UNIVERSITI MALAYSIA PERLIS

VISINERGY
EMPOWERED BY INNOVATION

From Fundamentals to Autonomy: Developing LabVIEW with NI DANI Robotics Kit 2.0 using myRIO

Name : Irfan Nur Raziq Bin Mohd Hairudin
No Matric : 211502829
Supervisor University : Dr. Mohd Wafi Bin Nasrudin
Supervisor Industry : Encik Mohamad Arif Bin Alias
Panel : Encik Mohd Hafizi Bin Omar

Table of Contents

01

Introduction

From Fundamentals to Autonomy:

Developing LabVIEW with NI DANI Robotics Kit 2.0 using myRIO

PROJECT BACKGROUND

- The myRIObot is a differential drive robot initially controlled remotely using LabVIEW software. Users can operate it without direct visual contact through a Human Machine Interface (HMI).

- To transform the myRIObot from a tele-operated system to an advanced autonomous mobile robot, starting with fundamental LabVIEW programming and progressing to complex autonomous navigation using dead reckoning and fuzzy logic control.

- To demonstrate reliable autonomous navigation, showcase the potential of integrating NI DANI Robotics Kit 2.0 with myRIO for intelligent systems.

02

Objective

From Fundamentals to Autonomy:

Developing LabVIEW with NI DANI Robotics Kit 2.0 using myRIO

OBJECTIVE

01

To integrate multiple hardware components and ensure effective communication and control among them using myRIO as a microcontroller.

02

To understand and implement the fundamentals of LabVIEW programming

03

To develop an autonomous mobile robot that integrates dead reckoning for navigation and fuzzy logic for control system.

03

Problem Statement

From Fundamentals to Autonomy:
Developing LabVIEW with NI DANI Robotics Kit 2.0 using myRIO

PROBLEM STATEMENT

01

Difficulty in real-time location tracking, especially with weak GPS signals.

02

Challenges in quick and accurate obstacle detection

03

Difficulty in maintaining accurate positioning in constantly changing surroundings.

04

Project Scope

From Fundamentals to Autonomy:

Developing LabVIEW with NI DANI Robotics Kit 2.0 using myRIO

RESOURCES

- NI myRIO student embedded device as the central control system
- NI DANI Robotics Kit 2.0 for the robot's chassis and differential drive
- Various sensors including Encoders for dead reckoning, IR Range Finder obstacle detection and GPS modules NEO 6M for navigation.
- LabVIEW for programming the control systems and Human-Machine Interface (HMI)
- Power supply components (LIPO batteries, DC motors, motor drivers)

DELIVERABLES

- A fully functional robot capable of navigating autonomously using dead reckoning and fuzzy logic-based control systems, where fuzzy logic controls the robot's speed by using an IR range finder.

- Various VI (Virtual Instrument) programs developed for sensor integration, motor control, obstacle avoidance, and autonomous navigation.

- An interactive HMI for remote control and monitoring the robot, facilitating user interaction with the system.

PROJECT ROADMAP AND TIMELINE

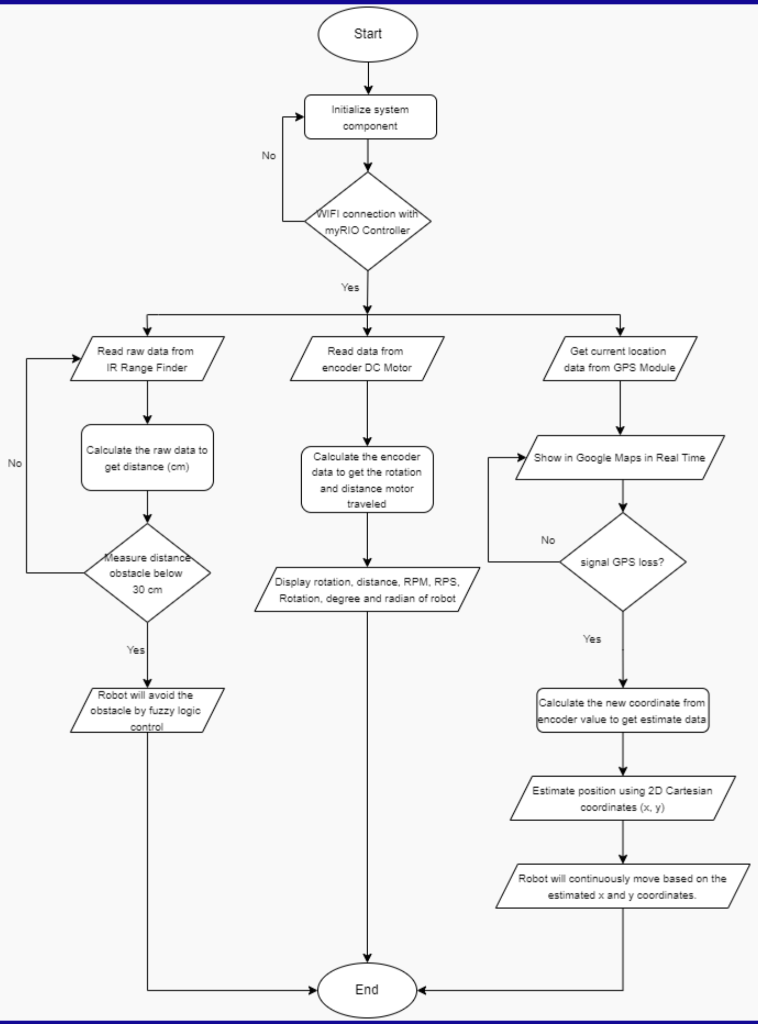
PHASE	OBJECTIVE	EXPECTED
1	Establish project foundations through planning	June
2	Design both hardware and software components, including wiring designs and component selection.	July
3	Integrate power systems, motors, encoders, IR Range Finder, and GPS with the myRIO controller.	August
4	Set up the development environment, program motor and sensor integrations.	August
5	Implement autonomous features, including obstacle avoidance, fuzzy logic integration, and dead reckoning.	September
6	Conduct extensive testing of the system's	September
7	Finalize project documentation and prepare for the presentation	September

05

Flow Chart

From Fundamentals to Autonomy:

Developing LabVIEW with NI DANI Robotics Kit 2.0 using myRIO

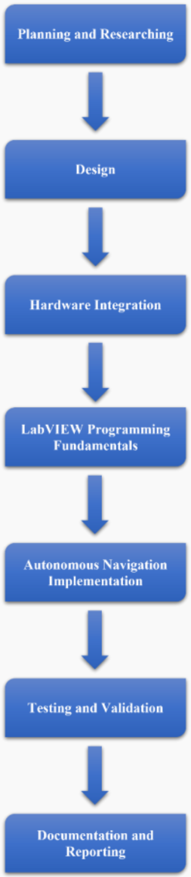

Start
Initialize system component
No
WIFI connection with myRIO Controller
Yes
Read raw data from IR Range Finder
Read data from encoder DC Motor
Get current location data from GPS Module
Calculate the raw data to get distance (cm)
No
Calculate the encoder data to get the rotation and distance motor traveled
Show in Google Maps in Real Time
Measure distance obstacle below 30 cm
No
signal GPS loss?
Display rotation, distance, RPM, RPS, Rotation, degree and radian of robot
Yes
Yes
Robot will avoid the obstacle by fuzzy logic control
Calculate the new coordinate from encoder value to get estimate data
Estimate position using 2D Cartesian coordinates (x, y)
Robot will continuously move based on the estimated x and y coordinates.
End

06

Methodology

From Fundamentals to Autonomy:

Developing LabVIEW with NI DANI Robotics Kit 2.0 using myRIO


Planning and Researching
Design
Hardware Integration
LabVIEW Programming Fundamentals
Autonomous Navigation Implementation
Testing and Validation
Documentation and Reporting

1. Planning and Researching

The project planning phase establishes the scope and title focused on robotics autonomy, develops a detailed proposal outlining goals, timeline, resources, and challenges, drafts a requirement document listing necessary hardware and software like LabVIEW and myRIO, creates a Gantt chart for task management, and conducts a comprehensive literature review on autonomous mobile robotics technologies and methodologies.

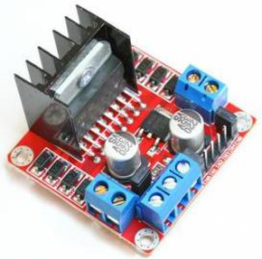
2. Design

Component Selection

- Robot frame using NI DANI Robotics Kit 2.0 from PITSCO
- MyRIO-1900 Student Embedded Device
- LIPO Battery 11.1V 5200mAh
- Motor Driver L298N
- IR Range Finder
- GPS Module NEO-6M
- PCB Terminal Block 5mm Pitch
- 2-pin rocker switch
- Cables
- PITSCO Education 12 VDC motors (x2)

    -152 rpm and 300 oz-in. of torque

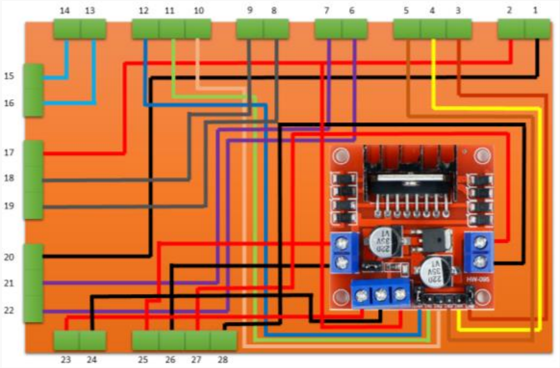
    Optical quadrature encoders with 400 pulses per revolution

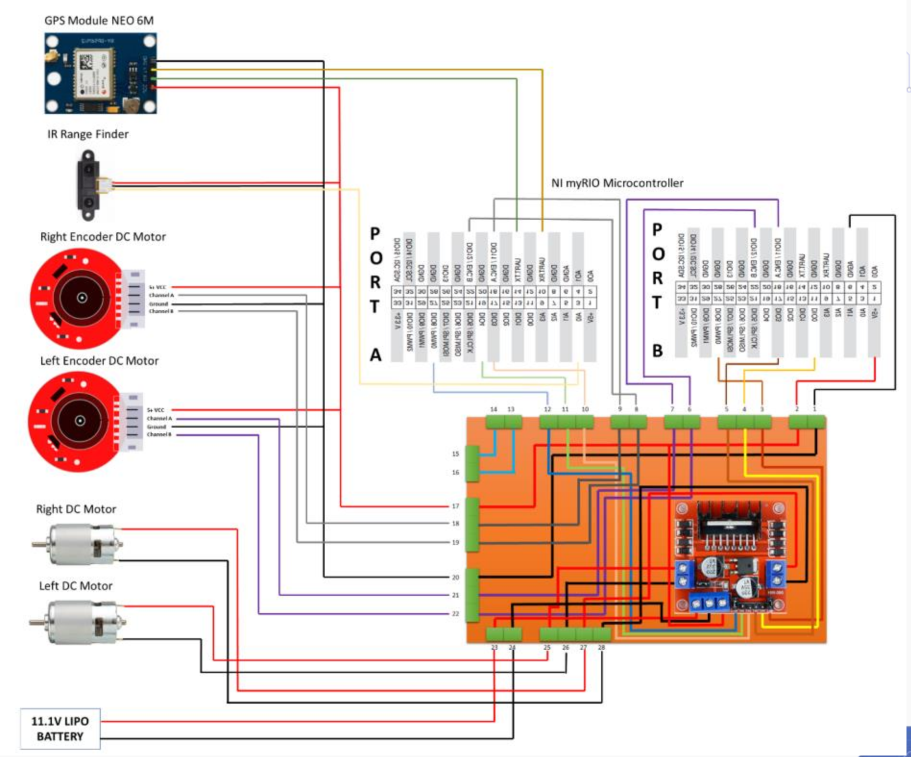


2. Design

Wiring Diagram Design

- Integrated Custom Board

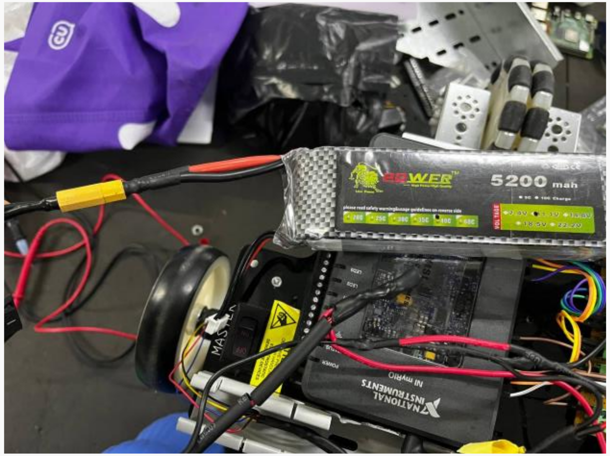




3. Hardware Integration

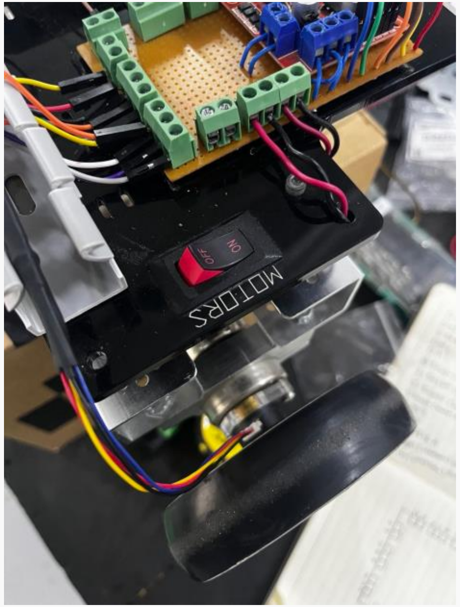
Power Supply Integration

- The robot’s power system is designed using a LiPo (Lithium Polymer) battery, which now supplies 12VDC directly to power the DC motors and other components.



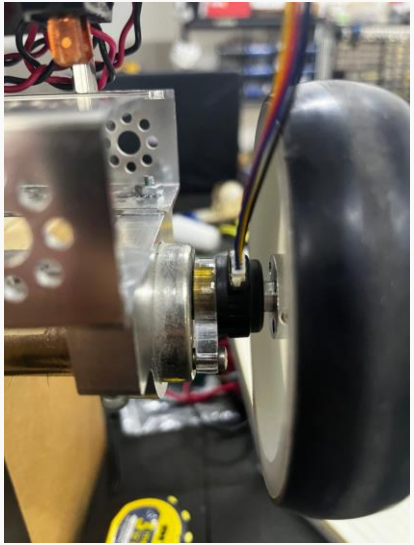
3. Hardware Integration

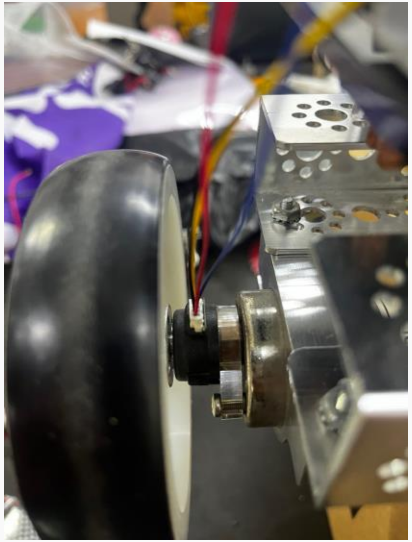
DC Motor Integration



3. Hardware Integration

Encoder DC Motor Integration

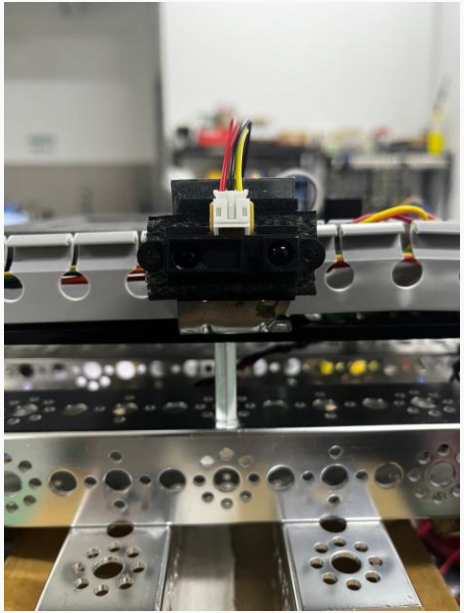




Integrated Board Pin	Description	myRIO Pin
6	Encoder Channel B	B (22/ENC.B)
7	Encoder Channel A	B (18/ENC.A)
8	Encoder Channel B	A (22/ENC.B)
9	Encoder Channel A	A (18/ENC.A)

3. Hardware Integration

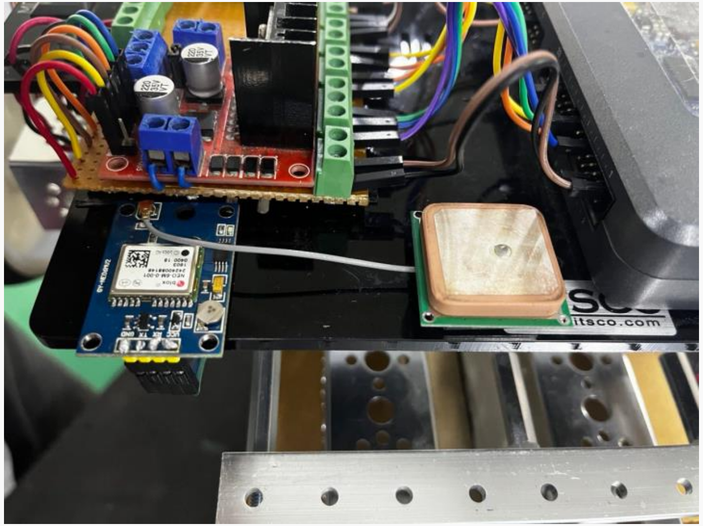
IR Range Finder Integration



Wire Color	Description	myRIO Pin
Red	+5V Supply	B (1/+5V)
Black	GND	B (6/AGND)
Yellow	Analog Signal	B (3/AI0)

3. Hardware Integration

GPS Module NEO 6M Integration



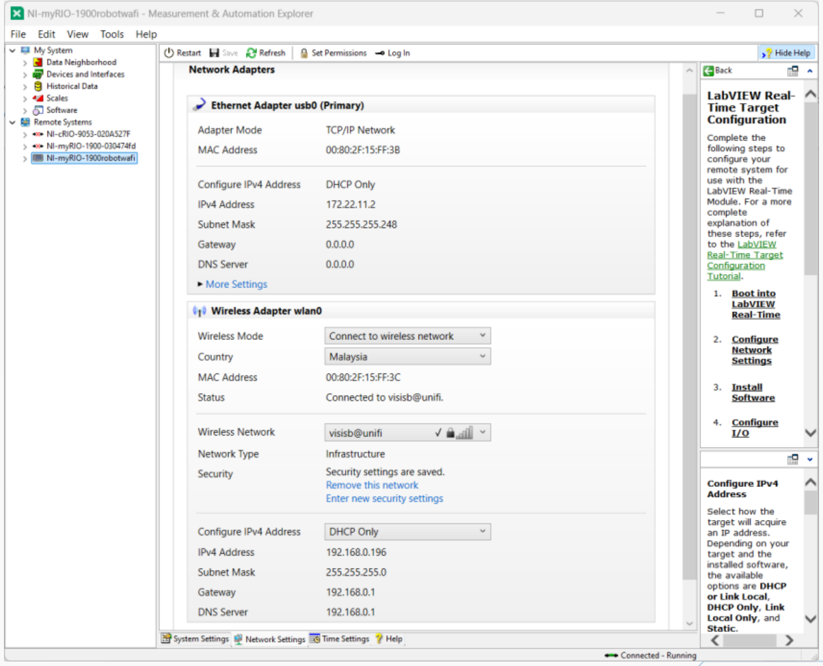
Description	myRIO Pin
+3.3 V Supply	A (33/+3.3V)
GND	A (30/GND)
RX	A (14/UART.TX)
TX	A (10/UART.RX)

4. LabVIEW Programming Fundamentals

Setup Software Development

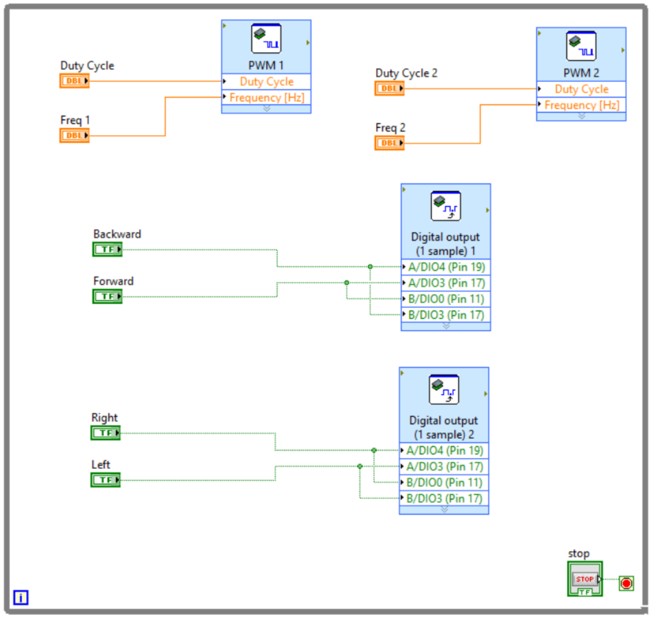
4. LabVIEW Programming Fundamentals

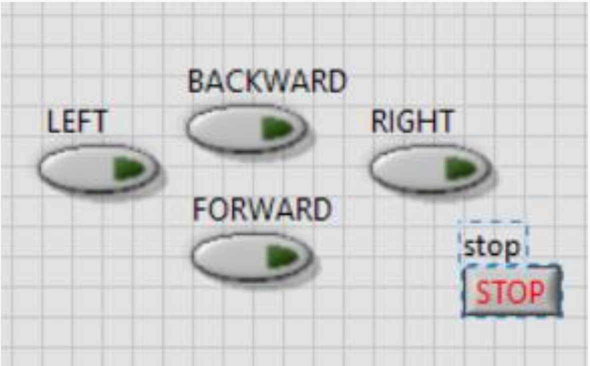
Setup Wireless connection to NI myRIO



4. LabVIEW Programming Fundamentals

Basic Programming for motor control





5. Autonomous Navigation Implementation

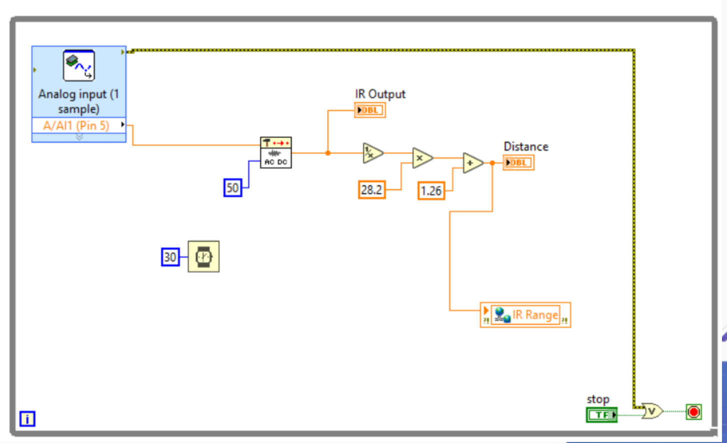
Develop program to read IR Range Finder Program

To implement that, we need to find the distance and use the formula below and apply it in LabView.

$$d = K_s\left(\frac{1}{V_o}\right) + K_o$$

Where:

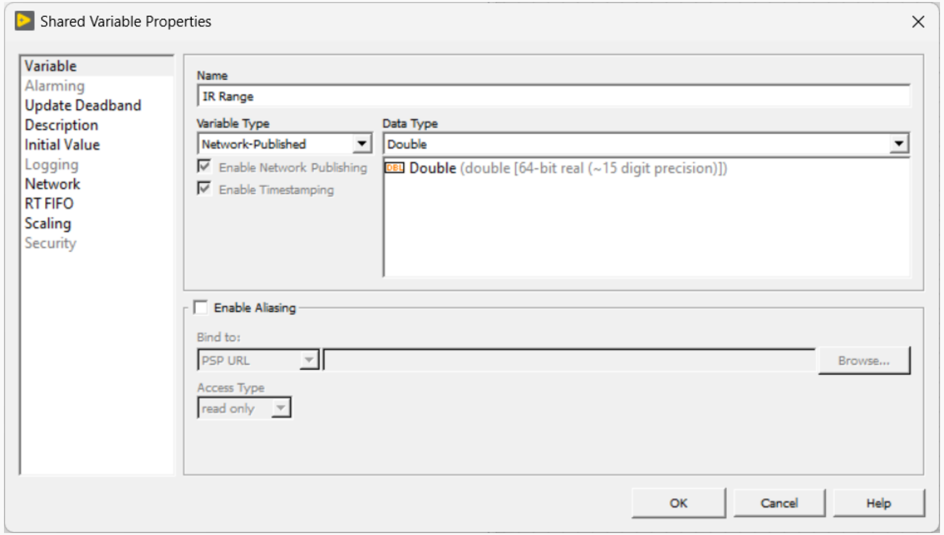
- d = distance (cm)
- K_s = 28.2 cm/V (sensitivity constant)
- V_o = output voltage from the IR sensor
- K_o = 1.26 cm (offset constant)



5. Autonomous Navigation Implementation

Create Network Shared Variable Function

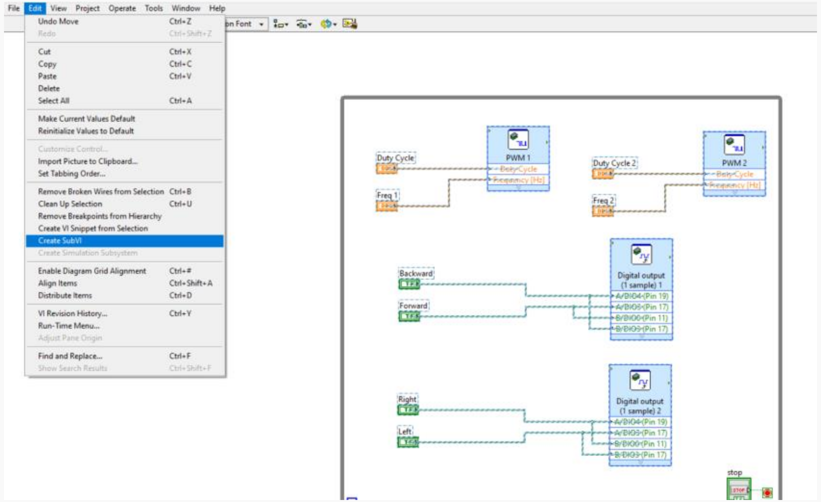
The shared variable is hosted on a network, making the data available for distributed systems, enabling other systems to read in real-time and act upon it.

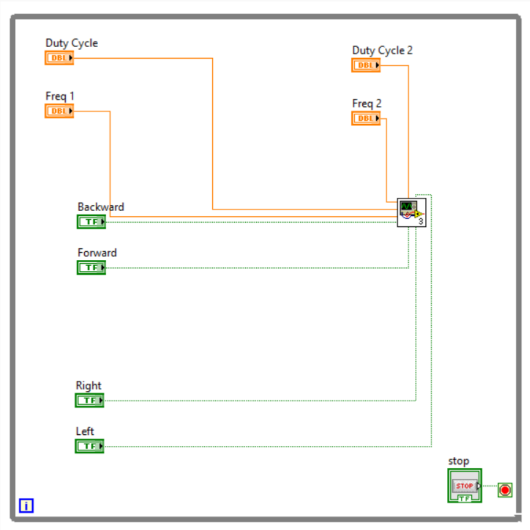


5. Autonomous Navigation Implementation

Create Sub VI Program for Motor Control

Creating the Motor Control Program Sub VI enhances the program's modularity, making it easier to maintain and reuse





5. Autonomous Navigation Implementation

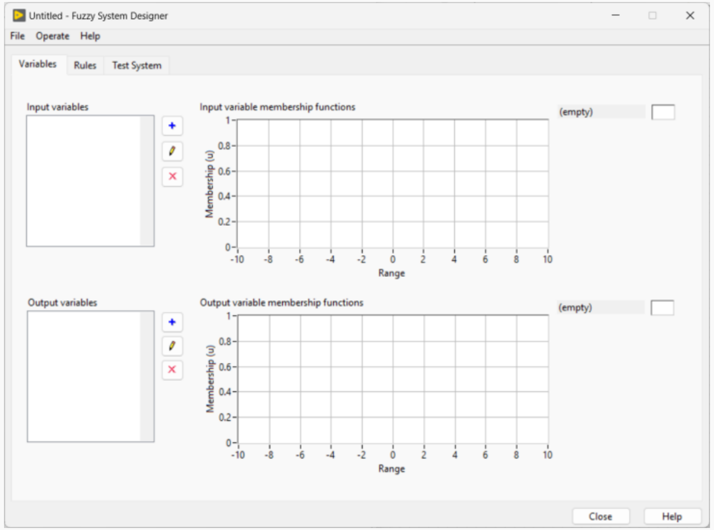
Fuzzy Logic

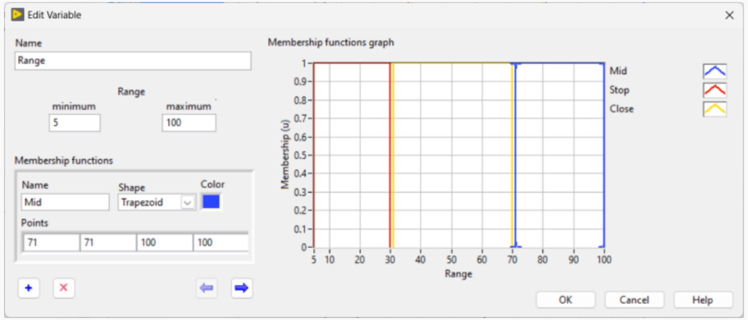
Fuzzy logic is used to control the speed of the robot's DC motor based on input from an IR sensor. When the IR sensor detects an object approaching closely, the fuzzy logic system automatically reduces the motor speed.

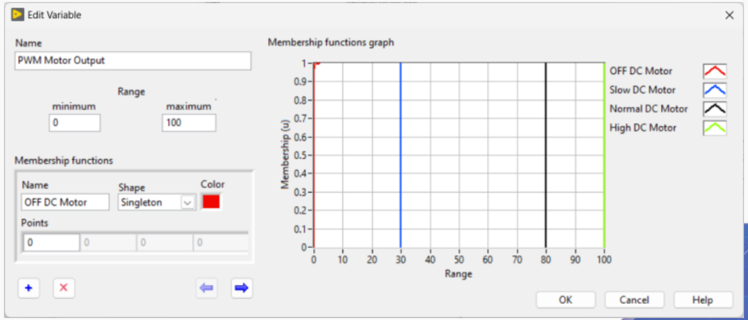
Range Distance (cm)	Speed (%)
5-30	0
31-70	30
71- 100	80
101 - above	100

5. Autonomous Navigation Implementation

Develop fuzzy logic using Fuzzy System Designer

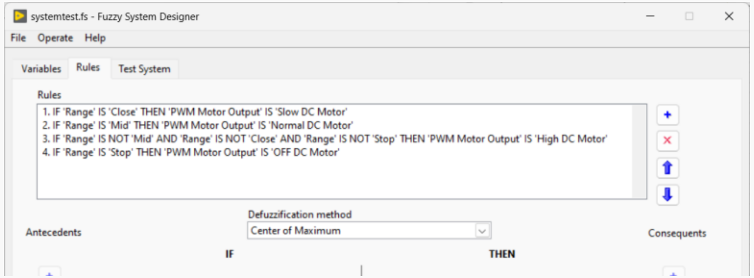


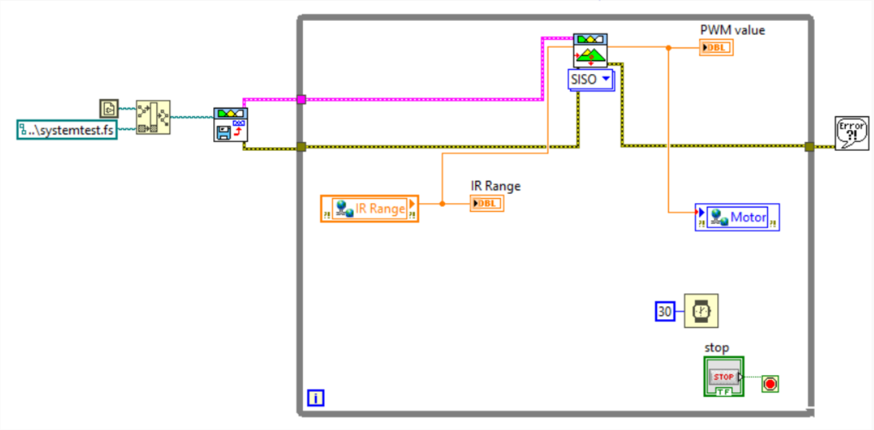


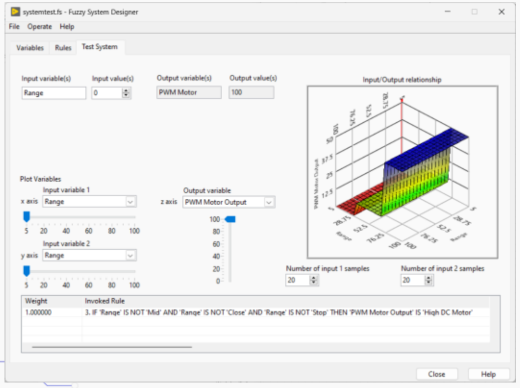


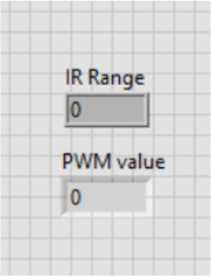
5. Autonomous Navigation Implementation

Develop fuzzy logic using Fuzzy System Designer









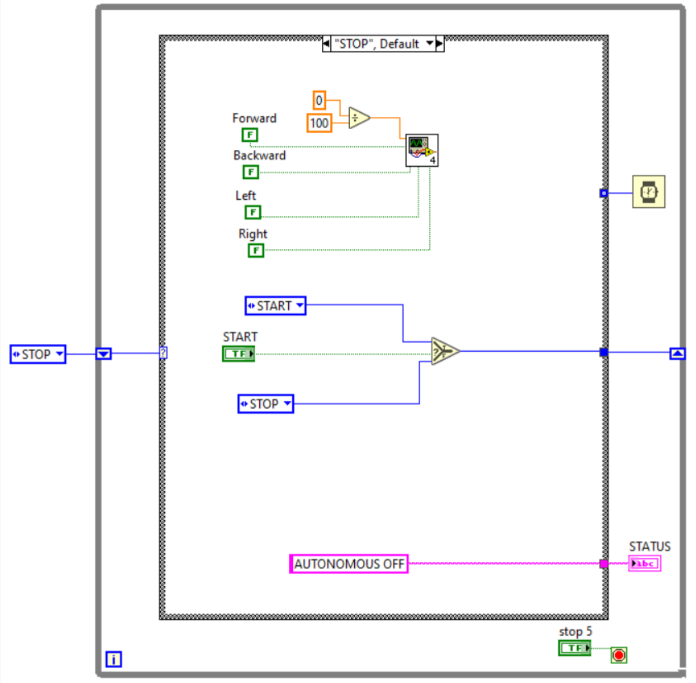
5. Autonomous Navigation Implementation

Obstacle Avoidance

A program was developed to integrate the Infrared (IR) Range Finder with the motor control system to enable real-time obstacle detection and avoidance for the robot. The system uses fuzzy logic to dynamically adjust the motor's duty cycle based on the proximity of detected obstacles. The program use state machine technique because it simplifies complex control flows by breaking them into manageable states, making it easier to maintain and debug. The program controls the robot's movement through a series of Enum input states—**STOP**, **START**, **STEP 1**, **STEP 2**, and **STEP 3**—each dictating specific motor behaviors.

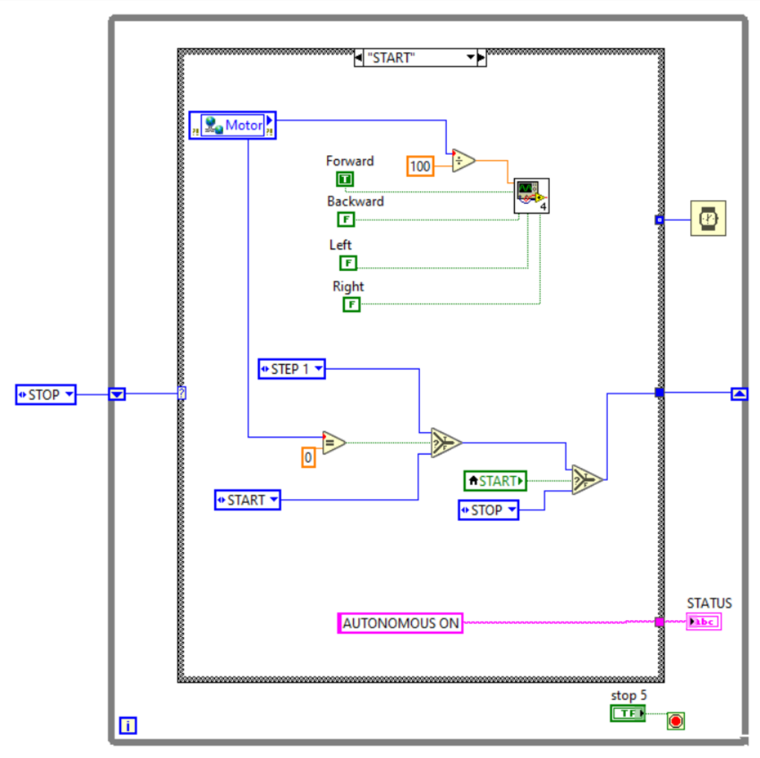
5. Autonomous Navigation Implementation

STOP STATE



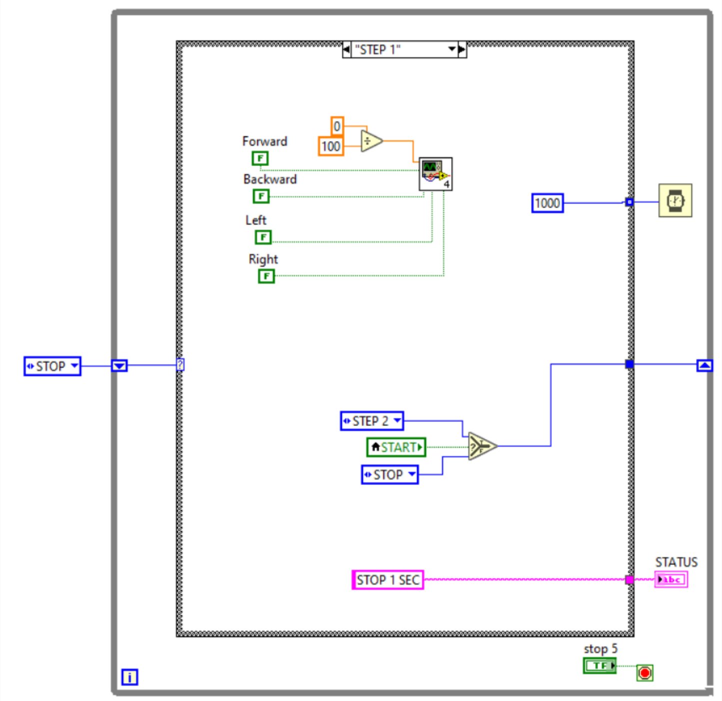
5. Autonomous Navigation Implementation

START STATE



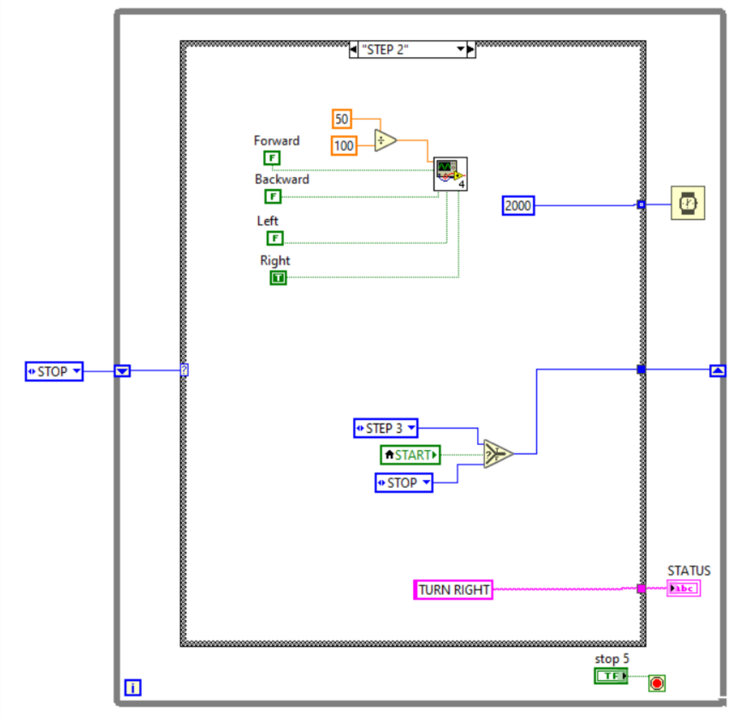
5. Autonomous Navigation Implementation

STEP 1 STATE



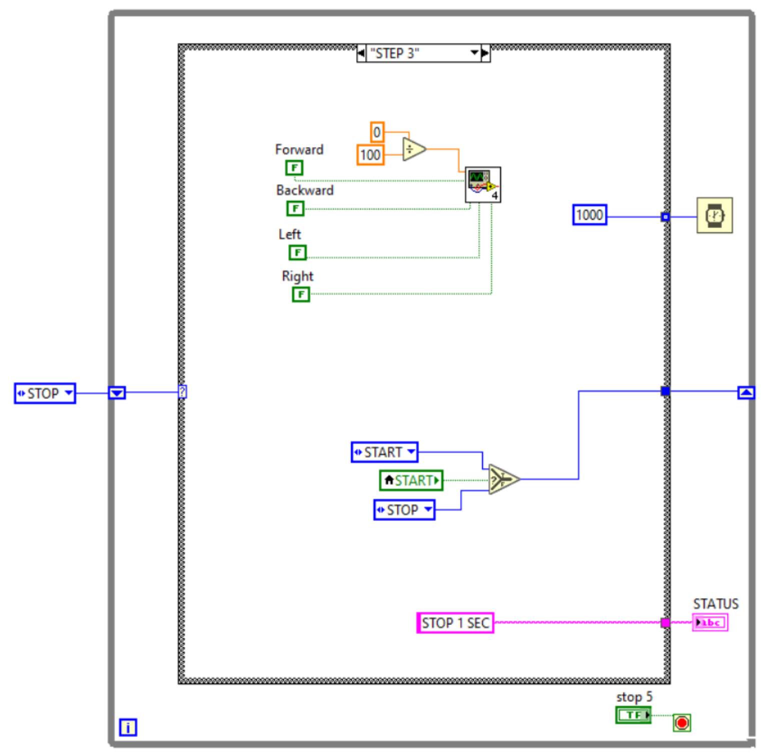
5. Autonomous Navigation Implementation

STEP 2 STATE



5. Autonomous Navigation Implementation

STEP 3 STATE



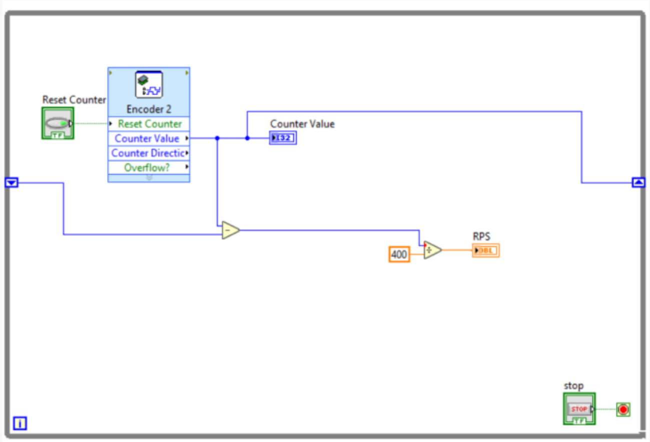
5. Autonomous Navigation Implementation

Encoder DC Motor Program

- Program Encoder DC Motor for RPS

$$RPS = \frac{(Current\ Count\ Value - Previous\ Count\ Value)}{Counts\ per\ Revolution}$$

So, when the dc motor is rotated 360 degrees, the counts per revolution that we find in 1 rotation is as many as 400, then applying the formula into the program below:



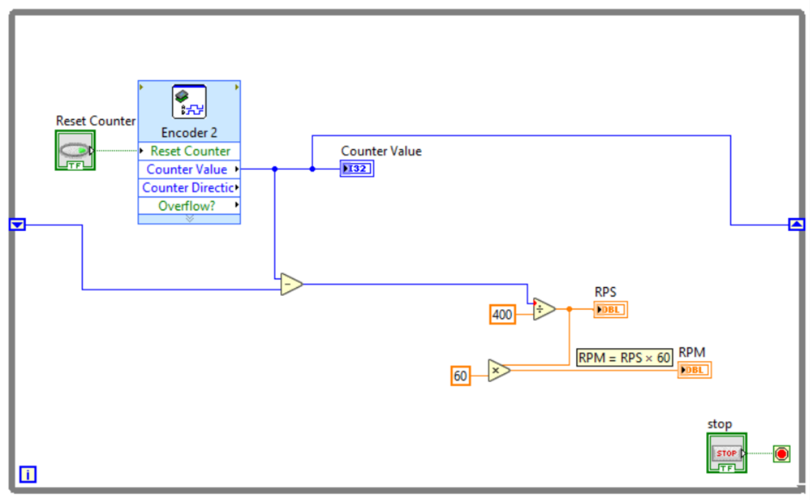
5. Autonomous Navigation Implementation

Encoder DC Motor Program

- Program Encoder DC Motor for RPM

$$RPM = RPS \times 60$$

The program takes the previously calculated **RPS** value as input

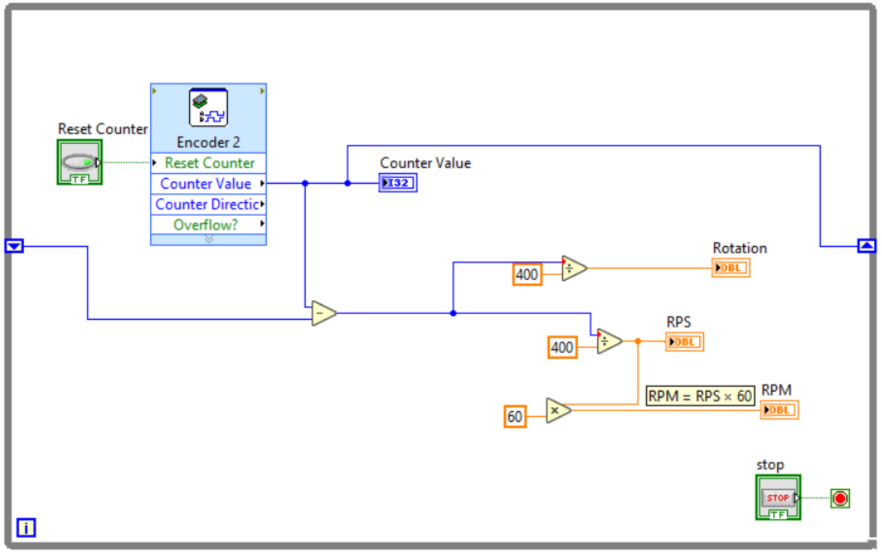


5. Autonomous Navigation Implementation

Encoder DC Motor Program

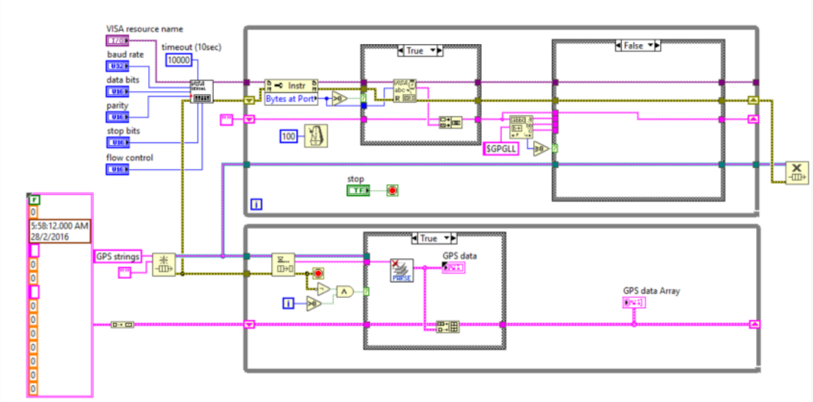
- Program Encoder DC Motor for Rotation

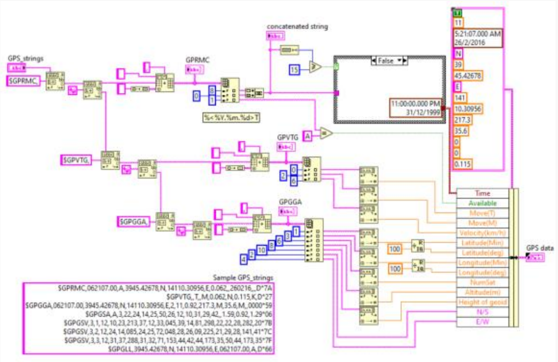
$$Rotations = \frac{Total\ Encoder\ Counts}{Counts\ per\ Revolution}$$

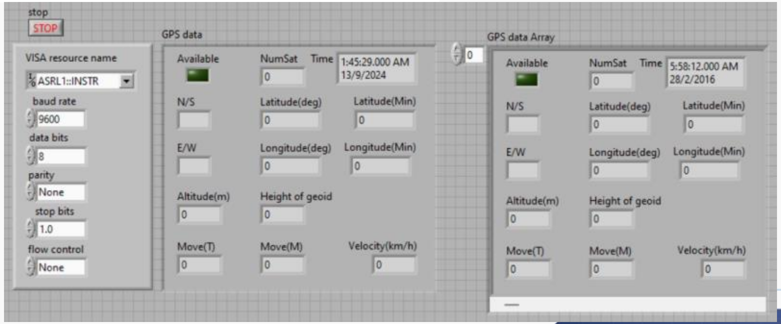


5. Autonomous Navigation Implementation

Tracking by GPS Data using GPS Module NEO 6-M







5. Autonomous Navigation Implementation

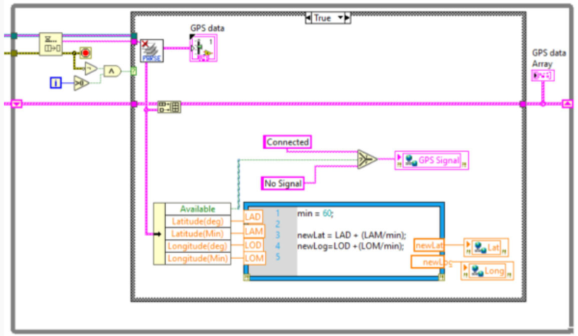
GPS Coordinate Conversion for Mapping Compatibility

In some GPS systems, coordinates are given in Degrees and Minutes (DM) format. For example:

- **Latitude:** 3° 5.3269'
- **Longitude:** 101° 32.0153'

However, most mapping tools (like Google Maps) need coordinates in Decimal Degrees (DD) format. To convert from DM to DD, use this formula and apply it on program

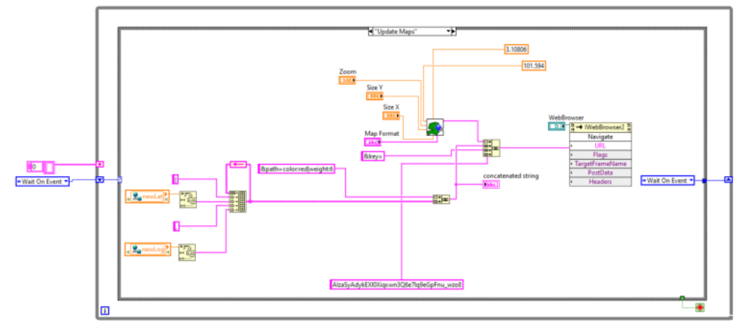
$$Decimal\ Degrees = Degrees + \frac{Minutes}{60}$$

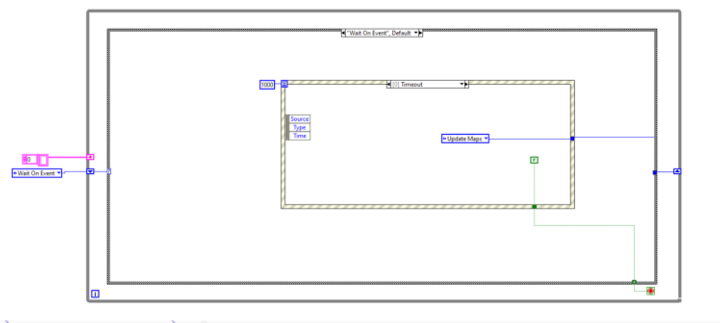


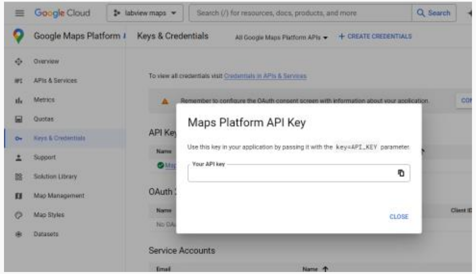
This converts the minutes into a decimal value that is added to the degrees for accurate mapping.

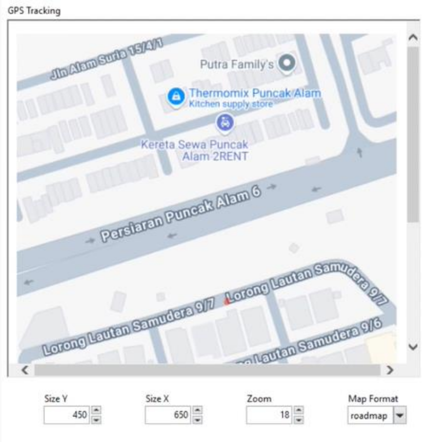
5. Autonomous Navigation Implementation

Visualizing Google Maps in LabVIEW for Position Tracking





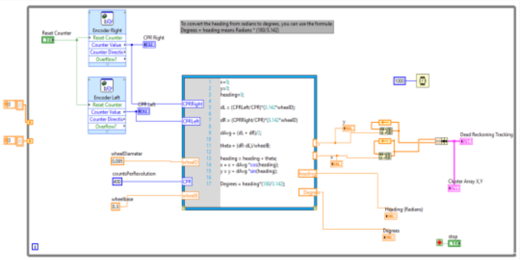




5. Autonomous Navigation Implementation

Use Dead Reckoning for estimated the position

Dead reckoning uses wheel encoder data to estimate the robot's position based on how far each wheel has traveled and the turns it has made. The algorithm estimates the robot’s position and direction using wheel encoder data. Here's how it works:



1. Distance Calculation (for each wheel):

$$dL = \left(\frac{CPRLeft}{CPR}\right) \times \pi \times wheelDiameter \qquad dL = \left(\frac{CPRRight}{CPR}\right) \times \pi \times wheelDiameter$$

2. Average Distance Traveled:

$$dAvg = \frac{dL + dR}{2}$$

3. Heading Calculation:

The change in heading ($\theta\theta$) is calculated by comparing the distances of both wheels:

$$\theta = \frac{dR - dL}{wheelBase}$$

4. Position Update:

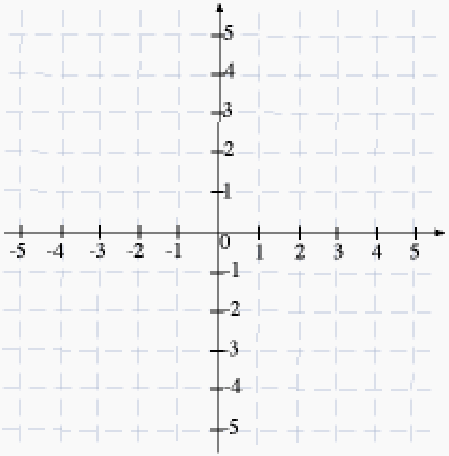
The robot’s position (x, y) is updated using the average distance and current Heading

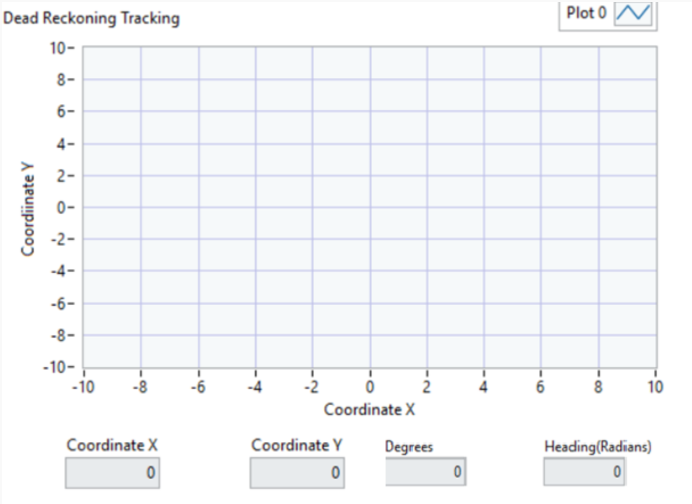
$$x = x + dAvg \times \cos(heading) \qquad y = y + dAvg \times \sin(heading)$$

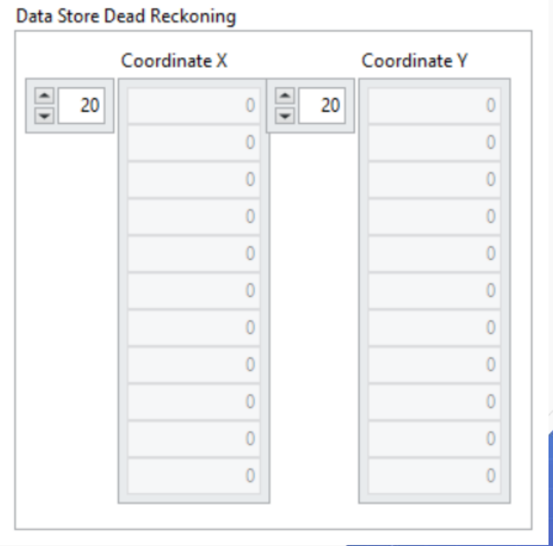
5. Autonomous Navigation Implementation

Use Dead Reckoning for estimated the position

Then, the position of the robot in a 2D Cartesian coordinate system is updated based on the average distance traveled and the current heading of the robot. These equations update the x and y coordinates by projecting the distance traveled along the current heading direction.

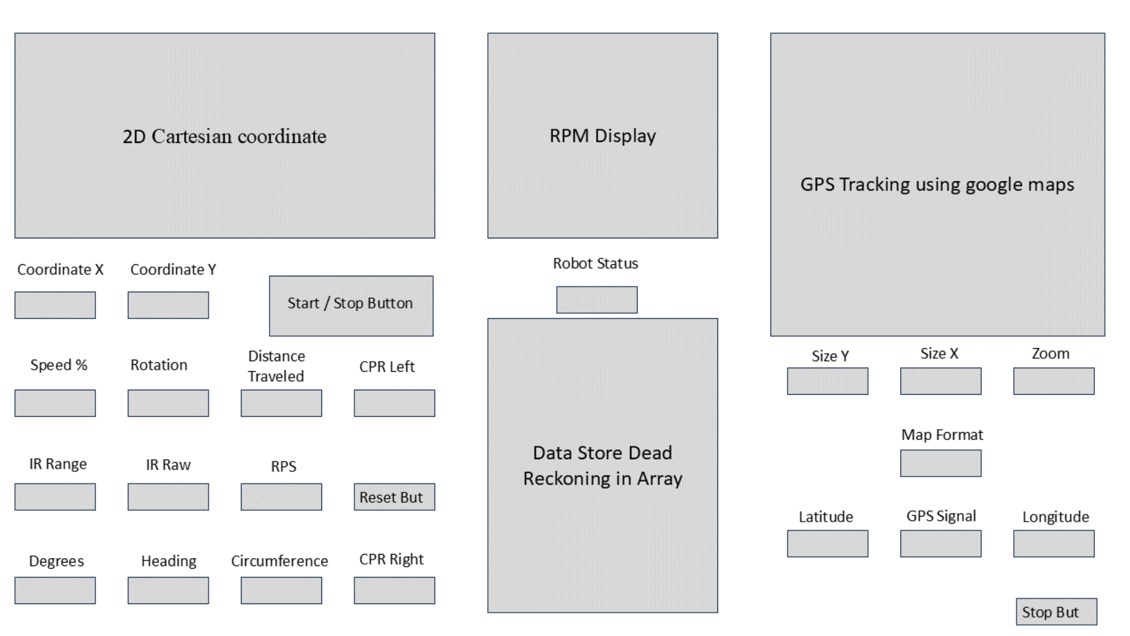






5. Autonomous Navigation Implementation

HMI Wireframe Design for Front Panel Data Representation



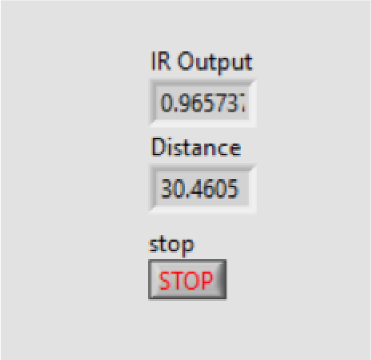
07

Result

From Fundamentals to Autonomy:
Developing LabVIEW with NI DANI Robotics Kit 2.0 using myRIO

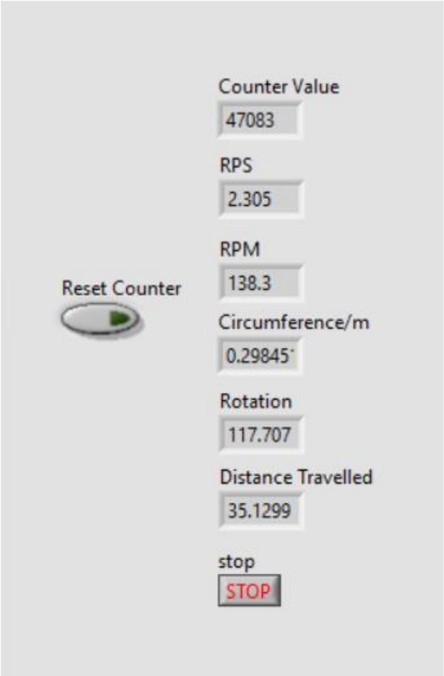
7. Result

Result of Distance Measurement Accuracy using IR Range Sensor



7. Result

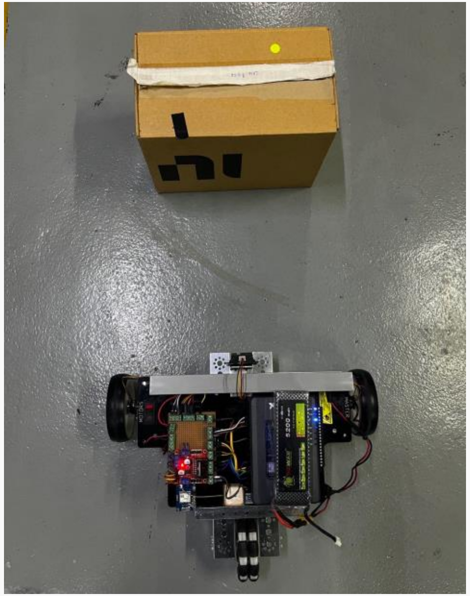
Result Encoder Output for DC Motor

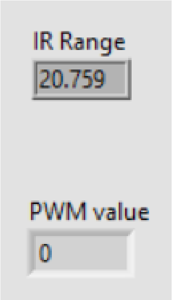


7. Result

Result Output using Fuzzy Logic Techniques

The IR sensor detected a distance of 20.75 cm

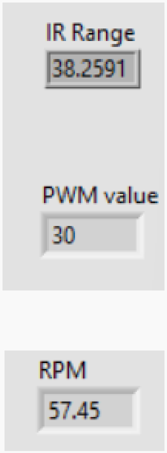




7. Result

Result Output using Fuzzy Logic Techniques

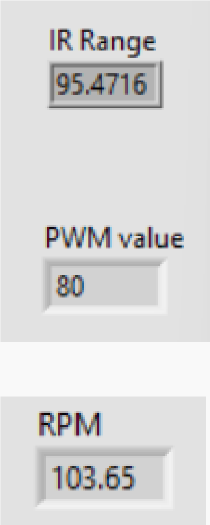
The IR sensor detected the object's distance at 38.25 cm



7. Result

Result Output using Fuzzy Logic Techniques

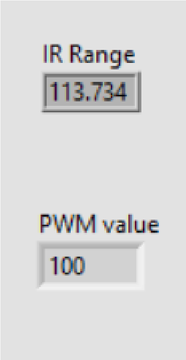
The IR sensor detected the object's distance at 87.41 cm



7. Result

Result Output using Fuzzy Logic Techniques

The IR sensor detected the object's distance at 113.734 cm



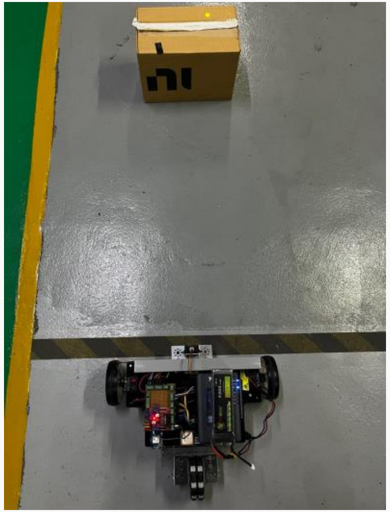
7. Result

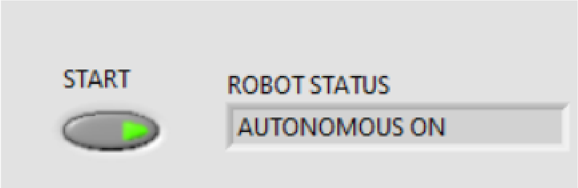
Result Output using Fuzzy Logic Techniques

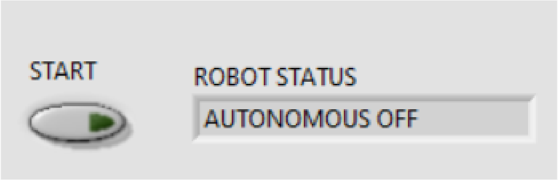
Range Distance (cm)	**Speed in Duty Cycle PWM %**	**RPM**
5-30	0	0
31-70	30	57.45
71-100	**80**	103.65
101 – above	**100**	135.45

7. Result

Result Obstacles Avoidance and Robot Status Monitoring

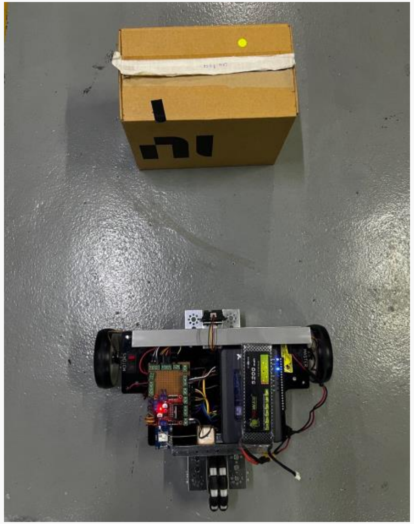


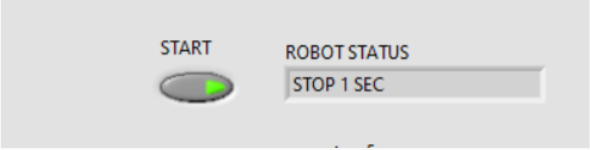




7. Result

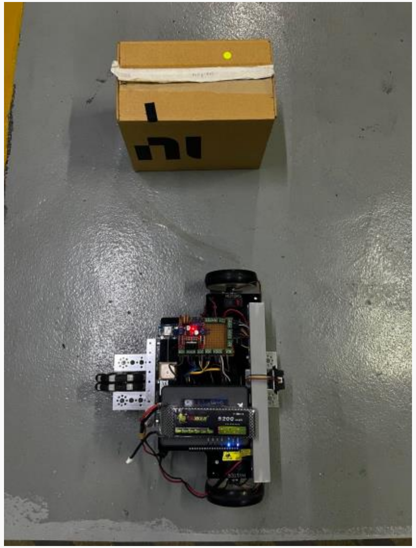
Result Obstacles Avoidance and Robot Status Monitoring

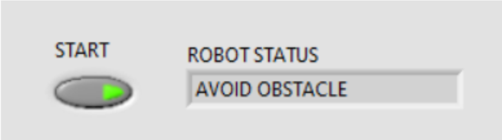




7. Result

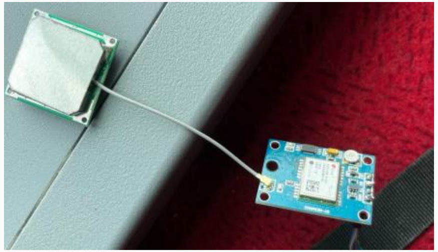
Result Obstacles Avoidance and Robot Status Monitoring

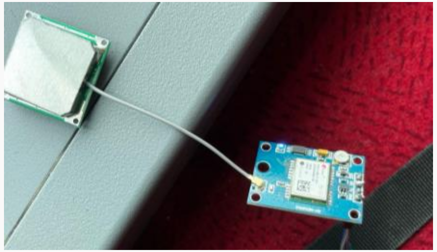


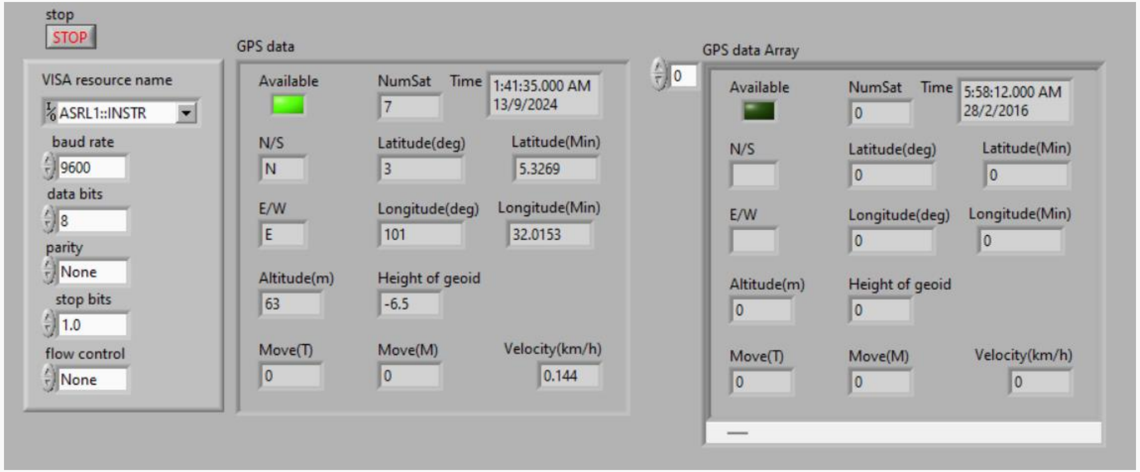


7. Result

Result Data from GPS Module NEO 6M



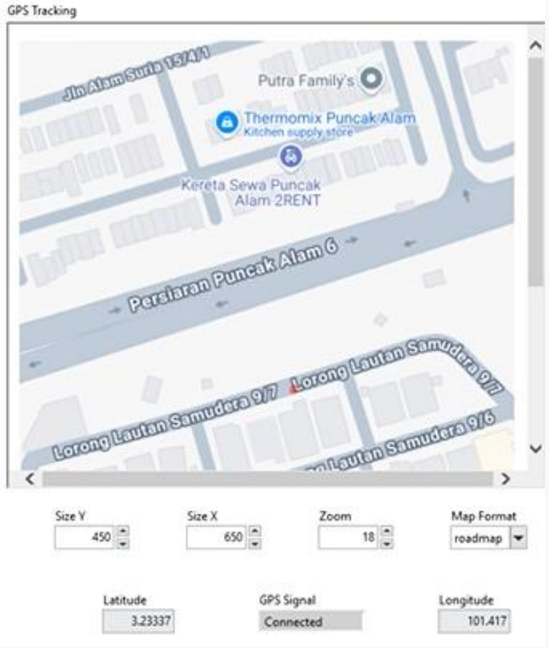


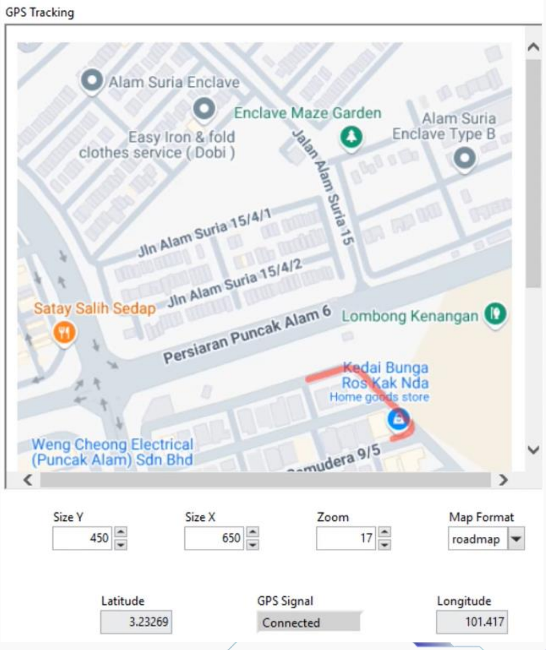


7. Result

Result tracking GPS data using Google Maps

The tracking began with an initial location at latitude 3.23337 and longitude 101.417, which was continuously updated until the final recorded location at latitude 3.23269 and longitude 101.417

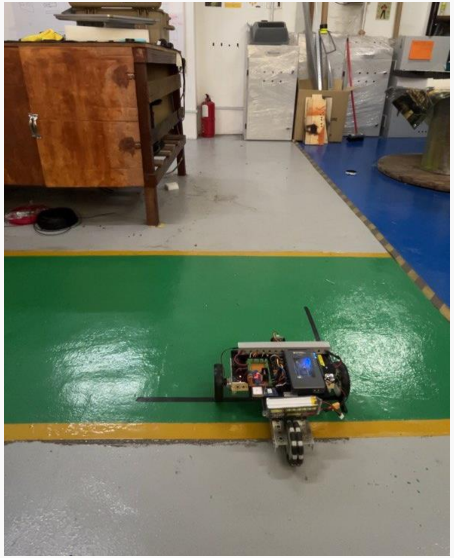




7. Result

Result Data by Dead Reckoning using 2D Cartesian coordinate when in unavailable signal

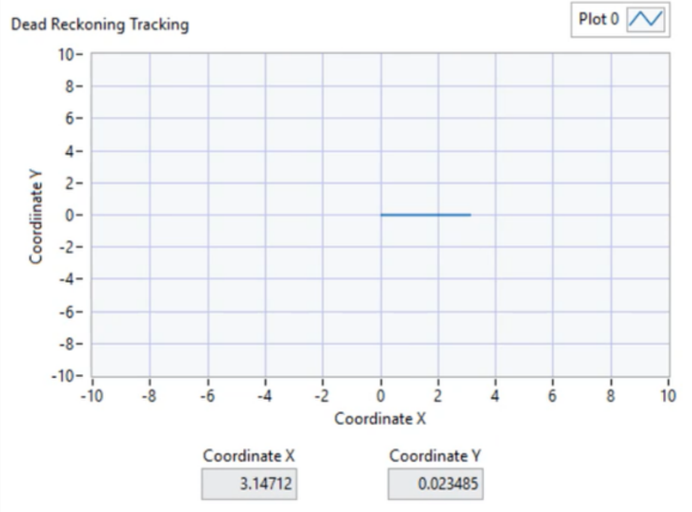
At the start of the testing, the robot was positioned at Coordinate X = 0 and Coordinate Y = 0. This initial point represents the origin of the 2D Cartesian plane, where the robot begins its movement.

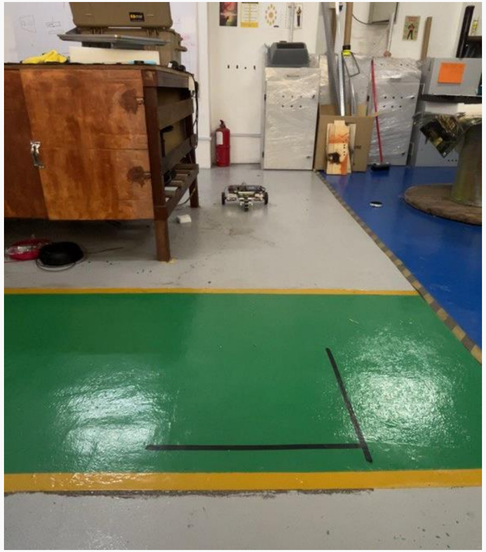


7. Result

Result Data by Dead Reckoning using 2D Cartesian coordinate when in unavailable signal

By the end of the testing, the final coordinates were recorded as approximately Coordinate X = 3.1 and Coordinate Y = 0.0, . This indicates that the robot moved 3.1 units along the X-axis, maintaining an almost perfect straight line with minimal deviation in the Y-axis (close to zero)

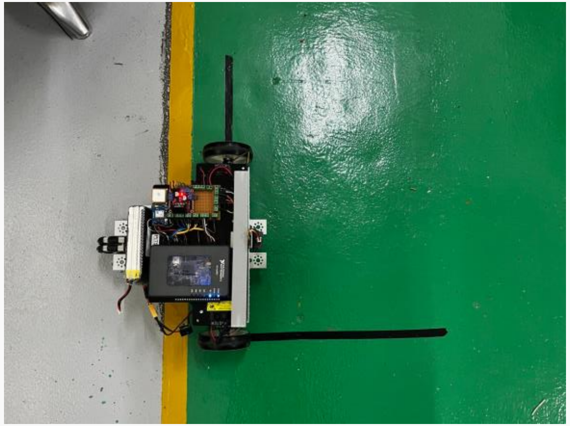


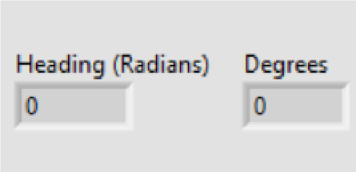


7. Result

Result Data by Dead Reckoning for Heading and Degrees

At the start of the testing, the robot was positioned with an initial heading of 0 radians and 0 degrees. This indicates that the robot was aligned with the reference direction, typically the positive X-axis, before initiating any movement or turns.

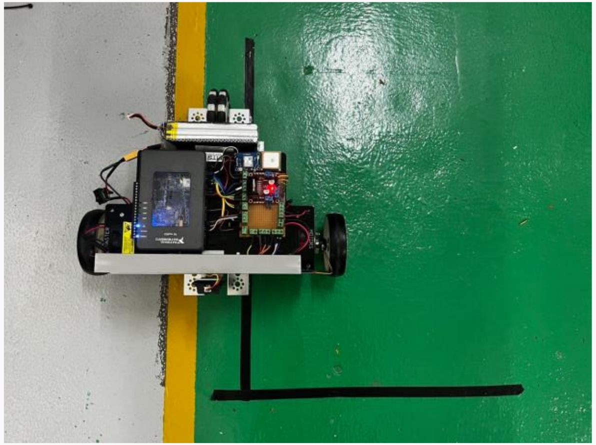


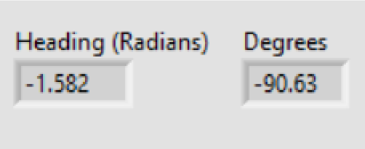


5. Result

Result Data by Dead Reckoning for Heading and Degrees

During the movement, the robot performed a right turn. The heading was recorded as -1.582 radians, which corresponds to approximately -90.63 degrees. The negative value indicates a clockwise rotation, confirming that the robot made a right turn from its initial orientation by approximately 90 degrees

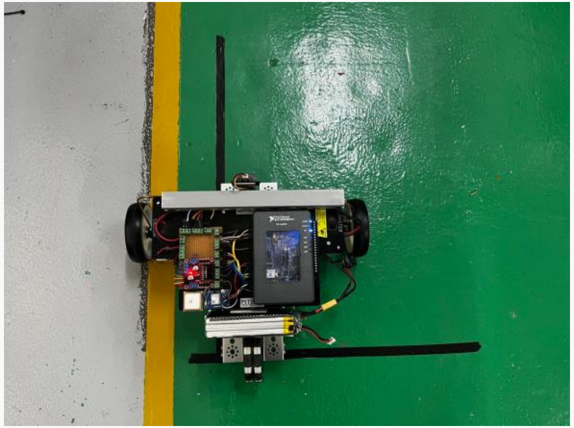


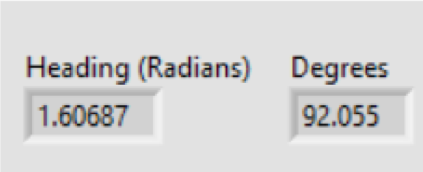


5. Result

Result Data by Dead Reckoning for Heading and Degrees

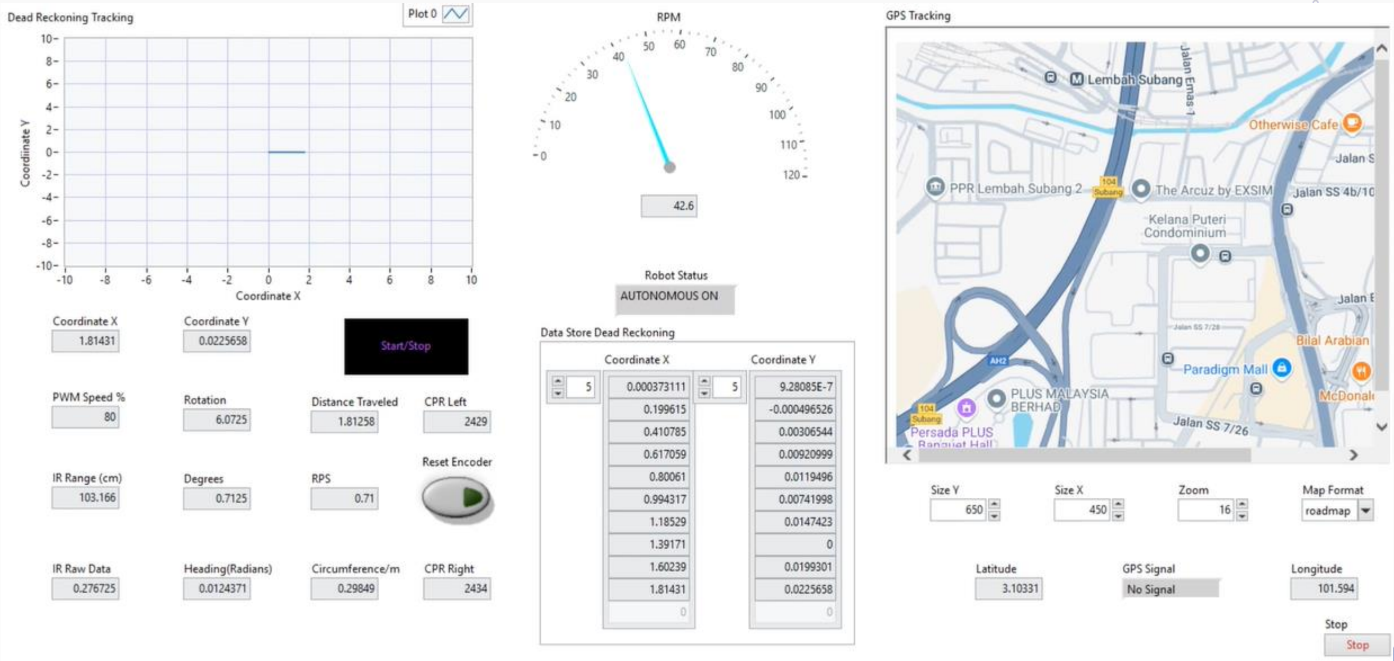
After completing the right turn, the robot subsequently performed a left turn. The heading for this turn was recorded as 1.60687 radians, equivalent to approximately 92.055 degrees. The positive value denotes a counterclockwise rotation, indicating that the robot turned to the left by approximately 92 degrees from its initial orientation.





5. Result

Result HMI using NXG Style using LabVIEW



07

Conclusion

From Fundamentals to Autonomy:

Developing LabVIEW with NI DANI Robotics Kit 2.0 using myRIO

CONCLUSION

This project successfully integrated multiple hardware components and utilized the myRIO microcontroller with LabVIEW programming to develop an autonomous mobile robot. Key achievements include:

- **Dead Reckoning**: Estimated position in a 2D Cartesian coordinate system (X, Y) to track location without GPS.
- **Fuzzy Logic**: Controlled motor speed and enabled effective obstacle avoidance for intelligent navigation.

The results demonstrate the effectiveness of NI tools and LabVIEW in autonomous robotics, paving the way for future advancements in challenging environments.

Thanks

Any Question ?

DEMONSTRATION VIDEO