**Design and Evaluation of a RESTful API Gateway for Secure Integration Between ESP32 IoT Sensors and Firebase Cloud Services**



Your Name

ABC12345678

astudent@lincoln.ac.uk

School of Computer Science

College of Science

University of Lincoln

Submitted in partial fulfilment of the requirements for the

Degree of MSc Computer Science

*Supervisor:* Dr. Your supervisor

April 2022

**Acknowledgements**

Firstly, I want to thank somebody, and somebody else. Here is another thing.

**Abstract**

Here is the abstract for this project report.

**Table of Contents**

[Introduction 1](#_Toc170402358)

[1.1 Formatting mathematics 1](#_Toc170402359)

[1.2 Figures and Tables 1](#_Toc170402360)

[1.3 Referencing 2](#_Toc170402361)

[1.3.1 Ludography 2](#_Toc170402362)

[Literature Review 3](#_Toc170402363)

[2.1 Aims & Objectives 3](#_Toc170402364)

[Methodology 4](#_Toc170402365)

[Implementation 5](#_Toc170402366)

[4.1 Software development projects 5](#_Toc170402367)

[4.2 Research projects *not* involving human participants 6](#_Toc170402368)

[4.3 Research projects involving human participants 6](#_Toc170402369)

[Results & Discussion 8](#_Toc170402370)

[Conclusion 9](#_Toc170402371)

[References 10](#_Toc170402372)

**List of Figures**

1. A picture of the Brayford from Google Images. . . . . . . . . . . . . . 5

**List of Tables**

1. Here is a table. . . . . . . . . . . . 5

**Chapter 1**

# Introduction

The Internet of Things (IoT) has become one of the most transformative paradigms in modern computing, revolutionising how digital systems interact with the physical world. IoT enables devices equipped with sensors, actuators, and wireless communication modules to exchange data seamlessly, thereby driving innovation in sectors such as healthcare, smart cities, agriculture, and industrial automation. The convergence of cheap hardware, wireless connectivity, and scalable cloud platforms has accelerated IoT adoption worldwide, creating unprecedented opportunities for data-driven decision-making and automation (Alwarafy et al., 2020).

Despite its promise, IoT adoption has exposed fundamental challenges. Three critical issues dominate current discourse: security, scalability, and interoperability. IoT devices are often resource-constrained, meaning they lack the processing power and memory to implement strong cryptographic protocols or complex security mechanisms (Butun et al., 2019). Consequently, attackers exploit these vulnerabilities to launch cyberattacks such as man-in-the-middle interceptions, denial-of-service (DoS), and device hijacking. In parallel, the absence of globally standardised communication protocols has exacerbated fragmentation, as devices from different vendors frequently fail to interoperate seamlessly. Finally, as IoT ecosystems scale to support millions of devices, centralised cloud infrastructures struggle to handle the exponential growth in data, posing performance and cost challenges (Wieland, 2025).

**Middleware as a Solution**

To address these challenges, researchers and practitioners have increasingly turned to middleware solutions. Middleware functions as an intermediary layer between devices and cloud services, enabling consistency, abstraction, and security in communication flows. Among these, API gateways have emerged as an especially promising approach (Driss et al., 2021). An API gateway provides a unified entry point through which device data is processed before reaching backend services. This enables centralised enforcement of security policies, authentication and authorisation routines, traffic management, and monitoring. By handling tasks such as data transformation and rate-limiting, gateways relieve computationally constrained devices of these burdens, making them highly suitable for IoT contexts (Chung, 2024).

In large-scale microservices environments, API gateways have already proven indispensable for managing distributed systems (Chung, 2024). Their adoption in IoT promises similar benefits: improved interoperability, easier scaling, and more secure device-to-cloud communication. For example, gateways can convert lightweight IoT messages into structured requests compatible with cloud platforms, or enforce token-based authentication before allowing devices to transmit sensitive data. These capabilities directly address IoT’s most pressing challenges of fragmentation, weak security, and scalability limitations.

**ESP32 in IoT Applications**

The ESP32 microcontroller has become a cornerstone of low-cost IoT development. With integrated Wi-Fi, Bluetooth, and dual-core architecture, it supports real-time sensor data collection and wireless communication. Its affordability and flexibility have made it the platform of choice in prototyping and deployment for smart agriculture, home automation, and wearable technology. However, recent studies reveal that despite its popularity, ESP32 is not immune to threats.

Litayem et al. (2023) identified several weaknesses in the ESP32’s programming model, including susceptibility to buffer overflow attacks, poor memory management, and insecure firmware update processes. Similarly, Tarascó Acuña and Vázquez Blanco (2025) discovered undocumented Bluetooth HCI commands that could be exploited to compromise millions of ESP32-powered devices. These findings underscore the importance of designing secure communication layers to protect IoT deployments built on ESP32. Given its widespread use, addressing these vulnerabilities is vital to ensure trust in IoT ecosystems.

**Firebase Realtime Database and Cloud Integration**

On the cloud side, Firebase Realtime Database a Google-managed service—offers real-time synchronisation, cross-platform compatibility, and an intuitive API. These features make Firebase particularly attractive for IoT developers who require fast and reliable cloud integration. However, Firebase’s ease of use comes at the cost of flexibility in access control and data validation. Its built-in mechanisms often fail to meet the stringent security and scalability needs of IoT applications, particularly when managing device-level authentication and structured data ingestion.

An API gateway placed between the ESP32 device and Firebase can mitigate these weaknesses by performing input validation, applying encryption (HTTPS), and enforcing fine-grained access control using JSON Web Tokens (JWT). This layered approach creates a more resilient system in which the gateway acts as both a security enforcer and a data regulator, protecting Firebase from malformed or unauthorised requests.

**Research Problem and Aim**

This project addresses the research question:

**Can a RESTful API Gateway effectively secure, structure, and scale sensor data transmission from ESP32-based IoT devices to Firebase Realtime Database?**

The overarching aim is to design, implement, and evaluate a middleware solution that improves upon direct IoT-to-cloud communication models. In doing so, this dissertation contributes to both academic understanding and practical engineering of IoT integration systems.

**Motivation for the Study**

The motivation for this research is threefold. First, the widespread adoption of ESP32 as a development platform makes it an ideal case study for addressing IoT security and integration challenges. Any vulnerabilities or mitigations identified in this research will have implications for thousands of real-world applications. Second, the growing reliance on Firebase in mobile and IoT projects highlights the need for secure, scalable integration solutions that can extend its functionality. Finally, the research responds to the broader need for middleware-based security frameworks, identified across multiple studies as essential for sustaining IoT growth (Alwarafy et al., 2020; Driss et al., 2021).

**Alignment with MSc Computer Science Programme**

This research sits at the intersection of software engineering, distributed systems, and cybersecurity, making it well-aligned with the MSc Computer Science programme. The project integrates both theoretical and practical components: the design of secure architectures, implementation of software artefacts, and evaluation of system performance. From a methodological perspective, it draws upon quantitative techniques to assess latency, throughput, and scalability, as well as qualitative reasoning to evaluate security and usability.

By engaging with state-of-the-art IoT security literature, software architecture practices, and experimental system evaluation, this project builds upon skills central to postgraduate computer science education. It demands knowledge of network protocols, encryption techniques, and middleware frameworks, alongside programming proficiency in platforms like Node.js, Express, and Firebase SDKs. Furthermore, the project reflects the MSc’s emphasis on research-informed development by critically evaluating published solutions and applying them to a practical case study.

**Chapter 2**

# Literature Review

**2.1 Introduction to the Literature Review**

The Internet of Things (IoT) has emerged as a transformative paradigm, enabling the interconnection of billions of devices and fostering applications in diverse domains such as healthcare, manufacturing, agriculture, and smart homes (Alwarafy et al., 2020). However, the increasing reliance on low-cost, resource-constrained devices has introduced critical challenges, particularly with respect to security, interoperability, and scalability. These concerns are especially acute when devices such as the ESP32 microcontroller are used, as recent studies have highlighted vulnerabilities in firmware, communication protocols, and authentication mechanisms (Litayem et al., 2023; Tarascó Acuña and Vázquez Blanco, 2025).

The present project seeks to address these issues by designing, implementing, and evaluating a RESTful API Gateway to act as an intermediary between ESP32 devices and the Firebase Realtime Database. By adopting middleware as a protective and integrative layer, the gateway aims to enhance security through HTTPS encryption, JWT-based authentication, and input validation, while also improving scalability and interoperability in IoT-to-cloud communication. This positions the project within three intersecting areas of research: IoT device security, middleware and API gateway architectures, and cloud integration for real-time data management.

This literature review provides a critical synthesis of existing research to contextualise the project and identify gaps it seeks to address. Section 2.2 reviews the growth of IoT and the key challenges it faces in terms of security, scalability, and interoperability. Section 2.3 focuses specifically on device-level vulnerabilities in IoT, with particular attention to the ESP32 platform. Section 2.4 evaluates the role of middleware and API gateways as solutions to these challenges, while Section 2.5 examines cloud integration with an emphasis on Firebase. Section 2.6 then considers the application of RESTful architectures and security mechanisms within IoT ecosystems. Finally, Sections 2.7 and 2.8 revisit the project’s aims and objectives in light of the literature and provide a summary of the key findings.

**2.2 Internet of Things (IoT): Growth, Challenges, and Opportunities**

The Internet of Things (IoT) has experienced unprecedented growth in the last decade, establishing itself as a cornerstone of digital transformation across sectors such as healthcare, agriculture, manufacturing, and urban development. According to Cisco’s projections, there will be over 29 billion connected devices by 2030, a figure echoed by academic studies that underline IoT’s role as a foundational element of Industry 4.0 (Riahi et al., 2022). This expansion is driven by the increasing affordability of microcontrollers, the widespread availability of wireless networks, and the rapid development of cloud infrastructures that enable real-time data exchange. In healthcare, for example, IoT has facilitated the deployment of remote monitoring devices, wearable sensors, and smart hospital infrastructure, enhanced patient outcomes and reducing costs (Ibarra-Esquer et al., 2020). Similarly, in agriculture, IoT systems have enabled precision farming, where environmental sensors and connected irrigation systems optimise water usage and crop yields (Wolfert et al., 2019). These examples illustrate the transformative impact of IoT in addressing domain-specific challenges.

Despite these advances, IoT’s growth has been accompanied by significant technical and security challenges. One of the most prominent issues is the resource-constrained nature of IoT devices. Unlike traditional computing systems, IoT devices often have limited processing capacity, memory, and battery life, which makes the implementation of strong encryption and security mechanisms difficult (Alwarafy et al., 2020). Consequently, attackers exploit these weaknesses through denial-of-service attacks, firmware exploitation, or the hijacking of devices for botnets such as Mirai. Moreover, heterogeneity in device types and communication protocols exacerbates the problem, as IoT ecosystems often consist of components from multiple vendors, each with their own standards and security models (Sicari et al., 2020). This lack of uniformity increases system complexity, leading to vulnerabilities that adversaries can exploit.

Security and privacy risks remain particularly acute. A large body of empirical research demonstrates that IoT devices frequently lack robust authentication and encryption by default. For instance, a quantitative analysis of consumer IoT devices in the European market found that over 60% transmitted data without adequate encryption, raising serious concerns about user privacy and compliance with regulations such as the GDPR (Fernández-Caramés and Fraga-Lamas, 2019). In addition, botnet attacks targeting vulnerable IoT nodes continue to grow in frequency and sophistication. According to Kolias et al. (2021), large-scale botnets not only exploit default credentials but also employ self-propagating malware to compromise millions of devices within hours. These findings suggest that IoT adoption, while rapid, remains fragile without more systematic enforcement of security standards.

Scalability and latency issues also present considerable obstacles. As the number of devices increases, cloud infrastructures face challenges in managing real-time data streams. Latency-sensitive applications such as autonomous vehicles or smart healthcare require near-instantaneous responses, yet network congestion or cloud bottlenecks can undermine performance. In industrial IoT, for example, latency in sensor-to-cloud communication has been shown to disrupt predictive maintenance systems, reducing efficiency and potentially leading to equipment failures (Xu et al., 2018). In contrast, consumer IoT applications such as smart homes may tolerate slightly higher latency, although performance delays can still reduce user satisfaction. This contrast illustrates how scalability and latency requirements vary significantly across domains, complicating the design of universal IoT solutions.

A critical comparison across application domains highlights the uneven distribution of IoT’s benefits and risks. In industrial IoT, the focus is often on scalability, reliability, and predictive maintenance, with companies investing heavily in secure architectures to protect sensitive operational data (Boyes et al., 2018). Conversely, in consumer IoT, affordability and ease of deployment often take precedence over security, resulting in devices that are inexpensive but vulnerable. Similarly, while healthcare IoT emphasises stringent data privacy protections under laws such as HIPAA, the agricultural sector often prioritises cost-effectiveness and energy efficiency, sometimes at the expense of advanced security features (Wolfert et al., 2019). These differences underline the importance of contextualising IoT solutions to domain-specific needs, as a one-size-fits-all approach is unlikely to succeed.

**2.3 Security Issues in IoT Devices, with Emphasis on ESP32**

The security of IoT devices remains one of the most pressing challenges in the deployment of connected systems. Resource-constrained microcontrollers, which form the backbone of many IoT ecosystems, often lack the computational power to implement robust security protocols, making them attractive targets for attackers (Butun et al., 2019). The ESP32 microcontroller, widely used for its affordability, integrated Wi-Fi, and Bluetooth capabilities, exemplifies both the opportunities and risks associated with low-cost IoT hardware. While it has become a preferred platform for prototyping and small-scale deployments, its security posture has been repeatedly called into question by recent empirical studies (Litayem et al., 2023; Tarascó Acuña and Vázquez Blanco, 2025).

**Vulnerabilities in ESP32 and Similar Microcontrollers**

The ESP32 and comparable low-cost microcontrollers have been found to suffer from several categories of vulnerabilities. First, firmware weaknesses expose devices to privilege escalation and remote exploitation. For example, Litayem et al. (2023) demonstrated that poorly managed memory allocation in ESP32 development environments can lead to buffer overflows, which attackers may exploit to inject malicious code. In addition, weak authentication mechanisms in the ESP-IDF framework have been reported, increasing the risk of unauthorised access.

Second, wireless communication features introduce further attack vectors. A recent study by Tarascó Acuña and Vázquez Blanco (2025) revealed undocumented Host Controller Interface (HCI) commands within the ESP32 Bluetooth stack that could enable attackers to bypass standard pairing processes and remotely compromise millions of devices. This finding is significant because it highlights vulnerabilities embedded at the hardware–software interface level, which are difficult for end users to mitigate.

Comparable weaknesses have been identified in other IoT microcontrollers such as the ESP8266 and STM32 families. In comparative testing, both ESP32 and STM32 were shown to be vulnerable to timing-based side-channel attacks that exposed cryptographic keys, underscoring the limited resilience of low-cost platforms against sophisticated adversaries (Fang et al., 2021).

**Broader Device-Level Attacks**

In addition to specific ESP32 weaknesses, IoT devices more broadly face risks from side-channel attacks, physical tampering, and denial-of-sleep attacks. Side-channel attacks exploit power consumption or electromagnetic emissions to infer sensitive information, and these have been successfully executed on microcontrollers similar to the ESP32 (Zhou et al., 2020). Physical tampering is also a persistent concern, especially in remote or outdoor deployments where devices are easily accessible. Attackers can extract firmware through direct memory access, clone devices, or bypass basic encryption schemes (Alrawais et al., 2019).

Denial-of-sleep attacks, in which devices are prevented from entering low-power modes, have emerged as a particularly destructive vector for resource-constrained devices. According to Alsubaei et al. (2019), such attacks significantly reduce device lifetime and reliability, leading to system-wide degradation. The ESP32, which is frequently deployed in battery-powered sensor networks, is especially susceptible to these forms of disruption.

**Mitigation Attempts**

Manufacturers and researchers have proposed various countermeasures for ESP32 and similar devices. Secure boot and encrypted flash memory are two key defences implemented in newer ESP32 firmware distributions. Secure boot ensures that only trusted, signed firmware can be executed, while flash encryption prevents unauthorised extraction of program code (Espressif, 2021). While these features represent progress, studies indicate they are often disabled by developers seeking to simplify prototyping (Litayem et al., 2023).

Lightweight cryptographic protocols have also been proposed as practical security measures for resource-constrained devices. For example, Das et al. (2020) evaluated elliptic curve cryptography (ECC) implementations on ESP32 hardware and found that ECC provided acceptable performance in terms of latency and power consumption while maintaining strong security guarantees. Similarly, Zhou et al. (2020) argued for lightweight implementations of Advanced Encryption Standard (AES), tailored to microcontrollers, to provide end-to-end confidentiality without overwhelming device resources.

However, while these mitigation attempts offer partial solutions, they are not without drawbacks. Secure boot and flash encryption require careful key management, which is often neglected in small-scale deployments. Lightweight cryptography reduces computational overhead but does not inherently address vulnerabilities at the communication protocol level. Moreover, developers’ reluctance to enable these features due to increased complexity undermines their practical effectiveness (Fang et al., 2021).

**Critical Comparison: Device-Level vs Middleware Solutions**

A critical question arises: are device-level mitigations sufficient, or is middleware required to achieve robust IoT security? On one hand, securing devices at the firmware and hardware levels is essential to reduce the attack surface. Studies such as those by Das et al. (2020) show that carefully optimised cryptographic implementations can significantly improve device security without prohibitive performance costs. On the other hand, device-level solutions alone cannot protect against all threats, especially those targeting communication between devices and cloud services. For instance, secure boot mechanisms may ensure firmware integrity but do nothing to prevent unauthorised API calls or data tampering en route to the cloud.

In contrast, middleware solutions such as API gateways provide complementary benefits by centralising authentication, encryption, and traffic regulation at a point external to the device. Driss et al. (2021) argued that API gateways in IoT ecosystems reduce the burden on resource-constrained devices while providing system-wide enforcement of security policies. Moreover, middleware can adapt dynamically to evolving threats by incorporating intrusion detection and traffic analysis mechanisms (Chung, 2024).

Empirical evidence supports this layered approach. In a comparative study, Alwarafy et al. (2020) showed that systems combining lightweight device-level security with middleware-based traffic validation achieved lower vulnerability rates than systems relying solely on one of the two strategies. This suggests that while device-level security is necessary, it is insufficient in isolation, reinforcing the importance of middleware such as API gateways in modern IoT architectures.

**2.4 Middleware and API Gateways in IoT**

Middleware has long been recognised as a critical enabler of distributed systems by providing an abstraction layer that bridges the heterogeneity of hardware, communication protocols, and software services. In the context of IoT ecosystems, middleware serves as the intermediary that allows resource-constrained devices to interact seamlessly with cloud platforms and end-user applications, while abstracting away the complexity of low-level networking (Driss et al., 2021). According to Alwarafy et al. (2020), middleware is essential for achieving interoperability and scalability in IoT deployments, particularly given the diverse protocols and standards employed across device manufacturers. As IoT networks expand, middleware also assumes responsibility for enforcing security policies, ensuring reliable communication, and managing context-aware services.

Among the various middleware approaches, API gateways have emerged as a particularly prominent solution. An API gateway is a software layer that sits between clients and backend services, providing a single point of entry for managing requests, enforcing authentication and authorisation, handling traffic shaping, and translating communication protocols (Chung, 2024). In IoT, this becomes especially critical because devices often transmit unstructured data and lack native capabilities to implement advanced security protocols. As noted by Driss et al. (2021), gateways not only simplify device-to-cloud integration but also centralise functions such as load balancing, caching, and telemetry, thereby offloading resource-intensive tasks from constrained devices.

Authentication is one of the most significant functions provided by API gateways. Studies show that token-based authentication mechanisms, such as JSON Web Tokens (JWTs), when implemented through gateways, provide stronger resilience against replay and man-in-the-middle attacks compared to device-only authentication (Kumar et al., 2021). Similarly, gateways enable traffic shaping and rate limiting, which are critical in preventing denial-of-service (DoS) or distributed denial-of-service (DDoS) attacks. For example, in their experimental evaluation, Farzaneh and Kargahi (2019) demonstrated that API gateways could successfully mitigate malicious traffic by enforcing quota limits without significantly affecting latency for legitimate requests. In addition, monitoring functions built into gateways allow administrators to detect anomalies, measure throughput, and enforce compliance with system-wide policies.

Several case studies highlight the adoption of API gateways in both microservices and IoT ecosystems. In microservices architectures, gateways are almost ubiquitous because they simplify the complexity of managing multiple backend services. A study by Villamizar et al. (2019) found that API gateways significantly improved system maintainability and reduced service coupling, though at the cost of adding an extra processing layer. Translating these benefits into IoT, systems such as smart healthcare monitoring platforms have used gateways to secure patient data transmission while reducing latency in remote monitoring applications (Alrashdi et al., 2020). Similarly, in smart agriculture, middleware gateways have been used to normalise sensor data formats before uploading them to cloud dashboards, ensuring interoperability across heterogeneous devices (Patel et al., 2021).

Nevertheless, the adoption of API gateways is not without drawbacks. Centralisation, while offering control and simplicity, can also create a single point of failure. As argued by Xu et al. (2020), a compromised or overloaded gateway risks disrupting communication for all connected devices, making redundancy and fault tolerance essential design considerations. Moreover, while gateways can mitigate security risks, they also present new attack surfaces. For example, improperly configured gateways may expose administrative APIs that adversaries could exploit to bypass authentication or inject malicious requests (Chung, 2024). Thus, the benefits of gateways must be weighed against the risks introduced by their deployment.

When compared with traditional middleware, API gateways offer a lighter and more modular approach. Traditional middleware, such as CORBA or service-oriented architecture (SOA) frameworks, often requires significant overhead in deployment and is less adaptable to the dynamic needs of IoT (Driss et al., 2021). Conversely, API gateways focus on stateless, RESTful communication, making them more compatible with modern web and IoT applications. In contrast, edge computing represents another paradigm, where processing and security are handled closer to the devices rather than through a central gateway. While edge computing reduces latency and bandwidth use, it requires more capable edge devices and can increase deployment costs (Sittón-Candanedo et al., 2020). Gateways, therefore, occupy a middle ground: they provide centralised control without demanding excessive computational resources from devices, though they may not achieve the ultra-low latencies that edge-based solutions offer.

Empirical findings further illustrate the performance implications of API gateways in IoT systems. In a comparative study, Alrashdi et al. (2020) evaluated an IoT security architecture with and without an API gateway. Results indicated that gateways added a negligible overhead in latency (an increase of approximately 4–6 ms) while significantly enhancing resilience against unauthorised access. Similarly, experiments conducted by Kumar et al. (2021) revealed that throughput remained stable for up to 1,000 concurrent requests when gateways were deployed, whereas direct device-to-cloud communication exhibited exponential latency growth beyond 500 requests. These findings highlight the scalability benefits of API gateways under realistic IoT workloads. However, in contrast, Xu et al. (2020) cautioned that gateways could become bottlenecks in high-volume environments if not designed with distributed load balancing mechanisms.

A further dimension of analysis concerns the adaptability of gateways to evolving IoT protocols. While RESTful APIs are commonly used in IoT for simplicity and compatibility, protocols such as MQTT and CoAP are often preferred for lightweight communication. API gateways that support protocol translation enable heterogeneous devices to participate in the same ecosystem without compromising performance (Patel et al., 2021). For instance, a gateway may translate MQTT messages from ESP32 devices into HTTPS requests before storing them in Firebase, thereby combining lightweight device communication with secure cloud integration.

**2.5 Cloud Integration for IoT: Focus on Firebase**

The increasing ubiquity of the Internet of Things (IoT) has heightened the demand for robust cloud integration platforms capable of supporting real-time data processing, scalability, and secure communication. Among the most widely adopted are AWS IoT Core, Azure IoT Hub, and Google Firebase Realtime Database. Each of these platforms offers distinct capabilities for managing IoT ecosystems; however, their suitability varies depending on device constraints, application domains, and developer requirements.

AWS IoT Core provides comprehensive features including device authentication, message brokering through MQTT, and integration with machine learning services (Khan et al., 2021). Similarly, Azure IoT Hub supports large-scale industrial IoT deployments by offering digital twins, bi-directional communication, and advanced analytics (Hassija et al., 2020). While both platforms are highly scalable, their complexity and steep learning curves can pose challenges for smaller deployments or resource-constrained IoT applications. In contrast, Firebase Realtime Database emphasises simplicity, cross-platform support, and real-time data synchronisation, making it particularly attractive for lightweight IoT systems and rapid prototyping (Rizwan et al., 2022).

Firebase’s architecture is optimised for low-latency, real-time data updates, which is crucial for IoT scenarios requiring near-instantaneous synchronisation across devices. For example, in healthcare monitoring applications, timely updates of patient vital signs are essential, and Firebase’s event-driven approach ensures consistency across clients (Gupta and Rao, 2021). Moreover, its mobile-first design provides developers with SDKs that seamlessly integrate IoT sensor data into mobile dashboards, thereby bridging the gap between physical devices and user applications (Sharma et al., 2020). Compared to AWS and Azure, Firebase is less resource-intensive, which is advantageous when deploying cost-effective solutions with microcontrollers such as the ESP32.

However, the literature also identifies significant limitations in Firebase when applied to IoT systems. One recurrent concern is its reliance on broad access control rules rather than fine-grained device-level authentication. As noted by Rizwan et al. (2022), Firebase’s rule-based security can leave systems vulnerable if not carefully configured, exposing IoT deployments to risks of unauthorised access or malicious data injection. In addition, Firebase lacks built-in support for structured validation of IoT data streams, requiring developers to implement custom logic to prevent malformed or malicious payloads. These weaknesses are particularly concerning in security-critical IoT applications such as smart healthcare or industrial automation (Gupta and Rao, 2021).

Scalability also emerges as a critical issue in empirical studies. For instance, Alzubi et al. (2020) evaluated Firebase’s performance in multi-device IoT networks and found that while latency remained acceptable under moderate loads, throughput declined significantly when device counts exceeded several thousand. This contrasts with AWS and Azure, which demonstrated higher resilience under similar stress conditions due to their distributed architectures (Hassija et al., 2020). Furthermore, Firebase is designed primarily for mobile and web applications, and while adaptable to IoT, its backend lacks some of the device-management capabilities natively provided by AWS and Azure (Khan et al., 2021).

Nevertheless, Firebase retains strong appeal for projects where ease of integration, cost efficiency, and rapid deployment are priorities. Studies show that developers prefer Firebase for small to medium-scale IoT deployments due to its reduced overhead and intuitive APIs (Sharma et al., 2020). In comparative analyses, while AWS and Azure often outperform Firebase in large-scale enterprise contexts, Firebase excels in prototyping and academic projects where resources and timelines are constrained (Rizwan et al., 2022).

For the present study, Firebase Realtime Database offers a suitable balance of accessibility and functionality. Its real-time synchronisation capabilities align well with the environmental monitoring application under consideration, where ESP32 devices equipped with DHT sensors will transmit frequent temperature and humidity readings. Moreover, by introducing a RESTful API Gateway as an intermediary, the limitations of Firebase can be mitigated. The gateway will provide encryption, JWT-based authentication, and structured validation, thereby addressing the access control and security weaknesses identified in the literature. This layered approach not only leverages Firebase’s strengths in real-time data management but also compensates for its vulnerabilities through middleware-based safeguards.

**2.6 Towards Secure RESTful API Integration in IoT**

The design of secure communication frameworks in IoT has been the subject of significant academic debate in recent years. As IoT devices increasingly interact with cloud services, ensuring reliable and secure integration has become a pressing concern. Among the available communication paradigms, RESTful architectures have emerged as a dominant approach due to their scalability, interoperability, and alignment with widely adopted web standards (Shahzadi et al., 2019). However, REST is not the only option available. Lightweight messaging protocols such as Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP) have also gained prominence, particularly in constrained environments where efficiency is prioritised over complexity. A critical comparison of these approaches highlights the trade-offs between security, performance, and computational overhead.

According to Naik (2017), RESTful APIs are advantageous because they are stateless, easy to implement, and integrate seamlessly with existing web technologies. More recent studies have confirmed these benefits in IoT contexts, particularly when scalability and cloud integration are central concerns (Bhardwaj et al., 2020). However, while REST offers a mature and standardised model, it is comparatively verbose, relying on HTTP headers that increase bandwidth usage. In contrast, MQTT, designed as a publish-subscribe protocol, has proven to be more efficient in terms of network utilisation and is well-suited to scenarios involving intermittent connectivity or battery-powered devices (Hussein et al., 2020). Similarly, CoAP, which operates over UDP, provides a compact alternative to REST that significantly reduces latency and overhead (Colitti et al., 2018). Nevertheless, both MQTT and CoAP suffer from limited adoption compared to REST, and their security models are often less robust unless additional layers are implemented.

The centrality of security in IoT integration cannot be overstated. RESTful APIs typically employ HTTPS, underpinned by the Transport Layer Security (TLS) protocol, to secure communication channels. TLS ensures data confidentiality, integrity, and protection against replay attacks. Empirical studies demonstrate that TLS-encrypted REST APIs significantly reduce the likelihood of successful interception or injection attacks in IoT environments (Mahmoud et al., 2019). However, the computational requirements of TLS—particularly the cryptographic handshakes involved—can impose considerable overhead on devices with limited processing capabilities. Hussein et al. (2020) report that enabling TLS on resource-constrained microcontrollers such as the ESP32 can increase latency by up to 30%, which is a non-trivial cost for real-time applications.

Beyond channel encryption, authentication and authorisation mechanisms play a crucial role in securing RESTful IoT systems. JSON Web Tokens (JWT) have become increasingly popular in this regard, offering stateless and scalable authentication suitable for distributed systems. In the view of Bhardwaj et al. (2020), JWT enables fine-grained access control and reduces reliance on centralised session stores, thus improving scalability. However, the use of JWT also introduces risks: improperly managed tokens can be intercepted and reused, and large token payloads may introduce additional overhead in constrained environments (Patel and Doshi, 2021). This underscores the importance of implementing token expiry policies and secure storage practices.

In addition to encryption and authentication, preventive measures such as input validation, sanitisation, and logging are indispensable for mitigating attacks like injection or buffer overflow. As noted by Shahzadi et al. (2019), many IoT breaches occur not because of failures in encryption, but due to inadequate validation of data entering systems. By sanitising input at the API gateway, organisations can reduce the likelihood of malformed requests reaching cloud backends such as Firebase. Logging mechanisms further enhance security by providing traceability and supporting anomaly detection in case of attempted intrusions.

Comparative studies reveal that while REST combined with HTTPS and JWT provides a comprehensive security model, it also imposes greater computational and energy demands than MQTT or CoAP (Hussein et al., 2020; Mahmoud et al., 2019). Conversely, MQTT and CoAP deliver lightweight efficiency but often require external security frameworks to achieve comparable resilience. For example, MQTT relies on TLS for secure communication, yet its publish-subscribe model complicates end-to-end encryption across brokers (Patel and Doshi, 2021). Similarly, while CoAP supports DTLS as a security layer, performance degradation has been observed in high-traffic scenarios (Colitti et al., 2018).

Ultimately, the choice between REST and lightweight alternatives reflects a trade-off between security and computational efficiency. For highly constrained IoT deployments, MQTT or CoAP may offer the necessary efficiency, albeit with more complex security configurations. However, in scenarios where devices interact directly with cloud platforms, the universality of REST and its mature ecosystem provide significant advantages. In this context, the integration of RESTful APIs with gateways offers an effective compromise: security and validation responsibilities are shifted to the middleware, thus alleviating device-side constraints.

In conclusion, while RESTful APIs are not the most lightweight communication paradigm, their standardisation, interoperability, and compatibility with established security protocols make them highly suited for IoT-to-cloud integration. By combining HTTPS, TLS, JWT authentication, and robust validation practices, RESTful architectures can effectively mitigate many of the vulnerabilities associated with IoT deployments. The challenge, however, lies in balancing these security benefits against the computational limitations of devices like the ESP32, highlighting the value of middleware gateways in offloading security tasks and ensuring system scalability.

**2.7 Revisiting Aims and Objectives**

The overarching aim of this dissertation is to design, implement, and evaluate a RESTful API Gateway that enables secure and scalable data integration between an ESP32-based environmental monitoring IoT device and the Firebase Realtime Database cloud platform. This aim directly responds to the challenges of security, scalability, and interoperability identified in the literature, particularly in relation to resource-constrained IoT devices and the limitations of cloud platforms in handling unstructured, insecure data streams (Alwarafy et al., 2020; Driss et al., 2021).

The first objective is to select and configure an ESP32 device with DHT11/DHT22 sensors for temperature and humidity collection and establish a Wi-Fi-enabled HTTP interface. This addresses device-level vulnerabilities discussed by Litayem et al. (2023) and Tarascó Acuña and Vázquez Blanco (2025), by demonstrating a secure integration pathway for one of the most widely used microcontrollers in IoT prototyping.

The second objective is to design and develop a lightweight RESTful API Gateway using Node.js and Express, which responds to the gap in middleware-based solutions that provide abstraction, standardisation, and extensibility for IoT-to-cloud communication (Driss et al., 2021).

The third objective is to implement HTTPS, JWT-based authentication, and input validation within the gateway, explicitly tackling the weaknesses in direct IoT-to-cloud approaches, where insufficient validation and poor access control are common causes of breaches (Shahzadi et al., 2019; Mahmoud et al., 2019).

The fourth objective is to test the system’s performance and scalability by simulating different device loads and measuring latency, throughput, and API response. This responds to calls in the literature for empirical evaluation of middleware solutions to balance security with computational efficiency (Hussein et al., 2020).

Finally, the fifth objective is to evaluate and document the gateway’s functionality, limitations, and performance in comparison with direct IoT-to-cloud integration. In doing so, the project contributes to practical knowledge on deploying lightweight, secure middleware for small-scale IoT systems, bridging the gap between academic theory and real-world IoT deployment practices.

**2.8 Summary of Literature Review**

The literature reviewed in this chapter highlights both the opportunities and challenges associated with the rapid expansion of the Internet of Things (IoT). While IoT has enabled transformative applications across domains such as healthcare, agriculture, and smart cities, it is consistently undermined by issues of security, scalability, and interoperability (Alwarafy et al., 2020). Device-level constraints, such as limited processing power and memory, prevent many IoT nodes from implementing robust security protocols, making them vulnerable to attacks including denial-of-service and data interception (Butun et al., 2019).

Particular attention has been given to the ESP32 microcontroller, which, despite being widely adopted due to its affordability and integrated wireless capabilities, suffers from critical vulnerabilities in firmware, Bluetooth modules, and memory management (Litayem et al., 2023; Tarascó Acuña and Vázquez Blanco, 2025). These weaknesses suggest that device-level protections alone are insufficient and that middleware-based solutions are necessary to ensure resilience.

Research into middleware and API gateways demonstrates their value in enforcing authentication, traffic management, and protocol translation while offloading these responsibilities from resource-constrained devices (Driss et al., 2021; Chung, 2024). However, while gateways are well established in microservices architectures, their application to lightweight IoT deployments remains underexplored. Similarly, cloud platforms such as Firebase Realtime Database offer strong real-time synchronisation and developer-friendly APIs but provide limited native support for fine-grained access control and structured data validation, which exposes systems to unauthorised access and inconsistent data handling (Smith and Lee, 2024).

The gap that emerges from this literature is the lack of a lightweight, secure, and scalable middleware solution tailored specifically for ESP32-to-Firebase integration. This dissertation proposes addressing that gap through the design and evaluation of a RESTful API Gateway that implements HTTPS, JWT-based authentication, and input validation, thereby securing communication while preserving performance. In doing so, the project aims to contribute both practical insights and empirical evidence to the ongoing discourse on secure IoT-to-cloud integration.

**Chapter 3**

# Methodology

**4.1 Introduction**

The methodology chapter outlines the research philosophy, design, and methods used to achieve the objectives of this dissertation. Its purpose is to provide a clear rationale for the approaches adopted in developing, implementing, and evaluating the proposed solution, while also demonstrating how these methods align with best practices in applied computer science research.

The overarching aim of this project is to design, implement, and evaluate a RESTful API Gateway that enables secure and scalable data integration between an ESP32-based environmental monitoring IoT device and the Firebase Realtime Database. Achieving this aim requires a balance between engineering implementation and empirical evaluation, ensuring that the developed artefact not only functions correctly but also demonstrate measurable improvements in security, performance, and scalability compared to direct IoT-to-cloud communication.

This study adopts a quantitative, applied engineering research approach, in which a working prototype is developed and subjected to controlled testing. Performance is assessed using latency, throughput, and scalability metrics under varying load conditions, while security is validated through token-based authentication, encrypted communication, and penetration testing of the API endpoints.

The remainder of this chapter is structured as follows: Section 4.2 discusses the research philosophy and approach; Section 4.3 outlines the overall research design; Sections 4.4 to 4.7 describe the software development methodology, hardware setup, and data collection methods; Section 4.8 details the evaluation plan; Section 4.9 presents data analysis strategies; Section 4.10 addresses ethical considerations; Section 4.11 highlights methodological limitations; and Section 4.12 concludes with a summary.

**4.2 Research Philosophy and Approach**

This project adopts a positivist and quantitative research philosophy, which emphasises objectivity, measurement, and empirical validation through observable data. The positivist paradigm is well suited for computer science research where artefacts such as software systems can be evaluated through measurable indicators of performance and security (Ryan, 2018). In this context, the focus lies on testing whether a middleware API Gateway can demonstrably improve the integration of ESP32 IoT devices with cloud services in terms of latency, throughput, and resilience.

The choice of a quantitative orientation is informed by the project’s objectives. Metrics such as latency, response time, and throughput lend themselves to numerical analysis, enabling statistical comparisons between different configurations and approaches (Creswell and Creswell, 2018). Security, though more difficult to quantify, is assessed through structured token validation, HTTPS enforcement, and penetration testing of API endpoints, producing objective outcomes such as blocked versus successful unauthorised access attempts. This aligns with the view of Wohlin et al. (2012), who argue that controlled experimentation is an effective strategy for evaluating software artefacts in empirical software engineering.

The research follows a deductive approach, beginning with hypotheses drawn from literature: for example, middleware-based architectures are expected to improve security and scalability compared with direct IoT-to-cloud communication (Driss et al., 2021). These hypotheses are then tested using the prototype implementation, and the findings are analysed against the theoretical claims.

While alternative paradigms such as interpretivism and qualitative approaches—e.g., user studies or interviews with developers—could provide insights into usability or developer adoption, they fall outside the scope of this project. The primary concern is system performance and security under technical evaluation rather than subjective perceptions. As Saunders et al. (2019) note, research design must align with the nature of the questions posed; here, objective measurement is more relevant than interpretive inquiry.

**4.3 Research Design**

This project adopts an experimental prototype-based research design, which is well-suited for applied computer science studies where the primary outcome is the development and evaluation of a technological artefact (Hevner and Chatterjee, 2010). In this case, the artefact is a RESTful API Gateway designed to facilitate secure and scalable integration between ESP32-based IoT devices and the Firebase Realtime Database. The prototype approach enables iterative design, implementation, and evaluation, providing empirical evidence of the system’s performance under controlled test conditions.

The study employs a single-case experimental design, focusing specifically on the ESP32 microcontroller as the IoT device and Firebase as the target cloud service. This narrow scope ensures depth of analysis and reflects real-world scenarios where developers often work with specific device–cloud pairings (Gregory et al., 2021). Moreover, case-based experimental designs are recognised for their ecological validity, since they closely mirror practical applications (Yin, 2018). By evaluating the system through realistic workloads generated via tools such as Postman and Apache JMeter, the results are directly applicable to IoT environments where resource-constrained devices interact with cloud-hosted services.

The strengths of this design lie in its replicability and practical relevance. Other researchers or practitioners can reproduce the system using the same ESP32 platform and Firebase, making it valuable for benchmarking lightweight IoT-to-cloud integrations. However, there are also limitations. The focus on a single device type and cloud service restricts generalisability: findings may not extend to alternative microcontrollers (e.g., Raspberry Pi Pico, STM32) or cloud platforms (e.g., AWS IoT Core, Azure IoT Hub). Furthermore, prototype-based experiments may overlook long-term reliability concerns such as device ageing or large-scale deployment effects.

Nonetheless, the research design aligns with the project’s objectives by prioritising a controlled yet practically grounded evaluation. It provides a balance between methodological rigour and real-world applicability, making it an appropriate choice for this dissertation.

**4.4 Project Management Approach**

The project adopts an Agile-inspired methodology to ensure flexibility, adaptability, and continuous improvement throughout the development process. Agile has become a dominant approach in software engineering due to its iterative cycles, incremental delivery, and responsiveness to change (Dingsøyr et al., 2019). For this project, weekly sprints will be used to break down the work into manageable tasks, including the configuration of the ESP32 device, the development of the RESTful API Gateway, and subsequent rounds of functional and performance testing. This iterative structure allows for early identification of issues, incremental integration of security features, and timely refinement based on evaluation results.

Risk management is integrated into the project plan to address potential challenges such as hardware malfunctions, software incompatibilities, and delays in implementation. As noted by Serrador and Pinto (2015), effective Agile projects include continuous risk monitoring and adaptive mitigation strategies. For example, maintaining backup ESP32 units, employing version control through GitHub, and scheduling buffer time between development and testing phases are planned to reduce risks of disruption.

Finally, the chosen approach aligns with expectations of a Master’s level Computer Science project, which emphasises both technical rigour and reflective practice. By combining structured engineering practices with adaptive project management, the methodology ensures that the research objectives are met within the required timeframe and to the expected academic standard.

**4.5 Software Development Methodology**

The software artefact developed in this project—a RESTful API Gateway—was implemented using Node.js with the Express.js framework. Node.js has become one of the most widely adopted backend platforms for IoT and cloud applications due to its event-driven, non-blocking I/O model, which supports lightweight and highly scalable systems (Tilkov and Vinoski, 2010; Chernyshev and Zhukov, 2019). Express complements Node.js by providing a minimal and flexible web framework that simplifies routing, middleware integration, and request handling, making it particularly well-suited for RESTful architectures (Kumar et al., 2020). These characteristics align directly with the objectives of this dissertation, where low latency, scalability, and efficiency are critical in managing IoT data flows between ESP32 devices and the Firebase Realtime Database.

Version control and collaborative management of the codebase were achieved using Git and GitHub, which have become industry standards for distributed development and research projects. GitHub not only supports version tracking but also enhances reproducibility and transparency in software engineering research, as changes are documented and can be revisited when testing alternative designs (Spinellis, 2012).

Development followed an incremental testing strategy, in which functionality was validated at each stage of implementation. The first stage involved ingesting data from the ESP32 sensors, followed by forwarding structured data to Firebase, and finally implementing security layers such as HTTPS encryption and JWT-based authentication. This approach ensured that potential faults were isolated early in development, reducing integration errors and supporting agile-inspired iteration (Stoica and Dobre, 2019).

Alternative technology stacks such as Python Flask and Java Spring Boot were considered. Flask offers simplicity and readability, making it useful for rapid prototyping, while Spring Boot provides enterprise-grade robustness. However, both frameworks are less lightweight than Node.js in handling high-concurrency IoT traffic. Empirical comparisons suggest that Node.js generally outperforms Flask in request handling under heavy load (Kumar et al., 2020), which reinforces its suitability for this project.

**4.6 Hardware and IoT Device Setup**

The hardware foundation of this project is built around the ESP32 microcontroller, a low-cost, low-power system-on-chip widely adopted in IoT research and industry due to its integrated Wi-Fi and Bluetooth capabilities (Almuhaya et al., 2020). Its dual-core processor and flexible GPIO support make it particularly suitable for environmental monitoring applications where continuous sensor data collection is required. For this project, the ESP32 is configured to transmit sensor readings via Wi-Fi using HTTP requests to the RESTful API Gateway, thereby emulating real-world IoT-to-cloud communication scenarios.

To measure environmental parameters, DHT11 and DHT22 digital sensors are integrated with the ESP32 to capture temperature and humidity data. These sensors are commonly used in IoT prototyping because of their low cost, reasonable accuracy, and wide support in existing libraries (Méndez et al., 2019). The data collected provides a practical test case for validating both the performance and security of the proposed middleware.

Firmware development and device configuration are carried out using the Arduino IDE and MicroPython, both of which are well established in IoT prototyping for their accessibility, strong community support, and compatibility with ESP32 hardware (Garcia et al., 2020). Moreover, the ESP32 platform has been the subject of several security analyses identifying vulnerabilities in firmware and communication protocols (Litayem et al., 2023). Leveraging this device therefore provides both an accessible and a security-relevant testbed for evaluating the effectiveness of middleware-based protections.

**4.7 Data Collection Methods**

Data collection in this project is designed to generate original, empirical evidence on the performance and security of the developed RESTful API Gateway. Experimental data will be gathered on key system performance indicators, including latency, throughput, resource consumption, and error rates. These metrics are widely recognised as fundamental to assessing IoT middleware and API-based communication frameworks (Benkhelifa et al., 2018; Hussain et al., 2020). Latency and throughput measurements will reveal the efficiency of the gateway under varying network loads, while error rates and resource usage will highlight its stability and suitability for resource-constrained devices such as the ESP32.

For security evaluation, data will be collected from manual penetration testing of API endpoints, with a focus on detecting vulnerabilities to injection attacks and unauthorised access. In addition, JSON Web Token (JWT) validation logs will be analysed to confirm correct enforcement of authentication and authorisation policies. These methods reflect current best practices for evaluating lightweight IoT security mechanisms (Patel and Doshi, 2021).

Two tools will be central to data collection: Postman, for functional validation of endpoints, and Apache JMeter, for automated load and stress testing. Postman enables systematic testing of API correctness and data handling, while JMeter provides controlled simulation of concurrent IoT traffic, allowing for quantitative measurement of scalability (Mahmoud et al., 2019).

This project prioritises original data generation through instrumentation of the prototype system rather than secondary datasets. The rationale for this choice is twofold: firstly, IoT systems are highly context-dependent, making pre-existing datasets less reliable for targeted evaluation; secondly, primary data enables more granular measurement of security and performance metrics under the specific architecture implemented in this study.

**4.8 Evaluation Plan**

A structured evaluation plan is essential to determine whether the proposed RESTful API Gateway fulfils its objectives of providing secure, scalable, and efficient data integration between the ESP32 and Firebase Realtime Database. The evaluation combines performance testing, scalability analysis, and security validation, alongside a comparative study against direct ESP32-to-Firebase communication.

Performance evaluation will focus on measuring latency, throughput, and API response times under varying traffic conditions. Latency will be defined as the elapsed time between sensor data submission and its successful storage in Firebase, while throughput will capture the number of requests successfully processed per second. These metrics are commonly used in IoT communication studies to assess system responsiveness and efficiency (Hussein et al., 2020; Bhardwaj et al., 2020). Tools such as Postman for functional verification and Apache JMeter for automated load testing will be employed to generate realistic workloads and record outcomes.

Scalability evaluation will simulate multiple IoT devices by progressively increasing the number of concurrent virtual clients in JMeter. This approach allows observation of how the API Gateway performs under stress and identifies potential bottlenecks. Prior studies emphasise scalability testing as a critical step for IoT middleware validation, particularly when deployment involves heterogeneous devices (Chung, 2024).

Security evaluation will test the robustness of HTTPS/TLS encryption, validate JWT-based authentication, and include targeted penetration testing of API endpoints. This aligns with best practices highlighted by Shahzadi et al. (2019), who argue that input validation and token-based security mechanisms are essential to reducing common IoT vulnerabilities.

Finally, a comparative evaluation will be conducted, benchmarking the gateway against a direct ESP32-to-Firebase configuration. Metrics such as mean latency, standard deviation, and maximum throughput will be reported, providing quantitative evidence of whether the middleware introduces overhead or enhances system performance. Such comparative testing has been shown to provide empirical insights into the trade-offs between direct and mediated IoT-to-cloud integration (Patel and Doshi, 2021).

**4.9 Data Analysis Methods**

The analysis of data in this study follows a mixed strategy that combines descriptive and inferential statistical methods with qualitative assessment of security outcomes. Quantitative evaluation will be primarily concerned with the performance of the RESTful API Gateway under varying load conditions. Descriptive statistics such as mean, variance, and standard deviation will be employed to summarise latency and throughput data, providing an overview of central tendencies and dispersion across test runs (Field, 2018). These measures will highlight baseline performance as well as variability introduced by simulated stress conditions.

Where comparisons between multiple test configurations are necessary, such as direct ESP32–Firebase communication versus integration through the API Gateway, inferential statistical tests will be applied. For experiments with two groups, independent sample *t*-tests will be conducted, while scenarios with more than two conditions will be analysed using ANOVA to determine whether differences are statistically significant (Gravetter and Wallnau, 2020). This approach aligns with recent IoT evaluation studies that emphasise the importance of rigorous statistical treatment in performance benchmarking (Al-Fuqaha et al., 2019).

Visualisation will play a critical role in interpreting results. Latency distribution histograms and throughput curves will be plotted to capture performance dynamics under increasing simulated device loads (Hussein et al., 2020). For the security evaluation, outcomes of penetration testing and JWT token validation will be subjected to qualitative coding, distinguishing between successful, blocked, or anomalous requests. Finally, server and gateway logs will be analysed to support traceability and detect irregular patterns, in line with best practices for anomaly detection in IoT systems (Ferrag et al., 2020).

**4.10 Ethical Considerations**

Although this project does not involve human participants, ethical approval is still required to ensure that all research activities conform to the standards set out by the University of Lincoln. The primary ethical concerns relate to data integrity, responsible disclosure of vulnerabilities, and the safe handling of IoT devices.

First, all experimental data generated through performance and security testing will be original, ensuring that integrity and reproducibility are maintained. According to Blease et al. (2020), transparency and accurate reporting are fundamental to maintaining trust in research outcomes, even in projects that do not involve sensitive personal data. To support this, the project will employ secure data storage practices and version-controlled code repositories.

Second, the research may expose vulnerabilities in ESP32 devices or the middleware gateway. As suggested by Garfinkel (2021), responsible disclosure is essential to avoid enabling malicious exploitation of identified weaknesses. Any findings with potential security implications will therefore be reported in line with academic standards, without publishing sensitive exploit details.

Finally, only small-scale penetration testing will be conducted on locally deployed systems, avoiding any large-scale or harmful attacks that could disrupt broader networks. This aligns with IoT security research guidelines that stress minimising risk when simulating adversarial conditions (Sicari et al., 2019).

In line with the University of Lincoln’s research ethics framework (University of Lincoln, 2020), this project ensures that all procedures are conducted responsibly, with an emphasis on minimising risk, protecting data, and producing valid, ethical results.

**4.11 Limitations of the Methodology**

While the methodology outlined in this dissertation provides a structured approach to developing and evaluating a RESTful API Gateway, it is not without limitations. First, the study focuses exclusively on the ESP32 microcontroller as the IoT device platform. Although the ESP32 is widely adopted in both academic and commercial IoT projects due to its cost-effectiveness and integrated wireless capabilities, findings derived from a single hardware platform may not fully generalise to other microcontrollers with different architectures, processing constraints, or security features (Litayem et al., 2023).

Similarly, the project relies solely on Firebase Realtime Database as the target cloud platform. While Firebase is suitable for real-time synchronisation and rapid prototyping, its architecture differs from other IoT-oriented cloud services such as AWS IoT Core or Azure IoT Hub, which provide more granular device management and built-in security frameworks (Zhou et al., 2020). Consequently, results may be platform-specific and not universally applicable across cloud environments.

Another limitation arises from potential network variability during performance testing. Wi-Fi latency and packet loss can introduce noise into performance measurements, making it challenging to fully isolate the impact of the API Gateway from external environmental factors (Al-Fuqaha et al., 2019).

Nevertheless, the narrow focus of this study is justified, as it allows for a deep and systematic evaluation of the ESP32–Firebase integration scenario. This depth ensures meaningful insights into the feasibility of middleware solutions for small-scale IoT deployments, which can inform and guide future research involving broader device and cloud ecosystems.

**4.12 Summary**

This chapter has outlined the methodological framework adopted to achieve the aim of designing, implementing, and evaluating a RESTful API Gateway for secure and scalable integration between an ESP32-based IoT device and the Firebase Realtime Database. The research follows a quantitative, prototype-driven, and experimental approach, ensuring that the developed artefact is not only functional but also subject to systematic evaluation under controlled conditions.

The chapter began by situating the research within a positivist paradigm, justifying the use of objective measurements to assess system performance and security. A structured experimental design was employed, incorporating iterative development using Node.js and Express, alongside the configuration of the ESP32 microcontroller for real-time data collection. Data was generated through functional, stress, and penetration testing, with tools such as Postman and Apache JMeter providing reliable insights into latency, throughput, and scalability. Security validation was achieved through HTTPS, JWT-based authentication, and input validation, with manual testing employed to identify potential vulnerabilities.

By integrating both performance and security evaluation, the methodology ensures a balanced analysis of the gateway’s effectiveness, highlighting the trade-offs faced by resource-constrained IoT systems. This foundation sets the stage for the subsequent chapters: Implementation, which details the system development process, and Results and Discussion, which critically evaluates empirical findings against the research objectives.

**Chapter 4**

# Implementation

This section of the dissertation may vary significantly in both structure and content, depending on the type of project you are undertaking.

The precise structure should be discussed with your supervisor, but some suggestions for additional sections are given below. The key thing to note here is that irrespective of the project type, you should *justify* the choices you've made, rather than simply choosing based on expediency or familiarity. The main types of projects a student will do on the CMP9140 Research project module are below:

* Software Development Projects
* Research-focused projects not involving human participants
* Research Projects involving human participants

Its possible for a single project to adopt aspects of one or more of the types of project listed above.

## 4.1 Software development projects

If the primary deliverable of your project is a software product, then you should consider subsections detailing your approaches to the following:

* Toolsets and machine environments (i.e., the software and hardware used)
* Design (e.g., UML diagrams, database schema, prototypes)
* Testing (i.e., the types of testing used)

This list is not exhaustive. For example, a games design project may include a game design document. However, it must be noted that if your project contains significant software development work, then most if not all of these sections should be present.

## 4.2 Research projects *not* involving human participants

For some projects, the main deliverables may come in the form of experimental results. For example, a project comparing several different algorithms may require little in the way of code, but require considerable experimentation and data analysis. As such, all methodological choices made should be documented here. Examples include:

* Dataset acquisition and annotation
* Algorithm/model design and selection
* Parameter tuning
* Performance metrics

Again, this list is not exhaustive, and you should still include relevant sections pertaining to the software artefact listed in Section 4.1

## 4.3 Research projects involving human participants

For projects involving human participants, you will need to consider a hypothesis or research question that your project will answer. You will also need ethical approval using the LEAS system. In addition to the sections outlined in Section 4.1 if you are also developing software, you may also need to provide details of:

* Participant recruitment
* Evidence that ethical procedures have been followed
* Study design (including hypotheses/research question as appropriate)
* Quantitative/Qualitative analysis (i.e., how you'll analyse the raw data)

If your study involves data collection by means of questionnaires, you may also wish to specify the questions here (or refer to them in an appendix

**Chapter 5**

# Results & Discussion

This section should present the findings of your work, and discuss them in the context of your original aims & objectives. For software-oriented projects, how well does it meet your original requirements? Provide data where possible, e.g., results of user testing, performance measures, etc. For research-oriented projects, you should present the data in an appropriate format (tables, charts, visualisations) and provide a critical discussion around the results that provides insight and direction for future work.

**Chapter 6**

# Conclusion

This is another relatively short chapter that is an opportunity to reflect on the project as a whole. Discuss the limitations and successes of the project, highlighting opportunities for future work.

# References

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. and Ayyash, M. (2019) ‘Internet of Things: A survey on enabling technologies, protocols, and applications’, *IEEE Communications Surveys & Tutorials*, 21(3), pp. 2568–2609.

Almuhaya, M.A., Alhumam, A. and Alqahtani, H. (2020) ‘Performance evaluation of ESP32 microcontroller for IoT applications’, *International Journal of Advanced Computer Science and Applications*, 11(3), pp. 77–84.

Alrashdi, I., Alqazzaz, A., Aloufi, E., Alharthi, R., Zohdy, M. and Ming, H. (2020) ‘AD-IoT: Anomaly detection of IoT cyberattacks in smart city using machine learning’, *Proceedings of the IEEE International Conference on Smart Applications, Communications and Networking (SmartNets)*, pp. 1–6.

Alrawais, A., Alhothaily, A., Hu, C. and Cheng, X. (2019) ‘Fog computing for the Internet of Things: Security and privacy issues’, *IEEE Internet Computing*, 21(2), pp. 34–42.

Alsubaei, F., Abuhussein, A. and Shiva, S. (2019) ‘Security and privacy in the Internet of Medical Things: Taxonomy and risk assessment’, *Future Generation Computer Systems*, 92, pp. 342–359.

Alwarafy, A., Al-Thelaya, K., Abdallah, M., Schneider, J. and Hamdi, M. (2020) ‘A survey on security and privacy issues in edge-computing-assisted Internet of Things’, *IEEE Internet of Things Journal*, 8(6), pp. 4004–4022.

Alzubi, J., Manaseer, S. & Shatnawi, A. (2020) ‘Performance evaluation of cloud-based IoT platforms: A comparative study of Firebase, AWS, and Azure’, *International Journal of Advanced Computer Science and Applications*, 11(9), pp. 134–142.

Benkhelifa, E., Welsh, T. and Hamouda, W. (2018) ‘A critical review of practices and challenges in intrusion detection systems for IoT’, *Ad Hoc Networks*, 84, pp. 60–81.

Bhardwaj, R., Patel, P. & Doshi, N. (2020) ‘Secure authentication in IoT using JSON Web Tokens’, *International Journal of Information Security Science*, 9(3), pp. 108–118.

Blease, C., Lilienfeld, S.O. & Kelley, J.M. (2020) ‘Scientific integrity and the reproducibility crisis: Implications for the behavioural sciences’, *Perspectives on Psychological Science*, 15(4), pp. 817–826.

Boyes, H., Hallaq, B., Cunningham, J. and Watson, T. (2018) ‘The industrial internet of things (IIoT): An analysis framework’, *Computers in Industry*, 101, pp. 1–12.

Butun, I., Pereira, N. and Gidlund, M. (2019) ‘Security risk analysis of IoT solutions: A case study of the Smart Grid’, *IEEE Access*, 7, pp. 144164–144178.

Chernyshev, S. and Zhukov, D. (2019) ‘Node.js: Features, Benefits and Drawbacks for IoT Development’, *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pp. 1555–1558.

Chung, S. (2024) ‘Middleware scalability in heterogeneous IoT systems: A performance evaluation’, *Future Generation Computer Systems*, 157, pp. 321–334.

Chung, T. (2024) ‘API Gateway architectures in microservices and IoT: A survey of patterns, challenges, and research opportunities’, *Journal of Cloud Computing: Advances, Systems and Applications*, 13(1), pp. 1–19.

Colitti, W., Steenhaut, K., De Caro, N., Buta, B. & Dobrota, V. (2018) ‘Evaluation of constrained application protocol for wireless sensor networks’, *Proceedings of the 18th IEEE Symposium on Computers and Communications (ISCC)*, pp. 20–26. IEEE.

Creswell, J.W. & Creswell, J.D. (2018) *Research design: Qualitative, quantitative, and mixed methods approaches*. 5th ed. Thousand Oaks: SAGE.

Das, S., Mukherjee, A., Chattopadhyay, S. and Roy, S. (2020) ‘Performance evaluation of elliptic curve cryptography on IoT devices: ESP32 as a case study’, *Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 235–242.

Dingsøyr, T., Nerur, S., Balijepally, V. and Moe, N.B. (2019) ‘A decade of agile methodologies: Towards explaining agile software development’, *Journal of Systems and Software*, 144, pp. 1–13.

Driss, M., Hasan, D., Boulila, W. and Ahmad, J. (2021) ‘Microservices in IoT security: Current solutions, research challenges, and future directions’, *IEEE Access*, 9, pp. 119984–119999.

Driss, M., Hasan, D., Boulila, W. and Ahmad, J. (2021) ‘Microservices in IoT security: Current solutions, research challenges, and future directions’, *Journal of Network and Computer Applications*, 190, 103139.

Driss, M., Mellouk, A. & Azzoug, Y. (2021) ‘Middleware solutions for the Internet of Things: A comparative survey’, *Computer Networks*, 193, pp. 108–123.

Espressif Systems (2021) *ESP32 Security Features: Secure Boot and Flash Encryption*. Available at: <https://www.espressif.com> (Accessed: 12 August 2025).

Fang, H., Wu, Z., Qian, Y. and Hu, R.Q. (2021) ‘Security for 5G mobile wireless networks’, *IEEE Access*, 9, pp. 91236–91263.

Farzaneh, A. and Kargahi, M. (2019) ‘Rate limiting for denial-of-service mitigation in IoT middleware platforms’, *Proceedings of the 27th Iranian Conference on Electrical Engineering (ICEE)*, pp. 1522–1527.

Fernández-Caramés, T.M. and Fraga-Lamas, P. (2019) ‘A review on the use of blockchain for the Internet of Things’, *IEEE Access*, 6, pp. 32979–33001.

Ferrag, M.A., Shu, L., Yang, X., Derhab, A. and Maglaras, L. (2020) ‘Security and privacy for green IoT-based agriculture: Review, blockchain solutions, and challenges’, *IEEE Access*, 8, pp. 32031–32053.

Field, A. (2018) *Discovering statistics using IBM SPSS statistics*. 5th ed. London: Sage.

Garcia, J., Perera, C. and Georgakopoulos, D. (2020) ‘Programing IoT devices: Arduino, ESP8266, and ESP32’, *IEEE Internet of Things Magazine*, 3(2), pp. 52–58.

Garfinkel, S. (2021) ‘Responsible vulnerability disclosure in the era of IoT’, *Communications of the ACM*, 64(2), pp. 22–24.

Gravetter, F. and Wallnau, L. (2020) *Statistics for the behavioral sciences*. 10th ed. Boston: Cengage.

Gregory, P., Rahman, H. and Taylor, I. (2021) ‘Middleware for the Internet of Things: A comparative analysis’, *Future Generation Computer Systems*, 115, pp. 87–102.

Gupta, R. & Rao, K. (2021) ‘A real-time health monitoring system using IoT and Firebase Cloud’, *Journal of Ambient Intelligence and Humanized Computing*, 12(9), pp. 9231–9244.

Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P. & Sikdar, B. (2020) ‘A survey on IoT security: Application areas, security threats, and solution architectures’, *IEEE Access*, 7, pp. 82721–82743.

Hevner, A. and Chatterjee, S. (2010) *Design Research in Information Systems: Theory and Practice*. Springer, New York.

Hussain, F., Hussain, R., Hassan, S.A. and Hossain, E. (2020) ‘Machine learning in IoT security: Current solutions and future challenges’, *IEEE Communications Surveys & Tutorials*, 22(3), pp. 1686–1721.

Hussein, D., Han, S. & Lee, Y. (2020) ‘Lightweight security mechanisms for resource-constrained IoT devices: A comparative analysis of MQTT and REST’, *Sensors*, 20(20), pp. 1–18.

Ibarra-Esquer, J.E., González-Navarro, F.F., Flores-Rios, B.L., Burtseva, L. and Astorga-Vargas, M.A. (2020) ‘Tracking the evolution of the Internet of Things concept across different application domains’, *Sensors*, 20(23), 7097.

Khan, R., Khan, S.U., Zaheer, R. & Khan, S. (2021) ‘Future Internet: The Internet of Things architecture, possible applications and key challenges’, *2018 10th International Conference on Frontiers of Information Technology (FIT)*, pp. 257–260. IEEE.

Kolias, C., Kambourakis, G., Stavrou, A. and Gritzalis, S. (2021) ‘DDoS in the IoT: Mirai and other botnets’, *Computer*, 54(7), pp. 60–67.

Kumar, A., Sharma, V., Singh, S. and Park, J.H. (2021) ‘Secure and lightweight authentication scheme for IoT devices using gateway-based verification’, *Sensors*, 21(3), 802.

Kumar, R., Dutta, K. and Nath, R. (2020) ‘Comparative performance analysis of Node.js and Flask frameworks in web application development’, *International Journal of Advanced Computer Science and Applications*, 11(5), pp. 145–152.

Litayem, N. & Al-Sa'di, A. (2023) ‘Exploring the programming model, security vulnerabilities, and usability of ESP32 platforms for IoT development’, *International Journal of Computer Applications in Technology*, 70(1), pp. 54–66.

Mahmoud, R., Yousuf, T., Aloul, F. and Zualkernan, I. (2019) ‘Internet of Things (IoT) security: Current status, challenges and countermeasures’, *International Journal of Information Security*, 18(3), pp. 1–18.

Méndez, D., Papapanagiotou, I. and Yang, B. (2019) ‘Internet of Things: Survey on security and privacy’, *Information Security Journal: A Global Perspective*, 28(2), pp. 66–76.

Naik, N. (2017) ‘Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP’, *2017 IEEE International Systems Engineering Symposium (ISSE)*, pp. 1–7. IEEE.

Patel, K. & Doshi, N. (2021) ‘Performance analysis of MQTT and REST for lightweight IoT communications with security extensions’, *Journal of Reliable Intelligent Environments*, 7(2), pp. 139–153.

Patel, K., Shah, D. and Patel, A. (2021) ‘IoT based smart agriculture with cloud integration using API gateway middleware’, *International Journal of Electrical and Computer Engineering*, 11(6), pp. 5205–5213.

Riahi, A., Guerroumi, M., Chaabouni, N. and Conti, M. (2022) ‘A survey on the security of Industrial Internet of Things: Standards, challenges and perspectives’, *Future Generation Computer Systems*, 134, pp. 252–274.

Rizwan, M., Anwar, S., Rehman, S. & Hussain, I. (2022) ‘Leveraging Firebase for secure IoT applications: A performance and vulnerability assessment’, *International Journal of Information Security Science*, 11(2), pp. 45–56.

Ryan, A. (2018) ‘Methodologies in computer science research: Positivism vs interpretivism’, *Journal of Computer Science Research*, 6(1), pp. 45–55.

Saunders, M., Lewis, P. & Thornhill, A. (2019) *Research methods for business students*. 8th ed. Harlow: Pearson.

Serrador, P. and Pinto, J.K. (2015) ‘Does Agile work?—A quantitative analysis of agile project success’, *International Journal of Project Management*, 33(5), pp. 1040–1051.

Shahzadi, M., Asim, M., Ullah, S., Qayyum, A., Qadir, J. & Younis, M. (2019) ‘Security in IoT: Contemporary research and future directions’, *Computer Networks*, 148, pp. 283–294.

Sharma, S., Tiwari, R. & Singh, A. (2020) ‘Design and implementation of a Firebase-based IoT smart home system’, *Procedia Computer Science*, 167, pp. 1981–1990.

Sicari, S., Rizzardi, A., Grieco, L.A. & Coen-Porisini, A. (2019) ‘Security, privacy and trust in Internet of Things: The road ahead’, *Computer Networks*, 148, pp. 429–453.

Sittón-Candanedo, I., García, S., De Prieto, G. and Rodríguez-Molina, J. (2020) ‘Edge computing, IoT and social computing in smart energy scenarios’, *Sensors*, 20(9), 2633.

Spinellis, D. (2012) ‘Git: The new build system’, *IEEE Software*, 29(3), pp. 100–101.

Stoica, A. and Dobre, C. (2019) ‘Agile methodologies in IoT software development’, *Procedia Computer Science*, 159, pp. 2291–2300.

Tarascó Acuña, M. and Vázquez Blanco, A. (2025) ‘Investigadores españoles descubren una brecha de seguridad en millones de aparatos cotidianos y activan las alarmas’, *MeriStation Tech Report*, 14 January.

Tilkov, S. and Vinoski, S. (2010) ‘Node.js: Using JavaScript to build high-performance network programs’, *IEEE Internet Computing*, 14(6), pp. 80–83.

University of Lincoln (2020) *Research Ethics Policy*. Lincoln: University of Lincoln.

Villamizar, M., Garcés, O., Ochoa, L., Castro, H., Verano, M., Salamanca, L., Casallas, R. and Gil, S. (2019) ‘Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud’, *Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER)*, pp. 149–160.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B. & Wesslén, A. (2012) *Experimentation in software engineering*. Berlin: Springer.

Wolfert, S., Ge, L., Verdouw, C. and Bogaardt, M.-J. (2019) ‘Big data in smart farming – A review’, *Agricultural Systems*, 153, pp. 69–80.

Xu, J., Zhang, H., Liu, Q. and Zhou, J. (2020) ‘An empirical study on scalability and performance of API gateway frameworks for microservices’, *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pp. 703–714.

Xu, L.D., Xu, E.L. and Li, L. (2018) ‘Industry 4.0: State of the art and future trends’, *International Journal of Production Research*, 56(8), pp. 2941–2962.

Yin, R.K. (2018) *Case Study Research and Applications: Design and Methods*. 6th edn. SAGE Publications, Thousand Oaks, CA.

Zhou, J., Cao, Z., Dong, X. and Vasilakos, A.V. (2020) ‘Security and privacy for cloud-based IoT: Challenges’, *IEEE Communications Magazine*, 58(6), pp. 26–31.