Blue Vision

Jake Pu, Jackson Ward, Carl Guo

What is Blue Vision?

Our prototype derives its origin from two different papers

- The first is the paper is from Zuo et al. [1] on BLE device fingerprinting via UUIDs.
- The second is the paper on Lumos [2], a Wifi Localization device using an application

We use a Raspberry Pi with BlueZ and an Inertial Measurement Unit MPU6050

- We find all the different bluetooth devices with the fingerprintings
- We synchronize the BLE data with MPU6050 measurements to localize these devices

[1] Chaoshun Zuo, Haohuang Wen, Zhiqiang Lin, and Yinqian Zhang. 2019. “Automatic Fingerprinting of Vulnerable BLE IoT Devices with Static UUIDs from Mobile Apps.” In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (*CCS '19*). Association for Computing Machinery, New York, NY, USA, 1469–1483. https://doi.org/10.1145/3319535.3354240

[2] Sharma, Rahul Anand, Elahe Soltanaghaei, Anthony Rowe, and Vyas Sekar. 2022. "Lumos: Identifying and Localizing Diverse Hidden IoT Devices in an Unfamiliar Environment." In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association. Boston, MA, USA,

Motivation

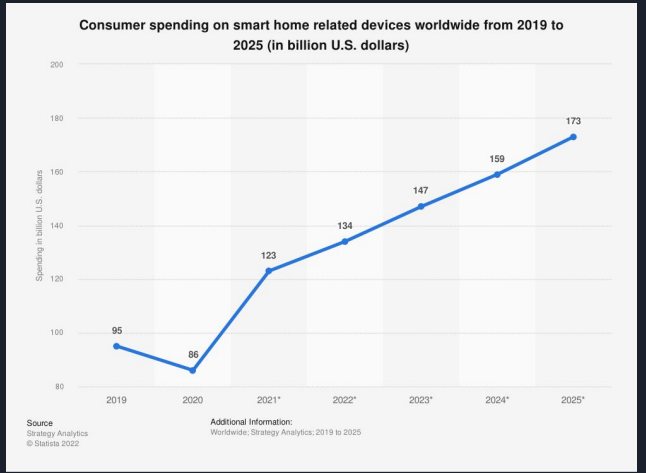
Adoption of home IoT devices are growing rapidly.

- In 2019, smart home IoT device spending reached $95 billion, forecasted $134 billion by 2022.

These devices could be used to monitor you in unfamiliar environments like AirBnB or hotel room.

- There are already been reports of surveillance devices used in AirBnBs [1], motels [2], and cruise ships [3].
- A 2019 survey [4] of American travellers showed 58% are concerned about hidden surveillance devices and 11% have actually found a hidden camera in past rental.

We want to empower users to detect where and what devices are in the nearby area.

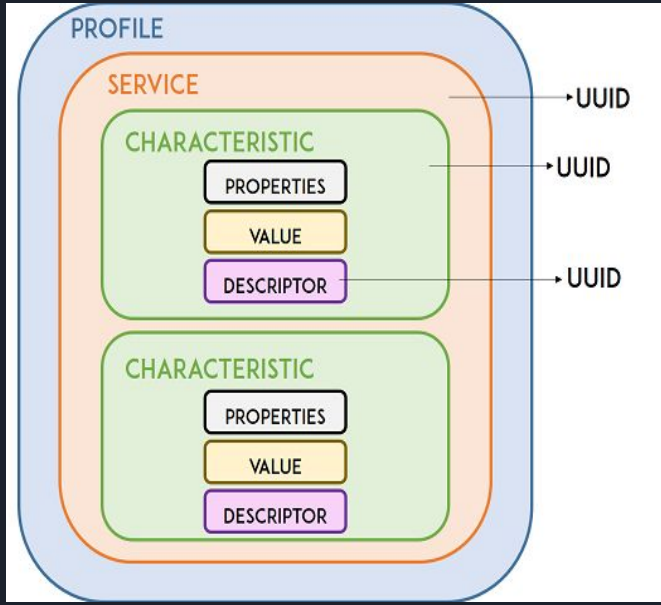


Strategy Analytics. (June 7, 2021). Consumer spending on smart home related devices worldwide from 2019 to 2025 (in billion U.S. dollars) [Graph]. In Statista. Retrieved April 28, 2022, from https://www.statista.com/statistics/873607/worldwide-smart-home-annual-device-sales/

[1] Airbnb Has a Hidden-Camera Problem . https://www.theatlantic.com/technology/ archive/2019/03/what-happens-when-you-find- cameras-your-airbnb/585007/.
[2] Hundreds of south korean motel guests were secretly filmed and live-streamed online. https://www:cnn: com/2019/03/20/asia/southkorea- hotel-spy- cam-intl/index:html.
[3] Couple says they found hidden camera pointing at their bed in carnival cruise room. https://www: insideedition:com/ couple-says-they-found-hidden-camera-pointingtheir- bed-carnival- cruise-room-47948.
[4] More than 1 in 10 airbnb guests have found hid- den cameras: Survey. https://www.inman.com/ 2019/06/07/more-than-1-in-10-airbnb-guest- have-found-cameras-in-rentals-survey/#:~: text=A%20recent%20survey%20shows%20that, have%20actually%20found%20surveillance% 20equipment.

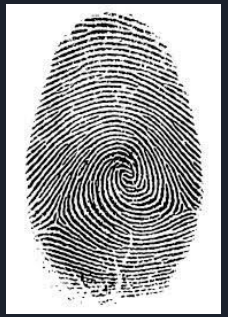
What are UUIDs?

- UUID stands for Universally Unique Identifier
- Each service, characteristic, and descriptor has a unique identifier.
- UUIDs are either in 16 bits for predefined services or characteristics or in 128 bits for custom ones.
- Usually, each manufacturer will have a unique set of UUIDs used for services and characteristics. We can exploit it to identify manufacturer, and even type, of the devices.



Bluetooth Fingerprinting

- Able to get BLE database collected from BLE device fingerprinting authors and used for our implementation.
- Sniff BLE advertisement packets to get their UUID information as well as signal strength (will be useful for localization.)
- Once UUID of packet is found, we reference with the database to detect device model and manufacturer.



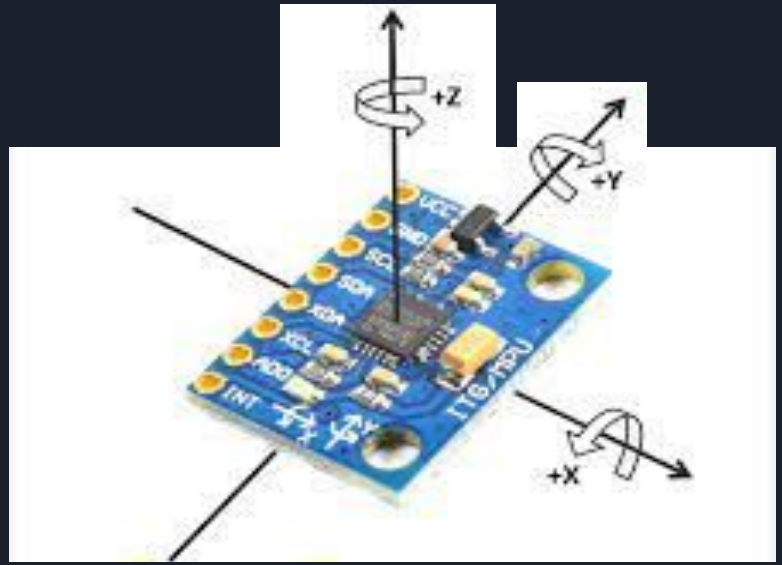
How are UUIDs extracted?

- From the static analysis of Android apps
- From the running these Android apps and placing hooks at Android functions that use UUIDs
  - getService()
  - getCharacteristic()

Tracking the Location - MPU6050

The MPU gives back several different important pieces of data:

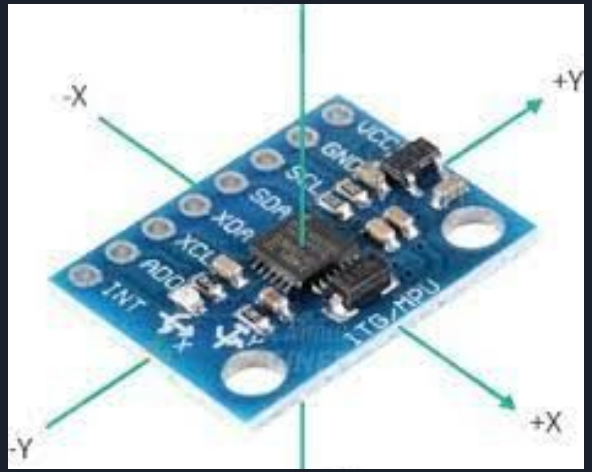
- Acceleration (x,y,z)
- Gyro - Angular Velocity (x,y,z)



Tracking the Location - MPU6050

For simplicity, we focus on the acceleration and maintain the orientation

- We need to integrate data from the accelerometer twice to find the relative position
- Due to the noise, this is a difficult problem



Tracking the Location - Calibration

Due to the noise we need to calibrate the MPU6050 and calculate offsets

Calculate values to calibrate:

- Run the localization program for 1000 runs and find the ranges
- Can also use extended ranges to account for those missed

```
mpu = MPU6050()
l=''
with open('cs598cg/code/Jackson Processing/nomu.txt','w') as the_in:
    for _ in range(1000):
        l += str(mpu.read_output()).strip('(').strip(')') + '\n'
    the_in.write(l)
ranges = find_offsets('cs598cg/code/Jackson Processing/nomu.txt')
print("Now Callibrated: offsets", ranges)
```

Tracking the Location - Calibration

Calculate acceleration values:

- Subtract the average from each measurement to normalize and reduce variation
- Zero out range and measure of extent above or below the range

```
#Now we try movement
ll = []
try:
    while True:
        ll += [list(mpu.read_output()) + [time.time()]]
        sleep(0.001)
except KeyboardInterrupt:
    with open('cs598cg/code/Jackson Processing/mo.txt','w') as the_out:
        l_write = ''
        for elem in ll:
            for i in range(len(elem)-1):
                if elem[i] >= ranges[i][0] and elem[i]  <= ranges[i][1]:
                    l_write += '0,'
                elif elem[i]  < ranges[i][0]:
                    l_write += str(elem[i]-ranges[i][0])+','
                elif elem[i]  > ranges[i][1]:
                    l_write += str(elem[i]-ranges[i][1])+','
            l_write+= str(elem[-1]) + ',' + '\n'
        the_out.write(l_write)
```

Tracking the Location - Calibration

Calculate acceleration values:

- Integrate one for the relative velocity
- Integrate again for the relative position

```
#velocity is found by integrating area
#position is found integrating area_list
areax_list = scipy.integrate.cumulative_trapezoid(areax, x=set_dx)
print(scipy.integrate.trapezoid(areax_list,x=set_dx[:-1]))

areay_list = scipy.integrate.cumulative_trapezoid(areay, x=set_dx)
print(scipy.integrate.trapezoid(areay_list,x=set_dx[:-1]))

areaz_list = scipy.integrate.cumulative_trapezoid(areaz, x=set_dx)
print(scipy.integrate.trapezoid(areaz_list,x=set_dx[:-1]))
```

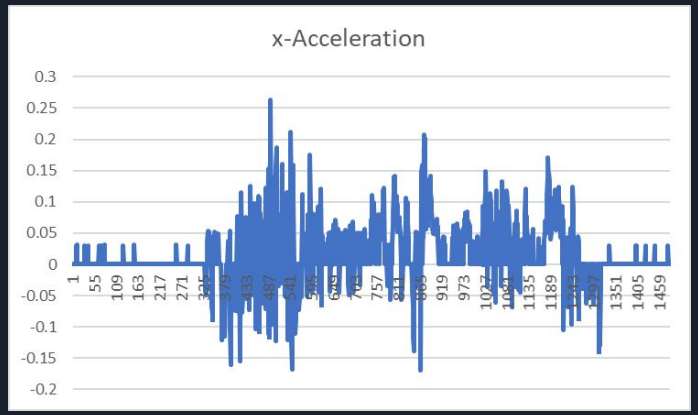
Challenges and Pitfalls

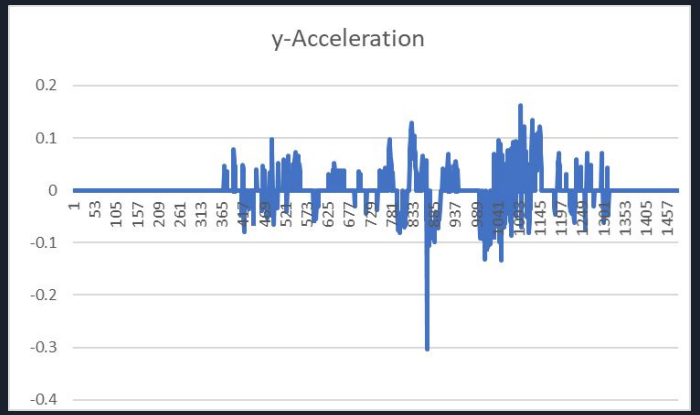
IMU Positioning

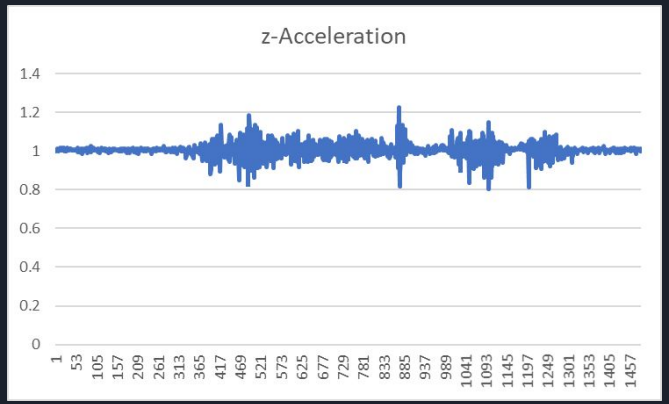
- Not initially calibrated correctly - (entire day of work lost)
- Noise and error exaggerated by integration (multiple implementations to mitigate)
  - Calibration - reducing by average of values measured while still
  - Zeroing out measurements in the noise range
  - Fast Fourier Transformation Direct Digital Integration [1]
- Further inquiry on Lumos Project [2]
  - 50 years of research behind IMU and position
  - Used a camera to mitigate noise and make reading more accurate

[1] Ribeiro, José Geraldo Telles, Jaime Tupiassú Pinho de Castro, and Marco Antonio Meggiolaro. "An algorithm to minimize errors in displacement measurements via double integration of noisy acceleration signals." *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 43.8 (2021): 1-21.
[2] Sharma, Rahul Anand, Elahe Soltanaghaei, Anthony Rowe, and Vyas Sekar. 2022. "Lumos: Identifying and Localizing Diverse Hidden IoT Devices in an Unfamiliar Environment." In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association. Boston, MA, USA,

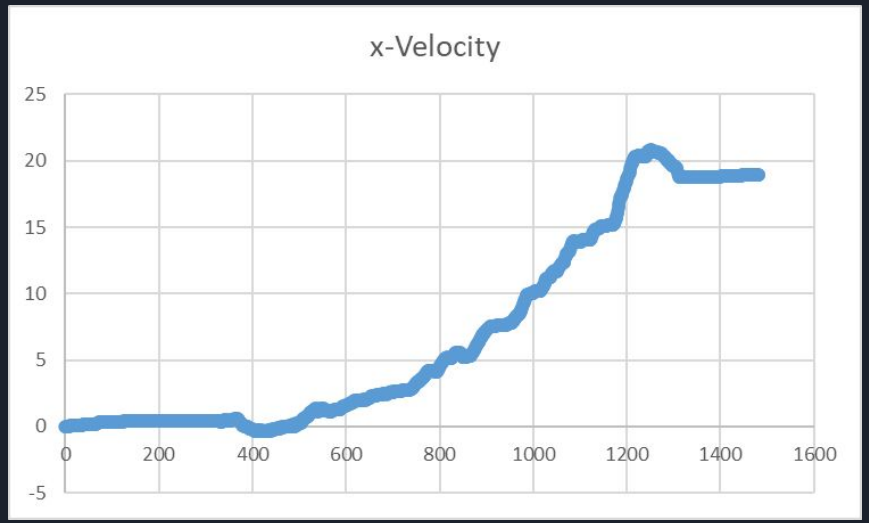
Acceleration Data - After Calibration - Issue

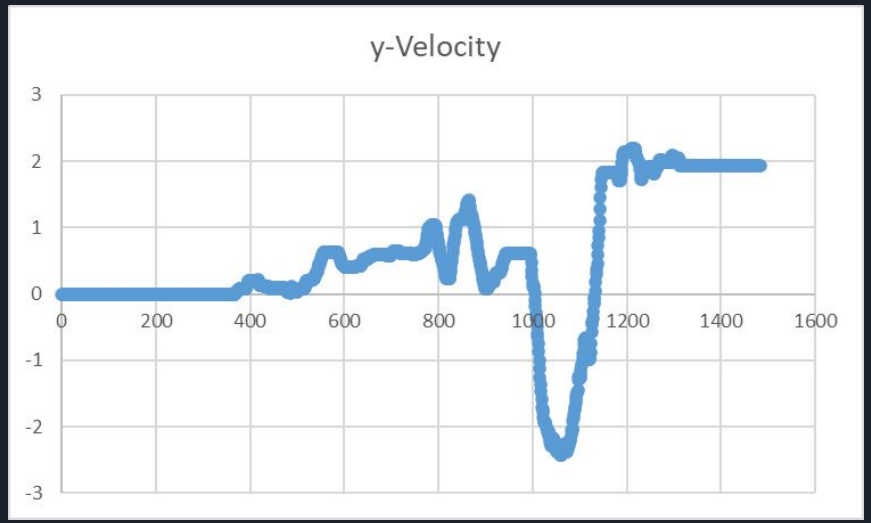




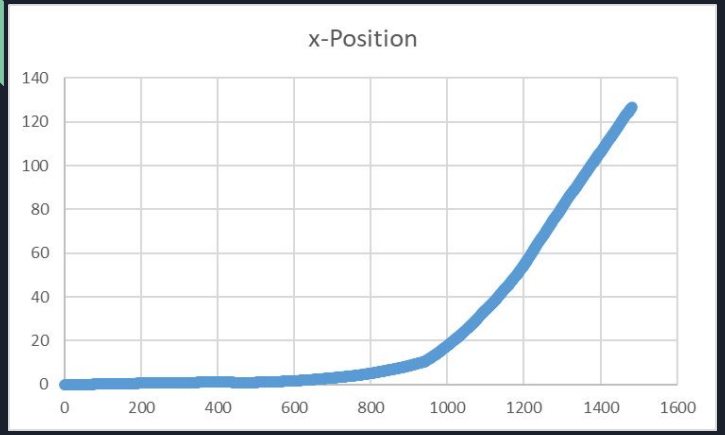


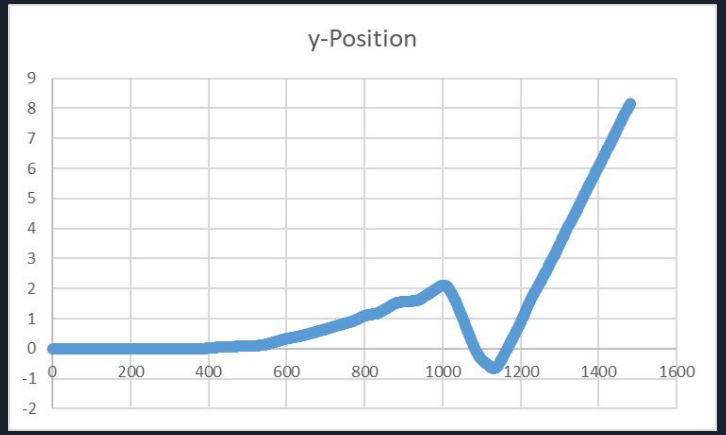
Velocity Data - After Calibration - Issue

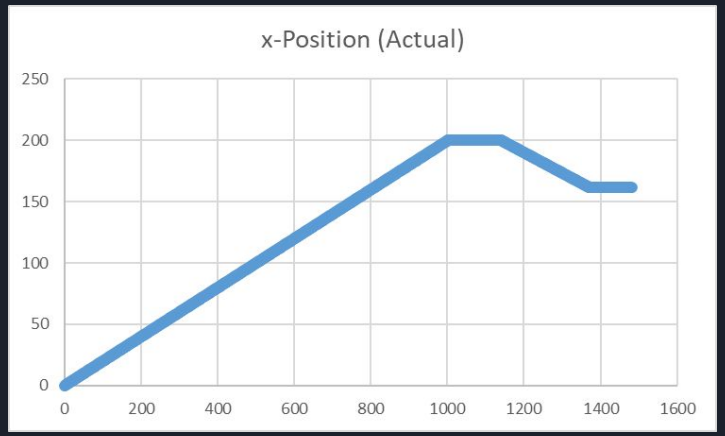


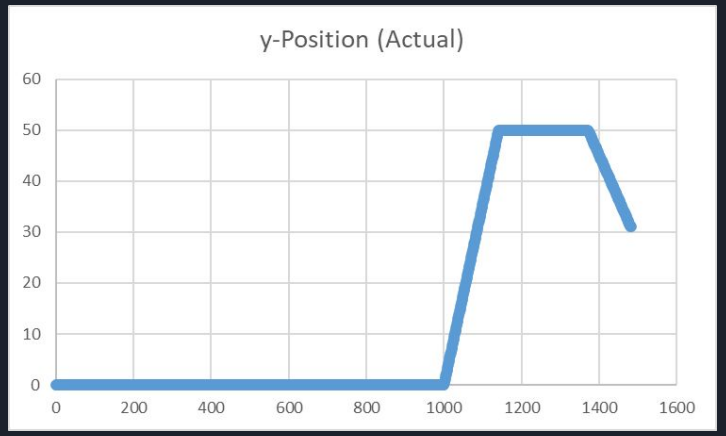


Position Data - Comparison - Issue

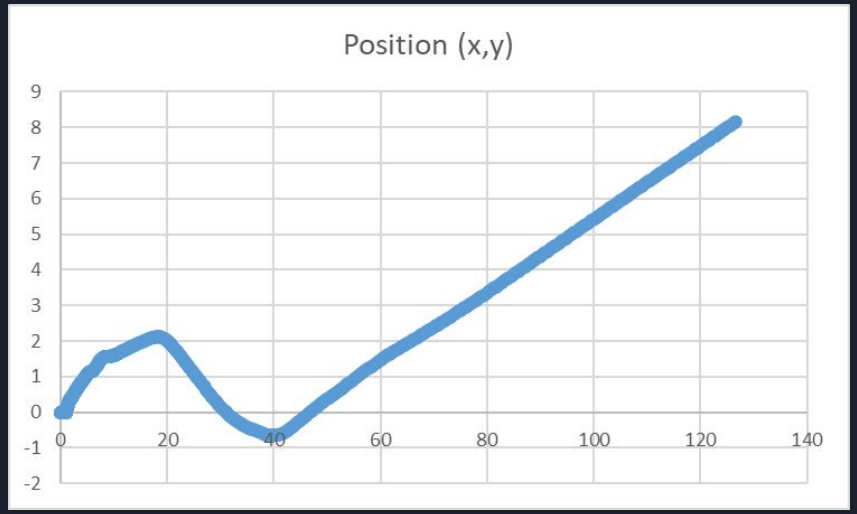


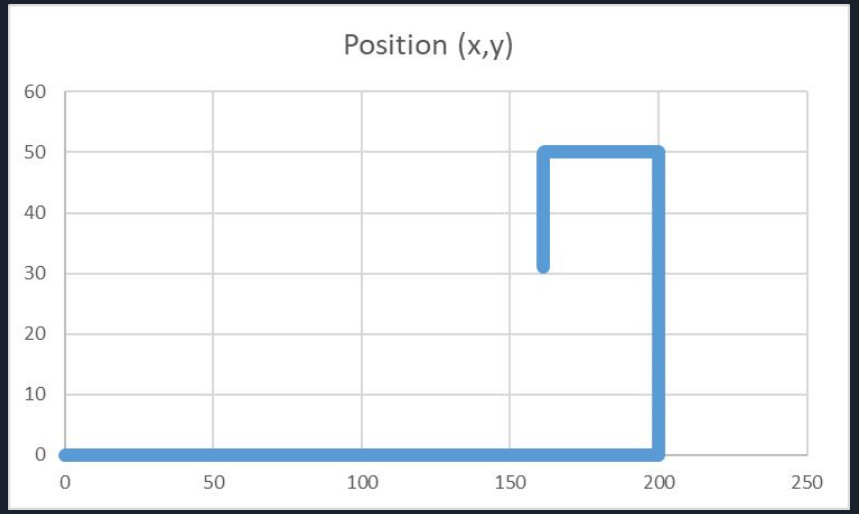






Position Data - After Calibration - Issue





Processing to Visual

Acceleration measurements and Bluetooth Data are both run and recorded simultaneously

The data for both of these is correlated to get:

- Signal Strength
- Sensor Position
- Time

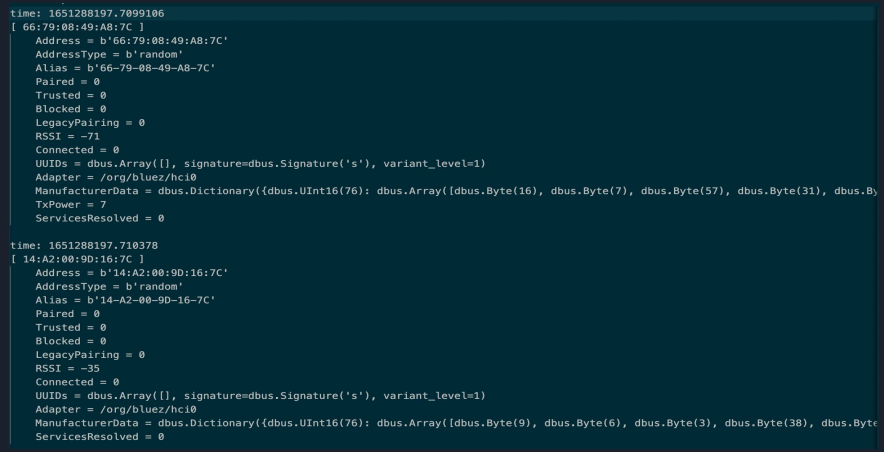
This data is then passed on to make the visualization

Experiment

Due to IMU measurements not working as expected, we planned a route and manually record timestamps once we reach checkpoints.

- Plan out a route on third floor of Siebel and measure distance segments based on length of each floor tile.
- Walk at constant pace holding raspberry-pi (discovering BLE packets), while another person records timestamps every few steps.
- Do interpolation of coordinates and timestamps to get millisecond level granularity.

Write collected data into files for visualization and fingerprinting purposes.



Retrieved Data

Two csv files are generated after experiment is run and data is processed:

- Packet file: unix timestamp, RSSI (signal strength), UUID
- Position file: unix timestamp, x-coordinate, y-coordinate

Need to merge data from both files together using closest timestamps to get entries with (UUID, x, y, rssi, timestamp) for localization.

```
1651288231.5473034, -46, 0000feed-0000-1000-8000-00805f9b34fb
1651288247.1648812, -33, 0000feed-0000-1000-8000-00805f9b34fb
1651288262.1976647, -83, a3e68a83-4c4d-4778-bd0a-829fb434a7a1
1651288262.7419221, -63, 0000feed-0000-1000-8000-00805f9b34fb
1651288264.5631726, -98, 0000fe9a-0000-1000-8000-00805f9b34fb
1651288264.7152574, -89, 0000fe78-0000-1000-8000-00805f9b34fb
1651288264.878878, -98, 0000fe9a-0000-1000-8000-00805f9b34fb
1651288266.502736, -76, 0000fe78-0000-1000-8000-00805f9b34fb
1651288266.7288334, -47, 0000feed-0000-1000-8000-00805f9b34fb
```

Packets

```
1651288197.1011,0.0,0.0
1651288197.3828535,0.0,0.5
1651288197.664607,0.0,1.0
1651288197.9463606,0.0,1.5
1651288198.2281141,0.0,2.0
1651288198.5098677,0.0,2.5
1651288198.7916212,0.0,3.0
1651288199.073375,0.0,3.5
1651288199.3551285,0.0,4.0
```

Positions

Fingerprinting

Authors of the BLE fingerprinting paper gave us their UUID database after we emailed them.

From UUID retrieved from sniffing BLE packets, we can determine possible device manufacturers of a device, based on the uuid matching with Google Play companion app id.

- For one device, there can be multiple matches as shown in later example.

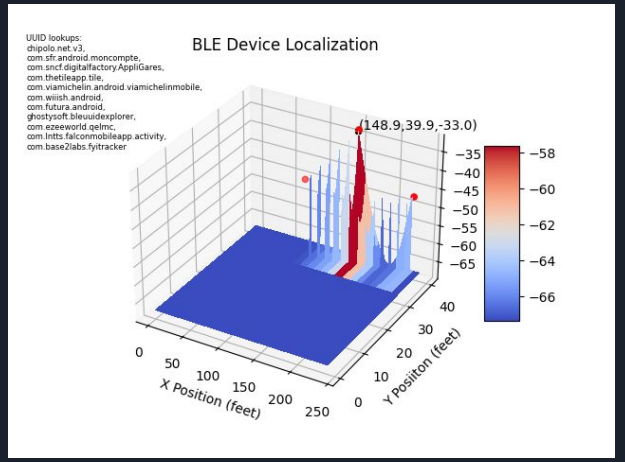
Visualization

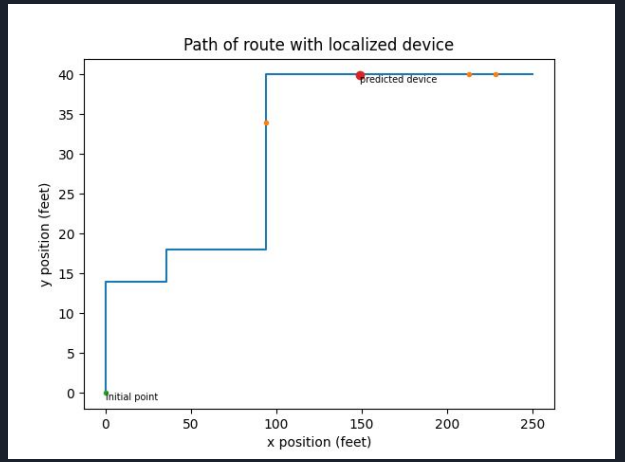
From location tracking with the IMU, we get (x,y) coordinates relative to initial starting point.

- From packet sniffing, we get RSSI
- Using coordinates and RSSI, we can localize device.

Similar to the paper in the previous work, we graph coordinates (x,y) along with RSSI values (z) and interpolate to find “peaks”

- The peak of the interpolated graph is where we predict where the device is.
- Localization and visualization run with python script.





Limitations

- Unable to automate path/position generation due to IMU inaccuracy.
- Only retrieved a small number of BLE advertising packets (around 4-5 of same device), making localization by interpolation possibly inaccurate.
    - BLE advertising interval is long: It might take seconds for a device to send a second packet. It means that the data collected is very sparse.
    - It is not very typical for a BLE device to enclose UUID in their advertising packets. A lot of the advertisement gives no UUID (since we are sniffing passively.)

Possible Future Work

- Integrate system into an Augmented Reality interface.
  - User can have a camera or headset which displays markers where a predicted hidden device could be located and what it is.
- Try to connect to each BLE device we find using the “just works” pairing. If we can connect to these devices successfully, we can get their GATT to extract most UUIDs for fingerprinting.
- Collect more BLE data in a slower manner - rest the collector at a location for a few seconds before moving to the next location.

Sarah Prange, Ahmed Shams, Robin Piening, Yomna Abdelrahman, and Florian Alt. 2021. PriView– Exploring Visualisations to Support Users' Privacy Awareness. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 69, 1–18. https://doi.org/10.1145/3411764.3445067

Questions?