Quality of Service Analysis of an IoT-based Monitoring and Control System using Arduino IoT Cloud for Smart Home Applications

Sharon P. Maramis
Department of Electrical Engineering
Manado State Polytechnic

Gleysia A. J. Sondakh
Department of Electrical Engineering
Manado State Polytechnic

Ronny Katuuk
Department of Electrical Engineering
Manado State Polytechnic

Ali Ramschie
Department of Electrical Engineering
Manado State Polytechnic

Johan Makal
Department of Electrical Engineering
Manado State Polytechnic

Ventje Aror
Department of Electrical Engineering
Manado State Polytechnic

ABSTRACT
The rapid advancement of Internet of Things (IoT) technology has significantly contributed to the development of smart home systems that enable remote monitoring and control of household devices through internet connectivity. This study proposes the design and implementation of an IoT-based monitoring and control system using ESP8266 and Arduino IoT Cloud for smart home applications. The proposed system integrates various sensors and actuator modules to perform environmental monitoring and remote control of electrical devices in real time. The system is capable of monitoring temperature and humidity conditions while simultaneously controlling several household devices, including lighting units, television systems, air conditioning units, and automated door mechanisms through a cloud-based platform.

The development process was carried out using a prototyping methodology consisting of requirement analysis, system design, implementation, and testing stages. System performance was evaluated based on Quality of Service (QoS) parameters, including delay, throughput, packet loss, command execution success rate, and connection stability. Experimental results showed that the local system response time was approximately 1 second, while the Arduino IoT Cloud response ranged from 2 to 3 seconds depending on network conditions and cloud synchronization processes. The average delay obtained from the QoS analysis was 2.5 seconds, with a throughput value of 800 bps and packet loss of 0%. In addition, the system achieved a command execution success rate of 100% and maintained stable communication throughout the testing process.

The results indicate that the proposed IoT-based monitoring and control system provides reliable communication performance, efficient bandwidth utilization, and stable remote operation for smart home environments. Therefore, the integration of ESP8266 and Arduino IoT Cloud can be considered an effective solution for implementing smart home monitoring and automation systems.

Keywords
IoT, Smart Home, Arduino IoT Cloud, ESP8266, Monitoring System, Control System, QoS

1. INTRODUCTION
The rapid development of information and communication technology has significantly influenced various aspects of human life, particularly in automation and intelligent systems. One of the most transformative technologies in this domain is the Internet of Things (IoT), which enables physical devices to be interconnected and communicate through the internet. IoT facilitates real-time data exchange, allowing systems to perform monitoring, control, and decision-making processes more efficiently across various domains such as smart homes, healthcare, transportation, and industrial automation [1][2][10].
In the context of smart home systems, IoT technology has been widely adopted to enhance user comfort, security, and energy efficiency. Smart home solutions allow users to remotely control and monitor household devices through smartphones or web-based platforms. This capability not only improves convenience but also enables optimized energy consumption through intelligent control strategies [3][4][11]. For instance, studies have shown that IoT-based home automation systems can significantly reduce energy usage by integrating sensor-based monitoring and adaptive control mechanisms [12].
One of the important subsystems within a smart home is the garage. Conventional garage systems typically rely on manual operation, which may lead to inefficiencies and security vulnerabilities. Users often face challenges such as forgetting to close the garage door or lacking real-time information about the garage status when away from home. To address these issues, several studies have proposed automated garage systems using microcontrollers and sensors. However, many of these systems are limited to local control and do not fully leverage IoT capabilities for remote access and real-time monitoring [5][13].
In addition to access control, environmental monitoring plays a crucial role in smart systems. Parameters such as temperature and humidity must be maintained within optimal ranges to ensure system reliability and protect stored assets. The integration of environmental sensors into IoT systems enables continuous monitoring and data-driven decision making. Research indicates that sensor-based IoT systems can improve environmental awareness and system responsiveness in smart environments [6][14].
From an educational perspective, the integration of IoT into learning environments is increasingly essential, particularly in vocational and engineering education. One effective

pedagogical approach is Project-Based Learning (PjBL), which emphasizes experiential learning through real-world project development. PjBL has been proven to enhance students' critical thinking, collaboration, and technical skills by engaging them in practical problem-solving activities [7][15].

Recent studies have demonstrated that the application of IoT-based projects in education can significantly improve student engagement and learning outcomes. For example, IoT platforms have been successfully utilized to teach embedded systems, networking, and automation concepts in higher education environments [8][9][16]. Furthermore, integrating IoT with cloud computing and mobile applications has been shown to provide scalable and flexible learning infrastructures. Despite these advancements, there is still a lack of research that combines IoT-based smart systems with structured Project-Based Learning approaches, particularly in the development of smart garage systems. Most existing studies focus either on system implementation or educational methods separately, without integrating both aspects into a unified framework.

Therefore, this study aims to design and implement an IoT-based smart garage system that supports remote control and real-time monitoring while also serving as an effective learning medium using the Project-Based Learning approach. The proposed system is expected to contribute not only to technological innovation in smart home applications but also to the development of practical IoT competencies in educational settings.

1.1. State Of The Art

The development of IoT-based systems has been widely explored in various domains, particularly in smart home automation and monitoring systems. Several studies have demonstrated the effectiveness of IoT in enabling real-time control and monitoring of devices through internet connectivity. For instance, proposed an IoT-based system for monitoring electrical energy consumption in air conditioning equipment, which successfully provided real-time data and cost estimation through smartphone and web applications.

In the context of smart home automation, previous research has focused on integrating IoT with mobile applications to control household devices such as lighting and air conditioning systems. Studies show that such systems improve user convenience and energy efficiency by enabling remote access and automation features [3][4][11]. Furthermore, IoT architectures combined with cloud computing have been proven to enhance scalability and data accessibility in smart environments.

Research on garage automation systems has also been conducted using microcontrollers and sensors. For example, smart garage systems utilizing ultrasonic sensors and microcontrollers can automate door operations based on object detection [5]. However, many of these systems are still limited to standalone or locally controlled implementations, lacking full integration with IoT platforms for real-time remote monitoring and control [13].

From an educational perspective, the application of IoT in learning environments has been explored to improve student engagement and practical skills. The implementation of IoT-based learning systems in higher education has shown positive results in enhancing students' understanding of embedded systems and networking concepts [8][16]. Additionally, Project-Based Learning (PjBL) has been widely recognized as an effective approach for engineering education, as it promotes active learning through real-world project development [7][15].

1.2. Research Gap And Novelty

Based on the analysis of previous studies, several research gaps can be identified:

1. **Limited integration between IoT systems and learning methodologies**
   Most studies focus on technical implementation without incorporating structured educational approaches such as Project-Based Learning.
2. **Lack of comprehensive smart garage systems**
   Existing garage systems are generally limited to automation features and do not provide integrated monitoring, control, and environmental sensing in a unified IoT platform.
3. **Insufficient real-time monitoring and multi-parameter integration**
   Many systems focus on single functionalities (e.g., door control only) without combining environmental monitoring (temperature, humidity) and device control simultaneously.
4. **Limited use of IoT systems as learning media**
   Research rarely explores IoT systems as practical learning tools that enhance student competencies in vocational education.

2. METHODOLOGY

The development process of the IoT-based monitoring and control system for smart home applications is conducted through a structured methodology involving requirement analysis, system architecture design, implementation of hardware and software components, and comprehensive testing to evaluate system performance.

2.1. Block Diagram System

The development of the proposed IoT-based monitoring and control system in a smart home environment is initiated by designing a block diagram representation. This diagram provides a comprehensive illustration of the system architecture, which serves as the basis for hardware configuration and subsequent system implementation, as presented in Figure 1.

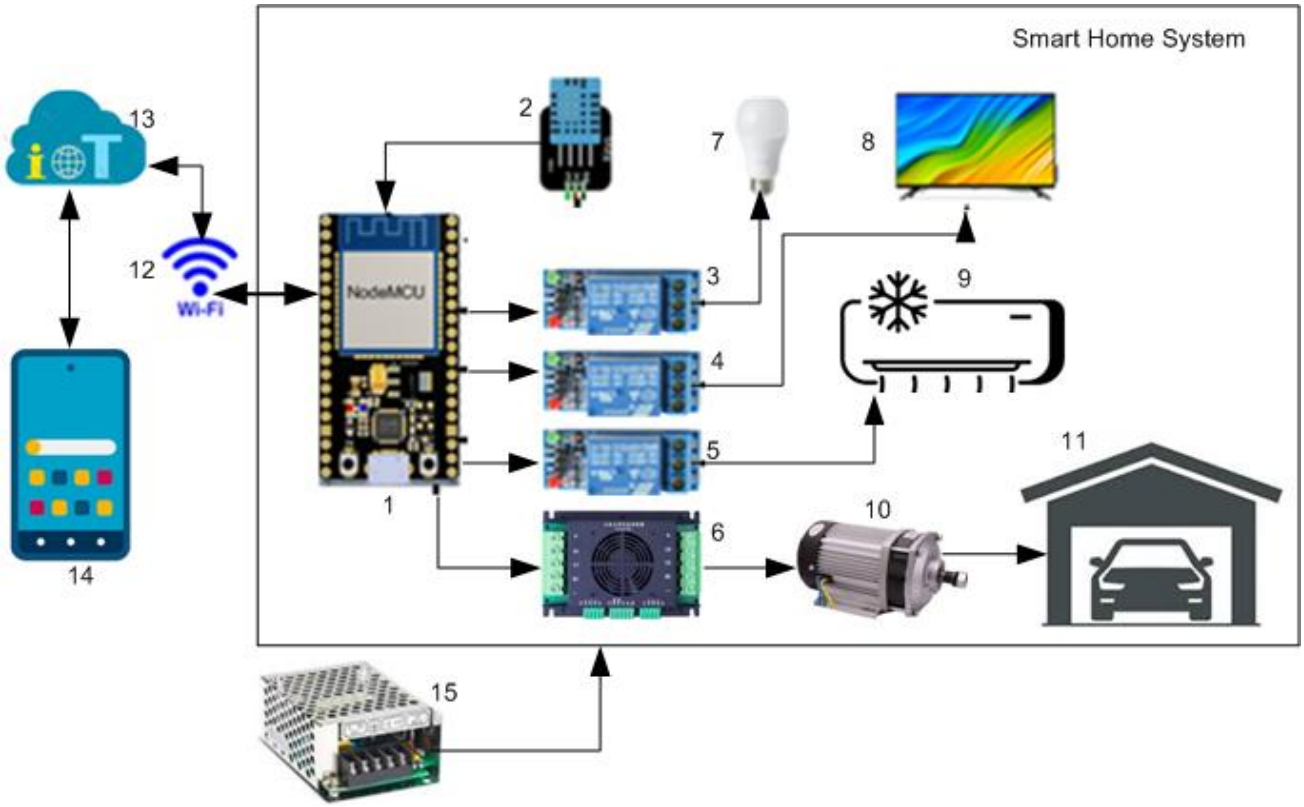

Fig 1: Block Diagram of the Proposed System

Description of the System Block Diagram:

1. The controller serves as the core processing unit of the system, responsible for acquiring and processing data for temperature and humidity monitoring, as well as managing door actuation and controlling the operation of electrical devices in the smart home environment.
2. The DHT11 sensor serves as an environmental sensing component responsible for acquiring temperature and humidity data from the surrounding area for monitoring purposes.
3. The relay driver (3, 4 and 5), serves as an electromechanical switching component that facilitates the control of electrical loads, including actuator-based operations and household electrical equipment, by interfacing low-power control signals with high-power devices.
4. The motor driver (6) serves as a control module responsible for regulating motor operation, enabling actuator-based opening and closing mechanisms within the smart home system garage.
5. The lighting unit (7) serves as an electrical load designed to provide indoor illumination and can be controlled as part of the smart home monitoring and control system.
6. The television (8) serves as a household electrical load that can be remotely controlled and integrated into the IoT-based monitoring and control system.
7. The air conditioning unit (9) serves as a controllable electrical load that regulates indoor temperature and enhances thermal comfort, and can be integrated into the IoT-based monitoring and control system.
8. The electric motor (10) acts as an electromechanical actuator responsible for generating the mechanical motion required for opening and closing mechanisms within the smart home system garage.
9. The door garage (11) serves as a physical access component within the system
10. Access Point (Wi-Fi) (12) serves as the internet communication medium between the controller and the web server for monitoring and controlling the room access system.
11. The web server (13) in this system functions as an intermediary platform that facilitates communication between the user interface and the IoT devices. It enables real-time monitoring and control by processing data received from the microcontroller and transmitting user commands back to the system through the internet.
12. The user device (14), in this case a smartphone, functions as a medium for monitoring and remotely controlling the smart home.
13. The DC power supply (15) functions as the primary source of electrical energy required to operate all components within the system. It provides a stable and regulated DC voltage to ensure proper operation of the microcontroller, sensors, and actuators.

2.2. System Algorithm

The software design of the proposed system is represented through a flowchart that illustrates the overall control workflow. The flowchart defines the sequence of operations, including data acquisition from sensors, data processing, communication with the cloud platform, and actuator control. This representation serves as the foundation for implementing the control algorithm in the microcontroller. The complete system control flowchart is depicted in Figure 2.

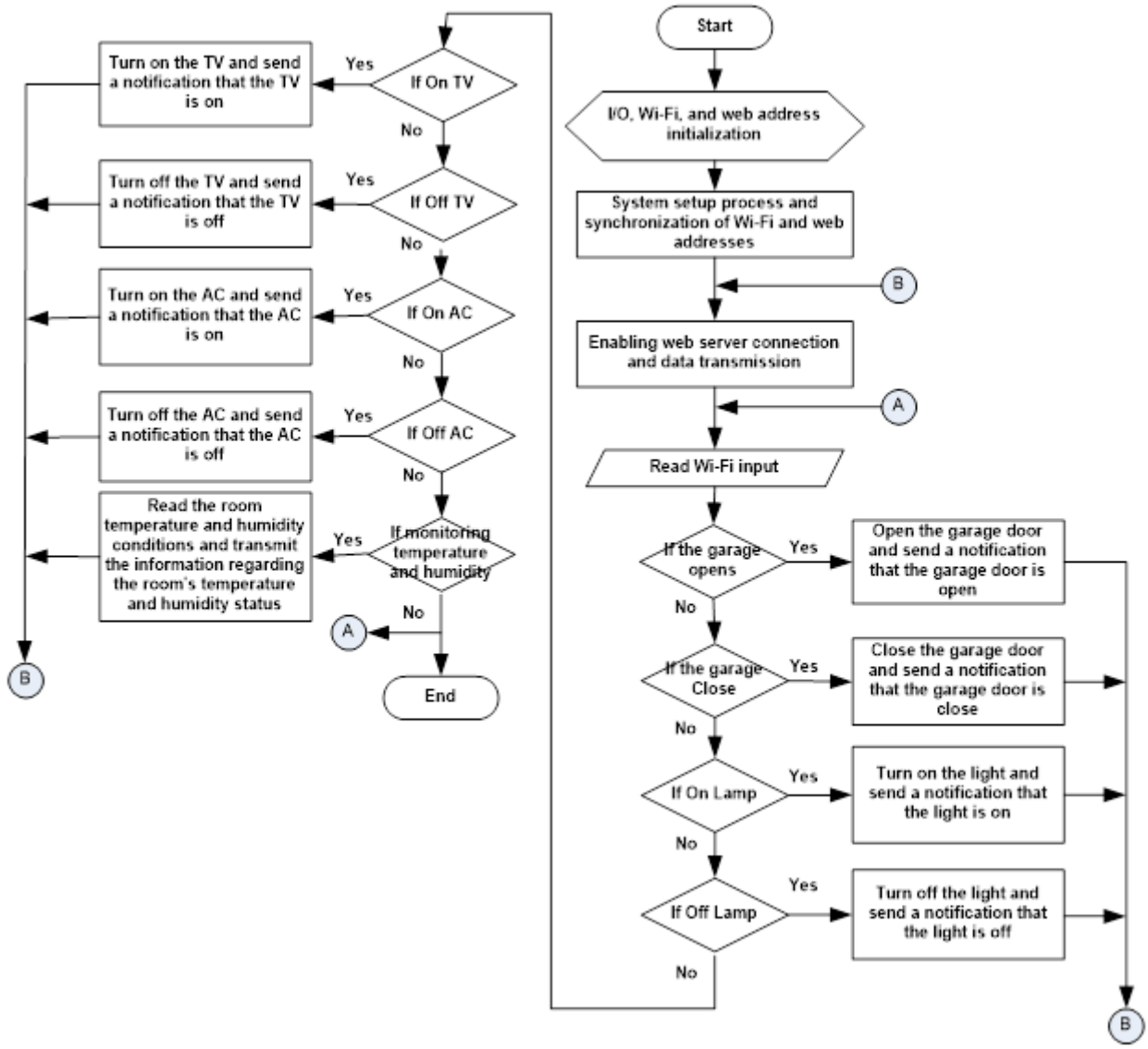

Fig 2: The Flowchart system

The Flowchart Working Procedure is as follows:

Upon initial system activation, an Input/Output (I/O) initialization phase is executed to define the controller's peripheral configuration and establish the Wi-Fi network parameters along with the designated web server address. Subsequently, the system initiates a connection between the controller and the web server via Wi-Fi communication to facilitate bidirectional data transmission. The system then retrieves user-defined data sent via a smartphone, which is analyzed and compared to determine the specific command intended for the Smart home system. The execution phase encompasses several controllable actions, including the operation of the garage door, management of lighting systems, television, and air conditioning, as well as the real-time monitoring of ambient temperature and humidity levels. The following section details the procedure for reading and comparing input data:

1. If the incoming data corresponds to the garage door activation command, the controller automatically initiates the opening sequence. Subsequently, the controller transmits a status update to the user via the web server, confirming that the garage door has been successfully opened.
2. In the event that the incoming data specifies a command to close the garage door, the controller automatically executes the closing sequence. Following this operation, the controller transmits a notification to the user via the web server, verifying that the garage door has been successfully closed.
3. If the incoming data identifies a command to activate the lighting system, the controller automatically triggers the power-on sequence. Subsequently, the controller provides feedback to the user via the web server, confirming that the lights have been successfully illuminated.
4. Should the incoming data contain a command to deactivate the lighting, the controller automatically initiates the power-off procedure. Following this, the controller transmits a status update to the user via the web server, confirming that the lights have been successfully extinguished.
5. If the incoming data corresponds to a command to engage the television, the controller automatically executes the power-activation sequence. Subsequently, the controller transmits a notification to the user via the web server,

verifying that the television set has been successfully activated.

6. In the event that the incoming data specifies a command to power down the television, the controller automatically executes the deactivation sequence. Following this, the controller transmits a status update to the user via the web server, confirming that the television has been successfully deactivated.
7. If the incoming data identifies a request to engage the air conditioning (AC) system, the controller automatically initiates the power-activation protocol. Subsequently, the controller transmits a status update to the user via the web server, confirming that the air conditioning unit has been successfully activated.
8. Should the received data specify a command to shut down the air conditioning system, the controller automatically executes the deactivation sequence. Following the completion of this process, the controller transmits a notification to the user via the web server, verifying that the air conditioning unit has been successfully powered off.
9. If the incoming data requests an environmental status update, the controller initiates the acquisition process from the temperature and humidity sensors. The system subsequently processes the raw sensor data and transmits the precise ambient conditions to the user via the web server, providing real-time information on the detected temperature levels.
10. In the event that the incoming data pertains to a request for environmental monitoring, the controller executes a data acquisition sequence from the humidity sensor. The system then processes the retrieved values and transmits the humidity levels to the user via the web server, providing an accurate report of the detected ambient moisture conditions.

This operational cycle functions continuously in a loop until the system is decommissioned or the primary power supply is disconnected. The iterative process ensures persistent monitoring and control readiness, terminating only upon the manual deactivation of the system's electrical power source.

2.3. System Manufacturing

The hardware construction was executed in accordance with the predefined block diagram design. This process utilized an IoT Trainer Kit module, involving the integration of various components into the Smart Garage system. Specifically, the DHT11 sensor was incorporated for ambient temperature and humidity detection, while a relay module served as the switching mechanism for the television and air conditioning units. Additionally, a dedicated lighting module was installed to facilitate illumination control, and a servo motor was employed to simulate the garage door's opening and closing mechanism. All aforementioned modules were integrated with a ESP8266 controller to form a cohesive Smart Home architecture. The finalized hardware prototype is illustrated in Figure 3.

Fig 3: The Prototype system

2.3.1 Software Development For Controller Operations

The software development for the Smart Garage control system, which was subsequently embedded into the ESP8266 controller, was executed based on the predefined software design and algorithmic stages outlined in the system flowchart. The development of both the firmware and the web server dashboard utilized the Arduino IoT Cloud platform. The systematic procedure for this software implementation encompasses the following stages:

- The firmware development for the Smart Home control system was conducted via the Arduino IoT Cloud platform, adhering strictly to the software design specified in the previously established flowchart. This phase involved translating the algorithmic logic into executable code to ensure that the operational requirements of the control system were met. The implementation process focused on synchronizing the cloud-based variables with the hardware's physical I/O ports based on the predefined functional flow. The results of the software development phase for the Smart Home system's operational framework are illustrated in Figure 4. This figure depicts the integrated firmware environment and the finalized logic implementation used to manage the system's automated functions.

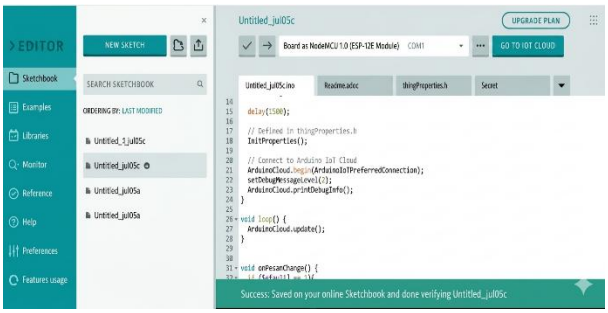

Fig 4: Software Development for controller

- Upon the completion of the programming phase, a compilation process is executed to verify the integrity of the code. This procedure serves as a critical validation step to ensure that the developed firmware adheres to the intended logic and is free from syntax or structural errors before deployment. Figure 5 illustrates the compilation process, demonstrating that the developed firmware is free from syntax errors. This successful verification confirms the integrity of the code and its readiness for deployment to the microcontroller unit. The post-compilation results, which validate the system's operational parameters, are summarized below:
  - Sketch uses 421068 bytes (40%) of program storage space. Maximum is 1044464 bytes.
  - Global variables use 36392 bytes (44%) of dynamic memory, leaving 45528 bytes for local variables. Maximum is 81920 bytes.
  - Sketch uses 421068 bytes (40%) of program storage space. Maximum is 1044464 bytes.
  - Global variables use 36392 bytes (44%) of dynamic memory, leaving 45528 bytes for local variables. Maximum is 81920 bytes.

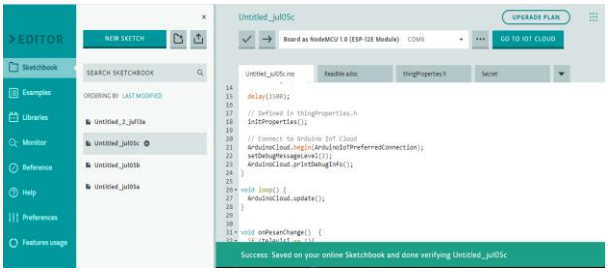

Fig 5: Process completion of the programming phase

- Once the program is validated, the subsequent phase involves embedding the firmware into the ESP8266 controller. This deployment process is executed through the following systematic steps: First, the ESP8266 is interfaced with the workstation via a USB data cable. Second, the communication port is configured, with Port 6 identified as the active interface. Finally, the upload command is initiated, and the process continues until the firmware is fully written to the controller's non-volatile memory. A 'Success' notification confirms the completion of the embedding procedure. Currently, the system is in the process of firmware deployment. Figure 6 illustrates the firmware deployment process, specifically highlighting the embedding of the operational logic into the ESP8266 controller. This visual documentation captures the data transfer phase between the workstation and the hardware, confirming that the system is successfully transitioning from the development environment to the physical implementation.

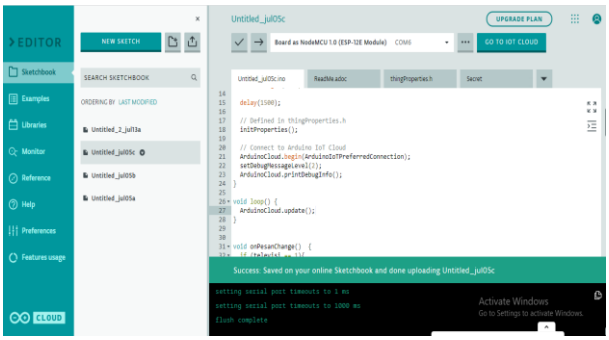

Fig 6: Process of embedding the software into the microcontroller

- Following the successful embedding of the firmware into the controller, the subsequent phase involves the configuration of a graphical user interface (GUI) dashboard via the Arduino IoT Cloud platform. This dashboard is designed to facilitate real-time monitoring and operational control of the Smart Home system. The finalized layout and functional elements of the control interface are presented in Figure 7.

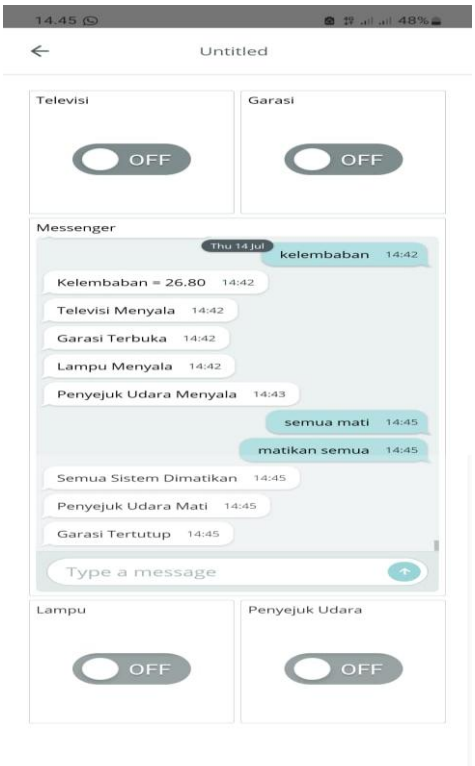

Fig 7: Web Server Development

3. RESULT AND DISCUSSION

This section details the experimental results of the proposed Internet of Things (IoT)-based smart home system, specifically designed for the remote monitoring and control of household electrical appliances. The primary objective of this evaluation is to validate the system's operational efficacy in managing these devices, ensuring both high accuracy and robust stability

under real-world conditions.

From an architectural perspective, an ESP8266 microcontroller functions as the central processing unit, orchestrating the system's overall control processes. It interfaces with peripheral control modules, notably relay drivers, to actuate key domestic loads—including lighting systems, air conditioning units, and televisions—as well as to govern the garage door mechanism. Furthermore, the microcontroller's integrated Wi-Fi capability facilitates seamless communication with the Arduino IoT Cloud platform. This integration provides users with a comprehensive, smartphone-based interface for robust remote monitoring and system control.

3.1. Functional performance evaluation of the smart home system

The operational control of the smart home system is executed through the Arduino IoT Cloud web interface. Taking the garage door mechanism as an illustrative example, the actuation process relies on binary state transmission. Initially, the dashboard toggle is in the 'OFF' position. When actuated by the user, the interface transitions to the 'ON' state and transmits a logical '1' to the ESP8266 microcontroller. Upon receiving this data, the microcontroller triggers the motor driver to actuate the motor, thereby opening the garage door. Conversely, toggling the interface back to 'OFF' transmits a logical '0' from the web server to the ESP8266. This prompts the microcontroller to reverse the motor's polarity via the motor driver, executing the door-closing sequence. Figure 8. depicts the functional evaluation workflow of the smart home system.

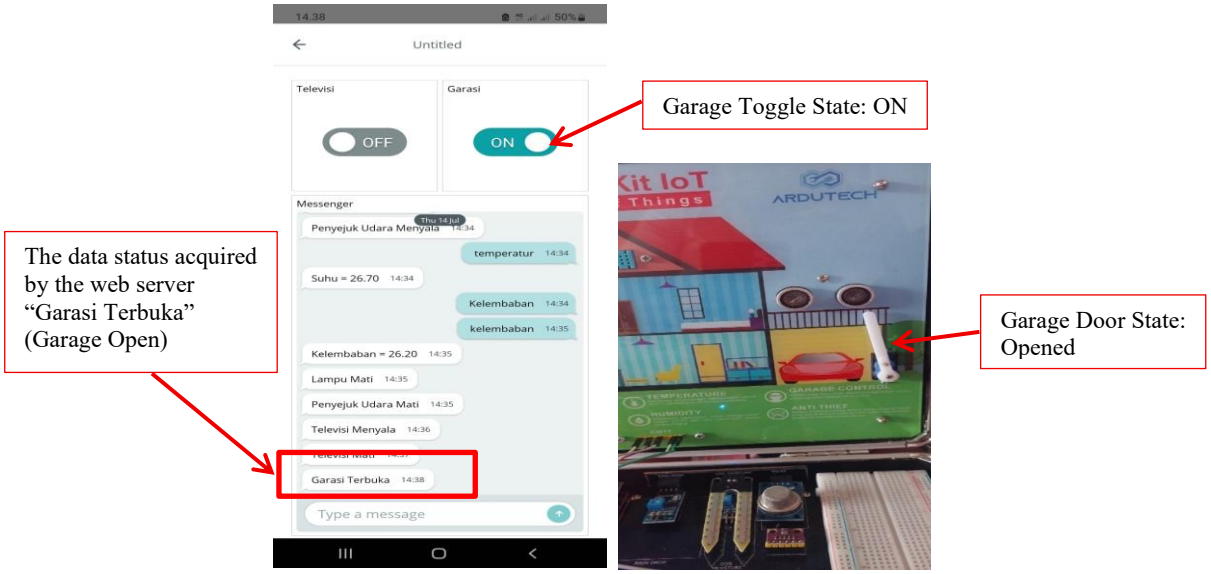

Fig 8: Depicts the functional evaluation workflow of the smart home system For Open Garage

Conversely, if the received status character is a logical '0', the microcontroller executes the closing sequence by actuating the servo motor to a 0° position. Following this physical actuation, the microcontroller transmits a feedback signal to the web server to confirm the action. This state update is subsequently registered on the server interface as a "Garage Closed" notification. Figure 9 depicts the closing sequence of the garage door.

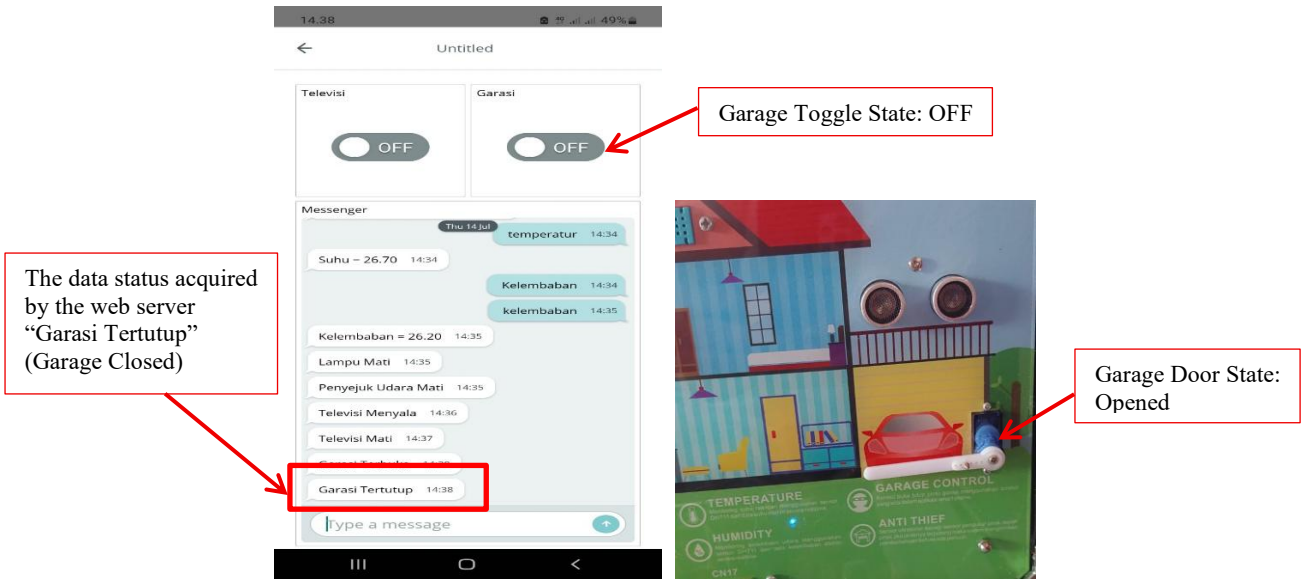

Fig 9: Depicts the functional evaluation workflow of the smart home system For Close Garage

3.2. Response Evaluation of the Smart Garage System

The response evaluation of the smart home system was conducted to analyze its operational behavior upon receiving user commands. This assessment specifically focuses on the data transmission pipeline between the Arduino IoT Cloud web server and the physical hardware, encompassing both the execution of control directives and real-time system monitoring. The procedural steps undertaken for this system response evaluation are outlined as follows:

- The system's response concerning Wi-Fi connectivity was evaluated to determine whether the smart home controller successfully established a wireless network connection. The empirical results of this network connection assessment are illustrated in Figure 10.

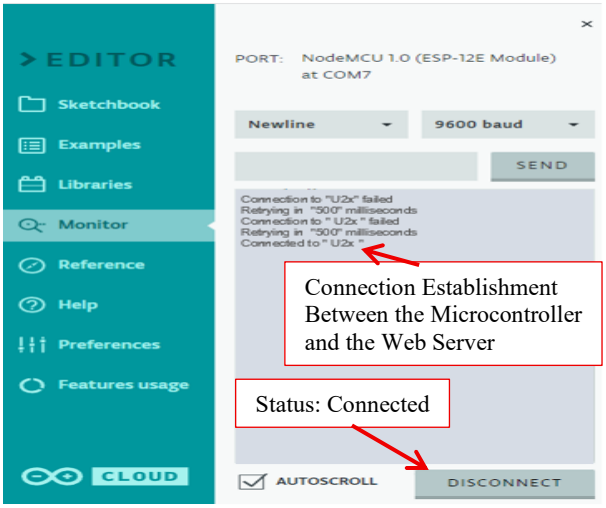

Fig 10: Network connection assessment

- Upon the successful integration of the firmware into the microcontroller, the subsequent phase focused on the configuration of a graphical user interface (GUI) dashboard via the Arduino IoT Cloud platform. This dashboard was engineered to enable real-time environmental monitoring and centralized operational control of the smart home system. The finalized interface design and its functional components are illustrated in Figure 11.

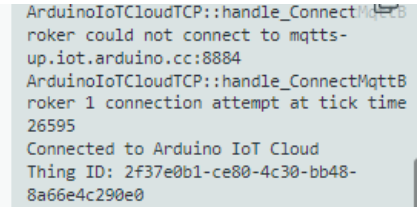

Fig 11: Once the controller establishes a connection with the Arduino IoT Cloud web server

As illustrated in the test results in Figure 11, the connection sequence to the Arduino IoT Cloud server involves establishing the MQTT (Message Queuing Telemetry Transport) protocol. Specifically, the process focuses on linking the controller to the IoT Cloud Thing identified by **ID 2f37e0b1-ce80-4c30-bb48-8a66e4c290e0**, which serves as the unique identifier for the garage system configured on the server. Once the Smart Home system establishes a stable connection with the web server, both monitoring and control functionalities become fully operational. The system's readiness for deployment is further evidenced in Figure 12.

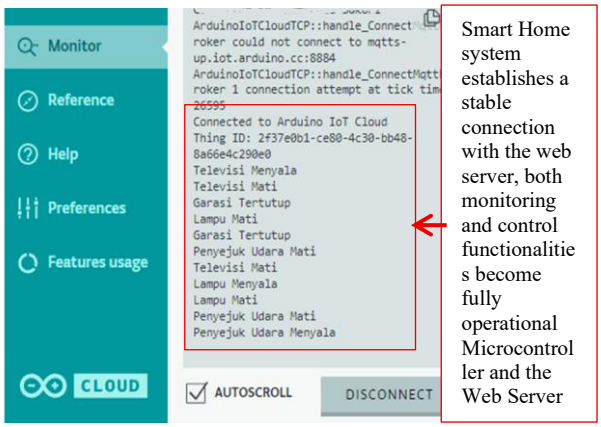

Fig 12: Information indicating that the Smart Home system has successfully established a connection with the Arduino IoT Cloud web server and is ready for operation

- An evaluation of the Smart Garage system's response was conducted to assess the effectiveness of its control and monitoring functionalities. This testing phase aimed to analyze the system's responsiveness to commands and data acquisition processes initiated via the Arduino IoT Cloud web server. The comprehensive results of the system's performance in handling these control and monitoring operations are documented in Table 1.

Table 1: System Response and Latency Analysis of the Smart Home Control and Monitoring Functions

Control/Monitoring Action	Smart Home System Response	System Response Time (s)	Arduino IoT Cloud Server Latency (s)
Open garage door	Actuates the motor to open the door and transmits a "Garage Opened" status to Arduino IoT Cloud	1	2
Close garage door	Actuates the motor to close the door and transmits a "Garage Closed" status to Arduino IoT Cloud	1	2
Switch on lights	Activates the lighting system and transmits a "Lights On" status to Arduino IoT Cloud	1	2
Switch off lights	Deactivates the lighting system and transmits a "Lights Off" status to Arduino IoT Cloud	1	3

Control/Monitoring Action	Smart Home System Response	System Response Time (s)	Arduino IoT Cloud Server Latency (s)
Activate Television	Power on the television and transmits a "TV Activated" status to Arduino IoT Cloud	1	3
Deactivate Television	Power off the television and transmits a "TV Deactivated" status to Arduino IoT Cloud	1	3
Activate Air Conditioner	Power on the AC unit and transmits an "AC Activated" status to Arduino IoT Cloud	1	3
Deactivate Air Conditioner	Power off the AC unit and transmits an "AC Deactivated" status to Arduino IoT Cloud	1	3
Request indoor temperature data	Executes sensor data acquisition, processes raw data, and transmits temperature values to Arduino IoT Cloud	2	2
Request indoor humidity data	Executes sensor data acquisition, processes raw data, and transmits humidity values to Arduino IoT Cloud	2	2

Based on the data presented in Table 1, an analysis of the smart home system's response can be conducted, focusing on several key performance indicators: system response time (latency), command execution success rate, connection stability, and Quality of Service (QoS). The observed results indicate that physical actuations, such as operating the garage door or lighting, achieve a consistent response time of 1 second, while data acquisition processes for temperature and humidity require 2 seconds for sensor reading and processing.Delay represents the time required by the system to respond to user commands until the command is successfully executed and displayed on the Arduino IoT Cloud platform.

1. Delay Analysis

Delay represents the time required by the system to respond to user commands until the command is successfully executed and displayed on the Arduino IoT Cloud platform.

Based on the testing data :

- the local system response ranged from **1–2 seconds**,
- while the Arduino IoT Cloud web server response ranged from **2–3 seconds**.

a. Delay Analysis for Actuator Control

Door Control Operation

During the door opening and closing operations:

- the system required approximately 1 second to activate the motor,
- and 2 seconds to synchronize the data with Arduino IoT Cloud.

This indicates that:

- the ESP8266 processed commands efficiently,
- while the additional delay originated from internet communication and cloud synchronization processes.

A delay of 2 seconds is still considered acceptable for smart home applications since the system does not require ultra-real-time response.

b. Lighting Controll

The lighting control operation showed:

- a local response time of 1 second,
- and a cloud response time of 2–3 seconds.

This delay difference occurred due to:

- device status transmission processes,
- internet network latency,
- and synchronization processes within the Arduino IoT Cloud dashboard.

Nevertheless, all commands were successfully executed without any failure.

c. Television and Air Conditioner Control

For television and air conditioner control:

- the local system response remained at approximately 1 second,
- while the cloud response increased to 3 seconds.

The increased delay for these electrical devices was influenced by:

- relay activation time,
- device status synchronization,
- and additional processing time on the cloud server.

Although the delay was slightly higher, the system still operated stably and responsively.

2. Delay Analysis for Monitoring System

a. Temperature Monitoring

During the temperature monitoring process:

- the system required approximately 2 seconds to read the DHT11 sensor,
- process the data,
- and transmit the data to the cloud server.

This delay was influenced by:

- sensor sampling time,
- data processing by the NodeMCU,
- and cloud upload processes.

b. Humidity Monitoring

Humidity monitoring tests showed a delay of approximately 2 seconds.

This result indicates that:

- sensor communication operated stably,
- the monitoring process functioned in near real-time,
- and the system maintained consistent response times.

3. Average Delay Calculation

The average delay was calculated using the following equation:

$$\bar{t} = \frac{\sum t_i}{n}$$

Based on the web server response data:

2, 2, 2, 3, 3, 3, 3, 3, 2, 2

Therefore:

$$\bar{t} = \frac{2+2+2+3+3+3+3+3+2+2}{10} = 2.5 \text{ seconds}$$

The calculation results indicate that the average system delay was **2.5 seconds**.

4. Command Execution Success Analysis

Based on the testing results :

- all monitoring and control commands were successfully executed,
- no response failures were observed,
- and all device statuses were successfully displayed on Arduino IoT Cloud.

Success Rate Calculation

The command execution success rate was calculated using:

$$\text{Success Rate} = \frac{\text{Successful Commands}}{\text{Total Commands}} \times 100\%$$

Substituting the testing data:

$$\text{Success Rate} = \frac{10}{10} \times 100\% = 100\%$$

The results indicate that the command execution success rate reached **100%**.

5. Connection Stability Analysis

Connection stability was evaluated based on the continuity of communication between:

- the ESP8266,
- Arduino IoT Cloud,
- and the user interface.

Based on the testing results:

- no connection disconnections occurred,
- all data transmissions were successfully completed,
- and all devices could be controlled properly.

These results indicate that the system achieved:

- high communication reliability,
- stable network performance,
- and consistent cloud synchronization.

6. Quality of Service (QoS) Analysis

a. Delay QoS

The average system delay was:

$$\bar{t} = 2.5 \text{ seconds}$$

According to the ITU-T standard:

- <150 ms → Very Good
- 150–300 ms → Good
- 300 ms → Poor

From a real-time communication perspective:

- a delay of 2.5 seconds is categorized as relatively high.

However, for IoT-based smart home applications:

- a delay of 2–3 seconds is still considered acceptable,
- since the system does not require millisecond-level critical response times.

b. Throughput QoS

Throughput was calculated using the following equation:

$$\text{Throughput} = \frac{\text{Total Data}}{\text{Transmission Time}}$$

Assuming:

- average data size = 200 bytes,
- transmission time = 2 seconds.

Therefore:

$$\text{Throughput} = \frac{200 \times 8}{2} = 800 \text{ bps}$$

The throughput value of **800 bps** indicates that:

- the system transmitted relatively small amounts of data,
- bandwidth usage was efficient,
- and the communication characteristics were suitable for IoT applications.

c. Packet Loss QoS

Based on the testing results:

- all transmitted data packets were successfully received,
- and no packet loss occurred.

The packet loss calculation was performed using:

$$\text{Packet Loss} = \frac{\text{Lost Packets}}{\text{Total Packets}} \times 100\%$$

Substituting the testing data:

$$\text{Packet Loss} = \frac{0}{10} \times 100\% = 0\%$$

The results indicate a packet loss value of **0%**, which is categorized as:

- **Very Good** according to the ITU-T standard.

7. Overall QoS Evaluation

Based on the overall testing results:

- average delay = **2.5 seconds**,
- throughput = **800 bps**,
- packet loss = **0%**,
- command execution success rate = **100%**,
- connection status = **stable**.

Therefore, the system can be categorized as:

- having good communication performance,
- stable for monitoring and control applications,
- efficient in bandwidth utilization,
- and suitable for implementation in IoT-based smart home applications.

Overall, the QoS evaluation demonstrates that the proposed IoT-based monitoring and control system achieved stable communication performance, reliable command execution, efficient bandwidth utilization, and consistent cloud synchronization. These results confirm that the integration of ESP8266, Arduino IoT Cloud, sensors, and actuator modules provides an effective solution for smart home monitoring and control applications.

4. CONCLUSIONS

This study successfully designed and implemented an IoT-based monitoring and control system for smart home applications using ESP8266 and Arduino IoT Cloud. The proposed system was capable of performing various monitoring and control functions, including door actuation, lighting control, television control, air conditioner control, temperature monitoring, and humidity monitoring through internet connectivity. The integration between sensors, actuator modules, cloud platforms, and the microcontroller enabled real-time communication and remote system accessibility.

Based on the testing results, the system demonstrated stable operational performance with a local response time of approximately 1 second and a cloud response time ranging between 2 and 3 seconds. The average delay obtained from the Quality of Service (QoS) analysis was 2.5 seconds, which is still acceptable for IoT-based smart home applications that do not require ultra-real-time communication. Furthermore, the system achieved a command execution success rate of 100%,

indicating that all monitoring and control commands were successfully processed and executed without failure.

The QoS evaluation also showed that the proposed system achieved a throughput value of 800 bps and a packet loss value of 0%, indicating efficient bandwidth utilization and highly reliable communication performance. In addition, the connection stability remained consistent throughout the testing process without communication interruptions or synchronization failures. These results confirm that the proposed system provides reliable monitoring and control capabilities with stable internet-based communication performance.

Overall, the implementation of Arduino IoT Cloud combined with ESP8266 proved to be effective for developing IoT-based smart home monitoring and control systems. The proposed system can therefore serve as a reliable and efficient solution for remote monitoring and automation applications in smart home environments.

5. ACKNOWLEDGMENTS

The author expresses his deepest gratitude to the organizers of the International Journal of Computer Application (IJCA) for providing the opportunity to publish this paper. Sincere appreciation is also extended to the Manado State Polytechnic, particularly to the advisors within the Department of Electrical Engineering and all parties who have provided support, enabling the successful completion of this research.

6. REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010, doi: 10.1016/j.comnet.2010.05.010.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013, doi: 10.1016/j.future.2013.01.010.

[3] M. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/COMST.2015.2444095.

[4] A. Alheraish, "Design and Implementation of Home Automation System," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 4, pp. 1087–1092, 2004, doi: 10.1109/TCE.2004.1362492.

[5] M. Díaz, C. Martín, and B. Rubio, "State-of-the-Art, Challenges, and Open Issues in the Integration of Internet of Things and Cloud Computing," *Journal of Network and Computer Applications*, vol. 67, pp. 99–117, 2016, doi: 10.1016/j.jnca.2016.01.010.

[6] H. Ning and H. Liu, "Cyber-Physical-Social Thinking Space Based Science and Technology Framework for the Internet of Things," *Science China Information Sciences*, vol. 58, no. 3, pp. 1–19, 2015, doi: 10.1007/s11432-014-5209-2.

[7] . E. Mills and D. F. Treagust, "Engineering Education—Is Problem-Based or Project-Based Learning the Answer?," *Australasian Journal of Engineering Education*, vol. 3, no. 2, pp. 2–16, 2003, doi: 10.1080/22054952.2003.11447551.

[8] D. Prihatmoko, "Penerapan Internet of Things (IoT) dalam Pembelajaran di UNISNU Jepara," *Jurnal SIMETRIS*, vol. 7, no. 2, pp. 567–574, 2016, doi: 10.24176/simet.v7i2.769.

[9] O. K. Sulaiman, M. Ihsan, and A. H. Rambe, "Cloud-Based Internet of Things Platform for Smart Monitoring System," *IOP Conference Series: Materials Science and Engineering*, vol. 309, no. 1, pp. 1–7, 2018, doi: 10.1088/1757-899X/309/1/012078.

[10] D. Bandyopadhyay and J. Sen, "Internet of Things: Applications and Challenges in Technology and Standardization," *Wireless Personal Communications*, vol. 58, no. 1, pp. 49–69, May 2011, doi: 10.1007/s11277-011-0288-5.

[11] M. A. Al-Qutayri and J. S. Jeedella, "Integrated Wireless Technologies for Smart Home Applications," *IEEE Systems Journal*, vol. 4, no. 4, pp. 431–441, Dec. 2010, doi: 10.1109/JSYST.2010.2058477.

[12] S. Li, L. D. Xu, and S. Zhao, "The Internet of Things: A Survey," *Information Systems Frontiers*, vol. 17, no. 2, pp. 243–259, Apr. 2015, doi: 10.1007/s10796-014-9492-7.

[13] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013, doi: 10.1016/j.future.2013.01.010.

[14] J. W. Thomas, *A Review of Research on Project-Based Learning*. San Rafael, CA, USA: Autodesk Foundation, 2000.

[15] M. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, Fourth Quarter 2015, doi: 10.1109/COMST.2015.2444095.

[16] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, Jun. 2009, doi: 10.1016/j.future.2008.12.001.