NYU | TANDON SCHOOL OF ENGINEERING

Advanced Mechatronics

ROB-GY-6103

Integrated Project - Autonomous Inventory Monitoring and Management Robot (AIMMR)

Team Members:

Rishabh Sivarajan(rs8426)

Kelvin Maliyakal(kpm8503)

Manan Malpani (mm11787)

Professor:

Dr. Vikram Kapila

Abstract

The Integrated Project involves the design and implementation of autonomous inventory monitoring and management robot (AIMMR) for a warehouse by identifying and counting defective and non-defective widgets. The robot will navigate using directional aids and count the widgets based on corresponding markers (ArUco tags) and display the count at the end of its course. The mock warehouse will have three sections, each with an object housing directional aids and a widget station. The robot needs to autonomously navigate based on the directional aids, detect the widget stations, and count the inventory at each station based on the tag ID. At the end of its course, the robot must reach its final destination without a path to follow and show the inventory count. This project has significant implications for warehouse management and can significantly improve inventory monitoring and management without any human intervention.

Table of contents

Introduction & Project Background

In today's business environment, effective inventory management is crucial to maintaining high levels of customer satisfaction. It is essential to have accurate inventory data that can help optimize the supply chain and ensure timely delivery of products. This is particularly true in a warehouse, where items need to be tracked efficiently to maintain their availability. The process of inventory management is, however, often manual, time-consuming, and prone to errors. Therefore, automation of the process can lead to increased efficiency and accuracy.

The objective of this project is to design and build an ***Autonomous Inventory Monitoring and Management Robot(AIMMR)*** for a warehouse by identifying and counting defective and non-defective widgets. The robot will navigate on the warehouse floor using directional aids to make turn decisions and will count the number of defective and non-defective widgets hosted by the corresponding markers (ArUco tags). The robot will display the number of defective and non-defective widgets at the end of its course.

The project's mock warehouse will have a layout with 3 major sections that will contain widgets of different types and sizes. The robot will have to navigate through the sections, detect the widget stations, and count the inventory at each station based on the tag ID. At the end of its course, the robot must reach its final destination, without a path to follow, and show the inventory count.

The robot in this project uses a combination of two boards, a Propeller microcontroller, and a Raspberry Pi 4 Model B. The Propeller microcontroller is responsible for line following and object detection while the Raspberry Pi 4 B is responsible for computer vision (CV) tasks. By combining the capabilities of the Propeller microcontroller and Raspberry Pi, this autonomous robot can navigate the warehouse floor, detect widget stations, and count inventory.

The Propeller microcontroller is interfaced with the robot's sensors to detect lines on the floor and obstacles in its path while also detecting the stations where the widgets are placed. The robot uses these inputs to follow a pre-defined path. Additionally, the Propeller microcontroller is also responsible for detecting the presence of widgets at each station. It identifies the widgets using sensors that detect the tags placed near the stations and records the count of defective and non-defective widgets.

The Raspberry Pi 4 B is used for computer vision tasks. The robot uses the Pi camera module to capture images of the environment. The Pi then processes these images to detect and recognize the ArUco tags used to identify the widget stations. This information is then used by the Propeller microcontroller to calculate the inventory count.

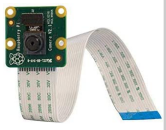
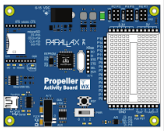
This integration of both boards allows for efficient and accurate inventory management, which can reduce manual labor, increase efficiency, and improve inventory accuracy in a warehouse setting.

Apparatus

The project required us to build an AIMMR and some of the parts were sourced from the previous Mechatronics project. Additionally, a Propeller activity board, Raspberry Pi 4 B, and its complementary components were purchased and the remaining components were taken from an Arduino Kit.

Bill of Materials(BOM)

Serial No.	Part Name	Quantity	Part Image	Unit Cost($)	Total Cost($)
1	Parallax Screwdriver	1		1.5	1.5
2	Resistor(220Ω)	3		0.33	0.99
3	RGB LED	1		2	2
4	IR Sensor	4		0.6	2.4
5	Ultrasonic Sensor	2		29.71	59.42
6	Machine screws, Phillips	4		0.25	1
7	Hex Nuts	4		0.14	0.56
8	Tail wheel ball	1		3.95	3.95
9	Boe-Bot aluminum chassis	1		29.99	29.99

10	Battery holder with cable and barrel plug	1		4.99	4.99
11	Cotter Pin	1		0.3	0.3
12	Plastic wheels	2		2	4
13	Continuous rotation servo	2		17.95	35.9
14	9V Battery	1		3.18	3.18
15	9v Battery Clip	1		5.99	5.99
16	Power bank	1			0
17	AA Batteries	4		1.33	5.32
18	Raspberry Pi 4 Model B	1		148.99	148.99
19	Raspberry Pi Camera Board v2 - 8 Megapixels	1		29.95	29.95
20	Propeller Activity Board	1		82.22	82.22

21	Breadboard mini	1		3.95	3.95
22	Data Cable	1		7.99	7.99
23	Jumper Cables	31		0.06	1.86
24	LCD Display	1		34.95	34.95
25	USB-C, 5.1V, 3A, US Plug	1		11.94	11.94
26	32GB Ultra microSDHC UHS-I Memory Card	1		7.39	7.39

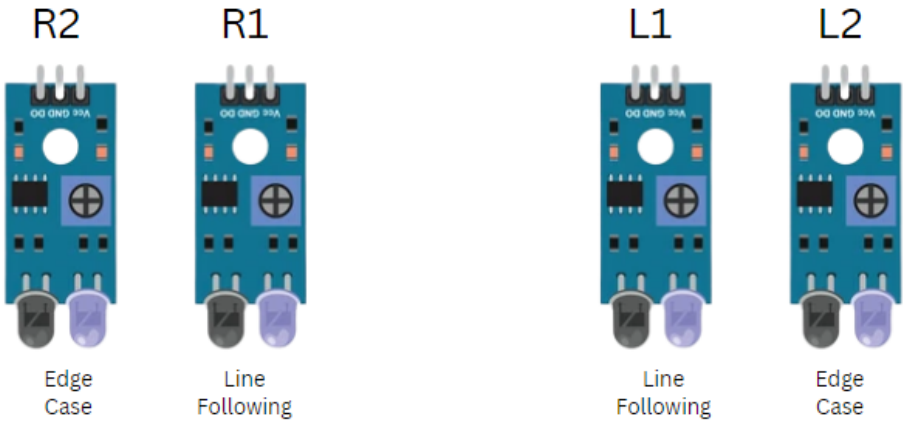
BOM: Table 1

Total cost: $490.73

Design Layout

In this design layout, the robot is equipped with four infrared (IR) sensors in the front, two Ultrasonic sensors on either side and a camera.

Infrared sensors(IR):



The 2 primary IR sensors (R1 & L1) are used for the line-following task, these are used to correct the direction of the robot for the general line-following applications. The 2 IR sensors placed on the extreme corners (R2 & L2) are used for the edge case scenarios.
For example:
If the robot turns right at intersection i1 and once it reads the widget data from A1 it must traverse back to B1. To reach B1 the robot will have to cross i1 again, in order to ensure the robot can differentiate between the i1 intersection and the interaction to enter B these IR sensors are used. At intersection i1 both L1 and L2 would get triggered which would allow the robot to know that this is an intermediate intersection.

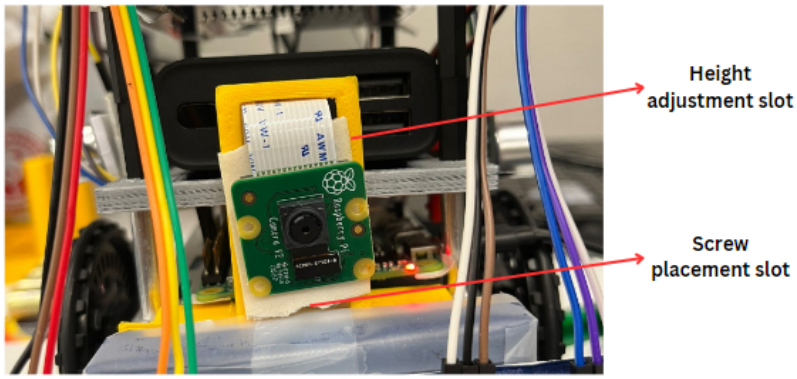
Ultrasonic:
The ultrasonic sensors are placed on either side of the robot above the wheel at a height to read the widgets at stations that are placed on a platform of a height of 50 mm.

RGB LED:
Blue: is used to indicate the intersections.
Red: is used to indicate defective widgets.
Green: is used for indicating non-defective(good) widgets.

LCD Display:
The LCD display is used at the end of the no-mans land to display the total number of defective and non-defective widgets.

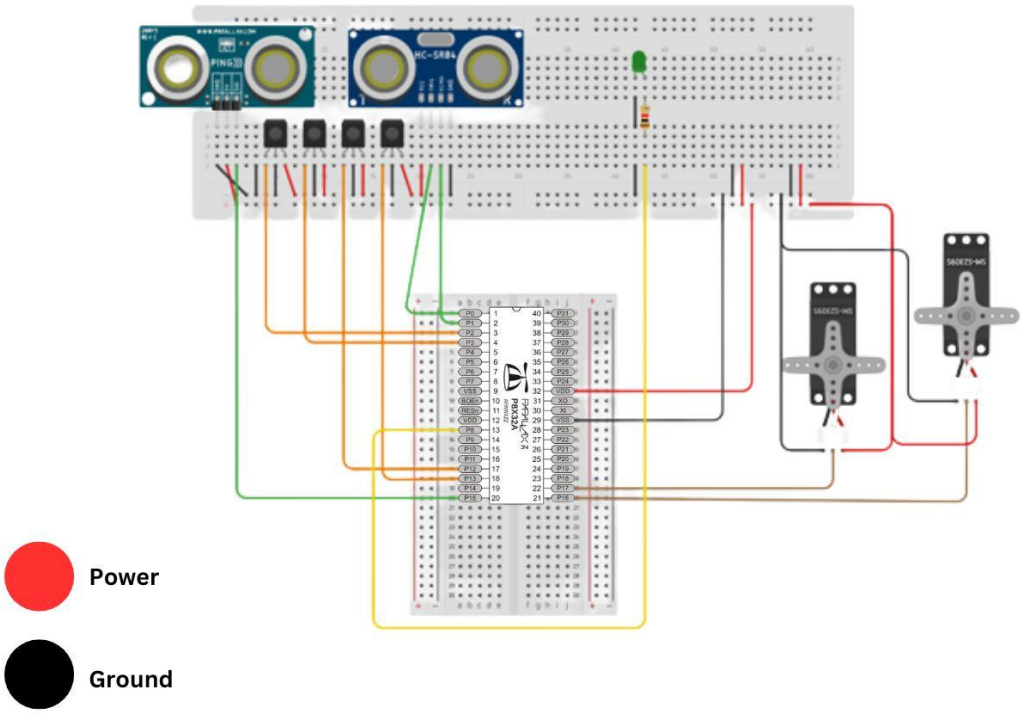
Camera:

For this project, the team has used a Raspberry Pi Camera Module V2-8 Megapixel,1080p (RPI-CAM-V2). It has been placed on a 3D-printed structure which can be secured on the aluminum chassis using a screw and nut. The design ensures that it remains lightweight but can securely mount the camera at the desired height based on how far the marks are placed.



Circuit Diagram

This is the circuit diagram for the sensors and actuators connected to the Propeller microcontroller.

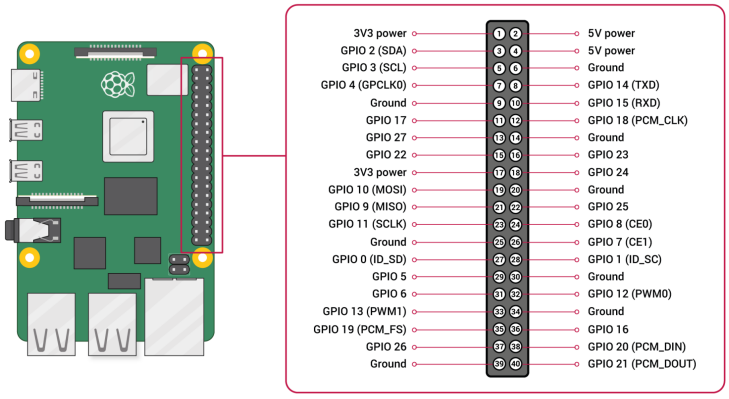


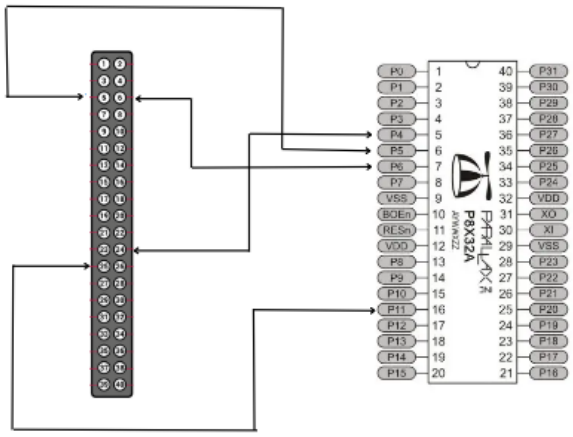
Approach

Raspberry Pi

A Raspberry Pi(RPi) is a small, affordable computer board designed for educational and hobbyist purposes. It can be connected to a monitor, keyboard, and mouse to function as a desktop computer. However, due to its small size and low power consumption, it can also be used for a wide range of projects that require high computational power, such as **computer vision**, machine learning, and robotics. For this project, the team has used a raspberry pi for the image processing functionality. The Pi communicates with the propeller board using the GPIO pins.

While the propeller board controls the IR sensors and ultrasonic sensors and the RPi controls the camera. When the propeller board is at any of the intersections in lane A, it communicates to the RPi to begin reading the directional aids. Once the RPi determines the direction it sends a return message to the propeller to initiate the desired movement. Similarly, the RPi and Propeller boards communicate with each other for the widget indication case and in the no-mans-land scenario.





RPie & Propeller communication connection

Raspberry Pi	Propeller
24	4
25	11
6	6
5	5

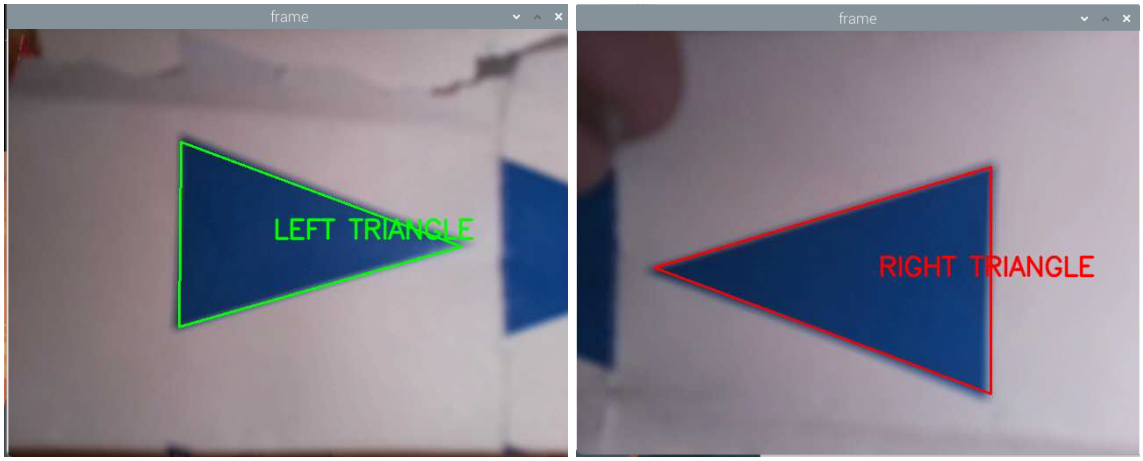
Open CV

OpenCV, or Open Source Computer Vision Library, is an open-source software library that includes hundreds of computer vision and machine learning algorithms. For this project, there were two major tasks to be performed by the open CV libraries.

1. Direction Markers:

For this project, a triangular direction marker was used, which allows the robot to decide whether to turn right or left. The library used was

aruco_dict = aruco.Dictionary_get(aruco.DICT_6X6_250)



Code Logic:

1. Import the necessary libraries, OpenCV and NumPy.
2. Initialize the Pi Camera by creating a VideoCapture object.
3. Start an infinite loop for capturing frames from the camera.
4. Capture a frame using the read() function.
5. Convert the frame from BGR color space to HSV color space.
6. Define the range of blue color in HSV using NumPy arrays.
7. Threshold the image to get only blue colors using the inRange() function.
8. Apply morphological operations such as erosion and dilation to reduce noise in the image.
9. Find contours in the thresholded image using the findContours() function.
10. Loop over the contours using a for loop.
11. Approximate the contour as a polygon using the approxPolyDP() function.
12. If the polygon has three sides, calculate the centroid of the triangle.
13. If the centroid is to the left of the frame, draw a green triangle and print "LEFT TRIANGLE" text on the frame.
14. If the centroid is to the right of the frame, draw a red triangle and print "RIGHT TRIANGLE" text on the frame.

15. Display the resulting frame using the imshow() function.
16. If the 'q' key is pressed, break the loop.
17. Release the Pi Camera and close all windows using the release() and destroyAllWindows() functions.

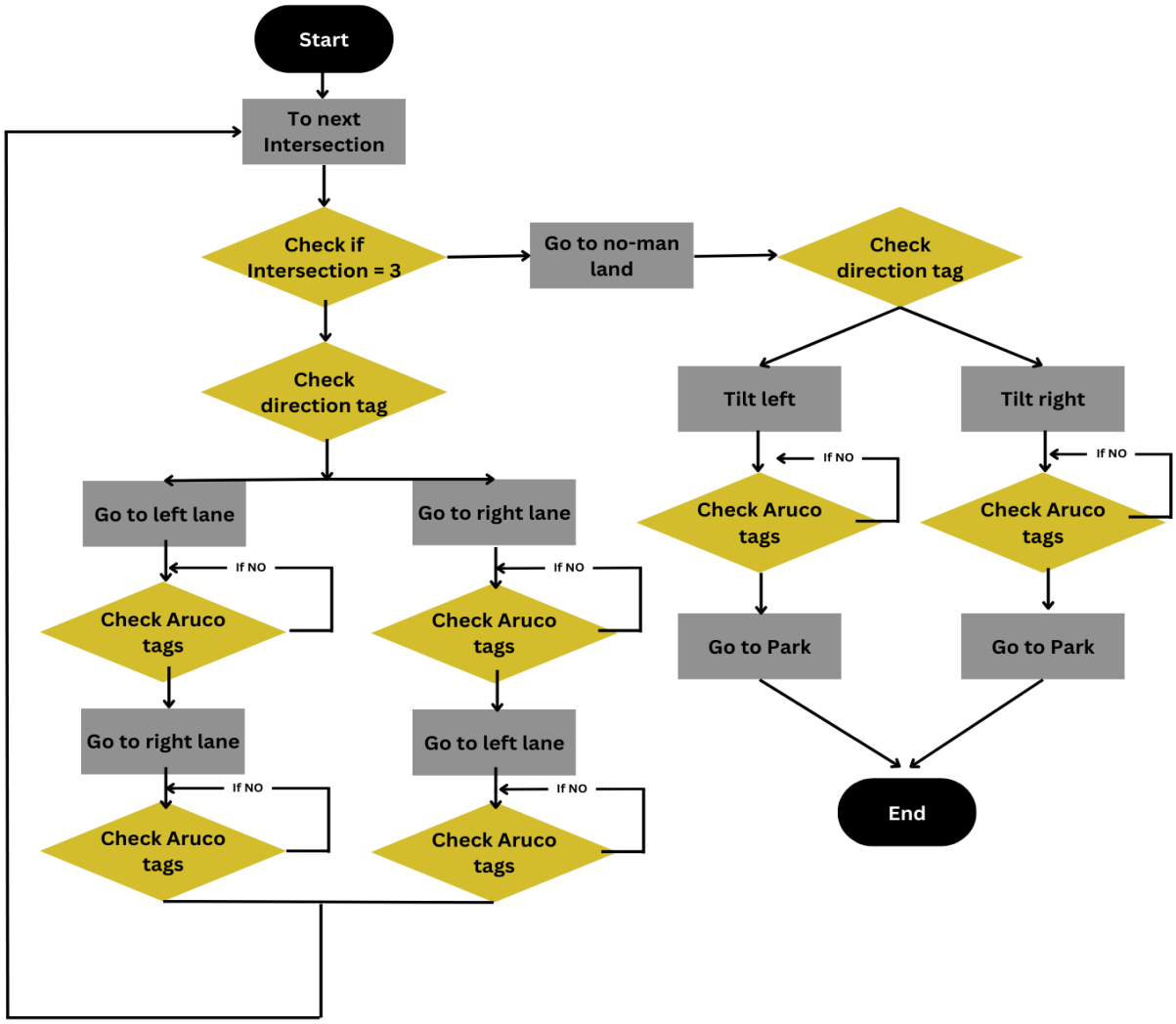
2. Aruco:

Aruco is a popular library for augmented reality (AR) applications. It provides a set of tools for detecting, tracking, and analyzing markers in real-time using computer vision techniques. The library uses square markers with a unique code, which are detected and tracked in the camera frame.

Code Logic:

1. Import libraries - cv2, cv2.aruco, numpy, and time
2. Set the capture device, '0' refers to the default camera.
3. Load the aruco dictionary
4. Initialize the detector parameters
5. Start a loop that will continuously capture frames from the camera
6. Convert the frame from the camera to grayscale
7. Detect the markers using the detectMarkers() function from the aruco module
8. Draw the detected markers on the frame
9. Display the resulting frame with imshow() function from cv2
10. Wait for a key press and check if the 'q' key was pressed. If 'q' key is pressed, exit the loop.
11. Release the capture and destroy all windows.

Workflow



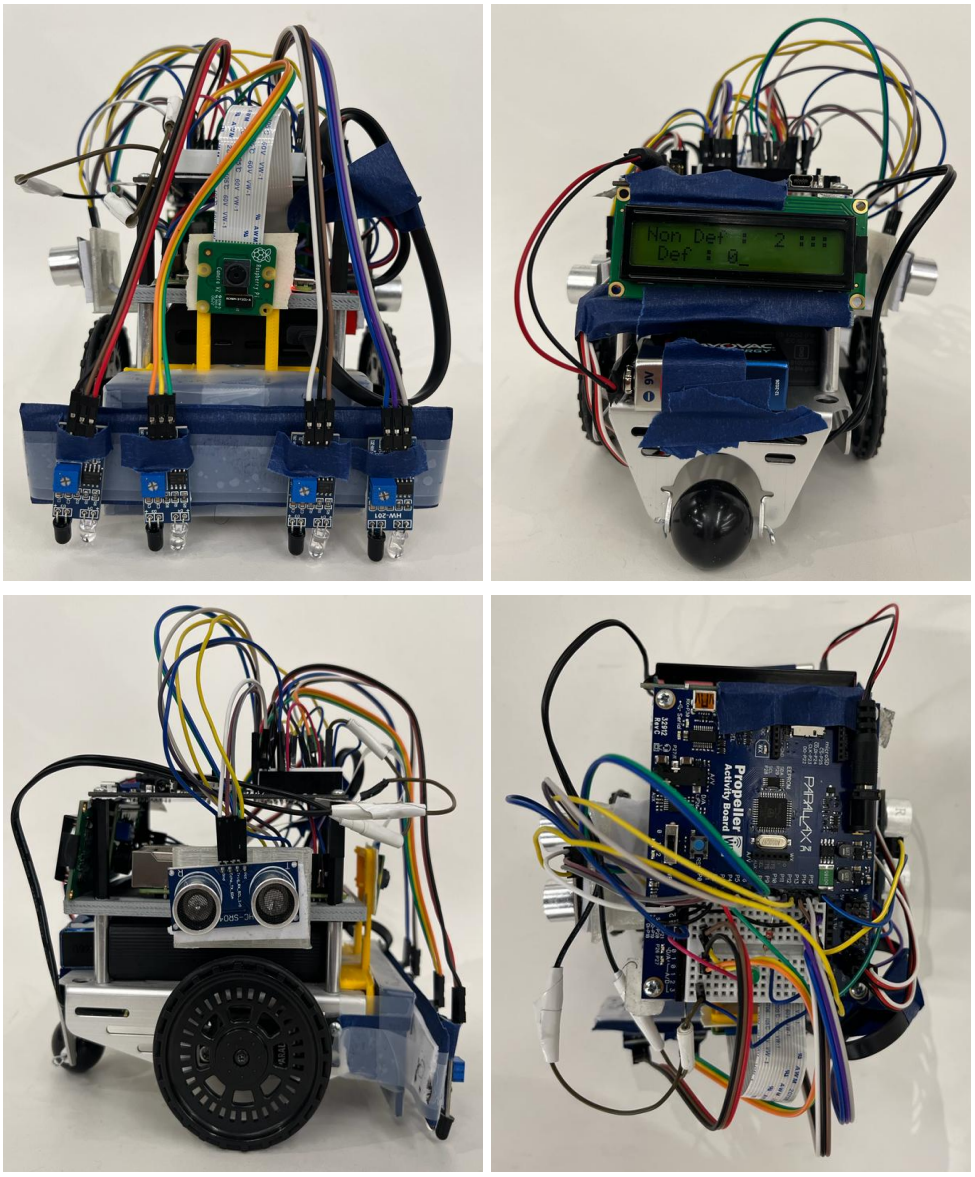
Method of Approach and Testing the Robot

- Developed the line-following code and integrated the IR sensors on the Propeller board.
- Write additional code for the edge case to avoid the junction when moving from the lane (A-to-B or B-to-A).
- Incorporate two additional sensors for the above point.
- Program the logic for intersection detection, and ignoring certain intersections.
- Developed and deploy the code for the direction markers and arco identification.
- Debugging the issues and calibrating based on the map and lighting.
- Developed and deploy the code for the no man's land implementation.
- Debugging the issues and calibrating based on the map and lighting.

Problems Faced

A. The team faced some challenges with creating serial communication between the Raspberry Pi and the Propeller board and therefore decided to use more rudimentary logic high/low interfacing between the processors.
B. Another challenge was a design change, in order to ensure smoother movement and be able to ignore the t-intersections when the robot moves between lanes A and B. This was resolved by using 2 additional IR sensors placed on the outer side of the primary IR sensors.

Robot Images



Scope of Improvement

A. The red light from the IR sensors could have disturbed the accuracy of the readings of the Pi Camera. In future iterations, it would be beneficial to design a casing around the IR sensors. For this project, the team used some dark masking tape to cover the lights.
B. Ensure there is no static to damage the camera. In the future, the camera and its operations would be safer if it would be protected by some form of Electrostatic Discharge (ESD) or alternatively using some form of shielding that would protect the camera and other sensors.

Conclusion

We developed an Autonomous Inventory Monitoring and Management Robot (AIMMR)that follows a line on the following grid. The AGV uses 4 IR sensors for the line following task and two ultrasonic sensors to identify locations for the widgets in lanes A and B respectively.

The tea mused Open Cv on libraries on the Raspberry Pi to identify Defective/Non-defective widgets and indicate them on an LCD display connected to the Raspberry Pi. The robot was also able to go to the no man's land where it identifies the direction tag and then moves to the final widget with the help of just Open CV. While moving through the map the robot was able to indicate the intersections (i1…i6) with a green-colored LED.

This project required us to interface the Raspberry Pi and the Propeller microcontroller while managing different tasks and communicating task-based information between the two points of processing.

The team was successful in performing all the requirements for this project.