Santa Clara University
Scholar Commons

Mechanical Engineering Master's Theses | Engineering Master's Theses

9-28-2010

Dynamic Control Migration Between a Base Station and a Remote Robot

Adam Davis Westgate
Santa Clara University

Follow this and additional works at: https://scholarcommons.scu.edu/mech_mstr

Recommended Citation
Westgate, Adam Davis, "Dynamic Control Migration Between a Base Station and a Remote Robot" (2010). *Mechanical Engineering Master's Theses*. 30.
https://scholarcommons.scu.edu/mech_mstr/30

This Thesis is brought to you for free and open access by the Engineering Master's Theses at Scholar Commons. It has been accepted for inclusion in Mechanical Engineering Master's Theses by an authorized administrator of Scholar Commons. For more information, please contact rscroggin@scu.edu.

Santa Clara University
DEPARTMENT of MECHANICAL ENGINEERING

Date: September 28, 2010

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Adam Davis Westgate

ENTITLED

Dynamic Control Migration Between a Base Station and a Remote Robot

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTERS OF SCIENCE IN MECHANICAL ENGINEERING

_____
CHRISTOPHER KITTS
THESIS ADVISOR

_____
TIMOTHY HIGHT
DEPARTMENT CHAIR

Dynamic Control Migration Between a Base Station and a Remote Robot

by

Adam Davis Westgate

MASTERS OF SCIENCE THESIS

Submitted in partial fulfillment of the requirements
for the degree of
Masters of Science in Mechanical Engineering
School of Engineering
Santa Clara University

Santa Clara, California

September 28, 2010

Dynamic Control Migration Between a Base Station and a Remote Robot

Adam Davis Westgate

Department of Mechanical Engineering

Santa Clara University 2010

Abstract

The paper introduces a new approach to adapting network control systems to changing network conditions. The meta-controller proposed here is capable of monitoring network communication delays and seamlessly switching from control loops executing on a base station computer to control loops on a remote robot. This allows the control system to handle unexpected communication delays or failures without halting operation or becoming unstable. It also allows for a high level of human in the loop operation or monitoring at the base station without sacrificing the autonomous behavior of the remote robot. The meta-controller can automatically transition control loops between the base station and remote robot as operator confidence in system performance increases. This research develops the meta-controller framework, proposes a switching strategy, and demonstrates the concept through simulation and experimental testing.

Keywords: Meta-control

Acknowledgements

I would like to thank my advisor, Dr. Christopher Kitts, for this guidance and support throughout this project and my graduate student career. His passion and dedication for his work is always inspiring.

Secondly, I would like to thank the members of the SCU Robotics System Lab for all of their assistance. Without their help my simulation and tests would not have gone as smoothly as they did.

Finally, I would like to thank my wife and family for the love and support that they have given me throughout my academic career. Without their help and encouragement, I would not have achieved my goals.

Table of Contents

List of Figures

List of Tables

1 Introduction

Networked control systems are becoming more common in a wide variety of fields. Applications range from the Mars rovers [1] to haptic manipulator control for teleoperated surgery [2] to multi-robot coordination for exploration and mapping [3]. All of these systems have some part of their control system operating over a network link and must be designed within the constraints imposed by those networks. A significant amount of research has been done focusing on how to design for these network delays and handle unexpected delays and communication failures [2, 4, 5]. This paper will present a new control system design which can improve performance on networks with large latency variance, handle network latency outside of the design parameters, and maintain control during communication failures. It also allows for a high level of human in the loop control without sacrificing system performance. The meta-controller can be used to automatically transition to autonomous control as the operator's confidence in the system grows.

1.1 Networked Control Systems

The design issues involved with operating a control loop over a wireless communication link have been explored in depth, in several different papers [9-13]. These papers discuss how to deal with network latency, bandwidth limitations, and interference which can help to make decisions about where control loops should be executed, at what rate, and what performance can be achieved. The described practices work well under average network conditions where a maximum latency and interference can be measured or reasonably estimated and included in the control system design.

Existing research into handling network anomalies include strategies such as dynamically reducing the performance of the control system [14] or safely stopping the remote operation until network performance improves or communication is restored [2]. This paper proposes an alternative design which will maintain operation even during a communication failure.

1.2 Autonomous vs. Human in the Loop Control

Another common design issue for remotely operated systems is the tradeoff between autonomous operation and human in the loop control. Allowing more autonomous operation frees the system from waiting for operator input and can significantly improve the overall performance of the system. Maintaining human control allows for closer management of system operation and may be preferable for difficult tasks where current levels of automation are unreliable or experimental. The controller proposed in this paper works for both fully autonomous systems and human in the loop systems where autonomous operation is possible but not necessarily desired. More specific details of applications are discussed below.

1.3 What is Meta-Control?

There are several different definitions and a wide variety of applications for meta-control which have been presented in multiple papers. These definitions span a wide range of topics from a web framework for dynamically enforcing context sensitive policies [6] to a framework for scheduling high level activities based on the tradeoff between quality and duration of the tasks [7]. All of these definitions follow the same underlying principle of altering the operation of the controller or control system based on the state of the system. This paper focuses the meta-controller definition on altering the

physical location of control loop execution, specifically, migrating control back and forth between a base station and a remote robot based on the amount of network communication delay.

1.4 Applications

1.4.1 Mars Rovers

NASA launched the Mars Pathfinder in December 1996. The Pathfinder landed successfully on the surface of Mars and became the first rover to drive on the Martian surface. The final transmission was received from Pathfinder in September of 1997. NASA has successfully operated two other rovers, the MER-A Spirit and the MER-B Opportunity for extended periods of time.

The several minute one-way communication delay between Earth and Mars makes it impossible to drive the rovers via a joystick on Earth. Earth based drivers would not be able to see and react to obstacles in time when the rover is traveling at any speed above a slow crawl. To solve this problem, the designers were forced to use a combination of Earth based and rover based control as well as autonomous and human in the loop control [1, 15]. The autonomous control allows for continuous navigation and operation despite the communication delay and the Earth based navigation planning allows operators to move the rovers to specific locations required to meet mission objectives.

Relinquishing human control of a multi-million dollar rover which cannot be serviced or retrieved is not something that is taken lightly. Operators prefer to start conservatively. They maintain human control and monitor the performance while they build confidence in the system. After a performance history has been built up, operators

disable human control and enable autonomous functions. This transition is performed manually, piece by piece, with separate pieces of behavior being enabled at different times. The meta-controller presents a way to handle this transition autonomously. The meta-controller allows for base station control, but will automatically switch to remotely executing control loops when system performance suffers. This allows for close operator monitoring and control without sacrificing system performance due to long communication delays.

1.4.2 Model Based Anomaly Management of Space Systems

This technique maintains a system model and attempts to diagnose and correct for anomalies on in-flight space system [8]. Anomalies are detected and mitigated by comparing the actual system state to the predicted output of the system model. When the anomaly detection system is first brought online, human operators closely monitor every aspect of it. Operators will enable autonomous base station and eventually autonomous remotely executed parts of the control system slowly as their confidence in the system builds. The transition process is performed manually for each section of the control loop which needs to be transferred. The meta-controller can be used to automate this transition process and allow for easy transition of small or large sections of the control loops. Communication link dependent sections of the control can be run and monitored at the base station and automatically switch to the remote computer during communication link lag or failure.

1.5 Objectives

The objective of this paper is to explore the automated migration of control for a remote robot as a function of communication link delay between the robot and its remote

operator control console. As an initial study of this capability, the basic navigation tasks of controlling heading and position have been selected for study. Performing this study required the formulation of a control switching architecture and the establishment of switching criteria based on task performance. To iteratively test and explore this approach, a simulation environment was developed and a number of simulated cases were evaluated. The technique was also evaluated through hardware experimentation with a simple mobile robot. Results show that the automated switching of the motion controller successfully allows performance to be maintained in the face of time-varying communication delay.

1.6 Reader's Guide

Chapter 1 describes the motivations and objectives of this thesis and briefly describes the achieved results. Chapter 2 reviews the classic design of networked control systems as well as the meta-controller theoretical design. Chapter 3 presents the simulation model that is used to test the meta-controller and the results shown by those tests. Chapter 4 describes the experimentation test setup used to test the meta-controller model in an example application as well as experimental results. Chapter 5 summarizes the accomplished work and discusses future research that can be done in this area.

2 The Meta-Controller

This chapter reviews classical network control system design and the implementation of the meta-controller control system. Section 1 overviews the classic approach for designing a networked control system and how to address issues such as latency, network bandwidth, and packet loss. Section 2 describes the meta-controller theory and design.

2.1 Network Control Systems

Network control system design must take into consideration three aspects of the network communication system when designing the controllers. These three aspects are network bandwidth, latency, and packet loss. The affects of each of these are reviewed below.

2.1.1 Network Bandwidth

Network bandwidth will set a maximum for the sampling and execution rate of the controller across the networks. Under optimal conditions the maximum sampling rate ($\omega_{s_{max}}$) can be computed using the following formula.

$$\omega_{s_{max}} = \frac{Bandwidth}{Data\ Length + Message\ Overhead} \quad (1)$$

Where bandwidth is the maximum bits/second which can be transmitted over the link, data length is the length in bits of the control data, and message overhead is the extra data overhead required to transmit a message. In simpler communication protocols, message overhead may include a message start flag, data length, checksum, and end flag.

In more complex protocols, overhead may also include extra data for more sophisticated error detection and correction algorithms, encryption, and so on.

In simple point to point networks it is acceptable to use a much larger percentage of the total network bandwidth, whereas wireless networks with more nodes should reserve a larger portion of the total network bandwidth to help reduce collisions and transmission waiting delays.

When following a classical design approach in the frequency domain, if the network bandwidth is the limiting factor on sample rate it also determines the Nyquist frequency of the controller. Standard practices to account for the phase and gain consumed by the digital effects should be used. For example, if a zero order hold is used to sample data at the maximum network sample frequency, the phase and gain consumed by the ZOH can be calculated using the following formulas.

$$|G(j\omega)| = T\frac{\left|sin\left(\pi\frac{\omega}{\omega_s}\right)\right|}{\pi\frac{\omega}{\omega_s}} \tag{2}$$

$$\measuredangle G(j\omega) = -\pi\frac{\omega}{\omega_s} \tag{3}$$

Where $G(j\omega)$ is the desired transfer function, ω is the gain crossover frequency of the desired transfer function, and ω_s is the sample frequency. These gain and phase numbers should be allocated during the design phase of the controller.

2.1.2 Network Latency

Network latency is simply the total amount of time it takes the controllers to pack, transmit, error check, and unpack the control data. Packing, error checking, and unpacking should be very close to constant time as they only rely on either the sending or

receiving processor. In a point to point network, transmission time can also be close to a constant value as there is no waiting for another controller to finish transmitting before this controller's transmission begins.

Analysis of network delay for multi-node networks can be significantly more complex. [16] shows that designing for the average network latency can produce acceptable results and that significantly better results can be attained by using a state estimator and feeding measured network latency as input. Both of these methods will eventually become unstable under long enough network transmission delays.

Figure 1 illustrates the stack up of network latency times.

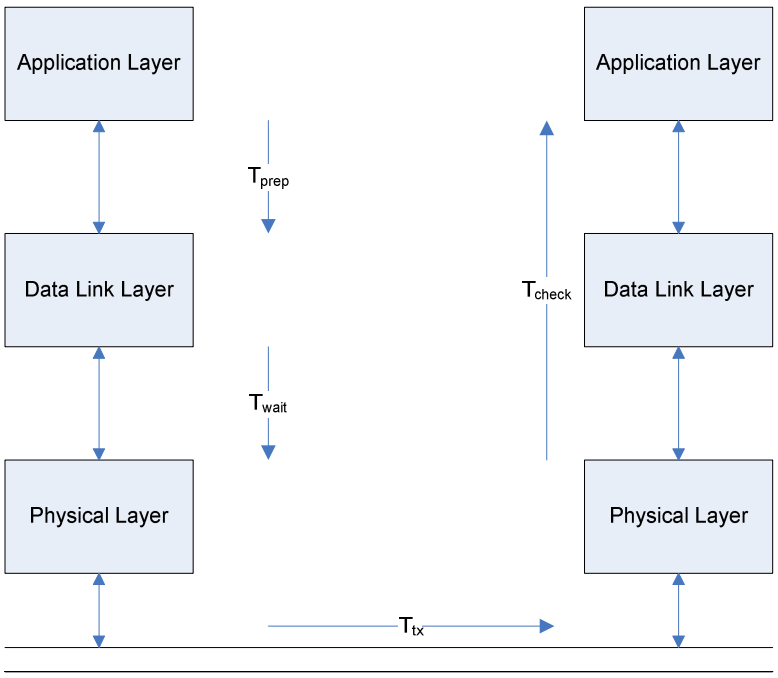


Figure 1 Network Time Delays

In a classical control system design network latency should be added to all other computational time delays. This total delay time does not alter the shape of the input so it

only contributes phase shift, or lag, to the system. The formula below, from [2], can be used to calculate the phase lag.

$$\varphi = t_d \omega \frac{180°}{\pi} \tag{4}$$

Where t_d is the total delay time including network latency and computational delays and ω is the crossover frequency of the desired transfer function.

2.1.3 Packet Loss

Both wired and wireless networks can suffer from information loss due to interference and network collisions. This packet loss can have one of two effects depending on the type of network used.

If the network protocol allows for error detection and packet retransmission, the lost packet will still reach its destination but will incur a longer time delay. The retransmission time multiplied by the expected packet loss percentage should be included in the total computation delay time calculated above. If the network protocol does not allow for packet loss detection and retransmission then the lost packet is effectively slowing the data sampling rate of the controller. Liu and Goldsmith analyze the effect of packet loss and derive the following formula for the effective sampling rate in [2].

$$\omega_{eff} = \omega_s r \tag{5}$$

Where ω_s is the previously calculated sample rate, r is the fractional percentage of messages expected to be successfully transmitted, and ω_{eff} is the new effective sample rate. This new sample rate should be used when compensating for gain and phase caused by digital sampling.

2.1.4 Placing Control Loops

To determine which control loops should be executed on the remote robot and which should be operated across the network, the designer should complete two separate performance analyses. The first should determine the system performance across the network using the techniques discussed above and the second should evaluate system performance when running on the remote robot using standard digital control design techniques.

In systems only concerned with performance, the comparison between the two analyses determines where to place the control loops. In complex systems, like those described in the applications section, it is common that the limited processor capabilities of the remote robot will perform worse than the networked control systems. The meta-controller discussed in this paper will improve the overall performance of these complex systems as well as systems where control loops are kept at the base station for non-performance reasons.

2.2 The Meta-controller

In the previous section we discussed the design of a networked control under normal operating conditions. The classical controller maintains system stability by maintaining phase margin at the controller crossover frequency and gain margin to account for inaccuracies in the model and various mechanical components. When the system is operating within the design limits, the phase and gain margins are sufficient to maintain stability and performance. However, when the operating conditions begin to change the margins shrink, the performance degrades, and the system eventually becomes unstable. The meta-controller addresses this problem for changes in network

performance of control systems by monitoring the communication link and dynamically switching between controllers.

2.2.1 Theory

The meta-controller model monitors network latency and switches control loops between local and base station execution to maintain stability in the system. To achieve this, both controllers must be capable of running stable control loops for the system and the remote robot must have some mechanism for measuring network latency. The general form of the meta-controller is shown in figure 2.

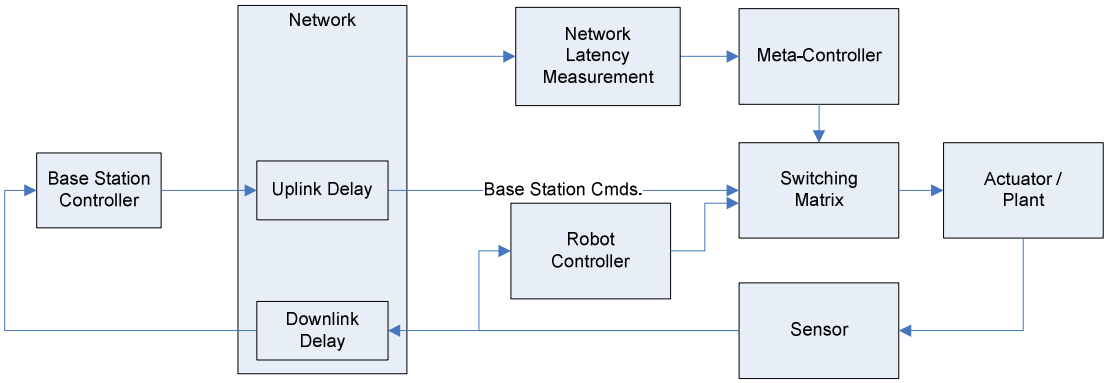


Figure 2 Meta-controller System Overview

A switching matrix is used by the meta-controller to select which control loop is active. The commands from each control source are multiplied by either a one or a zero to select which set of command is passed to the actuator. This is the simplest method to transition between two separate controllers, but can be expanded to allow more sophisticated switching techniques. For example, instead of swapping the ones and zeroes when a control transition occurs, a ramp function can be used over several sample periods to smoothly transition the zero to a one and the one to a zero. This will

effectively blend the two commands over the ramp transition period providing a smoother transition for the system.

The networked base station control loop is designed with a finite amount of gain and phase margin. The exact margin can be calculated by sine sweeping the system under normal operating conditions. Also, the nominal network latency should be measured and the phase lag caused by network latency should be added back to the measured phase margin. The meta-controller will allow most of this phase margin to be consumed before switching to remote robot control. The designer may place the phase margin switchover point wherever they desire, but simulation trial and error suggest maintaining at least 10 degrees of margin.

Once the designer determines how much phase lag can be caused by network latency the value can be back calculated into latency in seconds using a modified version of formula 4.

$$\frac{\varphi}{\omega}\frac{\pi}{180°} = t_d \tag{6}$$

The time delay is measured by the meta-controller and used to set the switching matrix to the desired control loop. A walkthrough of an example meta-controller design is shown in detail in the simulation portion of this document.

2.2.2 Transitioning

Transitioning between control loops, even in a P or PD system, can cause a discontinuity in the command output. If the system cannot handle a discontinuous step in command a ramp function can be used during the transition period to smooth transition between controllers.

2.2.2.1 P Transition

The maximum command step cause by a proportional gain during transition is a function of the time delay transition set point, the maximum expected system rate of change, and the P gains. The following formula is derived for calculating the maximum command step caused by switching proportional controllers.

$$\Delta_{cmd_{max}} = K_{P_{max}} v_{max} t_{d_{sp}} \quad (7)$$

Where $K_{P_{max}}$ is the larger of the two gains, v_{max} is the maximum expected system rate of change, and $t_{d_{sp}}$ is the lag time transition set point of the meta controller. The effects of this command step are demonstrated and discussed in further detail in the simulation section of this paper.

2.2.2.2 D Transition

The maximum step command caused by the derivative gain can be calculated in a similar manner to the proportional gain, substituting the maximum expected acceleration for maximum expected rate of change. The new formula follows.

$$\Delta_{cmd_{max}} = K_{D_{max}} a_{max} t_{d_{sp}} \quad (8)$$

Where $K_{D_{max}}$ is the large of the two gains, a_{max} is the maximum expected system acceleration, and $t_{d_{sp}}$ is the lag time transition set point of the meta controller.

2.2.2.3 I Transition

If an integrator is used in the controller, several methods can be used to prevent integrator wind up. The simplest solution is to zero the integrator of the controller that is

not active. This requires a signal be returned from the meta-controller on the robot to zero the integrator of the base station. Transitioning can cause a maximum step command equal to the maximum integrator command output.

$$\Delta_{cmd_{max}} = (K_I I)_{max} \quad (9)$$

Where K_I is the integrator gain, I is the maximum integrator value, and $(K_I I)_{max}$ is the larger value of the two controllers.

Alternatively, the integrator value for the active controller can be passed to the inactive controller and used to set the inactive controllers integrator value. If the control loops do not have identical integrator gains, the integrator value should be multiplied by the ratio of the gains before being used to set the integrator. While this method will give a smoother transition between controllers, it will also consume more network bandwidth to transmit both the active controller command and integrator values.

2.2.2.4 Total Transition Effects

The total worst case step in control command is simply the sum of the three P, D, and I command steps. If this step command is not acceptable in the system, more sophisticated transitioning methods should be explored.

3 Simulation

3.1 Simulation Setup

The meta-controller model is tested using a Simulink simulation which combines the base station, robot, and meta-controller as well as a simulated network delay. This is connected via Data Turbine to a Mobile Sim model of a single differential drive mobile ground robot. The simulation setup is shown below.

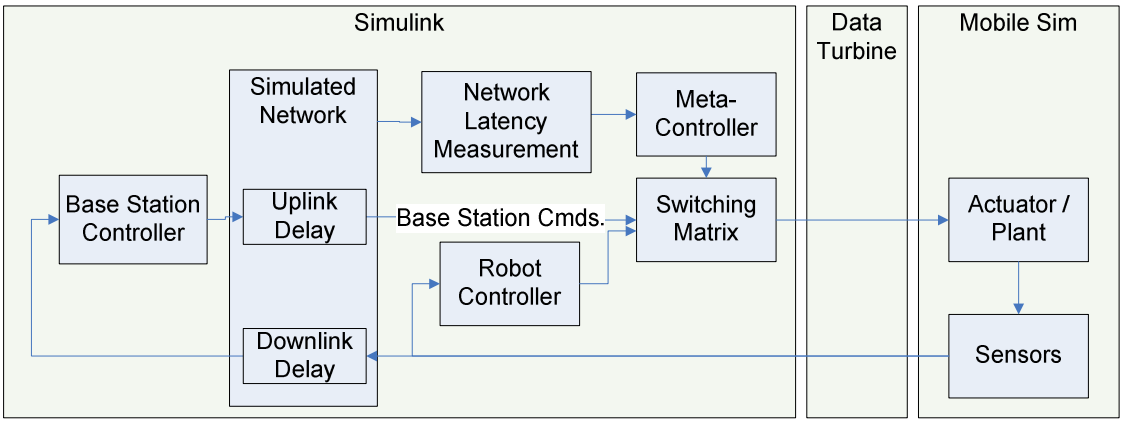


Figure 3 Simulation Configuration

Data Turbine is a communication application which makes data passing over a network between applications seamless. Different applications can be swapped out without reconfiguring other applications on the network for a different source or sink.

Mobile Sim runs real time models of different mobile robots. It receives heading and velocity commands from the controller via Data Turbine and sends position, heading, and velocity information back from the model. Data Turbine and Mobile Sim are used to allow an easy transition into controlling the demonstration robot.

The kinematics of the differential drive robot are described in the following figure and equations. The important feature of the differential drive configuration is the ability to turn in place. This decouples the heading control from the velocity control as long as the velocity control commands are not allowed to saturate the drive motor output. More information on differential drive robots can be found in [11] or a number of other books on the topic of mobile robot kinematics.

$$V_r = \omega\left(R + \frac{l}{2}\right) \tag{10}$$

$$V_l = \omega\left(R - \frac{l}{2}\right) \tag{11}$$

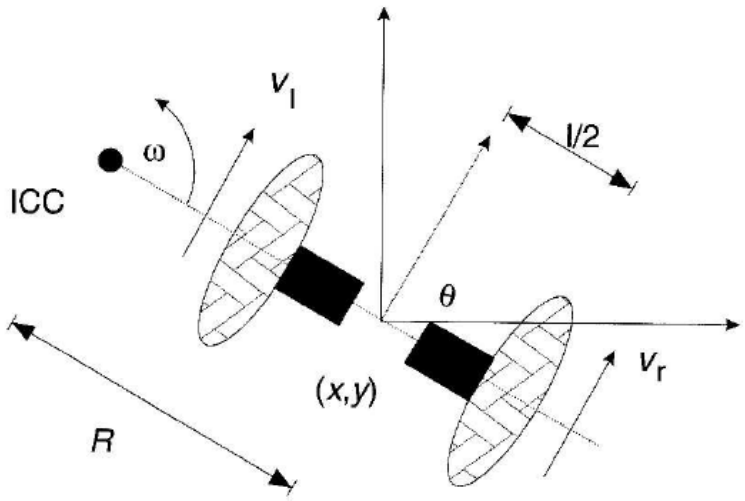


Figure 4 Differential Drive Robot Dynamics

The high level view of the Simulink model of the controllers is shown in figure 5. The main parts of the simulation are the base station controller, robot controller, meta-controller, network latency simulation, and the robot model simulation interface.

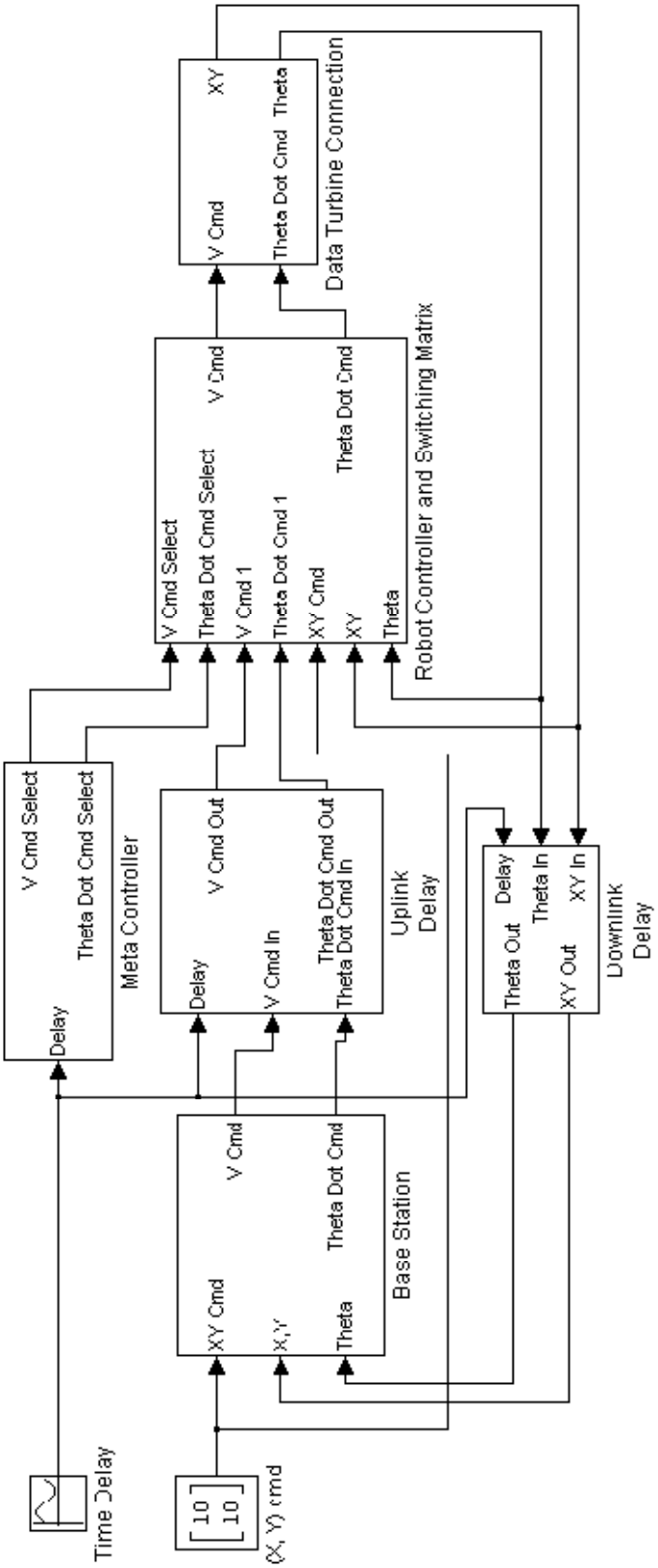


Figure 5 Meta-controller Simulation Model

Both the base station and robot controller models run identical copies of an inner heading control loop and outer position control loop. The outer position control loop calculates the heading and distance to goal and outputs a desired heading and a velocity command. The desired heading command is passed to the heading control loop which compares it to the actual heading and generates a differential speed command to turn the robot to the desired heading. The controller and equations are shown below.

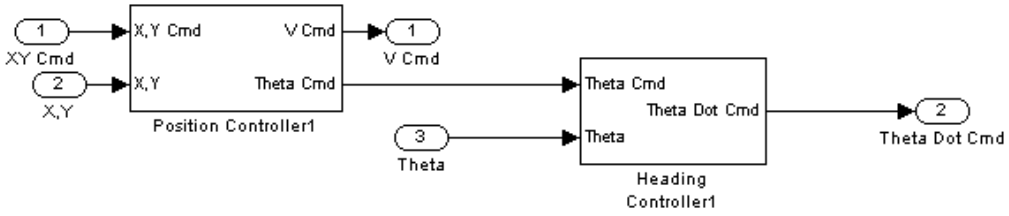


Figure 6 Base Station Controller Model

$$v_{cmd} = K_{pv}\sqrt{{x_e}^2 + {y_e}^2} \tag{12}$$

$$\theta_{cmd} = \tan^{-1}\frac{x_e}{y_e} \tag{13}$$

$$\dot{\theta}_{cmd} = K_{p\theta}\theta_e \tag{14}$$

The gains of position and heading controllers used in the simulation are set solely on trial and error until acceptable performance is achieved. The control gains are not optimal and are not product of a classic design for a desired transfer function. An optimized controller is not required to test the meta-controller as even a controller with low gains can be made unstable with sufficient communication delay.

The robot controller also contains the switching matrix which is controlled by the meta-controller and is used to switch between robot and base station control. A simple

discrete switching matrix is used and more advanced transitioning techniques are not explored in this paper.

The Data Turbine connection block encapsulates the Data Turbine interface to the Mobile Sim robot model. It contains both the uplink for forward and rotational velocity command and the downlink for position and heading feedback.

Finally, the network between the base station and controller is simulated by adding a variable uplink and downlink delay between the base station and robot. In the simulation this delay is set as a sinusoidal input to cause multiple switches between control loops on a single run. The amplitude of the delay is selected to cause instability when the robot is controlled in base station only mode.

3.2 Meta-controller Design

The meta-controller measures the network latency and uses simple thresholds to switch between controllers. When the latency exceeds a set threshold the meta-controller will switch to robot control. When the latency drops back below the threshold the meta-controller switches back to base station control. If there were sufficient noise on the measured latency, hysteresis could be added to the switching thresholds to prevent frequent transitioning when delay is near the cross over point.

The meta-controller uses two different switching thresholds for the inner and outer loop as the inner loop is operating at a higher bandwidth and is more sensitive to the network latency. For simulation purposes the meta-controller can perfectly measure the uplink and down link latency with zero time delay. In an actual implementation,

there may be both error and delay in this measurement. The designer can correct for expected error and delay by increasing the phase margin switch over point.

After the control gains are selected a sine sweep of the inner control loop is performed to measure the closed loop system response. For the specific physical system used for this research, the system response begins to roll off at command frequencies higher than 0.3 rad/sec. This is used as the gain crossover frequency when determining the meta-controller switching thresholds. The sine sweep also shows that the controller has about 25° of phase lag. This is used to determine the switching threshold for the meta-controller.

Equation 6 is used to calculate the time delay switch over point which will be used for the inner loop controller. Any system will become unstable if there is more than 180° of phase lag. This controller already has 25° of phase lag and we would like to keep 10° of margin. This leaves 145° of phase which can be consumed by the time delay. Using equation 6 with the calculated phase and gain crossover frequency gives the following.

$$\frac{145°}{0.3\ rad/s}\frac{\pi}{180°} = 8.4s \tag{15}$$

This number is used as the switching threshold for the inner control loop of the meta-controller model. This number seems excessively high because the Simulink model is running faster than real time and the Mobile Sim plant model is running in approximately real time. This makes the plant seem extremely slow from Simulink's perspective which is why this threshold is so high. The different time frames do affect the simulation results as all results are presented in the Simulink time frame.

A full analysis was not performed for the outer loop. The threshold was set approximately 50% higher than the inner loop threshold. Setting the thresholds close to each other allows for transitions of both control loops without extremely large differences in time delays.

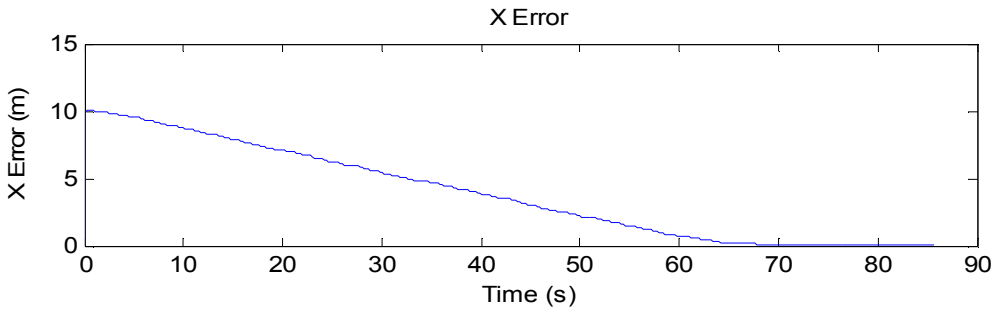
An alternative method for setting the switching thresholds is to gather simulation data with increasing time delays. The point at which the system becomes unstable or the performance is unacceptable can be used as the transition threshold for the system.

3.3 Simulation Results

For all tests the robot starts at position (0, 0) and is given a step command to (10, 10).

3.3.1 Robot Control

A baseline run is completed using only the robot controller. All meta-controller simulation results will be compared to this baseline. The robot trajectory as well as X, Y, and heading error are plotted below. As expected, the robot controller performs well and the robot drives smoothly to the goal position.



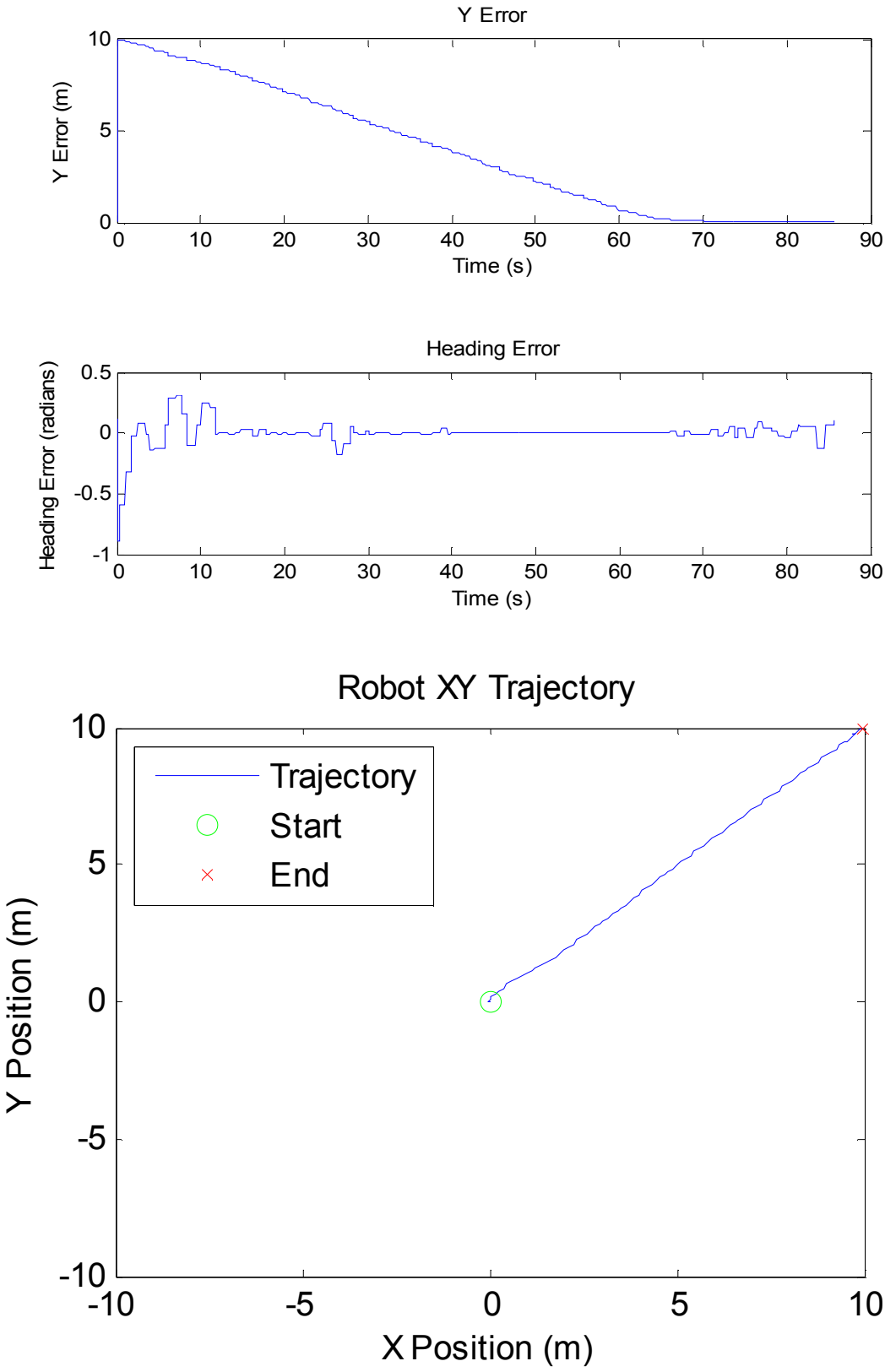
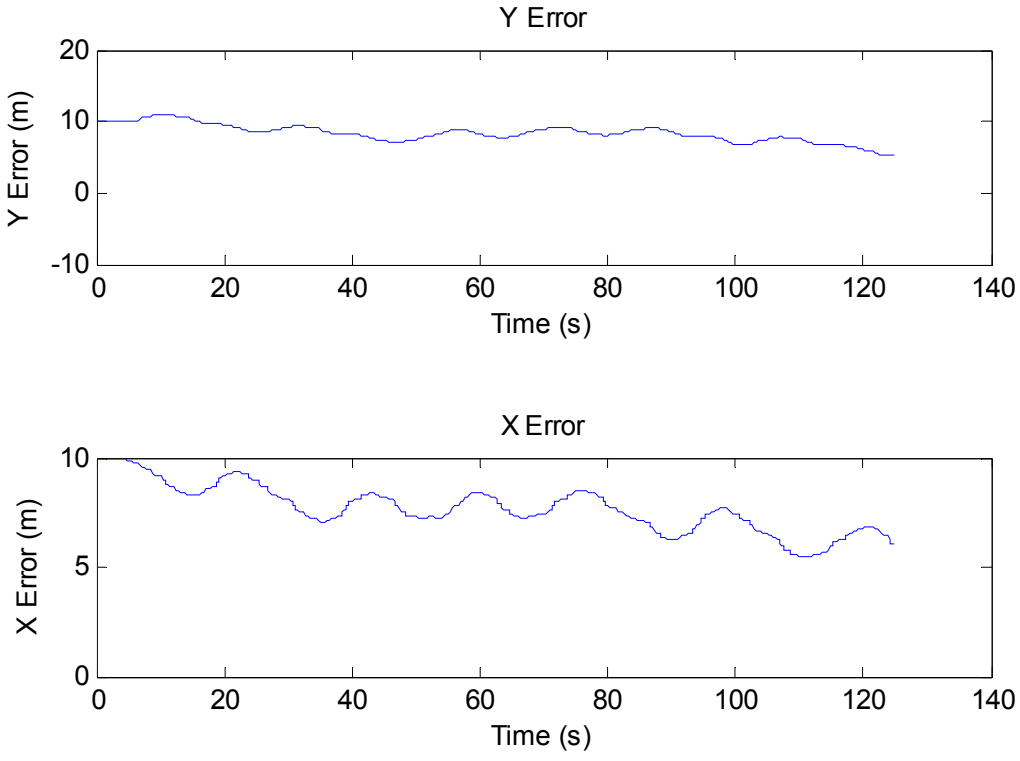


Figure 7 Robot Control Simulation Results

3.3.2 Base Station Control

This simulation forces control to the base station. The time delay input is a sine wave and is selected to cause the base station control to become unstable. The frequency is high enough that when the meta-controller is enabled there will be multiple switching events in a single run.

The following results show the instability of the base station controller under these conditions. The controller is unable to continuously maintain the stability of the inner heading control loop which causes the robot to drive away from the goal when the delays are high and turn back towards the goal when the time delays are low. The switching lines shown in the last graph are locked to base station control.



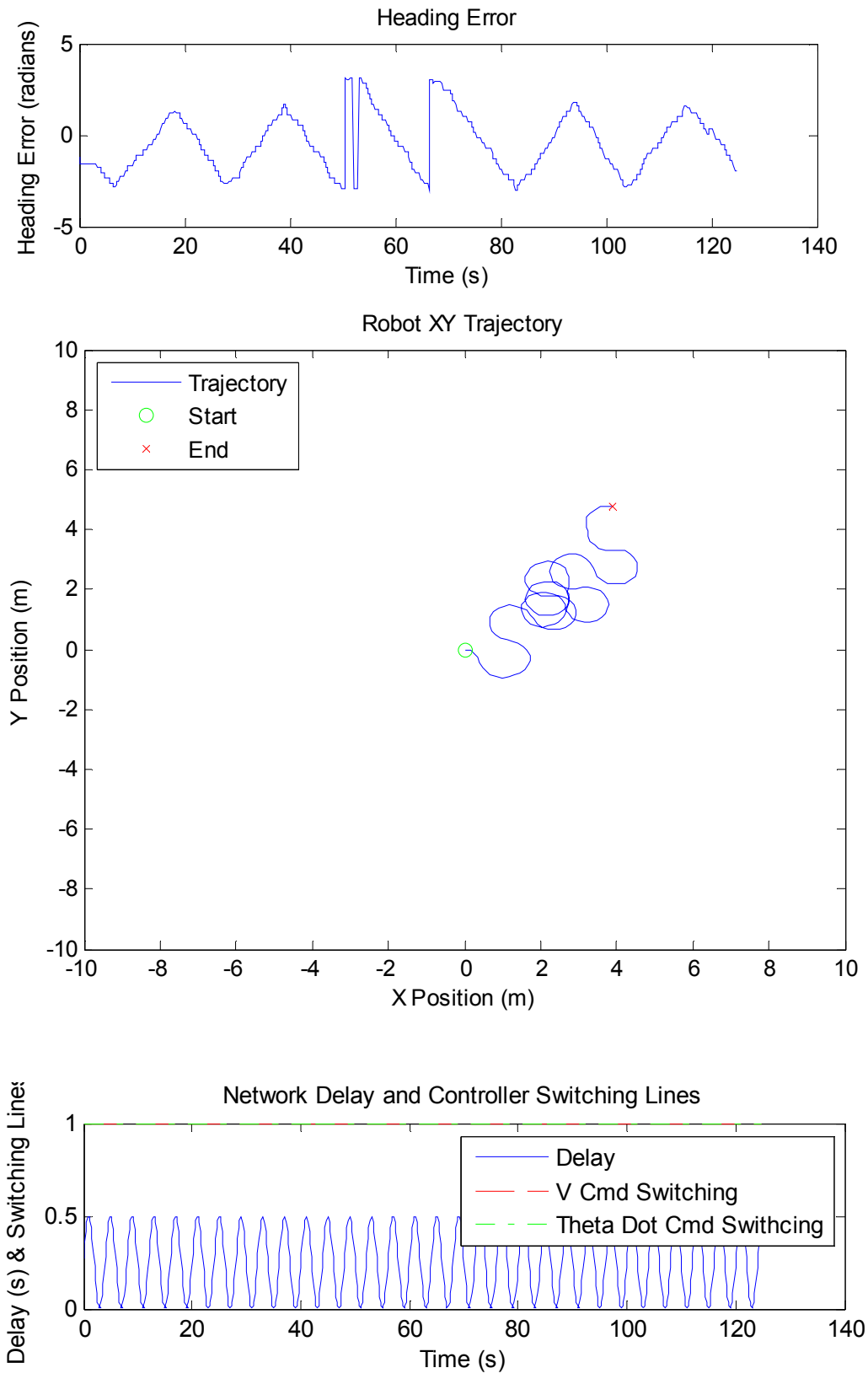
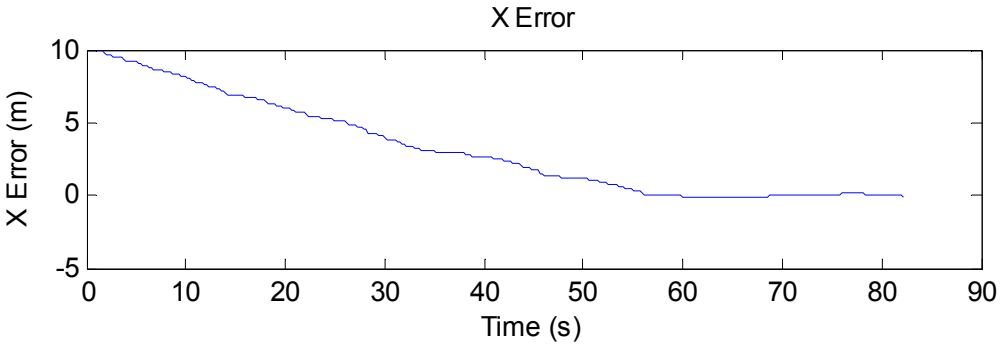
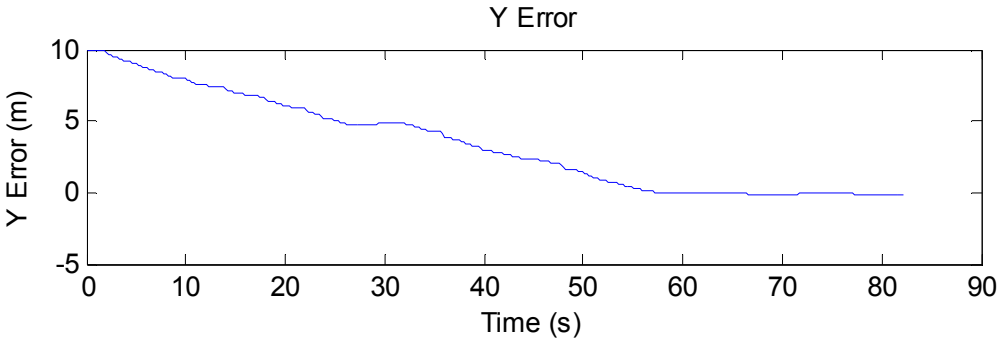


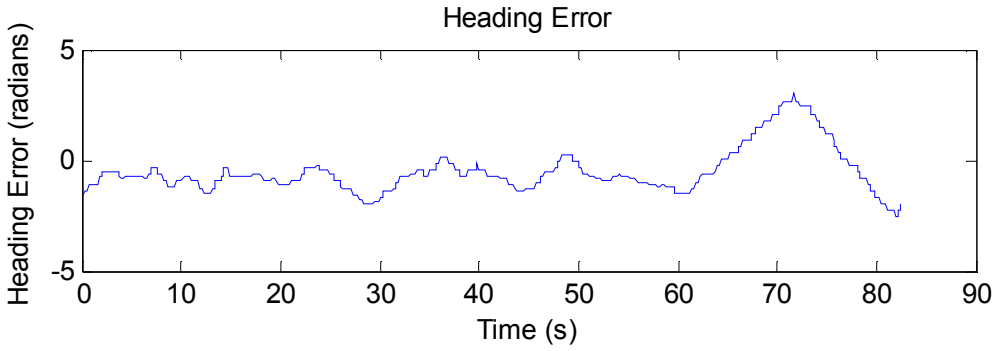
Figure 8 Base Station Control Simulation Results

3.3.3 Mixed Control

The final simulation run enables the meta-controller to allow switching between the base station and robot control. The time delay magnitude and period are kept the same as the previous run which cause instability in the base station. The controller is able to maintain performance and reach the goal position. The heading error oscillates more than with robot only control, but the goal is still reached in a comparable amount of time.







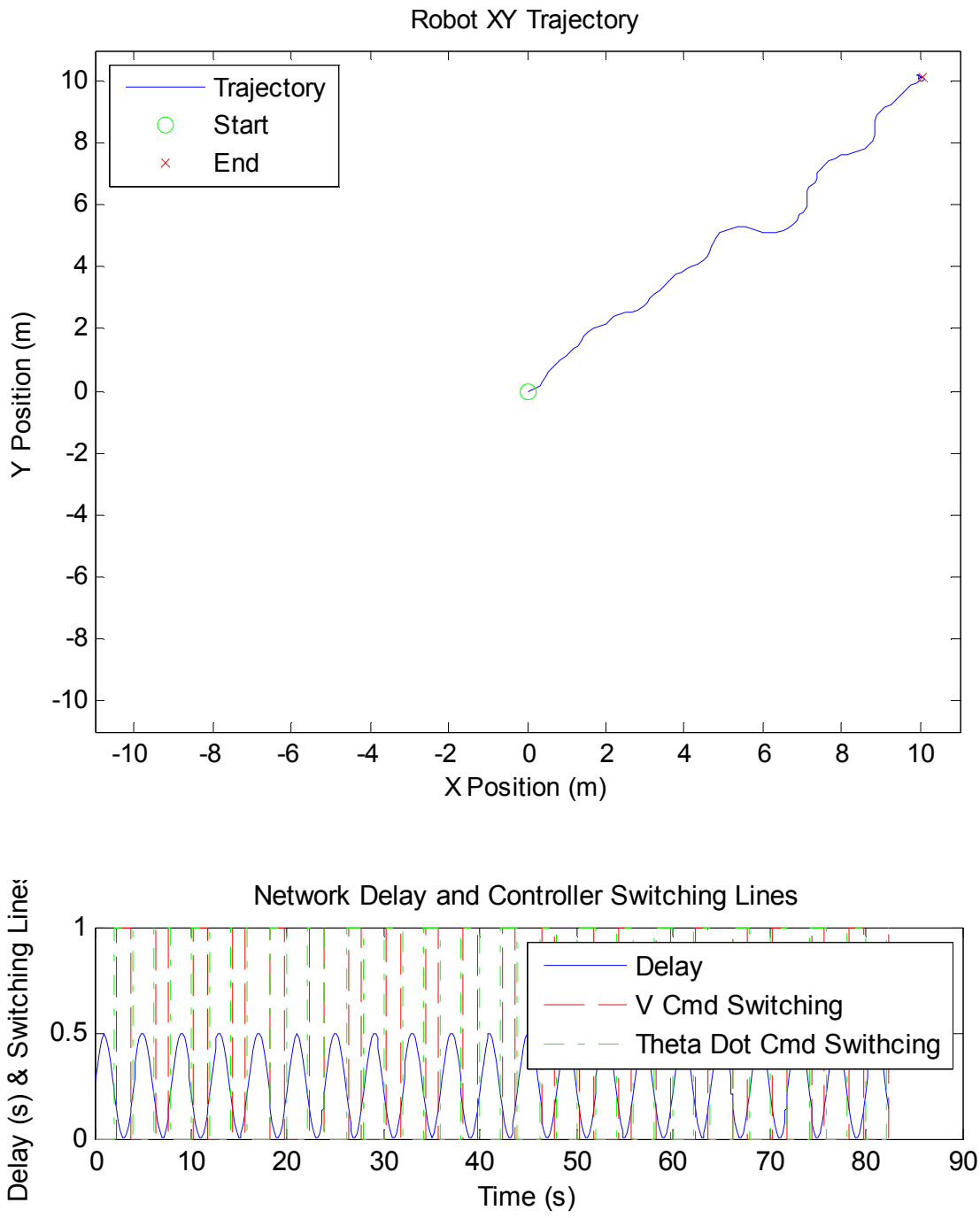


Figure 9 Meta-controller Simulation Results

3.3.4 Transition Effects

The following figure shows the heading rate command after the switching matrix as well as the switching line which selects the heading rate command. It is easy to see that transitioning between the base station and mobile robot can cause a large discontinuity in the command sent to the plant model. The robot dynamics work to dampen this response so the large command jumps appear only as a small wavering in the robot trajectory.

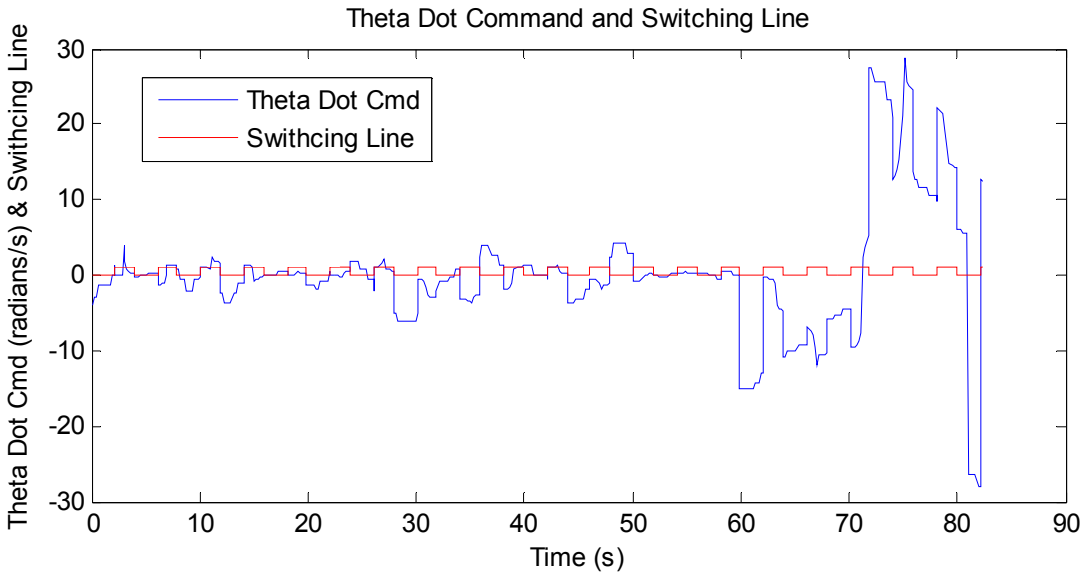


Figure 10 - Switching Effects

3.3.5 Simulation Summary

The simulation results of the meta-controller model show that a simple switching technique can be used to maintain system stability when a network experiences lag or packet loss that was not accounted for in the original design. In the simulation configuration the controller spent 50% of the time using the base station position control loop output and 35% of the time using the base station heading control loop output. The performance of each controller is shown in the following table.

Test	Heading Error Mean (radians)	Heading Error Std. Dev. (radians)	Position Error Mean (m)	Position Error Std. Dev. (m)
Robot	0.05	0.13	5.53	4.83
Base Station	1.38	1.58	11.36	1.53
Meta-Controller	0.97	1.02	4.73	4.54

Table 1 Simulation Results

4 Experimental Test

To test the meta-controller in a real world environment an experimental test bed was developed. A differential drive robot with GPS, compass, microprocessor, and wireless communication link was used as the remote robot. An overview of the test configuration is shown below and the details of each piece are discussed in the following section

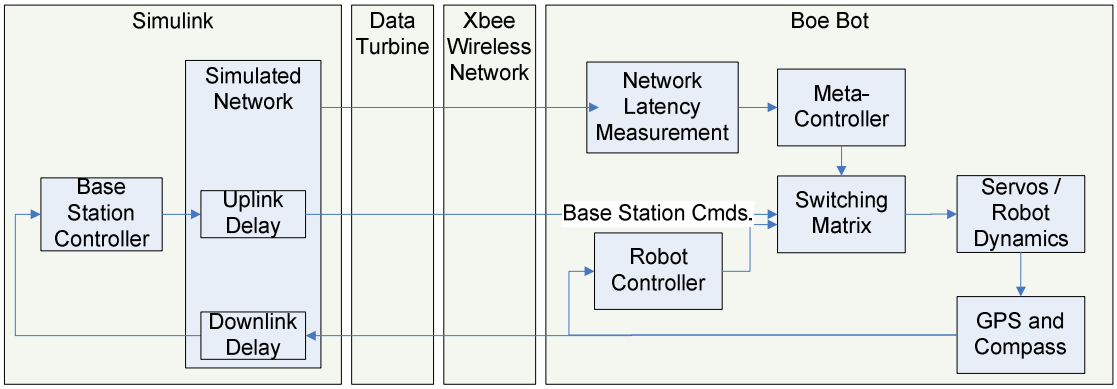


Table 2 Experimental Test Configuration

4.1 Robot Platform

A commercially available kit, the Parallax BoeBot, was used for easy construction and integration with the selected microcontroller development board.

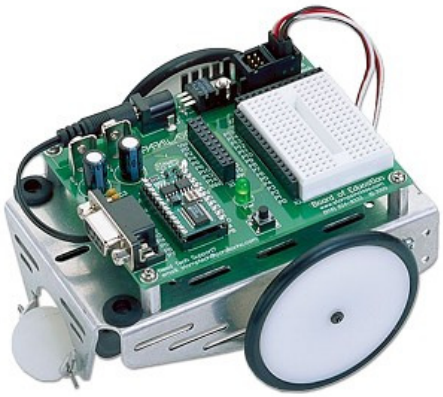


Figure 11 BoeBot

The robot base includes mounting provisions for a Basic Stamp development board. The board includes a small breadboard area which is used to mount and connect sensors and wireless communication.

4.2 Actuation

4.2.1 Servos

The BoeBot uses two continuous rotation hobby servo motors driven by a standard 1.3 to 1.7ms pulse every 20ms. The servos can be driven in both forward and reverse so the BoeBot is capable of any radius turn from zero (turn in place) to infinity (straight line).

4.2.2 ServoPal

The selected microcontroller is only capable of single loop execution and cannot guarantee the servo commands will be sent every 20ms. A Parallax ServoPal is used to receive commands from the microcontroller and maintain the appropriate pulse commands at the correct rate to the servos.

Figure 12 Servo Pal

The ServoPal connects directly between the standard BasicStamp servo connection and the servo wires and is capable of controlling two servos. The ServoPal will repeat any pulse width generated by the microcontroller between 0.5 and 2.5ms which is more than adequate for the BoeBot application.

4.3 Sensors

4.3.1 CMPS03 Digital Compass

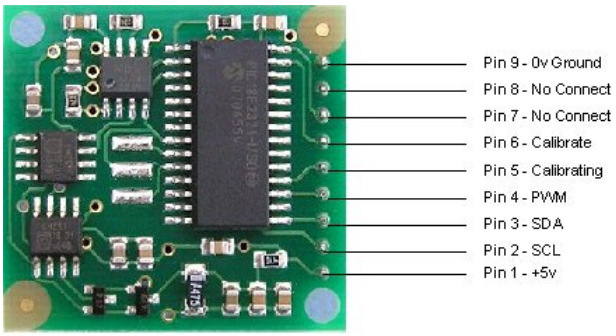


Figure 13 Digital Compass

The CMPS03 compass is a two axis digital compass capable of resolving heading down to 1.4°. The compass module outputs either over an I^2C bus or a variable pulse width on a single pin. The I^2C bus is used for noise immunity in this application. Heading output is given in binary radians (BRADs) which count from 0 to 255 startin from North and increasing in the clockwise direction. This one byte range allows for easy interpretation by the Basic Stamp as well as simple use of the Basic Stamp trigonometric functions which take BRADs as input units. The compass module can also be calibrated to adjust for differences between magnetic and true north based on location. The module is shipped with a calibration value for a location close to our test location so a recalibration is not required. This means that the heading output of the compass is a value from true north and can be used directly for navigation purposes.

4.3.2 Parallax GPS Receiver

Figure 14 GPS Receiver

The Parallax GPS Receiver is a standard GPS module capable of tracking up to 12 satellites simultaneously. It automatically locks onto GPS signals and calculates GPS and time coordinates as soon as 4 satellites are discovered. It is designed for easy integration with the BasicStamp and communicates over a single wire 4800 baud serial port. The module is capable of outputting raw NMEA0183 strings or internally parsing the data and transmitting only requested pieces of information. For our application we will be requesting only the latitude and longitude values. The module advertises an average position error of +/-5m but in practice provided much lower accuracy.

Unfortunately the 4800 baud serial connection takes about 150ms to read a single set of latitude and longitude coordinates. This becomes the limiting factor in control loop speed for the BasicStamp. To mitigate this time lag, a second basic stamp is added to the robot and used only to read serial data from the GPS and forward it over a faster serial link to the primary stamp. The primary stamp is free to run its control loops at a faster rate and update the GPS coordinates when new information becomes available. This

means that the primary stamp may execute several cycles of the control loops on the same GPS data but with new heading data, making the inner heading control loop higher bandwidth than the outer loop.

4.4 Communication Link

4.4.1 XBee 802.15.4

Figure 15 XBee Module

Platform	XBee® 802.15.4 (Series 1)
Performance	
RF Data Rate	250 kbps
Indor/Urban Range	100 ft (30 m)
Outdoor/RF Line-of-Sight Range	300 ft (100 m)
Transmit Power	1 mW (+0 dBm)
Receiver Sensitivity (1% PER)	-92 dBm

Table 3 XBee Specifications

A pair of XBee 802.15.4 radios is used to establish a wireless serial communication link between the base station and remote robot. The wireless serial link is a 38.4Kbaud bidirectional point to point serial link. The XBee modules have approximately a 300 ft outdoor range, but do suffer from interference problems and noise. A simple XOR

checksum is used to determine data packet integrity. This method works well for low levels of interference but begins to fail if obstacles are present between the robot and base station. Wireless through obstacles is not required for this test so the XOR checksum is sufficient to remove any corrupted data.

4.5 Microcontrollers

4.5.1 Parallax Basic Stamp 2

The Basic Stamp 2 module is a 20MHz processor that runs a PBasic interpreter at ~4,000 instructions per second. The Basic Stamp 2 and PBasic langue provide an easy to use development environment with many useful built in commands for communication interfaces and servo control. The Basis Stamp 2 is limited to running a single threaded application so all communication, control calculations, and actuator control must be done in a linear fashion.

A second Basic Stamp 2 module is used as a serial communication accelerator between the GPS module and primary control Basic Stamp. This Stamp performs no other control or communication functions.

4.6 Base Station

The base station application is run in Simulink on a standard Windows XP laptop. The laptop provides a mobile, easy to use, and powerful platform for the base station application.

4.6.1 XBee 802.15.4

The second XBee wireless communication module is used in conjunction with a USB to Serial converter breakout board and connected to the base station laptop computer. The computer sees this wireless link as a standard COM port.

4.6.2 Data Turbine

Data Turbine is a communication application used to connect the Simulink simulation to the serial COMM port. It is used to facilitate easy switching between the simulation setup discussed in section 3 and the real world experimental setup.

4.6.3 Simulink

The base station controller runs as a Simulink model and is nearly identical to the base station half of the simulation discussed in Section 3. The robot control portion is removed and other small modifications are made to properly format the commands for the BoeBot as opposed to the MobileSim model robot. The control gains as well as the meta-controller transition threshold are set via trial and error. A high level view of the controller is shown on the next page.

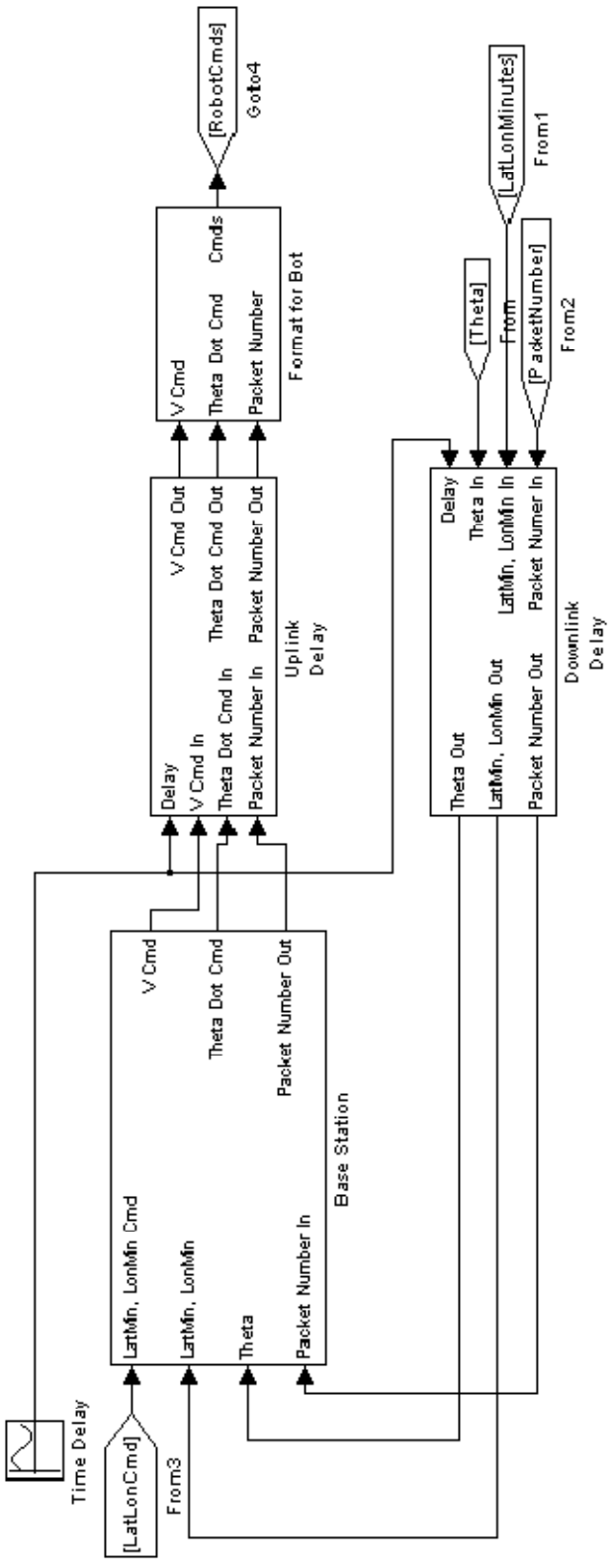


Figure 16 Base Station Controller Model

4.7 Experimental Results

4.7.1 GPS Issues

The Parallax GPS module did not perform to the +/-5m accuracy which was advertised. Stationary GPS data gathered over a 5 minute period had a standard deviation of 19m. The GPS module also exhibited around a 1 minute lag between the time the module was moved and when the coordinates would center on the new location.

Multiple filtering techniques were attempted. The most successful was a low pass filter which brought the stationary standard deviation down under 8m. This filter also added approximately one more minute of lag to the GPS centering time when the module was moved.

In test runs performed with the low pass filter the robot was able to head almost directly for the goal position but would begin veering away approximately half way to the goal. It was never able to reach the goal position. The closest run brought it to within approximately 6m of the goal. Furthermore, the trajectory produced from logged GPS data did not match well with the actual path taken by the robot. The GPS trajectory showed significantly exaggerated movements as the GPS location would jump around the actual robot location. Finally, the GPS data showed that the robot had reached and passed its goal several times even though this did not actually happen.

It was concluded that the GPS error was too large compared to the size and speed of the mobile robot. The GPS error also affected the analysis of the heading control loop because jumps in GPS location would cause command step inputs to the heading control loop.

4.7.2 Heading Control Loop Testing

The entire position control loop was removed from the base station and mobile robot controllers to allow for accurate testing of the meta-controller for the heading control loop. The heading control loop was given periodic 90° step command inputs with sufficient time between step inputs for the control loops to settle. The runs mimic the simulation runs with the first being robot only control, the second base station control, and the third using the meta-controller. The results from each run are presented in the following sections.

4.7.3 Robot Control Baseline

The self controlled robot performs very well. The heading control loop is able to quickly respond to the step input most of the time with little overshoot. The compass heading wraps from zero to two pi radians at due North so the heading actual vs. heading command looks significantly worse than it really is. The heading error gives a much more accurate perspective on the controller performance. Averaging the response characteristics of all the individual step commands gives a mean overshoot of 10%, mean settling time of 2.6s, and mean time to peak of 2.3 seconds. The maximum overshoot is approximately 63% but this is an outlier as all other step responses have less than 20% overshoot. Results are shown below.

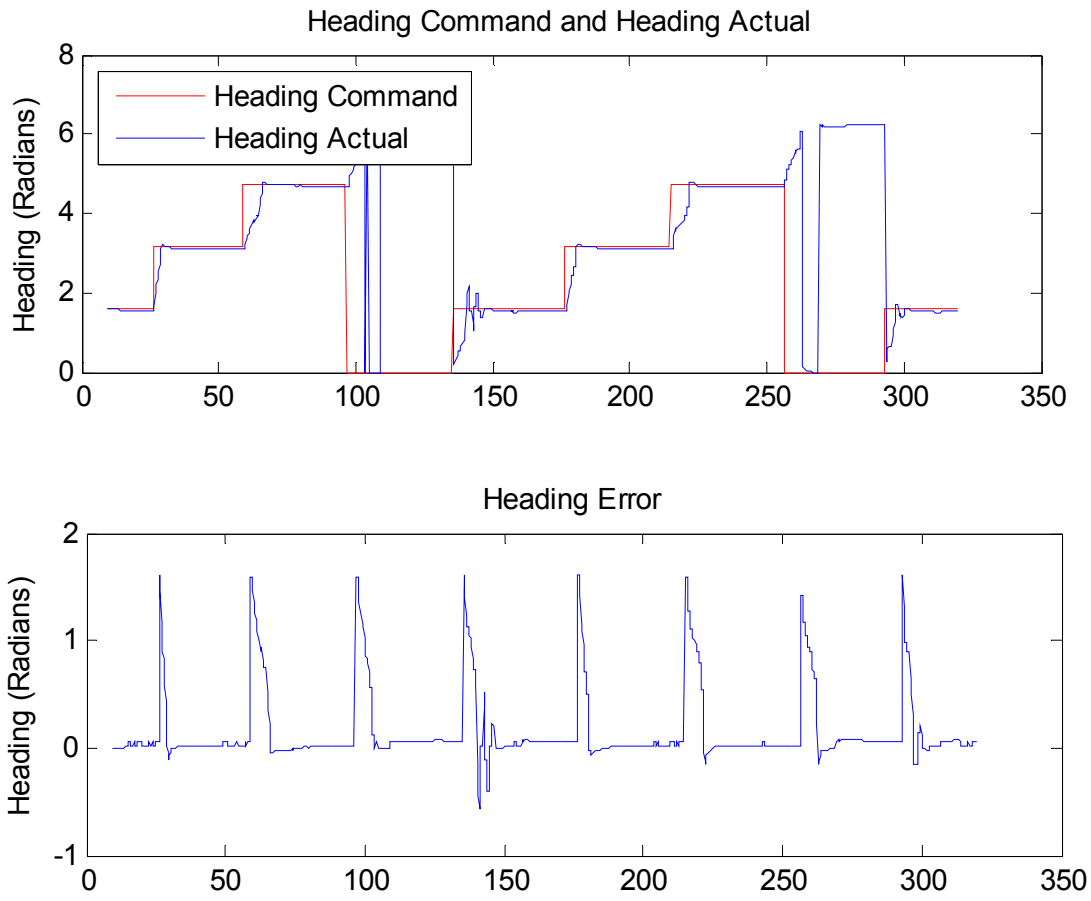


Figure 17 Robot Control Results

4.7.4 Base Station Control Without Delay

In this test the robot is forced to listen to the base station command with no artificial time delay between the base station and robot. This test will establish a baseline performance for base station control under normal network conditions. When the robot is listening to base station commands it skips the control loop section of code so the overall execution rate for the control loops is faster than when under robot control. However, the network latency has a stronger detrimental affect than the increased execution rate so the control performance is worse than on-board robot control. The mean overshoot is 17.7%, mean time to peak is 3.2s, and mean settling time is 5.3s. The results are shown below.

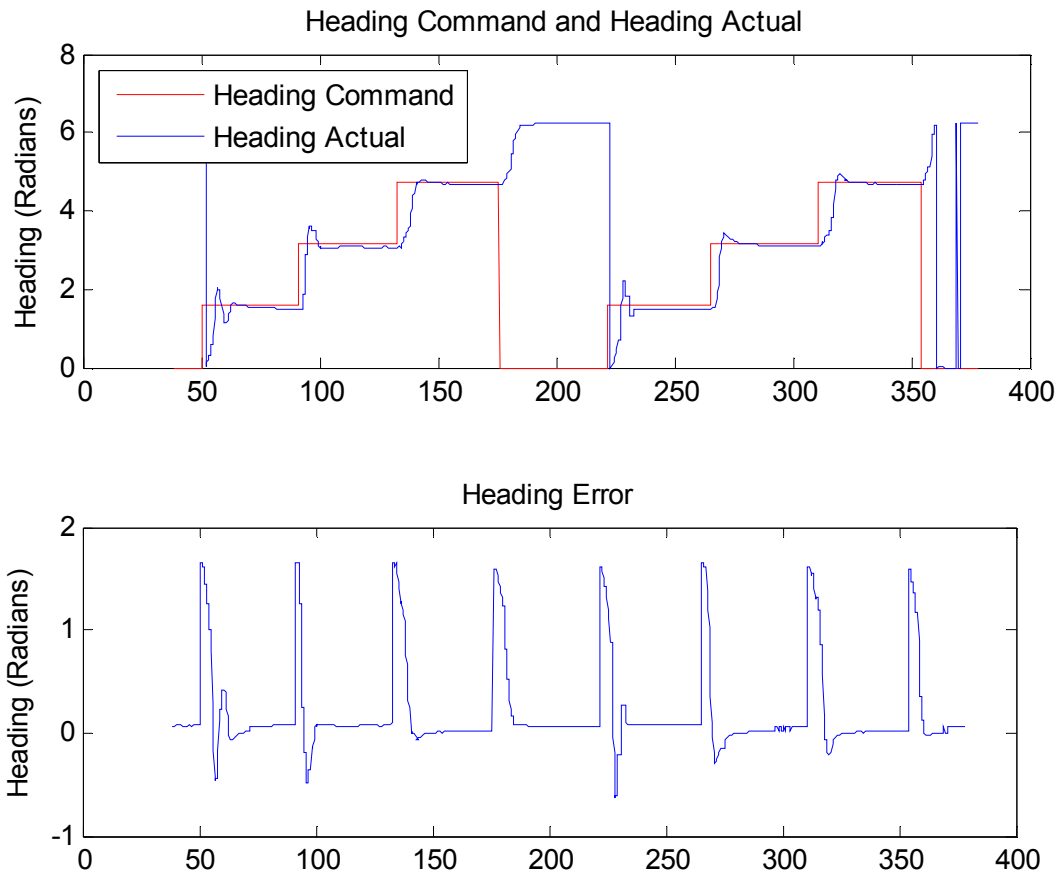


Figure 18 Base Station Control Results

4.7.5 Base Station Control With Delay

This test continues to force the robot to listen to base station commands and adds a sinusoidal network delay on top of the baseline network latency. The time delay is enough to cause system instability when large heading errors are present, but the robot is able to recover during periods of short delay time. About 30% of the step responses do not stabilize. The results are adjusted to exclude the unstable responses. The mean overshoot of the stable responses is 76%, mean time to peak is 3.4s, and mean settling time is 10.5s. The results are below.

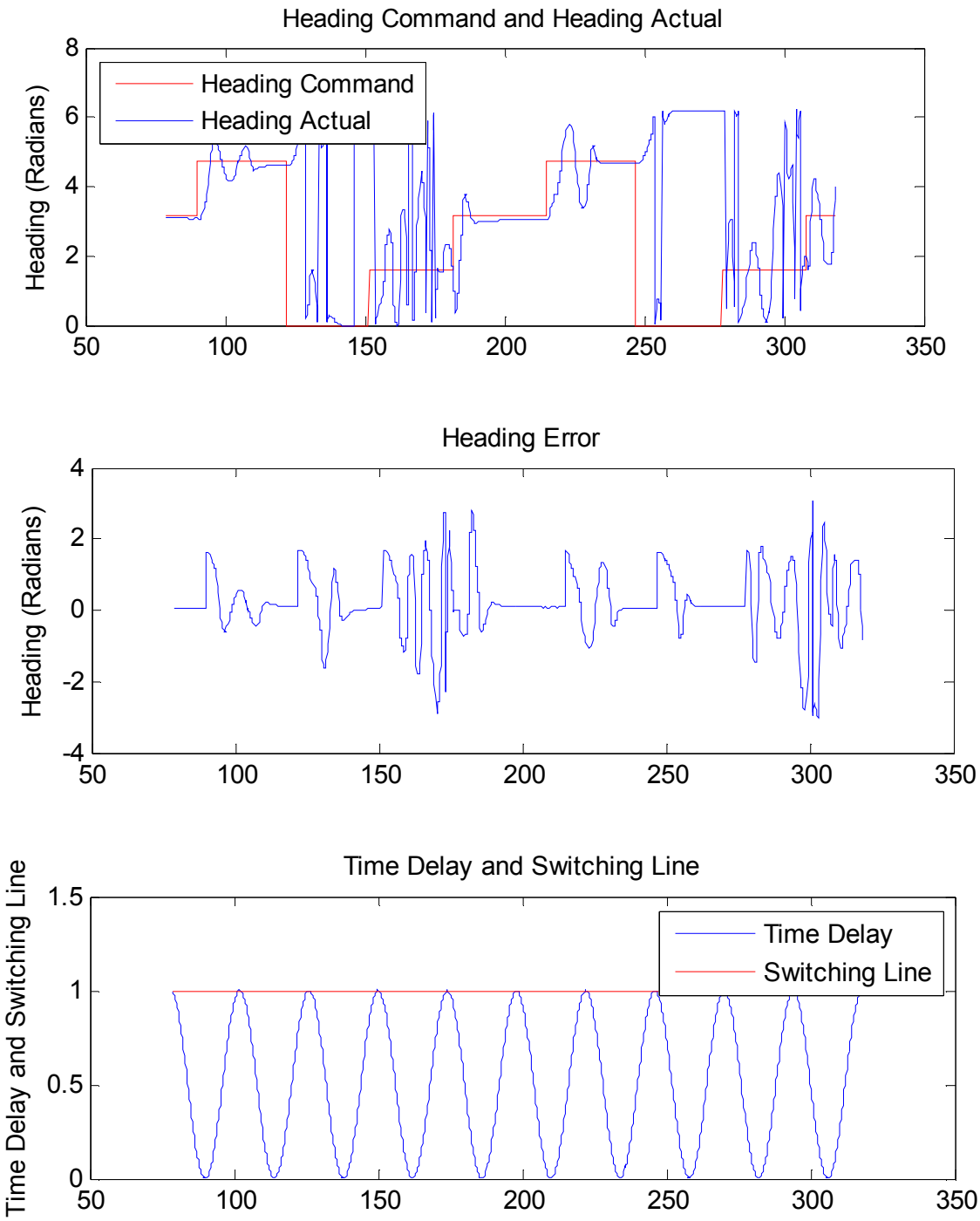


Figure 19 Base Station with Delay Results

4.7.6 Meta-Controller

The same added network delay signal as the base station control with delay test is used, but now the meta-controller is enabled on the robot to allow switching between command sources.

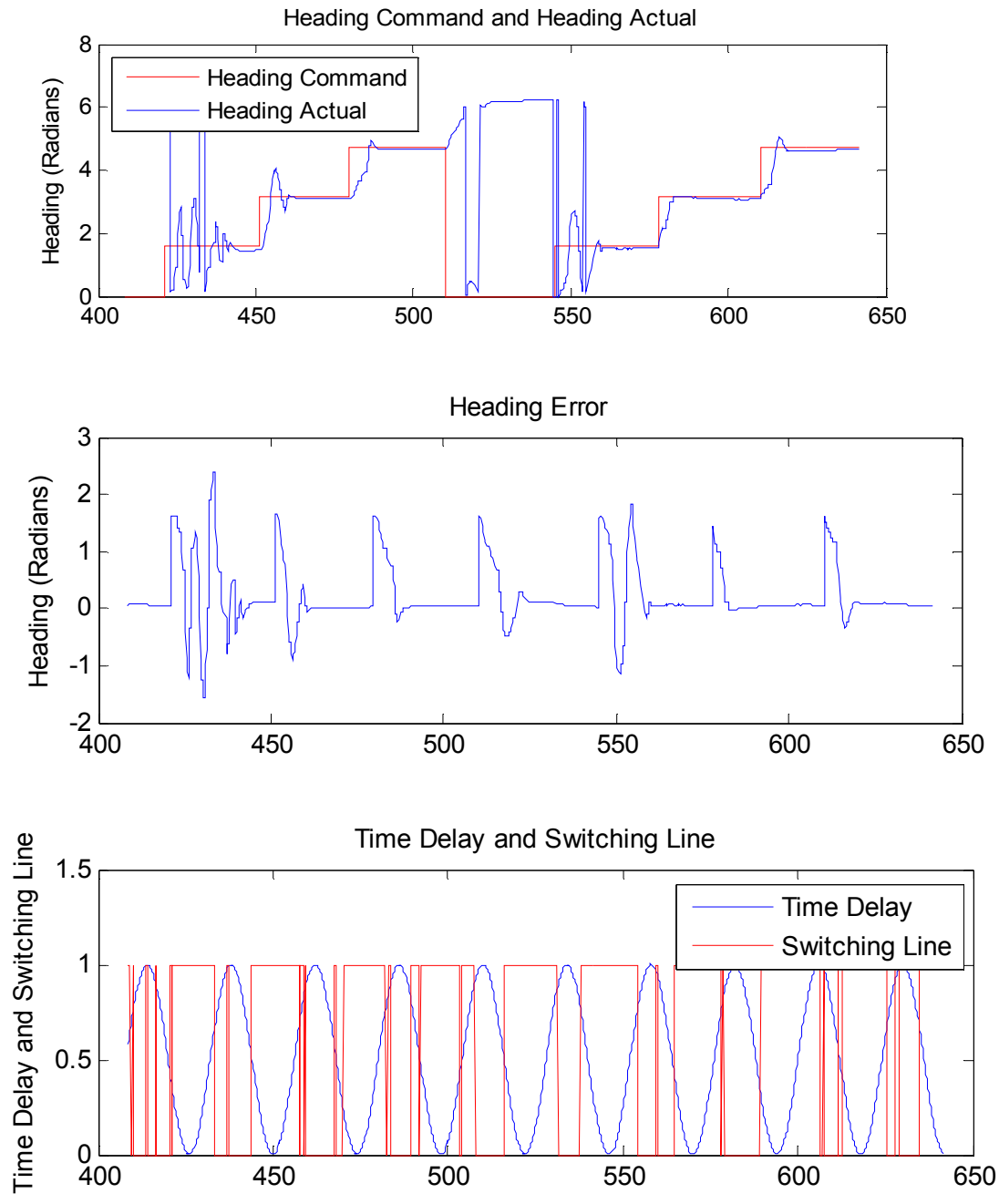


Figure 20 Meta-controller Results

With the meta-controller enabled, the system is stable and remains under base station control for a majority of the time. The controller spends 62% of the simulation time listening to the base station controller and the remaining 38% listening to the robot controller. The mean overshoot is 56.9%, mean time to peak is 3.9s, and mean settling time is 5.9s. The meta-controller had several responses with 100% overshoot but half were less than 63%.

4.7.7 Summary of Results

Test	Time to Peak (s)	Settling Time (s)	Overshoot (%)
Robot	2.3	2.6	10.1
Base Station	3.2	5.3	17.7
Base Station w/ Delay (Includes only stable responses)	3.4	10.5	76.0
Meta-Controller w/ Delay	3.9	5.9	56.9

Table 4 Test Results

The meta-controller with delay performs significantly better than the base station alone with the delay. It does not meet the performance of either the base station in normal network conditions or the robot controller, but this is expected because the network must first start to lag before control is switched over to the robot.

The meta-controller achieves the desired result of maintaining base station control as much as possible without sacrificing system stability. The transition between base station and robot control is very smooth for this system.

5 Conclusion

5.1 Summary

The meta-controller proposed in this paper has shown that it is a valid solution for networked control systems where communication lag and failure cannot cause loss of control. The controller is capable of smoothly transitioning back and forth between base station and remote operation as network performance moves in and out of the parameters of the classical control system design. The overall performance of the system is comparable to the networked or remote only systems. This system can be used for automatically transitioning between base station and remotely executed control loops as operator system confidence grows.

The mobile robot test bed used for this application was not an ideal system to test against. The cart's size and speed compared to GPS inaccuracies meant that the robot was never able to reach the goal position. This forced the testing to be constrained to the inner heading control loop. The meta-controller was able to stabilize the heading control loop while still spending 62% of the time under base station control with performance comparable to the networked control system operating under normal network conditions.

5.2 Future Work

There are several areas of the meta-controller which can be explored in more depth. First, a different system with both inner and outer loops could be developed to test the real world response of the meta-controller with nested loop controllers.

Secondly, more research can be done into applying the meta-controller model to control systems with derivative and integral control terms. These were only discussed in the theory section of the paper and were not tested in simulation or practical application. These alternate control methods may also require more research into smoothing the transition between the two controllers.

Finally, alternate switching methods could be developed and applied to the same meta-controller structure. Some ideas include a switching threshold with hysteresis or a controller which monitors system performance instead of network delay.

References

[1] H. W. Stone, “Mars Pathfinder Microrover. A Small, Low-cost, Low-power Spacecraft,” Jet Propulsion Laboratory, Pasadena, CA, 1996

[2] G. M. H. Leung, B. A. Francis, J. Apkarian, “Bilateral Controller for Teleoperators with Time Delay via μ-Synthesis,” Dept. of Electrical and Computer Engineering, University of Toronto, Ontario, Canada, 1997

[3] S. Thrun, W. Burgard, D. Fox, “A Read-Time Algorithm for Mobile Robot Mapping with Applications to Multi-Robot and 3D Mapping,” Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. Computer Science Department, University of Freiburg, Freiburg, Germany, 2000

[4] M. S. Branicky, S. M. Phillips, W. Zhang, “Stability of Networked Control Systems: Explicit Analysis of Delay”, EECS Department, Case Western Reserve University, Cleveland, OH, 2000

[5] W. Zhang, M. S. Branicky, S. M. Phillips, “Stability of Networked Control Systems,” IEEE Control Systems Magazine, Feb. 2001

[6] J. Rao and N. Sadeh, “Semantic Web Framework and Meta-Control Model to Enforce Context-Sensitive Policies,” School of Computer Science, Carnegie Mellon Univerisy, Pittsburgh, PA

[7] G. Alexander, A. Raja, E. H. Durfee, and D. J. Musliner, “Design Paradigms for Meta-Control in Multiagent Systems,” Department of Software and Information Systems, The University of North Carolina, Charlotte, NC. EECS Department, University of Michigan, Ann Arbor, MI. Honeywell Laboratories, Minneapolis, MN.

[8] R. M. Rasay, "A Graphical Model-Based Reasoning Analysis Environment for Space System Anomaly Management," C. Kitts, Adv., M.S. Thesis, Department of Mechanical Engineering, Santa Clara University, Santa Clara, CA, June 2007.

[9] X. Liu and A. Goldsmith, “Wireless Network Design for Distributed Control”, Department of Elec. Eng., Stanford University, Stanford, CA, Jan. 2004

[10] M. S. Branicky, S. M. Phillips, and W. Zhang, "Scheduling and Feedback Co-Design for Networked Control Systems", Proc. IEE CDC, Las Vegas, 2002

[11] M. Jenkin, "Computational Principles of Mobile Robotics", Cambridge University Press, 2000

[12] X. Liu, A. Goldsmith, "Wireless Medium Access Control in Networked Control Systems", Department of Elec. Eng., Stanford University, Stanford, CA, Jan. 2004

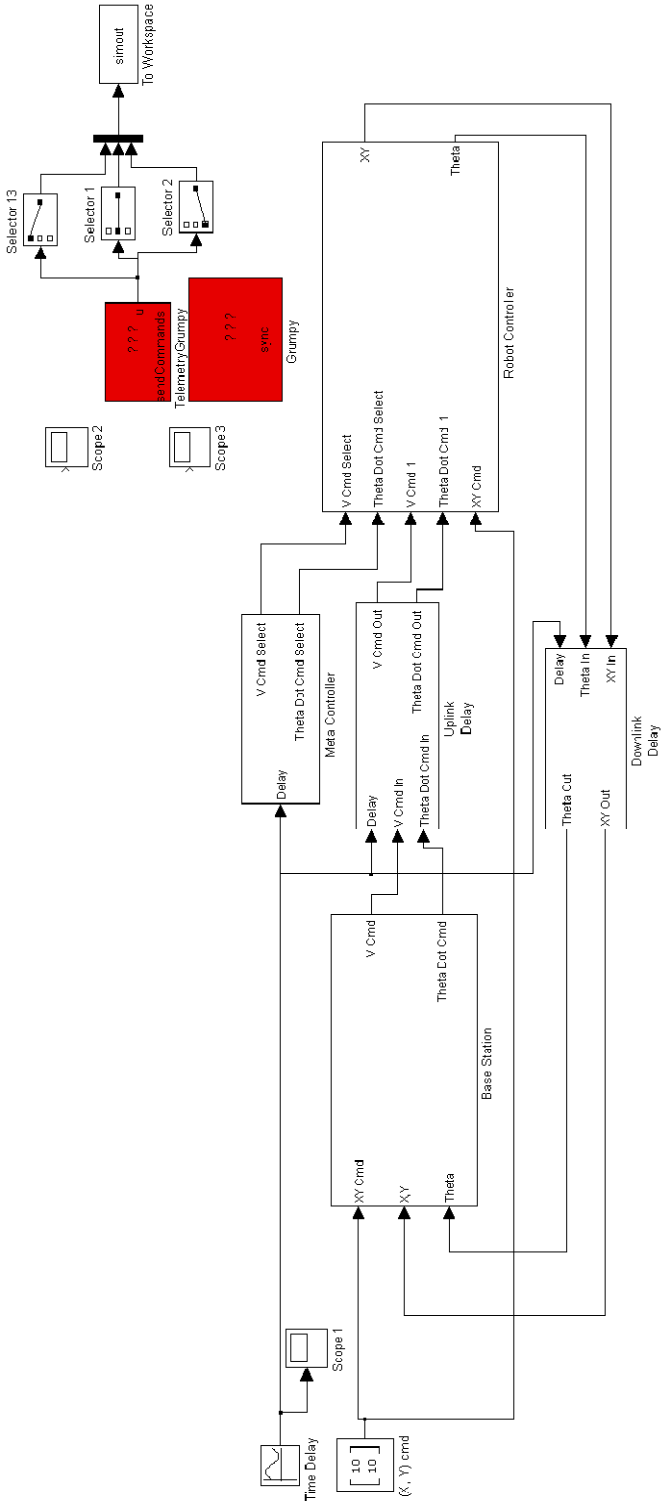
[13] Y. Tipsuwan, M.Y. Chow, "Control Methodologies in Networked Control Systems", Department of Elec. And Computer Eng., North Carolina State University, Raleigh, NC, Feb. 2003

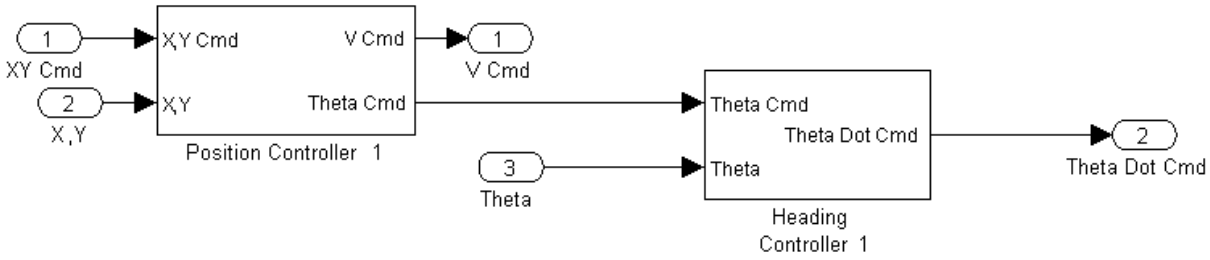
[14] L. Zhongkui, D. Zhisheng, H. Lin, "H ∞ control of networked multi-agent systems," Department of Mechanical and Aerospace Engineering, Peking University, Beijing, China, 2009

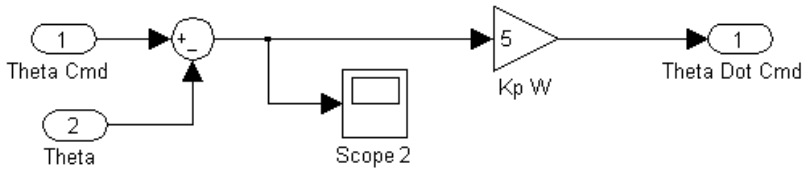
[15] J. J. Biesiadecki, C. Leger, M. W. Maimone, "Tradeoffs Between Directed and Autonomous Driving on the Mars Exploration Rovers," Jet Propulsion Laboratory, Pasadena, CA, 2005

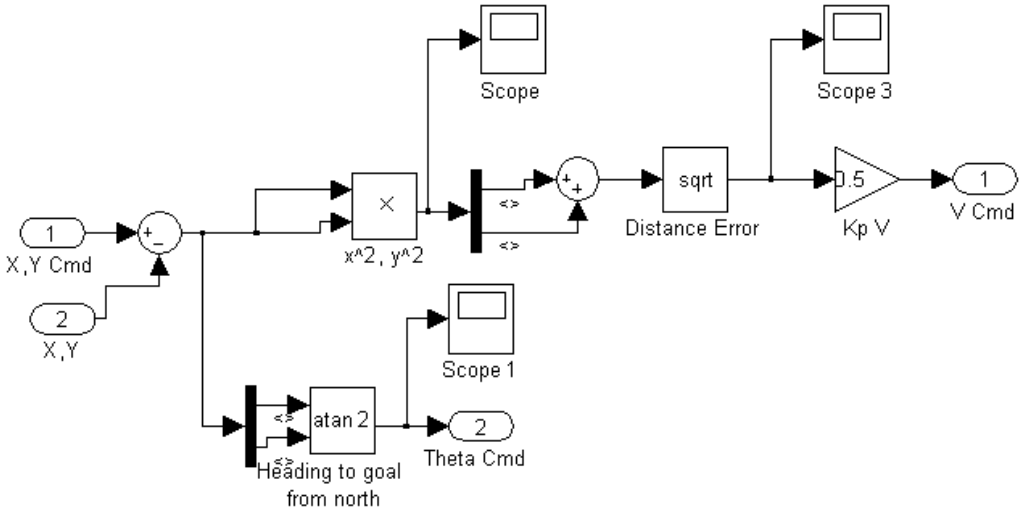
[16] J. Nilsson, "Read-Time Control Systems with Delays," Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 1998

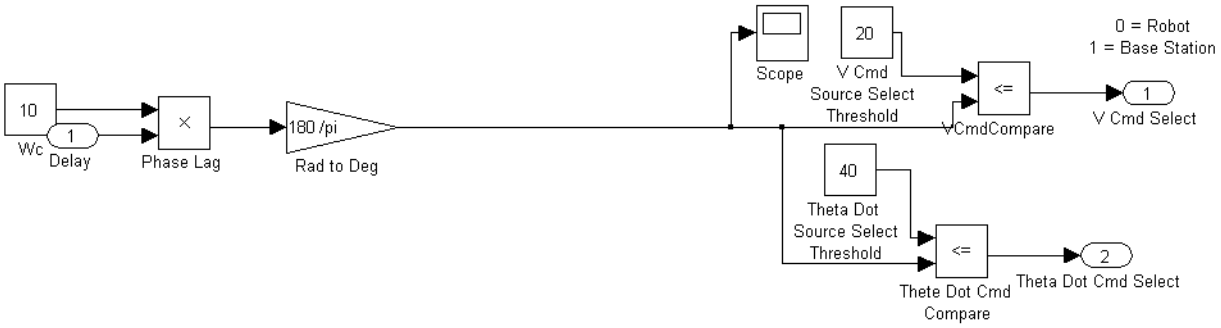
Appendix A – Meta-Controller Simulink Model

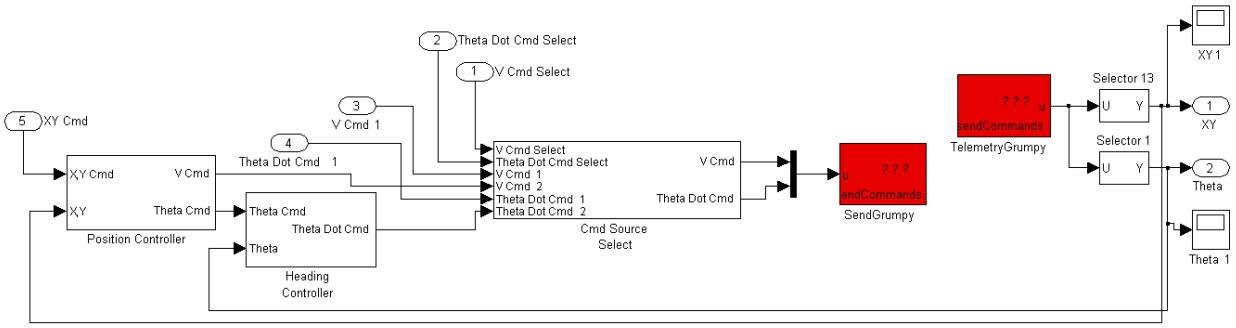


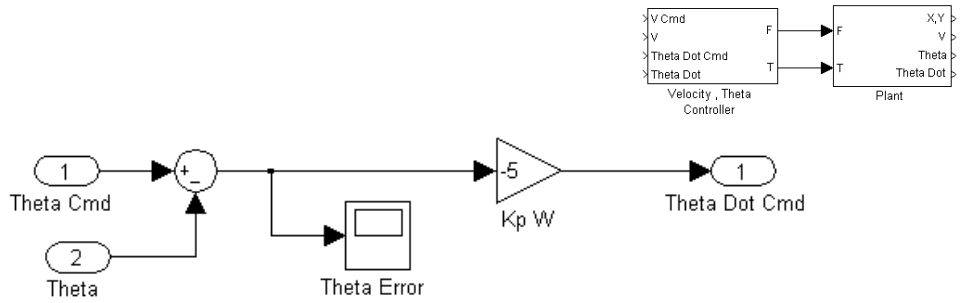

1
XY Cmd
2
X,Y
X,Y Cmd
XY
V Cmd
Theta Cmd
Position Controller 1
1
V Cmd
3
Theta
Theta Cmd
Theta Dot Cmd
Theta
Heading
Controller 1
2
Theta Dot Cmd

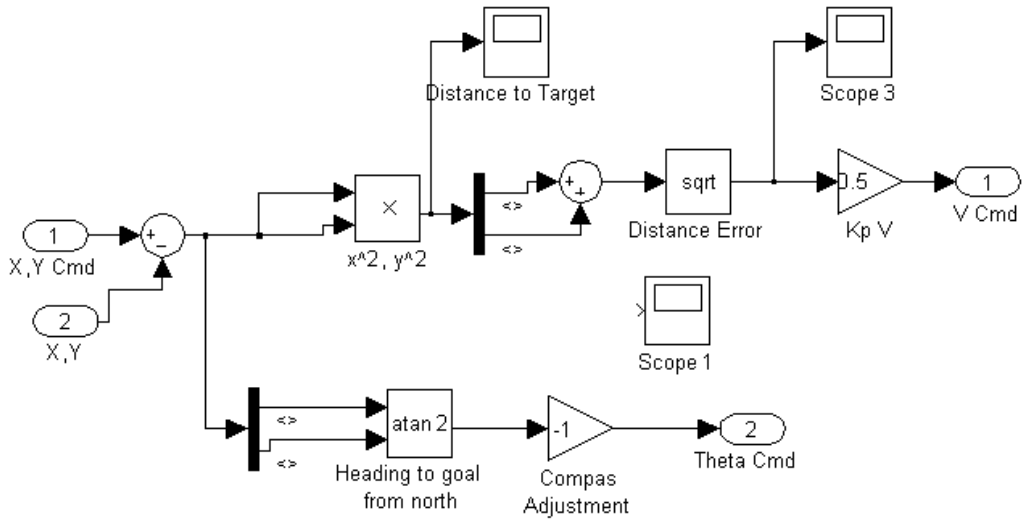

1
Theta Cmd
2
Theta
Scope 2
5
Kp W
1
Theta Dot Cmd

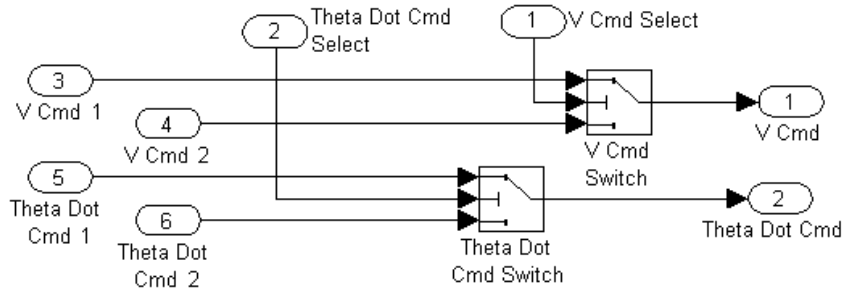

Scope
Scope 3
1
X,Y Cmd
2
X,Y
x^2, y^2
sqrt
Distance Error
0.5
Kp V
1
V Cmd
Scope 1
atan 2
Heading to goal
from north
2
Theta Cmd

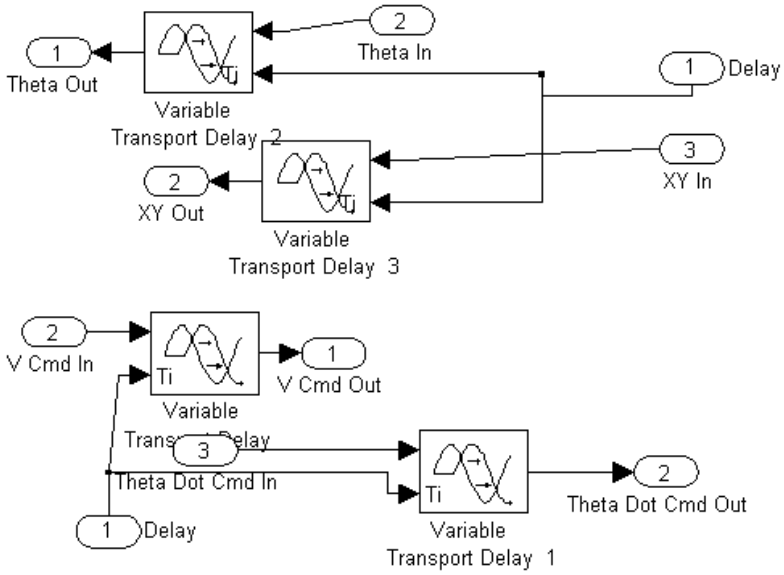

10
Wc
1
Delay
Phase Lag
180/pi
Rad to Deg
Scope
20
V Cmd
Source Select
Threshold
<=
VCmdCompare
0 = Robot
1 = Base Station
1
V Cmd Select
40
Theta Dot
Source Select
Threshold
<=
Thete Dot Cmd
Compare
2
Theta Dot Cmd Select


2
Theta Dot Cmd Select
1
V Cmd Select
3
V Cmd 1
5
XY Cmd
4
Theta Dot Cmd 1
XY Cmd
V Cmd
XY
Theta Cmd
Position Controller
Theta Cmd
Theta Dot Cmd
Theta
Heading Controller
V Cmd Select
Theta Dot Cmd Select
V Cmd 1
V Cmd 2
Theta Dot Cmd 1
Theta Dot Cmd 2
V Cmd
Theta Dot Cmd
Cmd Source Select
SendGrumpy
TelemetryGrumpy
Selector 13
Selector 1
XY 1
XY
Theta
Theta 1

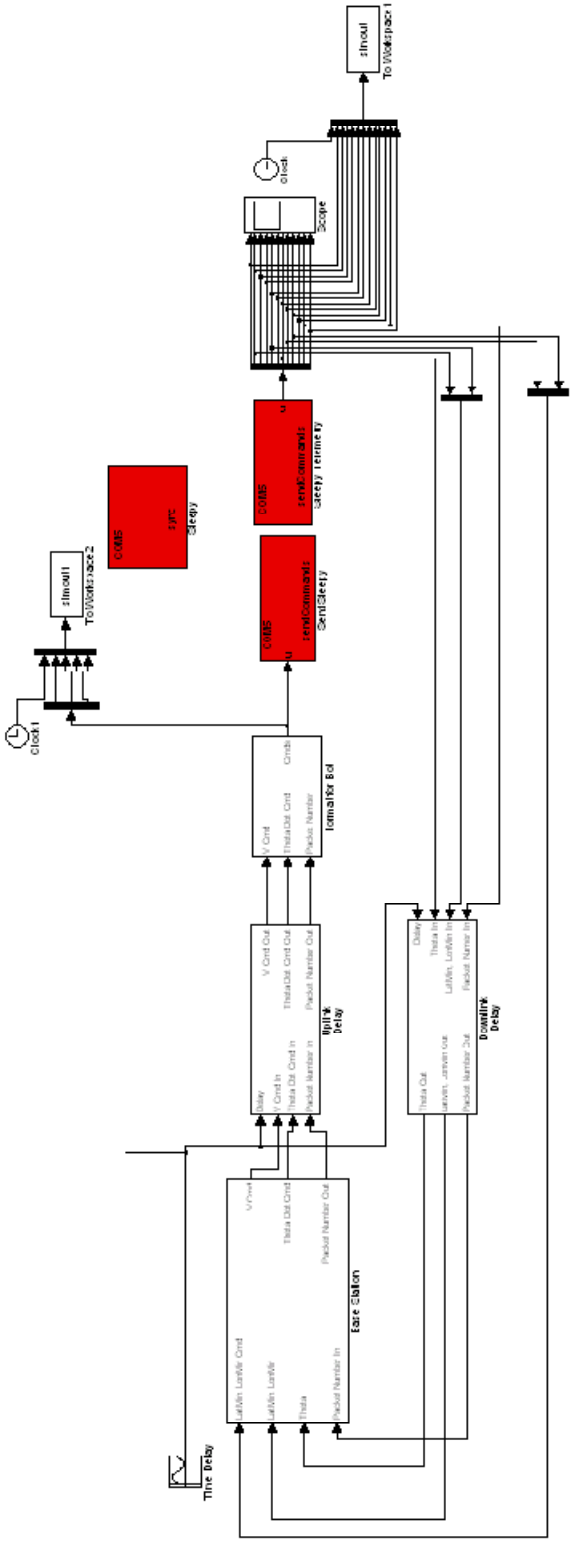

V Cmd
V
Theta Dot Cmd
Theta Dot
F
T
Velocity , Theta Controller
X,Y
V
Theta
Theta Dot
Plant
1
Theta Cmd
2
Theta
-5
Kp W
Theta Error
1
Theta Dot Cmd

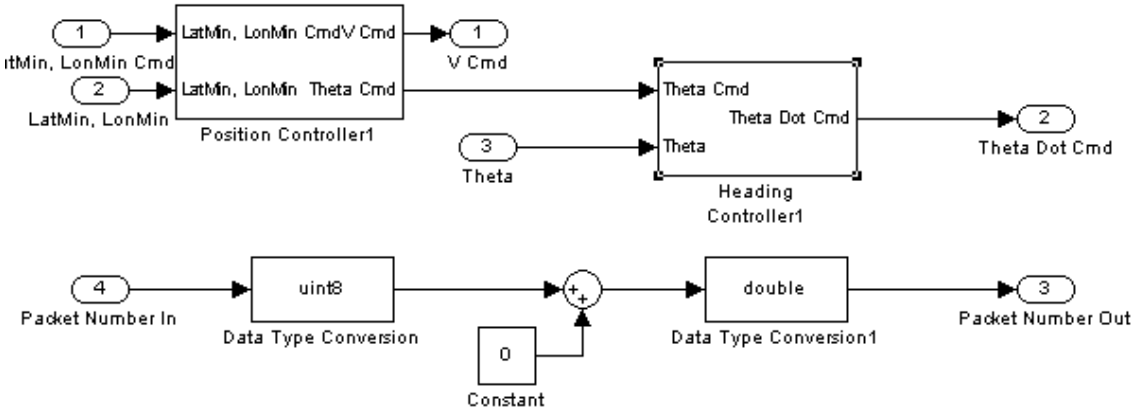

Distance to Target
Scope 3
1
X,Y Cmd
2
X,Y
x^2, y^2
sqrt
Distance Error
0.5
Kp V
1
V Cmd
Scope 1
atan 2
Heading to goal from north
-1
Compas Adjustment
2
Theta Cmd

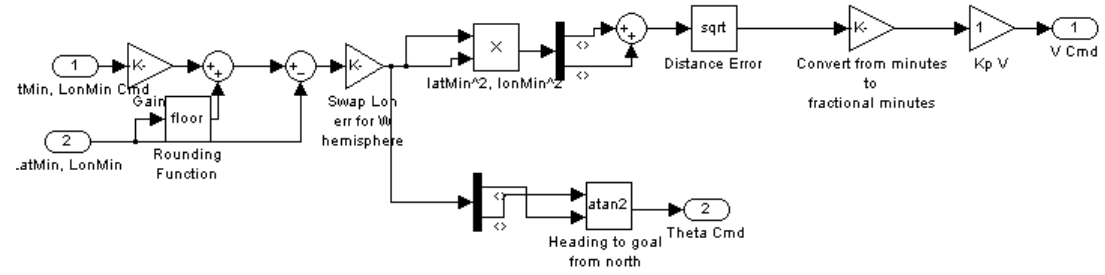

2
Theta Dot Cmd Select
1
V Cmd Select
3
V Cmd 1
4
V Cmd 2
5
Theta Dot Cmd 1
6
Theta Dot Cmd 2
V Cmd Switch
1
V Cmd
Theta Dot Cmd Switch
2
Theta Dot Cmd

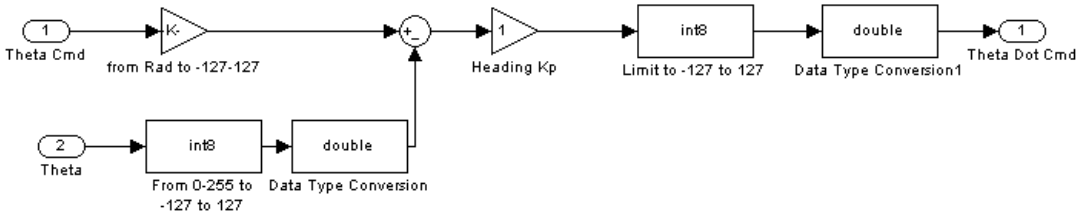

2
Theta In
1
Theta Out
Variable
Transport Delay 2
1
Delay
3
XY In
2
XY Out
Variable
Transport Delay 3
2
V Cmd In
Ti
1
V Cmd Out
Variable
3
Theta Dot Cmd In
Ti
2
Theta Dot Cmd Out
1
Delay
Variable
Transport Delay 1

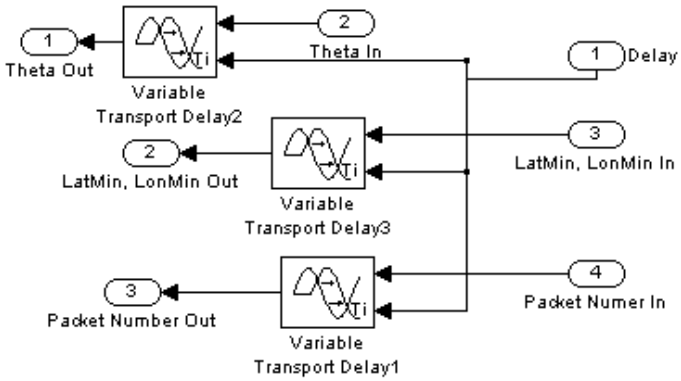
Appendix B – Base Station Simulink Model

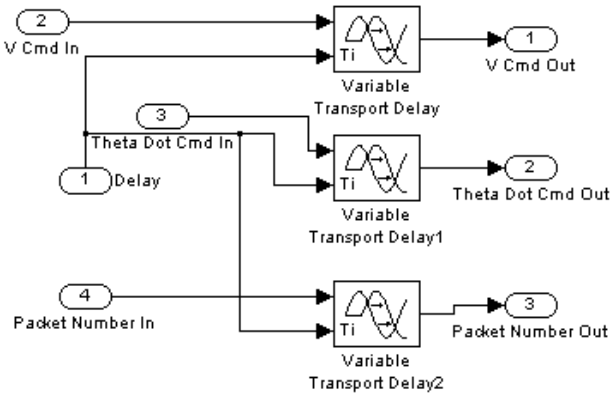


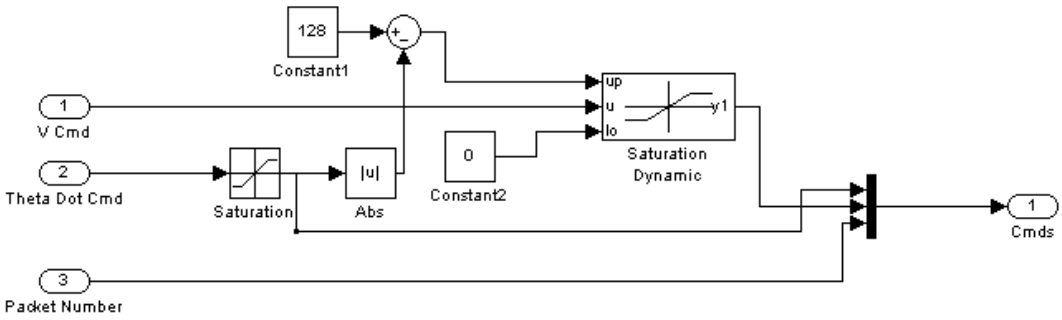

1
LatMin, LonMin Cmd
2
LatMin, LonMin
LatMin, LonMin CmdV Cmd
LatMin, LonMin Theta Cmd
Position Controller1
1
V Cmd
Theta Cmd
Theta Dot Cmd
Theta
Heading Controller1
3
Theta
2
Theta Dot Cmd
4
Packet Number In
uint8
Data Type Conversion
0
Constant
double
Data Type Conversion1
3
Packet Number Out


1
tMin, LonMin Cmd
Gain
2
atMin, LonMin
floor
Rounding Function
Swap Lon err for W hemisphere
latMin^2, lonMin^2
sqrt
Distance Error
Convert from minutes to fractional minutes
1
Kp V
1
V Cmd
atan2
2
Theta Cmd
Heading to goal from north


1
Theta Cmd
from Rad to -127-127
1
Heading Kp
int8
Limit to -127 to 127
double
Data Type Conversion1
1
Theta Dot Cmd
2
Theta
int8
From 0-255 to -127 to 127
double
Data Type Conversion


1
Theta Out
Variable
Transport Delay2
2
Theta In
1
Delay
2
LatMin, LonMin Out
Variable
Transport Delay3
3
LatMin, LonMin In
3
Packet Number Out
Variable
Transport Delay1
4
Packet Numer In


2
V Cmd In
Variable
Transport Delay
1
V Cmd Out
3
Theta Dot Cmd In
1
Delay
Variable
Transport Delay1
2
Theta Dot Cmd Out
4
Packet Number In
Variable
Transport Delay2
3
Packet Number Out


128
Constant1
1
V Cmd
2
Theta Dot Cmd
Saturation
Abs
0
Constant2
up
u
lo
y1
Saturation
Dynamic
1
Cmds
3
Packet Number

Appendix C – Basic Stamp Code

Robot Controller – Heading Control Only

```
' {$STAMP BS2}
' {$PBASIC 2.5}

SDA CON 10
SCL CON 11

RightServo CON 12
LeftServo CON 13

'--------------------------------------
' Temporary Variables
'--------------------------------------
'Used for I2C read/write buffer
TmpByte1        VAR Byte

'Used for receiving speed command
'Used for latDegrees
'Used for minutesError
'Used for errorLon
TmpByte2        VAR Byte

'Used for receiving packet count
'Used for errorLat
'Used for lonDegrees
TmpByte3        VAR Byte

'Used for receiving checksum
'Used for minutesDError
'Used for distanceError
TmpWord1        VAR Word

'Used to track sign bit
'Used for I2C Ack/Nak
sign            VAR Bit

'--------------------------------------
' Servo Pal
'--------------------------------------
ServoPalInp     PIN 12
ServoPalAlarm   PIN 13

ServoPalResetTime   CON 100
ZeroSpeedPulseWidth CON 750

'--------------------------------------
' Compass
'--------------------------------------
```

```
SDAin VAR IN10
SDAout VAR OUT10
SDAdir VAR DIR10

I2cData VAR Word                        ' Data to read/write
I2cAddr CON $c0                         ' Address of I2C device
I2cReg CON 1                            ' Register number within I2C
device

'---------------------------------
' General Serial
'---------------------------------
#SELECT $STAMP
  #CASE BS2, BS2E, BS2PE
    T1200       CON     813
    T2400       CON     396
    T9600       CON     84
    T19K2       CON     32
    T38K4       CON     6
  #CASE BS2SX, BS2P
    T1200       CON     2063
    T2400       CON     1021
    T9600       CON     240
    T19K2       CON     110
    T38K4       CON     45
  #CASE BS2PX
    T1200       CON     3313
    T2400       CON     1646
    T9600       CON     396
    T19K2       CON     188
    T38K4       CON     84
#ENDSELECT

'-------------------------------------------
' XBee Serial Variables and Consts
'-------------------------------------------
XBeeDataOut PIN 2
XBeeDataIn  PIN 3

XBeeBaud    CON T38K4

headingError VAR Byte
headingCmd     VAR Byte
bsHeadingCmd        VAR Byte
tmpBsHeadingCmd     VAR Byte

packetNumber VAR Byte
controlLoopSelector VAR Bit
packetNumberDifferenceThreshold CON 7

headingKp     CON 1

'-------------------------------------------
' Control loop input
'-------------------------------------------
desiredHeading   VAR Byte
headingIncrement CON 64
```

```
headingIncrementPeriod CON 128
headingIncrementCounter VAR Word


Main:
PAUSE 1000
DEBUG "Program Start", CR
SEROUT XBeeDataIn, XBeeBaud, ["Program Start", CR]

PAUSE 1000

'Reset ServoPal
LOW ServoPalInp
PAUSE ServoPalResetTime
HIGH ServoPalInp

INPUT SDA
INPUT SCL
INPUT RightServo
INPUT LeftServo

packetNumber = 0
controlLoopSelector = 1
headingCmd = 0

desiredHeading = 0
headingIncrementCounter = 0

StartingWait:
PAUSE 1000
DEBUG "."


Loop_Start:
DO

  IF headingIncrementCounter >= headingIncrementPeriod THEN
    headingIncrementCounter = 0
    desiredHeading = desiredHeading + headingIncrement
  ENDIF
  headingIncrementCounter = headingIncrementCounter + 1

  'Read Heading
  'Stored in TmpByte1
  GOSUB i2cByteRead

  SEROUT XBeeDataIn, XBeeBaud,
[187,TmpByte1,desiredHeading,packetNumber,ControlLoopSelector,

TmpByte1^desiredHeading^packetNumber^ControlLoopSelector, CR]

  packetNumber = packetNumber + 1

  'Read commands from basestation
  SERIN XBeeDataOut, XBeeBaud, [tmpBsHeadingCmd]
  DO WHILE tmpBsHeadingCmd <> 187
    'Keep reading until we see a start code
```

```
    SERIN XBeeDataOut, XBeeBaud, [tmpBsHeadingCmd]
  LOOP

  'READ heading command, COUNT, AND checksum
  SERIN XBeeDataOut, XBeeBaud, [tmpBsHeadingCmd, TmpByte3,
TmpWord1.HIGHBYTE]
  IF (tmpBsHeadingCmd^TmpByte3) <> TmpWord1.HIGHBYTE THEN
    'DEBUG "Checksum error", CR
  ELSE
    'DEBUG "Heading Cmd: ", DEC tmpBsHeadingCmd, " Count: ", DEC
TmpByte3, " Checksum: ", DEC TmpWord1.HIGHBYTE, CR

    bsHeadingCmd = tmpBsHeadingCmd

    'Decide to use local or remote commands
    IF packetNumber - TmpByte3 > packetNumberDifferenceThreshold THEN
     'Check to wrap condition
      IF 256 - TmpByte3 + packetNumber <
packetNumberDifferenceThreshold THEN
        'Use basestation control
        controlLoopSelector = 1
      ELSE
        'Use robot control
        controlLoopSelector = 0
      ENDIF
    ELSE
      'Use basestation control
      controlLoopSelector = 1
    ENDIF
  ENDIF

  IF controlLoopSelector THEN
    'Use on Base station control loops
    IF bsHeadingCmd > 128 THEN
      'removed direction bit from heading to goal
      bsHeadingCmd = 255 - bsHeadingCmd
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth - bsHeadingCmd
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth - bsHeadingCmd

      'Restore headingCmd in case no command is received next loop
      bsHeadingCmd = 255 - bsHeadingCmd
    ELSE
      'DEBUG "Heading cmd: ", DEC headingCmd, " dist cmd: ", DEC
velCmd, CR
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth + bsHeadingCmd
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth + bsHeadingCmd

    ENDIF

  ELSE
    'Use on robot control loops

    'Calculate heading error
    headingError = desiredHeading - I2cData

    IF headingError > 128 THEN
      headingError = 255 - headingError
```

```
      headingError = (headingError*headingKp) MAX 128

      PULSOUT ServoPalInp, ZeroSpeedPulseWidth - headingError
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth - headingError
    ELSE
      headingError = (headingError*headingKp) MAX 128
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth + headingError
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth + headingError
    ENDIF
  ENDIF

LOOP

END


'-----------------------------------------------------------------------
----------------------
' I2C subroutines follow
'-----------------------------------------------------------------------
----------------------

I2cByteWrite:                           ' writes I2cData.lowbyte to
I2cReg at I2cAddr
GOSUB I2cStart
TmpByte1 = I2cAddr
GOSUB I2cOutByte                        ' send device address
TmpByte1 = I2cReg
GOSUB I2cOutByte                        ' send register number
TmpByte1 = I2cData.LOWBYTE
GOSUB I2cOutByte                        ' send the data
GOSUB I2cStop
RETURN

I2cWordWrite:                           ' writes I2cData to I2cReg at
I2cAddr
GOSUB I2cStart
TmpByte1 = I2cAddr
GOSUB I2cOutByte                        ' send device address
TmpByte1 = I2cReg
GOSUB I2cOutByte                        ' send register number
TmpByte1 = I2cData.HIGHBYTE
GOSUB I2cOutByte                        ' send the data - high byte
TmpByte1 = I2cData.LOWBYTE
GOSUB I2cOutByte                        ' send the data - low byte
GOSUB I2cStop
RETURN

I2CByteRead:
GOSUB I2cStart
TmpByte1 = I2cAddr
GOSUB I2cOutByte                        ' send device address
TmpByte1 = I2cReg
GOSUB I2cOutByte                        ' send register number
GOSUB I2cStart                          ' repeated start
TmpByte1 = I2cAddr | 1
```

```
GOSUB I2cOutByte                          ' send device address (with
read set)
sign = 0                                ' send Nak
GOSUB I2cInByte
I2cData.LOWBYTE = TmpByte1                  ' read the data
I2cData.HIGHBYTE = 0
GOSUB I2cStop
RETURN

I2CWordRead:
GOSUB I2cStart
TmpByte1 = I2cAddr
GOSUB I2cOutByte                          ' send device address
TmpByte1 = I2cReg
GOSUB I2cOutByte                          ' send register number
GOSUB I2cStart                            ' repeated start
TmpByte1 = I2cAddr | 1
sign = 1                                ' send Ack
GOSUB I2cOutByte                          ' send device address (with
read set)
GOSUB I2cInByte
I2cData.HIGHBYTE = TmpByte1                 ' read the data
sign = 0                                ' send Nak
GOSUB I2cInByte
I2cData.LOWBYTE = TmpByte1
GOSUB I2cStop
RETURN

I2cOutByte:
SHIFTOUT SDA, SCL, MSBFIRST, [TmpByte1]
INPUT SDA
HIGH SCL                                  ' clock in the ack' bit
LOW SCL
RETURN

I2cInByte:
SHIFTIN SDA, SCL, MSBPRE, [TmpByte1]
SDAout = 0
SDAdir = sign
HIGH SCL                                  ' clock out the ack' bit
LOW SCL
INPUT SDA
RETURN

I2cStart:                                  ' I2C start bit sequence
HIGH SDA
HIGH SCL
LOW SDA
LOW SCL
RETURN

I2cStop:                                  ' I2C stop bit sequence
LOW SDA
HIGH SCL
HIGH SDA
RETURN
```

Robot Controller – Navigation Controller

```
' {$STAMP BS2}
' {$PBASIC 2.5}

SDA CON 10
SCL CON 11

RightServo CON 12
LeftServo CON 13

'--------------------------------------
' Temporary Variables
'--------------------------------------
'Used for I2C read/write buffer
TmpByte1        VAR Byte

'Used for receiving speed command
'Used for latDegrees
'Used for minutesError
'Used for errorLon
TmpByte2        VAR Byte

'Used for receiving packet count
'Used for errorLat
'Used for lonDegrees
TmpByte3        VAR Byte

'Used for receiving checksum
'Used for minutesDError
'Used for distanceError
TmpWord1        VAR Word

'Used to track sign bit
'Used for I2C Ack/Nak
sign            VAR Bit

'--------------------------------------
' Servo Pal
'--------------------------------------
ServoPalInp     PIN 12
ServoPalAlarm   PIN 13

ServoPalResetTime   CON 100
ZeroSpeedPulseWidth CON 750

'--------------------------------------
' Compass
'--------------------------------------

SDAin VAR IN10
SDAout VAR OUT10
SDAdir VAR DIR10

I2cData VAR Word                        ' Data to read/write
```

```
I2cAddr CON $c0                         ' Address of I2C device
I2cReg CON 1                            ' Register number within I2C
device

'---------------------------------
' General Serial
'---------------------------------
#SELECT $STAMP
  #CASE BS2, BS2E, BS2PE
    T1200       CON     813
    T2400       CON     396
    T9600       CON     84
    T19K2       CON     32
    T38K4       CON     6
  #CASE BS2SX, BS2P
    T1200       CON     2063
    T2400       CON     1021
    T9600       CON     240
    T19K2       CON     110
    T38K4       CON     45
  #CASE BS2PX
    T1200       CON     3313
    T2400       CON     1646
    T9600       CON     396
    T19K2       CON     188
    T38K4       CON     84
#ENDSELECT

'---------------------------------
' Stamp-Stamp Serial
'---------------------------------
Bs2SerialData     PIN 6
Bs2SerialFlow     PIN 5

Inverted          CON $4000
Open              CON $8000
Baud              CON T38K4 + Inverted

'-----------------------------------------
' XBee Serial Variables and Consts
'-----------------------------------------
XBeeDataOut PIN 2
XBeeDataIn  PIN 3

XBeeBaud    CON T38K4

'-----------------------------------------
' GPS Variables
'-----------------------------------------
latMinutes          VAR Byte
latMinutesD         VAR Word
latDir              VAR Byte

lonMinutes          VAR Byte
lonMinutesD         VAR Word
lonDir              VAR Byte
```

```
'----------------------------------
' Control Vars
'----------------------------------
'only works at 37 degrees north latitude
desiredDegreesLat CON 37
desiredDirectionLat CON 0
'desired minutes and decimal minutes are variable
desiredMinutesLat VAR Byte
desiredMinutesDLat VAR Word

'only works at 121 degrees west longitude
desiredDegreesLon CON 121
desiredDirectionLon CON 1
'desired minutes and decimal minutes are variable
desiredMinutesLon VAR Byte
desiredMinutesDLon VAR Word

headingToGoal VAR Byte
headingCmd    VAR Byte
velCmd        VAR Byte

packetNumber VAR Byte
controlLoopSelector VAR Bit
packetNumberDifferenceThreshold CON 6

headingKp     CON 1

setDesiredPin     PIN 1

Main:
PAUSE 1000
DEBUG "Program Start", CR
SEROUT XBeeDataIn, XBeeBaud, ["Program Start", CR]

PAUSE 1000

'Reset ServoPal
LOW ServoPalInp
PAUSE ServoPalResetTime
HIGH ServoPalInp

INPUT SDA
INPUT SCL
INPUT RightServo
INPUT LeftServo

packetNumber = 0
controlLoopSelector = 1
velCmd = 0
headingCmd = 0

StartingWait:
PAUSE 1000
DEBUG "."

'Wait for GPS and set as desired
```

```
SERIN Bs2SerialData\Bs2SerialFlow, Baud, 2, StartingWait, [TmpByte2,
latMinutes, latMinutesD.HIGHBYTE, latMinutesD.LOWBYTE, latDir,
TmpByte3, lonMinutes, lonMinutesD.HIGHBYTE, lonMinutesD.LOWBYTE,
lonDir]
desiredMinutesLat = latMinutes
desiredMinutesDLat = latMinutesD
desiredMinutesLon = lonMinutes
desiredMinutesDLon = lonMinutesD

Loop_Start:
DO
  'Read GPS from other Stamp
  'LatDeg, LatMin, LatMinD, LatDir, LonDeg, LonMin, LonMidD, LonDir
  SERIN Bs2SerialData\Bs2SerialFlow, Baud, 2, No_Data_Ready, [TmpByte2,
latMinutes, latMinutesD.HIGHBYTE, latMinutesD.LOWBYTE, latDir,
TmpByte3, lonMinutes, lonMinutesD.HIGHBYTE, lonMinutesD.LOWBYTE,
lonDir]
  Continue_Loop:

  'Read Heading
  'Stored in TmpByte1
  GOSUB i2cByteRead

  'Check if lat/lon should be set as desired
  sign = ~setDesiredPin
  IF sign THEN
    'DEBUG "Setting desired lat/lon", CR
    desiredMinutesLat = latMinutes
    desiredMinutesDLat = latMinutesD
    desiredMinutesLon = lonMinutes
    desiredMinutesDLon = lonMinutesD
  ENDIF

  'Send telemetry to basestation
  'LatDeg, LatMin, LatMinD, LatDir, LonDeg, LonMin, LonMidD, LonDir,
heading, setDesired
  'DEBUG "!",DEC TmpByte2," ",DEC latMinutes,".",DEC latMinutesD,"
",DEC latDir," ",DEC TmpByte3," ",DEC lonMinutes,".",DEC lonMinutesD,"
",DEC lonDir," ",DEC TmpByte1," ",DEC sign, CR
  'DEBUG "?",DEC TmpByte2," ",DEC desiredMinutesLat,".",DEC
desiredMinutesDLat," ",DEC latDir," ",DEC TmpByte3," ",DEC
desiredMinutesLon,".",DEC desiredMinutesDLon," ",DEC lonDir,CR
  SEROUT XBeeDataIn, XBeeBaud,
[187,desiredDegreesLat,latMinutes,latMinutesD.HIGHBYTE,latMinutesD.LOWB
YTE,latDir,desiredDegreesLon,lonMinutes,lonMinutesD.HIGHBYTE,lonMinutes
D.LOWBYTE,lonDir,TmpByte1,desiredMinutesDLat.HIGHBYTE,desiredMinutesDLa
t.LOWBYTE,desiredMinutesDLon.HIGHBYTE,desiredMinutesDlon.LOWBYTE,headin
gToGoal,packetNumber,ControlLoopSelector,

desiredDegreesLat^latMinutes^latMinutesD.HIGHBYTE^latminutesD.LOWBYTE^l
atDir^desiredDegreesLon^lonMinutes^lonMinutesD.HIGHBYTE^lonMinutesd.LOW
BYTE^lonDir^TmpByte1^desiredMinutesDLat.HIGHBYTE^desiredMinutesDLat.LOW
BYTE^desiredMinutesDLon.HIGHBYTE^desiredMinutesDLon.LOWBYTE^headingToGo
al^packetNumber^ControlLoopSelector, CR]

  packetNumber = packetNumber + 1
```

```
  'Read commands from basestation
  SERIN XBeeDataOut, XBeeBaud, [headingToGoal]
  DO WHILE headingToGoal <> 187
    'Keep reading until we see a start code
    SERIN XBeeDataOut, XBeeBaud, [headingToGoal]
  LOOP

  'Read heading command, speed command, count, and checksum
  SERIN XBeeDataOut, XBeeBaud, [headingToGoal, TmpByte2, TmpByte3,
TmpWord1.HIGHBYTE]
  'DEBUG "Received: ", DEC headingToGoal, " ", DEC TmpByte2, " ", DEC
TmpByte3, " ", DEC TmpWord1.HIGHBYTE, CR
  IF (headingToGoal^TmpByte2^TmpByte3) <> TmpWord1.HIGHBYTE THEN
    'DEBUG "Checksum error", CR
  ELSE
    'Good packet
    headingCmd = headingToGoal
    velCmd = TmpByte2
    'DEBUG "heading cmd: ", DEC headingToGoal, " vel cmd: ", DEC
TmpByte2," Packet number: ", DEC packetNumber, CR

    'Decide to use local or remote commands
    IF packetNumber - TmpByte3 > packetNumberDifferenceThreshold THEN
      'Check to wrap condition
      IF 256 - TmpByte3 + packetNumber <
packetNumberDifferenceThreshold THEN
        'Use basestation control
        controlLoopSelector = 1
      ELSE
        'Use robot control
        controlLoopSelector = 0
      ENDIF
    ELSE
      'Use basestation control
      controlLoopSelector = 1
    ENDIF
  ENDIF

  IF controlLoopSelector THEN
    'Use on Base station control loops
    IF headingCmd > 128 THEN
      'removed direction bit from heading to goal
      'velocity cmd is already capped
      headingCmd = 255 - headingCmd
      'DEBUG "Heading cmd: ", DEC headingCmd, " dist cmd: ", DEC
velCmd, CR
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth - headingCmd - velCmd
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth - headingCmd + velCmd
      'Restore headingToGoal to send to basestation
      headingToGoal = headingCmd
      'Restore headingCmd in case no command is received next loop
      headingCmd = 255 - headingCmd
    ELSE
      'DEBUG "Heading cmd: ", DEC headingCmd, " dist cmd: ", DEC
velCmd, CR
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth + headingToGoal - velCmd
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth + headingToGoal + velCmd
```

```
      'Restore headingToGoal to send to basestation
      headingToGoal = headingCmd
    ENDIF

  ELSE
    'Use on robot control loops

    'Calculate Latitude error
    IF latDir <> desiredDirectionLat THEN
      DEBUG "Only works in NW hemisphere.  Sorry.", CR
      END
    ENDIF

    'Calculate Longitude
    IF lonDir <> desiredDirectionLon THEN
      DEBUG "Only works in NW hemisphere.  Sorry.", CR
      END
    ENDIF

    'MinutesError = desiredMinutesLat - latMinutes
    TmpByte2 = desiredMinutesLat - latMinutes
    'MinutesDError = desiredMinutesDLat - latMinutesD
    TmpWord1 = desiredMinutesDLat - latMinutesD

    'MinutesDError cap to fit in byte errorLat
    sign = TmpWord1.BIT15
    TmpByte3 = (ABS TmpWord1) MAX 127
    IF sign THEN
      'Twos compliment to fix sign of errorLat
      TmpByte3 = (~TmpByte3)+1
    ENDIF

    'MinutesError = desiredMinutesLon - lonMinutes
    TmpByte2 = desiredMinutesLon - lonMinutes
    'MinutesDError = desiredMinutesDLon - lonMinutesD
    TmpWord1 = desiredMinutesDLon - lonMinutesD

    'MinutesDError cap to fit in byte errorLon
    sign = TmpWord1.BIT15
    TmpByte2 = (ABS TmpWord1) MAX 127
    'Invert sign on longitude because we are in the western hemisphere
and
    'the positive direction is West
    IF sign = 0 THEN
      'Twos compliment to fix sign of errorLon
      TmpByte2 = (~TmpByte2)+1
    ENDIF


    'Calculate desired heading
    'Swap X and Y to convert to compass heading
    headingToGoal = TmpByte3 ATN TmpByte2

    'DEBUG "X: ", SDEC TmpByte3, " Y: ", SDEC TmpByte2, CR
    'DEBUG "Heading to goal: ", DEC headingToGoal, CR

    'Calculate heading error
```

```
    headingToGoal = headingToGoal - I2cData

    'DEBUG "Heading error: ", DEC headingToGoal, " Heading: ", DEC
I2cData, CR

    'Calculate distance to goal
    IF TmpByte3 > 127 THEN
      TmpByte3 = 256 - TmpByte3
    ENDIF
    IF TmpByte2 > 127 THEN
      TmpByte2 = 256 - TmpByte2
    ENDIF

    'DEBUG "Lat Err: ", SDEC TmpByte3, " Lon Err: ", SDEC TmpByte2, CR

    TmpWord1 = SQR ((TmpByte3*TmpByte3)+(TmpByte2*TmpByte2))

    'DEBUG "Distance error: ", DEC TmpWord1, CR

    IF headingToGoal > 128 THEN
      headingToGoal = 255 - headingToGoal
      headingToGoal = (headingToGoal*headingKp) MAX 128
      TmpWord1 = TmpWord1 MAX (128 - headingToGoal)
      'DEBUG "Distance error capped: ", DEC TmpWord1, CR
      'DEBUG "Heading cmd: ", DEC headingToGoal, " dist cmd: ", DEC
TmpWord1, CR
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth - headingToGoal -
TmpWord1
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth - headingToGoal +
TmpWord1
    ELSE
      headingToGoal = (headingToGoal*headingKp) MAX 128
      TmpWord1 = TmpWord1 MAX (128 - headingToGoal)
      'DEBUG "Distance error capped: ", DEC TmpWord1, CR
      'DEBUG "Heading cmd: ", DEC headingToGoal, " dist cmd: ", DEC
TmpWord1, CR
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth + headingToGoal -
TmpWord1
      PULSOUT ServoPalInp, ZeroSpeedPulseWidth + headingToGoal +
TmpWord1
    ENDIF
  ENDIF

LOOP

END

'---------------------------------
' Stamp-Stamp Serial Functions
'---------------------------------
No_Data_Ready:
  'DEBUG "."
  GOTO Continue_Loop

'-----------------------------------------------------------------------
----------------------
' I2C subroutines follow
```

```
'-------------------------------------------------------------------------
----------------------

I2cByteWrite:                           ' writes I2cData.lowbyte to
I2cReg at I2cAddr
GOSUB I2cStart
TmpByte1 = I2cAddr
GOSUB I2cOutByte                        ' send device address
TmpByte1 = I2cReg
GOSUB I2cOutByte                        ' send register number
TmpByte1 = I2cData.LOWBYTE
GOSUB I2cOutByte                        ' send the data
GOSUB I2cStop
RETURN

I2cWordWrite:                           ' writes I2cData to I2cReg at
I2cAddr
GOSUB I2cStart
TmpByte1 = I2cAddr
GOSUB I2cOutByte                        ' send device address
TmpByte1 = I2cReg
GOSUB I2cOutByte                        ' send register number
TmpByte1 = I2cData.HIGHBYTE
GOSUB I2cOutByte                        ' send the data - high byte
TmpByte1 = I2cData.LOWBYTE
GOSUB I2cOutByte                        ' send the data - low byte
GOSUB I2cStop
RETURN

I2CByteRead:
GOSUB I2cStart
TmpByte1 = I2cAddr
GOSUB I2cOutByte                        ' send device address
TmpByte1 = I2cReg
GOSUB I2cOutByte                        ' send register number
GOSUB I2cStart                          ' repeated start
TmpByte1 = I2cAddr | 1
GOSUB I2cOutByte                        ' send device address (with
read set)
sign = 0                              ' send Nak
GOSUB I2cInByte
I2cData.LOWBYTE = TmpByte1                ' read the data
I2cData.HIGHBYTE = 0
GOSUB I2cStop
RETURN

I2CWordRead:
GOSUB I2cStart
TmpByte1 = I2cAddr
GOSUB I2cOutByte                        ' send device address
TmpByte1 = I2cReg
GOSUB I2cOutByte                        ' send register number
GOSUB I2cStart                          ' repeated start
TmpByte1 = I2cAddr | 1
sign = 1                              ' send Ack
GOSUB I2cOutByte                        ' send device address (with
read set)
```

```
GOSUB I2cInByte
I2cData.HIGHBYTE = TmpByte1                ' read the data
sign = 0                                 ' send Nak
GOSUB I2cInByte
I2cData.LOWBYTE = TmpByte1
GOSUB I2cStop
RETURN

I2cOutByte:
SHIFTOUT SDA, SCL, MSBFIRST, [TmpByte1]
INPUT SDA
HIGH SCL                                  ' clock in the ack' bit
LOW SCL
RETURN

I2cInByte:
SHIFTIN SDA, SCL, MSBPRE, [TmpByte1]
SDAout = 0
SDAdir = sign
HIGH SCL                                  ' clock out the ack' bit
LOW SCL
INPUT SDA
RETURN

I2cStart:                                  ' I2C start bit sequence
HIGH SDA
HIGH SCL
LOW SDA
LOW SCL
RETURN

I2cStop:                                  ' I2C stop bit sequence
LOW SDA
HIGH SCL
HIGH SDA
RETURN
```

GPS Serial Forwarding

```
' {$STAMP BS2px}
' {$PBASIC 2.5}
'-----------------------------------------------------------
' GPS constants and variables
'-----------------------------------------------------------
Sio             PIN     15     ' connects to GPS Module SIO pin

T4800           CON     823

GpsBaud         CON     Open | T4800    ' Open mode to allow daisy
chaining

MoveTo          CON     2     ' DEBUG positioning command
ClrRt           CON     11    ' clear line right of cursor
FieldLen        CON     22    ' length of debug text
```

```
DegSym          CON     176   ' degrees symbol for report
MinSym          CON     39    ' minutes symbol
SecSym          CON     34    ' seconds symbol

' GPS Module Commands
GetInfo         CON     $00
GetValid        CON     $01
GetSats         CON     $02
GetTime         CON     $03
GetDate         CON     $04
GetLat          CON     $05
GetLong         CON     $06
GetAlt          CON     $07
GetSpeed        CON     $08
GetHead         CON     $09

degrees   VAR      Byte     ' latitude/longitude degrees
minutes   VAR      Byte     ' latitude/longitude minutes
minutesD   VAR      Word     ' latitude/longitude decimal minutes
dir       VAR      Byte     ' direction (latitude: 0 = N, 1 = S,
longitude: 0 = E, 1 = W)

latDegrees VAR  Byte
latMinutes VAR  Byte
latMinutesD VAR Word
latDir      VAR Byte

'-----------------------------------------
' Serial Constants and Variables
'-----------------------------------------

Bs2SerialData     PIN 6
Bs2SerialFlow     PIN 5

#SELECT $STAMP
  #CASE BS2, BS2E, BS2PE
    T1200       CON     813
    T2400       CON     396
    T9600       CON     84
    T19K2       CON     32
    T38K4       CON     6
  #CASE BS2SX, BS2P
    T1200       CON     2063
    T2400       CON     1021
    T9600       CON     240
    T19K2       CON     110
    T38K4       CON     45
  #CASE BS2PX
    T1200       CON     3313
    T2400       CON     1646
    T9600       CON     396
    T19K2       CON     188
    T38K4       CON     84
#ENDSELECT


Inverted          CON $4000
```

```
Open              CON $8000
Baud              CON T38K4 + Inverted

Main:
DEBUG "Start", CR

Begin_Loop:
DO
  'Wait for GPS to lock
  DO
    GOSUB Get_Valid
    IF dir = 0 THEN
      GOSUB Get_Sats
      DEBUG DEC dir, "?"
      dir = 0
      PAUSE 1000
    ENDIF
  LOOP WHILE dir = 0
  GOSUB Get_Lat
  latDegrees = degrees
  latMinutes = minutes
  latMinutesD = minutesD
  latDir = dir
  GOSUB Get_Long

  DEBUG "Send", CR
  DEBUG "Lat: ", DEC latDegrees, " ", DEC latMinutes, ".", DEC
latMinutesD, " ", DEC latDir, CR
  DEBUG "Lon: ", DEC degrees, " ", DEC minutes, ".", DEC minutesD, " ",
DEC dir, CR, CR
  SEROUT Bs2SerialData\Bs2SerialFlow, Baud, [latDegrees, latMinutes,
latMinutesD.HIGHBYTE, latMinutesD.LOWBYTE, latDir, degrees, minutes,
minutesD.HIGHBYTE, minutesD.LOWBYTE, dir]
  PAUSE 1000
LOOP

END

'-----------------------------------------------------------------------
-------
' GPS Module Functions
'-----------------------------------------------------------------------
-------

Get_Valid:
  SEROUT Sio, GpsBaud, ["!GPS", GetValid]
  SERIN  Sio, GpsBaud, 3000, No_Response, [dir]

  RETURN

' -----------------------------------------------------

No_Response:
  DEBUG "No Response from GPS", CR
  GOTO Begin_Loop
```

```
' -----------------------------------------------------

Get_Lat:
  SEROUT Sio, GpsBaud, ["!GPS", GetLat]
  SERIN  Sio, GpsBaud, 3000, No_Response, [degrees, minutes,
minutesD.HIGHBYTE, minutesD.LOWBYTE, dir]

  RETURN

' -----------------------------------------------------

Get_Long:
  SEROUT Sio, GpsBaud, ["!GPS", GetLong]
  SERIN  Sio, GpsBaud, 3000, No_Response, [degrees, minutes,
minutesD.HIGHBYTE, minutesD.LOWBYTE, dir]

  RETURN

' -----------------------------------------------------

Get_Sats:
  SEROUT Sio, GpsBaud, ["!GPS", GetSats]
  SERIN  Sio, GpsBaud, 3000, No_Response, [dir]

  RETURN

' -----------------------------------------------------
```