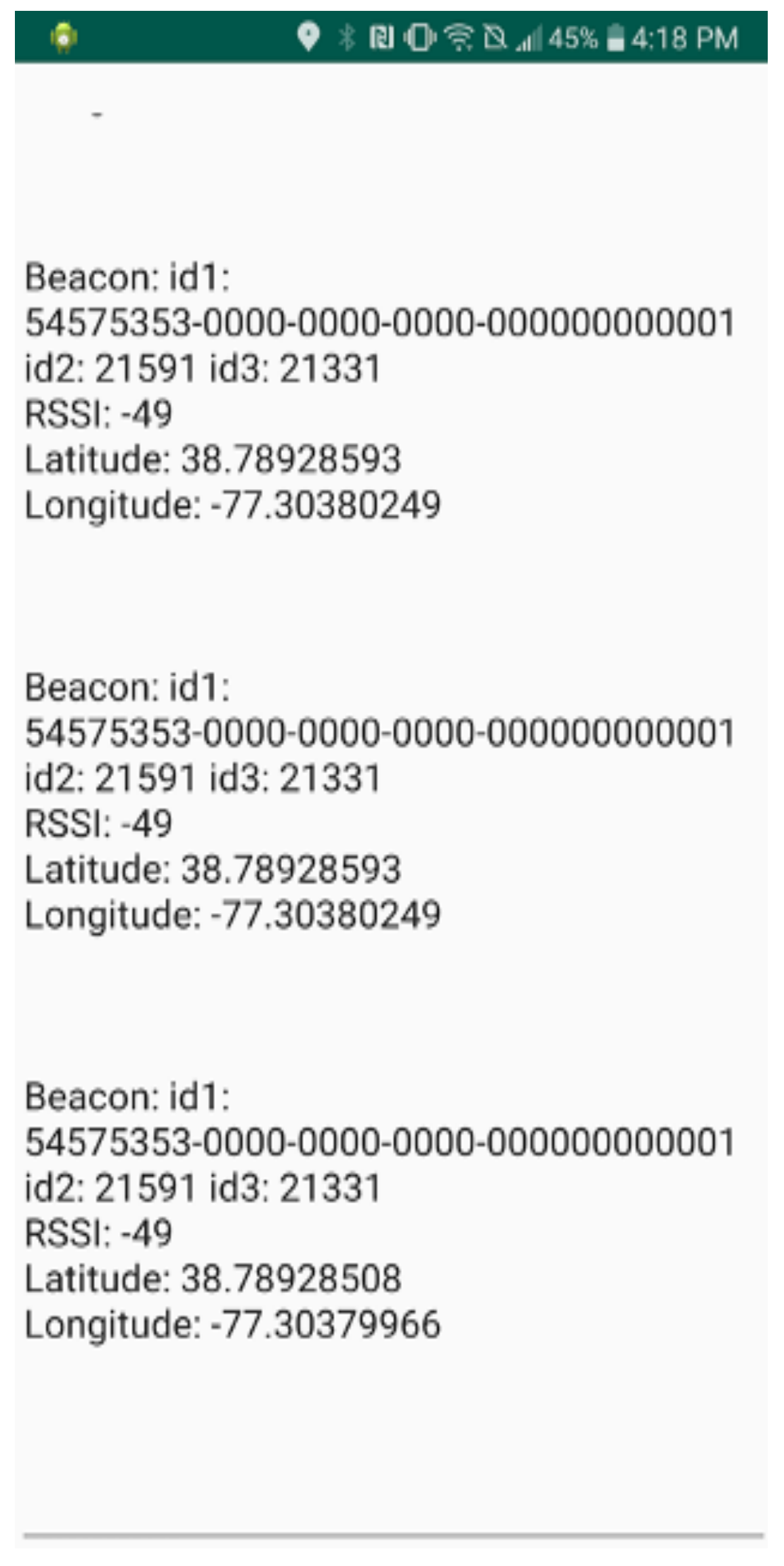
Enhanced Amber Alerting System

Ryan Chan, BS; Alexander Danchak, BS; Richa Malhotra, BS
George Mason University, Cyber Security Engineering BS

Introduction

GPS is the first thing that comes to mind if someone were to ask how an individual's location can be tracked. This is because GPS trackers are commonly used today, however GPS has two main limitations. The first limitation is that GPS loses its signal indoors, or due to obstructions in the sky. With limited signal reception, an individual or pet could not be tracked in areas that GPS cannot be received in. Lost pets will often seek shelter with a roof or canopy, which a GPS tracker would lose signal under. Additionally, individuals that are lost or kidnapped may be indoors, meaning that they couldn't be located through GPS. The second limitation of GPS is that it heavily consumes power. This power constraint will cause trackers to drain battery very quickly. Individuals or pets that are lost for a longer span of time than the battery life of a GPS tracker, will no longer be able to be tracked. These two limitations make GPS not a viable solution for quickly locating lost individuals or pets.

This poster discusses the steps our team took to tackle this problem using TWSS's idea of utilizing crowdsourcing and Bluetooth. Crowdsourcing is the practice of obtaining information for a specific task by enlisting the services of a large number of people. Applying the idea of crowdsourcing to tracking, TWSS wants to create a network of smartphone users. These users volunteer to download our applications to help find lost individuals or pets.



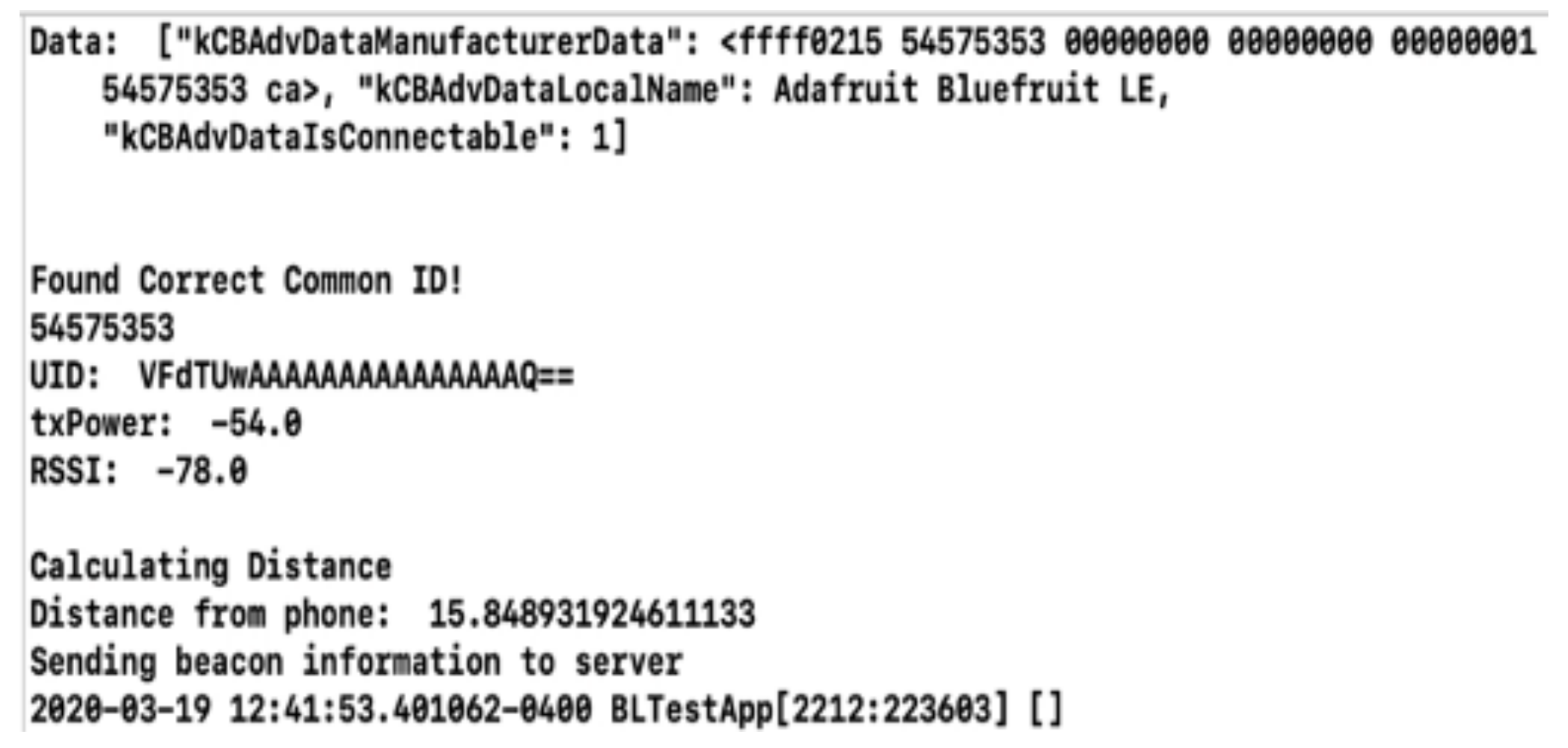
```
4:18 PM
-

Beacon: id1:
54575353-0000-0000-0000-000000000001
id2: 21591 id3: 21331
RSSI: -49
Latitude: 38.78928593
Longitude: -77.30380249

Beacon: id1:
54575353-0000-0000-0000-000000000001
id2: 21591 id3: 21331
RSSI: -49
Latitude: 38.78928593
Longitude: -77.30380249

Beacon: id1:
54575353-0000-0000-0000-000000000001
id2: 21591 id3: 21331
RSSI: -49
Latitude: 38.78928508
Longitude: -77.30379966
```

Figure 1: Android Parasite Debug



```
Data:  ["kCBAdvDataManufacturerData": <ffff0215 54575353 00000000 00000000 00000001
    54575353 ca>, "kCBAdvDataLocalName": Adafruit Bluefruit LE,
    "kCBAdvDataIsConnectable": 1]


Found Correct Common ID!
54575353
UID:  VFdTUwAAAAAAAAAAAAAAAQ==
txPower:  -54.0
RSSI:  -78.0

Calculating Distance
Distance from phone:  15.848931924611133
Sending beacon information to server
2020-03-19 12:41:53.401062-0400 BLTestApp[2212:223603] []
```

Figure 2: iOS Parasite Debug

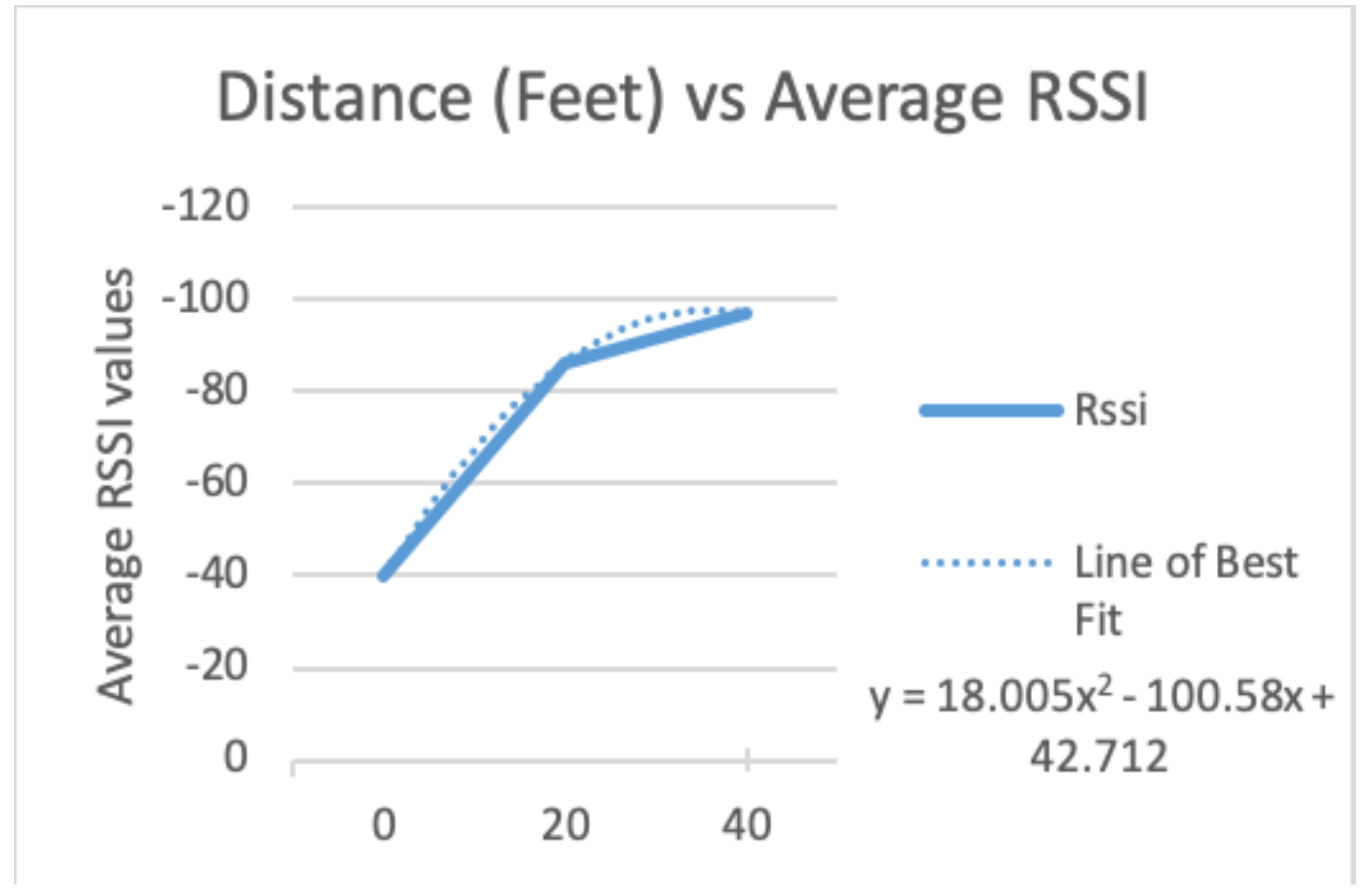


Figure 3: RSSI to Distance mapping 40 feet

Implementation

The tracking application system is made out of the following components:

Beacon:

- The physical beacon is an Adafruit Feather 32u4 Bluefruit LE, this is an Arduino microcontroller compatible with Bluetooth Low Energy. The beacon's firmware was developed with Arduino which is basically a C/C++ language. The beacon firmware's only function is to emit Bluetooth advertising beacon packets to nearby smartphones.

Parasite application (Android and iOS version):

- The background application made for Android and iOS is able to monitor and obtain RSSI values for nearby beacons. The Android parasite application needs the user to manually start the ranging activity that allows collection of the RSSI values. The iOS parasite application successfully connected to the API and was successful in executing a POST request. The data that was POSTed to the API was not able to be pulled by the server so data was manually copied into the server to further the implementation of the prototype.

Parent Application (iOS only version):

- This application is designed to acts as a user interface that connects the user with the application system. It is responsible for user account creation and login, pairing a beacon with a user account, and displaying the location of a user's lost beacon through a GUI map.

API:

- The custom API that was created to handle the connection between the applications and the server, was implemented as a SailsMVC application. SailsMVC is a JSON based API framework that allows an API to be built quickly. The API was hosted locally on a laptop.

Server:

- The server is hosted locally on a laptop. The server contains a database in which it has tables holding datastructures needed for system . The data structures contain users account information, sensor data, and geolocation data from the parasite application. Furthermore, the server runs a query of an implemented trilateralization geolocation algorithm written in Python. Trilateration is defined as the process of determining absolute or relative locations of points by measurement of distances, using the geometry of circles The connection to grab data from the tables with the implemented data structures was not successful. However, another query is implemented within the server, written in SQL, runs a geolocation algorithm with only 2 base stations and is able to successfully grab the necessary data from the server to perform an accurate location. The server also runs another query, written in SQL, that pulls the timestamp in which the sensor was last seen from each user account. Once the data from the parasite application is turned into coordinates that are manually copied to the parent application to see the calculated location of the beacon.

Conclusion and Future Implementations

Overall, the Enhanced Amber Alerting System that was developed will be able to help efficiently locate individuals or pets in places where GPS signals are not as prominent. This state-of-the-art technology that utilizes Bluetooth crowdsourcing to aid in tracking lost individuals or pets will revolutionize the way individuals are being tracked currently with GPS. However, there are still some implementations that can help improve the Enhanced Amber Alerting System. For example, a feature that wasn't implemented due to time restrictions was the geofence. The geofence was to be used as another way of determining if the beacon was lost. The owner of the beacon would define a set geofenced area around the beacon. If the server were to determine that the beacon has stepped outside of the geofenced area, the owner's parent application would be alerted. The parent application would then display the location of the beacon in order for a safe return home. Another feature that could improve the range of the parasite applications is using Bluetooth 5.0. This technology has more speed and range than BLE. Implementing Bluetooth 5.0 would allow the parasite applications to detect lost persons faster and from a further away distance.

Verification And Validation

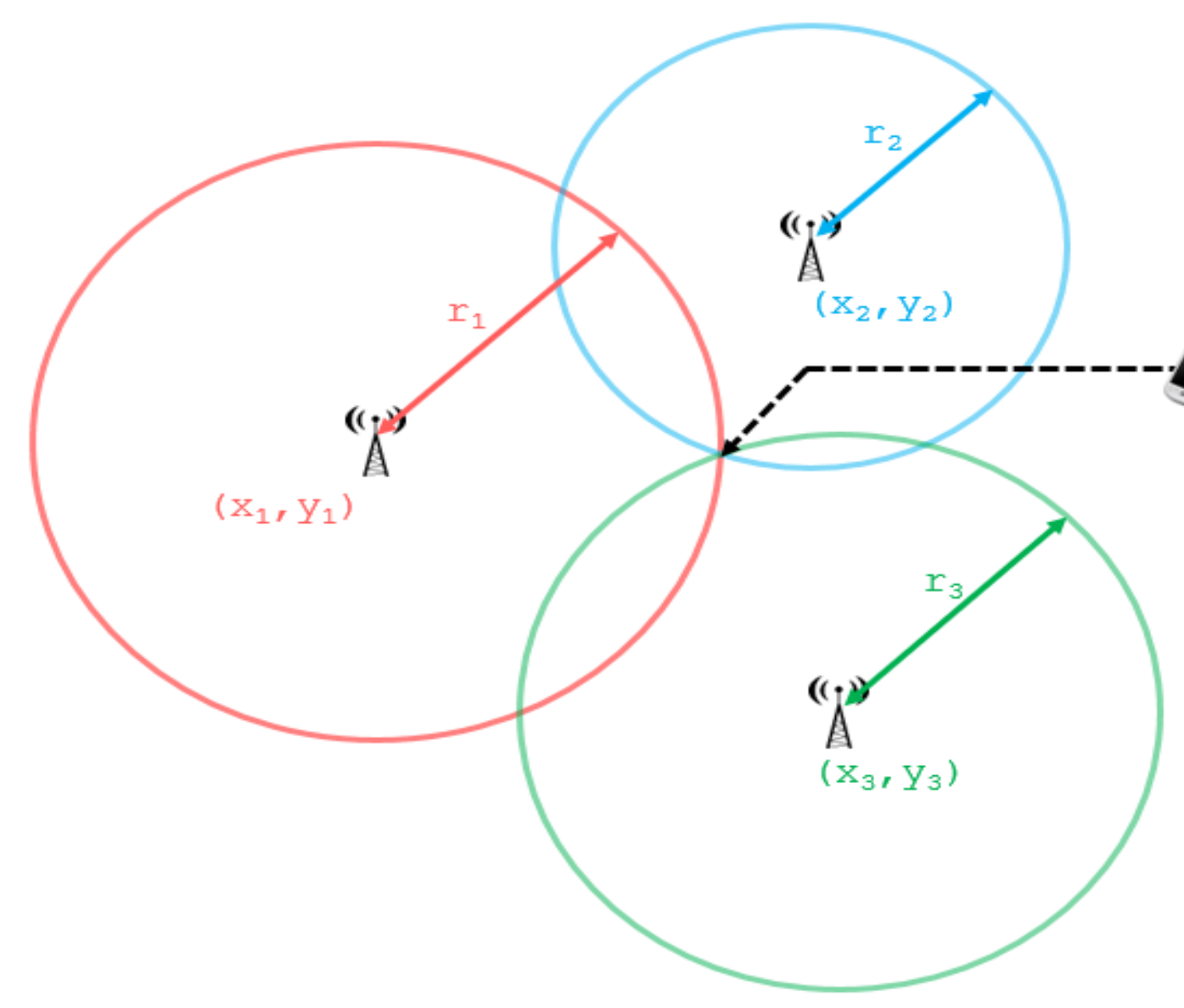
RSSI to Range Calibration

- To get the correct RSSI to range between the mobile device running the parasite application and the beacon device, RSSI values at varying distances had to be collected. We conducted this test with the provided android phones, which were LG V30s. More smartphones would need to be tested as hardware between phones are not always the same. So antenna sensitivity won't be consistent with different mobile phone makes and models. These test results will be used to map out the RSSI value to a range for similar devices.

Verify each component

- Android application
  - For the Android parasite application, beacons that were located had their data printed out and manually inspected through recorded logs, shown in Figure 4. "Beacon id1" is the beacon's UID, "id2" is the major value, "id3" is the minor value. Under the ID values is the recorded RSSI value. Lastly the Latitude and Longitude of the mobile device running the Android parasite application.
- iOS application
  - The debugging statements from the iOS are shown in Figure 5. The iOS parasite application was able to utilize the REST API. Correct transmission of data was verified through comparison of the output of the parasite application with that received by the API connection point. Since the server was not connected to the API, so this received data presented in Figure 5 was manually copied and pasted into the server.
- Server
  - To ensure that the trilateralization geolocation calculation displayed accurate results on the server, several test GPS coordinates were inputted using the algorithm.

Figure 4: Geolocation Algorithm



Acknowledgements

We would like to thank our mentor Dr. Walter Powell, our SME, Dr. Misty Blowers, and our customer, Rick Shaffer, of TWSS for the opportunity to help develop this technology and taking the time to offer insight and guidance for this project throughout the year. We wish you the best of luck in the future and hope to see this project take off in the near future!

References

- Admin, E. G. (2020, April 2). What is RSSI and its acceptable signal strength? Retrieved April 5, 2020, from https://helpcenter.engeniustech.com/hc/en-us/articles/234761008-What-is-RSSI-and-its-acceptable-signal-strength-
- E. Freeman, "Using Python Inside SQL Server", Jan 2018. Accessed on: Jan 5, 2020. [Online]. Available: https://blogs.endjin.com/2018/01/using-python-inside-sql-server/
- 101 Computing, "Cell Phone Trilateration Algorithm", Mar. 2019. [Online]. Available: https://www.101computing.net/cell-phone-trilateration-algorithm/?fbclid=IwAR1cF79LxcBYj_8ja96zC8K6TZST7ALOfSIPcQAAgCaSs24gBCeAC3xzg7w. [Accessed Jan. 7, 2020].
- D. G. Young, android-beacon-library-reference. [Online]. Washington D.C: Radius Networks. Available: https://github.com/AltBeacon/android-beacon-library-reference, Accessed on: Dec. 2019.
- N.A. "How to Calculate Distance from the RSSI value of the BLE Beacon". October 7, 2016. Available: https://iotandelectronics.wordpress.com/2016/10/07/how-to-calculate-distance-from-the-rssi-value-of-the-ble-beacon/