UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Anton Slavin

Delta Wi-Fi Positioning System

Bachelor's Thesis (9 ECTS)

Supervisor: Danielle Melissa Morgan, MSc

Tartu 2022

Delta Wi-Fi Positioning System

Abstract:
Many solutions exist for positioning users inside buildings, where access to satellites is limited. Conventional building materials, such as concrete and steel greatly reduce reception of satellites, creating the requirement for other methods of positioning.
The objective of this paper is to analyse the available solutions for indoor positioning, choose a suitable solution and implement an interactive application with a Graphical User Interface for use in the Delta building. The finished application is to provide a visual overview of a user's location on a floor map of the Delta building, approximate position of the user expressed in pixels and an accuracy metric of the calculated prediction.
While similar solutions have already been researched for use in the Delta building, no previous papers used weighted mean and trilateration methods as the positioning algorithm choice and none created an interactive map-based GUI as the application, with coordinates for the predicted user's position.
As a result of this thesis, a multi-platform application was successfully developed in Python for use in the Delta building. Using detailed floor maps and a list of routers located throughout the Delta building, the user's location can be predicted with adequate accuracy using weighted mean and trilateration algorithms. Further, the finished application contains additional functionality and an interactive tool to add new routers to the database. Despite the adequate accuracy of the final application, more advanced implementations of the positioning methods should be considered to further increase the accuracy. Further, additional ways to fine-tune variables used in the application should be discovered. Finally, despite the multi-platform support, not all platforms and adapters give optimal results due to variability in used hardware components and built-in software limitations.

Keywords:
Wi-Fi, indoor positioning, trilateration, weighted mean

CERCS:
P170 Computer science, numerical analysis, systems, control
P175 Informatics, systems theory

Delta Wi-Fi Positsioneerimise Süsteem

Lühikokkuvõte:
Inimese asukoha määramiseks hoone sees on olemas mitu lahendust. Traditsioonilised ehitusmaterjalid nagu betoon ja teras oluliselt vähendavad satelliitide signaali tugevust, seega on vajadus alternatiivsetele positsioneerimise meetoditele.
Selle töö eesmärk on uurida, millised on laialt kasutatavad lahendused positsioneerimiseks hoone sees, nendest valida kõige sobivam lahendus ning luua interaktiivne rakendus Delta hoone jaoks. Rakendus peab sisaldama hoone korruseplaane, mille peale kujutatakse kasutaja ennustatud asukoht.
Kuigi sarnased lahendused on juba uuritud Delta hoone jaoks, ükski eelmine töö ei kasutanud kaalutud keskmise ja triangulatsiooni meetodeid positsioneerimiseks ja ükski töö ei valmistanud interaktiivset rakendust hoone plaaniga, kus kasutaja asukoht on ennustatud koordinaatides.
Selle töö tulemusena on ehitatud rakendus keeles *Python* kasutamiseks Delta hoone sees. Rakendus ennustab kasutaja asukohta piisavalt täpselt hoone korruseplaanide, ruuterite andmebaasi ning kaalutud keskmise ja trilateratsiooni meetodite abil. Lisaks, valminud rakendus sisaldab lisafunktsioone ja interaktiivset vahendit uute ruuterite salvestamiseks andmebaasi.
Kuigi rakenduse ennustuste täpsus on piisav kasutamiseks Delta hoones, paremate tulemuste saavutamiseks võib rakendada ka tõsisemaid positsioneerimise meetodeid. Lisaks, tuleb leida rohkem võtteid optimeerida rakenduses kasutatud valemite muutujad. Lõpuks, kuigi rakendus töötab mitmel erineval platvormil, mõnedel süsteemidel pole tulemused sama optimaalsed, mis on tingitud riistvara ja sisseehitatud tarkvara erinevustest.

Võtmesõnad:
Wi-Fi, positsioneerimine, trilateratsioon, kaalutud keskmine

CERCS:
P170 Arvutiteadus, arvanalüüs, süsteemid, kontroll
P175 Informaatika, süsteemiteooria

Contents

Introduction

The Global Navigation Satellite System (GNSS), such as GPS and Galileo, is a widely used positioning solution around the globe, in a myriad of different environments outside [22]. The system relies on using multiple satellites with highly precise clocks to calculate a user's position. Despite the wide availability of GNSS in outdoor environments, the use of GNSS inside buildings is often unfeasible due to the poor propagation of signals through conventional building materials, such as concrete and steel.

Indoor positioning is beneficial in tracking users for safety purposes, marketing, statistics and overall surveillance [21]. For example, visitors who require immediate assistance can be quickly located using indoor positioning in a large building. The load of different rooms and areas can be measured to help with overcrowding, and users can use interactive applications as guides to find desired rooms quicker.

Thus, in order to perform positioning indoors, alternative methods must be considered. Some of these methods are fingerprinting, triangulation, trilateration, video cameras and video processing. The effectiveness of each method depends on a number of different factors, which are explored further in this paper.

The goal of this paper is to analyse and introduce the current state of the art methods for indoor positioning, choose a suitable method for use in the Delta building and implement the method in an application with a GUI. Despite indoor positioning inside the Delta building being already explored by G. Illus, the main differences come from the choice of positioning algorithms and the final GUI application design [19]. In this paper, weighted mean and trilateration methods are used instead of fingerprinting, and an auto-updating GUI with detailed maps of the Delta building is created for the application. Additionally, the implemented methods predict the user's location to a higher degree of accuracy in the form of coordinates and floor number. Thus, this thesis presents a unique solution for the Delta building.

The remainder of this paper adheres to the following structure: section 1 contains an analysis of commonly used positioning methods and a description of the chosen methods. Section 2 describes the application creation process in technical detail. Section 3 presents the results of the final application, including the accuracy and discussion. And the final section 4 concludes this paper. In the appendix section, a sample of the routers database file, snippets of scanning and rendering functions and the output from network scanning tools are provided.

Terms and Definitions

AP *Access Point* - A device that communicates with other wireless nodes [13]. In this paper, a router which communicates with laptops, smartphones and other devices.

GPS *Global Positioning System* - Space-based radio navigation system [8].

GNSS *Global Navigation Satellite System* - A system of satellites that transmit positioning data to receivers on Earth [10].

GUI *Graphical User Interface* - A computer interface that consists of visual elements other than just text [13].

MAC address *Media Access Control address* - A unique code for identifying hardware devices used for network communication [13].

RFID *Radio Frequency Identification* - A wireless system with tags that emit radio waves to communicate information to nearby readers [7].

RSSI *Received Signal Strength Indication* - A measurement of received signal power level between the Access Point and receiver (user) [23].

SSID *Service Set Identifier* - The unique name of a wireless network [9].

1 Positioning Algorithms

In this section, the most commonly used solutions for indoor positioning are introduced and compared by their cost of implementation, accuracy and complexity. Further, chosen algorithms for the application are described in more detail.

1.1 Fingerprinting

Using the fingerprinting method, a wireless network adapter[1] is used to make multiple measurements of Received Signal Strength Indication (RSSI) of Wi-Fi access points (AP) throughout the building. The measurements are then added to a database. When locating the user, a wireless network adapter is used to take measurements of Wi-Fi signals in the area [12]. These measurements are then compared to stored fingerprint values in the database. The closest found fingerprints indicate the likely position. An overview of this process is seen in Figure 1. After collecting multiple measurements at fixed locations, a live measurement of networks is matched with the database to find the most probable location as the predicted user's position.

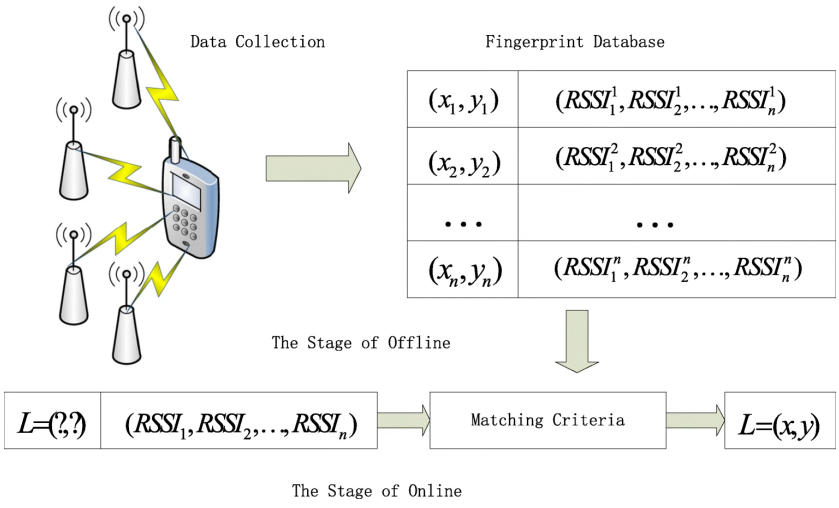

Figure 1. Fingerprinting positioning overview [15].

According to Bullmann et al (2020), the accuracy of this method depends on the layout and materials of the building, as well as the quality of the Wi-Fi adapter when

[1] In this paper, a wireless network adapter, either external or built into computers and other portable devices, is a device used to receive nearby Wi-Fi networks and send signals.

making measurements [12]. Additionally, the database of measurements must cover a large enough area of the building in order to be effective, requiring many measurements to be made, possibly multiple times.

1.2 Triangulation and trilateration

Triangulation methods for indoor positioning include angulation and lateration. According to El Ashry and Sheta (2019), angulation methods use the angles between APs and the user to calculate the user's location [17]. However, these methods require the use of an elaborate antenna setup to get precise measurements, making them more expensive and difficult to use in comparison to lateration methods [16]. As shown in Figure 2, two APs are located at some known points. Using the known angles α and β and distances between the APs, it is possible to estimate the user's location.

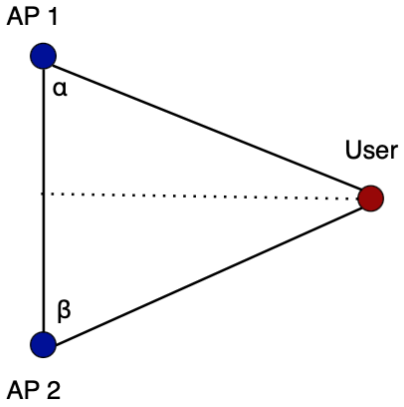

Figure 2. Triangulation with two fixed points at known locations.

In contrast, lateration methods use RSSI measurements, which are further converted into distances using formulae, or signal travel time in conjunction with mathematical modelling to calculate the user's location [16]. The solution can be found by solving a set of equations, or by approaching it as an optimisation problem. Thus, lateration methods do not require elaborate setups and can be performed using any Wi-Fi-compatible device. However, the accuracy of prediction using RSSI measurements depends on the layout of the building and the conversion formula used. The formulae to convert RSSI values to distances are commonly log-based [12], meaning that their accuracy fluctuates heavily and might require fine-tuning of variables to further improve the result.

1.3 Proximity

Proximity methods use Radio Frequency Identification (RFID) tags and Bluetooth devices as sensors to detect the user in proximity to them [16]. This method requires many sensors to be located throughout the building to be effective, as any large areas without sensors will not be included in the predictions made by the algorithm. In comparison to other positioning methods, the installation and use of proximity positioning methods may be more costly due to the additional hardware requirements.

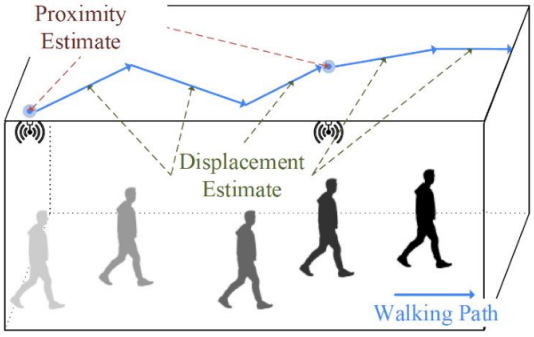

Figure 3. Proximity positioning visualisation [20].

In Figure 3 we can see an example of proximity positioning. As the user progresses through the environment, sensors along the walking path get triggered. Additionally, displacement is calculated to further improve accuracy using this method.

1.4 Computer Vision

According to Mendoza-Silva et al (2019), computer vision-based techniques use cameras located throughout the building in precisely known locations. The user can be identified using the camera's video feed [22]. Thus, this method could be mostly suitable for owners of the building or security personnel.

Images from video feeds can also be processed programmatically to provide more details about the user and to gather information such as movement speed.

Similarly to proximity-based positioning methods, computer vision methods require additional hardware to be placed throughout the building, but only if solutions such as security cameras or motion detectors do not exist or are not available beforehand [22]. Thus, the cost of this method is higher in comparison to other methods, but depends on the price of equipment used in the building.

1.5 Dead Reckoning

According to Maghdid et al (2019), the Dead Reckoning (DR) technique uses the user's smartphone sensors (gyroscope and accelerometer)[2] to calculate the user's offset from some fixed reference point in the building [21]. The accuracy of DR depends entirely on the accuracy of the sensors inside the user's smartphone, if they are present at all. To improve the accuracy and reduce drift, Maghdid et al. (2019) propose an approach to DR using additional real-time calibration and inclusion of RSSI values in the algorithm. Using a device with well-calibrated sensors and this enhanced algorithm, relatively high accuracy can be achieved.

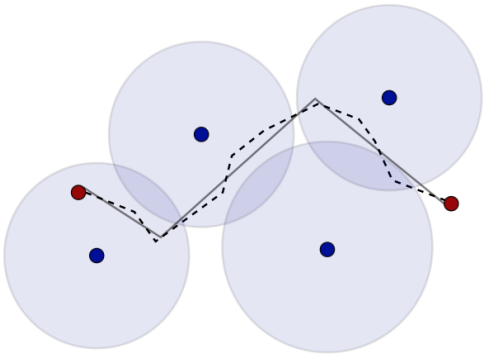

Figure 4. Dead reckoning positioning method.

In the Figure 4 above, we can see an example of DR positioning. Red dots represent the start and end locations of the user while the blue dots represent the locations of the APs. The dotted line represents the estimated path the user took with errors, as calculated by the DR algorithm. Finally, the straight line is the closest actual path.

1.6 Algorithm Choice

A comparison of the introduced positioning methods is available in Table 1. Values under "Cost" depend on the amount of equipment required, with Low being around 20 to 200 eur and High exceeding 2000 eur; "multi-step" shows if additional measurements are required prior to using the system, sometimes called the "offline phase" in literature; "accuracy" represents the ease of obtaining accurate predictions.

[2] A gyroscope detects changes in orientation and an accelerometer measures speed changes [2] [1].

Based on the commonly used methods listed in sections above, the trilateration method was chosen for implementation. The low cost, no requirement of special equipment and adequate accuracy provide the right balance for the scope of this thesis. Further, in comparison to methods like fingerprinting, trilateration does not require SQL databases to function, which further simplifies implementation.

Table 1. Comparison of indoor positioning methods.

Method	**Cost**	**Equipment needed**	**Multi-step**	**Accuracy**
Fingerprinting	Low	Scanner	Yes	Low
Triangulation	Medium	Directional antennae	No	High
Trilateration	Low	Adapter (usually built-in)	No	Medium
Proximity	Medium	RFID & Bluetooth sensors	Yes	Medium
Computer Vision	High	Cameras	No	Low
Dead Reckoning	Medium	Smartphone sensors	Yes	High

True-range multilateration (or trilateration) is a positioning method [24]. In practice, wireless transceivers, such as APs, are placed at fixed positions. A wireless receiver, such as a laptop, listens for signals from the placed APs and calculates RSSI.

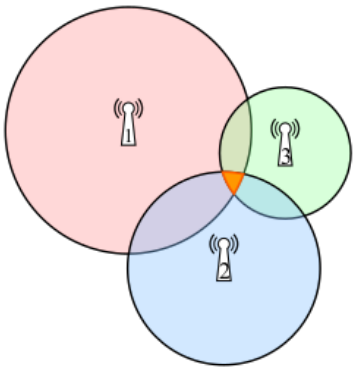

Figure 5. Trilateration with three routers [14].

The RSSI values are converted into distances between the receiver and APs. A circle is then formed around each AP using the calculated distances as radii. The intersection of

these circles represents the predicted location of the user [24]. A visualisation of such an algorithm can be seen in Figure 5, which contains three circles, representing the signals from three routers, with the intersection (in orange) representing the possible position of the user. The radius of each circle is calculated from a corresponding RSSI value. The larger the circle, the further the receiver is from the AP, and vice versa.

Additionally, due to its simple nature and accurate results, a second algorithm called *weighted mean* was chosen despite not being widely mentioned in literature. For this thesis, a weighted mean algorithm is one where the signal received from an AP is considered a weight and the user's location is determined from the mean of these weights. Each weight corresponds to the AP's RSSI value. Thus, stronger values influence the predicted position more.

All nearby APs are considered to form a single shape, the midpoint of which is then found by taking a weighted average of nearby APs coordinates. The RSSI value of each nearby AP is used as the weight, so stronger values influence the position of the midpoint more. An example of this algorithm can be seen in Figure 6. Here, the blue dots represent APs. The red dot is the predicted user's position, which is shifted towards the stronger signal. The numbers above the APs represent their RSSI values, with values closer to 0 meaning a stronger signal. Thus, in contrast to true-range multilateration, the larger the circle is around an AP, the closer the predicted position will be to it.

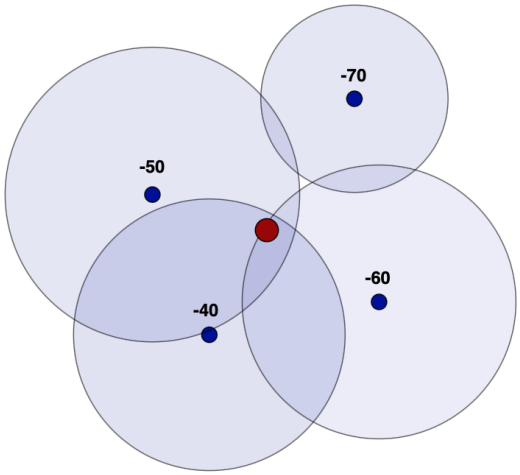

Figure 6. Weighted mean positioning.

As seen during testing, this method proved to be more reliable, as true-range trilateration requires signals from at least 3 or more APs to give a result, but the weighted mean

method can already function with a single signal. Most importantly, note the distinction between an AP and a signal. The APs in Delta (installed by the Institute of Computer Science for all students and staff) produce Wi-Fi signals on both the 2.4 GHz and 5 GHz frequency as well as have two different SSIDs (usually *ut-public* and *eduroam*). Therefore, if the adapter detects the 3 signals but 2 are from the same AP, true-range trilateration will fail. This is in contrast to the weighted mean method, where multiple signals from the same AP further increase the AP's weight.

The most crucial thing in using the weighted mean and trilateration algorithms is the conversion formula between RSSI (signal strength in dBm) and distance (in meters or pixels). As the main weakness of both these algorithms, the formula lacks accuracy, especially at very close and far distances. Multiple variables, such as the path loss exponent and router power[3], play a crucial role in determining the accuracy of the RSSI-to-distance conversion formula and must be tuned for different areas in the building.

It should also be noted that in this thesis, RSSI values are handled exclusively in the dBm (decibel-milliwatts) unit. Usually, RSSI is not a fixed and consistent metric. Often in literature, RSSI values range from 0 to 255 or 0 to 100 or -100 to 100. However, in this paper the value of -30 is near the maximum output for a router, and a value of -100 is near the absolute minimum, as verified with testing inside the Delta building. Thus, any further mention of RSSI should be considered to be in the dBm unit.

[3]In this case, the path loss exponent is used for describing the loss of signal power through walls and other materials, and the router power variable represents the power output of the APs.

2 Program Design and Technology

The chosen weighted mean and trilateration positioning methods require a database of routers, a detailed map, scanning of nearby wireless networks and coding to implement the positioning algorithms and make the application responsive. The sections below contain descriptions of the technologies used in building the application for this thesis, as well as the advantages of using these technologies over their alternatives.

2.1 Data Gathering

For the chosen positioning methods to function, a database of uniquely identifiable routers and their precise locations within a building needs to be constructed. Further, accurate maps of the Delta building are required to display the user's predicted position and surrounding routers.

With the generous help of the Delta IT support team, a list of routers within the Delta building with their MAC addresses and corresponding approximate positions on maps were acquired. Also, with the help of the author's supervisor, very precise maps of the Delta building were acquired as well. For every entry in the routers list, its exact coordinates were transferred from the approximate to the precise map using photos of each location on most floors of the building, taken by the author beforehand. This way, the location of most routers could be exactly verified.

After acquiring a list of routers and building the database, manual scanning was done inside the Delta building to verify the entries. As a result, multiple routers had outdated MAC addresses or were moved. These routers had to be manually measured and edited. Additionally, some new routers were also present in the building, which had to be added to the database as well.

2.2 Coordinate System

Coordinate systems are used for describing a position in a space or on a surface. In this thesis, a two-dimensional Cartesian coordinate system is used, with the Delta building floor map as the surface. Using the two-dimensional coordinate system allows positioning algorithms, which rely on calculating distances between fixed points and on the edges of circles, to be implemented and used.

The Delta building floor plans obtained and used for this thesis are represented as 5300x5553 pixel PNG images. The chosen size allows for sufficient details, file size

below 5 Megabytes (MB) for optimal loading time in the GUI, as well as the correct proportion of the plan. Surprisingly, this was the default size of the generated images when converting them from PDF (their default file format) to PNG, as used in the application. Measurements show that 1 meter of distance is equal to 40 pixels on the map, which is confirmed by measuring steps and furniture in the Delta building, as shown in Figure 7. This distance is used in estimating the distances between routers (at fixed points) as well as making predictions of the user's position.

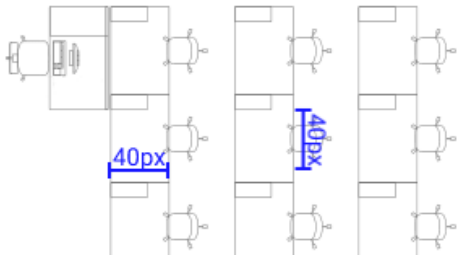

Figure 7. Measurements of 40px on the Delta maps, equal to 1 meter.

As a result, each router in the Delta building was tied to a particular x and y coordinate, floor number, MAC address(es)[4], frequency, and closest location name, such as room number or corridor near a room for the user's convenience.

2.3 Architecture

The following section describes the components and architectural aspects of the positioning application.

The application can be divided into four main states: *Idle*, *Scanning*, *Calculating*, and *Rendering*. During the *Idle* state, the application is ready to accept user input and is only running the GUI cycle. The user is free to explore the map and decide on what to do next during this state, such as initiate a scan. During the *Scanning* state, the application launches a network scanning tool as a subprocess and waits for its input. Next, during the *Calculating* state, the output from the network scan is parsed and processed to estimate the user's location. As a result, predicted coordinates and other information is created. Finally, during the *Rendering* state all elements of the map are drawn onto the screen.

Additionally in the *Idle* state, instead of choosing to perform a scan, the user can also add a new router into the database by choosing its location on the map and entering

[4]Each AP in the Delta building has multiple SSIDs, so each SSID requires a separate MAC address.

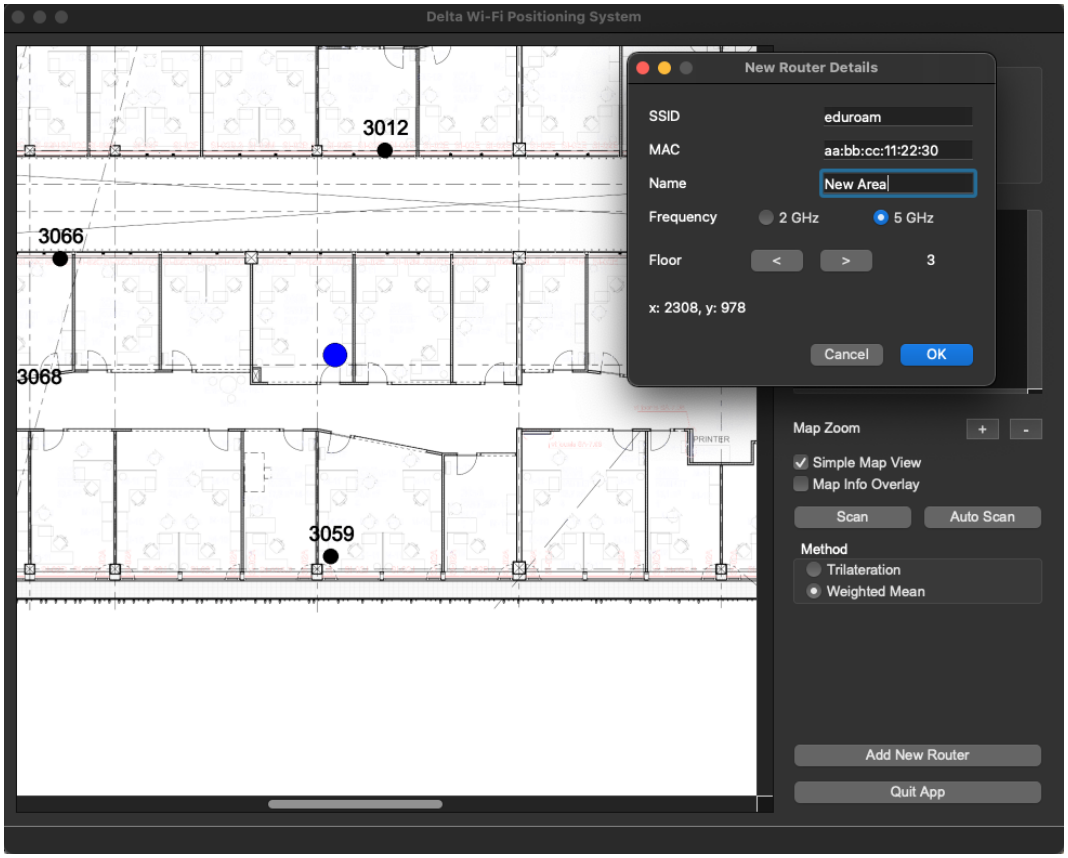

Figure 8. New router popup dialog with the to-be-added router selected.

details into a popup dialog window. After this, the entered details are saved into the database file and re-loaded with the new router included. Figure 8 contains a screenshot of the New Router Mode, with sample details inserted into the popup dialog and a router selected interactively, represented in blue. In case of invalid input, an error will be displayed in the application's status bar and no new data will be inserted into the database file.

2.3.1 Programming Language

Python is a dynamic programming language. This means that written *Python* code does not need compilation and can be run directly from the source code, which increases the speed of development significantly [11].

In this thesis, *Python* is used for the speed of development. Additionally, *Python* has

a wide variety of libraries available, some of which are used in this work. Additional libraries are installed using the *pip* utility, which is bundled with the *Python* distributable. Thus, to install the positioning application, one must install *Python* and run the *pip* utility with a single command, with a special requirements file as the argument (included with the application) [3].

Applications written in *Python* are supported on multiple platforms without additional setup, and it is enough to develop on a single platform only. This fact results in faster development, as only a single computer is required for both development and testing.

In the main script file, `app.py`, a main `Renderer` class was built to hold all code related to storing scanned network information, accessing stored routers database, creating and rendering the main map and displaying other information. A single instance of this class is used throughout the lifecycle of the application.

Additionally, three more script files were created to contain supplementary code. The `components.py` script contains classes for custom UI components used throughout the app, such as a custom Scene class, and a pop-up dialog window. The `locator.py` script contains code related to calculating the user's position, converting RSSI to distance and other functions to aid with these tasks. The `scanner.py` file contains code to identify the user's operating system, launch the corresponding network scanner tool and parse the output.

2.3.2 GUI Framework

Qt is a library for developing Graphical User Interfaces (GUIs). *Qt* is available on multiple platforms and it has bindings for the *Python* language [5]. The name of the binding library is *PySide* and the developed application uses version 6 of *PySide*, as it is the latest version. One advantage of this library is the cost - using *Qt* with the *PySide* library is free, and it is open source.

The main advantage of using *Qt* for this thesis is in the application *Qt Designer*, which is used specifically with *Qt*. *Qt Designer* allows for creating *Qt* interfaces without writing code, using a user-friendly GUI instead [6]. *Qt Designer* outputs a special UI file, which can be used by the target application to load all UI components and layout.

The application consists of a main window and a pop-up dialog window for adding a new router into the database. The main window contains the map view, sidebar information panels and sidebar controls. Sidebar information panels display the predicted user's location coordinates, floor, closest room or area name and a list of nearby detected routers. Using the controls, it is possible to change the map zoom, toggle map detail and

overlay modes, start a scan or an auto-scan, choose the positioning algorithm, toggle "add new router" mode and quit the application. *PySide* components such as `QPainter` and `QPixmap` are used for drawing the map, with simple circles drawn on top to represent locations of the APs and the user.

The function used for rendering is partly available in Appendix III. Render Function Snippets. A screenshot of the main application window can be seen in Section 3.

2.3.3 Storage

In order to store the list of known routers along with their MAC addresses and coordinates, a simple comma-separated values (`.csv`) file solution was used. The file format is easily accessible, readable and modifiable by both users and computers. Further, it is easy to transport the database by copying a single file. A snippet of the router database file used in this paper is available in Appendix I. Sample Routers Database File.

The main database file consists of rows of routers with a specific ordering of values, which can also be defined in the top row of the file:

`x,y,mac,ssid,floor,frequency,name`. Values `x` and `y` stand for the respective x and y coordinates of the router (as detailed in Section 2.2); `mac` represents the unique MAC address of the network; `ssid` stands for the name of the network; `floor` represents the floor number where the router is located in the building; `frequency` stands for the transmitting frequency (2.4 GHz or 5 GHz); `name` stands for the location or area name (as defined in Section 2.2).

Due to the relatively low amount of routers in the Delta building, there are no performance issues with using this file format for storage. As all of the routers are loaded into memory only once after launching the application, there are no speed issues when scanning for networks. However, in the case of buildings with extremely large numbers of routers, it would be more efficient to use an SQL solution for storage.

2.3.4 Configuration File

In order to save configurable variables and support further modification, some settings of the applications are stored in a configuration file called `config.json`. The file contains multiple variables such as file paths, map size parameters, threshold values and other constants used throughout the application. All of these settings are read once on program startup and saved into a dictionary for use in the app. The file is formatted as JSON due to its readability and convenience of use in *Python*.

The list of configurable variables includes `ADAPTER` for the name of wireless adapter used for network scanning, `UI_FILE_PATH` and `ROUTERS_FILE_PATH` for relative paths of the main UI file and routers database file, `IMG_W` and `IMG_H` for the width and height values of the building maps used in the application. Other important variables can be configured such as `DIST_THRESHOLD` which represents how far away a router should be to be included in the scanned list; `RAD_THRESHOLD` for the maximum possible radius inside the building when specifying precision. The `RAD_NORM` constant is also used when calculating the precision and acts as the default radius size; `PX_SCALE` represents the number of pixels in 1 meter of distance, which is 40 by default; `POWER` for the router power constant value and `PATH_LOSS` for the path loss exponent in RSSI to distance conversion formula (as described in Section 1.6). The `RSSI_MIN` value represents the minimum RSSI required for a network to be included in the scan results. Further, `AUTO_SEC` can be used to change the update frequency of the auto-scan mode, in seconds. Finally, `MIN_FLOOR` and `MAX_FLOOR` represent the numbers of the lowest and highest floors used in the application. In this thesis, the basement floor of the Delta building was not included.

2.3.5 Scanning

Scanning is done using built-in command-line tools and is different on each operating system. In the case of Linux it is called `iw`. On macOS it is `airport` and on Windows it is `netsh`. These tools were chosen due to their free and wide availability, ease of use and the option to run in the command line. Each tool outputs a list of nearby routers including relevant information about the network such as RSSI, SSID, and MAC values. An example of the output from each tool can be seen in Appendix IV. Network Scanning Output. Afterwards, the output is formatted as a list of objects (described in Section 2.3.6) and forwarded to methods related to calculating the position.

In order to perform scanning, only the built-in adapter inside the user's laptop is required. External adapters are supported, but their name must be known before use.

One important discovery was that on Windows machines, the output of the `netsh` command displays the RSSI value in percentage instead of dBm. However, according to the Windows App Development documentation, this value can be simply interpolated from the $[0, 100]$ range to $[-100, -50]$ to convert into the dBm unit [4]. This is done using a single function from the SciPy package: `interp1d([0, 100], [-100, -50])`.

As an example of parsing, the function used for scanning on macOS devices is available in Appendix II. MacOS Scan Function.

2.3.6 Locating

After scanning for nearby wireless networks, a list of objects with the values `{MAC address, RSSI value, SSID value}` is returned. Firstly, all router objects with RSSI values below the threshold variable `RSSI_MIN` are discarded. Then, using the MAC address of each object, a full router entry is fetched from the main routers database (dictionary). Further, the RSSI value is converted into a distance value in pixels using the following equation:

$$d = 10^{(A-RSSI)/(10 \cdot n)}, \tag{1}$$

where d is the converted distance in meters, A is the router power variable, $RSSI$ is the RSSI signal strength variable and n is the power loss exponent variable, derived from the formulae (1) and (2) by Ilci et al. [18]. In their work and other literature, the router power variable is also called "received signal strength in a meter's distance", and through testing, a default value of 1.68 was found to be the most suitable for the Delta building. Similarly, the power loss exponent is called the "signal propagation constant", and a value of 2.25 was chosen as the default for Delta. These values were found by incrementing them from a base value of 1 and comparing the precision radius results.

The resultant distance value is then further converted into pixels: d/p, where p is the pixel scale variable, equal to the number of pixels in one meter of distance (as previously mentioned, the default value for maps in the Delta building is 40).

For every router object in the nearby routers list, more filtering is done to exclude routers too far away (based on a configurable threshold value, `DIST_THRESHOLD`). When the weighted mean method is selected, a weight is assigned to each router based on its RSSI value. After testing multiple approaches, $1/RSSI$ was chosen as the weight formula by performing the best in all conditions[5]. This way, the routers with higher RSSI values (closer to the receiver) influence the final position more, and distant routers have a lesser influence. The weighted mean point is calculated using a function from the numpy package: `np.average(locations, axis=0, weights=weights)`, where `locations` is the list of nearby routers, `axis=0` ensures that both coordinates are considered for each router, and `weights=weights` forwards the corresponding list of weights of each router (relative to the router's RSSI value), calculated beforehand.

When trilateration is the selected positioning method, the three closest routers are selected with their estimated distances r_1, r_2, r_3. Using the Pythagorean theorem, the

[5] Alternative options such as $RSSI/10$ and $RSSI \cdot d$ (where d is distance to router) were considered, but turned out to be inferior to the chosen approach.

three distances can be expressed with the following equations (where x and y are the predicted coordinates):

$$r_1^2 = x^2 + y^2, \tag{2}$$

$$r_2^2 = (x - U_x)^2 + y^2, \tag{3}$$

$$r_3^2 = (x - V_x)^2 + (y - V_y)^2, \tag{4}$$

where U_x is the x coordinate of the second router and V_x, V_y are the x and y coordinates of the third router. From equations (2), (3) and (4), the following equations are derived to calculate the user's position:

$$x = \frac{r_1^2 - r_2^2 + U_x^2}{2 \cdot U_x}, \tag{5}$$

$$y = \frac{r_1^2 - r_3^2 + V_x^2 + V_y^2 - 2 \cdot V_x \cdot x}{2 \cdot V_y}. \tag{6}$$

It is assumed that the first router is located at a point $(0, 0)$, the second router is at a point $(U_x, 0)$ and the third router is at a point (V_x, V_y). Then, the calculated coordinates are offset by the first router's coordinates to finalise the result.

It should also be stated again that this algorithm does not work with less than three APs available nearby. If more than three routers are available, the closest ones are chosen. Including a fourth router in the true-range trilateration equations would not improve accuracy, but instead increase the number of predicted coordinates in dimensions. However, in this paper only a two-dimensional coordinate system is used for each floor.

After obtaining the predicted location, the precision of the location is calculated, called the *precision radius*. The distance of the predicted location from the furthest router is used as a radius to highlight other possible locations of the user in the area. By using the distance to the furthest router, the largest margin of error is assumed. After testing this approach in the Delta building, we confirmed its accuracy to be adequate.

Further, the floor value is calculated by looking at the most common floor value from all nearby routers. After testing we confirmed that this approach is highly accurate, with very rare exceptions, such as highly vertical open spaces and staircases.

Finally, the predicted *nearest location* value is chosen based on the closest router. This value is in the form of a room name (e.g. "Room 1021"), but can be customised for each router in the routers database file.

Calculated metrics are saved into a dictionary, which is then forwarded to the main `Renderer` to begin the drawing process. In case of a failure, an error is displayed in the GUI status bar and the previously known location is kept on screen to avoid confusion.

3 Results

The following chapter describes the results of the finished application, including its methods of testing, accuracy, and a discussion of both strong and weak sides, as well as possible future improvements and additions.

The finished application is available as an installable Python script from its open GitHub repository[6]. In order to install the app, Python 3 must be already installed on the system. After cloning or downloading the repository, the required libraries must be installed with the command `pip install -r requirements.txt`. Then, the application can be launched with the command `python3 app.py`. However, on macOS and Linux systems it is necessary to run the app under sudo privileges, thus the required command is `sudo python3 app.py`. This requirement is due to the command-line tools `iw` and `airport` used for scanning for nearby networks. Additionally, if the operating system is Linux or a custom adapter is used, the name of the adapter must also be set in the configuration file. To accomplish this, line 2 of `config.json` must be changed to include the name of the desired network adapter: `"ADAPTER": "adapter_name_here"`. Otherwise, the scanning will be unsuccessful.

Alternatively, on systems with the *bash* program installed, the included script `start.sh` can be launched, either by double-clicking on the script in a file explorer (if supported), or by running the command `sh start.sh`. The script will ask for a sudo password and launch the app with Python 3 and sudo privileges.

Figure 9 contains a screenshot of the main application window. From the figure, we can see 6 main elements of the GUI:

1. Map view of the Delta building. User's location with a precision radius marked in green, routers that are stored in the database marked as black circles and routers detected during the scan as black circles with red outlines.
2. Location information panel with predicted nearest location name, floor, coordinates and precision radius value.
3. List of nearby routers detected during the scan, the user's distance from them and the last bytes of their MAC address.
4. Map controls including zoom buttons, simple map view toggle and map info overlay toggle, which display more details on the map.

[6] https://github.com/tonysln/delta-wifi-pos

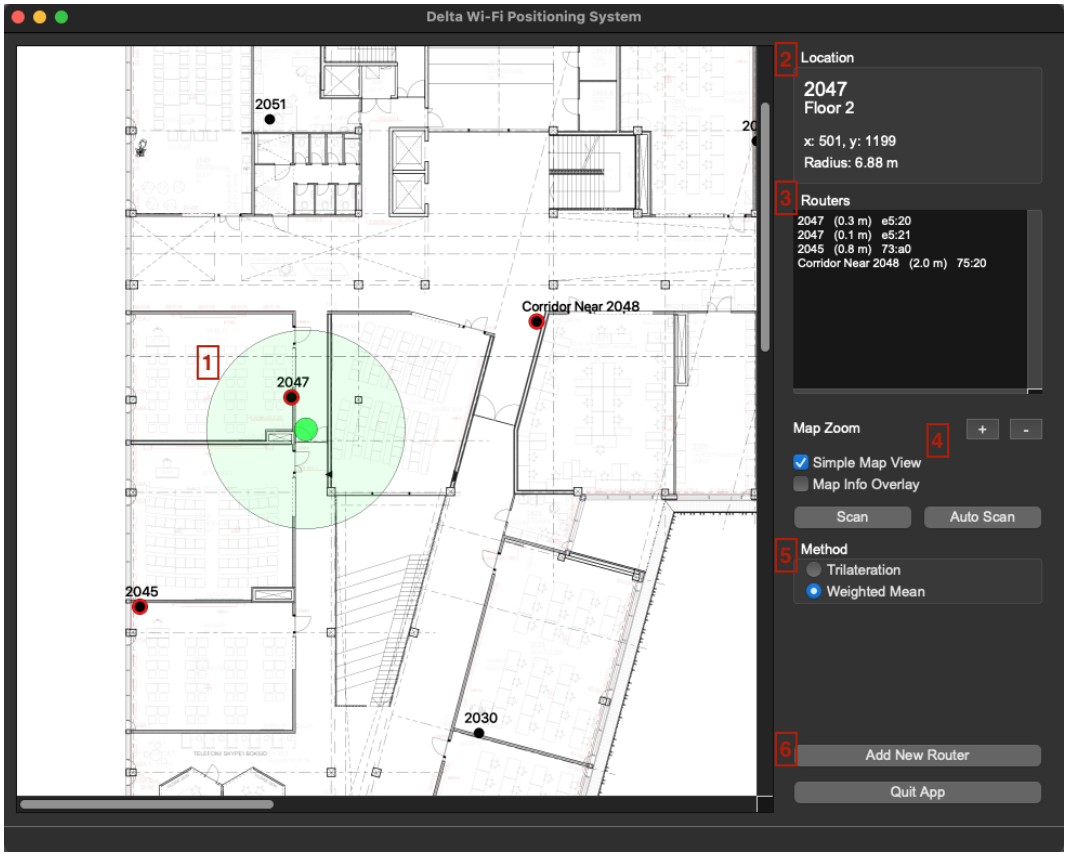

Figure 9. Main window of the finished application.

5. Scan and auto-scan buttons with the positioning algorithm toggle buttons for the user to choose from.
6. Buttons to open the "Add New Router" dialog window to add new routers as seen in Figure 8 and to stop scanning and exit the application.

Finally, the application has an option to display custom overlays on top of the building floor map. The overlays can be customised for each floor separately and can be used for displaying temporary or otherwise important information, such as names for areas, arrows or marked paths to selected areas.

All building floor maps and overlays are located in the `map` folder and are named `korrus-num.png`, where `num` is the floor number (1-4 in the case of the Delta building). Additionally, clean versions of the maps (with less details) are available under the names `korrus-num-c.png`. Finally, map overlays are named `korrus-num-overlay.png`.

3.1 Testing

In order to test the application, a Macbook Pro (2 GHz Intel Core i5 CPU, Broadcom BCM4364 802.11ac adapter) with macOS 12.1 and a HP Elitebook (1.6 GHz Intel Core i5 CPU, Intel(R) Dual Band Wireless-AC 8265 adapter) with Windows 10 and Ubuntu 22.04 were used. For multiple days, walks were made around all floors of the Delta building with the auto-scan mode enabled. The results were evaluated subjectively, as well as recorded to compare more objectively as well.

Special attention was brought to different areas within the building, such as classrooms, auditoriums, corridors, corners and stairs.

3.2 Accuracy

The final accuracy depends heavily on parameters set in the provided configuration file, chipset of the wireless adapter used and area inside of the building (e.g. open area, corridor, office).

An important discovery was that the weighted mean method performs surprisingly better than the trilateration method, but this could be due to the implementation, as alternative ways to implement it still remain. In many cases, the trilateration algorithm implemented in this paper performs much worse than the weighted mean method, especially when multiple signals are scanned from the same router. This is expected, as multiple measurements from a single point do not increase the overall number of routers nearby, of which at least three must be present for trilateration to function properly. This is in contrast to the weighted mean algorithm, where multiple signals from the same router move the predicted position closer to it.

In order to observe the effect of the configuration variables on the accuracy of the algorithm, an experiment was conducted. Two locations were chosen on the second floor of the Delta building, near rooms 2048 and 2047, due to multiple conflicting networks located there. Multiple scans were made and the minimum and maximum precision radius values saved into a text file. Then, the power and path loss variables were modified manually, the program was restarted and the scans done again. The results can be seen in Table 2. As can be observed, increasing path loss and router power variables increases the radius on average for both locations, thus decreasing the accuracy of the prediction.

Further, configuration variables such as `RSSI_MIN` and `DIST_THRESHOLD` play an important role in the final accuracy and should be tested and modified for different systems. As shown in Figure 10, by increasing the default distance threshold value and

Table 2. Comparison of accuracy with different variables used in RSSI conversion formula, using the weighted mean algorithm.

Location	Path loss	Router power	Precision radius
Room 2047	2.25	1.68	3 - 5 m
Room 2047	3.25	1.9	6 - 11 m
Corridor Near 2048	2.25	1.68	3 - 9 m
Corridor Near 2048	3.25	1.9	10 - 20 m

decreasing the minimum RSSI value, more routers can be included in the calculation. However, the overall precision radius also increases, which results in decreased accuracy. Still, this can be beneficial in locations with few routers available.

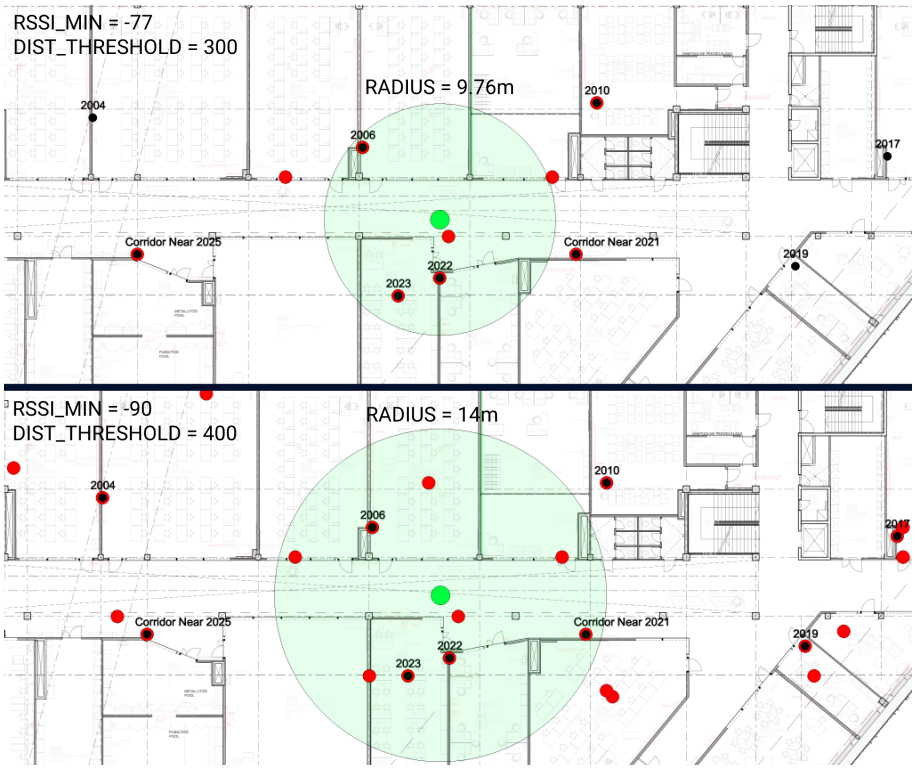

Figure 10. Comparison of results from changing configuration variables (using the weighted mean method).

Note the drastically increased amount of routers from other floors (red circles without black filling) after changing the variables. In some situations, this additional noise can negatively influence the predicted position, so further restricting some of the configuration variables would be a sensible choice. For example, the predicted floor might be wrong if too many APs are included in the scan through floors and ceilings.

Finally, the accuracy of the results depends on the specifics of the adapter used, whether 2.4 GHz and 5 GHz networks are both received, and how well the built-in tools function. Many older laptops are more likely to only support 2.4 GHz networks, so their precision is reduced for not being able to see the 5 GHz networks in the building.

3.3 Application

The application works on all planned platforms. However, some bugs appear due to the used libraries and different specific Linux versions. The best performance was noticed on macOS machines.

However, the map is rendered correctly on all platforms, there are no problems with reading the database and moving between all states of the application.

On Windows machines, the built-in network scanner performs noticeably worse than on other platforms, which makes it very hard to use the application in a sensible manner. The scanned networks list is very inconsistent, often entirely empty despite dozens of networks being available. Unfortunately, no fix was found for this issue due to its inconsistency.

3.4 Discussion

Regarding the difference between 2.4 GHz and 5 GHz networks, we found that adapters which are unable to capture 5 GHz networks greatly underperformed and resulted in worse precision. Due to the scarce placement of routers in the Delta building, it is highly recommended to use all available networks in positioning. This is also why the configuration variables `RSSI_MIN` and `DIST_THRESHOLD` play an important role in the final result and must be configured for each system.

The terminal window can be read during runtime to see the list of networks filtered out after scanning. This can aid in deciding on what adjustments to make and what unknown networks to add to the routers database to increase precision.

Moreover, the weighted mean method turned out to be much more straightforward to implement and much more reliable, giving optimal results at all locations in the building.

Despite this, the method is not mentioned widely in literature.

After discovering the weighted mean method, most of the effort was put into improving it and making it more accurate. Thus, less time was dedicated to the trilateration method implementation. For this reason, only a simple mathematical implementation was included in the application, without any optimisations.

Besides this, we experimented with different adapters (both external and built-in) to find optimal constant variables, such as the minimum RSSI value and distance thresholds. These variables should be configured for each device separately. Given that the wireless adapter supports both 2.4 GHz and 5 GHz networks, reasonable default values were found for the variables, such as 300 for the `DIST_THRESHOLD`, 14 for `RAD_THRESHOLD`, 1.68 for `POWER`, 2.25 for `PATH_LOSS` and -77 for `RSSI_MIN`.

For example, `DIST_THRESHOLD` can be increased by 100 and `RSSI_MIN` decreased by 6 to include more routers in scans, but possibly reduce precision on average, as mentioned in Section 3.2.

Besides the configurable variables, other weight formulas should also be considered for the weighted mean method. Currently, all weights are assigned by the formula $1/RSSI$, as it performed the best after multiple experiments. However, more advanced formulas can be considered in the future.

Regarding the software aspect of this paper, some weak sides of the application can be seen on the Windows platform. Due to the nature of the `netsh` command-line tool, the performance of the scanning is quite weak in comparison to other platforms, especially if the wireless adapter does not support 5 GHz networks. For unresolved reasons, the `netsh` tool often does not return any nearby networks, rendering the tool unusable.

Additionally, the scaling of the GUI is altered on Linux and Windows platforms due to unresolved issues with the PySide library. This often results in font sizing and spacing being altered.

Further, multiple bugs exist in PySide that seem to be unresolved, such as the following issue with loading fonts correctly: `qt.qpa.fonts: Populating font family aliases took 263 ms. Replace uses of missing font family "A" with one that exists to avoid this cost.`, despite the font name being set to "Arial". This bug appears in the console after starting the application and does not seem to affect the performance in any way.

Finally, one more weakness in the application is in the lack of easy way to edit routers in the database and configuration variables, especially during runtime. While adding routers is possible with an interactive tool, editing or removing them requires manual

work with the provided database file. This can be solved by using a different database solution (such as SQL), or adding functionality to the application similar to the "Add New Router" mode.

In addition to the existing features, many new features can be added to the application, such as the ability to preview other floors of the building more easily, the option to upload custom maps interactively or edit the maps from the application, and a convenient way for the user to pin favourite or important locations.

New GUI components can be easily added to the `ui/components.py` file, and new files can be created in the `ui` folder and included in the main `app.py` script file.

4 Conclusion

For this thesis, the most commonly used methods for indoor positioning were analysed and described. From them, methods best fit to implement were selected and described.

These methods were implemented in an application with a rich interactive GUI and an easy visualisation on a map of the Delta building. For this, a list of routers and detailed map images were gathered. Exact coordinates of each router were calculated and saved into a database file. Using network scanning command-line tools, a list of nearby wireless networks is acquired to be used in the positioning methods and estimate the user's position, which is then displayed on the building floor map.

As a result, the developed application performs well on two of the three planned platforms. Installing the application requires only the Python 3 runtime to be installed on the target computer, as all remaining libraries can be installed by following the simple provided guide or install script. Using the application, it is possible to navigate the Delta building and get location predictions accurate up to 3-6 meters in some cases.

However, some issues exist with network scanning on Windows, with different bugs in the libraries used, and most importantly, with the implementation of the trilateration algorithm. Much more improvements can be made to the implementation, such as choosing an optimisation approach instead of strictly following the formula. Moreover, the accuracy can be improved further by introducing better methods of converting the RSSI value to distance, as the current formula introduces a lot of error.

Additionally, more advanced features can be added, such as better database management, the option to edit maps from the application, and improved Windows support.

In the future, more research could be done on the weighted mean positioning method to improve its accuracy further, as it is very easy to implement and could be used in many situations instead of trilateration, or as a fallback method.

References

[1] accelerometer | instrument | britannica. https://www.britannica.com/technology/accelerometer. (2022-05-03).

[2] gyroscope | definition, physics, & uses | britannica. https://www.britannica.com/technology/gyroscope. (2022-05-03).

[3] Installing packages — python packaging user guide. https://packaging.python.org/en/latest/tutorials/installing-packages/#requirements-files. (2021-12-08).

[4] Microsoft docs. https://docs.microsoft.com/en-us/windows/win32/api/wlanapi/ns-wlanapi-wlan_association_attributes. (2022-04-14).

[5] Pyside faq - qt wiki. https://wiki.qt.io/PySide_FAQ. (2021-12-08).

[6] Qt designer manual. https://doc.qt.io/qt-5/qtdesigner-manual.html. (2021-12-08).

[7] Radio frequency identification (rfid). https://www.fda.gov/radiation-emitting-products/electromagnetic-compatibility-emc/radio-frequency-identification-rfid. (2022-04-22).

[8] Satellite navigation - global positioning system (gps). https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps. (2022-04-22).

[9] Understanding the network terms ssid, bssid, and essid - techlibrary - juniper networks. https://www.juniper.net/documentation/en_US/junos-space-apps/network-director4.0/topics/concept/wireless-ssid-bssid-essid.html#jd0e46. (2022-05-04).

[10] What is gnss? https://www.euspa.europa.eu/european-space/eu-space-programme/what-gnss. (2022-04-22).

[11] What is python? https://www.python.org/doc/essays/blurb/. (2021-12-08).

[12] Markus Bullmann, Toni Fetzer, Frank Ebner, Markus Ebner, Frank Deinzer, and Marcin Grzegorzek. Comparison of 2.4 ghz wifi ftm- and rssi-based indoor positioning methods in realistic scenarios. *Sensors*, 20(16), Aug 2020. pp 2–3.

[13] Andrew Butterfield and Ngondi Gerard Ekembe. *A Dictionary of Computer Science*. Oxford University Press, 2016.

[14] Wikimedia Commons. File:trilateration-with-3-stations.svg — wikimedia commons, the free media repository, 2020. [Online; accessed 7-December-2021].

[15] Xuerong Cui, Mengyan Wang, Juan Li, Meiqi Ji, Jin Yang, Jianhang Liu, Tingpei Huang, and Haihua Chen. Indoor wi-fi positioning algorithm based on location fingerprint. *Mobile Netw Appl*, 26:146–155, 11 2020.

[16] Tamer Dag and Taner Arsan. Received signal strength based least squares lateration algorithm for indoor localization. *Computers & Electrical Engineering*, 66:114–126, Feb 2018. pp 116–117.

[17] Abd Elgwad M. El Ashry and Bassem I. Sheta. Wi-fi based indoor localization using trilateration and fingerprinting methods. *IOP Conference Series: Materials Science and Engineering*, 610(1):012072, Sep 2019. pp 2.

[18] Veli Ilci, R Metin Alkan, V Engin Gülal, and Huseyin Cizmeci. Trilateration technique for wifi-based indoor localization. pages 25 – 28, 2015. pp 26.

[19] Geio Illus. Wi-fi positioning system. 2021.

[20] Daiki Kudou, Mitsuyoshi Horikawa, Tatsuya Furudate, and Azuma Okamoto. Indoor positioning method using proximity bluetooth low-energy beacon. 2016.

[21] Safar A. Maghdid, Halgurd S. Maghdid, Sherko R. HmaSalah, Kayhan Z. Ghafoor, Ali Safaa Sadiq, and Suleman Khan. Indoor human tracking mechanism using integrated onboard smartphones wi-fi device and inertial sensors. *Telecommunication Systems*, 71(3):447–458, Jul 2019. pp 4.

[22] Germán Martín Mendoza-Silva, Joaquín Torres-Sospedra, and Joaquín Huerta. A meta-review of indoor positioning systems. *Sensors (Basel, Switzerland)*, 19(20):4507, Oct 2019. pp 5–15.

[23] Martin Sauter. *From GSM to LTE: An Introduction to Mobile Networks and Mobile Broadband*. John Wiley & Sons, 2010.

[24] Dipl-Ing (FH) Christian Wolff. Multilateration - radartutorial. `https://www.radartutorial.eu/02.basics/rp52.en.html`. (2021-12-08).

Appendix

I. Sample Routers Database File

```
x,y,mac,ssid,floor,frequency,name

730,440,7c:21:0d:2e:d4:40,eduroam,0,2,Corridor Near 1037
730,440,7c:21:0d:2e:d4:4f,eduroam,0,5,Corridor Near 1037
730,440,7c:21:0d:2e:d4:41,ut-public,0,2,Corridor Near 1037
730,440,7c:21:0d:2e:d4:4e,ut-public,0,5,Corridor Near 1037

2020,160,7c:21:0d:2f:56:80,eduroam,1,2,1004
2020,160,7c:21:0d:2f:56:8f,eduroam,1,5,1004
2020,160,7c:21:0d:2f:56:81,ut-public,1,2,1004
2020,160,7c:21:0d:2f:56:8e,ut-public,1,5,1004

2450,160,7c:21:0d:2f:1a:e0,eduroam,1,2,1005
2450,160,7c:21:0d:2f:1a:ef,eduroam,1,5,1005
2450,160,7c:21:0d:2f:1a:e1,ut-public,1,2,1005
2450,160,7c:21:0d:2f:1a:ee,ut-public,1,5,1005

2860,160,7c:21:0d:2f:11:e0,eduroam,1,2,1006
2860,160,7c:21:0d:2f:11:ef,eduroam,1,5,1006
2860,160,7c:21:0d:2f:11:e1,ut-public,1,2,1006
2860,160,7c:21:0d:2f:11:ee,ut-public,1,5,1006

3920,160,7c:21:0d:2f:39:00,eduroam,1,2,1007
3920,160,7c:21:0d:2f:39:0f,eduroam,1,5,1007
3920,160,7c:21:0d:2f:39:01,ut-public,1,2,1007
3920,160,7c:21:0d:2f:39:0e,ut-public,1,5,1007
```

II. MacOS Scan Function

```
def scan_macos():
    # Run the airport utility as a subprocess
    # NB! The utility must be enabled/linked beforehand
    res = sp.run(['airport', '-s'], capture_output=True)

    # Parse output from the airport scan
    networks = []
    result = res.stdout.decode().split('\n')
    for idx,row in enumerate(result):
        row = row.strip().split(' ')

        # Filter out first and invalid rows
        if idx == 0 or row[0] == '':
            continue

        if not row[2].startswith('-'):
            continue

        # Check if any of the desired values are empty
        for i in range(0,3):
            if row[i] == '':
                print('[!] Incomplete data during nearby network
                    parsing')
                print('Did you run the app as sudo?')
                quit(1)

        # First columns contain the required values
        # Based on the output of airport -s command
        network = {
            'SSID': row[0],
            'MAC': row[1],
            'RSSI': int(row[2])
        }
        networks.append(network)

    return networks
```

III. Render Function Snippets

```
    def render(self):
        # Render the map
        print('Rendering...')

        # Use a simple/clean or full map
        map_mode = '-c' if self.window.simpleMapView.isChecked() else
            ''
        # Load map for the current floor
        path = f'map/korrus-{self.user["floor"]}{map_mode}.png'
        # Init a pixmap for the map
        pix = QPixmap(path)
        painter = QPainter(pix)
        painter.setFont(self.font)

        # Highlight all detected routers
        for router in self.nearby_routers:
            self.highlight_router(painter, self.routers[router['MAC'
                ]])

        # Draw all routers on this floor
        for mac,router in self.routers.items():
            if router['floor'] == self.user['floor']:
                self.draw_router(painter, router)

                # Draw router location name on map
                painter.drawText(router['x'] - 40, router['y'] - 28,
                    router['name'])

        self.draw_user(painter)

        # Init custom graphics scene
        scene = uic.CGraphicsScene()
        # Forward mouse click signals to update new router location
            info
        scene.signalMousePos.connect(lambda pos: self.
            add_router_on_click(pos))
        # If new router add mode is engaged, draw new router location
        # [...]

        painter.end()
```

```
36
37          # Scale map based on current zoom
38          pix = pix.scaled(self.img_w / self.map_scale,
39                           self.img_h / self.map_scale,
40                           Qt.AspectRatioMode.KeepAspectRatio,
41                           Qt.TransformationMode.SmoothTransformation)
42
43          # Add the pixmap to a scene in the QGraphicsView
44          scene.addPixmap(pix)
45
46          # Load the additional info overlay image if selected
47          # [...]
48          scene.addPixmap(pix_overlay)
49
50          self.window.mapView.setScene(scene)
51
52          # Move the map to give padding around all sides
53          self.window.mapView.setSceneRect(-100, -100, scene.width() +
                200, scene.height() + 200)
54
55          # Remap center coordinates based on the current map scale
56          rc_x,rc_y = self.remap_coords()
57
58          # Center map view on the user's location
59          if not self.add_new_router_mode:
60              center_x = rc_x(self.user['x'])
61              center_y = rc_y(self.user['y'])
62              self.window.mapView.centerOn(center_x, center_y)
63
64          self.window.mapView.show()
65
66          self.update_labels()
67          # List all routers and their distances
68          self.list_routers()
```

IV. Network Scanning Output

Output from `airport` on MacOS:

```
     SSID BSSID             RSSI CHANNEL HT CC SECURITY (auth/unicast/group)
ut-public 1c:d1:e0:3d:51:ae -92  44,+1   Y  EE NONE
eduroam   1c:d1:e0:3d:51:af -92  44,+1   Y  EE WPA2(802.1x/AES/AES)
ut-public 1c:d1:e0:42:a8:6e -91  136,-1  Y  EE NONE
eduroam   68:9e:0b:5c:e8:2f -91  100,+1  Y  EE WPA2(802.1x/AES/AES)
eduroam   7c:21:0d:2f:10:4f -90  64,-1   Y  EE WPA2(802.1x/AES/AES)
ut-public 7c:21:0d:2f:10:4e -90  64,-1   Y  EE NONE
eduroam   1c:d1:e0:3d:7b:af -89  52,+1   Y  EE WPA2(802.1x/AES/AES)
ut-public 1c:d1:e0:3d:7b:ae -87  52,+1   Y  EE NONE
ut-public 1c:d1:e0:3d:51:a1 -85  11      Y  EE NONE
ut-public 1c:d1:e0:3d:7b:a1 -82  1       Y  EE NONE
eduroam   1c:d1:e0:3c:16:4f -82  64,-1   Y  EE WPA2(802.1x/AES/AES)
ut-public 1c:d1:e0:3c:16:4e -82  64,-1   Y  EE NONE
eduroam   1c:d1:e0:3d:7b:a0 -82  1       Y  EE WPA2(802.1x/AES/AES)
ut-public 1c:d1:e0:45:1b:41 -53  6       Y  EE NONE
eduroam   1c:d1:e0:45:1b:40 -53  6       Y  EE WPA2(802.1x/AES/AES)
ut-public 1c:d1:e0:3c:99:8e -68  36,+1   Y  EE NONE
eduroam   1c:d1:e0:3c:99:8f -68  36,+1   Y  EE WPA2(802.1x/AES/AES)
ut-public 1c:d1:e0:45:1b:4e -53  100,+1  Y  EE NONE
eduroam   1c:d1:e0:45:1b:4f -53  100,+1  Y  EE WPA2(802.1x/AES/AES)
```

Output from `iw` on Linux:

```
BSS 1c:d1:e0:45:1b:4f(on wlp0s20f3) -- associated
  last seen: 175846.071s [boottime]
  TSF: 13687244872676 usec (158d, 10:00:44)
  freq: 5500
  beacon interval: 100 TUs
  capability: ESS Privacy SpectrumMgmt RadioMeasure (0x1111)
  signal: -57.00 dBm
  last seen: 1628 ms ago
  Information elements from Probe Response frame:
  SSID: eduroam
  Supported rates: 6.0* 9.0 12.0* 18.0 24.0* 36.0 48.0 54.0
  Country: EE Environment: bogus
    [...]
  Power constraint: 0 dB
  RSN:   * Version: 1
```

```
     * Group cipher: CCMP
     * Pairwise ciphers: CCMP
     * Authentication suites: IEEE 802.1X
     * Capabilities: 4-PTKSA-RC 4-GTKSA-RC (0x0028)
  BSS Load:
     * station count: 5
     * channel utilisation: 19/255
     * available admission capacity: 23437 [*32us]
  HT capabilities:
    Capabilities: 0x9af
      RX LDPC
      HT20/HT40
      SM Power Save disabled
      RX HT20 SGI
      TX STBC
      RX STBC 1-stream
      Max AMSDU length: 7935 bytes
      No DSSS/CCK HT40
    Maximum RX AMPDU length 65535 bytes (exponent: 0x003)
    Minimum RX AMPDU time spacing: 4 usec (0x05)
    HT RX MCS rate indexes supported: 0-31
    HT TX MCS rate indexes are undefined
  HT operation:
     [...]
  Extended capabilities:
     [...]
  VHT capabilities:
    [...]
  VHT operation:
     [...]
  WMM:   * Parameter version 1
     [...]
```

Output from `netsh` on Windows:

```
Wireless System Information Summary

[...]

============================================================
==================== SHOW NETWORKS MODE=BSSID ==============
============================================================
```

```
8
9  Interface name : Wi-Fi
10 There are 1 networks currently visible.
11
12 SSID 1 : Hidden-SSID
13     Network type            : Infrastructure
14     Authentication          : WPA2-Personal
15     Encryption              : CCMP
16     BSSID 1                 : d8:33:ef:xx:xx:xx
17          Signal             : 94%
18          Radio type         : 802.11ac
19          Channel            : 56
20          Basic rates (Mbps) : 6 12 24
21          Other rates (Mbps) : 9 18 36 48 54
22
23 [...]
```

V. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Anton Slavin**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Delta Wi-Fi Positioning System,

   supervised by Danielle Melissa Morgan.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Anton Slavin
10/05/2022