# Asset location using low-cost beacons, smart roaming devices and cloud computing

Progress Report

Samuel Nixon | 15444418

B.Eng. Electronic & Computer Engineering

Academic Supervisor: Dr. Edward Jones

## Abstract

The aim of this project is to design and test a fully scalable, low cost system to provide location tracking of assets, using a combination of intelligent hardware, simple hardware and cloud services.

This combination of different 'levels' of hardware allows a flexible on-the-ground solution that can be adjusted to suit real-life demands, with the ratio of smart to simple devices providing the accuracy desired for each application. Cloud Services used are provided by Amazon Web Services, a popular cloud services provider. In order to provide seamless on-demand scaling, the architecture of the cloud services is of particular interest. Such a system would find many applications in industry, particularly in transport and manufacturing.

In modern times, cloud computing allows organisations to quickly and effectively build IT infrastructure and services. Utilising cloud services to provide a service that can automatically scale to handle any demands seemed interesting. Access to intelligent hardware with current connectivity capabilities has never been easier, potentially providing a means for a low-cost solution to tracking of assets in a domain. This project seeks to explore the effectiveness of combining some of these intelligent devices with simpler, lower cost beacons to provide location information for objects, without the need to attach expensive data and GPS capable modules to every asset.

This report outlines the progress of the project thus far, discusses findings and explores future work.

Technologies of note are WiFi, Bluetooth Low Energy, GPS, Cloud Services, AWS, Raspberry Pi, Python, Node.js

## Table of Contents

# 1 Introduction

## 1.1 Background

Asset location has always been useful data to have, but companies generally have to rely on vague data such as last known city/distribution center and next city/distribution center. Fine grained data such as movement of an asset within a yard or site is normally unavailable. Solutions providing location data of Tractor/Trailer assets are common, but are tailored to long haul distances, rather than the real-time tracking of asset movement in a yard. With the increased ubiquity of hardware providing GPS monitoring, tracking of an expensive object such as a forklift within a yard has become possible, but what if the location of the loads the forklift was moving in the yard was also desired? Attaching GPS and network capable hardware to load beds would be financially excessive, particularly compared to a solution where a forklift was able to report nearby assets, with location sensitivity.

Transportation companies are increasingly turning to technology to assist their business. Some countries have passed legislation forcing the use of technology to provide overview and compliance with existing legislation, such as driving working hours. In the US, use of an ELD (Electronic Logging Device) by professional drivers will be mandatory after 16 December 2019 [1]. These devices must GPS capabilities, and almost always provide internet access and Bluetooth. With hardware such as this already a requirement, there is opportunity to utilize this pre-existing hardware to provide further location services e.g. if a Tractor/Trailer arrives in a yard, it can scan for nearby Bluetooth devices during yard movements and loading/unloading, thereby providing data points to allow for context aware tracking of other assets.

This use of existing devices, combined with simple, off the shelf beacons on assets would provide a low-cost solution. Tractor/trailer readings might not be frequent enough due to the sporadic nature of their being on site, so attaching some intelligent device to a moving asset permanently located onsite, such as a forklift, could provide good readings. Site management workers might also be furnished with a tablet with which to monitor orders, read emails etc., which could also be loaded with an application that could scan for nearby devices using the tablet's built in Bluetooth and upload location data from the tablet's GPS radio to some central processing center.

Access to fine-grained location data would provide the data to optimize operations with asset utilization, load-balancing of assets and improved security of assets. Benefits of a RTLS (Real-Time Location System)

- **Reduced Downtime.** Distance travelled by an asset can be tracked, allowing maintenance intervals to be adjusted.
- **Improved Security.** With real-time location data, the disappearance of an asset can be tracked
- **Safety.** With RTLS, location of assets relative to each other and employees can be monitored, and with enough data the trajectory of assets predicted.

- **Improved Vehicle utilistation.** With accurate vehicle monitoring, operational load can be balanced across a fleet, and empty runs avoided.

Here, an asset can refer to any of the following; a vehicle, a trailer, machinery, a container, cargo, workers.

## 1.2 Project Objectives

The objective of this project is to design and build a system capable of providing the following features:

- Network and GPS enabled devices ('smart') scanning for beacons ('dumb') in range
- Scalable cloud services to process data uploaded from 'smart' devices
- An API to allow querying of location information for a given device for reporting purposes
- Simulation to test system with simulated devices

The overall desired system is shown in Figure 1.1 below. The system must be capable of utilizing both real and simulated data and providing reporting capabilities on this data. Basic reporting capabilities should allow querying of a device's current location, which the system will determine based on historical locations for this device with some level of accuracy.

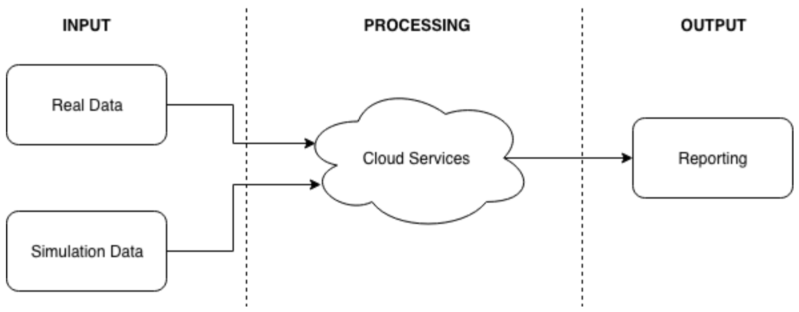


*Figure 1.1. Overall System.*

# 2 Background Research

## 2.1 Research and Analysis of Radio Frequency Communication

There are a wide variety of technologies used for wireless communication available commercially. Some of these technologies are very similar in nature but have very different use cases. These technologies have various ranges, complexities, costs and accuracies.

As this project is exploring the use of low-cost beacons that require little installation and maintenance, the beacons must be reliable and simple. To allow for quick installation, a beacon should be a standalone unit capable of operating without the need for external power. As such it will need to be battery powered, and battery-life will be a primary consideration. Generally, range and battery life are closely related, as the larger the range of a devices the more power it will consume to reach this range, so battery capacity along with power consumption will dictate battery life.

The mobile device will likely need to be connected to external power and can be assumed to have either a constant source of power (as in a machine when the machine is running), or some facility to charge the device. The mobile devices will need GPS radio in order to get a reading for its own location. It will also need to have an onboard module capable of communicating with the asset beacons. The mobile device must have a data connection, either in the form of a SIM card based Network connection or WiFi.

### 2.1.1 WiFi

WiFi uses radio waves to exchange data between devices. Based on the IEEE 802.11 [2] standards WiFi is commonly found in computers and mobile devices. It is primarily used to connect many devices to the same network, typically to allow access to the World Wide Web. The term WiFi encompasses multiple standards, all based on the 802.11 specification. They vary mostly in speed, range and frequency use. Many modern WiFi capable modules are capable of using IEEE 802.11b/g/n at speeds of 11/54/600Mbps respectively and operate at either 2.4GHz or 5GHz. Typically, these devices connect to a wireless access point within 100m.

### 2.1.2 RFID

Radio-frequency identification (RFID) uses a combination of readers and tags to identify objects. Tags are attached to objects to be identified and can be active or passive. Passive tags do not require a battery as they use some of the energy broadcast from the reader to send back a signal with their identification. As such, the greater the range desired from a passive system, the more powerful the reader needs to be in order to get enough energy to the tag, particularly if the location of the tag relative to the reader is unknown and the reader must broadcast over a wide area. Passive RFID is typically used where assets must pass through choke points, and the tag will not be far from the reader. Passive RFID systems typically have a range of 12m or less.

Active RFID tags use on board power to power their return signal. An RFID system using active RFID tags can be set up with a lower powered reader as the tag does not rely on drawing power from the reader. Active RFID systems offer a range of 100m or more.

### 2.1.3 Bluetooth

Bluetooth uses radio waves from 2.4 GHz to 2.485 GHz to transmit data [3]. Bluetooth is commonly found in consumer devices and is typically used for pairing devices together over a short range – up to 100m outdoors. Bluetooth was designer with the purpose of replacing data cables, for example streaming music from a mobile device to a speaker. Bluetooth allows for 2-way communication between devices.

Bluetooth Low Energy (BLE) operates on the same frequencies as Bluetooth and offers similar range but with significantly less power draw. BLE is only compatible with Bluetooth version 4.0 and onwards, as the same hardware can be used for both technologies. BLE is intended for use in the IoT area.

### 2.1.4 Ultra-Wideband

Ultra-Wideband (UWB) is a wireless technology designed to transmit data over a short range. UWB uses multiple frequency bands which reduces susceptibility to noise. UWB operates in the range between 3.1 GHz and 10.6 GHz. Because UWB uses such a wide frequency band to transmit data, it can transmit through objects more reliably (such as doors) than other radio frequencies. Thus, distance between devices can reportedly be measured within 10cm

# 3 Proposed System

## 3.1 System Architecture

Proposed System Architecture is shown in Figure 3.1 below.

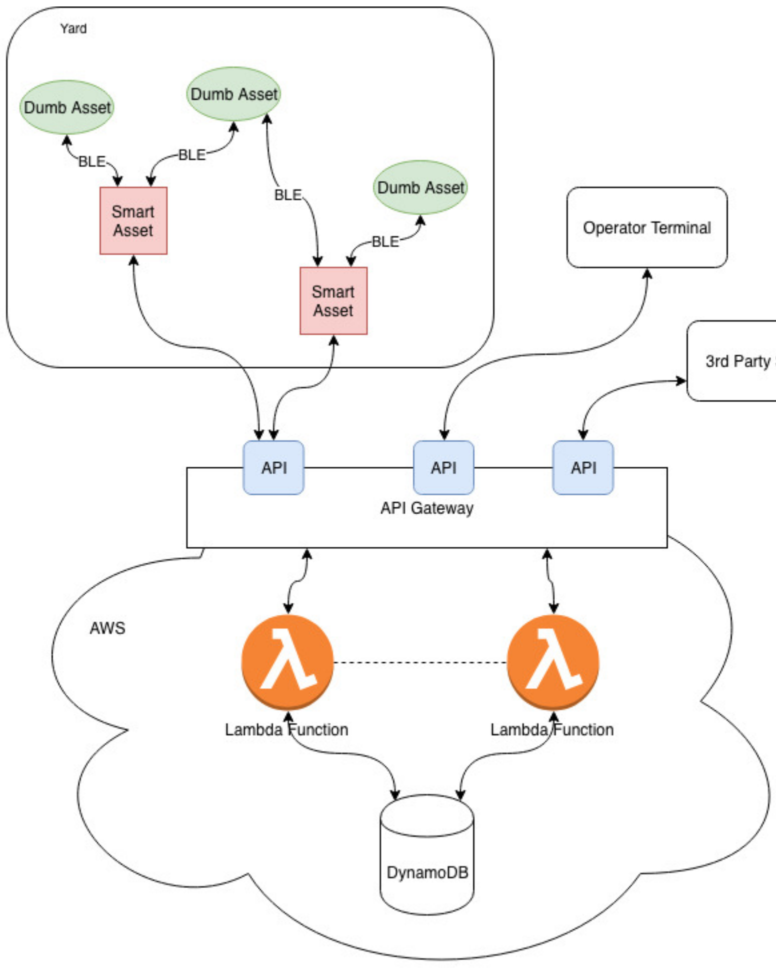


*Figure 3.1. System Architecture.*

This architecture shows an implementation of the desired features as shown in Figure 2.1. The 'Yard' is the physical domain in which the assets are located. Here we have many beacons ('Dumb' Assets) to a few Smart Assets. The smart assets scan for BLE (Bluetooth Low Energy) devices nearby, and upload a list of these devices, along with their own location and a timestamp to AWS (Amazon Web Services). For the purposes of testing this system, a smart device is a Raspberry Pi connected to the internet using WiFi.

The AWS portion of the system consists of three main components; API Gateway, Lambda Functions and a DynamoDB instance. The API Gateway provides a public facing API on a static endpoint and directs API requests to Lambda Functions. Lambda Functions perform computation on data input and data in the database, held in a DynamoDB instance.
The API Gateway provides access to resources that allow requesting of reporting services also, to be used by system operators and in reporting.

## 3.2 Hardware

For the purposes of the project, a Raspberry Pi 3 B+ [4] was chosen as the ‘smart’ device. As stated previously, Bluetooth and WiFi are needed capabilities and this device has both. The device has a Bluetooth 4.2 capable radio, which meets the requirements of being able to support BLE. The device has a 1.4GHz quad-core processor and is, if anything, over powered for the purposes of this project. Importantly, the Raspberry Pi does not have on-board GPS, but has 40 GPIO (General-purpose input/output) pins which give it the potential to utilise an external GPS module. The Raspberry Pi has a large community surrounding projects on the platform, which is an advantage when compared to other, less well-known devices.

The external GPS module selected is the Adafruit Ultimate GPS Breakout board [5]. This module can track up to 22 satellites on 66 channels with a -165dBm receiver. In practice this should yield very accurate location data. Importantly, the module can provide updates at a maximum rate of 10Hz, which will allow the Raspberry Pi to get near real-time updates if needed. There is a GPS antenna on-board (-165dBm), but with a uFL connector allowing the use of an external antenna if needed. This module will be controlled by the Raspberry Pi.

The Raspberry Pi runs a Python script to poll for nearby Bluetooth devices, get updated latitude and longitude and upload the resulting data as a JSON (JavaScript Object Notation) document to the API Gateway over WiFi.

## 3.3 Cloud Services

As previously stated, cloud services are built using components provided by AWS. These components consist mainly of an API Gateway, AWS Lambda Functions and a DynamoDB instance [6]. An outline of the cloud services architecture is provided in Figure 3.2 below. Note that each component of the cloud services solution is a stand-alone service, with little specific dependencies on the other components. This is deliberate, so that components can be changed and updated with as little impact on other components as possible. For example, the API Gateway could instead point requests at a more traditional server, which could host a database itself, all without any externally noticeable changes. This loose coupling of components is what will provide the desired scalability.

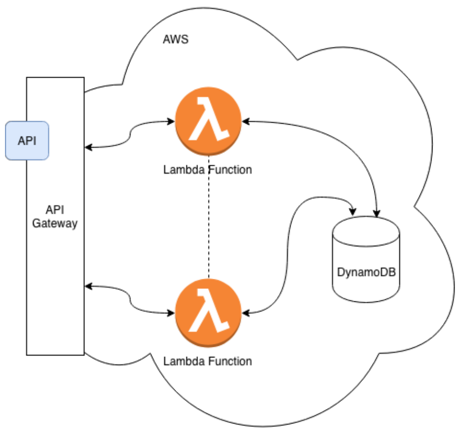


*Figure 3.2 Cloud Services Architecture.*

The API Gateway is the only public facing component and provides a RESTful API. It acts as the 'Front Door' to the cloud services. It exposes resources e.g. /api/readings for use with REST (Representational State Transfer) methods. API Gateway triggers Lambda Functions based on the resource requested, and acts as a load-balancer.

AWS Lambda provides high availability compute infrastructure, without the need to provision, scale or manage servers [6]. In essence, a Lambda Function is a script that is only run when an event is triggered. In this application, these events are triggered by API Gateway. Lambda Functions can be built in many languages, but for this project Node.js was used. Node.js is an open source project providing an asynchronous, event driven JavaScript runtime [7] which is ideal for implementation of a server-side application. Node.js can be run locally on a test machine to test cloud behaviour. Lambda Functions perform read and write operations to the DynamoDB instance. They validate data uploaded from devices and upload the DB (Database). Lambda Functions also perform computation based on data in the DB.

To deploy Node.js application code to AWS Lambda and API Gateway, Claudia.js was used. Claudia.js is an open source project that provides an automated deployment and configuration platform that increases the deployment speed and reduces configuration error when deploying application code to AWS [8].

Amazon DynamoDB is a key-value and document database that provides automatic scaling [6]. DynamoDB is serverless, meaning it runs as a stand-alone instance and doesn't require server management or provisioning. It automatically scales tables depending on capacity and currently needed performance. DynamoDB stores system data for the entire application.

## 3.4 Simulation

Testing of the constructed system requires a lot of data to be supplied to the system and queried from the system. The following things are required to be tested;

- **Load handling of the Cloud Services.** Can the system handle bursty traffic? How does the system respond to peak traffic?
- **Accuracy of location predictions.** How accurately can the system predict the location of an asset, given historical information? How does this accuracy change with noisy data, bursty data, varying ratios of devices?

In order to simulate locations of assets effectively, simulation software will be used. There is simulation software available that will allow modelling of a site/yard and will allow outputting of the resulting data. This simulation output will have to be converted to a format that the system expects, fed into the system and the reporting from the system analysed. The following Figure 3.3 shows the proposed simulation architecture.

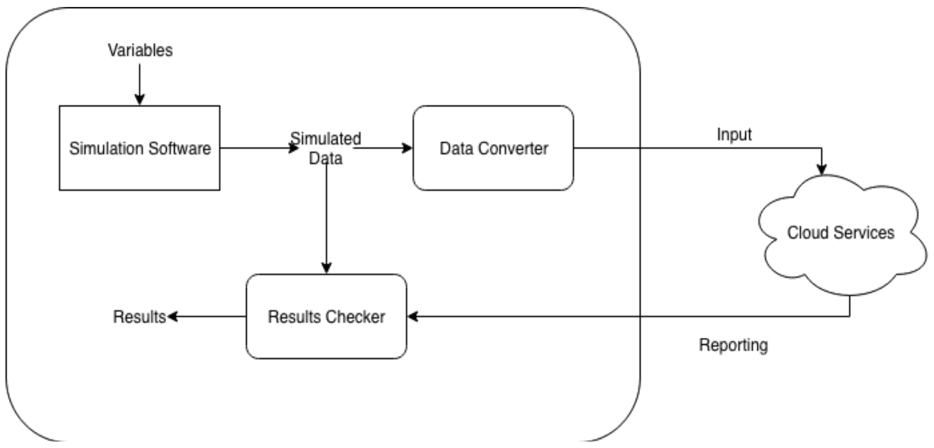


*Figure 2.3*

The proposed solution is to wrap the simulation software in a custom piece of software that will control the simulation software, convert the output data the format expected by Cloud Services and compare the results from Cloud Services against the simulated results. This solution would allow testing to be automated and would provide a platform for more extensive testing, such as testing of accuracy over a time duration of multiple days of data.

# 4 Results to Date

The progress thus far has largely involved putting in place the components and architecture needed to build features on top of. A diagram of the system currently in place is shown below in Figure 4.1.

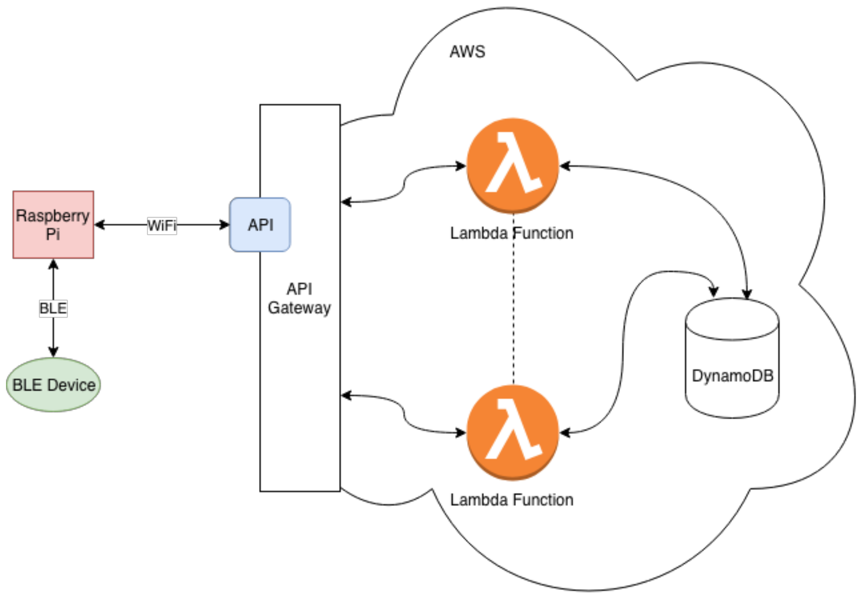


*Figure 3.1*

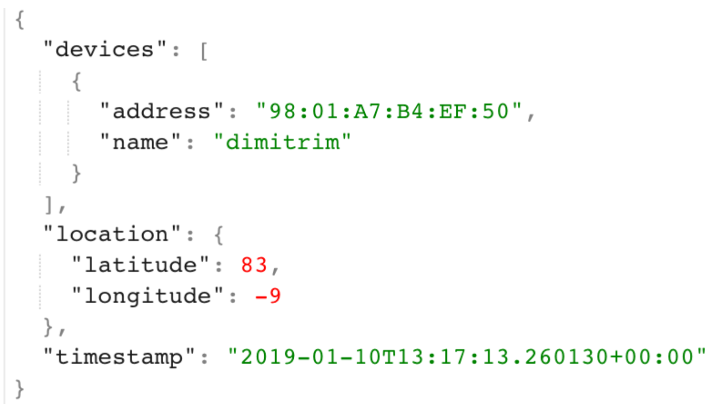
## 4.1 Hardware

A Raspberry Pi 3 B+ has been used as the trial run and is currently running a Python script that will scan for nearby Bluetooth devices and upload a list of nearby devices along with the current timestamp and mocked location data to the API Gateway using WiFi. The script is capable of continuous scanning or on-demand scanning.

The Adafruit Ultimate GPS Breakout module has been ordered.

## 4.2 Cloud Services

The format for input data is defined as shown in Figure 4.2 below. Note that the `devices` field is in fact a list of devices, in this instance containing a single device.



```
{
  "devices": [
    {
      "address": "98:01:A7:B4:EF:50",
      "name": "dimitrim"
    }
  ],
  "location": {
    "latitude": 83,
    "longitude": -9
  },
  "timestamp": "2019-01-10T13:17:13.260130+00:00"
}
```

*Figure 4.2. JSON input data.*

An API Gateway has been deployed in front of a Lambda Function. One resource has been exposed, `https://5o6pmquqyi.execute-api.us-east-2.amazonaws.com/latest/readings`, allowing a `PUSH` request to upload data from a device. The Lambda function writes this data to an instance of DynamoDB. The DynamoDB instance has one table, `readings`, which uses timestamp as the primary key, and shows `devices` and `location` as further attributes of the same item.

## 4.3 Simulation

Research as to possibilities of using existing simulation software tools has shown that there are multiple possibilities for simulation software. However, this simulation software will have to meet the following requirements;

- **Programmatic Control.** In order to run multiple simulations, the simulation software must be controllable programmatically to allow adjusting of variables such as data noise, frequency of asset movement, speed of asset movement, frequency of asset scanning etc.
- **GPS Location Simulation.** The software must allow modelling of some yard/site and assigning of latitude and longitude information to the yard/site area. This will also require that the software allow definition of finite domain.
- **Useable Output.** The output from such simulation software must be in some format so as to allow feeding into the Cloud Services.

With the above requirements in mind, it appears that simulation software must be Agent Based, i.e. the software must be centered around the modelling of Agents. Attributes can be assigned to Agents. The software must also have GIS (Geographic Information System) capabilities, so as to allow latitude and longitude simulation. The following simulation software packages have been found that meet the above requirements;

### 4.3.1 GAMA

GAMA is an open source simulation development environment for building spatially explicit agent-based simulations [9]. GAMA is based on Java and allows instantiation of agents from datasets including GIS data.

### 4.3.2 AnyLogic

AnyLogic is a company providing simulation software for use in multiple industries, most notably transportation [10]. AnyLogic provides software for a free trial and is Java based, allowing custom integrations in Java to be written. The software allows building of a custom area and complex agent behavior classification.

### 4.3.3 Brinkhoff Generator and variants

This simulation software was originally built by Thomas Brinkhoff to simulate behavior of moving objects [11]. This software has been extended by more recent efforts, including Hermoupolis, a trajectory generator [12]. Hermoupolis has been released as an open source project, with source code and datasets available on request [13].

# 5 Discussions and Conclusions

There remains plenty of work to be done. The foundation of the data input and cloud services sections of the project has been laid, features need only be built on top of these components.

## 5.1 Hardware

As far as data input is concerned, GPS data has been so far mocked up. When the GPS module is received, it needs to be connected to Raspberry Pi and the script updated to poll for GPS location also. Tests need to be run on the GPS module to determine module accuracy. This can take the form of leaving the GPS module at a known latitude and longitude and recording the location information the GPS module reports for 24 hours. This data will provide a baseline the noise that can be expected in GPS data, which will feed into simulation.

BLE beacons have not been used so far, testing has used Bluetooth devices such as a personal computer and a mobile phone. The specifics of the Bluetooth devices used thus far have been irrelevant, but purpose built BLE beacons will have to be acquired and tested to assess the feasibility of using such a beacon in real-life situations. This will require testing of BLE performance and sensitivity both in ideal situations and in real-life situations with obstacles, such as in a car park. Again, this data will help inform the accuracy of simulation.

## 5.2 Cloud Services

The outline of the cloud services has been assembled and the components are working together as intended. More resources must be exposed to allow querying of the estimated current location of an asset. This will require an implementation of an algorithm to look at historical device data and estimate current location based on such. More work is needed on what form these resources will take and what resources are needed to provide useful features from the system. Such features could include distance travelled for an asset, site hot-spots etc. The location prediction algorithm will either need to run on-demand as location predictions are requested, or on every location update that is received from a device. This algorithm will need to consider noise in input data and age of data – if a device appears to have moved 3 times in the last 10 minutes, data from 15 minutes ago will not point to the device’s current location.

## 5.3 Simulation

Simulation software remains to be selected. Both AnyLogic and GAMA look particularly promising with AnyLogic seemingly having more use in industry and a greater variety of use cases. GAMA has the advantage of being open source, and as the simulator will likely need to be wrapped in a custom program this could prove useful as the source code is freely available for inspection and modification. Mode research is needed.

Simulation of asset movements is the interesting part of this project and will allow testing of the system and particularly guide development of features. Once simulations can be run, the potential of what can be experimented on and built is huge.

## 6 References

[1] Federal Motor Carrier Safety Administration, “Electronic Logging Devices Implementation Timeline,” U.S Department of Transportation, 2018.

[2] IEEE, “IEEE std 802.11-2012 (Revision of IEEE std 802.11-20017),” 2012.

[3] Bluetooth, “Core Specification v5.0,” 2016.

[4] “Raspberry Pi 3 B+,” Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Accessed 6 January 2019].

[5] “Adafruit,” Adafruit, [Online]. Available: https://www.adafruit.com/. [Accessed 6 January 2019].

[6] “AWS Documentation,” Amazon Web Services, 2019. [Online]. Available: https://docs.aws.amazon.com/. [Accessed 6 January 2019].

[7] “Node.js,” [Online]. Available: https://nodejs.org/. [Accessed 6 January 2019].

[8] “Claudia.js,” [Online]. Available: https://claudiajs.com/. [Accessed 6 Jan 2019].

[9] UMNISCO, “GAMA Platform,” [Online]. Available: https://gama-platform.github.io/. [Accessed 6 January 2019].

[10] “Anylogic,” The AnyLogic Company, [Online]. Available: https://www.anylogic.com/.

[11] T. Brinkhoff, “Generating Network-Based Moving Objects,” GeoInformatica, Vol. 6, No.2, 2002.

[12] N. Pelekis, S. Sideridis, P. Tampakis and Y. Theodoridis, “Hermoupolis: A Semantic Trajectory Generator in the Data Science era,” he SIGSPATIAL Special Newsletter of the Association for Computing Machinery, Special Interest Group on Spatial Information, Vol. 7, Number 1, March 2015.

[13] “Hermoupolis Download Request,” [Online]. Available: http://infolab.cs.unipi.gr/?page_id=2255. [Accessed 6 January 2019].