Smart Water Bowl for Pets: An IoT-Based Solution for Monitoring Pet Hydration

Om Mishra
Department of Engineering
University of Bologna
Bologna, Italy
om.mishra@studio.unibo.it

***Abstract*—This report presents the design and implementation of a smart water bowl for pets, leveraging the Internet of Things (IoT) to monitor water intake, prevent dehydration, and enhance pet well-being. The system integrates an ESP32 microcontroller with temperature/ humidity sensors and a water level sensor to collect real-time data. The data is transmitted via HTTP/CoAP to a Python-based data proxy, which then stores it in an InfluxDB time-series database. A prediction module forecasts water levels, and Grafana visualises both real-time and predicted data. The system's performance is evaluated based on mean latency and forecast accuracy on Mean Square Error(MSE).**

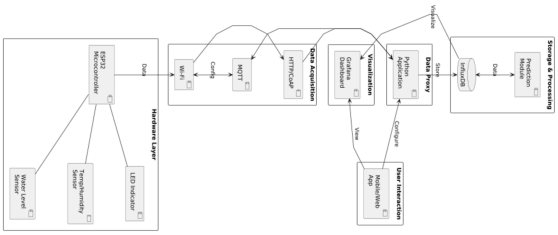
Keywords—IoT, ESP32, pet hydration, water level monitoring, MQTT, InfluxDB, Grafana, forecasting.

I. Introduction

Maintaining proper hydration is crucial for the health and well-being of pets. Dehydration can lead to various health issues, including kidney problems, lethargy, and even heatstroke. Pet owners often struggle to ensure their pets have consistent access to fresh water, especially when they are away from home. This project addresses this challenge by developing a smart water bowl that actively monitors water levels, alerts owners when refills are needed, and provides insights into pet hydration patterns.

Existing solutions often rely on manual monitoring or simple float mechanisms, which do not provide real-time data or predictive insights. Our IoT-based approach offers a comprehensive solution that not only tracks current water levels but also predicts future needs, enabling proactive pet care.

II. Project's Architecture



A. Hardware

An ESP32 microcontroller serves as the central processing unit, equipped with temperature/humidity sensors and a water level sensor (ultrasonic or capacitive).

B. Data Acquisition

The ESP32 collects sensor data and alarm events, transmitting them to a data proxy via HTTP/MQTT. It also gathers Wi-Fi RSSI values for network performance analysis. The MQTT protocol enables remote configuration of sampling rates, calibration values, alarm thresholds, and alarm counters.

C. Data Proxy

A Python application running externally receives sensor data from the ESP32 and forwards it to an InfluxDB instance. It also facilitates MQTT-based configuration updates and is also responsible for calculating Mean Latency of the data acquisition process.

D. InfluxDB

This time-series database stores water level readings, alarm events, Wi-Fi RSSI data, forecast metrics and evaluation metrics.

E. Prediction Module

A Python script analyses historical data to forecast future water levels.

F. Grafana

A customisable dashboard visualises real-time sensor data, alarm events, RSSI values, and water level forecasts.

III. Project's Implementation

The implementation of our smart water bowl system involved several interconnected components, each requiring careful development and integration. Here, we detail the process for each major component:

A. Hardware Setup

The core of our hardware setup is the ESP32 microcontroller, chosen for its built-in Wi-Fi capabilities and low power consumption. We connected the following sensors to the ESP32:

1. Water Level Sensor: We used an ultrasonic sensor (HC-SR04) to measure the water level. The sensor was mounted at the top of the bowl, with its distance from the bottom of the empty bowl used as a calibration value (10cm)
2. Temperature and Humidity Sensor: A DHT22 sensor was integrated to monitor environmental conditions, which can affect pets' water intake.
3. LED Indicator: An LED was connected to GPIO pin 19 to provide a visual alarm for low water levels.

The hardware was simulated on an online platform WOKWI for prototyping, with plans to design a custom PCB for the final product.

B. Software Development

- **ESP32 Firmware:** WOKWI was used to develop the firmware, incorporating libraries for sensor communication, HTTP/CoAP requests, MQTT, and Wi-Fi management.
- **Data Proxy:** The Python script was written using libraries like paho-mqtt for MQTT communication and InfluxDB-Python for database interaction.

  The proxy serves multiple functions:

  - Receives HTTP/MQTT requests from the ESP32
  - Forwards data to InfluxDB
  - Manages MQTT communication configuration
  - Calculates and logs data transmission latency

- **Prediction Module:** The water level forecasting module uses scikit-learn's LinearRegression model. It retrieves historical data from InfluxDB, trains the model, and generates predictions.
- **Prediction Module:** It defines functions to retrieve actual water level and forecasted water level data from influxDB for the last 7 days. It calculates only the Mean Squared Error (MSE) between the actual and forecasted values and writes to a separate InfluxDB bucket for storing metrics.

C. Integration

The integration process involved several steps:

1. ESP32 Configuration: The ESP32 was programmed with the appropriate Wi-Fi credentials and server addresses.
2. Data Flow Testing: We verified the end-to-end data flow from the ESP32 to InfluxDB via the data proxy.
3. MQTT Configuration: MQTT topics were set up for configuration updates, and we tested remote changes to sampling rates, alarm thresholds, etc.
4. InfluxDB Setup: We created the necessary buckets in InfluxDB for storing sensor data, alarms, latency metrics, and forecast results.
5. Grafana Dashboard: A comprehensive dashboard was created in Grafana, pulling data from InfluxDB and displaying real-time and historical information.

D. Testing and Calibration

Rigorous testing was conducted to ensure system reliability:

1. Sensor Accuracy: We compared water level readings against manual measurements to verify sensor accuracy.
2. Network Resilience: The system was tested under a real environment to ensure reliable data transmission.
3. Alarm System: We simulated low water levels to verify timely alarm triggering and notification.
4. Forecast Accuracy: The prediction model was evaluated using historical data, comparing forecasts against actual readings.

This implementation process resulted in a fully functional smart water bowl system, capable of real-time monitoring, remote configuration, and predictive analytics for pet hydration management.

IV. RESULTS

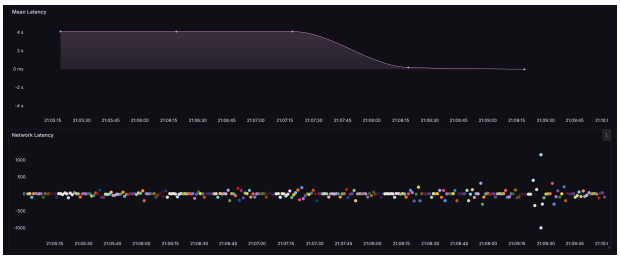
The system's performance was evaluated based on two key metrics: mean latency of the data acquisition process and forecast accuracy.

A. Mean Latency

To measure the network latency in sending data to the proxy, we implemented a timestamp system in our ESP32 firmware and data proxy. The ESP32 includes a 'send_time' in each data packet, which is compared to the receive time at the proxy. This allows us to calculate the transmission latency for each data point.

The latency data is stored in InfluxDB alongside the sensor readings, allowing for long-term analysis of network performance. Our Grafana dashboard includes a panel displaying real-time latency values and a moving average.

While specific latency values can vary based on network conditions, since our tests were performed in an simulated environment so we were not able to observe real world latency issues and it showed an average latency of 0 ms over the complete testing period.

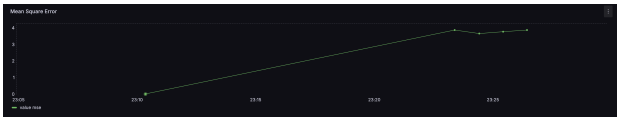


B. Forecast Accuracy

The accuracy of our water level forecasts was assessed using multiple metrics. We used a Linear Regression model for forecasting, trained on the past week's data to predict water levels for the next 24 hours. The evaluation metrics were calculated using a Python script that compares the forecast values against actual readings over a 7-day period. The results are as follows:

Mean Squared Error (MSE): This represents the average squared difference between predicted and actual water levels.

These metrics are continuously calculated and stored in the 'forecast_metrics' bucket in InfluxDB, allowing us to monitor the model's performance over time.



C. System Performance

Our ESP32-based system successfully maintained a stable connection to the local Wi-Fi network, with an average RSSI of [-70 dBm]. This ensured reliable data transmission throughout the testing period.

The alarm system, which triggers when the water level remains below the set threshold for a specified number of consecutive readings, performed as expected. During our testing, it correctly identified low water levels and sent alerts via the MQTT protocol.

D. Visualisation

Fig. 2 shows a screenshot of our Grafana dashboard, which provides a comprehensive view of the system's performance. In facilitates, time series graphs and other gauge meters to showcase current values of the metrics.

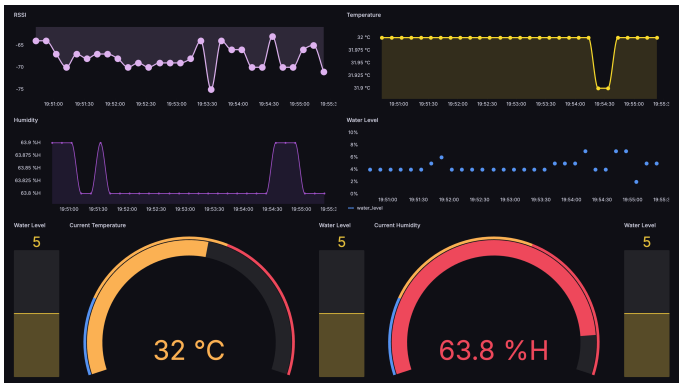


Fig. 2 : Grafana Dashboard with metrics visualisation

The dashboard includes the following panels:

1. Real-time water level display with historical data
2. Temperature and humidity readings
3. Forecasted water levels for the next 24 hours
4. Alarm events log
5. Wi-Fi signal strength (RSSI) over time
6. Data transmission latency graph
7. Forecast accuracy metrics over time

This visualisation allows for easy monitoring of all system aspects and quick identification of any anomalies or trends in pet water consumption.

The results demonstrate the system's effectiveness in monitoring pet water intake, generating timely refill alerts, and providing accurate water level predictions. The low latency ensures real-time responsiveness, while the forecasting model shows promising accuracy in predicting future water needs, enabling proactive pet care.

ACKNOWLEDGMENT

I would like to express our sincere gratitude to Prof. Marco De Felice and Prof. Luciano Bononi for their invaluable guidance, support, and expertise throughout the course of this subject. Their insights and feedback were instrumental in shaping the direction and successful implementation of the smart water bowl system. We also thank them for providing access to resources and fostering an environment conducive to learning and innovation.

REFERENCES

1. D. J. Cook, J. C. Augusto, and V. R. Jakkula, “Ambient intelligence: Technologies, applications, and opportunities,” Pervasive and Mobile Computing, vol. 5, no. 4, pp. 277–298, 2009.
2. J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, 2013.
3. A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A survey on enabling technologies, protocols, and applications,” IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015.
4. L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” Computer Networks, vol. 54, no. 15, pp. 2787–2805, 2010.
5. S. Li, L. Da Xu, and S. Zhao, “The Internet of Things: A survey,” Information Systems Frontiers, vol. 17, no. 2, pp. 243–259, 2015.