Case Study on Existing IoT Platform and Approach for Local IoT Platform

Sothea Phann[1,*], Morokot Cheat[1], Sombathmorhareach Neath[2], and Khemarin Sambath[1]

[1] Ministry of Post and Telecommunications, Phnom Penh, Cambodia
[2] ABA Bank Head Office, Phnom Penh, Cambodia
Email: Phannsothea17@gmail.com (S.P.); cheat.morokot@gmail.com (M.C.); neath.sombatmorha@ababank.com (S.N.); sambath-khemarin@mptc.gov.kh (K.S.)
[*]Corresponding author
Manuscript received December 12, 2023; revised January 12, 2024; accepted January 30, 2024; published May 13, 2024.

***Abstract*—The Internet of Things (IoT) is a paradigm facilitating the interconnection of physical objects and devices via the internet, enabling data collection and exchange. As the adoption of IoT continues to grow, the need for a deep understanding of IoT platforms becomes crucial. This paper provides an overview of the IoT architecture, comparing leading IoT platform such as AWS and Microsoft Azure vendor. The objective is to understand the overall architecture of these platforms and explore their main cores for potential customization in developing a proprietary IoT platform. Additionally, the study includes selecting a GPS device to get location for testing Asset Tracking in the platform as proof-of-concept, using an ESP32 acts as a communication gateway, and the software implementation is Arduino IDE. The research concludes by laying the groundwork for a case study, examining an existing IoT platform, and proposing an approach to develop an IoT platform based on local specific needs.**

***Keywords*—IoT platform, GPS, ESP32, Arduino IDE, IoT architecture**

I. Introduction

The emergence of the Internet of Things (IoT) has revolutionized the way we interact with our surroundings and has the potential to address challenges faced by society. With IoT, everyday objects are connected to the internet, enabling them to collect and exchange data, and ultimately make intelligent decisions [1]. With its ability to connect and communicate with physical objects, devices, and systems, the IoT has the potential to solve challenges faced by our society. From improving efficiency and productivity to enhancing safety and sustainability, the IoT is playing a crucial role in addressing pressing issues and shaping a more connected and intelligent world. One such case is in the healthcare industry, where IoT has proven to be instrumental in improving patient care, streamlining operations, and enhancing overall healthcare outcomes [2]. Similarly, IoT applications such as environmental monitoring, home automation, agriculture, aquaculture, health care, transportation and logistics have showcased their capacity to optimize processes and yield positive results [3]. The central component of an IoT solution is the IoT platform but there is not a single IoT platform that is universally useful for every use case, as the requirements and objectives of IoT projects can vary greatly. An effective operation of an IoT application relies on having a platform to run smoothly and exchange data [4]. There are several popular and versatile IoT platforms that are widely used across different industries and use cases like Amazon Web Service IoT, Microsoft IoT Core, Samsung SmartThings, OpenHAB, Apple HomeKit, FarmBeats, ThingWorx and , IBM Watson IoT platform [5]. In addition to utilizing existing platforms, there is also the option to develop new platforms, and customize them to meet specific requirements. Given the multitude of options, users encounter the challenge of selecting a platform that suits their applications. This has led to numerous studies on characteristics of IoT platforms and comparing the existing platforms based on various criteria to narrow down the selection process.

In previous research, Muhammed and Ucuz stated that "AWS dominates the market with the highest market share and extensive connections across devices, device-to-device, and device-to-cloud hubs. Conversely, while Microsoft Azure not leading in market share and connectivity, it compensates with a more comprehensive set of analytic services. Nevertheless, AWS is a better option in terms of security" [6].

Another comparison by Dr. Saraswat and Dr. Tripathi, presented that AWS has a vast global presence with many data centers, providing stable and reliable services across a wide range of products. It's an ideal choice for larger enterprises requiring versatile solutions, even if they come at a higher cost [7].

The comparative analysis by Bhonsle *et al*. [8], consists of the list of features and services offered by each vendor. The study recommended choosing AWS if scalability, availability, robust applications, reliability, and security are crucial factors. Conversely, Microsoft Azure is a better option if already invested in Microsoft and prefer a single provider.

Guth *et al.* [9] discussed about concepts, similarities, and differences of IoT platforms. This study includes IoT reference architectures of four open-source platforms namely FIWARE, OpenMTC, SiteWhere, Webinos, as well as four propriety IoT solutions: AWS IoT, IBM Watson's IoT Platform, Microsoft's Azure IoT Hub, and Samsung's SmartThings.

These studies offer substantial information and insights into IoT platforms, including platform comparisons and block diagram to build a platform. However, to the best of our knowledge, there is a gap in the existing research, with no studies introducing a fundamental and minimized IoT architecture specifically designed for local environments. In this paper, we address these issues by proposing an optimal architecture for local IoT platform.

Various IoT applications, such as sensor networks for agriculture, traffic control, Building Management Systems (BMS), and GPS, are gaining significant interest. As demand grows for autonomously managing all related data within an IoT platform, yet the complexity and cost associated with

doi: 10.18178/ijfcc.2024.13.2.614

current IoT platforms limit their accessibility, especially for small to medium-sized enterprises and local applications. This research seeks to bridge this gap by developing a minimalist architecture for a local IoT platform, focusing on simplification, cost reduction, scalability, and enhance ease of deployment. By analyzing existing platforms and streamlining their components, particularly through the use of GPS sensors, the study aims to propose a foundational IoT system that is both efficient and manageable. The anticipated results of this initiative include increased accessibility of IoT technology for small communities and localized projects, along with simplified scalability and management processes. Furthermore, the minimalist architecture is expected to foster innovation within specific IoT application areas, driving advancements and improvements in these fields.

The following sections of this paper are organized as follows: we introduce the reference architectures of IoT and outline all components in Section II. In Section III, we conduct a comparison and selection between Microsoft Azure and AWS vendors. The approach for developing a local IoT platform is detailed in Section IV.

II. Reference Architecture

This section introduces an IoT reference architecture illustrated in Fig. 1. Additionally, it provides overview of the reference architectures employed by Microsoft and AWS vendors.

A. IoT Reference Architecture

IoT architecture is a collection of component parts, network architecture, and cloud technologies that function in accordance with well-known IoT protocols and security standards. Linking these elements together, IoT architecture makes this all possible by ensuring data gets where it needs to and is processed correctly. IoT platforms usually based on currently employ no dominant standards or technologies. Instead, many platforms and gadgets consider many standards and technology. To implement an IoT system, users frequently need to research, set up, and integrate several designs and technologies. One of the fundamental structures in IoT is the three-layer architecture, but five-layer architecture is more suitable for the application involves detailed considerations, integrates diverse technologies, and spans a wide range of parameters [10].

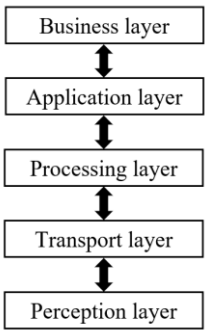

Fig. 1. IoT five-layer architecture.

1) Perception layer

This is where the sensors and connected devices come into play as they gather a huge amount of data as per the needs of the project. These can be the edge devices, sensors, and actuators that interact with their environment.

- Sensors or Actuators: these are the devices that are able to emit, accept and process data over the network. These devices may be connected either through wired or wireless. This contains GPS, Electrochemical, Gyroscope, RFID, etc. Most of the sensors need connectivity through sensor gateways. The connection of sensors or actuators can be through a Local Area Network (LAN) or Personal Area Network (PAN).
- Gateway: when a device cannot connect to other systems directly, it is linked via a Gateway. To convert between various protocols, communication technologies, and payload formats, a gateway offers the necessary technologies and procedures. The communication between devices and other systems is forwarded by it. To convert between various protocols, communication technologies, and payload formats, a gateway offers the necessary technologies and procedures. The communication between devices and other systems is forwarded by it.

2) Transport layer

The primary role of this layer is transportation [11]. As large numbers of data are produced by these sensors and actuator need high-speed Gateways and Networks to transfer the data. This network can be of type Local Area Network (LAN) such as Wi-Fi, Ethernet, Zigbee, NFC, Infrared, USB, and Wide Area Network (WAN) such as GSM, 5G, and LoRa WAN, among others.

3) Processing layer

In the world of IoT platforms, we focus on the moment when real things first link to the Internet—right before software takes control. These platforms are often called middleware software because they sit in the middle of connecting physical objects with digital systems. They are also recognized as a key component of an IoT system [12].

IoT Integration Middleware is a software layer or platform that facilitates seamless communication, integration, and interoperability between various components in an Internet of Things (IoT) system. It acts as a bridge between the diverse devices, protocols, applications, and data sources within the IoT ecosystem, enabling them to work together efficiently and effectively [13]. It is responsible for receiving data from the connected Devices, processing the received data, providing the received data to connected Applications, and controlling Devices.

4) Application layer

The application layer is the interface through which users interact with a system, delivering application specific services to the users. These applications include smart home, smart city, smart healthcare, and smart environment, among others [11]. In a smart home scenario, users can tap a button in the app to trigger actions like turning on a coffee maker.

5) Business layer

The business layer manages the core business logic and processes. It oversees the implementation of rules and policies, processes and analyses data, enforces security, integrates with enterprise systems, and strategizes monetization. Essentially, this layer ensures that IoT applications align with business objectives, fostering efficient and secure operations within the broader business ecosystem.

III. Comparison of AWS and Microsoft Vendor

A block diagram of a standard IoT platform based on Tamboli's work [13] is illustrated in Fig. 2. This diagram represents the distinctive architecture of the IoT solution, highlighting the building blocks and their separation based on the key aspects of a larger system.

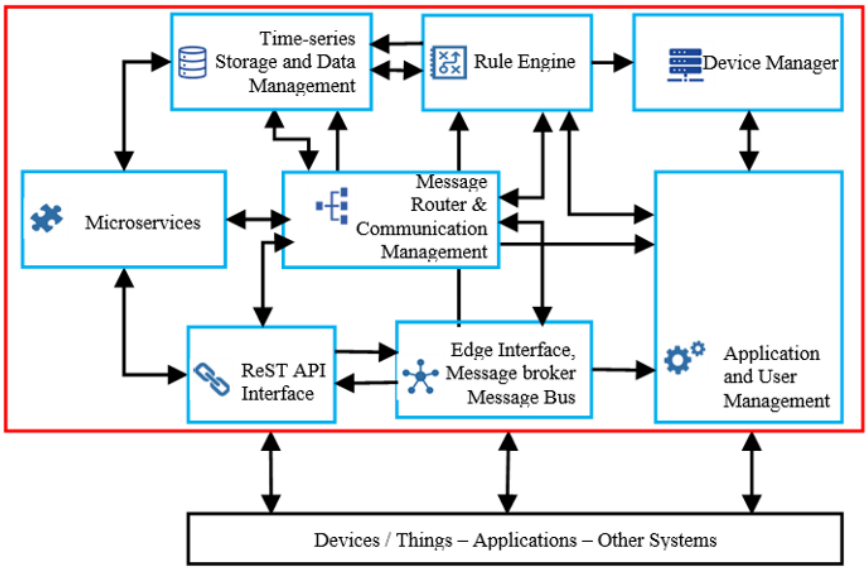

Fig. 2. Block diagram of a typical IoT platform.

A. Microsoft Reference Architecture

Azure stands as the cloud platform developed and operated by Microsoft [14]. Fig. 3 illustrates the architecture of Microsoft Azure for IoT architecture. The primary component in the architecture is the IoT Hub, serving as the central point to which all other components are connected. The gateway in this architecture offers two protocols: IP-capable devices have the capability to communicate either directly with IoT hub or through a Cloud Protocol Gateway. On the other hand, Personal Area Network (PAN)-devices require an additional Field Gateway [15]. IoT Integration Middleware consists of IoT Hub; Event Processing and Insight; Device Business Logic, Connectivity Monitoring; and Application Device Provisioning and Management components. Moreover, the Application Device Provisioning and Management component also facilitates the integration of additional applications.

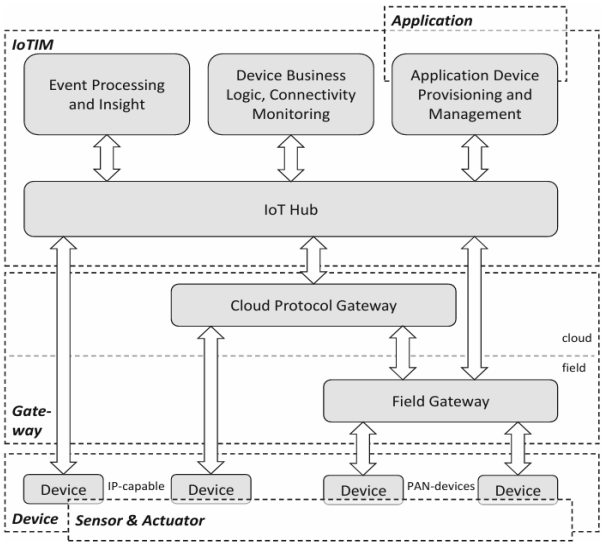

Fig. 3. Microsoft Azure IoT platform.

B. AWS Reference Architecture

The Amazon Web Services (AWS) IoT Core provides a secure and scalable platform for connecting, managing, and analyzing IoT devices and data. Figure 4 represents the architecture of AWS IoT. Within the AWS framework, the term "Thing" is referred to a device or object that is integrated with a sensor or actuator component. While the AWS IoT architecture may not depict a "Gateway" block, it's important to note that, based on the documentation, a Gateway component lies between the "Thing" and "Message Broker" components [15]. The Message Broker, Thing Shadows, Thing Registry, Rules Engine, and Security & Identity are the components of IoT Integration Middleware. These components play a vital role as core functions within the AWS IoT platform. Furthermore, the Application component contains various AWS data processing services that are already integrated into the platform. Similarly, the IoT Applications component streamlines the integration of further applications into the platform.

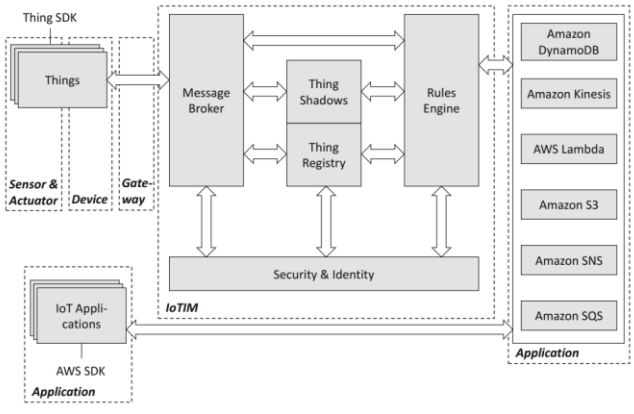

Fig. 4. AWS IoT platform.

C. Comparing between AWS and Microsoft

Table 1 provides a comparative analysis of essential components within IoT architecture, comparing standard

architecture with those of AWS IoT and Microsoft Azure IoT.

Table 1. IoT architecture components: Standard vs AWS vs microsoft

Standard	AWS	Microsoft
Device	Thing	Device
Edge Interface Message Bus Message Broker	Message Broker	Gateway
Message Router		IoT Hub
Rule Engine	Rule Engine	Event Processing and Insight Device Business Logic & Connectivity Monitoring
Device Manager	Security & Identity Thing Registry	Application Device Provisioning and Management
Time-Series Storage & Data Management	Thing Shadow	Event Processing and Insight
Microservice	Amazon S3 Amazon Kinesis Amazon Lambda Amazon SNS Amazon SQS Amazon DynamoDB	

The selection of an IoT cloud platform vendor, whether it is Microsoft Azure or AWS, relies on the specific requirements and preferences of the users. In this study, we conduct a comparison between these two cloud vendors based on three criteria namely: hub, analytic, and security. After a thorough literature review, it is evident that AWS emerges as a more preferable choice compared to Microsoft Azure, meeting specific user needs. Notably, Microsoft Azure's unavailability in Cambodia poses challenges for users in the region. Therefore, AWS stands out as a reliable and accessible option, offering services such as AWS IoT Core, AWS IoT Greengrass, AWS IoT Analytics, and AWS IoT Device Management. Additionally, AWS provides a 12-month free tier for new customers, allowing exploration of services at no initial cost, making it an attractive option for individuals and organizations to initiate their cloud computing journey.

IV. Approach

For this research, a case study approach is adopted to explore the integration and performance of AWS IoT in a real-world scenario. Focusing on the practical implementation of AWS IoT, the study involves testing with a GPS device and an ESP32 microcontroller serving as a gateway. This selection of devices ensures a contextually relevant examination of AWS IoT's capabilities in handling geospatial data within an IoT framework. The research goals include evaluating the platform's performance, assessing scalability, and identifying challenges associated with its implementation in a location-based IoT application. Subsequently, the study proposes an approach for a local IoT platform based on the insights gained from the evaluation.

A. Testing GPS with AWS IoT Core

This section details the process of testing GPS integration with AWS IoT Core, utilizing an ESP32 microcontroller as a key intermediary. The experimentation includes physical setup, software configuration, AWS configuration, and the subsequent results. By strategically configuring both hardware and software components, and AWS IoT Core, the section aims to demonstrate the real-time transmission of GPS data.

1) Hardware setup

Thoughtfully placing GPS sensors and configuring the ESP32 gateway are crucial aspects of the hardware setup. Considerations included ensuring unobstructed views of the sky for enhanced satellite signal reception. Factors like line-of-sight, environmental conditions and potential obstructions were carefully evaluated to guarantee accurate location data acquisition. Fig. 5 illustrates the hardware architecture for the experiment.

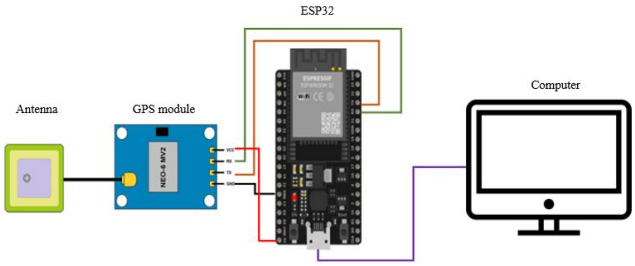

Fig. 5. Hardware setup diagram.

2) Software setup

As the ESP32 is a development board and needs instructions to perform specific functions, we need to use software to program the board properly. The flow of this experiment is shown in Fig. 6, by using the application of Arduino IDE to write code and upload it to the ESP32 board. The code is implemented to enable ESP32 to receive data from GPS sensor, establish a connection to a Wi-Fi network, and subsequently connect to AWS. MQTT publish/subscribe (pub/sub) topics are utilized for transmitting data between the ESP32 and AWS IoT Core. After reaching AWS IoT Core, the data is accessible and can be viewed through MQTT topics.

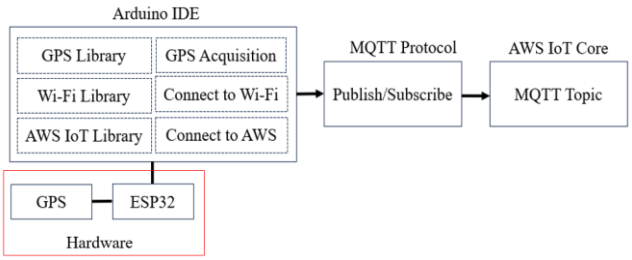

Fig. 6. Software setup diagram.

3) AWS IoT core configuration

The initial prerequisite for proceeding is to create an AWS account. Once the account is set up, we can have access to AWS services such as AWS IoT Core. Within the AWS IoT Core framework, the subsequent task is the creation of a "thing" to represent and identify the GPS sensors integrated into the IoT ecosystem, Fig. 7 displays the thing we have created. Following the creation of the thing, the next critical step is to attach a policy that defines specific permissions, such as allowing publish, subscribe, receive, and connect. As a final step, it is crucial to download the certificates associated with the thing, as they are essential for integrating the code into the Arduino IDE. It is important to note that

these certificates can be downloaded only once. The acquired certificates serve as key components in establishing a secure and authenticated connection between the GPS sensors and the AWS IoT Core platform.

Fig. 7. Created a thing in AWS IoT core.

4) Result

Following the successful upload of the code through the Arduino IDE, the GPS data is seamlessly transmitted to the AWS cloud. To observe the data, we need to navigate to the AWS IoT Core web page and access the MQTT Test Client located in the sidebar. In the topic filter section, fill in the predefined topic name (as defined in the Arduino sketch). After that, the MQTT Test Client will provide a real-time visualization of the transmitted GPS data. The accuracy of GPS data is primarily dependent on the GPS receiver and the conditions in which it operates. Both AWS IoT Core and the Arduino IDE receive data from the same GPS sensor, so the source of the data remains constant if there are no errors in the data acquisition or transmission process. The captured outcome in Fig. 8 indicates GPS data in serial monitor COM3 within Arduino IDE and received data in AWS IoT Core. The result verifies the successful transmission from the device to the cloud.

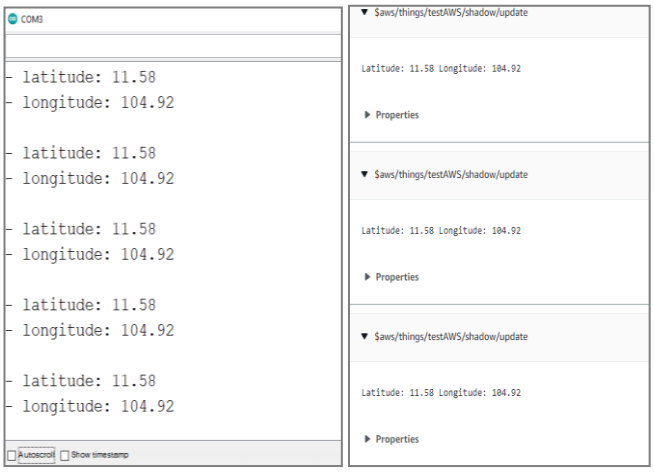

Fig. 8. GPS data in Arduino IDE serial monitor and AWS MQTT test client.

5) Discussion

Building a local IoT platform with GPS devices requires careful consideration of key components to ensure effective functionality within budget constraints.

B. Architecture Used for Testing

The experiment of IoT Core service of AWS and the reference architecture is shown in Fig. 4 above. The GPS device utilizes an ESP32 as a gateway to transmit its data through MQTT messaging protocol to the Message Broker. The GPS sensor, virtually represented, stores state information in a JSON document, encompassing both the last reported and desired states. The Message Broker relays these messages to subscribed clients. The AWS Rules Engine analyses and performs actions based on the MQTT topic stream. Additionally, it processes these messages for seamless integration with other AWS services, such as Amazon S3 which is a data storage service. It's important to highlight that, before initiating this process, a 'thing' for the GPS sensor was created (Thing Registry), and necessary credentials, including certificates, were required to ensure secure communication in AWS IoT (Security and Identity).

Although all the blocks in AWS IoT architecture define a well-architected IoT platform, not all are mandatory. The essential components include Things, Message Broker, Rule Engine, Thing Registry, Security and Identity, and Amazon S3 should be used for data storage. Each of these components plays a crucial role in the overall functionality of the IoT platform, enabling seamless communication, data processing, and security.

C. Proposed Local Architecture for IoT Platform

The proposed IoT structure is designed for the local condition to minimize resources for the beginning stage of local IoT platform development. The scope addresses the need of local IoT applications, focusing on scalability, cost efficiency, and deployment ease, tailored for small to medium-sized enterprises and localized projects. We can propose an architecture based on the insight from GPS testing, and the block diagram of a typical IoT platform presented by Tamboli [13].

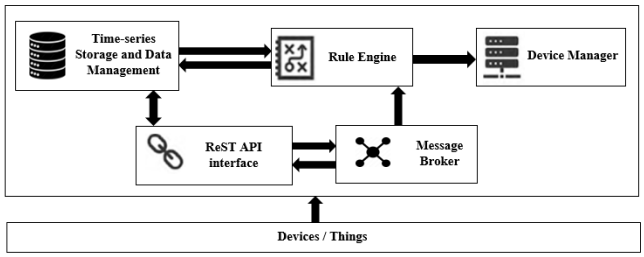

Fig. 9. Block diagram of proposed architecture.

A vital element is the data communication protocol to connect physical devices with the IoT platform. We have chosen the Message Broker via MQTT standard, as it is widely considered the default protocol for IoT applications. To ensure seamless integration, a REST API Interface has been incorporated, facilitating standardized communication between devices and the IoT platform for efficient data exchange. Following that, Device Manager is crucial for registration and monitoring IoT devices. The platform also requires a Rule Engine for trigger actions based on predefined rules such as alerts and optimize routes. Integrated rules dynamically respond to regional challenges, such as road conditions, weather patterns, and location-based events. Additionally, all messages should be stored in data storage for further analysis.

Each block in Fig. 9 can be developed by using either open-source or proprietary software. Nonetheless, our goal is to minimize costs, thus choosing open-source is preferable. For a single block, there are more than one open-source software options available. The development of an MQTT message broker can be accomplished using well-known solutions like Mosquitto, RabbitMQ, HiveMQ, and others. For creating a Device Manager, options such as Eclipse Kura and OpenRemote can be considered. Moreover, Express.js and FastAPI stand out as reliable platforms for building REST APIs. When it comes to Rule Engines, open-source choices like Drools, Easy Rules, and OpenL Tablets offer

flexibility. Moving on to the Time-Series Storage and Data Management, popular platforms such as InfluxDB, MySQL, and TimescaleDB provide robust solutions. There are several additional open-source options beyond the ones we mentioned, and we need to carefully choose the one that best suits our requirements.

V. Conclusion and Future Work

The increasing focus of major cloud platforms on the IoT sector is evident today. This research paper explores IoT platforms, comparing two key players: AWS IoT and Microsoft Azure IoT. AWS emerges as the preferred choice due to market dominance, strong connectivity, and superior security. The study successfully tested GPS sensors with AWS IoT Core, using ESP32 as a gateway, to evaluate platform performance, thus, laying the foundation for a localized IoT platform.

While the proposed architecture covers basic components, it signifies the initial step towards addressing specific requirements to minimize resources for local development in the first stage. However, constraints of our study include its exclusive focus on AWS and Azure, as well as the limited scope of our testing. Moving forward, our focus will shift to a more comprehensive exploration of additional components essential for building a robust IoT platform in the region. For the next research, a deeper exploration into the development of a proprietary IoT platform based on the proposed minimal local architecture should be undertaken by conducting a comparative analysis of various open-source solutions available for implementing each block.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Sothea Phann wrote the paper and carried out the experiments. Morokot Cheat came up with the idea and revised the paper. Sombatmorhareach Neath assisted in literature review. Khemarin Sambath created the design for the proposed system; all authors had approved the final version.

Funding

This work was generously funded by the APNIC Foundation through Switch! project.

Acknowledgment

We would like to express our gratitude to APNIC for the financial support received during the research and thank the Ministry of Post and Telecommunications for facilitating the research that led to these results.

References

[1] P. Gokhale, O. Bhat, and S. Bhat, "Introduction to IoT," *International Advanced Research Journal in Science, Engineering and Technology*, vol. 5, pp. 41–44, Jan. 2018. doi: 10.17148/IARJSET.2018.517

[2] S. Selvaraj and S. Sundaravaradhan, "Challenges and opportunities in IoT healthcare systems: a systematic review," *SN Appl. Sci.*, vol. 2, no. 1, p. 139, Jan. 2020. doi: 10.1007/s42452-019-1925-y

[3] H. Kotha and V. Gupta, "IoT application, a survey," *Int. J. Eng. Technol.*, vol. 7, p. 891, Mar. 2018. doi: 10.14419/ijet.v7i2.7.11089

[4] M. Ullah and K. Smolander, "Highlighting THE key factors of an IoT platform," in *Proc. 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia: IEEE, May 2019, pp. 901–906. doi: 10.23919/MIPRO.2019.8756748

[5] L. Babun, *et al.*, "A survey on IoT platforms: Communication, security, and privacy perspectives," *Comput. Netw.*, vol. 192, p. 108040, Jun. 2021, doi: 10.1016/j.comnet.2021.108040

[6] A. S. Muhammed and D. Ucuz, "Comparison of the IoT platform vendors, microsoft azure, amazon web services, and google cloud, from users' perspectives," in *Proc. 2020 8th International Symposium on Digital Forensics and Security (ISDFS)*, Beirut, Lebanon: IEEE, Jun. 2020, pp. 1–4. doi: 10.1109/ISDFS49300.2020.9116254

[7] M. Saraswat and R. C. Tripathi, "Cloud computing: comparison and analysis of cloud service providers-AWs, microsoft and Google," in *Proc. 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, Moradabad, India: IEEE, Dec. 2020, pp. 281–285. Accessed: Dec. 12, 2023.

[8] S. Bhonsle *et al.*, "Microsoft azure vs amazon cloud services: A comparative analysis," *IRE J.*, vol. 6, no. 9, pp. 1–6, Mar. 2023.

[9] J. Guth *et al.*, "A detailed analysis of IoT platform architectures: Concepts, similarities, and differences," *Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives*, 2018, pp. 81–101. doi: 10.1007/978-981-10-5861-5_4

[10] N. M. Kumar and P. K. Mallick, "The internet of things: Insights into the building blocks, component interactions, and architecture layers," *Procedia Comput. Sci.*, vol. 132, pp. 109–117, 2018. doi: 10.1016/j.procs.2018.05.170

[11] D. Navani, S. Jain, and M. S. Nehra, "The internet of things (IoT): A study of architectural elements," in *Proc. 2017 13th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS)*, Jaipur, India: IEEE, Dec. 2017, pp. 473–478. doi: 10.1109/SITIS.2017.83.

[12] V. Carchiolo *et al.*, "An efficient real-time architecture for collecting IoT data," presented at the 2017 Federated Conference on Computer Science and Information Systems, Sep. 2017, pp. 1157–1166. doi: 10.15439/2017F381

[13] A. Tamboli, *Build Your Own IoT Platform: Develop a Flexible and Scalable Internet of Things Platform*. Berkeley, CA: Apress, 2022. doi: 10.1007/978-1-4842-8073-7

[14] D. Bastos, "Cloud for IoT – A survey of technologies and security features of public cloud IoT solutions," *Living in the Internet of Things (IoT 2019)*, London, UK: Institution of Engineering and Technology, 2019, p. 43. doi: 10.1049/cp.2019.0168

[15] B. D. Martino *et al.*, *Internet of Everything*, in Internet of Things. Singapore: Springer Singapore, 2018. doi: 10.1007/978-981-10-5861-5