Lizar 4020 Model 1. Rev A

CPE 4020 Device Networks
12/9/2020

John Herring
Kwame Akuffo
Michael Young
Andrew Bluhm

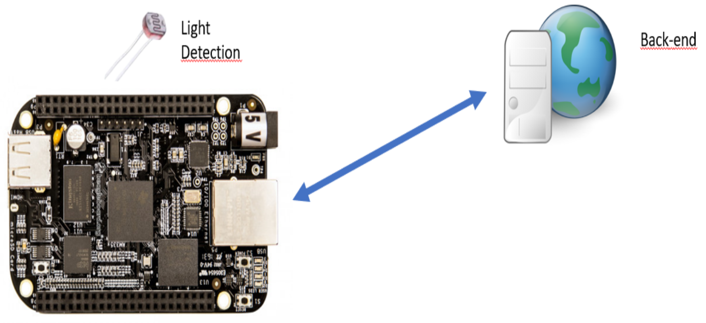
Links//
Task 1: https://www.youtube.com/watch?v=o9U9eGrNY0Q&ab_channel=AndrewBluhm

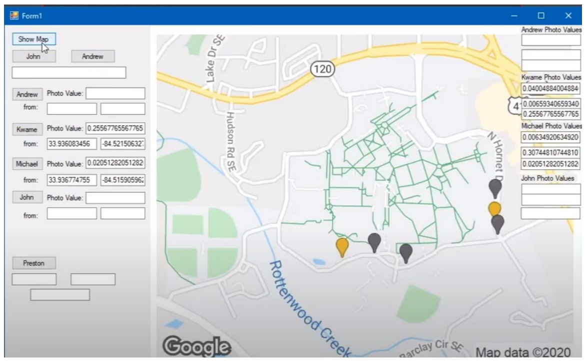
Task 2:
https://www.youtube.com/watch?v=06ikCQpJx3o&feature=youtu.be&ab_channel=AndrewBluhm

Final Presentation: https://www.youtube.com/watch?v=UjIX0vwLVgU&ab_channel=AndrewBluhm

<u>Summary of System</u>

Task 1 Use-Case:
IoT is all about creating value. We demonstrate this by building an IoT system which provides customers with remotely accessible sunrise and sunset information, specifically light data. Our IoT solutions company has just been contacted by Kennesaw State University's campus safety team. There have been a few complaints about how dark certain areas on campus get at night, and KSU has come to us wondering if we can help solve this issue. KSU's campus safety team is a huge proponent of energy efficiency but is also looking for a cost-effective solution. For these reasons, this team does not want to simply install street lights everywhere, but instead, install street lights where they are **actually** needed. We propose a solution to deploy our sensors in various parts of campus in order to determine which areas need better lighting. Along with this package, we provide KSU's safety team with a backend dashboard built in C#.NET that displays the light condition of each sensor location in real time. A picture of a similar dashboard is shown below. KSU's campus safety team is happy, because not only can they use our system for this project, but they can continue to use it to stay up to date on which street lights aren't working.





Andrew Photo Value:
from:
Kwame Photo Value: 0.25567765567765
from: 33.936083456 -84.52150632
Michael Photo Value: 0.02051282051282
from: 33.936774755 -84.51590596
John Photo Value:
from:

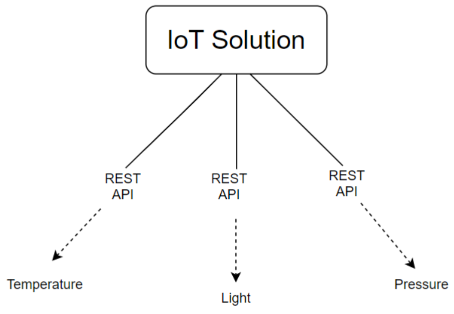
Andrew Photo Values

Kwame Photo Values
0.04004884004884
0.00659340659340
0.25567765567765

Michael Photo Values
0.00634920634920
0.30744810744810
0.02051282051282

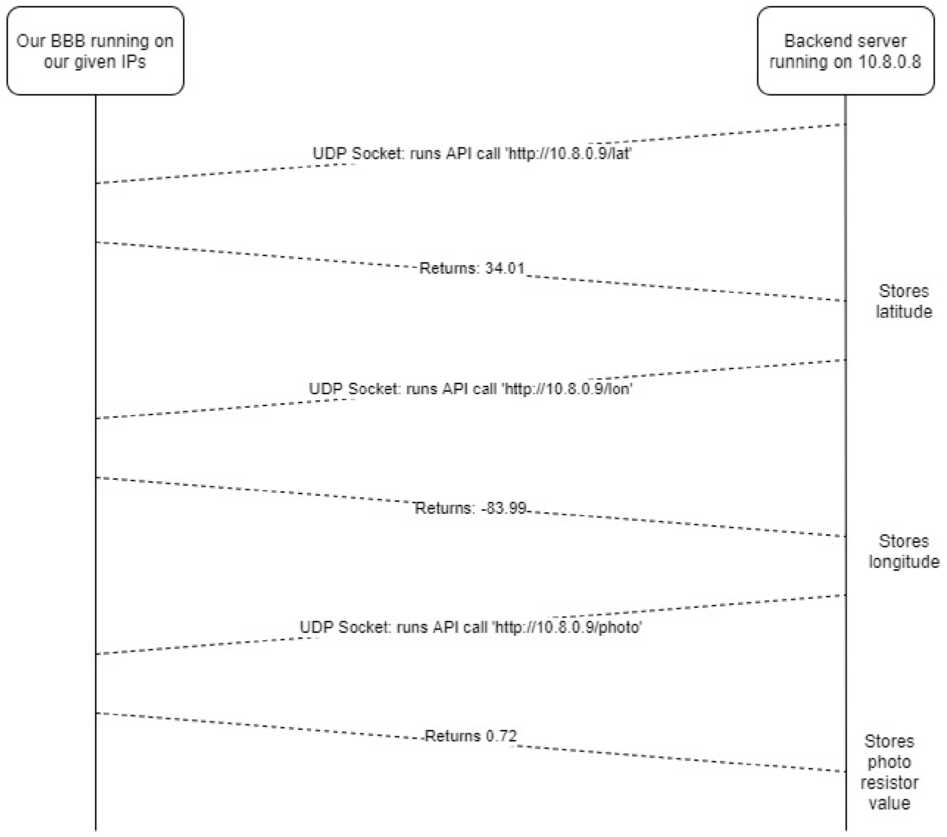
John Photo Values

Task 2 Use-Case:
One of KSU's sponsors, a food service carrier, has heard great things about our team and is in need of a solution. This company just lost thousands of dollars because a driver did not notice an opened door on the refrigerator truck, therefore causing a lot of food waste and unhappy customers. Said company would like to avoid this from happening again and is looking for a solution for all trucks. As an IoT solutions company who specializes in building IoT light systems, we would like to propose the use of our sensors, but we know that it will not be enough. We hold a meeting with our colleagues who have great, deployable IoT systems, specifically for pressure and temperature, and discover a solution. This value-driven solution integrates our use case with those of our colleagues through the use of API calls and is as follows: If a subscriber contacts our broker and receives a message back, stating that the photoresistor detects sunlight, the temperature sensor is above a certain temperature, and the pressure sensor is also below a specified temperature, then a door is opened while the vehicle is moving and must be shut.



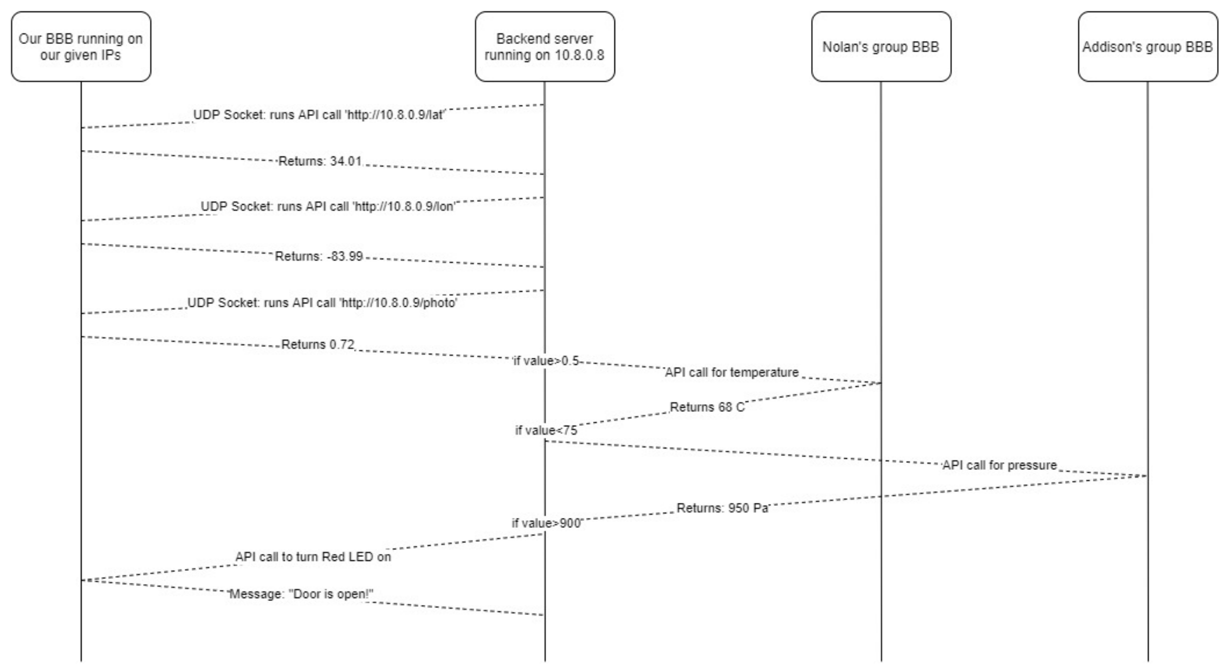
Publisher/ Subscriber protocols and architecture:

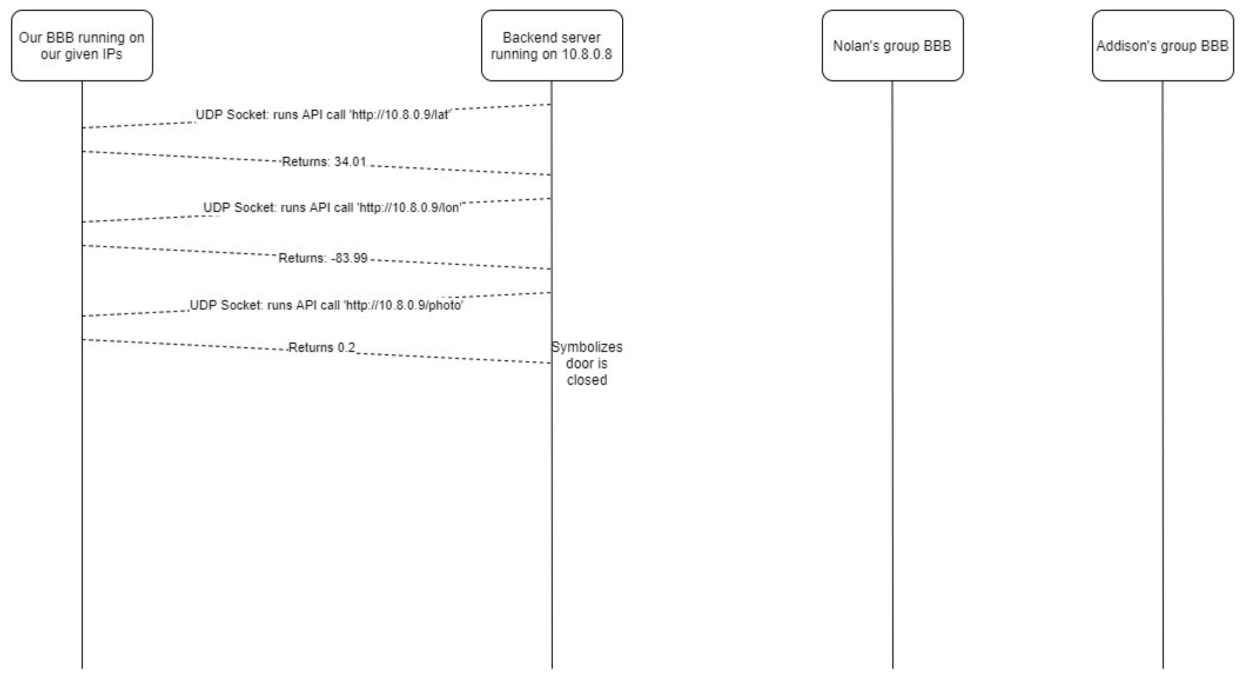
For task one, we use a pub/sub protocol as described in the waterfall plot below:

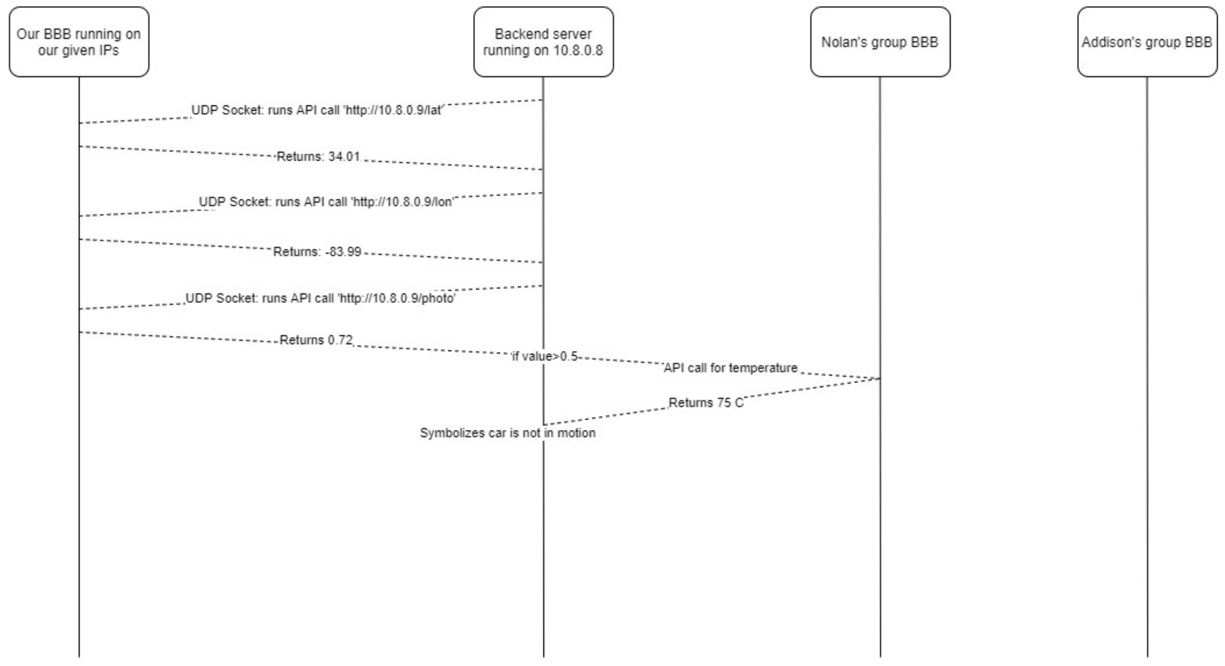

Our BBB running on our given IPs
Backend server running on 10.8.0.8
UDP Socket: runs API call 'http://10.8.0.9/lat'
Returns: 34.01
Stores latitude
UDP Socket: runs API call 'http://10.8.0.9/lon'
Returns: -83.99
Stores longitude
UDP Socket: runs API call 'http://10.8.0.9/photo'
Returns 0.72
Stores photo resistor value

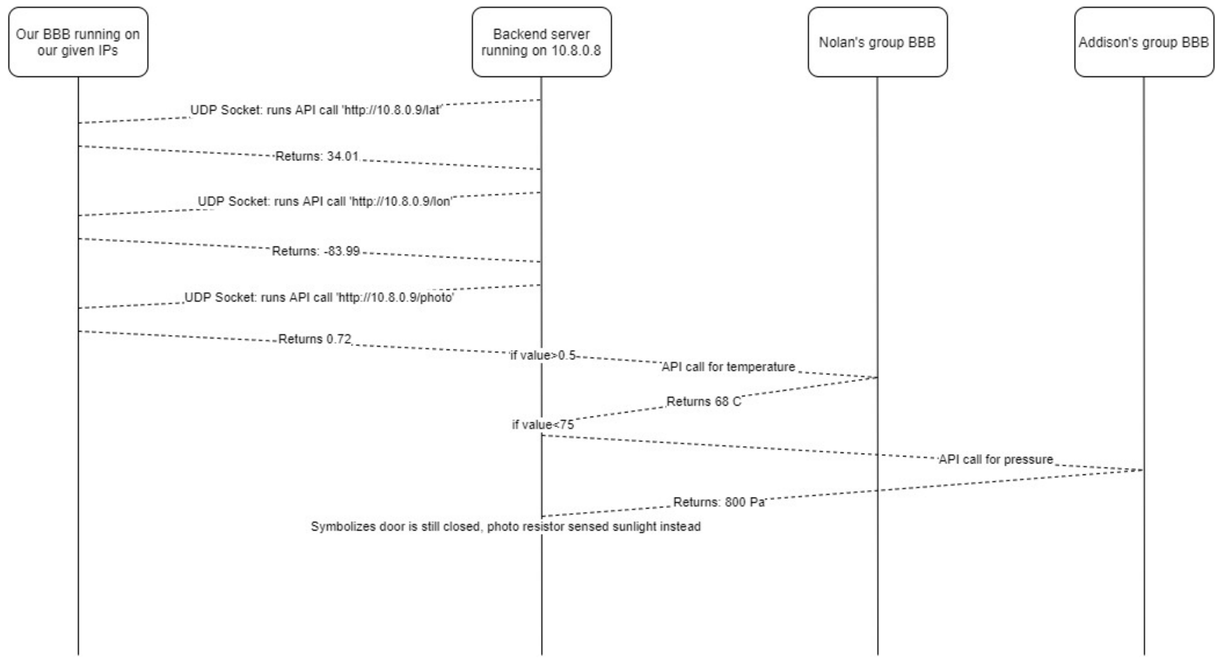
We set our server up so that when we start our API calls, it calls the Beaglebone Black 3 separate times, once to get the latitude, once to get the longitude, and once to get the photo resistor value. We set it up this way to make it look nicer on the server end. This also made it easier when working with the other groups. All of these calls are ties and connected to one timer that updates each of their respected text boxes respectively. In this case, the subscribers are each of our Beaglebone Blacks getting GPS data, and also the photoresistor data to determine if an area on campus is lit up or not. The publisher is the server we created to reach out to each of our devices. The call is the message run through the class VPN (message broker), and then the devices receive this data and send it back to the server. It maintains all of this through separate UDP sockets.

Task 2 is a bit more complicated. It still uses the same idea as task one, but now we need each and every group's sensor to be online for our use case. Here is every scenario that could possibly happen:


Our BBB running on our given IPs
Backend server running on 10.8.0.8
Nolan's group BBB
Addison's group BBB
UDP Socket: runs API call 'http://10.8.0.9/lat'
Returns: 34.01
UDP Socket: runs API call 'http://10.8.0.9/lon'
Returns: -83.99
UDP Socket: runs API call 'http://10.8.0.9/photo'
Returns 0.72
if value>0.5
API call for temperature
Returns 68 C
if value<75
API call for pressure
Returns: 950 Pa
if value>900
API call to turn Red LED on
Message: "Door is open!"


Our BBB running on our given IPs
Backend server running on 10.8.0.8
Nolan's group BBB
Addison's group BBB
UDP Socket: runs API call 'http://10.8.0.9/lat'
Returns: 34.01
UDP Socket: runs API call 'http://10.8.0.9/lon'
Returns: -83.99
UDP Socket: runs API call 'http://10.8.0.9/photo'
Returns 0.2
Symbolizes door is closed


Our BBB running on our given IPs
Backend server running on 10.8.0.8
Nolan's group BBB
Addison's group BBB
UDP Socket: runs API call 'http://10.8.0.9/lat'
Returns: 34.01
UDP Socket: runs API call 'http://10.8.0.9/lon'
Returns: -83.99
UDP Socket: runs API call 'http://10.8.0.9/photo'
Returns 0.72
if value>0.5
API call for temperature
Returns 75 C
Symbolizes car is not in motion


Our BBB running on our given IPs
Backend server running on 10.8.0.8
Nolan's group BBB
Addison's group BBB
UDP Socket: runs API call 'http://10.8.0.9/lat'
Returns: 34.01
UDP Socket: runs API call 'http://10.8.0.9/lon'
Returns: -83.99
UDP Socket: runs API call 'http://10.8.0.9/photo'
Returns 0.72
if value>0.5
API call for temperature
Returns 68 C
if value<75
API call for pressure
Returns: 800 Pa
Symbolizes door is still closed, photo resistor sensed sunlight instead

One element not stated in these waterfall plots, is that our group also gets their latitude and longitude data points from each sensor's data we collect. This help gives some visualization to the project and shows where each Beaglebone Black is.

API Documentation:

/Andrew - Used to test connection to BBB to make sure it was established

Returns: “Hello Andrew!”

This api call was used to test the connection for Andrew’s BBB to the back-end server by using a call and response for Task 1.

/John - Used to test connection to BBB to make sure it was established

Returns: “Hello John!”

This api call was used to test the connection for Andrew’s BBB to the back-end server by using a call and response for Task 1. We used two different buttons to make sure we were able to change the textbox on our backend server.

/lat - Reads Latitude value.

Returns Latitude as: “XX.XXXXXXXXX”

This api call was used to get the latitude from the gps that was collected from the gpsd modules/library. This call was used by the back-end server to plot the location of the photoresistor data received.

/lon - Reads Longitude value.

Returns Latitude as: “XX.XXXXXXXXX”

This api call was used to get the longitude from the gps that was collected from the gpsd modules/library. This call was used by the back-end server to plot the location of the photoresistor data received.

/photo - Reads Photoresistor value between 1 and 0.

Returns Light Value as: “X.XXXXXXXXXXXXXX”

This api call is used to get the photoresistor value. This value ranges between 0 and 1, 0 meaning there is no light, and 1 meaning the photoresistor is being fully illuminated with light.

/greenLED - Changes LED to Green

Returns: “Door is closed!”

This api call is used to change the color of the LED to green when the photoresistor does not detect light, the pressure sensor reading is below 950, and temperature sensor reading is above 70 degrees.

/redLED - Changes LED to Red when

Returns: “Door is open!”

This api call is used to change the color of the LED to red when the photoresistor detects light, the pressure sensor reading is above 950, and temperature sensor reading is below 70 degrees.

Security Measures:

There are no security measures implemented within our system. We wanted to make sure that all the groups involved could easily have access to our system without needing a key, token, or any other decryption device to access the system. There is a VPN that is used throughout our project, but this can prove to be very vulnerable in itself. The VPN is susceptible to being spoofed which would allow an attacker to take control of the network and the devices connected to this network.

Confidentiality:

Confidentiality is the concept that the information being transmitted is protected from being accessed by an unauthorized source. While this system is in use, the confidentiality will be maintained. Since the user will be inside a moving vehicle for the majority of the time while the system is operating the Wi-Fi network, we assume that the data being sent between the device(s) and the back-end system will be encrypted preventing others from intruding on the data that is being transmitted.

Authentication:

Authentication is making sure that both the sender and receiver confirm each other's identity to be real before transmitting information. This is where the problem lies within our system, there is no way to confirm authentication. The way our system is constructed, neither side confirms their identity. Both sides recognize the other side, but never make sure it is the correct device in use. As stated before, we wanted to make sure every group was able to access our data, so not having this policy in place allows for the ease of transferring data. This also leaves the door open for 'spoof' attacks, which can occur on the back-end side allowing for the data to never be collected properly, preventing the response from the system from occurring in these instances.

Message Integrity:

Message integrity is the concept of ensuring data does not get tampered with on the way to its destination. Within our system there is no way to ensure message integrity between the device(s) and the back-end API. Since we have no encryption mechanisms implemented or a hash function included, there could be a "man-in-the-middle" attack that could change the data that is being received by the API. Our data is sent as a plaintext string, to keep it simple for ourselves and others, which allows an outside user to easily change what value is received by the API.

Access and Availability:

Access and availability is the concept of making sure that the data being transmitted can be viewed and used when needed by a system. Since we wanted to make sure the other groups, along with all of our BBBs, were able to easily access and use our data we were collecting, we also left the door open for other users to access and use our data. We have no way of denying any outside sources implemented within our system. This allows for the possibility of multiple users to log in and attempt to overload our system with ease. Since all processes are single-threaded, it makes our system especially vulnerable to DoS attacks which could prevent us from collecting the data necessary to implement our system.

Summary and Challenges:

Our project had its complications as all projects do. The majority of our problems came from things that were not discussed in class or working with other groups. Fortunately we were able to overcome these challenges by collaboration with each other and connecting with other groups when necessary to hold each other accountable when working on this project. There was a lot to be learned from this project and we feel as though it will benefit us in the grand scheme of things within school and outside of school.

One of the key problems we ran into was making sure all the BBBs worked properly on the network we had set up for our project. Each BBB seemed to give us another problem that needed to be addressed when we went through the steps to make sure we stayed on the same page throughout the whole project and everyone understood what was happening. One of the common issues we were encountering as individuals, was connecting our BBB to the back-end server we set up throughout the project. We were able to overcome this by resetting our BeagleBones and attempting to reconnect again which usually worked, if that did not work then we would have to go through and make sure our BBBs had the Wi-Fi set up properly and we were able to ping each other's BBB before trying to connect again.

Another problem we ran into was when we finally got to driving around campus with our BBBs connected to the back-end server, some of our BeagleBones would cause the API and server to crash when trying to collect data which would cause us to have to restart the program over again and recollect any data that had been previously collected before the crash. It was interesting because all of our devices seemed to work when we were next to the back-end server, but as soon as we got into the vehicles to drive around campus we would have problems connecting back to the API. Most of the time, this was easily resolvable by resetting the BBB and attempting to connect again. If this would not work we would have to make sure all the files were correct on the BBB and try again, eventually we got them all to work and not crash the server so we could collect our data with our photoresistors through the API running on the BBB.

Outside of some other minor issues, another big problem was communicating with the other groups and working together with each other's busy schedules. This was also a problem because along with the teams having different schedules, we had to make sure everyone had their project up and running for us to be able to connect and communicate with each other. This caused us to have to wait to finish our task 2 and make sure everything was working and connected properly. Once all the groups were able to get up and running, we were able to troubleshoot with one another to clear up any errors and make individual calls for each group that allowed us to access each other's data for our system. We were able to get this done in a reasonable time that allowed us to complete all other tasks that were necessary to submit our project.

When it comes to code, the biggest problem we ran into was converting our code from python to javascript for our GPS and getting our API to read our latitude and longitude correctly. At first, we had to figure out how to get the data from the GPS to be collected and read in. This was the tough part since it was us pressing a button to "activate" the GPS and send us a response. Unfortunately, the GPS was very sensitive and would not pick up the data correctly due to our location when we were setting up the device. Once, we were able to get the code for the calling of the data from the GPS into javascript, we had to make sure all of our BeagleBones would pick up the data with latitude and longitude being correct. This was done by making sure we were all in the same area when running the code to make sure we were getting similar responses and that they showed up as latitude and longitude values. Once we accomplished this we were able to set up a backend server with buttons that were connected to our BBBs to call the values and display them on a map that we imported into our RESTful API.