Name: Lukas Jehle

ID: #20009320

Course: 158.335

Assessment: Assignment 1.1 IoT

Assessment Component: Report

Lukas Jehle, ID 20009320

The Problem

This prototype aims at solving two problems that are closely related to each other.

The first problem is that when someone wishes to take a shower or bath, they do not know if there will be a sufficient amount of hot water to do so. This will be especially relevant for households with a large number of inhabitants.

The second problem is that it is difficult for someone to control the temperature of their hot water tank quickly and easily. Unless they have the required skills, they would need to call a plumber every time they wanted the temperature of their hot water tank changed.

The Solution

The solution to both of these problems is to "thingify" the hot water tank in a building by turning it into a "smart hot water tank". In short, the hot water tank is the "thing" and, once digitally augmented, it will be a "smart thing". "Thingification" of the hot water tank will be achieved by installing a set of traffic light coloured LED indicator lights as well as a singular red LED indicator light on the outside of the hot water tank.
The traffic light coloured LED's will indicate to the user the current hot water temperature, while the singular red LED will indicate if the water is currently heating or not. The current temperature of the hot water tank will also be displayed in a web application that will be available to the user whenever they have access to a device connected to the internet.

Usage Scenarios

A typical scenario where the current temperature readout would be useful is when the user wishes to use the washing facilities after a gym session, but they prefer to use the facilities in their own home. In order to decide which facilities they should use, they could check the hot water temperature in their home by using the web application. Then, if there is enough hot water in their home, they could decide to wait to get home to use the shower or bath. Alternatively, if there is not enough hot

water, they could decide to either wait for the hot water to heat up at home or use the facilities at their gym.

Another useful feature that will be built into the web application will be the ability to control the temperature of the hot water. This will be especially useful for when the hot water will not be used for an extended period of time (e.g., when the house inhabitants all go away on holiday). The user could then simply set a lower temperature from the web application for when they are away from home.
Another usage scenario could be that the user wants to turn up the hot water temperature because they are finding that they are constantly running out of hot water. Again, they could then simply turn up the temperature from the web application.
Essentially, this feature gives the user full control over their hot water tank, allowing them to set the temperature to their own personal preferences. As a side effect, this also gives the user the potential to save money on their hot water bill as well as plumber call out fees for when they want the temperature changed.

Cloud IoT Concepts and Theory

A temperature sensor will be installed inside the hot water tank in order to read the temperature. The sensor is part of the sensing layer in the IoT architecture. The set of traffic light coloured LEDs and the singular red LED will also be installed on the outside of the hot water tank.

Both the sensor and the LEDs will then be connected to a Raspberry Pi, which will act as a gateway to the network layer. The network layer in the IoT architecture is what connects gateways to the data processing layer, which could be in a cloud environment.

Due to the small amount of data analysis required in this prototype, the decision was made to let the Raspberry Pi take on the role of the processing unit in the data processing layer. If the prototype were to be implemented on a larger scale, it would be likely that these services would be moved to a cloud-based system, which would be largely more dynamic and cost-effective.
In terms of data processing, the Raspberry Pi will be analysing the current

temperature and then setting the traffic light coloured LED's to the appropriate colour. Every 3 seconds, the Raspberry Pi will also be sending the current temperature to the Amazon Web Services (AWS) IoT service. Once processed, the current temperature state will be stored in a "shadow" device, which represents the physical state of the device in a virtual format.

There will also be another field called "temperatureSet" in the device shadow which will represent the temperature that the user wants the hot water tank to be set at. If the user wishes to change the temperature, they will need to enter the new temperature into the web application. The web application will then send an update to the temperatureSet field in the device shadow. Once the Raspberry Pi detects this request for temperature change, it will then set the status of the singular red LED accordingly.

Figure 1 below depicts IoT prototype system as a whole, including the different layers involved in a typical IoT system.

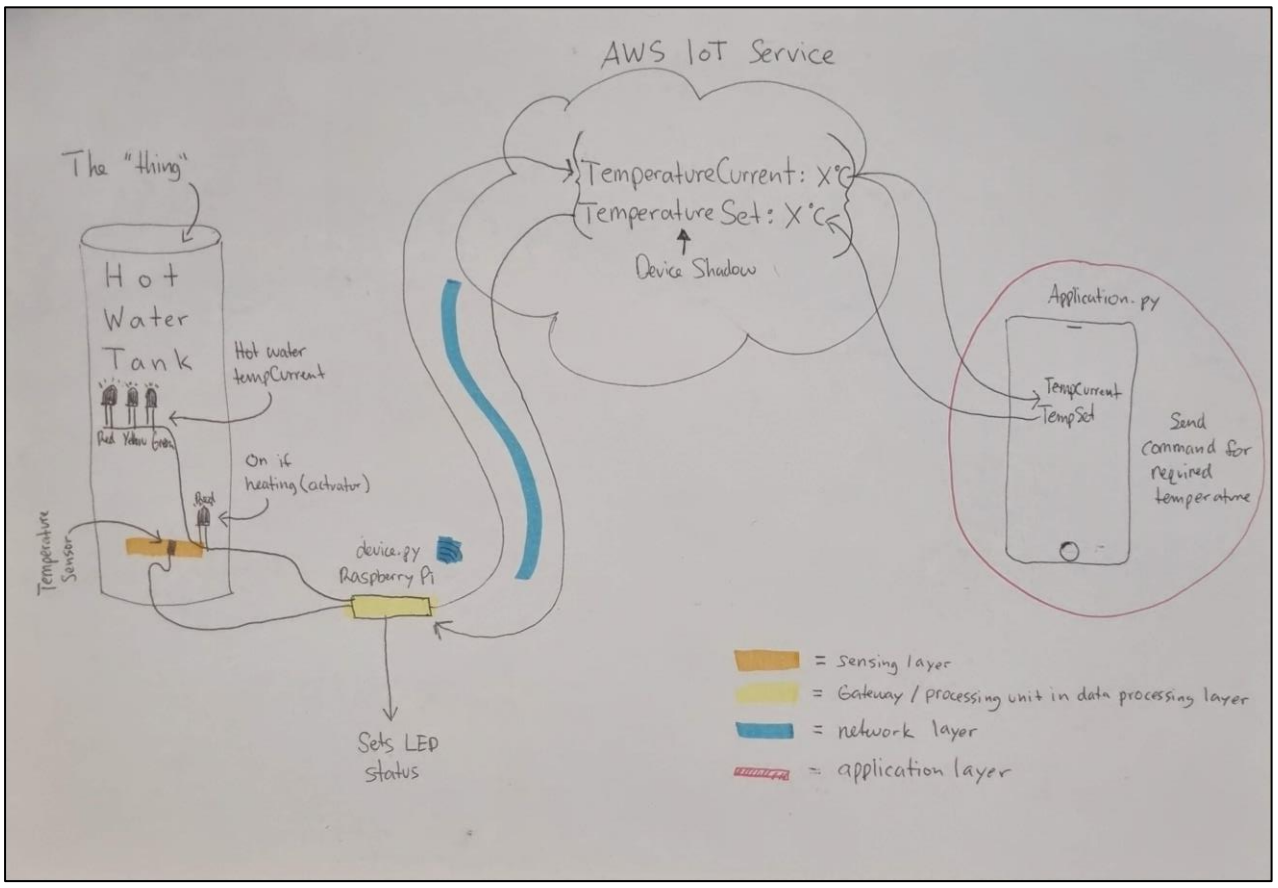

Figure 1: A Complete IoT Hot Water Tank System

Limitations in Implementation

Due to limitations in skill and time, several changes were made to the implementation in order to simulate an actual prototype.

Instead of connecting the Raspberry Pi to the heating element in the hot water tank, the singular red LED will be used to simulate the status of the heating element. Specifically, when the red LED is on, it simulates the heating element being on, while when the red LED is off, it simulates the heating element being off. It should be noted that in IoT architecture terms, the red LED/heating element would be known as the actuator in the IoT system.

Another change that was made was the setting of the actual temperatures used to trigger changes in the traffic light coloured LEDs. In order to simulate the changes in temperature quickly, the range for temperature change triggers was set as follows:

- Less than 25°C = red LED (not enough hot water)
- 25°C = yellow LED (just enough hot water)
- Greater than 25°C = green LED (sufficient amount of hot water)

In a properly implemented prototype, this temperature range would be closer to ~60°C which, according to Richards (2016), is hot enough to kill bacteria but not too hot to cause injury to users. The red LED in the traffic light coloured LEDs temperature setting (water too cold) would also depend on factors such as tank size, number of inhabitants, and whether the user wants enough hot water for a bath or for a shower.

Lastly, an application written in Python will be used on the Raspberry Pi to enable the user to set the current temperature of their hot water tank. This will be a simulation of the web application. Although this application is currently only built for use on the Raspberry Pi, it will still be sufficient for simulating the ability for the user to set the temperature of their hot water tank by sending a command “through the internet”.

Measurements

A measurement that was thought to be useful for this prototype was the amount of time it took for the updated temperature setting to be successfully reflected in the hot water tank. Essentially, this metric tells the user the performance of the IoT system as a whole. If the time taken to update the temperature starts to increase, it could mean that there is a bottleneck in the system somewhere. There could be a variety of reasons for this performance decrease, such as network issues, application issues, or even hardware issues with the device itself. Whatever the cause of the performance drop, this metric gives the user and/or technicians an indication that there is an issue somewhere that would need to be investigated further should the performance drop continuously arise.

Notes on the Video Demonstration

There were several instances when references were made to the device shadow in AWS IoT. The mouse can be seen pointing to the “desired” shadow state while it should have been pointing to the “reported” shadow state. Apologies for the error in the demonstration.

I have also added the functionality to store the metric data in a CSV file for later analysis which was not included in the video demonstration. The code for this function is in line 267 of “application.py” and it is called on line 152.

Figure 2 below is a simple graph showing the number of seconds it took to update “temperatureSet” taken from the CSV file created.

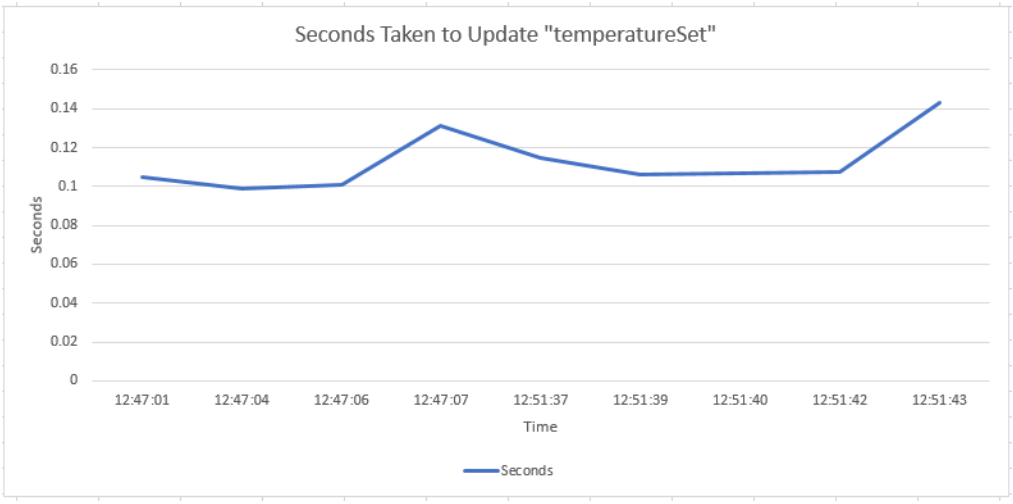

Figure 2

References

Richards, K. (2016) *Hot Water Temperature.* Premier Heating https://premierheating.net/2016/12/30/hot-water-temperature/