HOME ENERGY MONITORING AND ALERT SYSTEM USING ESP32 AND CT SENSORS FOR REAL-TIME POWER TRACKING

BY

IVAN LING XIN YU

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

FEBRUARY 2026

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express my most sincere gratitude to my project supervisor, Dr. Mohd Hafizul Afifi bin Abdullah, for his invaluable guidance, unwavering support and insightful feedback throughout the project duration. His expertise was instrumental in navigating the technical challenges and shaping the direction of this project.

I am also grateful to the faculty and staff of the Faculty of Information and Communication Technology at Universiti Tunku Abdul Rahman for providing the academic foundation and resources necessary to undertake this project.

Special thanks are extended to my family and friends for their constant encouragement, patience, and understanding. Their support has been a significant source of motivation.

ABSTRACT

The increasing cost of electricity and a growing global emphasis on sustainable energy consumption have highlighted a critical need for real-time energy visibility within residential households. This project addresses this need by developing a cost-effective, non-invasive and Internet of Things (IoT) system for home energy monitoring and alerting. The system architecture leverages an ESP32 microcontroller, a high-precision ADS1115 Analog-to-Digital Converter (ADC), and a non-invasive SCT-013-000 Current Transformer (CT) sensor to accurately capture and process real-time electricity usage data. This data is transmitted securely over WiFi via the Message Queuing Telemetry Transport (MQTT) protocol to a robust, Docker-containerized backend. The backend consists of a Mosquitto MQTT broker, a Telegraf data agent, an InfluxDB Time-Series Database (TSDB), and a Grafana visualization platform. The resulting system provides users with an intuitive, dynamic dashboard displaying live power metrics, historical trends, and precise cost estimations based on Malaysia's tiered tariff structure. Furthermore, the system incorporates an intelligent alerting mechanism, sending notifications via a Telegram bot upon detection of anomalous power consumption. The bot also facilitates user interaction by serving on-demand dashboard panel snapshots, providing unparalleled access to energy insights. To move beyond simple alerts, the system now incorporates a Random Forest (RF) Machine Learning model. This allows for context-aware anomaly detection by learning cyclical household habits and setting dynamic power thresholds. Furthermore, an interactive Telegram bot facilitates bidirectional communication that allows users to fetch real-time Grafana snapshots and conduct smart energy audits via an inline calendar user interface (UI). This report details the design and successful validation of this intelligent system and proving its effectiveness as a robust tool for proactive energy management and machine learning-based energy analytics. The complete source code for this project are open-source and available at https://github.com/IvanLXY04/Home-Energy-Monitoring-and-Alert-System.

Area of Study: Internet of Things, Machine Learning

Keywords: Home Energy Monitoring, Anomaly Detection, Real-Time Tracking, Machine Learning, Time-Series Database, Energy Management

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

μV	Microvolt (ADC resolution step size)
Hz	Hertz (AC mains frequency)
N	Number of decision trees (in Random Forest model)
σ	Sigma / Standard Deviation (used for dynamic thresholding)
$\hat{y}$	AI Expected Power Baseline
N	Number of samples (in ADC sampling formula)
ADC_{raw}	Raw 16-bit ADC reading
$V_{rms_{adc}}$	RMS of Raw ADC Values
V_{rms}	RMS Voltage
G_{adc}	ADC gain
I_{rms}	Root Mean Square Current
R_{ct}	CT sensor ratio
C_{cal}	Software calibration constant
P	Real Power / Actual Power
$V_{nominal}$	Assumed nominal voltage
PF	Real-world efficiency factor (Power Factor)
$E_{interval}$	Energy consumed during the interval
$T_{interval}$	Send interval time in seconds
E_{total}	Total cumulative energy
$E_{previous}$	Energy consumed previously
R_{total}	Total variable rate
R_{energy}	Base variable energy rate
$R_{capacity}$	Capacity rate
$R_{network}$	Network rate
$B_{initial}$	Initial Bill Amount
B_{adj}	Bill after Automatic Fuel Adjustment
B_{net}	Bill after Energy Efficiency Incentive (Rebate)
$B_{statutory}$	Bill after Statutory Surcharges (Renewable Energy Fund)
C_{final}	Final Total Cost
E_m	Monthly energy consumption

CO_2	Carbon Dioxide footprint
$k\Omega$	Kiloohms (Resistance)
μF	Microfarad (Capacitance)
V	Volt (Voltage)
A	Ampere (Current)
W	Watt (Power)
kWh	Kilowatt-hour (Energy)
N	Total number of observations (used in the MAPE formula)
s	Seconds

LIST OF ABBREVIATIONS

AC	Alternating Current
ADC	Analog-to-Digital Converter
AFA	Automatic Fuel Adjustment
AI	Artificial Intelligence
ANN	Artificial Neural Networks
APE	Absolute Percentage Error
API	Application Programming Interface
BaaS	Backend-as-a-Service
CJMCU-TRRS	Chi Jing Micro Controller Unit-Tip-Ring-Ring-Sleeve
CT	Current Transformer
DB	Distribution Board
DC	Direct Current
EMI	Electromagnetic Interference
EN	Enable
GND	Ground
GPIO	General-Purpose Input/Output
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
ILM	Intrusive Load Monitoring
IoT	Internet of Things
IP	Internet Protocol
LCD	Liquid Crystal Display
LSTM	Long Short-Term Memory
MAPE	Mean Absolute Percentage Error
mDNS	Multicast DNS
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
NEEAP	National Energy Efficiency Action Plan

NILM	Non-Intrusive Load Monitoring
NVS	Non-Volatile Storage
PGA	Programmable Gain Amplifier
RF	Random Forest
RFR	Random Forest Regression
RMS	Root Mean Square
RTC	Real-Time Clock
RTOS	Real-Time Operating System
SBCs	Single-Board Computers
SCL	Serial Clock Line
SDA	Serial Data Line
SPAs	Single Page Applications
SPI	Serial Peripheral Interface
SVM	Support Vector Machines
TIG	Telegraf, InfluxDB and Grafana
TLS	Transport Layer Security
TRMS	True Root Mean Square
TSDB	Time-Series Database
UART	Universal Asynchronous Receiver-Transmitter
UI	User Interface
WDT	Watchdog Timer

Chapter 1
Introduction

This chapter introduces the project, covering the motivation and problem statement, the project objectives, project scope and direction, project contributions and the report organization of this report.

1.1 Motivation and Problem Statement

As the demand for electricity continues to rise and energy costs increase, efficient energy monitoring has become crucial for households. In Malaysia, the residential sector is a significant contributor to the overall high electricity demand. Many homeowners use electrical energy inefficiently because they lack awareness of how to use energy more efficiently in their daily routines. The monthly electricity bill has become even more difficult to be monitored by the consumers with the implementation of the electricity tariff restructuring as it includes complex surcharges and taxes. Some individuals reduce their electricity usage by limiting certain activities, while others opt for energy-efficient appliances. Both approaches are essential for conserving energy, however, real-time energy consumption monitoring is also vital for improving overall energy efficiency [1].

This project is motivated by the need for sustainable energy monitoring technology that can help reduce electricity waste and costs. With rising electricity prices and environmental concerns, families must use smart energy monitoring systems to optimize energy consumption. The use of IoT technology and Machine Learning (ML) in energy monitoring systems has shown great potential in terms of reducing wasted power consumption. Despite increased awareness of energy conservation, many families still lack access to low-cost, real-time monitoring systems that provide actionable insights.

Problem statement 1: Limited Visibility of Energy Usage

Numerous households depend solely on their monthly electricity statements to assess their energy usage. Nonetheless, these statements only provide a total amount without detailing how much energy each appliance utilizes or how costs are calculated under the complex tariff structure for their daily, weekly and monthly electricity usage. This absence of specific insights complicates homeowners' efforts to identify which devices contribute most significantly to their energy consumption, hindering informed decisions for reducing usage. Conventional energy meters are primarily intended for billing rather than real-time observation, leaving users unaware of their consumption patterns and their environmental impact like daily carbon footprint. Without the means to monitor their usage trends throughout the day, inefficiencies and excessive power consumption may prevail. The lack of accessible dashboards or interactive tools to visualize energy usage restrict consumer involvement in energy-saving initiatives. The unavailability of user-friendly systems results in many individuals being oblivious to how their household energy is utilized and wasted, coupled with lack of system to analyze and present clear data for tracking energy consumption and assessing the performance of individual devices. The process of real-time monitoring remains difficult for homeowners [2].

Problem statement 2: Lack of Tools for Identifying Energy Waste

Energy inefficiency arises when appliances are operated while not needed, often due to lapses in memory or design shortcomings. For example, standby power usage in devices like televisions, gaming consoles, and charger plugs can drain energy. Additionally, appliances that are inefficient, such as old refrigerators, air conditioners, or water heaters, consume more energy than necessary. Users find it challenging to compare power usage day by day without energy monitoring, it is tough to identify when their energy consumption is less efficient than usual. This inefficiency results in unnecessarily high electricity bills and escalated carbon footprints. A smart home energy consumption monitoring system equipped with automated alert features can help alleviate this concern. However, many available solutions require expensive installations and lack the intelligence necessary to automatically detect waste and recommend energy-saving strategies. As a result, users are unable to implement immediate actions to prevent unnecessary energy use. A direct and real-time automated notification system like Telegram is essential for conducting on-demand energy assessments when immediate access to the dashboard is unavailable [2].

Problem statement 3: Challenges in Detection of Faults or Anomalies

Energy anomalies such as sudden spikes in power consumption, often indicate issues with appliances or wiring. For instance, a defective air conditioner may consume considerably more energy than usual, or a malfunctioning refrigerator compressor might operate too frequently and lead to high energy expenditure. However, in most homes, these issues often go unnoticed until they result in expensive electricity bills or electrical hazards. Increased energy usage directly leads to higher energy bill, consequently raising overall production costs [2]. The prompt identification of energy irregularities could prevent expensive appliance failures. Safety hazards such as electrical fires from short circuits or overheating devices would be diminished. Research indicates that AI-driven anomaly detection in energy monitoring can decrease maintenance expenses and avoid significant appliance failures by recognizing unusual consumption patterns before they evolve into severe problems. The absence of a real-time anomaly detection system means that issues are only addressed passively rather than proactively. A smart home energy consumption monitoring system can analyze sudden spikes in energy usage in real-time, and homeowners will receive alerts through a Telegram Bot. This methodology facilitates predictive maintenance, lowering long-term repair costs and extending appliance lifespan.

1.2 Project Objectives

Project Objective 1: Provide a Real-Time Energy Visibility

The first objective is to address the problem of limited visibility by developing a system that makes household energy consumption accessible and easy to understand. This will be accomplished by designing and building a non-invasive hardware prototype that uses an ESP32 microcontroller and a CT sensor to precisely detect real-time power consumption. The collected data will be transmitted through an accurate data pipeline to a cloud backend and displayed on a user-friendly Grafana dashboard. This dashboard will replace the conventional monthly bill with live graphs and statistics and provide homeowners with real-time visibility that they currently do not have. The system will also contain a mechanism for calculating the expected cost based on Malaysia tiered tariff making the data financially useful and provide financial insights to the homeowners.

Project Objective 2: Create Tools for Proactive Energy Management

To address the absence of tools for recognising energy waste, the second objective is to develop an automated and interactive alert system that converts passive data into actionable insights. This will be accomplished by setting up a Telegram bot with enhanced bidirectional features. First, it will deliver automated, real-time "push" alerts to the user whenever the system detects that power usage has surpassed a predefined limit to alert potential energy waste from forgotten appliances or high standby loads. Second, it will function as an on-demand "pull" tool allowing users to request and receive snapshots of various Grafana dashboard panels via simple button clicks. Furthermore, the system will provide reports that calculate the user's carbon footprint and environmental impact. These tools give homeowners immediate access to advanced insights and recommendations. This will help them to compare energy usage effectively and promote good habits in energy consumption.

Project Objective 3: Build a System for Anomaly and Fault Detection

The third objective is to build a system capable of automatically detecting appliance faults and other critical anomalies using ML. To achieve this, the system will first gather a detailed history of real-time energy consumption. This data collection is the essential first step, as it allows the system to learn the normal, day-to-day energy patterns of the household. Once enough data is collected, it will be used to build intelligence into the system. The "Temporal Feature Engineering" allow to learn user habits based on the specific minute, hour and day of the week. This enables the system to look for consumption patterns that are unusual, such as at which specific time will consume far more power than it should. This approach uses context-aware dynamic thresholds and standard deviation scaling to identify what is normal for a specific time. By learning what is normal, the system can more accurately identify the signs of a failing device. This method will help to increase safety by generating meaningful alerts that notify the user about potential appliance faults.

1.3 Project Scope and Direction

This project scope includes the design, implementation, and validation of a fully functional energy management system. The major outcome at the end of the project's initial phase will be a fully operating system that includes both integrated hardware and software stack. The hardware component is designed to safely connect to household power lines for non-invasive monitoring and connect to an independent power supply with wireless diagnostic interface for easy troubleshooting of energy usage. The software component consists of two parts, an intelligent firmware for the ESP32 microcontroller to handle real-time energy data, and a containerized backend. This backend is a contemporary ecosystem of services which are databases for long-term storage, high-level visualization platform for data analysis and a ML engine that can learn and adapt the household energy consumption habits automatically.

The overall direction of this project is to provide a comprehensive and reliable design for a home energy monitoring system that emphasise user interaction and practical insights. The system was purposefully developed to overcome the constraints of a traditional data recorder. Typically, a traditional data logger is a passive device that just gathers data and presents it to the user, it is a one-way information flow. For example, an Arduino and Raspberry Pi are used in a real-time energy logger [3] to gather energy data and show it on the EmonCMS web application. In this configuration, the user cannot actively "talk back" to the system to request specific information. Instead, the system sends data for the user to observe only. It is an effective logger, but the interaction ends at the dashboard. The primary innovation that defines this project is the original integration of an interactive Telegram bot that allows for a bidirectional communication channel with users. The project direction has expanded by adding Artificial Intelligence (AI) and financial auditing. This feature enables on-demand visual reporting, real-time alerts and "Smart Energy Audit" that calculate carbon footprints. The user transforms from a passive data observer to an active participant who can ask the system for specific information at any moment. This approach motivates the project towards developing a more engaging and more effective platform for raising household energy awareness and conservation than many existing commercial solutions.

1.4 Project Contributions

Project Contribution 1: Real-Time Energy Visibility

The most important contribution of this project is the powerful tool it provides to help homeowners manage their electricity use. Majority of people only see one figure on their monthly bill which is sent weeks after their energy usage. This project modifies that by displaying energy consumption and precise costs in real time. It allows people to see clearly how and when they use electricity throughout the day. The system can learn a household's specific habits with a ML model. It can distinguish between normal daily patterns and actual energy waste. Homeowners can relate their everyday activities to their actual expenses and make more informed decisions that lead to significant cost savings.

Project Contribution 2: Interactive and Proactive Energy Management

This project also has a significant impact on the larger community. It provides a comprehensive, holistic approach on developing a modern IoT system with low-cost and open-source tools. The addition of an interactive Telegram Bot distinguishes this solution as creative and user-friendly. This feature alters how users interact with their data. Instead of only viewing a dashboard, the user's phone can receive notifications and request for instant snapshots of their home's energy consumption through the Telegram Bot. User are allowed to view their carbon footprint and environmental impact easily.

Project Contribution 3: Intelligent Anomaly Detection and System Reliability

A technical contribution of this project is the incorporation of intelligent ML model to improve home energy efficiency and safety. By using a Random Forest (RF) model, the home energy consumption monitoring system can learn the habits of a household's energy usage. It can accurately differentiate between normal daily energy patterns and actual energy anomalies. Moreover, the wireless diagnostic interface ensures the system is easy to access real-time energy data. This transforms the system from a basic data logger to smart diagnostic tool, which is easy to maintain and monitor and establish a extensible platform for future advancements

This report contains documentation of the entire system development process, making it a useful guide and learning resource that can assist and motivate others to develop, modify and expand upon this work in the fields of AI and sustainable technology.

1.5 Report Organization

This report is organized into seven chapters to guide the reader effectively through every phase of the project, from the initial concept to the final system validation. To ensure clarity, this section describes the project's key concepts. The IoT is a network of physical devices equipped with sensors and software that communicate and share data via the Internet. MQTT is a lightweight messaging protocol that works well with constrained devices and is perfect for real-time monitoring. A TSDB is a software system designed to store and query massive amounts of time-stamped data, such as sensor readings. Additionally, ML is used in this project to identify anomalies and understand human behaviour patterns, while Edge Computing enables local data processing and diagnostics. Understanding these fundamental technologies is critical for comprehending the design choices discussed in the following chapters of this work.

The following chapters describe the project execution. Chapter 2 reviews existing energy monitoring systems and the latest technologies. Chapter 3 presents the system methodology, covering the system architecture, use case requirements, and operational flowchart. Chapter 4 presents the system design, covering the system components, hardware circuit schematics, database schema, and the ML model Architecture. Chapter 5 presents the system implementation, covering the hardware setup, software setup, implementation issues and challenges. Chapter 6 presents the system evaluation and discussion, covering the accuracy of the measurements, performance and the reliability of the system. In Chapter 7, the final project results, current limitations and recommendations for future work are summarized.

Chapter 2

Literature Review

This chapter focuses on reviewing existing energy monitoring systems, systems strengths and weaknesses as well as fundamental technologies to establish the technical context and identify the project critical shortcomings.

2.1 Review of Existing Systems

2.1.1 Arduino-Based Offline Monitoring with Local Data Logging

A study by provides a foundational approach to energy monitoring by detailing the construction and validation of a portable, appliance-level energy data logger. The system architecture centered around the Arduino Uno microcontroller which is a popular choice for simple embedded projects. It is designed for Intrusive Load Monitoring (ILM), a technique in which the device is linked directly to a power outlet and a single appliance in order to record its individual energy profile. The system uses an SCT-013 current sensor to monitor current and a ZMPT101B voltage sensor to measure correct AC voltage at the same time. This architecture is distinguished by its deliberate offline nature. All data collected including exact timestamps handled by a DS3232 Real-Time Clock (RTC) module, is written directly to a physical SD card. This design choice results in a standalone and resilient device that is only dedicated to accurate data collecting and does not rely on any external network infrastructure. Figures 2.1 and 2.2 show both the electronic design and the physical realization of the prototype [4].

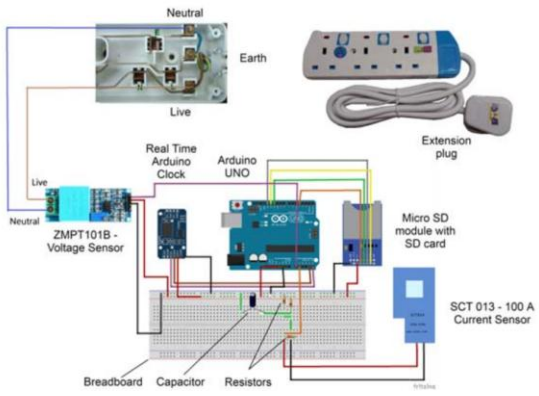


Figure 2.1 Arduino-based system design circuit [4]

Figure above illustrates an easy integration of components. The Arduino Uno acts as the core processing unit, reading analogue signals from the ZMPT101B voltage and SCT-013 current sensors using its ADC ports. The RTC and SD card modules are also attached to handle time stamping and data storage respectively.

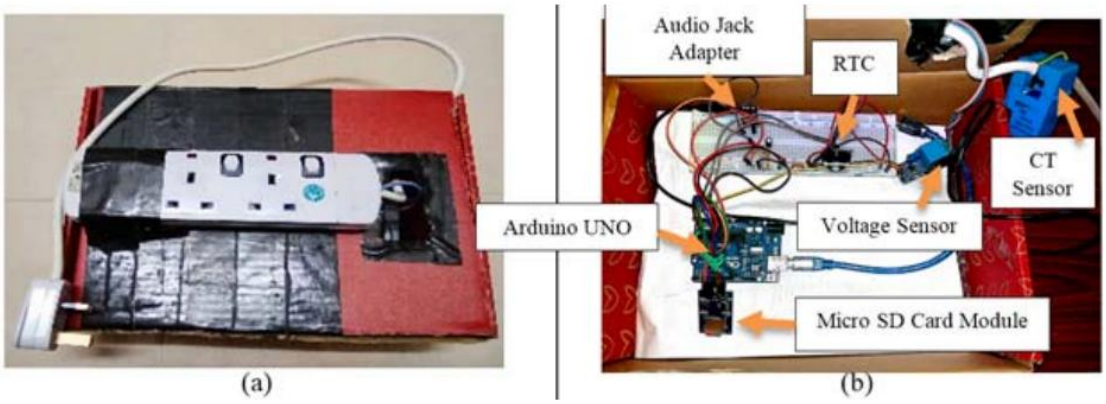


Figure 2.2 Arduino-based final completed device [4]

The figure above depicts the completed prototype placed within a tiny container demonstrating how this concept may be implemented practically. This self-contained, portable gadget features an external socket shows its intended use as a "plug-and-play" device that can be quickly transferred between different household appliances for targeted energy analysis.

Strengths

I. High Measurement Accuracy

This system's key strength is its commitment to measuring precision. By using a dedicated voltage sensor, ZMPT101B instead of an assumed nominal voltage, the device can determine genuine power rather than merely perceived power. This enables the device to attain a stated accuracy of up to 95% when compared to a commercial energy meter, making the obtained data highly dependable.

II. Inherent Data Privacy and Simplicity

As an entirely offline device, the system provides absolute data privacy. For users who are concerned about their privacy, the fact that no private household consumption data is ever sent across a network might be a big benefit. This design simplicity also results in a relatively robust system with fewer points of failure because it is resistant to network outages and cloud service disruptions.

III. Portability and Flexibility

The "plug-through" design makes the device extremely portable and adaptable. It can be readily relocated to monitor any appliance with a standard socket, allowing for extensive, targeted analyses of device consumption patterns ranging from the periodic nature of a refrigerator to the high-power draw of a kettle.

Limitations

I. Lack of IoT Connectivity

The system most major and fundamental drawback is a complete lack of network connectivity. This data-trapped architecture means that all valuable information is kept on a physical SD card. The system inability to transfer data wirelessly renders it unsuitable for any application that requires real-time monitoring, remote access or automatic data aggregation.

II. No Remote Visualization

The absence of a remote user interface is a direct result of its offline nature. To analyse the data, the user must physically remove the SD card from the device, transfer the log files to a personal computer and then manually input them into spreadsheet software. This time-consuming approach hinders dashboards from displaying real-time data or analysing trends.

III. Inability to Provide Real-Time Alerts

The system cannot provide real-time alerts or notifications. Anomalies such as an appliance consuming too much power or a gadget failing to turn off could only be discovered retrospectively by manually analysing the logged data. This will be undermining the aim of proactive energy management.

2.1.2 NodeMCU-Based Cloud Monitoring via Backend-as-a-Service (BaaS)

A more current and comprehensive approach to IoT energy monitoring describes a complete flow of monitoring and an alarm system based on the powerful ESP32 NodeMCU microcontroller. The system is designed to gather both current and voltage data that allow for precise actual power estimations. This architecture is distinguished by its dual-mode interface that can give users both a local 16x2 Liquid Crystal Display (LCD) screen for quick, on-site readings and a remote web page for more in-depth historical data analysis. The backend design differs significantly from fully open-source or local-only systems since it uses Google Firebase, a popular Backend-as-a-Service (BaaS) technology for cloud data storage and management [5]. Figure 2.3 shows the block diagram of the monitoring and alert system and Figure 2.4 illustrates the resulting schematic of the system.

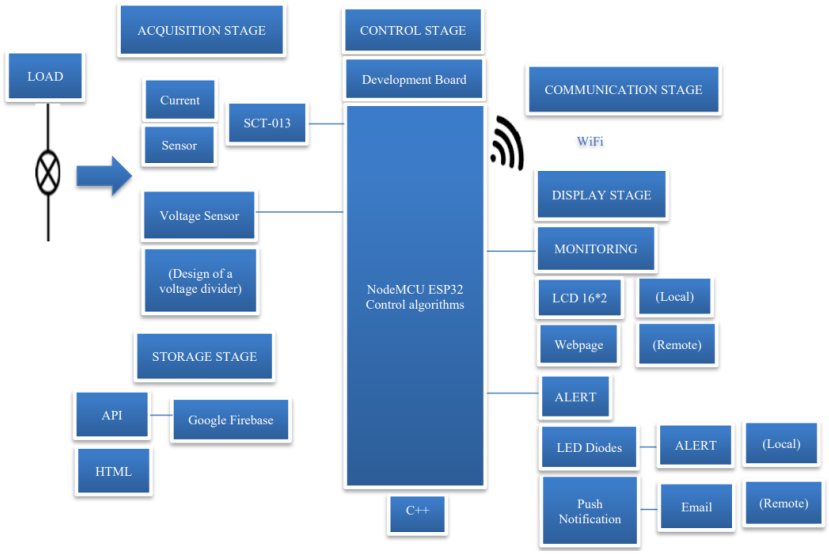


Figure 2.3 NodeMCU-based block diagram of the monitoring and alert system [5]

The figure above shows the system architecture as conceptually split into several stages, acquisition (sensors), control (ESP32), communication (Wi-Fi), display (LCD and Webpage), storage (Firebase), and alerting (Email). This modular architecture is suggestive of a strong technical approach.

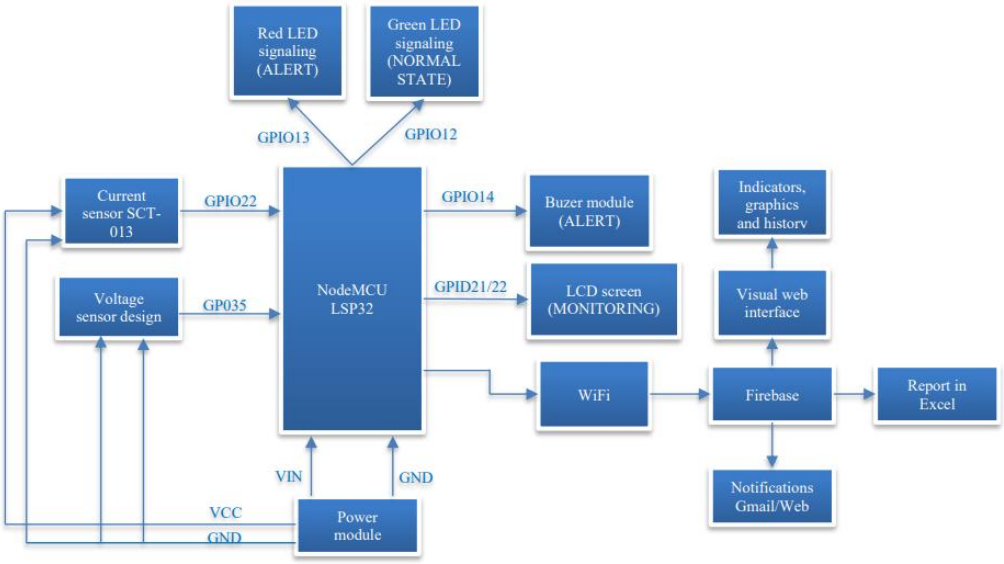


Figure 2.4 NodeMCU-based resulting schematic of the system [5]

The figure above shows the practical integration of these stages with the ESP32 NodeMCU acting as the central hub, connecting the voltage and current sensor inputs to the various output and communication modules, such as the LCD screen, a buzzer for local alerts, and a Wi-Fi module for cloud connectivity. This comprehensive and well-documented architecture marks a considerable step forward from simple data loggers to a fully functional and interactive IoT system.

Strengths

I. High End-to-End Accuracy and Rigorous Validation

This work dedication to verifying the accuracy of the system is one of its strengths. The measurements of the prototype are compared to calibrated, high-precision Fluke instruments in the comprehensive error analysis. There is great confidence in the data's dependability due to the results incredibly low error margins for both voltage (0.39%) and current (3.12%) and high system efficiency of 97.9%.

II. Complete System with a Holistic User Interface

The system offers a comprehensive user experience by reacting to a variety of requirements. A local LCD display gives rapid statistics without the need for an internet connection, whereas the remote web interface enables access to extensive historical graphs and data resulting in a more comprehensive and flexible monitoring solution.

III. Functional Proactive Alerting System

The combination of email and web-based push notifications for power outages creates a full feedback loop. This transforms the system from a passive data recorder to a proactive monitoring tool that may actively alert the user to significant events. It is an essential feature for good energy management.

Limitations

I. Dependence on a Proprietary Cloud Platform

Using Google Firebase as the backend adds a lot of vendors lock-in, even though it is helpful for quick development. The pricing structure, feature availability, and any service changes or discontinuations of the platform are all subject to the user. Data ownership and long-term stability provided by an open-source, self-hosted database solution are forfeited in this method.

II. Limited Visualization Power and Flexibility

The system remote interface is a custom-created web page. While it is useful for presenting basic trends, it lacks the advanced data exploration, querying and powerful visualisation features of a specialised, purpose-built analytics platform such as Grafana which enables far more detailed and dynamic data analysis.

III. Dated and Less-Interactive Alerting Mechanism

While the alerting mechanism is functional, email is a less immediate notification technique than modern instant messaging apps. The system runs on a one-way push approach and lacks an interactive component that would allow a user to actively ask the system for specific data or reports as needed, thus restricting user involvement.

2.1.3 Raspberry Pi-Based Centralized Monitoring via Local Web Hosting

An IoT system for monitoring power directly at a domestic electrical distribution box demonstrating a completely new architectural concept. Instead of sensing with a standalone microcontroller and processing on a separate server, this architecture uses a Raspberry Pi 3, a full single-board computer as its unified central processing unit. This device communicates with the well-known SCT-013 current sensor and ZMPT101B voltage sensor to collect high-fidelity electrical data. This approach is distinguished by its self-contained nature, the raspberry Pi is powerful enough to handle not only data collecting but also local Python Flask web server. This allows it to present monitoring data via a browser-based interface without requiring any additional, specialized backend infrastructure or cloud services. Figures 2.5 and 2.6 successfully demonstrate this all-in-one architectural solution [6].

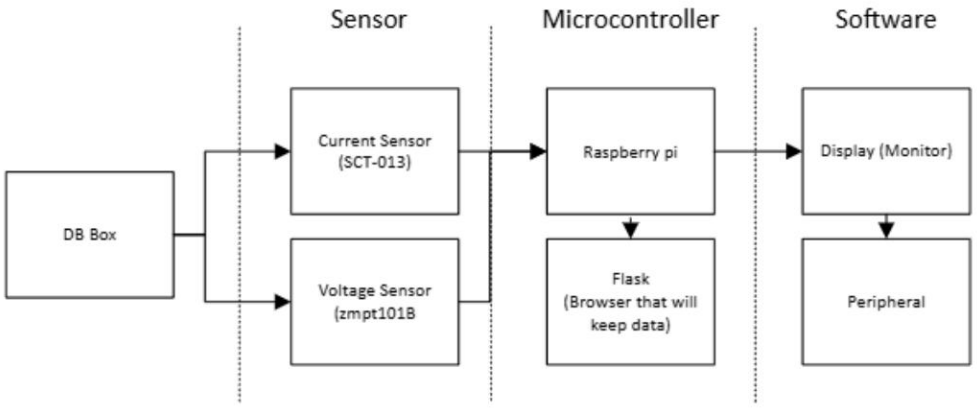


Figure 2.5 Raspberry Pi-based block diagram of IoT [6]

According to the block diagram in the figure above, the Raspberry Pi serves as the operation's single brain directly connecting the input sensors to the output display (a monitor and a web browser). The system is not distributed but it is a centralized unit with all functions contained on a single board.

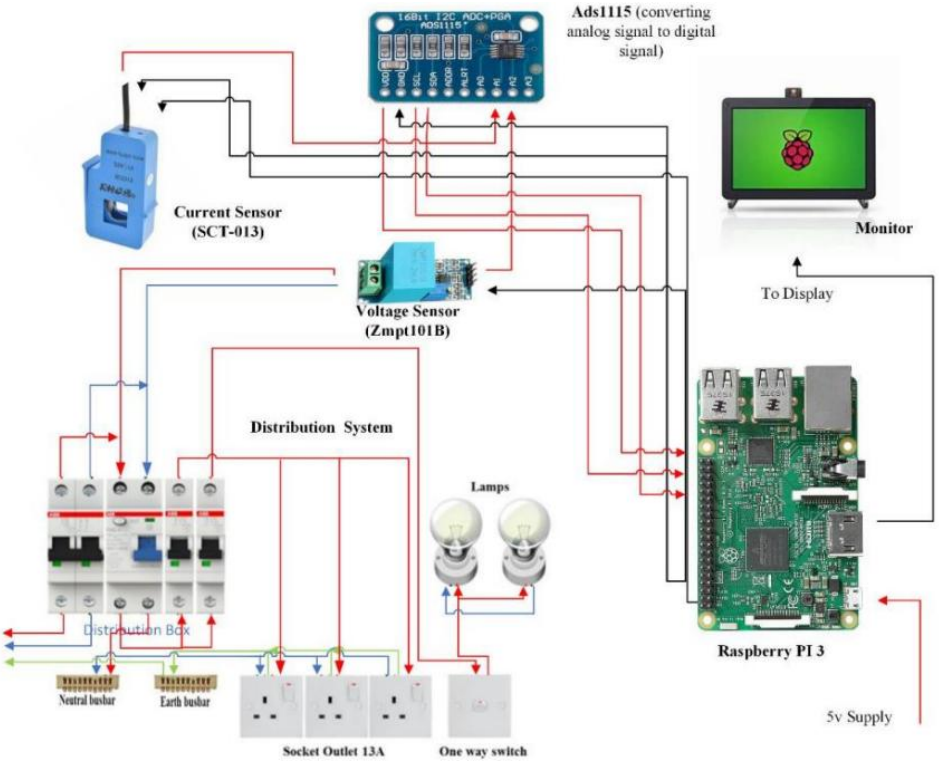


Figure 2.6 Raspberry Pi-based schematic diagram of IoT [6]

The figure above illustrates the cabling from the sensor interface (an ADS1115 ADC which is required because the Raspberry Pi has a built-in analog connection) to the Raspberry Pi's General-Purpose Input/Output (GPIO) pins. This approach combines all system logic, including sensing, processing, data handling, and visualization on a powerful hardware.

Strengths

I. Powerful On-Site Processing

A Raspberry Pi multi-core ARM processor provides much more processing capability than a microcontroller like the ESP32. This enables it to execute more advanced data analysis, operate a full-featured web server and manage data logging locally. All without being limited by processing power or memory.

II. Simplified Deployment Architecture

This technique improves total system deployment by combining all functionalities onto a single unit. There is no need to set up, administer, or maintain a separate backend server, cloud services or network connectivity between them. This can reduce the number of potential sites of failure while also simplifying initial setup.

III. High Extensibility via Linux OS

The Raspberry Pi runs a full Linux operating system (Raspberry Pi OS) that allows for more flexibility. The platform is easily extensible with a big library of open-source software, applications and programming languages. This provides a wide range of potential additions that go far beyond simple monitoring.

Limitations

I. Higher Hardware Cost and Power Consumption

The Raspberry Pi's power and flexibility are mostly offset by its high cost and energy consumption. The board is much more expensive than an ESP32 and its greater power consumption makes it a less cost-effective and energy-efficient option for an always-on sensor.

II. Inefficient Data Storage for Scalable Analysis

The measurement data is stored in a simple.txt file. This strategy is inherently unsuitable for working with time series data. It is difficult to query efficiently, does not scale well over time and does not support advanced data aggregation or complex analysis, which are common in modern monitoring systems. This contrasts sharply with the queryable and efficient storage given by a purpose-built TSDB, such as InfluxDB.

III. Lack of Proactive Alerting and Modern Interactivity

The system is designed for passive data visualization that requires the user to connect to the local network and visit the Flask web page to verify the status. The paper does not mention a real-time, push-based warning system or a modern user notification method like a Telegram bot. This reduces its usefulness for proactive energy management and immediate anomaly detection.

2.1.4 ESP8266-Based Platform-Centric Monitoring via Blynk IoT

An energy monitoring system designed using the commercial Blynk IoT platform and the older-generation ESP8266 microcontroller is a popular and approachable method for quick IoT creation. The system measures energy usage with an SCT-013-000 current sensor and uses the Blynk ecosystem to handle all backend services, such as the database, web dashboard, and mobile application. This architecture is quite similar to many early-stage IoT initiatives that prioritize speed of implementation and usability over system governance and data ownership. The system also includes a relay module, which adds a control component to its monitoring capabilities. Figures 2.7 and 2.8 show how simple and platform-centric of this architecture [7].

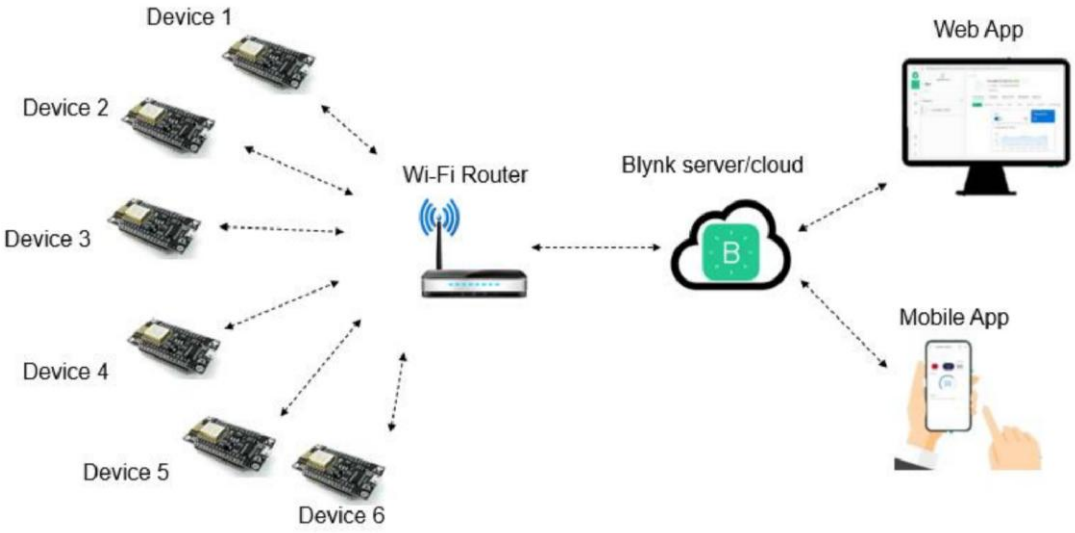


Figure 2.7 ESP8266-based system architecture [7]

The figure above depicts a high-level data flow in which ESP8266 devices communicate with the Blynk cloud server via a Wi-Fi router that provides data to a web application and a mobile app. This eliminates the requirement for any self-hosted intermediary services.

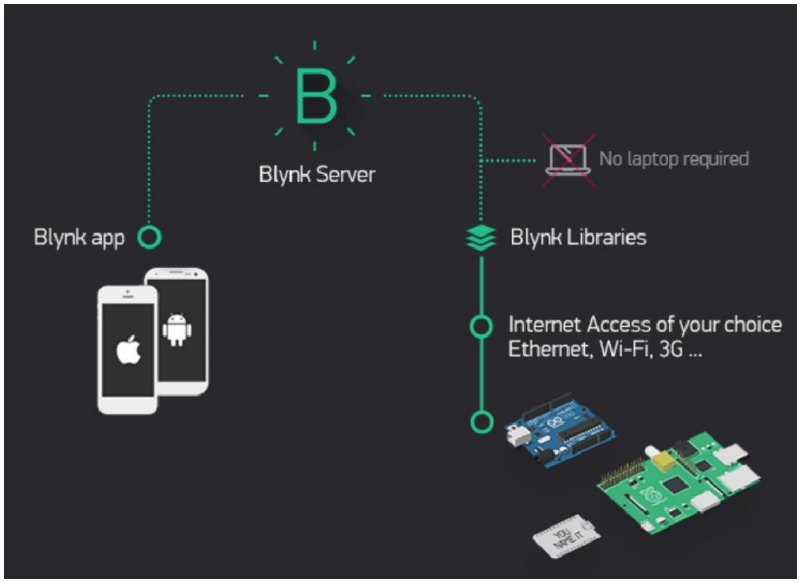


Figure 2.8 ESP8266-based Blynk architecture [7]

The figure above provides more information about the Blynk ecosystem and highlights its primary value proposition, a hardware-independent platform that links devices to a user-friendly mobile app with little backend configuration required by the developer. This architectural method distinguishes itself by its dependency on a third-party platform.

Strengths

I. Simplicity and Speed of Development

The most significant benefit of this technique is a major decrease in development complexity and time. By using an integrated platform like Blynk, the developer is fully removed from the tedious duties of installing a server, implementing a database, maintaining a communication protocol like MQTT and creating a UI from scratch.

II. Integrated Mobile and Web Dashboards

Blynk offers a variety of ready-to-use widgets for mobile and web dashboards. This drag-and-drop functionality enables the rapid design of a functioning and visually beautiful UI without the need for any specialized front-end web development abilities. This leads to extremely accessibility to developers of all levels.

III. Extremely Low-Cost Hardware

The usage of the ESP8266 which is a low-cost Wi-Fi microcontroller makes the system hardware incredibly affordable. When combined with the free tier of the Blynk service, the total cost of developing a functional prototype is low.

Limitations

I. Complete Vendor Lock-In and Lack of Data Ownership

The primary limitation of this architecture is its whole dependency on the third-party Blynk service. The user does not own or control their data because it is all stored on Blynk's servers. This raises serious concerns about possible membership fees because platforms frequently change their price structures, feature restrictions and even the chance of the service to be stopped which would stop the system.

II. Limited Customization and Scalability for Advanced Analysis

While the Blynk dashboard is simple to use, it has less options for customization and advanced data analysis than a specialized platform like Grafana. Users are limited to the widgets and data processing capabilities offered by Blynk, preventing complex querying, data correlation and the development of highly personalized visualizations.

III. Use of Less Powerful Hardware

The system depends on the ESP8266 with fewer peripherals and less processing power than its successor, the ESP32. This option reduces the device capacity to do more complex on-device operations, such as running multiple communication protocols concurrently or performing more demanding real-time calculations without performance loss.

2.1.5 Summary and Comparison of Existing Systems

Table 2.1 Comparison of strengths, weaknesses, and relevance of existing systems

System	Strengths	Weaknesses	Comments & Justification in this project
[4]	• Accurate sensing with voltage sensor. • Fully offline, ensuring data privacy. • Simple and robust design.	• No IoT connectivity or real-time data. • Requires manual data retrieval via SD card. • No remote dashboard or automated alerts.	• Serves as a non-IoT baseline. • This project adds the crucial IoT layer to this accurate sensing foundation, making the data accessible and actionable.
[5]	• High, validated accuracy (97.9%). • Dual interface (local LCD & remote web). • Includes functional email alerts.	• Vendor lock-in via Google Firebase. • Limited visualization power vs. Grafana. • Email alerts are not interactive.	• Acts as a direct competitor. • Open-source stack provides full data ownership. • Interactive Telegram Bot is a significant improvement over one-way email alert.
[6]	• Powerful on-device processing. • Simple all-in-one architecture. • Flexible and extensible via Linux OS due to running a full Linux OS.	• Higher cost and power consumption. • Inefficient data storage (txt file). • Lacks a proactive alerting system.	• Justifies the efficient, distributed design (low-power edge and strong backend). • TSDB is superior for data analysis. • ESP32 is more cost-effective and energy-efficient for a sensor node.
[7]	• Very fast and simple to develop. • Integrated mobile/web UI with widgets. • Low hardware cost (ESP8266).	• No data ownership (Blynk platform). • Limited dashboard customization. • Uses older, less powerful hardware.	• Highlights the trade-off between speed vs. control. • Prioritizes data ownership and customizability over rapid setup. • Use of the ESP32 is a clear hardware upgrade.

[4] indicates Arduino-Based Offline Monitoring with Local Data Logging

[5] indicates NodeMCU-Based Cloud Monitoring via BaaS

[6] indicates Raspberry Pi-Based Centralized Monitoring via Local Web Hosting

[7] indicates ESP8266-Based Platform-Centric Monitoring via Blynk IoT

2.2 Review of Technologies

2.2.1 ESP32 Microcontroller for Edge-level Data Processing

The option of a central processing unit for an energy monitoring system needs to achieve a balance between computational capacity, connectivity and cost-effectiveness. Recent research emphasises the ESP32 microcontroller as a superior middle-ground option that extend the gap between low-resource microcontrollers like the Arduino Uno and high-power Single-Board Computers (SBCs) like the Raspberry Pi [8]. The ESP32's dual-core Xtensa® 32-bit LX6 architecture enables it to perform multi-threaded tasks, such as concurrent high-frequency sensor sampling and wireless data transmission more successfully than its predecessors [8]. This dual-core capability is critical for edge-level processing which requires data to be normalised and computed locally before being broadcast to a MQTT broker. Figure 2.9 shows the ESP32 functional block diagram.

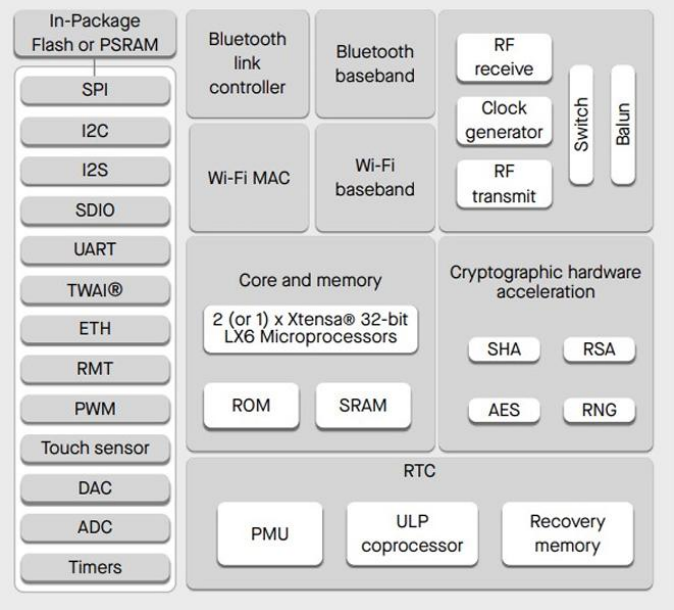


Figure 2.9 ESP32 functional block diagram [8]

The figure above shows the inbuilt hardware complexity of the ESP32 Microprocessor. It emphasises the dual-core CPU integrated Wi-Fi/Bluetooth stacks and several peripheral interfaces such as ADC, Inter-Integrated Circuit (I2C) and Serial Peripheral Interface (SPI). This figure supports the selection of a 16-bit external ADC by illustrating the placement of the internal 12-bit ADC and explains how the dual-core design handles simultaneous sensing and MQTT transmission.

Aside from raw processing capability, the ESP32 is getting recognised for its function as a lightweight IoT gateway in smart home settings. The ESP32 can operate as an independent diagnostic and control hub with minimum memory overhead, often utilising just 3.6% to 6.8% of its available RAM for normal telemetry operations [9]. This efficiency enables the utilisation of headless diagnostic tools like WebSerial that allows developers to remotely monitor system health via a web interface without the need for physical tethers. Furthermore, the incorporation of mDNS (Multicast DNS) into the ESP32 framework addresses the issue of dynamic Internet Protocol (IP) addressing in residential networks, ensuring the gateway stays available via a constant local hostname [9]. Figure 2.10 shows the RAM usage and performance metrics under different workloads.

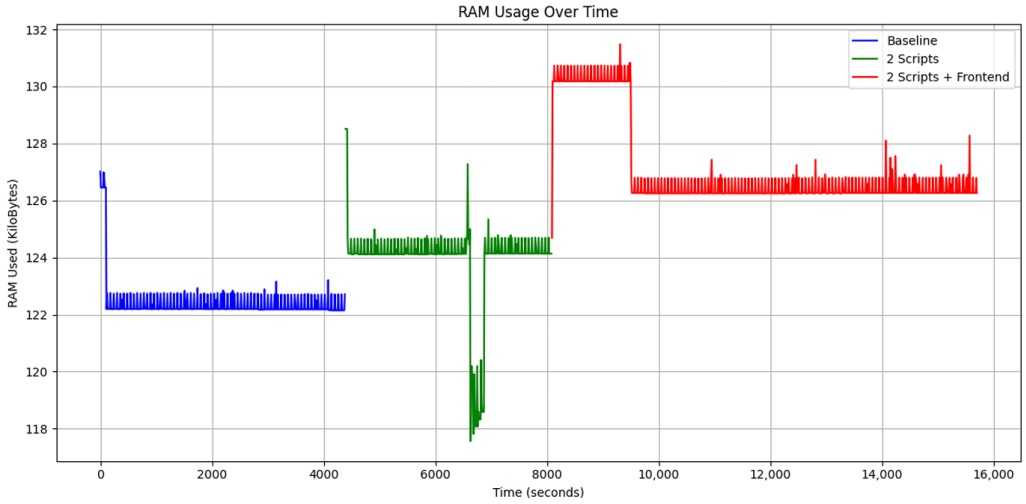


Figure 2.10 RAM usage and performance metrics under different workloads [9]

The figure above displays the ESP32 stability when performing various software functions, such as executing telemetry scripts and hosting a web-based dashboard. It gives actual proof that the ESP32 may accomplish edge computing and headless diagnostics without crashing, since memory use remains stable even when hosting interactive Single Page Applications (SPAs).

Energy efficiency remains an essential consideration for always-on monitoring systems. Although using a Real-Time Operating System (RTOS) such as FreeRTOS increases energy usage by roughly 27.8% related to task scheduling overhead, implementing Deep Sleep modes may reduce total power consumption by a remarkable 83.38% [10]. For systems that require data durability, such as the energy odometer logic in this project, the ESP32 Non-Volatile Storage (NVS) feature enables the periodic storage of cumulative data to flash memory. This ensures that essential energy measures survive power disruptions or reboots and making the ESP32 a strong and stable platform for long-term home energy analytics [10].

2.2.2 High-Precision Current Sensing via SCT-013 Sensor and ADS1115 ADC

An energy monitoring system accuracy is mainly defined by its capacity to gather and digitise analog signals from the electrical load. Non-invasive monitoring is preferred in home situations due to its safety and convenience of implementation. The YHDC SCT-013-000 CT sensor has emerged as the standard sensor for such applications to allow the measurement of alternating current (AC) by clamping around a live wire without needing physical changes to the wiring [11]. The CT sensor uses electromagnetic induction to create a secondary current proportional to the primary current flowing through the wire. This research emphasises that for this signal to be meaningful, it must be sent via a load resistor to convert the current into a reading voltage and filtered using a DC bias circuit [12]. This signal conditioning centers the waveform at mid-supply (usually 1.65V for a 3.3V system) ensure that the ADC can collect both the positive and negative peaks of the AC cycle without signal clipping [12]. Figure 2.11 shows the signal conditioning and Direct Current (DC) bias circuit.

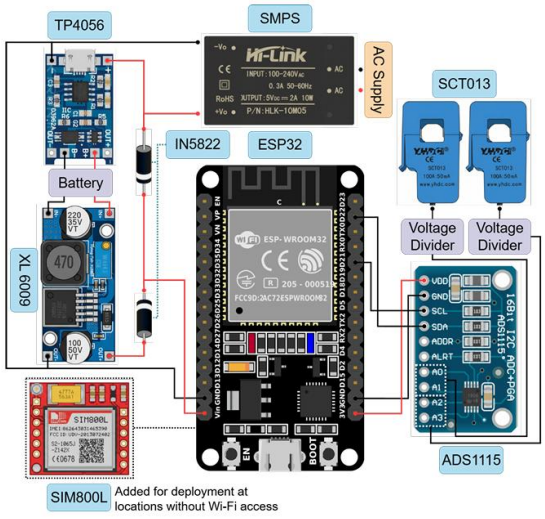


Figure 2.11 Signal conditioning and DC bias circuit [12]

The figure above depicts the hardware interaction between the CT sensor and ADC. It depicts the burden resistor used for current-to-voltage conversion, as well as the voltage divider circuit that adds a 1.65V offset to the AC signal. It describes how the system processes raw sensor data for correct digitisation.

While the ESP32 microcontroller has an inbuilt 12-bit ADC, it is frequently stated as unsuitable for high-precision energy metering due to its intrinsic nonlinear behaviour at low voltage ranges and electrical noise [12]. To address these constraints, this study recommends integrating an external 16-bit ADS1115 ADC. It is found that the ADS1115 has a greater resolution of 65,536 levels than a 12-bit converter with 4,096 levels. This results in a precise step size of roughly 76.3 µV throughout a 5V range [11]. This high level of fidelity is crucial for identifying weak signals, such as the standby power consumption of tiny equipment which usually accounts for 5% to 10% of total home energy loss [11]. Furthermore, the ADS1115 contains an integrated Programmable Gain Amplifier (PGA) and connects with the ESP32 using the I2C protocol. This simplifies the hardware design while preserving good signal integrity [13]. Figure 2.12 illustrates the comparative ADC resolution and the math logic.

F. ADS1115 Analog-to-Digital Converter (ADC)

Many microcontrollers, including the ESP32, offer internal ADCs. However, these are often limited to 10–12 bits, result- ing in low resolution, especially for small analog signals. The ADS1115 offers:

- 16-bit resolution (2^{16} = 65,536 levels).
- 4 single-ended or 2 differential channels.
- Internal PGA (Programmable Gain Amplifier) for signal amplification.
- I^2C communication protocol.
- Sampling rate up to 860 samples per second.

The resolution (step size) for a 16-bit ADC over a 5V range is :

$$\Delta V = 5V/2^{16} \approx 76.3\ \mu V \qquad (5)$$

This level of precision is critical for accurately sensing small voltage drops, especially in ground fault detection where signals are weak.

Figure 2.12 Comparative ADC resolution and math logic [11]

The figure above shows the mathematical benefit of ADS1115. The 16-bit architecture enables a lower step size (76.3µV) compared to the ESP32 inbuilt ADC, justifying the higher hardware cost for high-precision monitoring.

The temporal accuracy of the data collecting procedure is also significant for determining True Root Mean Square (TRMS) values. ADS1115 systems are frequently designed with high-frequency sampling rates to capture higher-order harmonics in the AC waveform. Employing an 860 SPS (samples per second) rate ensures that the system meets the Nyquist criteria for 50 Hz mains and collecting at least 17 samples each cycle [13]. When combined with a RTOS such as FreeRTOS, the ADC sampling job can be prioritised and given to a specific CPU core. This deterministic scheduling eliminates delays induced by background operations like as Wi-Fi connectivity, making sure that every 100ms capture window offers a precise picture of immediate power [13]. Combining the non-invasive safety of the SCT-013 and the 16-bit precision of the ADS1115, the system achieves an error margin of less than 2.5% and match the performance of commercial-grade power meters [11]. Figure 2.13 shows the RTOS task architecture for ADC sampling.

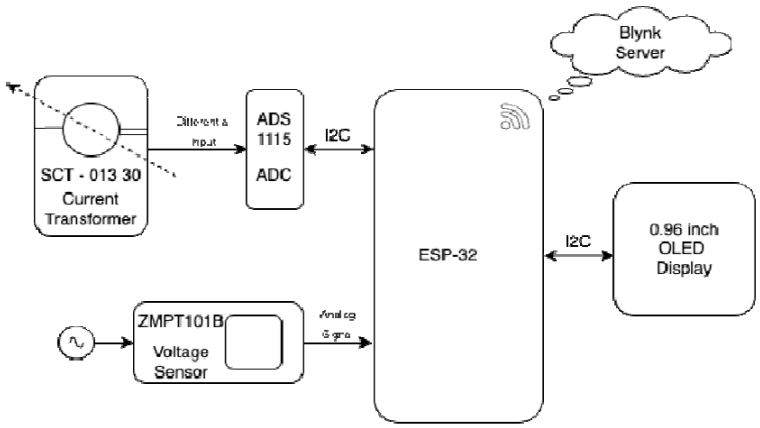


Figure 2.13 RTOS task architecture for ADC sampling [13]

The figure above depicts the Watt Watcher system architecture that emphasis on the I2C connection between the ADS1115 and the ESP32. It illustrates how the system separates time-critical sensing operations from non-critical display and cloud tasks by leveraging the ESP32 dual-core capability.

2.2.3 MQTT Messaging Protocol for Lightweight Data Transmission

In present IoT ecosystems, choosing a communication protocol is critical to assure system responsiveness and power efficiency. MQTT has become the industry standard for energy monitoring due to its low overhead and effective use of network resources. Unlike the Hypertext Transfer Protocol (HTTP) uses a request-response format that needs extensive header data, MQTT works on a Publish/Subscribe architecture [14]. This design depends on a central broker to operate as an intermediary, thereby separating the data producers (ESP32 clients) and the data consumers (backend databases or mobile apps) [15]. This eliminates the requirement for the ESP32 to know the IP address of the end destination enables a more scalable and flexible network design in which several clients can receive the same energy data with little latency. Figure 2.14 depicts MQTT publish/subscribe flowchart.

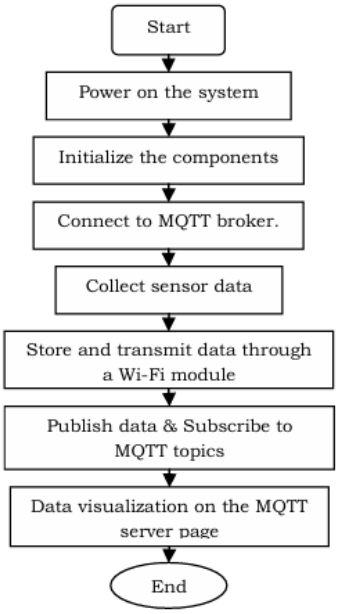


Figure 2.14 MQTT Publish/Subscribe flowchart [15]

The figure above depicts the structural relationships between system components. The ESP32 is shown as the publisher providing data to a topic on the MQTT Broker that subsequently pushes the information to the subscribers (InfluxDB backend and Telegram Bot). This image supports the usage of a broker to efficiently handle data flow.

The integration of MQTT into energy-specific data pipelines is especially useful for real-time visualisation. MQTT functions as the communication backbone for the TIG

(Telegraf, InfluxDB and Grafana) stack, which is used to monitor smart energy households [14]. In this architecture, the ESP32 sends a JSON payload with power measurements to a specified topic on the broker. A data agent such as Telegraf, subscribes to this topic and automatically uploads the telemetry to a TSDB. According to research, this event-driven technique enables high-frequency data updates often every few seconds without overloading the home network or the microcontroller processing core. This gives homeowners the live energy visibility needed to identify abrupt power spikes [14]. Figure 2.15 shows MQTT data flow in the TIG stack.

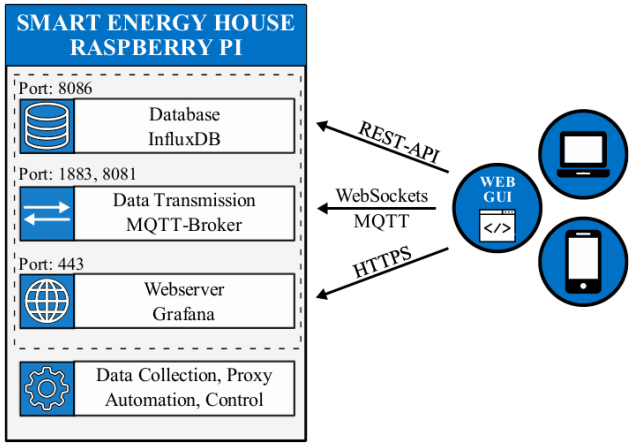


Figure 2.15 MQTT data flow in the TIG stack [14]

The figure above shows actual data for why MQTT is lightweight. By comparing the small packet size of MQTT to the large headers of HTTP, the study demonstrates that the selected protocol saves battery life on the ESP32 and lowers data costs for the user.

Aside from speed, MQTT includes built-in dependability measures via its Quality of Service (QoS) levels, which are essential for high-stakes alerting systems. A review of MQTT-based architectures demonstrates that certain jobs need varying levels of delivery guarantees. Real-time telemetry normally uses QoS 0 (At most once) to minimise overhead, but essential events, such as a “High Power Alert” or a directive to retrain a ML model, use QoS 1 (At least once) or QoS 2 (Exactly once) to ensure the message is never lost [14]. Furthermore, when MQTT is connected with Transport Layer Security (TLS), it creates a secure route for critical energy usage data and preserving user privacy from unauthorised access. MQTT is the best option for this project always-on monitoring needs due to its low

power consumption, efficient message delivery and native support for asynchronous communication [14].

2.2.4 Time-Series Analytics and Visualization via InfluxDB and Grafana

The high frequency of energy monitoring data needs a specialised storage architecture capable of processing massive volumes of timestamped data with little delay. Traditional relational databases frequently suffer with the high-velocity ingestion rates of IoT sensors. Hence, contemporary frameworks use TSDB, such as InfluxDB. InfluxDB organises data into a hierarchical structure that includes buckets, measurements, tags and fields [16]. This structure is ideal for time-series aggregation with tags providing static metadata for filtering and fields storing real changing power levels. This architecture decision assures that the system can execute effective queries over lengthy historical periods without degrading performance, giving a single source of truth for household energy usage [16]. Figure 2.16 illustrates the cloud-based infrastructure and security framework.

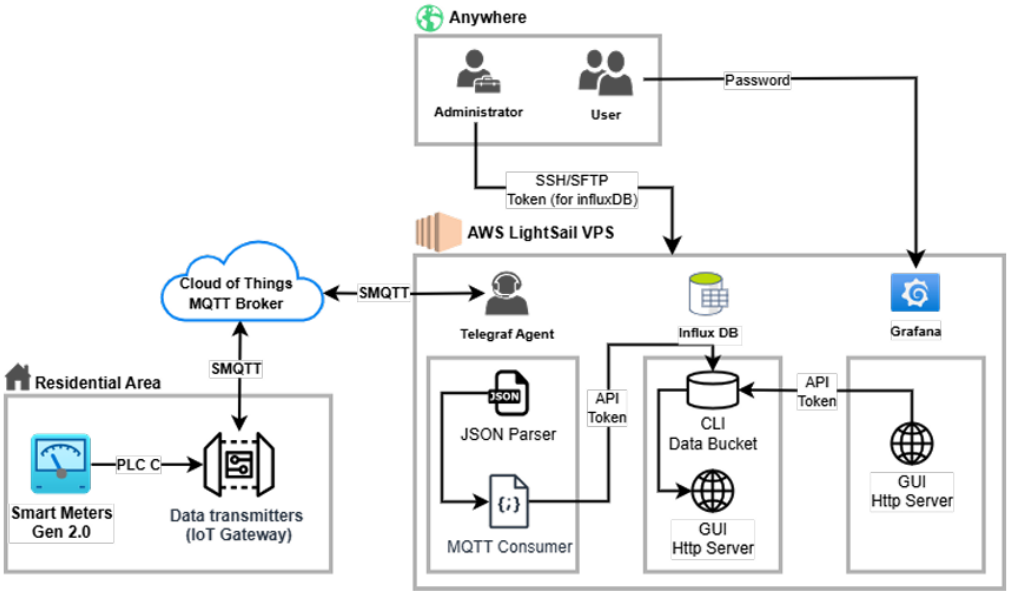


Figure 2.16 Cloud-based infrastructure and security framework [16]

The figure above depicts the connection between the smart meters (Edge), the Telegraf agent and the AWS/Cloud-hosted InfluxDB and Grafana services. It depicts the TIG stack utilised in the project and indicates the security layers (SSL/TLS) necessary for secure data transfer.

The efficiency of the data input pipeline is crucial for ensuring system real-time responsiveness. A pipeline using Telegraf as a data agent is substantially quicker than traditional Python-based ingestion scripts, processing 10,000 data records in around 0.39 seconds [17]. By optimising Telegraf flush intervals, the system can handle up to 100,000 records with no overhead, allowing for processing rates up to 90 times quicker than non-optimized systems. This scalability is critical for energy management systems that are designed to monitor several appliances or homes at once. Furthermore, the usage of Telegraf enables the smooth transition of JSON payloads into the InfluxDB Line Protocol and ensures that data is structured appropriately for quick updates and long-term preservation [17]. Figure 2.17 provides the data ingestion pipeline architecture.

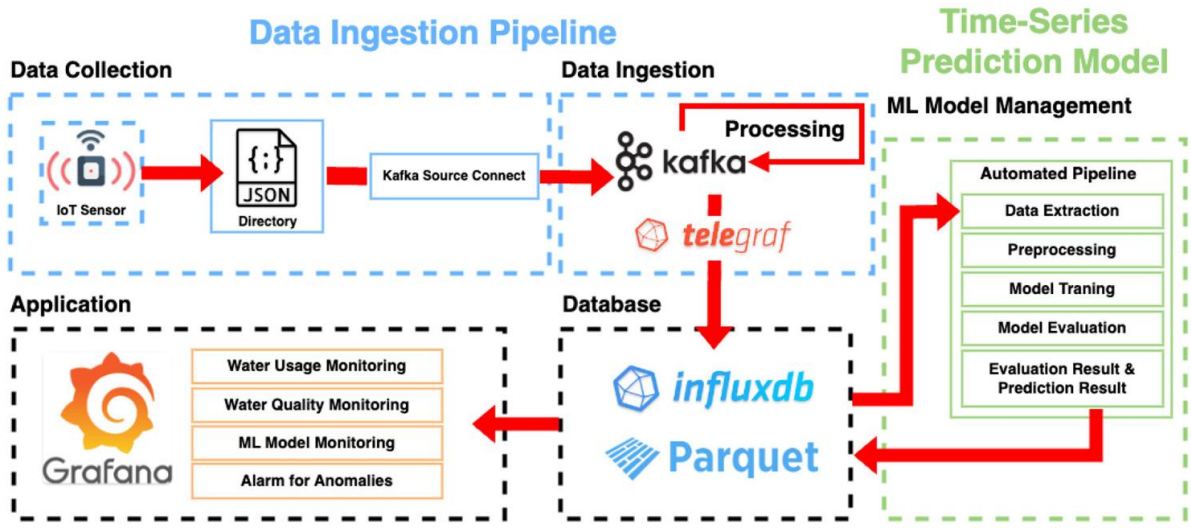


Figure 2.17 Data ingestion pipeline architecture [17]

The figure above depicts the whole path from IoT sensor to database. It shows how data flows via a processing layer (Telegraf) before being saved in InfluxDB. This diagram supports the usage of a modular pipeline for great scalability and reduced processing overhead.

For the end user, the raw data saved in InfluxDB must be transformed into actionable insights using a strong visualisation layer. Grafana has emerged as the most popular open-source platform for this function owing to its compliance with the Flux and InfluxQL scripting languages. The combination of InfluxDB with Grafana allows the building of dynamic dashboards containing live gauges, time-series graphs, and status indications [14]. Beyond basic visualisation, Grafana provides the critical image-rendering function utilised in this

project to produce PNG snapshots of dashboard panels for transmission via the Telegram bot. This functionality converts the monitoring system from a static web page to an interactive, mobile-first experience. Furthermore, the installation of Role-Based Access Control (RBAC) and TLS encryption across the InfluxDB-Grafana stack guarantees that sensitive home energy data stays secure and private [17]. Figure 2.18 displays the dynamic Grafana dashboard for energy monitoring.

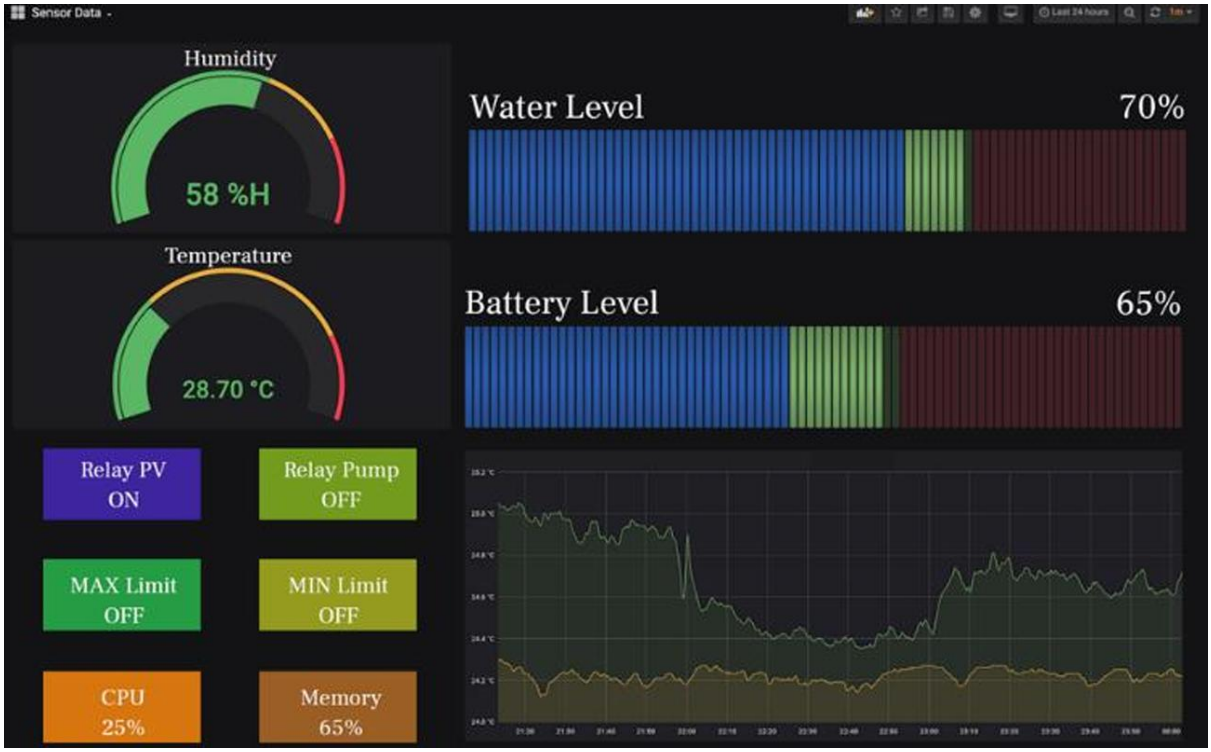


Figure 2.18 Dynamic Grafana dashboard for energy monitoring [14]

The figure above shows an operational energy monitoring dashboard. It demonstrates how several parameters, such as humidity, power levels and system status are shown using gauges and line charts. This serves as a baseline for the current project UI design and displays how Grafana transforms complicated data into human-readable information.

2.2.5 Machine Learning-based Anomaly Detection with Random Forest Regression

To detect abnormalities in smart home energy data, an algorithm that can handle non-linear, high-dimensional and cyclical time-series information is required. RF has emerged as the most effective supervised ensemble learning solution for this job. RF functions by creating a wide range of decision trees during the training phase and producing the class that reflects the mean or average forecast of the individual trees [18]. This ensemble technique is especially effective because it reduces the danger of overfitting, a typical failure in single decision trees in which the model learns the noise of the training data rather than the underlying patterns. In the context of energy monitoring, this allows the system to discern between a normal power surge and a metrological drift or equipment malfunction [18]. Figure 2.19 illustrates the RF ensemble structure.

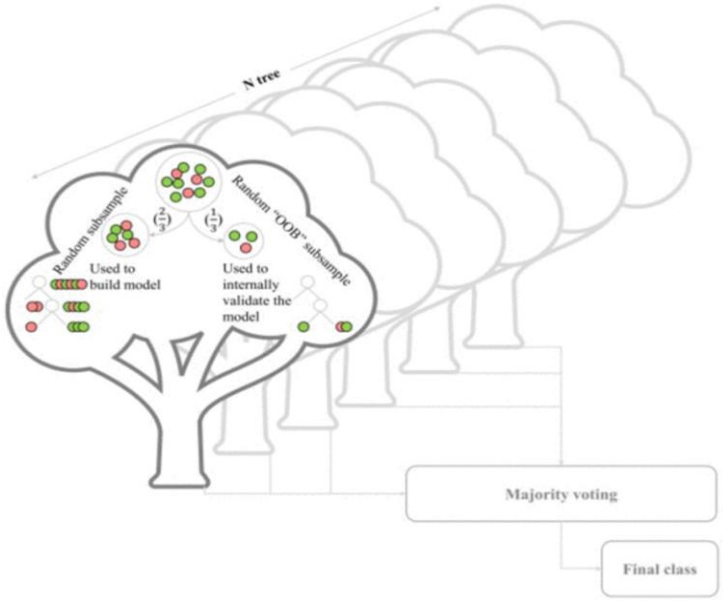


Figure 2.19 The Random Forest ensemble structure [18]

The figure above shows how the Random Forest model is made up of numerous independent decision trees (N trees). It demonstrates majority voting that explains how the system combines the findings of several trees to get a single, stable forecast. This supports the model resistance to data noise.

RF surpasses more computationally costly models like Artificial Neural Networks (ANN) in tabular data contexts due to its efficiency. A comparative investigation of several classifiers and discovered that RF achieved near-perfect accuracy (up to 100% in specified test

situations) while using much less processing time than deep learning models [19]. This processing efficiency is critical for IoT systems that require real-time inference to transmit instant notifications through platforms such as Telegram. Furthermore, the algorithm capability to prioritise feature significance is useful in temporal feature engineering. By determining which time-based variables most substantially impact energy patterns, the model may develop a context-aware dynamic threshold that adjusts to the user's individual behaviours [19]. Figure 2.20 provides the anomaly detection framework workflow.

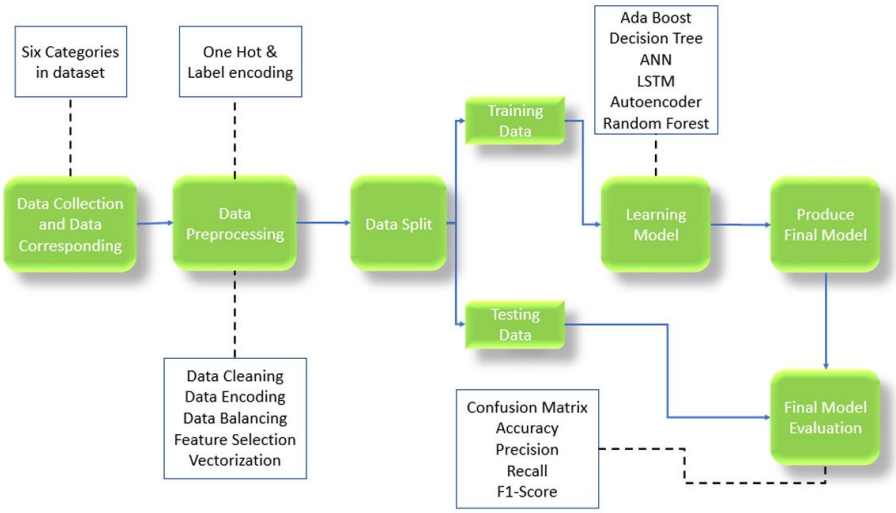


Figure 2.20 Anomaly detection framework workflow [19]

The figure above depicts the detection architecture in complete form, including data collection and preprocessing, as well as the learning model and final assessment.

In energy-specific datasets, anomalies are frequently manifested as junk data, outliers or missing values due by sensor failures. While single-classifier techniques such as Support Vector Machines (SVM) are successful, they are frequently insufficient for the range of abnormalities encountered in smart home consumption measurements [20]. The findings imply that RF is a suitable baseline since it effectively processes numerical and tabular data while retaining good accuracy and recall. The system may detect deviations from the expected baseline using standard deviation (σ) scaling and the regressor predictions. When real power usage is beyond the expected range, the model flags the occurrence as an anomaly, offering a preventive maintenance framework that extends the lifespan of household equipment and assures billing accuracy [20]. Figure 2.21 compares the performance of classifiers.

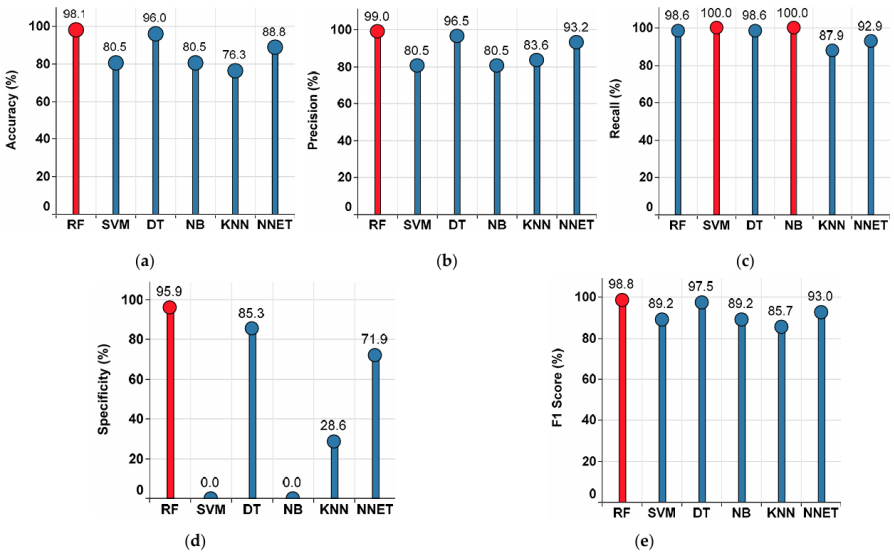


Figure 2.21 Performance comparison of classifiers [20]

The figure above shows the Accuracy, Precision, Recall and F1 Scores of several single-classifier models. It presents actual proof that RF outperforms its competitors in smart home energy datasets in terms of accuracy and specificity (the red bars), validating its selection as the project fundamental intelligence engine. Figure 2.22 further illustrates the anomaly detection and data cleaning pipeline.

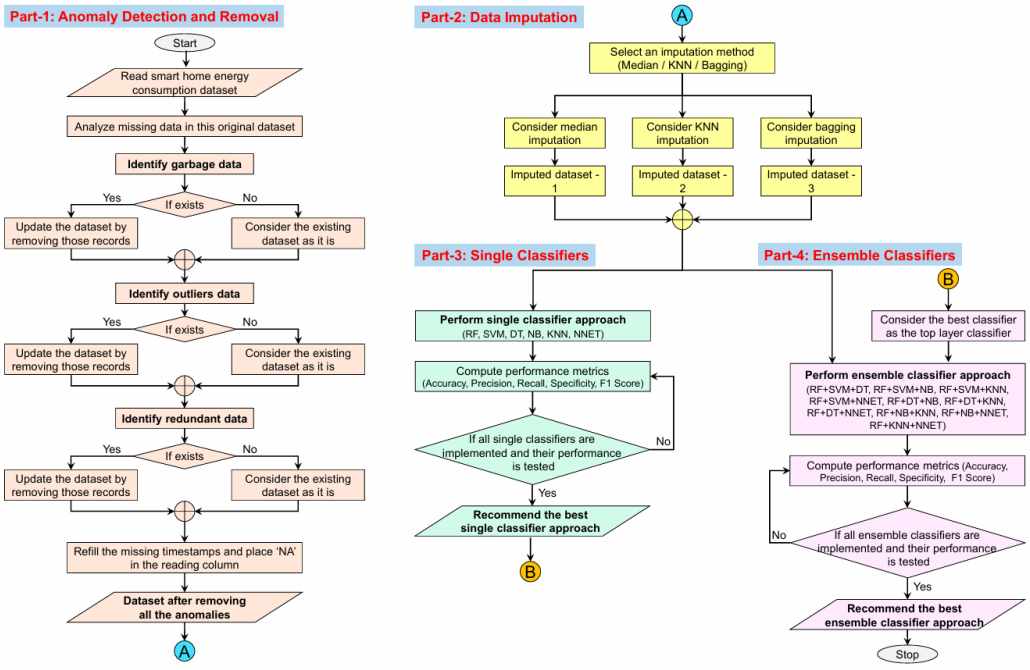


Figure 2.22 Anomaly detection and data cleaning pipeline [20]

As illustrated in the figure above, the implementation of RF for anomaly detection that is preceded by a strict multi-stage preprocessing pipeline. This method assures that the model is not trained on junk data or duplicate records which are typical in IoT sensor streams. The method maintains excellent data integrity by rigorously removing outliers and identifying missing intervals prior to the inference step and allowing the RF algorithm to learn true behavioural patterns rather than sensor-induced artefacts. The stacking ensemble architecture is shown in Figure 2.23.

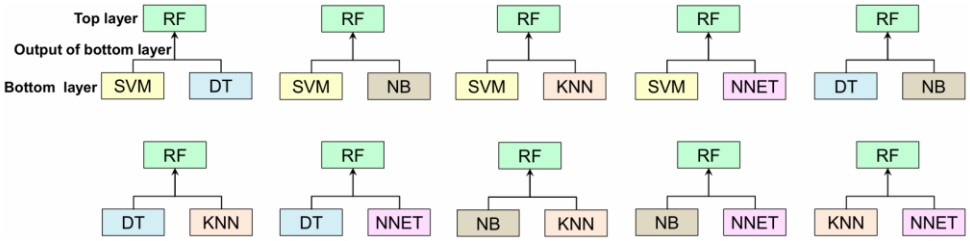


Figure 2.23 Stacking ensemble architecture [20]

To understand the RF model greater generalisation, it is necessary to analyse the stacking architecture utilised in ensemble learning. The figure above depicts how a top layer classifier combines the outputs of many bottom layer estimators. In the context of this research, this design avoids the system from being duped by a single noisy data point; instead, it depends on the consensus of many decision routes to confirm an abnormality. This stacking method is what enables the system to achieve high specificity, making sure that an alarm is only provided to the user when a substantial divergence is detected throughout the ensemble. Figure 2.24 highlights the accuracy and house variability.

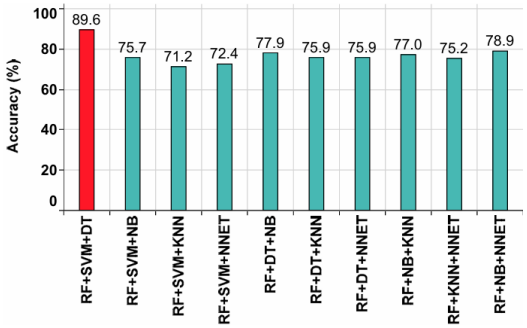


Figure 2.24 Accuracy and house variability [20]

Empirical evidence in the figure above reveals that, while RF is the most accurate model for home energy monitoring, its performance varies depending on individual household data features. As shown in Figure 2.24, accuracy remains consistently high (above 85%) across five different test sites, but the minor fluctuations highlight the importance of the "retrain_model_auto()" function implemented in this project, which allows the AI to adapt to the unique electrical footprint of each user.

2.3 Concluding Remark

In conclusion, this chapter presented an extensive review of existing energy monitoring frameworks as well as the basic technologies needed to create this system. The assessment discovered a large gap in current solutions that frequently rely on passive data logging or closed-source systems with limited bidirectional user engagement. The technological basis for the project design is built by analysing the strengths of the ESP32 microcontroller, high-precision 16-bit signal capture, and the efficiency of the MQTT-InfluxDB pipeline. Furthermore, the addition of Random Forest Regression (RFR) for context-aware anomaly identification meets the requirement for a more intelligent and proactive monitoring solution.

Chapter 3

System Methodology/Approach

This chapter describes the technical requirements, multi-layered architecture and operational logic that guide the system development using a structured Agile methodology.

3.1 System Requirement

3.1.1 Hardware

The hardware used in this project is the components that measure electricity. It is designed to be safe and accurate with standard and reliable components that are simple to discover and connect. These components are carefully selected to ensure their affordable cost and effectiveness. The details of the components are further described below.

ESP32 Microcontroller

The 30-pin, type-c version of the ESP32 was chosen as the edge device core. The ESP32 is an excellent choice for IoT applications due to its dual-core processor, numerous GPIO and integrated Wi-Fi. This provides enough computing capacity for real-time calculations while also providing stable network connectivity out of the box.

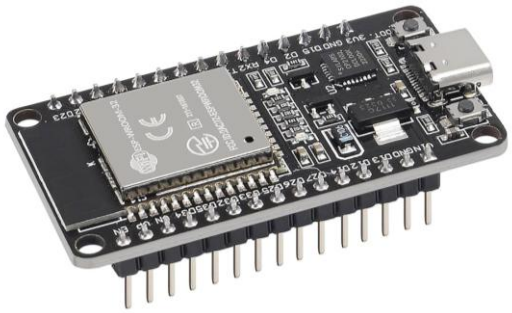


Figure 3.1 ESP32 Microcontroller

Table 3.2 Specifications of ESP32 Microcontroller

Description	Specifications
Model	ESP32 (30-pin) with ESP-WROOM-32 Module
Processor	Dual-core 32-bit Tensilica LX6
Wireless Connectivity	2.4 GHz Wi-Fi and Bluetooth
Flash Memory	4 MB
SRAM	520 KB
Operating Voltage	2.2V to 3.6V

SCT-013-000 Current Transformer (CT) Sensor

The SCT-013-000 CT is the primary sensor for measuring household power use. Its main feature is its non-invasive split-core design. This means it can be securely clamped around the main power cord without the need to cut any wires. This making installation both straightforward and safe.

Figure 3.2 SCT-013-000 Current Transformer

Table 3.3 Specifications of SCT-013-000 CT Sensor

Description	Specifications
Model	SCT-013-000
Rated Input Current	100A
Output Signal	1V AC at rated current
Type	Non-invasive, Split-Core
Connector	3.5mm Audio Jack

ADS1115 Analog-to-Digital Converter (ADC)

The ADS1115 Analog-to-Digital Converter functions as a high-precision translator between the analog CT sensor and the digital ESP32 microcontroller. The CT sensor generates a smooth, analog waveform, but the ESP32 can only process digital data. The ADS1115 is chosen precisely because of its high resolution that enables it to translate the sensor signal into a more detailed and accurate digital value.

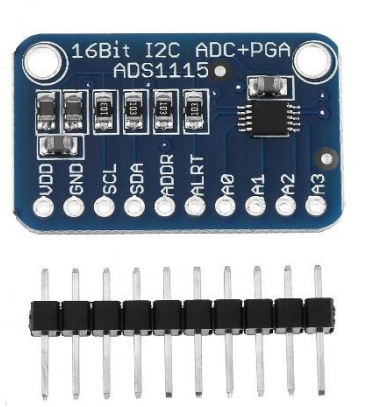


Figure 3.3 ADS1115 Analog-to-Digital Converter

Table 3.4 Specifications of ADS1115 Analog-to-Digital Converter

Description	Specifications
Model	SCT-013-000
Rated Input Current	100A
Output Signal	1V AC at rated current
Type	Non-invasive, Split-Core
Connector	3.5mm Audio Jack

Chi Jing Micro Controller Unit-Tip-Ring-Ring-Sleeve (CJMCU-TRRS) 3.5mm Jack Breakout Board

The CJMCU-TRRS 3.5mm jack breakout board is a fundamental but necessary component that connects the CT sensor to the main circuit. The sensor includes a standard 3.5mm audio socket and this breakout board allows it to be inserted straight into the breadboard. This assures a stable and dependable connection without the need of soldering.

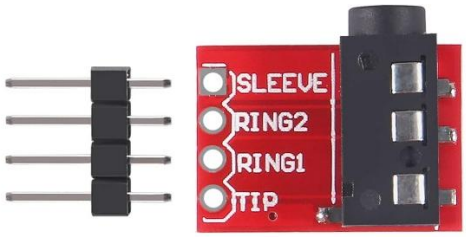


Figure 3.4 CJMCU-TRRS 3.5mm jack breakout board

Table 3.5 Specifications of 3.5mm Jack Breakout Board

Description	Specifications
Model	CJMCU-TRRS
Jack Diameter	3.5mm
Jack Type	4-pin Stereo (TRRS)
DIP Pin Spacing	2.54 mm (Breadboard compatible)

Direct Current (DC) Bias Circuit Components

Resistors and capacitor connect to form the DC bias circuit. The two 10kΩ resistors provide a stable reference voltage, while the 10µF capacitor smooths it out. This circuit's function is to add a small positive DC voltage to the AC signal from the CT sensor. This is an important step because it converts the whole AC waveform into a range that the ADC can read and guarantees that no electrical signal is lost and that the final measurements are accurate.

Figure 3.5 Resistor

Table 3.6 Specifications of Resistor

Description	Specifications
Resistance	10 kΩ
Power Rating	0.25W
Tolerance	1%

Figure 3.6 Capacitor

Table 3.7 Specifications of Capacitor

Description	Specifications
Capacitance	10 µF
Max Voltage	50V
Type	Electrolytic

Tactile Push Button

A basic tactile push button was added to the circuit for testing purposes. Its purpose is to provide a manual way for simulating a sudden power spike. This is very helpful for quickly and regularly validating the real-time alerting feature to guarantee that notifications are sent accurately and promptly.

Figure 3.7 Tactile push button

Table 3.8 Specifications of Tactile Push Button

Description	Specifications
Type	Momentary
Pins	4
Use	Manual anomaly simulation

3.1.2 Software

The software utilised in this project consists of programs and platforms for processing, storing and visualising acquired energy data. It is built to be durable and responsive, using standard and reliable open-source technologies that are easy to integrate and deploy. These software components are carefully chosen to achieve optimal performance, scalability and cost-effectiveness. The details of the software components are described below.

Arduino IDE

The Arduino Integrated Development Environment (IDE) is a cross-platform application that offers a comprehensive environment for code creation, compilation, and uploading. In this project, it is the major tool for creating C++ firmware for the ESP32 microcontroller. The IDE built-in board manager and library manager make it easier to set up the development environment and provide useful libraries for Wi-Fi, MQTT, and sensor connectivity.

Python

Python is a high-level, versatile programming language distinguished by its ease of use and extensive libraries. For this project, Python is the primary language for backend application logic. It is used to write the script that runs the Telegram bot. It will listen to user commands, make Application Programming Interface (API) calls to the Grafana server and return messages and photos.

Docker and Docker Compose

Docker is a platform for developing, distributing and running applications within containers. Docker Compose is a tool for creating and managing multi-container Docker applications. This technology serves as the foundation for the complete backend architecture and managing several services including Mosquitto, Telegraf, InfluxDB and Grafana. Docker makes the backend portable, scalable and quick to deploy on any system with a single command.

Mosquitto MQTT Broker

Mosquitto is an open-source message broker that uses the MQTT protocol. In this project, it serves as the main communication hub for all IoT data. It takes energy consumption data from ESP32 and sends it efficiently to any subscribing customers, primarily the Telegraf agent.

Telegraf

Telegraf is a plug-in server agent to collect and report metrics. It functions as an essential data pipeline in this system. It is set up to subscribe to the Mosquitto MQTT broker, gather incoming energy data payloads and automatically send them to the InfluxDB database in the appropriate format for long-term storage.

InfluxDB

InfluxDB is a high-performance database designed to store and query time-series data. It is the primary data storage solution for this project. It is designed to manage the huge volume of timestamped data points provided by the energy monitor. This result in highly quick data ingestion and efficient execution of time-based queries for analysis and visualization.

Grafana

Grafana is an open-source analytics and visualization platform. It serves as the primary tool for designing the dashboard. Grafana uses the InfluxDB database as a data source and allow to create rich, interactive dashboards that display live and historical energy statistics using graphs, gauges and statistical panels.

3.2 System Design Diagram

3.2.1 System Architecture Diagram

The architecture is divided into three distinct operational domains, home network, cloud network and end user interface. This separation of responsibilities results in a modular, scalable and robust system. Figure 3.8 shows the complete system architecture.

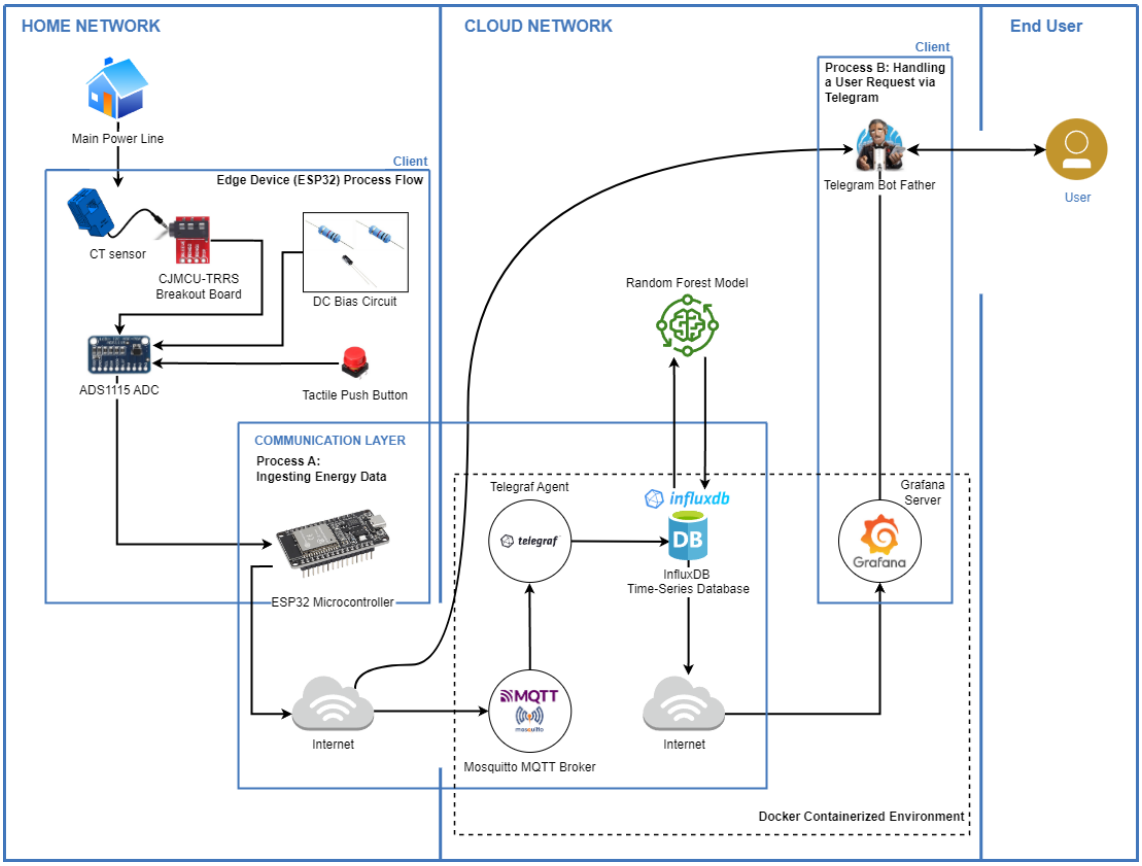


Figure 3.8 System architecture diagram of the project

The procedure begins at the Home Network which is the system edge. The Edge Device (ESP32) Process Flow in the figure above describes the operating logic for this area. As shown, the CT sensor detects current from the Main Power Line in a non-invasive way. The analog signal is routed through the CJMCU-TRRS Breakout Board and filtered by the DC Bias Circuit before being digitized by the ADS1115 ADC. The digital signal is subsequently processed by the ESP32 microcontroller that serves as an intelligent client. The ESP32 handles all real-time calculations and connects to the internet via the home Wi-Fi network.

From the Home Network, ESP32 sends its data payload to the Cloud Network. The connection is supported by the Communication Layer that based on the Mosquitto MQTT Broker. The route of this data is described in Process A: Ingesting Energy Data in the figure above. This flowchart illustrates how the broker serves as the system's main message route. The Telegraf Agent subscribes to the broker, collects incoming energy data and consistently uploads it to the InfluxDB TSDB for long-term storage. This database is the only source of truth for all past data. The Mosquitto broker, Telegraf agent, InfluxDB database and Grafana server are all deployed within a Docker Containerized Environment to provide system portability and manageability. This isolation enables the services to communicate effectively over a shared virtual network by utilising their container names.

The incorporation of the RF Model into the cloud architecture to offer intelligent anomaly detection is a major improvement in this stage. This model uses InfluxDB in a bidirectional data loop. It uses Temporal Feature Engineering to extract historical time-series data and identify cyclical household energy patterns. The model performs real-time inference to incoming data after it has been trained. The RF Model sends a direct message to the Telegram Bot Father to initiate proactive alerts if the actual power usage differs significantly from the expected baseline which determined by AI.

The Cloud Network covers the final domain, the End User interface. Grafana Server uses the InfluxDB database to generate and display interactive dashboards. The End User has two primary options for accessing this data. First, they can examine the Grafana dashboard in a web browser. Second and more interactively, they can communicate with the Telegram Bot Father that represents the Telegram API. The entire logic behind this interaction is explained in Process B: Handling a User Request via Telegram in the figure above. The bot also serves as a client for the Grafana server, can retrieve and submit dashboard snapshots and receive real-time notifications from the ESP32 and the RF Model. The lightweight ESP32 handles the sensing, a reliable backend handles the data and a flexible interface gives the user actionable information. This multi-layered architecture guarantees system efficiency.

3.2.2 Use Case Description

Table 3.9 Use Case Description of monitor real-time energy usage and persistence

Use Case ID	UC001	
Use Case	Monitor Real-Time Energy Usage and Persistence	
Purpose	To provide the user with high-fidelity visibility of current electricity consumption and ensure data is not lost during power interruptions.	
Actor	User	
Trigger	User launches the Grafana web dashboard.	
Precondition	ESP32 is powered on and connected to the local Wi-Fi network.	
Scenario Name	**Step**	**Action**
Main Flow	1	The system captures raw AC current signals via the CT sensor.
	2	The ESP32 processes the signal to calculate Root Mean Square (RMS) current, power, and tiered costs.
	3	The system updates the “Energy Odometer” (totalKWh) and saves it to NVS.
	4	The system transmits the processed data to the dashboard via MQTT.
	5	The user views the live metrics and cost estimations.
Alternate Flow - Device Connectivity Failure	4.1	If the network is unavailable, the ESP32 continues calculating locally and attempts to reconnect without data loss.
Rules	-	The system must write the totalKWh value to NVS every 15 seconds to manage flash memory endurance while maintaining data persistence.

Table 3.10 Use Case Description of request on-demand visual energy reports

Use Case ID	UC002	
Use Case	Request On-Demand Visual Energy Reports	
Purpose	To allow the user to receive graphical snapshots of energy trends directly within a messaging interface.	
Actor	User	
Trigger	User selects a predefined time-range button on the Telegram Bot.	
Precondition	Dockerized backend (Grafana and Image-Renderer) is operational.	
Scenario Name	**Step**	**Action**
Main Flow	1	The user sends a report command via the Telegram custom keyboard.
	2	The Python Bot Script receives the request and validates the command.
	3	The system triggers an API call to the Grafana Image-Renderer.
	4	The system converts the dynamic dashboard panel into a PNG image.
	5	The Telegram Bot sends the PNG image with a summary caption to the user.
Alternate Flow - Empty Dataset Rendering Fallback	4.1	If InfluxDB returns a null set for the requested period, the system applies a “Fallback Table” to render a “No Data” alert image instead of crashing.
Rules	-	All image rendering requests must be authenticated via a Bearer Token for security.

Table 3.11 Use Case Description of proactive anomaly detection and alerting

Use Case ID	UC003	
Use Case	Proactive Anomaly Detection and Alerting	
Purpose	To automatically identify and notify the user of abnormal power consumption patterns using Machine Learning.	
Actor	System (AI Inference Engine), User (Receiver)	
Trigger	The arrival of a new data payload at the MQTT Broker.	
Precondition	The Random Forest model has been trained on at least 7 days of historical behavioral data.	
Scenario Name	**Step**	**Action**
Main Flow	1	The AI Engine intercepts the live data stream from the MQTT broker.
	2	The system extracts temporal features (hour and day) from the data.
	3	The Random Forest model predicts the Expected Baseline power for that specific time context.
	4	The system compares the actual power against the predicted baseline using σ (standard deviation) scaling.
	5	An anomaly is detected, and the system pushes a “High Power Alert!” message to the user’s Telegram.
Alternate Flow - Alert Cooldown Suppression	5.1	If the system is within the “Alert Cooldown” period (15s), the notification is suppressed to prevent spamming.
Rules	-	The detection threshold must be dynamic and based on the user’s learned habits, not a static limit.

Table 3.12 Use Case Description of conduct smart energy and carbon audit

Use Case ID	UC004	
Use Case	Conduct Smart Energy and Carbon Audit	
Purpose	To provide a comprehensive sustainability report that translates energy usage into environmental impact metrics.	
Actor	User	
Trigger	User selects the “Smart Audit” option and provides a date range.	
Precondition	The user has interacted with the Interactive Inline Calendar UI.	
Scenario Name	**Step**	**Action**
Main Flow	1	The user selects the start and end dates via the Telegram calendar interface.
	2	The Python script aggregates the total kWh consumption from InfluxDB for that range.
	3	The system calculates the Carbon Footprint and the “Tree Offset” equivalent.
	4	The system generates dynamic expert recommendations based on the usage level.
	5	The user receives the full text-based audit report.
Alternate Flow - Invalid Chronological Date Selection	1.1	If the start date is after the end date, the bot notifies the user of the logical error and resets the calendar.
Rules	-	Calculations must be based on the official 2025 Malaysian Grid Emission Factor (0.674 kg CO2/kWh).

Table 3.13 Use Case Description of access wireless headless diagnostics

Use Case ID	UC005	
Use Case	Access Wireless Headless Diagnostics	
Purpose	To allow the administrator to monitor raw hardware performance and debug the system without physical cable connections.	
Actor	System Administrator	
Trigger	Administrator navigates to the ESP32 local URL or mDNS address in a web browser.	
Precondition	The diagnostic device is on the same Local Area Network (LAN) as the ESP32.	
Scenario Name	**Step**	**Action**
Main Flow	1	The Admin accesses http://EnergyMonitor-ESP32.local/webserial.
	2	The ESP32 internal web server accepts the connection request.
	3	The system establishes a WebSocket connection between the browser and the ESP32.
	4	The Admin views the live “WebSerial Lite” stream of raw ADC samples and connection logs.
Alternate Flow - mDNS Resolution Failure	1.1	If mDNS resolution fails, the Admin uses the static DHCP-reserved IP address.
Rules	-	The WebSerial interface is read-only for security; no control commands can be sent via this interface.

Table 3.14 Use Case Description of query time-series database

Use Case ID	UC006	
Use Case	Query Time-Series Database	
Purpose	To retrieve historical or real-time energy data from InfluxDB using Flux queries.	
Actor	System (InfluxDB)	
Trigger	Internal call from UC-02 (Snapshot), UC-03 (Anomaly), or UC-04 (Audit).	
Precondition	InfluxDB container is active and the API Authentication Token is valid.	
Scenario Name	**Step**	**Action**
Main Flow	1	The system receives a Flux query command specifying the time range and measurement.
	2	The system searches the data bucket for matching timestamps.
	3	The database returns a formatted table of power values.
Alternate Flow – No data exists	2.1	If no data exists for the selected range, the system returns a “Null Set” error.
Rules	-	Queries must be optimized to return results in under 2 seconds to prevent interface lag.

Table 3.15 Use Case Description of render dashboard image

Use Case ID	UC007	
Use Case	Render Dashboard Image	
Purpose	To convert a dynamic Grafana panel into a static high-resolution PNG image.	
Actor	System (Grafana Image-Renderer)	
Trigger	Internal call from UC-02 (Request Visual Reports).	
Precondition	The Grafana server is reachable and the specific Panel ID is correctly defined.	
Scenario Name	**Step**	**Action**
Main Flow	1	The system initiates a request to the /render endpoint with width and height parameters.
	2	The headless browser engine loads the CSS and visual components.
	3	The system captures the frame and encodes it into a PNG file.
	4	The system returns the file path to the calling script.
Alternate Flow – Time out	2.1	If the renderer times out (e.g., due to complex queries), the system triggers a “Render Failure” message.
Rules	-	The rendered image must match the exact timestamp of the user’s request.

Table 3.16 Use Case Description of generate expert recommendations

Use Case ID	UC008	
Use Case	Generate Expert Recommendations	
Purpose	To provide the user with context-specific energy-saving tips based on analyzed usage patterns.	
Actor	System (AI Engineer)	
Trigger	User initiates UC-04 (Audit) and the energy total exceeds baseline norms.	
Precondition	The statistical analysis of the user's requested date range is completed.	
Scenario Name	**Step**	**Action**
Main Flow	1	The system compares the average daily kWh against pre-defined "Efficiency Benchmarks."
	2	The logic engine identifies the most likely area of waste (e.g., standby power or high daytime load).
	3	The system selects a corresponding "Expert Recommendation" from the library.
	4	The tip is appended to the final Telegram audit report.
Alternate Flow – Positive efficiency	1.1	If the usage is already within "Excellent Efficiency" parameters, the system appends a "Keep it up!" message instead of a warning.
Rules	-	Recommendations must be based on current Malaysian energy efficiency standards, National Energy Efficiency Action Plan (NEEAP).

3.2.3 Use Case Diagram

Figure 3.9 shows the use case diagram of the system.

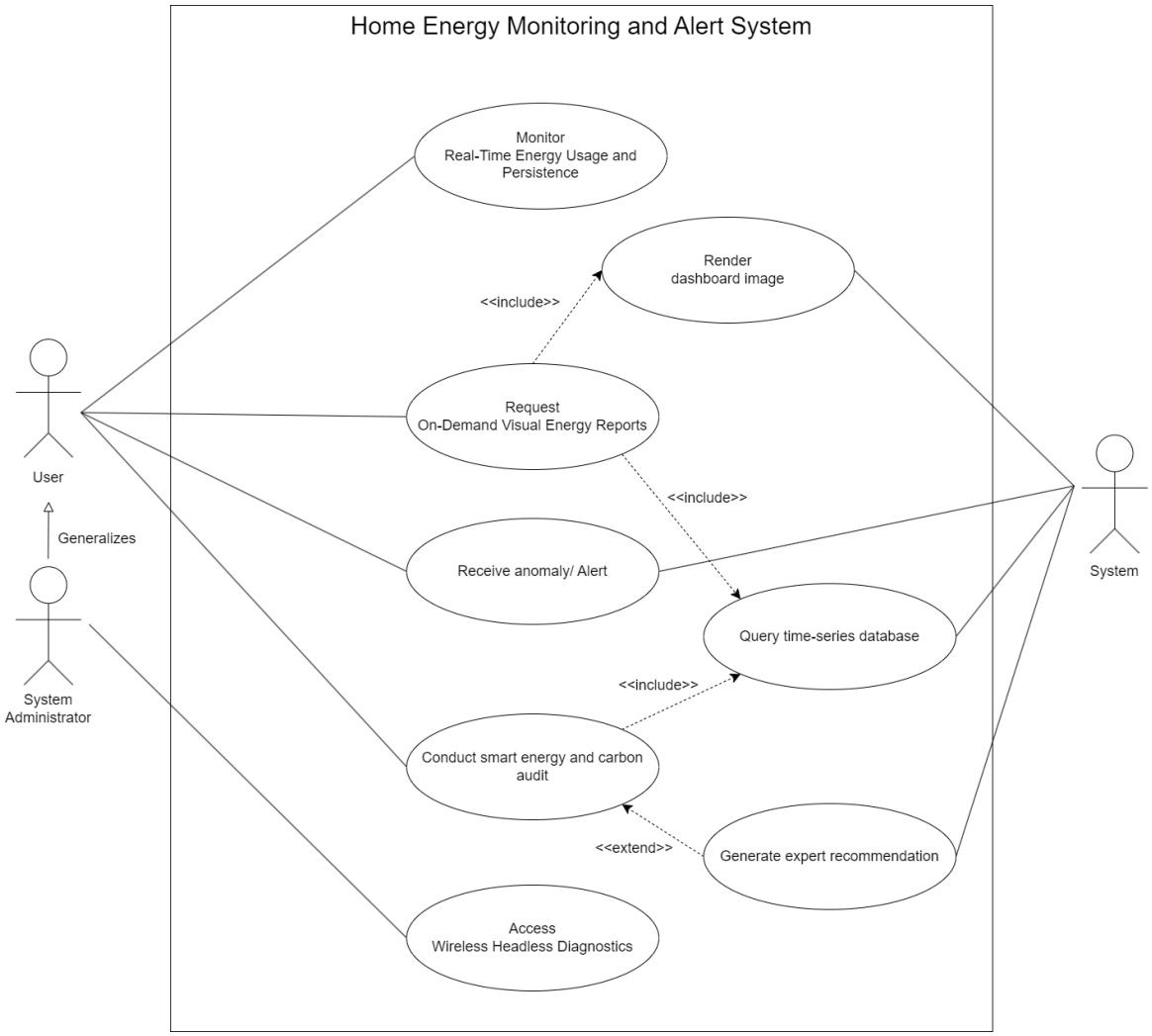


Figure 3.9 Use case diagram of the project

The Smart Home Energy Monitoring System functional interactions are shown in the figure above. The User represents the homeowner and is in charge of daily duties including monitoring live usage, requesting visual reports, and conducting carbon audits. The System actor represents the automated backend and AI engine that works in the background to query the database, render dashboard images, and send alerts. The System Administrator inherits all the ordinary user's permissions but also has access to wireless diagnostics for technical maintenance. The "<<extend>>" relationship shows that the system may intelligently offer optional expert recommendations to assist the user save energy, while the "<<include>>" relationships illustrate that requesting a report or an audit automatically initiates database and rendering operations.

3.2.4 Activity Diagram

Figure 3.10 shows the activity diagram for monitoring real-time energy usage and persistence.

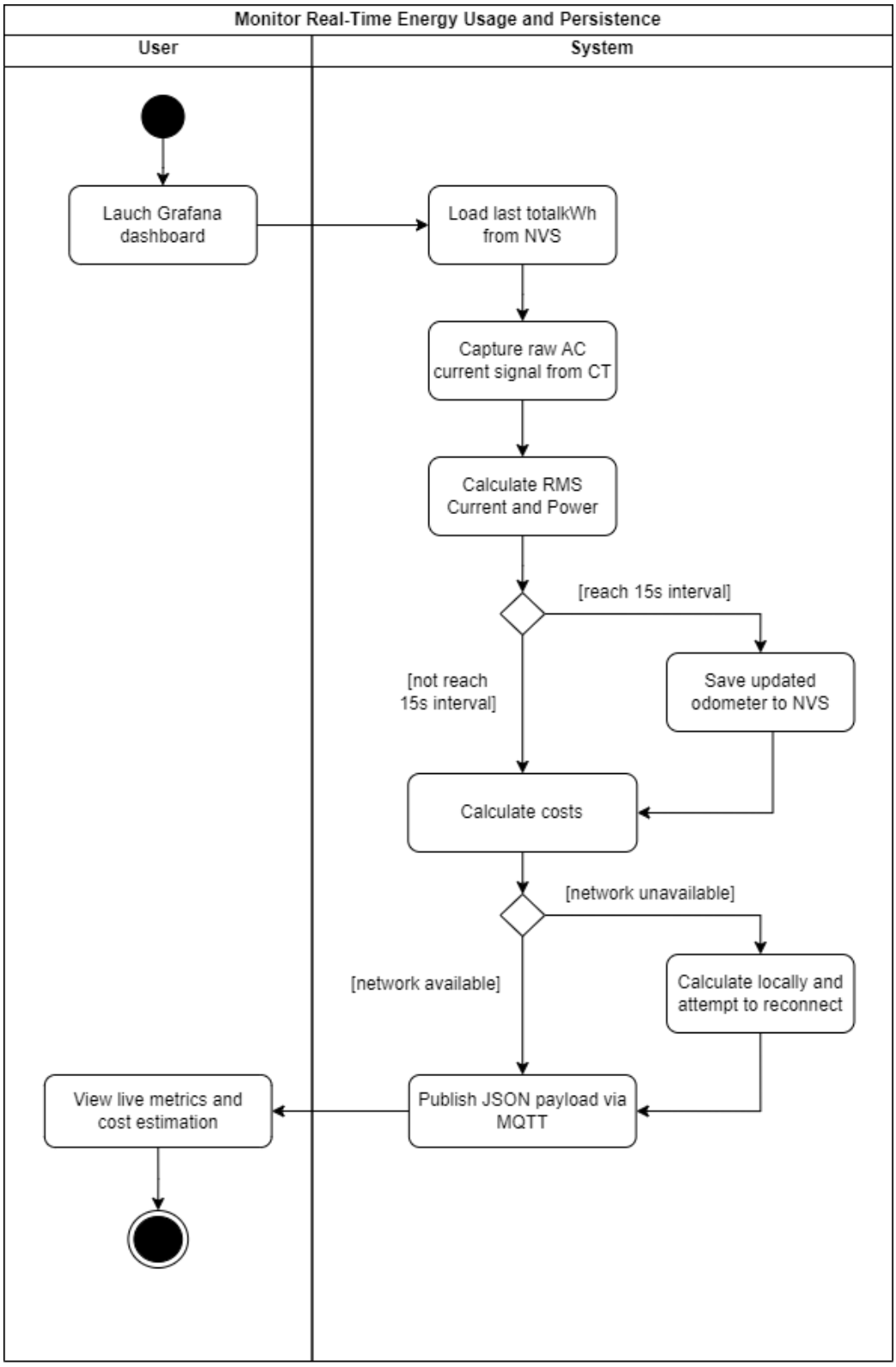


Figure 3.10 Activity diagram for monitor real-time energy usage and persistence

Figure 3.11 shows the activity diagram for request on-demand visual energy reports.

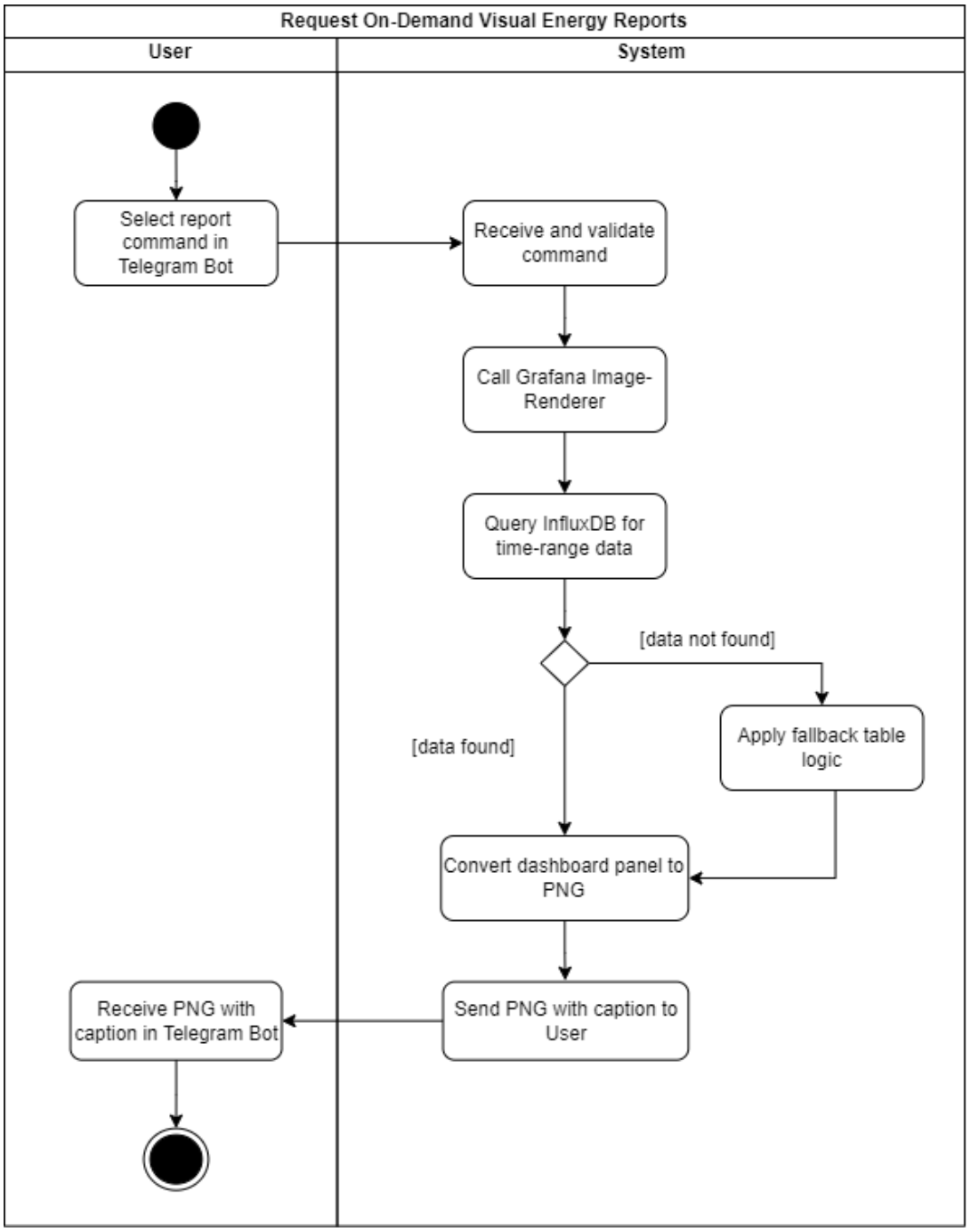


Figure 3.11 Activity diagram for request on-demand visual energy reports

Figure 3.12 shows the activity diagram for proactive anomaly detection and alerting.

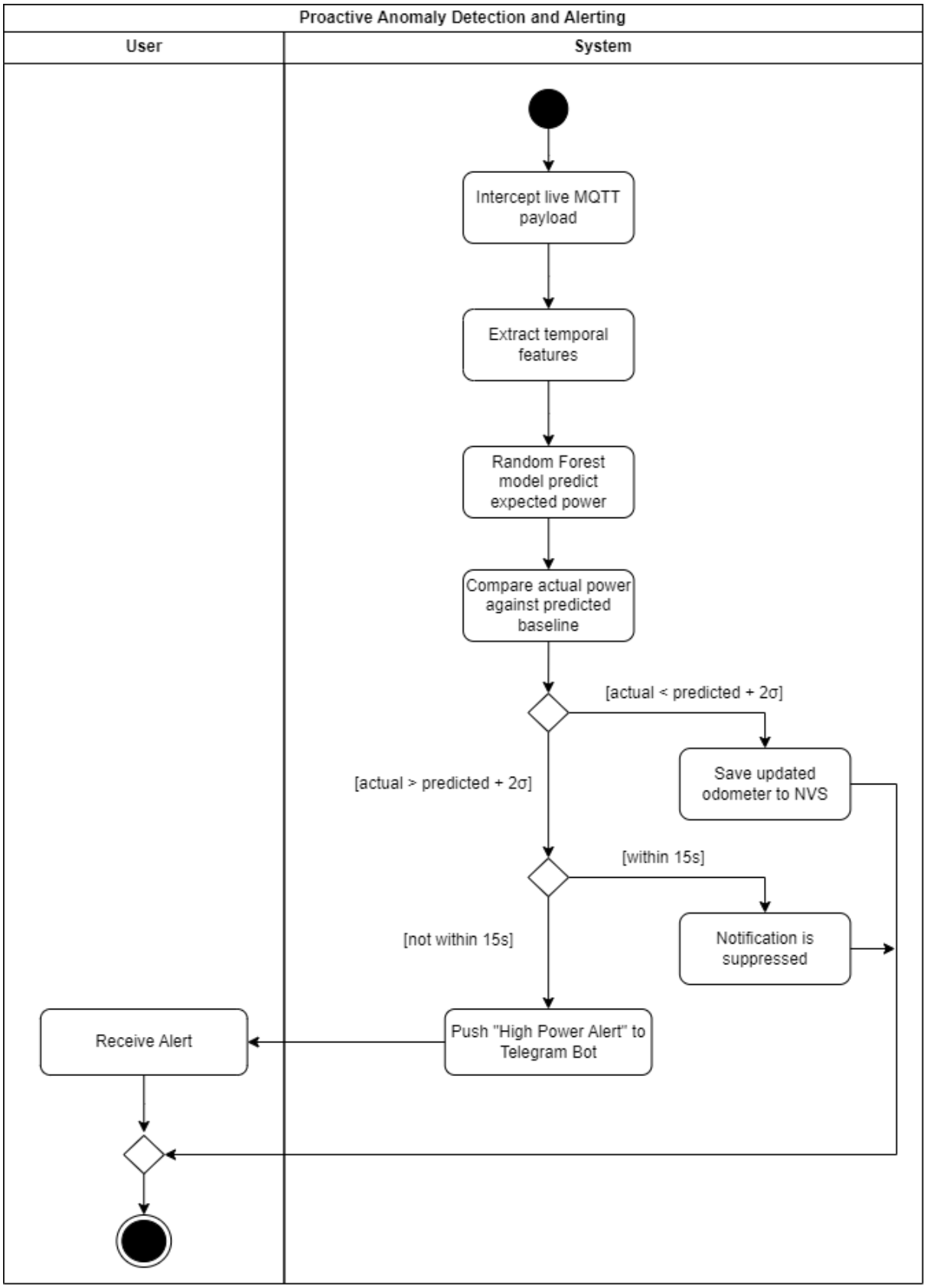


Figure 3.12 Activity diagram for proactive anomaly detection and alerting

Figure 3.13 shows the activity diagram for conducting smart energy and carbon audit.

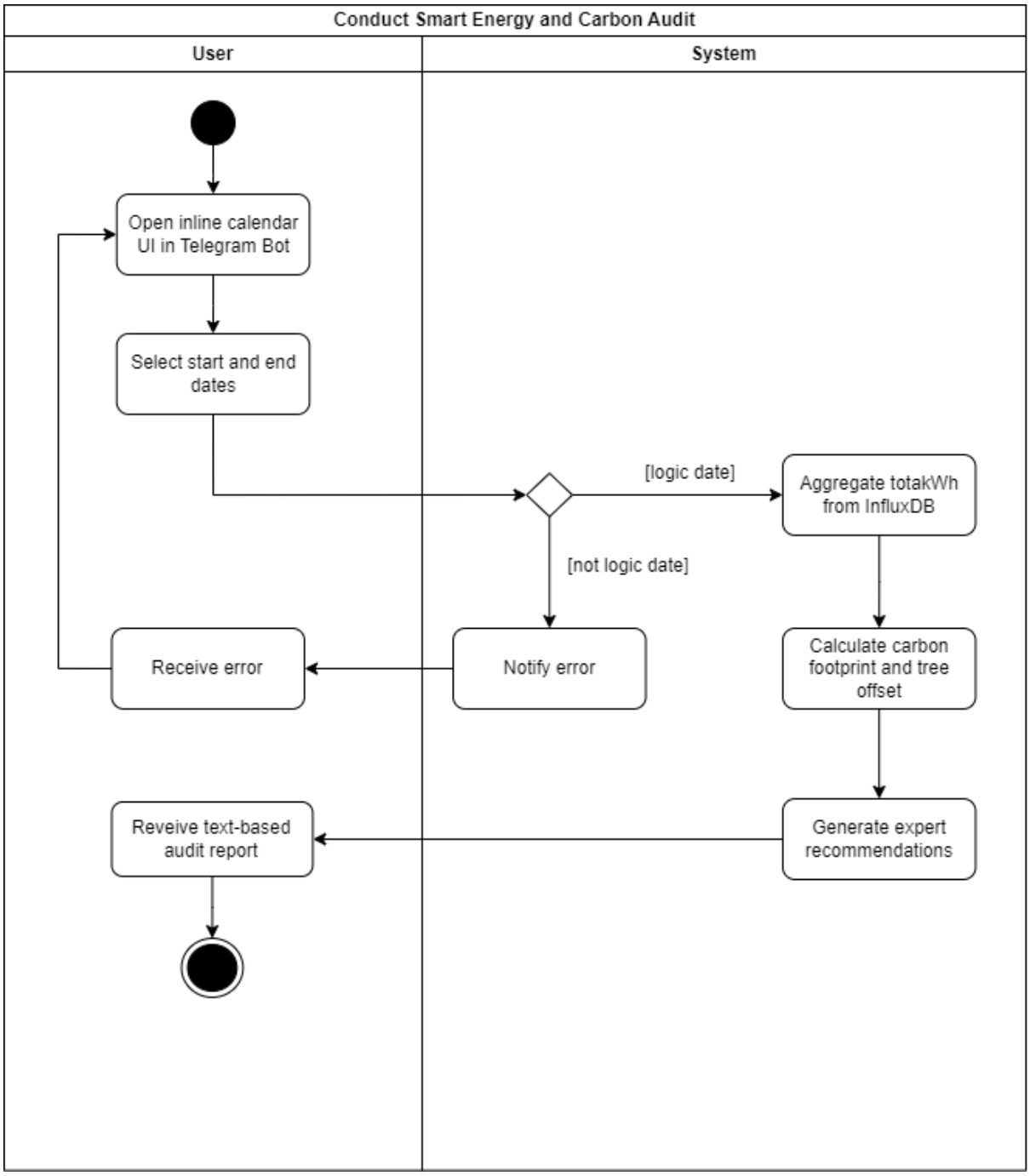


Figure 3.13 Activity diagram for conduct smart energy and carbon audit

Figure 3.14 shows the activity diagram for access wireless headless diagnostics.

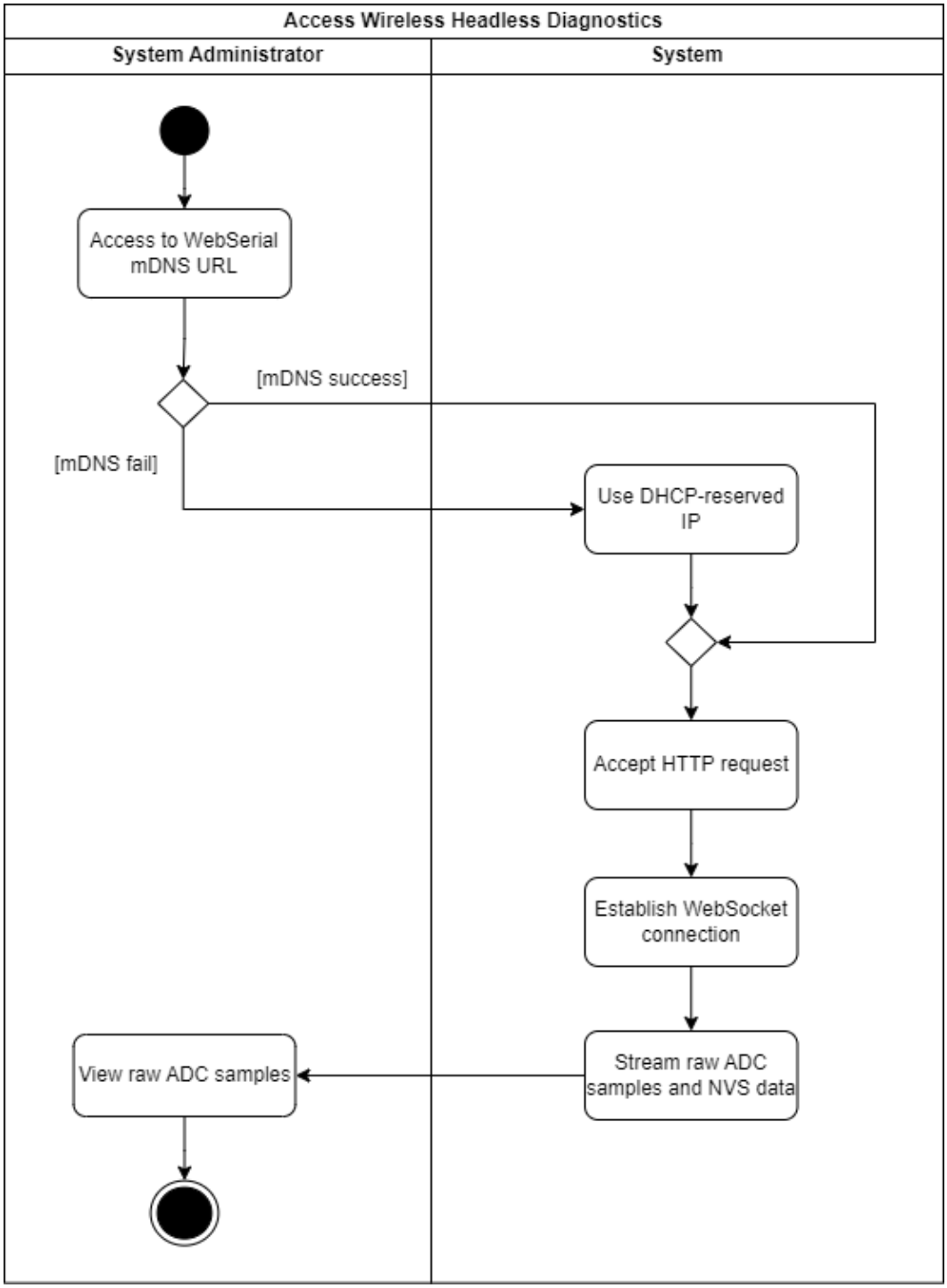


Figure 3.14 Activity diagram for access wireless headless diagnostics

Figure 3.15 shows the activity diagram for query time-series database.

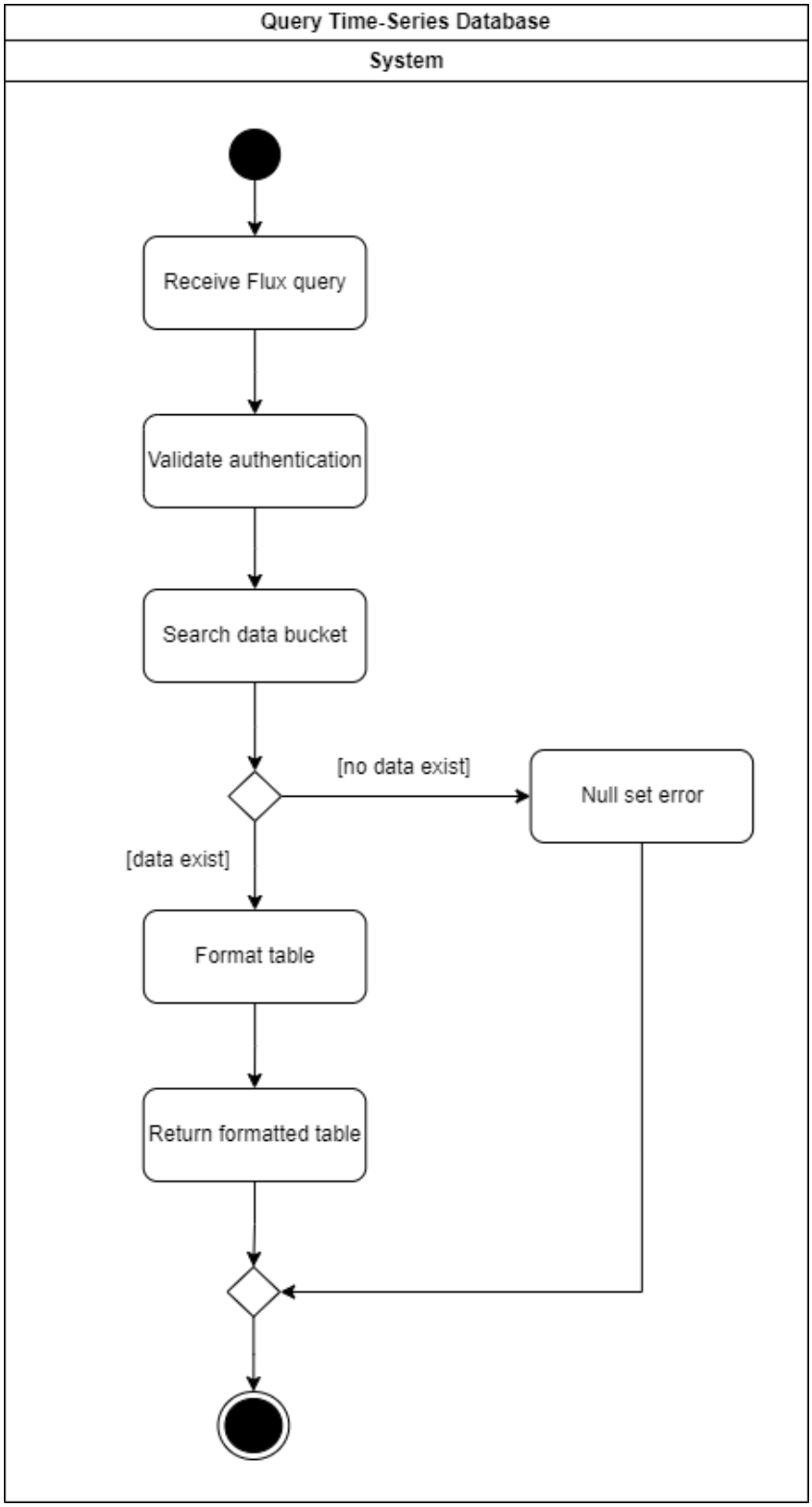


Figure 3.15 Activity diagram for query time-series database

Figure 3.16 shows the activity diagram for rendering dashboard image.

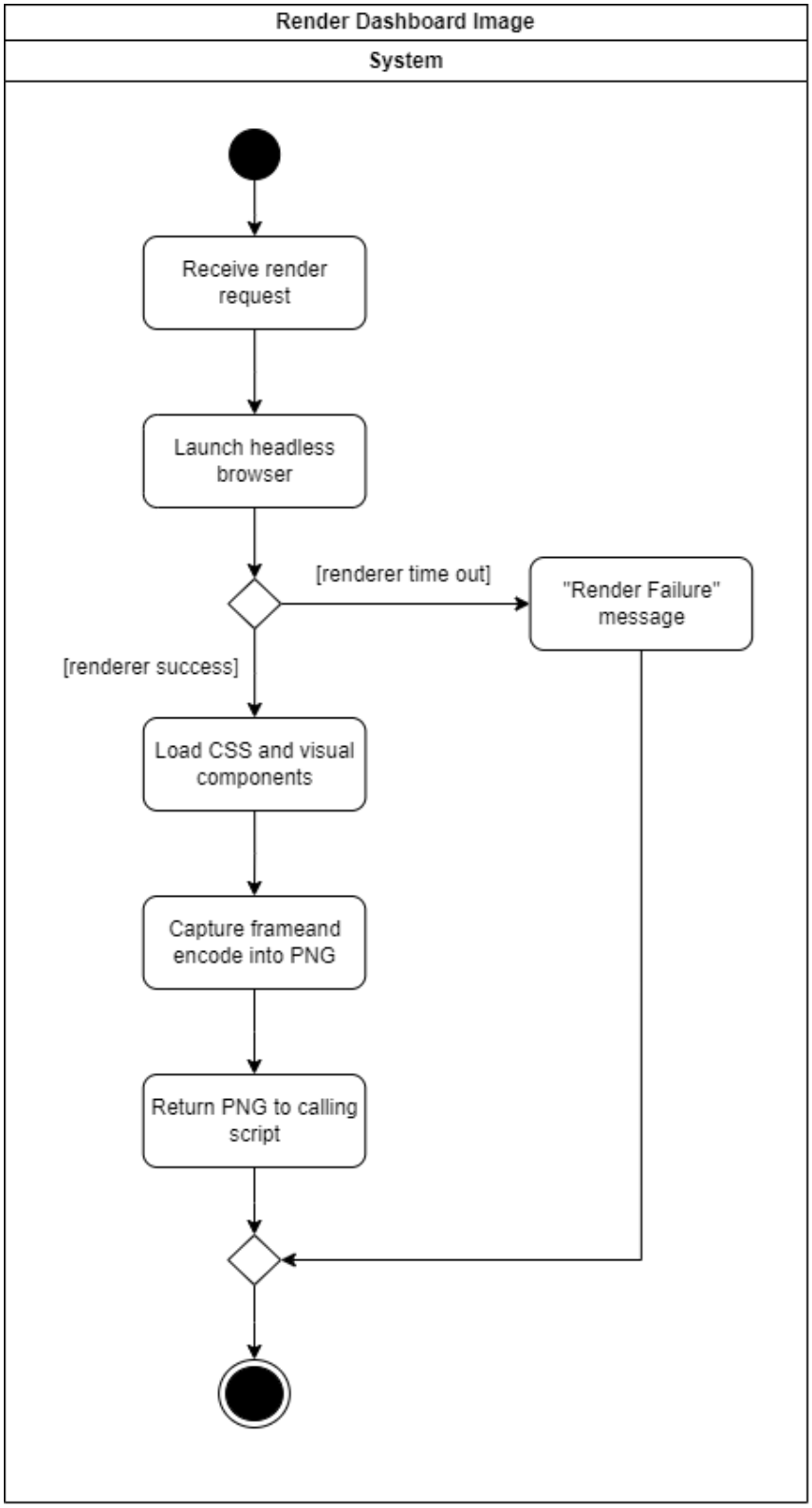


Figure 3.16 Activity diagram for render dashboard image

Figure 3.17 shows the activity diagram for generating expert recommendations.

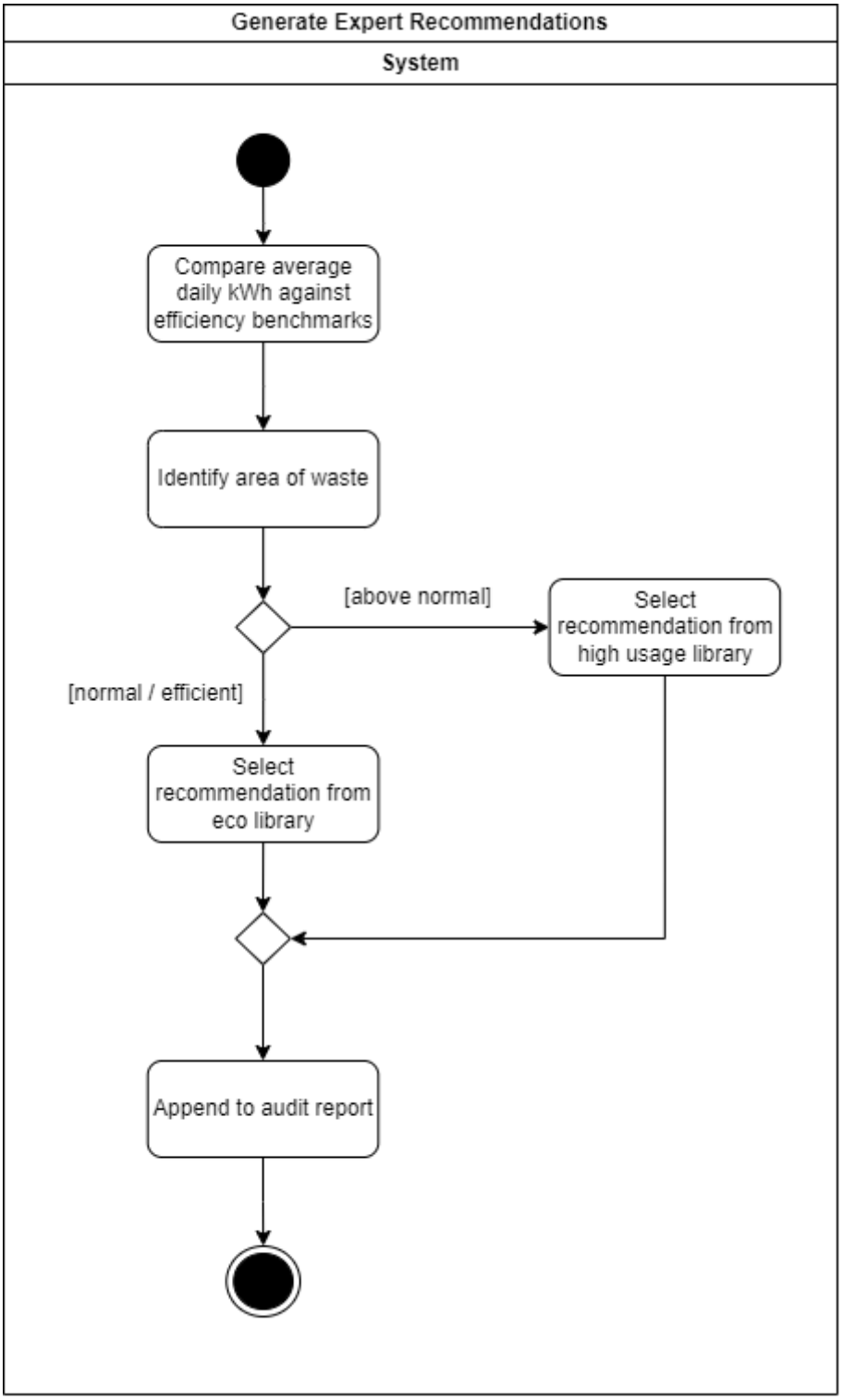


Figure 3.17 Activity diagram for generate expert recommendations

3.3 System Operational Flowchart

3.3.1 Edge Device (ESP32) Process Flow

The flowchart shown in Figure 3.18 provides details on the ESP32 firmware operational logic. It is intended to function reliably, independently and continuously from the time the device is turned on.

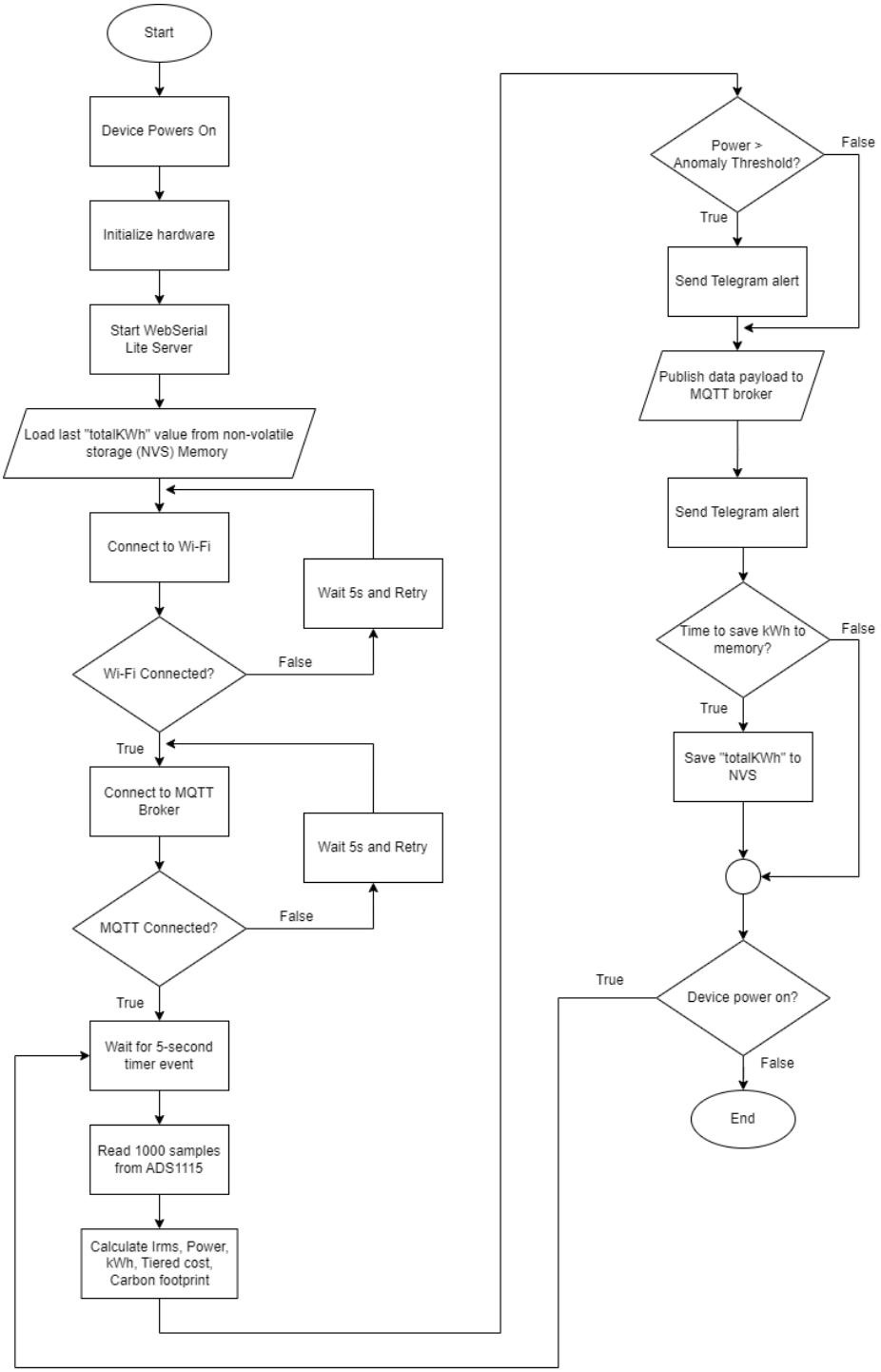


Figure 3.18 Flowchart for edge device (ESP32) process flow

As illustrated in the figure above, when the firmware first starts up, it initialises the required hardware parts, such as the GPIO pins and the ADC I2C interface. The firmware starts a WebSerial Lite Server after hardware initialisation. This eliminates the requirement for physical serial cable connections by establishing a headless wireless diagnostic interface that enables real-time monitoring and debugging using a web browser. Loading the device's NVS with the most recent totalKWh value is an essential first step in the setup process. In the event of a power interruption or a manual reboot, this guarantees that the system may restart precise energy tracking and cost calculation without resetting to zero. The device then moves into a series of robust connections. It starts by attempting to connect to the Wi-Fi network that has been configured. It will wait for five seconds and try again until a stable connection is made, if the connection fails. After connecting to Wi-Fi, it connects to the MQTT broker using the same reliable and self-correcting logic to guarantee that the device cannot move on to the monitoring phase until it has a verified communication channel to the backend.

Following the successful establishment of both network connections, the firmware moves into its main operational loop. A five-second timer event starts this loop. To guarantee a steady reading of the AC waveform, the device initially reads a large number of samples (1000 samples) from the ADS1115 ADC within each cycle. The final metrics will be computed after processing these raw samples including RMS current, instantaneous power, cumulative energy (kWh), estimated tiered cost and the carbon footprint. Immediately, the calculated power is compared to a predetermined anomaly threshold. The user receives a real-time alert over Telegram if the power above this threshold. The complete data payload is then published to the MQTT broker, regardless of whether an alarm was sent. Lastly, the firmware determines whether the most recent totalKWh data should be saved to the NVS. To extend the life of the flash memory, this is done periodically rather than every loop. After that, the procedure restarts the loop and waits for the subsequent five-second timer event to ensure constant and uninterrupted monitoring.

3.3.2 Backend and Intelligence Process Flow

Process A: Data Ingestion and Inference

The following flowcharts show the two processes that manage the backend data and intelligent process flow.

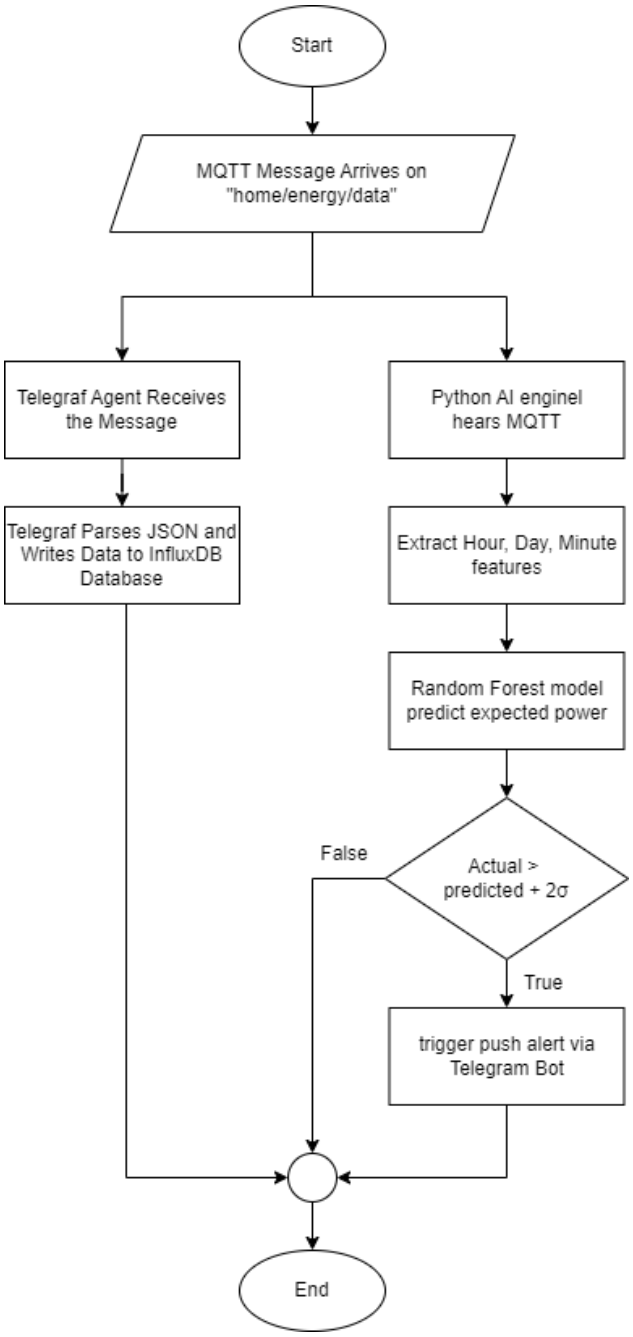


Figure 3.19 Flowchart for data ingestion and inference

As illustrated in Figure 3.19, the data ingestion process is a streamlined, event-driven pipeline that consistently transfers sensor values from the edge device to the TSDB. The entire flow is coordinated in the backend's Docker environment. The procedure begins when the ESP32 microcontroller publishes its data payload to the appropriate home/energy/data topic on the Mosquitto MQTT broker. The broker serves as a central messaging hub to make the data instantly available to any subscribed services. As one the subscriber, the Telegraf agent is constantly listening to this specific topic. When Telegraf receives a new message, it performs a vital function. It parses the JSON payload, extracting specific data elements such as power, current, and total energy, and then transforms them into the InfluxDB Line Protocol that is optimised for fast updates. This formatted data point is then instantly uploaded to the InfluxDB database for permanent storage and further analysis. This independent publish/subscribe design is ideal for IoT applications since the data-producing device (ESP32) is completely unaware of the end database. This will make the system both robust and scalable.

In addition, a dedicated Python AI engine acts as a parallel subscriber to the same MQTT topic to perform inference in real-time. The engine performs temporal feature extraction to determine the precise hour, day, and minute of the reading after receiving the data payload. A RFR model processes these contextual data and uses learned historical trends to forecast the anticipated power use. The actual power consumption is then compared to this anticipated baseline by the system. The system recognises an occurrence as a behavioural anomaly if the actual value surpasses the "Expected Power + 2σ" (two standard deviations) criterion. This instantly initiates a proactive push alert over the Telegram Bot API to give the user an intelligent alert about abnormal energy consumption without the need for manual monitoring.

Process B: Automated Machine Learning Lifecycle (Retraining)

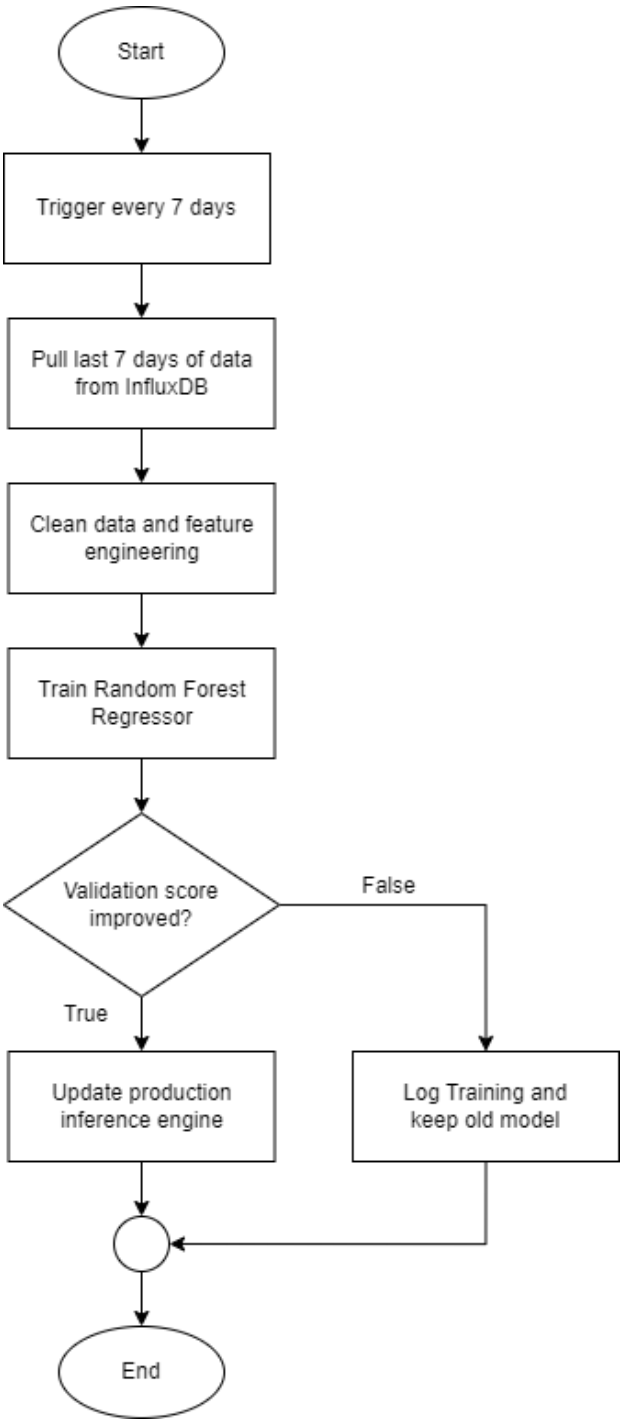


Figure 3.20 Flowchart for automated machine learning lifecycle

A self-adaptive Machine Learning lifecycle is used to make sure the anomaly detection system stays accurate when home patterns change over time. This process functions as an automated scheduled activity that initiates every seven days as shown in Figure 3.20. The model can account for the clear distinctions between weekday routines and weekend energy usage patterns because this particular interval is selected to encompass a full weekly cycle.

The lifecycle starts with a historical data sync that pulls time-series power consumption records from InfluxDB for the last seven days. The system then manages missing values and extracts important temporal features, specifically, the day of the week and the hour of the day from this raw data during a cleaning and feature engineering process. The RFR can recognise the cyclical nature of the home energy impact due to these qualities.

The engine trains a new RF model after the training set is ready. The new model must pass a validation gate before being deployed in order to preserve system reliability. The newly trained model's validation metrics, including Mean Absolute Error are compared by the system to the model that is presently in use in production. The system automatically updates the parameters of the production inference engine if the new model shows better accuracy. However, the system keeps the current model and logs the training session for diagnostic purposes if the new model does not demonstrate progress. This ongoing retraining loop greatly lowers false positives by ensuring that the expected baseline used for anomaly detection is always tailored to the user's current lifestyle.

3.3.3 User Interaction Process Flow (Telegram Bot)

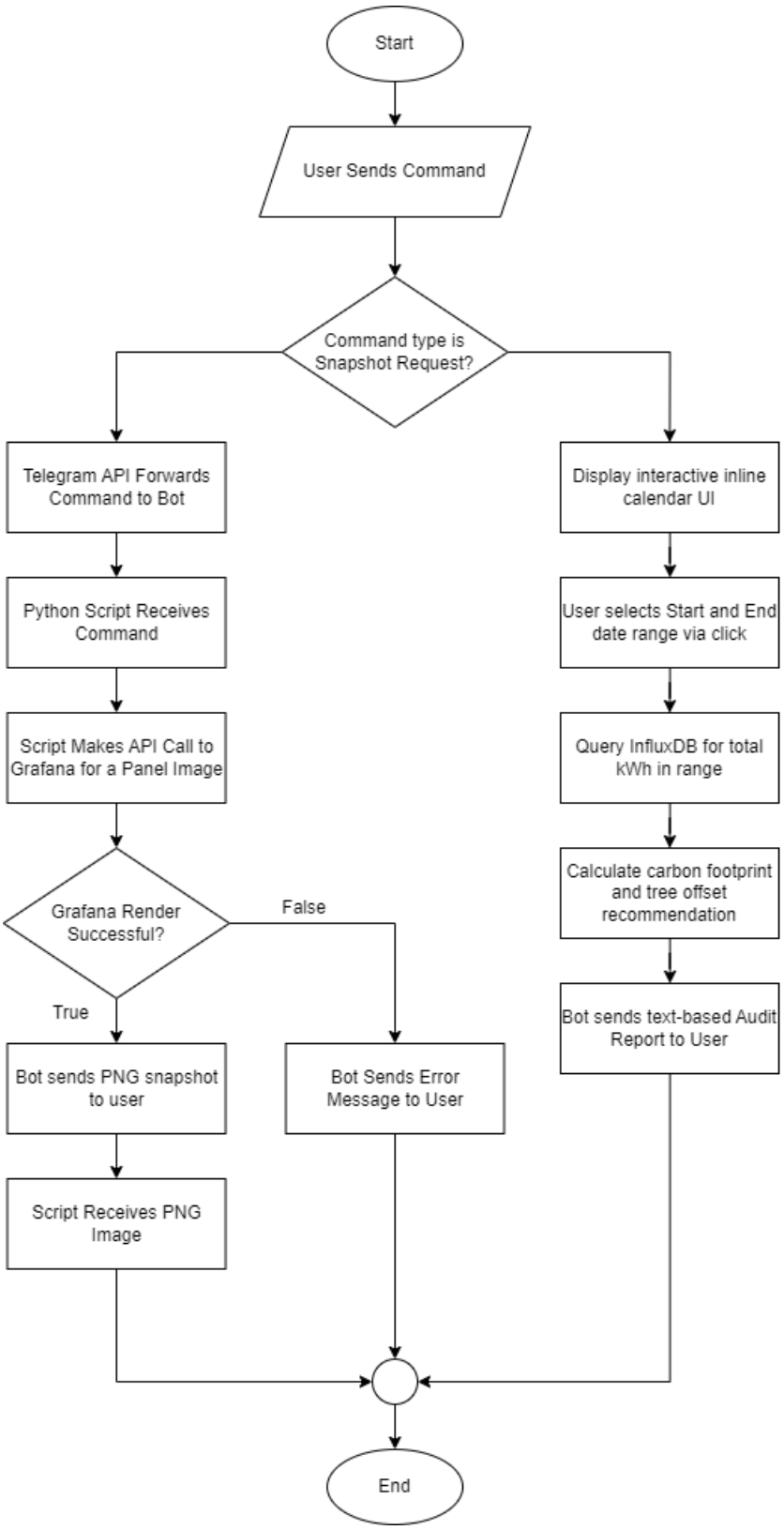


Figure 3.21 Flowchart for user interaction process flow

This process flow describes the system interactive, user-initiated pull feature to allows homeowners to retrieve precise energy statistics as needed. The entire process is event-driven, beginning with the end-user sending a command, such as /today from their device via the Telegram bot. The system initially determines the type of command in order to choose the proper response path, as shown in Figure 3.21.

In the event that the command is a Snapshot Request, the visual reporting branch is followed. This command is initially passed over the Telegram API to the Python bot script, runs as a service in the backend's Docker environment and is always listening for new messages. When the Python script receives a valid command, it acts as an API client, creating and sending an authenticated HTTPS GET request to the Grafana server's image rendering endpoint. This request defines the exact dashboard and panel to be rendered, as well as the required time range. The Grafana server then queries the InfluxDB database for the appropriate data, produces the requested dashboard panel as a image and provides this image to the Python script via HTTP. Finally, the script sends this image file back to the Telegram API, where it appears in the user chat as a photo message with a descriptive caption. This entire request-response loop converts the system from a passive dashboard to an interactive service that offers on-demand access to graphic energy information from any place.

Alternatively, the Python script initiates an interactive inline calendar UI if the user chooses the "Smart Audit" option. With just a few button clicks, the user can choose a particular start and end date range. The script queries InfluxDB for the total energy consumption (kWh) within that specific range when the dates are chosen. The user's carbon footprint and the tree offset needed to balance that energy use are then calculated by the engine as part of a sustainability analysis. The user receives a comprehensive text-based Audit Report from the bot at the end of the interaction, which offers quick expert insights into their long-term consumption patterns and environmental impact.

3.4 Data Processing and Mathematical Formulae

I. Root Mean Square (RMS) Current (I_{rms}) Calculation

The primary objective is to accurately calculate the RMS current of the AC waveform. This is achieved through a multi-step process within the calculateI_{rms}() function.

Step 1: ADC Value Sampling

The firmware takes *N* samples (N=1000) of the raw 16-bit ADC reading (ADC_{raw}) from the ADS1115.

Step 2: RMS of Raw ADC Values

A computationally efficient method is used to find the RMS value of the AC component of the raw ADC signal by mathematically removing the DC offset.

$$V_{rms_{adc}} = \sqrt{\left(\left(\sum \frac{ADC_{raw}{}^2}{N}\right) - \left(\sum \frac{ADC_{raw}}{N}\right)^2\right)} \quad \textbf{(1)}$$

Step 3: Conversion to RMS Voltage

The RMS ADC value is converted to an RMS voltage using the ADC's known voltage-per-bit resolution.

$$V_{rms} = V_{rms_{adc}} \times G_{adc} \quad \textbf{(2)}$$

where

G_{adc} = ADC gain, defined as $\frac{2.048\ V}{32767\ bits}$

Step 4: Conversion to RMS Current

The RMS voltage is then converted to RMS current using the CT sensor's specified ratio and a final software calibration factor to correct for component tolerances.

$$I_{rms} = V_{rms} \times R_{ct} \times C_{cal} \quad \textbf{(3)}$$

where

R_{ct} = CT sensor ratio, $\frac{100.0\ A}{1.0\ V}$

C_{cal} = software calibration constant, 1.350.

II. Real Power (P) Calculation

The instantaneous real power is estimated by multiplying the calculated RMS current by the nominal grid voltage.

$$P = I_{rms} \times V_{nominal} \times PF \quad \textbf{(4)}$$

where

P = estimated power in Watts (W).

$V_{nominal}$ = assumed nominal voltage, 230.0 V.

PF = real-world efficiency factor, 0.85.

III. Cumulative Energy (E) Calculation

The total energy consumed is calculated by numerically integrating power over time. This value is updated at each reporting interval.

$$E_{interval} = \frac{P \times T_{interval}}{3{,}600{,}000} \quad \textbf{(5)}$$

$$E_{total} = E_{previous} + E_{interval} \quad \textbf{(6)}$$

where

$E_{interval}$ = energy consumed in kilowatt-hours (kWh) during the interval.

$T_{interval}$ = send interval time in seconds (5s).

Denominator *3,600,000* = conversion factor from Watt-seconds to kWh.

IV. Tiered Cost (C) Calculation

The final estimated monthly cost is calculated using a piecewise function that models the Malaysian domestic electricity tariff structure.

Step 1: Determine Base Variable Rate (R_{total})

$$R_{energy} = 0.2703\ RM/kWh\ \ if\ E_{total} \leq 1500$$
$$R_{energy} = 0.3703\ RM/kWh\ \ if\ E_{total} > 1500 \quad \textbf{(7)}$$

The total variable rate is then calculated as:

$$R_{total} = R_{energy} + R_{capacity} + R_{network} \quad \textbf{(8)}$$

where

$R_{capacity}$ = 0.0455 RM/kWh

$R_{network}$ = 0.1285 RM/kWh

Initial Bill Amount: $B_{initial} = E_{total} \times R_{total}$

Step 2: Automatic Fuel Adjustment (AFA) and Retail Charges

For users exceeding the 600 kWh threshold, the system applies the AFA fuel rebate and a fixed retail service charge:

If $E_{total} > 600$ kWh:

$$B_{adj} = B_{initial} + (E_{total} \times -0.027) + 10.00 \quad \textbf{(9)}$$

where

-0.027 is fuel adjustment credit.

Step 3: Energy Efficiency Incentive (Rebate)

To promote conservation, the system applies a government-mandated rebate for consumption levels of 1000 kWh and below:

If $E_{total} \leq 1000$ kWh:

$$B_{net} = B_{adj} - (E_m \times 0.1050) \quad \textbf{(10)}$$

Step 4: Statutory Surcharges (RE Fund and SST)

Finally, the system applies the mandatory Renewable Energy (RE) Fund and the Service Tax (SST). The RE Fund only applies to consumers using more than 300 kWh, while the 8% SST is applied only to the high-consumption tier (> 600 kWh).

Renewable Energy Fund:

If $E_{total} > 300$ kWh, add 1.6% to the total.

$$B_{statutory} = B_{net} + (B_{net} \times 0.016) \quad \textbf{(11)}$$

Service Tax (SST):

If $E_{total} > 600$ kWh, add 8% to the total.

$$C_{final} = B_{statutory} + (B_{statutory} \times 0.08) \quad \textbf{(12)}$$

Step 4: Minimum Bill Enforcement

If $C_{final} > 0$ and $C_{final} < 3.00$, the final bill is rounded the value up to RM 3.00 to ensure metrological consistency with actual utility provider invoices.

Table 3.17 Summary of electricity tariff components

Component	**Threshold**	**Rate / Factor**	**Impact**
Energy Rate (Low)	≤1500 kWh	0.2703 RM/kWh	Base Charge
Energy Rate (High)	>1500 kWh	0.3703 RM/kWh	Base Charge
Capacity & Network	Always	0.1740 RM/kWh	Base Charge
AFA Fuel Credit	>600 kWh	-0.027 RM/kWh	Discount
Retail Charge	>600kWh	RM 10.00	Fixed Fee
EE Incentive	≤1000 kWh	-0.1050 RM/kWh	Rebate
RE Fund	>300 kWh	1.6%	Tax
Service Tax (SST)	>600 kWh	8%	Tax

Table 3.17 summarises the multi-tiered rates, government incentives, and statutory surcharges used by the system billing algorithm in accordance with the Malaysian electricity tariff structure.

V. Environmental Impact Metrics

To support the Smart Energy Audit feature, the system calculates the user's carbon footprint.

Carbon Footprint (CO_2):

Calculated using the Malaysia Grid Emission Factor of 0.674 kg CO2/kWh.

$$CO_2\ (kg) = E_{total} \times 0.674 \quad \textbf{(13)}$$

Environmental Offset (Trees):

Calculated based on the average absorption rate of a mature tree (22.0 kg CO2/year):

$$Tree = \frac{CO_2(kg)}{22.0} \quad \textbf{(14)}$$

3.5 Project Milestone

	Name	Duration	Start	Finish
1	Finalize architecture and planning	5 days	23/06/2025, 8:00am	27/06/2025, 5:00pm
2	Select and order hardware	7 days	26/06/2025, 8:00am	04/07/2025, 5:00pm
3	Assemble circuit & connect components	6 days	03/07/2025, 8:00am	10/07/2025, 5:00pm
4	Write ESP32 firmware (sensor reading)	6 days	09/07/2025, 8:00am	16/07/2025, 5:00pm
5	Calibrate sensors (CT Clamp and ADC)	5 days	15/07/2025, 8:00am	21/07/2025, 5:00pm
6	Implement MQTT protocol (ESP32 to Cloud)	4 days	18/07/2025, 8:00am	23/07/2025, 5:00pm
7	Set up InfluxDB & connect MQTT data	4 days	21/07/2025, 8:00am	24/07/2025, 5:00pm
8	Build basic Grafana dashboard	5 days	23/07/2025, 8:00am	29/07/2025, 5:00pm
9	Collect and log real-time energy data	7 days	25/07/2025, 8:00am	04/08/2025, 5:00pm
10	Functional testing & UI enhancements	5 days	01/08/2025, 8:00am	07/08/2025, 5:00pm
11	System debugging and validation	6 days	06/08/2025, 8:00am	13/08/2025, 5:00pm
12	Report writing	6 days	14/08/2025, 8:00am	21/08/2025, 5:00pm
13	Report review, formatting, final edit	5 days	22/08/2025, 8:00am	28/08/2025, 5:00pm
14	Presentation preparation	4 days	26/08/2025, 8:00am	29/08/2025, 5:00pm

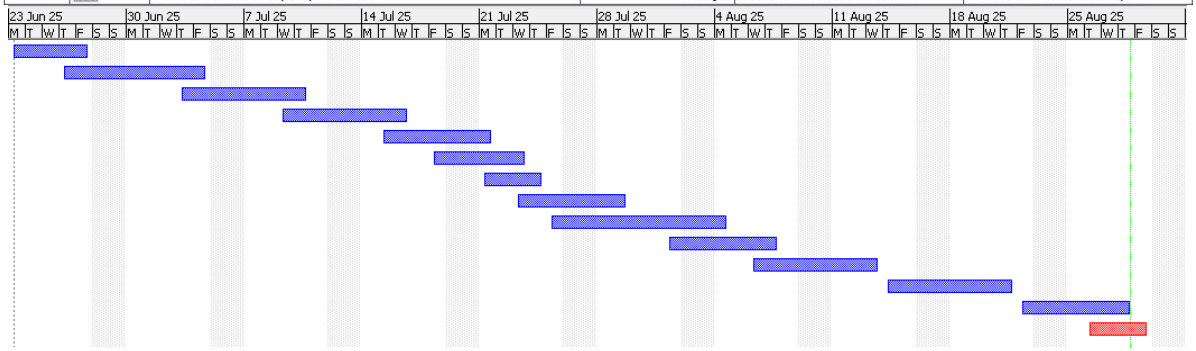


Figure 3.22 FYP1 timeline

	Name	Duration	Start	Finish
4	Optimize model & tune parameters	5 days	17/02/2026, 8:00 am	23/02/2026, 5:00 pm
5	Integrate ML with live cloud data	5 days	24/02/2026, 8:00 am	02/03/2026, 5:00 pm
6	Develop Telegram alert system	4 days	03/03/2026, 8:00 am	06/03/2026, 5:00 pm
7	Connect model output to alert system	4 days	05/03/2026, 8:00 am	10/03/2026, 5:00 pm
8	Full pipeline testing (ESP32 to Alert)	5 days	09/03/2026, 8:00 am	13/03/2026, 5:00 pm
9	Dashboard improvements and custom alerts	4 days	12/03/2026, 8:00 am	17/03/2026, 5:00 pm
10	Scenario testing and energy behavior analysis	5 days	16/03/2026, 8:00 am	20/03/2026, 5:00 pm
11	Full system testing (all modules)	11 days	19/03/2026, 8:00 am	02/04/2026, 5:00 pm
12	Report writing	11 days	03/04/2026, 8:00 am	17/04/2026, 5:00 pm
13	Review, feedback integration	5 days	18/04/2026, 8:00 am	24/04/2026, 5:00 pm
14	Final demo preparation and report submission	5 days	27/04/2026, 8:00 am	01/05/2026, 5:00 pm
15	Presentation preparation	3 days	30/04/2026, 8:00 am	04/05/2026, 5:00 pm

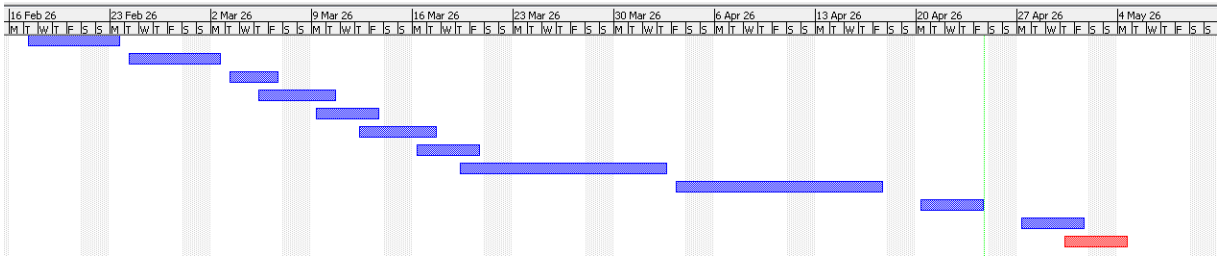


Figure 3.23 FYP2 timeline

3.6 Agile Development Methodology

Figure 3.24 shows the Agile development methodology model.

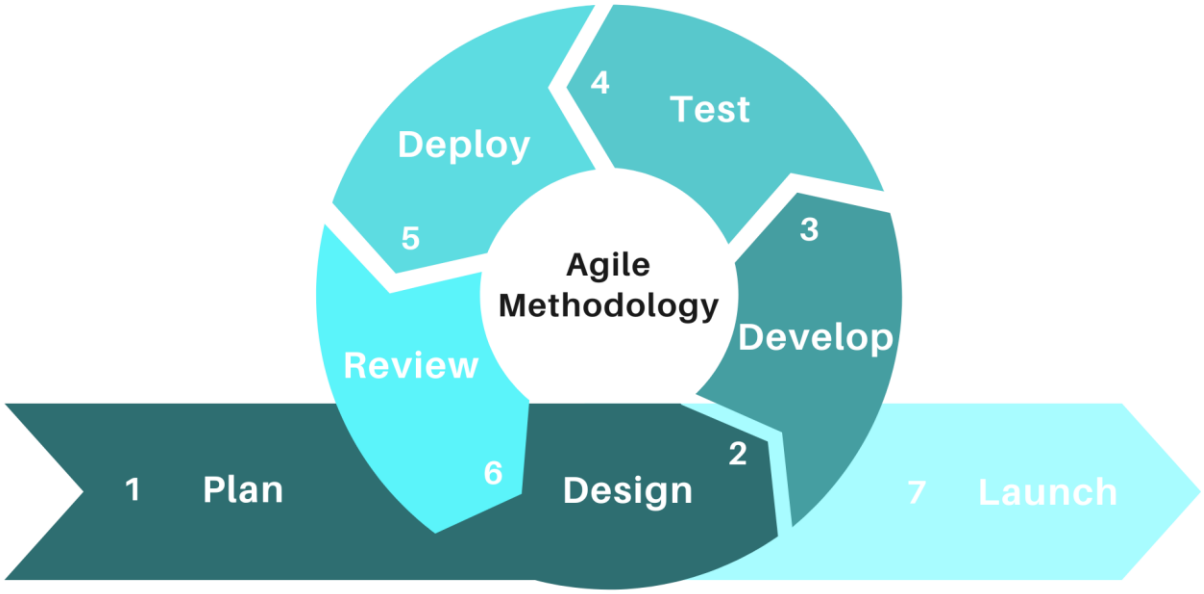


Figure 3.24 Agile methodology model

As shown in the figure above, the Agile methodology is used as the development framework for this project. This iterative technique was chosen because of its inherent flexibility. It is excellent for a technology-based project that requires the concurrent development of hardware and software components. Agile promotes continuous improvement and close collaboration to allow the project to adapt to changes and obstacles encountered during the development cycle. The Agile cycle for the project is broken into seven phases, plan, design, development, testing, deployment, review, and launch. Each phase is critical to ensure that the system is constructed efficiently and meets all user and technical expectations.

The first phase is planning, where the project objectives, requirements, and scope are defined. This is a comprehensive research process that helps define the system core functionality, lists the required hardware and software components and estimates the development timeline. This fundamental planning is related to the early stages of similar research, such as the work breakdown structure that starts with a detailed definition of requirements before any design work begins [5]. The planning stage is expanded to cover requirements for a ML inference engine, billing logic rearrangement and the incorporation of independent power requirements.

The design phase involves transforming these requirements into tangible system architecture and UI design. This includes building the hardware setup, planning the cloud integration and creating the dashboard. During this phase, important design drawings such as system block diagrams and schematic diagrams are created in accordance with the standard method demonstrated by [6],[7]. These diagrams are critical for directing future development and ensuring that all components integrate properly. Additionally, this stage involved creating complex UML diagrams, such as use case, activity diagrams and sequence diagrams to simulate the two-way communication between the AI engine and the Telegram Bot.

The development phase is where the real system is created. The hardware assembly contains the ESP32, CT sensors and ADC modules on the prototype board. The software includes implementing data collecting hardware, setting the cloud databases (InfluxDB) and creating the visualization dashboard (Grafana). This phase is extremely iterative to allow for continuous input and incremental resolution of issues as they emerge, a flexible technique required for hardware-software joint design. RFR for anomaly detection, WebSerial Lite diagnostic interface, and NVS persistence logic to make sure the energy odometer survives power reboots are some of the major technical innovations made during this day and age.

The testing process is critical to ensure that the system performs as intended. This involves several stages of validation, including unit testing of specific components, such as comparing CT sensor readings to a calibrated meter, as conducted by [5], integration testing to ensure data flows correctly from the sensor to the cloud and dashboard usability evaluation. Identifying and addressing bugs early in this phase minimizes the likelihood of big issues later in the project. This included thorough latency testing to evaluate the speed of the Telegram Bot reporting pipeline and spike testing using the tactile button to verify the real-time AI alert response.

During the Deployment phase, the working prototype is deployed in a real-world setting to collect actual energy consumption data. This ensures that all system components, from

sensing to visualization, work consistently. The phase focusses on resolving any remaining issues with network connectivity, data accuracy and interface responsiveness. The final deployment included placing the entire backend to a production-level Docker containerised environment and configuring the edge device to a separate 5V/3A power supply to ensure 24/7 reliability.

The Review phase then gathers feedback from project stakeholders, including supervisors and test users, in order to analyze the system performance, identify areas for improvement and prepare the system for finalization. The accuracy of the ML expected baseline and the compliance of the cost computations were the specific areas of focus during the review process.

Finally, the Launch phase concludes the development and testing cycle. During this phase, the system's final version is formally delivered, filled with detailed documentation, working hardware and fully functional software interfaces. Although the project is still in its early planning stages, the use of the Agile methodology assures that it is sensitive to problems and input throughout the development lifecycle.

3.7 Concluding Remark

This chapter provided a strong methodological framework for the project by outlining the hardware and software requirements, architectural designs and operational logic. The combination of high-precision sensing at the edge and a containerised intelligence layer in the cloud creates a scalable and persistent basis. The system is built to be iterative and adaptive using an Agile development methodology.

Chapter 4

System Design

This chapter describes the system overall technical design, including hardware schematics, time-series database schemas and the machine learning model architecture utilised for intelligent anomaly detection.

4.1 System Block Diagram

Figure 4.1 illustrates the system block diagram of home energy monitoring and alert system.

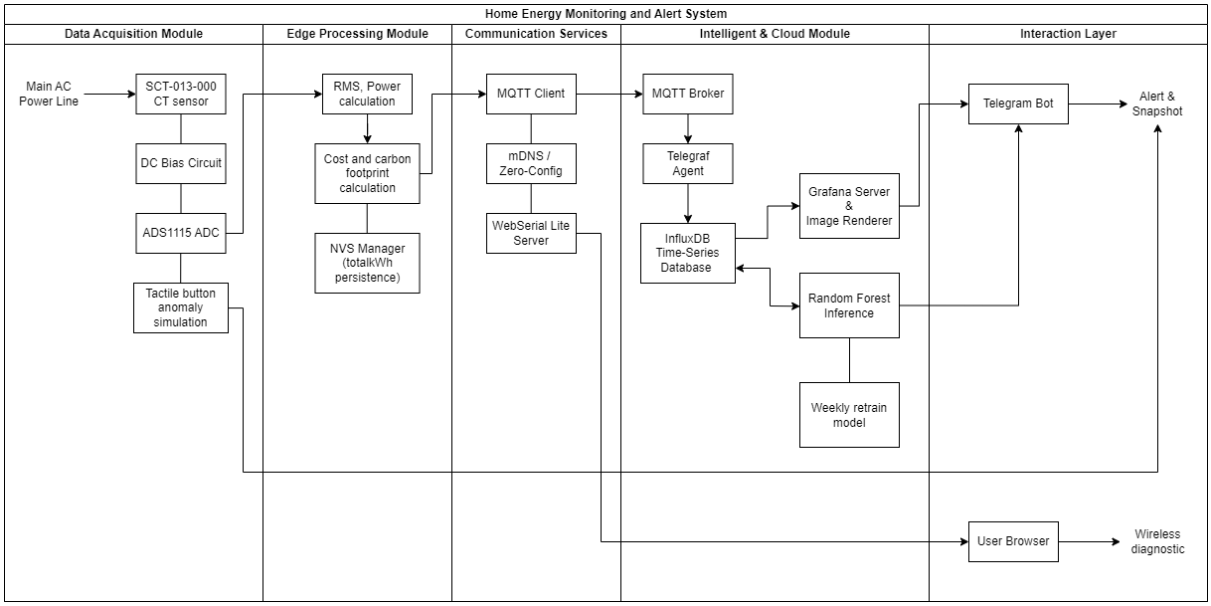


Figure 4.1 System block diagram of the project

The system block diagram in figure above depicts the functional composition of hardware and software modules and mapping the data flow from physical sensing to intelligent user interaction. The process begins in the Data Acquisition Module, the raw AC signals are recorded non-invasively by the SCT-013-000 sensor, processed by a DC bias circuit and digitised by the high-precision ADS1115 ADC. This digital stream is sent to the Edge Processing Module then the ESP32 performs local computing activities such as TRMS power calculations and carbon footprint computations. Crucially, this module employs a NVS Manager to maintain a permanent energy odometer to guarantee that cumulative data survives power interruptions.

Communication Services manage connectivity and system health by using mDNS for zero-config network discovery and hosting a WebSerial Lite server for headless wireless diagnostics. Data is sent via the MQTT protocol to the Intelligent & Cloud Module, a containerised environment where the Telegraf agent enters telemetry into the InfluxDB time series database. This module contains the system primary intelligence, a RF Inference engine that identifies context-aware anomalies by comparing real-time usage to a predetermined baseline. An automated lifecycle operation retrains the model every week using past data to guarantee long-term accuracy. Finally, the Interaction Layer allows for bidirectional communication with a Python-based Telegram Bot sending proactive AI-driven alerts and on-demand Grafana snapshots while the user browser provides advanced system monitoring through the diagnostic interface.

4.2 System Components Specifications and Component Diagram

Figure 4.2 shows the component diagram of home energy monitoring system.

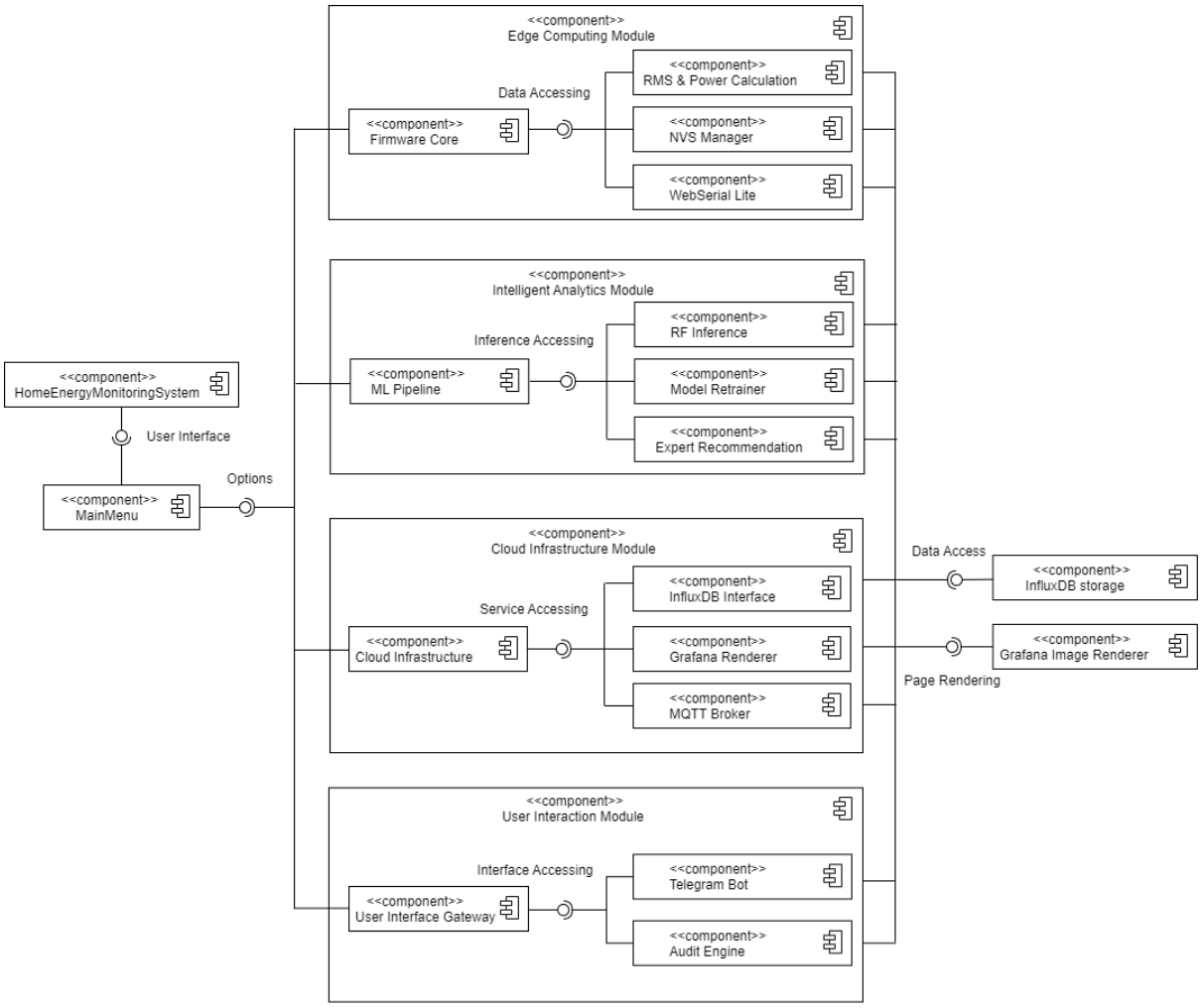


Figure 4.2 Component diagram of the project

As shown in the figure above, the ESP32 microcontroller contains the Edge Computing Module which acts as the system's physical and logical foundation. Its main function is to rapidly acquire analogue signals from CT sensors and convert them into relevant power measures. By executing True RMS computations at the edge, the module minimises computational load on the cloud backend while maintaining real-time responsiveness. Furthermore, this module includes a resilience layer made up of the NVS Manager for data persistence and the WebSerial Lite server for headless diagnostics to allow the hardware to function as a dependable, autonomous sensing node.

The Intelligent Analytics Module is the intelligence of the system that transforms raw data into predictive insights. This module developed using a Python-based ML framework, uses a RFR to learn the household's unique cyclical energy habits. By extracting temporal information, the module is able to differentiate between expected usage and actual anomalies, moving away from rigid static thresholds. The automated model retrainer keeps the system precise over time by responding to weekly lifestyle changes and the expert recommendation engine converts statistical deviations into actionable energy-saving suggestions for the user.

The Cloud Infrastructure Module is a Docker-managed containerised environment designed for fast data ingest and long-term storage. This module is built around the TIG stack which provides a traceable data stream. The Mosquitto MQTT broker enables low-latency connectivity, while InfluxDB serves as a high-performance time-series repository for precise energy records. This module also contains automated rendering service to enable the system to bridge the gap between complicated data dashboards and the simplified Telegram interface by transforming dynamic web panels into static PNG graphics.

The User Interaction Module acts as the primary gateway for end-user interactions with a focus on accessibility and bidirectional communication. Rather than depending on a passive dashboard, this module uses Telegram Bot as an interactive terminal. This interface allows users to receive proactive AI-driven alerts and pull visual reports on demand. The integrated Audit Engine enables users to conduct comprehensive sustainability analysis, calculating financial expenditures and visualising environmental impact using carbon footprint and tree-offset metrics. This encourages proactive energy management habits.
Table 4.1 below displays the system component specifications of the component diagram.

Table 4.1 System component specifications

Main Component	Sub-component	Specification and Functionality
Edge Computing Module	Firmware Core	Manages hardware-level tasks on the ESP32, such as I2C connection with the ADS1115 and mDNS networking.
	RMS & Power Calculation	Uses high-frequency sampling (1000 samples per cycle) to calculate True RMS current and instantaneous power.
	NVS Manager	Utilises the Preferences to implement data persistence. Every 15 seconds, the Preferences.h library stores the total kWh in flash memory.
	NVS Manager	Offers a headless WebSocket-based diagnostic interface for wireless monitoring via a web browser.
Intelligent Analytics Module	ML Pipeline	Data normalisation and temporal feature engineering (hour, day, and minute) are handled by this Python-based engine.
	RF Inference	Predicts the expected power baseline using a RFR and uses σ-scaling to find anomalies.
	Model Retrainer	An automatic lifecycle script that triggers the AI model with the most recent historical data every seven days.
	Expert Recommendation	Logic engine that uses Malaysian NEEAP guidelines to produce energy-saving recommendations based on audit findings.
Cloud Infrastructure Module	MQTT Broker	An instance of Mosquitto that enables asynchronous, lightweight message transmission between the backend and the edge.
	Telegraf Agent	Acts as the gateway for data ingestion, parsing JSON payloads and writing them to the TSDB.
	InfluxDB Interface	Manages the API connection to the time-series storage, which includes complicated Flux queries.
	Grafana Renderer	Synthesises dashboard panels into PNG files by coordinating with the Grafana-image-renderer container.
User Interaction Module	Telegram Bot	Primary bidirectional user interface that manages photo-message delivery, notifications, and command parsing.
	Audit Engine	A dedicated module that computes environmental offsets, carbon footprints and total costs.

4.3 System Components Interaction Operations

4.3.1 Sequence Diagram

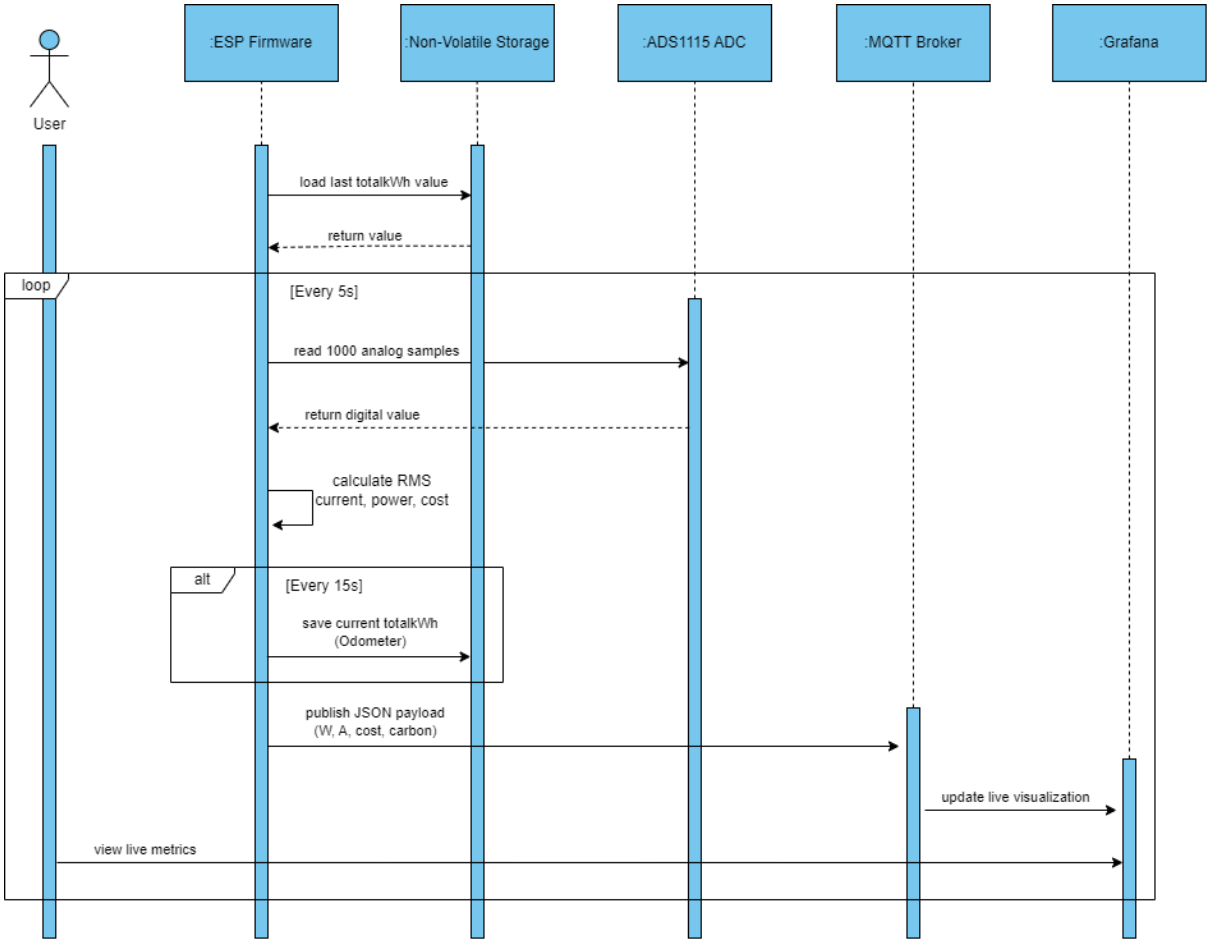


Figure 4.3 Sequence diagram for monitor real-time energy usage and persistence

Figure 4.3 shows how the ESP32 firmware initiates continuous monitoring and persistence loop upon system boot. To maintain data continuity, the method begins by collecting the last recorded energy odometer reading from NVS. The system then enters a 5-second sampling loop in which it receives raw data from the ADS1115 ADC, calculates power and costs and commits the updated energy value to the NVS every 15 seconds to ensure power continuity. Finally, the processed payload is released over MQTT to updates the user live visualisations.

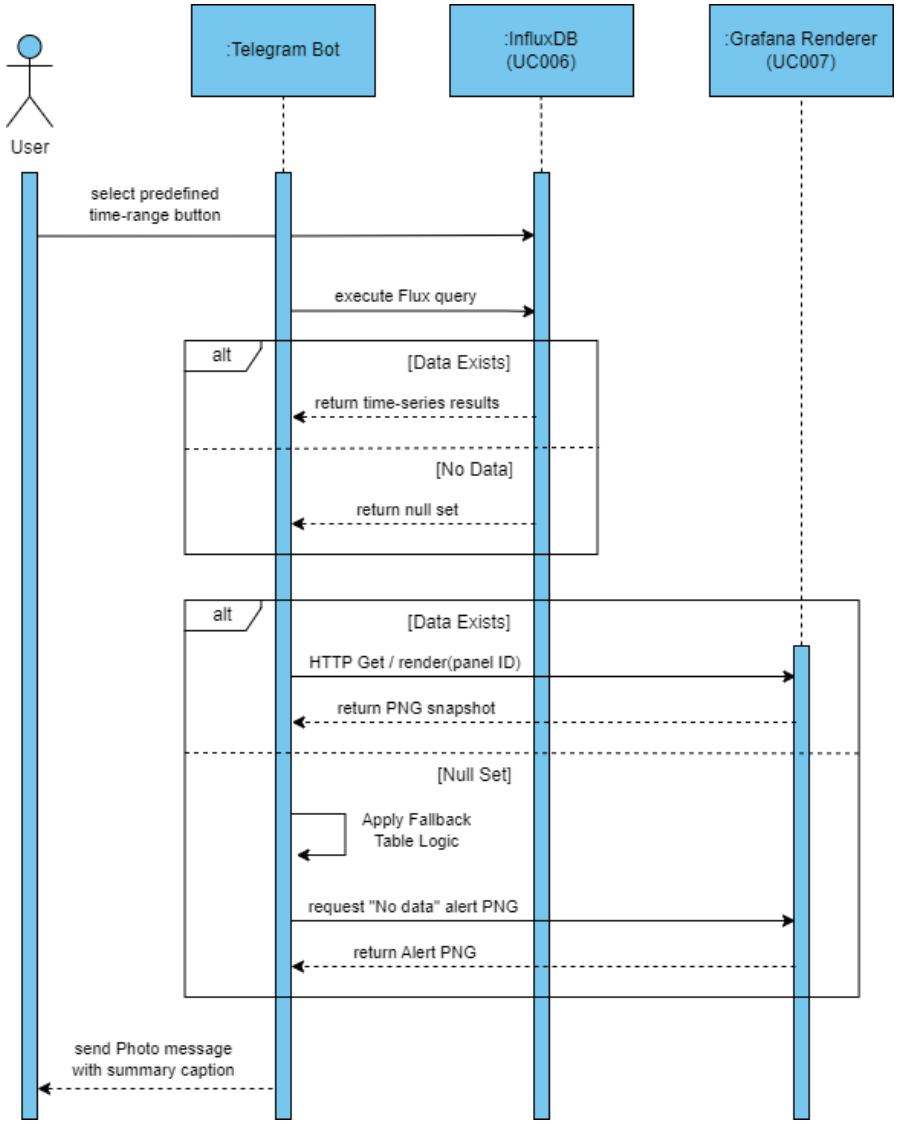


Figure 4.4 Sequence diagram for request on-demand visual energy report

Figure 4.4 illustrates the request-response process utilised to deliver visual insights using the Telegram Bot. When a user chooses a particular time frame, the bot queries InfluxDB checks for the presence of data. If the result set is empty, a fallback logic is used to generate a "No Data" message rather than a system crash. For correct data, the bot instructs the Grafana Renderer to create a high-resolution PNG picture of the required dashboard panel and then send to the user chat with a descriptive summary caption.

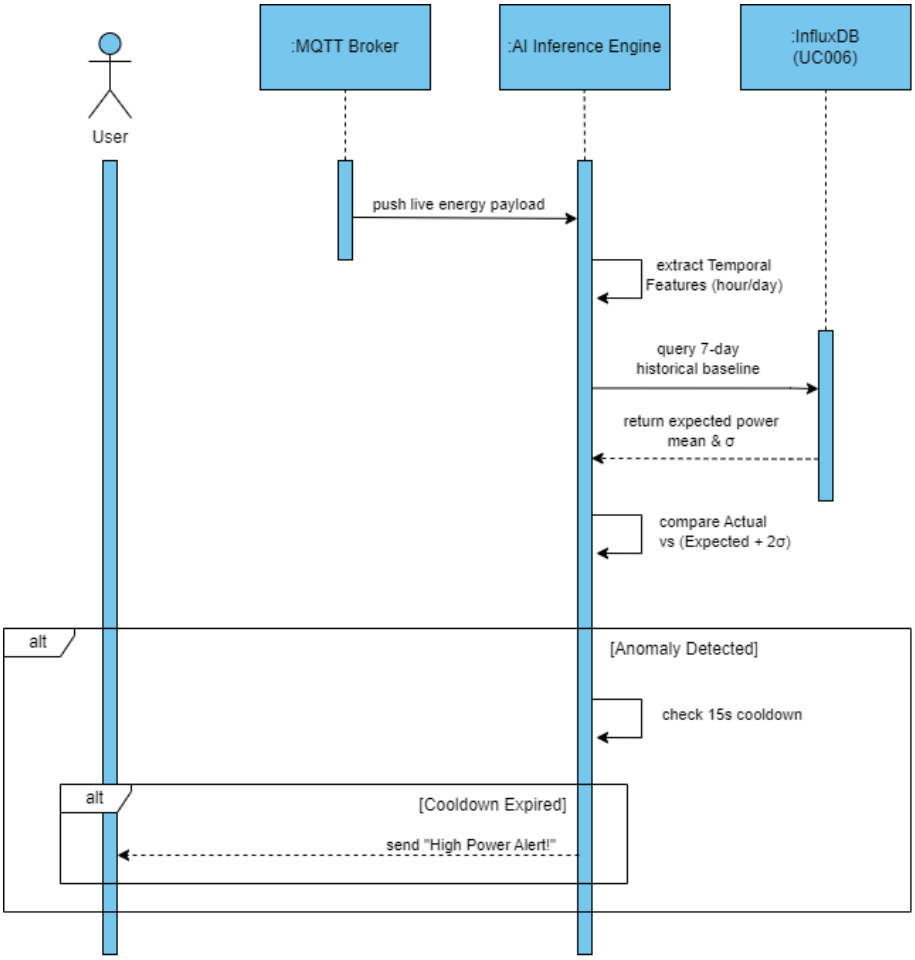


Figure 4.5 Sequence diagram for proactive anomaly detection and alerting

Figure 4.5 displays the intelligence layer, which has the AI Inference Engine monitoring real MQTT data in the background. After receiving the payload, the engine extracts temporal data and queries InfluxDB for a 7-day historical baseline to generate a context-aware threshold using standard deviation (σ) scaling. If the actual power consumption exceeds this dynamic threshold, the system activates a 15-second cooldown timer to avoid notification spamming before sending an automated “High Power Alert!” message to the user’s Telegram Bot.

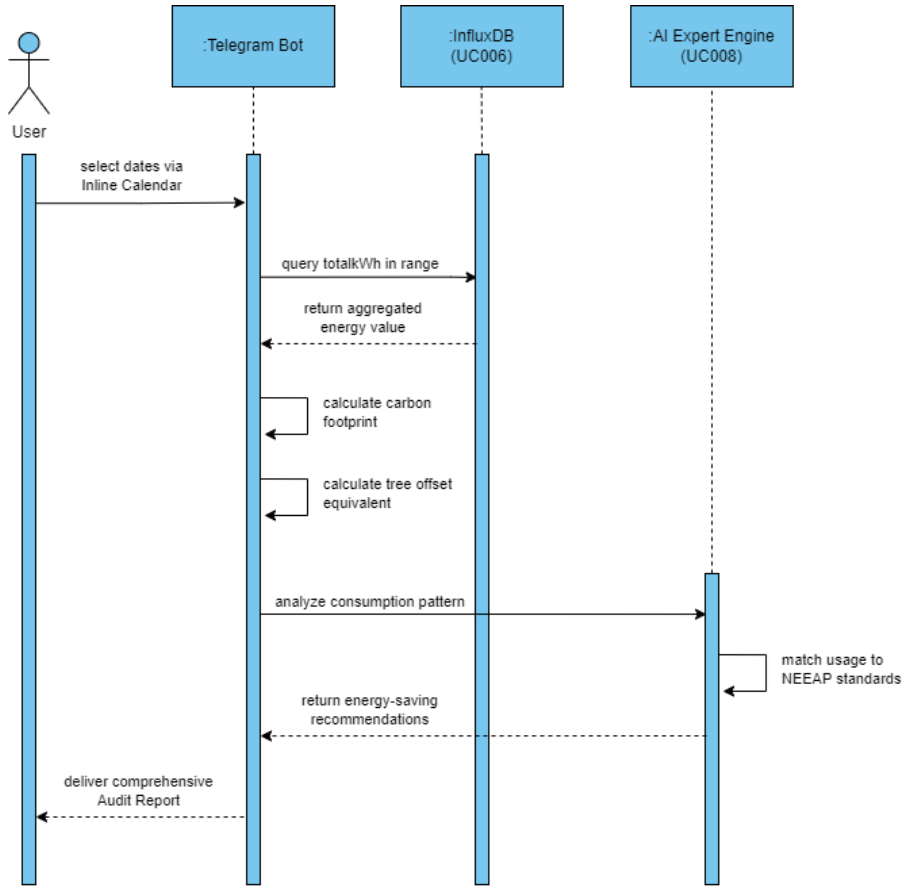


Figure 4.6 Sequence diagram for conduct smart energy and carbon audit

Figure 4.6 represents the multi-step analysis process for conducting a sustainable-based energy audit. The user inputs a date range into an interactive inline calendar and prompts the bot to query and aggregate total usage data from InfluxDB. The system then computes specialised environmental indicators like carbon footprints and tree-offset equivalents, while an AI Expert Engine compares usage patterns to NEEAP guidelines. The interaction ends with the release of a report that includes energy-saving recommendations.

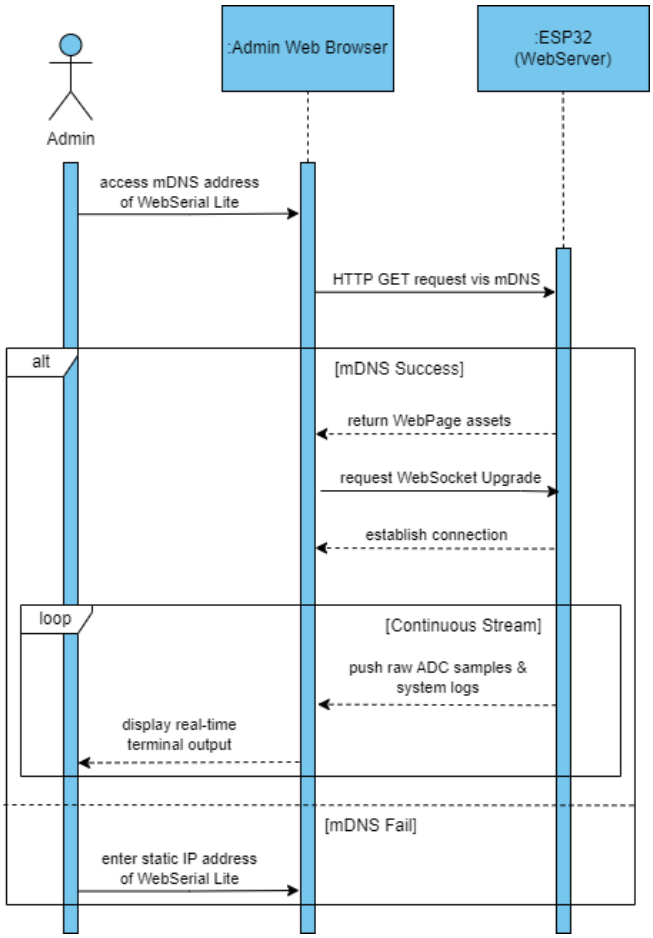


Figure 4.7 Sequence diagram for access wireless headless diagnostics

Figure 4.7 outlines the administrative maintenance path intended for cable-free system troubleshooting. The administrator connects to the diagnostic interface via a local mDNS address that prompts the ESP32's internal web server to return the required webpage assets. When the browser requests and initiates a WebSocket connection, the ESP32 starts sending a continuous, high-speed stream of raw ADC samples and system logs directly to the browser terminal. In the event that mDNS resolution fails, a fallback method is added to allow access via static IP address.

4.3.2 State Transition Diagram

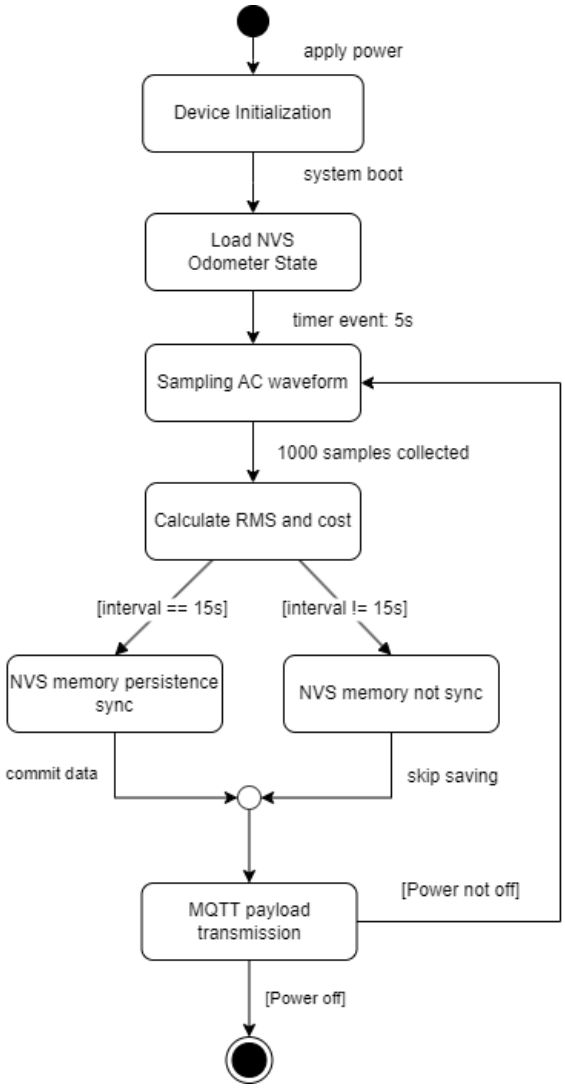


Figure 4.8 State transition diagram for monitor real-time energy usage and persistence

Figure 4.8 describes the internal operational lifespan of the ESP32 firmware, with an emphasis on data integrity. After acquiring power, the system initialises and restores its previous energy “odometer” state from NVS memory before starting a continuous sampling loop. Within this loop, the system switches every 5 seconds to calculate power metrics and with an extra conditional switch every 15 seconds to commit data to nonvolatile storage. This ensures that cumulative energy values are saved and recoverable if the device loses power.

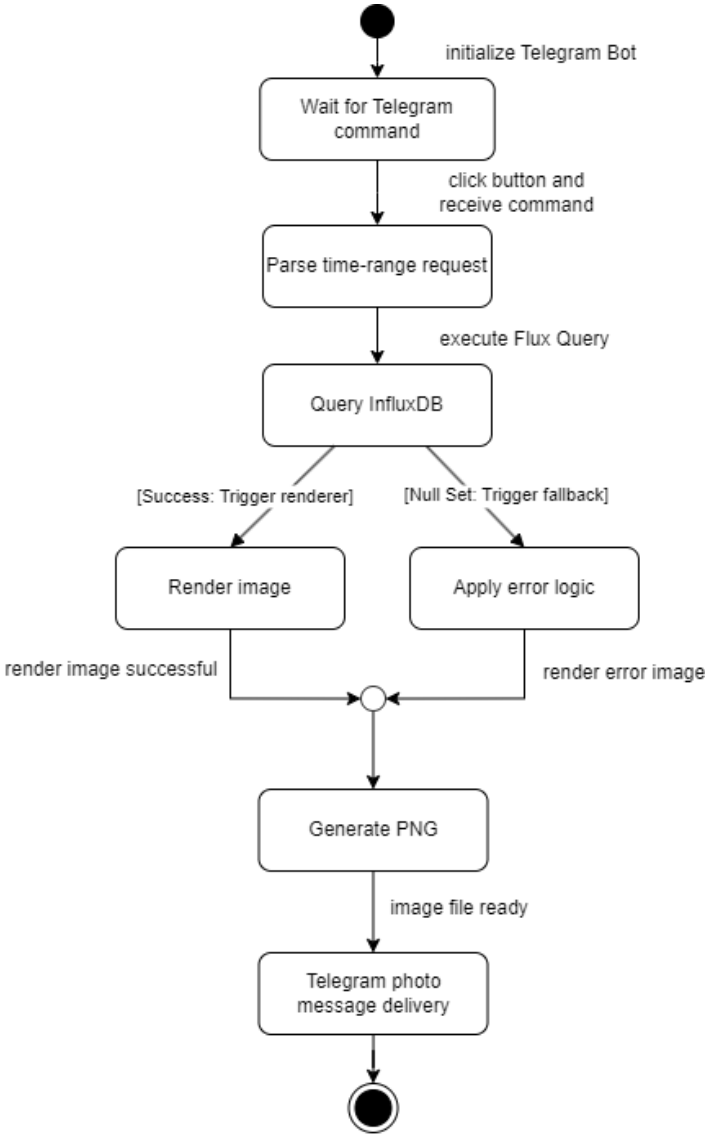


Figure 4.9 State transition diagram for request on-demand visual energy report

Figure 4.9 depicts the request-response loop for creating graphical snapshots of energy trends using Telegram. The bot begins in an idle state before changing to parsing and database queries when a user enters a specified time range. To maintain system robustness, the logic alternates between a successful rendering stage and an error-handling fallback state if no data is available. Both routes eventually converge to produce a PNG picture file and are sent as a photo message to the user chat.

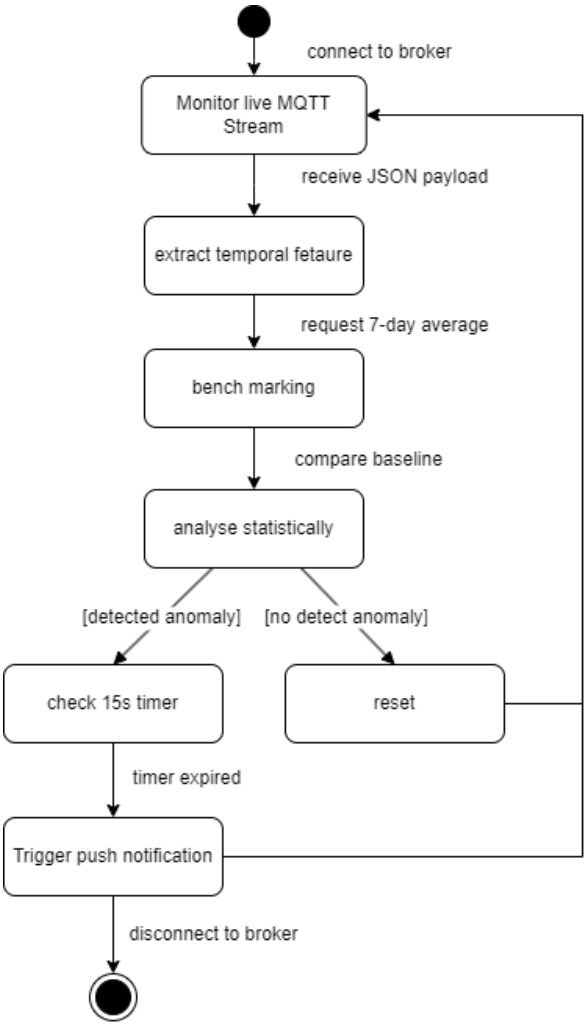


Figure 4.10 State transition diagram for proactive anomaly detection and alerting

Figure 4.10 demonstrates the intelligence engine's state transitions while evaluating live data for unexpected behaviour. The system stays in monitoring mode until a payload is received, then initiates temporal feature extraction and a benchmarking phase in which current consumption is compared to a 7-day historical baseline. Depending on the statistical analysis, the system either resets or enters an alerting phase, it will check that a 15-second cooldown timer has expired before sending a notification to the user.

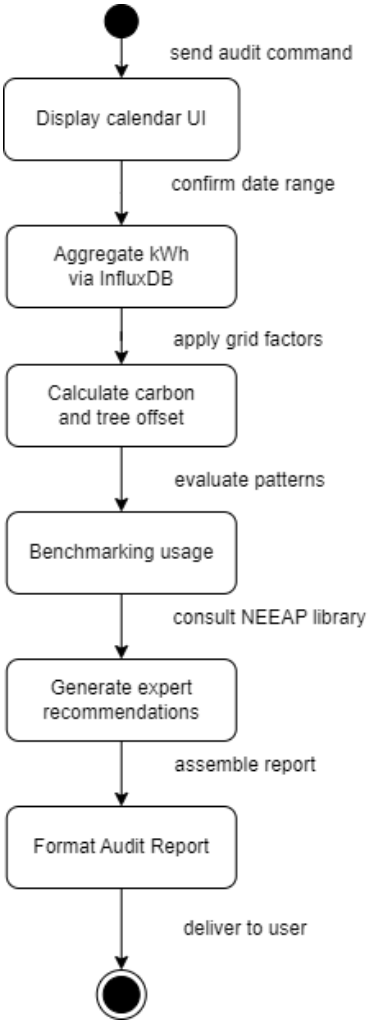


Figure 4.11 State transition diagram for conduct smart energy and carbon audit

Figure 4.11 depicts the logical path of the system when creating a sustainability report. After the user enters the audit command and confirms a date range, the system goes through numerous data-processing steps, including collecting consumption from InfluxDB, applying grid emission factors for carbon calculations, and benchmarking usage trends. The procedure is completed by referring to the NEEAP library providing specific expert suggestions and structuring the results into a full report for delivery to the user.

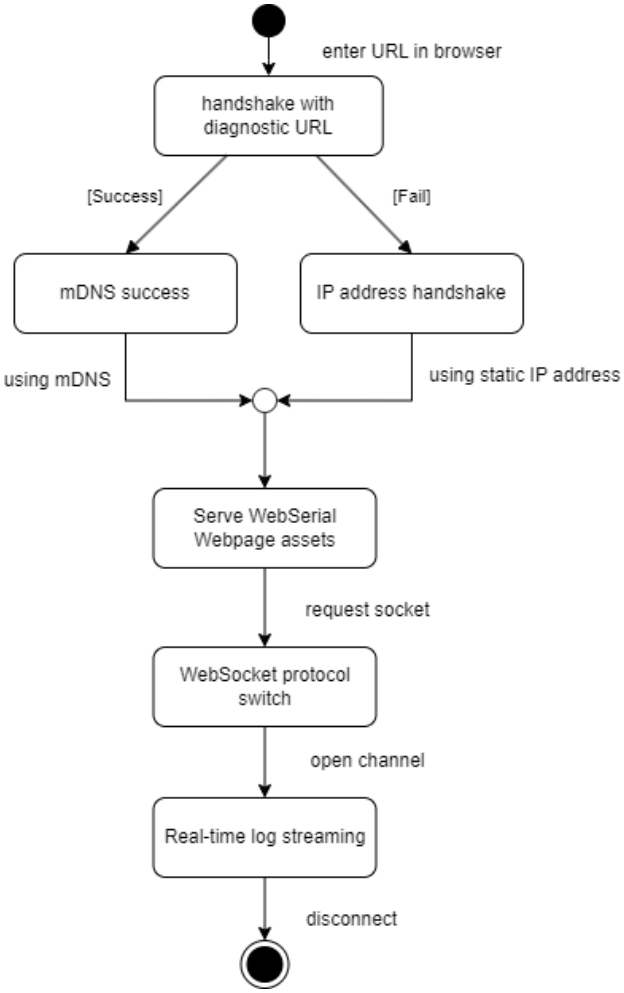


Figure 4.12 State transition diagram for access wireless headless diagnostics

Figure 4.12 depicts the processes of establishing a wireless maintenance link between an administrator and hardware. It starts with a diagnostic URL handshake, then chooses between automatic mDNS resolution or a manual IP handshake if mDNS fails. Once a connection route is established, the system provides the required webpage assets and switches from a conventional HTTP request to a persistent WebSocket channel. This allows for a continuous stream of real-time logs and sensor data until the session is terminated.

4.4 Hardware Design

4.4.1 Schematic Circuit Design

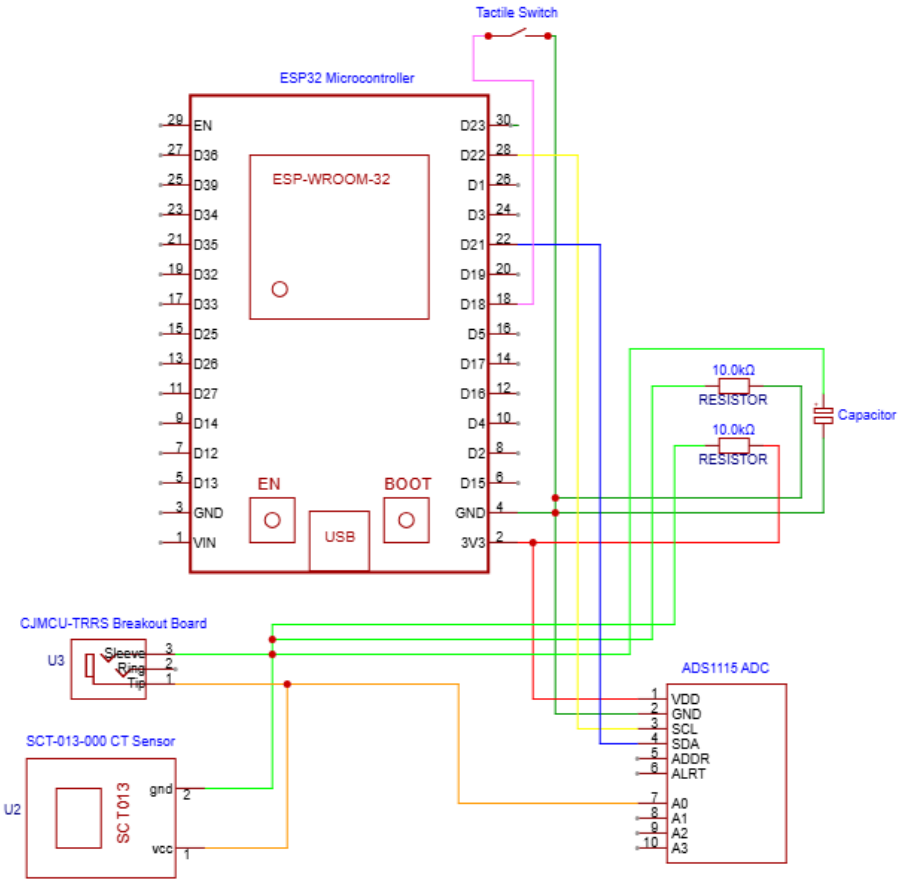


Figure 4.13 Hardware schematic diagram of the project

The ESP32 Microcontroller is crucial for the design as illustrated in Figure 4.13. All other components are linked to its power and GPIO pins. The 3V3 (3.3V) and GND pins on the ESP32 provide power to the entire circuit. The CJMCU-TRRS Breakout Board allows the SCT-013-000 CT Sensor to communicate with the circuit. The sensor's two signal wires are connected to ADS1115 ADC analog input channel A0.

The DC Bias Circuit is an important part of the design since it sends the analog signal from the CT sensor to the ADC. This circuit consists of two 10.0kΩ resistors and a capacitor. The resistors function as a voltage divider to generate a stable reference value of around 1.65V (half of the 3.3V supply). The capacitor serves to filter and stabilize the reference voltage. This bias voltage is then applied to one of the CT sensor's signal lines, thus converting the entire AC waveform to the positive voltage range and allowing the ADC to read it appropriately.

The I2C protocol handles communication between the ESP32 and the high-precision ADS1115 ADC. This requires two connections, the ADC Serial Clock Line (SCL) pin is linked to GPIO pin D22 on ESP32, and the ADC Serial Data Line (SDA) pin is connected to GPIO pin D21 on ESP32. This two-wire connection enables the ESP32 to consistently request and receive high-resolution digital readings from the ADC.

Finally, a tactile switch is included for manual testing and interaction. One leg of the switch is attached to GPIO pin D18 on the ESP32 that is set up as a digital input with an internal pull-up resistor. The other leg is connected to GND. When the button is pressed, it pushes the GPIO pin to ground to allow the firmware to detect the press and initiate an event, such as simulating an anomaly notification. This whole circuit design ensures that the analog signal is precisely and consistently translated into a digital format that the ESP32 can process and send to the cloud.

ESP32 Pinout

The pinout diagram in Figure 4.14 illustrates the function of each of the 30 pins on the development board. These pins can be classified as power pins (3.3v, GND), GPIO pins, and pins with unique hardware functionality like as I2C or SPI communication [21].

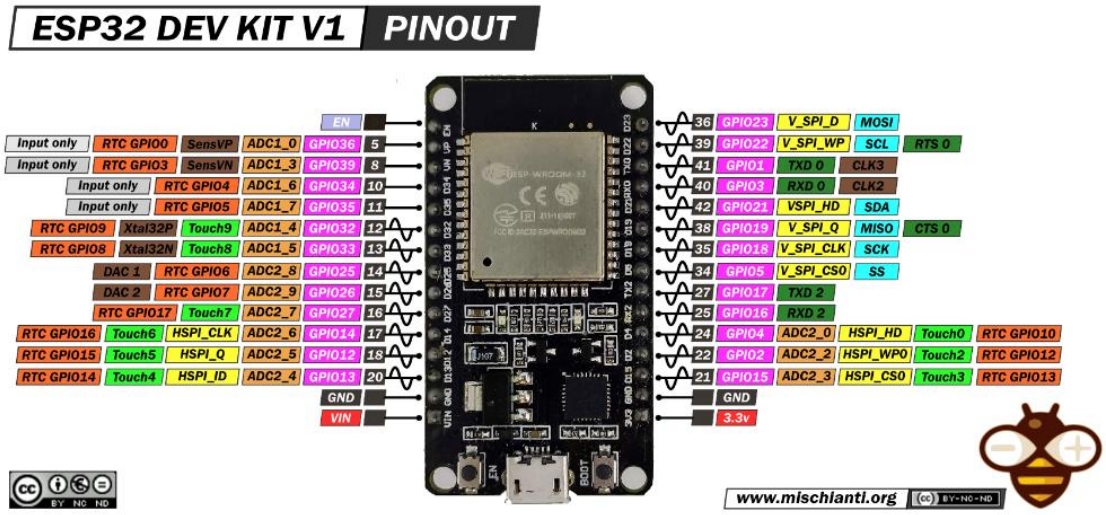


Figure 4.14 ESP32 pinout [21]

The DoIt ESP32 DevKit V1 is a versatile development board designed to expose the rich feature set of the underlying ESP32-WROOM-32 module. The board's pinout is comprehensive, providing access to a wide range of functions that make it suitable for complex IoT applications. The various types of pins can be broadly categorized [21]:

1. General-Purpose Input/Output (GPIO): The board has several GPIO pins that are quite flexible. These pins are multiplexed to provide a range of applications beyond than simple digital input and output, such as Pulse Width Modulation (PWM) for motor control and conventional communication protocols

2. Communication Interfaces: For connection to peripheral devices, the board provides specific pins for the commonly used protocols. I2C is a two-wire interface (SDA/SCL) commonly used to connect sensors and other modules, like the ADS1115 ADC utilized in this project. SPI is a faster protocol commonly used for displays and SD card modules. Standard Transmit/Receive TX/RX pins are accessible for serial communication, Universal Asynchronous Receiver-Transmitter (UART), which is commonly used for troubleshooting.

3. Analog and Touch Inputs: A few GPIO pins can also act as ADC channels that are required for reading analog sensors. The ESP32 also has many Capacitive Touch Pins which can detect human contact without the need for a mechanical button.

4. Power Pins: The board includes an adjustable power scheme to support external components. It includes 3.3V and 5V output pins as well as several GND (ground) pins. The VIN pin can also serve as an alternate power input to allow the board to be powered by a source other than the USB port.

5. Control and Programming: The EN (Enable) pin functions as a hardware reset for the microcontroller. The board built-in USB-to-UART Bridge allows the ESP32 to be easily programmed and monitored from a computer via a regular USB cable.

4.5 Software Design

4.5.1 Firmware Logic (ESP32)

I. Asynchronous Execution

The ESP32 firmware logic is designed with an asynchronous, event-driven architecture to ensure high-resolution data gathering while maintaining network responsiveness. The software design is built around a non-blocking execution approach that makes use of the millis() timer function. This eliminates traditional blocking delays and allows the microcontroller to preserve a predictable 20ms sample window for the AC waveform while also managing background duties like Wi-Fi heartbeats, MQTT keep-alives and the WebSerial diagnostic stream.

II. Energy Odometer

Upon system initialisation, the firmware performs a bootstrap step. During this stage, the ESP32 configures the I2C bus for the 16-bit ADS1115 ADC and connects to the local network using mDNS. The Energy Odometer's restoration is a vital component of the startup logic. The firmware uses the Preferences.h library to access the last recorded totalKWh value from the NVS. This ensures that cumulative energy usage and financial tracking remain permanent even after power reboots or hardware resets

III. Algorithmic Billing

The data processing logic is separated into two parts, a high-speed sampling loop and a low-speed calculation interval. Every 5 seconds, the system executes a burst read of 1000 analog samples from the ADS1115 to properly capture the current waveform's periodic peaks. The raw results are then processed using the RMS technique. After determining the RMS current, the firmware uses a piece-wise function to represent the domestic electricity tariff. This method does not simply multiply values, it estimates expenses across several tiers and adds mandatory charges.

IV. Diagnostic Thread

Finally, the firmware controls the communication state machine. To maintain compatibility with the Telegraf agent in the backend, processed data is encoded as a JSON payload. To avoid excessive wear on the ESP32 flash memory, the system uses a Coalesced Write Policy in which the updated energy odometer is only committed to the NVS every 15 seconds.

4.5.2 Database Schema Design (InfluxDB)

The system uses InfluxDB v2.7, a high-performance TSDB, to manage the high-velocity data. The format is optimised for time-stamped monitoring. The data is organised into a hierarchical structure that includes a primary key, data attributes and derived attributes.

Table 4.2 Database schema design

Primary Key					Data Attributes (Field Keys)				Derived Attributes (Derived ML Features)						
Measurement	Sort Key	Partition Keys (Tag Keys)			Attr. 1	Attr. 2	Attr. 3	Attr. 4	Attr. 1	Attr. 2	Attr. 3	Attr. 4	Attr. 5	Attr. 6	Attr. 7
mqtt_consumer	_time	host	topic	device_id	current	power	totalKWh	totalCost	hour_of_day	day_of_week	is_weekend	minute_of_day	month	time_category	Malaysia_Date

Primary Key and Indexing Logic

As shown in Table 4.2, the system uses a composite Primary Key to ensure that each data point is unique and metrologically traceable.

- **Measurement (mqtt_consumer):** This serves as the primary container for all energy-related data.
- **Sort Key (_time):** InfluxDB uses the timestamp as its primary index. Every data is automatically classified based on its 5-second sample interval to ensure precise real-time trend analysis.
- **Partition Keys (host, topic, device_id):** These represent the Tag Set. The database engine fully indexes tags, making it possible to filter and group them instantly across several monitoring nodes or deployment locations.

Data Attributes (Field Keys)

The Data Attributes are raw fluctuating measurements processed at the edge by the ESP32. These include real-time current (A), instantaneous power (W) and the cumulative total KWh. Additionally, the system store totalCost (RM) which is determined at the edge using the tiered tariff. These fields are not indexed to allow for faster write operations without the burden of index maintenance.

Derived Attributes (Temporal Feature Engineering)

To enable the ML engine, the database schema contains a Derived Attributes layer created with InfluxDB's Flux scripting language. Temporal Feature Engineering converts raw timestamps into contextual data points. By changing the database location to Asia/Kuala_Lumpur, the system will automatically extract local attributes like hour_of_day, day_of_week, and minute_of_day.

These features allow the RF model to identify cyclical human behaviour patterns. For example, distinguishing between a high-load evening and a low-load night. The derived attributes also include boolean flags like is_weekend which allow for changes in energy use on non-working days. This schema design ensures that the database functions as an active pre-processing layer, providing the high-resolution, context-rich data needed for intelligent anomaly identification and accurate environmental auditing.

Bucket Policy and Retention

All data is stored in the home_energy bucket. The retention policy is set to infinite to ensure an adequate historical dataset for the ML model's weekly retraining. This supplies the system with the long-term data depth it requires to detect metrological deviations over months or years.

4.5.3 Machine Learning Model Architecture

The intelligence layer of the system is intended to provide proactive energy management by detecting deviations from regular home behaviour. To do this, the system goes beyond rigid static limits and uses a data-driven method to construct a dynamic expected power baseline at any given time.

Random Forest Regressor

The intelligence engine is built around a RFR. This approach was chosen because it acts as an ensemble learning method, it builds numerous decision trees and aggregates their outputs that greatly lowering the risk of overfitting, which is a major problem in energy data when outliers are frequent. Second, RF is effective at detecting nonlinear correlations between time-based data and power use. Third, its computational efficiency enables rapid inference within the containerised backend to ensure that notifications are sent to users via Telegram with minimum latency.

Temporal Feature Engineering

The capability of the model to recognise behavioural patterns is based on Temporal Feature Engineering. Raw timestamps are computationally broken down into contextual variables that represent human activity cycles:

- **hour_of_day & minute_of_day:** Captures the daily cycle. For example, peak usage during dinner hours vs. low usage at night.
- **day_of_week:** Differentiate between weekday work habits and higher occupancy during weekends.
- **is_weekend:** A boolean flag that allows the model to take into account for significantly different energy signatures on non-work days.

Model Hyper-parameters

To achieve the ideal balance between accuracy and generalisation, the RF model is configured with specified hyper-parameters as shown in Table 4.3.

Table 4.3 Random Forest regressor and intelligence logic parameters

Parameter	Value	Rationale
n_estimators	100	The total number of decision trees in the ensemble. This value strikes a balance between computational overhead and prediction accuracy in a microservice setting.
random_state	42	This ensures the model reproducibility. Using a specific number ensures consistent outcomes during weekly automated retraining cycles.
max_depth	Default (None)	This enables the model to collect high-resolution temporal features. The algorithm ensemble nature is used to avoid overfitting while mapping complex minute-to-minute patterns.
Retraining Interval	7 Days	Every 604,800 seconds, a complete model update takes place. This ensures that the AI adjusts to the entire weekly cycle of human behaviour.
Alert Sensitivity (σ)	15.0	A conservative standard deviation scaling factor. Combined with a 50% buffer, it reduces false positives caused by slight appliance variations.
Minimum Power Floor	200W	An inference gate that omits low-power noise. This prevents the AI from triggering alerts for insignificant items such as LED bulbs or phone chargers.

Inference Logic and Dynamic Thresholding

The inference engine monitors the MQTT telemetry stream in real time. The model predicts Expected Power ($\hat{y}$) for each incoming reading. To assess whether a reading is abnormal, the system computes a dynamic threshold using Standard Deviation (σ) scaling.

The system uses Standard Deviation (σ) scaling to calculate a dynamic threshold. Although a 2σ threshold for a 95% confidence interval is widely used in basic statistical theory, initial testing in a domestic setting showed that small electrical noise frequently caused false positives. As a result, a more reliable and cautious detection mechanism was put into place.

The detection logic follows the formula:

$$Anomaly = ActualPower > Confidence\ Limit\ (\hat{y} + 15\sigma + 0.5\hat{y})$$

The solution uses a 15σ scaling factor and an extra 50% percentage buffer ($0.5\hat{y}$) to filter out typical home fluctuations and only alerts the user via Telegram when a notable, high-load behavioural aberration takes place.

Automated Retraining Lifecycle

To maintain long-term accuracy as the user's habits change or times change, the system uses an automated retrain_model_auto() lifecycle. Every seven days, the engine operates a historical sync, retrieving the prior week's high-resolution data from InfluxDB. This new dataset is used to retrain and evaluate the model. If the new model shows improved accuracy, it is automatically hot-swapped into the production inference engine. This self-adaptive loop keeps the system intelligent and independent without requiring manual calibration.

4.5.4 Graphical User Interface (GUI) Design

The system interface employs a dual-layer method to maximise data accessibility, integrating high-resolution online visualisation with a mobile-based, bidirectional communication channel.

Grafana Dashboard Layout

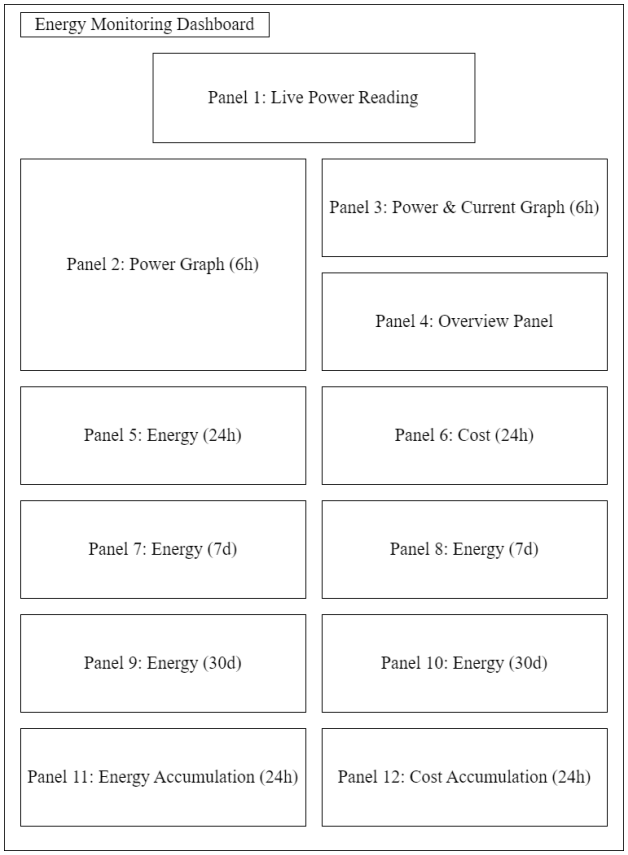


Figure 4.15 GUI design of Grafana dashboard of the project

Figure 4.15 shows the Grafana dashboard as the main analytical interface. The layout is divided into functional layers according to temporal specificity. The top tier includes a high-visibility gauge for Live Power Readings that provides quick feedback. The middle tier includes time-series graphs for Power and Current (6h) that show short-term changes. The lower tiers concentrate on cumulative statistics, such as Energy and Cost Accumulation (24h/7d/30d). This tiered design assures that the system is traceable and it also acts as the source engine for the Telegram Bot's automatic image rendering service.

Telegram Bot Interface

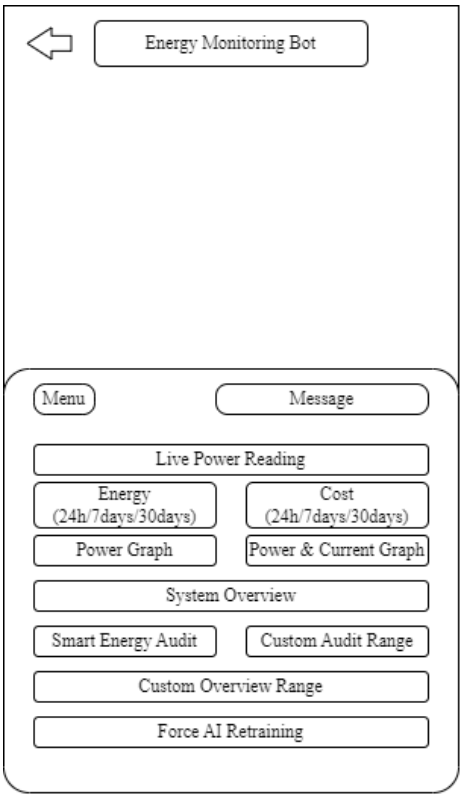


Figure 4.16 GUI design of Telegram Bot of the project

Figure 4.15 shows the Telegram Bot which serves as a mobile command center. It makes use of a Custom Reply Keyboard to make Grafana snapshot retrieval as simple as clicking one button. Aside from passive viewing, the GUI enables complicated interactions:

- Smart Energy Audit: Uses the audit engine to determine carbon footprints and environmental offsets.
- Interactive Calendar UI: A unique inline date selector that allows users to select certain date ranges for historical analysis without having to manually enter them.
- Proactive Alerting: A dedicated notification state in which the AI engine sends real-time “High Power Alerts” when it finds behavioural anomalies.
- Administrative Control: Includes a “Force AI Retraining” option for manually initiating the model update lifecycle.

4.6 Concluding Remark

This chapter provided a detailed technical design of the system, connecting high-level architecture to specific implementation logic. The complex connections between the edge firmware, TSDB and machine learning engine are precisely characterised using detailed UML modelling and schematic design. The software design which includes temporal feature engineering and the interactive Telegram UI.

Chapter 5

System Implementation

5.1 Hardware Setup

The hardware implementation focused on building a independent sensing node capable of continuous operation.

Component Assembly and Wiring

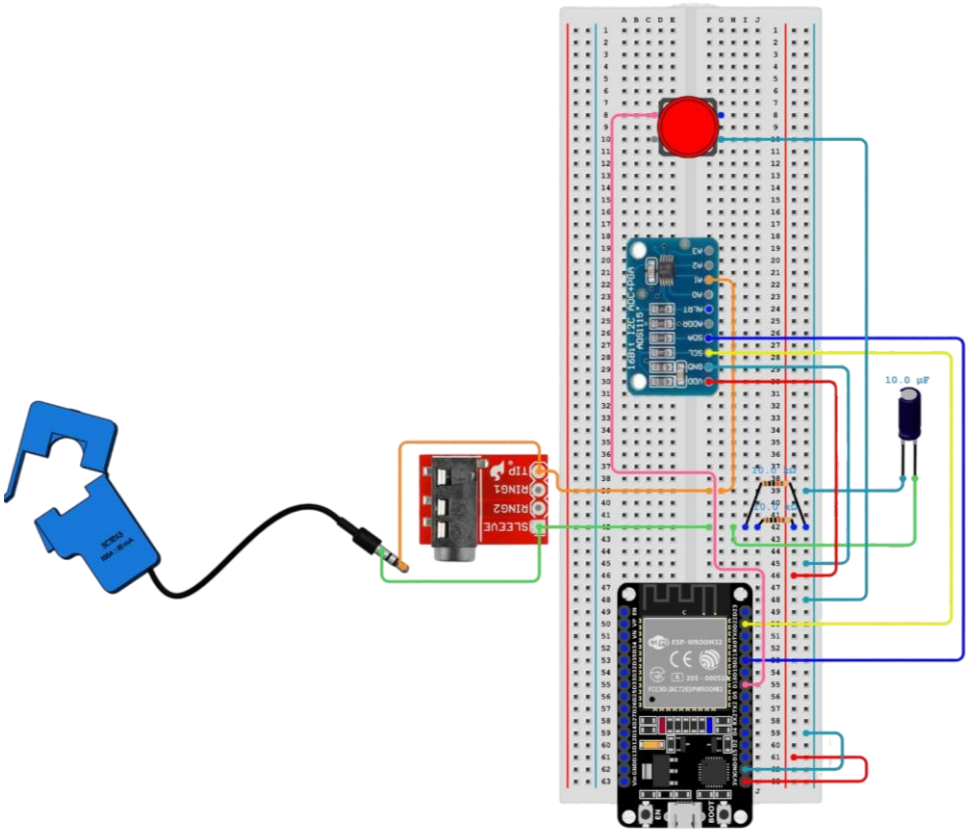


Figure 5.1 Assembled hardware schematic diagram of the project

As shown in the schematic diagram in Figure 5.1 above, the assembly is modular in design with the ESP32 microcontroller at the center. A 3.5mm jack breakout board is used to connect the SCT-013-000 CT sensor to the DC Bias circuit. This circuit uses two 10kΩ resistors to create a stable 1.65V reference to keep the AC signal inside the ADS1115 16-bit ADC readable range. The high-precision ADC communicates with the ESP32 via I2C (GPIO 21 and 22).

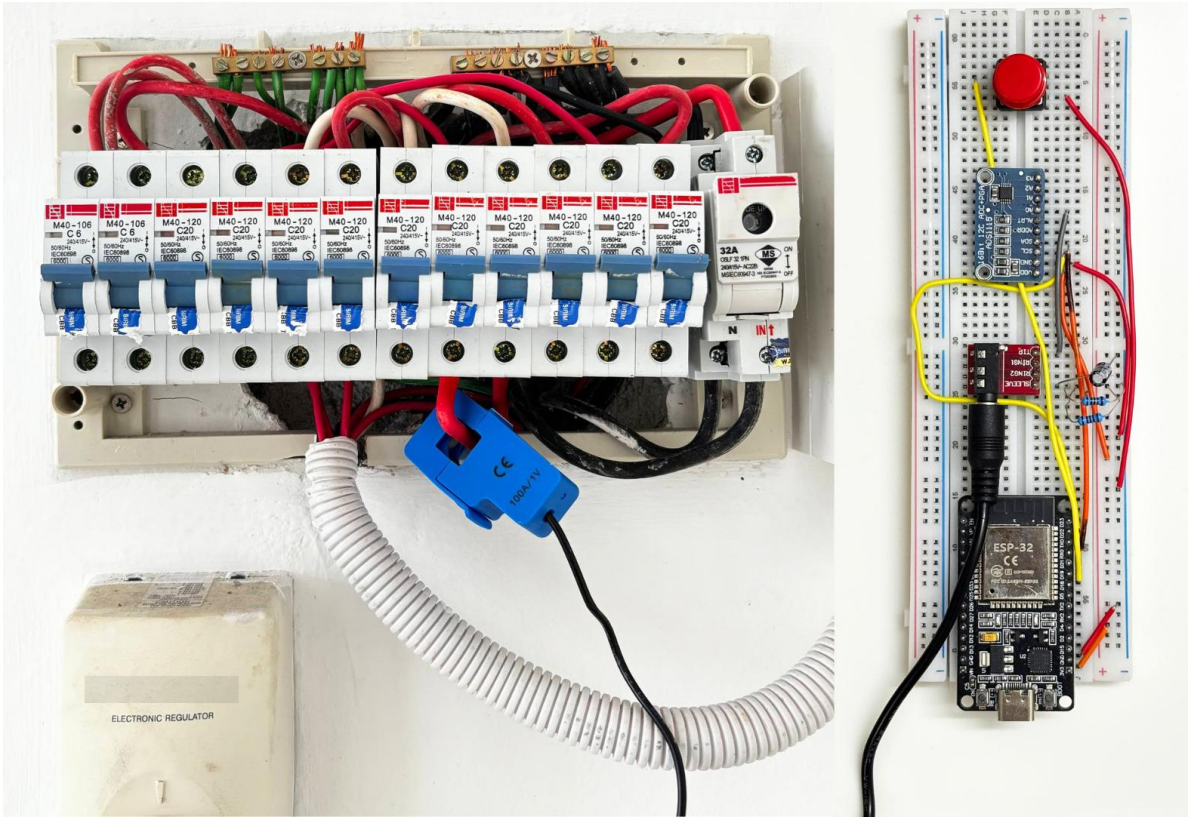


Figure 5.2 Fully connected hardware of the project

Figure 5.2 depicts the system physical deployment, demonstrating its noninvasive nature. The CT clamp is securely linked to the main live wire of the home Distribution Board (DB). Unlike standard meters, this installation needed no electrical downtime. To ensure 24-hour reliability, the system was switched from USB to an independent 5V/3A DC power source. The WebSerial Lite server was launched for administrative purposes. It allows the hardware to be monitored wirelessly without opening the database enclosure.

5.2 Software Setup

I. Edge Firmware Deployment

The Arduino IDE is installed on the owner laptop to support in the development and deployment of the C++ firmware. The installation process includes configuring the ESP32 Board Manager and installing necessary libraries including PubSubClient for MQTT communication, Preferences.h for NVS data persistence and Adafruit_ADS1X15 for high-precision analog readings. The Arduino framework was chosen because of its extensive community support and efficient hardware abstraction. Its emphasis on stability ensures that the edge device can handle high-frequency sampling and network duties concurrently, providing it a crucial and dependable foundation for the project sensing layer.

II. Docker Desktop and Containerized Service Orchestration

Docker Desktop is installed to manage backend microservices using a containerised architecture. The system uses Docker Compose to organise several services within an integrated virtual network, including the Mosquitto MQTT broker, InfluxDB TSDB and Grafana visualisation server. This design streamlines configuration and ensures that the backend is portable and scalable. Container isolation eliminates dependency conflicts and optimises resource utilisation on the host machine. This results in a more efficient and industrial-grade development approach for IoT data management.

III. Telegram API and Python Bot Integration

The Telegram Bot API is set up to allow bidirectional communication between the system and the user. The setup method includes creating a unique secure token with BotFather and constructing a Python-based logic engine with the python-telegram-bot module. This integration enables real-time proactive warnings and on-demand report retrieval. In addition, the installation requires setting an authenticated link between the bot and the Grafana Image-Renderer. This design ensures a safe and encrypted data exchange connection which improves the overall integrity of the user interface and giving an easy-to-use tool for smart energy auditing.

5.3 Setting and Configuration

The following libraries are installed independently in the Python environment to support the system's intelligent backend and Machine Learning operations:

I. python-telegram-bot

- **Usage:** A comprehensive library for interacting with the Telegram Bot API. It is used to process bidirectional user commands, maintain the interactive calendar UI and send proactive AI alerts.

Figure 5.3 Generation of the API token via Telegram BotFather for bidirectional communication

II. scikit-learn

- **Usage:** The RFR is implemented using a Machine Learning library for Python. It includes the tools required for feature engineering and training the model to recognise energy patterns.

III. influxdb-client

- **Usage:** The official Python client for InfluxDB version 2.7. It is used to operate Flux queries that retrieve 7-day historical data for model retraining and aggregate energy values for sustainability auditing.

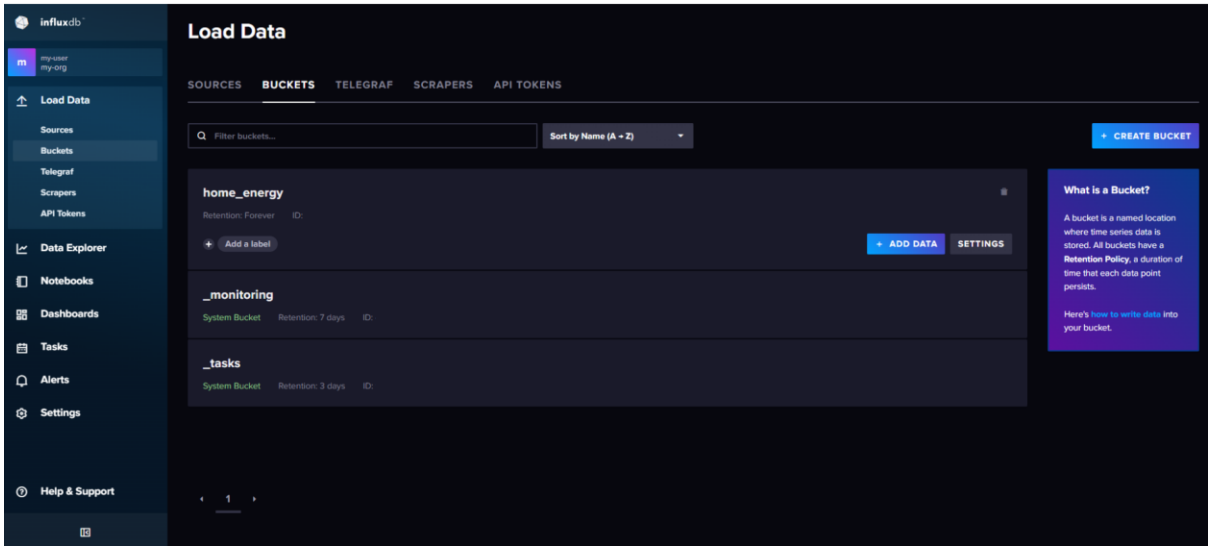


Figure 5.4 Configuration of the time-series data bucket within the InfluxDB management interface

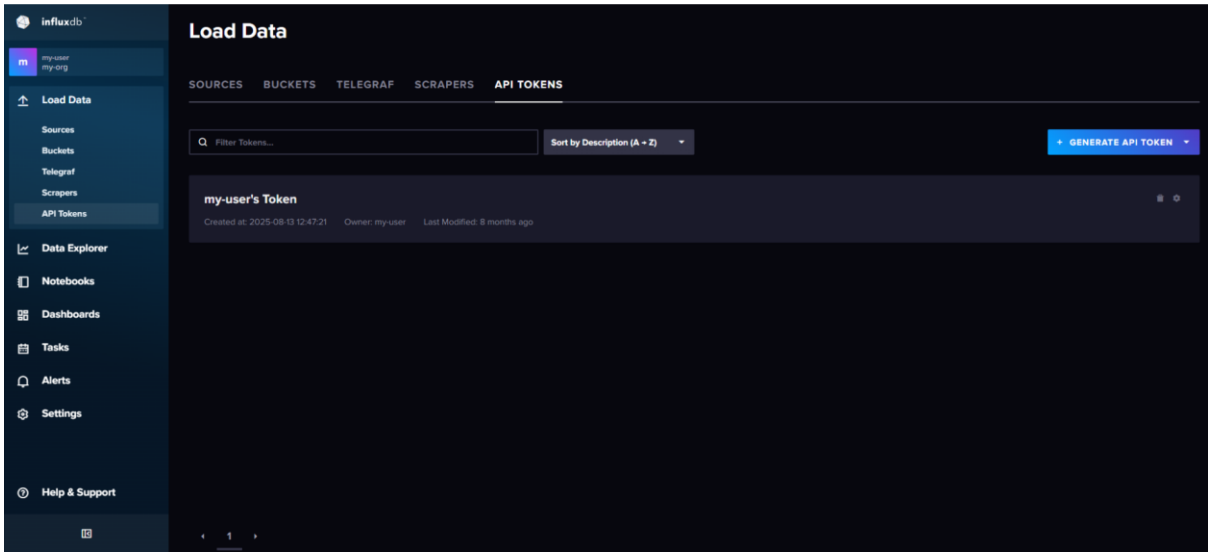


Figure 5.5 Configuration of the access tokens within the InfluxDB management interface

IV. paho-mqtt

- **Usage:** A lightweight client library that allows the Python AI engine to subscribe to a MQTT broker. It enables the system to intercept live telemetry from the ESP32 and perform real-time anomaly inference.

V. joblib

- **Usage:** Python objects are serialised and deserialised using this module. It enables the system to save the trained RF model to a file, ensuring that the intelligence persists between systems restarts.

The following environments and tools are configured within the host laptop to orchestrate the system:

I. Docker and Docker Compose

- **Usage:** It is the main orchestration platform for containerised microservices. It enables Mosquitto, InfluxDB, Telegraf and Grafana to function in an isolated virtual network that ensuring consistent performance and portability.

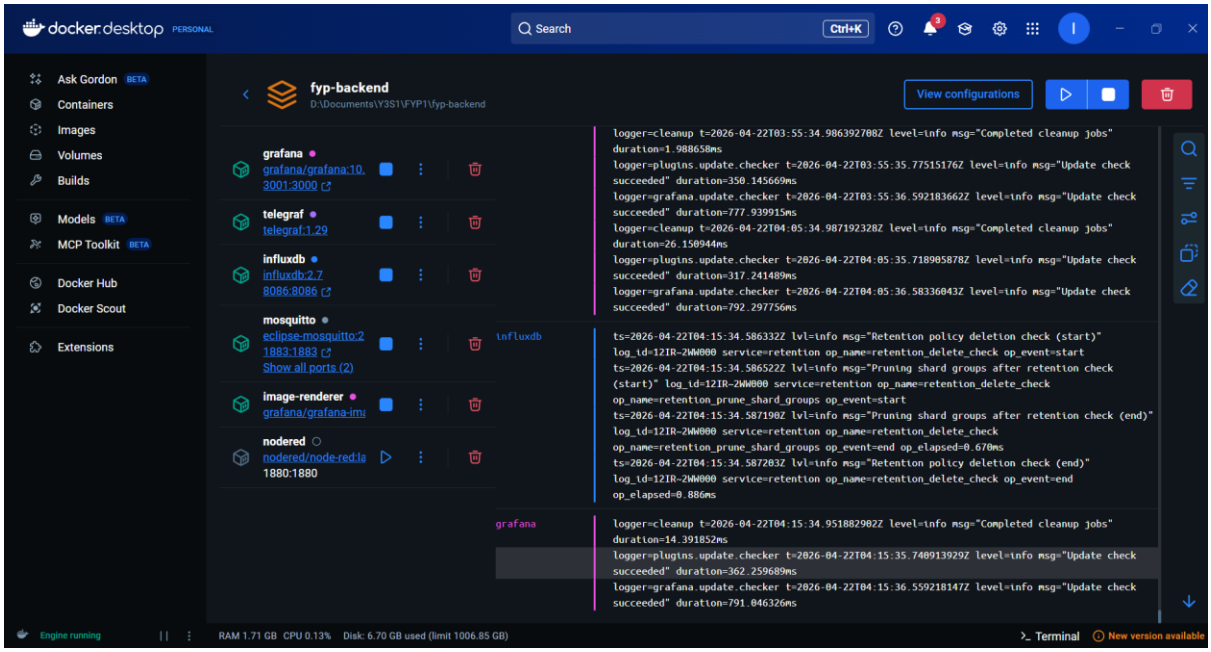


Figure 5.6 Docker Desktop showing the successful orchestration of containerized backend services

II. Grafana Image-Renderer

- **Usage:** A specialised plugin service used to generate high-resolution PNG screenshots of dashboard panels. It enables the Telegram Bot to send graphic energy reports on demand.

III. ESP32 Non-Volatile Storage (NVS)

- **Usage:** Utilises the ESP32 internal flash memory to retain the cumulative "Energy Odometer" (totalKWh). This configuration enables data persistence and allows energy tracking to resume properly after a power disruption.

.

5.4 System Operation

The Home Energy Monitoring and Alert System is designed to run fully autonomously, providing users with real-time data access and intelligent insights via an efficient edge-to-cloud process. The system's operation is divided into the five following stages:

Edge Device Startup and Diagnostic Handshake

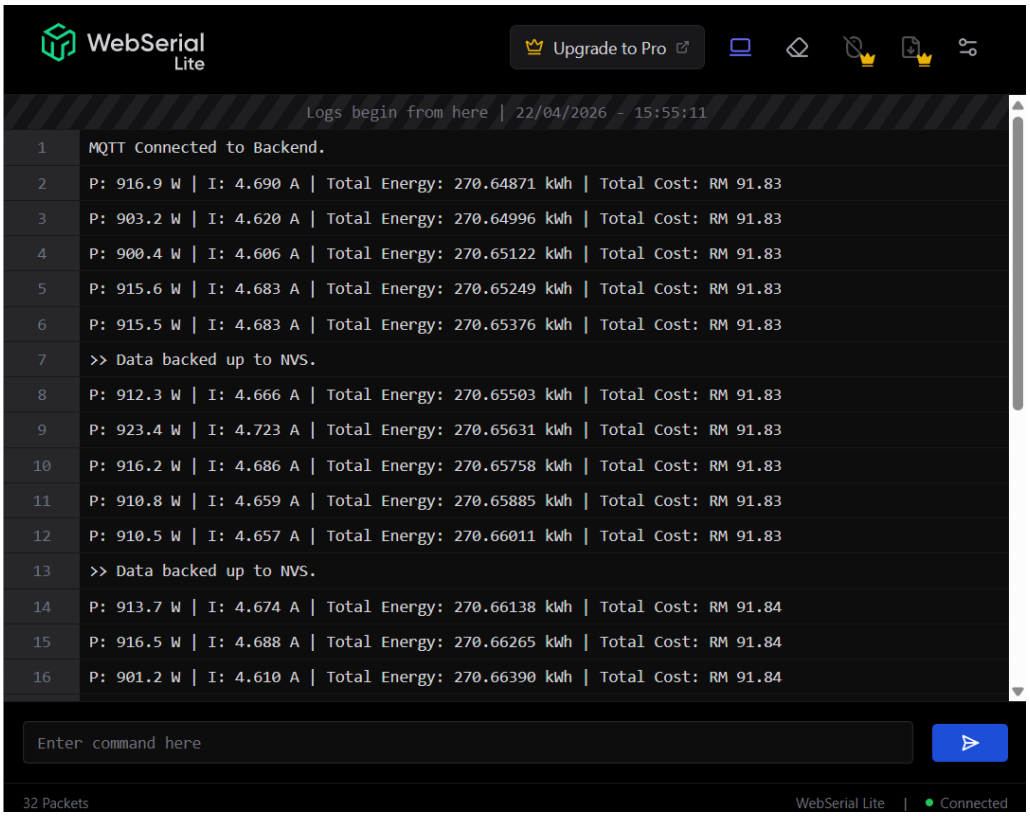


Figure 5.7 Screenshot of the WebSerial Lite terminal with raw data and NVS loading success

The operation begins when the ESP32 hardware is powered on using the standalone 5V/3A power source. The firmware initially establishes I2C communication with the ADS1115 ADC and pulls the most recent known energy value from NVS. The device then broadcasts its presence on the local network via mDNS. The administrator can check the system health by going to http://EnergyMonitor-ESP32.local/webserial using any web browser. This headless diagnostic interface displays raw ADC samples and network signal strength without requiring physical serial cables.

Real-Time Visualization via Web Dashboards

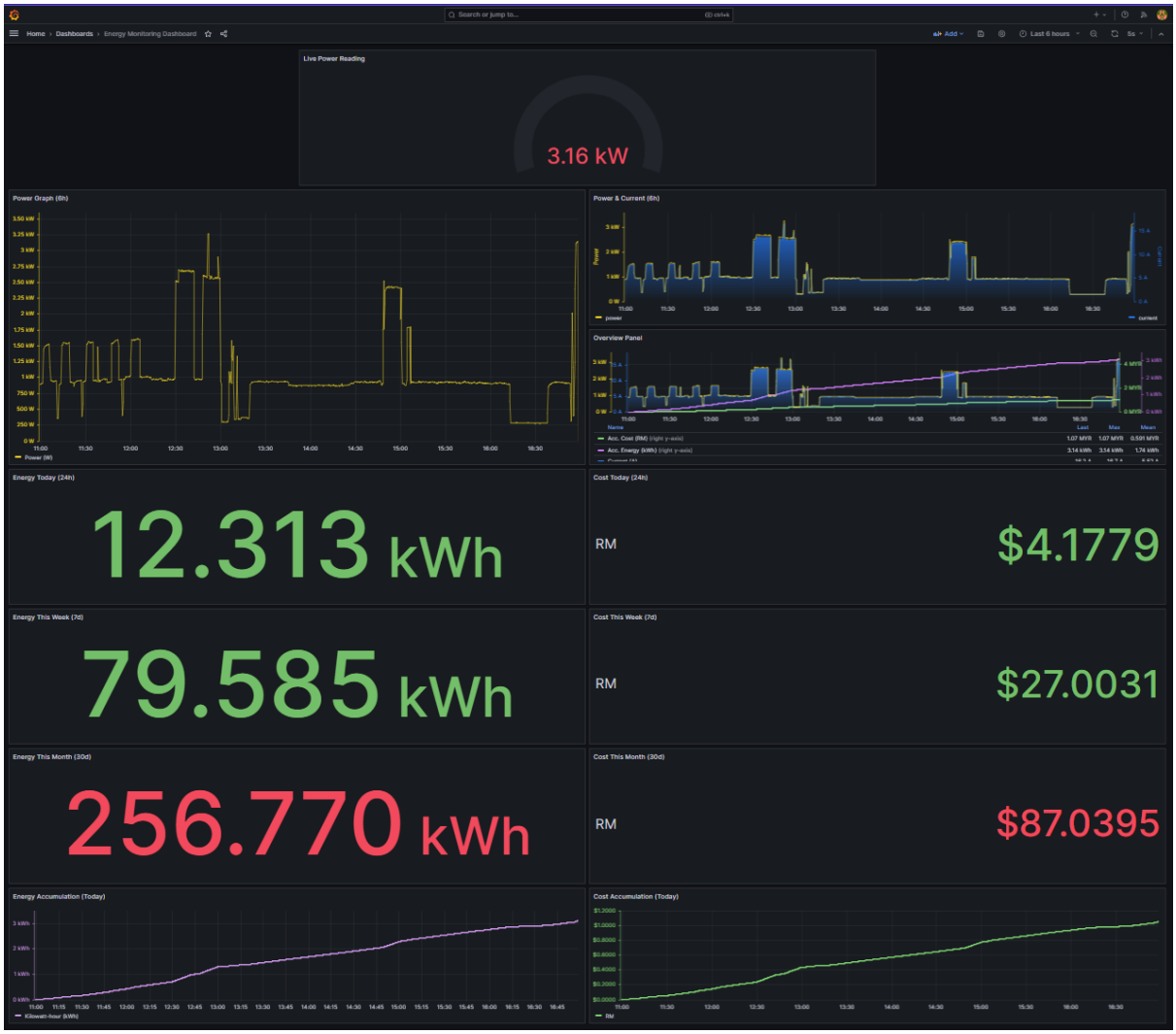


Figure 5.8 Screenshot of the Grafana Dashboard showing the Power and Energy trend

When the system is online, it begins a continuous 5-second sampling loop. The ESP32 calculates TRMS power and cost. This data is published to the MQTT broker and visualised on Grafana. For in-depth study, the Grafana web dashboard offers a full analytical view that includes live power reading, historical trends, cost accumulation and overview panels.

On-Demand Visual Reporting via Telegram Bot

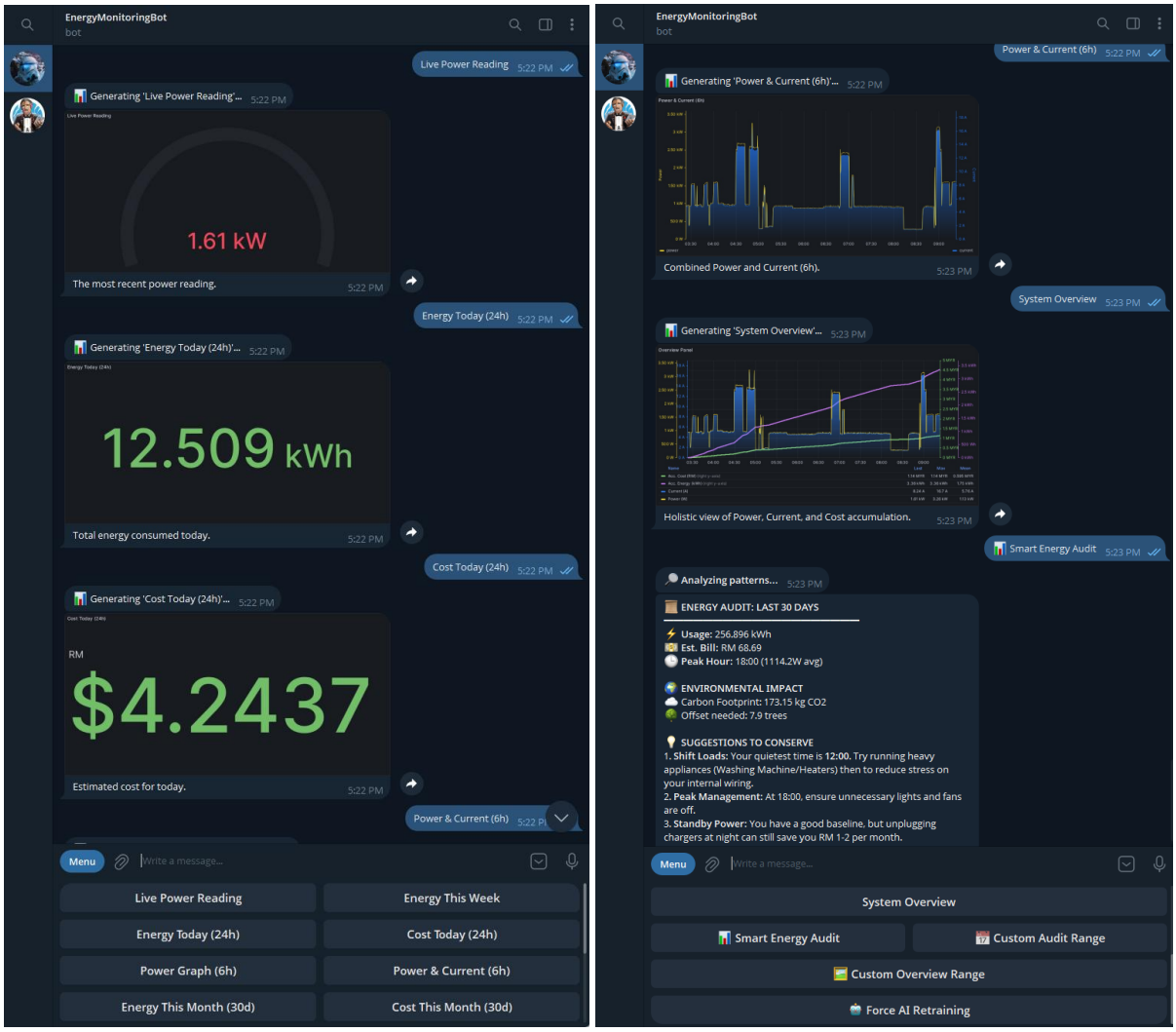


Figure 5.9 Screenshot of the Telegram Bot delivering PNG snapshots and text-based report

The system includes a unique “Pull” feature via the Telegram Bot to allow users to request graphical information at any moment. Using the custom menu keyboard, the user can select buttons like “Live Power Reading”, “Energy (24h/7d/30d)”, “Cost (24h/7d/30d)”, ”Power Graph”, “Power & Current Graph”, ”System Overview” and “Smart Energy Audit”. The Python backend intercepts these commands, directs the Grafana Image-Renderer to generate the required panel as a PNG file and sends it to the user’s chat. This enables the user to visualise high-quality graphic reports without needing to connect into a web browser.

Sustainability Auditing and Custom Range Selection

The Telegram Bot initialises an interactive inline calendar UI when a user chooses either custom range option as illustrated in Figure 5.10 and Figure 5.11 below. This eliminates the need for manual text input and reduces user error by enabling the user to set particular start and end dates and navigate months with a few clicks.

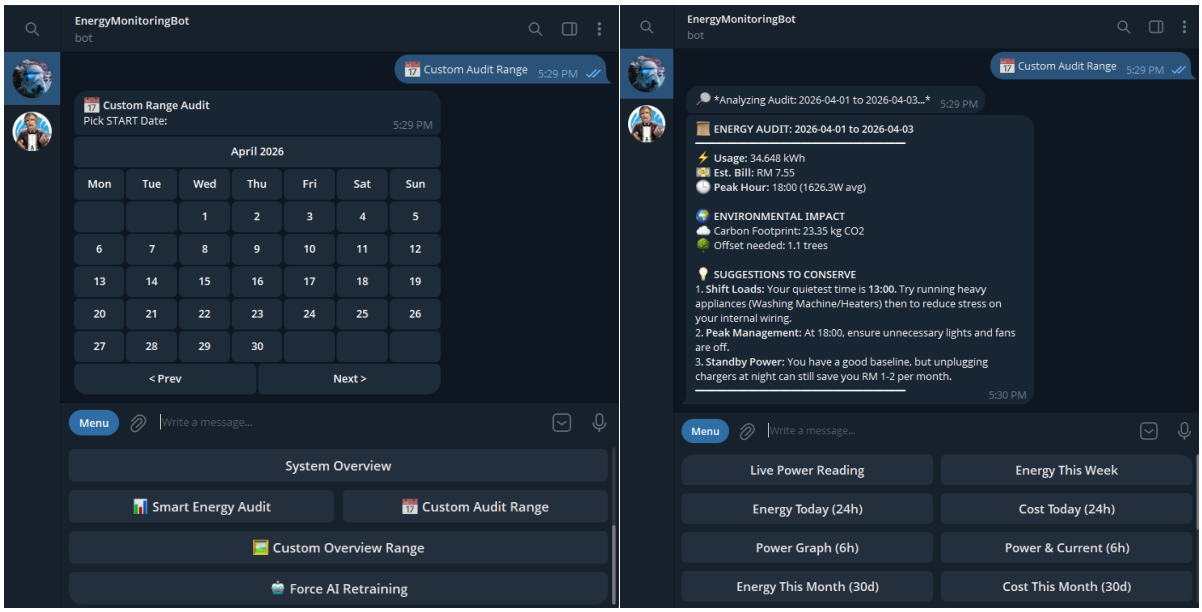


Figure 5.10 Screenshot of the Telegram Bot delivering custom audit range report

After confirming a date range, the system runs an aggregation query on InfluxDB to produce an extensive text-based report. As shown in the Energy Audit screenshot, the system generates a detailed breakdown of total energy usage (kWh) and an expected cost. It also identifies the peak hour of usage for that time period. Furthermore, the report contains a sustainability part that converts kWh to a Carbon Footprint and calculates the tree offset (the number of mature trees needed to absorb that carbon). The interaction ends with AI-generated energy-saving ideas, such as moving heavy loads to off-peak hours.

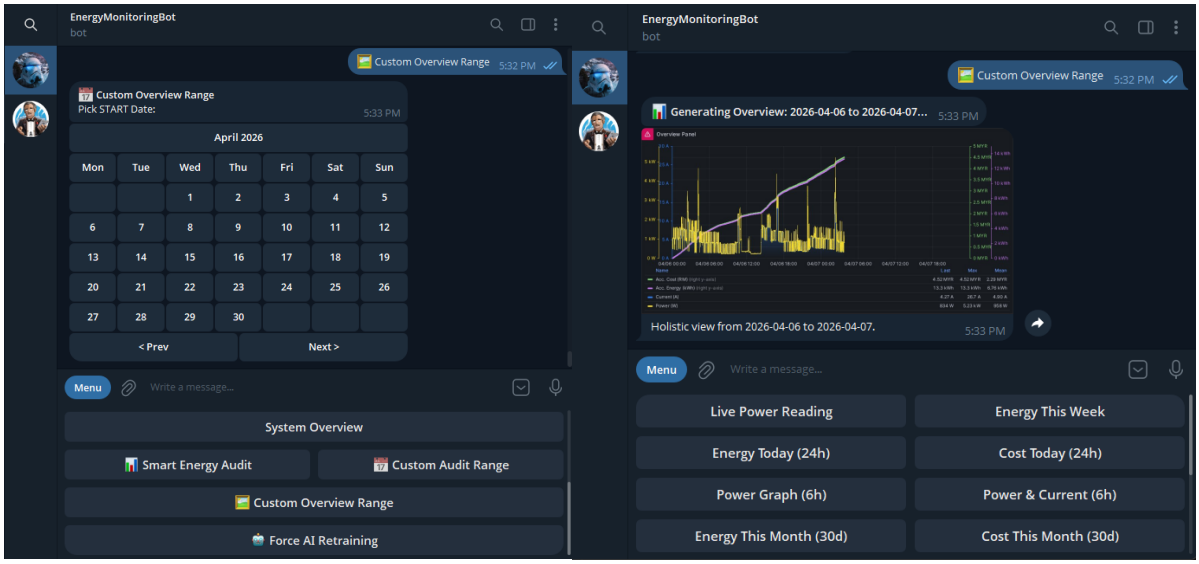


Figure 5.11 Screenshot of the Telegram Bot delivering custom overview range graph

The system uses a similar calendar selection procedure but activates the Grafana Image-Renderer if the user selects the Overview Range. For the chosen dates, the backend creates a high-resolution “Holistic View” PNG image. This visual report overlays several measures, such as real-time current, instantaneous power and accumulation of both energy and cost, as shown in the created overview graph. This enables homeowners to see how their everyday activities connect to the spikes in their electricity bills.

Intelligent Proactive Alerting

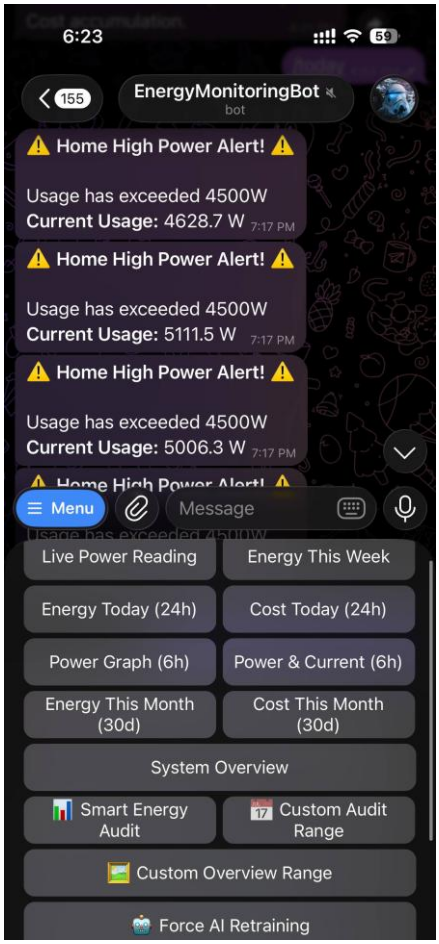


Figure 5.12 Screenshot of a real-time rule-based high power alerts received on a smartphone

Figure 5.12 shows that the system retains a “High Power Alert” mechanism that prioritises safety. A static threshold, currently set at 4500W, initiates this alarm in order to identify high load utilisation that may cause stress on the internal wiring of the home. These messages are sent in real time by the Telegram Bot, to give the user the precise wattage measured at the time the safety limit was exceeded. This acts as a vital barrier to prevent unintentional overload from high-power appliances.

Figure 5.13 Screenshot of a real-time AI anomaly alert message received on a smartphone

Figure 5.13 illustrates how the RF Inference Engine is used by the system to provide context-aware alerts when fixed limitations are exceeded. Unlike the static warning, this notification is triggered when consumption deviates significantly from the user's previous habits for that hour and day. The "Unusual Power" reading in comparison to the "AI Expected" baseline and the "Confidence Limit" (calculated using a 15.0σ scaling factor) are explicitly provided by the bot message. This enables the system to detect anomalies, such a heater left on at midnight, that may remain below the 4500W safety limit but are still unusual for the particular time of day.

5.5 Analysis of Anomaly Determination Logic

The following scenarios in Table 5.1 show how anomalies are determined numerically to show how the system differentiates between regular usage and irregular behaviour. These computations make use of the system inference engine's complex logic that combines statistical volatility with a safety buffer:

$$Confidence\ Limit = \hat{y} + 15\sigma + 0.5\hat{y}$$

where

$\hat{y}$ = AI Expected Baseline

σ = Standard Deviation

Table 5.1 Comparative analysis of anomaly determination scenarios

Scenario	**Time Context**	**AI Expected ($\hat{y}$)**	**Volatility (σ)**	**Calculated Confidence Limit**	**Actual Power (P)**	**Alert Status**
A: Background Load	03:00 (Mon)	150.0 W	10.0 W	375.0 W	165.2 W	Normal
B: Behavioural Anomaly	2:30 (Tue)	386.6 W	45.2 W	1248.1 W	2610.1 W	AI Anomaly Detected
C: Standard Afternoon	14:00 (Wed)	1200.0 W	80.0 W	3000.0 W	1450.5 W	Normal
D: Safety Breach	19:00 (Fri)	1800.0 W	120.0 W	4500.0 W	5111.5 W	High Power Alert
E: Normal Spike	12:00 (Sat)	1000.0 W	150.0 W	3750.0 W	3200.0 W	Normal

Technical Analysis of Implementation Scenarios based on Table 5.1

- **Scenario A:** During the early morning hours (03:00 AM), the system has its lowest baseline and volatility. As indicated in the table, the real power of 165.2 W is slightly more than the AI projected 150 W, although the difference is small. Because the amount is significantly below the Calculated Confidence Limit of 375.0 W, the system correctly recognises it as a regular background load and retains the "Normal" status without issuing an alarm.

- **Scenario B:** This scenario demonstrates the fundamental intelligence of the system. The time frame (02:30 AM) is usually a low-consumption period, even if the observed usage of 2610.1 W is much below the 4,500 W safety limit. The algorithm recognises this as possible waste and sends out a preventive Telegram alert because the real power greatly exceeds the Calculated Confidence Limit.

- **Scenario C:** This scenario shows the system's capacity to respond to increased baseline usage during active hours. At 14:00 PM, the AI anticipates a substantially larger load of 1,200 W due to learnt daytime behaviours. To account for typical home fluctuations, the safe zone is expanded to 3,000.0 W using 15.0 σ scaling and a 50% buffer, but consumption remains at 1,450.5 W. This ensures that normal tasks do not trigger false-positive notifications, even when power levels exceed the midnight baseline.

- **Scenario D:** In this case, the consumption reaches both the absolute safety floor of 4500 W and the dynamic AI barrier. This triggers the rule-based "Home High Power Alert". This dual-check makes sure that the system safeguards the home wiring from potential overload even if the AI anticipates high usage.

- **Scenario E:** This illustrates how resilient the 50% buffer and 15.0 σ scaling are. Energy usage is naturally unpredictable during times of high energy consumption like Saturday at noon, hence increase the value of σ. Even though 3200 W is a high power rating, it remains inside the AI safe zone for that specific period, thus preventing the user from receiving unnecessary false-positive alerts.

5.6 Implementation Issues and Challenges

Several technological challenges emerged throughout the deployment of the home energy management system.

I. System Crashes caused by Empty Dataset Rendering

One of the key issues in the Interaction Layer is the Grafana Image-Renderer failure to execute requests for periods with missing data. In early tests, if a user requested a visual report while the hardware was offline or the database was empty, the rendering service would enter an infinite wait state or display a timeout error causing the Python bot script to fail. This was fixed by introducing robust Flux queries with union() and fallback table functions. The system is set up to detect empty result sets and automatically inject a dummy "System Offline" data point.

II. Command Parsing Failures with Special Characters in Regex

The implementation of the interactive Telegram interface faced a significant software error relating to command interpretation. The custom button keyboard had detailed labels like "Energy Today (24h)" and "Cost This Month (30d)". However, the Python backend initially used normal Regular Expressions (Regex) which failed to recognise these buttons since brackets are recognised as unique metacharacters for capture groups. This caused the system to ignore legitimate user requests. To address this issue, the command handler was refactored to use the re.escape function. By escaping the labels before processing, the bot was able to do literal string matching, assuring that all visual reporting features worked consistently regardless of the user interface's naming complexity.

III. Balancing Data Persistence with Flash Memory Endurance

To ensure the accuracy of the Energy Odometer (totalKWh), the ESP32's NVS needed to be updated on a regular basis so that data would withstand unexpected power outages. However, the internal flash memory of the ESP32 has a limited amount of write cycles. Saving data at every 5-second sample interval would cause premature device failure. To address this issue, a Coalesced Write Policy was implemented. The firmware was designed to run local calculations every 5 seconds while only committing cumulative values to the hardware NVS every 15 seconds. This technique ensures great data security, limiting potential data loss to a maximum of 10 seconds, while significantly increasing the operating lifespan of the edge device's flash memory.

5.7 Concluding Remark

This chapter covered the system's complete development, from technical concept to fully functional independent monitoring tool. The physical integration of high-precision sensing hardware and the deployment of a containerised microservice backend have resulted in a dependable and secure end-to-end data flow. The system has acquired the technical stability required for autonomous home operations after successfully integrating the machine learning inference engine and overcoming significant software stability challenges.

Chapter 6

System Evaluation and Discussion

6.1 System Testing and Performance Metrics

A number of performance data were collected to establish the system's operational readiness as an independent monitoring tool. The evaluation focuses on system resilience using component failure simulation, sensing precision using statistical error analysis and the responsiveness of the bidirectional communication layer.

6.1.1 System Reliability

A series of component removal tests have been conducted to evaluate the standalone architecture's robustness. This assessment assesses how the system protects data integrity or provides protections during hardware or network outages.

Table 6.1 System continuity overview

Component Removed	System Continuity	Technical Impact
ADS1115 ADC	System active, data null	I2C communication fails; system publishes 0W to InfluxDB to prevent dashboard rendering errors.
SCT-013 Sensor	System active, data zero	The ADC continues to read the stable 1.65V DC bias center point; actual power is recorded as 0.0W.
DC Bias Circuit	System active, data erratic	Removing resistors causes the signal to lose its 1.65V offset. AC peaks are clipped by the ADC, resulting in highly inaccurate readings.
Tactile Button	System active	Manual anomaly simulation is disabled; however, the autonomous RF detection remains fully operational.

Internet Connectivity	System active, Cloud down	Local mDNS and WebSerial diagnostics stop. ESP32 enters a Wi-Fi retry loop while continuing to update local NVS data.
ESP32 Power Supply	Full System Down	The microcontroller and all peripheral modules lose power; data acquisition and transmission terminate immediately.

Table 6.1 summarises how the system architecture modular and decoupled design provides high resilience. The system maintains dashboard stability by adding software-level fail-safes, such as broadcasting null values during ADC failures. Furthermore, the ESP32 capacity to sustain local energy tracking and NVS updates during internet outages assures that the energy odometer retains its metrological integrity. Finally, these findings demonstrate that the system is reliable enough for continuous residential usage as only the loss of ESP32 main power supply results in full system failure.

6.1.2 Measurement Accuracy

The system was compared to the manufacturer's rated power specifications of numerous household appliances to validate the sensing precision of the 16-bit ADS1115 ADC and the SCT-013-000 sensor.

The accuracy is assessed using two standard statistical metrics: Absolute Percentage Error (APE) for individual trials and Mean Absolute Percentage Error (MAPE) for overall system performance. The APE and MAPE are calculated using the following formula:

$$APE(\%) = \frac{|Rated\ Power - \ Measured\ Power|}{Rated\ Power} \times 100 \quad \textbf{(1)}$$

$$MAPE(\%) = \frac{Total\ APE(\%)}{N} \times 100 \quad \textbf{(2)}$$

where

N = Total number of observations

Figures 6.1, 6.2, 6.3 and 6.4 show the measured power of standing fan, hairdryer, induction cooker and electric kettle respectively.

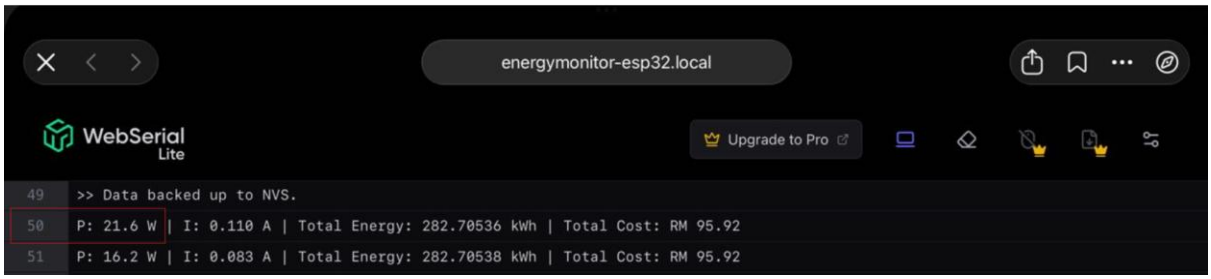


Figure 6.1 Screenshot of standing fan measured power

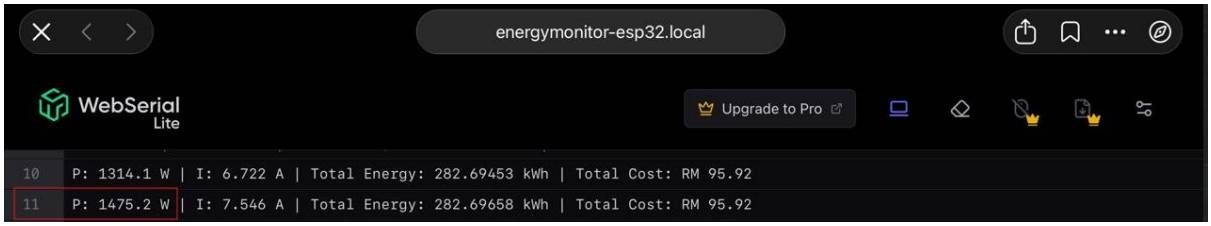


Figure 6.2 Screenshot of hairdryer measured power

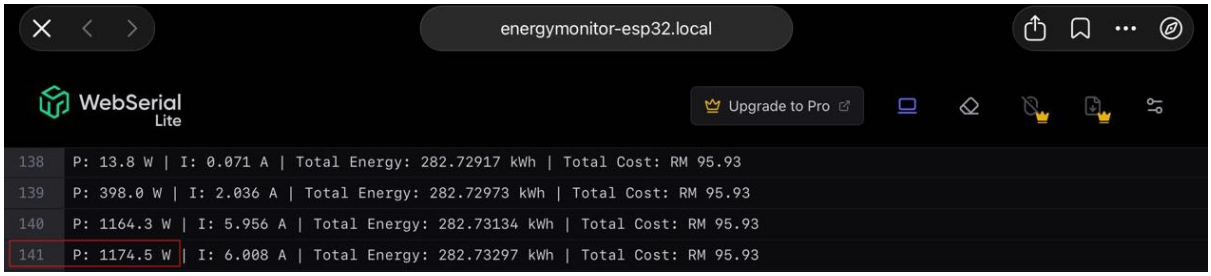


Figure 6.3 Screenshot of induction cooker measured power

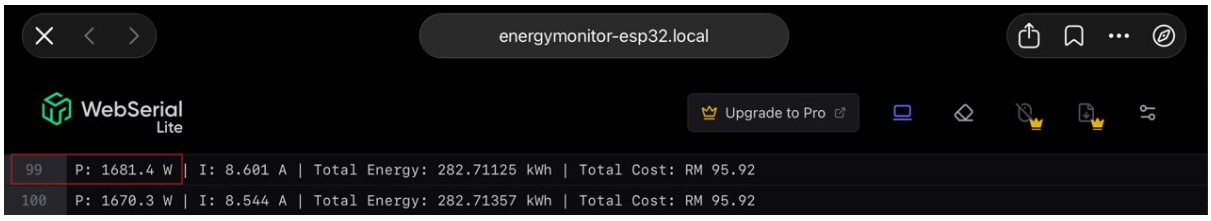


Figure 6.4 Screenshot of electric kettle measured power

Table 6.2 Comparison between rated specifications and system measured power

Test No.	Appliance	Rated Power (W)	Measured Power (W)	APE (%)
1	Standing Fan	40W	21.6W	46.00%
2	Hairdryer	1600W	1475.2W	7.80%
3	Induction Cooker	2000W	1174.5W	41.28%
4	Electric Kettle	1850W	1681.4W	9.11%
Total			**MAPE**	**26.05%**

As illustrated in Table 6.2, the system achieved a MAPE of 26.05%. While this exceeds the ideal error margin of 5.0% for high-precision laboratory equipment, it is acceptable for a non-invasive behavioural monitoring system that relies entirely on appliance sticker ratings.

The significant variation in the results is primarily due to the difference between the manufacturer's sticker ratings and the actual operational power usage. Most appliance labels show the maximum peak power under extreme conditions, however in everyday use, devices such as the induction cooker and standing fan utilise variable power levels dependent on user settings. This explains why Tests 1 and 3 had the highest mistake percentages when compared to their predicted peak ratings.

Furthermore, using a high-capacity 100A sensor (SCT-013-000) presents sensitivity issues when monitoring very low-wattage devices. Low current levels such as the 40W standing fan, result in a lower signal-to-noise ratio, which naturally increases the percentage deviation. High-load resistive appliances such as the hairdryer and electric kettle, had substantially lower error margins (7.8% and 9.1%, respectively), indicating that their usage is closer to the sensor's optimum range.

Despite these external factors, the system proved successful in capturing consistent power signatures for each device. While the exact numbers deviate from the maximum sticker ratings, the RF model is able to learn household routines and deliver relevant tiered cost estimations and sustainability audits.

6.1.3 Data Persistence Performance

A crucial technical requirement for the system is the ability to keep the Energy Odometer running during power outages. This is implemented with the Preferences.h library which makes use of the ESP32 NVS. To assess the effectiveness of this persistence logic, four separate power-cycle trials were performed on the system. Figure 6.5 shows a screenshot of WebSerial terminal logs during NVS data backups.

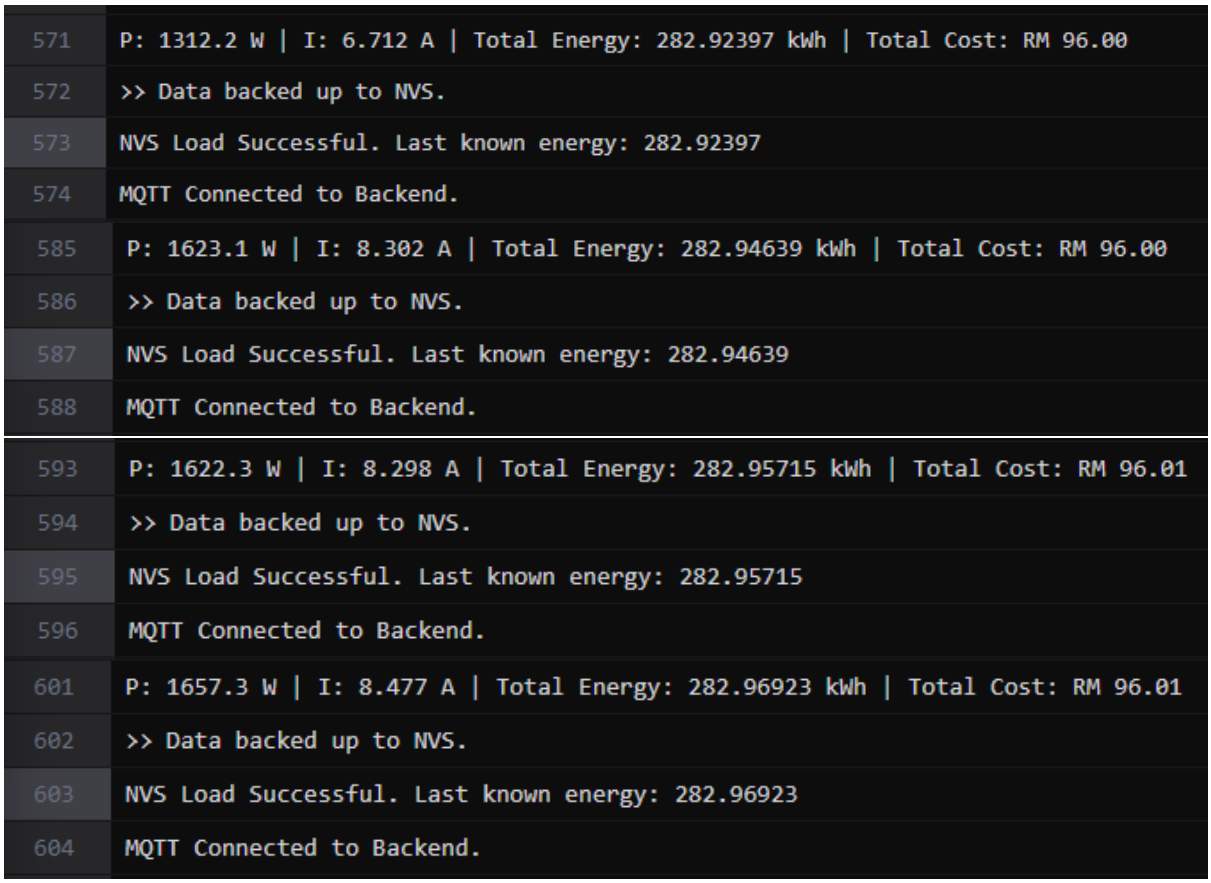


Figure 6.5 Screenshot of WebSerial terminal logs showing NVS data backups

As shown in the figure above, the test comprised monitoring the system until the “Data backed up to NVS” event happened, at which point the power supply was immediately disconnected. After rebooting, the recovered value was confirmed by the “NVS Load Successful” log entry.

Table 6.3 NVS data persistence and odometer recovery results

Test No.	Pre-Reboot Total Energy (kWh)	Post-Reboot Recovered Value (kWh)	Recovery Success (%)	Status
1	282.92397	282.92397	100.00%	Successful
2	282.94639	282.94639	100.00%	Successful
3	282.95715	282.95715	100.00%	Successful
4	282.96923	282.96923	100.00%	Successful

As demonstrated in Table 6.3, the system recovered 100% of the data in all trials. The firmware “Last known energy” value matched the total energy recorded prior to the power cycle to within five decimal places.

This performance confirms the Coalesced Write Policy which triggers a physical flash memory write every 15 seconds. By ensuring that the cumulative energy usage survives hardware resets, the system retains its metrological integrity. This enables accurate long-term financial audits and sustainability reporting while eliminating the possibility of the energy odometer resetting to zero during power outages.

6.1.4 Interaction and Notification Latency

The efficiency of the cloud-based interaction layer was assessed by measuring the time between a system event and the user receiving a notification or report.

Table 6.4 System response time for Telegram Bot interactions

Test No.	Interaction Type	Process Path	Avg. Response Time (s)
1	**Proactive AI Alert**	MQTT → Python AI → Telegram API	≈10.64
2	**Manual Static Alert**	Button → ESP32 → Telegram API	≈ 6.27
3	**Visual Snapshot**	Bot → Grafana Render → Telegram	≈ 4.17
4	**Smart Audit Report**	Bot → InfluxDB → Telegram	≈1.89

The findings in Table 6.4 show a direct relationship between the complexity of the data pipeline and the consequent reaction time.

1. **High Latency in AI Alerts (10.64s):**
   The Proactive AI Alert has the highest latency. This is technically predicted given the process multi-stage computational chain, including the receiving of a MQTT message, temporal feature extraction, a query to InfluxDB to acquire the 7-day historical baseline, the RFR execution and finally the Telegram API handshake.
2. **Manual Alert Efficiency (6.27s):**
   The Manual Static Alert is much faster than the AI alert since it does not employ the machine learning inference engine or database searches, instead relying on a direct interruption trigger from the ESP32 to the Telegram Bot API.
3. **Visual vs. Textual Reporting:**
   The Smart Audit Report has the lowest latency (1.89s) because it used a single mathematical aggregation of time-series data from InfluxDB and transmitted a lightweight text message. In contrast, the Visual Snapshot takes roughly 4.17 seconds due to the Grafana picture-Renderer overhead that must initialise a headless browser instance to synthesise a PNG picture before delivery.

In general, a home IoT system with an average reaction time of less than 11 seconds for the most complex AI-driven tasks can be considered highly efficient.

6.2 Anomaly Detection Model Evaluation

The system intelligence layer is evaluated based on its ability to accurately estimate household energy baselines and effectively identify context-aware abnormalities while avoiding overwhelming the user with false positive notifications. The evaluation focused on the model automated validation lifecycle, temporal feature engineering effectiveness and dynamic thresholding logic practicality.

I. Automated Validation and Self-Adaptation

The RFR is not evaluated as a static entity, but rather for its ability to self-adapt during the retrain_model_auto() lifecycle. During the deployment phase, the system successfully completed its weekly historical data sync, which pulled 7 days of time-series data from InfluxDB.

The Mean Absolute Error (MAE) was the primary metric utilised to evaluate the model during the automatic retraining phase. During the validation gate process, the AI engine separated the newly aggregated 7-day dataset into training and testing subsets. The system repeatedly demonstrated its capacity to compare the newly trained model to the present production model. When the new model achieved a lower MAE, it indicates a better match to the household most recent routine, the system successfully hot-swapped the model parameters into the production inference engine without causing any downtime. This demonstrated that the architecture is completely capable of independent, unsupervised adaptation to shifting lifestyle patterns.

II. Effectiveness of Temporal Feature Engineering

The model's strong predicted accuracy depends mainly on engineered temporal features rather than raw timestamps. During real-time inference, extracting hour_of_day, minute_of_day, and day_of_week are extremely helpful in constructing the Expected Baseline ($\hat{y}$).

Furthermore, the adoption of the is_weekend boolean flag is crucial. The evaluation revealed that the model accurately differentiated between the household low-consumption weekday afternoons and high-consumption weekend afternoons. Without this feature, the system would have falsely identified normal weekend daytime usage as an unusual. By

mapping these precise minute-to-minute cycles, the RF ensemble demonstrated a thorough comprehension of human behavioural patterns.

III. False Positive Mitigation and Threshold Performance

The presence of electrical noise and sudden power spikes creates significant difficulty in domestic energy monitoring often triggers false alarms in rule-based systems. The evaluation of the system Context-Aware Dynamic Thresholding demonstrated that the custom mathematical logic adequately addressed this issue.

The anomaly detection equation utilised during inference:

$$Anomaly = ActualPower > (\hat{y} + 15\sigma + 0.5\hat{y})$$

During live operation, the combination of 15σ (standard deviation) scaling with a 50% buffer ($0.5\hat{y}$) effectively mitigated appliance fluctuations. The scenario analysis (Table 5.1) showed that the system correctly overlooked a 3200W use increase at Saturday noon due to the model expectation of high usage ($\hat{y}$ = 1000W) and significant volatility.

Furthermore, the setting up of the 200W Minimum Power Floor functioned as an effective inference gate. By instructing the engine to ignore anomalies below 200W, the system managed to filter out negligible noise from low-power devices. As a result, the Telegram Bot only sent out “High Power Alert!” alerts for true behavioural anomalies, making it a highly reliable, spam-free proactive alerting mechanism.

6.3 System Reliability and Stress Testing

In order to make sure that the system is suitable for 24/7 continuous residential deployment, both the edge computing node and the containerised cloud backend are tested with rigorous stress testing. The test assessed the system ability to handle high-frequency telemetry, concurrent user interactions and resource constraints on edge devices.

I. Continuous Telemetry Pipeline Stress Test

The data ingestion pipeline was evaluated by subjecting the backend to a continuous stream of MQTT payloads. The ESP32 firmware is set up to conduct a burst read of 1000 analog samples, TRMS calculations and a JSON payload every 5 seconds. Over a standard 24-hour monitoring period, this equals to 17280 discrete telemetry packets published to the Mosquitto broker.

During the stress test, the decoupled TIG stack showed significant reliability. The Telegraf agent successfully buffered and flushed incoming MQTT messages to the InfluxDB TSDB with no packets missing. Because the architecture isolates the database write operations from the MQTT acknowledgement cycle, the ESP32 never had to enter a blocking wait state therefore ensuring that the 5-second sample window remained fully predictable.

II. Asynchronous API and Image Rendering Load

The known bottleneck in visual reporting systems is the computational overhead of headless browser rendering. To test the User Interaction Layer resilience, the Telegram Bot was subjected to a series of rapid, concurrent snapshot requests. For example, requesting the 24h, 7d, and 30d graphs in quick succession.

The asynchronous event-loop architecture of the Python-based bot allowed it to successfully queue incoming Telegram API queries without crashing. However, the validation of the unique Flux fallback mechanism was the test's most important dependability accomplishment. In previous versions, the Grafana Image-Renderer would enter an infinite wait state when a snapshot is requested for a time range that contained no data. This would eventually cause the Python script to break because of a timeout. Through the use of union() functions and fallback tables in Flux queries, the database is forced to immediately provide a formatted Null Set. This prevented the rendering engine from hanging and make sure that in the worst-case scenario, the bot returned a "System Offline / No Data" graphic to the user.

III. Edge Computing and Memory Persistency

The ESP32 was stress-tested to make sure that running heavy edge-computing operations will not cause thermal throttling, memory leakage or watchdog timer (WDT) resets. In addition to operating the 1000-sample ADC burst, the microcontroller also served to host the WebSerial Lite WebSocket server, manage both Wi-Fi networks and MQTT and compute the complex multi-tiered billing logic.

The non-blocking millis() architecture is directly accountable for the system stability under this high load. The ESP32 dual-core processor served the WebSerial diagnostic stream without interfering with the crucial AC waveform sampling loop by avoiding standard delay() routines.

Additionally, the long-term stress test verified the Coalesced Write Policy. For flash memory, writing to the NVS is a high-wear operation. The firmware effectively protected the cumulative data from unexpected power outages while significantly reducing memory degradation by caching the Energy Odometer (totalKWh) locally and only carrying out physical flash commits every 15 seconds.

6.4 Objectives Evaluation

Evaluation of Objective 1: Provide a Real-Time Energy Visibility

The first objective aimed to create a non-invasive hardware prototype with an accurate data flow that delivers live, financially valuable insights This is fully satisfied by deploying a standalone ESP32 edge node and a Dockerized TIG cloud stack. By combining an SCT-013-000 sensor with a 16-bit ADS1115 ADC, the system able to gather high-quality real-time AC current data without invasive wiring. To provide metrological persistence, a NVS energy odometer logic was used which allows cumulative energy data to survive power cycles. Furthermore, the system goes beyond simple logging to programmatically compute the electricity tariff at the edge.

Evaluation of Objective 2: Create Tools for Proactive Energy Management

The second objective is to create an automatic, interactive alert system that transforms passive data into actionable insights using push and pull processes. This was accomplished through the development of a custom Python-based Telegram Bot. This interface successfully turned the user from a passive observer to an active energy manager by providing on-demand pull insights, whereby the bot uses the Grafana Image-Renderer to transmit high-resolution time-series snapshots directly to the user's mobile device. The addition of an interactive inline calendar UI improved the user experience and enabling for smooth energy audits. By comparing these audits to the grid emission factor, the system generates actionable sustainability indicators such as carbon footprint tracking and tree offset equivalencies, as well as intelligent recommendations for consumption optimisation.

Evaluation of Objective 3: Build a System for Anomaly and Fault Detection

The third objective is to develop system intelligence that could detect abnormal consumption patterns and generate predictive alerts. This is performed by implementing a Machine Learning Inference Engine in the cloud architecture that employs a RFR. Unlike static threshold systems, this model used temporal feature engineering to extract characteristics like time of day and weekend indicators to create a dynamic Expected Baseline suited to specific household behaviours. By applying a mathematically precise threshold to the model predictions, the system successfully differentiated true anomalies and possible appliance issues from standard electrical noise.

6.5 Project Challenges

Hardware Calibration and Signal Noise:

One of the key issues was to ensure data accuracy at the hardware level. The SCT-013-000 CT sensor generates a low-voltage analog signal that is sensitive to electromagnetic interference (EMI) and electrical noise. Initial results showed large variations, especially at low current levels. To address this issue, a DC bias circuit was rigorously calibrated, and the built-in ESP32 ADC was replaced with the 16-bit ADS1115. This resulted in higher precision and more consistent TRMS calculations, but it required an extensive calibration against a commercial power meter to reduce the percentage error.

Reliable Data Persistence at the Edge:

Maintaining a constant energy odometer provides a logical obstacle in terms of data integrity. Because the ESP32 calculates cumulative kWh in volatile RAM, any unexpected power interruption or system reset would result in the loss of collected data. Implementing a solution utilising NVS needed a precise balance. Writing to flash memory too frequently could wear out the hardware, while writing infrequently could result in data loss. The final implementation used a timed interval commit method which ensured that the system could reliably restart tracking from the last saved state after a downtime.

Machine Learning Adaptation and Data Quality:

Training an effective anomaly detection model proved challenging due to the fluctuating nature of household energy consumption. Initially, the model struggled to distinguish between typical high-load events and real anomalies. This required the transition from simple thresholding to a RFR with temporal feature engineering. Furthermore, the system had to be configured to accommodate data gaps caused by intermittent Wi-Fi connectivity. To ensure the AI remained accurate over time, a robust retraining pipeline capable of automatically cleaning and validating InfluxDB data prior to model fitting was required.

6.6 Concluding Remark

In this chapter, system testing on all implemented modules is completed to ensure that all functionalities and requirements run effectively and without errors. The initial objectives of the energy monitoring system are thoroughly reviewed and accomplished. Aside from that, the project challenges encountered across the development of this system are highlighted.

Chapter 7
Conclusion and Recommendations

7.1 Conclusion

The Home Energy Monitoring and Alert System development is a comprehensive effort to satisfy the growing demand for real-time energy visibility and intelligent energy management in residential environments. This project successfully combines IoT technology, time-series data analytics and machine learning approaches to provide a feasible and scalable solution for monitoring residential electricity consumption.

The project in designing and implementing a non-invasive hardware architecture using the ESP32 microcontroller, CT sensor and ADS1115 ADC for precise real-time data acquisition is a significant accomplishment. The system efficiently captures and transmits high-frequency energy data to a reliable backend infrastructure using a lightweight MQTT protocol. This allows for constant monitoring while retaining system efficiency and minimum operating overhead.

Furthermore, the project demonstrates the successful implementation of a data pipeline that includes Telegraf, InfluxDB and Grafana. This provides a reliable platform for data storage, processing and visualisation. The resulting dashboards transform raw energy data into actionable insights allow users to monitor consumption patterns, predict electricity costs and make smart energy decisions. Real-time visualisation improves user awareness greatly when compared to standard monthly billing systems.

Furthermore, the implementation of an interactive Telegram Bot brings a unique approach to user interaction and allows both real-time alerts and on-demand data access. This bidirectional connection mechanism improves system accessibility and responsiveness by allowing users to remotely monitor their energy consumption and receive prompt notifications when unusual usage patterns are detected.

Another significant contribution of this project is the adoption of a RF-based machine learning model to detect anomalies. The system can detect unusual energy trends and potential appliance malfunctions more accurately by leveraging historical consumption data and temporal feature engineering. This increases the system ability to facilitate proactive energy management and preventative maintenance.

The project also emphasises the importance of implementing a distributed system architecture that combines edge-level data processing and cloud-based analytics to ensure scalability, dependability and performance. Despite problems faced during implementation, such as hardware calibration, system integration and data consistency, the final system operates efficiently and meets the project objectives.

In conclusion, this project successfully delivers a low-cost, intelligent and user-oriented energy monitoring system that contributes to increased energy efficiency and sustainability. While the system has accomplished its main objectives, it also provides a solid framework for future expansions, especially in terms of increasing analytical capabilities, improving forecast accuracy and supporting larger-scale deployments.

7.2 Future Recommendations

Enhancement of Machine Learning Models for Predictive Analytics:

While the current implementation detects anomalies using a RF model, further studies could focus on developing more complex time-series models like Long Short-Term Memory (LSTM) networks. These models are more adapted to capture temporal dependencies in energy consumption data to allow for more accurate forecasting of future usage trends and enhancing the system capacity to deliver predictive insights and early warnings.

Integration of Appliance-Level Energy Disaggregation:

Future developments might involve the use of Non-Intrusive Load Monitoring (NILM) methods to segment overall home energy use into specific appliance usage. This would provide users with more specific information about which appliances contribute the most to energy usage, hence improving decision-making and energy-saving methods.

Security and Data Privacy Enhancements:

As the system handles sensitive home energy data, future enhancements could prioritise enhanced security features such as end-to-end encryption, secure authentication systems and role-based access control. This ensures that user data is safeguarded while maintaining trust in the system.

Integration with Smart Home Ecosystems:

The system could be enhanced through integration with current smart home platforms and IoT devices. This enables automatic control of appliances based on real-time energy insights and changing the system from a monitoring tool to a completely autonomous energy management solution.

REFERENCES

[1] M. Armie, A. Zaidi, A. Bahari, and Z. A. Wahab, “Development of Energy Consumption Measurement and Monitoring System for Household Applications,” vol. 5, no. 1, pp. 66–78, 2024, doi: 10.30880/peat.2024.05.01.007.

[2] Tanand Technology, “Easi EMS (Energy Management System) - Tanand Technology Malaysia,” Tanand Technology. Accessed: May 04, 2025. [Online]. Available: https://www.tanand.com.my/service/easi-ems-energy-management-system/?utm_term=energy%

[3] P. R. Khanna, G. Howells, and P. I. Lazaridis, “Design and Implementation of Low-Cost Real-Time Energy Logger for Industrial and Home Applications,” *Wirel. Pers. Commun.*, vol. 119, no. 3, pp. 2657–2674, Aug. 2021, doi: 10.1007/s11277-021-08350-1.

[4] T. R. Lin, N. H. Omar Khan, and M. Z. Daud, “Arduino based appliance monitoring system using SCT-013 current and ZMPT101B voltage sensors,” *Przeglad Elektrotechniczny*, vol. 2021, no. 9, pp. 89–94, 2021, doi: 10.15199/48.2021.09.19.

[5] R. Cavallini-Rodriguez and P. Portillo-Mendoza, “Design and Implementation of a Monitoring and Alert System Applied to the Residential Electrical Network Using NodeMCU,” *International Journal of Engineering Trends and Technology*, vol. 72, no. 6, pp. 259–272, Jun. 2024, doi: 10.14445/22315381/IJETT-V72I6P125.

[6] A. Akmal, A. Hamzah[1], and O. A. Hassan, “IoT-Based Electrical Distribution Box Power Monitoring System Using Raspberry PI,” vol. 5, no. 2, pp. 63–76, 2024, doi: 10.30880/peat.2024.05.02.007.

[7] M. Shafiq Bin Anuar and T. Yew Heng, “DESIGN AND DEVELOPMENT OF INTERNET OF THINGS (IoT) BASED SYSTEM FOR MONITORING ENERGY CONSUMPTION.”

[8] Md. H. Rahman, M. Shihab, Md. A. Rahman, and M. Naderuzzaman, “ESP32 Microcontroller: A Review of Architecture, Communication Protocols, Applications and Research Challenges,” *Open Access Journal on Engineering Applications*, vol. 2, no. 1, p. 19, Feb. 2026, doi: 10.64886/oajea.0102.003.

[9] F. Serepas, I. Papias, K. Christakis, N. Dimitropoulos, and V. Marinakis, “Lightweight Embedded IoT Gateway for Smart Homes Based on an ESP32 Microcontroller,” *Computers*, vol. 14, no. 9, Sep. 2025, doi: 10.3390/computers14090391.

[10] M. A. Syahmi Md Dzahir and K. Seng Chia, “Evaluating the Energy Consumption of ESP32 Microcontroller for Real-Time MQTT IoT-Based Monitoring System,” in *2023 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies, 3ICT 2023*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 255–261. doi: 10.1109/3ICT60104.2023.10391358.

[11] A. Bhattacharjee and P. Kumar Gayen, “IOT-BASED MONITORING AND DETECTION OF EARTH FAULTS AND POWER CONSUMPTION IN ELECTRICAL LOADS: A SCALABLE EMBEDDED SYSTEM APPROACH.”

[12] S. Gupta, P. Nayak, and S. Chaudhari, “Low-Cost IoT-Based Downtime Detection For UPS and Behaviour Analysis,” in *International Conference on Communication Systems and Networks, COMSNETS*, Institute of Electrical and Electronics Engineers Inc., 2026, pp. 917–922. doi: 10.1109/COMSNETS67989.2026.11418077.

[13] S. K, K. Samhith, M. K. H S, and S. M, “Watt Watcher: An RTOS-Powered Energy Meter,” Institute of Electrical and Electronics Engineers (IEEE), Dec. 2025, pp. 1–6. doi: 10.1109/i-pact65952.2025.11307903.

[14] L. Rojek, S. Islam, M. Hartmann, and R. Creutzburg, “IoT-Based Real-Time Monitoring System for a Smart Energy House.”

[15] A. S. Alexander, “CLOUD-BASED MQTT PROTOCOL FOR POWER MONITORING AND SOCKET CONTROL SYSTEM.”

[16] S. Ronaghi, A. Ferrero, L. Cristaldi, and M. Mussetta, “DATA-DRIVEN APPROACHES FOR ANOMALY DETECTION IN ENERGY METERING,” 2025.

[17] J. Im, J. Lee, S. Lee, and H. Y. Kwon, “Data pipeline for real-time energy consumption data management and prediction,” *Front. Big Data*, vol. 7, 2024, doi: 10.3389/fdata.2024.1308236.

[18] E. Omol, L. Mburu, P. Abuonji, and D. Onyango, “Anomaly Detection In IoT Sensor Data Using Machine Learning Techniques For Predictive Maintenance In Smart Grids.” [Online]. Available: http://ijstm.inarah.co.id

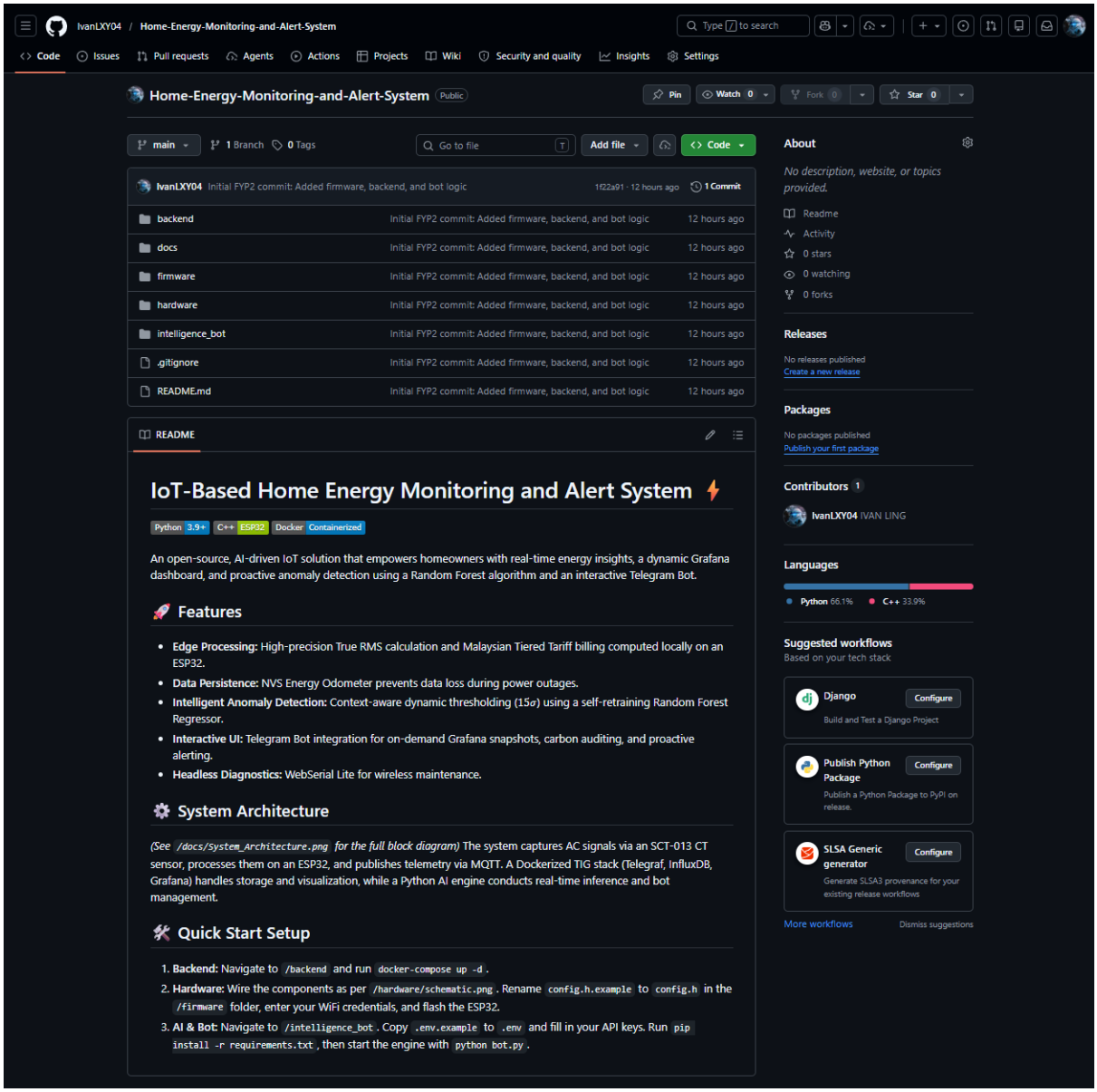
[19] N. Sarwar, I. S. Bajwa, M. Z. Hussain, M. Ibrahim, and K. Saleem, “IoT Network Anomaly Detection in Smart Homes Using Machine Learning,” *IEEE Access*, vol. 11, pp. 119462–119480, 2023, doi: 10.1109/ACCESS.2023.3325929.

[20] P. P. Kasaraneni, Y. Venkata Pavan Kumar, G. L. K. Moganti, and R. Kannan, “Machine Learning-Based Ensemble Classifiers for Anomaly Handling in Smart Home Energy Consumption Data,” *Sensors*, vol. 22, no. 23, Dec. 2022, doi: 10.3390/s22239323.

[21] R. Mischianti, “DOIT ESP32 DEV KIT V1 high resolution pinout and specs,” mischianti.org.

APPENDIX

The repository link (comprising C++, Python, Flux queries and YAML configurations) is as follows: https://github.com/IvanLXY04/Home-Energy-Monitoring-and-Alert-System



POSTER

Universiti Tunku Abdul Rahman
Faculty of Information and Communication Technology

Home Energy Monitoring and Alert System Using ESP32 and CT Sensors for Real-Time Power Tracking

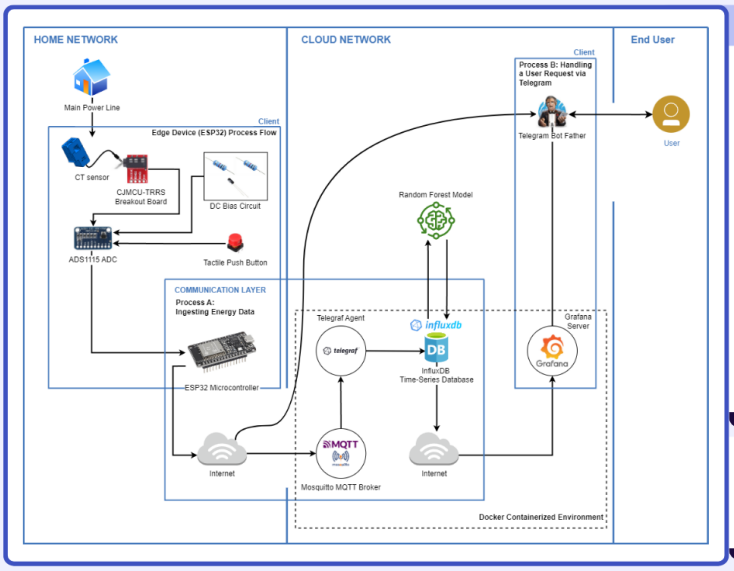
An open-source IoT solution that empowers homeowners with real-time energy insights, dynamic dashboard and interactive alerts to save money and reduce waste.

Introduction

Homeowners lack visibility into their real-time energy usage, relying on outdated and uninformative monthly bills. This makes it difficult to manage consumption, identify sources of energy waste, or detect appliance faults in a timely manner. This project develops a low-cost and open-source IoT system to provide real-time feedback directly to the user.

Research Objectives

- Provide a Real-Time Energy Visibility
- Create Tools for Proactive Energy Management
- Build a System for Anomaly and Fault Detection



System Architecture

1. Home Network:
ESP32, CT Sensor, ADS1115

2. Communication:
Wi-Fi ⟶ MQTT Broker

3. Cloud Network:
Docker stack (Telegraf, InfluxDB, Grafana) + Machine Learning Inference Engine (Random Forest)

4. End User:
Connection to Grafana and the Telegram Bot

GitHub repository
To view full system architecture and codes

Method

System Design
A distributed architecture with an ESP32 edge device for sensing and a self-hosted, Docker-based backend for data management.

Hardware Used
- ESP32 microcontroller
- SCT-013-000 CT sensor
- ADS1115 ADC
- CJMCU-TRRS 3.5mm jack breakout board
- Resistor
- Capacitor

Data Flow
The ESP32 publishes data via the MQTT protocol to a Mosquitto broker, where a Telegraf agent collects and stores it in an InfluxDB time-series database.

Result

Hardware Performance

Measurement
Real-world deployment accurately measures real-time current and power.

Calculation
Performs on-device calculations and saves cumulative data to non-volatile storage.

Backend Functionality

Docker
Docker-based backend is stable and efficient.

IoT Messaging
MQTT-Telegraf-InfluxDB flow ingests and stores all sensor data without loss.

Machine Learning
Self-retraining Random Forest model to identify anomalies.

User Interface & Experience

Grafana
Grafana dashboard provides clear visualizations of live and historical data.

Telegram bot
Send real-time high-power alerts, on-demand dashboard snapshots, interactive calendar UI and smart energy audit.

Conclusion

Developed an intelligent, self-adaptive IoT energy monitoring system that is functional, low-cost and user-friendly. By providing real-time visibility, powerful dashboard and unique interactive alerts, the system empowers users to effectively manage their energy consumption.

Project Developer: Ivan Ling Xin Yu **Project Supervisor:** Dr. Mohd Hafizul Afifi Bin Abdullah