Journal of Harbin Engineering University
ISSN: 1006-7043
Vol 45 No. 5
May 2024

Seamlessly Backing Up Smart Home Messenger Notifications to the Cloud

[1]K.**Krishna Teja, [2]V. Poorna Sasank, [3]M. Rama Sai , [4]DR M. Kavitha., [5]DR S.S Aravinth**
Department of CSE,
Koneru Lakshmaiah Education Foundation,
Vaddeswaram,AP,India.

Abstract
The synergy between IoT and smart home technologies has ushered in an era of unprecedented connectivity and automation. As smart homes become hubs of data collection through interconnected devices, cloud-based storage solutions emerge as vital for managing and utilizing this influx of IoT-generated data. This symbiotic relationship optimizes data management and enables sophisticated data analysis, predictive insights, and intelligent living spaces. In this research paper, we build upon our previous work developing a Smart Home Messaging Notification System using IoT technology, showcasing the setup of an Arduino-based smart home system with remote control capabilities. We now expand on these achievements by implementing cloud-based data storage to analyze device usage and empower users to make informed decisions based on usage patterns, simplifying data analysis processes and providing a comprehensive overview of device usage trends.

I. Introduction

The emergence of the Internet of Things (IoT) has led to the rapid transformation of conventional households into interconnected smart homes. These environments incorporate a myriad of smart devices, such as sensors, appliances, and security systems, all capable of collecting and exchanging data. Central to this paradigm is the seamless integration of these devices to enhance automation, energy efficiency, and user convenience. However, the proliferation of IoT devices results in a substantial volume of data generated in real time, necessitating efficient data management solutions.

In our prior publication [19], we interfaced the device with a breadboard and ascertained its real time status via an Arduino device. The means by which the device was controlled was the Blink app, enabling users to both monitor and modify its operational state. However, a notable limitation of the aforementioned study was the requisite proximity of all components , everything had to be physically co-located. Moreover, handling the substantial volume of IoT data locally posed challenges related to both storage capacity and data security.

To address these challenges, we propose a solution in this paper, one that involves the integration of IoT devices with the cloud. This integration empowers each device to be operated seamlessly from any location with an internet connection, effectively linking and managing the devices and its data through cloud-based infrastructure.

Storing the vast stream of IoT-generated data poses a significant challenge due to its sheer volume, velocity, and variety. Cloud computing has emerged as a promising solution to tackle this challenge. Cloud platforms provide scalable storage and computational resources that can accommodate the dynamic and unpredictable nature of IoT data flows. By offloading data storage and processing to the cloud, smart homes can alleviate the constraints of local storage and computational limitations. This not only ensures the integrity of IoT generated data but also opens avenues for advanced analytics, real-time monitoring, and predictive maintenance.

IoT enabled smart homes offer unparalleled convenience and efficiency through the integration of diverse devices. The utilization of cloud storage for IoT-generated data addresses the challenges associated with data management, offering scalability and accessibility. As IoT continues to evolve, leveraging cloud infrastructure will play a pivotal role in harnessing the full potential of smart homes while fostering innovation in data-driven applications and services.

In this paper we are extending the work of our previous paper [19] on Smart Home Messenger Notification system using IoT in which we have created a demo environment of a smart home by taking an Arduino setup with a bulb connected to the bread board. In that we have connected the entire setup to

Arduino software and pushed the code into the Arduino board. We have also connected to the Blynk IoT app to control the bulb by writing the Wi-Fi details and the unique ID. After this we have sent the status of bulb to the Eclipse where we have written a code to take the status of bulb from serial monitor to Java file and send notifications to the Telegram app. In extension to this we are now storing the status of the devices into Cloud so that we can have an analysis of the usage of device and we also share it to the user so that he can have an idea of usage of his device and can also be used in future for any requirement. By storing in cloud, it makes our work easy by running analysis on the data as how many times the bulb is on in a day.

II. Related Work

In contemporary times, individuals find themselves constantly immersed in a world of technology, encompassing devices such as televisions, smartphones, computers, and even everyday essentials. As time progresses, people's demands have significantly evolved, necessitating advancements in the capabilities of these machines. To fulfill these needs, it is imperative to turn to the development of solutions that are rooted in cloud-based technologies.

In [1] the author delves into the fusion of Cloud Computing and IoT to enhance service provision, focusing on ubiquity, dependability, and scalability. The envisioned paradigm, termed "Everything as a Service," aspires to establish a Cloud ecosystem empowered with cognitive-IoT capabilities. The article addresses challenges inherent to IoT and Cloud, underscoring the pivotal importance of reliability and scalability. Cloud Computing's resource sharing and virtualization attributes are extolled as potential remedies. The convergence of Cloud and IoT is dissected via the lenses of Cloud-based IoT and IoT-Centric Cloud frameworks. The latter accentuates the distribution of data processing to curtail latency and heighten performance. The concept of IoT-Centric Cloud encompasses Local Clouds serving specific locales and a Global Cloud that functions as a fundamental infrastructure. Challenges linked to data governance, real-time processing, and seamless interaction are underscored as natural by-products of this amalgamation.

In [2] the author delves into the realm of IoT-centric social networks using a Cloud computing framework. It introduces a model for the collaborative sharing of IoT devices among users, enabling data capture and remote control. The proposed concept envisions a platform fostering data sharing from a variety of IoT devices, benefiting individual users and the larger community. The architectural structure encompasses three tiers: a diverse IoT device stratum, an intermediary layer for standardized device access, and a top-tier layer for end-user applications. The integration of a Cloud Data Management Interface (CDMI) bridge enhances the platform's capabilities by enabling the storage of IoT data within the Cloud. The study envisions scenarios where users register their devices, share them with peers, visualize data insights, and construct customized data interpretations. The approach not only opens avenues for innovative application development but also promotes user engagement and introduces the possibility of revenue sharing through well-constructed business models.

In [3], the author emphasizes that Data deterioration stands out as a paramount challenge within cloud data processing, with ramifications not only for the fidelity of individual application data outcomes but also for the overall performance and availability of the entire data processing ecosystem. The consequences of data degradation are substantial: inadequately managed corrupted data blocks can lead to complete data loss. Furthermore, data corruption can precipitate job failures, interminable loops, and unsuccessful client requests for Hadoop services. Hadoop does offer a mechanism for identifying data corruption, but it frequently falls short in recognizing various data degradation issues originating from software glitches. To address data corruption, two viable approaches emerge:

Retrying unsuccessful commands: This entails re-executing commands that have failed due to data corruption. Restoring corrupted data by creating duplicates of unaltered blocks and removing the compromised ones.

In [4], the author delves into strategies for safeguarding data within cloud storage. The expenses associated with maintaining a data center have been on the rise, particularly for medium-sized data centers. Therefore, a cost-effective alternative is to leverage cloud computing and cloud storage services rather than managing a data center in-house. To ensure the security of data stored in the cloud, the authors advocate the use of Service Level Agreements (SLAs) as a universal benchmark for agreements between users

and service providers. They further explore various technologies such as storage protection, transfer security, and authorization. A widely employed service in cloud environments is WS-Security, which plays a pivotal role in fortifying system security. Within WS-Security, XML encryption and XML signature techniques are harnessed to guarantee data confidentiality and data integrity.

In this paper [5], the authors introduce a synchronization and mirror machine based protection principle that offers a remarkably high level of network attack protection for distributed Internet of Things (IoT) units. These units are characterized by lacking stringent real-time demands, particularly the most vulnerable ones, such as battery-powered and resource-constrained units. These vulnerable units typically don't necessitate strict real-time capabilities. This novel approach proves its utility across a wide array of distributed IoT use-case scenarios wherein resource-constrained IoT units necessitate safeguarding from network-based attacks. Conversely, the proposed methodology does not yield benefits for more potent distributed IoT units that operate under real-time requirements. However, such units possess alternative and efficient measures for shielding themselves against network-based attacks.

In this paper[6], the authors introduce the notion of local clouds to address the specific requirements of automation applications, including real-time capabilities, security, scalability, ease of engineering, and interoperability. They argue that self-contained local clouds offer distinct advantages over other cloud computing models such as fog or edge computing in meeting these requirements. The concept of local automation clouds is demonstrated through a practical example involving closed-loop control of compartment climate. The successful demonstration validates the feasibility of required automation functionalities and real-time performance. The implementation of this concept employs the Arrowhead Framework. Data collected from two companies indicates substantial reductions in engineering efforts for implementing automation solutions using the described local cloud concept and Arrowhead Framework. The achieved savings are noteworthy, ranging from 3 to 5 times compared to traditional legacy technology-based implementations.

The author in paper[7] addresses the complexity arising from the fusion of IoT and Cloud Computing, focusing on access control in Cloud-connected IoT systems. The surge in IoT device numbers necessitates streamlined and efficient access control techniques. Conventional encryption methods could overwhelm IoT devices, leading to an exploration of Attribute-Based Encryption (ABE) due to its granular access management abilities. However, current ABE methods relying on bilinear maps might pose computational challenges, particularly for resource-limited IoT devices. The proposed solution introduces a lightweight ABE model for Cloud-based IoT, leveraging a hierarchical structure involving Root and Domain Authorities for access control and policy management. The approach aims to mitigate scalability issues tied to increasing user and device counts. Additionally, elliptic curve cryptography is proposed as an alternative to bilinear maps, targeting the reduction of computational complexities while improving efficiency. Key goals include developing an efficient ABE model, hierarchical attribute management, and ensuring consistent computation costs for encryption and decryption. The envisioned system involves Domain Authorities, Root Authority, and Cloud Servers to facilitate secure data storage and retrieval. Standard algorithms for setup, encryption, key generation, and decryption are outlined.

The author in paper[8] offers a brief glimpse into the realm of Luminex, delving into the puzzle of Radiance Defense, unveiling the Harmony Notion. It explores the Lumina design and its potential. Additionally, it dissects security concerns, vulnerabilities, controls, and compliance within the brilliance framework, highlighting sky-bound patrons, celestial stewards, and nebula envoys. The significance of safeguarding in the celestial service cosmos is expounded. The swift metamorphosis of Celeste's computing has redefined resource distribution and integration, ushering in both prospects and misgivings. Although boons like expenditure reduction and streamlined procedures are apparent, reservations about data seclusion and protection persist. This anxiety is amplified by the challenge of gauging the scope of data oversight for authorized or unauthorized intents. Through the materialization of the Trust Shield Framework and a rational mist architecture, the document navigates the intricacies of security puzzles and heaven-bound topics in the radiant milieu. It delves into queries like the

management of safety puzzles, selection of benchmarks and scrolls, realization of skyward safety requisites, and comparison of sky security levels amid astral amenities. The Celestas domain's diversity, comprising stellar paradigms like Zenith Services, Pinnacle Assistance, and Apex Infrastructure, is outlined, alongside the roles of zenith clients, providers, intermediaries, and carriers. The hurdles of multifaceted tenancy, responsibility allotment, and an ever-evolving sphere confer intricate security riddles. The document dialogues about data ownership, utility rights, and data evanescence concerns. The notion of "stellar misting" ushers in complexities, necessitating vigilant stewardship and authentication. Access dominion, vigilant monitoring, scrutinizing, and adherence to celestial benchmarks emerge as pivotal dilemmas. The concept of "Embark Your Apparatus in Celestial" (EYAC) and kindred safety threats are elucidated.

The author in paper[9] Local clouds have evolved in the area of technological innovation, providing personalized solutions for automation applications as well as their particular requirements such as real-time capabilities, security measures, scalability, engineering simplicity, and smooth interoperability. Self-contained local clouds stand out in this scenario, particularly prepared to meet these needs, surpassing competing cloud computing paradigms such as fog or edge computing. The visionary notion of local automation clouds has not only been conceptualized, but has also been effectively demonstrated. This manifestation emerged within the framework of a closed-loop control situation, namely within the field of compartment climate regulation. The critical automation functions and their related real-time performance benchmarks were thoroughly evaluated here, attesting to their superiority. The Arrowhead Framework, a vital facilitator of this ambitious idea, was used to bring this amazing implementation to life. In a step towards quantifiable benefits, observations from two independent corporate entities revealed significant reductions in engineering overheads when implementing automation solutions built in the local cloud paradigm described here. Notably, the scale of these economies was startling, hovering around three to fivefold in compared to traditional solutions connected to older technology. Such insights highlight the revolutionary potential of this localized cloud strategy, when combined with the Arrowhead Framework, to reverberate throughout sectors and usher in a new era of efficiency and creativity.

The author in paper [10] Collaboration between industry and academics has yielded fruit in the shape of standardized communication protocols in the ever-expanding world of IoT applications. This critical milestone opens the door for the development of IoT-specific frameworks and platforms. As time goes on, industrial consortia take over the responsibility of building these frameworks and platforms, with the ultimate goal of providing a solid basis at the application layer. This foundation will be critical in allowing the deployment of large-scale IoT applications, both in terms of instance size and number of instances. This survey provides an overview of commercially available frameworks and platforms for both industrial and consumer-centric IoT applications. Each of these frameworks viewed the IoT domain through the perspective of their clients' needs and desires. Some have prioritized centralizing distributed data sources to power cloud-based applications, called the "global cloud" strategy. Others, known as the "peer-to-peer" strategy, have focused their efforts on promoting the integration of devices for home or building automation. There are also many who have concentrated on the combination of devices and clouds, notably in the field of factory and industrial automation systems, which is referred to as the "local cloud" approach. Key aspects such as industry support, adherence to standards-based protocols, interoperability, security measures, hardware prerequisites, governance, and support for rapid application development have been scrutinized using a careful comparative analysis. This analytical endeavor acts as a compass for both academics and industry, directing them towards frameworks that are most appropriate for their prospective undertakings. This examination has also shown gaps in the present framework environment, putting light on areas ripe for innovation and development. Certain requirements must be met in order for a framework or platform to be successful. First and foremost, devices, apps, and systems must be able to securely expose APIs to third-party systems while also allowing API maintenance. Second, compatibility with protocols from other third-party APIs must be easily integrated, as well as the flexibility to extend to accept new protocols. Third, the integration of resource-constrained devices into application networks is vital, with characteristics such as size, bandwidth, power

supply (typically battery-based), and computing power all being important considerations. Finally, governance takes front stage, allowing for the effective administration of heterogeneous networks brimming with various devices and applications. The synergy of the whole exceeds the mere sum of its parts in this sophisticated mosaic of IoT frameworks and platforms. Collaboration, vision, and the constant pursuit of innovation characterize the route towards complete IoT integration and deployment.

Within this paper [11], as IoT smart devices engage in data exchange with other devices, a host of security concerns manifests, including the potential for data exposure, tampering, and unauthorized access. The authors put forth an agile cryptographic approach designed to facilitate data sharing among IoT smart devices operating at the edge of cloud-assisted IoT ecosystems, where all security-related procedures are outsourced to nearby edge servers. A robust data-sharing strategy is introduced, wherein IoT smart devices employ a combination of clandestine key encryption and public key encryption. Additionally, a search mechanism is proposed to enable authorized users to securely locate specific data within the encrypted dataset. In the realm of clandestine key encryption, a user device initially generates a confidential key, which is then used to encrypt the data before transmission to the recipient's device. Subsequently, the recipient device, utilizing the same key, can decrypt the data, thereby reverting it to its original form. Conversely, in the public key generation process, a single key is created and distributed to all authorized users. The forthcoming research agenda of the authors will focus on addressing authentication and access control challenges within this domain.

In this paper[12] the rapid adoption of Internet of Things (IoT) technology has led to increased demands on cloud-based datacenters for storage, processing, and management. However, despite their integration, IoT and cloud services still lack a uniform software layer for complex applications. IoT components and cloud services are developed separately, causing communication protocol and provisioning mismatches. Current cloud services lack coordination with IoT operations, hindering efficient load management. Integrating IoT and cloud services requires a cohesive software layer that allows communication and coordination between them. The concept of a Software-Defined Machine (SDM) is proposed to bridge this gap by enabling dynamic configuration and control of IoT elements within a virtualized environment. This approach could facilitate a more unified and coordinated execution environment for IoT cloud systems, leading to better scalability and management.

In the realm of advancing Internet-connected devices, the author in this paper [13] delves into an open-source IoT service known as "Link Sense," offering remote supervision of diverse parameters and objects through wireless sensors. These sensors capture real-time data and transmit it to a cloud-based repository via the internet. IoT has triggered a transformation in data aggregation and communication by intertwining physical gadgets and automobiles via digital sensors and cyberspace. A sensing component typically encompasses a plethora of detectors, encompassing aspects like temperature, humidity, light intensity, and multi-dimensional motion measurement. On the other hand, monitoring units focus on aspects such as electrical current and voltage consumption in household appliances. Wireless Sensor Networks (WSNs) have pioneered several applications, spanning from military operations to environmental monitoring. Nevertheless, WSNs confront limitations such as constrained computational capabilities and data transmission potential. Cloud-based computing emerges as a remedy to augment sensor efficiency, endowing instant access to computing assets via the internet. It boasts attributes like adaptability, automation, cost-effectiveness, and capacious data warehousing.

The document sheds light on the fusion of wireless device networks with cloud-linked computing to streamline the oversight of remotely linked device nodes and the ensuing data. It underscores the imperativeness of data governance in WSNs, especially sensor-derived data, which poses conundrums for extant structural frameworks.

In this paper[14] the author explained about the rise of Quantum Computing which has given rise to a new era of computational capabilities, challenging traditional computing paradigms. This article delves into the potential of Quantum Computing to revolutionize various fields, from cryptography to optimization problems.

Quantum Computing leverages the principles of quantum mechanics, harnessing quantum bits or "qubits" to perform complex calculations at exponentially faster speeds than classical computers. One notable application is in cryptography, where Quantum

Computers could break existing encryption methods, necessitating the development of quantum-resistant cryptography.

Furthermore, Quantum Computing shows promise in solving optimization problems, which have vast implications in fields like finance, logistics, and drug discovery. Quantum algorithms like Grover's and Shor's have demonstrated significant speedup in searching and factoring tasks. Despite the immense potential, Quantum Computing faces substantial challenges, including error correction and scalability. Researchers are actively working on quantum error correction codes to make quantum computers more robust. In conclusion, Quantum Computing has the potential to reshape various industries by addressing complex problems that classical computers struggle with. However, overcoming technical hurdles remains crucial to unlock its full capabilities.

In this document [15], the author introduced a broad Arrowhead methodology, elucidating its ability to accommodate services that cater to the needs of Quality of Experience (QoE) for interactions between consumers and providers. It delineated an overarching structure encompassing the elements involved in QoE administration and the services that facilitate their interactions. We advocate the utilization of Service Level Contracts (SLC) to articulate QoE requisitions, with the Arrowhead Synchronization System being responsible for negotiations with the QoE Management System to coordinate the services essential to support the desired QoE.

This design will serve as a foundation for forthcoming research endeavors, including the development of algorithms aimed at enhancing the Algorithms segment and broadening our methodology to situations beyond FTT-SE (Section VI). According to the author, forthcoming research will also investigate the influence of this approach on other facets of Arrowhead-compliant local clouds, such as safety and scalability.

In this article, the researcher [16] conducted an extensive exploration of a wide range of IoT solutions in the industry landscape, with a primary focus on context-sensitive computing. We briefly outlined the evolution of context-sensitive technologies, underscoring their increasing importance in modern applications. To commence, the investigator closely examined various IoT products to identify the context-sensitive functionalities they include. Following that, we divided the IoT solutions in the market into five distinct categories: intelligent wearables, connected residences, technologically advanced urban areas, responsive environments, and innovative businesses. Finally, the author identified and discussed seven notable findings and opportunities for future research and progress in the field of context-sensitive computing. The author's goal is to establish a foundation that promotes an understanding of the historical developments in the IoT sector, thereby enabling researchers to chart more efficient and effective pathways for the future.

The author [17] provided text discusses the challenges in ensuring the trustworthiness of data collected from Internet of Things (IoT) devices, particularly in mission-critical scenarios. It introduces a framework called TruSense designed to establish trust in IoT sensing data from end to end, encompassing IoT devices and cloud services. The framework includes a small sensing board, a communication protocol, and a cloud service. The main challenge addressed is the potential manipulation of IoT data by malicious actors, which could lead to significant consequences in areas like infrastructure and healthcare. TruSense aims to create a secure channel between sensors and cloud services to guarantee data integrity and authenticity. The paper discusses the implementation of a Trust Zone-based IoT sensing board and a cloud service for trusted sensing. It emphasizes the need for self-contained, secure code running in a Trusted Execution Environment (TEE) and direct control of sensors by the TEE. The framework also requires secure storage for cryptographic keys and integrity values and involves a trusted boot process for integrity measurement. The provided text discusses the challenges in ensuring the trustworthiness of data collected from Internet of Things (IoT) devices, particularly in mission-critical scenarios. It introduces a framework called TruSense designed to establish trust in IoT sensing data from end to end, encompassing IoT devices and cloud services. The framework includes a small sensing board, a communication protocol, and a cloud service. The main challenge addressed is the potential manipulation of IoT data by malicious actors, which could lead to significant consequences in areas like infrastructure and healthcare. TruSense aims to create a secure channel between sensors and cloud services to guarantee data integrity and authenticity. The paper discusses the implementation of a Trust Zone-based IoT sensing board and a cloud service for trusted sensing.

It emphasizes the need for self-contained, secure code running in a Trusted Execution Environment (TEE) and direct control of sensors by the TEE. The framework also requires secure storage for cryptographic keys and integrity values and involves a trusted boot process for integrity measurement.

The author [18] provided text discusses the emergence of cloud-sensor systems in the context of smart cities and IoT (Internet of Things) applications. It highlights the need for scalable architectures to handle the massive deployment of sensors and devices in smart cities. The Cloud-Edge-Beneath (CEB) architecture is introduced as a solution to address scalability and enable an open-ended application development ecosystem. The architecture separates device integration from service/application development and emphasizes programmability through an event-driven programming model. It also considers the scalability and energy efficiency of the system, given the extensive interactions between cloud services and physical sensors. The CEB architecture consists of four layers: Beneath (physical sensors and sensor platforms), Edge (intermediate layer managing sensor groups), Cloud (where sensor-based services are developed), and Applications (end-user applications). This architecture aims to provide a framework for deploying, programming, and managing cloud-sensor systems in a scalable and energy-efficient manner.

The paper [19] discusses how Internet of Things (IoT) technology can be used to develop smart homes that automatically control lighting, temperature and security based on the presence of occupants. The authors developed an IoT-based application that notifies users about the status of devices using WhatsApp or Telegram APIs.

The system consists of an Arduino board, NodeMCU Wi-Fi module, LED light bulb and resistor. Code written in Arduino controls the LED bulb through the Wi-Fi module. The status of the LED bulb is sent to a Java program using the serial monitor, which then shares the status with users using WhatsApp or Telegram APIs. The algorithm describes the steps to connect the Arduino serial monitor with Java and configure the Blynk app to control the LED bulb.

The paper demonstrates a simple IoT system to notify users about the status of devices (LED bulb) using chat apps like WhatsApp and Telegram. It uses affordable hardware like Arduino, NodeMCU and a light bulb and provides an easy interface for non-technical users. However, the authors suggest that the system can be extended further to control more devices and improved with additional sensors.

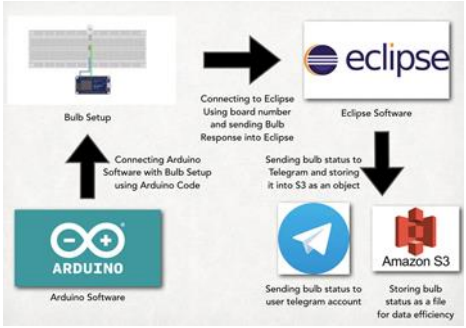

Figure 1. Smart home architecture

III. Problem Statement

IoT has gained a lot of popularity in recent times. We in our previous paper have created a demo environment for Smart Home Implementation and sent the smart device notifications to telegram account of the user. In this our problem statement is to simplify the storage of smart device status by storing the status into AWS cloud platform S3 bucket as an object. AWS platform is scalable and efficient to use, in this way we can store the vast IoT data in Cloud and handle the data in a better and efficient way.

IV. Implementation

In the context of Java Eclipse, we can monitor the operational state of a light bulb and transmit this information to the cloud for storage. Initially, we set up an Amazon Web Services (AWS) S3 bucket to act as our data repository. Subsequently, we establish an AWS user account and grant it the necessary permissions, specifically "AWSS3FULLACCESS," to enable interactions with the S3 service.

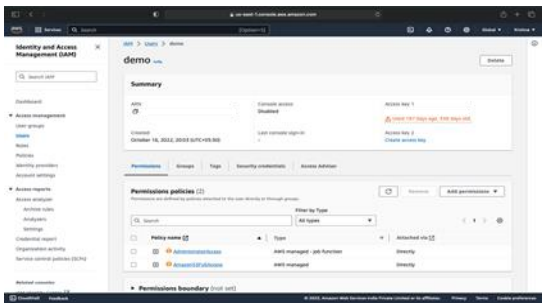

Figure 2. Creating IAM User in AWS

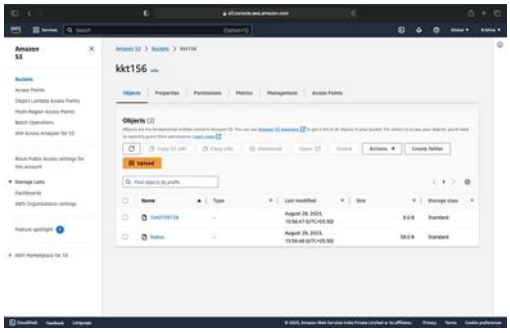

Figure 3. Creating S3 Bucket in AWS

Next, we configure this AWS user account on our local development environment by providing the user's authentication credentials. Within the Eclipse integrated development environment (IDE), we craft a Java program responsible for generating a file that contains the current status of the light bulb. This program requires inputs, including the data to be stored, the desired file name, and the name of the S3 bucket along with its designated region.

When the light bulb is operational, the Eclipse code executes, generating a file that encapsulates the bulb's status. Upon completion of this process, the file is promptly transferred and stored directly within the specified S3 bucket. Retrieving the data stored in the S3 bucket involves downloading the file. Upon inspection, we can confirm that the status of the light bulb has been successfully preserved within the S3 bucket as an object.

This methodology presents a highly efficient way to manage data for Internet of Things (IoT) devices, delivering a dependable and adaptable framework for storing and retrieving data in cloud-based environments.

By integrating AWS S3 storage with IoT device management, we achieve a streamlined and organized approach to handling data. This means that data generated by IoT devices, such as the operational status of a light bulb in our scenario, can be seamlessly stored, managed, and accessed through cloud infrastructure. The reliability aspect ensures that data is securely preserved without the risk of loss or corruption. AWS S3 offers robust data durability and redundancy, minimizing the chances of data loss due to hardware failures or unexpected issues.

Furthermore, the scalability of this approach allows us to accommodate a growing volume of data effortlessly. As IoT ecosystems expand and more devices come online, the cloud-based storage system can adapt to handle increasing data loads without requiring extensive modifications or infrastructure overhauls.

Overall, this approach empowers organizations to efficiently harness the potential of IoT devices, enabling them to collect, store, and utilize data effectively while benefiting from the reliability and scalability of cloud-based solutions. It not only simplifies data management but also lays a strong foundation for data-driven insights and applications in the IoT space.

4.1. Algorithm for Storing Status Sf Bulb into S3

1. Append the 'status' parameter to the 'ddd' variable
2. Create an Amazon S3 client
3. Define the S3 bucket name and object key
4. Convert the 'ddd' string to bytes using UTF-8 encoding
5. Create metadata for the S3 object
6. Create a PutObjectRequest with the S3 bucket, object key, content, and metadata
7. Upload the object to Amazon S3
8. Print a success message
9. Helper function to convert a string to bytes
10. Helper function to create an input stream from byte array

The provided algorithm accomplishes the task of uploading data to Amazon S3 (Simple Storage Service). It starts by appending the 'status' parameter to an existing string variable named 'ddd,' effectively adding new content to it. Next, it establishes an Amazon S3 client, which is essential for interacting with Amazon S3 services. The code then specifies the S3 bucket name and object key, defining where the data will be stored within the chosen S3 bucket. To prepare the data for upload, it converts the 'ddd' string into a byte array using the UTF-8-character encoding. Metadata for the S3 object is created, including information about the content length. The 'PutObjectRequest' is constructed, encapsulating the necessary details for the upload, such as the bucket name, object key, content in byte array format, and metadata. The object is subsequently uploaded to Amazon S3, and a success message is printed to the console. Additionally, two helper functions are defined—one to convert a string to bytes using UTF-8 encoding and another to create an input stream from a byte array. These functions aid in the data conversion and streaming required for the upload process.

4.2. Pseudo Code for Storing Status of Bulb Into S3

Input:

- `status` (string) - The data to be uploaded.
- `bucketName` (string) - The name of the Amazon S3 bucket where the data will be stored.
- `objectKey` (string) - The object key, which is the path within the bucket where the data will be saved.

Output: Data Stored Successfully into S3

1. Initialize an empty string variable `ddd`.
2. Append the `status` parameter to the `ddd` variable.
3. Initialize an Amazon S3 client (`s3Client`) to interact with Amazon S3 services.
4. Convert the `ddd` string into a byte array `contentBytes` using the UTF-8-character encoding.
5. Create metadata for the S3 object:
   - Set the content length of `metadata` to the length of `contentBytes`.
6. Create a `PutObjectRequest` to specify the upload details:
   - Initialize a `PutObjectRequest` object (`request`) with the following parameters:
   - bucketName`: Set to the provided `bucketName`.
   - `objectKey`: Set to the provided `objectKey`.
   - `content`: Create an input stream from `contentBytes` and associate it with the request.
   - `metadata`: Attach the previously defined `metadata` to the request.
7. Upload the object to Amazon S3 using the S3 client by executing `s3Client.putObject(request)`.
8. Print a success message to indicate that the object was uploaded successfully.
9. End the procedure.

4.3. Flow Chart for the Implementation

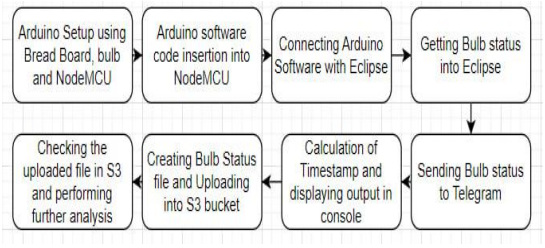

Figure 4. flow chart

To begin, let's establish an intelligent home system using a breadboard, a light bulb, and a NodeMCU board. We'll proceed by crafting a program that facilitates connectivity with these devices through Wi-Fi and enables control via the BlynkIOT app. Subsequently, we'll embed this program into the devices via the Arduino software. Following that, we'll retrieve the device's operational status and feed it into an Arduino Spreadsheet. This spreadsheet will be linked to the Eclipse platform to obtain the bulb's status within Eclipse's code. In Eclipse, we'll author a program that establishes a connection with the Arduino software to retrieve the bulb's status. We'll then compose a script to share the bulb's status via the Telegram messaging platform. After the status has been shared, our next task involves generating a file to store this status and storing it as an object within the Amazon S3 cloud storage service.

V. Results and Discussions

After the Eclipse Code successfully executes, it creates a file that directly finds its place in an AWS S3 bucket, serving as an efficient repository for the bulb's operational status. This integration is pivotal, ensuring that the data reflecting the bulb's performance is not only logged but also readily accessible for analysis and monitoring. The Eclipse console provides real-time feedback on the bulb's status, aiding in immediate troubleshooting and monitoring.

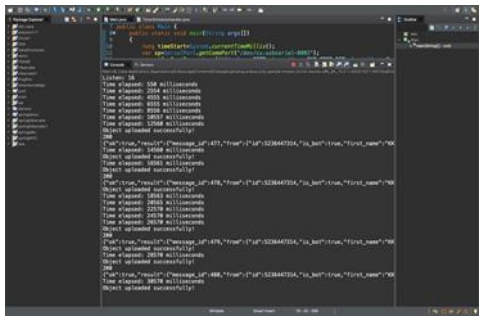

Figure 5. Output in the console

Within the output console, we obtain critical information, including the elapsed time, calculated as the disparity between the present moment and the initiation time of the execution process. Additionally, we display the listening port number, signifying the specific communication endpoint where incoming requests are being received.

An acknowledgment of the successful S3 upload reinforces data security and durability. What sets this system apart is its continuous monitoring capability; as

long as the bulb operates, the Eclipse code keeps updating the status file, offering a valuable historical record.

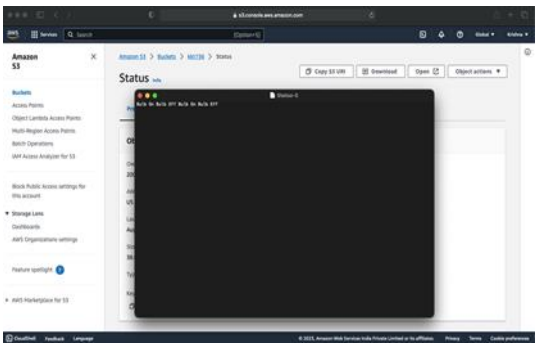

Figure 6. File Stored in S3 as an object

The S3 bucket acts as a central hub for these files, providing high availability, durability, and controlled access.

Retrieving 'Status' files can be done programmatically or manually, allowing for historical analysis, reporting, and performance optimization. In essence, this system ensures a seamless flow from status capture to storage and retrieval, empowering users with valuable insights into the bulb's behavior.

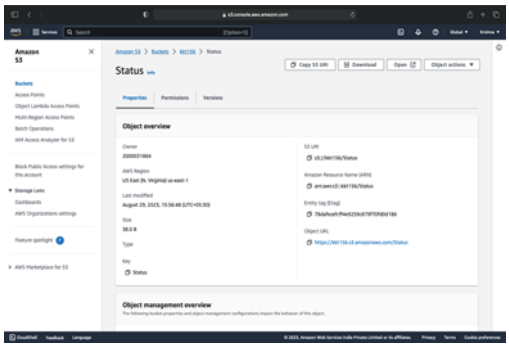

Figure 7. File Downloaded and Verify Status of the bulb

Vi. Conclusion

In this paper, the authors tackle the challenge of efficiently managing and storing smart device status data in the context of the burgeoning Internet of Things (IoT) ecosystem. With IoT gaining significant traction, their previous work laid the foundation by creating a demonstration environment for Smart Home Implementation and transmitting smart device notifications to users via Telegram. However, the primary focus of this paper revolves around simplifying the storage of smart device status data. To achieve this, the authors propose leveraging the AWS cloud platform's S3 bucket as a robust and scalable storage solution. AWS is renowned for its efficiency, making it an ideal choice for handling the vast amounts of data generated by IoT devices.

The implementation detailed in the paper unfolds within the Java Eclipse environment. It effectively monitors the operational status of a light bulb and then securely transmits this crucial information to an AWS S3 bucket. The initial setup involves creating an AWS S3 bucket to serve as the data repository and establishing an AWS user account with the necessary permissions. Subsequently, a Java program is crafted within the Eclipse integrated development environment. This program is responsible for generating a file encapsulating the current status of the light bulb. The process includes specifying the data to be stored, the desired file name, and the target S3 bucket's name and region. When the light bulb operates, the Eclipse code executes, generating and promptly transferring the status file to the designated S3 bucket. This method proves highly efficient for IoT data management, providing a dependable and adaptable framework for cloud-based data storage and retrieval.

The key advantages of this approach are noteworthy. Firstly, it streamlines IoT data management, simplifying the often complex task of handling data generated by smart devices. Secondly, the integration with AWS S3 ensures data reliability and security, minimizing the risks of data loss due to hardware failures or unforeseen issues. Additionally, the system's scalability means it can seamlessly accommodate the increasing data loads associated with expanding IoT ecosystems. In practice, this approach empowers organizations to harness the full potential of IoT devices. It not only simplifies data management but also lays a strong foundation for data-driven insights and applications within the IoT realm. The authors have also provided detailed algorithms and pseudo-code, enhancing the paper's practicality and utility for those looking to implement a similar solution. Ultimately, this paper serves as a valuable contribution to the field, presenting a robust framework for efficiently managing and utilizing IoT data in cloud-based environments.

VII. Future Work

Our future plans involve extending our project to conduct a thorough analysis of device statuses, which are currently stored within an AWS S3 bucket. To achieve this, we intend to leverage the capabilities of AWS Quick Sight, a service that empowers us to create a

wide array of data visualizations, including pie charts and various other forms of data representation. This will significantly elevate the quality and precision of our data analysis and presentation efforts.

Furthermore, these well-crafted visualizations will be readily shareable with our end-users. By doing so, we aim to provide our users with a comprehensive understanding of the device usage patterns .

References

[1] Biswas, Abdur Rahim, and Raffaele Giaffreda. "IoT and cloud convergence: Opportunities and challenges." *2014 IEEE World Forum on Internet of Things (WF-IoT)*. IEEE, 2014.

[2] Benazzouz, Yazid, et al. "Sharing user IoT devices in the cloud." *2014 IEEE world forum on internet of things (WF-IoT)*. IEEE, 2014.

[3] Wang, Peipei, Daniel J. Dean, and Xiaohui Gu. "Understanding real world data corruptions in cloud systems." *2015 IEEE international conference on cloud engineering*. IEEE, 2015.

[4] Zhang, Xiao, et al. "Ensure data security in cloud storage." *2011 International Conference on Network Computing and Information Security*. Vol. 1. IEEE, 2011.

[5] Gehrmann, Christian, and Mohamed Ahmed Abdelraheem. "IoT protection through device to cloud synchronization." *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2016.

[6] Zhen Ling∗ , Junzhou Luo∗ , Yiling Xu∗ , Chao Gao† , Kui Wu‡ and Xinwen Fu† "Security Vulnerabilities of Internet of Things: A Case Study of the Smart Plug System" 2327-4662 (c) 2016 IEEE.

[7] Naregal, Keerti, and Vijay Kalmani. "Study of lightweight ABE for cloud based IoT." *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE, 2020.

[8] Soni, Dheresh, Vibhor Sharma, and Deepak Srivastava. "Optimization of security issues in adoption of cloud ecosystem." *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*. IEEE, 2019.

[9] Delsing, Jerker, et al. "Enabling IoT automation using local clouds." *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016.

[10] Derhamy, Hasan, et al. "A survey of commercial frameworks for the internet of things." *2015 ieee 20th conference on emerging technologies & factory automation (etfa)*. IEEE, 2015.

[11] Mollah, Muhammad Baqer, Md Abul Kalam Azad, and Athanasios Vasilakos. "Secure data sharing and searching at the edge of cloud-assisted internet of things." *IEEE Cloud Computing* 4.1 (2017): 34-42.

[12] Truong, Hong-Linh, and Schahram Dustdar. "Principles for engineering IoT cloud systems." *IEEE Cloud Computing* 2.2 (2015): 68-76.

[13] Mhatre, Leena, and Neha Rai. "Integration between wireless sensor and cloud." *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE, 2017.

[14] Aleisa, Mohammed, et al. "Performance analysis of two cloud-based iot implementations: Empirical study." *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE, 2020.

[15] Ferreira, Luis Lino, Michele Albano, and Jerker Delsing. "QoS-as-a-Service in the Local Cloud." *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2016.

[16] Perera, Charith, et al. "A survey on internet of things from industrial market perspective." *IEEE Access* 2 (2014): 1660-1679.

[17] Park, Sungjin, Jaemin Park, and Jisoo Oh. "Design and implementation of trusted sensing framework for IoT environment." *Journal of Communications and Networks* 23.1 (2021): 43-52.

[18] Xu, Yi, and Abdelsalam Helal. "Scalable cloud–sensor architecture for the Internet of Things." *IEEE Internet of Things Journal* 3.3 (2015): 285-298.

[19] Sai, M. Rama, et al. "Smart Home Messenger Notifications System using IoT." *2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS)*. IEEE, 2023.