

Dartmouth College

Dartmouth Digital Commons

Dartmouth College Undergraduate Theses

Theses, Dissertations, and Graduate Essays


Spring 2021

An inside vs. outside classification system for Wi-Fi IoT devices

Paul Gralla

Dartmouth College, paul.gralla@outlook.com

Follow this and additional works at: https://digitalcommons.dartmouth.edu/senior_theses

 Part of the [Computer Sciences Commons](#), and the [Digital Communications and Networking Commons](#)

Recommended Citation

Gralla, Paul, "An inside vs. outside classification system for Wi-Fi IoT devices" (2021). *Dartmouth College Undergraduate Theses*. 215.

https://digitalcommons.dartmouth.edu/senior_theses/215

This Thesis (Undergraduate) is brought to you for free and open access by the Theses, Dissertations, and Graduate Essays at Dartmouth Digital Commons. It has been accepted for inclusion in Dartmouth College Undergraduate Theses by an authorized administrator of Dartmouth Digital Commons. For more information, please contact dartmouthdigitalcommons@groups.dartmouth.edu.



AN INSIDE VS. OUTSIDE CLASSIFICATION SYSTEM FOR
WI-FI IOT DEVICES

Paul Gralla

Supervisor

Prof. David Kotz

Undergraduate Computer Science Thesis

Dartmouth College

June 1, 2021

Abstract

We are entering an era in which Smart Devices are increasingly integrated into our daily lives. Everyday objects are gaining computational power to interact with their environments and communicate with each other and the world via the Internet. While the integration of such devices offers many potential benefits to their users, it also gives rise to a unique set of challenges. One of those challenges is to detect whether a device belongs to one's own ecosystem, or to a neighbor – or represents an unexpected adversary. An important part of determining whether a device is friend or adversary is to detect whether a device's location is within the physical boundaries of one's space (e.g. office, classroom, home). In this thesis we propose a system that is able to decide with 82% accuracy whether the location of an IoT device is *inside* or *outside* of a defined space based on a small number of transmitted Wi-Fi frames. The classification is achieved by leveraging a machine-learning classifier trained and tested on RSSI data of Wi-Fi transmissions recorded by three or more observers. In an initialization phase the classifier is trained by the user on Wi-Fi transmissions of a variety of locations, inside (and outside). The system can be built with off-the-shelf Wi-Fi observing devices that do not require any special hardware modifications. With the exception of the training period, the system can accurately classify the indoor/outdoor state of target devices without any cooperation from the user or from the target devices.

Contents

1	Introduction	4
1.1	Definitions	4
1.2	Motivation	5
1.3	Proposed approach	8
1.4	Contributions	10
2	Background	10
2.1	IoT space	10
2.1.1	Smart home	11
3	Related work	13
3.1	Indoor localization	13
3.2	Location based access control	15
3.3	Wi-Fi characteristics	16
4	System and implementation	18
4.1	Components	18
4.1.1	Home hub	19
4.1.2	Observer	19
4.1.3	Target and training target	21
4.2	RSSI collection	23
4.2.1	Frame generation	23
4.2.2	Frame sniffing and filtering	25
4.2.3	Data transfer	27
4.3	Data processing	28
4.3.1	Missing frames	28
4.3.2	Merging	29
4.3.3	Data standardization	30
4.4	ML model – training and testing	30
4.4.1	Metrics	31
4.4.2	Candidate models	33
4.4.3	One class vs. Binary	34
4.4.4	Evaluation methods	34

5	Experimental evaluation	35
5.1	Experiment setup	35
5.1.1	Experiment site	36
5.1.2	Test execution	37
5.1.3	Variables and test runs	38
5.2	Measurements and results	39
5.2.1	Dataset	39
5.2.2	Effects of evaluation methods	40
5.2.3	Classifier performance	43
5.2.4	Classifier comparison	45
5.2.5	Varying transmission power	46
5.2.6	Different observer placement	47
5.2.7	One class vs Binary	48
5.2.8	Using three observers	48
6	Discussion and Conclusion	49
	Appendices	57
A	Observer Code	57
B	Floor Plan	58
C	Observer Positions	58

1 Introduction

There is a trend to integrate Internet of Things (IoT) devices into our everyday lives. The advent of these so-called “Smart Devices” offers a variety of benefits and advantages for consumers, but also poses some critical security challenges.

1.1 Definitions

The goal of this thesis is to develop a system that, for a given ‘target’ device detected by a residential Wi-Fi system, determines whether it is *inside* or *outside* the residence. To clearly understand this goal we need to first define a few key terms and system components.

Key Terms:

- **Smart Things** refers to IoT devices [14]. They “typically have the ability to sense their physical environment (through sensors) and, sometimes, to act on the environment (through actuators). They may or may not have a human user interface. They may be stationary or mobile. They may be small (like a remote control) or large (like a refrigerator). They may be battery powered or line powered.”
- A **target** is a Wi-Fi device that transmits on any channel. In our scenario, a target is a Smart Thing, but it may be any Wi-Fi transmitter.
- A **residence** refers to a single-story multi-room contiguous space, defined by a simple convex polygon that forms its **boundary**. This could be an apartment or a stand-alone house.
- **Inside** describes every location that is inside the boundary.
- **Outside** describes every location that is not *inside*.

System Components:

- **Home hub** is the central unit of our system. The home hub processes the data and determines whether a target is inside or outside of the boundary.
- An **Observer** refers to a Wi-Fi receiver that records RSSI values of Wi-Fi frames transmitted by the target; it reports these readings to the Home hub.
- The **training target** is used in the training phase to create sample training data.

1.2 Motivation

A key step in the security of personal IoT networks is to make sure that no unwanted devices can join the network. One important factor in determining whether a connection request stems from a friendly device or an adversarial device is the physical location of the device sending the connection request. A connection request sent from outside of the physical bounds of one's residential space is more likely to be an adversarial request than a request sent from the inside. Frames transmitted from the MAC address of a known device expected to be inside, but that are determined to be arriving from outside, may indicate an adversarial attempt to spoof the known device. So, it would be helpful to distinguish the location of a transmitter as being *inside* or *outside* the home.

We propose such a system. It determines whether a *target* device is *inside* or *outside* the boundary of a physical space. Such a system can, for example, determine whether an incoming connection request is coming from *outside* of a house or list all the devices whose transmissions are coming from the *inside* of an apartment.

A possible use case evolves around preventing adversaries from gaining access to a home network. This use case is based on the assumption that new connections from devices physically

inside the apartment are more trustworthy than connection requests originating from the outside. Obviously, determining whether a device is inside or outside should not be the only method of authentication – but it is an important additional context when authenticating new devices and new connections. Whenever a device sends a new connection request to the home system, the Wi-Fi frames of the request are analyzed and if they are classified as *outside*, a warning is issued to the user that the device connecting is outside of their apartment. The user can then decide what to do with this information. For example, if the request is from the smart garden robot, or other device known or expected to be outside, they can approve the connection request. However, if the request stems from an unknown device, or a device known to be indoors but unexpectedly classified as outside, the user should reject the request and investigate – perhaps permanently banning that device from their network.

Consider a rental apartment as one possible scenario. It is now a common practice for AirBnB hosts to install monitoring equipment to ensure that their tenants abide by their stated policies [8]. With AirBnB hosts already deploying such devices, it is likely that renters of long-term rentals will follow suit and try to protect their property by installing such Smart Things. These networked devices might include temperature, humidity, sound monitoring devices, or even cameras. The monitoring devices are owned and operated by the owner, not the renter, and may be positioned in such a way that makes it hard for the renter to detect them. Further, these devices might be configured to be non-cooperative and thereby not follow common device-discovery communication protocols.

A failure to detect these devices might have severe security and privacy implications for renters, as the landlord could potentially gain access to the renter’s private space. A thorough manual inspection of the apartment by the renter can be tedious and error prone; even with such effort,

a renter can never be sure that no monitoring devices are hidden. Here, the system developed in this paper comes into play. The user can train and query our system to display all the devices that are inside the apartment. The system then proceeds to monitor all Wi-Fi traffic and can display all the devices whose location is determined to be inside the apartment, based on Wi-Fi frames transmitted by them. If this list includes an unexpected or unknown device, the user can use other tools to locate and identify the device. If all the devices in the list are known to the user, the user has increased confidence there are no other Wi-Fi-based monitoring devices installed by the landlord or other parties.

These use cases could perhaps be addressed by accurate localization techniques. Although an extensive amount of research has been done on the problem of indoor localization, many of these indoor localization methods are either complex and sensitive, require special hardware or configuration, are insufficiently accurate to distinguish inside from outside, or are impractical to apply to the IoT space, as described in more detail in the next section. Some of them also require location-specific, labelled training data to determine the location – data that can be hard to generate for private homes. To achieve the goal of inside vs. outside classification it is further necessary to combine these indoor localization methods with a floor plan and the relative positions of the observing devices. Obtaining this information poses an additional challenge to private users, increasing the effort that needs to be put into configuring the system to achieve accurate results.

Of course, it is unnecessary to determine the exact location of a transmitter to determine whether it is located inside or outside of a physical space. In this thesis we propose a system that is able to determine whether the transmitter of received Wi-Fi frames is inside or outside a physical space with 82% accuracy, without determining its exact position. It requires brief user effort in the training phase, and the *observer* components can be built using commercially available

hardware that does not require special modifications.

The system can easily be deployed in a variety of different environments. While we were only able to demonstrate its effectiveness in one private-home setting, we are hoping that similar results can be achieved in many other environments and building types. To set up the system, the user places three or more observers at different locations around the residence, and connects them to the *home hub*. The user then walks around the inside and outside of the residence with their smartphone (or other Wi-Fi transmitter device), while the home hub and observers collect Wi-Fi signal strength and use that data to train a classifier. Notably, our approach does not rely on any other data than the received Wi-Fi frames and the data collected in the training phase.

1.3 Proposed approach

We propose a system composed of three or more *observers* and a *home hub* that processes the collected data. The system can then classify whether the location of a *target* is *inside* or *outside*.

The basic workflow of the system can be described as follows. The observers are monitoring Wi-Fi traffic and capture all frames they observe. The observers measure the received signal strength indicator (RSSI) for frames sent by the target device; the observers forward only those values to the home hub. The home hub collects these RSSI values, aligns the values of corresponding frames, and retains data only for frames for which at least three RSSI readings from different observers were received.

The information flow of the system is similar during the training and operating phase. During both phases the Observers forward their readings to the home hub, which then joins the RSSI values for each frame and further processes them.

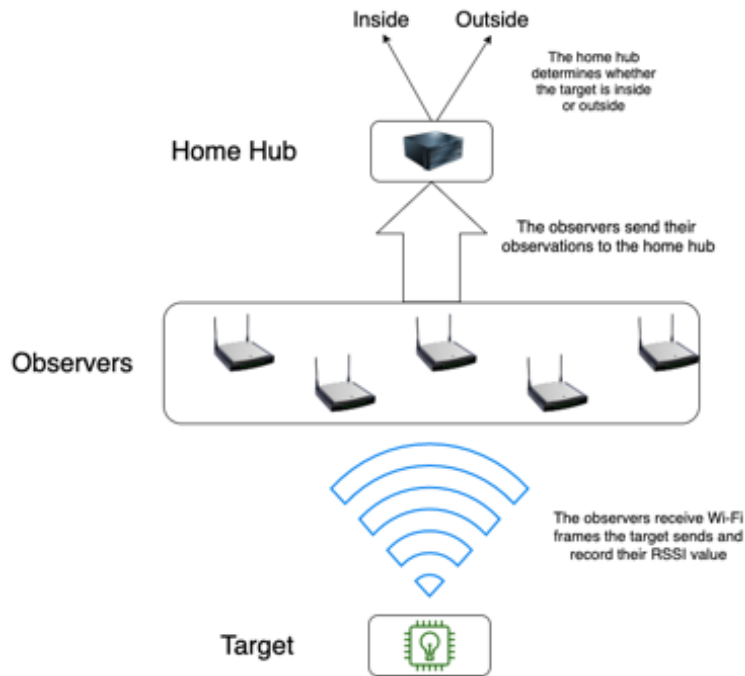


Figure 1: Information flow overview

During training the user transmits frames from the *inside* and *outside* using the “training target”. The transmissions are done in a way such that the frames can be class-labelled by the *home hub*. The training transmissions can be sent by any Wi-Fi transmitter; one suitable approach would be a smartphone running an app co-designed with the system. All the frames that are collected in the training phase are used to train a machine-learning classifier.

When the training period is completed the system enters operational mode. During operation, it is only necessary to record one frame that is captured by three or more observers to infer the inside/outside status of the transmitter. Each frame is classified individually based on the RSSI values recorded by the observers. Because there can be slight variations between recorded RSSI values, the technique could be extended to increase accuracy by considering multiple captured frames.

1.4 Contributions

The key contribution of this thesis is the development of a system that

- can detect whether a Wi-Fi transmitting Smart Thing is inside or outside of a residence with 82% accuracy,
- only needs training data labelled for two classes (inside and outside), and
- can be built using commercially available hardware that does not require special hardware modifications.

2 Background

As the system developed in this thesis is tailored to the IoT space, it faces a unique set of constraints and requirements. To understand these constraints and the general necessity for the system, it is helpful to have a understanding of the general IoT space, and the smart home subspace to which the system is tailored.

2.1 IoT space

The Internet of Things (IoT) is transforming our everyday lives by bringing controlling, monitoring, and connective capabilities to all sorts of smart devices. *Smart Things* can be applied to a variety of contexts like smart homes, cars, wearables, factories, infrastructure, agriculture, and more [9]. Smart Things can have different sensing and actuating capabilities but all share one common property: their connection to the Internet. For the purposes of this thesis, the Internet of Things refers to “a vision in which a wide range of everyday objects become *Smart Things* through

the inclusion of digital electronics and a network interface that allows them to communicate with other Things and, directly or indirectly, through the Internet with remote services” [14].

2.1.1 Smart home

An important subset of the general IoT space are “smart homes”. Smart homes include smart things like cameras, speakers, door locks, kitchen appliances, thermostats, and lights. Connecting all these devices to the home network gives the user the ability to control virtually all features of their house with a finger tap on their phone. The system proposed in this thesis is focused on indoor vs. outdoor classification of Smart Things in connected homes and buildings.

This integration of Smart Things into a residence – one of the most personal of all spaces – raises many security and privacy challenges. Many of these Things have the capability to sense and record private habits, conversations and actions. The ability of Smart Things to record private moments and to communicate over the Internet gives adversaries a variety of possible ways to infringe upon security and privacy. Furthermore, due to hardware limitations on older Smart Things, Internet-strength security features are sometimes hard to realize on some already deployed devices [27].

Smart Things also come with a set of constraints. We need to consider four main constraints when designing our system.

The first constraint is that these devices have limited transmission capabilities. While some of these devices are able to communicate using protocols like RFID, NFC, Bluetooth, Zigbee, and Wi-Fi, few Smart Things are able to communicate using all of these protocols. One of the most commonly used protocol is the 802.11 Wi-Fi protocol that many Smart Things support. The commonality of Wi-Fi support on Smart Things made us design our system to use only Wi-Fi

signals, unlike some of the related work on indoor localization that proposes the use of different protocols and methods.

The second important constraint is the rate of Wi-Fi transmissions Smart Things send. While a smart camera may be frequently communicating with the home network to send its recordings, a smart thermostat, for example, might send data only once every half hour. The reason for these infrequent communications are many: some devices might be battery powered and try to limit their power consumption while some application domains of Smart Things just do not require frequent data transmissions. Limited Wi-Fi transmissions also mean limited data to localize a device. The limited amount of data needed to be considered when designing our system that is able to classify a device's location based on only one frame that is captured by three observers (although accuracy increases the more frames are collected).

The third important constraint is the availability of additional sensors in Smart Things. Some of the current research on accurate indoor localization requires additional sensors like a gyroscope or compass to be present in the device that is being localized (e.g., [16, 4]). However, most Smart Things are not equipped with these additional sensors. Even though some Smart Things might have additional sensors like a gyroscope, the presence of such a sensor cannot be assumed for all Smart Things that our system tries to classify. Our system only needs the data that is obtained by analyzing Wi-Fi frames and does not require additional sensor data to work.

The final constraint is the ability of Smart Things to cooperate for the purpose of localization. Many indoor localization methods (such as the Return Time of Flight method used in μ Locate [18]) require the device to cooperate. Usually this cooperation consists of following a certain response, time stamping, or channel-switching protocol. However, most, if not all, Smart-Home Things do not follow these specialized protocols. Some devices are just passive observers that transmit

their readings and do not even listen for incoming Wi-Fi signals. Furthermore, adversarial devices cannot be expected to cooperate, or may cooperate adversarially, misleading any attempt to locate them. Our system classifies targets using only passive observations of Wi-Fi transmissions.

3 Related work

Since GPS localization is unreliable in indoor environments, a different approach is needed. A substantial amount of research has addressed the problem of indoor localization, with a focus on exploiting the characteristics of Wi-Fi transmissions. The problem of inside-outside classification is, in a sense, a coarse version of localization because it needs only to distinguish between two classes and not among a continuous set of locations. Indeed, for many interesting application scenarios, a location is not needed – only an indication of whether the device is inside or outside a defined space. Further, an inside-outside classification system may require less training data, which is easier to collect as the labels are just drawn from two classes. Still, we can learn a lot from prior work on indoor localization and may adapt some of those techniques.

3.1 Indoor localization

The goal of indoor localization is to determine the *location* of a *target* device, typically on a coordinate system defined by a building floorplan. There are two key factors differentiating proposed indoor-localization systems: which system computes the location, and which wireless technology is used for measurements.

The first differentiating factor is whether the location is determined by the device (self-localization) or by an external system (monitor). In a device-based system the target device determines its own

position within an indoor space; in a monitor-based system an *observer* (or multiple observers) determine the location of the target device. Our approach is the latter.

The second differentiating factor is the wireless technology used to make measurements. Bluetooth, ZigBee, ultrasound, LiDar, and Wi-Fi have all been proposed in the literature (e.g., [15, 10, 19, 20]). No one technology is feasible for the entire IoT ecosystem, as not all Smart Things have the capability to communicate over radios like Ultra-wideband, ZigBee, or ultrasound. The widespread adoption of Wi-Fi, particularly in Smart Things, make Wi-Fi the most attractive option for our system. Indeed, the trends point to an increasing emphasis on Wi-Fi for even the smallest devices. Hence, this analysis of related work will focus on Wi-Fi based systems. In the literature, several localization systems use Wi-Fi, including SpotFi [13], Chronos [22], ArrayTrack [26] and a variety of approaches that use RSSI-Fingerprinting [7, 11, 12, 21, for example]. These methods use properties of Wi-Fi Signals like the return time of arrival, received signal strength indicator (RSSI), or channel state information (CSI).

All of these approaches, however, come with certain limitations that make it hard to apply them to the smart home IoT ecosystem. ArrayTrack, for instance, requires a special antenna setup that consists of multiple antennas aligned in a 40cm long array [26]. The antenna array is then used to estimate the angle of arrival of Wi-Fi signals emitted by the target. The location of the target is estimated by multi-angulation using the measurements of multiple antenna arrays. However, having multiple antenna arrays in a residence for the pure purpose of inside-outside classification seems hardly practical as they are bulky and likely expensive.

The Chronos system, on the other hand, requires cooperation from the target [22]. The observer and target both follow a specific channel-hopping protocol to generate transmissions on different bandwidths, which are then used to estimate the distance between the two. However, it is not fea-

sible to expect such cooperation from Smart Things as described in the background section. On paper, the SpotFi system seemed to be a good candidate, yet we were not able to reproduce the results in our own experiments [13]. In addition, the system also required a special arrangement of antennas. Finally, the fingerprint-based approaches [7, 11, 12, 21] work without device cooperation or specialized hardware, but they require location-labeled training data to create an RSSI fingerprint of each location in the environment. This labeled training data is complex and time consuming to collect, which makes it impractical to apply these methods to use cases in private residences.

We considered building our system on top of one of the above-described indoor localization systems. Using indoor localization methods for the problem of inside-outside classification, however, also requires one to have access to a floor plan (or map of the boundary) and knowledge about the location (and for some systems, the orientation) of the observing devices. The need for these additional data provides another hurdle to building a private-use, inside-outside classification system on top of an indoor localization system.

We drew a lot of inspiration for our work from these described indoor localization methods; our system uses similar ideas, but applies them to the problem of inside-outside classification instead of localization. Precise indoor localization is not required to achieve our goal.

3.2 Location based access control

Another related area of work tries to accomplish a similar objective, but targeted at a different use case [5]. This approach tries to determine whether smartphones are within certain physical bounds of a space, for example a café, to determine whether they should have access to a wireless network.

The goal is to limit access to a certain qualifying user group, letting customers of the café connect to the Wi-Fi network while preventing passersby or people working in a nearby office to gain access. It uses the fact that smartphone users move around the physical space while their smartphones are constantly transmitting signals over the wireless network. The system then collects the RSSI data of these measured sequences of Wi-Fi frames. A classifier is trained on those RSSI sequences and can then determine whether they belong to an inside or outside class. This approach is thereby heavily geared towards smartphones that are actively used by people and the method is hard to transfer to the IoT space as most Smart Things remain stationary and it is thereby hard to leverage the additional information that RSSI sequences provide. Our work, however, still benefits from some of the observations made in this paper, as will become clear in following sections.

3.3 Wi-Fi characteristics

There are many Wi-Fi signal characteristics and properties used in the literature to estimate location. The most prominent are Time of Flight (ToF), Return Time of Flight (RToF), Angle of Arrival (AoA), Time difference of Arrival (TDoA), and Received Signal Strength indicator (RSSI) [28]. Each of these characteristics has its own set of advantages and disadvantages and different localization systems proposed in research build on different ones.

The ToF method measures the time the signal takes to propagate from the transmitter to the receiver. Since the propagation speed is known to be the speed of light, this in theory is enough information to calculate the distance between transmitter and receiver. With the use of multiple receivers one can perform multilateration to determine the relative position of the transmitter to the receivers. Since this method requires exact time measurements, the transmitter and receiver

need to be equipped with high-precision clocks, must be closely synchronized and, depending on protocol, also transmit timestamps [6]. These requirements poses a challenge for the application of this method to the IoT space as most Smart Things do not have high-precision clocks, the capacity to be strictly synchronized with the observers, or are not configured to send timestamps along with regular communication.

Similarly, the RToF method measures the time it takes for a signal to propagate from the observer to the device and back to the observer [6]. The method requires that the device that is subject to localization to receive the signal and immediately send a response. The time it takes for the signal to be returned to the observer heavily depends on the time it takes the device to process the signal. The time it takes to return signals is not standardized across all Smart Things, posing a critical challenge for the application of this method to our use case. A proposed new Wi-Fi protocol may be able to mitigate this problem [24], but this standard has not yet been adopted so this approach is unfeasible for the current IoT infrastructure.

In contrast, the TDoA method measures the time difference it takes for the signal to reach different observers. This requires strict synchronization between observers and highly accurate clocks. The biggest drawback of this method is that location accuracy suffers from multi-path fading in indoor environments. Otherwise, the system could support the goal of indoor-outdoor distinction. We chose RSSI, however, due to the complexity of collecting TDoA data.

The AoA method, which is used by ArrayTrack [26], estimates the angle of the Wi-Fi signals the observers receive from the transmitter. The angle is usually estimated using an antenna array or exploiting CSI information and therefore often specialized hardware is required to collect accurate results. And accurate results are important since, as the distance between transmitter and receiver grows, even small measurement errors can result in big localization errors [16]. In addition, multi-

path effects, which are common in indoor environments, make it hard to estimate the correct AoA.

After thorough analysis, we decided RSSI was best suited for the purpose of this thesis. The main advantages are the ease of collection, the low hardware requirements on transmitter and receiver side, and no need for cooperation from the target. Nearly every off-the-shelf commodity Wi-Fi card provides RSSI values for received frames. The main drawback of RSSI is its proneness to multi-path fading and indoor noise [25]. However, for our system these effects are not too much of an issue as they stay relatively constant for the same location [23].

4 System and implementation

The system proposed in this thesis determines whether the *target* is *inside* or *outside* the *boundary* of a residence. The system performs this classification by recording the RSSI values of transmissions of the target. These recordings are performed by the *observers*, which then forward the readings to the home hub, which runs the computation to classify the target as inside or outside.

As in most related work on indoor localization, we consider the problem of inside-outside classification only in the two-dimensional plane. It should be feasible to expand our approach to three dimensions.

4.1 Components

The system consists of three components. These are the home hub, the observers, and the target. The observers work together with the home hub to classify the location of the target. To understand how these RSSI values are collected and then processed it is first necessary to understand how each of the components is implemented.

4.1.1 Home hub

The home hub is the central computing unit of the system, where all the recorded RSSI values are aggregated and processed. The home hub can theoretically be any computer. For the implementation in this thesis we used a 2017 MacBook Pro running MacOS BigSur. In a real home, the home hub might be an embedded appliance, such as a Wi-Fi router or a “smart hub” like the voice-based personal assistants now available from several vendors. A sophisticated home hub could act as the trusted interface to a Smart Home and handle authentication, access management, configuration and decommission of Smart Things, and allow users to securely manage their Things. We envision a device that looks somewhat similar to, for example, an Apple HomePod or an Amazon Echo.

We perform the data-processing in Python using the *pandas* and *scikit-learn* libraries. In principal, any device running Python can perform the processing. It would, for example, be feasible to run the code on a Raspberry Pi.

4.1.2 Observer

The system is made up of three or more *observers*. Our implementation used four. As we are only considering the two-dimensional case, all observers were placed on the same plane, parallel to the floor. The *target* device is also located on this plane. Placing the observers and the target on the same plane also ensured reproducibility and comparability of different experiment runs. Investigating the effect of not having all devices on the same plane will be left to future research. Each observer is a MacBook Pro from 2012 running Ubuntu 20.04 Linux and equipped with a tp-link Archer T2U Plus [1] external USB Wi-Fi antenna (Figure 2). The observers are able to receive frames on the 2.5 GHz and 5 GHz Wi-Fi bands with the tp-link antennas. For the frame



Figure 2: Observer (pictured on the ground, was placed on a stool during experiments)

collection we use Python 3 and the *scapy* packet manipulation and sniffing library.

The antenna is positioned orthogonal to the floor. Since the target is on the same plane as the observers, the orthogonal placement of the antenna means that, regardless of the target's position on the plane, the direct signal propagation path from the target points to the antenna at the same 90 degree angle. Positioning the antenna at a non-orthogonal angle to the floor would mean that signals originating from different points on the plane would have different angles of arrival at the antenna. As the angle of arrival affects the measured RSSI [17], angled antennas would create measurement biases towards certain locations. Therefore, the best orientation for the antenna is orthogonal to the plane. The antennas we used had a hinge that could be set into fixed positions and the position we used for the experiments was orthogonal to its USB port. Since we always placed the observers on flat surfaces, this ensured that the antenna was orthogonal to the floor. For the sake of this thesis we assumed that minor inaccuracies of the antenna angle ($< 2^\circ$) would be

negligible. We placed all the observers at a height of 1m on wooden chairs and stools to reduce signal attenuation of the floor.

Because the system is operating in an indoor environment, multipath effects occur and reflected signals will also arrive from angles not orthogonal to the antenna. Multipath effects here refer to “the reflection, diffraction, and scattering of electromagnetic waves on the propagation path” [23]. These effects are dependent on the location of the target and observer, as well as the surrounding environment (walls, floors, furniture, people, etc.). This location dependency, however, also means that multipath signals (at least those caused by stationary objects like walls and furniture) contribute to the uniqueness of the RSSI “fingerprint” for each location and area, likely improving the accuracy of our system. As these multipath effects are hard to capture and quantify, investigating the specific impacts of multipath effects on the accuracy of our system was out of the scope of this thesis.

The system is based on the observers being in the same location during the training and the operational phase. It was out of the scope of this work to determine how a change in location of one of the observers would affect the performance of the system.

4.1.3 Target and training target

When our system is operational it determines whether the target is inside or outside. The target is an IoT device. During the training phase a *training target* is used to produce training data for our classifier. For a practical implementation, we envision using a smartphone app to produce the training data by sending Wi-Fi frames from the user’s smartphone. Writing such a smartphone app, however, was out of the scope of this thesis.

Therefore, we used a Raspberry Pi (Figure 3) to act as the target and training target during our



Figure 3: Target and training target: Raspberry Pi 4 model B

experiments. We used the Kali distribution of Linux (version 2021.1) as it comes with versatile pre-installed wireless drivers and tools. For the frame transmissions we employed Python 3 and the *scapy* packet manipulation and sniffing library. We used the Raspberry Pi for the roles of target and training target because it was easy to record reliable and reproducible measurements. The Raspberry Pi also supports manipulating the transmission power of its antenna. The ability to transmit at different power levels allowed us to emulate the use of different devices that might also transmit at different power levels. The goal of our system is to accurately classify targets regardless of their respective transmission strength. Hence, being able to experiment with different transmission powers was essential. For the experiments we connected a simple button to the GPIO pins of the Raspberry Pi to initiate the transmission of a set amount of frames.

The antenna of the Raspberry is the silver rectangle that can be seen at the top left of Figure 3. While we were not able to find specifications about the exact build or orientation of the antenna we found in initial experiments that the orientation around the vertical axis of the Raspberry Pi had negligible effect on the RSSI.

The system should be able to classify Wi-Fi frames independent of the device that sent them. A Wi-Fi signals sent from, for example, a Smartphone or Smart Thing are not different to the signals sent using the Raspberry Pi. Thereby the classification accuracy should be independent to the type of transmitter. We leave it to future research to experimentally confirm this claim.

4.2 RSSI collection

Our system uses RSSI data recorded by the observers to classify a target. In operational mode the system filters frames for the MAC address of the target and processes the recorded RSSI values for each frame. For our experiments, however, we had more control over the generation of the Wi-Fi frames and therefore employed slightly different techniques for the frame filtering as described in the next few sections. For a real deployment only minor changes to the filtering conditions would be necessary to obtain the same results. Depending on the use-case scenario one could filter for a single target, a set of targets, or just classify all observed frames.

4.2.1 Frame generation

The frames that are used for the classification could be connection requests or normal data transmissions. The target may or may not be connected to the home network at time of classification. Hence, our system is designed to classify frames regardless of their destination or the network connection of the target. Therefore, for our implementation of the system we put the wireless card in “monitor mode,” allowing it to transmit without being associated with a Wi-Fi network router. Setting the card to monitor mode also allowed us to select the channel on which we send the frames.

To activate monitor mode on the card we used the *aircrack-ng* wireless tool suite for Linux.

Aircrack-ng comes with a tool to activate monitor mode on Wi-Fi cards called *airmon-ng*. The tool creates a virtual interface, which is set to monitor mode, on top of the existing Wi-Fi interface. We used the tool with the following commands. Here the wireless interface is called `wlan0` and the virtual monitor interface is then named `wlan0mon`.

```
sudo airmon-ng check kill  
sudo airmon-ng start wlan0
```

The first line kills background processes that might interfere with the monitor mode, like for example the Linux network manager. The second line then starts the virtual monitoring interface. After setting the wireless interface to monitor mode, we only needed to set the desired channel and the transmission power (Here channel 36, and transmission power 30dbm) with the following commands.

```
sudo iwconfig wlan0mon channel 36  
sudo iwconifg wlan0mon txpower 30
```

After the wireless interface is set to monitor mode we can begin transmitting the frames. We decided to send data frames to an artificial MAC address (01:02:03:04:05:06), which no real device would use. We chose this address because it simplified the filtering on the receiving side. In a real deployment, any frame transmitted from the target would suffice. We wanted our experiments to be as accurate and reproducible as possible. Therefore, we wanted to avoid accidentally using Wi-Fi traffic, which was not explicitly sent by us, for our data analysis. Reproducibility can only be achieved if we only train and classify on frames that we transmit specifically for our experiments. We kept the frame that we are sending as simple as possible. Using the *scapy* library we can craft a simple Wi-Fi 802.11 data-frame with a payload using the following line in Python.

```
pkt = Dot11(addr2=rx_addr, type=2)/Raw(load=id)
```

The data we send in each frame is a unique ID. Such a unique ID makes it easy to join the observations of the different observers. For our experiments we used a script that transmits 50 frames every time the button is pressed. We also implemented a sleep timer (after the button is pressed and before the frames are sent) to allow some time to walk away from the Raspberry Pi, so our presence would not interfere with the transmissions. The script we used can be found in Appendix A; the current id is loaded from a persistent file to prevent duplicate IDs in between restarts of the script.

4.2.2 Frame sniffing and filtering

Because we are interested in the RSSI of frames sent by devices that might currently not be connected to the home Wi-Fi network we need to consider all frames that are observable at any time. We do this by setting the Wi-Fi card of the observer into *monitor mode*. In monitor mode, the card receives all frames that it can detect, regardless of their origin and destination. Independent of the mode, the employed Wi-Fi antennas were only able to sniff frames on a single channel at a time. For the sake of simplicity we conducted all our experiments on channel 36, which has a center frequency of 5.18 GHz.

We activated the monitor mode and set the channel with the following Linux commands (in this case, for a wireless interface called wlan0):

```
sudo ifconfig wlan0 down
sudo iwconfig wlan0 mode monitor
sudo ifconfig wlan0 up
```

```
sudo iwconfig wlan0 channel 36
```

Note we also needed to turn off the Linux network-manager. (With the network manager still running the wireless card was only able to sniff management frames but not the data frames sent by the Raspberry.) The network manager can be turned off with the command:

```
sudo service network-manager stop
```

The scapy library provides a sniffing function that takes a callback function, which handles each received frame, as an argument. The sniffing function can be called as follows. Here the sniff function is called on the interface wlan0 and the callback function is called PacketHandler.

```
sniff(iface=wlan0, monitor=True, prn = PacketHandler)
```

The real filtering work then happens in the callback function. For our experimental implementation it looked like this:

```
# type(seen) == set()
# type(count) == int

def PacketHandler(pkt):
    global count, seen, output_file
    if pkt.haslayer(Dot11): # Wi-Fi 802.11 Frame
        if pkt.type == 2: # Data Frame
            if pkt.addr1 == target_mac_address and pkt.addr2 == "01:02:03:04:
                05:06":
                ID = int(pkt[Raw].load)
                rssi = int(pkt.dBm_AntSignal)

                if ID not in seen:
                    print(f'received id: {ID} rssi: {rssi} count: {count}')
                    output_file.write(f'{ID},{rssi}\n')
                    count += 1
                    seen.add(ID)
```

Note also that scapy sent each frame multiple times to prevent frames being lost because of collision. We found that these multiple frames almost always had the exact same RSSI value. Therefore, we only recorded the RSSI value of the first received frame for each ID. After receiving the

ID of a frame we add it to the set of seen IDs and we ignore all frames whose ID we had already seen.

4.2.3 Data transfer

While the data is recorded at each observer, it needs to be transferred to the home hub for processing. For the sake of simplicity, we used a USB stick to transfer the data manually from the observers after the transmissions were completed. There are, however, multiple approaches for data transfer in a real deployment.

The first option would be to connect all the observers to the home network (and thereby to the home hub) with an Ethernet cable. The obvious downside of this is the need for additional cables, unless the home is equipped with an extensive Ethernet or power-line networking capabilities.

The second option is to equip the observers with a second Wi-Fi interface that maintains a connection to the home network. An added Wi-Fi interface would eliminate additional cabling and would therefore be more convenient to install. This option also requires additional hardware (another Wi-Fi interface) for each observer.

The third option is to transfer the data by sending frames on the same Wi-Fi interface the observer uses for sniffing. For this approach to work, the home hub needs to be equipped with a Wi-Fi interface that is set to monitor mode and able to sniff the injected frames. Using this approach would require no additional cabling or network interfaces on the observers. The home hub, however, would most likely need an (additional) Wi-Fi interface, which is set to monitor mode.

In any of these approaches, it is possible that some data is lost between the observer and hub, perhaps due to network collisions. Furthermore, the transmission should use an efficient protocol –

likely batching multiple observations in a single frame, applying compression techniques to reduce the bandwidth needed for observer-hub communications.

All aspects of observer-hub communication are left as future work.

4.3 Data processing

The observation data must be pre-processed before it can be used to train a machine-learning classifier. There are two main processing steps: first merging the readings from the observers, and then dealing with frames that were only captured by a subset of the observers.

4.3.1 Missing frames

There are two common reasons for some observers not receiving individual frames. The first is that the observer is simply out of range of the transmitter. Wi-Fi signals can only travel a limited distance and are attenuated by features of the indoor environment. Furthermore, IoT devices might have limited transmission power (to preserve battery life), further limiting their signal range. Another reason some observers do not receive frames is frame collision. In our experiments, we send the frames in monitor mode (without using request-to-send (RTS/CTS) on the link layer) so there are no guarantees that frames actually reach their destination. During the experiments we conducted it was hard to distinguish between the two causes missing frames. On average we saw around 0.25% of unobserved frames.

We drop frames that are only received by one or two observers, as they do not contain enough data for localization. For frames that only one of the four observers missed we use a null value of -100 for the RSSI recorded by that observer.

Table 1: Format of data-frame with sample values

ID	$RSSI_1$	$RSSI_2$	$RSSI_3$	$RSSI_4$
1	-88	-66	-85	-100
2	-86	-65	-86	-90
3	-72	-53	-78	-82

4.3.2 Merging

After the data transfer is concluded the home hub has four different sets of measurements of RSSI values (since we use four observers). As mentioned in the previous section, if RSSI values are missing they are replaced by the value -100 . Before these measurements can be fed to the classifier, they need to be joined into a single table where each row corresponds to one frame and each column to the measurements of one observer (Table 1).

The frames that we sent during our experiments have a unique id as their payload, which made it easy to join the recorded values into a single table after missing data have been handled.

In a real deployment, the transmitted frames will not have an ID as their payload. During previous experiments we explored the potential for matching frames using their frame checksum (FCS). In the scapy library the FCS can easily be obtained as follows:

```
fcs = int(pkt[Dot11FCS].fcs)
```

We found virtually no duplicated FCS between frames, even for some Wi-Fi management frames. More work is needed to validate this approach. By combining the FCS with a timestamp produced at the observer, even with loosely synchronized clocks, we believe it may be possible to uniquely match frames across observers.

4.3.3 Data standardization

As stated before, one goal of our system is to classify targets regardless of the transmission power of their antenna. Our system should be able to be trained using a target with a transmission power of e.g., 30dBm and then classify transmissions with a send from a target with a transmit power of 15dBm. As described by Cheng et al. [5] discussed in Section 3.2, the change of RSSI caused by a different transmission strength stays relatively stable across different positions. If this claim holds true a $X\%$ decrease in transmission power will lead to the same $Y\%$ decrease of recorded RSSI across all observers. Therefore, it should be enough to standardize the training data and then standardize testing samples by the same factor.

Standardization refers to the process of transforming the data to have a mean of 0 and a standard deviation of 1 using the formula:

$$Z = \frac{x - \mu}{\sigma} \quad (1)$$

Where μ is the mean and σ the standard deviation of the entire dataset. Every single RSSI value x in the dataset is changed to its standardized value Z . In a real deployment, during operational mode, every new sample would be standardized using Equation 1, with μ and σ calculated from the training set before classification.

We evaluated this method in our experiments.

4.4 ML model – training and testing

Once the training data is collected and organized as seen in Table 1; the home hub trains a machine-learning model that classifies measurements during operational mode.

There are different consideration we had to make when deciding between different machine-learning models and evaluation methods, which we describe in the next few sections.

4.4.1 Metrics

To evaluate the success of our machine-learning classifier we need to decide how to quantify its results. The problem at hand can be described as a binary classification problem, distinguishing between the two classes *inside* and *outside*. In binary classification problems the two classes are referred to as positive and negative. Every sample is labeled either positive (inside) or negative (outside); we refer to these labels as the *ground truth*. The classifier classifies every sample to be either positive or negative.

Therefore, there exist four different possible outcomes of each classification as seen in Table 2. A true positive (TP) occurs when the classifier correctly infers positive: the ground truth and inferred class are both positive. A true negative (TN) occurs when the classifier correctly infers negative: the ground truth and inferred class are both negative. A false positive (FP) occurs when the classifiers falsely infers positive: the ground truth is negative but the inferred class is positive. A false negative (FN) occurs when the classifiers falsely infers negative: the ground truth is positive but the inferred class is negative.

After samples have been classified one can produce a *confusion matrix* that depicts the degree to which the classifier ‘confuses’ the two classes. The confusion matrix has the shape of Table 2 and contains the number (or fraction) of each outcome happening.

Binary classifiers are evaluated with one or more of these common metrics.

- **Precision**, $TP/(TP + FP)$, the ratio of true positives to all positive predictions. The higher the precision the fewer false positives occur. A perfect classifier has 1.0 precision.

Table 2: Overview of classification cases

		Ground Truth	
		Positive	Negative
Prediction	Positive	TP	FP
	Negative	FN	TN

- **Sensitivity/Recall**, $TP/(TP + FN)$, the ratio of true positives to total (ground truth) positives. A perfect classifier has 1.0 recall.
- **Specificity**, $TN/(TN + FP)$, the ratio of true negatives to total negatives. A high specificity minimizes false positives. A perfect classifier has 1.0 specificity.
- **Accuracy**, $(TP + TN)/(TP + FP + FN + TN)$, the ratio of correct predictions to total predictions. A perfect classifier has 1.0 accuracy.
- **Balanced Accuracy**: can be calculated as $((TP/(TP + FN) + (TN/(TN + FP)))/2 = (sensitivity + specificity)/2$. The metric weights the accuracy based on the size of the underlying classes and is often used for imbalanced datasets. If, for example, a dataset consist of 90% positive and 10% negative sample, predicting positive for every single sample yields an accuracy of 90% but a balanced accuracy of 50%.
- **AUROC** is the *area under receiver operating characteristic curve*, which we refer to as just **AUC** for *area under curve*. The ROC curve plots the relation between the sensitivity and the False Positive Rate ($FPR = 1 - specificity$) at different prediction probability cutoffs. The area under the ROC curve then is a measurement of separability of the data. The AUC ranges from 0–1 where a classifier that predicts everything wrong has 0.0 AUC and a perfect classifier has 1.0 AUC.

When deciding which metric to use, one needs to look at the specific use case and the desired

outcome. In related research, the accuracy and the AUC are often used to determine the effectiveness. As our dataset is unbalanced (more samples outside than inside), we use *balanced accuracy* to evaluate the performance of our classifier. There is often a trade-off between high precision (few false positives) and high recall (few false negatives). For our specific use case we want to emphasize avoiding false positives and therefore want a high precision.

We prefer false negatives over false positives for security reasons: it is better if a trusted device is classified as outside (negative) when it is inside, resulting in a false negative, than to classify an outside adversary as inside (a false positive). In cyber-security this is often referred to as the *fail-safe default* principle: when the class of a sample is ambivalent, the class that can do less harm should be chosen.

Therefore, we evaluate our model based on its precision, balanced accuracy, and AUC.

4.4.2 Candidate models

Many classifiers can distinguish between two classes. As we have access to labelled training data, we focus on *supervised* classifiers. We chose three supervised machine-learning classifiers:

- Random Forest
- K-Nearest Neighbors (KNN)
- Support Vector Machine (SVM)

We use SVM as the baseline and compare the other two classifiers to it. The results of this comparison can be found in Section 5.2.3.

4.4.3 One class vs. Binary

A deciding factor in the training phase of a machine-learning classifier is the training data that is used. While most classifiers only work if they are trained on both positive and negative samples, SVMs also have the ability to train on only positive examples. Training a binary classifier on only one class is also known as *one vs. rest* classification or *outlier detection*. Such a one-class classifier could be useful in for our system in the case that the outside of a residence is not accessible. Inaccessibility is especially a problem in multi-apartment buildings where it can be unfeasible to access neighboring apartments. In these scenarios a one-class classifier could perform the inside vs. outside classification with only training data from the inside of the apartment.

We tested such a one-class classifier on our data and present the results in the section 5.2.7.

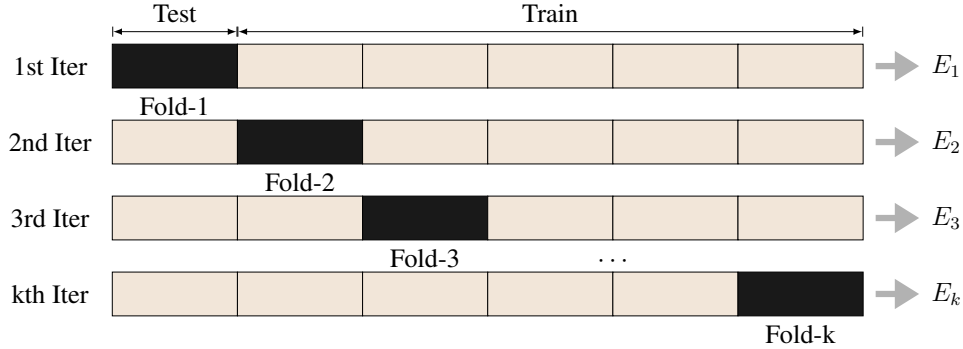
4.4.4 Evaluation methods

In our evaluation we employ two methods for evaluating our classifier: (1) k -fold cross-validation and (2) hold-out validation.

A **k -fold cross-validation** splits the dataset into k different subset of equal size. Usually the subsets are sampled randomly from the data, but we also use cross validation without random sampling as described in 5.2.2. After the dataset is split into k subsets, $k - 1$ subsets are used for training while the last one is used for testing. The training and testing process is repeated k times with a different subset used for testing every time as depicted in Table 3. The result of a k -fold cross-validation tells us how well a classifier performs on a dataset.

We also use **hold-out validation**, which essentially is a 2-fold cross validation. The dataset is divided into two subsets, one training and one testing set. Usually the two sets are produced by

Table 3: Schematic of k -fold cross-validation, code for this table was taken from [2]



randomly sampling from the dataset. Usually the training set is chosen to be the larger one.

5 Experimental evaluation

As the culmination of this thesis we tested our system in a real home environment. The results obtained during these experiments can be seen as a preliminary indicator of our system’s performance. Because of the physical properties of Wi-Fi signals, each environment will be different; further experiments will be needed to confirm whether our approach generalizes.

5.1 Experiment setup

When conducting the experiments our main focus was on reproducibility. As we were repeating the experiments multiple times to investigate the effects of different variables, it was key that the produced measurements could be compared between different runs. The following sections describe how we achieved our methods.



Figure 4: The house used for our experiments

5.1.1 Experiment site

We conducted all our experiments in a single-floor, single-family, stand-alone home in Hanover, New Hampshire (Figure 4). The house footprint was about $160m^2$. We conducted our experiments on the first floor as this allowed us access to all the outside walls. The first floor was made up of one large living space on one side, and multiple private bed-rooms on the other side (see floor plan in Appendix B).

The house had the shape of a rectangle with a little ‘bump’ on one side. The house was built from wood with relatively thin outside walls. In addition, it had a large window facing the backyard. Finally, there was a $2.50m$ by $0.7m$ brick chimney near the middle of the living room.

When considering this testing site the most important thing to note is the material composition of the house. Because wood attenuates Wi-Fi signals less than materials like concrete or stone (Figure 5), transmissions from the outside will be less attenuated in our test site compared to a house made of solid stone or concrete.

We were not able to test the system on different types of buildings. The system should work the same way regardless of whether the residence is an apartment in a multi-apartment building or

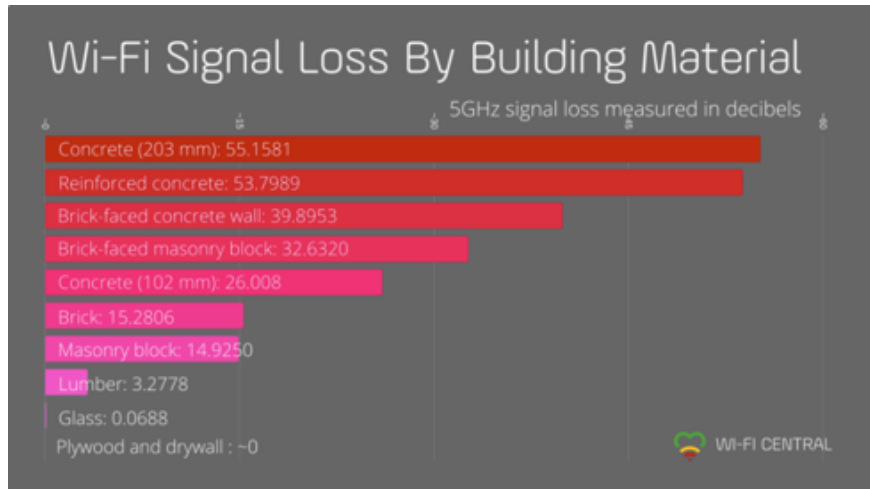


Figure 5: Signal attenuation through different materials, picture taken from [3]

standalone house. Only the collection of sufficient outside data might be an issue in some settings. We leave it to future research to confirm that the system achieves similar performance in apartment buildings.

5.1.2 Test execution

To ensure reproducibility, we conducted all experiments following the steps as described below.

We place all the observers on one plane. The target is also on that plane. For our experiments, we set the height of the plane at about 1m above the floor using wooden furniture to reduce signal attenuation. The variations of height between the different devices did not exceed 5cm and we assume their effects to be negligible. The Raspberry Pi was placed on a rolling cart and we used a mobile battery to power it.

An essential part in ensuring reproducibility was to transmit signals from the same set of locations on every run of the experiment: we sent a fixed number of transmissions from each location in a fixed set of locations. The locations inside were distributed in a grid with a margin of 1.5m between gridpoints. Outside we chose locations located on three separate rings around the house

at distances of 20cm, 50cm, and 3m respectively. On these rings the transmission locations were spaced 1.5m apart as well. A sketch of all transmission locations can also be found in Appendix B. We marked these locations with small strips of tape on the ground so we could reliably position the cart with the Raspberry Pi on top of it. Unfortunately, we had to leave out some of these gridpoints because of furniture that was in the way.

To ensure reproducibility, no people were present in the house during the data collection. The sleep timer in the transmission script allowed the person pushing the button on the Raspberry Pi to leave the house in time before the transmissions occurred. We leave it to future research to determine the effects of the presence of people on the performance of the system.

5.1.3 Variables and test runs

Overall, we performed three different runs of the experiment, as shown in Table 4.

1. High transmission power, observers just inside the outside walls.
2. Low transmission power, observers just inside the outside walls.
3. High transmission power, observers moved further inside.

We changed the position of the observers to determine the effect of their location on the performance of our classifier. The **Outer positions** had the observers placed in the four corners of the house with about 30cm separation to the wall. The **Inner positions** had the observers placed in the same formation, but closer to center of the house (approximately 4m away from the corner). A schematic of these positions is shown in Appendix C.

Table 4: Different experiment configurations

#	Tx power	Observer placement	Dataset name
Run 1	30dbm	Outer positions	dataset1
Run 2	15dbm	Outer positions	dataset2
Run 3	30dbm	Inner positions	dataset3

Table 5: Standardization factors of each dataset

#	μ	σ
dataset1	68.55	14.41
dataset2	69.71	18.34
dataset3	67.43	13.98

5.2 Measurements and results

When analyzing our datasets and performance of the machine-learning classifiers we use dataset1 for the baseline dataset and compare the results of the other ones to it.

5.2.1 Dataset

The three datasets contain around 8,000 measurements each. There is an imbalance towards the outside set of measurements as we took more measurements outside. Before the datasets are further processed they are standardized as described in 4.3.3; the standardization factors are shown in Table 5.

It is difficult to visualize the datasets, which are four dimensional; to get a general understanding of the shape we used the t-distributed Stochastic Neighbor Embedding (TSNE) algorithm for dimensionality reduction. We used the algorithm to reduce the dataset from four to two dimensions, and show the result in Figure 6. TSNE tries to identify similarities and cluster in the higher dimensional data and transfer them down to lower dimensions. Therefore, the absolute values at the lower dimension are relatively meaningless. Meaningful are the relative positions and clusters of the individual datapoints.

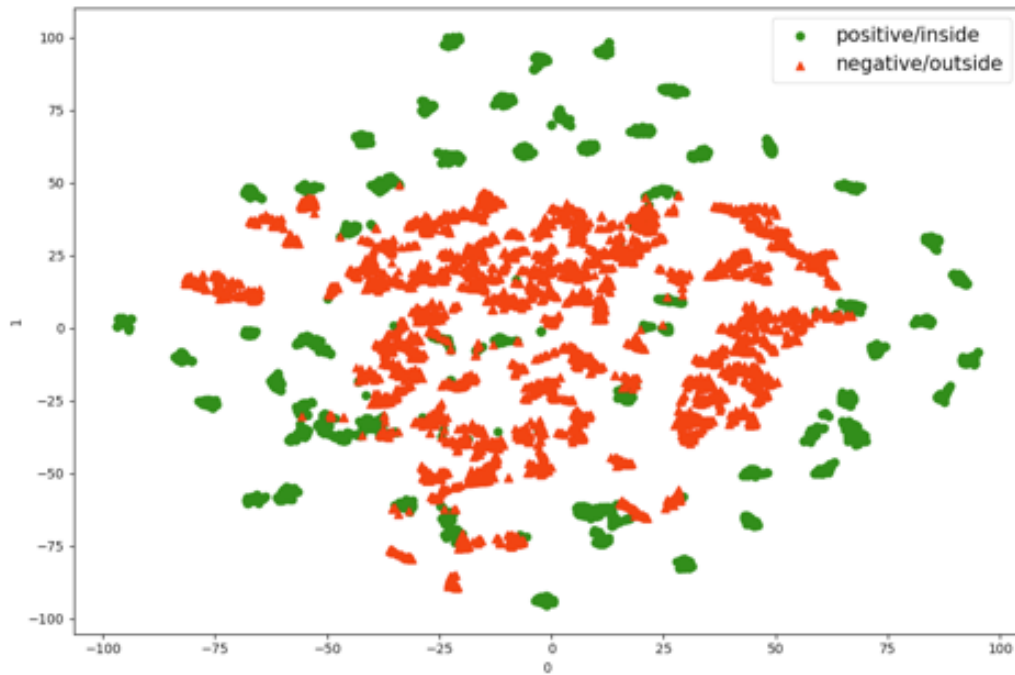


Figure 6: Result of TSNE dimensionality reductions

Interestingly, it is easy to see how the transmissions from the same location, especially on the inside, are clustered together. The clustered measurements seem to back up the claim from Section 4.1.2 that each location has its own unique RSSI ‘fingerprint’. It can also be seen right away that, while there is some overlap between the positive and negative samples, the data is at least to some degree separable. Just how separable the data is will be determined in the following sections.

5.2.2 Effects of evaluation methods

As noted above, we use k -fold cross validation and hold-out tests (2-fold cross validation) to evaluate our classifier. After working with our data we realized that the the parameters of the evaluation method make a substantial difference when using the data we collected. These effects of the eval-

uation methods are largely due to the nature of the data and how we collected it. We recorded 50 frames per location. Each transmission location has its own RSSI fingerprint, which means that readings recorded at the same location have similar RSSI values recorded by each observer and therefore form a cluster in the dataset. There are 160 locations in our dataset, although some of their ‘fingerprints’ will likely overlap. When looking at Figure 6 it becomes clear that overlaps were especially common for the inside locations and less so for the outside locations.

In its essence, k -fold cross validation splits the data into a training set and testing set k times. Now, if the training set includes a sample from each of the location clusters, the classifier should find it relatively easy to recognize a new sample from one of those clusters – because it will be a close neighbor to points seen during training. Classifying samples that have essentially been seen before is not a good indication of how well our system is able to distinguish inside from outside. If we shuffle the dataset before applying 10-fold cross validation, for example, each of the 10 subsets will contain a sample from almost every location cluster. Hence, the classifier is only asked to classify samples for which it has already seen close neighbors, and is therefore extremely effective (in our case scoring around 99.5% accuracy). The high accuracy is simply the result of the way we trained and tested the classifier.

This observation begs the question: what evaluation method should we use? To begin, we arrange the samples in the order they were recorded. Therefore, measurements that were taken at the same location will be adjacent in the dataset. If we perform a k -fold cross validation *without* shuffling the dataset – instead, chopping the sequence of samples into k subsequences – each of the subsets will largely contain points recorded at different locations. Thereby, in each iteration, the samples in the test set were recorded (mostly) at locations that are not included in the training set. Here, the classifier is actually tested on how well it can classify transmissions from locations

it hasn't seen before, which is a better indication of how well it may perform in a practical setting. While such a non-shuffled cross validation is not frequently performed in related research work, it is an effective way for us to test our classifier on locations that it has not seen before. For future research we suggest to record the location of each transmitted frame so that they can be separated more easily during the evaluation.

The question now is, how many folds we should use for the k -fold cross validation. If k is too small we may leave out large regions of training data (e.g., a whole side of the house), and if k is too large we run into the same issue that occurs when shuffling the data; nearly every location cluster is part of the training data. We can see how the choice of k affects the accuracy of the classifier in Figure 7. The accuracy increases quickly until $k = 70$, then it begins to stabilize and only increases slightly. With small k , we are missing rather large patches of locations in our training data – yet we test on those locations. As we use more folds, we leave out smaller location patches during training, giving the classifier a more complete training set. In the extreme case of $k = 160$, in theory we test on exactly one transmission location and train on the rest. However, due to some frames dropped in the data processing step (as described in Section 4.3.1) this situation is unlikely, as the folds will not align with the boundaries of the location clusters within the dataset.

For our evaluations and comparisons we chose $k = 80$, for two reasons: first, this is the value where the accuracy starts to level off in Figure 7; second, 80 is half of the number of location clusters (160) in our dataset. Therefore, with $k = 80$ we include at least one location cluster in each training set that had no samples included in the training data. Thereby we mimic a 'leave one location out' approach that should give a better indication of how well our classifier is able to distinguish inside from outside.

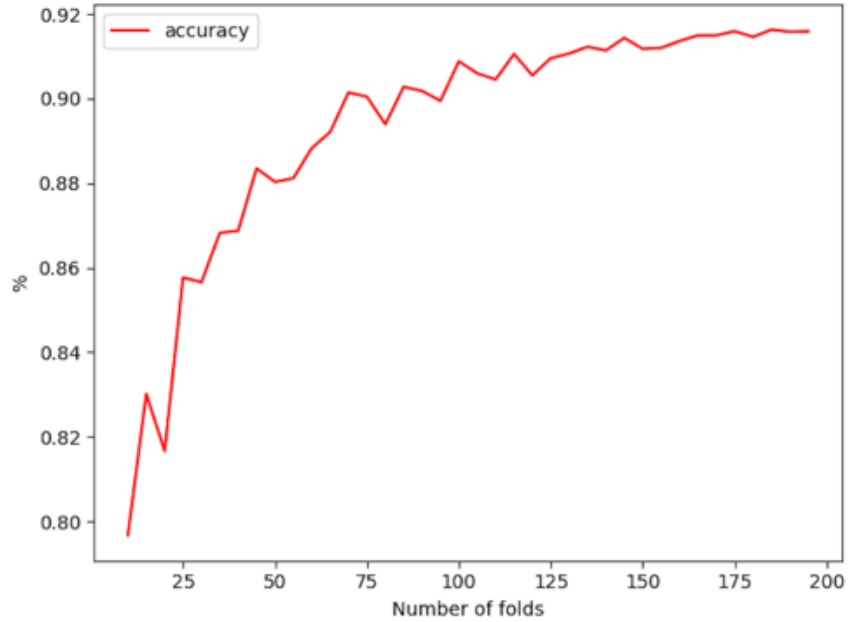


Figure 7: Accuracy against the number of folds used for non-shuffled k -fold cross validation with a SVM

5.2.3 Classifier performance

As mentioned in section 4.4.2 we are using a SVM with an rbf kernel as our baseline model. When running a 80-fold cross validation on dataset1, the SVM produced the following confusion matrix:

$TN = 2434$	$FP = 555$
$FN = 308$	$TP = 4839$

Therefore, the classifier achieved a precision of 0.894 and a balanced accuracy of 0.74. In the context of our experiments these values mean that the classifier predicts 74% of the overall samples, and 89.4% of the inside samples, correctly.

The ROC curve of the classifier produced running a 80-fold cross validation can be seen in Figure 8. The plot shows each of the 80 runs of the cross validation and the average of all the runs. The AUC of the ROC graph is a promising 0.97.

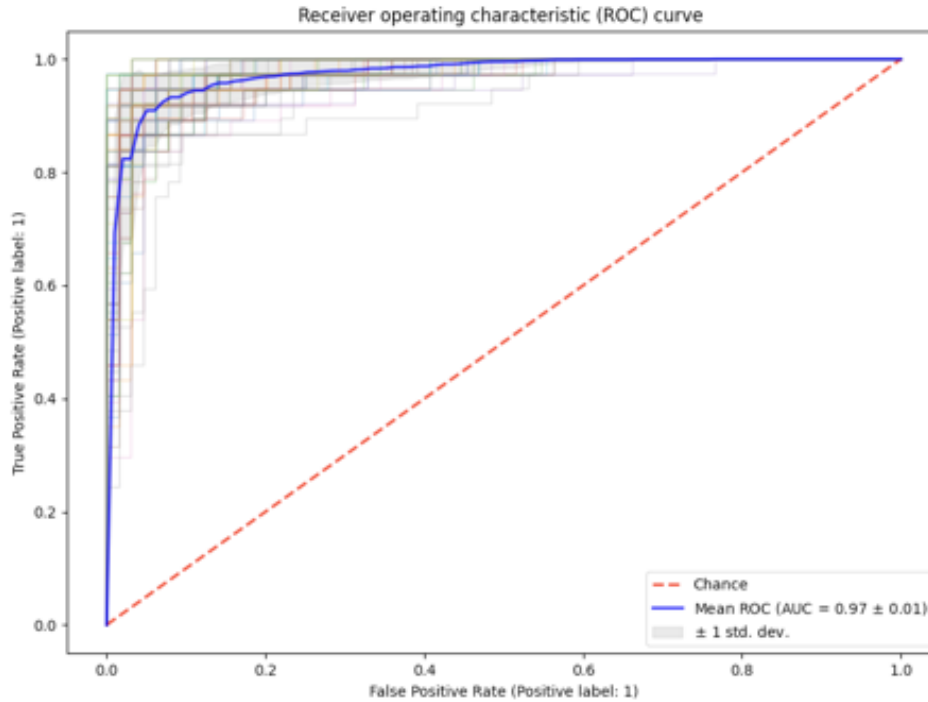


Figure 8: ROC curve produced by testing a SVM with rbf kernel using 10-fold cross validation on dataset1. The thin colored lines indicate the ROC curve of each fold.

Another interesting property of the classification to look at is the certainty the classifier has for each prediction. Most classifiers (including the three compared in this thesis) produce a probability as output. If the probability is above 0.5, the sample is predicted to belong to the positive class and if it is below 0.5 the sample is predicted to belong to the negative class. Plotting these prediction probabilities for the entire dataset gives a good idea of how certain the classifier is on predictions. The plot also provides an insight into the separability of the dataset. We created the prediction probability plot seen in Figure 9 by performing a 80-fold cross validation. The plot confirms what the numbers indicate as well. The classifier is more likely to classify a frame that was transmitted from the inside to belong to the outside class than the other way around. Therefore, we see more false negatives than false positives (compared to the amount of samples in each class) which is a desired behavior as described in 4.4.1.

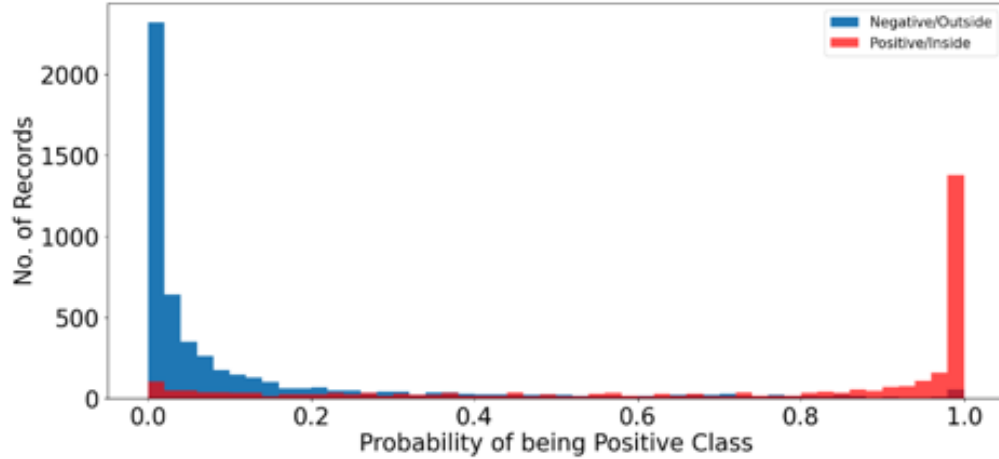


Figure 9: Prediction probabilities produced by testing a SVM with rbf kernel using a 80 cross validation

Table 6: Results the different classifiers achieve when running a 80-fold cross validation on dataset1

Classifier	bal. Accuracy	Precision	AUC
SVM	0.74	0.90	0.97
KNN	0.79	0.98	0.97
Random Forest	0.80	0.97	0.99

5.2.4 Classifier comparison

As mentioned in Section 4.4.2, we are comparing three different classifiers to see which one performs best on our data. We use dataset1 for the comparison and the results can be found in Table 6.

While the differences are subtle, the results show that for this dataset the Random Forest Classifier performs slightly better than the other two. Comparing the classifiers on the two other datasets gave similar results. It is hard to find a clear reason for the Random Forest classifier performing best.

We suspect, however, that the Random Forest classifier can capture more complexity. By using an ensemble of decision trees, it is able to apply more non-linearities to the data than, for example, a SVM. Thereby, it is more flexible in fitting to the data and can capture more complicated patterns.

5.2.5 Varying transmission power

As described in section 4.3.3, one goal of our system is to classify devices regardless of their transmission power. Ideally the system can be trained using a device with a transmission power of (for example) 30dBm and then classify a device with a transmission power of (for example) 15dBm. We try to accomplish such a classification by standardizing the data and leveraging the fact that differences in transmission power have a proportional effect on the RSSI readings of different locations [5].

To test whether our system is able to accomplish classifying different transmission power levels we use dataset1 and dataset2 that we recorded using a transmission power of 30dBm and 15dBm respectively. For this test we are using the Random Forest classifier as we showed that it is the best performing classifier for our data. We trained the classifier on dataset1 and then tested it on dataset2, and vice versa. By doing so the classifier only uses measurements of one transmission power level to classify the transmission done at a different power level. For these results to be meaningful we also had to adapt the standardization factors; instead of standardizing each dataset using its own μ and σ we used the μ and σ from the training set for the testing set as well. Thereby, we avoided using any prior knowledge about the testing set before classifying it.

The results of these experiments can be seen in the following table:

Training	Testing	bal. Accuracy	Precision	AUC
dataset1	dataset2	0.65	0.60	0.77
dataset2	dataset1	0.67	0.65	0.79

Comparing the above results with the results from Table 6 shows us that the system is still

distinguishing between the two classes, but less successfully than when just using the same transmission strength. It should be noted that these metrics were calculated using a different evaluation method, than in the previous sections. Here, we train and test on two entirely different datasets instead of using k -fold cross validation. The different evaluation method might at least partially explain the drop in performance. Further experiments are needed to confirm that the performance of the classifier suffers when using a different transmission power.

5.2.6 Different observer placement

The last variable that we changed during our experiments was the location of the observers. In dataset3 we used the *inner positions* for the observers as defined in Section 5.1.3. Using a Random Forest classifier we ran the same 80-fold cross validation on both dataset1 and dataset2, which were both recorded using the same transmission strength but different observer positions. The results of these experiments can be seen in the following table:

dataset	bal. Accuracy	Precision	AUC
dataset1	0.80	0.97	0.99
dataset3	0.82	0.98	0.99

As we can see the accuracy and precision improved marginally. It thereby seems that placing the observers further inwards improves the performance of the classifier. Notably the inner observer positions were in proximity to the brick fireplace and we theorize that it might have influenced our measurements. Another possible explanation for these results is that the observers in the inner positions are closer to all inside measurement locations. Thereby, locations on the inside might become less ambiguous, while locations close to the boundary stay as ambiguous as before.

5.2.7 One class vs Binary

As discussed in Section 4.4.3, we also investigated whether we can use a one-class classifier. Such a classifier would allow us to train using only data collected from the inside. Scikit-learn, the Python library we use for the machine-learning algorithm, only provides a one-class version of the SVM. KNN and Random Forest only work for binary classification problems. We evaluated this SVM on dataset1 using holdout validation with a test set size of 30%. That is, we trained the SVM only on the positive examples of the training set and tested it on both classes of the test set. The one-class SVM performed with a balanced accuracy of 0.61 and a precision of 0.56, which are rather poor; after all, random guessing would lead to an accuracy of 0.50. Unfortunately the one-class SVM did not support probability predictions and we could therefore not calculate the AUC. These results, especially the precision, are significantly worse than the ones produced by binary classifiers. We leave it to further research to find ways to increase this accuracy or develop techniques on how to work with limited outside training data.

5.2.8 Using three observers

As a final experiment, we wanted to see how the system performs when only data of three observers is used. We simulated using three observers by dropping the recordings made by one of the observers. The location of each observer can be seen in Appendix C. The results of dropping single observers are listed in Table 7.

We can see that the accuracy decreases slightly when using three observers. Leaving out each of the observers produces different results. Notably, the accuracy and precision almost stay almost unchanged when leaving out observer 3.

Table 7: Comparison of classifier performance when leaving out recordings of different observers. Results obtained using 80-fold cross validation with a Random Forest classifier on dataset 1.

Classifier	bal. Accuracy	Precision	AUC
All 4 observers	0.80	0.97	0.99
Without Obs. 1	0.75	0.91	0.99
Without Obs. 2	0.74	0.91	0.98
Without Obs. 3	0.78	0.96	0.99
Without Obs. 3	0.76	0.93	0.98

We can conclude from these observations, that while reducing the number of observers to 3 decreases the performance of the system, a three-observer system may still be feasible because the decrease is not substantial. Depending on the placement of the observers, just using three observers only affects the performance slightly.

6 Discussion and Conclusion

Not a lot of research has been done on the problem of indoor vs. outdoor classification. Therefore, there is no established way to evaluate the performance of such a classification system. In this thesis, we use non-shuffled 80-fold cross validation to mimic a leave-one-out evaluation method. While this might be an unconventional way to score a machine-learning classifier it is a better indicator of the performance of our system than using randomly sampled cross validation. In hindsight, we should have also recorded the transmission location, or what cluster each frame belongs to. Such labels would have allowed us to, for example, perform true leave-one-out evaluation or other, better suited evaluation methods.

Our results are also dependent on several other factors. One of these factors are the transmission locations. If we had chosen different locations (e.g., outside locations that are further or closer to the boundary) our results would have likely been different. Just how different results

would be in other environments will have to be determined by future research. Another factor that influenced our measurements was the experiment location. As seen in Figure 5 the attenuation of Wi-Fi signals heavily depends on the materials from which a house is built. Different building materials might produce different results. We theorize that a house made out of concrete would produce better results as there should be a greater distinction between inside and outside signal strengths. While the absolute values of the results might differ in other environments, the comparisons between different set-ups we drew should be generalizable.

Note also that our evaluation is based on an attempt to classify the location of a device from a single transmitted frame. Averaging the values over multiple transmitted frames might increase the accuracy of the system.

Because of all these factors, our results need to be interpreted in the context of the data collection and evaluation methods. We leave it to further research to determine how different environments influence the performance of the system.

For the system to be deployed in the ‘real world’ some further features will need to be developed and some questions answered. In a real implementation it would be useful to continue training while in operational mode. For example, transmissions from Smart Things known to be ‘inside’ could be used to refine the model over time. Continuous training may be necessary because the RSSI readings are affected by, for example, the arrangement of furniture in the residence. The new training data could be used to account for such changes in the environment. In such an approach, one could apply a function to give greater weight to newer training examples to ‘update’ the model over time. Future research should conduct long-term experiments to determine how robust the system is to changes in the environment. Other experiments should determine the effect of a multichannel environment.

We aimed to determine whether it is possible to build an ‘inside vs. outside’ RSSI-based classification system that could be used in private homes. We showed that it is possible to design such a system, but there is a need to explore several ways to improve the system and to better evaluate its performance. We hope that future research can answer some of these questions.

Acknowledgements

Many people helped me during the process of writing this thesis. First and foremost, my thesis advisor Prof. David Kotz who provided guidance, mentorship and always held my work to a high standard. A special thank you also goes to my co-advisor Dr. Beatrice Perez, who provided advice on the many issues that I ran into and made sure I was always on the right track. Whenever I needed specialized knowledge, tips or tricks on how to deal with various Wi-Fi problems Dr. Tim Pierson was always available and took the time to help me out, a big thanks for that. I could not have conducted my experiments without my friends and teammates Alex Lehr, Bobby Paré, Will Schultz and Jacob Hudgins who kindly allowed me to use their house. Another thank you goes to Joseph Hajjar, Gus Emmett and Mahalia Dalmage who helped me resolve various technical problems, correct many of my typos and always motivated me to keep on going. Lastly, thanks to all my other friends and family who always had my back and made this an enjoyable experience.

This research results from the SPLICE research program, supported by the National Science Foundation under award number CNS-1955805, and Grant number 2030859 to the Computing Research Association for the CIFellows Project. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors. Any mention of specific companies or products does not imply any endorsement by the authors, by their employers, or by the sponsors.

References

- [1] TP-link Archer T2U Plus. Online at <https://www.tp-link.com/us/home-networking/usb-adapter/archer-t2u-plus/>.
- [2] k-fold cross validation schematic. Online at <https://tex.stackexchange.com/questions/434358/draw-customized-table-with-tikz>, visited May 2021.
- [3] Signal attenuation comparison. Online at <https://eyenetworks.no/en/wifi-signal-loss-by-material>, visited May 2021.
- [4] Imran Ashraf, Soojung Hur, and Yongwan Park. Smartphone Sensor Based Indoor Positioning: Current Status, Opportunities, and Future Challenges. *Electronics*, 9(6), 2020. DOI 10.3390/electronics9060891.
- [5] Linsong Cheng and Jiliang Wang. How can I guard my AP? Non-intrusive user identification for mobile devices using WiFi signals. In *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, volume 05-08-July, pages 91–100, New York, NY, USA, 7 2016. Association for Computing Machinery. DOI 10.1145/2942358.2942373.
- [6] Waltenegus Dargie and Christian Poellabauer. *Fundamentals of Wireless Sensor Networks: Theory and Practice*. 1 2011. DOI 10.1002/9780470666388.
- [7] Marzieh Dashti, Simon Yiu, Siamak Yousefi, Fernando Perez-Cruz, and Holger Claussen. RSSI localization with Gaussian processes and tracking. In *2015 IEEE Globecom Workshops, GC Wkshps 2015 - Proceedings*, 2015. DOI 10.1109/GLOCOMW.2015.7414106.
- [8] Rajib Dey, Sayma Sultana, Afsaneh Razi, and Pamela J Wisniewski. Exploring Smart Home Device Use by Airbnb Hosts. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems, CHI EA '20*, page 1–8, New York, NY, USA, 2020. Association for Computing Machinery. DOI 10.1145/3334480.3382900.

- [9] R M Gomathi, G H S Krishna, E Brumancia, and Y M Dhas. A Survey on IoT Technologies, Evolution and Architecture. In *2018 International Conference on Computer, Communication, and Signal Processing (ICCCSP)*, pages 1–5, 2 2018. DOI 10.1109/ICCCSP.2018.8452820.
- [10] Chih-Ning Huang and Chia-Tai Chan. ZigBee-based indoor location system by k-nearest neighbor algorithm with weighted RSSI. *Procedia Computer Science*, v.5 pages 58–65, 2011. DOI <https://doi.org/10.1016/j.procs.2011.07.010>.
- [11] Rayana H. Jaafar and Samer S. Saab. A Neural Network Approach for Indoor Fingerprinting-Based Localization. *2018 9th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2018*, pages 537–542, 2018. DOI 10.1109/UEMCON.2018.8796646.
- [12] Jin Woo Jang and Song Nam Hong. Indoor Localization with WiFi Fingerprinting Using Convolutional Neural Network. *International Conference on Ubiquitous and Future Networks, ICUFN*, v.2018-July pages 753–758, 2018. DOI 10.1109/ICUFN.2018.8436598.
- [13] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. SpotFi: Decimeter Level Localization Using WiFi. *Computer Communication Review*, 45(4) pages 269–282, 2015. DOI 10.1145/2785956.2787487.
- [14] David Kotz and Travis Peters. Challenges to Ensuring Human Safety throughout the Life-Cycle of Smart Environments. In *Proceedings of the 1st ACM Workshop on the Internet of Safe Things, SafeThings' 17*, page 1–7, New York, NY, USA, 2017. Association for Computing Machinery. DOI 10.1145/3137003.3137012.
- [15] Pavel Kriz, Filip Maly, and Tomas Kozel. Improving Indoor Localization Using Bluetooth Low Energy Beacons. *Mobile Information Systems*, v.2016 page 2083094, 2016. DOI 10.1155/2016/2083094.
- [16] Swarun Kumar, Stephanie Gil, Dina Katabi, and Daniela Rus. Accurate Indoor Localization with Zero Start-up Cost. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking, MobiCom '14*, pages 483–494, New York, NY, USA, 2014. ACM. DOI 10.1145/2639108.2639142.

- [17] Marko Malajner, Peter Planinšič, and Dušan Gleich. Angle of Arrival Estimation Using RSSI and Omnidirectional Rotatable Antennas. *IEEE Sensors Journal - IEEE SENS J*, v.12 pages 1950–1957, 2012. DOI 10.1109/JSEN.2011.2182046.
- [18] Rajalakshmi Nandakumar, Vikram Iyer, and Shyamnath Gollakota. 3D localization for subcentimeter-sized devices. *Communications of the ACM*, 64(3) pages 117–125, 3 2021. DOI 10.1145/3446782.
- [19] Jun Qi and Guo Ping Liu. A robust high-accuracy ultrasound indoor positioning system based on a wireless sensor network. *Sensors (Switzerland)*, 17(11), 2017. DOI 10.3390/s17112554.
- [20] E Sánchez, M Botsch, B Huber, and A García. High precision indoor positioning by means of LiDAR. In *2019 DGON Inertial Sensors and Systems (ISS)*, pages 1–20, 2019. DOI 10.1109/ISS46986.2019.8943731.
- [21] Xudong Song, Xiaochen Fan, Chaocan Xiang, Qianwen Ye, Leyu Liu, Zumin Wang, Xi-angjian He, Ning Yang, and Gengfa Fang. A Novel Convolutional Neural Network Based Indoor Localization Framework with WiFi Fingerprinting. *IEEE Access*, v.7 pages 110698–110709, 2019. DOI 10.1109/ACCESS.2019.2933921.
- [22] Deepak Vasisht, Swarun Kumar, and Dina Katabi. Decimeter-Level Localization with a Single WiFi Access Point This paper is included in the Proceedings of the Decimeter-Level Localization with a Single WiFi Access Point. *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation*, pages 165–178, 2016.
- [23] Jingjing Wang and Joongoo Park. An Enhanced Indoor Positioning Algorithm Based on Fingerprint Using Fine-Grained CSI and RSSI Measurements of IEEE 802.11n WLAN. *Sensors (Basel, Switzerland)*, 21(8), 4 2021. DOI 10.3390/s21082769.
- [24] Roy Want, Wei Wang, and Stan Chesnutt. Accurate Indoor Location for the IoT. *Computer*, 51(8) pages 66–70, 8 2018. DOI 10.1109/MC.2018.3191259.
- [25] J Xiao, K Wu, Y Yi, L Wang, and L M Ni. Pilot: Passive Device-Free Indoor Localization Using Channel State Information. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 236–245, 7 2013. DOI 10.1109/ICDCS.2013.49.

- [26] Jie Xiong and Kyle Jamieson. ArrayTrack: A fine-grained indoor location system. *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2013*, pages 71–84, 2013.
- [27] Yuchen Yang, Longfei Wu, Guisheng Yin, Lijie Li, and Hongbin Zhao. A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet of Things Journal*, 4(5) pages 1250–1258, 10 2017. DOI 10.1109/JIOT.2017.2694844.
- [28] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. A Survey of Indoor Localization Systems and Technologies. *IEEE Communications Surveys and Tutorials*, 21(3) pages 2568–2599, 2019. DOI 10.1109/COMST.2019.2911558.

Appendices

A Observer Code

```
from scapy.all import *
from gpiozero import Button
import time

iface = 'wlan0mon'
curr_id_fp = '/home/kali/curr_id'

def read_id():
    # Load id from text file
    with open(curr_id_fp, 'r') as in_file:
        x = in_file.read()
        return int(x.strip())

def write_id(new_id):
    # Write id to text file
    with open(curr_id_fp, 'w') as out_file:
        out_file.write(str(new_id))

curr_id = read_id()
to_send = 50

rx_addr = '01:02:03:04:05:06'

button = Button(3)

print('ready to transmit')

while True:
    button.wait_for_press()
    button.wait_for_release()
    time.sleep(5) # Time to walk away
    for i in range(curr_id, curr_id + to_send):
        pkt = Dot11(addr2=rx_addr, type=2)/Raw(load=f'{i}')
        sendpkt = RadioTap()/pkt

        sendp(sendpkt, iface=iface, verbose=1, inter=0.01)

    curr_id += to_send
    write_id(curr_id)
```

B Floor Plan

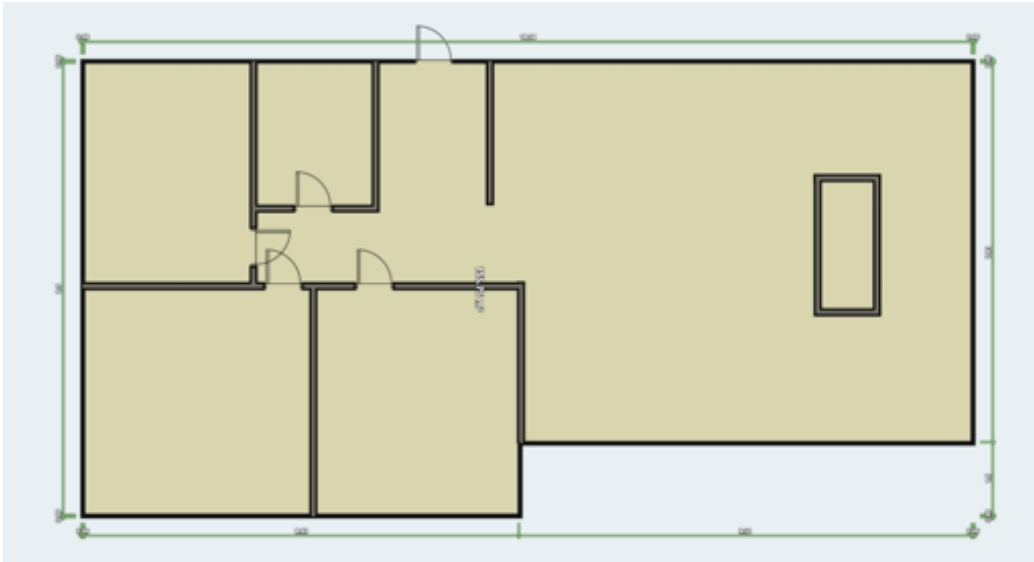


Figure 10: Floor Plan

C Observer Positions

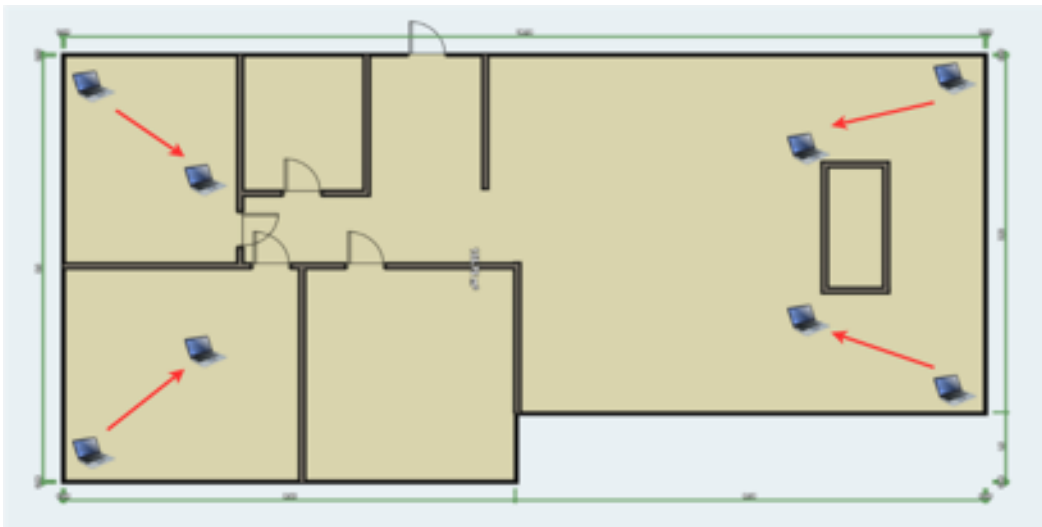


Figure 11: Inner and outer observer positions. The observers are numbered clockwise, with Observer No. 1 in the top right corner.