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 profit of the pet industry more than doubles every year over 10 years, from $48.4 billion in 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.

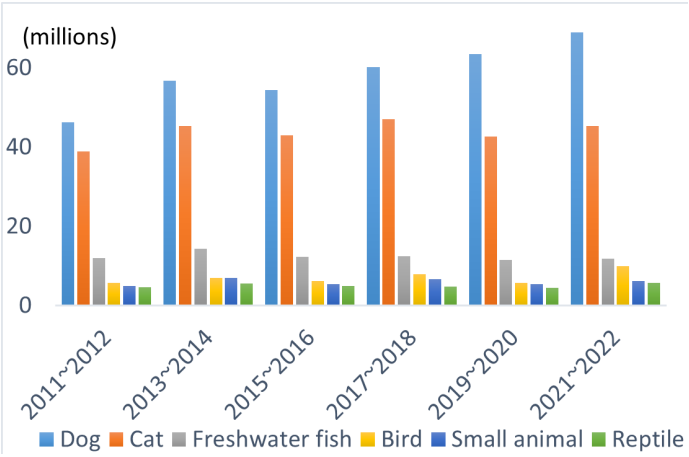

Fig. 1. Number of U.S. Households That Own a Pet, by Type of Animal

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]–[5], and 'Adafruit IO' [6] and 'Freeboard IO' [7] are used as well. Moreover, while tracking water and food consumption and automatic feeding are commonly supported by previous implementations, there are few solutions that support device status information and error notification.

For this reason, not to be locked on the limited feature that the free plan of commercialized IoT platform provides, the proposed IoT solution named 'Petification' is implemented using Node-RED which is open-source as IoT platform. And also, the Petification provides device connectivity, amount

of remaining water and food for each device, and error notification service when the water or food is running out of empty as well as tracking consumption and automatic feeding.

As a flow-based open-source visual programming tool [8], Node-RED makes possible the implementing IoT platform much faster and more user-friendly with no feature limitations. Moreover, as a lot of nodes and flows are shared in its official website and 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. MQTT Protocol is suitable for various IoT solutions because of its lightweight characteristic and publish/subscribe model [9].

In Petification, the water supplier and feed machine is attached to platform to provide water and food to the pet and scale the weight of the water and food by load cell. In the feed machine, servo motor is attached so that user can control the served amount of food. All the load cells and servo motors are mounted to Raspberry Pi for each device. 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.

II. Related Literature

This chapter will be explained with hardware and software 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 software part.

A. Hardware

P. N. Vrishanka *et al.* [11] proposed an automated pet feeder which serve food to the pet according to the remaining amount of food in the food bowl. An ultrasonic distance sensor and SG90 servo motor were mounted to Arduino Uno R3 in this research. 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.

Rogerio Nogueira *et al*, [12] proposed system that provides food and water to the pet, takes picture of the pet, and provides intelligent interface to the user with the messenger. In this research, rotary valve and DC motor were used to provide the food to the pet. Vania *et al*, [13], proposed IoT solution which provides RFID to identify the pet, serve food by schedule, and provides application to user. This research also uses propeller blade and DC motor for remote feeding.

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. Software

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.

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.

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.

Also, 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 device's 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.

III. Methodology

In this chapter, the overview and used technologies are explained. The high-level outline of petification is explained in the overview part, and used technology including sensor, actuator, program, and protocol is explained in the technology part.

A. Overview

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

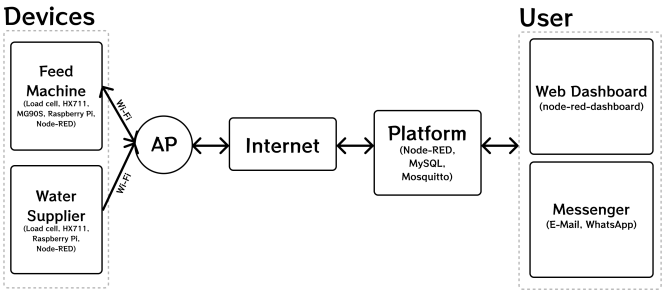

Fig. 2. Petification overview.

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 [18]. It is originally developed by International Business Machines Corporation (IBM), and IBM made it as an open-source project in 2013 [18]. 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. 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*. Thus, Node-RED v2.2.1 is installed in not only the platform server, but also Raspberry Pi of water supplier and feed machine.

3) MQTT: MQTT stands for Message Queuing Telemetry Transport. It is an OASIS standard protocol, and provides light-weight, publish/subscribe messaging transport for IoT [9]. One characteristic of MQTT is light-weight. The MQTT protocol has a small message size and message overhead, and consumes less power and resources than Advance Message Queuing Protocol (AMQP) and Hypertext Transfer Protocol (HTTP) [21]. With its light-weight characteristic, it is suitable for communication in IoT fields where network bandwidth is at a premium [27]. MQTT uses TCP as transport layer protocol and TLS/SSL can be used for security. It makes MQTT more reliable than Constrained Application Protocol (CoAP) which uses UDP as the transport protocol [21]. All MQTT messages have their own topics, and MQTT clients can send data for certain topic by publishing messages to MQTT broker or receive data by subscribing to topics for those messages. In this research, Eclipse Mosquitto v1.4.15 is used as a MQTT Message Broker. Eclipse Mosquitto is an open-source message broker implementing MQTT protocol versions 5.0, 3.1.1, and 3.1 [22].

4) 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]. The used version of MySQL is v5.7.37.

IV. Hardware Design

A. Water Supplier

The water supplier is also a device attached to the proposed platform. The main functionalities for the water supplier are supplying the water to the pet and publishing the weight of the remaining water. However, servo motor is not used for the water supplier as controlling the serving amount of the supplied water isn't necessary. 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.

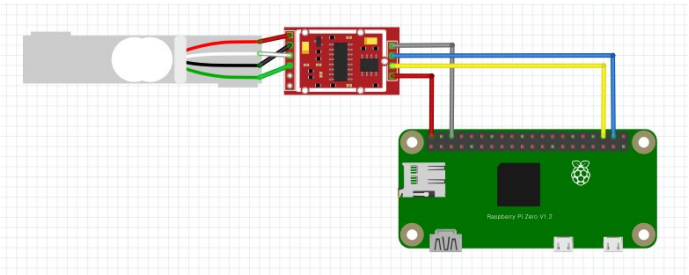

Fig. 3. Circuit diagram of the water supplier.

B. 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 publishing 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.

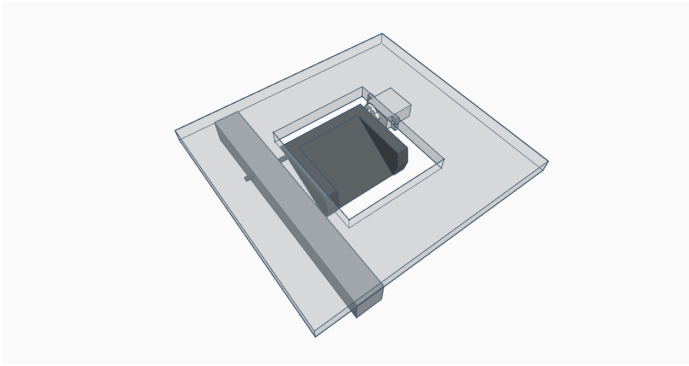

Fig. 4. Food gate of the feed machine

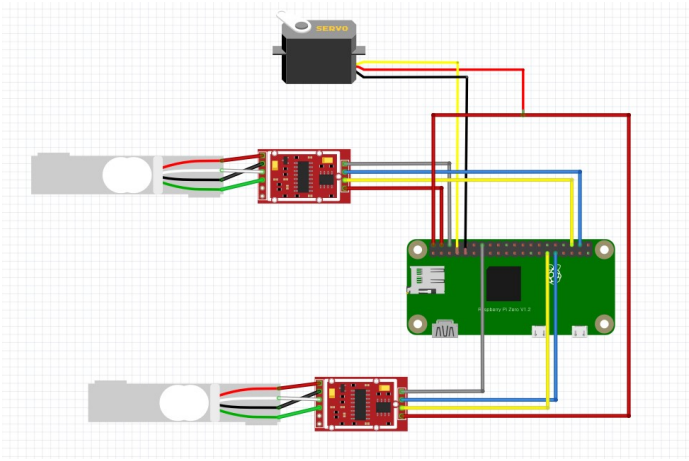

Fig. 5. Circuit diagram of the feed machine

V. Software Design

A. Software Design for Devices

1) 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.

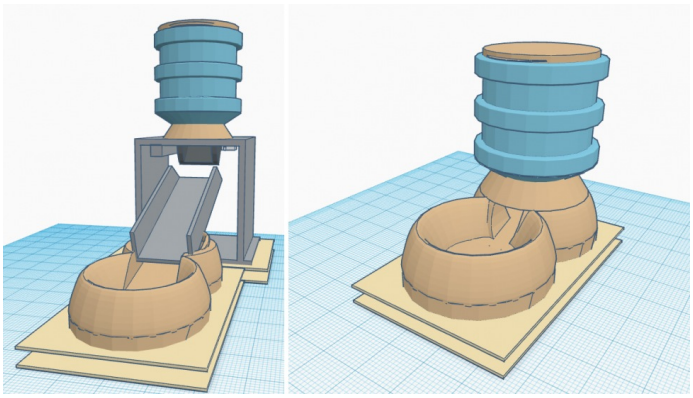

Fig. 6. CAD design for the feed machine and water supplier

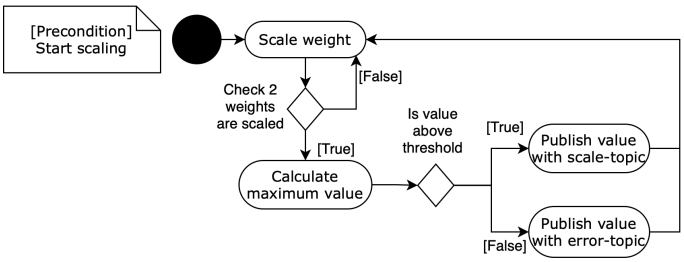

Fig. 7. Activity Diagram for Publishing Scale and Error Detection

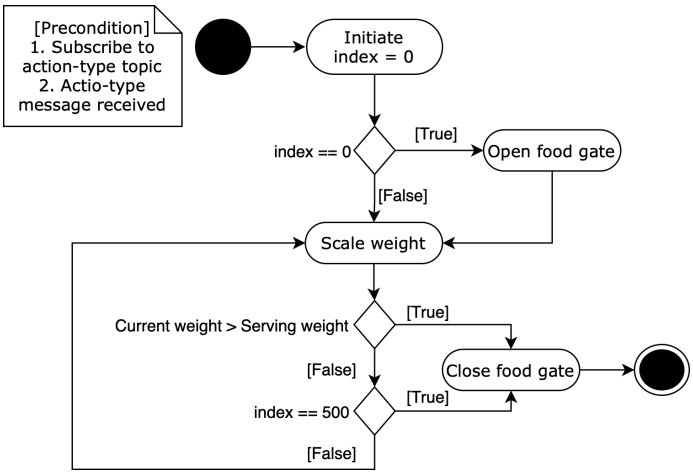

Fig. 8. Activity Diagram for Automatic Feeding.

2) 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.

B. Software Design for Platform

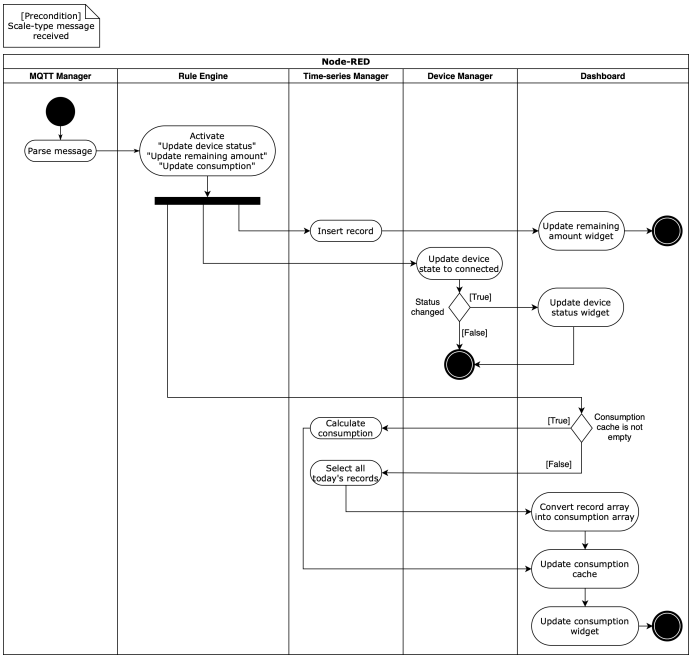

Fig. 9. Activity Diagram for Receiving scale message.

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 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.

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 $prevConsumption + decrease$

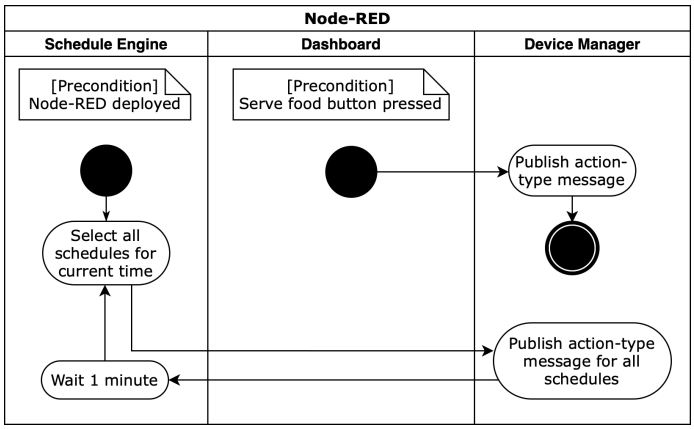

Fig. 10. Activity Diagram for Publishing Action Message.

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 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.

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

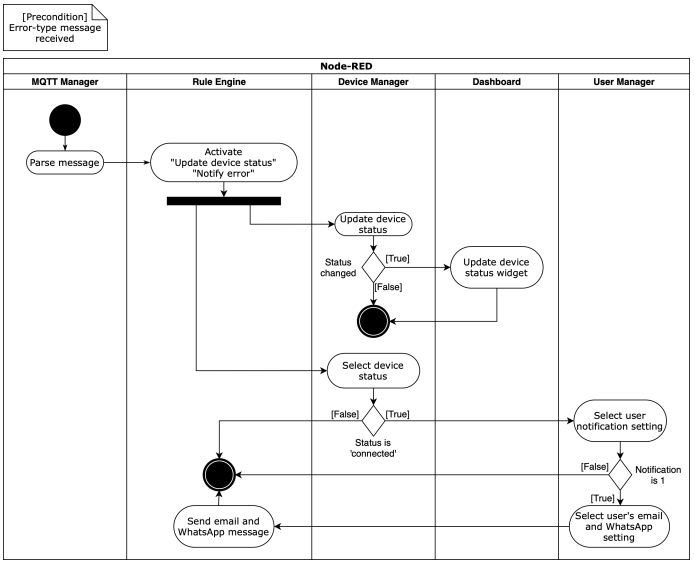

Fig. 11. Activity Diagram for Receiving Error Message.

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

The result of the implementation for the water supplier is shown in figure 16, and that for feed machine is shown in figure 17.

1) Calibration and weight measurement: To convert the measurement of the load cell to the weight data, the calibration process has proceeded to get the reference unit. The process of calibration takes two steps. First, measure repeatedly the force of the standard object whose weight is known applies to the load cell. In the calibration process of the petification, a 500mL bottle of water is used as a standard object for 500g weight. Second, get a reference unit by calculating the average of measurement of standard object and divide it by actual weight, which is 500g in this case.

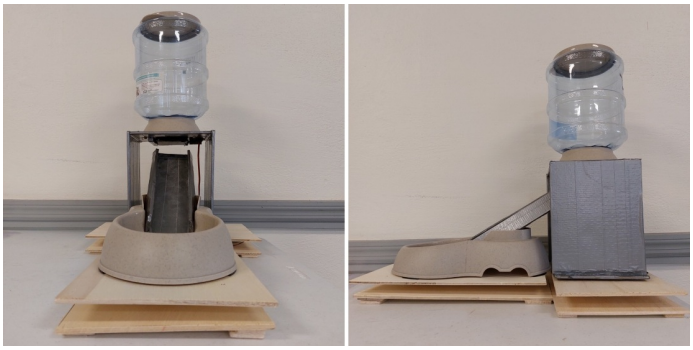

Fig. 12. Result of implementation for water supplier

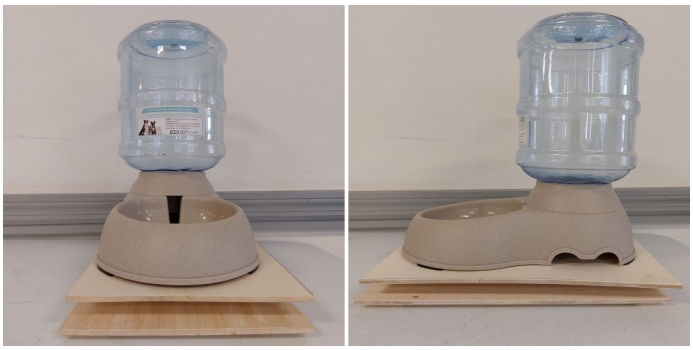

Fig. 13. Result of implementation for feed machine

Measuring the weight of the water and food is done with python code. By using 'exec' node of the Node-RED in Raspberry Pi to execute python code, result of measurement can be used in Node-RED. However, even if calibration has been proceeded, there is always the possibility of measurement error. Thus the 'smooth' node of the 'node-red-node-smooth' library has been used to get a maximum value of two measurement result. By calibrating the measurement and calculating the maximum value, each Raspberry Pi in water supplier and feed machine can get the correct weight.

2) Publishing weight or error message: Before publishing the weight of the water and food, error detecting process has been progressed with using 'switch' node of the Node-RED. The 'switch' node checks if the weight is below the threshold value which is 10g for this implementation. When it's true, each device publishes error message instead of scale message. Error message and scale message is distinguished by topic and payload of the message is either error message or weight value. The implemented Node-RED flow for publishing weight or error message is shown in figure 18.

3) Automatic feeding: The action message is structured to contain 'action' in the message topic and desired serving amount in the message payload. When an action message is arrived, the bowl's weight and message payload which is desired food weight are combined to one message. It is sent to a 'loop' node that is supported by 'node-red-contrib-loop-processing' library and if the index is 0 and the bowl weight is less than the user's input, the container's servo motor opens the

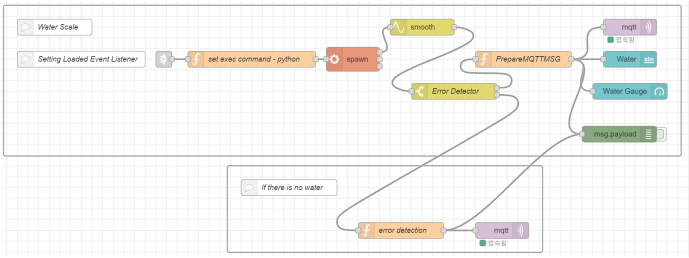

Fig. 14. Node-RED screenshot for publishing weight or error message.

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 weight becomes bigger than the user's input, the loop ends and the servo motor closes the door and ends feeding. The implemented Node-RED flow for automatic feeding is shown in figure 19.

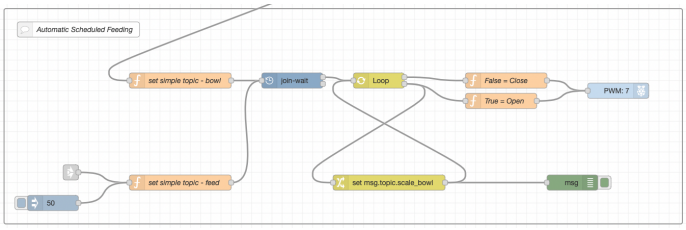

Fig. 15. Node-RED screenshot for automatic feeding.

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.

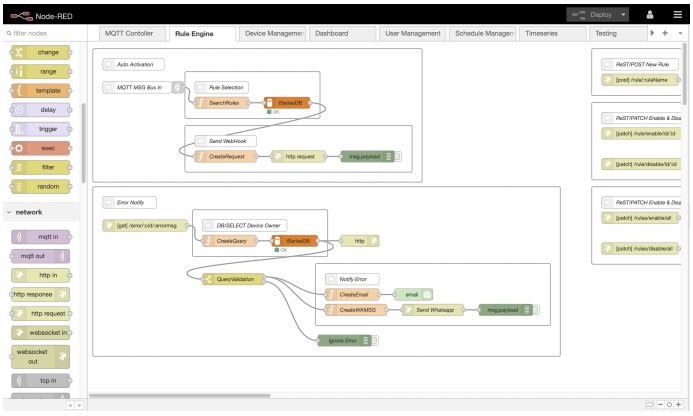

Fig. 16. Screenshot for Node-RED flow of platform.

1) MQTT Controller:
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.

2) 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.

3) 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.

4) Dashboard:
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.

5) User Management:
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.

6) 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.

7) Timeseries 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).

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 and is 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.

Fig. 17. Screenshot for 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. 18. Screenshot for Water Supplier tab.

3) User Settings tab: The User Settings tab provides five widgets: User Information widget, Timezone settings widget, Notification settings widget, E-Mail address settings widget, and WhatsApp settings widget. In the User Information widget, users can sign in to the system by entering the token, which is an identifier for the user. After sign in, user can see the token and username for the current user. 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.

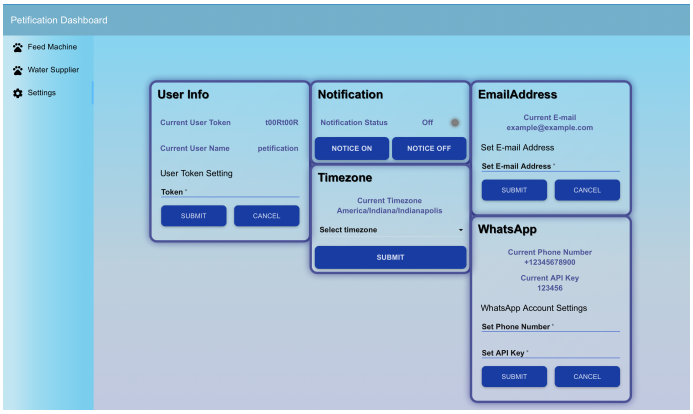

Fig. 19. Screenshot for User Settings tab.

VII. Experiment and Testing

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 the weight measurement and that of the automatic feeding.

A. Weight measurement testing

In this test, a total of 8 tests were performed using a load cell sensor in which calibration was completed, and actual weight is compared with the measured weight in each test.

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. 20. 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.

B. Automatic feeding testing

In this test, a total of 10 tests were performed to figure out the automatic feeding feature works as expected. The desired serving amount is compared with the actual serving amount in each test.

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. 21. Screenshot for User Settings tab.

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 food is often stuck in front of the food gate and the food serving speed is too fast for the load cell to detect. It is clearly a limitation for the feed machine and can be solved if the design is modified correctly.

VIII. Conclusion

In the present research, pet-care IoT solution named 'Petification' is proposed to take care of user's pet when they are not at home. The water supplier is supported in the petification to supply water to the pet and report the current amount of the remaining water. The feed machine is also supported to serve certain amount of food to the pet and report the current amount of the remaining food. The IoT platform for the petification is implemented using Node-RED to take advantage of open-source and flow-based visual programming. Also, the message flow for the petification is controlled with Eclipse Mosquitto, which is open-source implementation of the MQTT Protocol. Users for this solution can check the water and food consumption of the pet and serve the food to the pet on the web based dashboard. The dashboard of the petification also provides device connectivity and amount of remaining food and water to the user. When the water or food for each device is empty, the user can get notifications for it.

While all other functionalities are working correctly, the accuracy of serving exact amount food to the pet is relatively low. The low accuracy is caused by design of the food gate and sensitivity of the load cell. Thus, the future plan can be enhancing the design of the food gate and the sensitivity and stability of the load cell to serve the exact amount of food. Also, attaching another type of device such as device for taking picture of the pet can be the possible improvements for petification.

IX. Acknowledgement

"This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the National Program for Excellence in SW) supervised by the IITP(Institute of Information & communications Technology Planing & Evaluation) in 2021"(2021-0-01435) The authors of this study are grateful to Professor Minsun Lee of Chungnam National University, Professor Eric T. Matson and Professor Anthony H. Smith of Purdue University, and Teaching 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).
[27] OASIS Open [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt