p-ISSN: 2252-3006
e-ISSN: 2685-2411

# Automatic Pet Feeder Rotational Model Using MQTT and Mobile Application

**Putu Gede Krisna Mahadiputra[a1], I Made Agus Dwi Suarjaya[a2], Kadek Suar Wibawa[a3]**

[a]Information Technology Department, Udayana University, Indonesia
e-mail: [1]maha.krisna035@student.unud.ac.id, [2]agussuarjaya@it.unud.ac.id, [3]suar_wibawa@unud.ac.id

**_Abstrak_**

*Sistem pemberi pakan otomatis dengan model dispenser memiliki keterbatasan pada jenis pakan yang ditampung hanya bisa pakan kering, sedangkan binatang peliharaan (kucing) dengan kecenderungan mengkonsumsi pakan kering rentan terkena penyakit pada sistem urinaria. Penelitian ini berfokus pada perancangan sistem pemberian pakan peliharaan otomatis model rotasional yang dapat menampung berbagai variasi pakan dan pengaturan jadwal melalui aplikasi mobile via MQTT, dengan memanfaatkan mikrokontroler ESP32 yang terhubung dengan motor stepper 5V, RTC dan sensor load cell. Sistem ini memiliki tingkat keberhasilan 90% pada otomatisasi pemberian makan hewan peliharaan, namun tutup atas wadah memiliki perbedaan sudut sekitar 5.4$^o$ dengan posisi yang diharapkan (tingkat kesalahan 6%). Setiap sensor berat memiliki tingkat kesalahan secara berturut-turut sebesar 1.41% (blok pakan ke-1), 1.91% (blok pakan ke-2), dan 0.68% (blok pakan ke-3). Aplikasi Android hanya membutuhkan waktu 1-1,5 detik untuk menampilkan data berat feed terbaru.*

***Kata kunci:*** *Internet of Things (IoT), ESP32, MQTT, Pemberi pakan peliharaan otomatis, Aplikasi mobile.*

**_Abstract_**

*Automatic pet feeding systems with dispenser models have limitations on the type of feed that is only accomodate dry food, while pets (cats) with a tendency consuming dry food are prone to disease in the urinary system. This research focuses on designing a rotational model automatic pet feeding system that can accommodate both dry food and wet food, also set the feeding time through mobile applications via MQTT by utilizing an ESP32 microcontroller connected to a 5V stepper motor, RTC and load cell sensor. The system has success rate about 90% on pet feeding automation, but the container upper fold has difference angel about 5.4$^o$ with the expected position (error rate 6%). Every weight sensor has the successive error rates are 1.41% (1st feed block), 1.91% (2nd feed block), and 0.68% (3rd feed block). The Android application takes only 1-1.5 seconds to displaye the latest feed weight data.*

***Keywords:*** *Internet of Things (IoT), ESP32, MQTT, Automatic pet feeder, Mobile application.*

## 1. Introduction

Nowadays, smart devices are increasingly popular and have been widely used in various aspects of daily life[1]. One context that also utilizes smart device technology is pet care. Referring to the five principles of animal welfare proposed by FAWC (1993), the first principle focuses on pets being free from hunger and malnutrition by providing food to maintain health and keep animals strong. Based on this principle, pet owners should be able to provide healthy food for their pets on a regular basis. However, the tight and busy schedules of pet owners sometimes result in their pets not getting regular food, leading to deteriorating health conditions and even death due to starvation and neglect[2][3].

The needs for pet feeding regularly is a challenge for pet owners with busy schedules[4]. According to the Iams Company, pets such as cats should be given basic food (not treats) at least two to three times a day[5]. To solve this problem, an automatic pet feeder device can be an effective solution, making it easier for pet owners to feed their pets according to a predetermined schedule.

Dispenser pet feeders model designed to feed pets automatically using a dispenser mechanism, which has the ability to set and release feed at scheduled times with adjustable portion settings[2]. Although the portion size can be adjusted and is very beneficial in the long run. Unfortunately, the workings of the automatic pet feeder with a dispenser model can only accommodate dry food[6]. Cats (especially male cats), that consume dry food are more susceptible to diseases of the urinary system because they get only a little water intake[7]. The problems related to the accuracy of food portions and the limited types of food that can be given have prompted the author to conduct this research on pet feeder model[2][6]. This research entitled “Design of Automatic Pet Feeding System Rotational Model Based on MQTT and Mobile Application.” This research aims to design a pet feeder that can provide food portions completely according to the user's wishes by providing direct doses in each block where the food is provided.

There is a related study about pet feeder entitled “Design and Manufacture of Automatic Pet Feeder Using Arduino and Mobile Apps” by Wilyanto et. al. that aims to help pet dog owners feed their pets when they are away from them. The study uses components consisting of a stepper motor, a Real Time Clock (RTC), and a load cell sensor, which are controlled by an Arduino microcontroller through an Android application interface with the MQTT communication protocol. User can input schedules in the form of days and hours recorded in the Firebase database, and the feeders can communicate with the database via the MQTT protocol[8]. There are two main differences between related study and this research, where the related study uses a pet feeder dispenser model that is limited to dry feed but is able to provide feed for several days, while this research uses a rotational pet feeder model that is able to accommodate dry food as well as wet food and even meat and fish because the provision of feed is only for a day. The second difference is that the related research utilizes Firebase RTDB to record the feeding schedule, while this research utilizes local storage so that the MQTT broker becomes a direct bridge between the device and the android application.

Specificly about implementation of MQTT communication protocol, there is a study entitled “Implementation of IoT eith ESP8266 Part II – Home Automation” by Reddy that aims to sense various environmental parameters like temperature, light and smoke using sensors in the field of Smart Home automation. The study uses components including Light Dependent Resistor (LDR), LM35, LPG and Smoke Sensor, LM7805, Dual Channel Relay, Buzzer, 12V DC Fan and 230V AC Bulb which are controlled by ESP8266 through an Android application interface with the MQTT communication protocol. User can control light, fan, and buzzer trough Android application that developerd with MIT App Inventor, and the status of every device is read by ThingSpeak.com as a real-time cloud that fascilitating MQTT communication protocol and fed every device values into Android App[9]. There are two main differences between related study and this research, where the related study uses MIT App Inventor to develop a simple Android application with MQTT configuration, while this research uses Android Studion and Paho Android Service to develop a complex Android application with MQTT communication protocol. The second difference is that the related research uses ThingSpeak cloud to provide MQTT broker with graphic dashboard, while this research uses HiveMQ platform as the MQTT broker.

Another study about the implementation of MQTT communication protocol entitled “Design of Chicken Weight Monitoring System Based on MQTT Protocol” by Ramadhani et. al. that aims to helps farmers in monitoring chicken weight progression without involving much interaction between humans and chickens, because broilers are prone to stress. The study only uses load cell sensor which controlled by an NodeMCU ESP8266 that transfer weights data continuously to website dashboard using MQTT communication protocol. Then the farmer as a user can monitor the broiler’s weight graphic trough Node-Red website dashboard[10]. There are three main differences between related study and this research, where in the related study the MQTT communication protocol transfer weight data only from 1 weight sensor continuously to website dashboard (using only 1 topic), while in this research the MQTT communication protocol transfer weight data from 3 weight sensors continuously to Android application (3 different topics). The second difference is that the related study only utilize MQTT communication protocol to do monitoring from 1 weight sensor which only weight data, while this research utilize MQTT communication protocol to do monitoring from 4 different topics about 3 weight sensors (weight data) and 1 actuator (stepper motor position status), and also utilize MQTT communication protocol to transfer control data with 4 different topics about 3 feeding times and 1 skip feeding day (totally 8 different topics). The third difference is that the related study uses Node-Red platform which already has a built-in broker (but with less configuration) in facilitating the MQTT communication protocol for website dashboard, while this research uses a stand-alone broker namely HiveMQ and utilizes the Paho Android Service library in facilitating communication protocols for Android applications.

This research is proposed to be an alternative solution in developing automatic pet feeding system that can accommodate any kind of feed, and timed feeding also remote monitoring by the feed weight through an android app on a daily basis.

## 2. Research Method

This research applied the SDLC Waterfall Model, a structured sequential system development method where every stage must be completed properly before moving on to the next stage, similar to a waterfall. Although the waterfall method is typically linear, it can be recursive allowing each stage to be repeated until satisfactory results are achieved[11]. The flows beginning with the requirement Requirement Analysis to analyzing the electronic components used and the libraries required for the Android application and each electronic component. The second stage is System Design to involves designing a schematic circuit of the electronic components and the Android application interface. The third stage is Coding Implementation where the design is implemented by coding the electronic components and developing the Android application, then integrating them. In the System Testing stage which is the fourh stage, the system enters the testing phase, where all functions and features are tested. The final stage is Maintenance Stage, which includes regularly checking hardware conditions, ensuring both hardware and software are running well and remain connected.

### 2.1 General Description

Design and modeling are important parts of the research. This section includes a general description that explains how all components are connected and work together properly. The system's general description is shown in Figure 2 below.

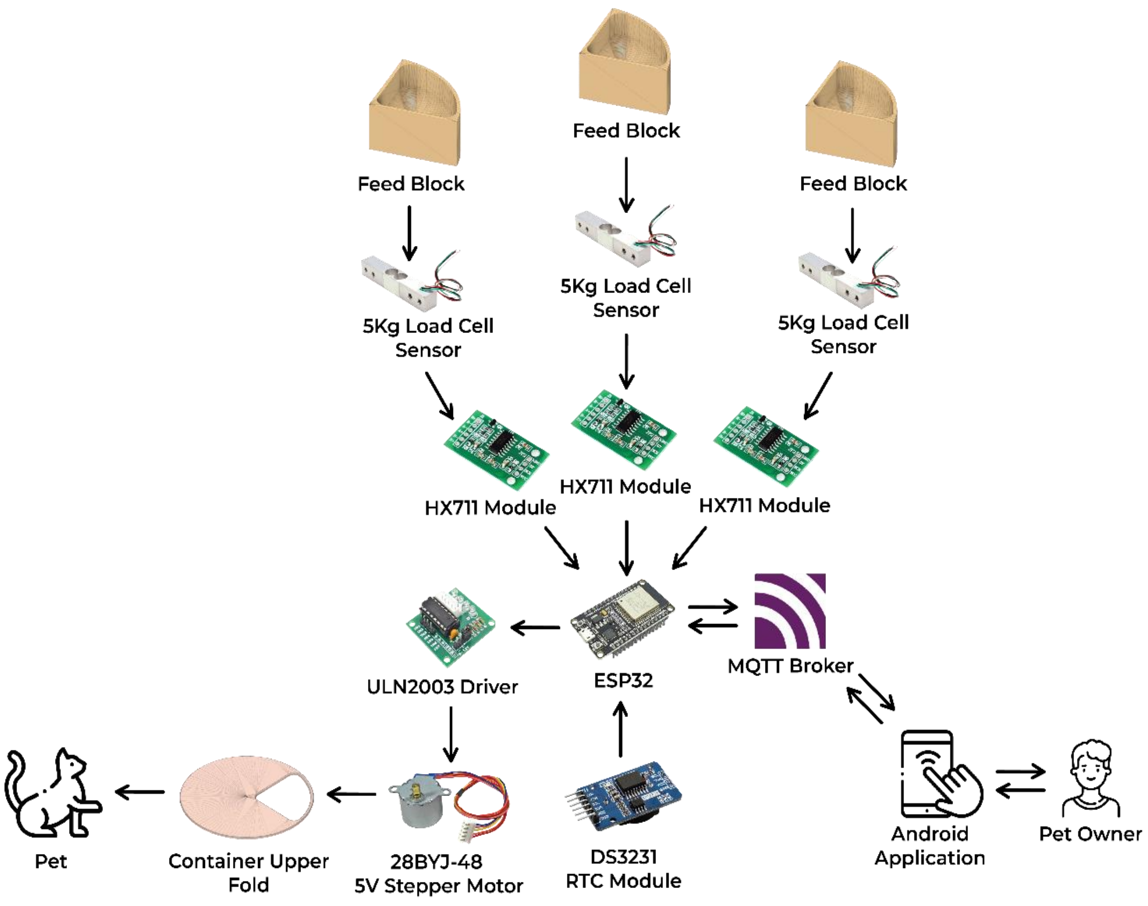


Figure 1 System's General Description

Figure 1 explain the general description of the system including ESP32 which is the main controller that connected to the broker and also controlling the stepper motor 28BYJ-48. ESP32 also read the weight measurement results that forwarded by every HX711 module from each load cell sensors. ESP32 also connected to the broker. User can set the feeding time and monitoring the feed weight in each feed compartment through an Android application which is also connected to the same broker as the ESP32. The broker acts as a third party in forwarding data from the ESP32 to the Android application and vice versa, which includes feed weight data, feeding time data, and the stepper motor position. The stepper motor will move when the current time that obtained by the ESP32 is the same as the time that sent by the Android application, and the Android application displays the feed weight and current status based on the data sent by the ESP32 in a certain time interval.

## 2.2 System Flow

Generally, the system delivers the feeding time inputted by the user through the Android application as a parameter for rotating the container upper fold. The system flow diagram is shown in Figure 3 below.

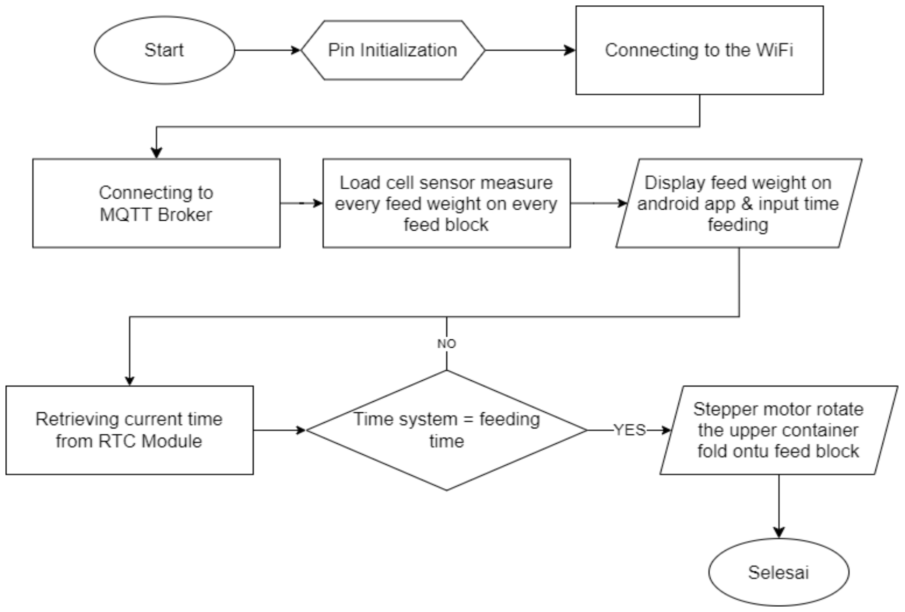

Start
Pin Initialization
Connecting to the WiFi
Connecting to MQTT Broker
Load cell sensor measure every feed weight on every feed block
Display feed weight on android app & input time feeding
NO
Retrieving current time from RTC Module
Time system = feeding time
YES
Stepper motor rotate the upper container fold ontu feed block
Selesai

Figure 2 System Workflow

Figure 2 shows the workflow of the rotational model automatic pet feeding system. The process begins by initializing the ESP32's pins that are connected to electronic components. Then, the ESP32 attempts to connect to the WiFi using the SSID and password parameters. Once connected to WiFi, the microcontroller attempts to connect to the broker using the previously made credentials. The load cell sensor located on each food block measures the feed weight, which the user can monitor through the Android application interface. The user can also set the feeding time on the Android application. The microcontroller retrieves the feeding time from the broker and the current time from the RTC module, then matches them. If the values match, the microcontroller sends a command to the stepper motor to move the container upper fold 90 degrees clockwise.

## 2.3 Container Design

3D container design is created by using Fusion 360 software and printed by 3D printing technology thas has a good durability[12]. The container looks like a tube lunch box as shown in Figure 4 below.

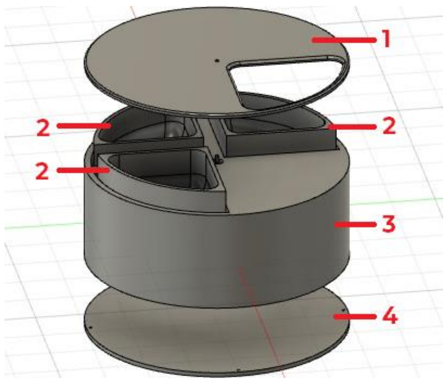


Figure 3 Container Design

Figure 3 is the container design from top corner view. Overall dimension of the container is 22cm diameter with 15cm height. The container can sparate into 4 parts as numbered in the figure above. Those parts are: 1) Upper container fold, which can be moved by the stepper motor; 2) Three feed blocks, that hold the feeds; 3) Main body, that hold the feed blocks and electronic components; 4) Lower container fold.

## 2.4 Schematic Circuit

Schematic circuit represents illustration about how every electronic component are connected to each other by their own pins. The schematic circuit is shown in Figure 5 below.

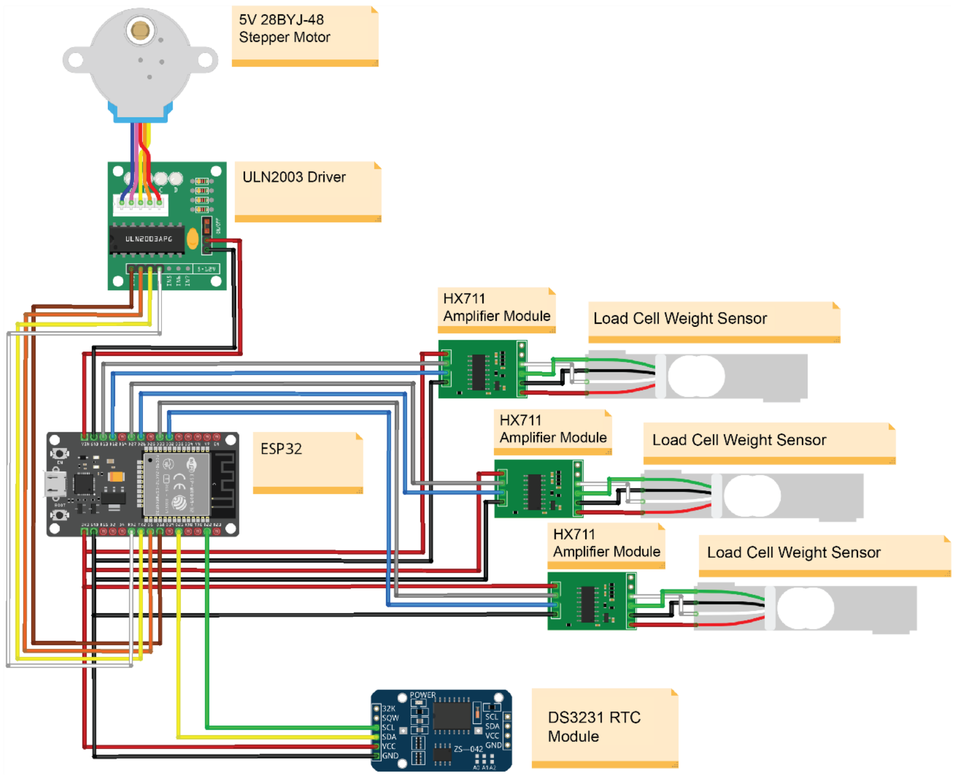


Figure 4 Schematic Circuit

As shown in Figure 4 above, the stepper motor connected to the ESP32 by ULN2003 as the driver. The similar condition applies to load cell sensors, which are connected to the ESP32 by HX711 as the amplified modul. It happends because those sensors and actuator work with analog signal and the driver and the amplifier acts as the ADC which is convert analog signal into digital signal and vice versa.

## 3. Literature Study

Literature study is an essential step in this research, which involves looking for and examining a variety of textual sources including books, journals, articles, and other pertinent materials. This procedure aids in gaining a thorough grasp of the body of information already known in a certain discipline, particularly in the areas of IoT and pet feeding, and also identifying previous research pertinent to the subject under investigation.

### 3.1. Telemetry

Telemetry is one of the technologies capable of retrieving data from a remote location, then sending the data to a center for processing[13]. One of the popular protocols in telemetry is Message Queuing Telemetry Transport (MQTT), which a Client-Server messaging protocol that is lightweight, open, simple and easy to implement[14]. The MQTT protocol uses a Publisher-Subscriber (Pub-Sub) model and each of them is a client. The publisher is responsible for sending information stored in a topic, while the Subscriber gets the information sent by the publisher by subscribing to the topic[15][16]. The MQTT protocol provides 3 QoS options, including QoS 0 (at most once), where messages are sent at most once according to the operating environment, so there is a possibility of losing messages; QoS 1 (at least once), where messages will be delivered at least once, but duplication is certainly possible; QoS 2 (exactly once), where messages will definitely be delivered exactly, so that if there is an unusual disconnection, interested parties can find out about it[17].

### 3.2. NodeMCU ESP32

The primary job of the microcontroller NodeMCU ESP-32, which was introduced by Espressif System, is to accept and process all ports and integrated circuits. Therefore, NoceMCU ESP32 can effectively control any drivers that are linked to any pins. Because this microcontroller already has a WiFi module built into the chip, it can also connect to the internet wirelessly without the need for extra boards, making it ideal for building Internet of Things application systems[18]. This NodeMCU ESP32 module has a role to control every component such as RTC module, weight sensor, and stepper motor.

### 3.3. Real-Time Module

An RTC (Real-Time Clock) module is an electronic component used to keep accurate time in electronic devices. DS3231 RTC module is a timer module which is an electronic component that functions as a real time clock, which is able to provide time information from seconds, minutes, dates, months, and years[19]. Another feature possessed by this RTC module is the ability to measure temperature, in addition to its main function as a digital timer. The DS3231 RTC module can be accessed with an I2C interface, so that its use is sufficient with only 4 pins. This RTC mo dule is commonly used in digital clock projects, alarms, measurement time recording, and scheduling of commands. In this research, this RTC module is used to help the device to accurately retrieve real time and use it as a parameter in moving the actuator according to the schedule given through the user interface.

### 3.4. Weight Sensor

One popular type of weight sensor is the load cell, an electrical signal produced by a transducer that changes mechanical force into electrical energy. Industrial scales and weighing systems employ load cells to measure weight precisely. When a load is applied, the load cell deforms, altering the resistance of strain gauges configured in a Wheatstone bridge, which produces a voltage signal proportional to the force[20]. This signal is amplified and processed by an analog-to-digital converter (ADC), commonly HX711 module to provide a digital reading[21]. In this research, this weight sensor is use to measure the feed weight on every compartment.

### 3.5. Stepper Motor

The 5V 28BYJ-48 Stepper Motor is a stepper motor with a 28 mm winding diameter and a 5V operating voltage. This motor is a member of the four-pole, unipolar stepper motor family, which regulates its rotational motion using a shuttle driver[22]. Systems that use Arduino may be linked to the 5V 28BYJ-48 Stepper Motor by utilizing a shuttle driver, such as the ULN2003 or L293D. In this research, this motor is use to In this research, this motor is the only actuator used to move the container lid with a turning radius of 45 degrees in one motion.

## 4. Result and Discussion

Same as the name, this section contains the results and discussion of research. The topics covered in the research include system testing, container design implementation, Android application design, and testing analysis.

### 4.1 Implementation of Container Design

As mentioned on previous section, the container design printed with 3D printing technology. Every printed component can be shown as figures below.

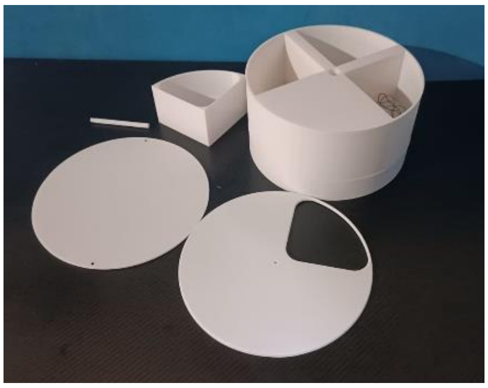


Figure 5 3D Printed Result

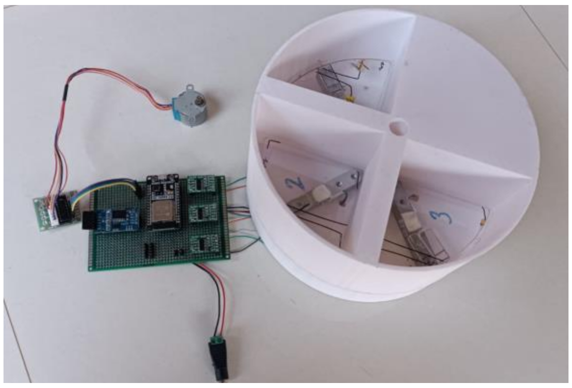


Figure 6 Schematic Circuit & Load Cell Installation

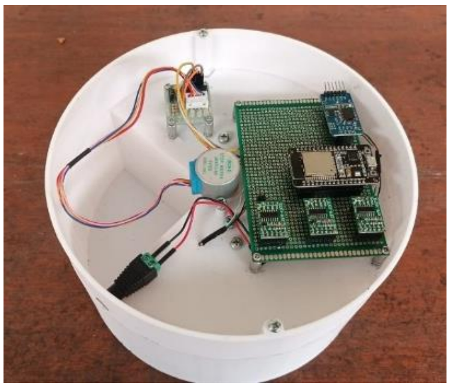


Figure 7 Electronic Components Installation

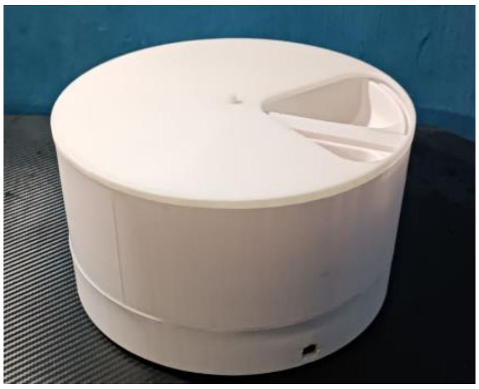


Figure 8 Final Assembling

Figure 5 are the printed of the 3D model that has been designed before using Fusion 360 software. How every electronic component connected and installed on the food container are shown by Figure 6 and Figure 7. Then, the final assembled include electronic components and food container shown as the Figure 8 above.

### 4.2 Implementation of Android Application

The android application developed using Java programming language and supported with Paho library to integrate the application with MQTT communication protocol. As mentioned on previous section, user can set the the feeding time trough this application. Then the application will publish the feeding time to ESP32 using MQTT communication protocol.

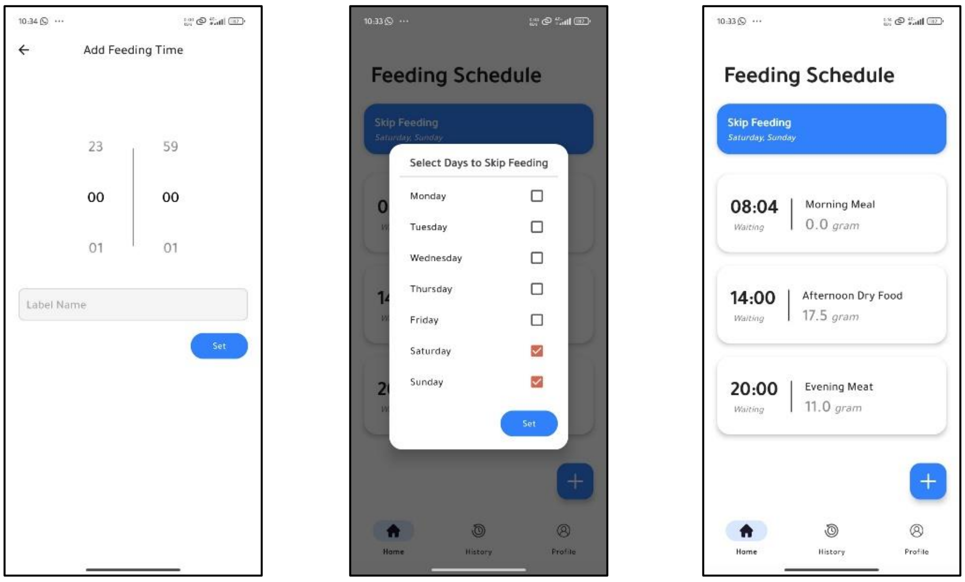
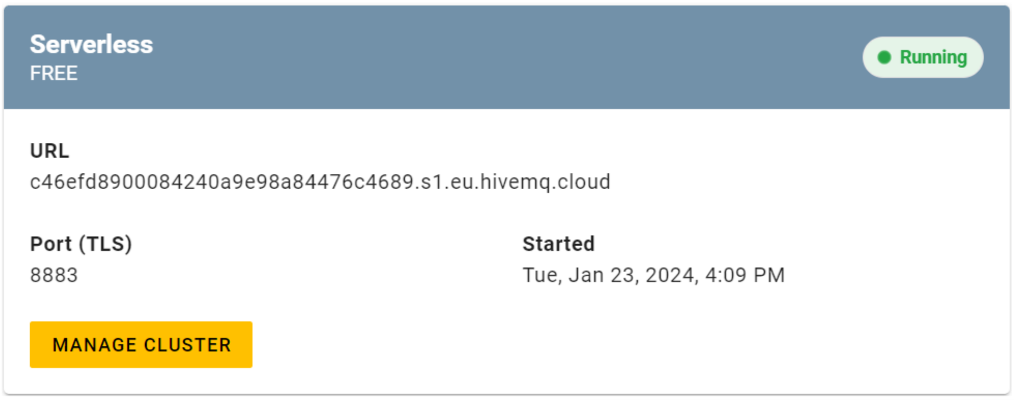


Figure 9 User Interface of The Android Application

Figure 9 is the display ot the android application that can be use to set the feeding time and monitoring the feed weight. User also can edit the feeding time and day to skip the feeding time within the application. Every feeding time schedule will recorded to the history, and user can see trough the History tab. Importance thing that user should do is create the MQTT credential and use it to loging in to the dashboard.

## 4.3 Broker Configurations

This research used HiveMQ broker to support in implementing MQTT communication protocol and it is free to use with 100 device and 10GB data traffic. The broker configurations using HiveMQ are really easy to do, just create an account and it will generate the broker URL that should be declared on the ESP32 and Android application. The configurations show as figure below.


Serverless
FREE
Running

URL
c46efd8900084240a9e98a84476c4689.s1.eu.hivemq.cloud

Port (TLS)
8883

Started
Tue, Jan 23, 2024, 4:09 PM

MANAGE CLUSTER

Figure 10 Broker Configuration

It's important enough to know that the broker URL declaration on the Android applcation should be adding postfix which is the MQTT port that is 8883, simply `<broker_url>:8883`. Other configuration is make sure that the device credential has Publish and Subscriber permission, so that the ESP32 and Android application can connected to each other.

## 4.4 System Test Result

This section describing the system testing, includes data transmission testing, load cell sensor testing, stepper motor rotation testing, and feeding time testing

### 4.4.1 Load Cell Sensors

This test was conducted to verify that every load cell sensor on each feed block measure the weight accurately. This test using a food scales and compare its measured value with every load cell sensors value.

Table 1 Load Cell Sensors Test Results

| No | Scales | Weight Sensors (grams) | | | Error Rate(%) | | |
|---|---|---|---|---|---|---|---|
| | | 1st Block | 2nd Block | 3rd Blok | 1st Block | 2nd Block | 3rd Blok |
| 1 | 25.3 | 25.0 | 25.1 | 25.0 | 1.19 | 0.79 | 1.19 |
| 2 | 169.9 | 169.5 | 168.5 | 169.7 | 0.24 | 0.82 | 0.12 |
| 3 | 52.0 | 51.5 | 51.5 | 51.8 | 0.96 | 0.96 | 0.38 |
| 4 | 144.0 | 143.8 | 144.3 | 143.8 | 0.14 | 0.21 | 0.14 |
| 5 | 10.2 | 10.0 | 9.5 | 10.0 | 1.96 | 6.86 | 1.96 |
| 6 | 28.1 | 27.5 | 27.6 | 27.7 | 2.14 | 1.78 | 1.42 |
| 7 | 160.8 | 160.7 | 159.3 | 160.6 | 0.06 | 0.93 | 0.12 |
| 8 | 63.7 | 63.5 | 62.9 | 63.5 | 0.31 | 1.26 | 0.31 |
| 9 | 8.5 | 8.1 | 7.9 | 8.3 | 4.71 | 7.06 | 2.35 |
| 10 | 74.9 | 74.6 | 74.1 | 74.8 | 0.40 | 1.07 | 0.13 |
| 11 | 56.3 | 55.9 | 55.6 | 56.1 | 0.71 | 1.24 | 0.36 |
| 12 | 76.5 | 76.1 | 75.2 | 76.4 | 0.52 | 1.70 | 0.13 |
| 13 | 34.7 | 34.3 | 34 | 34.5 | 1.15 | 2.02 | 0.58 |
| 14 | 13.7 | 13.1 | 13.5 | 13.5 | 4.38 | 1.46 | 1.46 |
| 15 | 76.1 | 75.5 | 75.4 | 75.9 | 0.79 | 0.92 | 0.26 |
| 16 | 30.9 | 30.0 | 30.1 | 30.7 | 2.91 | 2.59 | 0.65 |
| 17 | 34.5 | 33.8 | 33.9 | 34.3 | 2.03 | 1.74 | 0.58 |
| 18 | 59.6 | 59.2 | 58.7 | 59.4 | 0.67 | 1.51 | 0.34 |
| 19 | 36.3 | 35.6 | 36.0 | 36.1 | 1.93 | 0.83 | 0.55 |
| 20 | 33.6 | 32.9 | 32.8 | 33.4 | 2.08 | 2.38 | 0.60 |
| Average Error | | | | | 1.46% | 1.91% | 0.68% |

Table 1 is the results of weight measure using load cell sensor on each feed block. This result obtained by putting twenty different things on each feed block alternately. The table shows that the result of every weight sensor measured smaller or equal to the real weight based on scales, except for load cell on 2nd feed block. Load cell sensor which placed on 2nd feed block measure that the weight of the 3rd thing was bigger than the real weight (0.5 grams bigger than 144.3 grams).

### 4.4.2 Data Transmission From ESP32 to Android Application

This test was conducted to measure delay of data transmission from the device to the android application using MQTT protocol. The transmitted data are the feed weight data from first feed block that was measured by one of three load cell sensors.

Table 2 Delay of Data Transmission Test Results

| No | Value (grams) | Delay (ms) | | |
|---|---|---|---|---|
| | | 1st Feed Block | 2nd Feed Block | 3nd Feed Block |
| 1 | 25.9 | 1,098 | 708 | 1,114 |
| 2 | 24.8 | 1,117 | 667 | 1,246 |
| 3 | 21.8 | 1,082 | 1,134 | 1,395 |
| 4 | 20.0 | 1,217 | 836 | 1,167 |

| | | | | |
|---|---|---|---|---|
| 5 | 19.6 | 1,033 | 1,253 | 1,119 |
| 6 | 19.0 | 1,184 | 1,155 | 1,119 |
| 7 | 18.5 | 1,184 | 1,122 | 1,200 |
| 8 | 17.2 | 1,148 | 1,188 | 902 |
| 9 | 16.4 | 1,465 | 1,059 | 835 |
| 10 | 15.3 | 1,201 | 905 | 958 |
| 11 | 14.6 | 1,447 | 1,103 | 871 |
| 12 | 13.1 | 1,386 | 1,059 | 965 |
| 13 | 12.3 | 1,416 | 1,099 | 1,139 |
| 14 | 11.1 | 1,183 | 1,114 | 813 |
| 15 | 10.3 | 1,256 | 1,199 | 843 |
| 16 | 9.4 | 1,235 | 1,131 | 1,013 |
| 17 | 8.2 | 1,298 | 1,322 | 802 |
| 18 | 7.6 | 1,199 | 1,140 | 1,028 |
| 19 | 6.2 | 1,233 | 1,137 | 696 |
| 20 | 5.3 | 1,099 | 1,247 | 865 |
| Average Delay (ms) | | 1,224.05ms | 1,078.90ms | 1,004.50ms |
| Average Delay (s) | | 1.2s | 1.1s | 1.0s |

Table 2 shows that the delay of the data transmission from device to the android application using MQTT protocol has average delay of 1.173ms. There are some factors those affect the delay time, include replacing data on shared preference when the weight data was received by Android application, internet connection trough ESP32 and Android application, and program logic on ESP32 and Android application.

### 4.4.3 Accuracy of The Stepper Motor Rotation

This test was conducted to verify that the stepper motor can rotate $90^o$ clock wise even though there are any friction between the upper container fold and the container body.

Table 3 Accuracy of The Stepper Motor Test Results

| No | Starting Position | Final Position | Difference from Final Position ($^o$) | Error Rate (%) |
|---|---|---|---|---|
| 1 | Default | 1st Feed Block | 5 | 5.56 |
| 2 | 1st Feed Block | 2nd Feed Block | 3 | 3.33 |
| 3 | 2nd Feed Block | 3rd Feed Block | 5 | 5.56 |
| 4 | 3rd Feed Block | Default | 3 | 3.33 |
| 5 | Default | 1st Feed Block | 5 | 5.56 |
| 6 | 1st Feed Block | 2nd Feed Block | 0 | 0.00 |
| 7 | 2nd Feed Block | 3rd Feed Block | 0 | 0.00 |
| 8 | 3rd Feed Block | Default | 10 | 11.11 |
| 9 | Default | 1st Feed Block | 3 | 3.33 |
| 10 | 1st Feed Block | 2nd Feed Block | 3 | 3.33 |
| 11 | 2nd Feed Block | 3rd Feed Block | 7 | 7.78 |
| 12 | 3rd Feed Block | Default | 12 | 13.33 |
| 13 | Default | 1st Feed Block | 7 | 7.78 |
| 14 | 1st Feed Block | 2nd Feed Block | 5 | 5.56 |
| 15 | 2nd Feed Block | 3rd Feed Block | 3 | 3.33 |
| 16 | 3rd Feed Block | Default | 12 | 13.33 |
| 17 | Default | 1st Feed Block | 7 | 7.78 |
| 18 | 1st Feed Block | 2nd Feed Block | 5 | 5.56 |
| 19 | 2nd Feed Block | 3rd Feed Block | 1 | 1.11 |
| 20 | 3rd Feed Block | Default | 12 | 13.33 |
| Average | | | $5.4^o$ | 6.0% |

The table above shows the results of stepper motor rotation in 5 cycle, and there are only two results where the stepper motor rotate accurately in $90^o$ clock wise (6 and 7 test number).

According to the test results, that friction between the upper container fold and the container body was affect the stepper motor torque to rotate the upper container fold, althogh the axle was placed inside a bearing in the container body. In another case, the precision level between the axle and the upper container fold also affected the accuracy that caused by square hole for the axle on the upper container fold was bigger than the axle connector size.

### 4.4.4 Feeding Time Test Result

This test was conducted to verify that device can feed properly based on feeding time and exclude the feeding time according to user input on the android application.

Table 4 Result of The Feeding Time Test

| No | Condition | Expected Result | Test No. | User Input | Status |
|---|---|---|---|---|---|
| 1 | The first feeding time is set or edited by the user. | Feeding time on the device updated to the latest based on user input and stepper motor rotate at that time | 1 | 09:21 | Valid |
| | | | 2 | 09:23 | Valid |
| | | | 3 | 09:25 | Valid |
| | | | 4 | 09:27 | Invalid |
| | | | 5 | 09.29 | Valid |
| | | | 6 | 09.31 | Valid |
| | | | 7 | 09.33 | Valid |
| | | | 8 | 09.35 | Valid |
| | | | 9 | 09.37 | Valid |
| | | | 10 | 09.39 | Valid |

Table 4 shows the feeding time test results. There is 1 test number that got the invalid status from the expected result, which is the 4th test number. It was caused by the 4th feeding time was set during the execution of the 3rd test. Important things to know is when after the stepper motor rotated, there are some delay until the minute change. So if the feeding set during the delay, ESP32 will not receive any message from the broker altough the broker has already receieved the message from the Android application.

## 5. Conclusion

Form the previous section about the system test, this proposesd system has success rate about 90% on pet feeding automation where it can run properly to rotating the upper container fold based on user input. But the container upper fold could not rotate accurately in 90$^o$, because most of the test results the final position can not be reached with the average difference angel is 5.4$^o$ (error rate 6%). Every weight sensor on each feed block can measure the feed weight almost same as the scales measure, with the successive error rates are 1.41% for the 1st feed block, 1.91% for the 2nd feed block, and 0.68% for the 3rd feed block. On the monitoring side, the Android application was running properly and connected to the broker without any problem. User can monitor the feed weight on each feed block to ensure their pet eat the feed, and the Android application takes only 1-1.5 seconds to display the latest feed weight data. For future improvement, this research can utilize the higher stepper motor that has a bigger torque to rotate the upper container fold and using a RTDB to replace the shared preference or using level 0 MQTT QoS that can make the Android application takes less time to displaye the latest feed weight data.

## References

[1] V. Rahmadhani and Widya Arum, “Literature Review Internet of Think (Iot): Sensor, Konektifitas Dan Qr Code,” *J. Manaj. Pendidik. Dan Ilmu Sos.*, vol. 3, no. 2, pp. 573–582, 2022, doi: 10.38035/jmpis.v3i2.1120.

[2] R. H. Subrata, A. Andrew, and S. Sulaiman, “Perancangan Sistem Automatic Pet Feeder Berbasis Internet of Things,” *Jetri J. Ilm. Tek. Elektro*, vol. 18, no. 1, pp. 17–30, 2020, doi: 10.25105/jetri.v18i1.7343.

[3] L. Aldino Ismail and B. Tjahjono, “Pemberian Makan Hewan Berbasis Internet of Things,” *Ikraith-Informatika*, vol. 7, no. 2, pp. 49–57, 2023, doi: 10.37817/ikraith-

informatika.v7i2.2250.

[4] H. Ngarianto and A. A. S. Gunawan, “Pengembangan Automatic Pet Feeder Mengunakan Platform Blynk Berbasis Mikrokontroller ESP8266,” *Eng. Math. Comput. Sci. J.*, vol. 2, no. 1, pp. 35–40, 2020, doi: 10.21512/emacsjournal.v2i1.6260.

[5] C. ORIJEN, “Orijen Feeding Guides.”

[6] W. Masril, “Rancang Bangun Pet Feeder Berbasis Arduino,” 2023.

[7] B. D. A. Riesta and I. W. Batan, “Cystitis Haemoragics and Urolithiasis in Domestic Male Local Cat: a Case Report,” *Indones. Med. Veterinus*, vol. 9, no. 6, pp. 1010–1023, 2020, doi: 10.19087/imv.2020.9.6.1010.

[8] E. Wilyanto, A. Noertjahyana, and R. Lim, “Perancangan dan Pembuatan Automatic Pet Feeder Menggunakan Arduino dan Mobile Apps,” *J. Infra*, vol. 7, no. 1, pp. 3–6, 2019.

[9] K. P. S. Reddy, “Implementation of IoT eith ESP8266 Part II – Home Automation,” vol. 3, no. 4, pp. 10–19, 2019, [Online]. Available: http://ijsrmme.com/paper/IJSRMME19346.pdf

[10] A. D. Ramadhani, M. Aly Afandi, and D. Anggraeni, “Design of Chicken Weight Monitoring System Based on Mqtt Protocol,” *J. Elektro Telekomun. Terap.*, vol. 8, no. 2, pp. 1116–1126, 2021, [Online]. Available: https://doi.org/10.25124/jett.v8i2.4175

[11] T. Akbar and I. Gunawan, “Prototype Sistem Monitoring Infus Berbasis IoT (Internet of Things),” *Edumatic J. Pendidik. Inform.*, vol. 4, no. 2, pp. 155–163, 2020, doi: 10.29408/edumatic.v4i2.2686.

[12] K. S. Putra and U. R. Sari, “Pemanfaatan Teknologi 3D Printing Dalam Proses Desain Produk Gaya Hidup,” *Semin. Nas. Sist. Inf. dan Teknol. Inf. 2018*, pp. 1–6, 2018.

[13] Munarso and Suryono, “SISTEM TELEMETRI PEMANTAUAN SUHU LINGKUNGAN MENGGUNAKAN MIKROKONTROLER DAN JARINGAN WIFI,” *Youngster Physic J.*, vol. 3, no. 3, pp. 249–256, 2014.

[14] HiveMQ, *MQTT Essentials The Ultimate Guide to the MQTT Protocol for IoT Messaging*. 2023. [Online]. Available: www.hivemq.com

[15] S. B. Pratama, R. Munadi, and A. Syauqi, “Analisis Performansi Protokol Coap Dan Mqtt-Sn Pada Sistem Smarthome Dengan Cooja Network Simulator,” *e-Proceeding Eng.*, vol. 5, no. 2, pp. 1982–1991, 2018.

[16] Nilam Andi Safitri and Ardy Seto Priambodo, “MQTT and CoAP Communication Protocol Analysis in Internet of Things System for Strawberry Hydroponic Plants,” *J. Robot. Autom. Electron. Eng.*, vol. 1, no. 1, pp. 45–56, 2023, doi: 10.21831/jraee.v1i1.69.

[17] M. Diono, H. Azwar, and W. Khabzli, “Sistem Monitoring Jaringan Sensor Node Berbasis Protokol MQTT,” *J. Elektro dan Mesin Terap.*, vol. 7, no. Vol. 7 No. 2 (2021), pp. 120–126, 2021, doi: 10.35143/elementer.v7i2.5232.

[18] D. D. Mahendra and A. Zarkasi, “Rancang Bangun Sendok Parkinson Menggunakan ESP-32 Dan Metode Complementary Filter,” *J. Ilmu Komput. dan Teknol. Inf.*, vol. 12, no. 2, pp. 46–51, 2020.

[19] M. Rizky, K. M. . Ismail, and S. Lamtiar S., “Rancang Kontrol Lampu Penerangan Koridor dan Air Conditioner Pada Asrama Tower Di Sekolah Tinggi Penerbangan Indonesia,” *J. Ilm. Aviasi Langit Biru*, vol. 13, no. 2, pp. 25–36, 2020, doi: 10.21608/pshj.2022.250026.

[20] M. P. Sulistyanto, “Pengolahan Sinyal Load Cell 5kg Menggunakan Metode Moving Average,” *J. Penelit.*, vol. 19, no. 2, pp. 138–145, 2016, [Online]. Available: https://e-journal.usd.ac.id/index.php/JP/article/view/833

[21] Y. Mukhammad, A. Santika, and S. Haryuni, “Analisis Akurasi Modul Amplifier HX711 untuk Timbangan Bayi,” *Med. Tek. J. Tek. Elektromedik Indones.*, vol. 4, no. 1, pp. 24–28, 2022, doi: 10.18196/mt.v4i1.15148.

[22] E. Yilmazlar, V. Erdemir, H. Kuscu, and A. Güllü, “Design Of Stepper Motor Control Interface With Embedded Systems,” *Int. J. Eng. Res. Dev.*, vol. 14, no. 6, pp. 17–22, 2018, [Online]. Available: www.ijerd.com