WIFI-BASED INDOOR POSITIONING

by

CHARIS SAVVA
(MTP218)

B.Sc., Hellenic Mediterranean University, 2019

A THESIS

submitted in partial fulfillment of the requirements for the degree.

MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SCHOOL OF ENGINEERING

HELLENIC MEDITERRANEAN UNIVERSITY

2023

Approved by:

Major Professor
Spyros Panagiotakis

ΣΥΣΤΗΜΑ ΕΣΩΤΕΡΙΚΟΥ ΕΝΤΟΠΙΣΜΟΥ ΜΕ ΒΑΣΗ ΤΟ WI-FI

από

ΧΑΡΗΣ ΣΑΒΒΑ
(MTP218)

Πτυχίο, Ελληνικό Μεσογειακό Πανεπιστήμιο, 2019

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υποβάλλεται σε μερική εκπλήρωση των απαιτήσεων για την απόκτηση του πτυχίου

ΜΕΤΑΠΤΥΧΙΑΚΟ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΕΛΛΗΝΙΚΟ ΜΕΣΟΓΕΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

2023

Εγκρίθηκε από:

Επιβλέπων καθηγητής
Δρ. Σπύρος Παναγιωτάκης

Acknowledgments

In the journey of crafting this master's thesis, I am profoundly indebted to my primary supervisor, Associate Prof. Spyro Panagiotaki. His sagacious guidance and unwavering support have been invaluable. I must also extend my appreciation to my esteemed colleagues, whose insightful recommendations enriched my research. Lastly, a heartfelt tribute to my family, whose constant encouragement and unwavering presence provided an anchor throughout my academic sojourn.

Abstract

In this dissertation, the primary focus is the development of an indoor positioning system predicated on WiFi technology, harnessing both Round Trip Time (RTT) and Received Signal Strength Indication (RSSI) methodologies. RTT serves as a mechanism to quantify the duration required for WiFi signals to traverse between a specific device and its corresponding access point, thus facilitating the extrapolation of the distance between any Wireless Local Area Network (WLAN) device and said access point. Conversely, RSSI operates on the premise of ascertaining the triangulated coordinates of a designated target, achieved by a meticulous analysis of signal amplitude at diverse receiver locales. Within the framework of this research endeavor, an ESP32 microcontroller has been employed as the pivotal access point, purposed to approximate the spatial positioning of any WiFi-integrated device within a delineated architectural space or chamber.

We use the ping method to collect RTT and RSSI values of different APs in this project. We gather different RTT values from different esp32 microcontroller access points placed in the building and use them to locate the position for our device. Similarly, we gather RSSI values from different locations' access points, and compare them to get more accurate and closer to real values results. Then we compare the final values of both RSSI and RTT to the original positioning value to evaluate which method is more accurate for indoor positioning.

This thesis can be useful for object tracing or object detection projects using WLAN network. For example, we can use it to locate our laptop or smartphone inside our house or office, or to detect any unauthorized devices in a restricted area.

The main findings of this thesis are that RTT method is more accurate than RSSI method for indoor positioning, and that combining both methods can improve accuracy even further. We also discuss the limitations and challenges of this project, such as the interference of other Wi-Fi signals and the calibration of different devices.

Σύνοψη

Σε αυτή τη μεταπτυχιακή διατριβή, η κύρια επικέντρωση είναι η ανάπτυξη ενός συστήματος εσωτερικού εντοπισμού βασισμένου στην τεχνολογία WiFi, αξιοποιώντας και τις μεθοδολογίες Round Trip Time (RTT) και Received Signal Strength Indication (RSSI). Το RTT λειτουργεί ως μηχανισμός για τον προσδιορισμό της διάρκειας που απαιτείται για τα σήματα WiFi να διασχίσουν μεταξύ ενός συγκεκριμένου συσκευής και του αντίστοιχου σημείου πρόσβασης, διευκολύνοντας έτσι την εξαγωγή της απόστασης μεταξύ οποιασδήποτε συσκευής του Δικτύου Τοπικής Περιοχής (WLAN) και του εν λόγω σημείου πρόσβασης.

Αντιθέτως, το RSSI λειτουργεί στη βάση του προσδιορισμού των τριγωνισμένων συντεταγμένων ενός καθορισμένου στόχου, που επιτυγχάνεται με μια λεπτομερή ανάλυση της πλάτους του σήματος σε διάφορα σημεία λήψης. Στο πλαίσιο αυτής της ερευνητικής προσπάθειας, ένας μικροελεγκτής ESP32 έχει χρησιμοποιηθεί ως το κύριο σημείο πρόσβασης, με σκοπό την προσέγγιση της χωρικής θέσης οποιασδήποτε συσκευής με ενσωματωμένο WiFi εντός ενός καθορισμένου αρχιτεκτονικού χώρου ή δωματίου.

Χρησιμοποιούμε τη μέθοδο ping για τη συλλογή τιμών RTT και RSSI διαφορετικών σημείων πρόσβασης σε αυτό το έργο. Συλλέγουμε διαφορετικές τιμές RTT από διαφορετικά σημεία πρόσβασης μικροελεγκτή esp32 τοποθετημένα στο κτίριο και τα χρησιμοποιούμε για τον προσδιορισμό της θέσης της συσκευής μας. Ομοίως, συλλέγουμε τιμές RSSI από διαφορετικά σημεία πρόσβασης και τα συγκρίνουμε για να πάρουμε πιο ακριβείς και κοντά στα πραγματικά αποτελέσματα. Στη συνέχεια, συγκρίνουμε τις τελικές τιμές των RSSI και RTT με την αρχική τιμή εντοπισμού για να αξιολογήσουμε ποια μέθοδος είναι πιο ακριβής για τον εσωτερικό εντοπισμό.

Αυτή η διατριβή μπορεί να είναι χρήσιμη για έργα εντοπισμού ή ανίχνευσης αντικειμένων χρησιμοποιώντας δίκτυο WLAN. Για παράδειγμα, μπορούμε να τη χρησιμοποιήσουμε για τον εντοπισμό του φορητού ή του κινητού τηλεφώνου μας μέσα στο σπίτι ή στο γραφείο μας, ή για την ανίχνευση οποιασδήποτε μη εξουσιοδοτημένης συσκευής σε μια περιοχή με περιορισμούς.

Τα κύρια ευρήματα αυτής της διατριβής είναι ότι η μέθοδος RTT είναι πιο ακριβής από τη μέθοδο RSSI για τον εσωτερικό εντοπισμό και ότι ο συνδυασμός και των δύο μεθόδων μπορεί να βελτιώσει ακόμη περισσότερο την ακρίβεια. Συζητούμε επίσης τους περιορισμούς και τις προκλήσεις αυτού του έργου, όπως η παρεμβολή άλλων σημάτων Wi-Fi και η βαθμονόμηση διαφορετικών συσκευών.

Table of Contents

Table of Figures

List of Tables

Chapter 1 - Introduction

In today's interconnected world, our reliance on accurate positioning systems, such as the Global Positioning System (GPS), is undeniable. However, GPS's penetration weakens when we venture indoors, leading to a gap in precise location tracking. This limitation manifests itself in everyday scenarios: a shopper wandering in a large mall searching for a specific store, a healthcare worker trying to locate critical equipment in a vast hospital, or a museum enthusiast looking for an exhibit. Given these myriad applications, the need for an effective indoor positioning system (IPS) becomes paramount.

The contemporary environment presents a ubiquitously installed infrastructure: WiFi. Present in nearly every commercial building, retail establishment, and many homes, WiFi networks offer a tantalizing proposition: can these omnipresent networks be harnessed for accurate indoor positioning?

Cost considerations further bolster the argument for WiFi. Setting up new infrastructure for indoor positioning can be prohibitively expensive and logistically challenging. Leveraging existing WiFi infrastructure could circumvent these obstacles, providing a cost-effective solution.

Historically, various techniques have been proposed for indoor positioning, from ultrasonic signals to infrared. Yet, these solutions often required specialized hardware and were not universally adaptable. With the proliferation of WiFi-enabled devices, from smartphones to tablets and laptops, the transition towards exploring WiFi as an IPS solution was a logical evolution.

This thesis aims to delve deep into the intricacies of WiFi-based indoor positioning. It will explore the methodologies behind using WiFi signals for positioning, identify the inherent challenges of this approach, and illuminate the potential future trajectories of this technology. In doing so, we hope to provide a comprehensive overview of the current state of WiFi indoor positioning and chart a course for its future advancements.

In the subsequent chapters, readers will be acquainted with the foundational concepts underpinning this technology, a comparative analysis with other indoor positioning contenders, detailed exploration of algorithms and methodologies, real-world applications, challenges, and potential future trends.

Wireless Fidelity (Wi-Fi) is conventionally recognized for its capabilities in facilitating internet connectivity and communication between devices. Yet, its utility extends beyond mere networking, with positioning emerging as a significant application. Wi-Fi-based positioning systems (WPS) harness Wi-Fi signals to ascertain the geographical coordinates of a specific device or entity. Such systems have been instrumental in advancing indoor navigation, optimizing asset tracking, enhancing location-centric services, and bolstering security protocols. Notwithstanding its merits, WPS encounters a series of challenges. These encompass signal interference, the phenomenon of multipath propagation, and the dynamic nature of environmental conditions. As such, a comprehensive comprehension of Wi-Fi positioning's underlying mechanisms is imperative, paired with concerted efforts to ameliorate its precision and dependability.

The aim of this thesis is to investigate the performance of Wi-Fi positioning in different scenarios and environments, and to propose some methods to enhance its accuracy. The main research question of this thesis is: How can Wi-Fi positioning be optimized for various use cases and conditions? To answer this question, this thesis will conduct the following tasks:

- Review the existing literature on Wi-Fi positioning and its techniques, challenges, and applications.
- Design and implement a Wi-Fi positioning system using ESP32-S2 devices as access points (APs) and smartphones as clients.
- Undertake empirical investigations to gauge the precision of the Wi-Fi positioning system across a spectrum of both indoor and outdoor settings, encompassing academic spaces, libraries, recreational parks, and urban thoroughfares. Following the data collection, it is essential to execute a comprehensive analysis of the experimental findings. This should elucidate determinants that influence the accuracy of Wi-Fi-based location estimation, such as the spatial distribution and quantity of Access Points (APs), signal amplitude and fidelity, as well as the introduction of barriers and potential sources of interference.
- Advocate for a series of innovative solutions aimed at augmenting the precision of Wi-Fi-based positioning systems. Potential strategies could encompass the deployment of machine learning algorithms, designed to adaptively refine positioning accuracy based on

real-time data. Additionally, the integration of advanced filtering techniques, which can mitigate noise and aberrations in signal data, may prove beneficial. Furthermore, consideration should be given to the formulation of hybrid methodologies, amalgamating multiple techniques to achieve superior location estimation fidelity.

1.1 Benefits

Unlike other indoor positioning solutions such as RFID, BLE beacons, and other solutions for indoor positioning, Wi-Fi doesn't require any extra infrastructure hardware or maintenance. Almost all Wi-Fi indoor positioning is giving base capabilities without requiring additional investment. Many hardware solutions require manual deployment, which is time-consuming and expensive. That's why other techniques require time and money to deliver environment conductive to Wi-Fi indoor positioning [33].

Wi-Fi is an all-encompassing indoor positioning technology that can be scaled up without any manual intervention. For example, installing Wi-Fi positioning within 25 warehouses instead of manually setting up RFID beacons or BLE beacons in 25 warehouses will save you time and energy. Furthermore, since Skyhook keeps an international map that includes more than five billion access points to Wi-Fi, your positioning system will be in operation all over the world, inside and outside [35].

Although indoor Wi-Fi positioning implies instant-on capability, businesses have a variety of ways to improve accuracy depending on the specific requirements of a use case. The precision of positioning offered by the system can be increased, for instance, by conducting a survey to determine the precise locations of all Wi-Fi APs in the building. If the system requires more accuracy, the business can install further Wi-Fi access points on their premises. Although this is an additional cost, it doesn't need the customer to master how to deploy, manage, and maintain an additional technology within their infrastructure. This can be accomplished through the integration of a wireless network enhancement project with the addition of more Wi-Fi access nodes in a precise range [33].

Adopting more modern technology will increase the precision of Wi-Fi as an indoor positioning system, which is an additional benefit. Round-trip timing for Wi-Fi (RTT) is an element of the IEEE 802.11 protocol and the finest illustration for improving the precision of indoor Wi-Fi positioning. By utilizing Wi-Fi RTT equipment, endpoints may estimate the distance to the AP using time-of-flight (ToF) instead of signal intensity monitoring [34]. This is how Wi-Fi RTT can be used to define the location with one to three meters. This approach is widely adopted today and supported by Android 9, evaluating, and showing achievable accuracy. Wi-Fi RTT was not fully functional in 2020, but current investment in technology can define the rise in the value in industries.

1.2 The proposed method of implementing indoor Wi-Fi.

Wi-Fi indoor positioning makes use of signal propagation models, which determine the range across a device and an access point by measuring the intensity of the signal it receives from the AP [32]. However, this technique suffers from multipath fading, which occurs when the signal reflects or scatters on various surfaces and paths [35]. Another technique is location fingerprinting, which compares the RSS measurements from multiple APs with a pre-recorded database of RSS values at different locations [33]. This technique can achieve higher accuracy than signal propagation models, but it requires a large amount of training data and frequent updates. In addition to RSS, other parameters can be used for Wi-Fi indoor positioning, such as Time Difference of Arrival (TDOA), Angle of Arrival (AOA), and Time of Flight (TOF). TDOA measures the difference in the arrival time of signals from two or more APs to a device [36]. AOA measures the angle of signals from APs to a device or vice versa [37]. TOF measures the round-trip time of signals between a sender and a receiver [38]. These parameters can provide more information about the location of a device, but they also require precise processing time and synchronization.

A new cloudlet-based Wi-Fi indoor locating system is proposed in this thesis. To deliver high-bandwidth and low-latency services to local mobile devices, "Cloudlets" are small-scale cloud servers placed at the edge of the internet [29]. Cloudlet can improve the efficiency, speed, and accuracy of Wi-Fi indoor location. Wi-Fi indoor location relies mostly on two methods: fingerprinting and signal propagation. When using fingerprinting, data from numerous access points (APs) are compared to a database of previously recorded RSS values in various places [33]. This method, however, calls for a substantial quantity of training data and regular updates. Based on the RSS, a device's distance from an AP may be calculated during signal propagation [32]. However, this technique suffers from multipath fading, which occurs when the signal reflects or scatters on various surfaces and paths [35]. Therefore, this thesis proposes to use cloudlet as a platform to process and analyze the Wi-Fi signals and provide accurate location information to the mobile devices. Cloudlet has several advantages over traditional cloud servers, such as:

- Cloudlet can deliver powerful computing resources and speed up the execution of mobile applications through Wi-Fi, which is located on the top of the wireless network [34].
- Cloudlet can provide real-time response by accessing one high hop bandwidth to minimize transmission delay [34].
- Cloudlet can reduce cellular energy consumption problems and wide area network latency by connecting mobile devices to cloud servers [29].
- In cloud computing systems, Cloudlet can enhance the user experience by delegating computation duties to nearby cloud servers [30].

This thesis presents a cloud-based computing system that utilizes cloudlets to enable tracking and navigation through Wi Fi. Cloudlets are small scale cloud servers strategically placed within the network range of devices [29]. They can be installed in locations, like theaters, retail centers, conference rooms and office buildings to enhance mobile device connectivity [34]. The system offers a range of location-based services such as navigation for humans and robots asset monitoring, assistance for impaired individuals, factory automation, workplace safety measures and location targeted advertising.

The proposed system is composed of an array of autonomous vehicles, a compact data center interfaced with the principal cloud infrastructure, and wireless access nodes. The Received Signal Strength Indicator (RSSI) methodology is employed to deduce the spatial separation between the interior carriage and designated reference access points. Through identification of proximity to the most adjacent reference node, the system can extrapolate the precise location of the interior cart. Additionally, the overarching cloud infrastructure plays a pivotal role, furnishing essential data pertaining to the device network topology as well as delineating the trajectory of the indoor cart.

Cloud networks provide wireless location access points and destination maps, as well as the power to transport the vehicle to the desired location. The movement's path, direction, and position are determined by the cloud's undulating edge. The results of the implementation proved the viability of the indoor Wi-Fi concept. Additionally, the technique can be used to locate a patient in a hospital or assist impaired patients. It may be improved in the future by reducing carriage movement and internal processing periods.

We typically lay the foundation for the creation of a transportable Indoor Positioning System that is totally independent of the Global Positioning System. An entity with wireless capabilities will be located using triangulation in relation to other beacons. Different types of Access points will use fingerprinting to pinpoint where they are. Based on the signal strength received, the RSSI (Received Signal Strength Indicator) will help us determine the precise locations of beacons or access points. This is because to maintain its position, every positioning system needs a triangulation system. To locate an object on a fixed plane using this method of triangulation, at least three beacons or access points must be used.

The number of beacons increases the overall method's coverage and accuracy. Distance estimation based on the signal intensity of the scenario's wireless technology also forms the premise of the fundamental concept. This is precisely how the Global Positioning System functions, although it depends on satellites and requires a clear sky or both for excellent accuracy and real-time tracking when localizing or tracking an object. In addition, because it is primarily designed for monitoring locations, such as locations on a map or a traveling route, it cannot detect subtle changes within a structure, such as the floor or room you are on. To localize or track an object in real-time while it is indoors, we have developed our own indoor triangulation using multiple ESP32 beacons as opposed to relying on satellites. An Android application that collects localization data from an internet server displays the tracked object. The localization of an object can be affected by several variables, but this method of tracking an object using numerous tiny Wi-Fi or WIFI beacons is effective. As opposed to satellites, whose transmissions are unaffected by tiny objects in their route, our microcontroller based WIFI Low Energy system will have a much higher frequency but shorter wavelength.

If an intervening object is situated between the object being tracked and the beacon, the anticipated distance between them may experience deviations. This parameter can be further modulated by factors such as transmission power, the variety of the antenna, its specific dimensions, and its orientation. While this constitutes a potential limitation of the method, the adverse effects can be attenuated by opting for industrial-grade beacons as opposed to more economical microcontrollers.

Chapter 2 - Related Technologies

In today's interconnected world, our reliance on accurate positioning systems, such as the Global Positioning System (GPS), is undeniable. However, GPS's penetration weakens when we venture indoors, leading to a gap in precise location tracking. This limitation manifests itself in everyday scenarios: a shopper wandering in a large mall searching for a specific store, a healthcare worker trying to locate critical equipment in a vast hospital, or a museum enthusiast looking for an exhibit. Given these myriad applications, the need for an effective indoor positioning system (IPS) becomes paramount.

The contemporary environment presents a ubiquitously installed infrastructure: WiFi. Present in nearly every commercial building, retail establishment, and many homes, WiFi networks offer a tantalizing proposition: can these omnipresent networks be harnessed for accurate indoor positioning?

Cost considerations further bolster the argument for WiFi. Setting up new infrastructure for indoor positioning can be prohibitively expensive and logistically challenging. Leveraging existing WiFi infrastructure could circumvent these obstacles, providing a cost-effective solution.

Historically, various techniques have been proposed for indoor positioning, from ultrasonic signals to infrared. Yet, these solutions often required specialized hardware and were not universally adaptable. With the proliferation of WiFi-enabled devices, from smartphones to tablets and laptops, the transition towards exploring WiFi as an IPS solution was a logical evolution.

This thesis aims to delve deep into the intricacies of WiFi-based indoor positioning. It will explore the methodologies behind using WiFi signals for positioning, identify the inherent challenges of this approach, and illuminate the potential future trajectories of this technology. In doing so, we hope to provide a comprehensive overview of the current state of WiFi indoor positioning and chart a course for its future advancements.

In the subsequent chapters, readers will be acquainted with the foundational concepts underpinning this technology, a comparative analysis with other indoor positioning contenders, detailed exploration of algorithms and methodologies, real-world applications, challenges, and potential future trends.

2.1 WiFi-based Positioning

2.1.1 Background of WiFi as a communication protocol.

Wireless Fidelity (Wi-Fi), rooted in the IEEE 802.11 suite of standards set forth by the Institute of Electrical and Electronics Engineers, has been adopted as a pivotal wireless networking methodology since its inception in the late 20th century. It facilitates digital communication between devices and networks without the need for physical wired connections. Initially designed to wirelessly connect devices to LANs, its widespread adoption and infrastructure have made it a useful tool for positioning.

2.1.2 Mechanism of Wi-Fi-based indoor positioning.

WiFi-based indoor positioning systems (WIPS) generally work through either triangulation or trilateration. When a device is searching for nearby networks, it captures the signals from various Access Points (APs). The strength and time of these signals' arrival help in determining the device's relative position. The more APs available, the more accurate the location estimate.

2.1.3 Strengths and weaknesses in positioning applications.

Strengths:

- **Ubiquity:** WiFi is widespread, reducing infrastructure costs for positioning.
- **Dual utility:** Beyond positioning, WiFi also serves network connectivity.

Weaknesses:

- **Accuracy:** Typically, WiFi can offer accuracy up to 2-5 meters, which might not be precise enough for all applications.

- **Signal interference:** Physical objects, other electronics, and even human bodies can distort WiFi signals, affecting accuracy.

2.2 Round Trip Time (RTT)

Round-Trip Time (RTT), alternately referred to as Round-Trip Delay Time (RTD), delineates the temporal measure required for a data packet to traverse from one terminus of a network to the other and back to its point of origin.

The propagation times for the communication channels connecting the two ends are included in this delay.

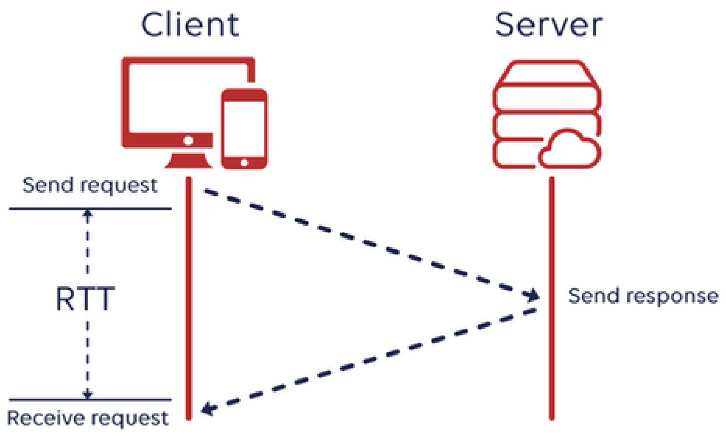


Figure 1 - Round Trip Time - RTT

2.2.1 Basics of RTT and its significance.

Wi-Fi Fine Time Measurement (FTM) is a method for calculating the round-trip time (RTT) of wireless signals between a client device and an access point (AP). Accurate distance measurements are provided by this technology, which can be applied to location-based services and indoor positioning. The IEEE 802.11mc protocol, which enables the sending and receiving of time-stamped signals between devices, is the foundation of FTM. FTM can calculate the separation between the client and AP by measuring the time it takes for signals to travel between them.

FTM provides an accurate and reliable method of measuring distances for indoor positioning, which can benefit a range of industries from retail to healthcare. For example, retailers can use FTM to offer personalized offers to customers as they walk past certain products, while hospitals can use FTM for tracking medical devices and personnel. Overall, FTM offers a promising solution for improving indoor positioning accuracy and innovative applications in many industries.

The time it takes for a connection to go from sender to receiver is known as the round-trip time (RTT). The time it takes a payload to go from one device to another and back again may be used to estimate this measure.

The formula for calculating RTT is as follows: RTT = (time of receipt at destination - time of sending at source) + (time of acknowledgement at destination - time of receipt at source).

For example, if a device sends a packet at 10:00:00.000 and receives an acknowledgement from the destination at 10:00:00.500, and then receives the response at 10:00:01.000, the RTT would be calculated as follows: RTT = (10:00:00.500 - 10:00:00.000) + (10:00:01.000 - 10:00:00.500) = 1 second

Therefore, the round-trip time in this example is 1 second.

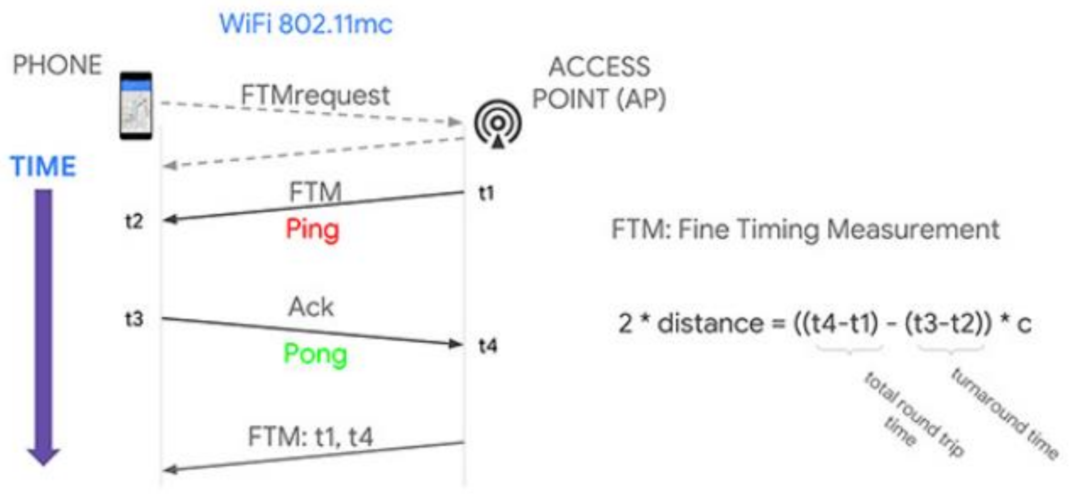


Figure 2 - RTT calculation

Notably, the access point's departure and arrival times, t1 and t4, are contained in the second FTM "Ping" it sends out. Round-trip times are not perfectly accurate because of various measurement error sources, RF interference, and the positions and movements of nearby objects. A slight quality improvement may be seen after several measurements:

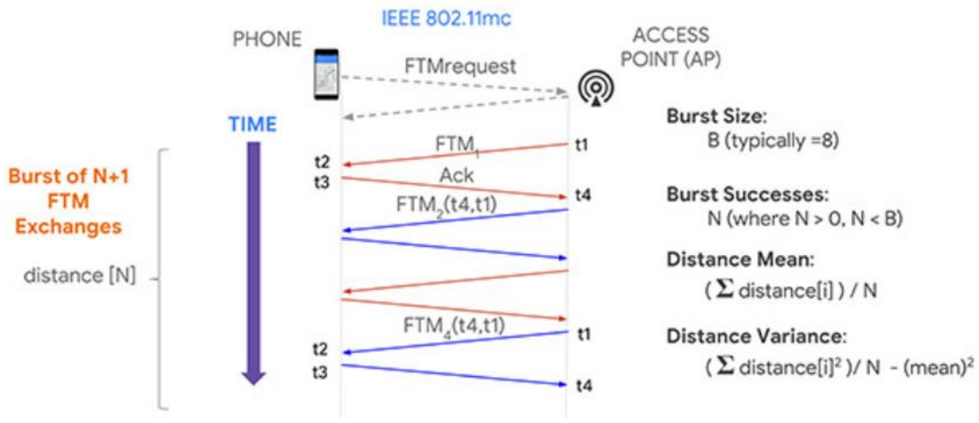


Figure 3 - Indoor Positioning System

The round-trip time for the final interchange is unknown because there is no signal carrying t4 and t1 after the final interchange ("Ping" and "Pong"). As a result, only (N-1) round trip timings for a string of N interchanges can be calculated.

2.2.2 Application of RTT in indoor positioning: Time-of-Flight (ToF) methods.

Time-of-Flight (ToF) methodologies determine distances by leveraging Round-Trip Time (RTT) in conjunction with the propagation speed of radio waves. For the purpose of ascertaining the spatial separation between a device and an Access Point (AP), instrumentation designed for indoor positioning meticulously gauges the duration required for a signal to traverse to an AP and subsequently revert to its origin.

2.2.3 Advantages of RTT-based methods over others.

- **High accuracy:** With proper calibration, RTT can achieve sub-meter accuracy.
- **Less influence from signal strength fluctuations:** Unlike RSSI-based methods, which can be influenced by signal strength variations, RTT focuses on time, often leading to more consistent readings.

2.2.4 Limitations and challenges of using RTT.

- **Requires specialized hardware:** Not all devices or APs support RTT measurements.
- **Signal reflection:** Signals can bounce off walls or objects, leading to inaccurate readings.

2.3 Received Signal Strength Indicator (RSSI)

2.3.1 Understanding the RSSI and its role in wireless communications.

RSSI is a metric that measures the power level a device is receiving from the wireless network. It is used in positioning to estimate distance based on signal strength, with a weaker signal indicating a greater distance.

One of the parameters that is used for RSSI is the obtained signal intensity indicator (Wi-Fi internal positioning). RSSI is a measure of the received wireless signal's intensity, typically expressed in decibels per milliwatt (dBm) [25]. The RSSI value reflects the signal quality and strength of a wireless connection. Generally, a higher RSSI value means a stronger and more reliable signal, while a lower RSSI value means a weaker signal that may be affected by interference or noise. RSSI is used in various wireless communication systems, such as WiFi, Bluetooth, and cellular networks, to optimize the signal strength, range, and data throughput. Data throughput is the amount of data that can be transferred over a wireless connection in each time [26]. A higher data throughput indicates a faster and more efficient wireless connection.

RSSI is a useful metric for measuring wireless signal strength in real-world environments. It is important to note that RSSI is not a direct measurement of signal quality, as factors like noise and interference can affect the quality of the signal even if the RSSI value is high. However, RSSI can be used in conjunction with other metrics like signal-to-noise ratio (SNR) and error rate to assess signal quality more accurately. In addition to optimizing signal strength and range, RSSI can also be used to implement location tracking in indoor and outdoor environments through techniques like triangulation. Overall, RSSI is an important tool for wireless communication systems to maintain reliable and high-quality connections.

Among the most popular theoretically based RSSI ranging approaches are the path loss model of free-space propagation [26] and the block model of logarithmic normal [26] (Shadowing model). It is well knowledge that the energy required to transmit a signal decrease with distance, and that this model is unaffected by obstructions and dispersed reflection. The energy required to send a signal and the distance at which it must travel both follow a linear relationship. The path loss model is as described in [26]:

$$\text{Loss} = 32.44 + 10n\log(d) + 10n\log(f).$$

In this above equation, the terms loss and d, which stand for the wireless signal's frequency in megahertz, f, and meters, respectively, and n, which stands for the path attenuation factor in the actual environment, represent the signal energy lost along the path. Wireless sensor signal applications must account for shading and absorption brought on by obstructions as well as interference from scattered reflection in real-world environments like indoor buildings or industrial sites. Long-distance channels attenuate according to the lognormal distribution, which is frequently used in conjunction with the logarithmic normal block model; the path loss model is as follows:

$$P_L(d) = P_L(d_0) + 10n\log\left(\frac{d}{d_0}\right) + X_\sigma.$$

In the formula, the path loss exponent specific to a given environment is denoted by n. Concurrently, the rate of path loss, which intensifies with escalating distance, is represented. When the measurement distance is quantified as d (in meters), PL(d) signifies the path loss of the received signal. Conversely, PL(d0) designates the path loss of the received signal when the reference distance is established as d0. In scenarios where the shadowing factor is symbolized by X (expressed in dB) and the standard deviation spans a range of 4-10, the intensity of the signal received by nodes can be formulated as follows:

$$\text{RSSI} = P_t - P_L(d).$$

When the distance is d, the signal transmission power and the path loss are both expressed in dBm and are known as Pt and PL(d), respectively.

The letter A represents the signal strength from the reference nodes at d0 and is as follows:

$$A = P_t - P_L\,(d)\,.$$

The path loss model, which is quantified at the actual distance d (m), is as follows:

$$P\,(d) = P\,(d_0) - 10n\log\left(\frac{d}{d_0}\right) - X_\sigma.$$

P(d) reflects the received signal intensity when the actual distance being measured is d(m). When the reference distance is d0, XN(0,2), P(d0) displays the received signal intensity.

2.3.2 Utilizing RSSI for indoor positioning: Proximity and Triangulation methods.

- **Proximity:** The device identifies its position based on the strongest signal from an AP, suggesting its closest to that particular AP.
- **Triangulation:** Using signal strengths from multiple APs to determine a more accurate position.

2.3.3 Factors affecting RSSI measurements.

Physical obstructions, atmospheric conditions, interference from other devices, and even changes in device orientation can influence RSSI values.

2.3.4 The pros and cons of using RSSI for indoor positioning.

Pros:

- **Simplicity:** RSSI is easy to implement and doesn't require specialized hardware.
- **Cost-effective:** Utilizes existing WiFi infrastructure.

Cons:

- **Lower accuracy:** Typically offers accuracy in the range of 5-15 meters.

- **Sensitive to environmental changes:** Walls, furniture, and even people moving around can affect readings.

2.4 Bluetooth Low Energy (BLE) and Beacons

2.4.1 Introduction to BLE and its evolution.

BLE, a power-efficient version of Bluetooth, was introduced with Bluetooth 4.0. Designed for short bursts of data, it's ideal for applications that don't need continuous communication.

Bluetooth beacons are tiny wireless devices that transmit signals using Bluetooth low Energy (BLE) technology, and this paragraph explains how they work and what capabilities they have. BLE is a standard for wireless communication that permits low-power, close-range interaction between electronic devices [38]. A Bluetooth beacon is a device that uses BLE to transmit an audio signal that consists of a combination of numbers and letters, in short and frequent intervals. The audio signal acts as an identifier for the beacon, which can be detected by other BLE-enabled devices, such as smartphones, gateways, or access points.

The analogy of a lighthouse can be used to explain how Bluetooth beacons work. A lighthouse emits visible light that can be seen by sailors who are looking for their location. Similarly, a Bluetooth beacon emits an audio signal that can be seen by devices that are looking for its presence or proximity.

Smartphones can determine an approximate context or location thanks to Bluetooth beacons' indoor positions. Smartphones can locate themselves in relation to Bluetooth beacons in stores with the aid of Bluetooth beacons. Beacons are used in brick-and-mortar stores for mobile commerce, enabling mobile payments through points of sale systems and providing customers with special offers through mobile marketing [45].

Based on their location, Bluetooth beacons can be distinguished from other technologies. Beacons are transmitted by a one-way transmitter to a smartphone or other receiving device. With this method, a specific app must be installed on the device to communicate with the beacons. This indicates that only the app that has been installed and the Bluetooth beacon transmitter can monitor users. There are many different Bluetooth beacon transmitters on the market, including tiny coin-cell gadgets, USB sticks, and the more popular Bluetooth 4.0 equipped USB dongles. [46]

A variety of algorithms have been proposed for this. However, the most efficient method for indoor positioning uses Bluetooth beacons, which are combined with accelerometers and geomagnetic sensors [38]. When it comes to the case of indoor navigation for pedestrians, pedestrians typically walk to their destination and are constantly checking their current location. Therefore, it is vital to ensure a steady location. The proximity of Bluetooth beacons to determine the most reliable radio field intensity measurement techniques. But, as it only displays the distance that is very rough to the beacons of interest using it, it is a huge issue [46].

Bluetooth Beacons form the heart of indoor technology for location. It is among the most well-known technologies that have emerged to aid indoor positioning and is now a standard feature on most smartphones in the present. It utilizes so-called BLE beacons (or iBeacons), which are cheap and small [45]. They can function independently of the power grid and have a very extended battery life. By picking up the beacon's signal and calculating the distance to it, the device can roughly estimate the user's position inside the building. [46]

2.4.2 BLE Beacons and their role in indoor positioning.

Beacons are small devices that emit BLE signals at regular intervals. By placing them at known positions, a device can determine its location based on the signals it receives from these beacons.

2.4.3 Strengths of BLE Beacon-based positioning.

- **Energy Efficient:** Beacons have long battery lives due to BLE's efficiency.
- **High Accuracy:** Achieves sub-meter accuracy in ideal conditions.
- **Scalability:** Easy to deploy additional beacons for better coverage.

2.4.4 Limitations and real-world challenges.

- **Infrastructure requirement:** Requires deployment of beacons throughout the facility.
- **Interference**: Other electronic devices can interfere with BLE signals.

2.5 Wi-Fi Access Point

To pinpoint the connected device or item, the Wi-Fi access point employs several different positioning techniques. With the help of Wi-Fi access points (APs) and the existing network architecture, a device's position may be calculated using Wi-Fi location. The device must be able to detect the Wi-Fi AP, however joining is optional. Skyhook can locate the device by analyzing the signal strength of many Wi-Fi connections and knowing the specific locations of these access points. [37]

Although its range is limited, Wi-Fi can still cover up to 150 meters. The quality of the signal is greatly affected by factors such as the proximity of other APs and their placement. The more APs there are in each area, the more precisely their location will be determined. When using crowdsourced, uncalibrated Wi-Fi networks, Wi-Fi can deliver accuracy of about 20 m [41]. In indoor environments, Wi-Fi can achieve accuracy of up to 5-8 meters through calibration, surveying, and fine-tuning.

Wi-Fi-based indoor positioning systems (WIPS) are increasingly becoming an instrumental mechanism for indoor spatial recognition, serving a diverse array of applications. Several salient factors contribute to this burgeoning adoption. Primarily, a significant proportion of contemporary smartphones are inherently equipped with Wi-Fi modules. Secondly, the ubiquity of Wi-Fi-based localization infrastructures, combined with the pervasive distribution of Wi-Fi access points, fortifies this trend [37]. Moreover, Wi-Fi obviates the necessity for bespoke equipment; the analysis of Received Signal Strength (RSS) values emanating from access points simplifies the process of location triangulation. To cater to the exigencies of elevated data transmission velocities, the bandwidth capacities of Wi-Fi systems have witnessed considerable amplifications [37].

The cloudlet is the most popular method of positioning indoor Wi-Fi. The technology used to offer cloud-based mobile services, known as Cloudlet, is at the very edge of the Internet. [41] Cloudlets provide a resource- and latency-sensitive application that brings cloud computing closer to the user. They are often accessed over wireless networks. To speed up the delivery of cloud computing applications to nearby mobile devices like tablets, smartphones, and wearable technology, small-scale data centers dubbed "cloudlets" have been developed. [41]

2.6 MEMS

Microelectromechanical systems (MEMS) based on inertial measurement units (IMU) are frequently inexpensive localization sensors. By logging angular rates and linear accelerations, they can continuously deliver extremely high-frequency location, speed, and attitude results [52]. While they are independent of the environment around them, the error-to-error ratio is inevitable and will become larger as time passes.

To limit the accumulation of errors in an indoor environment, MEMS-based IMU sensors typically use the use of light detection (Lidar) or visual sensors to provide simultaneous localization as well as mapping (SLAM) [48]. Although this strategy of integrating sensors aids in reducing the total amount of attitude and position errors, the cost associated with the devices is too high for most LBS users, and the environment must be extremely textured to ensure robust localization (Debeunne and Vivet,).

The indoor positioning system of MEMS sensors comprises three Axis accelerometers, a three-axis magnetometer, and three-axis Gyroscope. This system can help achieve precise positioning in an indoor environment. It can also be utilized for outdoor use due to its unique characteristics. It's a wearable foot device that uses a wireless network to send data about the movement to computers that determine the foot's position and show the route that you have taken [48]. The principle behind the system for positioning is a dead reckoning and inertial navigation technology. The time taken to get there affects the displacement because the location is determined by doing a double integration of the acceleration. There are two primary tracks for MEMS development. The first uses principles from inertial navigation [52].

This is accomplished by mounting the inertial sensors on the feet, determining the foot's position, translating its acceleration into earth coordinates, and then double integrating the acceleration to determine the foot's position. Its Double Integration Method is the name given to this method (DIM). Another method is to keep track of how many steps you take and use that information to calculate your distance. Even if they are held in the hands, sensors can be worn at the waist or on the foot. To determine the location, it estimates the length of the step-through acceleration and calculates steps. This approach is known as the Pedometer Method (PM). [52]

2.7 GPS Indoor Positioning

The Global Positioning System (GPS), when deployed in open terrains, boasts commendable accuracy and reliability. However, its utility indoors is hampered by the intrinsic challenges associated with decoding and discerning GPS signals. This degradation can be largely attributed to the signal attenuation caused by architectural barriers, such as walls and edifices [50]. To circumvent this impediment associated with indoor coverage, a sophisticated positioning system has been devised, underpinned by GPS repeaters and an augmented positioning algorithm. The prototype, representative of 1D/2D indoor positioning capabilities, incorporates directional GPS antennae and low-noise amplifiers (LNA). Real-time GPS data is meticulously processed through the refined algorithm. An amalgamation of these system components provides a holistic positioning solution, the efficacy of which is subject to rigorous evaluation [50].

2.8 A-GPS Indoor Positioning

A-GPS indoor positioning is a term for a location-based technology that uses A-GPS to pinpoint a user's or a device's precise location inside an enclosed area. A-GPS indoor positioning overcomes the limitations of traditional GPS technology, which is frequently inaccurate indoors due to weak signal strength and obstructions like walls and ceilings. The device's position is refined with the use of supplementary data, such as Wi-Fi hotspots and Bluetooth beacons. [43]

A-GPS indoor positioning works by using a database of reference points within the indoor space and comparing the device's signal strength to each reference point to triangulate its location.[43] This information can be used to provide location-based services for users within a building, such as turn-by-turn directions, real-time location tracking, and proximity-based notifications.

A-GPS indoor positioning has a wide range of applications, including indoor navigation, asset tracking, employee tracking in large buildings, healthcare monitoring systems, and retail analytics. By giving precise location data in the case of an emergency, it may also help reduce response times. [43]

2.9 Different algorithms for indoor position technology

2.9.1 Time of Arrival

The length of time needed for a wireless signal to travel from a transmitter to a receiver is calculated using the time of arrival (TOA) approach. ToA combines with other location-based technologies like GPS and A-GPS to determine a device's or user's exact position. [50]

The Time-of-Arrival (ToA) metric quantifies the duration a signal necessitates to traverse between two distinct points. Conventional communication systems comprise a transmitter, such as a cellular tower or Wi-Fi access point, and a receiver, exemplified by mobile devices. By evaluating the signal's transit duration from the transmitter to the receiver, one can infer the interspatial distance. When ToA is synergistically employed with additional parameters like Angle-of-Arrival or Received Signal Strength, it becomes instrumental in ascertaining the precise geospatial coordinates of a device or an end-user [47].

The Time-of-Arrival (ToA) methodology finds applicability across a spectrum of domains, encompassing emergency response mechanisms, asset monitoring, interior navigational systems, and geospatially-anchored services. To furnish pinpoint locational intelligence contemporaneously, ToA is frequently amalgamated with complementary location-determination technologies [50].

2.9.2 Triangulation

Triangulation is a method of localization that makes use of the distance and angles between many places of measurement [42]. This may be done by using the angle-side-angle triangle congruency theorem to determine where an item is. Triangulation is a challenging task. It necessitates understanding not only the precise location of BLE beacons but also the direction in which they rotate.

The calculations aren't as complicated as trilateration. However, they are considerably more sensitive because of how they are determined [42]. While trilateration relies on the strength of signals as an analogy for distance, triangulation relies on the differences in timing when it comes to receiving signals from tags. Since the signals are transmitted at the speed of light, the transmission time variations are very minimal. This means that measuring instruments are more costly.

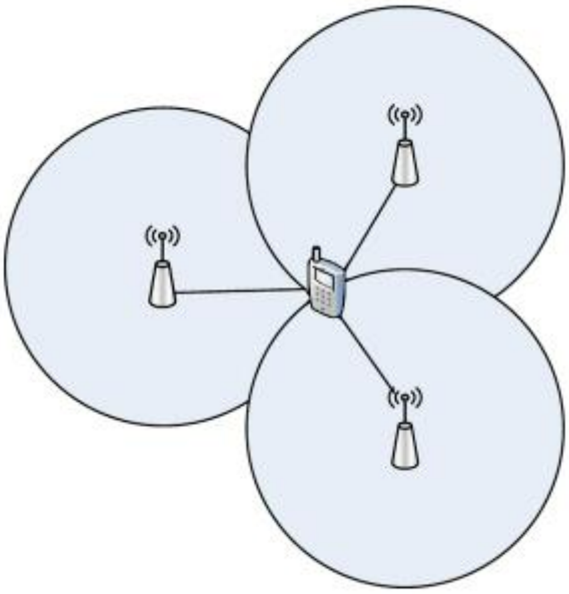


Figure 4 - WiFi Triangulation

2.9.3 Multilateration

Using the time difference of arrival (TDOA) of signals from several sources, multilateralism may be used to pinpoint the precise position of an item like a mobile phone. By using the intersection of numerous circles or spheres, the location of an item may be triangulated using several sensors like GPS satellites or cell towers.

By measuring the time, it takes for signals to travel between the object and each sensor, the system can calculate the distance between the object and each sensor, and then use this data to determine the object's location. Multilateration is widely used in navigation and positioning systems, such as GPS, as well as in mobile phone networks for location-based services. [51]

2.9.4 Fingerprinting

In contrast, indoor fingerprint positioning mostly uses signal fingerprints. It relies on the received signal strength for each place (as fingerprints are stored in the database for fingerprints) to correlate the strength of signals measured at users' location for positioning [40]. The technology of Wi-Fi fingerprint positioning has the benefits of being low-cost and high precision. Because of the broad application and deployment of Wi-Fi across the globe, The fingerprint positioning method is relatively inexpensive since it can be used in any indoor place with Wi-Fi networks without the need for extra gear.

The system uses the strength of Wi-Fi signals to measure and model without pinpointing exactly where to locate the APs. In an indoor setting with a lot of complexity with low-cost conditions, space-time characteristics like the timing of arrival and angle could result in large errors. Still, the signal strength is stable, which makes positioning with this method more accurate than others [40].

2.9.5 Dead Reckoning

Dead reckoning is a method of navigation that involves estimating one's current position based on the distance and direction traveled from a known starting point. It relies on the individual's ability to keep track of their speed and course, as well as any changes in direction due to wind or currents. Dead reckoning can be used in situations where other means of navigation, such as GPS or landmarks, are not available or reliable. However, it is subject to errors and inaccuracies, particularly in situations where external factors like weather and terrain are unpredictable. [44]

Chapter 3 - System design

3.1 System description

Indoor navigation, in its essence, is a symphony of technology and design aimed at guiding users within confined spaces. While GPS revolutionized our understanding of outdoor positioning, indoor spaces, with their unique challenges, require equally innovative solutions. Against this backdrop, WiFi-based indoor positioning systems (IPS) have emerged as a prominent contender, capitalizing on the ubiquity of WiFi infrastructures. The inherent advantage of leveraging pre-existing WiFi networks means that no additional hardware installations are necessary, making such systems both cost-effective and universally adaptable.

Received Signal Strength Indicator (RSSI) and Round-Trip Time (RTT) are two WiFi-based techniques that have been at the forefront of the quest to improve indoor location. While the RTT technique determines location using the time it takes for a signal to complete a round trip, the RSSI approach bases its calculations on the signal intensity received.

To assess the viability and accuracy of these methods, an experimental setup has been designed. Within a defined indoor space, four strategically placed hotspots will serve as the primary signal sources. As a user navigates this space, the system will continuously track and pinpoint the user's location using both RSSI and RTT methods. By capturing these real-time positional data points, we aim to contrast the performance of both methods, determining which offers superior accuracy in tracking a user's movement within the indoor environment.

This chapter will detail the design considerations and implementation aspects of this experimental indoor navigator system, setting the foundation for the subsequent comparative analysis of RSSI and RTT.

3.2 Machine Learning

In an endeavour to push the boundaries of accuracy and granularity in positioning, this research introduces an extended implementation fortified with machine learning capabilities. This advanced system harnesses the power of data generation to build a robust model or classifier, intricately mapping the space in which navigation occurs. Such a model is then deployed to predict the position of a mobile device, navigating the space in real-time. Impressively, the granularity of this system is such that it doesn't merely identify rooms, but distinct areas within these rooms.

The starting block of this transformative journey was the EloquentArduino article titled “Wi-Fi indoor positioning using Arduino and Machine Learning in 4 steps”. Although a valuable resource for initiating classifier training and usage, it was inherently limited, being compatible exclusively with RSSI measurements and the Arduino Integrated Development Environment (IDE). Addressing these constraints, modifications were undertaken to seamlessly integrate these principles into our broader system and crucially, to incorporate RTT measurements, enriching the positioning data.

3.3 Plan layout

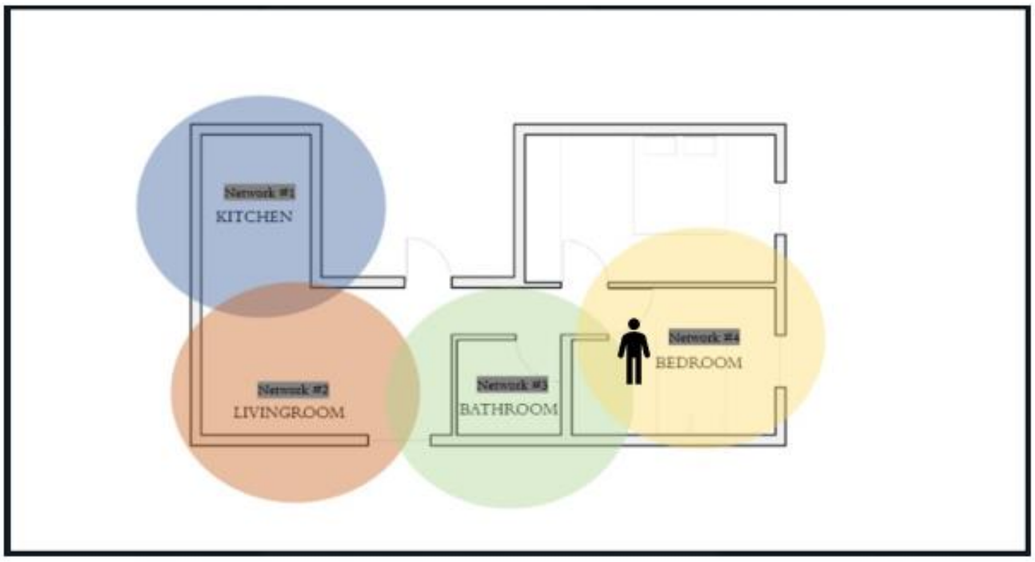


Figure 5 - Indoor positioning floor plan layout

This paragraph discusses the limitations and challenges of the proposed system and suggests some possible improvements. The proposed system uses ESP32 devices as access points (APs) and Wi-Fi Low Energy (WIFI LE) signals as beacons to track the location of an object. However, this system has low accuracy and coverage area due to the hardware constraints. The system could use different kinds of beacons, such as Radio-Frequency Identification (RFID) beacons, Wi-Fi beacons, or GPS signals, to improve accuracy and coverage area. These beacons have higher power, range, and precision than WIFI LE signals. However, they also have higher complexity and cost, both at the hardware level and at the server level. The server needs to handle multiple beacons with different physical addresses (mac addresses) and different technologies. The server also needs to use different methods to convert the received signal strength indicator (RSSI) values to distance estimates for each technology. Therefore, this thesis proposes to use WIFI LE signals as a straightforward and effective internal positioning solution but acknowledges that other solutions may offer better performance in some scenarios.

3.4 Hardware.

3.4.1 ESP32-S2 Microcontroller by Espressif Systems

The ESP32-S2 microcontroller, a product of Espressif Systems, represents an evolution in the domain of embedded systems optimized for Wi-Fi connectivity. It follows the lineage of the widely acknowledged ESP32 microcontroller, yet, distinctively, forgoes Bluetooth capabilities to exclusively focus on Wi-Fi.

Figure 6 - ESP32-S2

3.4.2 Architectural Features

The architecture boasts an Xtensa single-core 32-bit LX7 microprocessor that operates at speeds up to 240 MHz. This is complemented by an internal memory configuration comprising 320 KB of SRAM and 128 KB of ROM. For applications demanding augmented memory, the architecture facilitates support for external flash and SRAM.

3.4.3 Wi-Fi Capabilities

Operating within the 2.4 GHz band, the ESP32-S2 conforms to 802.11 b/g/n Wi-Fi standards. Its versatility is evident in its support for multiple Wi-Fi modes, including STA, SoftAP,

SoftAP+STA, and P2P. Furthermore, it upholds contemporary security standards with WPA3 security integration.

3.4.4 Interface and Peripherals

A broad spectrum of applications is facilitated through its 43 programmable GPIOs and 14 capacitive touch sensing IOs. Standard communication protocols such as SPI, I^2C, I^2S, and UART are natively supported. Unique to its design is the USB OTG and the presence of ADC and DAC capabilities.

3.4.5 Emphasis on Security

Recognizing the exigencies of modern connected devices, the ESP32-S2 is engineered with a secure boot process, flash encryption, and provisions for Trusted Applications supported from isolated flash areas.

3.4.6 Power Management

Addressing the growing need for energy-efficient solutions, the microcontroller offers various power modes. It integrates a battery charging circuit, making it compatible with lithium-ion and lithium-polymer energy sources. A notable feature is its ultra-low-power co-processor that can sustain functionality during deep sleep, allowing for tasks like sensor monitoring.

3.4.7 Software Development Ecosystem

The ESP32-S2 is backed by Espressif's official IoT Development Framework, known as ESP-IDF. The robust community support manifests in a plethora of libraries, especially in the Arduino ecosystem. Additionally, compatibility with platforms like MicroPython broadens its accessibility for developers.

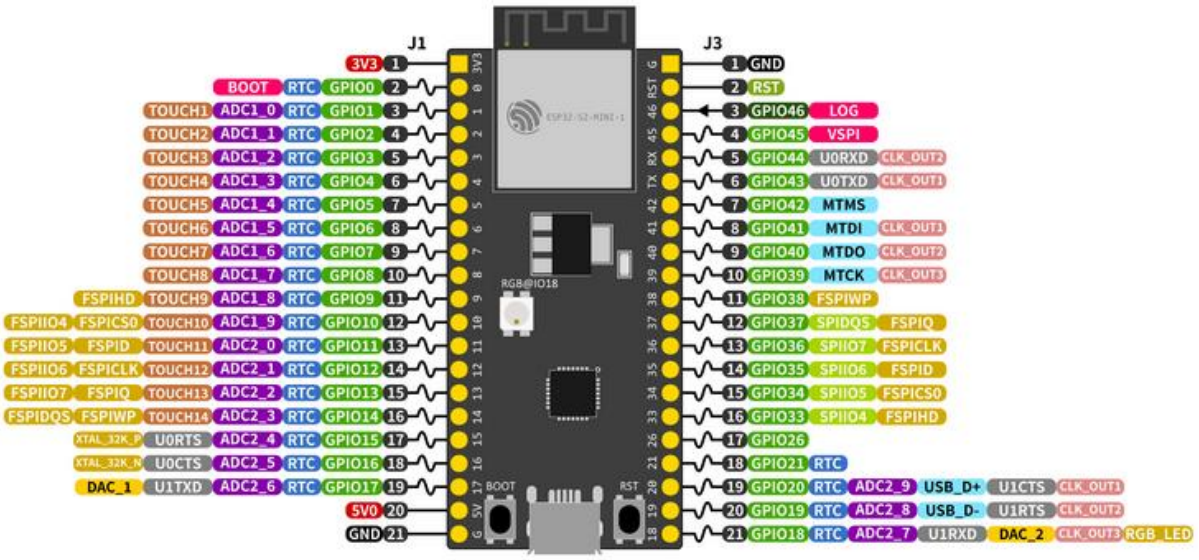

J1
J3
3V3 1
BOOT RTC GPIO0 2
TOUCH1 ADC1_0 RTC GPIO1 3
TOUCH2 ADC1_1 RTC GPIO2 4
TOUCH3 ADC1_2 RTC GPIO3 5
TOUCH4 ADC1_3 RTC GPIO4 6
TOUCH5 ADC1_4 RTC GPIO5 7
TOUCH6 ADC1_5 RTC GPIO6 8
TOUCH7 ADC1_6 RTC GPIO7 9
TOUCH8 ADC1_7 RTC GPIO8 10
FSPIHD TOUCH9 ADC1_8 RTC GPIO9 11
FSPIIO4 FSPICS0 TOUCH10 ADC1_9 RTC GPIO10 12
FSPIIO5 FSPID TOUCH11 ADC2_0 RTC GPIO11 13
FSPIIO6 FSPICLK TOUCH12 ADC2_1 RTC GPIO12 14
FSPIIO7 FSPIQ TOUCH13 ADC2_2 RTC GPIO13 15
FSPIDQS FSPIWP TOUCH14 ADC2_3 RTC GPIO14 16
XTAL_32K_P U0RTS ADC2_4 RTC GPIO15 17
XTAL_32K_N U0CTS ADC2_5 RTC GPIO16 18
DAC_1 U1TXD ADC2_6 RTC GPIO17 19
5V0 20
GND 21
1 GND
2 RST
3 GPIO46 LOG
4 GPIO45 VSPI
5 GPIO44 U0RXD CLK_OUT2
6 GPIO43 U0TXD CLK_OUT1
7 GPIO42 MTMS
8 GPIO41 MTDI CLK_OUT1
9 GPIO40 MTDO CLK_OUT2
10 GPIO39 MTCK CLK_OUT3
11 GPIO38 FSPIWP
12 GPIO37 SPIDQS FSPIQ
13 GPIO36 SPIIO7 FSPICLK
14 GPIO35 SPIIO6 FSPID
15 GPIO34 SPIIO5 FSPICS0
16 GPIO33 SPIIO4 FSPIHD
17 GPIO26
18 GPIO21 RTC
19 GPIO20 RTC ADC2_9 USB_D+ U1CTS CLK_OUT1
20 GPIO19 RTC ADC2_8 USB_D- U1RTS CLK_OUT2
21 GPIO18 RTC ADC2_7 U1RXD DAC_2 CLK_OUT3 RGB_LED
RGB@IO18
BOOT
RST

Figure 7 - ESP32 S2 Pin Layout

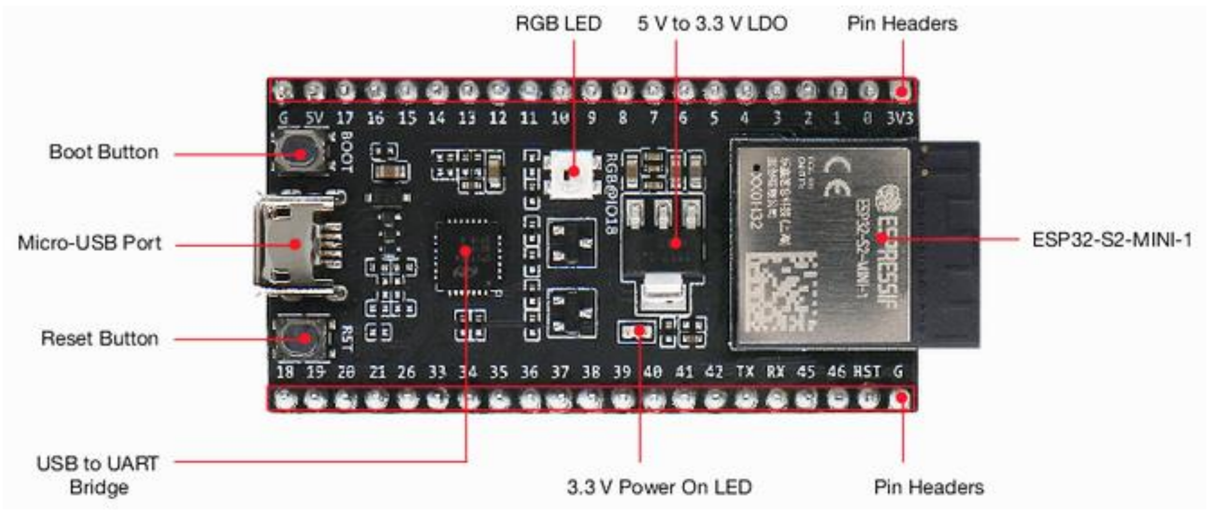


Figure 8 - ESP32 S2 front

3.5 Software

3.5.1 Espressif's IoT Development Framework (ESP-IDF)

The technological foundation for Wi-Fi and Bluetooth microcontrollers developed by Espressif Systems lies in its dedicated development framework, the ESP-IDF. This section provides a comprehensive analysis of the ESP-IDF, highlighting its features, setup process, and relevance in the IoT ecosystem.

3.5.2 Core Attributes of the ESP-IDF

ESP-IDF is endowed with a wide-ranging set of libraries catering to diverse peripherals, protocols, and services, from foundational ones like Wi-Fi and Bluetooth to sophisticated functionalities like Over-the-Air (OTA) updates. One of its distinctive attributes is the inclusion of the FreeRTOS real-time operating system, specifically tailored for Espressif's microcontrollers, enabling intricate multitasking capabilities. Complementary development utilities are integrated, such as tools for building, flashing, and application monitoring. The framework's intrinsic modularity allows for easy augmentation with custom components, and its menuconfig tool facilitates user-driven configuration.

3.5.3 Establishing the ESP-IDF Development Environment

The instantiation of the ESP-IDF environment mandates several procedural steps:

1. **Preparation**: Based on the operative system—whether Windows, Linux, or macOS—certain foundational software, including Python, might be prerequisite.
2. **Installation**: This entails cloning the ESP-IDF repository from its official GitHub repository, followed by the initialization of environment variables and the essential toolchain acquisition.
3. **Project Creation:** Leveraging templates or commencing a bespoke project.
4. **Configuration:** Utilization of `menuconfig` for software customization.
5. **Compilation and Deployment:** The software code is compiled and subsequently flashed onto the designated ESP32 device.
6. **Real-time Monitoring:** Employing inherent tools for debugging and log inspection.

Figure 9 - Espressif ESP-IDF

Chapter 4 - Implementation methods

4.1 Moving device firmware architecture.

The main components of the moving device firmware look like:

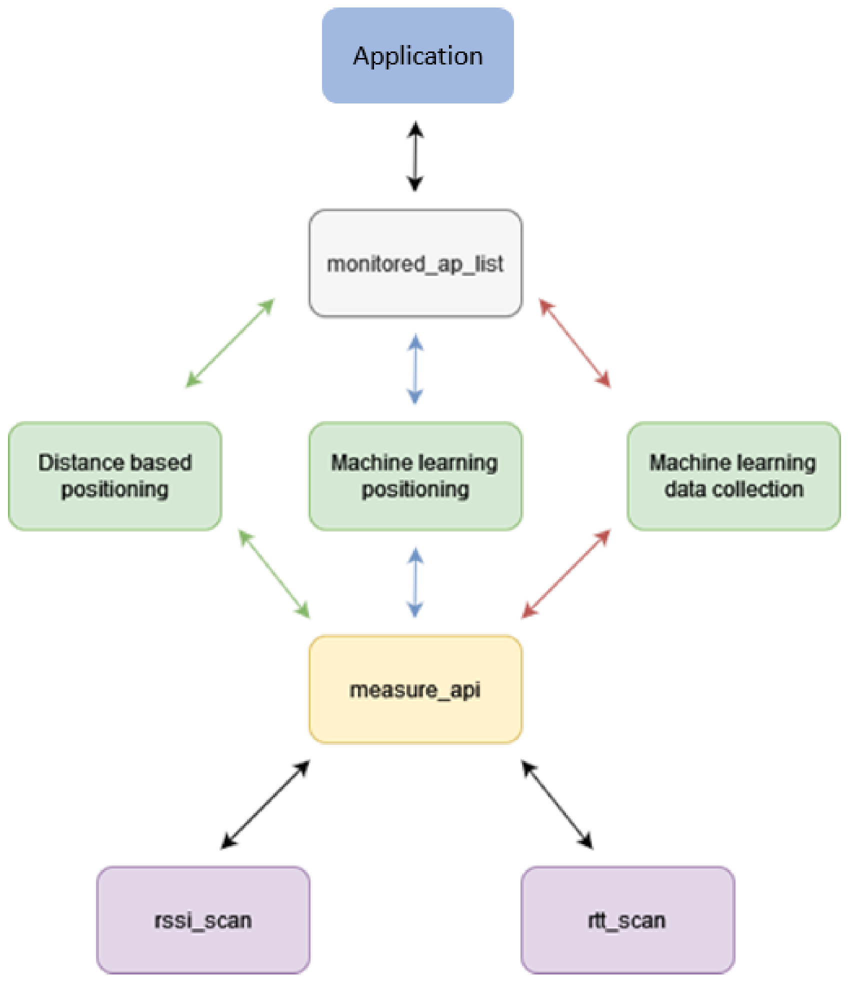

Application
monitored_ap_list
Distance based positioning
Machine learning positioning
Machine learning data collection
measure_api
rssi_scan
rtt_scan

Figure 10 - Flowchart Moving Device Firmware Architecture

4.2 WiFi RSSI Measurement

Within the context of indoor positioning systems, the precision with which the Received Signal Strength Indicator (RSSI) of extant Wi-Fi Access Points (APs) is gauged remains paramount. This chapter delves into the intricacies of a software module meticulously crafted for the ESP32-S2 platform, the primary purpose of which is to probe and quantitatively measure the RSSI values corresponding to accessible Wi-Fi APs.

The primary goal of this module (rssi_scan) is to initialize the necessary hardware and software components for Wi-Fi scanning, perform the scan, and retrieve the RSSI values of available APs.

4.2.1 Code Overview

The provided module consists of two main functions: one for initializing Wi-Fi scanning capabilities and another to perform the actual scan.

```
/**
 * Initializes the WiFi scanning
 *
 * @return void
 */
void wifi_scan_init() {
    /* Initialize the network interface and the WiFi */
    ESP_ERROR_CHECK(esp_netif_init());
    ESP_ERROR_CHECK(esp_event_loop_create_default());
    esp_netif_t *sta_netif = esp_netif_create_default_wifi_sta();
    assert(sta_netif);

    /* Create a WiFi config with the default values */
    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    /* Initialize the WiFi with the config */
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));

    /* Set the mode to STA and start the WiFi */
    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA));
    ESP_ERROR_CHECK(esp_wifi_start());
}

/**
 * Performs a scan for all the available WiFi APs and measures their RSSI values
 *
 * @return void
 */
void wifi_scan(wifi_ap_record_t* ap_info, uint16_t ap_info_size, uint16_t* ap_count) {
    /* Start a WiFi scan */
    esp_wifi_scan_start(NULL, true);
    /* Get the scan results to the provided structure */
    ESP_ERROR_CHECK(esp_wifi_scan_get_ap_records(&ap_info_size, ap_info));
    /* Get the number of scanned APs */
    ESP_ERROR_CHECK(esp_wifi_scan_get_ap_num(ap_count));
    /* Log the number of scanned APs */
    ESP_LOGD(TAG, "Total APs scanned = %u", *ap_count);
}
```

```
enum measurement_method_e {
    RSSI_MEASUREMENT = 0,
    RTT_MEASUREMENT
};

/* Select measurement type here (RTT or RSSI) */
static const measurement_method_e MEASUREMENT_METHOD = RSSI_MEASUREMENT;
```

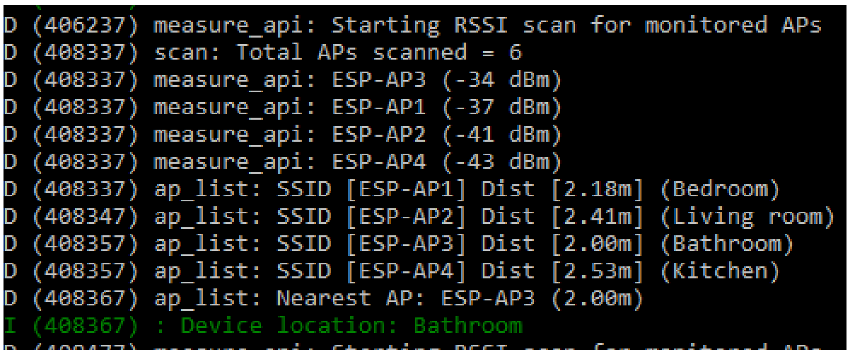
4.2.2 Explanation of the Code:

Initialization of WiFi Scanning (wifi_scan_init):

- The function starts by initializing the network interface required for Wi-Fi operations.
- An event loop is created to handle Wi-Fi-related events.
- A default station interface (sta_netif) is created for the WiFi operations. This station interface acts as a Wi-Fi client.
- The function then sets up a default Wi-Fi configuration and initializes the Wi-Fi driver using this configuration.
- Finally, the Wi-Fi mode is set to "station" (STA), and Wi-Fi is started.

Performing the Wi-Fi Scan (wifi_scan):

- The function triggers a Wi-Fi scan. This scan searches for all available APs in the vicinity.
- After the scan is complete, the function retrieves the scan results (AP details and RSSI values) into the provided structure (ap_info).
- The total number of APs scanned is also retrieved and logged.



```
D (406237) measure_api: Starting RSSI scan for monitored APs
D (408337) scan: Total APs scanned = 6
D (408337) measure_api: ESP-AP3 (-34 dBm)
D (408337) measure_api: ESP-AP1 (-37 dBm)
D (408337) measure_api: ESP-AP2 (-41 dBm)
D (408337) measure_api: ESP-AP4 (-43 dBm)
D (408337) ap_list: SSID [ESP-AP1] Dist [2.18m] (Bedroom)
D (408347) ap_list: SSID [ESP-AP2] Dist [2.41m] (Living room)
D (408357) ap_list: SSID [ESP-AP3] Dist [2.00m] (Bathroom)
D (408357) ap_list: SSID [ESP-AP4] Dist [2.53m] (Kitchen)
D (408367) ap_list: Nearest AP: ESP-AP3 (2.00m)
I (408367) : Device location: Bathroom
```

Figure 11 - Output of RSSI metrics

4.3 Wi-Fi RTT Measurement

Wi-Fi Round Trip Time (RTT) quantifies the duration requisite for a signal to traverse from its origin to its target and subsequently return. Given this metric, it becomes feasible to deduce the linear distance between two devices, assuming an unobstructed line of sight. The ESP32-S2 microcontroller is endowed with specific attributes tailored to facilitate these evaluations with efficacy.

The term "RTT" represents "Round Trip Time", which denotes the duration required for data packets to journey between two distinct nodes. A meticulous comprehension of the Time of Flight (ToF) between communications facilitates the calculation of the distance separating them. Occasionally, this measurement is referenced as FTM (Fine Timing Measurement), highlighting the imperative of a high-precision timing mechanism for such assessments.

The wireless protocol 802.11mc delineates the RTT mechanism adopted for Wi-Fi, as detailed in the IEEE standards (refer to IEEE 802.11mc). The choice of the ESP32-S2 microcontroller was predicated upon its explicit compatibility with the 802.11mc standard. Within the ESP32-S2 environment, a distinctive Wi-Fi API is employed for measurements.

To compute the distance, one employs the established formula:

Distance = RTT × c

Here, cc represents the speed of light.

The determination of Wi-Fi RTT employs a meticulous procedure titled "Fine Timing Measurement (FTM)". During FTM, a succession of action frames is transmitted from one apparatus to another, each eliciting an acknowledgment (ACK). Integral hardware within the devices registers the Time-of-Arrival (TOA) and Time-of-Departure (TOD) pertaining to each set of action frames and ACKs. Subsequently, the instigating entity aggregates data from all frame pairs, determining the RTT for each using the equation:

RTT[i]=(T4−T1) −(T3−T2)

Where:

- T1: signifies the moment of dispatch of the action frame from the responder.
- T2: marks the moment the action frame reaches the initiator.
- T3: denotes the departure moment of the ACK from the initiator.
- T4: indicates the moment the ACK is received by the responder.

This metric ultimately elucidates the duration required for data packets to navigate between the two nodes.

4.3.1 Code Overview & Explanation

Initialization: ftm_wifi_init() function initializes the WiFi peripheral, sets it in station mode, and readies it for RTT measurements by registering appropriate callbacks.

```
/**
 * Initializes the WiFi peripheral for FTM measurements
 *
 * @return void
 */
void ftm_wifi_init(void) {
    /* Set the log level of the WiFi component according to the configured global log level */
    if(MOVING_DEV_LOG_LEVEL == ESP_LOG_DEBUG) {
        esp_log_level_set("wifi", ESP_LOG_WARN);
    } else {
        esp_log_level_set("wifi", ESP_LOG_ERROR);
    }

    /* If the measurement has already been initialized, return */
    static bool initialized = false;
    if(initialized) return;

    /* Initialize the network interface */
    ESP_ERROR_CHECK(esp_netif_init());
    /* Create the event group used for signaling results */
    ftm_event_group = xEventGroupCreate();
    /* Configure the WiFi with the default parameters */
    ESP_ERROR_CHECK(esp_event_loop_create_default());
    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));
    /* Register the FTM event handler */
    ESP_ERROR_CHECK(esp_event_handler_instance_register(WIFI_EVENT,
                    WIFI_EVENT_FTM_REPORT,
                    &ftm_report_handler,
                    NULL,
                    NULL));
    /* Store the WiFi config only in RAM */
    ESP_ERROR_CHECK(esp_wifi_set_storage(WIFI_STORAGE_RAM));
    /* Set the WiFi mode to STA and start it */
    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA));
    ESP_ERROR_CHECK(esp_wifi_start());
    /* Set the initialized variable to true to ensure that it's only done once */
    initialized = true;
}
```

WiFi Scanning: ftm_wifi_scan(bool internal) kicks off a scanning process for available APs. Scan results get buffered in g_ap_list_buffer.

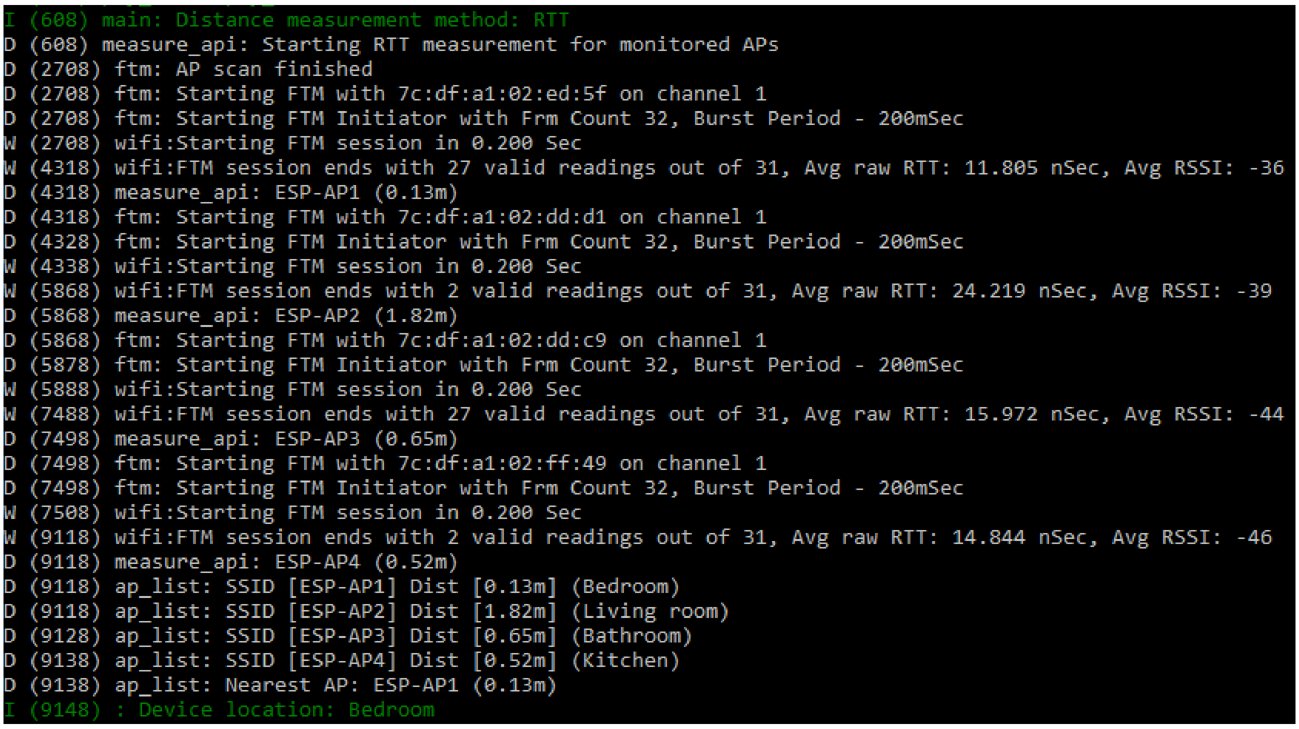
```
/**
 * Performs a WiFi scan for available APs
 *
 * @param bool internal - on 'true' no details about the scan will be logged
 *
 * @return bool - 'true' if the scan was successful, 'false' otherwise
 */
bool ftm_wifi_scan(bool internal) {
    /* Start a scan for the available APs */
    ESP_ERROR_CHECK(esp_wifi_scan_start(NULL, true));

    /* Get the number of the scanned APs */
    esp_wifi_scan_get_ap_num(&g_scan_ap_num);
    /* If there are no APs available, return after printing a warning message */
    if(g_scan_ap_num == 0) {
        ESP_LOGD(TAG_STA, "No APs found");
        return false;
    }

    /* If g_ap_list_buffer is already initialzed, free it */
    if(g_ap_list_buffer) {
        free(g_ap_list_buffer);
    }
    /* Allocate memory for g_ap_list_buffer */
    g_ap_list_buffer = (wifi_ap_record_t*)malloc(g_scan_ap_num * sizeof(wifi_ap_record_t));
    /* If the allocation fails, return with an error */
    if(g_ap_list_buffer == NULL) {
        ESP_LOGE(TAG_STA, "Failed to malloc buffer to print scan results");
        return false;
    }

    /* Get the scan results to g_ap_list_buffer */
    if(esp_wifi_scan_get_ap_records(&g_scan_ap_num, (wifi_ap_record_t *)g_ap_list_buffer) ==
ESP_OK) {
        /* If the scan was not internal, print the results */
        if(!internal) {
            for(uint8_t i = 0; i < g_scan_ap_num; i++) {
                ESP_LOGD(TAG_STA, "[%s][rssi=%d]""%s", g_ap_list_buffer[i].ssid,
g_ap_list_buffer[i].rssi,
                         g_ap_list_buffer[i].ftm_responder ? "[FTM Responder]" : "");
            }
        }
    }

    /* Return with success */
    ESP_LOGD(TAG_STA, "AP scan finished");
    return true;
}
```

Responder Detection: Prior to the actual RTT measurement, the ftm_find_responder_ap(const char* ssid) function is used to identify whether the chosen AP supports FTM.

```
/**
 * Checks that the specified AP is available and is capable of FTM measurements
 *
 * @param const char* ssid - SSID of the AP to be checked
 *
 * @return bool - 'true' if the specified AP is available and FTM capable, 'false' otherwise
 */
wifi_ap_record_t* ftm_find_responder_ap(const char* ssid) {

    /* If there was no SSID provided, return */
    if(!ssid) return NULL;

    /* If the scan result buffer is uninitialized, or there were no APs in range, return */
    if(!g_ap_list_buffer || (g_scan_ap_num == 0)) {
        return NULL;
    }

    /* Iterate through the scan results */
    for(uint8_t i = 0; i < g_scan_ap_num; i++) {
        /* If the requested SSID matches the current record */
        if(strcmp((const char *)g_ap_list_buffer[i].ssid, ssid) == 0) {
            /* Return the ap_record for the requested SSID */
            return &g_ap_list_buffer[i];
        }
    }

    /* Reaching this point means that the requested SSID was not found */
    return NULL;
}
```

RTT Measurement: The main functionality is in the ftm_wifi_measure function. It reaches out to the chosen AP to estimate distance. Should the measurement fail, an error gets returned.

```
/**
 * Performs an FTM distance measurment with the provided AP using the provided parameters
 *
 * @param const char* ssid - SSID of the AP to be measured
 * @param int frm_count - FTM frame count
 * @param int burst_period - length of the FTM burst period divided by 100
 * @param int mode - FTM mode
 * @param uint32_t* result - pointer to the variable where the measured distance will be stored
 *
 * @return bool - 'true' if the measurement has been successful, 'false' otherwise
 */
bool ftm_wifi_measure(const char* ssid, int frm_count, int burst_period, int mode, uint32_t*
result) {

    /* The AP record which will hold information about the measured access point */
    wifi_ap_record_t *ap_record;

    /* The FTM config */
    wifi_ftm_initiator_cfg_t ftmi_cfg;
    ftmi_cfg.frm_count = 32;
    ftmi_cfg.burst_period = 2;

    /* Get the AP record for the measured device */
    ap_record = ftm_find_responder_ap(ssid);
    /* If an ap_record is available */
    if(ap_record) {
        /* Log the starting message */
        ESP_LOGD(TAG_STA, "Starting FTM with %s (" MACSTR ") on channel %d", ssid,
MAC2STR(ap_record->bssid), ap_record->primary);
        /* Configure the FTM measurement with the SSID and the channel */
        memcpy(ftmi_cfg.resp_mac, ap_record->bssid, 6);
        ftmi_cfg.channel = ap_record->primary;
    } else {
        ESP_LOGE(TAG_STA, "No matching AP found for %s", ssid);
        return false;
    }

    /* Set the FTM frame count after range-checking it */
    uint8_t count = frm_count;
    if(count != 0 && count != 16 && count != 24 &&
        count != 32 && count != 64) {
        count = 0;
    }
    ftmi_cfg.frm_count = count;

    /* Set the FTM burst period after range-checking it */
    if(burst_period >= 2 && burst_period < 256) {
        ftmi_cfg.burst_period = burst_period;
    } else {
        ftmi_cfg.burst_period = 0;
    }

    /* 0 indicates no preference for the FTM burst period */
    if(ftmi_cfg.burst_period == 0) {
        ESP_LOGD(TAG_STA, "Starting FTM Initiator with Frm Count %d, Burst Period - No
Preference",
                 ftmi_cfg.frm_count);
    } else {
        ESP_LOGD(TAG_STA, "Starting FTM Initiator with Frm Count %d, Burst Period - %dmSec",
                 ftmi_cfg.frm_count, ftmi_cfg.burst_period * 100);
    }

    /* Start the FTM measurement session */
    if(ESP_OK != esp_wifi_ftm_initiate_session(&ftmi_cfg)) {
        /* If the start failed, return with an error */
        ESP_LOGE(TAG_STA, "Failed to start FTM session");
        return false;
    }

    /* Wait for the ftm_event_group bits to be set by the FTM callback */
    EventBits_t bits = xEventGroupWaitBits(ftm_event_group, FTM_REPORT_BIT | FTM_FAILURE_BIT,
pdFALSE, pdFALSE, portMAX_DELAY);
    bool ret = false;
    /* If the session was successful */
    if(bits & FTM_REPORT_BIT) {
        /* Get the measurement result */
        *result = measured_distance;

        /* Print the detailed RTT log if it's enabled */
        if(RTT_DETAILED_MEASUREMENT_LOGGING_ENABLED) ftm_log_detailed_report();

        /* Free the report */
        free(g_ftm_report);
        g_ftm_report = NULL;
        g_ftm_report_num_entries = 0;

        ret = true;
    } else if(bits & FTM_FAILURE_BIT) {
        ESP_LOGE(TAG_STA, "Measurement failed");
    } else {
        /* If the session failed/timed out, return with an error */
        ESP_LOGE(TAG_STA, "No response");
    }

    /* End the session */
    esp_wifi_ftm_end_session();
    /* Clear the event bits */
    xEventGroupClearBits(ftm_event_group, FTM_REPORT_BIT);
    xEventGroupClearBits(ftm_event_group, FTM_FAILURE_BIT);

    return ret;
}
```



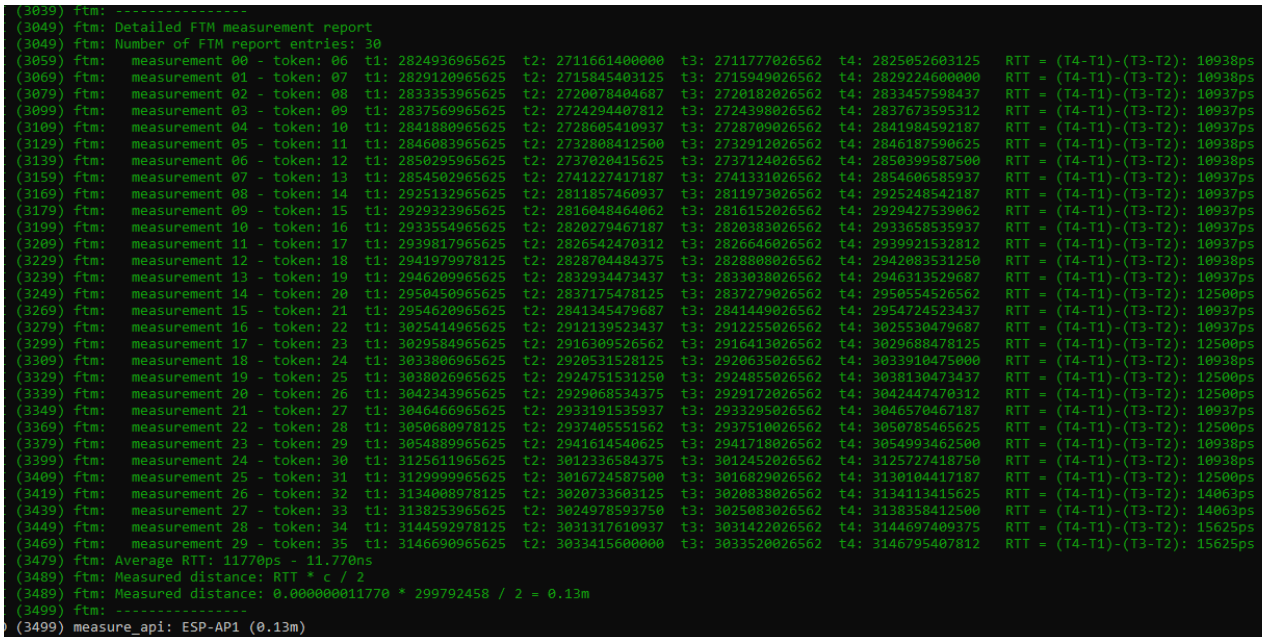
```
I (608) main: Distance measurement method: RTT
D (608) measure_api: Starting RTT measurement for monitored APs
D (2708) ftm: AP scan finished
D (2708) ftm: Starting FTM with 7c:df:a1:02:ed:5f on channel 1
D (2708) ftm: Starting FTM Initiator with Frm Count 32, Burst Period - 200mSec
W (2708) wifi:Starting FTM session in 0.200 Sec
W (4318) wifi:FTM session ends with 27 valid readings out of 31, Avg raw RTT: 11.805 nSec, Avg RSSI: -36
D (4318) measure_api: ESP-AP1 (0.13m)
D (4318) ftm: Starting FTM with 7c:df:a1:02:dd:d1 on channel 1
D (4328) ftm: Starting FTM Initiator with Frm Count 32, Burst Period - 200mSec
W (4338) wifi:Starting FTM session in 0.200 Sec
W (5868) wifi:FTM session ends with 2 valid readings out of 31, Avg raw RTT: 24.219 nSec, Avg RSSI: -39
D (5868) measure_api: ESP-AP2 (1.82m)
D (5868) ftm: Starting FTM with 7c:df:a1:02:dd:c9 on channel 1
D (5878) ftm: Starting FTM Initiator with Frm Count 32, Burst Period - 200mSec
W (5888) wifi:Starting FTM session in 0.200 Sec
W (7488) wifi:FTM session ends with 27 valid readings out of 31, Avg raw RTT: 15.972 nSec, Avg RSSI: -44
D (7498) measure_api: ESP-AP3 (0.65m)
D (7498) ftm: Starting FTM with 7c:df:a1:02:ff:49 on channel 1
D (7498) ftm: Starting FTM Initiator with Frm Count 32, Burst Period - 200mSec
W (7508) wifi:Starting FTM session in 0.200 Sec
W (9118) wifi:FTM session ends with 2 valid readings out of 31, Avg raw RTT: 14.844 nSec, Avg RSSI: -46
D (9118) measure_api: ESP-AP4 (0.52m)
D (9118) ap_list: SSID [ESP-AP1] Dist [0.13m] (Bedroom)
D (9118) ap_list: SSID [ESP-AP2] Dist [1.82m] (Living room)
D (9128) ap_list: SSID [ESP-AP3] Dist [0.65m] (Bathroom)
D (9138) ap_list: SSID [ESP-AP4] Dist [0.52m] (Kitchen)
D (9138) ap_list: Nearest AP: ESP-AP1 (0.13m)
I (9148) : Device location: Bedroom
```

Figure 12 - Output of RTT measurements

In the provided above log output snapshot concerning RTT measurement, we observe the systematic execution of processes across four pre-configured access points (APs). Initially, there is an evident AP scan, followed by a quartet of RTT measurements executed under predetermined parameters. The culmination of each measurement presents its respective result. Conclusively, the log delineates the proximate AP location.

4.3.2 RTT Calculation

In the below figure, a meticulously constructed report detailing each measurement, we are presented with a structured analysis that sheds light on the intricacies of the measurement process. This format serves to break down the multifaceted components of the measurement for clarity and comprehensive understanding.

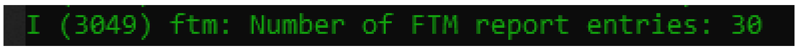


```
(3039) ftm: ----------------
(3049) ftm: Detailed FTM measurement report
(3049) ftm: Number of FTM report entries: 30
(3059) ftm:   measurement 00 - token: 06  t1: 2824936965625  t2: 2711661400000  t3: 2711777026562  t4: 2825052603125   RTT = (T4-T1)-(T3-T2): 10938ps
(3069) ftm:   measurement 01 - token: 07  t1: 2829120965625  t2: 2715845403125  t3: 2715949026562  t4: 2829224600000   RTT = (T4-T1)-(T3-T2): 10938ps
(3079) ftm:   measurement 02 - token: 08  t1: 2833353965625  t2: 2720078404687  t3: 2720182026562  t4: 2833457598437   RTT = (T4-T1)-(T3-T2): 10937ps
(3099) ftm:   measurement 03 - token: 09  t1: 2837569965625  t2: 2724294407812  t3: 2724398026562  t4: 2837673595312   RTT = (T4-T1)-(T3-T2): 10937ps
(3109) ftm:   measurement 04 - token: 10  t1: 2841880965625  t2: 2728605410937  t3: 2728709026562  t4: 2841984592187   RTT = (T4-T1)-(T3-T2): 10937ps
(3129) ftm:   measurement 05 - token: 11  t1: 2846083965625  t2: 2732808412500  t3: 2732912026562  t4: 2846187590625   RTT = (T4-T1)-(T3-T2): 10938ps
(3139) ftm:   measurement 06 - token: 12  t1: 2850295965625  t2: 2737020415625  t3: 2737124026562  t4: 2850399587500   RTT = (T4-T1)-(T3-T2): 10938ps
(3159) ftm:   measurement 07 - token: 13  t1: 2854502965625  t2: 2741227417187  t3: 2741331026562  t4: 2854606585937   RTT = (T4-T1)-(T3-T2): 10937ps
(3169) ftm:   measurement 08 - token: 14  t1: 2925132965625  t2: 2811857460937  t3: 2811973026562  t4: 2925248542187   RTT = (T4-T1)-(T3-T2): 10937ps
(3179) ftm:   measurement 09 - token: 15  t1: 2929323965625  t2: 2816048464062  t3: 2816152026562  t4: 2929427539062   RTT = (T4-T1)-(T3-T2): 10937ps
(3199) ftm:   measurement 10 - token: 16  t1: 2933554965625  t2: 2820279467187  t3: 2820383026562  t4: 2933658535937   RTT = (T4-T1)-(T3-T2): 10937ps
(3209) ftm:   measurement 11 - token: 17  t1: 2939817965625  t2: 2826542470312  t3: 2826646026562  t4: 2939921532812   RTT = (T4-T1)-(T3-T2): 10937ps
(3229) ftm:   measurement 12 - token: 18  t1: 2941979978125  t2: 2828704484375  t3: 2828808026562  t4: 2942083531250   RTT = (T4-T1)-(T3-T2): 10938ps
(3239) ftm:   measurement 13 - token: 19  t1: 2946209965625  t2: 2832934473437  t3: 2833038026562  t4: 2946313529687   RTT = (T4-T1)-(T3-T2): 10937ps
(3249) ftm:   measurement 14 - token: 20  t1: 2950450965625  t2: 2837175478125  t3: 2837279026562  t4: 2950554526562   RTT = (T4-T1)-(T3-T2): 12500ps
(3269) ftm:   measurement 15 - token: 21  t1: 2954620965625  t2: 2841345479687  t3: 2841449026562  t4: 2954724523437   RTT = (T4-T1)-(T3-T2): 10937ps
(3279) ftm:   measurement 16 - token: 22  t1: 3025414965625  t2: 2912139523437  t3: 2912255026562  t4: 3025530479687   RTT = (T4-T1)-(T3-T2): 10937ps
(3299) ftm:   measurement 17 - token: 23  t1: 3029584965625  t2: 2916309526562  t3: 2916413026562  t4: 3029688478125   RTT = (T4-T1)-(T3-T2): 12500ps
(3309) ftm:   measurement 18 - token: 24  t1: 3033806965625  t2: 2920531528125  t3: 2920635026562  t4: 3033910475000   RTT = (T4-T1)-(T3-T2): 10938ps
(3329) ftm:   measurement 19 - token: 25  t1: 3038026965625  t2: 2924751531250  t3: 2924855026562  t4: 3038130473437   RTT = (T4-T1)-(T3-T2): 12500ps
(3339) ftm:   measurement 20 - token: 26  t1: 3042343965625  t2: 2929068534375  t3: 2929172026562  t4: 3042447470312   RTT = (T4-T1)-(T3-T2): 12500ps
(3349) ftm:   measurement 21 - token: 27  t1: 3046466965625  t2: 2933191535937  t3: 2933295026562  t4: 3046570467187   RTT = (T4-T1)-(T3-T2): 10937ps
(3369) ftm:   measurement 22 - token: 28  t1: 3050680978125  t2: 2937405551562  t3: 2937510026562  t4: 3050785465625   RTT = (T4-T1)-(T3-T2): 12500ps
(3379) ftm:   measurement 23 - token: 29  t1: 3054889965625  t2: 2941614540625  t3: 2941718026562  t4: 3054993462500   RTT = (T4-T1)-(T3-T2): 10938ps
(3399) ftm:   measurement 24 - token: 30  t1: 3125611965625  t2: 3012336584375  t3: 3012452026562  t4: 3125727418750   RTT = (T4-T1)-(T3-T2): 10938ps
(3409) ftm:   measurement 25 - token: 31  t1: 3129999965625  t2: 3016724587500  t3: 3016829026562  t4: 3130104417187   RTT = (T4-T1)-(T3-T2): 12500ps
(3419) ftm:   measurement 26 - token: 32  t1: 3134008978125  t2: 3020733603125  t3: 3020838026562  t4: 3134113415625   RTT = (T4-T1)-(T3-T2): 14063ps
(3439) ftm:   measurement 27 - token: 33  t1: 3138253965625  t2: 3024978593750  t3: 3025083026562  t4: 3138358412500   RTT = (T4-T1)-(T3-T2): 14063ps
(3449) ftm:   measurement 28 - token: 34  t1: 3144592978125  t2: 3031317610937  t3: 3031422026562  t4: 3144697409375   RTT = (T4-T1)-(T3-T2): 15625ps
(3469) ftm:   measurement 29 - token: 35  t1: 3146690965625  t2: 3033415600000  t3: 3033520026562  t4: 3146795407812   RTT = (T4-T1)-(T3-T2): 15625ps
(3479) ftm: Average RTT: 11770ps - 11.770ns
(3489) ftm: Measured distance: RTT * c / 2
(3489) ftm: Measured distance: 0.000000011770 * 299792458 / 2 = 0.13m
(3499) ftm: ----------------
(3499) measure_api: ESP-AP1 (0.13m)
```

Figure 13 - RTT Detailed Log

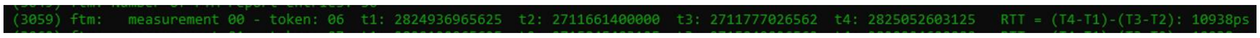
Calculation Explanation:

The first line states the number of measurements done with the FTM responder.



```
I (3049) ftm: Number of FTM report entries: 30
```

Next, we can look at each measurement in detail:



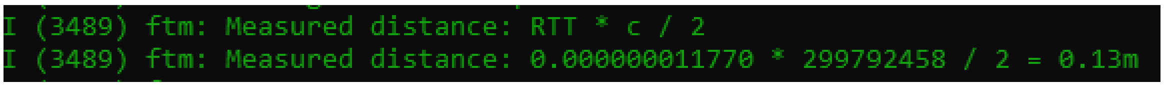
```
(3059) ftm:   measurement 00 - token: 06  t1: 2824936965625  t2: 2711661400000  t3: 2711777026562  t4: 2825052603125   RTT = (T4-T1)-(T3-T2): 10938ps
```

Explanation of each part:

- measurement 00 - the number of the measurement
- token - the ID of the measurement between the initiator and responder
- t1 - Time of Departure of action frame from the responder
- t2 - Time of Arrival of action frame at the initiator
- t3 - Time of Departure of ACK (acknowledge) from the initiator.
- t4 - Time of Arrival of the ACK at the responder
- RTT - round trip time calculated form the t(n) values, RTT = (t4-t1) -(t3-t2)

The numbers seen at t1, t2, t3 and t4 are raw timer values that only hold significance in relation to each other.

The below figure is the average RTT from all the measurements in picoseconds and in nanoseconds.



```
I (3489) ftm: Measured distance: RTT * c / 2
I (3489) ftm: Measured distance: 0.000000011770 * 299792458 / 2 = 0.13m
```

Figure 14 - Calculation the average RTT value

The above figure presents the formula for calculating the distance, and the measurement results applied to the formula with the final measured distance result in meters.

4.4 Measure API

The objective of this implementation is to provide a middle layer between the presence detection system and the distance measurement code. By introducing this abstraction layer, the upper layer of the software doesn't need to be aware of the specific method used for distance measurement.

4.4.1 Code Overview & Explanation

The provided code below is for an ESP32-S2-based indoor positioning system. It acts as a middle layer, offering a bridge between presence detection and distance measurement functionalities. The primary goal of this API (MeasureAPI) is to allow users to swap between two distance measurement techniques easily: RSSI (Received Signal Strength Indication) and RTT (Round-Trip Time). Here's a concise breakdown:

Distance Conversion: The `rssi_to_distance` function translates an RSSI value into a distance in meters, based on predefined RSSI values at specific reference distances.

```
/**
 * Converts an RSSI value to distance (meters)
 *
 * @param int rssi - RSSI value to be converted
 *
 * @return double - the RSSI value converted to meters
 */
double rssi_to_distance(int rssi) {
    return ((double)rssi) / (RSSI_AT_1_METER_DISTANCE - RSSI_AT_ZERO_DISTANCE);
}
```

RSSI-based Measurement:

`init_measurement_rssi()`: Initializes measurement using the RSSI method.

```
/**
 * Initializes the RSSI based measurement
 *
 * @return void
 */
void init_measurement_rssi() {
    wifi_scan_init();
}
```

`measure_ap_distances_rssi()`: Measures the distance to specified APs based on their RSSI values. It performs a WiFi scan to get the RSSI values for the APs and calculates the distance using the conversion function.

```
void measure_ap_distances_rssi(monitored_ap* aps_to_measure, uint16_t number_of_aps_to_measure)
{

    ESP_LOGD(TAG, "Starting RSSI scan for monitored APs");

    /* Create a structure which will hold the WiFi scan results */
    wifi_ap_record_t* ap_info = (wifi_ap_record_t*)malloc(DEFAULT_SCAN_LIST_SIZE *
sizeof(wifi_ap_record_t));
    memset(ap_info, 0, DEFAULT_SCAN_LIST_SIZE * sizeof(wifi_ap_record_t));
    uint16_t ap_count = 0u;

    /* Perform a scan for the available APs */
    wifi_scan(ap_info, DEFAULT_SCAN_LIST_SIZE, &ap_count);

    /* Iterate through the list of the scanned APs */
    for(int i = 0; (i < DEFAULT_SCAN_LIST_SIZE) && (i < ap_count); i++) {

        /* Iterate through the received AP list */
        for(int j = 0; j < number_of_aps_to_measure; j++) {
            /* If the current 'scanned AP' matches with the current 'received AP' */
            if(strcmp(aps_to_measure[j].ssid, (const char*)ap_info[i].ssid) == 0) {
                /* Store the results in the received AP list */
                aps_to_measure[j].distance_in_meters = rssi_to_distance(ap_info[i].rssi);
                aps_to_measure[j].in_range = true;
                /* Log the results */
                ESP_LOGD(TAG, "%s (%d dBm)", ap_info[i].ssid, ap_info[i].rssi);
            }
        }

    }

    /* Free the structure holding the WiFi scan results */
    free(ap_info);
}
```

RTT-based Measurement:

`init_measurement_rtt()`: Initializes the RTT-based measurement.

```
/**
 * Initializes the RTT based measurement
 *
 * @return void
 */
void init_measurement_rtt() {
    ftm_wifi_init();
}
```

`measure_ap_distances_rtt()`: Measures the distance to specific APs using RTT values. It initiates an RTT measurement for each AP, then logs and stores the results.

```
/**
 * Measures the distance to the monitroed APs using RTT values
 *
 * @param monitored_ap* aps_to_measure - List of APs to be measured
 * @param uint16_t number_of_aps_to_measure - The number of APs to be measured
 *
 * @return void
 */
void measure_ap_distances_rtt(monitored_ap* aps_to_measure, uint16_t number_of_aps_to_measure) {

    ESP_LOGD(TAG, "Starting RTT measurement for monitored APs");

    /* Perform an AP scan, return if it failed */
    if(!ftm_wifi_scan(true)) {
        ESP_LOGE(TAG, "WiFi discovery failed!");
        return;
    }

    /* Iterate through the received AP list and measure each one via the RTT method */
    uint32_t result = 0u;
    for(int i = 0; i < number_of_aps_to_measure; i++) {

        /* Perform a single RTT measurement with the configured values */
        if(ftm_wifi_measure(aps_to_measure[i].ssid, FTM_FRAME_COUNT, FTM_BURST_PERIOD,
FTM_INITIATOR_MODE, &result)) {
            /* The result is in 'cm', convert it to meters */
            double distance_in_meters = (double)result / 100;
            /* Store the results in the received AP list */
            aps_to_measure[i].distance_in_meters = distance_in_meters;
            aps_to_measure[i].in_range = true;
            /* Log the results */
            ESP_LOGD(TAG, "%s (%.02lfm)", aps_to_measure[i].ssid, distance_in_meters);
        } else {
            /* Error message when the RTT measurement fails for a certain configured AP */
            ESP_LOGE(TAG, "RTT measurement failed for %s", aps_to_measure[i].ssid);
        }

        /* Give the control back to the OS between each measurement to allow the WiFi stack to
perform */
        taskYIELD();
        vTaskDelay(100 / portTICK_PERIOD_MS);

    }

}
```

Unified Measurement Initialization:

`init_measurement()`: Depending on the configured method (either RSSI or RTT), it initializes the respective measurement method.

```
/**
 * Initializes the configured measurement
 *
 * @return void
 */
void init_measurement() {
    if(MEASUREMENT_METHOD == RSSI_MEASUREMENT) {
        init_measurement_rssi();
    } else if(MEASUREMENT_METHOD == RTT_MEASUREMENT) {
        init_measurement_rtt();
    }
}
```

Unified Distance Measurement:

`measure_ap_distances()`: Measures the distance to the monitored APs using the configured method (either RSSI or RTT), providing a consistent interface irrespective of the underlying measurement technique.

```
/**
 * Measures the distance to the monitroed APs using the configured method
 *
 * @param monitored_ap* aps_to_measure - List of APs to be measured
 * @param uint16_t number_of_aps_to_measure - The number of APs to be measured
 *
 * @return void
 */
void measure_ap_distances(monitored_ap* aps_to_measure, uint16_t number_of_aps_to_measure) {
    if(MEASUREMENT_METHOD == RSSI_MEASUREMENT) {
        measure_ap_distances_rssi(aps_to_measure, number_of_aps_to_measure);
    } else if(MEASUREMENT_METHOD == RTT_MEASUREMENT) {
        measure_ap_distances_rtt(aps_to_measure, number_of_aps_to_measure);
    }
}
```

4.5 Machine Learning

Indoor positioning systems are paramount for various applications ranging from asset tracking to location-based services. The effectiveness of these systems often depends on the quality and granularity of the data they collect. This chapter delves into the implementation details of the data collection module we devised for our indoor positioning system. We adopted the ESP32-S2 platform for this purpose, which provides a myriad of benefits, including efficient power consumption and Wi-Fi capabilities.

Our choice of model for position prediction is the Random Forest, as wrapped by the Eloquent's ML library for embedded systems. Random forests are ensemble learning methods, combining multiple decision trees to produce a more accurate and stable prediction.

Random Forest is an ensemble learning technique that builds upon the foundation of decision trees. Its name, "Random Forest," paints an accurate picture of its methodology: constructing a 'forest' of multiple 'trees' (decision trees) and using randomness during their creation. This chapter delves deep into the mechanics, advantages, and considerations of using the Random Forest classifier, particularly in the context of our indoor positioning system.

4.5.1 ML Model Train

The code presented in this section is tailored for the ESP32-S2, aiming to collect proximity data of monitored access points (APs). We chose a CSV-compatible format for data representation, making it more conducive for further analysis and processing.

4.5.2 Code Overview & Explanation

```
/**
 * Initializes the device for the Machine Learning classifier data collection
 *
 * @return void
 */
void ml_collect_data_init() {
    /* Wait a second */
    vTaskDelay(1000 / portTICK_PERIOD_MS);

    /* Configure UART0 for the data output */
    /* https://github.com/espressif/esp-
idf/blob/f3704f027e70a370bed3852f39c2ce48b9e8c7dc/examples/peripherals/uart_echo/main/uart_echo_
example_main.c */
    uart_config_t uart_config;
    uart_config.baud_rate = 115200;
    uart_config.data_bits = UART_DATA_8_BITS;
    uart_config.parity    = UART_PARITY_DISABLE;
    uart_config.stop_bits = UART_STOP_BITS_1;
    uart_config.flow_ctrl = UART_HW_FLOWCTRL_DISABLE;

    uart_param_config(UART_NUM_0, &uart_config);
    uart_set_pin(UART_NUM_0, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE,
UART_PIN_NO_CHANGE);
    uart_driver_install(UART_NUM_0, 2048u, 0, 0, NULL, 0);

    /* Raise the log level to silence other modules (to have a cleaner output) */
    esp_log_level_set("*", ESP_LOG_ERROR);
}
```

Function: *ml_collect_data_init()*

- Purpose: Prepares the ESP32-S2 for data collection.
- Procedure:
    1) Incorporates a delay for task synchronization.

2) Sets up UART0 for data output with specific configurations.
3) Adjusts the system's log level to prioritize essential messages.

```c
void ml_collect_data() {
    /* Clear the previous measurement */
    monitored_ap_list_clear();
    /* Perform a measurement with the configured method */
    monitored_ap_list_measure();

    /* If none of the configured APs are in range */
    if(!monitored_ap_list_check_any_in_range()) {
        ESP_LOGE(TAG, "All APs are out of range, data cannot be collected");
        return;
    }

    /* Buffer to hold the dataline which will be printed */
    char dataline[200] = "";

    /* Iterate through all the monitored APs and prepare a CSV compatible output from them */
    for(int i = 0; i < NUMBER_OF_MONITORED_APS; i++) {
        /* Buffer for the current data point */
        char datapoint[10];
        /* The delimiter character between the datapoints */
        char delimiter = ',';
        /* Don't put the delimiter after the last datapoint, put a newline instead */
        if(i == NUMBER_OF_MONITORED_APS - 1) delimiter = '\n';
        /* Print the current data to the datapoint buffer */
        sprintf(datapoint, "%.02lf%c", monitored_ap_list[i].distance_in_meters, delimiter);
        /* Concatenate the current datapoint buffer with the dataline buffer */
        strcat(dataline, datapoint);
    }

    /* Print the prepared dataline buffer to the output */
    uart_write_bytes(UART_NUM_0, dataline, strlen(dataline));

    /* Wait before the next measurement */
    vTaskDelay(1000 / portTICK_PERIOD_MS);
}
```

Function: *ml_collect_data()*

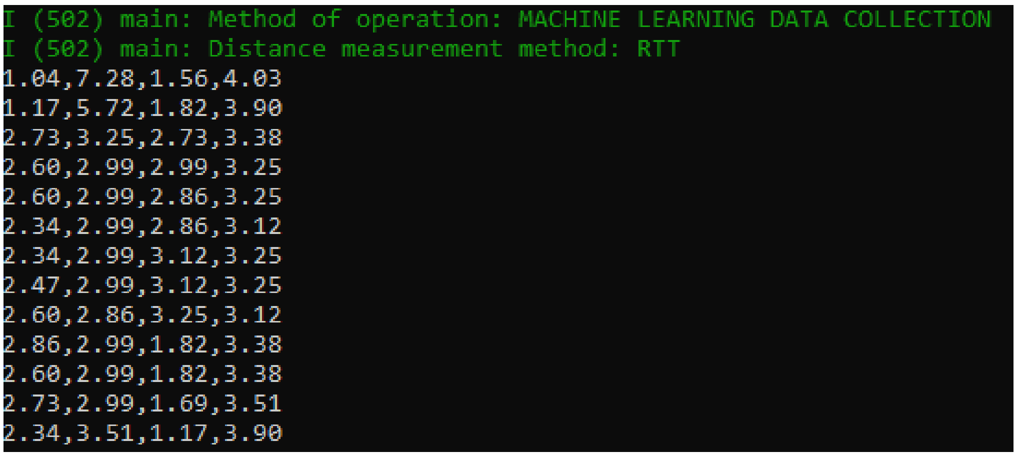
- Purpose: Periodic data collection from monitored APs.
- Procedure:
    1) Resets previous measurements.

2) Collects distance data of all APs in the list.
3) Constructs a CSV-compatible line for measurements.
4) Dispatches the data via UART.
5) Delays the process to maintain a consistent interval between measurements.

```
enum method_of_operation_e {
    OP_POSITIONING = 0,
    OP_ML_DATA_COLLECTION,
    OP_ML_POSITIONING
};

/* Select method of operation here */
static const method_of_operation_e METHOD_OF_OPERATION = OP_ML_DATA_COLLECTION;
```

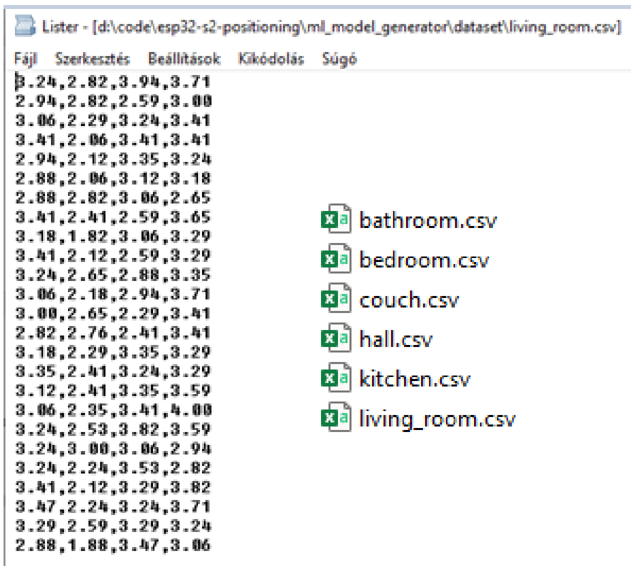
In the developed system, operational modes play a pivotal role in dictating the behavior and response of the system. To facilitate this, an enumerated data type, *method_of_operation_e*, was introduced. This enumeration encapsulates three distinct operational modalities:

OP_POSITIONING, which signifies a rudimentary positioning mechanism; *OP_ML_DATA_COLLECTION*, intended for the gathering of data to facilitate machine learning processes; and *OP_ML_POSITIONING*, which indicates a mode where positioning is achieved through machine learning algorithms. As part of the system's design and for the purposes of the experiments documented in this thesis, the default mode was judiciously set to *OP_ML_DATA_COLLECTION*, emphasizing the system's capability to autonomously collect data pertinent to machine learning tasks.



```
I (502) main: Method of operation: MACHINE LEARNING DATA COLLECTION
I (502) main: Distance measurement method: RTT
1.04,7.28,1.56,4.03
1.17,5.72,1.82,3.90
2.73,3.25,2.73,3.38
2.60,2.99,2.99,3.25
2.60,2.99,2.86,3.25
2.34,2.99,2.86,3.12
2.34,2.99,3.12,3.25
2.47,2.99,3.12,3.25
2.60,2.86,3.25,3.12
2.86,2.99,1.82,3.38
2.60,2.99,1.82,3.38
2.73,2.99,1.69,3.51
2.34,3.51,1.17,3.90
```

Figure 15 - Log output of a ML model train values using RTT method.



```
3.24,2.82,3.94,3.71
2.94,2.82,2.59,3.00
3.06,2.29,3.24,3.41
3.41,2.06,3.41,3.41
2.94,2.12,3.35,3.24
2.88,2.06,3.12,3.18
2.88,2.82,3.06,2.65
3.41,2.41,2.59,3.65
3.18,1.82,3.06,3.29
3.41,2.12,2.59,3.29
3.24,2.65,2.88,3.35
3.06,2.18,2.94,3.71
3.00,2.65,2.29,3.41
2.82,2.76,2.41,3.41
3.18,2.29,3.35,3.29
3.35,2.41,3.24,3.29
3.12,2.41,3.35,3.59
3.06,2.35,3.41,4.00
3.24,2.53,3.82,3.59
3.24,3.00,3.06,2.94
3.24,2.24,3.53,2.82
3.41,2.12,3.29,3.82
3.47,2.24,3.24,3.71
3.29,2.59,3.29,3.24
2.88,1.88,3.47,3.06
```

Figure 16 - ML Model Train measurements

4.5.3 ML Position Prediction

In our endeavour to realize an indoor positioning system, merely collecting data from access points (APs) is insufficient. Transforming this raw data into valuable positional information through the lens of machine learning is the next pivotal step. This chapter elaborates on the implementation of our machine learning classifier which predicts positions using the ESP32-S2 platform, based on the proximity data of APs.

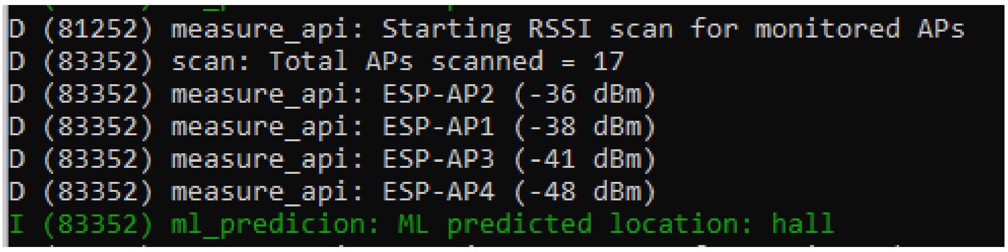
We continue our journey with the ESP32-S2 platform, leveraging its capabilities to not just collect data but also perform in-device predictions. This in-device computation eliminates the need for an external server, thereby promoting efficiency and faster response times.

4.6.2 Code Overview & Explanation

```
Eloquent::ML::Port::RandomForest classifier;

void ml_predict_position() {
    /* Clear the previous measurement */
    monitored_ap_list_clear();
    /* Perform a measurement with the configured method */
    monitored_ap_list_measure();

    /* If none of the configured APs are in range */
    if(!monitored_ap_list_check_any_in_range()) {
        ESP_LOGE(TAG, "All APs are out of range, position cannot be determined");
        return;
    }

    /* Create a float array for the measurement results */
    float measured_data[NUMBER_OF_MONITORED_APS];
    /* Copy the measurement result into the float array */
    for(int i = 0; i < NUMBER_OF_MONITORED_APS; i++) {
        measured_data[i] = (float)monitored_ap_list[i].distance_in_meters;
    }

    /* Call the ML model with the measurement results */
    ESP_LOGI(TAG, "ML predicted location: %s", classifier.predictLabel(measured_data));

    /* Wait before the next measurement */
    vTaskDelay(1000 / portTICK_PERIOD_MS);
}
```

Function: *ml_predict_position()*

- Purpose: To utilize the trained machine learning model for predicting the position based on monitored APs' data.
- Procedure:
    1) Resets previous measurements.
    2) Captures fresh proximity data from APs.
    3) Validates the availability of APs in range. If not found, logs an error, and exits the function.
    4) Converts the captured measurements into a float array for compatibility with the machine learning model.
    5) The machine learning model, represented by the classifier, takes this array as input, and predicts the position, which is then logged for observation.
    6) Introduces a delay for task synchronization and to maintain a steady interval between consecutive predictions.



```
D (81252) measure_api: Starting RSSI scan for monitored APs
D (83352) scan: Total APs scanned = 17
D (83352) measure_api: ESP-AP2 (-36 dBm)
D (83352) measure_api: ESP-AP1 (-38 dBm)
D (83352) measure_api: ESP-AP3 (-41 dBm)
D (83352) measure_api: ESP-AP4 (-48 dBm)
I (83352) ml_predicion: ML predicted location: hall
```

Figure 17 - Output of ML prediction.

Chapter 5 - Experiments and Results

5.1 Measurement setup

The following measurements were performed:

- RSSI and RTT measurement at 1m distance in line of sight
- RSSI and RTT measurement at 3m distance in line of sight
- RSSI and RTT measurement at 1m distance with a 35cm wide brick wall between
- RSSI and RTT measurement at 3m distance with a 35cm wide brick wall between

The measured AP was always in vertical level with the measuring device. During the measurement 7-10 other APs were in range, some of them broadcasting on the same channel as the measurement. The distance between the AP and the measuring device was verified with a tape measure. In each scenario 10 measurements were taken.

5.1.1 Measurement results

RSSI at 1m, line of sight (in dBm):

-42	-40	-42	-42	-42	-43	-41	-42	-43	-48

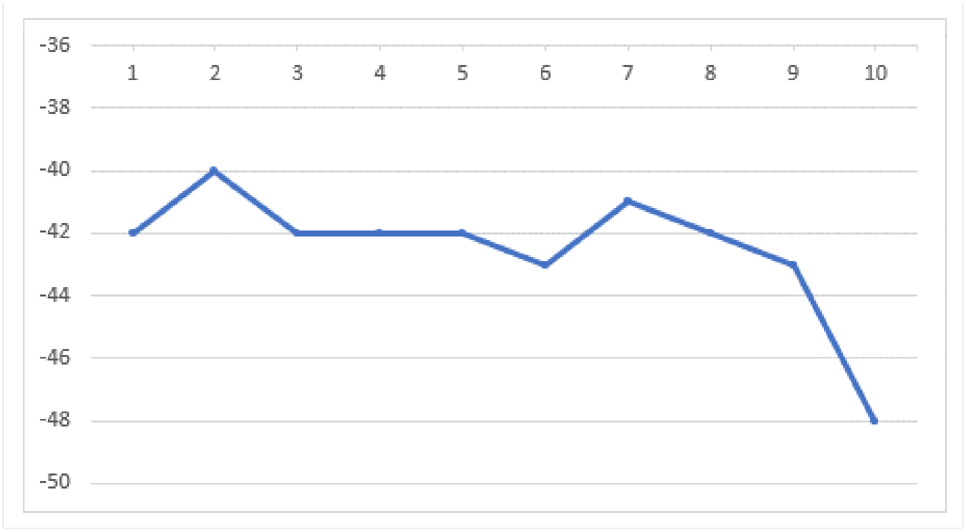


Figure 18 - Results for RSSI at 1m LoS

Average: -42.5 dBm

RSSI at 3m, line of sight (in dBm):

-54	-52	-52	-53	-54	-55	-55	-55	-61	-53

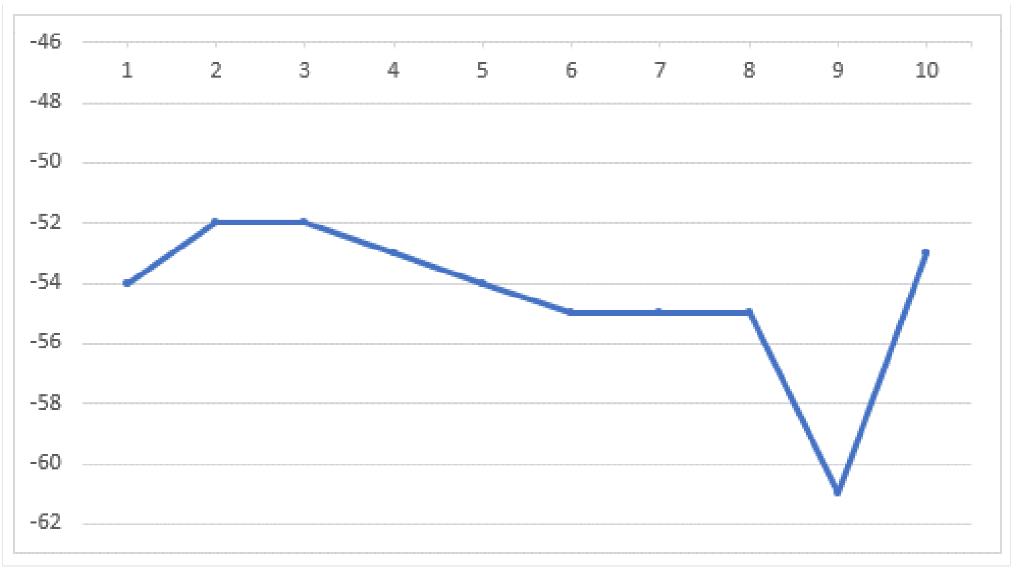


Figure 19 - Results for RSSI at 3m LoS

Average: -54.4 dBm

RSSI at 1m, 35cm wide brick wall between (in dBm):

-49	-48	-48	-49	-49	-49	-50	-51	-50	-48

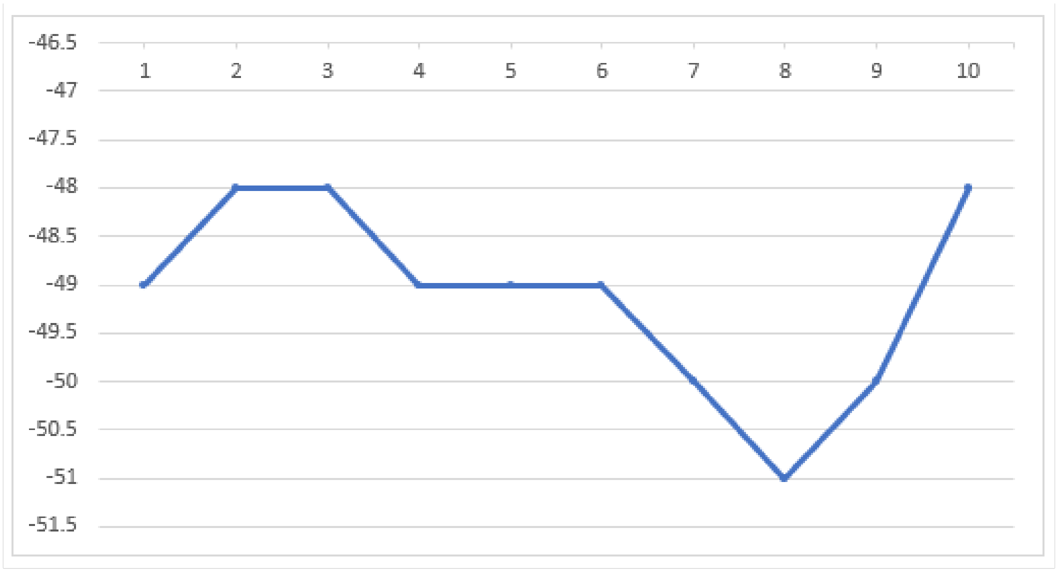


Figure 20 - Results for RSSI at 1m with wide brick wall

Average: -49.1 dBm

RSSI at 3m, 35cm wide brick wall between (in dBm):

-62	-63	-69	-62	-64	-64	-64	-63	-65	-64

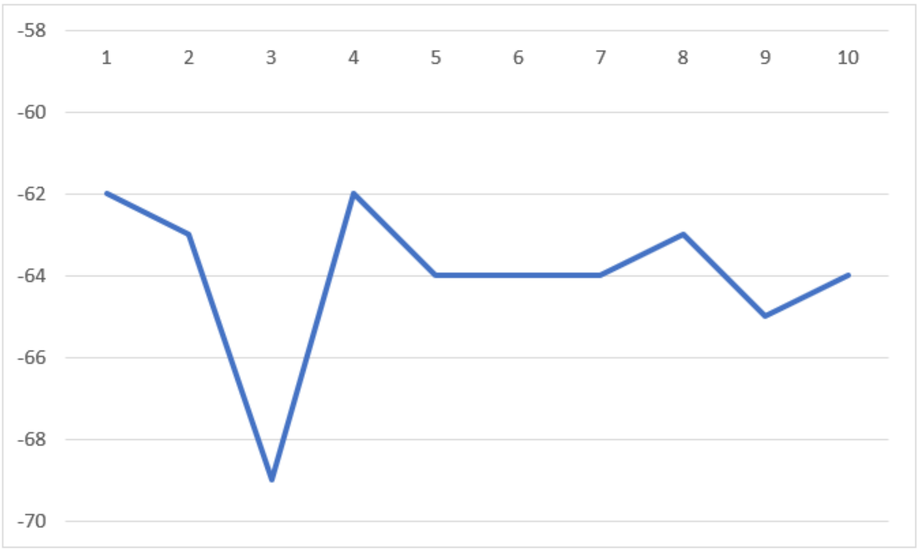


Figure 21 - Results for RSSI at 3m with wide brick wall

Average: -64.0 dBm

RTT at 1m, line of sight (in meters):

0.78	0.91	1.17	1.04	1.17	1.17	1.04	1.17	1.17	1.17

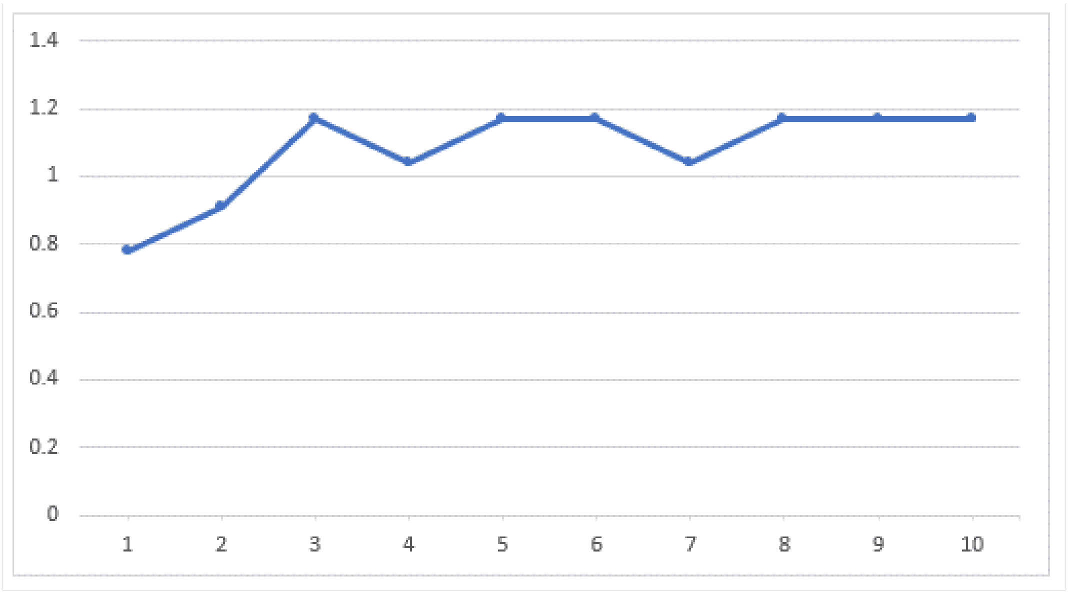


Figure 22 - Results for RTT at 1m LoS

Average: 1.079m

RTT at 3m, line of sight (in meters):

4.33	4.20	4.98	4.33	4.11	4.29	4.03	3.77	3.77	4.03

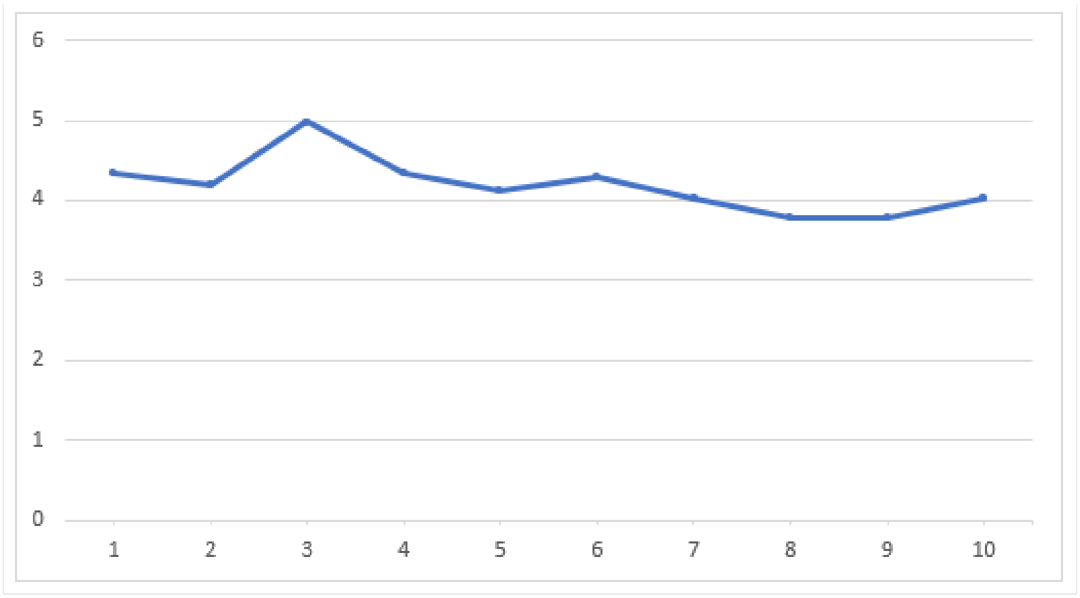


Figure 23 - Results for RTT at 3m LoS

Average: 4.184m

RTT at 1m, 35cm wide brick wall between (in meters):

4.29	4.03	3.64	3.12	3.12	3.38	3.12	4.03	4.29	3.77

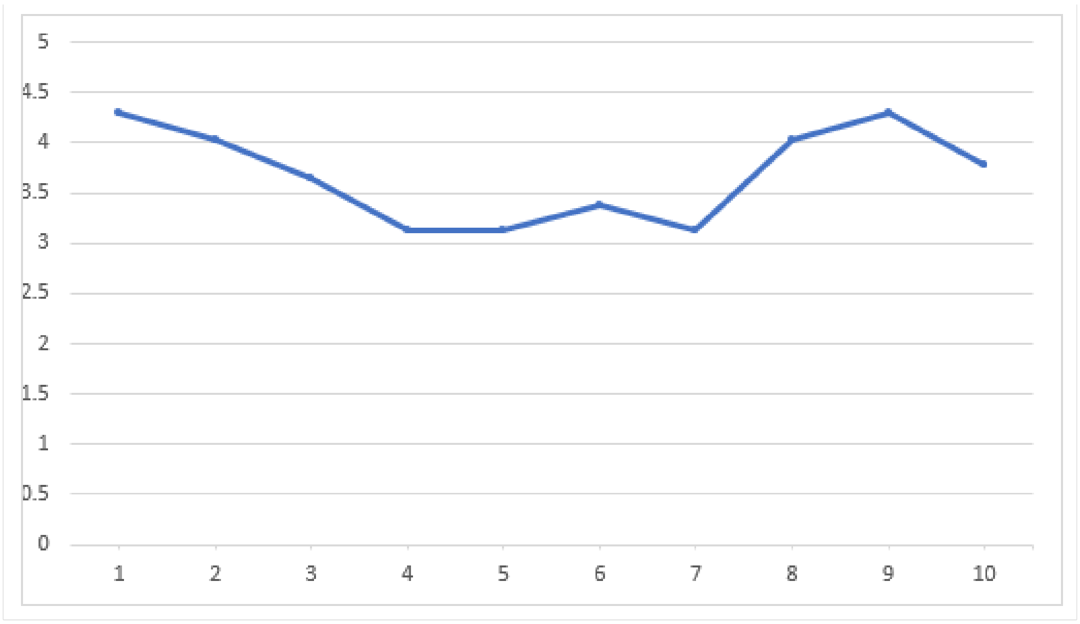


Figure 24 - Results for RTT at 1m with wide brick wall

Average: 3.679m

RTT at 3m, 35cm wide brick wall between (in meters):

5.33	5.20	5.20	4.94	5.33	5.85	6.11	5.59	5.85	5.46

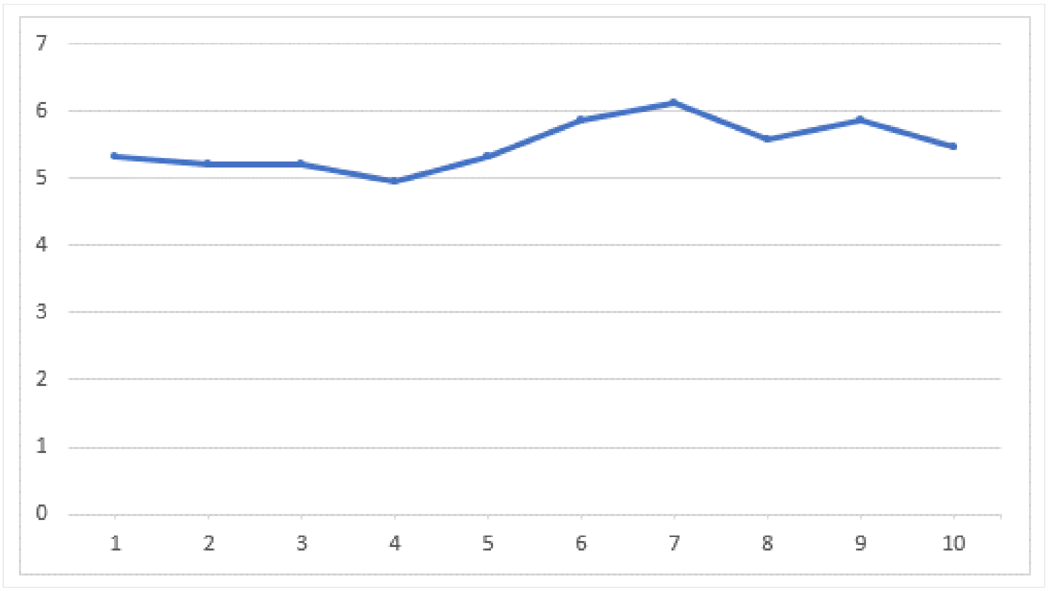


Figure 25 - Results for RTT at 3m with wide brick wall

Average: 5.486m

If we take the 1m line of sight RSSI measurement as the baseline for one meter, we can calculate the other RSSI results in meters:

- 1m = -42.5 dBm
- -1dBm = 0.0235m

The following table contains all the measurement result averages in meters:

	1m, line of sight	**3m, line of sight**	**1m, wall**	**3m, wall**
RSSI	1m (baseline)	1.28m	1.15m	1.51m
RTT	1.079m	4.184m	3.679m	5.486m

Table 1 - Averages of measurement results in meters

The following table contains the calculated measurement accuracy values:

	1m, line of sight	**3m, line of sight**	**1m, wall**	**3m, wall**
RSSI	0m (baseline)	1.72m	0.15m	1.49m
RTT	0.079m	1.184m	2.679m	2.486m

Table 2 - Calculated measurement values

5.2 Comparison of accuracy (line of sight)

The RTT method was much more accurate without the need for any baseline values. These results could be reinforced by measuring them outside over bigger distances and with less interfering APs.

5.2.1 Comparison of accuracy (with obstructions)

The RSSI method yielded better results - the RTT is much more sensitive to obstructions.

5.2.2 Other comparison factors

As for speed - the RSSI method has the advantage, RTT measurements on the ESP32-S2 take a much longer time. RSSI measurements can be done with any WiFi capable device, RTT requires specific hardware (with 802.11mc support). The changes in RSSI values over distance are not linear, but close to it - this measurement calculates RSSI changes on a linear scale.

5.3 Conclusion

The RTT method is better for accurate measurements, however the RSSI is better for speed.

5.4 Machine Learning measurements

5.4.1 Measurement setup

The moving device is trained with an ML model consisting of three rooms. Two different ML models were created for the same setup - one with 5 data lines for each room, and one with 15 data lines. We repeated this with both RSSI and RTT models.

We took 10 measurements while standing in a room and at the edge of the room. We took note of the correctness of the predicted location.

Living room - RSSI - model with 5 datapoints:

correct	correct	false	correct	correct	false	false	correct	false	false

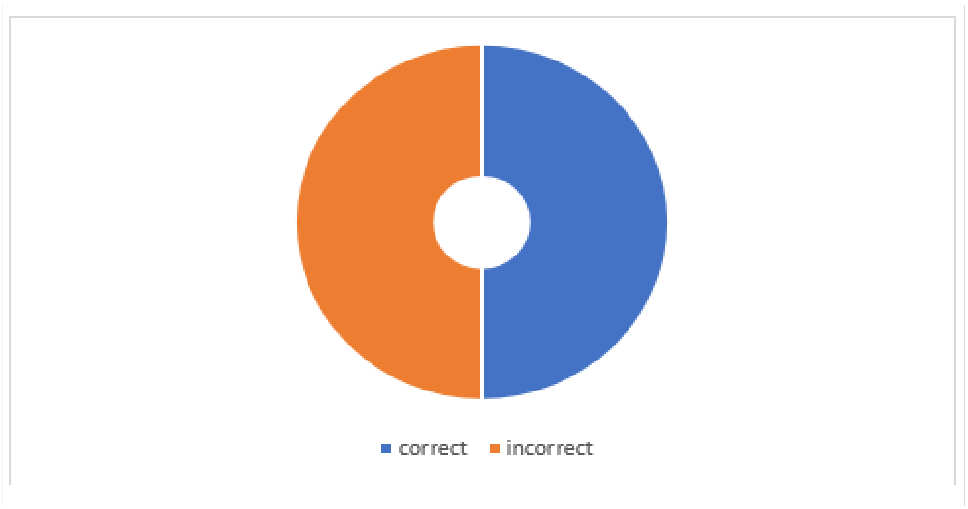


Figure 26 - Living room - RSSI - model with 5 datapoints.

Correct ratio: 5/10 (50%)

Living room - RTT - model with 5 datapoints:

false	correct	correct	false	false	false	false	correct	correct	correct

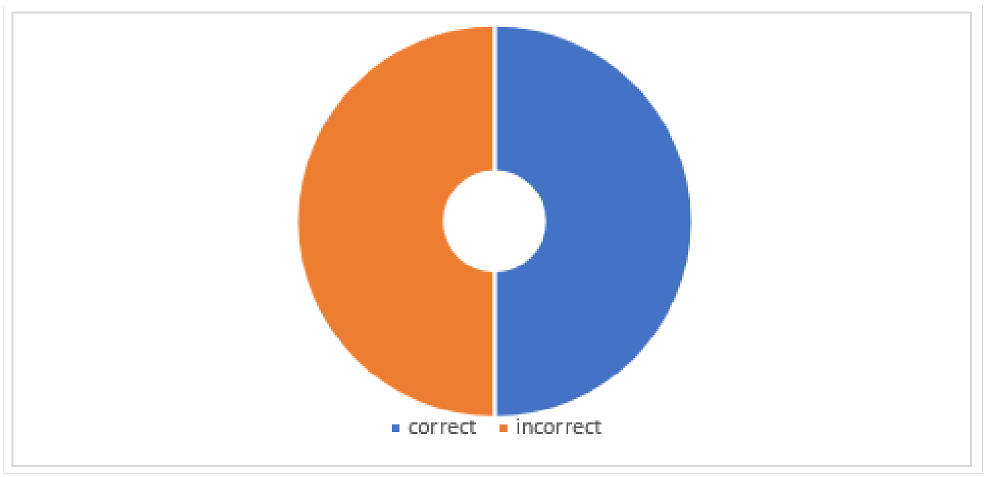


Figure 27 - Living room - RTT - model with 5 datapoints.

Correct ratio: 5/10 (50%)

Living room - RSSI - model with 15 datapoints:

correct	correct	correct	correct	correct	false	false	correct	correct	correct

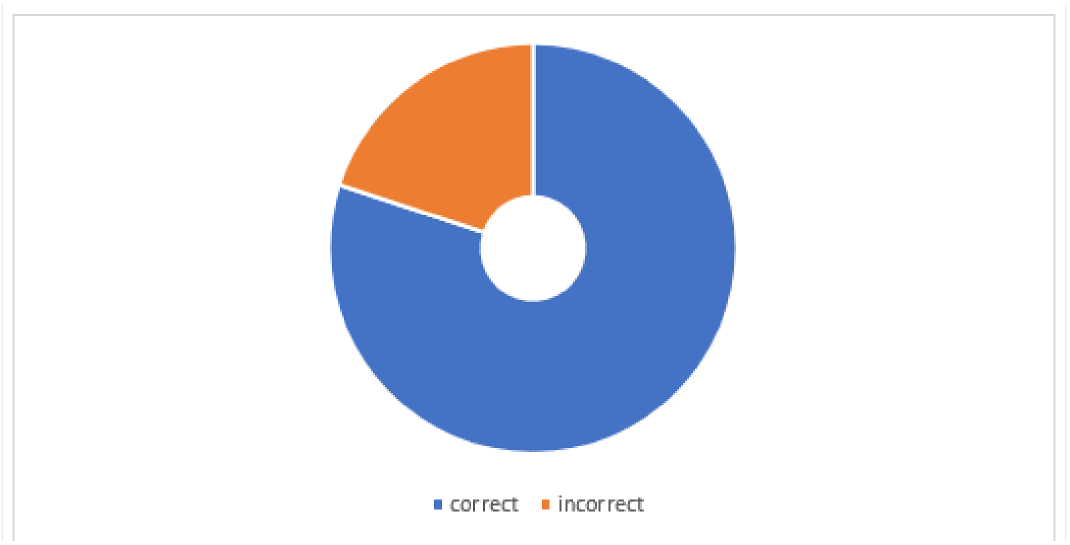


Figure 28 - Living room - RSSI - model with 15 datapoints.

Correct ratio: 8/10 (80%)

Living room - RTT - model with 15 datapoints:

correct	correct	false	correct	correct	correct	correct	correct	correct	correct

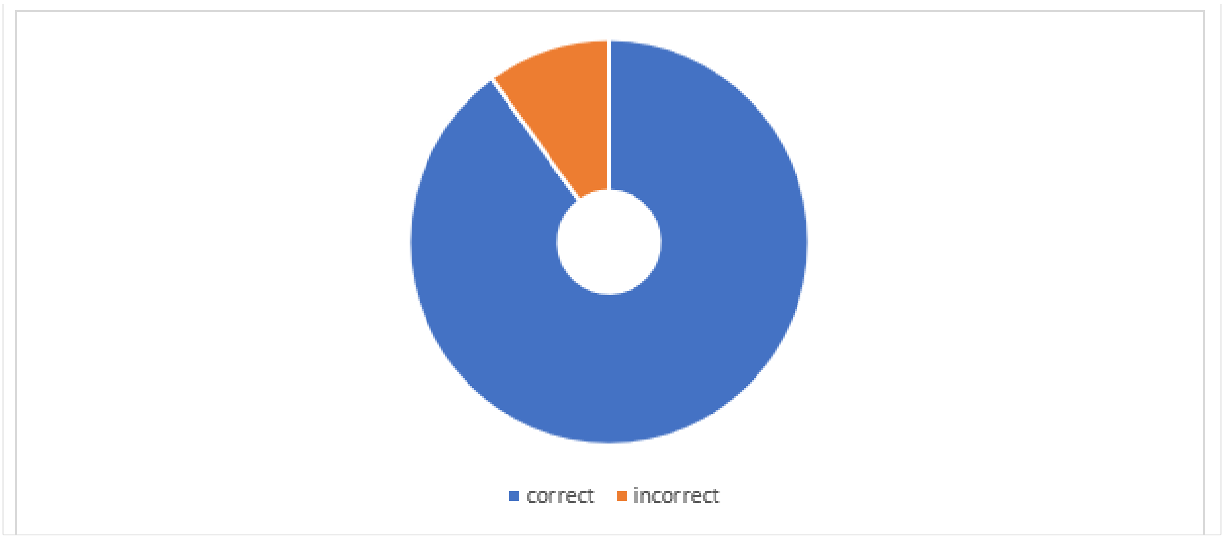


Figure 29 - Living room - RTT - model with 15 datapoints.

Correct ratio: 9/10 (90%)

Conclusion

The ML model works the best with as many datapoints as possible. The accuracy difference between the RSSI and RTT method is negligible.

5.5 Machine-Learning Classifiers for Positioning Accuracy Measurements

To evaluate the positioning accuracy, four distinct machine-learning classifiers were employed. The classifiers and their fundamental principles are delineated below:

5.5.1 Random Forest (RF):

The Random Forest algorithm embodies an ensemble learning technique, predicated on the formulation of a multitude of decision trees throughout the training phase. The resultant prediction emerges from an aggregation of the determinations proffered by each individual tree. A salient virtue of this algorithm is its intrinsic capacity to mitigate overfitting through its ensemble stratagem, concurrently bestowing a significance score upon each constituent feature.

5.5.2 Support Vector Classifier (SVC):

The Support Vector Classifier (SVC), colloquially known as the Support Vector Machine (SVM) in classification contexts, functions by discerning the optimal hyperplane that delineates classes within the feature space. Through the employment of kernel methodologies, SVC possesses the capability to accommodate non-linear data structures. It achieves this by transposing the data into an augmented dimensional space, therein facilitating the identification of an appropriate separating hyperplane.

5.5.3 Gaussian Naïve Bayes (NB):

Gaussian Naïve Bayes is a probabilistic classifier rooted in Bayes' theorem with an assumption of independence between features. While the term "naïve" stems from this independence assumption, the classifier often performs surprisingly well in practice. For continuous data, a Gaussian distribution is assumed, hence the name.

5.5.4 Linear Regression:

Linear Regression, conventionally requisitioned for regression endeavors, can be ingeniously repurposed for classification tasks, particularly when outcomes are binary in nature. This technique postulates and models the relationship between a designated dependent variable and its associated independent variables through the alignment of a linear equation. Within the ambit of classification, a definitive threshold is established. Data points surpassing this demarcation are ascribed to one categorical class, whilst those falling below are allocated to an alternative category.

5.6 Experimental Setup

5.6.1 Access Point Placement

Four Access Points (APs) were strategically installed in each of the primary rooms: Living Room, Bathroom, Bedroom, and Kitchen.

5.6.2 Areas of Study

In total, six distinct areas were identified for the experiment. These comprised the four primary rooms with APs, and two areas without any APs - the Hall and the Couch area.

5.6.3 RSSI Data Collection

RSSI, or Received Signal Strength Indicator, was employed as the primary measurement methodology. For each of the six areas, 15 unique measurements were performed to capture the variations in signal strength.

5.6.4 Training Phase and Deployment

Based on the collected RSSI data, individual machine learning models were trained for each classifier using a dedicated Python script.

Once trained, each model was flashed onto a moving device for on-ground testing and evaluation.

5.6.5 Walking Route and Data Collection

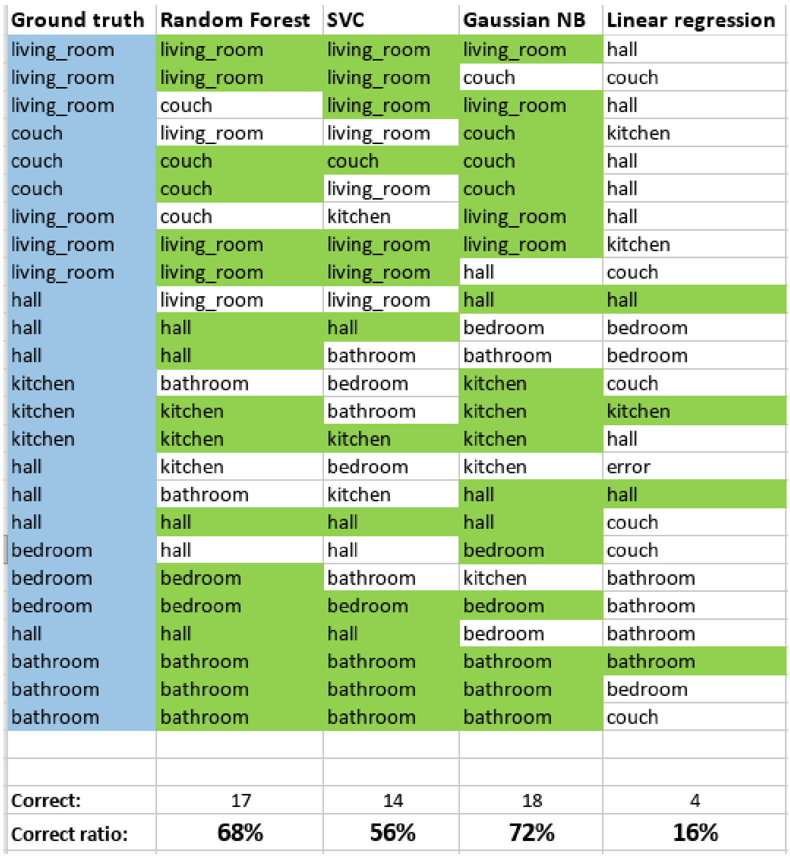
To evaluate the performance of the flashed models, a specific route was walked multiple times, collecting predicted location samples at each key point. The route followed was:

Living Room (3 samples) → Couch (3 samples) → Living Room (3 samples) → Hall (3 samples) → Kitchen (3 samples) → Hall (3 samples) → Bedroom (3 samples) → Hall (1 sample) → Bathroom (3 samples)

The numbers in parentheses indicate the number of location samples taken at each respective area.

5.6.6 Data Analysis

In total, 25 location prediction samples were captured for each classifier, aggregating to 100 samples across all classifiers. To ascertain the reliability and accuracy of the models, these predicted samples were juxtaposed against the ground truth. The ensuing discrepancies or matches formed the basis for determining the final performance metrics for each classifier.



Ground truth	Random Forest	SVC	Gaussian NB	Linear regression
living_room	living_room	living_room	living_room	hall
living_room	living_room	living_room	couch	couch
living_room	couch	living_room	living_room	hall
couch	living_room	living_room	couch	kitchen
couch	couch	couch	couch	hall
couch	couch	living_room	couch	hall
living_room	couch	kitchen	living_room	hall
living_room	living_room	living_room	living_room	kitchen
living_room	living_room	living_room	hall	couch
hall	living_room	living_room	hall	hall
hall	hall	hall	bedroom	bedroom
hall	hall	bathroom	bathroom	bedroom
kitchen	bathroom	bedroom	kitchen	couch
kitchen	kitchen	bathroom	kitchen	kitchen
kitchen	kitchen	kitchen	kitchen	hall
hall	kitchen	bedroom	kitchen	error
hall	bathroom	kitchen	hall	hall
hall	hall	hall	hall	couch
bedroom	hall	hall	bedroom	couch
bedroom	bedroom	bathroom	kitchen	bathroom
bedroom	bedroom	bedroom	bedroom	bathroom
hall	hall	hall	bedroom	bathroom
bathroom	bathroom	bathroom	bathroom	bathroom
bathroom	bathroom	bathroom	bathroom	bedroom
bathroom	bathroom	bathroom	bathroom	couch
Correct:	17	14	18	4
Correct ratio:	**68%**	**56%**	**72%**	**16%**

Figure 30 - Comparison of machine learning classifiers

5.7 Results of comparison

From the evaluation of the classifiers based on the collected data, distinct performance differences emerged:

5.7.1 Gaussian Naïve Bayes (NB)

The Gaussian NB classifier emerged as the foremost performer in the comparison. It demonstrated high accuracy, consistently predicting locations with remarkable precision.

5.7.2 Random Forest (RF)

Following closely was the Random Forest classifier, which also showcased commendable accuracy in most of the measurements, solidifying its robustness in the domain of positioning systems.

5.7.3 Support Vector Classifier (SVC)

The SVC classifier occupied the third spot, providing accurate predictions in just over half of the measurements. While its performance was satisfactory, it lagged the Gaussian NB and Random Forest in terms of consistent accuracy.

5.7.4 Linear Regression

The Linear Regression classifier was the least impressive among the four. Its predictions were predominantly erroneous, accurately determining locations only 16% of the time. Such a performance indicates potential issues with the model's adaptability to this specific problem space or might highlight the necessity for a more feature-rich dataset.

5.7.5 Considerations for Future Improvements

The observed performance, especially of the lower-ranking classifiers, suggests that enhanced training data could ameliorate the models' accuracy. Further research and experimentation with a larger or more varied dataset might enable these classifiers to provide better predictions, bridging the gap between their current output and the ground truth.

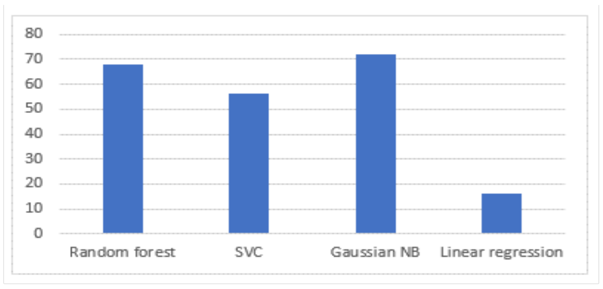


Figure 31 - Accuracy percentage of each classifier

Chapter 6 - Future Work

The first prototype method was built to process raw RSSI data. However, as time progressed, it became clear that we had to first filter and map the values to a much lower number to smooth them out. To further pinpoint the monitored entity's whereabouts, we had originally planned to use an algorithm using triangulation formulae.

Nonetheless, we were able to depict the graph using a kind of shortest route approach, often known as dead reckoning, by making use of the substantially mapped and smoothed variables. This method allows us to pinpoint the location of the beacon's closest available access point. Second, depending on how far away it is from the beacon, we find the subsequent closest access point, and then we plot the location.

As we went along, the prototype underwent significant changes that gave it its final form. However, we don't have the resources to make it even more reliable, accurate, and precise. Although the prototype exhibits accuracy and precision in good working order, there is still much room for improvement.

6.1 Using RFID

RFID (radio-recurrence distinguishing proof), a developing positioning innovation that considers versatility following of items or people, uses radio frequencies to remotely transmit an item's identity (such as its serial number) and other attributes. RFID isn't suitable for complete confinement because it only provides a limited range of about a meter, but rather for a specific article distinguishing proof. It is useful, easy to maintain, and provides both area and distinguishing proof.

This makes RFID-based restriction particularly suitable for following arrangements under mechanical conditions (for example resource the executives). Due to its extremely constrained range of less than a meter, RFID technology does not enable region-wide confinement like Wi-Fi, WIFI Low Energy reference points, or Ultra-wideband (UWB) does. Instead, it only facilitates local confinement.

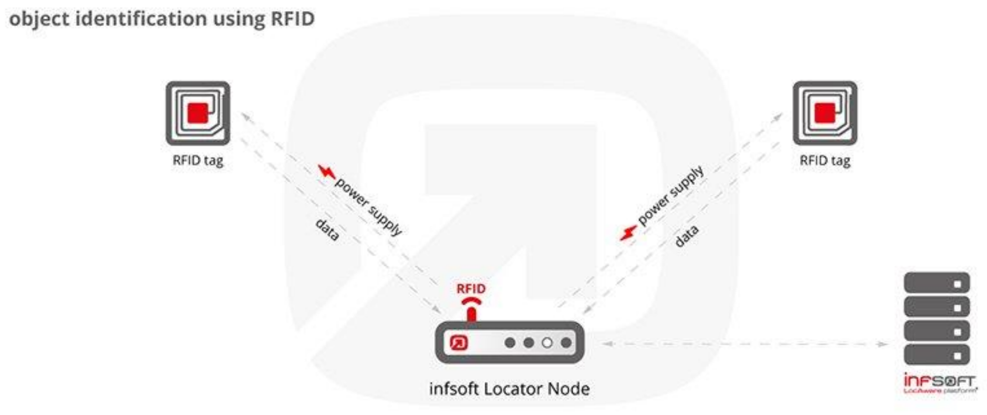


Figure 32 - RFID based IPS by Insoft

In a typical RFID-based indoor positioning system, such as the infsoft Locator Node, the foundational components comprise a reader and transponders, which can be strategically affixed to individuals or objects. In passive RFID configurations (termed as remote-coupling), the Locator Node acts as the primary energy source, disseminating radio-frequency energy to proximate transponders. Upon the acquisition of identification and ancillary data from the transponder, the Locator Node relays this information to the infsoft LocAware platform® for subsequent processing.

6.2 Using Wi-Fi

There have been several recent proposals for WiFi-, WIFI-, and UWB-based indoor limitation systems. An accurate situating framework may still be developed at a reasonable cost because to the confinement and intricacy of the interior environment. This article presents a WiFi-based positioning strategy that can improve confinement performance by removing the ToA/AoA bottleneck. Unlike conventional methodologies, our proposed component uses the transmission of many predefined messages while maintaining high-accuracy execution, thereby reducing the need for high sign transfer speed and large numbers of radio wires. By demonstrating confinement execution for various message quantities used in WiFi APs with 20/40 MHz transfer speeds, the general architecture of the system is demonstrated. The results of our replication show that our WiFi-based positioning method is considerably better to the typical methods from both academia and industry in terms of the trade-off between cost and framework complexity, achieving accuracy to within 1 m without forcing businesses to upgrade their WiFi equipment.

There are a lot of reasons why Wi-Fi based indoor confinement systems have become a popular tool for indoor placement. To begin, a Wi-Fi chip is already implicitly included in almost every modern mobile phone. Second, an indoor limitation framework that uses Wi-Fi is not only cheap but also simple to implement. The use of Wi-Fi hotspots has become commonplace. Third, Wi-Fi requires no specialized add-ons. The received signal strength (RSS) measurements from a Wi-Fi tunnel may be used as a reliable indicator of the quality of an area. Finally, Wi-Fi frameworks' transmission capacities have risen greatly to accommodate the needs of large information rates.

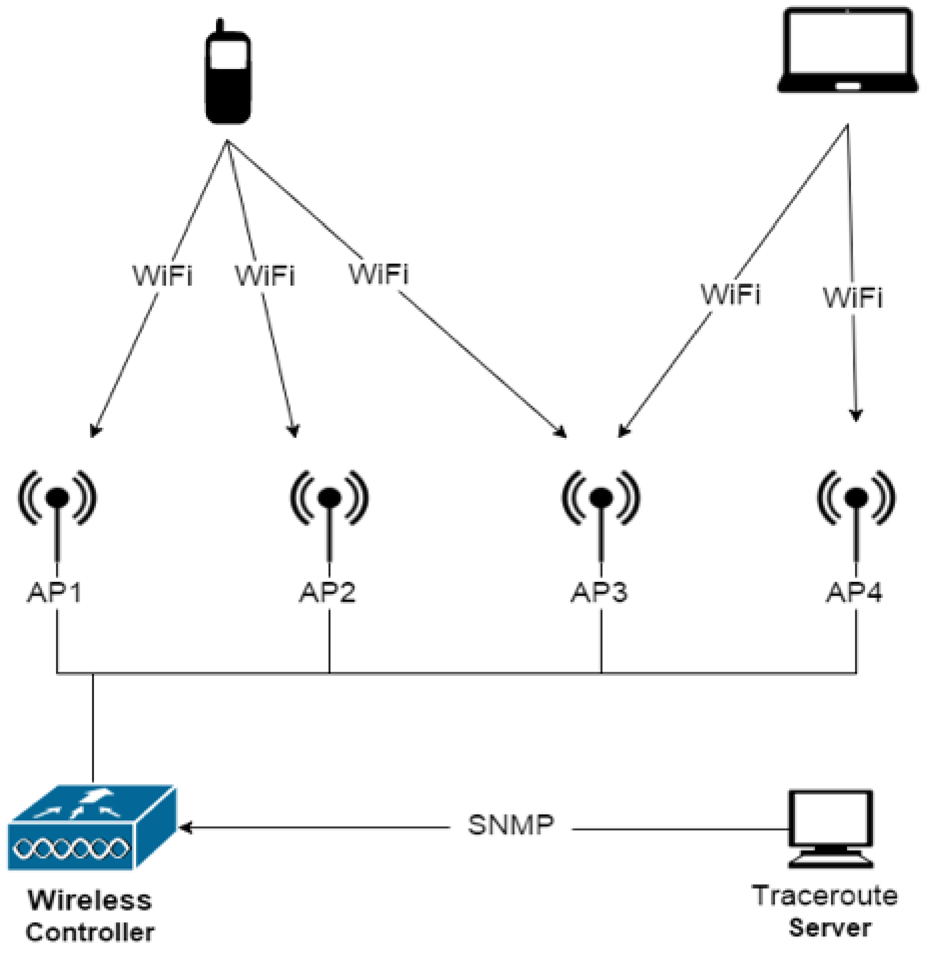


Figure 33 - WiFi Based IPS

However, there are two distinct classes of Wi-Fi indoor situating techniques: area fingerprinting and signal proliferation models. Section 1 introduces the connection between sign spread models and the area fingerprinting technique. On a few pathways, multipath distortion is encountered when attempting to calculate the route to the station from handheld devices using an indoor positioning system based on the hour of appearance (ToA) and time contrast of appearance (TDoA). In addition to using the porta client's data, the point of appearance (AoA) and point of flight (AoF) techniques may be employed to determine the significance of an area.

Angle-of-Arrival (AoA) predicated systems predominantly ascertain positional data by evaluating the distances between signals converging at disparate anchor nodes. Consequently, both the port client and Wi-Fi gateway necessitate cognizance of the orientation of the receiving antennae. The Round-Trip Time of Flight (RToF) metric quantifies the interval required for a signal to traverse from its origin to its destination and reciprocally return. Implementing such mechanisms demands meticulous planning and extended durations. The inaugural phase in geospatial fingerprinting encompasses the compilation of a dataset, capturing estimates of signal strengths at myriad reference points within a wireless LAN coverage domain.

Area fingerprinting-based indoor positioning systems compare the reference data and remote sign estimations. In any case, this technique needs an old and stable data set. Executing signal engendering is simpler than area fingerprinting. However, multipath spread and signal engendering are still very difficult to understand because of things like entrance mistakes through walls and floors. To achieve a superior presentation, a novel Wi-Fi-based indoor situating framework was proposed.

6.3 Multiple Technologies’ Beacons or Hybrid Tech

Trustworthy indoor arrangement is a critical foundation for emerging indoor region-based organizations. Most existing indoor arranging suggestions rely upon a lone distant development, e.g., Wi-Fi, WIFI, or RFID. A combination arranging system unites such headways and achieves better arranging accuracy by exploiting the different limits of the different developments. In a hybrid structure reliant upon Wi-Fi and WIFI, the past capacities as the essential establishment to remarks finger impression-based arranging, while the last referenced (through space of interest devices) portions the indoor space similarly as a tremendous Wi-Fi radio aide. In this way, the Wi-Fi based online position evaluation is dealt with in a hole and-defeat way.

We focus on three pieces of a special creamer indoor arranging system. In any case, to try not to huge position botches achieved by relative reference puts that are hard to remember, we plan a sending estimation that recognizes and detaches such circumstances into different more unassuming radio aides by passing on WIFI spaces of interest at explicit positions. Second, we plan procedures that further foster the section trading that happens when a customer leaves the disclosure extent of a WIFI space of interest. Third, we propose three structure options for circumstance of the estimation obligation.

We evaluate all proposals using both entertainment and walkthrough tests in two indoor states of different sizes. The results show that our recommendations are suite and successful in achieving commonly astounding indoor arranging execution.

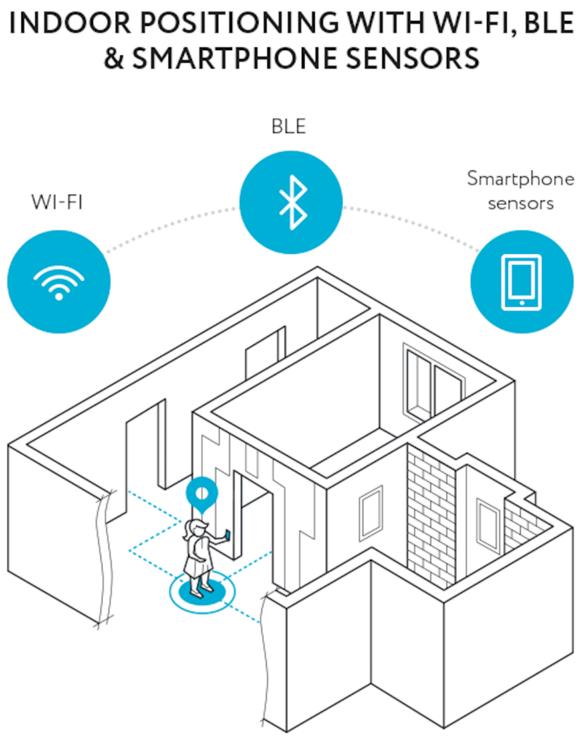


Figure 34 - Hybrid IPS

On the other hand, there are two categories for Wi-Fi indoor situating procedures: area fingerprinting and signal proliferation models. Ta 1 introduces an investigation of sign proliferation models and the area fingerprinting approach.

Within an indoor positioning paradigm predicated on the Time of Arrival (ToA) and Time Difference of Arrival (TDoA) of a signal propagation model, one encounters multipath distortions when gauging distances between mobile devices and the primary station, manifesting on several trajectories [5]. Contrastingly, by harnessing signal positions accrued from the mobile user and the Wi-Fi conduit, Angle of Arrival (AoA) and Angle of Departure (AoD) methodologies can be employed to discern the spatial extent. It is noteworthy that a significant proportion of AoA-oriented systems mandate a comprehensive understanding of the antenna orientations for both the porta client and the Wi-Fi gateway. This knowledge serves as a bedrock for approximating positions via juxtaposition of distances originating from diverse anchor nodes.

The full circle season of flight (RToF), which measures how long it takes for a signal to go from sender to client and back, is used to describe an area. However, some planning and time management are needed. A database including estimations of distant signals at different reference foci in a remote LAN inclusion zone must be built before conducting area fingerprinting.

Area fingerprinting-based indoor positioning systems compare the reference data and remote sign estimations. Nevertheless, this technique needs a strong information base. In contrast to area fingerprinting, signal engendering is a simple process. However, signal engendering and multipath proliferation are still very complicated due to things like entrance mistakes through dividers and floors. To achieve a better presentation, a clever Wi-Fi-based indoor situating framework was suggested.

Chapter 7 - References

[1] Abowd, Gregory D.; Brumitt, Barry; Shafer, Steven (2001). [Lecture Notes in Computer Science] Ubicomp 2001: Ubiquitous Computing Volume 2201 || Low Cost Indoor Positioning System. ,

[2] A. Bekkali, H. Sanson, and M. Matsumoto, “RFID indoor positioning based on probabilistic RFID map and Kalman filtering,” in Proc.3rd IEEE Int. Conf. Wireless Mobile Comput., Netw. Commun., 2007, pp. 21–28. [3] H. Hua and B. Javidi, “A 3D integral imaging optical see-through head-mounted display,” *Opt. Express*, vol. 22, no. 11, p. 13484, 2014.

[3] Y.-H. Chen, S.-J. Horng, R.-S. Run, J.-L. Lai, R.-J. Chen, W.-C. Chen, Y. Pan, and T. Takao, “A novel anti-collision algorithm in RFID systems for identifying passive tags,” IEEE Trans. Ind. Informat., vol. 6, no. 1, pp. 105–121, Feb. 2010.

[4] S. S. Saab, W. Mhanna, and S. Saliba, “Conceptualisation study of using RFID as a stand-alone vehicle positioning system,” Int. J. Radio Freq. Identification Technol. Appl., vol. 2, no. 1/2, pp. 27–45, Feb. 2009

[5] Ç. Genç, S. Soomro, Y. Duyan, S. Ölçer, F. Balcı, H. Ürey, and O. Özcan, “Head Mounted Projection Display &amp; Visual Attention: Visual Attentional Processing of Head Referenced Static and Dynamic Displays while in Motion and Standing,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI ’16*, 2016, pp. 1538–1547.

[6] Saad, S.S.; Nakad, Z.S. (2011). A Standalone RFID Indoor Positioning System Using Passive Tags.

[7] Fukuju, Y.; Minami, M.; Morikawa, H.; Aoyama, T. (2003). [IEEE Comput. Soc IEEE Workshop on Software Technologies for Future Embedded Systems. WSTFES 2003 - Hokkaido, Japan (15-16 May 2003)] Proceedings IEEE Workshop on Software Technologies for Future Embedded Systems. WSTFES 2003 - DOLPHIN: an autonomous indoor positioning system in ubiquitous computing environment

[8] Yasir, Muhammad; Ho, Siu-Wai; Vellambi, Badri N. (2014). Indoor Positioning System Using Visi Light and Accelerometer. Journal of Lightwave Technology

[9] Y. Gu, A. Lo, and I. Niemegeers, “A survey of indoor positioning systems for wireless personal networks,” IEEE Commun. Surveys Tuts., vol. 11, no. 1, pp. 13–32, First Quarter 2009.

[10] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of wireless indoor positioning techniques and systems,” IEEE Trans. Syst., Man, Cybern. C, Appl. Rev., vol. 37, no. 6, pp. 1067–1080, Nov. 2007

[11] H. Kim, D. Kim, S. Yang, Y. Son, and S. Han, “An indoor visi light communication positioning system using a RF carrier allocation technique,” J. Lightw. Technol., vol. 31, no. 1, pp. 134–144, Jan. 2013.

[12] C. Sertthin, T. Ohtsuki, and M. Nakagawa, “6-axis sensor assisted low complexity high accuracy-visi light communication based indoor positioning system,” IEICE Trans. Commun., vol. E93-B, no. 11, pp. 2879– 2891, Nov. 2010.

[13] S.-H. Yang, E.-M. Jeong, and S.-K. Han, “Indoor positioning based on received optical power difference by angle of arrival,” IEEE Electron. Lett., vol. 50, no. 1, pp. 49–51, Jan. 2014.

[14] A. Arafa, X. Jin, and R. Klukas, “Wireless indoor optical positioning with a differential photosensor,” IEEE Photon. Technol. Lett., vol. 24, no. 12, pp. 1027–1029, Jun. 2012.

[15] S.-Y. Jung, S. Hann, and C.-S. Park, “TDOA-based optical wireless indoor localization using LED ceiling lamps,” IEEE Trans. Consum. Electron., vol. 57, no. 4, pp. 1592–1597, Nov. 2011.

[16] F. Gfeller and U. Bapst, “Wireless in-house data communication via diffuse infrared radiation,” Proc. IEEE, vol. 67, no. 11, pp. 1474–1486, Nov. 1979.

[17] T. Komine and M. Nakagawa, “Fundamental analysis for visi-light communication system using LED lights,” IEEE Trans. Consum. Electron., vol. 50, no. 1, pp. 100–107, Feb. 2004.

[18] Fang, F., Zhao, S., Guo, P.The range analysis of RSSI-basedChinese Hournal of Sensing Technology Journal2007201125262530

[19] Rappaport, T. S. Wireless Communications: Principles and Practice1996Upper Saddle River, NJ, USAPrentice Hall PTR

[20] Violettas, George E., Tryfon L. Theodorou, and Christos K. Georgiadis. "Netargus: An snmp monitor & wi-fi positioning, 3-tier application suite." *2009 Fifth International Conference on Wireless and Mobile Communications*. IEEE, 2009.

[22] Youssef, Moustafa, and Ashok Agrawala. "The Horus WLAN location determination system." *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. 2005.

[23] Chen, Yongguang, and Hisashi Kobayashi. "Signal strength based indoor geolocation." *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No. 02CH37333)*. Vol. 1. IEEE, 2002.

[24] Golovan, Andrei A., et al. "Efficient localization using different mean offset models in Gaussian processes." *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2014.

[25] Hähnel, Brian Ferris Dirk, and Dieter Fox. "Gaussian processes for signal strength-based location estimation." *Proceeding of robotics: science and systems*. 2006.

[26] Ferris, Brian, Dieter Fox, and Neil D. Lawrence. "Wifi-slam using gaussian process latent variable models." *IJCAI*. Vol. 7. No. 1. 2007.

[27] Lymberopoulos, Dimitrios, et al. "A realistic evaluation and comparison of indoor location technologies: Experiences and lessons learned." *Proceedings of the 14th international conference on information processing in sensor networks*. 2015.

[28] Laoudias, Christos, et al. "The airplace indoor positioning platform for android smartphones." *2012 IEEE 13th International Conference on Mobile Data Management*. IEEE, 2012.

[29] Abbas, Nasir, et al. "Mobile edge computing: A survey." *IEEE Internet of Things Journal* 5.1 (2017): 450-465.

[30] Shalini Lakshmi, A. J., and M. Vijayalakshmi. "CASCM2: Capability-Aware Supply Chain Management Model for QoS-driven offload-participator selection in Fog environments." *Sādhanā* 45.1 (2020): 1-14.

[31] Jararweh, Yaser, et al. "SDMEC: Software defined system for mobile edge computing." *2016 IEEE international conference on cloud engineering workshop (IC2EW)*. IEEE, 2016.

[32] Khanh, Tran Trong, et al. "Wi-Fi indoor positioning and navigation: a cloudlet-based cloud computing approach." *Human-centric Computing and Information Sciences* 10.1 (2020): 1-26.

[33] Liu, Fen, et al. "Survey on WiFi-based indoor positioning techniques." *IET communications* 14.9 (2020): 1372-1383.

[34] Liu, Yanchen, Myung J. Lee, and Yanyan Zheng. "Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system." *IEEE Transactions on Mobile Computing* 15.10 (2015): 2398-2410.

[35] Mendoza-Silva, G. M., et al. "Long-Term Wi-Fi fingerprinting dataset and supporting material." *Zenodo repository* (2017).

[36] Ramasubbareddy, Somula, et al. "Cavms: Application-aware cloudlet adaption and vm selection framework for multicloudlet environment." *IEEE Systems Journal* 15.4 (2020): 5098-5106.

[37] Bai, Yuntian Brian, et al. "A new method for improving Wi-Fi-based indoor positioning accuracy." *Journal of Location Based Services* 8.3 (2014): 135-147.

[38] Bekkelien, A., Deriaz, M. and Marchand-Maillet, S., 2012. Bluetooth indoor positioning. Master's thesis, University of Geneva.

[39] Choi, M.S. and Jang, B., 2017. An accurate fingerprinting based indoor positioning algorithm. International Journal of Applied Engineering Research, 12(1), pp.86-90.

[40] Jekabsons, G., Kairish, V. and Zuravlyov, V., 2011. An Analysis of Wi-Fi Based Indoor Positioning Accuracy. Computer Science (1407-7493), 47.

[41] Koyuncu, H. and Yang, S.H., 2010. A survey of indoor positioning and object locating systems. IJCSNS International Journal of Computer Science and Network Security, 10(5), pp.121-128.

[42] Li, X., 2018. A GPS-based indoor positioning system with delayed repeaters. IEEE Transactions on Vehicular Technology, 68(2), pp.1688-1701.

[43] Liu, R., Yuen, C., Do, T.N. and Tan, U.X., 2017. Fusing similarity-based sequence and dead reckoning for indoor positioning without training. IEEE Sensors Journal, 17(13), pp.4197-4207.

[44] Namiot, D., 2015. On indoor positioning. International Journal of Open Information Technologies, 3(3), pp.23-26.

[45] Satan, A. and Toth, Z., 2018, January. Development of Bluetooth based indoor positioning application. In 2018 IEEE international conference on future IoT technologies (Future IoT) (pp. 1-6). IEEE.

[46] Song, Z., Jiang, G. and Huang, C., 2011, May. A survey on indoor positioning technologies. In International conference on theoretical and mathematical foundations of computer science (pp. 198-206). Springer, Berlin, Heidelberg.

[47] Wu, C., Mu, Q., Zhang, Z., Jin, Y., Wang, Z. and Shi, G., 2016, June. Indoor positioning system based on inertial MEMS sensors: Design and realization. In 2016 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER) (pp. 370-375). IEEE.

[48] Xu, R., Chen, W., Xu, Y. and Ji, S., 2015. A new indoor positioning system architecture using GPS signals. Sensors, 15(5), pp.10074-10087.

[49] Yang, C. and Shao, H.R., 2015. WiFi-based indoor positioning. IEEE Communications Magazine, 53(3), pp.150-157.

[50] Yang, J., Lee, H. and Moessner, K., 2016, October. Multilateration localization based on Singular Value Decomposition for 3D indoor positioning. In 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN) (pp. 1-8). IEEE.

[51] Zheng, L., Zhou, W., Tang, W., Zheng, X., Peng, A. and Zheng, H., 2016. A 3D indoor positioning system based on low-cost MEMS sensors. Simulation Modelling Practice and Theory, 65, pp.45-56.

[52] Aguilar, Wilbert G and Luna, Marco A and Moya, Julio F and Abad, Vanessa and Ruiz, Hugo and Parra, Humberto and Lopez, William. Cascade classifiers and saliency maps based people detection. 2017.

[53] Brena, Ramon F and Garc{\'\i}a-V{\'a}zquez, Juan Pablo and Galv{\'a}n-Tejada, Carlos E and Mu{\~n}oz-Rodriguez, David and Vargas-Rosales, Cesar and Fangmeyer, James. "Evolution of indoor positioning technologies: A survey." Journal of Sensors (2017).

[54] Gentner, Christian and Ulmschneider, Markus and Kuehner, Isabel and Dammann, Armin. Wifi-rtt indoor positioning. 2020.

[55] Yang, Qing and Zheng, Shijue and Liu, Ming and Zhang, Yawen. "Research on Wi-Fi indoor positioning in a smart exhibition hall based on received signal strength indication." EURASIP Journal on Wireless Communications and Networking (2019).

[56] "Research on Wi-Fi indoor positioning in a smart exhibition hall based on received signal strength indication." EURASIP Journal on Wireless Communications and Networking (2019).

[57] Yue, Hanhui and Zheng, Xiao and Wang, Juan and Zhu, Li and Zeng, Chunyan and Liu, Cong and Liu, Meng. Research and Implementation of Indoor Positioning Algorithm for Personnel Based on Deep Learning. 2018.

[58] Zhou, Cheng and Yuan, Jiazheng and Liu, Hongzhe and Qiu, Jing. "Bluetooth indoor positioning based on RSSI and Kalman filter." Wireless Personal Communications (2017).