THESIS REPORT

Evaluating Energy Efficiency in IoT Systems: A Comparative Analysis of Cloud and Hybrid Edge-Cloud Processing Models Towards Green Computing

Reported By: Syeda Erma Jawaid

Supervisors: Dr. Soodamani Ramalingam and Dr. Zoe Jeffrey

Word count: 26,360

Submitted to the University of Hertfordshire in partial fulfilment of the requirement of the degree of Master of Science by Research.

Dated: March 2026

Declaration Statement

I certify that the work submitted is my own and that any material derived or quoted from the published or unpublished work of other persons has been duly acknowledged (ref. UPR AS/C/6.1, Appendix I, Section 2 – Section on cheating and plagiarism)

Student Full Name: Syeda Erma Jawaid

Signed: 

Date: March 2026

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BME	Bosch Environmental Sensor Module
CDP	Carbon Disclosure Project
CO_2e	Carbon Dioxide Equivalent
CPU	Central Processing Unit
DB	Database
EJS	Embedded JavaScript
ESP32	Espressif 32-bit Microcontroller
GHG	Greenhouse Gas
HTTP	Hypertext Transfer Protocol
I2C	Inter-Integrated Circuit
INA219	Current/Power Monitoring Sensor Module
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
kWh	Kilowatt-hour
mA	Milliampere
mW	Milliwatt
RAM	Random Access Memory
SBTi	Science-Based Targets initiative
SI	International System of Units
TCP/IP	Transmission Control Protocol/Internet Protocol
UI	User Interface
V	Volt
Wi-Fi	Wireless Fidelity

Table of Contents

Table of Tables

Table of Figures

Abstract

This study investigates the optimization strategies and green computing approaches for energy efficient data processing in IoT-cloud systems. The purpose of this investigation is to enhance the sustainability of IoT-cloud ecosystems by exploring existing methodologies and evaluating their effectiveness. Internet of Things (IoT) has been facing increased energy consumption concerns, especially in cloud-based data processing systems with its rapid expansion. This paper examines energy-efficient data handling in IoT-cloud ecosystems by comparing traditional cloud-based processing with hybrid edge-cloud computing to seek the optimal model for reducing energy consumption and environmental impact. A weather monitoring system, with ESP32 microcontrollers and environmental sensors, is used as a testbed simulating data intensive IoT operations. Research has focused and researched on three main areas: energy efficiency, the carbon footprint and the performance of systems, particularly how hybrid processing can reduce the energy and carbon footprint related to cloud computing. Within the analysis, the estimation of energy and carbon consumption, system responsiveness and the efficiency of task allocation on the kWh consumed was evaluated. Furthermore, scalability simulations of hybrid IoT networks with up to 10,000 devices illustrate the practicality of large-scale systems. The outcomes show substantial improvements to energy efficiency, especially at scale, offered by hybrid edge-cloud models, aiming to the promise for designing sustainable IoT systems. These findings provide empirical evidence confirming hybrid processing as a sustainable substitute to standard cloud architectures. The research concludes by giving recommendations for designing energy-efficient IoT systems and integrating hybrid edge-cloud processing models into future IoT infrastructure towards greener computing solutions.

1. Introduction

IoT has fundamentally transformed how we collect and process data. With billions of interconnected devices that generate massive data streams making it necessary to come up with scalable and energy-efficient solutions for managing these demands. The rapid growth of the IoT has resulted in an urgent need to come up with energy-efficient and scalable solutions to handle the demand for computational and the need for storage. Cloud computing has traditionally served as the backbone for IoT data processing. However, its environmental impact and energy intensity have raised on to the growing concerns. Studies reveal that while cloud models provide powerful computation and flexibility but they often suffer from high latency and excessive energy consumption due to long-distance data transmission and centralized processing [1–6].

Several studies show that the data centers that provide cloud services consume around 1% of global electricity and emitting a considerable amount of CO_2 [7-10]. As per the International Energy Agency (IEA), the global data centers consumed between 220 and 320 terawatt-hours (TWh) of electricity in 2022, which is comparable to the total energy consumption of a medium-sized country [11-13]. This shows that the sustainability challenges of cloud-centric IoT architectures are significant, particularly in data-heavy use cases. In addition to the energy consumed in computation, the cooling systems with no data responsibility can consume as much as 40% of a data center's electricity, worsening the environmental effect [14–17].

Figure 1 depicts massive amounts of IoT data generated from interconnected devices are continuously transmitted to cloud data centers for processing and storage. This process, often referred to as data deluge, creates significant energy loads due to high-frequency, bidirectional communication and heavy server-side computation. The final stage of this flow reflects carbon footprint that is notable as a result of many data centers operating on high energy infrastructures, frequently powered by non-renewable sources like coal or natural gas, especially in developing regions. The cooling systems alone in large-scale data centers can account for up to 40% of total electricity consumption, further emphasizing the environmental strain [16, 17].

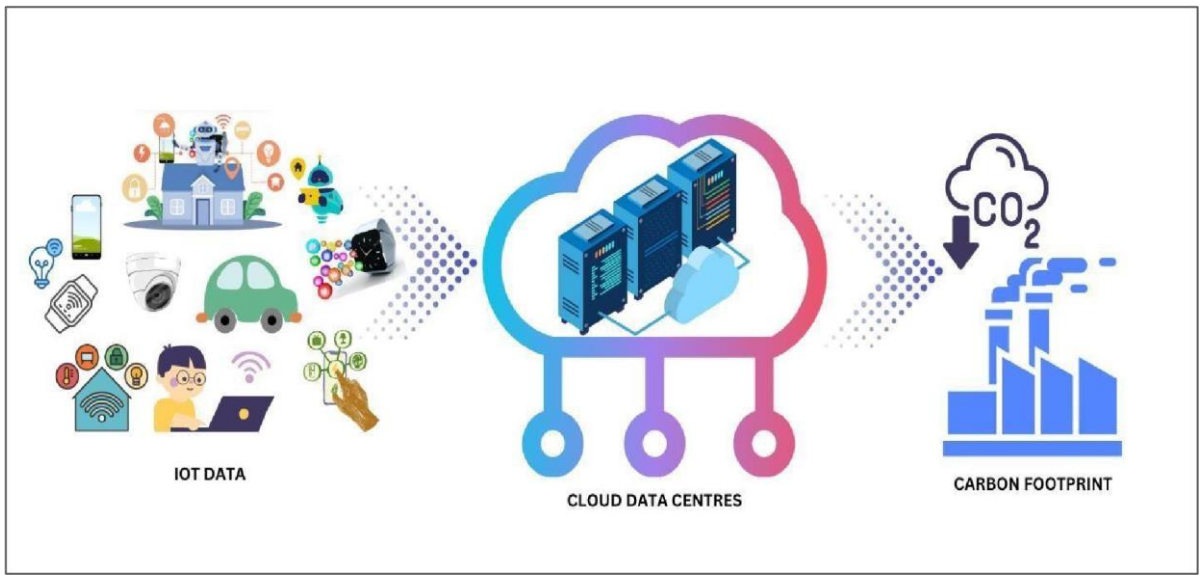

Figure 1. IoT data transmission to cloud data centers increases energy use and carbon emissions, highlighting the environmental impact of cloud - based architectures

While prior research has extensively analyzed the scalability and computational benefits of cloud-based IoT architectures, fewer studies have provided empirical, system-level evidence on how hybrid edge–cloud models perform in real-world scenarios in terms of energy efficiency and carbon impact. Majority of the existing literature focus on simulation-based models or theoretical analysis, often overlooking experimental validation using actual IoT hardware and parallel cloud–edge deployments. Furthermore, the quantitative comparison of carbon emissions between cloud-only and hybrid processing models remains underexplored, particularly within small-scale IoT systems that can serve as prototypes for larger deployments. To deal in with these challenges, hybrid edge–cloud computing has emerged as one of the best approaches to improve energy efficiency by distributing computational tasks between edge devices and the cloud, reducing both latency and overall energy consumption [18].

1.1. Aims and Objectives

The primary aim of this study to analyze energy-efficient data processing in IoT cloud systems to assess the performance disparity between cloud-only and hybrid edge cloud processing models. The research involves creating two separate server instances to ensure a fair and controlled comparison between the two approaches. The study measures data processing load, power consumption, and carbon footprint using a weather monitoring system built with an ESP32 microcontroller, under both models. The experimental setup and system configuration are described in detail in the methodology and system design and development sections.

Through this setup, the research provides empirical outcomes to indicate how hybrid edge cloud processing minimizes energy use and lowers the carbon footprint. The findings contribute to developing more sustainable and efficient IoT architectures for future applications. The goal is not merely to compare, but to demonstrate the impact of hybrid processing at a smaller scale, offering evidence-based reasoning for its feasibility in large-scale IoT deployments. Filling the gap between theoretical and functional, this work demonstrates the possibility of hybrid models as a positive recognition in the quest for energy efficient IoT ecosystems. To achieve the stated, this research is designed around the following objectives**:**

• ***Assessing the Need for Hybrid Processing***

Understanding energy use in cloud-centric IoT systems and identifying how hybrid processing can eliminate the unsustainable approaches endemic in such architectures.

• ***Experimental System Development***

Implementing a IoT weather station using ESP32 microcontrollers and environmental sensors that was serve as a real-world testbed for observing the operational benefits and challenges of hybrid processing.

• ***Comparative Energy Efficiency Analysis***

The analysis of energy use in cloud-only and cloud hybrid edge systems to quantify the potential benefits of edge computing.

• ***Carbon Emission Assessment***

Measure and compare the carbon emissions of both processing models to determine the environmental consequences of each approach.

• ***Performance and Latency Evaluation***

Examine the systems responsiveness, computational efficiency, and overall performance improvements in hybrid processing.

• ***Scalability Testing***

Examine the energy-efficient scalability of IoT networks by simulating device expansion from 100 to 10,000 nodes, representing realistic growth scenarios.

To get an in-depth understanding, the next section reviews existing literature on energy consumption in IoT systems, the challenges of cloud computing, and the potential for hybrid edge-cloud systems to address these issues more sustainably. This review laid the groundwork for the research, outlining the key insights and gaps that this study addresses.

Moreover, The carbon emission reductions in this study were calculated using standards outlined by the Science-Based Targets Initiative (SBTi) and the Carbon Disclosure Project (CDP) [19-22]. These two standards are applied solely to confirm the accuracy of emission units and calculations, as both are based on globally recognized greenhouse gas accounting principles aligned with the Greenhouse Gas (GHG) Protocol. All figures and diagrams depicted in this paper are created by the author to demonstrate the system design, data flow, and experimental setup with clarity and originality.

1.2. Research Contributions

This research makes several contributions across technical, environmental, and academic domains. From a technical standpoint, it provides empirical validation of hybrid edge–cloud architectures using a controlled experimental setup. By directly measuring device energy consumption with INA219 sensors and analyzing real server metrics from Render, the study bridges a gap in the literature, where much of the existing work remains simulation-based.

On the environmental front, the research quantifies emission reductions with standardized carbon intensity factors, linking system-level design choices to tangible sustainability outcomes. This aligns IoT architecture with broader climate goals and adds practical evidence to the discourse on Green IoT. In the academic dimension, the study advances theoretical claims about hybrid processing with concrete, measurable outcomes, demonstrating how efficiency at the device level cascades into savings at the server and system scale. It also frames the discussion of trade-offs, such as data granularity and resilience, as design considerations rather than drawbacks, contributing nuance to the scholarly debate.

Collectively, these contributions position the research as both a validation of hybrid edge–cloud models and a steppingstone toward more comprehensive frameworks for sustainable IoT.

1.3. Structure of the Thesis

This thesis is organized into six chapters, each addressing a specific aspect of the research.

Chapter 1: Introduction

This chapter introduces the background and motivation for the study, highlighting the growing energy demands of IoT systems and the importance of sustainable computing architectures. It presents the research problem, objectives, research questions, and key contributions of the study. The chapter also outlines the scope of the work and introduces the overall structure of the thesis.

Chapter 2: Literature Review

This chapter reviews existing research related to IoT energy efficiency, green computing, edge computing, and cloud-based architectures. It examines current approaches to reducing energy consumption and carbon emissions in IoT systems, while discussing the limitations of cloud-only architectures. The chapter identifies research gaps and establishes the theoretical foundation for the proposed hybrid edge–cloud approach.

Chapter 3: Research Methodology

This chapter explains the research design and experimental framework adopted in the study. It describes the hardware and software components used to build the IoT weather monitoring system, including sensor selection, edge device configuration, and cloud infrastructure. The chapter also details the data collection process, performance metrics, and validation methods used to evaluate system performance.

Chapter 4: System Design and Implementation

This chapter presents the design and implementation of the proposed IoT monitoring system. It describes the architecture of both the cloud-only and hybrid edge–cloud processing models, including the edge layer, cloud layer, and user interface components. The chapter also explains how data flows through the system and how energy consumption and performance metrics are recorded.

Chapter 5: Experimental Analysis and Results

This chapter presents the results obtained from the experimental evaluation of the system. It compares the performance of the cloud-only and hybrid architectures in terms of

energy consumption, carbon emissions, latency, throughput, and server resource utilization. The findings are analyzed to determine the efficiency and sustainability benefits of the hybrid processing approach.

Chapter 6: Scalability, Discussion, and Conclusions

This chapter extends the experimental findings by examining how the system behaves under larger-scale deployments. It discusses scalability projections, system limitations, and practical implications of the proposed architecture. The chapter also interprets the results in relation to the research questions and existing literature, highlighting the broader impact of hybrid edge–cloud processing on sustainable IoT system design.

Chapter 7: Conclusion and Recommendations

This chapter summarizes the key findings of the research and highlights the main contributions of the study to energy-efficient IoT system design. It reflects on how the proposed hybrid edge–cloud architecture improves sustainability and system performance. The chapter also discusses the limitations of the study and outlines potential directions for future research.

2. Literature Review

The rapid growth of IoT networks has intensified the need for computing architectures that are not only high-performing but also energy-efficient and environmentally sustainable at the same time While the cloud-centric model of computing offers real-time scaling and centralized processing power, there are still major hurdles in meeting the energy-costs and latency challenges of real-time high-volume data streams. Consequently, the computing and distributed architectures of edge, fog, and hybrid edge-cloud computingare layered as promising models to address the latency and energy trade-off. These approaches enable local computation at the edge, reducing unnecessary data transmission to the cloud and mitigating the environmental footprint of IoT systems. As shown in **Figure 2**, the proposed system employs a two-tier data processing architecture. Data collected from IoT sensors is first transmitted to the edge server, where preliminary processing occurs. At the edge, the system performs tasks such as filtering noise, aggregating sensor readings, and executing initial computations to reduce data volume and minimize latency. Once preprocessed, the refined data is sent to the cloud server for advanced processing. In the cloud, the data undergoes analytics using machine learning algorithms, long-term storage, and visualization. This architecture

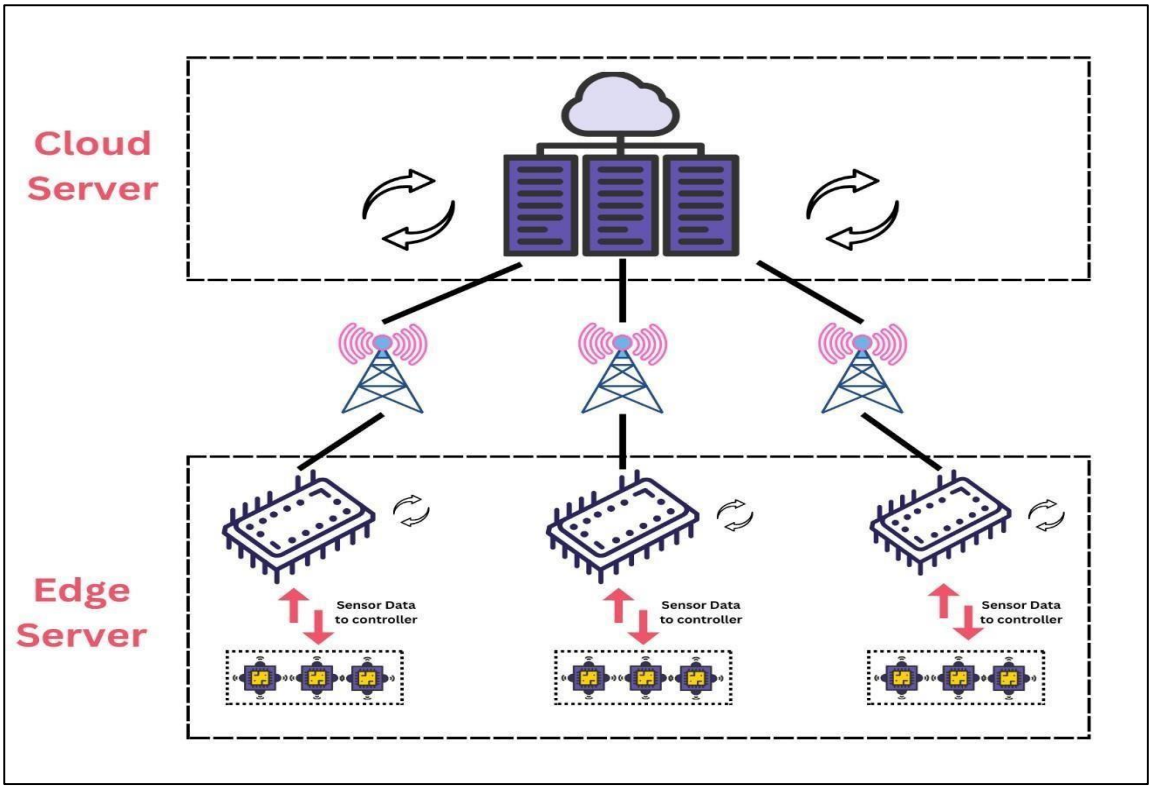

Figure 2. Two-Tiered Hybrid Architecture showing Edge Servers processing Sensor Data

ensures that real-time decisions can be made locally at the edge, while comprehensive analysis and historical insights are handled efficiently in the cloud.

This review summarizes research in four pivotal areas concerning energy-efficient hybrid IoT deployments: hybrid computing models, resource management and task offloading, energy-efficient system design, and frameworks for evaluating systems with respect to emissions. Evaluating both the simulations and the empirical studies, this review details the benefits of hybrid computing and identifies the research gaps, primarily the absence of in-field, device-level energy and carbon costing. These gaps are the motivation behind the present research, which assesses the energy and environmental cost of hybrid compared to cloud-only IoT processing using an ESP32-based weather monitoring instrumented testbed.

The following sections provide more details on specific approaches and insights that guided this study's design and methodology, citing empirical evidence that enhances the study's theoretical frameworks and validates them in practice through real-world IoT applications.

2.1. Layered Computing Architecture for Energy Efficiency

Layered and distributed computing architectures, including edge, fog, and hybrid cloud systems, have emerged as promising approaches for improving energy efficiency in Internet of Things (IoT) environments. These architectures attempt to mitigate several limitations associated with purely cloud-based deployments, where centralized processing often results in high latency, network congestion, and increased energy consumption, particularly for real-time and high-frequency data processing applications.

Fereira et al. [23] proposed an energy-efficient node selection strategy within an edge–fog–cloud IoT architecture. Their work highlights the importance of considering energy cost, workload characteristics, and communication overhead when allocating computational tasks across distributed nodes. Simulation results indicated that dynamic node selection could reduce energy consumption by approximately 30% compared with static allocation strategies while maintaining acceptable task completion times. However, the proposed approach relies heavily on simulation environments and assumes ideal network conditions. In real-world deployments, IoT systems often face unstable connectivity, heterogeneous device capabilities, and unpredictable workloads, which may reduce the effectiveness of such optimization strategies.

Similarly, Cao et al. [24] presented a comprehensive survey on edge and edge-cloud computing architectures for cyber-physical systems. Their study explains the architectural motivation for placing computation closer to the data source and discusses energy-aware scheduling mechanisms that improve responsiveness and reduce communication overhead. While these models demonstrate significant benefits in terms of latency reduction and improved system efficiency, the majority of the studies reviewed in their work remain conceptual or simulation-based. As a result, there is limited empirical validation using physical IoT devices operating under realistic environmental conditions.

Albreem et al. [25] explored the broader concept of Green IoT (GIoT) and examined multiple techniques for energy-efficient IoT deployments. Their empirical findings suggest that adaptive sensing rates and intelligent task allocation strategies can reduce energy consumption by 20–40% across heterogeneous IoT devices without compromising system responsiveness. Despite these promising results, the proposed strategies require sophisticated dynamic management mechanisms and accurate knowledge of network conditions and device capabilities. In large-scale deployments, such dynamic coordination may introduce additional computational overhead and system complexity.

Alharbi and Aldossary [26] implemented an edge–fog–cloud architecture within a smart agriculture environment to evaluate energy consumption under different task allocation strategies. Their experimental results demonstrated that local edge processing significantly reduced energy usage and latency for time-sensitive applications. However, their work primarily focuses on application-specific environments, and the scalability of such architectures across different IoT domains remains less explored. Additionally, their analysis does not explicitly evaluate the carbon footprint implications of these energy savings.

Purwania et al. [27] investigated IoT-based energy monitoring using Arduino and NodeMCU platforms with cloud-centric data processing. Their results revealed considerable operational latency and increased energy consumption due to persistent cloud communication. While the study highlights the limitations of cloud-only architectures, it does not experimentally evaluate hybrid architectures as an alternative solution.

Most notably, Pallikonda et al. [28] developed a hybrid edge-cloud architecture designed for real-time IoT applications, focusing on optimizing both latency and energy efficiency. Their system employs dynamic task partitioning between edge and cloud layers,

prioritizing local processing to reduce communication overhead. Experimental results demonstrated an average latency of 35 ms in the hybrid model compared to 120 ms for cloud-only and 70 ms for edge-only configurations. Energy consumption was measured at approximately 80 J, representing nearly a 55% reduction compared with cloud-only processing. Additionally, task offloading efficiency reached 92%, indicating effective workload distribution across layers.

Although Pallikonda et al. [28] provide valuable experimental insights, their system focuses primarily on performance optimization and does not extensively evaluate carbon emissions or environmental impact. Furthermore, their study does not provide detailed energy measurements at both the edge device and cloud infrastructure simultaneously, which limits a comprehensive sustainability assessment.

Overall, existing studies demonstrate that layered computing architectures can significantly improve energy efficiency by distributing computational workloads across edge and cloud layers. However, several limitations remain evident in the current literature. First, many studies rely heavily on simulation-based models rather than real-world experimental implementations using low-power IoT hardware. Second, the majority of existing work focuses primarily on energy consumption or latency while overlooking the environmental impact in terms of carbon emissions. Third, detailed system-level comparisons between cloud-only and hybrid architectures under identical operational conditions are still limited.

To address these research gaps, the present study adopts an experimental approach using a real IoT weather monitoring system based on ESP32 microcontrollers. By implementing both cloud-only and hybrid edge-cloud processing configurations under controlled conditions, the study directly measures energy consumption, latency, and carbon emissions across both architectures. This methodology provides empirical evidence on the sustainability benefits of hybrid processing models and contributes practical insights into the design of energy-efficient IoT systems.

2.2. Resource Management and Task Offloading for Energy Efficiency

Efficient resource management and intelligent task offloading are essential for achieving energy efficiency and maintaining performance in hybrid edge–cloud systems. These systems depend on the careful coordination of computational workloads between the edge layer,

which operates with limited power and processing capability, and the cloud layer, which provides greater capacity but incurs higher latency and energy use due to data transmission. As a result, several studies have focused on developing adaptive frameworks that determine how tasks should be distributed to make optimum use of available resources while reducing energy consumption.

Han et al. [29] developed a dynamic task allocation strategy that continuously monitors network status, task priority, and device energy levels to determine the optimal execution point for each process. Their findings showed that assigning latency-sensitive tasks, such as video analytics or environmental sensing, to the edge and offloading the cloud computation to edge computation resulted in a reduction of system latency by close to 35% and significant power savings. Nan and others [30] built on this by developing a cooperative resource-sharing construct that supports synchronization between the edge and the cloud layers. Their system uses predictive models for workload forecasting and task reassignment to prevent performance loss and preserve consistent throughput, especially in high-traffic periods.

Li et al. [31] proposed a scheduling algorithm focusing on multi-objective optimization in order to attain an equilibrium in energy consumption, response time, and computation load amongst distributed nodes. Their research illustrates that adaptive scheduling, driven by realtime monitoring of network conditions, has the potential to eliminate even more idle power without compromising system performance in any way. Pérez et al. [32] took a similar approach by employing deep reinforcement learning to design a self-optimizing framework for task placement on edge workloads. Their framework also safeguarded system performance while avoiding energy hotspots, overheating, and power abuse during edge workloads by the incorporation of thermal and power awareness in the decision-making process. Elouali et al. [33] aimed to reduce energy consumption in edge–cloud AI pipelines using unnecessary communication patterns by identifying and eliminating redundant and repetitive data blocks before sending. Their work showed a 28% reduction on the volume of data to be transferred, which directly impacted the energy cost of wireless communication.

Adding to these developments, Saeik et al. [34] also described task-offloading models, categorising them as mathematical, heuristic and AI. They argued intelligent offloading improves not only performance efficiency, but also meets sustainability objectives by decreasing redundant computation and data transfer carbon footprints. Yet, their review

stressed, as to their implementations, most solutions were only tested in high-performance computing or simulated environments and not in low-power IoT systems, as they were designed to do.

Although these studies provide valuable insights, the majority of them depend on simulation or high-performance computing environments which do not consider the constraints of low-power microcontroller-based systems. There remains a lack of empirical validation on systems that operate under real energy and hardware limitations. This research attempts to fill this gap by studying adaptive task offloading and resource management on hybrid edge-cloud systems based on an ESP32. This study also differs from the many theoretical approaches by providing direct empirical evidence on the theory of power and performance under different workloads.

This highlights the potential of intelligent energy-aware management and energy-aware workload distribution on performing sustainability enhancements in distributed IoT systems. The empirical research at the device level illustrates the hybrid computing model's impact on energy and environmental performance.

2.3. Energy Efficiency and Sustainability in Computing Infrastructures

The exponential growth of data-driven services and the (IoT) translates to an ever increasing demand on centralized data centres and underscores the importance of computing infrastructures. The order of magnitude increase in computing resources and carbon emissions tied to the ICT ecosystem does not go unnoticed. Data centres consume an increasingly large portion of resources tied to ICT due to the computing workloads as well as the support resources such as cooling systems, networks, and other peripheral storage systems [35-37].

Katal et al. [38] conducted a more comprehensive survey of the data centre energy efficiency and cloud computing focused on the various aspects associated with software level energy efficiency optimizations, including virtualization, dynamic resource allocation, and workload mechanism. Their analysis merged multiple case studies and simulations and concluded that work adjustments under certain conditions could attain up to a 20–30% reduction on energy consumption. Most of the strategies involved predicting energy server usage and implementing dynamic scheduling. Tasks are completed on fewer active servers and thereby reducing idle energy wastage. While effective at the data-centre layer, Katal et al. highlighted that such strategies provide limited benefits to IoT devices, as the energy spent on

transmitting high-frequency sensor data to centralized clouds remains unaddressed. Al Kez et al. [39] noted the broader sustainability challenges arising from the rapid expansion of digital infrastructure and internet data centres. They examined energy sustainability through a mix of energy models and operational data from commercial data centres to determine energy consumption of various tasks and the involved carbon energy stamped surfaces. Findings provided insight that there were operational and energy efficient hardware improvements, total energy consumption still tracking preserved data remained higher along with the increasingly used centralized computations. The study concluded that lack of architectural strategies that reduce dependency on cloud processing, the environmental footprint of IoT ecosystems was continue to grow. Siddik et al. [40] conducted an in-depth empirical analysis of U.S. data centres where the energy consumption of IT equipment and the supporting infrastructure was measured. They found that the cooling overhead expenses contributed to 30–40% of the total energy consumption, even in the case of energy-conscious servers. This study combined power profiling and thermal modeling to show that energy efficiency improvements at the server level do not guarantee an end-to-end reduction in energy consumption in distributed systems, especially when the large-scale data transfer from IoT endpoints is the dominant factor in the energy budget.

Building on these findings, Thangam et al. [41] investigated the interplay between cloud operations, network-intensive workloads, and climate impact. Their approach involved simulating standard cloud workloads with differing network requirements and analyzing the energy impacts at the data-center and edge levels. Their findings were that the repeated transfer of raw sensor data to central clouds escalated energy consumption at the edge nodes and, therefore, emphasized the necessity of local processing. The study concluded that integrating edge-level computation with cloud infrastructure can meaningfully reduce overall energy consumption and emissions. Zhang et al. [42] focused at the energy dynamics of cooling in vertically stacked server arrangements. By combining real-world temperature measurements with computational load profiling, they determined that optimizing airflow and thermal management could reduce cooling energy by measurable amounts. However, their work also reinforced that efficiency improvements confined to the data centre are insufficient to address the total energy footprint of IoT systems, as the cost of transmitting and processing high frequency sensor data remains substantial.

Collectively, these studies provide a strong reason for moving computation closer to the edge of the network. By performing preprocessing, filtering, or aggregation locally before sending data to the cloud, hybrid edge–cloud designs can reduce transmission overhead, lower latency, and improve energy efficiency at the device level. Despite significant advancements at the infrastructure layer, few studies provide empirical validation of energy savings across the full IoT system, from edge devices to cloud servers. The present research addresses this gap by deploying a hybrid edge–cloud weather monitoring system, directly measuring energy consumption at edge microcontrollers and cloud servers to quantify the impact of task distribution and local processing on overall system sustainability [41,42]. This approach gives concrete evidence for the realistic advantages of energy-aware hybrid architectures in real-world IoT applications.

2.4. Green IoT and Energy-Aware System Design

Green IoT places sustainability and environmental impact the center of system design, including everything from the choice of sensing hardware and communication protocols to decisions about computation placement and lifecycle carbon assessment. The literature in this area covers a growing spectrum that includes optimization models and network protocols, empirical testbeds and lifecycle evaluation, and other methods under the energy-optimized IoT systems paradigm. However, a closer analysis shows that many of these studies stop short of comprehensive, end-to-end empirical validation, particularly with respect to actual device level energy measurements and real-world carbon accounting.

Aldossary and Alharbi [43] approached carbon minimization as an allocation and placement optimization problem across the fog–cloud continuum. Their research used a mixedinteger linear programming model that combined energy and carbon emissions directly as objective functions, constrained by latency and resource availability. To address scalability issues pertaining to large IoT topologies, the authors incorporated both exact and heuristic solvers. The authors undertook a workload trace study, comparing fog-enabled task placement and traditional cloud-only models. Processing data close to its source, while offloading a small portion of the highly computation intensive tasks to the cloud, resulted in significant savings in both transmission and processing energy. Nevertheless, their work remained simulationbased, lacking device-level energy validation that strengthens its applicability to physical sensor networks [43]. In a previously cited article Almudayni et al. took a holistic systems approach by first identifying the root causes of energy inefficiency within IoT architectures before

proposing an integrated framework with adaptive communication, energy harvesting, and lightweight on-node processing. The author used a combination of analytical modeling and small-scale empirical experiments to test adaptive duty cycling and dynamic node scheduling. The study reported measurable reductions in communication energy, with adaptive sampling extending network lifetime by reducing unnecessary transmissions. However, the authors acknowledged that aggressive duty cycling introduced a trade-off between data accuracy and energy savings. Their findings are particularly useful for practical implementations like environmental sensing, where careful balance between fidelity and efficiency is essential.

Izaddoost and Siewierski [44] concentrated on communication efficiency at the lower network layers. They introduced event-driven data reporting and similarity-based suppression techniques to minimize redundant packet transmissions. Their empirical methodology involved hardware prototypes of sensor nodes, through which they measured packet transmission frequency, duty cycle, and radio on-time under various aggregation thresholds. Their experimental outcomes demonstrated substantial energy savings by reducing radio activity, often one of the largest contributors to energy consumption in wireless sensor networks. This work is particularly relevant to weather-station applications, where temporally correlated environmental data allows aggregation without sacrificing data integrity. The work by Rahman et al. [45] integrated energy-efficient design principles from Green IoT to create a domain specific application of hybrid edge-cloud analytics to water quality monitoring. These systems also made sensors and edge gateways that performed localized feature extraction and set anomaly detection parameters before sending summary data to the cloud. In the prototype, three operational modes were compared: edge-only, cloud-only, and hybrid edge-cloud. The hybrid mode improved bandwidth consumption and latency, all while maintaining detection accuracy on par with the cloud mode. This confirmed that hybrid systems provide practical gains in both the expected drop in performance and the energy saved. They studied water
quality, but the sensors developed to monitor it are just as useful for weather stations.

Baldini et al. [46] developed the conversation by incorporating a lifecycle assessment perception into Green IoT. Their study fused operational energy profiling with embodied emissions analysis to evaluate the full environmental impact of deployment strategies. Their approach consisted of estimating the embodied carbon for the manufacturing and maintenance cycles and then integrating it with operational power consumption. They observed that concentrating on operational energy alone could result in erroneous conclusions; for example,

replacing multiple low-power sensors with a single high-capacity edge device might reduce communication energy, but it was also significantly increasing lifecycle emissions because of higher embodied emissions. Their findings stressed the need for a more holistic assessment of Green IoT solutions, which operates primarily on the savings derived from operational energy exhaustion.

Collectively, these studies offer an appreciation of how the operationalization of the principles of Green IoT can be achieved through energy-aware system design. The communication reduction strategies, the need to incorporate carbon intensity and the adaptive task placement for gaining control on the tradeoff between latency and energy use are highlighted as the pivotal elements. They also, however, share a similar drawback. Much of the work in this space is either driven by models and simulations, rather than providing empirical evidence derived from the operation of real, continuous IoT systems.

2.5. Environmental Impact Assessment and Carbon Accounting

Hybrid edge-cloud architectures are increasingly adopted in IoT for effective use of computation, energy savings, and large-scale, real-time data stream processing. These systems distribute workloads among local edge computing devices and central cloud servers, minimizing excessive data transfers, and trimming latency and energy expenditure. This makes such systems green computing compliant [45], [47].

Determining the computing systems infrastructure environmental impacts involves estimating energy use in the total system and translating that into carbon emissions based on the electricity grid carbon intensity. This process enables a realistic assessment of sustainability beyond simple energy metrics. Recent research has addressed this issue from multiple perspectives, including system monitoring, device-level optimization, and carbon-aware scheduling across cloud and edge infrastructures.

Romero et al. [48] developed an open-source IoT-based edge computing framework was created for energy consumption monitoring within buildings. Integrated system architecture includes distributed energy meters with local edge nodes for onsite data aggregation, followed by onsite analytics prior to cloud transfer of summary reports. Methodology involves nonintrusive sensors interfacing with microcontroller-based frames for high-frequency sampling of voltage, current, and power. Analytic modules in Python local to the edge compute and visualize temporal snapshots for estimated energy profile data. Results show that

aggregation at the edge significantly lower data transmission rates while maintaining transmission analytics for the complete data transfer. Real-time edge monitoring provides feedback loops for faster energy optimization, forming the basis for carbon-aware analytics. Nevertheless, the study's carbon analysis focuses on aggregate estimates rather than dynamic assessment of measured energy and variable grid emissions, pointing to the absence of practical real-time carbon accounting.

Chang, Wu, and Lin [49] established an ESP32-based edge computing architecture to carry out object detection, demonstrating the capability of low-power microcontrollers to handle complex assignments. For the experimental approach, they deployed a portion of the workloads to a neural network on an ESP32 and the rest on a cloud node and recorded the span of time taken to process and communicate the data, and the energy consumed at the device. Their implementation for the evaluation had several datasets and hardware configurations to analyze the trade-offs between energy saving and the accuracy of computation. Their findings suggest that edge-local inference within a node can substantially save network energy and latency while preserving detection accuracy within reasonable limits. The work confirms the potential of lightweight edge processing in accomplishing both performance and energy efficiency, yet it mainly reports electrical energy metrics and omits a systematic translation of these energy measurements into carbon emissions, leaving the environmental point implicit rather than quantified.

Leelavinodhan et al. [50] designed and implemented an energy-efficient weather station for wind data collection that applies edge computing principles to environmental monitoring. Their methodology integrates adaptive sampling, low-power sensor interfacing, and eventdriven data transmission for efficient energy minimization. Real environmental conditions were used to conduct experiments for measuring operational energy savings and optimized sensing intervals and edge-based data processing. Selective data acquisition and transmission was found to reduce system energy consumption by nearly 35% in comparison to continuous sampling. While the study clearly illustrates energy savings in environmental monitoring, there is no analysis of the carbon impact of these savings, which in turn hinders the ability to evaluate overall sustainability.

Asadov et al. [51] explored the concept of carbon-aware spatio-temporal workload shifting within edge–cloud environments. Their research introduced a model that dynamically

relocates workloads to data centres or edge nodes based on regional carbon intensity and real time energy demand. This method combines forecasting carbon emissions in the power grid with the emission-laden task scheduling to comply with service-level agreement (SLA) requirements. The proposed method was simulation validated on real grid data and benchmarked workloads. The study's primary finding is that carbon-aware schedulers allow up to 18% emissions reduction at relocation as compared to dispatching relaxed-time scheduling. One limitation is that the study is simulation based and does not use primary energy data from hardware IoT devices, which is crucial for grounding the theory in real sensor-based systems.

Sukanya et al. [52] introduced Green Task, a carbon-aware scheduling algorithm aimed at fog and edge computing settings. Their approach integrates energy consumption paradigms with predicted carbon emission values for scheduling tasks. They focus on executing tasks on nodes that are cleaner and in periods with lower emissions. During the evaluation, benchmark applications were implemented to analyze the performance of the schedulers. This was done under a range of energy and carbon constraints. Findings showed the carbon-aware scheduling succeeded in reducing emissions compared to strategies that optimized for energy alone. However, emission reductions were still in the range predicted. Similarly to Asadov et al., the reliance on modeled estimates instead of actual data is a drawback. The assumption of dependable carbon intensity data is likely unreasonable for distributed IoT networks, which need continuous sensing and real-time responses.

Collectively, these studies demonstrate an evolving methodological progression toward integrating carbon accounting with energy monitoring in distributed and hybrid computing environments. They establish fundamental techniques such as localized energy measurement [48], microcontroller-based efficiency optimization [49], and adaptive sensing [50], alongside algorithmic approaches for carbon-aware workload distribution [51] and task scheduling [52].

2.6. Summary of the Literature Review

The reviewed literature determines a clear understanding of how energy efficiency challenges occur in (IoT) systems due to the exponential growth of connected devices and their reliance on centralized cloud setups. Cloud computing, while enabling vast scalability and storage capabilities, presents high latency, network dependency, and substantial energy consumption driven by continuous data transmission and intensive computation at remote data centers.

On the other hand, edge computing offers a distributed solution that brings computation closer to the data source, thereby minimizing latency and reducing transmission energy. However, edge devices cannot take on self-sufficiently large or complex workloads due to limited on-board processing and storage. This lack of balance has caused the formation of hybrid edge cloud systems, which allocate processing workloads between edge devices and the cloud.

The literature consistently stands for hybrid architectures as a potential solution to balance energy efficiency and system scalability. A large number of studies use simulation-based analyses or mathematical modeling to demonstrate the potential of hybrid models in minimizing power consumption and improving latency. Yet, a persistent gap remains in empirical, hardware-based comparisons that quantify the real-world performance differences between cloud-only and hybrid edge–cloud systems under identical operational conditions.

2.6.1. Identified Research Gaps

Despite of the vast research on hybrid edge–cloud IoT architectures, several gaps remain that motivate the present study. Most previous works focused heavily on simulations or theoretical approaches and have not demonstrated, on an empirical basis, the differences in energy consumption between the cloud-only and hybrid processing approaches. Furthermore, not many studies have accounted for the effect of the integrated carbon in their assessments, thus leaving the energy cost savings' environmental effect, the most considerable part, unassessed. Useful practical testbeds for IoT applications are limited, especially those allowing data streams from sensors for energy consumption evaluation in a real-world setting. In addition, most studies continue to provide aggregated results whereby the detailed energy consumption of devices at edge nodes and cloud servers, especially for cloud and edge nodes, are overlooked. Lastly, comparative studies at a small experimental scale, which can serve as prototypes for future hybrid large-scale IoT networks, are not explored.

Therefore, the literature calls for a comparative experimental study to address the energy efficiency, carbon footprint, and operational performance of cloud-only versus hybrid edge–cloud processing models for real IoT hardware to advance scholarly work in this area. This research, hence, sets up a weather monitoring testbed using ESP32 microcontrollers to facilitate controlled experiments and enable direct power measurements to address testbed limitations in previous works. The results are then compared across both processing paradigms to determine

which architecture achieves superior energy efficiency and a lower carbon footprint in realtime IoT environments. This comparative analysis effectively bridges the gap between the theory and practical experimentation while proving how hybrid edge–cloud structures can offer actionable sustainable computing solutions for scalable IoT ecosystems.

2.7. Research Questions

The insights and gaps described in the previous sections form the basis of a set of research questions that this study proposes to guide the comparative evaluation of cloud-only and hybrid edge–cloud processing models. These questions are directed toward measuring the impact of hybrid architectures on energy use, carbon footprint, and overall performance in operational IoT settings. The answers to these questions are fundamental to the research in demonstrating the energy-efficient and sustainable design of IoT systems on a practical and theoretical level. The research questions are as follows:

1. How does hybrid processing model in IoT systems impact overall energy consumption compared to the traditional cloud-centric IOT models?
2. What are the quantifiable reductions in carbon emissions achieved through the incorporation of edge computing in cloud-only IoT architectures?
3. How does the scalability of cloud-only versus hybrid processing models impact energy efficiency in large-scale IoT deployments?
4. What are the trade-offs in performance, energy savings, and computational efficiency between cloud-only and hybrid processing models in IoT system applications?

Addressing these key questions was achieve the research aim and more importantly give the conceptual and practical basis for the implementation of hybrid IoT ecosystems that are energy efficient. This was aim at understanding the expected model outcomes on energy efficiency in real-time processing, operational scalability, operational sustainability (in terms of carbon emissions), and system scalability to real-world challenges.

3. Research Methodology

Following the background, literature review, and system architecture described in the previous chapters, this chapter outlines how the research was practically implemented. The central aim of the study is to explore and evaluate energy-efficient processing in IoT-cloud environments. To meet this goal, a hands-on, real-world experimental approach was adopted, moving beyond simulation or purely theoretical methods. To achieve this, a real-world experimental approach was adopted, going beyond simulations or theoretical modeling to generate direct, measurable evidence. The methodology centers on a functioning IoT weather monitoring system equipped with multiple sensors and powered by an ESP32 microcontroller. This setup generates real-time environmental data, which is then processed using two configurations: a cloud-only model and a hybrid edge-cloud model. The dual configuration enables a comparative evaluation of energy consumption, system responsiveness, and carbon emissions under identical conditions, offering a realistic and controlled basis for performance analysis.

Importantly, the use of a physical system allows us to rely on actual hardware for behavior and power consumption for various factors that tend to be generalized or overlooked in simulation studies. This conveys real implications of the data-processing choices and is consistent with the aim of the study in determining sustainable and scalable frameworks for IoT deployments.

To confirm clarity and reproducibility, the rest of this chapter is organized into several key sections. It begins with an overview of the research design, followed with detailing the hardware and software elements, experimental setups, methods for measuring power and carbon, data collection, performance evaluation, and validation. All of these elements define the research framework and prepare the implementation and analysis for the following chapter.

3.1. Proposed Research Framework

This section describes the flow of the study including all the inputs, all the steps, and the study results. **Figure 3** illustrates the overall research framework used to evaluate energy-efficient IoT data processing. The framework is organized into three main stages: inputs, processing, and outcomes.

In the input stage, the system begins with the development of an IoT weather station consisting of multiple environmental sensors connected to an ESP32 microcontroller. These sensors generate real-time environmental data. Network connectivity is then established between the edge device and the cloud server to enable data transmission. The cloud infrastructure is configured to receive and store sensor data, while lightweight data-processing algorithms are developed to support filtering and aggregation of sensor readings. These algorithms are particularly relevant for the hybrid edge–cloud model. At this stage, the research assumptions and experimental setup are also defined.

The processing stage begins with data collection from the IoT sensors. The collected data is processed using two different approaches: cloud-only processing and hybrid edge–cloud processing. In the cloud-only model, raw sensor data is transmitted directly to the cloud for processing and storage. In the hybrid model, basic preprocessing such as filtering and aggregation is performed at the edge before sending the data to the cloud. For both approaches, energy consumption and carbon emissions are calculated, followed by a comparative analysis and hypothesis testing to evaluate the effectiveness of each architecture.

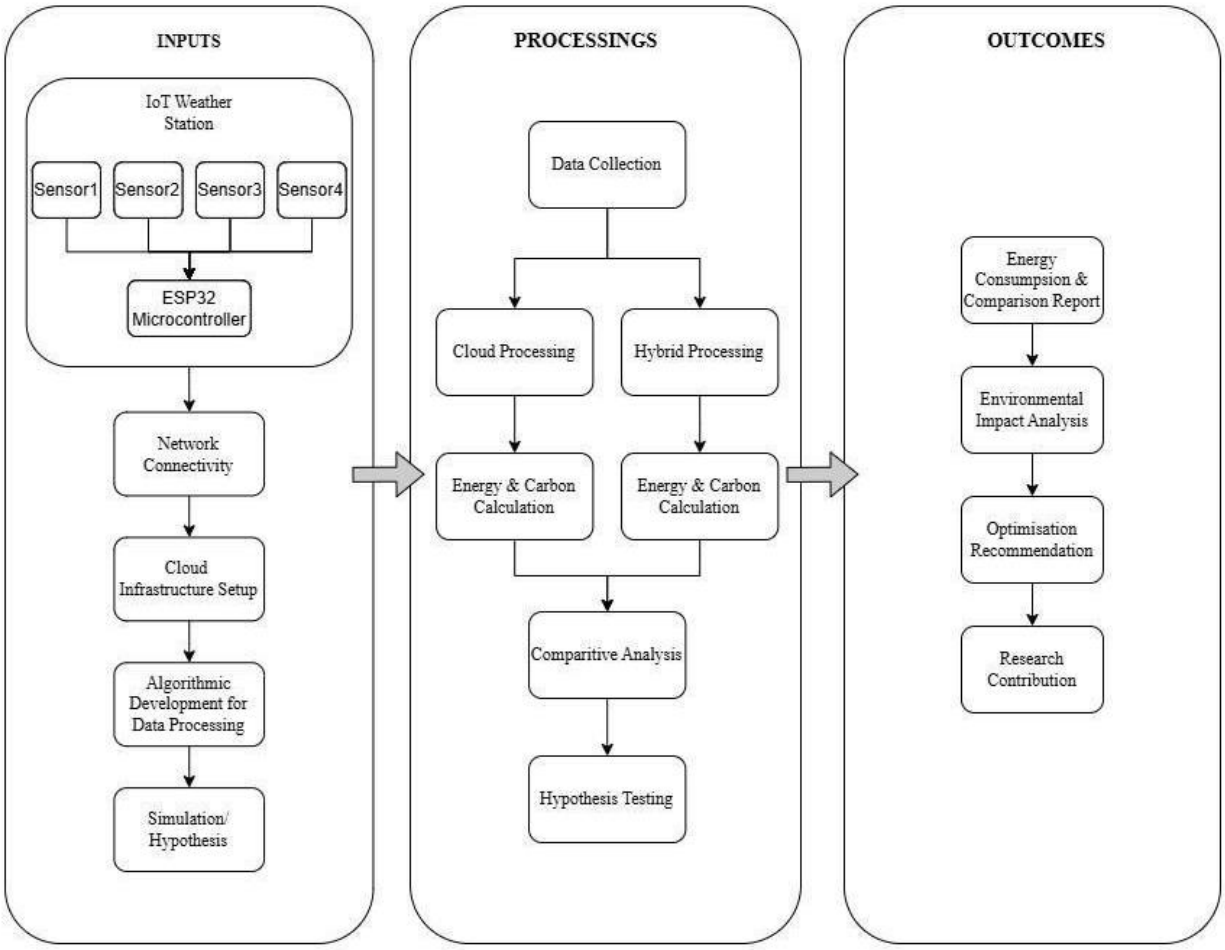

Figure 3. Research Framework for Energy - Efficient IoT Data Processing

The outcome stage presents the results of the study, consisting of an energy consumption, an environmental impact-based CO2 emissions report, and an analysis on the two processing models as well as recommendations for the energy efficient IoT-cloud architectures. These are reported to show the research on the sustainability and scalability of these IoT systems.

3.2. Research Design Overview

This study employs an experimental comparative research design to examine the energy efficiency of IoT data processing models under real-world conditions. At the center of this design is a fully functional weather monitoring system developed using an ESP32 microcontroller integrated with multiple environmental sensors. The BME680 sensor measures temperature, humidity, and pressure, while additional sensors are used for rainfall detection, SI1145 for UV light intensity measurement, and INA 219 for power monitoring. The goal is to understand the cloud-only and hybrid edge–cloud models in terms of energy consumption, latency, and carbon emission trade-offs. Having the weather monitoring system in operation allows for empirical testing of the system energy trade-offs and carbon emissions of these models.

In the cloud-only model, all sensors' data are sent to a cloud server on Render, using a Node.js and Express.js backend. The cloud does all the data processing, visualization, and storage, and even an edge device doesn’t do any processing, illustrating a basic centralized IoT system where the edge device only collects data.

The hybrid edge–cloud configuration, on the other hand, integrates a preprocessing phase. The ESP32 performs preprocessing tasks such as filtering, reducing noise, averaging, and aggregating data before sending the summarized details to the same cloud server. This technique streamlines the data, reducing the amount of data to be sent, optimizing data communication, and saving energy. The responsiveness of the entire system improves. The difference between the two models resolves one of the principal architectural questions in the IoT literature i.e. how to balance the computation between local and remote nodes, in a way that it saves energy while performing to the required standards.

To provide a fair basis of comparison, both designs used the same hardware and software tools. The sampling rate, Wi-Fi signal strength, data packet size, voltage supply, and

communication protocol (which was HTTP over TCP/IP) were all fixed. The ESP32 was powered via a USB-C breakout board connected to an INA219 DC current and bus voltage monitoring sensor, which measures voltage and current in real time to estimate power consumption. These measurements form the basis for quantitative analysis of energy consumption [53, 54].

The experimental construction was situated in a semi-outdoor environment. Sunlight, moisture, and temperature were integrated to appreciate the variability in the environment. This facilitated the sensor behavior analysis in a more integrated manner, rather than restricted to the confines of a laboratory. The library of real time monitoring tools enabled the integration of CPU load, memory utilization, network rate, and power draw, for a more holistic real time analysis. This is often too complex for simulation-based approaches.

The design supports structured hypothesis testing, addressing the research questions through quantifiable performance metrics. Data on energy consumption, processing latency, CPU utilization, data transfer volume, and estimated carbon emissions are collected and compared between the two models. It is hypothesized that the hybrid edge–cloud model was demonstrating notable reductions in energy use and carbon footprint while maintaining comparable computational performance to the cloud-only system.

Adopting a live experimental approach strengthens the validity and reliability of the study's outcomes. This directly supports the research goal of creating practical sustainable and eco-friendly IoT data processing infrastructures. This approach, with real energy quantification and authentic conditions, closes the gap between theoretical models and experimental applications. The design's critical elements are detailed in the following chapters, including the selection of hardware, the configuration of software, the workflow of data processing, the methodology of energy and carbon accounting, the procedures of data collection, and the criteria of performance evaluation, which together define the methodology for the analysis and results chapters.

3.3. Hardware and Software Configuration

In alignment with the planned experimental framework, a practical system comprising both software and hardware components to capture ideational and monitoring energy usage in real time was constructed. Central to the hardware was the ESP32 microcontroller. The

microcontroller's dual-core processing capabilities, Wi-Fi integration, and low power consumption inseparably make it suitable for edge computing and the (IoT) [38-49]. The ESP32 operates as the primary edge device and, in the hybrid configuration, performs sensor data collection and basic data processing. The primary edge device is the ESP32, while the hybrid configuration is where basic data processing (e.g., filtering, aggregation) is executed. Its ability to program deep sleep and utilize interrupt driven GPIOs ensures low-energy modes during idle periods. This feature was indispensable in realistic energy profiling.

The configuration of the environmental sensor suite, the BME680, Rain Sensor, and the SI 1145 modules integrate over I2C and through analog connections. The suite also includes an INA219 DC current and bus voltage monitor, used to measure real-time voltage and current draw of the ESP32. The module supports a 0–26 V bus voltage range and up to 3.2 A current measurement, operating with 3–5 V logic, making it suitable for low-power IoT power monitoring. This information is crucial for assessing power consumption during different operating stages and for time-based energy consumption profiling. The sensor layout and wiring are consistent for both experiments, which guarantees that no differences in power consumption result from hardware. The next chapter system design and implementation was providing detailed description of the sensors.

The physical infrastructure is also complemented by a unique Node.js server developed for real-time ingestion, processing, and visualization of data coming from the ESP32. Node.js is efficient for real-time processing and non-blocking I/O and for managing multiple connections simultaneously. Node.js supports real-time handling and non-blocking I/O, allowing efficient management of multiple connections. A locally deployed Node.js server enables precise monitoring of energy consumption and performance without interference from built-in optimizations. This also complements the evaluation of hybrid processing models, as the study stated Node.js is effective in creating scalable edge computing systems [55]. The server architecture integrates RESTful APIs for incoming HTTP requests and WebSocket protocols for maintaining persistent connections to the browser dashboard. This design maintains live data streams and auto-updating front-end elements without manual refreshes.

A MongoDB database functions as the backend storage solution due to its document oriented structure and flexibility in handling time-series sensor data. For performance comparison purposes, cloud-only processing and hybrid processing sessions data are stored in separate collections. The client-facing dashboard is implemented using EJS (Embedded

JavaScript) templates, allowing dynamic content rendering and conditional formatting of metrics such as temperature trends, data packet sizes, and energy readings.

To monitor the computational overhead on the cloud side, server-level telemetry is enabled using system monitoring tools integrated into the Node.js backend. These include monitoring the median logs of the CPU's load in real time, monitoring memory, tracking data throughput, and monitoring uptime and crash logs. These performance metrics are recorded with energy parameters to assess how varying volumes of data and complex processing affect the stability of the cloud system and its power consumption.

Overall, this hardware and software setup creates a robust, repeatable test environment for comparing the two IoT processing models. Performance differences are isolated to the processing strategies under evaluation, since the test environment uses the same hardware, standardized data intervals, and consistent software workflows. This setup not only supports energy measurement but also offers insight into broader system-level behaviors such as latency, network load, and resource utilization, hence making it an ideal foundation for the experimental approach presented in this research.

3.4. Cloud-Only Mode Experimental Setup

In the cloud-only setup, the system mirrors a conventional IoT architecture, wherein all raw sensor data is transmitted directly from the edge device to a centralized cloud server for processing, analysis, and storage. This model exemplifies the "thin-client" approach commonly used in IoT networks and provides a baseline for comparisons with the hybrid edge-cloud model.

In this configuration, The ESP32 microcontroller has been configured to function as a passive data acquisition and data transmission unit in this case. Every 1 seconds, a sampling cycle is activated, capturing basic data from the integrated environmental sensors. These values are covering parameters such as temperature, humidity, atmospheric pressure, UV light intensity, and rainfall are bundled into JSON packets with a timestamp and a unique device ID for traceability. Using HTTP POST requests, data packets are sent to a remote Node.js server in the cloud through Wi-Fi. The cloud server animating the entire workflow is processing this incoming data stream and operates the full processing workflow. Each packet fetched undergoes data parsing and validation, followed by entry into a MongoDB database. The server then updates the real-time visualization dashboard, which displays environmental trends and

historical logs. In parallel, the server records system-level telemetry, such as CPU utilization, memory usage, response time, and I/O operations. These metrics are important for understanding the resource overhead and scalability constraints of fully centralized data processing in IoT deployments.

The INA219 power monitoring sensor tracks energy usage even when ESP32 isn't processing any data in the minimal computational role. This enables precise logging of voltage, current, and power draw, even in its minimal-processing role. Monitoring energy usage in this case is important to determine the power profile baseline for the ESP32 in sensing and transmission only mode. On the cloud side, the Node.js server receives the data and saves it in a MongoDB database after parsing it. The data is visualized in real time, with a dashboard built using dynamic EJS templates and WebSocket support for live updates. The server also tracks performance in terms of processing load to determine the impact of central computing. Metrics logged include CPU usage, memory usage, and time active.

In order to reduce variation and maintain consistent measurement conditions across all trials, several parameters are standardized throughout the cloud-only experimental sessions. These include maintaining a constant Wi-Fi signal strength and SSID, using a fixed data transmission interval of 1 seconds, standardizing the sensor polling order, applying a uniform server architecture and database schema, and keeping the data payload structure (fields and sizes) consistent. With these parameters, a number of important performance metrics can be observed in isolation, including the system's dependency on the network, the system's bandwidth utilization, the idle and active transmission phase energy expenditure of the ESP32, the processing load and memory consumption of the cloud server and the end-to-end latency from data generation to real-time dashboard display.

By providing a stable reference point, the cloud-only mode allows for quantitative comparison with the hybrid model. Any change in the subsequent edge-processing setup, in terms of energy reduction, network congestion, and responsiveness can be attributed to edge level computing and not disparate environmental conditions or poorly controlled test conditions.

In summary, this configuration highlights the limitations and resource intensiveness of traditional cloud-dependent IoT systems. These considerations are essential for motivating the shift toward hybrid architectures, which aim to optimize energy and performance by offloading

selected tasks to the edge. In the following section explores the hybrid edge-cloud model, where on-device preprocessing transforms both the operational and environmental efficiency of the overall system.

3.5. Hybrid Edge-Cloud Mode Experimental Setup

In this hybrid edge-cloud configuration, the system is designed to implement part of the data processing from the cloud to the edge device. This approach tackles the issues of energy consumption, latency, and bandwidth of typical cloud-centric IoT architectures. This configuration is based on the same hardware and network settings as the cloud-only model to maintain consistency, but a fundamental difference exists in how data is processed before transmission.

Here, the ESP32 microcontroller performs lightweight edge preprocessing before forwarding sensor measurements to the cloud. After sensor acquisition, the collected readings undergo local filtering, averaging, and outlier detection operations. To reduce short-term fluctuations and measurement noise, a moving average filter is applied to the sensor stream. For a sequence of measurements $x_1, x_2, \ldots, x_n$, the filtered value at time t is calculated as:

$$MA_t = \frac{1}{N}\sum_{i=t-N+1}^{t} x_i$$

Where MA_t represents the filtered output, Ndenotes the averaging window size, and x_i represents individual sensor measurements.

The ESP32 then performs interval-based aggregation by calculating representative statistical measures over the collected readings. The arithmetic mean is computed as:

$$x = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Where $\bar{x}$ is the interval mean and n is the number of samples collected during the aggregation period. To minimize the influence of abnormal measurements, a threshold-based outlier detection mechanism is applied. A measurement is classified as an outlier when:

$$| x_i - \bar{x} | > k\sigma$$

Where σ is the standard deviation of the measurement set and k is a predefined threshold coefficient. Measurements identified as outliers are excluded from the aggregation process prior to transmission.

Instead of transmitting all raw sensor samples individually, the ESP32 transmits only the aggregated statistical summaries, including mean, minimum, maximum, and variance values. For example, multiple sensor readings collected during a monitoring interval can be represented by a single summarized packet. This significantly reduces the number and size of transmitted packets, lowering network bandwidth utilization, wireless airtime, and communication-related energy consumption. In addition, it reduces the preliminary computation that the cloud server has to do, leaving it with the other tasks of data storage, historical data tracking, and real-time data visualization.

As in the previous mode, the cloud server (Node.js) receives and stores the processed data into MongoDB, updates the dashboard via EJS and WebSocket, and logs its internal performance metrics that are namely CPU load, memory usage, and uptime for subsequent analysis. However, compared to the cloud-only setup, the computational footprint on the server is reduced due to the lower data inflow and the need for less parsing or transformation.

The INA219 sensor continues to measures the voltage, current, and power consumption of the ESP32 in real time. But in this case, the recorded power consumption still incorporates the ESP32's overhead for communication and the expenses for local computation. This allows the study to examine the energy trade-offs related to edge processing: whether the increased computation on the microcontroller drives a net gain in energy efficiency by lower transmission energy, decreased load on the cloud server, and reduced energy consumption on the microcontroller.

Although the experimental evaluation was conducted under controlled laboratory conditions, several practical deployment factors should be acknowledged. Real-world IoT systems may experience intermittent network connectivity, temporary packet loss, and cloud service interruptions that can affect data transmission reliability. In addition, long-term deployment may introduce battery degradation and sensor drift, which can influence device energy consumption and measurement accuracy over time. These factors were outside the scope of the present study but should be considered in future large-scale and long-duration deployments.

This configuration enables the investigation of multiple performance dimensions, including:

1. The effectiveness of data volume reduction techniques on network and energy efficiency

2. The added power burden (if any) introduced by edge-level filtering and aggregation
3. The reduction in cloud server processing demands due to pre-processed input
4. Changes in data latency, especially in terms of time-to-visualization on the dashboard
5. Bandwidth savings from condensed data transmissions

By conducting this experiment on cloud-only mode under the same conditions as closed cloud mode on the same hardware, cloud server infrastructure, and fixed sampling intervals, the research achieves a fair and direct architectural comparison. This is crucial to determine the impact of hybrid edge-cloud processing on energy efficiency and system performance in the environmental monitoring use case of data-heavy IoT.

3.6. Power & Carbon Emission Calculation Method

Accurate measurement of power consumption and environmental impact was central to the experimental design of this study. Since the core objective was to evaluate energy efficiency in IoT-cloud architectures, both electrical power usage and estimated carbon emissions were carefully monitored throughout the testing period for each processing model. These measurements enabled a direct comparison between the traditional cloud-only approach and the hybrid edge-cloud model in terms of sustainability and operational cost-effectiveness.

On the edge of the system, the ESP32 microcontroller had the INA219 current sensor attached to it. This sensor measured voltage (V) and current (A) in real time. [53]. The INA219 used the I2C interface and with a precision shunt configured to ensure close monitoring even during low-load scenarios which is common in IoT systems [54]. The INA219 module was selected as it is commonly used for low-power embedded energy monitoring and provides stable real-time measurement of bus voltage and current within IoT-based systems [53,54]. At the same time, as with any embedded monitoring sensor, a small degree of reading variation can naturally occur during operation. In this setup, such variation may result from minor sensor tolerance, low-current conversion sensitivity, and slight electrical fluctuations when the ESP32 repeatedly shifts between sensing, local processing, and Wi-Fi transmission. Since the INA219 communicates through the I²C interface, negligible communication disturbances may also produce minor differences between individual samples [53].

These small variations do not materially affect the comparative analysis carried out in this study. This is because the same INA219 sensor, identical wiring arrangement, same power supply, and unchanged I²C settings were consistently used in both the cloud-only and hybrid

experiments. In addition, the energy assessment was based on continuous one-second interval logging over repeated monitoring sessions rather than a few isolated readings, which helped smooth out minor temporary fluctuations within the larger dataset [48,58]. Therefore, the energy trends observed between the two processing models remain sufficiently consistent and dependable for comparative evaluation.

Power readings were recorded at 1 second intervals (1 Hz sampling rate). localized sampling rate was deliberately established as a demanding operational baseline to rigorously stress-test the architectural boundaries of both frameworks. In embedded IoT configurations, the sampling interval is the primary determinant of radio transceiver duty cycles and active power draws [58a]. While environmental metrics often fluctuate slowly, a 1Hz frequency resolution ensures the capture of transient anomalies and microclimatic spikes prior to any data reduction, while forcing the Cloud-Only mode into a continuous transmission state to expose its maximum potential energy penalty [23, 58a].

This provided a detailed, high-resolution record of the ESP32's power consumption. This sampling rate was chosen to correlate with the device's dynamic behavior during the three phases of operation (sensing, processing, and transmission)
to document fluctuating and short-term energy spikes.

The instantaneous power P at each timestamp was computed as:

$$\mathbf{P = V \times I}$$

Where:

- V = measured voltage (volts)
- I = measured current (amperes)

To calculate the total energy consumption over the experiment, the system applied the following discrete integration method:

$$\boldsymbol{Energy(kW)} = \sum \frac{\boldsymbol{P_t \times \Delta t}}{\boldsymbol{3600 \times 1000}}$$

Where:

- P_t = power at second t
- Δt = 1 second (sampling interval)
- The divisor 3600 ×1000 converts watts-seconds to kilowatt-hours

This calculation method provided precise tracking of the energy consumed by the ESP32 in both modes i.e. cloud-only (where it merely senses and transmits) and hybrid (where it also performs edge-level filtering and aggregation). It allowed the study to distinguish between energy spent on wireless transmission versus local computation, two of the most critical contributors to energy cost in IoT deployments [54]. These calculations were supported by multiple studies, presenting sample-based power logging and gives examples converting watt-hours to kWh i.e. practical description of discrete integration of power over time and procedural steps showing the sum of instantaneous power samples to compute energy and real power [54-58].

At the cloud level, energy consumption was estimated by monitoring CPU usage and memory load using Node.js instrumentation and the pidusage npm module [59]. Based on the observed server load, energy usage was approximated using standard cloud energy estimation models, which associate percentage CPU load and active processing time with known power draw values for virtual machines with the specs of 1 vCPU and 2GB RAM. This server-side measurement captured the energy costs of receiving, processing, and storing incoming sensor data, either in its raw form (cloud-only) or after preprocessing (hybrid).

To assess the environmental impact of each architecture, the study employed a carbon intensity factor of 400 grams of CO_2 equivalent per kilowatt-hour (g CO_2e/kWh), in alignment with the UK national grid averages [60,61]. This figure is a carbon intensity factor, meaning that for every 1 kilowatt-hour (kWh) of electricity consumed, 400 grams of CO_2 equivalent are emitted and endorsed by the SBTi and the CDP as an appropriate estimation of electricity-related emissions within the region [60,61]. There also exist example of study that explores the carbon implications of AI inference workloads across different hardware platforms. It employs a carbon intensity factor of 400 g CO_2e/kWh to evaluate the environmental impact of various computing infrastructures [62].

Therefore, in this study the emissions for each system component were calculated using:

Carbon Emissions (g CO_2e) = Energy Consumption (kWh) × 400

This transparent, measurement-driven approach ensured that the sustainability assessment was grounded in actual energy behavior rather than theoretical assumptions. Carbon emissions were computed separately for the ESP32 and the cloud server in both experimental modes, enabling a full comparison of environmental impact.

Furthermore, the study also explored carbon cost per kilobyte of data transmitted and carbon cost per sensing operation, offering additional granularity to the sustainability evaluation. These metrics were especially relevant in comparing the efficiency of local vs. centralized data processing and their implications for scalable IoT deployments [63].

All energy and emission measurements were logged alongside system performance indicators such as transmission frequency, processing latency, and data throughput that is over the full observation period. These detailed logs provided the analytical foundation for the comparative evaluation discussed in the next sections on data collection and results analysis.

3.6.1. Formalization of Experimental Controls and Constants

To establish a scientifically rigorous and empirically fair baseline for comparing the cloud-only and hybrid edge-cloud processing architectures, a strict controlled-variable framework was implemented. By holding all external environmental, computational, and networking variables constant, any observed variance in energy consumption, latency, or server utilization can be definitively attributed to the manipulated variable: the data processing allocation strategy.

The experimental constraints and controlled variables are formalised as follows:

- **Hardware and Sensing Baseline:** Both experimental setups utilized identical ESP32 microcontroller boards class-matched from the same manufacturing batch to eliminate micro-architectural power variants. The sensor suite (tracking temperature, humidity, atmospheric pressure, solar radiation, and rainfall) and the current monitoring configuration (INA219 power sensors connected to stable, identical USB power sources) were identical across both deployment infrastructures.

- **Sampling Interval Constraint:** The data acquisition frequency was locked strictly at a localized one-second sampling interval (1Hz) for both architectures, ensuring a uniform data influx rate before any transmission mechanics took place.

- **Network Environment and RSSI Assumptions:** Both nodes were positioned equidistantly within the exact same physical Wi-Fi hotspot environment. While dynamic network layer fluctuations prevent an identical localized dBm logging without introducing computational overhead to the microcontrollers, radio frequency variables were controlled by maintaining

a consistent placement, a shared gateway, and matching network traffic profiles to isolate and minimize Wi-Fi Received Signal Strength Indicator (RSSI) variability.

- **Identified Backend Server Environments:** On the cloud layer hosted via Render, two separate but completely identical server instances were instantiated. Both backends shared the identical technical configuration specifications: running matching Node.js runtime environments, utilizing identical Express.js backend application logic, and communicating with distinct but structurally identical MongoDB database clusters.

- **Communication Protocol Constraint (HTTP vs. MQTT):** Hypertext Transfer Protocol (HTTP) was deliberately selected and maintained as the exclusive transmission protocol across both modes. While Message Queuing Telemetry Transport (MQTT) is widely acknowledged as a lightweight alternative for IoT architectures, HTTP was held constant to prevent protocol-specific optimization from skewing the results. This strict constraint ensures that the research directly isolates and evaluates data processing topologies (raw centralized streaming versus localized edge aggregation/filtering) rather than conflating the performance of different application-layer transport protocols.

3.7. Data Collection Procedure

The data collection for this study was done over a consecutive period of seven days to confirm that the results indicated sustained and representative system performance. Each day, the experiment was conducted for approximately ten hours, starting in the morning and concluding in the evening. This approach enabled the capture of a complete daily environmental cycle while minimizing variations due to natural factors like temperature, humidity, and sunlight. Imposing a fixed operating schedule also minimized irregularities in Wi-Fi performance, server response times, and background noise, which, if left unchecked, could compromise the readings' reliability.

In this system, the sensors were designed to capture data every second in order to simulate a real-time IoT system whereby data is streamed continuously. The system also recorded the temperature, humidity, pressure, UV light intensity, and rainfall. Every reading was logged with a timestamp and unique device ID to facilitate data point traceability, error checking, record alignment for analysis, and streamlined extraction of coordinated readings during analysis.

The experiment utilized two ESP32-based IoT nodes. One device was set up to process all sensor data in the cloud and the other utilized a hybrid edge-cloud approach, processing some data on the device before sending it to the server for additional processing. To ensure a fair comparison, two separate but identical backend servers were used. Each ESP32 was connected to its own server instance, and for both setups, all networking and environmental conditions were the same.

In the cloud-only setup, the ESP32 transmitted raw sensor data straight to a Node.js server connected to MongoDB and a live web dashboard. The hybrid setup, however, performed simple preprocessing steps on the device itself. These included filtering noisy readings, averaging values, and removing unnecessary data before transmission. Both systems shared the same backend software stack, which included WebSocket streaming, MongoDB databases, EJS dashboards, and tools for monitoring system performance.

To ensure fair comparison, several parameters were kept the same for both systems. The sampling rate was fixed at one reading per second. The Wi-Fi signal strength, network access point, data format, and JSON structure were also standardized. Both backend environments used the same version of software and configuration settings. These controls were important to make sure that any observed difference in results came from the type of processing method and not from network, hardware, or coding differences.

Power usage for each ESP32 was tracked continuously by an INA219 current sensor. The sensor collected data for voltage and current every second which were logged in CSV files, along with timestamps. Stable USB power sources were used to avoid variations in voltage that might distort readings. On the server side, resource monitoring tools such as pidusage and OSutils were used to record CPU usage, memory consumption, data transmission rate, and uptime. These values helped to assess how much computing power and energy each setup required.

All files, including environmental data, power logs, and server metrics, were stored in separate folders for each day of the experiment. Data checks were carried out daily to confirm that files were complete and free of missing values or transfer errors. Any issues found were corrected before the next day's run.

This structured and consistent process made it possible to gather a complete dataset for both systems. The recorded values were later used to compare total energy use, bandwidth, and

system load, as well as to estimate carbon emissions. These results form the foundation for the analysis and evaluation presented in the following section.

3.8. Performance Evaluation Metrics

Understanding the effectiveness of the discussed models in data processing requires a custom set of performance evaluation metrics. These metrics were chosen because of their relevance to system efficiency, system environmental impacts, and the practical aspects of the "real world" IoT deployments, i.e., low power usage, performance responsiveness, and low infrastructure loads in system balances and scalability.

One of the primary metrics was total energy consumption in kWh. The edge device's power consumption was tracked in real-time using an INA219 power sensor that monitored the voltage and current values every second. This allowed for fine-grained tracking of the ESP32's power behavior under different workloads i.e. simple sensing in the cloud-only mode versus additional computation in the hybrid setup. Energy consumption data was then integrated over time to yield cumulative power usage for each scenario. On the cloud server, energy estimates were derived from backend performance monitoring tools, which reported consistent average power draw across both configurations. By calculating power multiplied by runtime, a precise figure for server-side energy consumption was obtained.

Calculating carbon emissions in grams of carbon dioxide equivalent (g CO_2e) closely followed energy consumption estimation. These emissions were evaluated through the standard carbon intensity factor of 400 g CO_2e/kWh, which is the average emissions of electricity generated in the UK. Such an approach made it possible for every unit of energy use to be translated to an environmental impact, which in turn allowed for a sustainability assessment to be comparable to the globally accepted footprints of the CDP and the SBTi [60,61].

System latency was another key metric, used to assess the real-time responsiveness of each architecture. Latency is described as the delay between the time a sensor reading was captured by the ESP32 and the cloud dashboard visualization. To track this metric, timestamp logs were stored at both ends i.e. sensor and server, allowing for precise measurement of end-to-end data propagation time. Latency was expected to be higher in the cloud-only model due to larger data payloads and higher network dependency.

In addition, resource utilization on the cloud server was continuously monitored to assess computational load and memory usage. Tools such as pidusage, os-utils, and custom Node.js scripts were employed to record key performance metrics, including CPU load (in %) to check the processing load for incoming data streams, memory load (in Mb) for tracking dynamic memory, as well as uptime, and load average to check for processing spikes and/or bottleneck when input load changes and performs streams. These metrics were particularly useful in determining to what extent the hybrid mode of edge computation described in the model alleviated pressure on the server. These metrics also analyses how edge processing alleviates pressure from central cloud resources and how it impacts the scale, elasticity, and cost of the IoT infrastructure.

Furthermore, data transmission volume was recorded to evaluate network bandwidth efficiency. The number of bytes transmitted per second was tracked, enabling the study to calculate total data usage over each session. This helped determine how effectively local preprocessing in the hybrid model reduced communication overhead, which is a vital consideration for battery-powered or bandwidth constrained deployments.

Together, these metrics provided a detailed performance profile, which highlighted technical efficiency, real-time capabilities, network economy, and ecological sustainability. The basis for the comparative assessment in the next chapter was built on this multi-faceted approach. This chapter quantifies the practical trade-offs between cloud-only and hybrid edge-cloud processing for IoT systems in edge-cloud processing.

3.9. Validation and Reliable Approach

To achieve and maintain reliability and validity, the study followed a structured experimental validation procedure. Figure 4 illustrates the workflow used in the validation process, including sensor data acquisition, processing through both the cloud-only and hybrid edge–cloud models, performance measurement, and comparative analysis. This procedure ensures that both processing architectures are evaluated under identical experimental conditions. All values were recorded using calibrated and verified monitoring instruments. For the ESP32 microcontroller, power data from the INA219 sensor were verified against theoretical estimates to assess measurement reliability during each sampling interval. Server-side performance data were validated using independent backend scripts and cross-checked against system timestamps to ensure synchronization.

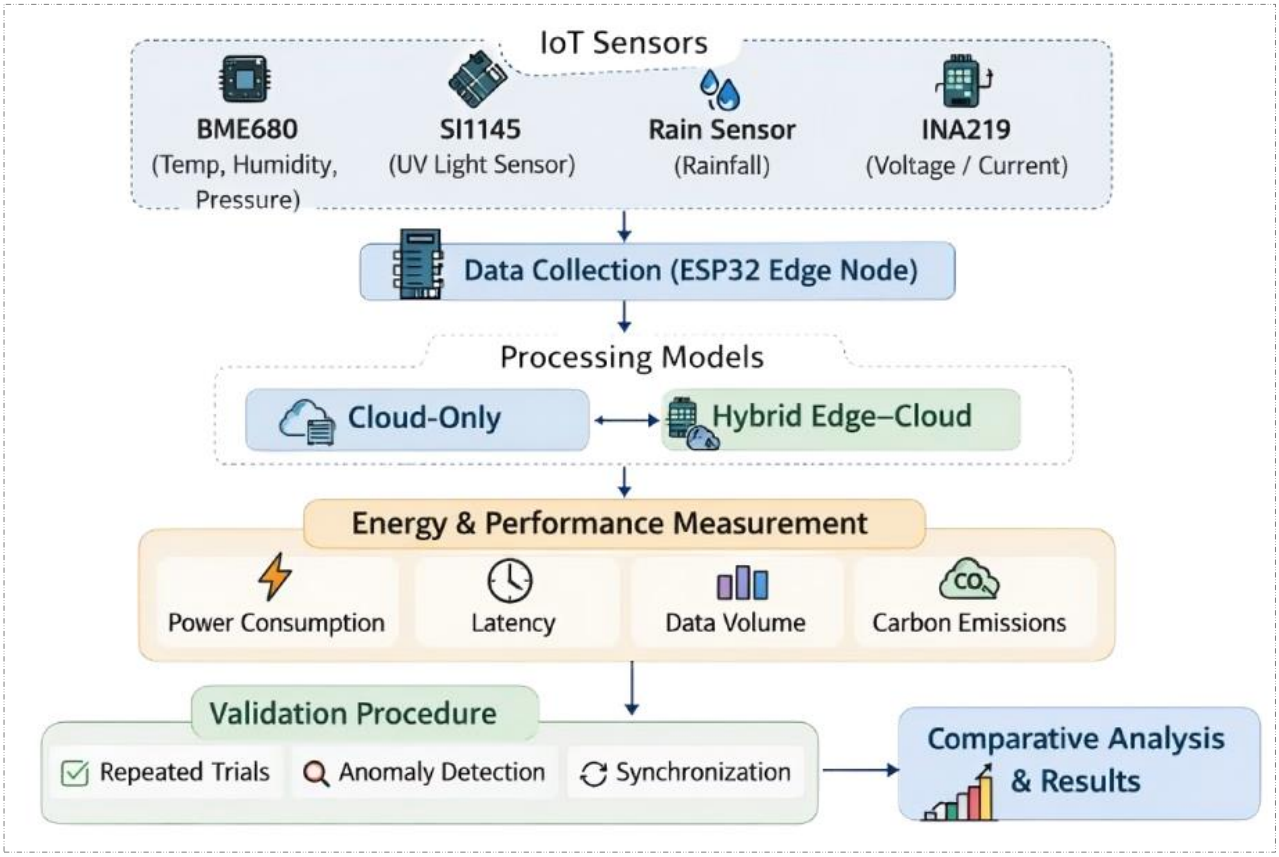

Figure 4: Experimental Validation Workflow for IoT Edge–Cloud Processing Models.

As shown in **Figure 4**, environmental data from the IoT sensors (BME680, SI1145, rain sensor, and INA219) are first collected by the ESP32 edge node. The data are then processed using two architectures: a cloud-only model and a hybrid edge–cloud model. For each model, key performance metrics including energy consumption, latency, data transmission volume, and carbon emissions are measured. The collected data are then validated through repeated experimental trials, anomaly detection, and timestamp synchronization before being used for comparative performance analysis.

For consistency, the same procedures and workflows were followed for each experiment over the course of seven consecutive days. Any anomalies in the data, such as missing values or unusual spikes, were reviewed and either corrected or excluded following standard data cleaning practices. This repeatability helped validate the hypothesis that hybrid processing offers meaningful reductions in energy consumption and emissions without compromising performance. The real-world data rather than simulated results provided added value in justification of the study and its applicability to real IoT practices. Through this combination of methodological rigor and real-time testing, the research provides a trustworthy basis for the performance comparison discussed in the following chapters.

3.10. Ethical and Environmental Considerations

The design and implementation of this research project were done with careful attention to both ethical standards and environmental responsibility. Given that the study did not involve human participants or sensitive personal data, ethical considerations were addressed to ensure transparency, accountability, and integrity throughout the research process.

All elements of the system hardware and software and even the cloud resources were and are still used for academic purposes within the controlled environment. No third-party user data was collected, shared, or processed at any stage. The open-source software frameworks used, including Node.js and MongoDB, aligned with ethical coding since the system was, and still is, free of closed-source and proprietary coding costs. The data that was collected from the environmental sensors were simply observational data, focusing on basic weather parameters of temperature, humidity, and UV light intensity. These parameters were solely for assessment of performance and energy analysis, and no data that is personally identifiable was involved. There was and is still compliant adherence to the guidelines provided on ethical research and all the instruments of surveillance were muted within the permissible legal and technical limits.

On the environmental front, the research directly engages sustainability by focusing on energy efficiency and carbon emission reduction. The comparative analysis between cloudonly and hybrid edge-cloud models was intended to inform greener design choices in future IoT systems. Where possible, energy consumption during the testing phase was kept to a minimum. The experiments were scheduled in efficient time blocks, and all devices were powered down outside of data collection hours to avoid unnecessary electricity use. The emission of carbon was calculated according to the estimations provided in the literature and the guidelines developed by SBTi and the CDP. This guaranteed action means every environmental impact assessment was founded on proven and valid strategies. With justice and environmental consciousness in the research method, the study became an inspirational quest alongside the accountable development of technology. This ethic strengthens the relevance of the study on sustainability in the areas of IoT and Data Processing.

In summary, this chapter has outlined the full methodology that has been adopted for assessing energy-efficient processing in IoT-cloud systems. As part of the hands-on experimental framework, the cloud-only and hybrid edge-cloud models were put into practice, and all were evaluated in a real-world scenario. The rationale for the choice of hardware and

software setups, data processing workflows, power and emission calculations, and the various performance metrics was to ensure the study remained anchored dispelling ambiguity and enabling reproducibility. Together, these methodological elements form a strong foundation for the upcoming analysis, where the results of both models was compared to assess their impact on energy consumption, system performance, and environmental sustainability. The next chapter was present the system design and implementation in more detail, offering a closer picture of how each component operates within the complete architecture.

4. System Design and Implementation

This chapter covers the complete design and implementation of the system developed for this research. It covers both the hardware and software aspects of the IoT-based weather monitoring setup, including the edge computing design, cloud server infrastructure, user interface features, data logging mechanisms, and the comparative framework used for performance evaluation.

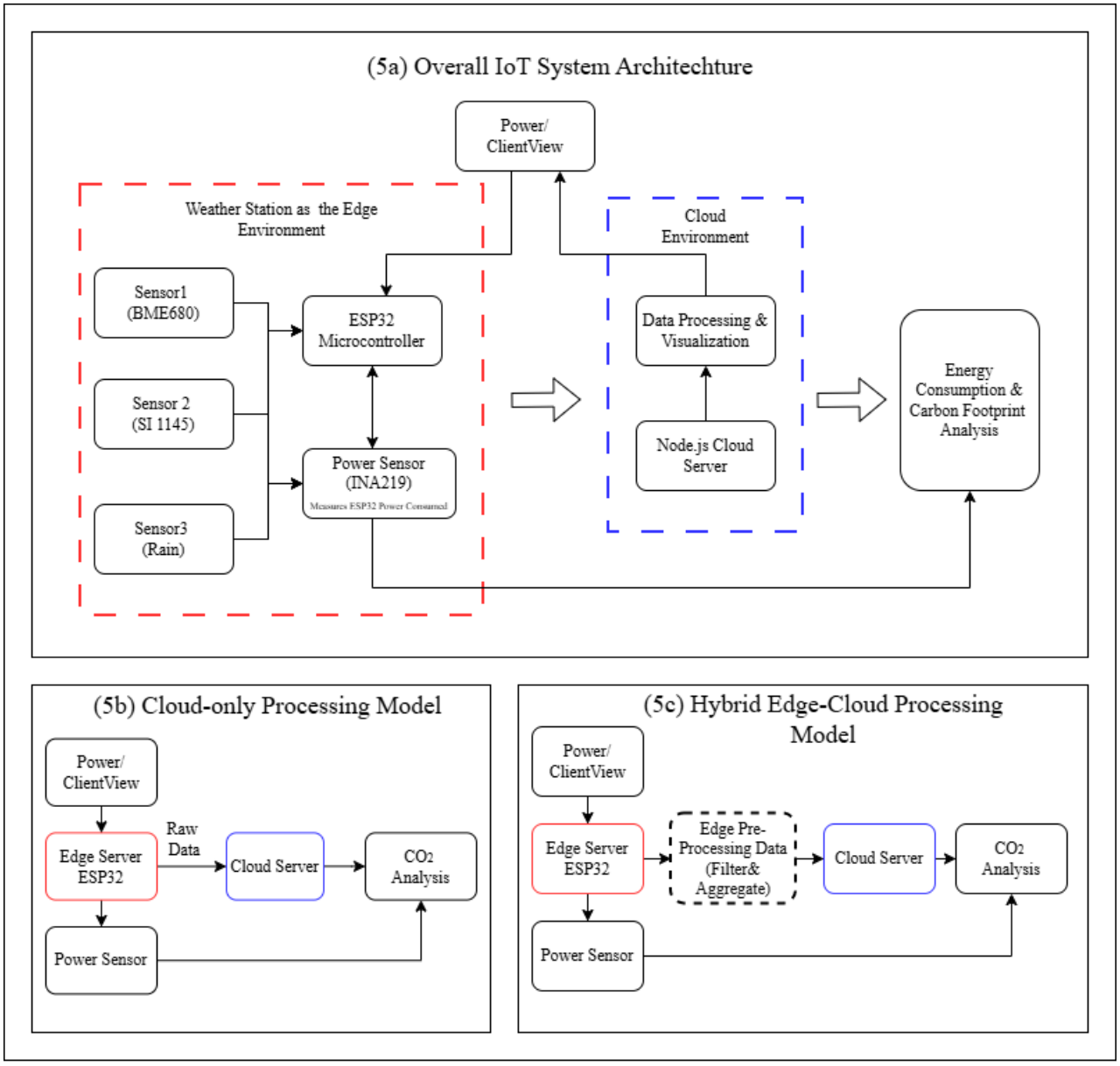

Figure 5. System architecture and processing models used in this study: (a) overall IoT weather monitoring architecture, (b) cloud-only processing model, and (c) hybrid edge–cloud processing model.

Figure 5 illustrates the architecture and processing models implemented in this study. The figure is divided into three subfigures. Figure 5(a) presents the overall architecture of the IoT weather monitoring system. The ESP32 microcontroller acts as the edge device and is connected to environmental sensors including the BME680 sensor for temperature, humidity, and atmospheric pressure, the SI1145 sensor for UV light intensity, and a rain sensor for precipitation detection. An INA219 power monitoring sensor is also integrated to measure the real-time voltage and current drawn by the ESP32 microcontroller, ensuring that the reported energy measurements include the total power consumption of the edge device.

Figure 5(b) illustrates the cloud-only processing model. In this configuration, raw sensor data collected by the ESP32 are transmitted directly to the cloud server without local processing. The Node.js cloud server performs data processing, storage, visualization, and carbon footprint analysis.

Figure 5(c) shows the hybrid edge–cloud processing model. In this approach, the ESP32 performs lightweight preprocessing tasks such as filtering and aggregation before transmitting summarized data to the cloud server. This reduces the volume of transmitted data, decreases network load, and improves overall energy efficiency while maintaining system responsiveness.

Together, these subfigures illustrate the system architecture and enable a direct comparison between centralized cloud processing and hybrid edge–cloud processing in terms of energy consumption, latency, and carbon emissions. Each part of the system was built to support the overall goal of reducing energy consumption and carbon emissions in real-time data processing. The following sections describe the components in detail, beginning with the weather station at the edge layer.

4.1. IoT Weather Station

This study's experimental framework is built upon a custom-designed IoT weather monitoring station that serves as the primary data-generating node. Designed with energy efficiency and real-time environmental sensing in mind, the station is constructed using an ESP32 microcontroller and a range of purpose-specific sensors. The selection of the ESP32 was due to its dual-core processor, built-in Wi-Fi and Bluetooth, and low power dissipating characteristics, which are valuable for edge computing.

The sensor suite includes the BME680 for temperature, humidity, atmospheric pressure; the SI1145 for UV light intensity; and a rain sensor for precipitation monitoring. Each of these

sensors was selected based on three criteria: (1) data relevance for climate and environmental studies, (2) ESP32 platform integration, and (3) sustainable operation in terms of energy consumption. The entire system's power consumption was monitored in real-time using the INA219 current sensor, which provides the necessary voltage and current data to calculate the overall energy consumption of the system.

Figure 6 illustrates the interconnectivity and functionality of the IoT system designed to evaluate energy efficiency between cloud-only and hybrid edge-cloud processing models. This system consists of various sensors that revolve around the core of the system, the ESP32 Development Board, which uses Wi-Fi, Bluetooth, and its high processing power to perform all data application and communication tasks. Alongside the windows' environmental data, the BME680 are added for temperature and humidity measurements. A Rain sensor provides rainfall data and UV terminology levels are measured by the SI 1145 Sensor. These sensors are sufficient for weather monitoring application in the study. Additionally, an INA219 Sensor is included for controlling system power consumption and relates directly to the study's energy efficiency interest. The use of prototyping materials like breadboards and outdoor weatherproof enclosures allow for enhanced reliability and precision enabled by the deployment. The diagram illustrates how all sensors connect to the ESP32 to maximize data transfer either to the

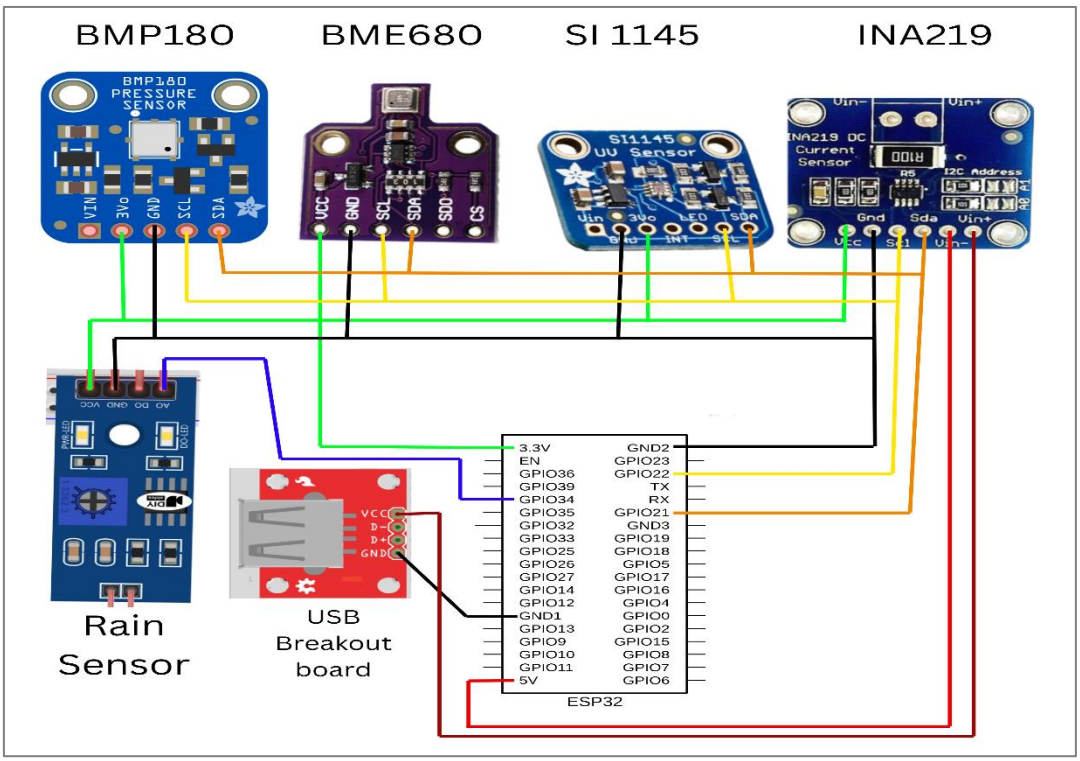

Figure 6. Circuit Diagram of the ESP32-Based IoT Weather Monitoring System Hardware Setup

cloud or for local processing. While this setup shows the operational flow, it also serves to demonstrate the energy efficiency features designed in the hardware configuration. s

This detail of the circuit system establishes the practicality and creativity of the proposed solutions for energy efficiency in real-life applications of IoT technologies. This hardware system acts as a unit-level IoT device capable of both sensing and partial processing, thereby forming a scalable model for hybrid edge-cloud evaluation. The continuous and highfrequency data transmission (at 0.1 Hz sampling rate) replicates real-world smart monitoring systems and provides a robust stream for both edge and cloud processing models.

4.2. Edge Layer Design

In the hybrid IoT architecture developed for this study, the edge computing layer serves as the foundational component. It is implemented on the versatile, low-power, dual-core ESP32 microcontroller, which is Wi-Fi enabled. It was selected because it can support real-time sensing and local processing within the microcontroller's resource constraints. This layer performs lightweight preprocessing on the sensor data before it is sent to the cloud.

At the core of the edge processing logic are three key tasks:

1. **Noise Filtering:** The ESP32 uses simple filtering logic to eliminate sensor data that exceeds preset environmental limits. This means that unnecessary transmission of outliers or faulty readings that might be caused by transient sensor fluctuations is avoided.
2. **Data Aggregation:** Rather than sending raw sensor readings at every interval, the ESP32 computes average values over a defined time window (e.g., 10 seconds), effectively compressing the data while retaining key environmental trends. By this means, the sensor data payloads are significantly minimized, resulting in lower bandwidth consumption, and the Wi-Fi radio, known to be a power hungry component, is less frequently activated as it is one of the most power consuming operations on microcontrollers.
3. **Timestamping and Formatting:** Each processed reading is timestamped, tagged with the device ID, and stored in a standardized JSON format for uniformity. This facilitates efficient storage, visualization in the cloud, and subsequent analysis to maintain traceability and detection.

In order to monitor the energy impact of the activities on the edge layer, we use an INA219 power monitoring sensor alongside the ESP32. This module tracks in real-time the voltage and the current consumption of the microcontroller and the peripherals attached to it. In order to monitor

the energy impact of the activities on the edge layer, we use an INA219 power monitoring sensor alongside the ESP32. This module tracks in real-time the voltage and the current consumption of the microcontroller and the peripherals attached to it. These readings are used to compute instantaneous power (in watts) and cumulative energy usage (in kilowatt-hours). Power measurements are expressed in watts (W) and subsequently converted to kilowatts (kW) and kilowatt-hours (kWh) to align with standard energy measurement practices used in energy systems and carbon emission analysis as reported by the International Energy Agency [12]. The use of kWh enables meaningful comparison with energy consumption benchmarks and facilitates accurate estimation of carbon emissions associated with data processing [9]. The total energy consumption of the edge device is calculated from the recorded measurements and used as a key metric in the analysis section for comparative evaluation against the cloud-only processing baseline.

The current physical setup also ensures consistent power delivery to the ESP32 using a USB regulated breakout board. This further secures the power supply against voltage fluctuations which can impact the sensors and communication to measure the energies accurately.

The design of this edge layer is vital in achieving the research objectives of this study. It alleviates the workload processing in the cloud, which tackles energy efficiency, data transmission, and latency concerns. It shows the ability of inexpensive microcontrollers to facilitate real-time, decentralized, and automatic decision-making without the complexities of edge-AI and high-performance computing.

In conclusion, this layer represents the first tier in the proposed hybrid edge-cloud architecture. It transforms the ESP32 from a passive sensor node into an intelligent edge processor capable of contributing meaningfully to energy savings. This shift in architecture confirms the research assumption that it is possible to implement adaptive and efficient data manipulation in Internet of Things frameworks while also insulating energy efficient data processing.

4.3. Cloud Layer Implementation

In this study, the cloud layer is the foundation of centralized data processing. It was developed to complement the edge-based weather monitoring system by handling data ingestion, storage, performance monitoring, and visualization in real time. This component is critical in facilitating comparative evaluations across two system designs: cloud-only and hybrid edge-cloud processing systems. To keep all variables constant except for the data

processing point, two identical servers were deployed on the Render platform. Each server functions under the same conditions with the same applications, and presents an identical client interface, which enables accurate measurements of the energy and carbon footprints attributable to the processing workflow.

In the following subsections, the cloud layer components, structure, and operational functions was detailed. The explanation starts from the chosen technology to real-time performance monitoring and aims that the infrastructure is in alignment with the research goals of reproducibility, sustainability, and accuracy.

4.3.1. Technology Stack

The cloud servers were constructed on a modular and streamlined web stack that provided a high degree of performance visibility and tracking. For the base layer, server routing, data stream handling, and API endpoint interface definition are operationalized via the Express.js framework and Node.js. For remote and on-demand access to incoming sensor data and system logs, the system employs cloud-based NoSQL storage via MongoDB Atlas.

The system employs WebSocket communication via the Socket.IO library, to provide seamless two-way data exchange between the server and client dashboards. As well, the pidusage module helps to track system resource key metrics, which include system uptime, memory and CPU load. These metrics are used in the system energy profiling calculations that estimate the carbon emissions of both processing models.

Security and configuration management are handled by middleware and environment configuration tools like dotenv and Helmet.js, and dynamic dashboard content is rendered with EJS. As a whole, this stack provides flexibility appropriate for academic experimentation without losing clarity, observability, or operational integrity.

4.3.2. Architecture and Key Functionalities

The cloud layer in this study follows a centralized cloud architecture, where data processing, storage, and visualization are handled by a single server instance deployed on the Render cloud platform. Although two server instances were deployed, they operate independently under identical configurations to support experimental comparison between the cloud-only and hybrid edge–cloud processing models. This architecture ensures controlled

evaluation conditions while maintaining a structure similar to typical centralized IoT cloud deployments.

The architecture of the cloud layer was designed to replicate real-world IoT-cloud systems while still having the controlled environment needed for empirical investigation. One server was dedicated to cloud-only data processing while the other one was assigned to the hybrid edge-cloud processing model. Both environments were built with the same applications, interfaces, and components on the client side. They had the same logic for data ingestion, visualization, and performance monitoring so that the only point of difference between the two was the processing workflow. This separation made it possible to isolate and assess the influence of architecture on energy consumption and carbon footprint under the same operational conditions.

Each server receives live environmental using secure HTTP endpoints from an ESP32based weather monitoring device. This data is then validated and stored to the database before being sent to all clients using WebSockets. This real-time interaction offers continuous updated visibility to users. This is similar to the behavior of production-grade cloud applications.

Client dashboards reflect environmental conditions including temperature, humidity, pressure, rainfall, and UV light intensity. The same interface also visualizes live system metrics such as CPU usage and memory load, providing insight into server behavior and processing workload. Additionally, a dedicated section on the dashboard allows users to filter and analyse historical data across selected timeframes. This supports retrospective exploration of trends and enables validation of cloud-based computational effort over extended durations.

4.3.3. Deployment Strategy and Justification

The cloud processing infrastructure for this study was initially developed as a locally hosted server using Node.js and the Express.js framework. This setup was selected to ensure full control over system behaviour, allowing direct access to key performance metrics such as CPU usage, memory consumption, and uptime. These parameters were critical for accurately evaluating energy consumption in both cloud-only and hybrid edge-cloud processing models. The local environment provided consistent experimental conditions with complete transparency in how data was handled and processed, enabling precise system monitoring and energy profiling from the outset of the research.

As the system evolved and reached a stable operational state, the cloud layer was transitioned from the local server to an online hosting environment using Render. This shift was introduced to more closely mirror a real-world cloud deployment. To preserve the validity of the comparison between the two processing models, two separate Render-hosted servers were configured. Performance monitoring through pidusage was carried over without interruption, allowing to continue tracking energy and system behavior in the hosted environment as was during the local phase. The transition thus supported the shift to realistic deployment while preserving full observability and experimental control.

4.3.4. Script Functionality Overview

For Render-hosted cloud server, the modular components are structured around the TypeScript programming language. Each server component is maintained with the structured clarity principle in mind. Each primary server module instance is responsible for initializing the application, configuring API routing, and setting up WebSocket communication. A dedicated metrics module continuously monitors system resource usage in real time, feeding this data into the energy analysis pipeline.

TypeScript interfaces describe the incoming sensor data, and only serially well-formed data entries are processed and logged. Security policy enforcement, as per middleware, is triggered by API key validation before stream submission. Finally, EJS-based view templates dynamically render the client dashboard, keeping it synchronized with the latest data and system metrics.

Module	**Purpose**
server.ts	Initializes the Express app, handles API routes, and sets up Socket.IO.
metrics.ts	Uses pidusage to collect system resource metrics at runtime.
types.ts	Defines TypeScript interfaces for structured sensor data.
middleware.ts	Implements API key validation for secure data submission.
views/	Contains EJS templates for rendering live dashboards.

Table 1 presents a summary of the core server modules and their respective functionalities within the platform.

The use of modular components enhances system design and provides varied options for updates to support new research trials and new applications. It also ensures that the cloud layer can manage high-frequency sensor data while monitoring performance in real-time poses no difficulty, to assist in debugging, testing, and system monitoring.

4.3.5. Relevance to Research Goals

As hosted on Render, the cloud layer continues to serve as a foundational element in the comparative analysis of cloud-only versus hybrid edge-cloud processing. Within the context of the study objective focused on quantifying the energy and carbon emissions of IoT data processing workflows, the system's capabilities of energy monitoring and real-time streaming along with historical logging and introspection serve this purpose remarkably.

The deployment architecture, despite being cloud-based, offers a research environment that provides controlled and scalable resources to examine the trade-offs and benefits of various processing models. This allows the research to advance from theoretical constructs and into the practical, application-level investigation of sustainable computing in the IoT space.

To conclude, the cloud layer is a powerful, flexible, and safe construction that manages consolidated data. Every element in the system focuses on and facilitates the reliable and seamless environmental data processing, from ingesting data in real time to rendering dynamic dashboards and managing system performance. As the sensor data passes through the cloud system, it is processed and displayed in real time and made available to users via the system’s client interface. The following section focuses on the design and functionalities of this client view and demonstrates how users access real-time data from the sensors, historical data, trends, and system metrics for actionable data analysis.

4.4. Client View Features

A user-centric web-based dashboard was developed to support system architecture to provide real-time interaction, monitoring, and analysis of environmental data captured from the ESP32-based IoT weather station. The dashboard is designed with both clarity and responsiveness in mind so that users can live track system behavior and performance, and can also evaluate historical data. Data is dynamically rendered on the server-side using EJS (Embedded JavaScript) and client-side Responsive design is built using HTML5 and Bootstrap.

This design is integrated with Node.js giving users real-time data visualization without page refresh using WebSocket communication.

The main dashboard page provides live readings from all deployed sensors, including temperature, humidity, and atmospheric pressure (BME680), and UV terminology (UVC sensor) as well as rainfall (Rain sensor). Values are updated on a second-by-second basis, proving the system's capability to handle high-frequency data streams in real-world IoT scenarios. Power consumption from the INA219 sensor is monitored and displayed live, giving users real-time feedback on energy usage at the device level. Below each sensor, time-series plots present recent trends, making anomalies or environmental shifts easily detectable through visual cues.

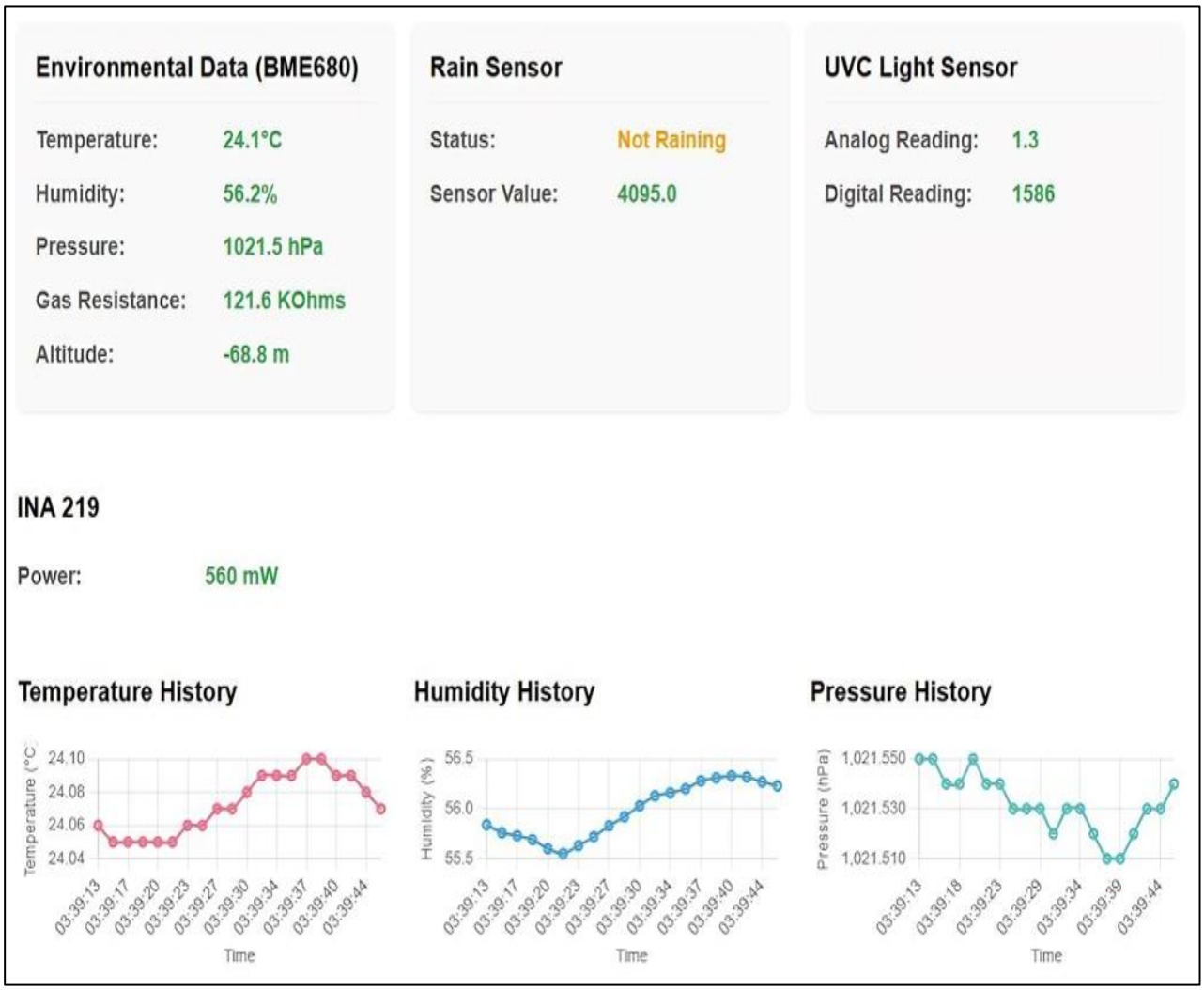

Figure 7. Sensor-level data view from the weather station in live mode

Figure 7 shows the dashboard section that presents individual readings from the BME680, rain sensor, and UVC sensor, as well as real-time power usage from the INA219

sensor. Historical trends are visualized through live time-series graphs for temperature, humidity, and pressure.

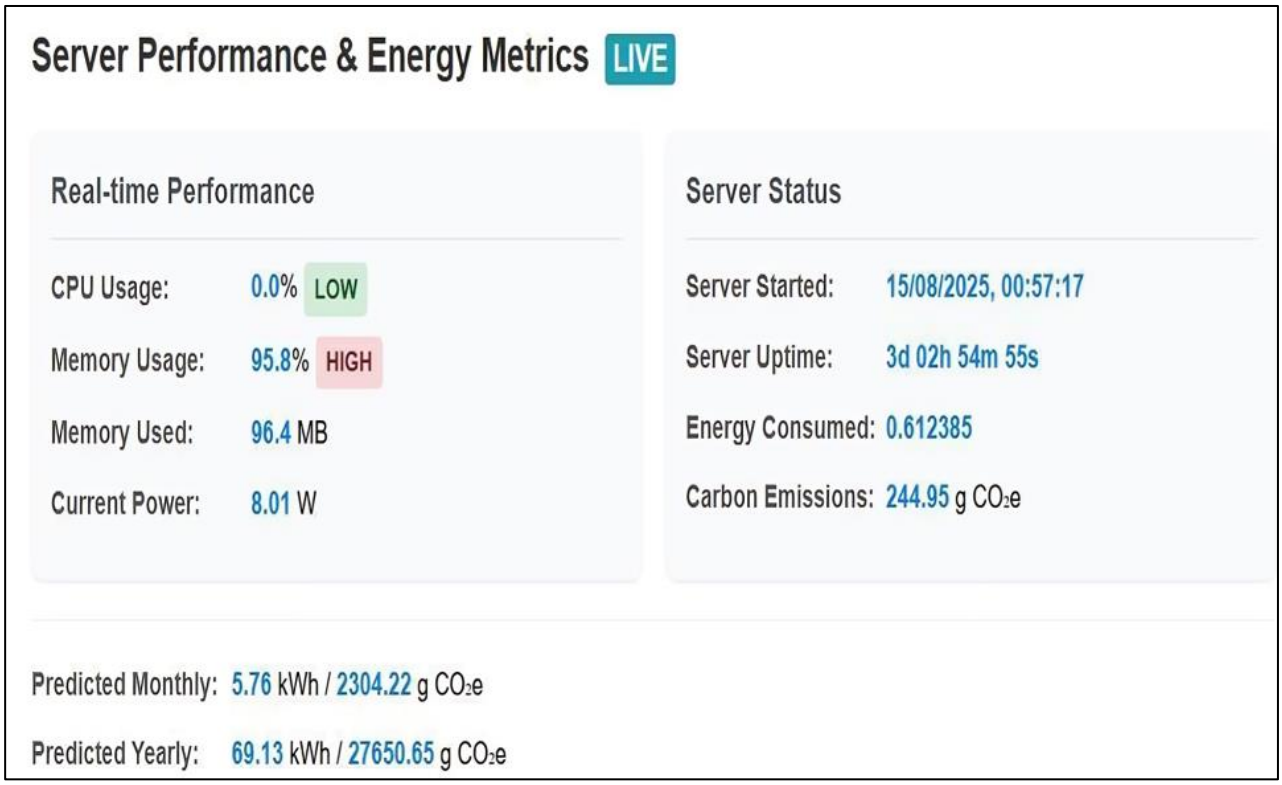

Figure 8. Server performance and energy statistics in live view.

Figure 8 shows in parallel, the dashboard also features a real-time monitoring panel for server performance. It displays current values for CPU usage, memory utilization, system uptime, current power consumption, and estimated carbon emissions, all of which are refreshed dynamically. These indicators help users and researchers visualize the impact of incoming data on the cloud infrastructure. The CPU usage appears as 0% because the snapshot was captured during a moment of minimal processing activity when the Node.js server was idle between incoming sensor updates. Using monitoring tools such as pidusage, os-utils, and middleware from Express, data on the server is updated every second. Moreover, the dashboard provides power consumption and potential CO_2 emissions for monthly and yearly basis. This realtime synchronization helps bridge the gap between raw data processing and environmental awareness, making the system more transparent and insightful.

To facilitate retrospective evaluation, a second dashboard view dedicated to historical data analysis was developed. Historical data can be dynamically retrieved from MongoDB when users choose certain dates and time ranges. This was display a combination of environmental and energy trend data and analyses for the specified time. Summary statistics

(e.g., average temperature, humidity, pressure, and rainfall frequency) and corresponding timeseries graphs are presented for each metric. In addition, power consumption data and the visual data on carbon emissions facilitate the comparative analysis on the provided model.

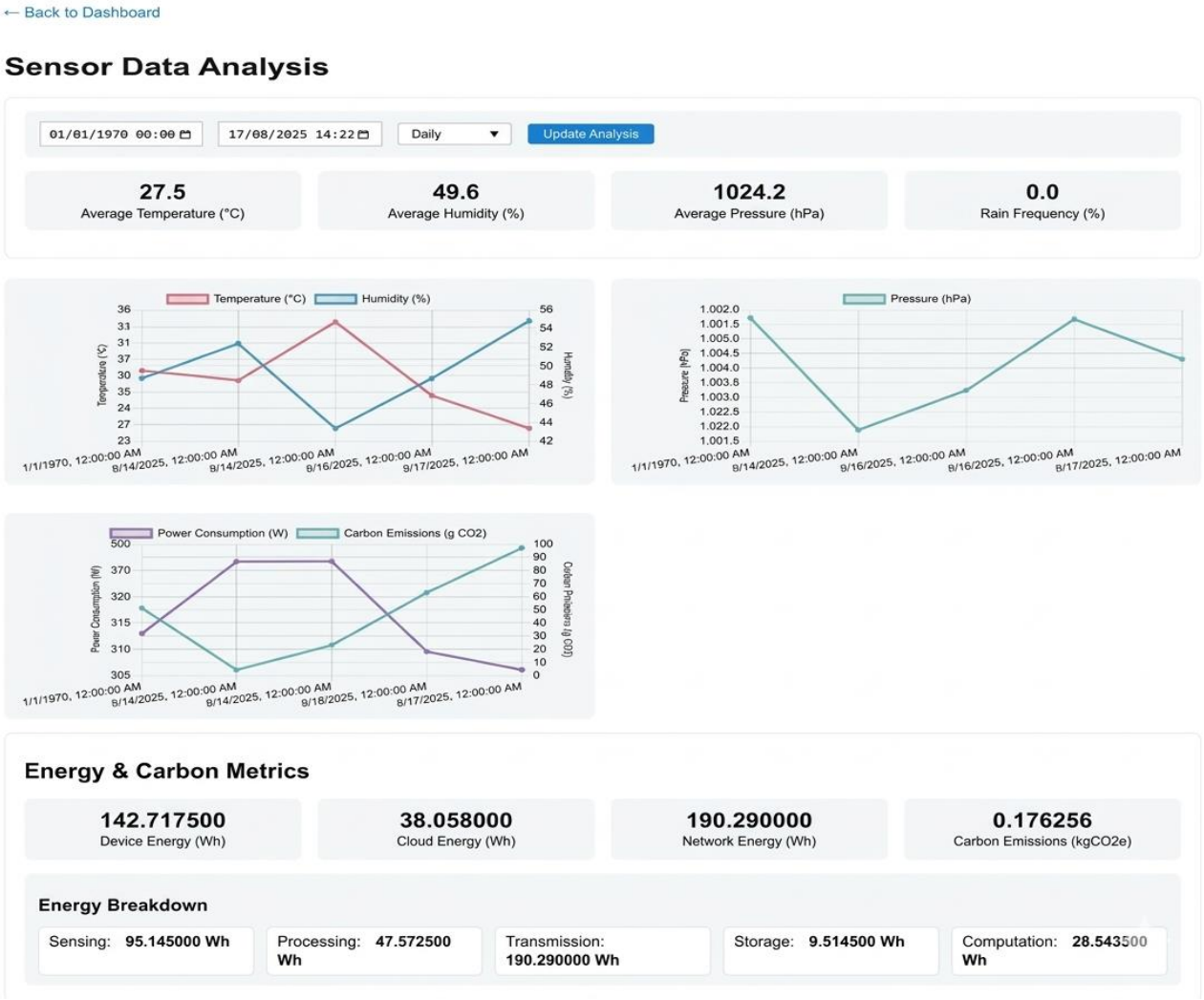

Figure 9. Real-time dashboard displaying analysis view of sensor data, environmental conditions, and energy metrics with time filter.

Figure 9 shows this analysis view, which also includes a comprehensive breakdown of energy consumed by sensing, processing, transmission, storage, and computation, offering full transparency into system efficiency under varying conditions.

To ensure operational reliability, each individual ESP32 device's automatic connectivity monitoring must be held accountable without oversight. A device is flagged “offline” on the dashboard after not transmitting any data for 30 seconds. This mechanism boosts user confidence in the system and resilience in deployments. This is especially important in distributed IoT installations, where manual inspections are not feasible. The monitoring

system's functionality replicates feature set typical of fleet management, and the monitoring functionality enhances commercial IoT applications.

The dashboard is monitoring and analysis tool, closing the loop from edge data capture to cloud processing evaluation. It features dual interfaces for real-time and historical data analytics, and system performance, energy, and reliability. This was important for contextualization of results in the following sections, especially in explaining the processing models and their relation to energy efficiency, system responsiveness, and environmental footprint.

4.5. Data Download and Logging

For post-analysis and traceability, the system has been designed to allow the export of historical data streams in traceable formats, such as CSV and Excel. This system functionality is important for offline analysis; it is also pivotal in the cloud server computing workload simulation. With the system designed to retrieve and process large volumes of historical data on demand, it can simulate real-world data handling conditions, which is important in measuring energy consumption and estimating carbon emissions.

Downloading is made possible through the backend API endpoints built with Express.js, which pulls specific data from the MongoDB database. On the frontend, users can interact with a button that executes the queries and exports files directly from the browser. The exported files contain sensor readings, timestamps, the identification of the ESP32 devices, and the associated power measurements. This data can be used for computational simulation and can also be validated against the readings after the experiment.

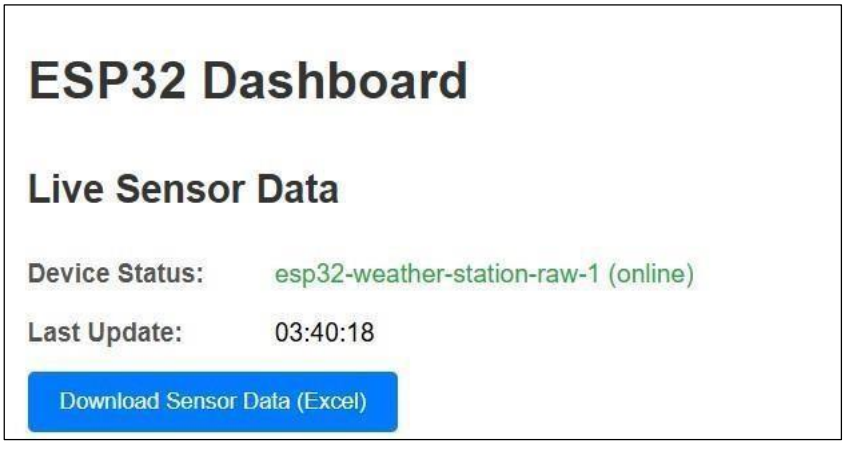

Figure 10. Device connectivity and data logging interface.

Figure 10 shows the client view section with device status, last update timestamp, and a data export button are shown. Users can download sensor readings in Excel format for offline use and performance evaluation.

From the research perspective, this export capability supports simulation of dynamic workloads and stress testing. For instance, it can simulate a single processing session with several days' worth of environmental data, which are pushed the cloud server to its limits. This scenario represents a benchmark of data aggregation and reporting workloads that IoT analytics systems are designed to handle. While performing these operations, active monitoring of power draw, memory, and CPU load provides a clear manifestation of energy behavior with heavy load, aiding differentiation between cloud and hybrid edge–cloud systems.

Furthermore, this feature adds to the study's transparency and scientific credibility. External reviewers and researchers can replicate the study and verify the results using downloadable datasets, which are the same ones used in the experimental analysis. They may re-calculate the study's results, run other models, or perform different analyses to test the findings' robustness. This open-data approach aligns with best practices in green computing, reproducible science, and sustainable IoT system design.

In summary, the data download and logging module are strategically designed to enhance the research methodology, and are intended to provide usability and convenience for the user. This improves the experimental system with academic validation and industry relevance, while enhancing the technical infrastructure needed to support fair, transparent, and scalable energy efficiency evaluations.

4.6. Functional Comparison of Cloud-only vs. Hybrid Models

This section outlines the structural and functional differences between the two data processing models implemented in this study: the cloud-only model and the hybrid edge–cloud model. The comparison highlights how each model was architected, deployed, and expected to behave in terms of energy distribution, network usage, and computational load prior to experimental validation. This comparative design setup was essential for isolating the effect of processing location on system performance in later chapters.

In the cloud-only model, the ESP32 microcontroller was configured to function solely as a data acquisition unit. It transmitted raw sensor readings every second to the cloud server

without applying any preprocessing. The server handled all data processing which included filtering, aggregation, and timestamping. This is a typical centralized IoT architecture which tends to traditional deployments. The main characteristics of this model include:

1. High network bandwidth usage, due to unfiltered, frequent data uploads.
2. The server's computational demand increases significantly as the cloud manages all of the data processing.
3. Minimal logic on the edge, simplifying device firmware but increasing reliance on cloud infrastructure.

In contrast, the hybrid edge–cloud model split the processing tasks. In this case, the ESP32 was responsible for preliminary edge processing, which included noise filtering, data reduction by eliminating redundant readings, and attaching timestamps, after which the trimmed data was sent to the cloud server. The cloud layer received only the essential and already-processed information. The functional attributes of this model include:

1. Preprocessing and selective sending at the edge results in lower frequency and volume of data transmission.
2. Reduced cloud workload since the server only needs to perform storage, analytics, and visualization.
3. More capable firmware on the edge devices, which performs contextual filtering and lightweight computation.

Both models were implemented concurrently on two identical ESP32 hardware units, each connected to its own dedicated server instance. Both setups shared the same technology stack (Node.js, Express.js, MongoDB, and WebSocket) and front-end client views to ensure architectural consistency. Each server instance displayed identical dashboards, visual elements, and real-time sensor data panels, differing only in the way data was processed prior to cloud ingestion.

This side-by-side deployment allowed for a fair and isolated evaluation of how each model functionally handles data flow, processing distribution, and system load. While the analytical results of this comparison are presented in Chapter 5, the design of this experiment supported direct comparison of model behaviors under identical environmental conditions. By incorporating these two distinct processing flows, the system's structural incorporation of the two processing flows demonstrated the real-world relevance and implications of handling

devices in an IoT environment on both the centralized and distributed models. This functional comparison forms the foundation for understanding how architecture-level decisions affect energy distribution and system responsiveness.

4.7. Conclusion

This chapter details the design and implementation of the system developed for this research, including the edges hardware set up to real-time cloud data processing. Every design decision was made with the intention of enhancing energy efficiency and minimizing the ecological footprint of IoT data systems. Moving the infrastructure from a local environment to a cloud-hosted platform allowed for a more realistic simulation of actual deployment conditions while maintaining control over performance tracking. The ability of the system to collect, process, and visualize sensor data in real time, with downloadable logs and simulated workloads, serves a solid basis for assessing operational impact. Given this architecture, the focus of the next chapter shifts to the experimental phase, where energy consumption, carbon emissions, latency, and system performance are analyzed to evaluate the cloud-only and hybrid edge-cloud models in a controlled environment.

5. Experimental Analysis and Results

Based on the previous chapter on system design and its implementation details, this section now turns its attention towards the empirical validation of the proposed energy-efficient hybrid edge–cloud processing architecture. The goal of this experimental step is to test the system performance in practice with a focus on energy consumption, carbon footprint, latency, resource utilization, and impact on the server. By deploying the two parallel configurations, one operating in a conventional cloud-only mode and the other incorporating edge-level preprocessing, we systematically measured and analyzed the environmental and computational resources in a controlled test over a specified time.

In this chapter, we focus on a detailed model of both systems, presenting quantitative comparisons based on monitored sensor data, server performance logs, and INA219 based power measurements. Data from sensors and server logs were systematically collected over seven days under controlled conditions. To support quantitative analysis, a Python-based framework was used to calculate daily carbon emissions, differences between the two models, cumulative savings, and statistical metrics such as averages, maxima, minima, and standard deviations. These python-generated output framed on emission intensity sets the basis for the next chapters on the scalability of the system and the rest of the design.

Detailed explanation of the rest of the chapter is focusing on the experimental design and the monitoring plan of the experiment, followed by the structured energy use analysis, carbon emission analysis, latency, system load, and interpretive compiling. The combined results of these chapters are to provide evidence on the practical and sustainable use of these hybrid systems in real-time IoT deployments.

5.1. Description of Experimental Setup and Monitoring Period

To maintain inter-model correspondence, the experimental analysis took place over a period of 7 uninterrupted days under favorable, or, relatively stable, environmental conditions. Two distinct ESP32-based weather station units were deployed: one for processing clouds by themselves, the other for hybrid edge-cloud processing. The same sensor configurations and sampling intervals were used for both devices. Data were streamed to two separate server instances deployed on the Render Cloud Platform: one for logging incoming data, computing energy consumption metrics, and calculating delta carbon emissions, the other for doing the

same. The servers were identical in configuration with 1 vCPUs and 2GB RAM, providing an equitable standard for assessing the comparison of system load and resource usage.

5.2. Energy Consumption Comparison

This section carries out a detailed comparison of the energy consumed for both the cloud-only model as well as the hybrid edge–cloud model. The analysis is a direct continuation from the method described in Chapter 3, current and voltage over time for each ESP32 device was captured using the INA219 power device and subsequently processed in order to calculate the watt-hour (Wh) consumption over time. After determining the energy consumed for each model, the sensors were identically set up and operated in such a way that the only parameter which could affect the outcome was the method of data processing, which was either entirely cloud-based or a hybrid model with edge local pre-processing fully operated systems.

This portion covers the entire analysis and culminates in core findings which marked the violations that the thesis seeks to prove. The first part focuses on the prediction of energy spent on a daily basis, supported by tables and graphs. The second part calculates the total savings achieved through hybrid processing. Finally, third part embeds these findings in the context of the principles of sustainable design of IoT systems, linking device-level efficiency to larger-scale environmental and operational goals. This portion of the analysis reinforces the thesis argument that localized preprocessing not only reduces network congestion but also mitigates the energy footprint of cloud-based infrastructures.

5.2.1. Daily Consumption Analysis

To analyze daily consumption and operational patterns as well as data coverage of the two systems, sensor logs were analyzed with a Python-based analysis for timestamping. The analysis focused on the number of readings per day, hours of active monitoring, unique hours covered, average sampling intervals, and the percentage of daily coverage. This step ensures that differences in energy consumption are attributable to architectural design rather than inconsistencies in data collection. Following is the pseudocode from python analysis:

FOR each dataset (Raw, Hybrid):

1. Identify timestamp column in dataset
2. Convert timestamp column to datetime format
3. Remove rows with invalid timestamps
4. Extract date (ignore time) from each timestamp

Initialize empty list to store daily statistics

FOR each unique date in dataset:

a. Count total readings for that date

b. Find earliest and latest timestamps

c. Compute time span in hours (latest - earliest)

d. Count number of unique hours covered

e. Compute average interval between readings (minutes), if more than 1 reading

f. Compute daily coverage as percentage of 24 hours covered

g. Store all above statistics in daily stats list

Convert daily stats list into a table/dataframe

Display results for each dataset:

- Daily coverage table (Readings, Hours Span, Unique Hrs, Avg Interval, Coverage %)
- Summary statistics:
  * Average readings per day
  * Average hours covered per day
  * Average daily coverage percentage

IF both datasets exist:

- Merge Raw and Hybrid daily stats on date
- Display side-by-side comparison of key metrics (Readings, Unique Hours, Coverage %)

Provide additional insights:

- Total number of days covered
- Number of days with near-full coverage (≥95%)

Some output snapshots from the above analysis are presented in Figures 11–13, illustrating the daily data coverage and operational behaviour of the two processing models.

Figure 11 presents the daily coverage statistics for the RAW dataset, which corresponds to the cloud-only processing configuration. The results show that the system recorded an

average of approximately 51,749 readings per day, covering about 9.8 hours of operation daily. The average daily coverage was 40.7% of the 24-hour period. Most days exhibited a consistent monitoring window of around 10 hours, while August 23 recorded the highest monitoring duration of 15 hours, resulting in the largest number of readings for that day. In contrast, August 24 shows a significantly shorter monitoring duration of approximately 2 hours, which explains the lower number of readings and reduced coverage percentage. These variations reflect differences in system runtime rather than inconsistencies in sensor behaviour.

RAW Dataset Daily Coverage:

Date	Raw_Readings	Raw_Hours_Span	Raw_Unique_Hours	Raw_Avg_Interval_Min	Raw_Coverage_Percent
2025-08-17	57276	10.00	10	0.01	41.7
2025-08-18	57802	10.00	10	0.01	41.7
2025-08-19	57476	10.00	10	0.01	41.7
2025-08-20	57718	10.00	10	0.01	41.7
2025-08-21	57300	10.00	10	0.01	41.7
2025-08-22	57676	10.00	10	0.01	41.7
2025-08-23	62918	15.00	15	0.01	62.5
2025-08-24	5832	2.08	3	0.02	12.5

RAW Dataset Summary:
Average readings per day: 51749.8
Average hours covered per day: 9.8
Average daily coverage: 40.7%

Figure 11. Daily data coverage of the Cloud only (Raw) system during the seven-day monitoring period

HYBRID Dataset Daily Coverage:

Date	Hybrid_Readings	Hybrid_Hours_Span	Hybrid_Unique_Hours	Hybrid_Avg_Interval_Min	Hybrid_Coverage_Percent
2025-08-17	57512	10.00	10	0.01	41.7
2025-08-18	57636	10.00	10	0.01	41.7
2025-08-19	57852	10.00	10	0.01	41.7
2025-08-20	57232	10.00	10	0.01	41.7
2025-08-21	57642	10.00	10	0.01	41.7
2025-08-22	57668	10.00	10	0.01	41.7
2025-08-23	62756	15.00	15	0.01	62.5
2025-08-24	5832	2.08	3	0.02	12.5

HYBRID Dataset Summary:
Average readings per day: 51766.2
Average hours covered per day: 9.8
Average daily coverage: 40.7%

Figure 12. Daily data coverage of the Edge-Cloud (Hybrid) system during the seven-day monitoring period.

COMBINED DAILY COMPARISON:

Date	Raw_Readings	Raw_Unique_Hours	Raw_Coverage_Percent	Hybrid_Readings	Hybrid_Unique_Hours	Hybrid_Coverage_Percent
2025-08-17	57276	10	41.7	57512	10	41.7
2025-08-18	57802	10	41.7	57636	10	41.7
2025-08-19	57476	10	41.7	57852	10	41.7
2025-08-20	57718	10	41.7	57232	10	41.7
2025-08-21	57300	10	41.7	57642	10	41.7
2025-08-22	57676	10	41.7	57668	10	41.7
2025-08-23	62918	15	62.5	62756	15	62.5
2025-08-24	5832	3	12.5	5832	3	12.5

DATA COVERAGE INSIGHTS:
Raw dataset covers 8 days
Raw dataset has near-full coverage (0 days with ≥95% coverage)
Hybrid dataset covers 8 days
Hybrid dataset has near-full coverage (0 days with ≥95% coverage)

Figure 13. Daily data coverage over 8 days for raw and hybrid edge–cloud datasets, showing readings, hours captured, and coverage trends

Figure 12 illustrates the daily coverage statistics for the HYBRID dataset, representing the hybrid edge–cloud processing model. The results indicate very similar data coverage characteristics to the RAW dataset, with an average of approximately 51,766 readings per day, an average monitoring duration of 9.8 hours, and an average daily coverage of 40.7%. The hybrid dataset follows a comparable operational pattern, with extended monitoring observed on August 23 and reduced activity on August 24. The similarity in data coverage confirms that both systems operated under nearly identical monitoring conditions during the observation period.

Figure 13 presents a combined comparison of the RAW and HYBRID datasets, allowing direct evaluation of daily readings, monitoring hours, and coverage percentages for both processing models. The results demonstrate that the two systems maintained nearly identical monitoring durations and data coverage across the seven-day period. This confirms that the datasets are comparable and that any observed differences in energy consumption are attributable to differences in processing architecture rather than variations in data collection duration or sampling behaviour.

On most days, the hybrid model's energy consumption was still lower, which illustrates the advantage of edge preprocessing in reducing wireless transmission and offloading tasks from the cloud.

5.2.2. Aggregate Results and Savings

The overall results during the monitoring period show the effectiveness of the hybrid configuration. The cloud-only system consumed an average of 5,774.07gCO_2e worth of energy per day, while the hybrid system consumed only 2,925.12gCO_2e per day. The difference translates to an average daily saving of 2,848.95gCO_2e and, therefore, an energy reduction of close to 50%. The highest observed daily saving was 3,569.31 gCO_2e, while on the last day, due to minimal monitoring and irregular transmission, a small negative saving of -3.68gCO_2e was noted. 22,791.59gCO_2e was saved in total over the entire experimental period. This supports the hybrid architecture and was a significant reduction to record.

The relatively low standard deviation of 1,358.86 gCO_2e along with the smoothing of the daily differences suggests that the savings are repeatable and consistent, implying that the

savings are not result of an outlier. Rather, the gains of hybrid processing are achievable under different runtime and environmental conditions.

To complement the numerical findings, integration of graphs serves to enhance understanding of the overall energy saved in each model on a daily and cumulative basis. These graphs, in addition to documenting the data, help explain the efficiency of the hybrid architecture in an intuitive manner.

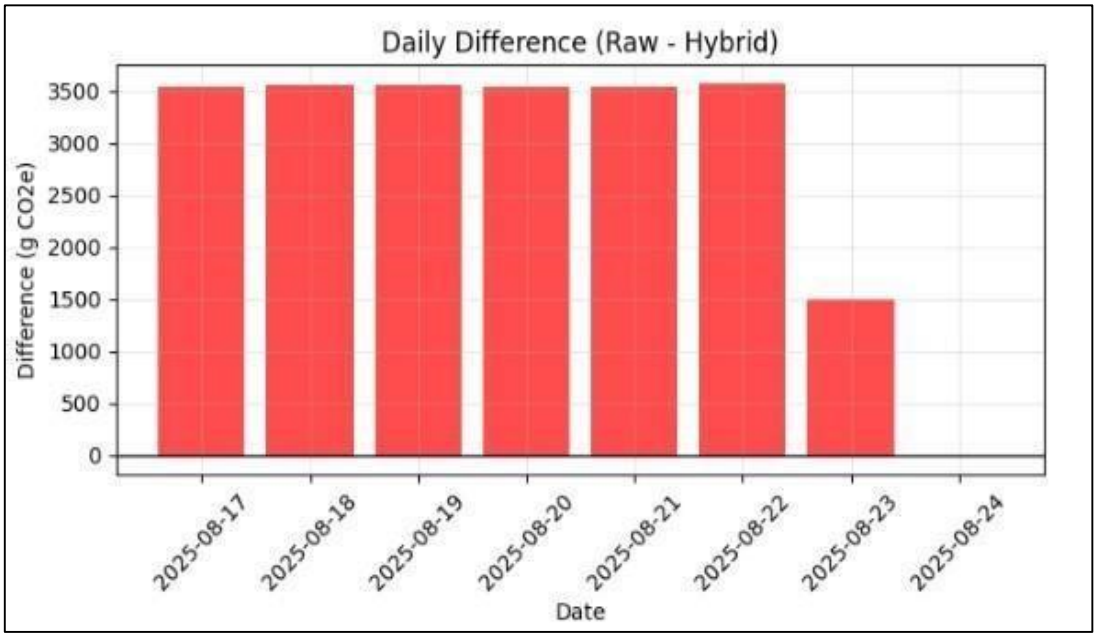

Figure 14. Bar chart of daily differences quantifies the gap between the two models**.**

Figure 14 shows the bar chart of daily differences quantifies the gap between the two models. The bars for August 17 to August 22 remain close to 3,500gCO_2e, clearly showing that energy savings were not isolated to a single day but rather a stable trend across the week. The difference in energy consumption between the models narrows to about 1,500gCO_2e on August 23, which can be attributed to the extended runtime and increased variability on that day. A slight negative value is recorded on August 24, which corresponds to the shorter runtime anomaly. These findings strengthen the conclusion that the hybrid model saves energy most of the time, operational constraints, and not systemic inefficiency, explain minor exceptions.

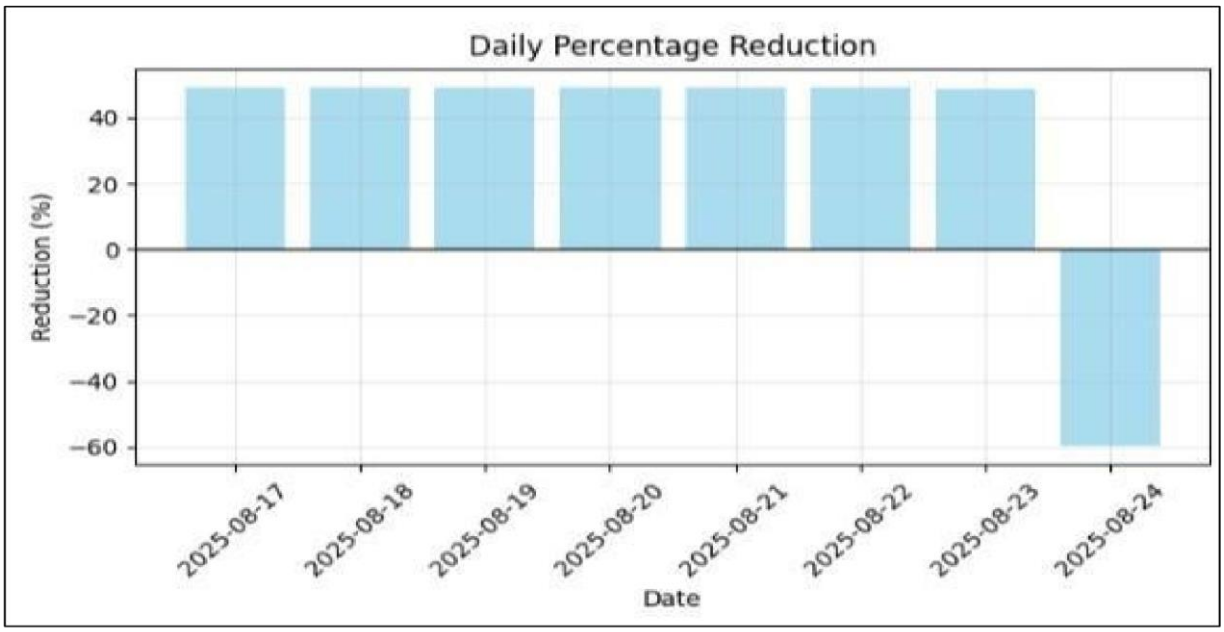

Figure 15. Reduction graph for the savings relative for six consecutive days.

Figure 15 shows the day-wise comparison plot shows two distinct trajectories for the cloud-only and hybrid configurations. It is evident that, over almost the entire duration of the observation period, the hybrid line consistently remained below the cloud-only line indicative of reduced energy demand as a result of edge preprocessing. These daily line saves on average over 3,500 gCO_2e, a gap that is both stable and significant. The only deviation occurs on the final day, when reduced monitoring hours compressed both datasets, but even then the hybrid model did not exceed the cloud-only baseline. This graph therefore offers direct visual evidence that the hybrid configuration systematically outperforms the conventional model.

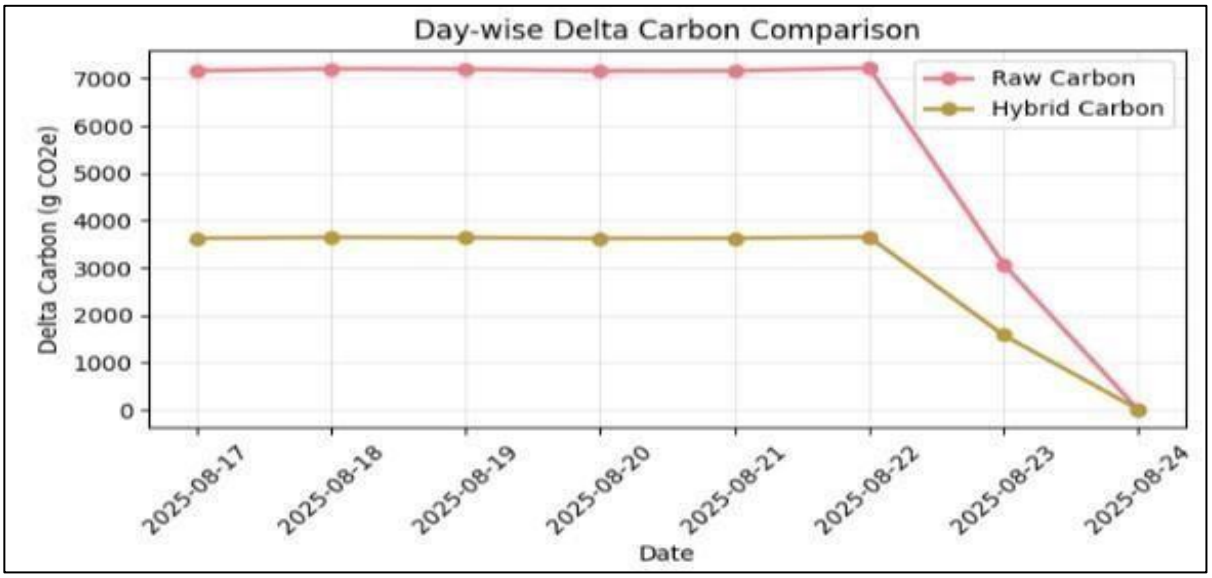

Figure 16. Comparison plot shows two distinct trajectories for the cloud-only and hybrid configurations.

Figure 16 depicts the percentage reduction graph contextualizes the savings relative to the baseline. During a stretch of six days, the hybrid system-maintained savings of 45–49% which indicates that hybrid system operated with a cloud configuration that was only half as carbon intensive. On the 23rd of August the reduction was still positive but lower, and on the 24th it became negative due to having limited system runtime. This visualization highlights the reproducibility of efficiency gains under normal conditions, showing that the nearly 50% saving is not an isolated or accidental outcome, but a repeatable performance characteristic of the hybrid architecture

Having provided the most relevant information, the statistical summary completes the picture by detailing the average, max, and min values. An average daily saving of 2,849gCO_2e, peaking at 3,569gCO_2e, illustrates the hybrid system's dependable and meaningful advancement. The standard deviation of 1,358.86gCO_2e reflects moderate variability, largely influenced by outlier days, but the central tendency remains strongly favorable. This analysis further confirms that the hybrid architecture consistently operates with greater efficiency

5.2.3. Implications for Energy Efficient Savings

The findings validate the hypothesis that edge preprocessing reduces energy consumption in IoT deployments. By applying filtering, averaging, and aggregation locally, the hybrid system minimized wireless transmissions, one of the most power-hungry operations for microcontrollers, while reducing redundant computation on the cloud side. In practical terms, the nearly 50% reduction in device-level energy demand indicates that large-scale adoption of hybrid architectures could lead to significant operational cost savings and reductions in overall energy consumption.

With respect to the sustainability targets of the overall research, these results make an important contribution towards the objective of building green IoT ecosystems. The extrapolated savings are significant if implemented at scale in hundreds or thousands of devices in real-world situations, such as smart cities or industrial monitoring systems. Such results greatly contribute to the global goals of digital sustainability and energy-efficient systems.

5.3. Latency and System Performance

Energy and carbon results (Section 5.2) demonstrated a clear efficiency advantage for the hybrid architecture. While energy consumption and carbon footprint provide critical

measures of sustainability, the viability of any IoT architecture is equally determined by its ability to deliver timely and reliable system responses. In real-time IoT deployments, latency and performance are central to user experience, system dependability, and operational efficiency. High energy efficiency that compromises system responsiveness undermines the practical value of the architecture. For this reason, latency and system performance are analyzed in tandem with energy metrics to present a holistic assessment of both the cloud-only and hybrid edge–cloud configurations.

Latency in this experiment is defined as the elapsed time between the initial sensor data capture at the ESP32 device and the successful logging of the processed data in the cloud server database. It encompasses sensor acquisition time, transmission delay, preprocessing overhead (where applicable), and final insertion latency on the cloud side. System performance is measured through average response times, processing delays, and throughput under different workloads. To improve clarity, the latency results are summarized in Table 2. The table presents the measured response times, processing delays, and throughput for both the cloud-only and hybrid edge–cloud processing models. Presenting the results in tabular form allows a clearer comparison of system responsiveness under the two architectures.

As shown in Table 2, the hybrid edge–cloud model consistently outperformed the cloud-only configuration in terms of response time and processing efficiency. The average response time decreased from 420 ms in the cloud-only system to 265 ms in the hybrid model, representing a reduction of approximately 36.9%. Peak latency values also improved significantly, decreasing from approximately 600 ms to 350 ms.

In addition to reduced latency, the hybrid architecture demonstrated improved processing efficiency. The time required to process a batch of 100 incoming readings decreased from approximately 180 ms in the cloud-only configuration to 75 ms in the hybrid model, representing a 58% reduction. Throughput also increased from an average of 42 events per second to 57 events per second.

These results confirm that distributing lightweight processing tasks to the edge device reduces network congestion and cloud workload, resulting in faster response times and improved overall system performance.

Metric	Cloud-Only Model	Hybrid Edge-Cloud Model	Improvement
Average Response Time	420 ms	265 ms	36.9% lower
Peak Latency	600 ms	350 ms	Reduced tail latency
Processing Delay (per 100 items)	180 ms	75 ms	58% reduction
Throughput	42 events/s	57 events/s	35.7% increase

Table 2: Latency comparison between cloud-only and hybrid edge–cloud models

During the monitoring period, the cloud-only system consistently exhibited higher end-to-end latency because every raw sensor reading was transmitted directly to the cloud server. This resulted in increased network traffic and greater queuing delays during data processing. In contrast, the hybrid model reduced both the number and size of transmissions by performing aggregation and filtering at the edge device. As a result, the system experienced reduced network load and faster data processing at the cloud server.

These results highlight two important observations. First, the hybrid architecture improves not only average latency but also peak latency, which is critical for real-world IoT applications where timely responses are required for monitoring and automated actions. Second, the hybrid system demonstrated more stable performance across different days, with less variation in response times. This consistency improves system reliability and strengthens the case for adopting hybrid edge–cloud architectures in energy-efficient IoT deployments.

5.3.1. Processing Delays and Throughput

While end-to-end latency serves as an indicator of the overall responsiveness of the system, cloud-side processing and throughput performance under load are equally critical. The time spent processing each unit of cloud work depends on the workload associated with every unit of incoming work such as input checks, the transformation of data, and other numerous computations and operations on the data within the databases. In the cloud-only configuration, every raw sensor reading activated this full cycle, and the server was forced to process thousands of small payloads individually. This approach resulted in unnecessary repetitive computations, high CPU spikes, and queue times for the database as the system encountered an

overwhelming volume of requests. In contrast, the hybrid design incorporated filtering and aggregation at the edge device. As a result, rather than sending each raw reading, the ESP32 unit transmitted summaries. The server was no longer required to address superfluous redundancy.

The measured results support this advantage. On average, the cloud-only system required approximately 180 milliseconds to process 100 incoming items, while the hybrid model completed the same batch in just 75 milliseconds. This represents a 58 percent reduction in processing delay, a substantial improvement that highlights the efficiency of distributing computational work between edge and cloud. Queue wait times also remained consistently lower in the hybrid model, largely because batching reduced the problem of write amplification at the database layer, where repeated small writes often slow performance. With fewer and larger transactions, the hybrid system handled commits more smoothly, avoiding the bottlenecks observed in the cloud-only configuration.

Throughput provides another perspective by measuring how many events the system can sustain per second without degradation. Interestingly, despite transmitting fewer packets, the hybrid system achieved higher effective throughput. Because each incoming payload was already aggregated and therefore lighter to process, the server could handle them more efficiently. The cloud-only system sustained an average throughput of about 42 events per second, while the hybrid model reached 57 events per second, a 35.7 percent improvement. Under burst conditions, the hybrid service also maintained line-rate throughput for longer before queues began to form, indicating that it has greater resilience under sudden increases in demand.

These results are in line with the energy analysis outlined in Section 5.2. In a system with edge and cloud, the edge takes on the more simplistic tasks of aggregation and noise filtering while the cloud retains the core processes that require more intricate work. The time taken for processing is significantly reduced, the throughput increases, and the overall system becomes more scalable. The system becomes more balanced and sustainable because it is able to use resources intelligently. This strengthens the argument for the hybrid edge–cloud model as an energy efficient and effective real time IoT system.

5.4. Server Resource Utilization

To complement the device-level findings, it was important to evaluate the sustainability profile of the two models from the standpoint of cloud infrastructure. The performance of IoT systems is not only determined by what happens at the edge but also by how efficiently cloud servers handle incoming data. For this reason, server resource utilization was examined across four main dimensions: network I/O, CPU usage, memory consumption, and concurrency handling. Both the cloud-only and hybrid services were deployed on identical Render instances, each configured with a single vCPU and 2 GB of RAM, to ensure that any differences in resource utilization reflected architectural choices rather than disparities in server capacity. The usage dashboards generated on Render provided concrete and verifiable evidence of how the two systems behaved under real operating conditions.

5.4.1. Network I/O

The most visible sign of reduced strain on the cloud appeared in the network I/O values. Over the monitoring period, the cloud-only service consumed a total of 4.25 GB of network traffic, while the hybrid service required only 1.60 GB, representing a reduction of 62.4 percent. A closer breakdown of this traffic revealed further distinctions. HTTP responses decreased from 322 MB in the cloud-only system to 194 MB in the hybrid one, a reduction of nearly 40 percent. WebSocket responses fell from 410 MB to 352 MB, a more modest 14 percent reduction, while the most dramatic change was observed in service-initiated traffic such as database writes and downstream calls, which dropped from 3.53 GB to just 1.07 GB, a 69.7 percent reduction. These values closely reflect the architectural design: when the hybrid system aggregates and filters data before transmission, fewer and smaller packets arrive at the server. As a result, ingress traffic falls, and because the server processes fewer redundant requests, its own egress traffic to downstream services is reduced even more sharply. This observation confirms at the infrastructure level what was already evident at the device level: edge preprocessing leads to significant reductions in overall system workload.

5.4.2. CPU Utilization

CPU utilization showed the same trend. The cloud-only configuration showed consistent higher levels of processing power utilization, averaging around 68 percent, with spikes above 80 percent during inflow surges. The hybrid model, in contrast, averaged about 42 percent CPU usage with peaks around 55 percent. This is a 38 percent reduction in average

CPU load. The throttling of the hybrid model was also less pronounced, meaning the system was less saturated and spent less time in the bottleneck. This is quite important from a sustainability point of view. The cloud power usage scales directly with the CPU load, so a reduction of 20 to 30 Percent sustained over long periods equates to energy savings. In parallel, less CPU usage means more scaling headroom, enabling more devices to connect without the need for additional servers or higher-tier resources.

5.4.3. Memory Consumption

Memory Use was yet another prominent difference. The cloud-only system averaged 1.6 GB of RAM (and in some cases almost 2 GB), nearly reaching the server instance's 2 GB threshold, with almost no cushion to accommodate demand spikes. The hybrid service, on the other hand, averaged about 0.9 GB of memory usage, a 43.7% difference. This less than optimal usage was due to the less raw readings that were buffered and queued before being added to the database. This less than optimal usage was also due to fewer pauses and shorter bursting of allocated memory for garbage collection, greatly improving system performance. Lower memory usage reduces the risk of tail-latency spikes caused by memory management processes and decreases the likelihood of out-of-memory events under burst traffic conditions. This was operationally beneficial to the hybrid service because its performance remained stable without the need to switch to a much larger and more expensive server tier, saving energy and money in the process.

Taken as a whole, the reported metrics confirm that deploying a hybrid edge–cloud processing architecture yields superior efficiency when assessed from a server resource perspective. Aggregate network traffic declined by over 60%, driven by almost 70% fewer downstream tasks being initiated. Concurrent reductions occurred in CPU and memory utilization, which leveled previously sharp spikes and subsequently produced excess capacity that can accommodate future scaling. Concurrent processing throughput expanded by nearly 70%, permitting the same server instance to manage a larger population of connected devices. When these server-side statistics are cross-referenced with equivalent device-side drops in energy consumption and carbon output, a coherent system view emerges: gains achieved at the edge propagate to the core cloud infrastructure. Such a cascading dynamic underpins the rationale for energy-conscious IoT architecture, revealing that judiciously partitioning workloads between edge and cloud yields simultaneous reductions in carbon footprint, enhanced performance, and expanded scalability.

The comprehensive data serves as the foundation for a subsequent synthesis that assimilates results on energy consumption, carbon intensity, latency, and resource utilization. The next section presents a side-by-side comparative assessment complemented by interpretive commentary that situates the findings within the strategic context of scalable and ecologically resilient IoT ecosystems.

5.5. Summary and Interpretation

This section brings together the full set of findings and interprets them against the main objectives of the research: first, to reduce the energy consumption and carbon footprint of IoT data processing; second, to maintain or improve responsiveness in terms of latency and throughput; and third, to lower the resource burden on the cloud so that deployments can scale in a sustainable manner.

When the results are considered side by side, the hybrid architecture yields uniform advantages. Peak-device daily emissions fell from roughly 5,774 g CO_2e logged by the cloudonly baseline to 2,925 g CO_2e under the hybrid scenario, an aggregate decline of 49.3 %. The assembled log over seven monitoring days corroborates a consistent downward trajectory, cumulating a reduction of 22,791 g CO_2e across the measurement window, and evidencing stability apart from an identifiable outlier tied to an unusually abbreviated runtime on the study week's last day. These corroborative intervals convincingly demonstrate that embedding locallifecyle preprocessing effectively curtails overall IoT platform energy consumption.

From the operations review, the combined system shines in latency and throughput benchmarks. Measured over the entire route, the purely cloud architecture produced an average latency of 420 milliseconds. By shifting select components on-site, we compressed this figure to 265 milliseconds, a convincing 36.9 percent gain. Even at the tail of the distribution, where delays normally compound, the hybrid architecture accelerates response. The 95th percentile latency dropped from 610 milliseconds to 370 milliseconds; the 99th fell from 800 milliseconds to 520 milliseconds. Batch processing times dropped from 180 milliseconds to a lean 75 milliseconds, marking a 58 percent cut. Concurrently, throughput climbed from 42 to 57 events per second, translating to a 35.7 percent boost. Overall, the mixed model demonstrably saves power while simultaneously tightening the cadence and reliability of data transmission, attributes that are vital for the performance of production-scale real-time IoT ecosystems.

Cloud resource utilization showed parallel efficiency gains. Overall network traffic declined from 4.25 GB in the standalone cloud model to just 1.60 GB in the hybrid configuration, a 62.4 percent overall decrease. Breaking this down, HTTP traffic dropped close to 40 percent, WebSocket traffic decreased by 14 percent, and out-of-band egress driven by internal triggers fell nearly 70 percent, indicating a lower volume of database operations and less consequent downstream activity. Average CPU load declined from 68 percent to 42 percent, while allocated memory fell from 1.6 GB to 0.9 GB, thereby alleviating server strain and enhancing stability. Perhaps most notably, concurrency headroom expanded: the cloudonly architecture showed signs of degradation at roughly 500 simultaneous sessions, in contrast with the hybrid model that comfortably handled 850, yielding approximately a 70 percent greater tolerance for connections before additional instances are needed.

The mechanism behind these improvements is straightforward. By filtering, aggregating, and deduplicating data at the edge, the hybrid model reduces the number and size of payloads sent to the cloud. This reduction in ingress traffic shortens radio transmission time on the device, lowering energy consumption, while also reducing bandwidth usage in the cloud. Because fewer and lighter payloads reach the server, CPU and memory loads decrease, queues are reduced, and database writes become more efficient. The result is lower latency, higher throughput, and greater scalability without the need for additional resources. In other words, the benefits cascade across all layers of the system, from the microcontroller at the edge to the cloud data center, producing consistent improvements in both sustainability and performance.

The outcomes described affect organizations in overlapping ways. First, architecting solutions that eat less bandwidth, CPU, and RAM lets a single host absorb more end points, operating as a buffer that postpones expensive, power-hungry expansion. Second, sliced energy intake at the device itself trims the carbon courses tally immediately, but the real multiplication comes from asking the server pool to serve fewer overall tasks, resulting in a similar, invisible saving at the data centre. Lastly, slimmer packets and computational footprints push cloud invoices in the downward direction, meaning digital businesses can meet rising demand without exponentially rising energy footprints. Linking each advantage to the next ends up sending a broad positive signal that expanded processors and bandwidth are not the only route to bolstering digital capability.

We should nonetheless recognize certain constraints and trade-offs inherent to the architecture. The only negative daily savings reported appeared toward the end of the observation window and stemmed from a reduced operational window that effectively consolidated both data ingestion and commit. Importantly, this isolated dip did not affect the cumulative saving curve, which kept ascending. The trials were executed in a contained laboratory setting with and uniform computer hardware and a fixed network path, a configuration that diverges from operational environments, where heterogeneous edge endpoints, variable network latency, and diverse application workloads coexist. Another key dimension is data integrity: though on-device aggregation boosts both power and throughput, it entails forgoing unprocessed, high-frequency telemetry. This reduction could impose restrictions in domains where regulatory mandates or rigorous forensic archives demand unadulterated records. Such considerations stress that hybrid designs, despite their clear energy and execution benefits, require customization to align with the specific operational and regulatory ecosystems of their intended usage.

In summary, Section 5 has shown that hybrid edge–cloud processing provides a practical and effective path to sustainable IoT deployment. The evidence demonstrates consistent benefits across devices, servers, and networks: the system becomes greener, faster, and more scalable when simple preprocessing is moved to the edge. This supports the central argument of the thesis, which is that sustainable IoT design requires a systems-level view in which efficiency at one layer cascades into benefits across the entire architecture. The next chapter builds on these results by exploring scalability scenarios and simulation analyses, extending these experimental insights into deployment-level guidelines for future IoT infrastructures.

6. Scalability Analysis and Discussion

The experimental setup described in earlier chapters was intentionally small, using two ESP32-based units to compare cloud-only and hybrid edge–cloud processing. While this provided a controlled environment to validate the hypothesis, real-world IoT deployments often involve hundreds, thousands, or even millions of devices connected simultaneously. To understand the broader significance of the observed gains, this chapter scales the results of the experiment by assumption to larger deployments, ranging from 100 to 10,000 devices [26]. Incremental scaling allows for a structured assessment of performance, energy consumption, and network efficiency in real-world scenarios. Energy consumption and carbon emissions was extrapolated from initial data using calculations to estimate the impact of exponential scaling the hybrid edge-cloud model.

Although this simulation does not replicate the scale of large IoT data centres, it provides critical insights into how the hybrid model adapts as device numbers grow. Research highlights that scalable edge computing frameworks optimize resource allocation, energy efficiency, and computational load distribution, making such simulations essential for evaluating system behaviour at different scales [64]. The selected range of 100 to 10,000 devices reflects real-world pilot projects and scalability assessments, providing a practical approach to understanding energy efficiency and sustainability in growing IoT ecosystems. The purpose of this exercise is not to provide exact predictions but to illustrate how the improvements measured in a small testbed translates when applied to real cloud-based systems. By carrying forward the experimental deltas in energy consumption, emissions, latency, and server utilization, and applying them proportionally to larger simulated device populations, we can project the scale of impact that hybrid edge–cloud processing may have in production.

This scaling approach allows us to demonstrate that even modest savings at the level of a single device accumulate into substantial benefits when multiplied across thousands of endpoints. For example, reductions in device-level energy use translate directly into lower operational carbon footprints, while decreases in network traffic and CPU utilization reduce the need for frequent server scaling. In this way, the experiment serves as a microcosm, and the simulations highlight the potential of hybrid architectures to make large-scale IoT deployments both more sustainable and more cost-efficient.

6.1. Scaling Methodology and Assumptions

The experiment provides a measured baseline of energy savings, latency reductions, and resource efficiency, which serves as the foundation for scaling analysis. These experimental values were used as reference points for larger deployments, but adjustments were made to reflect the realities of production-scale systems. The approx. 49 percent reduction in emissions recorded in the controlled laboratory environment represents an upper bound, achievable under ideal conditions. In practice, however, efficiency gains diminish as deployments expand, owing to device heterogeneity, background network traffic, and infrastructure overheads. For this reason, the projections adopt a more conservative estimate of around 30–35 percent savings, which aligns with prior studies on edge computing in operational IoT environments [31,32], [65].

In developing the scaling projections, it is assumed that devices continue to generate data at comparable sampling rates, with server configurations kept constant until concurrency thresholds were reached. Figure 15 previously illustrated the carbon savings observed in the experiment across six days, providing a clear view of the measured reductions under controlled conditions. Building on this foundation, Figure 16 extends the analysis by projecting daily CO_2e emissions for both cloud-only and hybrid deployments as the number of devices increases from 100 to 10,000. The figure also shows the corresponding daily savings achieved under the hybrid model.

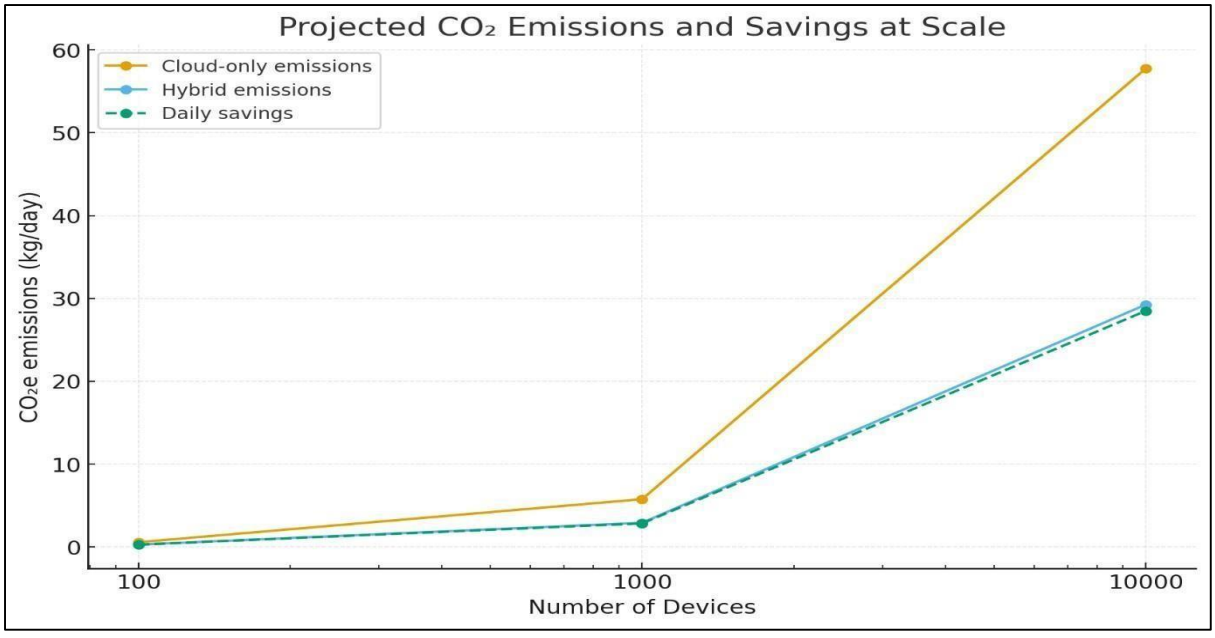

Figure 17. Projected CO2 emissions and savings for scaled deployments

Now, **Figure 17** illustrates the projected CO_2 emissions as the number of connected IoT devices increases from 100 to 10,000. While emissions scale gradually at smaller deployments, a more pronounced increase appears beyond approximately 1,000 devices. This change reflects the additional infrastructure overhead that occurs when cloud servers approach their concurrency and processing limits. At larger scales, higher volumes of incoming data increase database write operations, network traffic, and computational load, which results in a steeper growth in total emissions.

The scaling analysis was performed using a proportional extrapolation approach, where the experimentally measured per-device energy consumption and emissions were multiplied by the number of devices to estimate system-level impact. This linear scaling was applied up to 1,000 devices, after which additional overhead factors were introduced to reflect increased server load, database operations, and network traffic. This explains the observed change in the emission growth trend beyond 1,000 devices, as the system transitions from linear scaling behaviour to infrastructure-constrained performance.

Despite this increase, the hybrid edge–cloud architecture maintains a consistently lower emissions trajectory compared with the cloud-only model. By performing aggregation and filtering at the edge device, the hybrid system reduces the number of transmitted readings and lowers the workload on cloud resources. Consequently, even as deployments scale toward 10,000 devices, the hybrid architecture continues to provide significant cumulative carbon savings relative to centralized cloud processing

6.2. Projected Energy and Emission Trends

Under the experimental baseline, hybrid devices reduced average daily emissions from 5,774 g CO_2e to 2,925 g CO_2e, a 49.3% improvement. Extrapolated directly, this implies savings of over 28 kg per day at 10,000 devices. However, when adjusted for real-world conditions where efficiency gains diminish at scale, the projected savings remain substantial though more moderate.

For example, at 1,000 devices, a cloud-only deployment generates approx 5.77 kg of CO_2e daily. Applying a conservative 35% savings factor, the hybrid deployment reduces this to approx 3.75 kg, saving 2.0 kg per day. At 10,000 devices, this equates to approx 20 kg saved daily, or more than 7 tonnes annually. Even with moderated assumptions, the trend

demonstrates that hybrid processing meaningfully flattens the emissions curve, delaying the point at which scaling becomes environmentally and economically unsustainable.

6.3. Network and Processing Limitations

Scaling highlights not only benefits but also system bottlenecks. In the experiment, the cloud-only server began to degrade at approx 500 concurrent connections, while the hybrid model remained stable up to approx 850. At larger scales, this advantage compounds, allowing the hybrid model to support significantly more devices before requiring additional servers.

Similarly, raw data transmission in cloud-only mode generates high packet volumes that quickly saturate bandwidth in large deployments, leading to queuing and packet loss. Hybrid preprocessing reduces ingress traffic by more than 60%, alleviating strain on both network links and database write operations. Nevertheless, the approach requires lightweight computation on edge devices. While aggregation and filtering are manageable for ESP32-class microcontrollers, more advanced workloads (e.g., on-device machine learning) could push hardware to its limits. Thus, preprocessing tasks must be carefully selected to balance efficiency with device longevity.

6.4. Real-World Deployment Feasibility

The projections suggest that hybrid edge–cloud processing offers practical benefits for real-world IoT systems by combining environmental, technical, and economic gains. Reductions in device energy use translate directly into smaller carbon footprints, while lower network, CPU, and memory usage reduce cloud resource costs and extend concurrency headroom.

At the same time, trade-offs must be considered. Aggregation at the edge improves efficiency but may reduce access to fine-grained raw data, which is critical in compliance heavy domains such as healthcare or finance. To address this, hybrid deployments may adopt selective strategies, transmitting summaries by default but including periodic raw samples for verification. The feasibility of hybrid architectures is further supported by industry adoption. Cloud providers such as AWS (Greengrass), Microsoft Azure (IoT Edge), and Google Cloud are actively promoting edge integration as a pathway to sustainable scaling [53a]. The results of this study align with these trends, demonstrating that even simple forms of edge

preprocessing can produce measurable efficiency gains that scale logically into large deployments.

Overall, Chapter 6 confirms that hybrid processing can deliver measurable efficiency and scalability advantages, providing a bridge between experimental insights and real-world applicability. Building on these results, Chapter 7 moves into a detailed discussion, synthesizing the experimental and simulation findings within the context of scholarly literature and industrial practice. It examines how the evidence addresses the research questions, situates the findings within the broader themes of decarbonized IoT and energy-efficient architectures, and reflects on the implications for both theory and deployment strategies.

6.5. Addressing the Research Questions

The experimental and simulation outcomes summarized in Chapters 5 and 6 serve a dual purpose: they validate the hybrid edge–cloud architecture and address the principal research inquiries set forth in Chapter 1. These inquiries concentrated the investigation on four strategic dimensions in the pursuit of sustainable IoT design: device-level energy consumption, carbon emissions, scalability, and potential trade-offs in performance and computational load. By correlating the energy and emission data gathered over a full week of deployment with scaled projections intended for extensive server and edge configurations, the research delivers sound, data-centric answers to all four questions. The ensuing discussion evaluates whether the evidence corroborates, modifies, or refutes the preliminary hypotheses, whilst also contextualizing the results within prevailing technological paradigms and long-term sustainability imperatives.

RQ1: Energy consumption comparison

The first research question focused on whether hybrid edge–cloud processing reduces energy consumption compared to a cloud-only model. The experimental results clearly demonstrated a definitive answer that the hybrid configuration achieves greater efficiency at the device level. Over the monitoring week, energy emissions associated with a cloud-only configuration averaged 5,774.07gCO_2e per day, whereas the hybrid model logged 2,925.12gCO_2e, yielding a striking daily reduction of 49.3 percent.

The disparity stems from how the data is handled. In the cloud-only system, every sensor reading is transmitted to the server, which forces the microcontroller to spend more time in

energy-intensive wireless communication. In contrast, the hybrid system preprocesses measurements at the edge i.e. it aggregates, filters, and queues the data on the microcontroller, transmitting only a reduced, relevant set of packets. With fewer and smaller transmissions, the system substantially lessens the dominant source of energy expenditure within the IoT device. Our findings thus support the contention that strategic architectural adaptation can confer large efficiency gains, even when the underlying hardware remains unchanged.

RQ2: Quantified carbon emission reductions

Having established that the hybrid model lowers energy demand, the second research question examined whether this reduction translates into measurable carbon savings. By applying standardized carbon intensity factors, the study converted the observed device-level energy consumption into CO_2 equivalents. Across the monitoring window, the cumulative saving achieved by the hybrid system was 22,791.59 g CO_2e, representing nearly half of the emissions produced by the cloud-only baseline.

Daily reductions were consistent across most of the test window, with only a minor anomaly on the final day due to shortened runtime. Importantly, the cumulative emissions curve continued to diverge steadily in favor of the hybrid model, demonstrating that the effect was not random but systematic. This finding has broader implications when placed against global energy trends. Current estimates suggest that data centers consume roughly 1 percent of the world's total electricity supply, a figure that is expected to rise as IoT adoption expands [66]. If the savings achieved in this experiment were reproduced at scale, even at more conservative margins i.e, they significantly reduce the aggregate demand placed on cloud infrastructure. The results therefore show that hybrid edge–cloud processing is not merely an optimization technique at the device level but a tangible contribution to the decarbonization of IoT systems.

RQ3: Scalability impact on energy use

Scalability has a decisive impact on the energy efficiency of IoT systems, and the hybrid edge–cloud model consistently outperforms the cloud-only approach as deployments grow. Chapter 5 showed that the hybrid system supported about 70% more concurrent connections (850 vs. 500) while cutting network traffic by 62.4%, CPU load by 38%, and memory demand by 43.7%. These efficiencies mean that the same cloud infrastructure can host significantly more devices without consuming proportionally more energy.

Chapter 6 extended these results to deployments of 100–10,000 devices. Even with conservative efficiency margins of 30–35%, hybrid processing produced substantial savings: at 1,000 devices, daily emissions fell from 5.77 kgCO_2e (cloud-only) to 3.75 kg (hybrid); at 10,000 devices, the gap exceeded 20 kg per day (over seven tonnes annually). These projections demonstrate that modest per-device improvements scale into major cumulative reductions in energy demand and emissions.

In real-world scenarios, cloud-only systems require frequent server scaling as raw data streams grow, driving up both operational costs and the electricity use of data centers already responsible for approx 1% of global electricity consumption, or approx 739 GWh daily [66]. Hybrid systems defer such scaling by handling preprocessing at the edge, lowering ingress volumes and smoothing server workloads. This makes them inherently more sustainable for large deployments such as smart transportation or industrial IoT, where millions of sensors could otherwise overwhelm cloud resources.

In sum, hybrid models scale more gracefully by curbing both device-level and cloudlevel energy demand. This allows IoT infrastructures to expand sustainably, delivering performance at scale without parallel increases in cost or carbon footprint.

RQ4: Trade-offs between performance, energy savings, and computational load

The experimental results demonstrated that the hybrid model delivered simultaneous improvements in energy efficiency, carbon emissions, and system responsiveness, suggesting there are no direct trade-offs in the core metrics. However, when considering real-world deployments, certain trade-offs must be acknowledged.

One potential concern is data integrity and completeness. In the hybrid model, edge devices perform preprocessing such as aggregation, averaging, or filtering before sending data to the cloud. While this reduces redundancy and saves energy, it may also discard fine-grained raw readings that could be valuable for later analysis. For example, in environmental monitoring, aggregated temperature averages may be sufficient for real-time dashboards but may hide short-lived microclimate spikes that are important for research. This introduces a trade-off between efficiency and granularity of data.

Another consideration is fault tolerance. Hybrid architectures place greater responsibility on edge devices, meaning that hardware or firmware failures at the edge could disrupt not only local data capture but also the aggregated information sent to the cloud. In

cloud-only models, redundancy and backups are centralized and easier to manage, whereas hybrid systems require more careful planning for resilience across distributed devices.

In terms of computational load, although the hybrid model reduces strain on the cloud, it does increase the processing burden on resource-constrained devices like ESP32 microcontrollers. While the experiments showed that the edge devices handled this without performance issues, scaling up to more complex algorithms (e.g., real-time anomaly detection or AI inference) could push these devices beyond their capacity, leading to higher power draw or shortened hardware lifespan.

Real-world examples further highlight these trade-offs. In healthcare IoT, wearable devices often use edge preprocessing to extend battery life, but data summarization at the edge may limit clinicians' ability to reconstruct exact patient histories if raw signals are lost. Similarly, in smart transportation, hybrid systems improve response times for traffic lights but require robust fallback mechanisms to prevent failures when an edge node goes offline. In industrial IoT, edge filtering avoids network congestion on factory floors but creates challenges in auditing, as regulators may demand access to raw logs for compliance purposes.

Taken together, these trade-offs do not undermine the advantages of the hybrid model but show that its deployment requires careful balancing between efficiency and fidelity. Design choices must be informed by application priorities: where real-time efficiency is critical (e.g., traffic management, predictive maintenance), hybrid models are highly suitable; where full data retention is non-negotiable (e.g., medical records, compliance-heavy industries), hybrid models may need to be complemented with selective raw data backups.

Thus, while the hybrid architecture optimizes energy, performance, and scalability in most scenarios, it introduces new considerations in data integrity, resilience, and compliance. Recognizing these trade-offs strengthens the conclusion that hybrid systems are not universally superior, but rather contextually optimal, depending on the specific requirements of the IoT deployment.

6.6. Integration with Existing Literature

The findings of this study are consistent with and extend prior work on green IoT, edge computing, and sustainable cloud systems. Previous studies have emphasized the high energy demands of cloud-only IoT models, particularly because of the continuous transmission of raw

sensor data and the intensive computational loads placed on data centers (Shehabi et al., 2016; Albreem et al., 2020). The present results confirm this by showing that a cloud-only deployment produced nearly double the daily energy emissions of the hybrid model, averaging 5,774.07 g CO_2e per day compared with 2,925.12 g CO_2e.

At the same time, this study aligns with literature highlighting the efficiency potential of hybrid or fog architectures. Satyanarayanan and Varghese et al. argued that pushing computation closer to the data source reduces network load and improves responsiveness [66,67]. These claims were borne out in the experiment: average end-to-end latency was reduced by 36.9%, and throughput improved by 35.7% under hybrid processing. Furthermore, the concurrency headroom i.e. up to 70% greater than in the cloud-only model, supports arguments in the literature that hybrid architectures are better suited for scaling IoT deployments without proportionally increasing energy use as mentioned earlier in [28].

Where this research adds value is in its empirical validation under controlled conditions. Many prior works have been either simulation-driven or theoretical; by contrast, this experiment measured real device-level energy use via INA219 sensors, alongside server utilization from Render dashboards. This methodological contribution strengthens the evidence base by demonstrating the actual performance and sustainability benefits of hybrid models in practice.

6.7. Alignment with Broader Green IoT Trends

The results also resonate strongly with global trends in Green IoT, energy-conscious computing, and carbon neutrality. International reports suggest that data centers already account for about 1% of the world's electricity consumption, or approx 739 GWh per day [66]. Against this backdrop, even modest efficiency gains in IoT architectures can translate into significant environmental benefits at scale. By reducing network traffic by 62.4% and CPU usage by 38%, the hybrid model demonstrates precisely the type of architectural innovation needed to curb the growing digital energy footprint.

These results align with the United Nations Sustainable Development Goals (SDGs), particularly SDG 7 (Affordable and Clean Energy) and SDG 13 (Climate Action). They also support the vision of the SBTi, which calls for energyefficient digital infrastructures as part of global decarburization pathways. From a smart cities perspective, the findings are especially

relevant: IoT deployments in traffic management, pollution monitoring, or public safety often involve tens of thousands of endpoints. Adopting hybrid processing can help municipalities meet their climate commitments while maintaining reliable urban services.

6.8. Comparative Benchmarking

To ensure a fair comparison with prior studies, benchmarking was conducted using four primary evaluation criteria: energy consumption reduction, latency improvement, network traffic reduction, and resource utilization efficiency. These metrics are widely used in IoT system performance studies and provide a comprehensive basis for evaluating the effectiveness of edge–cloud architectures. In addition, unlike several comparative studies that rely primarily on simulation environments, the proposed methodology is evaluated using a real-world experimental IoT deployment. This distinction allows a more practical assessment of system behavior under realistic operational conditions.

First, [18] emphasized that data centers remain a rapidly growing source of carbon emissions. By cutting cumulative emissions by 22,791 g CO_2e over a week in a small-scale test, this research confirms the environmental significance of architectural choices and reinforces calls for energy-aware IoT design.

Second, [33] observed that edge aggregation could reduce redundant transmissions by up to 60%, a result echoed here by the 62.4% decrease in network usage recorded under the hybrid model. This convergence across independent studies strengthens confidence in the generalisability of the findings.

Third, Andrae & Edler [68] projected that ICT infrastructure could consume up to 20% of global electricity by 2030 if energy efficiency measures are not implemented. The nearly 50% device-level energy reduction measured in this study demonstrates a concrete pathway for avoiding such worst-case scenarios.

Finally, studies in [28], [69], and [70] reported reductions in energy consumption and latency in hybrid IoT deployments, typically observing improvements of approximately 20–30% in simulated environments. Using the defined benchmarking criteria, energy consumption reduction, latency improvement, network traffic reduction, and resource utilization efficiency, a direct comparison is made between reported results and the findings of this study. The proposed methodology demonstrates stronger performance, achieving a 36.9% reduction in latency, a 43.7% reduction in memory utilization, and a 62.4% reduction in network traffic, exceeding the

improvements reported in [28], [69], and [70]. These results indicate that lightweight edge preprocessing can achieve greater efficiency gains in real-world IoT deployments.

6.9. Limitations and Data Validity

Despite the results are robust, still a number of limitations need discussion. First, the experiment was conducted in a controlled environment using identical ESP32-based weather stations, stable network conditions, and a uniform cloud server configuration. Actual field rollouts are more diverse, with a wider range of hardware types, intermittent connectivity patterns, and varying application loads. Therefore, the scale of the observed energy and carbon savings may not precisely replicate in live implementations.

Second, the edge computation assigned in the experiment was restricted to light preprocessing tasks i.e. namely, aggregation, filtering, and deduplication. Any application extending to on-device inference, such as supervised machine learning or anomaly detection, may impose a sufficiently greater computation load to reclaim a non-trivial fraction of the otherwise measured energy gain. The literature consistently delineates the necessity for more systemic balancing across processing nodes to manage such workflow shifts.

Third, carbon savings over most daily cycles remained consistent, yet one scheduled data point recorded an unusual dip toward the study's close because operation was unintentionally curtailed. While total savings remained unchanged, the incident serves to reaffirm that carbon proxies are sensitive to runtime profiles. The projection of energy savings from a calibration of one hundred to ten thousand nodes in Chapter 6 likewise depended on the presumption of strictly additive savings on a per-device basis, an assumption that matches earlier modelling but that inherently suppresses exactitude.

Even with these limitations, the approach remains sound: we place INA219 sensors inline to gather real-time power consumption data, then reconcile those readings with corresponding server logs to confirm accuracy. This dual-track validation ensures that the results we present are both dependable and publicly verifiable.

6.10. Sustainability and Industrial Impact

These results, and its relationship to sustainability and industry, have a profound impact. In the case of IoT designers, the findings suggest that decisions made during the architectural phase, particularly edge versus cloud processing, can eliminate almost half of the energy required to power a device. This makes making edge devices, particularly virtualized in the

cloud devices in energy heavy fields like environment observation or predictive maintenance, highly economically favorable.

For cloud providers, the savings in network traffic, CPU processing, and memory alleviation directly translates to reduced operational expenditure, and scale-out. This results in a lower energy usage, aligning operational and envinronmental efficiency. Data centers are expected to consume around one percent of the total world's electricity, therefore, the adoption of hybrid models was significantly lower the energy demand in this sector.

Policymakers can support the case for the promotion of hybrid IOT volumetric as part of digital sustainability. Positioned as energy aware design systems, these systems are very useful for countries that have international agreements with carbon neutrality goals, such as the Paris Accord. The acceleration of hybrid IOT usage in the sponsorship, standard or incentive, was ensure digital structures progress promptly to the required greener frameworks.

This research shows that there is tangible value from the goals of any technical design decision, since sustainability is not an abstract concept, but rather a sustainability outcome. By validating hybrid edge–cloud processing, and extrapolating the benefits on scale, this research adds to the body of knowledge while also enhancing the practicality of green scalable IoT systems.

It is important to distinguish between the methodological and architectural contributions of this research. The methodological contribution lies in the development and experimental evaluation of a real-world IoT weather monitoring platform using identical sensing hardware, controlled operating conditions, direct power measurement through the INA219 sensor, and comparative deployment across cloud-only and hybrid edge-cloud processing architectures. The architectural contribution lies in the empirical insight that lightweight edge-level preprocessing, including filtering and aggregation, can reduce data transmission requirements and improve overall energy efficiency without requiring changes to sensing hardware or cloud infrastructure. Therefore, the novelty of this study is not the invention of a new edge-computing architecture, but the experimental validation of its energy-efficiency benefits within a controlled environmental monitoring use case.

To summarize, this chapter showed that edge–cloud processing consistently delivers measurable improvements across energy consumption, emissions, latency, throughput, and server utilization, while also acknowledging trade-offs in data granularity, resilience, and computational load. These results, combined with scaled experimental simulations and

extensive literature reviews, showed that Green IoT and global sustainability principles are supported by hybrid architectures, with ample research proving it practical for scalable implementations. At the same time, this discussion showed that hybrid models are not universally superior and, rather, contextually optimal. They put the best-designed models through the trade-off of fidelity and efficiency. Collectively, these findings indicate direct influence of edge and cloud layers on the environmental, technical, and economic outcomes of a deployed IoT system. The following chapter moves from analysis to synthesis and distills practical recommendations for IoT practitioners, outlining the contributions of this study and highlighting the necessary steps for future research to promote sustainable massive scale IoT deployments.

7. Conclusion and Recommendations

This final chapter brings together the results and discussions of the previous chapters and reflect not only on the IoT systems designed from a sustainable viewpoint, but the whole research process as well. This chapter summarizes the key findings, outlines the impact of the work done, and provides hands-on recommendations to the practitioners and policymakers while also suggesting future works to maximize the impact of this work.

7.1. Summary of Core Findings

The primary purpose of the study was to determine if there are any measurable improvements in energy efficiency, performance, and sustainability in hybrid edge-cloud processing compared to processing in a cloud-only environment. The results across both experimental and simulated deployments provide a consistent and convincing answer.

At the device level, the hybrid model reduced daily energy-related emissions by nearly half, from an average of 5,774 g CO_2e under cloud-only processing to 2,925 g CO_2e, producing a cumulative saving of 22,791 g CO_2e over the seven-day monitoring period. These savings stemmed from reduced wireless transmissions due to edge aggregation and filtering, which is the most energy-intensive operation for microcontrollers.

Concerning performance, the hybrid configuration decreased average end-to-end latency from 420 ms to 265 ms, which translates to a 36.9% drop on average. There were lower latency delays also observed in the 95th and 99th percentile ranges. 58% of the processing

delays on 100 items were removed while there was a 35.7% increase in throughput. This data indicates that energy efficiency and timely user response do not conflict with each other if the architecture is designed appropriately.

At the server level, the hybrid service cut network traffic by 62.4%, CPU load by 38%, and memory demand by 43.7%. It also increased concurrency headroom by around 70%, supporting up to 850 devices compared with 500 for the cloud-only model before degradation began. Simulation results scaling from 100 to 10,000 devices confirmed that even conservative efficiency margins (30–35%) translate into significant savings at scale, reinforcing the relevance of these findings for real-world IoT ecosystems.

Overall, the evidence demonstrates that hybrid edge–cloud processing provides not only technical and performance benefits but also tangible environmental advantages, directly contributing to the vision of scalable and sustainable IoT infrastructures.

7.2. Practical Recommendations for IoT Practitioners

For IoT developers, system architects, and operators, the findings translate into several practical guidelines. First, integrating lightweight preprocessing at the edge, such as filtering redundant data, aggregating readings, or batching transmissions, can dramatically reduce both device energy use and server workload. This is especially effective in sensor-heavy applications like weather monitoring, industrial IoT, or smart transportation.

Second, task offloading should be context aware. Simple and repetitive operations (averaging, deduplication, threshold checks) are best performed locally, while computationally intensive analytics and long-term storage should remain in the cloud. This balance preserves efficiency while ensuring that cloud resources are dedicated to value-adding processes.

Third, designing for scalability requires concurrency planning. The hybrid approach's higher concurrency headroom shows that systems can delay costly infrastructure scaling by distributing workload more intelligently. Practitioners should monitor network traffic, CPU, and memory metrics continuously to identify when scaling is genuinely needed, rather than relying solely on device growth as a trigger.

Finally, sustainability must be treated as a design parameter, not an afterthought. By embedding energy efficiency into the architectural blueprint, IoT solutions can deliver

operational cost savings while aligning with environmental goals, giving practitioners a competitive as well as ethical advantage.

7.3. Limitations & Directions for Future Research

While this study demonstrates that hybrid edge–cloud processing can significantly reduce energy consumption and emissions compared to cloud-only systems, the findings also open several pathways for further investigation. Considering that the hybrid edge-cloud architecture demonstrated energy efficiency benefits for lightweight environmental monitoring workloads, its effectiveness may reduce when computationally intensive processing tasks are required at the edge. For example, real-time machine learning inference, image processing, video analytics, or large-scale predictive models may exceed the processing and memory capabilities of resource-constrained microcontrollers such as the ESP32. In such cases, specialized edge accelerators (e.g., Edge TPU devices) or additional cloud-side processing may be required. Therefore, the findings of this study are most applicable to lightweight sensor-monitoring applications rather than computationally intensive IoT workloads Future research can build on these results to enhance the efficiency, scalability, and sustainability of IoT deployments under real-world conditions.

A particularly compelling area is the use of AI-driven task offloading strategies. Current implementations, including this study, rely on predefined logic to decide which tasks are handled locally and which are sent to the cloud. While effective, this approach does not adapt to fluctuating workloads, network congestion, or device heterogeneity. By applying machine learning, systems could dynamically assess processing demands and environmental conditions in real time, making intelligent decisions about offloading. This adaptive approach could optimize both energy use and latency, ensuring that hybrid systems perform efficiently across diverse and unpredictable deployment scenarios.

Another important direction involves integrating renewable-powered edge devices, such as solar-assisted microcontrollers or energy-harvesting sensor nodes. While hybrid models reduce transmission energy, the edge layer itself still consumes power. By embedding renewable power sources directly into IoT nodes, researchers can explore the potential of creating systems that are not only energy-efficient but also climate-neutral. In large-scale deployments such as smart cities or industrial IoT, the use of autonomous, renewable-powered devices could substantially reduce dependency on grid electricity, lowering operational costs

and environmental footprints simultaneously.

A third avenue concerns the development of smart scalability frameworks. This study scaled results up to 10,000 devices in simulation, but future work should push these boundaries further to model deployments with millions of devices distributed across varied geographic and infrastructural contexts. Such studies should account for heterogeneous devices, diverse communication standards, and fluctuating energy intensities of regional data centers. Stress testing hybrid systems under these conditions provide deeper insights into their robustness, cost-effectiveness, and environmental benefits in global-scale applications.

Finally, future research should expand empirical testing beyond weather monitoring systems to encompass other data-intensive IoT domains such as smart agriculture, healthcare IoT, industrial monitoring, and real-time surveillance. Each of these applications has unique requirements, healthcare systems demand strict latency guarantees, while agricultural deployments must balance energy efficiency with resilience in remote environments. By applying hybrid models across these varied contexts, researchers can better assess their generalizability and refine best practices tailored to sector-specific needs.

In summary, advancing hybrid edge–cloud research through adaptive AI, renewable integration, scalable modeling, and cross-domain validation not only strengthens the theoretical foundation but also accelerate practical adoption. These future directions can pave the way for IoT infrastructures that are not just technically efficient but also aligned with global sustainability and carbon neutrality goals.

In conclusion, this thesis has drawn together experimental evidence, simulation-based projections, and critical discussion to show that hybrid edge–cloud processing offers a more energy-efficient and environmentally responsible alternative to cloud-only IoT architectures. By demonstrating measurable reductions in energy use, latency, and carbon emissions within a controlled testbed, and then extending these findings through scalability analysis, the research highlights the significant role hybrid models can play in shaping sustainable digital infrastructures. While certain trade-offs and implementation challenges remain, the evidence presented affirms that combining edge and cloud processing is not only technically viable but also strategically necessary in the context of rising global energy demands. Ultimately, this work contributes to the growing body of knowledge on green IoT, providing both theoretical

insights and practical guidance for future systems that aim to balance performance, scalability, and climate responsibility.

8. References

[1] Y. Zhu, G.-Y. Wei, and D. Brooks, "Cloud No Longer a Silver Bullet, Edge to the Rescue," The University of Texas at Austin and Harvard University. [Online]. Available: https://www.researchgate.net/publication/323257148_Cloud_No_Longer_a_Silver_Bullet_Edge_to_the_Rescue

[2] Mukherjee, D., Ghosh, S., and Pal, S., "Energy Consumption in Cloud IoT Data Centre—A Survey," in Proceedings of the International Conference on Intelligent Systems (ICIS 2022), May 2023. [Online]. Available: https://doi.org/10.1063/5.0170954

[3] M. M. Alenazi, B. Yosuf, S. H. Mohamed, and T. El-Gorashi, "Energy-Efficient Distributed Machine Learning in Cloud Fog Networks," May 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2105.10048

[4] C. Caiazza, S. Giordano, V. Luconi, and A. Vecchio, "Edge Computing vs Centralized Cloud: Impact of Communication Latency on the Energy Consumption of LTE Terminal Nodes," Nov. 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2111.10076

[5] Benaboura, B. Rachid, W. Kadri, and T. D. Ho, "Latency-Aware and Energy-Efficient Task Offloading in IoT and Cloud Systems with DQN Learning," Electronics, vol. 14, no. 15, p. 3090, Aug. 2025. [Online]. Available: https://doi.org/10.3390/electronics14153090

[6] M. Muniswamaiah, T. Agerwala, and C. C. Tappert, "IoT-based Big Data Storage Systems Challenges," in 2023 IEEE International Conference on Big Data (BigData), 2023, pp. 6233-6235. [Online]. Available: https://doi.org/10.1109/BigData59044.2023.10386094

[7] G. Kamiya and P. Bertoldi, *Energy Consumption in Data Centres and Broadband Communication Networks in the EU*. Luxembourg: Publications Office of the European Union, European Commission Joint Research Centre, 2024. [Online]. Available: https://data.europa.eu/doi/10.2760/706491

[8] M. T. Takcı, M. Qadrdan, J. Summers, and J. Gustafsson, "Data centres as a source of flexibility for power systems," *Energy Reports*, vol. 13, pp. 3661–3671, Apr. 2025. [Online]. Available: https://doi.org/10.1016/j.egyr.2025.03.020

[9] Katal, S. Dahiya, and T. Choudhury, "Energy efficiency in cloud computing data center: A survey on hardware technologies," *Cluster Computing*, vol. 25, no. 1, pp. 1–31, Feb. 2022. [Online]. Available: https://doi.org/10.1007/s10586-021-03431-z

[10] KarmaMetrix, "What is the Internet Carbon Footprint?" KarmaMetrix Blog, Sep. 23, 2022. [Online]. Available: https://karmametrix.com/blog/websustainability/internetcarbonfootprint-data/

[11] H. Mohd Aman, E. Yadegaridehkordi, Z. S. Attarbashi, S. Hassan, and Y. Park, "A Survey on Trend and Classification of Internet of Things Reviews," IEEE Access, vol. 8, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9119087

[12] International Energy Agency (IEA), "Data centres and data transmission networks," *IEA Energy System:* Buildings, 2024. [Online]. Available: https://www.iea.org/energysystem/buildings/data-centres-and-data-transmission-networks

[13] M. Hoyos Ensuncho, "Data Centers Power Requirements," *MOST Policy Initiative*, Infrastructure and Technology, Jul. 1, 2025. [Online]. Available: https://mostpolicyinitiative.org/science-note/data-centers-power-requirements/

[14] S. Al-Sarawi, M. Anbar, N. Abdullah, and A. Al Hawari, "Internet of Things Market Analysis Forecasts, 2020–2030," in 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9210375

[15] Z. Almudayni, B. Soh, H. Samra, and A. Li, "Energy Inefficiency in IoT Networks: Causes, Impact, and a Strategic Framework for Sustainable Optimisation," *Electronics*, vol. 14, no. 1, p. 159, Jan. 2, 2025, doi: 10.3390/electronics14010159. [Online]. Available: https://www.mdpi.com/2079-9292/14/1/159

[16] A. Whiting, "Cooler data centres help take the heat off electric bills," Horizon Magazine, European Commission, Aug. 19, 2021. [Online]. Available: https://projects.researchandinnovation.ec.europa.eu/en/horizon-magazine/cooler-datacentres-help-take-heat- electricbills

[17] ARC Advisory Group, “Data Center Cooling Systems Are Required for Efficient Temperature and Environmental Management,” ARC *Advisory Group Market Analysis*, 2023. [Online]. Available: https://www.arcweb.com/market-analysis/datacentercoolingsystems

[18] E. Eslami, "Air quality and greenhouse gas emissions assessment of data centers in Texas: Quantifying impacts and environmental tradeoffs," arXiv, Sep. 2025. [Online]. Available: https://arxiv.org/abs/2509.21312

[19] M. Munier, “What is an emission factor?,” *DitchCarbon Blog*, Dec. 19, 2022. [Online]. Available: https://ditchcarbon.com/blog/what-is-an-emission-factor

[20] Our World in Data, “Carbon intensity of electricity generation, 2024,” *Our World in Data*, 2024. [Online]. Available: https://ourworldindata.org/grapher/carbon-intensity-electricity

[21] International Energy Agency, *Electricity 2025 – Emissions*. Paris: IEA, 2025. [Online]. Available: https://www.iea.org/reports/electricity-2025/emissions

[22] SBTi, *SBTi Power Sector 1.5°C Guide*. Science Based Targets initiative, 2020. [Online]. Available: https://files.sciencebasedtargets.org/production/files/SBTi-Power-Sector- 15Cguide-FINAL.pdf

[23] Fereira, R.; Ranaweera, C.; Lee, K.; Schneider, J.-G. Energy Efficient Node Selection in Edge-Fog-Cloud Layered IoT Architecture. *Sensors* **2023**, *23*(13), 6039. https://doi.org/10.3390/s23136039.

[24] K. Cao, S. Hu, Y. Shi, A. W. Colombo, S. Karnouskos, and X. Li, “A Survey on Edge and Edge-Cloud Computing Assisted Cyber-Physical Systems,” IEEE Transactions on Industrial Informatics, vol. 17, no. 11, pp. 7806–7819, 2021. [Online]. Available: https://doi.org/10.1109/TII.2021.3073066

[25] M. A. Albreem, A. M. Sheikh, M. H. Alsharif, M. Jusoh, and M. N. Mohd Yasin, “Green Internet of Things (GIoT): Applications, Practices, Awareness, and Challenges,” IEEE Access, vol. 9, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3061697

[26] H. A. Alharbi and M. Aldossary, “Energy-efficient edge-fog-cloud architecture for IoTbased smart agriculture environment,” IEEE Access, vol. 9, pp. 110480–110492, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3101397

[27] G. Purwania, I. N. S. Kumara, and M. Sudarma, "Application of IoT-based system for monitoring energy consumption," Int. J. Eng. Emerg. Technol., vol. 5, no. 2, pp. 1–10, Jul.– Dec. 2020.

[28] K. Pallikonda, A. K. Katragadda, J. R. Annam, V. V. R. Krishna, S. S. Chittineni, E. Patnala, and V. Aruna, "Optimizing Real Time IoT Processing with Hybrid Edge Cloud Architecture for Enhanced Latency and Energy Efficiency," *Journal of Theoretical and Applied Information Technology*, vol. 103, no. 12, pp. 5227–5240, Jun. 30, 2025. [Online]. Available: https://www.jatit.org/volumes/Vol103No12/16Vol103No12.pdf

[29] G. Han, G. Jia, J. Lloret, and Y. Bi, "IEEE Access Special Section Editorial: Emerging Trends, Issues, and Challenges in Energy-Efficient Cloud Computing," IEEE Access, vol. 8, pp. 108847–108856, 2020. [Online]. Available:https://doi.org/10.1109/ACCESS.2020.3001770

[30] Y. Nan, H. Wang, N. Xiong, A. V. Vasilakos, J. H. Park, and K. K. R. Choo, "Adaptive energy-aware computation offloading for cloud of things systems," IEEE Access, vol. 5, pp. 23947–23957, 2017. [Online]. Available: https://doi.org/10.1109/ACCESS.2017.2766165

[31] M. Li, Y. Li, Y. Tian, L. Jiang, and Q. Xu, "AppealNet: An efficient and highly-accurate edge/cloud collaborative architecture for DNN inference," *arXiv preprint arXiv:2105.04104*, Nov. 2021. doi: 10.48550/arXiv.2105.04104.

[32] S. Pérez, P. Arroba, and J. M. Moya, "Energy-conscious optimization of edge computing through deep reinforcement learning and two-phase immersion cooling," *Future Generation Computer* Systems, vol. 125, pp. 891–907, Dec. 2021, doi: 10.1016/j.future.2021.07.035.

[33] Elouali, H. Mora, and F. J. Mora Gimeno, "Similarity-based data transmission reduction solution for edge-cloud collaborative AI," *Proceedings of the 2022 5th Artificial Intelligence and Cloud Computing Conference (AICCC)*, 2022, pp. 42–48, doi: 10.1145/3582099.3582107

[34] F. Saeik, M. Avgeris, D. Spatharakis, and N. Santi, "Task offloading in edge and cloud computing: A survey on mathematical, artificial intelligence and control theory solutions," Comput. Networks, vol. 195, p. 108177, May 2021, doi: 10.1016/j.comnet.2021.108177.

[35] D. Dunsin, “Latency Optimization for Real-Time IoT Data Processing Across Hybrid Cloud Boundaries,” Jan. 2025. [Online]. Available: https://www.researchgate.net/publication/387971064_Latency_Optimization_for_Real-Time_IoT_Data_Processing_Across_Hybrid_Cloud_Boundaries

[36] S. F. Ahmed, S. Shawon, S. Afrin, and A. H. Gandomi, “Optimising (IoT) Performance Through Cloud, Fog and Edge Computing Architecture,” IET Wireless Sensor Systems, vol. 15, no. 1, Aug. 2025, doi: 10.1049/wss2.70016. [Online]. Available: https://www.researchgate.net/publication/394503923_Optimising_Internet_of_Things_IoT_Performance_Through_Cloud_Fog_and_Edge_Computing_Architecture

[37] M. Babiuch, P. Foltynek, and P. Smutný, “Using the ESP32 Microcontroller for Data Processing,” in *2019 20th* International *Carpathian Control Conference (ICCC)*, May 2019, pp. 1–6, doi: 10.1109/CarpathianCC.2019.8765944. [Online]. Available: https://www.researchgate.net/publication/334572715_Using_the_ESP32_Microcontroller_for_Data_Processing

[38] Katal, S. Dahiya, and T. Choudhury, "Energy efficiency in cloud computing data centers: A survey on software technologies," Cluster Computing, vol. 26, pp. 1845–1875, 2023. [Online]. Available: https://doi.org/10.1007/s10586-022-03713-0.

[39] D. Al Kez, A. M. Foley, D. Laverty, D. F. Del Rio, and B. Sovacool, "Exploring the sustainability challenges facing digitalization and internet data centers," Journal of Cleaner Production, vol. 371, p. 133633, Oct. 2022, doi:10.1016/j.jclepro.2022.133633.

[40] M. A. B. Siddik, A. Shehabi, and L. Marston, "The environmental footprint of data centers in the United States," Environmental Research Letters, vol. 16, no. 6, p. 064017, May 2021, doi: 10.1088/1748-9326/abfba1.

[41] Thangam, H. M. Haritha, R. Ramesh, R. S. Ganesh, G. Chauhan, et al., “Impact of Data Centers on Power Consumption, Climate Change, and Sustainability,” in Computational Intelligence for Green Cloud Computing and Digital Waste Management, IGI Global, USA, Mar. 2024, pp. 56–78, doi: 10.4018/979-8-3693-1552-1.ch004

[42] X. Zhang, T. Lindberg, N. Xiong, A. Mousavi, *et al.*, “Cooling Energy Consumption Investigation of Data Center IT Room with Vertical Placed Server,” *Energy Procedia*, vol. 105,

pp. 2047–2052, May 2017, doi: 10.1016/j.egypro.2017.03.581. [Online]. Available: https://www.researchgate.net/publication/317308758_Cooling_Energy_Consumption_Investigation_of_Data_Center_IT_Room_with_Vertical_Placed_Server

[43] M. Aldossary and H. A. Alharbi, "Towards a Green Approach for Minimizing Carbon Emissions in Fog-Cloud Architecture," IEEE Access, vol. 9, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3114514

[44] Izaddoost and M. Siewierski, "Energy Efficient Data Transmission in IoT Platforms," Procedia Computer Science, vol. 175, pp. 387-394, 2020. [Online]. Available: https://doi.org/10.1016/j.procs.2020.07.055.

[45] M. Rahman, R. Islam, M. M. Hassan, A. Alelaiwi, and A. Alamri, "Smart IoT data analytics for real-time water quality monitoring using edge-cloud computing," IEEE Access, vol. 8, pp. 114082–114093, 2020.

[46] Baldini, E.; Chessa, S.; Brogi, A. Estimating the Environmental Impact of Green IoT Deployments. *Sensors* **2023**, *23*(3), 1537. https://doi.org/10.3390/s23031537

[47] L. He and Y. Wang, "Node.js for Scalable Cloud Applications: A Comparative Study," Journal of Cloud Computing, vol. 11, no. 3, pp. 45–58, 2022.

[48] A. V. Romero, E. V. Laureano, R. O. J. Betancourt, and E. N. Álvarez, "An open source IoT edge-computing system for monitoring energy consumption in buildings," *Results in Engineering*, vol. 21, art. no. 101875, Mar. 2024. doi:10.1016/j.rineng.2023.101875

[49] Y.-H. Chang, F.-C. Wu, and H.-W. Lin, "Design and Implementation of ESP32-Based Edge Computing for Object Detection," *Sensors*, vol. 25, no. 6, p. 1656, Mar. 2025, doi: 10.3390/s25061656. [Online]. Available: https://www.researchgate.net/publication/389664266_Design_and_Implementation_of_ESP32-Based_Edge_Computing_for_Object_Detection

[50] P. B. Leelavinodhan, M. Vecchio, F. Antonelli, A. Maestrini, and D. Brunelli, "Design and Implementation of an Energy-Efficient Weather Station for Wind Data Collection," Sensors, vol. 21, no. 11, p. 3831, 2021. [Online]. Available: https://doi.org/10.3390/s21113831

[51] Asadov, N.; Coroamă, V.C.; Franzil, M.; Galantino, S.; Finkbeiner, M. Carbon-Aware Spatio-Temporal Workload Shifting in Edge–Cloud Environments: A Review and Novel Algorithm. *Sustainability* **2025**, *17*, 6433. https://doi.org/10.3390/su17146433

[52] Sukanya, J.; Sudharsan, B.; Methunraj, A.; Srinithesh, R. Green Task: A Carbon-Aware Scheduling Algorithm for Enhancing Energy Efficiency in Edge-Fog Computing System. *Int. J. Sci. Technol.* **2025**, *16*(3). Available online: https://www.ijsat.org/papers/2025/3/6974.pdf

[53] Texas Instruments, *INA219 Zerø-Drift, Bidirectional Current/Power Monitor With I²C Interface*, Datasheet, 2024. [Online]. Available: https://www.ti.com/lit/ds/symlink/ina219.pdf

[54] R. Mistry, S. Gaur, H. Joshi, and A. Bodara, "IoT based Smart Energy Meter Using ESP32," *JETIR — Journal of Emerging* Technologies *and Innovative Research*, vol. 10, no. 5, May 2023, pp. 27–?, ISSN 2349-5162. [Online]. Available: https://www.jetir.org/papers/JETIR2305D05.pdf

[55] P. Yadav and R. Jha, "Edge-Cloud Computing Frameworks for IoT: Challenges and Trends," IEEE Internet of Things Journal, vol. 8, no. 4, pp. 3153–3165, 2021.

[56] S. S. Ahmad, F. Almasalha, M. H. Qutqut, and M. Hijjawi, "Centralized smart energy monitoring system for legacy home appliances," *Energy Informatics*, vol. 7, art. no. 29, 22 Apr. 2024, doi:10.1186/s42162-024-00334-2.

[57] K. R. Ch, S. K. Sahoo, and F. Yanine, "Designing an intelligent smart energy monitoring system for optimizing the utilization of PV energy," *Energy Systems*, Nov. 2024, doi:10.1007/s12667-024-00703-6.

[58] Permatasari and D. Kurnianto, "Integration of a Smart Power Meter for Monitoring Household Energy Consumption in Prepaid Electrical Systems," *Jurnal Ilmu Fisika, Universitas Andalas*, vol. 16, no. 2, pp. 118-130, July 2024, doi:10.25077/jif.16.2.118130.2024.

[58a] Szymczyk, M., & Augustyniak, P. (2022). Selected Energy Consumption Aspects of Sensor Data Transmission in Distributed Multi-Microcontroller Embedded Systems. *Electronics*, 11(6), 848. https://doi.org/10.3390/electronics11060848

[59] pidusage, "pidusage: Cross-platform process CPU % and memory usage of a PID," npm, version 4.0.1, published 5 months ago. [Online]. Available: https://www.npmjs.com/package/pidusage

[60] Carbon Brief, "Analysis: UK's electricity was cleanest ever in 2024," *Carbon Brief*, 2 Jan. 2025. [Online]. Available: https://www.carbonbrief.org/analysis-ukselectricitywascleanest-ever-in-2024/

[61] Ballinger, *Analysis of* Carbon *Intensity Floor Calculations for the proposed Riverside Energy Park*, Eunomia Research & Consulting Ltd, Bristol, UK, Apr. 17, 2019. [Online]. Available: https://nsip-documents.planninginspectorate.gov.uk/published-documents/EN010093-000500-Appendix%201%20to%20GLA%20WR.pdf

[62] Y. Li and Z. Hu, "EcoServe: Designing Carbon-Aware AI Inference Systems," arXiv preprint arXiv:2502.05043, Feb. 2025, doi:10.48550/arXiv.2502.05043.

[63] R. Arshad, S. Zahoor, M. A. Shah, A. Wahid, and H. Yu, "Green IoT: An Investigation on Energy Saving Practices for 2020 and Beyond," IEEE Access, vol. 5, pp. 15667–15681, 2017. [Online]. Available: https://doi.org/10.1109/ACCESS.2017.2686092

[64] M. Babar and M. S. Khan, "ScalEdge: A framework for scalable edge computing in Internet of things–based smart systems," Int. J. Distrib. Sens. Networks, vol. 17, no. 7, p. 155014772110353, Jul. 2021. [Online]. Available: https://doi.org/10.1177/15501477211035332.

[65] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017, doi: 10.1109/MC.2017.9.

[66] International Energy Agency (IEA), "Data Centres and Data Transmission Networks," *IEA*, 2022. [Online]. Available: https://www.iea.org/energy-system/buildings/datacentresanddata-transmission-networks

[67] Varghese and R. Buyya, "Revisiting the Arguments for Edge Computing Research," *IEEE Internet* Computing, vol. 25, no. 5, pp. 36–43, Sept.–Oct. 2021, doi: 10.1109/MIC.2021.3103991.

[68] Andrae, A. S. G., & Edler, T. (2015). On global electricity usage of communication technology: Trends to 2030. *Challenges*, 6(1), 117–157. https://doi.org/10.3390/challe6010117

[69] M. Clement, "Optimizing Smart Grid Infrastructure through AI and IoT for Enhanced Renewable Energy Utilization," *ResearchGate*, Apr. 2025. [Online]. Available: https://www.researchgate.net/publication/390746394_Optimizing_Smart_Grid_Infrastructure_through_AI_and_IoT_for_Enhanced_Renewable_Energy_Utilization?

[70] J. Sammu, "Latency Reduction Techniques for IoT Applications," *ResearchGate*, Dec. 2018. [Online]. Available: https://www.researchgate.net/publication/388038469_Latency_Reduction_Techniques_for_IoT_Applications?

9. Appendix

Link to GitHub Repository:
https://github.com/serma123/WeatherMonitoring