Localization with LoRa

Independent of Network coverage

Bachelor Thesis

Marcel Würsten

University of Bern

September 2020

Erklärung

gemäss Art. 30 RSL Phil.-nat.18

Name/Vorname: Marcel Würsten

Matrikelnummer: 16-802-712

Studiengang: Bachelor

Titel der Arbeit: Localization with LoRa, Independent of Network coverage

LeiterIn der Arbeit: Prof Dr. Torsten Braun

Ort/Datum:

Unterschrift:

Abstract

Bee colonies are the most valuable resource of a beekeeper. The BeePhone, developed by Abilium, is a small intelligent LoRa device to monitor the beehive without requiring to open the beehive and without additionally stressing the bees and weakening their immune system. Since beehives are also stolen, it should be possible to locate a BeePhone inside the beehive. Since there is no localization technique on the BeePhone and the LoRaWAN used to transmit data lacks redundancy for classic trilateration, we evaluate the possibility to locate a BeePhone with mobile gateways and TDOA or RSSI. TDOA is not possible due to timing problems on our gateway. But we propose and implement an RSSI based trilateration algorithm which uses map data to model the environment. The algorithm improves with each additional measurement.

Prof. Dr. Torsten Braun, Communication and Distributed Systems, Institute of Computer Science, University of Bern, Supervisor

Jakob Schaerer, Communication and Distributed Systems, Institute of Computer Science, University of Bern, Assistant

Contents

1 Introduction

In recent years, more and more devices have been connected to the Internet to provide additional services. These are often summarized under the term Internet of Things (IoT), or IoT devices. Many of these devices are battery-powered and to extend the battery life there are often various limitations. These limitation can contain the necessity to use a deep sleep or restrictions on which sensors can be used.

To provide additional services, the devices often need to know where they are located. Depending on the restrictions that apply to the IoT device in question, this is not a trivial matter. Depending on the restrictions to save battery, the area of application, the required accuracy and possibly other factors, different localization technologies are used.

If a device only needs to know whether it is at home or work, it may be sufficient to scan and compare the WiFi networks. If you want to know where you are in a building or room, you have to look at indoor localisation. But there are also different variants for outdoor use. Global navigation satellites systems such as the American "Global Positioning System (GPS)", the Russian "Globalnaya navigatsionnaya sputnikovaya sistema (GLONASS)", the European Galileo, the Chinese "BeiDou Navigation Satellite System (BDS)" or the numerous regional supplements such as the Japanese "Quasi-Zenith Satellite System (QZSS)" or the Indian "Indian Regional Navigation Satellite System (IRNSS)" are probably the easiest way to locate a device, if you have no restrictions that prevent their usage. But outdoor you can determine your position more or less exactly based on the propagation of radio signals. First of all you can also receive WiFi data at many outdoor locations and if you compare it with a database you might determine the position. Furthermore, there is the possibility that a network can determine the position of a device. If several receivers receive the same signal, the differences in arrival time or signal strength can be compared to determine the location.

For GSM the easiest way to locate a device is to look at the current cell the device is registered. The disadvantage is the relatively low accuracy which is directly related to the density of base stations. However, in GSM timing advance (TA) is a value used to synchronise mobile phone and base station, so that with the signal runtime taken into account, a signal is received at the given time slot without collision. With this information you can now calculate the distance between the mobile phone and the base station. Due to the system specific steps of $3.7\mu s$ only an accuracy of about 550 metre is possible.

1.1 Motivation

Colonies of bees and their hives, together with the honey they contain, are a beekeeper's most valuable resource. Because the colony with the honey is the most valuable resource of a beekeeper, he wants to supply the colony in the best possible way. The varroa mite is a feared enemy of the honey bees[1]. A mite infestation weakens the bees in various ways. The larvae lose weight, the hatched bees are smaller and have a significantly shortened lifespan. Additionally, they have a worse learning performance and more often do not return to the hive. In order to know what his colony is doing, the beekeeper must open the hive frequently. This means stress for the bees and stress can lead to a higher susceptibility to disease. Until today, beekeepers could not avoid to open and look through their colonies.

Unfortunately, it happens that beehives are stolen. Either the thief only wants the honey, or he/she wants to use the colony for himself/herself. Thanks to the latest research and technology, it is now possible to place a small intelligent device to monitor the colony into the beehive. With the BeePhone Abilium has developed such a device [2]. The BeePhone gives valuable information about the current state of the colony without requiring the beekeeper to open the beehive. Important conditions such as brood care or folk strength can be recognized, for example. With the help of artificial intelligence the buzzing of the bees is analyzed and the swarm behaviour as well as the presence or absence of the queen will be detected fully automatically. Additionally, the BeePhone provides the beekeeper with information such as the internal temperature and relative humidity. These indicators help to estimated the health of a colony.

With its dimensions of $14 \times 33 \times 41$mm, a BeePhone is so small that it can be placed in a beehive without any problems. Despite its small size, the BeePhone has a battery life of about 90 days and can be easily charged via USB. To send all the collected data to a central server, the BeePhone uses the Low Power Network (LPN) network of Swisscom, a LoRaWAN® network (more on that in section 2.1 LoRa / LoRaWAN). The beekeeper can view all collected information via website or app and learn a lot about his colonies without opening the hives.

If a beekeeper places such a BeePhone in his beehive, it would be practical if the theft protection would also be improved. Well, a BeePhone cannot possibly prevent the theft. But it can detect a theft through its acceleration sensors and alert the beekeeper. If the beekeeper no longer catches the thief during act, it would be advantageous to locate the BeePhone and thus the beehive. The BeePhone should be as small and relatively cheap as possible, and with that come many restrictions. Because of these restrictions, it is impossible to use the already well-known tracking technologies like global navigation satellites.

As already mentioned, the BeePhone uses the Swisscom LPN, a commercial LoRa network, which is available almost everywhere in Switzerland. But especially in rural areas, there are usually not enough gateways within range for the classic trilateration. According to Abilium, Swisscom can pinpoint the location of a BeePhone to within 750m, using Receiver Signal Strength Indicator (RSSI), empirically measured loss maps and possibly other proprietary information. This accuracy is not sufficient to recover a stolen BeePhone.

1.2 Contributions

In this thesis, we present a localization system independent of the LPN coverage and without changing any hardware on the BeePhone. For our approach, we make some assumptions, which are specific for this application. The most important assumption is that the BeePhone is at a fixed location after the initial movement of being stolen. We assume the beehive is stolen and then placed at another location to produce honey for the thief. The BeePhone has no value for the thief and is most likely not even noticed. We use mobile gateways with accurate GPS position to get more information about the location than the fixed gateways. With these mobile gateways we search the missing BeePhone and then locate them. We also use publicly available map data to get more information about the environment around the mobile gateway and with that also of the environment around the BeePhone.

2 Theoretical Background

If you want to get a location from a signal, you are limited to the physical appearance of the signal at the receiver. This includes the angle of arrival (AOA)[3], the time of arrival (TOA)[4] and the received signal strength (RSSI). In our case, we couldn't specify the angle of arrival since our gateway has only one omnidirectional antenna. So we have two possibilities. We receive the same signal by at least 2 gateways and calculate the difference in arrival time to get a line on which the transmitter lies and repeat this process to reduce the possibility to one point. Or we take the signal strength on multiple locations and guess where based on this information the source is.
We wanted to compare the two possibilities (TDOA and RSSI) and find the method which is better suited for our situation.

2.1 LoRa / LoRaWAN

Long Range Wide Area Network (LoRaWAN®) is a low-power wireless network protocol. It is freely available and uses a proprietary and patented transmission method based on a chirp spread spectrum modulation technique called "LoRa" from Semtech Corporation. Often the abbreviations "LoRa PHY" is used for the proprietary physical LoRa layer, "LoRaWAN" for the open source network stack, and "LoRa" for the entire system.

2.1.1 LoRa PHY

The physical layer of LoRa often referenced as "LoRa PHY" used a proprietary spread spectrum modulation that is a derivative of chirp spread spectrum (CSS) modulation. The spread spectrum modulation in LoRA is performed by representing each bit of payload information by multiple chirps of information[5]. LoRa can trade off data rate for sensitivity with a fixed channel bandwidth by selecting the amount of spread used. A lower spreading factor (SF) means more chirps are sent per second, thus more data can be transferred per second. A higher SF implies fewer chirps per second, thus less data transferred per second. Sending with a higher SF needs more transmission time, meaning the device is running longer and consuming more energy. The benefit is, that the receiver has more opportunities to sample, resulting in a better sensitivity [6]. Furthermore, LoRa uses forward error correction coding to improve resilience against interference. The high range of LoRa is characterized by the high wireless link budget of around 155dB to 170dB[7].

2.1.2 LoRaWAN

LoRaWAN® is one of several protocols that defines the upper networking layers missing from "LoRa PHY". LoRaWAN® is open source, but defined and maintained by the LoRa Alliance. An open, nonprofit association with over 500 members, created to support LoRaWAN® as well as to ensure interoperability.
LoRaWAN® is responsible for managing all the physical parameter of a communication link like the frequencies, data rate and power for all devices. In LoRaWAN®, devices are asyncronous and transmit when they have data available to send. The data is received by multiple gateways, which forward the packets to a centralized network server. The network server filters duplicate packets, tries to decrypt the packet with the correct AES key before forwarding to the application server. The technology shows high reliability for moderate load, but it has performance issues when sending acknowledgements. LoRaWAN® is designed for energy efficiency over ranges of 10km while the data transmission rate is between 292 bit/s and 50kbit/s[7][8].

2.1.3 Limitations of LoRa

LoRa is extremely low power. While it is beneficial for the battery life of the device, it also has some disadvantages. Especially in urban areas and with not an ideal antenna placement the SNR (Signal to Noise Ratio) is a more limiting factor than the signal strength. While LoRa can recover the data even below the noise floor there is a limit between $-15dB$ and $-20dB$ where you could no longer successfully retrieve a packet.
To achieve long ranges of 10km or more with such low power the data rate is extremely low and one transmission can take multiple seconds on the highest spreading factor (SF12 = highest distance). When a LoRa device is moving during transmission and a high SF is used the packet loss rate increases. The Doppler effect is the change in frequency if transmitter and receiver move relative to each other. The frequency shift due to the Doppler effect causes the autocorrelation peak on the receiver to shift in time. If using a low SF like SF7, the chirp rate is high enough that the time shift can be neglected, making LoRa resilient against the Doppler effect when using a low SF. But when using a lower chirp rate like SF12 the duration of each chirp increases and thus the time shift is not longer negligible[9].
As always with radio transmissions, you have to follow the regulations of the local authorities. In our case, LoRa operates on the unlicensed $868MHz$ spectrum. This comes with the condition to only transmit with a maximal power of $25mW$ and a duty cycle of $< 1\%$. Therefore, we are not allowed to transmit more than 1% of the time, at any time. This means that we are restricted in how many packets we can send in a given time.

2.2 Time Difference of Arrival (TDOA)

A few factors are influencing the precision of TDOA. We just want to briefly mention only the most important ones in our opinion.

2.2.1 Sample Rate and Bandwith

LoRa operates on $868MHz$ in Europe. When using the same sample rate (because increasing the sample rate beyond the modulation rate, does not improve results), we get a time resolution of around $1.15ns$ (Equation 1), which corresponds to approximately $0.3m$ (Equation 2). This is accurate enough for our application and we do not have to worry about that.

$$time = \frac{1}{frequency} = \frac{1}{868MHz} \approx 1.15ns \tag{1}$$

$$distance = speed\ of\ light \times time \approx 3 \times 10^8 m/s \times 1.15ns \approx 0.3m \tag{2}$$

2.2.2 Network Geometry

When using TDOA between two gateways you get a hyperbole, telling gateway A is x metres closer to the transmitter than gateway B. Since these measurements have a few variations, it is more a corridor of gateway A is $x \pm \mu$ closer than gateway B (Figure 1, the yellow lines). When taking multiple measurements the intersection area of these corridors is smaller the more orthogonal 2 corridors are to each other (Figure 1, location probability heat map with scale from black/blue = low probability to yellow = medium probability to red high probability. The red zone is the intersection between the 2 corridors. The left situation is far better than the right). In reality, this means when using just 2 gateways, that to get an optimal result the transmitter should be in the middle of them, and after one successful transmission, the gateways should rotate 90° around the transmitter. In practise, this is probably impossible, but when receiving from multiple locations, it should still improve the overall result.

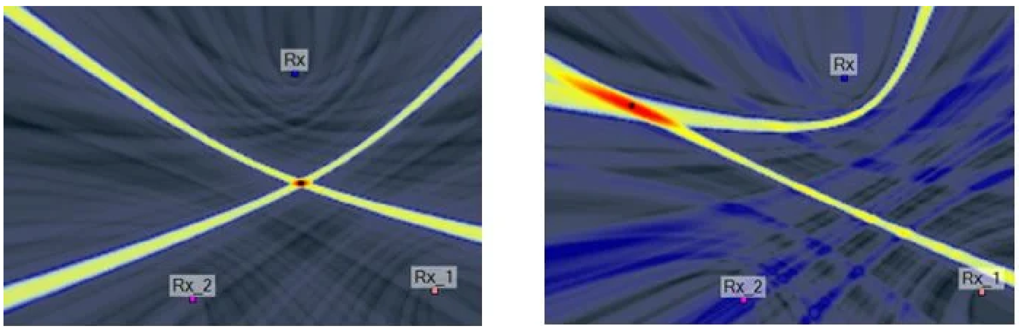


Figure 1: Different Location probability when receiver placed different relative to transmitter
Source: [10]

2.2.3 Time Accuracy

Precise synchronization between receivers is essential for high-accuracy TDOA. The simplest method is to use a GNSS (Global Navigation Satellite System) like GPS. When using the GPS system the accuracy strongly depends on the quality of reception, but since our application is outdoors, this should be possible. Good GPS condition (free view into the sky) should allow the synchronisation error between receivers to be less than $30ns$ which corresponds to $3 \times 10^8 m/s \times 30ns = 9m$ but, under typical conditions it's a bit less accurate. We use the GPS module on our gateway witch specifies a PPS accuracy of $10ns$ to the GPS time which corresponds to $3 \times 10^8 m/s \times 10ns = 3m$. So our accuracy depends primarily on the quality of the GPS signal quality. Typically, we can assume an accuracy of about 12m.

2.3 Receiver Strength Signal Indicator (RSSI)

Basically, we want to estimate the distance the signal has travelled from the signal strength and when we have multiple measurements at different locations, we should get one point where all these distances match. Here should the transmitter lie. This is the classic trilateration model. For the gateway measurement point P with the distance d to the transmitter and the transmitter location B it can be expressed as a simple sphere equation (Equation 3).

$$d^2 = (P_x - B_x)^2 + (P_y - B_y)^2 + (P_z - B_z)^2 \tag{3}$$

Simplified in a world with 3 measuring points something like in Figure 2 should be the result.
So our main problem is, how to estimate the distance. Currently it is pure guessing. We use

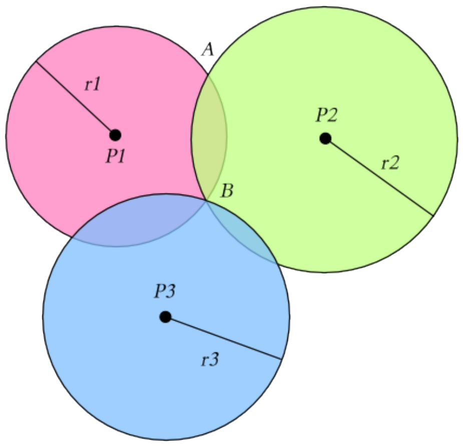


Figure 2: Simple trilateration
Source: [11]

the a signal propagation model proposed and evaluated by Oguejiofor [12]. This model predicts that the path loss $P_L(d_i)[dB]$ over the distance d_i between the transmitter and the receiver is define as equation 4, where n is the path loss exponent and $P_L(d_0)[dB]$ the path loss over a known reference distance d_0.

$$P_L(d_i)[dB] = P_L(d_0)[dB] + 10n\ log_{10}(\frac{d_i}{d_0}) \tag{4}$$

Oguejiofor[12] tells us, that in the free space n is regarded as 2 and n is higher if obstacles are in line of sight. He empirically measures these values to test it.

2.4 Bresenham's Algorithm

Bresenham's algorithm is an efficient algorithm to render a line with pixels. It was developed by Jack Bresenham at IBM in 1962 [13]. The remarkable about the algorithm is that it minimizes rounding errors caused by discretization and is at the same time very easy to implement. Also, the addition of integers is the most complex operation and, therefore, it is very fast even on the simplest hardware[14].

The algorithm is the simple to calculate for a line from (x_1, y_1) to (x_2, y_2) under the conditions $x_1 \leq x_2$, $y_1 \leq y_2$ and $0 < dy \leq dx$ with $dx = x_2 - x_1$, $dy = y_2 - y_1$. All the other cases can be covered with case discrimination and changing the signs of dx and dy, as well as swapping x and y like in Listing 1. In Figure 3 there is such an example. The algorithm iterates along the longer axis (in this case the x-axis) and choses the integer y corresponding to the pixel center that is the closest to the ideal y for the same x. As x is iterated and with the previous conditions, y stays the same, or increases by one on each iteration.

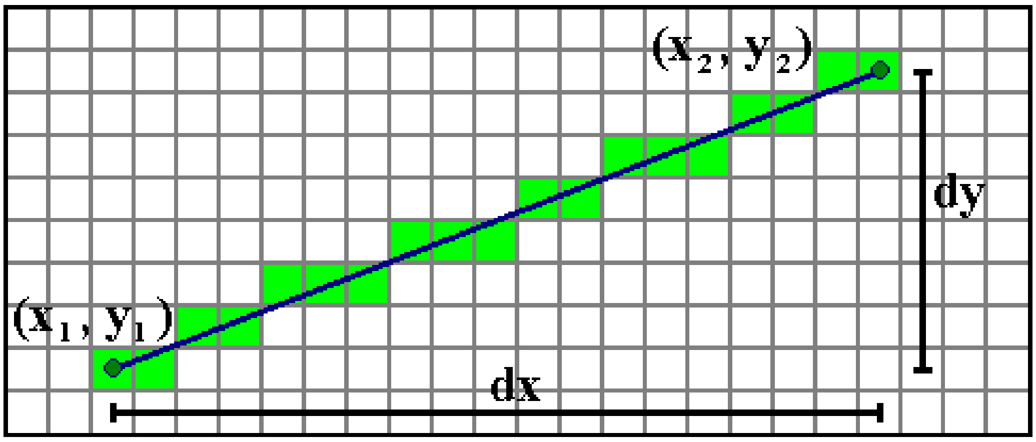


Figure 3: A simple example for Bresenham's Algorithm
Source: [15]

```c
void bresenham(int x0, int y0, int x1, int y1) {

  int dx = abs(x1-x0), sx = x0<x1 ? 1 : -1;
  int dy = abs(y1-y0), sy = y0<y1 ? 1 : -1;
  int err = (dx>dy ? dx : -dy)/2, e2;

  for(;;){
    setPixel(x0,y0);
    if (x0==x1 && y0==y1) break;
    e2 = err;
    if (e2 >-dx) { err -= dy; x0 += sx; }
    if (e2 < dy) { err += dx; y0 += sy; }
  }
}
```

Listing 1: A compact version of Bresenham's algorithm in C

3 Related Work

Because precise location is important for many applications, there is also a lot of research into this field. As there is a very well known, reliable and in many cases usable system for outdoor use with global navigation satellite systems (with GPS as the best-known representative), a lot of research is needed for an indoor positioning system, where these systems do not work as well. Here are many different approaches, which do not use global navigation satellites. Due to the widespread use of WiFi, fingerprinting in combination with other sensors can already provide a very good location [16]. Machine learning algorithms for indoor localisation, in particular, have become increasingly popular in recent years. Therefore, it is not surprising that many machine learning algorithms have already been tested for their suitability for localization determination. Especially, with semi-supervised learning [17] and reinforcement learning [18] some success seems to be achieved.

We could name countless promising approaches like particle filter-based reinforcement learning (PFRL) [18][19], hidden Markov chain [20] or Bayesian hierarchical model [21] but they all share the commonality of ML and thus the assumption that the algorithm can learn. But in order to train the algorithm you need a sufficiently large data set, most likely not available in the area we need localization.

An extremely wide field of research is multilateration localization, a fundamental localization technology. Multilateration is the extension of trilateration to more than 3 anchor to determine the location of a transmitter. Typically, these approaches are based on various types of measurements like Time of Arrival (TOA), Angle of Aarrival (AOA) or Received Signal Strength (RSS). Then there are countless approaches to determine that exact location from these distances, each focusing on a specific factor. But there also exist work on general multilateration researching general topics like the influence of anchor location uncertainties [22] or the robustness [23].

Angle of arrival (AOA) localization has been a popular research topic, since it can locate a source in a passive manner and compared to time of arrival (TOA) and time difference of arrival (TDOA) does not require synchronization with the signal source or other distributed receivers. There are many different ideas how to predict the location from the measured angle[24] [25] [26] [27] [28]. Even if AOA has advantages as the not required synchronization, it has the disadvantage of needing an antenna array, making it useless for our application.

Time of Arrival (TOA) or Time Difference of Arrival (TDOA) to eliminate the unknown time of emission has already been analyzed in many different scenarios [29][30][31][32][33]. TOA measurements can be made with only one receiver, but time synchronisation between the transmitter and receiver is needed [34]. TDOA on the other hand needs at least two receivers and time synchronisation between them is needed. The extreme time sensitivity makes this approach challenging in various ways. TOA returns the distance between the receiver and transmitter, while TDOA returns a hyperbole, on which the transmitter lies, since it tells you receiver 1 is closer to the transmitter by x meters compared to receiver 2.

In our case only TDOA is a possibility, since there is not a sufficiently accurate clock on the BeePhone for TOA. Additionally, we have no possibility to achieve the necessary synchronisation accuracy with the given hardware.

Receiver Signal Strength (RSS) or often Receiver Signal Strength Indicator (RSSI) are often used with multilateration due to the wide availability in most radio communication links. It is highly dependent on the environment and, thus, provides additional challenges. But as with all the other types of measurement, there is a lot of research done in this field [35][36][37].

Furthermore, there are some approaches to try classical multilateration and still to meet the particularities of the power dissipation by sight and not sight. For example, there is an approach to change the geometric centre of gravity of the polygon surrounding the sensors by weighting it so that it corresponds to the position of the device [38]. A very well known work of Bahl and Padmanabhan[39] is that the classical multilateration by receiver signal strength, where the signal is received by several stations and empiric measurement values are combined with theoretical signal propagation theories for location [39].

Furthermore, there are some approaches to try classical multilateration and still meet the special features of power dissipation by sight and not sight. Alyafari et al. [40] compare e.g. two different proximity-based algorithms CDRSS and WCC. CDRSS (Combined differential receiver signal strength) builds upon a differential receiver signal strength approach, while WCC (weighted circumcenter) uses a geometric approach by shifting the circumcenter to the position of the transmitter by using weights [40]. The advantage of this work is that it is completely passive, i.e. the device to be located does not send any information about its surroundings, or may not even know that it is being located. This also means that no specific software needs to be executed on the device, which is an advantage for IoT devices. Even if this was developed for GSM, this could probably work on other radio communication links providing all the required features.

Even if most of the above cited studies do not use LoRa, some might be adapted since they all use some sort of radio link. However, there are also some studies about localization with LoRa [36][41][42][43][44]. But they all just use a fixed and highly redundant gateway system not available to us.

However, Delafontaine et al. showed in [45] that extending an existing fixed LoRa gateway system with a swarm of mobile gateways placed in UAVs helps localizing a LoRa device. Since using a swarm of UAVs is a big effort, we want to use only a few ground based mobile gateways.

4 Feasibility of TDOA and RSSI Localization

4.1 TDOA

Due to the theoretical background, we would first like to evaluate what is the highest possible accuracy that we can achieve on our hardware with the means at our disposal. With TDOA we are mainly concerned about time accuracy. We are primarily interested in the variance of time accuracy and less in the absolute time. Therefore, we evaluate different sources of inaccuracy in the time domain and how they influence our localization accuracy.

4.1.1 Linux Kernel

A great source of inaccuracy lies in the detection of receiving a LoRa-packet. We do not care about the delay as long as it is constant. Our Raspberry Pi based gateway, the Dragino PG1301 (more on that in section 5.2 LoRa Gateway) only connects the Serial Peripheral Interface (SPI) of the Semtech SX1301. SPI uses a master-slave architecture, so the Raspberry Pi needs to pull data from the gateway[46]. Pulling the data over SPI adds immense uncertainty because each loop takes a few microseconds and we do not know when in this time the packet has arrived. Additionally, the loop used to continuously pull data does not always take the same amount of time, because it might get interrupted sometimes by the Linux process scheduler. This uncertainty when exactly a packet has arrived leads to an inaccuracy of the timestamp taken when a packet arrives. Each μs of inaccuracy leads to a $\approx 300m$ wider corridor and thus to a less accurate position. A try to fix this is by applying the "PREEMPT_RT" patch to add real-time capabilities to the Linux kernel. We did not measure the variation without the "PREEMPT_RT" patch, but according to multiple online reviews it significantly reduces from the millisecond range to microseconds.

We also soldered a cable directly to the SX1301 GPIO[4] and connected it to the Raspberry Pi, as this GPIO pin can be configured to give a signal when a packet arrives. To eliminate context change on interrupt, we have written a small Linux kernel module that registers an interrupt and handles it by storing the timestamp and making it available to user space so it can be used there in the packet processing.
To measure the variation in the interrupt time, we changed the module to switch another GPIO pin instead of saving the time. We used a dual-channel oscilloscope to define the time difference between the edge of the signal on the interrupt pin and the edge of the response pin.
The mean of around $40.6\mu s$ is irrelevant, but a standard deviation of around $4.7\mu s$ adds an inaccuracy corresponding to $3 \times 10^8 m/s \times 4662ns \approx 1.4km$ wide corridor.

The only option would be to abandon the Linux kernel and try the bare metal approach. This, however, is not an option for us, since we have too many dependencies and we would have to rewrite all that functionality our selves. This leads to the conclusion, that in our current setup, time measurement with the Raspberry Pi with the needed precision is impossible.

4.1.2 Wiretapping SX1301

The SX1301 is the Chip on our gateway, which does the digital signal processing (DSP) of the I/Q-lines and outputs LoRa-packets over SPI. We can not only configure the frequency on the SX1301, we also have very limited other configuration. One of them is, that GPIO[4] indicates when a packet is received. So, an alternative to time taking with the Raspberry Pi would be to use an external FPGA. But first, we measure how precise the GPIO[4] of the SX1301 functions. For this test, we set up two identical Raspberry Pi (cloned MicroSD cards), place them near each other and connect to each of the GPIO[4] a channel of our oscilloscope. Then we send a LoRa packet from a LoRa-node and measure the difference of the pins (test setup sketch

Figure 4). They have high differences of up to $8\mu s$. This $8\mu s$ correspond to an inaccuracy of a $3 \times 10^8 m/s \times 8000ns = 2.4km$ wide corridor. The differences might come from the fact, that first the packet is processed with error correction and then the chip decides to notify us about the packet or not.

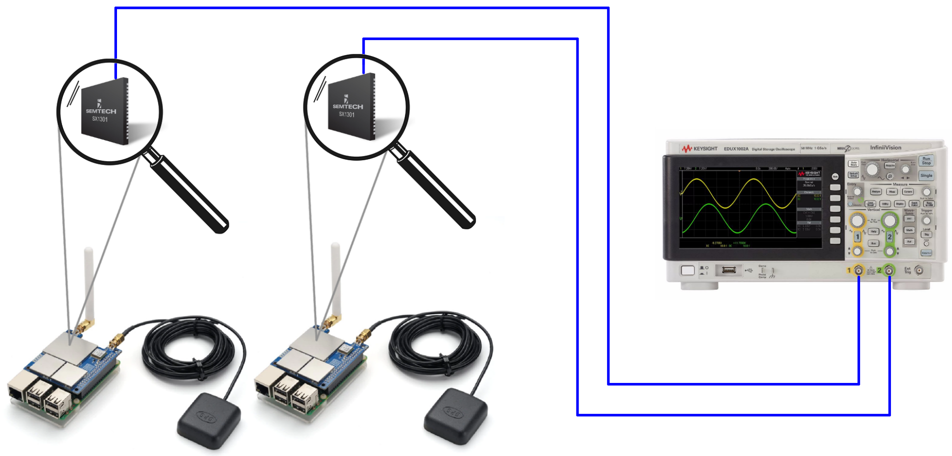


Figure 4: sketch of test setup

4.2 RSSI

With the knowledge from the theoretical background, we try to create an algorithm that fits our specific situation. We implement this algorithm and evaluate if and how well it works.

4.2.1 Algorithm

If we look at Oguejiofor's RSSI model from section 2.3, then path loss exponent n, describing the environment effects on the signal propagation, is missing. We only know that for free space $n \geq 2$. We cannot measure it like Oguejiofor, because our algorithm should work in all parts of Switzerland and the effort for detailed empiric measurements would exceed the scope of this work. To compensate this we need some assumptions and some statistics. We assume the path loss over a known reference distance is constant for all measurements during one search. To further simplify the problem, we assume our reference distance is $1m$. For each coordinate, we calculate the variance between the value over the reference distance between all the measurements. Therefore, we take the path loss as the difference between the known transmission power of 25mW and the RSSI value measured. We define d_i as the distance between the measuring coordinates and the coordinates to test. The path loss exponent gets derived from preprocessed map data. So we have Equation 5.

$$P_L(d_0)[dB] = Send[dB] - Receive[dB] - 10n\, log_{10}(d_i) \tag{5}$$

$$Var(P_L(D_0)[dB]) = (P_L(D_0)_i[dB] - \mu)^2 p_i \tag{6}$$

To be more precise we take Equation 6 with $P_L(D_0)_i[dB]$ as the calculated value over the reference distance for the i-th measurement, μ the mean overall $P_L(D_0)_i[dB]$, and p_i the weight factor. We weight those values with a lower loss higher than those with a higher loss. Therefore, define p_i as $min(log_{10}(\frac{P_L(D_i)_i[dB]}{100})), 0)$ to get a positive factor over the negative loss.

For the path loss exponent we take publicly available map data, which we preprocessed (section 4.2.2). On this data we run Bresenham's algorithm (section 2.4) to get the values for each square meter on the straight line between the gateway and the possible location as fast as possible.

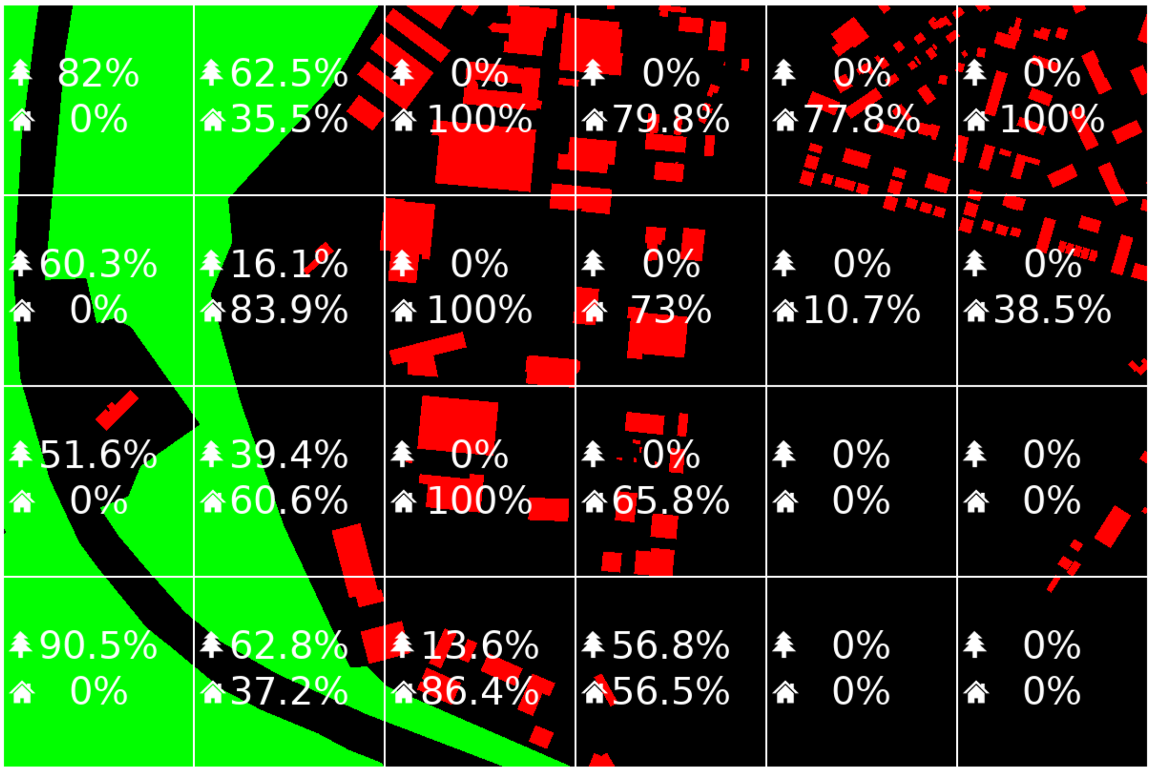


Figure 5: Comparison between the 2 grid
200m grid with percentage for forest and residential area
1m grid as background with colors green = forest, red = buildings

For each line we count the fields in forest and buildings and normalize this value by $\frac{distance}{\#of\ fields}$ to get values $\in [0, 1]$. We multiply values by the c_f and c_b for forests and buildings respectively. In our case the values $c_f = 2.5$ and $c_b = 5$ seemed to work best. To visualize everything on the map, we use a simple linear green red scale.

4.2.2 Map data processing

We needed specific map details in the affected area in an open format for our calculations. We downloaded the raw vectorised data from the Open Street Map project. At first, we thought that we would achieve our goal by aggregating the data into a grid before. But we found out, that this does not lead to exact results, because then we would have to regenerate discarded data. We do not know where a building is in this grid, whether there are several buildings or what the shape of them is. Additionally, we took the residential area polygons instead of each individual building. In Figure 5 there are a few additional inaccuracies we had from this choice. This can be compared to the upscaling problem with photos when the computer would have to generate additional details in photos if the user wants to zoom in more. In order to minimize real-time calculations, in our second approach, we processed them into raster images with only 2 bits of information per square meter (one bit, if there is a building, and a second if there is a forest). Since we want to make use of compression to minimize the needed disk space and since for one big file this is quite slow, we split the map data into square kilometres. For the simple usage, we decided to use PNG file on the highest lossless compression with 4 indexed colours (green = forest, red = building, yellow = building in a forest, black = neither forest nor buildings). This also allows for a really simple verification if the map data is correct. Since we only want to store the map data of Switzerland, we use the LV03 coordinate system (EPSG:21781)[47] and the Java implementation of Swisstopo-WGS84-LV03 [48]. We are aware that this system has been officially replaced by LV95 (EPSG:2056)[49] but since we use the same library for the transformation from WGS85 to LV03 and back, we believe that errors are minimal.

To convert all the data into a raster image and to process them into a Geo Tagged Image File Format (GeoTIFF) file we used “QGIS”, a powerful Open Source Geographic Information System[50].
To split the GeoTIFF into Tiles per square kilometre and save as Portable Network Graphics (PNG) we used the GeoTools[51] library, an open-source Java library that provides tools for geospatial data.

5 Implementation of BeePhone

5.1 BeePhone

In the end, only a small extension of the BeePhone firmware was needed, so that if the device had been stolen a packet is sent every 100s. But to develop and test our algorithm, we have completely changed the firmware to use the public "The Things Network" (TTN) and to send packets every 100s. During this time, we had many problems with the only partially correct manual we found [52] for the AT software of the LoRa Transceiver on the BeePhone.

5.2 LoRa Gateway

In the beginning, we tried to evaluate the available gateways that are on the market and which were satisfying our criteria the best. In general, one can divide the LoRa gateways with GPS receiver into two groups. On one side there are the carrier-grade solutions, which easily cost several thousand dollars, and on the other side there are the cheaper systems still costing at least hundreds of dollars. We wanted a "low-cost" approach, with a system (only) costing several hundred dollars. Our focus was a LoRa gateway with an extremely precise GPS module since each nanosecond time difference later adds to the inaccuracy of the whole system. An internal LTE modem would be nice, but if we have the option to add one externally it is also no problem. In the evaluation process, we had to decide if a gateway with only one channel is sufficient. We denied this, since this would limit our options quite a bit. This further limited our chooses, because we need a gateway that supports at least the 8 channels available in Europe. In the end, we choose the PG1301 from Draguino, a Hardware Attached on Top (HAT) expansion board for a Raspberry Pi.

On the Raspberry Pi 4, we run Raspian as OS and a complex stack of different Docker Containers, each providing different functions on our Raspberry Pi. We use Docker so that the applications run isolated and do not disturb each other. And of course, there are all the other advantages of Docker, such as the fact that the containers can be easily distributed to other systems. We want to receive and decrypt the LoRa packets of our BeePhone with our

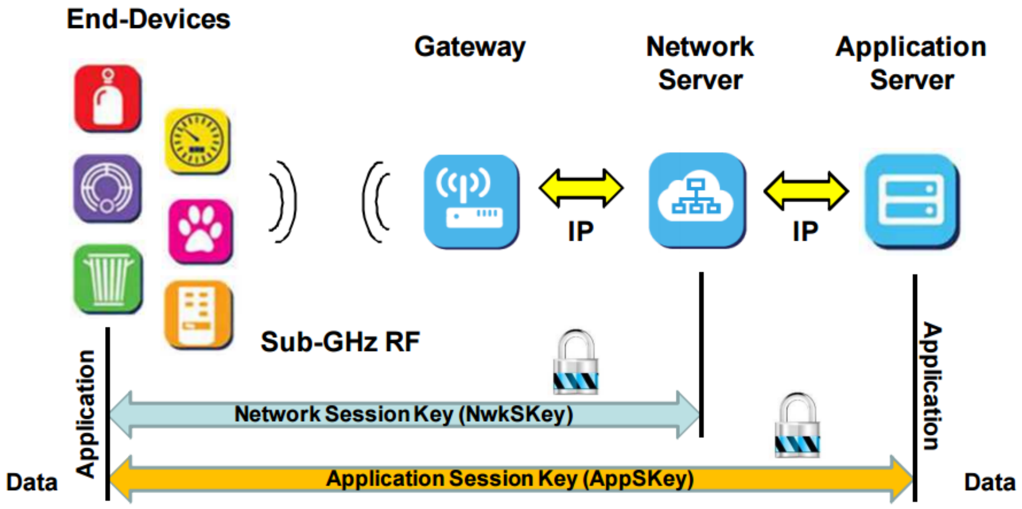


Figure 6: LoRaWan Encryption
Source: [53]

own gateways, even if the BeePhones are still connected to the Swisscom LPN. For this, we need the device address and the 2 session keys (Network-Session-Key and Application-Session-Key). These keys are generated when the BeePhone joins the LPN with Over-the-air-activation (OTAA).

There are two procedures for a LoRa device to join a LoRa network. Over-the-Air Activation (OTAA) is the preferable and most secure way to connect with a LoRa network. Devices perform a join-procedure with the network, during which a dynamic device address is assigned and security keys are negotiated with the device. The second procedure is Activation by Personalization (ABP). In some cases it might be required to hardcode the device address as well as the security keys in the device. This strategy might seem to be simpler because you skip the join procedure, but it has some downsides related to security. In addition, it is not future-proof in any respect because a device cannot react to possible changes in the network. Even a simple operation, such as if a security key changes by the network operator changes for some reason, means the end of life for the device.

To handle and decrypt the packet we use ChirpStack, an Open Source software stack for private LoRa server infrastructure. Basically, Chirpstack is far too powerful for our needs, but it can do a lot of things that we otherwise would have to implement ourselves. Chirpstack is designed to have many gateways and to get a corresponding number of encrypted LoRa packets. In Chirpstack the gateway bridge is the first element in a chain of applications and receives the encrypted LoRa packets from the gateways via the Semtech UDP packet forwarder protocol.
The Semtech UDP packet forwarder protocol is a purposefully very basic protocol and was according to the comments in the source code, indented for demonstration purpose only. This explains the lack of basic security features like authentication of the gateway[54].
The gateway bridge now transmits the encrypted packets via MQTT to the network server. Message Queuing Telemetry Transport (MQTT) is a simple, open-source publish/subscribe network protocol developed by Andy Stanford-Clark and Arlen Nipper in 1999 for monitoring oil pipelines in the desert via satellite communications [55]. But it was the advent of IoT devices in recent years that made it a well-known protocol. In our case, we use Mosquitto as MQTT broker[56].

Since there are probably more LoRa networks from other operators in the area, the network server looks at the encrypted packet, and if there is a session key for the dynamic address given in the LoRa packet in its own lookup table, it tries to decrypt the packet with it. If the network server succeeds, it forwards the packet encrypted with a second key to the application server defined in the lookup table.

This is done using the publish/subscribe system from Redis. Redis is mainly an in-memory non-SQL database with a simple key-value store which is considered to be fast, but Redis also implements a publish/subscribe system which is used here [57]. For the lookup tables, the network and application server also need a PostgresSQL database [58].

The application server looks up the appropriate key in the lookup table. Once the application server has decrypted a packet, it forwards the data to a predefined interface. In our case, the application server makes a POST request to a predefined URL and passes the data as a parameter.

We use the default packet forwarder software for our gateway provided by the manufacturer, using the Semtech UDP Forwarder Protocol.

To visualize the GPS Position of the gateway on a map, we created a small C-Application, sniffing the serial GPS Data and forwarding them to the Raspberry Pi Controller.

The Raspberry Pi Controller is a custom Java Application using the well known Spring Framework[59], which combine and manages everything on the Raspberry Pi. It configures ChirpStack over the REST-API for the wanted BeePhones and uploads all the information gathered to an online SQL database.

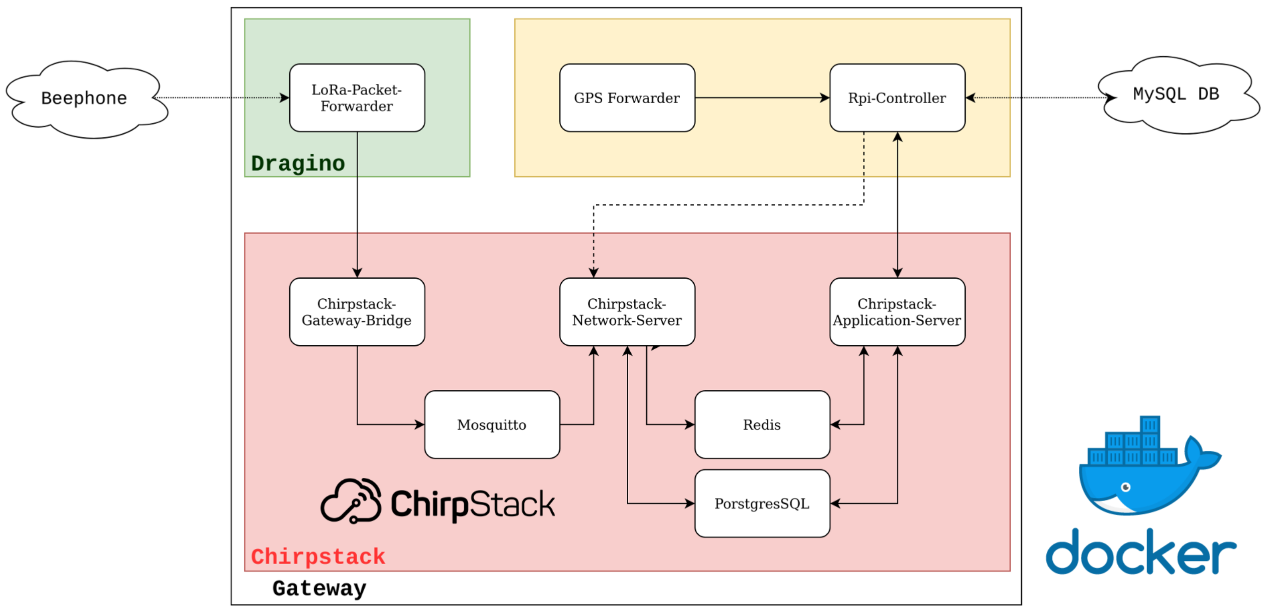


Figure 7: Software stack on gateway

5.3 Server / User Interface

In general, all the technologies for the server are given by the existing BeePhone environment, so it should be easy to integrate our results into the productive system. The online server features an SQL database to store all gathered information at one location. Additionally, our implementation features a website using the Angular framework, where the system displays all the information to the user. All information is provided over a Java Spring API. The Java Spring Application is where all the location algorithm is located. Here the map information gets combined with the received packet location and signal strength and the result gets forwarded to the user. The API gets the raw information from a MySQL database and all 3 elements together create a typical REST application (Figure 8).

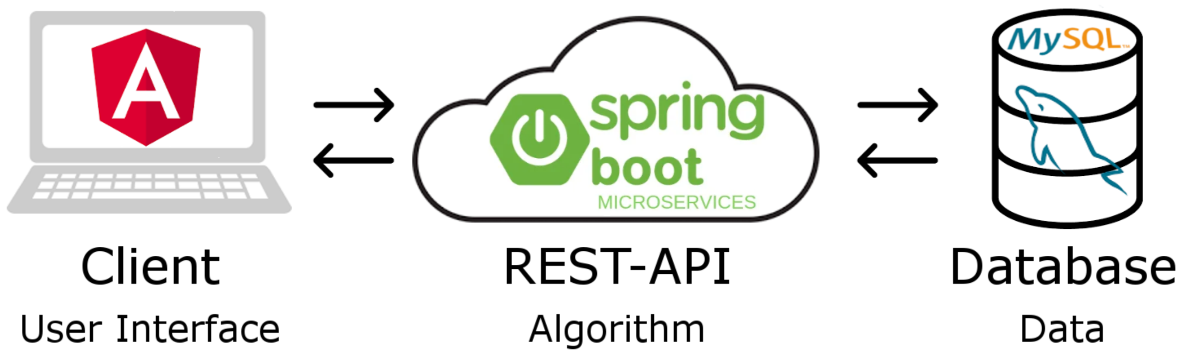
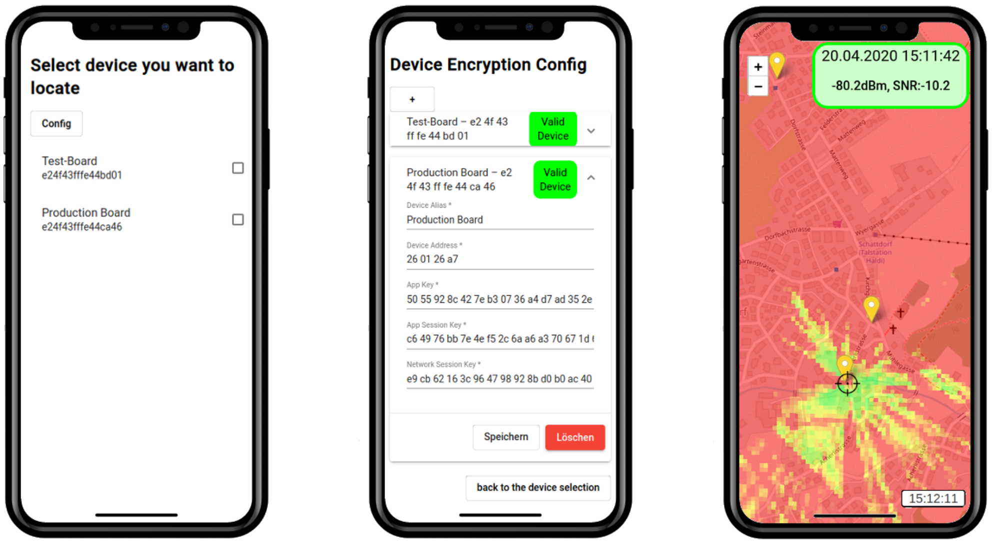


Figure 8: Server / User Interface Architecture

During development, the frontend features a dialogue where you can select the BeePhone you want to locate (Figure 9, left screen). Additionally, you have to provide the Device Address, the Network-Session-Key and the Application-Session-Key in order to intercept the packets sent by that BeePhone (Figure 9, middle screen). When integrating this into the existing BeePhone software stack, it should be easily replaceable by just a list of BeePhones registered onto the current user and gathering the encryption data over an API. Currently, all gateways are visible to all users. When integrating into the productive environment, this should be replaced by a filter, where just the gateways of a user are visible to them, the position of a beekeeper searching for a BeePhone should not be available for everyone. When a device is selected the user sees a map (Figure 9, right screen). Also, the current location of the gateway is rendered, to display the location of the user on the map. When receiving a packet from the BeePhone a pointer is set at the given location. When clicking/tapping on this pointer a small popup with information about this reception is displayed. If more than 2 receptions are present, then the algorithm automatically calculates the probability of the BeePhone and displays a kind of a heat map.


Select device you want to locate

Config

Test-Board
e24f43fffe44bd01

Production Board
e24f43fffe44ca46

Device Encryption Config

+

Test-Board – e2 4f 43 ff fe 44 bd 01 Valid Device

Production Board – e2 4f 43 ff fe 44 ca 46 Valid Device

Device Alias *
Production Board

Device Address *
26 01 26 a7

App Key *
50 55 92 8c 42 7e b3 07 36 a4 d7 ad 35 2e

App Session Key *
c6 49 76 bb 7e 4e f5 2c 6a a6 a3 70 67 1d

Network Session Key *
e9 cb 62 16 3c 96 47 98 92 8b d0 b0 ac 40

Speichern Löschen

back to the device selection

20.04.2020 15:11:42
-80.2dBm, SNR:-10.2

15:12:11

Figure 9: Screenshots of the User Interface

6 Evaluation

6.1 TDOA

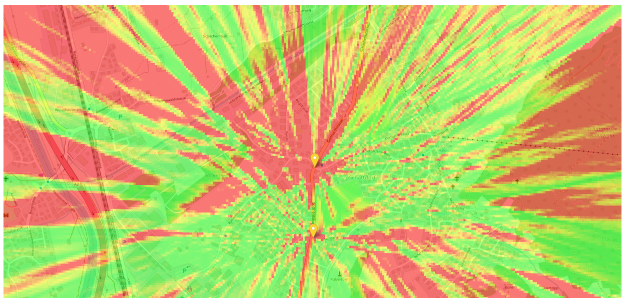
Due to the large time inaccuracy of $8\mu s$ from the SX1301 and $4.7\mu s$ from the Linux kernel, the accuracy of TDOA is low. The $8\mu s$ time deviation result in mean about $2.4km$ distance inaccuracy. If we want to get more accurate timestamps for the arrival of a packet, we would have to tap the I/Q lines from the SX1257 and process the signal. This would require digital singal processing and is not in the scope of this thesis. Since the SX1301, which does this job on the Dragino, already creates too much uncertainty.

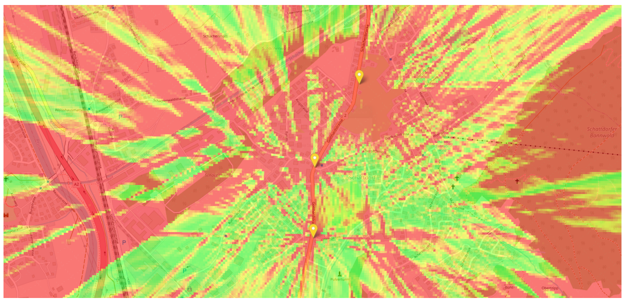
6.2 RSSI

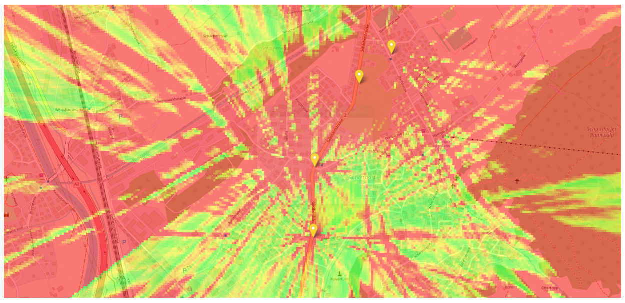
It's challenging to evaluate our algorithm based on facts since there are so many influences and it's practically impossible to take multiple measurements under the same conditions.

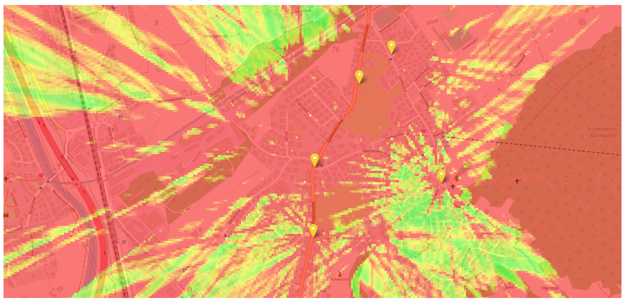
To evaluate the performance of our algorithm we first needed some data. We took a BeePhone and placed it at a fixed location. The BeePhone would then continuously send a packet every 100s. We then took our gateway and walked away a certain distance. We randomly stopped and waited for a received packet, since our gateway had difficulties receiving packets while we were walking. The random stops are important, to create different scenarios, since we logged all receptions and thus could test our algorithm afterwards with different parameters. We repeated this process with multiple different locations for our BeePhone to prevent over fitting parameters for one location.

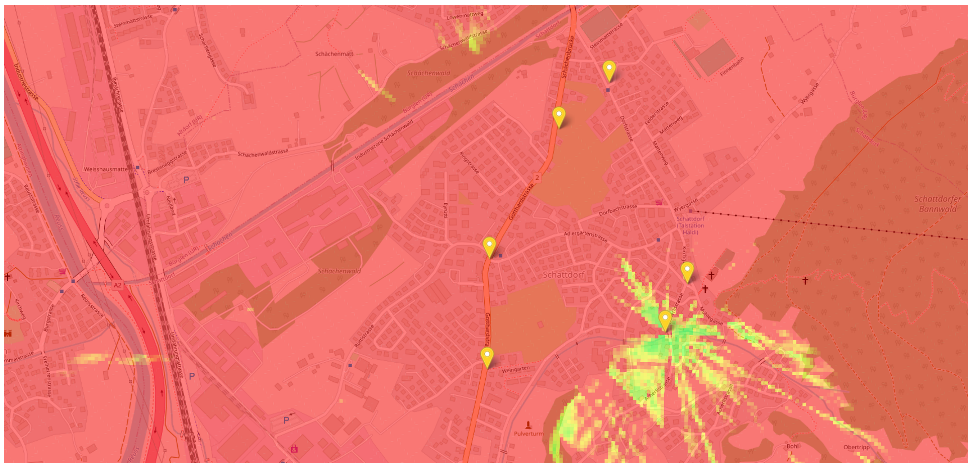
Afterwards, we evaluated over all data sets, if the algorithm improves by each additional measurement.
Just to give you an example of how the algorithm performs, Figure 10 shows the development of measurement 2 to 6, where 6 is already relatively close to the BeePhone.
In the example of Figure 10, we are, at the measurement points, between 100 and 900 meters away from the location of the BeePhone. The first 4 measurement points are about 300 to 350 meters away from each other. They are more or less aligned in a line. The second direction added with the 5th measurement points, helps the algorithm significantly in limiting where the


(a) With 2 measurements


(b) With 3 measurements


(c) With 4 measurements


(d) With 5 measurements



(e) With 6 measurements

Figure 10: Prediction with multiple measurements

BeePhone is located. Since the 6th point is only about 100 meters away from the BeePhone, we could strongly limit the area in which the BeePhone is located.

The example in Figure 10 clearly shows that the improvements are hugely influenced by the quality of the additional measurement point. If they are more or less in one line (like the first 4 points) then the quality of the prediction much slower increases than afterwards. But in general, the prediction quality increases with each measurement.

6.3 Limitations

But we also regognized some limitations. Most limitations come from the underlying LoRa technology and are already discussed in 2.1.3. Just to recall and visualize what happens if you ignore those limitations. Figure 11 is from the neighbourhood, because the wrong antenna position, very close to the back in a backpack, didn't allow higher ranges. We were walking quite fast so that the gateway had difficulties to receive the packets. The bad antenna position also leads to wrong and inconsistent RSSI values an thus the algorithm has difficulties.

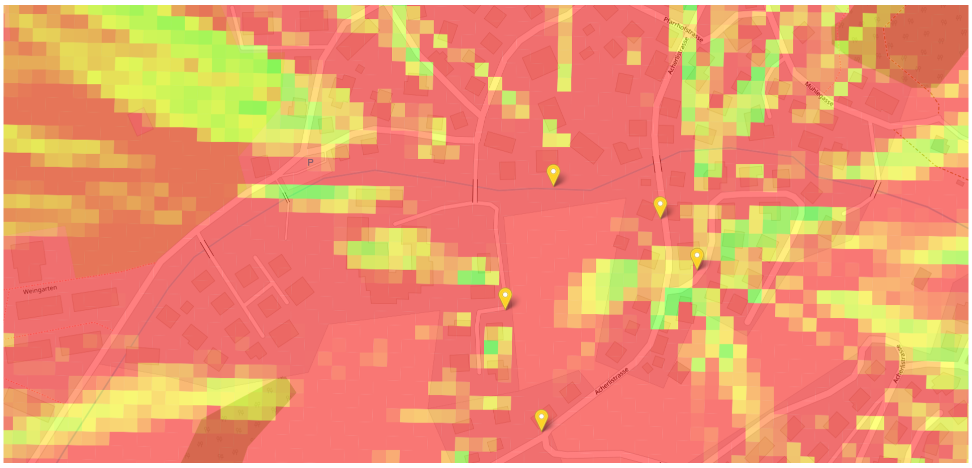


Figure 11: Inaccurate results on a short-range, when violating multiple limitations

The firmware of the LoRa-Chip on the BeePhone has an undocumented limitation, that the minimum payload is 10 bytes. In our case, one byte per package would be enough. Together with the 11 byte header, this would cut the transmission time nearly half, allowing for a more frequent transmission and less sensitive to gateway movement.

The algorithm has problems if it receives inconsistent values. For example, if a measuring point near the BeePhone has a very poor reception value, although the algorithm expects a good reception value. But also, several measured values at the same position with a strongly deviating reception value cause problems for the algorithm, because the variance is necessarily increases. In some cases, this can be partly compensated by several measurements at a similar location.

7 Conclusion and Future work

In this bachelor thesis, a system is presented to locate a stolen BeePhone more accurately than with previous solutions. We have extensively tested whether TDOA or RSSI using LoRa is the better option for our specific application. Due to the large time inaccuracy of at least $8\mu s$ (mean about $2,4km$), we could not achieve usable results with TDOA. The results with RSSI were much more promising as long as we respected the limitations of LoRa. We used publicly available map data to model the environment. With the help of these additional data sources, the path loss exponent can be estimated more accurately and the results could be improved. Especially an altitude model would be a useful extension.

There are multiple possibilities to extend this work. Just to give a few ideas, first, one could further improve the location prediction by adding a height map, by fine-tuning parameters, by adding additional data sources, and etc. Alternatively, one could try TDOA with different hardware, by processing the I/Q-Signal with an FPGA or something similar.

Bibliography

References

[1] Wikipedia contributors. *Varroa destructor — Wikipedia, The Free Encyclopedia.* [Online; accessed 20-July-2020]. 2020. URL: `https://en.wikipedia.org/w/index.php?title=Varroa_destructor&oldid=967523669`.

[2] Abilium GmbH. *BeeSmart - Die Platform für Imker.* [Online; accessed 05-July-2020]. 2020. URL: `https://www.beesmart.org/beephone`.

[3] Wikipedia contributors. *Angle of arrival — Wikipedia, The Free Encyclopedia.* [Online; accessed 5-July-2020]. 2020. URL: `https://en.wikipedia.org/w/index.php?title=Angle_of_arrival&oldid=965919117`.

[4] Wikipedia contributors. *Time of arrival — Wikipedia, The Free Encyclopedia.* [Online; accessed 5-July-2020]. 2019. URL: `https://en.wikipedia.org/w/index.php?title=Time_of_arrival&oldid=923877422`.

[5] Matt Knight. *Decoding the LoRa PHY (33c3) - YouTube.* [Online; accessed 10-February-2020]. 2020. URL: `https://www.youtube.com/watch?v=NoquBA7IMNc`.

[6] Qoitech. *How Spreading Factor affects LoRaWAN device battery life.* [Online; accessed 7-July-2020]. 2020. URL: `https://www.thethingsnetwork.org/article/how-spreading-factor-affects-lorawan-device-battery-life`.

[7] Wikipedia contributors. *LoRa — Wikipedia, The Free Encyclopedia.* [Online; accessed 7-July-2020]. 2020. URL: `https://en.wikipedia.org/w/index.php?title=LoRa&oldid=967476725`.

[8] LoRa Alliance. *Home page | LoRa Alliance.* [Online; accessed 25-July-2020]. 2020. URL: `https://lora-alliance.org/`.

[9] Juha Petäjäjärvi et al. "Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage". In: *International Journal of Distributed Sensor Networks* 13.3 (2017), p. 1550147717699412.

[10] CRFS. *How accurate is TDOA geolocation? - CEFS - Spectrum Monitoring and Geolocation.* 2020. URL: `https://www.crfs.com/blog/how-accurate-tdoa-geolocation/; %20https://www.crfs.com/wp-content/uploads/2017/09/heatmap_comparison_tdoa_network_bound.jpg`.

[11] Wikimedia Commons. *File:Trilateration.png — Wikimedia Commons, the free media repository.* [Online; accessed 05-February-2020]. 2020. URL: `https://commons.wikimedia.org/w/index.php?title=File:Trilateration.png&oldid=402089262`.

[12] Obinna Oguejiofor. "Outdoor localization system using RSSI measurement of wireless sensor network". In: *International journal of Innovative technology and exploring Engineering(IJITEE)* Volume 2 (Jan. 2013), Pages 1–6.

[13] Paul E. Black. *'Bresenham's algorithm', in Dictionary of Algorithms and Data Structures [online], Paul E. Black, ed. 21 January 2020.* [Online; accessed 20-July-2020]. 2020. URL: `https://www.nist.gov/dads/HTML/bresenham.html`.

[14] Jack Bresenham. "Algorithm for Computer Control of a Digital Plotter". In: *IBM System Journal* 4 (1965), pp. 25–30.

[15] Wikimedia Commons. *File:BresenhamLine2.png — Wikimedia Commons, the free media repository.* [Online; accessed 3-August-2020]. 2012. URL: `https://commons.wikimedia.org/w/index.php?title=File:BresenhamLine2.png&oldid=77541374`.

[16] Suining He and S-H Gary Chan. "Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons". In: *IEEE Communications Surveys & Tutorials* 18.1 (2015), pp. 466–490.

[17] Robin Wentao Ouyang et al. "Indoor location estimation with reduced calibration exploiting unlabeled data via hybrid generative/discriminative learning". In: *IEEE transactions on mobile computing* 11.11 (2011), pp. 1613–1626.

[18] José Luis Carrera Villacrés et al. "A Particle Filter-based Reinforcement Learning Approach for Reliable Wireless Indoor Positioning". In: (2019). DOI: `10.7892/BORIS.130748`. URL: `https://boris.unibe.ch/130748/`.

[19] José Luis Carrera Villacrés. "Indoor Positioning and Tracking Methods For Mobile Wireless Devices". In: (2019). DOI: `10.7892/BORIS.133396`. URL: `https://boris.unibe.ch/133396/`.

[20] Zoubin Ghahramani. “An introduction to hidden Markov models and Bayesian networks”. In: *Hidden Markov models: applications in computer vision*. World Scientific, 2001, pp. 9–41.

[21] David Madigan et al. “Bayesian indoor positioning systems”. In: *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 2. IEEE. 2005, pp. 1217–1227.

[22] Yifeng Zhou, Jun Li, and Louise Lamont. “Multilateration localization in the presence of anchor location uncertainties”. In: *2012 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2012, pp. 309–314.

[23] Wei Liu, Enqing Dong, and Yang Song. “Robustness analysis for node multilateration localization in wireless sensor networks”. In: *Wireless Networks* 21.5 (2015), pp. 1473–1483.

[24] Wade H Foy. “Position-location solutions by Taylor-series estimation”. In: *IEEE Transactions on Aerospace and Electronic Systems* 2 (1976), pp. 187–194.

[25] Minghui Li and Yilong Lu. “Angle-of-arrival estimation for localization and communication in wireless networks”. In: *2008 16th European Signal Processing Conference*. IEEE. 2008, pp. 1–5.

[26] Martin Hurtado and Arye Nehorai. “Performance analysis of passive low-grazing-angle source localization in maritime environments using vector sensors”. In: *IEEE Transactions on Aerospace and Electronic Systems* 43.2 (2007), pp. 780–789.

[27] Chang Wang et al. “Convex combination based target localization with noisy angle of arrival measurements”. In: *IEEE Wireless Communications Letters* 3.1 (2013), pp. 14–17.

[28] Oren Jean and Anthony J Weiss. “Geolocation by direction of arrival using arrays with unknown orientation”. In: *IEEE transactions on signal processing* 62.12 (2014), pp. 3135–3142.

[29] Don J Torrieri. “Statistical theory of passive location systems”. In: *IEEE transactions on Aerospace and Electronic Systems* 2 (1984), pp. 183–198.

[30] Regina Kaune. “Accuracy studies for TDOA and TOA localization”. In: *2012 15th International Conference on Information Fusion*. IEEE. 2012, pp. 408–415.

[31] Fiona Fletcher, Branko Ristic, and Darko Musicki. “Recursive estimation of emitter location using TDOA measurements from two UAVs”. In: *2007 10th International Conference on Information Fusion*. IEEE. 2007, pp. 1–8.

[32] Regina Kaune. “Gaussian mixture (GM) Passive localization using time difference of arrival (TDOA)”. In: *Informatik 2009–Im Focus das Leben* (2009).

[33] Rabih Chrabieh et al. “Enhanced Multilateration Methods With A Global Approach”. In: *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE. 2020, pp. 1070–1078.

[34] Christian Steffes et al. “Determining Times Of Arrival Of Transponder Signals In A. Sensor Network Using Gps Time Synchronization.” In: *GI-Jahrestagung*. Citeseer. 2011, p. 481.

[35] Xinwei Wang et al. “Localization in wireless ad-hoc sensor networks using multilateration with RSSI for logistic applications”. In: *Procedia Chemistry* 1.1 (2009), pp. 461–464.

[36] Hussein Kwasme and Sabit Ekin. “RSSI-based localization using LoRaWAN technology”. In: *IEEE Access* 7 (2019), pp. 99856–99866.

[37] Lei Wang et al. “WSN multilateration algorithm based on Landweber iteration”. In: *2009 9th International Conference on Electronic Measurement & Instruments*. IEEE. 2009, pp. 1–250.

[38] Jan Blumenthal et al. “Weighted centroid localization in zigbee-based sensor networks”. In: *2007 IEEE international symposium on intelligent signal processing*. IEEE. 2007, pp. 1–6.

[39] Paramvir Bahl and Venkata N Padmanabhan. “RADAR: An in-building RF-based user location and tracking system”. In: *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*. Vol. 2. Ieee. 2000, pp. 775–784.

[40] Islam Fayez Abd Alyafawi, Desislava Cvetanova Dimitrova, and Torsten Braun. “SDR-based Passive Indoor Localization System for GSM”. In: (2014). DOI: 10.7892/BORIS.53704. URL: https://boris.unibe.ch/53704/.

[41] Bernat Carbonés Fargas and Martin Nordal Petersen. “GPS-free geolocation using LoRa in low-power WANs”. In: *2017 global internet of things summit (Giots)*. IEEE. 2017, pp. 1–6.

[42] Mateo Marcelić et al. “Determining Location in LPWAN using Multilateration”. In: *2019 2nd International Colloquium on Smart Grid Metrology (SMAGRIMET)*. IEEE. 2019, pp. 1–4.

[43] Mohd Ismifaizul Mohd Ismail et al. “An RSSI-based Wireless Sensor Node Localisation using Trilateration and Multilateration Methods for Outdoor Environment”. In: *arXiv preprint arXiv:1912.07801* (2019).

[44] Cosmin Rus et al. “LoRa communication and geolocation system for sensors network”. In: *MATEC Web of Conferences*. Vol. 305. EDP Sciences. 2020, p. 00043.

[45] Victor Delafontaine et al. “Drone-aided Localization in LoRa IoT Networks”. In: *arXiv preprint arXiv:2004.03852* (2020).

[46] Wikipedia contributors. *Serial Peripheral Interface — Wikipedia, The Free Encyclopedia*. [Online; accessed 6-August-2020]. 2020. URL: `https://en.wikipedia.org/w/index.php?title=Serial_Peripheral_Interface&oldid=971155390`.

[47] OGP Geomatics Committee. *CH1903/LV03*. 2020. URL: `https://epsg.org/crs_21781/CH1903-LV03.html`.

[48] Sacha Bron Valentin Minder. *Swisstopo-WGS84-LV03*. `https://github.com/ValentinMinder/Swisstopo-WGS84-LV03`. 2016.

[49] OGP Geomatics Committee. *CH1903+/LV95*. 2020. URL: `https://epsg.org/crs_2056/CH1903-LV95.html`.

[50] QGIS contributors. *Welcome to the QGIS project!* 2020. URL: `https://www.qgis.org`.

[51] GeoTools contributors. *GeoTools The OpenSource Java GIS Toolkot - GeoTools*. 2020. URL: `https://geotools.org/`.

[52] USIAndyrich. *USI_I-NUCLEO-LRWAN13*. `https://github.com/USIWP1Module/USI_I-NUCLEO-LRWAN1`. 2020.

[53] Shan-x. *encryption - Security of an IoT network using AES (LoRaWAN) - inofmation Security Stack Exchange*. 2020. URL: `https://security.stackexchange.com/questions/126987/security-of-an-iot-network-using-aes-lorawan`; `https://i.stack.imgur.com/Zs9CV.png`.

[54] Michael Coracin. *packet_forwarder/PROTOCOL.TXT*. `https://github.com/Lora-net/packet_forwarder/blob/master/PROTOCOL.TXT`. 2017.

[55] Wikipedia contributors. *MQTT — Wikipedia, The Free Encyclopedia*. [Online; accessed 20-July-2020]. 2020. URL: `https://en.wikipedia.org/w/index.php?title=MQTT&oldid=961832124`.

[56] Eclipse Foundation, Inc. *Eclipse Mosquitto*. 2020. URL: `https://mosquitto.org`.

[57] Redis contributors. *Redis*. [Online; accessed 20-July-2020]. 2020. URL: `https://redis.io`.

[58] PostgresSQL contributors. *PostgresSQL: The world’s most advanced open source database*. [Online; accessed 20-July-2020]. 2020. URL: `https://www.postgresql.org/`.

[59] VMware, Inc. *Spring | Home*. 2020. URL: `https://spring.io`.