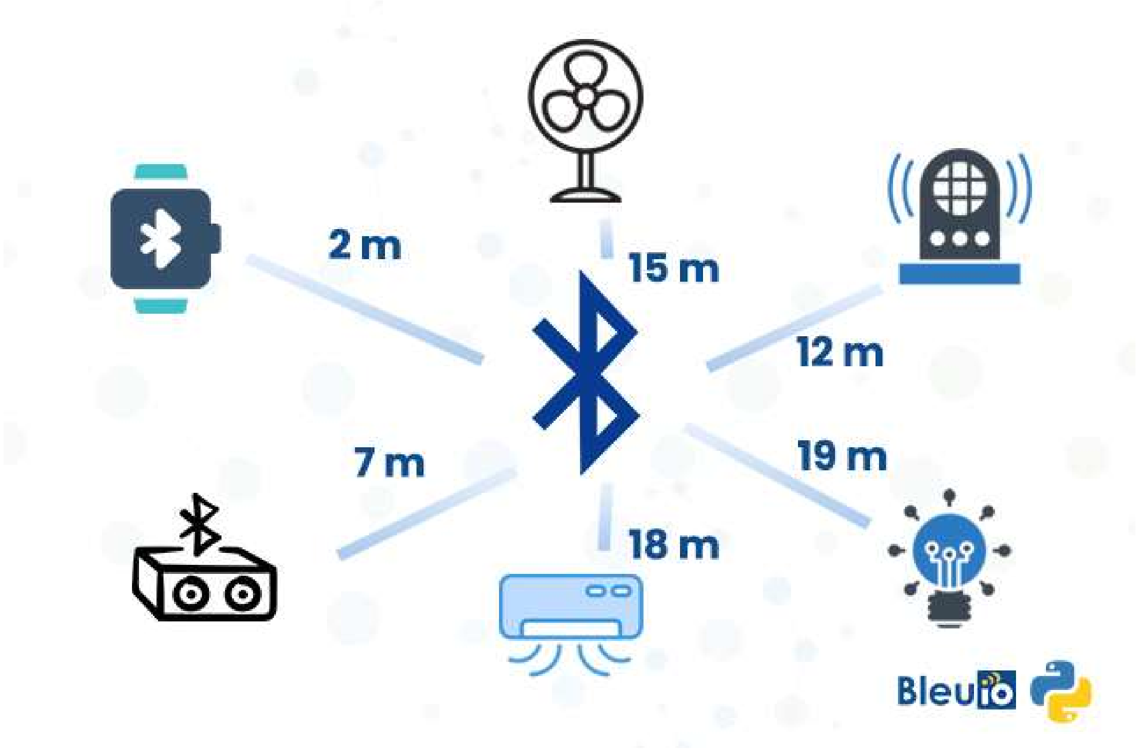
Project 7/8: Integration beacons with Social-Buddy

## General information

| | |
|---|---|
| **Project code:** | **CMI-TI-23-TINPRJ0478** |
| **Project goal:** | 'Buddy app verrijken met een Flic button en Tile tag' |
| **Team Members:** | Ahmet Oral (1023107)<br>Khizer Butt (1052313)<br>Nguyen Do (1057048) - Author<br>Terrence Zhong (1028516) |
| **Skills Coach:** | Sandra Hekkelman |
| **Technical Coach:** | Alexander Slaa |
| **Document Version:** | 1 |



Source: [https://www.bleuio.com/blog/measuring-distance-with-bluetooth-in-indoor-environment-using-python/](https://www.bleuio.com/blog/measuring-distance-with-bluetooth-in-indoor-environment-using-python/)

## Table of Contents

# 1. Glossary

| | |
|---|---|
| BLE | Its full name is “Bluetooth Low Energy”. This technology uses less energy than Bluetooth, which can last longer with the tracking tags' limited battery capacity. |
| BLE module | A module (part) within a device that is used to send and receive BLE signals. |
| Buddy Bot app | An application developed by Social Buddy to aid the elderly in their life. The app features a Buddy Bot avatar to make the application more user-friendly. |
| Companion app | An application developed by Social Buddy to aid caregivers in caring for the elderly. Caregivers can notify and check on the patient when needed. |
| Tablet | A tablet is a mobile device that is bigger than a smartphone but without a built-in hardware keyboard. In this context, this device hosts the Buddy Bot app and the BLE module. |
| Tag | A BLE tracking tag is commonly used to track your items. Some known tracking tags are the Apple Airtag and the Samsung SmartTag. This word will be used in the report with its associated brand. |
| Tile Pro | A tag (see “Tag”) from the company Tile Inc. It is a pro version with a replaceable battery and increased range compared to the normal version. |
| RSSI (BLE) | Received Signal Strength Index. It is a measurement of the relative BLE signal strength received on a device. |
| UWB | Ultra-Wide Band. It is a wireless communication technology with its main feature that enables high-precision positioning. |

## 2. Executive summary

This project is to determine the reliability of using the BLE tag and a tablet in the context of determining the distance between these 2 with RSSI values. This is experimental research, as such the conclusions are made from the observations of the tester and the researched references.

The results are rather unfavorable. The RSSI values fluctuate too much. Some of those values may appear in different ranges, which can cause the calculation for the distance unreliable. This can come from the fact that the environment can affect the RSSI values. Even letting the tablet face the wrong way can be detrimental to the distance value.

Closed-source tags are also unreliable for this use case. The manufacturer may hide its core functionalities behind its app, which is unsuitable for this project.

In conclusion, this method of distance calculation is too unreliable. Too many factors are at play that negatively impact the reliability of the distance determination.

## 3. Introduction

This project is part of a bigger project of Social Buddy. It made its way into a list of projects which the students of Hogeschool Rotterdam can choose from and our group has chosen this to work on. As a team, the integration of a tag and the Flic button with the existing Buddy system must be done within 13 weeks.

Binh Nguyen Do, a member of the team, was tasked with the research of the distance determination with the tag. As such, this report focuses on integrating a tag with the Buddy Bot application. This feature enables caregivers to check if the elderly individuals under their care have (accidentally) left the house with the Companion app on their samrtphone. Following the product owner’s request, this feature will only show the tag's distance from the tablet that hosts the Buddy Bot application and not the precise location itself.

To determine the most appropriate tag for the new feature, research was conducted. The report was then used in the meeting with the product owner to determine a tag to use. A tag must meet the following requirements:

- The tag must use BLE/Bluetooth communication technology.
- The tag must have a reach of at least 100 meters.
- The tag must last long with each change/recharge of its battery

After the meeting concluded, the Tile Pro was chosen, as its specifications met the requirements for this project [1]. This device uses Bluetooth Low Energy (BLE) [2] to communicate with the Buddy Bot application.

The tag must then be integrated into the main Buddy Bot application. Before that, the distance determination must be done. A main way to do this is to use the Received Signal Strength Indicator (RSSI) [3]. However, the calculated distance is rather inaccurate, more so when the distance becomes higher. [2] As such, some research question(s) were formulated:

- <u>Question Q1</u>: How reliable is the RSSI value of a BLE device for determining the distance between the tag and the tablet hosting the Buddy Bot application?

## 4. Theoretical Framework

The problem of tracking the location of the patient, or even the staff and equipment is not a new one. Many solutions using different technologies were made to address this problem, such as BLE, Ultra-Wide Band (UWB) [4], or Wi-Fi. [5] [6]

These solutions are feature-rich. Take Centrak's Senior Living Location System for example. It has emergency calls, geofencing alerts, contact tracing for infections, TruLocation™, and more. [7] These features are useful for keeping seniors safe. However, it is rather hard to get the hardware need to use in this project. Besides, the software is close-sourced; as such, it is hard to establish communication between their product with the system of Social Buddy. Centrak also only sells their product to companies, which makes it impossible for the user to get their hands on it. The product owner wants his application to be easy to use for both seniors and caregivers, subsequently the hardware must also be readily available in the market. This causes Centrak's solution to be unsuitable for this project and the whole Social Buddy system.

Besides, the product owner wants the distance determination to be done with a BLE device, thus the project will then rely on the BLE tags that are available on the market and after some research and consideration, the Tile Pro was chosen.

## 5. Method(s)

The determined formula for calculating the distance with RSSI is:

$$(1)\ DISTANCE = 10^{\left(\frac{Power_{measured} - RSS\ instance}{10 \times N}\right)}$$ [8]

Where:

- Distance: the distance between the tablet and the tag in meters (m).
- Power$_{measured}$: The measured power of the tag in decibels-milliwatts (dBm) with a 1-meter distance.
- RSSI$_{instant}$: The signal strength in decibels-milliwatts (dBm) with a 1-meter distance.
- N: attenuation of the signal, based on the environment. Usually, it is between 2 and 4. [8][9]

Of these values, the RSSI instant value can be measured directly based on real verifiable distances using a tape measure roll construction ruler. Multiple distance values were used to observe the change in the RSSI value. The value changes with each page refreshes within the test application.

The observation took place both indoors and outdoors, as 10m is not feasible indoors. A test app was used to observe the changing RSSI values. The app's main user interface can be found in **Attachment 1**.

The measured RSSI values would then be used to calculate the measured power and N-value. This was a trial-and-error process that was partly automated by a small C++ code, which is to be found in **Attachment 3**. The N-value had values between 2 and 4 with steps of 0.1 and the measured power had values higher than -100 dBm and lower than 0 dBm with steps of 1 dBm, such as -80, -60, -22, or any value within the given range.

The higher the RSSI value, the better it is. For example, a value of -53 dBm means a stronger signal strength (better) than a value of -70 dBm. A RSSI value of -100 dBm indicates no signal at all. [10]

These values are tested with a 1-meter distance and then tested again with different distance values for accuracy.

## 6. Results

After experimenting on the field and indoors, some conclusions could be made:

- C1: The back of the tablet must face the tag unobstructed for the most consistent results. Otherwise, they will fluctuate quite a lot.
- C2: Even with the best conditions (unobstructed and the BLE module of the tablet faces directly towards the tag), the RSSI values are still quite varied.
- C3: The tag’s range may not be as advertised.

Those conclusions are made from the 2 tables of RSSI values, found in **Attachment 2**. These tables are filled in while observing changes in RSSI value after each test app’s successful refresh to update the list within the test app (**Attachment 1**). Thus, only refreshes that show the target device (the tag) are taken into account.

The tables can be understood by knowing the meaning of the row a column. Each column represents RSSI values at different app refreshes. Each row represents RSSI values at varying distances (in meters). Besides the distance, you can also see if the testing table is lying down or standing up by looking at “LU” and “SA” respectively. For example, for an RSSI value at its 4th refresh with a 3-meter distance between the tablet and the tag and the tablet must be lying down, you can search for row “3 m *LU” and column “4”.

The conclusion 1 (C1) was made by observing the tag during the experimentation and looking at the tables. When the tablet was lying down, the BLE module was facing the ground and not the tag, consequently obstructing the signal between the 2 devices. The frequent fluctuations of RSSI values can be seen with distances with “*LU” notation. The ones with “*SA” notation do not fluctuate as much.

Even then, the variations of RSSI value within the same condition are quite troublesome. Imagine that the app should alarm the caregiver should an elderly walked 50 meters from the tablet. The app may not be able to differentiate between 49 and 50 meters. Take a look at the RSSI values of 1 and 2 meters with “*SA” notation. The differences between them are not much, or even if there are any at all. An RSSI value can be -60 at 1 meter, but also be -57 dBm at 2 meters, how does that work? The second conclusion (C2) was made from these results.

At last, when looking at the second table (**Attachment 2 Figure 3**), something might have caught your eyes. The RSSI values for a 20-meter distance ended at the 9th successful refresh. Why is that? Take a look at the whole table first. Which information may not be included? There are many kinds of information that you might want to see in a table like this, but the important ones are:

- The number of (unsuccessful) refreshes until a successful refresh, and
- The time it took for a successful refresh.

The second one is rather unreliable. Sometimes the tester may be distracted and not doing the test. The app could also freeze during the test, affecting the time. However, it still has its value to a certain degree. The number of (unsuccessful) refreshes until a successful one, is however, more reliable to evaluate. The app does not refresh automatically, so the RSSI value cannot be updated until the next refresh. The tester can observe the values whenever it is possible without significantly affecting the result. The number of refreshes only increases when pressing the refresh button.

How reliable or unreliable the 2 information are, they do tell the same thing. Can the tag be scanned and show up in the test app consistently at different distances? For the smaller distances (less than 10m), the number of refreshes was almost always 1 or 2, hence the time between successful refreshes became insignificant. At 10m however, it took on average 5 to 15 refreshes to successfully scan the tag. It translated to 14 to 60 seconds to scan a tag. It is very unreliable to get an alert about an elderly walking out of this range because the alert is maybe up to 60 seconds behind. 20 meters was where the consistency was nowhere to be seen. It took between 20 to 50 (or more) refreshes to find the tag, which could be more than 3 minutes. After some time, the tester concluded that the test had given him enough information and stopped the experiment.

The second table shows that at 20 meters, the RSSI values are already at the -90 dBm range. How would this work with 100 meters as advertised? The RSSI value can only go down to -100 dBm until there is no signal. One of the values is already at -97 dBm, this would leave no room for the values at 21 to 100 meters. Following this result, conclusion 3 (C3) was concluded.

This problem may come from the nature of closed-source projects or products. The outsiders do not know the inner workings of the Tile app and the Tile tag, so they cannot make use of the full range of 120 meters [1] as advertised. There is possibly something in the code that no one else than Tile knows, that can use the full range. There is not even a datasheet for the Tile Pro on the internet for the public, so no one can easily study its inner workings.

After these 3 conclusions were made, there was no reason to continue the experimentation, as unfavorable results were expected. However, a smaller-scale test was done with the same C++ code (**Attachment 3**) with an RSSI value of -63 dBm.

At 1 meter, the result (in meters) in **Figure 5** was expected. When the measured power is equal to the RSSI value in Equation 1, the result will always be 1 (meter) no matter which N-value it takes. This is caused by:

- The numerator is always 0 when measured power is equal to the RSSI value:

$$(2)\ measured\ power - RSSI\ value = 0$$

- When the numerator is 0, the result of the fraction will always be 0, except when the denominator is equal to 0. It will then cause the result to be undefined.

$$(3)\ \frac{numerator\ (0)}{denominator\ (10 * N)} = 0\ where\ denominator\ \neq 0\ and\ 2 \leq N \leq 4$$

- When the fraction is equal to 0, everything to the power of 0 is equal to 1.

$$(4)\ Distance = 10^{fraction} = 10^{0} = 1(meter)$$

However, the result has too many N-values. Therefore, they could not be used to reliably calculate the distance. Besides, this value is affected by the environment, such as, where there are a lot of other signals, or somewhere that has signal-reflective objects [3]. These cause the signal to not follow a linear path from tag to tablet. As a result, it is hard to use an N-value in one environment in another. However, it was still tested by adding the code in **Figure 6** into the main() function of **Figure 4** in **Attachment 3**. Its result is in **Figure 7** of the same attachment. The code takes the value of the measured power as -63, the RSSI values of the tester, and the N-value of 3.3 and gives the distance in meters as a result. The result speaks for itself about how reliable it is for determining the distance between the tablet and the tag. At smaller distances, the difference between 2 RSSI values are small and acceptable. However, the tag must be able to be used with bigger distances, and the result is rather unfavorable.

The 2 last distance values in **Figure 7** are 8.69749 and 10.7227 meters, with their respective RSSI values of -90 and -94 dBm. In the table in **Figure 3**, these RSSI values are in the 20-meter row, with the former also appearing in the 10-meter row. These give enough information about the accuracy of the distance determination with RSSI.

## 7. Conclusion/Advice

The result from the experimentation is rather unfavorable for determining the distance between the tag and the Buddy Bot application hosting tablet. The calculated distance at low ranges is reasonably accurate. However, for the use case that the product owner wants (high range), the calculated distance values are rather too inaccurate.

It is also affected by the environment, which is hard to take account of. Each place that uses this system is different. Some places may have a lot of other signals that interfere with the BLE signal. Other places may have some objects that reflect this signal. These cause the signal not to follow a linear path, which decreases the accuracy of the distance determination.

It is advisable not to use a BLE tag to do distance determination. The technology is not yet efficient for such tasks. It is also not advisable to use closed-source tags for your system, as you may not be able to get your hands on the product’s necessary information to develop. Then you have wasted your money on something that you cannot use.

## 8. References

[1] "Pro 1-Pack(Black)," *Tile Store*. https://nl.tile.com/en/product/686590/pro-1-packblack

[2] Abaltatech, "Micro-Location Part 2: BLE and RSSI," *Abalta Technologies, Inc.*, Apr. 05, 2022. https://abaltatech.com/blog/2021/01/microlocation2/

[3] Mia, "Bluetooth RSSI, Cost-effective Distance Determination Technology," *DusunIoT*, Oct. 14, 2023. https://www.dusuniot.com/blog/bluetooth-rssi-positioning/

[4] "What is ultra-wideband (UWB) wireless communication? | Murata Manufacturing Articles," *Murata Manufacturing Articles*. https://article.murata.com/en-eu/article/what-is-uwb-wireless-communication

[5] olivia@webfx.com, "What are RTLS Systems in Healthcare?," *CenTrak*, Jan. 02, 2024. https://centrak.com/resources/blog/rtls-for-hospitals

[6] Mtahca, "Senior Living Safety: RTLS for Senior Care | LITUM," *Indoor Tracking With RTLS - Real Time Location Systems | Litum*, May 29, 2024. https://litum.com/rtls-senior-living-monitoring/

[7] "CenTrak | TruViewTM for Senior Living | Software Solutions," *CenTrak*, Oct. 26, 2023. https://centrak.com/solutions/truview-senior-living

[8] "How to get distance from beacons," *Stack Overflow*. https://stackoverflow.com/questions/65124232/how-to-get-distance-from-beacons

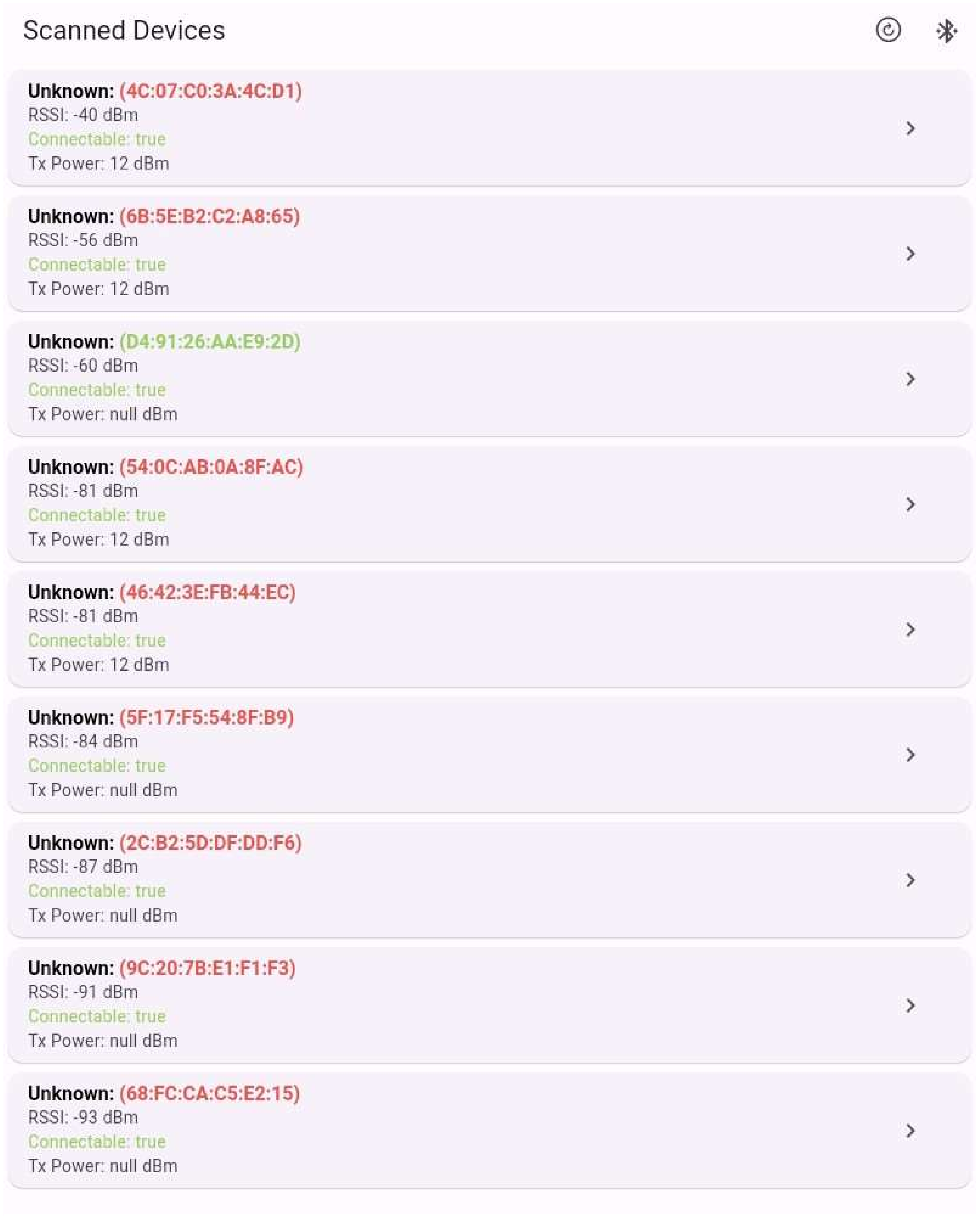
[9] L. Liu, B. Li, L. Yang, and T. Liu, "Real-Time indoor positioning approach using iBeacons and smartphone sensors," *Applied Sciences*, vol. 10, no. 6, p. 2003, Mar. 2020, doi: 10.3390/app10062003.

[10] M. Li, "Understanding the Measures of Bluetooth RSSI - mokoblue," *MOKOBlue: Original Bluetooth/BLE IoT & Smart Devices Manufacturer*, Mar. 16, 2023. https://www.mokoblue.com/measures-of-bluetooth-rssi/

## 9. Change Log

| Version | Date | Changes |
|---|---|---|
| 1 | 16/05/2024 | First version |
| 2 | 30/05/2024 | Changes to default’s format |
| 3 | 02/06/2024 | Updated with latest information |

## Attachment 1


Scanned Devices

Unknown: (4C:07:C0:3A:4C:D1)
RSSI: -40 dBm
Connectable: true
Tx Power: 12 dBm

Unknown: (6B:5E:B2:C2:A8:65)
RSSI: -56 dBm
Connectable: true
Tx Power: 12 dBm

Unknown: (D4:91:26:AA:E9:2D)
RSSI: -60 dBm
Connectable: true
Tx Power: null dBm

Unknown: (54:0C:AB:0A:8F:AC)
RSSI: -81 dBm
Connectable: true
Tx Power: 12 dBm

Unknown: (46:42:3E:FB:44:EC)
RSSI: -81 dBm
Connectable: true
Tx Power: 12 dBm

Unknown: (5F:17:F5:54:8F:B9)
RSSI: -84 dBm
Connectable: true
Tx Power: null dBm

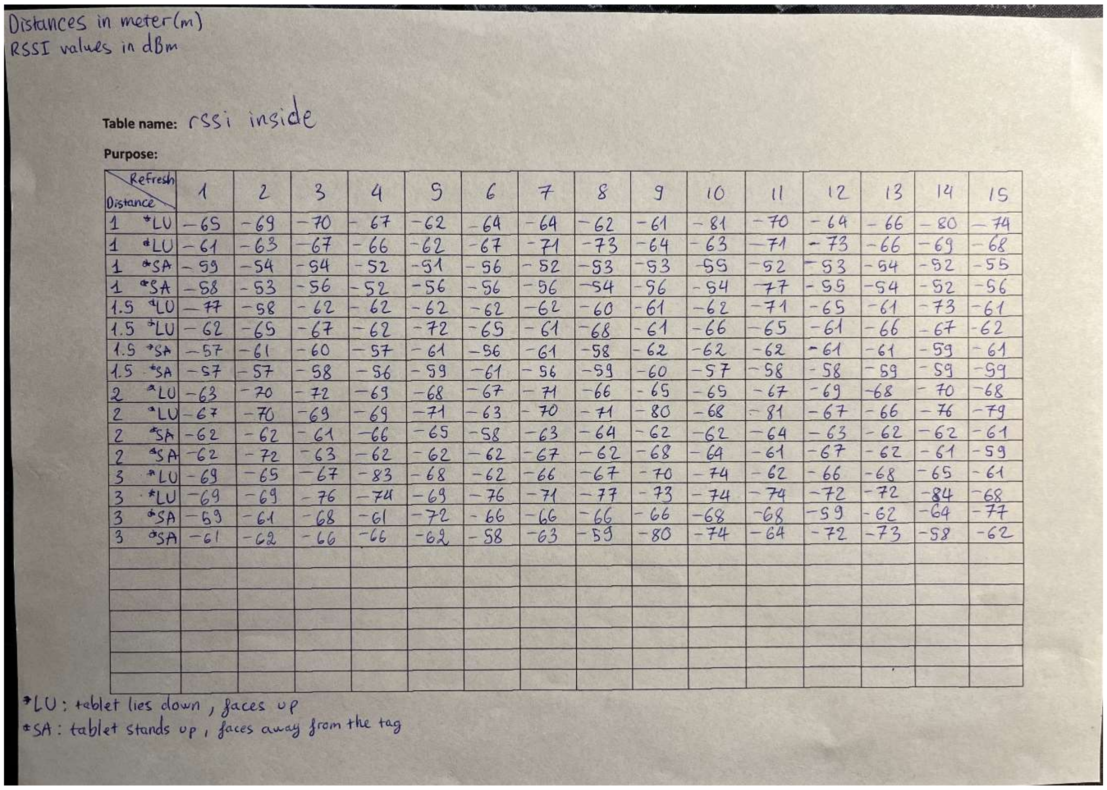
Unknown: (2C:B2:5D:DF:DD:F6)
RSSI: -87 dBm
Connectable: true
Tx Power: null dBm

Unknown: (9C:20:7B:E1:F1:F3)
RSSI: -91 dBm
Connectable: true
Tx Power: null dBm

Unknown: (68:FC:CA:C5:E2:15)
RSSI: -93 dBm
Connectable: true
Tx Power: null dBm

*Figure 1: Test app's interface that was used during the observation. The tag that was used during testing can be found highighted in green (3rd device counting down).*

## Attachment 2



Distances in meter(m)
RSSI values in dBm

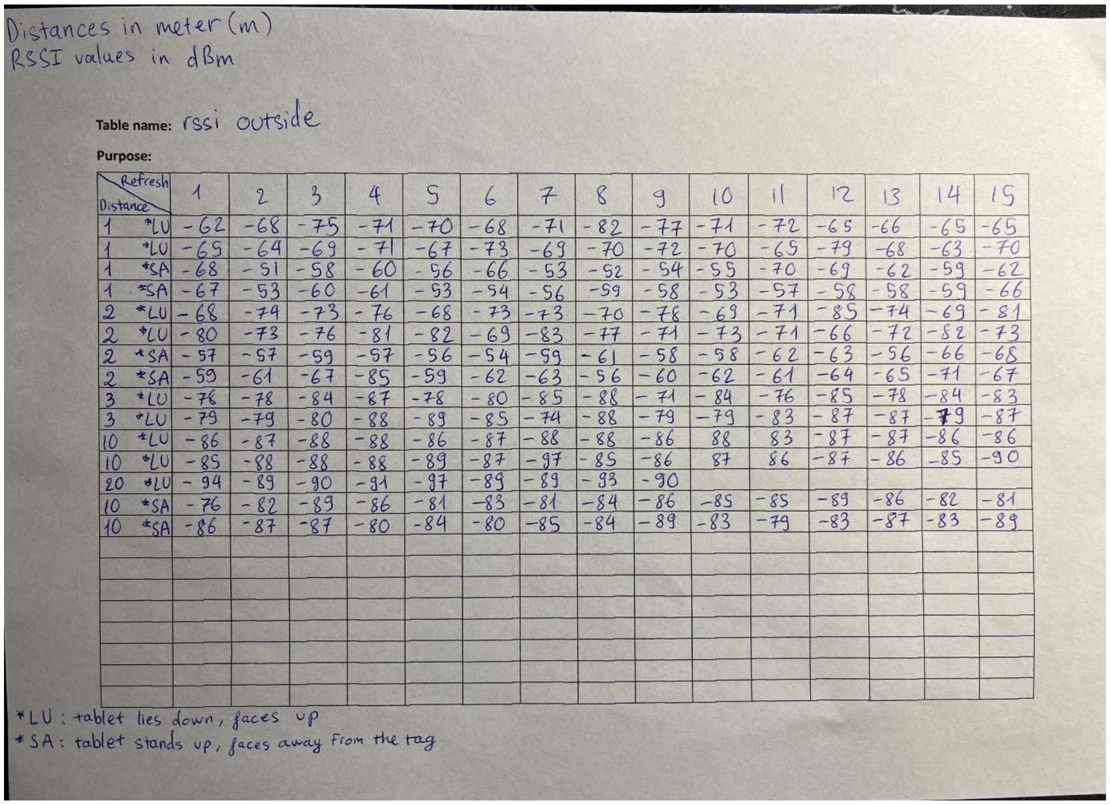
**Table name:** rssi inside

**Purpose:**

| Distance \ Refresh | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 *LU | -65 | -69 | -70 | -67 | -62 | -64 | -64 | -62 | -61 | -81 | -70 | -64 | -66 | -80 | -74 |
| 1 *LU | -61 | -63 | -67 | -66 | -62 | -67 | -71 | -73 | -64 | -63 | -71 | -73 | -66 | -69 | -68 |
| 1 *SA | -59 | -54 | -54 | -52 | -51 | -56 | -52 | -53 | -53 | -55 | -52 | -53 | -54 | -52 | -55 |
| 1 *SA | -58 | -53 | -56 | -52 | -56 | -56 | -56 | -54 | -56 | -54 | -77 | -55 | -54 | -52 | -56 |
| 1.5 *LU | -77 | -58 | -62 | -62 | -62 | -62 | -62 | -60 | -61 | -62 | -71 | -65 | -61 | -73 | -61 |
| 1.5 *LU | -62 | -65 | -67 | -62 | -72 | -65 | -61 | -68 | -61 | -66 | -65 | -61 | -66 | -67 | -62 |
| 1.5 *SA | -57 | -61 | -60 | -57 | -61 | -56 | -61 | -58 | -62 | -62 | -62 | -61 | -61 | -59 | -61 |
| 1.5 *SA | -57 | -57 | -58 | -56 | -59 | -61 | -56 | -59 | -60 | -57 | -58 | -58 | -59 | -59 | -59 |
| 2 *LU | -63 | -70 | -72 | -69 | -68 | -67 | -71 | -66 | -65 | -65 | -67 | -69 | -68 | -70 | -68 |
| 2 *LU | -67 | -70 | -69 | -69 | -71 | -63 | -70 | -71 | -80 | -68 | -81 | -67 | -66 | -76 | -79 |
| 2 *SA | -62 | -62 | -61 | -66 | -65 | -58 | -63 | -64 | -62 | -62 | -64 | -63 | -62 | -62 | -61 |
| 2 *SA | -62 | -72 | -63 | -62 | -62 | -62 | -67 | -62 | -68 | -64 | -61 | -67 | -62 | -61 | -59 |
| 3 *LU | -69 | -65 | -67 | -83 | -68 | -62 | -66 | -67 | -70 | -74 | -62 | -66 | -68 | -65 | -61 |
| 3 *LU | -69 | -69 | -76 | -74 | -69 | -76 | -71 | -77 | -73 | -74 | -74 | -72 | -72 | -84 | -68 |
| 3 *SA | -59 | -61 | -68 | -61 | -72 | -66 | -66 | -66 | -66 | -68 | -68 | -59 | -62 | -64 | -77 |
| 3 *SA | -61 | -62 | -66 | -66 | -62 | -58 | -63 | -59 | -80 | -74 | -64 | -72 | -73 | -58 | -62 |

*LU: tablet lies down, faces up
*SA: tablet stands up, faces away from the tag

*Figure 2: RSSI values table that was filled indoors.*



Distances in meter (m)
RSSI values in dBm

**Table name:** rssi outside

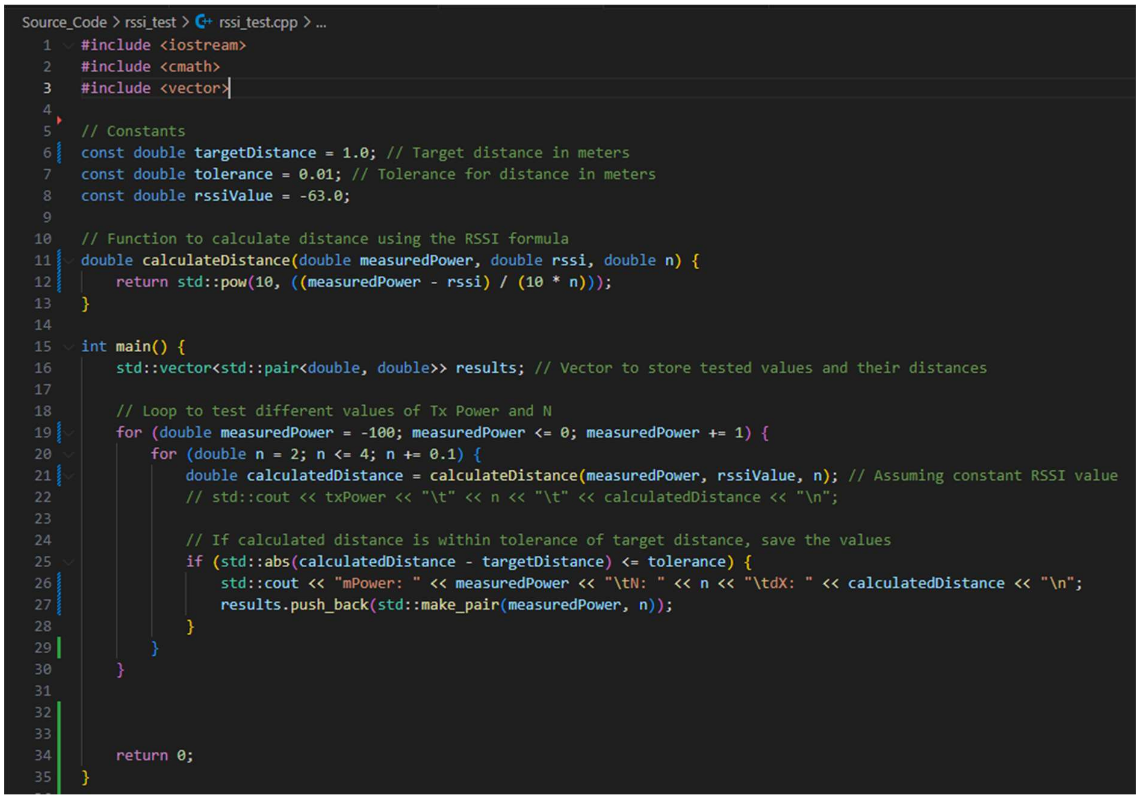
**Purpose:**

| Distance \ Refresh | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 *LU | -62 | -68 | -75 | -71 | -70 | -68 | -71 | -82 | -77 | -71 | -72 | -65 | -66 | -65 | -65 |
| 1 *LU | -65 | -64 | -69 | -71 | -67 | -73 | -69 | -70 | -72 | -70 | -65 | -79 | -68 | -63 | -70 |
| 1 *SA | -68 | -51 | -58 | -60 | -56 | -66 | -53 | -52 | -54 | -55 | -70 | -69 | -62 | -59 | -62 |
| 1 *SA | -67 | -53 | -60 | -61 | -53 | -54 | -56 | -59 | -58 | -53 | -57 | -58 | -58 | -59 | -66 |
| 2 *LU | -68 | -74 | -73 | -76 | -68 | -73 | -73 | -70 | -78 | -69 | -71 | -85 | -74 | -69 | -81 |
| 2 *LU | -80 | -73 | -76 | -81 | -82 | -69 | -83 | -77 | -71 | -73 | -71 | -66 | -72 | -82 | -73 |
| 2 *SA | -57 | -57 | -59 | -57 | -56 | -54 | -59 | -61 | -58 | -58 | -62 | -63 | -56 | -66 | -68 |
| 2 *SA | -59 | -61 | -67 | -85 | -59 | -62 | -63 | -56 | -60 | -62 | -61 | -64 | -65 | -71 | -67 |
| 3 *LU | -76 | -78 | -84 | -87 | -78 | -80 | -85 | -88 | -71 | -84 | -76 | -85 | -78 | -84 | -83 |
| 3 *LU | -79 | -79 | -80 | -88 | -89 | -85 | -74 | -88 | -79 | -79 | -83 | -87 | -87 | -79 | -87 |
| 10 *LU | -86 | -87 | -88 | -88 | -86 | -87 | -88 | -88 | -86 | 88 | 83 | -87 | -87 | -86 | -86 |
| 10 *LU | -85 | -88 | -88 | -88 | -89 | -87 | -97 | -85 | -86 | 87 | 86 | -87 | -86 | -85 | -90 |
| 20 *LU | -94 | -89 | -90 | -91 | -97 | -89 | -89 | -93 | -90 | | | | | | |
| 10 *SA | -76 | -82 | -89 | -86 | -81 | -83 | -81 | -84 | -86 | -85 | -85 | -89 | -86 | -82 | -81 |
| 10 *SA | -86 | -87 | -87 | -80 | -84 | -80 | -85 | -84 | -89 | -83 | -79 | -83 | -87 | -83 | -89 |

*LU : tablet lies down, faces up
*SA : tablet stands up, faces away from the tag

Figure 3: RSSI values table that was filled outdoors.

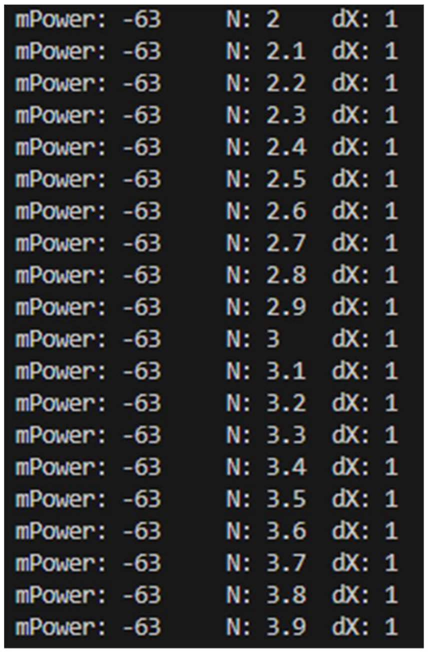
## Attachment 3



Source_Code > rssi_test > rssi_test.cpp > ...

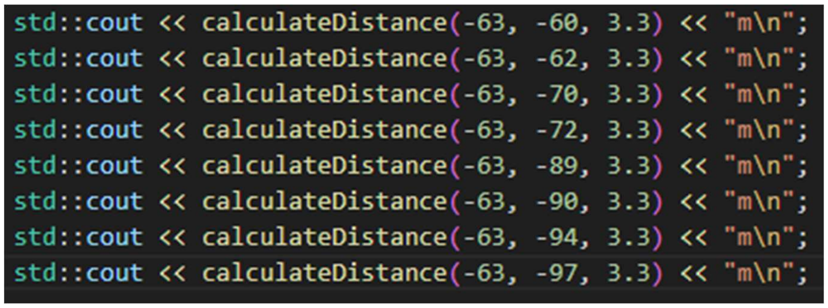
```cpp
#include <iostream>
#include <cmath>
#include <vector>

// Constants
const double targetDistance = 1.0; // Target distance in meters
const double tolerance = 0.01; // Tolerance for distance in meters
const double rssiValue = -63.0;

// Function to calculate distance using the RSSI formula
double calculateDistance(double measuredPower, double rssi, double n) {
    return std::pow(10, ((measuredPower - rssi) / (10 * n)));
}

int main() {
    std::vector<std::pair<double, double>> results; // Vector to store tested values and their distances

    // Loop to test different values of Tx Power and N
    for (double measuredPower = -100; measuredPower <= 0; measuredPower += 1) {
        for (double n = 2; n <= 4; n += 0.1) {
            double calculatedDistance = calculateDistance(measuredPower, rssiValue, n); // Assuming constant RSSI value
            // std::cout << txPower << "\t" << n << "\t" << calculatedDistance << "\n";

            // If calculated distance is within tolerance of target distance, save the values
            if (std::abs(calculatedDistance - targetDistance) <= tolerance) {
                std::cout << "mPower: " << measuredPower << "\tN: " << n << "\tdX: " << calculatedDistance << "\n";
                results.push_back(std::make_pair(measuredPower, n));
            }
        }
    }



    return 0;
}
```

*Figure 4:C++ code to calculate measured power and N-value.*



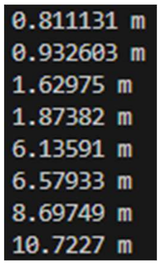
```
mPower: -63     N: 2    dX: 1
mPower: -63     N: 2.1  dX: 1
mPower: -63     N: 2.2  dX: 1
mPower: -63     N: 2.3  dX: 1
mPower: -63     N: 2.4  dX: 1
mPower: -63     N: 2.5  dX: 1
mPower: -63     N: 2.6  dX: 1
mPower: -63     N: 2.7  dX: 1
mPower: -63     N: 2.8  dX: 1
mPower: -63     N: 2.9  dX: 1
mPower: -63     N: 3    dX: 1
mPower: -63     N: 3.1  dX: 1
mPower: -63     N: 3.2  dX: 1
mPower: -63     N: 3.3  dX: 1
mPower: -63     N: 3.4  dX: 1
mPower: -63     N: 3.5  dX: 1
mPower: -63     N: 3.6  dX: 1
mPower: -63     N: 3.7  dX: 1
mPower: -63     N: 3.8  dX: 1
mPower: -63     N: 3.9  dX: 1
```

*Figure 5: Result from Figure 4*



```
std::cout << calculateDistance(-63, -60, 3.3) << "m\n";
std::cout << calculateDistance(-63, -62, 3.3) << "m\n";
std::cout << calculateDistance(-63, -70, 3.3) << "m\n";
std::cout << calculateDistance(-63, -72, 3.3) << "m\n";
std::cout << calculateDistance(-63, -89, 3.3) << "m\n";
std::cout << calculateDistance(-63, -90, 3.3) << "m\n";
std::cout << calculateDistance(-63, -94, 3.3) << "m\n";
std::cout << calculateDistance(-63, -97, 3.3) << "m\n";
```

*Figure 6: Added to main to calculate distances at different RSSI values*



```
0.811131 m
0.932603 m
1.62975 m
1.87382 m
6.13591 m
6.57933 m
8.69749 m
10.7227 m
```

*Figure 7: Result from Figure 6*