Petification: Node-RED based Pet Care IoT Solution using MQTT Broker

Haeram Kim
Computer Science and Engineering
Chungnam National University
Daejeon, Korea
haeram.kim1@gmail.com

Hyejong Kang
Computer Science and Engineering
Chungnam National University
Daejeon, Korea
kanghyejong1001@gmail.com

Sunghan Kim
Computer Science and Engineering
Chungnam National University
Daejeon, Korea
seonghan.kim.cnu@gmail.com

Dukho Choi
International trade / software convergence
Chungnam National University
Daejeon, Korea
dukho.fin@gmail.com

Jihyun You
Cybersecurity
Purdue University
West Lafayette, IN, USA
you62@purdue.edu

***Abstract*—While there are an increasing number of households owning pets, it is challenging for owners who leave home often to take good care of their pets. The previous studies which are conducted to solve this problem use free plan of paid IoT platform. As features which free plan of paid IoT platform supports are very limited, the proposed IoT solution named 'Petification' uses open-source IoT platform Node-RED with MQTT messaging protocol. In Petification, the water supplier and feed machine is attached to platform to provide water and food the pet and scale the weight of the water and food. The web-based dashboard is supported to show the remaining amount of water and food, show the water and food consumption, show the device connectivity, and serve the food by button or by time schedule. The user of the Petification gets notification when the water or food is running out of empty. The load cell, HX711 amplifier, and Raspberry Pi Zero W are mounted to water supplier and feed machine to scale the water and food. In feed machine, MG90S servo motor is mounted to Raspberry Pi to serve the food to the pet. With the Petification, users can take care of their pets remotely as well as check the device's status. However, while serving the food to the pet, served amount mismatches with the desired amount. Thus, future plan can be enhancing the food gate to serve the exact amount of food and adding more devices.**

***Index Terms*—IoT platform, Node-RED, MQTT, Smart pet care service**

I. Introduction

As the number of households living alone increases and the culture of raising pets spreads compared to the past, the number of households raising pets is increasing. This trend is shown in the growth of the pet industry. The growth of the pet industry is one of the indicators of this trend. The profit of the pet industry more than doubles every year over 10 years, from /48.4*billionin*2010*to*/109.6 billion in 2020 [1]. Figure 1 shows that households owning a dog consists the largest portion, and the number of families owning dogs has increased from 46.3 million in 2011 to 69 million in 2021.

However, along with this trend, demand for tracking pet wellness is increased accordingly. One of the demanded services is tracking a pet's status when the pet is left alone. For those who left home often, it is challenging to take good care of their pets.

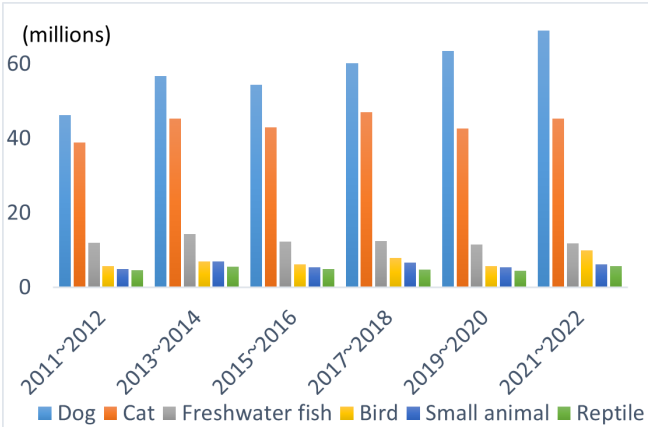

Fig. 1. Number of U.S. Households That Own a Pet, by Type of Animal

As a way to solve this problem, Internet of Things (IoT) technology has emerged. A lot of IoT solutions are introduced in the market, and many studies and implementations are suggested. However, most of the suggested IoT solutions uses paid IoT platform or at least a free plan with limited feature. For example, 'Blynk' IoT platform is most commonly used [2] [3] [4] [5], and 'Adafruit IO' [6] and 'Freeboard IO' [7] are used as well. And while water and food consumption and automatic feeding are commonly supported by previous implementations, there are few solutions that support device status information such as device connectivity and the remaining amount of food and water. Moreover, an error notification is also a less supported feature.

For this reason, The proposed IoT solution provides various notification services for the pets and status of the devices as well as tracking water and food consumption. Furthermore,

not to be locked on the limited feature that the free plan of commercialized IoT platform provides, the proposed IoT solution is implemented using an open-source IoT platform. Thus, the proposed IoT solution uses Node-RED to take advantage of open-source characteristics and many resources.

As a flow-based visual programming tool [8], Node-RED makes possible the implementing IoT platform much faster. And also, unlike paid IoT platforms, Node-RED is an open-source project. By this characteristic, Node-RED supports making a user-friendly IoT platform with no feature limitations. Moreover, as a lot of nodes and flows are shared in Node Package Manager (NPM), implementing an IoT platform using Node-RED has the advantage of utilizing resources.

Also, MQTT Protocol is used to manage overall manage flow. Not only the powerful support for MQTT Protocol by Node-RED, but also it is suitable for various IoT solutions because of its lightweight characteristic. MQTT is an OASIS standard that provides lightweight message delivery by publish/subscribe model [9].

The name of the proposed IoT solution is 'Petification', which includes two key points of the system, 'pet' and 'notification'. The definition for pet care includes overall fields such as providing feed, providing clean drinking water, providing rest areas, exercising regularly, and checking health through veterinarians. [10] Among these many pet care fields, Petification is an IoT solution focused on providing feed and water to pets and informing owners of meaningful information. The functionalities which Petification provides are as follows:

- Supplying water with Water Supplier device.
- Feeding the pet with Feed Machine device.
- Tracking water and food consumption.
- Tracking water and food remaining amount for each device.
- Serving the food by manually or scheduled time.
- Notifying when water or food is empty.
- Providing the device status.
- Displaying visual data with the web-based dashboard.

II. Related Literature

This chapter will be explained with hardware and platform perspectives respectively. In the hardware part, used devices are compared with them of Petification, and used IoT platform and provided feature will be compared with present research in the platform part.

A. Hardware

In the previous study conducted by P. N. Vrishanka *et al.* [11], an automated pet feeder is proposed. To implement it, an ultrasonic distance sensor and SG90 servo motor were mounted to Arduino Uno R3. An ultrasonic distance sensor is used to determine the remaining food amount by measuring the distance from the entrance of the feed container to the inside of the bowl.

However, the feed machine of Petification used load cell sensors to determine the remaining food amount, as it is difficult to use ultrasonic waves to measure the amount of feed accurately.

In the previous study conducted by Rogerio Nogueira *et al*, [12] , Rotary Valve and DC motor were used to provide feed. And in the previous study conducted by Vania *et al*, [13], propeller blade and DC motor were used.

However, the problem with using a rotary valve or propeller blade is that the feed can't be provided with the exact weight. This is because the amount of feed to be provided to the pet is determined based on the serving unit contained in one space. Thus, when the feed machine tries to provide a certain weight of feed with the rotary valve or propeller blade, there is always the possibility of providing more than the intended weight, even if the error for load cell is not concerned. The Petification feed machine does not use a rotary valve but uses a gate that can block the feeder container outlet because it has the advantage of being able to adjust the amount of feed to be supplied in more detail than the rotary valve method.

In addition, both [12] and [13] uses a DC motor to provide feed through continuous rotation of the rotary valve. However, the feed machine of Petification uses a servo motor because they need more precise control than free continuous rotation. [14]

B. Platform

The previous study conducted by Y. Chen *et al.* [5] proposed a pet care IoT system that provides food and water consumption, number of defecation, and defecation duration. It was developed through Arduino IDE and provides features above with numbers and visual statistics in real-time using "Blynk" as an IoT platform.

The previous study conducted by T. Sangvanloy *et al.* [4] proposed a pet care IoT solution that visualizes the daily food consumption in real-time and automatically feeds the pet according to the scheduled time. It was also implemented using "Blynk" as an IoT platform.

But, non of the two studies above provide information on device status. Although tracking water and feed consumption is an important feature in pet care IoT solutions, keeping track of IoT devices' status is also an important feature. Petification provides the status of feed machines and water supply machines, such as device connectivity, error existence, and the amount of feed or water remaining in the device. Moreover, Petification provides an error notification feature that notifies users when the error occurred, such as when water or feed is empty.

From the used platform perspective, the above two studies commonly used 'Blynk' as an IoT platform. However, Petification uses Node-RED instead of 'Blynk' to take advantage of visual programming, openly pre-developed flows and nodes, and various notification means.

III. Methodology

A. Overview

As MQTT is the main messaging protocol for petification and MQTT uses TCP/ IP protocol stack, each device is con-

nected to the Access Point (AP) through Wi-Fi. Thus, all the messages published from devices are sent to the AP with Wi-Fi, and then they are sent to the Petification platform through the internet. Petification platform processes the messages and sends the processed data to the users in form of a web dashboard page, E-Mail, and WhatsApp messages. On the contrary, users can send requests with the web dashboard to the platform. While some requests require only the platform, the requests to activate the actuator are forwarded to the device with the help of the platform. In summary, users and devices communicate bi-directionally through the platform. The overview for Petification is shown in figure 2.

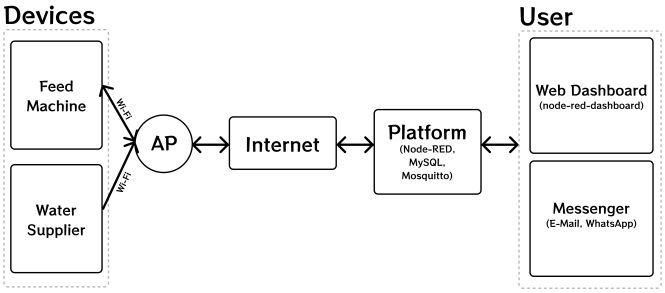

Fig. 2. Petification overview.

B. Used Technology

1) Hardware: The load cell sensor serves to convert the weight or force acting into an electrical analog signal [15]. In the present research, three load cells that can weigh up to 5kg with 1g accuracy are used.

All the load cells are mounted to the microcontroller through a load cell amplifier. In Petification, three HX711 modules are used as load cell amplifiers. HX711 converts an analog signal into a digital signal, amplifies a measured value of a low load cell, and then transmits the value to another microcontroller [16].

The servo motor is a rotary or linear actuator that rotates and pushes a machine [17] and one MG90S servo motor is used.

The role of the microcontroller is orchestrating all the sensors and actuators and communicating with the IoT platform. For the microcontroller, two Raspberry Pi Zero W are used.

2) Node-RED: Node-RED is a flow-based visual development tool that is easy for developers and non-developers to develop programs. Node-RED is popularly used to connect hardware devices, APIs, and online services [18]. One of the Node-RED's operational strengths is that it can run on various environments such as local, Raspberry Pi, Docker, Cloud Instance, and *et al*.

In the Node-RED core package, a lot of nodes are pre-installed by default. 'HTTP node' and 'MQTT node' are provided to help configure the network. Also, the Node-RED core package provides the 'exec node' which helps execute system commands. Being able to execute system commands means that other programming languages can be integrated into the flow. For example, a python code file can be executed with the help of the 'Exec node'.

Node-red is an open-source project and not only Node-RED itself but also various nodes are being distributed through Node Package Manager (NPM). Thus, users can share their nodes or flows, and a lot of contributed nodes are available in NPM. User interface (UI) can be configured using Node-RED's 'node-red-dashboard' package. It also supports appending Cascade Style Sheet (CSS) and JavaScript code to the dashboard, so user-friendly UI is easily configured [19]. Furthermore, Node-RED has the advantage of allowing users to easily access the database and manipulate data. There are various nodes to access the database, such as 'node-red-node-MySQL which helps to access the MySQL server. Also, a lot of nodes are distributed to send notifications to users, such as 'node-red-node-email' that helps send e-mail and 'node-red-contrib-whatsapp-cmb' that helps send WhatsApp messages.

Node-RED is made by International Business Machines Corporation (IBM), and IBM plans to increase its contributor base as an open-source for the sustainability of Node-RED. [20].

3) MQTT: MQTT stands for Message Queuing Telemetry Transport and is the oldest Machine to Machine (M2M) protocol for IoT. It is an ISO standard Publishing-Subscribing-based messaging protocol designed for lightweight M2M communication in a limited network. All MQTT messages have their own topics, and MQTT clients can send and receive data by posting messages to MQTT or subscribing to topics for those messages. TCP is used as a transport protocol, and TLS/SSL can be used for security. This means connection-oriented between Client-Brokers. In the MQTT protocol, three levels of quality of service (QoS) are used: QoS0(), QoS1(), and QoS2(). The MQTT protocol has a small message size and message overhead. In addition, due to low power consumption and resource consumption, it is evaluated as a suitable protocol for use in the IoT field. [21]

a) MQTT vs Other Protocols(CoAP, AMQP, HTTP): Protocols to consider in an IoT platform aside from MQTT that was mentioned earlier include CoAP, AMQP, and HTTP. They could be categorized by CoAP which uses UDP as the transport protocol, and MQTT, AMQP, and HTTP that uses TCP. From this project's perspective, CoAP uses UDP as the transport protocol. The size of the message and the overhead is small which is an advantage, however, as the UDP's characteristic the packet delivery is not guaranteed. Furthermore, CoAP is specialized to devices that support UDP and UDP analog so it is not suitable to support all devices. Next, AMQP uses TCP as the transport protocol. Compared to MQTT, reliability, security, provisioning, and interoperability is higher. However, because of that the message size and the overhead is a lot bigger compared to MQTT and the power and resource consumed by the device are large. Finally, HTTP is a protocol for the web, thus it could result in delays between the device and the platform's data flow. Additionally, the power and resource consumed by the device are similar to AMQP which is a downfall.

Therefore, MQTT utilized TCP as the transport protocol to guarantee the packet delivery and has an appropriate amount of message size and overhead. Furthermore, the consumption of power and resource is minor which is an advantage since the time of using the device is important. This concluded that the MQTT protocol is suitable for the IoT platform project composing the message broker. The message broker uses Mosquitto which is the MQTT protocol implemented. [21]

4) Mosquitto: Mosquito is an open-source message broker implementing MQTT protocol versions 5.0, 3.1.1, and 3.1 and is provided by Eclipse. It is also lightweight and suitable for use in all devices, from low-power single-board computers to full servers. [22]

Several programs are implementing MQTT, such as EMQ, HiveMQ, Mosquito, and VerneMQ. A study comparing each of the programs suggested that in situations where availability is not significant, using Single Broker and Persistence Setting is a simpler way to configure Message Broker than using clustered Broker [23]. In addition, in the context of using Persistence Setting in Single Broker, Mosquitto had close to 0 percent probability of message loss and message order mixing, and the probability of receiving duplicate messages was significantly lower than other programs [23]. Thus, this research uses Mosquitto with persistency setting as a single message broker.

5) MySQL: MySQL is a fast, flexible, and easy-to-use database with RDBMS (relational database management system). The combination of MySQL's secure processing and reliable software provides effective transactions for large projects, so it is flexible with open sources. MySQL is also excellent in terms of data security because it is evaluated as having the safest and most reliable database management system [24]. In addition, MySQL is considered a suitable database to manage effective data flows because it has good compatibility with Node-RED [24].

IV. Hardware Design

A. Feed Machine

Feed Machine is one of the devices attached to the proposed platform. The Feed Machine has 2 main functionalities: Serving food to the pet and Reporting the weight of the food bowl and container to the platform. To serve food to the pet, the servo motor is mounted to open and close the food gate. As in Figure 3, the servo motor gives food in an open/close type. Food serving is done with the help of gravity; After the food gate is opened, food rolls down the slope, and thus food is served. By closing the food gate Feed Machine stops the serving of the food. To measure the weight of the food bowl and container, two load cell is mounted under each of them. Like in Figure 4, each load cell is connected with an HX711 Amplifier to convert the analog signal to a digital signal. Also, as in Figure 5., Load cell sensors are installed between wooden boards to measure the weight. Additionally, in Figure 5, a food container carries food through a slide into a food bowl. All the sensors and a servo motor are attached to Raspberry Pi Zero W.

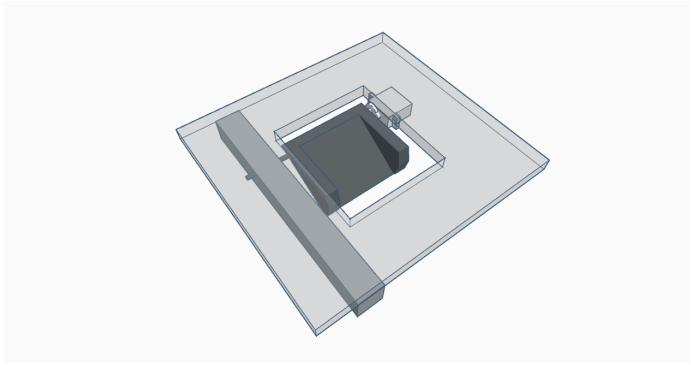

Fig. 3. Block diagram for Feed machine and Water supplier.

B. Water Supplier

The water supplier is also a device attached to the proposed platform. The water supplier has functionality that reports the total weight of the water in the water supplier. Similar to a feed machine, to measure the weight of water in a water supplier a load cell is mounted under the water supplier. In addition, like in Figure 4., the HX711 amplifier is used to convert analog signals into digital signals. As in Figure 5., like a feed machine load cell sensors are installed between wooden boards to measure the weight. Both types of sensor and RaspberryPi are the same as the feed machine.

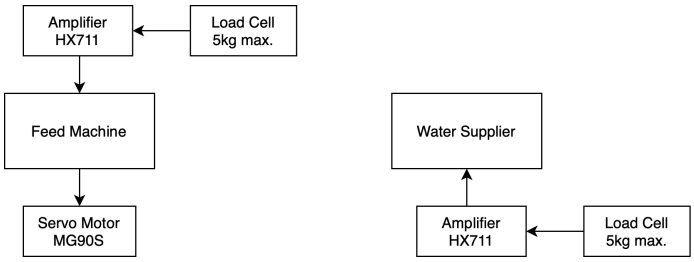

Fig. 4. Block diagram for Feed machine and Water supplier.

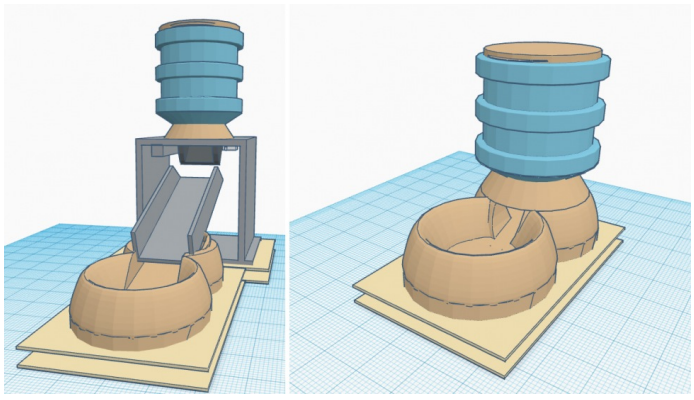

Fig. 5. Feed Machine and Water Supplier CAD Image

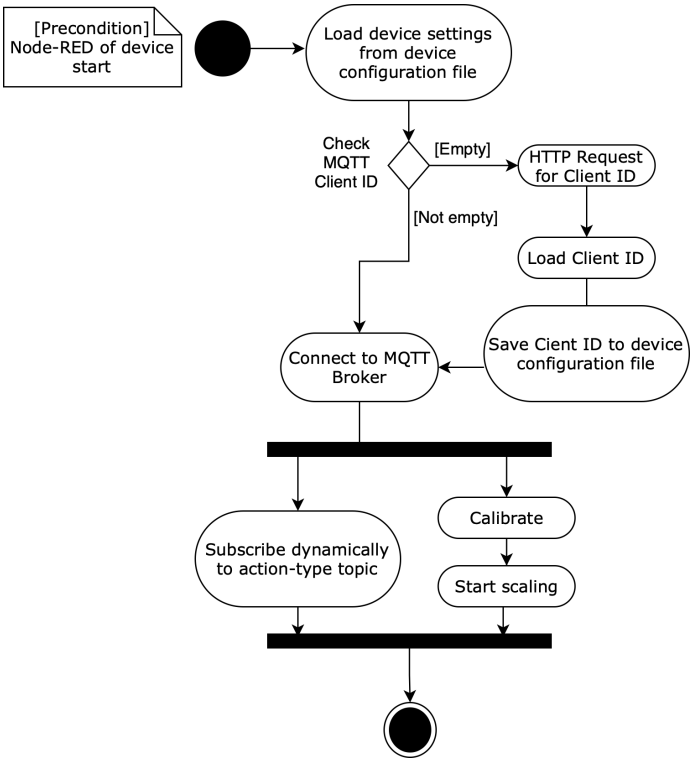

Fig. 6. Activity Diagram for Device Setup.

C. Activity Diagram for Hardware Design

a) Device Setup: In Figure 6 when Node-RED is initially executed in the device, the device settings file is loaded from the configure files. Thereafter, check the MQTT Client ID to connect directly to the MQTT broker if the MQTT Cid exists, and if the Cid does not exist, the Cid is issued through HTTP Request, stored in the setup file, and connected to the MQTT broker. When connected to an MQTT broker, the device commonly starts scale weight after calibration, and at the same time subscribes to the action type topic to listen to action messages coming from outside.

b) Calibration: Petification uses load cell weight function to measure the weight of the pet's food and water, calibration takes place to measure the weight with load cells. In the process of calibration, we need to choose an object to be used as the standard weight. In this project, we set a 500ml bottle of water as 500 g. Using the selected object (500ml bottle of water in this case), the load cell was used to measure the force the object applies to the load cell due to its weight. The bottle of water has been measured several times and the average of the data is calculated. Then the reference unit can be calculated by dividing the calculated average by the weight of the object. Depending on the load cell and the reference unit, there is a possible error. The reference unit is used as the factor of set_reference_unit() which is the method of hx711 to calculate the measurement and calculate the measurement depending on the standard weight to be ready for wight measurement. Using this, it is possible to measure the desired weight, and the weight could be calculated through the measured force measured by the load cell.

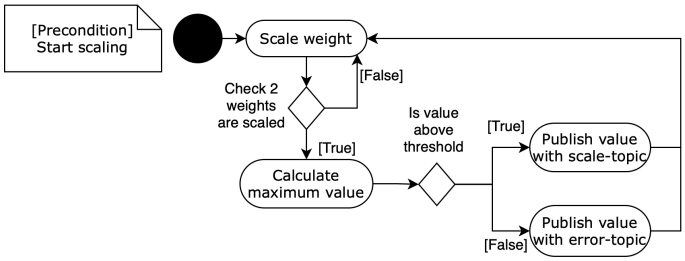

Fig. 7. Activity Diagram for Publishing Scale and Error Detection

c) Publishing Scale and Error Detection: In Figure 7, when weight data begins to be collected after scaling in Device, two weight values are collected. If two do not gather, repeat the scaling process to collect two values and process them at once. Afterward, a large value is selected from the two scaled data, and if the scale value is large compared to the threshold, the scaled weight is published, and if not, an error-topic message is published.

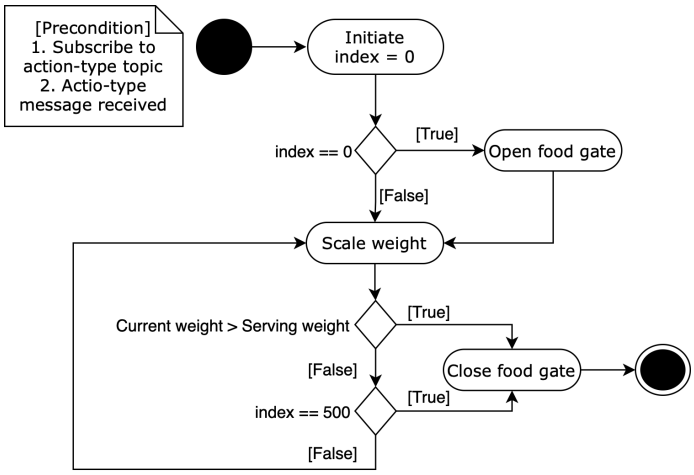

Fig. 8. Activity Diagram for Automatic Feeding.

d) Automatic Feeding: In Figure 8, the device subscribes to MQTT messages for Action-type topics. Accordingly, the device can receive an Action-type message, and if it receives a message, it initializes the index to 0 using a loop statement, opens the food gate, and feeds. Weight scaling continues and repeats while the current weight is less than the weight input by the user. If the current weight is greater than the user input value, close the food gate. The max iteration count is 500, and during that loop, the current weight is less than the input weight, and the food gate is automatically closed when the iteration ends.

V. Software Design

A. Blocks

The platform for petification consists of 9 blocks: MQTT Message Broker and Database are running on independent

processes, while MQTT Manager, Time-series Manager, Rule Engine, Device Manager, User Manager, Schedule Engine, and UI Dashboard Manager are running on Node-RED. Figure 9 shows the block diagram of Petification and interaction between devices and platform blocks.

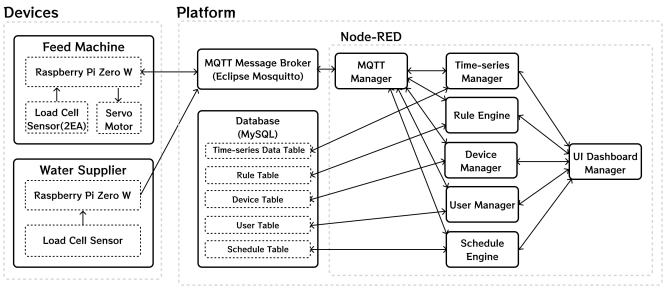

Fig. 9. Platform block diagram and interaction between devices.

1) MQTT Message Broker:
Petification uses MQTT Protocol to control the message flow. Thus, Mosquitto v1.4.15 was installed and running on the platform server to act as an MQTT Message Broker. Feed machine, water supply machine, and Node-RED are connected to the MQTT Message Broker as a client, and MQTT Message Broker mediates Node-RED to Device communication and reverse-way communication (Device to Node-RED).

2) Database:
To store data efficiently and safely, a Database block is included in the platform. As a Database block, MySQL v5.7.37 is installed and running on the platform server. The database stores various data, such as all the MQTT messages (in a Time-series Data Table), message-based action execution rules (in a Rule Table), information for the connected devices (in a Device Table), user information (in a User Table), and time-based action execution rules (in a Schedule Table).

3) MQTT Manager:
MQTT Manager in the Node-RED is the gateway for MQTT messages to enter Node-RED. The main purpose of this block is to convert incoming MQTT messages to give convenience to other Node-RED-based blocks. To achieve it, this block receives all the messages by subscribing to all the MQTT message topics. And then it parses message topic and payload to provide useful information such as the MQTT username and client id. These pieces of information are used in other Node-RED-based nodes in further progress.

4) Time-series Manager:
Managing Time-series Data Table of Database is the main purpose for Time-series Manager block. This block provides 3 types of ReST APIs for managing time-series data: Inquiring record feature (with HTTP GET method), Changing record validity (with HTTP PATCH method), and Deleting record (with HTTP DELETE method).

5) Rule Engine:
The purpose of Rule Engine is to activate actions according to MQTT messages. It cooperates with the Rule Table of Database to activate the action. Designing logic of the Rule Engine is inspired by the book, "Build Your Own IoT Platform" [25]. For every published message, Rule Engine searches for all rules where the rule's message pattern satisfies the message content. Actions that corresponded to the rules are defined as ReST API form, thus activating action will be progressed as sending an HTTP request. It also provides ReST APIs for adding, modifying, and deleting rules. Some ReST APIs for actions are defined in another block, whereas some APIs are defined in Rule Engine, such as sending notifications.

6) Device Manager:
Handling devices that are attached to the platform by users is the main purpose of the Device Manager block. It provides ReST APIs that can add a new device to the platform (with HTTP POST method), modify the status of the device (with HTTP PATCH method), delete a device (with HTTP DELETE method). And it also provides functionality for publishing an action-type message to a specific device.

7) User Manager:
The purpose of the User Manager block is to provide an interface for modifying user settings. Users can manage these 4 settings: Notification, Email address, WhatsApp information, and timezone where the user lives. To support global users, Timezone settings are included. As all the times that platform uses and database stores use Coordinated Universal Time (UTC) timezone, converting the time of the user to the UTC is necessary. This setting is especially important when a user adds a new schedule by using Schedule Engine, and when providing local time to the user in the Dashboard.

8) Schedule Engine:
Handling and executing schedule is the main functionality for this block. For schedule execution, it cooperates with the Schedule Table in the Database. Every minute, the Schedule Engine checks the Schedule Table and executes the actions that are scheduled to be activated at that time. Also, this block provides ReST APIs that can create, read, update and delete the schedule.

9) UI Dashboard Manager:
Dashboard Manager is to provide Graphical User Interface (GUI) to the user. By using this block, users can be provided the status of water and food remaining and consumption visually. It also provides buttons to serve food and input areas to set user settings.

B. Activity Diagram for Software Design

1) Receiving Scale Message: Activity Diagram for Receiving scale message is shown in figure 10. When a scale-type message is received to the platform, MQTT Manager parses the message and toss it to the Rule Engine. After that, Rule Engine activates 3 rules, and these rules proceeds in parallel: update device status, update remaining amount and update consumption. For activating update device status, Device Manager block updates the device's status and toss the changed status to the Dashboard block only if the status of the device is changed. For Updating remaining amount, the scale-type message is stored in the Time-series Data table by the Time-series Manager block. After storing the message, the payload

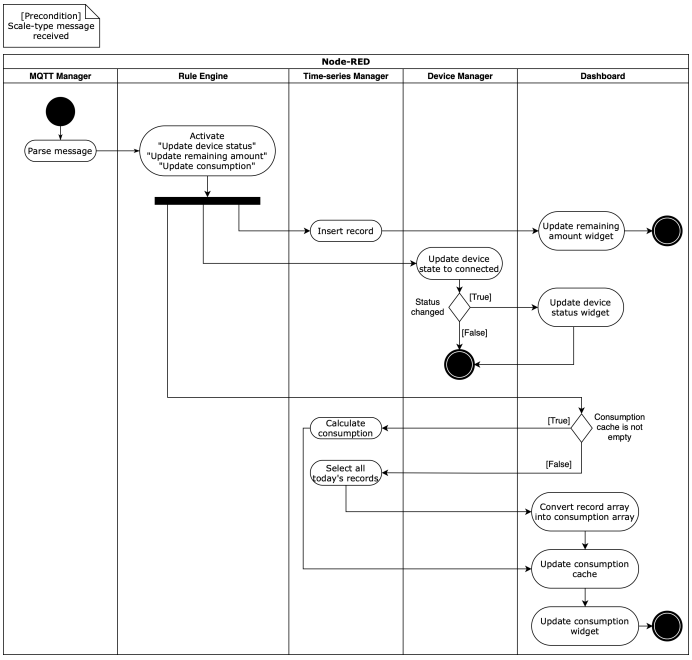

Fig. 10. Activity Diagram for Receiving scale message.

of the message is tossed to the Dashboard block to update the remaining amount widget. Lastly, for updating consumption, the platform checks the consumption cache firstly. When the consumption cache is empty, the consumption array is loaded to the cache, and when it's not, consumption is calculated and appended to the consumption array of the cache. To load cache, Dashboard request all the records that are stored after midnight of the day for the topic of the received message. The consumption array is prepared by calculating all of the consumptions for that record. After the consumption cache is updated, Dashboard block updates the consumption widget based on the prepared consumption cache. The basic idea for calculating consumption is comparing current and previous scales. When a pet consumes food or water, the scale for that will be decreased. Thus, consumption can be derived by accumulating each decreased amount. The pseudo-code for calculating consumption is shown in algorithm 1. The initial value for the decrease amount is set to 0, and the decrease amount is updated to the difference between the previous and current scale only when the previous scale is bigger than the current scale. By adding the decreased amount to the previous consumption, current consumption can be derived.

2) Publishing action message: Activity diagram for Publishing action message is shown in figure 11. Publishing action message has two entry points: When the serve button is pressed by the user, and when Node-RED flow is deployed. When the serve button is pressed by the user, the event is tossed to the Device Manager block, and publish an action-type message with the serving amount is stored to message payload. The second entry point is to publish action-type messages according to schedule. After Node-RED flow is deployed, the Schedule Engine selects all the schedules where the schedule's

Algorithm 1 Calculate consumption

procedure CALCCONSUMPTION
  $prevConsumption \leftarrow$ previous *consumption*
  $currScale \leftarrow$ *scale* of received message
  $prevScale \leftarrow$ *scale* of previous record
  $decrease \leftarrow 0$
  if $precScale \neq$ *null* & $prevScale > currScale$ **then**
    $decrease \leftarrow prevScale - currScale$
  return $prevConsumption + decrease$

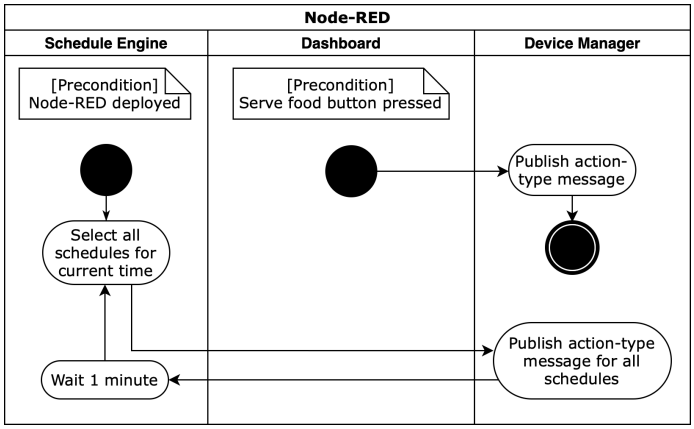

Fig. 11. Activity Diagram for Publishing Action Message.

execution time matches the current time. And then, Schedule Manager publishes all the action-type messages that are stored in schedule with the help of Device Manager. Finally, after all the action-type messages are published, the Schedule Engine waits for another minute and repeats selecting the scheduled procedure.

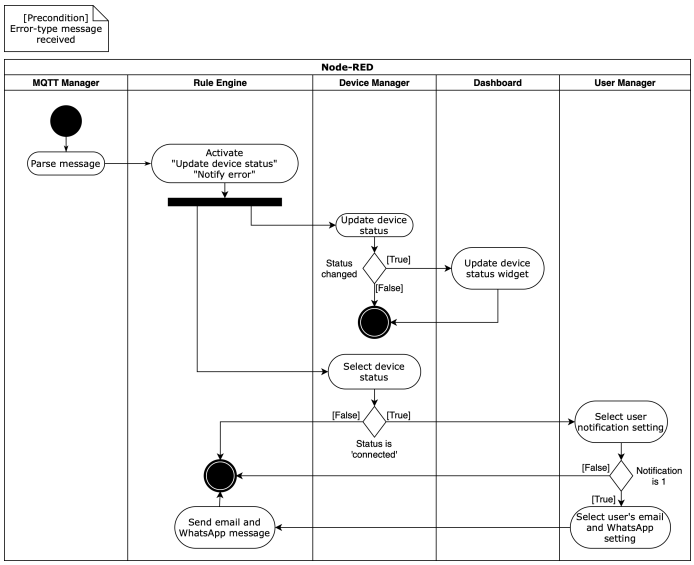

Fig. 12. Activity Diagram for Receiving Error Message.

3) Receiving Error Message: Activity diagram for Receiving error message is shown in figure 12. Similarly, with the receiving scale message, the error-type message is parsed in the MQTT Manager block and tossed to the Rule Engine block when the platform receives it. After that, the Rule Engine block activates two rules in parallel: Update device status, Notify error. To update the status of the device, the Rule Engine block sends a signal to the Device Manager block. Device Manager block updates the status of the device to error and sends the signal to the Dashboard block to update the device status widget only if the status of the device is changed. To notify the error to the user, the Rule Engine block sends a signal to the Device Manager block to check the status of the device. As sending the notification message is done only if the status of the device equals 'connected'. Device Manager block ignores errors when the status of the device is not 'connected'. After checking the device status, the Device Manager sends a signal to the User Manager block to check the user setting for notification. In case of the notification being set to 0, the User Manager block ignores the error as well. When notification is set to 1, User Manager sends the signal to the Rule Engine block, with user email and WhatsApp account setting. Finally, the Rule Engine block sends email and WhatsApp messages based on the user notification information.

VI. Implementation

A. Hardware

1) Device Setup: Each device was configured to automatically execute Node-Red during Raspberry Pi booting using the Process Manager 2 (PM2) library of NPM. 'Inject node' automatically starts all the device setup flow right after Node-RED is executed. The first step for device setup is loading the 'settings.json' file to connect to the MQTT broker. It is done by 'Read file node', 'Parse JSON node'. When MQTT client ID is not saved to the 'settings.json' file, flow for getting client ID from platform server and saving it is executed with 'HTTP Request node' and 'Write file node'. After all the settings are loaded, MQTT dynamic connection and subscription take place, which is introduced for new feature of release v2.1.0 [26]. The implemented Node-RED flow for device setup is shown in figure 13.

2) Feed Machine: The feed machine uses 2 load cells, and each of the load cells measures the weight of the bowl and the container. Exec node in Node-RED is used to execute the python file to use the measurement using loadcell. At this time, the code is executed as a spawn to continue weighing and uses a smooth node to provide the data flexibly (for example, calculating the average of 5 measurements). Then the error detector switch node is used to check for errors. If there are no errors, the message set is taken into action for MQTT message publishing. The measurement of the bowl and the container are each published as .../scale/bowl and .../scale/container form as the topic so that it is provided to the client who has subscribed to that topic. The user can use the dashboard to schedule a time for feed or the amount of feed. To do this, a servo motor is attached to the container device to open and close the door

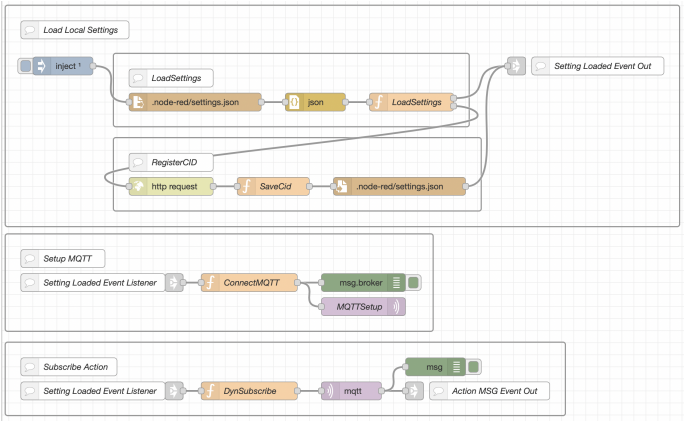

Fig. 13. Node-RED flow for device setup.

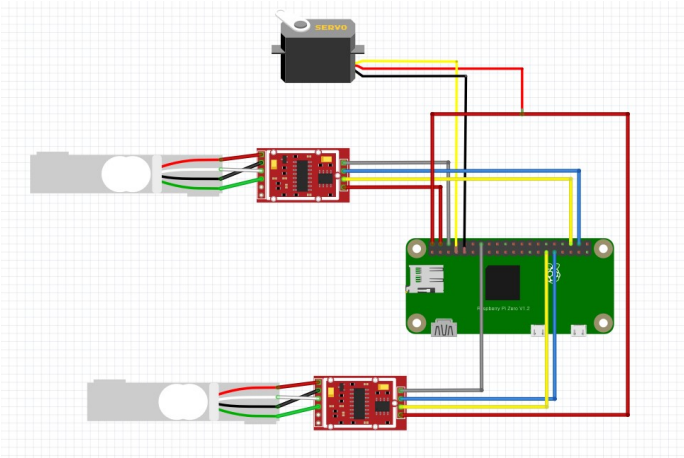

Fig. 14. Feed machine circuit diagram.

of the container to supply food to the bowl. The user's initial input or the scheduled feeding's action message is led to the device action flow through the common flow's action MSG event out node. Then the bowl's weight and the user's input of desired food weight are combined to one message. It is sent to a loop node and if the index is 0 and the bowl weight is less than the user's input, the container's servomotor opens the door to provide food. It continues to compare the two weights, the bowl weight is measured, and the payload of the message sent is continuously updated. During the loop, when the bowl

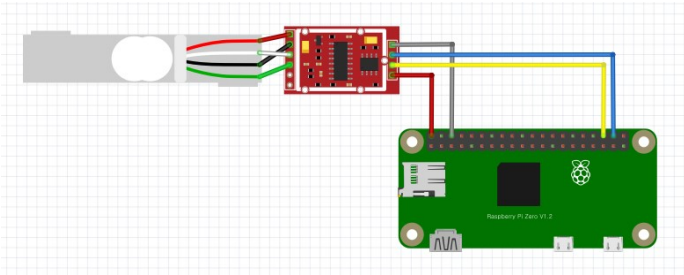

Fig. 15. water supplier circuit diagram.

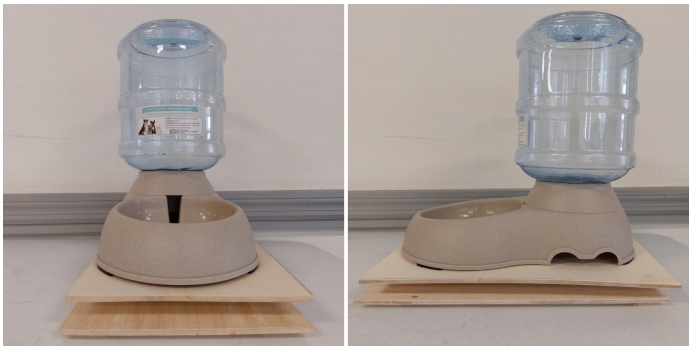

Fig. 16. Feed machine device.

weight becomes bigger than the user's input, the loop ends and the servo motor closes the door and ends feeding.

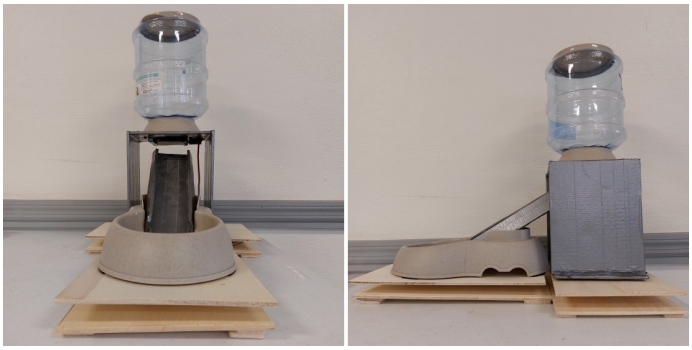

Fig. 17. Water supplier device.

3) Water Supplier: The water supplier uses one load cell to weigh the water. The process of measurement of the weight is the same as the feed machine. The measured weight is published as ../scale/water form of topic, and it is provided to the client who has subscribed to that specific topic.

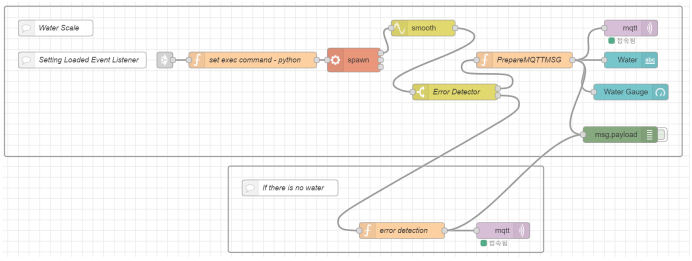

Fig. 18. Water Supplier Error Detection Flow.

4) Error Detection: Each device measures the weight of the food and water left, and if the food or water is insufficient it will send an error message to the platform server and check the user's settings on the platform server, and send a message to the user.

B. Platform

The platform for Petification uses cloud instances for deployment. On the cloud instance, Mosquitto, MySQL, and Node-RED are installed and firewall, DNS, and certificate for TLS/SSL communication are configured for networking. In the Node-RED, 7 blocks are implemented as 7 flows. The screenshot for implemented Node-RED flow is shown in Figure 19.

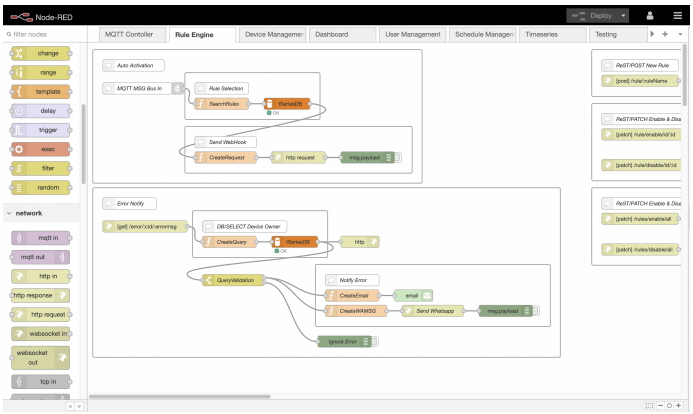

Fig. 19. Screenshot for Node-RED flow of platform.

C. Dashboard

Petification's dashboard was implemented using 'node-red-dashboard', which provides a set of nodes to make a live data dashboard [19]. The dashboard consists of three tabs: Feed Machine tab, Water Supplier tab, and User Settings tab.

1) Feed Machine tab: In the Feed Machine tab, Device Information widget, Feed Machine Statistics widget, and Serve Food widget are provided. As all the functionalities this tab provides are device-specific, selecting a device is included in the Device Information widget. Thus, all the devices which the user has and the device type is water supplier are inquired from the database, and they are displayed as a dropdown. Also, the status for the selected device is provided in the text and LED-shaped icon. The color of the LED-shaped icon changes to green when the device is connected, red for disconnected status, and yellow for error status. In the Feed Machine Statistics widget, the Remaining amount of food and container and food consumption is provided. The remaining amount of food bowl and container is displayed as gauge and food consumption is displayed as a line graph. In the Serve Food widget, the user input interface is provided to publish feed serving action by button. The serve Food widget also provides the interface for automatic food feeding (scheduled feeding). Users can see all the feeding schedules in the table format, add a new schedule by user input interface, and delete the existing schedule by clicking the row of the table. Figure 20 shows the screenshot of implemented Feed Machine tab.

2) Water Supplier tab: Similar to the Feed Machine tab, the Water Supplier tab provides two widgets: Device Information widget and Water Supplier Statistics widget. The mechanism for each widget is identical to the Feed Machine tab. Figure 21 shows the screenshot of implemented Water Supplier tab.

Fig. 20. Screenshot for Feed Machine tab.

Fig. 21. Screenshot for Water Supplier tab.

3) User Settings tab: The main purpose of the User Settings tab is to manage user information and the settings. This tab provides five widgets: User Information widget, Timezone settings widget, Notification settings widget, E-Mail address settings widget, and WhatsApp settings widget. Handling user information is the main functionality of the User Information widget. User information is defined with two values: token and username. Each user account has a unique token and username, and the token is to authorize the user when handling the HTTP requests. For username, it is used when the MQTT Broker block authenticates the user. Thus, for users to use all the features petification provides, the user must be registered to petification with a valid token and username. In the User Information widget, the user can see the token and username, and by typing the token to the user input interface, the user can change the user account. To support global users, the Timezone settings widget is included in the tab. As all the times that platform uses and database stores use Coordinated Universal Time (UTC) timezone, converting the time of the user to the UTC is necessary. Notification settings widget, E-Mail address settings widget, and WhatsApp settings widget are to control error notification. Users can turn notifications on and off with the Notification settings widget and decide which accounts receive email and WhatsApp messages. Figure 22 shows the screenshot of the implemented User Settings tab.

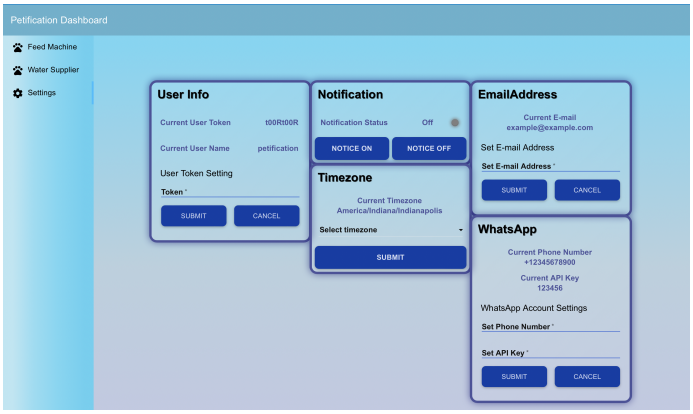

Fig. 22. Screenshot for User Settings tab.

VII. Experiment

Testing is conducted based on the results of the implementation of the Petification IoT Platform. Testing is conducted only on quantitatively verifiable results, and a total of two cases are covered: accuracy testing of load cell weight sensors that have undergone calibration and scheduled/Manual Feeding testing. The results of the testing are conducted in an integrated connection environment including a Petification IoT platform that processes logic, a device that plays the role of sensing with an actuator, and a dashboard to check the results.

A. Calibration Testing

Calibration must be performed initially to use the load cell. In this study, the weight of 500ml bottled water was fixed at 500g for calibration. Thereafter, a total of 8 tests were performed using a load cell sensor in which calibration was completed, and the weight measurement results for 500, 1000, 1500, and 2000 (g) were confirmed.

Trial	Reference Weight (g)	Measured Weight (g)	Accuracy (%)
1	500	499	99.8
2	500	499	99.8
3	1000	999	99.9
4	1000	998	99.8
5	1500	1498	99.86666667
6	1500	1498	99.86666667
7	2000	1995	99.75
8	2000	1996	99.8
Average Accuracy (%)			99.82291667

Fig. 23. Screenshot for User Settings tab.

It was confirmed that the average accuracy was 99.8%, which resulted in high accuracy. Through this, it can be confirmed that the weight measurement error by the sensor hardly occurs as the result of subsequent testing.

Trial	User Input Weight (g)	Measured Weight (g)	Accuracy (%)	Error Rate (%)
1	15	14	93.33333333	6.666666667
2	15	17	86.66666667	13.33333333
3	20	17	85	15
4	20	23	85	15
5	50	52	96	4
6	50	62	76	24
7	60	84	60	40
8	60	75	75	25
9	200	183	91.5	8.5
10	200	203	98.5	1.5
Average Accuracy (%) / Average Error Rate (%)			84.7	15.3

Fig. 24. Screenshot for User Settings tab.

B. Scheduled / Manual Feeding Testing

Testing was conducted to see if the operation and result of the scheduled feeding and direct input feeding set by the user were well-reflected. For the weight values set in User input and Scheduled Feeding, the difference error from the weight value after feeding was tested. As a result of the measurement, it was confirmed that 40% of the highest error was found, and 1.5% of the lowest error was found. The reason for the error is that the amount of feed cannot be adjusted when feeding the device, which can be seen as a limitation of the device. If the amount of feed in the device can be supplemented, good results can be expected.

As a result of the testing, no functional errors occurred. However, it is thought that it is necessary to supplement the adjustment of feed volume in the device of the Schedule / Manual Feeding Test, which is planned to be improved through a future plan.

VIII. Conclusion

This research paper is about the design and implementation of pet care IoT solutions via IoT devices. For pet caring, IoT devices for feeding and providing water are implemented. For an effective connection between the device and the user, the platform uses an MQTT Message broker to receive data from the device, a database to store data, and Node-RED to execute and visualize logic and data. The user interface was designed using Node-RED-dashboard and WhatsApp was used to send notifications to the users via SNS to provide useful information and services such as the remaining measurement, consumption, automatic/manual feeding, and notification. To test the ability and the flow of the platform, we limited the device to feeding and water supplying, however, users can use Node-RED to easily create a flow and add their own devices to the platform. Furthermore, the user can design the database as they want since they've got direct access to the database and create flows freely using Node Package Manager(NPM). This research implemented a servo motor to provide food from the container to the bowl on the feed machine. This resulted in limitations to control the amount of food being provided which led to an error in the result value. A future recommendation to this research is to redesign the device to weigh the food amount to provide just the amount that the user wants to fix the problem rather than just using the servo motor to open and close the container door to provide food. Through this research, we were able to show that the pet owners could utilize the Node-RED-based platform that could collect data and create a visualization of the flow.

The fact that this research encourages the users to easily design the platform's services, the research is not limited to the original platform's ability. We look forward to a variety of IoT platform research not just on pet care services.

IX. Acknowledge

The authors would like to appreciate the support of Chungnam National University, Software-oriented university council, Purdue University, and the Institute for Information & communication Technology Planning & evaluation(IITP). Especially, the authors of this study are grateful to Professor Minsun Lee of Chungnam National University, Professor Eric and Tony of Purdue University, and Assistant Minji Lee for helping us participate in the project.

References

[1] M. Hanson. "Pet Industry Statistics" spots.com. https://spots.com/pet-industry-statistics/ (accessed Jan. 25, 2022).
[2] Accessed: Feb. 1, 2022. [Online]. Available: https://www.instructables.com/IOT-Pet-Feeder-Using-the-Blynk-Mobile-App-an-ESP82/
[3] Accessed: Feb. 1, 2022. [Online]. Available: https://www.instructables.com/IoT-Pet-Feeder/
[4] T. Sangvanloy and K. Sookhanaphibarn, "Automatic Pet Food Dispenser by using Internet of Things (IoT)," 2020 IEEE 2nd Global Conference on Life Sciences and Technologies (LifeTech), Kyoto, Japan, Mar. 10-12, 2020.
[5] Y. Chen and M. Elshakankiri, "Implementation of an IoT based Pet Care System," 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, Apr. 20-23, 2020.
[6] Accessed: Feb. 4, 2022. [Online]. Available: https://iotdesignpro.com/projects/google-assistant-controlled-iot-pet-feeder-using-esp8266
[7] Accessed: Feb. 5, 2022. [Online]. Available: https://create.arduino.cc/projecthub/circuito-io-team/iot-pet-feeder-10a4f3
[8] Node-RED [Online]. Available: https://nodered.org/about/
[9] MQTT [Online]. Available: https://mqtt.org
[10] Theanimalfoundation. [Online]. Available https://animalfoundation.com/whats-going-on/blog/basic-necessities-proper-pet-care
[11] P. N. Vrishanka, P. Prabhakar, D. Shet and K. Rupali, "Automated Pet Feeder using IoT," 2021 IEEE International Conference on Mobile Networks and Wireless Communications (ICMNWC), Tumkur, Karnataka, India, Dec. 3-4, 2021.
[12] R. Nogueira, H. Araújo and D. Prata. (Apr. 2019). Robot Chow: Automatic Animal Feeding with Intelligent Interface to Monitor Pets. International Journal of Advanced Engineering Research and Science. [Online]. Available: https://ijaers.com/detail/robot-chow-automatic-animal-feeding-with-intelligent-interface-to-monitor-pets/
[13] Vania, K. Karyono and I. H. T. Nugroho, "Smart dog feeder design using wireless communication, MQTT and Android client," 2016 International Conference on Computer, Control, Informatics and its Applications (IC3INA), Tangerang, Indonesia, Oct. 3-5, 2016.
[14] Thepihut. https://thepihut.com/blogs/raspberry-pi-roundup/whats-the-difference-between-dc-servo-amp-stepper-motors (accessed Feb. 04, 2022).
[15] Accessed: Feb. 15, 2022. [Online]. Available: https://instrumentationtools.com/load-cell-working-principle/
[16] Accessed: Feb. 19, 2022. [Online]. Available: https://www.seeedstudio.com/blog/2019/11/26/10-things-you-can-do-with-your-hx711-and-load-cell/
[17] Accessed: Feb. 6, 2022. [Online]. Available: https://www.fujielectric.com/products/column/servo/servo_01.html

[18] N. B. Kamarozaman and A. H. Awang, ”IOT COVID-19 Portable Health Monitoring System using Raspberry Pi, Node-Red and ThingSpeak,” 2021 IEEE Symposium on Wireless Technology & Applications (ISWTA), Shah Alam, Malaysia, Aug. 17-17, 2021.
[19] Node-RED [Online]. Available https://flows.nodered.org/node/node-red-dashboard
[20] Node-RED [Online]. Available https://nodered.org/blog/2020/10/13/future-plans
[21] N. Naik, ”Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP,” 2017 IEEE International Systems Engineering Symposium (ISSE), Vienna, Austria, Oct. 11-13, 2017.
[22] Eclipse Mosquitto [Online]. Available https://mosquitto.org/
[23] S. Gruener, H. Koziolek and J. Rückert, ”Towards Resilient IoT Messaging: An Experience Report Analyzing MQTT Brokers,” 2021 IEEE 18th International Conference on Software Architecture (ICSA), Stuttgart, Germany, Mar. 22-26, 2021.
[24] Datamation. [Online]. Available: https://www.datamation.com/storage/8-major-advantages-of-using-mysql/
[25] A. Tamboli, “Build Your Own IoT Platform,” in *Apress*, 1st ed, 2019
[26] N. O’Leary. ”Version 2.1 released.” Node-RED. https://nodered.org/blog/2021/10/21/version-2-1-released#dynamic-mqtt-nodes (accessed Fab. 16, 2022).