Rainwater Tank Monitoring: Taking the Smart Home to the Yard

Yasmine D. Subbagh

***Abstract*—This paper presents the design and implementation of a remote monitoring system for residential rainwater harvesting tanks, aimed at improving water management for home gardeners. The system utilizes an ESP32 microcontroller and an ultrasonic sensor to measure water levels in the tanks. This data is transmitted to AWS cloud services via MQTT, using AWS IoT Core, DynamoDB, and Lambda functions for real-time monitoring and data processing. The system also includes an EC2 hosted web application that allows users to view tank levels remotely and receive notifications when the water levels reach predefined thresholds.**

The project addresses the challenges of water tank monitoring by automating the data collection process and providing notifications, reducing the need for manual checks. It is designed to be low-power, affordable, and easily replicable, leveraging widely accessible hardware and AWS's free-tier services. Additionally, the system is adaptable for future enhancements, including the integration of machine learning models to predict harvesting times and consumption patterns, further optimizing rainwater use. The design also considers outdoor durability, suggesting 3D printing for custom enclosures and the potential use of mechanical sensors for increased longevity.

This work not only provides an efficient solution for home gardeners but also lays the foundation for a scalable platform that could be expanded to meet broader agricultural needs. By combining IoT, cloud computing, and low-cost hardware, the project demonstrates the potential of smart technologies in sustainable water management.

***Index Terms*— Amazon Web Services (AWS), Automated Reports, ESP32, Internet of Things (IoT), Smart Home, Rainwater Harvesting, Water Tank.**

I. Introduction

PRACTICING environmentally friendly activities in ones day to day life is becoming more popular as global warming warnings rise. It is important to reduce, reuse, and recycle in order to limit our individual negative footprints and impact on the environment. One popular method to conserve water for gardeners who live in rainy climates is to collect rain water to use later when watering their plants. However, many gardeners struggle with monitoring the water levels in their rain collection tanks. This project aims to create a system that allows green thumb gardeners to focus their energy on other green initiatives and stay dry rather than having to manually check the state of their water drums. The reason why I chose this project is because I have relatives close to me who have faced issues with monitoring their rainwater tanks and losing out on free resources because they didn't want to bear the cold or merely forgot about them.

Living in Washington state, while it may feel gloomy and saddening, we are lucky to receive 66 inches of annual average rainfall in western Washington and an average of 156 days of rain annually. The average water consumption of a single-family home in Seattle, WA is 52 gallons [1], and according to the EPA, 30% of residential water use is used outdoors [2]: equating to the average single-family home in Seattle, WA using 15 gallons of water a day on their yards (lawn and gardens). Collecting rainwater for reuse is not only beneficial to the environment, but also benefits the homeowners economically. The average cost of a gallon of water in Seattle, WA is three cents [3], while this may seem like a nominal amount, with the average water consumption of a garden, it comes out to $11.79 a month.

While the rain collector can see direct economic gains from collecting rainwater, there are even more environmentally friendly reasons to do so. Rainwater is better for the plants being gardened. The water is 100% soft, meaning it is not chlorinated and is higher in nitrogen and has a more elevated pH compared to tap water [5]. Additionally, by collecting rainwater, less water is going to rain runoffs [4] which helps save pipes and avoids corrosion and solves drainage problems on the property. Lastly, rainwater can be used in emergencies for indoor use when properly treated [6].

In order to maximize their efforts to help the environment and to minimize public utilities water bills, many households have multiple water tanks. The issue with multiple tanks is managing when to fill which tank as they fill up, many homes have one singular tank collecting run off at a time and shuffle them around as they fill up. Another issue that is present for all water tank users is being able to monitor their consumption during the gardening season. This makes creating a monitoring system accompanied by a notification service an appropriate step in adding the yard into a smart home household, for a tech forward and simplified lifestyle.

By creating a system to help gardeners manage their water

This paper was submitted for review on December 2nd, 2024. The project was in support by the authors parents and close relatives.

Yasmine D. Subbagh is a graduate student in Master of Science in Computer Science and Software Engineering at the University of Washington – Bothell (UWB) is affiliated with UWB's Inclusivity, Diversity, Equity, and Accessibility (IDEA) Lab and worked previously at OneRadio Corporation. (email: ydns@uw.edu)

collection system, a single household can save and repurpose tens if not hundreds of gallons of water a year. The system will be implemented using an ultrasonic sensor which will monitor the level of the water in the drums. The water level collected will then be viewable for immediate monitoring and will also trigger notifications to the users if the tanks are nearing full or empty. The system will use Amazon Web Services (AWS) to store and process the data in the cloud.

With the increased availability and accessibility of microcontrollers and peripherals and cloud computing services offering free services with improved documentation. This system can easily be replicated by determined gardeners to create a smart yard. The goal of the project is to create a system that can be easily replicated.

The contributions made to this project include the hardware setup and creation of the programs to be included in both the device-level and cloud-level sections of the system, in addition to the configuration of a server. The development of the programs required research for not only the devices and cloud service documentation: but also, tutorials that instruct on how to integrate the different hardware and software systems together, as well as tutoring on the hardware and development of a data pipeline in the cloud.

II. Related Work

While there are commercial water level monitoring systems available to purchase, they vary in scale and methodology. Many of these systems focus on systems that are similar in size for smaller tanks, but more importantly, singular device systems. Other papers focus on the benefits of rainwater harvesting.

In a paper by Abdullah et al. a water level monitoring system is explored using a PIC microcontroller 18F452, alarms, and LCD screens [7]. The system is able to monitor the water level in a tank and notify users via an alarm and LCD screen of the current water level. While the user goals and hardware are similar, this paper focuses on a local development of the system, the water levels are not able to be monitored remotely.

A study completed by Das et al. again uses a microcontroller to monitor the water levels of a tank and process the data on-board to pump water into the tank as water levels drop [8]. The system is fully autonomous, again however, is at a local development scale. The system does not have a singular goal they are trying to solve but is able to be reused for specialized uses.

Another study by Ismail, Azizi, and Zariman studies the validity of an internet of things (IoT) system that tracks the water levels of damns/rivers and allows damn technicians to open and close the damn gates from their personal devices [9]. It was found that the hardware system itself could lead to safer working conditions and allows more visibility of the damn status for residents along the river that could be impacted.

While there are contributions to previous studies on the validity and benefits of water level monitoring systems, none aim to solve the same problem, nor do they use the same hardware and software combination. In a study by Baballe a similar system is built, however a wire water level sensor is used [10], while the system works, for larger scale systems and water drums, it can be costly and technically expensive to maintain such a sensor.

Work completed by Campisano et al. explores the benefits and downfalls of urban rainwater harvesting systems. It is found that while pure rainwater is not the safe for human consumption unpurified, with treatment, becomes potable. It was also found that at scale that rainwater harvesting is not financially viable, but with smaller raw water extraction that is strictly outdoors (do not need treatment and plumbing), the return on investment is positive.

The main contributions of the project to the problem are to provide an automated water level monitoring system for outdoor residential use. This is achieved by using readily available and affordable resources, while still achieving the goal of creating a system that maintains accurate report of current water levels and triggers notifications to users of discrepancies. The impact of this approach is that an ultrasonic sensor is used to determine the water level as opposed to mechanical approaches or wire water monitors, this allows for better precision and less room for hardware failure. Another impact of this approach is the use of cloud services computing rather than on-board computing, in addition to multiple tanks being monitored in contrast to a singular device. This allows for precise remote monitoring of many water tanks, giving home gardeners accurate results within the comfort of inside their home or on the go.

III. System Model, Problem Statement, & Analysis

A. System Model

The key modules of the system can be broken down into two major groups: the physical hardware devices, and the cloud resources and applications. Within the physical hardware devices lie the ESP32 microcontroller and ultrasonic sensor. The cloud resources include AWS IoT Core, AWS Lambda Function, Amazon DynamoDB, Amazon Application Programming Interface (API) Gateway, and Amazon Elastic Compute Cloud (EC2). Figure 1 shows the relationship between each module within the system.

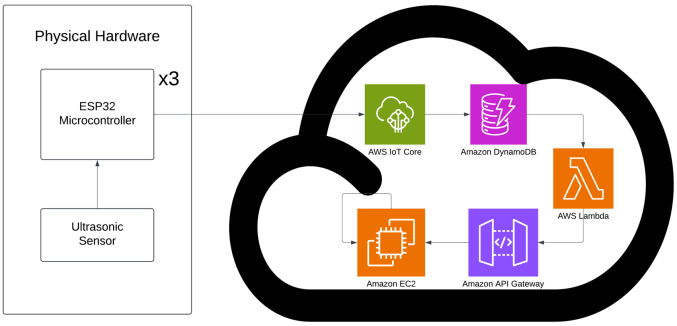

Figure 1: System Diagram of Hardware and Cloud Resources

The ESP32 is a microcontroller released in 2015 by Espressif Systems with Wi-Fi and Bluetooth capabilities built-in while still maintaining affordability and security at just $5 a unit [11]. As seen in Figure 2, the ESP32 is compact and supports the use of many types and quantities of peripherals. Maier, Sharp, and Vagapov showcase that the ESP32 is capable of excellent performance and is great for real time applications (like IoT) [12]. The ESP32 was chosen for this project due to its small package, Wi-Fi capabilities, affordability, and the ample support and documentation for developers along with simple integration use with AWS. The integrated development environment chosen to implement on the ESP32 was the Arduino IDE, as it includes libraries and support for the ESP32 devices as well as its simplicity, it was chosen over PlatformIO via VSCode because of its native integration. On the ESP32, the Arduino library and other open-source libraries will be used, including the ArduinoJSON, PubSubClient, WiFiClientSecure, Wifi, and Wire libraries.

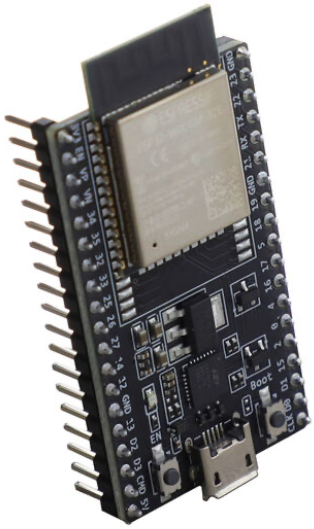

Figure 2: Image of the ESP32 Microcontroller

For the sensor collecting data within the system, an ultrasonic sensor was used. An image can be seen in Figure 3 below. The ultrasonic sensor emits high-frequency sound waves (ultrasonic waves, they are above the human hearing range) towards the surface of the water where the waves bounce off and return to the sensor, using the time amount for the waves return (speed of sound), the distance from the sensor to the water surface can be calculated [13]. This sensor was used due to its affordability, ease of access, and distance accuracy. The main input variable for this project system is the distance of the water surface from the sensor (mounted at the top of the water tanks), the output will be notifying the user of predefined alert levels, as well as a web app for on-the-fly monitoring.

Figure 3: Image of an Ultrasonic Sensor

AWS acts as the main hub for the system, facilitating the MQTT communication from the devices and the cloud, as well as data and application storage, and serving as an endpoint for data retrieval. It offers a user interface (UI) that is easy to use while still allowing for complex functionality and processing of the collected data. AWS IoT core acts as the main entry point of the collected data into the cloud service. The distance data from each of the devices is published to AWS IoT Core via MQTT where it is then redirected to AWS DynamoDB via a message SQL rule. AWS DynamoDB is where the data is housed within the cloud, it was chosen because of the scale of the project and the simplicity of the data being inputted, the NoSQL database service that supports key-value storage and retrieval [14]. An AWS Lambda function is used to support an AWS API Gateway. The lambda function retrieves the distance of the water surface within a certain water tank given the water tank device identification number from the DynamoDB table. The AWS API Gateway GET call uses this lambda function to allow external and simple retrieval of the data, returning the values within a JSON message. An AWS EC2 server is used to host and run the web apps that connect the user to the data. A python script on the server is ran every 30 minutes via a cron job, this application notifies the user via email if each individual tank is near empty or full (by 20%). Also hosted on the same EC2 server is a flask web application that gives the user remote monitoring of the water tanks levels, allowing them to track the water level for all the tanks in one location, no matter how full the tanks are. Using this collection of AWS services allows for the entire system to remain within the free tier provided by AWS, making the cost burden of the system only the upfront hardware costs.

B. Problem Statement & Analysis

The problem that is being addressed in this project is the difficulty to monitor the water levels of residential gardeners' rainwater harvesting tanks, as it is a burden to do so during the rainy season in the pacific northwest (PNW) with multiple water tanks. Many factors make monitoring the water tanks a hassle, mainly the inability to predict when the tanks near full

during collection season, in addition to needing to switch which tank is currently connected to the roof water drainage system. Many residential gardeners have multiple water tanks, but only a few have a sophisticated system that are able to continuously automatically fill all of the tanks within the system, this creates an issue of needing to rearrange the tanks or change the tank currently hooked up to the roof water draining system. However, depending on the rain fall, a tank can fill up anywhere between within a day or up to a month, making predicting when to change the hookup very difficult. Another barrier is the poor environment often the process must occur in, going outside in the wet and cold is less than ideal and leads to long periods where rainwater collection can be wasted. The project's main purpose is to aid the home gardener by removing the necessity of having to predict the water levels in the tanks and keep the gardener in the more favorable warm and dry indoors. In doing so, the gardener will be less deterred by the processes of home gardening and will yield a higher return on interest (ROI). By implementing this system, efficient monitoring and usage of rainwater can save household tens of gallons of water usage annually. This contributes to both environmental conservation and reduced utility bills, encouraging sustainable practices among gardeners.

The assumptions in place for this project are that the developers trying to implement this project has access to the aforementioned physical devices. The implementations on the cloud resources for this project can be done within the Free Tier of AWS cloud services. In addition, it is assumed that this project is to be used by residential rainwater harvesters within a small scale that do not intend on purifying the water they collect. Larger scale (commercial) projects require more infrastructure that will push the data being transferred and stored in the cloud to push cloud resource consumption over what is allowed within the AWS Free Tier. Additionally, water purification is costly and complicated, which would lead the financial burden of the system to equate to a negative ROI. But as this system is intended for home use, purification and large-scale projects will not be considered.

IV. Design and Implementation

There are subgroups within the project system that require different techniques and tools in order to implement the system to satisfactory.

A. Physical Devices

The physical devices involved in this system are the ESP32, ultrasonic sensor, and external power supply (portable power bank). Both the ultrasonic sensor and the power supply are connected to the ESP32, the ultrasonic sensor is connected to the general-purpose input/output (GPIO) pins using female-to-female jumper wires and the power supply is connected directly to the microcontroller via USB-C. During the implementation for this project, it was found that the voltage common collector (VCC) pin on the ultrasonic sensor had to be connected to the 5v output on the ESP32 opposed to the 3.3v output to properly power the device. The pins used for this implementation were P12 (echo) and P13 (trigger).

With regards to the system, since it is an electrical circuit, it is susceptible to water damage, and as the system resides within a water tank, it is at high risk to damage. While a custom designed 3D printed enclosure would be ideal, any plastic enclosure sealed with duct tape is used within the implementation of the project. It is also important to note that the ultrasonic sensor must be oriented in the correct direction to accurately measure the distance of the water surface within the tank. The sensor should be aimed directly down towards the floor of the drum. Additionally, the project was implemented with 33.5-inch (internal) height water drums, this is within the sensors reading distance, large tanks with a height over 60-inches would not be able to calculate the distance to the bottom of the drums.

The voltage for the sensor as previously mentioned is five volts, this makes for a low-voltage option. While most ultrasonic sensors are 5v, some can draw as little power as 3.3v and would allow for the power supply life to be extended. The power supply has not been pushed to full drainage, currently the expected battery life is unknown. This is likely due to the small power draw of the microcontroller and its peripheral, running in deep sleep when not transmitting data.

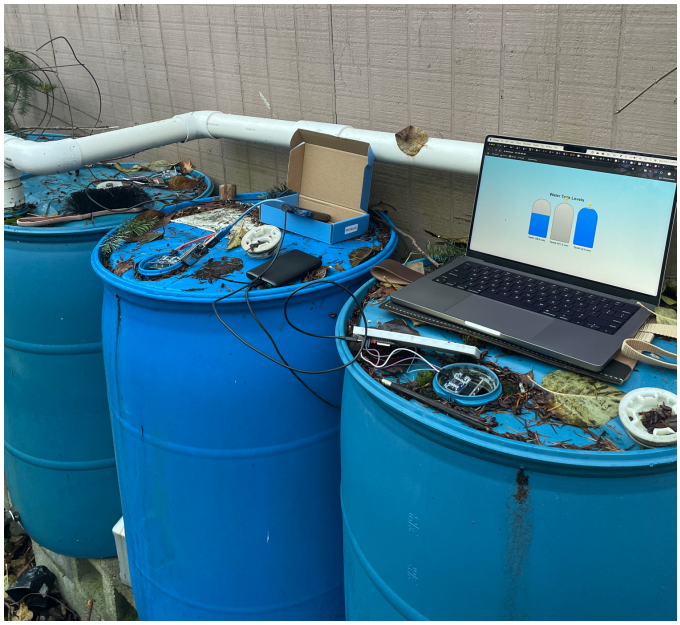

Figure 4: An image of the Physical System, including: microcontroller, ultrasonic sensor, and external power supply.

The physical device will have a program for implementing the functionality of the microcontroller, as well as connecting to Wi-Fi and authentication information needed to access the cloud services. The initial setup method in the code base on the physical devices connects the devices to Wi-Fi, then creates a connection to AWS, and lastly sets the pins to the appropriate modes (input and output). The continuously running code base is compiled and uploaded to the devices (ESP32) via the Arduino IDE. The distance measurements are taken every 20 seconds, this is to allow for accurate readings during heavy rainstorms which can lead to fast rising water tanks levels. When taking the distance measurements, the distance is

calculated via the time it takes for the ultrasonic waves to return to the sensor, requiring on board mathematical calculations. Once the distance is calculated, the data must be converted into a JSON message format in order to be sent to AWS IoT core via MQTT protocol. This will be done using the open-source ArduinoJSON and PubSubClient library.

When uploading this program to the ESP32, it is possible to run into many issues and/or failure messages. The main one being timeout issues. This is because the Wi-Fi connection handler, AWS connection handler, and main program, all run within a while-loop, if incorrect credentials are passed it can cause timeout issues.

B. Cloud Resources

In the AWS IoT Core, the three ESP32s are setup as new Things (can be done individually or as a family), this is where all MQTT certificates and keys are created. These certificates and keys need to be saved and kept within the secrets file of the code base onboard the physical devices to be used when connecting the ESP32s to the cloud resource. For the Thing policy of each of the ESP32s, a policy must be attached to each of their certificates, the policy should allow the device to connect, publish, and subscribe to IoT Core. Each device has its own topic it publishes to: "water1/pub", "water2/pub", "water3/pub".

In AWS DynamoDB a table must be created to store the data and easily lookup the data given the key. From IoT Core, a message routing rule must be made for each channel to repost the data that is sent into IoT Core to the DynamoDB table. Each device will post to its own topic on IoT Core, each rule will redirect all posts from its topic to its corresponding row in DynamoDB (given by the device identification number). The key for the table is the device_id (number: 1-3) and the current distance will be continuously updating (past values will not be stored).

An AWS Lambda function will support the AWS API Gateway REST API. The python script embedded in the lambda function queries the DynamoDB table, retrieving the most recent distance value for the given device. The API Gateway passes a device_id number to the lambda function, where the function then retrievers the most recent distance value from the DynamoDB table, from there the lambda function passes the value back to the API Gateway where it is packaged into a JSON message for the response to the GET call. The REST API can be used from anywhere, in and out of AWS.

To support the user accessed applications, an AWS EC2 VM server hosts two applications. The first application sends the user email notifications every 30 minutes if the water level is within a warning zone. This is accomplished using a cron job that executes a python script that checks the current water level of each water tank, and for each tank, if it is within a full or empty warning zone, sends an email to the user.

The second application that is hosted on the EC2 server that is presented to the user is a web application. The web app serves as a way for the user to use the system year-round. By allowing for the user to see the current water levels year-round, they are able to predict how long the harvested water will last them and when all tanks are full to disconnect the system to avoid flooding. The web app is hosted on the EC2 server, viewable form any device with a minimal but clean design, and up to date with helpful icons to indicate when tanks are near full or empty.

C. Connecting the Cloud Resources to the Physical Devices

The certification files and private keys from the previous section should be stored where accessible from the device-level program. In this implementation, a secrets.h header file was created to hold all the keys and other private information including the Wi-Fi SSID and passwords.

To connect the device to the cloud, the device-level program must contain steps to enable connecting to AWS IoT Core, publishing messages, collecting, and processing distance data. In the step to connect to AWS IoT Core, first the program must connect to the internet using the Arduino Wi-Fi library and SSID information in the secrets.h header file. Next, the client is to connect to the AWS IoT Core endpoint using PubSubClient library. This step involves the use of the certification files and keys. Once completed, the client will be connected to its individual Thing (ex. Esp32-water) and be subscribed to its owned topic (ex. "Water2/pub").

In the code to publish a message, a JSON document must be created so that it can be used to transfer the data using the MQTT protocol. The JSON document will include the sensor device identification number and the computed distance value for the water surface from the ultrasonic sensor. This is then published use the PubSubClient to the water(1/2/3)/pub topic.

The MQTT topics used in this system are water1/pub, water2/pub, and water3/pub. The raw data collected by the ultrasonic sensors is sent to the devices assigned topic. Since the AWS IoT console allows users to modify the message routing and publish their own messages to topics, users are able to modify the notifications that are sent (or lack of) via email. The device, topic, database pipeline architecture that the applications retrieve that data from can be seen in figure 5.

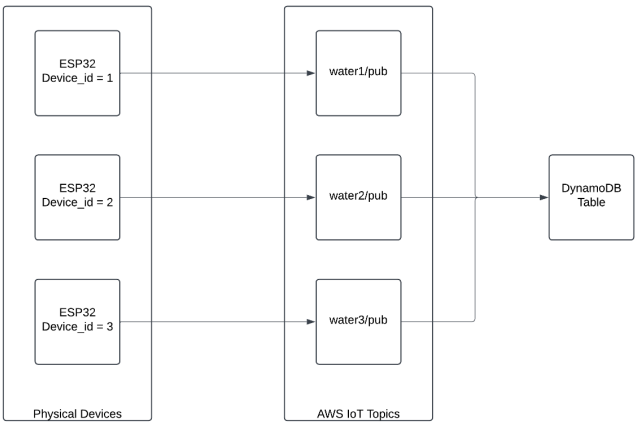

Figure 5: Physical Device to AWS IoT Topic Publishing Route

V. EVALUATION

In order to evaluate the accuracy and efficiency of the system, an experiment was conducted. Additionally, with the

rising concerns to IoT security, some research was completed. To assess the accuracy and efficient of the system, manual measurements of water level distances were taken. To review the security of the system, more research was completed on the security of ESP32s and the possibility and ease of penetration to the system and its data.

In order to evaluate the accuracy of the system, an experiment was pursued to capture the distance readings and accuracy in comparison to manual measurements. To do so, the readings of the water tank were taken with the device and compared to a manual reading. Manual readings were taken via a ping pong ball attached to a string and dropped into the tank, measuring the full length of the ball and string with no slack (the ball floats on the water surface). The mean value for the difference in measurements was .5cm (devices measure at whole unit increments). The discrepancy could be a result in human error during manual measurement, or errors in the sensor. What could cause errors in the sensor are: miss calculations, ultrasonic waves bouncing off other surfaces, etc.

Next, to evaluate efficiency in the system, ensuring the system was able to continually measure water levels as they rose needed to be tested. To align with the goals of the system, a 5-gallon bucket was used instead of the 55-gallon water tanks the device normally monitors. The device was placed at the top of the bucket and the code was modified to post to IoT Core every second. The bucket was slowly filled up with water via a garden hose. Results found that as the water flowed into the bucket, the distance value from the device continued to decrease. This shows that the system was able to accurately and continuously calculate and post the distance data.

Lastly, an experiment was created to check the devices robustness via its ability to reestablish an internet and AWS connection. With the device functioning as intended, it was brought away from the Wi-Fi router to cause the internet and therefor AWS connection to be lost, it was then returned to its original position. This experiment was to mimic conditions if electricity were to go out, internet outage, and to measure how far the system could reach away from the home. It was found that the device was able to reconnect to the cloud services 80% of the time (8/10). The failures happened due to the device booting into sleep mode before it was able to reconnect.

These tests prove that the system is accurate, efficient, and robust. While the sensor was not able to complete exactly accurate distance reads with and was therefore rounded for simpler handling, given the needs of the users, it is far more than accurate to complete the tasks at hand. And by using the sensor over manual efforts to monitor, time and energy is saved. The system is efficient and reliable, the system was able to accurately monitor the water levels at they rose, ensuring the users that the system can be trusted to given accurate results in real-time. Lastly, the system is robust in being able to reconnect itself without the need of user intervention. Allowing for a hands-off system that can withstand outside forces.

However, the ESP32 is at risk to security threats. A study done by Barybin et al. found that even unexperienced hackers can easily penetrate the systems hosted on ESP32s and send fake data over the to their corresponding web interfaces [17]. While this poses as a threat to loss of user data, this system does not host data or operate outside of the system that poses any real risk to users. Modifying of data, while can lead to users being misled on their current rainwater tanks levels, pose no information or physical threat. However, if personal data were to be stored on board within the code base of the device, private data leakage could become a larger issue.

VI. Conclusion

The purpose of this project was to create a system to aid the home gardener with monitoring their rainwater harvested collection tanks by creating a remote access monitoring application in addition to pre-defined triggered notifications. By integrating the ESP32 microcontroller, ultrasonic sensor, and AWS cloud services, the system automates real-time monitoring and notifications, removing the need for manual checks and enhancing the convenience of residential rainwater management.

The use of AWS IoT Core, DynamoDB, Lambda Function, API Gateway and an EC2 hosted web application ensures seamless data collection, processing and user accessibility. The system's low power consumption, accessible hardware, and adherence to AWS Free Tier makes it an affordable option for home gardeners. The equipment used for the project was affordable and easily accessible for any persons for this project to be replicated.

The project not only address the immediate challenges of water tank monitoring but also lays the groundwork for a scalable and adaptable platform that could a broader audience with additional features (outlined in the next section). It serves as a valuable step towards leveraging IoT and cloud technologies for a smarter yard.

VII. Future work

The implementation of this project included measuring and calculating the distance / level of the water tanks and notifying the users when the tanks are near empty or full. In future iterations many modifications and improvements can be made to create a more complex and robust system that can serve more of the user's needs. The system can be better developed to withstand the wet and outdoor environment. In addition, collecting outdoor data and implementing a machine learning (ML) model can help gardeners predict harvesting times and consumption use.

The physical device sits at the top of the tank peeping through one of the water pass holes, it is currently protected via another plastic tub on top of it. Over time, this tub can be removed by wind or other circumstances, additionally, water can seep in or rise out of the tank and compromise the device. The first addition to the project to help protect the device is design and creating a secure housing case for the device to better protect it from the outdoors. With affordability and customization in mind, this could be done via 3D printing. This would allow for custom measurements and low costs that would allow for an iterative and fast design.

Another sensor system that could be used as opposed to the ultrasonic sensor is using a mechanical water level measuring system as seen in fuel gasoline tanks. While this would increase the cost of the physical device drastically and require much more expertise, it would ensure the longevity of the system. The mechanical sensor would not be prone to damage by rising water levels in the tank.

To create a more complex system, a ML model could be implemented to forecast the harvesting and consumption of rainwater. By collecting forecasted precipitation data from an outside source like the NOAA or Open-Metro's API [16] (available for free), as input data for the first ML model and the corresponding rise in water levels from previous rainstorms as the output, the model could predict when the tanks will fill up. A second ML model could take previous water consumption as an input in addition to forecasted temperature data to predict when the water tanks will be empty / rate of consumption.

By using these ML models, the remote web application would become a powerhouse for the home gardener and rainwater harvester who is trying to maximize their water efficiency and conservation. The web app in addition to the notifications would be able to notify the user in advance and give an estimated date of empty and full tanks. In addition, it could help the harvester compute if more tanks would be in their best interest moving forward and denote a positive or negative ROI.

All the above methods of modifying and improving this system could be useful to increase the longevity and resourcefulness of this project.

REFERENCES

[1] "Residential Drinking Water Rates - Utilities | seattle.gov," *www.seattle.gov*. https://www.seattle.gov/utilities/your-services/accounts-and-payments/rates/water/residential-water-rates (accessed Dec. 03, 2024).

[2] US EPA, "US Outdoor Water Use | WaterSense | US EPA," *Epa.gov*, 2019. https://19january2017snapshot.epa.gov/www3/watersense/pubs/outdoor.html (accessed Dec. 03, 2024).

[3] G. B. / F. Guy, "Rain-soaked Seattle has nation's highest water bills," *The Seattle Times*, Apr. 30, 2015. https://www.seattletimes.com/seattle-news/data/rain-soaked-seattle-has-nations-highest-water-bills/ (accessed Dec. 03, 2024).

[4] L. Grossman, "Rainwater collection - Washington State Department of Ecology," *ecology.wa.gov*. https://ecology.wa.gov/Water-Shorelines/Water-supply/Water-recovery-solutions/Rainwater-collection (accessed Dec. 03, 2024).

[5] B. Grant, "Pros And Cons Of Using Rain Water For Plants," *Gardening Know How*, Dec. 14, 2022. https://www.gardeningknowhow.com/garden-how-to/watering/rainwater-versus-tap-water.htm (accessed Dec. 03, 2024).

[6] G. Author, "Rain Collection: How Using a Rainwater Collection System Can Help You Survive," *Valley Food Storage*, May 30, 2023. https://valleyfoodstorage.com/blogs/inside-vfs/rainwater-collection-water-collection-system?srsltid=AfmBOop_fnF5yStG6aHObVdLSdr8ZBKhQhZjiMpIH0rJ3C-ToPnIW_oq (accessed Dec. 04, 2024).

[7] A. A. Ismail, M. A. Azizi, and A. Zariman, "Smart Water Level Indicator," *International Journal of Recent Technology and Applied Science*, vol. 2, no. 1, pp. 48–58, Mar. 2020, doi: https://doi.org/10.36079/lamintang.ijortas-0201.59.

[8] S. Das, S. Dhar, P. Deb, and P. S. Majumdar, "Microcontroller Based Water Level Indicator and Controller," *Asian Journal of Applied Science and Technology (AJAST)*, vol. 1, no. 5, pp. 181–182, Jun. 2017, Accessed: Dec. 03, 2024. [Online]. Available: https://iijsr.com/data/uploads/999.pdf

[9] A. A. Ismail, M. A. Azizi, and A. Zariman, "Smart Water Level Indicator," *International Journal of Recent Technology and Applied Science*, vol. 2, no. 1, pp. 48–58, Mar. 2020, doi: https://doi.org/10.36079/lamintang.ijortas-0201.59.

[10] Muhammad Ahmad Baballe, M. ibrahim Bello, A. S. Muhammad, and A. Muhammad, "Automatic Water Level Indicator: A Review," *3rd International Conference on Applied Engineering and Natural Sciences*, Jul. 2022, doi: https://doi.org/10.5281/zenodo.8146408.

[11] "ESP32-C61 Delivering Affordable Wi-Fi 6 Connectivity | Espressif Systems," *Espressif.com*, 2024. https://www.espressif.com/en/products/socs/esp32-c61 (accessed Dec. 04, 2024).

[12] M. Babiuch, P. Foltynek, and P. Smutny, "Using the ESP32 Microcontroller for Data Processing," *2019 20th International Carpathian Control Conference (ICCC)*, May 2019, doi: https://doi.org/10.1109/carpathiancc.2019.8765944.

[13] J. S. Cook, "Ultrasonic Sensors: How They Work (and How to Use Them with Arduino)," *Arrow.com*, Apr. 04, 2018. https://www.arrow.com/en/research-and-events/articles/ultrasonic-sensors-how-they-work-and-how-to-use-them-with-arduino (accessed Dec. 03, 2024).

[14] AWS, "Amazon DynamoDB - Overview," *Amazon Web Services, Inc.*, 2024.https://aws.amazon.com/dynamodb/ (accessed Dec. 03, 2024).

[15] "Senix Ultrasonic Sensor FAQs | Senix Corporation," *Senix Distance and Level Sensors*. https://senix.com/faqs/ (accessed Dec. 03, 2024).

[16] "⚡ Features | Open-Meteo.com," *Open-meteo.com*, 2022. https://open-meteo.com/en/features#available-apis (accessed Dec. 04, 2024).

[17] O. Barybin, E. Zaitseva, and V. Brazhnyi, "Testing the Security ESP32 Internet of Things Devices," *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, Oct. 2019, doi: https://doi.org/10.1109/picst47496.2019.9061269.