CSC 453 (001) COURSE PROJECT

Weather Alert Station

SUBMISSION DATE: APRIL 25, 2021

GROUP 6: MASON ROWLAND, SOUMYA GADE, NICK RICHARDSON, CARTER THUNES, NAINIKA ELETY.

Contribution

Below is table of the project tasks and each team member's contribution.

The per student aggregate contribution is calculated using the sum of each individual's contribution to a component based on the component weightage.

Component	**Component Weightage**	**Mason Rowland contribution**	**Soumya Gade contribution**	**Nick Richardson contribution**	**Carter Thunes contribution**	**Nainika Elety contribution**
Project ideation/ Design	0.25	20%	20%	20%	20%	20%
Weather station hardware/code	0.15	100%	0%	0%	0%	0%
Device B code	0.15	0%	100%	0%	0%	0%
Device B frontend	0.15	0%	0%	100%	0%	0%
Broker/Wireframes	0.05	0%	0%	0%	100%	0%
Debugging/Assist	0.05	0%	0%	0%	0%	100%
Presentation slides	0.1	0%	0%	0%	100%	0%
Report writing	0.1	0%	0%	0%	0%	100%
Per student aggregate contribution		20%	20%	20%	20%	20%

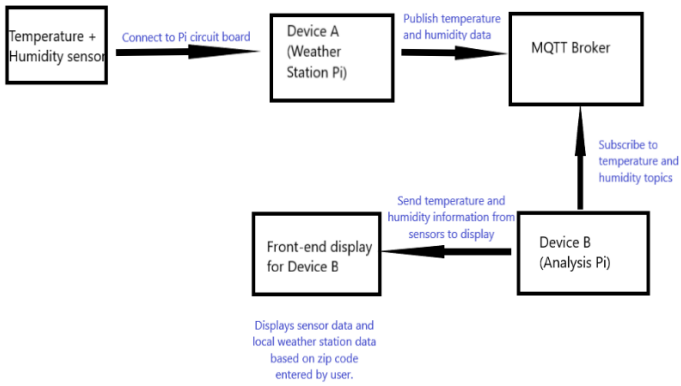
Introduction

For our course project, we decided to create a Weather Alert Station for an indoor home or area. The Weather Alert Station provides the user of the indoor climate (based on temperature and humidity) and how this climate compares to the local weather station data. The objective of the project is to provide the user with a system that updates the temperature and humidity data regularly, with descriptions about the weather.

Many times, the weather in the local weather station may not reflect what the climate is in someone's home. Therefore, this system allows people to have accurate data on how the climate is in their home, and how this climate compares to the data projected by local weather stations. Moreover, our system provides general descriptions about the climate based on certain temperature and humidity thresholds. This allows users to be aware of the weather conditions and prepare for different weather scenarios.

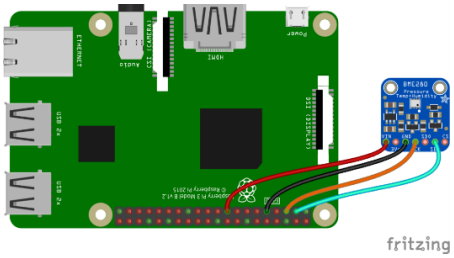
Design

Below is a block-diagram on how our weather system works. This design helped us decide the functionality of each device or file used.



Block Diagram

For the first block in the diagram, shows how we planned to connect our temperature+humidity sensor to our weather system. We decided to connect a temperature and humidity sensor to a Raspberry Pi device (Device A) to collect the temperature and humidity data. The sensor design below shows how the temperature+humidity sensor was connected to the Raspberry Pi device circuit board.

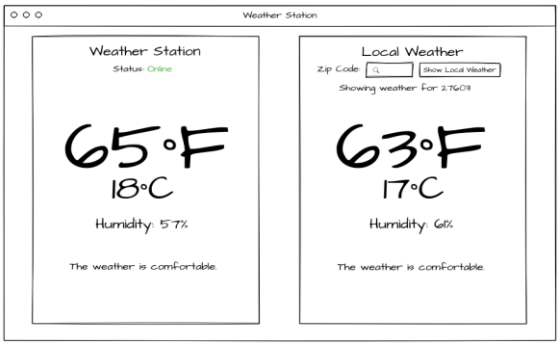


Sensor diagram

The next block shows how the Device A collects the sensor information, and then connects to the broker to publish frequent updates of temperature and humidity values. Device A only chooses to publish these values to the broker if there is a change in temperature or humidity values.

The broker block oversees the transmission of data from Device A (which collects the sensor data) and Device B (which analyzes the sensor data). When the broker is online, Device B will subscribe to the temperature and humidity topics to analyze the data.

Device B oversees analyzing the data to give weather descriptions on the weather based on temperature and humidity thresholds. After this, Device B sends the weather descriptions, humidity, and temperature data over to the front-end app to display the information to the user. The front-end app displays the weather information and gets the local weather station information based on the zip code entered by the user. Below is a wireframe design of the front-end.



Device B Wireframe

Implementation

For our implementation, we bought a sensor which had temperature and humidity detecting abilities to make processes easier. Moreover, we decided to connect the sensors to a Raspberry Pi device to be able to use a MQTT broker easily. The reason we implemented our system with an MQTT protocol is due to its ability to receive messages even when the broker is offline, and the feasibility of the publish/subscribe methodology.

Our sensors were connected to the Raspberry Pi circuit board as seen in the design section. We decided to collect the sensor data on Device A using a python file. We decided to use python as our main programming due to the mqtt packages that python offers, which are easy to understand.

In our Device A python file (weatherstation.py), Device A connects to the broker and subscribes to topics “Status/WeatherStation”, “Temperature/F”, “Temperature/C”, and “Humidity”. Device A converts the temperature data into Celsius and Fahrenheit and decides to publish the temperature values to the broker (topics “Temperature/C” and “Temperature/F”) if there is a change greater than 0.2 of the previous published temperature value.

Similarly, for Humidity, there needs to be a change greater than 0.2 for the value to be published to the "Humidity" topic. Device A also publishes a status of "Online" to the "Status/WeatherStation" topic to show that this topic is online.

Device A was implemented in Mason Rowland's location which was different from Device B location.

Thereafter, Device B was implemented in a python file called device_b.py. The device subscribed to the following topics from the broker: "Status/WeatherStation", "Temperature/F", "Temperature/C", and "Humidity". Device B then analyzed the data from these topics and assigned a weather message based on certain thresholds.

If the temperature was greater than or equal to 78 Fahrenheit, the message was set to "comfortable". If the temperature was greater than 75 Fahrenheit, and the humidity was greater than or equal to 55, the message was set to "muggy". Furthermore, if the temperature was less than or equal to 60 Fahrenheit, the message was set to "chilly". Finally, if the temperature was less than or equal to 45 Fahrenheit, the message was set to "cold". Thereafter, the Device B sent this weather information out as a json file, which will be used by the front-end app.

Finally, the front-end was implemented using javascript through an app.js file. The front-end file used various API calls to get the data it needed, including the json data from Device B. The front-end then displayed the weather information as shown in the wireframes, based on the json object. We also implemented a way for the user to see the local weather station information, by having an auto-zip code generation button which receives the zip code of the user based on the user's latitude and longitude coordinates to get the postal code. After getting the zip code, the front-end finds the local weather station data for that zip code, determines the weather message for the weather station based on the thresholds mentioned before. Therefore, the local weather station data is reported like how the sensor data is reported in the wireframe.

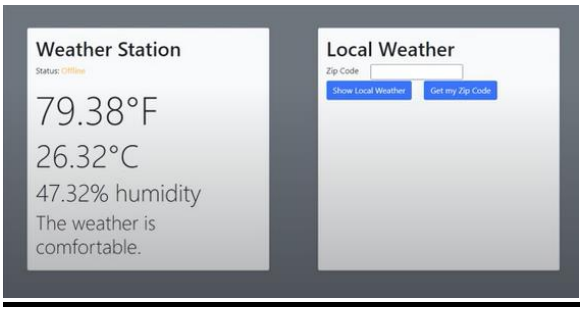
The sensor was bought from the following website:

https://www.amazon.com/BME280-Environmental-Sensor-Temperature-barometric/dp/B07GZCZFVG/ref=sr_1_2?dchild=1&keywords=waveshare+environmental+temperature+barometric+detection&qid=1619300250&sr=8-2

Results and Discussion

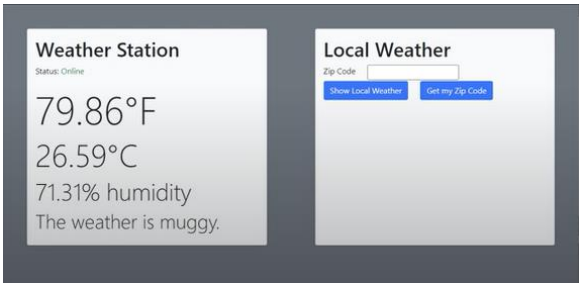
During our demo, we tried to test different temperature and humidity situations to see if the weather system would update appropriately with the weather descriptions as well. Our experimental

results test the thresholds required for weather to be “comfortable”, “muggy”, “chilly”, and comparing the data to the local weather from a weather station.



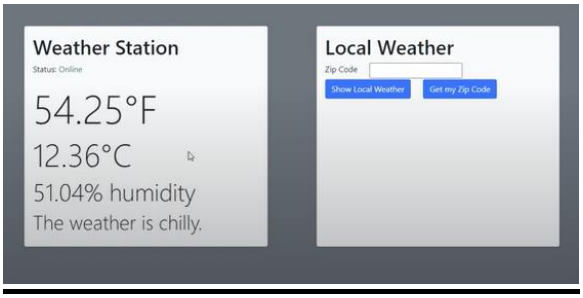
Comfortable Weather Result

The image above shows our result from the demo when the broker was connected, and the front-end displayed the conditions of the current climate at that time. The temperature and humidity values were in the “comfortable” description and corresponded to the threshold we applied for a “comfortable” weather.



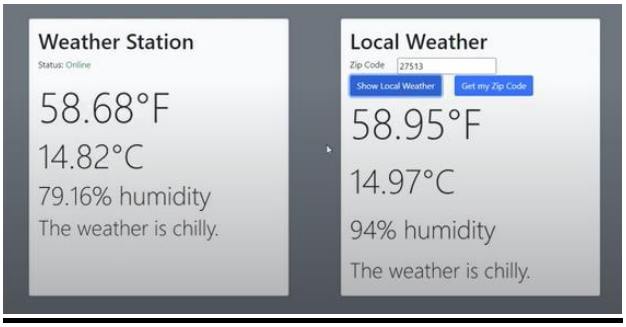
Muggy Weather Result

This image shows the result for muggier weather, which meant a high temperature and humidity was required. To produce this result, we made the climate more humid to test this weather description.



Chilly Weather Result

The image above demonstrated the result for “chilly” weather, which meant weather that we lower than 60 degrees Fahrenheit. To test this, we put ice near the temperature sensor to lower the temperature of the climate.



Local Weather Result

Finally, the image above shows the result of comparing the sensor data with the local weather data. To gather the local weather data, we clicked the “Get my Zip Code” button which filled in the zip code for us and displayed the weather information for that zip code area. As the result shows, the local weather and the sensor weather seem to have similar temperature, but the sensor weather seems to have a

lower humidity. This observation is interesting, as it shows that there is a difference in data for indoor and outdoor weather, which is useful to compare.

Nevertheless, the demo results show that our weather alert system works under different weather scenarios and is useful to compare the indoor climate with the local weather station data as well.

Related Work and References

Robbins, Chris. July 21st, 2016. “Heat Index and Calculator & Charts”. https://www.iweathernet.com/educational/heat-index-calculator-and-conversion-table

Sensor bought from: https://www.amazon.com/BME280-Environmental-Sensor-Temperature-barometric/dp/B07GZCZFVG/ref=sr_1_2?dchild=1&keywords=waveshare+environmental+temperature+barometric+detection&qid=1619300250&sr=8-2