ASHESI UNIVERSITY

CATTLE TRACKING WITH LORA AND MACHINE LEARNING

CAPSTONE PROJECT

B.Sc. Computer Engineering

Sula Thembumenzi Mabuza

2021

ASHESI UNIVERSITY

CATTLE TRACKING WITH LORA AND MACHINE LEARNING

CAPSTONE PROJECT

Capstone Project submitted to the Department of Engineering, Ashesi University in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Engineering.

Sula Mabuza

2021

DECLARATION

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature: S.Mabuza

……

Candidate's Name: Sula Mabuza

……

Date: 27/04/2021

……..……..……

I hereby declare that preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University College.

Supervisor's Signature:

……

Supervisor's Name:

……

Date:

……

Acknowledgments

To my supervisor, Dr. Nathan Amanquah, whose encouragement and academic advice helped me undertake this project.

Abstract

To curb the increase in livestock theft in Sub-saharan countries, this project looked at machine learning applications like cattle tracking on a farm. Received signal strength indicators from home available WiFis were utilized to fingerprint locations on the farm. Where WiFis were not available, ESP866 Node MCUs were used to deploy access points. These ESP8266 Node MCUs were also given SSIDs that could easily be used for classification using the SSIDs as labels in the program. Two classifiers were explored, the support vector machine (SVM) and the Decision Tree. A linear kernel with a very low gamma value of 0.001 was used for the SVM classifier. The decision tree classifier yielded an average accuracy of 0.89683659, while the support vector machine yielded an average accuracy of 0.915699. The SVM classifier emerged as a suitable classifier to use to achieve reliable accuracy in cattle tracking. Lora proved to be an essential tool in wireless communication of the prediction results from the pendant/transmitter designed to the receiver, hence enhancing mobility in tracking. Future works include incorporating distance calculation to make the tracking process more straightforward and efficient. Sensor nodes that will further provide the machine learning algorithm designed with unique data values will also be incorporated to compensate for longer farms that might need the deployment of more Node MCU access points. This is an effort to maintain low costs in tracking using the solution provided by this project. Therefore, the solution presented by this project proved to be feasible.

Table of Content

Chapter 1: Introduction

Ever since the establishment of the Nguni people, livestock in cattle has played a pivotal role in African countries, especially Southern Africa. The use of cattle in Southern Africa ranges from meat products and milk products to paying a bride price to acquire land through chiefdom negotiations referred to as Kukhonta in Siswati. Cattle are also used for traditional activities like offerings to ancestors as well as cleansing. As the importance of cattle increases in Africa, with the rise in poverty and lack of food parcels, cattle theft increases during the COVID-19 pandemic. In the year 2020 alone, at least 218000 farm animals, including cows, sheep, and goats, were stolen in South Africa [9]. This drastic increase from the 180000 livestock reported missing in the past five years [9]. The total monetary value lost due to livestock theft in 2020 is estimated to be about nine-hundred million rand which is about sixty million dollars; this economic value loss remained constant in the past two years. It is projected that by the end of 2021, the number of livestock stolen will increase as the pandemic continues to make earning a living lawfully tricky [9].

Undoubtedly livestock theft is a serious problem not just in South Africa but in all Sub-Saharan countries. There remains a need for a robust and improved way of ensuring the security of cattle in Africa. Some of the current tools that farmers use to track and monitor their cattle include using closed-circuit television to monitor farms, using drones to surveil farmers, and using GPS – tagging to track livestock as they graze. These are high-cost means of curbing the increase of cattle theft, hence leaving low-income earning cattle farmers are at risk as they cannot afford such means [9].

Therefore, one of the pressing problems that this project is solving is the high cost of cattle tracking systems. The approach introduced by this project ensures the successful

tracking of cows at 10 km at a meager price. This project is in the case of South Africa but can be spread to other communities. The target is low-income farm earners whose main aim is to grow a business in commercial livestock farming and subsistence livestock farming, producing mainly for family-related activities. The project introduces a pendant that can be tagged on cattle for remote tracking using received signal strength indicators and machine learning.

1.1 Background

Tracking was introduced alongside the Global Positioning System (GPS) introduction in the 1960s [11]. GPS Tracking Technology was initially designed for military and intelligence applications, with its main inspiration coming from the Soviet spacecraft Sputnik in 1957 [11]. The global positioning system consists of satellites orbiting the earth at fixed points above the planet and transmit signals to anyone on earth with a GPS tracking device. The first movement tracking was introduced in 1978 with a vehicle tracking system when the first GPS Satellite called Block – I was launched into space [11]. Rockwell International designed this first satellite. In 1985, over 10 Block – I satellites were launched into space [11]. Since then, tracking has increased with its application extending to our daily life activities.

In 1997, another means of wireless communication that this project will use for localization named Wireless Fidelity (WiFi) was introduced [12]. A committee created WiFi called 802.11 [12]. It permitted wireless transmission of data between devices. This led to the creation of IEEE802.11, which refers to a set of standards that define communication for wireless local area networks (WLAN). It further sparked development in prototype equipment like routers to comply with IEEE802.11, and in 1999, WiFi was introduced for home use [12]. WiFi uses electromagnetic waves to communicate data that runs at two main frequencies: 2.4Ghz and 5Ghz. WiFi networks have a limited range,

mainly influenced by the frequency, transmission power, antenna type, location, and environment. A typical wireless router has a range of between 50 meters to 100 meters.

In 2009, Lora was introduced by Nicolas Sornin and Olivier Seller [10]. The main aim of developing Lora was to create a long-range and low-power modulation technology. Lora was then taken under Cycleo before being acquired by Semtech in May 2012 [10]. They further improved the technology and finalized the chips required for the end devices to use Lora to transmit data. Semtech currently has outsourced necessary tools that permit the use of Lora to determine location. One of the services Semtech provides is the Lora Cloud Geolocation [10].

To curb the high prices associated with location tracking and minimize power consumption, this project uses the off the shelf WiFi and Lora to determine location. Lora permits wireless communication of data using a radio modulation technique generated by the Lora transceiver chips used in this project. This modulation allows long-range sending of small amount of data, high immunity to interference while minimizing power consumption. Therefore, it will enable long-distance communication with low power requirements. Hence, Lora is used for sending the location results from the WiFi module used in this project to the cattle farmer's computer, where the cattle's location will be monitored.

WiFi can make known the received signal strength indication. The received signal strength indication is the measurement of the power present in a received radio signal. Signal strength can vary greatly and affect wireless networking functionality. IEEE 802.11 devices make these measurements available to users. The variance in the signal strength is vital to this project. It allows for mapping where the signal strengths were collected to where a packet was sent from, possible. It provides for the classification of locations

according to their signal strengths. Machine Learning will then be used to classify and predict location using received signal strength indications.

Machine Learning (ML) uses algorithms and neural network models to assist computer systems in predicting and progressively improving their performance. Machine Learning algorithms automatically build a mathematical model using sample data referred to as training data – to make decisions without being specifically programmed to make those decisions. Machine Learning was first introduced in the 1950s through a computer program for playing checkers by Arthur Samuel of IBM [13]. It was first referred to as Machine Learning in 1952 [13]. Machine Learning can be used in various ways, such as learning to play games, speech recognition, and facial recognition.

1.2 Summary

The pressing problem that this project seeks to solve is cattle theft in Sub-saharan countries. To do that, the project uses off-the-shelf resources like WiFi, Lora, and Machine Learning. The main aim is to produce a cattle tracking solution that is cheap, accurate, and reliable.

Chapter 2: Literature Review

This literature review covers some of the projects that have been carried out before, what was discovered on each location tracking approach, and how the lessons learned from them can shape the outcomes of this project.

2.1 Location Tracking

A device's position is usually estimated by monitoring a location-dependent parameter (LDP). Such as received signal strength indicator, time of arrival of sent data packets, time difference of arrival of sent data packets, etc., from another device whose location is known. Localization is done by applying estimation techniques on the location-dependent parameters to find the transmitting device's position [5]. The localization accuracy is mainly a factor of the location-dependent parameters' measurement nature and precision, the wireless standard, and the estimation technique.

The main estimation techniques are multilateration, trilateration, and fingerprinting. In multilateration in a 2D coordinate system, the distance r from a point p specifies a circle of possible positions.

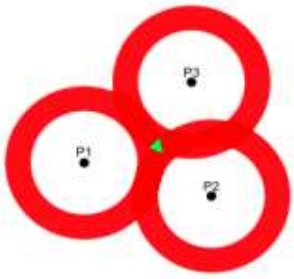

Figure 1: Point P in a circle of radius R

When three or more approximate distances (radius) are known, an area of confidence can be determined if the boundaries overlap, as shown in figure 1 [8].

Fingerprint algorithm-based positioning consists of two phases: the offline and online phase. In the offline phase, signals' characteristics to predict the position are collected and stored in a database for sample points in the service area. A site visit is mandatory to collect the necessary data/signal to identify that location. An end-device location is estimated and provided through a database search [2].

2.1.1 Using GPS

A readily available technique of determining location is using GPS chips, which are available almost at any tech market. These GPS chips can provide real-time location. Global Positioning Systems and Global Navigation Satellite Systems (GNSS) have offered, up to now, localization and navigation services. GPS and GNSS can only perform localization if the GPS receiver is visible with at least four satellites. In indoor and dense environments, this condition is usually not guaranteed [5].

Global positioning system chips have a relatively high cost and power consumption, making them unsuitable for long-range positioning determination and use in low-power Internet of Things (IoT) devices [2].

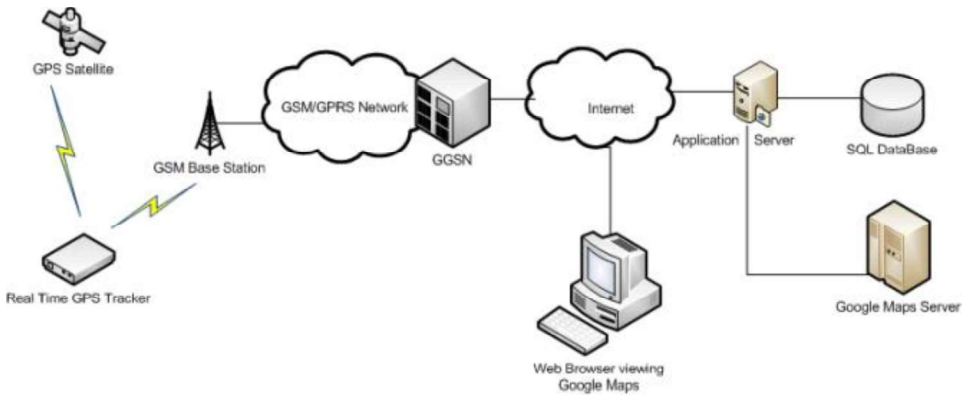

Figure 2: Demonstration of Real-time GPS tracking

2.1.2 Using Lora

Semtech designed and created Lora gateways and Lora transmitters, which, together with the Lora cloud geolocator, can determine location. Cattle farmers can easily use such a concept to track and monitor their cattle.

In Semtech's design, A Lora gateway consisting of a Lora transmitter module is used as the object to be tracked or located. The transmitter sends data packets to nearby gateways. The Lora gateways are each fixed with effective timers, and they are in a designated and known location. It is easier to find nearby gateways with an available gateway to connect to when using Semtech's The Things Network (TTN), which offers location calculation tools to help track a cow's location. Whenever a transmitted data packet is received, information about it is stored in a server in TTN; the information includes the received signal strength, time of arrival, and signal to noise ratio. From this server, information about the data packet is retrieved by Semtech's Lora cloud geolocator algorithm. The Lora cloud geolocation algorithm uses a geometric approach to determine the transmitting object's exact location – the Lora transmitter.

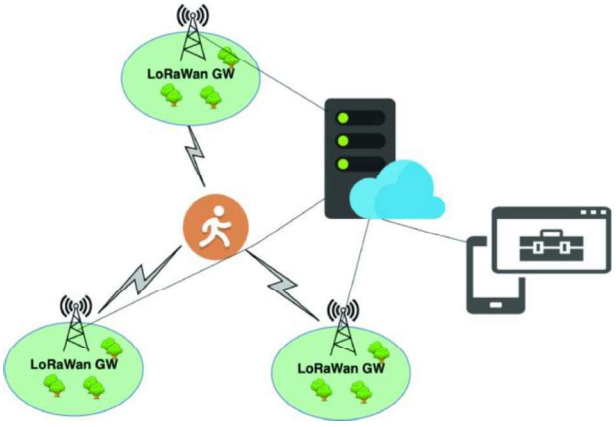

Figure 3: Semtech's approach for tracking

Semtech's design requires three or more Lora gateways depending on the location estimation approach being used and the accuracy. The method also requires about three

GPS modules to locate the three Lora gateways used. The registration of a new gateway on TTN leaves a possibility of it being used by another person for their tracking purposes, especially if the gateway is the only one nearby. All location determining calculations performed in this design is done in the cloud; hence there is a need for an HTTP protocol and internet connectivity.

2.1.3 Using TDoA and RSSI

In a project to track a robot car's location, Choi set up a hybrid localization project that used the time difference of arrival (TDoA) and received signal strength indication (RSSI) to track a robot car's location [5]. Two test boards were used in setting up for the RSSI, one transmitting a radio frequency (R.F.) signal on two alternating frequencies. The other board was receiving and taking measurements based on the alternating frequencies received. The receiving board was left stationary while an interval of five centimeters incremented the transmitting board's distance. For the arrival time difference, an omnidirectional microphone, a piezo buzzer, and an eZ430-RF2500 board with tones generated via an onboard pulse with modulator were used. Audio signals were sent as data packets, and their received signal strength indication was extracted to determine the position of the robot car.

A sound-reflecting cone was incorporated above the buzzer, projecting the audio pulse omnidirectionally about the robot's workspace. To digitize and process the audio signal to detect when the robot arrived with the R.F. packet, an MSP430 was used. The MSP430 had an internal 10-bit analog-to-digital converter (ADC10) and could transfer at most 255 continuous samples directly into memory without CPU intervention or interruptions. The ADC10 was programmed with a sampling rate of 80kilosamples/sec, providing a range of 1.09 m. To increase the distance, two options were applied; the first one was to slow their clock rate at the cost of distance resolution or delay the onset of

sample acquisition to shift the sampled range away from the microphone. For distance calibration using the arrival time difference, one board was set to listen for the arrival broadcast's time difference. Another with a buzzer was set to broadcast the time difference of arrival signals. Measurement resolution was taken to be the distance equivalent to one standard deviation of measurement error. This experiment showed that received signal strength indication and time difference of arrival could be used for distance sensing [5].

Machine learning techniques could potentially make use of more complex patterns in the measurement data, allowing for increased accuracy in location tracking [5].

The two methods by Semtech and Choi introduce important concepts that were used in this project. The first one is setting up the project, using two boards, one transmitting data packet and the other receiving these data packets dispatched using radio frequency signals. Even though Choi had a problem with the range of transmission, Semtech's Lora can solve this problem as it can achieve 10 km of data transmission without a repeater requirement. Repeaters are network devices that amplify or regenerate an incoming signal before retransmitting. The only thing that could be a problem with Choi's approach is audio as the data packets being transmitted, like the cowbell, if applied in cattle tracking, would alert everyone in the community that nearby cattle could yield negative results.

This project did not fully apply Semtech's set up as it required at least three Lora gateways to determine location. Even though this setup perfectly covers long distances, the cost to set it up was high. The cost of each Lora module was 157 GHS. Four Lora modules were required in total, which would have escalated the setup costs to 628 GHS. Semtech's setup also requires that each Lora gateway be constructed with a Raspberry Pi to register it on The Things Network properly. The Things Network is an open-source web

application by Semtech that provides access to a Lora Wide Area Network, which can be used for various IoT applications, including location tracking. Each Raspberry Pi for each Lora gateway would have cost about 201.25 GHS, adding to the setup costs. While high-income-earning cattle farmers could easily afford this, this project had to explore a much cheaper alternative to accommodate all. Hence, the project used two Lora gateways for long-range sending of scan results to allow remote tracking and a Node MCU to enable received signal strength indication scans of nearby WiFi access points.

A Node MCU is a low-cost open-source Internet of Things platform. The name Node MCU combines node and Micro-controller Unit. It is a firmware for which prototyping board designs are available. This micro-controller unit can be easily interfaced with Arduino Integrated Development Environment to be easily programmed and connected with WiFi. Arduino Integrated Development Environment (IDE) contains a text editor for writing code. Figure 4 below shows an example of a Node MCU.

Figure 4: ESP8266 Node MCU

2.2 Characteristics of Lora, GPS, and WiFi

The table below presents the advantages and disadvantages of incorporating Lora, GPS, and WiFi in a location tracking device.

Technology	Advantage	Disadvantages
GPS	Increase accuracy	High power consumption
	Ranged unlimited	Requires a clear sky
WiFi	Good in areas with WiFi routers	Requires good WiFi coverage
	Free to use	Requires Google database
Lora	Very low power	Requires at least three gateways
	Good building penetration	Requires good timing and location records
	Good distance performance	

Chapter 3: Design Requirements

In this chapter, design considerations are presented, including hardware specifications and software specifications. The proposed solution will be introduced, including how farmers can make use of it.

The primary aim of this project is to develop a cheaper solution to cattle tracking on a farm. It is meant to be used by subsistence and commercial farmers looking for a cheap and easy way to monitor their cattle.

3.1 Solution and User Requirements

The primary use for this solution is a low-income-earning cattle farmer. The system would be accessible via computer software that will permit monitoring on the screen, i.e., Arduino IDE's Serial Monitor.

3.1.1 Use Cases

Scenario:

Tebogo Zuma is a cattle farmer who has been in the cattle farming business for a year. Unfortunately, Tebogo is among the cattle farmers whose livestock had been stolen, as reported by the Economist [9]. This has put a burden on Tebogo's family as they rely primarily on her cattle farming business to make ends meet as compared to her sewing business. Tebogo cannot physically go to the farms and look after her cattle for the whole day as she has to keep her sewing business running. Luckily for Tebogo, she is presented with this solution; upon accessing it, she can track her cattle from her room where she is sewing. She can see when the cattle are moving closest to the farm fence; hence can always go outside to stop them from going into the nearby forests where they are prone to be stolen. This puts security in her yet-to-boom cattle farming business.

3.1.1 User requirements

The user should be able to:

- Track cattle offline
- Accurately track cattle at low costs
- Track remotely over a distance of 10 km
- Access the scan results via computer software that permits tracking offline

3.1.2 Solution requirements

The solution should be able to:

- Track at lower costs compared to already existing solutions
- Be able to track offline using already available resources
- Permit mobility, such that the user can follow along while tracking
- Localize regardless of the weather conditions and the thickness of foliage it is put under.

3.2 Project Design Objective

The project's main objective is to use machine learning and received signal strength indications from nearby WiFi access points to create a cheaper and accurate version of a cattle tracker. The main parameters driving the development of this project are costs, accuracy, and mobility. The design should provide an easily implementable solution regardless of the availability of WiFi routers. Figure 5 below shows a high-level architecture of the design, demonstrating how it works.

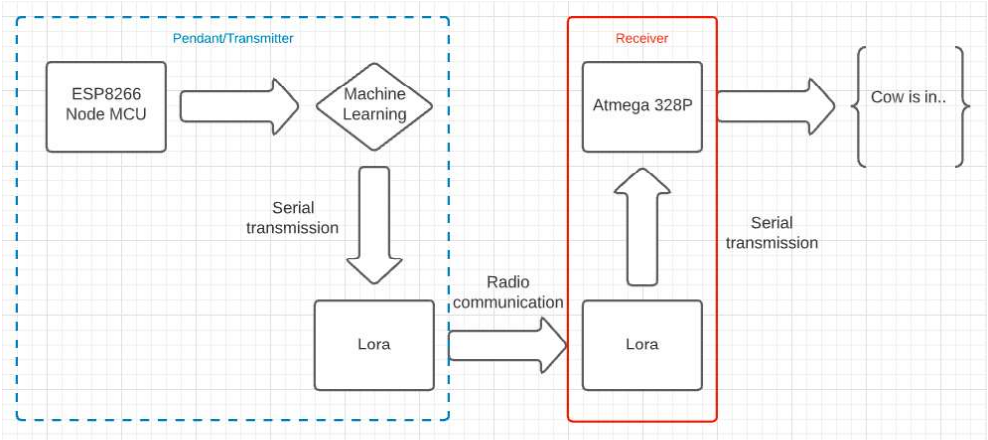

Figure 5: High-Level Architecture Design

In this project, a Node MCU ESP8266 is used to scan for nearby access points and determine their signal strength. The acquired information on access points available and their signal strength is then structured in a vector form in a string using the embedded software running on the Node MCU. The obtained information containing the Service Set Identifiers (SSIDs) and the received signal strength indications are then fed to a machine learning algorithm that then uses supervised learning to determine the animal's location. A service set identifier is a 32-character sequence that uniquely identifies a wireless local area network (WLAN). SSIDs are simply the names of the wireless networks scanned. The prediction results are then sent via Lora to the receiver for monitoring.

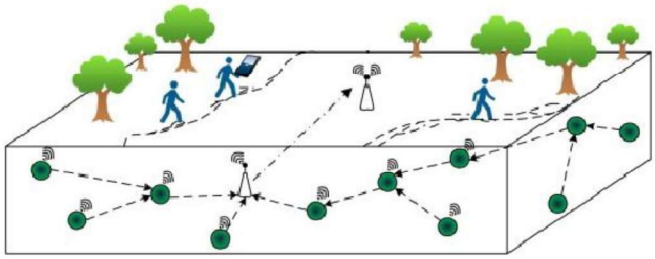

Figure 6: Scan being done on nearby WiFi access points

Figure 6 demonstrates how a scan will be done when implementing the machine learning algorithm's classifier. To create a classifier, the first thing that will be done is moving around the area and collecting SSIDs and RSSIs from nearby routers, as shown in figure 3. This information will be stored either in a CSV format or in a vector format. This information will then be used to train the algorithm and use as a reference when new scans from the pendant that the cow will have in the farm determine its location.

Machine Learning offers three types of learning, supervised learning, unsupervised learning, and reinforcement learning. **Supervised learning (S.L.)** is the machine learning process that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of the training example. The data set is appropriately labeled so that the algorithm can know what the data represents. **Unsupervised learning (U.L.)** is a type of algorithm that learns patterns from unlabelled data. The hope is that through imitation, the machine is forced to build a compact internal representation of its world.

This project uses supervised learning. Therefore, the proper hardware must be set to collect and label the data accordingly to improve the algorithm's learning and better location predictions.

3.3 Hardware Design Decisions

This section presents the hardware behind the pendant that will be pinned to the cow that will be tracked and the hardware behind the receiver. It also elaborates on how the necessary data will be retrieved from the cows and determine location remotely. The transmitter side represents all the hardware that will be on the pendant. The receiver side represents the hardware used to receive the data and allows interaction between the software and the hardware components.

3.3.1 Pendant/Transmitter

The components used in this circuit design that represents the transmitter or pendant pinned on the cow consist of a Lora transceiver module, a breadboard power supply, and a Node MCU ESP8266. A transceiver is a device that can both transmit and receive communications, in particular a combined radio transmitter and receiver. The Lora transceiver module allows for the transmission of scan results of nearby wireless networks (WiFi) from the receiver's pendant. The machine learning algorithm will use the data to predict the location. The breadboard power supply supplies the necessary voltage to the components that make up the pendant. The Node MCU is used for scanning.

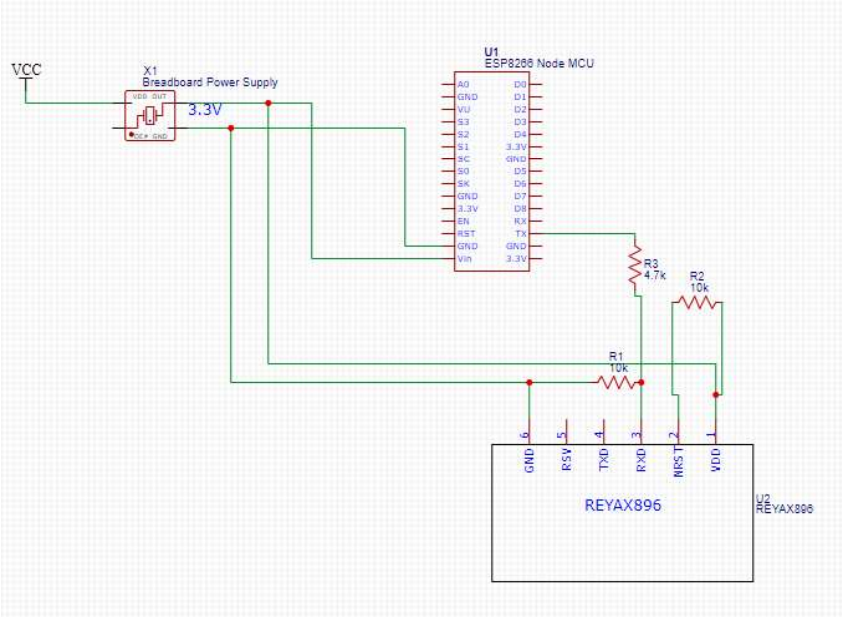

Figure 7: Pendant/Transmitter circuit design

The Lora transceiver has six pins, GND, RSV, TXD, RXD, NRST, and VDD. The operating voltage for the Lora transceiver is 3.3V. A 4.7K ohms resistor is connected from the RXD pin of the Lora transceiver to the TX of the ESP8266 Node MCU. Such a connection sets this transceiver as the transmitter. The resistor's use reduces the amount of power going to the Lora transceiver through the RXD pin from the Node MCU. This ensures that the module is not damaged. The Lora transceiver's NRST pin is connected to

the VDD of the Lora module using a 10K ohms resistor to enable the negative reset options of the Lora transceiver. The R.X. pin is also connected to the GND of the Lora module using a 10K ohms resistor; this creates a voltage divider that steps down the current going through the R.X. pin while completing the circuit. It prevents the module from being spoilt.

3.3.2 Receiver

The components in this circuit design representing the receiver used to receive and load the acquired data into the computer for use by the machine learning algorithm include an Atmega 328 p, a Lora transceiver module, and a breadboard power supply. The Atemga serially sends the acquired data packets from the Lora transceiver (configured as a receiver on the receiving end) to the computer using a USB to serial converter. The Lora transceiver is configured as a receiver and used to receive the sent data packets, and the breadboard power supply supplies the right power for all the components.

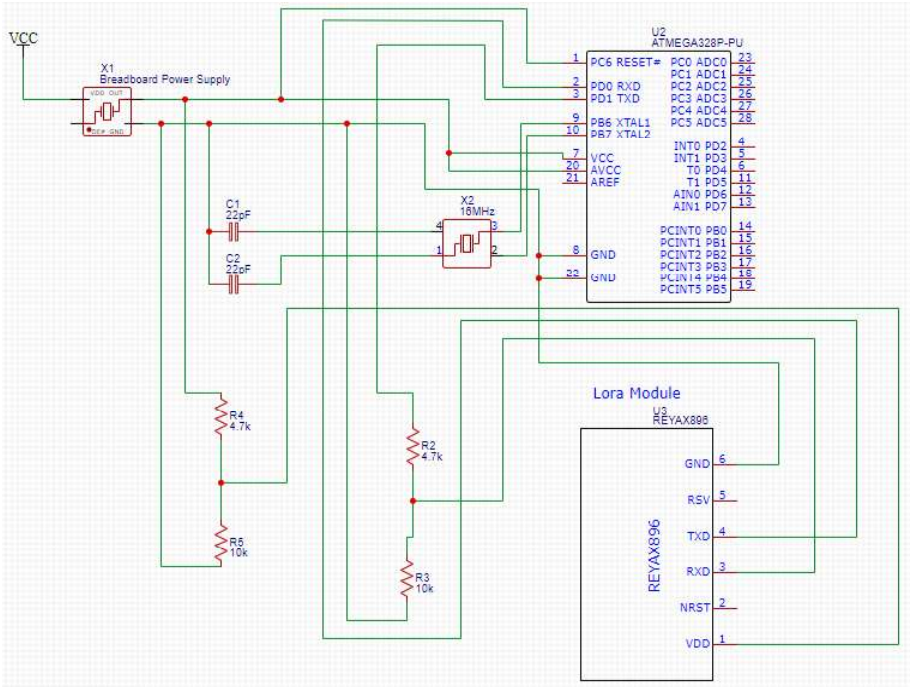

Figure 8: Receiver circuit

Figure 8 above shows how the connections were made using an Atmega 328p chip. The output voltage of the power supply was 5V. The reset pin of the Atmega 328p was connected to positive for the normal running of the chip; to reset it, the pin is connected to the ground. The inbuilt clock of the Atmega chip is 8 MHz, the connected 16 MHz crystal permits it to operate at 16 MHz, similar to Arduino. To configure the transceiver as a receiver, the TXD pin is connected to the RXD of the Atmega 328p. Lora uses 3.3V. Therefore the 4.7K – 10K voltage divider steps down the voltage to the right amount to not damage the module. A PCB can be created from the above circuit in figure 8. The data was acquired from the Atmega chip serially using an FTDI (USB to serial converter). Figure 7 below demonstrates the connections made to permit that.

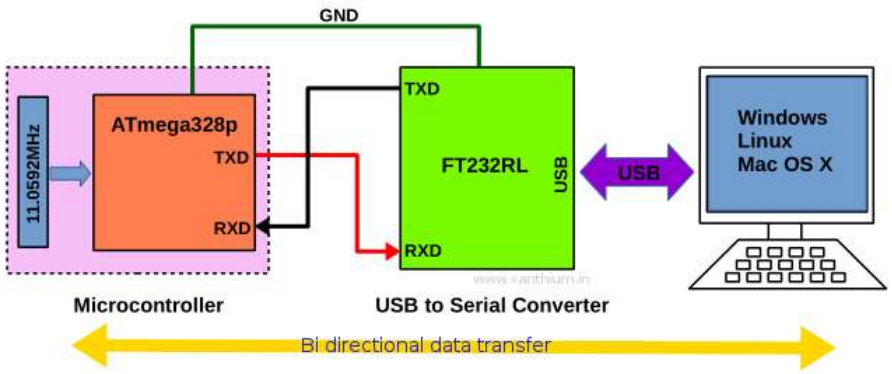

Figure 9: Serially acquire data from the atmega

Figure 9 below demonstrates how the stated components will communicate with each other to enable successful cattle tracking. Pendants will be attached to the cattle's necks, and they will use ESP8266 Node MCU to scan for nearby access points, as demonstrated in figure 10. Every time a scan is done, the machine learning algorithm uploaded on the ESP8266 Node MCU predicts the location. The results are sent to the receiving Lora on the receiver.

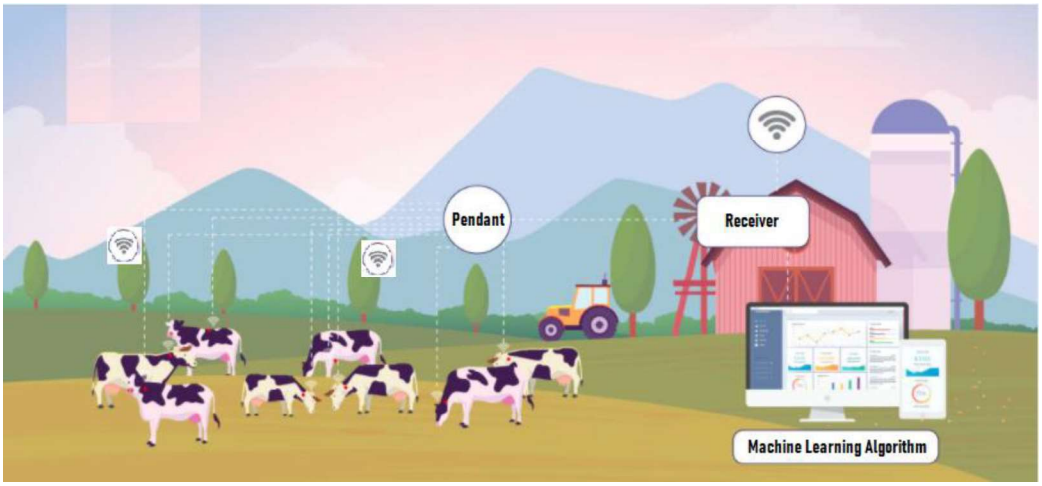

Figure 10: Cattle tracking on a farm

3.4 Software design decisions

This software design presents specific software tools to enable successful communication between this project's hardware components and the machine learning algorithm. The software design's primary purpose is to ensure affordability, reliability, accuracy, and security in cattle tracking.

3.4.1 Software tools for machine learning

3.4.1.1 Anaconda

Anaconda is a python distribution platform. In this project, Anaconda is used to enable offline access of Jupyter notebooks to write python scripts for the machine learning classification method used in this project. A Jupyter notebook is an open-web application that allows a person to create documents that contain live code, equations, visualizations, and narrative text. It is a platform that is mainly used for machine learning and data science.

3.4.1.2 Machine Learning algorithms

Supervised learning will be used, and Python will write the code that enables the classifiers under supervised learning. One of those classifiers is decision tree learning. In

decision tree learning, data is continuously split according to a specific parameter. Figure 11 below shows an example of how a decision tree classifier works.

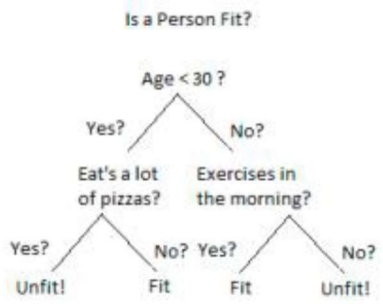

Figure 11: Example of a decision tree

3.4.1.3 Embedded systems software

The Arduino Integrated Development Environment permits the programming of the ESP8266 Node MCU used in this project. The script to control how the data acquired through the ESP8266 Node MCU will be used is written in C/C++. Through Arduino IDE, the project can acquire RSSIs and SSIDs, which can then be used to classify locations according to the RSS and SSIDs' variance as they differ from place to place. Their variation is an important feature that enables proper tracking as the machine learning algorithm will predict the specific location that the cow is.

Chapter 4: Implementation

To achieve the set goals, there needs to be a sufficient amount of data containing RSSIs alongside SSIDs to train the machine learning algorithm, which will predict the location of cattle. Therefore, this chapter presents the hardware and software used to gather the necessary data and perform the predictions.

4.1 Hardware Implementation

The hardware implementation for this project is in two parts. The first one is the hardware representing the pendant on the cow. The second one is the hardware representing the receiving end, whereby monitoring will be done on a computer screen.

4.1.1 Pendant Hardware

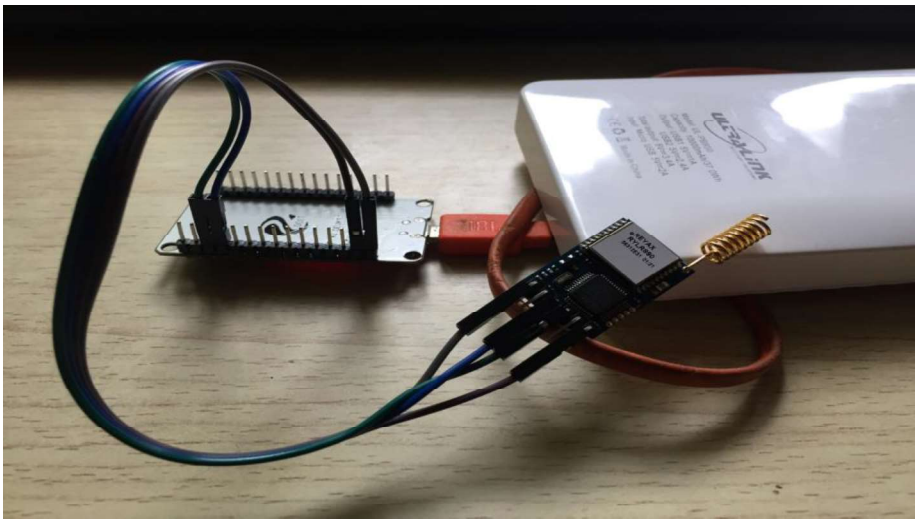

Figure 12: Pendant hardware connection

The pendant hardware in figure 12 consists of an ESP8266 Node Microcontroller Unit (Node MCU) and a Reyax RYLR896 Lora transceiver. The Node MCU acts as a scanner of nearby WiFi access points. The microcontroller unit also consists of the machine learning program that is used to predict the location. The Lora transceiver transmits the results wireless to the receiver – where the cattle farmer will be monitoring.

It is set up using AT commands to permit the transmission of data. Appendix B goes into detail on the specific AT commands and demonstrates the sending of data. The power bank represents a power supply that powers the Node MCU. From the Node MCU, 3.3V is tapped to power up the Lora transceiver.

4.1.1 Pendant Hardware

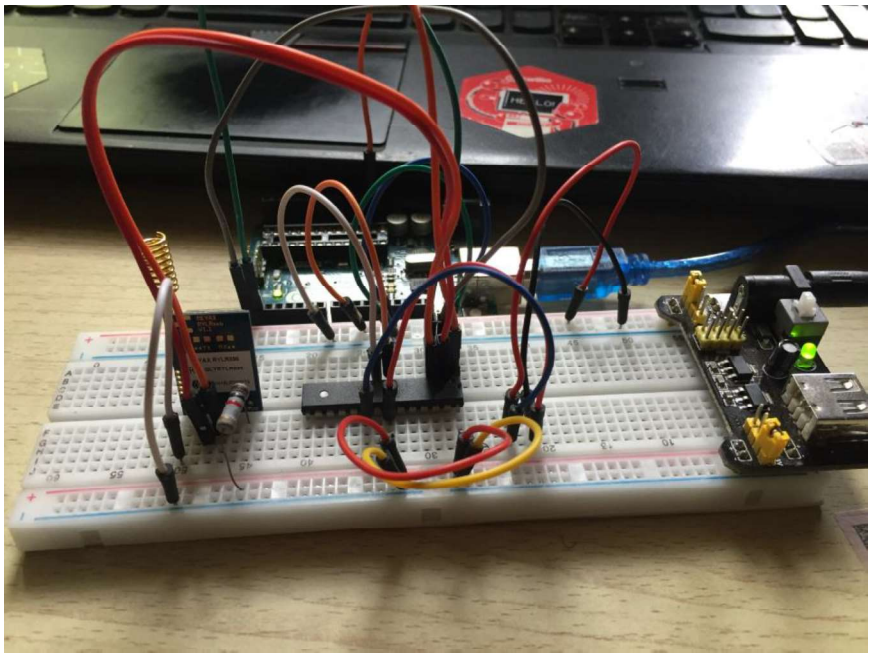

Figure 13: Receiver hardware connection

The hardware connection shown in figure 13 is the receiver at the monitoring side, where the farmer will receive the cattle being tracked. An Atmega 328P U was connected to the Reyax RYLR896 transceiver using the R.X. and TX pins. The Atmega chip contains the code that permits the receiving of the sent location predictions from the pendant. When the location predictions are received, the Lora transceiver adds the received signal strength indicators. These RSSIs can better the location prediction and calculate the distance to know how far the cow is. These results are printing on the Serial Monitor, where the farmer will be monitoring the location.

4.2 Software Implementation

An ESP8266 Node MCU was to be used as a scanner of nearby WiFi access points to enable the proper data collection. To enable such functionality of the ESP8266 Node MCU, appropriate code had to be written in Arduino IDE.

4.2.1 Code Preparation

Specific libraries had to be installed to enable the Arduino IDE to interface with the ESP8266 Node MCU. The libraries installed included the ESP8266 WiFi.h library. For successful programming of the ESP8266 Node MCU, the right board had to be installed. Therefore, before coding, ESP8266 Boards (2.7.4) was installed via Board Manager on Tools in Arduino IDE.

To enable scanning without connection to WiFi, the ESP8266 Node MCU had to be disconnected from any WiFi access point it had previously connected to. The upload speed for the ESP8266 Node MCU is 115200, and it was essential to specify the correct upload speed.

```
void setup(){

  Serial.begin(115200);
  delay(3000);
  WiFi.disconnect();
  }
```

Figure 14: Disconnecting from previously connect WiFis

The code collecting data, which is attached as an appendix, was prepared to work in the following steps:

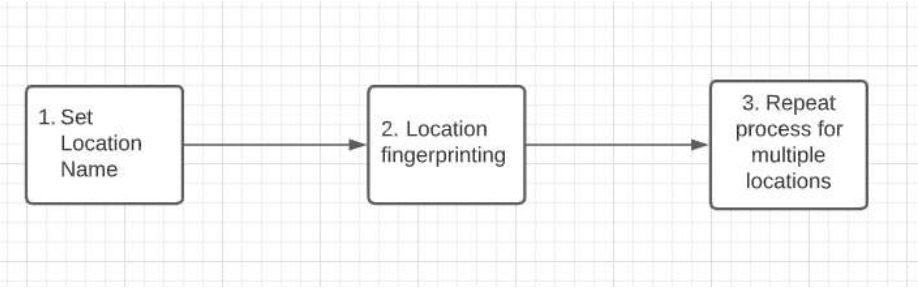

Figure 15: Steps for fingerprinting locations

1. Set the location name to where the data is being collected. The name of the location can be a prominent and specific feature in the farm. A good example would be a boulder or a tree name. To set the location name, the program listens for the command "Scan," followed by the location name inserted on the serial monitor.

   ```
   >> Print "Enter' scan location' to start scanning"
   >> Scan New Jacaranda Tree by White Boulder.
   ```

   The location's name automatically becomes New Jacaranda Tree by White Boulder; all the RSSIs and SSIDs collected on that location will be classified using the location name. After the command "Scan" with the location name has been executed, the scan begins.

2. Scan the location and store the data with the right SSIDs assigned to their RSSIs before printing them to the Serial monitor.

   ```
   Print '{"__location"'
   Print ':'
   Print "locationame"
   For(int i = 0; i < numberofNetworks; i++){
   Print ","
   Print SSID
   Print ":"
   ```

```
Print RSSI
Print (i == numberofNetworks – 1? "}\n")}
```

The above pseudocode prints the results as like, `{"__location": New Jacaranda by White Boulder, Network One Name: RSSI for Network One, Network Two Name: RSSI for Network Two}.`

The acquired information printed on the Serial monitor was then copied and inserted in the machine learning classifier.

The above procedure was redone at different locations that cattle can go. Each acquired information being put in the machine learning classifier.

4.2.2 Classifier Preparation

Identifying the appropriate classifier to use to achieve high accuracy was the main objective. Machine Learning offers many classifiers, including perceptron, naïve Bayes, decision tree, logistic regression, K-Nearest neighbor, artificial neural networks/deep learning, and support vector machine. Since labeled data is fed to the machine learning program during training, the type of learning used is supervised learning. Under supervised learning, the options are decision tree, regression, naïve Bayes, random forest, and support vector machines (SVM). This project used a decision tree as it is the most straightforward and lightweight classifier. The project also experimented on the support vector machine.

Python was used to prepare the decision tree classifier. The appropriate library was installed to enable the classifier's interfacing and acquiring Arduino program written earlier. Micromlgen was the best library to use. Micromlgen, often called MicroML, is a library by Eloquent Arduino that enables machine learning algorithms to be used in microcontrollers. The microcontroller used in this project is an ESP8266 Node MCU. The

functionality that MicroML made available was the port_wifi_indoor_positioning. This functionality allows for converting the collected data samples into a C code that can be used to compare future acquired data with default data.

Figure 16 below shows the conversion of location fingerprinting data samples into a C code embedded in the Node MCU. The generated C code contains conditional statements that describe the locations according to their SSIDs and their RSSIs.

```
from micromlgen import port_wifi_indoor_positioning


if __name__ == '__main__':
    samples ='''
    {"__location": "New Jacaranda Tree by White Boulder ", "WiFi Name 1": 50, "WiFi Name 2": 80}
    {"__location": "Peach Tree by the Gate", "WiFi Name 3": 30, "WiFi Name 2": 70}
    {"__location": "Cattle Kraal", "WiFi Name 1": 100, "WiFi Name 4": 50}
    {"__location": "Biggest Guava Tree by the River", "WiFi Name 5": 100, "WiFi Name 4": 60}
    '''

    X,y, classmap, converter_code = port_wifi_indoor_positioning(samples)
    print(converter_code)
```

Figure 16: MicroML's port_wifi_indoor_positioning functionality in Python

The classier preparation's important and last step was specifying what kind of classifier the project would use. The code from figure 16 was maintained, and the only lines added to it to complete the classifier preparation stage was as follows:

```
clf = DecisionTreeClassifier ()

clf.fit(X,y)
```

These two lines above create the classifier object and then use it to fit or find the relationship between the input X and the output y. X would be the new scans, and y would be the already stored, earlier acquired data. Micromlgen also offers the support vector machine as a classifier; hence the classifier was the main classifier explored by this project, and its accuracy was discussed. To use the SVM classifier, the libraries imported in addition to those in figure 16 were the scikit-learn and micromlgen. The line of code added to figure 16's option of a classifier was as follows:

```
clf = SVC(kernel = 'linear', gamma = 0.001).fit(X,y)
```

Setting the kernel to be linear meant that the mathematical function used was linear – meaning that a straight line referred to as a hyperplane was used to separate and classify data (RSSIs and SSIDs) points. Refer to figure 17. Nonlinear functions are also available in the support vector machine. For a linear kernel, the equation for prediction for a new input between the input (x) and each support vector (xi) is calculated as follows:

f(x) = B(0) + sum (ai * (x,xi))

It involves calculating the inner products of a new input vector (matrix formulated through a scan made) with the support vectors in the training data. The training data were the SSIDs and RSSIs that were used to train the algorithm. The coefficients B0 and ai for each input are estimated from the training data by the algorithm.

The gamma parameter defines how far the influence of a single training example reaches. A low gamma means that points far away from the separation lines are considered in calculating the separation line and the classification. This makes the support vector machine classifier more accurate hence the choice of a very low gamma value.

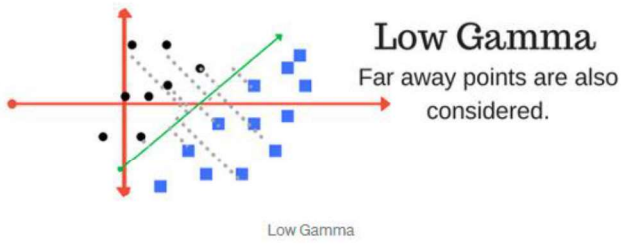

Figure 17: Linear hyperplane with a low gamma

4.3 Setting up Access Points

To help set up access points in farms that do not have nearby WiFi access points, this project also introduced an approach of using Node MCU as access points. Arduino

IDE was used to program an ESP8266 Node MCU, converting it from a regular WiFi device to an access point.

A call is made to the method WiFi to set up an ESP8266 Node MCU as a WiFi access point.softAP("SSID", "password"), passing in the appropriate parameter for AP SSID password as strings. These Node MCU Access Points can be deployed in the field, as demonstrated in figure 18.

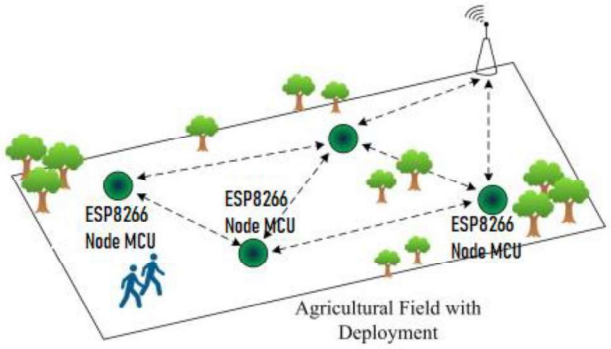

Figure 18: Deployment of Access Points

Chapter 5: Testing & Results

This chapter presents the type of tests done to evaluate the feasibility and the reliability of the solution proposed by this project.

5.1 Accuracy testing using decision tree

Six locations in total were used to test the machine learning algorithm's accuracy for this project. The locations were named Potatoes Field, Sorghum Field, Gum Tree, Cattle Kraal, Cassava Field, and Sweet Potatoes Field. These are typical examples of positions that a cattle farmer can use as physical features to identify a cow's position on the farm. At most, testing was repeated three times in one location. Every time a test was done, the data was doubled to monitor how the algorithm behaves and how best to increase accuracy. Figure 19 below shows a step-by-step layout of how testing was done.

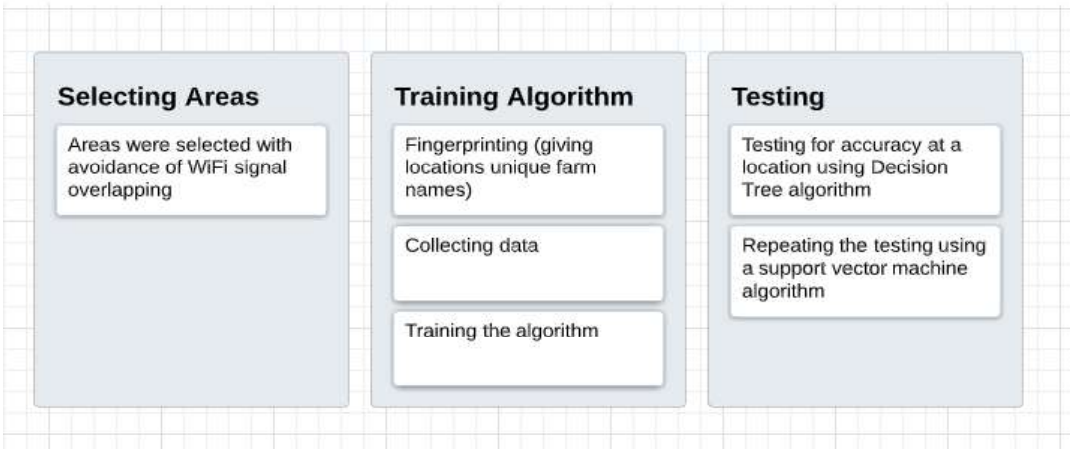

Figure 19: Step by step process of how testing was done

Testing was done on campus at Ashesi University. The receiver was set stationed at Lab 221. Lab 221 was given the name (fingerprinted as) Potatoes Field. Five other locations were selected: the Dean of Students Office, Lecture Hall 216, E.E. 201-W3, Design Lab, and Hakuna Matata. These locations were then fingerprinted as Sweet Potatoes Field, Cassava Field, Cattle Kraal, Gum Tree, and Sorghum Field, respectively.

Places adjacent to each other were not picked; this was to avoid an overlap in WiFi signals. An overlap in WiFi signals would make the classification difficult as the received SSIDs and RSSIs served as the collected data would be almost identical. During fingerprinting, the Node MCU embedded with a program that printed all the SSIDs and RSSIs available at a location was moved from one place to another. Every time a scan was done, the information was stored in a vector format inside the classifier code. The stored data was then used to train the algorithm before making predictions.

During testing, the receiver was kept stationary while the transmitter was moved from one location to another. This was to simulate tracking as the project is focusing on the tracking of cattle. Upon completing a prediction at a location, the transmitter sent the results to the receiver using Lora. The predictions were then printed in the Serial Monitor in Arduino IDE.

To calculate accuracy, the following formula was used:

$$\text{Accuracy} = \frac{Number\ of\ Correct\ Predictions}{Total\ number\ of\ predictions}$$

Correct predictions meant areas that the machine learning algorithm was able to identify correctly during the testing stage. For instance, if the area was fingerprinted as Cassava Field, the machine learning program can identify the same area as Cassava Field during tracking or testing, making it a correct prediction. This is referred to as true positives in machine learning.

Table 1: Test 1

Location	No. of Samples	No. of Predictions	No. of correct predictions	Accuracy
Potatoes Field	15	42	37	0.88
Gum Tree	15	51	44	0.862745
Sweet Potatoes	15	74	71	0.959459

Field				
Cattle Kraal	15	100	64	0.64
Sorghum Field	15	77	69	0.8961039
Cassava Field	15	163	121	0.742331

During test 1, demonstrated in Table 1, the number of data samples used to train the algorithm was kept constant – 15 samples were used for all locations. Samples were the number of rows of data taken from one place and fed to the program. The accuracy of machine learning was observed for all six areas. The test was on how much data should be supplied to the algorithm to ensure improved accuracy in tracking. It mainly was to monitor the correlation between the number of data samples and the algorithm's accuracy.

Figure 20 shows an example of samples provided to the machine learning program from Sorghum Field.


```
{"_location": "Sorghum Field", "Sula's iPhone": -18, "ASHESI-GUEST": -68, "AshesiAir": -67}
{"_location": "Sorghum Field", "Sula's iPhone": -20, "ASHESI-GUEST": -68, "AshesiAir": -69}
{"_location": "Sorghum Field", "Sula's iPhone": -20, "ASHESI-GUEST": -66, "AshesiAir": -67}
{"_location": "Sorghum Field", "Sula's iPhone": -25, "ASHESI-GUEST": -65, "AshesiAir": -67}
{"_location": "Sorghum Field", "Sula's iPhone": -25, "ASHESI-GUEST": -67, "AshesiAir": -66}
{"_location": "Sorghum Field", "Sula's iPhone": -23, "ASHESI-GUEST": -89, "ASHESI-GUEST": -67, "AshesiAir": -67}
{"_location": "Sorghum Field", "Sula's iPhone": -22, "ASHESI-GUEST": -68, "AshesiAir": -67}
{"_location": "Sorghum Field", "Sula's iPhone": -23, "ASHESI-GUEST": -67, "AshesiAir": -67}
{"_location": "Sorghum Field", "Sula's iPhone": -23, "AshesiAir": -87, "ASHESI-GUEST": -68, "AshesiAir": -66}
{"_location": "Sorghum Field", "Sula's iPhone": -25, "ASHESI-GUEST": -67, "AshesiAir": -67}
{"_location": "Sorghum Field", "Sula's iPhone": -21, "ASHESI-GUEST": -66, "AshesiAir": -66}
{"_location": "Sorghum Field", "Sula's iPhone": -21, "AshesiAir": -90, "ASHESI-GUEST": -67, "AshesiAir": -66}
{"_location": "Sorghum Field", "Sula's iPhone": -43, "ASHESI-GUEST": -87, "ASHESI-GUEST": -68, "AshesiAir": -68}
{"_location": "Sorghum Field", "Sula's iPhone": -22, "ASHESI-GUEST": -88, "AshesiAir": -88, "ASHESI-GUEST": -67}
{"_location": "Sorghum Field", "Sula's iPhone": -43, "ASHESI-GUEST": -76, "AshesiAir": -76, "AshesiAir": -86}
```


Figure 20: 15 samples from Sorghum Field

For test 2, shown below in table 2, the number of samples was double of those used in test 1. Thirty samples were used for test 2, and accuracy was calculated, and the results were recorded in Table 2 for all six locations.

Table 2: Test 2

Location	No. of Samples	No. of Predictions	No. of correct predictions	Accuracy
Potatoes Field	30	92	88	0.956522
Gum Tree	30	48	45	0.953333

Sweet Potatoes Field	30	91	91	1
Cattle Kraal	30	85	84	0.988235
Sorghum Field	30	195	195	1
Cassava Field	30	196	173	0.882653

Test 1 and Test 2 hinted that to increase the program's accuracy, a unique access point had to be installed in the location. This access point acted as the unique identifier of the area. This led to a third test being conducted, with a unique access point being made available. Therefore, Potatoes Field was put to the third test with 60 samples being supplied to the program and a new access point introduced using ESP8266 NodeMCU. Forty-four predictions were made, with all of them being correct, an accuracy of 1 was achieved.

A graph of the outcomes showed the correlation between the number of data samples supplied to the program and the program's accuracy. Potatoes Field data and prediction outcomes were used to visualize the correlation. Refer to Figure 21 below.

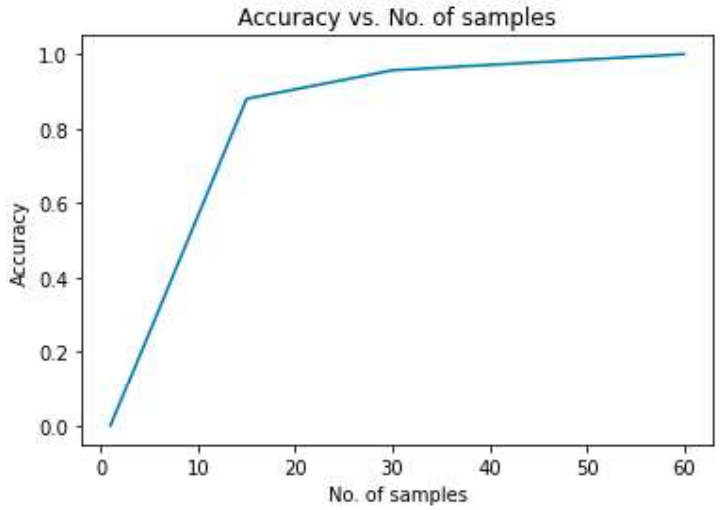

Figure 21: Accuracy vs. No. of samples

5.2 Accuracy testing using SVM

Micromlgen provides other machine learning classifiers, one of which is the supervised vector machine (SVM). SVM, like the decision tree, is another supervised learning model which can be used. This test aimed to compare the decision tree model with the SVM model to see best which classifier can provide higher accuracy. The area named Cassava Field was used for testing – this is Lecture Hall 216 in Ashesi. The receiver was stationed at Lab 221. A scan was performed, and a total number of two hundred predictions were done using the decision tree. Every time a scan was done, a prediction was made. The results were then sent using Lora to the receiver in Lab 221. The same procedure was repeated except that now the support vector machine was used to make predictions, and the results were sent to Lab 221 using Lora. Accuracy was then calculated using the formula:

$$\text{Accuracy} = \frac{Number\ of\ Correct\ Predictions}{Total\ number\ of\ predictions}$$

The support vector machine yielded an accuracy of 0.877193, while the decision tree yielded an accuracy of 0.81372. The support vector machine takes longer to predict, while the decision tree took less time. The code to both classifiers is presented on the appendix page.

The support vector machine algorithm was then tested for consistency with accuracy. In testing for consistency, the receiver was positioned at Lab 221, labeled as Potatoes Field. The transmitter was then moved to three places: Hakuna Matata, Lecture Hall 216, Dean of Students' Office labeled as Sorghum Field, Cassava Field, and Sweet Potatoes Field, respectively. A scan was done one place at a time. The algorithm was designed so that a prediction using a support vector machine classifier is made every time

a scan is done. The prediction results were sent to the receiver and then printed on the Serial Monitor of the Arduino IDE of the personal computer stationed in Lab 221. The number of predictions was doubled every after a set of predictions. A total of four prediction sets were done for every location. For example, in figure 22, the first set of Esibayeni predictions were 83; they then got double to 166 predictions, then 249 predictions, eventually 334 predictions. This was done to monitor how the algorithm performed when the number of predictions was increased. The accuracy was calculated and documented with every increase in predictions.

Cattle Kraal	Number of Total Predictions	Number of Correct Predictions	Accuracy
	83	39	0.963855
	166	77	0.975904
	249	116	0.97992
	334	158	0.97005988
Sorghum Field	Number of Total Predictions	Number of Correct Predictions	Accuracy
	40	39	0.975
	80	77	0.9625
	120	116	0.966667
	162	158	0.9753009
Cassava Field	Number of Total Predictions	Number of Correct Predictions	Accuracy
	70	60	0.857143
	140	113	0.807143
	210	165	0.785714
	281	233	0.829181

Figure 22: Testing for consistency using SVM

Visualization of the performance of the support vector machine was made simpler using a plot of the number of predictions versus the accuracy. Figure 23 shows the

consistency of the support vector machine over an increase in the number of predictions. The flat-top demonstrates consistency.

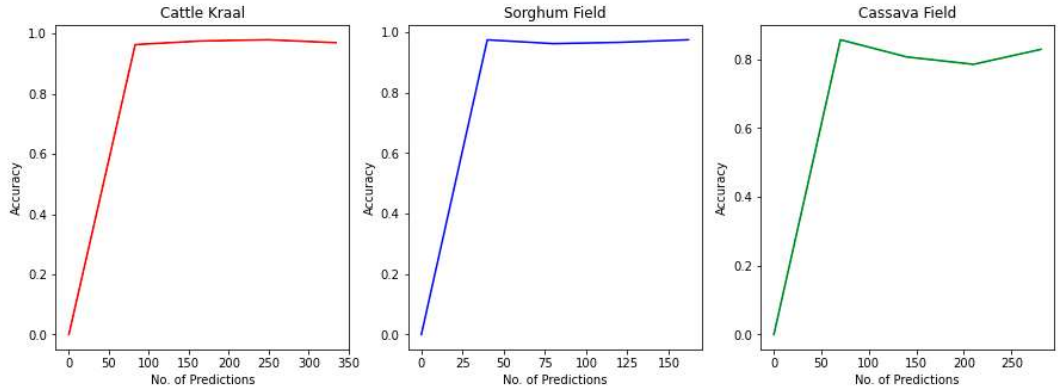

Figure 23: Consistency of SVM

5.3 Control experiment

This control experiment explores the use of a GPS module. GPS coordinates were sent to the receiver using Lora; they were saved in a MySQL database using a PHP script. Python was used to retrieve the coordinates and plot them on an ipyleaflet map. These coordinates were retrieved from Ashesi University, and the movement tracked from the Dean of Students' office to lecture hall 218 and back, as shown in figure 24.

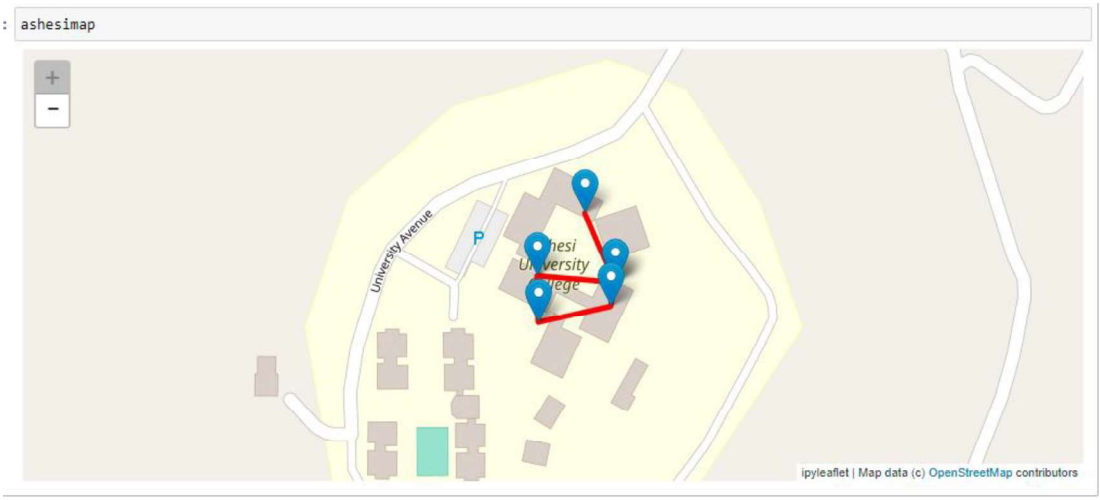

Figure 24: Tracking using GPS

The use of GPS required a steady internet connection for real-time location predictions. Even though the database used to store the GPS coordinates was a local one, some of the libraries used for this approach required internet access. These libraries were mainly the ipyleaflet and the geocoders. The intended use of ipyleaflet, which allowed offline tracking per user requirements, proved to be impossible. This meant that using an ipyleaflet map was almost the same as using a Google Map in terms of internet access. The GPS module used showed less response inside buildings. It could not locate and connect to satellites that would permit tracking. This was observed as it was moved to the far end of the Dean of Students' office, closer to the ODIP's office. This was a location with no clear line of sight. Unlike Google Maps, ipyleaflet maps did not give a detailed view of the path taken. Google maps could identify the precise locations with their names; for instance, the Cornfield and Archer Courtyard could be located on Google Maps.

Chapter 6: Conclusion

This chapter discusses the results obtained during testing and what can be learned from the tests conducted. Limitations encountered in this project will be addressed, the adjustments that can be made in future work, and the future plans of this project.

6.1 Discussion

The constant demand for a more objective approach to tracking and localization means that each method proposed needs to be thoroughly examined and its application justified. This project looked at the application of machine learning in IoT projects like cattle tracking on a farm. It seeks to reduce cattle theft. Received signal strength indicators from home-available WiFis were utilized to fingerprint locations on the farm. The location meant proximity to a combination of access points. If WiFis were not available, ESP866 Node MCUs were used to deploy access points whose received signal strength indicator could be mapped to the location where it was deployed. These ESP8266 Node MCUs were also given SSIDs that could easily be used for classification using the SSIDs as labels in the program.

Two classifiers were explored, the support vector machine (SVM) and the Decision Tree. The decision tree used the varying RSSIs and SSIDs from nearby access points to create conditional statements that were mapped to the location at which the scan was done. By giving the site a name, the area became fingerprinted or labeled as the name given, allowing the decision tree classifier to recognize the place using the SSIDs and RSSIs in that area. The support vector machine classifier used the varying RSSIs, and SSIDs scanned at a location to create a matrix representation of the data called support vectors using kernels. Kernels are mathematical functions that take data as an input and transform it into a model (required form) to predict the area in the future. Below is an

example of a linear kernel mathematical function that shows the relationship between the input (x) and each support vector (xi):

f(x) = B(0) + sum (ai * (x,xi))

The coefficients B(0) and ai for each input are estimated from the training data by the algorithm

The kernel used in this SVM to create the support vectors was linear. This means that linear hyper-planes were used to separate data during the classification. Figure 25 shows a visual representation of how classification is done using a linear kernel in an SVM classifier.

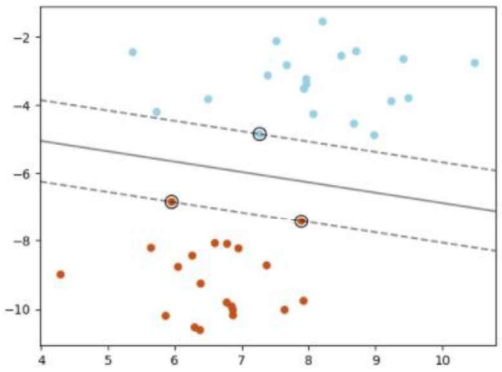

Figure 25: Linear hyperplanes used to separate and classify data

The support vector machine uses a mathematical and numerical approach to map areas to RSSIs and SSIDs instead of conditional statements. Hence the SVM classifier tends to be more precise in its predictions. Three tests were done using the Decision Tree. On average, at least three access points were detected in one place. The first test demonstrated in Table 1 under the results section showed a mean accuracy of 0.83010648.

This was a fair but unsatisfactory result. Hence there was a need for a second test. During the first test, it was discovered that noise played a pivotal role in the results

acquired. The noise came in the form of multiple access points, which were constantly changing. These made them not to be trustable identifiers of the location at which the tests were being made. Access points that amounted to the nose were WiFi hotspots from nearby mobile devices. Therefore, there was a need for constant and unique access points to be deployed. ESP8266 Node MCUs served as unique and permanent identifiers of the locations at which the data was acquired. Even though this was a problem in a busy place, such a setup on a farm could yield better results than those obtained in test 2. The second test with unique and permanent access points yielded a mean accuracy of 0.9635667. This was a very satisfactory result, and it proved the high accuracy that the program could achieve at low costs. Considering the average of the two accuracies achieved from Test 1 and Test 2, the average accuracy of the decision tree classifier was 0.896883659.

There was a need for both accuracy and reliability. Hence, the project further explored the use of a support vector machine algorithm. Testing on a new location with multiple noise sources was done to see how best SVM would work compared to the already then existing approach. The SVM classifier obtained a starting accuracy of 0.877193, while the Decision Tree obtained an accuracy of 0.81372, implying that the SVM classifier could be trusted. Reliability comes with a method that can maintain high accuracy throughout predictions. The SVM classifier was then tested for consistency. The test was done at three locations, and the number of predictions done increased at a fixed interval. The mean accuracy obtained from the three locations was 0.915699. This implied that every time this classifier was used for cattle location prediction, there is a confidence interval of more than 90%. Hence this a reliable solution.

Since prediction results were sent via Lora to the receiver, an observation was made on how Lora performed during the accuracy tests. Whenever a prediction was received, the Lora module attached an RSSI of its own to the result string, separating the

prediction and the received signal strength with a comma. Refer to Appendix B. A fluctuation in the RSSIs was observed. Hence a reliable and unique Lora RSSI for which the predictions were being sent could not be determined. This prevented a precise and linear calculation of the distance from where data packets were being sent to where the receiver was from being made. Instead, a range was used to create a conclusion of where the predictions were coming from. This left room for improvement in future works to incorporate distance calculations to improve cattle prediction further as then a farmer would be able to know how far the cattle are.

The control experiment revealed that the GPS module relied on a clear line of sight to make an accurate prediction. Under dense foliage, the module fails to connect to a GPS satellite, yielding no results. Hence, this limits GPS tracking for cattle tracking as cattle go under trees, forests, and shelters that might make GPS tracking challenging.

6.2 Limitations

This subsection presents the two main challenges that were faced when conducting this project. They are as follows:

- The Lora module used showed a fluctuating, inconsistent addition of RSSIs to the predictions sent. Hence it made it challenging to rely on that data to introduce calculations for distance to enhance tracking. The disturbance and fluctuation in the added Lora RSSIs are suspected of having been caused by the multiple numbers of buildings and the topography of the testing environment. It was not as flat as cattle farms are. This should not be a problem in farms but should be kept into consideration in future work.
- The solution provided through this project currently uses received signal strength indicators. A massive farm without nearby WiFi access points would require more

ESP8266, gradually increasing the costs. Therefore, there stands a challenge with the range that needs to be resolved through future work.

6.3 Future Work

To compensate for places where there is a need for more deployment of ESP8266 Node MCUs due to the lack of nearby WiFis, future developments would incorporate sensor nodes. These sensor nodes would provide the algorithm with data unique to that location, which can then train the algorithm. When conditions like the ones recorded are supplied to the algorithm, it will successfully predict the site. This sensor data can be sent using the same Lora technique used in this project, as Lora permits a distance over 10km.

The project was limited to tracking the location of cattle, but in the future could be used to alert drivers on cattle near a road. To enable such, the pendants that use this method for localization will be made to appear as hotspots. Hence, allowing their presence to be monitored over the phone through WiFi connections available nearby. This will help decrease accidents caused by cattle on the road. The primary concern with such functionality is security. Therefore, to better improve this solution without giving information on the whereabouts of the cattle to thieves, such an improvement will have to be carefully considered.

A more consistent Lora module would have to be selected in the future to make sure that the attached Lora RSSIs on the predictions sent as data packets are constant for a fixed location. This will then permit the solution to be enhanced with distance calculations which will make tracking more efficient. The Lora RSSIs would be used to calibrate measurements with an RSSI in dBm being mapped to actual distance measurement units (cm) to allow distance calculations.

Considering the need for such a solution in the real-life, the next step would be to create commercial prototypes of this technology and supplying it to cattle farmers to ensure that the intended impact of reducing cattle theft is achieved.

References

[1]Aziz, M. I., Owens, T., Khaleeq-uz-Zaman, U., & Akbar, M. B. "RSSI based localization of Bluetooth devices for visually impaired," Journal of Signal and Information Processing, vol. *10, no.* 2, pp 37-57, 2019. doi:10.4236/jsip.2019.102004

[2] Choi, W., Chang, Y., Jung, Y & Song, J. "Low-Power LoRa Signal-Based Outdoor Positioning Using Fingerprint Algorithm," International Journal of Geo-Information, 2018. Accessed on: Dec. 16, 2020. [Online]. Available: https://www.mdpi.com/2220-9964/7/11/440

[3] Danebjer, J., Halldorsson, V. "GPS-free geolocation for low-cost IoT devices," *LUP Student Papers*, 2018. Accessed on: Dec. 16, 2020. [Online]. Available:https://lup.lub.lu.se/student-papers/search/publication/8962876

[4] Fargas, B., C. & Petersen, M., N. "GPS – free Geolocation Using LoRa in Low-Power WANs," Proceedings of 2017 Global Internet of Things Summit, 2017. Accessed on: Dec. 16, 2020. [Online]. Available: https://orbit.dtu.dk/files/130478296/paper_final_2.pdf

[5]Laaraiedh, M., Avrillon, S., Amiot, N., & Uguen, B. "A semidefinite programming approach to Hybrid localization using RSSI and TOA," 2011 8th Workshop on Positioning, Navigation and Communication, 2011. Accessed on: Dec. 16, 2020. [Online]. Available: doi:10.1109/wpnc.2011.5961024

[6]Perkins, C., Lei, L., Kuhlman, M., Lee, T. H., Gateau, G., Bergbreiter, S., & Abshire,P. "Distance sensing for mini-robots: Rssi Vs. TDOA," 2011 IEEE International Symposium of Circuits and Systems (ISCAS), 2011. Accessed on: Dec. 16, 2020. [Online]. Available: doi:10.1109/iscas.2011.5937980

[7] Podevijn, N., Plets, D., Trogh J., Martens, L., Suanet P., Hendrikse, K., Wout, J. "TDoA-Based Outdoor Positioning with Tracking Algorithm in Public LoRa Network," Hindawi, 2018. Accessed on: Dec. 16, 2020. [Online]. Available: hindawi.com/journals/wcmc/2018/1864209

[8] Rus, C., Leba, M., Marcuş, R., Pellegrini, L. & Constandoiu, A. "LoRa communication and geolocation system for sensors network," MATEC Web Of Conferences, 2019. Accessed on: Dec. 16, 2020. [Online]. Available:https://www.matec-conferences.org/articles/matecconf/pdf/2020/01/matecconf_sesam20_00043.pdf

[9] Livestock theft is becoming more common in South Africa. Accessed on: Apr. 01, 2021. [Online]. Available: https://www.economist.com/middle-east-and-africa/2020/11/19/livestock-theft-is-becoming-more-common-in-south-africa

[10] Wikipedia contributors. "Long-range Wi-Fi," Wikipedia, 2021. Accessed on: Apr. 22, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Long-range_Wi-Fi#:%7E:text=Wi%2DFi%20networks%20have%20a,(160%20ft)%20or%20less.

[11] Security, R. "GPS Tracking Devices: A Brief History," Rewire Security, 2019. Accessed on: Apr. 22, 2021. [Online]. Available: https://www.rewiresecurity.co.uk/blog/gps-tracking-satellite-history

[12] Wikipedia contributors. "WiFi," Wikipedia, 2021. Access on: Apr. 22, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Wi-Fi

[13]Foote, K. D. "A Brief History of Machine Learning," DATAVERSITY, 2019. Accessed on: Apr. 22, 2021. [Online]. https://www.dataversity.net/a-brief-history-of-machine-learning/

Appendix A

The code down below is the python implementation of the support vector machine classifier. The code uses the Micromlgen library from Eloquent Arduino to convert the classifier into a C code. The C code is then saved as a header file that can be imported on Arduino IDE. The C code is saved as a model.h file.

```
from micromlgen import port_wifi_indoor_positioning

from sklearn.svm import SVC

from micromlgen import port

if __name__ == '__main__':

    samples = '''

{"__location": "Lab 221", "ASHESI-GUEST": -59, "AshesiAir": -58, "ASHESI-GUEST": -77, "AshesiAir": -76, "ASHESI-GUEST": -70, "AshesiAir": -69, "ASHESI-GUEST": -83, "AshesiAir": -86}

{"__location": "Lab 221", "ASHESI-GUEST": -59, "AshesiAir": -59, "ASHESI-GUEST": -77, "AshesiAir": -77, "ASHESI-GUEST": -69, "AshesiAir": -68, "AshesiAir": -86, "AshesiAir": -87}

{"__location": "Lab 221", "ASHESI-GUEST": -61, "AshesiAir": -59, "ASHESI-GUEST": -77, "AshesiAir": -74, "AshesiAir": -91, "ASHESI-GUEST": -86, "AshesiAir": -68}

{"__location": "Lab 221", "ASHESI-GUEST": -60, "AshesiAir": -60, "AshesiAir": -77, "ASHESI-GUEST": -69, "AshesiAir": -66, "ASHESI-GUEST": -85}


{"__location": "Esibayeni", "AshesiAir": -83, "ASHESI-GUEST": -85, "Morakane": -31, "ASHESI-GUEST": -91, "AshesiAir": -87}

{"__location": "Esibayeni", "ASHESI-GUEST": -87, "AshesiAir": -80, "Morakane": -30, "ASHESI-GUEST": -87, "AshesiAir": -87}

{"__location": "Esibayeni", "ASHESI-GUEST": -82, "AshesiAir": -82, "Morakane": -32, "ASHESI-GUEST": -87, "AshesiAir": -92}

{"__location": "Sorghum Field", "Sula’s iPhone": -18, "ASHESI-GUEST": -68, "AshesiAir": -67}
```

{"__location": "Sorghum Field", "Sula’s iPhone": -20, "ASHESI-GUEST": -68, "AshesiAir": -69}

{"__location": "Sorghum Field", "Sula’s iPhone": -20, "ASHESI-GUEST": -66, "AshesiAir": -67}

{"__location": "Sweet Potatoes Field", "Menzi": -30, "AshesiAir": -83, "ASHESI-GUEST": -82}

{"__location": "Sweet Potatoes Field", "PurpleAir-cb44": -88, "ASHESI-GUEST": -83, "Menzi": -29, "AshesiAir": -83, "Amen": -87}

{"__location": "Sweet Potatoes Field", "PurpleAir-cb44": -84, "ASHESI-GUEST": -84, "AshesiAir": -85}

{"__location": "Cassava Field", "PurpleAir-cb44": -67, "ASHESI-GUEST": -73, "AshesiAir": -74, "AshesiAir": -74, "ASHESI-GUEST": -74}

{"__location": "Cassava Field", "PurpleAir-cb44": -69, "ASHESI-GUEST": -71, "AshesiAir": -76, "Support Center": -88, "AshesiAir": -88, "Elijah’s iPhone": -91, "ASHESI-GUEST": -75, "AshesiAir": -74}

{"__location": "Cassava Field", "PurpleAir-cb44": -70, "ASHESI-GUEST": -72, "AshesiAir": -73, "ASHESI-GUEST": -74, "AshesiAir": -72}

{"__location": "Drinking Water Dam", "PurpleAir-cb44": -78, "LAPTOP-VPF503PB 3048": -90, "ASHESI-GUEST": -86, "AshesiAir": -86, "Support Center": -84, "ASHESI-GUEST": -87, "ASHESI-GUEST": -90, "ASHESI-GUEST": -90}

{"__location": "Drinking Water Dam", "LAPTOP-VPF503PB 3048": -87, "ASHESI-GUEST": -86, "ASHESI-GUEST": -81, "AshesiAir": -83, "AshesiAir": -87, "Support Center": -85, "ASHESI-GUEST": -87, "AshesiAir": -90, "PurpleAir-cb44": -78}

{"__location": "Drinking Water Dam", "PurpleAir-cb44": -76, "LAPTOP-VPF503PB 3048": -87, "ASHESI-GUEST": -80, "AshesiAir": -83, "Support Center": -90, "AshesiAir": -85, "ASHESI-GUEST": -90, "AshesiAir": -90}

{"__location": "Drinking Water Dam", "PurpleAir-cb44": -79, "ASHESI-GUEST": -81, "AshesiAir": -82, "AshesiAir": -87, "AshesiAir": -90}

{"__location": "Gum Tree", "ASHESI-GUEST": -53, "AshesiAir": -53, "ASHESI-GUEST": -75, "AshesiAir": -76, "Support Center": -87, "ASHESI-GUEST": -79, "AshesiAir": -79}

{"__location": "Gum Tree", "ASHESI-GUEST": -50, "AshesiAir": -52, "ASHESI-GUEST": -72, "AshesiAir": -71, "Support Center": -81, "ASHESI-GUEST": -80, "AshesiAir": -80}

```
{"__location": "Gum Tree", "ASHESI-GUEST": -52, "AshesiAir": -51, "Support Center": -80,
"ASHESI-GUEST": -72, "AshesiAir": -75, "AshesiAir": -79, "ASHESI-GUEST": -77, "ASHESI-
GUEST": -80, "AshesiAir": -81}

    '''

    X,y,classmap,converter_code=port_wifi_indoor_positioning(samples)

    clf = SVC(kernel = 'linear', gamma = 0.001).fit(X,y)

    print(port(clf))
```

The code down below is the implementation of the solution using the support vector machine on Arduino IDE. The support vector machine model gives the results in IDs or numbers hence the method idxToLabel converts those IDs into readable location names using a switch case.

```
#include <ESP8266WiFi.h>
#include <ConverterSVM.h>
#include <model.h>
Eloquent::Projects::WifiIndoorPositioning positioning;
Eloquent::ML::Port:: SVM classifier;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  WiFi.softAP("SulaNodeMCU", "76747773");
}
void loop() {
  // put your main code hereere, to run repeatedly:


  positioning.scan();
  Serial.print("Jamludi is in ");
  //Serial.println(classifier.predict(positioning.features)); //returns a number
  /*
  This will convert the numbers the model gives out to readable names; these can be updated
on the setup side.
  */
  Serial.println(idxToLabel(classifier.predict(positioning.features)));
  delay(3000);
}
const char* idxToLabel(uint8_t classIdx) {
                        switch (classIdx) {
```

```
                        case 0:
                        return "Cassava Field";
                        case 1:
                        return "Drinking Water Dam";
                        case 2:
                        return "Esibayeni";
                        case 3:
                        return "Gum Tree";
                        case 4:
                        return "Lab 221";
                        case 5:
                        return "Sorghum Field";
                        case 6:
                        return "Sweet Potatoes Field";
                        default:
                        return "We have a problem";
                    }
}
```

Below is a similar implementation on Arduino IDE of the Decision Tree. The predictLabel does the same task as the idxToLabel method implemented on the SVM.

```
Eloquent::Projects::WifiIndoorPositioning positioning;
Eloquent::ML::Port::DecisionTree classifier;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
   WiFi.softAP("SulaNodeMCU", "76747773"); //to allow drivers to be alerted if a cow is approaching the road
}
void loop() {
  // put your main code here, to run repeatedly:
  positioning.scan();
  Serial.print("Jamludi is in ");
  Serial.println(classifier.predictLabel(positioning.features));
  delay(3000);
}
```

Appendix B

The image below shows how the location that was predicted by the machine learning algorithm in the ESP8266 Node MCU was sent from the transmitter to the receiver. This was a continuous process. AT+SEND=0 is a command to send on the Reyax Lora module, the ADDRESS, set as 0 for this project. +RCV=0 means received from the address = 0. Addresses should be the same.

```
COM4
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
AT+SEND=0,7,Lab 221
```

Figure 26: Sending location predictions from lab 221

```
COM7
+RCV=0,7,Lab 221,-43,45
+RCV=0,7,Lab 221,-43,39
+RCV=0,7,Lab 221,-44,46
+RCV=0,7,Lab 221,-44,47
+RCV=0,7,Lab 221,-45,39
+RCV=0,7,Lab 221,-45,43
+RCV=0,7,Lab 221,-45,41
+RCV=0,7,Lab 221,-46,41
+RCV=0,7,Lab 221,-46,43
+RCV=0,7,Lab 221,-46,43
+RCV=0,7,Lab 221,-45,45
+RCV=0,7,Lab 221,-49,45
+RCV=0,7,Lab 221,-50,44
+RCV=0,7,Lab 221,-48,45
+RCV=0,7,Lab 221,-47,40
+RCV=0,7,Lab 221,-45,43
+RCV=0,7,Lab 221,-49,48
+RCV=0,7,Lab 221,-49,43
+RCV=0,7,Lab 221,-49,45
```

Figure 27: Receiving location predictions from lab 221