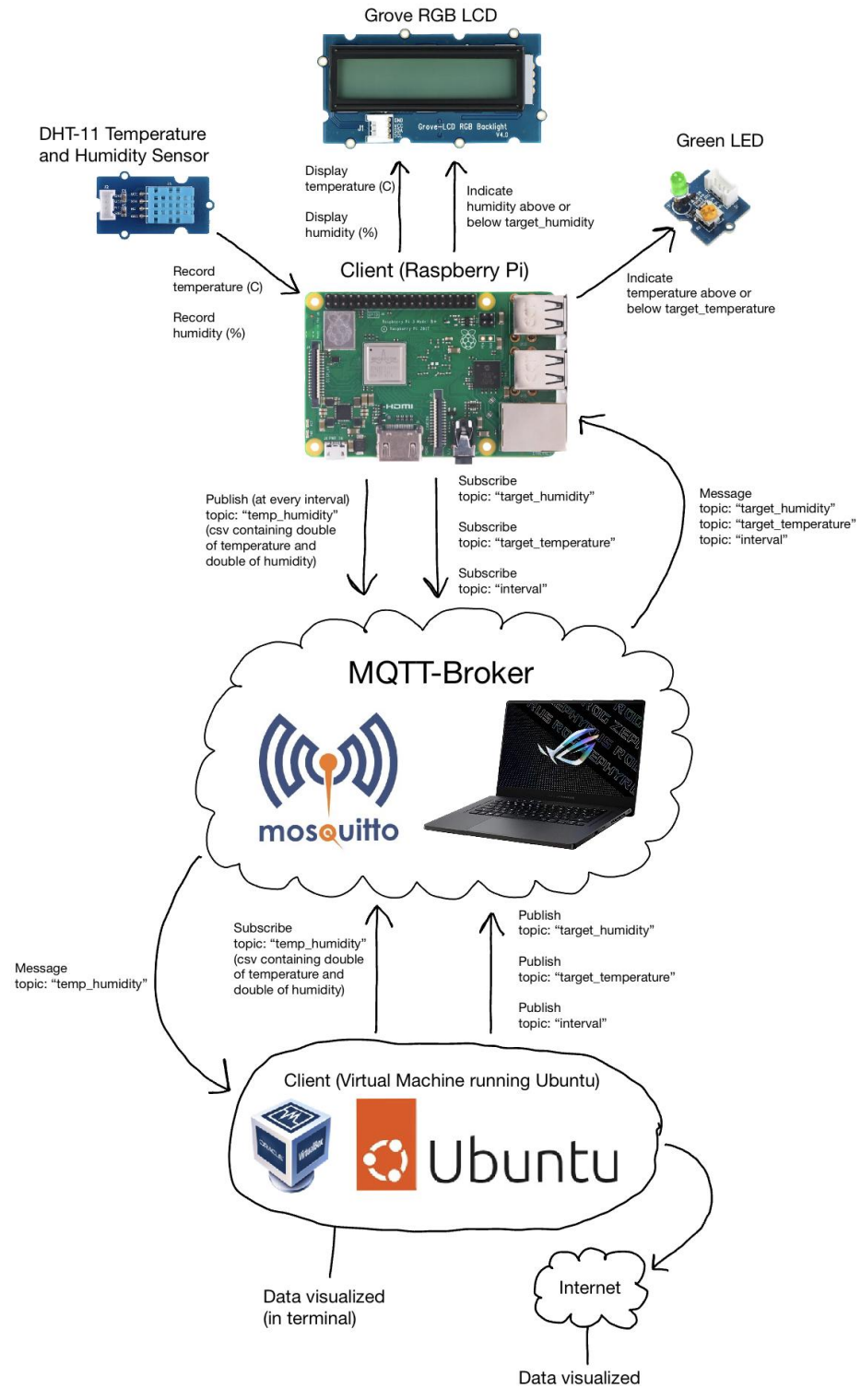
Ian Gravallese

Prof. Krishnamachari

EE-250 Distributed Systems for the Internet of Things

08 May 2023

Final Project Write Up



My project is an IoT system with three nodes (block diagram to the left). The center node is an MQTT-Broker which I hosted on my Windows laptop running an Eclipse Mosquitto Broker, though this MQTT broker could feasibly be hosted on any machine or server, so I have represented it as the cloud. The two other nodes are clients to this MQTT-Broker: my Raspberry Pi and my Virtual Machine running Ubuntu. Both of these client nodes connect to the MQTT-Broker using Eclipse Paho-MQTT Client software. On the Rpi side, a DHT-11 Temperature and humidity sensor records the temperature and humidity about every second, and those values are displayed on the LCD. There is a target temperature and a target humidity as well as a measurement interval (all with initial default values) which are updated

when the VM node publishes to each of their respective topics on the MQTT-Broker. When the current humidity is at or above the target humidity, the LCD is backlit green, and when it is not the LCD is backlit red. When the current temperature is at or above the target temperature the green LED is lit, and when it is not the LED is turned off. The interval determines how often the Rpi publishes the current temperature and humidity to the MQTT broker, in a CSV (comma separated values) string containing two doubles representing each. On the VM side these CSVs are decoded into their separate values and saved into arrays which record the temperature and humidity readings over time. I decided to send both values together as a CSV because I wanted to ensure that they were sent at the same moment, to avoid any mismatched recordings in the time domain and also because publishing as a CSV was fairly simple to encode and decode in python. This is because they can be sent as simple UTF-8 strings (as are all of the published topics to the MQTT-Broker) which can easily be manipulated by python string functions (I used .split(“,”) to separate the CSV into a list with both separate values). The other posts were also easily convertible between strings and floats with the python float() function. On the VM side the user interacts with a menu (simple user interface) that allows them to see the latest published temperature and humidity (as well as all other conditions); update the target humidity, the target temperature, and the interval; view the existing recorded data both as a dataframe and as a graph (the graph is both saved locally and also displayed online); and also reset the existing recorded data. There is input and exception handling for all of these options.

The biggest limitations I see to my project are that the data could be more effectively stored on the VM side for visualization (for example, all data must be erased when the user decides to update the measurement interval to ensure visualizations are accurate in the time domain), and also the visualizations could have been more extensive, perhaps with separate visualizations for each measurement type and/or interactive real-time visualizations of measurements on the VM side rather than just the Rpi side. These limitations mostly derived from my having limited time to invest in figuring out how to manipulate stored data on the VM side, and limited experience in front-end web development. If there’s anything this project made me want to learn more about, it would definitely be front-end web development, as it’s clear to me now the power that interesting and interactive front-end web development can have in enabling and enhancing any project.