Wi-Fi Analysis and Development of the Wi-Fi Scanning Module

Team Lev Tours
Kyle Savery, Erik Clark, David Robb,
Ariana Clark-Futrell, Alexis Smith

Team Sponsor
Dr. Michael Leverington

Team Mentor
David Failing

Northern Arizona University
April 2021

Overview

This document provides an analysis on the accuracy of our Wi-Fi localization system using the Engineering building's Wi-Fi, located on campus at Northern Arizona University. As well as justifications for several design decisions within our team's software.

Table of Contents

1 Introduction

The original concept behind our team's project was to have us navigate a building using Wi-Fi information and create a Graphical User Interface (GUI). Our team was successfully able to implement a GUI that allowed users to receive tours around any building with an adequate to-scale map. However, the final software product does not rely on Wi-Fi to navigate the building. The main reason for this is that in the given time frame our team was unable to determine a device's position to within the target range of 2 meters. Moreover, the process of retrieving nearby Wi-Fi information (known as a *Wi-Fi Scan*) was much slower than anticipated which complicated development further. Consequently, our team needed to pivot and focus on completing, testing, and delivering the GUI. The final product does include a Wi-Fi component, but in its current state it is just a demonstration of the progress made so far toward giving the robot a way to navigate with Wi-Fi. This document is meant to give Dr. Leverington and upcoming teams a complete understanding about what our team investigated, how effective our Wi-Fi module ended up being, and where future work could be started.

2 Developing the Wi-Fi Scanning Module

A requirement of this project was that the software needed to use a building's existing Wi-Fi infrastructure to localize (determine the position of) the running device and navigate from point to point. Our team was not allowed to use other forms of localization such as GPS or Bluetooth. As such we began this project by researching how Wi-Fi could be readily used to navigate an indoor space. Several techniques exist to accomplish this such as Angle of Arrival (AoA), Time of Flight (ToF), and fingerprinting.

AoA operates by measuring the incoming angles of signals from nearby Access Points (AP's) (generally, these are the routers within a building). As well as the known coordinates of those AP's to determine the device's position. However, this technique requires the hardware to employ multiple antennas which was not the case for our team's provided laptop, so this strategy was not investigated further.

For ToF, the time it takes for a signal to travel to and from each AP is used to determine the current position. Again, this method uses exact positions of AP's within a space. This was an issue for our software as the locations of routers within the buildings on campus was unavailable. Our team realized that going around and collecting this data from scratch was not a worthwhile direction to take, as this would make setup in new buildings exponentially more time consuming. So, the technique that our team decided was best suited for our purposes was fingerprinting.

2.1 Fingerprinting

Fingerprinting relies on a measure of the signal strength from detected AP's known as the Relative Signal Strength Indicator (RSSI). RSSI values range from 0 to -100 dBm with 0 being the strongest possible signal. During the fingerprinting setup process, some quantity of Wi-Fi scans are taken at some location within a building. This creates an RSSI *data set,* such that a single scan performed after the setup process can be compared with each established location's

data set to determine the device's current location. Now that our team had decided upon how we wanted to try and navigate a building solely using Wi-Fi, the next step was to figure out how we could retrieve these RSSI values from within our software.

2.2 How our Software Scans

The provided laptop runs Ubuntu 20.04, so the processes that were investigated for obtaining RSSI information were focused on operations compatible with Linux machines. We found a command line call, *iwlist*, that served this purpose. The command *iwlist* returns the Media Access Control (MAC) address (a unique identifier for network interfaces) and an RSSI value for each detected AP across each detected network. To be specific, an AP can no longer be detected by a device once the device moves far enough away from it, so not every AP in a building will be accessible from every location. Other Linux wireless tools for the command line our team investigated were *iwconfig* and *iwspy*, both of which do not return RSSI information for all detected AP's. Another operation that was promising at first was *iw dev station dump*, which returned RSSI data for the current connectly AP. We looked into whether this command could be configured to return all nearby AP's information, but no procedure was found. As such our team had to move forward with *iwlist*.

The *iwlist* command was the first that our team found, but the reason we continued to research alternatives was that the scanning operation carried out by *iwlist* takes a relatively substantial amount of time. Scans typically last between 3 and 4 seconds (see section 3.2), which would hinder an autonomous robot from being able to navigate effectively. From this point, our team realized there may be an issue with attempting to navigate via Wi-Fi (at least in the form we were able to achieve in the scope of this project). Regardless of this issue, we still needed a way to compare recent scans with data sets acquired during the fingerprinting setup process in order to determine the device's location.

2.3 Rating Calculation

There already exist software products pioneering the field of Wi-Fi localization, but most of them rely either on expensive hardware, the techniques of AoA or ToF, or highly involved mathematics. As such our team approached the problem of comparing RSSI data sets with the mindset that our project is based in academia. That is, we decided to dedicate the time we could afford to spend on the Wi-Fi scanning module on creating our own method of comparison, rather than trying to understand and use someone else's work which may not even perform the way that we need.

In our system's implementation, an RSSI data set consists of the minimum, maximum, and average RSSI value that a location receives from each detected AP over the course of 25 setup scans. The data set also includes the corresponding MAC address for each AP. As for the comparison process, each established location is assigned a *rating* that depicts how similar its data set is to a recent Wi-Fi scan. The location with the highest rating is selected as the position that the device is most likely closest to. Our software's comparison algorithm follows the basic idea that, for a given location, the closer a sample RSSI value is to the stored average then the

more “points” the rating will gain. In our software’s final state, the algorithm we employ is referred to by us as a *penalty assisted comparison*. This algorithm works as shown below.

- ❏ For each location you want to compare a recent scan to, assign a rating of 0 and calculate the number of “penalties” this location is able to receive before having its rating reduced as the floor of 10 percent of the number of AP’s for this location’s data set.

- ❏ For each sample AP in a recent scan, get the sample’s MAC address.

- ❏ For each AP in the location being currently compared, find the corresponding RSSI data set to match the sample AP.

- ❏ Now, using the stored data set:
  - ❏ If the sample RSSI value ($current$) is greater than or equal to the minimum stored value (min) and less than the average stored value (avg), then increase this location’s rating by$(1 - ((avg - current)/(avg - min))$. Unless the average is equal to the minimum, then increment by 1.

  - ❏ If the sample RSSI is greater than or equal to the average stored value and less than or equal to the maximum stored value (max), then increase this location’s rating by$(1 - ((current - avg)/(max - avg))$. Likewise, if the average equals the max, increment by 1.

  - ❏ If the RSSI sample is outside the min and max range, then calculate the amount of points to deduct from the rating by $(min - current)/(avg - min)$ or $(current - max)/(max - avg)$, if the current value is less than or greater than the range, respectively. Before deducting however, decrement the amount of allowed penalties. If this value has reached zero, then reduce the rating by the above amount, otherwise do nothing.

- ❏ Finally, return the location that received the largest rating.

To summarize, points are awarded to a location’s rating for each RSSI value from a recent scan that falls within the stored range. Otherwise, it reduces the rating after a certain number of allowable fails. Though this algorithm utilizes three nested for loops, our team did not bother redesigning it as the true limiting factor in our Wi-Fi operation is the time required to scan. The number of locations and AP’s is small enough that it does not noticeably impact the overall time that it takes to receive a location estimate.

Along with the above procedure, there were other comparison methods that our team considered. However, these consistently performed slightly worse in terms of how close the locations defined during setup could be to placed next to each other. That is, the measure of how accurate these comparisons were (and our Wi-Fi scanning module as a whole) is based on achieving a success percentage when determining the correct location. This percentage is

referred to as the *level of accuracy,* and the goal for our system to be 95% accurate within 2 meters. Admittedly, our module performs quite poorly with respect to this original target range. The actual accuracy of our system, along with how it was tested is discussed in the following sections.

3 Wi-Fi Tests and Results

To determine the final required distance between saved locations to achieve a level of accuracy of at least 95%, our team conducted the following test which was performed in the Engineering building on Northern Arizona University's campus (NAU).

- ❏ Throughout one floor (as depicted in Figure 3.1), locations were set up via the previously described fingerprinting process. These locations were placed at intervals of 20, 19, 18, …, 2, 1 yards for a total of 20 tests. **Note:** the reason our team switched from meters to yards is that the tiles on the floor of the engineering building are exactly 1 foot across. For each of these tests, the tester would walk around until a sample size of 250 location estimates was reached. To reach this, at each saved location they would stop, perform a Wi-Fi scan, and mark down if the result was correct or not. The results for these tests are shown in Figure 3.2.

To supplement the above final testing process, we also tested the behavior of scan times in different buildings as well as the optimum quantity of scans to perform when setting up a location's RSSI data set.

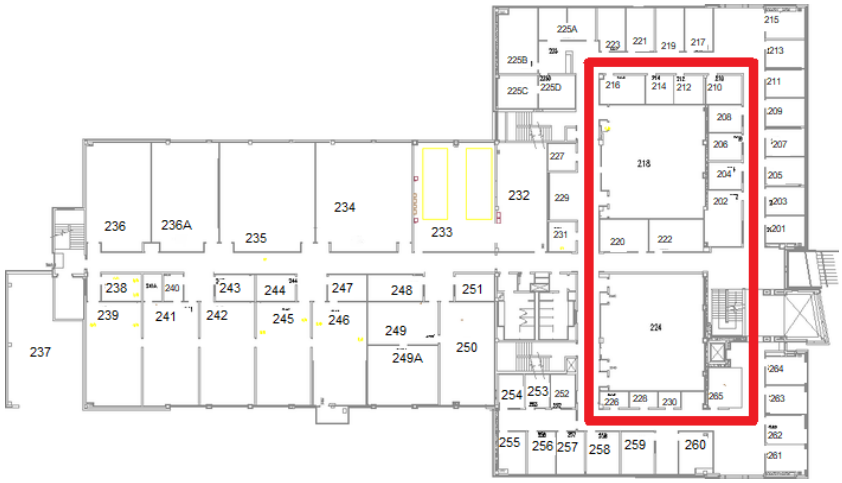

Figure 3.1: Path used for testing on the 2nd floor of Engineering

The testing process for the Wi-Fi scanning module was very limited by the amount of time scanning takes. Our team had to choose the types of tests we wanted to conduct carefully as the majority of our time and effort had been focused on completing the main part of our software, the GUI.

3.1 Range of Wi-Fi Localization

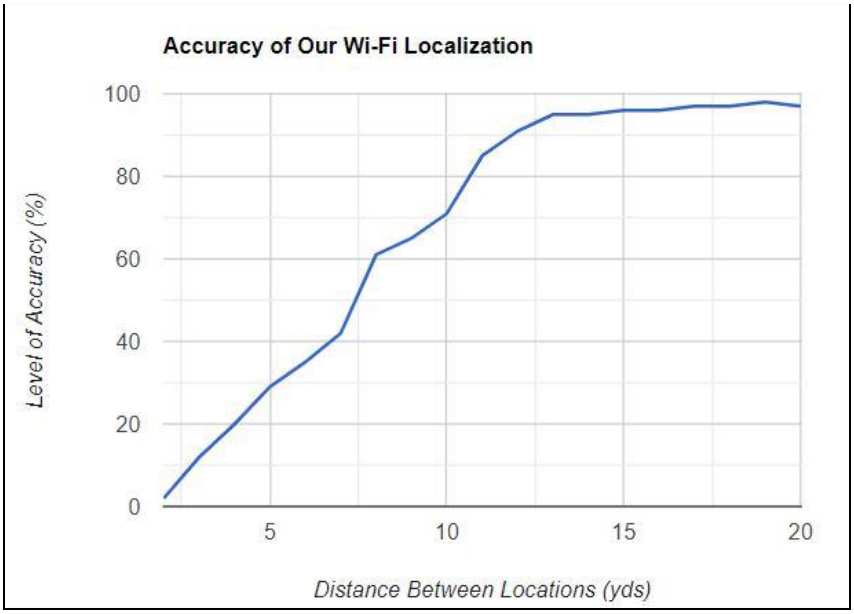

Figure 3.2: How distance between locations affects accuracy

Clearly, our software is incredibly inaccurate when the saved locations are less than than the target goal of 2 meters (~2.19 yds) apart. And it is not until approximately **12 yards** (~10.9m) that our system is able to reliably achieve **95%** accuracy.

3.2 Scanning Times

These tests were conducted with a sample size of 300 in each building, and while the scans were executing the tester walked nearly everywhere in each building. The 'Avg Networks' and 'Avg AP's' indicate the average number of networks and AP's detected across all scans.

Building Name	Avg Networks	Avg AP's	Avg of Scan Time (Sec)	Std Dev of Scan Time (Sec)
Engineering	15	71	3.34	0.31
Cline Library	23	83	4.12	0.53
Nursing	8	62	2.87	0.25
Social and Behavioral Sciences	12	65	3.13	0.34
Adel Mathematics	6	57	2.56	0.29

Figure 3.3: Average scan times through 5 different buildings at NAU

The table above depicts how the average amount of scan time differs between buildings, but more importantly it offers some insight into why scanning times behave the way they do. Specifically, how the quantity of networks and AP's effect scan time. It is worth noting that scanning the AP that a device is currently connected to takes a matter of milliseconds. Using the command brought up in section 2.2, *iw dev station dump*, the current AP's RSSI signal is consistently returned in less than 10 ms. To reiterate however, this function does not scan all nearby AP's. The command *iwlist* takes so long because the actual process of *switching* to look at a non-connected AP or network is where the vast majority of time is spent. And as demonstrated in the table, it appears that switching to look at AP's on a different network takes longer than switching to an AP on the device's currently connected network.

3.3 Optimum Scan Quantity for Setup

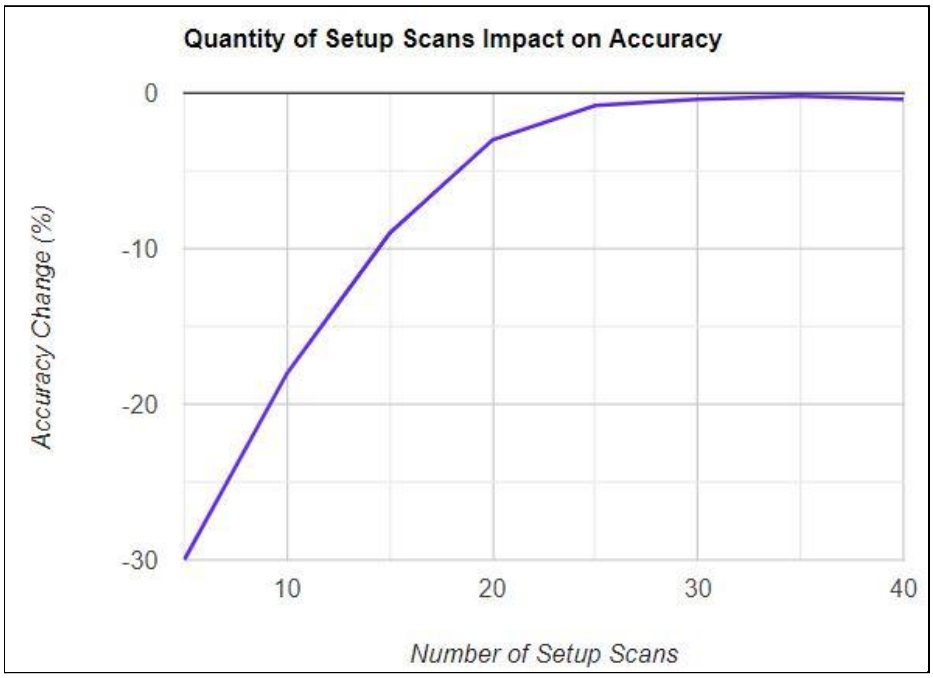

Figure 3.4: Number of setup scans and its effect on accuracy

In section 2.3, it was stated that 25 scans is the amount of Wi-Fi scans that should be performed when fingerprinting a location. To reach this number, our team took locations that were 12 yards apart and tested how the level of accuracy, with respect to 95%, was affected by changing the amount of setup scans to the set {5, 10, … , 35, 40}. So for the sample of 10 setup scans, the new level of accuracy was 95 - 18 = 77%. During the original testing from section 3.1 to arrive at the 12 yard distance minimum, all locations were set up with 40 scans. This number was chosen somewhat arbitrarily, but turned out to be a pretty decent guess as to how many scans we would need. The reason we chose 40 is that we wanted to get a sufficiently accurate representation of a location's RSSI ranges, while doing it in the least amount of time. Keep in mind that this system is supposed to be *feasible* and going around to each location to scan for a few hours at a time was certainly out of the question. The purpose behind these tests for optimum scan time was to discover if there was a quantity of setup scans such that any more

scans and the level of accuracy would not significantly improve or any less and the accuracy began to suffer. This value turned out to be approximately 25 scans.

4 Further Investigation and Unperformed Tests

The following are several tests that our team wanted to conduct, but were forced to abandon due to either time constraints or insufficient tools.

- ❏ Do RSSI data sets drift over time? That is, will they still be usable after a few days/weeks or do these areas need to be re-fingerprinted frequently to maintain the system's level of accuracy.

- ❏ Does repeatedly scanning nearby Wi-Fi information have any impact on accuracy? Since this concept is meant to be integrated with the robot which (hopefully) will be operating for hours or entire days at a time, it may be worth investigating if constant calls to *iwlist* affect the level of accuracy, if at all.

- ❏ Does only using access points on a single network (particularly NAU's main network) improve accuracy, reduce scan time, or both? As mentioned previously, *iwlist* grabs RSSI information from all available sources. There is no way to filter the output to include only a single network. In terms of parsing the *iwlist* return value, we can specify which networks we want to see, but *iwlist* will still scan all nearby networks and AP's so scan time remains unaffected.

The first of these questions is likely the most important one that future teams will need to answer if they continue down the path of using RSSI. If the data sets do become obsolete after a short period of time the only solution that our team can come up with is the robot will need to dynamically update its data sets as it moves around the building.

5 Conclusion

The purpose of the Wi-Fi scanning module was to give our software, and eventually the robot, a way to determine its position within a building using only the existing Access Points (AP's). Originally, this position was meant to be 95 percent accurate within a target range of 2 meters (~2.19 yds). However, after plenty of complications our actual required distance to hit a 95 percent level of accuracy is about 12 yards, which is closer to about 11 meters. Truthfully, a lot more work needs to be done if the robot is to be capable of navigation via Wi-Fi. That work may involve artificial intelligence and/or more advanced hardware, but whatever the solution there needs to be some pretty fundamental changes to the Wi-Fi localization system our team developed. But again this is academia, where these types of complications exist to offer future capstone teams and students plenty of room to research, learn, or even create new technologies.

6 Glossary

Access Point (AP). A hardware device that allows users to connect to a wired network, generally via Wi-Fi.

Device. For our project, the device we used was a laptop provided to us by Dr. Leverington. However, in this context a device refers to the physical piece of hardware that our software is running on.

Fingerprinting. A technique in Wi-Fi localization that is useful for estimating position. A location within a building is “fingerprinted” when some number of Wi-Fi setup scans are taken at that location in order to build up a data set that describes the ranges of RSSI values for each detected AP. Once a building is fingerprinted, then a single scan can be compared to each fingerprinted location and the data set that it is most similar to is chosen as the closest location.

Graphical User Interface (GUI). A type of user interface that allows the user to interact with the software visually by using graphical icons instead of relying only on text based input.

Localization. The process of using some set of information (in our software’s case, RSSI values) to determine something’s position in 3-dimensional space.

Location Rating. The rating of a location in the context of our software is how close of a match the location is to a recent Wi-Fi scan. The higher the rating of a location, the greater the probability of our system choosing this location as a position estimate.

Media Access Control (MAC) address. A unique identifier that is given to a network interface. This is used to keep track of every AP within a building.

Relative Signal Strength Indicator (RSSI). A measurement of the strength in a received radio signal. These values range from 0 to -100 dBm with 0 being the strongest signal.

Wi-Fi Scan. The process of getting the current RSSI values for each detected AP near the user’s device. A typical scan using our system takes between 3 and 4 seconds.