EE368 Final Project

Authors: Marc Ivan Mañalac, Osiel “Alex” Montoya, Eliya Nakamura, Andee Gary

Date: May 10, 2022

Abstract: This paper covers the development, design, and operation of our Watchtower IoT system, which will allow us to monitor the current conditions and alerts at four selected beaches. We discuss how we created our Node-Red flows both in IBM Cloud and on the Raspberry Pi. Furthermore, we explain how we assembled the hardware of the system, including the Servo and LED.

I. Introduction

From the years 2008 to 2017 there have been 368 reported ocean related deaths in Hawai'i. Of those deaths, over half were from visitors, and 305 came from our island of O'ahu. These activities included some of Hawai'i's favorite pastimes such as surfing, free diving, scuba diving, and boating [1]. The question we try to answer in this paper is how can we assist in spreading ocean safety awareness using an internet of things (IoT) application? This question led us to develop an IoT solution for sharing real-time beach condition information to the general public for beaches without lifeguards.

In the next few sections we will discuss Watchtower, our IoT solution for real time beach safety notification. Section two covers our system architecture which is split into two subsections discussing our system development and its operation. Section three concludes this report.

II. Architecture

A. System Development

When initially designing our IoT system, our main objective was to access near real-time data which was available through application programming interfaces (APIs) and transmitting that data to our Pi devices for use. Our solution meets this goal by using the Hawai'i Beach Safety REST API [2].

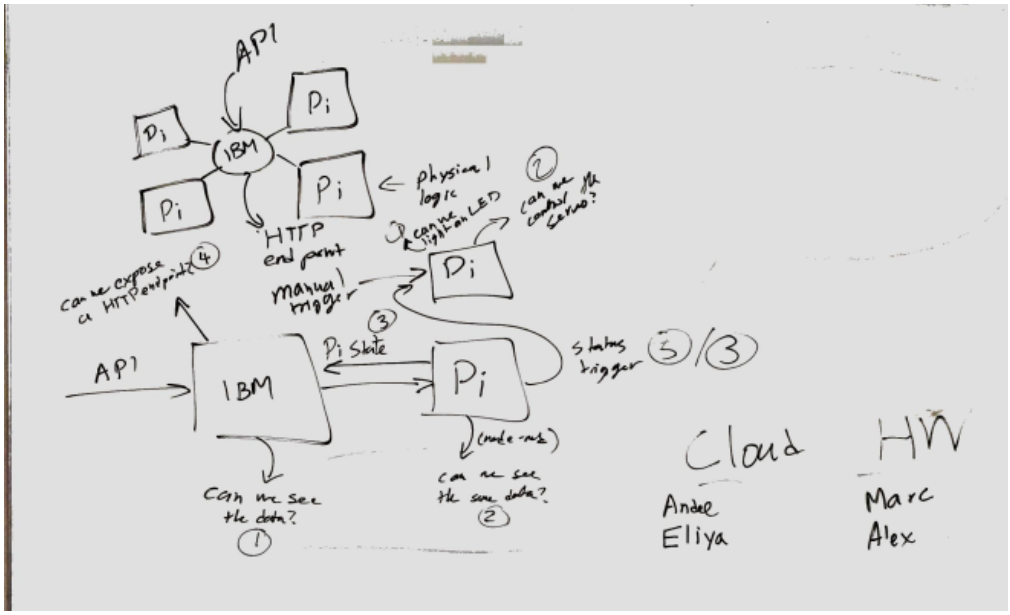

Figure 1. Our initial whiteboard design, including milestones

Our initial drawing board can be seen in Figure 1, above. During our first planning meeting, we determined the scope of our project, the data that we could access, and the functionality of our system. As shown at the top left of Figure 1, we planned for each of our Pis to act as individual nodes, connecting to a single IBM Cloud instance. These nodes would represent physical beach locations, and react to beach information exposed from the API. The Cloud would connect to and request data from the API. It would send individual beach data to each node, then expose a HTTP endpoint to show the status of all nodes to the user.

Below the network diagram in Figure 1, we planned out milestones to ensure the proper flow of data. These are shown as circled numbers in the diagram. First, we had to ensure the REST API call successfully returned data for us to work with. Next, that same data should be able to be transmitted to a Pi node.Third, we planned on retransmitting some sort of status back to the Cloud. We eventually decided against this, as we did not plan on attaching any sensors that would send any data back to the Cloud. Fourth, we had to expose a HTTP endpoint with the beach data for review. We then additionally had milestones for the Pi node electronics. First, lighting an LED through digital pin control, then controlling a servo.

A.1 Group Contributions

To implement the functionality of our system, the team split into two subgroups working in parallel: Cloud and Pi.

The Cloud group consisted of Eliya and Andee, who worked in the IBM Cloud to request data from the REST API. They began by setting up a flow in their IBM Cloud Node-Red instance as shown in Figure 2. They added an HTTP input node and connected it to two separate HTTP request nodes. Each node requested JSON data about four test beaches, separately returning beach condition and beach alert data.This data is then forwarded to JSON parsing nodes, which convert the data to Javascript objects to be used in the program.

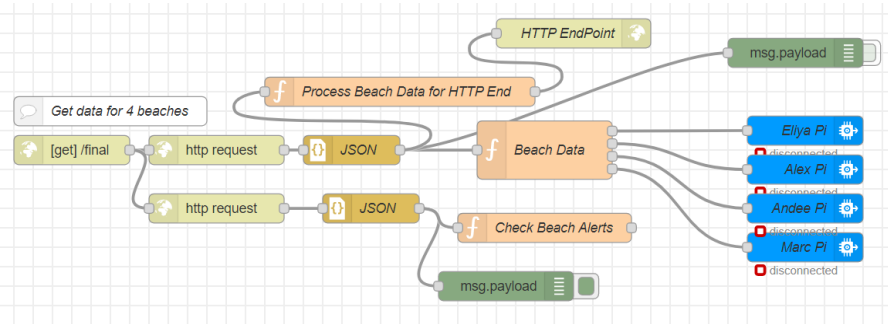

Figure 2. Cloud Flow

The alert data is sent to a function node, which determines the alert status of the four beaches and stores them in the flow. The beach condition data is sent to two function nodes. The first generates HTML containing the beach names and conditions to be forwarded to the HTML endpoint. The second extracts each of the four beach conditions and alerts so that they can be individually forwarded to the Pi nodes. Finally, each of the outputs of the beach data node is connected to four separate IBM Watson IoT output nodes, each corresponding to an individual Pi.

The Pi group consisted of Alex and Marc, who developed the functionality of the Pi nodes that each team member would operate. An individual Pi node consisted of a Raspberry Pi connected to the Internet, with a servo and LED attached to its General Purpose Input/Output (GPIO) pins. Upon receiving beach data from the Cloud, the Pi node would parse the nearshore conditions and send a Pulse-Width Modulation (PWM) signal to the servo to point an arrow to the corresponding position on the display. The Pi node would also parse whether that beach had an active alert condition, and would light an LED to alert users to check the Hawai'i Beach Safety website for further information. The overall flow of data from the API to a Pi node is shown in Figure 3.

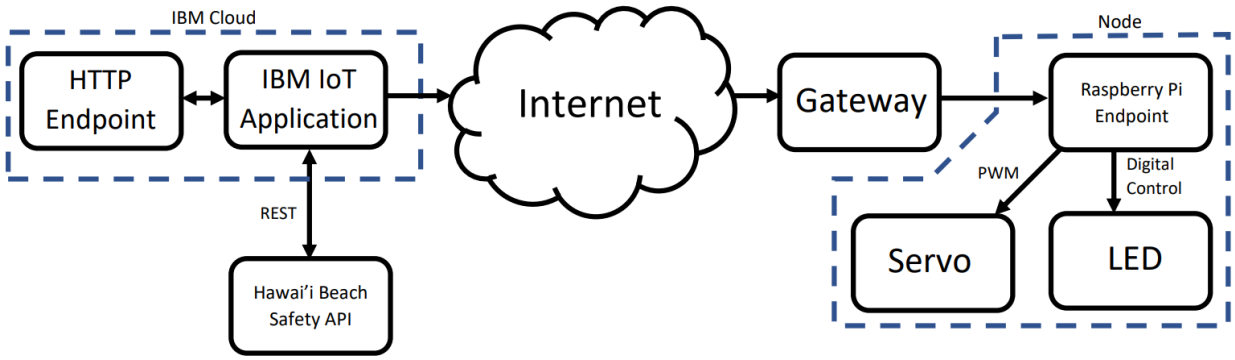

Figure 3. The data flow diagram.

B. System Operation

B.1 Hardware Setup

The wiring diagram for each Pi is shown below in Figure 4. Two devices are attached to each Pi: a servo, and an LED. The servo is connected to pins 2 and 6 for power and ground, and controlled by pin 3 [3]. The LED is connected to pin 5 for power and control, then connected to ground through a 220Ω resistor to pin 9.

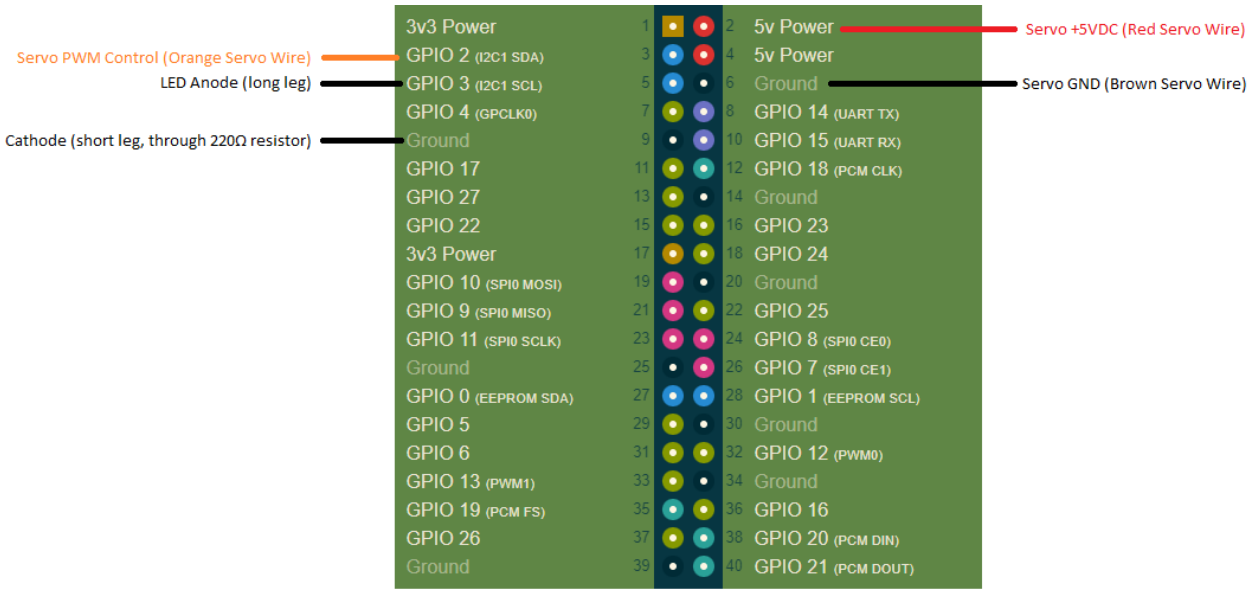

Figure 4. Our wiring diagram for each pi device [4].

The physical setup of the display hardware involves a cardboard display with the four possible conditions printed on it, with cutouts for the servo arm and LED. This display is shown in Figure 5. The servo is mounted with wires pointing up, such that 0° rotation is pointing left, and 180° rotation is pointing right. The connections between the servo, LED, resistor, and breadboard are made using Dupont wires.

Figure 5. The event display.

B.2 Software Setup

In order to properly use the servo with the Raspberry Pi, an additional Node-Red node, node-red-node-pi-gpiod, must be added in addition to the standard nodes. This is because the built-in rpi gpio out node is not adequate for controlling the servo.

To install this node,

1. Open a terminal, and run sudo vi /etc/rc.local
    a. rc.local is a script that runs on boot. This command edits the file with superuser privileges.
2. At the bottom of the file, above the line exit 0, add /usr/bin/pigpiod in its own line.
    a. This is the location of the PiGPIO daemon, which will now run on boot.
3. Save and close the file.
4. Change directory to ~/.node-red
5. Run npm install node-red-node-pi-gpiod
6. Reboot the Pi

On reboot, the node should be installed and visible in the palette [5].

B.3 Cloud Control

To update the API data in the Cloud, use the Cloud application URL and append /final to the end. This will both command the Cloud flow to request new API data, and direct the user to the HTTP endpoint that displays the four beach names and the conditions, as shown in Figure 6. In addition, the flow will update each connected Pi node with the most recent condition and alert data for its assigned beach.

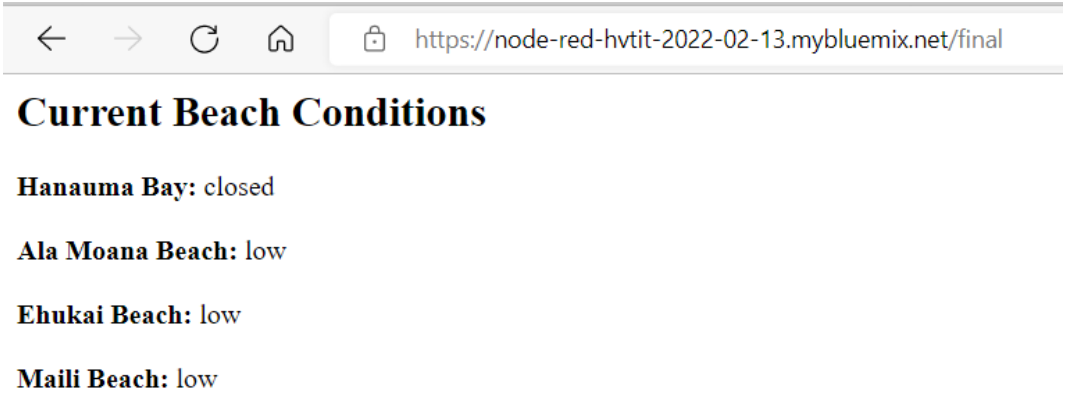

Figure 6. HTTP Endpoint.

B.4 Pi Control

The Pi node flow is illustrated in Figure 7 below. Debug nodes are connected both to the Cloud Input node and the output of the parse function nodes to observe the functioning of the flow. By default, these debug nodes are disabled. For manual testing, inject nodes for all possible conditions are attached to the inputs of the parse function nodes.

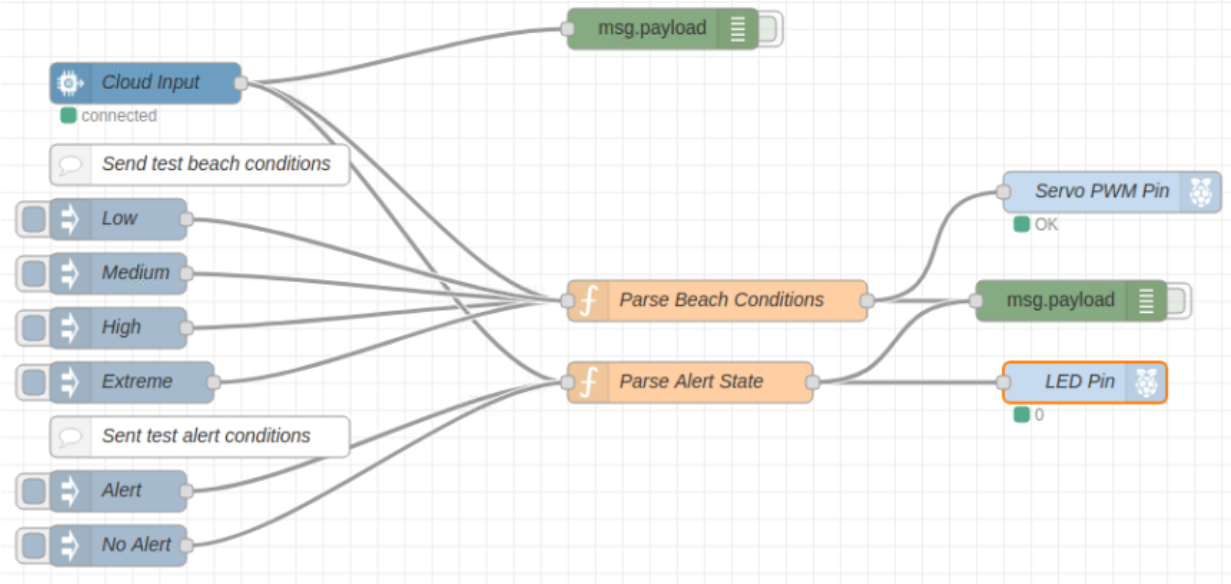

Figure 7. The Pi Flow

To properly align the servo arm, inject a “Low” beach condition and attach the servo arm such that it is pointing to the left. Cycling through the Low, Medium, High, and Extreme conditions should point the servo arm to approximately 0°, 60°, 120°, and 180° respectively.

In the Pi flow, the “Servo PWM Pin” node controls the servo by sending Pulse Width Modulation (PWM) signals at an operating frequency of 50 Hz. The limits of 0° and 180° are set in the node as the minimum and maximum duty cycle times. For the servos included in our kits, a minimum of 600 µs and maximum of 2500 µs allowed proper alignment of the servo arm to the display. Manual adjustment of these values may be needed in the event of discrepancies in servo arm angle.

III. Conclusion

Watchtower is capable of growing its capabilities with an endless amount of applicable technologies. We can utilize solar arrays and battery storage for long term operation without grid connection. It is also possible to equip LoRA RF PCBs for long range communication at some of the more hidden beaches like the ones in Kapa'a, Kaua'i. With the low cost of parts and integrability of the device, it can be easily built upon to increase the efficiency of our solution.

In this paper we have discussed the problem of beach related deaths and our possible IoT solution for increasing ocean hazard awareness. We have discussed our solutions architecture, how we developed the device utilizing real-world data from an API, how it is operated, and its future integration with other technologies. For every Hawai'i based problem there is an IoT solution waiting to be discovered.

References

[1] "Drowning & Spinal Cord Injury." HiOceanSafety.com. https://hioceansafety.com/drowning-spinal-cord-injury/ (accessed May 9, 2022).

[2] *Hawaii Beach Safety API,* HawaiiBeachSafety.com. [Online]. Available: https://hawaiibeachsafety.com/api

[3] "Servo Motor SG90 Data Sheet." Accessed: May 9, 2022. [Online]. Available: http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

[4] "Raspberry Pi Pinout." Pinout.xyz. https://pinout.xyz/ (accessed May 9, 2022).

[5] "node-red-node-pi-gpiod (node)." NodeRed.org. https://flows.nodered.org/node/node-red-node-pi-gpiod (accessed May 9, 2022).