Journal of Applied Informatics and Computing (JAIC)
Vol.9, No.4, August 2025, pp. 1593~1601
e-ISSN: 2548-6861

Design of an Internet of Things (IoT)-Based Fish Feeder System Using an Android Application

Zulham Ariyandi [1]*, Taufiq [2]*, Nunsina [3]*
* Informatics Engineering, Malikussaleh University
zulham.210170152@mhs.unimal.ac.id [1], taufiq.te@unimal.ac.id [2], nunsina@unimal.ac.id [3]

Article Info

Article history:

Received 2025-06-04
Revised 2025-07-09
Accepted 2025-07-13

Keyword:

Internet of Things (IoT),
automatic fish feeder,
ESP32,
Firebase,
Android application.

ABSTRACT

Fish farming plays a crucial role in aquaculture, where feed management is a key factor affecting productivity and operational costs. This research presents the design and implementation of an Internet of Things (IoT)-based automatic fish feeder system, integrated with a custom Android application. The system uses an ESP32 microcontroller to control a load cell sensor for accurate feed weighing, an ultrasonic sensor to monitor feed availability, servo motors for feed release mechanisms, and a DC motor for feed dispersion. Firebase Realtime Database serves as the data communication medium between the hardware and mobile application, enabling real-time control and monitoring. A rule-based control logic is implemented to execute scheduled or manual feeding processes. Experimental results show a feed weight accuracy of ±5 grams, with feeding operations completed within 1.5 minutes and an average throw distance of 287.8 cm. The system supports automatic alerts, scheduling, feed history logging, and remote access via the application. Compared to conventional manual methods, the system reduces feed waste, increases portion accuracy, and decreases feeding time by over 75%. These features demonstrate the system's capability to enhance feeding efficiency, reduce labor dependency, and support sustainable and scalable fish farming practices through automation and real-time monitoring.

I. INTRODUCTION

Fish farming is one of the key sectors in the fisheries industry that continues to grow rapidly in Indonesia. One crucial aspect of fish farming is timely and adequate feed distribution, as it directly affects fish growth, health, and feed cost efficiency, which constitutes the largest portion of operational expenses [1]. Improper feeding—either in quantity or timing—can result in suboptimal fish growth, feed waste, and declining water quality due to feed residue accumulation.

The Internet of Things (IoT) technology was first introduced in 1999. IoT is a system that enables multiple smart devices to connect and communicate within a network, as well as interact with their surrounding environment [2]. It is closely related to the concept of machine-to-machine (M2M) communication. Devices with M2M capabilities are often referred to as smart devices, which are designed to assist humans in performing various tasks efficiently [3].

In practice, manual feeding poses many challenges, especially for fish farmers with busy schedules or who live far from their ponds. Dependence on human labor also increases the risk of mistimed feeding, which can negatively affect productivity [4]. In large-scale fish farming operations, manual feeding becomes inefficient in terms of both time and cost [5].

In the current era of digitalization, the Internet of Things (IoT) technology offers innovative solutions to improve efficiency across various sectors, including aquaculture. IoT enables integration between hardware devices and internet connectivity, allowing systems to be monitored and controlled remotely. Therefore, to assist fish farmers, there is a need for an automated system capable of performing feeding tasks using an ESP32 microcontroller and accessible through a custom-designed Android application. ESP32 is an

integrated System-on-Chip (SoC) microcontroller equipped with 802.11 b/g/n WiFi and Bluetooth 4.2 connectivity, along with various built-in features. It includes a processor, memory, and general-purpose input/output (GPIO) pins, enabling it to replace boards like the Arduino. Its native ability to connect to WiFi networks makes it ideal for IoT-based applications [6]. This tool facilitates remote control of fish feeding even when the farmer is not physically present at the site [7].

To further improve feeding efficiency, a load cell sensor is used to measure the required feed quantity, minimizing feed wastage and preventing excess feed from dissolving in the pond, which can alter the water's pH level and harm or kill the fish. Additionally, if feed is dispensed in only one spot and not evenly spread, fish growth will be uneven. Therefore, a DC motor (Direct Current) is used to distribute the feed more evenly across the pond, supporting optimal fish growth. The use of an automatic fish feeder also reduces dependency on manual labor. With this system, farmers can focus more on other management aspects or even manage more ponds within the same time frame [8].

Several previous studies have explored this field. An automatic feeder based on ESP32 and Telegram was developed, although it lacked a weighing mechanism and advanced control [9]. A system utilizing the Wemos D1 Mini and Telegram Bot, but without feed dispersion capability, was also introduced [10]. Another study focused more on mechanical aspects without mobile application integration [11]. A more advanced system that used a servo motor and load cell to weigh feed before dispensing it into the pond—allowing feed quantity adjustment by weight and incorporating an RTC module for scheduled feeding—was designed in [12]. Other systems using servo motors as the main actuator were also proposed [13], [14]. Some of these integrated ultrasonic sensors to monitor feed levels and trigger buzzer alerts when the feed was low. These limitations highlight the need for a more integrated and intelligent system as proposed in this study.

Based on the above background, the authors propose a solution through a thesis project entitled "Design of an Internet of Things (IoT)-Based Fish Feeder System Using an Android Application". Therefore, this study aims to design an IoT-based fish feeding system equipped with an automatic scheduling feature and integrated with a custom Android application to serve as a control and monitoring tool, enhancing productivity and efficiency in fish farming operations.

II. METHOD

This research adopts an experimental approach involving the development of both hardware and software to design, implement, and test an Internet of Things (IoT)-based automatic fish feeder system. The main focus is on integrating sensors, actuators, and a mobile application to automate the feeding process through an internet connection.

A. Rule-Based Method

The control system in this tool employs a rule-based logic approach using IF–THEN patterns. This method is designed to automate the feeding process based on sensor inputs and user commands. The algorithm works according to a predefined set of rules without any learning or adaptive capabilities. Each condition (IF) is evaluated to determine the appropriate action (THEN), such as opening the valve, weighing the feed, or triggering feed dispersal. This process is repeated as long as the conditions are met; otherwise, the system exits the loop. This approach is chosen for its ease of implementation and clear logic, making it suitable for systems with fixed scenarios that can be explicitly defined through rules [15]. The system follows a series of predefined rule-based conditions to manage the automatic feeding process. The logic is implemented using IF–THEN statements as described below.

1) *Feed Command Initialization Rule*:
- IF the system receives a feed command from the application,
- THEN the system will determine the amount of feed (0.3 kg or 0.5 kg), begin dispensing feed from the storage container, send a confirmation to the application that the process has started.

2) *Target Feed Weight Achieved Rule*:
- IF the dispensed feed weight matches the target weight,
- THEN the system will stop the dispensing process, open the channel to pour the feed into the pond, notify the application that the feed is ready for dispersal.

3) *Low Feed Stock Warning Rule*:
- IF the remaining feed in the container is less than 20%,
- THEN the system will send a warning to the application that the feed stock is running low.
- ELSE, the system will deactivate the low-feed warning.

4) *Feeding Timeout Rule*:
- IF the feeding process exceeds the predefined duration (e.g., 1 minute) and the target weight has not been reached,
- THEN the system will automatically stop all operations, send a notification stating: "Failed: Target feed weight not reached within allowed time." return to standby mode.

5) *Feeding Completion Rule*:
- IF the feed pouring process is completed,
- THEN the system will close the feed channel, send a report to the application: "Feeding process completed.", reset all parameters to prepare for the next operation.

These rules are essential for ensuring that the feeding process operates safely, efficiently, and with minimal human intervention. By structuring the control logic in this rule-based format, the system can make decisions autonomously in response to real-time sensor inputs and user commands. This not only reduces the risk of overfeeding or underfeeding but also enhances reliability by detecting abnormal conditions, such as feed stock depletion or mechanical delays.

Furthermore, the inclusion of automated feedback and alert mechanisms through the Android application ensures that users remain informed of the system's status at all times. This logical framework serves as the backbone for the IoT-based fish feeder's intelligent and responsive behavior, supporting consistent and optimized aquaculture operations.

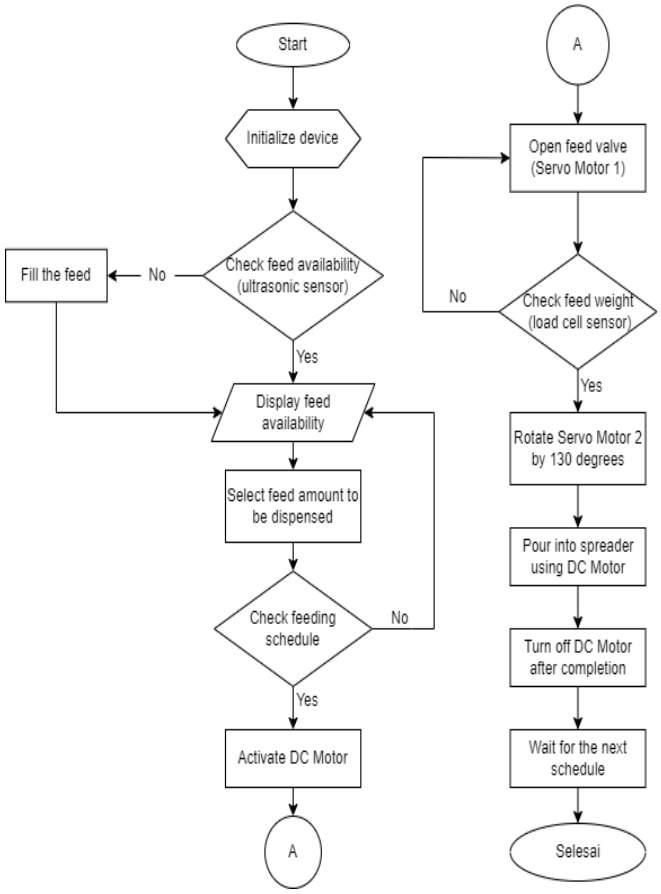

Figure 1. System Flow Explanation

Figure 1 illustrates the flowchart of the automated fish feeder system based on Internet of Things (IoT). The process begins with the initialization of the device, which includes activating all necessary hardware components such as sensors, microcontroller, and actuators. The system then checks the availability of feed using an ultrasonic sensor. If the feed level is below the required threshold, the system prompts the user to refill the feed container.

Once the feed is available, the system displays the current feed availability status through the application interface. The user can then select the desired amount of feed to be dispensed. The system subsequently checks the feeding schedule, which has been predefined and stored in the Firebase Realtime Database.

If the current time matches the feeding schedule, the DC motor is activated to begin the feeding process. At this point, the flow continues to the next stage marked as connector A. In the next phase (connector A), the system opens the feed valve using Servo Motor 1. After the valve is opened, the system checks the feed weight using a load cell sensor to ensure the correct amount is being dispensed. Then, Servo Motor 2 is rotated by 130 degrees to move the feed into the dispensing mechanism.

The feed is poured into a rotating spreader powered by a DC motor, which evenly distributes the feed across the fish pond. After the dispensing process is complete, the system turns off the DC motor and returns to standby mode, waiting for the next scheduled feeding cycle.

This automated flow reduces the dependency on manual labor and ensures timely, measured, and efficient feed distribution, supporting optimal fish growth and better feed utilization.

B. Sytem Design

To ensure efficient programming and prevent missing components during development, a complete system schematic is required. Figure 1 below illustrates the overall design of the IoT-based fish feeder system with automatic scheduling integrated into an Android application.

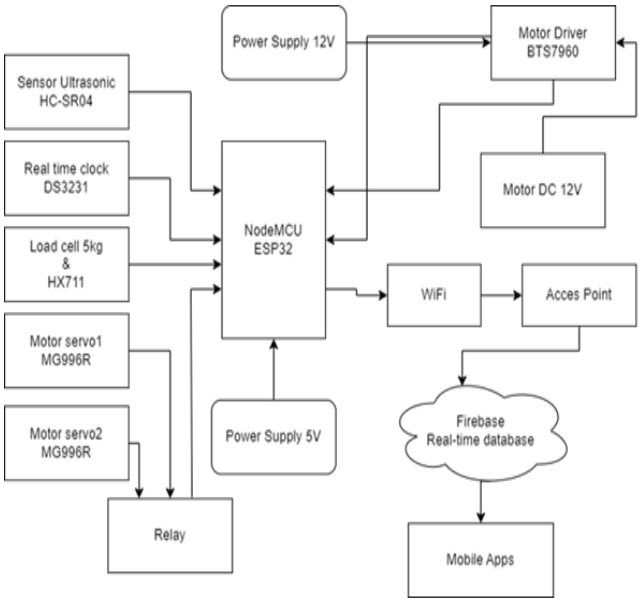

Figure 2. System Design

As shown in Figure 2, the 5V power supply is used to power the NodeMCU ESP32. Components such as the MG996R servo motors, HC-SR04 ultrasonic sensor, DS3231 real-time clock, load cell sensor, and HX711 amplifier are connected to the ESP32. The DC motor is controlled using the BTS7960 motor driver, which receives power from a 12V power supply. A one-channel relay is used to regulate the current to the DC motor before being routed to the ESP32. After the ESP32 is programmed, it is connected to the

Firebase Realtime Database and the custom mobile app built with Flutter. A REST API handles the communication between the application and the hardware modules and sensors.

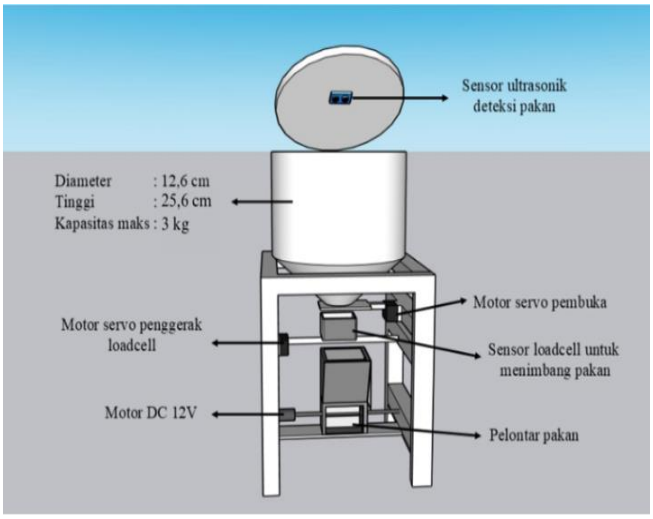

Figure 3. System Workflow Illustration

Figure 3 illustrates how the system operates. It begins by detecting feed levels using the ultrasonic sensor, while reading the scheduled time set via the Android application. The servo motor opens the feed valve, dispensing feed onto the load cell for weight measurement according to the programmed amount. Then, another servo motor directs the feed toward the dispersing mechanism powered by a 12V DC motor, which can be controlled to adjust feed-throwing distance using the motor driver. The feed container has a diameter of 12.6 cm and a height of 25.6 cm, with a maximum capacity of 3 kg.

C. Hardware Circuit Design

To ensure proper functionality and integration of all components, the hardware configuration was carefully designed by interconnecting various modules with the ESP32 microcontroller. The complete wiring process is described in Figure 4 detailing each connection required for power distribution, signal control, and communication among modules and sensors.

Figure 4. illustrates the hardware design process by connecting each module using jumper wires. The initial step begins with supplying power from a 12V power source (V+) to the COM terminal of a 1-channel relay and to the 12V terminal on the motor driver. Next, the NO terminal on the relay is connected to the 5V terminal of the motor driver. The DC motor is then connected to the OUT1 and OUT2 terminals of the motor driver. The D– terminal of the relay and the GND terminal of the motor driver are both connected to the negative (–) rail of the breadboard. Meanwhile, the D+ terminal of the relay is connected to the positive (+) rail of the breadboard to supply power to the ESP32.

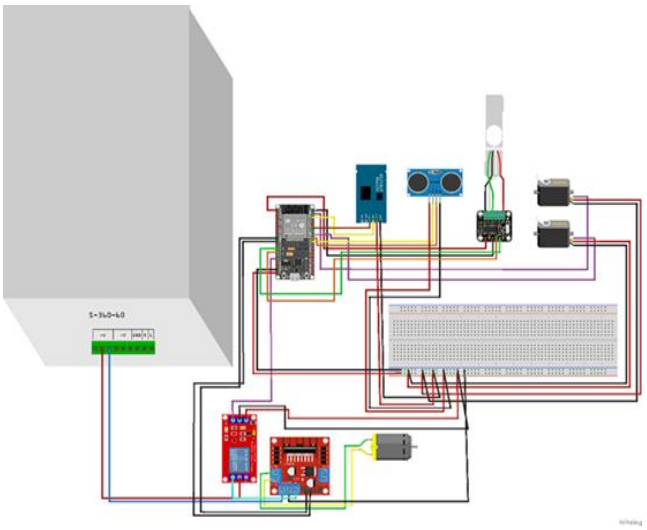

Figure 4. Hardware Circuit Design

The GPIO13 pin of the ESP32 is connected to the D+ terminal of the relay. The IN1 and IN2 terminals of the motor driver are connected to GPIO25 and GPIO26 of the ESP32, respectively. The 5V and GND pins of the NodeMCU ESP32 are connected to the breadboard to distribute power to the connected modules and sensors. The VCC and GND terminals of each module or sensor are also connected to the positive (+) and negative (–) rails of the breadboard. For the real-time clock (RTC) DS3231 module, the SCL and SDA terminals are connected to GPIO21 and GPIO22 on the ESP32. The ultrasonic sensor's Trig and Echo terminals are connected to GPIO18 and GPIO5, respectively. The SCK and DATA terminals of the load cell sensor are connected to GPIO12 and GPIO14. Finally, servo motor 1 is connected to GPIO19, and servo motor 2 is connected to GPIO23 of the ESP32.

III. RESULTS AND DISCUSSION

A. System Implementation

The IoT-based automatic fish feeder system was successfully implemented using the ESP32 microcontroller connected to the Firebase Realtime Database. The system integrates an ultrasonic sensor to detect remaining feed, a load cell sensor to measure feed weight, two servo motors to control the valve and direction of feed flow, and a DC motor to disperse the feed. The user interface was developed as an Android mobile application using the Flutter framework, enabling real-time monitoring and control.

Figure 5. Final Implementation of the Automatic Fish Feeder Tool

B. Sensor Component Testing

1) *Load Cell Sensor:* The load cell sensor was calibrated using a factor of 359 and successfully read feed weights in grams. It was placed in the feed container and tested by gradually adding weight. The results showed accurate readings with data sent to Firebase closely matching the actual weight, with an error tolerance of ±5 grams.

Figure 6. Load Cell Sensor Serial Monitor Output

2) *Ultrasonic Sensor:* The ultrasonic sensor successfully measured feed availability as a percentage based on the measured distance, with category-based notifications (full, sufficient, low, nearly empty, empty) sent to the application. The following formula was used to calculate the remaining feed percentage:

$$\%_{feed} = \left(\frac{H_{container} - D_{sensor}}{H_{container}}\right) x\ 100\%$$

Information:

- $\%_{feed}$: Feed percentage remaining in the container
- $H_{container}$: Total height of the container (cm)
- D_{sensor}: Distance measured from the sensor to the feed surface (cm)

The results of feed availability monitoring using an ultrasonic sensor display percentages based on the distance of the feed, where each percentage triggers a warning level: full, adequate, low, almost empty, or empty. The following data were obtained from the monitoring test of feed availability in a container with a height of 26.5 cm.

TABLE I.
FEED AVAILABILITY MONITORING RESULTS

No	Distance (cm)	Feed Percentage	Notification
1	1.97 cm	92.3 %	Full
2	14.86 cm	42.0 %	Low
3	9.11 cm	64.4 %	Sufficient
4	7.45 cm	70.9 %	Sufficient
5	13.18 cm	48.5 %	Low
6	10.47 cm	59.1 %	Low
7	15.88 cm	38.0 %	Low
8	21 cm	17.97 %	Nearly Empty
9	20,4 cm	20,31 %	Low
10	25 cm	2,3 %	Empty

C. System Control Testing

Control commands were sent through the mobile app, allowing the user to select between 0.3 kg and 0.5 kg feed portions. Upon command, the ESP32 executed a sequence of operations: opening the valve, weighing the feed, and dispersing it. The results showed the system could stop the motor automatically when the target weight was reached and issue a timeout notification if the target was not met within 60 seconds.

TABLE II.
FEED CONTROL PROCESS TESTING RESULTS

No	Target Weight	Actual Weight	Motor Status	Delay
1	0,3 kg	311 grams	Active	2 seconds
2	0,5 kg	508 grams	Active	3 seconds
3	0,3 kg	304 grams	Active	1 second
4	0,5 kg	498 grams	Active	2 seconds
5	0,3 kg	250 grams	Timeout	6 seconds

To evaluate the system's weighing accuracy and timing consistency, experiments were conducted by performing five trials for each target feed weight, namely 0.3 kg and 0.5 kg. In each trial, the user manually selected the desired portion

size via the Android application. The system then executed the full feeding sequence: opening the servo valve, weighing the feed using the load cell sensor, and activating the DC motor to disperse the feed. During each test, the actual weight dispensed was measured and recorded, along with the time delay from activation to completion, and the motor status (successful or timeout). These values were then tabulated to assess consistency. The results showed that for 0.5 kg target weight, the deviation from the target was within ±8 grams, while for 0.3 kg, the deviation was within ±11 grams, except in cases where the feed supply was insufficient. This test approach allowed the system's accuracy (±5 grams tolerance) and consistency in timing to be verified through replicated experiments under similar conditions.

Testing on the DC motor was also conducted to evaluate its operational duration during the feed dispersal process based on selected feed portions. Results showed the motor run time was proportional to the amount of feed: an average of 50 seconds for 0.3 kg and 65 seconds for 0.5 kg. The motor was automatically activated after weighing and stopped once dispersal was complete.

TABLE III.
DC MOTOR TESTING BASED ON FEED VOLUME

No	Feed Amount (kg)	Motor DC Run Time	Remark
1	0,3 kg	50 seconds	Successful throw
2	0,5 kg	65 seconds	Successful throw
3	0,3 kg	50 seconds	Successful throw
4	0,5 kg	65 seconds	Successful throw
5	0,3 kg	50 seconds	Successful throw

From Table III, it can be concluded that the DC motor required approximately 50 seconds to disperse 0.3 kg of feed and 65 seconds for 0.5 kg. The time varied slightly depending on motor speed and feed availability.

In addition to evaluating the DC motor duration, a test was also conducted to determine how far the feed could be thrown after being dispensed from the container and rotated by the DC motor. The throw distance is influenced by the motor's rotation speed, the design of the throwing mechanism, and the weight of the feed. In this experiment, feed weighing 0.3 kg and 0.5 kg was launched, and the farthest landing distance was measured using a measuring tape from the device's position. The following are the results:

TABLE IV.
FEED THROW DISTANCE TESTING RESULTS

No	Feed Amount (kg)	Throw Distance (cm)	Remark
1	0,3 kg	274 cm	Successful throw
2	0,3 kg	287 cm	Successful throw
3	0,5 kg	286 cm	Successful throw
4	0,3 kg	290 cm	Successful throw
5	0,3 kg	302 cm	Successful throw

From the results shown in Table IV, it is evident that the feed-throwing mechanism can distribute the feed up to an average distance of approximately 287.8 cm, with the maximum distance being 302 cm and the minimum distance 274 cm. These findings suggest that the system is capable of spreading the feed evenly across a fish pond area of 3×5 meters, as used in this study, supporting optimal feeding coverage.

D. Comparison With Convetional Methods

The designed IoT-based fish feeder system significantly improves performance compared to traditional manual feeding methods. Based on observations and interviews with local fish farmers, conventional feeding typically involves manual measurement and hand-spreading of feed, which takes an average of 6–8 minutes per feeding cycle for a medium-sized pond (approximately 3×5 meters). In contrast, the automatic system completes a 0.3 kg or 0.5 kg feeding process in an average of 1.5–2.5 seconds for weighing and 50–65 seconds for dispersal, totaling less than 1.5 minutes, resulting in a time saving of over 75%.

In terms of feed efficiency, manual feeding often leads to overfeeding or uneven distribution, which can result in 10–20% feed waste due to clumping or sinking before being consumed. The automated system, which uses a load cell and a rotating dispersal mechanism, reduces feed waste by ensuring accurate portions (±5 grams) and even distribution up to 3 meters, contributing to more uniform fish growth and better water quality.

TABLE V.
THE KEY DIFFERENCES BETWEEN MANUAL AND AUTOMATED FEEDING APPROACHES

No	Parameter	Manual Feeding	IoT-Based Automated System
1	Feeding Duration	6–8 minutes	1.5–2 minutes
2	Portion Accuracy	Estimated (high variance)	±5 grams (tested)
3	Feed Waste	10–20% (estimation)	<5%
4	Distribution Coverage	Limited to reachable area	Up to 302 cm radius
5	Labor Dependency	High	Low (remote-controlled)
6	Notification & Monitoring	None	Real-time via mobile app
7	Scheduling	Manual	Automatic (with RTC & app)

These findings highlight the system's advantage in time efficiency, feed utilization, and remote management, making it more suitable for scalable and modern fish farming practices.

E. Monitoring dan Real-Time Sycnchronization

Sensor data was transmitted to Firebase in less than one second, ensuring nearly instant synchronization between the device and the mobile application. The app displays connection status, feed level, current time, and feeding history. The monitoring interface includes visual indicators of feed percentage and manual control options.

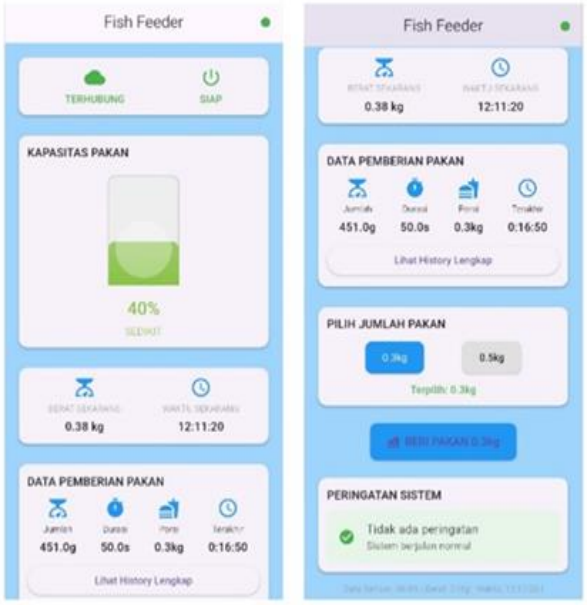

Figure 7. Application Monitoring Interface

The developed mobile app enables real-time monitoring and control of the fish feeding system via Firebase Realtime Database integration. The main screen displays device connection status, feed capacity based on ultrasonic sensor readings, and feed weight from the load cell. The app also shows a history of feedings, including amount, time, and duration. The control feature allows users to select feed portions (0.3 kg or 0.5 kg) before initiating the process. Additionally, the system is equipped with automatic alerts to detect abnormal conditions, offering users a responsive, informative, and adaptive control experience.

F. Comparison of Rule-Based vs. Real-Time Systems

To evaluate system performance, a comparison was made between a local rule-based method and a real-time Firebase-based system. Table V presents the results.

TABLE VI.
PERFORMANCE COMPARISON OF RULE BASED VS. REAL-TIME METHODS

No	Test Parameter	Rule Based	Realtime
1	System Response Time	3–5 seconds	2 seconds
2	Feed Weighing Accuracy	±2-4 grams	±5-10 grams
3	Remote Control	Not available	Available
4	Notifications	Not available	Available
5	Feed Portion Configuration	Static (only 0.3 kg)	Dynamic (0.3 kg or 0.5 kg)
6	Feed Level Monitoring	Not available	Real-time available
7	Feed History Logging	Not available	Real-time available
8	Internet Connectivity Requirement	Not required	Available in Firebase
9	Field Adaptability	Less adaptive to environmental changes	Adaptive, due to its sensor-based and real-time data system
10	Data Security & Control	No authentication	Firebase user authentication

The comparison results indicate that the IoT-based real-time system excels in flexibility, operational accuracy, remote monitoring, and control. Although it relies on internet connectivity, it proves to be more adaptive and efficient.

G. Statistical Performance Evaluation

To validate the system's performance in terms of responsiveness, reliability, and power efficiency, a series of statistical tests were conducted based on repeated experiments during system operation. The performance parameters evaluated include response latency, weighing time variation, and power consumption behavior.

1) System Response Latency: Latency is defined as the delay between when a command is sent from the Android application and when the ESP32 microcontroller starts executing the feeding process. Ten trials were conducted for each feed portion (0.3 kg and 0.5 kg), and the latency was measured using serial timestamps and synchronized Firebase logs.

TABLE VII
SYSTEM RESPONSE TESTING

Feed Portion	Average Latency (ms)	Standard Deviation	Max Latency	Min Latency
0.3 kg	845 ms	±92 ms	1012 ms	710 ms
0.5 kg	862 ms	±88 ms	998 ms	731 ms

The results show that the system consistently responds in under 1 second, supporting the classification of real-time system performance with an average latency below 900 milliseconds.

2) Feed Dispensing Time Variability (Weighing Time): To measure consistency in the weighing process, the time taken from the start of feed release to reaching the target weight was recorded over ten iterations for each target. The variation was analyzed to evaluate system precision.

TABLE VIII
FEED DISPENSING TIME TESTING

Feed Target	Avg. Dispensing Time	Std. Deviation	Error Rate (outside ±5g)
0.3 kg	1.54 seconds	±0.23 s	10% (1 of 10 tests)
0.5 kg	2.48 seconds	±0.18 s	0%

The standard deviation for both targets remained below 0.25 seconds, indicating high timing consistency. Only one outlier occurred during the 0.3 kg test due to feed depletion, demonstrating the system's robustness.

3) *Power Consumption Analysis:* Using a current sensor module and USB power meter, the system's average power draw was measured during three phases: idle (standby), active feeding, and data synchronization.

TABLE IX
POWER CONSUMPTION ANALYSIS RESULTS

System State	Active Components	Avg. Current Draw	Avg. Voltage	Power Consumption
Standby Mode	ESP32 + WIFI	82 mA	5.1 V	0.42 W
Feeding Operation	ESP32 + MOTOR	800 mA	12.0 V	9.6 W
Data Sync (WiFi)	ESP32 + WIFI	230 mA	5.0 V	1.15 W

Based on the table above, the system's power consumption varies significantly depending on its operational state and active components. In standby mode, when the ESP32 microcontroller is idle but maintaining Wi-Fi connectivity, the current draw is approximately 82 mA at 5.1V, resulting in a power consumption of 0.42 watts. This indicates that the system is highly energy-efficient during idle periods and can remain in this state for extended durations with minimal power usage.

During data synchronization, where the ESP32 is actively communicating with the Firebase Realtime Database via Wi-Fi, the average current draw increases to 230 mA at 5.0V, translating to 1.15 watts. This higher consumption reflects the additional workload required for wireless data transmission and real-time updates to the mobile application.

The feeding operation represents the most power-intensive state. In this phase, both the ESP32 and the DC motor (powered via a 12V external supply) are active. The motor draws an average current of 800 mA at 12V, resulting in a peak power consumption of 9.6 watts. Despite the high momentary load, the duration of each feeding cycle is relatively short—approximately 50 to 65 seconds—which keeps the total energy expenditure per cycle relatively low.

These results demonstrate that the system is optimized for energy efficiency during non-critical operations and is capable of handling higher loads during active phases without compromising functionality. This performance profile makes the system suitable for outdoor and remote fish farming environments, especially when paired with a high-capacity power bank, sealed lead-acid battery, or solar-powered energy source.

H. Discussion

Overall, the system successfully reduces the need for manual intervention while improving feeding accuracy and operational efficiency. The integration of sensors and a cloud database supports data-driven decision-making in real time. The main challenges include reliance on a stable internet connection and the need for periodic sensor calibration to maintain long-term accuracy.

Another critical consideration is the security of the IoT system, particularly because it relies on cloud-based services such as Firebase Realtime Database for data transmission and remote control. Without proper safeguards, the system may be vulnerable to unauthorized access, data breaches, or manipulation of feeding commands. In this prototype, basic Firebase authentication using email and password is implemented to restrict access to verified users. However, to enhance security, future improvements should include encrypted communication (e.g., HTTPS), role-based access control, real-time logging of user activity, and automatic logout features. Implementing these measures would significantly mitigate potential cybersecurity risks and ensure the integrity and safety of the fish feeding operation.

However, one limitation of the current prototype is the lack of waterproofing or environmental shielding, which poses a risk when deployed in outdoor environments such as fish ponds or aquaculture farms. Exposure to rain, humidity, and extreme temperatures could affect the durability of the electronics, especially the ESP32 microcontroller, servo motors, and sensors. Therefore, future development should consider the use of waterproof enclosures (e.g., IP65-rated casing), corrosion-resistant materials, and protective covers to ensure the system can operate reliably under harsh environmental conditions. Additionally, testing under long-term outdoor exposure is recommended to evaluate the system's resilience and durability in real-world scenarios.

IV. CONCLUSION

Based on the results of this study, it can be concluded that the Internet of Things (IoT)-based automatic fish feeder system integrated with an Android application was successfully implemented using the ESP32 microcontroller connected to the Firebase Realtime Database. This configuration allows real-time control and monitoring directly from the mobile application. The system utilizes a load cell and an ultrasonic sensor, both of which provide accurate measurements for feed weight and availability level, with a tolerance of ±5 grams and reliable readings based on calculated distance, respectively.

The automatic feeding control operates effectively, consistently achieving the targeted feed weight and generating appropriate notifications in cases of abnormalities, such as feed timeout when the supply runs out. Furthermore, the DC motor-based feed dispersal mechanism was capable of distributing feed evenly across the pond area, with a measured throw distance ranging from 274 cm to 302 cm, and an average distance of 287.8 cm. This demonstrates the system's ability to efficiently spread feed, supporting uniform fish growth.

The integration with Firebase and the Flutter-based mobile application enables users to remotely monitor feeding schedules, feed levels, and historical data in real time, reducing labor dependency and minimizing human error. Compared to manual methods, the system reduces feeding time by over 75%, decreases feed waste to less than 5%, and increases feed portion accuracy to within ±5 grams, supporting more sustainable aquaculture practices.

In addition, power consumption tests show that the system is energy-efficient in standby and communication states, while remaining sufficiently robust during feeding operations. These characteristics make it suitable for remote or outdoor fish pond environments, particularly when combined with battery or solar-powered energy sources.

Future development should address environmental protection such as waterproofing and enclosure sealing to ensure long-term durability in outdoor conditions. The implementation of advanced security measures (e.g., encrypted communication, role-based access) is also essential to protect user data and system commands. Furthermore, integration with data analytics or AI-based decision-making systems could enhance feed optimization based on fish growth patterns, water quality, or weather data.

Overall, this system presents a reliable, scalable, and efficient solution that contributes to automation, cost reduction, and enhanced productivity in the aquaculture sector.

REFERENCES

[1] M. I. Nadhir, W. Hayuhardhika, N. Putra, and E. Setiawan, "Pengembangan Sistem Otomatisasi Pemberian Pakan Ikan Nila Berbasis Internet of Things (Studi Kasus:Kampung Sawah Baru)," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 1, no. 1, 2017, [Online]. Available: http://j-ptiik.ub.ac.id

[2] M. Ula, R. Tjut Adek, and Muklish, "Towards The Secure Internet of Things: Threats and Solution," *Proc. Malikussaleh Int. Conf. Multidiscip. Stud.*, vol. 3, pp. 66–78, Dec. 2022, doi: 10.29103/micoms.v3i.49.

[3] A. Selay *et al*., "Internet of Things," *Karimah Tauhid*, vol. 1, no. 2963–590X, pp. 861–862, 2022.

[4] F. Wahyuni Sabran and E. Zalfiana Rusfian, "Penggunaan Internet of Things pada eFishery untuk keberlanjutan Akuakultur di Indonesia," *J. Soc. Sci. Res.*, vol. 3, no. 2, pp. 8142–8156, 2023.

[5] N. Busaeri, R. Nurdiansyah, and A. Rahman, "Penerapan Teknologi Penebar Pakan Ikan Otomatis Berbasis IoT di Dusun Citengah Kecamatan Cihaurbeuti," *BERNAS J. Pengabdi. Kpd. Masy.*, vol. 4, no. 2, pp. 1490–1498, 2023, doi: 10.31949/jb.v4i2.4725.

[6] M. Nizam, H. Yuana, and Z. Wulansari, "Mikrokontroller ESP32 Sebagai Alat Monitoring Pintu Berbasis Web," *J. Mhs. Tek. Inform.*, vol. 6, no. 2, 2022.

[7] M. H. Wiwi and D. Prasetyo Isnandar, "Prototipe Alat Pemberi Pakan Ikan Berbasis Internet of Things," *J. Ilm. Teknol. Inf. Asia*, vol. 18, no. 02, 2024.

[8] A. Tahir and N. Nurdjan, "Konstruksi Mesin Pakan Ikan Otomatis," *J. Vokasi Tek. Mesin dan Fabrikasi Logam*, vol. 2, no. 1, pp. 54–63, 2023.

[9] E. M. Indrawati, B. Suprianto, and U. T. Kartika, "Pemberi Pakan Ikan Otomatis berbasis IoT dengan FLC Berdasarkan Kualitas Air (Suhu , PH , Kekeruhan)," vol. 13, no. 3, pp. 383–394, 2024.

[10] H. Chaerisma, F. Teknik, T. Informatika, and U. Pamulang, "Prototype Sistem Monitoring dan Pemberi Pakan Ikan Hias Berbasis Wemos D1 Mini," vol. 2, no. 1, pp. 101–112, 2023.

[11] E. H. Tiarto, S. Mansur, and A. Kinasih, "Rancang Bangun Purwarupa Mekanik Alat Pemberi Pakan Ikan Otomatis," *Sebatik*, vol. 28, no. 1, pp. 206–212, 2024, doi: 10.46984/sebatik.v28i1.2458.

[12] A. M. Putra and A. B. Pulungan, "Alat Pemberian Pakan Ikan Otomatis," *J. Tek. Elektro dan Vokasional*, vol. 06, pp. 113–121, 2020, [Online]. Available: http://ejournal.unp.ac.id/index.php/jtev/index

[13] Hayatunnufus and D. Alita, "Sistem Cerdas Pemberi Pakan Ikan Secara Otomatis," *J. Teknol. dan Sist. Tertanam*, vol. 01, no. 01, pp. 11–16, 2020.

[14] D. Permana and S. Doni, "Alat Pakan Ikan Aquarium Otomatis Berbasis Arduino UNO," *J. Ilm. Mhs. Kendali dan List.*, vol. 2, no. 2, pp. 2723–598, 2020, doi: 10.33365/jimel.v1i1.

[15] A. Supratman, B. Indarmawan Nugroho, and R. Dwi Kurniawan, "Penerapan Metode Rule-Based System untuk Menentukan Jenis Tanaman Pertanian Berdasarkan Ketinggian dan Curah Hujan," *Innov. J. Soc. Sci. Res.*, vol. 4, pp. 7879–7890, 2024.