

**Smart Household Energy Monitoring and Management: A Comprehensive IoT-Based Solution for Sustainability.**

Matthew Mc Daid

BSc (Hons) in

Applied Computing

2023/2024

**Smart Household Energy Monitoring and Management: A Comprehensive IoT-Based Solution for Sustainability.**

Author: Matthew Mc Daid

Supervised by: Thomas Devine

A thesis submitted in partial fulfilment of the requirements for “Research in Computing with Emerging Technologies” and “Project Development”

BSc (Hons) in Computing in Applied Computing

Submitted to Atlantic Technological University

*Arna chur isteach chuig Ollscoil Teicneolaiochta an Atlantaigh* April 2024

# Declaration

I hereby certify that the material, which l now submit for assessment on the programmes of study leading to the award of Master of Science in Computing in Applied computing, is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution. I understand that it is my responsibility to ensure that I have adhered to ATU’s rules and regulations.

I hereby certify that the material on which I have relied on for the purpose of my assessment is not deemed as personal data under the GDPR Regulations. Personal data is any data from living people that can be identified. Any personal data used for the purpose of my assessment has been pseudonymised and the data set and identifiers are not held by ATU. Alternatively, personal data has been anonymised in line with the Data Protection Commissioners Guidelines on Anonymisation.

I consent that my work will be held for the purposes of education assistance to future students and will be shared on the ATU Donegal (Computing) website (www.lyitcomputing.com) and Research THEA website (https://research.thea.ie/). I understand that documents once uploaded onto the website can be viewed throughout the world and not just in Ireland. Consent can be withdrawn for the publishing of material online by emailing Thomas Dowling; Head of Department at thomas.dowling@lyit.ie to remove items from the ATU Donegal Computing website and by email emailing Denise McCaul; Systems Librarian at denise.mccaul@lyit.ie to remove items from the Research THEA website. Material will continue to appear in printed formats once published and as websites are public medium, ATU cannot guarantee that the material has not been saved or downloaded.

Signature of Candidate - Matthew Mc Daid Date – 26/04/2024

# Acknowledgements

I would like to express my deepest gratitude to those who have significantly influenced my academic path. Foremost, I extend my sincerest thanks to Thomas Devine for his steadfast support, insightful guidance, and invaluable mentorship throughout this research project. His expertise has not only shaped this endeavour but also my broader academic perspective.

Additionally, I am deeply grateful to all my lecturers, whose profound knowledge and dedication have been crucial in shaping my educational journey. Their commitment to excellence has inspired and propelled me towards achieving my academic goals.

I must also acknowledge the support and camaraderie of my peers—Shane, Andrae, Conor, and Jack. Their support during the challenging initial phases of this project was indispensable. Their encouragement and collaboration have not only eased my journey but have also enriched my experience, making this academic venture a truly collaborative effort.

Each of you has played a pivotal role in this journey, and for that, I am eternally appreciative.

# Abstract

This study investigates the application of Internet of Things (IoT) technology in residential energy monitoring, specifically focusing on real-time electricity consumption tracking within individual household circuits. Central to this research is the use of a 4 Channel ESP-8266 Mains Current Sensor, akin to an Arduino development board, for data collection. Complementing this is the ESP-8266 Wi-Fi Module, a comprehensive System on a Chip (SoC) equipped with an in-built TCP/IP stack, enabling seamless integration into domestic Wi-Fi networks. The pivotal component of this system is the Current Transformer (CT) Clamp, designed to safely attach to either Live or Neutral wires, facilitating an installation process that is both user-friendly and compliant with standard electricity provider regulations.

The primary challenge addressed by this project is the conspicuous absence of efficient and accessible tools for homeowners to monitor and manage their energy consumption. In a context where energy efficiency and environmental sustainability are increasingly vital, as underscored by recent research (Pope, 2023)., the need for accurate and instantaneous access to energy usage data is paramount. The current deficit in such tools not only leads to inefficient energy use and higher costs but also contributes significantly to environmental degradation. This research aims to bridge this gap by providing a holistic solution that empowers individuals with the necessary information to make informed decisions, reduce energy wastage, and contribute effectively to the global pursuit of sustainability.

# Acronyms

|  |  |  |
| --- | --- | --- |
| Acronym | Definition | Page |
| CT | Current Transformer | 5 |
| IoT  TCP  IP  MQTT | Internet Of Things  Transmission control Protocol  Internet Protocol  Messaging Queuing Telemetry Transport | 1  5  5  12 |
| UI | User Interface | 14 |
| MQTT | Message Queuing Telemetry Transport | 13 |
| MAUI  WI-FI  CoAP  HTTP  kWh  USB  HDMI  DIY  IO  IDE  SQL  AC  DC | Multi-platform App UI  wireless fidelity  Constrained Application Protocol  Hypertext Transfer Protocol  kilowatt hour  Universal Serial Bus  High-Definition Multimedia Interface  Do It yourself.  Input Output  Integrated Development environment  Structured Query Language  Alternating Current  Direct Current | 16  18  18  18  20  22  22  22  22  23  24  26  26 |

# Table of Contents

[Declaration 3](#_Toc164990438)

[Acknowledgements 4](#_Toc164990439)

[Abstract 5](#_Toc164990440)

[Acronyms 6](#_Toc164990441)

[Table of Contents 7](#_Toc164990442)

[Table of Figures 13](#_Toc164990443)

[Table of Tables 15](#_Toc164990444)

[Table of Code Listings 15](#_Toc164990445)

[1. Introduction 1](#_Toc164990446)

[1.1. Purpose 2](#_Toc164990447)

[1.2. Background 3](#_Toc164990448)

[1.3. Research Question 4](#_Toc164990449)

[1.4. Research Aims 4](#_Toc164990450)

[1.5. Research Objectives 5](#_Toc164990451)

[1.6. Report Outline 7](#_Toc164990452)

[2. Literature review 8](#_Toc164990453)

[2.1. Introduction 8](#_Toc164990454)

[2.2. What is Internet of Things (IoT) 9](#_Toc164990455)

[2.2.1. How does IoT Function? 9](#_Toc164990456)

[2.3. Raspberry Pi Vs 4 Channel ESP-8266 Mains Current Sensor 11](#_Toc164990457)

[2.3.1. Raspberry Pi 11](#_Toc164990458)

[2.3.2. 4 Channel ESP-8266 Mains Current Sensor 11](#_Toc164990459)

[2.4. Wemos D1 Mini 13](#_Toc164990460)

[2.5. MQTT Protocol 14](#_Toc164990461)

[2.6. Arduino IDE vs Home Assistant 15](#_Toc164990462)

[2.6.1. Arduino IDE 15](#_Toc164990463)

[2.6.2. Home Assistant using ESP. 15](#_Toc164990464)

[2.6.3. IDE choice 16](#_Toc164990465)

[2.7. Essentials of Electricity: An Overview 17](#_Toc164990466)

[2.7.1. Types of Current 17](#_Toc164990467)

[2.7.2. Electricity in The Home 18](#_Toc164990468)

[2.7.3. Ohm’s Law 19](#_Toc164990469)

[2.8. Methods of Measuring electricity 20](#_Toc164990470)

[2.8.1. Recent method of measuring electricity 21](#_Toc164990471)

[2.9. Current Transformer Clamp 22](#_Toc164990472)

[2.10. What is a cloud database? 23](#_Toc164990473)

[2.11. Comparing Cloud Databases 25](#_Toc164990474)

[2.11.1. Firebase 25](#_Toc164990475)

[2.11.2. SQLite 26](#_Toc164990476)

[2.12. Comparing Cross platform technologies 27](#_Toc164990477)

[2.12.1. What is .NET MAUI? 27](#_Toc164990478)

[2.12.2. Flutter 28](#_Toc164990479)

[2.12.3 React Native 29](#_Toc164990480)

[2.13. Similar Projects 30](#_Toc164990481)

[2.13.1. Design of an IoT Energy Monitoring System (Chooruang and Meekul, 2018). 30](#_Toc164990482)

[2.13.2. OWL Micro+ - Simple Energy Monitoring. (OWL Micro+, 2023). 31](#_Toc164990483)

[2.14. Conclusion 32](#_Toc164990484)

[3. Design 33](#_Toc164990485)

[3.1. Introduction 33](#_Toc164990486)

[3.2. Hardware Requirements 34](#_Toc164990487)

[3.2.1 Connecting the CT clamp to ESP-8266 35](#_Toc164990488)

[3.3. Software Requirements 36](#_Toc164990489)

[3.4. Functional and Non-Functional Requirements 37](#_Toc164990490)

[3.4.1. Functional Requirements 37](#_Toc164990491)

[3.4.2. Non-Functional Requirements 38](#_Toc164990492)

[3.6. Data Flow 39](#_Toc164990493)

[3.7. Process Flowchart 40](#_Toc164990494)

[3.7.2. Choose Circuit to Monitor 41](#_Toc164990496)

[3.7.3. Clamp CT to Wire 41](#_Toc164990497)

[3.7.4. Power Up the Devices 41](#_Toc164990498)

[3.7.5. Read Sensor Value 41](#_Toc164990499)

[3.7.6. Sensor Reading 41](#_Toc164990500)

[3.7.7. Upload to Firebase 41](#_Toc164990501)

[3.7.8. Connection Success 41](#_Toc164990502)

[3.7.9. Save Results to File 41](#_Toc164990503)

[3.7.10. Fetch Data from .NET MAUI 41](#_Toc164990504)

[3.7.11. Connection Success 42](#_Toc164990505)

[3.7.12. Visualise Data 42](#_Toc164990506)

[3.8. Use Case Diagram 43](#_Toc164990507)

[3.10. UI Design 44](#_Toc164990508)

[3.10.1. Flyout navigation of application UI 45](#_Toc164990510)

[3.10.2. Setting Electricity Threshold UI 46](#_Toc164990511)

[3.10.3. Notifications section 47](#_Toc164990512)

[3.10. Conclusion 48](#_Toc164990513)

[4.0. Implementation 49](#_Toc164990514)

[4.1. Introduction 49](#_Toc164990515)

[4.2. Setting Up ESP32 49](#_Toc164990516)

[4.2.1. ESP32 49](#_Toc164990517)

[4.2.2. Configuring ESP32 Hardware for Mains Current Sensing 50](#_Toc164990518)

[4.3. ESP-WROOM-32 51](#_Toc164990519)

[4.3.1. ESP-WROOM-32 51](#_Toc164990520)

[4.3.2. Configuring The ESP-Wroom-32 Hardware 52](#_Toc164990521)

[4.4. CT Clamp Connections 53](#_Toc164990522)

[4.5. Arduino IDE 54](#_Toc164990523)

[4.5.1. Downloading Necessary boards via Board Manager 54](#_Toc164990524)

[4.5.2. Downloading Port Drivers 55](#_Toc164990525)

[4.5.3. Selecting Board and Port 56](#_Toc164990526)

[4.5.4. Installing Library’s necessary. 56](#_Toc164990527)

[4.5.5. Testing If the IDE is talking to the ESP32. 57](#_Toc164990528)

[4.6. Coding of The Arduino 58](#_Toc164990529)

[4.6.1. Establishing Network Connectivity and Cloud Integration 58](#_Toc164990530)

[4.6.2. Data Acquisition and Processing 59](#_Toc164990531)

[4.6.3. Data Management and Communication 60](#_Toc164990532)

[4.7. Firebase 61](#_Toc164990533)

[4.7.1. Firebase Database Creation and structure 61](#_Toc164990534)

[4.7.2. Average Database Structure (24 Hours) 62](#_Toc164990535)

[4.7.3. Rules and Configuration 62](#_Toc164990536)

[4.8. .NET MAUI Blazor Application 63](#_Toc164990537)

[4.8.1. NuGet Packages Required 63](#_Toc164990538)

[4.8.2. Setting up Twilio. 64](#_Toc164990539)

[4.8.4. Setting Up Real-Time Data Fetching 65](#_Toc164990540)

[4.8.5. Continuous Data Retrieval 66](#_Toc164990541)

[4.8.6. Real-Time Data Update Event. 66](#_Toc164990542)

[4.8.7. Alerting the User of Tariff Updates 67](#_Toc164990543)

[4.8.8. Implementation APEX charts 68](#_Toc164990544)

[4.8.9. Setting a Threshold on Electricity Consumption 68](#_Toc164990545)

[5.0. Testing Strategy 69](#_Toc164990546)

[5.1. Introduction 69](#_Toc164990547)

[5.2. White-Box Testing 69](#_Toc164990548)

[5.2.1. Unit Testing 69](#_Toc164990549)

[5.2.2. Integration Testing 70](#_Toc164990550)

[5.3. Black-Box Testing 70](#_Toc164990551)

[5.3.1. Functionality Testing 71](#_Toc164990552)

[5.3.2. Exploratory Testing 71](#_Toc164990553)

[5.3.3. Sensor Testing 74](#_Toc164990554)

[5.3.4. How to Test the Sensor 74](#_Toc164990556)

[5.3.5. Ensuring Measurement Accuracy in Energy Monitoring 75](#_Toc164990559)

[6.1. Results and Evaluation 76](#_Toc164990560)

[6.1.1. Circuit Monitoring and Alert System on Smart Home Dashboard 76](#_Toc164990561)

[6.1.2. Live Data and Circuit Selection 77](#_Toc164990562)

[6.1.3. Setting Electrical consumption Limit. 78](#_Toc164990563)

[6.1.4. Individual Breakdown of Live Consumption. 79](#_Toc164990564)

[6.1.5. Historical Weekly Data Table. 80](#_Toc164990565)

[6.1.6. Notification Outcomes for Circuit Limit Monitoring 81](#_Toc164990566)

[6.1.7. Adaptive Tariff Rate Notifications for Optimised Energy Usage 82](#_Toc164990567)

[Chapter 7: Conclusion 83](#_Toc164990568)

[7.1. Introduction 83](#_Toc164990569)

[7.2. Summary of Findings and Conclusions 83](#_Toc164990570)

[7.2.1. Literature Review on IoT and Energy Monitoring 83](#_Toc164990571)

[7.2.2. Implementation and Testing of the IoT System 83](#_Toc164990572)

[7.3. Recommendations for Future Work 84](#_Toc164990573)

[7.4. Self-reflection 84](#_Toc164990574)

[7.5. Final Thoughts 85](#_Toc164990575)

[Appendices A: References 86](#_Toc164990576)

[Appendices B: (Code Listings) 91](#_Toc164990577)

[B.1 - System Configuration and initialisation 91](#_Toc164990578)

[B.2 - Data Acquisition and Processing 92](#_Toc164990579)

[B.3- Data Management and Communication 93](#_Toc164990580)

[B.4- Rules & Configuration 94](#_Toc164990581)

[B.5- Model Class for Live Data 94](#_Toc164990582)

[B.6- Setting Up Real-Time Data Fetching 95](#_Toc164990583)

[B.8- Real-Time Data Update Event. 96](#_Toc164990585)

[B.9- Alerting the User of Tariff Updates. 97](#_Toc164990586)

[B.10- Apex Charts 98](#_Toc164990587)

[B.11- User Defined Threshold for Electricity Consumption 98](#_Toc164990588)

[GitHub Link for Project 98](#_Toc164990589)

[Appendices C: Additional Images 99](#_Toc164990590)

# Table of Figures

[Figure 1. 4 Channel ESP-8266 Mains Current Sensor 12](#_Toc164989345)

[Figure 2. D1 Mini NodeMcu 13](#_Toc164989346)

[Figure 3. Schneider MCB, with load cable terminal. 18](#_Toc164989347)

[Figure 4. Ohms Law Triangle Technique 19](#_Toc164989348)

[Figure 5.Thomas Edison – Chemical Meter made in 1882. 20](#_Toc164989349)

[Figure 6. 24-hour Mechanical Meter. 21](#_Toc164989350)

[Figure 7. CT Clamp (Current transformer clamp). 22](#_Toc164989351)

[Figure 8. ONL Micro+ Product Pictures 31](#_Toc164989352)

[Figure 9. Connecting the CT clamp to ESP-8266 via 3.5mm connection 35](#_Toc164989353)

[Figure 10. Dataflow of the system 39](#_Toc164989354)

[Figure 11. Process Flowchart- See below for explanations of each stage. 40](#_Toc164989355)

[Figure 12 Use case diagram. 43](#_Toc164989356)

[Figure 13. Home Page. 44](#_Toc164989357)

[Figure 14. Flyout Navigation. 45](#_Toc164989358)

[Figure 15. Setting Electricity Threshold. 46](#_Toc164989359)

[Figure 16. Notification centre to view alerts. 47](#_Toc164989360)

[Figure 17 ESP32 CT Board 50](#_Toc164989361)

[Figure 18 Wroom WIFI Chip 51](#_Toc164989362)

[Figure 19 ESP32 and ESP-WROOM Boards Connected 52](#_Toc164989363)

[Figure 20 Pins Responsible for Connecting The 2 Boards 52](#_Toc164989364)

[Figure 21 Completed Hardware for Project 53](#_Toc164989365)

[Figure 22 ESP32. 54](#_Toc164989366)

[Figure 23 Installing Port Drivers 55](#_Toc164989367)

[Figure 24 Board and Port GUI 56](#_Toc164989368)

[Figure 25 Showcasing Available Examples 57](#_Toc164989369)

[Figure 26 Firebase Database Structure Circuit 1 & 2 61](#_Toc164989370)

[Figure 27 Firebase Average Section 62](#_Toc164989371)

[Figure 28 Successful Twillo Setup 64](#_Toc164989372)

[Figure 29. Extension lead used during testing. 74](#_Toc164989373)

[Figure 30 AC Clamp Meter 75](#_Toc164989374)

[Figure 31 Main Screen of Application 76](#_Toc164989375)

[Figure 32 Live Data Details and Circuit Selection 77](#_Toc164989376)

[Figure 33 Set Limit Section 78](#_Toc164989377)

[Figure 34 Individual Live Circuit View 79](#_Toc164989378)

[Figure 35 Historical Weekly Data Table 80](#_Toc164989379)

[Figure 36 Alert Notifications 81](#_Toc164989380)

[Figure 37 Result of Tariff Changes 82](#_Toc164989381)

[Figure 38 Full View Circuit Details 99](#_Toc164989382)

# Table of Tables

[Table 1. Advantages of Cloud databases 24](#_Toc164990726)

[Table 2. Drawbacks of Cloud Databases 24](#_Toc164990727)

[Table 3. Critique of Firebase 25](#_Toc164990728)

[Table 4. Critique of SQLite 26](#_Toc164990729)

[Table 5. Hardware Requirements 34](#_Toc164990730)

[Table 6. Software Requirements 36](#_Toc164990731)

[Table 7 Required Packages 57](#_Toc164990732)

[Table 8 NuGet Packages Required 63](#_Toc164990733)

[Table 9. Testing Table 72](#_Toc164990734)

# Table of Code Listings

[Listing 1 System Configuration and initialisation 58](#_Toc164985517)

[Listing 2 Data Acquisition and Processing 59](#_Toc164985518)

[Listing 3 Data Management and Communication 60](#_Toc164985519)

[Listing 4 Rules & Configuration 62](#_Toc164985520)

[Listing 5 Model Class for Live Data 64](#_Toc164985521)

[Listing 6 Setting Up Real-Time Data Fetching 65](#_Toc164985522)

[Listing 7 Continuous Data Retrieval 66](#_Toc164985523)

[Listing 8 Real-Time Data Update Event. 66](#_Toc164985524)

[Listing 9 Alerting the User of Tariff Updates 67](#_Toc164985525)

[Listing 10 Apex Charts 68](#_Toc164985526)

[Listing 11 User Defined Threshold for Electricity Consumption 68](#_Toc164985527)

# Introduction

In a world where energy conservation and sustainability are of great importance(Pope, 2023)., this thesis explores a comprehensive IoT-based solution for household energy monitoring and management. The project centres around the use of Current Transformer (CT) clamps, cloud databases, C++, and IoT sensors to empower individuals to monitor and manage their energy consumption effectively.

This study will focus on the potential of Internet Of Things (IoT) sensors within the field of electrical systems, focusing on the real-time data collection from individual household circuits. In the vast realm of energy monitoring, the integration of Internet Of Things (IoT) sensors stands out as a revolutionary force. These intelligent devices, seamlessly embedded within electrical systems, empower real-time data collection, providing an intricate understanding of energy consumption patterns. These collections of sensors facilitate more efficient distribution and optimise usage. Whether in industrial contexts or residential domains, these sensors create a comprehensive overview of energy utilisation, enabling proactive management, minimising wastage, and fostering sustainability. The seamless pairing of IoT technology with energy monitoring not only enhances operational efficiency but also lays the groundwork for a more intelligent and adaptive energy ecosystem.

## Purpose

The primary objective of this research is to develop a robust and user-friendly IoT-based system that harnesses MQTT messaging, user-defined thresholds, and the capability to monitor individual circuits. This system aims to empower homeowners in managing their household energy consumption in real-time.

The widely adopted "smart meter" initiative, while efficient for electricity providers, falls short in providing substantial benefits to homeowners in monitoring their electricity usage. The distinction between a "Smart energy meter" and a "Smart energy monitor" is crucial. A meter gauges the overall usage of the entire home, offering limited utility for someone seeking to reduce their bill. Homeowners lack insights into which room consumes the most electricity, making it challenging to implement effective energy-saving measures.

To address this gap, a cross-platform application accessible on IOS, Android, and PCs will be developed, offering a more intuitive monitoring solution. This application allows homeowners to measure individual circuits, displaying their usage on an easy-to-navigate interface with graphs illustrating peak electricity usage. This approach facilitates a seamless experience for accessing data, setting thresholds, and receiving real-time alerts.

The study's goal is to enhance user control over energy consumption, optimise electricity usage during off-peak hours, and promote sustainability. Importantly, it ensures data security and privacy in a cloud-based environment.

## Background

The background of this research project is rooted in the ever pressing need to address energy efficiency and sustainability in the context of modern households. It stems from a history of escalating energy consumption, a reliance on electricity, and the associated environmental implications. As households continue to evolve and become increasingly dependent on electrical appliances and technologies, the demand for user-friendly tools to monitor and manage energy consumption in real-time has become paramount (EIA, 2023).

This research seeks to build upon the historical developments in the field of energy monitoring and management, which have predominantly focused on macro-level solutions for industrial and commercial sectors.

The current methods for measuring electricity consumption, such as traditional meters, often prove to be insufficient for empowering homeowners in effective energy management. (Ferrero, 2008). While these methods are instrumental for billing purposes, they fall short in providing homeowners with detailed insights into their electricity usage patterns. Unlike the proposed Smart energy monitor, conventional meters offer a generalised view of total household consumption. They lack granularity in identifying specific areas or appliances contributing to high energy consumption. This limitation hinders homeowners from making informed decisions to optimise their electricity usage and implement targeted energy-saving strategies (Vadda and Seelam, 2013). The need for a more sophisticated and granular approach to electricity monitoring becomes evident in enhancing the efficiency of energy consumption within households (Ed, 2018).

In contrast, this study fills a critical research gap by focusing on energy thresholds. In practical terms, this means that users can establish a predefined energy consumption limit. When their household electricity usage approaches a specific percentage of this limit, the system will promptly notify the user (Despa et al., 2018).

The project's interdisciplinary nature involves the integration of electrical engineering, data analysis, Cross platform Development and cloud services to create an innovative IoT-based system that enables homeowners to make informed decisions regarding energy consumption.

The significance of this research lies in its potential to empower individuals to actively participate in reducing energy wastage, minimising costs, and contributing to a more sustainable future. By bridging the gap between existing technologies and the specific needs of households, this study aims to provide practical and user-friendly solutions that align with the current trend of environmentally conscious living.

Ultimately, this research seeks to enhance our understanding of household energy consumption and promote sustainable practices that benefit both homeowners and the environment.

## 1.3. Research Question

How does the implementation of a user-friendly, real-time feedback IoT-based system for household energy monitoring influence measurable reductions in electricity consumption and costs, and how does it contribute to the adoption of sustainable energy practices?

## 1.4. Research Aims

This research aims to develop an IoT-based system for comprehensive household energy monitoring and management. The core focus is on implementing Current Transformer (CT) clamps to capture real-time data from individual electrical circuits within a residential setting. The project seeks to design a user-friendly cross-platform application that visualises energy consumption data and provides real-time insights to homeowners. Additionally, the research endeavours to explore cost-effective electricity usage strategies, informed by real-time data and time-of-use pricing models. The implementation of MQTT messaging thresholds is another key aim, enabling users to receive alerts when their electricity consumption approaches predefined limits. Security and privacy considerations are integral, with a particular focus on safeguarding user data stored in the cloud-based Firebase Realtime Database. The research will culminate in a practical test within a residential environment to evaluate the system's impact on energy consumption and user behaviour. Ultimately, the study will provide valuable recommendations and insights for homeowners, enabling them to make informed decisions about energy consumption and contribute to a more sustainable future.

## 1.5. Research Objectives

* **Research into the current methods of measuring electricity consumption:**

Understanding the current methods of measuring electricity consumption is crucial for developing a more efficient and accurate system. This involves identifying gaps, challenges, and potential improvements to inform the development of the project.

* **Research how what units CT clamps measure in:**

Investigating the units measured by CT clamps is essential to ensure compatibility with project goals. Knowing the units helps in designing a system that accurately interprets and utilises the data collected, contributing to the overall precision and effectiveness of the electricity consumption measurement.

* **Research and investigate what hardware is best suited to the project:**

Choosing the right hardware is pivotal in the success of the project. Through thorough research and investigation, the aim is to identify and select hardware that aligns with project requirements, offering the necessary functionalities and capabilities for seamless integration and optimal performance.

* **Research and investigate which software IDE is best suited to the Hardware:**

Determining the most suitable software IDE is a vital aspect of the project. It requires exploring a wide range of available options, each with distinct features and capabilities, to ensure optimal compatibility and functionality.

* **Research on MQTT and how to implement it:**

Researching MQTT and its implementation is crucial for establishing effective communication between devices in the IoT system. Understanding MQTT protocols and mechanisms will enable the creation of a robust and reliable messaging system, facilitating seamless data transmission and reception.

* **Research on cloud databases and which is most compatible:**

Investigating cloud databases is essential for determining the most suitable platform to store and manage data collected from IoT devices. Compatibility considerations involve assessing factors such as accessibility, and integration capabilities to ensure a seamless and efficient data storage solution.

* **Investigate what cross-platform development is best suited:**

Exploring all aspects of programming languages is essential for evaluating its suitability in developing a cross-platform application. Understanding its features, advantages, and limitations will inform the decision-making process, ensuring that the chosen technology aligns with project goals and user interaction requirements.

* **Exploring C++ code for the first time:**

Delving into C++ for the first time is a valuable learning objective. C++ versatility and readability make it a popular choice for various applications, expanding programming skills and potentially offering solutions and insights for the project.

* **Use C++ to configure the ESP-8266:**

utilising C++ to configure the ESP-8266 microcontroller is a practical application of programming skills. This objective aims to leverage C++ capabilities to streamline the configuration process, enhancing efficiency and effectiveness in setting up the hardware for data collection.

* **Use C++ to convert Amps to watts:**

Employing C++ for converting Amps to watts is a specific application of the programming language to handle essential calculations within the project. This objective ensures accuracy in translating electrical measurements into meaningful units for analysis and data interpretation.

* **Sending the data to Firebase using C++ code:**

Using C++ to send data to Firebase is a critical step in establishing a seamless connection between the IoT system and the chosen cloud database. This objective focuses on integrating C++ capabilities with Firebase, ensuring efficient data transfer and storage.

* **Design a front-end UI:**

Designing a user interface (UI) is crucial for creating a user-friendly and intuitive experience. This objective emphasises the importance of a well-designed front-end to enhance user interaction, providing a visually appealing and functional interface for accessing and interpreting electricity consumption data.

## 1.6. Report Outline

1. **Chapter 1: Introduction**

* Overview of the Project
* Research Objectives
* Scope and Limitations

1. **Chapter 2: Literature Review**

* Evolution of Energy Monitoring Systems
* Review of Current IoT Technologies
* Identification of Research Gaps

1. **Chapter 3: Design**

* Hardware and Software Design
* Multiple Diagrams for the project.
* System Architecture and User Interface

1. **Chapter 4: Testing strategy**

* Overview of Testing Methodologies
* Testing Table With functional requirements.
* Sensor Testing and Calibration

# Literature review

## Introduction

The literature review chapter serves as a comprehensive exploration, drawing from peer reviewed publications to gather curated knowledge about the exploration and general overview of electricity, delving into its monitoring methodologies, historical evolution, and contemporary practices.

Electricity, a crucial force fuelling the machinery of modern life, serves as the beating heart of our interconnected world. From lighting up our homes to driving industries, it plays an essential role in shaping our daily routines and global economies. Recognising its pivotal role, it's imperative to acknowledge the need for responsible energy consumption and conservation. As people bask in the conveniences powered by electricity, the demand for sustainable practices becomes increasingly evident. In an era of rapid technological advancement, where innovation often comes with a hefty energy cost, embracing judicious energy use is not just a choice but a necessity. So, as people traverse through the everchanging landscape of electrical progress, a call to prioritise sustainability and conservation echoes in the background, guiding us toward a more responsible electrified future.

The demand for energy has been on a steady rise, with notable increases on power consumption observed annually. Large energy consumers witnessed a 17% surge in consumption from 2020 to 2021, and a substantial 80% increase spanning the period from 2015 to 2021. This upward trajectory is indicative of a growing reliance on energy resources, reflecting both expanding urban residential needs, which accounted for 21% of the total metered electricity consumption in 2021, and rural residential requirements, contributing to 12% of the overall consumption during the same year (CSO, 2022). These escalating consumption trends emphasise the imperative for effective energy management and sustainable practices such as IOT devices to address the rising demand for electricity.

## What is Internet of Things (IoT)

The Internet of Things (IoT) has revolutionised the way we interact with the world, and at the heart of this transformation are IoT sensors. These sensors are sophisticated devices designed to collect and transmit data from the physical world to the digital realm, enabling seamless connectivity and intelligent decision-making (Mouha, 2021). In today's world, IoT sensors are ever-present, embedded in various applications such as smart homes, industrial machinery, healthcare systems, and environmental monitoring. Their effectiveness lies in their ability to provide real-time, detailed data, allowing for precise monitoring, analysis, and control of diverse processes. This real-time insight not only enhances operational efficiency but also facilitates proactive responses to changing conditions. From optimising energy usage to improving health by way of a wearable device, IoT sensors play a pivotal role in shaping a smarter and more connected world, driving innovation and efficiency across industries (Nižetić et al., 2020).

### 2.2.1. How does IoT Function?

The Internet Of Things (IoT) operates on a network of interconnected devices, enabling them to communicate and share information seamlessly. At the core of IoT functionality are sensors embedded in devices, ranging from everyday objects to industrial machinery (Gokhale et al., 2018). These sensors are designed to capture real-world data, such as temperature, humidity, motion, or other environmental factors. The collected data is then processed locally or transmitted to a centralised system via communication protocols like Wi-Fi, Bluetooth, or cellular networks (Krishnamurthi et al., 2020).

IoT devices communicate with each other through various types of networks. The specific network protocols and technologies used in IoT communication can be seen in the list below:

1. **Wi-Fi**: Much like laptops, smartphones, and other conventional computing devices, a lot of IoT devices connect to the internet and communicate with one another through Wi-Fi networks.
2. **Bluetooth:** Short-range communications between IoT devices, like tying a smartphone to a smartwatch or connecting sensors in a particular vicinity, is frequently accomplished via Bluetooth.
3. **RFID:** RFID technology is used to track and identify individuals or objects. It is frequently used in IoT applications for supply chain management, access control, and asset tracking.
4. **MQTT:** MQTT is a low-bandwidth, high-latency, or unstable network messaging protocol that is both lightweight and effective. It is frequently used in IoT to facilitate communication between servers and devices.
5. **Cellular network:** Traditional cellular networks like 4G LTE or 5G can be used by IoT devices for communication. This is especially helpful for Internet Of Things devices that need to be mobile and have wide coverage areas. (Triantafyllou et al., 2018)

This communication allows devices to share information, coordinate actions, and respond to changes in their environment. For instance, in a smart home, temperature sensors may communicate with thermostats to adjust the climate, while motion sensors might trigger security cameras or turn on lights (Vila et al., 2023).

In summary, IoT devices work by leveraging sensors to gather data from the physical world, processing and interpreting that data, and then communicating with other devices or systems to create intelligent and responsive networks. This interconnectedness allows for the creation of smart environments, where devices collaboratively enhance efficiency, convenience, and decision-making processes.

## Raspberry Pi Vs 4 Channel ESP-8266 Mains Current Sensor

### 2.3.1. Raspberry Pi

Developed by the Raspberry Pi Foundation, the Raspberry Pi is a credit card-sized single-board computer. This highly adaptable and reasonably priced computing device boasts GPIO pins, USB ports, memory, processing power, and HDMI outputs, making it suitable for a variety of tasks. Since its inception, the Raspberry Pi has gained popularity among professionals, DIY enthusiasts, and hobbyists. Originally designed to support computer science education and provide an accessible platform for learning programming, the Raspberry Pi has become a go-to solution for tasks ranging from embedded systems and home automation to educational programming, thanks to its versatility and ability to run multiple operating systems and applications. (BasuMallick, 2022).

Despite the undeniable strength and adaptability of the Raspberry Pi, this research brought a crucial limitation to light in that the Raspberry Pi With only a single I/O port, proves unsuitable for the project, considering the typical household includes several circuits.

### 2.3.2. 4 Channel ESP-8266 Mains Current Sensor

The 4 Channel ESP8266 Mains Current Sensor shown in Figure 1 is an IoT monitoring system designed to measure electrical current in mains circuits. It employs the ESP8266 Wi-Fi module for data acquisition from four independent channels, each monitored through CT clamps that encircle the conductors carrying current. The 3.5mm headphone-style jack provides a standardised connection for the CT clamps. The ESP8266 likely facilitates data transmission over Wi-Fi, allowing remote monitoring and logging of electrical currents. This device is valuable in industrial settings, smart home energy management, or any application requiring comprehensive power consumption insights. For specific details, consult the manufacturer's documentation.

The primary function of the ESP-8266 microcontroller in this context is to digitise the analogue signals emitted by each CT clamp. These analogue signals, representative of electrical current measurements, are converted into digital form, making them comprehensible and processable by computers. The integration of the D1 Mini, a development board engineered by WeMos, augments the capabilities of the ESP-8266. Displayed in Figure 1, the D1 Mini serves as an intermediary, streamlining both the programming process and the interaction with the ESP-8266. This addition not only enhances the overall functionality of the ESP-8266 but also significantly eases its usability, providing a more efficient and user-friendly experience in handling and manipulating the data from the CT clamps.

A blue circuit board with many small round metal components

Description automatically generated with medium confidence

Figure 1. 4 Channel ESP-8266 Mains Current Sensor

(“Mottramlabs-Products,2023”).

## Wemos D1 Mini

The Wemos D1 mini is a compact development board shown in Figure 2 for the ESP-8266 wireless microcontroller, providing seamless integration with the Arduino platform. This miniature board transforms the ESP-8266 into a comprehensive development environment. Programming the D1 mini is straightforward, like programming other Arduino-based microcontrollers. The module incorporates a built-in micro-USB interface, enabling direct programming from the Arduino IDE without the need for extra hardware. (Mohan, 2020).

A small blue circuit board

Description automatically generated

Figure 2. D1 Mini NodeMcu

(Boyce, 2017).

## MQTT Protocol

MQTT is an acronym for Message Queue Telemetry Transport, a lightweight messaging transport commonly employed in scenarios demanding a minimal code footprint. This protocol is particularly useful in situations where there are constraints on code size or network bandwidth, which is often the case with small devices, such as those used in home automation. (Yokotani and Sasaki, 2016).

In the MQTT protocol, message transmission follows a publish/subscribe model. The sender publishes a message on a specific topic without prior knowledge of the subscribers (receivers). Subscribers, in turn, receive messages related to the topics they have subscribed to. An advantageous feature of MQTT is its utilisation of an MQTT broker, facilitating one-to-many communication. The broker receives, filters, and disseminates messages to all subscribed clients. The preference for MQTT over HTTP is attributed to its commitment to a high delivery rate, offering three distinct quality-of-service levels. Notably, MQTT boasts a speed advantage, being 93 times faster than HTTP. (Chaudhary et al., 2018).

## Arduino IDE vs Home Assistant

### 2.6.1. Arduino IDE

The Arduino Integrated Development Environment (IDE) is a software application used for writing and uploading code to Arduino-compatible boards. It provides an accessible, user-friendly interface for programmers of all skill levels to create and compile code. The IDE supports various Arduino board models and includes a text editor for writing code, a message area, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them. The IDE's simplicity, coupled with its comprehensive library of pre-written code, makes it an excellent choice for developing a wide range of electronic and IoT projects.(adnanaqeel, 2018).

### 2.6.2. Home Assistant using ESP.

Home Assistant is an open-source home automation platform that enables the integration and control of a wide range of smart home devices. When used with ESP (Espressif Systems Processor). devices, like the ESP-8266 or ESP32, Home Assistant becomes a powerful tool for creating customised smart home solutions. It allows users to automate tasks, manage devices, and create complex scenarios for home automation. ESP devices, known for their Wi-Fi capabilities and low cost, can be programmed to communicate with Home Assistant, providing a flexible and scalable approach to building smart home systems. This combination offers enthusiasts and developers a platform to innovate and expand their smart home capabilities beyond standard offerings. (Beginners, 2023).

### 2.6.3. IDE choice

The decision to utilise the Arduino Integrated Development Environment (IDE) over Home Assistant for this project was driven by several key factors, primarily its simplicity and the extensive range of built-in libraries available. Arduino IDE's user-friendly interface and comprehensive library support streamline the development process, particularly for projects involving microcontrollers like the ESP-8266. These libraries offer a wealth of pre-written code, significantly reducing development time and complexity.

While Home Assistant is a robust platform with impressive built-in features, including the ability to create complex graphs and manage home automation effectively, it does not align perfectly with the project's requirements. For this specific project, the need for such advanced features was minimal, as the data visualisation and graphing functionalities are being externally handled by the .NET MAUI application. The .NET MAUI application is tailored to process and display the data collected by the system, rendering Home Assistant's advanced graphing capabilities redundant in this context.

Additionally, implementing Home Assistant would require setting up a virtual machine (VM) to host the software, introducing additional layers of complexity and potential maintenance challenges. The requirement of a VM for running ESP Home could complicate the setup for users who are not as technically inclined, moving away from the project's goal of simplicity and accessibility.

In summary, the Arduino IDE was chosen for its ease of use, comprehensive library support, and straightforward approach, aligning well with the project's goal of developing a user-friendly and efficient energy monitoring system. The choice to forego Home Assistant's advanced features in Favor of a simpler, more focused approach using .NET MAUI for external data handling ensures a streamlined and accessible experience for the end-users.

## Essentials of Electricity: An Overview

Toward the end of the 19th century, a period marked by remarkable scientific advancements, the world was on the brink of transformative changes. The prospect of Telephones and Car’s promised to revolutionise transportation, while electric power was becoming increasingly integrated into homes. Surprisingly, despite the rapid progress, electricity remained somewhat shrouded in mystery for scientists of the time. The turning point came in 1897 when the existence of electrons was unveiled, shedding light on the elusive nature of electricity. This discovery marked the starting point of a deeper understanding of the fundamental particles driving electrical phenomena, propelling science and technology into a new era. (Marshall Brain et al, 2023).

Using these electrons, scientists could gain a deeper understanding of electricity discovering that the way in which electrons are transported is by way of conductive material such as metal. Electrons are simply being passed through a conductive material to get from one place to another. The electrical force that “pushes” of electrons is better known as Voltage. Current refers to the quantity of electrons that move through a conductive material. (Rao, 2023).

### 2.7.1. Types of Current

There are two basic forms of electrical voltage: direct current (DC) and alternating current (AC), each with unique properties. In DC, a steady one-way flow of electrons keeps the voltage constant. Simple electronic devices and batteries frequently contain this kind of current. However, AC produces a waveform by reversing the direction of electron flow on a regular basis. The electricity that is supplied to homes and businesses is known as alternating current (AC), which can be effectively transferred over long distances by using transformers to change the voltage levels. Although DC is recognised for its stability, AC is more versatile for wide distribution and a wider range of applications due to its capacity to change voltage. (Rose, 2023).

### 2.7.2. Electricity in The Home

Electricity in the Home is managed and distributed through a central hub known as the fuse board, or more commonly in modern homes, the distribution board. This board is the primary point of control and safety for the household electrical system. It typically houses multiple circuit breakers, or Miniature Circuit Breakers (MCBs), each responsible for regulating the flow of electricity to different sections or appliances within the home. These MCBs are crucial as they automatically shut off the electrical supply when they detect an overload or a short circuit, thereby preventing potential hazards like electrical fires. The electricity from the main supply is first channelled through a main switch in the distribution board and then distributed to various circuits. Each circuit is connected to a different set of electrical fixtures and outlets through an array of cables, typically made of copper or aluminium, due to their excellent conductivity. These cables are insulated and sheathed for protection against external damage and to ensure safety. The entire system is designed to efficiently distribute electricity throughout the home, while simultaneously providing a high level of safety by isolating electrical faults and preventing disaster. See Figure 3 for image of modern MCB’s. (Consumer World, 2023).

A circuit breaker with instructions

Description automatically generated

Figure 3. Schneider MCB, with load cable terminal.

(RS, 2023).

### 2.7.3. Ohm’s Law

Ohm's Law, a cornerstone of electrical circuit theory, summarises the relationship between voltage, current, and resistance. The electrical potential difference that drives electrons through a circuit is known as voltage, and it is frequently represented by the letter V. The flow of electrons in response to this potential difference is represented by the symbol I, which stands for current. The symbol R denotes resistance, which is expressed in ohms and obstructs the flow of current. This relationship can be expressed mathematically as V = I \* R, where voltage is equal to the product of current and resistance. This is known as Ohm's Law. To put it another way, in a circuit, the current is inversely proportional to the resistance and directly proportional to the voltage. Ohm's Law provides a fundamental framework for understanding and forecasting the behaviour of electrical circuits. By manipulating these variables, one can analyse, design, and optimise electrical systems for various applications. (Sheikh, 2020).

Referring to Figure 4, the diagram illustrates Ohm's Law in a straightforward manner. This visual aid simplifies the process of calculating resistance (R). For a researcher seeking to determine the value of resistance, the method involves a simple calculation: by covering the 'R' in the diagram, the formula becomes evident – multiply voltage (E) by current (I) to find resistance, represented as R = E \* I. This technique can be adapted flexibly for different calculations within Ohm's Law, ensuring ease of use for various electrical measurements.

A diagram of a triangle with a red x and black text

Description automatically generated

Figure 4. Ohms Law Triangle Technique

(“All About Circuits, 2023.)

## Methods of Measuring electricity

The exploration of methods for measuring electricity spans a historical journey from early innovations to contemporary technologies. This section delves into the evolution of techniques employed in quantifying electrical consumption, offering insights into the milestones that have shaped the field. From the rudimentary approaches of the past to the sophisticated methods utilised in the present day, understanding the progression of electricity measurement methods is fundamental to contextualising the current landscape and illuminating the path toward future advancements.

Thomas Edison (1847-1931), a renowned American inventor and businessman known for creating the electric light bulb and phonograph, was the first to devise a method for measuring electricity consumption. His innovation was driven by the need to accurately bill his customers for electricity usage, leading to the development of the earliest electrical measuring device. As illustrated in Figure 5, this initial device comprised two copper rods immersed in a solution. As electricity was consumed, one of the copper electrodes would gradually dissolve, indicating the amount of electricity used based on the mass lost. However, this method lacked precision, leading to dissatisfaction, and paving the way for the development of more accurate alternatives in electrical measurement (International, 2006).

A black and white photo of two metal containers

Description automatically generated

Figure 5.Thomas Edison – Chemical Meter made in 1882.

(American History Museum, 2023).

### 2.8.1. Recent method of measuring electricity

Figure 6, a familiar sight meets the eye a standardised meter, commonly known as a Mechanical or spinning disk meter. This device provides a clear display of electricity consumption in kilowatt-hours (kWh). (“ESB Networks, 2023”).



Figure 6. 24-hour Mechanical Meter.

(“ESB Networks, 2023”).

Mechanical electric meters operate using two conductor coils, each playing a distinct role in measuring electricity usage. The first coil responds to the voltage across the conductor, while the second coil is influenced by the current flowing through it. These coils generate magnetic fields that cause a thin aluminium disc to rotate at a speed proportional to the electricity consumption. This rotation directly influences the meter's dials, which display the total electricity used in kilowatt-hours.

For billing purposes, a representative from the electricity provider typically needs to visit the site to manually read and record these meter readings. However, consumers also have the option to monitor their electricity usage independently. By reading the meter themselves, they can verify the accuracy of utility charges and keep track of their electricity consumption (Palmetto, 2023).

## Current Transformer Clamp

A non-invasive sensor called a CT clamp shown in Figure 7, also known as a current transformer clamp, can measure electric current flowing through a conductor without requiring physical contact. Usually, it has a hinged jaw as shown in Figure 7, that makes it simple to clamp around a live wire. The CT clamp works based on electromagnetic induction, in which a voltage is induced in the clamp's coil by the magnetic field created by the current passing through the conductor (Greg, 2017).



Figure 7. CT Clamp (Current transformer clamp).

(Amazon, 2023a).

To measure current in real-time without disturbing the electrical circuit, CT clamps are frequently used in electricity monitoring.(Evenergy, 2023). Applications such as load profiling, power quality evaluations, and energy audits derive substantial advantages from the use of current transformer (CT) clamps. Load profiling involves monitoring and analysing the electrical consumption patterns of a system over time. The CT clamp, which encircles either the live or neutral wire, provides precise current measurements critical for load profiling.

Simultaneously, power quality assessments focus on evaluating the stability and reliability of the electrical supply. By utilising CT clamps, users can gain insights into potential issues such as overloads or irregularities in the electrical system.(Ciontea, 2021). This non-invasive feature allows both professionals and enthusiasts to observe power usage trends, identify problems, and make informed decisions for optimising systems and enhancing energy efficiency. Whether conducting load profiling or assessing power quality, CT clamps offer a convenient and secure method to obtain valuable information about electrical systems. As the research discovers non-invasive electricity monitoring, like CT clamps for real-time current measurement, the need for effective data storage and management becomes evident. This transition prompts the discussion of cloud databases, which play a vital role in securely storing and accessing the valuable information collected through devices like CT clamps.

## What is a cloud database?

A cloud database refers to a database service that is hosted and managed on a cloud computing platform, allowing users to access and interact with their data over the internet. It offers the capability to store, manage, and retrieve data in a scalable and flexible manner without the need for physical infrastructure or on-premises hardware. (IBM, 2023).

In a cloud database, the data is stored in a distributed manner across multiple servers and locations within the cloud infrastructure. This distributed nature enhances reliability, scalability, and accessibility. Users can access the database using the internet, making it suitable for businesses, applications, and services that require seamless and remote data management. (Google Cloud, 2021).

Relational databases in the cloud comprise one or more tables made up of columns and rows, providing a structured framework to logically organise data with predefined relationships. These databases adhere to a fixed data schema and utilise Structured Query Language (SQL) for querying and manipulating data. Known for their high consistency and reliability, relational cloud databases excel in managing large volumes of structured data. (Oracle, 2022). Oracle, MySQL, Microsoft SQL Server, and PostgreSQL are notable examples of relational databases. (Mitiku, 2022).

Non-relational cloud databases are tailored for handling unstructured data, which includes various formats such as emails, text messages, documents, surveys, multimedia files, and sensor data. These databases are characterised by their flexibility in data organisation, as they do not require a strict schema. This attribute allows users to store and manage data in diverse formats without conforming to a predetermined structure. Examples of non-relational databases include MongoDB, Redis, Cassandra, and HBase, each offering unique capabilities for managing unstructured data efficiently (Pattinson, 2022).

Cloud databases provide several advantages, as shown in table 1.

Table 1. Advantages of Cloud databases

|  |  |
| --- | --- |
| **Advantage** | **Description** |
| Scalability | Cloud databases can adapt to varying application needs, scaling up or down for storage and performance. |
| Accessibility | They offer remote access from any location with internet connectivity, supporting collaborative work and data management. |
| Cost-Efficiency | Utilising a pay-as-you-go model, these databases are often more economical than maintaining physical infrastructure. |
| Reliability | Providers usually have redundant systems and backup solutions, ensuring data safety and reducing data loss risks. |
| Managed Services | Many cloud database services include maintenance, updates, and security management, easing operational tasks for users. |

(“Google Cloud,” 2021).

Cloud databases offer many benefits, they also have some drawbacks, as shown in Table 2.

Table 2. Drawbacks of Cloud Databases

|  |  |
| --- | --- |
| **Disadvantages** | **Description** |
| Security Concerns | Cloud databases face security challenges due to the remote storage of sensitive information. Issues such as data breaches, unauthorised access, and potential vulnerabilities in cloud infrastructure pose significant risks to the confidentiality and integrity of stored data. |
| Downtime and Reliability | Reliability can be a drawback for cloud databases, as they are susceptible to downtime, impacting accessibility. Factors such as network outages, server failures, or maintenance activities by the cloud service provider can lead to temporary unavailability, affecting the continuous operation of applications reliant on the database. |

(Obrutsky, 2016).

## Comparing Cloud Databases

Selecting an appropriate database is essential for efficient application development and smooth data processing in the field of modern data management. This section compares Firebase, SQLite, and ThingSpeak, three different cloud databases. These databases all have distinct features that meet various needs for data management. It is critical to comprehend the fundamentals and important differentiators that shape their roles in the ever-changing landscape of cloud-based data storage as we delve into their benefits in the text that follows. The hope is that this comparative analysis will help in finalising the database selection for this project.

### 2.11.1. Firebase

Firebase is a valuable option for developing web applications, offering the advantage of not having to deal with infrastructure management. Firebase is owned and maintained by Google. Before we get into the Pros and Cons of firebase its worth mentioning some key services. Such services include Authentication this is for sign up and sign in. Cloud Storage this is perfect for storing images or files. Cloud Functions this offers a saleable solution to process data and call many other actions. As you can see in Table 3 there’s a list of advantages and disadvantages to help decide. (Bogdan, 2021).

Table 3. Critique of Firebase

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Wide range of Database capabilities | Little querying capabilities |
| Vast range of services and features | Limited Data migration |
| Free basic plan | Platform Dependent |
| Popular, meaning great documentation online | Suited towards Android |
| Great for beginners, easy setup and integrated | Less support for IOS or Mac |

(Bogdan, 2021).

### 2.11.2. SQLite

Users can store and retrieve data in a structured format using SQLite, a cloud-based relational database management system. Because SQLite is serverless and runs directly on the device, as opposed to traditional databases, it is a popular option for embedded and mobile applications. It is a lightweight, extremely efficient, zero-configuration, self-contained database engine. Because of its well-known simplicity, usability, and compatibility with a wide range of programming languages, SQLite is a flexible option for resource-constrained or locally stored application needs. (SqLite, 2023).

View the numerous advantages and disadvantages of SQLite below in table 4.

Table 4. Critique of SQLite

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Serverless Operation | Limitations in terms of data table size |
| Zero Configuration | No built-in data encryption methods/ techniques |
| Cross-Platform Compatibility | Not suitable for large applications |
| Lightweight | Not multiuser friendly |
| Self-contained | The transaction moves slowly. |

(Bogdan, 2021).

## Comparing Cross platform technologies

### 2.12.1. What is .NET MAUI?

.NET MAUI (Multi-platform App UI) is an open-source framework that streamlines the development of cross-platform mobile and desktop applications by providing a single codebase for iOS, Android, Windows, macOS, and more. Using the power of C# and the .NET ecosystem, it allows developers to create adaptive user interfaces using XAML, promoting the Model-View-ViewModel (MVVM) pattern for efficient app development across multiple platforms. (Microsoft, 2023).

**Strengths:**

* Native-like Performance: .NET MAUI leverages the native user interface controls, resulting in excellent performance on both iOS and Android devices. It provides a truly native experience.
* Code Sharing: The user interface, business logic, and data models can all be shared across platforms usually thanks to.NET MAUI. This can cut down on effort and time spent developing.
* Integration with C and .NET: Since C is the main programming language used by NET MAUI, it is simpler to locate knowledgeable developers and utilise pre-existing codebases.
* XAML for UI: Using XAML for UI design makes it easier to design and modify the application's user interface by offering a versatile and expressive method to do so.(Matos, 2023).

**Weaknesses:**

* Learning Curve: There might be a learning curve for.NET MAUI for developers who are unfamiliar with C and.NET.
* Restricted Ecosystem: Although.NET MAUI is a promising framework, its ecosystem (third-party libraries, plugins) may not be as large as that of other options due to its recent relative newness. (Singh, 2023).

### 2.12.2. Flutter

Flutter is a revolutionary app development tool that fundamentally alters the way you create apps. Flutter makes it simple to develop, test, and run stunning desktop, mobile, and embedded applications all from a single codebase.

Google created the open-source Flutter framework, which enables programmers to create visually stunning and remarkably effective natively compiled cross-platform applications. Its reputation as a dependable technology backed by Google and embraced by well-known global brands is evidence of its versatility and dependability.

What makes it so amasing is its collaborative nature, which is maintained by a wide community of dedicated developers from all over the world. It assures continued development and expansion. (Wu, 2018).

**Strengths:**

* Single Codebase: Flutter does a great job of minimising development and maintenance work by implementing a single codebase for both iOS and Android.
* Rich Widget Set: Flutter provides an extensive collection of modifiable widgets, which leads to a UI that is visually appealing and consistent on all platforms.
* High Performance: Near-native app experiences are made possible by Flutter's architecture and compiled code.
* Language: The Dart language has a simple syntax and is easy to learn, despite not being as well-known as JavaScript. (Bhanderi, 2023).

**Weaknesses:**

* Smaller Community: Flutter has a growing but smaller community compared to React Native, which may lead to fewer resources and plugins.
* Few experienced Flutter developers: There are still relatively few experienced Flutter developers on the market. That’s why it can be harder to build a team of seasoned professionals if you choose Flutter over React Native.
* Large App Size: Flutter apps can have larger file sizes compared to native or other cross platform solutions. (Bhanderi, 2023).

### 2.12.3 React Native

Facebook created the open-source React Native framework to help developers create cross-platform mobile applications with React and JavaScript. It offers a practical and effective way to build applications while providing a native look and feel using native components, enabling developers to create mobile apps that run on both the iOS and Android platforms with a single codebase. Its "Hot Reload" feature speeds up the development process, and a thriving community and abundance of libraries add to its appeal as a flexible mobile app development solution. (Boduch, 2017).

**Strengths:**

* JavaScript: React Native uses JavaScript, a widely known and accessible language, which can attract a large pool of developers.
* Great Ecosystem: React Native boasts a thriving ecosystem of libraries and plugins that can accelerate development and provide access to a wide range of native device features.
* Live Reload: Developers can take advantage of the "hot reloading" feature to see changes instantly, speeding up the development process. (Dekkati et al., 2019).

**Weaknesses:**

* Performance: While React Native offers good performance, it may not match the native feel provided by .NET MAUI due to the JavaScript bridge that communicates between JavaScript and native code.
* Platform Specific Code: For certain features or optimisations, you may need to write platform specific code in native languages (Java/Kotlin for Android, Swift/Objective C for iOS).
* UI Consistency: Achieving pixel perfect UI consistency across platforms can be challenging, as UI components may render differently on iOS and Android. (Dekkati et al., 2019).

## Similar Projects

### 2.13.1. Design of an IoT Energy Monitoring System (Chooruang and Meekul, 2018).

My project, titled "Smart Household Energy Monitoring and Management: A Comprehensive IoT-Based Solution for Sustainability," distinctively differs from the system developed by Komkrit Chooruang and Kraison Meekul in several critical aspects. A standout feature of my project is its capability to simultaneously monitor four different circuits within a household. This is achieved through the integration of a 4 Channel ESP-8266 Mains Current Sensor, coupled with an ESP-8266 Wi-Fi Module and CT Clamps. This multi-circuit monitoring not only enhances the system's depth in analysing energy consumption but also provides homeowners with a comprehensive understanding of their energy usage across various sections of their household.

Additionally, my project is designed with universal compatibility in mind, featuring a cross-platform application that functions seamlessly across a diverse array of devices, including Android, iOS, and PCs. This broad compatibility ensures that the system is accessible to a wide user base, thus democratising the monitoring and management of household energy.

In contrast, Chooruang and Meekul’s project pursues a broader application scope. Their system, based on the PZEM-004T energy meter and the ESP-8266 Wemos D1 mini microcontroller, is tailored more towards general applications such as smart grids and home automation, particularly in resource-limited settings of developing countries. However, it does not specifically address the nuanced needs of multi-circuit household monitoring or the cross-platform accessibility my project offers.

Furthermore, while both projects employ MQTT for efficient data communication, Chooruang and Meekul extend their system’s capabilities with additional tools like Python, Influx DB, and Grafana, which are pivotal for robust server-side data processing and visualisation. In summary, my project carves out its unique position by focusing on detailed, user-friendly energy management at the household level, offering a highly accessible, multi-platform solution that provides granular insights into domestic energy consumption.

### 2.13.2. OWL Micro+ - Simple Energy Monitoring. (OWL Micro+, 2023).

My project and the OWL Micro+ Energy Monitor product both aim to provide insights into household energy usage. A key difference is in the user interface and data accessibility. The OWL Micro+ features a dedicated wireless display for real-time information, but it lacks the capability for online monitoring or access via mobile phones and desktops.

In contrast, this project employs a cross-platform application accessible on various smart devices, allowing users to monitor and manage their energy consumption remotely. Additionally, this system offers the ability to monitor multiple circuits simultaneously, providing a more detailed overview of household energy use. This multi-circuit monitoring and the flexibility of remote access through mobile and desktop platforms make my project a more versatile and integrated solution for contemporary users. See Figure 8 for product.



Figure 8. ONL Micro+ Product Pictures

(Amazon, 2023b).

## Conclusion

This review provided an educative journey exploring IoT sensors, electricity measurement methods, and cloud databases. Gaining knowledge about historical milestones like Thomas Edison's work and modern advancements like the 4 Channel ESP-8266 Mains Current Sensor.

Researching about cloud databases found that while they offer scalability and accessibility, security concerns and downtime are drawbacks. The comparison of Firebase, SQLite, and ThingSpeak guided the project to utilise Firebase and the 4 Channel ESP-8266 Mains Current Sensor.

Examining cross-platform technologies, Comparing .NET MAUI, Flutter, and React Native. The project was more suited for .NET MAUI for its native-like performance.

This literature review has been a learning experience, revealing challenges and solutions. It equipped the author with a solid understanding of IoT, electricity measurement, and cloud databases. Decisions made choosing the 4 Channel ESP-8266 Mains Current Sensor and Firebase combine theoretical insights with practical choices, setting the stage for a successful project. The path forward holds both challenges and succusses, and armed with the knowledge gained, I'm ready to navigate it confidently and creatively.

# Design

# 3.1. Introduction

In this design chapter, the integration of the 4 Channel ESP-8266 Mains Current Sensor with Firebase is expanded upon, building upon the historical studies explored in the previous chapter. The objective here is to provide a detailed exploration of the design features, covering both functional and non-functional requirements, to craft a versatile, cost-effective, and intelligent electricity monitoring solution.

The hardware and software requirements are crucial aspects of the system design. In terms of functional requirements, specific attention is given to the role of the 4 Channel ESP-8266 Mains Current Sensor in monitoring electricity consumption. Additionally, the software requirements are outlined, with a focus on Firebase as the selected platform for data storage and retrieval.

Non-functional requirements play a significant role in ensuring the system's robustness and adaptability. The discussion encompasses considerations for system reliability, scalability, and performance, considering the potential expansion of the monitoring system. Security requirements are also addressed to safeguard data integrity and user privacy.

In the subsequent sections, the design features are further elucidated. Hardware and software requirements are detailed, encompassing not only the functional aspects but also the non-functional dimensions critical for system performance. Diagrams, including dataflow diagrams, use case diagrams, and process flowcharts, are incorporated to visually represent the intricate relationships and processes within the system, enhancing the clarity and understanding of the proposed design.

## 3.2. Hardware Requirements

This project focuses on developing a cost-effective and efficient energy monitoring system. Initially, we considered using the Raspberry Pi for its affordability, but our research revealed its limitations, notably its single I/O port, which is inadequate for monitoring multiple circuits in a household. To address this, we shifted to integrating a multi-I/O device, which better suits comprehensive home monitoring needs.

A crucial aspect of this project is simplifying the installation process to enhance end-user accessibility. After thorough research, we chose the ESP-8266 and D1 Mini for their compatibility and user-friendly design. These components are specifically selected for their straightforward connection, eliminating the need for complex wiring or soldering. This approach aligns with our commitment to providing a user-friendly and accessible energy monitoring solution. For a detailed list of the hardware components and their functionalities used in this project, please refer to Table 5.

Table 5. Hardware Requirements

|  |  |  |
| --- | --- | --- |
| **Hardware** | **Description** | **Cost** |
| 4 Channel Mains current sensor | A 4-channel mains sensor board that can be equipped with an ESP-12 Wi-Fi module or a Wemos D1 Mini. There are four 3.5mm Jack sockets included for connecting directly to the well-liked SCT-013 current transformer. | £8.99 via eBay |
| Wemos D1 mini | The Wemos D1 mini is a wireless 802.11 (Wi-Fi) microcontroller development board in a small package. It converts the widely used ESP-8266 wireless microcontroller into a full-featured development board. | £7.10 via Amazon |
| SCT-013 Current transformer | The SCT013 sensors are current transformers, which are instrumentation devices that measure the intensity with which a circuit crosses. Electromagnetic induction is used to make the measurement. | £15.97 Amazon |

### 3.2.1 Connecting the CT clamp to ESP-8266

Illustrated in Figure 9 is the straightforward wiring configuration that will be followed to establish a seamless connection between the CT clamp and the 4-channel mains current sensor. This design embodies a plug-and-play mechanism utilising a 3.5mm headphone style jack, ensuring a user-friendly experience for individuals with varying technical expertise. The simplicity of the setup streamlines the implementation process, making it accessible to a broad user base. All that remains is the insertion of the D1 Mini into the designated 3.5mm slot, completing the integration effortlessly.

A close-up of a circuit board

Description automatically generated

Figure 9. Connecting the CT clamp to ESP-8266 via 3.5mm connection

## 3.3. Software Requirements

To facilitate the functionality of the energy monitoring system, specific software requirements have been identified. Drawing insights from existing studies in the field, the selection of software components was guided by a thoughtful consideration of various factors. For example, preference played a role in choosing certain software, such as opting for a cloud database and leveraging .NET MAUI for its user-friendly interface and broad accessibility, thanks to the cross-compatible nature inherent in .NET MAUI. This meticulous software selection ensures not only functionality, but also user ease of use. Some software choices such as Arduino IDE were restricted due to the communication to the ESP-8266.

A detailed overview of the software requirements integral to this project, are outlined in Table 6.

Table 6. Software Requirements

|  |  |  |
| --- | --- | --- |
| **Software (Version)** | **Description** | **Cost** |
| Arduino IDE | The Arduino Software (IDE) makes it simple to write code and upload it to the board even while not connected to the internet. (Documentation, 2023). | Opensource |
| Firebase | The Firebase Realtime Database is a NoSQL database hosted in the cloud that allows you to store and sync data between your users in real time. (Firebase, 2023). | Offers 1GB of storage for free, More than enough for this project. |
| .NET MAUI | (.NET MAUI) is a cross-platform framework that uses C# and XAML to create native mobile and desktop apps. (Microsoft, 2023). | Opensource |
| C# | Programming language | N/A |
| XAML | Programming language | N/A |
| C++ | Programming language | N/A |

## 3.4. Functional and Non-Functional Requirements

A functional requirement is a specific and detailed description of the intended capabilities, features, and interactions that a system or product must possess. These requirements outline the functional aspects, behaviours, and performance expectations, specifying what the system is supposed to do and how it should respond to various inputs or conditions. Functional requirements serve as a blueprint for the development team, providing a clear roadmap for building a system that meets the desired functionalities and user expectations. (Heralgi, 2019).

Non-functional requirements describe the qualities and characteristics that define how a system should perform rather than what it should do. These requirements encompass aspects such as performance, reliability, security, usability, and scalability. Unlike functional requirements, which focus on specific features and behaviours, non-functional requirements provide criteria for evaluating the overall effectiveness and efficiency of the system. For example, non-functional requirements may specify response times, system availability, or security measures. Addressing non-functional requirements is crucial for ensuring that the system not only meets functional expectations but also delivers a satisfactory user experience and adheres to broader performance and quality standards. (Heralgi, 2019).

### 3.4.1. Functional Requirements

* **FR1**: CT Clamp must be encircling the live or neutral cable.
* **FR2:** System must be able to measure the amperage from the cable.
* **FR3:** Real-time monitoring of electricity is a must for this system.
* **FR4:** The system must be able to send amperage to firebase the cloud database.
* **FR5:** .NET Maui application must be able to fetch the data from the cloud database.
* **FR6:** The system must be able to display the real-time electricity usage in the form of kilowatt hours for the user.
* **FR7:** A user must be able to set a usage threshold.
* **FR8:** The system must notify the user if approaching the set limit of electricity consumption.
* **FR9:** The system should provide historical information on electricity consumption.

### 3.4.2. Non-Functional Requirements

* **NFR1**: The system should be able to accurately measure electricity consumption.
* **NFR2:** The system should be able to perform when not connected to external monitors.
* **NFR3:** The system should send the data to the cloud In a secure manner using Firebase’s built in data encryption using HTTPs.
* **NFR4:** The application itself should also be secured possibly via a login.
* **NFR6:** The system should be cost-effective.
* **NFR7:** The system should have a user-friendly UI.
* **NFR8:** The system should be reliable with minimal downtime.
* **NFR9:** The system should be capable of being Cross compatible.
* **NFR10:** The system should be scalable allowing for more devices in future.

## 3.6. Data Flow

Figure 10 presents a detailed visualisation of the system's data flow for the electricity monitoring process. The process begins with the flow of live current through a designated live cable, typically the brown wire. A current transformer is then conveniently clamped around this wire to monitor the electricity flow. The data collected from this setup is transmitted to the ESP-8266 and D1 Mini for initial processing.

Following this, the processed data is sent to a cloud database. This step ensures both secure storage and the provision of real-time updates. The data, now readily available and updated in the cloud, is accessed by a cross-platform application developed using .NET MAUI. This application serves as the final interface, presenting the comprehensive results of the monitoring process to the end user in a coherent and user-friendly manner.

See Figure 10 for a visual representation of the dataflow of the system.

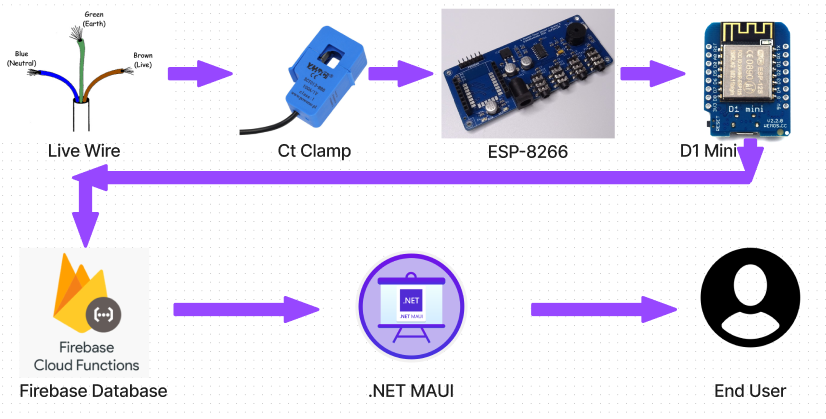


Figure 10. Dataflow of the system

## 3.7. Process Flowchart

A process flowchart is a visual diagram that displays the sequential steps or activities in a process, using standardised symbols to represent tasks and their interconnections. It serves as a tool for analysing and communicating processes, aiding in identifying inefficiencies and improvement opportunities. (IBM, 2021).

Figure 11 features a detailed process flowchart that effectively illustrates the key initiation and termination points of the system. This flowchart methodically traces each step in the process, delivering an in-depth view of how the system progresses from start to finish. Each phase of the flowchart is meticulously outlined, providing a comprehensive understanding of the operational flow. For further clarity, a concise description of each segment within the flowchart is provided below Figure 11, ensuring a thorough comprehension of each stage in the system's workflow.

# 

Figure 11. Process Flowchart- See below for explanations of each stage.

3.7.2. Choose Circuit to Monitor

Circuits in homes are essential for distributing electricity safely and efficiently. They provide organised pathways for the controlled flow of electrical current, connecting power sources to various devices and appliances. Circuits prevent overloading, regulate power distribution, and contribute to the seamless functioning of modern conveniences within a household. Knowing more about circuits allows the user to make an informed decision on which circuit to choose.

### 3.7.3. Clamp CT to Wire

The user effortlessly opens the hinge on the CT clamp, as illustrated in Figure 6, and proceeds to secure it around the live wire of their selected circuit.

### 3.7.4. Power Up the Devices

Powering the devices is a straightforward process, requiring a 12V DC power supply and a micro-USB. The wiring is entirely plug-and-play for user convenience.

### 3.7.5. Read Sensor Value

Next, the system reads the sensor value.

### 3.7.6. Sensor Reading

It essentially translates into a diagrammatic IF statement: if the sensor doesn't output a value, the system prompts the user to go back and verify that the CT clamp is securely clamped onto the live wire.

### 3.7.7. Upload to Firebase

Upon successful sensor value reading, the process proceeds to upload the data to Firebase.

### 3.7.8. Connection Success

This involves another IF-style statement: it checks for the success or failure of the connection and appropriately handles each scenario.

### 3.7.9. Save Results to File

In case of an unsuccessful connection, the results will be saved to a file.

### 3.7.10. Fetch Data from .NET MAUI

Upon a successful connection, the system fetches data from Firebase using a .NET MAUI application.

### 3.7.11. Connection Success

If the data fetching process is successful, the system proceeds to handle it accordingly.

### 3.7.12. Visualise Data

The data is then visualised for the user to view and interact with as they please.

## 3.8. Use Case Diagram

Shown in Figure 12 is a use case diagram, a visual representation often used in system design. While it shares some similarities with a process flow chart, there are key differences between the two. A process flow chart provides a sequential depiction of internal activities and data flow within a process, emphasising the chronological order of tasks.

In contrast, a use case diagram focuses on interactions between a system and external entities, showcasing the system's functionality from a user's perspective. The process flow chart emphasises internal workflow, while the use case diagram captures system behaviour in response to external interactions. Both tools offer valuable insights in system development, each addressing specific aspects of analysis and design. (IBM, 2021).

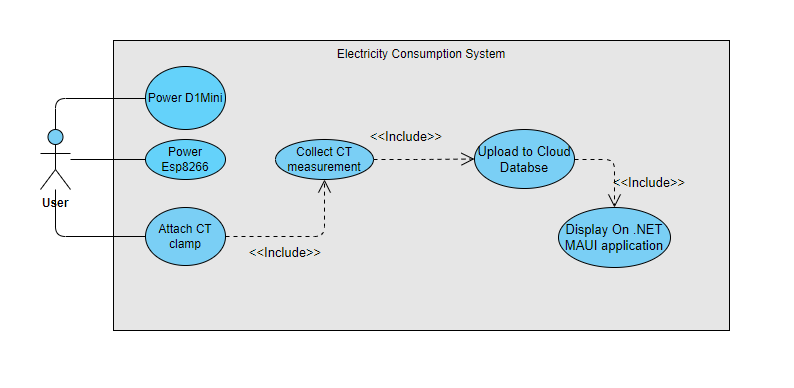


Figure 12 Use case diagram.

## 3.10. UI Design

## A screen shot of a phone Description automatically generatedThe significance of a simple and intuitive User Interface (UI) cannot be overstated, particularly in the context of the energy monitoring system outlined in Chapters 1 and 2. A well-crafted UI serves as the bridge between users and complex technological functionalities. (Toal, 2015) Keeping the UI simple is essential to ensure that users, irrespective of their technical proficiency, can effortlessly navigate and comprehend the system. Simplicity in design minimises the learning curve, reduces the likelihood of errors, and fosters a positive user experience (Singh, 2020). By prioritising simplicity in the UI, the energy monitoring system aims to democratise access to energy data, making it more inclusive and user-friendly for a diverse audience. Figure 13 depicts the Homepage of the application.

Figure 13. Home Page.

### 3.10.1. Flyout navigation of application UI

A screen shot of a phone

Description automatically generatedFor the project navigation a Flyout menu was used, Using a left flyout menu is advantageous for user interface design as it enhances navigation efficiency and keeps the main content unobstructed. Placed discreetly on the left side, this menu provides a streamlined approach, allowing users to access key features or sections with a simple click. It promotes a clean and organised layout, ensuring a seamless user experience by reducing clutter and maintaining focus on the central content. The left flyout menu is an intuitive design choice, offering users quick and convenient access to essential functions while maintaining a visually appealing and user-friendly interface. The implementation of this can be seen in Figure 14 (HoTran, 2019).

Figure 14. Flyout Navigation.

### 3.10.2. Setting Electricity Threshold UI

Setting an electricity threshold is key for saving money and promoting efficiency. A simple user interface is crucial to facilitate easy threshold adjustments, ensuring users can effortlessly manage their energy consumption. The uncomplicated design encourages widespread adoption, empowering users to take control of their electricity usage for both financial savings and environmental sustainability. This can be seen in Figure 15.

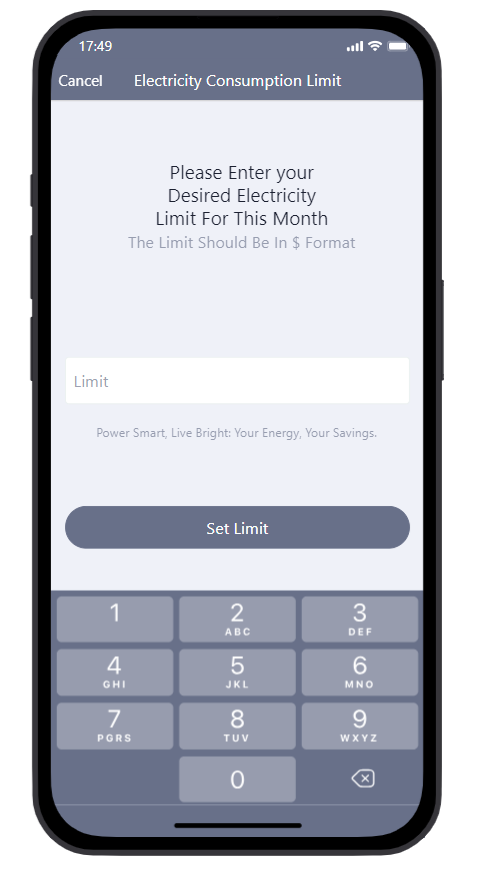


Figure 15. Setting Electricity Threshold.

### 3.10.3. Notifications section

The notifications page keeps users informed about their electricity threshold, providing timely alerts and updates. This feature enhances user awareness, encouraging proactive management of energy usage for cost-effective practices. The implementation of this is shown in Figure 16.

A cell phone with a screen and a price

Description automatically generated with medium confidence

Figure 16. Notification centre to view alerts.

## 3.10. Conclusion

In conclusion, the design chapter outlines a meticulous approach to achieving the project's goals for a cost-effective and user-friendly energy monitoring system. The hardware selection, featuring ESP-8266 and Wemos D1 Mini, emphasises simplicity with a plug-and-play mechanism, eliminating complex wiring. Table 3 provides a detailed list of affordable hardware components. The software choices, including .NET MAUI and Firebase, prioritise functionality and user ease of use (Table 4). Functional and non-functional requirements guide development, ensuring intended capabilities and a positive user experience. Figures 7 and 8 depict the systematic data and process flow, while the UI design (Figures 11, 12, 13, 14) highlights simplicity. In summary, the design chapter establishes a robust foundation for the development of an accessible, affordable, and user-friendly energy monitoring solution.

# 4.0. Implementation

## 4.1. Introduction

Chapter 4 outlines the setup process of the ESP32, a low-cost and low-power system on a chip that is integral to Internet of Things (IoT) applications, renowned for its broad feature set and ease of use. The chapter guides through the hardware configuration of the ESP32, highlighting its user-friendly design that simplifies the connection of sensors and the Wroom Wi-Fi Chip.

On the software front, it details the installation of board packages, port drivers, and essential libraries within the Arduino IDE, necessary for the ESP32 to function correctly. This preparation is crucial for the microcontroller to handle energy monitoring tasks, collecting, and sending data to a Firebase database. The database setup is also covered, showing how it organises energy usage data from multiple circuits for efficient retrieval and analysis, ultimately interfacing with a .NET MAUI application for user interaction.

## 4.2. Setting Up ESP32

### 4.2.1. ESP32

The ESP32 is a series of low-cost, low-power system on a chip (SoC) microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. Developed by Espressif Systems, a Shanghai-based Chinese company, the ESP32 is a successor to the ESP8266 microcontroller. It was launched in September 2016 and has since become popular in the Internet of Things (IoT) projects and applications due to its wide range of features, affordability, and ease of use. The ESP8266 (Hübschmann, 2020; Teel, 2024)

The ESP32 is selected for its integration of 2 SAR (Successive Approximation Register) ADCs, supporting a total of 18 measurement channels (analogue enabled pins), enabling concurrent readings from multiple sensors without extra hardware. This feature is crucial for real-time environmental monitoring. Coupled with its comprehensive capabilities, including Wi-Fi/Bluetooth, high processing power, and low energy consumption, the ESP32 simplifies design and enhances efficient, reliable data acquisition. (Espressif, 2021)

### 4.2.2. Configuring ESP32 Hardware for Mains Current Sensing

The image below showcases the ESP32 Development Board, designed with simplicity in mind to avoid complex wiring and soldering, thus enhancing user-friendliness. This board features four CT (Current Transformer) clamp inputs and a designated area on the right for installing a Wroom WiFi Chip. Notably, it lacks a separate power port, as it receives power through the Wroom chip, which in turn powers both boards. Configuring this board is straightforward; it requires selecting the appropriate pin for sensor readings from the available 18 GPIO (General Purpose Input/Output) pins, simplifying the setup process for various applications. The ESP32 Can be seen below in Figure 17.

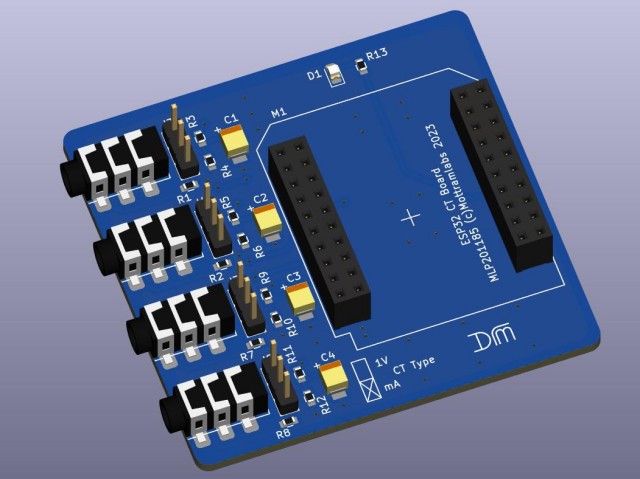


Figure 17 ESP32 CT Board

## 4.3. ESP-WROOM-32

### 4.3.1. ESP-WROOM-32

The ESP32-WROOM-32 module stands as a versatile and potent microcontroller unit (MCU) equipped with Wi-Fi, Bluetooth®, and Bluetooth Low Energy (LE) capabilities. It is designed to cater to a broad spectrum of uses, from energy-efficient sensor networks to challenging applications including voice processing, streaming music, and decoding MP3 files. This module's wide-ranging applicability and robust features make it suitable for both simple and complex projects needing reliable wireless communication and processing capabilities. This can be seen below in Figure 18.



Figure 18 Wroom WIFI Chip

### 4.3.2. Configuring The ESP-Wroom-32 Hardware

Configuring the ESP-WROOM is remarkably straightforward. As depicted below in Figure 19, the board is powered through a Micro-USB port, which allows for flexible power options, including standard USB wall adapters or USB power banks. The design ensures seamless integration between this board and its counterpart; they slot together effortlessly and are secured in place with pins, as illustrated below in Figure 20. Correct orientation is key to establishing successful communication between the boards. It's essential that the Micro-USB charging port aligns with the “DM” logo on the adjacent board, guaranteeing proper connection of all pins and ensuring optimal functionality.

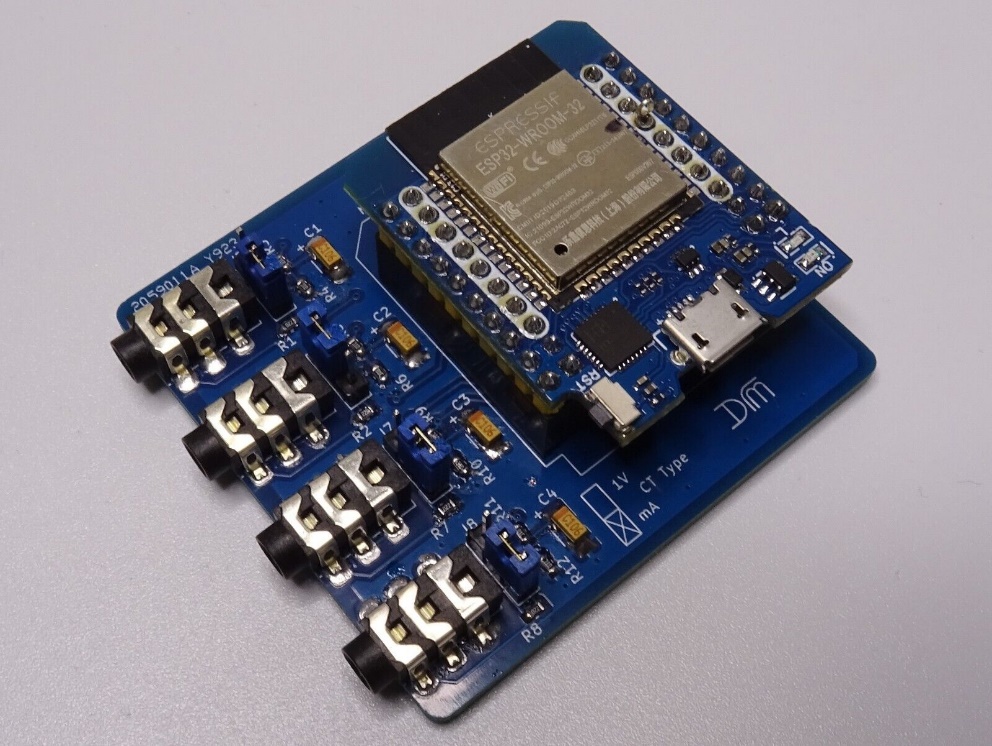


Figure 19 ESP32 and ESP-WROOM Boards Connected

A black rectangular object with gold sticks

Description automatically generated

Figure 20 Pins Responsible for Connecting The 2 Boards

## 4.4. CT Clamp Connections

Figure 21 illustrates that the clamp can be connected to any of the four available inputs on the board, offering the flexibility to monitor either a single circuit or up to four separate circuits simultaneously within a household. The critical aspect to note during setup is the specific input used for each clamp. These input IDs are marked on the board clearly. This detail is crucial for the subsequent programming and calibration stages of the sensors, ensuring accurate data collection and analysis later in the project.

A blue circuit board with black wires

Description automatically generated

Figure 21 Completed Hardware for Project

## 4.5. Arduino IDE

### 4.5.1. Downloading Necessary boards via Board Manager

To begin programming the ESP32, it's necessary to install the appropriate board package. For this project, we've selected the ESP32 by “Espressif Systems” as the preferred board package, due to its extensive capabilities and compatibility with the project’s objectives. The required board package can be seen below in Figure 22.

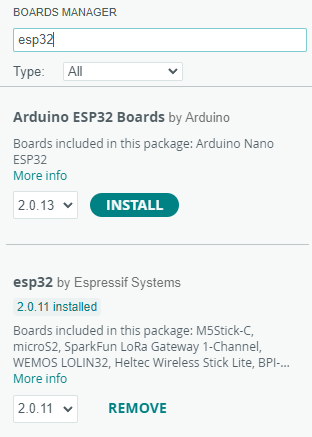


Figure 22 ESP32.

### 4.5.2. Downloading Port Drivers

Installing port drivers for the Arduino IDE is crucial as they enable the IDE to establish communication with the microcontroller board via USB. This ensures the operating system recognises the connected device, allowing the IDE to detect the board, upload compiled programs, and facilitate serial communication for debugging and monitoring. Without these drivers, programming and interacting with the board would not be possible. The drivers for this project can be found at the link below. Also shown below in Figure 23 shows the window in which you can install these drivers.

https://sparks.gogo.co.nz/assets/\_site\_/downloads/CH34x\_Install\_Windows\_v3\_4.zip

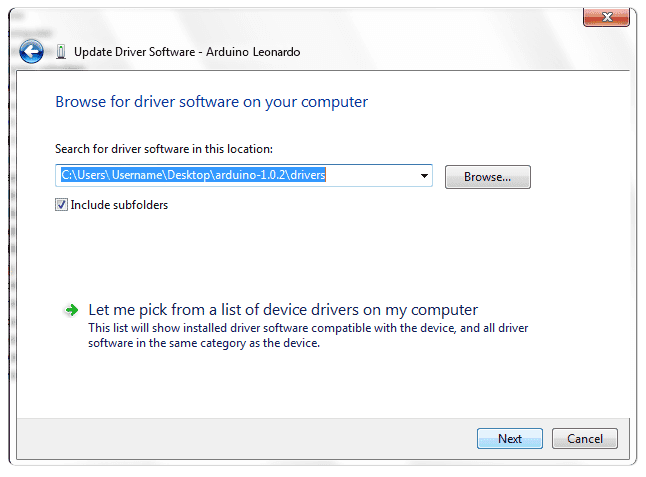


Figure 23 Installing Port Drivers

### 4.5.3. Selecting Board and Port

After installing the required board and port drivers, you can connect the ESP32 to your laptop or PC. Figure 24 demonstrates the graphical user interface (GUI) within the Arduino IDE, where you can select the specific board, you're using and identify the port it's connected to on your device. This step is crucial for ensuring smooth communication between your computer and the ESP32 for programming and project development.

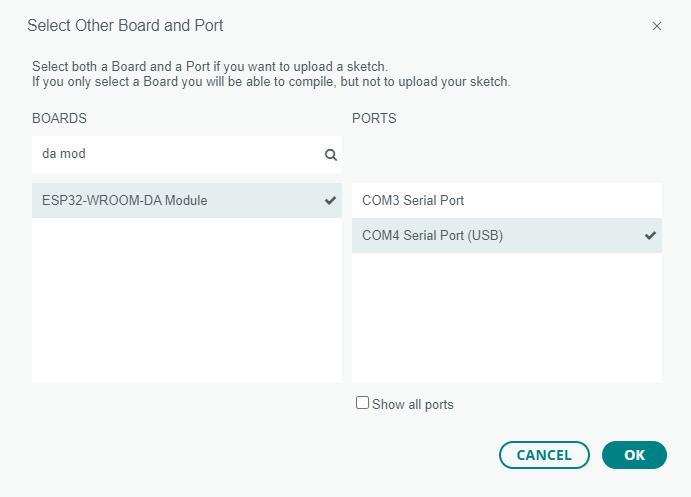


Figure 24 Board and Port GUI

### 4.5.4. Installing Library’s necessary.

Installing package libraries is crucial as they provide ready-made code for common tasks, simplifying development by avoiding the need to write basic functionalities from scratch. This approach saves time, ensures reliability, and allows developers to concentrate on the unique features of their projects, enhancing overall efficiency and capability. View Table 7 below to see what libraries are required and what their purpose is for this project.

Table 7 Required Packages

|  |  |  |
| --- | --- | --- |
| **Library Name** | **Authur Name** | **Purpose** |
| EmonLib | Open Energy Monitor | EmonLib is an Arduino-compatible library designed to simplify energy monitoring by providing easy-to-use functions for measuring current, voltage, power, and energy consumption. |
| ESP32 Firebase | Rupak Poddar | The ESP32 Firebase library provides a set of tools to easily connect and interact with Firebase, Google's platform for building mobile and web applications, enabling real-time data storage, retrieval, and synchronisation for IoT projects. |
| NTP CLient | Fabrice Weinberg | The NTP Client Library is a software package that allows devices to synchronise their internal clocks with internet time servers using the Network Time Protocol, ensuring accurate timekeeping. |

### 4.5.5. Testing If the IDE is talking to the ESP32.

The Arduino IDE includes example codes like "WIFI Scan" and "Firebase Test" to verify package and board functionality. "WIFI Scan" detects nearby networks, while "Firebase Test" sends test data to Firebase, showcasing setup effectiveness as seen in Figure 25.

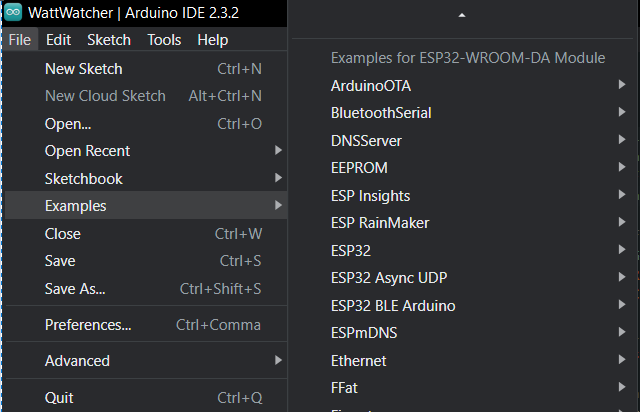


Figure 25 Showcasing Available Examples

## 4.6. Coding of The Arduino

### 4.6.1. Establishing Network Connectivity and Cloud Integration

In the **setup()** function of the IoT device, the ESP32 microcontroller is initialised to connect to a Wi-Fi network and synchronise its internal clock with global time standards using NTP servers. This setup is crucial for the smart energy monitoring system as it ensures the device can reliably transmit real-time energy consumption data to a remote server for analysis. The function configures the microcontroller to station mode, disconnects any existing Wi-Fi connections for a clean state, and establishes a new connection using predefined SSID and password. Once connected, it accesses **pool.ntp.org** and **time.nist.gov** to obtain precise time, essential for accurate data logging and timestamping, which are fundamental for effective energy management and analysis in residential settings. This can be seen below in Listing 1.



Listing 1 System Configuration and initialisation

### 4.6.2. Data Acquisition and Processing

The loop() function within the IoT-based smart energy monitoring system plays a critical role in continuously measuring and calculating the energy consumption data. It operates by assessing the time elapsed since the last measurement (set at one-second intervals) to provide real-time analysis. Within this function, the emon1 and emon2 sensors calculate the root mean square (Irms) current values, which are used to compute the corresponding power in watts and energy in kilowatt-hours. These calculations are based on a predefined voltage value, ensuring that the measurements of energy consumption are accurate and updated every second. This continual monitoring and calculation allow for precise tracking of energy usage across different circuits in a household, forming the basis for effective energy management and sustainability analysis. The Loop function can be seen below in Listing 2.



Listing 2 Data Acquisition and Processing

### 4.6.3. Data Management and Communication

In of the **loop()** function, the smart energy monitoring system performs critical updates by sending real-time energy usage data to Firebase for circuit1+ Circuit 2. This includes power in watts, energy in kilowatt-hours, current in amperes, and a timestamp, ensuring continuous monitoring and immediate accessibility of the data. Furthermore, the system accumulates daily totals of power, current, and energy to compute daily averages. Once every 24 hours, calculated averages for power, current, and energy are also uploaded to Firebase. This methodological updating not only facilitates real-time energy tracking but also enables historical data analysis, vital for assessing energy consumption patterns and implementing efficient energy management strategies in residential settings. The entire code section can be viewed in Code section in appendices B. View Listing 3 for a summary of the code section.



Listing 3 Data Management and Communication

## 4.7. Firebase

### 4.7.1. Firebase Database Creation and structure

Below in Figure 29 illustrates the data structure for a monitored circuit in which there are 4 of these in the Firebase Realtime Database, indicating the headings under which energy consumption metrics are recorded. The structure includes headings for amperage, circuit name, identifier, energy usage, timestamp, and power consumption, providing a comprehensive framework for capturing and organising the energy data of a household circuit in real time.

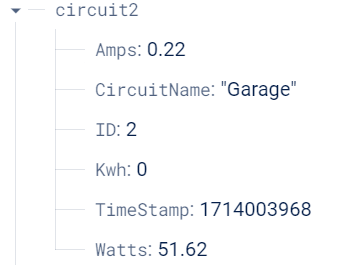
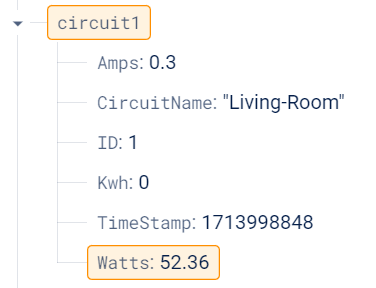


Figure 26 Firebase Database Structure Circuit 1 & 2

### 4.7.2. Average Database Structure (24 Hours)

The structure shows historical energy consumption data for a circuit, indexed by a UNIX timestamp. This timestamp acts as a unique identifier for recording average amperage, kilowatt-hours, and wattage, allowing for precise tracking and analysis of energy usage over time. View below at Figure 27 to get a visual representation of the structure.

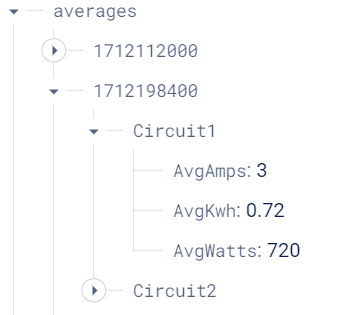


Figure 27 Firebase Average Section

### 4.7.3. Rules and Configuration

Setting the .read and .write rules to true in a Firebase Realtime Database configuration allows unrestricted access to the database, meaning any user can retrieve or modify data without authentication. This setting can be useful during development phases of a project for ease of testing and debugging. Below in Figure 28 depicts how this is done.

A computer code with text

Description automatically generated with medium confidence

Listing 4 Rules & Configuration

## 4.8. .NET MAUI Blazor Application

### 4.8.1. NuGet Packages Required

The table enumerates a curated selection of NuGet packages essential for enhancing the functionality of a .NET MAUI application. These packages range from front-end design with Bootstrap, real-time data handling via Firebase.Database, communication services through Twilio, to dynamic data presentation with APEX Graphs, each contributing specialised capabilities to the overall robustness and user experience of the application. See Table 8 Below.

Table 8 NuGet Packages Required

|  |  |
| --- | --- |
| **Name** | **Description** |
| Bootstrap | Bootstrap for .NET MAUI Blazor is a front-end framework that provides a set of design templates and tools for creating responsive and visually appealing user interfaces within a Blazor hybrid application. |
| Firebase.Database | The Firebase.Database NuGet Package is a library that enables .NET applications to interact with Firebase Realtime Database for seamless data storage, retrieval, and synchronisation in real-time. |
| Twilio | Twilio is a cloud communications platform that allows developers to programmatically send and receive text messages using its web service APIs. |
| APEX Graphs | Apex Charts for .NET MAUI offers a versatile library of rich, interactive, and animated charting capabilities designed to integrate seamlessly with .NET MAUI applications for visually compelling data visualisations. |

### 4.8.2. Setting up Twilio.

To use Twilio for WhatsApp messaging, register for a Twilio account, activate the WhatsApp Sandbox, and configure Webhook URLs. Join your device to the Sandbox with a code sent to your Twilio WhatsApp number. With the setup complete, you can now send and receive messages using the Twilio WhatsApp API.

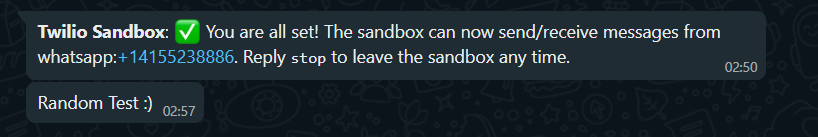


Figure 28 Successful Twillo Setup

4.8.3. Data Model & Data Service Classes

In .NET MAUI and Blazor applications, a data model and service class are crucial for efficient data management. The data model outlines the structure, ensuring data integrity and consistency, while the service class manages business logic and data access, promoting reusability and separation of concerns. This architecture simplifies maintenance, enhances scalability, and supports seamless interactions with external systems. Below in Listing 5 is my live data Circuit Data model Class.

A screenshot of a computer code

Description automatically generated

Listing 5 Model Class for Live Data

### 4.8.4. Setting Up Real-Time Data Fetching

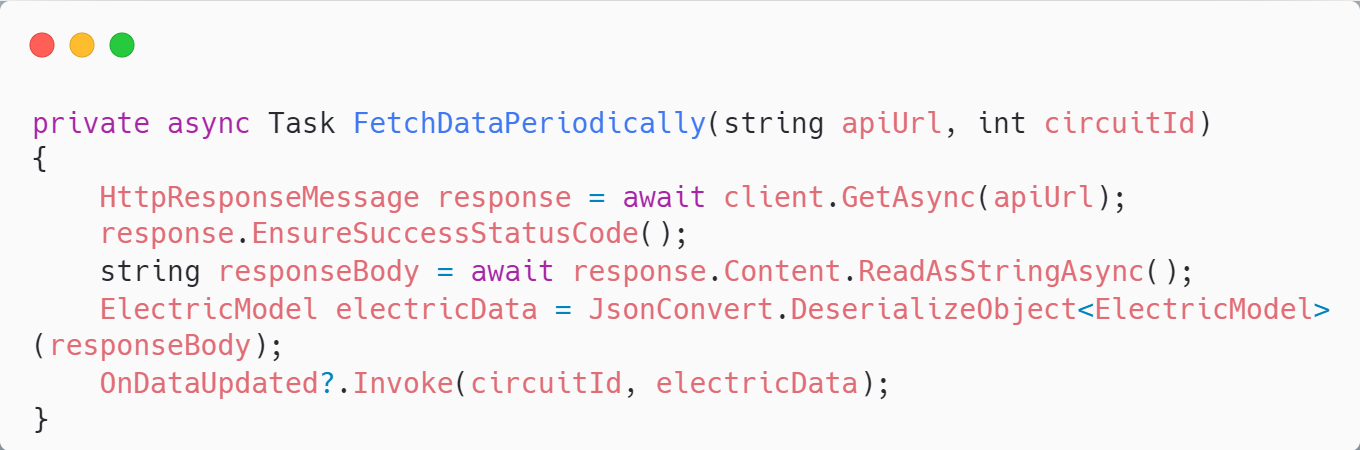
This segment prepares the application for real-time data interaction by establishing a connection to the Firebase database. A timer is configured to fetch data periodically from various circuits, ensuring that the IoT based system maintains continuous access to up-to-date energy usage statistics, which is crucial for effective energy monitoring and management. This section can be seen below in Listing 6.



Listing 6 Setting Up Real-Time Data Fetching

### 4.8.5. Continuous Data Retrieval

Central to the data acquisition process, this method initiates a GET request to the Firebase Realtime Database at regular intervals for each circuit. By employing an asynchronous task, the system processes, and updates data without interrupting the main thread, maintaining responsive interaction with the user while managing live data streams. This Can be depicted in Listing 7.



Listing 7 Continuous Data Retrieval

### 4.8.6. Real-Time Data Update Event.

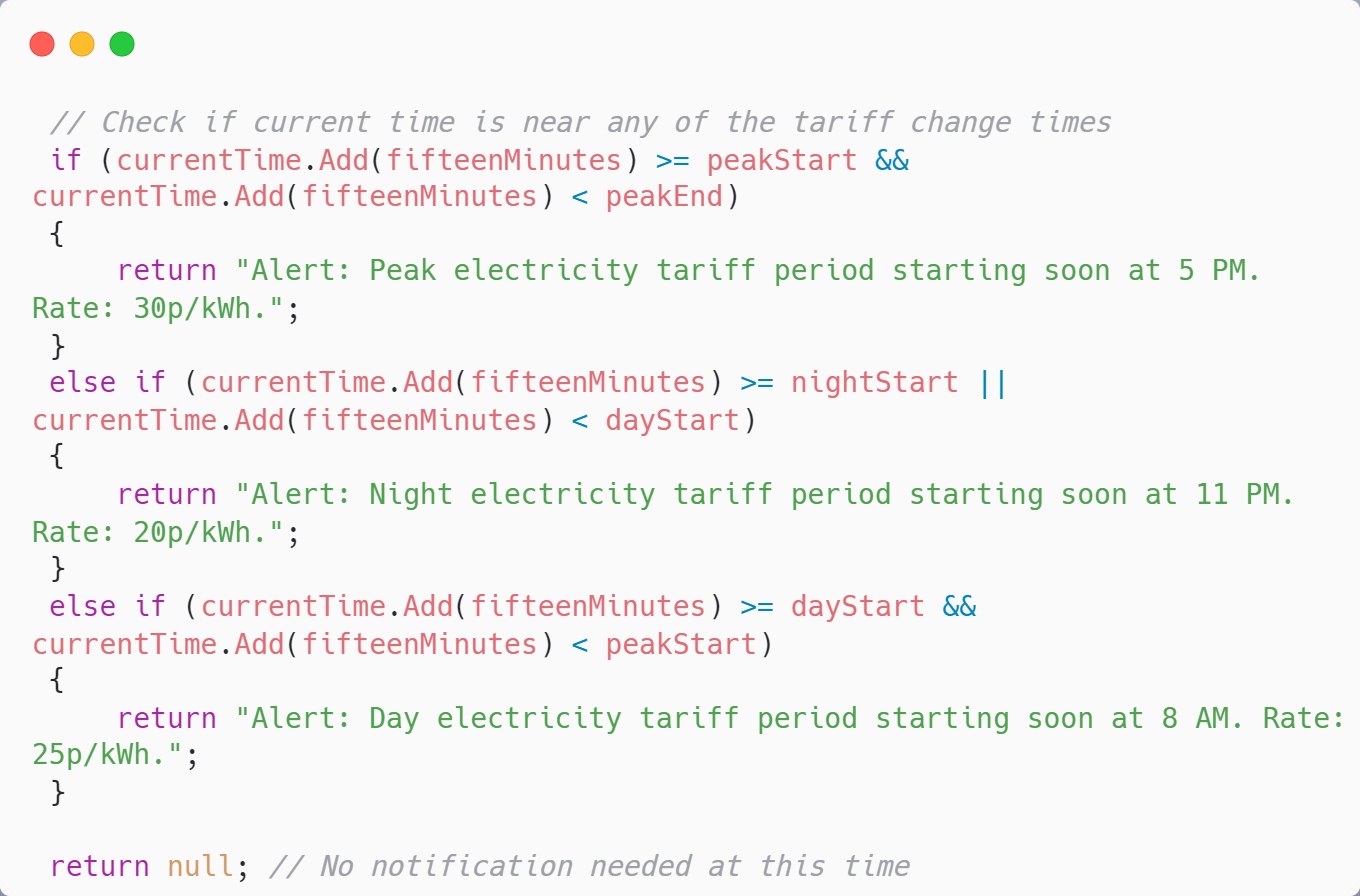
An event is declared that serves as a notification mechanism for new data retrieval. This event is critical for triggering updates throughout the application, assuring that components subscribed to the event are promptly informed of the latest measurements. This facilitates immediate responses, such as display updates, analytics, or activation of alerts for notable energy usage patterns.



Listing 8 Real-Time Data Update Event.

### 4.8.7. Alerting the User of Tariff Updates

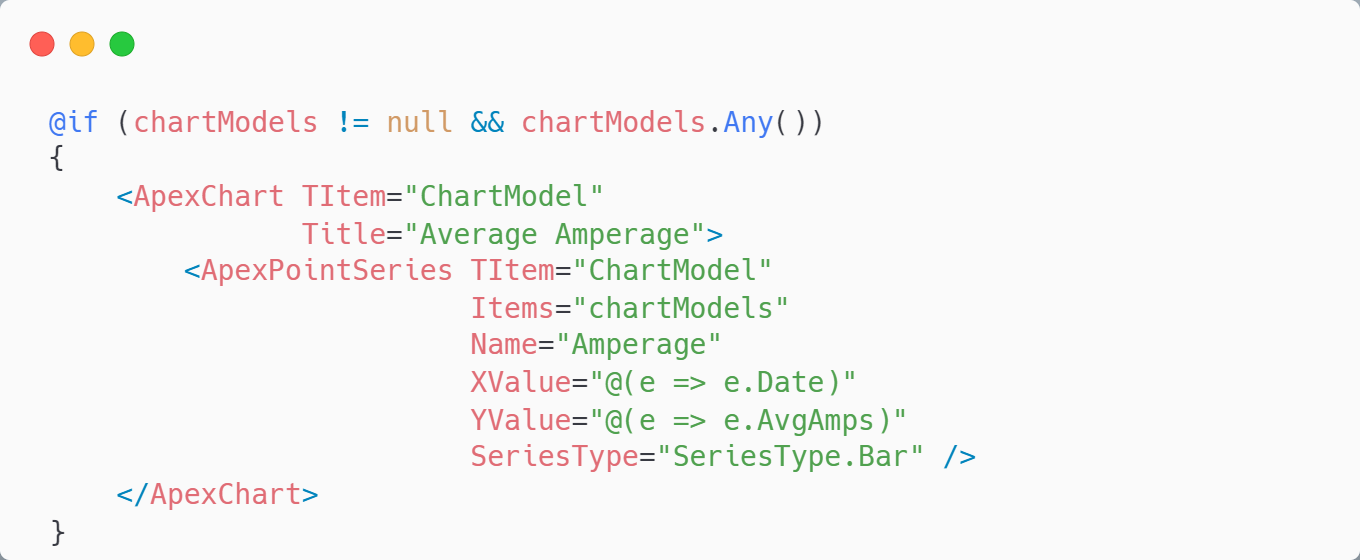
This code is designed to alert users in advance of changing electricity tariffs. By comparing current time with scheduled peak and off-peak periods, it triggers timely notifications. The Goal of this is to allow users to adjust their energy usage to take advantage of lower rates, promoting cost-effective consumption patterns. The intention is to provide a proactive tool for energy expense management within the household. You can see a section from the code in Listing 9.



Listing 9 Alerting the User of Tariff Updates

### 4.8.8. Implementation APEX charts

This code snippet uses the **ApexChart** component to display a bar chart of average amperage in a Blazor application. It only renders the graph if **chartModels** contains data, plotting dates against average amperage values. This visualisation represents the final step in data presentation, if prior extensive calculations and data processing to compute these averages are handled elsewhere in the application. View this code in Listing 10



Listing 10 Apex Charts

### 4.8.9. Setting a Threshold on Electricity Consumption

This code snippet monitors electricity costs and alerts users when spending approaches a predefined limit for specific circuits, such as the "LIVING ROOM". If the total cost nears within 10% of the set limit and no alert has been sent, it automatically sends an SMS to the user. This feature allows users to set precise cost thresholds for individual circuits, enhancing awareness and enabling proactive management of energy consumption to avoid overspending. This tailored approach helps users maintain better control over their electricity budget.



Listing 11 User Defined Threshold for Electricity Consumption

## 5.0. Testing Strategy

## 5.1. Introduction

In the intricate process of software development, the Testing Strategy chapter assumes a critical role, ensuring the reliability and functionality of the devised solution. This chapter delves into the realm of testing, unravelling its significance and various methodologies. From the broad spectrum of testing types to a focused examination of functional requirements, the chapter provides insights into the meticulous process of validating the robustness of our project. The overarching goal is to establish a comprehensive testing framework, guaranteeing a seamless user experience and adherence to defined requirements.

## 5.2. White-Box Testing

White box testing, also known as clear, glass box, or structural testing, is a method that examines a software application's internal logic, structure, and code. White box testing, as opposed to testing methods that focus solely on the application's functionality, seeks to evaluate the software's internal workings. Testers with access to the application's source code create test cases to investigate different paths and conditions within the codebase. The primary goal is to ensure that all code paths are executed correctly, improving code quality, and revealing potential vulnerabilities. White box testing is an effective method for achieving comprehensive test coverage and validating the correctness of the internal logic of software.(Nidhra, 2012).

### 5.2.1. Unit Testing

Unit testing is a critical component of the software development process that focuses on examining individual units or components of a software application. These units, which frequently consist of specific functions or modules, are examined in isolation to ensure that their functionality is consistent with the intended design. Unit tests are an important quality assurance measure because they allow developers to identify and correct errors in the early stages of development. Unit tests contribute to the overall reliability and maintainability of software by isolating and assessing discrete elements of the codebase. They are an essential component of a robust testing strategy.(Olan, 2003).

### 5.2.2. Integration Testing

Integration testing is a software testing methodology that evaluates the interaction and combination of various components or systems to ensure they work as a unified whole. The goal of integration testing is to find flaws in the interfaces and interactions between integrated components and validate that they work together. This type of testing is critical for identifying issues that may arise when integrating individual components or modules, thereby ensuring the overall system's reliability and robustness. Integration testing can be done at different levels, such as module-to-module, system-to-system, and client-to-server interactions, to ensure that the integrated parts of a software application work together to deliver the intended functionality.(Leung and White, 1990).

## 5.3. Black-Box Testing

Black box testing is a method of software testing that focuses on evaluating a system's functionality without delving into its internal code or logic. Testers evaluate the system solely on its inputs and expected outputs, treating the software as a "black box" with unknown internal workings. This method validates the system's specifications, requirements, and overall functionality while ignoring the internal structure. Black box testing is effective for identifying errors or discrepancies between expected and actual results, providing insight into how well the software meets its specified requirements from the perspective of the end user. To ensure comprehensive validation of the software's external behaviour, test cases covering various scenarios are created during the testing process. (Nidhra, 2012).

### 5.3.1. Functionality Testing

Functionality testing ensures that a software system performs as intended, meeting specified requirements. This type of testing examines various features, such as user interfaces, APIs, databases, and security, to validate that they work correctly. Tester’s design specific test cases to identify errors or deviations from expected behaviour, ensuring the software functions seamlessly and delivers a positive user experience. (Baresi and Pezzè, 2006).

### 5.3.2. Exploratory Testing

Exploratory testing is a dynamic, unscripted approach to software testing where testers explore the application without predefined test cases. Testers simultaneously design and execute tests, relying on their expertise, intuition, and creativity. This method is particularly effective for uncovering unforeseen issues, emphasising adaptability, and learning throughout the testing process. Exploratory testing is often used in conjunction with scripted testing to provide comprehensive test coverage. (Baresi and Pezzè, 2006)

See Table 9 below for a list of requirements and the tests for each.

Table 9. Testing Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test ID** | **Requirements** | **Testing Type** | **Steps** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| TC01 | CT Clamp must be encircling the live or neutral cable. | Black-Box Testing | Verify the CT Clamp is properly installed around the cable. | CT Clamp is correctly encircling the cable without any issues. | CT Clamp is properly installed around the cable without any issues. | Pass |
| TC02 | System must be able to measure the amperage from the cable. | Black-Box Testing | Measure amperage from the cable using the system. | System accurately measures the amperage. | System measures the amperage accurately. | Pass |
| TC03 | Real-time monitoring of electricity is a must for this system. | Black-Box Testing | Monitor the electricity in real-time using the system. | System displays real-time electricity data accurately. | System displays real-time electricity data accurately. | Pass |
| TC04 | The system must be able to send amperage to Firebase, the cloud database. | White-Box Testing | Send amperage data to Firebase and verify data integrity. | Data is sent to Firebase accurately and promptly. | Amperage data is successfully sent to Firebase and verified for integrity.  (Figure 27) | Pass |
| TC05 | .NET Maui application must be able to fetch the data from the cloud database. | Black-Box Testing | Fetch data from Firebase using the .NET Maui application. | Application successfully retrieves and displays data from Firebase. | .NET Maui application successfully retrieves and displays data from Firebase.  (Figure 29) | Pass |
| TC06 | The system must be able to display the real-time electricity usage. | Black-Box Testing | Check the display of electricity usage on the system. | The system displays the electricity usage accurately | The system displays the electricity usage accurately.  (Figure 32) | Pass |
| TC07 | A user must be able to set a usage threshold. | Black-Box Testing | Attempt to set a usage threshold in the system. | The system allows setting a usage threshold easily and correctly. | The system allows setting a usage threshold easily and correctly.  (Figure 31) | Pass |
| TC08 | The system must notify the user if approaching the set limit of electricity consumption. | Black-Box Testing | Approach the set limit to trigger a notification. | The system sends a notification when nearing the consumption limit. | The system sends a notification when nearing the consumption limit.  (Figure 36) | Pass |
| TC09 | The system should provide historical information on electricity consumption. | Black-Box Testing | Access historical electricity consumption data in the system. | The system displays accurate historical data. | The system displays accurate historical data.  (Figure 33) | Pass |

The outcomes of the described tests in Table 9 are detailed in the results section below. Each test verifies specific functionalities such as real-time monitoring, data integrity, and notification mechanisms. The results confirm that the system meets its design specifications, with visual evidence provided in the referenced figures for each test case.

### 5.3.3. Sensor Testing

## Smart Household Energy Monitoring and Management, sensor testing would be vital to ensure the precision and reliability of the IoT sensors, particularly the current transformer (CT) clamps and the ESP-32 module, in monitoring and reporting household energy consumption. This testing is essential for the overall effectiveness and user trust in the energy monitoring system.

### 5.3.4. How to Test the Sensor

## Based on the research conducted in the previous chapters, it has been identified that the practical implementation of the CT clamp may present challenges, particularly regarding the accessibility and positioning of the fuse board or distribution board. To address this issue, a modified solution is proposed: the use of a specially adapted extension lead. This is done by removing the outside sheath of the cable, still ensuring the cables safety features.

## This modified extension lead will serve as a temporary testing apparatus, facilitating easier and more flexible testing of the sensor. This approach not only simplifies the testing process but also ensures the effective evaluation of the sensor's performance in a controlled and accessible environment before its final installation in the less accessible fuse board area.

Below in Figure 29 shows the extension lead used for testing purposes.

A white extension cord with a blue and black cord

Description automatically generated

Figure 29. Extension lead used during testing.

### 5.3.5. Ensuring Measurement Accuracy in Energy Monitoring

The Current Transformer (CT) clamp, integral to the IoT energy monitoring system, requires accuracy verification to ensure reliable data. This is where a standard AC clamp meter comes in. By measuring the same conductor, the clamp meter confirms the CT clamp's readings. Regular calibration checks with the clamp meter help maintain the monitoring system’s accuracy, essential for effective energy management.

A red and black device with a digital display

Description automatically generated

Figure 30 AC Clamp Meter

## 6.1. Results and Evaluation

### 6.1.1. Circuit Monitoring and Alert System on Smart Home Dashboard

The Home Screen displayed in Figure 29 offers a comprehensive view of the cumulative electricity consumption across various circuits in a household, such as the living room, garage, central heating, and master bedroom, presented in a clear and visually engaging pie chart. This holistic perspective is beneficial as it provides the homeowner with a real-time snapshot of total energy usage, enabling them to monitor the aggregate electrical load. Moreover, the system enhances home energy management by integrating an SMS alert feature, which proactively notifies the homeowner if the total consumption approaches a critical threshold—10% below the main fuse capacity, set at 80 Amps. This alert mechanism serves as a precautionary tool, allowing residents to take timely actions to prevent potential overloads, thereby maintaining the safety and efficiency of the household's electrical system.

A screenshot of a computer

Description automatically generated

Figure 31 Main Screen of Application

### 6.1.2. Live Data and Circuit Selection

The dashboard interface, as presented below in Figure 30, is a user-centric design that facilitates monitoring and management of household energy consumption. Individual panels for the 'Entire Home', 'Living Room', 'Kitchen', and 'Central Heating' provide weekly cost metrics with actionable insights for energy savings, such as optimising usage times and maintaining appliances. This design enables users to navigate detailed consumption data and customise thresholds for efficient energy expenditure, thereby promoting sustainability and cost-effectiveness within the domestic setting.

A screenshot of a computer

Description automatically generated

Figure 32 Live Data Details and Circuit Selection

### 6.1.3. Setting Electrical consumption Limit.

In the user interface depicted in Figure 31, the energy monitoring application empowers users to establish a weekly expenditure ceiling for electricity costs. This proactive feature allows the entry of a numerical value, in this instance '15 euros', which upon confirmation through the 'SET LIMIT' button, is retained within the application's local storage. The integration with Twilio's SMS service adds a layer of responsiveness; as electricity costs approach within a 10% margin of the pre-set threshold, the system triggers an automated alert to the user. This function, combined with the displayed metrics of weekly energy usage, costs, and average daily consumption, offers a sophisticated tool for managing and optimising household energy expenditures, ensuring adherence to budgetary constraints. The stored value and the reactive alert system collectively contribute to a highly interactive and user-centric experience in monitoring energy consumption. View this user defined threshold below in Figure 31.

A screenshot of a computer

Description automatically generated

Figure 33 Set Limit Section

### 6.1.4. Individual Breakdown of Live Consumption.

The 'Live Energy Consumption' panel in the energy monitoring application provides real-time updates on power usage for a specific circuit, refreshing data every 5-10 seconds. Displayed metrics include power draw in watts, energy consumed in kilowatt-hours, and current in amperes, allowing users to observe and manage their immediate electricity usage effectively.

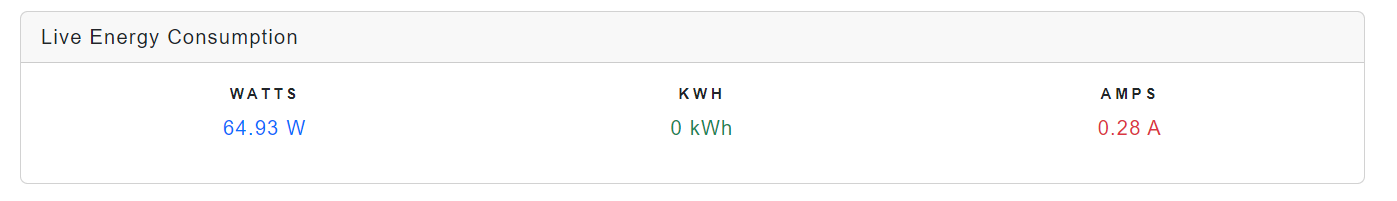


Figure 34 Individual Live Circuit View

### 6.1.5. Historical Weekly Data Table.

Figure 35 Historical Weekly Data Table

The interface in Figure 33 presents a 'Historical Data' table within the energy monitoring application that captures and displays historical electricity usage by circuit, such as the living room. Users can filter this data by circuit type and view specific electrical parameters like amperage (A), energy (kWh), and power (W), enabling a detailed examination of consumption patterns. The data can also be sorted by date, providing temporal insights into energy usage.

The ability to filter by circuit or date is instrumental for energy management, as it allows users to pinpoint specific areas where consumption is highest and identify periods of peak usage. This granular level of detail supports informed decision-making to optimise energy usage, leading to potential cost savings and promoting efficient energy practices. The feature also serves as a valuable tool for tracking the effectiveness of energy-saving measures implemented over time.

### 6.1.6. Notification Outcomes for Circuit Limit Monitoring

The displayed alerts represent an integrated energy monitoring alert system's output, communicated via Twilio for effective user notification. This system meticulously tracks energy costs across different household circuits, providing two simultaneous alerts: one indicating that the living room's weekly electricity cost is nearing its set threshold with a cost of €18.60, alerting the user of the proximity to the €18.00 limit, and another more urgent notification for the kitchen, where the cost has overshot the €20.00 limit by over 10%, totalling €36.41, thereby prompting immediate energy consumption review. Grouping these alerts demonstrates the system's capability to concurrently monitor and report on multiple circuits, offering nuanced, circuit-specific consumption feedback that enables users to make informed energy-saving decisions. View these WhatsApp Notifications below in Figure 35.

A screenshot of a phone

Description automatically generated

Figure 36 Alert Notifications

### 6.1.7. Adaptive Tariff Rate Notifications for Optimised Energy Usage

Below in Figure 36 showcases a series of alerts grouped together for demonstration purposes, each indicating the commencement of different electricity tariff periods within the day. Under normal circumstances, these alerts would be sent individually to inform users in real-time: one indicates the start of the peak tariff at 5 PM with a rate of 30p/kWh, followed by the night tariff at 11 PM at a lower rate of 20p/kWh, and lastly, the day tariff beginning at 8 AM at 25p/kWh. This illustrative grouping is intended to showcase the system's ability to deliver timely and relevant cost-saving information to users, prompting efficient energy usage aligned with fluctuating tariff rates.

A screenshot of a phone

Description automatically generated

Figure 37 Result of Tariff Changes

## Chapter 7: Conclusion

## 7.1. Introduction

This chapter revisits the overarching aims and specific objectives of the project, summarizing the outcomes and offering conclusive insights. It concludes with recommendations for further research and a personal reflection on the research process. The primary goal of this project was to develop an IoT-based system for real-time household energy monitoring and management, aimed at empowering homeowners to manage their energy usage efficiently.

## 7.2. Summary of Findings and Conclusions

### 7.2.1. Literature Review on IoT and Energy Monitoring

The literature review underscored the significant potential of IoT technologies in enhancing energy monitoring systems, particularly highlighting gaps in real-time, user-friendly applications for household energy management. The findings demonstrate a growing need for systems that provide homeowners with detailed insights into their energy consumption patterns. The conclusions drawn emphasise the critical role of IoT in achieving enhanced energy efficiency and user engagement in managing energy resources.

### 7.2.2. Implementation and Testing of the IoT System

Initially, the project was designed to utilise an ESP8266 microcontroller; however, challenges related to the serial port's inability to handle multiple inputs necessitated a switch to the ESP32 microcontroller. This adjustment allowed for greater flexibility and reliability, as the ESP32 supports multiple input channels and offers improved processing power and connectivity options. The software system constructed around the ESP32, integrated with Current Transformer (CT) clamps and MQTT protocol for real-time data processing, and Firebase for data storage, successfully provided a user-friendly interface for homeowners. This interface enabled real-time energy consumption monitoring across iOS, Android, and PC platforms, effectively meeting the project's objectives.

## 7.3. Recommendations for Future Work

While the project achieved its intended objectives effectively, future research could extend in several promising directions. Further integration of the IoT energy monitoring system with broader smart home systems could provide more holistic energy management solutions. This would allow for seamless control and automation of various household appliances, enhancing overall energy efficiency.

Applying machine learning algorithms to predict future energy usage patterns based on historical data could offer proactive energy management strategies. Such predictive analytics could help homeowners anticipate and adjust their energy consumption before peak demand periods, potentially reducing costs and enhancing system responsiveness.

Adapting the system for commercial applications could test its scalability and functionality in a different context, providing insights into its adaptability and robustness. This expansion would help explore the potential of the IoT system in larger, more complex environments, such as business settings or industrial applications, where energy demands, and usage patterns significantly differ from residential contexts.

## 7.4. Self-reflection

Reflecting on the project lifecycle, from conception through implementation and testing, several key lessons emerged. One of the most significant is the importance of clear planning and the need for adaptive problem-solving strategies in dealing with unforeseen challenges, such as the need to switch microcontrollers for technical compatibility. This experience has underscored the value of perseverance and flexibility in the research process. Future researchers should consider the scalability of their proposed solutions from the outset and prepare for iterative testing and refinement phases.

## 7.5. Final Thoughts

This research contributes to the evolving field of IoT and energy management, providing a practical solution that enhances homeowners' ability to monitor and manage energy consumption. The knowledge gained through this project is a testament to the potential of IoT technologies to transform our daily lives through smarter energy solutions.

# Appendices A: References

1. What Is React Native? - Learning React Native [Book] [WWW Document], n.d. URL https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html (accessed 10.27.23).

adnanaqeel, 2018. Introduction to Arduino IDE - The Engineering Projects [WWW Document]. URL https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html (accessed 12.14.23).

Amazon, 2023a. UEETEK SCT-013-000 Non-Invasive AC Current Sensor Split Core Transformer 100A for Arduino Project : Amazon.co.uk: Business, Industry & Science [WWW Document]. URL https://www.amazon.co.uk/UEETEK-SCT-013-000-Non-Invasive-Current-Transformer/dp/B0727WRRSJ/ref=sr\_1\_3?crid=2ROP09L4HOE0T&keywords=ct+clamp&qid=1700443658&sprefix=ct+clam%2Caps%2C76&sr=8-3 (accessed 11.20.23).

Amazon, 2023b. OWL Micro+ CM180 Wireless Home Energy Monitor : Amazon.co.uk: DIY & Tools [WWW Document]. URL https://www.amazon.co.uk/dp/B007W0SQ3Y (accessed 12.14.23).

American History Museum, 2023. Edison chemical-type electric meter [WWW Document]. National Museum of American History. URL https://americanhistory.si.edu/collections/search/object/nmah\_712826 (accessed 11.10.23).

Arduino, 2024. Arduino Integrated Development Environment (IDE) v1 | Arduino Documentation [WWW Document]. URL https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics/ (accessed 3.1.24).

Arduino, 2024. What is Arduino? | Arduino Documentation [WWW Document]. URL https://docs.arduino.cc/learn/starting-guide/whats-arduino/ (accessed 3.1.24).

Baresi, L., Pezzè, M., 2006. An Introduction to Software Testing. Electronic Notes in Theoretical Computer Science, Proceedings of the School of SegraVis Research Training Network on Foundations of Visual Modelling Techniques (FoVMT 2004) 148, 89–111. https://doi.org/10.1016/j.entcs.2005.12.014

Beginners, E. for, 2023. ESPHome and Home Assistant Integration (2023) [WWW Document]. URL https://www.espforbeginners.com/guides/esphome-home-assistant-integration/ (accessed 12.14.23).

Bhanderi, P., 2023. Flutter’s Benefits and Drawbacks. Medium. URL https://medium.com/@parthbhanderi01/flutters-benefits-and-drawbacks-b268c1fe0f7f (accessed 11.20.23).

Boduch, A., 2017. React and React Native. Packt Publishing Ltd.

Bogdan, G., 2021. Firebase Pros and Cons: When You Should and Shouldn’t Use Firebase [WWW Document]. OSDB. URL https://osdb.io/firebase-pros-and-cons-when-you-should-and-shouldnt-use-firebase-osdb/ (accessed 11.20.23).

Boyce, S., 2017. Attack On The Clones: A Review Of Two Common ESP8266 Mini D1 Boards. Hackaday. URL https://hackaday.com/2017/05/15/attack-on-the-clones-a-review-of-two-common-esp8266-mini-d1-boards/ (accessed 11.20.23).

Chaudhary, S., Bhargave, V., Kulkarni, S., Puranik, P., Shinde, A., 2018. Home Automation System Using WeMos D1 Mini 05.

Chooruang, K., Meekul, K., 2018. Design of an IoT Energy Monitoring System, in: 2018 16th International Conference on ICT and Knowledge Engineering (ICT&KE). Presented at the 2018 16th International Conference on ICT and Knowledge Engineering (ICT&KE), pp. 1–4. https://doi.org/10.1109/ICTKE.2018.8612412

Ciontea, C.I., Iov, F., 2021. A Study of Load Imbalance Influence on Power Quality Assessment for Distribution Networks. Electricity 2, 77–90. https://doi.org/10.3390/electricity2010005

Consumer World, 2023. What is an MCB and how does it work? Miniature Circuit Breakers Explained [WWW Document]. URL https://www.consumerunitworld.co.uk/what-is-an-mcb-and-how-does-it-work-328-c.asp (accessed 12.14.23).

Dekkati, S., Lal, K., Desamsetti, H., 2019. React Native for Android: Cross-Platform Mobile Application Development. Global Disclosure of Economics and Business 8, 153–164. https://doi.org/10.18034/gdeb.v8i2.696

Despa, D., Nama, G.F., Muhammad, M.A., Anwar, K., 2018. The Implementation Internet of Things(IoT) Technology in Real Time Monitoring of Electrical Quantities. IOP Conf. Ser.: Mater. Sci. Eng. 335, 012063. https://doi.org/10.1088/1757-899X/335/1/012063

Documentation, A., 2023. Using the Arduino Software (IDE) | Arduino Documentation [WWW Document]. URL https://docs.arduino.cc/learn/starting-guide/the-arduino-software-ide (accessed 11.28.23).

Ed, 2018. Substation Monitoring for Future Networks [WWW Document]. LOGIC ENERGY. URL https://www.logicenergy.com/post/substation-monitoring-for-future-networks (accessed 11.19.23).

Espressif, 2021. Analog to Digital Converter (ADC) - ESP32 - — ESP-IDF Programming Guide v4.4 documentation [WWW Document]. URL https://docs.espressif.com/projects/esp-idf/en/v4.4/esp32/api-reference/peripherals/adc.html (accessed 3.1.24).

Evenergy, 2023. Why do I need a CT clamp on my charger? [WWW Document]. ev.energy. URL https://support.ev.energy/en/support/solutions/articles/80000604907-why-do-i-need-a-ct-clamp-on-my-charger- (accessed 11.20.23).

Ferrero, A., 2008. Measuring electric power quality: Problems and perspectives. Measurement, Advances in Measurements of Eelectrical Quantities 41, 121–129. https://doi.org/10.1016/j.measurement.2006.03.004

Firebase, F., 2023. Firebase Realtime Database | Store and sync data in real time [WWW Document]. Firebase. URL https://firebase.google.com/products/realtime-database (accessed 11.28.23).

Gokhale, P., Bhat, O., Bhat, S., 2018. Introduction to IOT 5, 41–44. https://doi.org/10.17148/IARJSET.2018.517

Greg, 2017. IoT applications - monitoring energy consumption using Pressac CT Clamps [WWW Document]. URL https://www.rs-online.com/designspark/iot-applications-monitoring-energy-consumption (accessed 11.20.23).

Heralgi, G., 2019. Functional and Non-Functional Requirement Gathering. Medium. URL https://monocept.medium.com/functional-and-non-functional-requirement-gathering-f122f810e522 (accessed 11.28.23).

HoTran, T., 2019. 10 pros and cons of the hamburger menu (with examples) | Inside Design Blog [WWW Document]. URL https://www.invisionapp.com/inside-design/pros-and-cons-of-hamburger-menus/ (accessed 12.11.23).

How to Read Your Meter | ESB Networks [WWW Document], n.d. . Default. URL https://www.esbnetworks.ie/existing-connections/meters-and-readings/how-to-read-your-meter (accessed 11.11.23).

Hübschmann, I., 2020. ESP32 for IoT: A Complete Guide. Nabto. URL https://www.nabto.com/guide-to-iot-esp-32/ (accessed 3.1.24).

IBM, 2023. What is cloud computing? | IBM [WWW Document]. URL https://www.ibm.com/topics/cloud-computing (accessed 11.20.23).

IBM, 2021. IBM Documentation [WWW Document]. URL https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case (accessed 12.11.23).

Krishnamurthi, R., Kumar, A., Gopinathan, D., Nayyar, A., Qureshi, B., 2020. An Overview of IoT Sensor Data Processing, Fusion, and Analysis Techniques. Sensors 20, 6076. https://doi.org/10.3390/s20216076

Leung, H.K.N., White, L., 1990. A study of integration testing and software regression at the integration level, in: Proceedings. Conference on Software Maintenance 1990. Presented at the Proceedings. Conference on Software Maintenance 1990, pp. 290–301. https://doi.org/10.1109/ICSM.1990.131377

Matos, L., 2023. Why is .NET MAUI the best tool for cross-platform mobile development? Luis Matos. URL https://luismts.com/dotnet-maui-best-tool-for-mobile-development/ (accessed 10.27.23).

Metered Electricity Consumption 2021 - CSO - Central Statistics Office [WWW Document], 2022. URL https://www.cso.ie/en/releasesandpublications/ep/p-mec/meteredelectricityconsumption2021/ (accessed 11.19.23).

Microsoft, 2023. What is .NET MAUI? - .NET MAUI [WWW Document]. URL https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui (accessed 10.27.23).

Mitiku, A., 2022. Data warehouse tech stack with MySQL, DBT, Airflow. Medium. URL https://medium.com/@abel-blue/data-warehouse-tech-stack-with-mysql-dbt-airflow-eb31ee15e570 (accessed 11.20.23).

Mohan, S., 2020. IoT With Wemos. Medium. URL https://medium.com/@sujaymohan/iot-with-wemos-b4fe945c9633 (accessed 11.20.23).

Mottramlabs-Products [WWW Document], n.d. URL https://www.mottramlabs.com/esp\_products.html (accessed 10.22.23).

Mouha, R.A., 2021. Internet of Things (IoT). Journal of Data Analysis and Information Processing 9, 77–101. https://doi.org/10.4236/jdaip.2021.92006

Nidhra, S., 2012. Black Box and White Box Testing Techniques - A Literature Review. International Journal of Embedded Systems and Applications 2, 29–50. https://doi.org/10.5121/ijesa.2012.2204

Nižetić, S., Šolić, P., López-de-Ipiña González-de-Artaza, D., Patrono, L., 2020. Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future. Journal of Cleaner Production 274, 122877. https://doi.org/10.1016/j.jclepro.2020.122877

Obrutsky, S., 2016. Cloud Storage: Advantages, Disadvantages and Enterprise Solutions for Business.

Ohm’s Law - How Voltage, Current, and Resistance Relate | Ohm’s Law | Electronics Textbook [WWW Document], n.d. URL https://www.allaboutcircuits.com/textbook/direct-current/chpt-2/voltage-current-resistance-relate/ (accessed 11.20.23).

Olan, M., 2003. Unit testing: Test early, test often. Journal of Computing Sciences in Colleges - JCSC 19.

Oracle, 2022. What is a relational database? [WWW Document]. URL https://www.oracle.com/ie/database/what-is-a-relational-database/ (accessed 11.20.23).

OWL Micro+, 2023. Energy Monitors | OWL Micro+ | The OWL [WWW Document]. URL https://www.theowl.com/index.php/energy-monitors/standalone-monitors/owl-micro/ (accessed 12.14.23).

Pope, C., 2023. Pricewatch: Will energy prices remain high for the foreseeable future? [WWW Document]. The Irish Times. URL https://www.irishtimes.com/your-money/2023/08/28/is-there-any-hope-of-cheaper-energy-for-irish-homes-this-winter/ (accessed 10.19.23).

Rao, R., 2023. How does electricity work? Let’s demystify the life-changing physics. [WWW Document]. Popular Science. URL https://www.popsci.com/technology/how-does-electricity-work/ (accessed 11.19.23).

Rose, B., 2023. What are the two main types of current? Medium. URL https://medium.com/@Breadarose/what-are-the-two-main-types-of-current-c527ac3a9267 (accessed 11.19.23).

RS, I., 2023. MCB (Miniature Circuit Breakers) Guide | RS [WWW Document]. URL https://uk.rs-online.com/web/content/discovery/ideas-and-advice/mcb-guide (accessed 12.14.23).

Sheikh, F., 2020. Ohm’s Law: Theory, Circuit, And Application - EEE PROJECTS. URL https://eeeproject.com/ohms-law/ (accessed 11.20.23).

Singh, M., 2023. Flutter vs .NET MAUI — Features, Differences and Limitations. Medium. URL https://dev-malwinder.medium.com/flutter-vs-net-maui-features-differences-and-limitations-d5dc369bbc77 (accessed 10.27.23).

Singh, Y., 2020. Simplicity is prerequisite for reliability and less cognitive load in user experience. Medium. URL https://medium.com/@uxbond/simplicity-is-prerequisite-for-reliability-and-less-cognitive-load-in-user-experience-c7217a35e1d6 (accessed 11.29.23).

SqLite, 2023. About SQLite [WWW Document]. URL https://www.sqlite.org/about.html (accessed 11.14.23).

Teel, J., 2024. Introduction to the ESP32 Family of Wireless Microcontrollers (Updated 2024) [WWW Document]. PREDICTABLE DESIGNS. URL https://predictabledesigns.com/introduction-to-the-esp32-wifi-bluetooth-wireless-microcontroller/ (accessed 3.1.24).

Toal, R., 2015. What’s the difference between UX and UI Design? Code Institute IE. URL https://codeinstitute.net/ie/blog/the-difference-between-ux-and-ui-designers/ (accessed 11.29.23).

Triantafyllou, A., Sarigiannidis, P., Lagkas, T.D., 2018. Network Protocols, Schemes, and Mechanisms for Internet of Things (IoT): Features, Open Challenges, and Trends. Wireless Communications and Mobile Computing 2018, e5349894. https://doi.org/10.1155/2018/5349894

Vadda, P., Seelam, S.M., 2013. Smart Metering for Smart Electricity Consumption.

Vila, M., Sancho, M.-R., Teniente, E., Vilajosana, X., 2023. Critical infrastructure awareness based on IoT context data. Internet of Things 23, 100855. https://doi.org/10.1016/j.iot.2023.100855

What Is a Cloud Database? [WWW Document], 2021. . Google Cloud. URL https://cloud.google.com/learn/what-is-a-cloud-database (accessed 11.14.23).

Wu, W., 2018. React Native vs Flutter, cross-platform mobile application frameworks.

Yokotani, T., Sasaki, Y., 2016. Comparison with HTTP and MQTT on required network resources for IoT, in: 2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC). Presented at the 2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), pp. 1–6. https://doi.org/10.1109/ICCEREC.2016.7814989

Yulchon-Howstuffworks-How-Electricity-Works-091116.pdf, n.d.

# Appendices B (Code Listings)

### B.1 - System Configuration and initialisation

void setup() {

Serial.begin(115200);

WiFi.mode(WIFI\_STA);

WiFi.disconnect();

delay(100);

Serial.println("Connecting to: ");

Serial.println(\_SSID);

WiFi.begin(\_SSID, \_PASSWORD);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("WiFi Connected");

configTime(0, 0, "pool.ntp.org", "time.nist.gov");

while (!time(nullptr)) {

Serial.print(".");

delay(1000);

}

Serial.println("Time set");

emon1.current(34, 22);

emon2.current(35, 22);

}

### B.2 - Data Acquisition and Processing

void loop() {

unsigned long currentMillis = millis();

time\_t now = time(nullptr);

if (currentMillis - lastMillis >= 1000) {

lastMillis = currentMillis;

double elapsedTime = 1.0; // Elapsed time is 1 second

double Irms1 = emon1.calcIrms(1480);

double powerWatts1 = Irms1 \* voltage;

double currentAmps1 = Irms1;

double energyKWh1 = (powerWatts1 \* elapsedTime) / 3600000.0;

double Irms2 = emon2.calcIrms(1480);

double powerWatts2 = Irms2 \* voltage;

double currentAmps2 = Irms2;

double energyKWh2 = (powerWatts2 \* elapsedTime) / 3600000.0;

...

}

}

### B.3- Data Management and Communication

void loop() {

...

// Update Firebase with real-time data

firebase.setFloat("circuit1/Watts", powerWatts1);

firebase.setFloat("circuit1/Kwh", energyKWh1);

firebase.setFloat("circuit1/Amps", currentAmps1);

firebase.setFloat("circuit1/TimeStamp", now);

// Accumulate daily totals

dailyTotalPowerWatts1 += powerWatts1;

dailyTotalCurrentAmps1 += currentAmps1;

dailyTotalEnergyKWh1 += energyKWh1;

...

// Send daily averages to Firebase once every 24 hours

if (currentMillis - lastDailyUpdateMillis >= dailyUpdateInterval) {

double dailyAvgPowerWatts1 = dailyTotalPowerWatts1 / dailyMeasurementCount;

double dailyAvgCurrentAmps1 = dailyTotalCurrentAmps1 / dailyMeasurementCount;

double dailyAvgEnergyKWh1 = dailyTotalEnergyKWh1 / dailyMeasurementCount;

...

// Update Firebase with daily averages

firebase.setFloat("averages/circuit1/AvgWatts", dailyAvgPowerWatts1);

firebase.setFloat("averages/circuit1/AvgKwh", dailyAvgEnergyKWh1);

firebase.setFloat("averages/circuit1/AvgAmps", dailyAvgCurrentAmps1);

...

}

}

### B.4- Rules & Configuration

{

"rules": {

".read": true,

".write": true

}

}

### B.5- Model Class for Live Data

public class ElectricModel

{

[JsonProperty("Amps")]

public decimal Amps { get; set; }

[JsonProperty("CircuitName")]

public string CircuitName { get; set; }

[JsonProperty("ID")]

public int ID { get; set; }

[JsonProperty("Kwh")]

public double Kwh { get; set; }

[JsonProperty("TimeStamp")]

public long TimeStamp { get; set; }

[JsonProperty("Watts")]

public double Watts { get; set; }

}

### B.6- Setting Up Real-Time Data Fetching

public ElectricService()

{

// Initialize and start the timer to fetch data every second from both circuits

timer = new Timer(async \_ =>

{

await FetchDataPeriodically("https://wattwatcher-pro-default-rtdb.firebaseio.com/circuit1.json", 1);

await FetchDataPeriodically("https://wattwatcher-pro-default-rtdb.firebaseio.com/circuit2.json", 2);

await FetchDataPeriodically("https://wattwatcher-pro-default-rtdb.firebaseio.com/circuit3.json", 3);

await FetchDataPeriodically("https://wattwatcher-pro-default-rtdb.firebaseio.com/circuit4.json", 4);

}, null, TimeSpan.Zero, TimeSpan.FromSeconds(1));

}

B.7- Continues Data Retrieval

private async Task FetchDataPeriodically(string apiUrl, int circuitId)

{

try

{

HttpResponseMessage response = await client.GetAsync(apiUrl);

response.EnsureSuccessStatusCode();

string responseBody = await response.Content.ReadAsStringAsync();

ElectricModel electricData = JsonConvert.DeserializeObject<ElectricModel>(responseBody);

// Invoke the event with circuit ID and data

OnDataUpdated?.Invoke(circuitId, electricData);

}

catch (Exception ex)

{

Console.WriteLine($"Error fetching data for circuit {circuitId}: {ex.Message}");

}

}

### B.8- Real-Time Data Update Event.

### 

// Define an event to notify subscribers of new data

public event Action<int, ElectricModel> OnDataUpdated;

### B.9- Alerting the User of Tariff Updates.

private string GetTariffChangeNotification(TimeSpan currentTime)

{

// Define your tariff change times and rates

var peakStart = new TimeSpan(17, 0, 0); // 5 PM

var peakEnd = new TimeSpan(19, 0, 0); // 7 PM

var nightStart = new TimeSpan(23, 0, 0); // 11 PM

var dayStart = new TimeSpan(8, 0, 0); // 8 AM

var fifteenMinutes = TimeSpan.FromMinutes(15);

// Check if current time is near any of the tariff change times

if (currentTime.Add(fifteenMinutes) >= peakStart && currentTime.Add(fifteenMinutes) < peakEnd)

{

return "Alert: Peak electricity tariff period starting soon at 5 PM. Rate: 30p/kWh.";

}

else if (currentTime.Add(fifteenMinutes) >= nightStart || currentTime.Add(fifteenMinutes) < dayStart)

{

return "Alert: Night electricity tariff period starting soon at 11 PM. Rate: 20p/kWh.";

}

else if (currentTime.Add(fifteenMinutes) >= dayStart && currentTime.Add(fifteenMinutes) < peakStart)

{

return "Alert: Day electricity tariff period starting soon at 8 AM. Rate: 25p/kWh.";

}

return null; // No notification needed at this time

}

B.10- Apex Charts

@if (chartModels != null && chartModels.Any())

{

<**ApexChart** **TItem**="ChartModel"

**Title**="Average Amperage">

<**ApexPointSeries** **TItem**="ChartModel"

**Items**="chartModels"

**Name**="Amperage"

**XValue**="@(e => e.Date)"

**YValue**="@(e => e.AvgAmps)"

**SeriesType**="SeriesType.Bar" />

</**ApexChart**>

}

else

{

<p>Loading data...</p>

}

B.11- User Defined Threshold for Electricity Consumption

if (costLimit > 0 && !alertSent && Math.Abs(totalCost - costLimit) / costLimit <= 0.1m)

{

SmsService.SendSms("+353871146998", $"Alert: Your LIVING ROOM electricity cost this week is €{totalCost:N2}, which is within 10% of your set limit of €{costLimit:N2}.");

alertSent = true;

}

### GitHub Link for Project

<https://github.com/L00160463/WattWatcher>

## A screenshot of a graph Description automatically generatedAppendices C: Additional Images

Figure 38 Full View Circuit Details

A screenshot of a web page

Description automatically generated