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. However, most of the previous studies which are conducted to serve food and water remotely use the limited featured IoT platforms and do not provide the remaining amount of the food and water and empty notification service to the user. The proposed IoT solution named "Petification" uses the open-source project Node-RED with MQTT messaging protocol to provide information about device connectivity and the remaining amount of food and water as well as consumption via a web-based dashboard. Additionally, Petification provides an empty notification service with a remote feeding service. The water supplier and feed machine are attached to the platform to provide water and food to the pet and scale the weight of the water and food. The load cell, HX711 amplifier, and Raspberry Pi Zero W are mounted to the water supplier and feed machine and an MG90S servo motor is mounted to the Raspberry Pi of the feed machine. However, served amount of the food sometimes mismatches with the desired amount while testing the implementation. Thus the future plan can be to enhance the food gate to serve the exact amount of food.**

***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 is more than doubled every year over 10 years, from $48.4 billion in 2010 to $109.6 billion in 2020 [1]. 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. Because for those who left home often, it is challenging to take good care of their pets.

As a way to solve this problem, Internet of Things (IoT) technology has emerged. A lot of pet care IoT solutions are introduced in the market, and many studies and implementations are suggested. However, the IoT platforms which are used to implement pet care IoT solution has limitations. For example, a most commonly used IoT platform is "Blynk" [2]–[4], but the latest version of it is not open-source and the open-source version is no longer maintained. Also, "Adafruit IO" [5] and "Freeboard IO" [6] are other commonly used IoT platforms, but they are not open-source. Moreover, while tracking water and food consumption and automatic feeding are commonly supported by previous implementations, few solutions support device status information and error notification.

The proposed IoT solution named "Petification" uses open-source projects for the IoT platform. Node-RED which is a flow-based open-source visual programming tool [7] is used to implement the APIs of the IoT platform. By using Node-RED, Petification can get the benefit of providing user-friendly UI, fast development, and a lot of resources shared by users. Also, Eclipse Mosquitto which is one of the open-source implementations of the MQTT messaging protocol is used to manage overall message flow. Using MQTT protocol as a message broker allows the IoT platform to connect the user with devices in a lightweight way [8].

Petification provides information about device connectivity and the amount of remaining food/water for each device as well as food/water consumption. Additionally, Petification provides an error notification service when the water or food is running out of empty among with automatic feeding service. The water supplier and feed machine are attached to the platform to feed and water the pet and to scale the weight of the food/water. While both the water supplier and feed machine use the load cell to scale the weight, a servo motor is attached only to the feed machine so that the user can control the served amount of food. All the load cells and a servo motor are mounted to Raspberry Pi for each device. The web-based dashboard is supported to provide this information and services visually. The food/water empty notification is also provided to the user via email and WhatsApp mobile

application.

II. Related Literature

T. Sangvanloy *et al.* [3] 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. The conducted research also provides an automatic serving amount adjustment feature based on the pet's species and weight. The IoT solution was implemented with ESP32 Wi-Fi micro-controller, servo motor, load cell, and "Blynk" IoT platform.

Y. Chen *et al.* [4] proposed a pet care IoT solution that provides information such as food and water consumption, the number of defecation, and defecation duration. It was implemented using Arduino Uno and ESP8266 Wi-Fi Module with load cell, servo motor, and motion sensor. A mobile application is provided as a UI with the help of the "Blynk" IoT platform.

P. N. Vrishanka *et al.* [9] proposed an automated pet feeder which serves 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.

Rogerio Nogueira *et al.* [10] proposed a system that can serve food and water according to pre-scheduled times and amounts. Additionally, the system provides a notification service for abnormal situations with photos and alerts and a "chat-based" intelligent interface to the user. This research is conducted with Raspberry Pi, Python, Telegram cloud service, and IBM Watson Assistant.

Vania *et al.* [11] proposed an IoT solution that identifies each pet via RFID and serves food by schedule. The dog feeder in the research was developed using Arduino Uno and ESP8266 Wi-Fi Module with load cell and DC motor. An Android mobile application is provided to the user and the IoT platform which is developed with Node JS and MySQL database controls device-user communication using the MQTT protocol.

III. Methodology

The system architecture for Petification is designed to connect the feed machine and water supplier with the users. Each user and device can communicate bi-directionally through the platform. The feed machine and water supplier are 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. The overview for Petification is shown in figure 1.

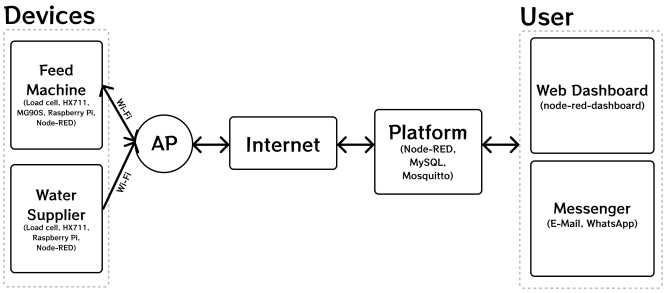

Fig. 1. Petification overview

A. Water Supplier

Measuring the weight of the water is done with one load cell which scales up to a maximum of 5kg with 1g accuracy. A load cell is installed between the two plates to measure the pressure applied to an upper plate as shown in figure 2 and is attached to the Raspberry Pi Zero W through the HX711 amplifier which converts the analog signal to the digital signal as shown in figure 3. 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 is shown below:

$$referenceUnit = \frac{average(loadCellMeasurements)}{actualWeight}$$

A 500mL water bottle which is used as an actual weight of 500g is measured 500 times to get a reference unit.

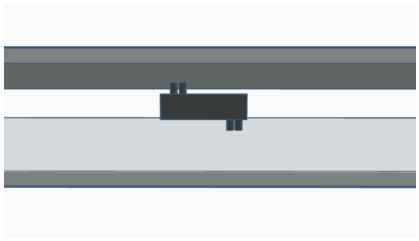

Fig. 2. Installaion of a load cell

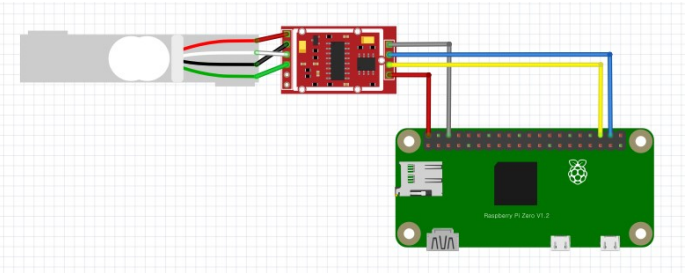

Fig. 3. Circuit diagram of the water supplier

B. Feed Machine

To serve food to the pet, an MG90S servo motor is mounted to open and close the food gate. As in figure 4, 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. Two load cells and two HX711 amplifiers are installed in the same way as the water supplier under the food bowl and container to measure the weight of the food. All the load cells, HX711 amplifiers, and an MG90S servo motor are attached to the Raspberry Pi Zero W as shown in figure 5.

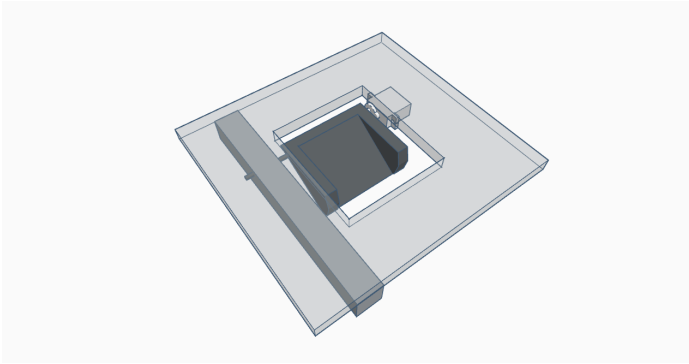

Fig. 4. Food gate of the feed machine

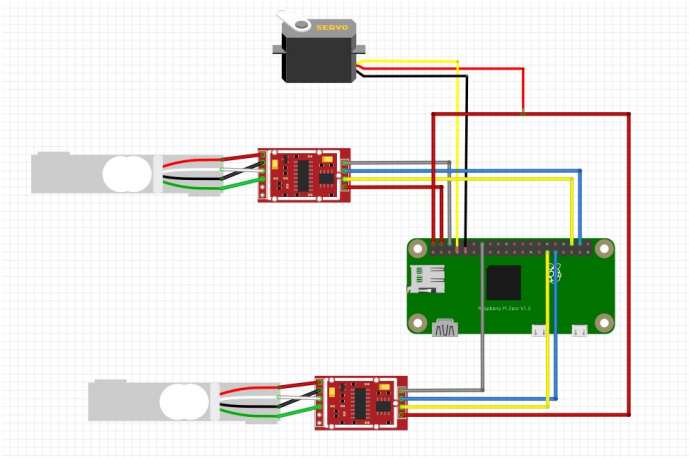

Fig. 5. Circuit diagram of the feed machine

C. IoT Platform

The user-to-device communication occurred when the user wants to serve the food to the pet, and device-to-user communication occurred to provide information and notification to the user. To serve the food, the user can either set the serving schedule or press the button in the dashboard. The serving schedules are stored in the database and checked every minute to send the message to the feed machine when the scheduled time arrives. The food serving message is also sent when the user presses the button. After the feed machine receives the message, it opens the food gate and scales the food bowl repeatedly to serve the desired amount of food to the pet. The detailed flow for the food serving is shown in figure 6.

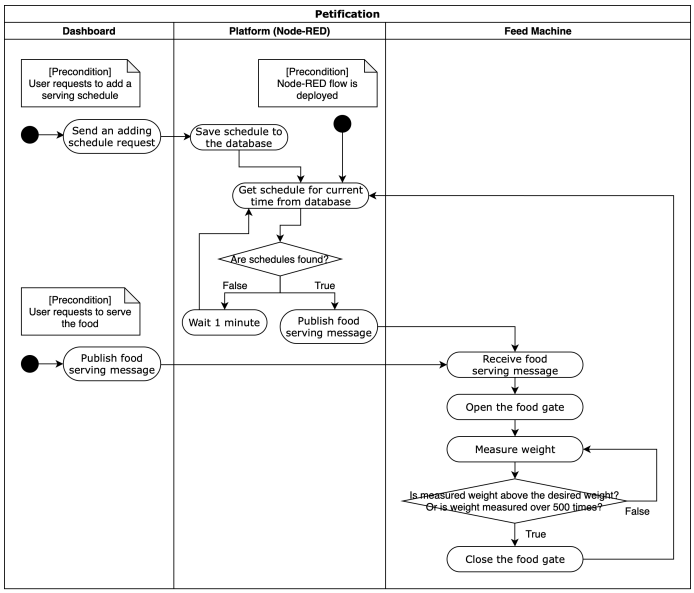

Fig. 6. Activity diagram for user-to-device communication

To provide information and notification to the user, both the water supplier and feed machine scale what they stores. When the measured weight is above a certain threshold, they send a message containing the measured weight to the platform. This data message is used to update and show the remaining amount, consumption, and device status to the user. However, when the measured weight is below the threshold, then an error message is sent to the platform to update the device status and to notify the user that the food/water is empty. The detailed device-to-user flow is shown in figure 7 and the pseudo-code for calculating the consumption is shown in algorithm 1.

Algorithm 1 Calculate consumption

```
procedure CalcConsumption
    prevConsumption ← previous consumption
    currScale ← scale of received message
    prevScale ← scale of previous record
    decrease ← 0
    if precScale ≠ null & prevScale > currScale then
        decrease ← prevScale − currScale
    return prevConsumption + decrease
```

Three open-source projects are used in the platform: Node-RED, Eclipse Mosquito, and MySQL.

1) Node-RED: Node-RED is a flow-based open-source visual development tool that is easy for developers and non-developers to develop programs [12]. One of the strengths of the Node-RED is its powerful community. Not only Node-RED itself but also various nodes such as dashboard widgets and database drivers are being distributed through Node

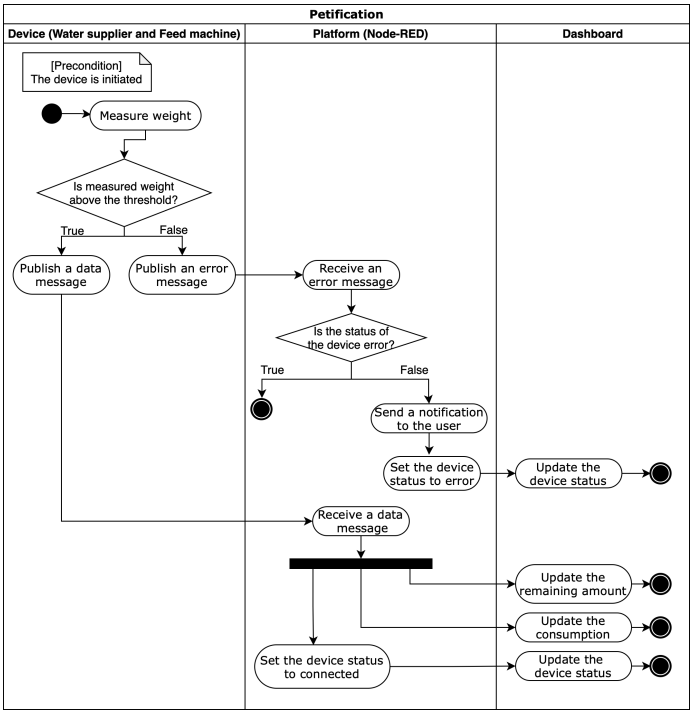

Fig. 7. Activity diagram for device-to-user communication

Package Manager (NPM). Another strength is that it can run on various environments such as local, Raspberry Pi, Docker, and Cloud Instance. Thus, Node-RED v2.2.1 is installed in not only the platform server but also Raspberry Pi of water supplier and feed machine.

2) Eclipse Mosquitto MQTT message broker: MQTT (Message Queuing Telemetry Transport) is an OASIS standard protocol and provides lightweight, publish/subscribe messaging transport for IoT [8]. The publish/subscribe model of the MQTT protocol allows the IoT platform to support bi-directional communication in a way that one can subscribe to the ”topic” of the message to receive a message published to that ”topic” by another. In this research, Eclipse Mosquitto v1.4.15 is used as an MQTT Message Broker. Eclipse Mosquitto is an open-source message broker implementing MQTT protocol versions 5.0, 3.1.1, and 3.1 [13].

3) MySQL: MySQL is a fast, flexible, and easy-to-use open-source database with RDBMS (relational database management system). In addition to the performance and security perspectives, MySQL is considered a suitable database to manage effective data flows because it has good compatibility with Node-RED. For this reason, MySQL v5.7.37 is integrated as a component of the platform.

IV. Implementation

A. Water supplier

Calibrating and measuring the weight of the water is done with python code. By using the ”exec” node of the Node-RED in Raspberry Pi to execute python code, the result of a measurement can be used in Node-RED. However, even if the calibration has 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 results. After the weight is measured, the ”switch” node determines what type of message (weight data or error) has to be sent by checking if the weight is below the threshold value which is 10g for this implementation. The messages are sent after the ”function” node prepares the message and the ”mqtt out” node publishes the message. The Node-RED flow for publishing weight or error messages is shown in figure 8 and the implemented water supplier is shown in figure 9.

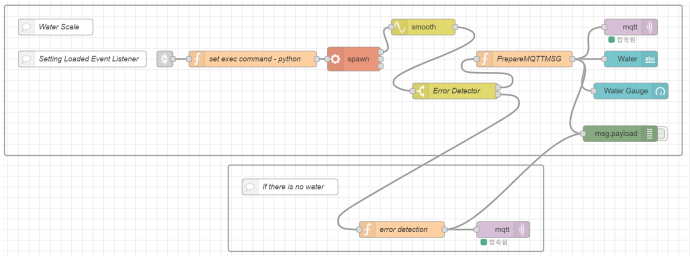

Fig. 8. Node-RED screenshot for publishing weight or error message

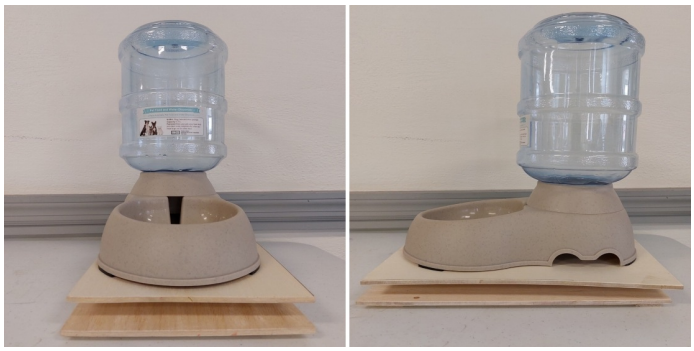

Fig. 9. Result of implementation for water supplier

B. Feed machine

In addition to the weight measuring flow which is the same as that of the water supplier, the flow for serving a certain amount of the food to the pet is implemented in the feed machine. Food serving flow does not begin before a message which contains desired serving amount at the message payload arrives. The ”join-wait” node of the ”node-red-contrib-join-wait” node module waits for a food serving message and returns an object which contains a current weight for that time and desired weight after the serving message has arrived. Then, the current weight is measured and compared repeatedly by a ”loop” node of the ”node-red-contrib-loop-processing” and a ”change” node. The food gate is opened for the first loop and is closed when the current weight is above the desired weight or the loop is repeated 500 times by the ”pi-gpiod out” node of the ”node-red-node-pi-gpiod” node module. The Node-RED flow for serving food is shown in figure 10 and the implemented feed machine is shown in figure 11.

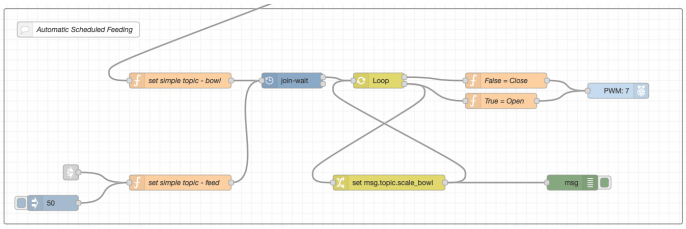

Fig. 10. Node-RED screenshot for automatic feeding

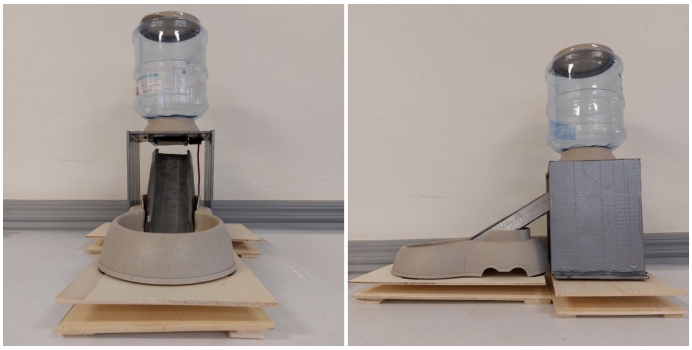

Fig. 11. Result of implementation for feed machine

C. IoT Platform

The platform for Petification uses cloud instances for deployment. On the cloud instance, Mosquitto, MySQL, and Node-RED are installed and the firewall, DNS, and certificate for TLS/SSL communication are configured for networking. In the Node-RED, 7 IoT platform components are implemented as 7 flows. A screenshot for implemented Node-RED flows is shown in Figure 12.

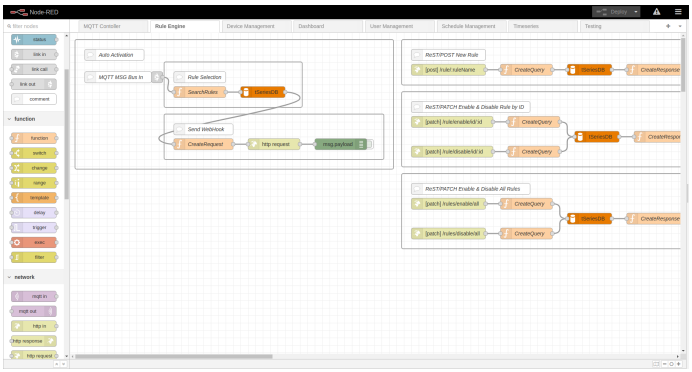

Fig. 12. Node-RED flow screenshot of IoT platform

1) MQTT Controller: MQTT Controller flow in the Node-RED is the gateway for MQTT messages to enter Node-RED. The main purpose of this flow is to parse the topic and payload of the incoming MQTT message to provide useful information such as the MQTT username and client id to other flows. The ”mqtt in” node is used to subscribe and receive the MQTT message and the ”function” node is used to parse the message and convert it to an object.

2) Rule Engine: The purpose of Rule Engine is to activate actions according to MQTT messages. Designing logic of the Rule Engine is inspired by the book, “Build Your Own IoT Platform” [**?**]. 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.

3) Device Manager: Handling devices that are attached to the platform is the main purpose of the Device Manager flow. It provides ReST APIs that can add a new device to the platform, modify the status of the device, and delete a device. And it also provides functionality for publishing an action-type message to a specific device.

4) Dashboard: Dashboard Manager is to provide a Graphical User Interface (GUI) to the user. By using this flow, 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. A screenshot of the implemented dashboard is shown in figure 13.

Fig. 13. Screenshot for Feed Machine tab of the dashboard

5) User Management: The purpose of the User Manager flow is to provide a ReST API for modifying user settings. Users can manage these 4 settings: Notification enable/disable, Email address, WhatsApp account information, and timezone where the user lives. Also, a ReST API for sending email and WhatsApp notifications is included in this flow using the ”email” node of the ”node-red-node-email” node module and the ”whatsapp-bot” node of the ”node-red-contrib-whatsappbot” node module.

6) Schedule Engine: Handling and executing schedules is the main functionality of this flow. Every minute, the schedule engine checks the schedule table of the database and executes the actions that are scheduled to be activated at that time. Similarly, with the rule engine, each corresponded action is defined as a ReST API form, thus an HTTP request is sent when the schedule is executed.

7) Time-series Manager: Managing the time-series data table of the database is the main purpose of the Time-series Manager flow. As all the published MQTT messages are stored

in a time-series data table, other flows such as dashboard flow can get and utilize the time-series data.

V. Experiment and Testing

Testing is conducted based on the results of the implementation. Testing is conducted only on quantitatively verifiable results, which is accuracy testing of the food serving feature. A total of 10 tests were performed and 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. 14. Testing result of automatic feeding

As a result of the test shown in figure 14, 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 a limitation for the feed machine and can be solved if the design is modified correctly.

VI. Conclusion

In the present research, a pet-care IoT solution named "Petification" is proposed to take care of users' pets when they are not at home. The IoT platform for Petification is implemented using Node-RED and Eclipse Mosquitto MQTT message broker to take advantage of open-source. 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 a certain amount of food to the pet and report the current amount of the remaining food. The web-based dashboard is supported to allow the user to keep track of the remaining amount, consumption, and connectivity for each device. Users can also get the notification for the emptiness of each device via email and WhatsApp messenger. While all other functionalities are working correctly, the accuracy of serving the exact amount of food to the pet is relatively low. The future plan can be to enhance 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 a device for taking pictures of the pet can be a possible improvement for Petification.

VII. 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, Professor Anthony H. Smith, and Teaching Assistant Minji Lee of Purdue University 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] 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.
[4] 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.
[5] Accessed: Feb. 4, 2022. [Online]. Available: https://iotdesignpro.com/projects/google-assistant-controlled-iot-pet-feeder-using-esp8266
[6] Accessed: Feb. 5, 2022. [Online]. Available: https://create.arduino.cc/projecthub/circuito-io-team/iot-pet-feeder-10a4f3
[7] Node-RED [Online]. Available: https://nodered.org/about/
[8] MQTT [Online]. Available: https://mqtt.org
[9] 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.
[10] 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/
[11] 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.
[12] 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.
[13] Eclipse Mosquitto [Online]. Available: https://mosquitto.org/
[14] A. Tamboli, "Build Your Own IoT Platform," in *Apress*, 1st ed, 2019