Design and Development of Intelligent Navigation Control Systems for Autonomous Robots that Uses Neural Networks and Fuzzy Logic Techniques and Fpga For Its Implementation

Christopher James Jeanniton

Follow this and additional works at: https://digitalcommons.georgiasouthern.edu/etd

Recommended Citation

This thesis (open access) is brought to you for free and open access by the Graduate Studies, Jack N. Averitt College of at Digital Commons@Georgia Southern. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of Digital Commons@Georgia Southern. For more information, please contact digitalcommons@georgiasouthern.edu.
DESIGN AND DEVELOPMENT OF INTELLIGENT NAVIGATION CONTROL SYSTEMS FOR AUTONOMOUS ROBOTS THAT USES NEURAL NETWORKS AND FUZZY LOGIC TECHNIQUES AND FPGA FOR ITS IMPLEMENTATION

by

CHRISTOPHER JAMES JEANNITON

Under the Direction of M. Rocio Alba-Flores

ABSTRACT

This research compares the behavior of three robot navigation controllers namely: PID, Artificial Neural Networks (ANN), and Fuzzy Logic (FL), that are used to control the same autonomous mobile robot platform navigating a real unknown indoor environment that contains simple geometric-shaped static objects to reach a goal in an unspecified location. In particular, the study presents and compares the design, simulation, hardware implementation, and testing of these controllers. The first controller is a traditional linear PID controller, and the other two are intelligent non-linear controllers, one using Artificial Neural Networks and the other using Fuzzy Logic Techniques. Each controller is simulated first in MATLAB® using the Simulink Toolbox. Later the controllers are implemented using Quartus II® software and finally the hardware design of each controller is implemented and downloaded to a Field-Programmable Gate Array (FPGA) card which is mounted onto the mobile robot platform. The response of each controller was tested in the same physical testing environment using a maze that the robot should navigate avoiding obstacles and reaching the desired goal. To evaluate the controllers’ behavior each trial run is graded with a standardized rubric based on the
controllers’ ability to react to situations presented within the trial run. The results of both the MATLAB® simulation and FPGA implementation show the two intelligent controllers, ANN and FL, outperformed the PID controller. The ANN controller was marginally superior to the FL controller in overall navigation and intelligence.

INDEX WORDS: Intelligent Controller, Autonomous Robot, PID, Neural Networks, Fuzzy Logic, FPGA, Georgia Southern University
DESIGN AND DEVELOPMENT OF INTELLIGENT NAVIGATION CONTROL SYSTEMS FOR AUTONOMOUS ROBOTS THAT USES NEURAL NETWORKS AND FUZZY LOGIC TECHNIQUES AND FPGA FOR ITS IMPLEMENTATION

by

CHRISTOPHER JAMES JEANNITON

B.S., Alfred State College of Technology, 2009

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

STATESBORO, GEORGIA

2010
DESIGN AND DEVELOPMENT OF INTELLIGENT NAVIGATION CONTROL SYSTEMS FOR AUTONOMOUS ROBOTS THAT USES NEURAL NETWORKS AND FUZZY LOGIC TECHNIQUES AND FPGA FOR ITS IMPLEMENTATION

by

CHRISTOPHER JAMES JEANNITON

Major Professor: M. Rocio Alba-Flores

Committee: Fernando Rios-Gutierrez

Frank Goforth

Robert Cook

Electronic Version Approved:
December 2010
DEDICATION

To my Dad....

......If I were a rich man,

Ya ha deedle deedle, bubba bubba deedle deedle dum.

All day long I'd biddy biddy bum........

I miss you so much

March 29, 1949 - July 31, 2010
ACKNOWLEDGMENTS

I would like to thank the many people in my life that have always believed in me and pushed me to do my best. I would especially like to thank Professor Rocio Alba-Flores and Professor Fernando Rios-Gutiérrez for all the years of continuous motivation, and guidance; for without them my college career would have been different. To all my friends that took time to help in any way they could; especially Sam Parent, Mary Hoffman, Ryan Smith, and Sami Jacobs.

I would like to thank my family for all their love and encouragement. My Dad who always told me: if you're going to do something than do it with all your heart and if not don’t do it at all. My Mom for always listening to me even when I talk too much and she has no clue what I'm talking about. Dedo and Grandma, for being like a second father and mother to me, always believing that I can do anything, and for being there whenever I needed that little push to keep going. Above all, my wife, who has always supported my decision to pursue my Masters, helped edit this thesis, probably learned more than she cared to about robotics, and is very happy that I'm done.
# TABLE OF CONTENTS

ACKNOWLEDGMENTS........................................................................................................7

LIST OF TABLES..................................................................................................................11

LIST OF FIGURES................................................................................................................12

PREFACE.............................................................................................................................17

INTRODUCTION TO THE STUDY.....................................................................................18

1.1 Control Systems Terminology....................................................................................18

1.2 PID Controllers............................................................................................................22

1.3 Artificial Neural Networks........................................................................................23

1.4 Fuzzy Logic Controllers.............................................................................................26

1.5 Objective.....................................................................................................................28

REVIEW OF RELATED LITERATURE ..............................................................................30

2.1 PID Controller Literature Review..............................................................................30

2.2 Artificial Neural Network Literature Review..........................................................32

2.3 Fuzzy Logic Literature Review..................................................................................33

ROBOT HARDWARE DESIGN..........................................................................................36

3.1 Mobile Robot Platform Description..........................................................................36

3.2 Hardware Connections and Overview......................................................................37

3.1.1 Sonar Sensors .......................................................................................................38

3.1.2 RFID Tag Reader .................................................................................................42

3.1.3 Basic Stamp ..........................................................................................................43

3.1.4 FPGA Prototyping Board ....................................................................................44

3.1.5 Motor Driver .........................................................................................................45

3.1.6 Motors ..................................................................................................................46

MOBILE ROBOT PLATFORM MATHEMATICAL MODEL.............................................47
CONTROLLER DESIGN ........................................................................................................ 51

5.1 PID ................................................................................................................................ 51

5.1.1 Definitions of Proportional, Integral, and Derivative Terms ........................................ 52

5.1.2 PID Tuning .................................................................................................................... 54

5.1.3 PID MATLAB® Simulation .......................................................................................... 56

5.1.4 PID Hardware Implementation .................................................................................... 60

5.2 Artificial Neural Network .................................................................................................. 67

5.2.1 Biological Neural Networks .......................................................................................... 68

5.2.2 Artificial Neural Networks .......................................................................................... 70

5.2.3 ANN Training ................................................................................................................. 74

5.2.4 Artificial Neural Network MATLAB® Simulation .......................................................... 76

5.2.5 Artificial Neural Network Hardware Implementation ................................................... 87

5.3 Fuzzy Logic ........................................................................................................................ 92

5.3.1 Fuzzification, Rule Processing, and Defuzzification ...................................................... 93

5.3.2 Fuzzy Logic Tuning ....................................................................................................... 98

5.3.3 Fuzzy Logic MATLAB® Simulation .............................................................................. 99

5.3.4 Fuzzy Logic Hardware Implementation ....................................................................... 105

CONTROLLER TESTING AND PERFORMANCE EVALUATION ........................................ 112

6.1 Testing Environment .......................................................................................................... 112

6.2 Rubric ................................................................................................................................. 115

FINDINGS OF THE STUDY ...................................................................................................... 118

7.1 MATLAB® Simulation Results ......................................................................................... 118

7.1.1 Proportional, Integral and Derivative Controller Response ......................................... 119
LIST OF TABLES

Table 1: PING)))™ Communication Protocol (Parallax Inc., 2009) ............................... 40
Table 2: Single to Dual Output Converter ......................................................................... 50
Table 3: Gain Value Consequences .................................................................................. 55
Table 4: Ziegler-Nichols Tuning Method (National Instruments, 2006) ......................... 56
Table 5: Value equivalent to Number of Right Shifts ....................................................... 65
Table 6: Excel program to find PID Gains ......................................................................... 66
Table 7: Basic Structures of a Biologic Neuron (Hill, Wyse, & Anderson, 2004) .......... 68
Table 8: Activation Functions (Skapura, 1996) ................................................................. 72
Table 9: ANN Unit-less Value Output Matrix ..................................................................... 91
Table 10: Fuzzy Values ..................................................................................................... 94
Table 11: General Membership Function (Zhao, & Bose, 2002) .................................... 95
Table 12: Five Main Defuzzification Methods (Namazov, & Basturk, 2010) ............... 97
Table 13: Sensor Input & Corresponding Fuzzy Values per Linguistic Variable ......... 108
Table 14: Geometric Shaped Static Objects ..................................................................... 113
Table 15: Simple Shaped Static Objects ........................................................................... 113
Table 16: PID Controller’s Situational Scores ................................................................. 122
Table 17: ANN Controller’s Situational Scores ................................................................. 123
Table 18: FL Controller’s Situational Scores .................................................................... 123
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open Loop Control System</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>Closed Loop Control System</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>Typical Controller Response</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>Mobile Robot Platform</td>
<td>37</td>
</tr>
<tr>
<td>5</td>
<td>Mobile Robot Platform Hardware Flow Diagram</td>
<td>38</td>
</tr>
<tr>
<td>6</td>
<td>PING)))™ Ultrasonic Distance Sensor (Parallax Inc., 2009)</td>
<td>39</td>
</tr>
<tr>
<td>7</td>
<td>PING)))™ Communication Protocol (Parallax Inc., 2009)</td>
<td>39</td>
</tr>
<tr>
<td>8</td>
<td>Sonar Sensor Layout and Weights</td>
<td>41</td>
</tr>
<tr>
<td>9</td>
<td>RFID Card Reader (Parallax Inc., 2010)</td>
<td>42</td>
</tr>
<tr>
<td>10</td>
<td>BASIC Stamp 2e Module (Parallax Inc., 2005)</td>
<td>43</td>
</tr>
<tr>
<td>11</td>
<td>ALTERA UP3-1C12 Education Kit (Altera, 2004)</td>
<td>44</td>
</tr>
<tr>
<td>12</td>
<td>Sabertooth Dual 5A Motor Driver (Dimension Engineering, 2007)</td>
<td>45</td>
</tr>
<tr>
<td>13</td>
<td>Gear Head Motor (Lynxmotion Inc. 2010)</td>
<td>46</td>
</tr>
<tr>
<td>14</td>
<td>Electromechanical representation of DC motors</td>
<td>47</td>
</tr>
<tr>
<td>15</td>
<td>DC Motor Model Test Configuration</td>
<td>48</td>
</tr>
<tr>
<td>16</td>
<td>MATLAB Simulation Motor Step Response</td>
<td>49</td>
</tr>
<tr>
<td>17</td>
<td>Single to Dual Output Converter</td>
<td>49</td>
</tr>
<tr>
<td>18</td>
<td>PID Controller</td>
<td>51</td>
</tr>
<tr>
<td>19</td>
<td>PID MATLAB® Simulink Design</td>
<td>57</td>
</tr>
<tr>
<td>20</td>
<td>Combined Reactive Direction</td>
<td>58</td>
</tr>
<tr>
<td>21</td>
<td>PID Controller Gains</td>
<td>58</td>
</tr>
<tr>
<td>22</td>
<td>PID Simulation Results</td>
<td>60</td>
</tr>
<tr>
<td>23</td>
<td>PID controller on FPGA</td>
<td>61</td>
</tr>
<tr>
<td>24</td>
<td>Controller Input Block on FPGA</td>
<td>62</td>
</tr>
</tbody>
</table>
Figure 25: State Machine and PID controller ................................................................. 63
Figure 26: P, I, and D Terms on FPGA ........................................................................ 63
Figure 27: Proportional Term of PID on FPGA ................................................................. 64
Figure 28: Integral Term of PID on FPGA ....................................................................... 64
Figure 29: Derivative Term of PID on FPGA ................................................................... 65
Figure 30: Excel Program PID Controller Response ....................................................... 66
Figure 31: PID Controller Output Block on FPGA .............................................................. 67
Figure 32: Biologic neuron (Hill, Wyse, & Anderson, 2004) .............................................. 69
Figure 33: ANN Structure ............................................................................................... 70
Figure 34: Single Neuron ............................................................................................... 71
Figure 35: Biological and Artificial Neural Network Structural Similarities ................... 73
Figure 36: Biological and Artificial Neuron Similarities .................................................. 73
Figure 37: Artificial Neural Network MATLAB® Simulink Design ................................... 76
Figure 38: Artificial Neural Network MATLAB® Block Diagram (Left and Right) .......... 77
Figure 39: Artificial Neural Network Hidden Layers in MATLAB® Design ....................... 78
Figure 40: Left Side Untrained Artificial Neural Network MATLAB® Simulation .......... 79
Figure 41: Left Side Trained Artificial Neural Network MATLAB® Simulation ............... 80
Figure 42: Left Side Artificial Neural Network Training Simulation Results ................... 81
Figure 43: Left Side Artificial Neural Network Simulation Regression Plots .................... 82
Figure 44: Right Side Untrained Artificial Neural Network MATLAB® Simulation ........ 83
Figure 45: Right Side Trained Artificial Neural Network MATLAB® Simulation .......... 84
Figure 46: Right Side Artificial Neural Network Training Simulation Results ................ 84
Figure 47: Right Side Artificial Neural Network Simulation Regression Plots ................ 85
Figure 48: Artificial Neural Network Controller MATLAB® Simulation Results ............. 86
Figure 49: ANN Controller on FPGA ............................................................................. 88
PREFACE

This document is organized to allow the reader to examine the theoretical design and implementation aspects of three different mobile robot navigation controllers. The text is divided into eight chapters as follows. In Chapter 1 an introduction to control systems, PID, Artificial Neural Networks and Fuzzy Logic is given. In Chapter 2, Review of Related Literature, past research is discussed in the field of autonomous navigation and some controllers that aid in this task. In Chapter 3 the physical hardware aspects of the mobile robot platform are explained which includes: Sonar Sensors, RFID Tag Reader, Basic Stamp, FPGA, Motors Driver, and Motors. Chapter 4 discusses the mathematical model of the mobile robot platform that is used in testing the three different controller models. In Chapter 5 the basic terminology, Simulink simulations, and hardware implementation of the PID, Artificial Neural Networks, and Fuzzy Logic control are discussed. Chapter 6 includes the description of the environment and the rubric that is used to test and compare the three different types of navigation controllers. In Chapter 7 the results of the simulations and physical trial runs of the three different navigation controllers are evaluated and discussed. Chapter 8 includes the conclusion, recommendations, and summary of the study.
CHAPTER 1
INTRODUCTION TO THE STUDY

Navigation is one of the biggest hurdles to overcome to make a fully autonomous mobile robot. The design and implementation of an intelligent controller has been the basis of many studies that try and conquer the problem of autonomous navigation. One of the main issues in autonomous navigation is ensuring safety of the robot and the environment it travels through while also maintaining high efficiency performance levels. Giving consideration to both safety and efficiency, intelligent controllers have been researched to achieve the highest level of both factors. In this study one traditional non-intelligent controller, a PID controller, and two intelligent controllers, based on Artificial Neural Networks and Fuzzy Logic, are designed, implemented, and tested in real unknown indoor environment. The safety and navigation abilities of each controller are compared to determine the controllers’ advantages and disadvantages in different situations within the testing environment.

1.1 Control Systems Terminology
Automation is all around us. It has become so commonplace now that we do not even realize how much it impacts our day to day lives. In a single morning we can get ready while the coffee pot automatically perks your favorite brew, drive to work using cruise control to regulate your speed, tell your phone to ‘call mom’ while you walk in the automatic doors to an air conditioned workplace. All of this is accomplished by control systems.

A control system consists of interconnected components that take in a user input or set point to produce a desired output with desired performance. There are two basic
configurations of control systems: open-loop and closed-loop. A generalized overview of a straightforward open-loop control system is shown in Figure 1. An open-loop system lacks a feedback path. In other words, this simplified system has a cause and effect relationship described with the terms input and output. The input is the desired set point for which the controlled variable should reach and maintain. The process or plant is the component of the system driven by the controller. The output of the system is the ‘effect’ of the process or plant with any disturbances applied. The open-loop configuration does not compensate for any disturbances added to the system; therefore, if disturbances arise, they become part of the output. Open-loop systems are not even able to detect disturbances as they occur.

An example of an open-loop control system is a sprinkler system. The input command to the system is the timer stating how often the sprinkler waters the lawn and the length of time the water is left on. This system has no way to detect if it is raining out or if the ground is already saturated with water. The advantage of an open-loop control system is the simple and straightforward input-output relationship. The disadvantages are found in the inability to detect and compensate for disturbances to the system. These disadvantages can have detrimental consequences depending on the nature and purpose of the system.
The closed loop system attempts to overcome the disadvantages experienced by the open-loop configuration. A basic closed-loop system (Figure 2) compensates for disturbances by adding a feedback path. The input or set point of the system is set by the user to the desired value the manipulated variable should reach and maintain. The first summing junction connects the input with the output via the feedback path. Here the output value is subtracted from the input value to find the error. The comparison of these values drives the process or plant to make the necessary corrections if needed. If there is no difference between the desired input and the output, the system is already producing the desired output, and no correction is needed at that time. The sensors utilized in the feedback path continuously supply feedback to the controller in order for the system to constantly monitor for disturbances that could affect the desired output. The error of the system allows the controller to drive the process to continually reduce the difference between the set point and output.

A classic example of a closed-loop control system is a temperature controller. The system is given a desired temperature as the set point. Temperature sensors continuously monitor the temperature and provide feedback to the controller. If there is a difference between the desired set point and the current temperature, this error...
signals the controller to drive the process to correct the temperature difference. The advantages of the closed-loop feedback path control system is greater flexibility and accuracy of the system overall. The system is able to sense disturbances and allow for their correction. The disadvantages of the closed-loop system are the general increased complexity of adding the feedback loop, and also tuning the system by potentially amplifying the error in order to produce the desired output and maintain the desired performance.

The performance of a control system (Figure 3) can generally be evaluated with a few basic terms relating to the controller’s response. The controller’s response is equivalent to the rise of the manipulated variable over time. The manipulated variable should gradually rise until it reaches the set point of the system. The set point is equivalent to the desired value of the output. In many cases, the controller overshoots this set point and the response fluctuates around the set point until leveling out. The response from initial system start to when the set point is reached is called the transient response. A well designed controller will have minimal to no overshoot of the set point. The portion

**Figure 3: Typical Controller Response**
of the response in which the manipulated variable is within two percent of the desired set point is termed the **steady state response**. The margin between the set point and the steady state response is designated the **steady state error**. Since not all controllers are the same type or serve the same purpose, the design of the controller and nature of the system dictates the criteria for performance satisfaction.

### 1.2 PID Controllers

Elmer Sperry created the first Proportional-Integral-Derivative (PID) type controller in 1912 to help with ship steering (Bennett, 1979). A PID controller is referred to as a three-term controller using a proportional term, integral term, and derivative term combined in a linear algorithm. The proportional term calculates the gain based on present error. The integral term calculates the sum of all past errors. The derivative term uses the rate at which the error has been changing to predict future error. This controller also uses a feedback loop to compensate for error. The error is described as the difference between the desired set point of the system and the measured variable calculated by the P, I, and D terms. Once a PID controller is designed, a tuning process must follow in order for the controller to meet the needs of a specific system. The first theoretical study of a PID controller used for ship steering is credited to Nicholas Minorsky in 1922 (Bennett, 1979). Minorsky used a PID controller for steering the US Navy’s USS *New Mexico*. He first experimented with a PI controller and resulted in a ±2° error. When he added the derivative term, the error margin reduced to ±1/6°. This ±1/6° error is smaller than the helmsman’s human error when steering the ship manually. Minorsky achieved more in the theoretical realm of the PID controller than in physical implementation because building reliable controllers at this time was
inconsistent. By 1930, Minorsky sold his patents of the three term controller to the Bendix Aviation Company. Once more reliable controllers were manufactured and designed, PID controllers evolved into an industry standard controller today.

1.3 Artificial Neural Networks
Automatic control systems now being more precise than humans, a new wave of control theory involving artificial intelligence with robots is evolving. Robots controlled with artificial intelligence can also take the place of the human element in dangerous or life-threatening situations. Artificial Neural Networks (ANN) explores a parallel between the human nervous system and processing systems for multiple applications. ANN’s are a form of artificial intelligence controllers and are modeled after biological neural networks. The discovery of biologic neural networks dates back to the 1800’s. It was accepted that organisms were composed of cells that each had both specific structure and function; however, when it came to the nervous system, cell theory was highly debated (Hill, Wyse, & Anderson, 2004). It wasn’t until 1906 that current understanding of the nervous system structures was discovered. Santiago Ramón y Cajal theorized the neuron doctrine depicting the neuron as a structural unit, that when combined, organized the body’s nervous system (Jain, Mao, & Mohiuddin, 1996).

Before describing the functional unit of the nervous system, the hierarchal organization of the nervous system as a whole must be understood. The nervous system is structurally composed of the central nervous system (CNS) and the peripheral nervous system (PNS). The CNS includes the brain and spinal cord. The PNS contains the neurons and pathways associated with sensory inputs and motor response outputs. The input impulses travel via the sensory portion of the PNS to the CNS for higher level
interpretation. The CNS formulates a response and it is sent out to the correct location in the body via the motor portion of the PNS. In a simplistic approach of describing a biological neuron, it essentially has four main parts: dendrites, cell body, axon, and pre-synaptic terminals. The dendrites are branching structures that receive electrical impulses or signals from other neurons. The cell body structurally houses the nucleus and organelles, but functionally processes the incoming signal from the dendrites. The axon is the portion of the neuron that takes the electrical impulses or signals from the cell body to the pre-synaptic terminals. Pre-synaptic terminals form the end of the axon where it junctions with another neuron at a specialized location called a synapse. A synapse is where the axon of one neuron communicates with the dendrites of another neuron (Hill, Wyse, & Anderson, 2004). Biological neurons are arranged in network architecture with vast numbers of neurons interconnected to each other allowing for rapid communication spanning throughout all areas of the body. Biological neural networks are much higher in complexity than this representation but it is this basic structure that ANN’s model.

In 1943 McCulloch and Pitts published a paper that discussed biological neuron function in the body, as well as going a step further to design and build a primitive artificial neural network made of simple electronics (McCulloch, & Pitts, 1943). ANN's are arranged in similar network architecture as their biological model; composed of singular and simplistic neurons that communicate rapidly through a network. ANN's have artificial neurons arranged in three basic layers. An ANN starts with an input layer containing an equal number of neurons to inputs. A middle or hidden layer performs computations to create an output. The final layer, the output layer, sends the controller output to the
plant portion of the system. Each artificial neuron, excluding input neurons in the first layer, can have multiple inputs. The artificial neuron sums the weighted inputs and formulates a single output that can be propagated to multiple neurons in the next layer after processing through an activation function. By combining multitudes of singular artificial neurons into a vast processing network, ANN's are capable of complex problem solving and control.

The first practical ANN was built by Frank Rosenblatt, a neurobiologist at Cornell University, in 1958. His ANN, the Perceptron, was based on research he was doing with a fly's eye. A book titled *Perceptrons*, was published in 1969 by Marvin Minsky and Seymour Papert showing severe limitations of Rosenblatt's Perceptron. Both Minsky and Papert were influential men in the research field at the time, and their bad review of ANN's led to a drastic decrease in this topic of research (Skapura, 1996). With the bad press for the Perceptron and media of the 1970's depicting artificial intelligence (AI) as something to potentially fear, funding for research in the field of AI deteriorated as well.

A resurgence of interest in AI did not come until the 1980's with the work of John Hopfield at the California Institute of Technology. He presented a method to problem solving AI by using concepts known about the human brain. 1986 saw the creation of the back-propagation algorithm by Rumelhart, Hinton, and Williams. Back propagation is a popular method of training a feed-forward multi-layer ANN through supervised learning (Koynov, 1999). After the re-emergence of interest in ANN along with much technological advancement, ANN's are now a substantial field of research and a leading artificial intelligence controller.
1.4 Fuzzy Logic Controllers
Another controller classified in the artificial intelligence category is the Fuzzy Logic (FL) controller. FL controllers can interpret data that falls in the gray area much like a human mind can make cognitive decisions when there is no distinct answer. Fuzzy Logic is unlike many traditional logic systems in that the reasoning is approximate and not exact. It is this logic approximation also done by humans with commonsense reasoning that makes FL a form of artificial intelligence. “Fuzzy Sets” were introduced in 1965 by Lotfi Zadeh from the University of California at Berkeley (Zadeh, 1965). Zadeh formulated a mathematical analysis allowing data partial membership of a set instead of distinct membership versus non-membership categories. Fuzzy sets allow for gradual transition of data classification with permissible overlap between membership groups. This revolutionary logic system provides a way to describe systems or data that may be too complex or ill-defined for traditional analysis using precise mathematical methods. Zadeh’s ideas were not presented as a method of control, but were later applied to control theory and Fuzzy Logic controllers evolved.

The term ‘fuzzy’ almost give this controller the misnomer that it is imprecise, but in fact it is the data that is described as imprecise, vague, or ill-defined. The controller is expertly capable in interpreting this ‘fuzzy’ data to produce a straightforward output. Fuzzy Logic is represented by three parts: (1) linguistic variables in place of numerical values using natural language terms such as ‘very,’ ‘not,’ or ‘most,’ (2) fuzzy conditional statements to form IF, THEN statements, and (3) fuzzy algorithms that creates an order to the rules or instructions (Zadeh, 1990). An FL controller works through the process of receiving distinct input data, a fuzzification step using membership functions to prepare the data for use in a rule matrix, and a defuzzification step to create a crisp
output. This is accomplished by designing a membership function which combines fuzzy sets that allow distinct categories as well as functional overlap between them. This overlap corresponds to an ambiguous value belonging to more than one distinct set. The process that follows is a rule matrix defined with IF, THEN statements conjugated by AND, or OR. The fuzzy set values are processed through the defined rule matrix to create a fuzzy output. The defuzzification process uses the fuzzy value and a separate output membership function to transform the result into a crisp output to be performed by the system.

Fuzzy Logic has met great resistance since its origin in 1965. With its initial debut in the field of mathematics, Fuzzy Logic was harshly criticized for its qualitative and imprecise approach that contradicted well established quantitative and precise notions of mathematics (Zadeh, 1990). A response from Professor R.E. Kalman to one of Zadeh's presentations on Fuzzy Logic shows how hostile and unreceptive this concept was:

“Fuzzification” is a kind of scientific permissiveness; it tends to result in socially appealing slogans unaccompanied by the discipline of hard scientific work and patient observation. I must confess that I cannot conceive of “fuzzification” as a viable alternative for the scientific method. (Zadeh, 1990, p.97)

Although not well received by American researchers, Fuzzy Logic found an international home early on. Leading countries on the subject include Japan, China, and Russia. Japan has created a LIFE facility, the Laboratory of International Fuzzy Engineering, designated to Fuzzy Logic research (Zadeh, 1990). The Japanese have explored the use of Fuzzy Logic in applications ranging from train control to medical diagnosis. In
1985 Togai and Watanabe working at Bell Telephone Laboratories created the first fuzzy logic chip. 1988 saw the first fuzzy logic operated subway system in Sendai, Japan. The fuzzy logic subway outperforms human operators and standard automatic controllers in acceleration, slowing, and breaking. A fuzzy logic washing machine has also been made to adjust individual cleaning cycle depending on the dirtiness of the clothes. An optical sensor detects the clarity of the water and adjusts the cycle time to more efficiently and completely clean the clothes. Canon H800 hand held camcorders autofocus using fuzzy rules. General Motors has come out with a fuzzy transmission for a line of Saturn cars. A complex fuzzy system in operation is a model helicopter by Sugeno at the Tokyo Institute of Technology. The fuzzy logic control of the helicopter allows the vehicle to hover in place; a difficult task for human pilots (Kosko and Isaka, 1993). Many of these advances in fuzzy logic controls are successes of Japan and China leaving the United States and European nations lagging in production and research in this field.

1.5 Objective
Navigation of autonomous mobile robots is presently an important field of research because of the recent increase in security and reconnaissance needs. Questions that arise while conceptually modeling, designing, and implementing a mobile robot include what type of controller to use, what hardware or software to use, compatibility of components, size and speed of robot base, etc. The questions and variables are endless. Due to their complexity, behaviors and tasks are narrowed for the specific application the mobile robot is created for and based on the characteristics needed. Another consideration is the environment the mobile robot is responsible for
autonomously navigating. Is it a known environment or unknown environment? What sort of obstacles will the robot potentially encounter? Are there any environmental conditions such as terrain or changing weather patterns to deal with?

Of key importance when undergoing research in robot navigation is whether the research is concluded after software simulation, or if it is pertinent to develop the physical implementation of the design. This thesis revolves around the comparison of PID, Artificial Neural Network (ANN), and Fuzzy Logic (FL) controllers. Both simulation models in MATLAB® and physical hardware implementation on Field Programmable Gate Array (FPGA) are designed and developed to be compared within this research. These controllers will utilize the same mechanical platform for testing the navigation of the mobile robot in an indoor unknown environment.
CHAPTER 2

REVIEW OF RELATED LITERATURE

2.1 PID Controller Literature Review

Traditional PID controllers, such like the initial controllers created by Elmer Sperry and Nicholas Minorsky (Section 1.2), have been heavily researched and implemented in various applications. Research with PID controllers has shifted focus from designing the most efficient controller toward designing the most efficient method of implementation. Designing implementation schemes that allow faster processing capabilities is the new motivation for working with this traditional industry standard controller.

The basic design of a PID controller is rarely disputed; however, the most efficient method of implementing this controller has led to the research performed by Gupta, Khare, and Singh (2009). This group set about to design a digital PID controller designed for Field-Programmable Gate Array (FPGA) implementation. Their research focuses around creating a multiplierless PID controller for simulated hardware implementation on the FPGA card. MATLAB® and Simulink are used as the simulation software.

The concept behind using an FPGA device is to gain faster processing capabilities than can be accomplished with software based PID controllers. The use of the FPGA in addition to eliminating the large computations is tested for increased speed of operation. The multiplierless PID is achieved by the use of a Look-up table stored within ROM memory on the FPGA device. The look up table is generated and used for computational efficiency and replaces the actual computations of the controller to save
processing time. This study provides sixteen possible input combinations that reference a specific address on the look up table. The output values are calculated prior to simulation and are stored in ROM under the corresponding address location. The results of Gupta et al, (2009) show the multiplier-less PID controller on simulated FPGA provide improvements in rise and settling time.

FPGA hardware implementation for controllers has transitioned into a standard implementation option. Once the decision to use an FPGA device is reached, the next step is determining whether to use parallel or serial architecture. Zhao, Kim, Larson, and Voyles (2005) compared parallel and serial architectures for PID implementation on FPGA. The two designs are compared in FPGA area, speed of processing, and power consumption. The parallel design allows an input to propagate through all terms of the PID controller simultaneously to quickly produce an output value. Within each term, P, I, and D, the mathematical functions are needed. The authors used four adders and three multipliers within the parallel architecture for the PID controller on the FPGA. A serial structure allows an input to enter the FPGA for processing, but only one term of the PID controller can process the input at a time. This design only requires a single adder and a single multiplier. A multiplexor is used to switch between P, I, and D terms. They concluded the parallel design requires more hardware area for implementation, but provides an advantage in processing speed. The serial architecture gives a space advantage of 24 percent less hardware area used, but exhibits a disadvantage in speed since more clock pulses are needed to execute serial design. Both structures underwent power analysis, but minimal differences were noted between the parallel and serial architectures.
2.2 Artificial Neural Network Literature Review

Artificial Neural Networks (ANNs) were a revolutionary concept in 1943 when McCulloch and Pitts first implemented this controller (Section 1.3). Without the use of computers, ANNs were formulated purely with mathematical models. Advancements in computer science have led to easier simulation techniques allowing for results to be generated more quickly. The process from conceptual design to simulation of a working controller can be completed in an efficient time frame due to the evolution of simulation software. Now that ANNs are more accessible through computer simulation, more research applications and advanced controller designs are being studied.

Singh and Parhi (2009) designed simulation research around a four layer neural network controller to navigate a crowded unknown environment. The goal was to reach a specified target while maintaining collision free movements around static and dynamic obstacles. The simulations were completed on ROBNAV software. The designed neural network contains 4 input neurons in the input layer, 10 neurons in the first hidden layer, 3 neurons in the second hidden layer, and a single neuron in the output layer. The proposed model of the mobile robot includes an array of sensors for obstacle detection. These sensors form the four inputs: left sensor obstacle distance, right sensor obstacle distance, front sensor obstacle distance, and target angle. The simulated output is the steering angle to avoid obstacle collision. While training in simulation, the network was provided with 200 patterns of varying scenarios such as corridors, rooms, walls, and intersections. The final result of the simulation provided a trained proposed neural network. The simulation results show this controller was capable of path optimization, obstacle avoidance, smooth navigation through a crowded simulation environment, and target location.
A reinforced learning model of an artificial neural network controller for implementation on a Khepera robot kit was researched by Rios-Gutiérrez (2000). Khepera simulation software was utilized for simulation and training of this controller. The mobile robot used eight infrared sensors for inputs to the network. The sensor values are pre-processed before entering the network. The sensor values are converted to binary inputs for edge, wall, and hole detection. The binary numbers are created by applying a threshold and other pre-calculations to the sensor values. The neural network takes in these binary inputs and transforms them to heading directions of left, straight, or right. The overall design consists of two on-board neural networks. The first transforms inputs to outputs. The second provides critical evaluations of the first network's actions to create a system of reward signals for reinforcement for the purpose of re-weighting connections. The second network is an on-board trainer to the system. After 50,000 training trials were completed in simulation, the network achieved 95 percent efficiency in dealing with proposed random environments.

2.3 Fuzzy Logic Literature Review
Zadeh’s contribution to mathematical logic models led to a wave of research based around his concept of ‘fuzzy sets’ (Section 1.4) (Zadeh, 1965). An area of research being explored with his notion of fuzzy sets is the design and implementation of an intelligent control systems termed Fuzzy Logic (FL). With only a short time span since the concept’s introduction, explorations into control design have been an area of heavy interest. Current research with this controller revolves around experimenting with different applications as well as speeding the design of implementation.
Peri and Simon (2005) designed an autonomous wheeled wall-following robot using ultrasonic sensors for inputs to traverse a known indoor environment for an IEEE competition. They designed the FL controller for this robot using MATLAB® and Simulink for simulation and utilization of a PIC microcontroller for implementation. The MATLAB® simulation model was developed using the kinematics equations for this differential drive robot. The FL controller has two inputs: the position error, and the angle error. These values are gathered from the three mounted ultrasonic sensors on the front and two sides of the robot base. The controller employs the use of 18 rules to process the fuzzy data. Through defuzzification, two outputs are generated for position correction and angle correction sent to the servo motors. A hurdle to overcome by Peri and Simon while implementing the controller was a processing time issue. The system clock of the microcontroller was 4MHz, which translated into a 0.4 second processing time from fuzzification, rule processing, and defuzzification. To bypass this issue, the pair generated a look up table to load onto the microcontroller in place of the FL controller. The results included an efficiently performing controller able to reach a referenced wall distance from any angle starting position.

Ono, Uchiyama, and Potter (2004) designed and created a controller for testing a mobile robot base for corridor navigation. The research was done in hopes of future expansion in intelligent wheelchair implementation. They used four agents responsible for different aspects of the control system such as sensor handling, machine vision, collision avoidance using FL, and locomotion. The focus of the review of the research is on the FL controller aspect. FL is used to detect and avoid collisions with obstacles within the corridor situation. The mobile robot platform used was a purchased ER1
Personal Robot System kit with additions of infrared sensors and a laptop computer. The infrared sensors are spaced in order to provide 360 degree coverage. The fuzzy collision avoidance portion of the controller utilizes one input fuzzy set for the sensor inputs and three output fuzzy sets. Seventeen rules are employed by the rule processing section of the FL controller. The outputs include distance, velocity, and turn angle. The distance output membership function (MF) determines if the output should move the robot forward or backward. The velocity output MF uses the linguistic variables slow, medium, and fast to generate an appropriate speed. The turn angle output MF divides the total angle, pre-set to 60 degrees, into sections of positive left, negative left, positive center, negative center, positive right, and negative right. The results were a mobile robot that avoided collisions with both obstacles and walls in a real indoor environment. The FL controller produced at times an unwanted zig-zag path pattern. It was also determined the infrared sensors were negatively affected by the ambient indoor lighting.
CHAPTER 3

ROBOT HARDWARE DESIGN

The mobile robot platform that was designed for this study uses simple hardware elements. This keeps the complexity of the overall physical robot platform to a minimum, and the concentration weighted more on the controllers’ designs.

3.1 Mobile Robot Platform Description

The implemented mobile robot platform (Figure 4) consists of a two level structure made of two 6.35mm thick polypropylene discs with a diameter of 30.48 cm. Four aluminum standoffs measuring 13 cm separate the two polypropylene discs. The platform contains eight parallax PING))™ sonar sensors arranged every 45 degrees around the circumference of the lower level disk. A Radio Frequency Identification (RFID) Tag Reader and Basic Stamp microcontroller (I/O Processing Unit) are mounted to the top side of the lower level disc. Attached to the underside of the upper level disc is a Field Programmable Gate Array (FPGA) card. A two channel motor driver is fixed to the underside of the lower level disc. Two DC motors are attached 9 cm away from the center point to drive two 7.3 cm diameter rubber wheels with a width of 10 mm. The two caster wheels are located 90 degrees from the rubber wheels for stability and smooth turning. All of the electronics are powered by two battery packs (7.2VDC, 3300 mAh each). See APPENDIX A for visuals of the mobile robot platform.
3.2 Hardware Connections and Overview

This section concentrates on the electronic hardware elements of the mobile robot platform. Figure 5 is created as a visual representation of how these elements are interconnected. The FPGA Card is the main processing block, since it is used to implement the different controllers. It receives and transmits signals to and from the I/O processing unit that is implemented using a BASIC Stamp 2e. The Basic Stamp will control the collection of data from the eight parallax PING)))™ sonar sensors and RFID Tag Reader. It also communicates the required motor speed signals to the Sabertooth Dual 5A motor driver. The motor driver generates the corresponding voltage level to each motor, needed to change the direction the mobile robot platform is traveling.

Each controller, PID, ANN, and FL is individually implemented onto this mobile robot platform. This allows for consistency and focuses the research comparisons on navigational abilities of each controller.
3.1.1 Sonar Sensors

The sonar sensors (Figure 6) use ultrasonic sound waves to measure the distance the sensor is from an object. The sensors have a three pin header used to supply the
Figure 6: PING))) ™ Ultrasonic Distance Sensor (Parallax Inc., 2009)

5VDC (Vdd), ground (Vss), and signal pin (SIG Pin). The signal pin serves both as an input and an output function. The input function is a 2μs to 5μs activation pulse needed to have the sensor start measuring the distance to an object (Figure 7 and Table 1). The sensor is capable of detecting objects from 2cm to 3m away. The input function is sent to the sonar sensors using the signal pin. The sensor works by transmitting (TX) a 200μs at 40 kHz burst of sound waves and then returns the time it takes to receive the burst’s echo through the signal pin to the host device (Parallax Inc., 2009). A host device can be, but not limited to, a microcontroller, computer, or hardware controller. In this case a Basic Stamp 2e microcontroller is the sonar sensors’ host device.

Figure 7: PING))) ™ Communication Protocol (Parallax Inc., 2009)
The minimum time that can be returned to the host device for an object 2cm away is 115μs and the maximum time returned for an object 3m is 18.5ms. The output signal the host device receives is easily converted into the distance the object(s) is from the sensors in centimeters. The equation used to perform the conversion is (Parallax Inc., 2009):

\[
1cm = \frac{\text{Time in } \mu\text{s}}{29.033\mu\text{s/cm}} \times 0.5
\]

(Equation 3.1)

The Time in microseconds is equivalent to the length of time it takes the sonar sensor to receive the burst of ultrasonic sound from initial activation to when the burst’s echo is returned. The conversion factor of 29.033μs/cm is the length of time it takes the ultrasonic sound burst to travel one centimeter. The value is then multiplied by 0.5 to divide the distance in half because the time in microseconds covers both the time the burst is sent out to an object and then echoed back to the sonar sensor.

There are eight sonar sensors used to detect objects that are around the mobile robot platform as it moves throughout the environment. The sensors are located every 45 degrees around the lower level of the robot platform at a height of 10cm from the ground. This layout puts four sensors on each side of the platform (Figure 8). The arrangement allows for 360 degree coverage for object detection.
Figure 8: Sonar Sensor Layout and Weights

The eight sensors’ values for object distance away are summed together through mathematical equations to produce two values that are used as the input to each of the three different controllers. The sonar sensors on each half of the mobile robot platform are fused and weighted to minimize the number of inputs to each controller. This allows one sonar sensor input value from the left side and one sonar sensor input value from the right side. The two input values are calculated using Equation 3.2 and Equation 3.3. Both equations weight the sensors’ value then sums the values together to come up with one value per side.

\[
Left \text{ Side} = 50\% \times \left( \frac{\text{Time in } \mu \text{s from } S1}{29.033 \mu \text{s/cm}} \times 0.5 \right) + 25\% \times \left( \frac{\text{Time in } \mu \text{s from } S2}{29.033 \mu \text{s/cm}} \times 0.5 \right) \\
+ 15\% \times \left( \frac{\text{Time in } \mu \text{s from } S3}{29.033 \mu \text{s/cm}} \times 0.5 \right) + 10\% \times \left( \frac{\text{Time in } \mu \text{s from } S4}{29.033 \mu \text{s/cm}} \times 0.5 \right)
\]

(Equation 3.2)
$$Right\ Side = 50\% \times \left( \frac{\text{Time in } \mu\text{s from } S5}{29.033\mu\text{s/cm}} \times 0.5 \right) + 25\% \times \left( \frac{\text{Time in } \mu\text{s from } S6}{29.033\mu\text{s/cm}} \times 0.5 \right)$$

$$+ 15\% \times \left( \frac{\text{Time in } \mu\text{s from } S7}{29.033\mu\text{s/cm}} \times 0.5 \right) + 10\% \times \left( \frac{\text{Time in } \mu\text{s from } S8}{29.033\mu\text{s/cm}} \times 0.5 \right)$$

(Equation 3.3)

The sonar sensor values are weighted in order to prioritize the readings. The mobile robot platform is mainly moving in a forward direction. Due to this direction of movement the sensor values coming from the front of the robot platform take precedence over the sensor values toward the back. There is a gradual decrease in weight of the sensor values importance from the front to the back of the platform. The importance of each sonar sensor is translated by the weight placed on each value (Figure 8).

3.1.2 RFID Tag Reader

The RFID Tag Reader (Figure 9) can identify passive RFID tags. The reader has a four pin header: 5VDC (Vdd), ground (Vss), enable, and signal.

![RFID Card Reader](Parallax Inc., 2010)

Figure 9: RFID Card Reader (Parallax Inc., 2010)

The enable pin has to be activated with a logic low signal from the host device in order for the reader to identify RFID tags. The signal pin allows the RFID Tag reader to read
a passive RFID tag’s identification number and communicate this ID number to the host device. The maximum distance the RFID Tag reader can read a tag from is 10cm.

In this study the goal of each controller’s run through the varying environment is to find the location of the passive RFID Tag. The RFID Tag Reader is continuously scanning for the passive RFID Tags response as the mobile robot platform traverses through the environment. It communicates to the Basic Stamp 2e microcontroller which decides what action to take based on the incoming information.

### 3.1.3 Basic Stamp

The Basic Stamp 2e microcontroller (Figure 10) is the main processing unit used for interfacing the inputs and outputs. This unit has 16 independent input/output pins and two additional pins dedicated to serial communications. The Basic Stamp 2e is capable of handling approximately 4,000 instructions before the 16K bytes EEPROM memory is full. The program is processed with a clock speed of 20MHz (Parallax Inc., 2005). The programming language used to implement instructions is the PBasic programming code developed by the Parallax Company and is basically a sublet of the Basic Programming Language.

![Figure 10: BASIC Stamp 2e Module](Parallax Inc., 2005)

The Basic Stamp 2e is used to input and output data in order to reduce the complexity of the FPGA implemented controllers. The Basic Stamp 2e is the host device for the
sonar sensors and RFID Tag Reader. This microcontroller receives all eight sonar sensor values and performs the mathematical operation that fuses their values (Equation 3.2 and Equation 3.3) to produce the output(s) sent as pulses (ms) to the FPGA card. The Basic Stamp 2e receives data from the FPGA card used for motor control signals that are then sent to the Sabertooth Dual motor driver to control the speed of the motors. The Basic Stamp 2e also receives data from the RFID Tag Reader to determine if a passive tag has been located. If a tag has been found, the Basic Stamp sends a signal to the motor driver to halt movement of the mobile robot platform. This indicates the mobile robot platform has reached its goal.

3.1.4 FPGA Prototyping Board
The FPGA board is a programmable hardware system that a user can configure to meet the needs of a design. The user can perform any number of logical functions by configuring logic blocks and interconnects. Wiring the blocks together, the user can produce designs to perform complex operations. The FPGA Card used in this study is an ALTERA UP3-1C12 Education Kit that utilizes the Cyclone EP1C6Q240C8 (Figure 11).

![Figure 11: ALTERA UP3-1C12 Education Kit (Altera, 2004)](image-url)
The FPGA card has 63 general purpose pins that can be configured as either inputs or outputs. This model is capable of allowing up to 12,060 logic elements to be implemented into a design (Altera, 2004).

The three controllers, PID, Artificial Neural Networks, and Fuzzy Logic, are implemented independently on this FPGA Card. The configuration of each controller is explored in more detail in Chapter 5.

3.1.5 Motor Driver
The Sabertooth Dual 5A Motor Driver (Figure 12) allows for the control of two DC motors. The output voltage range of the motor driver is 6-20VDC. The motor driver also allows for a continuous current of up to 5A per output channel.

![Figure 12: Sabertooth Dual 5A Motor Driver](Dimension Engineering, 2007)

There are two inputs to this device: signal one (S1) and signal two (S2). This driver has two different methods of controlling DC motors: mixed mode and independent mode. Mixed mode controls the motors through differential drive capabilities. This allows the control of forward or back motion on S1 and the steering on S2. Independent mode controls each motor’s forward and backward rotational speed through only one of the signal inputs. This means one motor is controlled through S1 and the second motor is controlled through S2. The motor driver also has different modes for the signal inputs: analog, R/C, or serial. Analog input mode uses a voltage from 0VDC-5VDC, R/C input
mode uses pulses from 1ms to 2ms, and serial input mode uses TTL level RS-232 serial data to control the two DC motors (Dimension Engineering, 2007).

In this case, the Sabertooth is used in the independent mode, allowing for independent control of each motor. Signal one is used to control the left motor and signal two controls the right motor by the Basic Stamp 2e from motor control signals from the FPGA card. The signal mode used is the R/C mode with microcontroller capability. By using the microcontroller capability, a continuous signal is not necessary to keep performing an action. Once the initial signal is sent, the motor driver will reproduce the signal until a different signal is received.

3.1.6 Motors
The mobile robot platform uses two DC gear head motors for a means of movement (Figure 13). The maximum allowable voltage is 12VDC per motor. The motors have a gear ratio of 30:1 with a maximum 200 RPM on the 6mm output shaft that turns the wheels (Lynxmotion Inc., 2010). There is another shaft that extends out the back of the motor that is un-gereared and allows for the attachment of optical encoders.

![Figure 13: Gear Head Motor](Lynxmotion Inc. 2010)
CHAPTER 4

MOBILE ROBOT PLATFORM MATHEMATICAL MODEL

Modeling of mobile robot platform is done using MATLAB® Simulink toolbox. By modeling the DC motors in this software, the plant portion of the control system is produced. MATLAB® Simulink toolbox is also used to model the PID, Artificial Neural Network, and Fuzzy Logic controllers (Chapter 5). By having both the plant and control portions of the overall control system modeled in the same software, a computer simulation is used to test the response of each controller.

4.1 Model of Plant for Mobile Robot Platform

The mobile robot platform has two plants which are modeled after the two DC motors. The models are derived from the electromechanical representation of the DC motors (Figure 14). The DC motors that are used for the mobile platform have an armature resistance of 1Ω, armature inductance of 500mH, and a motor inertia of 0.01Kg-m² (Lynxmotion Inc., 2010). The gear ratio is 30:1. The voltage source has a maximum voltage of 12VDC, and the output is measured in RPMs.

Figure 14: Electromechanical representation of DC motors
In this model, the input can be 0-12VDC which is connected to a Simulink converter that converts the input into the correct unit for the next portion of the model. The electrical portion of the model is shown with the resistor, inductor, and ground. A segway into the mechanical section is made as the model transitions into the electromechanical converter. The mechanical portion of the model is shown with the inertia and gear box that simulates the motor shaft to the output shaft which has the 30:1 ratio. The ideal rotational motion sensor measures the shaft rotation in RPMs, which can then be converted to the unit necessary for the output.

After the complete model is designed (Figure 14), a test is configured to simulate potential outputs (Figure 15). The 6V step input stays at 0VDC for one second, then steps up to 6VDC at one second which is equivalent to half the maximum input value.

![Diagram](image)

**Figure 15: DC Motor Model Test Configuration**

This input is processed through the DC motor model and a measurable output is graphed by the motor response scope. The scope plots the output response on an X, Y graph in RPM versus time (sec) (Figure 16). The expected response is for the motor to reach 100 RPM with an input of 6VDC. The 6VDC input corresponds to half the maximum voltage, so it is expected that half the maximum RPMs (100) would be reached as the steady state. The graph also shows the response time to reach the
steady state. This simulation produced a settling time to the steady state of approximately five seconds.

![Motor Response Graph](image)

**Figure 16: MATLAB Simulation Motor Step Response**

### 4.2 Single to Dual Output Converter

This converter (Figure 17) is a subsystem portion of the controller. It functions to take the controller output’s single unit-less denomination of 0-100 and convert it into two complemented voltage outputs between -6VDC to 6VDC. These converted outputs are used by the motor driver to run the left and right DC motors. The 0-100 range is a scale used to distinguish how quickly the mobile robot platform must turn in either direction to avoid collision with an object. The range is divided into two equal selections. The first
selection is from 0-50; this range controls the left hand turning of the mobile robot platform. The closer a value is to zero the quicker the robot must left turn. The second selection is from 50-100; this range controls the right hand turning of the mobile robot platform. The closer a value is to 100 the quicker the robot must right turn. If a value is equal to 50 the mobile robot platform will move in a straight forward movement. Using the controller’s output, Table 2 shows how the converter calculates the appropriate voltages to supply the DC motors. Logic operations and mathematical equations are used to accomplish this conversion.

<table>
<thead>
<tr>
<th>Unit-less Scalar Value (x)</th>
<th>Right Motor Voltage (VDC)</th>
<th>Left Motor Voltage (VDC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>x &gt; 90</td>
<td>-6</td>
<td>6</td>
</tr>
<tr>
<td>50 &lt; x ≤ 90</td>
<td>( VDC = -0.15 \times x + 13.5 )</td>
<td>6</td>
</tr>
<tr>
<td>x = 50</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>10 ≤ x &lt; 50</td>
<td>6</td>
<td>( VDC = 0.15 \times x - 1.5 )</td>
</tr>
<tr>
<td>x &lt; 10</td>
<td>6</td>
<td>-6</td>
</tr>
</tbody>
</table>

Table 2: Single to Dual Output Converter
5.1 PID

Proportional-Integral-Derivative (PID) controllers have become the conventional controllers of industry. They are capable of controlling many types of systems to meet specific needs while giving a strong performance. Their popularity can be attributed to the straightforward manner they operate as well as their wide range of functional ability. These controllers can control variables such as temperature, pressure, and speed. As the name indicates, a PID controller uses an algorithm consisting of three terms: proportional, integral, and derivative. These components are combined in a closed loop system (Figure 18) to create a desired output response. A PID controller functions to regulate an output based on the error value processed by using the feedback that the closed loop configuration provides. The error is calculated from the established set point and the output of the PID algorithm once processed through the plant. This

![Figure 18: PID Controller](image)
controller attempts to minimize the error; however, a “tuning” stage is needed for optimal response. Tuning is done by altering the three terms until the most favorable output response is accomplished. Manual (trial and error) and Ziegler-Nichols methods of tuning are commonly used for small scale products. PID tuning software is now available for large scale industrial purposes.

### 5.1.1 Definitions of Proportional, Integral, and Derivative Terms

A PID controller uses a linear algorithm (Equation 5.1) to calculate the controller output.

\[
Controller \ Output = P \ out + I \ out + D \ out
\]

\hspace{1cm} (Equation 5.1)

The proportional term is responsible for the majority of the output change and uses the difference between the set point and the process variable. The proportional gain, \( K_p \), is directly proportional to the speed of the response of the system. Modifying the \( K_p \) modifies the behavior of the controller. \( K_p \) is multiplied by the current error to produce the proportional response of the output (Equation 5.2). The greater the value of the proportional gain (\( K_p \)) the faster the response to the current error (Bräunl, 2003). If the proportional gain is set too high, then undesired oscillation of the process variable will result. If increased above this point, it causes the system to become unstable. On the other hand, if \( K_p \) is set too low, the controller response may be too small to create an efficient response to the disturbance or error. The preferred value for \( K_p \) leads to a fast controller response to the current error, but does not cause the system to overshoot the set point by a large margin or cause the system to oscillate out of control (National Instruments, 2006). A purely proportional term controller (lacking outside disturbances) will not settle at the given set point, but instead a steady-state error results (Bräunl,
This error can be corrected two ways. Either the user can set the set point above the true desired response value to offset the settling of a P controller, or an integral term can be added to correct the steady state error.

\[ P_{out} = Kp \cdot e(t) \]  

\textbf{(Equation 5.2)}

The integral term is responsible for summing the past errors over time. Both the magnitude and the duration of the past errors are considered when determining the sum of past errors. The constant \( K_i \) is multiplied by the accumulated error to calculate the integral term of the controller (Equation 5.3). The correct value for \( K_i \) is determined during tuning which is discussed in Section 5.1.2. The calculated integral term is then added to the P term for the effect of eliminating the steady state error. The steady state response is reached later than in a pure P controller, but again the steady state error has been diminished to zero (Bräunl, 2003). Even a system experiencing small errors will see the integral term slowly increase in order to eliminate error all together. The drawback of an integral term is that is uses past errors to diminish steady state error and this can cause the controller to overshoot the set point in the present (National Instruments, 2006).

\[ I_{out} = Ki \cdot \int_0^t e(t)\,dt \]  

\textbf{(Equation 5.3)}

The third term, the derivative term, is added to the PI controller to compensate for the overshoot of the set point in the present by the I term. The D term works change the rate of the response of the P controller, and it is most noticeable near the set point of the system. The derivative term is calculated by taking the last error minus the current
error and multiplying by the constant $K_d$ (Equation 5.4). The correct value for $K_d$ is determined in the tuning process discussed in Section 5.1.2. Most PID controllers utilize a small D term because it has such a strong impact on the overall response. A small D term is sufficient enough to have the proper effect. The higher the $K_d$ is the stronger the reaction to the error term will be. A large D term will cause the system to become unstable especially is there is a large amount of noise in the error term (National Instruments, 2006).

$$D_{out} = Kd \times \frac{d}{dt}e(t)$$

(Equation 5.4)

The use of the three term controller, PID, allows for a system to generate an output response considering the error occurring. By combining these weighted terms, the controller quickly responds to an error with little steady-state error and minimal overshoot of the set point. These are the characteristics that have led this controller to become the industry standard it is today. The complete linear algorithm for the PID controller is as follows:

$$Controller\ Output = Kp \times e(t) + Ki \times \int_0^t e(t)dt + Kd \times \frac{d}{dt}e(t)$$

(Equation 5.5)

5.1.2 PID Tuning

In order to obtain the optimal response from the control loop, the gains for P, I, and D must be set in a tuning process. The basic requirement for all control systems is stability. If the gains are chosen incorrectly, it will lead to instability of the system. Table 3 details the consequences of incorrect gain values.
<table>
<thead>
<tr>
<th>Gain</th>
<th>Value too large</th>
<th>Value too small</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_p$</td>
<td>Fast response could lead to larger error and potential instability if overshoot is too great</td>
<td>Slow response or insufficient response to a disturbance</td>
</tr>
<tr>
<td>$K_i$</td>
<td>Steady-state error eliminated quickly, but greater potential for overshoot of set point</td>
<td>Longer time to eliminate steady-state error, but will have less overshoot</td>
</tr>
<tr>
<td>$K_d$</td>
<td>Overshoot compensated for quickly, but leads to instability due to noise in feedback signal</td>
<td>Overshoot not dealt with in a timely manner, but this term alone will not lead to instability due to noise amplification</td>
</tr>
</tbody>
</table>

Table 3: Gain Value Consequences

As mentioned previously, there are multiple methods to tune a PID controller. The first method is a trial and error manual method done by a person with background knowledge of the significance of each gain. The first step is to find the correct $K_p$ since the bulk of the response is determined by the P term. To do this, $K_i$ and $K_d$ are set to zero and $K_p$ is increased until the output oscillates around the referenced set point. Once the system achieves adequate response time by adjusting $K_p$, $K_i$ is adjusted to stop the oscillating effect. This value is fine-tuned to minimize the offset in a timely manner, but will increase the overshoot of the set point. Once $K_p$ and $K_i$ have been set to allow the system to respond in the desired time with minimal steady-state error, $K_d$ is slowly increased to achieve a system that reaches and maintains the set point within an acceptable time after a disturbance. Increasing $K_d$ decreases the overshoot of the set point and allows quick response of the system accompanied by stability. $K_d$ usually remains a small value as to not make the system sensitive to noise (National Instruments, 2006). Although the principles of this tuning process seem simple to
describe, tuning can become a lengthy process to ensure the controller satisfies the needs of the system.

Another popular method of tuning this controller is the Ziegler-Nichols method. The first step parallels the manual trial and error method. $K_i$ and $K_d$ are set to zero, and $K_p$ is increased until the loop oscillates around the set point. At this point, the ultimate gain ($K_u$) and oscillation period $P_u$ are noted. These values are then used to tune the gain parameters of the 3 terms using the following table:

<table>
<thead>
<tr>
<th>Controller</th>
<th>$K_p$</th>
<th>$K_i$</th>
<th>$K_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>0.50 $K_u$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PI</td>
<td>0.45 $K_u$</td>
<td>$P_u/1.2$</td>
<td>-</td>
</tr>
<tr>
<td>PID</td>
<td>0.60 $K_u$</td>
<td>0.50 $P_u$</td>
<td>$P_u/8$</td>
</tr>
</tbody>
</table>

Table 4: Ziegler-Nichols Tuning Method (National Instruments, 2006).

A third method of tuning a PID controller is by using PID tuning software. This method is popularized by industry to obtain consistency among systems. A person using either the manual or Ziegler-Nichols method takes time to obtain the optimal responses, and to industry, time equals money. The software provides a faster and more consistent method of tuning these controllers. Many software packages are available that tune according to certain performance criteria required by a specific system depending on its design use.

### 5.1.3 PID MATLAB® Simulation

MATLAB® Simulink Toolbox creates a simulated environment used to design, build, and test a controller’s performance on a system. Within this research, the three compared controllers are simulated using this software. The first controller simulated is the PID.
Within the simulation, the DC motors are modeled and used in conjunction with the singular to dual output converter (Chapter 4). Figure 19 is the overview of the complete PID controller design including the systems used in the mobile robot platform. Each controller design incorporates two plants which are the simulated left DC motor and the right DC motor of the mobile robot platform.

![PID MATLAB® Simulink Design](image)

**Figure 19: PID MATLAB® Simulink Design**

In this PID controller a closed loop configuration is used to provide feedback for the system. The set point of this specific system is 30cm. This set point is chosen as the most favorable distance for the mobile robot platform for navigation and obstacle avoidance. With this set point, the simulation requires the robot platform to react to a sensor value less than or equal to 30cm. A random number generator is used to simulate random sensor values between 0-30cm for both the left and right sides. The sensor values are the feedback for this system. The summing blocks subtract the current random sensor value from the set point to generate an error. The error value is propagated to the unit-less value converter. This converter takes the left side error and converts it into a unit-less value between 0-50. The right side error is converted into a unit-less value of 50-100. The two unit-less values are then averaged together by the
adder and 0.5 gain to produce a combined value. This new value represents the combined reactive direction (Figure 20) in which the mobile robot platform should navigate to avoid objects.

![Combined Reactive Direction Diagram](image)

**Figure 20: Combined Reactive Direction**

The single unit-less value scalar serves as the input for the PID controller. The Simulink designed PID controller uses the gains shown in Figure 21. After the PID controller calculates the corresponding output, the output is processed by the singular to dual output converter. One output is sent to the left DC motor, and the second output is sent to the right DC motor. A resultant change in the RPM of each motor corrects course navigation of the mobile robot platform.

![PID Controller Gains](image)

**Figure 21: PID Controller Gains**
The results of the simulated PID controller are dependent on the gain values of $K_p$, $K_i$, and $K_d$. The optimal gain values produced the outputs graphed in Figure 22. The graph shows the left and right motor results with respect to RPM versus time. The middle graph represents the direction and degree of turning performed by the simulated robot platform in relation to time.

The three graphs (Figure 22) are grouped together for straightforward comparison at any given time of the simulation. When the middle graph shows a scalar value of less than 50, this implies the robot must turn some degree to the left. In order to accomplish this, the left side motor RPM decreased and the right side motor stay at a constant 100 RPM. A scalar value of greater than 50 implies the robot must turn some degree to the right. In order to accomplish this, the right side motor RPM decreases and the left side motor stays at a constant 100 RPM. Throughout these graphical results, the response of each motor follows these guidelines. Visually shown on the graphs, a peak on the right motor graph corresponds to a trough on the left motor graph, and vice versa.

The overall performance analysis of this simulated PID controller in relation to the ANN and FL controllers is discussed in the Results in Chapter 7.
5.1.4 PID Hardware Implementation

The physical implementation of each controller is completed through a hardware controller, designed and simulated in Quartus II® software and later implemented in the FPGA card by downloading the design onto the card. Input/output processing is performed by the BASIC Stamp 2e microcontroller. The FPGA card is chosen for its rapid processing ability and solely contains the individual controller. The BASIC Stamp 2e processes input and output data using PBASIC programming language (Parallax Inc., 2005). All controllers utilize the same mobile robot platform to perform navigations through the unknown indoor environments (See Chapter 3). By using the same robot
base, this eliminates comparison issues dealing with the physical components, and focuses the research on the controllers’ ability to generate appropriate output actions. The process of this controller implementation starts with reading the sensor values, weighting the individual sensors, combining the left side sensor values into a single value, and combining the right side sensor values into a single value. The error is calculated for each side using the set point of 30 and the sensor feedback. The error is then converted into a scalar value using a mathematical equation in BASIC Stamp code. The scalar value for the left side is averaged with the scalar value from the right side. The new single scalar value is then sent from the BASIC Stamp 2e to the error input on the FPGA card with the hardware implemented PID controller. The scalar value represents an equivalent millisecond pulse from 0-100. For example, if the scalar value is 15, the pulse to the FPGA is 15 milliseconds long. The complete BASIC Stamp commented code that produces the scalar value is given in APPENDIX B.

![Figure 23: PID controller on FPGA](image)

The overview diagram of the PID controller design on the FPGA card is shown above in Figure 23. The overall design is made of three main blocks: the input block to the FPGA card, the PID controller, and the output block from the FPGA card. The system clock runs at a speed of 48MHz. The input block in hardware design for the FPGA is shown in expanded version in Figure 24. This portion is responsible for reading in the pulse width input in milliseconds generated by the BASIC Stamp.
The first counter and comparator combination divides the system clock into one millisecond pulses. The 48MHz speed of the system clock equals 20.8 nanoseconds. The time of 20.8 nanoseconds is compared to 48007 to convert the nanoseconds into approximately one millisecond fragments. Once the counter reaches 48007, it resets itself. This accomplishes a pulse produced every one millisecond. The counter and register combination (DFF) counts the BASIC Stamp input pulse and stores it as an integer value. This counter activates with a high pulse from the error input and counts the number of one millisecond pulses. Once the input to the FPGA returns to a low, the load data pin receives a command from the BASIC Stamp to load the register with the counted value of one millisecond pulses. The BASIC Stamp also sends a signal to reset count pin to reset the counter back to zero in preparation for the next input. The number stored into the register is the integer value processed by the PID controller.

The PID controller block on the FPGA starts with a state machine to sequence and time the order of operations within the PID controller (Figure 25). The first inputs loaded are the sum of the errors and the previous error. These values are stored into registers within each term of the PID controller. The second process controlled by the state machine is to load a register with the answer each term produces. Once in the register, the terms are summed together and loaded into another register.
The essential configuration of the parallel PID controller architecture in Quartus II® software design is shown in Figure 26. The generalized PID controller configuration can be referenced for similarities and basic design in Figure 18. The $K_p$ section calculates the proportional term which is accomplished in this controller by multiplying the error by
a $K_p$ gain of 0.125 (Figure 27). The $K_i$ section calculates the integral term by summing all of the past errors and multiplying by a $K_i$ gain of 0.5 (Figure 28). The $K_d$ section calculates the derivative term by subtracting the current error from the previous error and multiplying the resultant number by the $K_d$ of 0.007813 (Figure 29). The method used in this controller to multiply by a gain term is shifting of the values to the right. Each subsequent shift divides the value in half. Therefore, the $K_p$ gain of 0.125 ($1/8$) is accomplished by three shifts to the right (Figure 27). The $K_i$ gain of 0.5 ($1/2$) is completed with one shift to the right (Figure 28). The $K_d$ gain of 0.007813 ($1/2^7$) is accomplished by seven shifts to the right (Figure 29). The three terms of the PID controller are summed together and placed in a register to produce the output value of the controller.

Figure 27: Proportional Term of PID on FPGA

Figure 28: Integral Term of PID on FPGA
A gain calculator was produced in Microsoft Excel to calculate the gains used in the FPGA implemented PID controller. The program requires a $K_p$, $K_i$, and $K_d$ value equivalent to a right shift value (Table 5).

<table>
<thead>
<tr>
<th>Value</th>
<th>Shift to the Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>1 times</td>
</tr>
<tr>
<td>0.25</td>
<td>2 times</td>
</tr>
<tr>
<td>0.125</td>
<td>3 times</td>
</tr>
<tr>
<td>0.0625</td>
<td>4 times</td>
</tr>
<tr>
<td>0.03125</td>
<td>5 times</td>
</tr>
<tr>
<td>0.015625</td>
<td>6 times</td>
</tr>
<tr>
<td>0.007813</td>
<td>7 times</td>
</tr>
</tbody>
</table>

Table 5: Value equivalent to Number of Right Shifts

The program with the actual $K_p$, $K_i$, and $K_d$ values used in this PID controller are shown in Table 6. This Excel program also requires RPM set points for RPM 1 and RPM 2. These values are equivalent to the range of RPM the actual controller produces. They are used to produce a graph showing the smoothness in transition between RPM outputs (Figure 30).
The output block converts the positive integer value controller output to an output pulse sent back out and read by the BASIC Stamp (Figure 31). The first counter and comparator combination mimics the same grouping in the input block. This combination divides the speed of the system clock to one millisecond pulses. The second counter and comparator combination in the output block is responsible for producing a high pulse in milliseconds equivalent to the output value of the PID controller.
The BASIC Stamp converts the millisecond pulse from the FPGA to an equivalent integer value. An example of this is a 10 millisecond pulse from the FPGA corresponds to a scalar value of 10 in the BASIC Stamp. The scalar value is converted into two corresponding values to send to the motor driver to control the speed of the left and right motor. The complete BASIC Stamp commented code for the left and right speeds is given in APPENDIX B.

![Figure 31: PID Controller Output Block on FPGA](image)

### 5.2 Artificial Neural Network

Artificial Neural Networks are a loose interpretation of biological neural networks. But why model their biological counterparts? The human brain is able to solve complex problems very rapidly. The mammalian neuron axon is able to conduct impulses at speeds of 20-100m/s. They are also able to send 100+ impulses within a single second (Hill, Wyse, & Anderson, 2004). This rapid processing ability combined with traits such as learning and adaptation provide the framework to model intelligent machines. Current uses for ANN’s are pattern classification, clustering/categorization, function approximation, prediction, optimization, retrieval of data by content, and control (Jain, Mao, & Mohiuddin, 1996). To understand how this research uses ANN’s for the purpose of control, it is key to understand their biological model.
5.2.1 Biological Neural Networks
The overall structure of a biological nervous system can be divided into two main parts: the central nervous system (CNS) and peripheral nervous system (PNS). The CNS includes the brain and spinal cord, while the PNS includes all sensory and motor neurons. The PNS is responsible for carrying input sensory information to the CNS for higher interpretation. The CNS then sends out a response impulse via the motor portion of the PNS to the organ or receptor to produce a reaction (Hill, Wyse, & Anderson, 2004). The structural units that carry out the impulse transmission are neurons. They are arranged in a network allowing for prompt communication throughout the body.

A neuron is the basic unit of the nervous system designed to generate an electrical impulse. The neuron is composed of four basic parts that each carry out a specific function for the cell (Table 7 and Figure 32).

<table>
<thead>
<tr>
<th>Structure</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dendrites</td>
<td>Input</td>
</tr>
<tr>
<td>Cell body</td>
<td>Integration</td>
</tr>
<tr>
<td>Axon</td>
<td>Conduction</td>
</tr>
<tr>
<td>Pre-synaptic terminals</td>
<td>Output</td>
</tr>
</tbody>
</table>

Table 7: Basic Structures of a Biologic Neuron (Hill, Wyse, & Anderson, 2004)
Neurons communicate with each other at specialized contact points called synapses. This junction is where a neuron receives input signals from other neurons. A single neuron can have contact with thousands of other neurons via synaptic junctions. The post-synaptic structures to take in the input impulses are the dendrites. The dendrites collect the impulses and pass them to the cell body. The cell body is the site of signal processing as well as impulse generation. The cell membrane supporting the cell body is responsible for summing all the excitatory and/or inhibitory inputs. If an action potential, also known as an impulse, is generated, it is propagated away from the cell body by the axon. The axon transmits the impulse to the pre-synaptic terminals. The
pre-synaptic terminals form synapses with the next neurons or receptor cells in order to communicate the output (Hill, Wyse, & Anderson, 2004).

An action potential is voltage dependent. This means the summing of the incoming impulses must initiate depolarization of the cell membrane. It is an all or none response. The depolarization must reach a voltage threshold in order to open voltage-gated ion channels on the membrane. When these channels open, rapid flow of ions creates the action potential. If the depolarization does not meet the voltage threshold, no impulse is generated. If the depolarization reaches suprathreshold levels, an action potential results. The impulses generated by a neuron are the same in amplitude and duration no matter how far above the voltage threshold the depolarization reaches (Hill, Wyse, & Anderson, 2004).

5.2.2 Artificial Neural Networks

An artificial neural network is made of up connections of simple processing units (neurons). The structure of an ANN mimics the network structure and communication abilities of a biological neural network (Figure 33).

![ANN Structure](image)

Figure 33: ANN Structure
Each circle represents an artificial neuron. They are arranged in a layered pattern with connecting lines and arrows indicating communication between layers. The basic neural network architecture consists of an input layer, one or more middle or hidden layers for processing and computation, and an output layer. The number of neurons in the first input layer is equivalent to the number of inputs into the system. Each input neuron having only one source. The number of hidden layers and the number of neurons per hidden layer are user defined for the processing abilities needed by the specific system. The number of neurons in the output layer corresponds to the number of outputs from the controller. A single neuron can have multiple input connections as well as multiple connections to the next neural layer. This is true except for the input layer which can only have one input per neuron (Skapura, 1996). Each neuron produces a single output, but it can be propagated to multiple neurons in the following layer (Figure 34).

![Single Neuron](image.png)

**Figure 34: Single Neuron**

Each artificial neuron or “unit” performs a mathematical computation. Within the computation, the input values are multiplied by the weight of the connection then summed together (Equation 5.6).

$$\sum_{n=\# \text{ of inputs}}^{n} \text{Weight}_n \ast \text{Input}_n$$

(Equation 5.6)
This computed value is termed the activation value and is used in an activation function which serves to produce a single output for an individual unit. The three most popular activation functions used are the linear, binary threshold, and sigmoid (Table 8).

<table>
<thead>
<tr>
<th>Activation Function</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>[ f(x; A, B) = \frac{B - x}{B - A} ]</td>
</tr>
<tr>
<td>Binary Threshold</td>
<td>[ f(x; A) = \begin{cases} 1 &amp; \text{if } x \geq A \ 0 &amp; \text{otherwise} \end{cases} ]</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>[ f(x; A, B, C) = \frac{1}{1 + e^{-\frac{x-B}{\tau}}} ] ( \tau = \text{control the shape of the curve} ) ( B = \text{control the transition of the curve} )</td>
</tr>
</tbody>
</table>

**Table 8: Activation Functions** (Skapura, 1996)

Unlike biological neurons that possess the all or none response in terms of generating an impulse, an ANN always propagates a value to the next neural layer. The output of each artificial neuron can still be described with the terms inhibitory or excitatory. If the value determined by the activation function is a zero, it is an inhibitory signal to the next layer. If the value is greater than zero, it will be added into the summation in the next neural layer and be considered excitatory to some degree.
Once the input signals are processed through the layered neural network, an output is sent to the plant portion of the system to elicit a response. The ANN control system implemented adapts and learns through a training process that is explained in the next section.

ANN’s are modeled after biological neural networks based on structure and arrangement. The following flow diagrams show these similarities side by side (Figure 35 and Figure 36)

**Figure 35: Biological and Artificial Neural Network Structural Similarities**

**Figure 36: Biological and Artificial Neuron Similarities**
5.2.3 ANN Training

ANN controllers are not tuned like PID and FL controllers. ANN systems incorporate learning. Here learning means self-adjusting connection weights between neurons until efficiency is reached. Connection weights are changed or learned through a training process. A ‘trainer’ formatted in software or hardware automatically updates the connection weights to improve system performance (Skapura, 1996). Training occurs in iterations of example situations. The number of iterations needed to achieve a trained ANN is based on system complexity and performance efficiency required.

There are three generalized learning models for an ANN. The first is the supervised learning model. In this method of learning, a correct response is known for every possible input. If a system’s input values range from 0-30, then 31 correct responses are provided for training purposes. Through the training process, each example input may lead to connection weight adjustment until the iterations lead to a desired amount of the provided correct responses. The second learning model is the unsupervised version. This method is not supplied with the desired correct response, but is allowed to formulate and organize data patterns. The third method combines the previous two into hybrid learning. Hybrid learning determines part of the connection weights through supervised learning and the other part unsupervised (Jain, Mao, & Mohiuddin, 1996).

Training an ANN can be done in simulation, implementation, or may be needed in both. To train in simulation, first a complete system must be designed within the simulation software. If the supervised learning model is being used, the range of inputs values with correct desired responses must be written into the simulation. Initially the connections weights between neurons are set to a default value. The training software
provides the ANN with input values simulating test runs known as iterations. The input value propagates through the network to generate an output. This output response is compared with the predetermined response for that input value. If the responses are the same, the connection weights do not change. If the responses are different, the software trainer automatically adjusts the network connection weights of the neurons that affected the output. Iterations continue until the network provides a series of correct outputs. The percentage of correct responses or percent error of each response that indicates a trained ANN is user defined.

From here the simulated trained connection weights can be applied to a physical controller. If both the simulated and implemented controllers have the same design, the connection weights determined in simulation should provide a trained controller in the implementation. Modifications to the network may need to be made if there are variations in design from simulation to implementation.

ANN’s can be trained purely after implementation. An implemented controller needs an on-board trainer designed in hardware or software to provide reinforcements of either/both punishments and rewards. A reward indicates a correct response and the connection weights do not change. A punishment indicated the network gave an incorrect response and the connection weight of the affecting neurons need adjusting. Training after implementation is a continuous as long as the on-board trainer is enabled. The ANN is able to learn and adapt in real time. Exponential growth of network learning results. The tradeoffs for a more precisely trained network lie in the time and effort required to design and build the on-board trainer. The pro’s and con’s of simulation
training versus training after implementation must be evaluated on a system to system basis dependent on system performance requirements.

5.2.4 Artificial Neural Network MATLAB® Simulation
The ANN controller was designed in MATLAB® using the Simulink Toolbox. This software is also used to simulation and train the controller. Figure 37 is the overview of the ANN design.

![Figure 37: Artificial Neural Network MATLAB® Simulink Design](image)

The set point is set to 30cm which is a pre-determined number to be used during physical implementation and navigation. The system is divided into two sides. A random number generator simulates sensor values within a 0-30cm range for both the left and right sides. The random number generators (sensors) provide feedback to the system. The error is the value propagated into the controller. This controller’s architecture utilizes two identical ANNs; one for the left side and one for the right side. The outputs produced by the ANN's are scalar values in the range of 0-100. The left side ANN is designed to produce an output in the range of 0-50. The right side ANN is designed to produce an output in the range of 50-100. The two outputs are summed and multiplied by a 0.5 gain to create an average. This average represents the
combined reactive direction (Figure 20). The single value enters the singular to dual output converter (Section 4.2) to produce outputs to the MATLAB® modeled plant: the left and right motors.

Both ANN's utilized within this controller have an identical structure, but produce scalar output values in different ranges (Figure 38).

![Artificial Neural Network MATLAB® Block Diagram (Left and Right)](image)

**Figure 38: Artificial Neural Network MATLAB® Block Diagram (Left and Right)**

The input layer collects data from a source. For this controller, the input data is the error calculated via the feedback. Each network has only a single neuron in the first layer because there is only one input data value per side. The two hidden layers provide the main computational processing (Figure 39). The first hidden layer is made up of five neurons. The input to this layer is weighted and a bias is added. The connection weights are determined during training which is explained at length later in this section. The bias is used to shift the activation function. A positive value shifts the function left and a negative value shifts the function right. This too is set by the software trainer. The activation function used by the first hidden layer is a sigmoid. It determines the output activation to the second hidden layer. The second hidden layer has a single neuron and receives five inputs with weighted connections from the first hidden layer. The inputs are summed together by this single neuron. The activation function for the second hidden layer is linear and it determines the output activation to the output layer.
The output layer processes a scalar value to propagate to the remainder of the system. After the controller interprets the data, the two ANN's produce a scalar value each that are averaged together.

![Diagram of Artificial Neural Network Hidden Layers in MATLAB® Design](image)

**Figure 39: Artificial Neural Network Hidden Layers in MATLAB® Design**

After designing the controller and system within MATLAB®, a training process within this software follows. The following MATLAB® code is used.

```matlab
>> T = [10 10 10 10 10 10 11 13 14 16 18 19 21 22 24 26 27 29 30 32 34 35 37 38 40 42 43 45 46 48 50];
>> P = [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30];
>> net = newff(P,T,5);
>> Y = sim(net,P);
>> plot(P,T,P,Y,'o')
```

The values following the letter 'T' are the correct response to each input used by the trainer to calculate when to adjust connection weights. The values following the letter 'P' are all the input values used by the system. P and T are cross reference by
MATLAB® in a P and T matrix. Net=newff indicates the set up of a new feed forward network using the inputs, P, compared to the correct responses, T, utilizing the five neurons in the first hidden layer. The next line of code directs the software to simulate the untrained network at default connection weight with the given inputs. The results of the untrained network are shown in Figure 40.

![Figure 40: Left Side Untrained Artificial Neural Network MATLAB® Simulation](image)

The solid line represents the correct responses in correlation with the inputs. The circles show the output response of the untrained ANN with each input. It is evident this untrained ANN performs inconsistently and rarely provides the correct output.

After initial views of the untrained network response, the training session is set up using the following code:

```matlab
>> net.trainParam.epochs = 50;
>> net = train(net,P,T);
>> Y = sim(net,P);
>> plot(P,T,P,Y,'o')
```

The first line indicates the maximum number of iterations, called epochs, which can be run to train the network. The second line of code simply states to train the network using the P and T matrix. The third line gives the signal to simulate, and then the
results are plotted (Figure 41). The solid line still represents the correct responses. The circles represent the outputs of the trained ANN with adjusted weight connections. The trained ANN shows almost perfect output responses when graphed with the given correct responses in set T. The more circle points that fall on the solid line, the more ideal the controller performance.

![Image](https://via.placeholder.com/150)

**Figure 41: Left Side Trained Artificial Neural Network MATLAB® Simulation**

MATLAB® gives options of different training algorithms to run. The trainer chosen for the ANN simulation in the research is the Levenberg-Maquardt. This is set at the default for feedforward networks. It is the "fastest training function" for this type of network (The Mathworks Inc., 2010). To calculate the performance error of the controller’s output to the correct response, the mean squared error is used. The mean squared error averages all the errors by taking the previous errors and averaging them with the current error. The default value for the mean squared error is +/- 0.1. When a controller output falls within this margin of error, the output is classified as correct. When the controller is able to elicit correct responses, a validation sequence ensures the ANN is consistently producing correct responses. The default number of validation
epochs set by MATLAB® is six. The following graph shows the training progress of the left side ANN in terms of mean squared error versus epoch (Figure 42).

![Graph showing training progress](image)

**Figure 42: Left Side Artificial Neural Network Training Simulation Results**

The simulation divides the input data into sections designated for testing, training, and validation. Testing runs the input data to understand how the untrained network responds with the default connection weights. Training runs the input data and adjusts the connection weights until the controller output is within the mean squared error margin. Then a minimum of six validation epochs run to ensure the network is trained properly. The lines on the graph exhibit a sharp decline as the error diminishes and comes within reach of the target mean squared error. The left side ANN took 15 epochs to run the test, train, and validation epochs. At epoch nine, the best performance is circled and coordinates to the x and y axes are shown. The six validation epochs follow resulting in 15 epochs total.

Typical post-analysis of an ANN using MATLAB® generates regression plots of the three areas of the training process (Figure 43). Each graph illustrates the best fit linear regression between the controller outputs and the correct responses. The dashed line
represents if the outputs of the network were equivalent to the correct responses. The calculated $R$ value represents the relationship between the outputs and the correct responses. An $R$ value of 1 indicates exact linear relationship. An $R$ value of 0 indicates no linear relationship (The Mathworks Inc., 2010).

![Regression Plots](image)

**Figure 43: Left Side Artificial Neural Network Simulation Regression Plots**

The right side ANN is functions approximately the same as the left side. The difference here is the correct response scalar value. Remember the right side ANN must produce an output in the 50-100 range. The code to initiate set up of the right side ANN is as follows:
Notice the values in the T set starting at 90 and slowly decreasing until the minimum of scalar value of 50. The network is simulated untrained and produces Figure 44.

![Distance vs. Unitless Value](image)

**Figure 44: Right Side Untrained Artificial Neural Network MATLAB® Simulation**

The inconsistency and lack of correct response is no surprise in the untrained network. The connection weights have not been adjusted at this point and are set only at default values. Set up for training the right side network uses the same code and explanation as the left side.

```matlab
>> net.trainParam.epochs = 50;
>> net = train(net,P,T);
>> Y = sim(net,P);
>> plot(P,T,P,Y,'o')
```

The trained right side ANN results are plotted in Figure 45. The right side training completed in 34 out of 50 possible epochs with nearly perfect responses.
The Levenberg-Marquardt trainer and mean squared error are again utilized by the right side ANN training simulation.

The right side ANN took 34 epochs to run the test, train, and validation epochs. At epoch 28, the best performance is circled and coordinates to the x and y axes are shown. The six validation epochs follow resulting in 34 epochs total. The graphed
testing line is spaced farther from the other two lines due to the larger inaccuracy of the untrained network.

The regression plots produced by the right side ANN signify the same concepts as the left side regression plots (Figure 47). The test graph shows a less linear result than the left side ANN due to the original untrained network producing erratic response (See Figure 44). Although the graph gives the perception of the data and correct response having a poor linear relationship, it is important to look at the scale on the axes as well as the R value. The R value is 0.98282, and a value of one represents exact linear relationship.

![Figure 47: Right Side Artificial Neural Network Simulation Regression Plots](image)
The MATLAB® Simulink simulation produced the following results for the ANN controller (Figure 48). The results of the individual motors in relation to controller output are shown for an untrained, partially trained, and fully trained ANN. The untrained network reveals both understated and exaggerated reactions. The partially trained network provides a clear view of the stepping stones the training process provides. The fully trained network provides the correct responses for the system and can be referenced for comparison. The overall performance analysis of this simulated ANN controller in relation to the PID and FL controllers is discussed in the Results in Chapter 7.

![Figure 48: Artificial Neural Network Controller MATLAB® Simulation Results](image-url)
5.2.5 Artificial Neural Network Hardware Implementation

This ANN controller again utilizes the communication between the input/output processing device (BASIC Stamp 2e) and the hardware implemented controller (FPGA). The BASIC Stamp code is similar to the code used in the PID controller; however, instead of a single value sent to the FPGA from the BASIC Stamp, two values are sent. The left and right side error values are not combined before communication to the ANN controller.

The BASIC Stamp code starts the same with the sensor values read in, individual sensors weighted, the left side sensor values combined into a single value representing the left side, and the right side sensor values combined into a single value representing the right side. The error is calculated for each the left and right side using the established set point of 30 cm and the sensor feedback values. The left side error and right side error are sent as two equivalent millisecond pulses to the FPGA card. See APPENDIX B for complete commented BASIC Stamp code for the ANN.

The overview of the ANN controller implemented in hardware on the FPGA card is shown in Figure 49. It contains two input blocks to process the two input pulses (ms) from the BASIC Stamp, the controller block, and an output block. The input blocks have the same design configuration and purpose as stated in the PID controller Hardware Implementation Section 5.1.4. The input block diagram (Figure 24) and operational explanation can be referenced from the aforementioned section. The input blocks here in the ANN controller will of course produce two separate error integers to be processed through the controller section.
The expanded hardware design of the ANN controller block is shown in Figure 50. The left and right side error values are used to locate an address of the correct output response. The left side error is multiplied by the number of rows in the matrix (31). The right side value enters a converter to convert it from a 16 bit number to a 21 bit number for the mathematical operations. The left and right side values are added together to produce an address location. This location value is converted from a 21 bit value to a 10 bit value before entering the ROM memory. Within the ROM memory a look up table is stored holding scalar value outputs.

The ANN Look-up table was created using Microsoft Excel to produce all possible combinations of inputs to the network versus all possible outputs. A complex in-depth matrix used as a look up table is generated in place of a hardware designed network.
The reasoning for creating a look up table in place of a hardware designed network is that the physical implementation of the ANN controller is meant to use an already trained network. This is because the simulation trained the network and provided the correct weights and biases for each neuron in the network. The look up table provides the same output response the actual network would create if physically implemented using the same weights and biases the simulation provided.

The matrix of outputs was created in Excel by using the mathematical calculations each simulated neuron from MATLAB® would make. The simulation provided the correct connection weights between neurons and each neuron’s bias. These simulated results in addition to the activation function for each neuron can be computed in Excel using every possible input combination from the left and right side errors. The process of this computation can visually be explained in an ANN structure (Figure 51).

The resultant matrix of scalar output value contains 961 possible combinations (Table 9). The number of combinations results from 31 possible inputs from the left side and 31 possible inputs from the right side. The matrix is color coded to allow visualization of scalar output trends. The location of each scalar output value is determined by multiplying the left side error value by 31 and adding the right side error value. This calculation provides the address location of the correct scalar output response to be sent via the FPGA output block back to the BASIC Stamp.
Figure 51: ANN Structure
Once the output response has been located, the scalar output is sent to the output block. The output block of the ANN controller is structurally and functionally identical to the output block explained in the PID controller Hardware Implementation (5.1.4). The BASIC Stamp again converts the singular output from the controller into two outputs for

|       | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Right Side Value</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Table 9: ANN Unit-less Value Output Matrix
the motor driver to change the speed of the left and right motors. See APPENDIX B for complete BASIC Stamp commented code for the ANN.

5.3 Fuzzy Logic

Fuzzy Logic (FL) plays off human ability for commonsense reasoning. Humans can reason an answer even when the information used to process the result is ambiguous or uncertain. Fuzzy Logic control systems are able to formulate a definitive output even when given an input that is not completely or clearly defined. FL uses linguistic variables to represent a range of values. Within this language, the input is a noun such as speed or distance and the linguistic or fuzzy variable is an adjective such as weak, strong, slow, or fast. Although this language gives the impression of being imprecise, to a human this language can be very descriptive when processed by our cognitive inferences. By using linguistic variables in a FL controller, the controller takes on a persona similar to a human allowing it to be classified in the artificial intelligence realm.

An FL controller works in a progression of three steps (Figure 52). First it receives input data that is processed through a fuzzification step. Fuzzification involves pre-set membership functions for data interpretation as defined by the user. This data then enters a rule matrix of IF-THEN statements to create a fuzzy output. In order for the controller to use the processed output, one last step, a defuzzification process turns the fuzzy output into a clear and concise output value to be performed by the system.

![Figure 52: Fuzzy Logic Controller](image-url)
5.3.1 Fuzzification, Rule Processing, and Defuzzification

Fuzzification involves taking in a distinct input value that may belong fully or partially to a membership category. The fuzzification process transforms and reorganizes this input based on pre-determined membership categories. Membership categories are grouped together and collectively termed ‘fuzzy sets’ (Zadeh, 1965). Examples of membership categories are weak, medium, and strong. When used together, these categories become a fuzzy set describing strength of an input or output. These fuzzy sets are used to create membership functions (MF). Membership functions are depicted in a graph showing the degree of participation the data has to each category in the fuzzy set (Figure 53).

![Figure 53: Generic Membership Function](Kaehler, 1998)

The x-axis of the graph represents the degree of membership the data has based on a scale determined by the user. The higher on the y-axis a data value falls, corresponds to a higher degree of membership to that category. The x-axis represents the range of...
values of an input or output in the units used by the system. Processing the input through the MF results in multiple fuzzy values to be used in the next step. The fuzzification step can include multiple membership functions. The number of fuzzy values the MF produces is equivalent to the number of categories within the each fuzzy set. This is how partial membership is determined. Using Figure 53, the example input (red dotted line) will separate into three fuzzy values (Table 10).

<table>
<thead>
<tr>
<th>Membership Categories</th>
<th>Fuzzy Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative</td>
<td>0.75</td>
</tr>
<tr>
<td>Zero</td>
<td>0.25</td>
</tr>
<tr>
<td>Positive</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 10: Fuzzy Values

The graphical representation of MFs can take on many shapes. The three most common shapes are (1) triangular, (2) trapezoidal, and (3) shoulder (Zhao, & Bose, 2002). Table 11 shows these generic membership function forms. Also explained in Table 11 are the equations used to create the MF using a pre-defined range of values set by the user.
The output membership function has the same design as an input membership function, but is used during the defuzzification step of the controller. The stage before the output membership function is rule processing.

The goal of rule processing is to create a fuzzy output given the fuzzy input(s) from the previous input membership function(s). The rule base is the predetermined rule matrix defined by the user to generate an appropriate output based on the fuzzy inputs. Rules are stated in conditional IF-THEN statements conjugated by logic operations AND or OR. When AND is used in a rule statement, the minimum value between the compared membership functions is propagated to the defuzzification step. When OR is used in a rule statement, the maximum value between the compared membership functions is propagated to the defuzzification step. The logic operation is chosen by the user to
create an FL controller specific to the system needs. The number of rules is determined by both the number of input membership functions and the number of linguistic variables per membership function. The formula for the number of rules an FL controller has is shown in Equation 5.6.

\[
\text{# of Rules} = \prod_{n=\#MF}^{\infty} \text{# of Linguistic Variables in } MF_n
\]

(Equation 5.6)

A rule matrix is created and stored in the rule base in the controller’s memory. It is read as a table used to compare MF’s. The user defines the rules with appropriate reactions to be taken to the defuzzification step. The number of fuzzy outputs corresponds to the number of linguistic variables in the output MF.

Defuzzification takes the fuzzy output from the rule processing and transforms it into a distinct output using an output MF. The clear output is propagated to the plant for processing. The output MF has the same design concept as the input MF described previously. The rule matrix produces a fuzzy output for each linguistic variable in the output MF. These values fall on the y-axis of the output MF, and are then graphed on each corresponding linguistic variable. The area below each of these lines is used to calculate the distinct output through the defuzzification process. The value on the x-axis is the value sent to the plant.

There are five methods of defuzzification to produce the x-axis value. They all map the fuzzy outputs in the same way, but calculate the distinct output in various ways. Table 12 lists the five methods of defuzzification as well as the mathematical equations to calculate the crisp output value. The most commonly used method is the centroid
method. Below the table is a graphical representation (Figure 54) of the outputs calculated on an example output MF.

<table>
<thead>
<tr>
<th>Defuzzification Method</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Centroid</strong></td>
<td>[ \frac{\sum i \mu_c(x_i)x_i}{\sum i \mu_c(x_i)} ]</td>
</tr>
<tr>
<td><strong>Bisector</strong></td>
<td>[ \sum_{i=1}^{j} \mu_c(x_i) - \sum_{i=j+1}^{i_{\text{max}}} \mu_c(x_i) \mid, \quad i &lt; x_j &lt; i_{\text{max}} ]</td>
</tr>
<tr>
<td><strong>Mean of Maximum</strong></td>
<td>[ \frac{\sum i \in I x_i}{</td>
</tr>
<tr>
<td><strong>Smallest of Maximum</strong></td>
<td>[ x_{\text{min}}(I) ]</td>
</tr>
<tr>
<td><strong>Largest of Maximum</strong></td>
<td>[ x_{\text{max}}(I) ]</td>
</tr>
</tbody>
</table>

\( i = \# \text{ of output value created by the rule processing} \)
\( x_i = \text{output values of the rule processing} \)
\( \mu_c(x_i) = \text{membership value of the output function set} \)
\( x_j = \text{abscissa of the bisector} \)
\( |I| = \text{number of memberships} \)
\( I = \text{maximum membership value} \)

**Table 12: Five Main Defuzzification Methods** (Namazov, & Basturk, 2010)

**Figure 54: Five Main Defuzzification Methods** (The Mathworks Inc., 2010)
5.3.2 Fuzzy Logic Tuning

Tuning an FL controller can become quite complex very quickly. There are many variables within the controller design that can be tweaked in order to adjust the final output. Background knowledge of each section of controller design and fundamental understanding of the combination of sections aid in a successful tuning session. With so many variables to manipulate to reach the optimal response, considerable time may need to be invested in tuning an FL controller. A controller may be tested in a computer simulation before physical implementation. Tuning may need to be done in both of these areas of controller design and implementation. As discussed in the last section, an FL controller can be broken down into three main steps. Each of these steps has areas within it that can be adjusted to fine tune the overall output.

In the fuzzification step, the range values of each linguistic variable in the input MF can be changed. By changing these values to cover either a shorter or broader range, changes the slope of the shape of the MF. The slope corresponds to the degree of membership the input value produces. The second step, the rule processing referencing the rule base, can be revised to create rules to generate slight to substantial differences in the fuzzy outputs. The last section, defuzzification, has two areas that can be adjusted. The first is the output MF. It can be modified the same way as the input MF by changing the range of the linguistic variables that compose the output MF. The second portion of defuzzification that can be changed is the method of defuzzification that determines the final output value. Out of all the areas to be fine-tuned, the method of defuzzification is generally the first element to change. Tuning and experimentation with these five methods (Table 12) may be enough to generate an output to better suit the needs of the system. If the optimal response is not evoked by
changing the method of defuzzification, then the user can go back and tweak the other steps of the FL controller.

5.3.3 Fuzzy Logic MATLAB® Simulation

MATLAB® Simulink software is used to design and simulate the Fuzzy Logic controller and plant portion of the overall system. The FL MATLAB® Simulink design is shown in Figure 55.

![Fuzzy Logic MATLAB® Simulink Design](image)

**Figure 55: Fuzzy Logic MATLAB® Simulink Design**

The simulation design begins with a random number generator to create sensor values in the range of 0-30 cm. This range is based on pre-planning for physical implementation. The simulation produces two of these inputs; one for the left side and one for the right side of the simulated system. Both values are propagated straight into the FL controller. The MATLAB® FL controller begins with two input MFs. One MF
calculates the fuzzification for the left side input and the second MF does the same for
the right side input. These values are processed by the rules to generate a fuzzy
output. There is one MF responsible for defuzzification of the fuzzy value. The FL
controller yields are unit-less value between 0-100. The singular to dual output
converter (Section 4.2) processes an output for the left motor and an output for the right
motor.

The FL controller designed in this research contains two input MFs (Figure 56). The
functions are identical, but one processes the left sensor inputs and the other processes
the right sensor inputs. The input values simulate sensor readings of the distance the
sensors are from an object. The linguistic variables used by the input MF describe this
input in terms of sensor signal distance strength. The five linguistic variables chosen
are Very Strong (VS), Strong (S), Medium (M), Weak (W), and Very Weak (VW). An
input falling within the VS membership category indicates an object is very close, and
conversely an input within the VW membership category means an object is a safe
distance away. The input MF’s degree of membership located on the y-axis uses a
scale of 0 to 1. Each end of the linguistic variable spectrum has a shoulder shaped MF.
The middle linguistic variables are triangular shaped.

![Figure 56: Fuzzy Logic Input Membership Functions](image)
After each input MF produces five values as fuzzy inputs for the rule processing, they are referenced to the rule base. Using Equation 5.6, this simulated FL controller requires 25 rules to define the fuzzy outputs. All rules use an AND logic operation and are stated as follows:

1. If (Right-Side is VW) and (Left-Side is VW) then (Unit-less is ZR)
2. If (Right-Side is VW) and (Left-Side is W) then (Unit-less is SR)
3. If (Right-Side is VW) and (Left-Side is M) then (Unit-less is SR)
4. If (Right-Side is VW) and (Left-Side is S) then (Unit-less is MR)
5. If (Right-Side is VW) and (Left-Side is VS) then (Unit-less is MR)
6. If (Right-Side is W) and (Left-Side is VW) then (Unit-less is SL)
7. If (Right-Side is W) and (Left-Side is W) then (Unit-less is ZR)
8. If (Right-Side is W) and (Left-Side is M) then (Unit-less is SR)
9. If (Right-Side is W) and (Left-Side is S) then (Unit-less is MR)
10. If (Right-Side is W) and (Left-Side is VS) then (Unit-less is MR)
11. If (Right-Side is M) and (Left-Side is VW) then (Unit-less is SL)
12. If (Right-Side is M) and (Left-Side is W) then (Unit-less is SL)
13. If (Right-Side is M) and (Left-Side is M) then (Unit-less is ZR)
14. If (Right-Side is M) and (Left-Side is S) then (Unit-less is SR)
15. If (Right-Side is M) and (Left-Side is VS) then (Unit-less is SR)
16. If (Right-Side is S) and (Left-Side is VW) then (Unit-less is ML)
17. If (Right-Side is S) and (Left-Side is W) then (Unit-less is ML)
18. If (Right-Side is S) and (Left-Side is M) then (Unit-less is SL)
19. If (Right-Side is S) and (Left-Side is S) then (Unit-less is ZR)
20. If (Right-Side is S) and (Left-Side is VS) then (Unit-less is SR)
21. If (Right-Side is VS) and (Left-Side is VW) then (Unit-less is ML)
22. If (Right-Side is VS) and (Left-Side is W) then (Unit-less is ML)
23. If (Right-Side is VS) and (Left-Side is M) then (Unit-less is SL)
24. If (Right-Side is VS) and (Left-Side is S) then (Unit-less is SL)
25. If (Right-Side is VS) and (Left-Side is VS) then (Unit-less is ZR)

The fuzzy outputs correspond to the linguistic variables in the output MF. The goal of this controller is to control the motors, so the output linguistic variables reference steering direction with Medium Left (ML), Slight Left (SL), Zero (ZR), Slight Right (SR), and Medium Right (MR). The rules can also be visualized in a rule matrix (Figure 57).
The fuzzy outputs with the same color shading represent all possible outputs for that linguistic variable. Since this controller uses the AND logic operation, the minimum value for each output linguistic variable is processed by the output MF.

The output MF (Figure 58) is responsible for defuzzification. Here the five fuzzy inputs from the rule processing coordinate to the y-axis. The final distinct output that is propagated to the plant is determined by the centroid method. The corresponding value on the x-axis is a unit-less output processed by the remainder of the system.

Figure 58: Fuzzy Logic Output Membership Function

Figure 59 illustrates two example inputs processed through the input MF graphs and gives the corresponding output MF graph. They are shown in the format the rules are read in. The first column shows an input sensor reading from the right side at 15cm.
The line down the column shows where this value falls on the input MF. The middle column illustrates the same, but represents an input value of 15cm from the left side. The last column shows the corresponding location on the output MF the rule falls. The last box in this column represents the total area of the output MF used and the centroid calculation giving a resultant unit-less output of 50.

Figure 59: Fuzzy Logic Example Input Processing
When every rule is analyzed with every possible input for both the left and right sides, along with every possible calculated output, a surface graph is produced. Figure 60 is a three dimensional representation of the controller surface. A surface graph is a tool used in tuning the FL controller. The graph shows every possible output and is compared to the expected performance criteria the controller is expected to meet.

Figure 60: Fuzzy Logic Surface Graph

The MATLAB® Simulink simulation produced the following results for the FL controller (Figure 61). The results indicate a very quick and precise controller reaction to fluctuations in input sensor values. The simulation results correspond with the expected reactions of the system. The overall performance analysis of this simulated FL controller in relation to the PID and ANN controllers is discussed in the Results in Chapter 7.
5.3.4 Fuzzy Logic Hardware Implementation

The FL controller keeps with the same component communication design as the PID and ANN controllers. The BASIC Stamp 2e is the input/output processing device and the FPGA card is used to physically implement the controller in hardware for rapid processing.

As with the two previous controllers, the BASIC Stamp starts the FL system processing by reading in the sensor values, weighting individual sensors, combining the left side sensor values to a single value, and combining the right side sensor values to a single value. The two resultant values are sent to the FPGA card as equivalent millisecond
pulse widths. Complete commented BASIC Stamp code for the FL controller can be
referenced in APPENDIX B.

The overview of the FL controller is structurally similar to both the PID and ANN
controllers. The Quartus II® design consists of two input blocks, each corresponding to
either the left or right side inputs, an FL controller block, and an output block (Figure
62). The FL input blocks are consistent in structure and function as the previous two
controllers. See Figure 24 and the operational explanation of the input block for details
(Section 5.1.4).

Figure 62: Fuzzy Logic Controller on FPGA

After the two values have been inputted to the FPGA card and stored into the register in
the input block, they are sent to the FL controller for processing. The schematic of the
FL controller in hardware is shown in Figure 63.

The three distinct sections an FL controller, fuzzification, rule processing, and
defuzzification, are represented. The fuzzification section holds two input membership
functions. The rule processing section compares the left and right fuzzy inputs to the
rule base to produce fuzzy outputs. The defuzzification section uses the centroid
method to produce a clear output.
The fuzzification section is shown in expanded version in Figure 64. This diagram represents only one of the input membership functions because the design for each input MF is the same. The input is received from the input block on the FPGA card from either the left or right side. It propagates to the five ROM blocks shown. Each ROM block represents an individual linguistic variable of the input MF. Within each block, the sensor value input provides the address for the fuzzy value for the particular linguistic variable. Table 13 shows all the sensor inputs and their corresponding fuzzy values per linguistic variable. The top ROM block (rom3) contain linguistic variable ‘Very Strong.’ The fuzzy values in the VS column of Table 13 are found here. This pattern continues for all 5 ROM block linguistic variables and corresponding columns from Table 13.
Each input MF produces five fuzzy values. Overall the fuzzification section produces 10 outputs, five per input MF, used as fuzzy inputs to the rule processing section.

![Figure 64: Input Membership Function on FPGA](image)

<table>
<thead>
<tr>
<th>Input Sensor Value (cm)</th>
<th>VS</th>
<th>S</th>
<th>M</th>
<th>W</th>
<th>VW</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>2</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>29</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 13: Sensor Input & Corresponding Fuzzy Values per Linguistic Variable
The rule processing section in Figure 63 shows five rule blocks on the left and five rule blocks on the right. The left rule blocks process the rules for the FL controller. The physically implemented FL controller uses the same rules as the MATLAB® simulation but exchanges the logic operation in all rules from AND to OR. The rules are listed in section 5.3.3. The OR operation takes the largest value between compared values.

An expanded version of each rule block is shown in Figure 65. A single rule block compares a single linguistic variable from the right side to all linguistic variables from the left side. Figure 66 gives an in-depth look into each rule block in Figure 65. Within each of the rules blocks in Figure 65, the OR logic operation is performed. This is what is represented in Figure 66.

Figure 65: Rule Processing on FPGA

Figure 66: OR Operation on FPGA
The right side rule blocks from Figure A use the completed rule matrix from the left side rule blocks. The right side rule blocks perform the OR logic operation for all like output linguistic variables. (See color coded rule matrix in Figure 57) Each right side rule block processes a single output linguistic variable to be narrowed to a single fuzzy value through the OR operation. The right side rule blocks thus produce five fuzzy outputs to the output MF in the defuzzification section.

The defuzzification section uses the centroid method equation from Table 12. The expanded version of this section is shown in Figure 67.

![Figure 67: Defuzzification Section on the FPGA](image)

The five fuzzy outputs produced by the rule processing are summed together to provide the first value for the centroid calculation. The fuzzy outputs are also multiplied by their corresponding centroid value from each other their respective linguistic variables. This produces the second value for the centroid calculation. The first value is divided by the
second value to produce the clear output of the FL controller. This output is sent to the output block on the FPGA to be converted to an equivalent millisecond pulse to be sent to the BASIC Stamp. The BASIC Stamp reads the output pulse from the FPGA and converts this value to the appropriate signals to send to the motor driver board to change the speed of the left and right motors. Complete commented BASIC Stamp code for the FL controller is found in APPENDIX B.
CHAPTER 6

CONTROLLER TESTING AND PERFORMANCE EVALUATION

Each controller was independently implemented onto the FPGA card and mounted to the mobile robot platform. They were tested on their ability to deal with different situations in real time in an unknown indoor environment. Each controller was tested in the same 10 environmental layouts, creating different testing situations. After the first testing environment is set up, the PID, Artificial Neural Network, and Fuzzy Logic controllers each were independently evaluated on how well they navigate the environment. After all three controllers have had their trial run in the same testing environment; a different environment is then configured. The pattern of allowing each controller a trial run within a specific layout before re-configuring the environment continues for the remainder of the 10 layouts. The goal of each run is to successfully traverse the real environment to ultimately locate the RFID tag. Each run by each controller is graded with a standardized rubric.

6.1 Testing Environment
Testing of the controllers is done in a physical indoor environment with a variety of static objects places in configurations unknown to each controller. The unknown indoor environment was constructed of 6.35mm thick plywood arranged in a 2.4m square with 30cm high walls. The testing environment is built on top of a tile floor. Located within the testing environment is a set of static objects. The objects are placed in distinct positions to present specific situations to test the controllers’ ability to respond. The static objects are simple shapes made of cardboard. These shapes and sizes are presented in Table 14 and Table 15.
### Table 14: Geometric Shaped Static Objects

<table>
<thead>
<tr>
<th>Shape</th>
<th>Size (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangle</td>
<td>15 x 30</td>
</tr>
<tr>
<td>Square</td>
<td>15 x 15</td>
</tr>
<tr>
<td>Triangle</td>
<td>15 x 15 x 15</td>
</tr>
<tr>
<td>Circle</td>
<td>15 Dia.</td>
</tr>
</tbody>
</table>

### Table 15: Simple Shaped Static Objects

<table>
<thead>
<tr>
<th>Shape</th>
<th>Size (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>30 x 30</td>
</tr>
<tr>
<td></td>
<td>60 x 60</td>
</tr>
<tr>
<td></td>
<td>30 x 90</td>
</tr>
<tr>
<td>Wall</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>90</td>
</tr>
<tr>
<td>T</td>
<td>60 x 60</td>
</tr>
</tbody>
</table>
The height of all the static objects is consistent with the height of the walls: 30cm. Each testing environment contains three different shape static objects from Table 14 and two different shape static objects from Table 15. The static objects are placed in a pre-planned layout that is unknown to the controllers before each trial run. See APPENDIX C for the 10 unknown indoor environment layouts. Once an environment is set up, each of the three controllers will independently navigate it before the environment is broken down and a new layout set up.

The tile floor is covered with a roll of paper. The mobile robot platform has a marker attached to the underside to trace the path through the testing environment. This path is recorded onto the paper to document the controllers’ actions when presented with the situations within the environment. For each testing environment layout, the three controllers’ paths are traced by the mobile robot platform on the same sheet of paper. The PID controller’s path is denoted by a blue maker. The Artificial Neural Network controller’s path is denoted by a red marker. The Fuzzy Logic controller’s path is denoted by a green marker. By using the same paper for each environment, the three paths determined by the controllers' decisions can later be referenced during analysis for comparative purposes.

Within the testing environment, the constants include the number of static objects, the overall dimensions of the environment, and the starting location and direction on the mobile robot platform. The unknowns of the environment are the shape and size of the static objects, the configuration of the objects, and the location of the RFID Tag goal. The RFID Tag location changes with the environmental layout in the same manner as the static objects.
6.2 Rubric

A standardized rubric was created for grading all controllers. Scoring in this study is done largely on a situational basis. Using the static objects arranged in the 10 layouts shown in APPENDIX C, the controller must navigate in or around the situations presented. The rubric scores for the following seven situations: room, corridor, hole, small object, large object, angular approach from the left, and angular approach from the right. A room is defined as any three walled area smaller than the overall environment. A corridor situation presents with two parallel walls creating a hallway wide enough for the mobile robot platform to traverse. A hole situation is described as two static objects creating 30cm to 60cm space between that the robot can navigate through. A small object situation presents when the robot approaches any of the static objects that has at least one side smaller than the diameter of the mobile robot base (<30cm). A large object situation is described as the mobile robot platform approaching any static object with all sides equal or larger than the diameter of the mobile robot base (≥30cm). The last two situations are defined as the mobile robot platform approaching any 90cm section of wall at any angle other than 90 degrees. One situation is defined with a left side approach, while the other is defined with a right side approach. Also on the rubric are two overall scores for navigation ability and intelligence.

Each situation or overall evaluation is graded on a scale of one to four, one being the worst, and four being the best. Located under each score for each situation is a descriptive performance guideline to consider when the evaluator is scoring. For this study four individuals score each trial run for each controller. This allows for more data compilation as well as more than one individual's perspective. A large comment
section is provided on the bottom of the rubric for the evaluator to make observations or brief explanations of scoring. The rubric can be found at the end of this chapter.

With 10 different testing environment layouts, not all situations will be encountered in each run, but a minimum of five out of seven situations are present in each environment. If the controller locates the RFID Tag before encountering the minimum five situations, the overall score for the run is based only on the situations encounters.

The evaluator also gives a pass/fail score to each run. A pass indicates the controller navigated through the testing environment and found the RFID Tag goal in a timely manner. A fail indicates the controller either did not find the RFID Tag goal in a timely manner, or it was unable to successfully navigate the environment. Each controller is given a time limit of 5 minutes to complete the course and find the goal. If the controller is experiencing extreme difficulty within a situation, it is given a 60 second time limit before the run is terminated.
Situational Rubric for PID, ANN, and FL controllers navigating a real unknown indoor environment

Evaluator: _______________________
Robot Marker Color:    Red    Blue    Green
Environment Layout #: __________

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td></td>
</tr>
<tr>
<td>Corridor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small Object</td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td></td>
</tr>
<tr>
<td>Large Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angular Approach Left</td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td></td>
</tr>
<tr>
<td>Angular Approach Right</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score</td>
</tr>
<tr>
<td>Navigation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall Intelligence</td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td></td>
</tr>
</tbody>
</table>

Total:

Comments:

Time: __________________________ Pass [ ] Fail [ ]
CHAPTER 7
FINDINGS OF THE STUDY

7.1 MATLAB® Simulation Results

In this chapter we present an analysis of the results obtained when testing each of the three controllers implemented. Each controller was downloaded to the FPGA card and then tested to control the navigation of the robotic base.

In Figure 68 we show the results obtained from the MATLAB® simulation of the three controllers, similar to the individual results discussed in Chapter 5. The only difference in this graph is that the results are arranged so that the graph is scaled to show the results in the same axes for comparison purposes.

The first and third graphs show the changes in the RPM for the right and left motors respectively. The second graph plots the controller output scalar values over time. These values are used by the single to dual output converter to send the two appropriate outputs on the left and right motors. A scalar value of 50 means the robot moving in a straight line, a value between 0-49 indicates the controller commanding a left turn. For the simulated system to perform a left turn, the right motor should stay at a constant 100 RPM, and the left motor should decrease in RPM directly proportional to the controller output. A scalar value between 51-100 indicates the controller commanding a right turn. For a right turn to be performed, the left motor should stay at a constant 100 RPM while the right motor decreases in RPM directly proportional to controller output.
7.1.1 Proportional, Integral and Derivative Controller Response
As shown in graph (Figure 68) the PID controller exhibits a slow response overall, although it is most evident between 0-8 seconds. The delayed reaction time for the PID controller may be inhibited by the gain values. $K_i$ is established to reduce the steady state error, but in turn increases the overshoot. $K_d$ diminishes the overshoot. A large $K_d$ value should also help quicken the overall response of the system; however, too large of a $K_d$ makes the system subject to much more noise disturbance. For this system, the gain values allow for stability but do not imply prompt responses to inputs. The right motor controlled by the PID controller shows a delayed reaction time between 6-10 seconds. The controller starts to initiate the correct response, but is not able to meet the expected response value before the input value changes again.

7.1.2 Artificial Neural Network Controller Response
The ANN controller displays accurate responses overall. The controller appears to be running at peak performance with no significant issues throughout the simulation (Figure 68). It is able to meet the RPM changes within a reasonable time. Most responses look as if they are smooth transitions; however, from 6-8 seconds and 15-17 seconds, the simulation graph shows steep slopes indicating fast decreases in RPM. In the actual implementation this may result in too sharp turning that could lead to skidding. Training the ANN provides previous encounters with the same input values, so the controller has already learned and adapted to deal with these values to generate an accurate response.

7.1.3 Fuzzy Logic Controller Response
The FL controller shows appropriate responses for most of the simulation (Figure 68). At time 3-6 seconds the FL controller shows better resolution to small scale fluctuations.
The scalar value increases slightly above 50, and the FL controller communicates a small decrease in the right motor RPM while keeping the left motor at 100 RPMs. The only real discrepancy in FL controller simulation is from 18-20 seconds. The graph indicates an inconsistency in reaction to the increase in scalar value. These two distinct reactions may be explained by the combination of the rules and membership functions that interpret the data and where a threshold to response is located.

7.1.4 Controllers Overall Comparison
By comparing the response of the three controllers overall, the poorest performance is executed by the PID controller, and the ANN and FL controllers show similar performance levels. The PID controller is evidently much slower and imprecise in reactions and timing. The ANN controller shows consistency, but a discrepancy lies in the 3-6 second interval. In this interval, the FL controller exhibits accurate performance in decreasing the right motor RPM, while the ANN controller appears to ‘glance over’ this small unit-less value fluctuation. The FL controller demonstrates smooth transitions with mostly accurate responses except for its discrepancy found in time interval 18-20 seconds. Both the ANN and FL controllers show adequate responses, but with minor discrepancies. With this simulation data, it is difficult to determine a clear controller with the best overall performance. Further comparison of the three controllers is done through actual implementation with comparison emphasis on situational responses.
7.2 Robot’s Navigation Testing & Rubric Results

The physical implementation of the controllers was tested throughout ten environment layouts (the layouts are shown in APPENDIX C: TESTING & EVALUATION ENVIRONMENT CONFIGURATIONS). Each test was scored using a standardized rubric that was pre-developed to grade the controllers’ reactions to specific situations, overall navigation, and overall perceived intelligence (Section 6.2). Four individuals observed and graded each controller as it navigated each of the ten testing environments. Two of the four individuals had no prior knowledge of the order the controllers were tested in. The only correlation these individuals were aware of was that a color scheme for path tracing was developed to mask the identification of the
controller. The color scheme was assigned as follows: Blue- PID, Red- ANN, and Green- FL.

The collective scores of each controller’s test run through each testing environment are displayed in collaborative rubrics given in APPENDIX E: RUBRICS OF CONTROLLERS’ TESTING ENVIRONMENT RUNS. The four scores were combined to produce a single rubric per controller per test environment. Section 6.2 highlights the criteria for using the rubric to grade the controllers. The collective scores were combined in a table per controller to give a condensed look for easier comparison (Table 16, Table 17, and Table 18). The percentage score of each controller’s test run is broken down by situation displayed in these tables. An overall average for each situation is presented in the right column of each table. This data is used to generate a bar graph to compare the three controllers (Figure 69).

<table>
<thead>
<tr>
<th>Situation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>0.50</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50.00</td>
</tr>
<tr>
<td>Corridor</td>
<td>0.56</td>
<td>0.25</td>
<td>0.56</td>
<td>0.44</td>
<td>0.44</td>
<td>0.44</td>
<td>0.44</td>
<td>44.79</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hole</td>
<td>0.75</td>
<td>0.56</td>
<td>1.00</td>
<td>0.50</td>
<td>0.75</td>
<td>0.69</td>
<td>0.63</td>
<td>0.69</td>
<td>0.63</td>
<td>0.50</td>
<td>66.88</td>
</tr>
<tr>
<td>Small Object</td>
<td>0.69</td>
<td>0.69</td>
<td>0.44</td>
<td>0.56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57.81</td>
</tr>
<tr>
<td>Large Object</td>
<td>0.69</td>
<td>0.69</td>
<td>0.44</td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>61.46</td>
</tr>
<tr>
<td>Left Approach</td>
<td>0.81</td>
<td>0.63</td>
<td>1.00</td>
<td>0.75</td>
<td>0.50</td>
<td>0.50</td>
<td>0.56</td>
<td>0.63</td>
<td>0.81</td>
<td>0.50</td>
<td>66.88</td>
</tr>
<tr>
<td>Right Approach</td>
<td>0.81</td>
<td>0.63</td>
<td>0.69</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.56</td>
<td>0.63</td>
<td>0.81</td>
<td>0.50</td>
<td>62.50</td>
</tr>
<tr>
<td>Navigation</td>
<td>0.63</td>
<td>0.63</td>
<td>1.00</td>
<td>0.50</td>
<td>0.63</td>
<td>0.50</td>
<td>0.44</td>
<td>0.50</td>
<td>0.69</td>
<td>0.44</td>
<td>59.38</td>
</tr>
<tr>
<td>Intelligence</td>
<td>0.69</td>
<td>0.50</td>
<td>0.75</td>
<td>0.56</td>
<td>0.56</td>
<td>0.50</td>
<td>0.44</td>
<td>0.44</td>
<td>0.63</td>
<td>0.50</td>
<td>55.63</td>
</tr>
<tr>
<td>Total</td>
<td>0.71</td>
<td>0.56</td>
<td>0.94</td>
<td>0.59</td>
<td>0.58</td>
<td>0.52</td>
<td>0.51</td>
<td>0.54</td>
<td>0.75</td>
<td>0.48</td>
<td>58.37</td>
</tr>
</tbody>
</table>

Table 16: PID Controller’s Situational Scores
Table 17: ANN Controller’s Situational Scores

<table>
<thead>
<tr>
<th>Situation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>0.94</td>
<td>1.00</td>
<td>1.00</td>
<td>0.88</td>
<td></td>
<td></td>
<td></td>
<td>0.75</td>
<td></td>
<td></td>
<td>91.25</td>
</tr>
<tr>
<td>Corridor</td>
<td>0.38</td>
<td>0.56</td>
<td>0.94</td>
<td></td>
<td>0.94</td>
<td>0.75</td>
<td>0.88</td>
<td></td>
<td></td>
<td></td>
<td>73.96</td>
</tr>
<tr>
<td>Hole</td>
<td>0.75</td>
<td>0.38</td>
<td>0.69</td>
<td>0.69</td>
<td>0.81</td>
<td>0.94</td>
<td>0.94</td>
<td>1.00</td>
<td>0.94</td>
<td>0.88</td>
<td>80.00</td>
</tr>
<tr>
<td>Small Object</td>
<td>0.75</td>
<td>0.38</td>
<td>0.44</td>
<td>0.50</td>
<td>0.69</td>
<td>1.00</td>
<td>0.88</td>
<td>0.94</td>
<td>0.81</td>
<td>0.69</td>
<td>70.63</td>
</tr>
<tr>
<td>Large Object</td>
<td>0.38</td>
<td>0.25</td>
<td>0.56</td>
<td>0.69</td>
<td>0.94</td>
<td></td>
<td></td>
<td></td>
<td>0.69</td>
<td>0.94</td>
<td>63.39</td>
</tr>
<tr>
<td>Left Approach</td>
<td>0.63</td>
<td>0.50</td>
<td>0.44</td>
<td>0.63</td>
<td>0.94</td>
<td>0.88</td>
<td>1.00</td>
<td>1.00</td>
<td>0.75</td>
<td>0.81</td>
<td>75.63</td>
</tr>
<tr>
<td>Right Approach</td>
<td>0.50</td>
<td>0.44</td>
<td>0.63</td>
<td>0.88</td>
<td>0.94</td>
<td>1.00</td>
<td>1.00</td>
<td>0.75</td>
<td>0.88</td>
<td></td>
<td>77.78</td>
</tr>
<tr>
<td>Navigation</td>
<td>0.69</td>
<td>0.44</td>
<td>0.63</td>
<td>0.81</td>
<td>0.88</td>
<td>0.88</td>
<td>0.94</td>
<td>0.75</td>
<td>0.81</td>
<td></td>
<td>72.50</td>
</tr>
<tr>
<td>Intelligence</td>
<td>0.56</td>
<td>0.50</td>
<td>0.50</td>
<td>0.56</td>
<td>0.75</td>
<td>0.81</td>
<td>0.88</td>
<td>0.88</td>
<td>0.69</td>
<td>0.75</td>
<td>68.75</td>
</tr>
<tr>
<td>Total</td>
<td>0.68</td>
<td>0.44</td>
<td>0.50</td>
<td>0.59</td>
<td>0.84</td>
<td>0.90</td>
<td>0.92</td>
<td>0.96</td>
<td>0.76</td>
<td>0.83</td>
<td>74.88</td>
</tr>
</tbody>
</table>

Table 18: FL Controller’s Situational Scores

<table>
<thead>
<tr>
<th>Situation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>0.56</td>
<td></td>
<td>0.81</td>
<td>0.69</td>
<td>0.75</td>
<td></td>
<td></td>
<td>0.75</td>
<td></td>
<td></td>
<td>71.25</td>
</tr>
<tr>
<td>Corridor</td>
<td>1.00</td>
<td>0.94</td>
<td>0.69</td>
<td>0.75</td>
<td>0.75</td>
<td></td>
<td></td>
<td></td>
<td>0.75</td>
<td>0.75</td>
<td>80.36</td>
</tr>
<tr>
<td>Hole</td>
<td>0.88</td>
<td>0.56</td>
<td>0.94</td>
<td>0.81</td>
<td>0.88</td>
<td>0.69</td>
<td>0.81</td>
<td>0.94</td>
<td>0.81</td>
<td>0.69</td>
<td>80.00</td>
</tr>
<tr>
<td>Small Object</td>
<td>0.63</td>
<td>0.63</td>
<td>0.75</td>
<td>0.63</td>
<td>0.75</td>
<td>0.56</td>
<td>0.75</td>
<td>0.63</td>
<td>0.63</td>
<td>0.50</td>
<td>64.38</td>
</tr>
<tr>
<td>Large Object</td>
<td>1.00</td>
<td>0.56</td>
<td>0.75</td>
<td>0.81</td>
<td>0.56</td>
<td>0.81</td>
<td></td>
<td></td>
<td></td>
<td>0.50</td>
<td>71.43</td>
</tr>
<tr>
<td>Left Approach</td>
<td>0.56</td>
<td>0.69</td>
<td>0.94</td>
<td>0.81</td>
<td>0.69</td>
<td>0.81</td>
<td>0.81</td>
<td>0.75</td>
<td>0.56</td>
<td>0.69</td>
<td>73.13</td>
</tr>
<tr>
<td>Right Approach</td>
<td>0.56</td>
<td>0.69</td>
<td>0.94</td>
<td>0.81</td>
<td>0.69</td>
<td>0.81</td>
<td>0.81</td>
<td>0.75</td>
<td>0.56</td>
<td>0.69</td>
<td>73.13</td>
</tr>
<tr>
<td>Navigation</td>
<td>0.63</td>
<td>0.63</td>
<td>0.81</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.75</td>
<td>0.56</td>
<td>0.69</td>
<td>70.63</td>
</tr>
<tr>
<td>Intelligence</td>
<td>0.69</td>
<td>0.63</td>
<td>0.81</td>
<td>0.75</td>
<td>0.75</td>
<td>0.63</td>
<td>0.69</td>
<td>0.69</td>
<td>0.50</td>
<td>0.75</td>
<td>68.75</td>
</tr>
<tr>
<td>Total</td>
<td>0.74</td>
<td>0.62</td>
<td>0.86</td>
<td>0.76</td>
<td>0.76</td>
<td>0.69</td>
<td>0.77</td>
<td>0.75</td>
<td>0.63</td>
<td>0.68</td>
<td>72.56</td>
</tr>
</tbody>
</table>

Figure 69 was created to show overall trends of the controllers as they navigated throughout the testing environment. The observed controllers’ reactions to each situation are discussed at length in Section 7.3. The overall rankings based of the rubric analysis place the ANN controller with best overall performance followed closely by the FL controller.
7.3 Results Analysis
As shown in the Figure 69 and Table 16, 17 &18, the PID controller grades resulted in a third place ranking based on these scores. The ANN controller score averages range from approximately 63%-92% with best performance in the room situation and poorest performance in large object approach. The FL controller score averages range from
approximately 64%-80% with best performance in corridor navigation and poorest performance in small object approach. The PID controller score averages range from approximately 44%-66% with best performance in left angular approach and poorest performance in corridor navigation.

The situational results for the room, corridor, hole, small object, large object, left angular approach, and right angular approach situations are clearly indicated in the bar graph in Figure 69. The controllers’ traced paths through these test environments (APPENDIX D: RESULTS OF ENVIRONMENT RUNS) support these results. The traced paths are compared by situation in the observation portion of the results (Section 7.3).

The overall navigation scores (Figure 69) for the three controllers give the rankings: First-ANN, Second-FL, Third-PID. The ANN controller received multiple rubric comments on its smooth, fluid turning ability as well as only a few numbers of minor collisions. Along with the positive comments, there were also a few negative comments about navigation in certain testing environment layouts. The ANN controller was noted to experience delays in reacting to some objects as well as a comment indicating the match between the environment and controller were not a good complement from a navigation stand point. The FL controller was observed to avoid obstacles and successfully navigate most situations, but at times exhibited non-fluid movements and obvious delayed reactions. A rubric comment about the FL controller’s course through a corridor situation details successful navigation; however, the controller’s path was zigzag patterned instead of a direct straight route. The PID controller was noted to produce very rapid, sharp turning when an object was detected inside the controller’s set point. When traversing a situation the PID controller had difficulty with, the
controller’s reaction appeared confused and produced choppy non-fluid movements. It was also commented that the PID controller performed extremely inconsistently and produced excessive amounts of movements while navigating.

The overall intelligence score was formulated by each grader based on perceived controller intelligence. The rubric provides basic guidelines for evaluating intelligence, but it is the individual grader’s perception of the controllers’ actions during the run that produced the final scores. The final rankings for intelligence yielded a tie between the ANN and FL controllers for most intelligent controller. PID again placed third. There is a strong correlation between overall navigation scores and overall intelligence scores.

The run time of each controller was also noted on the rubrics. The time began when the controller was started in the testing environment and concluded when the controller reached the RFID tag goal. Cumulative run times for all ten testing environments are as follows: ANN- 31 minutes and 39 seconds, FL- 24 minutes and 31 seconds, and PID-19 minutes and 05 seconds. A pass/fail indication is found in the bottom right corner of the rubric. A pass signified the controller located the RFID tag within a reasonable time. A fail indicated the controller did not locate the RFID Tag before the completion time cap of 5 minutes. Any run exceeding 5 minutes was terminated. The ANN controller received 3 fails out of 10 runs. The FL controller received zero fails. The PID controller received 1 fail out of 10 runs. The pass/fail is not indicative of the controller’s ability to navigate the environment, but only served as a general method of time limitation.

Table 16, 17, and 18 are used to generate an additional bar graph indicating the overall controller performance score per testing environment (Figure 70).
Figure 70: Average Scores per Controller by Environment

Figure 70 is presented to show general trends on a testing environment level. The ANN controller appears to significantly outperform the FL and PID controllers in 4 out of 10 testing environments, but also marginally produced the poorest performance in 4 out of 10 testing environments. The FL controller shows great general consistency throughout all testing environments. The PID controller, while ranking third in all situational categories still produced a leading run in testing environment 3 as well as outperforming the top ranking ANN controller in 4 out of 10 testing environments.

7.4 Robot’s Navigation Testing Observation Results
This section of the results uses the three controllers’ traced paths to give observational comparison results. Specific situations within the testing environments are highlighted and discussed. Complete diagrams of the controllers’ paths per testing environment are
found in APPENDIX D: RESULTS OF ENVIRONMENT RUNS. Just as a reminder, the PID controller path is blue, the ANN controller path is red, and the FL controller path is green.

Figure 71: Room Situation Excerpt from Test Environment #6

Figure 71 exhibits the paths of the PID, ANN, and FL controllers navigating a room situation in testing environment 6. The PID controller enters the situation with erratic behavior then produces a smooth transition to the left portion of the test environment before encountering the RFID tag. The FL controller encounters the room situation with smooth beginnings, shows course redirection when approaching the walls and produces an exaggerated zigzag pattern within. The ANN controller shows a smooth fluid navigation path throughout the room portion of test environment 6.
Figure 72: Corridor Situation Excerpt from Test Environment #1

Figure 72 highlights the comparison in corridor navigation abilities between the PID and FL controllers. The PID controller shows excessive movements throughout this section as well as a major collision as the mobile robot platform went directly through the wall. The FL controller clearly shows fluid navigation within the corridor situation and remains collision free.

Figure 73: Corridor Situation Excerpt from Test Environment #8
Figure 73 shows the ANN controller giving a fluid navigation performance and smooth turning transitions as the controller encounters the walls of the corridor. The numerous red lines show every pass the ANN controller made in the corridor throughout the entire run, and all navigations were collision free. The PID controller shows obvious hardship with this situation and appears to ignore the right corridor wall. Once the PID controller reacts, the loops shown indicate sharp turns away from the detected object.

![Diagram of navigation](image)

**Figure 74: Corridor Situation Excerpt from Test Environment #9**

The apparent zigzag pattern of the FL controller in Figure 74 shows a characteristic performance of this controller in corridor situation. While the zigzag pattern is not the most direct route, the controller managed to navigate the mobile robot platform through this situation without collisions. The ANN controller shows an effortless path through the corridor and an appropriate reaction to the wall on the right. Both controllers here exhibit successful navigation with consistency and ease.
Figure 75: Hole Situation Excerpt from Test Environment #8

Figure 75 shows many hole situations exist for the controller to navigate the mobile robot platform through. All three controllers navigate the three holes present within this excerpt. Starting with the hole between the small triangle and small square, clearly the ANN controller path appears the most direct and collision free. Here the FL controller shows two paths through this space: one smooth and collision-free and the other is more non-fluid and results in a small collision with the square. The PID controller also navigated this situation twice, once producing optimal results, the other showing major collision as the PID controller doesn’t sense or disregards the small square and runs directly through the object. The hole situation between the small square and the jutting
wall on the left was successfully navigated by all controllers, but the ANN controller path appears to look effortless and shows obvious anticipation of the approaching objects and situation. The third hole situation within Figure 75 is found between the left jutting wall and the small rectangle. The ANN and FL controllers both exhibit fluid navigation, and the PID controller displays sharp turns and also results in a major collision with the rectangle.

![Diagram of small object approach](image)

**Figure 76: Small Object Approach Excerpt from Test Environment #1**

Approaching and avoiding the small rectangle present in the Figure 76 excerpt is a task all three controllers took on. The PID controller’s sharp quick movement away from the wall on the left did not leave it with enough time to anticipate the rectangle before encountering a collision with it. Only after the collision begins, does the PID controller start to change course direction. The FL controller approaches the small rectangle from two different directions. The FL controller’s paths on the left show ease of movement and clear object approach anticipation resulting in slight redirection. The FL controller’s path originating in the bottom right of this excerpt shows the controller approaching the
object, but a delayed reaction to the impending situation producing a sharper turn to avoid major collision. The ANN controller once again navigates the situation in an unproblematic and efficient manner showing object anticipation and collision free movements.

![Figure 77: Large Object Approach Excerpt from Test Environment #9](image)

Successful navigation around the large object in Figure 77 is completed by all three controllers. The ANN and FL controllers experience no difficulty traversing around or away from the object, and the smooth lines of their paths depict ease of movement and transition. The PID controller, while successfully navigating around this large object, clearly shows sharp turns away from the object and produces a less fluid looking path.
CHAPTER 8
CONCLUSIONS, FUTURE WORK AND SUMMARY

8.1 Conclusion
Comparing the MATLAB® simulation results to the physical implementation results yields similar performance evaluations. The rankings for both place the Artificial Neural Networks (ANN) controller with the best overall performance, followed closely by the Fuzzy Logic (FL) controller. The PID controller was the poorest performer in both simulation and physical implementation. The simulation results (Figure 68) show the ANN and FL controllers function very similarly and the PID controller lags behind these two intelligent controllers. The physical implementation support the simulation results by showing overall rubric scores of ANN = 74.88%, FL = 72.56%, and PID = 58.37% (Table 17, Table 18, and Table 16).
The PID controller actions were consistent throughout the 10 testing environments; however, these actions were not optimal for this type of unknown environment navigation. Overall the controller produced excessive amounts of movements, choppy non-fluid reactions, and rapid transitions in course redirection. The reactions generated were an over compensation to a situation. The PID controller was not able to avoid major collisions in multiple test environments. Although the controller quickly completed most test runs, this completion time did not coincide with a more efficient controller.
Comparing the ANN controller’s performances throughout the ten testing environments, it appears to show some inconsistency. It is unclear if the controller is inconsistent, or if the testing environment layouts were a factor is producing this trend. Although these inconsistencies exist, it was not a deterrent in ranking this as the top performing
controller overall. When this controller exhibited optimal performance, the navigation through the environments was slow and steady. This allowed the controller appropriate time to anticipate upcoming situations and react with enough time to avoid major collisions. The overall motions of this controller when producing optimal behavior is described as smooth and fluid. In the environments or situations this controller received poor reviews; it was noted to experience reaction delays, minor collisions, and numerous hesitations.

The FL controller was shown to be the most consistent performer in situational analysis as well as environment layout comparison. A generalized description of its navigational movements is semi-fluid transition abilities largely accompanied by zigzag pattern movements. The FL controller displayed an overall delay in reaction as it approached objects and led to numerous minor collisions. These collisions did not affect the controller’s ability to successfully navigate throughout the unknown testing environments.

8.2 Future Work
Future research with the three controllers would involve developing more complex implementations for the controllers, more time invested in tuning and training the controller designs to elicit more optimal responses from each controller. Increasing the complexity of the design of each controller by allowing more variables to be controlled: speed, turning angle and others. Increase the number of neurons to the ANN and membership variables to the FL which may result in more intelligent navigation.
8.3 Summary
This thesis covered all aspects of controller design, simulation, implementation, and testing. Two intelligent controllers, ANN and FL, were compared to a traditional industry standard controller, PID. The comparison was performed through MATLAB® simulation, Quartus II® hardware design and simulation, and an Altera FPGA card hardware implementation for comparison in a real unknown indoor environment. The results of this research generated three successful controllers. The ANN and FL controllers were definitively superior to the PID controller in overall navigation and intelligence.
REFERENCES


Rios-Gutiérrez, F. (2000). A control scheme based on decomposition to primitive behaviors and a reinforcement learning technique. Tulane University, New Orleans, LA.


APPENDIX A: ROBOTIC BASE IMPLEMENTATION DETAIL

Figure 78: Top Side of the Lower Level of Mobile Robot Platform

Figure 79: Bottom Side of the Lower Level of Mobile Robot Platform
Figure 80: Bottom Side of the Upper Level of Mobile Robot Platform
APPENDIX B: CONTROLLERS’ BASIC STAMP CODE

PID BASIC Stamp Code

'==============================================================================
' File....... PID.bs2e
' Purpose.... Robot Navigation
' {$STAMP BS2e}
' {$PBASIC 2.5}
'==============================================================================

' -----[ Constants ]----------------------------------------------------------
'PID
SetPoint  CON  30  ' Set point

'Sonar Sensors
Constant  CON  2257

'RFID
T2400  CON  396
Baud  CON  T2400
LastTag  CON  3

' -----[ I/O Definitions ]----------------------------------------------------
Enable  PIN  13  ' low = reader on
RX  PIN  14  ' serial from reader

' -----[ Variables ]----------------------------------------------------------
'PID
error  VAR  Byte  ' Error value of setpoint-feedback
Value  VAR  Word  ' Motor output value

'Sonar Sensors
Sonar  VAR  Nib
time  VAR  Word  ' Sonar time in msec.
sensorInput  VAR  Byte  ' Sensor input variable
L_sensor_value  VAR  Word  ' Left sensor value
L_side_value  VAR  Word  ' Left side sensors add together value
R_sensor_value  VAR  Word  ' Right sensor value
R_side_value  VAR  Word  ' Right side sensors add together value

'RFID
buf  VAR  Byte(10)  ' RFID bytes buffer
tagNum  VAR  Nib  ' from EEPROM table
idx  VAR  Byte  ' tag byte index

' -----[ Program Code ]-------------------------------------------------------
'Main Program
Main:

' Set both right and left sensor values to zero
R_side_value = 0
L_side_value = 0

' Read sonar sensor distance value
FOR Sonar = 11 TO 4
PULSOUT Sonar, 5  ' Enable sonar sensor to measure distance
PULSIN Sonar, 1, time  ' Read in sonar distance
IF Sonar = 11 THEN
R_sensor_value = ((Constant ** time)) MIN 0 MAX 15  ' Calc. the weighted sonar distance
R_side_value = R_side_value + R_sensor_value  ' Add weighted sonar value to right side total
ELSEIF Sonar = 10 THEN
R_sensor_value = (25*(Constant ** time))/100 MIN 0 MAX 7  ' Calc. the weighted sonar distance
R_side_value = R_side_value + R_sensor_value  'Add weighted sonar value to right side total

ELSEIF Sonar = 9 THEN
R_sensor_value = (15*(Constant ** time))/100 MIN 0 MAX 5 'Calc. the weighted sonar distance
R_side_value = R_side_value + R_sensor_value  'Add weighted sonar value to right side total

ELSEIF Sonar = 8 THEN
R_sensor_value = (10*(Constant ** time))/100 MIN 0 MAX 3 'Calc. the weighted sonar distance
R_side_value = R_side_value + R_sensor_value  'Add weighted sonar value to right side total

ELSEIF Sonar = 7 THEN
L_sensor_value = (10*(Constant ** time))/100 MIN 0 MAX 3 'Calc. the weighted sonar distance
L_side_value = L_side_value + L_sensor_value  'Add weighted sonar value to left side total

ELSEIF Sonar = 6 THEN
L_sensor_value = (15*(Constant ** time))/100 MIN 0 MAX 5 'Calc. the weighted sonar distance
L_side_value = L_side_value + L_sensor_value  'Add weighted sonar value to left side total

ELSEIF Sonar = 5 THEN
L_sensor_value = (25*(Constant ** time))/100 MIN 0 MAX 7 'Calc. the weighted sonar distance
L_side_value = L_side_value + L_sensor_value  'Add weighted sonar value to left side total

ELSEIF Sonar = 4 THEN
L_sensor_value = ((Constant ** time)) MIN 0 MAX 15 'Calc. the weighted sonar distance
L_side_value = L_side_value + L_sensor_value  'Add weighted sonar value to left side total
ENDIF
NEXT

R_side_value(error) = SetPoint - R_side_value  'Calculate error of right side of weighted sonar value feedback.
L_side_value(error) = SetPoint - L_side_value  'Calculate error of left side of weighted sonar value feedback.

R_side_value(error) = (50-((R_side_value(error) * 1666)/1000)) 'Convert error calc. to unit-less value error
L_side_value(error) = (((L_side_value(error) * 1666)/1000)+50) 'Convert error calc. to unit-less value error

sensorInput = ((R_side_value(error) + L_side_value(error))/2) 'Take the average of both side unit-less value error

PULSOUT 0, sensorInput*500  'Send out Pulse to FPGA equal to average unit-less value error

'Load Register on FPGA with average unit-less value error
HIGH 15
PAUSE 10
LOW 15

PULSIN 12,1, Value  'Read in pulse from FPGA
Value = Value/500  'Convert pulse to value to use to send motor speeds to motor driver

'CLEAR counters on FPGA
HIGH 3
PAUSE 10
LOW 3

'Send out Control Signals for the right & left motors
IF Value>50 THEN  'If value is greater than 50 turn robotic mobile platform to the right
R_side_value = 990-((68*Value)/10) 'Calc motor speed for right motor
PULSOUT 7,R_side_value  'Motor speed sent to motor driver for right motor
IF Value >= 64 THEN  'If value is greater than 64 turn robotic mobile platform to the right quickly
PULSOUT 8,800  'Motor speed sent to motor driver for left motor
ELSE
PULSOUT 8,675  'Motor speed sent to motor driver for left motor
ENDIF
ELSEIF Value<50 THEN  'If value is less than 50 turn robotic mobile platform to the left
L_side_value = ((68*Value)/10)+309 'Calc motor speed for Left motor
PULSOUT 8,L_side_value  'Motor speed sent to motor driver for left motor
IF Value <= 36 THEN  'If value is less than 36 turn robotic mobile platform to the left quickly
PULSOUT 7,800  'Motor speed sent to motor driver for right motor
ELSE
PULSOUT 7,675  'Motor speed sent to motor driver for right motor
ENDIF
ELSEIF Value = 50 THEN  'If value is equal to 50 make robotic mobile platform go straight
PULSOUT 7, 675  'Motor speed sent to motor driver for right motor

PULSOUT 8, 675  'Motor speed sent to motor driver for left motor
ENDIF

'RFID tag reader
'ReadTag
LOW Enable
SERIN RX, T2400, 200, Main,[WAIT($0A),STR buf:10]
' activate the reader
' wait for Tag ID, if tag found stop robotic mobile platform
' If tag not found goto beginning of program to start again

DO
  PULSOUT 7, 735  'Right motor stopped
  PULSOUT 8, 735  'Left motor stopped
LOOP
Artificial Neural Network BASIC Stamp Code

' ============================================================================
' File....... Artificial Neural Network.bs2e
' Purpose.... Robot Navigation
' {$STAMP BS2e}
' {$PBASIC 2.5}
' ============================================================================

' ----[ Constants ]-----------------------------------------------------------
'ANN
SetPoint   CON    30  ' Set point

'Sonar Sensors
Constant   CON    2257

'RFID
T2400     CON    396
Baud       CON    T2400
LastTag    CON    3

' ----[ I/O Definitions ]-----------------------------------------------------
Enable     PIN    13  ' low = reader on
RX         PIN    14  ' serial from reader

' ----[ Variables ]-----------------------------------------------------------
'ANN
error      VAR    Byte  ' Error value of setpoint-feedback
Value      VAR    Word  ' Motor output value

'Sonar Sensors
Sonar      VAR    Nib
Time       VAR    Word  ' Sonar time in msec.
sensorInput VAR    Byte  ' Sensor input variable

L_sensor_value   VAR    Word  ' Left sensor value
L_side_value     VAR    Word  ' Left side sensors add together value

FOR Sonar = 11 TO 4
PULSOUT Sonar, 5
PULSIN Sonar, 1, time 'Enable sonar sensor to measure distance
IF Sonar = 11 THEN
R_sensor_value = ((Constant ** time)) MIN 0 MAX 15 'Calc. the weighted sonar distance
R_side_value = R_side_value + R_sensor_value  'Add weighted sonar value to right side total
ELSEIF Sonar = 10 THEN
R_sensor_value = (25*(Constant ** time))/100 MIN 0 MAX 7  'Calc. the weighted sonar distance
R_side_value = R_side_value + R_sensor_value  'Add weighted sonar value to right side total
ELSEIF Sonar = 9 THEN
R_sensor_value = (15*(Constant ** time))/100 MIN 0 MAX 5  'Calc. the weighted sonar distance

ENDIF
R_side_value = R_side_value + R_sensor_value  'Add weighted sonar value to right side total

ELSEIF Sonar = 8  THEN
R_sensor_value = (10*(Constant ** time))/100 MIN 0 MAX 3  'Calc. the weighted sonar distance
R_side_value = R_side_value + R_sensor_value  'Add weighted sonar value to right side total

ELSEIF Sonar = 7  THEN
L_sensor_value = (10*(Constant ** time))/100 MIN 0 MAX 3  'Calc. the weighted sonar distance
L_side_value = L_side_value + L_sensor_value  'Add weighted sonar value to left side total

ELSEIF Sonar = 6  THEN
L_sensor_value = (15*(Constant ** time))/100 MIN 0 MAX 3  'Calc. the weighted sonar distance
L_side_value = L_side_value + L_sensor_value  'Add weighted sonar value to left side total

ELSEIF Sonar = 5  THEN
L_sensor_value = (25*(Constant ** time))/100 MIN 0 MAX 3  'Calc. the weighted sonar distance
L_side_value = L_side_value + L_sensor_value  'Add weighted sonar value to left side total

ELSEIF Sonar = 4  THEN
L_sensor_value = ((Constant ** time))/100 MIN 0 MAX 3  'Calc. the weighted sonar distance
L_side_value = L_side_value + L_sensor_value  'Add weighted sonar value to left side total

ENDIF

NEXT

R_side_value(error) = SetPoint - R_side_value  'Calculate error of right side of weighted sonar value feedback.
L_side_value(error) = SetPoint - L_side_value  'Calculate error of left side of weighted sonar value feedback.

PULSOUT 0, R_side_value*500  'Send out Pulse to FPGA equal to right side value error
PULSOUT 1, L_side_value*500  'Send out Pulse to FPGA equal to left side value error

'Load Register on FPGA with average unit-less value error
HIGH 15
PAUSE 10
LOW 15

PULSIN 12,1, Value  'Read in pulse from FPGA
Value = Value/500  'Convert pulse to value to use to send motor speeds to motor driver

'Clear counters on FPGA
HIGH 3
PAUSE 10
LOW 3

'Send out Control Signals for the right & left motors
IF Value>50 THEN
   R_side_value = 990-((68*Value)/10)  'Calc motor speed for right motor
   PULSOUT 7,R_side_value  'Motor speed sent to motor driver for right motor
   IF Value >= 64 THEN
      PULSOUT 8,800  'Motor speed sent to motor driver for left motor
   ELSE
      PULSOUT 8,675  'Motor speed sent to motor driver for left motor
   ENDIF
ELSEIF Value=50 THEN
   L_side_value = ((68*Value)/10)+309  'Calc motor speed for Left motor
   PULSOUT 8,L_side_value  'Motor speed sent to motor driver for left motor
   IF Value <= 36 THEN
      PULSOUT 7,800  'Motor speed sent to motor driver for right motor
   ELSE
      PULSOUT 7,675  'Motor speed sent to motor driver for right motor
   ENDIF
ELSEIF Value<50 THEN
   ELSEIF Value = 50 THEN
      PULSOUT 7, 675  'Motor speed sent to motor driver for right motor
      PULSOUT 8, 675  'Motor speed sent to motor driver for left motor
      ENDIF
ELSEIF Value<50 THEN
      M_side_value = ((68*Value)/10)+309  'Calc motor speed for Left motor
      PULSOUT 8,M_side_value  'Motor speed sent to motor driver for left motor
      IF Value <= 36 THEN
         PULSOUT 7,800  'Motor speed sent to motor driver for right motor
      ELSE
         PULSOUT 7,675  'Motor speed sent to motor driver for right motor
      ENDIF
ELSEIF Value = 50 THEN
      PULSOUT 7, 675  'Motor speed sent to motor driver for right motor
      PULSOUT 8, 675  'Motor speed sent to motor driver for left motor
      ENDIF

'RFID tag reader
'ReadTag
LOW Enable  ' activate the reader
SERIN RX, T2400, 200, Main, [WAIT($0A),STR buf10]  ' wait for Tag ID, if tag found stop robotic mobile platform
   ' If tag not found goto beginning of program to start again
DO
  PULSOUT 7, 735
  PULSOUT 8, 735
LOOP

'Right motor stopped
'Left motor stopped
Fuzzy Logic BASIC Stamp Code

'==========================================================================
' File....... Fuzzy_Logic.bs2e
' Purpose.... Robot Navigation
' {$STAMP BS2e}
' {$PBASIC 2.5}
'==========================================================================

'-----[ Constants ]--------------------------------------------------------
'Sonar Sensors
Constant CON 2257

'RFID
T2400 CON 396
Baud CON T2400
LastTag CON 3

'-----[ I/O Definitions ]--------------------------------------------------
Enable PIN 13    ' low = reader on
RX PIN 14       ' serial from reader

'-----[ Variables ]-------------------------------------------------------
Value VAR Word    'Motor output value

'Sonar Sensors
Sonar VAR Nib     'Sonar time in msec.
time VAR Word

L_sensor_value VAR Word    'Left sensor value
L_side_value VAR Word     'Left side sensors add together value

R_sensor_value VAR Word    'Right sensor value
R_side_value VAR Word     'Right side sensors add together value

'RFID
buf VAR Byte(10)    ' RFID bytes buffer
tagNum VAR Nib      ' from EEPROM table
idx VAR Byte        ' tag byte index

'-----[ Program Code ]-----------------------------------------------------
'Main Program
Main:

'Set both right and left sensor values to zero
R_side_value = 0
L_side_value = 0

'Read sonar sensor distance value
FOR Sonar = 11 TO 4
PULSOUT Sonar, 5       'Enable sonar sensor to measure distance
PULSIN Sonar, 1, time   'Read in sonar distance
IF Sonar = 11 THEN
R_sensor_value = ((Constant ** time)) MIN 0 MAX 15    'Calc. the weighted sonar distance
R_side_value = R_side_value + R_sensor_value    'Add weighted sonar value to right side total
ELSEIF Sonar = 10 THEN
R_sensor_value = (25*(Constant ** time))/100 MIN 0 MAX 7    'Calc. the weighted sonar distance
R_side_value = R_side_value + R_sensor_value    'Add weighted sonar value to right side total
ELSEIF Sonar = 9 THEN
R_sensor_value = (15*(Constant ** time))/100 MIN 0 MAX 5    'Calc. the weighted sonar distance
R_side_value = R_side_value + R_sensor_value    'Add weighted sonar value to right side total
ELSEIF Sonar = 8 THEN
R_sensor_value = (10*(Constant ** time))/100 MIN 0 MAX 3    'Calc. the weighted sonar distance
R_side_value = R_side_value + R_sensor_value    'Add weighted sonar value to right side total


}{
ELSEIF Sonar = 7 THEN
L_sensor_value = (10*(Constant ** time))/100 MIN 0 MAX 3  'Calc. the weighted sonar distance
L_side_value = L_side_value + L_sensor_value         'Add weighted sonar value to left side total
ELSEIF Sonar = 6 THEN
L_sensor_value = (15*(Constant ** time))/100 MIN 0 MAX 5  'Calc. the weighted sonar distance
L_side_value = L_side_value + L_sensor_value         'Add weighted sonar value to left side total
ELSEIF Sonar = 5 THEN
L_sensor_value = (25*(Constant ** time))/100 MIN 0 MAX 7  'Calc. the weighted sonar distance
L_side_value = L_side_value + L_sensor_value         'Add weighted sonar value to left side total
ELSEIF Sonar = 4 THEN
L_sensor_value = ((Constant ** time)) MIN 0 MAX 15  'Calc. the weighted sonar distance
L_side_value = L_side_value + L_sensor_value         'Add weighted sonar value to left side total
ENDIF
NEXT
PULSOUT 0, R_side_value*500  ' Send out Pulse to FPGA equal to right side value
PULSOUT 1, L_side_value*500  ' Send out Pulse to FPGA equal to left side value

'Load Register on FPGA with average unit-less value error
HIGH 15
PAUSE 10
LOW 15
PULSIN 12,1, Value         'Read in pulse from FPGA
Value = Value/500           'Convert pulse to value to use to send motor speeds to motor driver

'Clear counters on FPGA
HIGH 3
PAUSE 10
LOW 3

'Send out Control Signals for the right & left motors
IF Value>50 THEN           'If value is greater than 50 turn robotic mobile platform to the right
R_side_value = 990-((68*Value)/10)  'Calc motor speed for right motor
PULSOUT 7,R_side_value   'Motor speed sent to motor driver for right motor
IF Value >= 64 THEN       'If value is greater than 64 turn robotic mobile platform to the right quickly
PULSOUT 8,800            'Motor speed sent to motor driver for left motor
ELSE
PULSOUT 8,675            'Motor speed sent to motor driver for left motor
ENDIF
ELSEIF Value<50 THEN     'If value is less than 50 turn robotic mobile platform to the left
L_side_value = ((68*Value)/10)+309  'Calc motor speed for Left motor
PULSOUT 8,L_side_value   'Motor speed sent to motor driver for left motor
IF Value <= 36 THEN       'If value is less than 36 turn robotic mobile platform to the left quickly
PULSOUT 7,800            'Motor speed sent to motor driver for right motor
ELSE
PULSOUT 7,675            'Motor speed sent to motor driver for right motor
ENDIF
ELSEIF Value = 50 THEN    'If value is equal to 50 make robotic mobile platform go straight
PULSOUT 7, 675           'Motor speed sent to motor driver for right motor
PULSOUT 8, 675           'Motor speed sent to motor driver for left motor
ENDIF

'RFID tag reader
'REadTag
LOW Enable               ' activate the reader
SERIN RX, T2400, 200, Main,[WAIT($0A),STR buf 10]  ' wait for Tag ID, if tag found stop robotic mobile platform
    ' If tag not found goto beginning of program to start again
DO
    PULSOUT 7, 735       'Right motor stopped
    PULSOUT 8, 735       'Left motor stopped
LOOP
APPENDIX C: TESTING & EVALUATION ENVIRONMENT CONFIGURATIONS

Figure 81: Testing Environment #1

Figure 82: Testing Environment #2
Figure 83: Testing Environment #3

Figure 84: Testing Environment #4
Figure 85: Testing Environment #5

Figure 86: Testing Environment #6
Figure 87: Testing Environment #7

Figure 88: Testing Environment #8
Figure 89: Testing Environment #9

Figure 90: Testing Environment #10
APPENDIX D: RESULTS OF ENVIRONMENT RUNS

Figure 91: Controllers’ Run through Testing Environment #1
Figure 92: Controllers’ Run through Testing Environment #2
Figure 93: Controllers' Run through Testing Environment #3
Figure 94: Controllers’ Run through Testing Environment #4
Figure 95: Controllers’ Run through Testing Environment #5
Figure 96: Controllers' Run through Testing Environment #6
Figure 97: Controllers’ Run through Testing Environment #7
Figure 98: Controllers’ Run through Testing Environment #8
Figure 99: Controllers’ Run through Testing Environment #9
Figure 100: Controllers’ Run through Testing Environment #10
### APPENDIX E: RUBRICS OF CONTROLLERS’ TESTING ENVIRONMENT RUNS

**Situational Rubric for PID, ANN, and FL controllers navigating a real unknown indoor environment**

Evaluator: Overall

Robot Marker Color: Blue- PID

Environment Layout #: 1

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Small Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Large Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Left</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Right</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score</td>
</tr>
<tr>
<td><strong>Navigation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Intelligence</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Comments:**
- Corridor: not very fluid, but no collisions. Straight on collisions not so good, better with small objects
- Robot seemed to struggle with objects directly in front of it, overall good navigation, but had collisions with wall
- Did not collide in the corridor or hole situations, but exhibited non fluid movement through quick/choppy turns, led to decent performance overall
- Had many non-fluid movements when going to through the corridor

Time: 1 minute 10 seconds

Pass ✓ Fail
### Situational Rubric for PID, ANN, and FL controllers

**navigating a real unknown indoor environment**

**Evaluator:** Overall  
**Robot Marker Color:** Blue - PID  
**Environment Layout #:** 2

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>8/16</td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td>4/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td>9/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Small Object</strong></td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td></td>
<td></td>
<td>11/16</td>
</tr>
<tr>
<td><strong>Large Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Left</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Right</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall Navigation</strong></td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>10/16</td>
</tr>
<tr>
<td><strong>Overall Intelligence</strong></td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td>8/16</td>
</tr>
</tbody>
</table>

**Comments:**
- Avoided corridor
- Trouble in corners and small areas of room
- Many hole situations within this environment, some successful navigations, some more collisions
- Some major hits

**Total:** 81/144

**Time:** 3 minutes 22 seconds  
**Pass:** ✓  
**Fail:**
**Situational Rubric for PID, ANN, and FL controllers**

**Navigating a real unknown indoor environment**

**Evaluator:** Overall  
**Robot Marker Color:** Blue - PID  
**Environment Layout #:** 3

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Small Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Large Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Left</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Right</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall Navigation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall Intelligence</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Comments:**
Intelligence= did not anticipate situations or object to qualify for a perfect score for overall intelligence, but was able to quickly provide correct response once situation encountered  
Quick Luck!

Time: 0 minutes, 16 seconds  
Pass ✓ Fail

16/16
Situation Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

Evaluator: Overall
Robot Marker Color: Blue - PID
Environment Layout #: 4

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>8/16</td>
</tr>
<tr>
<td>Corridor</td>
<td>Hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small Object</td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td>7/16</td>
</tr>
<tr>
<td>Large Object</td>
<td>Angular Approach Left</td>
<td>Angular Approach Right</td>
<td>Overall</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall Navigation</td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>8/16</td>
</tr>
<tr>
<td>Overall Intelligence</td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td>9/16</td>
</tr>
<tr>
<td>Total:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>66/112</td>
</tr>
</tbody>
</table>

Comments:
Small objects= multiple approaches within this environment, one approach should be the poorest score, but the majority of small object approaches the robot tried to navigate around
Choppy quick movements, especially when having difficulty with a situation, but fluid movements otherwise

Time: 1 minute 07 seconds
Pass ✗ Fail
### Situational Rubric for PID, ANN, and FL controllers

**navigating a real unknown indoor environment**

**Evaluator:** Overall

**Robot Marker Color:** Blue

**Environment Layout #:** 5

<table>
<thead>
<tr>
<th>Situation</th>
<th>Room</th>
<th>Corridor</th>
<th>Hole</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>12/16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Situation</th>
<th>Small Object</th>
<th>Large Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
</tr>
<tr>
<td>Collides with object with no attempt to navigate around</td>
<td>9/16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Situation</th>
<th>Angular Approach Left</th>
<th>Angular Approach Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
</tr>
<tr>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>10/16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Overall</th>
<th>Overall Navigation</th>
<th>Overall Intelligence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
</tr>
<tr>
<td>Shows no ability to react to situations</td>
<td>9/16</td>
<td></td>
</tr>
</tbody>
</table>

**Total:** 56/96

**Comments:**

**Time:** 1 minute, 09 seconds

**Pass** ✓  **Fail**
### Situational Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

**Evaluator:** Overall  
**Robot Marker Color:** Blue- PID  
**Environment Layout #:** 6

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td><strong>8/16</strong></td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td>9/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td>11/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Small Object</strong></td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td><strong>7/16</strong></td>
</tr>
<tr>
<td><strong>Large Object</strong></td>
<td>8/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Left</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Right</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall Navigation</strong></td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td><strong>8/16</strong></td>
</tr>
<tr>
<td><strong>Overall Intelligence</strong></td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td><strong>8/16</strong></td>
</tr>
</tbody>
</table>

**Total:** 67/128

**Comments:**
Reaction time so quick to object detected, that seems like it doesn't have enough time to take in new sensor readings before avoiding collision in area turning into (away from original obstacle)

**Time:** 1 minute, 33 seconds  
**Pass ✔**  
**Fail**
### Situational Rubric for PID, ANN, and FL controllers

Navigating a real unknown indoor environment

**Evaluator: Overall**  
**Robot Marker Color:** Blue- PID  
**Environment Layout #:** 7

<table>
<thead>
<tr>
<th>Situation</th>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>3</td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
</tr>
<tr>
<td>Corridor</td>
<td>1</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
</tr>
<tr>
<td>Hole</td>
<td>7/16</td>
<td>Avoids situation or unable to successfully navigate through</td>
</tr>
<tr>
<td>Small Object</td>
<td>8/16</td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
</tr>
<tr>
<td>Large Object</td>
<td>8/16</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
</tr>
<tr>
<td>Angular Approach Left</td>
<td>9/16</td>
<td>Collides with object while attempting to navigate around</td>
</tr>
<tr>
<td>Angular Approach Right</td>
<td>9/16</td>
<td>Collides with object with no attempt to navigate around</td>
</tr>
</tbody>
</table>

**Overall Navigation**  
**Score:** 7/16

**Overall Intelligence**  
**Score:** 7/16

**Total:** 65/128

**Comments:**
Extremely inconsistent!
Choppy excessive movements
Appears to have moments of ‘clarity’, but confusion in the next moment
Controller able to navigate platform, but not in a way I would term efficient or successful

**Time:** greater than 5 minutes = run termination

**Pass**   
**Fail ✓**
**Situational Rubric for PID, ANN, and FL controllers**

navigating a real unknown indoor environment

**Evaluator:** Overall  
**Robot Marker Color:** Blue- PID  
**Environment Layout #:** 8

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>7/16</td>
</tr>
<tr>
<td>Corridor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11/16</td>
</tr>
<tr>
<td>Hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small Object</td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td>8/16</td>
</tr>
<tr>
<td>Large Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angular Approach Left</td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>8/16</td>
</tr>
<tr>
<td>Angular Approach Right</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10/16</td>
</tr>
<tr>
<td>Overall</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score</td>
</tr>
<tr>
<td>Navigation</td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td>7/16</td>
</tr>
<tr>
<td>Overall Intelligence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Total: 61/112</td>
</tr>
</tbody>
</table>

**Comments:**
Inconsistent with small objects- went through 2 objects, but went perfectly around 1 at a later point
Turning- seems to turn so rapidly that it overshoots and produces a turn between 180-270 degrees

Time: 3 minutes, 21 seconds  
Pass ✔  
Fail
## Situational Rubric for PID, ANN, and FL controllers

navigating a real unknown indoor environment

**Evaluator:** Overall  
**Robot Marker Color:** Blue- PID  
**Environment Layout #:** 9

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corridor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angular</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angular</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approach Right</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

### Overall Navigation
- Traverse through testing environment fluidly, with no difficulty, and collision free
- Increased effort in fluid movement with minimal collisions
- Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects
- Shows difficulty moving through the environment, and unable to avoid collision

### Overall Intelligence
- Shows ability to adapt and learn, shows improved performance throughout run
- Shows ability to anticipate and react to situations well before approach
- Reacts to situation, but has inconsistent reaction upon approach
- Shows no ability to react to situations

Total: 84/112

**Comments:**

Does poorly in with hole situation into a corner of the environment, the rapid turns of the platform get the robot "stuck" in the situation.

Time: 1 minute, 14 seconds  
Pass ✔  
Fail
Situational Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

Evaluator: Overall  Robot Marker Color:  Blue- PID
Environment Layout #:  10

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>7/16</td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Small Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Large Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Left</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Right</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score</td>
</tr>
<tr>
<td><strong>Overall Navigation</strong></td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>7/16</td>
</tr>
<tr>
<td><strong>Overall Intelligence</strong></td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td>8/16</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>61/128</td>
</tr>
</tbody>
</table>

Comments:
Unable to navigate a corridor without taking down a wall of the corridor in the process

Time: 1 minute, 53 seconds  Pass ✓  Fail
Situational Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

Evaluator: Overall
Robot Marker Color: Red- ANN
Environment Layout #: 1

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corridor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12/16</td>
</tr>
<tr>
<td>Large Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angular Approach Left</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10/16</td>
</tr>
<tr>
<td>Angular Approach Right</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall Navigation</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score 11/16</td>
</tr>
<tr>
<td>Overall Intelligence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9/16</td>
</tr>
</tbody>
</table>

Comments:

Time: 0 minutes, 18 seconds
Pass ✔ Fail
Situational Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

Evaluator: Overall

Robot Marker Color: Red - ANN
Environment Layout #: 2

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>6/16</td>
</tr>
<tr>
<td>Corridor</td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td>6/16</td>
</tr>
<tr>
<td>Hole</td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>7/16</td>
</tr>
<tr>
<td>Small Object</td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td>8/16</td>
</tr>
<tr>
<td>Large Object</td>
<td>Angular Approach Left</td>
<td>Angular Approach Right</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angular Approach Left</td>
<td>8/16</td>
<td>8/16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall Navigation</td>
<td>Overall Intelligence</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall Navigation</td>
<td>7/16</td>
<td>8/16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comments:</td>
<td>Maybe not the correct environment for this controller</td>
<td>Angular approaches- fluid movements with attempt at redirection, noticeable delay before reaction though</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time:</td>
<td>greater than 5 minutes=run termination</td>
<td>Pass</td>
<td>Fail ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>49/112</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Situational Rubric for PID, ANN, and FL controllers

navigating a real unknown indoor environment

**Evaluator: Overall**  
**Robot Marker Color:** Red - ANN  
**Environment Layout #:** 3

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td></td>
<td>15/16</td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Small Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Large Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Left</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Right</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score</td>
</tr>
<tr>
<td><strong>Overall Navigation</strong></td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>7/16</td>
</tr>
<tr>
<td><strong>Overall Intelligence</strong></td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td>8/16</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>72/144</td>
</tr>
</tbody>
</table>

**Comments:**  
Room situation navigation- had minor collisions, but a grade of 4 makes sense

**Time:** greater than 5 minutes=run termination  
**Pass**  
**Fail ✓**
Situational Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

Evaluator: Overall  Robot Marker Color: Red - ANN
Environment Layout #: 4

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>9/16</td>
</tr>
<tr>
<td>Corridor</td>
<td>11/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hole</td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td>8/16</td>
</tr>
<tr>
<td>Small Object</td>
<td>9/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large Object</td>
<td>9/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angular Approach Left</td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>10/16</td>
</tr>
<tr>
<td>Angular Approach Right</td>
<td>10/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall Navigation</td>
<td>10/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall Intelligence</td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td>9/16</td>
</tr>
<tr>
<td>Overall</td>
<td>76/128</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Comments:
Corridor- non-fluid movement throughout corridor path
angular approach- would have been 4 except for long hesitations, but smooth movements and reactions

Time: 2 minutes, 53 seconds  Pass ✔  Fail
### Situational Rubric for PID, ANN, and FL controllers

**Navigating a Real Unknown Indoor Environment**

**Evaluator:** Overall  
**Robot Marker Color:** Red - ANN  
**Environment Layout #:** 5

<table>
<thead>
<tr>
<th>Situation</th>
<th>Score</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td>16/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td>13/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Small Object</strong></td>
<td>11/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free with no hesitation</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Large Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Left</strong></td>
<td>15/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Right</strong></td>
<td>14/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score</td>
<td></td>
</tr>
<tr>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall Navigation</strong></td>
<td>13/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall Intelligence</strong></td>
<td>12/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td>94/112</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Comments:**
Small object- fluid movements however seems like delayed reaction to sensing objects  
Good in room

**Time:** 3 minutes, 08 seconds  
**Pass ✔**  
**Fail ✗**
## Situational Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

**Evaluator:** Overall

**Robot Marker Color:** Red - ANN

**Environment Layout #:** 6

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corridor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angular Approach Left</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angular Approach Right</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall Navigation</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score</td>
</tr>
<tr>
<td>Overall Intelligence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Comments:
Only one collision in entire run

**Time:** 2 minutes, 50 seconds

**Pass ✓ Fail**
Situational Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

Evaluator: Overall
Robot Marker Color: Red - ANN
Environment Layout #: 7

<table>
<thead>
<tr>
<th>Situation</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>14/16</td>
</tr>
<tr>
<td>Corridor</td>
<td>15/16</td>
</tr>
<tr>
<td>Hole</td>
<td></td>
</tr>
<tr>
<td>Small Object</td>
<td>14/16</td>
</tr>
<tr>
<td>Large Object</td>
<td>15/16</td>
</tr>
<tr>
<td>Angular Approach Left</td>
<td>16/16</td>
</tr>
<tr>
<td>Angular Approach Right</td>
<td>16/16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Overall Navigation</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>14/16</td>
</tr>
<tr>
<td>Increased effort in fluid movement with minimal collisions</td>
<td></td>
</tr>
<tr>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td></td>
</tr>
<tr>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Overall Intelligence</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>14/16</td>
</tr>
<tr>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td></td>
</tr>
<tr>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td></td>
</tr>
<tr>
<td>Shows no ability to react to situations</td>
<td></td>
</tr>
</tbody>
</table>

Total: 118/128

Comments:
Minor collisions with objects, but the slow steady pace and fluid movements allowed for anticipation of next situation
Extremely smooth movements throughout
Great run!

Time: 1 minute, 15 seconds

Pass ✔ Fail
### Situational Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

**Evaluator:** Overall  
**Robot Marker Color:** Red - ANN  
**Environment Layout #:** 8

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>15/16</td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16/16</td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Small Object</strong></td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td>15/16</td>
</tr>
<tr>
<td><strong>Large Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16/16</td>
</tr>
<tr>
<td><strong>Angular Approach Left</strong></td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td>16/16</td>
</tr>
<tr>
<td><strong>Angular Approach Right</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score</td>
</tr>
<tr>
<td><strong>Overall Navigation</strong></td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>15/16</td>
</tr>
<tr>
<td><strong>Overall Intelligence</strong></td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td>14/16</td>
</tr>
</tbody>
</table>

**Comments:**
No collisions, navigates well in tight areas  
Extremely fluid throughout environment!

**Time:** 3 minutes, 08 seconds  
**Pass** ☑  
**Fail**

**Total:** 107/112
Situational Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

Evaluator: Overall  Robot Marker Color: Red- ANN
Environment Layout #: 9

<table>
<thead>
<tr>
<th>Situation</th>
<th>Score</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12/16</td>
</tr>
<tr>
<td>Corridor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12/16</td>
</tr>
<tr>
<td>Hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15/16</td>
</tr>
<tr>
<td>Small Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13/16</td>
</tr>
<tr>
<td>Large Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11/16</td>
</tr>
<tr>
<td>Angular</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12/16</td>
</tr>
<tr>
<td>Approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12/16</td>
</tr>
<tr>
<td>Left</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Right</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Navigation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12/16</td>
</tr>
<tr>
<td>Intelligence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11/16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Overall:** 110/144

Comments:
Appeared to have difficulty avoiding collision in first half of run, but second half= fluid navigation with minimal collisions

Time: 3 minutes, 07 seconds  Pass ✓  Fail
Situational Rubric for PID, ANN, and FL controllers navigating a real unknown indoor environment

Evaluator: Overall
Robot Marker Color: Red- ANN
Environment Layout #: 10

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>14/16</td>
</tr>
<tr>
<td>Corridor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small Object</td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td>11/16</td>
</tr>
<tr>
<td>Large Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angular Approach Left</td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Collides with object while attempting to navigate around</td>
<td></td>
<td></td>
<td>13/16</td>
</tr>
<tr>
<td>Angular Approach Right</td>
<td>Collides with object with no attempt to navigate around</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score</td>
</tr>
<tr>
<td>Overall Navigation</td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>13/16</td>
</tr>
<tr>
<td>Overall Intelligence</td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td>12/16</td>
</tr>
</tbody>
</table>

Total: 106/128

Comments:
Amazing run!
Smoothly moved in and around objects as if following a maze path
Mostly collision free
A shame this run has a ‘fail’ connotation because it did not find the goal, when the navigation was nearly perfect

Time: greater than 5 minutes = run termination
Pass               Fail ✔
### Situational Rubric for PID, ANN, and FL controllers
#### navigating a real unknown indoor environment

Evaluator: Overall  
Robot Marker Color: Green- Fuzzy Logic  
Environment Layout #: 1

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>16/16</td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td>9/16</td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>10/16</td>
</tr>
</tbody>
</table>

| **Small Object**| Shows ability to adapt and learn, shows improved performance throughout run | Shows ability to anticipate and react to situations well before approach | Reacts to situation, but has inconsistent reaction upon approach | Shows no ability to react to situations | 11/16 |
| **Large Object**| Shows ability to adapt and learn, shows improved performance throughout run | Shows ability to anticipate and react to situations well before approach | Reacts to situation, but has inconsistent reaction upon approach | Shows no ability to react to situations | 11/16 |
| **Angular Approach Left** | Shows ability to adapt and learn, shows improved performance throughout run | Shows ability to anticipate and react to situations well before approach | Reacts to situation, but has inconsistent reaction upon approach | Shows no ability to react to situations | 11/16 |
| **Angular Approach Right** | Shows ability to adapt and learn, shows improved performance throughout run | Shows ability to anticipate and react to situations well before approach | Reacts to situation, but has inconsistent reaction upon approach | Shows no ability to react to situations | 11/16 |

**Overall Navigation**:  
**Score**: 10/16

**Overall Intelligence**:  
**Score**: 11/16

**Total**: 95/128

**Comments**:  
Batteries running low, may have effected robot’s approach ability

**Time**: 3 minutes, 15 seconds  
**Pass**: ✔  
**Fail**
### Situational Rubric for PID, ANN, and FL controllers

Navigating a real unknown indoor environment

**Evaluator: Overall**

**Robot Marker Color:** Green - Fuzzy Logic

**Environment Layout #:** 2

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>9/16</td>
</tr>
<tr>
<td>Corridor</td>
<td>Small Object</td>
<td>Large Object</td>
<td>Angular Approach Left</td>
<td>Angular Approach Right</td>
<td>Overall</td>
</tr>
<tr>
<td>Hole</td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td>10/16</td>
</tr>
<tr>
<td></td>
<td>10/16</td>
<td>9/16</td>
<td>11/16</td>
<td>11/16</td>
<td>10/16</td>
</tr>
</tbody>
</table>

### Overall Navigation

- **Score:** 10/16

<table>
<thead>
<tr>
<th>Overall Intelligence</th>
<th>Traverse through testing environment fluidly, with no difficulty, and collision free</th>
<th>Increased effort in fluid movement with minimal collisions</th>
<th>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</th>
<th>Shows difficulty moving through the environment, and unable to avoid collision</th>
<th>10/16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time: 1 minute, 58 seconds</td>
<td>Pass ✓</td>
<td>Fail</td>
<td></td>
<td></td>
<td>79/128</td>
</tr>
</tbody>
</table>
### Situational Rubric for PID, ANN, and FL controllers

Navigating a real unknown indoor environment

Evaluator: Overall

Robot Marker Color: Green - Fuzzy Logic

Environment Layout #: 3

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>15/16</td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td>15/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td>15/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Small Object</strong></td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td>12/16</td>
</tr>
<tr>
<td><strong>Large Object</strong></td>
<td>12/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Left</strong></td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>13/16</td>
</tr>
<tr>
<td><strong>Angular Approach Right</strong></td>
<td>15/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score</td>
</tr>
<tr>
<td><strong>Overall Navigation</strong></td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td>13/16</td>
</tr>
<tr>
<td><strong>Overall Intelligence</strong></td>
<td>13/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td>110/128</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Comments:**
Room situation- small collision, but did not affect navigation ability, time for completion, or overall fluid movement

Time: 4 minutes, 02 seconds

Pass ☑️ Fail
### Situational Rubric for PID, ANN, and FL controllers

*navigating a real unknown indoor environment*

Evaluator: Overall

Robot Marker Color: Green - Fuzzy Logic

Environment Layout #: 4

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>11/16</td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Small Object</strong></td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td>13/16</td>
</tr>
<tr>
<td><strong>Large Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Left</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Right</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score</td>
</tr>
<tr>
<td><strong>Overall Navigation</strong></td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>12/16</td>
</tr>
<tr>
<td><strong>Overall Intelligence</strong></td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td>12/16</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td>97/ 128</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Comments:**

Corridor - successful in all navigations of corridors, but non-fluid movement throughout length - chose zig-zag pattern instead of most direct path straight through

Time: 3 minutes, 13 seconds  
Pass ✔  
Fail
# Situational Rubric for PID, ANN, and FL controllers navigating a real unknown indoor environment

Evaluator: Overall  
Robot Marker Color: Green - Fuzzy Logic  
Environment Layout #: 5

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13/16</td>
</tr>
<tr>
<td>Corridor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14/16</td>
</tr>
<tr>
<td>Hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small Object</td>
<td></td>
<td></td>
<td>12/16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large Object</td>
<td></td>
<td></td>
<td>11/16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angular Approach</td>
<td></td>
<td></td>
<td>11/16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Right</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score</td>
</tr>
<tr>
<td>Overall Navigation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12/16</td>
</tr>
<tr>
<td>Overall Intelligence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12/16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comments:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room- fluid movements, with delayed time in completing</td>
</tr>
<tr>
<td>Controller exhibits consistent performance overall, even though not a perfect controller</td>
</tr>
</tbody>
</table>

Time: 2 minutes, 51 seconds  
Pass ☑  
Fail ✗
# Situational Rubric for PID, ANN, and FL controllers navigating a real unknown indoor environment

**Evaluator:** Overall  
**Robot Marker Color:** Green- FL  
**Environment Layout #:** 6

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>11/16</td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td>12/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td>11/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Small Object</strong></td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
<td>9/16</td>
</tr>
<tr>
<td><strong>Large Object</strong></td>
<td>9/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Left</strong></td>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>12/16</td>
</tr>
<tr>
<td><strong>Angular Approach Right</strong></td>
<td>13/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Score</td>
</tr>
<tr>
<td><strong>Overall Navigation</strong></td>
<td>100/144</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall Intelligence</strong></td>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td>Shows no ability to react to situations</td>
<td>10/16</td>
</tr>
</tbody>
</table>

**Comments:**
Overall navigation consistent to this controller; however, it reacts to objects with obvious delays- these delays affect its ability to navigate collision free

**Time:** 1 minute, 32 seconds  
**Pass ✔ Fail ✗**
Situational Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

Evaluator: Overall
Robot Marker Color: Green- FL
Environment Layout #: 7

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avoids situation or unable to successfully navigate through</td>
<td>12/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12/16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td></td>
<td></td>
<td>13/16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collides with object while attempting to navigate around</td>
<td></td>
<td>12/16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collides with object with no attempt to navigate around</td>
<td></td>
<td>13/16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small Object</td>
<td>(12/16)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large Object</td>
<td>(13/16)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Left</strong></td>
<td>(13/16)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Right</strong></td>
<td>(13/16)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall 4</td>
<td>(11/16)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traverse through testing environment fluidly, with no difficulty, and collision free</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12/16</td>
</tr>
<tr>
<td>Increased effort in fluid movement with minimal collisions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall Navigation</td>
<td>(111/144)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shows no ability to react to situations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11/16</td>
</tr>
<tr>
<td>Total:</td>
<td>(111/144)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Comments:

Time: 1 minute, 32 seconds

Pass ✅
Fail
Situational Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

Evaluator: Overall  Robot Marker Color: Green- FL
Environment Layout #: 8

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>15/16</td>
</tr>
<tr>
<td>Corridor</td>
<td>Small Object</td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td>Collides with object with no attempt to navigate around</td>
</tr>
<tr>
<td>Hole</td>
<td>Large Object</td>
<td>Angular Approach Left</td>
<td>Angular Approach Right</td>
<td>Overall</td>
<td>Overall Navigation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Score</td>
</tr>
</tbody>
</table>

Comments:
Inconsistent with angular approaches- some collisions, while others perfectly smooth collision free

Time: 2 minutes, 37 seconds  Pass ✔ Fail
Situational Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

Evaluator: Overall
Robot Marker Color: Green- FL
Environment Layout #: 9

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>12/16</td>
</tr>
<tr>
<td>Corridor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12/16</td>
</tr>
<tr>
<td>Hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13/16</td>
</tr>
<tr>
<td>Small Object</td>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td></td>
<td></td>
<td>10/16</td>
</tr>
<tr>
<td>Large Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8/16</td>
</tr>
<tr>
<td>Angular Approach Left</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9/16</td>
</tr>
<tr>
<td>Angular Approach Right</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9/16</td>
</tr>
<tr>
<td>Overall</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>90/144</td>
</tr>
</tbody>
</table>

Overall Navigation
- Traverse through testing environment fluidly, with no difficulty, and collision free
- Increased effort in fluid movement with minimal collisions
- Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects
- Shows difficulty moving through the environment, and unable to avoid collision

Overall Intelligence
- Shows ability to adapt and learn, shows improved performance throughout run
- Shows ability to anticipate and react to situations well before approach
- Reacts to situation, but has inconsistent reaction upon approach
- Shows no ability to react to situations

Total: 90/144

Comments:
Not the best performance from the "green marker" controller

Time: 3 minutes, 02 seconds

Pass ✓ Fail
### Situational Rubric for PID, ANN, and FL controllers
navigating a real unknown indoor environment

Evaluator: Overall  
Robot Marker Color: Green- FL  
Environment Layout #: 10

<table>
<thead>
<tr>
<th>Situation</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Room</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enters situation fluidly, successfully navigates through without collisions or hesitations in a timely manner</td>
<td>Enters situation fluidly but may show hesitation and/or minimal collisions with delayed completion time</td>
<td>Enters situation with non-fluid movement, encounters multiple collision, completes situation in excessive time</td>
<td></td>
<td>Avoids situation or unable to successfully navigate through</td>
<td>12/16</td>
</tr>
<tr>
<td><strong>Corridor</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hole</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anticipates object and shows course redirection while fluidly moving around object collision free with no hesitation</td>
<td>Anticipates object and shows attempt at course redirection with non-fluid movement while remaining collision free</td>
<td>Collides with object while attempting to navigate around</td>
<td></td>
<td>Collides with object with no attempt to navigate around</td>
<td>11/16</td>
</tr>
<tr>
<td><strong>Small Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8/16</td>
</tr>
<tr>
<td><strong>Large Object</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Angular Approach Left</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traverse through testing environment fluidly, without difficulty, and collision free</td>
<td>Increased effort in fluid movement with minimal collisions</td>
<td>Moves through environment with inconsistent movements (fluid and non-fluid), inconsistently collides with objects</td>
<td></td>
<td>Shows difficulty moving through the environment, and unable to avoid collision</td>
<td>11/16</td>
</tr>
<tr>
<td><strong>Angular Approach Right</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall Navigation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11/16</td>
</tr>
<tr>
<td>Shows ability to adapt and learn, shows improved performance throughout run</td>
<td>Shows ability to anticipate and react to situations well before approach</td>
<td>Reacts to situation, but has inconsistent reaction upon approach</td>
<td></td>
<td>Shows no ability to react to situations</td>
<td>12/16</td>
</tr>
<tr>
<td><strong>Overall Intelligence</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>76/112</td>
</tr>
</tbody>
</table>

Comments: 

Time: 0 minutes, 29 seconds  
Pass ✓  Fail