ASHESI UNIVERSITY

A NAVIGATION AID FOR VISUALLY IMPAIRED PEOPLE

CAPSTONE PROJECCT

B.Sc. Computer Engineering

Ayebilla Avoka

2020

ASHESI UNIVERSITY

A NAVIGATION AID FOR VISUALLY IMPAIRED PEOPLE

CAPSTONE PROJECT

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

Ayebilla Avoka

2020

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:

……

Candidate's Name:

Ayebilla Avoka

……

Date:

29/05/2020

……

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

Supervisor's Signature:

…….

Supervisor's Name:

……

Date:

……

Acknowledgments

To my supervisor, Dr. Stephen K. Armah whose encouragement and academic advice helped me undertake this project, I say a big thank you. I also want to thank Dr. Nathan N. Amanquah, acting Dean of Engineering, Ashesi and Dr. Ayorkor Korsah, Head of Computer Science Department, Ashesi, for their advice and guidance in my choice of courses every semester throughout my four years at Ashesi.

I would like to thank the MasterCard Foundation and the Scholarship Management team at Ashesi University for the financial support for all four years. To Aunty Araba, Director of Admissions and Financial Aid, Ashesi University, I say thank you. To my friend and brother, Daniel Atewin Amoshie, c2023, who offered to design a package for the final product of this project, I say thank you.

Finally, I thank God for seeing me through the hard times during these four years at Ashesi that creeped by

Abstract

The United Nation's 2018 report states that about 1.3 billion people suffer some form of visual impairment with 36 million of them being blind. The number of blind people is projected to reach 115 million by 2050**.** Some of these people find it difficult to move around because they have difficulty identifying their location, orientation and paths leading to their destinations in unknown territories. While on the road, they have difficulty detecting and avoiding obstacles along the road. The white cane and guide dogs have been popularly used by blind people to guide them along the road and help them avoid obstacles. However, white cane and dogs cannot communicate to the victim, his or her current location or tell him or her the kind of obstacle identified.

This report proposed a low-cost system that uses raspberry pi, camera, GPS receiver, ultrasonic and infrared sensors to help the user identify his or her current location, guide him or her through a path while detecting and avoiding obstacles. The system also names obstacles to the user to give him or her more perspective about the surrounding. In this report, only the location identification and path following are implemented and tested and the results showed a mix of true and false turns.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

This Chapter gives a general overview of the report, defines the problem, states the proposed solution, the motivation, and the objective of the report. It also defines the scope which outlines what this report does or does not include.

1.1. Background

According to the United Nations' 2018 report, 1.3 billion people globally suffer some form of visual impairment and 36 million are blind, with Africa and Asia having the highest percentage [1]. It is projected that the number of blind people will reach 115 million by 2050. These people bear the pain of not being able to walk around freely. They mostly rely on some form of support such as a trained dog, a white cane, or any of such things to help them move from one point to the other which is inconvenient in most cases. There are instances where a third person , mostly a child is dedicated to help the victim move. This interferes with the productivity or school attendance of the person dedicated to help the victim. Trained dogs and white canes solve this problem of having to dedicate a person to help but are inconvenient when it comes to interacting with the victim or being able to know where the victim wants to go. Another problem with the issue of using dog as a guide is that, in places like some parts of Ghana where dogs are not given such special training, it may be difficult to even find an expert dog-trainer to give a dog such special training to be of help to a blind person. This report seeks to develop an electronic aid that will help visually impaired people to navigate their environment in a more convenient and autonomous way as described in the solution.

1.2. Motivation

The pain of living with my blind brother, Francis, and seeing how he has to depend on the assistance of other people for his movements forced me to begin looking for better ways to assist him rather than rely on people who may not always be around him to help. With my exposure to technology and my interest in developing assistive technologies, I considered making a personal assistant for Francis so that he will not have to rely on other people for his movement. I drew inspiration from autonomous vehicles and robots and begun to explore how I can adapt the technologies behind their breakthrough to assist Francis. Apart from embarking on the journey as a learning process, I am particularly excited because of the impact this project will make in the life of my brother, Francis and others who suffer the same fate in society.

1.3. Problem Definition

Visually impaired people need a way to navigate their environment autonomously because, relying on third party assistance for their movement inconveniences them and limits their freedom of movement.

Visually impaired people do not have the same freedom of movement as compared to their sighted counterparts in society because their sense of sight is partially or completely lost. Having sight problems means one can only rely on non-visual information from the environment for his or her independent mobility [2]. Non-visual information may help but can never do as much as eyes can do for a person. In my opinion, aside the awkward experience, it is much riskier for a blind person to use his or her hands, legs and other body parts to try and feel his surrounding as a means of identifying objects to help him know his or her location and direction. This risk can be mitigated by using electronic devices such as smart phones equipped

with object detectors and GPS capabilities to collect information about the surrounding environment. However, smart phones as they are right now are not user-friendly for the blind aside other challenges like short battery life.

1.4. Objectives

1.4.1. Main Objective

The aim of this project is to develop a low cost, power efficient, portable GPS and camera assisted navigation system that will guide a blind person to move from point A to B safely.

1.4.2. Specific Objectives

The specific objectives of this report are:

1. Discuss the design of all parts of the project
2. Demonstrate the representation of the road network of the user's vicinity on appropriate graph
3. Write a program that is able to select the shortest path from the graph, given a source and destination
4. Develop an image recognition algorithm that can identify images with at least 90% accuracy

1.5. Proposed Solution

This report proposes a raspberry pi based, low-cost, blind navigation system that takes voice commands from the user and gives audio output that the user can rely on to move from point A to point B without kicking his legs against obstacles along the way. This system is

unique because it uses GPS for path-following and camera to capture images of obstacles along the path and identify such obstacles to give the user a better feel of his or her environment unlike similar projects which either use GPS or camera only. It also makes use of both ultrasonic(sonar) and infrared sensors for obstacle detection and avoidance unlike similar works that use one or the other. Another important feature of this system is that, it works offline – it does not need the internet or any data charges to function. All processing will be done locally on the raspberry pi.

1.6. Scope

This project serves as a proof of concept that this approach can make a more exciting and reliable solution to the blind navigation problem compared to others. Sensors and other components used are those used in a typical student project, not necessarily ones used in production environment. The focus of implementation is on the obstacle detection and avoidance, location identification, graphical representation of the road network and path-following. For the sake of time and challenges posed by the COVID-19, all the other features of the project are described but not implemented in this report.

Chapter 2: Literature Review and Related Work

This Chapter opens with a brief discussion on how technological contribution to improving the lives of people living with disabilities is important aside governmental and non-governmental policies designed to that effect. It wraps up with a discussion on some selected reports that presents solutions to the blind navigation problem and how they compare with the solution this report is presents.

2.1. Literature Review

Improving the lives of people living with disabilities in society has been a topmost priority of many organizations and governments around the world [3]. Policies on inclusivity and making public facilities such as buildings and parks accessible to people living with disabilities are very common in many countries. This is commendable however more still needs to be done in finding solutions that will make people living with disability more independent. Advances in technology offers a great opportunity for researchers and scientists to develop products and solutions that can truly make life better for the disable in society. A good example is the breakthrough in the design of prosthetic limbs that are now more lifelike which has made life more exciting for amputees than before [4].

Many individuals and companies have attempted to design products and systems to help a category of people living with disability — the visually impaired yet there are few commercially viable products at a cheaper price to help the visually impaired to navigate the environment. In my opinion, the one biggest achievement in helping the visually impaired is the invention of the tactile that helps the blind to read and write. The challenge remains developing a product or a system that will help visually impaired people to walk around

independently. The white cane which has been around for a while now has been of great benefit but still needs to be done [5]. One interesting technology that could be a of great help is the Google Maps Directions API feature which allows users to request for directions to unfamiliar places [6]. Having used this service many times as a sighted person shows that it lacks the ability to guide a user through a specific path. What it does best is to guide you along your chosen direction. That is, once you are moving towards the right direction, Google Maps leaves the rest of the journey into your hands.

2.2. Related Work

2.2.1. **Visible light communication and compensated geomagnetic sensing**

This report titled “Indoor navigation system for visually impaired people using visible light communication and compensated geomagnetic sensing” explores the use of visible lights installed in buildings to help visually impaired people know their current position, travel direction and distance to their destination, making use of the already installed geomagnetic sensor in smartphones [7]. As a Japan based solution, the researchers made use of the already existing Visible Light Communication System in Japan to design their solution. How it works is the user has to long- press a button on his or her smartphone to indicate that he or she wants to move. The smartphone requests for the position information from a cloud base positional information system. The user then enters his or her destination in the form of voice input and the system guides him or her to the destination by receiving visible light rays intermittently to re-calculate the distance remaining and position of the user.

One problem with this approach is that it relies heavily on already mapped obstacle information and does not consider the fact that new obstacles may be introduced in a room.

Since this project does not include any obstacle detection and avoidance in real-time, it may be dangerous for a blind person to use it in an unknown space. Also, such a solution will not work in individual homes that do not have such detailed obstacle mapping information.

2.2.2. **Indoor positioning and obstacle detection based on LSD-SLAM**

"Indoor positioning and obstacle detection for visually impaired navigation system based on LSD-SLAM" is about Simultaneous Localization and Mapping (SLAM) technology-based indoor navigation system that uses both self-positioning and environmental map to guide visually impaired people in indoor spaces [8]. The system provides route guidance to a destination and event and obstacle alerts while the user moves on the selected path. The implementation features the use of a previously mapped obstacle information (global map) of the area and a real-time obstacle map (local map) which is started when the user starts the system. The system progressively calculates the nearest point of interest and guides the user to that point and repeats this till the end.

The approach is similar to the implementation proposed in this report except that the focus of this report is outdoor based and uses route map instead of obstacle map. In this implementation, instead of keeping a global map of obstacles, it keeps a global map of the route (GPS coordinates) to be used as a reference route during navigation. This is because, obstacles may change positions or new ones may be introduced which renders the stored obstacle map useless. This report also proposes the use ultrasonic and infrared sensors for obstacle detection and avoidance during navigation.

2.2.3. **Assistive navigation for blind**

"A Low Cost Outdoor Assistive Navigation System for Blind People" is a simple low-cost solution proposed by Raj and Das which uses GPS receiver to get the current location information about the user and compare that to an already saved GPS coordinates information about the route [9]. The path information is saved in an external drive as a database and passed to the system as the reference path information. As the user moves along the path, he is given an audio output information as to whether he is on the right track or not by computing next neighbor coordinates.

The authors of this work did not include information an obstacle detection and avoidance which may be a challenge to the user. In today's systems, even just avoiding obstacles may not be an exciting solution without obstacle recognition to give the user a sense of what is it that is around him or her.

The implementation proposed in this report includes infrared and ultrasonic sensors for the purpose of obstacle detection and avoidance and a camera to take pictures of obstacles for identification. By this approach, the user gets a feel of the kind of object that is before him or her which gives him or her a better perception of the surrounding.

2.2.4. **Navigation System for blind - Third Eye**

Third Eye, a solution proposed by Koharwal, Awwad, and Vyakaranam which is a raspberry pi-based system that uses infrared, and sonar sensors for distance and orientation prediction [10]. It uses the pi camera with image recognition algorithms to identify obstacles and name them to the user. It uses three infrared and sonar sensors to have a 180-degree view of the path to make sure the user is able to avoid obstacles.

What is missing in their work is details on how the user is able to identify his or her current location without the use of GPS receiver. Their work does not say what happens when the user misses the road and is on a completely new road. Since the user's aim is not just to follow a road and avoid obstacles but to follow a desired road to a desired location, a solution without a way to check the user's location may be problematic.

This report proposes a system that uses all of the sensors used in their project and has also included a GPS receiver that is able to tell the current location of the user. For details on the architecture of Third Eye, the reader can refer to [10, fig. 8]

Chapter 3: Design Methodology

This Chapter discusses the design methodology of the project. It highlights the design requirements, component selection, schematic diagrams, flowcharts, system architecture, and the database design.

3.1. Requirements

- The following are the requirements of the project.
  - ✓ The system has to be friendly and easy to operate by the user
  - ✓ The system must identify correctly, the current location of the user
  - ✓ The system should be able to identify the shortest path to the user's destination in the case of multiple paths leading to the same place
  - ✓ The system must give audio output to the user
  - ✓ In the case of unexpected behavior or failures, the system shall raise alarm for the user's rescue
  - ✓ The total cost of the system must be no more than $500
  - ✓ The image recognition algorithm must identify objects correctly every time with at least 90% accuracy.
  - ✓ The system will operate on 9v battery
  - ✓ The total weight of the system must be less than 3kg

3.2. Component Selection

Component selection is based mainly on functionality, cost, ease of use and availability of a support community. Most of these decisions were made based on previous experience working with these components as well as reviewing similar projects like the ones in the Chapter

- Embedded computer/microcontroller: Table 3.1 shows the embedded computer/microcontroller selection process. Raspberry pi 3 B is chosen for this project. Raspberry is a System on Chip (SoC) – a small computer with much more extended capabilities than a bare microcontroller or development board. Its ability to interface with the pi camera, support for different programming languages and support for many sensors is what makes it the most preferred choice.
- Obstacle detection sensors: infrared and sonar sensors are used in couple because of the added advantages of using both but not one. Sonar sensor is affected by noise in the surrounding more than infrared sensor. The infrared sensor is affected in some cases by visible lights in the environment. Using both sensors in couple helps to deal with the limitations of one. Other sensors such as RADAR could be used for the same purpose but the long-term effect of exposing the user to stronger radio waves makes it undesirable.
- Raspberry pi camera: Raspberry pi camera module v2.1 is used in this project mainly to capture obstacles which will be analyzed by an image recognition algorithm so that the user may know the kind of obstacles before him or her. This pi camera was chosen because of its lower cost and high resolution.

GPS receiver module (GY-NE06V2) is used to receive GPS signals that provides the longitude and latitude information of the user at every point. GY-NEO6V2 module is chosen for its availability, accuracy of 2.5m, update frequency up to 5Hz which is enough to get location information without overwhelming the raspberry pi.

Table 3.1: Pugh chart

	Arduino	Weight	Raspberry pi	NXP	Black Beaglebone
Criteria					
Community support	0	4	0	-4	-4
Availability	0	3	0	-3	-3
Cost	0	5	-5	-5	-5
Architecture	0	3	+3	+3	+3
Compatibility	0	5	+5	+5	+5
Total score	0	5	+3	-4	-4

Figure 3.1 shows a block diagram showing how the various parts of the system work with each other. The sensors collect selected data from the environment under control of raspberry pi. The data is either stored or processed to give output to the user.

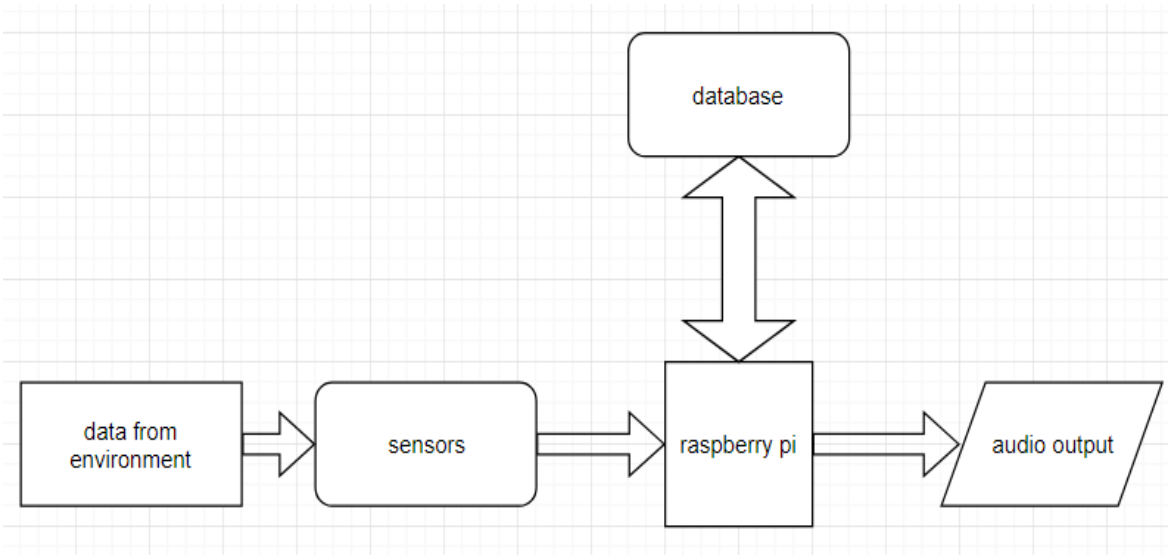


Figure 3.1: Functional block diagram

Figure 3.2 is a system architecture diagram that shows how all the different components interfaces with the raspberry pi module. The raspberry pi is the main controller in this project. All the sensors interface with it through either the GPIO pins, serial pins or the available USB ports on it.

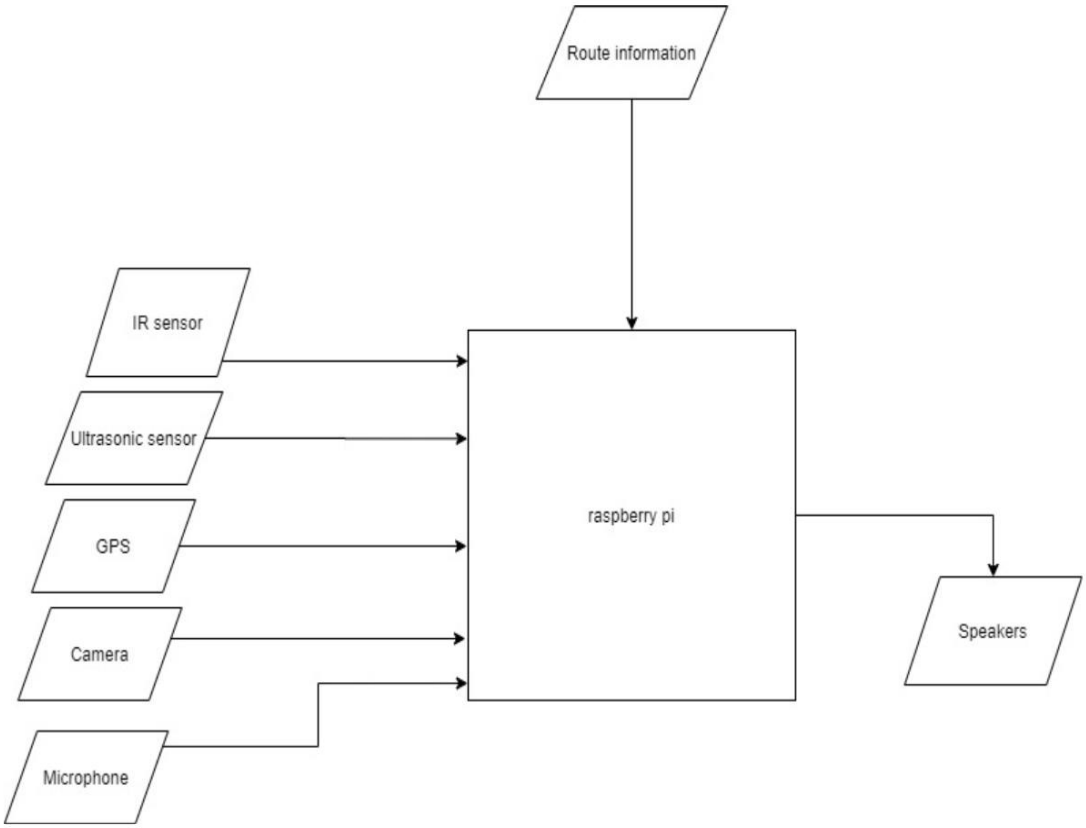


Figure 3.2: Architecture diagram

3.3. Description of System Operation

When the system is powered on, the GPS receiver module is turned on automatically to determine the coordinates of the user's current location. The raspberry pi controller does reverse geocoding to identify the name of the place. The controller then gives out the name of the place as output to the user through the headset connected to it. Now that the user knows where he is, it is time to move to where he wants to. The user mention's the name of the place he or she wants to go to. A speech recognition module which is always listening to the user as soon as the system is powered on process the speech from the user to identify a valid command in his or her speech. There are a finite set of commands that are known to the system. All destination

names are commands. If the user wants to go to a place that does not exist in the routes map, it is regarded as invalid command and the system will take no action. Once a valid command is identified in the user's speech, a voice recognition module will be executed to make sure the valid command is coming from the user but not a random person.

Once this security check is done to make sure the command is from the right user, the next step is to run the Dijkstra's Shortest Path algorithm to determine the shortest path to the destination in the case where there are many paths leading to the destination (see Figure 3.5 for a illustrative route map). Once the best path is determined, it is time to determine the orientation of the user to make sure he or she is facing the right direction. This is a crucial step that requires the use of magnetometer (magnetic compass) for accurate and quick determination of the direction.

However, for the sake of keeping the project simple, this is ignored. The only problem with this is that, the user may have to wander for a few minutes before getting the correct direction. The obstacle detection and avoidance sensors are started as well to make sure the user does not bump into obstacles. The GPS receives longitude and latitude coordinates in real time, and these are plotted on the map and compared with the previously stored route coordinates (reference coordinates). When the user deviates from the path by more than ±30 degrees, he or she is prompted to turn by the angle of deviation. A negative degree means the user has crossed to the left side of the road and so will be asked to turn right by the angle of deviation while positive means he or she is on the right hand side of the road and so will have to turn left by the angle of deviation.

While he or she moves, the ultrasonic and infrared sensors detect obstacles along the path. When an obstacle is detected, the camera module is started to capture the obstacle. An image

recognition algorithm is executed to identify the obstacle and the name of the obstacle is communicated to the user through the headphone. The user at this point does not only know that there is an obstacle before him or her, he or she also knows the kind of obstacle. This process of determining coordinates, capturing and identifying obstacles, and guiding the user along the pre-determined path continues until he or she reaches the destination where the final pair of real-time coordinates matches the destination address. Figure 3.3 below is a flow chart that summarizes the logical flow of the system as described.

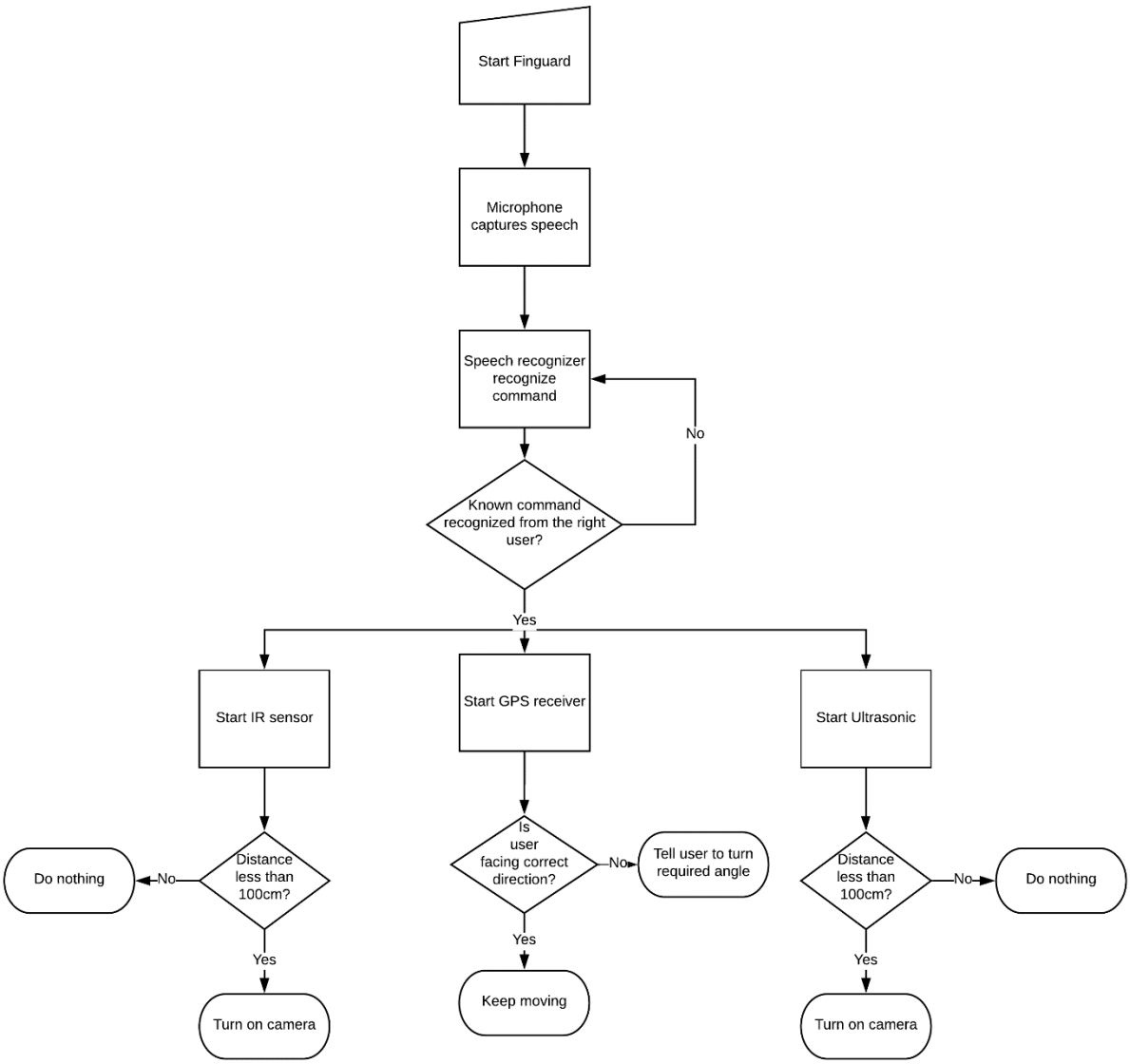


Figure 3.3: Design flowchart

3.4. Route Mapping

GPS coordinates information about routes within the reach of the user are collected first and stored in a database together with the distance between each pair source-destination. The source-destination pair is added to the graph forming the road network of within the vicinity of the user. As described in Section 3.3 above, once the user is ready to move from where he or she is (P1) to another place that exist on the route map (P2), the coordinates for that route is

loaded from the database and plotted into a path. As can be seen in the graph below, at every point in time, three points (P1, P2, P3) are needed to determine if the user is keeping along the path. These three points are made of two points from the reference coordinates (P1, P2) and the current coordinate point (P3) of the user. The two points from the reference coordinates are the previous point (P1) and the current point (P2) loaded. The previous point (P1) always serves as a common point for the two lines formed by connecting the previous and current point on the reference coordinates (P1 , P2) and the current point (P3) from the real-time GPS receiver and the previous point (P1) from the reference point. See Figure 3.4 below for this illustration.

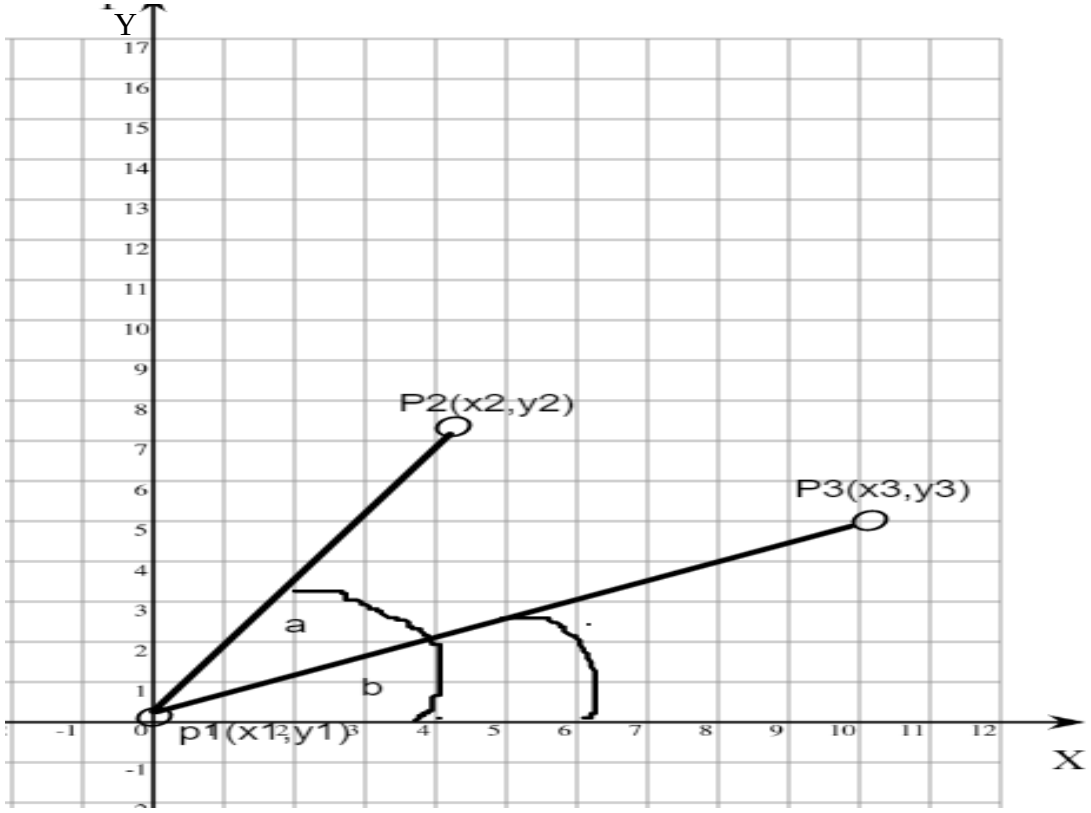


Figure 3.4: Reference angle graph

The desired path the user should be moving on is P1 to P2, but he is currently at P3 with a deviation of theta. He or she is asked to turn by that same angle to either left or right depending on the sign of the angle as discussed above. When the angle is 0, that means the user is moving on the reference path and there is no need to turn. Note that in some jurisdictions, the angle between two parallel lines is 0 or 180 degrees. In this implementation, the assumption is that the user will not be moving in a path parallel to the reference path. Figure 3.5 below is an illustration of a routes map. It shows that moving from point A to B may have different paths but the shortest among them is considered as discussed above.

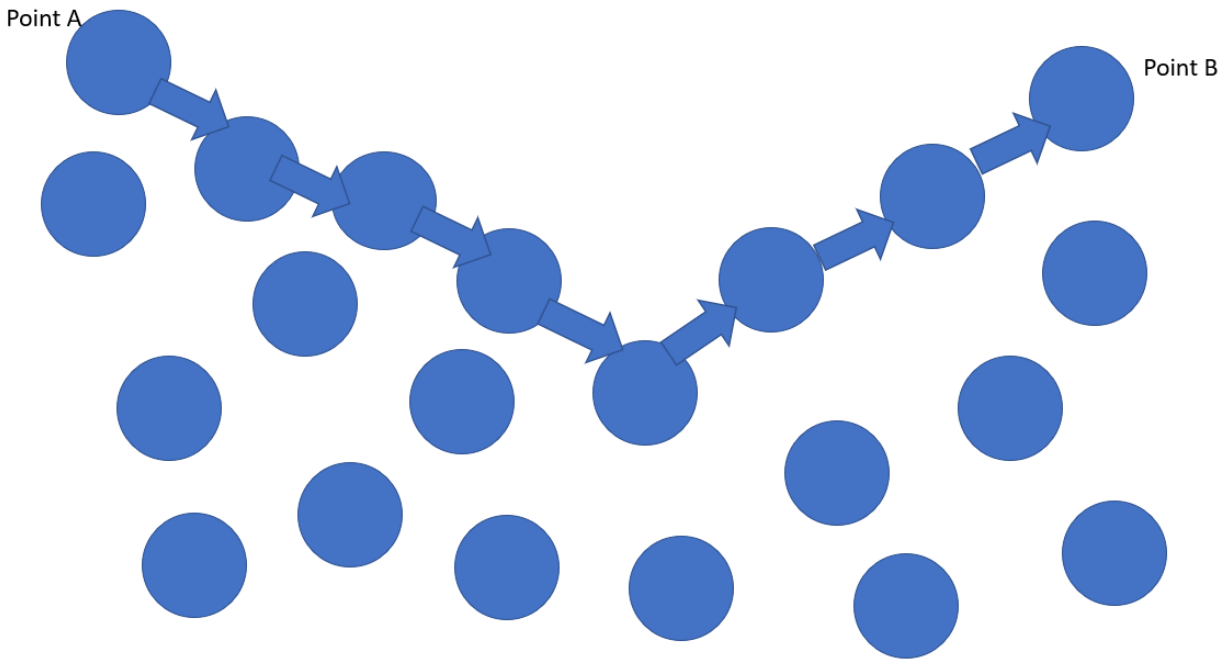


Figure 3.5: Example path

3.5. Path-following

Figure 3.6 shows the flow of activities in the path-following. The decisions that are taken at each step is shown in this flow chart.

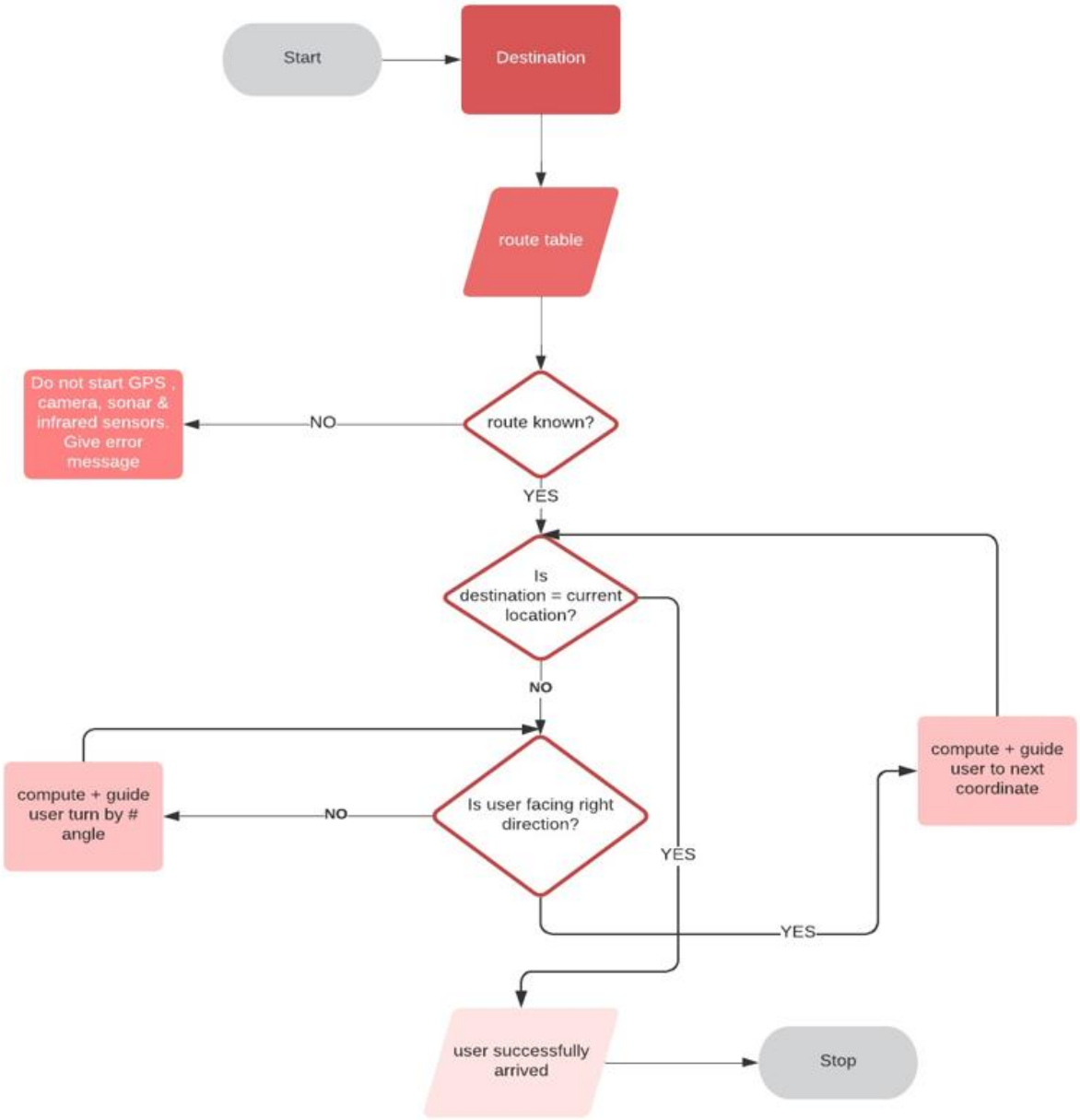


Figure 3.6: Path-following flowchart

3.6. Database Selection and Design

This project is designed to work entirely offline as stated in previous Chapters. This means that all the data relating to this system must be stored and processed locally within the raspberry pi computer. To conserve space for other important functions of the raspberry pi, the sqlite3 database which is built into the Python programming language used in this project is selected because of its light weight [11]. . Each route is stored as an entry on a routes table in the database as longitude, latitude and length which serves as the path id. To make loading paths easier, the length (id) of the path is repeated for every longitude-latitude pair but only the longitude latitude pair is loaded from the database during processing. The length just serves as a unique identifier to all coordinates belonging to the same route and makes loading such points together easier. Table 3.2 below is an example entry in the database table. It reflects two roads — one with a length of 80 from some starting point and the other with a distance of 120 from another point with their associated longitude latitude pairs. Given the path id, which is the length, all the coordinates of that path can be loaded row-wise into a Python list for processing.

Table 3.2: Sample database entry

Length (m)	Lattitude	Longitude
80	10.935947	-0.482683
80	10.936082	-0.482737
80	10.936125	-0.482755
80	10.936222	-0.482773
80	10.936318	-0.482797
80	10.936412	-0.482797
80	10.93651	-0.482818
80	10.936738	-0.482853
120	10.333223	-1.342423
120	10.333224	-1.424234
120	10.333225	-1.32434
120	10.333233	-1.322422
120	10.332234	-1.324324
120	10.334221	-1.342444
120	10.33433	-1.243552
120	10.33452	-1.432543
120	10.34334	-1.4325543

3.7. Power Requirements and Sketch of Electrical Circuitry

The table below shows the power requirements of Findguard based on the specifications of the different components that make up Findguard. Information on the voltage and current requirements of each component is obtained from their datasheets online and reading through the literature of other projects which have used it before

Table 3.3: Power requirements

Component	Voltage (V)	Current (mA)
Raspberry pi 3 B	5.1	700-1000mA
GY-GPS6MV2	3.3 - 6	67
HC-SR04	5	15
GP2Y0A02YK0F IR sensor	5	33
Headset Microphone	1.8-2.9	

As can be seen from the table, the system can run on a 9V battery as required because no component needs more than 5V to run. Figure 3.7 shows a sketch of the schematics of the entire hardware design.

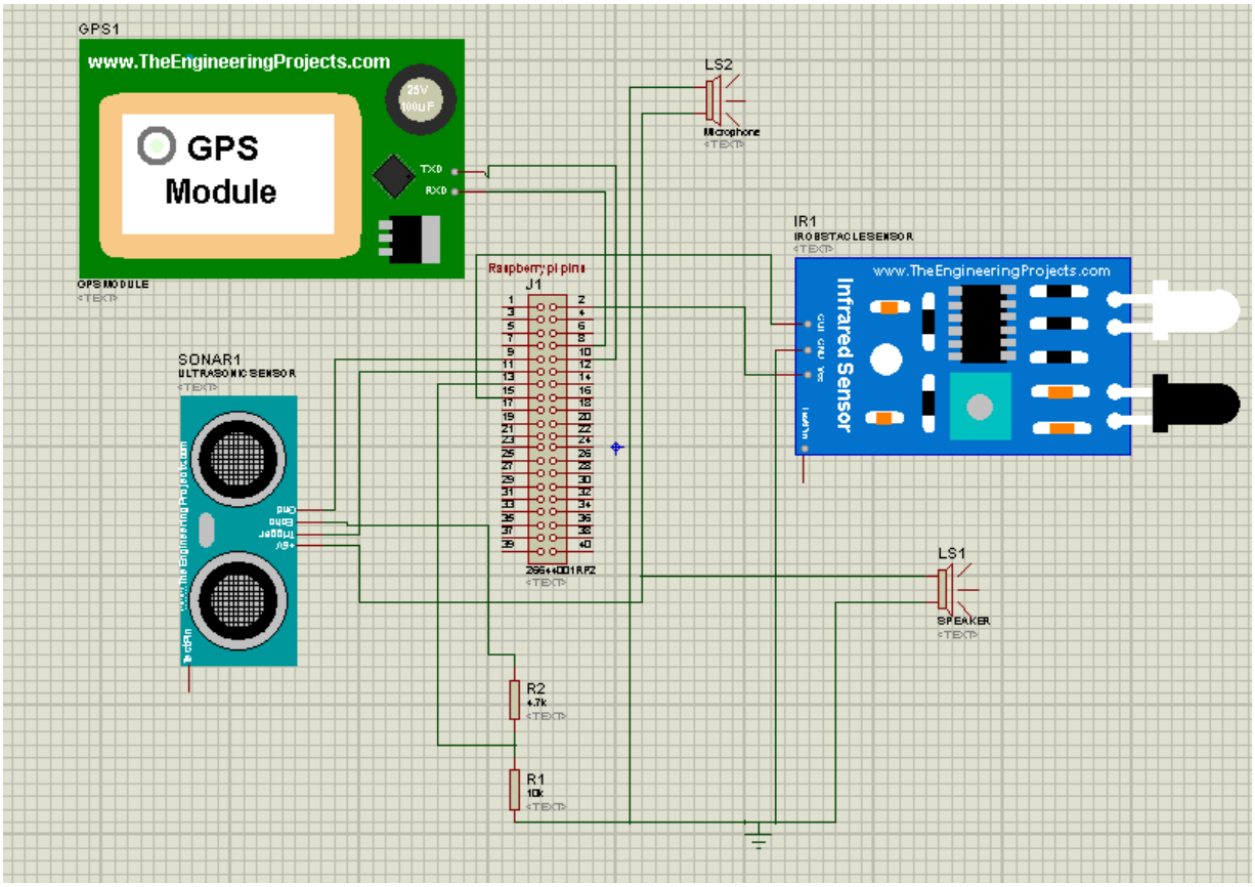


Figure 3.7: Schematic diagram

Chapter 4: Implementation

This Chapter highlight the implementation process of the solution.

4.1. Choice of Programming Language and Coding

The Python 3.6 Programming language was chosen to implement this project. Aside being easy to use, Python is a well-known programming language that comes pre-installed in the Raspbian operating system for the pi. As mentioned in Section 1.5 of this report, COVID-19 led to the closure of schools in Ghana since March 13th, 2020. This has restricted the implementer's access to a screen and HDMI cable to perform a network change on the raspberry pi or do a fresh installation of the Raspbian image. The implementer is therefore unable to implement the project on the raspberry pi as intended due to inability to access the pi console. All coding and testing are therefore done on a PC with the confidence that it will work seamlessly on the pi too. Therefore, the choice of Python as the main programming language of implementation is appropriate as pi OS comes with Python installed.

4.2. Graph Implementation

As stated in Chapter 3, the road network of the user's vicinity is modeled as a graph with the source and destination as the nodes and road itself, the edge with its length, being the edge cost. The graph is implemented as an **Adjacency list** where all the nodes connected to a particular node are added to its neighbors list [12]. From algorithmic point of view, an edge can always be added to the graph at O(1) time complexity. The reader may refer to Appendix A for the Adjacency list implementation of the graph.

4.3. Implementation of Dijkstra's Single Source Shortest Path Algorithm

The Dijkstra's Single Source Shortest Path algorithm has the ability to find the shortest path from a given source to a given destination. The implementation features the use of a priority queue which is implemented with a binary heap data structure to store the graph nodes. The overall runtime of this algorithm is O (ElogV) where E is the number of edges and V is the number of nodes in the graph [12]. See the details of this code in Appendix B.

4.4. Model Road Network

A destination (point D) with three alternative ways of reaching it from a source (point A) was considered to test this program. Point D is reachable from point A directly at a distance of 80 m without going through B or C. You can reach D from A using C as an intermediate point at a total distance of 230 m. When B and C are used as intermediate points, you can reach D at a total distance of 255 m. Figure 4.1 is a graphical representation of the points and the paths connecting them to form a road network.

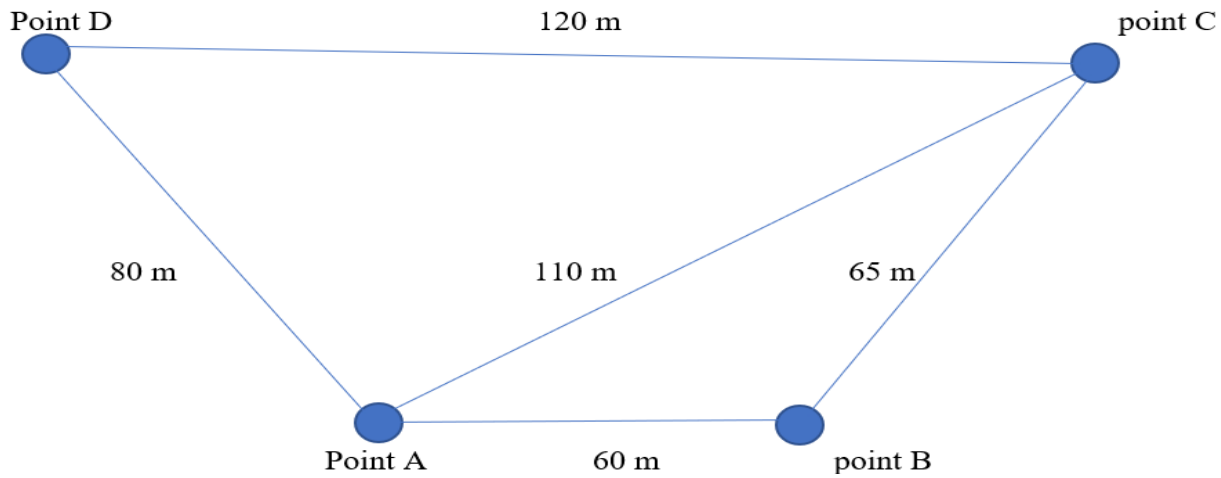


Figure 4.1: Graphical representation of nodes

This graph is represented in adjacency list form with the points A to D as the nodes of the graph and the distances associated with each pair of nodes as the edge cost. The graph is then passed as input to Dijkstra's Single Source Shortest Path algorithm as described in Section 4.4, specifying the starting point as node 0 (that is point A) and the destination as D (that is node 3). The results of executing the program is discussed in Chapter next Chapter. Note that, any node could be selected as source or destination.

4.5. Gathering GPS Coordinates for the Roads

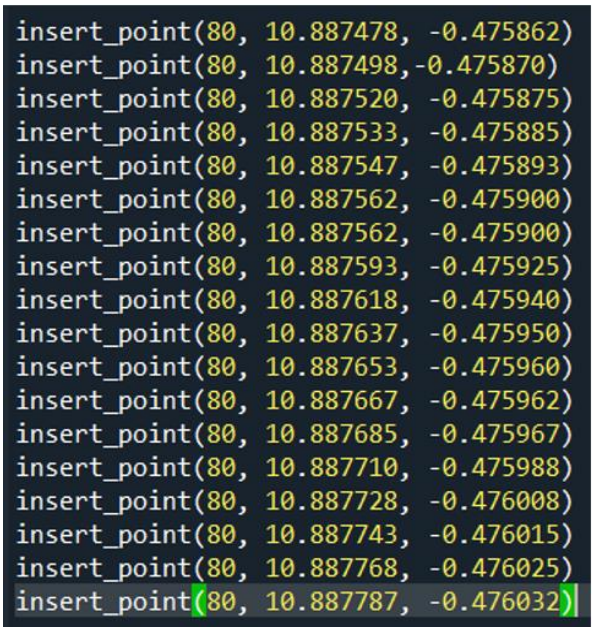
The GY-NE06V2 GPS receiver could not be used because of the problem stated in Section 4.1. A free GPS application on the Google Play Store called **GPS Coordinates** (see appendix C for the app's interface) was used to collect the GPS latitude and longitude pairs of the roads needed for this project. Accuracy of GPS signal varied between 1 m to 5 m. The app had a feature to allow for setting of the time interval and the distance intervals with which readings were taken. In practice, no distinguishable readings were obtainable for distance intervals less than 1 m. To make sure readings were accurate, markings were made on the road at a 1 m interval where points were collected. The arrows indicated in Figure 4.2 below shows the markings. Each latitude and longitude pair were stored in the database as shown in Table 3.2 and as discussed in the database design Section. Figure 5.1(b) shows how the data was put into the database. Obstacles were not catered for because the obstacle detection and avoidance sensors were not implemented due to the problem stated here.

Figure 4.2: Test route

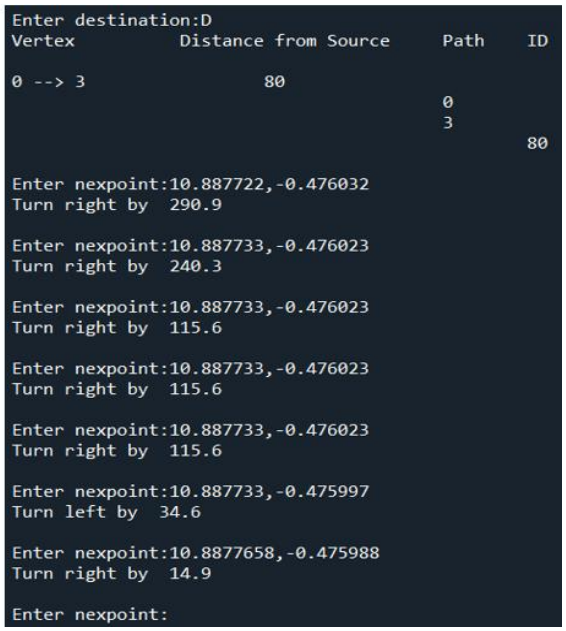
Chapter 5: Results and Discussion

This Chapter presents the results of testing the program as outlined in the previous Chapter. The result is discussed here and possible reasons for the kind of output obtained are highlighted. Suggestions are made on what could be done differently to achieve the desired results.

As shown Figure 5.1(b), the algorithm rightly selected the direct edge joining A to D as the shortest cost path with a cost of 80 m as can be seen in the Figure 4.1. Figure 5.1(a) shows the coordinate pairs that represent this path with the distance used as the path id in the database table. The coordinates of the edges selected by the algorithm as part of the shortest path are then loaded into a python list as tuples. This represents the reference path coordinates. At this point, the user is ready to move to the destination. As described in Section 3.5, to follow a path, the GPS receiver receives real-time fixed interval and plots the path with reference to the reference path. The angle between the reference path and the real-time path determines whether the user is one the right track or not. Since the GPS module could not be used, a free android app (GPS Coordinates) mentioned in Section 4.5 was used to collect GPS coordinates for testing. For this reason, the code was made interactive, so it asks for the next real-time coordinate and computes the angle with reference to the reference path (see Figure 5.1(b)). Figure 5.1(b) shows the output of entering each real-time coordinate. As can be seen, when the user veers to the left side of the road, he or she is prompted to turn right by the angle of deviation to keep on the road and vice versa. However, the output from the program was not consistent. In other times, the direction of turn was right, but the angle was wrong and in other times the direction was wrong.



a. Sample input



b. Sample output

Figure 5.1: Sample input and output

The inconsistency in the output can be attributed to the strength and accuracy of the GPS signals received. The GPS signal kept varying and the app could record with error margins up to 5 m. It was also almost impossible to get the same readings for the same spot many times as the signal strength kept varying. One other thing which might have affected the outcome is the way points were loaded from the database for reference path. Points were processed sequentially, that is, the program processed the next point in the list after the current point when there is an available real-time coordinate. As it is not possible that the user will take same steps at the same interval, the nth point on the reference coordinates could be many meters behind or ahead of the nth real-time coordinate. This implies that, the angle being calculated may not reflect the true angle at the nth point. This could be improved this by using a different algorithm such as the A* algorithm which is an improvement of the Dijkstra algorithm which keeps track of both the distance covered so far and the remaining distance. With this, the nth point could be

approximated from the reference coordinates by averaging some points closer to nth point from both directions of the path. For example, if the path is 10 m long and the user has covered 3 m already, the nth point could be approximated by averaging points that fall within 2 m and 4 m to get the point at the 3 m mark under strict assumption that reference coordinates were collected at constant interval. This ensures that the program is always loading the right point instead of the next point which may not reflect the current position of the user. Keeping track of the distance is as easy as keeping track of the time as the GPS receiver is able to display the speed of the user as well. The distance could then be calculated from the speed and time at every instance.

Chapter 6: Conclusions, Limitations and Future Work

This Chapter presents the conclusions of this project described in the body of this report. The objectives of the project are reviewed, and key takeaways are highlighted. Proposals and directions for future work are outlined in this Chapter as well.

6.1. Conclusion

The objective of this project was to develop a low cost, power efficient, portable GPS and camera assisted navigation system that will guide a blind person to move from point A to B safely.

To achieve this objective, all other parts of the project were discussed and designed for but not built except the GPS part which was implemented and tested to validate the potential of using GPS to guide the user along a path.

The results showed that the success of this project does depend greatly on other measures to improve the quality and strength of the GPS signals for accuracy. The author could not confirm or reject that one can help a visually impaired person to walk along a path using GPS because of the mixed outcome of correct and wrong turns which could not be attributed to the GPS accuracy only. Therefore, further work has to be done and all parts of the project implement as described to make determination.

6.2. Limitations

1 As stated earlier, the COVID-19 pandemic which forced the closure of schools since March 13th in Ghana affected the availability of components to implement and test other parts of the project. The raspberry pi computer could not be used in the final implementation and

testing as the device could not connect to the author's home network. The solution to this is as easy as making a fresh installation of the Raspbian image, but the researcher had no access to the pi screen and HDMI cable to do that. This led to the implementation and testing of the system on a PC with the assumption that, codes written in python on the PC will work fine on the python in the raspberry pi hardware.

2 The current implementation of the graph representing the road network makes it impossible to identify the notes involved as 'source' and 'destination' to make processing of a path in both directions correctly when there are more than two nodes involved. That is, if the coordinates of the path from A to B were collected and stored in the database while moving from A to B, it is not possible to load the same points but in the reverse direction for moving from B to A. The challenge is as a result of the fact that a single node can be both a source and destination depending on where you are reaching it from making it impossible to keep only one state as either source or destination.

6.3. Future Work

1. This project is currently designed to work offline to make it usable in rural areas where there may not be good internet connectivity. As internet coverage and speed continue to increase, this project could be made to work online to make use of other services like Google Maps and other navigation services for ease of implementation and efficiency.
2. The algorithm chosen to process the graph is not the most efficient. Future works could explore the possibility of using the A* algorithm which is an improvement over the Dijkstra algorithm. Other algorithms fit for the problem may be used.
3. The remaining parts of the project which were only discussed but not implemented could be implemented to make a fully-fledged solution to the problem.

References

[1] World Health Organization, "Blindness and Vision Impairment," World Health Organization, 11 October 2018. [Online]. Available: https://www.who.int. [Accessed 07 October 2019].

[2] "Indoor Movement and Orientation: Use Your Senses," VisionAware, [Online]. Available: https://www.visionaware.org. [Accessed 4 March 2020].

[3] P. Thomas, "Disability, poverty and the Millennium Development Goals: Relevance, challenges and opportunities for DFID," 2005.

[4] "Advances in human bionics may require us to rethink our concepts of what it is to be human," Australian Academy of Science, 11 02 2016. [Online]. Available: https://www.science.org.au. [Accessed 09 May 2020].

[5] Y. Shiizu, Y. Hirahara, K. Yanashima and K. Magatani, "The development of a white cane which navigates the visually impaired," in 2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Lyon, 2007.

[6] P. Doshi, P. Jain and A. Shakwala, "Location based services and integration of Google Maps in android," International Journal Of Engineering And Computer Science, vol. 3, no. 3, pp. 5072-5077, 2014.

[7] M. NAKAJIMA and S. HARUYAMA, "Indoor navigation system for visually impaired people using visible light communication and compensated geomagnetic sensing," in 2012 1st IEEE International Conference on Communications in China (ICCC), Beijing, 2012.

[8] Y. Endo, K. Sato, A. Yamashita and K. Matsubayashi, "Indoor positioning and obstacle detection for visually impaired navigation system based on LSD-SLAM," in 2017 International Conference on Biometrics and Kansei Engineering (ICBAKE), Kyoto, 2017.

[9] V. Raj and A. R. Das, "A low Cost outdoor assistive navigation system for blind people," International Journal of Innovative Research in Computer and Communication Engineering, vol. 3, no. 6, pp. 5767-5773, 2015.

[10] S. Koharwal, S. B. Awwad and A. Vyakaranam, "Navigation system for blind - Third eye," International Journal of Innovative Technology and Exploring Engineering, vol. 8, no. 5, pp. 1086-1092, 2019.

[11] L. Junyan, X. Shiguo and L. Yigie, "Application Research of Embedded Database SQLite," in 2009 International Forum on Information Technology and Applications, Chengdu, 2009.

[12] D. Mehta, "Dijkstra's Algorithm for Adjacency List Representation | Greedy Algo-8," GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/. [Accessed 10 May 2020].

Appendix A

```
# -*- coding: utf-8 -*-
"""
Created on Sun May  3 12:22:53 2020

@author: avoka
"""

#from collections import defaultdict
#import sys
class Heap():
    def __init__(self):
        self.array = []
        self.size = 0
        self.pos = []
    def newMinHeapNode(self, v, dist):
        minHeapNode = [v, dist]
        return minHeapNode
    # A utility function to swap two nodes
    # of min heap. Needed for min heapify
    def swapMinHeapNode(self,a, b):
        t = self.array[a]
        self.array[a] = self.array[b]
        self.array[b] = t

    # A standard function to heapify at given idx
    # This function also updates position of nodes
    # when they are swapped.Position is needed
    # for decreaseKey()
```

```
def minHeapify(self, idx):
    smallest = idx
    left = 2*idx + 1
    right = 2*idx + 2

    if left < self.size and self.array[left][1]  < self.array[smallest][1]:
        smallest = left

    if right < self.size and self.array[right][1]< self.array[smallest][1]:
        smallest = right

    # The nodes to be swapped in min
    # heap if idx is not smallest
    if smallest != idx:

        # Swap positions
        self.pos[ self.array[smallest][0] ] = idx
        self.pos[ self.array[idx][0] ] = smallest

        # Swap nodes
        self.swapMinHeapNode(smallest, idx)

        self.minHeapify(smallest)

# Standard function to extract minimum
# node from heap
def extractMin(self):

    # Return NULL wif heap is empty
    if self.isEmpty() == True:
```

```
        return

    # Store the root node
    root = self.array[0]

    # Replace root node with last node
    lastNode = self.array[self.size - 1]
    self.array[0] = lastNode

    # Update position of last node
    self.pos[lastNode[0]] = 0
    self.pos[root[0]] = self.size - 1

    # Reduce heap size and heapify root
    self.size -= 1
    self.minHeapify(0)

    return root

def isEmpty(self):
    return True if self.size == 0 else False

def decreaseKey(self, v, dist):

    # Get the index of v in  heap array

    i = self.pos[v]

    # Get the node and update its dist value
    self.array[i][1] = dist
```

```
    # Travel up while the complete tree is
    # not hepified. This is a O(Logn) loop
    while i > 0 and self.array[i][1] < self.array[(i - 1)//2][1]:

        # Swap this node with its parent
        self.pos[ self.array[i][0] ] = (i-1)//2
        self.pos[ self.array[(i-1)//2][0] ] = i
        self.swapMinHeapNode(i, (i - 1)//2 )

        # move to parent index
        i = (i - 1) // 2;

# A utility function to check if a given
# vertex 'v' is in min heap or not
def isInMinHeap(self, v):

    if self.pos[v] < self.size:
        return True
    return False
```

Appendix B

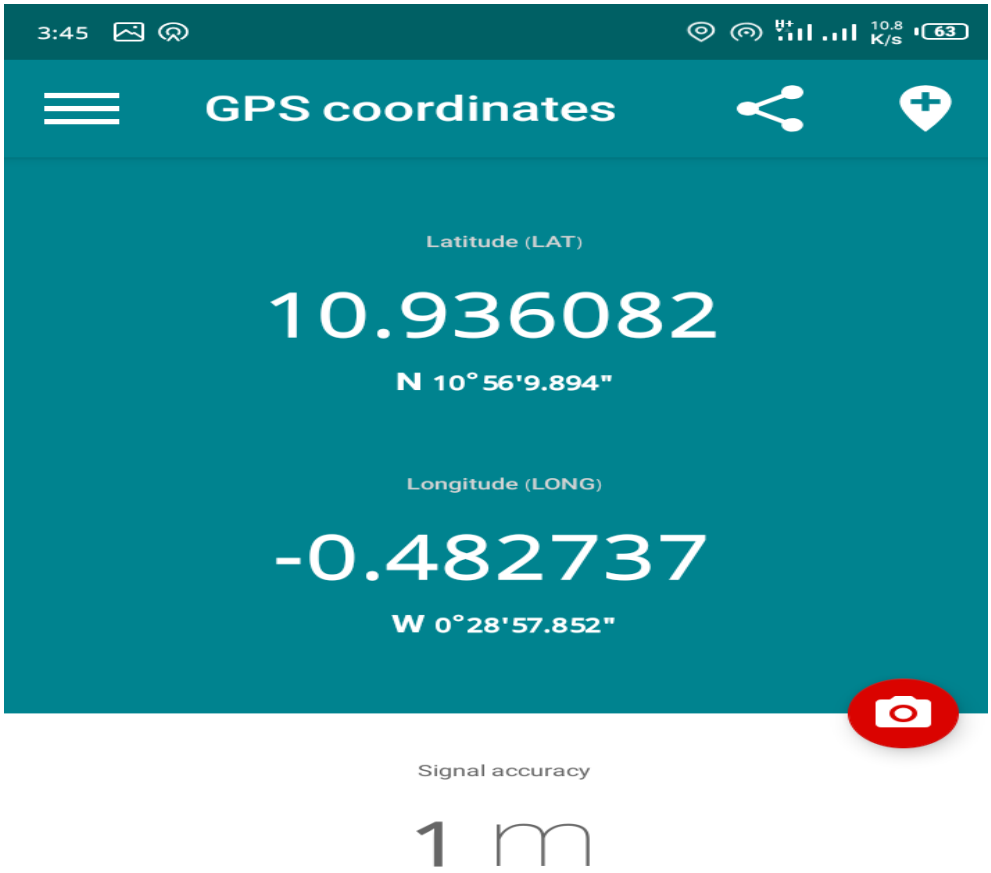
```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 21 19:56:18 2020
@author: avoka
"""
# original sourcecode: https://www.geeksforgeeks.org/
# A Python program for Dijkstra's shortest
# path algorithm for adjacency
# list representation of graph


#from Programs import capstone_db
from Programs import heapPQ
from collections import defaultdict
import sys
paths = []
pathID = []

class Graph():
    def __init__(self, V):
        self.V = V
        self.graph = defaultdict(list)
    # def printId(self, path):
    #     for g in path:
    #         print("\t\t\t\t\t \t",self.graph[g][2][1])



    def printPath(self, parent, j):
        #Base Case : If j is source
        if parent[j] == -1 :
            print ("\t\t\t\t\t", j)
            if parent[j] != -1:
                print ("\t\t\t\t\t\t", parent[j][1])
            return
        self.printPath(parent , parent[j][0])
        print ("\t\t\t\t\t", j)
        if parent[j] != -1:
                print ("\t\t\t\t\t\t", parent[j][1])
                pathID.append(parent[j][1])
                return pathID
        paths.append(j)
            # Function to print shortest path from source to on node
```

```
# A function to print out path.
    def printSolution(self, dist, parent):
        src = 0
        destinations = ["A", "B","C", "D","E","F","G","H", "I"]
        destination = input("Enter destination:" )
        if destination in destinations:
            target = destinations.index(destination)
            print("Vertex \t\tDistance from Source\t Path \t ID ")
            print("\n%d --> %d \t\t%d" % (src, target, dist[target])),
            self.printPath(parent,target)

        else: print("destination not found")


    # Adds an edge to an undirected graph
    def addEdge(self, src, dest, weight):

        # Add an edge from src to dest.  A new node
        # is added to the adjacency list of src. The
        # node is added at the beginning. The first
        # element of the node has the destination
        # and the second elements has the weight
        newNode = [dest, weight]
        self.graph[src].insert(0, newNode)

        # Since graph is undirected, add an edge
        # from dest to src also

        newNode = [src, weight]
        self.graph[dest].insert(0, newNode)
        newNode = [src, weight]
        self.graph[dest].insert(0, newNode)


    # The main function that calulates distances
    # of shortest paths from src to all vertices.
    # It is a O(ELogV) function
    def dijkstra(self, src):

        V = self.V  # Get the number of vertices in graph
        dist = []   # dist values used to pick minimum
                    # weight edge in cut

        # minHeap represents set E
        minHeap = heapPQ.Heap()
```

```
#  Initialize min heap with all vertices.
# dist value of all vertices
for v in range(V):
   dist.append(sys.maxsize)
   minHeap.array.append( minHeap.newMinHeapNode(v, dist[v]) )
   minHeap.pos.append(v)

# Make dist value of src vertex as 0 so
# that it is extracted first
minHeap.pos[src] = src
dist[src] = 0
minHeap.decreaseKey(src, dist[src])

# Initially size of min heap is equal to V
minHeap.size = V;
parent = [-1] * V




# In the following loop, min heap contains all nodes
# whose shortest distance is not yet finalized.
while minHeap.isEmpty() == False:

   # Extract the vertex with minimum distance value
   newHeapNode = minHeap.extractMin()
   u = newHeapNode[0]


   # Traverse through all adjacent vertices of
   # u (the extracted vertex) and update their
   # distance values
   for pCrawl in self.graph[u]:

      v = pCrawl[0]

      # If shortest distance to v is not finalized
      # yet, and distance to v through u is less
      # than its previously calculated distance
      if minHeap.isInMinHeap(v) and dist[u] != sys.maxsize:
        if pCrawl[1] + dist[u] < dist[v]:
            dist[v] = pCrawl[1] + dist[u]
            parent[v] = (u, pCrawl[1])
```

```
                    # update distance value
                    # in min heap also
                    minHeap.decreaseKey(v, dist[v])
        self.printSolution(dist, parent)

#program to test the above functions
graph = Graph(4)
graph.addEdge(0, 1, 120)
graph.addEdge(0, 3, 80)
graph.addEdge(1, 2, 65)
graph.addEdge(1, 3, 110)
graph.addEdge(2, 3, 60)
graph.addEdge(3, 2, 60)
graph.dijkstra(0)
```

Appendix C


3:45
GPS coordinates
Latitude (LAT)
10.936082
N 10°56'9.894"
Longitude (LONG)
-0.482737
W 0°28'57.852"
Signal accuracy
1 m