE DIAGRAM

4.2 SYSTEM WORKFLOW:

The operation of the IoT-based Smart Maintenance Scheduling and Monitoring System is initiated by the ESP32 microcontroller, which collects real-time data from various sensors. These sensors monitor critical parameters such as:

- Power Consumption
- CPU Temperature
- Air Quality

Once collected, the ESP32 performs basic local processing on this data and then transmits it via Wi-Fi to the Blynk IoT Cloud Platform for real-time visualization and alerting.

Simultaneously, Blynk forwards this sensor data to the backend server, which performs threshold analysis. If any parameter exceeds its predefined safety limits, the backend initiates automated control actions, such as:

- Device shutdown to prevent damage
- Triggering alerts or maintenance actions

All data, including sensor readings, events, and control actions, is securely stored in a centralized database. This database supports:

- Tamper-proof historical data logging
- Audit trails for compliance and traceability

The web-based user interface enables users to:

- View real-time and historical data
- Receive alerts and notifications via SMS or Email
- Remotely monitor and control devices
- Manage and schedule maintenance tasks
- Access role-based dashboards with varying privileges

The system supports facility-wide integration, enabling smart maintenance planning and rapid fault remediation in industrial settings. Its unified platform offers enhanced remote

management, efficient decision-making support, and instant notifications to ensure operational continuity and equipment safety.

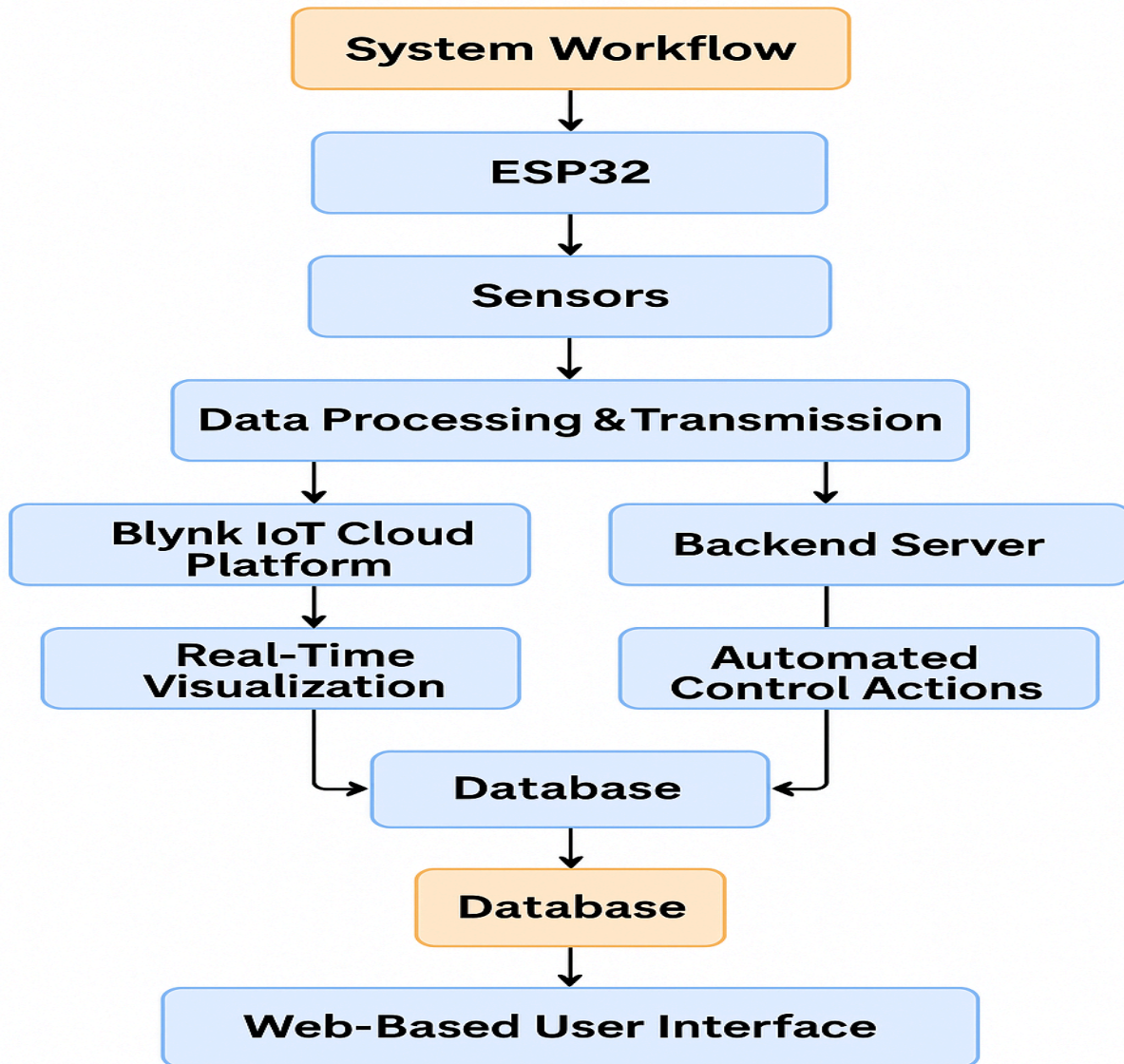


Fig 4.1. Flow Diagram for the System workflow

CHAPTER 5

IMPLEMENTATION AND RESULTS

5.1 IMPLEMENTATION

The implementation of the IoT-based Smart Maintenance Scheduling and Monitoring System involves several stages, starting with hardware setup, software development, integration, and testing. The system relies on the ESP32 microcontroller, which interfaces with various sensors like temperature, humidity, air quality, and power consumption sensors. These sensors collect real-time data from industrial equipment and send it to the cloud via Wi-Fi. The data is processed locally by the ESP32 before being transmitted to the Blynk IoT platform, where it is visualized in real-time. The data is also sent to a custom backend server built using web frameworks like Node.js or Python Flask, hosted on scalable cloud platforms like AWS or Azure. The backend server compares incoming sensor data against predefined thresholds, triggering alerts if any parameter exceeds the safe limits. Alerts are sent via SMS, email, or the Blynk app to maintenance personnel. Additionally, automated actions such as equipment shutdown are triggered to prevent potential failures. A web-based dashboard, developed using frameworks like React or Angular, provides a user interface for real-time data visualization, historical record access, and maintenance scheduling. The dashboard is connected to the backend, enabling users to monitor the system and take corrective actions as needed. Finally, the system undergoes rigorous testing to ensure the correct integration of sensors, cloud communication, and backend logic. Once successful, the system is deployed for real-time monitoring and maintenance automation in industrial environments.

5.2 OUTPUT SCREENSHOTS

The maintenance scheduling system that deploys IOT is now being used in a laboratory environment of an institution and is integrated with computers that are already controlled by ESP32-based smart plug and DHT22 temperature sensors to monitor temperature and MQ135-based air quality monitoring. To evaluate the performance of the deployment with a very holistic scope, an entire web application was built with all modules including user login authentication, role-based dashboards, hardware asset management, maintenance records, user tracking, and smart notifications functions and experiences.

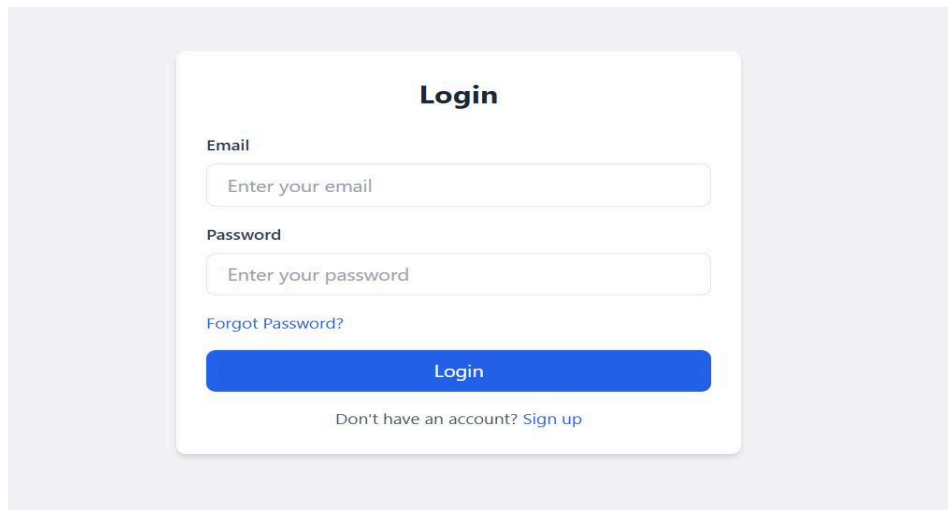


Fig 5.1. Secure Login interface for authenticated system access

To make a secure login page strictly for authorized personnel who are logged in to the system. It authenticates the credentials according to user roles Admin, Lab-Incharge, or Technician-and subsequently redirects to an individual dashboard for that user. Secure design includes encryption and password protection for system integrity.

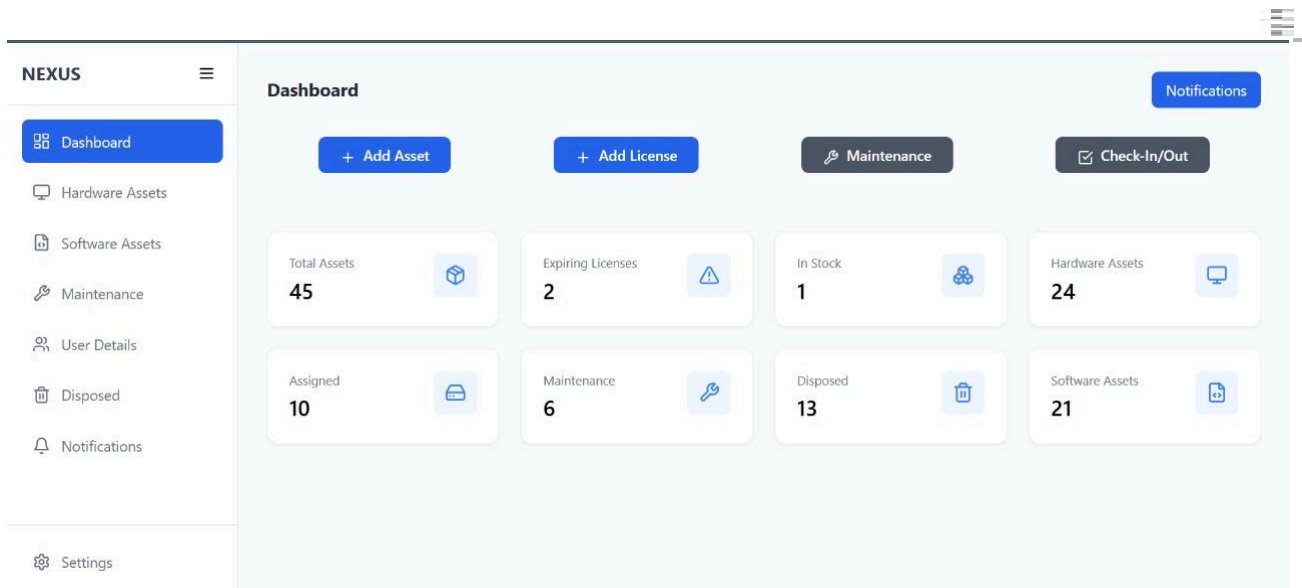


Fig 5.2. Centralized dashboard showing asset status, maintenance, and activity metrics.

The dashboard is the main control panel for lab incharges as well as administrators. This will show real-time data, including the total number of assets, license validity, maintenance alerts, check-in/check-out, and more. Visibility and operational awareness are improved across all related systems.

The screenshot shows the NEXUS Maintenance Records table. The sidebar menu is the same as in Fig 5.2, but the 'Maintenance' option is highlighted. The table has a search bar and an 'Add New' button. The table columns are: Asset ID, Maintenance ID, Issue, Expected Date, Cost, Vendor, Approval Status, Comments, Request Date, and Actions. The table contains six rows of data:

| Asset ID | Maintenance ID | Issue | Expected Date | Cost | Vendor | Approval Status | Comments | Request Date | Actions |
|----------|----------------|-----------------------------------|---------------|---------|---------------------|-----------------|--|--------------|--------------------|
| ZA-LTP12 | 2004 | Screen replacement needed | Feb 01, 2025 | 300.00 | Display Repairs Ltd | Approved | | N/A | View, Edit, Delete |
| ZA-LTP88 | 2 | keyboard issue | Mar 02, 2025 | 900.00 | Chennai computers | resolved | new laptop keyboard change within monday | Feb 27, 2025 | View, Edit, Delete |
| ZA-LTP15 | 2005 | Keyboard not functioning properly | Feb 03, 2025 | 300.00 | Peripheral Fix Co. | Pending | | N/A | View, Edit, Delete |
| ZA-LTP09 | 2003 | Hard drive failure | Jan 16, 2025 | 1000.00 | Data Recovery Inc | Resolved | | N/A | View, Edit, Delete |
| ZA-LTP04 | 2002 | Battery replacement required | Jan 19, 2025 | 1500.00 | Battery World | Pending | | N/A | View, Edit, Delete |
| ZA-LTP02 | 2001 | Overheating issues in processor | Jan 13, 2025 | 2500.00 | Tech Solutions Ltd | Approved | | N/A | View, Edit, Delete |

Fig 5.3. Hardware asset management table with asset IDs, categories, users, and OS details.



This module enumerates all hardware assets in terms of their IDs, different types, assigned users, operating systems and checkout status of such assets. Administrators can edit and update devices or even reassign them with ease. This ensures a streamlined tracking and management of all devices across the lab.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

In conclusion, the IoT-based Smart Maintenance Scheduling and Monitoring System presents a modern, efficient solution for managing industrial equipment through real-time monitoring, intelligent scheduling, and automation. By integrating IoT technology, cloud computing, and a web-based interface, the system addresses the limitations of traditional maintenance methods, which are often reactive, time-consuming, and inefficient. The use of the ESP32 microcontroller with sensors enables continuous data collection on key parameters such as power consumption, CPU temperature, and air quality. This data is processed and transmitted via the Blynk IoT platform to a cloud backend, where it is analyzed and compared against threshold values to detect potential issues before they result in equipment failure. The system's ability to send alerts and trigger automated actions, such as shutting down malfunctioning devices, greatly reduces downtime and maintenance costs. Its centralized database ensures secure, synchronized storage of historical and real-time data, providing valuable insights for performance analysis and long-term planning. The user-friendly web dashboard enhances visibility and control, allowing maintenance personnel to manage schedules, monitor equipment status, and respond to alerts from any location. Furthermore, the modular architecture of the system allows for future scalability and easy integration with existing enterprise systems. As industries increasingly shift toward automation and smart technologies, this system provides a foundation for adopting predictive maintenance strategies and achieving higher operational efficiency. With planned enhancements such as machine learning, edge computing, and mobile support, the system is well-positioned to evolve alongside Industry 4.0 trends. Overall, it offers a cost-effective, intelligent, and proactive approach to industrial maintenance, contributing to improved equipment longevity, reduced risks, and optimized resource utilization in a rapidly advancing technological landscape.

6.2 FUTURE ENHANCEMENT

Future enhancements for the IoT-based Smart Maintenance Scheduling and Monitoring System aim to significantly improve its intelligence, scalability, and user experience. Integrating machine learning algorithms will enable the system to analyze historical and real-time sensor data, accurately predicting equipment failures and optimizing maintenance schedules. Edge computing will be incorporated to allow the ESP32 to process data locally, reducing latency and dependency on constant internet connectivity. The system will also support multi-device synchronization and seamless integration with enterprise platforms such as ERP and CMMS, ensuring efficient data flow and coordinated operations. Enhanced security features, including end-to-end encryption, secure authentication, and role-based access control, will be implemented to protect critical operational data. User interaction will be improved through dedicated mobile applications for Android and iOS, providing remote access, real-time monitoring, and control. Additional features like voice command support and AI-powered virtual assistants will offer hands-free interaction and intelligent troubleshooting. Energy efficiency analytics will be added to monitor consumption patterns and suggest optimizations, contributing to sustainability goals. These advancements will collectively enhance the system's performance, reliability, and alignment with Industry 4.0 standards, ensuring it remains a robust and future-ready solution for smart industrial maintenance.

REFERENCES

- [1] B. Narayanan and M. Sreekumar, “Design, modelling, optimisation and validation of condition-based maintenance in IoT enabled hybrid flow shop,” *Int. J. Comput. Integr. Manuf.*, vol. 35, no. 9, pp. 927–941, Sep. 2022.
- [2] R. C. Parpala and R. Iacob, “Application of IoT concept on predictive maintenance of industrial equipment,” *MATEC Web Conf.*, vol. 121, p. 02008, 2017.
- [3] M. H. Rahdar, F. Nasiri, and B. Lee, “Availability-based predictive maintenance scheduling for vibrating-grate biomass boilers,” *Saf. Reliab.*, vol. 39, no. 2, pp. 165–187, Apr. 2020.
- [4] K. Belkacem, N. Bali, and H. Labdelaoui, “Optimal preventive maintenance scheduling of multi state systems. A comparative study of different meta-heuristic algorithms,” *Algerian Journal of Signals and Systems*, vol. 9, no. 2, pp. 114–120, Jun. 2024.
- [5] S. Magdum and Dr.B.F.Momin, “Predictive maintenance and preventive measures for calibration devices: A Mobile Application approach,” *July - August 2023*, vol. 7, no. 4, pp. 252–256, 2023.
- [6] Z. Wang, M. Goudarzi, and R. Buyya, “ReinFog: A DRL empowered framework for resource management in edge and cloud computing environments,” *arXiv [cs.DC]*, 2024. doi: 10.48550/ARXIV.2411.13121.
- [7] Z. Wang, M. Goudarzi, M. Gong, and R. Buyya, “Deep Reinforcement Learning-based scheduling for optimizing system load and response time in edge and fog computing environments,” *Future Gener. Comput. Syst.*, vol. 152, pp. 55–69, Mar. 2024.
- [8] P. Choppara and S. Mangalampalli, “An efficient deep reinforcement learning based task scheduler in cloud-fog environment,” *Cluster Comput.*, vol. 28, no. 1, Feb. 2025, doi: 10.1007/s10586-024-04712-z.
- [9] Z. Cao, H. Zhang, Y. Cao, and B. Liu, “A Deep Reinforcement Learning approach to multi-component job scheduling in edge computing,” *arXiv [cs.DC]*, 2019. doi: 10.48550/ARXIV.1908.10290.
- [10] D. M. Rani, Supreethi, and B. Bihari Jayasingh, “Deep Reinforcement Learning for

dynamic task scheduling in edge-cloud environments,” *Int. J. Electr. Comput. Eng. Syst.*, vol. 15, no. 10, pp. 837–850, Nov. 2024.



A Data-Driven System for Predictive Maintenance Scheduling and Environmental Monitoring

Ajitha B
Dept.of Computer Science and
Engineering
Rajalakshmi Engineering College
Chennai, India
220701019@rajalakshmi.edu.in

Aldrin Roger S
Dept.of Computer Science and
Engineering
Rajalakshmi Engineering College
Chennai, India
220701023@rajalakshmi.edu.in

Hareesh SS
Dept.of Computer Science and
Engineering
Rajalakshmi Engineering College
Chennai, India
220701079@rajalakshmi.edu.in

M.Ayyadurai
Dept.of Computer Science and
Engineering
Rajalakshmi Engineering College
Chennai, India
ayyadurai.m@rajalakshmi.edu.in

Abstract:

In modern computer laboratories and working environments, computers and laptops could be running all the time, even during idle states, resulting in unnecessary power consumption, a hefty energy bill, with the risk of hardware damage due to overheating. In this paper, we propose an intelligent scheduling system that uses IoT-based smart plugs to monitor and manage the power states of electrical devices. IoT sensors and cloud automation will be integrated into the system to record usage patterns of devices and detect inactivity, after which it may apply scheduled shutdowns or power-saving modes during non-usage hours. This will help conserve energy and extend the life of devices while also minimizing downing time due to thermal stress or overuse. However, the system was validated in a control lab environment that provided substantial energy savings while providing efficient management of assets.

Keywords: IoT, ESP32, Smart Plugs, Predictive Maintenance, Computer Lab Automation, DHT22, MQ135, Blynk, Energy Monitoring, Remote Control

I. INTRODUCTION

The rapid digital transformation of educational institutions, corporate offices, and research labs has almost made it mandatory for the increase in substantial computing devices like desktops, laptops, and workstations in the premises. These systems became a prerequisite for everyday activities such as programming, simulation,

documentation, and data processing. One of the commonly shared problems related to all these environments is leaving computers literally switched on—an example can be night, weekends, or extended periods of no use. This, apart from consuming unnecessary energy, causes overheating, faster degradation of hardware, and increased maintenance costs.

Yet most of the institutions rely on users to even carry out this protocol in power shutdown management manually or expect them to remember turning off their systems, which people often do not do, mostly due to human error or because of expectations of short-term reuse. In addition, the usual methods for saving power like sleep mode or screensavers do not suffice to address relatively longer inactivity or misconfigurations of the system. Collectively, these issues lead to energy inefficiency, environmental impact, and operational costs when scaled across dozens or hundreds of machines, however.

The birth of the Internet of Things (IoT), however, presented entirely new possibilities for automating and optimizing usage of devices much more intelligently. Smart plugs connected to IoT can also be integrated with existing infrastructure for real-time monitoring of power consumption and determine the nonuse periods of operation, all setting up an automated scheduling policy without user intervention. Smart systems come with an added advantage of providing data analytics dashboards to administrators for more informed decision-making and proactive maintenance.

Introducing schedule management systems developed particularly for computer labs and such environments feature the combination of the IoT-based smart plugs; cloud connectivity; intelligent scheduling algorithm. This aims to minimize wasteful power consumption and extend the useful lifespan of devices while on the other hand ensuring that the systems are running only when they are actually required. Testing was done on the proposed solution in a controlled lab setting, and evaluation criteria were based on energy savings, cost-effectiveness, and ease of deployment.

II. LITERATURE SURVEY

An analytical model-based maintenance scheme is presented for an IoT-enabled hybrid flow shop (HFS). The model evaluates machine health and identifies hidden risks through performance metrics such as real-time cycle time, work in progress (WIP), and squared coefficient of variance (SCV) in service time. Maintenance action is initiated by comparing these metrics against threshold values so that machine failure could be prevented. The case study of an IoT-enabled HFS validates the scheme optimized using interval type-2 fuzzy logic systems (IT2FLSs).[1]

In this paper, we present a simple technique for online monitoring and predictive maintenance in connected manufacturing systems of the future via the Internet of Things. While IoT platforms support predictive maintenance using data from a variety of machines, they all still suffer from communication incompatibility with industrial protocols. The proposed method solves this problem by enabling real-time process monitoring for quality assurance as well as condition monitoring to avoid unplanned downtimes. Provisions are made for a case study to demonstrate the feasibility.[2]

This study proposes an availability-oriented maintenance scheduling policy for vibrating-grate biomass boilers with the rationale of reducing costs in achieving the intended availability of the system. From constant failure rates, we use Weibull regression for time-dependent hazard functions and validate them with Kolmogorov–Smirnov tests. Ans maintenance impact is one of the most critical factors influencing the prioritization of the components relative to their availability impact. Based on fault tree analysis, the system availability estimation shows that the proposed predictive maintenance model can reduce 32% of the maintenance requirements.[2], [3]

This paper focuses on a comparative study of the various meta-heuristic algorithms, namely Genetic Algorithm, Particle Swarm Optimization, Artificial Bee Colony, and Grey Wolf Optimizer deployed in multi-state system maintenance optimization under time and availability constraints. The aim is to minimize total costs of preventive maintenance by optimizing the inspection and maintenance intervals. The performance measures of the algorithms were assessed based on solution quality, stability, and convergence speed, with an intent to identify the most efficient approach.[4]

In this paper ,the dissertation explores the development of easy-to-use mobile applications that focus on mechanical device maintenance including bicycles along with automobiles and air conditioners for predictive maintenance needs. Predictions related to device failure and health status and service notices are achieved through algorithm fine-tuning of an external calibration device within the application system. The application uses data from pressure controller sensors together with calibration data along with age and component failure information for optimal maintenance scheduling which reduces equipment downtime. The proposed solution increases reliability alongside efficiency through the execution of prompt preventive actions relying on acquired data.[5]

This paper presents ReinFog, a deep reinforcer learning (DRL)-based distributed framework for effectively managing resources between an edge and a cloud under a unified dynamic scheduling for an application of IoT in scalable and adaptable systems that are energy-efficient and demonstrate significant reductions in response time and cost of systems. [6]

In this paper, the DRLIS algorithm is presented; it implements deep reinforcement learning to improve the load distribution and response times for IoT applications running in edge-fog-cloud systems. The scheduler can adapt to changing environments with superiority over conventional metaheuristic and RL methods.[7]

In this paper DRLMONT, which is a deep Q-learning based task scheduling technique that efficiently allocates tasks between fog and cloud layers. The technique optimizes energy consumption, latency of processing, and cost under heterogeneous dynamic workloads in a cloud-fog environment.[8]

Application of deep reinforcement learning schedules multi-component jobs in edge-cum-cloud computing environments. In modeling inter-task dependencies and server constraints, improvement in throughput is achieved along with a reduction in job completion time under varying loads.[9]

In this paper, the establishment of a DRLF, which refers to a deep reinforcement learning framework designed to schedule dynamic tasks in edge-cloud infrastructures. The RNN architecture with deep reinforcement learning will efficiently capture the temporally inclined patterns of IoT workloads. The simulation results conducted on CloudSim and iFogSim were compared with conventional scheduling techniques and proved that the proposed reinforcement learning-based dynamic scheduling (RLbDS) algorithm achieved better adaptability and efficiency through all experimental results.[10]

III. PROPOSED MODEL

This section describes the system architecture and design methodology of the IoT-based Smart Maintenance Scheduling and Monitoring System.

A. System Architecture and Workflow

In the proposed system, three-layer architecture is implemented: The sensor/controller layer (ESP32 + sensors), the cloud/communication layer (Blynk IoT), the web application/backend layer (custom server and dashboard).

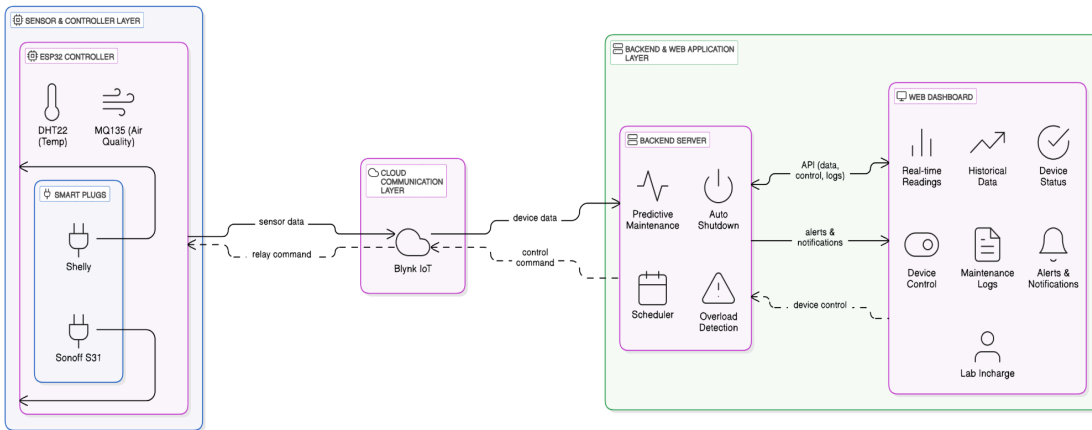


Fig. 3.1 System architecture

Shows System Architecture of IoT Maintenance System showing real time data being processed from ESP32 sensors through Blynk to backend and web dashboard.

Data from power consumption, CPU temperature, and air quality are collected and processed by ESP32. It sends these readings over Wi-Fi to Blynk, where they are visualized and sent to a backend server for checking against threshold values and automation. The backend can shut down the device if the parameters exceed the safe thresholds. All records get synchronized with a centralized secure database.

B. Database Schema Design

The back-end of the proposed IoT-based control system is under-girded by a strong and normalized relational database schema that efficiently handles data integrity, access control, and scalability. The schema consists of interrelated tables that track users, devices, sensors, sensor readings, maintenance actions, and system events. Each table is designed considering data consistency, and foreign key relationships are maintained to enforce referential integrity and facilitate secure role-based data access.



Fig. 3.2. Entity - Relationship Diagram

The Entity-Relationship Diagram of the IOT Maintenance System. This is a diagram that shows all types of relationships between users, devices, sensors, readings, events, or maintenance logs that will be possible for secure and normalized data representation.

To enhance the performance of queries, indexing into timestamp and sensor_id fields in the readings table has been deployed to enable real-time data retrieval for purposes of graph plotting and threshold comparison. Similarly, device_id and event_type fields will be indexed into the events table for rapid log access and pattern recognition.

Moreover, the schema is modular enough to horizontally scale the system. This means that adding any new lab or device would only require simple modifications in the devices and sensors tables without any effect on core logic or user interface. It also includes proper provisions for audit logging as well as compliance reporting, which are essential for the needs of institutional IT environments.

C. Maintenance Workflow

The system consists of a four-stage maintenance workflow:

1. **Data Acquisition:** Periodically poll sensor readings and store them at Blynk.
2. **Analysis & Detection:** Backend algorithm compares sensor data with threshold values. For example, the CPU should be closed if its temperature goes beyond 70°C or AQI goes above 150.
3. **Automated Control:** Shut down or create alarms with relays to smart plugs.
4. **Monitoring & Logging:** Events, status updates, etc., are displayed on a dashboard and stored in logs for later reference.

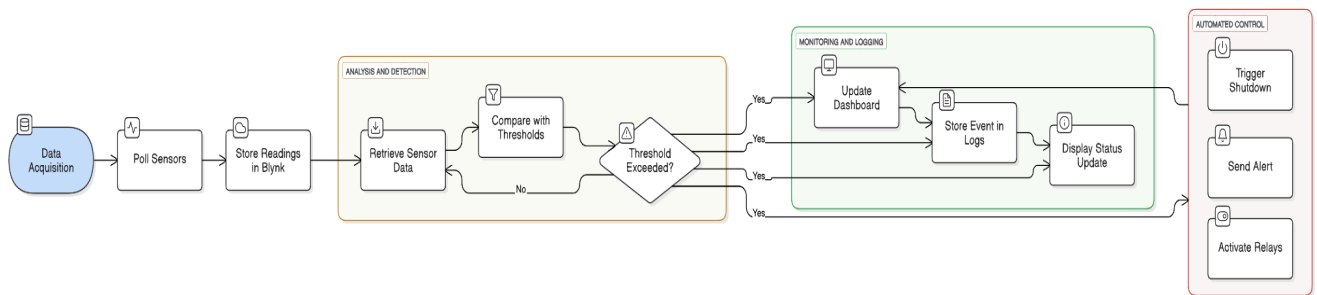


Fig 3.3. Workflow chart

Workflow chart of maintenance that sheds light on how the data of sensors gets to the backend logic in order to enable shutdowns to be undertaken and dashboards to be updated.

D. Features and Flexibility

This is a comprehensive feature set for laboratory and mass deployment room management, including a role-based web dashboard for efficient monitoring and control, especially for lab in-charges. The system continuously monitors the sensors, enabling an environmental and operational real-time alert for out-of-compliance conditions. Historical logs paired with secure access ensure data integrity and accountability. The system uses predictive scheduling through trend analysis for maintenance anticipation before failures become apparent. It is modular by design to allow seamless integration

with REST APIs and Firebase for real-time data communication while extensible to allow horizontal scaling across multi-lab or multi-campus environments. This architecture is also forward-looking because with little disruption to core infrastructure, AI models can be integrated for intelligent prediction and automations.

IV. RESULTS AND DISCUSSIONS

The maintenance scheduling system that deploys IOT is now being used in a laboratory environment of an institution and is integrated with computers that are already controlled by ESP32-based smart plug and DHT22 temperature sensors to monitor temperature and MQ135-based air quality monitoring. To evaluate the performance of the deployment with a very holistic scope, an entire web application was built with all modules including user login authentication, role-based dashboards, hardware asset management, maintenance records, user tracking, and smart notifications functions and experiences.

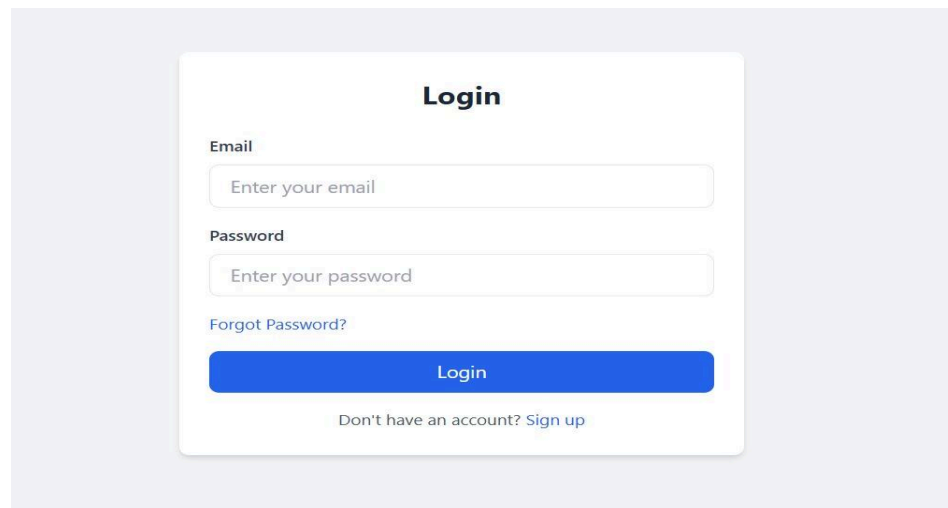
The image shows a web-based login form titled "Login" in bold black text. Below the title, there are two input fields: "Email" with a placeholder "Enter your email" and "Password" with a placeholder "Enter your password". Below the password field is a link "Forgot Password?". At the bottom of the form is a blue button labeled "Login". Below the button is a link "Don't have an account? Sign up". The entire form is centered on a light gray background.

Fig 4.1. Secure Login interface for authenticated system access

To make a secure login page strictly for authorized personnel who are logged in to the system. It authenticates the credentials according to user roles Admin, Lab-Incharge, or Technician-and subsequently redirects to an individual dashboard for that user. Secure design includes encryption and password protection for system integrity.

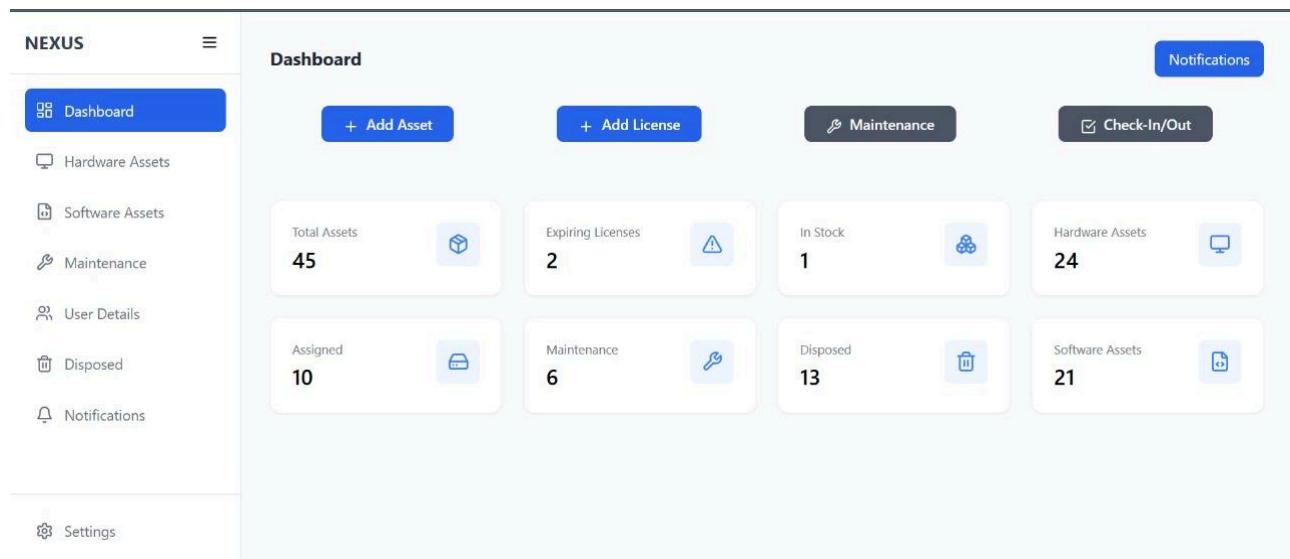


Fig 4.2. Centralized dashboard showing asset status, maintenance, and activity metrics.

The dashboard is the main control panel for lab incharges as well as administrators. This will show real-time data, including the total number of assets, license validity, maintenance alerts, check-in/check-out, and more. Visibility and operational awareness are improved across all related systems.

NEXUS

Dashboard

Hardware Assets

Software Assets

Maintenance

User Details

Disposed

Notifications

Settings

Maintenance Records

Search...

+ Add New

| Asset ID ^ | Maintenance ID ^ | Issue ^ | Expected Date ^ | Cost ^ | Vendor ^ | Approval Status ^ | Comments ^ | Request Date ^ | Actions ^ |
|------------|------------------|-----------------------------------|-----------------|---------|---------------------|-------------------|--|----------------|-------------|
| ZA-LTP12 | 2004 | Screen replacement needed | Feb 01, 2025 | 300.00 | Display Repairs Ltd | Approved | | N/A | <div></div> |
| ZA-LTP88 | 2 | keyboard issue | Mar 02, 2025 | 900.00 | Chennai computers | resolved | new laptop keyboard change within monday | Feb 27, 2025 | <div></div> |
| ZA-LTP15 | 2005 | Keyboard not functioning properly | Feb 03, 2025 | 300.00 | Peripheral Fix Co. | Pending | | N/A | <div></div> |
| ZA-LTP09 | 2003 | Hard drive failure | Jan 16, 2025 | 1000.00 | Data Recovery Inc | Resolved | | N/A | <div></div> |
| ZA-LTP04 | 2002 | Battery replacement required | Jan 19, 2025 | 1500.00 | Battery World | Pending | | N/A | <div></div> |
| ZA-LTP02 | 2001 | Overheating issues in processor | Jan 13, 2025 | 2500.00 | Tech Solutions Ltd | Approved | | N/A | <div></div> |

Fig 4.3. Hardware asset management table with asset IDs, categories, users, and OS details.

This module enumerates all hardware assets in terms of their IDs, different types, assigned users, operating systems and checkout status of such assets. Administrators can edit and update devices or even reassign them with ease. This ensures a streamlined tracking and management of all devices across the lab.

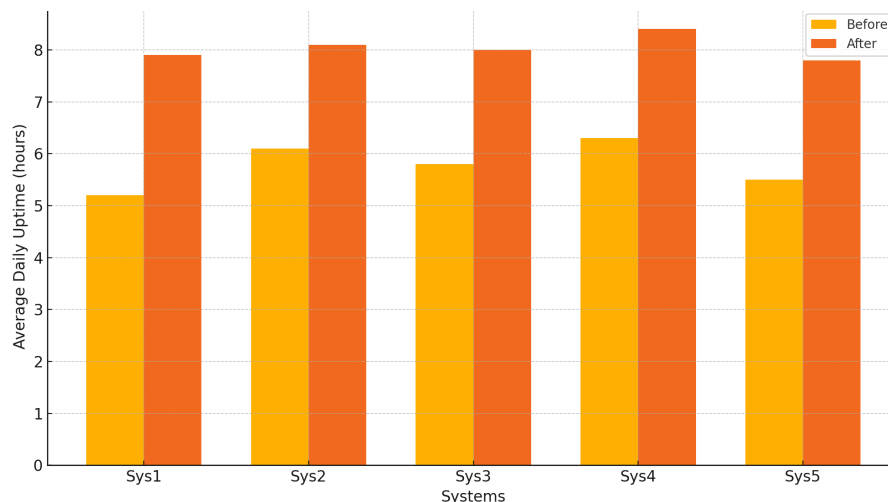


Fig 4.4. System Uptime Before vs After

Statistics of average daily uptime for the system pre- and post-utilization of the maintenance scheduler. Discussion: All five systems show highly improved operational hours: from an average of ~5.6 hours, it increased to ~8 hours per day after the intelligent scheduling system was put into operation. This is an indication that unplanned downtimes are reduced and system availability is reliable.

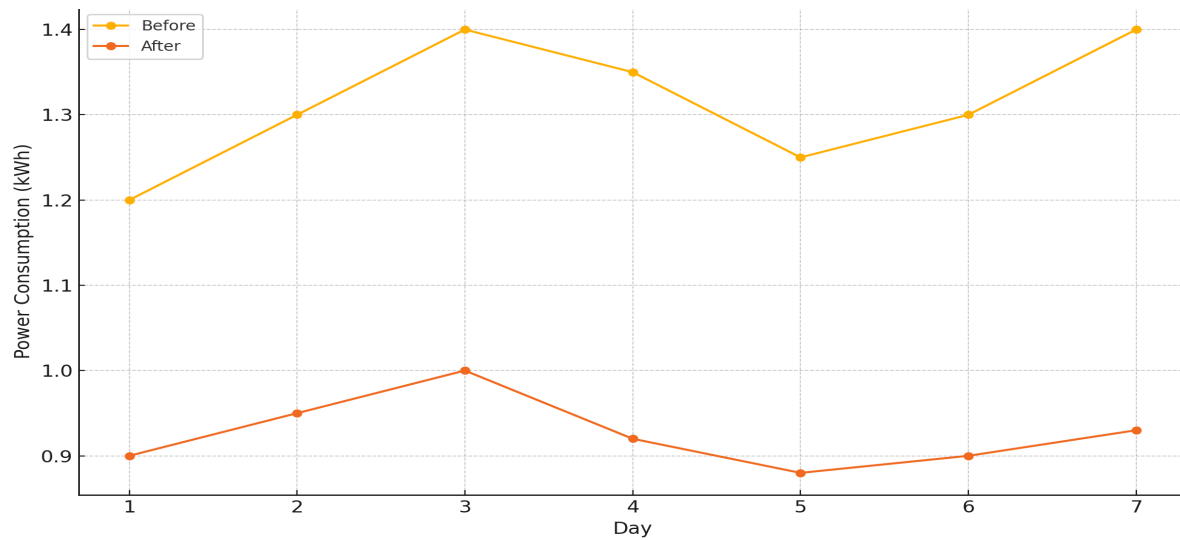


Fig 4.5. Before Installation of Scheduler vs After

The average daily power consumption of the laboratory systems before and after installation of the scheduler. Shutting systems down automatically and putting them in power-saving mode decreased the daily energy consumption from an average of 1.33 kWh down to 0.92 kWh, which translates to approximately 30% lesser power consumption. This, therefore, attests to the effectiveness of the system in conserving energy during idle states.

Several quantitative benefits of energy saving, uptime in systems, and proactive maintenance, the intelligent maintenance scheduling, mobility IoT-based smart plugs, and environmental sensors solution. The automated power management and predictive scheduling could reduce energy consumption by as much as 30% and increase overall system uptime between 2 and 3 hours daily. Early detection and automated maintenance shutdown significantly reduced hardware risks associated with thermal and overuse. Hence, the proposed system is cost-efficient, scalable for computer lab infrastructure management, increases the lifespan of devices, and decreases operational costs. Future work could include using AI models for the further optimization of predictive maintenance to potentially provide better accuracy in the fault detection and reliability of the system.

V. CONCLUSION

This article describes a well-rounded IoT approach to smart maintenance scheduling

and monitoring for institutional computer labs. The smart plug enabled by **ESP32** with sensing components like DHT22 and MQ135 has facilitated real-time acquisition of data for remote monitoring and intelligent automation of functions such as shutdown automation, overload detection, and predictive scheduling based on sensor data. A web interface, with different access levels and roles corresponding to stakeholder interaction with the system, centralizes the access of performance, maintenance records, and operational logs which enhances transparency and accountability. Its modular scalable architecture guarantees compatibility in different settings within an institution, while integration with modern platforms such as Firebase and REST APIs caters to seamless connectivity. Future developments may include AI-based predictive analytics, mobile application interfacing and wider adoption in multi-campus networks and will go a long way in contributing towards sustainable intelligent infrastructure management.

VI. REFERENCES

- [1] B. Narayanan and M. Sreekumar, "Design, modelling, optimisation and validation of condition-based maintenance in IoT enabled hybrid flow shop," *Int. J. Comput. Integr. Manuf.*, vol. 35, no. 9, pp. 927–941, Sep. 2022.
- [2] R. C. Parpala and R. Iacob, "Application of IoT concept on predictive maintenance of industrial equipment," *MATEC Web Conf.*, vol. 121, p. 02008, 2017.
- [3] M. H. Rahdar, F. Nasiri, and B. Lee, "Availability-based predictive maintenance scheduling for vibrating-grate biomass boilers," *Saf. Reliab.*, vol. 39, no. 2, pp. 165–187, Apr. 2020.
- [4] K. Belkacem, N. Bali, and H. Labdelaoui, "Optimal preventive maintenance scheduling of multi state systems. A comparative study of different meta-heuristic algorithms," *Algerian Journal of Signals and Systems*, vol. 9, no. 2, pp. 114–120, Jun. 2024.
- [5] S. Magdum and Dr.B.F.Momin, "Predictive maintenance and preventive measures for calibration devices: A Mobile Application approach," *July - August 2023*, vol. 7, no. 4, pp. 252–256, 2023.
- [6] Z. Wang, M. Goudarzi, and R. Buyya, "ReinFog: A DRL empowered framework for resource management in edge and cloud computing environments," *arXiv [cs.DC]*, 2024. doi: 10.48550/ARXIV.2411.13121.
- [7] Z. Wang, M. Goudarzi, M. Gong, and R. Buyya, "Deep Reinforcement Learning-based scheduling for optimizing system load and response time in edge and fog computing environments," *Future Gener. Comput. Syst.*, vol. 152, pp. 55–69, Mar. 2024.
- [8] P. Choppa and S. Mangalampalli, "An efficient deep reinforcement learning based task scheduler in cloud-fog environment," *Cluster Comput.*, vol. 28, no. 1, Feb. 2025, doi: 10.1007/s10586-024-04712-z.
- [9] Z. Cao, H. Zhang, Y. Cao, and B. Liu, "A Deep Reinforcement Learning approach to multi-component job scheduling in edge computing," *arXiv [cs.DC]*, 2019. doi: 10.48550/ARXIV.1908.10290.
- [10] D. M. Rani, Supreethi, and B. Bihari Jayasingh, "Deep Reinforcement Learning for dynamic task scheduling in edge-cloud environments," *Int. J. Electr. Comput. Eng. Syst.*, vol. 15, no. 10, pp. 837–850, Nov. 2024.

