Smart Light Management System for Personalized Lighting Experience

Nitin Pawar <nt.pawar@stud.fh-sm.de>, 313936
Brijesh Yadav <br.yadav@stud.fh-sm.de>, 313945
Melissa Dmello <m.dmello@stud.fh-sm.de>, 312748
Nikhil Meduri <n.meduri@stud.fh-sm.de>, 313959
Vatsalya Singhi <v.singhi@stud.fh-sm.de>, 313897

Abstract: Do you know that lighting at home or work has become smarter with nascent technology like the Internet of Things (IoT)? This IoT project is all about making those smart lights even better. We have developed a Smart Light Management System for Personalized Lighting Experience that manages lights in a smart and secure manner. Our paper aimed to achieve three main goals. First, we created a lighting solution that allows users to choose light conditions based on data collected from various sensors. Second, we developed a versatile system that can incorporate different hardware to achieve various lighting goals in the home or workplace. Third, we established a scalable and robust data collection and processing pipeline capable of handling millions of sensor readings. This robust pipeline can be integrated with other Smart Home devices as well. In essence, we are flipping the switch on lighting to illuminate a path towards healthier, more efficient homes and workspaces.
Keywords: PineCone BL602, Smart Lighting, Internet of Things (IoT), Healthcare, Big Data, Data Analysis, Data Visualization, Scalability, Kafka, MQTT.

1 Introduction

Between 2023 and 2028, it is estimated that the number of smart home devices will increase by more than 117.69%. This presents both a significant challenge and an opportunity to create Smart Home Devices that can be personalized according to the user's needs. Smart Lights are no exception in this regard. Light, being a basic necessity for humans, has seen minimal changes in its core functionality from the first commercial incandescent light bulb in 1879 to 2024. This is in contrast to other utilities, such as Smart Watches, which

have witnessed vast development over the years and made possible through the collection, processing, and creation of value from data.

"Data is the new oil of the 21st Century," and our homes are no exception. With correct data collection and analytics, a better living experience becomes possible. Proper lighting conditions are crucial for health, as issues like eye strain, headaches, low mood, poor concentration, absenteeism, and job dissatisfaction can be linked to insufficient lighting in the home or workplace. For instance, in Germany, the minimum illumination for warehouses ranges from 50 to 300 lux, based on the work area [6]. Smart Lighting Systems (SLS) come with many benefits in this regard. They surpass traditional light sources by offering enhanced lighting intensity and providing personalized lighting zones, allowing the adjustment of lighting intensity to meet the specific needs of individual workers[20].

In today's world, carbon dioxide emissions are alarmingly high, causing significant damage to our environment. Almost 20% of all power use and 6% of all CO2 emissions globally come from lights. If nothing is done now, the world's energy usage for lighting would rise by 60% by 2030 [8]. However, there is good news. We belong in an innovative period with a focus on smart homes, intelligent cities, and advanced factories. LEDs, combined with light control systems powered by advanced sensors, minimize energy usage and help improve health ¸

We are showcasing a system that makes use of smart technology such as the PineCone BL602 board in support of Wi-Fi and Bluetooth. This system facilitates data transmission, enabling smart spaces with optimum personalized lighting. This system is robust to outside interference and authorized changes. It is flexible and efficient. With the PineCone BL602 board, you can adjust the lighting to your preference and the atmosphere of the space. The integration of the PineCone BL602 controller board allows for scalability, enabling multiple PineCones to connect and integrate seamlessly with various smart home devices. Because it is open-source, users may customize their lighting experiences and it is compatible with upcoming technologies, encouraging cooperation.

In summary, the PineCone BL602 controller board, when combined with connected actuators and light sensors, creates a smart lighting system that is flexible, efficient, and economical. This technology opens the door to customized and creative linked living experiences by encouraging energy conservation, improvised control, and flexibility in managing lighting solutions.

2 Background

Let's put some light on lighting not just any lighting but the kind that makes spaces brighter, smarter, and surprisingly good for every user. Traditional lighting just brightens up places, but we are talking about smart lighting systems (SLS) here lighting that is way smarter and friendlier. These systems help sophisticated technologies like LEDs, IoT integration, and sensor networks to revolutionize how we illuminate and interact with our environments.

Imagine lights that don't just switch on and off but they adjust to the user's needs and ease of access. They have got these advanced sensors that make them respond to your presence or the sunlight outside. SLS are not just for homes and offices, these systems

work almost everywhere indoors, outdoors, and even on public streets. These lights are not just illuminating, they are part of the whole Internet of Things (IoT) scene.

Smart lighting systems have several components and IoT technology, and many studies and research projects have researched them. This study performed a thorough examination highlighting how IoT integration helps smart lighting systems optimize energy efficiency and user experience. The results of their study indicated how IoT-enabled systems have a major influence on occupancy and environment-based adaptive lighting control [11].

We are discussing health risks and advantages because the majority of research papers focus on energy efficiency. Furthermore, current study on the developments in lighting technology, particularly the move towards LEDs and digital control systems. In addition to energy savings, their work pointed at the potential for Human Centric Lighting (HCL) aligning lighting with natural circadian rhythms to positively impact user's health, mood, and overall well-being [20].

Exposure to artificial light at night has been identified in numerous studies as a negative effect on health, with increased risk for: Sleep disorders, Depression, Obesity, Diabetes, Heart disease and Cancer. As claimed by the American Medical Association, "It is estimated that white LED lamps have five times greater impact on circadian sleep rhythms than conventional street lamps. Recent large surveys found that brighter residential nighttime lighting is associated with reduced sleep times, dissatisfaction with sleep quality, excessive sleepiness, impaired daytime functioning and obesity." [17] [3] About a quarter of a million persons in the European Union suffer from light-sensitive diseases, which are exacerbated or caused by light itself. Those with light-emitting UV or blue light disorders are most impacted [4].

The introduction of smart lighting has opened up a plethora of new possibilities for adjusting one's own health. Proper colour temperature and timing can help improve concentration throughout the day and facilitate sleep patterns at night [16].

3 Literature Review

Smart technology and Internet of Things (IoT) applications have changed building management, energy efficiency, and urban infrastructure in recent years. With a focus on energy-efficient lighting systems, Internet of Things applications in smart cities, the health effects of lighting practices, indoor environmental quality, and cloud integration in IoT. These literature reviews focuses at the state of research in several important areas linked to smart technology. This review attempts to offer insights into the developments, difficulties, and possible future paths in these fields by examining a collection of existing literature. Designing environmentally friendly and resilient urban environments and maximizing resource use for improved quality of life require an understanding of the effects of smart technology and Internet of Things applications.

3.1 Internet of Things and it's applications

IoT was first proposed in 1982, but it became well-known after British entrepreneur Kevin Ashton first used the term "Internet of Things" in 1999. IoT nowadays facilitates interactions between smart devices and appliances by allowing communication between machines or between humans and machines. As a result of push protocols scalability to low-bandwidth networks and they have emerged as the primary communication method for IoT devices. Message transfer between devices is essential for system management. IoT applications can share messages more efficiently due to push protocols like MQTT, XMPP, and CoAP. MQTT is especially well-liked because of its lightweight design[35].

IoT platforms provide the infrastructure for device connectivity, data collection, and analysis, acting as the framework for IoT ecosystems. These platforms usually have three layers in their architecture: the platform, gateway, and device layers. Every layer is essential for enabling effective data processing and smooth communication. Scalability, affordability, and availability are given top priority in sustainable IoT architectures and guaranteeing the smooth integration of IoT solutions into the present system.IoT applications can be discovered in many different fields such as connected devices, smart cities, industrial automation, smart homes, and healthcare. These applications make use of servers, cloud platforms, actuators, sensors, and automation to facilitate data-driven decision-making and real-time monitoring. Particularly smartwatches and smart homes have become more well-known for their capacity to improve quality of life and increase convenience[7].

The article wrintten by et al. Sikder [34] shows that the speed of urbanization has increased greatly in the past decades. Improved services and applications are needed in urban areas to enable better lifestyles. Smart cities, which are the concept of connecting modern digital technologies in a city context, are a potential solution to improve the quality and performance of urban services. With the advent of the Internet of Things (IoT) in smart cities, new opportunities have emerged to develop new services and integrate different application areas using information technology and the media. However, to ensure seamless services in IoT-enabled smart city environments, all applications must be maintained with limited power resources [34].

3.2 Energy Efficiency and Smart Lighting Systems

The paper wrintten by et al. Modabbir and Mohammad [21] discuss how using smart lighting system and LEDs in street lights have reduced energy consumption by 96-97% by using the PIR sensor and GPS to detect drivers. Using all these data matrices the street lights are turned on/ off or the luminance of lights is adjusted for a certain length of the streets. Also, it provides an analysis of the energy consumption of different lighting systems such as Conventional LED, Solar Power LED, Part-Night LED PIR Sensor LED [21].

The research paper written by et al. Catalina Spataru [37] shows PIR (Passive Infrared Sensor) radiation motion sensors can be used to count the number of times people move through a space In addition to testing various systems for monitoring occupants, environmental factors, including air temperature and relative humidity in living area and

sleeping room or kitchen has been monitored, using HOBO-U12 devices [37].

The research paper written by RatnaKala Sithravel et al. [30] explains how the work-from-home culture has impacted the role of houses in general. The paper explains how hybrid workstyles have led to longer occupancy hours at home which in turn has increased household electricity consumption. This results from to high demand for heating, cooling, charging of electronics, and mainly lighting. This led to rising energy prices and building energy-efficient solutions a potential sector of focus. Reducing electric lighting energy consumption in residential environments is crucial for cost savings and reducing the environmental impact of the built environment [30].

3.3 Health Implications of Lighting Practices

The impact of artificial lighting on human health, particularly its role in disrupting circadian rhythms, has become a topic of growing concern. This literature "Lighting for the Human Circadian Clock. Recent Research Indicates That Lighting Has Become a Public Health Issue" review explores the existing body of knowledge on lighting practices and their potential implications for human well-being, focusing on circadian health. It gives practices about Circadian Rhythms and Light Exposure, Daytime Lighting, and Street Lighting. Also, this literature review concludes by emphasizing that lighting practices should not focus solely on visual acuity or decorative effects but should also prioritize circadian health. With mounting evidence suggesting potential health risks associated with Light At Night (LAN) exposure, the need for more research and preventive measures in lighting design and application is underscored. Acknowledging the body's need for darkness during sleep and recognizing the protective role of melatonin, the review calls for a holistic approach to lighting that aligns with circadian biology [28].

3.4 Indoor Environmental Quality and Occupant Well-being

This paper written by Yousef Al Horr et al. explores the varied relationship between sustainable building practices, indoor environmental quality (IEQ), and occupant well-being. Also, it highlights the need for ongoing monitoring of building and occupant performance during operations [14].

In benefits of smart lighting literature review explores the benefits of smart lighting systems, particularly those utilizing 100% LED bulbs or fixtures. Addressing key questions surrounding the implementation, energy conservation, indoor comfort, and practical advantages of smart lighting systems, this article aims to provide a comprehensive overview for those considering the adoption of these technologies. It also provides benefits like Cost and Energy Savings, increased Lifespan of LED Fixtures, comfortable Indoor Spaces, security Features, and Components of a Smart Building System. Also, this research highlights, that the adoption of smart lighting systems, especially those utilizing LED technology, brings multifaceted benefits ranging from energy conservation to improved indoor comfort and convenience. As the costs of implementation decrease and operational benefits increase, smart lighting emerges as a practical and effective solution for enhancing building sustainability in 2023 [1].

3.5 Security and Cloud Integration research in IoT

The security-related research paper written by Muhammad Shafiq et al.[31] provides a review of the state-of-the-art and research challenges related to detecting and preventing security attacks on IoT devices. The rapid increase in security threats targeting IoT devices, which are often vulnerable, has spurred significant research efforts, including the use of machine learning techniques to detect attacks. The main objective of this survey is to summarize recent advances and identify trends and research gaps in the field, thereby guiding future research directions [31].

4 Methodological Approach

The methodological design in Figure 1 provides users with useful insights by fetching key sensor data and metrics from PineCone BL602 using the MQTT communication protocol and then pushing it into the Kafka pipeline for analysis. To provide thorough analytics, this cluster of metric data is pooled according to the time and position of the sensors. Additionally, by querying MongoDB for specific patterns and data charts, the system also maintains the status of the lights, giving users direct and manual control over the lights through a web application. React, a popular JavaScript framework is used to implement the visualization of the data and toggle light status, producing a visually appealing and informative user interface. Furthermore, the system allows commands to be sent to PineCone at the touch of a button and executes LED or light bulb related tasks in response to predefined values, ensuring effective and responsive closed two-way operation.

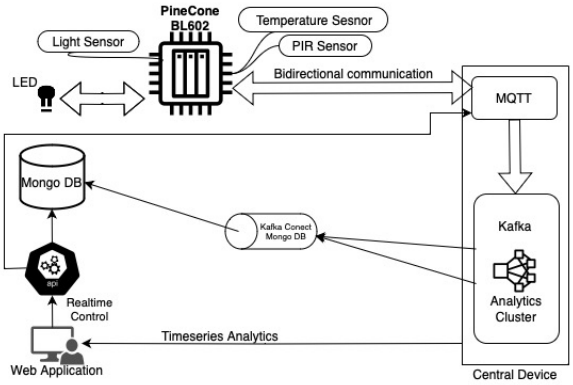

Figure 1: Smart Light System Architecture

4.1 Technology Stack

We have incorporated the following components and technology stack into the project. They have been effectively utilized to enhance its functionality.

4.1.1 Hardware

PineCone BL602 was integrated for powerful hardware capabilities.

The PineCone BL602 is a wireless microcontroller developed by the company Pine64. It utilizes the RISC-V architecture. This architecture depends on reduced instruction set computing (RISC) standards and gives a measured and extensible ISA that can be modified for explicit applications and use cases. The open source nature of RISC-V, which makes processor designs more adaptable and customizable, is one of its advantages. Adherence to RISC standards prompting further developed execution and energy productivity. The measured and extensible ISA permits new directions and usefulness to be added on a case by case basis to meet explicit application necessities.[33]

Environmental data was captured with precision using the Temperature Sensor (DHT22), light levels were effectively monitored with the Luminosity Sensor (TSL2561), and security and automation were enhanced with the PIR Sensor.

The DHT22 sensor is a low-cost humidity and temperature measuring sensor. It consists of thermister for temperature measurement and capacitive humidity sensor humidity measurement. It measures temperature within the range of -40 to +125 degree Celsius [47].

The Luminosity Sensor (TSL2561) approximates human eye response. It precisely measures illuminance in diverse lighting conditions. Some of the features include a Temperature range between -30 to 80 degree Celsius, dynamic Lux range between 0.1 - 4 0,000 Lux, voltage range between 2.7 - 3.6V [41].

The PIR sensor can distinguish animal/human movement in a prerequisite reach. A pyroelectric sensor that is capable of detecting various levels of infrared radiation makes up PIR.

The above hardware components described were combined to control and monitor a connected bulb and LED strip.

4.1.2 Infrastructure

MQTT and Kafka were used to create an adequate architecture for scalable and dependable message processing.

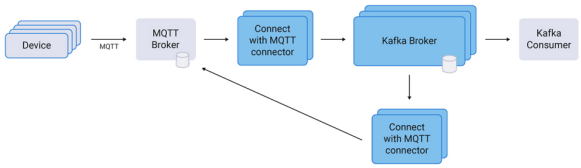

Figure 2: MQTT and KAFKA connection process[1]

MQTT: MQTT is an OASIS (Organization for the Advancement of Structured Information Standards) standard messaging protocol for the Internet of Things (IoT). In order to connect remote devices with a small code footprint and minimal network bandwidth, it

[1]Image source: `https://www.confluent.io/de-de/blog/iot-with-kafka-connect-mqtt-and-rest-proxy/`

is designed as an extremely lightweight publish/subscribe messaging service [25] . MQTT Architecture consists of 2 main components i.e. Client and Broker.

1. Client: Consists of a Publisher or a Subscriber. The client can publish messages, subscribe to subjects of interest, detach from the broker, etc.

2. Broker: A Broker's duty is to control the distribution of information, receive messages from a publisher, filter them to send the messages to the respective subscribed clients. The broker can accept client requests, receive published messages, process different requests, etc [36].

MQTT however useful as it is has certain limitations. The fact that MQTT gives a lot of configurable options can lead to transmission errors. Initiation of an MQTT connection is required every time a device wakes up. This can lead to less battery life due to the amount of data consumption [18].

Difference between of MQTT and HTTP Internet of Things (IoT) provides mechanical and electronic device control and connectivity. The two main communication protocols for Internet of Things systems are HTTP (Hyper Text Transfer Protocol) and MQTT (Message Queue Telemetry Transport)[45].

Comparison with HTTP and MQTT In Internet of Things

1. **Speed and Delivery:** MQTT supports four different quality of service (QoS) options: assured delivery, assured at least once delivery, assured exactly once delivery, and last will and testament. When compared to HTTP, MQTT shows a much higher performance, especially on 3G networks. MQTT is faster than HTTP[27].

2. **Message Size and Complexity:** MQTT reduces development for developers by providing a short specification and fewer message kinds (CONNECT, PUBLISH, SUBSCRIBE, etc.). MQTT optimizes resource use with its tiny message headers and 2-byte packet sizes. HTTP specifications are larger and it improves human readability and enables more sophisticated headers and messages[46][32].

3. **Power Utilization:** MQTT consumes less electricity than HTTP for connection maintaining connection, and sending and receiving messages. MQTT consumes less power than HTTP for connection maintenance in both high and low connection time scenarios. Thus power efficiency is better in MQTT[45].

MQTT has simpler requirements, faster efficiency, and reduced power consumption and it makes the ideal solution for Internet of Things communication. MQTT is ideal for Internet of Things applications because of its overall efficiency in power utilization and message delivery, and it could take a bit more time to establish the initial connection. Thus, based on the comparison results, MQTT is recommended for IoT communication.

Kafka: Apache Kafka is a distributed data store optimized for the ingestion and processing of streaming data in real time. A streaming data is a stream of data that has been continuously generated by thousands of sources, which typically transmit the records in parallel. Kafka is used to build real time streaming data pipelines and real time stream applications. A data pipeline is a reliable application that processes and transfers data from

one system to another, while streaming applications are an application which consumes streams of data [44].

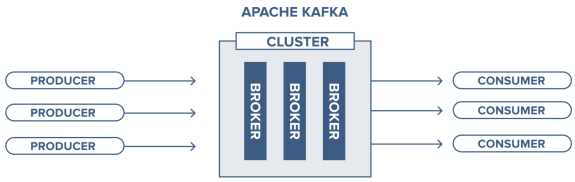

Figure 3: Apache KAFKA [2]

How does Apache Kafka work? Kafka's architecture comprises of brokers, controllers, partitions, consumers, producers, topics, schema registries and Zookeeper.

Apache Kafka Broker: A broker's the only Kafka server in existence. Kafka brokers receive messages from producers, assign them offsets, and commit the messages to disk storage. An offset is a special integer value that Kafka increments and adds to each message when it's created. In case of failure or interruption, offset is essential to maintain data consistency because consumers use offsets in order to go back to the last notification after a malfunction. The brokers are responding to calls from consumers requesting a partition, as well as return messages that have been stored on the disk. The specific hardware and its functional characteristics are the basis for a single broker.

Apache Kafka Controller: By sharing information directly or indirectly, Kafka brokers are part of the cluster. One broker is serving as Apache's controller in a Kafka cluster. The controller shall be responsible for the management of partition and replica status, as well as administration tasks that include reassigning partitions and registration entities to receive notification of changes.

Partitions in Apache Kafka: In the majority of distributed systems, partition is a fundamental principle. In Kafka, a topic is divided into multiple partitions. The partition is a separate log file. Kafka's writing records on each partition in an appendonly fashion. To put it another way, you divide and store all records relating to a given topic in separate folders.

Apache Kafka Consumers: Consumers are applications or machines that subscribe to topics and process published news feeds. Consumers read the messages in the order in which they were generated, sometimes referred to as subscribers or readers. The offset is used by the consumer to track the messages that have already been consumed. For each partition, a consumer stores an offset of the last consumed message so that it will be able to stop and restart without losing its place.

Apache Kafka Producers: The producers are the clients' applications that publish events to Kafka. Producers distribute data to topics, sometimes referred to as publishers or writers, by selecting the appropriate partition in the topic. They're allocating messages to a specific topic partition in sequence.

Apache Kafka Topics: In Kafka, the topics are specified for classifying messages. The topic is very similar to the database table or folder. A number of sections are also divided

[2]Image source: `https://www.cloudkarafka.com`

up, as mentioned earlier.

Apache Kafka Zookeeper: The metadata of Kafka brokers is stored by Zookeeper. It facilitates the exchange of information between broker and customer by means of a common central namespace for data registers called znodes, enabling distributed processes to communicate with one another.

Apache Schema Registry: The message schemas are managed in the Schema Registry and dispatched to a topic. Schemas implemented for Kafka messages are verified and maintained by the schema registry. Compatibility is also imposed prior to the addition of a message [10].

To establish a connection between MQTT and Apache Kafka a connector is used. This connector acts as a bridge between MQTT broker and Kafka. This allows MQTT messages to be read on Kafka topics. In this integration, MQTT topics are mapped to Kafka topics, and MQTT messages are converted into Kafka records, keeping the publish/subscribe model of both systems. MQTT to Kafka provides efficient capture and transmission of messages published by MQTT producers to Kafka topics through streaming pipelines. The distributed nature of the platform makes it possible to process these messages in parallel across multiple partitions and brokers once they are within Kafka.

We have used MongoDB as our database. MongoDB is a NoSQL database. Instead of depending on tables and rows in relational databases, MongoDB depends on collections and documents. Every database in MongoDB is made up of collections, which are then accompanied by documents. Depending on the number of fields, each document could be different. There may be differences in the size and content of each document. One of the plus point is that there is no need to define a schema for documents stored in MongoDB before. The fields can be created at any time [40].

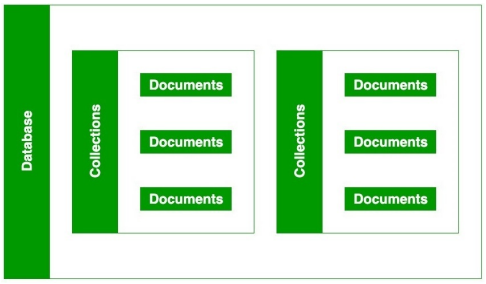

Figure 4: MongoDB Architecture [3]

We store time series data in MongoDB thanks to it's support with this functionality. A series of data points where insights can be derived from analyzing changes over time are referred to as the Time Series Data. These components are usually in the form of time series data:

Time: The time it took to record a data point.

Metadata : A label or tag that uniquely identifies a series and rarely changes.

[3]Image source: `https://www.geeksforgeeks.org/what-is-mongodb-working-and-features/`

Measurements: The data points measured in increments of time. In general, these are pairs of key values that change over time.

The underlying columnar storage format is used for the time series collections and the data is stored in time order. The following advantages are provided by this format:

- Reduced complexity for working with time series data
- Improved query efficiency
- Reduced disk usage
- Reduced I/O for read operations
- Increased WiredTiger cache usage

The collections in the time series are similar to common collections. Insert and query the data as usual. MongoDB is treating time series collections as writable, non materialized views that are backed by an internal collection. The internal collection automatically organizes time series data into an optimized storage format when you insert data [2].

MongoDB efficiently stored and handled data, while Eclipse Mosquitto served as an MQTT broker to provide safe and effective device-to-device communication.

Eclipse Mosquitto is an open-source message broker that uses the MQTT (Message Queuing Telemetry Transport) protocol. MQTT is a protocol for constrained devices with limited bandwidth, making it ideal for applications such as Machine toMachine and Internet of Things which use network bandwidth at premium.

The lightweight design of Mosquitto MQTT is one of its major characteristics. It is therefore ideal for devices with limited resources, such as sensors, microcontrollers and others connected to the Internet of Things, because it requires a minimum amount of system resources. In addition, it can work well in environments with poor or insufficient network connectivity thanks to its effective use of available capacity.

It is possible to run Mosquitto MQTT on a variety of operating systems, including Linux, Windows, macOS and even embedded computers such as Raspberry Pi. It's also compatible with Docker containers.

Bridge connections allow Mosquitto MQTT servers to communicate with other MQTT servers on the same or different network, respectively. If you need to combine data from multiple networks or if you want to connect separate networks, the Bridging feature is a useful option.

A message retention feature is provided by Mosquitto MQTT. It means that if the intended recipients are not currently connected, it can store messages that it receives. The stored messages are sent to those recipients once they have reached the Internet. This shall ensure that no data is lost, even in the case of temporarily disconnected services. This feature is especially helpful for Internet of Things applications, where devices may be disconnected or unreachable on a variety of reasons like power loss or network difficulties.

[4]Image source: `https://developer.community.boschrexroth.com/t5/Store-and-How-to/Cedalo-Eclipse-Mosquitto-MQTT-Broker/ba-p/50927`

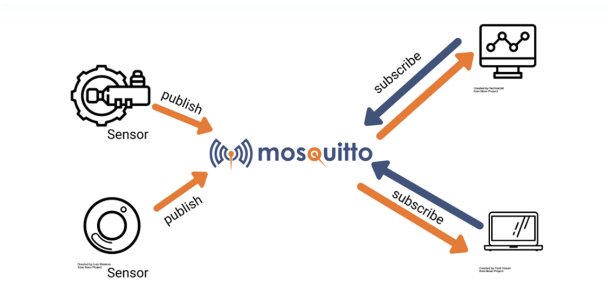

Figure 5: Mosquitto Workflow[4]

4.1.3 Protocols

Effective communication was assured by using Web Sockets protocols such as MQTT to exchange data in real-time. Additionally, Rest API's flexibility was utilized to seamlessly integrate and communicate with other components.

A REST API also called a RESTful API or RESTful web API, is an application programming interface (API) that conforms to the design principles of the representational state transfer (REST) architectural style.A flexible, lightweight way of integrating applications and connecting components to microservices architectures is provided by REST APIs [43].

An application programming interface is a set of definitions and protocols for developing and integrating applications. Sometimes referred to as a contract between an information provider and an information user that specifies the content requested by the consumer (call) and the content requested by the producer (response).

REST is not a protocol or a standard, it's a set of architectural constraints. REST may be implemented in different ways by API developers. A representation of the state of the resource is transferred to the requester or endpoint when a client request is made using the RESTful API. This representation is transferred in one of several formats via HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP, or plain text. It is because, despite its name, JSON is languageagnostic and readable by both humans and machines that it is a very popular file format in general.

The API must meet these criteria in order for it to be considered RESTful:

- A client/server architecture consisting of a client, server, and resources where requests are managed over HTTP.
- Stateless client/server communication, that is, no client information is stored between get requests, and each request is separate and unrelated.
- Cached data to optimize the client server interaction.
- A single interface for the exchange of information in a standard form between components. This requires that: Resources required shall be identified and distinct from the representations sent to a client. The client may manipulate the resources by means of a representation they receive, given that it contains sufficient information

to do so. The self descriptive messages sent back to the client contain sufficient information on how they should be processed by the customer. hypertext hypermedia is available, which means that the client should be able to use hyperlinks to find other currently available actions after accessing the resource.

- In a multi-tiered system that organizes each type of server (servers responsible for security, load balancing, etc.), the requested information must be obtained in a hierarchy that is invisible to the client.

- Code on Demand (optional): The ability to send executable code from the server to the client upon request, extending the functionality of the client. [42].

4.1.4 Server

Express JS, a quick and simple Node.js framework, powered the backend.

Express is a node js web application framework that provides features for building web and mobile applications. It's used to create a single page, multipage and web application in combination. This is a layer on top of nodejs that helps manage servers and routes.

Express is a tool to make APIs and Web Applications Simpler, It saves a lot of coding time almost by half and still makes web and mobile applications are efficient. Another reason to use express is that it's written in javascript, and if you don't know any other language, JavaScript can be used as a simple language. Express makes it easy for a lot of new developers to get into web development.

Express JS features include:

- Routing Capabilities: ExpressJS facilitates robust and flexible routing capabilities. It allows developers to map the URL endpoints into a specific function of their application. In order to maintain the state of the web application, it uses a refined URL mechanism. This simplifies the implementation of a complex routing scheme which does not contain any coding.

- Express Middleware functionality: One of the core features of ExpressJS is Middleware capability. This has had a major impact on the growing popularity of this framework. It enables developers to modify the request and response of objects before they are transmitted to other middleware features. The features, such as logging, authentication and error management, are also simplified.

- Templating Engine Support: ExpressJS will allow you to choose from a variety of templating engines, such as Handlebars, Pugs and EJS for delivering dynamic HTML pages. It also facilitates HTML pages, email templates, and much more.

- Error Handling Abilities: The robust errorhandling capabilities can be facilitated by ExpressJS. Synchronous and interrupted errors can be handled. The handling of errors is a crucial part of any web application and ExpressJS allows you to track and resolve bugs much more quickly.

- JSON Web Token (JWT) Integration: The ExpressJS framework is a convenient way to integrate with the JSON web token authentication scheme. It's got a good security feature. It is a secure and rapid way of verifying the identity of users, enabling them to take advantage of secured resources.
- Encourage Scalability and Performance: The ExpressJS framework is intended to be a flexible and rapid framework. It is well suited to the development of high performance applications, supporting low weight footprints and minimal overhead. Without crashing or slowing down the application, it can also handle a lot of requests [9].

MongoDB, a NoSQL database, was employed to easily store and retrieve data while maintaining server architectural flexibility and scalability.

4.1.5 User Interface

ReactJS skills were employed for developing dynamic user interfaces, complemented by Material UI's attractive design features. Chart.js was utilized to easily visualize data and provide an engaging and dynamic user experience.

5 Implementation

The PineCone BL602 IoT system setup phase consists of several interrelated processes, from server-side programming and database configuration to sensor integration and software initialization. The implementation phase enables the successful reality of the expected IoT system through full testing, iterative development cycles, and planning.Moreover, it shows more possibilities for real-time data management, data analysis, and data visualization.

5.1 Sensor Selection Process and Hardware Integration:

Selecting sensors that work with the PineCone BL602 requires careful consideration of several parameters. The selection procedure is significantly impacted by factors like accuracy, power needs, interface compatibility, and environmental factors. For example, the Temperature Sensor (DHT22) is flexible in a variety of voltage conditions because it can operate at both 3.3V and 5V. Its ability to produce digital bit sequences makes data processing less difficult, which makes it a great option for applications involving temperature monitoring. In contrast, the Light Sensor (TSL2561) is notable for its inbuilt analog-to-digital (ADC) and low power consumption, which enable effortless integration with microcontrollers such as the PineCone BL602 even when analog input is not available. And, due to its sensitivity to light levels, it can be used in applications that require monitoring ambient light. Motion detection is made achievable by the PIR Sensor's pyroelectric sensor, which picks up infrared radiation released by warm objects [15].

Its dual-slot design improves accuracy by identifying changes in both positive and negative differences, ensuring accurate motion detection in a range of conditions. The next step

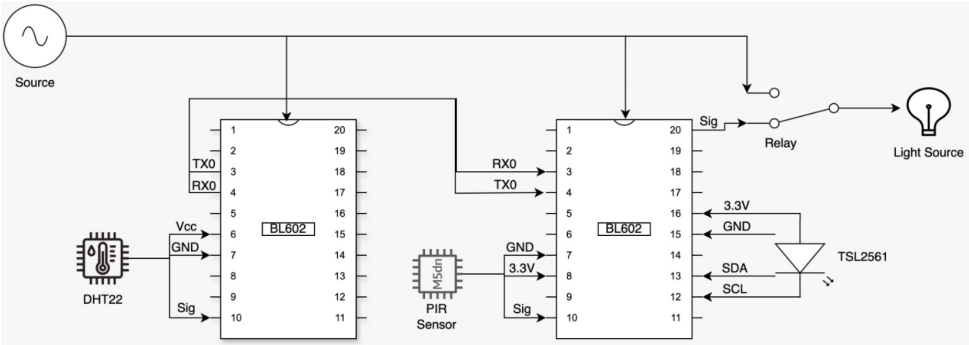

Figure 6: Hardware Integration setup for Light Management System

is to manually integrate the selected sensors with the PineCone BL602 microcontroller. Hardware compatibility, voltage levels, and pin configurations must all be carefully considered during this procedure. Specifications and schematic diagrams are necessary for ensuring accurate connections and appropriate voltage management. To improve signal integrity as well as minimize interference, factors including signal conditioning and noise reduction techniques may also be implemented [29].

1. **Sensor Initilization:** The sensors (TSL and PIR) and the relay's GPIO pins are initialized via the *sensor_init()* function. It initializes GPIO pins with their default state and configures them for input/output.

2. **MQTT Callbacks:** When the MQTT client connects to or disconnects from the broker, the callback function *mqtt_connection_cb* is triggered. The callback function *mqtt_pub_request_cb* manages publish requests. The outcome of the publish operation is printed. The callback function *mqtt_incoming_publish_cb* is triggered when incoming MQTT publish messages are received. It detects the message's subject and responds to it appropriately. The callback function for incoming MQTT data is called *mqtt_incoming_data_cb*. After parsing and processing the incoming data, it uses the message payload to control the relay.

3. **MQTT Connection:** Provides a connection with the MQTT broker using *example_do_connect*. It configures the client ID and other MQTT client data. Both the port number and IP address of the broker are provided in this connection. It establishes callback functions and subscribes to a specific MQTT topic after a successful connection.

4. **Publishing Data:** Sensor data is published to the MQTT broker by *example_publish*. Sensor readings are assembled into a JSON payload, which is then published to a selected topic. Additionally, the feature manages possible errors that may arise throughout the publishing the process.

5.2 Software Initialization and Communication between Sensors and PineCone BL602:

It's critical to ensure a seamless connection between sensors and the microcontroller when implementing an Internet of Things system with the PineCone BL602 board. This procedure includes a number of key components that ensure dependability, efficiency, and compatibility. In order to communicate with connected sensors, the PineCone BL602 microcontroller must have its software configured and its GPIO (general-purpose input/output) pins configured. Initialization procedures can involve configuring interrupt handlers, establishing sensor-specific settings, and setting up communication protocols. Error-handling procedures are also put in place to identify and minimize communication errors, providing the system executes effectively.

Optimal performance and functionality in the constantly developing field of embedded systems depend heavily on the effective coordination of tasks and resources. The C code that is provided shows the careful planning and execution that is necessary for these kinds of systems, and it makes use of the FreeRTOS framework to enable smooth functioning in limited settings. This explores the specifics of task management, network configuration, and system initialization, providing developers attempting to traverse the challenges of embedded system development with a guide. We explore the key components and features contained in the source code and providing an understanding of its usefulness in facilitating resilient and adaptable embedded systems [29].

1. **Heap Region Definition:** The *HeapRegion_t* structure is used by the code to define heap fields in order to control dynamic memory allocation. These segments define dedicated memory spaces for dynamic memory allocation which is essential for storing buffers and data structures during runtime.

2. **Static Task Stacks and Containers:** The static task containers *wifi_task* and *mqtt_task* are used to store important task data, while the static arrays wifi_stack and mqtt_stack are assigned to act as task stacks. Within the system, these structures help with effective task management and resource allocation.

3. **UART Initialization:** The initialization of the UART interface, which is essential for serial communication, involves setting up certain settings such buffer sizes, bit rates, and port numbers. The smooth communication with external devices or peripherals made possible by this configuration is essential for data sharing and system monitoring.

4. **Task Creation:** Using the *xTaskCreateStatic* method the program generates two unique tasks that are task_mqtt and task_wifi. These tasks have the ability to manage WiFi functionality and MQTT communication because they have libraries with appropriate stack pointers, priorities, and sizes.

5. **TCP/IP Stack Initialization:** The networking infrastructure required for IP-based communication is established by initializing the TCP/IP stack with the *tcpip_init* function. This crucial step allows the embedded system to operate in a networked fashion and allowing for remote access and data exchange.

6. **Scheduler Start:** Furthermore, the vTaskStartScheduler method is used to call the FreeRTOS (real-time operating system) scheduler and signaling the start of task execution and scheduling. The system responsiveness and efficient resource use are ensured by this dynamic scheduler.Also, it controls job prioritization and timing.

As a result, it sets up the components of the system, generates tasks to carry out designated responsibilities and starts the scheduler to manage task completion in an embedded real-time environment.

The FreeRTOS operating system's handles initialization and error-handling processes. These features are essential for maintaining the stability and trustworthiness of embedded systems running FreeRTOS. Every function performs an essential part in maintaining system integrity and enabling efficient operation. This includes handling assumption failures, controlling memory allocation, and reacting to stack overflows. This summary provides the foundation for a detailed knowledge of how these capabilities assist the stability of embedded applications running on FreeRTOS.

5.3 Communication Protocols and MQTT Integration:

The communication protocols implemented significantly affect the effectiveness and performance of the Internet of Things system. Data is often transmitted between PineCone BL602 and connected sensors using UART (universal asynchronous receiver/transmitter) and Wi-Fi protocols. UART makes short-range communication between microcontrollers and sensors possible, which offers an accurate and easy-to-use serial communication interface [26]. On the other hand, Wi-Fi provides a wireless connection and makes it possible to remotely monitor and manipulate sensor data [19]. The choice of communication protocol depends upon many factors, including but not limited to data transfer rate, power consumption, range, and application requirements. Message Queuing Telemetry Transport (MQTT) protocol configuration and optimization for the system, with a focus on the PineCone BL602 microcontroller[29].

1. **Configuring MQTT Broker Port:** The port number for the MQTT broker is normally 1883 for insecure MQTT connections or 8883 for secure connections using TLS/SSL (Transport Layer Security/Secure Sockets Layer) [39].This can be changed with the following line of code: **#define MQTT_BROKER_PORT 1883** Setting up the port on which the MQTT broker waits for incoming connections from clients including PineCone BL602 microcontroller—requires this setting. The PineCone BL602 microcontroller, along with other clients, must be connected to the port that the MQTT broker is listening on. This setup is necessary to define the port.

2. **Defining IP Address:** An IP(Internet Protocol) address is defined using four byte-parts (a,b,c,d) via the *IP4_ADDR macro*. This macro is frequently used to programmatically configure network settings or assign an IP address to a device. **#define IP4_ADDR(ipaddr,a,b,c,d)** For example, *IP4_ADDR(ipaddr,192,168,1,100)* sets the IP address 192.168.1.100 [39].

3. **Raising the Size of the MQTT Message Buffer:** The size of the buffer used to store incoming MQTT messages before processing is defined by the MQTT message buffer size *(MQTT_VAR_HEADER_BUFFER_LEN)*. It has a value of 128 bytes by default. However, this size could need to be modified to avoid fragmented messages or buffer overflows depending on how long the incoming MQTT topics and payloads are. We must edit the *mqtt_opts.h* file and modify the value of *MQTT_VAR_HEADER_BUFFER_LEN* to optimize this buffer size[39]. It is advised that the size be at least the maximum payload length+8 bytes (for adding overhead) plus the size of the longest incoming topic. This modification aids in making sure incoming messages are processed without loss or fragmentation. Also, we can use the *#ifndef* prefix command followed by #define to define the default buffer length if one isn't already defined.

   #define MQTT_VAR_HEADER_BUFFER_LEN

   for instance, A custom value is supplied by *YOUR_MESSAGE_LENGTH* is used to set the buffer length[39].

5.3.1 WiFi connectivity on a Pinecone board

This program consists of features like configuring WiFi networks, connecting to access points, and managing different WiFi-related events. The program ensures strong and dependable WiFi functionality for the Pinecone board in various networking scenarios through a rigorous dependent on events design and startup procedures [29]. This is WiFi task implementation for managing WiFi connectivity on a Pinecone board. Definitions of Functions:

- Receive the Device Tree Source (DTS) address for configuration using the *get_dts_addr* function.
- *_configure_wifi* sets up the WiFi configuration parameters initially.
- *_start_ap_wifi* initiates a WiFi network access point (AP).
- *_connect_sta_wifi* establishes an STA (Station)mode connection to a WiFi access point.
- The *event_cb_wifi_event* callback function handles WiFi events, including IP address acquisition, connection, disconnection, and initialization.
- *task_wifi* is the primary task function for managing WiFi.

wifi_conf_t, which includes configuration information such as the country code (for eg., EU for europe). Also, the WiFi module is initialized and started in the background by the function _configure_wifi(), which makes sure it is prepared for additional actions. WiFi Task: Configure up the required components GPIO (general purpose input and output), event processing, and UART (Universal Asynchronous Receiver/Transmitter). Initiates the task of WiFi firmware. Registers WiFi event classifiers for use. Responds to and manage WiFi events such as establishing a network connection or activating an access point. Carries out the WiFi task management main event cycle.

5.4 Mosquitto Integration and Data Transmission and Publication:

An MQTT (Message Queuing Telemetry Transport) protocol implementation, such as Mosquitto, is included in the system to enable smooth communication between the PineCone project internal and external devices or services [5]. For Internet of Things applications, Mosquitto provides a lightweight messaging protocol that makes data sharing between sensors, microcontrollers, and backend servers efficient. For the MQTT client to perform its tasks securely and dependably, it must be configured with the broker's address, port, and authentication credentials provided. Furthermore, callback methods are integrated to manage MQTT events, including successful connection, message receipt, and error management, thereby guaranteeing strong communication among hardware components[5]. Sensor data is read and broadcast to specific MQTT topics when the MQTT client has been established and configured. This enables real-time sensor data updates to be received by other devices or services that have subscribed to the subjects. A publish-subscribe architecture regulates data transfer over MQTT, with sensor nodes serving as publishers and central servers or applications as subscribers. Real-time sensor data monitoring and analysis are made easier by this asynchronous communication approach, which enables effective and scalable data distribution. The integration of Kafka and MQTT (Message Queuing Telemetry Transport) is crucial for the functioning of our smart lighting management system since it allows for effective data processing and transmission. In this, we analyze how to use MQTT and Kafka using a set of protocols for publishing, subscribing, and processing messages. In our system architecture, each program has a specific purpose and helps the data flow between IoT devices, the server, and the database to be seamless [29].

1. **mqtt_sub.js:** This program subscribes to topics on MQTT. When messages are received on topics it has subscribed then it waits for them and responds. Upon receiving a message, the program records the topic and sends it to Kafka for additional processing. It essentially acts as a link between Kafka, which processes and analyzes data and MQTT which transmits sensor data. To publish the message to Kafka, it invokes the *'produceKafkaMessage'* function.

2. **mqtt_pub.js:** It is responsible for the of data publication to MQTT topics. It receives payload data, probably comprising sensor measurements or details about the status of the system. After that, this data is published to the specified MQTT topic in JSON format. Any errors that happen during publishing are recorded to help with debugging and troubleshooting.

3. **kafka_pub.js:** It is concerned with the creation of messages associated with Kafka topics. It connects to the Kafka broker, which acts as the main hub for exchanging data. After connecting, the program uses compression parameters to maximize the effectiveness of data transfer before sending the message to the specified Kafka topic. The program disconnects from the Kafka broker once the message has been sent successfully and guaranteeing efficient use of available resources and appropriate connection handling.

4. **kafka_sub.js:** A Kafka consumer is initialized as a result. This consumer publishes messages to be received by producers such as *'mqtt_sub.js'* by subscribing to Kafka topics. The program receives a batch of signals and processes each one in keeping with the system specifications. Tasks including parsing message data, executing required calculations or transformations, and possibly saving the processed data in a database or starting other processes are all handled by it.

The continuous data flow within the smart lighting management system works cooperatively. Kafka ensures excellent data processing, analysis, and distribution and it allowing the system to react effectively to changing environmental conditions and user inputs. MQTT manages real-time data transmission from sensors to the system.

5.5 API Endpoints Functionality:

The Express server acts as the brain and nervous system of the Internet of Things, offering a centralized platform for sensor data management and enabling smooth connection with internal and third-party apps and services. The Express server exposes API endpoints that allow users to communicate with the system, receive sensor data, and carry out different tasks like data aggregation, visualization, and data analysis. The Express server's API endpoints support many different functions, such as data retrieval, aggregation, and visualization. Endpoints provide scalability and flexibility to meet a wide range of user requirements. Endpoints under the "iot_dumps" collection, for instance, enable users to collect recent entries within a given time range, obtain lists of unique device names and place IDs, and retrieve all sensor data. Also, endpoints included in the "time series" collection contain features designed specifically for examining time-series sensor data, like retrieving aggregated information based on device, sensor type, or time interval [29].

5.5.1 Data Aggregation:

The Express server uses powerful data aggregation methods to efficiently evaluate and compile sensor data. Averaging, adding, or counting sensor readings over predefined intervals of time or geographic areas are examples of aggregation processes. Users may make well-informed choices and execute the necessary activities by using aggregated data, which offers insightful information about trends, patterns, and problems.

5.6 Integration with React UI:

The IoT system becomes more interactive and user-friendly when the Express server is integrated with a React user interface (UI). Users can view sensor data in real-time with React components like charts, graphs, and data tables, which helps with data-driven analysis and decision-making. Users can analyze sensor data, personalize visualizations, and carry out interactive data analysis tasks with the help of a user-friendly and clear UI [29].

1. **MQTT pub code:** This function aims to publish the LED status to the designated MQTT topic. It accepts a value parameter that indicates the LED's status, most

likely as on or off. Within the operation: The value is published to the designated MQTT subject by calling the *client.publish()* method. A success message indicating that the message has been published is logged to the console if there are no issues while publishing. The value is sent to the *setLedStatus()* function, which is usually invoked to update the LED status on the client side.

2. **Heat index code:** This function determines the heat index by taking the humidity and temperature in Celsius. It requires two inputs: humidity and temperature in Celsius. Within the operation: The *Number()* function is used to convert the input temperature (Celsius) and humidity to numbers[13][12]. The formula **(temperatureCelsius*9/5)+32** is used to convert the temperature from Celsius to Fahrenheit, most likely to update the LED status on the client side. The function returns the actual temperature in Celsius when the humidity or temperature is below 40% or 80°F. Because these conditions do not meet those needed to calculate the heat index [13][12].

   If not, it uses the given Fahrenheit formula to calculate the heat index and uses **(heatIndexFahrenheit-32)*5/9** to convert the result back to Celsius. The value of the computed heat index is returned and rounded to the closest integer.

These functions will probably come into play in an Internet of Things application where environmental data (temperature and humidity) and LED status can be monitored and may be modified remotely over MQTT communication. Thermal comfort can be estimated based on the environment with the help of the heat index calculation.

5.6.1 Handling JSON Payloads from MQTT to Kafka

JSON payloads from MQTT topics are handled by the system before being sent to Kafka for further processing. Essential data including device identification, sensor readings, and environmental information are contained in these payloads. Sensor data may be easily collected and analyzed within the system due to the payload structure[29].

Field	Description
device_id	Unique identifier for the IoT device
device_name	Name of the IoT sensor device
place_id	Location identifier where the sensor is deployed
date	Date of the sensor reading in ISO 8601 format
timestamp	Timestamp representing the time of the sensor reading
payload	Object containing sensor data including temperature, humidity, LED status, luminosity, and proximity
temperature	Temperature reading in degrees Celsius
humidity	Humidity level as a percentage
led_status	Status of the LED (ON/OFF/AUTO)
luminosity	Luminosity level measured in lux
proximity	Proximity status (true/false)

Table 1: JSON Payloads description from MQTT to Kafka

MQTT sample message:

```
{"device_id":"bl602_alpha","device_name":"iot_sensor_123",
"place_id":"defaultplace","date":"2024-01-22T3:47:42.440Z",
"timestamp":"1234567890","payload":{"temperature":23.45,"humidity":32,
"led_status":%d,"luminosity":%d,"proximity":%d}}
```

5.6.2 Heat Index Payload

Our system includes a heat index payload in addition to raw sensor readings to give detailed environmental conditions. The heat index is generated using temperature and humidity data and provides important details about experienced temperature and related safety precautions. This payload includes color codes and descriptive language that indicate possible threats or safety precautions in addition to the calculated heat index value. These components improve situational awareness and help with proactive decision-making in response to changing environmental conditions[13][12].

Field	Description
value	The calculated heat index value (e.g., "30")
text	A descriptive text indicating precautionary measures (e.g., "No suspected precautions necessary.")
colorCode	Color code representing the severity or category of the heat index

Table 2: Heat index field payloads description

5.6.3 Meaning of LED_STATUS Enum

The LED_STATUS enum augments our system's functionality by standardizing LED control states, thereby enabling consistent interpretation and manipulation of LED behavior. Enumerating three distinct states—ON, OFF, and AUTO—this enum assigns numeric values to each state, facilitating seamless integration and interoperability within our IoT ecosystem. By adhering to a standardized representation of LED status, our system ensures uniformity and coherence in LED control logic, enhancing system reliability and ease of maintenance[29].

The different modes for managing an LED are defined by the LED_STATUS enum. Every state has a corresponding numerical value:

ON (0): Denotes the presence of an illuminated LED. **OFF (1):** Shows that the light emitter is off. **AUTO (2):** Denotes the LED's automatic control mode.

These numerical values offer a uniform depiction of LED statuses and making system interpretation and manipulation easy.

5.6.4 Performance Optimization:

The Express server's scalability, responsiveness, and dependability are ensured by performance optimization algorithms. Asynchronous processing, load balancing, paging, and caching are some of the techniques used to improve system performance under different

usage scenarios. By reducing latency, increasing throughput, and optimizing resource usage, these strategies guarantee that the Internet of Things system operates flawlessly even when there are significant volumes of traffic.

5.7 Data Collection, Storage and Data retention in MongoDB:

The main database used to store sensor data for both general and time-series collections is MongoDB. MongoDB's scalability, extensive query language, flexible schema, and indexing features make it an excellent choice for effectively managing massive amounts of sensor data[23][22][24]. MongoDB utilizes data retention mechanisms to efficiently manage storage space and adhere to legal requirements. To guarantee effective data storage and retrieval, time-based retention regulations, data aging, and partitioning techniques are utilized. These strategies save storage costs and maximize database performance by automatically deleting old data and archiving historical data [23][22][24].

We primarily focused on middleware configuration, router path definition, security protocols, and developing instance API calls to communicate with a MongoDB database [29].

1. **Security Measures:** Various HTTP headers are set by helmet middleware to protect the application from widespread online threats.By removing the default *X-Powered-By* header, *app.disable('x-powered-by')* improves security by minimizing information leakage. *rateLimit* middleware prevents misuse or denial-of-service attacks by restricting the number of requests from a single IP address within a given time frame.

2. **Middleware Setup:** *app.use(express.json())* parses are JSON requests for sending the data. Cross-Origin Resource Sharing (CORS) headers are configured by *app.use(cors())* to allow requests from designated origins. It indicates that CORS is enabled for all origins.

3. **Router Paths:** *app.use()* is used to define independent router pathways for handling different API endpoints. A particular resource or feature, such as IoT dumps, temperature, humidity, proximity, luminance, LDR (light-dependent resistor), and LED status falls under the control of each router.

4. **Sample API Calls to MongoDB Database:** There are multiple developed GET endpoints for retrieving data from the MongoDB database. An additional query parameter limit can be utilized to limit the number of results returned when retrieving data from the collection via the / endpoint. A list of the different device names included in the database collection can be obtained via the */device_name_list* endpoint. The */place_id_list* API returns a list of each distinct location ID identified in the database collection.

5.8 Schema Design and Shared Key Selection:

MongoDB collections have carefully designed schemas to support the variety of data types and attributes related to sensor readings. Individual sensor data points, including

timestamped readings and metadata like device ID and sensor type, are represented by documents in the "iot_dumps" collection. In comparison, records in the "time series" collection show combined sensor data collected over specific periods of time making time-series data querying and analysis more effective [23][22][24]. Sharding is used to increase scalability and performance by distributing data among several nodes. To achieve uniform data distribution, reduce hotspots, and maximize query performance, the right shard key must be specified. When choosing a shard key, various factors are taken into account, including data distribution patterns, access patterns, and query needs. Moreover, cluster monitoring is done using management and monitoring technologies.

5.9 Overcoming Challenges in Implementing an Efficient IoT Data Management System

Implementing an efficient IoT information management system usually involves overcoming a number of challenges. In this paper, we highlight the main difficulties that develop during the development process and the approaches used to overcome them. Every task had a different set of challenges that needed to be carefully considered and solved, such as implementing up appropriate topic names, configuring MQTT over Websockets, and running MQTT and Kafka clients locally.

1. Running MQTT and Kafka Clients Locally: The specifics of running MQTT and Kafka clients locally were studied in great detail utilizing a variety of documentation sources. To make sure that both clients were configured and set up precisely, a variety of documentation sources were reviewed.

2. Finding a Suitable Database Solution: MongoDB Timeseries was found to be the best option for managing billions of IoT sensor data with low latency and lightning-fast querying after a thorough lot of research. The database's features and compatibility for the project's requirements were taken into consideration when making the decision.

3. Building Kafka Pipelines for Data Aggregation: Sensor readings were combined using Kafka pipelines and then stored in the MongoDB Timeseries database. In order to maximize bulk storage and effectively use sensor data packets, configuration settings were adjusted.

4. Integrating MQTT Client Side: As a solution for web browser limitations on direct MQTT calls, MQTT through Websockets was implemented. Using the Postman client application, the integration procedure was verified, and port configuration was improved via trial and error.

5. Configuring MQTT Broker for Websockets: For support configuring the MQTT broker to handle and accept MQTT over Websockets, the documentation, GitHub, and Stack Overflow forums have been reviewed. Sufficient modifications were implemented to provide smooth communication between Websockets and the MQTT broker.

6 Results

1. The device-based implementation includes a cohesive integration of sensors with the PineCone BL602 microcontroller. We have been successful in achieving secure and effortless communication between the sensors and the microcontroller. MQTT helps achieve secure data communication. The buffer implementation helps achieve data reliability. I2C sensor porting on the SDK provides a compatible data exchange.

2. We have included 2 modes (manual and auto) in this system to provide the user the liberty of selecting the way he would like the system to function. This improves the user experience in such a way that he has the assurance of the system taking control over the different aspects of maintaining an energy-efficient home eco-system while the user is away.

3. With the help of the luminosity sensor, we detect the optimum level for human comfort and adjust the lights as and when needed. Detecting the luminosity has various benefits related to one's health. It promotes eye health and overall well-being. Lighting based on the luminosity readings supports the regulation of the circadian rhythm. The key to a healthy sleep pattern is to have exposure to natural light during the day and dimmer, warmer light in the evening. Our system being able to achieve to regulate such lighting standards is one of the major positive results of this project.

4. The user has the option to switch off the lights directly from the UI and control the brightness level. We have included a calculation based on the temperature and humidity data collected by the respective sensors that help in mold prevention. Mold has several health risks, it can cause allergic symptoms. Mold can also compromise the structural integrity of one's home [38].

5. With the help of the PIR sensor, if a motion is detected the light turns on automatically. In our current system it has been programmed to turn off the lights after 5 seconds if no motion is detected. This however can be changed based on user requirements.

Figure 7: UI Result Representation

6. The UI incorporates a graph that shows the various temperature levels. This helps users monitor the temperature levels in their homes. It could be used to provide better insights into the environment present in one's home. communication between the sensors and the microcontroller. MQTT helps achieve secure data communication. The buffer implementation helps achieve data reliability. I2C sensor porting on the SDK provides a compatible data exchange.

7 Future Scopes

The integration of sensors and PineCone BL602 has proved to be a strong foundation of a Smart Light Management System. In the tech world there is always room for improvements and enhancements. And so, keeping this in mind we have a couple of potential future scope ideas:

Integration of Additional Sensors: By adding additional Sensors we can expand our Smart Light Management System to a Smart Home Management System. We can also integrate systems developed by other group members to make this application broader.

Machine Learning Algorithms: Machine learning algorithms help in analyzing behavioral patterns. This aids in recognizing the user's preferences and can be used in improving and adapting the systems operations accordingly. This can optimize energy consumption and many other things based on the user's routines. This leads to not only a smart but an advanced intelligent system.

Improved Security Measures: It is always beneficial to improve and strengthen a system's security. Introducing advanced encryption techniques and multi-factor authentication can help build the system's security. Continuous monitoring of unusual activities could also be a plus point.

Voice-Activated Controls: Integration of Amazon Alexa or Google Assistant to help the user control their light is one of the future scope that can help make our system for desirable to the customers. With addition of other sensors or the systems developed by other groups integrating voice-commands to control the system would be a real convenience to the user.

Integration with Smart Appliances: Our long- term goal is to collaborate with manufacturers of smart appliances. Doing so we can create an entire smart home ecosystem.

8 Conclusion

Our implementation of the Smart Light Management System for Personalized Lighting Experience involves a coherent communication framework between sensors and the PineCone BL602. The selection of sensors using I2C protocols ensures optimum compatibility. The ability to manage and control lights in one's home will always be considered a comfort aspect. Being able to control and monitor the light management system is a technological aspect that can provide leisure and at the same time promote energy efficiency.

The integration of technologies in this project ensures a robust and secure system that can be positioned at the forefront of the IoT revolution. The ability of customization as per user requirements enables users to improve their experiences with a dynamic ecosystem. Moving further towards the future of smart living environments, our Smart Light Management System serves as an instrument to enhance our daily lives. Promoting energy efficiency, and personalized control demonstrates the great impact that innovative technology can have on molding the way we interact with our surroundings at home. In conclusion, the Smart Light Management System represents a technology that promotes a comfortable, sustainable, and advanced technological way of living.

References

[1] *3 Main Benefits of Smart Lighting*. Accessed on [2023]. URL: https://www.cencepower.com/blog-posts/3-main-benefits-of-smart-lighting#:~:text=Smart%20lighting%20allows%20for%20buildings,a%20person's%20physical%20core%20temperature.

[2] Paoletti Alessia. *Time series with MongoDB — medium.com*. [Accessed 14-11-2023]. URL: %5Curl%7Bhttps://medium.com/data-reply-it-datatech/time-series-with-mongodb-dd60f8d6acd6%7D.

[3] *AMA adopts guidance to reduce harm from high intensity street lights — ama-assn.org*. [Accessed 15-03-2024]. URL: %5Curl%7Bhttps://www.ama-assn.org/press-center/press-releases/ama-adopts-guidance-reduce-harm-high-intensity-street-lights%7D.

[4] *Artificial Light: 7. Are there potential health risks linked to artificial lights? - European Commission — ec.europa.eu*. [Accessed 11-11-2023]. URL: %5Curl%7Bhttps://ec.europa.eu/health/scientific_committees/opinions_layman/artificial-light/en/l-2/7-health-risks.htm%7D.

[5] Rachmad Atmoko, Rona Riantini, and M Hasin. "IoT real time data acquisition using MQTT protocol". In: *Journal of Physics: Conference Series* 853 (May 2017), p. 012003. DOI: 10.1088/1742-6596/853/1/012003.

[6] BAUA (Bundesanstalt für Arbeitsschutz und Arbeitsmedizin). "ASR A3.4 Beleuchtung - Technische Regeln für Arbeitsstätten". In: (2020). Accessed November, 2023. URL: https://www.baua.de/DE/Angebote/Rechtstexte-und-Technische-Regeln/Regelwerk/ASR/ASR-A3-4.html.

[7] Afrah Dawood. "Internet of Things (IoT) and its Applications: A Survey". In: *International Journal of Computer Applications* 175 (Sept. 2020), pp. 975–8887. DOI: 10.5120/ijca2020919916.

[8] UN Environment. *The rapid transition to energy efficient lighting: an integrated policy approach — unep.org*. https://www.unep.org/resources/report/rapid-transition-energy-efficient-lighting-integrated-policy-approach. [Accessed 15-11-2023].

[9] *ExpressJs Tutorial | A Comprehensive Guide on ExpressJS Framework — radixweb.com.* [Accessed 12-11-2023]. URL: %5Curl%7Bhttps://radixweb.com/introduction-to-expressjs%7D.

[10] Jerry Franklin. *Apache Kafka Architecture: What You Need to Know | Upsolver — upsolver.com.* [Accessed 14-11-2023]. URL: %5Curl%7Bhttps://www.upsolver.com/blog/apache-kafka-architecture-what-you-need-to-know#:~:text=Kafka%20is%20comprised%20of%20brokers,external%20data%20sources%20and%20sinks.%7D.

[11] Marc Füchtenhans et al. "Using Smart Lighting Systems to Reduce Energy Costs in Warehouses: A Simulation Study". In: *International Journal of Logistics Research and Applications* 26.1 (2023), pp. 77–95. DOI: 10.1080/13675567.2021.1937967.

[12] *Heat index.* Accessed on [2023]. URL: https://en.m.wikipedia.org/wiki/Heat_index.

[13] *Heat index calculator.* Accessed on [2023]. URL: https://www.calculator.net/heat-index-calculator.html.

[14] Yousef Al Horr et al. "Impact of indoor environmental quality on occupant well-being and comfort: A review of the literature". In: *International Journal of Sustainable Built Environment* 5.1 (2016), pp. 1–11. ISSN: 2212-6090. DOI: 10.1016/j.ijsbe.2016.03.006. URL: https://www.sciencedirect.com/science/article/pii/S2212609016300140.

[15] "How PIRs workd". In: (2023). URL: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work.

[16] *How smart lighting affects your health | Digital Trends — digitaltrends.com.* [Accessed 11-11-2023]. URL: %5Curl%7Bhttps://www.digitaltrends.com/home/how-smart-lighting-affects-your-health/#:~:text=The%20advent%20of%20smart%20lighting,ease%20sleep%20schedules%20at%20night.%7D.

[17] *Light pollution affects human health — darksky.org.* [Accessed 15-03-2024]. URL: %5Curl%7Bhttps://darksky.org/resources/what-is-light-pollution/effects/human-health/%7D.

[18] Amar M. *A deep dive into the pros and cons of using MQTT for IoT Applications — ellenex.com.* [Accessed 12-11-2023]. URL: %5Curl%7Bhttps://www.ellenex.com/post/a-deep-dive-into-the-pros-and-cons-of-using-mqtt-for-iot-applications%7D.

[19] Saurabh Malgaonkar et al. "Research on Wi-Fi Security Protocols". In: *International Journal of Computer Applications* 164 (Apr. 2017), pp. 30–36. DOI: 10.5120/ijca2017913601.

[20] Eric H. Grosse Marc Füchtenhans and Christoph H. Glock. "Smart lighting systems: state-of-the-art and potential applications in warehouse order picking". In: *International Journal of Production Research* 59.12 (2021). DOI: 10.1080/00207543.2021.1897177.

[21] Modabbir and Arshad Mohammad. "Energy and Economic Analysis of Smart Technologies on Street Lighting System". In: *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*. Vol. 1. 2021. DOI: 10.1109/ICACCS51430.2021.9441734.

[22] *MongoDB Documentation*. Accessed on [2023]. URL: https://www.mongodb.com/docs/atlas/.

[23] *MongoDB Manual*. Accessed on [2023]. URL: https://www.mongodb.com/docs/manual/.

[24] *MongoDB Reference*. Accessed on [2023]. URL: https://hub.docker.com/_/mongo.

[25] *MQTT - The Standard for IoT Messaging — mqtt.org*. [Accessed 12-11-2023]. URL: %5Curl%7Bhttps://mqtt.org/%7D.

[26] Umakanta Nanda and Sushant Pattnaik. "Universal Asynchronous Receiver and Transmitter (UART)". In: Jan. 2016, pp. 1–5. DOI: 10.1109/ICACCS.2016.7586376.

[27] Vamsikrishna Patchava, Hari Babu Kandala, and P Ravi Babu. "A Smart Home Automation technique with Raspberry Pi using IoT". In: *2015 International Conference on Smart Sensors and Systems (IC-SSS)* (2015), pp. 1–4. URL: https://api.semanticscholar.org/CorpusID:30911062.

[28] Stephen M. Pauley. "Lighting for the human circadian clock: recent research indicates that lighting has become a public health issue". In: *Medical Hypotheses* 63.4 (2004), pp. 588–596. DOI: 10.1016/j.mehy.2004.03.020.

[29] *Project repository - Github pinecone*. Project repository. URL: %5Curl%7Bhttps://github.com/brijesh445/Pinecone-SL%7D.

[30] Thomas Olsson RatnaKala Sithravel and Myriam Aries. "Optimizing Presence Sensing Lighting for Energy Efficiency and User Behavioral Needs in Small Swedish Homes". In: *LEUKOS* 0.0 (2023), pp. 1–19. DOI: 10.1080/15502724.2023.2198670.

[31] Muhammad Shafiq et al. "The Rise of Internet of Things: Review and Open Research Issues Related to Detection and Prevention of IoT-Based Security Attacks". In: *Applied Sciences* 10.12 (2022). DOI: 10.1155/2022/8669348. URL: https://www.mdpi.com/2076-3417/10/12/4102.

[32] Idris Shah, Faizan Malik, and Syed Ahmad. "Enhancing security in IoT based Home automation using Reed Solomon Codes". In: Mar. 2016, pp. 1639–1642. DOI: 10.1109/WiSPNET.2016.7566417.

[33] Shreyas Sharma. *RISC-V Architecture: A Comprehensive Guide to the Open-Source ISA — wevolver.com*. https://www.wevolver.com/article/risc-v-architecture-a-comprehensive-guide-to-the-open-source-isa. [Accessed November, 2023].

[34] Amit Kumar Sikder et al. "IoT-enabled Smart Lighting Systems for Smart Cities". In: Mar. 2018. DOI: 10.1109/CCWC.2018.8301744.

[35] Dipa Soni and Ashwin Makwana. "A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT)". In: Apr. 2017.

[36] Dipa Soni and Ashwin Makwana. "A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT)". In: Apr. 2017.

[37] Catalina Spataru and Stephanie Gauthier. "How to monitor people 'smartly' to help reducing energy consumption in buildings?" In: *Architectural Engineering and Design Management* 10.1-2 (2014), pp. 60–78. DOI: 10.1080/17452007.2013.837248.

[38] State of Rhode Island. "Mold Health Risks". In: *Official Website of the State of Rhode Island © 2024 Department of Health* (). URL: https://health.ri.gov/healthrisks/mold/.

[39] *Statista,Number of users of smart homes worldwide 2019-2028*. Accessed on [2023]. URL: https://www.nongnu.org/lwip/2_1_x/index.html.

[40] *Top Features of MongoDB | Board Infinity — boardinfinity.com*. [Accessed 12-11-2023]. URL: %5Curl%7Bhttps://www.boardinfinity.com/blog/top-features-of-mongodb/#:~:text=MongoDB%20stores%20all%20the%20data,document%20contains%20a%20unique%20ID.%7D.

[41] *TSL2561 Luminosity Sensor — learn.adafruit.com*. [Accessed 12-11-2023]. URL: %5Curl%7Bhttps://learn.adafruit.com/tsl2561/overview%7D.

[42] *What is a REST API? — redhat.com*. [Accessed 12-11-2023]. URL: %5Curl%7Bhttps://www.redhat.com/en/topics/api/what-is-a-rest-api%7D.

[43] *What is a REST API? | IBM — ibm.com*. [Accessed 12-11-2023]. URL: %5Curl%7Bhttps://www.ibm.com/topics/rest-apis#:~:text=the%20next%20step-,What%20is%20a%20REST%20API%3F,transfer%20(REST)%20architectural%20style.%7D.

[44] *What is Kafka? - Apache Kafka Explained - AWS — aws.amazon.com*. [Accessed 12-11-2023]. URL: %5Curl%7Bhttps://aws.amazon.com/what-is/apache-kafka/#:~:text=Apache%20Kafka%20is%20a%20distributed,the%20data%20records%20in%20simultaneously.%7D.

[45] Bharati Wukkadada et al. "Comparison with HTTP and MQTT In Internet of Things (IoT)". In: *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. 2018, pp. 249–253. DOI: 10.1109/ICIRCA.2018.8597401.

[46] Tetsuya Yokotani and Yuya Sasaki. "Comparison with HTTP and MQTT on required network resources for IoT". In: *2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*. 2016, pp. 1–6. DOI: 10.1109/ICCEREC.2016.7814989.

[47] zahidali. *Introduction to DHT22 - The Engineering Projects — theengineeringprojects.com*. [Accessed 12-11-2023]. URL: %5Curl%7Bhttps://www.theengineeringprojects.com/2019/02/introduction-to-dht22.html%7D.

Appendix

No	Reviewer Feedback	Response
1	1. Revise Abstract 2. Improve Background Section 3. Restructure Methodology 4. Remove "Research Question" Section	1. The abstract has been revised to use past tense for clarity and consistency. 2. An overview of the overall structure has been included in the background section with more details. 3. The methodology section hasn't been moved above the literature review as suggested as in the guideline provided in this course the Methodology comes after the Literature Review. 4. We have also decided to keep the Research Questions section in order to give more insights as to what our initial goals of the research were and the paper further on provides answers to these questions.
2	1. Explain LED-related tasks in Methodology 2. Mention protocols used by each sensor in Implementation 3. Indicate temperature units in UI 4. Remove redundant sentences in Results 5. Provide more detail on technique in Results	1. LED-related tasks in the methodology have been expanded upon for clarity. 2. Protocols used by each sensor have been mentioned in the implementation section. 3. Temperature units have been indicated in the UI section. 4. Redundant sentences in the results have been removed. 5. More detail on the technique has been provided in the results.

3	1. Improve clarity in Architecture 2. Add more graphical representations 3. Strengthen Conclusion and correct errors in Referencing 4. Provide details or example in Results 5. Provide Android app details	1. The system hardware and software architecture explanation has been clarified. 2. Additional graphical representations have been added in methodology and implementation section. 3. The conclusion has been updated with new results and the introduction's typos and mistakes have been corrected. 4. More details and examples have been provided in the results. 5. We have implemented web UI instead of Android app and other information is added.
4	1. Improve the sentence formation. 2. Use consistent capitalization for PineCone BL602 3. Provide brief explanation of components in technology stack 4. There is an inconsistency in capitalization 5. Provide details in result section and Remove repeated sentence in result section	1. The sentence formation has been improved. 2. Consistent capitalization for PineCone BL602 has been ensured. 3. Brief explanations of components in the technology stack have been provided. 4. Capitalization inconsistency has been addressed and more details have been provided in the result section. 5. The repeated sentence has been removed.
5	1. Inaccuracies and typographical errors in implementation and remove irrelevant Literature Review 2. Provide more detailed insights into power consumption	1. Inaccuracies have been corrected. and corrected formatting and updated section and subsection for irrelevant content of implementation. 2. Power consumption few details added which is relevent to the light management system