ON-CAMPUS DELIVERY ROBOT

By

MUHANNAD SAEED ALGHAMDI 1846525 COE
SULAIMAN ABDULLAH ABBAS 1845862 COE
WAEL RABAH ALDHAHERI 1846987 BME

TEAM NO.: 03 FALL-2021 INTAKE

Project Advisor

DR. MOHAMMED BILAL

Project Customer

DR. MOHAMMED BILAL

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
FACULTY OF ENGINEERING
KING ABDULAZIZ UNIVERSITY
JEDDAH – SAUDI ARABIA

JUN 2022 G – SHAWWAL 1443 H

ON-CAMPUS DELIVERY ROBOT

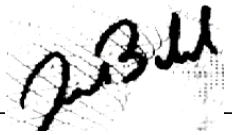
By

MUHANNAD SAEED ALGHAMDI 1846525 COE
SULAIMAN ABDULLAH ABBAS 1845862 COE
WAEL RABAH ALDHAHERI 1846987 BME

TEAM NO.: 03 FALL-2021 INTAKE

A senior project report submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE
IN
ELECTRICAL AND COMPUTER ENGINEERING

CHECKED AND APPROVED (ADVISOR): _____  _____

SDP EVALUATOR: _____

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
FACULTY OF ENGINEERING
KING ABDULAZIZ UNIVERSITY
JEDDAH – SAUDI ARABIA

JUN 2022 G – SHAWWAL 1443 H

بسم الله الرحمن الرحيم

والصلاة والسلام على خير الورى عدَّ الحصى والرمل والثرى

وعلى آله وأصحابه ومن تبع هداهم واهتدى

نحمد الله العلي القدير الذي قدَّرنا على إنهاء هذا المشروع في الوقت المحدد

We Dedicate this project to our parents and families and everyone who supported us throughout this journey.

ABSTRACT

On-Campus Delivery Robot

The According to our information gathering, it is apparent that there is a need for an on-campus delivery solution. And the robot would be beneficial to administrators and students in the university. The campus consists of different terrains in addition having moving pedestrians throughout the campus

We want to create a unified and comprehensive delivery network across the KAU campus without human involvement.

The customer requested a delivery robot that can carry heavy packages, has a range of at least 2 km, has a battery can last for a complete trip. The packages must be secure such that no one but the receiver can access them. The robot is also expect to withstand normal heat

We generated different alternatives using different techniques the top 3 options were the RoboDog, Robot Train and the Ground Robot. After comparing the alternatives based on different factors, we chose the Ground robot as the best solution.

The three main systems of the ground robot are sensor fusion, navigation, and obstacle avoidance. Each of which has gone through multiple trials to reach the final product.

For sensor fusion different sensors were used and throughout the trials some were removed due to unreliable results. In the final phases a complementary filter was used to combine different sensors giving us accurate latitude and longitude and yaw for the robot.

Initial trials of navigation were done indoors. Starting with different calibration methods to ensure accurate results. In later stages GPS was introduced to navigation. Using the data after sensor fusion, we developed a PD controller for both linear and angular speeds to navigate to a waypoint based on its coordinates.

Obstacle avoidance was also implemented using an Oak-D camera for depth estimation. By using a depth map from the oak-d and segmenting the frame into three parts we were able to detect where the obstacle is. In the final product the robot would turn away from the obstacle faster the closer it is.

Index Terms — Robotics, Navigation, Obstacle avoidance, Sensor fusion, Delivery robots.

ACKNOWLEDGEMENT

First of all, we’d like to thank Dr. Mohammed Bilal for his immense help throughout this project. We would also like to thank Eng. Raid for assisting us in the 3D printing parts.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER – 1 INTRODUCTION

1.1 ABOUT THE PROJECT

The project is an ambitious attempt at creating a delivery network robot. Furthermore, this robot is capable of delivering different types of shipments. An important feature is the autonomy of the robot, needing just the destination and package to be given in order to deliver.

The main issues this project is trying to reduce or solve are plenty. To name a few, saving time for the workers since they have to leave their office and deliver papers themselves, aiming to reduce the usage of fuel and manpower and encourage automation in KSA.

1.2 BACKGROUND

There are a few markets that might benefit from delivery services within the campus, in this section we explore and gather more information about some of our target users and we have also surveyed the campus to study the infrastructure.

1.2.1 Potential customers

First, to find out if there is a problem with document delivery on campus, we consulted with some of the administrators in the faculty and we learned that most of their administrative work is done online. Although, some documents that need to be filled/signed are delivered physically.

Another potential customer are students who sometimes need to send/receive items such as books, lab equipment, snacks even. Especially since for example, engineering students have classes in different buildings that aren’t very close to each other.

After surveying the campus, we noticed the following things:

1. There are different terrains in the campus: streets, tiles, mud, grass, etc.
2. Often, there are students in the halls between buildings.
3. Rarely are the streets empty during classes times.
4. The are some ramps for wheelchair users.

1.2.2 Campus map

As shown in the figure, the distance between the faculty and the deanship of administration is approximately 240 meters. And the distance between the faculty of engineering and the faculty of science is 400 meters.[1]

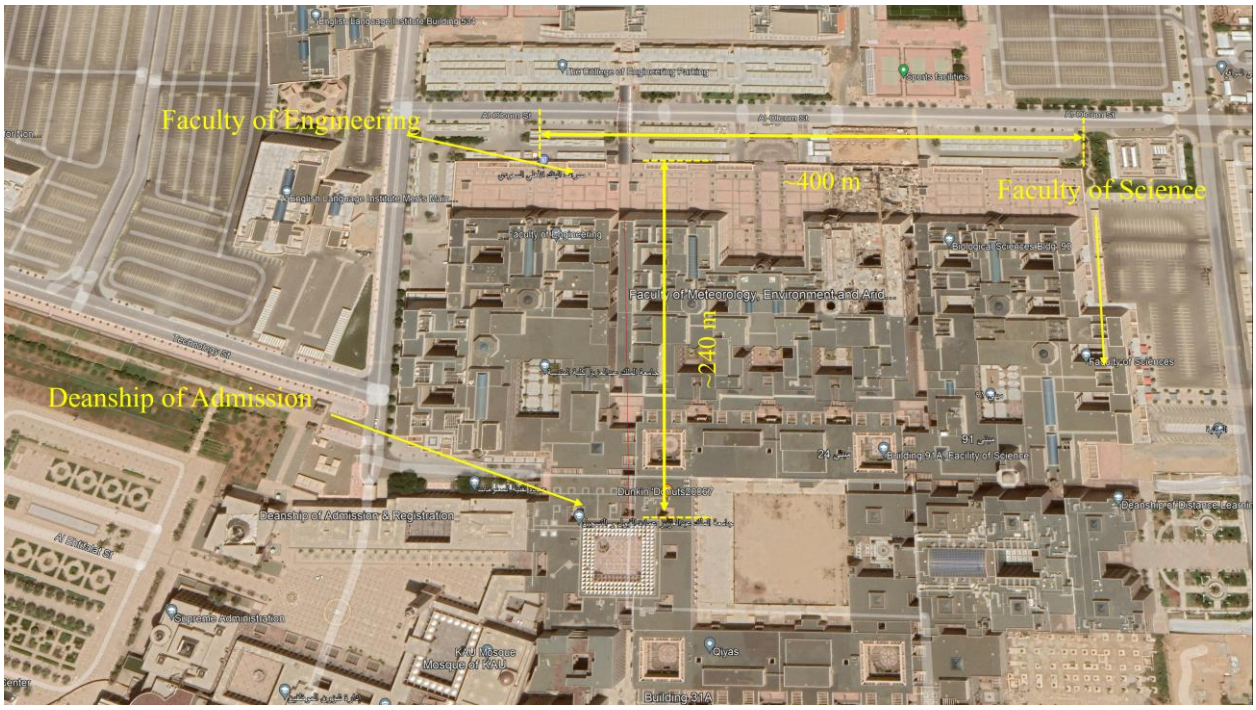


Figure 1 - Campus map. [1]

CHAPTER – 2 CONCEPTUAL DESIGN

2.1 SITUATION DESCRIPTION

The university campus consists of different terrains: roads, tiles, grass which might be challenging to traverse for a ground robot. In addition to that, there are moving objects (people, cars) which might necessitate obstacle avoidance.

2.2 DEFINING THE PROBLEM

Create a unified and comprehensive delivery network across the KAU campus without human involvement.

2.3 PROJECT OBJECTIVES

Lower level:

1. Connect the Whole University buildings into a single automated delivery network.
2. Improve productivity of employees/students by saving their time.
3. Reduce the use of fuel and manpower in the delivery process.

Higher Level:

1. Push to increase development in the tech field industry in Saudi Arabia
2. Raise awareness to decrease the carbon emission, by providing an electrical alternative to gasoline-based vehicles.
3. Encourage upcoming generations to research & develop autonomous solutions.

2.4 PRODUCT DESIGN SPECIFICATIONS (PDS)

Table 1 - Product Design Specifications

Functions	• Traversal of the campus • Carrying the package • Navigation • Maintain the safety of the package
Specifications	• Carry packages up to 80 Kg • The Battery life lasts for a complete trip • Operate within 2 Km range • Operate within 5 km/h.
Constraints	• The cost of the project must not exceed 5000 SAR. • Project must be completed before the end of Term-2. • Hits 0 obstacles during a complete trip.* • Battery life lasts for at least one complete trip (2 km). • The artifact should withstand normal heat (36-40° C) for the duration of the trip (2 km)
Musts	• The ability to move within 2 km range of the Engineering building autonomously on paved roads. • Ensures the safety of the packages. • Includes a storage unit for the shipments. • Tamper proof electronic components. • Made from durable material. • Operate within 5 km/h. • Can carry weight within (80kg). • Guarantees the privacy of the packages.
Wants	• The ability to move around the whole campus autonomously on paved roads. • Can carry weight within (120kg). • Can charge using charging stations. • Costs less than 2500 SAR.
Assumptions	• The university network covers the whole campus or at least a 4G connection is available. • No lock-down or any action that can limit our visits to the targeted campus. • No temporary or constructional change on the campus map. • It is allowed to operate prototypes within the campus roads & facilities.
Scope	For this project, the following must be clear for all parties: • The design is targeted to be specific for the KAU Campus. • The information gathering will be done with the guidance of the advisor and includes all the team members. • Team members will provide all the required resources needed to implement the solution.

	• Team members are responsible of any financial obligations that could be a result of purchasing a required component or subscribing to a license. • The solution will be designed for only outdoor use. • The solution is limited to paved roads with no stairs.

Risks and Remedies	
Risk	**Remedy**
Weak wide fidelity (Wi-Fi) signal.	Use 4G network.
A part breaks down.	Replace with higher quality part.
Shipping delay/dead on arrival.	Order them early/Find local alternative.
A team member quits.	Find another, otherwise just keep going with two.
Term concludes sooner than scheduled	Work for extra hours, fulfill the musts in worst case scenario.

2.5 APPLICABLE ENGINEERING STANDARDS

Most of the connections in our robot are wireless. The wireless communication between the different components of the robot follows the **TCP** communication protocol. TCP stands for transmission control protocol. Which is a standard communication protocol that is connection oriented. Meaning that it establishes a connection between the sender and the receiver before communication is initiated.[2]

OSHA- PUB 8-1.3 (A-4): In this guideline several guarding methods are recommended especially for robots that could be hazardous to the enviroment and One of the recommended guarding methods is a presence sensing device which is implemented in our robot. The presence sensing device is the Oak-D camera which estimates depth, and it facilitates obstacle avoidance.[3]

OSHA- PUB 8-1.3 (A-6): The portable programming control device contains an emergency stop. One of the guidelines is that the control device of the robot has an emergency stop. In our case we have two emergency stop buttons one is using a wireless pen that is connected to the computer controlling the robot and it is used to stop the robot in case of an emergency. In addition to the power button on the robot itself. Another one of the guidelines is about ensuring that the robot can be moved manually even if the system was powered off. And in our robot the robot can be moved manually by just pushing the robot if it's turned off. Another relevant guideline is that the robot should be secured to prevent vibration and to keep the

robot from tipping over. Our robot is quite stable and might even be difficult to tip over, even intentionally.[3]

2.6 LITERATURE REVIEW

As engineers, we are tasked to fully study a situation before engaging in any developmental and or design activities. Furthermore, to understand the current technology and innovation of our century we will conduct a literature review on three cases. Undoubtedly, these cases will cover autonomous robots and some of their uses. It should be noted that most of these cases are within the past few years. Which will give us an accurate lookout on the current technologies. On the other hand, we believe these cases might give us inspiration to lead us into some ideas for our problem solution.

2.6.1 Smart drone delivery system

With the ever-growing usage of online shopping and e-markets, the usual petrol operated vehicles that succumb to gravity is not enough to cover the demand that is skyrocketing. Consequently, the first case study published by Shivaji University in India talks about a smart drone delivery system. In this delivery system, it is proposed that a Quadcopter (QC), which is an Unmanned Aerial Vehicle (UAV), delivers orders requested through online shops autonomously by the use of Google Maps and its own processing unit. The paper also mentions “the QCs capability of delivering parcel ordered by online and coming back to the starting place.”.

The QC’s that will be deployed should have a vertical take-off and landing protocol in order to reduce the area required for functionality. Furthermore, this will allow the QC’s to function in small neighborhoods and streets which many cramped cities have. A 10–15-mile radius can be covered by a single QC. In addition, the vertical approach to these QC’s allows them to carry more payload, this in turn will yield better results for the online shops.

A basic working principle for the QC’s was stated in the paper. A total of four rotary motors at equidistance from each other and a central driver is suggested for a vertical takeoff/landing protocol. To be clear, this configuration functions in a

specific way. As such, opposite rotaries spin in opposite directions while adjacent ones function similarly. Using this design, any gyroscopes controlling the QC's is not needed.

Finally, the methodology for delivering the shipments is as follows; The processing unit of the QC's, which is a raspberry pi, is interfaced with a camera, video streaming, SD card and GPS. When an order is placed, all these technologies work together to find the correct address of the house and deliver the shipment. Below is a figure that showcases the order of operations. [4]

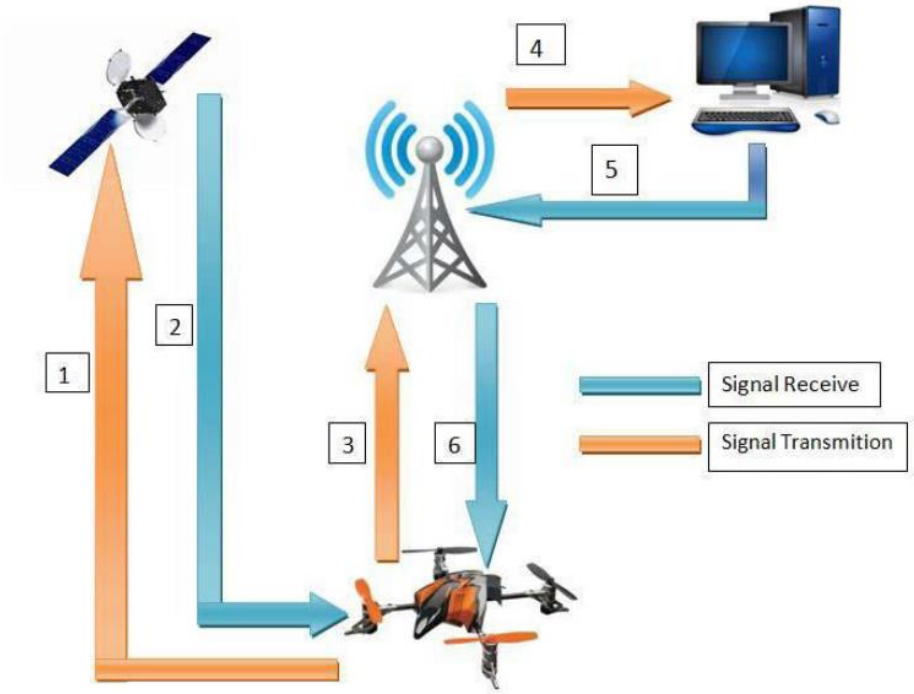


Figure 2 - Showcasing the QC commands. [4]

2.6.2 Prototype design of medical round supporting robot "Terapio"

In this case study, the main focus is a robot that is implemented inside an important ecosystem. Terapio is a medical assistant autonomous robot that will revolutionize the healthcare scene. Its main functions are to deliver armamentarium and provide health care data to the doctors. Whether it be a scalpel, a syringe or even a dental kit, this robot can fetch and receive any equipment a doctor need. Furthermore, the robot has the medical history of the patient and can assist the doctor in evaluating a patient's needs. The robot boasts an internal storage and object detection which can prove to be helpful in navigating the hospital hallways that are clustered with personnel and objects.

Fascinatingly, the robot has three main modes of operation. First, human tracking mode. The robot follows the specified doctor and accompanies him wherever he goes. This mode is useful for the doctor since he does not have to worry about carrying the robot around. Second, Power assisting mode is where the robot can hold the patient, hold a tool or even detect force specified by the doctor. To explain, a dentist might need a tool to be held in a specific way to access a wisdom tooth. The robot can hold the tool precisely and accurately. This in turn can reduce the burden on the dentist and allow him to complete the surgery effectively. Finally, the round mode is an interesting mode. The robot listens to the consultation and records it. Furthermore, the robot denotes everything and organizes it for the doctor for later use. An extra feature of the robot is its facial expressions. These facial expressions help reduce the emotionlessness of the robot which in turn can increase the healing of patients via emotions. Below is a figure representing the modes of Terapio. [5]

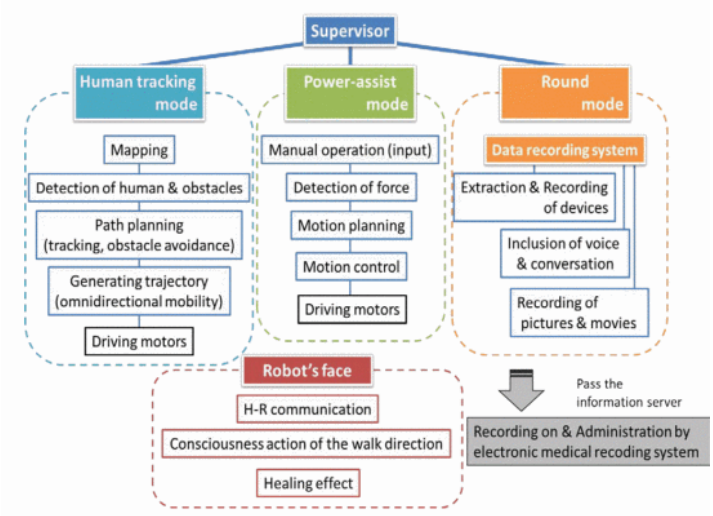


Figure 3 - Showcasing Terapios functions. [5]

2.6.3 : Design and development of autonomous delivery robot

This research published by Visvesvaraya National Institute of Technology discusses the design of an autonomous delivery robot with some limitations. This project is capable of carrying a maximum weight of 2kg this low-bar line is due to the use of low torque motors. To carry higher weights the use of higher torque motors is essential. On the other hand, the motors used have 300rpm which increases the ability of moving the robot much faster ensuring it is within the road

speed limits. For the main board the project uses Nvidia Jetson TX1 for controlling and running global and local planning algorithms. It also uses an Arduino Mega to control the motors and manage the sensors' readings. The two boards communicate using rosserial to publish the sensors' readings and to receive the moving commands controlling the motors. The design uses multiple components to provide very accurate localization and obstacles avoidance, one main component used is a RGBD camera which is used to get the front view with a 3D depth map in addition to a laser range finder which gives a 2D depth map. Another component used for an accurate localization of the robot and for the purpose of determining the orientation of the robot is the inertial measurement unit (IMU) which includes an accelerometer, a gyroscope and magnetometer for a better estimate of the position. To avoid collision, an IR sensor with a range of 4-30 cm is used as the last option to save the robot from collision.

For this project, a map-based localization approach is implemented. Which leads to some potential problems that could be a result of the accumulative errors of the used sensors (e.g., the GPS could produce a 10 meters error in some cases), For this reason the paper suggests the use of statistical filters for more accurate localization. The paper discusses another essential topic for navigating an autonomous robot which is the use of high-definition maps or simply ADAS maps, which could lead to an accuracy of 10 cm. Although the project did not use SLAM (Simultaneous Localization and Mapping), but the paper suggests the use of SLAM as a solution to improve the accuracy of mapping localization of the robot. The paper also discusses the used algorithm for path finding, in this case it is the A-star algorithm. Although it is not the best algorithm in finding the shortest path, but the researcher justifies that the project needs a fast algorithm more than an accurate but slow one to avoid getting stuck. however, the researcher suggests further research to be carried out for the optimization of the grid generation. [6]

2.6.4 : A Raspberry pi delivery robot controlled by live stream chat

Droiid is a delivery robot created by Even Kouao a youtuber and a software engineer at Vodafone. Droiid is a delivery robot controlled through the commands sent on a live stream chat on Twitch. Unlike other robots, Droiid has six wheels instead of four this could make its movement smoother than other four wheels robots. The outer design of the robot is a custom 3D printed design to ensure it is

not very heavy. However, the downside is it may not survive under strong circumstances. Droiid is not fully autonomous as it still needs to receive its commands from the live stream chat. The robot has two main features, Speech, and movement in both cases the commands are received from the live chat. Which convert the text message received form the Droiid server to a speech sent to the output speaker[7]

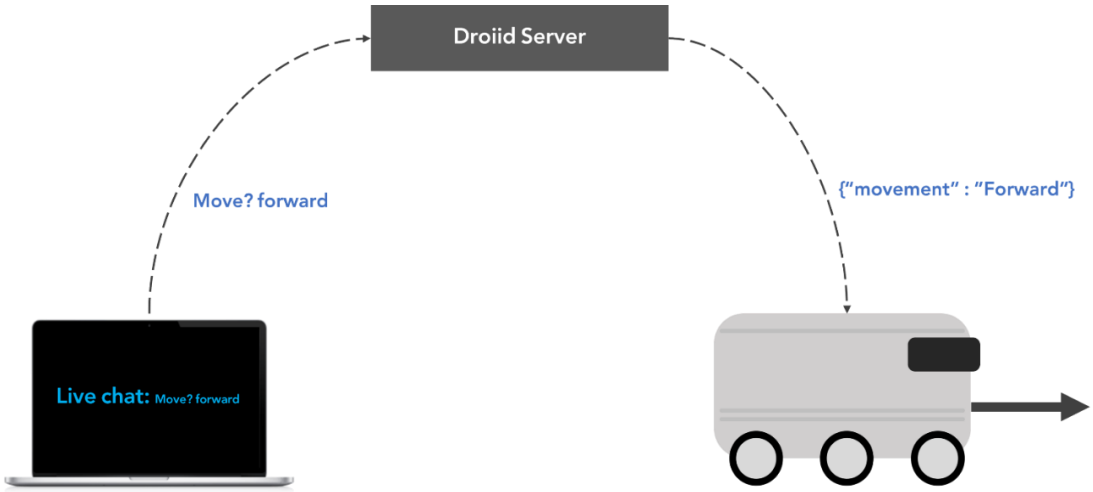


Figure 4 - Droiid movement and commands. [7]

2.7 THEORETICAL BACKGROUND AND ANALYSIS

In term-1 different concepts in addition to components and their types were discussed. Including motors, dc motor drivers, ultrasonic sensors, and more. This section was added to introduce some sensors that weren't discussed in term 1.

2.7.1 Sensors

Sensors are an important part of any robot; they can retrieve information that is easily accessible to us humans. Temperature, distance, oxygen levels to name a few are converted into an electrical reading that can be processed by the brains of the robot. Furthermore, they can be used as a way to give important parameters to a robot in order to function properly. Below you will find a few sensors that could be used to further enhance our project idea.

In term-1 we discussed (LIDAR, ultrasonic, and IR sensors). And while these sensors could be useful in some applications, they arent optimal in our case. First of all, because we are dealing with open spaces and these sensors usually have a short range. In addition to these sensors only detecting things in a narrow field of view.

Oak-D Camera

The Oak-D camera is a device that contains three cameras. Two of which are called stereo cameras and can only see in greyscale, these are used for depth estimation. The third camera is a color camera. The Oak-D is also capable of running neural network models. So, it is capable of things like face recognition, object recognition, etc. Although in this project, the main focus will be on the stereo cameras. Namely for depth estimation.[8]

Figure 5 - Oak-D Camera. [8]

The depth is calculated using the following equation:

$$Depth = \frac{focal\_length\_in\_pixels \times baseline}{disparity\ in\ pixels}$$

where the focal length is equal to:

$$Focal\_length\_in\_pixels = \frac{image\_width\_in\_pixels \times 0.5}{\tan\left(\frac{HFOV \times 0.5 \times \pi}{180}\right)}$$

So, for 400p mono camera resolution where HFOV=71.9 degrees

The focal length would be equal to **441.25 pixels**

The baseline is the distance between the two stereo cameras and is **7.5** cm for the Oak-D camera.

Using the values above the depth equation with the resolution set to 400p is:

$$Depth = \frac{441.25 \times 7.5}{disparity}$$

Assuming the disparity was 50 pixels the depth would be equal to: **66.19 cm.** [9]
The disparity will be discussed in more depth in the obstacle avoidance implementation section.

IMU Sensor

Inertial measurement unit or IMU for short, is a conglomerate of sensors containing an accelerometer, gyroscope and sometimes a magnetometer. [10]

The accelerometer detects linear acceleration. Because gravity is downwards acceleration, it can also measure gravity. The figure below shows the relative axes of an android phone. The accelerometer can detect the linear velocity component in each direction (x, y, z). [11]

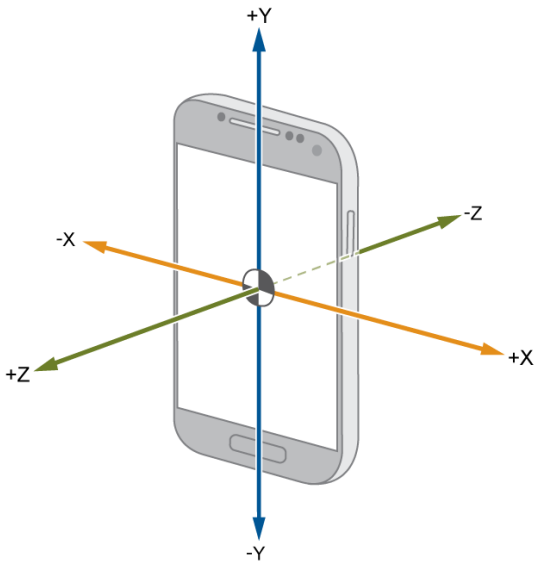


Figure 6- Relative axes of an android phone. [11]

If the phone was laid down on a flat surface, rotation around both the x and y axes can be detected by the accelerometer because there is a gravity component that is affected by that rotation whereas the gravity in all directions would be the same if the phone was rotated around the z-axis. Because of this “blindness” of the accelerometer to changes in yaw (z-axis rotation), it isn’t used in our robot. The unit for each component is in **m/s²**. [11]

Unlike accelerometers which measure linear acceleration, gyroscopes measure angular velocity. Gyroscopes can also be used to determine the object’s orientation in 3d space using roll, pitch yaw. Which are rotations in the x-axis, y-axis, and z-axis, respectively. The output of gyroscopes is in **rad/s**. [12]

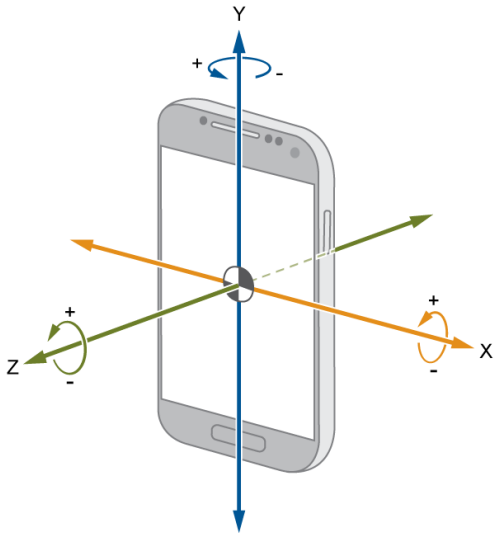


Figure 7 - Rotation around the relative axes [12]

Magnetometers measure magnetic field. Because Earth has a magnetic field this is also detected by magnetometers. One example of a magnetometer is compasses. A Compass when there are no other forces acting on it, points north due to the earth magnetic field. Magnetometers outputs are in **µT**. And it can be used to determine orientation[12]

Finally, Android phones also have a GPS sensor giving us the latitude, longitude, and altitude. Which is done using antennas that connect to a network of satellites to determine position.

Magnetometers data can be fused with data from accelerometers and gyroscopes using sensor fusion. Which is used to determine the movement and orientation of the IMU. The limitations of these sensors will be further discussed in the sensor fusion implementation section. [13] [14]

2.8 ANALYZING ALTERNATIVE SOLUTIONS

We start by generating different ideas using brainstorming and a morphological chart. Then the alternatives are described using a glass box and a cost analysis table and then the alternatives are compared using Kepner Tregoe decision analysis (KTDA). The cost analysis will be for implementation as a senior design project as some projects might become more feasible on a larger scale

2.8.1 Morphological chart

Table 2 - Morphological chart

Function	**Means**				
	1	**2**	**3**	**4**	**5**
1. Traversal of the campus	Wheels	Propellers	Continuous wheel track	Quadruped/Biped	Conveyer belt
2. Carrying the package	Wagon	Carried Box	Pneumatic tubes		
3. Navigation	GPS	Predefined path	Pilotage [i]		
4. Maintain the safety of the package	Digital lock	Combination Lock	2-step verification		
Feature	**Means**				
	1	**2**	**3**	**4**	**5**
1. Fast	Motors speed	Wheel size	Lightweight		
2. Long battery life	Energy consumption	Capacity	Recharge time		

[i] navigating by reference to visible landmarks

- **Alterative A:** Quadruped robot with a wagon attached to it, navigates using pilotage, the packages are secured using a 2-step verification system (2FA)
- **Alternative B**: Conveyer belt where the packages are put in a plastic box and placed on it, the conveyer belt spans the campus on a predefined path, the packages are secured using a combination lock.
- **Alternative C**: Ground robot that uses wheels to traverse the campus, the package is loaded on the robot (carried box), it navigates using GPS, the packages are secured using a Digital lock

Ideas from initial brainstorming:

- **Alternative D:** Robot Train
- **Alternative E:** Pneumatic System
- **Alternative F:** Drone delivery

2.8.2 Alternative analysis

Ground Robot

A ground robot with a built-in storage cart uses four wheels to move between buildings within the university campus. The robot needs to be able move safely using obstacle avoidance techniques to avoid objects around it. It uses battery, to supply enough power for a complete back and forth trip. To be able to move between buildings, the ground robot needs a navigation algorithm to guide the robot through the whole trip.

Pros:

- Dynamic stop points
- Flexible pathfinding
- Doesn’t require special roads
- Moderate complexity

Cons:

- Requires obstacle avoidance algorithms
- Higher probability of getting stuck
- Easily tampered with
- Training phase is required

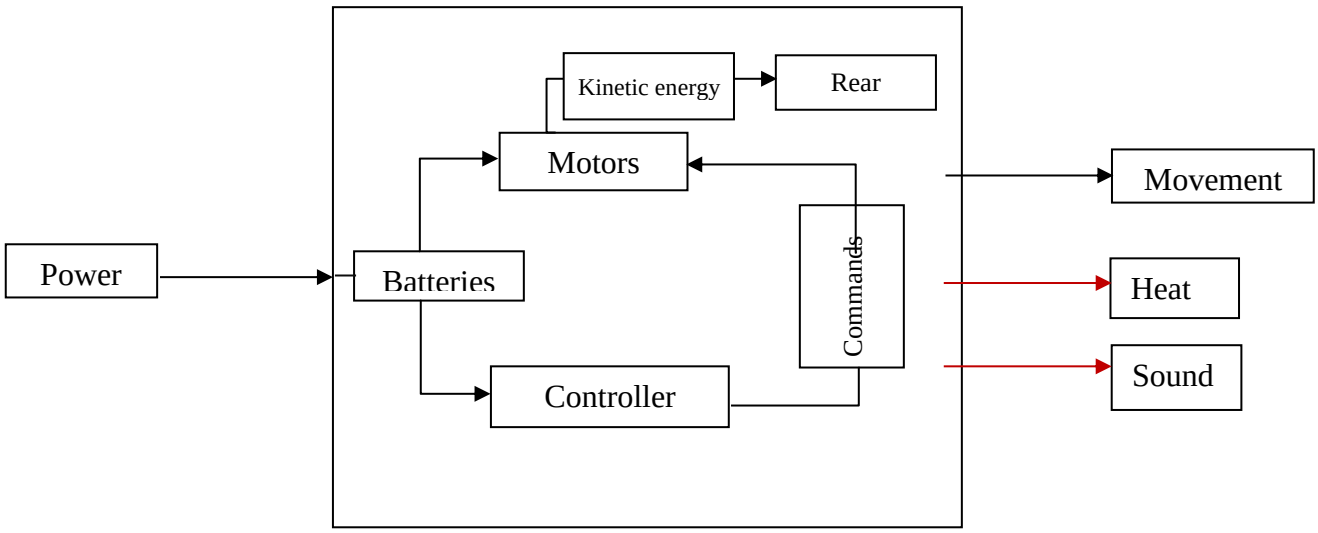


Figure 8 - Ground robot glass box

Table 3 - Cost analysis for ground robot

ITEM	Expected Price	Quantity	Total
Jetson nano	$75.00	1	$75.00
Stereo Camera Module Compatible with Jetson Nano	$45.00	1	$45.00
Motor driver	$12.00	4	$48.00
Platform	$100.00	1	$100.00
Battery	$20.00	1	$20.00
Wheels	$12.00	4	$48.00
Motors	$15.00	4	$60.00
Total cost			$336.00

RoboDog

This alternative is a quadruped robot. It uses two front legs and two hind legs to move just like an animal would. Furthermore, it can traverse at acceptable speeds and carry a heavy load. This alternative can also walk on the campus and roads.

Pros:

1- Innovative
2- Can climb stairs
3- Can drag or carry the packages

Cons:

1- Complex
2- Easily tampered with (pushing, vandalizing, etc.)

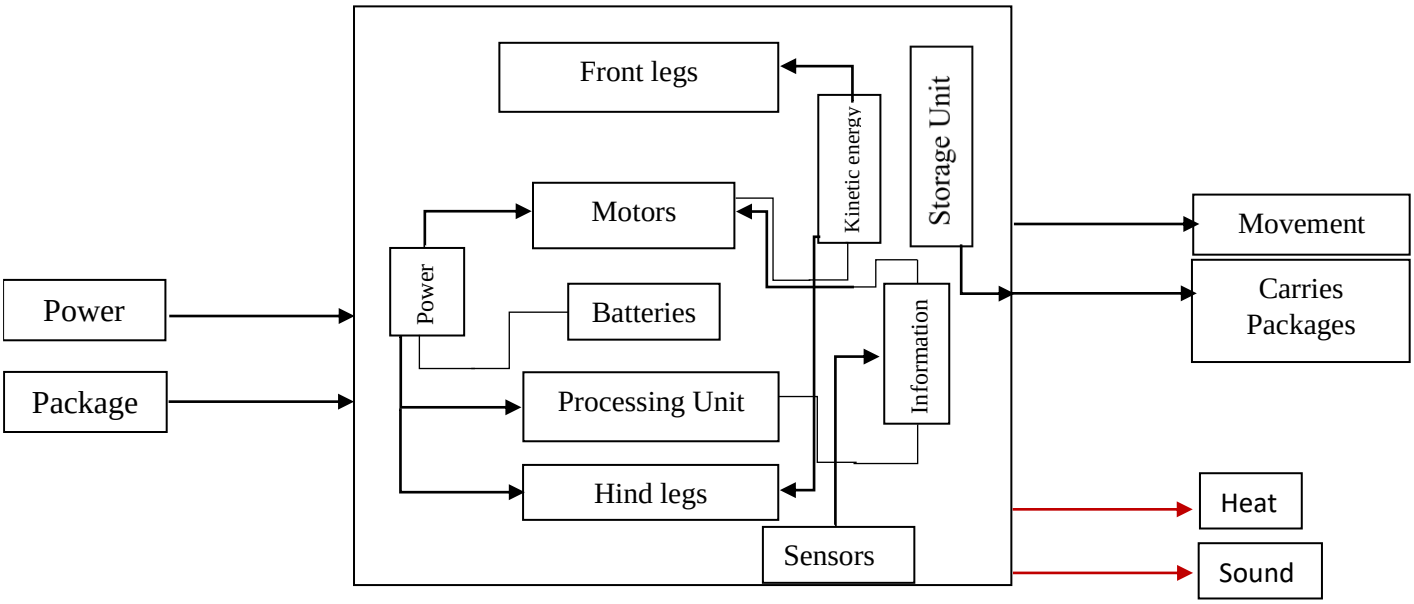


Figure 9 - RoboDog glass box

ITEM	Expected Price	Quantity	Total
Jetson nano	$75.00	1	$75.00
Stereo Camera Module Compatible with Jetson Nano	$45.00	1	$45.00
LIDAR sensor	$100.00	1	$100.00
Gyroscope sensor	$15.00	1	$15.00
Wagon	$60.00	1	$60.00
Battery	$20.00	1	$20.00
Artificial limbs (3D printed)	$40.00	2	$80.00
Servo motors	$20.00	8	$160.00
Total cost			$555.00

Table 4 - Cost analysis for RoboDog

Train robot

To be able to use a mini train within campus, we need train tracks that connect the buildings we want to deliver to and from. By having different stops at different faculty buildings, and smart train junctions that can save time by switching to shorter routes the user can drop off the packages at 1 of the stops and then the receiver would take the package at the stop nearest to him/her.

Pros:

1. Eliminates the need for navigation algorithms.
2. If pedestrians avoid the tracks, no need for obstacle avoidance.
3. Can be fast.

Cons:

1. Requires train tracks to function.
2. Sub-optimal routes
3. Limited to set of stops

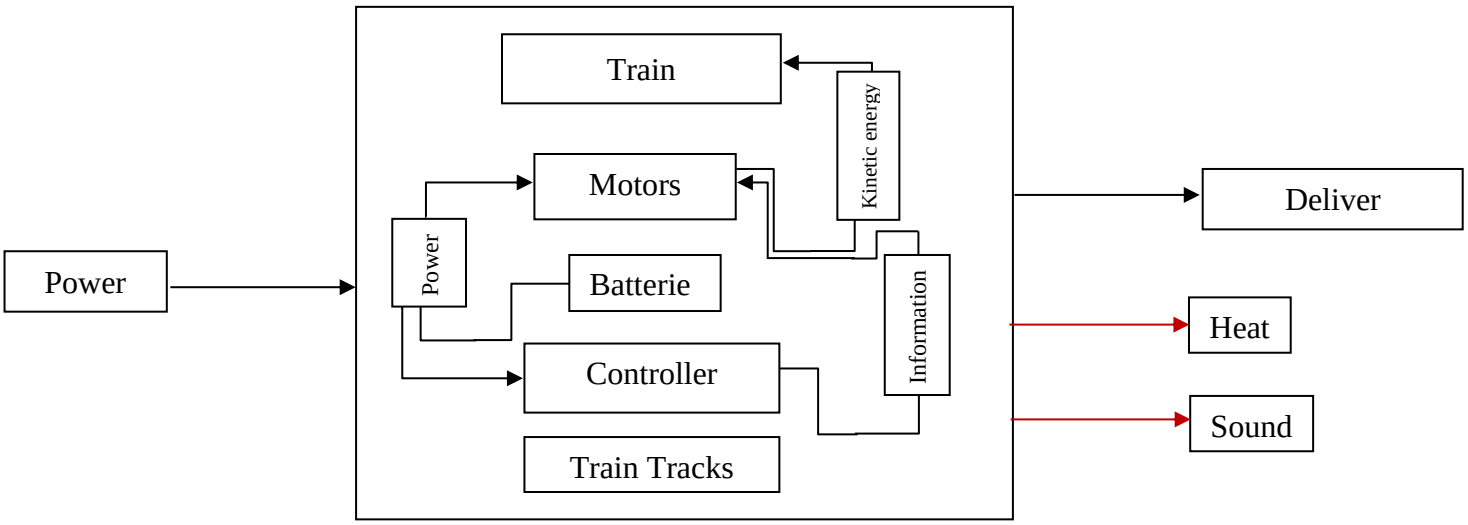


Figure 10 - Train robot glass box

Table 5 - Train robot cost analysis

ITEM	Expected Price	Quantity	Total
Arduino uno	$40.00	1	$40.00
Wagon	$60.00	3	$180.00
Battery	$20.00	1	$20.00
Traction motor	$450.00	2	$100.00
Total cost			$340.00

2.8.3 Choosing baseline design

Table 6 - KTDA analysis

Musts/Alternatives	A: Quadruped Robot	B: Conveyer Belt	C: Ground Robot	D: Robot Train	E: Pneumatic tube system	F: Drone
The ability to move within 2 km range of the Engineering building	GO		GO	GO		
Ensures the safety of the packages.	GO	GO	GO	GO	GO	GO
Includes a storage unit for the shipments.	GO	GO	GO	GO	GO	GO
Tamper proof electronic components	GO	GO	GO	GO	GO	GO
Made from durable material.	GO	GO	GO	GO	GO	GO
Operate within 5 km/h.	GO	NO GO	GO	GO	GO	GO
Can carry weight within (80kg).	GO		GO	GO	NO GO	NO GO
Constraints						
The cost of the project must not exceed 5000 SAR.	GO		GO	GO		
Project must be completed before the end of Term-2.	GO		GO	GO		
Causes no harm to the surroundings.	GO		GO	GO		
Battery life lasts for at least one complete trip (2 km).	GO		GO	GO		
Guarantees the privacy of the packages.	GO		GO	GO		
The artifact should withstand normal heat (36° C) for the duration of the trip (2 km)	GO		GO	GO		

After analyzing the alternatives, we chose the **Ground Robot** to be our baseline design. Due to it having the lowest cost, balanced complexity, and the availability of the parts. We modeled our idea using SolidWorks to be able to imagine our baseline design.

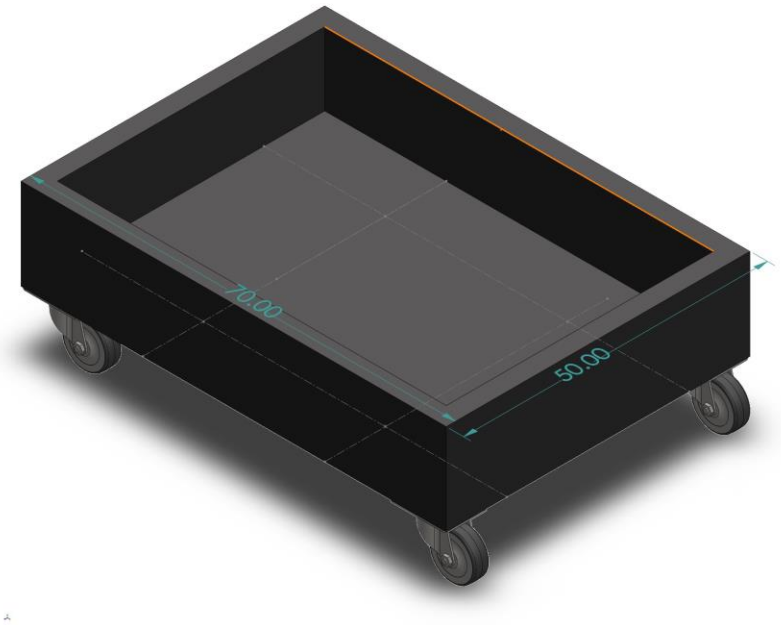


Figure 11 - Initial SolidWorks model

2.9 MATURING BASELINE DESIGN

To further improve our base model, we started thinking of things we could change. We thought of reusing a hoverboard instead of buying the different motors and motor drivers. Hoverboards contain powerful motors that could carry people (~100 kg) which makes the design sturdier. Furthermore, wheel encoders are included in the hoverboard eliminating the need to buy them. In addition to that, the hoverboards have a recharging capability that is simple to use. We also thought of adding ventilation slits to the wagon to improve thermal dissipation. In addition to that, one idea was to add a hole in the wagon for cable management to eliminate the possibility of unwanted friction of the wires with the ground. The changes mentioned above are what we thought of in term-1. Further improvements are discussed in chapter 4 (implementation).

CHAPTER – 3 PRODUCT BASELINE DESIGN

3.2 BLOCK DIAGRAM

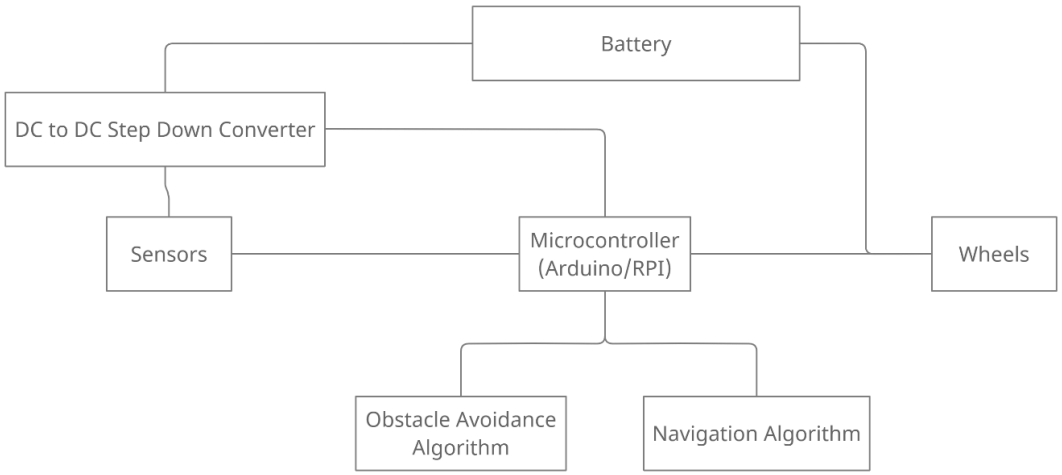


Figure 12 - Block diagram

The figure above showcases the main subsystems we projected in term-1.

3.2 SYSTEM DESCRITPION

As previously mentioned, the communication protocol used for wireless communication in the robot is the **TCP/IP** protocol. The Oak-D camera supports both USB2 and USB3 communication. With speeds of 5Gbps, 10 Gbps respectively. The figure below illustrates the specifications of the sensors in the Oak-D. [9]

Sensor specifications

Camera Specs	Color camera	Stereo pair
Sensor	IMX378	OV9282
DFOV / HFOV / VFOV	81° / 69° / 55°	82° / 72° / 50°
Resolution	12MP (4032x3040)	1MP (1280x800)
Focus	Auto-Focus: 8cm - ∞	Fixed-Focus: 19.6cm - ∞
Max Framerate	60 FPS	120 FPS
F-number	2.0	2.2
Lens size	1/2.3 inch	1/4 inch
Pixel size	1.55µm x 1.55µm	3µm x 3µm

Figure 13 - Oak-D sensor specifications [9]

3.2.1 Circuit schematics

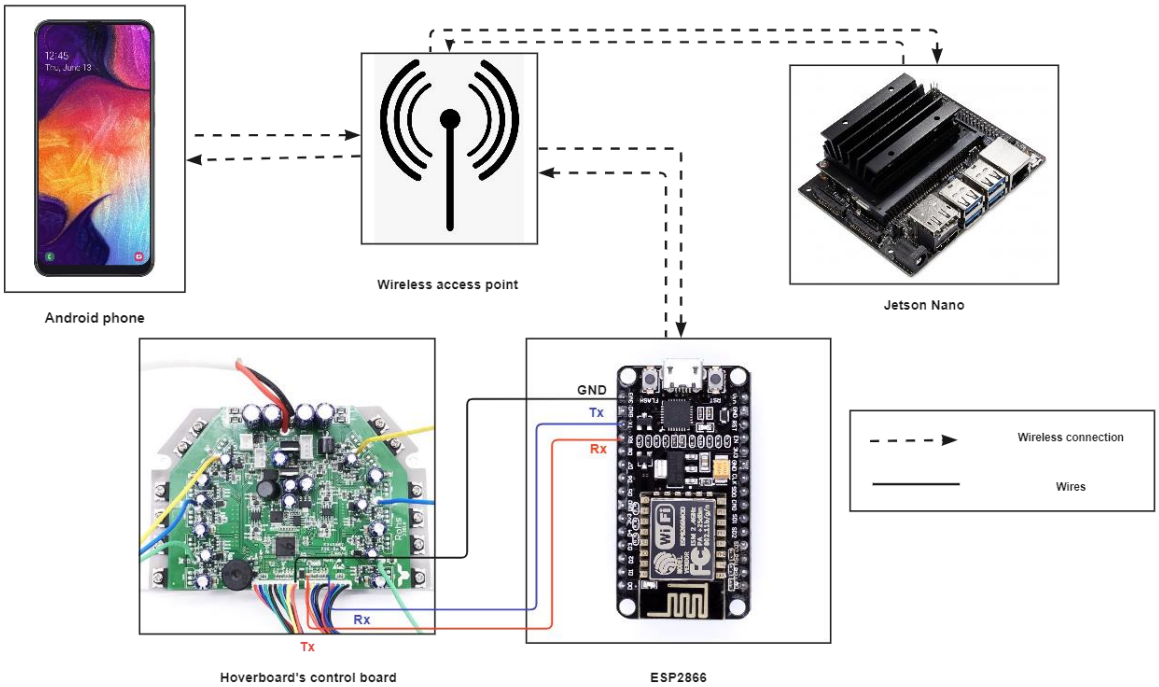


Figure 14 - Main system circuit schematic.

The figure above illustrates both the hardware wired connections and wireless connections. The hardware wires are represented by solid lines and the dotted lines represent wireless connections.

3.2.2 Circuit component specifications

Table 7 - Circuit components specifications

Component	Specifications
Jetson Nano	**Manufacturer:** Nvidia **GPU:** 128-core NVIDIA Maxwell **CPU:** Quad-core ARM® A57 **Memory:** 2 GB 64-bit **Module Size:** 70mm x 45mm
NodeMCU ESP8266	**Microcontroller:** ESP-8266 32-bit **Module Size:** 49mm x 26mm **Clock speed:** 80 MHz **Operating Voltage:** 3.3V **Input voltage:** 4.5V-10V **SRAM:** 4 MB / 64 KB **Digital Pins:** 11 **Analog Pins:** 1 **Built-in Wi-Fi:** 802.11 b/g/n **Temperature range:** -40C - 125C
IMX219-83 8MP 3D Stereo Camera Module	**Megapixels:** 8 Megapixels **Resolution:** 3280 × 2464 (per camera) **Dimensions:** 24mm × 85mm **Gyroscope Operating Current:** 1.23mA **Angle of View:** 83/73/50 degree (diagonal/horizontal/vertical)
dc-to-dc convertor 12v to 5v	**Manufacturer:** Chuanguifa **Type:** step down dc-to-dc convertor **Input voltage:** 12V **Output Voltage:** 5V **Output current:**3A **Output power:**15W **Efficiency:** ＞96% **Operating temperature:** (-40°, 80°) **Size:** 45x26x12 (mm)
Rechargeable battery	**Voltage:** 36 V **Discharge Current:** 75A **Dimensions:** 101mm x 90mm x 70mm **Weight:** 1.62 Kg **Internal Resistance at 1KHz:** 34 mΩ
Motors	**Manufacturer:** huhu **Speed:** 100 RPM **Operating voltage:** 36 volts **Torque:** 10 Kg.cm **Weight:** 158.75 g

3.2.3 Flowcharts for software blocks

Navigation flowchart

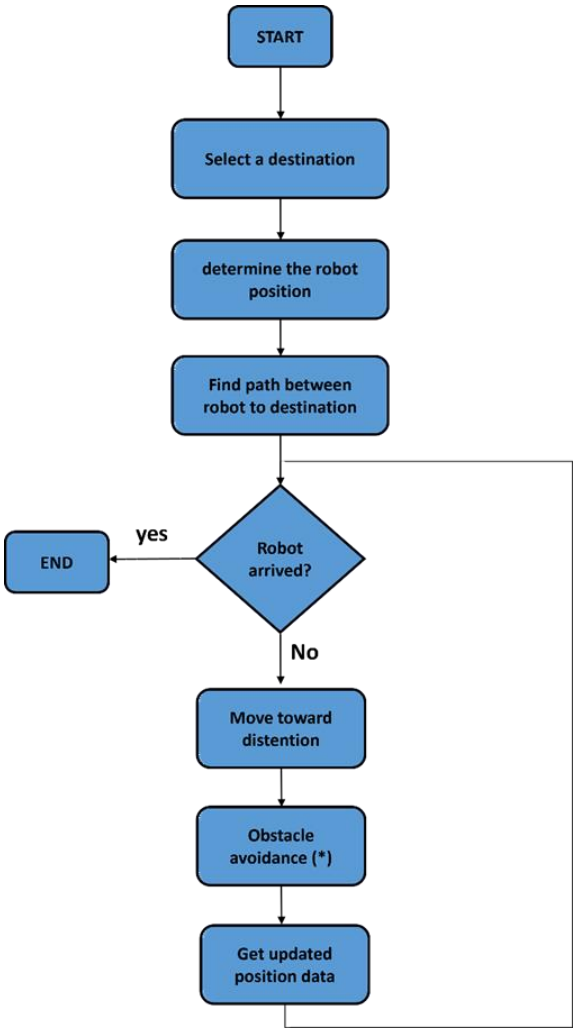


Figure 15 - Navigation flowchart

Obstacle avoidance flowchart

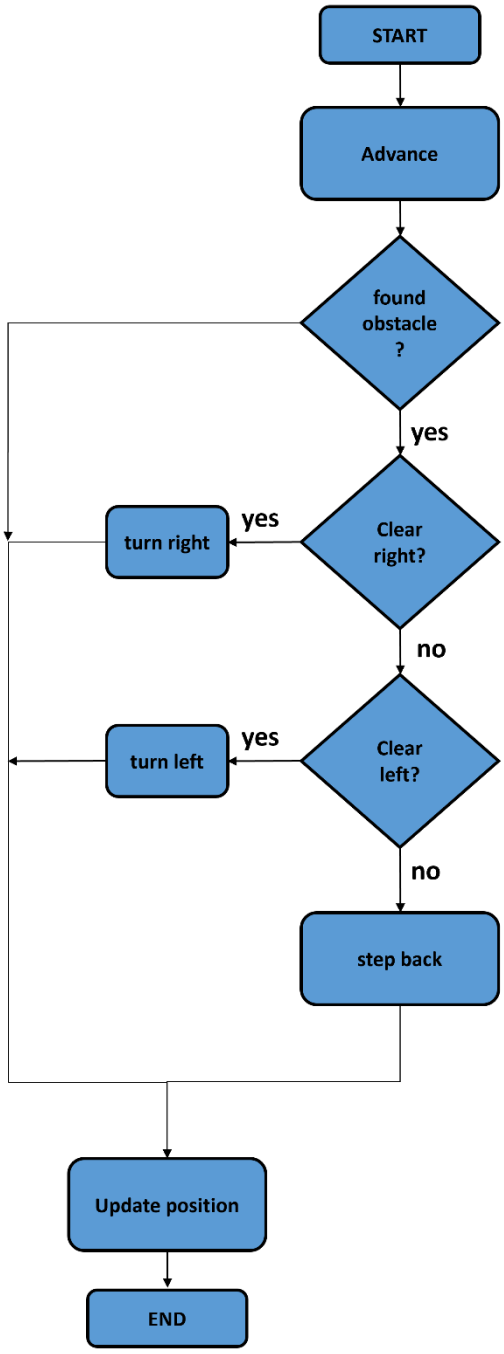


Figure 16 - Obstacle avoidance flowchart

3.2.4 Mechanical specifications of the case

The figure below shows a SolidWorks model of the cart after adding the changes discussed in maturing the baseline design. The cart is a rectangular prism with the dimensions (L×W×H) (70×50×20 cm).

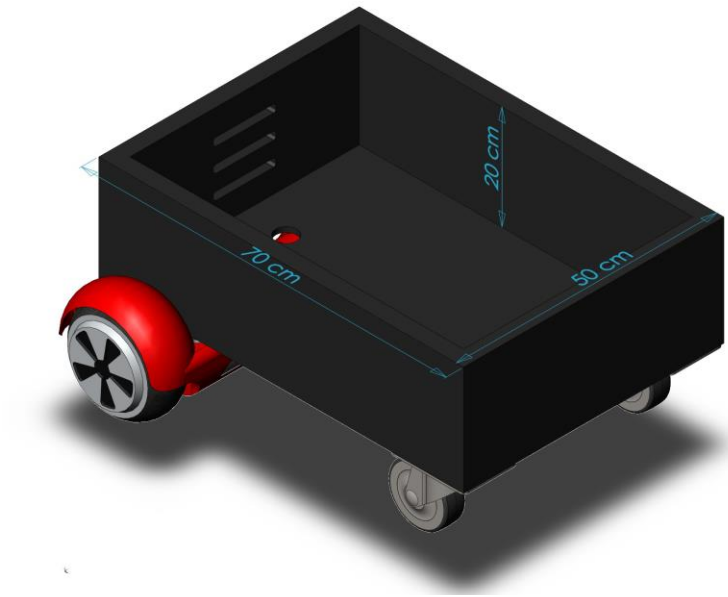


Figure 17 - SolidWorks model of the robot

3.2.5 Possible aesthetics

Aesthetics are an important part to make the user feel more connected to the robot. There are two main aesthetical changes that would be beautiful if applied to the artifact. First, stickers that can be applied to the robot with the logo of KAU and Engineering Faculty. These stickers can be applied to the sides of the platform to showcase that the robot is from the students of the faculty. This in turn may help the users feel a sense of home.

Another beautification idea is painting the robot white in color. This can allow the robot to be seen clearly from far away which can help people recognize the robot. Even though the robot might not need this, it helps promote a beauty aspect in robots.

Figure 18 - KAU icons

3.2.6 Input/output specifications

An AC input power source of 220Vac with 60Hz frequency, single phase is needed to supply the battery with charging power. Max input current is 2A current with max power consumption of 440W.

3.2.7 Operating Instructions

Operation of the robot is very simple; any layman can use it without issue.

1- Locate the robot within a stop point.

2- Place shipment/package inside the luggage compartment.

3- Open the robot UI using any smart device.

4- Enter destination for the shipment and receiver details.

5- Enter lock # for security.

6- Press Start.

7- Wait for confirmation from receiver. (done)

3.3 SIMULATION RESULTS

For simulation we made a localization program using MATLAB. We connected the Arduino to MATLAB, and we used ultrasonic sensors to measure distance and find out the location of the robot relative to the “walls” or in our case, the box’s boundaries.

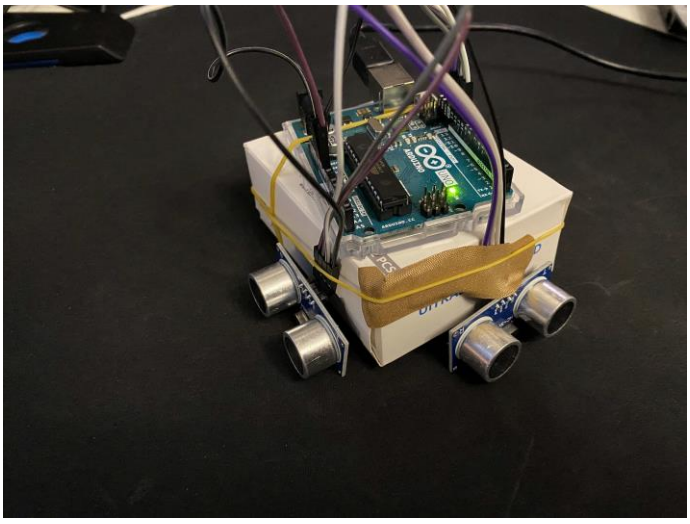


Figure 19 - Localization prototype

The prototype uses 2 ultrasonic sensors 1 for each coordinate in the xy-plane.

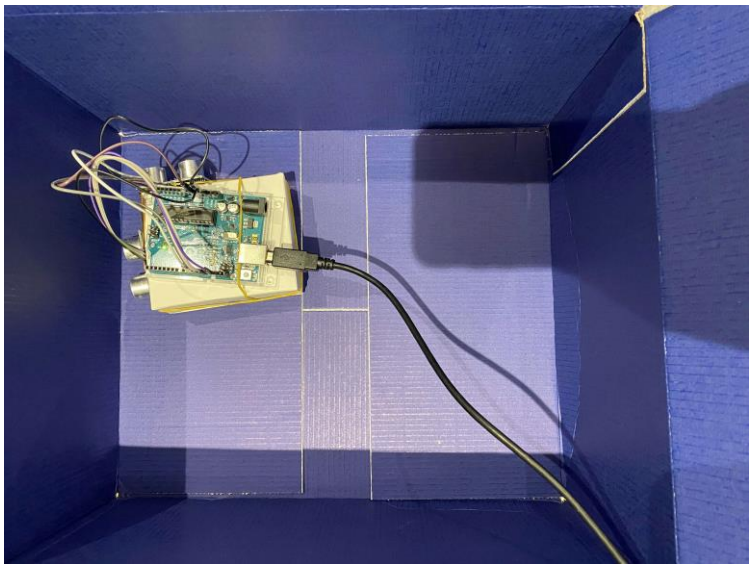


Figure 20 - Box used in localization simulation

We placed the prototype in a box and started the program. Giving us the result shown in the figure below.

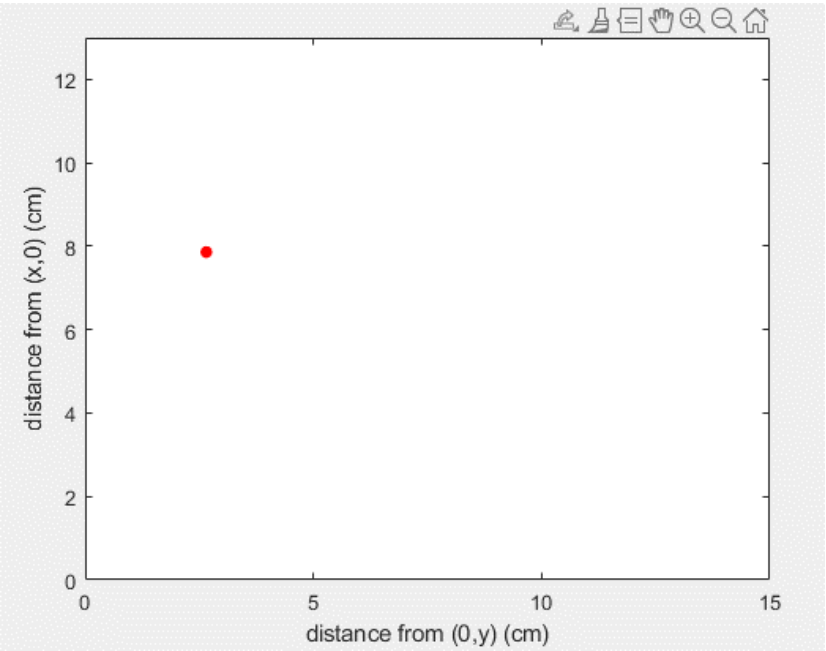


Figure 21 - Resultant MATLAB plot showcasing location.

The red dot represents the prototype, and the boundaries of the plot represent the walls of the box. The plot is dynamic, meaning that the red dot moves with the movement of the prototype. It is not just a snapshot of the location at a particular moment.

CHAPTER – 4 IMPLEMENTATION

4.1 MOTOR CONTROL

Before starting to implement our project. We needed to be able to control the hoverboard. To do that we used an open source hoverboard firmware hack code without modification. The hoverboard is then connected to an ESP8266 and receives commands through it. [15]

4.2 SENSOR FUSION

Sensor fusion is an important concept in robotics, it is the ability to combine data from multiple sources to derive information that is accurate and reliable. Due to the project using a variety of sensors, fusing them together is a hurdle that we have to overcome in order to complete the main functionality. To integrate this into our project, we must make sure that the data we receive is reliable and actually represents what is going on. Furthermore, the robot has three main sensors that can be integrated with one and another.

1) **Hoverboard odometry**
2) **GPS**
3) **Inertial measurement unit (IMU)**

4.2.1 First trial

Before any fusion happens, we had to test to make sure every sensor was working and providing acceptable information to the robot. To start off, the first barebones trial was that we test the hoverboard odometry readings. These readings provide us with the distance (x,y) and the angle that the robot was facing. The robot was placed indoors inside a marked course moving in a straight line to test the odometry readings, if the robot says it has moved 2 meters, we bring a measuring tape and check if it has actually moved the 2 meters. This was done for many iterations until it was clear that the hoverboard provided accurate (up to 15cm error) distance readings. **Fig.22** showcases a simple layout of how this was done.

Figure 22 - Simple odometry.

The next thing was to check the angle readings from the hoverboard. The hoverboard was set in the middle of the room and was set to spin a single complete circle, if it reaches the starting point exactly then that would mean it is perfect. This was done in many iterations and different speeds to figure out if the hoverboard angle can be used. After conducting many tests, the hoverboard angle was not too reliable. At this point, there was no sensor fusion yet as there was only one sensor implemented.

4.2.2 Second trial

In the second trial, we have implemented an IMU using a smartphone. The IMU provides us with many sensor readings like the magnetometer, accelerometer and gyroscope. For our implementation, we only require the yaw to get the angle. Using the IMU to find the yaw, it requires us to use the magnetometer and gyroscope. We use these two sensors together because the magnetometer is excellent and accurate but it suffers from noise. Meanwhile, the gyroscope is great but suffers from drift. Implementing these two together gives us an excellent yaw reading that is consistent. Looking at **Fig.23** below, we can see the gyroscope drift in an example where an IMU was used. [16]

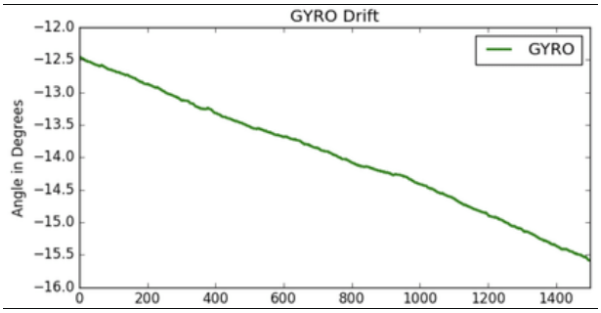


Figure 23 - Example of gyroscope drift. [15]

In this trial, we have used the readings from the smartphone as the angle. This made the robot have a better awareness of the angle and resulted in even better readings. However, due to the magnetometer being easily susceptible to interference there were issues that came up. One of these is that while moving in a straight line, the angle would change drastically if there was interference. As with the previous trial, we have conducted the same experiment where the robot does a complete circle. To our surprise the robot completed a single circle perfectly but doing more than one the drift of the gyroscope kicks in.

GPS was implemented as well in the second trial, and it was used to find the latitude and longitude by using the smartphone built in GPS sensor. In this trial, it was used by the navigation algorithm with no sensor fusion. The robot was given waypoints in a square in the faculty of engineering parking lot and was required to move around in a loop. This worked well but there had to be no metals covering the sky for the GPS signals to reach the phone

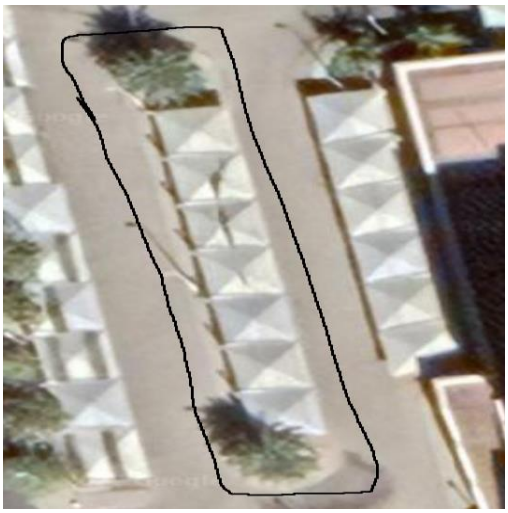


Figure 24 - The path of the robot using GPS.

4.2.3 Third trial

In the third trial, we have implemented sensor fusion. Complimentary filter is a simple technique used where two sensors are fused together to provide a better estimation. It is used as a low-pass high pass filter that complement each other. This was implemented in the GPS and IMU to provide a better heading angle. The complementary filter is as follows:

$$\emptyset = (0.95) * \emptyset_{GPS} + (0.05) * \emptyset_{IMU}$$

After implementing the complementary filter, now we have a steady and accurate angle reading that can be used reliably with no drift. As long as the robot is moving, it will receive GPS signals that update its position and angle. Furthermore, the hoverboard odometry can be fused with GPS using the same complementary filter but with testing we have figured that GPS does the job well enough.

Sensor fusion can be implemented as well for the GPS readings. Our hoverboard has wheel encoders that return their values. From this we have found the PosX and PosY values as stated above. To add on, since we have two readings for distance/location we can fuse them as well using the complementary filter discussed above. In this trial, we have fused GPS and Hoverboard Odometry together. It works by taking the first reading as a GPS reading and then we apply the complementary filter as follows:

$$Lat = (0.975) * Hoverboard_{lat} + (0.025) * GPS_{lat}$$

$$Lon = (0.975) * Hoverboard_{lon} + (0.025) * GPS_{lon}$$

This fusion allows us to move the robot accurately within a small margin of error (~2m) to our waypoints.

Finally, we began setting the waypoints for the robot to follow using the GPS, Hoverboard odometry and IMU sensors. This has led us to the main functionality being completed. The robot is given waypoints and can follow them accurately within GPS margin of error (~3 meters). The readings for the smartphone IMU and GPS are sent via TCP link to the jetson nano and we can implement the other algorithms via that.

4.3 NAVIGATION

One of the main tasks of this robot is to be able to navigate within the KAU campus. Navigation in simple terms means the ability to move from a predefined origin point to another waypoint. To implement this task, we will have to find a way to get some essential readings from the robot as a feedback and base on this feedback we will determine at which direction and with what speed should the robot move. We can now divide the exchanged messages between the main controller and the hoverboard into two main categories as follows:

A) **Commands:** steer and speed.

B) **Feedback:** odometry, orientation.

In the different trials we will see what changes we made on these main categories and why we have taken these changes. Mostly, the commands (steer and speed) will remain the same but the changes are done in the feedback to ensure that the controller based its decision on accurate readings.

4.3.1 First trial

At the first trial we used the default feedback and commands mentioned above to start moving the wheels inside the lab. **Fig.25** shows the general block diagram used for the first trial. In this trial our objectives were simply to publish a speed and steer commands to the robot and then stop it after certain time. Then we have the distance the robot moved and the time it used to move that distance so we can calculate the actual speed and compare it to the one we published. And to compare the coming odometry and orientation feedback with the actual measured values. For this trial the goal was to adjust the speed and steer commands to get the actual values on the ground. For instance if we published a command to move the robot by 1m/s then and we let the robot operate for 10 seconds then it is supposed to 10 meters away from the origin. So at first we calibrated the speed the command to match the actual distance by adjusting the factor int which the speed command is multiplied by. After doing the speed and distance calibration. We started publishing steer commands in which the robot should steer by a given angle published through a command message. As a result of this trial, we were able to ensure that whatever command we publish to the robot it will be accurately executed. However, in this trial we have not yet tested the feedback as we were

comparing the commands we published with the actual distance and orientation measurements.

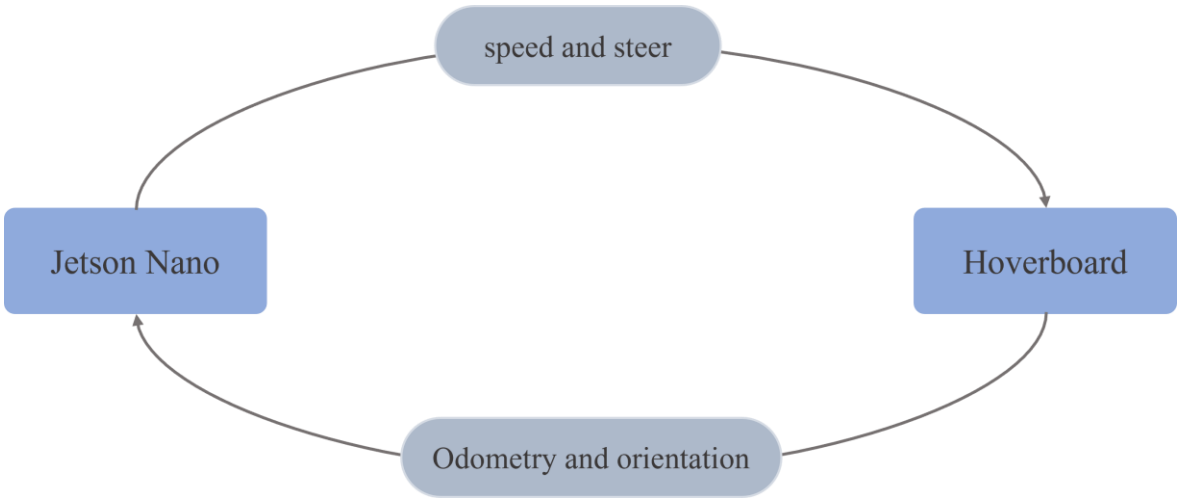


Figure 25 – Data flow of the first trial.

4.3.2 Second trial

After we have ensured that the commands published are well executed in the first trial. We now have to use the feedback coming for the hoverboard and build the next command based on the coming feedback. The modified block diagram for this trial is shown in **Fig.26**. We notice that in the second trial we have to enter a waypoint in the form of (x, y). And the jetson nano is used to generate the speed and steer commands based on the feedback coming from the hoverboard in the form of odometry and orientation. The point of origin (0,0) is considered to be the point where the robot is started. The goal of this trial is to start actually navigating the robot in doors and to check the how accurate the feedback is. For this we have set the waypoints accurately in the lab and measured the actual distances before we start this trial. This way we can see if the feedback is accurate then we can extend this trial operate in larger space and then to operate out-doors. Although after doing this trial we noticed that the accuracy of the odometry was acceptable, but the orientation feedback is not reliable to be used for out-door navigation.

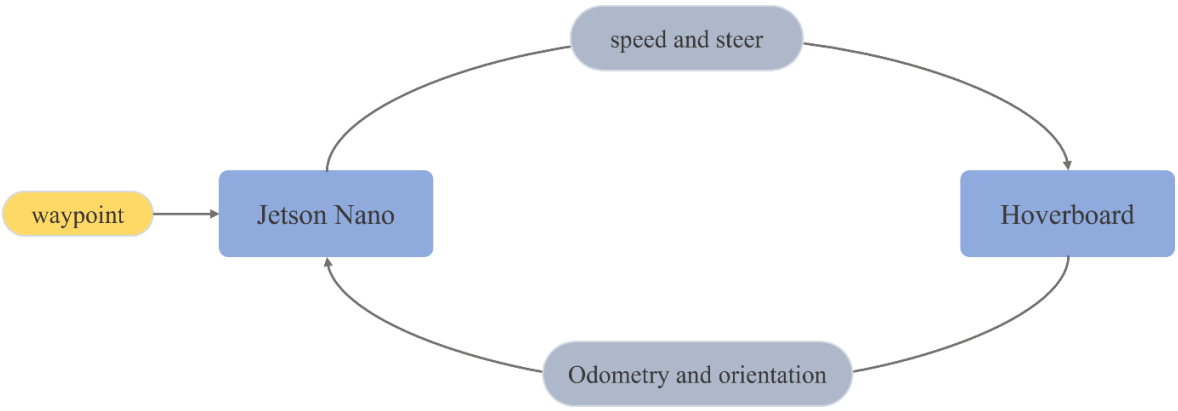


Figure 26 – Data flow of the second trial.

4.3.3 Third trial

In the third trial we decided to improve the accuracy of the orientation and also the odometry so it can be used for out-door navigation and for larger distance. This can be done by using GPS for larger distances to get rid of the accumulative error coming from the odometry and orientation and an IMU readings to improve the orientation readings coming from the hoverboard. However, in this trial we will just take the readings coming from the IMU without using the GPS so we make sure that the fusion of the odometry coming from both sources is valid. For this we decided to fuse the readings coming from the hoverboard as odometry and orientation with the readings we get from the IMU of a smart phone we placed in top of the robot. A question arises here, why a smartphone instead of using dedicated IMU and GPS modules? The answer is that using a smartphone could help if this trial was not accurate enough, we can simply add a network card to improve the GPS reading. And in addition to the fact that we used a TCP link to connect to the both the jetson nano and the ESP8266 module. **Fig.27** shows the block diagram for the third trial after doing what we call sensor fusion to combine the readings coming from different sources. In our case from the hoverboard and the smartphone.

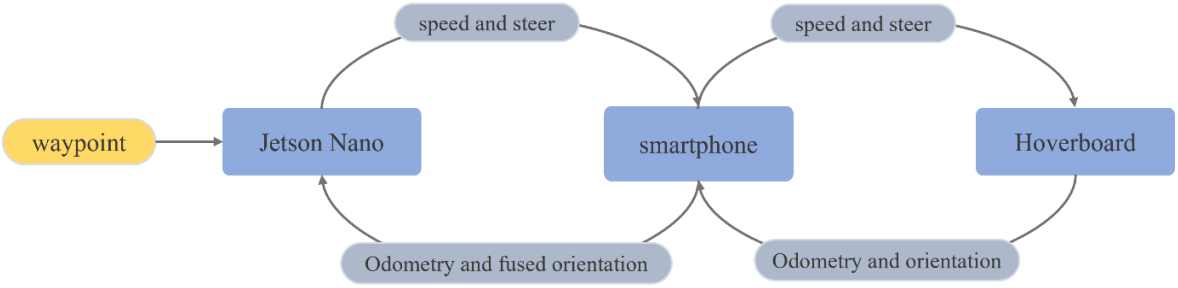


Figure 27 – Data flow of the third trial.

For fusing the readings from the hoverboard and the smartphone we used a complementary filter which was discussed in the sensor fusion section. Since we had an issue with the orientation in the previous trials we decided to add a PD controller, so the robot keeps correcting its angle continuously. The results were much accurate than what we were getting the previous trials. So, we decided to move on with using the smartphone and start using the GPS readings for navigation.

4.3.4 Fourth trial

After using fused readings and the results we got from the previous trial. We decided to move on and start using the GPS available in the smartphone. For that we also made some changes in the format of the waypoint we input to the system. Previously, the waypoints were in the form of (x, y) coordinate but this is not practical outdoors. Thus, for this trial we are using the format (latitude, longitude) for the waypoints instead of the previous format. Now, the Jetson take the current GPS coordinates from the smartphone and then it uses *Haversine Formula*[1] to find the distance and the angle it needed to navigate the robot to. At each iteration the difference in the current orientation and the desired orientation is considered to be the error and enters the PD controller to keep correcting the robot orientation. In this trial the Haversine formula is used to calculate the distance and the bearing (angle) between two GPS points, it takes two points in the form of (latitude, longitude), the distance then calculated as follows:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Where the ϕ1 and ϕ2 are the latitude of point 1 and the latitude of the point 2. And the λ1 and λ2 are the longitude of the point 1 and point 2. The following formula is used to calculate the angle between the two GPS coordinates:

$$\theta = \tan^{-1}\frac{\cos(\phi_1)\sin(\phi_2) - \sin(\phi_1)\cos(\phi_2)\cos(\lambda_2 - \lambda_1)}{\sin(\lambda_2 - \lambda_1)\cos(\phi_2)}$$

The modified flowchart is shown in **Fig.28.** The Haversine Formula is implemented in the jetson nano block.

[1] The *haversine formula* is used to calculate the distance and the bearing (angle) between two GPS points.

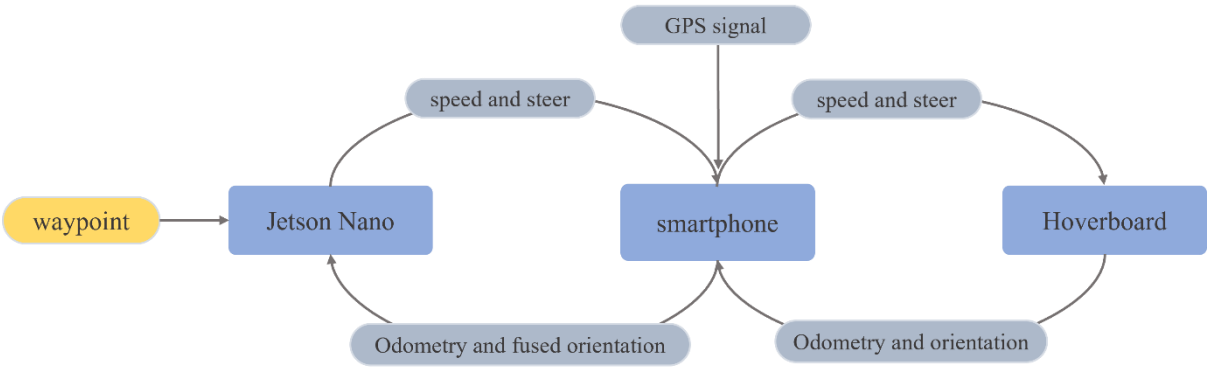


Figure 28 – Data flow of the fourth trial.

At this stage we started doing some optimization for the robot movements. One change we implemented at this stage is while turning the robot to the right angle the robot does not stop but it will gradually increase the speed if the different in angle is less and will decrease the linear speed if the angle difference is high. This can be done following the graph of the function:

$$V = Vmax \times \ e^{-2\,|m|}$$

Where the Vmax is the maximum linear velocity, the robot should move on with and m is the difference between the current orientation of the robot and the desired orientation. **Fig.29** shows the graph of the above function.

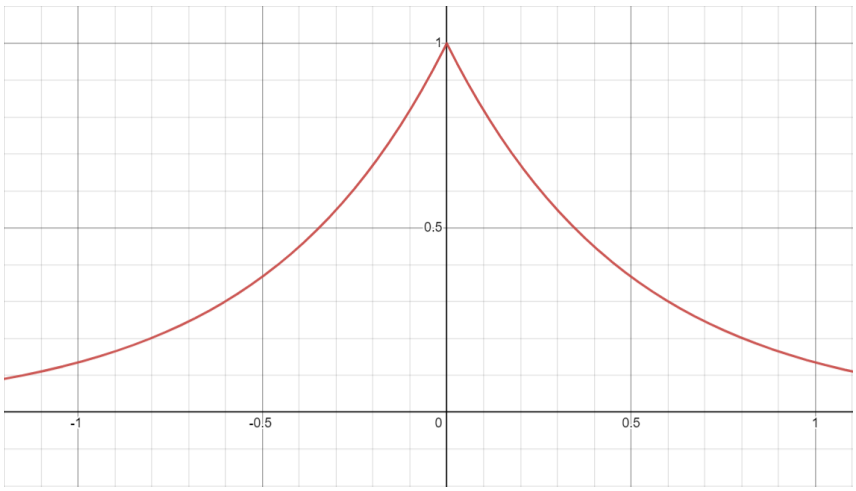


Figure 29 - The graph of the function for gradual change in the linear velocity.

4.4 OBSTACLE AVOIDANCE

Obstacle avoidance in our project is based mainly on depth estimation using a stereo camera. Because we are using a stereo camera, we have two images (right and left), and the way depth estimation works, is by putting those two images on top of each other and then matching each pixel to its counterpart in the other image and then measuring the distance between the pixel in the right image and its counterpart in the left image. Which is what is called disparity. Objects that are closer to the camera have a higher disparity and objects that are further away from the camera have a lower disparity. As shown in the figure, the disparity for the hand is higher than that of the doors. Which means that his hand is closer to the camera than the doors.

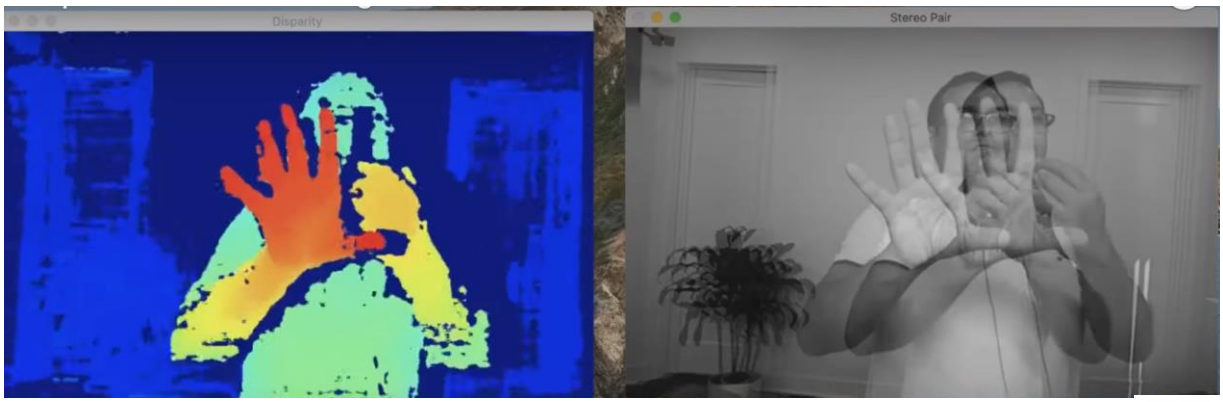


Figure 30 - (a) depth map. (b) overlapped images of stereo pair. [16] a b

But how does this pixel matching occur? Do we go through every single pixel? No, there are some optimization processes that make this matching faster. First of all, we use something called rectification which makes epipolar lines horizontal. In other words, the pixel and its counterpart will be in the same horizontal line in both images. So now we have narrowed down the search to only one line. We also only search the “neighborhood” of the pixel, for even faster processing.

There are certain limitations to this technique in depth estimation. Namely, a limitation in the minimum perceivable depth and difficulty when dealing with smooth featureless surfaces. The first of which is further exacerbated by the optimization done by searching only in the neighborhood of the pixel. Because at closer ranges the corresponding pixel in the other image might be outside of the neighborhood and is therefore not found. At default settings, this blind spot is at around 70 cm away from the camera. The other problem arises due to the difficulty of finding the points correspondences in featureless surfaces. Because

normally each pixel has a single counterpart in the other image, but in the case of surfaces that are all white for example it would be difficult to find the counterpart of each pixel because all the pixels in wall are the same. [17] [18]

As shown in the figure below, we are dividing each frame of the depth map into three sections (middle, right and left). The average intensity for each section is then taken and compared to a threshold to determine whether that section is clear of obstacles or not. Higher intensities indicate that there are obstacles.

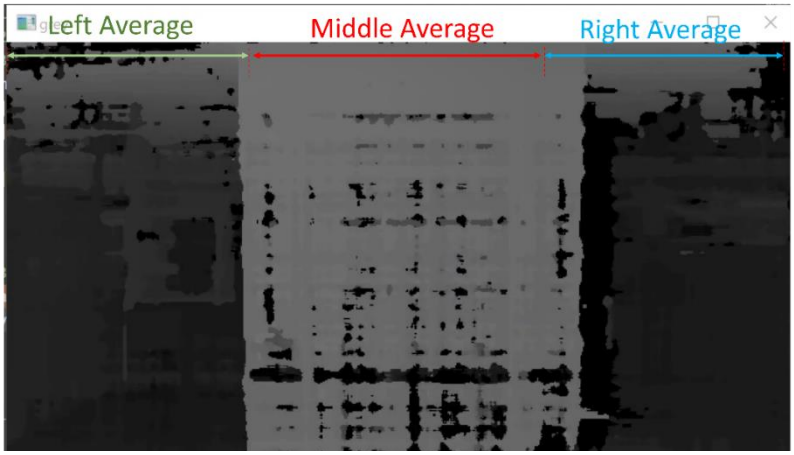


Figure 31 - Depth map sections

4.4.1 First trial

In the preliminary trial/ initial testing, we started by displaying a disparity map and calculating the average intensity for each section of the frame. Although disparity maps are usually color mapped like the figure shown in the previous section, we used a greyscale map to facilitate average intensity calculation. Each average is then compared with a certain threshold to determine if there are any obstacles in the robot's path. In this phase, we were just displaying the average intensities of each section and then printing a sentence based on whether there are any obstacles (based on the threshold) in any of the sections. So "obstacle in the middle" would be displayed if there are any obstacles in the middle section and "obstacle on the right", "obstacle on the left" for obstacles on the right or left, respectively. But these messages had no effect on the robot's movement at the time.

4.4.2 Second trial

In this phase, we started to implement actual obstacle avoidance. We started by thresholding the frames. Which is setting all pixel values above a certain threshold

to 255 (white) and anything below that threshold to 0 (black). The robot then moves with a constant velocity to either the right or to the left if there are obstacles or moves forward if there are no obstacles. The figure below shows thresholded depth maps at different distances. Some optimization was done by changing the intensity threshold which changes the distance of the obstacle that would result in the pixel being white in the depth map. The image below was at a threshold of 70. Setting the threshold to 80 would cause even the depth map of the obstacle at 100 cm away to be almost all black as well.

Figure 32 - Thresholded depth map of obstacle at (a) 100 cm, (b) 120 cm away from the robot. a b

The angular velocity in this phase depending on the obstacle location would be:

- Ang_vel = 0 # if there are no obstacles
- Ang_vel = 0.1 # if there is an obstacle in middle (turns left)
- Ang_vel = -0.1 # if there are obstacles in the middle and on the left (turns right)
- Ang_vel = 0 # if there are obstacles in all directions (the robot stops)

4.4.3 Third trial

Although the approach in the previous trial is easier to implement, the problem is that it is binary. Which means there is either an obstacle or no obstacle, the robot doesn't perceive how close the obstacle is. Which is why in this phase we reverted back to the normal raw depth map which is in greyscale. The depth map is considered "raw" because it is usually mapped to certain color maps similar to the depth map shown in the beginning of the section. In this trial, the intensity of each section is taken into account when calculating the angular velocity. So that the robot turns faster if the obstacle is closer. The figure below shows the same depth map from the previous trial but not thresholded. As shown in the figure the

obstacle is seen by the robot in both cases. Although it would turn faster when the obstacle is closer.

Figure 33 - Depth map of obstacle at (a) 100 cm, (b) 120 cm away from the robot. a b

Note: Avg1: average intensity of left section of the image, avg2: average intensity of middle section, avg3: average intensity of right section.

The flowchart below illustrates the angular velocity and the linear velocity in different cases

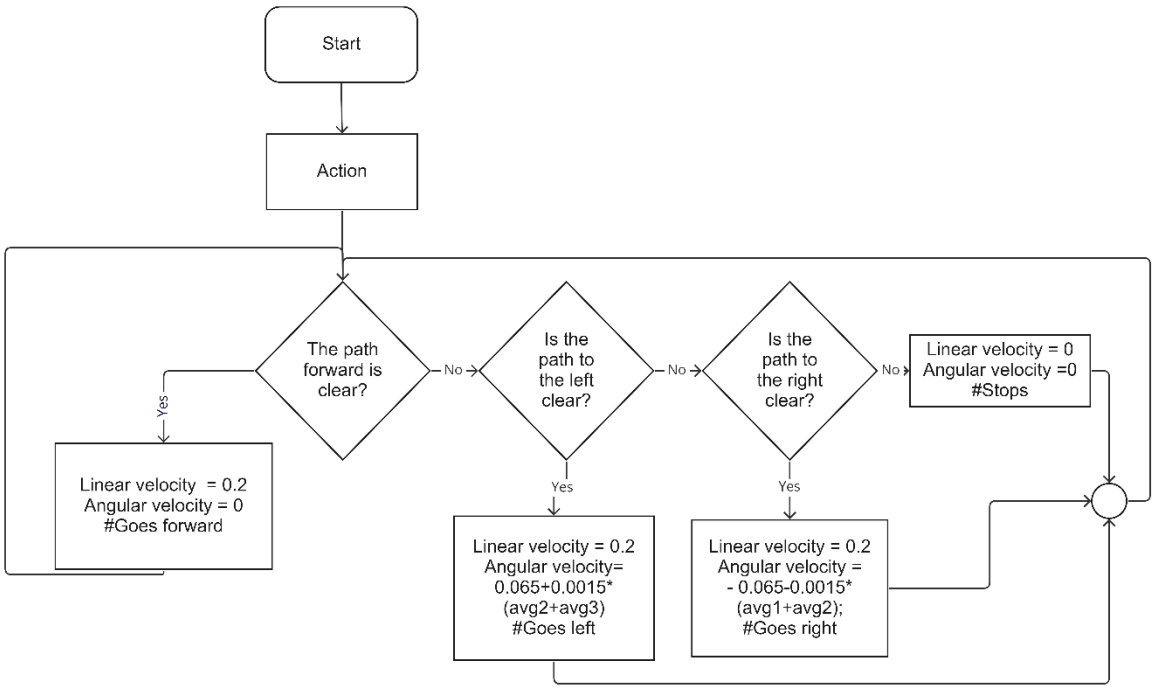


Figure 34 - Obstacle avoidance flowchart

4.5 THE USER MOBILE APPLICATION

The sections discussed above represent the main parts of our project. Which were essential in satisfying the customer's needs. The two coming sections are additional parts that we decided to add to our project to further improve it.

Although our customer agreed on providing a prototype without the need of building the user interface where we would have to make the delivery requests using a simple script to illustrate the working mechanism. However, Engineers do not leave things open, thus we decided to build a functional user interface to serve the purpose of creating new delivery requests and other extra features to our customer. The Application we have built has two modes:

a. **Operational mode**: The user creates a delivery request, selecting the destination and locking the package to ensure the safety of the package and when it arrives the user can unlock the package using the application.
b. **Maintenance Mode**: it enables the maintenance privileges to the engineering team responsible for maintaining the robot, it enables other extra features which will be discussed in the upcoming sections.

4.5.1 Operational mode

The landing page for the users to select the destination is shown in **Fig.35** with steps of selecting a destination point until receiving and unlocking the package hatch. When the user selects the destination, the robot now locks to protect the shipment. The robot then starts navigating to the selected destination and when it is arrived it waits for the user to unlock the shipment and take the package, We have summarized the steps in the following points:

1) Place the shipment inside the package hatch.
2) Select the destination and press START.
3) The robot now is locked.
4) The robot starts navigating toward the selected destination.

5) When the robot arrives, it will be waiting for the user to unlock the hatch and take the shipment.

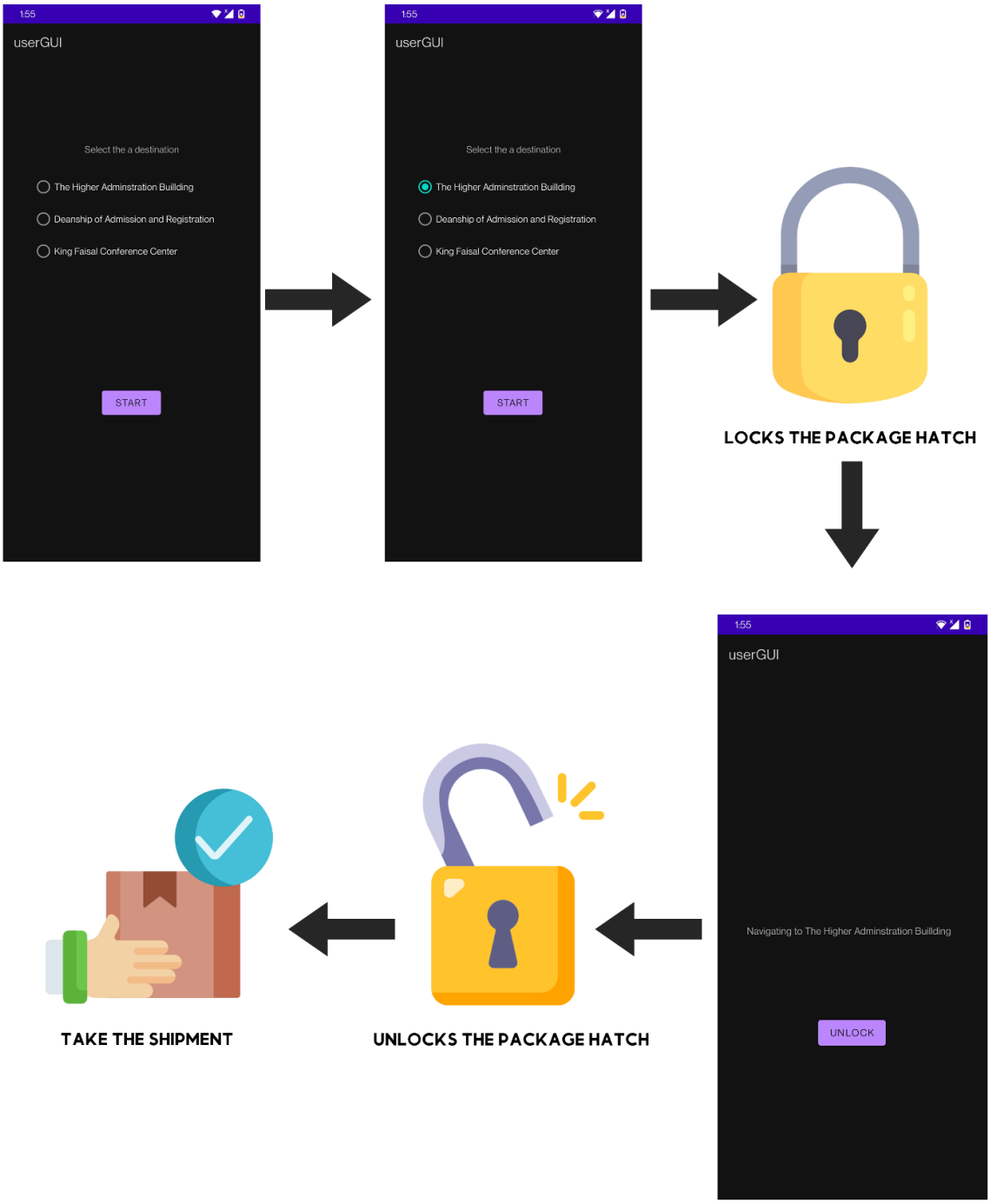


Figure 35 - The steps for delivering a package using the mobile application

4.5.2 Maintenance mode

It is unfair to design and build a great project without thinking of the future maintenance required to ensure that the robot is operation smoothly all the time. Hence, we decided to develop a mode for the maintenance of the KAU Delivery Robot. To enter the maintenance mode, we made it similar to entering the developer mode for android phones, by pressing START button without selecting a destination point for three times and then the maintenance mode is on and the wiring hatch is unlocked. To turn it off just press the cancel button which will close the wiring hatch with it. **Fig.36** shows how to access the maintenance mode through the mobile application.

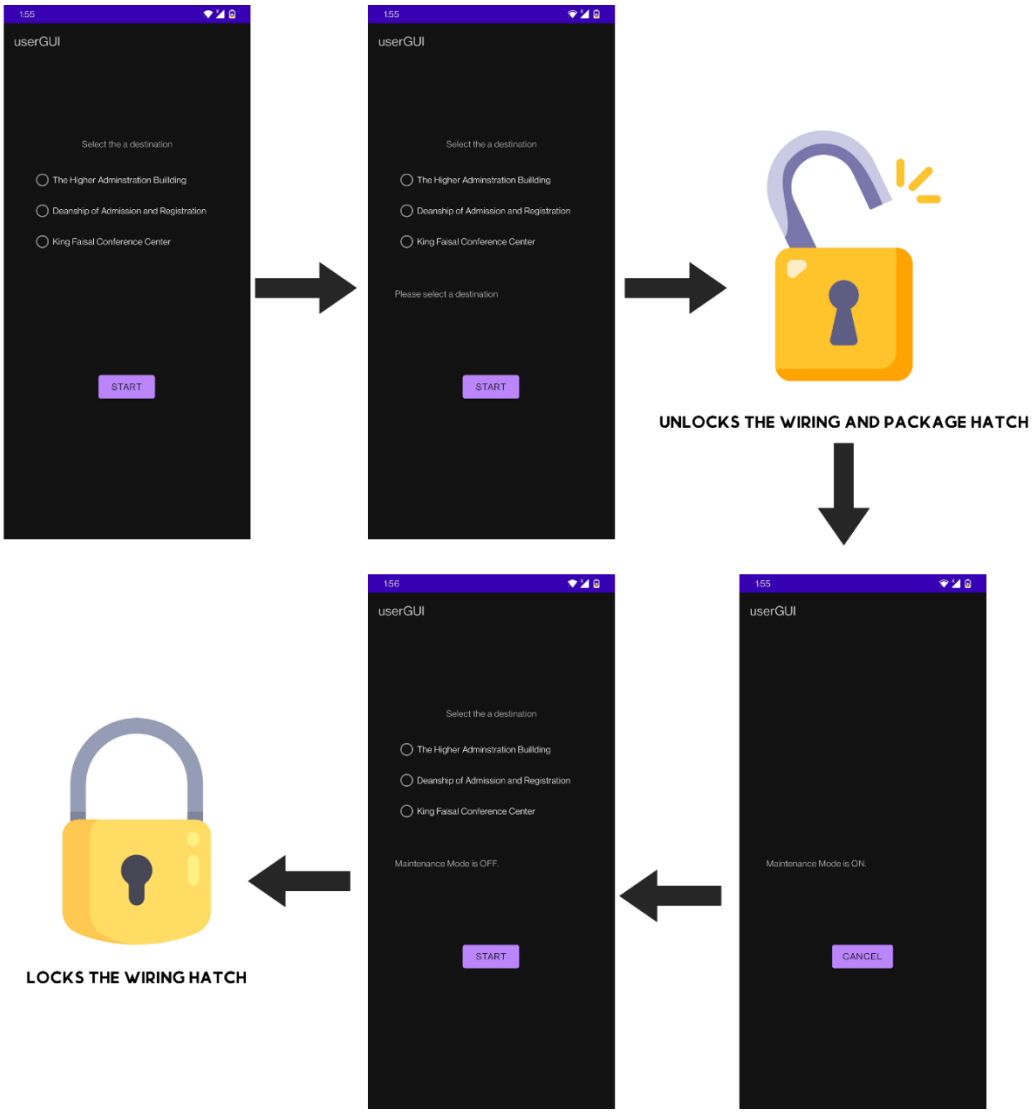


Figure 36 - The steps to enter and exit the maintenance mode

In this mode, several features are available in this mode we will discuss these features individually as follows:

A) Access to the wiring and controllers

One of the main purposes for the maintenance mode is to have access to the hardware in order to perform the maintenance, this also means having access to the robot scrips and code in each component. And also, the access to the components that may need to be replaced for some reasons.

B) Access to trip logs and report

One great feature we have provided is the datalogging for the whole trip taken by the robot. This report provides the path of the robot during the trip and what is supposed path the robot should follow and what is the exact path the robot followed instead. When the trip is started and when it was finished. And the details of the obstacles and where and when these obstacles were found, and to which decision the robot took when these obstacles came. This report also, can help in determining which destinations are the most demanding ones and which are the ones with less delivery requests this can help the marketing department to change the distinctions or provided more similar robots at this station as more requests are coming from there and so on.

C) Create new delivery destinations

The idea of having a maintenance mode does not only mean fixing hardware problems. But it also helps in making the solution more generic solution. A person may ask, what if the operator (our customer) wanted to add more waypoints? Shall we re-design the solution for the scratch? For sure no. A good Engineer always tries to create generic flexible solutions. For our project we provided that in way that the operator can append any new waypoint easily and start covering more areas in the campus. Not just inside the campus the whole project can be used outside the campus by setting all the waypoints outside the campus and this can be easily done inside the maintenance mode.

D) Remove an existing delivery destination

Similar to what we have done in appending new delivery station, we can for instance remove the delivery stations that are not commonly used based on

the reports we provided in the maintenance mode. And replace them with new destinations.

E) Calibrate the robot readings and manual control

Let us imagine a scenario where the engineering team for instance need to take the robot to certain place which is not in the dentation lists let us say they want to make the maintenance somewhere else. Do they have to carry the robot inside someone's car? For sure this is inefficient way of doing it. So, in the maintenance mode we add a manual control access to navigate the robot using joystick application we built or even wireless keyboard to manually control the robot and calibrate the reading if needed.

F) Interrupt a delivery process

For emergency it could help for instance if a user did not collect his shipment for long time and for instance left the campus. Then if no one has access to open the lock except the user then the robot will stay in its place with no use. Thus, we decided to add this feature in case that happened the operator may access the maintenance mode and open the shipment lock and put it in the lockbox and the robot continue serving other customers.

4.5.3 The Shipment Privacy

To ensure the privacy of the shipments our robot delivers, our goal was develop a mechanism to allow the user lock and unlock the shipment hatch whenever the shipment is delivered to its final destination. The locking mechanism implemented in this project is a custom locking mechanism for locking both the shipment hatch and the wiring hatch. The **Fig.37** shows the locking mechanism used for this project. It uses an ESP8266 connected the KAU_DELIVERY_ROBOT network which will be waiting for the user to select a destination and when a destination is selected a jetson nano sends one of the four possible commands to the ESP8266. **Table.8** shows the four possible commands with their descriptions.

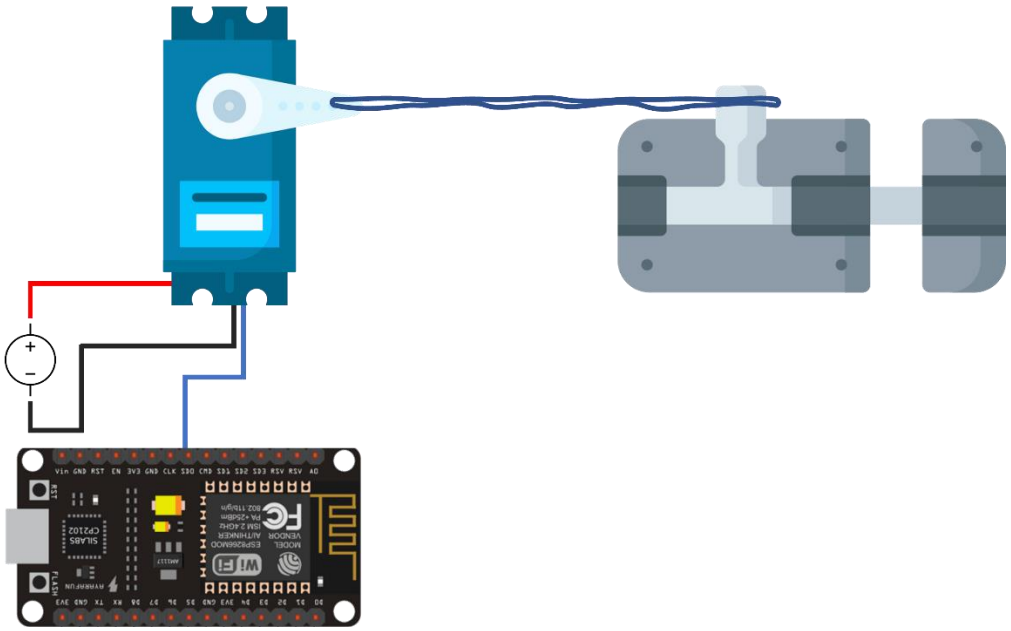


Figure 37 - The locking mechanism used for locking the package and wiring hatches

Table 8 - The four possible locking commands received from the Jetson Nano

Received Command	Interpreted Command	Package Lock	Wiring Lock	Description
1	Start Navigation	Lock	-	Robot is navigating to the desired location, the package hatch locks, while the wiring lock is not changed (locked).
2	Exit Maintenance	-	lock	Robot is leaving the maintenance mode; the wiring hatch locks.
3	Exit Navigation	unlock	-	The package has arrived, the user unlocked the package hatch, the wiring hatch is not changed (locked)
4	Enter Maintenance	unlock	unlock	The robot is in the maintenance mode, both locks must be opened to do the maintenance needed for the robot.

4.5.4 Tamper proof system

A project running in the public is more at risk than other project running indoors with less interacting with outdoor environment. Hence, to protect the KAU Delivery Robot we agreed with our customer to provide a similar mechanism to the one we provided to secure the user shipment but this one is meant to contain and protect all the wiring and tools inside the wiring hatch. The wiring hatch is locked using the same mechanism described in the above section. The goal is to eliminate all the vulnerabilities that could be found by any black hat user . So, when we started designing the robot we made all the wiring in a single place for two reasons the first is to secure this area to guarantee a tamper proof system, the second reason is to make things easier when a maintenance is required. The wiring hatch includes all the following:

- The Jetson Nano
- The ESP8266
- The smartphone
- The Access Point
- The battery
- The wiring cables
- Cooling and ventilation system

And although the robot body frames are made of metal material to handle very strong weights, but this part of the robot where the wiring cables and controllers are, is totally isolated to ensure the safety of the robot and no electrical shortage due to the metal frames.

[1] A black hat user is known terminology for a person uses the user services gain access to the admin privileges with malicious intent to harm the running system.

4.6 HEAT MANAGEMENT

Living in Saudi Arabia, a country located close to the equator line, this means that the country can experience hot weather throughout the year. Temperatures can reach up to 52 degrees Celsius in Jeddah on certain summer days where we conduct our project. Consequently, our prototype which functions in the outside should have measures to manage the heat. [19]

One of the first ideas that is implemented are ventilation holes. Since the robot is moving with speed, air circulation can improve the temperature inside the prototype. Since the air coming in should be at the current atmosphere's temperature, we can assume that the inside of the robot will have new air circulating with lower temperatures. Electronics can produce heat inside the robot's developer segment, but if we circulate air this can help reduce the overall temperature. **Fig.38** showcases the ventilation holes located at the front of the robot marked by a red square.

Figure 38 - Ventilation holes

Second, the materials used in building the robot itself can withstand heat. The robot is built up from metal that is used in outside doors in Jeddah. These metals are exposed to the sun all year round and have no issues withstanding the heat. The robot is also engulfed with yoga mats that can absorb the heat before it reaches the metal casing of the prototype. For the user's comfort, the handles to open the containers are made up from wood so that it does not transfer heat quickly to the hands of the user.

Finally, a case was designed for the mobile phone to include a fan that actively cools the heat coming from the phone. The case was produced in SolidWorks and 3D printed the case to include the measurements for the fan and phone. **Fig.2** showcases the 3D model of the phone case and **Fig.3** showcases the final design with the fan and phone mounted.

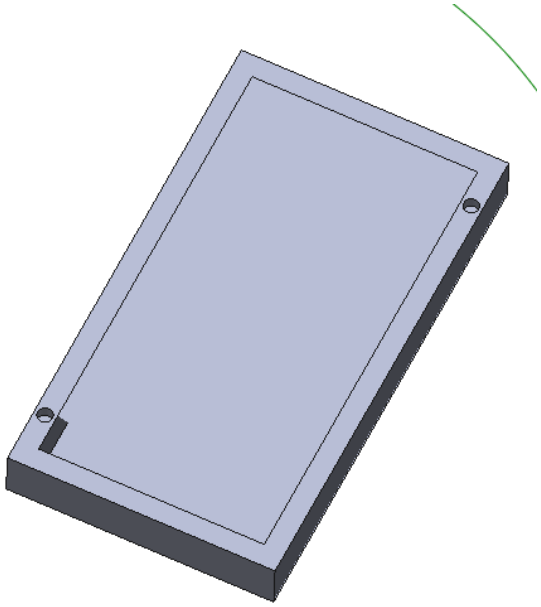


Figure 39 - 3D design of the case

Figure 40 - Phone and fan mounted on the case

4.7 FINAL PRODUCT

To get out final product we needed to combine the functionalities of these different blocks. We also added some improvements like lining the outer surface of the robot with soft material for both aesthetics reasons, and to function as a bumper, reducing collision force.

Figure 41 - Outer lining of robot

For combining sensor fusion and navigation that was already done. Because it was required for navigation itself as it required the fused data. For combining obstacle avoidance with the navigation, the idea is simple. Instead of moving forward in the case of no obstacles, the robot follows the navigation algorithm. Although practically, it wasn't that simple because the navigation code a quite tight loop, meaning commands needed to be executed in a timely manner. To fuse the subsystems, we implemented a dynamic delay and tried to remove as much unnecessary printing, displaying, etc. The figure below illustrates the flowchart of the code after combining the systems. Where Vmax is the maximum velocity set by the designers, diffttheta is the difference between the current yaw and the targeted yaw (towards the waypoint).

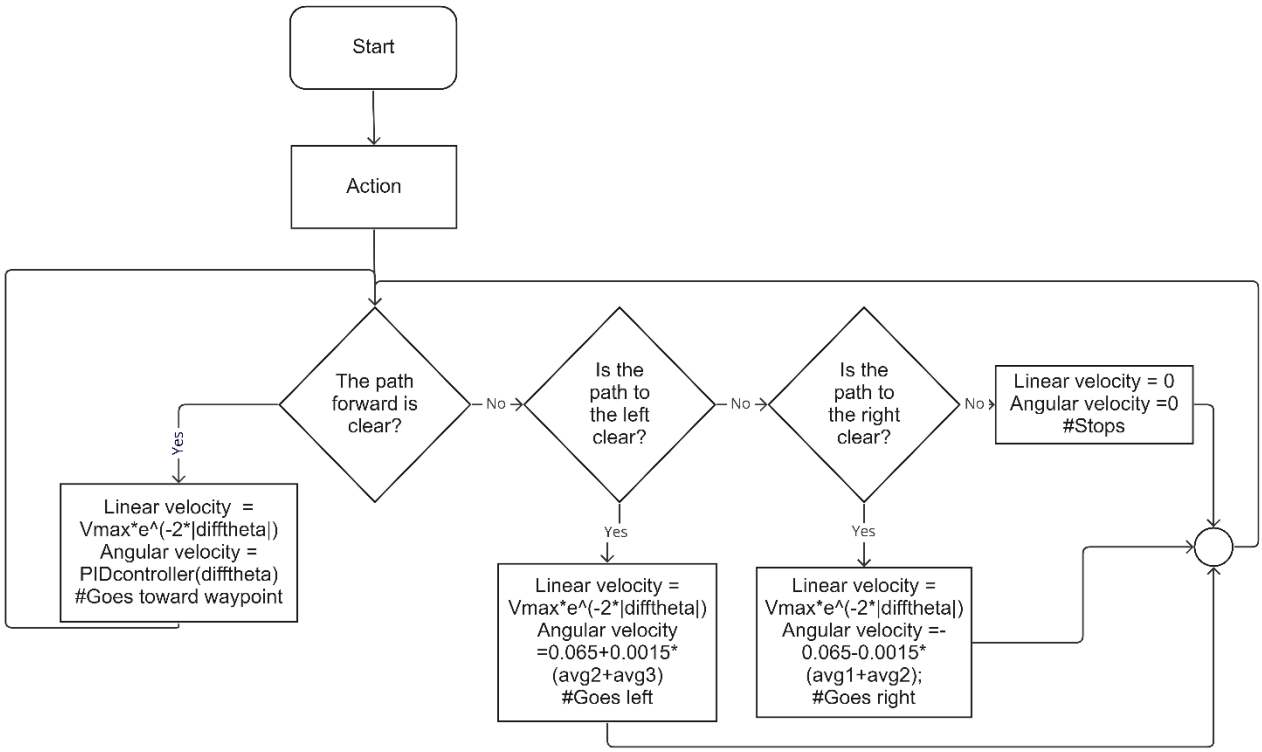


Figure 42 - Final product's flowchart

CHAPTER – 5 RESULTS, DISCUSSION, AND CONCLUSIONS

5.1 RESULTS AND DISCUSSION

5.1.1 Sensor fusion

To summarize, we have an IMU that gives us a yaw reading from the gyroscope and magnetometer and then we have the yaw reading from the GPS. We have fused these two readings together to produce a final yaw that is accurate and reliable via the complementary filter discussed above. The hoverboard odometry has an angle reading that can be fused with the given values above but after some testing it was found out that the reading is not too accurate due to slippage and should not be used.

Furthermore, the IMU magnetometer has weird behavior around many objects. This caused us to remove the magnetometer part from the IMU and rely on the gyroscope. However, the issue of the gyroscope drift was solved by the implementation of the sensor fusion with the GPS so as long as there is a GPS signal coming, the robot has a non-drifting angle reading which is attainable in our implementation. Because sensor fusion and navigation are interconnected, the validation of sensor fusion will be done as a part of the navigation validation experiment.

5.1.2 Navigation

After going the four trials discussed above, in addition to fusing the readings from the smartphone with the reading coming from the hoverboard the results of the navigating the robot between two different GPS coordinates was successfully achieved through the use of the fused odometry data with the GPS readings coming from the smartphone in addition to the fusing the orientation to get better estimate of the current orientation of the robot. This is done by using the IMU data coming from the smartphone. We have considered the state in which the hoverboard states that the robot is moving the other direction while the

smartphone sensors tells that the robot is moving in the opposite direction this is solved by the complementary filter implemented in the sensor fusion section as the robot will correct itself over the time by combining both readings. In addition we have used the feedback coming after fusing the data to plot the path of the robot as in **Fig.43** while navigating to ensure we have accurate feedback the image in the top shows the navigation path before adjusting the complementary filter to ensure the accurate readings while the other picture shows the path of the robot after the adjustment of the complementary filter.

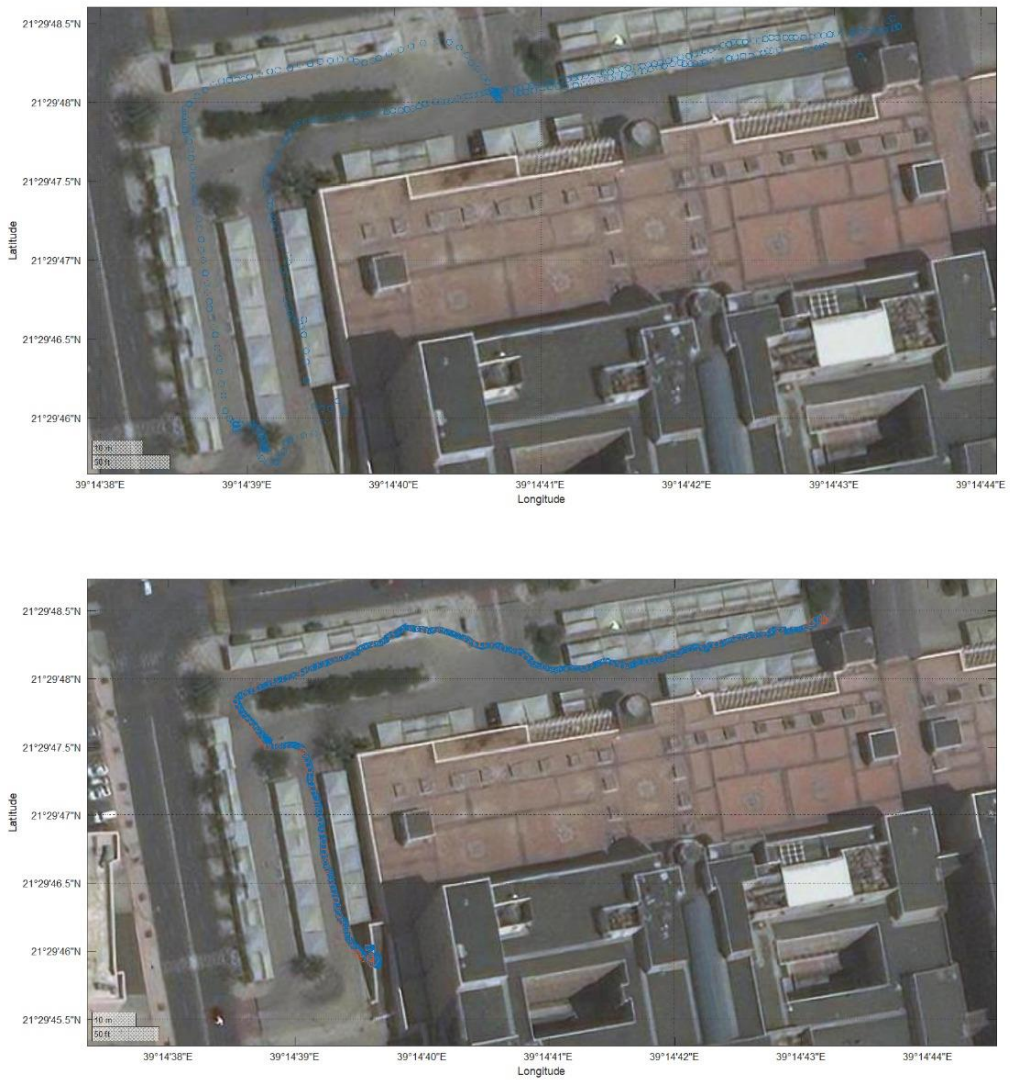


Figure 43 -The path the robot took to navigate in the Engineering parking area with different complementary filters.

5.1.3 Obstacle avoidance

In the plot below the x-axis represents the distance from the robot to the obstacle. Such that the 0 represents the obstacle. Each graph represents a test for a starting distance.

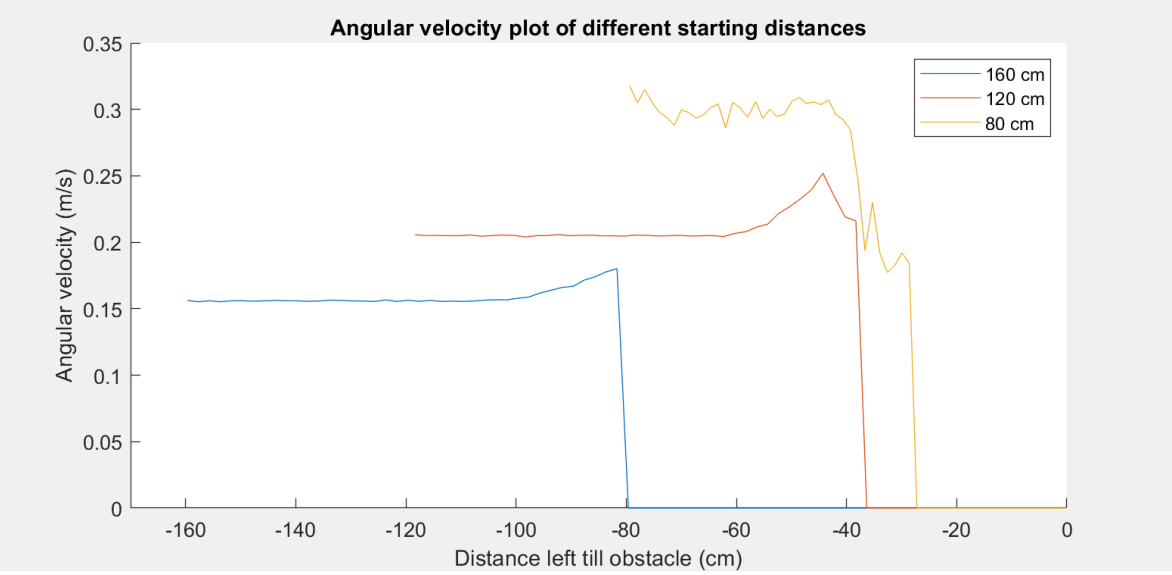


Figure 44 **- Angular velocity vs distance (cm) for different starting points.**

As shown in the figure, for tests that started at a closer distance to the obstacle, the angular velocity was higher. Which is the expected result. Although we noticed that the plot isn't as smooth as we expected. This could be due to the previously mentioned errors; like the background being part of the average and random noise perceived by the camera due to not being able to find the point correspondences for some of the featureless surfaces in the lab.

Table 9 - Results summary of various obstacle avoidance tests

Starting point distance to obstacle (cm)	Orientation (degrees)	Pass/Fail
160	0	Pass
120		Pass
80		Pass
40		Pass
160	-15 (right)	Pass
120		Pass
80		Pass
40		Failed
160	15 (left)	Pass
120		Pass
80		Pass
40		Failed

As shown in the table, the robot passed almost all the obstacle avoidance tests and only failed the tests where we place the obstacle in the camera's blind spot. However, for the 0 degrees test where the robot is directly facing the obstacle, we adjusted the code to measure the variance of each section to try and account for the blind spot. And it was able to pass the test because it stopped and didn't hit the obstacle. Whereas for the tests where it was oriented right or left it wasn't able to detect the obstacle even with the variance and it hit it. This is likely due to the variance itself having a very high variability with time. Although practically, no object should reach the blind spot and that only happened in testing because we deliberately placed the robot very close to the obstacle and then started it. Another case where that might happen is if someone deliberately tries to jump in front of the robot.

While doing the validation experiments, the angular velocity logs were a bit erratic but in testing, the obstacle avoidance program worked fine and avoided the obstacles. This erratic behavior is happening for a short enough time that it isn't actually affecting the behavior of the robot.

5.1.4 Final product

To ensure that our robot is working correctly even after adding all the parts together we had to validate the robot after the amalgamation of the different subsystems.

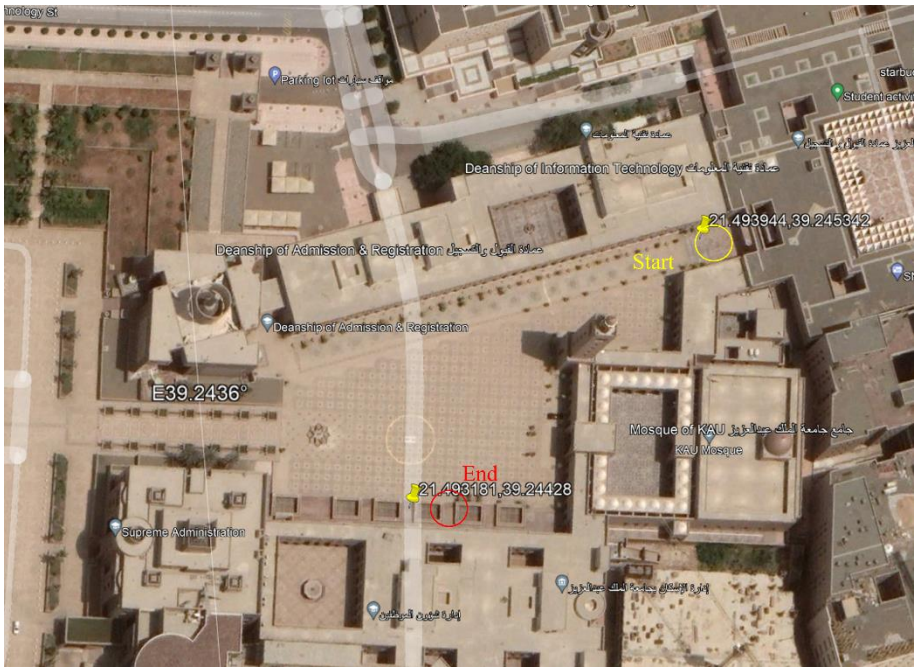


Figure 45 - Start and end of validation experiment

The figure above illustrates the start point and the end point of the experiment. We start by putting the robot at the start point and choosing the endpoint as the destination. Many plant pots are around its path which should serve as a test for the obstacle avoidance part of the project. The robot was able to reach the destination correctly three times on the day of the test which is sufficient enough to say that it is working correctly. There are some practical issues due to the inaccuracies of the GPS sensor. But with calibration, the errors often subside. The figure below illustrates the path that the robot took to get from the start point to the end point. The points might be off due to two possibilities. One is that the satellite image map from google earth isnt exact. The other is that the coordinates are rounded to the fifth decimal.

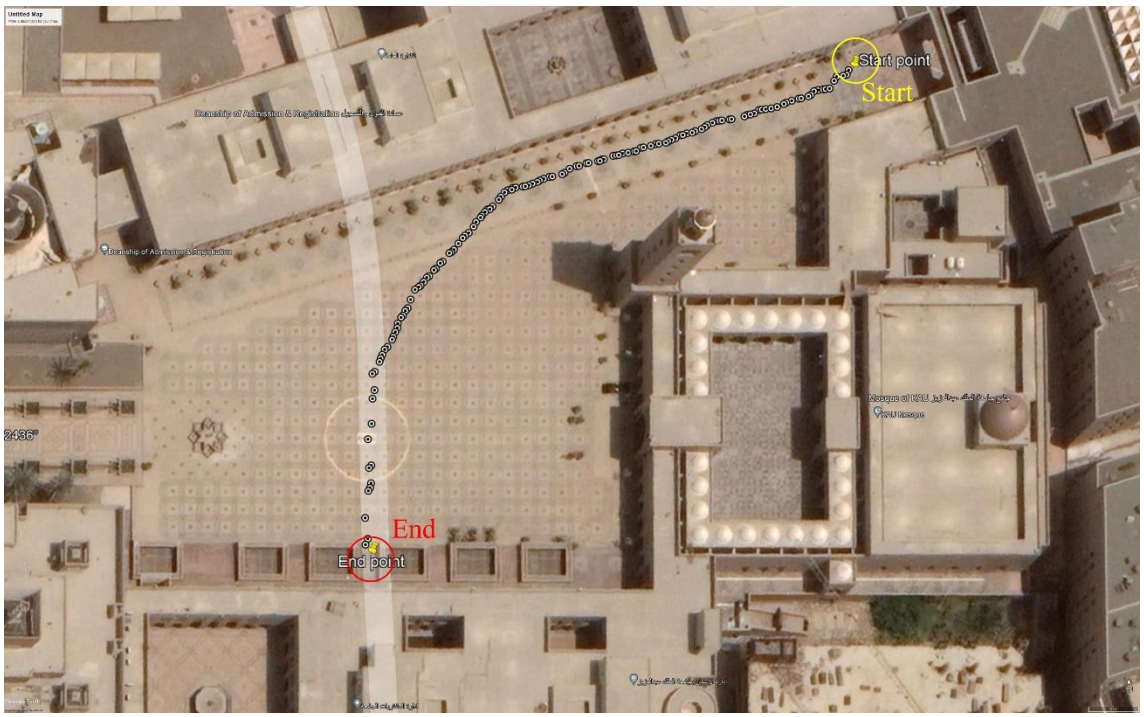


Figure 46 - Path that the robot took in the validation experiment

The figure above illustrates the path the robot took to reach the destination. The robot was coded to log the latitude and longitude for 5 minutes (duration of the trip). The log would then be exported to an excel file at the end of the trip. At first the excel file had 2950 points. Many of which had the same coordinate due to not enough movement. We then filtered out the data by removing values that had very little or no change from the previous value giving us the 138 points plotted above. This was done by converting the excel file into a kml file and loading it to google earth. [20], [21]

5.2 EVALUATION OF SOLUTIONS

5.2.1 Technical Aspects

Thankfully, we were able to satisfy our customer's needs. The robot generally works well enough. And is able to navigate from point A to point B while avoiding obstacles. Although of course there are still some things that could be improved. In this section we'll discuss some of the improvements that could be made in projects build upon this one.

For future versions, our robot currently uses a simple Complimentary filter that does the job. There are many other types of filters that can be included. An example is the Extended Kalman Filter, it uses covariance to dynamically set the percentage weights of each sensor. We have tested it briefly in our implementation and it works using ROS, the node is ready to be used and just requires brief setup. The main advantage of EKF is that it's ideal for systems that continuously change, and it requires very little memory.

One limitation of the Oak-D camera that we used is when dealing with blank texture-less walls and in dark environments. We sometimes used flashlights for the dark environments. One possible solution to both of these problems is actually another Oak-D camera that was released in May 2022 which is why we weren't able to use it seeing as it released during the final months of term-2. The newer pro version uses active stereo depth estimation. Which utilizes an IR laser projector that projects a lot of small dots in front of the camera. Basically adding "texture" to allow for easier pixel correspondence between the two cameras. In addition to an IR (infrared) illumination LED which might be helpful in dark environments.

5.2.2 Environmental Impacts

It is known for years that our vehicles produce harm gases to our environment. However, we cannot refrain from using them. The only thing we can do, is to reduce our usage and to eliminate unnecessary trips. One service our product provides is to reduce the need of using a car in delivering shipments within the campus.

Although our product does not produce green gases, it still has some impacts on the environment. For instance, the baseline design uses a lead acid rechargeable battery which could cause potential threat to people and to the environment if not properly discarded. However, we encourage having a facility that recycles dead batteries to reduce their potential impact on the environment by extract the lead.

In the baseline design, although we could have included other power options that may produce harm gases to our environment we preferred not to, for the sake of our environment. Another way of visualizing the big picture is to consider that the product reduces the need of having duplicated versions of certain things (e.g., textbooks, papers, plastic tools) by allowing these gadgets to be easily transferred from building to another. Which expected to play good for our environment.

5.2.3 Safety Aspects

5.2.3.1 Chemical risk

Because our robot contains batteries, those include a potential chemical risk if they were to be damaged. The batteries are enclosed in a case and the inner surface of the case is lined up with soft material reducing impact.

5.2.3.2 Electrical risk

As with anything electrical the risk of a short circuit is always there. To mitigate this issue the electronic parts are insulated from the robot body by using a non-conducting material and lining the inside of the robot with it.

5.2.3.3 Mechanical Energy Risk

If a person were to jump directly into the blind spot of the robot, the robot might collide with the person. One idea to mitigate this risk is to warn students that the robot cannot see objects that are very close to it, and that they should avoid being in that zone. Another idea is to reduce the overall speed such that even if the robot collided it wouldn’t cause any damage.

5.2.4 Financial Aspects

Table 10 - Final cost analysis

ITEM	Price	Quantity	Total
Jetson nano	$75.00	1	$75.00
Oak-D camera	$200.00	1	$200.00
Android phone	$100.00	1	$100.00
Platform	$100.00	1	$100.00
Battery	$20.00	1	$20.00
Hoverboard	$50.00	1	$50.00
Arduino uno	$20.00	2	$40.00
Wifi adapter	$5.00	1	$5.00
Total cost			$590.00

Robotics is a very interesting subject; the ever-growing revolution of robots is rapidly increasing. Technological advancements in engineering, automation, AI and machine learning will revolutionize the capabilities of robots and they will take over tasks that human laborers use to do.

The robotics industry growth will boost the economy and productivity. Furthermore, it will lead to many new jobs that are yet to exist. Consequently, around 20 million manufacturing jobs may be lost by 2030 to robots. However, robots can be much more efficient than humans at these jobs and can provide better outcomes. Moreover, humans can monitor these robots and make sure the job gets done well.

The effects of job losses have a big variance between countries/regions. In lower skilled regions, the robots can cause twice the unemployment rate by taking up many jobs. This in turn can cause economic and social distress where the regions are already under big issues.

5.2.5 Social Impacts

Students and faculty members going to campus are usually there to finish certain tasks. Whether its studying, attending a class for students or research, teaching for faculty members. But certain things could shift the focus away from the main goal of visiting the campus. Like getting food or getting a book from the campus bookstore, etc.

The robot might boost productivity by allowing students and faculty members to reduce the time wasted on delivering or picking up things from around the campus. However, it might also have a negative effect on people. One of the possible negative effects is that our robot might promote laziness. Because people would get used to receiving things with minimal movement. Despite that, we hope that our robot would be a net positive on society.

5.3 CONCLUSIONS

Based on our information gathering and asking people who work in administration, we believe there is a need for our project even for administrative use alone. In addition to that the robot could serve as a general delivery robot for books, notes, etc. During testing we also noticed quite a bit of interest by people who saw us while we were testing the project. We learned about the implementation of many things including sensor fusion, obstacle avoidance, PID controllers, and navigation. We were able to satisfy the customer's requirements and some of the needs as well. The showcased project is a robot able to deliver packages to different points on campus. Finally, the novelty of this project is that it's a relatively low-cost solution in addition to being a new product in the country. For future work we recommend testing the Oak-D pro and testing EKF filters as they may improve the system. In addition, further analysis may be required for feasibility on a large scale.

REFERENCES

[1] "Google Earth." https://earth.google.com/web/@21.49419087,39.24724142,38.13685635a,1289.6303686d,35y,8.9851946h,40.03009141t,0r (accessed May 23, 2022).

[2] "What is Transmission Control Protocol (TCP)? Definition from SearchNetworking." https://www.techtarget.com/searchnetworking/definition/TCP (accessed May 23, 2022).

[3] "Guidelines For Robotics Safety | Occupational Safety and Health Administration." https://www.osha.gov/enforcement/directives/std-01-12-002 (accessed May 23, 2022).

[4] J. Waykule, "(PDF) Smart Drone Delivery System." https://www.researchgate.net/publication/344513432_Smart_Drone_Delivery_System (accessed Nov. 30, 2021).

[5] R. Tasaki, M. Kitazaki, J. Miura, and K. Terashima, "Prototype design of medical round supporting robot 'Terapio,'" *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2015-June, no. June, pp. 829–834, Jun. 2015, doi: 10.1109/ICRA.2015.7139274.

[6] A. Gujarathi *et al.*, "Design and Development of Autonomous Delivery Robot," Mar. 2021, Accessed: Nov. 30, 2021. [Online]. Available: https://arxiv.org/abs/2103.09229v1.

[7] "This Raspberry Pi Delivery Robot is Controlled by Live Stream Chat | Tom's Hardware." https://www.tomshardware.com/news/raspberry-pi-delivery-robot-droiid (accessed Nov. 30, 2021).

[8] "OAK-D – Luxonis." https://shop.luxonis.com/products/oak-d (accessed May 23, 2022).

[9] "OAK-D — DepthAI Hardware Documentation 1.0.0 documentation." https://docs.luxonis.com/projects/hardware/en/latest/pages/BW1098OAK.html (accessed May 23, 2022).

[10] "The importance of IMU Motion Sensors - CEVA's Experts blog." https://www.ceva-dsp.com/ourblog/what-is-an-imu-sensor/ (accessed May 23, 2022).

[11] "Measure linear acceleration along X, Y, and Z axes - Simulink."

https://www.mathworks.com/help/supportpkg/android/ref/accelerometer.html (accessed May 23, 2022).

[12] “Measure magnetic field along X, Y, and Z axes - Simulink.” https://www.mathworks.com/help/supportpkg/android/ref/magnetometer.html (accessed May 23, 2022).

[13] “Global Positioning System (GPS) Sensors: Receivers with antennas.” https://www.campbellsci.eu/gps (accessed May 23, 2022).

[14] “Measure GPS latitude, longitude, and altitude - Simulink.” https://www.mathworks.com/help/supportpkg/android/ref/locationsensor.html (accessed May 23, 2022).

[15] “GitHub - EFeru/hoverboard-firmware-hack-FOC: With Field Oriented Control (FOC).” https://github.com/EFeru/hoverboard-firmware-hack-FOC (accessed May 23, 2022).

[16] C. K. Mummadi *et al.*, “Real-time and embedded detection of hand gestures with an IMU-based glove,” *Informatics*, vol. 5, no. 2, Jun. 2018, doi: 10.3390/INFORMATICS5020028.

[17] “Depth Estimation using OAK-D and it’s other variant | OAK Series - YouTube.” https://www.youtube.com/watch?v=fx4uXLQWWi4&t=905s (accessed Apr. 21, 2022).

[18] “Depth perception — DepthAI documentation | Luxonis.” https://docs.luxonis.com/en/latest/pages/depth/ (accessed Apr. 21, 2022).

[19] “9 Countries Have Recorded Hottest-Ever Temps This Year.” https://www.treehugger.com/countries-have-recorded-hottest-ever-temps-this-year-4858653 (accessed May 21, 2022).

[20] “Google Earth.” https://earth.google.com/web/ (accessed Nov. 30, 2021).

[21] “Excel To KML.” https://www.earthpoint.us/exceltokml.aspx#GoogleEarthIcons (accessed May 23, 2022).

APPENDIX – A: VALIDATION PROCEDURES

This page is left blank on purpose to add the documents in the next pages. Note that the obstacle avoidance report also contains appendix A in it.

بسم الله الرحمن الرحيم

King Abdulaziz University - Faculty of Engineering – EE-499

Senior Design Project

Report#6

Validation Experiment

Name	ID
Muhannad Saeed Alghamdi	**1846525**

Introduction

The navigation is one of the main tasks for the On-campus Delivery Robot. The robot is expected to move within the campus using the GPS coordinates. In this experiment we are going to test the distance and bearing (angle) of the moving robot and compare it to the calculated distance and bearing using the Haversine formula. For this experiment, we will fix the origin point (constant) and keep changing the waypoints (independent variable). For each point we will measure the distance and bearing (dependent variables) and compare them to the ones calculated using the Haversine formula. The experiment is a part of the validation procedure to satisfy the customer need of navigating the robot out-doors within the KAU campus.

Objectives

The main objective for this experiment Is to test the implementation of the *Haversine formula* [(1)]. for calculating the distance and bearing between two GPS points in the form of (latitude, longitude) and to make sure that the robot uses the calculated bearing and distance for its motion.

Safety/ Environmental issues

Although this test was done outdoors. To ensure the safety of the enviroment and that no harm is done to pedestrians. The robot was supervised at all times and we had two different emergency brakes and we would walk next to the robot in case of any malfunctions.

Tools

- 1 x Electric Hoverboard
- 1 x esp8266 module
- 1 x android smartphone with its GPS enabled
- 1 x access point
- 1 x jetson nano with python installed
- Set of wires
- 1 x meter
- 1 x protractor
- 1 x pen/marker

The Block Diagram

Fig.1 shows the block diagram for the connection needed for doing this experiment. The connections detailed in the work plan section above as it is required before starting the data collection for this experiment.

(1) The *haversine formula* is used to calculate the distance and the bearing (angle) between two GPS points, it is defined in the mathematical formulas section.

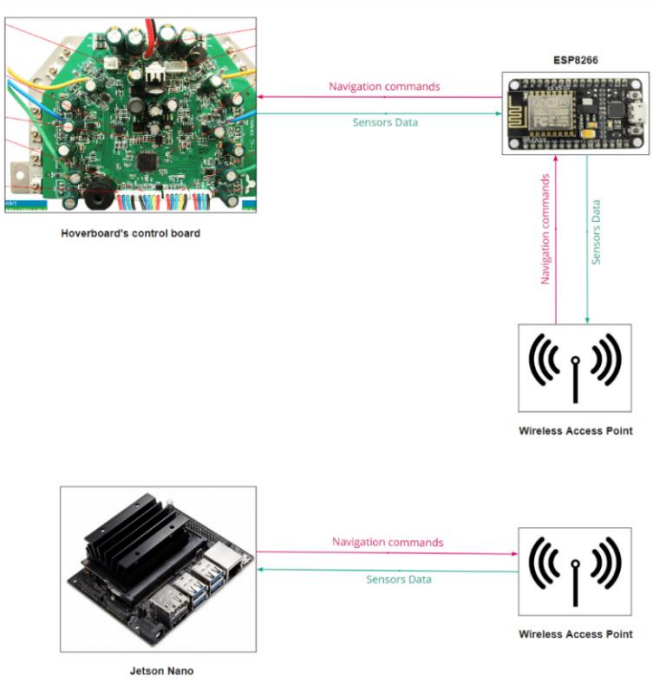


Figure 1 the block diagram for the connection need for the experiment

Work Plan

1) First step is to program the hoverboard to receive the commands through the UART.
2) Connect the RX of the esp8266 to the TX of the hoverboard, and the TX of the esp8266 with the RX of the hoverboard and the common ground
3) Program the ESP8266 to provide a TCP server and receive the commands over the TCP link and pass it to the hoverboard.
4) Program the android phone to send the current GPS latitude and longitude to the jetson over the TCP link and to receive commands from jetson and pass it to the esp8266.
5) On Google map, select waypoints and the origin to conduct this test.
6) In the jetson nano, Implement the *Haversine formula* on a python script to calculate the bearing and distance between the origin and the desired waypoints.
7) In the jetson nano, input the waypoints and origin points to the python script to calculate the bearing and distance.
8) In the jetson nano, modify the python script to connect to the phone to send the commands over the TCP link based on the calculated distance and bearing from previous step.
9) On the robot, use the pen/marker to mark the center of the robot.
10) To fill the bearing column, set the maximum linear speed to zero so the robot just points to the waypoint and measure the angle using the protractor start where east is 0^{o}.
11) To measure the distance, set the maximum angular speed to zero so the robot moves with distance that was calculated on the step 6.
12) Place the robot on the origin point selected and start the python script.
13) Write down the result and calculate the *error* [(2)].

(2) The error formula is defined in the mathematical formulas section.

Waypoints selection

For this experiment we have selected 16 waypoints inside the KAU campus that are away from an origin point. We considered selecting waypoints that not too far from the origin in order to measure the distance and angle with tools mentioned in tools section. **Fig.2** illustrates the selected waypoints and the origin on the map.

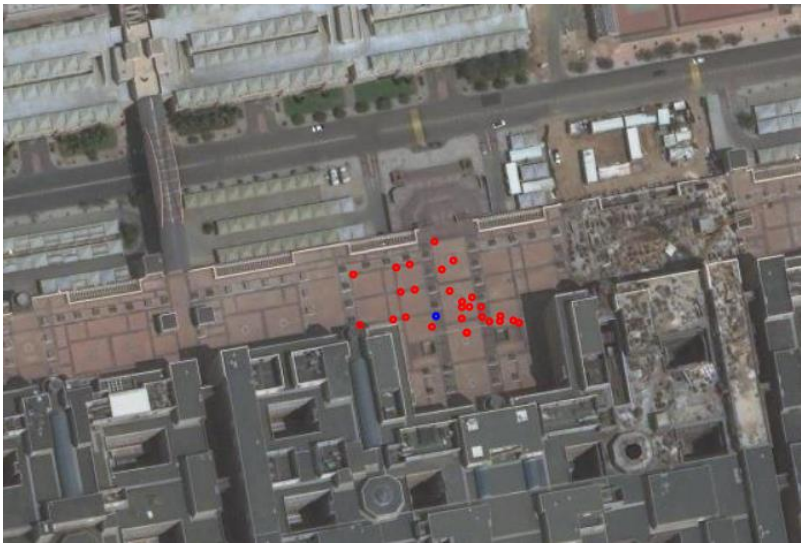


Figure 2 the selected waypoints and the origin points on the campus map.

Mathematical Formulas:

I) Haversine formula

The Haversine formula is used to calculate the distance and the bearing (angle) between two GPS points, it takes two points in the form of (latitude, longitude), the distance then calculated as follows:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Where the ϕ_1 and ϕ_2 are the latitude of point 1 and the latitude of the point 2. And the λ_1 and λ_2 are the longitude of the point 1 and point 2. The following formula is used to calculate the angle between the two GPS coordinates:

$$\theta = \tan^{-1}\frac{\cos(\phi_1)\sin(\phi_2) - \sin(\phi_1)\cos(\phi_2)\cos(\lambda_2 - \lambda_1)}{\sin(\lambda_2 - \lambda_1)\cos(\phi_2)}$$

II) Error formula

In this experiment we need to calculate the error. Between the measured and calculated values of both the bearing and distance:

$$error = \frac{measured - caluated}{caluated}$$

Data Collection

For the data collection, we have conducted the experiment three times using the same waypoints to ensure higher precision of data collected. The collected data were used for analysis and conclusion.

I) First Test

iteration	origin		waypoint		calculated		measured		Distance error	Bearing error
	latitude	longitude	latitude	longitude	distance	bearing	distance	bearing		
1	21.49655091	39.24645493	21.49654745	39.24634252	29.8254	-0.0562	29.16	6.23082543	-2%	0%
2	21.49655091	39.24645493	21.49653934	39.24629503	11.0259	1.0738	11.21	1.085594795	2%	1%
3	21.49655091	39.24645493	21.49663485	39.24632304	32.1322	-0.0786	31.09	6.20639082	-3%	0%
4	21.49655091	39.24645493	21.49669524	39.24614698	13.4407	-0.4941	12.81	5.841617006	-5%	1%
5	21.49655091	39.24645493	21.49671952	39.24630705	17.7544	-0.015	18.42	6.103416394	4%	-3%
6	21.49655091	39.24645493	21.49673011	39.24635672	13.4795	-0.4846	14.51	6.119124357	8%	6%
7	21.49655091	39.24645493	21.49654273	39.246551	35.6742	2.675	32.98	2.885029254	-8%	8%
8	21.49655091	39.24645493	21.49651387	39.24643945	11.4831	0.5053	10.24	0.53581608	-11%	6%
9	21.49655091	39.24645493	21.4966009	39.24655205	20.673	-0.0994	18.67	5.827654372	-10%	-6%
10	21.49655091	39.24645493	21.4968096	39.24644944	29.5577	-3.0248	26.76	3.56919832	-9%	10%
11	21.49655091	39.24645493	21.49653415	39.24669281	13.1209	2.2513	11.23	2.05425253	-14%	-9%
12	21.49655091	39.24645493	21.49649444	39.24657021	18.1003	1.4462	16.07	1.279326342	-11%	-12%
13	21.49655091	39.24645493	21.49653246	39.24665376	16.5933	-3.064	14.32	3.597123588	-14%	12%
14	21.49655091	39.24645493	21.49658316	39.24657929	22.3664	2.0424	18.91	2.330014551	-15%	14%
15	21.49655091	39.24645493	21.49655105	39.24669462	24.198	2.2552	22.18	2.79950812	-8%	24%
16	21.49655091	39.24645493	21.49653584	39.24674275	10.5488	0.3373	8.84	0.254818071	-16%	-24%

II) Second test

iteration	origin		waypoint		calculated		measured		Distance error	Bearing error
	latitude	longitude	latitude	longitude	distance	bearing	distance	bearing		
1	21.49655091	39.24645493	21.49654745	39.24634252	29.8254	-0.0562	29.07	6.248278722	-3%	0%
2	21.49655091	39.24645493	21.49653934	39.24629503	11.0259	1.0738	11.26	1.052433539	2%	-2%
3	21.49655091	39.24645493	21.49663485	39.24632304	32.1322	-0.0786	31.35	6.145304296	-2%	-1%
4	21.49655091	39.24645493	21.49669524	39.24614698	13.4407	-0.4941	12.61	5.859070299	-6%	1%
5	21.49655091	39.24645493	21.49671952	39.24630705	17.7544	-0.015	18.23	6.033603224	3%	-4%
6	21.49655091	39.24645493	21.49673011	39.24635672	13.4795	-0.4846	14.61	6.085963102	8%	5%
7	21.49655091	39.24645493	21.49654273	39.246551	35.6742	2.675	33.2	2.972295716	-7%	11%
8	21.49655091	39.24645493	21.49651387	39.24643945	11.4831	0.5053	10.14	0.46774824	-12%	-7%
9	21.49655091	39.24645493	21.4966009	39.24655205	20.673	-0.0994	18.47	5.775294495	-11%	-7%
10	21.49655091	39.24645493	21.4968096	39.24644944	29.5577	-3.0248	25.69	3.565707662	-13%	9%
11	21.49655091	39.24645493	21.49653415	39.24669281	13.1209	2.2513	11.2	2.057743188	-15%	-9%
12	21.49655091	39.24645493	21.49649444	39.24657021	18.1003	1.4462	15.35	1.286307659	-15%	-11%
13	21.49655091	39.24645493	21.49653246	39.24665376	16.5933	-3.064	14.35	3.591887601	-14%	12%
14	21.49655091	39.24645493	21.49658316	39.24657929	22.3664	2.0424	19.21	2.331759881	-14%	14%
15	21.49655091	39.24645493	21.49655105	39.24669462	24.198	2.2552	19.97	2.666863097	-17%	18%
16	21.49655091	39.24645493	21.49653584	39.24674275	10.5488	0.3373	12.24	0.274016693	16%	-19%

III) Third test

iteration	origin		waypoint		calculated		measured		Distance error	Bearing error
	latitude	longitude	latitude	longitude	distance	bearing	distance	bearing		
1	21.49655091	39.24645493	21.49654745	39.24634252	29.8254	-0.0562	28.76	6.192428186	-4%	-1%
2	21.49655091	39.24645493	21.49653934	39.24629503	11.0259	1.0738	11.2	1.087340124	2%	1%
3	21.49655091	39.24645493	21.49663485	39.24632304	32.1322	-0.0786	31.31	6.138322979	-3%	-1%
4	21.49655091	39.24645493	21.49669524	39.24614698	13.4407	-0.4941	12.71	5.893976884	-5%	2%
5	21.49655091	39.24645493	21.49671952	39.24630705	17.7544	-0.015	18.54	6.014404602	4%	-4%
6	21.49655091	39.24645493	21.49673011	39.24635672	13.4795	-0.4846	14.71	6.049311187	9%	4%
7	21.49655091	39.24645493	21.49654273	39.246551	35.6742	2.675	33.23	2.928662485	-7%	9%
8	21.49655091	39.24645493	21.49651387	39.24643945	11.4831	0.5053	10.22	0.46774824	-11%	-7%
9	21.49655091	39.24645493	21.4966009	39.24655205	20.673	-0.0994	18.37	5.719443959	-11%	-8%
10	21.49655091	39.24645493	21.4968096	39.24644944	29.5577	-3.0248	25.54	3.488913175	-14%	7%
11	21.49655091	39.24645493	21.49653415	39.24669281	13.1209	2.2513	11.22	1.987930018	-14%	-12%
12	21.49655091	39.24645493	21.49649444	39.24657021	18.1003	1.4462	15.42	1.268854366	-15%	-12%
13	21.49655091	39.24645493	21.49653246	39.24665376	16.5933	-3.064	14.37	3.537782394	-13%	10%
14	21.49655091	39.24645493	21.49658316	39.24657929	22.3664	2.0424	19.27	2.394591734	-14%	17%
15	21.49655091	39.24645493	21.49655105	39.24669462	24.198	2.2552	20.1	2.693043036	-17%	19%
16	21.49655091	39.24645493	21.49653584	39.24674275	10.5488	0.3373	12.42	0.275762022	18%	-18%

Results Analysis

In this section, we will use the data collected from the previous section to plot the error in distance and bearing it each iteration. For this, we will use Excel to plot the data. **Fig.3** shows the plots for the first test conducted for both distance error and angle error over the iterations.

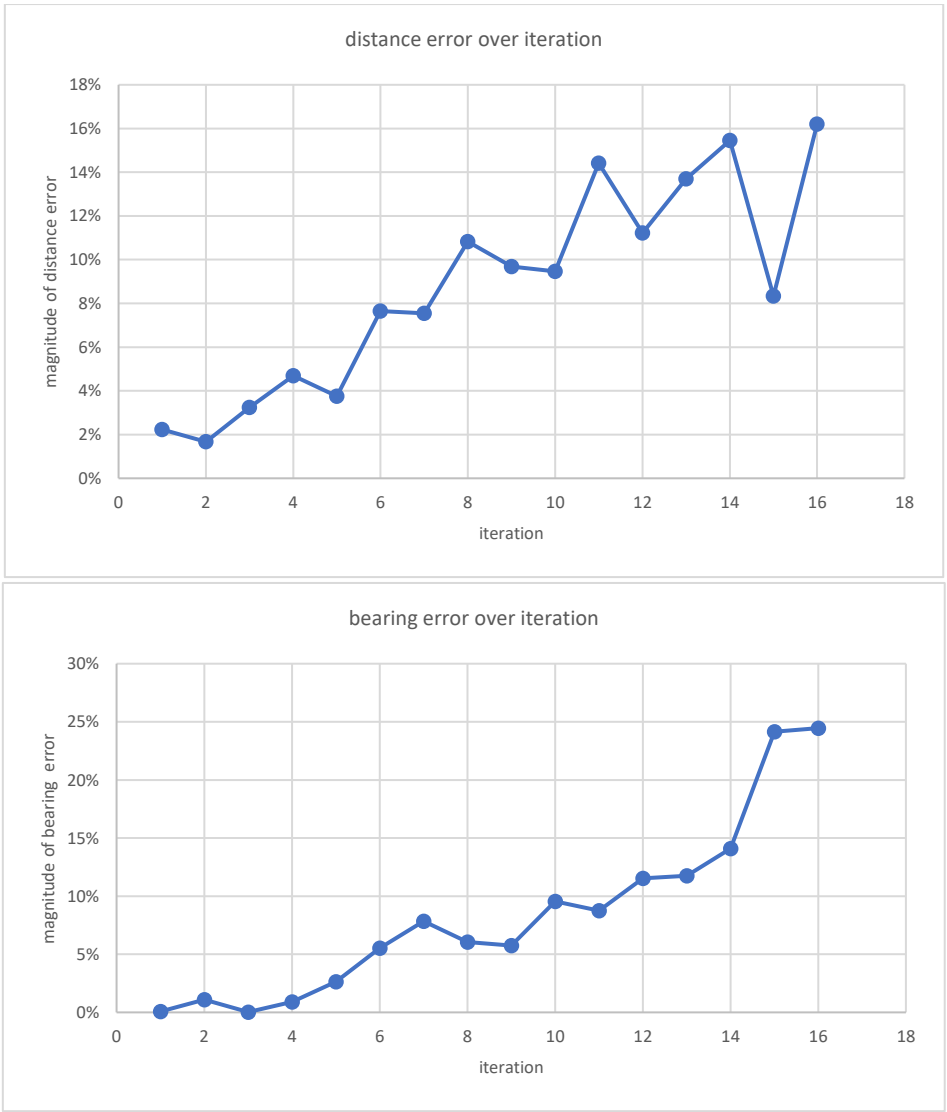


Figure 3 the distance and bearing error for the first test over the iterations

The second and third test results were also plotted using Excel to graph the error in distance and in angle over the iterations. **Fig.4** shows the plot for the second test distance and angle errors.

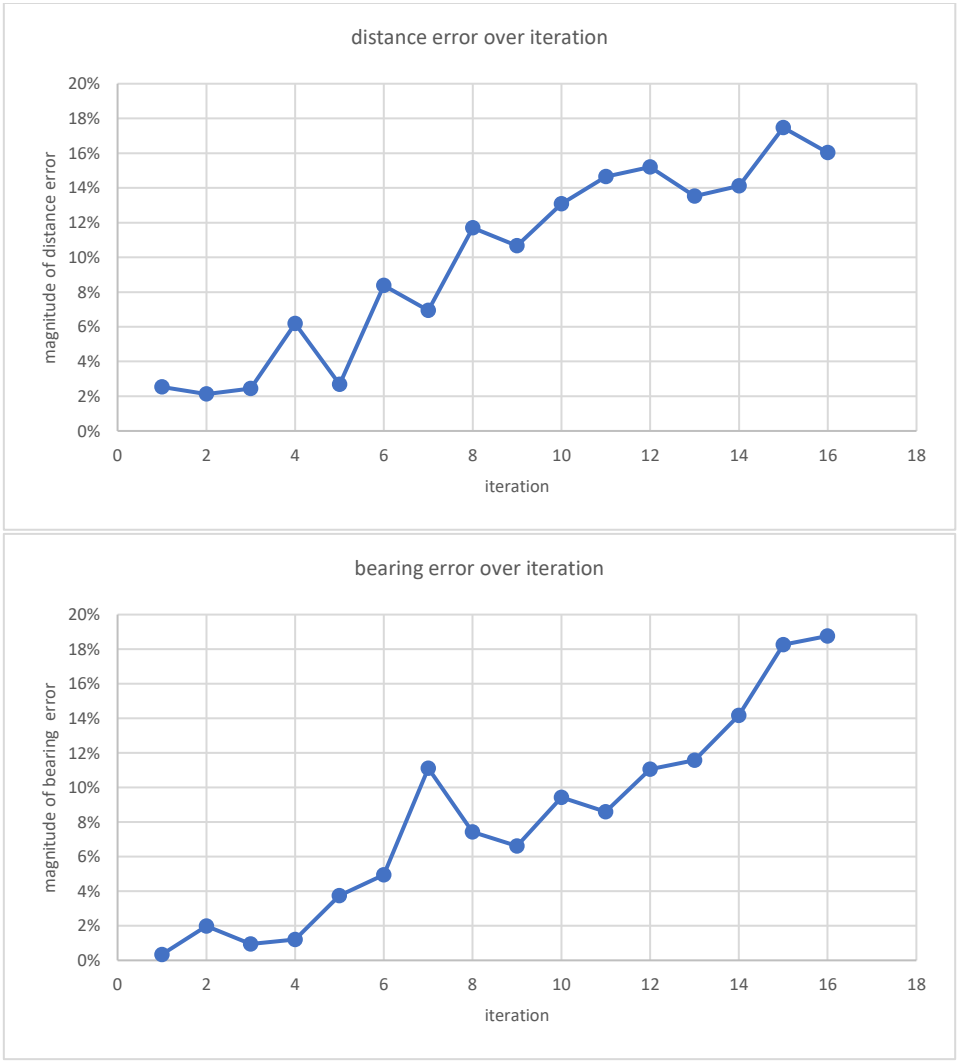


Figure 4 the plot for the distance and bearing errors for the second test

The distance error and the bearing error for the third test have been plotted in the **Fig.5**.

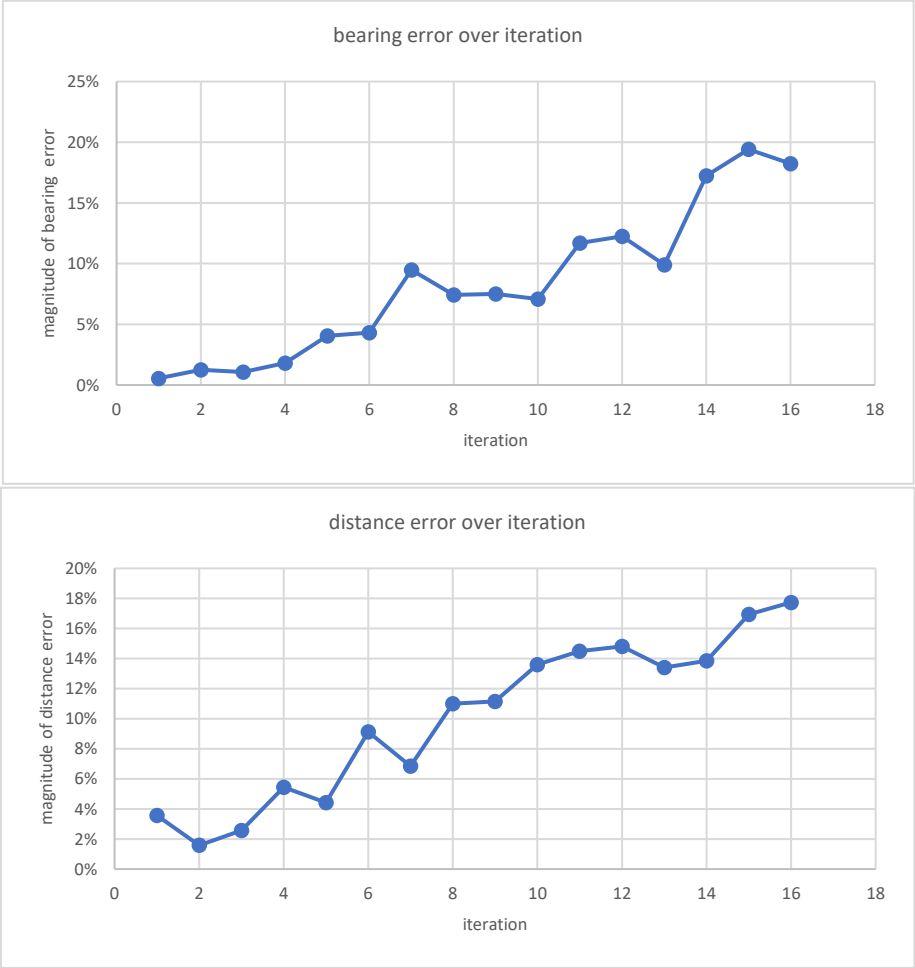


Figure 5 the distance and bearing errors over iterations for the third test

Engineering Standards

In this experiment we had to establish a UART communication between the esp8266 and the hoverboard. For such a communication a knowledge in the number systems is required as the data sent serially after the start frame we defined. This follows the Saudi engineering standards [**GE-T3-02** Recognize the number systems (binary, octal, and hexadecimal)]. The high-level design of the experiment was done initially on Whiteboard then moved to a pseudocode format then implemented. This follows the Saudi engineering standard [**GE-T3-05** Analyse algorithms in flowchart or pseudo code formats]. In our discussion of the reasons behind the growing error in bearing and distance it followed the Saudi engineering standard [**GE-T14-01** Use analytical thinking (logical deductions, statements and assumptions, cause and effect, verbal reasoning, analysing arguments, statements and conclusions, break a complex problem into smaller problems and solve them)].

Discussion

In this section, after we analyzed the collected data from the above tests. It is time to make sense of our above analysis on the distance and bearing measurements. We have validated the use of the Haversine formula in the navigation algorithm in our robot to find the distance and bearing but on the other hand we found that when we start the each of the above tests the robot moves with high accuracy but over the time it starts deviating and creating much higher error rates. This could be due to slippage of the wheels and over the time its cumulative effect starts to grow in addition to the GPS signal accuracy which can be affected by the surrounding buildings. However, the effect of these factors could be reduced for instance by combining other readings such as the IMU readings and fusing it with the GPS updates. In addition, a PID controller can be implemented to keep correcting the orientation of the robot and reduce the angle error.

Conclusion

In this experiment, we have validated the use of the Haversine formula in the navigation algorithm in our robot to find the distance and bearing and that the robot is able to move to certain waypoint using the GPS coordinate of latitude and longitude converting it to distance and bearing to move with certain orientation for the goal distance. However, It is recommended to fuse other readings with the GPS updates so when the tall buildings block or affects the accuracy of the GPS signal, the robot can handle that using for instance the odometry alone and giving less weight to the GPS signal. It is also recommended to implement a PID controller to keep correcting the orientation while the robot moves as the bumpy roads may cause some deviation in the orientation in which we need the PID controller to reduce that effect.

References

[1]: Formula to Find Bearing or Heading angle between two points: Latitude Longitude - (igismap.com)

[2]: How to Convert Distances From Degrees to Meters (sciencing.com)

[3]: Project | Hoverboards for Assistive Devices | Hackaday.io

[4]: Python Mobile App Tutorial: GPS Marker and Android GPS Permissions | Kivy/KivyMD (morioh.com)

بسم الله الرحمن الرحيم

King Abdulaziz University - Faculty of Engineering – EE-499

Senior Design Project

Report #6

Validation Experiment

Team#3

Advisor: Dr. Muhammed Bilal

No.	Member	
1	Sulaiman Abdullah Abbas	F21_T03_M2_1845862

Introduction

An experiment was conducted to test the effect of weight on the speed of a battery powered delivery robot. It is required to verify that a mechanical load of up to 80kg should not interfere negatively with the speed of the robot. The speed error should not exceed 5% of the theoretical speed. This experiment was conducted indoors with calibrated weights and distances. This test is considered a specification verification experiment.

Body

Safety/ Environmental issues

This experiment was done indoors in the CEIS robotics labs. We had access to an emergency brake that would stop the robot immediately at all times.

Tools

The following list is a representation of tools used in this experiment:

- Measuring tape
- Calibrated weights (up to 80kg)
- Stopwatch
- Laptop
- Scale
- Duct tape (to mark distances)

Work Plan:

1. Setup the robot on a known starting location (d=0) while maintaining a straight line to the other marker.
2. Make sure there is no obstacles in the way.
3. Power up the robot and the laptop.
4. Test the connection between the robot and the laptop by using the “ping” command in CLI.
5. Using the laptop and the stopwatch, synchronize the start as accurately as possible to reduce human error. (We kept facing the issue of human error because we cannot react in time to start the stopwatch perfectly).
6. Provide the GO signal to the robot and monitor its speed by stopping the stopwatch when it reaches its destination
7. In each iteration, we will add 5kg onto the robot and move it back to the starting point.
8. Repeat each iteration and jot down the total weight, actual speed and theoretical speed.

Experimental Uncertainty

In this test case, the weights were with known value, but with each iteration we placed the weights on the scale to make sure they are within 10% of the given value. This ensured that we avoid experimental uncertainty and have an accurate reading each time.

Data Collection

While performing the experiment these values were noted down as part of the data collection part of the report. As we can see, each iteration we increment the weight by 5Kg and calculate the actual speed to compare it with the theoretical speed. Furthermore, the distance is not changing because we want to see the effects of weight on our speed.

Weight (Kg)	Theoretical Time (s)	Distance (m)	Measured Time (s)	Measured Speed (m/s)	Calculated Speed (m/s)	Error %
5	16	8	16.03	0.499	0.5	0.19
10	16	8	16.27	0.492	0.5	1.69
15	16	8	16.3	0.491	0.5	1.88
20	16	8	16.3	0.491	0.5	1.88
25	16	8	16.78	0.477	0.5	4.88
30	16	8	16.5	0.485	0.5	3.13
35	16	8	16.4	0.488	0.5	2.50
40	16	8	16.42	0.487	0.5	2.63
45	16	8	16.3	0.491	0.5	1.88
50	16	8	16.47	0.486	0.5	2.94
55	16	8	16.7	0.479	0.5	4.38
60	16	8	16.45	0.486	0.5	2.81
65	16	8	16.27	0.492	0.5	1.69
70	16	8	16.31	0.490	0.5	1.94
75	16	8	16.27	0.492	0.5	1.69
80	16	8	16.7	0.479	0.5	4.38

Table 1 Data collection for the experiment

Results Analysis

This part of the reports is where we will use Excel to graph our data and make some comments on them. Our main purpose is finding out if adding weights does anything to our speed. We will plot a graph to showcase this and then some comments to analyze our data.

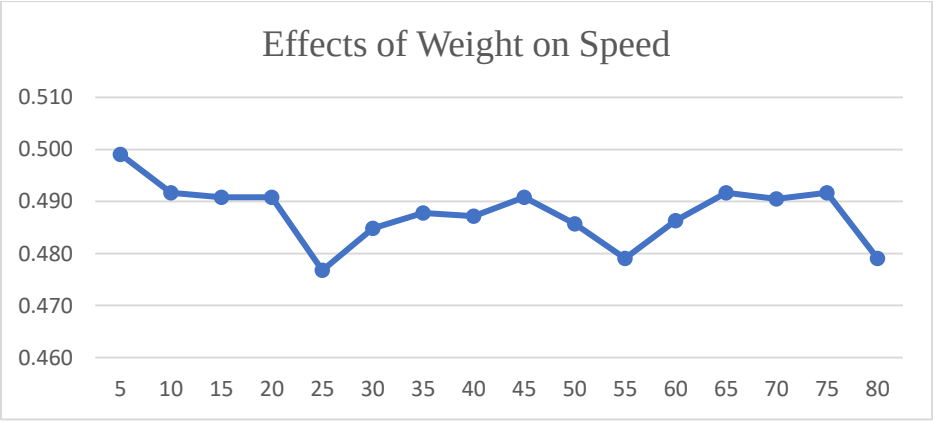


Figure 1 Showcasing the graph

Our graph plots the weights in the data collection vs the calculated speed from the experiment.

Discussion

As engineers, we are expected to understand results of experiments and make notes about them. In this part of the report, we will discuss the outputs of our experiment and how our prototype has reacted to them.

Looking at **fig 1** we can see that the data is following a straight line but with some outliers, this can be attributed to the human error when starting the experiment. Humans possess an average of 25ms visual reaction speed, this has affected our data but it is still within the margin of acceptable error. When looking at the graph we can clearly deduce that the weight has no effect on the prototype and that our structure is good enough to withstand 80Kg. Furthermore, the steel design that we have chosen can actually take more than 80Kg but we will have to test that further. Finally, when looking at the error percentages of the data we can see that it is around 4% to 2% which can be attributed to either human error or slippage due to the nature of indoor flooring.

Engineering Standards

ME-T5-01: Identify measurement concepts such as uncertainty analysis and instrumentation specifications

EE-T7-10: Demonstrate awareness of instrumentation aspects, including measurements, data acquisitions, and transducers, as well as explain their functions in practice

EE-T9-10: Demonstrate knowledge and awareness of software techniques in real life applications

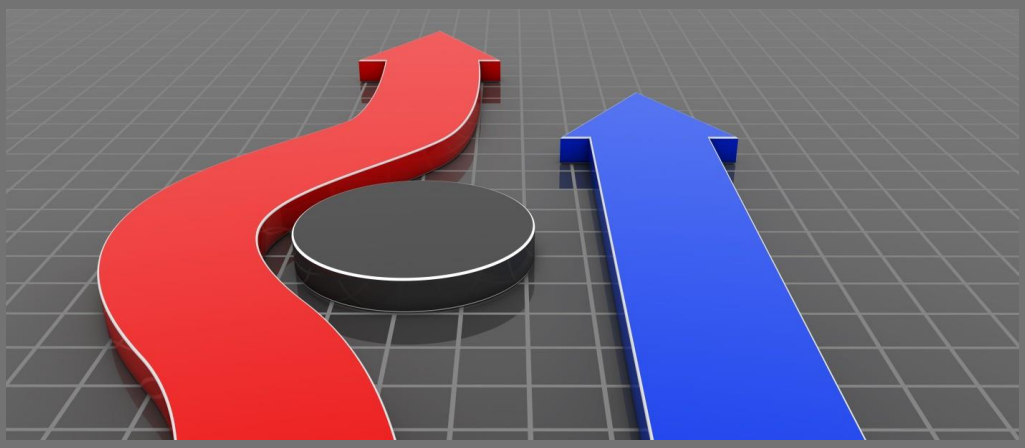
Conclusion

In this experiment we have tested the effects of weight on the speed of our robot and acknowledged that it is within acceptable ranges. We have supplied the required tools and working procedures should anyone wish to replicate this test. Furthermore, our data collection has been showcased. To conclude, this experiment has been fruitful to make sure that one of our musts is completed and in the future, we can add more weights and see the effects of that on our robot.

References

[1] Saudieng.sa. 2022. [online] Available at: <https://www.saudieng.sa/Arabic/Documents/General%20Eng%20Standards.pdf> [Accessed 21 April 2022].

[2] Saudieng.sa. 2022. [online] Available at: <https://www.saudieng.sa/Arabic/Documents/Mechanical%20Eng%20Standards.pdf> [Accessed 21 April 2022].



KAU Delivery Robot – Obstacle Avoidance

-Spring 2022-

SDP – Term 2

Member03: Wael Aldhaheri Supervisor: Dr. Mohammed Bilal

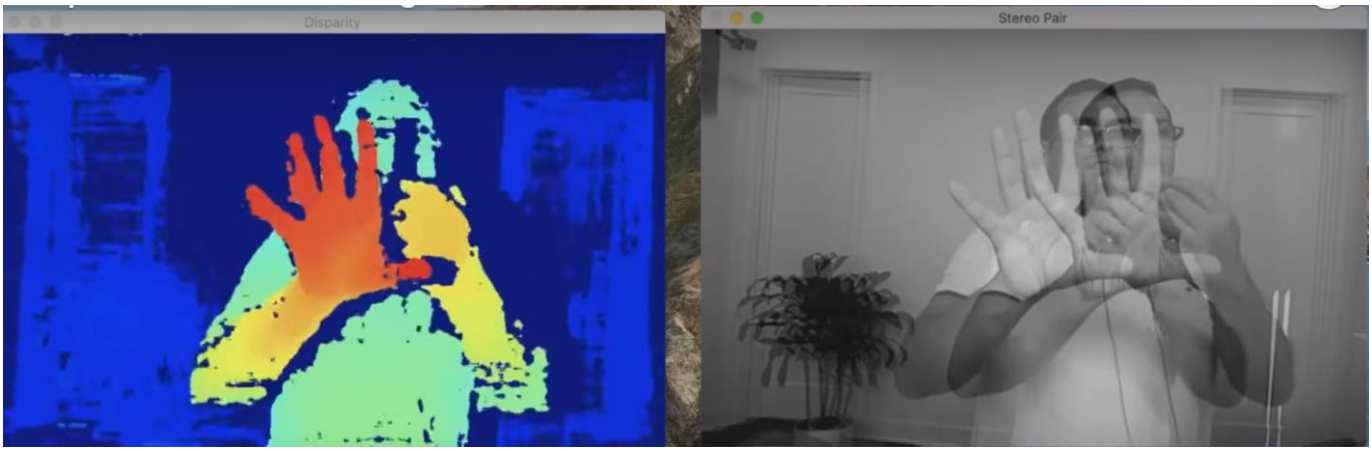
Table of Content

1 Introduction

This validation report was done as a part of the validation stage for our project titled "**KAU Delivery Robot**". This report focuses on the **obstacle avoidance** part of our project. Two experiments were done as a part of this report. The first is more of a pre-validation experiment which aims to map the distance (between the robot and the obstacle) to an intensity level in an image taken from the stereo camera. And the second was to test if the robot would avoid or hit an obstacle at various distances. In these experiments we will be using an Oak-D Camera which is a device that contains two stereo cameras for depth estimation and a color camera it can also run neural networks. Although in this report our main focus will be on the stereo cameras.

1. Experiment background

Obstacle avoidance in our project is based mainly on depth estimation using a stereo camera. Because we are using a stereo camera, we have two images (right and left), and the way depth estimation works, is by putting those two images on top of each other and then matching each pixel to its counterpart in the other image and then measuring the distance between the pixel in the right image and its counterpart in the left image. Which is what is called **disparity**. Objects that are closer to the camera have a higher disparity and objects that are further away from the camera have a lower disparity. As shown in the figure the disparity for the hand is higher than that of the doors. Which means that his hand is closer to the camera than the doors.



a b

Figure 1 – (a) depth map. (b) overlapped images of stereo pair.[1]

But how does this pixel matching occur? Do we go through every single pixel? No, there are some optimization processes that make this matching faster. First of all, we use something called rectification which makes epipolar lines horizontal. In other words, the pixel and its counterpart will be in the same horizontal line in both images. So now we have narrowed down the search to only one line. We also only search the "neighborhood" of the pixel, for even faster processing.

There are certain limitations to this technique in depth estimation. Namely, a limitation in the minimum perceivable depth and difficulty when dealing with smooth featureless surfaces. The first of which is further Exacerbated by the optimization done that searching only in the neighborhood of the pixel. Because at closer ranges the corresponding pixel in the other image might be outside of the neighborhood and is therefore not found. At default settings, this blind spot is at around 70 cm away from the camera. The other problem arises due to the difficulty of finding

the points correspondences in featureless surfaces. Because, normally each pixel has a single counterpart in the other image, but in the case of surfaces that are all white for example it would be difficult to find the counterpart of each pixel because all the pixels in wall are the same. [1][2]

2. Objectives

The goal of this report first and foremost is to verify that our obstacle avoidance is working properly. And that our robot could avoid obstacles at various distances. We could also use some of the data acquired for further optimization of our algorithms.

3. Assumptions

Throughout this document when referring to distances, we are referring to the distance to the robot itself. Although practically the distance to the camera would be about 10 cm more than the distance to the robot. But we consider this distance negligible because we aren't concerned with the actual amplitudes but only how much the speed changes relative to the distance.

4. Variables

1. **Pre-validation – Depth mapping**
    1. Independent variable: distance (cm)
    2. Dependent variable: intensity (/255)

2. **Validation - Obstacle avoidance**
    1. Independent variable: distance (cm) , angle (degrees) between different runs
    2. Dependent variable: angular velocity

5. Constants

1. Environment – all the tests were done at the same place
2. Camera placement
3. Angle for every set of tests
4. Obstacle size

6. Safety/ Environmental issues

The tests were done indoors in the lab to reduce safety risks. We also had access to an emergency button at all times so that we can stop the robot if an accident was about to happen.

2 Tools

- Robot (Hoverboard, ESP8266)
- Measuring tape
- Oak-D Camera
- MATLAB for plotting

3 Procedure

1. Pre-validation – Depth mapping

1. Start by putting the robot at about 40 cm away from the obstacle.
2. Start recording the depth map by recording the screen.
3. Then start moving the robot continuously away from the obstacle.
4. Log elapsed time in seconds and the average of the middle part of the depth map.
5. View the video and use a color picker to determine the intensity of a pixel at different distances from the obstacle

2. Validation – Obstacle avoidance

1. Start by putting the robot at about 160 cm away from the obstacle.
2. Then start moving the robot continuously towards the obstacle.
3. Log the elapsed time in seconds and the angular velocity.
4. Stop the robot and put it 40 cm closer than the last test.
5. Repeat until 40 cm away from obstacle.
6. Change angle by 15 degrees to the right and repeat steps 1-5 then change angle to 15 degrees to the left and repeat steps 1-5.

4 Practical issues

There were some issues with data logging in our initial test runs. The issue was that the data logging started before the robot started moving which made it hard to combine the plots for better information extraction. We fixed this by making it so that the data logging starts when the robot starts moving

5 Data collection & Discussion

1. Pre-validation – Depth mapping

As shown in the figure below, we are dividing each frame of the depth map into three sections (middle, right and left) sections which will facilitate obstacle avoidance in the next experiment. In this experiment we are logging the average intensity of the middle section and then logging a manually selected pixel intensity at different

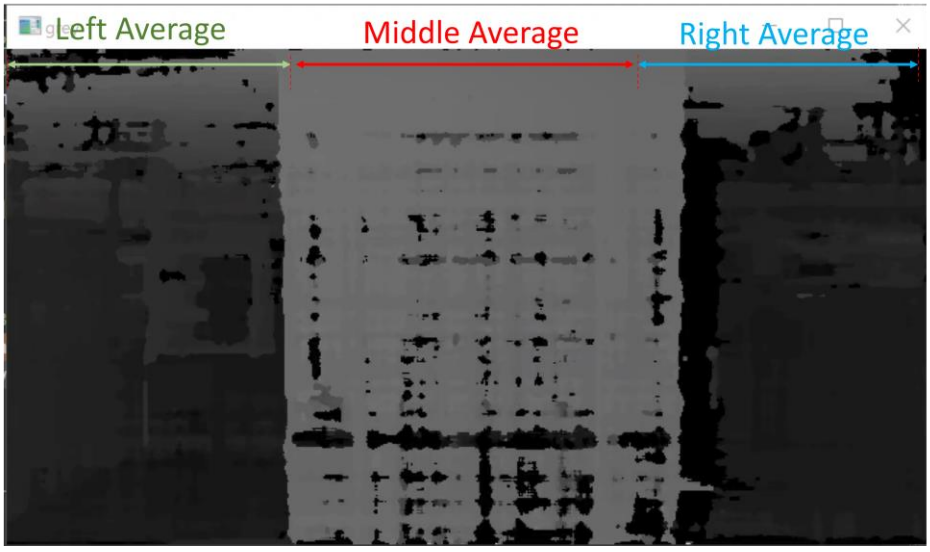


Figure 2 - Sections of depth map

During testing of this experiment, there was a delay in the robot movement by about 13 seconds so the first part of plot that is noisy is while the robot is standing still at about 40 cm away from the obstacle. This noise is due to the camera's minimum perceivable distance.

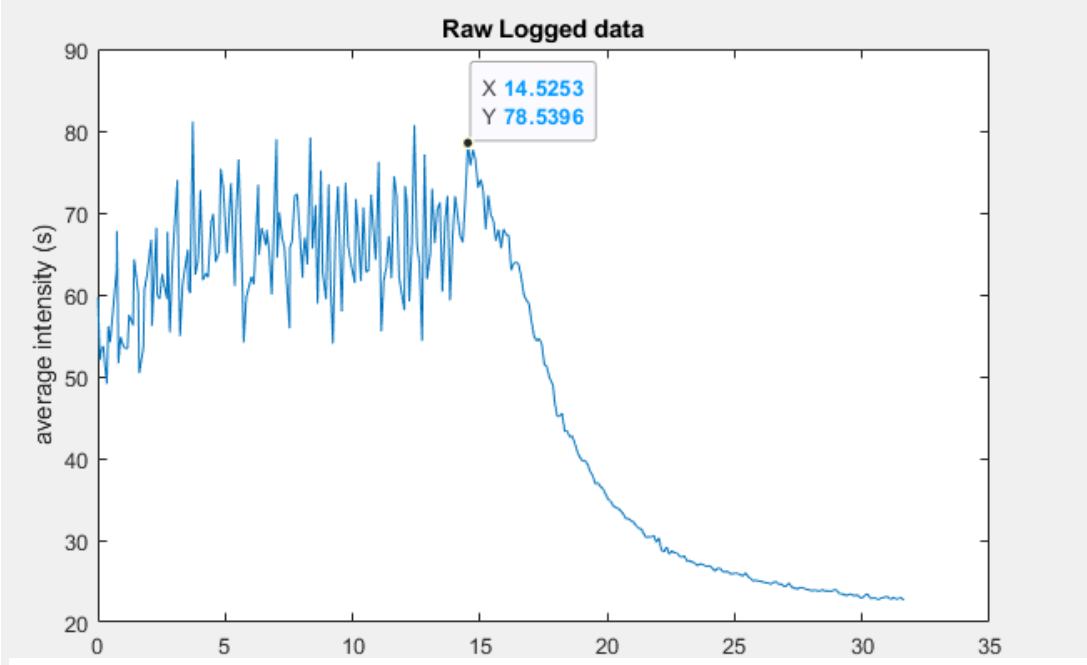


Figure 3 - Raw logged data

The figure below is an example of the depth map when there are obstacles in the blind spot. As shown in the figure the image is very noisy which causes the average to change rapidly with time.

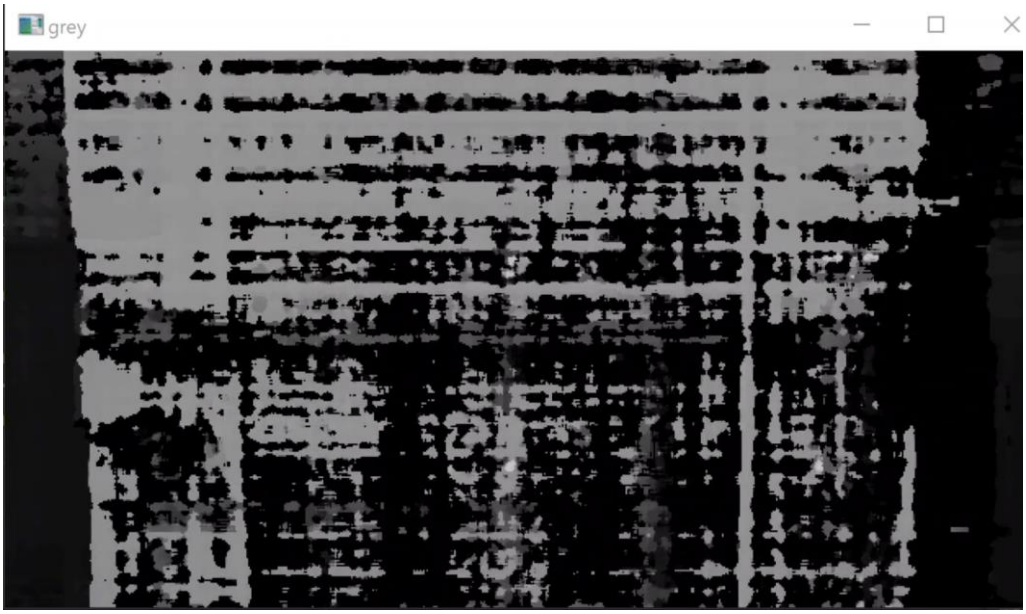
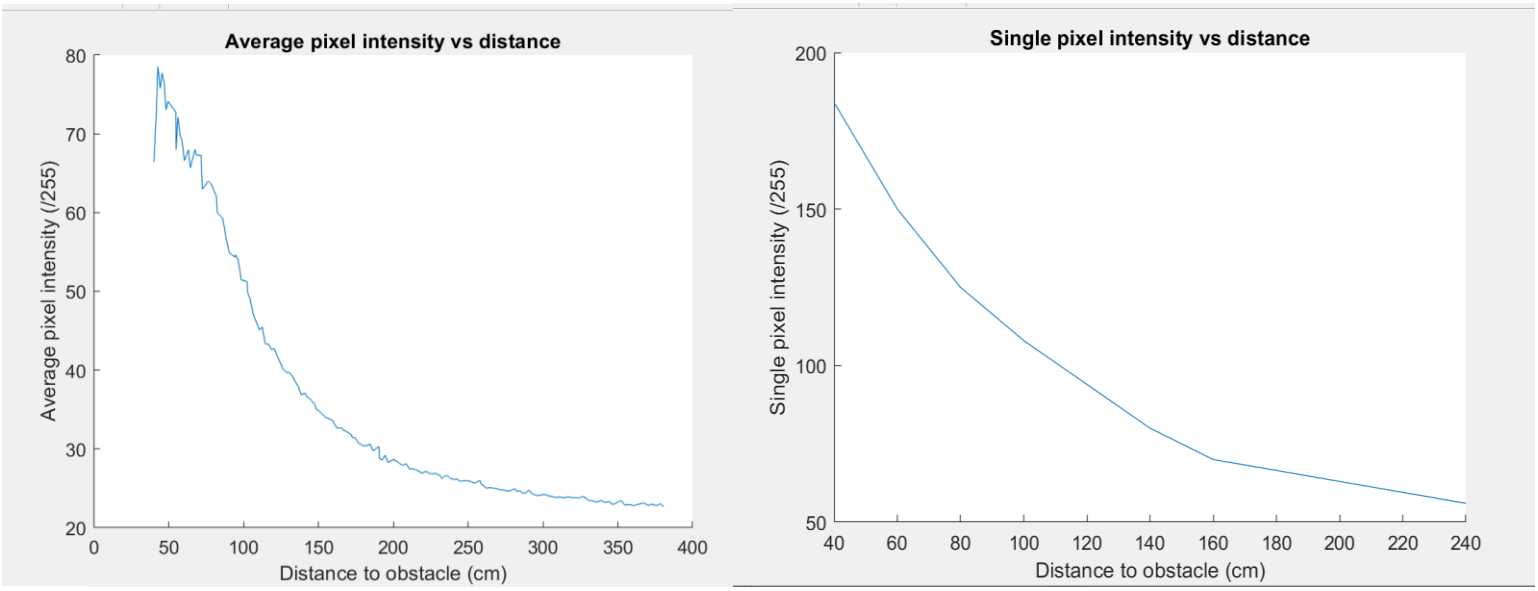


Figure 4 - Raw depth map of obstacle that is 40 cm away

We then removed the part of the plot during which the robot was stationary. And because the robot was moving backwards (away from the obstacle) at a constant rate we were able to convert the time to distance to obstacle.



a b

Figure 5 - (a) Average pixel intensity over distance. (b) single pixel intensity vs distance.

As shown in the figure above the overall trend of both images are very similar. Although the amplitude when selecting a single pixel is obviously higher because we were manually selecting pixels that were representative of the distance. Whereas for the average the background (black/ 0 intensity) is taken as a part of the average.

We then normalize both plots and plot them together. Although there were some differences in amplitude but in our case, we are more interested in the relative intensity so taking the normalized versions allow us to compare. And as shown in the figure the average pixel intensity is closely correlated with the value of the manually selected pixel. Consequently, the average pixel intensity is sufficient in our case and it easier to implement in real time

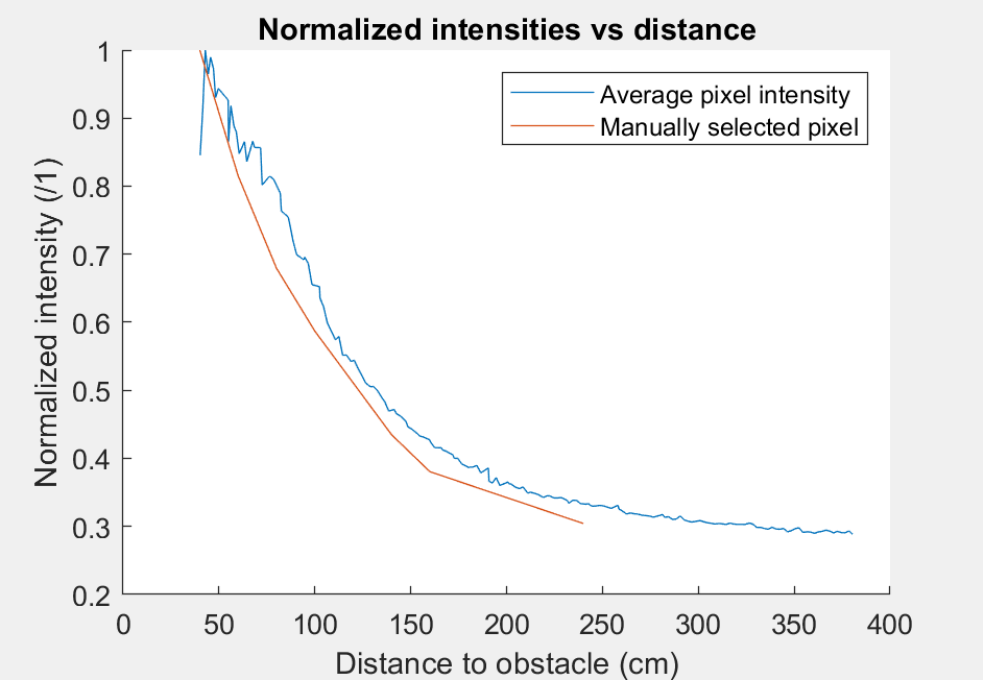


Figure 6 - Normalized intensities vs distance

2. Validation - Obstacle avoidance

As we previously mentioned dividing the image into sections facilitates obstacle avoidance. Which is done by calculating the average of each section and if the average in middle is above a certain threshold (the road isn't clear) the robot then tries to move left with an angular speed proportional to the sum of the middle average and right average if the left average is also above the threshold the robot then tries to move right and similarly with an angular velocity that is proportional to the middle average and the left average. If it also cant, it stops moving. The equation for calculating the angular velocity is :

ang_vel = 0.065+0.0015*(avg2+avg3); # for turning left

ang_vel =-0.065-0.0015*(avg1+avg2); # for turning right

In the plot below the x-axis represents the distance left until the robot hits the obstacle. Such that the 0 represents the obstacle. Each graph represents a test for a starting distance.

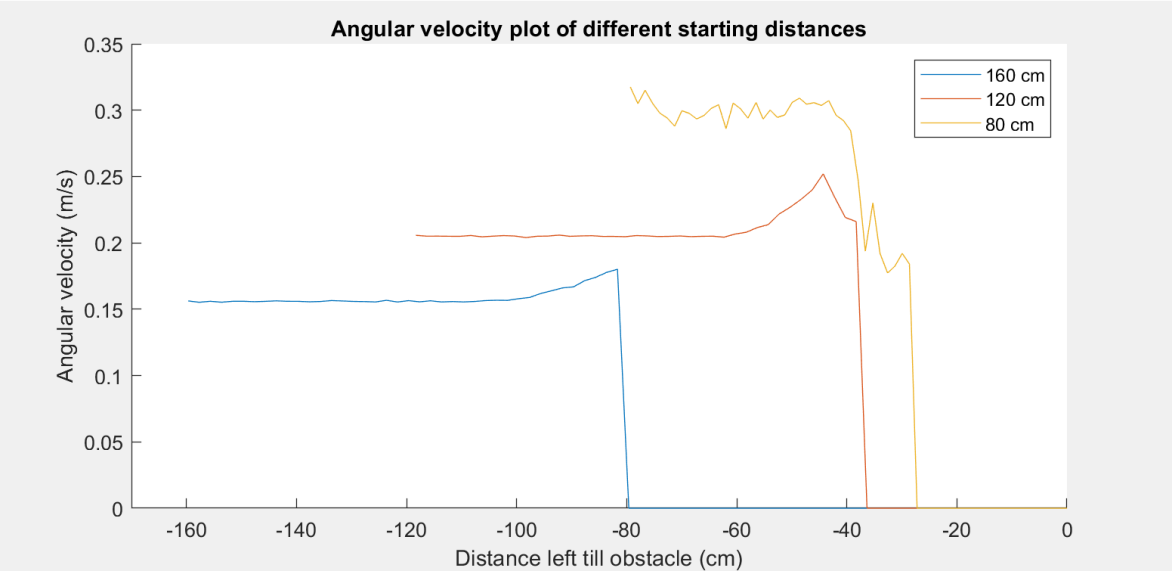


Figure 7 - Angular velocity vs distance (cm) for different starting points

As shown in the figure for tests that started at a closer distance to the obstacle, the angular velocity was higher. Although we can notice that the plot isn't as smooth as we expected. This could be due to the previously mentioned errors; like the background being part of the average and random noise perceived by the camera due to not being able to find the point correspondences for some of the featureless surfaces in the lab.

Table 1 - Results summary of various obstacle avoidance tests

Starting point distance to obstacle (cm)	Orientation (degrees)	Pass/Fail
160	0	Pass
120		Pass
80		Pass
40		Pass
160	-15 (right)	Pass
120		Pass
80		Pass
40		Failed
160	15 (left)	Pass
120		Pass
80		Pass
40		Failed

As shown in the table, the robot passed almost all the obstacle avoidance tests and only failed the tests where we place the obstacle in the camera's blind spot. However, for the 0 degrees test where the robot is directly facing the obstacle, we adjusted the code to measure the variance of each section to try and account for the blind spot. And It was able to pass the test because it stopped and didn't hit the obstacle. Whereas for the tests where it was oriented right or left it wasn't able to detect the obstacle even with the variance and it hit it. This is likely due to the variance itself having a very high variability with time. Although practically, no object should reach the blind spot and that only happened in testing because we deliberately placed the robot very close to the obstacle and then started it. Another case where that might happen is if someone deliberately tries to jump in front of the robot.

6 Conclusion

1. Pre-validation – Depth mapping

For the depth mapping after comparing the average intensities with the intensities of the manually selected pixels. It can be concluded that the average is a good indication of distance and is less computationally expensive. We also notice that the accuracy decreases the further away from the camera you go. Which is to be expected because the difference in disparity becomes smaller.

2. Validation - Obstacle avoidance

For the obstacle avoidance tests the angular velocity logs were a bit erratic but in testing it worked fine and avoided the obstacles. This erratic behavior is happening for a short enough time that it isn't actually affecting the behavior.

7 Engineering Standards

The following general engineering standards were a part of this experiment:

- GE-T2-01 Recognize the concept of mean, mode, standard deviation – The average intensity was used for depth estimation.

- GE-T2-05 Apply hypothesis testing (One sample test of mean and population proportions) – The average was compared to more accurate results using manual pixel selection.

- GE-T14-01 Use analytical thinking (logical deductions, statements and assumptions, cause and effect, verbal reasoning, analyzing arguments, statements and conclusions, break a complex problem into smaller problems and solve them) – Various assumptions and cause and effect observations were done as a part of the experiments.

8 References

[1] "Depth Estimation using OAK-D and it's other variant | OAK Series - YouTube." https://www.youtube.com/watch?v=fx4uXLQWWi4&t=905s (accessed Apr. 21, 2022).

[2] "Depth perception — DepthAI documentation | Luxonis." https://docs.luxonis.com/en/latest/pages/depth/ (accessed Apr. 21, 2022).

9 Appendix A – Data tabulation

Table 2 - Average intensity table

Distance to obstacle (cm)	Average intensity (/255)
40.17934799	66.40568627
41.56708717	71.8020028
42.84368515	78.53955182
44.39959049	75.76908964
45.5772686	77.71504202
47.13217735	76.39214286
48.2140255	73.05511204
49.65003014	74.11439776
54.79622364	72.76222689
54.97780323	67.982507
56.19251728	72.13166667
57.80848026	69.73981793
59.02492046	69.11068627
60.52098751	66.58791317
63.31353188	67.97778711
64.51027393	65.67128852
67.61055946	68.02116246
68.50895882	67.30778711
71.69964314	67.27635854
72.51760483	62.96854342
75.78866959	63.83130252
76.52690411	63.98429972
78.50147724	63.61988796
81.97216988	62.01371148
82.51069546	59.96918768
86.04147911	59.2970028
86.51995182	58.88903361
88.50900173	56.64193277
90.5027771	54.92766106
94.23028946	54.33740896
94.62864876	54.65357143
96.52410984	53.93592437
98.51819515	51.46885154
102.4284172	51.23488796
102.6870155	49.90407563
104.5221567	48.93665266
106.516881	47.04712885

110.6656885	45.13553221
111.0646248	45.15701681
112.5208616	45.49584034
114.5185566	43.3307423
116.5137959	43.33726891
118.8070107	42.60904762
120.5038691	42.74148459
122.4971294	41.80889356
124.9556828	40.79134454
126.5115881	40.11579832
129.0447283	39.67710084
130.5207586	39.72236695
132.5154114	39.28106443
135.1883316	38.34259104
136.5057039	37.94222689
138.5193157	36.87414566
141.3518333	37.06186275
142.5285912	36.56887955
144.5088577	36.33460784
147.5001001	35.67971989
148.5181284	35.06302521
150.5325842	34.80008403
153.6236954	34.23971989
154.5215464	34.01526611
156.5160131	33.88635854
159.7771358	33.57127451
160.5155134	33.23627451
162.511797	32.65292717
165.9007263	32.64326331
166.5191269	32.3972549
168.5190582	32.24098039
172.0696068	31.80354342
172.608223	31.43243697
174.5230913	31.42529412
176.5176535	30.79966387
180.2632999	30.34770308
180.8415937	30.40812325
182.5171232	30.40994398
184.5123482	30.6035014
186.5263462	29.74438375
190.4957819	30.30614846
190.8148766	28.78259104

192.510891	28.56565826
194.5648336	29.19277311
196.5031004	28.28897759
200.7682705	28.6994958
201.0475111	28.49963585
202.5235605	28.43205882
204.5089865	28.10117647
206.8620491	27.91581232
208.5182285	28.13619048
210.971036	27.41812325
212.5268698	27.51113445
215.0800228	27.34386555
216.5163183	27.23726891
219.1690588	26.91526611
221.203661	27.0987535
222.5202179	27.0880112
224.5347404	26.86112045
226.510129	26.84519608
228.5041332	26.89556022
231.4562416	26.587493
232.5332975	26.23560224
234.4998884	26.58764706
236.5346479	26.56239496
238.5291338	26.19166667
241.6833353	26.13156863
242.5400019	26.18585434
244.5147943	25.89508403
246.5292692	25.92376751
248.5240364	25.98988796
250.5249691	25.93563025
252.5395012	25.79239496
254.5141983	25.65366947
258.0849504	25.99147059
258.6432743	25.5654902
260.505929	25.31738095
262.5198793	25.00498599
264.5145416	25.11273109
266.5291452	25.02212885
268.523798	24.99079832
270.5185127	24.85514006
272.5577641	24.81323529
274.532423	24.76236695

276.4871883	24.62571429
280.6361341	24.88134454
281.2544966	24.95656863
282.4915457	24.61691877
284.5057154	24.67984594
286.5248823	24.3730112
288.4996176	24.41063025
290.5541611	24.77522409
292.5288057	24.32484594
294.565115	24.16896359
296.498189	24.0667507
298.4928942	24.13970588
301.0868788	24.23469188
302.5021267	24.12064426
304.4967794	23.99557423
306.535573	23.92861345
308.5109425	23.83652661
310.5057764	23.90191877
312.5195694	23.86354342
314.5103455	23.78236695
316.5233135	23.94843137
318.4987497	23.83491597
320.5325747	23.80147059
322.5072908	23.80413165
324.5020151	23.79471989
326.5172338	23.95607843
328.5119629	23.81572829
330.5065823	23.4435014
332.5012875	23.4559944
334.5161152	23.31180672
336.5104961	23.2467507
338.5054827	23.46840336
340.5390596	23.26864146
342.5112629	23.24036415
344.5067644	23.33303922
346.5204144	22.92655462
348.5000086	23.0864986
350.5144215	23.31376751
352.5103283	23.40532213
354.5033407	22.87819328
356.6175652	22.93837535
358.492589	22.92012605

360.5290794	22.76918768
362.5271893	22.92393557
364.5418644	22.99079832
366.5165615	23.13638655
368.5112619	23.02754902
370.506711	22.80081232
372.5028181	22.99042017
374.515729	22.85533613
376.5097523	22.82443978
378.5244036	23.05322129
380.4991055	22.6547479

Table 3 - Angular velocity table (start at 160 cm away from obstacle)

Distance till obstacle (cm)	Angular velocity (m/s)
-159.6833181	0.156266
-157.6858044	0.15522
-155.6689787	0.156004
-153.6366701	0.155256
-151.6617346	0.155985
-149.6597481	0.156027
-147.6820612	0.155677
-145.6771183	0.155958
-143.6879158	0.156297
-141.650362	0.155996
-139.6535683	0.155962
-137.6565313	0.155587
-135.6581402	0.155838
-133.6773205	0.156506
-131.6636562	0.156221
-129.6411562	0.155897
-127.4091053	0.155733
-125.6572342	0.155462
-123.6561108	0.156719
-121.6660118	0.155448
-119.6584415	0.156458
-117.6407862	0.155582
-115.6541681	0.156356
-113.6594057	0.155469
-111.6253853	0.155782

-109.6476269	0.155466
-107.687006	0.155853
-105.6564474	0.156496
-103.6564016	0.156768
-101.6798353	0.156662
-99.64697361	0.157924
-97.66183376	0.158851
-95.64255238	0.161848
-93.67416382	0.163874
-91.63655758	0.166093
-89.66831684	0.166846
-87.64111519	0.171517
-85.66226959	0.173976
-83.68926048	0.177716
-81.68615341	0.180117
-79.62046146	0
-77.68130779	0
-75.68090439	0
-73.68118763	0
-71.65141106	0
-69.64927197	0
-65.96250534	0
-65.53682327	0
-63.65308285	0
-61.67109013	0
-59.65569973	0
-57.66670704	0
-55.66843033	0
-53.66454124	0
-51.69132233	0
-49.60748196	0
-45.61151028	0
-45.01767159	0
-43.58533382	0
-41.66868687	0
-39.63932991	0
-37.67965317	0
-35.67815781	0
-33.67210388	0
-31.17026329	0
-29.67651844	0
-27.66334534	0

-25.67514896	0
-23.65908146	0
-20.91867924	0
-19.71039295	0
-17.67985821	0
-15.66466331	0
-13.67398739	0
-11.73670769	0
-8.633728027	0
-7.685065269	0
-5.692954063	0
-3.688116074	0
-1.74926281	0
0.317015648	0
2.31071949	0
4.341745377	0
6.306815147	0
8.273262978	0
10.28152943	0
12.32928753	0
14.25394058	0
16.37963772	0
18.31315994	0
20.31920433	0

APPENDIX – B: SELF ASSESSMENT CHECKLIST

Use student outcomes (SOs 1 - 7) rubrics to fill the following table. Each member needs to fill in this table; it is important to enable the department to know to what degree the EE programs have been able to achieve the required KPIs of each SO. Please use the following grading letters:

E: Exemplary, **S:** Satisfactory, **D:** Developing, and **U:** Unsatisfactory.

Student Outcome (SO)	**Key Performance Index (KPI)**	**Self-assessment (E, S, D, or U)**		
		M1	**M2**	**M3**
1. an ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics	1.1. Problem Identification	S	S	S
	1.2. Problem formulation	S	D	D
	1.3. Problem solving	E	E	E
2. an ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors	2.1. Design Problem Definition	S	D	S
	2.2. Design Strategy	D	S	S
	2.3. Conceptual Design	D	S	S
3. an ability to communicate effectively with a range of audiences	3.1. Effective Written Communication	E	E	S
	3.2. Effective Oral Communication	D	S	S
4. an ability to recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts	4.1. Recognition of Ethical and Professional Responsibility	E	S	E
	4.2. Consideration of Impact of Engineering Solutions	S	S	S
5. an ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives	5.1. Effective Team Interactions	D	S	S
	5.2. Use of Project Management Techniques	D	U	D
6. an ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions	6.1. Developing Appropriate Experiment	S	S	S
	6.2. Conducting Appropriate Experiment	S	S	S
	6.3. Analysis and interpretation of Experiment Data and Drawing Conclusions	S	S	E
7. an ability to acquire and apply new knowledge as needed, using appropriate learning strategies	7.1. Effective Access of information	S	S	S
	7.2. Ability to learn and apply new knowledge independently	S	S	S