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

Nikita Bahhir

Positioning using WiFi

Bachelor's Thesis (9 ECTS)

Supervisor(s): Danielle Melissa Morgan

Tartu 2024

Positioning using WiFi

Abstract:
This thesis looks into the problem of outdoor positioning and whether WiFi can be used to estimate the position. The most popular method of estimating position using GPS can be unreliable in certain situations, such as cloudy weather or a high density of buildings. We tried using the abundance of WiFi networks to offer an alternative way of estimating positioning. Here, we show the app, which acts as a tool for positioning and data analysis, and the results of using the app in cities, along with its accuracy. The results revealed that GPS accuracy is still better overall, but the app could still be used in situations where GPS is unavailable. On several occasions, the WiFi positioning produced better results than GPS. In the end, there is still a lot of improvement space, but the app could already be used in the cities to get a fairly accurate position and gather useful data.

Keywords:
BSSID, SSID, Beacon Frame, Trilateration, Triangulation, WiFi, positioning, WiGLE, GPS

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

Positsioneerimine WiFi abil

Lühikokkuvõte:

See töö uurib välistingimustes positsioneerimise probleemi ja seda, kas WiFi-võrke saab kasutada asukoha määramiseks. Kõige populaarsem meetod asukoha määramiseks GPSi abil võib teatud olukordades, nagu pilvine ilm või suur hoonete tihedus, olla ebakindel. Proovisime kasutada WiFi-võrkude rohkust alternatiivse positsioneerimise meetodina.Siin näitame rakendust, mis toimib positsioneerimis- ja andmeanalüüsi tööriistana, ning rakenduse kasutamise tulemusi linnades koos selle täpsusega. Tulemused näitasid, et GPSi täpsus on siiski üldiselt parem, kuid rakendust saab kasutada olukordades, kus GPS pole saadaval. Mitmel korral andis WiFi positsioneerimine paremaid tulemusi kui GPS. Lõpuks on veel palju arenguruumi, kuid rakendust saab juba kasutada linnades üsna täpse asukoha saamiseks ja kasulike andmete kogumiseks.

Võtmesõnad:
BSSID, SSID, Beacon Frame, Trilateratsioon, Triangulatsioon, WiFi, positsioneerimine, WiGLE, GPS

CERCS:
P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)
P175 Informaatika, süsteemiteooria

Contents

1 Introduction

Navigation has significantly evolved over recent decades, transitioning from traditional methods to technologically advanced systems that seamlessly integrate with our daily lives. Today, navigating our world is more accessible than ever. As a result, individuals routinely rely on navigation services for various activities: from planning trips and sightseeing to finding points of interest and checking bus schedules.

A quintessential example of a navigation tool that has become integral to our routines is Google Maps. According to Gitnux—a reputable and frequently cited source—Google Maps ranked as the second most downloaded Google app, boasting over 13 million downloads in July 2022 [1]. Beyond its standalone application, it powers many other services, providing navigation-related functionality. Approximately 5 million websites utilize the Google Maps service daily to enhance their user experience [2]. These figures underscore the ubiquity and utility of navigation software in the present era.

The backbone of most civilian navigation systems is the GPS or Global Positioning System. Over 30 satellites orbit Earth, designed in such a way that a minimum of four satellites are always available from any point on the globe to triangulate the location of a receiver device [3]. However, satellite signals are not infallible. As they traverse from orbit to the Earth's surface, various elements—ranging from clouds and buildings to competing signals—can hinder their clarity. This signal attenuation is especially pronounced in densely urbanized environments, a phenomenon extensively studied and addressed in numerous scholarly works. [4]

This paper pivots its focus toward an alternative localization method: WiFi. In urban areas with vast amounts of active WiFi networks, interference between GPS and WiFi signals can compromise the efficiency of the former. Harnessing the dense WiFi networks might promise a more precise and streamlined localization process in such settings. The forthcoming sections provide a comprehensive overview of:

- Various localization techniques, elaborating on their advantages and limitations.
- Information about WiFi, emphasizing unique network identifiers such as Beacon frames, SSID, and BSSID.
- The WiGLE database, a pivotal resource for correlating WiFi networks with specific locations.

Section 2 looks at some popularly studied and applied positioning methods. It also provides an overview of WiFi transmission details, explaining terms related to this research. The section ends with an explanation of the WiGLE database and app and how and why they were used in this paper.

Subsequently, Section 3 will cover the methodology of creating an application that uses WiFi networks to estimate user device locations. Each tab in the application is explained in the subsections.

Afterwards, Section 4 will explore the results gathered from the testing and will provide an overview of the findings from them.

Chapter 5 provides various discussion points and potential future improvements and questions.

Chapter 6 concludes this thesis. Finally, the Reference and Appendix chapters are provided after Chapter 6.

Grammarly was used to make the text more readable and grammatical. This application is based on AI; however, it does not create new content but processes and analyzes existing texts, helping to improve the writing and fix various text-related mistakes. It can pick more accurate and suitable words, change the order of words, and provide general writing support. It was solely used as a supporting tool and was not used to generate any sort of new content.

2 Background

This paragraph intends to give an overview of a more theoretical aspect of this paper.

Firstly, different methods for localization in indoor and outdoor areas will be discussed, such as fingerprinting, triangulation, and trilateration. Although not directly used in this work, their general concepts have greatly influenced it. Knowing or being acquainted with said concepts will help to understand this paper's topic and proposed solutions.

The succeeding section explains how phones receive and recognize the WiFi network around them. More specifically, it will cover how helpful information can be extracted and used in identification and later localization.

Lastly, it is important to explain precisely how to map uniquely identified WiFi networks and real-world locations to the reader. This will be the purpose of the section about the WiGLE database, which can be described as the core component of this paper's solution.

2.1 Positioning Methods

The effectiveness and accuracy of WiFi positioning heavily rely on methods that help identify the exact location of objects and individuals in a given environment. Some approaches have emerged regarding WiFi positioning, each offering a distinct viewpoint and method for precisely localizing objects. Some are better suited to work in an enclosed environment, while others are more efficient in the outside areas. This section will delve into the details and complexities of three WiFi positioning methods: trilateration, triangulation, and fingerprinting. The principles of how these methods work in practice will be explored and evaluated based on previously made research.

2.1.1 Fingerprinting

Fingerprinting consists of making multiple measurements using any sort of device able to receive a WiFi signal. Access points (AP) of WiFi networks are placed around the navigable area first. Their placement can range from random to pre-planned and calculated, which can severely affect location estimation results, for example, wall obstruction and signal interference. [5] A device capable of receiving the signal is then used throughout the affected area to measure the signal strength to every accessible AP around it. RSSI, the measurement of WiFi signal power, is most widely used as a signal strength unit. All acquired information, which is the RSSI measurements/values/signal strength of APs at different locations, is stored in the database. [6]

If the device wants to estimate its position in such a place, it uses the database and its measurements of nearby WiFi AP points' signal strength to find the closest match in the database. The matching positional data is used to estimate the device's location. [6]

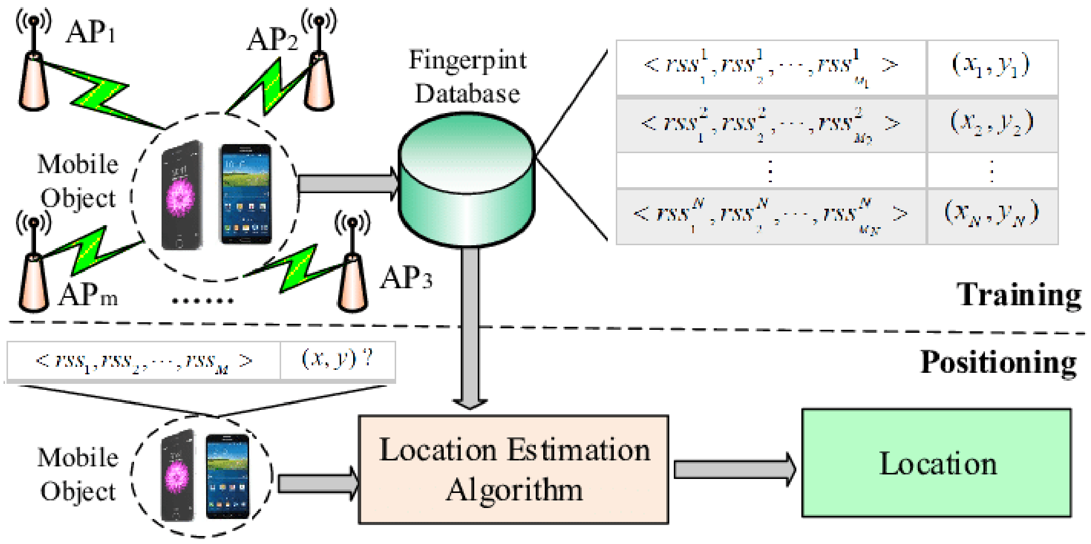

AP1
AP2
Fingerpint Database
Mobile Object
APm
AP3
......
$< rss_1^1, rss_2^1, \cdots, rss_{M_1}^1 >$
(x_1, y_1)
$< rss_1^2, rss_2^2, \cdots, rss_{M_2}^2 >$
(x_2, y_2)
$< rss_1^N, rss_2^N, \cdots, rss_{M_N}^N >$
(x_N, y_N)
Training
Positioning
$< rss_1, rss_2, \cdots, rss_M >$
(x, y) ?
Mobile Object
Location Estimation Algorithm
Location

Figure 1. The process of WiFi Indoor Fingerprinting Method. [7]

Figure 1 clearly distinguishes two main stages of the fingerprinting process – training and positioning. Training is when the required data is captured and stored in the database. In contrast, the positioning stage depicts looking for the closest match in the database and applying algorithms to transform the data in some given way, finally getting the estimated location.

Countless algorithms aim to improve the accuracy of estimations. For example, a recently published article by Jin Zheng, Kailong Li, and Xing Zhang discusses using a robust localization model and a standard particle swarm optimization (SPSO) algorithm to boost estimation accuracy considerably [8]. The increase in accuracy, ranging from 15.32 to 36.64 per cent, was reported by the research group compared to commonly used estimation algorithms. Those include K-Nearest Neighbors, Support Vector Machine, Logistic Regression, and Random Forest algorithms. [8]

K-Nearest Neighbors (KNN) is an algorithm that predicts or classifies data by looking at its nearest neighbours. This algorithm is often used in different fields, such as machine learning or remote sensing. [9]

Support Vector Machine (SVM) algorithm is a learning algorithm, which tries to find a two-dimensional plane in a three-dimensional space that separates different classes in the feature space in the best way. Feature space can be described as the set of all possible values for a chosen set of features from that data. It is supervised, meaning it is used to make predictions or classify the data. [10]

Logistic Regression (LR) is a model, which is commonly used to understand the relationship between one binary variable and other variables. All variables are independent. Logistic Regression calculates a binary outcome probability by using a logistic function.

Often used in psychology to evaluate multivariate screening accuracy. [11]

Finally, the Random Forest (RF) algorithm constructs a range of decision trees during its training. Each tree is independent of others and makes its own prediction. The final prediction is an aggregation of decisions of individual trees. Sometimes, the average of all decisions is taken as a final result; other times, the most popular decision is taken as the final one. [12]

Jin Zheng, Kailong Li, and Xing Zhang explain the SPSO algorithm as a probabilistic evolutionary algorithm, where the potential solution can be represented as a position of elements in the swarm [13]. The specifics of said algorithm are pretty complex and out of scope for this paper.

Fingerprinting offers a good opportunity for outdoor positioning because many accuracy improvement methods have been researched. If the positioning data is provided by the database, then the AP's position does not need to be calculated.

2.1.2 Trilateration

This method relies mainly on distance measurement between APs or emitters and receivers. In contrast to triangulation, where angles are used, the angles play no role here. There are numerous articles on trilateration. However, the article from Oguejiofor O. S, Aniedu A. N, Ejiofor H. C, and Okolibe A. U, posted in 2013, seems to be the clearest one explaining the topic as such they will be referred to while discussing the trilateration method of positioning [14].

If the position of some receiver needs to be estimated in relation to one AP, then it is possible to measure the distance d_1 from the receiver to the AP with coordinates (x_1, y_1). This scenario is depicted in Figure 2. The method of acquiring and calculating the distance will be touched upon later. For now, it can be assumed that the distance d_1 has already been accurately measured. The distance from this AP can be represented as a circumference around it, with the receiver's location possibly being at any given point in this circumference. Currently, estimating the receiver's coordinates (x, y) is impossible, as the number of possible points is infinite. Now let's consider the scenario with 2 access points as seen in Figure 3.

The second access point with coordinates (x_2, y_2) has been added, with its distance to the receiver d_2 represented as a circumference. Currently, only two possible points match distances d_1 and d_2 to the emitters, namely point 1 with coordinates (x_p1, y_p1) and our receiver with coordinates (x, y). The number of possible positions was reduced from infinite to just two, but the estimation can still result in serious error. To mitigate this, a third point can be added as seen in Figure 4.

The third AP has been added to Figure 4 with its distance d_3, coordinates (x_3, y_3), and the circumference. Now, only one particular point matches all three given distances d_1, d_2 and d_3 – our receiver at (x, y).

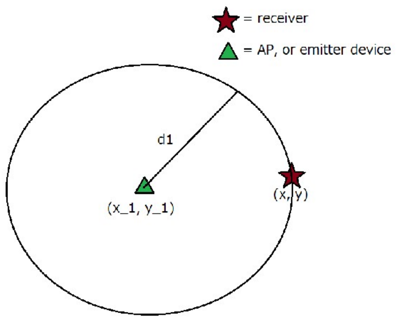


Figure 2. Trilateration with 1 AP

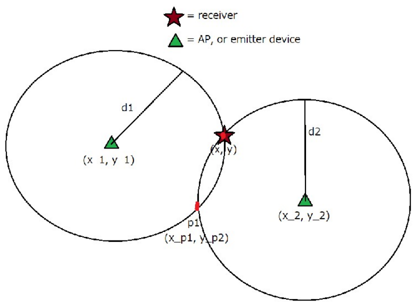


Figure 3. Trilateration with 2 APs

The basic equation for the 2D circle on one plane is:

$$d^2 = x^2 + y^2$$

For a circle, centered at the point (x_0, y_0) the formula is expressed as:

$$d^2 = (x - x_0)^2 + (y - y_0)^2$$

For our three circles from the diagram, the equations become:

$$d_1^2 = (x - x_1)^2 + (y - y_1)^2$$

$$d_2^2 = (x - x_2)^2 + (y - y_2)^2$$

$$d_3^2 = (x - x_3)^2 + (y - y_3)^2$$

After this system of equations is solved for x and y, the receiver's location can be calculated. However, as stated in the article, these are independent non-linear simultaneous equations which cannot be solved mathematically. The solution involves different transformations applied to the formulas. [14]

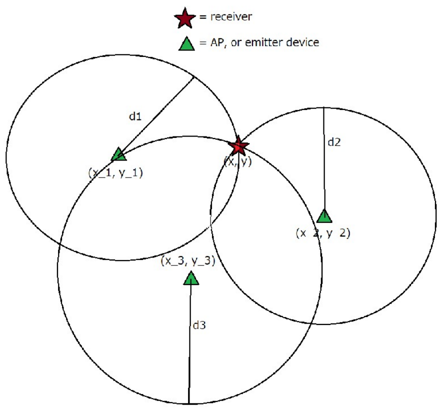


Figure 4. Trilateration with 3 APs

This method, however, can be pretty inaccurate, primarily due to difficulties in converting RSSI (signal strength) into distance. RSSI differs across various emitter types and models; obstructions can interfere with signal propagation, and the receiving device influences the resulting value. Multiple proposed algorithms exist to optimize and estimate the distance based on RSSI value [15]. Still, they often include complex mathematics, others are inaccurate, and some require additional information, which is frequently unavailable. As such, trilateration does not seem to be a suitable method for outside WiFi positioning because the information about WiFi networks is minimal, and multiple obstructions will make it extremely hard to use trilateration in highly populated areas with chaotic spread of wireless networks and building interferences. The information is minimal because the only useful information that is possible to gather is identifiers for the network and RSSI or signal strength. The positions of these APs, which are essential to using this method, are still unknown.

2.1.3 Triangulation

Triangulation is often confused with trilateration, possibly because of similar spelling and overall foundation. This method, as well as trilateration, measures the distance between one or more APs and a receiver, but more importantly, it also uses angles [16].

Figure 5 shows us the fundamental inner workings of an algorithm for using angles. In this case, Point 3 is the receiver, while Points 1 and 2 are some emitters. We have a Baseline length, a known distance between two emitters. Angle-of-Arrival (AoA) is used to calculate the angle in this triangle. These angles can then be used to calculate the location of Point 3. [18]

This method is also unsuitable for outdoor WiFi positioning, as it is not possible to gather information about the positions of the emitters and the angles between the points.

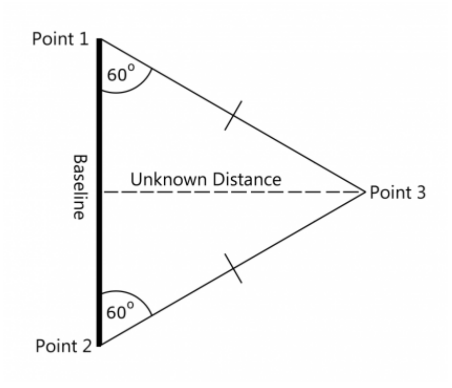


Figure 5. Basic triangulation method. [17]

2.2 WiFi Beacons

Beacons are important in managing communications between devices using the **IEEE 802.11** standard, known as WiFi. They aim to periodically propagate short-length transmissions to tell possible candidate receivers about nearby access points and available WiFi services. This way, receivers can process this information to make decisions regarding different WiFi networks [19]. The transmission period is also easily configurable, which provides control over how much it pollutes the air with this transmission [20].

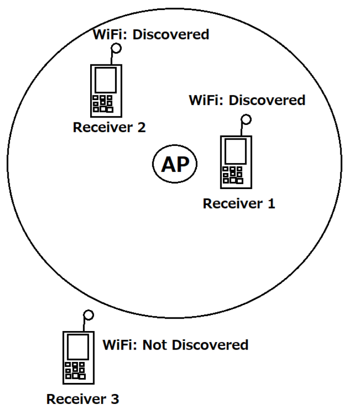


Figure 6. AP advertising itself to receivers

As shown in Figure 6, two receivers (receiver 1 and receiver 2) were in the range of the Beacon broadcast, which enabled them to identify and discover the network, while the third receiver was out of range and was not notified about the advertised access point. The range of the transmission depends on many factors, including the operating frequency, the power of the antenna or the presence of any physical obstructions. [21]

The most prominent example of this process is something that can be unavoidably

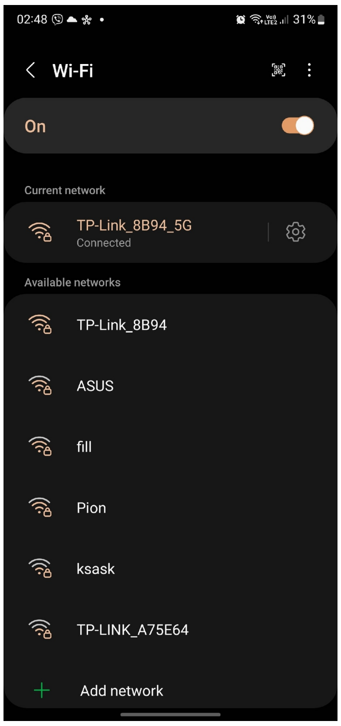


Figure 7. An example of a Beacon advertisement – WiFi network search

encountered every day – WiFi network search in smartphones. In Figure 7, a Samsung Galaxy S22 Ultra is used as a Beacon frame broadcast receiver to identify nearby WiFi networks and, if needed, to connect to one of them. The phone internally processes the received transmission, displaying the information conveniently and user-friendly.

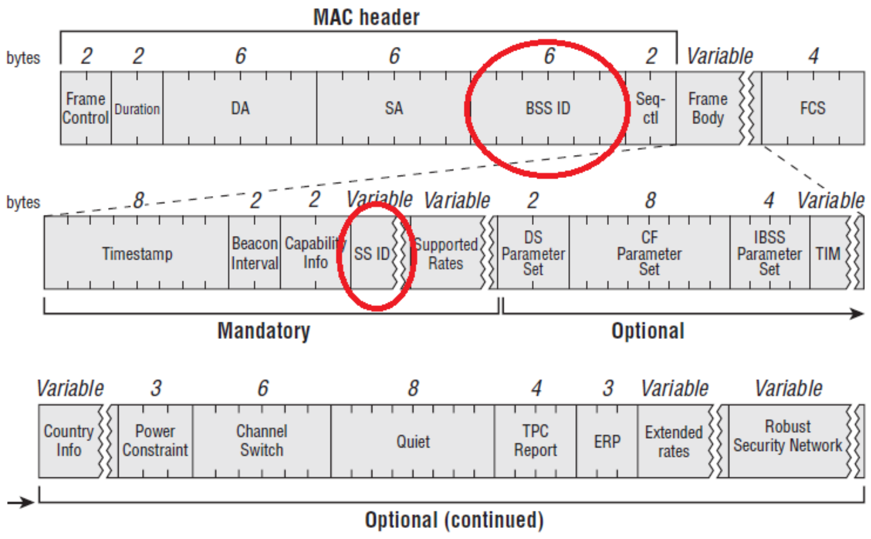


Figure 8. Beacon Frame structure. [22]

The Beacon Frame contains a lot of information regarding the network and the frame itself. It mainly consists of two parts - Medium Access Control (MAC) Header and message body. The MAC Header is a part of the Beacon Frame that contains information, which identifies the transmitter and tells the receiver if it can accept the message. [20] Figure 8 demonstrates the structure of a Beacon Frame. For the purposes of this paper, BSSID contained in the MAC Header and SSID contained in the message body are the most important ones, as the identification of a WiFi network is needed for the location estimation [20]. They have been marked in Figure 8.

2.2.1 BSSID

The BSSID, or Basic Service Set Identifier, as mentioned previously, is present in every Beacon transmission as a part of the MAC Header. The Header provides the MAC address of the AP transmitter, a unique address assigned to the device to allow communication on a network, meaning it can be used to identify the access point. [23]

2.2.2 SSID

The SSID, or Subscription Service Identifier, represents the identity of an Extended Service Set (ESS). In simpler terms, it acts as a standard network name for this access point and its network [20]. The SSID can be seen on a smartphone screen (Figure 7) during a WiFi network search and what is most familiar to people. However, it is essential to note that the SSID **does not** uniquely identify a WiFi network or an AP, meaning multiple APs worldwide can have identical SSID, which is not valid for the BSSID.

2.3 WiGLE

After reviewing several positioning methods, it was decided to focus on Fingerprinting. A database is required to store information about networks and their locations. With such a database, it would be possible to map networks to specific coordinates and estimate their locations. One of the largest ones is WiGLE.

WiGLE, which stands for Wireless Geographic Logging Engine, is a platform and database designed to collect and manage network information, specifically on WiFi and cellular networks. It relies on a community of users who actively contribute by collecting and sharing data about the networks they come across. This data includes details about the locations of WiFi access points, cellular towers, and their characteristics. The database is maintained through users' submissions through their mobile app or dedicated hardware devices that scan and record nearby network information passively. By following this community-driven approach, WiGLE ensures its database remains up-to-date and can provide network localization, research, and analysis resources. Only registered users can access the database, visualize network information through the WiGLE website or use their custom API. This makes it an essential tool for localization-based services. [24].

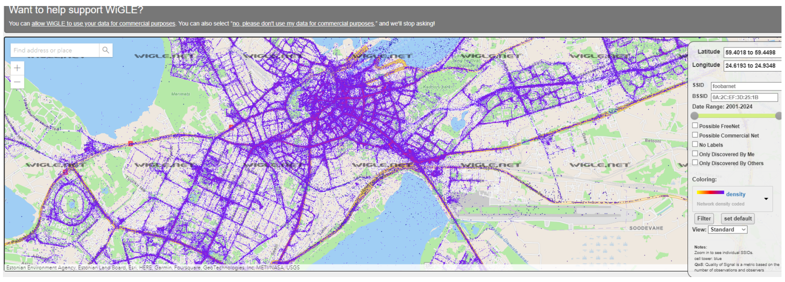
WiGLE database was chosen for mapping WiFi networks to coordinates because it is the most extensive public database available now. It fulfils the requirements for an extensive database, counting more than 1 billion networks present, and it counts as a training stage for fingerprinting [24]. It also provides and documents various ways to interact with its database and system, providing an overall good experience working with it. It is easy to establish a connection between their server and a software, which is crucial for the practical part of this thesis.

2.3.1 Database

The database is the main product provided by WiGLE. As of the time of writing this, the WiGLE database has over 1.1 billion WiFi networks and over 21 million cellular towers saved in it [24]. Each entry in the database contains a lot of information:

- names (SSIDs);
- MAC addresses (BSSIDs);
- signal strengths;
- times of capturing;
- address information if available, like city, region, country;
- estimated coordinates
- technical fields, such as frequency, encryption, etc.

The full JSON structure of the WiGLE database entity is available in Appendix I.WiGLE Database Entry Example.



Want to help support WiGLE?

You can allow WiGLE to use your data for commercial purposes. You can also select "no, please don't use my data for commercial purposes," and we'll stop asking!

Latitude 59.4018 to 59.4498
Longitude 24.6193 to 24.9348
SSID foobarnet
BSSID 0A:2C:EF:3D:25:1B
Date Range: 2001-2024
☐ Possible FreeNet
☐ Possible Commercial Net
☐ No Labels
☐ Only Discovered By Me
☐ Only Discovered By Others
Coloring: density
Network density coded
View: Standard

Notes:
Zoom in to see individual SSIDs.
cell tower: blue
QoS: Quality of Signal is a metric based on the number of observations and observers

Figure 9. WiGLE database, accessible from their website

Figure 9 shows what users typically see when entering the WiGLE official website. This is a graphical interface to query data from their database. It includes an interactive map with multiple purple dots – each representing a network placed in a location calculated by WiGLE. Each can be pressed to provide more detailed information regarding this specific network.

They also have an API, which allows users to access data in a more programmatic way by using different WiGLE API endpoints. Endpoints are the location, where API receives a request regarding specific functionality and they are typically provided in the form of URL [25]. These endpoints allow the user to communicate with the WiGLE server by sending requests to them. Users need to register and possess a non-expirable token, which can be generated on the same website for free and forever.

There are numerous ways to contribute to the database. Uploading a strictly formatted CSV file containing the network information is possible. DataHub defines CSV, Comma Separated Values, as a "very old, very simple and very common ‘standard‘ for (tabular) data". The format is supported by many tools ranging from sheets, like Google Sheets, to databases, to programming languages, like Python. [26]

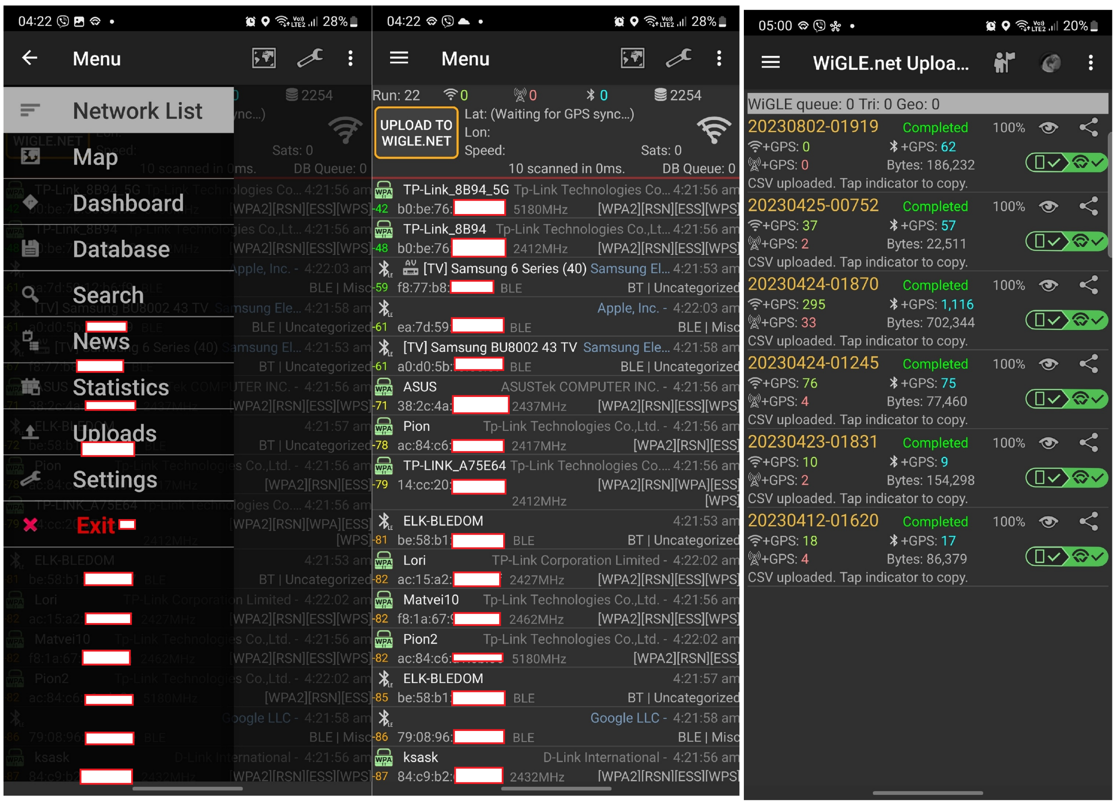
It is also possible to use their dedicated app to automate this process and eliminate the possibility of incorrectly formatted CSV files.

Even though WiGLE offers a convenient GUI on their website (see Figure 9), the API can be used to support better automation and communication with different software. In order to have access to it, the user needs to be registered on their website and be provided an API key. API key can be acquired either manually from the profile page on their website, or as an alternative there is a dedicated API endpoint, which provides the API key in exchange for the username and the password.

2.3.2 WiGLE Wardriving App

WiGLE has a dedicated app to make accessing the database, passively capturing nearby WiFi networks, and processing and uploading data to the WiGLE database as easy as possible. It is available via both the Play Store and Apple Store. It is open source, which means the code is open to the public.

It is possible to use the app in various ways, including accessing the database and the capability to upload CSV files to the server, but the most interesting feature is the ‘Wardriving‘. Wardriving is in the app name, and in the networking context, it means driving or moving around and capturing information about nearby WLANs (Wireless Local Area Networks). [24]



```
04:22                                   28%
Menu
Run: 22     0      0      0      2254
UPLOAD TO   Lat: (Waiting for GPS sync...)
WIGLE.NET   Lon:
            Speed:                      Sats: 0
                 10 scanned in 0ms.     DB Queue: 0
TP-Link_8B94_5G  Tp-Link Technologies Co... 4:21:56 am
-42 b0:be:76:[redacted]  5180MHz     [WPA2][RSN][ESS][WPS]
TP-Link_8B94   Tp-Link Technologies Co.,Lt... 4:21:56 am
-48 b0:be:76[redacted]  2412MHz      [WPA2][RSN][ESS][WPS]
[TV] Samsung 6 Series (40) Samsung El... 4:21:53 am
-59 f8:77:b8:[redacted]  BLE             BT | Uncategorized
                               Apple, Inc. - 4:22:03 am
-61 ea:7d:59[redacted]  BLE                      BLE | Misc
[TV] Samsung BU8002 43 TV  Samsung Ele... 4:21:58 am
-61 a0:d0:5b:[redacted]  BLE            BLE | Uncategorized
ASUS              ASUSTek COMPUTER INC. - 4:21:56 am
-71 38:2c:4a:[redacted]  2437MHz     [WPA2][RSN][ESS][WPS]
Pion           Tp-Link Technologies Co.,Ltd. - 4:21:56 am
-78 ac:84:c6:[redacted]  2417MHz          [WPA2][RSN][ESS]
TP-LINK_A75E64 Tp-Link Technologies Co.... 4:21:56 am
-79 14:cc:20:[redacted]              [WPA2][RSN][WPA][ESS]
                         2412MHz                     [WPS]
ELK-BLEDOM                                      4:21:53 am
-81 be:58:b1[redacted]  BLE              BT | Uncategorized
Lori          TP-Link Corporation Limited - 4:22:02 am
-82 ac:15:a2:[redacted]  2427MHz     [WPA2][RSN][ESS][WPS]
Matvei10       Tp-Link Technologies Co.,Ltd. - 4:21:56 am
-82 f8:1a:67:[redacted]  2462MHz     [WPA2][RSN][ESS][WPS]
Pion2          Tp-Link Technologies Co.,Ltd. - 4:22:02 am
-82 ac:84:c6:[redacted]  5180MHz          [WPA2][RSN][ESS]
ELK-BLEDOM                                      4:21:57 am
-85 be:58:b1[redacted]  BLE              BT | Uncategorized
                               Google LLC - 4:21:58 am
-86 79:08:96:[redacted]  BLE                     BLE | Misc
ksask              D-Link International - 4:21:56 am
-87 84:c9:b2:[redacted]  2432MHz     [WPA2][RSN][ESS][WPS]
```

```
05:00                                   20%
WiGLE.net Uploa...
WiGLE queue: 0 Tri: 0 Geo: 0
20230802-01919   Completed   100%
+GPS: 0            +GPS: 62
+GPS: 0            Bytes: 186,232
CSV uploaded. Tap indicator to copy.
20230425-00752   Completed   100%
+GPS: 37           +GPS: 57
+GPS: 2            Bytes: 22,511
CSV uploaded. Tap indicator to copy.
20230424-01870   Completed   100%
+GPS: 295          +GPS: 1,116
+GPS: 33           Bytes: 702,344
CSV uploaded. Tap indicator to copy.
20230424-01245   Completed   100%
+GPS: 76           +GPS: 75
+GPS: 4            Bytes: 77,460
CSV uploaded. Tap indicator to copy.
20230423-01831   Completed   100%
+GPS: 10           +GPS: 9
+GPS: 2            Bytes: 154,298
CSV uploaded. Tap indicator to copy.
20230412-01620   Completed   100%
+GPS: 18           +GPS: 17
+GPS: 4            Bytes: 86,379
CSV uploaded. Tap indicator to copy.
```

Figure 10. Wardriving App different tabs. A - tab list, B - Network List, C - Uploads

As seen in Figure 10A, multiple functionalities are present in the app:

- Network List
- Map
- Dashboard
- Database
- Search

- News
- Statistics
- Uploads
- Settings

For the purposes of this paper, the Network List and Uploads tabs were used. Network list, as seen in Figure 10B, lists all currently passively discovered networks in the area surrounding the receiver. It contains a large amount of information, including encryption types, location, signal strength, SSID, BSSID, type of emitter, general information about the search (how many have been found, what types, etc), and even the speed of the receiver as of the scanning time. It is intended to work even if the app runs in the background. There is also a button to upload the scanned networks to the WiGLE database. Before persisting the data in the database, the WiGLE estimates the location of the AP based on the location of the user. This process is described by the WiGLE team as "weighted-centroid trilateration", which is the average of the latitudes and longitudes gathered using the squared signal strength as a weight. [24] In even more basic explanations, the WiGLE app makes numerous measurements, applies the algorithms above, and finds the average of those calculations, then uploads the results to the database.

The status and history of uploads can be seen in the "Uploads" tab, depicted in Figure 10C. It also contains a lot of information specific to each upload attempt. The upload usually consists of 2 stages: processing and upload. Processing uses the defined algorithms to estimate the position of the discovered AP. Upload is responsible for uploading the information to the database. When both of these processes are completed, the data is available in the WiGLE database.

3 Methods

One of the main goals of this thesis is to develop an app, which could be used as a tool to get location estimation using WiGLE and analyze the data related to GPS and manually entered coordinates.

This section will delve deeper into the process and methods employed in developing this app with the objective of using the abundance of WiFi networks in urban areas to estimate the user's position with adequate accuracy and compare the results with GPS.

The methodology adopted in this research covers the entire development cycle from conceptualization to practice. It involves software development, data collection and processing, optimization, and algorithmic design, each playing a crucial role in achieving the result.

The tools section covers the software and hardware used in this paper and their configurations and specifications.

3.1 Tools

The following subsection will focus on the tools used to gather data, the coding choice, and the development environment. The Data Gathering section will explain what was used to capture the information needed for this thesis. Then, the focus will shift to the software side of the project - what coding languages were used, and why they were picked. Finally, the section will end with a subsection about the development environment.

3.1.1 Data Gathering

A Samsung Galaxy S22 Ultra smartphone was used to capture GPS and WiFi data using apps and in-built features. The device supports 2.4 GHz, 5 GHz, and 6 GHz signals, however, only 2.4 GHz and 5 GHz signals are covered.

As mentioned previously, fingerprinting was chosen as a method of estimating a user's position. A huge database would be needed to accomplish this, and it was decided to use the WiGLE Database, as it contains a huge amount of networks with estimated coordinates. However, there is no guarantee that the information about local networks is up-to-date or even if the networks are present there. To augment the database for Estonian cities and make sure that the networks there are updated, the Samsung Galaxy S22 Ultra, together with the WiGLE Wardriving App, was used extensively to populate the database. As the phone supports all currently used WiFi frequencies, there is no potential issue of losing some of the networks. Based on the information from the Wardriving app, more than 6000 networks were added to and updated in the WiGLE database using Samsung Galaxy S22 Ultra.

Google Maps service was used to gather manual coordinates, which can be used as a target to test the accuracy of GPS and WiGLE..

3.1.2 Coding

Two primary candidates were considered when choosing the working language of the future app. One is Python, and the second one is Kotlin. Kotlin is a high-level programming language built upon Java. It is statically typed, meaning every data typed needs to be explicitly stated at the definition stage. It puts certain constraints but makes the code overall safer and cleaner, as the program always knows what data it can and cannot use. It is also compiled, meaning it is entirely translated into the assembly code. This makes startups a bit slower because the program needs some time to be compiled into an assembly, but makes the actual performance a lot faster. [27]

Python, on the other hand, is not statically typed. Variables are declared without any types. It is also an interpreted language. In short, every line of code is being compiled and run step-by-step instead of compiling the whole code at the startup. This, in essence, makes the startup faster but slower overall performance. However, debugging is more straightforward, as the code can be effortlessly stopped and examined at any line. [28]

Both languages have a range of frameworks for Android Development. Python's most prominent framework is Kivy. It allows programmers to create GUI applications for both desktop and mobile operating systems, including the Android operating system. Kotlin has a lot more going on with Android, as it is considered one of the primary usages of the language.

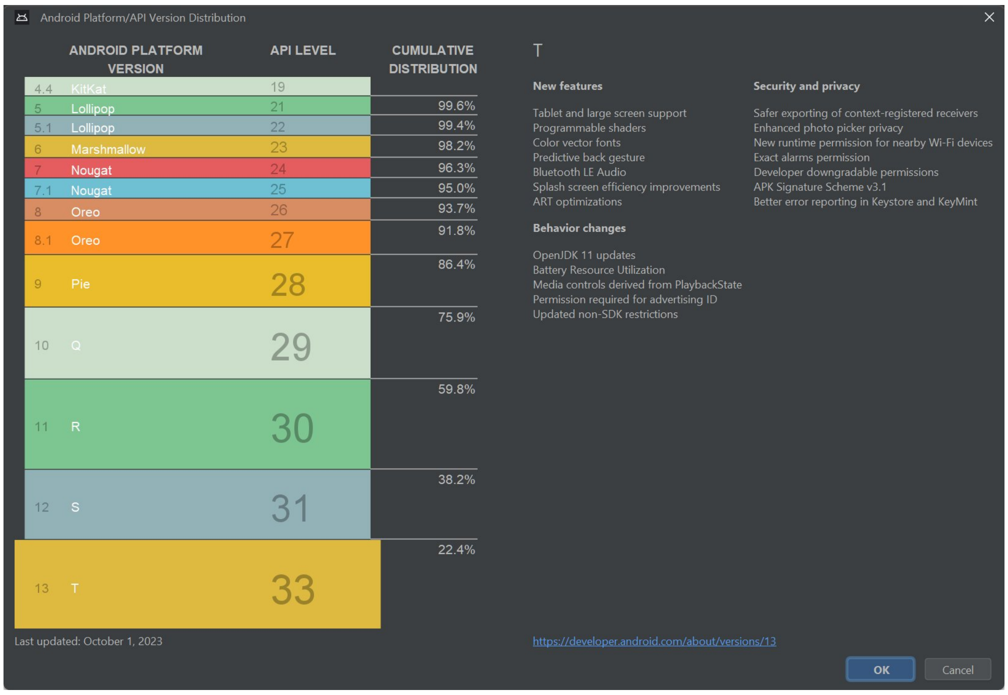
Kotlin was chosen as the language for developing this app. The familiar Java foundation and vast helpful libraries and documentation targeted at mobile development were key factors in choosing this language. Namely, Kotlin is the language Android Jetpack uses, a collection of Android libraries. [29]

3.1.3 Development Environment

Android Studio is the official Integrated Development Environment (IDE) for Android applications. It offers a comprehensive set of tools for Android developers. It supports the latest Android Software Development Kits (SDK) and provides a powerful tool for emulating different hardware devices, such as tablets and mobile. This makes the development and testing process very fast and convenient, as there is no need to build and transfer the app to the physical phone every time some change is made. Code can be compiled, run and tested with a single button press without using any other device. Emulated hardware makes testing the app against different hardware easy, which would have been difficult otherwise.

The choice of Android Studio for app development is motivated by its excellent compatibility with the Android ecosystem, a collection of useful Android libraries, and its hardware emulation capabilities. Android Studio was configured to suit the specific needs of the project. One of the more critical decisions was to determine the minSdkVersion and targetSdkVersion. In other words, a minimally supported Android

API version and the one for which the app is built. This was a difficult choice to make, as the current Android market is overflown with a wide range of devices with varying system versions.


Android Platform/API Version Distribution

ANDROID PLATFORM VERSION		API LEVEL	CUMULATIVE DISTRIBUTION
4.4	KitKat	19	
5	Lollipop	21	99.6%
5.1	Lollipop	22	99.4%
6	Marshmallow	23	98.2%
7	Nougat	24	96.3%
7.1	Nougat	25	95.0%
8	Oreo	26	93.7%
8.1	Oreo	27	91.8%
9	Pie	28	86.4%
10	Q	29	75.9%
11	R	30	59.8%
12	S	31	38.2%
13	T	33	22.4%

Last updated: October 1, 2023

T

New features

Tablet and large screen support
Programmable shaders
Color vector fonts
Predictive back gesture
Bluetooth LE Audio
Splash screen efficiency improvements
ART optimizations

Behavior changes

OpenJDK 11 updates
Battery Resource Utilization
Media controls derived from PlaybackState
Permission required for advertising ID
Updated non-SDK restrictions

Security and privacy

Safer exporting of context-registered receivers
Enhanced photo picker privacy
New runtime permission for nearby Wi-Fi devices
Exact alarms permission
Developer downgradable permissions
APK Signature Scheme v3.1
Better error reporting in Keystore and KeyMint

https://developer.android.com/about/versions/13

Figure 11. Android Version distribution in 2023, provided by Google, obtained in Android Studio.

At the end of the year, Google shows these statistics to Android developers to help them decide which Android version they should use. These statistics can be viewed in Android Studio at the startup. Figure 11 shows the informational window, which is displayed upon the startup. For this app, the minSdkVersion was set to 26 and the target to 33. As seen in this statistics window, at the end of 2023, around 93.7% of users could run the apps made for Android API version 26. This is a reasonable balance between the older versions' availability and the newer ones' innovations. Every Android library that was needed for this project also supported this version. The targetSdkVersion was set to 33, simply because it is reasonable to develop the app for the latest available stable version, and the available hardware also runs on the latest Android API version.

The IDE was configured to use an emulator with these versions in mind. Two devices were used during testing, but only one was emulated. Google Pixel 3a with SDK version 26 was emulated in the IDE to test the app against the smartphone with the lowest supported SDK version to check compatibility and different screen sizes to ensure that the app looks the same way on a different screen. Samsung Galaxy S22 Ultra was paired to the IDE using the WiFi Development feature, which uses the WiFi medium to install the app to the actual hardware with one button press. This feature was used a lot during the development process.

A few libraries should be mentioned, as app critical flows heavily rely on them. Retrofit library provided a convenient web client to send and receive HTTP requests and responses. They have also provided a flexible converter to handle serialization and deserialization of the content. The location package and basic Android libraries were included. The method to calculate the distance between two sets of coordinates was used to compare results from GPS and WiGLE.

3.2 Android App

After the initial experiments with positioning, it was decided that trilaterating the receiver's position using transmitters' coordinates data from WiGLE with RSSI data would be impossible, as the necessary data is absent to make accurate predictions. Furthermore, the observed difficulties in obtaining and using the coordinates measurements showed the need for a more convenient tool. The following subsection will cover the basic requirements for the app and provide a brief overview of the layout chosen. Further subsections will provide a more detailed insight into the development of each main tab.

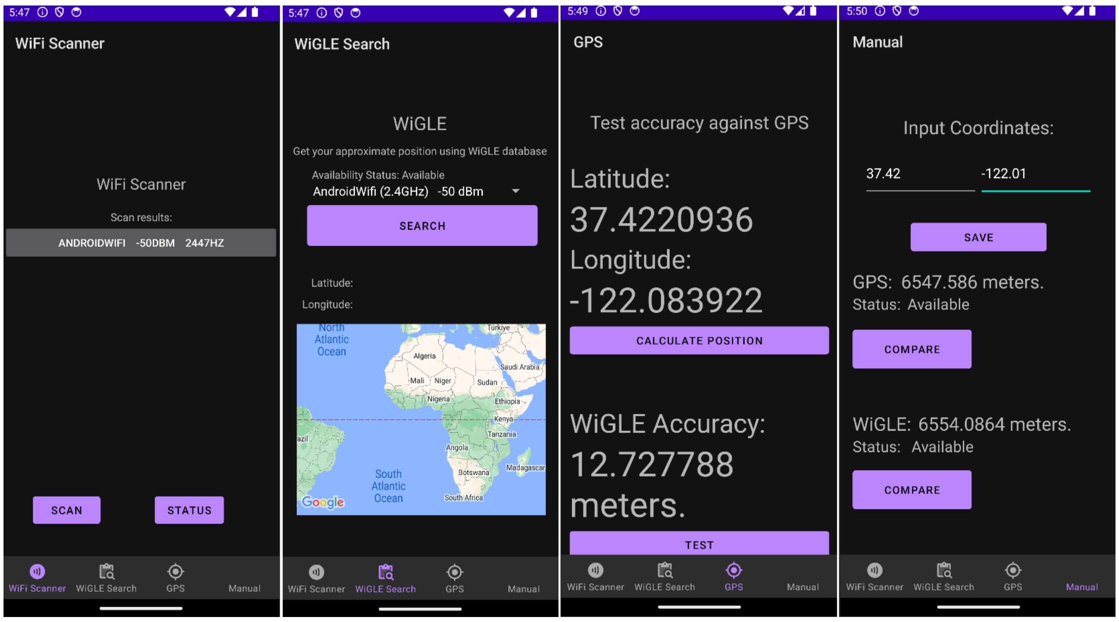
3.2.1 Setup and Basic Layout

The development of an app began with setting up the working environment, finding the official documentation and deciding on the basic app layout. There were several requirements which needed to be met. The app can:

- Capture the data of nearby WiFi networks (this includes RSSI, SSID, BSSID and operating frequency);
- Send the captured WiFi data to WiGLE;
- Receive an answer containing positional data from WiGLE
- Show the estimated WiGLE position on the map
- Use GPS to capture current positional data
- Compare GPS positional data with WiGLE positional data
- Compare manually set coordinates with GPS and WiFi data

GitHub was also set up at this stage, providing a decent versioning and allowing it to work comfortably on multiple machines.

Figure 12 shows the final build of the app. Four tabs can be separately seen: WiFi Scanner, WiGLE Search, GPS and Manual.



ANDROIDWIFI -50DBM 2447HZ

Get your approximate position using WiGLE database

Availability Status: Available

AndroidWifi (2.4GHz) -50 dBm

Latitude:

Longitude:

Test accuracy against GPS

Latitude:
37.4220936
Longitude:
-122.083922

WiGLE Accuracy:
12.727788
meters.

Input Coordinates:

37.42 -122.01

GPS: 6547.586 meters.
Status: Available

WiGLE: 6554.0864 meters.
Status: Available

Figure 12. The look of the app

WiFi Scanner acts as a home screen of the app and the first tab the user sees. By pressing the "Scan" button, the user is prompted to give the app permission to allow the app to access WiFi data and perform scans. After giving permission, the app scans all nearby WiFi networks, displaying their SSIDs and RSSIs.

The WiGLE Search tab aims to communicate with the WiGLE database and collect positional information based on the WiFi data captured in the WiFi Scanner tab. The user can select which of the captured WiFi networks will be used to request the positional data from WiGLE. By default, the app selects the strongest signal. It is possible to choose the network, as not all surrounding networks could be present in the WiGLE database. Google Maps embedded map is also present on this tab to provide a clear view on the estimated location. To allow the app to communicate with Google Maps and use the embeeded map, an API key, which are basically identifiers, must be requested from Google. Google has a separate web platform for developers that includes a lot of functionalities related to their geographical products, such as Google Places, Google Routes, Google Environment, and Google Maps. API keys were also created on this platform. They are provided freely and can be used by anyone.

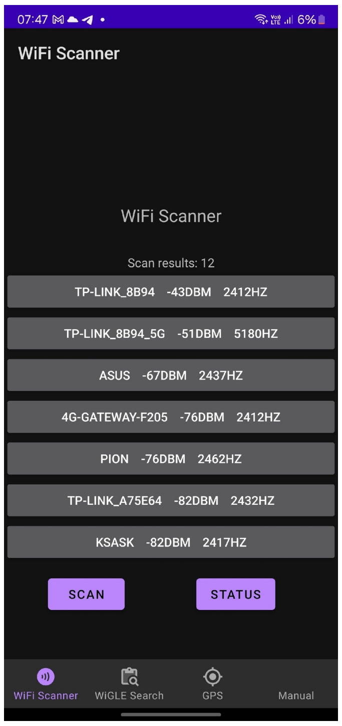
The GPS tab collects the device's GPS positional data, and it requires the user's permission to do so. After getting the WiGLE response and GPS positional data, the user can compare those two sets of estimated coordinates and get the difference in meters.

Finally, the Manual tab compares WiGLE and GPS coordinates to the user's input. It works similarly to the comparison functionality from the GPS tab, allowing the user to see the meter difference between the coordinates.

The app also has logging capabilities to collect the data in a more convenient format. Several actions result in logs, specifically performing a WiFi scan, making a WiGLE request (logs both the requested network and the answer from WiGLE), and all comparison operations. Logs are sorted in corresponding folders ("scan", "wigle", "gps", "manual") and use the timestamp as names in the format of year-month-day hour-minute-second.

The next sections will explain the technical aspects of the different tabs and their inner workings in more detail.

3.2.2 WiFi Scanner



```
Scan results: 12
TP-LINK_8B94   -43DBM   2412HZ
TP-LINK_8B94_5G   -51DBM   5180HZ
ASUS   -67DBM   2437HZ
4G-GATEWAY-F205   -76DBM   2412HZ
PION   -76DBM   2462HZ
TP-LINK_A75E64   -82DBM   2432HZ
KSASK   -82DBM   2417HZ
```

Figure 13. WiGLE tab in the latest application version.

This tab acts as the app's home screen, meaning this is the first thing the user sees after launching it. Its main components are the view of the results, which becomes scrollable once it exceeds the visible area, and two functional buttons: "Scan" and "Status".

Pressing the "Status" button displays the current status of the app's accessibility to capture the WiFi data as a short-timed pop-up on the bottom side of the screen. The "Scan" button requests permission to use WiFi from the user, scans the surrounding WiFi networks and adds those results to the results view.

These two buttons use the WifiManager class to check the WiFi state and perform WiFi operations. This class is a part of the "android.net.wifi" package, providing an interface for managing WiFi connectivity [30]. The code snippet from Listing 1 shows the usage of the "getWifiState" public method. This way, upon pressing the "Status" button, the user knows if WiFi functionality is ready.

Listing 1. Code for checking the status of the WiFi.

```
binding.statusButton.setOnClickListener {
            val statusInfo = when (wifiManager.wifiState) {
                WifiManager.WIFI_STATE_DISABLING -> "Disabling"
                WifiManager.WIFI_STATE_DISABLED -> "Disabled"
                WifiManager.WIFI_STATE_ENABLING -> "ENABLING"
                WifiManager.WIFI_STATE_ENABLED -> "ENABLED"
                else -> {
                    "Unknown"
                }
            }
```

Scanning is performed using the "startScan" public method. As a result, WifiManager keeps all found WiFi networks in a "scanResults" list. It maps the network information to the "ScanResult" class containing multiple fields. Only a few of those fields are needed, and these objects are transformed into instances of the "WifiScanResult" class, representing the scanned network model in the app structure. "SSID, "BSSID, "frequency," and "level" are retained with the data transformation and saved in the data context of the app. The code snippet from Listing 2 shows the code that starts the scanning process and maps the results to the model from the app's infrastructure.

Listing 2. Scanning code.

```
val success = wifiManager.startScan()
if (success) {
                val scanResult: List<WifiScanResult> = wifiManager.scanResults.stream()
                .map {result -> WifiScanResult(result.SSID, result.BSSID,
                result.frequency, result.level) }
                .toList();
            }
```

Here, the SSID and BSSID correspond to the SSID and BSSID from the scanned network. The frequency is the same, which is either 2.4 GHz or 5 GHz in the scope of this thesis. The level is the signal strength.

3.2.3 WiGLE Search

The WiGLE database played a pivotal role in this development. It provided the necessary positional information, which would have been impossible to get without the usage of GPS. This tab hosts the functionalities related to WiGLE communication.

Figure 14 shows the tab's appearance. The network needs to be selected in the provided spinner, which opens up a list of available WiFi networks when pressed (See the red circle in Figure 14). The app gets the list of available WiFi networks from the process in the WiFi Scanner tab. Once the scan has been performed, the found networks are available in the shared data context used in this tab.

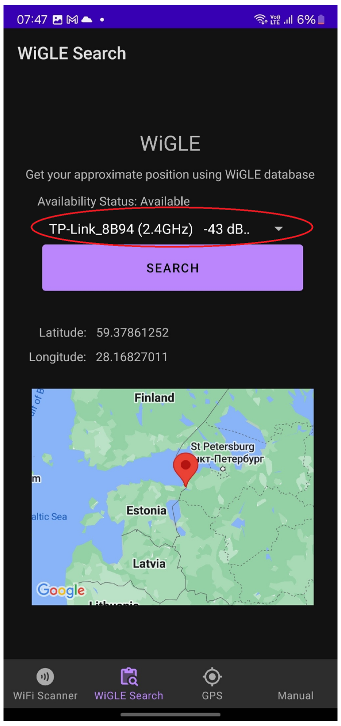


Figure 14. WiGLE tab

When the "Search" button is pressed, an HTTP request to the WiGLE database is made. The client, a class responsible for handling requests and responses from a server, was needed. The Retrofit library was used for this purpose. Retrofit is a type-safe HTTP client for Kotlin. Its defining feature is the ability to represent API endpoints by defining interfaces and annotating their methods [31]. Listing 3 shows the code snippet from the WigleApi interface class. Here, the endpoint "network/search" is defined by one method. It is also possible to introduce additional parameters to the request using the "@Query" annotation on method parameters. Here, it is used to indicate to the Retrofit that this endpoint should have "netid" as the query parameter. In the WiGLE database context, "netid" is just another way of saying "BSSID". They are the same property.

Listing 3. Defining an API point for Retrofit

```
interface WigleApi {
    @GET("network/search")
    fun getNetwork(@Query("netid") netid: String): Call<WigleResponseDto>
}
```

Retrofit also handles many other functions, making the process of creating a client easier. It handles URL construction, request dispatching, and the serialization of parameters and response objects. The integration with serialization libraries is also present [31]. Listing 4 presents how the client is defined in the RetrofitClient class, which acts as a factory for the client. The timeouts, base URL for the web server, base client, interceptors and converters are configured here. Interceptors are meant to intercept the requests or responses to perform any sort of action on them before retrieving or sending them. In this case, the only interceptor, "RequestInterceptor", simply logs any requests made through the client. Retrofit uses the base client to make HTTP calls. The converter is a class that serializes objects into JSON messages and vice versa. The default converter will suffice as the objects are plain, meaning they are simply key-value pairs.

Listing 4. Defining a Retrofit client

```
object RetrofitClient {
    private const val WIGLE_URL = "https://api.wigle.net/api/v2/"

    val okHttpClient = OkHttpClient()
        .newBuilder()
        .connectTimeout(Duration.ofSeconds(30))
        .addInterceptor(RequestInterceptor)
        .build()

    fun getClient(): Retrofit =
        Retrofit.Builder()
            .client(okHttpClient)
            .baseUrl(WIGLE_URL)
            .addConverterFactory(JacksonConverterFactory.create())
            .build()
}
```

WiGLE API contains multiple endpoints dedicated to various purposes. An API key must be used to call the endpoints. Users can create such a key on their WiGLE website profile page. The one endpoint used here is "/api/v2/network/search." This endpoint accepts GET requests containing query parameters, based on which its server performs a search, filtering out results that do not fit the provided search parameters and returning the list of networks fulfilling the conditions. For the positioning estimation to be precise enough, some property that uniquely identifies WiFi networks must be used to get exactly one result. BSSID fits this purpose and can be passed as a query parameter to their server by specifying it as a "netid" in the search string. For example, passing "/api/v2/network/search?netid="b1:b1:b1:b1:b1:b1" (this is not a proper representation, as some characters would need to be escaped when used in the URL) will yield a list with its only element being the network with that BSSID. [32]

After the "Search" button is pressed on the tab, the method "getNetworkByBSSID" is called. It accepts the network's BSSID, which is taken from the user's chosen network. Listing 5 shows this class and method.

Listing 5. Using the client to make the request

```
class WigleClient {
    private val retrofit = RetrofitClient.getClient();
    private val wigleApi = retrofit.create(WigleApi::class.java)

    fun getNetworkByBSSID(bssid: String) : WigleDto? {
        val wigleResponse = wigleApi.getNetwork(bssid)
```

```
            .execute()
        if (!wigleResponse.isSuccessful) {
            return null;
        }

        if (wigleResponse.body()?.results?.size == 0) {
            return null;
        }
        return wigleResponse.body()?.results?.get(0);
    }

}
```

The client is created using the code from Listing 4, and the implementation of API defined in Listing 3 is made using the "create" method. After that, the only remaining thing is to call the endpoint, providing it a BSSID, and validate response. The response is considered successful if it contains a non-null body with at least one result. In case of an unsuccessful result, null is returned, while the successful call results in one instance of "WigleDto" returned. This class is what the response message is being deserialized in and contains positional data (latitude and longitude, SSID, quality of signal and BSSID).

When the controller class receives the instance of WigleDto, it is converted into "WigleCalculatedLocation," which contains the same fields but is made to represent data inside the app specifically and saved to the shared data context. This tab's text elements are listening for the updates in the data context. When they detect that the instance of WigleCalculatedLocation is being saved, they also update the coordinates on the screen. If the controller class receives null, the "Not found" message is displayed in the coordinates window instead.

This tab also introduces an interactable map provided by Google Maps libraries (see Figure 14). When the controller receives a valid response from the WigleClient, the point of corresponding coordinates is added to the map (the location marker in Figure 14).

3.2.4 GPS

One critical aspect of this work is assessing the accuracy of GPS against the WiGLE-provided location. This tab is seen in Figure 15 and serves two purposes: data collection and comparison. When the "Calculate Position" button is pressed, the controller uses the "LocationServices" class to get the user's most recent location. If the location is fetched successfully, an instance of "GpsCoordinates" is created. This class represents the last known GPS location, and it only contains latitude and longitude. It is saved in the shared data context and displayed in the corresponding text area.

The "Test" button starts the comparison process. The gathered GPS coordinates are compared to the ones received as a response from the WiGLE database, and the difference is displayed in meters. If WiGLE Search has not been performed yet, the pop-up informs the user that the WiGLE data is missing. As GPS and WiGLE data are stored in the same data context, they are available and accessible from any tab.

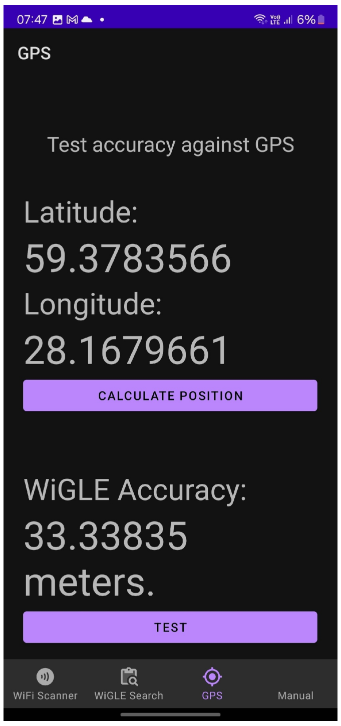


Figure 15. GPS tab

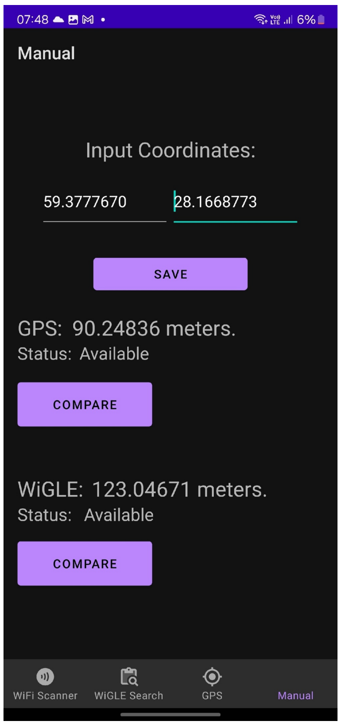
The "Location" class from the "Android.location" package calculates the difference in meters between the coordinates. It uses its own internal methods to do so by representing location coordinates in the WGS84 standard. [33]

3.2.5 Manual

The Manual tab allows users to compare the previously gathered GPS and WiGLE data against manually entered coordinates. The purpose of this tab is to provide some definitive target coordinates against which to test accuracy. GPS is not reliable, as location estimation could be inaccurate. By using this tab the user can test both GPS and WiGLE coordinates if they know their exact location (provided by a GPS device or some other way). It is seen in Figure 16.

Users can input the coordinates into the dedicated text input area. If they are valid, their coordinates are saved in the shared data context as an instance of the GpsLocation class. Validation checks that the entered coordinates fall into the range of possible latitude and longitude values. This means if the latitude is not between -90 and 90 or if the longitude is not between -180 and 180, then the validation fails, and the user is informed that the entered coordinates are invalid.

This tab has two distinct areas dedicated to GPS and WiGLE comparison. Both


Manual

Input Coordinates:

59.3777670 28.1668773

GPS: 90.24836 meters.
Status: Available

WiGLE: 123.04671 meters.
Status: Available

Figure 16. Manual tab

work similarly. Each one has a special text area, informing the user about the status of this particular comparison. For the comparison to be available, both manually entered coordinates and corresponding GPS or WiGLE coordinates should be present in the shared data context. When the requirements are met, the status says "Available" and the buttons become active. Distance in meters is calculated the same way it is done in the GPS tab.

3.2.6 Logging

During testing, it became apparent that gathering the results was quite difficult. To preserve the results for further analysis, a screenshot for each tab was taken during each measurement session, which proved inconvenient. To provide simpler data analysis, logging was introduced into the app. The app logs the following actions:

- Performing a WiFi scan (See Appendix II.A);
- Requesting the WiGLE database search through the client and receiving a response (See Appendix II.B);
- Comparing the accuracy of GPS against WiGLE (See Appendix II.C);

- Comparing the accuracy of manually entered coordinates against WiGLE/GPS (See Appendix II.D).

Logs are kept in the device's internal memory and separated into four folders: "gps," "manual," "wifiScan," and "wigle." Each log has a timestamp in its name, with accuracy up to seconds. Some parts of it might need clarification for the reader. In Listing 7 a few fields can be misleading, such as 'trilat' and 'trilong'. They are basically the latitude and longitude provided by WiGLE. Here is also the field 'qos', which is the quality of the signal. At some point, there was a thought that this might be useful for the app functionality, but it is not used anywhere. The logs also specify the SSID used, so it would be easy to look at the log and see what SSID was used to request the estimation from WiGLE.

Listing 6 has a lot of fields, which might confuse. WiGLE collect a wide range of data regarding the networks and a lot of those fields are left unused, although the app logs every bit of information in case it might be useful.

Listing 9 and Listing 10 are from the Manual tab. One log is for GPS comparison and the other one is for WiFi comparison. The thought behind this is to make logs more readable by separating those two comparisons into separate logs. Firstly, the difference in meters is logged. Then, the basic information about the comparison actors is also logged. In the case of GPS and manual coordinates, they are both logged as GpsLocation with just latitude and longitude. The WiGLE comparison actor is logged as WigleCalculatedLocation with already seen fields.

4 Results

The following chapter describes the results of the finished application, including testing methods and conclusions, and discusses a few solutions for problems that arose during the research.

The research resulted in an app that can be used as a tool for analysing the accuracy of using the surrounding WiFi networks and an internet connection to estimate the location without the usage of GPS. The app is available through a built APK file from the Github[1] and can be installed on Android devices with an API version above or equal to 26, the recommended however is 33. After installation, the app will request a few permissions related to WiFi and GPS functionalities. The corresponding services must also be available in order to use the main app functionalities.

4.1 Testing

The app passed multiple testing stages during its development. The testing process consisted of visiting areas of cities with different densities of buildings/people/networks and using the app to get estimations. Before the logging feature was implemented, the results were saved as screenshots of different tabs.

Testing was done in three cities: Narva, Tallinn, and Budapest. Most of the tests were conducted in Narva. Budapest was chosen to test the app in a completely different environment, specifically, in a much denser, bigger and more populated environment. Additionally, as the tests in Budapest were done by the tester, this helped to see how the tester would go through the installation process and interact with different flows in the app.

As it was mentioned before, most of the readings were done in Narva using a Samsung Galaxy S22 Ultra. The process for testing was the following:

1. Find a spot, where at least one WiFi signal is available;
2. Perform a WiFi scanning for surrounding networks;
3. Choose the network to request WiGLE information. If the network is not in the database, try another;
4. Go to the GPS tab, save the GPS coordinates and compare them to WiGLE;
5. Manually gather the coordinates by pinpointing a location using Google Maps;
6. Input the location into the Manual tab, and save it;
7. Compare the manual location to the GPS and WiGLE coordinates;

[1]https://github.com/nbahh/wifi-positioning-app

8. Repeat steps 3-7, but choose the network with another frequency, so that it is possible to compare the accuracy with different frequencies;

9. Examine logs;

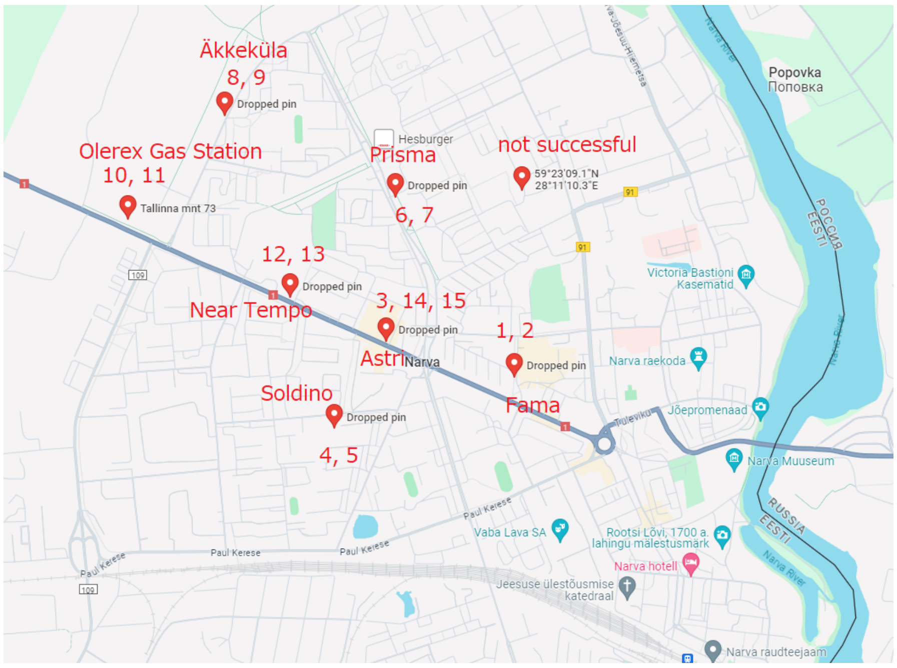


Figure 17. Test locations in Narva

Figure 17 shows the testing locations chosen for Narva. In total, there are eight sites where it was attempted to get the reading, and fifteen readings were made - two readings for most testing sites, in order to compare the influence of the operating frequency of the network. One reading was made for a 2.4GHz network, and another for a 5GHz network. Each site on every map has a number next to it, this is the unique ID for the network, which are assigned later in the Findings section. Fourteen of those readings were successful, while one reading failed, as the WiFi networks that surround the areas were not present in the WiGLE database. The locations were chosen to be at different distances from the city centre; as such, the testing sites could be arranged in a line from the centre to the outer regions of the city, with the outermost site located at around 2 km from the city centre.

To improve the accuracy of the WiGLE estimation the WiGLE Wardriving App was used in the city. The process involved passing different areas of the town while capturing as many WiFi networks as possible and sending them to the WiGLE database so that they could perform their estimation. This app is explained in section 2.3.2 of this paper. As a

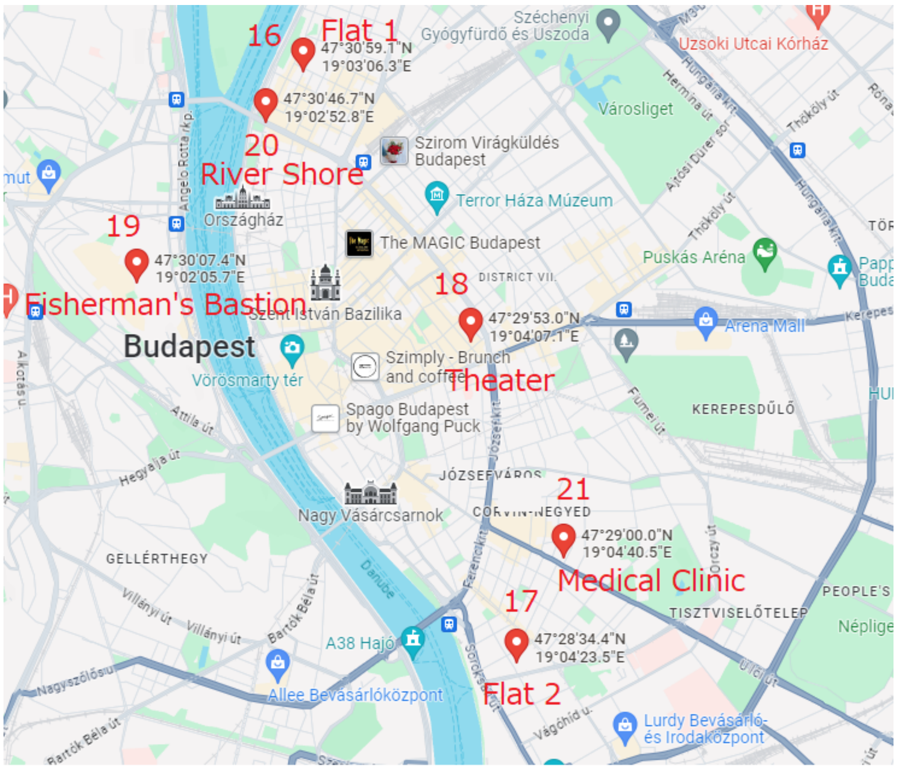

16
Flat 1
47°30'59.1"N 19°03'06.3"E
47°30'46.7"N 19°02'52.8"E
20
River Shore
19
47°30'07.4"N 19°02'05.7"E
Fisherman's Bastion
18
47°29'53.0"N 19°04'07.1"E
Theater
21
47°29'00.0"N 19°04'40.5"E
Medical Clinic
17
47°28'34.4"N 19°04'23.5"E
Flat 2
Budapest
Danube

Figure 18. Test locations in Budapest

result of using this Wardriving app, around 1000 networks were added to the WiGLE database.

Seven readings were done in Budapest. Unfortunately, only two of those were successful. Testing sites for Budapest were chosen to be all near the city centre in order to measure the accuracy in the denser area. Figure 18 shows those locations on the map. These locations are more interesting, as several measurements took place inside big and highly popular places, such as the clinic and the theatre. Some readings were taken in areas which are full of tourists, like the river shore and the Bastion, and some readings were taken in a fairly normal environment, such as flats.

A volunteer performed the reading in Budapest. The volunteer was instructed on how to install and use the app. This was successful, and it indicated that the app works on a device other than the Samsung Galaxy S22, as the volunteer was using a Realme 8 5G. These devices have different Android versions and different screen sizes, which indicates good compatibility. The feedback provided by the tester was quite useful and helped to improve the app.

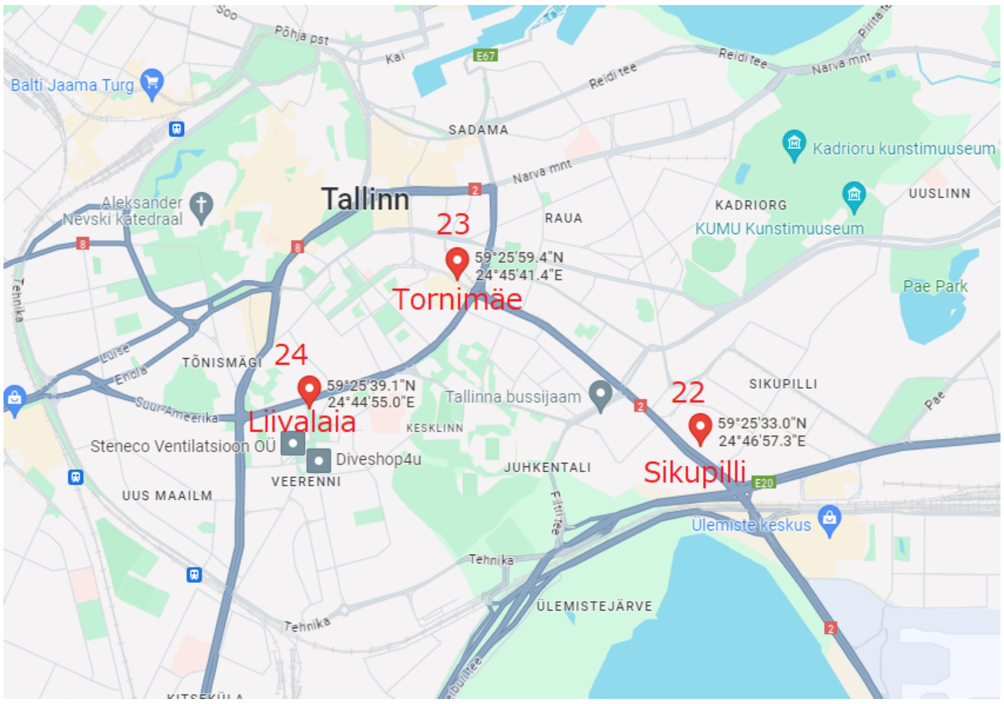


Figure 19. Test locations in Tallinn

Finally, Tallinn had the fewest testing sites. Only three spots were measured with the app with all of them being a successful reading. All readings were taken near the city centre. Its testing sites are visible in Figure 19.

4.2 Findings

Most of the information was collected from Narva; as such, the most important findings are related to this city. Table 1 shows some of the information extracted from the logs, which were collected while testing in Narva. Table 3 and Table 4 show results in the same format but from Budapest and Tallinn respectively. There are fewer results from those cities, but they still have a few interesting points.

Table 1. General information in the table format from Narva.

ID	Place	SSID	Freq	WiGLE	GPS	WiGLE Acc	GPS Acc
1	Fama	48	2.4GHz	53.5m	22.8m	MEDIUM	MEDIUM
2	Fama	tp link	5GHz	164.7m	22.9m	LOW	MEDIUM
3	Astri	IBSD	5GHz	66.1m	5.1m	MEDIUM	HIGH
4	Soldino	Puskina23	2.4GHz	934.5m	4.0m	INCORRECT	HIGH
5	Soldino	masja_5G	5GHz	46.3m	3.4m	MEDIUM	HIGH
6	Prisma	Boss	2.4GHz	36.6m	40.2m	MEDIUM	MEDIUM
7	Prisma	Igor 5G	5GHz	41.5m	14.6m	MEDIUM	HIGH
8	Äkkeküla	#Telia-0CBACA	5GHz	10.7m	8.8m	HIGH	HIGH
9	Äkkeküla	TP-LINK_A25A	2.4GHz	38.5m	8.8m	MEDIUM	HIGH
10	Gas Station	OLX-FREE	2.4GHz	42.4m	25.8m	MEDIUM	MEDIUM
11	Gas Station	OLX-OTHER	5GHz	22.3m	20.9m	MEDIUM	MEDIUM
12	Near Tempo	MyVolvor6kXHL	2.4GHz	689.4m	10.1m	INCORRECT	HIGH
13	Near Tempo	Fill_Point_26	5GHz	302.5m	10.1m	LOW	HIGH
14	Inside Astri	Tasuta Wifi Astri	5GHz	41.6m	26.3m	MEDIUM	MEDIUM
15	Inside Astri	optium	2.4GHz	30.8m	32.5m	MEDIUM	MEDIUM

Each table consists of the following columns:

- ID - an identification number given to each reading in the table for easier reference;
- SSID - the name of the network;
- Freq - the operating frequency of the network - either 2.4 or 5 GHz;
- WiGLE - the distance from manually entered coordinates to the WiGLE coordinates, computed using the app;
- GPS - the distance from manually entered coordinates to the GPS coordinates, computed using the app;
- WiGLE Acc - the accuracy level, which is given to the WiGLE estimation;

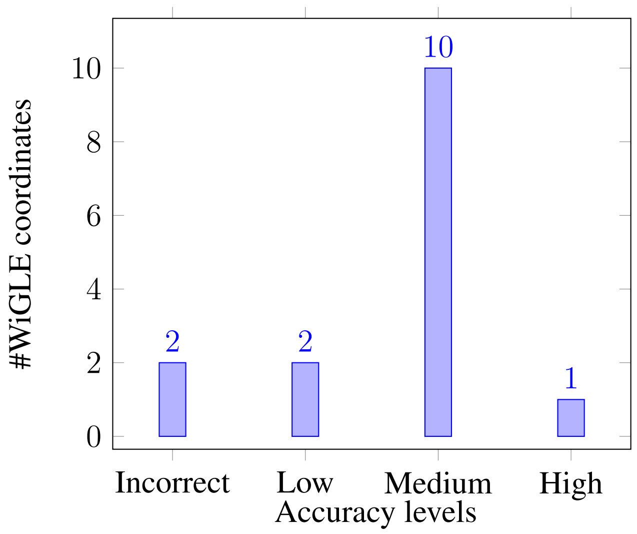
- GPS Acc - the accuracy level, which is given to the GPS-acquired coordinates;

Figure 17 also specifies where exactly the measurements were taken by using the IDs assigned in the table in Table 1. Accuracy level is defined in the context of this paper as a means to categorize WiGLE estimations and GPS coordinates into several groups so that it can be used to make some conclusions regarding the accuracy of said method. Table 2 defines the accuracy levels and explains how it can be assessed. During testing, the manually entered coordinates were considered a target against which the precision was tested. If the distance between target coordinates and WiGLE estimated coordinates is 20m or less, then the estimation is considered to be a "HIGH" level. If the distance lies between 20 meters and 100 meters, the accuracy is "MEDIUM". Between 100 meters and 500 meters is "LOW" accuracy. Everything beyond, meaning more than 500 meters, is considered to be "INCORRECT", as the error is too high to consider this estimation correct. Boundary values for these groups were chosen so that every group was represented reasonably. The highest accuracy WiGLE estimation was network 8 from Table 1, so the boundary for the "HIGH" level was set to 20 meters to place this network and a few others into the "HIGH" accuracy bucket. The "MEDIUM" level was limited to 100 meters because the average of the readings excluding the abnormal ones, like 600 meters or 900 meters, is 61 meters, so it is reasonable to consider a deviation of more than 50 meters to be "LOW" level. For the sake of making the boundary values easier to read, the boundary for the "LOW" level was rounded down to 100 meters.

Table 2. Accuracy levels definition.

Distance (x meters)	Accuracy level
x <= 20	HIGH
20 < x <= 100	MEDIUM
100 < x <= 500	LOW
500 < x	INCORRECT

This method of assessing the accuracy levels shown in Table 2 was used to construct a graph showing how accurately WiGLE estimation was performed. Figure 20 shows this distribution. It is clearly seen that WiGLE estimations are less accurate than the GPS ones. It is further confirmed by the average values, with WiGLE estimations being 176.8 meters on average from the target coordinates, while GPS averages 17.1 meters from the target coordinates.



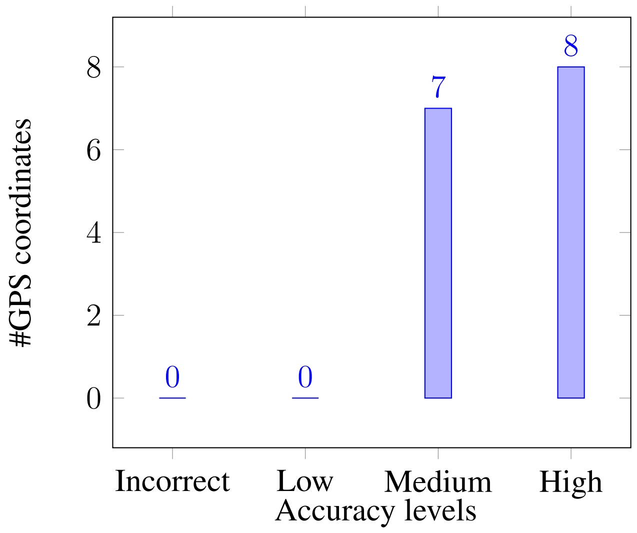


Figure 20. Distribution of networks by accuracy level in Narva.

Table 3. General information in the table format from Budapest.

ID	Place	SSID	Freq	WiGLE	GPS	WiGLE Acc	GPS Acc
16	Flat 1	TELEKOM-0599	5GHz	15.0m	9.2m	HIGH	HIGH
17	Flat 2	TELEKOM9 2.4G	2.4GHz	-	-	-	-
18	Theater	VODAFONE-0834	2.4GHz	27.8m	-	MEDIUM	-
19	Bastion	ADRIENS S23 ULTRA	2.4GHz	-	-	-	-
20	River Shore	REDMI NOTE 8 PRO	2.4GHz	-	-	-	-
21	Medical Clinic	S EPULET	5GHz	126.8m	210.9m	LOW	LOW

Budapest observed a different conclusion. Table 3 shows the results of readings in Budapest. Several cells are missing data, because there were some complications which have different causes. There were three main failure causes: WiFi scanning issues, WiGLE estimation issues and GPS issues. The distribution of those issues is described in Figure 21. "GPS issue" means that those reading encountered problems while reading GPS, this can happen when the signal reception is bad or there are in general some sort of problem with the smartphone's GPS service. "WiGLE issue" means that there were problems with WiGLE estimations. This can be either that the networks data is absent from the WiGLE Database or that the server is unavailable. "WiFi Scan issue" means that something went wrong while scanning the WiFi networks. This also means that the app scanned something that it should normally avoid, such as smartphone's hotspots. Some of the cells in Table 3 are empty in relation to the errors depicted in Figure 21. Network 17, 19 and 20 are missing the WiGLE estimation data, as networks 19 and 20 suffered a problem with WiFi scanning, specifically, smartphone hotspots were picked for the WiGLE estimation, while network 17 had used a normal WiFi network, but that network was missing from the database, which classifies its issue as the WiGLE issue. The GPS comparison was not done in case of WiFi scanning issues, which happened with networks 19 and 20, or in case of WiGLE issues, which happened with network 17. GPS issue happened with network 18, as the signal reception was bad, so in that is why in total 4 networks have missing GPS data.

The average WiGLE estimation in the area, is 56.5 meters from the target coordinates, while GPS shows 110.05 meters from them (see Table 3). However, it is most likely caused by such a small sample group, where abnormal deviation can pollute the results. The network with an ID equal to 21 is one such abnormality, where the GPS showed worse results than the WiGLE estimation. This reason might be that the readings were taken inside the clinic (see Figure 18), where the GPS signal is obstructed by the building, which might have resulted in low accuracy. When taking a better look at networks 19 and 20 in Table 3, it is clear that those network sources are smartphones. Specifically, a smartphone that acts as a network hotspot. The probability of them being in the WiGLE database is quite low, as they move very frequently and their SSID can be changed very easily in the smartphone's option menu. And indeed, when requesting a WiGLE

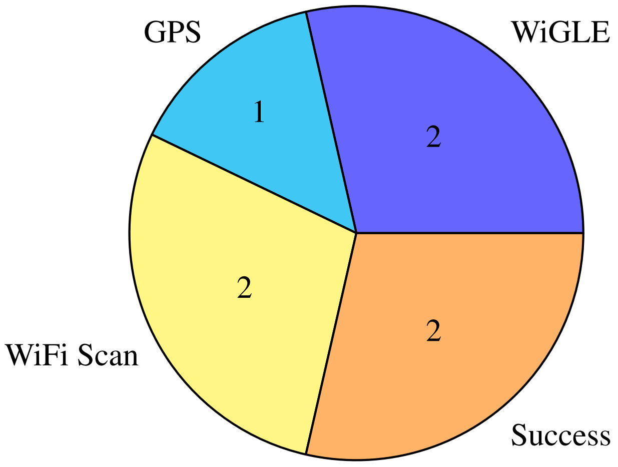


Figure 21. Reading results from Budapest

estimation, those networks yielded no results. However, this problem was classified as a WiFi Scanner problem, as such networks should not be used for the estimation due to their unreliable nature. At first, the app chose the strongest signal and it was not possible to manually choose the networks to use. After this, however, this feature was implemented.

Table 4. General information in the table format from Tallinn.

ID	Place	SSID	Freq	WiGLE	GPS	WiGLE Acc	GPS Acc
22	Sikupilli	POLA-LASNA2-SK1	5GHz	525.3m	72.1m	INCORRECT	MEDIUM
23	Tornimäe	TELIA-8D0DE2	2.4GHz	9m	9.7m	HIGH	HIGH
24	Liivalaia	Smartphone_connect_73a28a	2.4GHz	21.2m	11.1m	MEDIUM	HIGH

Tallinn, however, followed Narva's tendencies with the average being 185.2 meters for the WiGLE estimation and 92.9 meters for GPS (as seen in Table 4).

However, as Table 1 shows, there are occasions when WiGLE produces more accurate results. Networks 6 and 15, categorized as Medium accuracy level networks, are such examples from Narva. Networks 21 and 23 are similar examples from Budapest and Tallinn respectively.

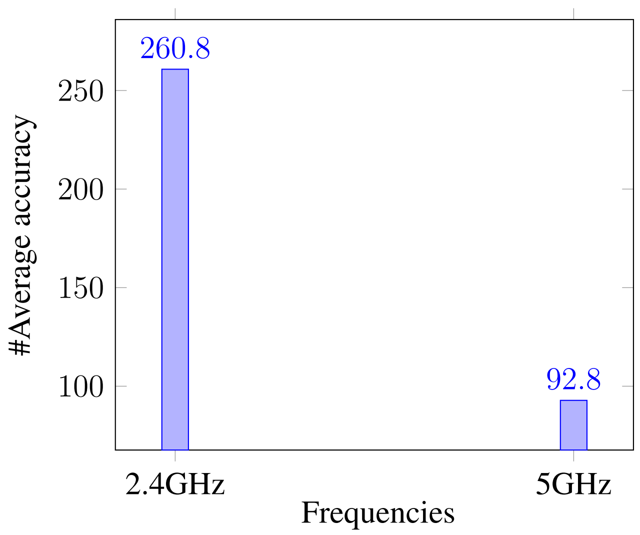


Figure 22. Average of networks based on frequencies

Interestingly enough, the frequencies seem to influence the estimations quite considerably. Figure 22 demonstrated a significant difference in accuracy between the 2.4 GHz and 5 GHz frequency networks. The y-coordinate represents the distance between the target and estimated coordinates, so a higher number means a less accurate result. This could indicate that WiGLE is more accurately localizing 5 GHz networks than 2.4 GHz networks. One possible reason might be that due to their nature, 2.4 GHz networks can be detected farther away than 5 GHz networks. A bigger radius of detection implies more room for mistakes.

5 Discussion

A few interesting difficulties came up during the research process, which redirected the project and required us to come up with new solutions. One example is the tourist hotspot problem described in the Results section. At that stage, the app did not have the functionality that allows the user to choose the network they wanted to use for WiGLE estimation. The app chose the strongest available network instead. This is the reason why the tester had so many difficulties with getting the WiGLE estimations in areas with high tourist density. As mobile hotspots are highly movable WiFi points, the possibility of one of those networks being in the WiGLE database is low. And if the network is present in the database, then that could produce an incorrect estimation instead.

After receiving the feedback from the tester, the app received an update to allow the user to choose the WiFi network to avoid such situations in the future. However, there is still space for app updates. One possible addition to the app might be the addition of local database storage for scanned networks. This way, the app can work in offline mode if the networks have already been seen or even sent to the WiGLE database to improve their network coverage.

There is more room for research regarding the 2.4 GHz and 5 GHz networks. Results show better accuracy with 5 GHz networks, which was mentioned before, as well as the possible reason. It would be interesting to test the accuracy of 6 GHz networks. Unfortunately, no networks with such operating frequencies were found in any of the testing sites. Would the accuracy of WiGLE be even higher with a 6 GHz network, given their even smaller radius?

There were also a lot of difficulties with the design of the app. It was quite a challenge to design a cohesive app interface which would not break in case the target phone has a screen with a different size. There were multiple occasions when the app APK file was sent to some person's phone to test its functionality, only to be stopped by the interface. Buttons especially were vulnerable to the screen size change and frequently flew out of the screen.

During testing in Budapest, there was also one case, where the GPS failed, but WiGLE was still able to give an accurate estimation. Network 18 from Table 3. The tester reported that the GPS was not functional, but they were still able to estimate the location using WiGLE service along with the app's WiFi Scanner. The reading was taken inside the theatre (see Table 3 or Figure 18), so the error in the GPS system might have been caused by the obstruction of the signal by the building.

Some readings were really inaccurate. Table 1 holds an especially strange network, namely number 4. It is not special in any regard in comparison to the other networks, however, the deviation is more than 900 meters. This is really interesting, as the reading was taken in the fresh air with no obstructions. It is still unknown how such an error was produced, but most probably it is related to how WiGLE performs its estimation when the network enters its database. The other possible explanation is that when this network was saved in the WiGLE database, its transmitter was located in the other area. An owner of this network might have moved out to another apartment, which would have explained some strong deviations. One potential solution might be to limit the year range for results to have only up-to-date information. WiGLE API provides the capability to restrict the search results only to the one after a specific date. Another potential solution could be to apply the weighted approach, which basically means do readings of multiple AP, get their positional information from the WiGLE database, and choose the closest one to the user.

Another way to improve accuracy might be to capture the data and send it to WiGLE more frequently. If the database updates more frequently, then the locations stay up to date, and the estimation can incorporate any updates or changes made to the localization algorithm by WiGLE. Multiple attempts at capturing the data can produce more accurate data. Also, the WiGLE database have the cellular data available, which could potentially help to boost accuracy.

6 Conclusion

For this thesis, common positioning methods were analysed and explained, such as trilateration, triangulation and fingerprinting. As a part of the research, the topic of WiFi and the contents of its wireless transmission was also touched upon. WiGLE, one of the biggest network databases, was explored and heavily used throughout the thesis. As a result of this research, an Android app was developed to act as a tool to investigate the phenomena of WiFi and GPS positioning.

The app is fully functional and has been tested on different devices in different cities, including Tallinn, Narva and Budapest. The app is available in Github, which was one of the main development environments used for the purposes of this paper. The app requires at least the API version of 26 to run, and the target version is 33. It holds several vital functionalities, such as a WiFi scanner, GPS service to save the coordinates in the app along with the comparison tool, a client to communicate with the WiGLE database and a separate tab to compare the result to manually entered coordinates. Using this app, 24 readings were taken and analysed, which showed some interesting points, such as how the efficiency of WiGLE estimations change with 2.4 GHz and 5 GHz networks.

However, there is still a lot of room for improvement. The app interface is not very stable and can still be broken for several screen sizes. The overall looks could be improved. The localization algorithm can probably be improved to produce more accurate results.

In the future, more improvements could be done on positioning algorithms using WiFi, because there are scenarios in which relying on GPS becomes unsustainable, especially inside of buildings and highly dense areas.

References

[1] Gitnux. *Google Maps Usage Statistics 2023: The Most Important Facts*. Market Data Coverage. https://blog.gitnux.com/google-maps-usage-statistics/. Mar. 2023.

[2] Joseph D’Souza. *The most widely used navigation application worldwide is Google Maps*. Press Release article based on Enterprise Apps Today data. https://www.mynewsdesk.com/11press/pressreleases/the-most-widely-used-navigation-application-worldwide-is-google-maps-3234407. Feb. 2023.

[3] United States Department of Transportation. *Satellite Navigation - GPS - How It Works*. https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps/howitworks.

[4] Khavari A. et al Tabatabaei A. Mosavi M.R. “Reliable Urban Canyon Navigation Solution in GPS and GLONASS Integrated Receiver Using Improved Fuzzy Weighted Least-Square Method”. In: *Wireless Pers Commun* 94 (June 2017), pp. 3181–3196.

[5] Xuan Du and Kun Yang. “A Map-Assisted WiFi AP Placement Algorithm Enabling Mobile Device’s Indoor Positioning”. In: *IEEE Systems Journal* PP (Mar. 2016), pp. 1–9. DOI: 10.1109/JSYST.2016.2525814.

[6] Jongtaek Oh and Jisu Kim. “AdaptiveK-nearest neighbour algorithm for WiFi fingerprint positioning”. In: *ICT Express* 4.2 (2018). Artificial Intelligence and Machine Learning Approaches to Communication, pp. 91–94. ISSN: 2405-9595. DOI: https://doi.org/10.1016/j.icte.2018.04.004. URL: https://www.sciencedirect.com/science/article/pii/S240595951830050X.

[7] Fuqiang Gu et al. “Indoor Localization Improved by Spatial Context - A Survey”. In: *ACM Computing Surveys* 52 (July 2019), 64:1–35. DOI: 10.1145/3322241.

[8] Jin Zheng, Kailong Li, and Xing Zhang. “Wi-Fi Fingerprint-Based Indoor Localization Method via Standard Particle Swarm Optimization”. In: *Sensors (Basel, Switzerland)* 22 (July 2022). DOI: 10.3390/s22135051.

[9] A. T. Hudak et al. “Nearest neighbor imputation of species-level, plot-scale forest structure attributes from lidar data”. In: *Remote Sensing of Environment* 112 (5 2008), pp. 2232–2245. DOI: 10.1016/j.rse.2007.10.009.

[10] N. Farjallah and A. Sghaier. “Machine learning based modelling of economic growth and quality of governance: the mena region”. In: (2022). DOI: 10.21203/rs.3.rs-1772256/v1.

[11] Y. Petscher and S. Koon. “Moving the needle on evaluating multivariate screening accuracy”. In: *Assessment for Effective Intervention* 45 (2 2018), pp. 83–94. DOI: 10.1177/1534508418791740.

[12] IBM. *What is random forest?* https://www.ibm.com/topics/random-forest.

[13] Zhang X Zheng J Li K. “Wi-Fi Fingerprint-Based Indoor Localization Method via Standard Particle Swarm Optimization”. In: *Sensors (Basel)* (July 2022). DOI: 10.3390/s22135051.

[14] Ejiofor H. C Oguejiofor O. S Aniedu A. N and Okolibe A. U. “Trilateration Based localization Algorithm for wireless Sensor Network”. In: *IJISME* 1 (Sept. 2013), pp. 21–27. ISSN: 2319-6386.

[15] R. Safwat et al. “Fingerprint-based indoor positioning system using ble: real deployment study”. In: *Bulletin of Electrical Engineering and Informatics* 12 (1 2023), pp. 240–249. DOI: 10.11591/eei.v12i1.3798.

[16] Yapeng Wang et al. “Bluetooth positioning using RSSI and triangulation methods”. In: Jan. 2013, pp. 837–842. ISBN: 978-1-4673-3131-9. DOI: 10.1109/CCNC.2013.6488558.

[17] Nur Rose, Low Tan, and Muneer Ahmad. “3D Trilateration Localization using RSSI in Indoor Environment”. In: *International Journal of Advanced Computer Science and Applications* 11 (Jan. 2020). DOI: 10.14569/IJACSA.2020.0110250.

[18] GISGeography. *How GPS Receivers Work – Trilateration vs Triangulation*. https://gisgeography.com/trilateration-triangulation-gps/. Sept. 2022.

[19] IEEE. “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”. In: *IEEE* (2016).

[20] Vishal Gupta and Mukesh Kumar Rohil. “Bit-Stuffing in 802. 11 Beacon Frame: Embedding Non-Standard Custom Information”. In: *International Journal of Computer Applications* 63 (Feb. 2013), pp. 6–12. DOI: 10.5120/10436-5115.

[21] I. Dhanapala et al. “Modeling wifi traffic for white space prediction in wireless sensor networks”. In: (2017). DOI: 10.1109/lcn.2017.30.

[22] Hosam Alamleh. “Unobtrusive Location-Based Access Control Utilizing Existing IEEE 802.11 Infrastructure”. PhD thesis. May 2019. DOI: 10.13140/RG.2.2.34451.50725.

[23] J. Martin et al. “A study of mac address randomization in mobile devices and when it fails”. In: *Proceedings on Privacy Enhancing Technologies* 2017 (4 2017), pp. 365–383. DOI: 10.1515/popets-2017-0054.

[24] WiGLE. *WiGLE FAQ*. https://wigle.net/faq.

[25] Jamie Juviler. *What Is an API Endpoint? (And Why Are They So Important?)* `https://blog.hubspot.com/website/api-endpoint`.

[26] DataHub. *CSV - Comma Separated Values*. `https://datahub.io/docs/data-packages/csv`.

[27] Google. *Kotlin Overview*. `https://developer.android.com/kotlin/overview`.

[28] Python. *Python Overview*. `https://www.python.org/doc/essays/blurb/`.

[29] Google. *About Android Jetpack*. `https://developer.android.com/jetpack/getting-started`.

[30] Google Android Studio. *WifiManager documentation*. `https://developer.android.com/reference/android/net/wifi/WifiManager`.

[31] Retrofit. *Retrofit Client website*. `https://square.github.io/retrofit/`.

[32] WiGLE. *WiGLE API documentation*. `https://api.wigle.net/swagger`.

[33] Google Android Studio. *Android Developer Location Documentation*. `https://developer.android.com/reference/android/location/Location`.

7 Appendix

7.1 I. WiGLE Database Entry Example

```
    {
            "trilat": 59.38251114,
            "trilong": 28.16660690,
            "ssid": "Elion",
            "qos": 0,
            "transid": "20050822-00000",
            "firsttime": "2001-01-01T00:00:00.000Z",
            "lasttime": "2005-08-22T18:00:00.000Z",
            "lastupdt": "2005-08-22T16:00:00.000Z",
            "netid": "00:02:2D:B2:D7:98",
            "name": "Neste-A24",
            "type": null,
            "comment": null,
            "wep": "N",
            "bcninterval": 0,
            "freenet": "?",
            "dhcp": "?",
            "paynet": "?",
            "userfound": false,
            "channel": 9,
            "rcois": null,
            "encryption": "none",
            "country": "EE",
            "region": "Ida-Viru maakond",
            "housenumber": "44",
            "postalcode": "21005",
            "road": "Tallinna mnt",
            "city": null
        }
```

7.2 II. Logs from the app

7.2.1 II.A WiFi Scanner Log

Listing 6. WiFi Scanner log (BSSIDs hidden for privacy)

```
SSID: "Wallet_May_Cry_Peak_Of_Thesis_5G", BSSID: ac:4c:a5:**:**:**,
capabilities: [WPA2-PSK-CCMP+TKIP][RSN-PSK-CCMP+TKIP][WPA-PSK-CCMP+TKIP]
[ESS][WPS], level: -36, frequency: 5220, timestamp: 19296281705,
distance: ?(cm), distanceSd: ?(cm), passpoint: no,
ChannelBandwidth: 0, centerFreq0: 5220, centerFreq1: 0,
standard: 11ac, 80211mcResponder: is not supported,
Radio Chain Infos: null, interface name: wlan0

SSID: "05a1e0", BSSID: 00:71:c2:**:**:**, capabilities:
[WPA2-PSK-CCMP+TKIP][RSN-PSK-CCMP+TKIP][WPA-PSK-CCMP+TKIP][ESS][WPS],
level: -76, frequency: 2412, timestamp: 19296281700,
distance: ?(cm), distanceSd: ?(cm), passpoint: no,
ChannelBandwidth: 0, centerFreq0: 2412, centerFreq1: 0,
standard: 11n, 80211mcResponder: is not supported,
Radio Chain Infos: null, interface name: wlan0
```

7.2.2 II.B WiGLE Search Log

Listing 7. How WiGLE requests are logged (BSSIDs hidden for privacy)

```
SSID used: TP-Link_8B94_5G, -----> WigleCalculatedLocation(trilat=59.37862778,
trilong=28.1688633, ssid='TP-Link_8B94_5G', qos=4, netid='B0:BE:76:**:**:**')
```

7.2.3 II.C GPS Log

Listing 8. GPS log (BSSIDs hidden for privacy)

```
Difference: 930.5388 meters,
gps: GpsLocation(latitude=59.377205, longitude=28.1727997)
------
wigle: WigleCalculatedLocation(trilat=59.37882233,
trilong=28.18885994, ssid='Puskina23', qos=0,
netid='C0:25:E9:**:**:**')
```

7.2.4 II.D Manual Logs

Listing 9. Manual log - for GPS comparison

```
Difference: 3.4203334 meters,
gps: GpsLocation(latitude=59.3771981, longitude=28.172673)
------
manual: GpsLocation(latitude=59.3771952, longitude=28.1727329)
```

Listing 10. Manual log (BSSIDs hidden for privacy) - for WiGLE comparison

```
Difference: 934.47577 meters,
wigle: WigleCalculatedLocation(
trilat=59.37882233, trilong=28.18885994, ssid='Puskina23',
qos=0, netid='C0:25:E9:**:**:**')
------
manual: GpsLocation(latitude=59.3771952, longitude=28.1727329)
```