Thomas Barker

Geolocation in a Rainforest

Computer Science Tripos – Part II

Pembroke College

May 13, 2022

Declaration

I, Thomas Barker of Pembroke College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose. I am content for my dissertation to be made available to the students and staff of the University.

Signed: Thomas Jake Barker

Date: 13th May 2022

Proforma

Candidate Number:	**2392F**
Project Title:	**Geolocation in a Rainforest**
Examination:	**Computer Science Tripos – Part II, 2022**
Word Count:	**10612**[1]
Code Line Count:	**2127** [2]
Project Originator:	Dr S. Keshav
Supervisor:	Dr S. Keshav

Original Aims of the Project

This project aims to build a system to localise a device using Bluetooth Low Energy signal strength values. I wanted to implement a typically indoor positioning approach to see how it performed in an outdoor environment.

Work Completed

First, I constructed a prototype Bluetooth beacon before writing a configuration script using BLE commands. Next, I implemented the method for the target device to retrieve the radio signal strength information from the beacon. After this, I implemented the localisation algorithms including a Gaussian process based method. This was based on a paper [8]. I then implemented statistical techniques to reduce the error including a Kalman Filter. After this point, I added some extension algorithms notably creating my own hybrid algorithms which combines a Gaussian process with other techniques. Finally, I evaluated the systems accuracy in different environments.

Special Difficulties

None

[1]Calculated using `detex diss.tex | tr -cd '0-9A-Za-z \n' | wc -w` (for the chapters between the introduction and conclusion inclusive)

[2]Computed as the sum of all the lines of code in the git repository

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Challenges of the Project 2

1.3 Previous Work 2

1.3.1 Time of Flight 2

1.3.2 Radio Signal Strength 3

2 Preparation **5**

2.1 Concepts 5

2.1.1 Radio Signal Strength 5

2.1.2 Fingerprinting 6

2.1.3 Localisation algorithms 6

2.1.4 Filters 8

2.2 Factors affecting signal strength 10

2.2.1 Flight Path Interference 10

2.2.2 Emission power 10

2.2.3 Antenna Position 10

2.2.4 Noise in measurements 11

2.3 Bluetooth Low Energy 11

2.4 Gaussian Process 13

2.4.1 Detail 13

2.4.2 Learning Kernel Hyperparameters 14

2.4.3 Application to signal strength modelling 14

2.5 Dataset 15

2.5.1 Metrics for Evaluation 15

2.6 Requirements Analysis 16

2.7 Tools Used 17

2.7.1 Hardware 17

2.7.2 Software 17

2.8 Software engineering techniques 18

2.8.1 Development Environment 18

2.8.2 Backup Strategy 19

2.9 Personal Starting Point 19

3 Implementation **21**
3.1 Hardware Implementation . . . 21
3.1.1 Hardware . . . 21
3.1.2 Configuration for positioning . . . 22
3.2 Positioning Models . . . 22
3.2.1 General Positioning Algorithm . . . 22
3.2.2 Proximity . . . 23
3.2.3 K-Nearest Neighbours . . . 23
3.2.4 Path Loss . . . 23
3.2.5 Gaussian Process . . . 23
3.2.6 Algorithm . . . 25
3.2.7 Gaussian Hybrid (Extension) . . . 25
3.2.8 Weighted KNN (Extension) . . . 26
3.3 Obtaining and processing the measurement data . . . 27
3.3.1 Filtering . . . 27
3.4 Navigating the Repository . . . 28

4 Evaluation **29**
4.1 Experimental Setup . . . 29
4.1.1 Obtaining the Ground Truth . . . 29
4.1.2 Training Data . . . 30
4.1.3 Evaluation Data . . . 30
4.2 Measuring the Criteria . . . 30
4.3 Analysis of Results . . . 31
4.3.1 General Model Comparison . . . 31
4.3.2 Gaussian Parameters . . . 34
4.3.3 Impact of Filtering . . . 35
4.3.4 How density affects performance (Extension) . . . 36
4.3.5 Improving Performance . . . 37

5 Conclusions **39**
5.1 Project Achievements . . . 39
5.2 Lessons Learnt . . . 39
5.3 Future Development . . . 40

Bibliography **40**

A Tables **43**

B Experiments **47**

C Project Proposal **49**

Chapter 1

Introduction

Radiolocation is the process of finding the location of something through the use of radio waves. A common approach measures the distance to an emitter by using the difference in the power of the received signal strength (RSSI) when compared with the originating signal strength. Another approach to calculate the distance uses the time of flight from emission to reception [17]. Data from multiple emitters at different known locations are then combined to estimate a position. Radiolocation has many applications from navigation to tracking objects.

Traditionally GPS is used to provide radiolocation outside with successful results; however in some environments such as the rainforest this provides inaccurate results due to environmental factors. This project aims to build a Bluetooth low energy (BLE) localisation system to produce more accurate results. It will also evaluate the performance of the implemented models against baseline algorithms.

1.1 Motivation

As the world tries to slow the speed of climate change, deforestation is having a detrimental impact on carbon emissions. This has been seen in recent years with deforestation being one of the contributing factors to the Amazon rainforest being turned from a carbon sink into a carbon source [14]. It is more important than ever to protect our natural resources from destruction. Some research suggests that supplying indigenous communities with satellite data we can significantly reduce illegal deforestation and mining in the Peruvian Amazon rainforest[20]. By enabling communities to accurately map important resources we may be able to help restrict further damage to the environment. Precise mapping is also useful to measure field plots in environmental science.

Global Navigation Satellite Systems (GNSS), such as GPS, have allowed for accurate, positioning outdoors, but the environmental conditions of rainforests means that accuracy is reduced leading to a positional error in the range of 10-30 metres for GPS [12]. The radio signals emitted by the satellites are too weak to penetrate the dense vegetation and this combined with the high humidity environment makes it difficult to get an accurate location fix.

This low accuracy is insufficient to conduct fieldwork, so an alternative positioning method must be used to provide better accuracy. GPS signals are also unable to provide accurate location fixes indoors, this is due to the construction materials reducing signal penetration and electromagnetic interference from personal computing devices. These factors produce an environment quite similar to a rainforest. This project aims to apply techniques used for indoor positioning to build a system for outdoor localisation which is more accurate than GNSS systems in harsher environmental conditions such as rainforests.

1.2 Challenges of the Project

Radio signals are particularly susceptible to noise. This makes creating a mapping system from them complex due to the variation in signal strength values. Smoothing these values can be computationally expensive.

Another challenge was that I had no previous experience programming with wireless signals. This lack of experience meant I had to quickly learn about the field.

1.3 Previous Work

Positioning is a mature field, with many proposed techniques and technologies. In this section we will focus on radio positioning, first with a quick look at time of flight systems (ToF). Then Radio signal strength (RSS) based signals with a specific focus on empirical fingerprinting.

1.3.1 Time of Flight

Time of Flight based methods work by having time-synchronised emitters or beacons at known locations and a receiving device we want to locate. The emitting device emits a signal to the receiving devices with a precise timestamp. The time difference between emission and reception relates to the distance between the two and can be used to locate the receiver using a method such as trilateration or the min-max method.

$$\delta d = c \delta t \tag{1.1}$$

where c is the speed of light.

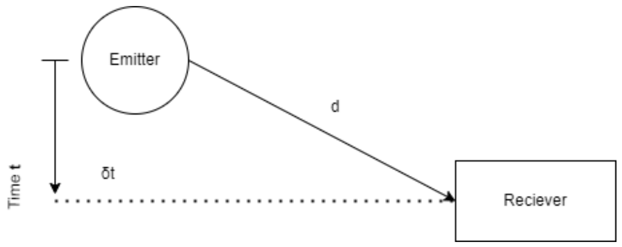

Figure 1.1: Time of Flight diagram: One-way range measurement(Synchronised Clocks)

Trilateration can be used to estimate a device's location, it creates a circle (2D) or sphere(3D) around each beacon, with a radius proportional to the estimated distance, the intersection of the circles or spheres is then the estimated position. You need at least 3 beacons in 2D and at least 4 in 3D.

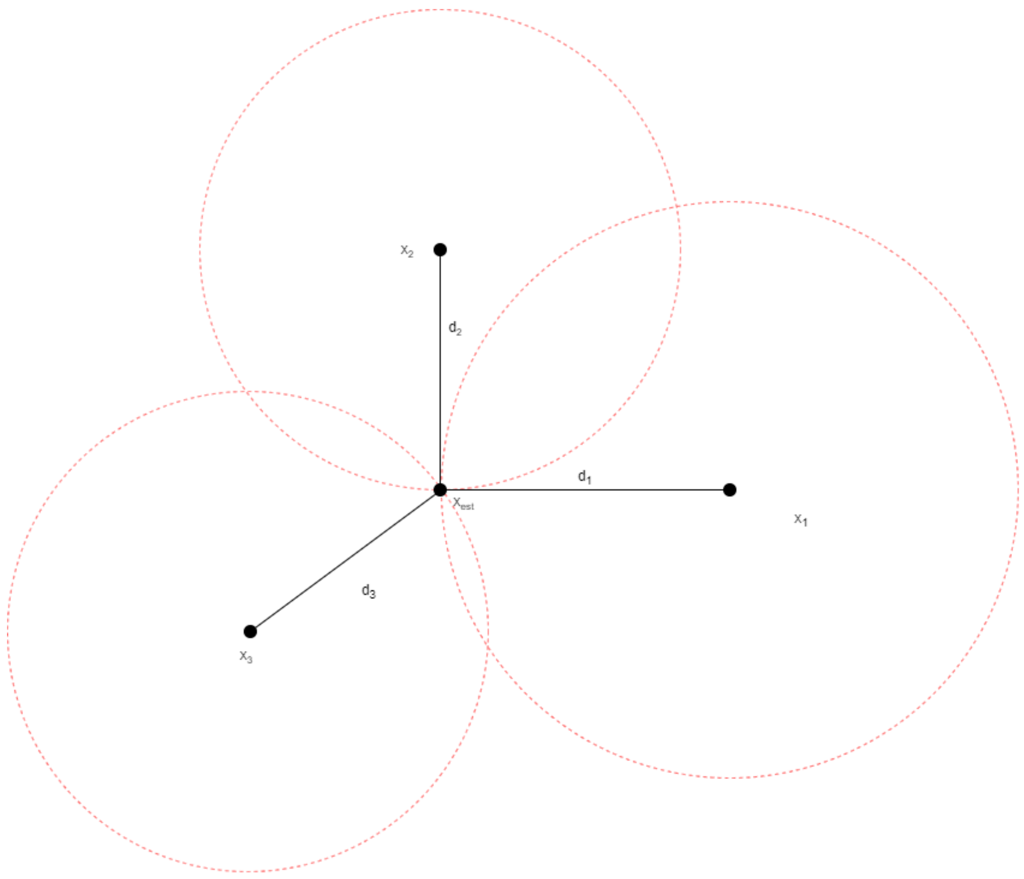

Figure 1.2: 2D Trilateration example

GPS and other GNSSs are examples of a ToF system with moving satellite transmitters. Ultra-wideband positioning is also a notable example of ToF which can work indoors. However, its high cost makes it expensive to build a positioning system with at this point in time [11].

1.3.2 Radio Signal Strength

Radio Signal Strength positioning methods use the difference in the power of the received signal strength (RSSI) as compared to the known originating signal strength.

Proximity

The simplest method is proximity based. This method uses the position of the beacon with the largest collected RSSI value to approximate the position.

Fingerprinting

Here positioning is performed using a previously constructed radio map. This has been shown to provide more accurate results than proximity-based systems [8]. This has historically been used with Wi-Fi signals [5], but in the last few years location fingerprinting with Bluetooth low energy beacons for use indoors has been explored.

In 2015 Robert Harle and Ramsey Faragher from the University of Cambridge demonstrated how location fingerprinting can be done with Bluetooth low energy beacons. In there implementation they used Gaussian process regression to estimate continuous signal maps from discrete BLE RSS data, they then sampled this map at to form a grid of points, before selecting the most likely point. [8] Other algorithms have been shown to work for indoor fingerprinting including K Nearest Neighbour regression (KNN) and Weighted KNN regression (WKNN) [19]. These methods are based off the 'distance' of RSS values from training points. They then use the lowest k distances to calculate an estimate.

Filtering

RSSI values have a lot of process noise. Smoothing these values can make them more suitable for use in positioning. There are a variety of filtering techniques from simple moving average filters [13] to particle filters [21]. Filters have also been used to incorporate past position estimates to to inform the most likely next position estimate.

Filtering algorithms attempt to use noisy/partial data to estimate the current value. This data consists of current and past observations. Different filtering methods have varied effectiveness based on their complexity with more complex filters often providing better results at the cost of performance[13].

Chapter 2

Preparation

This chapter introduces the core ideas behind the frameworks used in this project. It also discusses how radio signals can vary in strength. A detailed requirements analysis and a discussion of the tools used for the implementation are also included.

2.1 Concepts

This section is designed to give the reader a brief introduction to radio-based positioning before giving an overview of the Gaussian process model in the context of RSSI-based positioning.

2.1.1 Radio Signal Strength

Radio signal strength refers to the power output received by a reference antenna. In low power systems such as Bluetooth, the beacons signal strength is expressed in decibels below a reference level of one milliwatt (dBm).

Received signal strength (RSS) is the strength of a received signal measured at the receivers' antenna. Received signal strength indicator (RSSI) quantifies this. RSSI is an indication of the level of power received by the receiving radio after antenna and cable loss. The greater the RSSI value the stronger the signal. RSSI values are commonly expressed in negative form (e.g -42 dBm = $10^{-4.2}$mW), the closer the value to 0, the stronger the received signal is.

RSSI and Distance

Path loss is the decline in power density of an electromagnetic wave as it propagates through space. By modelling path loss, we can link distance to signal power and hence RSSI.

A common way of modelling this is the log-distance path loss model. Below is the equation for this model where r is the estimated received power, r_o is the received RSSI at a point p in dbm, γ is the path loss exponent, d is the distance of the mobile device from the transmitter, and d_0 is the distance from the transmitter to point p. Point p can be any known position provided the distance to the transmitter and the RSSI value at the point are known (d_0 and r_0 respectively).

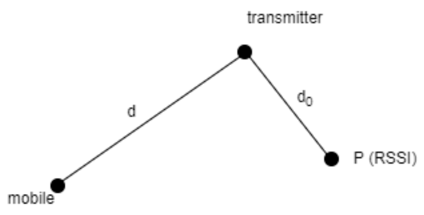

Figure 2.1: log-distance model setup

$$r = r_0 - 10\gamma \log_{10}(\frac{d}{d_0}) \tag{2.1}$$

This can then be rearranged to get an estimate for distance from the receiver:

$$d = d_0 10^{\frac{r_0 - r}{10\gamma}} \tag{2.2}$$

2.1.2 Fingerprinting

An RF-based fingerprinting method can usually be broken down into two phases: offline and online [15]. During the offline stage the environment is surveyed. This involves obtaining the location coordinates and respective signal strengths from the nearby beacons. During the online stage a location positioning technique uses the currently observed (live) signal strengths and the training data to estimate the location of a target device.

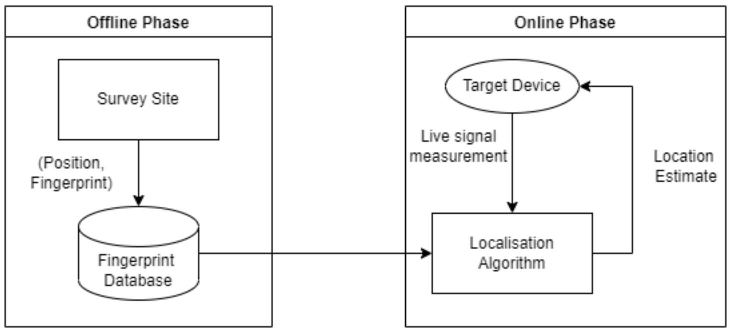

Figure 2.2: Standard fingerprinting algorithm overview

2.1.3 Localisation algorithms

Localisation algorithms take a live signal measurement and the training data and return an estimate for the location of the target device. Many of these algorithms use a model to calculate distances between known points and the target position. Once these distances have been obtained you can then use several methods to estimate the target position. A few of these are discussed below.

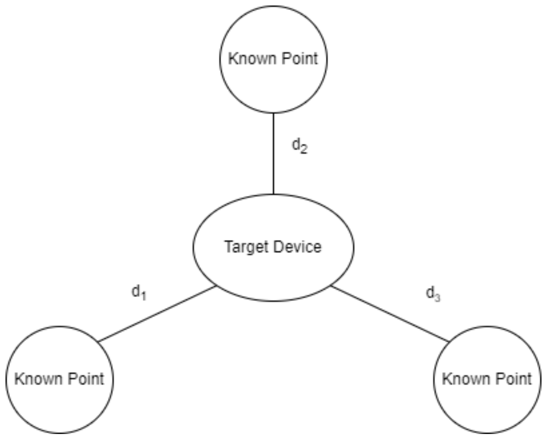

Figure 2.3: General setup for 3 known points, d_i is the distance between the target device and a known point

Min-Max Method

The min-max or bounding-box method takes distance estimates and their corresponding known positions. The target node then constructs a bounding box around each reference node, where the reference node is at the center of the box and the box length is $2d_i$, where d_i is estimated distance between the point x_i and the target device. The target node determines the intersection of these boxes with its boundaries given by $x_{min}, x_{max}, y_{min}$ and y_{max}. The center of the intersection box is then the estimated target position. [22]

$$\begin{aligned} x_{min} &= max_{1<=i<=n}(x_i - d_i) \\ y_{min} &= max_{1<=i<=n}(y_i - d_i) \\ x_{max} &= min_{1<=i<=n}(x_i + d_i) \\ y_{max} &= min_{1<=i<=n}(y_i + d_i) \end{aligned}$$

$$\begin{aligned} x_{est} &= \frac{(x_{min} + x_{max})}{2} \\ y_{est} &= \frac{(y_{min} + y_{max})}{2} \end{aligned}$$

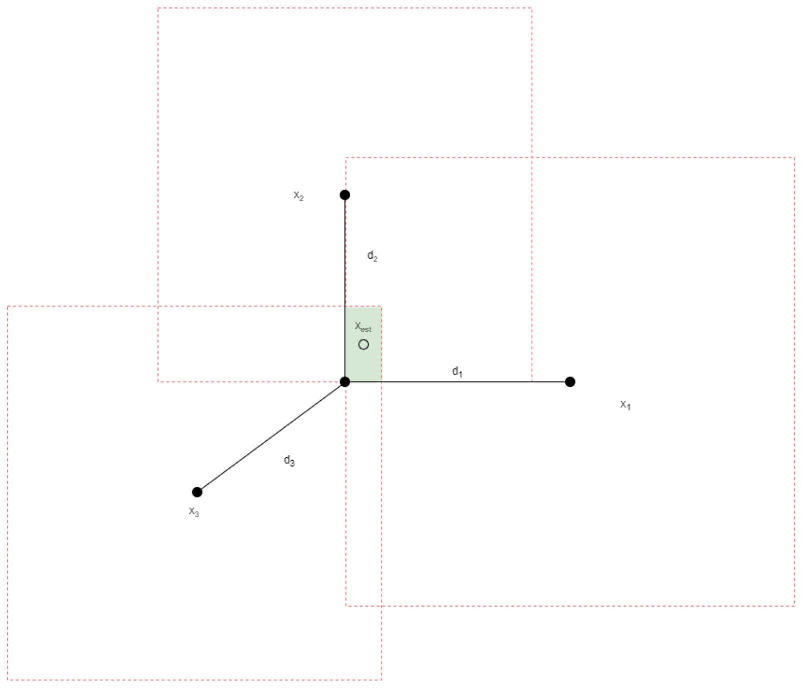

Figure 2.4: Simple 2D bounding box example

Weighted Centroid Localisation

The Weighted Centroid Localisation (WCL) method estimates the target position by calculating the weighted centroid of the coordinates of the reference nodes at known positions. The weight are created such that a larger weight means that a reference node contributes more to the position estimate. In the case of distances this means that w_i and d_i are inversely proportional[22]. If only one known point is used it assigns the estimated position to that point.

$$x_{est} = \sum_{i=1}^{n} \frac{w_i \times x_i}{\sum_{i=1}^{n} w_i} \tag{2.3}$$

$$w_i = \frac{1}{d_i} \tag{2.4}$$

2.1.4 Filters

Filters reduce statistical noise and other inaccuracies and produce an estimate of unknown variables. This tends to be more accurate than a single measurement on its own.

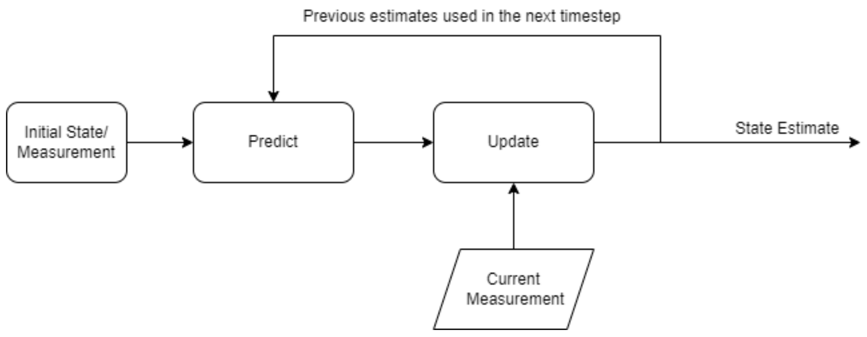
Previous estimates used in the next timestep
Initial State/ Measurement
Predict
Update
State Estimate
Current Measurement

Figure 2.5: General operation of a filter

Simple Moving Average Filter

This filter is the unweighted mean of the previous k data-points. This is cheap to compute with a FIFO buffer of size k.

$$SMA_k = \frac{1}{k} \sum_{i=n-k+1}^{n} p_i \tag{2.5}$$

Kalman Filter

The Kalman filter produces an estimate of the systems state as an average of the system's predicted state and of the new measurement using a weighted average. These weights allow values with smaller estimated uncertainty to be "trusted" more. Covariance, a measure of estimated uncertainty in a state, is used to calculate the weights. The result of the weighted average is the next state estimate. The covariance is updated each iteration. This filter only requires the last estimate compared to the entire history in some methods. The Kalman filter is optimal assuming that errors have a Gaussian distribution and if the process and measurement covariances are known, the Kalman filter is the best linear estimator in regard to minimum mean-square-error[10].

This model assumes the true state at time k evolves from the state at $(k-1)$ according to the below model:

$$x_k = F_k x_{k-1} + B_k u_k + w_k \tag{2.6}$$

Where F_k is the state transition model that is applied to the previous state x_{k-1}, B_k is the control-input model which is applied to the control vector u_k, which represents the controlling input at time k. w_k is the process noise, which is assumed to be drawn from a zero mean multivariate normal distribution.

At time k an observation z_k of the true state x_k is made using the below equation:

$$z_k = H_k x_k + v_k \tag{2.7}$$

Where H_k is the observation model, this maps the true state space into the observed space and v_k is the observation noise, which we assume is a zero mean Gaussian white noise.

Many systems are not linear however, but generalisations such as the Extended Kalman filter have been developed for non-linear systems, but these are not optimal estimators for non-linear systems.

When applied to signal strength measurements we only need a 1D Kalman filter.

2.2 Factors affecting signal strength

Various factors lead to the variance in signal strength at points. By understanding these factors we can build better positioning systems by informing our algorithm design choices. For example introducing filtering of RSSI values or how to position a beacon to provide the best range. A few of these factors are discussed below.

2.2.1 Flight Path Interference

Body Interference

Radio waves cannot pass through electrical conductors such as water and metals. As a significant percent of the human body is water(around 50-65% for men) it follows that a body cutting through a radio wave will lead to lower RSSI value [22].

Multi-path interference

Multipath interference occurs when a radio signal from a transmitter arrives at a receiver via two or more routes. This can be caused by a variety of factors such as atmospheric ducting, reflection and refraction. Multipath interference may cause a radio signal to become too weak in certain areas to be received. This effect is more prevalent indoors, due to the number of walls. In the rainforest this can occur when signals reflect off water particles in the humid air.

2.2.2 Emission power

The emission power is directly proportionate to the RSSI value meaning that a higher emission power will lead to an increase in the RSSI value.

2.2.3 Antenna Position

The position of the antenna can affect the RSSI values. If the antenna is directional, this can lead to varied measurement depending on the rotation. To investigate this I performed an investigation into how rotation of the antenna affects the RSSI values at 1 m from the receiver. To set up this experiment I had a beacon indoors 1 m from the target device, the only changing variable was the rotation of the beacon and the time (Appendix 2). The below figure demonstrates how the different angles have notably different average RSSI values. This change may be due to multipath interference. From the experiment I was able to ensure to implement more measures in the evaluation such as ensuring the beacons were at level heights and keeping the beacon positions constant throughout.

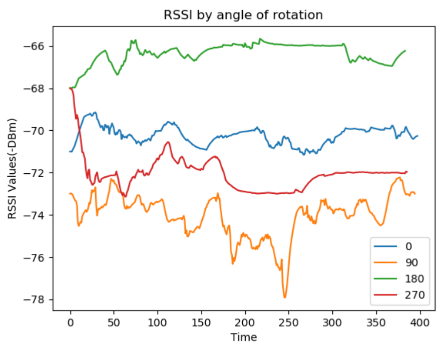

Figure 2.6: Figure demonstrating how the angle affects the RSSI value (Appendix 2)

2.2.4 Noise in measurements

Even if we do not move our devices or interfere directly with the signal the strength would not be constant. This comes from electromagnetic noise.

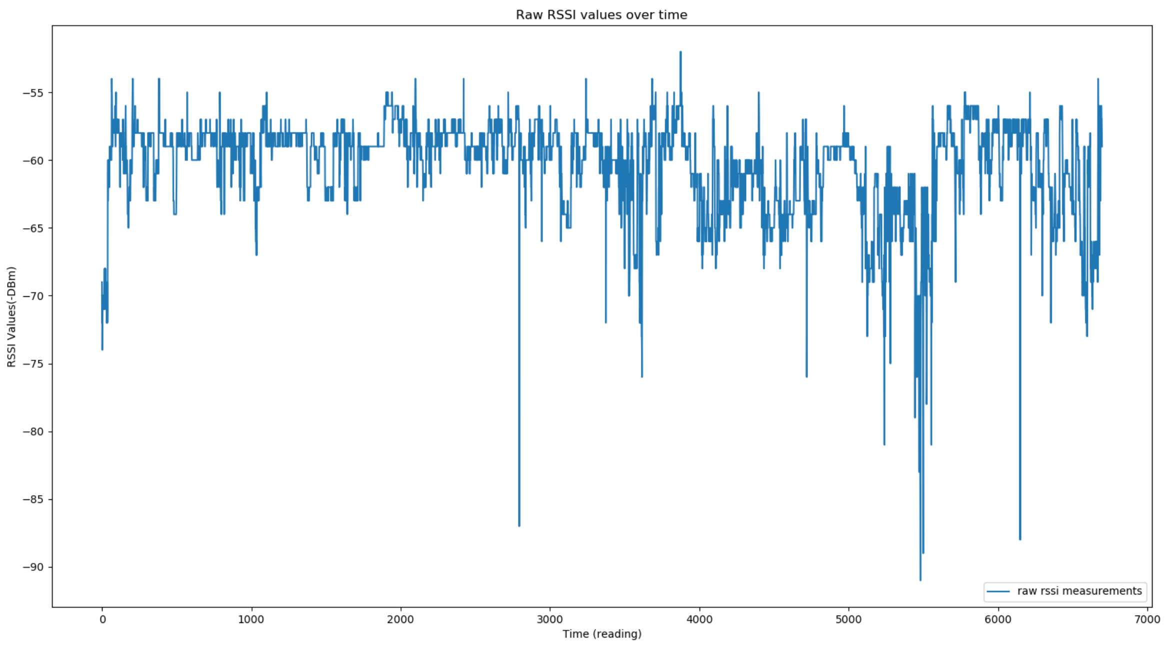

Figure 2.7: Figure demonstrating the variance in RSSI readings over time at 1 m from transmitter (Appendix 2)

2.3 Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a wireless communication protocol that has been designed for low power operation. It uses 40 channels, each 2 MHz in width and uses the 2.4 GHz radio band [8]. The protocol consists of short duration messages which reduces battery consumption compared to Bluetooth Classic. Messages in BLE are either data or advertisement messages.

Advertisement messages are broadcast messages that are used for device discovery and contain a payload such as the current sensor state. Advertisement messages are needed to enable communication between two devices. This makes them useful for dedicated and opportunistic positioning. Advertisement messages are broadcast on 3 channels. These are commonly given the numbers 37, 38 and 39 and are widely spaced at 2402 MHz, 2426 MHz and 2480 MHz respectively. Notably these channels have different mean RSSI levels [8]. This is caused by multipath interference and uneven channel gain. Uneven channel gain occurs as antenna usually do not have a standard response across the entire 2.4GHz band this gain is exacerbated by the widely spaced channels. By default, our receiving device operates a scan cycle which cycles over the three advertising channels, pausing to listen to each one. Due to the unknown cycle and pause time it makes BLE unsuitable for ToF based systems. We are also not guaranteed to be able to map an advertising packet to the channel it was sent on.

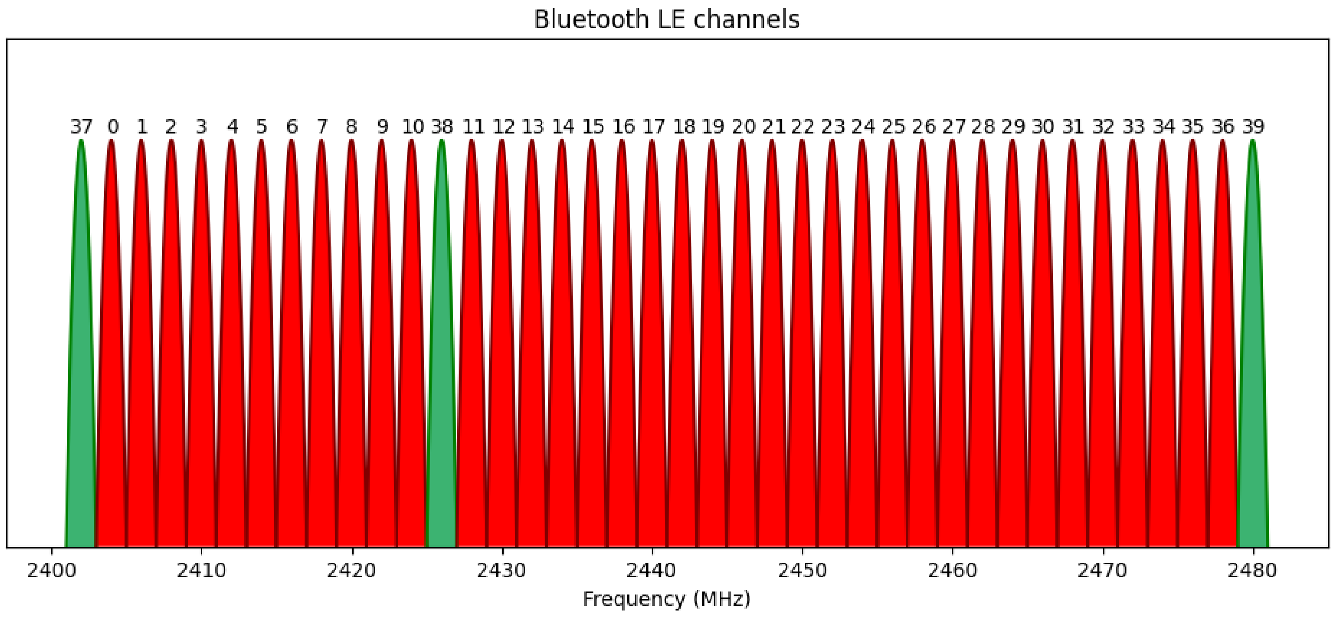

Figure 2.8: The 40 BLE channels. BLE advertising only occurs on channels 37,38,39. This figure was inspired by [8]

BLE scans at a receiver continue indefinitely and report advertisement packets as soon as they are received. Due to this we form the current fingerprint from the advertising packets received during a set time period (window duration). In each window we may have advertising packets from any or all of the three channels.

The beacon advertising period or beacon interval is the time elapsed between advertising packets being sent by a beacon. A short period uses more power, but sends more packets.

The window duration is the time for which the receiver listens for advertisement packets before forming a window. If the window duration is greater than the beacon advertising period we may receive more than one advertisement packet for a single beacon per window. A longer duration may lead to an increased accuracy of reading, but the user may have travelled further in that time. For example at a 1s window duration a person walking at 1.5 m/s may have moved 15 m since the window start.

Advertising devices can be configured to be connectable or not. If a device is connectable an advertising device allows a connection to be established. Advertising devices can also be configured with directed advertising or undirected advertising. In directed advertising connection requests are accepted from known peer devices. While in undirected connection requests are accepted from any peer device.

2.4 Gaussian Process

A Gaussian process is a collection of random variables, such that every finite collection of those random variables has a multivariate normal distribution i.e. every finite linear combination of them is normally distributed. They have advantage of representing the uncertainty in the prediction, taking both sensor noise and model uncertainty into account.

2.4.1 Detail

For the derivation we follow the function-space view as detailed in [9].

Let $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ be the set of training samples drawn from a noisy process.

$$y_i = f(x_i) + \epsilon \tag{2.8}$$

Where y_i is a target value such that $y_i \in \mathbb{R}$, x_i is an input sample from $\mathbb{R}^d$ and ϵ is a zero mean, additive Gaussian noise in $\mathbb{R}$ with variance σ_n^2.

A Gaussian process effectively estimates the posterior distributions over functions f from the training data D. These distributions are represented in terms of the training points. Gaussian processes assume that function values at different points are correlated. This means that the covariance between two function values $f(x_p)$ and $f(x_q)$, depends on the input values x_p and x_q. We represent this dependency with a covariance function(kernel) $k(x_p, x_q)$. There are various choices of kernel function with the most widely used being the homogeneous Gaussian kernel:

$$cov(f(x_p), f(x_q)) = k(x_p, x_q) = \sigma_f^2 exp(-\frac{1}{2l^2}|x_p - x_q|^2) \tag{2.9}$$

In the Gaussian kernel, σ_f^2 is the signal variance and l is the length scale. These parameters control the smoothness of the functions predicted by a Gaussian Process (GP). In the Gaussian kernel we can see the covariance between function values decreases as the distance between the input values decreases. The Gaussian kernel function above calculates the covariance function for direct function values x_i, but we only have noisy observations y_i. Due to this we need to add Gaussian noise. Note how σ_n^2 is the Gaussian noise from ϵ.

$$cov(y_p, y_q) = k(x_p, x_q) + \sigma_n^2 \delta_{pq} \tag{2.10}$$

δ_{pq} is one if $p = q$ and zero otherwise. Now, generalising for an entire vector of input values X and their corresponding observations values Y, the covariance becomes.

$$cov(Y) = K + \sigma_n^2 I \tag{2.11}$$

Where K is the $n \times n$ covariance matrix of the observed input values. This equation denotes a prior over all functions and for any X we can generate a corresponding matrix K such that we can sample a set of corresponding targets Y that have the desired covariance. The sampled values are then jointly Gaussian with $Y \sim \mathcal{N}(0, K + \sigma_n^2 I)$.

We want to be able to predict the function value y_* for an arbitrary point x_*, with relation to X, Y. From our kernel function, it follows the posterior over function values is a Gaussian with mean μ_{x_*} and variance $\sigma_{x_*}^2$.

$$\begin{aligned} p(y_*, x_*, X, Y) &= \mathcal{N}(f(x_*); \mu_{x_*}, \sigma_{x_*}^2) \\ \mu_{x_*} &= k_*^T (K + \sigma_n^2 I)^{-1} Y \\ \sigma_{x_*}^2 &= k(x_*, x_*) - k_*^T (K + \sigma_n^2 I)^{-1} k_* \end{aligned} \tag{2.12}$$

Here k_* is the $n \times 1$ vector of covariances between x_* and our n training points in X. We can now get our predictive distribution for a noisy observation y_* by adding the observation noise:

$$p(y_*, x_*, X, Y) = \mathcal{N}(y_*; \mu_{x_*}, \sigma_{x_*}^2 + \sigma_n^2) \tag{2.13}$$

2.4.2 Learning Kernel Hyperparameters

We want to be able to learn the kernel parameters $\theta = (\sigma_n^2, l, \sigma_f^2)$based on the training data X, Y. We estimate these parameters by maximising the log likelihood of the observations Y. The limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) algorithm is a popular algorithm for parameter estimation in machine learning [16]. The log likelihood of the observations is given by:

$$\log p(Y|X, \theta) = -\frac{1}{2} Y^T (K + \sigma_n^2 I)^{-1} Y - \frac{1}{2} \log |K + \sigma_n^2 I| - \frac{n}{2} \log 2\pi \tag{2.14}$$

This follows from the fact that the observations are jointly Gaussian. The log likelihood can be maximised using an optimiser such as LBFGS. The algorithm initially starts with an estimate of the optimal parameters value, then proceeds to iteratively refine the estimate with a series of better estimates. It does this by using the partial derivatives of the log likelihood to identify the direction of steepest descent and estimate the second derivative of the log likelihood.

2.4.3 Application to signal strength modelling

Gaussian Processes have been applied successfully in the past to signal strength modelling [9]. For each beacon we generate a predictive distribution or beacon survey map. We train the kernel parameters using our input values X correspond to locations, and the observations Y_b correspond to signal strength measurements obtained at these locations for a specific beacon b.

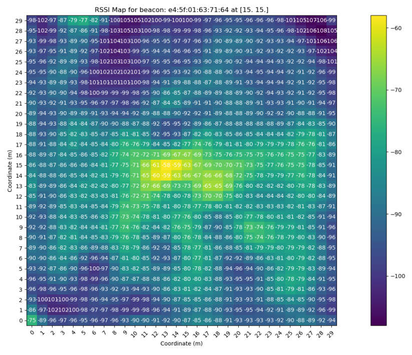

Figure 2.9: Example Gaussian process map from the 30x30m outdoors evaluation.

To position a device we divide the area of interest into a grid of cells. We then calculate the probability that a given fingerprint corresponds to each cell. To do this, we use the Euclidean distance between the currently processing RSSI fingerprint and the map cells estimated fingerprint. The map cells fingerprint is estimated using the Gaussian processes which form maps between positions and RSSI values.

$$distance(B, m, M, c) = \sqrt{\sum_{i=1}^{N} \frac{(m_{b_i} - M_{b_i}(c))^2}{N}} \tag{2.15}$$

The above equation calculates the estimated distance for a cell c, for the current fingerprint m containing RSSI values for beacons $B = b_1, ..., b_N$ and the set of Gaussian process RSSI maps M. Once the distance is obtained a probability can then be estimated for each cell.

$$p = \exp(-\frac{distance^2}{2\sigma^2}) \tag{2.16}$$

Where σ is the standard deviation associated with the fingerprint measurement noise. Note this is independent of any filtering.

Finally, we take the cell with the greatest probability as our position estimate.

2.5 Dataset

All training data for the algorithms was collected by hand at a variety of locations. The experimental setup is explained in detail in Chapter 4.

2.5.1 Metrics for Evaluation

Several metrics will be used in the evaluation of the requirements. Note that y_i is the prediction of location, x_i is the true value and n is the total number of estimates.

Root Mean Square Error

Root Mean Square Error (RMSE) is the square root of the average of the squared errors. This means that each error that is part of the RMSE is proportional to the size of the squared error, hence larger errors have a larger effect on the RMSE. This lack of proportionality to the error can make this metric hard to interpret. A smaller RMSE indicates that a model performs better.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2} \tag{2.17}$$

Mean Absolute Error

The Mean Absolute Error (MAE) is a measure of errors between paired observations expressing the same phenomenon. It is the arithmetic average of the absolute errors. This measurement uses the same scale as the measured data which makes it easy to interpret as any error can be thought of as a real world distance.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i| \tag{2.18}$$

Confidence Intervals

Confidence intervals are a range of values about the mean in which we are n% sure that the true population mean lies between these values. I will use a 95% confidence level due to it being the standard in most positioning papers such as [8].

Below is the formula for a 95% confidence interval; S_n is the standard deviation of the data and n is the number of samples.

$$CI_{95\%} = \pm 1.960\frac{S_n}{\sqrt{n}} \tag{2.19}$$

2.6 Requirements Analysis

After reading up on relevant concepts and models in the field of radio positioning, the criteria given in the project proposal (Appendix C) was revised and evaluated based on their importance to the project and difficulty to implement. These requirements can be seen in the table below in which the higher priority requirements are needed for a functional project and the medium priority requirements are needed for a successful project. The low priority requirements are not needed for the project to be a success, but are extensions to the core of the project. All criteria that have been met, in this document, have been clearly marked as completed.

Requirement	Priority	Difficulty
A mobile Bluetooth beacon should be built	High	Low
A beacon can communicate with the target device	High	Medium
The system should be able to estimate the position of the target device within a mapped area.	High	Low
Implement the Gaussian process model	High	Medium
Implement the path loss model	High	Medium
Implement the K-Nearest Neighbour model	High	Low
Implement the Proximity Model	High	Low
Implement filtering of the RSSI values	Medium	Medium
Gaussian Model should outperform the K-nearest Neighbour and Proximity Models in terms of mean absolute error for a beacon density of $1/30m^2$	Medium	Medium
Gaussian Model should have a lower mean absolute error less than 5-10 m of the actual device position at a beacon density of $1/30m^2$	Medium	Medium
Gaussian Model should have an average standard deviation of less than 5 m at a beacon density of $1/30m^2$	Medium	Medium
The system's accuracy should be evaluated at multiple beacon densities	Low	Medium
Implement the Weighted K-Nearest Neighbour model	Low	Medium
Implement a hybrid model using the Gaussian model combined with another localisation technique	Low	Medium
Ad hoc positioning e.g. place beacons as you go	Low	High

Table 2.1: Summary of the projects high-level success criteria

2.7 Tools Used

2.7.1 Hardware

For the project beacons needed to be constructed. Raspberry Pi 4Bs were used as they are popular single board computers that are low cost, support Linux and have a Bluetooth modem with support for Bluetooth 5.0 built in. The beacons were contained in boxes to protect them from the environment. A 10,000 Mah powerbank was used to allow the beacons to operate all day with needing to be recharged.

2.7.2 Software

A core part of the project was to allow the beacons to communicate with the target device. The Linux Bluetooth stack BlueZ and corresponding Python module bluepy were chosen due to there ability to run on both the Raspberry Pis and my personal Laptop.

Python is the language of choice due to it having an available Bluetooth library and its clear syntax. There are two main libraries which can implement Gaussian process models in Python: gpflow and scikit-learn. Scikit-learn was chosen due to its regarded as being easier to learn than gpflow and has more comprehensive documentation. The below table summarises the tools used throughout the project and their main purpose.

Libraries & Programs	Primary Usage
BlueZ	Linux Bluetooth stack
Bluepy	Interfacing with Bluetooth stack
NumPy	Efficient mathematical computation python library
Scikit-learn	Provides Gaussian Model
Matplotlib	Plotting graphs of results
Git	version control
GitHub	Backup the project

Table 2.2: Libraries and Programs used for project development

2.8 Software engineering techniques

During the project, an iterative development model was adopted. This comprised of development cycles of planning, design, implementation, testing and evaluation. This makes it easy to adjust the models and parameters after each iteration leading to a better final result, this is one of the most common agile techniques in industry [4].

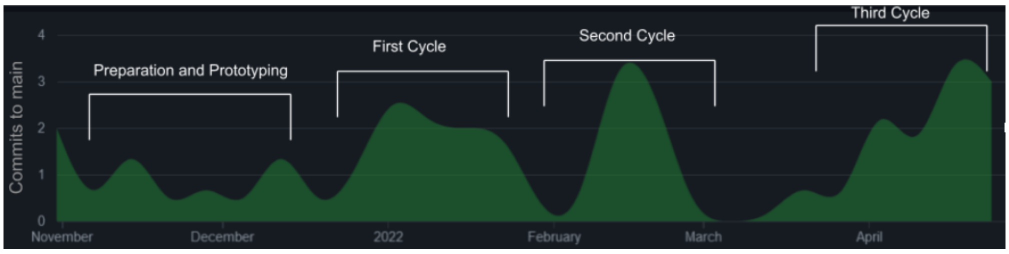

Figure 2.10: A summary of when commits were made to main. The development can be seen by peaks in the commits and the testing and/or evaluation took place in the troughs.

2.8.1 Development Environment

Most of the development for this project was done in the Visual Studio Code integrated development environment. This was mainly chosen due to its high adaptability through extensions and its support for Git version control. The equipment for this project was provided by Dr Keshav, which include five Raspberry Pi 4s, five 10,000MaH powerbanks and five 8 GB micro SD cards.

2.8.2 Backup Strategy

Several measures were taken to ensure all parts of the project were secure. Git commits was used for version control of the code and the project was pushed to GitHub for back up. A recent version of the repository was stored physically on two USB sticks one stored at the Cambridge Computer Lab and one stored in my personal residence. This follows the 3-2-1 backup rule which states that for your data to truly be protected you need three copies of the data, stored on two different types of media, with one backup copy being stored offsite. The report was backed up similarly in its own repository.

2.9 Personal Starting Point

I have studied general regression as part of my degree. I otherwise had no experience in regression models prior to starting the project. I have never studied any radio positioning before and my knowledge of radio waves did not go beyond high school level. With regard to localisation and filtering I attended the lectures for the "Introduction to Robotics" course at the University of Cambridge which contains general details on these.

To learn more, I read through lots of articles and papers, only a few of which are referenced in this document. Additionally, I learnt how to use BlueZ and bluepy, which comprise the Linux Bluetooth stack and Python package.

Chapter 3

Implementation

This chapter takes you through the development of the project to its current form. It details the hardware and its configuration, then how the various models to conduct indoor positioning were implemented It also gives an overview of the data-processing required. The next chapter will compare these models to each other.

3.1 Hardware Implementation

This section provides details on the hardware used in the project and how the beacons were programmed.

3.1.1 Hardware

To construct the positioning system, I constructed 5 beacons and a target device.

For the beacons themselves I was able to obtain 5 Raspberry Pi 4Bs each with a corresponding MicroSD card and 10,000Mah powerbank. Raspberry Pi 4Bs were chosen due to their customisability, affordability and ability to support Bluetooth 5.0.

For the target device a laptop running Ubuntu 20.04 was used. This laptop had a Bluetooth adapter which supported Bluetooth 5.1. However, any Linux based device able to run BlueZ and Python should be sufficient (e.g. another Raspberry Pi 4B).

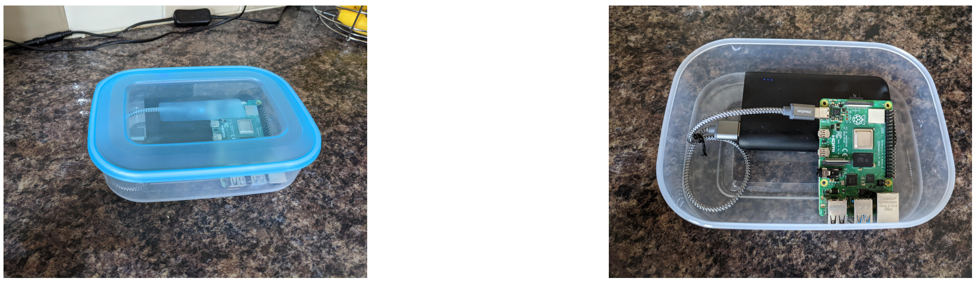
(a) The beacon boxed (b) The beacon unboxed

Figure 3.1: A look at a single beacon

Criterion met: *A mobile Bluetooth beacon should be built.*

3.1.2 Configuration for positioning

The beacons themselves had to be configured so that they broadcast the Bluetooth advertising packets correctly. To do this the ***advertising.bash*** script was executed on each beacon when it is turned on. This script executes some Bluetooth LE commands[3] which perform the following functions:

1. Set the advertising rate to less than 100ms.
2. Set the Bluetooth channel to be used at 38
3. Set the advertising mode to non-connectable

The beacon advertising period was set to be less than 100ms then a window duration of 1s was used to aim for at least 10 packets a window. This was because in the Harle, Faragher paper, at a rate of 10Hz, the error was negligible compared to a rate of 20Hz [8]. The channel was set to 38 to attempt to help mitigate the uneven channel gain and multipath interference caused by having all 3 channels enabled. The non-connectable mode was used as some systems can only report a single observation per window otherwise.

Criterion met: *A beacon can communicate with the target device.*

3.2 Positioning Models

This section describes the implementation of the positioning algorithm and the various models used to estimate the position.

3.2.1 General Positioning Algorithm

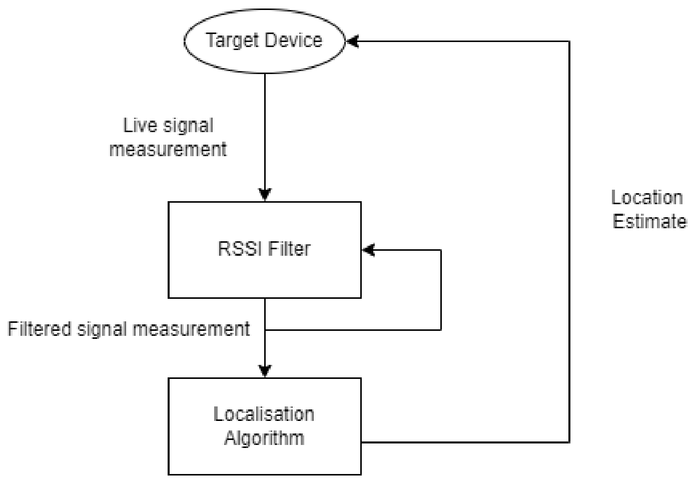

Figure 3.2: The general live positioning algorithm

3.2.2 Proximity

The proximity based algorithm is quite simple it selects the beacon with the strongest RSSI value from the current measurement and then co-locates the user with it. It is also equivalent to using KNN with k=1.

Criterion met: *Implement the Proximity Model.*

3.2.3 K-Nearest Neighbours

The k-Nearest Neighbours (KNN) algorithm uses k = 3 and takes the three strongest signals and their beacon positions, then uses weighted centroid localisation to estimate the target devices position. The Weighted Centroid Localisation (WCL) method estimates the target position by calculating the weighted centroid of the coordinates of the beacons. The weights are assigned such that they are inversely proportional to RSSI values. This method only requires the beacon's positions and no additional training. This is based on the KNN implementation in the Harle, Faragher Paper [8].

$$w_i = \frac{1}{r_i} \tag{3.1}$$

Criterion met: *Implement the K-Nearest Neighbour model.*

3.2.4 Path Loss

In the propagation or path loss model, we initially create a linear path loss model for each beacon. This model allows us to predict distance from a beacon to the target device (section 2.1.1). For a fingerprint the distance to each beacon is estimated. The target devices position is then estimated using the Min-Max method discussed in section 2.1.4 with the 3 smallest distances and their beacons corresponding location. This method estimates the target devices position by constructing bounding boxes around each selected beacon with the box's length being proportional to the estimated distance. The center of the intersection box is then the estimated target position.

For each beacon the initial point is chosen from the training data to be as close to 1 m as possible this is because 1 m is a commonly used initial value for d_0, but any point can be used.

$$d = d_0 10^{\frac{r_0 - r}{10\gamma}} \tag{3.2}$$

Criterion met: *Implement the path loss model.*

3.2.5 Gaussian Process

This uses the Gaussian process to produce the cell map then co-locates to the cell point with the highest probability which passes the variance/standard deviation threshold. I defined this threshold using the distribution of the cells standard deviations(std). If the cells personal std was more than the sum of standard deviation and median of all the cells then it would be discarded. This model can be used in two ways: in a stationary or tracking mode. In the stationary mode, the model has a uniform prior, meaning there are no limits on the location of the target device. While in the tracking mode a non-uniform prior is used, it is assumed that the target device is within 1 cell of the previous calculated position estimate.

Initialisation

In the initialisation portion of the algorithm. The Gaussian process regression models are fitted for each beacon, this uses the training data collected in the offline phase which maps points to RSSI values.

Cell Map

The cell map is a data structure which is used to estimate the position of the target device. It breaks down the area of interest into cells. These cells are uniform with the map providing the cell size (or length). A smaller cell size means the area has more cells and can lead to more accurate results, but this requires more computation (if we halve the cell size we quadruple the number of cells).

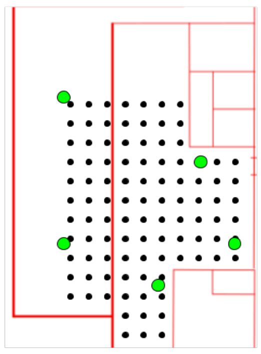

Figure 3.3: An example cell map in the University of Cambridge's Intel Lab. The Green dots represent beacons and the black dots represent the cell centers.

Calculating the cell probabilities

For each cell we calculate the probability that it contains the target device. To do this we calculate an estimated distance from the cell center to the target device. This is done by using the Gaussian processes to predict the RSSI value for each beacon at each cell center. This is then passed into the distance function to obtain an estimate. In the below equation B are the beacons, m is the current RSSI measurement and M is a function which returns the predicted RSSI for a beacon at the current cell center c.

$$distance(B, m, M, c) = \sqrt{\sum_{i=1}^{N} \frac{(m_{b_i} - M_{b_i}(c))^2}{N}} \tag{3.3}$$

The probability would then be calculated with the equation below. In the implementation the negative logarithm of p is used as it keeps the values more tractable and removes the requirement for the list of sorted probabilities to be reversed. The standard deviation used is the fingerprint measurement noise and is independent of filtering. I set this to 3.2 as this was the standard deviation of the RSSI values in the RSSI over time experiment (Appendix 2).

$$p = exp(-\frac{distance^2}{2\sigma^2}) \tag{3.4}$$

3.2.6 Algorithm

The algorithm is as follows:

1. Initialise the cell map and created Gaussian process regression models
2. Calculate cell probabilities
3. Discard cells which surpass the variance/standard deviation threshold.
4. Select most likely cell's center as the estimated position

Prior

In the Gaussian model we can add a prior to the probability, this may be local or uniform. A uniform prior indicates each iteration of the positioning is independent while a local prior assumes the next position is close to previous predicted position. This is implemented by only allowing the next estimate to be a neighbour or the same cell as the previous iteration, but this can vary based on implementation with some using models which attempt to estimate how the target device will move. Having a local prior may lead to better results when trying to converge on a single position.

Criterion met: *Implement the Gaussian process model.*

3.2.7 Gaussian Hybrid (Extension)

The original Gaussian process algorithm, uses the Gaussian process to produce a probability cell map then locates to the cell center point with the highest probability. This is similar to how the proximity algorithm determines the position, but instead of using the current measurement it uses the cell probabilities. In the Gaussian process algorithm a cell's probability gives an indicator of how close the algorithm believes the true value is to the cell center. The cell centers closest to the actual point would be expected to have the highest probability then. It follows that we may be able to position more accurately in a cell by using the position and probability information from multiple cells. This forms the basis behind the Hybrid Gaussian algorithms which locate using the 3 most probable centers and there respective probabilities. Both the KNN and Bounding Box Gaussian Models were developed on my own and my contribution from this project.

Criterion met: *Implement a hybrid model using the Gaussian model combined with another localisation technique.*

KNN Gaussian (Extension)

The KNN Gaussian method proceeds similarly to the regular Gaussian method, except the step where it locates to the nearest cell center (step 4). Instead, the KNN Gaussian model uses weighted centroid localisation technique discussed in section 2.1.3 to estimate the position. This method estimates the target position by calculating the weighted centroid of the coordinates of the nearest k cell centers. In the Gaussian method cell probabilities are proportional to the confidence that a cell center is near the target device, using this fact our we create our weights. (Note these weights become $w_i = 1/p_i$ when adjusted for the logarithm of the probability).

$$w_i = p_i \tag{3.5}$$

Bounding Box Gaussian (Extension)

The Min-Max/Bounding Gaussian method proceeds similarly to the regular Gaussian Method, except the final step where it locates to the nearest cell center (step 4). Instead, the Min-Max Gaussian method uses the Min-Max method discussed in section 2.1.4 to estimate the target devices position. This method estimates the target devices position by constructing bounding boxes around each cell center with box length proportional to the probability. The center of the intersection box is then the estimated target position. The values were transformed similarly to the weights in the Gaussian KNN algorithm with $w_i = p_i$ becoming $w_i = 1/p_i$ when adjusted for the logarithm of the probability.

3.2.8 Weighted KNN (Extension)

The Weighted KNN method is one of the most popular choices in the design of fingerprinting indoor positioning systems based on WiFi received signal strength (RSS) [7]. This method is a modified version of the k nearest neighbours algorithm. We utilise all the training data in this method. This method compares the input measurement to all training data points to produce a distance for each point. We then estimate the position by using weighted centroid localisation on the known points corresponding to the k smallest distance values. Distance was calculated using the Euclidean distance metric. This metric was used as it is a measure of the true distance between two points in euclidean space. Weights for the points can be formed from the reciprocal of the distance values.

$$w_i = \frac{1}{d_i} \quad d_i = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2} \tag{3.6}$$

Criterion met: *Implement the Weighted K-Nearest Neighbour model.*

3.3 Obtaining and processing the measurement data

The positioning algorithms require RSSI measurements to predict the position. These form the basis for the training data and live data. As discussed in section 2.3, advertisement packets are split into windows depending on the time they are received. The RSSI measurements in a window are then processed to produce the final measurement. In this processing we want to reduce the possibility of multipath interference. To counter this we can employ various strategies including taking the median of the window's measurements. The training data was constructed by taking the quantiles of a window of measurements. This enabled the data to better represent the variance in RSSI values.

3.3.1 Filtering

As discussed in section 2.1.4 filters can be used to reduce statistical noise and other inaccuracies in this case a filter is applied to a beacons RSSI values. Several filters were implemented including the simple moving average filter, simple median filter and Kalman filter. Different filters may be more appropriate in different situations. For example my implementation of the Kalman filter may be more suitable for stationary deployments as it assumes little movement this makes it unsuitable in a tracking environment where the RSSI value is changing fast. In this case the median filter may be a more appropriate option. Its evident though by Figure 3.3 that any filter provides less volatile measurements. In the evaluation a Kalman filter is used on the measurements this was due to the evaluation being focused around stationary points. For the filter the process noise was set to 0.008 [6] and the measurement noise was set to the variance of the RSSI measurements.

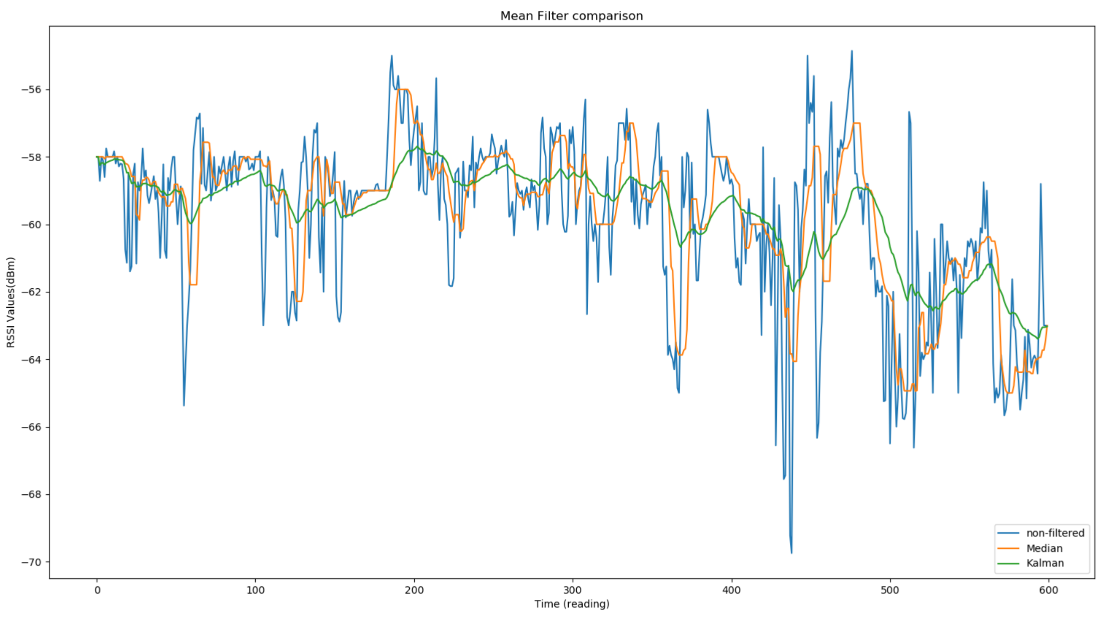

Figure 3.4: Filtering comparison for when applied at the window mean RSSI experiment. Note: The mean was emitted for brevity and performs similarly to the median filter.

Criterion met: *Implement filtering of the RSSI values.*

3.4 Navigating the Repository

As mentioned in section 2.9, I learned how to use bluepy through the documentation [2]. This provided several examples which I was able to adapt and build on top of. The scikit-learn module [18] provides the Gaussian process regressor and a method to optimise the hyperparameters, I learnt how to use the module from the documentation which contains examples [1].

There are many parts which make this project work. The code has been split into several folders based on their use. For example, the code for localisation models is inside the "Models" folder. On the top level of the directory, there are the files that are designed to be run directly from the command line. A detailed structure can be seen below:

```
/
├── data/ ........................................... Data obtained in the project
│   ├── training/ .................................. Files in the training dataset
│   ├── evaluation/ ............................... Files in the evaluation dataset
│   ├── experiment/ ......................... Data from various RSSI experiments
│   └── results/ .............................................. Example outputs
├── src/ ............................................... The Python source code
│   ├── Models/ ................................ Code for the localisation models
│   ├── Processing/ ............................ Code for filtering measurements
│   ├── Utils/ ............................ Code for utility methods such as file IO
│   ├── plotting.py ........................ Run-able file used for plotting graphs
│   ├── measurement.py .. Run-able file used for gathering/processing measurements
│   ├── evaluation.py ......... Run-able file used for evaluating localisation models
│   └── localisation.py ............ Run-able file used for running localisation live
├── advertising.bash .......................... Bluetooth script run on beacons
├── install.bash .................................. Install script for the project
└── requirements.txt ...................... List of Python project dependencies
```

Chapter 4

Evaluation

This chapter gives a detailed look at the results of the different positioning algorithms with respect to the evaluation metrics. It also contains a look at the effect of filtering and beacon density on accuracy.

4.1 Experimental Setup

I evaluated the project in two types of environment, outdoors and indoors. For the indoor experiment the Intel laboratory located in the Computer laboratory, University of Cambridge was chosen. This environment was chosen as it was a relatively open space which has lots of multipath interference from its electronic devices and architecture this should produce a similar environment to a rainforest. The outdoors experiments were conducted in open green spaces around the West Cambridge site.

To conduct a fair experiment several variables were controlled. All measurements were taken by hand at a height of 1 m above the ground. The position and rotation of beacons were kept constant during the tests to avoid unwanted effects. Beacons were also always placed symmetrically around the mapped area on the edges. All 5 beacons were used in all the environments. All the random variables were seeded to ensure that that results are reproducible. This was used in the Gaussian Process Model and Kalman Filters.

4.1.1 Obtaining the Ground Truth

For the indoor test in the intel lab the ground truth was obtained from the carpet grid which covers the floor. Each tile was 0.6 x 0.6 m which could then be used to calculate the actual ground truth.

For the outdoors tests no grid was available, so I had to construct one to do this I used a measuring wheel. The wheel I used was precise to the nearest centimetre. To construct the grid markers were placed on the ground with there position from the origin measured 3 times to ensure accuracy.

(a) Inside: Intel Lab has carpet tiles

(b) Outside:The measuring wheel

Figure 4.1: Ground Truth sources

4.1.2 Training Data

The training data was obtained by taking a single measurement window (all the RSSI values received in 1 second) at a uniform grid in the mapped space. The key values for each environment are detailed below. The grid cell length is the distance between measurement points in a grid. The Measurement density is the average number of training measurements taken per metre squared, and the beacon density is the number of beacons per metre squared.

Environment	Grid Cell Length (m)	Measurement Density (m^{-2})	Beacon Density (m^2)
Indoor(17.2x13.2m)	1.2	0.39	1/45
Outdoor(15x10m)	2	0.31	1/30
Outdoor(30x30m)	2.5	0.15	1/180

Table 4.1: Environment Measurement Distances

4.1.3 Evaluation Data

For the evaluation data random points within the mapped area were generated. For each point 5 RSSI window readings were taken to establish an average accuracy for the point.

4.2 Measuring the Criteria

As discussed in section 2.5.1 two different metric of evaluation will be used: root mean square error (RMSE) and mean absolute error (MAE). These metrics each have a 95% confidence intervals on all their values. For each environment the metric was run on the results of the localisation algorithm for each model. The standard deviation was also computed on the error results. These results can be used to evaluate their respective success criteria quantitatively. While the beacon density evaluation is more qualitative, I will pull observations from the different metrics to draw conclusions.

4.3 Analysis of Results

This section discusses the results of the models for the various environments and contains reference to various tables and graphs in this section. If a graph exists without a corresponding table in this chapter the table will be available in appendix A.

In general the results show that all the Gaussian methods outperform the KNN, Proximity and Propagation by a clear margin. Additionally, filtering and adjusting the Gaussian models parameters can lead to better performance.

Criterion met: *The system should be able to estimate the position of the target device within a mapped area.*

4.3.1 General Model Comparison

This section is a discussion of the general system performance in relation to the two environments and the evaluation metrics. These evaluations are all done with a Gaussian cell size of 0.25 m and a uniform prior.

Indoors

In this environment the Gaussian Models outperform the other tested models, the hybrid Gaussian models also provide an increase in performance (around 0.4 m) over the Gaussian Model though this performance increase is within margin of error. Surprisingly, the WKNN model performs as well or worse than the baseline localisation algorithms, we would expect it to perform closer to the Gaussian algorithms due to its use of additional data. The standard deviations are all in a similar range apart from the WKNN value of 3.18 this value is large given MAE value of 4.25 m ± 0.44 this suggests there was a lot of variance in the accuracy of the WKNN model which may explain it's lower than expected accuracy. These results are demonstrated in the table 4.2 and figure 4.2.

Model	MAE (m)	S_n (m)	RMSE (m)
KNN	4.57 ± 0.36	2.56	5.24 ± 4.37
Proximity	4.32 ± 0.28	2.03	4.78 ± 3.27
Propagation	4.15 ± 0.31	2.26	4.72 ± 3.31
Gaussian	3.72 ± 0.35	2.5	4.48 ± 3.84
Gaussian KNN	3.31 ± 0.30	2.11	3.95 ± 2.55
Gaussian MinMax	3.36 ± 0.29	2.15	3.97 ± 2.55
WKNN	4.25 ± 0.44	3.18	5.31 ± 5.42

Table 4.2: Table of results for Inside Approximately $1/40m^2$ density. cell size = 0.25 and a uniform prior

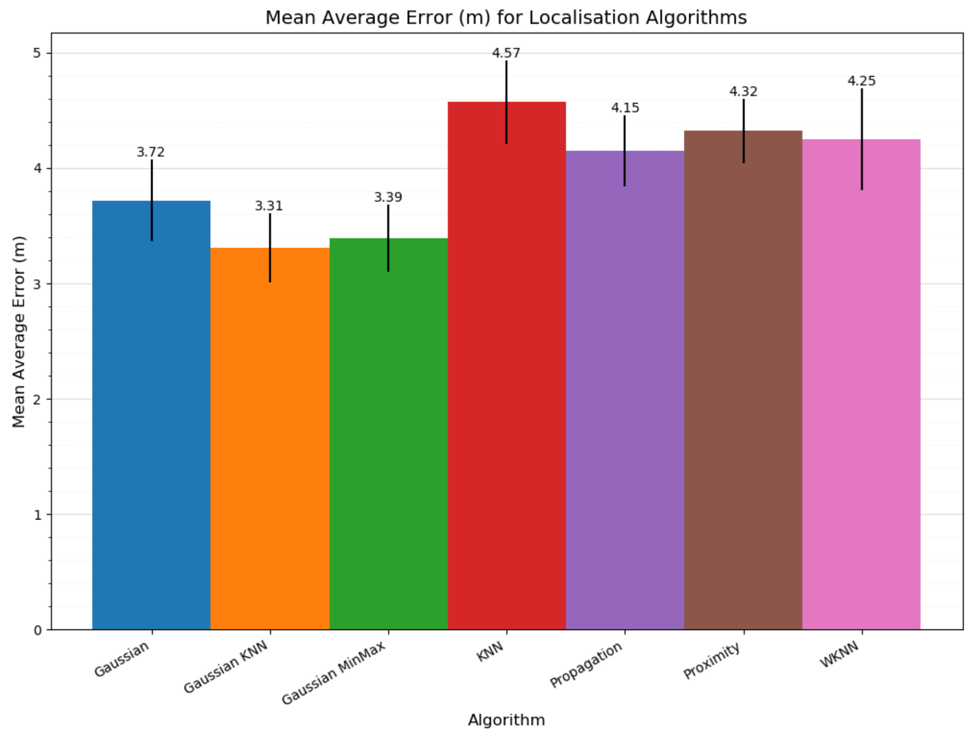

Figure 4.2: Plot of results for Inside Approximately $1/40m^2$ density. With a cell size = 0.25 and a uniform prior

Outdoors

To measure the general use outdoor performance the 15x10m grid was used. In this environment we can see that the Gaussian based and WKNN models outperform the other models significantly. This is expected, due to the additional training data they use in their predictions. In my results the Gaussian hybrid models perform best with the Gaussian KNN variant having a slightly better MAE and RMSE values though taking into account margin for error they all perform similarly. If we assume the error is normally distributed then the Gaussian model has accuracy of at least 5.42 m 95% of the time.

When comparing the outdoors environment to indoors, a smaller 15x10m grid was used this was to fulfil the success criteria, due to the higher beacon density the measurement grid size was increased from 1.2 m indoors to 2 m outdoors to make the two environments roughly comparable though this is only the case for algorithms which use training data. In the case of the algorithms without training data we would expect this to lead to better performance outdoors due to the higher beacon density.

When comparing outdoor and indoor results we would expect the outdoor results to perform better due to less electromagnetic interference. A surprising result was the propagation model having worse performance outdoors; you would expect that outdoors there would be less interference and hence follow the model more accurately. However, this does not seem to be the case. In general when looking at the Gaussian and WKNN models they perform better than indoors as expected. These results are demonstrated in the table 4.3 and figure 4.3.

g

Model	MAE (m)	S_n (m)	RMSE (m)
KNN	4.19 ± 0.32	2.02	4.65 ± 2.81
Proximity	3.78 ± 0.23	2.10	4.05 ± 2.10
Propagation	5.33 ± 0.29	1.82	5.63 ± 3.17
Gaussian	2.24 ± 0.25	1.59	2.74 ± 2.51
Gaussian KNN	2.17 ± 0.26	1.62	2.71 ± 2.51
Gaussian MinMax	2.19 ± 0.26	1.62	2.72 ± 2.53
WKNN	2.32 ± 0.30	1.88	2.99 ± 2.86

Table 4.3: Table of results for Outside $1/30m^2$ density. With a cell size = 0.25 and a uniform prior

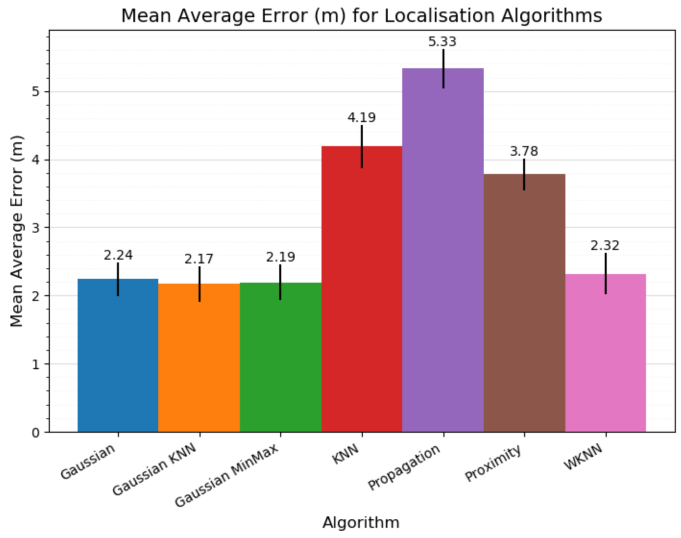

Figure 4.3: Plot of results for Outside $1/30m^2$ density. With a cell size = 0.25 and a uniform prior

Criterion met: *Gaussian Model should outperform the K-nearest Neighbour and Proximity Models in terms of mean absolute error for a beacon density of* $1/30m^2$.

Criterion met: *Gaussian Model should have a lower mean absolute error less than 5-10 m of the actual device position at a beacon density of* $1/30m^2$.

Criterion met: *Gaussian Model should have an average standard deviation of less than 5 m at a beacon density of* $1/30m^2$

4.3.2 Gaussian Parameters

The Gaussian model has a couple of parameters which can affect the performance; the cell size and prior distribution.

Effect of Cell Size on Gaussian Model

As the cell size decreases the number of total cells increases leading to a finer grid of points which the Gaussian process is evaluated at. Due to this we would expect a more accurate system up to an optimal point, but this comes at the cost of performance. Figure 4.4 indicates a correlation between cell size and accuracy, but it also includes a lot of variance in its results indicating that the cell size may need to be tweaked for individual environments to provide the best accuracy to performance ratio.

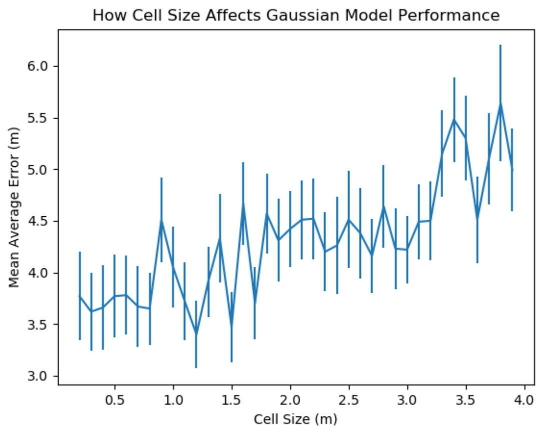

Figure 4.4: MAE for Gaussian Process Model for varying cell size on the indoor environment. (Table: Appendix A)

Local vs Uniform Prior

The choice of prior is dictated by the type of tracking. By assuming a local prior we envision that the target device will stay close to its current position. In my results this can provide improvements in results such as in the indoor test (figure 4.5), but it can also have no or even a detrimental effect as shown Gaussian results and in the outdoor results(Appendix A). Introducing a more complex prior than the nearest neighbour cell may lead to better results this could include creating a more advanced motion model.

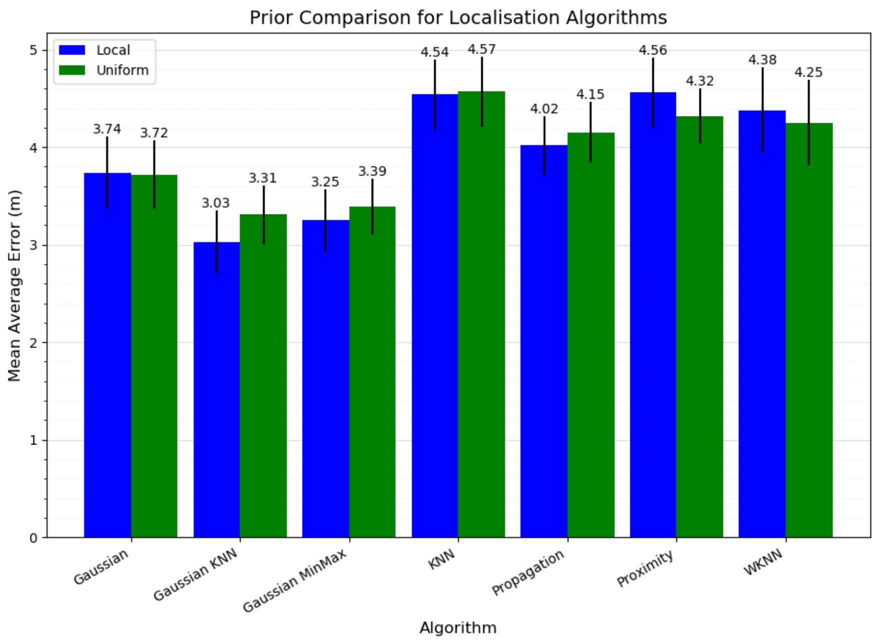

Figure 4.5: MAE prior comparison for the Indoor environment. (Table: Appendix A)

4.3.3 Impact of Filtering

In general filtering leads to an improvement in the accuracy of the models as expected, though it can also have a detrimental effect in some cases this may be caused by the filtered signal being sluggish and not adapting fast enough to a change in the RSSI value.

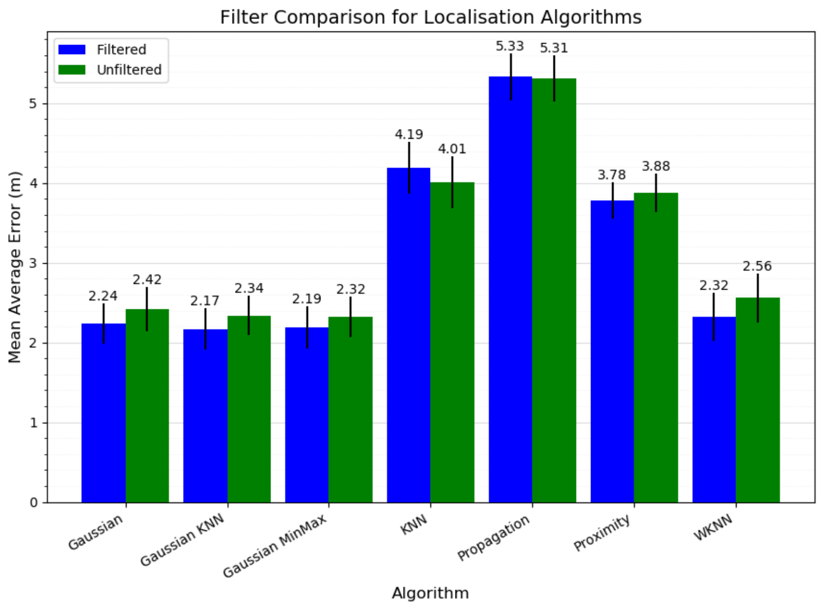

Figure 4.6: MAE filter results comparison for the 15x10m outdoor environment. (Table: Appendix A)

4.3.4 How density affects performance (Extension)

As the density of beacons decreases we have less information in each fingerprint, this leads to a reduced accuracy in the localisation system. The system was evaluated outside at 2 locations, 10x15m and 30x30m the aim of this was to see how decreasing the density affects the performance of the localisation. For each of the locations four beacons were placed evenly around the location with a beacon at the center. This center beacon could be removed from the training data to produce localisation results with a lower density of results. It should be noted that the two outdoor environments had different measurement distances (Table 4.1) of 2 m and 2.5 m. While this makes the environments not empirically comparable it was done due to the practicalities of surveying such a large space.

Density Comparison				
	15x10m Outdoors		30x30m Outdoors	
Model	$1/30m^2$	$1/37.5m^2$	$1/180m^2$	$1/225m^2$
Gaussian	2.24 m ± 0.25	3.51 m ± 0.37	8.34 m ± 0.85	8.74 m ± 0.89
Gaussian KNN	2.17 m ± 0.26	3.17 m ± 0.31	6.97 m ± 0.75	7.22 m ± 0.64
Gaussian MinMax	2.19 m ± 0.26	3.10 m ± 0.30	6.81 m ± 0.70	7.38 m ± 0.63
KNN	4.19 m ± 0.32	3.67 m ± 0.34	7.03 m ± 0.48	7.46 m ± 0.57
Propagation	5.33 m ± 0.29	5.39 m ± 0.32	8.73 m ± 0.52	9.26 m ± 0.61
Proximity	3.78 m ± 0.23	5.02 m ± 0.40	8.28 m ± 0.51	9.12 m ± 0.56
WKNN	2.32 m ± 0.30	2.45 m ± 0.26	18.99 m ± 1.40	20.95 m ± 1.34

Table 4.4: Table of MAE results for different outdoor densities

It can be seen in table 4.4 that as expected in general a higher beacon density leads to a lower accuracy. When comparing different densities within the same environment the results always decrease in accuracy except for KNN which increases in 15mx10m, but this may just be in margin of error. When drawing observations between the different environments its prevalent that accuracy does decrease significantly, with the KNN and hybrid Gaussian models performing the best. Notably is the performance of WKNN at the lower densities which has double the error of the other algorithms this is likely caused by the additional training data possibly misleading the model. For the 30x30m environment the best algorithms only perform as well as KNN (within margin of error), this could suggest that there is a density were a fingerprint doesn't have enough information for the additional training data to lead to a lower error. This additional data may even be detrimental to accuracy, in the case of WKNN the MAE is more than twice than the KNN MAE. To conclude as expected increasing the density of beacons increases performance, though more work needs to be done to investigate how to place beacons optimally within a space.

(a) Outdoors 15mx10m

(b) Outdoors 30x30m

Figure 4.7: Outdoor locations

Criterion met: *The system's accuracy should be evaluated at multiple beacon densities.*

4.3.5 Improving Performance

To improve performance enabling a local prior can helps if the mobile device is in a constant position. Also, implementing a particle filter with a more advanced motion model for the position may also lead to better results. Increasing the density of beacons also leads to an increased performance, though these must be placed in area with low coverage to have the best effects.

Chapter 5

Conclusions

This chapter summarises the project with a discussion of what went well, what lessons I have learnt and a few thoughts on applications of this project and potential ways to improve upon it.

Working on this project has been an enjoyable experience and has given me the opportunity to learn about localisation techniques and radio signals. I have particularly enjoyed working with the Raspberry Pis with this being the first time I have built a system relying on communication from other devices.

5.1 Project Achievements

As set out in previous chapters, this project has been a success in that it has met all of the high and medium priority criteria. It has also met the extension criteria for implementing the additional models and evaluation of the system at multiple densities.

My first step was to build and program the Bluetooth beacons to send out the advertising packets. This proved useful as once the prototype was built I was then able to experiment how RSSI was affected by various factors. Next I developed the localisation system itself, the main model is the Gaussian process model, this produces better results than the baseline models both indoors and outdoors. These results are a promising indicator that the model may be able to be applied to mapping in the rainforest.

I also designed and implemented the hybrid Gaussian models as an extension. These models produced more accurate results than the regular Gaussian model in general and are my contribution from this project. The WKNN model was also implemented as an extension along which provided as good or inferior results to the Gaussian models. Also, an investigation into how beacon density affected the accuracy of the system was conducted.

5.2 Lessons Learnt

I have enjoyed experimenting to see the effects of different objects on RSSI value, this was quite illuminating in learning how we would construct a positioning system which took advantage of these facts. Another important lesson I have learnt is how Bluetooth Low

Energy works and how to use it. This had helped my general understanding of wireless communication and I have no doubt the knowledge will be useful in the future.

There were also a few oversights made in the development of the project. In particular, estimating the time to collect measurement and evaluation data. This had to be done by hand and proved to be a colossal task on my own with the 30x30m area alone requiring over 144 training points. In future, I would enlist the help of someone to reduce the workload.

I was unable to complete the final extension success criteria. This involved creating ad hoc positioning this was not completed due to time constraints and the complexity of the extension.

5.3 Future Development

Deploying the project in a real-world environment faces several challenges the main being scalability. To achieve the beacon density required for accurate localisation its likely lots of beacons would be needed. A hybrid system with some areas of high and low beacon densities may give the advantage of a larger scale without the entire area needing to be mapped.

Despite this the project has been a success and met all of the original success criteria, there are still some ways I can see that the system being improved in the future. I have summarised a few possible improvements below:

1. See how the models performs in an actual rainforest. This was not possible due to the limitations on where the system could be tested, but this could provide good insights into how the model could be improved and perform.

2. Ad hoc Positioning: Placing the beacons down as you go ”like breadcrumbs”. This is effectively Simultaneous localisation and mapping (SLAM) using radio signal strength, this is a difficult problem outdoors due to the lack of landmarks which are present indoors (i.e. walls). Without GPS it is likely a system would also likely suffer from divergence over time.

3. Consider using UWB beacons for a localisation system. The cost of UWB is likely to decrease in time which may make it a more suitable candidate to build a system with due to its increased accuracy.

4. Using a DNN for to produce the RSSI map. This may produce better results than the Gaussian models. Problems could arise from gathering enough training data for this approach though.

5. Automatic mapping: given the area have a robot map the area.

Bibliography

[1] 1.7. Gaussian Processes. https://scikit-learn.org/stable/modules/gaussian_process.html.

[2] bluepy - a Bluetooth LE interface for Python — bluepy 0.9.11 documentation. https://ianharvey.github.io/bluepy-doc/.

[3] Core Specification 5.0 – Bluetooth® Technology Website.

[4] Is Iterative Development the New Black of Software Development? https://www.uptech.team/blog/iterative-development.

[5] Paramvir Bahl and Venkata Padmanabhan. Radar: An in-building rf-based user location and tracking system. *Proceedings - IEEE INFOCOM*, 2, 01 2000.

[6] Wouter Bulten, Anne C. Van Rossum, and Willem F. G. Haselager. Human slam, indoor localisation of devices and users. In *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 211–222, 2016.

[7] Giuseppe Caso, Luca de Nardis, and Maria-Gabriella di Benedetto. A Mixed Approach to Similarity Metric Selection in Affinity Propagation-Based WiFi Fingerprinting Indoor Positioning. *Sensors (Basel, Switzerland)*, 15(11):27692–27720, October 2015.

[8] Ramsey Faragher and Robert Harle. Location Fingerprinting With Bluetooth Low Energy Beacons. *IEEE Journal on Selected Areas in Communications*, 33(11):2418–2428, November 2015.

[9] Brian Ferris, Dirk Hähnel, and Dieter Fox. Gaussian processes for signal strength-based location estimation. In *Robotics: Science and Systems*, 2006.

[10] Jeffrey Humpherys, Preston Redd, and Jeremy West. A fresh look at the kalman filter. *SIAM Review*, 54(4):801–823, 2012.

[11] Michael Jenkins. An Examination of Ultra-Wideband (UWB) For Positioning & Location Tracking | Blog | Link Labs. https://www.link-labs.com/blog/ultra-wideband-positioning-location-tracking.

[12] juliaaltenbuchner. GPS signal in the rainforest? https://uclexcites.blog/2013/09/04/gps-signal-in-the-rainforest/, September 2013.

[13] Moses A. Koledoye, Daniele De Martini, Simone Rigoni, and Tullio Facchinetti. A comparison of rssi filtering techniques for range-based localization. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 761–767, 2018.

[14] Olivia Lai. Amazon rainforest is now a source of co2 instead of absorbing it. `https://earth.org/amazon-rainforest-now-a-co2-source/`.

[15] Rui Ma, Qiang Guo, Changzhen Hu, and Jingfeng Xue. An improved wifi indoor positioning algorithm by weighted fusion. *Sensors*, 15(9):21824–21843, 2015.

[16] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, page 1–7, USA, 2002. Association for Computational Linguistics.

[17] Guoqiang Mao, Barış Fidan, and Brian D.O. Anderson. Wireless sensor network localization techniques. *Computer Networks*, 51(10):2529–2553, 2007.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] Xuesheng Peng, Ruizhi Chen, Kegen Yu, Feng Ye, and Weixing Xue. An improved weighted k-nearest neighbor algorithm for indoor localization. *Electronics*, 9:2117, 12 2020.

[20] Tara Slough, Jacob Kopas, and Johannes Urpelainen. Satellite-based deforestation alerts with training and incentives for patrolling facilitate community monitoring in the peruvian amazon. *Proceedings of the National Academy of Sciences*, 118(29):e2015171118, 2021.

[21] Janja Svecko, Marko Malajner, and Dušan Gleich. Distance estimation using rssi and particle filter. *ISA Transactions*, 55, 11 2014.

[22] Thradon Wattananavin, Kiattisak Sengchuai, Nattha Jindapetch, and Apidet Booranawong. A Comparative Study of RSSI-Based Localization Methods: RSSI Variation Caused by Human Presence and Movement. *Sensing and Imaging*, 21(1):31, July 2020.

Appendix A

Tables

Filtering

All results were collected with uniform prior and a cell size of 0.25.

Model	Filtered (m)	Unfiltered (m)
Gaussian	3.72 ± 0.35	4.09 ± 0.4
Gaussian KNN	3.33 ± 0.29	3.41 ± 0.32
Gaussian MinMax	3.32 ± 0.3	4.02 ± 0.39
KNN	4.57 ± 0.36	4.55 ± 0.36
Propagation	4.15 ± 0.31	4.08 ± 0.28
Proximity	4.32 ± 0.28	4.71 ± 0.36
WKNN	4.25 ± 0.44	4.42 ± 0.42

Table A.1: MAE values Indoor Filtering table

Model	Filtered (m)	Unfiltered (m)
Gaussian	2.24 ± 0.25	2.42 ± 0.28
Gaussian KNN	2.17 ± 0.26	2.34 ± 0.25
Gaussian MinMax	2.19 ± 0.26	2.33 ± 0.25
KNN	4.19 ± 0.32	4.01 ± 0.33
Propagation	5.33 ± 0.29	5.31 ± 0.29
Proximity	3.78 ± 0.23	3.88 ± 0.24
WKNN	2.32 ± 0.3	2.56 ± 0.31

Table A.2: Outdoor 10x15 Filtering table

Prior

All results were collected with filtering enabled and a cell size of 0.25.

Model	Uniform (m)	Local (m)
Gaussian	3.72 ± 0.35	3.74 ± 0.37
Gaussian KNN	3.31 ± 0.29	3.03 ± 0.32
Gaussian MinMax	3.32 ± 0.3	3.25 ± 0.32
KNN	4.57 ± 0.36	4.54 ± 0.36
Propagation	4.15 ± 0.31	4.02 ± 0.3
Proximity	4.32 ± 0.28	4.56 ± 0.36
WKNN	4.25 ± 0.44	4.38 ± 0.44

Table A.3: Indoor Prior table

Model	Uniform (m)	Local (m)
Gaussian	2.24 ± 0.25	2.31 ± 0.28
Gaussian KNN	2.17 ± 0.26	2.19 ± 0.26
Gaussian MinMax	2.19 ± 0.26	2.22 ± 0.26
KNN	4.19 ± 0.32	4.19 ± 0.34
Propagation	5.33 ± 0.29	5.33 ± 0.29
Proximity	3.78 ± 0.23	3.86 ± 0.24
WKNN	2.32 ± 0.3	2.39 ± 0.29

Table A.4: Outdoor 10x15 m Prior table

Cell Size

Cell Size	MAE (m)
0.2	3.77 ± 0.43
0.3	3.62 ± 0.38
0.4	3.66 ± 0.41
0.5	3.77 ± 0.4
0.6	3.78 ± 0.38
0.7	3.67 ± 0.39
0.8	3.65 ± 0.35
0.9	4.51 ± 0.41
1.0	4.05 ± 0.39
1.1	3.72 ± 0.38
1.2	3.4 ± 0.33
1.3	3.91 ± 0.34
1.4	4.33 ± 0.43
1.5	3.47 ± 0.34
1.6	4.67 ± 0.4
1.7	3.7 ± 0.35
1.8	4.57 ± 0.39
1.9	4.31 ± 0.4
2.0	4.42 ± 0.37
2.1	4.51 ± 0.38
2.2	4.52 ± 0.39
2.3	4.2 ± 0.38
2.4	4.26 ± 0.47
2.5	4.51 ± 0.47
2.6	4.38 ± 0.44
2.7	4.16 ± 0.36
2.8	4.64 ± 0.4
2.9	4.23 ± 0.39
3.0	4.22 ± 0.33
3.1	4.49 ± 0.36
3.2	4.5 ± 0.38
3.3	5.15 ± 0.42
3.4	5.48 ± 0.41
3.5	5.3 ± 0.41
3.6	4.51 ± 0.42
3.7	5.1 ± 0.44
3.8	5.64 ± 0.56
3.9	4.99 ± 0.4

Table A.5: Indoor Cell Size Table

Appendix B

Experiments

A couple of experiments were carried out to investigate how RSSI values changed under certain conditions.

RSSI over time

In this experiment how RSSI changed over time was investigated to conduct this, a beacon and target device were kept at a constant distance of 1 m from each other and the received RSSI values from the beacon recorded for 1000 seconds. The raw data can be found in the `results/test_measurement.csv` file in the repo.

Metric	Value
Mean (dBm)	-60
Median (dBm)	-59
S_n	3.2

Table B.1: Over time results data

Rotation

In this experiment how rotation effected RSSI was investigated. To conduct this, a beacon and target device were kept at a constant distance of 1 m from each other and the received RSSI values from the beacon recorded for 1000 seconds. With the only changing factor being the rotation of the beacon. The raw data can be found in several files being located at `results/test_rotation_DEGREE_measurement.csv` in the repo.

Appendix C

Project Proposal

Project Proposal - Geolocation in a Rainforest

Candidate Number : 2392F

October 2021

Introdution and Description

In rainforests, GPS signals only have a typical accuracy of around 10-30m due to harsh environmental conditions such as the thick forest canopy. This accuracy is insufficient for fieldwork, so this project proposes implementing an indoor wayfinding technique in an outdoor context to achieve better accuracy.

This radio-based positioning technique involves setting up an area of radio beacons, then using their Received Signal Strength Indicator (RSSI) to determine the position of a device within the range. The position will be calculated by using a two-phase radio frequency based fingerprinting method. In the first phase, we train the system offline at set reference points. We then use this data in the second phase to determine the position of the device.

Bluetooth Low Energy (BLE) will be used due to it being much more accessible than alternatives such as Ultra-wideband (UWB). This is due to BLE being available in devices such as old smartphones and affordable microcontrollers (i.e. raspberry pi).

The construction of the Bluetooth beacons has been considered and will consist of a microcontroller with Bluetooth low energy combined with a portable power device. The microcontroller will likely be a Raspberry Pi. Raspberry Pis are affordable microcontrollers with inbuilt Bluetooth modules in all mainline models released after the 3b. The rainforest will lead to the microcontroller being exposed to harsh environmental conditions. Due to this the microcontroller will need to be stored in a decently sealed box to help protect against the humidity of rainforests.

To evaluate the effectiveness of the system, an experiment will be carried out where the accuracy of the system is evaluated. The evaluation will consider the environmental conditions the system is exposed to, such as humidity, to see how this affects the accuracy of the system.

Starting Point Statement

A paper exists which describes using Bluetooth low energy for indoor positioning with Bluetooth based devices [1].

In addition, in 2020 a UROP project was conducted this summer evaluating the feasibility for different low-cost outdoor location determination methods for rainforests; this study contains a brief look at general radio-based solutions but doesn't investigate Bluetooth-based beacon solutions.

Description of the substance and structure of the project

Major Work Items

- Beacon Construction
- Training Algorithm

- Positioning Algorithm

The training algorithm depends on the beacons existence and the positioning algorithm depends on the training algorithms produced data, but they can be developed independently.

Project Success Criteria

- A beacon is constructed which uses Bluetooth to communicate with a Bluetooth-enabled device.
- The system should determine the position of the device.
- The system should be accurate within 5-10m of the actual device position.
- The system should have a standard deviation of less than 10m.

Project Plan

The project will be completed using an agile methodology employing scrums with two-week sprints to complete the project.This will allow an easier evaluation of current progress and help focus areas to divert attention too.

Timeline

1. Preparatory Work: Research into relevant papers 18/10/21 - 25/10/21
2. Preparatory Work: Build Beacon 25/10/21 - 08/11/21

   Milestone: Create Prototype Beacon
3. Practical Work: Writing code to connect a beacon and the device and obtain RSSI value 08/11/21 - 06/12/21

   Milestone: Connect a beacon to the device
5. Practical Work: Connecting multiple beacons to the device at once to obtain their RSSI values 06/12/21 – 20/12/21

   Milestone: Connect multiple beacons to the device
6. Practical Work and Complete Progress Report: Implement the training phase 20/12/21 - 03/01/22
7. Practical Work: Implement the location determination algorithm on the device 03/01/22 – 31/01/22
9. Practical Work: Physical evaluation of system 31/01/22 – 28/02/22 Milestones:
   - Train the system for a set layout indoors

- Determine the location of a device indoors
- Train the system for a set layout outdoors
- Determine the location of a device outdoors

11. Writing Dissertation: Writeup of development 28/02/22 – 14/03/22

12. Writing Dissertation: Writeup of algorithms 14/03/22 – 28/03/22

13. Writing Dissertation: Evaluation Writeup 28/03/22 – 11/04/22

Risk Management

All the project code will be stored in a git repository which will be regularly committed to. This will ensure that the code is backed up sufficiently. The code should be able to be developed on an MCS machine in the case of a personal machine failure. In the case of a beacon failure, the tests will have to be evaluated on a smaller space if an alternative cannot be sourced.

Resource Declaration

This project will require the construction of Bluetooth beacons; these will likely be Raspberry Pis with a Bluetooth module (3b, 3b+, 4 or zero w) this is due to their affordability and low power usage. These will be combined with portable power sources to create the beacons. A device to locate will also be required which will be a portable device with Bluetooth (portable computer or phone). These will be provided by Dr S Keshav.

In addition to this, I will be developing the project on my personal laptop. I accept full responsibility for this machine, and I have made contingency plans to protect myself against hardware and/or software failure.

Bibliography

[1] Zixiang Ma, Stefan Poslad, John Bigham, Xiaoshuai Zhang and Liang Men: A BLE RSSI Ranking based Indoor Positioning System for Generic Smartphones