HALMSTAD UNIVERSITY

Master's Degree

MASTER THESIS

# Providing Location Based Security to an Unencrypted WiFi Network

Final Report

Anders Nylander and Henry Andersson

Department of Computer Science

Halmstad University, June 9, 2017–version 1.0

## ABSTRACT

For this project, we investigate different methods of adding location-based security to a WLAN network. A literature review is done to confirm the current state-of-the-art on the subject, and we scrutinize the available methods based on practicality, security, and simplicity. We then further delve into a few specific methods with good properties based on the prior review, to confirm if these are suitable for a proof-of-principle implementation. Finally, if a suitable method is found, we develop the proof-of-principle to show that the system can work in practice.

# ACKNOWLEDGEMENTS

Thanks to Bengt Eriksson, Magnus Porsgaard and Omid Manikhi of HMS Industrial Networks (HMS) for lending technical assistance.

Professor Alexey Vinel for academic support and guidance.

# CONTENTS

# LIST OF FIGURES

# ACRONYMS

**RSS** Received Signal Strength

**RSSI** Received Signal Strength Indicator

**ToF** Time of Flight

**RTT** Round Trip Time

**ToA** Time of Arrival

**TDoA** Time Difference of Arrival

**dBm** Decibels per milliwatt

**LUT** Look-Up Table

**HMS** HMS Industrial Networks

**WPA** WiFi Protected Access

**WPS** WiFi Protected Setup

**WEP** Wired Equivalent Privacy

**BT** Bluetooth

**PEAP** Protected Extensible Authentication Protocol

**WLAN** Wireless Local Area Network

**MIMO** Multiple Input Multiple Output

**CSMA/CA** Carrier Sense Multiple Access with Collision Avoidance

Part I

# INTRODUCTION

# 1 INTRODUCTION

In many industries today, WiFi have become the norm for enabling employees access to the company networks. At these industry locations, typically only the employees have physical access to the location, as it's usually protected by various measures like fences, card-and-pin security locks, and others. These physical barriers mean little to WiFi and other RF signals that can typically pass through most materials unhindered, and as a result, a company's WiFi network is usually exposed to more than just the company's building itself. In these cases, companies have had little choice but to protect their networks using data transmission encryption schemes like WPA, at the cost of usability of the network, but this means the network is still accessible from outside the premises; All that is required for access to the WiFi network is the proper authentication credentials e.g. WPA login details.

An available alternative to password by WPA is WPS with push button, but due to the nature of embedded industrial networks this is impossible to apply. If the module responsible for Wireless Local Area Network (WLAN) is built in as a part of a larger device it is impossible to access a button on this device. This is also the case for our project, the device that HMS provides is meant to be built in as a part of a larger system. This is where our project is accommodating, a way to allow instant access for everyone that have physical access.

This raises a question: Is there a way to use the physical location based security measures, e.g. localization, to provide a layer of security in-place of, or perhaps in-addition to, WPA? For this paper we

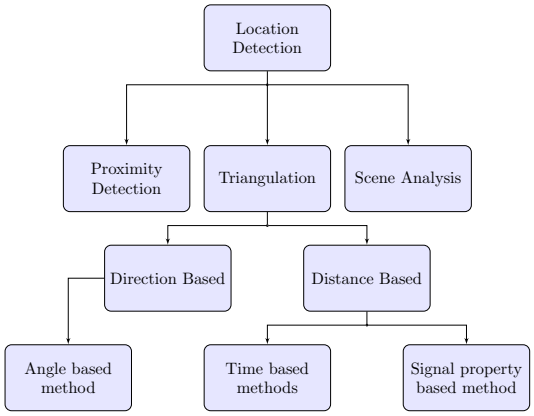


Figure 1: Location detection based classification.[10]

will look at various means of localization through the use of RF signals, what the curren state-of-the-art research in this area can tell us, and use this as the basis for this project to provide WiFi with another layer of security, that potentially being localization.

This paper has been divided into chapters: In chapter 2 we will elaborate on the problem this project intends to address and outline the specific requirements placed on the solution. In chapter 3 we briefly introduce some of the software used in this project, and give a brief explanation of the WLAN protocols we've explored. In chapter 4 we discuss existing methods for distance estimation and localization, detail some of the research being done for these, and assess their applicability to the problem statement. In chapter 5 we take a closer look at the different methods we experimented with, and the solution we arrived at. In chapter 6 we present our acheived results. In chapter 7 we analyze the results and discuss the outcome of the project. In chapter 8 we make our concluding remarks.

Part II

# MAIN CONTENT

# 2 PROBLEM FORMULATION

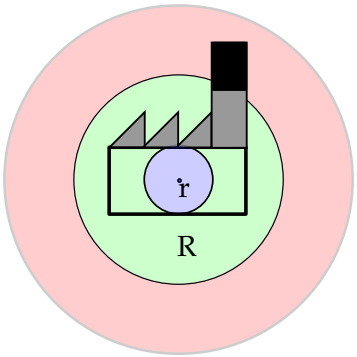


Figure 2: A figure describing the problem. r (blue circle) is the authentication radius, R (green circle) is the access radius, and the red circle represents the unauthorized area from where access to the network should be actively denied. The objects are not to scale.

As can be seen in figure 2, this security system will estimate distance between access point and node, and use this to determine when and how a node should be able access a network, with the requirements:

- Allow authorized personell access to the wireless network, where authorized personell are defined as people with physical access to be within the networks' maximum allowed authentication radius (r),
- Deny access to nodes attempting access from outside the networks' maximum allowed authentication radius (r),
- Maintain access for authenticated nodes that remain within the networks' maximum allowed access radius (R),
- Drop access for authenticated nodes that leave the maximum allowed access radius (R)
- Use the existing hardware offered in products from HMS.
- Do not only use password-based authentication such as Wired Equivalent Privacy (WEP) or WiFi Protected Access (WPA), or pushbutton style security like WiFi Protected Setup (WPS) to leverage security. Initially, the solution should work by itself without any additional security features.

- Deny access to network for nodes that attempt to circumvent the above restrictions by various means such as amplified signal, timing fraud, and proxy fraud.

In addition to these requirements, there are several other requirements that we want to fulfill, but may be too complex to accomplish within the project timespan:

- Try not to use more than one antenna to accomplish the goal (this would exclude localization-based solutions such as trilateration),
- Do not use specialized hardware besides what is already available inside HMS products,
- Produced code must be in c/c++ to adhere to HMS standards,
- Find a simple solution to the problem.

To aid in the development of the project, HMS will provide us with the necessary development tools, which include an EVK-W262U-00 evaluation kit from u-blox[19].
Based on these requirements, our problem becomes thus:
How can we use the physical layer properties of a WLAN to provide location-based security to a network?
Are there other ways to accomplish the same goal in a simpler manner?

# 3 BACKGROUND

To be able to understand the problem it is necessary to have some basic background information. In this project we needed to use new tools that that were crucial during both experimentation and implementation phase.

## 3.1 MBED

mbed[3] is an operating system intended for devices connected to the internet. The mbed operating system is designed for the ARM Cortex-M microprocessors. In our case we use an ARM Cortex M4-based microprocessor, STM32F429.

There are several different versions of the mbed platform, the most recent one at the time of writing is mbed OS 5. The hardware, supplied by HMS (Anybus Wireless Bolt, Wireless Bridge 2), uses several mbed software modules to enable functionality in the chipset. These modules are only supported in mbed OS 3 and mbed OS 5.

A big issue with mbed is the lack of compatibility between different versions. For example the two versions that we are interested in, mbed OS 3 and mbed OS 5, have completely independent APIs. This makes it very hard to upgrade from mbed OS 3 to mbed OS 5. It is also a bit hard to get started with mbed due to this, it can be difficult to find API and example to the right version of mbed.

When using mbed OS extra software is packaged in modules. To be able to manage all modules and building a project one needs to use a build tool. The most common one for mbed is yotta.

## 3.2 YOTTA

yotta[4] is a build tool intended for mbed OS 3, in later version of mbed different tools are also available.

yotta is built on the use of modules, a module can be added to yotta by either installing it or manually configuring a project module file and then allowing yotta to re-interpret the project dependencies. This allows the users to specify a certain version of a module.

When building a project in yotta it is necessary to select a target

Figure 3: A Debugger for ARM Cortex devices. source: http://se.farnell.com/stmicroelectronics/st-link-v2/icd-programmer-for-stm8-stm32/dp/1892523

platform, the target being the hardware platform that the code is intended to run on. This is one of the big benefits of using the mbed OS and yotta, created code can supports several different microprocessors and chipsets. Only need to change the specified target before building the project.

## 3.3 GIT

git[11] is a version control management tool, it allows the user to track the history of a project by collecting all changes into specific *commits*, saved states of the project that can be returned to or modified later on. It is useful for all scales of projects, from small one-man operations to large corporate projects where many participants contribute to the same files.

## 3.4 OPENOCD AND ARM TOOLCHAIN

OpenOCD[15] is a development tool for enabling the use of a debugger, in the case of our project we have used both a Segger j-link and a ST-link v2 to perform our development. OpenOCD is used to open a serial I/O channel to the target that can be used to flash, erase and troubleshoot the device. This is only a part of the information that we need during our development.

To compile our code for the intended ARM processor we use the GNU ARM Embedded Toolchain. The toolchain gives us vital tools

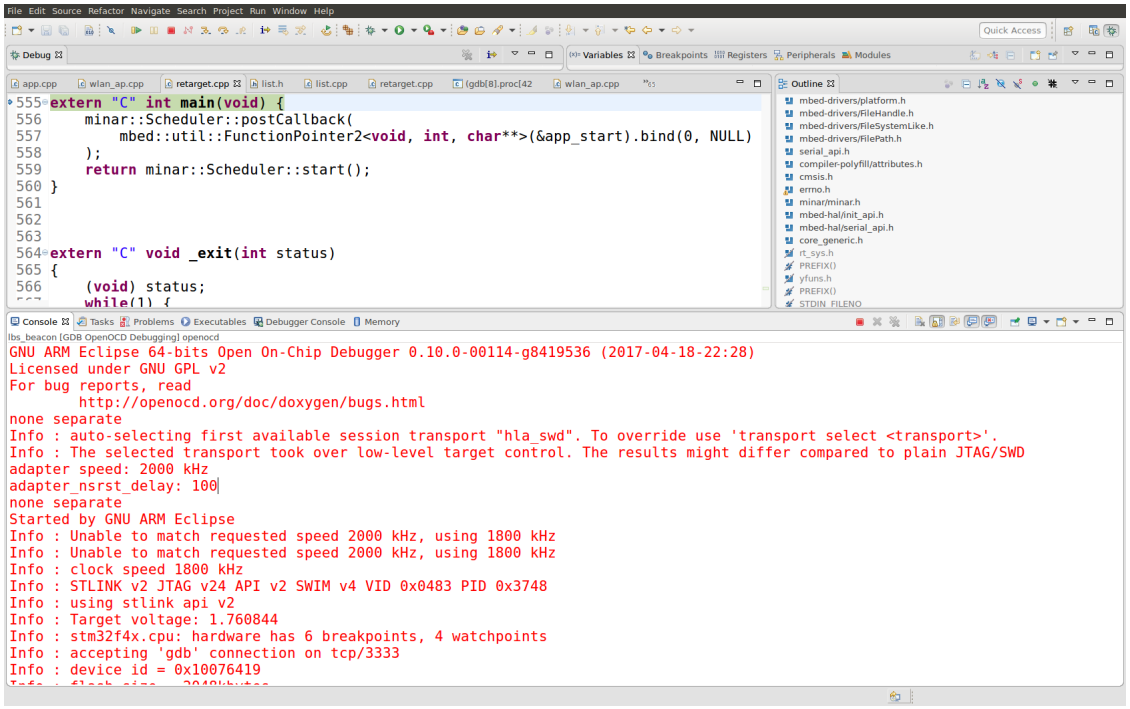


Figure 4: Screenshot of our working environment, including OpenOCD debugging

such as.

- gcc, makes it possible to compile our code.
- gdb, make it possible to debug a running program.
- objcopy, to convert .ELF files into .BIN files, preparing the compiled program for the target platform.

To combine all of this together into a simple to use development environment we use Eclipse[9]. In the Eclipse development environment there are available plugins that handle the process of compiling, erasing, flashing and debugging in an automated fashion. This by calling GNU ARM Embedded Toolchain[1] to compile the code, OpenOCD to erase and flash the new compiled program and calling the ARM Embedded Toolchain to start a GDB server to receive debug messages. You always get a gdb console that can be used to print out debug messages.

## 3.5 WLAN AND BT

In WLAN, two of the most common methods for devices connecting is the protocols WLAN and Bluetooth. Both of these operate on similar frequencies ( 2.4GHz), but differ in how they allow many devices to access the medium simultaneously. Bluetooth connections do this by rapidly switching between many different channels, the order of the channels decided based on a secret known only to each device pairing.

WLAN connections on the other hand, use what is known as Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), meaning they listen to the channel first for a period of time, then attempt to send a data packet after confirming the channel is free, and waits for an acknowledgement packet from the access point. If the acknowledgement does not arrive in a timely manner, it assumes the packet collided, and enters a binary exponential backoff before attempting to retransmit.

During this project we also get insight of the usage of BT and WLAN in embedded systems. For this chipset BT will be handled by the chipset driver from ublox. While WLAN adopts the common library Light Weight IP to manage TCP/IP stacks. LWIP is a common industry standard for using TCP/IP on embedded platforms.

## 3.6 PROJECT PLAN

This project was divided into six phases based on our initial projections. These can be distinguished as: investigation phase, experimentation phase, selection phase, implementation phase, testing phase, presentation phase.

- In the investigation phase, we studied the literature to deepen our understanding of the current state-of-the-art within the subject area.
- During the experimentation phase, we used the provided hardware equipment to more deeply evaluate a few select methods to better determine the feasibility of these methods.
- In the selection phase, based on earlier evaluations, a method most suited for implementation is chosen based on applicability, feasibility, security, simplicity, and economy.
- During the implementation phase, work to implement a working Proof Of Principle (POP) for the solution based on the earlier findings.
- During the testing phase the product will be tested in accordance with a test specification. The milestone for this phase is the product passing the tests laid out by the test specification.
- The final review phase is where we'll present our findings within the subject area, present and hand over the working POP to HMS and finalize our thesis.

These phases have to a certain extent overlap each other, but put simply, the investigation phase started in January,the experimentation phase started in February and ended in March, selection phase started in March and ended in April, Implementation phase lasted the entire duration of April, leaving May for testing, and the presentation phase in accordance with the project course schedule. The investigation phase technically does not have an end, since new papers related to the problem may be published during the course of this project to shed new light on the subject.

# 4 LITERATURE REVIEW

In this chapter we review some of the known methods for indoor distance estimation and localization techniques, assess their applicability and suitability for our project based on some given criteria such as accuracy, security, complexity, requirements and cost.

## 4.1 RECEIVED SIGNAL STRENGTH

One of the most basic way of determining distance between nodes in wireless communications is done by measuring the RSS. This makes several assumptions about the nodes: Both nodes use a known antenna type where properties such as directivity, direction and amplification factor are known to both parties. Assuming both nodes have no particular directivity and a constant, unchanging amplification, the distance between them can be calculated.

$$\text{RSSI} = -(10 \times n) \log_{10}(d) - A \tag{1}$$

In equation 1, RSSI is the RSS indicator measured in Decibels per milliwatt (dBm), $n$ is the signal propagation constant or exponent, $d$ is the relative distance between nodes, and $A$ is a reference RSS of a fixed distance between nodes. This is however a naïve approach that assumes the other node can be trusted. The other node could be lying about the type of antenna used, and infact be using a much stronger antenna, boosting its signal and appear to be much closer than it really is. As such, this technique, by itself, is deemed insufficient in a scenario where security is vital. For more in-depth information about RSS, see [8].

## 4.2 TIME OF FLIGHT

There are two well-known examples of how ToF is used to determine distance: Time of Arrival (ToA) and RTT. In ToA, both parties share a common, high-precision time base, and distance between nodes is estimated by comparing the time of transmission to when it arrives. The distance can then be calculated:

$$d = c \cdot t_p \tag{2}$$

In equation 2, $c$ is the propagation speed in a given medium, and $t_p$ the propagation time between the nodes. However, as this technique relies on both entities sharing a common timebase, and the

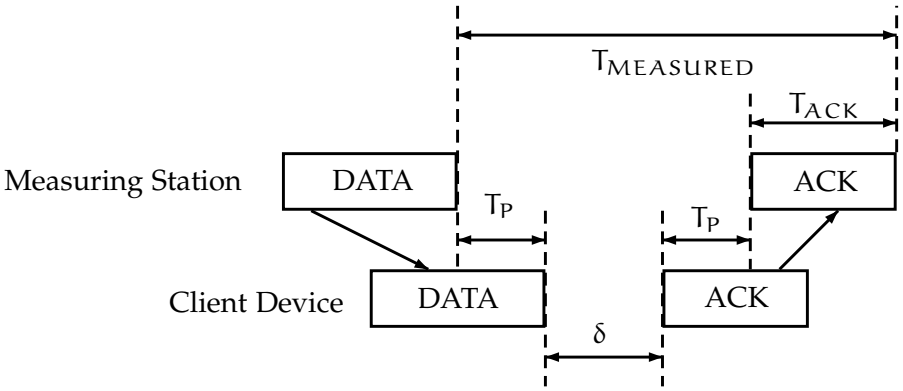


Figure 5: A figure describing time of flight using the Roundtrip Method. $T_p$ is the signal propagation time, δ is the client system response time, and $T_{ACK}$ is the measuring station's response acknowledgement time.

other entity needs to be trusted, the validity of the results cannot be guaranteed, as it's possible the sender might provide an intentionally incorrect time value to appear closer to, or further away than it really is.

In the other example, RTT, only one node needs to measure time, as the distance is estimated based on the time taken for a message to reach the other node and the reply sent back. The equation here is equally simple as before, just multiply $t_p$ by 2 and divide d by 2 to get the result. However, this technique is also not trustworthy; If the frequency of the RTT measurements are known and can be predicted, a malicious entity attempting to access the system could start sending the reply before the initial message has been received, in order to appear closer, or delay the response to appear farther away.

However, there is a major problem with using ToF in practice in the context of this project, that must not be ignored: The propagation speed of RF signals is the speed of light, while the maximum effective range of WLAN tend to not exceed 100 meters. This leads to a maximum roundtrip time, not counting the remote system response time, of about

$$\frac{2 * 100}{3 \times 10^8} = 666ns \tag{3}$$

meaning we need a high precision clock capable of measuring time on about the nanosecond scale to accurately measure distance when using ToF. Even higher precision is required for the more typical scenario where the distance between nodes is between 3 to 20 meters. [16]

Another method of time-based distance determination is by using Time Difference of Arrival (TDoA). In this technique, multiple radio receivers are used to intercept signals from the node to be located, and the arrival time difference of a packet between the receivers is

used to estimate possible points of origin. This technique is more well known as multilateration. This technique holds the advantage of not requiring the remote system to provide any particular function in order to estimate distance, or position, as all the necessary calculations are done by the system controlling the receivers, but is disadvantaged by requiring the use of multiple receivers, adding additional cost and complexity to the system[12].

## 4.3 DISTANCE BOUNDING

Distance bounding takes advantage of cryptography to ensure a response cannot arrive sooner than expected when using ToF with RTT. Assume there is a cryptographically secure function $G(x)$ that takes a random number or a string of characters as input and generates a unique answer $y$ for each input. In distance bounding, the access point would send out $x$, and expect $y$ in return. Measuring the time with RTT, and assuming $y$ was correct, the distance could be measured. There are however, some issues with this solution; Namely, the remote unit whose' location distance needs to be authenticated needs access to $G(x)$, and $G(x)$ needs to generate $y$ within a fixed amount of time. The nature of modern WLAN devices, where many different microprocessors are used would lead to variations in the response time if $G(x)$ returns a response faster with higher performance processors.

It may be possible to work around this weakness by simplifying the burden on the remote node using *pre-commitment* or *pre-computation* protocols. In this variation of the system, the access point sends out a challenge with a pre-determined number of answers known to the remote node in the form of a Look-Up Table (LUT). Another variation of the system would have the access point generate a challenge string, the remote node generate a response string. After receiving the response string, the access point sends the challenge string, and the remote node performs a simple XOR operation to calculate the response. This would significantly cut down the processing burden on the remote node. However, as demonstrated in [7], a distance bounding scheme can be susceptible to attacks that exploits the latency introduced in the physical and packet layers to mask its' own distance. Therefore, a separate channel would need to be reserved for the timed exchange between nodes. [2]

## 4.4 RF FINGERPRINTING

Fingerprinting is the ability to recognize a specific node only based on its signals analog properties. There exist multiple ways to detect the fingerprint, one of which will now be briefly explained.
One method of generating a fingerprinft is described by Bonne and Capkun[14]. It builds on the extraction of the signal transient which can be either the start of a transmission or the beginning of a packet. The transient is the part where the signal becomes more powerful than the background noise.
The main problem with fingerprinting is that the analog signal pattern that is recognised and classified as a fingerprint can change. In some cases the fingerprint can be heavily dependent on the surrounding environment. If some furniture were to change place the fingerprint characteristics would also change, making it hard or impossible to identify the device using the old fingerprint[5]. This is the main drawback of fingerprinting, but there are also more problems regarding security called "wormhole attacks".

## 4.5 ANGLE OF ARRIVAL

Angle of arrival measures the angle of the incoming signal by using many additional Multiple Input Multiple Output (MIMO) antennas, measuring the time delay between received signals and thus calculating the angle. This system works well when several conditions are fulfilled, such as large open spaces and many antennas. The system is particularly prone to errors in indoor scenarios where signals reflect and bounce off of various materials, causing the angle of a signals' arrival to not always reflect the direct path between nodes. [6]

## 4.6 TRILATERATION

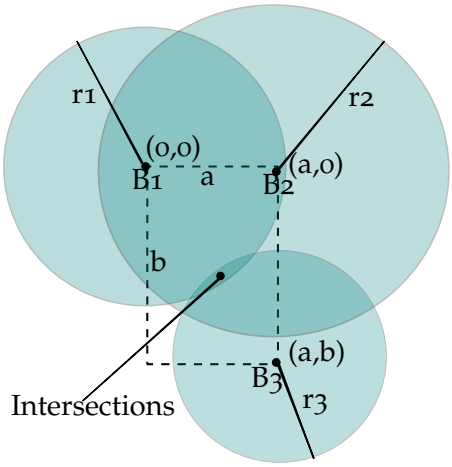


Figure 6: A figure describing Trilateration.

Trilateration is the process of determining absolute or relative locations based on measured distances between a point and several measuring beacons. This technique can be combined with methods for distance estimation such as RSSI and ToF to accurately determine the point of origin for an incoming signal, with the main drawback being that it requires more hardware, and more complex calculations.

## 4.7 CHRONOS

Chronos is a recently developed system that uses a MIMO WLAN antenna to accurately determine the location of clients to within tens of centimeters, from a single physical access point. It estimates the distance between access point and client by hopping across multiple frequency bands. By measuring the received signal phase offset across all measured channels, it's possible to narrow down the distance between client and access point by utilizing the Chinese Remainder Theorem[20]. This system is not without its' drawbacks however: Requiring the use of many channels, the ability to accurately measure phase offsets of incoming packets, and utilizing multiple antennas in a MIMO setup all add to the complexity of the system, and makes the system incompatible with clients that haven't been developed specifically to support Chronos.

# 5 METHOD

## 5.1 DESIGN CHOICES

Regardless of the intended technique we need to be able to control our hardware, in our case a ODIN-W2[18] chipset from u-blox[17]. Initially, we were given evaluation kits based on the hardware used in existing products from HMS. Specifically, these were the u-blox ODIN-W2 Evaluation Kit (version no longer in production). These allowed us to evaluate the hardware functionality of the ODIN-W2 module via AT commands, but we were limited to the firmware blobs provided by u-blox and the AT commands available. After an evaluation period, we were able to conclude that these kits were insufficient for the task of this project. A meeting with HMS later, we moved to new hardware based on the ODIN-W2.

That is, HMS own developed circuit board based on the ODIN-W2, reprogrammable over the JTAG interface we would be able to flash these devices with our own custom firmware, circumventing the problems we forecasted with using the evaluation boards. This required a change in our development platform.

To develop firmware for this hardware we use yotta to build the firmware, while using u-blox own proprietary closed source drivers for low level hardware control. We decided to develop our code using mbed OS 5. This due to the fact that mbed OS 5 is a more mature OS that mbed OS 3.

During the start of development, there was not yet any official support for mbed OS 5 in the u-blox binary blob, so yotta+mbed OS 3 was our initial choice. This caused us problems until official support for mbed OS 5 landed midway through the project.

We used OpenOCD to flash the target hardware. To receive feedback from the hardware, the firmware was modified to support sending serial data back over the JTAG interface. This was accomplished by adding an existing library (Trace from the µOS++ IIIe project[13]) and modifying it for our hardware.

Being able to debug our firmware using hardware debuggers gave us a huge advantage, but we were also wary of the potential impact that placing our CPU into debug mode would have. Because of this,

there was a need to verify that the timer used would be able to operate at the intended resolution, and so we experimented with different methods of storing this information:

- Sending the timer values as packets over the network
- Storing the timer values as data on the built-in flash memory
- Sending the timer values over the serial interface provided by the debugger to the host machine.

In the end, sending packets over the network and sending data to the host machine via serial interface allowed us to verify the proper operation of the hardware timer.

## 5.2 EXPERIMENTATION PHASE

### 5.2.1 *Time of Flight*

ToF was initially implemented under the assumption that only a single trusted node (the Access Point) is available. Experiments were done to measure the RTT of packets over the network to estimate the distance and get an idea for the overhead involved in this method, over both WLAN and BT protocols. To measure the RTT we used a built in timer in the CPU.

For a CPU that clocks up to 168MHz, this meant that we needed to use a 32-bit timer for our timing measurements if we wanted to verify the proper resolution of the timer with reasonable accuracy (168 million ticks per 1 second period, for instance). However, on the ODIN-W2 implementation, both the 32-bit timers are in use by the system and cannot be reprogrammed without fatal results. The solution to this problem was to configure 2 of the 16-bit timers into a master-slave relationship. Each time the master timer overflows, it generates an event that counts up the slave timer. Combining the results from both timers effectively gives us a 32-bit timer.

Activating two of the unused hardware timers on the STM32F429 processor allowed us to do time measurements synced to the CPU clock frequency. With a CPU clock frequency of 168 MHz this gives us a resolution of $\approx 5.96ns$ per tick. Compare this to the propagation speed of radio waves in free space ( $\approx 3.33ns$ per meter) and this gives us a theoretical max precision of within 1.79 meters of the true distance between access point and connecting node when implementing a ToF based solution for distance estimation. This can be compared with the preliminary requirements of being able to distinguish between a node remaining connected from different distances.

| Methods | Average | Std. Dev |
|---|---|---|
| ToF(WL) | 1.673ms | 0.1340ms |
| ToF(BTPC) | 3.543s | 1.2513s |
| ToF(BTSP) | 1.643s | 1.0671s |

Figure 7: Table of ToF RTT measurements

After the experimental implementation of ToF was completed, a series of tests was performed to judge if it was suitable as a solution. The results of this method are detailed in figure 7.

### 5.2.2 *Analysis of the Method*

The outcome of these tests confirmed our suspicions regarding the suitability of this method. Large 'jitter'in the output, caused partly by the inherent behavior of the communication protocols, and partly by inconsistencies in how the low level driver code in mbed and the binary ODIN-W2 driver behaved, rendered this initial approach to ToF using a single measuring station wholly unusable. Hence, a different approach was needed.

### 5.2.3 *RSSI using several BT Beacon*

After that we had ruled out ToF as a possible solution, we started to focus on a solution involving measurement of RSSI. We were investigating solutions involving both WLAN and BT. After a short time of further study we decided to focus only on BT, this due to the fact that is it easier to collect RSSI with BT than WLAN. A simple BT inquiry can receive all surrounding devices RSSI levels.

A single measuring station would present a major security flaw in the system; A simple change in the output power of an antenna could be performed to make a client appear closer to the Access Point than it really is. As ToF was unusable, we investigated using RSSI measured from multiple measuring stations, under the assumption that it would be more stable than the previous ToF single station solution; The RSS can be compared between nodes to estimate not only from where the transmitting signal is coming from, but also how trustworthy the signal is.

A simple initial attempt at localization was using the RSSI values to decide if a connection is coming from inside a given area, or from outside. For this we created the 'Triangle'filter event, an algorithm that performs the actual calculation to decide if a station is within that area or not. In a system with no less than three (3) BT beacons,

the simplest implementation is a simple boolean function, as seen in figure 8.

```
if( k*rss(AB,D) >= rss(C,D) &&
    k*rss(AC,D) >= rss(B,D) &&
    k*rss(BC,D) >= rss(A,D))
    return true;
return false;
```

Figure 8: The Triangle Filter. A, B and C are BT beacons placed strategically in a formation around the connectable region, D is the client that attempts a connection, and $\mathtt{rss}(A, B)$ is the RSSI of B as seen by A. A 2 letter code such as AB indicates the RSS value has been averaged between A and B, and k is a scaling factor that decides the size of the connectable region.

### 5.2.4 *Analysis of the Method*

After our initial testing we could conclude that the measured RSSI value was unreliable, we detected a large jitter between measurements. This made the results from trilateration unreliable and rendered it useless. We therefore focused on the comparably more reliable 'Triangle'filter and compared this to a simple static filter, that simply removes RSSI measurements above a statically assigned value. The static filter can be seen in figure 9.

```
 if( rss(A,D) >= minRSSI &&
     rss(B,D) >= minRSSI &&
     rss(C,D) >= minRSSI )
    return true;
return false;
```

Figure 9: The Static Filter. 'minRSSI'is chosen based on experimentation and desired connectable region alongside the output power of the beacon antennas.

## 5.3 IMPLEMENTATION

In the implementation, as the goal of this project was to determine if a client should be allowed access, and then grant them access, without requiring a password, it was necessary to run a network 'unsecured'. For this implementation, due to project time constraints, it was decided to implement a simple MAC filter that'll allow devices to connect if and only if they passed one of the proximity tests described in the experimentation phase. A device not using such a whitelisted MAC address would be denied access during the early stages of establishing a connection.
However, due to issues with the binary blob we found this particular functionality was missing (The ability to disconnect a given station), meaning a very rudimentary implementation of the system using MAC filters couldn't be completed within the project schedule, though most of the functionality is present.
Other methods for establishing a secure connection after a client has been verified to be allowed access were discussed, but due to time complexity issues weren't implemented. These are discussed in chapter 7.

We decided to reduce the maximum allowed Transmit Power (Tx) of the BT Beacons to the lowest possible value. This reduced the number of devices that could detect the BT Inquiry events transmitted by the beacons, and prevented inconsistent results that could happen as a result of unknown-to-us powersaving features built into the binary blob.

# 6 RESULTS

## 6.1 POSITION ESTIMATION

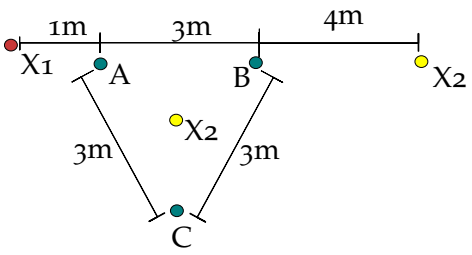


Figure 10: An overhead illustration showing how the BT beacons were arranged.

| Test | Real Distance from center | Triangle Filter | Static Filter |
|---|---|---|---|
| One PC | 1m outside 'triangle' | 58.29% | 46.86% |
| Two PCs | 1m outside, 1 at center | 0%, 41.11% | 0%, 97.78% |
| Two PCs | 1m outside, 4m outside | 22.95%, 0% | 18.03%, 0% |

Figure 11: Table showing connection success rates for the different setups for each client using both types of filters

As can be seen in figure 11, our testing shows that the Localization Filter *can* be used to gate access to a network based on proximity to the center when using the triangle filter (refer to figure 8), however, the results are too inconsistent and dependent on the presence or absence of other devices. A single PC connecting from outside the designated area should be denied access, but is being allowed a connection over 50% of the time. This is, security wise, completely unacceptable when using the triangle filter.

The static filter (Refer to figure 9) was far more reliable and deterministic. This due to the fact that it is not as important to get RSSI measurements with the property that the RSSI is proportional to the distance between station and beacons.

## 6.2 OVERALL PERFORMANCE

As may be inferred from the results in figure 11, the performance of at least the static filter can be considered reliable, with the caveat that it might not protect against a very determined attacker using a high sensitivity antenna with strong amplification. It should also be noted that the current implementation doesn't use encryption, with all the security implications that follow.

The connection time when using any of the BT beacon methods can be as fast as 15 seconds. This is totally dependent on the radio environment and the range from the client to the BT beacons. To show this we calculated the average time taken from starting a BT device until it was granted access. This was done under near ideal circumstances where the connecting client was placed in the center of the BT beacon triangle. A brief test showed the average time taken to establish connection to be 18.32s.

# 7 DISCUSSION

## 7.1 GOALS

Several of the initially set goals/requirements needed to be adjusted over the course of the project as it developed. For instance, one initial goal was to investigate methods that utilized only one module. When it became apparent that this wasn't going to work, the project moved to utilizing multiple modules.
Another goal was to produce a full proof-of-principle including the ability to connect to the network using any external device to show the system in action. However, we discovered that this was not possible due to certain software limitations, so this goal was scaled back to simply show which devices in range are allowed a connection.

## 7.2 FLAWS

Over the course of this project, multiple solutions have been considered. ToF based calculation based on measuring the RTT of data packets proved unreliable due to jitter, and too-low resolution timers present in the embedded hardware. As a result of this, the project moved to focus instead on an RSSI based solution, using multiple bluetooth beacons to perform localisation. However, even the RSSI-based measurements had its own share of problems, but much milder than ToF.

Time constraints caused the final implementation to rely on a subpar MAC-filter. This is far from ideal, since it introduces a major security flaw, in the form of impersonation attacks via MAC spoofing.
The system in its current form does not use encryption, but ideally this system would require a custom per-client password system (something akin to Protected Extensible Authentication Protocol (PEAP) or similar) to function; Each passphrase generated would have a limited lifespan, intended only to be used by the connecting station once during connection, and later expire after a set period of time (10 seconds or less). This would strengthen the security of the network by ensuring all data travelling over the air is properly encrypted, and make certain attack vectors, such as impersonation via MAC spoofing, unviable.

## 7.3 STRENGTHS

The biggest strength of this implementation is that it is independent from use of a client application, and has a very simple implementation. This results in an much easier adoption of this type of authentication than otherwise. This also means better cross platform support. It works with all devices that are equipped with both BT and WLAN.

# 8 CONCLUSION

## 8.1 OUTCOME

Due to the limitations in ublox binary driver several of our implementation suggestions ended up as impossible to implement. Due to this the end result became the only solution that we could get working. This means that limitations in the hardware and software effectively limited our implementation.

## 8.2 EXPERIENCES

This project has been a great opportunity to get insight on wireless system development with BT and WLAN for today's industry. To get familiar with tools such as yotta, mbed and OpenOCD. It has been very educational to work with embedded development regarding BT and LWIP. Our usage of LWIP have given us a glance into one of the biggest industry standards for WLAN TCP/IP management. We have also gained a deepened understanding of BT in embedded system platforms, especially for ublox.

## 8.3 FUTURE WORK

Form the experience that we got working on this project, we would suggest two topics for future work.

- A System based on Chronos[20] over BT, using appropriate hardware. This would cover adding the functionality needed by Chronos to existing bluetooth antenna hardware, implementing Chronos-style distance estimation, and would require features like phase offset detection and MIMO antennas.
- Time of Flight - with suitable hardware this includes more MIMO antennas, higher clocked processor and lower hardware access, that accommodate the implementation that we initially intended. This type of solution is intended for use with currently available WLAN devices, without the use of additional software, such as an client application.

# BIBLIOGRAPHY


[1] 2017. URL https://gnuarmeclipse.github.io/.

[2] A. Abu-Mahfouz and G. P. Hancke. Distance bounding: A practical security solution for real-time location systems. *IEEE Transactions on Industrial Informatics*, 9(1):16–27, Feb 2013. ISSN 1551-3203. doi: 10.1109/TII.2012.2218252.

[3] ARM. mbedos, operating system for embedded systems, 2017. URL https://www.mbed.com.

[4] ARM. yotta, software module system for mbedos 3, 2017. URL https://www.mbed.com/en/platform/software/mbed-yotta/.

[5] Wan Mohd. Yaakob Wan Bejuri, Mohd Murtadha Mohamad, Maimunah Sapri, and Mohd Adly Rosly. Ubiquitous wlan/camera. *CoRR*, abs/1204.2294, 2012. URL http://arxiv.org/abs/1204.2294.

[6] *Wi-Fi Location-Based Services 4.1 Design Guide*. Cisco Systems, Inc., 4.1 edition, May 2008.

[7] Jolyon Clulow, Gerhard P. Hancke, Markus G. Kuhn, and Tyler Moore. *So Near and Yet So Far: Distance-Bounding Attacks in Wireless Networks*, pages 83–97. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-69173-0. doi: 10.1007/11964254_9. URL http://dx.doi.org/10.1007/11964254_9.

[8] Qian Dong and Waltenegus Dargie. Evaluation of the reliability of rssi for indoor localization, 2012.

[9] Eclipse. Eclipse, ide for java and embedded software development, 2017. URL http://www.eclipse.org/.

[10] Z. Farid, R. Nordin, and M. Ismail. Recent advances in wireless indoor localization techniques and system. *Journal of Computer Networks and Communications*, 2013, 2013. URL http://dx.doi.org/10.1155/2013/185138.

[11] Git. git, a free and open source distributed version control system, 2017. URL https://git-scm.com/.

[12] K. C. Ho and Wenwei Xu. An accurate algebraic solution for moving source location using tdoa and fdoa measurements. *IEEE Transactions on Signal Processing*, 52(9):2453–2463, Sept 2004. ISSN 1053-587X. doi: 10.1109/TSP.2004.831921.

[13] µOS++ IIIe team. µOS++ IIIe, 2017. URL https://micro-os-plus.github.io.

[14] K. Bonne Rasmussen and S. Capkun. Implications of radio fingerprinting on the security of sensor networks. In *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm 2007*, pages 331–340, Sept 2007. doi: 10.1109/SECCOM.2007.4550352.

[15] Dominic Rath. Open on-chip debugger, 2017. URL http://openocd.org/.

[16] L. Schauer, F. Dorfmeister, and M. Maier. Potentials and limitations of wifi-positioning using time-of-flight. In *International Conference on Indoor Positioning and Indoor Navigation*, pages 1–9, Oct 2013. doi: 10.1109/IPIN.2013.6817861.

[17] u blox. u-blox, 2017. URL https://www.u-blox.com.

[18] u blox. Odin-w2 series, stand-alone iot gateway modules with wi-fi and bluetooth, 2017. URL https://www.u-blox.com/en/product/odin-w2-series.

[19] u blox. u-blox evaluation kit for odin-w2 series, 2017. URL https://www.u-blox.com/en/product/odin-w2-series.

[20] Deepak Vasisht, Swarun Kumar, and Dina Katabi. Decimeter-level localization with a single wifi access point. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 165–178. USENIX Association, 2016.

## COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/


PO Box 823, SE-301 18 Halmstad
Phone: +35 46 16 71 00
E-mail: registrator@hh.se
www.hh.se

[June 2017 at 7:53 – classicthesis version 1.0 ]