Cairo University
Faculty of Engineering
Department of Mechanical Engineering
Mechatronics Program

Autonomous Landmine Detection Robot

Advanced Robotics & Artificial Intelligence MPDS453

Course Project

Submitted by:

Abdelrahman Adel Ramadan [1190140]

Abdulhamid Walid Abdulhamid [1190463]

Mohamed Adel Hamzawi [1190500]

Mostafa Mohamed Mahmoud Shafik [1190485]

Submitted on: 2nd of June, 2024

Submitted to: Dr. Hossam Ammar

Table of Contents

I. Introduction

1.1 Overview

Autonomous vehicles have gained massive popularity in recent years for there ability to carry tasks that saves man's effort and time, not only does it save time and effort but also saves lives and health by working in hazardous environments. That is the aim of our project where we utilize an autonomous vehicle for the detection of a landmine in a field and marking its coordinates on a map. This technical report presents a comprehensive study of a Robot Operating System (ROS)-based project that leverages a TurtleBot Burger model for autonomous landmine detection.

This project's main goal is to create and put into place a system that will allow the TurtleBot to map landmine sites, detect their presence, and navigate across a given area. The hardware base for this project is the TurtleBot Burger, a small and adaptable mobile robot. A variety of sensors, such as motor encoders for odometry data and an MPU6500 for inertial measurement, guaranteeing precise orientation of the robot in the Gazebo simulation.

1.2 Proposed Landmine Detection Algorithm

The proposed landmine detection algorithm primarily relies on the accurate navigation of the robot as well as a computer vision detection system. The algorithm starts by moving the robot starting from the beginning of the map, it keeps moving forward up until the computer vision system detects an object in the frame, the robot stop, compare the object to the pretrained dataset, if it is a landmine, it sends a pose message with the current position to the simulation program, which accuraretly marks the position of the landmine with a specific marker on the map. After that, the robot maneuvers around the object to avoid it and continue moving through the map unitll a goal coordinate is reached. This goal is said to be reached when either the computer vision system detects a landmark indicating a specific location, or the position of the robot reaches this position.

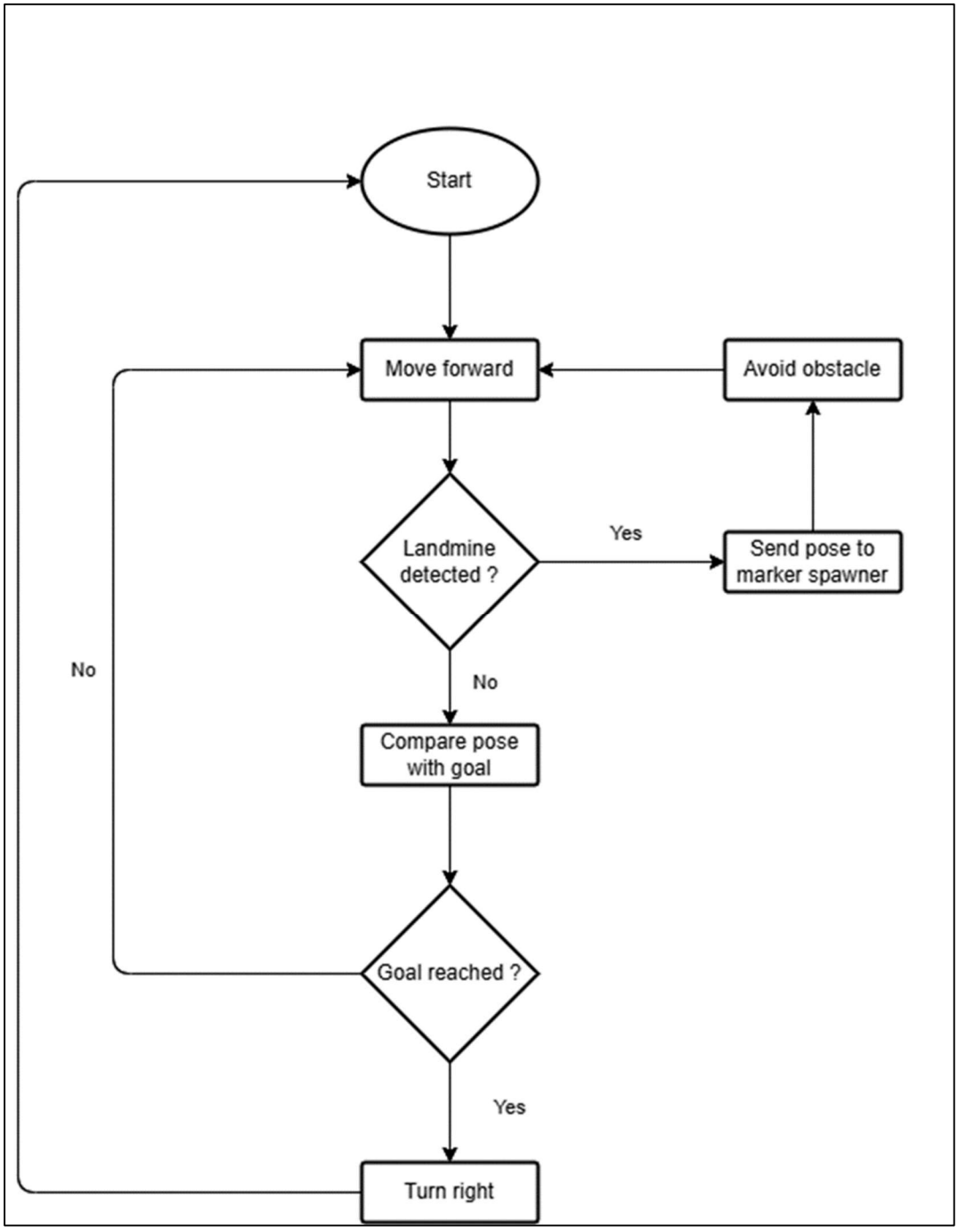


Figure 1 Landmine detection algorithm flowchart

II. Communication Through ROS Nodes

Communication between the robot motors, joystick, and gazebo simulation is done on a python node called the "Master" which coordinates the communication between the node as in the following diagram:

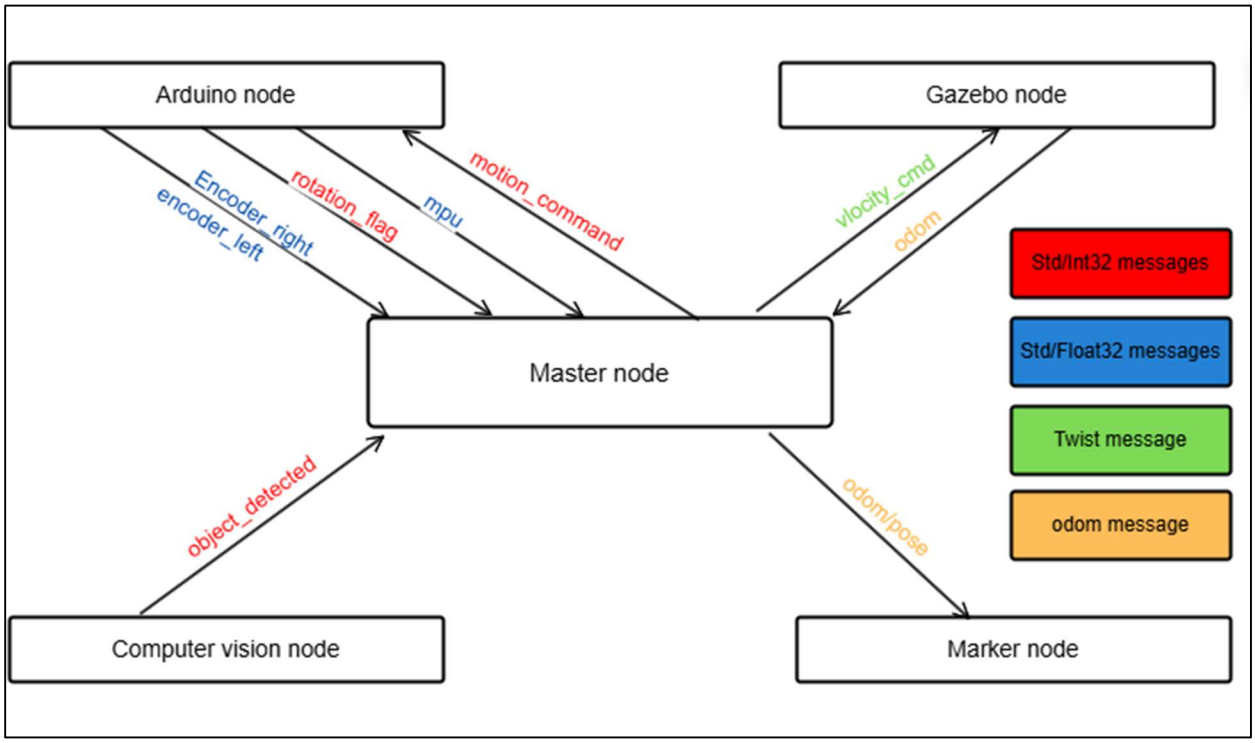


Figure 2 Ros Nodes Communication

III. Embedded Hardware & Software

3.1 CAD Modeling & Robot Mechanical Structure

The TurtleBot Burger, a versatile and widely-used robot platform in robotics education and research, serves as the foundation for our project. In this report, we detail the fabrication and development process of our differential drive robot based on the TurtleBot Burger platform. Our objective was to create a robot capable of autonomous navigation and environment mapping, utilizing ROS (Robot Operating System) for communication between the hardware, Gazebo simulation environment, and the physical robot.

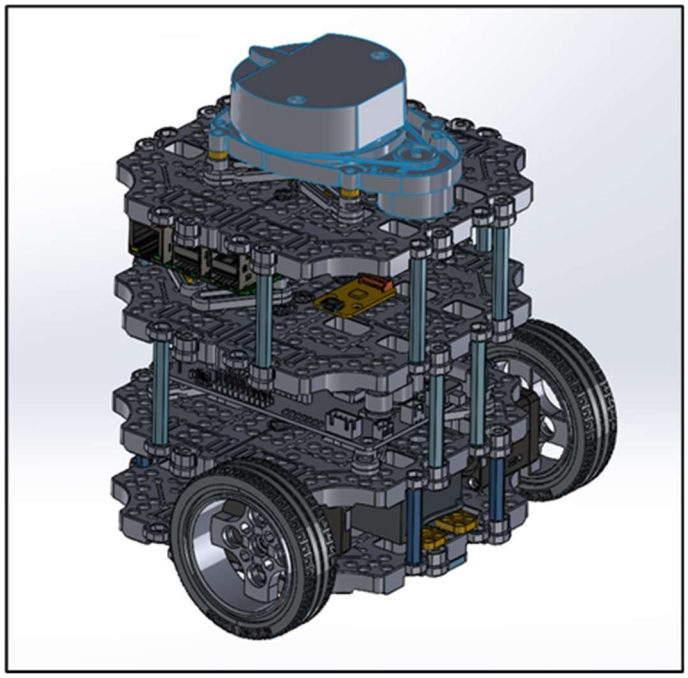


Figure 3 Turtlebot Burger Published CAD Model

4.2 Kinematic Model

Since the TurtleBot Burger is a two-wheel robot with a caster wheel, we could utilize the simplified version of the kinematic model of two-wheel drive robot, which are:

$$\omega_r = (2V - \omega b)/2r_r$$

$$\omega_l = (2V + \omega b)/2r_l$$

Where:

- ω_r & ω_l are the right and left motors rotational speeds respectively.
- V is the linear velocity of the robot
- ω is the rotational velocity of the robot
- b is the wheelbase
- r_r & r_l are the wheel radii

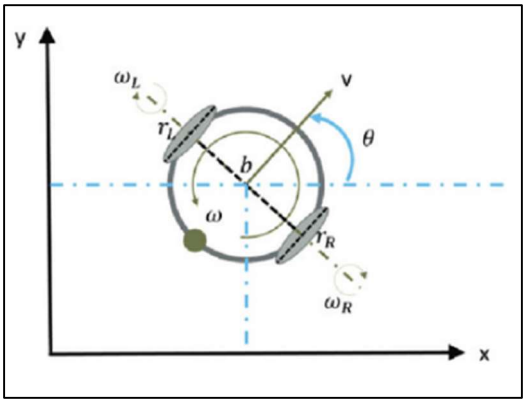


Figure 4 Two Wheel Differential Drive Robot Kinematic Diagram & Parameters

4.3 Embedded Hardware

4.3.1 Total System Requirements & Description

This section of the report comprehensively summarizes the embedded system implementation of the project, stating the components used to assemble the project as well as the hardware interfacing code and how it communicates with the ROS side. The hardware and software requirements from the embedded system were designed so that the Realtime robot accurately move inside a specified area in the correct pathway based on commands sent from the navigation code on the ROS side, accurately send robot feedback data (velocity and orientation) to the navigation code and the simulation code on the ROS side so they could (through kinematics and odometry equations) determine the current position of the robot, and accurately stop at landmine locations and send these locations to the Gazebo simulation map, then perform an obstacle avoidance pattern to avoid this landmine and continue moving in its predefined path.

3.1.2 Hardware Requirements & Components

In this project, the design of the Turtlebot burger robot is used as a differential drive two wheeled robot. To accurately navigate the robot and simulate its behavior, the following requirements were demanded:

- DC motors with encoders to have feedback data of the motor velocities. This is to achieve two main goals; accurate speed control of the motor, and velocity feedback for the navigation and simulation programs.

- Inertial Measurement Unit (IMU). An IMU is used to accurately have feedback data from the current angular velocity and angular position of the robot (orientation). This is to ensure accurate control over the orientation while moving forwards to reject disturbances, accurate control over turning and changing orientation and most importantly, accurate navigation and simulation on the python controller and the Gazebo node.

- Bluetooth communication. A communication medium is required to communicate serially with the ROS side via the rosserial library.

To achieve these requirements, the following components are used in the project:

- Arduino Mega development board.
- 12VDC 210RPM DC Motors with encoders.
- L298N 2 channel motor driver.
- MPU9250/6500 IMU.
- HC-05 Bluetooth Module.
- 12V Battery.
- Switches & Wiring.

3.1.3 Circuit Schematics

Schematic Circuit for the Arduino Hardware Connections:

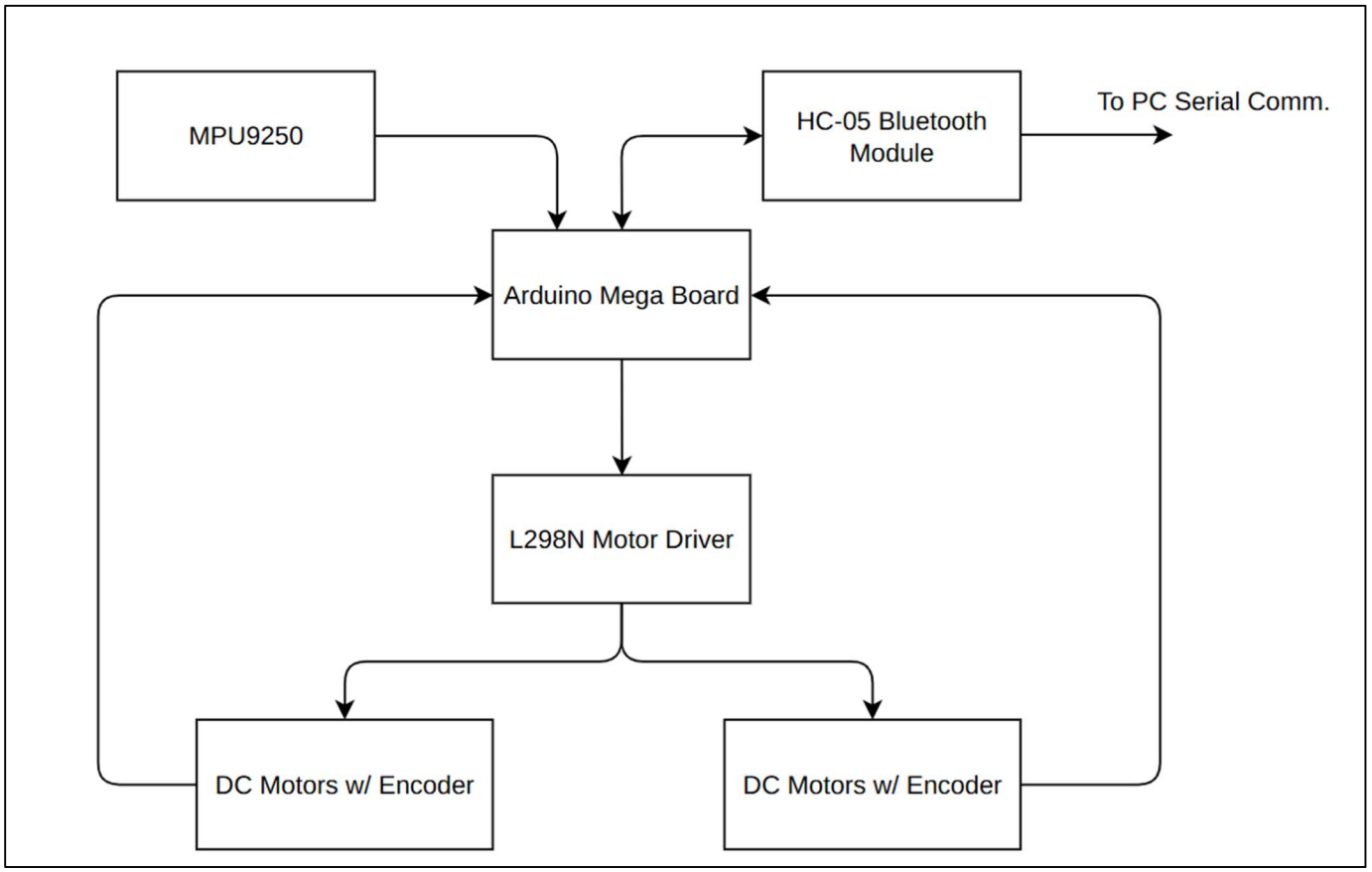


Figure 5 Schematic Diagram for Electronic Circuit Connections

Connection Notes:

Above in figure (1) is a schematic block diagram for the connections of the robot electronics, a detailed wiring for the connections could be found in the appendix of this report. Additionally, below are some connection notes for the hardware components:

- The IMU is connected on the I2C Pins of the Arduino.
- The Encoders are connected to a digital interrupt pin (explained later in this report).
- The Bluetooth module is connected to the serial communication pins (Rx & Tx) of the Arduino.
- The Arduino is powered by the 5V regulator available in the L298N driver module, which is powered by the battery.

3.4 Embedded Software Architecture

3.4.1 System Requirements

There are several requirements were demanded from the embedded hardware code to accurately let the robot perform the desired task:

- Accurate IMU data acquisition (specifically the Z-axis gyroscope DOF).
- Accurate motor encoders data acquisition.
- Interfacing with the ROS side, receiving motion commands and sending back odometry data for navigation and simulation.
- Accurate speed control on the DC motors.
- Accurate orientation control on the robot to allow the robot accurately move in straight lines and corners without drifting away from the desired orientation.

3.4.2 DC Motors Constant Speed PID Controller

Due to hardware mechanical and electrical issues, when the DC motors are given a specific speed setpoint, i.e., a specific speed, the DC motors do not actuate with the same power. This could be an issue either while moving forward or while turning, since the robot motion is controlled by controlling the speed of the left and the right wheel velocities. Therefore, a PID controller was

designed to compensate the difference between the desired speed and the actual speed feedback from the encoders and accordingly, output a compensated PWM signal.

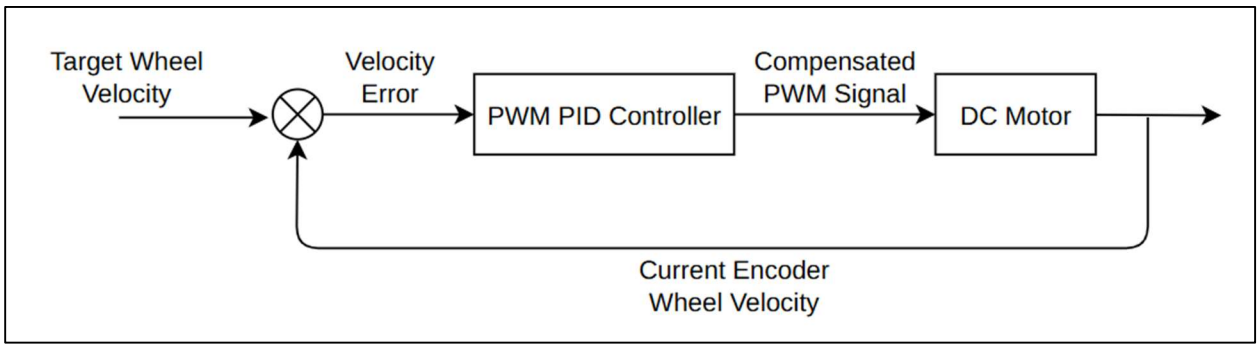


Figure 6 DC Motors Voltage PID Schematic

3.4.3 Robot Orientation PID Controller

To accurately turn the robot into specific orientations according to the desired positioning and to further enhance forward movement of the robot, an orientation PID controller is used. This PID controller takes feedback data from the IMU to measure the current yaw orientation, calculate its error from the desired orientation, then compensate this error by and subtract/add the control signal from the left and right wheel velocities accordingly to ensure the robot follows this orientation.

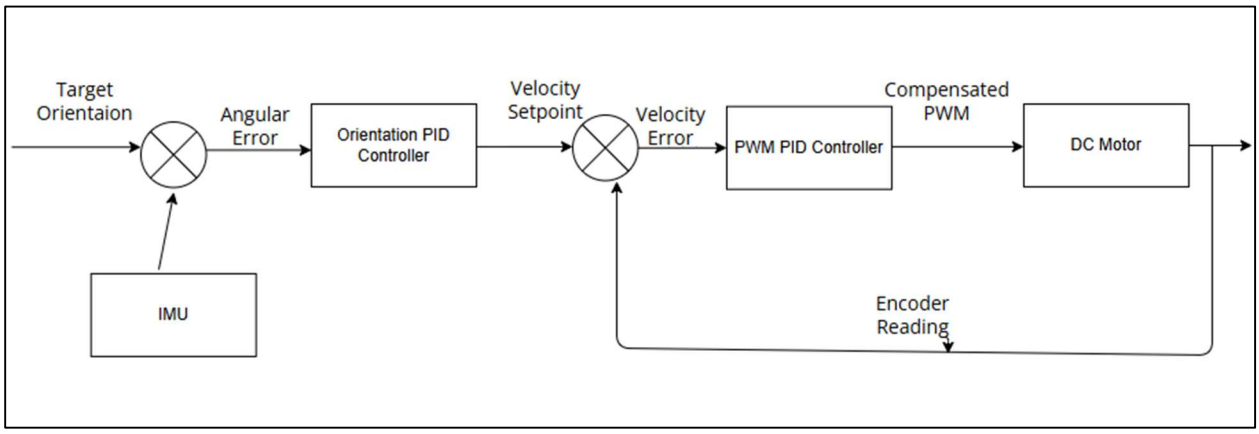


Figure 7 Orientation PID Schematic

IV. Computer Vision System

4.1 Landmine Detection using YOLOV8

4.1.1 Overview

This ROS node implements a landmine detection system using the YOLOv8 object detection model. It captures video frames from a specified webcam URL, processes these frames to detect objects, and identifies landmines based on their classification as "frisbee." The detected landmine's position is determined and outputted for further processing or action.

4.1.2 Functionality

Initialization

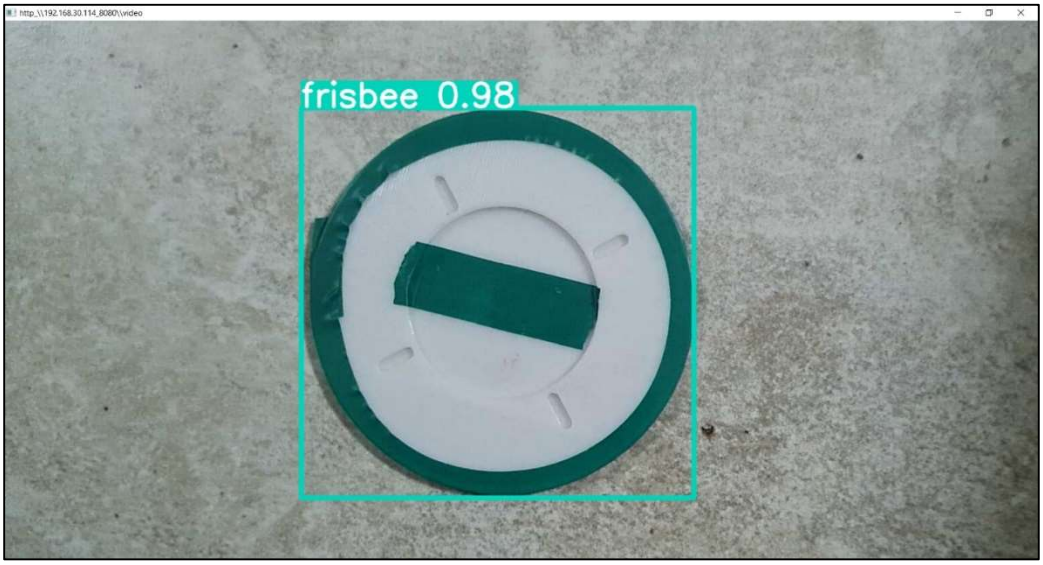
- Import necessary libraries and modules.
- Load the YOLOv8 model pre-trained on the COCO dataset.
- Initialize variables to store the position of the detected landmine.

Main Loop

- Continuously capture frames from the webcam.
- Process each frame using the YOLOv8 model to perform object detection.
- Identify and locate the detected landmine, if any, based on its class label.

Detection and Output

- For each frame, check if the detection includes an object classified as "frisbee," which is considered a proxy for a landmine.
- Calculate the center coordinates of the detected landmine and its confidence score.
- Print the type of detected object and its position coordinates.



4.2 Landmark Detection using OpenCV

4.2.1 Overview

The code provided performs two main tasks: detecting the presence of a red color in a video stream and detecting objects using a YOLO model. This report focuses on the color detection part, detailing the methods and techniques used to identify red objects within a video frame.

4.2.2 Color Detection Workflow

The process of detecting the red color in the provided code involves several key steps:

1) Frame Capture: Capturing frames from a video stream.

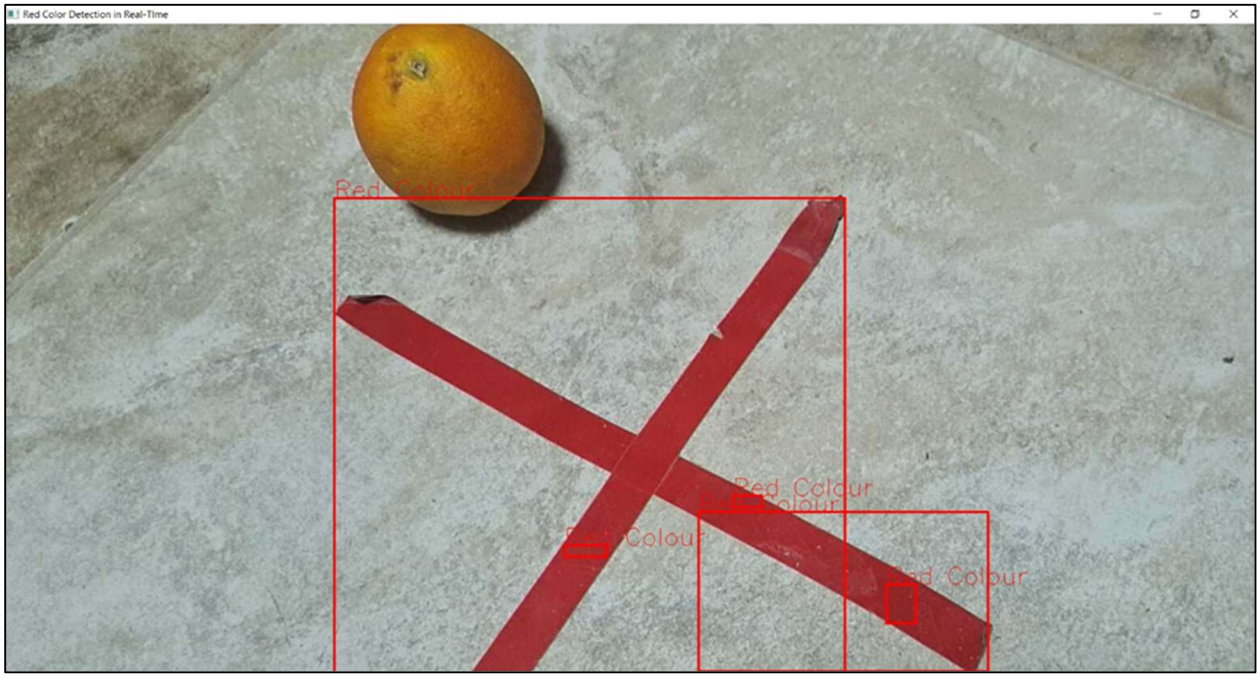
2) Color Space Conversion: Converting the captured frame from BGR to HSV color space.

3) Thresholding: Defining a range for the red color in HSV and creating a mask.

4) Morphological Transformations: Applying dilation to enhance the red regions in the mask.

5) Contour Detection: Identifying contours in the mask to locate red objects.

6) Bounding Box and Label: Drawing bounding boxes around detected red objects and labeling them.


Red Color Detection in Real-Time
Red Colour
Red Colour
Red Colour

V. Localization & Navigation

5.1 Localization Using Odometery Data & Landmarking

Localizing a differential drive robot relative to an enviroment is a huge challenge due to the absence of a Lidar or a depth camera. Thus we needed to localize to robot using wheel encoders and IMU. Robot odometry can be represented using the IMU and wheel encoders as the following chart:

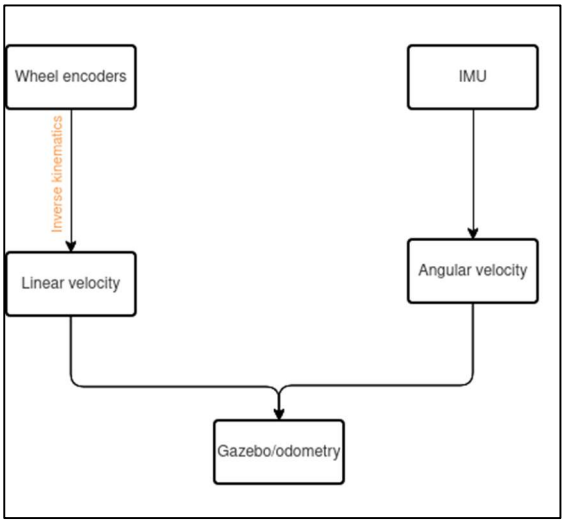


Figure 8 Robot Localization Flowchart

Throughout setting the Gazebo's simulation, we encountered some issues that we tried to resolve such as:

- The model rotating around itself in the environment without any input i.e. no feedback from the robot, yet the mode is rotating. This was due to the noise accompanied with the IMU readings. We resolved this by creating a filter for low readings to be zero. The threshold of the filter was found by printing the angular velocity (Gazebo's input) on the terminal and see the value at which the robot is stationary yet the angular velocity is changing.

- The model at the instant of robot stop, this was due to the fact that the friction parameters were not correctly configured in the Gazebo's plugin in the URDF file.

VI. Gazebo Simulation

6.1 Environment & URDF Model

In order to simulate the robot behavior with the sensors feedback data on gazebo, we first downloaded the raw pre-defined URDF TurtleBot Burger file from RosWIKI then we edited this file to include the Gazebo plugin and include the topics we run, which is the twist message published by the python node. We then created a launch file, which connects the URDF file with Gazebo and runs gazebo with the URDF file. Since we edited the URDF file and included our twist topic, the robot model on Gazebo works only when feedback data (encoder & IMU readings) are present.

6.2 Landmine Coordinates Marking

The final part of our project is to allow the robot to accurately mark the coordinates of the landmine detected on the Gazebo simulation with a specified marker, in our case a green circle. We achieve this by spawning an SDF model. Upon the detection of a landmine by our camera, we call a function called spawn_marker. This function calls the SDF file containing the marker model using the following command line:

```
spawn_sdf = rospy.ServiceProxy('/gazebo/spawn_sdf_model', SpawnModel)
```

the function then implements the model into gazebo using the following commands

```
resp = spawn_sdf("marker_model", sdf, "", pose, "world")
```

it takes the required pose to the specific coordinates of the landmine using the (x,y) coords of the robot at the moment it detected.

A screenshot from the gazebo environment containing the landmine detection marker could be found in figure.

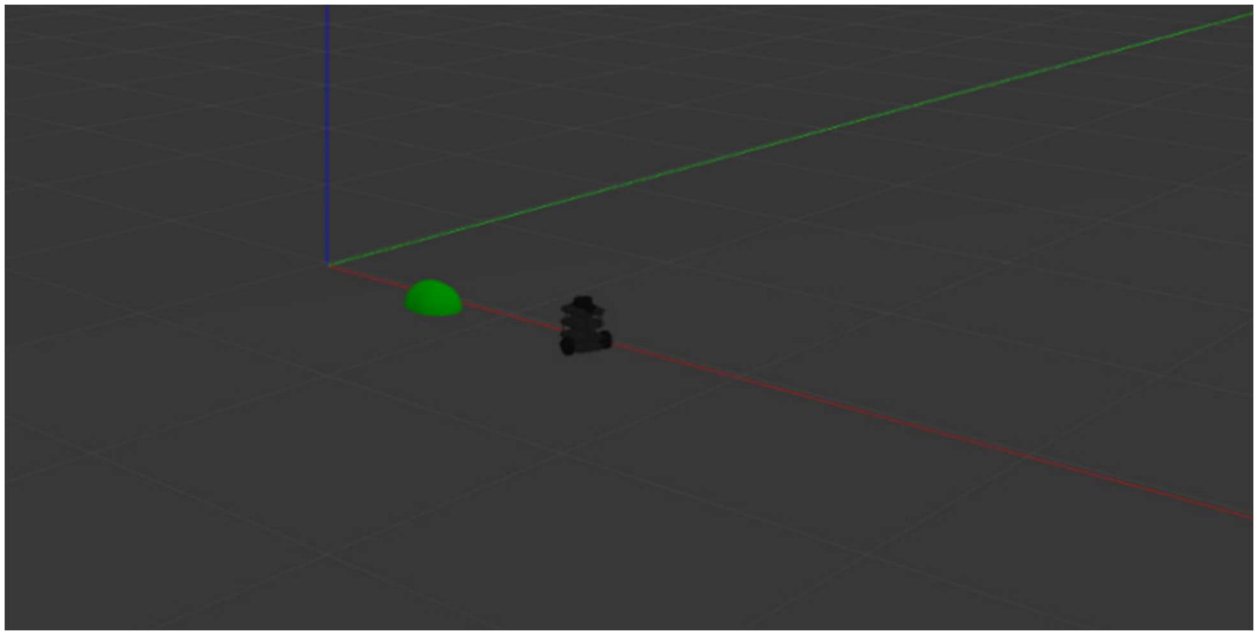
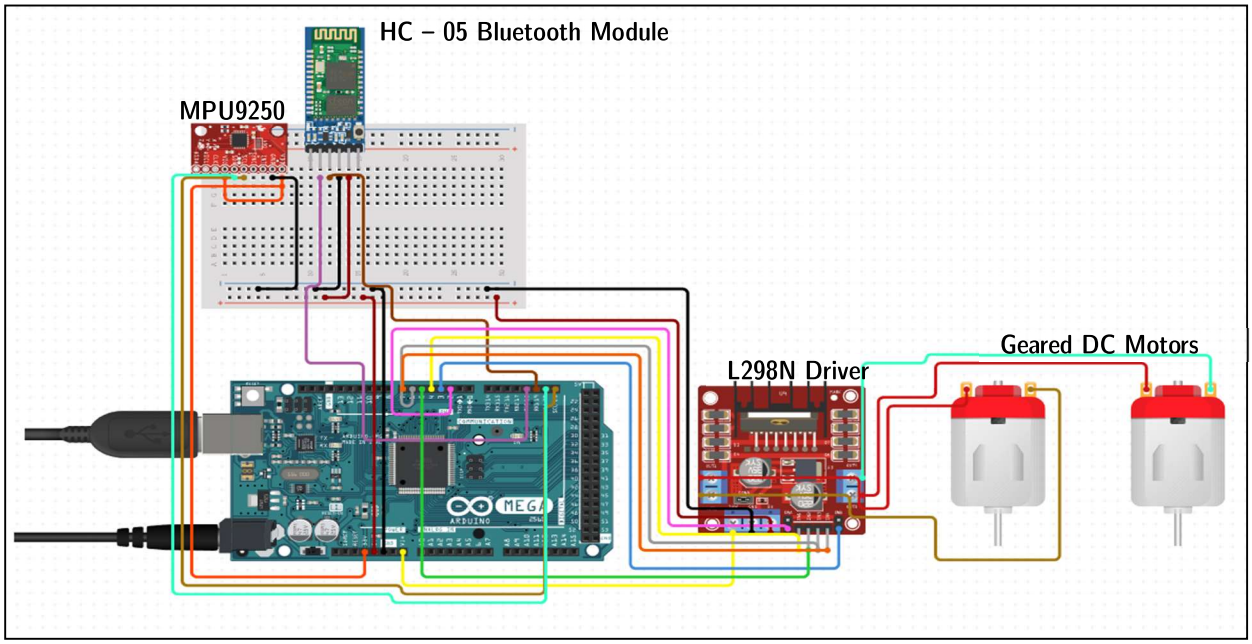


Figure 9 Landmine Marker Accurately Marked on Gazebo

VII. Appendix

7.1 Arduino Hardware Detailed Circuit



7.2 Project Budget List

Component	Unit Price	Quantity	Price
Arduino Mega Development Board	1450	1	1450
12V210RPM DC Motor with Encoder	700	1	700
55mm Wheels	300	2	600
HC-05 Bluetooth Module	250	1	250
MPU9250 IMU	750	1	750
12V 5A Battery	500	1	500
Laser Cut Robot Chassis Piece	135	5	675
12V Adapter	120	1	120
Total Price			5045