GUIDANCE NAVIGATION AND CONTROL IMPLEMENTATION FOR UNMANNED GROUND VEHICLE USING NI-myRIO

by

ISHWARYA SRINIVASAN

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTERS IN AEROSPACE ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON
August 2016

Dedicated to my mom, dad, Badri, my family, admins and my friends who have encouraged me to do better, and supported me throughout my life.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervising professor Dr.Atilla Dogan for constantly guiding me throughout my Masters at University of Texas at Arlington. During this period I have learnt a lot about unmanned vehicle systems and modeling a system. I am grateful to Dr.Dogan for providing me an opportunity to learn under him. I would like to convey my regards to Dr.Alan Bowling, Dr.Donald Wilson and Dr. Brian Huff for taking time out to be a part of my thesis committee. I am thankful to Dr. Huff for providing me with the required hardware for carrying out the real time experiments. I am grateful to Christopher Abrego for helping me out with LabVIEW. I am highly obliged to National Instruments application manager for helping me through troubleshooting blocks used in my GNC algorithm. I would also like to thank Sampath Reddy for his insights and support. Lastly, I am indebted to my family, teachers and friends for their support and inspiration. I would also like to express sincere gratitude to my ICC family and friends who, directly or indirectly, have lent their helping hand in this venture.

August 12, 2016

ABSTRACT

GUIDANCE NAVIGATION AND CONTROL IMPLEMENTATION FOR UNMANNED GROUND VEHICLE USING NI-myRIO

Ishwarya Srinivasan, M.S.

The University of Texas at Arlington, 2016

Supervising Professors: Dr. Atilla Dogan

This research effort aims at investigating alternative real-time implementation software and hardware for Guidance, Navigation and Control (GNC) algorithms for an Unmanned Ground Vehicle (UGV). A GNC algorithm was previously developed for a skid-steered tracked UGV to go through assigned waypoints based on encoder counts. The UGV has two electric motors driving the tracks on each side and two encoders providing the speeds of the drive wheels. This algorithm was implemented using Matlab/Simulink-based model running on a mini computer, interfacing with the electric motors and encoders through a speciality control board. This current effort is to implement the same GNC algorithm for the same UGV, but using LabVIEW, a graphical programming environment by NI (National Instrument Corporation) for programming the GNC algorithm and NI-myRIO, an embedded hardware device, for running the LabVIEW-based GNC algorithm and interfacing with the electric motors and encoders. A kinematic model of the UGV is also developed in LabVIEW and a closed loop simulation with the GNC-algorithm is carried out. The LabVIEW-based simulation results are compared with the Matlab/Simulink-based simulation to verify

the accuracy of the GNC implementation in LabVIEW. Then, the NI-myRIO running the GNC-algorithm is used to carry out experiments of the UGV going through specified waypoints. Based on this overall project, the LabVIEW and NI-myRIO solutions is found to be user-friendly and very effective and reliable for the purpose of realtime implementation of GNC algorithms.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

This chapter explains the reasons for carrying out the following research. It also gives a brief of all the chapters covered.

1.1 Research Motivation

Unmanned vehicle systems (UVS) include aircrafts, ground vehicles like cars, tanks, underwater and water surface vehicles [2]. For the autonomous operation of UVS, they should be equipped with a GNC(Guidance, Navigation and Control) algorithm. The GNC is designed to control the UVS motion, collect data and make it perform a given task. The platform used for designing the GNC algorithm should be user-friendly and have faster real-time data collection and interpretation so that the UVS is faster and quicker in generating response. The main motivation of this research effort is to investigate and experiments alternative programming and hardware development environments for the realtime implementation of guidance, navigation and control algorithms for unmanned vehicle systems in academic setting. The specific focus of this research is on the use of LabVIEW to create the GNC algorithm and use a myRIO board to run the LabVIEW flow diagram real time on an Unmanned Ground Vehicle (UGV).

1.2 Application of Unmanned Vehicle Systems

The unmanned vehicle systems and their application range has been expanding over the years [3]. The development of unmanned ground vehicle dates back to

the 1930s where a tank armed with a machine gun was controlled by radio from another tank, and the most recent being the Google car [2]. It has a wide range of applications areas which include agricultural monitoring, military purpose, law enforcement, carry out naval operations, telecommunication, wildfire mapping and disaster management [4]. In military sector, UGV is used to carry out surveillance, detect unexploded bombs, and to deliver supplies and ammunition to troops [5]. In agriculture activities, UGV places an important role in crop dusting and management of large agricultural fields [6]. Use of bomb disposable robot by the police for law enforcement [7] and using path planning techniques for cleaning the house are few examples of UGV. The development of a driver-less car, Google car, and a drone to deliver package, Amazon drones, has caught public's attention towards unmanned systems and has encouraged further development of such systems and widening their applications.

1.3 Thesis Objective

There are many different platforms on which the GNC algorithm to drive a UVS can be implemented. The objective of this research is to evaluate a NI (National Instruments) based tools, LabVIEW as software to program the GNC algorithm and myRIO for real time implementation of the algorithm. At UTA's AVL (Autonomous Vehicles Lab), MATLAB/Simulink programming language along with a micro PC-based computer called "NUC" and phidgets have been tried, tested and used to have a realtime implementation on a UGV to perform various tasks like waypoint navigation, obstacle avoidance and path planning using image processing. NI myRIO is an embedded hardware device used for real-time implementation performing task like detecting obstacle distance using sonar, driving a motor, measuring temperature

and many more data acquisition applications [8]. Specific steps taken to achieve the research objective are listed below:

1. Familiarize with the LabVIEW graphical programming environment to be able to program GNC algorithms based on prior Matlab/Simulink-based implementation.
2. Familiarize with myRIO and mechatronics kit to better understand the sensor-actuator interfacing.
3. Develop LabVIEW implementation of GNC algorithm based on prior Simulink-implementation.
4. Develop LabVIEW model of the kinematics of the UGV and run closed loop LabVIEW-based simulations
5. Carry out hardware integration of myRIO with the electric motors and encoders of the UGV
6. Control the UGV with myRIO for waypoint navigation

1.4 Thesis Organization

Chapter 2 provides a literature review of the work done on choosing LabVIEW for implementing the GNC algorithm. This chapter also deals with the previous platform used for GNC algorithm implementation. Chapter 3 presents a basic explanation of what LabVIEW is and how it is used along with a brief about myRIO. It also has details of the experiments performed to become familiar with the software tool and hardware platform, especially for interfacing with sensors and and controlling actuators. Chapter 4 presents a description of the UGV used for real time application. The vehicle's dimensions and parameter calculations are also discussed in this section. Chapter 5 discusses the requirements and design details of the GNC algorithm and its implementation in a MATLAB/Simulink environment, along with the mathematical

model of the UGV motion platform. Chapter 6 adds on to the LabVIEW part of the GNC implementation and the results of the closed loop simulation process. After this is a chapter discussing the details of real-time implementation and results of the experiment. Finally, chapter 8 concludes the thesis and gives a brief about the future work.

1.5 Contribution

The contribution of this work to the ongoing research at University of Texas, Arlington in the area of guidance, navigation and control is to use LabVIEW along with myRIO to control a UGV system. An existing GNC simulink model is used as a reference to create the LabVIEW flow diagram. This led to the introduction of a new embedded system which allows control of motors and reading of encoders through mere pin connections.

CHAPTER 2

Introduction to LabVIEW and myRIO

In this chapter, a brief description of LabVIEW and myRIO is given along with description of the various experiments performed using the device to become familiarized, especially, with running LabVIEW VI, and interfacing with sensors and actuators/servos.

2.1 Brief Introduction to LabVIEW

2.1.1 Description

Laboratory Virtual Instrument Engineering Workbench commonly known as LabVIEW, is a design platform/environment as a visual programming language from National Instruments [8]. It is a graphical design platform wherein users can create a flow diagram to perform any type of mathematical, control system, measurement and data acquisition operation. It is compatible with various embedded system devices like CompactRIO, myRIO, Robotics module for real-time hardware implementation. LabVIEW has many in-built modules which has blocks for design, analysis and visualization of data. The GNC algorithm can be implemented in LabVIEW with various user-friendly blocks that can be used for programming any mathematical and logical expression. Some uses of LabVIEW are listed as [8]:

1. Instrument Control
2. Automation Industry
3. Data Acquisition
4. Embedded Control systems

2.1.2 LabVIEW Programming Environment

LabVIEW's graphical interpretation of any model is called a virtual instrument or VI in short. Each VI has a front panel and a block diagram. The front panel is the user interface VI which has the input and output control, indicators and graphs [8]. Figure 2.1 shows a front panel VI with an LED indicator and a STOP button, these are user friendly indicators and controllers used in development of the algorithm.

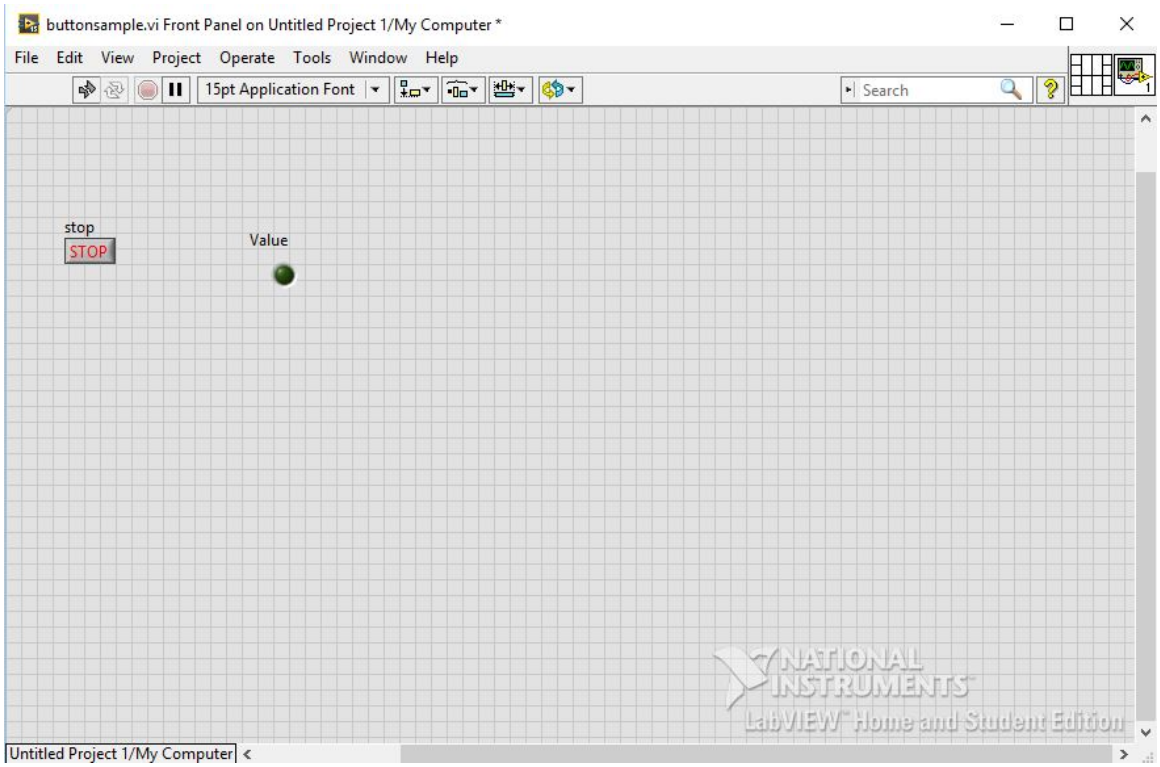

Figure 2.1: Example of Front Panel

Figure 2.1 shows a control palette, which contains the controls and indicators used to create the front panel. The control palette can be accessed from the front panel window by selecting "View" and then "Controls Palette" from the top menu of the front panel window or by right clicking on the empty space in the front panel [9].

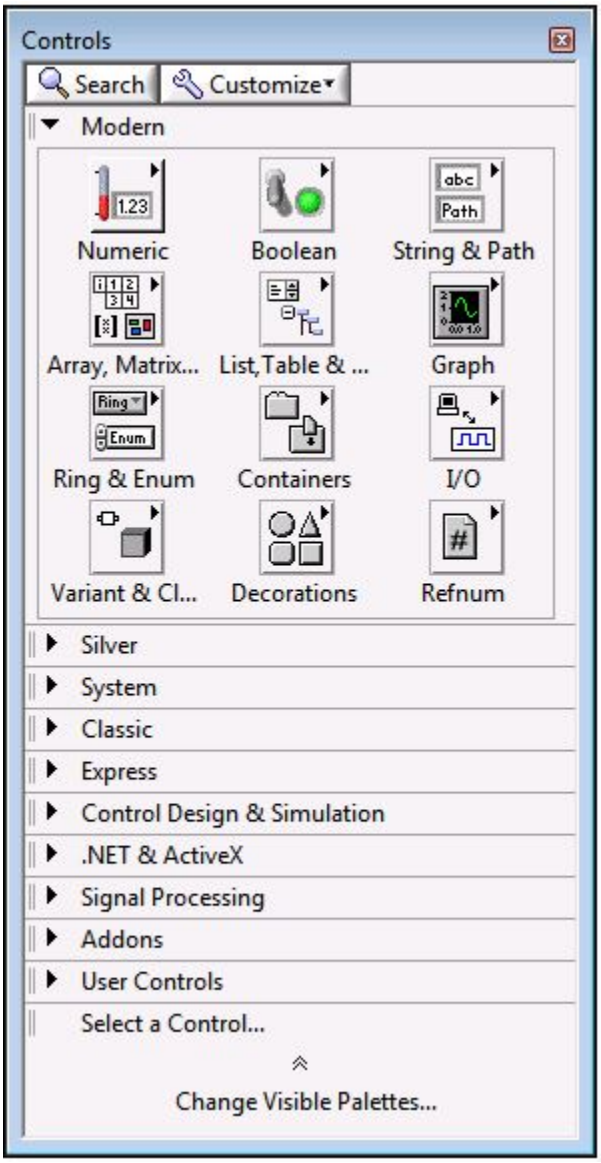

Figure 2.2: Control Palette

A block diagram panel (an example is shown in Fig. 2.3) contains the functions and graphical code [9]. The wiring and actual modeling of a program is done in the block diagram panel. [8]

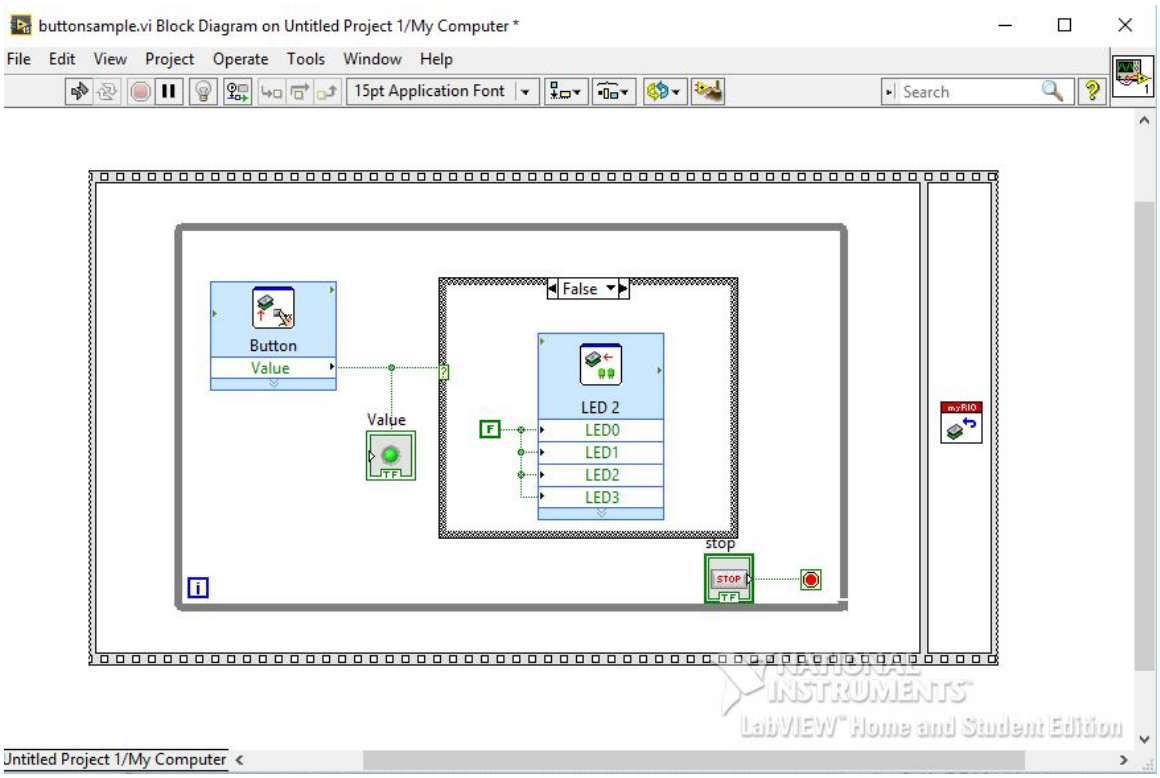

Figure 2.3: Example of Block diagram

Functions palette (see Fig. 2.4)contains various structural, mathematical, comparison, timing and other blocks, functions and constants which can be used to create a block diagram and is accessed by selecting "View" and then "Functions Palette" from the top menu of the block diagram window or by right clicking on empty space in the block diagram.

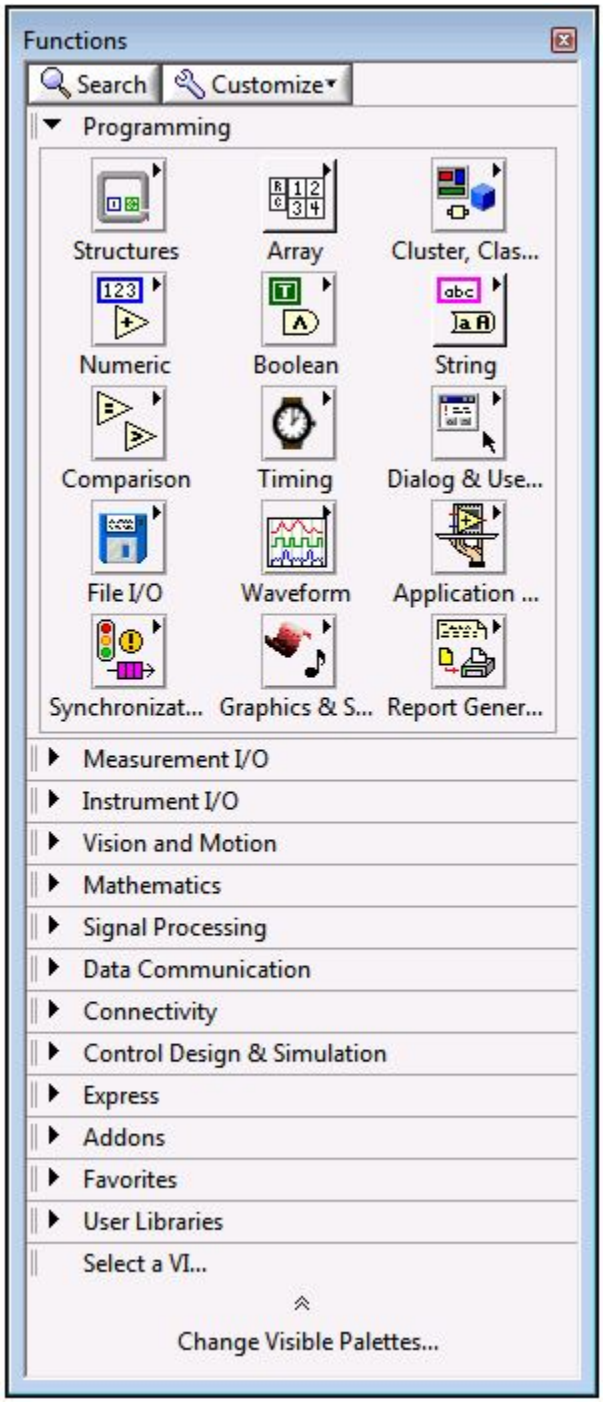

Figure 2.4: Functions Palette

2.2 Brief Introduction to myRIO

2.2.1 Description

NI myRIO is a portable embedded hardware device that uses LabVIEW to allow the real-time implementation of guidance, navigation and control algorithms, and to enable the interfacing the algorithms with actual sensors and actuators. myRIO board has onboard accelerometer, LED, push button and WiFi for wireless connection.

2.2.2 I/O Interface

Its features include [10]

1. 10 analog inputs, 6 analog outputs, 40 digital I/O lines- These ports are divided in 3 channels and used for connecting the hardware to be able to read data into the LabVIEW program.
2. Power Input Cable
3. USB Device Cable
4. Audio In/Out Cables
5. myRIO Expansion Port (MXP) Breakouts

Figure 2.5: myRIO

2.2.3 Communication with Laptop

Connect the myRIO to a laptop through USB connection for first time interface, this will open a window as shown in Fig. 2.6. A LabVIEW Getting Started Window would open up as soon as myRIO is connected through USB (see Fig. 2.7). Following the steps mentioned in the manual for interfacing, a successful connection is established with myRIO. After configuration, select "Start my first project now". Following the configuration myRIO, I/O-specific Express VI can be used to program with a hardware in the loop. myRIO can be connected to any laptop through a USB cable or through WiFi for having access when conducting experiments with a moving vehicle [11]. After myRIO is connected with LabVIEW over USB or WiFi the GNC

flow diagram can be created in a new myRIO project. Steps to connect myRIO with computer:

1. Plug the myRIO, it will switch ON the myRIO
2. Connect the myRIO to a computer through USB connection.

   Once it us connected over USB, a window will pop up

   Select Launch myRIO configuration.

   "Getting Started with myRIO" window will show up

   Click on Next and check the accelerometer and LED on board.

   Click on Next and the window would ask to Launch LabVIEW.
3. For connecting myRIO over WiFi, the WiFi LED on the myRIO should be ON (orange light indicates ON)

   On computer WiFi networks connect to $myRIO\_wifi$.

   Then manually start LabVIEW with myRIO.

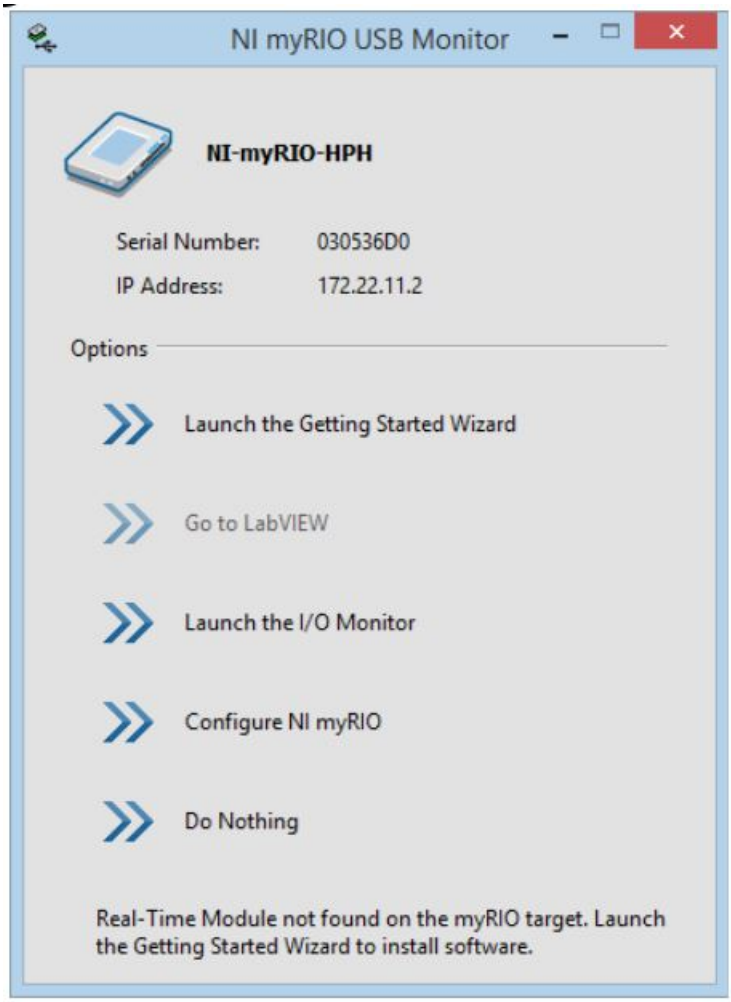

Figure 2.6: myRIO USB monitor

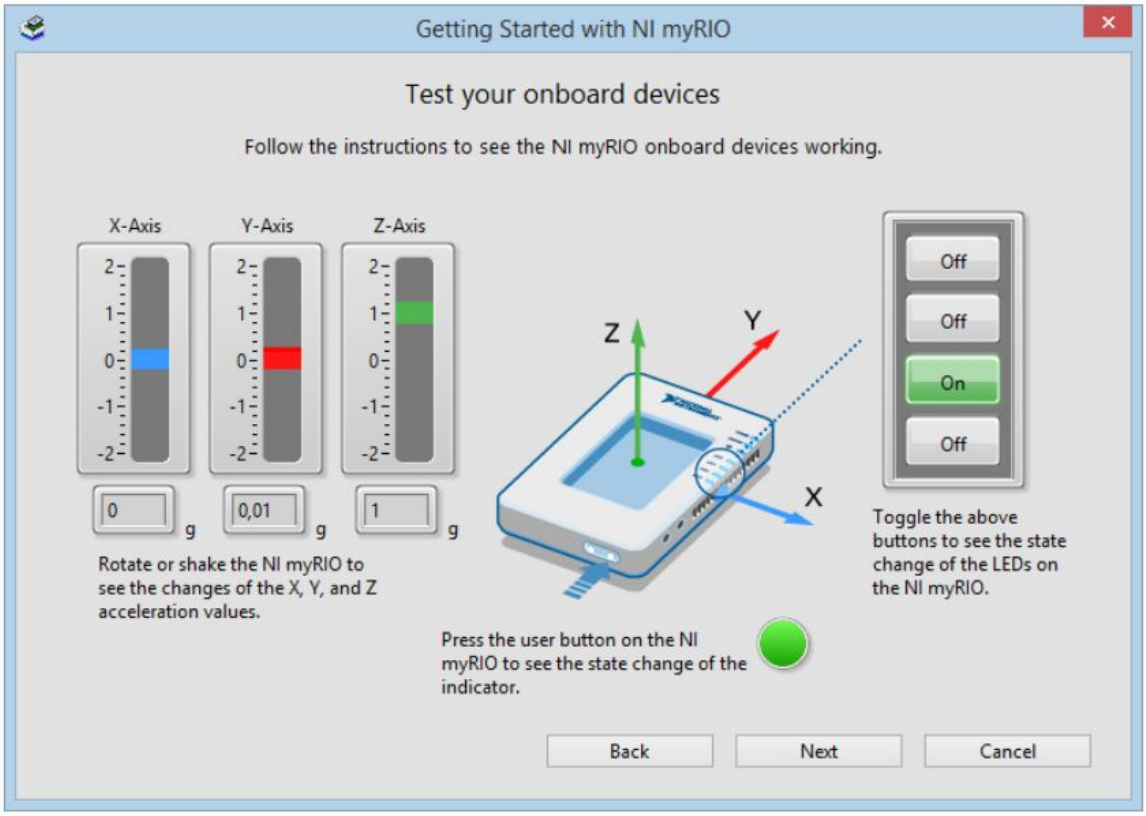

Figure 2.7: Getting Started myRIO

Steps to start a New Project in LabVIEW:

1. Open LabVIEW
2. Select "Create a Project" to design a new project in LabVIEW. This project can be a LabVIEW project or a myRIO project. If the objective is to use a hardware and work on that, then select myRIO project.

Shortcut keys to switch between Front panel and Block diagram window is CTRL+E and to run the program is CTRL+R.

2.3 Specialized LabVIEW Modules

LabVIEW 2015 myRIO Software Bundle Web-Based Installer is the software used to perform experiments [8]. It is programmable with LabVIEW or C. For designing the GNC algorithm in LabVIEW, the following LabVIEW modules are used [10]:

1. LabVIEW Real-Time Module
2. LabVIEW myRIO Toolkit
3. LabVIEW Control Design and Simulation Module
4. LabVIEW MathScript RT Module

These are the additional modules used along with LabVIEW 2015 full Development version to be able to design the GNC algorithm. In order to carry out the simulation of the GNC algorithm with the tank block a Control Design and Simulation Module (CDSM) is used. Within this module is a Simulation loop within which the GNC algorithm is created. With this module, the simulation time, step time and the solver type used are specified within the loop parameter. The myRIO toolkit provides modules for hardware-software connections. For example, an encoder express vi from myRIO Toolkit module is used for reading the rotation from the myRIO and display the results into the front panel of LabVIEW.

2.4 Integration with various sensors and actuators

For driving the UGV with myRIO, communication with sensors and actuators should be established, specifically with electric motors and encoders in this project. As the first step, myRIO connection is established with various components available from NI as the mechatronics kit. This is done to familiarize with the procedure on the LabVIEW side and the hardware side to read from sensors and control actuators [10]. The following describes these attempts.

2.4.1 Servo-motor

Figure 2.8: Servo Motor with myRIO

The GWS S03N STD servo is a part of the NI-myRIO mechatronics kit, comes with different types of rotary shafts [10]. A servo motor (Fig.2.8) is a rotary actuator which allows control of angular position, velocity and acceleration of a rotating shaft [12]. It includes a DC motor, gearbox potentiometer and a controller. The servo motor requires 5V power supply and a single PWM signal which is connected to the myRIO. It has been used in remote controlled aircraft, cars and boats to manipulate the control surfaces and steering [10]. Application areas of a servo-motor include robot arm, sensor, scanner and actuator control. The servo-motor communicates with the myRIO through the pin connections specified in Table 2.1:

Table 2.1: Servo-Motor Pin Connection to myRIO

Servo-motor	PIN Number
Vcc	B(+5V) Pin 1
Ground	B/GND Pin 6
Command Signal	B/PWM Pin 27

2.4.2 H-Bridge and Geared Motor

Pmod HB5 is provided in the mechatronics kit, which can drive the geared motor [10]. H-bridge is a compact circuit board which provides power to the motors and helps in connecting the motor to the myRIO board. Gear motor is an electric motor which has a type of gear system on the output, this gear system is the gear box [13]. This combination of gear box and motor helps in changing speed, torque and direction of power easily. Fig.2.10 shows a 12V DC geared motor used for the experiment. The pin connections on the H-Bridge is for giving Pulse Width signals to the motor and have a digital out connection to display the output on the LabVIEW front panel VI.

Figure 2.9: H-Bridge

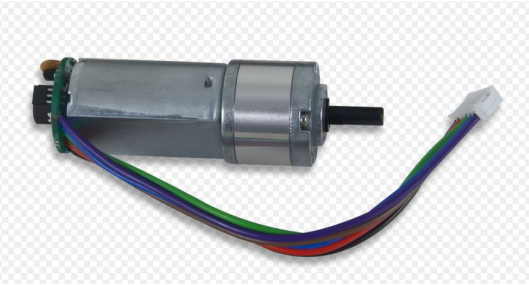

Figure 2.10: Geared Motor

The Pmod HB5 communicates with the motor and the myRIO through the pin connections specified in Table 2.2.

Table 2.2: H-Bridge and Geared Motor Pin Connection to myRIO

H-Bridge	PIN Number
J2.VM	B(+5V) PIN 1
J2.Ground	B/GND PIN 6
J1.Vcc	A/3.3V PIN 33
J1.GND	A/GND PIN 30
J1.ENABLE	A/PWM PIN 27
J1.SENSOR A	A/ENCA PIN 18
J1.SENSOR B	A/ENCB PIN 22
J1.DIR	A/DIO0 PIN 11

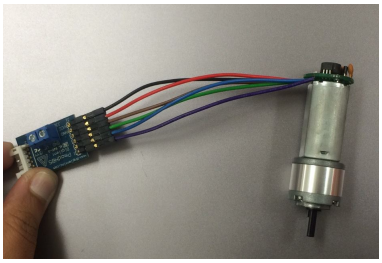

Figure 2.11: H-Bridge Connection with Geared Motor

Fig. 2.11 shows the connection of Pmod H-Bridge with the geared motor. The other end of this H bridge is for connections to myRIO. The H-bridge driven motor provides high torque and is suitable to drive a train of robotic platform [10]. The geared motor has encoders connected on the top, these encoder is also connected through the H-bridge to the myRIO to display the number of rotations calculated.

2.4.3 IR Range Finder

Figure 2.12: IR Range Finder

An Infra-red Range Finder is a sensor that uses a beam of reflected infrared light to sense the distance between the reflected surface and the sensor by the process of triangulation [10]. The sensor is connected to the myRIO through pins given in Table 2.3. The obstacle distance is proportional to reciprocal of IR range finder's

Table 2.3: Infra-red Range Finder Pin Connection to myRIO

Infra-red	PIN Number
5V Power	B/+5V PIN 1
Ground	B/GND PIN 6
Output	B/AIO PIN 3

output voltage. Range of this sensor is 10-80 cm [10].

2.4.4 Sonic Range Finder

Figure 2.13: Sonic Range Finder

Sonic Range Finder, also known as the sonar, is a small range finder. It provides accurate reading of obstacles upto 6m distance. The transducer of the sonar generates short pulses of sound that are detected by the same transducer when echoed back [10]. PIN Connection to myRIO is specified in Table 2.4.

Table 2.4: Sonar Range Finder Pin Connection to myRIO

Sonar Range Finder	PIN Number
Power Supply	A/+3.3V PIN 33
Ground	A/GND PIN 30
Output	A/UART.RX PIN 10

2.4.5 Accelerometer

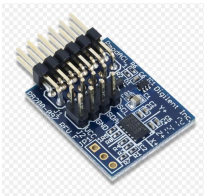

Figure 2.14: Accelerometer

Accelerometer is an instrument that measures the acceleration of a vehicle, aircraft or ship [14]. The accelerometer in the mechatronics kit has I2C-bus communication. The PIN connection of accelerometer with the myRIO kit is specified in Table 2.5.

Table 2.5: Accelerometer Pin Connection to myRIO

Accelerometer	PIN Number
3.3V supply	A/3.3V PIN 33
Ground	A/GND PIN 30
Serial Data	A/I2C.SDA PIN 34
Serial Clock	A/I2C.SCL PIN 32
Interrupt#2	A/DIO0 PIN 11
Interrupt#1	A/DIO0 PIN 13

CHAPTER 3

Unmanned Ground Vehicle and Hardware Integration

This chapter describes the vehicle used for practical implementation of GNC algorithm along with other hardware components used as shown in Fig. 3.2.

3.1 Vehicle Platform

UGV platform used in this research is a Wild thumper, all-terrain six wheeled robot chassis. The vehicle dimensions are 17 inches $\times$ 12 inches $\times$ 6.5 inches as shown in the Fig. 3.1. Platform has six motors with steel gear boxes and two encoders to count the number of rotations by the wheel. The suspension has a torsion spring design with rubber mounts for flexible joint. The chassis and the top has 4 mm holes punched every 10 mm for mounting of sensors and other hardware [15].

Figure 3.1: Unmanned Ground Vehicle

3.2 Electrical Components

All the hardware components in-built and externally used with the unmanned ground vehicle are discussed in this section.

3.2.1 Electric Diagram of the Whole System

Figure 3.2 displays the electrical diagram of the entire system. It includes two LiPo batteries to power the vehicle's motors and one LiPo battery with DC-to-DC convertor to power the myRIO. A myRIO board is used for real-time implementation of the GNC algorithm through LabVIEW. The vehicle's motors are connected to myRIO through an H-Bridge. The encoders are connected directly to the myRIO for data acquisition.

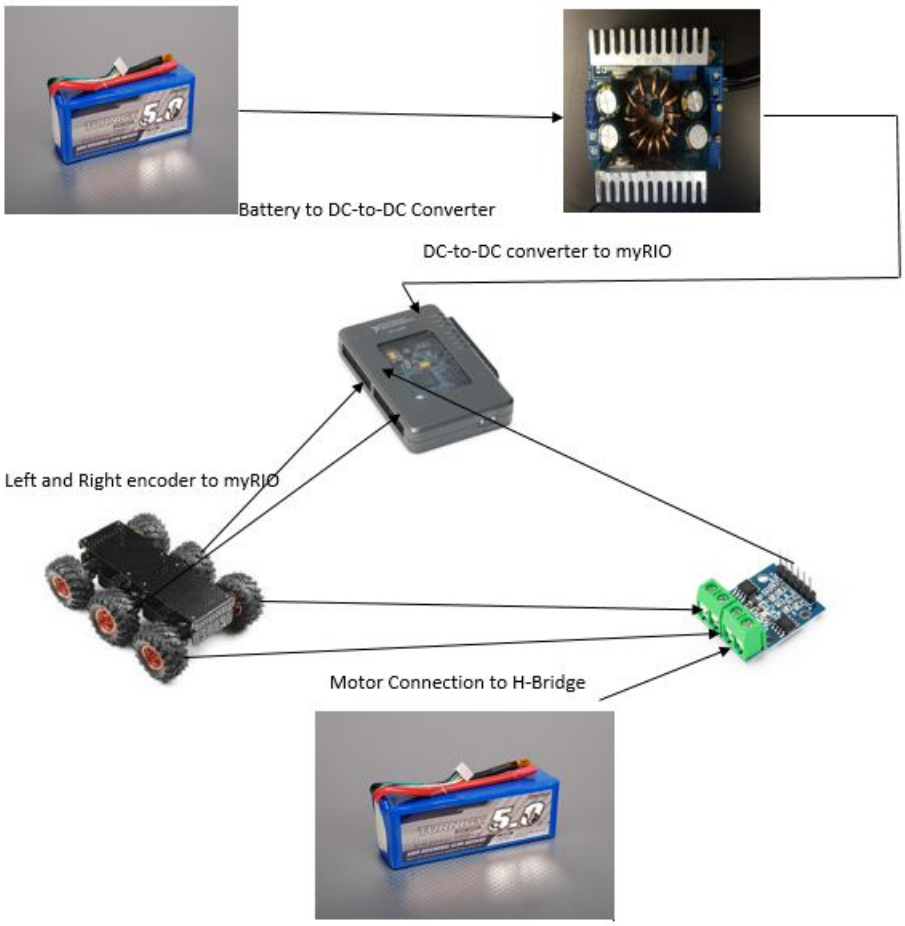

Figure 3.2: Electrical Diagram of the Whole System

3.2.2 myRIO

NI-myRIO, the portable embedded hardware device, is placed over the top chassis of the thumper and held on it through zip ties. The motor and encoder are connected to the myRIO A and B terminals. The left encoder is connected to channel A and the right encoder is on channel B. The motor connection to myRIO is through the H-Bridge. The myRIO is powered through a DC-to-DC convertor using a 14.4V LiPo battery.

3.2.3 Electric Motors

The thumper has six motors to drive the vehicle. The motor is controlled through the Pulse Width Modulation (PWM) signals given by the LabVIEW PWM VI via myRIO connection to the motor through an H-Bridge. The left and right side motor connection goes in the H-Bridge and from the H-Bridge 6 pins connect to the myRIO. The Pin connection to myRIO is as shown in Table 5.4.

Table 3.1: Motor Connection to myRIO

H-Bridge Pin	PIN Number
Power	C/+5V PIN 1
Ground	A/GND PIN 8
IN1 (Left PWM)	A/ PIN 31
IN2 (Left Digital output)	A/ PIN 32
IN3 (Right PWM)	A/ PIN 27
IN4 (Right Digital Output)	A/DIO0 PIN 26

3.2.4 Encoder

There are two encoders attached to the motors of the center wheels of the thumper. These are rotary encoders. It uses optical sensors to provide signals in

the form of discrete pulses which is translated into motion or position [16]. The encoder counts from each of the shafts are represented in a LabVIEW program with a ENCODER VI module. This represents the number of rotation value. Using two code tracks with sectors positioned 90 degrees out of phase is an encoder output channels provide position and direction of rotation of each wheel [1]. The encoders calculate the number of rotations by the wheel and display the results in the front panel VI. The encoder has four pin connections numbered 1-4. Two signal ports named signal A and signal B respectively are the output channels of quadrature encoder indicating position and direction of rotation [16], one +5V power supply and one ground, as listed in Table 3.2. The signal A is connected to the source terminal from which pulses are counted. Signal B is for the up/down terminal, if the up/down pulses are high then counting up takes place and if it is low the counter decrements the count. For realtime implementation, position information from quadrature encoder hardware is decoded. The decoded signals are increment or decrement counter value [16]. The left encoders are connected on Channel A on the myRIO and right encoders are on Channel B. The table 3.2 displays the PIN connection of the encoders to myRIO, the only change in left and right connections will be that the Enable Phase A and Phase B will be reversed for the right side connection.

Table 3.2: Encoder Connection to myRIO

Encoder Left	PIN Number
Power supply	A/+5V PIN 1
Ground	A/GND PIN 6
Enable Phase A	A PIN 18
Enable Phase B	A PIN 22

3.2.5 H-Bridge

An H-bridge is an electronic circuit which allows the DC motor to move forwards and backwards [17]. The two motors are connected to the H-bridge, and the supply to run the motor is also through the H-bridge. There are 6 connections between the H-bridge and the myRIO, one for supply and ground, two ports for receiving the PWM signal to control each motors and two for each of the motors digital outputs.

Figure 3.3: H-bridge

3.2.6 DC to DC converter

A DC-to-DC converter is an electronic circuit which converts a source of direct current from 12V level to 5V [18]. A step down DC-to-DC converter is used in this experimentation to power myRIO through a 14.4V LiPo battery.

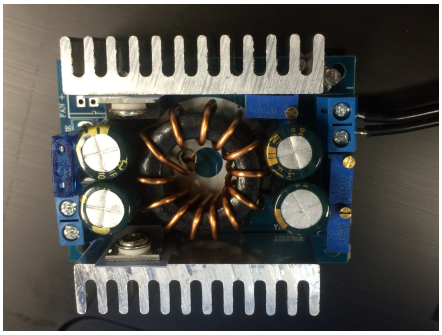

Figure 3.4: DC-to-DC Converter

3.2.7 Batteries

Two 14.4V Lithium polymer (LiPo) batteries are used to power the myRIO and the motors. These are rechargeable batteries of lithium-ion technology.

Figure 3.5: LiPo Battery

CHAPTER 4

UGV Model and GNC Algorithm

This chapter presents (1) the model of the UGV including the kinematics-based motion model, the sensor model, and the electric motor model, (2) a generic description of the GNC (Guidance, Navigation, and Control) algorithms and their interconnections.

4.1 UGV Model

This model is designed to graphically represent the UGV within the programming platform for carrying out simulations. It is also useful for checking the GNC algorithm within the platform before its real time hardware application. The UGV block diagram includes the parameter data such as size of vehicle, slippage factor etc. This block takes inputs from the duty cycle table, the duty cycle values are again converted back to the wheel speed values through the table data. Further mathematical operations generates the vehicle's actual position, velocity and orientation data. When carrying out simulations, this UGV graphical model is used along with the GNC in closed loop system. The encoder counts which we receive after the operations performed in UGV model are given as input to the GNC and so it becomes a closed-loop system.

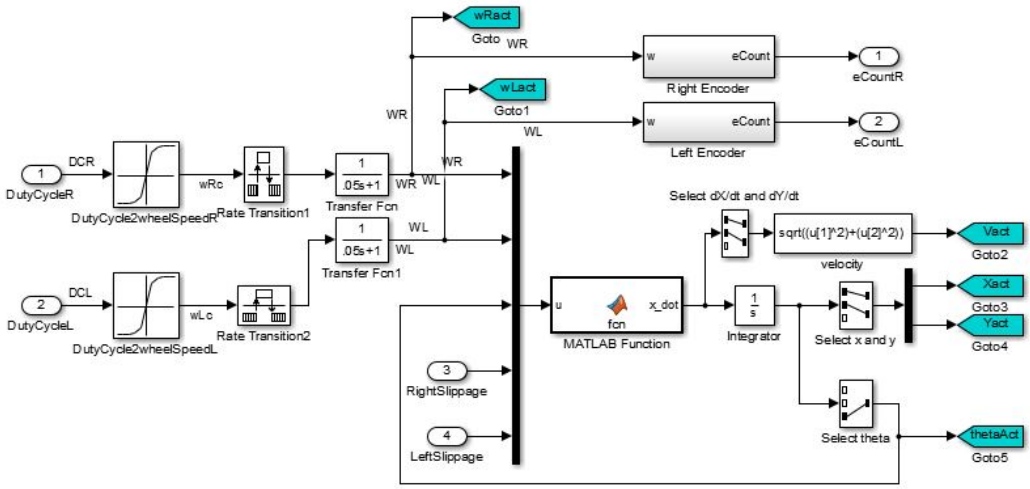

Figure 4.1: Block Diagram of Tank Model

4.1.1 Motion Kinematics

The translational and rotational velocity of the tank is calculated using the following formula:

$$\dot{X} = [(1 - s_r) \quad rr \quad \omega_r + (1 - s_l) \quad rl \quad \omega_l] \quad \frac{\cos\theta}{2} \tag{4.1}$$

$$\dot{Y} = [(1 - s_r) \quad rr \quad \omega_r + (1 - s_l) \quad rl \quad \omega_l] \quad \frac{\sin\theta}{2} \tag{4.2}$$

$$\dot{\theta} = [(1 - s_r) \quad rr \quad \omega_r - (1 - s_l) \quad rl \quad \omega_l] \quad \frac{1}{b} \tag{4.3}$$

where ω_r & ω_l angular velocities of the right and left wheel, s_r & s_l are slippage constants for right and left wheel, rr & rl are radius of the right and left wheel, θ is Heading Angle of the UGV, b is diagonal width of the vehicle, $\dot{X}$ & $\dot{Y}$ are velocity components in X and Y direction and $\dot{\theta}$ is turn rate of the Tank [19].

4.1.2 Encoder Model

The encoder model is used for simulation in the software platform to test the working of GNC algorithm. This model section generates encoder values which are used in the closed-loop simulation to input this to the input to the Navigation block. Encoder counts are calculated based on the simulated angular speeds of each wheel. By experiment, the encoder count increment per a specific distance traveled is determined. The distance traveled is proportional to the wheel speed through the wheel radius. Through these relations, encoder count is computed from the angular position of the wheels, which is obtained by integrating the wheel angular speed. Figure 4.2 shows the Simulink representation of the vehicle's encoder block.

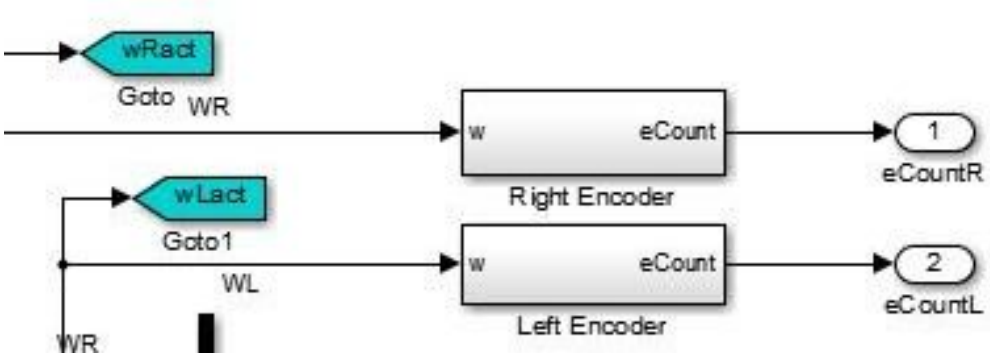

Figure 4.2: Encoder Model

4.1.3 Electric Motor Model

Electric motor is modeled to relate the PWM signals to the corresponding wheel angular speed. By experiments, a table is constructed that relates the duty cycle commands to angular speed of corresponding wheel. A specific percent duty cycle is commanded to the motors for a specific time. The distance traveled by the vehicle during this time is measured. These data are used to relate the specific duty cycle command to angular speed of the wheel. This test is repeated to cover the whole

range of the duty cycle. Further, a first order transfer function is added to represent the un-modeled dynamics of the drive-train of the vehicle [1].

4.2 GNC Algorithm

The GNC algorithm has three main sections with specific roles, as depicted in Fig. 4.3. The detailed working of each module are discussed in the following sections.

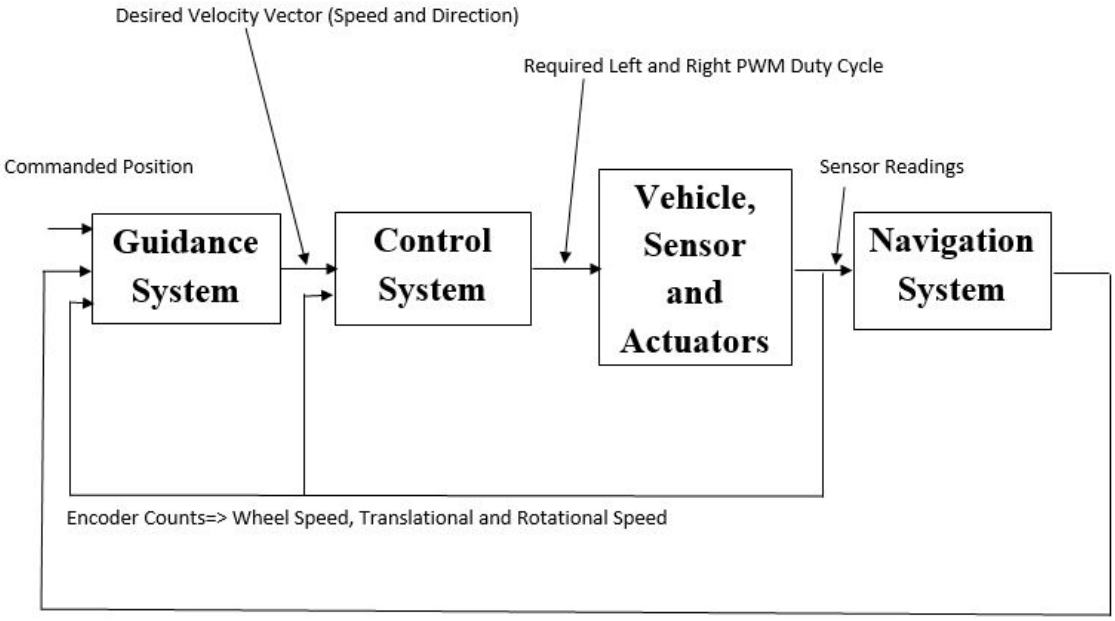

Figure 4.3: GNC of UGV [1]

4.2.1 Navigation

Navigation in this context is to estimate the position and orientation of the UGV based on the encoder readings from the left and right wheels. The specific method used for navigation is called the dead reckoning and utilizes the kinematics model of the skid-steered platform. Dead reckoning is subject to cumulative errors [20]. During navigation it is a method of advancing position based on known or estimated speed over elapsed time [20]. Rotary encoders are provided on each of the wheel. There is an increment in the encoder count which indicates rotation of the attached shaft or wheel. The counts are in terms of the ticks per inch traveled by the vehicle [1]. Wheel speed based on the increments in encoder counts within a sampling period is calculated by

$$\omega_k = (eCount(k) - eCount(k-1))/(eTick \times r \times TsampleEncoder) \tag{4.4}$$

where the numerator represents the number of ticks in one sampling period, $eTick$ is ticks per minute, r is wheel radius and $TsampleEncoder$ is sampling period in seconds. The number of ticks helps in detecting overflow and underflow. If absolute value of the tick increment per a sampling period is greater than 32000, overflow is detected because the vehicle does not move that fast. The wheel speed calculation is done separately for each wheel. When an over/underflow is detected, the speed calculation, as formulated in Eq. 4.4, uses encoder counts from k-1 and k-2 samples (where k is the current discrete time), instead of k and k-1 samples. In the dead reckoning section, no slippage is assumed, also the left and right wheel radii are considered to be the same. Thus, the kinematics equation becomes

$$\begin{aligned} \dot{X} &= [\quad r/2 \times \quad \omega_l + \omega_r] \quad \frac{\cos\theta}{2} \\ \dot{Y} &= [\quad \omega_r + \quad \omega_l] \times \quad r/2 \times \quad \frac{\sin\theta}{2} \\ \dot{\theta} &= [\quad \omega_l - \quad \omega_r] \times r/ \quad b \end{aligned} \tag{4.5}$$

Numerical integration of these three results in estimated position (in terms of x and y coordinates) and orientation (angle θ) of the vehicle. Fig. 4.4 shows the Simulink implementation of this navigation algorithm.

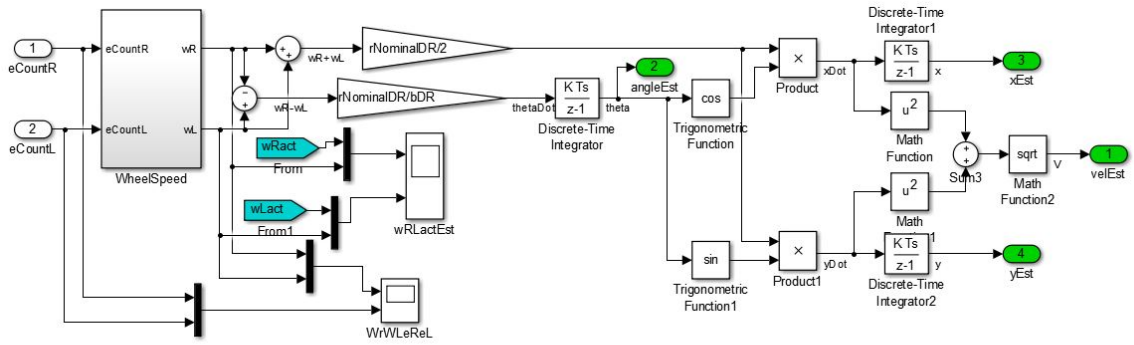

Figure 4.4: Navigation Simulink Algorithm

The output we receive from the navigation section- estimate position coordinates of x- and y-axis, along with velocity and theta estimate values are used as input for the other two blocks.

4.2.2 Guidance

Guidance algorithm generates the speed and heading commands for the vehicles based on the current position and orientation of the UGV to go through a given set of waypoints with a specified speed schedule. Figure 4.5 displays the guidance section in the entire GNC algorithm. It takes in estimated x and y- coordinate values calculated through dead reckoning. Based on the estimated position and the position of the immediate way point to go to, commanded/desired heading angle θ_c is computed as the angle of the direct line from the current position to the position of the waypoint. In

addition to the orientation, it uses the distance formula to determine its distance of the vehicle from the desired waypoint. Once it reaches the particular waypoint, it inputs the next waypoint from the look up table. The distance between the waypoint is compared with the rp2 constant value, which is the radius of the second circle around the waypoint. Once the UGV reaches the last waypoint, it halts the simulations, hence brings the vehicle to stop. Along side, the commanded/desired velocity of the vehicle is also calculated depending on the distance output. If the vehicle is close to the waypoint, then the velocity decreases, this change in speed is modulated by the Proportional, Integral and Derivative control algorithm used in the Control section.

$$\theta_C = atan2((y - yi)/(x - xi)), r = \sqrt{(y - yi)^2 + (x - xi)^2} \tag{4.6}$$

where θ_C is angle the UGV would turn towards the next waypoint, r is the distance to the waypoint and i represents the index of the immediate next waypoint [1].

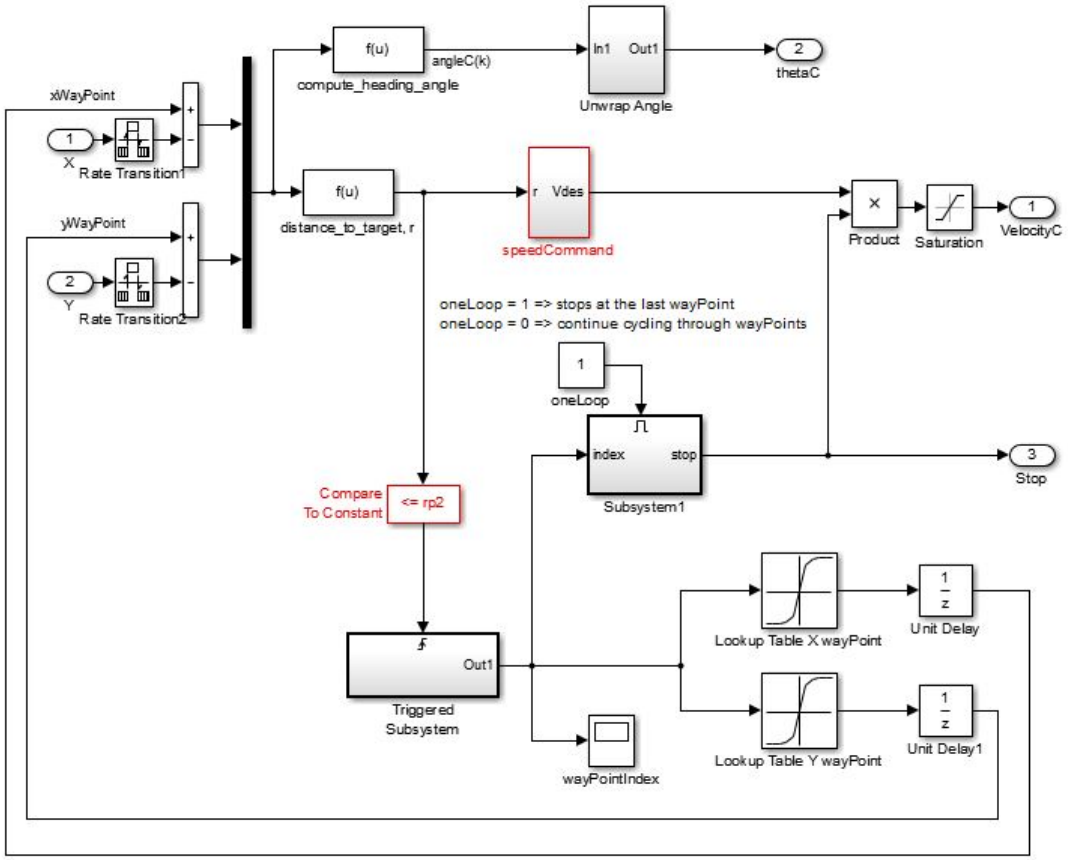

Figure 4.5: Guidance Algorithm

4.2.3 Control

Control module is to generate the right and left wheel speed commands given the current speed and heading of the vehicle and the commanded speed and the heading coming from the guidance module. The PID controllers are used to compute the mean wheel speed and differential wheel speed commands. Since the mean wheel speed is related with the speed of the vehicle platform, based on the kinematics of the vehicle, the mean speed PID controller is fed with the error in speed, i.e. the difference between the estimated speed by the navigation module and the commanded speed

calculated by the guidance module. Similarly, the differential speed is related with the rotation of the vehicle, and thus the differential speed PID is fed with the error in heading, i.e., the difference between the current heading and the desired heading towards the way point. From the mean and differential speed, the desired left and right wheel speeds are computed. Using the look-up tables that relate the wheel speeds with the duty cycles supplied to each motor, the required duty cycle for each wheel based on the desired wheel speed is computed.

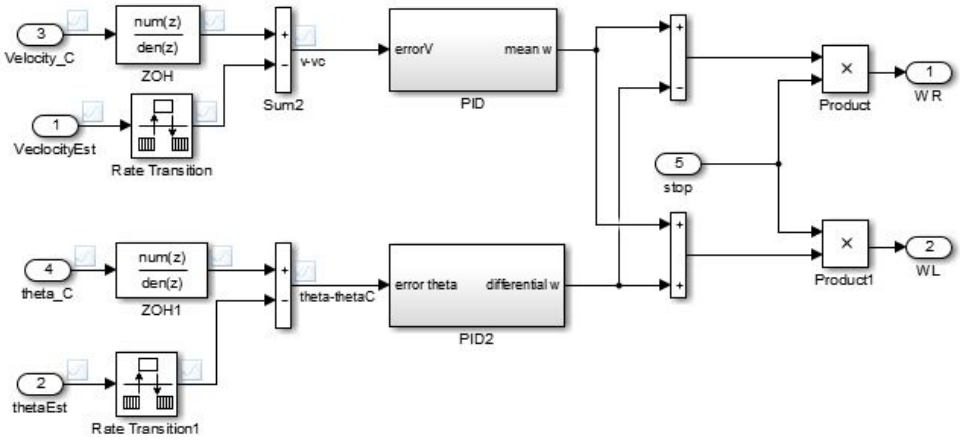

Figure 4.6: Control Algorithm

4.3 MATLAB/Simulink Implementation

In MatLAB/Simulink, the GNC blocks are designed as subsystems. Each block has different type of subsystem called in to perform required task. Fig. 4.7 shows the GNC subsystem with the vehicle subsystem in a closed loop form. The Simulink version of the GNC algorithm has some parts in which the MATLAB code is called in for data. A MATLAB file is created which has the vehicle and GNC parameters

values like the eTick, wheel radius, sampling times and the lookup table data. These files are run first such that the parameter values get stored in the memory and then the simulink file is run.

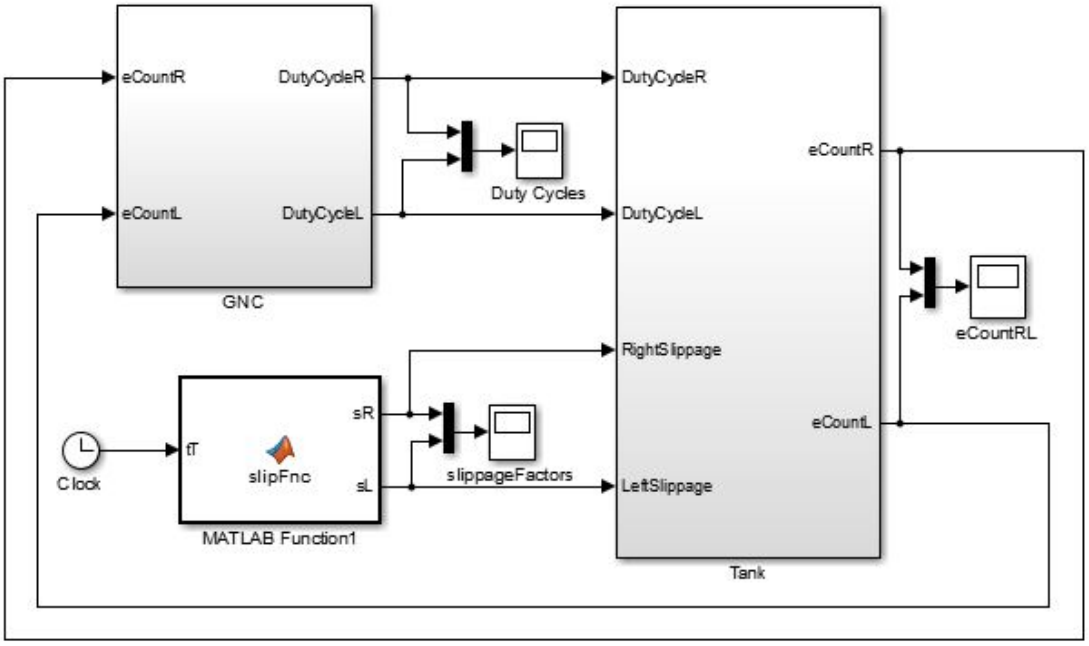

Figure 4.7: GNC with Tank

Figure 4.8 depicts the hardware implementation that has the NUC, phidgets and the vehicle as the three main components. For using this algorithm on a UGV, a Intel NUC Mini PC is used, which is a tiny PC with quad-core Intel Pentium processor, with built-in WiFi, Bluetooth and 8-channel audio along with HDMI and USB ports [21]. The purpose of NUC placed directly on the vehicle, and connected with the encoders and motors through a specialty board, is to run the MATLAB/Simulink model consisting of the GNC algorithm, and read encoder counts and computes duty

cycles for the PWM signals sent to the electric motors. Phidgets, the specialty board serving as the USB-based interface between the PC and the encoders and the electric motors, provides the GNC Simulink model with the encoder readings, and generates the PWM signals for the electric motors based on the duty cycle commands computed by the GNC algorithm [22]. The PWM signals generated by the Phidget boards are supplied to the electric motors that drives the left and right wheels to control the direction and the speed of the vehicle.

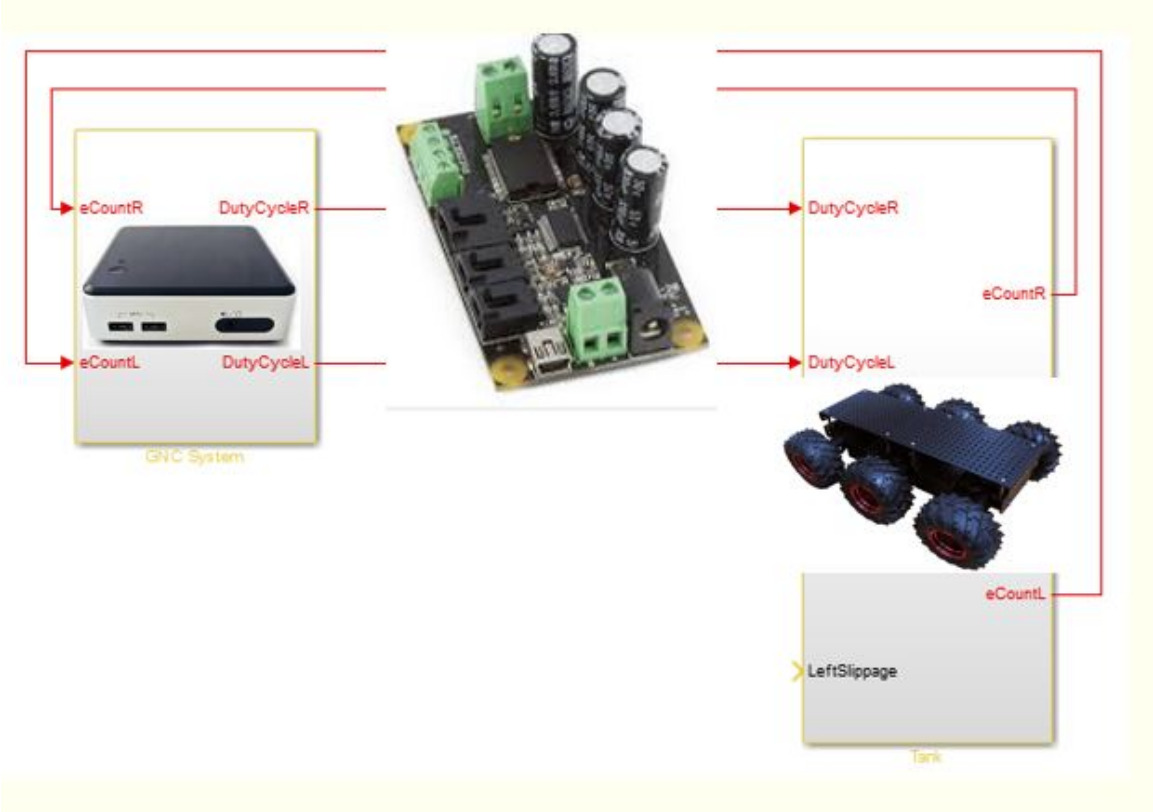

Figure 4.8: Hardware Implementation

CHAPTER 5

Implementation of UGV Model and GNC algorithm in LabVIEW

This chapter describes the design of GNC algorithm in LabVIEW along with the results obtained from the LabVIEW simulation of GNC and tank block.

5.1 Control and Simulation Loop

The closed-loop system (the vehicle model and the GNC connected through feedback) created within the Control and Simulation loop is similar to that in MATLAB/Simulink. The Control and Simulation module provides blocks for simulation of dynamic systems, design GNC systems and their real-time hardware implementation. It allows mathematical operations to be performed using individual blocks or codes written within the LabVIEW MathScript RT module. It also provides discrete blocks such as the unit delay, transfer function and integrator which otherwise cannot be accessed in normal LabVIEW module. For the simulation of the closed loop system, discrete solver method is used. Unlike Simulink where the GNC and the vehicle modules are set up in the form of subsystems, here in LabVIEW the entire closed-loop system is within one single Control and Simulation loop with simulation time set as 300 seconds and discrete step size as 0.1.

5.2 UGV Model

In LabVIEW, the entire GNC algorithm and the tank block is designed within a single Control and Simulation loop. Unlike Simulink where subsystems could be used, in LabVIEW with a Control and Simulation loop creating subsystems results

in one particular loop running till the simulation time ends for that loop and then proceed to the next subsystem.

5.2.1 Motion Kinematics

In this section, the vehicle's current position, orientation and velocity is calculated by representing Eqs. (4.1), (4.2) and (4.3) in graphical form in LabVIEW. Figure 5.1 shows the representation of the kinematic equations of the vehicle, which help in determining the vehicle's position, orientation and velocity in LabVIEW model.

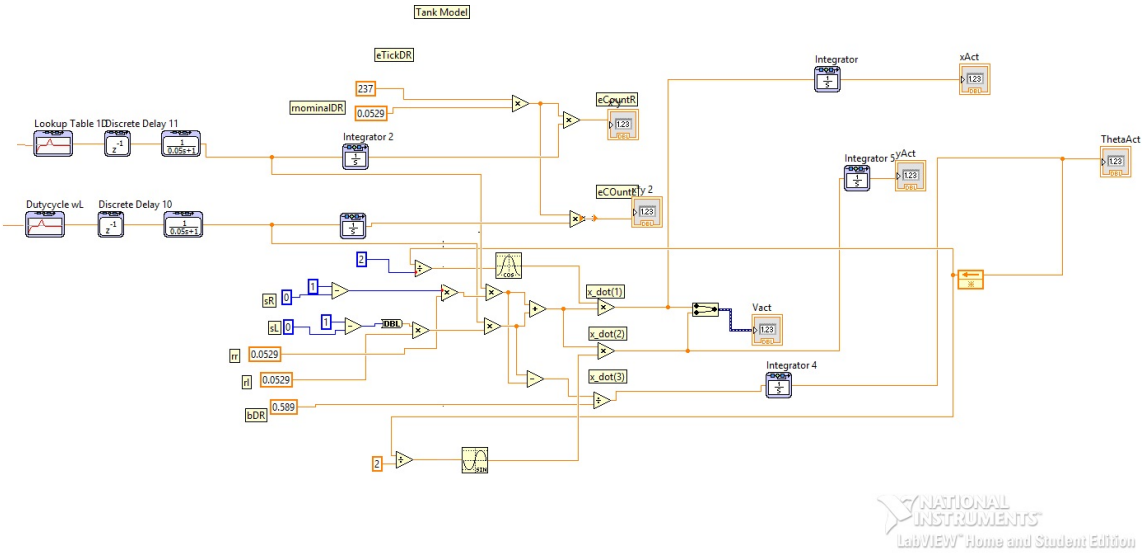

Figure 5.1: Motor Model

5.2.2 Encoder Model

The encoder model representation in LabVIEW basically include the eTick value multiplied with the wheel radius (rNominal). Fig. 5.2 shows the right wheel eTick value multiplied with rNominal and this output is multiplied with the discrete integrator output from the motor model section to generate right wheel encoder values.

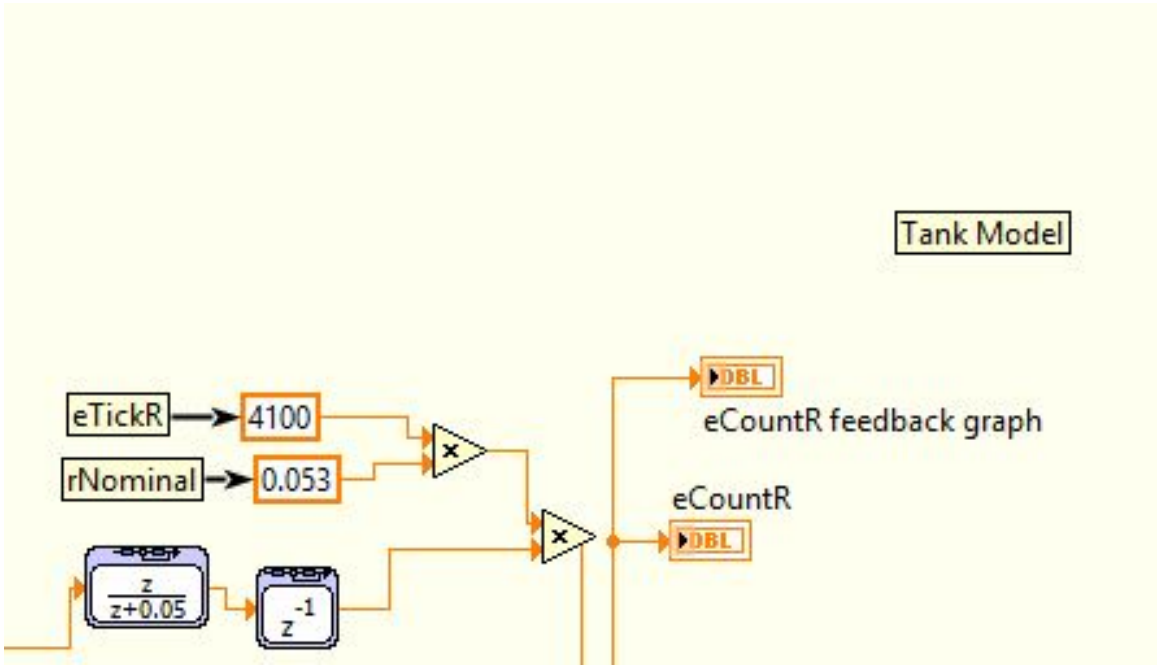

Figure 5.2: Right Encoder Model

Similarly, Fig. 5.3, shows the encoder calculations of the left wheel, obtained by multiplying the left eTick with rNominal and the left wheel motor model output.

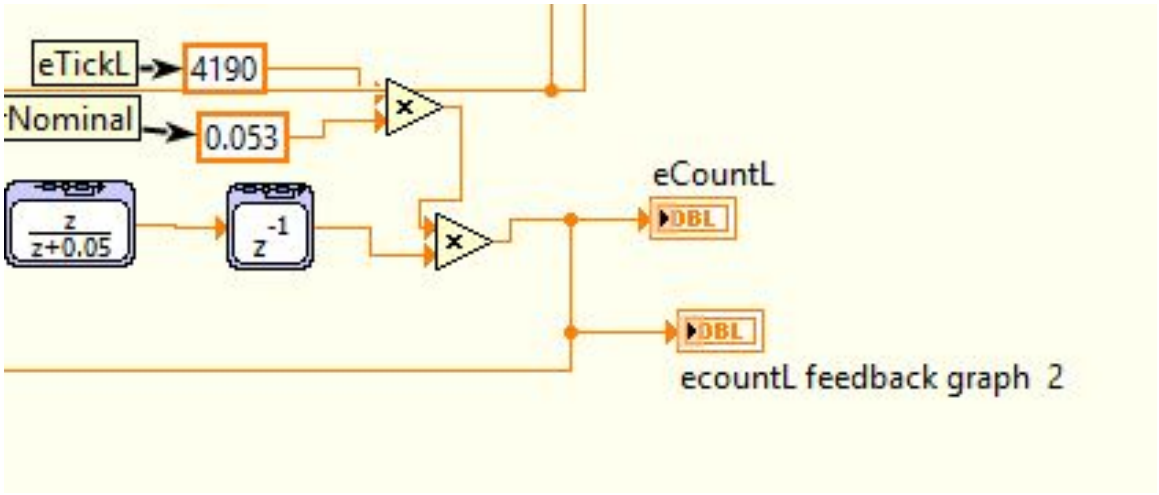

Figure 5.3: Left Encoder Model

5.2.3 Motor Model

The motor model developed in simulink had continuous transfer function and unit delays to represent the vehicle's motion and generate required data. In LabVIEW these block are replaced by discrete transfer function and unit delay as the overall solver for the Control and Simulation loop is "Discrete States only".

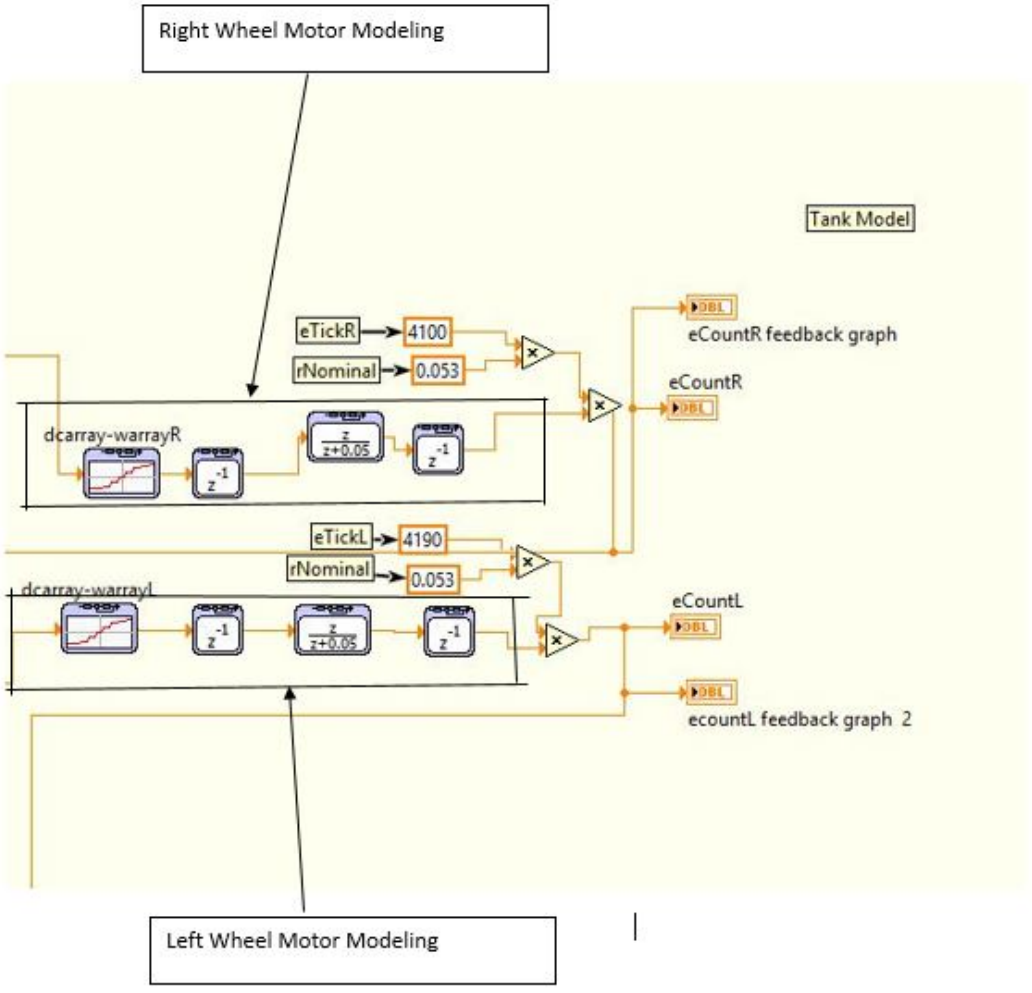

Figure 5.4: Motor Model

Figure 5.4 shows the right and left motor model sections, wherein discrete transfer function and unit delay are used.

5.3 GNC algorithm

In this section, implementations of the three main modules of GNC algorithm in LabVIEW are discussed.

5.3.1 Navigation Module

The navigation section is block by block representation of the Simulink module in terms of performance. Fig. 5.5 shows the blocks used in the wheel speed calculation process. In LabVIEW implementation, the constant parameters used in angular speed calculation are directly entered into the model as constant blocks. In Simulink, these vehicle parameters were called in from the workspace, and these values are loaded into the workspace by a MATLAB script file, called before the start of simulation.

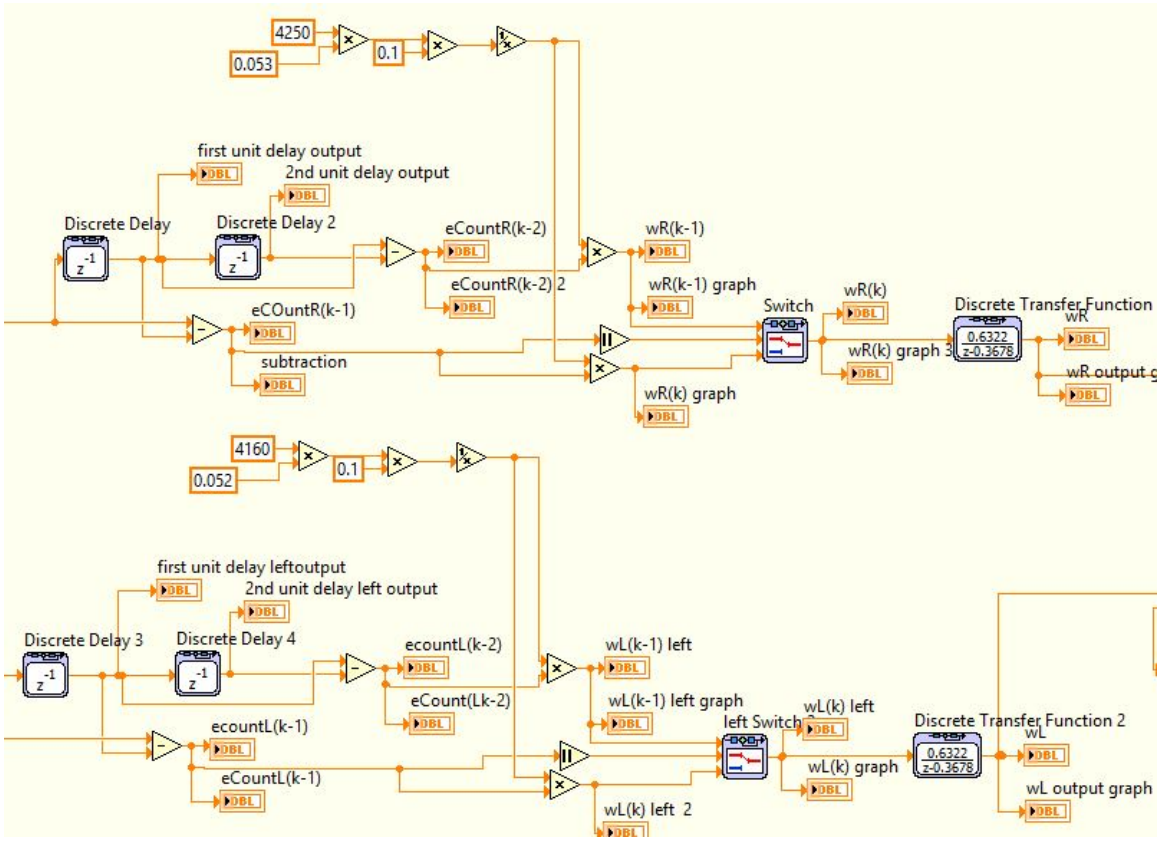

Figure 5.5: Block Diagram of Speed Calculation of Navigation Section

The output of the wheel speed is used for further calculations for computing the xEst (estimated position of vehicle on the x axis), yEst (estimated position of vehicle on the y axis), thetaEst (estimated orientation of the vehicle) and the velEst (estimated velocity) as shown in Fig. 5.6. These are computed according to the movement of the vehicle. This section where the wheel speed data is used to compute the vehicle's position and orientation is called deadreckoning.

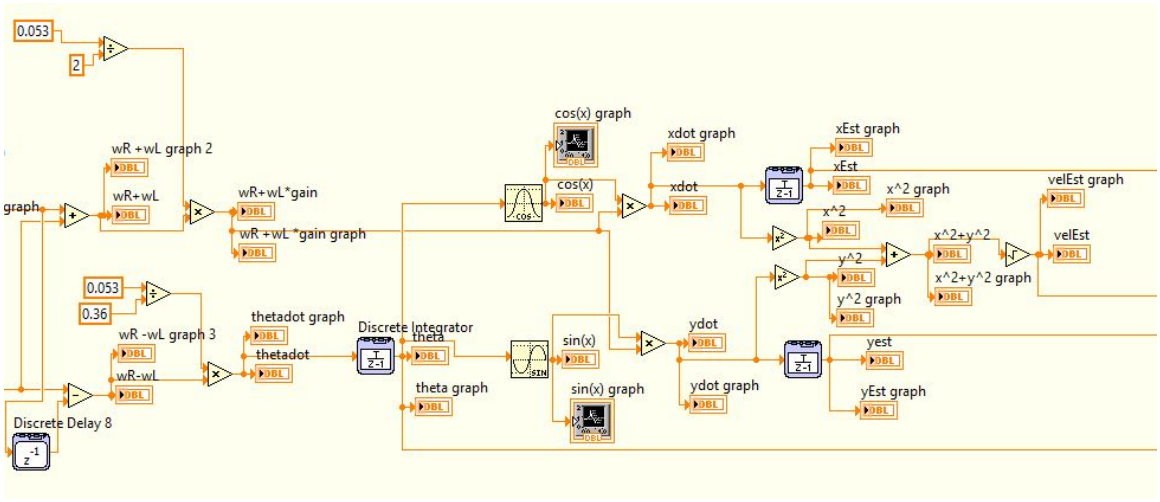

Figure 5.6: Block Diagram of Deadreckoning Calculation of Navigation Section

5.3.2 Guidance Module

The guidance module has three sections, the unwrap angle, speed command and triggered subsystem. In simulink there is a triggered subsystem module which is used within these three sections, this triggered subsystem is replaced by a formula node within LabVIEW depending on the type of usage. In the unwrap angle and triggered subsystem section, the formula node is used. Within the formula node, an if-statement is used. If the atan2 angle is greater than 3.14 then the counter is incremented and if the angle is less than -3.14 then the counter is decremented as shown in Fig. 5.7. Other than this change the overall operation within the unwrap angle is the same as the simulink implementation.

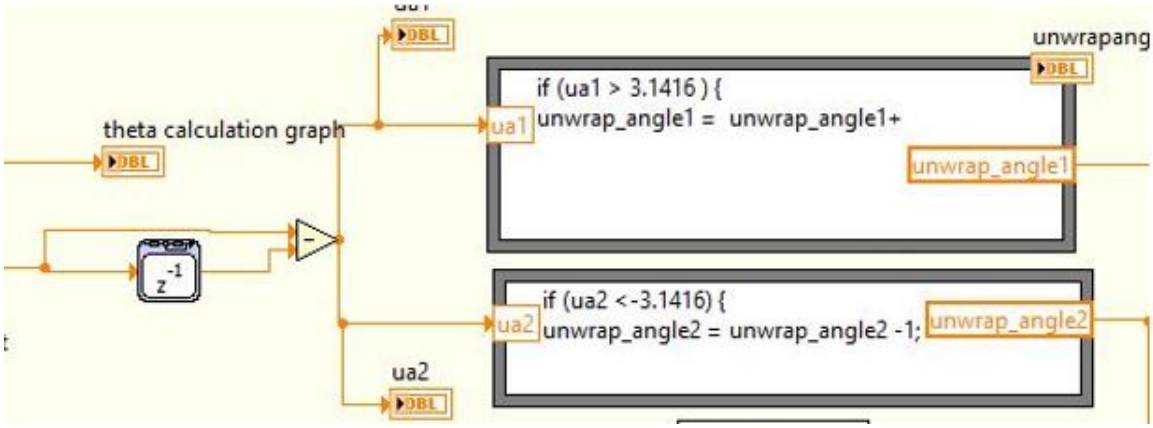

Figure 5.7: Block Diagram of Unwrap Angle Section

In the triggered subsystem section where it is required to generate waypoint index a formula node is used. Here again an if-condition is applied to increment the counter. If the counter reaches the length of waypoint array then the simulation and the vehicle stops. This is represented in Fig. 5.8.

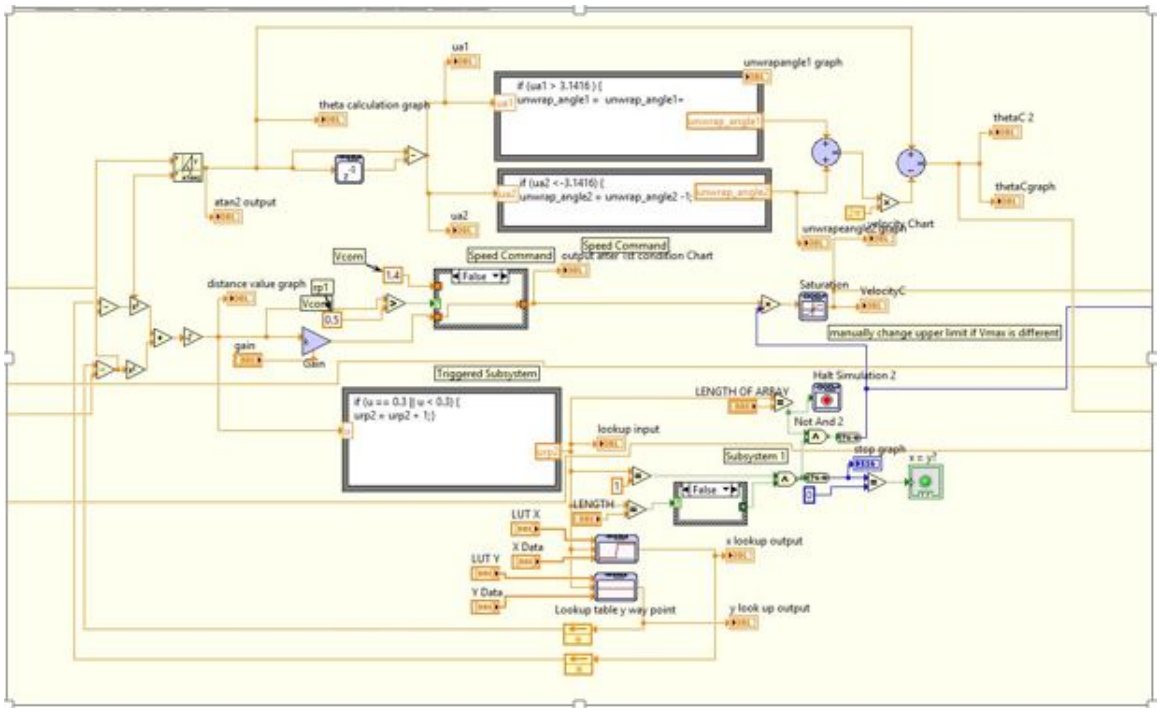

Figure 5.8: Block Diagram of Guidance Section

The results generated from the guidance algorithm is verified with the Simulink to test the accuracy of this section.

5.3.3 Control Module

LabVIEW provides different types of PID modules for correcting and controlling the vehicle's speed and orientation. Fig. 5.9 represents the Control module design in LabVIEW. Fig. 5.9 shows LabVIEW representation of Control Section which is similar to that in Simulink.

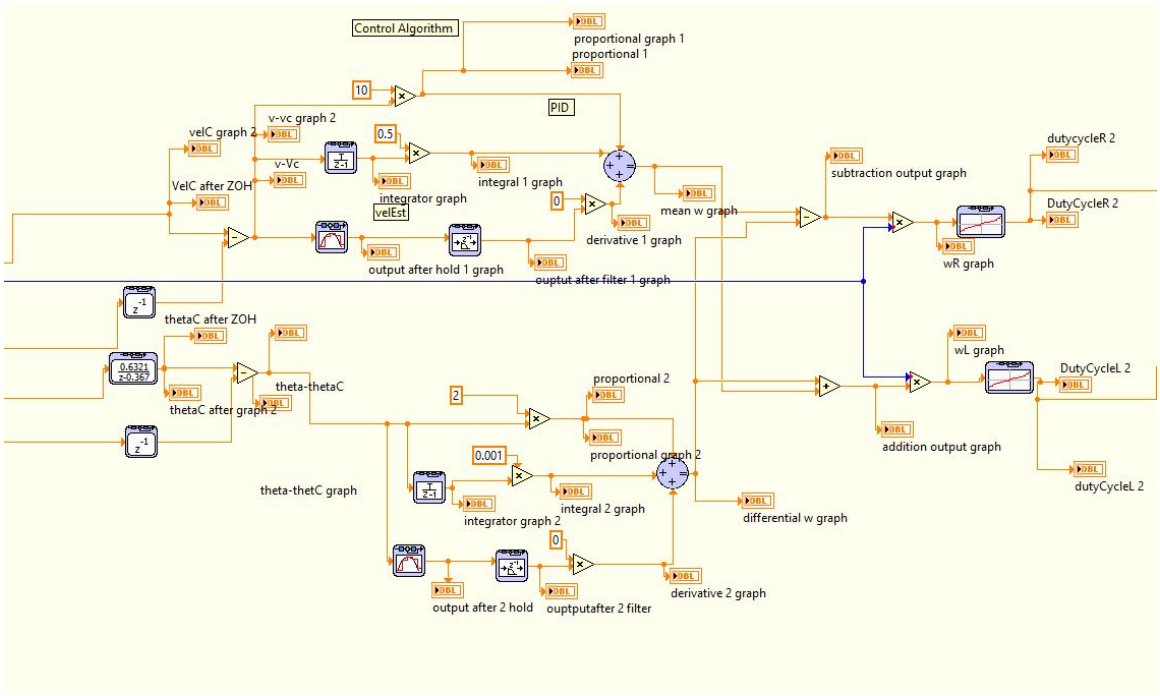

Figure 5.9: Block Diagram of Control Section

5.4 Results of Closed Loop Simulations

Before running the closed-loop simulation for any set of waypoints, in the front panel following parameters and waypoints are to be entered.

1. KP1 and KP2 (speed and turn angle gain values) are to be entered. The values used are KP1 = 14 and KP2= 5.
2. LUT X and LUT Y are the Index values of the x and y coordinates of the waypoints, these are not to be altered, but needs to be increased in size according to the number of waypoints we specify.
3. X data and Y data array are to enter the x and y coordinates of the waypoints the vehicle is commanded to go to.

4. Length of Array specifies the number of waypoints the vehicle is expected to travel so as to stop at the last waypoint. The value entered here should be one greater than the actual length of the waypoint array.

5.4.1 Case 1

In this case the vehicle runs in L shape, turning both left and right.

Table 5.1: x- and y- coordinates for waypoint Navigation Case 1

x	y
3	0
3	2

Output for the right turn L shape (x- and y- coordinates are given in Table 5.1) is as shown in Fig. 5.10.

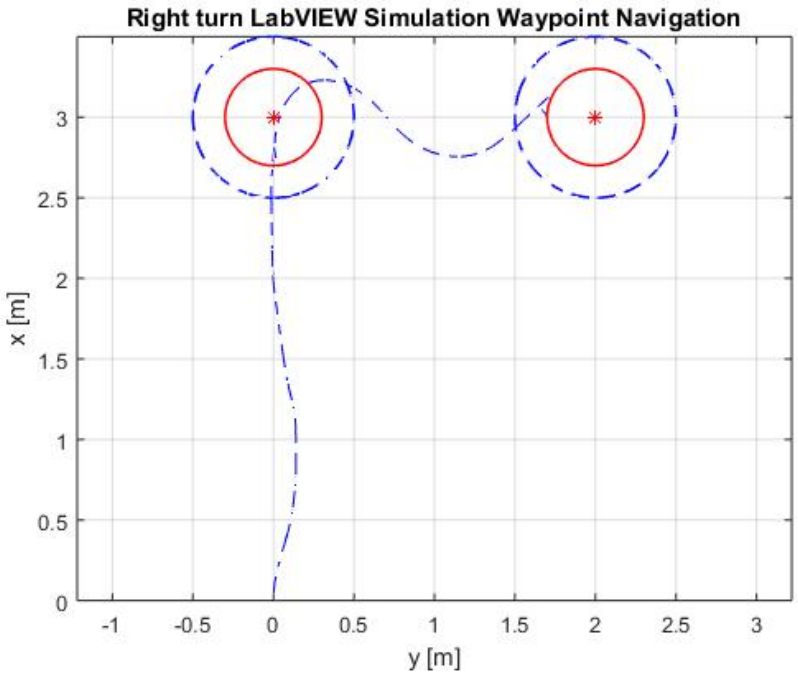

Figure 5.10: waypoint Navigation Output for Case 1

Table 5.2: x- and y- coordinates for waypoint Navigation Case 1b

x	y
3	0
3	-2

Output for the left turn L shape (x- and y- coordinates are given in Table 5.2) is as shown in Fig. 5.11.

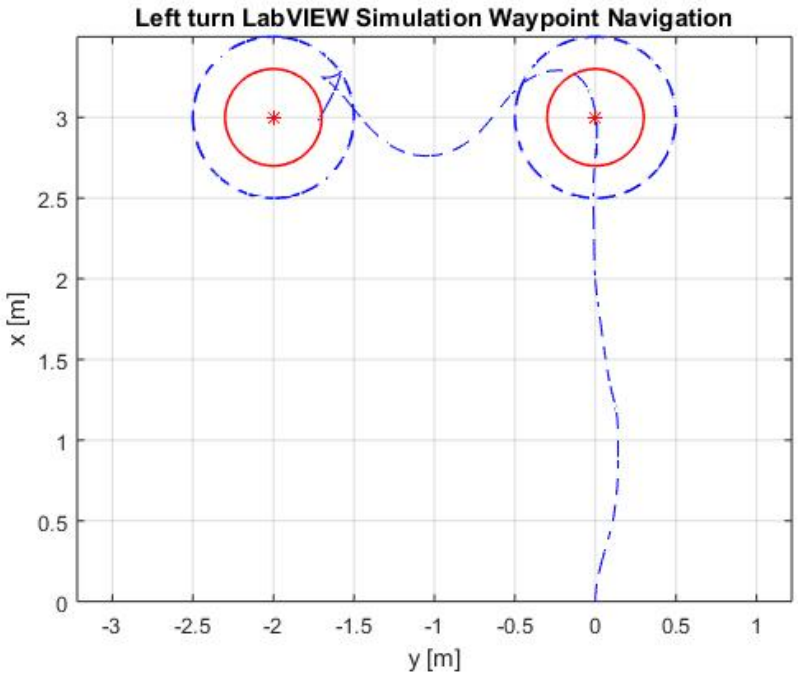

Figure 5.11: waypoint Navigation Output for Case 1b

5.4.2 Case 2

In this case the vehicle runs in triangle shape, making two left turns. Output for this simulation (x- and y- coordinates are given in Table 5.3) is as shown in Fig. 5.12.

Table 5.3: x- and y- coordinates for waypoint Navigation Case 2

x	y
3	2
3	-2
0	0

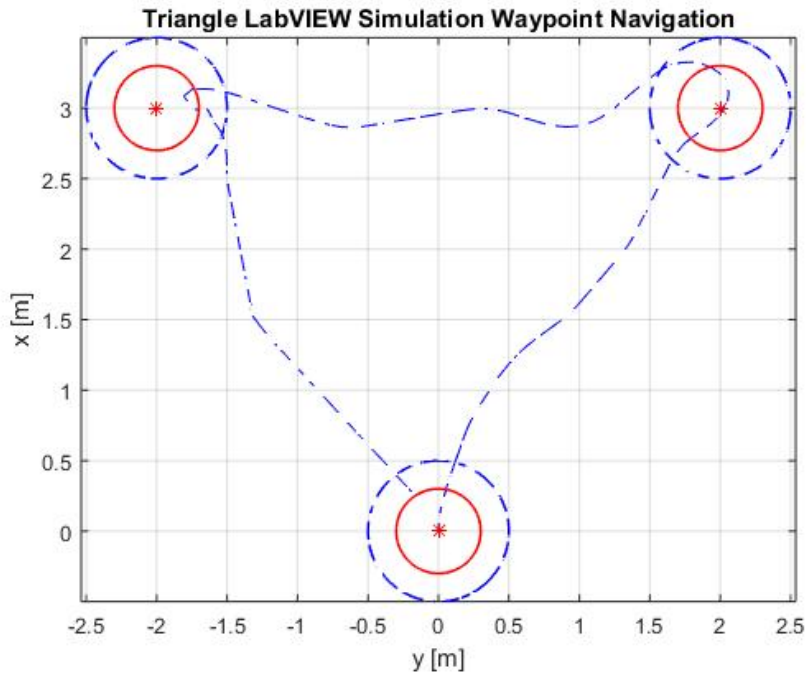

Figure 5.12: waypoint Navigation Output for Case 2

5.4.3 Case 3

In this case the vehicle runs in Z shape, turning both left and right. The

Table 5.4: x- and y- coordinates for waypoint Navigation Case 3

x	y
2	0
2	-3
4	-3

coordinates of the waypoints are given in Table 5.4 and the output of the simulation is shown in Fig. 5.13.

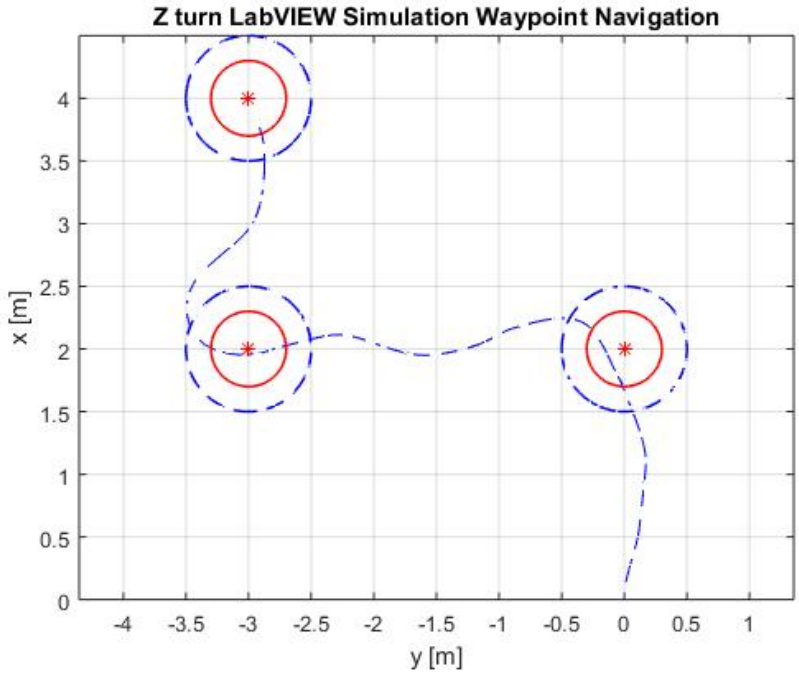

Figure 5.13: waypoint Navigation Output for Case 3

CHAPTER 6

Implementation of GNC algorithm in myRIO

6.1 Modifications to GNC VI for myRIO implementation

The tank VI in the GNC algorithm is replaced with the Encoder Express VI and Motor Control Model. These express VI are provided by the myRIO module to have real-time data collection from the encoders and continuously control the motor speed by varying the dutycycle. The final setup of the vehicle with myRIO is shown in Fig. 6.1.

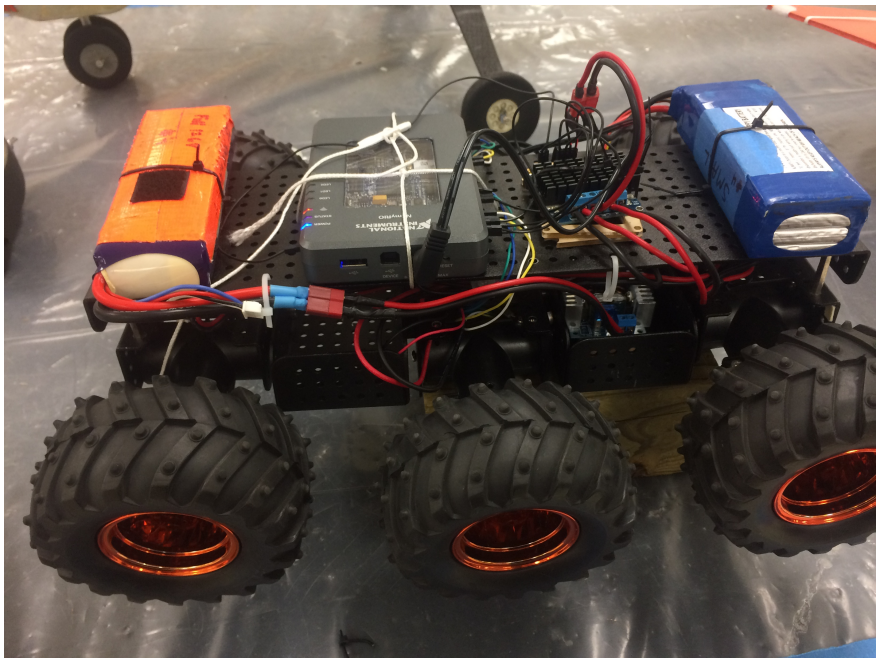

Figure 6.1: Vehicle and myRIO setup

6.1.1 Encoder VI

Encoder express VI (see Fig.6.2) is used to read and decode signals from the encoder connected to myRIO kit. It reads the number of ticks/rotations generated by the motion of the UGV wheels. The channel parameter specifies the channel to which the encoder is connected. The right encoder is connected to channel A and left encoder is connected to channel B. The Pin connections are as mentioned before. Encoder output signal is a quadrature phase signal (see Fig.6.3). A set of experimental run is done so as to calculate the different parameter values. Thus eTickR (Right wheel eTick) = 4100 ticks per meter and eTickL (Left wheel eTick)= 4190 ticks per meter. Two encoder express VI's are used, one for each wheel.

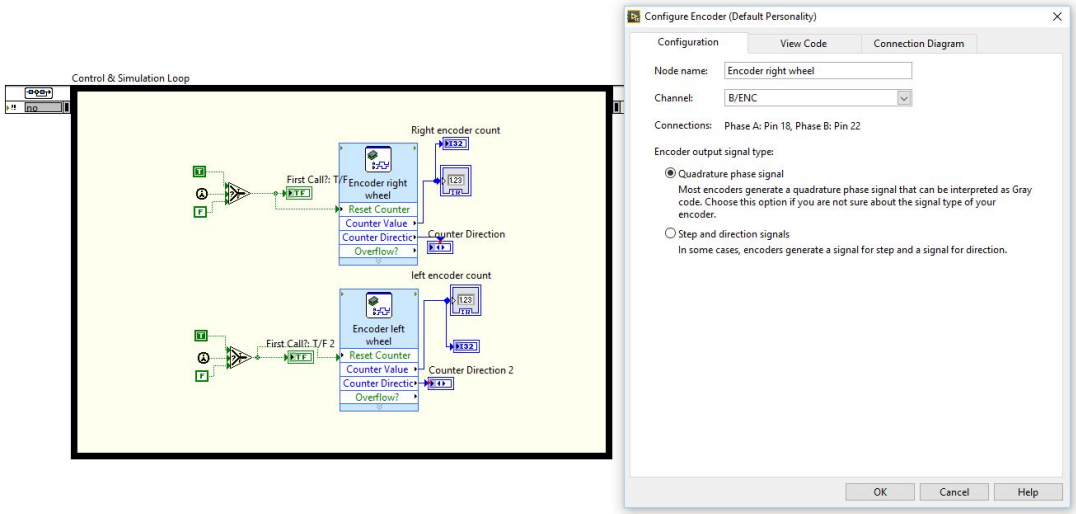

Figure 6.2: Encoder VI with Configuration Block

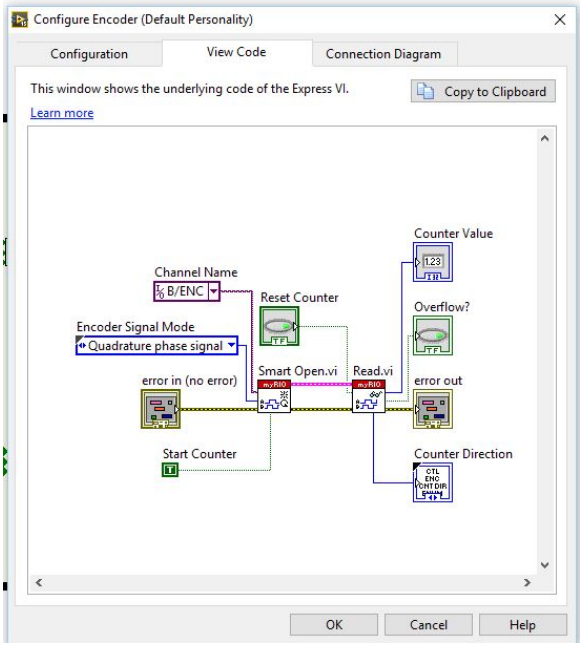

Figure 6.3: Encoder Code Diagram

6.1.2 Motor Model VI

For motor control, two express VIs are used. One is the PWM VI and another is Digital output VI. PWM VI takes in the duty cycle values, and makes the motor move according to the duty cycle input given. The motion of the motor has a digital value; to write this digital output on the myRIO, a Digital Output Express VI is used. The PWM VI takes in duty cycle in the range of -1 to 1 so the dutycycle value from the lookup table is divided by 100. Fig. 6.4 shows the use of a case structure to design a motor control VI; the reason for this is to allow the motors to move straight, left, right and reverse with a single VI program. With the case structure, it ensures that if the dutycycle input to the PWM VI is positive the the vehicle moves forward, True test case, and when the dutycycle is negative then the case structure condition is False and thus the vehicle will move according to the condition.

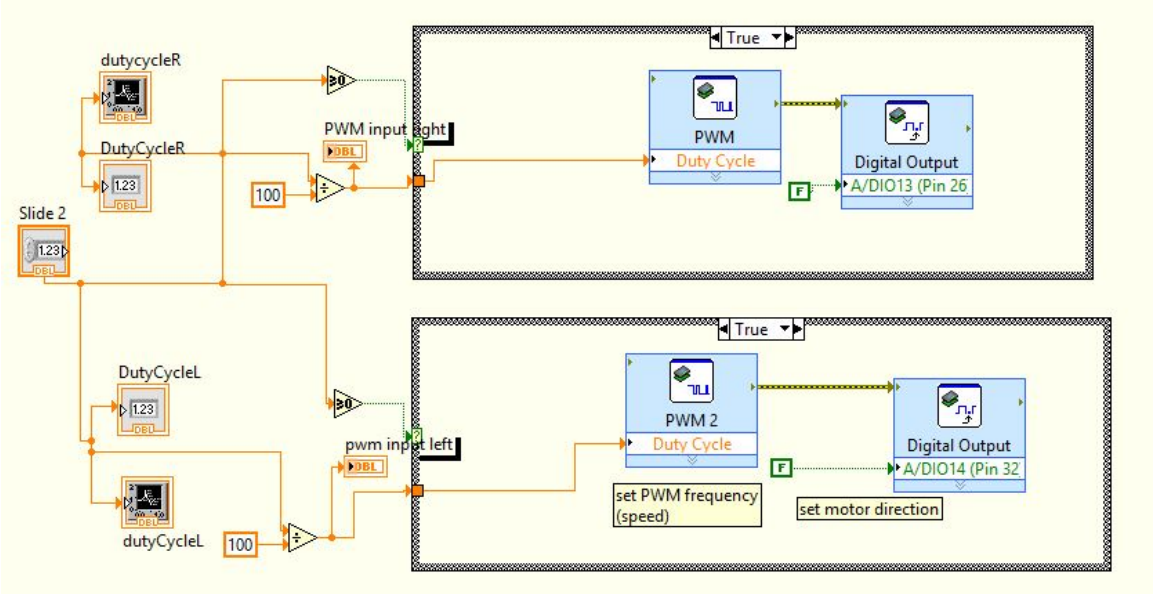

Figure 6.4: Motor Control VI

PWM Express VI generates a pulse width modulation (PWM) signal given to the motors through the PWM channels on the myRIO. Within this PWM VI, there are three inputs, duty cycle, frequency and error in and one error out as the output for the VI. The duty cycle input to the PWM is given from the lookup table that maps desired wheel speed to percent duty cycle. The frequency is set at 40Hz constant value (see Fig.6.4). The error out from the PWM VI contains the standard error information which is fed as input to the Digital Output VI error in port. The pin connection for left PWM signal is 27 and for the right PWM signal is 31, these are connected on port A of the myRIO board. From the myRIO, these connections go to the H-Bridge IN1 and IN3 ports, respectively.

Digital Output Express VI writes values to the digital output channel on the myRIO. Fig.6.4 shows the digital output VI connection with the PWM VI. This VI has two inputs, one is the error in, getting signal from error out of the PWM VI, and a channel name where the value is written to the output channel chosen. For the left

wheel Pin 26 and for the right wheel pin 32 is chosen as the digital output ports. The other side of myRIO connections are to the H-Bridge IN2 and IN4 pin, respectively, for left and right wheels.

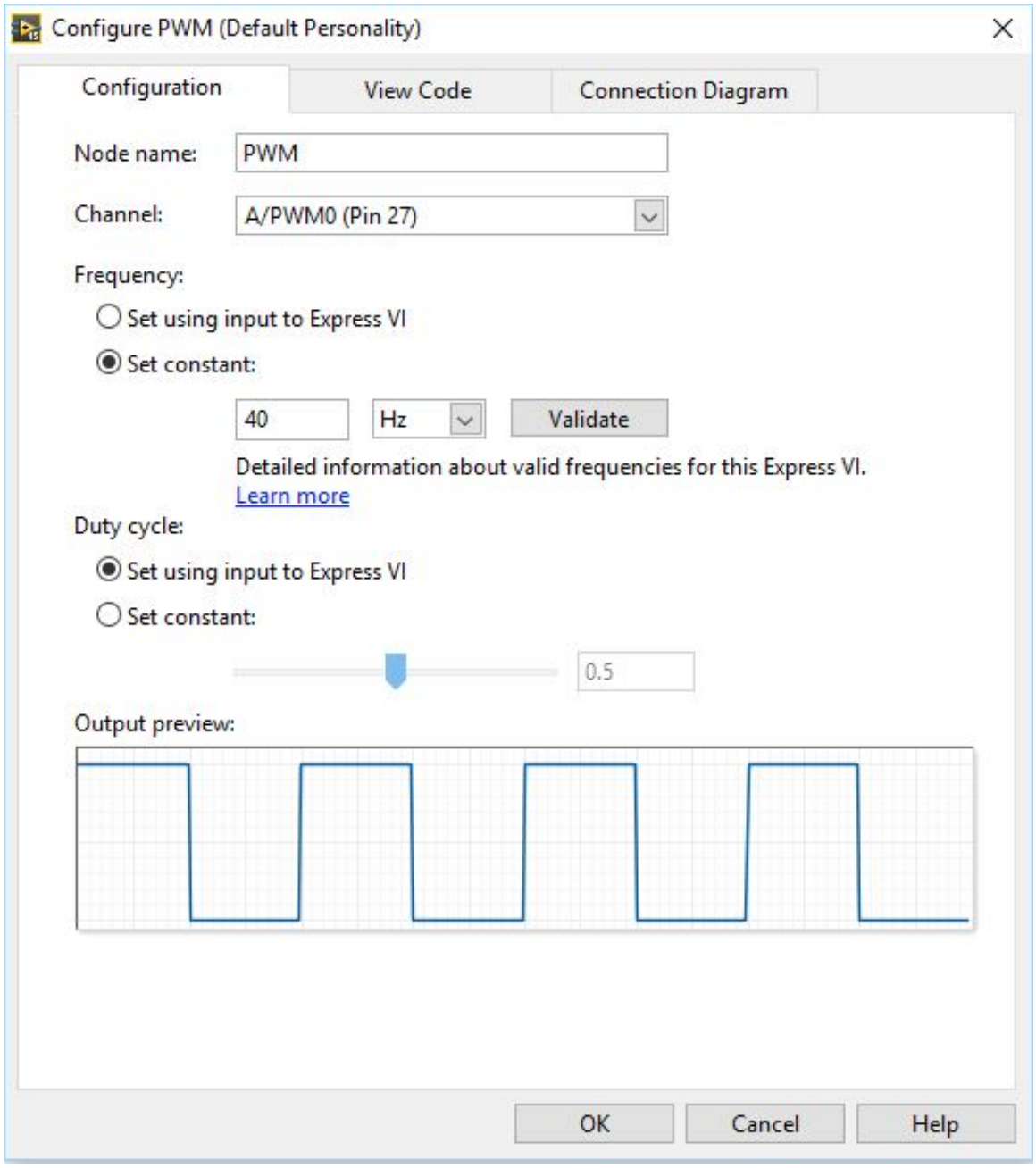

Figure 6.5: Configuration of PWM signal

6.2 Results of Experiments

Through the calibration of the vehicle, its parameters are chosen as follows:

1. rNominalDR = 00529 meters. It is the nominal wheel radius in meters
2. bDR = 0.39 meters It is the effective platform width
3. eTickR = 4100 Number of Ticks per meter
4. eTickL = 4190 Number of Ticks per meter
5. $dcArray = [-50, -10, 0, 10, 50]$
6. $wArray = [-8.119, -0.008119, 0, 0.008119, 8.119]$

With myRIO kit and LabVIEW flow diagram, waypoint navigation was successfully achieved for the same set of waypoints used in the simulation as discussed in the previous section. After the vehicle reached the last waypoint, it was stopped by to the condition mentioned earlier by halting the program.

6.2.1 Case 1

The waypoints in this experiment are the same as the ones given in Table 5.1. The result of the experiment in terms of the estimated position by the navigation module is shown in Fig. 6.6.

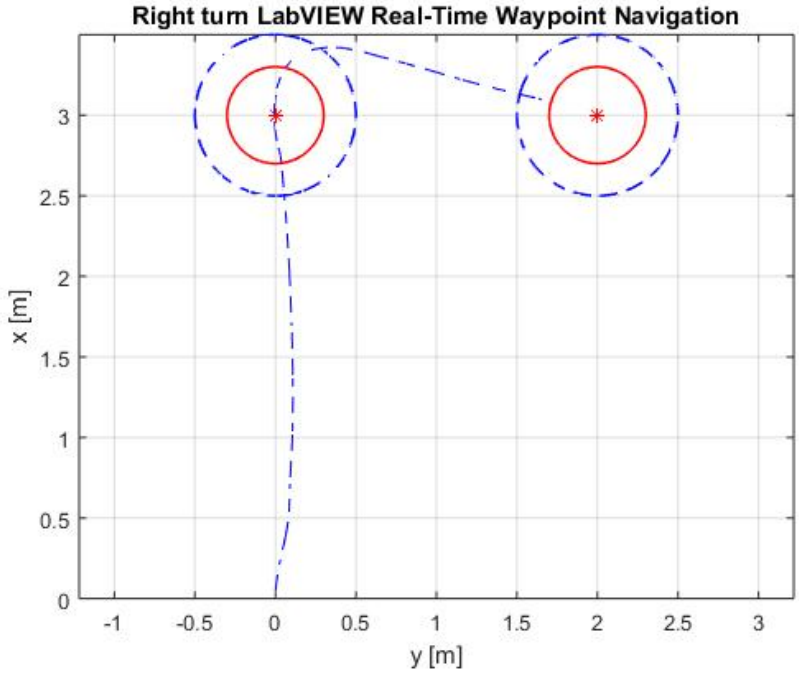

Figure 6.6: waypoint Navigation Output for Case 1

The second experiment of this case has the waypoints as specified in Table 5.2. The result of the experiment in terms of the estimated position is shown in Fig. 6.7.

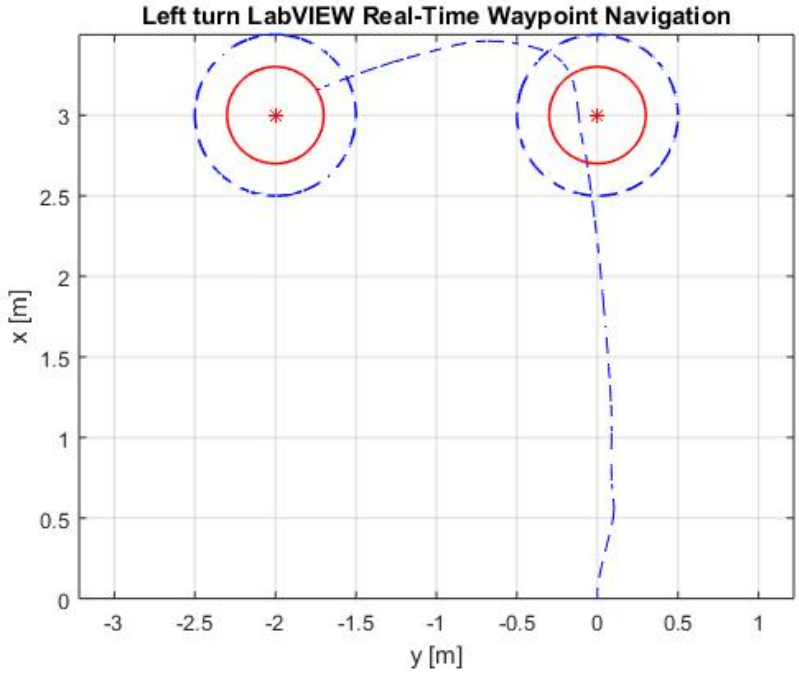

Figure 6.7: waypoint Navigation Output for Case 1b

6.2.2 Case 2

The way points is this experiment is the same as the ones given in Table 5.3. The result of the experiment in terms of the estimated position by the navigation module is shown in Fig. 6.8.

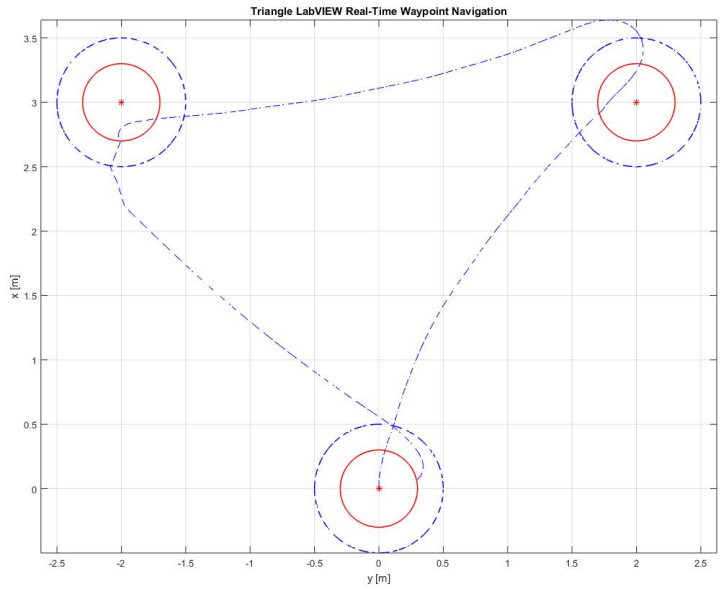

Figure 6.8: waypoint Navigation Output for Case 2

6.2.3 Case 3

The way points is this experiment is the same as the ones given in Table 5.4. The result of the experiment in terms of the estimated position by the navigation module is shown in Fig. 6.9. Note that the actual trajectory of the vehicle during the experiments are not as good as the results shown here in terms of the estimated position. This is due to the error in navigation module in estimating the position and orientation.

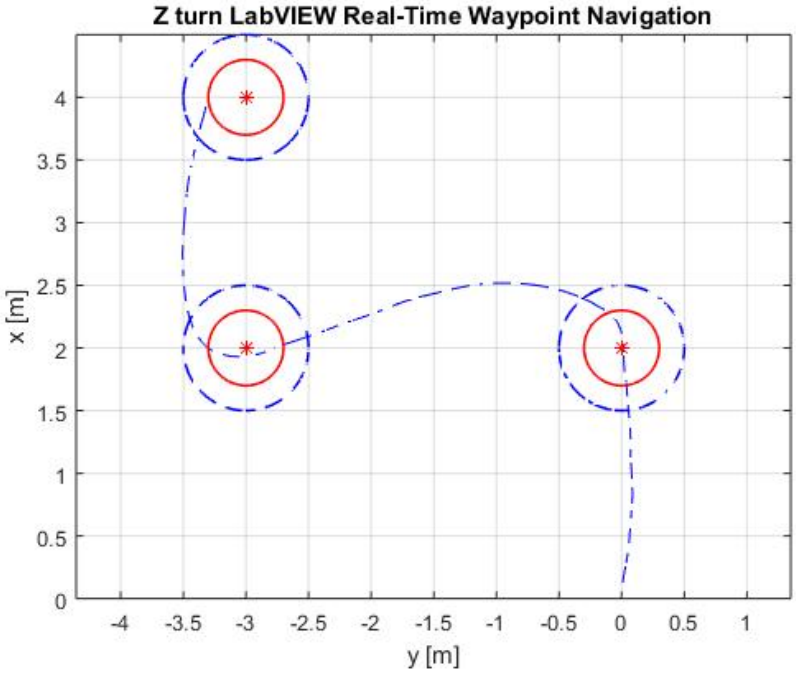

Figure 6.9: waypoint Navigation Output for Case 3

CHAPTER 7

Conclusion

LabVIEW was adequate to build a block diagram representation of a GNC (Guidance, Navigation, and Control), originally developed in MATLAB/Simulink for a UGV (Unmanned Ground Vehicle), based on encoder readings and PWM (Pulse Width Modulation) signals controlling the speed of the electric motors driving the wheels. For the development and testing of this GNC algorithm in simulation before hardware implementation, a model of the mobile platform of the UGV is also developed in LabVIEW. These two components were successfully used in simulating the closed-loop system. The real-time implementation of the GNC algorithm was successfully achieved using myRIO as the computing platform. This is successfully used for way point navigation of a skid-steered UGV. Most of the LabVIEW implementation easily follow Simulink implementation except a few cases where LabVIEW does not have specific blocks and the equivalent blocks worked slightly differently. One of the main differences in implementations in LabVIEW and Simulink is the hierarchical structure that can be set up in Simulink using "subsystem" blocks while the whole GNC algorithm was put in one single level within a "control and simulation loop" in LabVIEW. Overall, LabVIEW was as easy as Simulink in building the block diagrams while LabVIEW has a clear advantage in terms of the graphical user interface it provides in the form of a "front panel". Further, it was seamless in running a LabVIEW algorithm in real time using myRIO as the computing platform and it was very easy to interface with the sensor (encoder in this case) and actuator (electric motors in this case).

REFERENCES

[1] A. Dogan, "Unmanned vehicle system course notes," in *Modeling Simulation of UGV with Simulink, GNC UGV, Parameter Estimate*, August 2015.

[2] J. P. Lee, "Future unmanned system design for reliable military operations," September 2012.

[3] M. H. Kim, "Labview gnc," in *Development of Guidance, Navigation and Control System Software for Autonomous Unmanned Vehicle Based on LabVIEW*, June 2015.

[4] C.R.Gavhane, "Unmanned ground vehicle," in *Unmanned Ground Vehicle*, Aug 2013.

[5] "UGV Military Application," https://www.ni.com/getting-started/labview-basics/environment.

[6] "UGV Agricultural Uses," https://www.uavs.org/commercial.

[7] "UGV Law Enforcement Example," http://www.fas.org/sgp/crs/misc/IN10537.pdf.

[8] "LabVIEW national instruments," https://www.ni.com/.

[9] "Getting Started national instruments," https://www.ni.com/getting-started/labview-basics/environment.

[10] E. Doering, "Project guide," in *NI myRIO Project Essentials Guide*, April 2014.

[11] G. I. of Technology, "Mechatronics and pneumatics kit manual," January 2015.

[12] "Servo Motor wikipedia," https://en.wikipedia.org/wiki/Servomotor.

[13] "Geared Motor Information," http://www.globalspec.com/learnmore/motion_controls/motors/gearmotors.

[14] "Accelerometer wikipedia," https://en.wikipedia.org/wiki/Accelerometer.

[15] "Thumper," http://www.robotshop.com/blog/en/all-terrain-dagu-robot-wild-thumper-391.

[16] "Encoder national instruments," http://www.ni.com/tutorial/7109/en/.

[17] "H-bridge," https://en.wikipedia.org/wiki/H_bridge.

[18] "DC-to-DC Converter wikipedia," https://en.wikipedia.org/wiki/DC-to-DC_converter.

[19] G. Rajashekaraiah, "Moving obstacles detection and avoidance for unmanned ground vehicle," Master's thesis, The University Of Texas at Arlington,Texas, United States, Dec 2014.

[20] "Dead Reckoning wikipedia," https://en.wikipedia.org/wiki/Deadreckoning.

[21] "Intel NUC," http://www.intel.com/content/www/us/en/nuc/nuc-kit-nuc5pgyh.html.

[22] "Phidget wikipedia," https://en.wikipedia.org/wiki/Phidget.

BIOGRAPHICAL STATEMENT

Ishwarya Srinivasan was born in Chennai, Tamil Nadu, India in 1991. She received her Bachelor of Engineering in Instrumentation from University of Mumbai. She is currently pursuing her Master's in Aerospace Engineering from University of Texas at Arlington and her research is on Guidance,Navigation and Control algorithm Implementation in NI-LabVIEW for an Unmanned Ground Vehicle System Using myRIO.