The Pennsylvania State University
The Graduate School
College of Engineering

ROBOT FOR OBSERVATIONAL GAIT ASSESSMENT AND REHABILITATION

A Thesis in
Mechanical Engineering
by
Andrew J. Homich

© 2014 Andrew J. Homich

Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

August 2014
The thesis of Andrew J. Homich was reviewed and approved* by the following:

Jason Moore  
Assistant Professor of Mechanical Engineering  
Thesis Adviser

Reuben Kraft  
Assistant Professor of Mechanical Engineering

Karen Thole  
Head of the Department of Mechanical Engineering

*Signatures are on file in the Graduate School.
Abstract

Andrew Homich, Mechanical Engineering, The Pennsylvania State University
Robot for Observational Gait Assessment and Rehabilitation

This thesis explores the design of a robotic device for observational gait assessment and rehabilitation, a method to estimate a patient’s orientation within the rehabilitation device, as well as an optimal state space controller to actuate the rehabilitation device. Current rehabilitation methods require the patient to propel the assistive device or offer limited walking distance. Additionally, current devices do not measure the patient’s reliance on the assistive device, possibly prolonging the rehabilitation period or even preventing satisfactory function to be regained. A novel Robot for Observational Gait Assessment and Rehabilitation (ROGAR) was designed to address the shortfalls of current assistive devices. A complementary filter was developed to estimate the patient’s orientation within the device using a magnetometer and gyroscope. Experiments of the complementary filter on a test platform show that the filter provides estimates within 5 degrees of the true value over a range of angular velocities. An optimal state space controller was implemented using a linear quadratic regulator. The controller preforms well both in simulation and on the actual device. The simulations is able to predict the linear behavior with an average error of 2.7cm and the angular behavior with an average error of 8 degrees.
Contents

List of Figures vii

List of Tables x

List of Symbols xi

Dedication xiii

Acknowledgments xiv

Chapter 1 Introduction 1
  1.1 Motivation 1
  1.2 Methods of Gait Assessment and Physical Therapy 3
    1.2.1 Video Analysis 3
    1.2.2 Robotic Devices 3
    1.2.3 Wearable Sensors 4
  1.3 Research Goals 5
  1.4 Thesis Outline 7

Chapter 2 Robotic Platform 8
  2.1 Frame Design 8
  2.2 Hardware 9
    2.2.1 National Instruments cRIO 9076 9
    2.2.2 Golden Motor Company HUB24 10
2.2.3 Advanced Motion Control 25a8 11
2.2.4 Sensors 11
2.3 Electrical Design 14

Chapter 3 Estimating Patient Position and Orientation 17
3.1 Patient Sensors 17
3.2 Complementary Filter Design 19
3.3 Experimental Setup and Procedure 21
3.4 Experimental Results and Conclusions 22

Chapter 4 Controller Design 27
4.1 System Model 27
4.2 State Space Controller Theory 31
4.3 Linear Controller 33
4.4 Angular Controller 35
4.5 Controller Implementation 37

Chapter 5 Controller Testing 39
5.1 Simulated Response to Step Input 39
5.2 Device Response to Step Input 44
5.3 Device Versus Simulated Response to Step Input 47
5.4 Device Response to User Input 48
5.5 Conclusions 54
5.6 Future Work 56

Bibliography 58

Appendix A. LabView and Device References 61
A.1 National Instruments References 61
A.2 Charging ROGAR 63
A.3 LabView Program Layout 64
A.4 Running the State Space Controller Program 66
Appendix B.  Device Photos and Mechanical Drawings  68
  B.1  Frame Pictures  69
  B.2  Reference Graphics  74
  B.3  Shop Drawings  75

Appendix C.  Electrical Schematics and Diagrams  77

Appendix D.  MATLAB Code  84

Appendix E.  LabView Code  99
  E.1  FPGA  99
  E.2  Sub VIs  102
  E.3  Component Blocks  108
  E.4  Controller  113

Appendix F.  Arduino Code  114

Appendix G.  Data Sheets  121
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>System design process</td>
<td>6</td>
</tr>
<tr>
<td>1.2</td>
<td>System associated errors</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>ROGAR frame</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Wheel configuration of ROGAR</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>NI cRIO-9076</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>HUB24</td>
<td>10</td>
</tr>
<tr>
<td>2.5</td>
<td>AMC 25A8</td>
<td>11</td>
</tr>
<tr>
<td>2.6</td>
<td>Sensors used on ROGAR</td>
<td>12</td>
</tr>
<tr>
<td>2.7</td>
<td>Arduino with magnetometer and gyroscope</td>
<td>13</td>
</tr>
<tr>
<td>2.8</td>
<td>IR calibration curve</td>
<td>13</td>
</tr>
<tr>
<td>2.9</td>
<td>Peripheral hub board</td>
<td>15</td>
</tr>
<tr>
<td>2.10</td>
<td>Peripheral hub board I/O layout</td>
<td>15</td>
</tr>
<tr>
<td>3.1</td>
<td>Position and orientation error ($d_e$ and $\theta_e$) with system equilibrium(E)</td>
<td>18</td>
</tr>
<tr>
<td>3.2</td>
<td>Patient view with mounted IR Sensor and coordinate system relative to ROGAR</td>
<td>18</td>
</tr>
<tr>
<td>3.3</td>
<td>Complementary filter design</td>
<td>20</td>
</tr>
<tr>
<td>3.4</td>
<td>Complementary filter test platform</td>
<td>21</td>
</tr>
<tr>
<td>3.5</td>
<td>Typical trial for testing complementary filter</td>
<td>23</td>
</tr>
<tr>
<td>3.6</td>
<td>Average error and standard deviation over full range of filter gains</td>
<td>24</td>
</tr>
<tr>
<td>3.7</td>
<td>Average error and standard deviation over subset of filter gains</td>
<td>25</td>
</tr>
<tr>
<td>3.8</td>
<td>Average error over varying angular velocities</td>
<td>26</td>
</tr>
<tr>
<td>4.1</td>
<td>Single-output system</td>
<td>28</td>
</tr>
</tbody>
</table>
4.2 Wheel on ROGAR  
4.3 Block diagram for state space controller  
4.4 Optimal linear controller gains for varying values of $\rho$  
4.5 Optimal angular controller gains for varying values of $\rho$  
5.1 Linear simulation: States vs. Time  
5.2 Linear simulation: Outputs vs. Time  
5.3 Angular simulation: States vs. Time  
5.4 Angular simulation: Outputs vs. Time  
5.5 Linear and angular simulation: States vs. Time  
5.6 Linear and angular simulation: Outputs vs. Time  
5.7 Device response to 0.5m step input  
5.8 Device response to 1m step input  
5.9 Device response to 22.5deg step input  
5.10 Device response to 45deg step input  
5.11 Device and simulation response to 0.5m step input  
5.12 Device and simulation response to 22.5deg step input  
5.13 Overhead camera view  
5.14 MATLAB view after image processing  
5.15 Linear controller: Simulated response vs. device response with patient input  
5.16 Linear controller: Error between simulated response and device response  
5.17 Angular controller: Simulated response vs. device response with patient input  
5.18 Angular controller: Error between simulated response and device response  
A.1 State space controller program  
B.1 Frame front isometric  
B.2 Frame rear isometric  
B.3 Frame rear  
B.4 Frame side  
B.5 Left side hardware  
B.6 Right side hardware  

viii
B.7 Gear ratio for hub motors 74
B.8 Hub motor assembly 75
C.1 Hub board I/O layout 77
C.2 cRIO Channel Layout 78
C.3 Switch, battery, and charging port layout 79
E.1 FPGA code 101
E.2 IMU Read code 102
E.3 Bias Readings code 103
E.4 Mag Read code 105
E.5 Complementary Filter code 107
E.6 Encoder Read code 109
E.7 IR Read code 110
E.8 Wheel Write code 111
E.9 Complementary Filter Package code 112
E.10 State Space Controller code 113
List of Tables

3.1 Experimental parameters used in complementary filter tests 22
4.1 $Q$ and $R$ matrix properties for linear system 34
4.2 $Q$ and $R$ matrix properties for angular system 36
List of Symbols

\begin{align*}
d_m & \quad \text{Patient’s measured distance from IR sensor} \\
G_m & \quad \text{Gyroscope’s measured angular velocity} \\
M_m & \quad \text{Magnetometer’s measured angle} \\
V & \quad \text{Voltage} \\
\phi & \quad \text{Angle estimated by the complementary filter} \\
\phi_{-1} & \quad \text{Angle estimated from the complementary filter’s previous iteration} \\
M & \quad \text{ROGAR mass} \\
J & \quad \text{ROGAR mass moment of inertia} \\
F & \quad \text{Force in a direction or Force applied by} \\
M_o & \quad \text{Moment about the center of ROGAR} \\
\tau & \quad \text{Wheel torque} \\
T & \quad \text{Torque applied to ROGAR} \\
a & \quad \text{ROGAR acceleration} \\
\alpha & \quad \text{ROGAR angular acceleration} \\
r & \quad \text{Wheel radius} \\
L & \quad \text{ROGAR center-to-wheel length} \\
I & \quad \text{Current} \\
K_T & \quad \text{Motor torque constant} \\
\delta t & \quad \text{Loop iteration time} \\
\tau_c & \quad \text{Time constant} \\
x & \quad \text{State in the state space representation}
\end{align*}
\( u \) Input in the state space representation
\( A \) State matrix in the state space representation
\( B \) Input matrix in the state space representation
\( C^* \) Controllability test matrix
\( Q \) State constraint matrix
\( R \) Input constraint matrix
\( P \) Solution to the Algebraic Riccati Equation
\( q_i \) Typical constraint on the \( i^{th} \) state
\( r_i \) Typical constraint on the \( i^{th} \) input
\( \rho \) Constant to determine tracking versus control effort; system stiffness
\( t_{si} \) State controller settling time
\( K_f \) Complementary filter gain (determines accuracy of orientation estimate)
\( K \) State feedback gain matrix (determines aggressiveness of ROGAR’s controller)
\( K_{\phi} \) Angle gain (element within K)
\( K_{\dot{\phi}} \) Angular velocity gain (element within K)
\( K_x \) Position gain (element within K)
\( K_{\dot{x}} \) Velocity gain (element within K)
\( \dot{} \) First derivative of a quantity
\( \ddot{} \) Second derivative of a quantity
\( (\cdot)^{-1} \) Inverse of a matrix
\( (\cdot)^T \) Transpose of a matrix
Dedication

This thesis is dedicated to my parents, John and Christine, and my sister, Laura.
Acknowledgments

I would like to thank my research adviser Dr. Jason Moore for his guidance and support in completing this research. I would also like to thank Dr. Stephen Piazza and Dr. Everett Hills for their participation in this research. I’d like to thank the many friends I have had the pleasure to know while in State College. I will always remember the good times we spent together. Finally, I’d like to thank my parents and sister who have love, supported, and encouraged me throughout life and my academic career.
Chapter 1

Introduction

1.1 Motivation

At this moment, millions of American adults are going through physical therapy to improve their ability to walk. Some of these people are geriatric patients whose old age has weakened their muscles and coordination; some have recently undergone lower-limb amputations and are taking their first steps with prosthetic feet or knees; others have had strokes or traumatic brain injuries that have impaired the control of their locomotion. Currently, mobility assessment is primarily done through observational gait assessment by a highly trained therapist visually watching a patient’s gait. This is costly in terms of time required by the therapist or physician; and difficult due to change in patient gait caused by pain and other medically related factors, the subjective nature of the observations, and limitations to the accuracy of the information visually observed by the therapist or physician. A gait assessment robot that moves along with the patient and is able to quickly and accurately provide quantitative measures of gait proficiency could revolutionize how mobility is assessed in the therapy clinic.

One in every three adults over 65 experience a fall each year [1]. In 2010 there were 2.3 million nonfatal fall injuries treated in emergency departments at an estimated cost of $30 billion [2, 3]. Currently the best predictor of falling is a history of previous falls [4]. Small changes in gait serve as an early indicator for increased risk of falling and can be an early
indicator for the onset of dementia [5]. An accurate quantitative gait assessment tool would have its greatest impact on society by identifying these small changes and would allow for appropriate proactive intervention to prevent falls from occurring.

Quantitative gait assessments would be valuable for a prosthettist attempting to fit a patient with new lower-limb prosthesis and make the necessary adjustments to achieve a natural, symmetric, and energy-efficient gait pattern. Gait analysis of amputee patients is particularly helpful because patients often develop unusual gait patterns as they learn to walk with their new prosthesis [6]. Prosthetists often lack access to gait laboratories and traditionally rely on observational analysis to assess gait patterns [7]. A gait assessment robot could help the prosthetist perform a more quantitative analysis of the patient’s gait and determine how small adjustments in the prosthetic device positively and negatively impact the patient’s gait. Such an assessment would allow optimization of the fit of the prosthesis in a shorter amount of time.

Physical therapists and physiatrists face the challenge of making an accurate assessment of the patient’s gait to prescribe the most effective therapy possible for patients who suffer from mobility impairment. This could be a neurological impairment caused by a stroke or traumatic brain injury, or it could be a muscular impairment that prevents normal gait from being achieved. Being able to freely move around one’s environment and maintain specific body positions is a fundamental requirement for an independent lifestyle [8]. A patient who fails to regain the ability to walk well faces obstacles that extend beyond lost mobility; such patients see their quality of life reduced and have elevated risk of heart disease and depression.

Clinicians seeking quantitative gait analysis of their patients have options at the extremes in terms of technology and cost. At one extreme, measurements of spatiotemporal gait characteristics (speed, cadence, stride length) are made easily in the clinic using a stopwatch and a measured distance in a hallway. At the other end of the spectrum is video-based clinical gait analysis, which provides a wealth of kinematic and kinetic data (three-dimensional joint angles, moments, and powers), but which is impractical for gauging improvement on a regular basis in the clinic. Such analyses typically require a large
dedicated space for several video cameras and force-measuring platforms that are mounted in the floor. Highly-trained personnel are needed to operate the expensive video-based motion analysis systems and the process of applying markers to the body and calibrating the system requires 3-4 hours to complete. For patients that must rely on assistive walking devices during therapy, neither of these methods are capable of determining the patients’ reliance on that assistive device, limiting the opportunity for accurate assessment of the patient’s progress. A limited cost, easy to use system capable of collecting high level quantitative gait analysis information while recording patient-device reliance would be very helpful in assessing patient performance.

1.2 Methods of Gait Assessment and Physical Therapy

1.2.1 Video Analysis

The current gold standard for analysis of human walking is video-based motion analysis in which multiple cameras track reflective markers placed on the patient’s skin [9]. An example of such a system is the eight-camera digital system (Eagle; Motion Analysis Corp.) utilized by Dr. Piazza in Penn State’s Biomechanics Laboratory. This system has eight 1.3 megapixel cameras and advanced real-time software that permit tracking of individual markers with submillimeter accuracy. Use of these video-based motion analysis systems has been limited to dedicated motion analysis laboratories as they are very expensive, require experts for data interpretation, and some systems require patients to perform difficult calibration tasks which would not be suitable for geriatric or cognitively impaired patients.

1.2.2 Robotic Devices

Interest in the application of robotic technology for improving and assessing rehabilitation outcomes has rapidly increased in the past fifteen years, as evidenced by the exponential growth in the number of published reports on rehabilitation robotics during this period [10]. This interest is driven by several factors, including an aging U.S. population, the increasing numbers of patients who survive following stroke and brain injury [11], greater numbers
of patients who have limbs amputated due to complications following diabetes [12], and health care costs that are projected to outpace the growth of U.S. gross domestic product. Robotic technology offers the potential for improving outcomes and lowering costs, but early experience with robots in the therapy clinic has shown that significant hurdles must be overcome and the problems to which robots are applied to solve must be carefully chosen if progress in this area is to be realized. Successful implementation of robots in the clinic requires demonstration of cost savings and improved outcomes compared to conventional therapy, but there are tradeoffs: A highly effective robot may be desirable even if the cost is greater than that for conventional therapy [10].

Gait-training robots; such as the Lokomat [13], the WalkTrainer [14], LOPES [15], and AutoAmbulator [16] typically have featured mechatronic feedback systems that guide leg movements to correct gait deviations. Such systems have been shown to be effective to varying degrees in improving the gait of stroke patients [17, 18]. The Lokomat system also has some limited assessment capabilities including the ability to assess: spasticity disorder (velocity dependent stiffness to muscle stretching) [19, 20]; leg strength [19, 21]; and range of motion [22]. None of these devices, however, are capable of measuring a patient’s natural gait pattern because all of them require contact with the legs. This contact disrupts the natural gait pattern and this makes it impossible to determine whether the patient’s gait signals an inordinately high risk of falling, a poor prosthetic fit, etc.

### 1.2.3 Wearable Sensors

The idea of placing electronic sensors on the body to analyze human motion has been around since the 1970’s [23]. Investigators have explored placing foot switches, accelerometers, and gyroscopes on patients to measure various parts of the human body [24, 25, 26, 27]. Although useful for determining accelerations of specific body segments, these sensors have not been useful for estimating spatiotemporal gait parameters such as average gait speed or step length. Aminian et al. [24] and Zijlstra [28] have attempted to utilize complex models to estimate velocity and stride length from accelerometer, gyroscopic, and foot switches, but these approaches suffer from inherent errors because the relative time and position of
points on the body are not being directly measured. In addition, the additional mass of the sensors have the potential to alter the patient’s natural gait.

1.3 Research Goals

This thesis presents the design of a robot for observational gait assessment and rehabilitation, or ROGAR. The device will be used as a platform to develop a mobile lab to measure gait characteristics and assist with physical therapy. In order to utilize the robotic platform, the robot must be able to accurately track the patient standing and walking inside of it, as well as adjust itself to changes in patient location or orientation. Therefore, this research includes the following goals:

1. Design and build a robotic platform that will accommodate the necessary hardware to control it, while allowing for future additions to the device.

2. Develop and implement a method to accurately track the patient’s linear position and angular orientation within the device.

3. Design and optimize a controller to move the robot with changes in the patient’s position and orientation.

To achieve these goals, the robot was first modeled in SolidWorks and the electrical schematics were developed. Necessary parts were ordered or fabricated, and the electrical system was wired. Second, a complementary filter was developed and tested to determine the optimal gains to best estimate orientation. Next, a state space model and controller were developed and simulated using MATLAB. Finally, the data gathered from the simulation was used to implement the controller on the actual robotic device. A flowchart of the design process is seen in Figure 1.1.
Throughout the research, a key goal was to minimize the total position and orientation error between the patient and device. In order to achieve this goal, errors related to the state space controller had to be minimized as well as errors associated with the sensors such as sensor drift with the gyroscope and the slow response of the magnetometer. Figure 1.2 shows how data is passed and errors accumulate in the system.
1.4 Thesis Outline

This thesis consists of five chapters. Chapter 1 covers the motivation for the research, current methods for gait assessment and rehabilitation, and the research goals. Chapter 2 provides an overview of the robotic platform that was developed and used for this research. It includes a brief overview of the frame design, major hardware components, and electrical design. Chapter 3 describes the complementary filter design, the experimental setup and procedure to test the filter, and the results from the tests. Chapter 4 details the state space model and controller design, and the use of a linear quadratic regulator to optimize the controller. Chapter 5 covers a simulation of the device’s step response, the device’s actual step response, a comparison between the simulated and actual step response, and concludes with testing the device response with a patient input and a comparison to a simulated response with the same input.
Chapter 2

Robotic Platform

2.1 Frame Design

Figure 2.1 shows the completed ROGAR (Robot for Observational Gait Assessment and Rehabilitation) frame which was designed in SolidWorks and fabricated from square steel tubing, stainless steel, and aluminum. The stresses on the frame were modeled using the SolidWorks build-in finite element analysis (FEA) package. It was determined that the weakest component of ROGAR is the bent stainless steel tubing that supports the parallel bars. Each tube is capable of supporting 350lbs before yielding. That being said, the force sensors used under each corner of the parallel bars are rated for 100lbs each.

The device uses four wheels; two hub motors and two caster wheels. The hub motors are mounted towards the rear of the device and the casters at the front, as seen in Figure 2.2. This wheel configuration allows the device to pivot about the patient’s center of mass.

The sides of the frame are used to mount the hub motor wheels. Brackets were designed
with a slot to make the wheel height adjustable; two thrust nuts on either side of the brackets are used to hold the wheels at the desired height. The sides also feature aluminum boxes to hold the three batteries required to run the device (see Section 2.3 for the electrical design). Additionally, polycarbonate battery and wheel covers provide ample space to mount the additional hardware required to run ROGAR.

The parallel bar height can be varied from a minimum of 25 inches to a maximum of 44 inches; therefore, allowing accommodation to patients ranging in height from 5’ to 6’7”. The parallel bars feature force sensors under each end of the bars so that the patient’s reliance on the device can be monitored and tracked over time.

Pictures and drawings with overall dimensions of the frame are shown in Appendix B.

2.2 Hardware

2.2.1 National Instruments cRIO 9076

ROGAR uses a National Instruments cRIO-9076 controller. The cRIO is small (7”x3.5”x4”), lightweight (1.5 lbs), and runs independently without the need to communicate to a host computer. The cRIO is programmed using LabView FPGA and Labview Real-Time; both are graphical languages that utilize blocks and wires to program the controller. There is a learning curve to LabView, even if one is well versed with in-line coding; however, it is well supported both online and by phone. Useful LabView reference numbers and websites can be found in Appendix A.
The cRIO model that was chosen for the parallel bars contains four slots for different input/output (IO) modules. These slots contain two digital modules (NI 9401 and NI 9402), one analog output (NI 9263), and one analog input (NI 9205). The NI 9401 is an 8-channel module that interfaces with other devices through terminal blocks; this module supports the SPI drivers from National Instruments. The NI 9402 is a 4-channel module that interfaces with other devices through BNC cables; this module supports the I2C drivers from National Instruments. Ideally, the accelerometer and magnetometer used on ROGAR would utilize the I2C driver to directly interface with the cRIO. The channel layout for ROGAR can be seen in Appendix C.

2.2.2 Golden Motor Company HUB24

ROGAR uses two 8 inch diameter, brushed 24VDC hub motors made by the Golden Motor Company for wheels. These wheels have a pancake motor inside that is then geared at a ratio of 0.0481:1 to the outer wheel, see Figure B.7 in Appendix B for the gear diagram. Both wheels were modified by boring out the mounting peg opposite the two control wires to accommodate an encoder, see Figure B.8. The encoder is interfaced directly with the pancake motor via a #3 threaded rod that screws into a hole that was drilled and tapped into the shaft of the pancake motor. The motor curve and specifications for the hub assembly are provided in Appendix G.
2.2.3 Advanced Motion Control 25a8

ROGAR uses two Advanced Motion Control (Camarillo, CA) 25A8 PWM servos to drive the wheels. The 25A8 servos have a supply voltage of 20-80V and a peak current of 25A. The servos take a command input of \(\pm 10\text{VDC}\) and outputs either a current signal or PWM voltage signal. The current mode of operation was chosen so that the wheel torque could be directly controlled. The switches and potentiometers on the servos were set according to the manufacturer's instructions for current operation mode. Additionally, the emergency stop for ROGAR was tied directly to the servos, so that power could be cut from the wheels without the need to stop the controller. To do this, one side of the E-stop was wired to \textit{Inhibit In} (pin 11) and the other side was wired to the servo's ground (pin 2); refer to the wiring diagram in Appendix C. More specifications for the servos are provided in Appendix G.

2.2.4 Sensors

The gyroscope is part of an inertial measurement unit made by InvenSense, the MPU-6050, and is mounted on the GY-521 breakout board, Figure 2.6a; its data is output via I2C. Currently, National Instruments does not have a company-supported driver to implement I2C on the cRIO, therefore the gyroscope data is first sent to an Arduino Uno microprocessor. The Arduino reads the I2C data from the gyroscope and converts it to a PWM signal based on the measurable range of the gyroscope. This PWM value is much easier to integrate with the cRIO processor.
The magnetometer is a separate unit produced by Freescale Semiconductor, the MAG3110 (Figure 2.6b). It is mounted on a breakout board of the same name by Sparkfun. Like the gyroscope, the output of this device is also in I2C and therefore uses the same Arduino and the gyroscope to read the incoming data. The $x$ and $y$ components of the magnetic field are used in the $\text{atan2}(x, y)$ function to calculate a heading. The heading is then output as a PWM signal.

There are two Arduinos that read the gyroscope and magnetometer, Figure 2.7. One is affixed to a belt which the patient will have to wear; the other is mounted on ROGAR. The outputs of the Arduino are tied into the processor unit (NI 9402) using BNC cables.
The infrared (IR) sensor, Figure 2.6c, made by Sharp (GP2Y0A02YK0F) and is rated for use at 20 to 150 centimeters; however, it was found that it loses accuracy past 120 centimeters. The sensor runs on 5VDC and output a 0 to 5V signal to the cRIO processor. The IR was calibrated by plotting the voltage output every 10cm as seen in Figure 2.8 and then a calibration curve, Equation 2.1, was calculated to relate output voltage to distance. There is an average error of 5.9cm or 9.5% between the calculated distance and the true distance.

\[ d_m = 1.6059V^{-1.517} \] (2.1)

The encoders, Figure 2.6d, are from US Digital and are model E4P (E4P-200-250-N-S-D-D-3). They have 200 counts per encoder revolution, which when paired with the gear ratio of the hub motors works out to 4153 counts per wheel revolution, or an accuracy
of 0.0867 degrees. The encoder disks are mounted by pressing the 1/4” bore over nylon spacer and then screwing this spacer onto a #3 threaded rod, refer to Appendix B. The encoder is powered with 5VDC and has two digital, single ended, outputs (channel A and B) so that both position and direction can be calculated. For wiring the wheel encoders, see Appendix C. The data sheet for the encoders are provided in Appendix G.

2.3 Electrical Design

The electrical design for ROGAR is outlined in Appendix C. The main power for the wheels comes from two 12V 15Ah, sealed lead acid (SLA) batteries wired in series to get the necessary 24V to run the hub motors. Also in series with the batteries are a 25A fuse and a keyed switch to toggle the main power. The battery circuit has a DPDT switch wire to each battery along with a pair of banana plug ports. The switch can be toggled to Charge, which directly ties the positive and negative terminals of the battery to their respective charging ports. This eliminates the need to remove the covers on a regular basis to charge the batteries.

Using the Hitec charger (Hitec X1AC Plus) purchased for this project, the batteries should be charged on the Pb Charge setting. The current should be set to no more than $1/10$ the capacity, or C rating. The batteries for the wheels are rated at 15Ah, so limit the charging current to no more than 1.5A. The voltage should be set to the battery’s rating and number of cells. For the wheel batteries, that is 12V and 6 cells. The settings can be switched by pressing start once and then toggling through the options. To start the charging cycle, hold the start button for approximately 3 seconds.

The power from the battery circuit is run through 12 gauge, multi-strand wire. The wires run out the side of the battery boxes and through the self-adhesive wire ducts mounted on the frame of ROGAR. The wires have been left long enough that the covers can be removed and set on the ground with out straining the wire. The 24V are run to the ground and high voltage terminals on the PWM servos mounted above the wheels.

The servos have an emergency stop (E-stop) button tied directly to them using the Inhibit In and Signal Ground terminals. The E-stop cuts power directly from the wheels,
eliminating any interaction with the controller. To start the device again, the E-stop must be rotated to release the switch.

Most of the sensors on ROGAR are run to a peripheral hub board (Figure 2.9 and Figure 2.10) before running to the cRIO. The hub board provides 5V power to the individual sensors and arranges the outputs in a way that easily interface with the cRIO. The hub board also has a green LED to show when the sensors have power.
The power for the hub board is provided from a separate battery (12V 5Ah, SLA). Like the wheels, this battery can be charged by flipping a switch and using the charging ports, however only 0.5A should be used to charge it. This battery also has a main switch to toggle the power on and off. In addition to powering the hub board, this battery also provides power to the cRIO, wireless router, and the arduino that is used to read the orientation of ROGAR. There is a power distribution board under the cRIO that provides 12V connections for each of these devices.
Chapter 3

Estimating Patient Position and Orientation

3.1 Patient Sensors

An arrangement of sensors utilizing an infrared (IR) sensor, magnetometer, and gyroscope provides information on the orientation and position of a patient standing within ROGAR. The IR sensor is mounted on ROGAR frame along with a gyroscope and magnetometer. Another gyroscope and magnetometer are affixed to a belt on the patient’s waist. These sensors work together to determine the error in the patient’s position, $d_e$, and error in orientation, $\theta_e$, relative to the equilibrium vector inside the device. The equilibrium vector is defined as centered in the device, aligned with the wheels’ axis of rotation, and facing forward as illustrated in Figure 3.1.

The IR sensor determines the patient’s location forward or backward of the wheels’ axis of rotation to calculate the value of $d_e$, using Equation 2.1. Figure 3.2 shows the IR sensor mounted at the front of ROGAR. The sensor is mounted using 80/20, a cable carrier, and linear bearings so that the sensor can be adjusted to the patient’s waist height.
Figure 3.1: Position and orientation error ($d_e$ and $\theta_e$) with system equilibrium (E)

Figure 3.2: Patient view with mounted IR Sensor and coordinate system relative to ROGAR

The Arduino module, Figure 2.7, is used to read both the gyroscope and magnetometer
sensors. The readings from the Arduinos on the patient and the device are sent to the cRIO to be converted to a continuous scale. This eliminates the 360 degree jump in readings once either module completes a full rotation. A complementary filter, discussed in Section 3.2, is used to merge the gyroscope and magnetometer readings into one heading for either the patient or ROGAR. The difference in the device’s heading from the patient’s heading is $\theta_e$.

Another solution to getting ROGAR’s heading would be to solely calculate the angle from the encoder data. This method works, but is sensitive to sensor drift from the encoders after extended periods of use. This problem could be mitigated by biasing the encoder data once the drift from the encoder data was noticeable to the patient.

### 3.2 Complementary Filter Design

A complementary filter (Figure 3.3) was chosen to track changing angular position because it combines the best attributes of the gyroscope and the magnetometer. The gyroscope can accurately measure angular velocity, but cannot keep track of angular position over long periods of time due to sensor drift. The magnetometer is better at measuring a steady orientation, but is slow to respond to quick changes in angular position. Therefore, a complementary filter was implemented to take advantage of the strong points for each device while minimizing the inherent undesired attributes. The complementary filter is easy to code and provides fast estimates of orientation. The complementary filter, as shown in Figure 3.3, sends the magnetometer signal through a low-pass filter which is then summed with the time-integrated gyroscope signal which is sent through a high-pass filter.
The complementary filter is easily implemented using Equation 3.1, where $\phi$ is the complementary filter angle estimate, $k_f$ is the complementary filter gain, $\phi_{-1}$ is the complementary filter’s previous angle estimate, $G_m$ is the gyroscope angular velocity measurement, $\delta t$ is the loop iteration time, and $M_m$ is the magnetometer measurement.

$$\phi = k_f(\phi_{-1} + G_m \delta t) + (1 - k_f)M_m$$ \hspace{1cm} (3.1)

The right-hand side of Equation 3.1 acts as a low-pass filter for the magnetometer values by forcing changes to build up slowly over time, thus only allowing long-term changes to pass. The left-hand side works in just the opposite way for the high-pass portion: short-term changes are allowed to pass while long-term changes are blocked. This process is used to block the sensor drift associated with the gyroscope [29]. The time constant, $\tau_c$, for the filter is calculated using Equation 3.2.

$$\tau_c = \frac{k_f \delta t}{1 - k_f}$$ \hspace{1cm} (3.2)
3.3 Experimental Setup and Procedure

A test platform, shown in Figure 3.4, was developed to evaluate the accuracy of the complementary filter. A rotary encoder was fixed to the bottom of the test platform and was clamped to a bench top. An Arduino with sensors attached was mounted to a swivel base that rotated relative to the encoder.

![Figure 3.4: Complementary filter test platform](image)

Individual functions were written to read and convert the gyroscope, magnetometer, and encoder signals to usable values. The values for each sensor were then called by the filter program executing the complementary filter. The filter program graphed the magnetometer value, time-integrated gyroscope value, estimated complementary filter value, and reference value from the encoder. Additionally, the filter program graphed the error of each signal relative to the reference signal. Position data for the encoder, gyroscope, magnetometer, and filter estimates were saved in a text file for analysis by a separate program executed on the host computer.

Before an individual trial was started, the complementary filter gain, $k_f$, was set, the output file path was created, and all the signals were biased to zero. Once an individual
trial was started, the Arduino was rotated 90 degrees. This position was held for 15 seconds and then the test was stopped.

Table 3.1 lists the parameters examined for the three experiments that were conducted. The experiments were as follows: 1) Error vs. Filter Gain over entire range of gains, 2) Error vs. Filter Gain for subset of gains, and 3) Error vs. Angular Velocity. The experiments are summarized in Section 3.4.

Table 3.1: Experimental parameters used in complementary filter tests

<table>
<thead>
<tr>
<th>Exp</th>
<th>Complementary Filter Gain</th>
<th>Angular Velocity (deg/s)</th>
<th>Measured</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp 1</td>
<td>0.05-0.95, 0.1 inc.</td>
<td>90</td>
<td>$\phi_{\text{error}}$</td>
</tr>
<tr>
<td>Exp 2</td>
<td>0.76-1.00, 0.02 inc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exp 3</td>
<td>0.92-0.98, 0.01 inc.</td>
<td>22.5, 45, 60, 90, 180</td>
<td></td>
</tr>
</tbody>
</table>

3.4 Experimental Results and Conclusions

A typical filter response as the test platform was rotated 90 degrees is shown in Figure 3.5. This 90 degree rotation represents a worst-case scenario where a patient would be turning around in a clinical setting. For rehabilitation therapy where the patient is walking in a shallow arc, a realistic angle change would be between 15 and 45 degrees. The true angle, or the reference, is given by the dashed line, the complementary filter’s estimate is given by the solid line, and error from the true value is given by the dot/dash line. For experiments 1 and 2, the platform was rotated at approximately 90 deg/s; this angular velocity was varied for experiment 3.
Figure 3.5: Typical trial for testing complementary filter

The results from experiment 1 are shown in Figure 3.6. The average error and standard deviation of this error was taken over the 0 to 90 degree turn. A test was run for complementary filter gains ranging from 0.05 to 0.95 in 0.1 increments. As shown in Figure 3.6 the filter was more accurate when heavily weighted toward the gyroscope readings: complementary filter gains above 0.75. The large error at lower gains was due to the slow response of the magnetometer relative to quick changes in angle.
The results from experiment 2 are shown in Figure 3.7. The average error and standard deviation were calculated in the same manner as experiment 1. Complementary filter gains above 0.75 were again tested in 0.02 increments. The average error across this range of complementary filter gains was minimized at a complementary filter gain of 0.94. The standard deviation was also smallest around this complementary filter gain. As before, the magnetometer lagged the reference angle at the lower complementary filter gains causing errors in the filtered signal. For higher complementary filter gains, the errors were caused due to a drifting gyroscope reading. Around a complementary filter gain of 0.94, the magnetometer was weighted enough to keep the filtered signal from drifting and the gyroscope was still able to account for the quick changes in angle.
Figure 3.7: Average error and standard deviation over subset of filter gains

Figure 3.8 shows the results of experiment 3 which focused on the effect of angular velocity on the average error. For this experiment the platform was rotated 90 degrees over varying angular velocities of 22.5, 45, 60, 90, 180 deg/s. The average filter error was again calculated during the 0 to 90 degree turn as shown in Figure 3.5. The filter performed best over all the various angular velocities at a complementary filter gain of 0.97.
The complementary filter successfully estimated true position. The optimal complementary filter gain for angular velocities ranging from 22.5 to 180 deg/s was found to be 0.97. This result strongly weights the filter on the gyroscope side, but allows the gyroscope to respond to fast changes in the patient’s orientation while the magnetometer keeps the signal from drifting away from the true angle. While there are errors associated with the filter, the errors are generally no more than 5 degrees and the average is much closer to 1 degree. In the context of ROGAR, the complementary filter is more than adequate as it is unlikely that a patient will notice errors of a few degrees while using the device.
Chapter 4

Controller Design

4.1 System Model

Before starting to design the controller, the system was simplified to a single-output system, by splitting ROGAR in half as seen in Figure 4.1. The system can be reduced to a single-output system because the two outputs (the hub motors) are dependent on each other. For example, if the device is to move forward 1 meter, both wheels must move 1 meter. If one wheel were to move the required distance without the other, ROGAR would make an arc instead of a line.
Figure 4.1: Single-output system

Figure 4.2: Wheel on ROGAR
Two independent system models were then developed: one for the linear movements and another for the rotational. It was assumed that the wheels do not slip, the wheels have no internal viscous friction, and the wheels have no back electromotive force (EMF). Equation 4.1 models the linear system by summing the forces along the Y-axis.

\[ \sum F_y = Ma = F_{\text{wheel}} \quad (4.1) \]

Where \( M \) is the mass of ROGAR, \( a \) is the linear acceleration, and \( F_{\text{wheel}} \) is the force on ROGAR due to the wheel. The wheel torque, \( \tau \), is related to \( F_{\text{wheel}} \) by multiplying by the wheel’s radius, \( r \), as seen in Figure 4.2 and given in the equation below.

\[ \tau = F_{\text{wheel}}r \quad (4.2) \]

A relationship between the linear acceleration, \( a \), and the wheel’s torque is achieved by substituting Equation 4.2 into Equation 4.1 and rearranging terms.

\[ a = \frac{1}{Mr} \tau \quad (4.3) \]

Let the wheel’s torque also be defined by the motor’s torque constant, \( K_\tau \), multiplied by the current, \( I \), supplied to the motor as seen in Equation 4.4.

\[ \tau = K_\tau I \quad (4.4) \]

By substituting Equation 4.4 into Equation 4.3, the final form of the device’s linear acceleration is obtained.

\[ a = \frac{K_\tau I}{Mr} \quad (4.5) \]

Equation 4.6 models the angular system by summing the moments about the center of ROGAR. Let the center be defined as \( O \), the intersection of the lines separating the left
from the right and the rear wheels axis of rotation as shown in Figure 4.1.

\[ \sum M_0 = J\alpha = T_{\text{wheel}} \quad (4.6) \]

Where \( J \) is ROGAR’s moment of inertia, \( \alpha \) is the angular acceleration, and \( T_{\text{wheel}} \) is the torque on ROGAR due to the wheel. \( T_{\text{wheel}} \) is defined as the wheel force, \( F_{\text{wheel}} \), multiplied by the center-to-wheel length, \( L \), as shown in Equation 4.7.

\[ T_{\text{wheel}} = F_{\text{wheel}}L \quad (4.7) \]

Equation 4.8 shows a relationship between ROGAR’s angular acceleration and the wheel’s torque by substituting Equations 4.7 and 4.2 into Equation 4.6 and rearranging terms.

\[ \alpha = \frac{L}{Jr} \tau \quad (4.8) \]

As before, substituting Equation 4.4 into Equation 4.8 yields the final form for the device’s angular acceleration.

\[ \alpha = \frac{LK\tau}{JrI} \quad (4.9) \]
4.2 State Space Controller Theory

For both the linear and angular case, consider the state space controller in Figure 4.3, and given by Equations 4.10 and 4.11.

\[ \dot{x} = Ax + Bu \]  \hspace{1cm} (4.10)

\[ y = Cx + Du \]  \hspace{1cm} (4.11)

\( A \) is the state matrix, \( B \) is the input matrix, \( C \) is the output matrix, \( D \) is the feedforward matrix, \( x \) is the state vector, and \( u \) is the control vector. Assume that the system has no feedforward component, therefore \( D \) is zero. Let the state-feedback control law take the form of Equation 4.12.

\[ u = r - Kx \]  \hspace{1cm} (4.12)

Figure 4.3: Block diagram for state space controller
Where $r$ is the reference input and $K$ is the matrix of control gains. For an optimal control of the system, a Linear-Quadratic Regulator (LQR) was used. To do this, assume that the reference, $r$, in Equation 4.12 is zero for equilibrium; then, minimize the quadratic cost function, Equation 4.13.

$$\int_{0}^{\infty} [x^T Q x + u^T R u] dt$$

(4.13)

Choose $Q$ and $R$ such that they constrain each state. Use $q_i$ and $r_i$ as typical choices for the diagonals of $Q$ and $R$, respectively [30].

$$Q = \begin{bmatrix} q_1 & & \\ & q_2 & \\ & & \ddots \\ & & & q_n \end{bmatrix}$$

(4.14)

$$R = \begin{bmatrix} r_1 & & \\ & r_2 & \\ & & \ddots \\ & & & r_n \end{bmatrix}$$

(4.15)

$$q_i = \frac{1}{t_{si}(x_{imax})^2}$$

(4.16)

$$r_i = \frac{1}{(u_{imax})^2}$$

(4.17)

Where $t_{si}$ is the desired settling time of the state $x_i$, $x_{imax}$ is the maximum acceptable value of the state $|x_i|$, $u_{imax}$ is the maximum acceptable value of input $|u_i|$, and $\rho$ is the positive constant to determine the system tracking versus control effort, or the system’s stiffness.
Equation 4.18 gives the optimal stabilizing control for the system after the quadratic cost function is minimized.

\[ u(t) = -R^{-1}B^T P x \]  

(4.18)

Therefore, the optimal gains are given in Equation 4.19.

\[ K = -R^{-1}B^T P \]  

(4.19)

Where \( P \) is the solution to the Algebraic Riccati Equation shown in Equation 4.20.

\[ 0 = PA + A^T P + Q - PBP^{-1}BP \]  

(4.20)

MATLAB will solve the LQR cost function by using the built in `lqr()` function [31]. By providing MATLAB the system matrices \( A, B, Q, \) and \( R \), MATLAB will return the state-feedback gain matrix, \( K \), the solution the Algebraic Riccati Equation, \( P \), and the closed-loop eigenvalues, \( e \), as seen below.

\[ [K, P, e] = lqr(A, B, Q, R); \]

### 4.3 Linear Controller

The state equations for the linear case are defined in Equation 4.21 and 4.22.

\[ \dot{x}_1 = x_2 \]  

(4.21)

\[ \dot{x}_2 = \frac{1}{Mr}u \]  

(4.22)

Where state one, \( x_1 \), is linear position and state two, \( x_2 \), is linear velocity. The state equations are derived from the system model from Section 4.1. Let the system be represented
by Equation 4.23.

\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{1}{M_F} \end{bmatrix} u \tag{4.23}
\]

The system was checked using the Controllability Test Matrix, Equation 4.24. The test matrix was full rank, therefore the system is controllable.

\[
C^* = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix} \tag{4.24}
\]

The \( Q \) and \( R \) matrices, Equations 4.14 and 4.15 respectively, were chosen with the properties listed in Table 4.1.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
Matrix & Property & Value \\
\hline
\( Q \) & \( t_{si} \) & 1 sec \\
& \( x_{1\text{max}} \) & 0.01 m \\
& \( x_{2\text{max}} \) & 0.7 m/s \\
\hline
\( R \) & \( \rho \) & TBD \\
& \( u_{\text{max}} \) & 9 Nm \\
\hline
\end{tabular}
\caption{\( Q \) and \( R \) matrix properties for linear system}
\end{table}

Using MATLAB, the optimal gains were calculated with varying \( \rho \) values. The lower values of \( \rho \) yielded a faster, stiffer system, while the higher values of \( \rho \) yielded the opposite. The results can be seen in Figure 4.4 and can be calculated using Equation 4.25 and 4.26.

\[
K_x = 900\rho^{-0.5} \tag{4.25}
\]

\[
K_{\dot{x}} = 558.56\rho^{-0.25} \tag{4.26}
\]
4.4 Angular Controller

As before, the physical model was used to derive the state space controller with optimal feedback. The system is described in Equation 4.27.

\[
\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{L}{J} \end{bmatrix} u
\]  

Equation 4.27

The angular system was proven to be controllable using the same technique as the linear system, and the \(Q\) and \(R\) matrices were chosen with the properties listed in Table 4.2.
Table 4.2: $Q$ and $R$ matrix properties for angular system

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q$</td>
<td>$t_{si}$</td>
<td>1 sec</td>
</tr>
<tr>
<td></td>
<td>$x_{1\text{max}}$</td>
<td>3 deg</td>
</tr>
<tr>
<td></td>
<td>$x_{2\text{max}}$</td>
<td>45 deg/s</td>
</tr>
<tr>
<td>$R$</td>
<td>$\rho$</td>
<td>TBD</td>
</tr>
<tr>
<td></td>
<td>$u_{\text{max}}$</td>
<td>9 Nm</td>
</tr>
</tbody>
</table>

The optimal gains were calculated with varying $\rho$ values. As before the lower values of $\rho$ yielded a faster, stiffer system. The results can be seen in Figure 4.5 and can be calculated using Equations 4.28 and 4.29.

$$K_\phi = 515.66\rho^{-0.5}$$ \hspace{1cm} (4.28)

$$K_\phi' = 327.92\rho^{-0.25}$$ \hspace{1cm} (4.29)
4.5 Controller Implementation

The total force required to control the single-input system can be define as the sum of the force required to move ROGAR linearly plus the force required to rotate ROGAR as seen in Equation 4.30.

\[ F_{total} = F_{linear} + F_{angular} \]  \hspace{1cm} (4.30)

From this relationship, the total wheel torque can be derived for the single-input system as seen in Equation 4.31.

\[ \tau_{total} = \tau_{linear} + \tau_{angular} \]  \hspace{1cm} (4.31)

Using Equation 4.31, the state space feedback becomes a sum of state variables multiplied...
by the state’s respective gain, as seen in Equation 4.32.

$$u = K_x x + K_{\dot{x}} \dot{x} + K_\phi \phi + K_\dot{\phi} \dot{\phi}$$  \hspace{1cm} (4.32)

Taking Equation 4.32 and converting it to a multi-output system for ROGAR’s two hub motors gives:

$$u_R = \frac{1}{2}[(K_x x + K_{\dot{x}} \dot{x}) + (K_\phi \phi + K_\dot{\phi} \dot{\phi})]$$  \hspace{1cm} (4.33)

$$u_L = \frac{1}{2}[(K_x x + K_{\dot{x}} \dot{x}) - (K_\phi \phi + K_\dot{\phi} \dot{\phi})]$$  \hspace{1cm} (4.34)

The $\frac{1}{2}$ distributes half of the required torque to the right wheel, and the other half to the left. The opposite signs in the middle of Equations 4.33 and 4.34 force one wheel to rotate forward while the other rotates backwards, thus forcing ROGAR to rotate about point $O$ as described in Figure 4.1.
Chapter 5

Controller Testing

5.1 Simulated Response to Step Input

After modeling the system and determining the gains, ROGAR was simulated in MATLAB. The simulation source code is available in Appendix D. Three simulations were conducted: 1) Linear Step Input Only, 2) Angular Step Input Only, and 3) Both Linear and Angular Step Input.

Figure 5.1 and Figure 5.2 show the results from the first simulation where ROGAR responded to a step input of 0.5 meters. As expected, the ROGAR only moved forward without rotation. The torques on both were equal in magnitude and direction.
Figure 5.1: Linear simulation: States vs. Time

Figure 5.2: Linear simulation: Outputs vs. Time
Figures 5.3 and 5.4 show the results from the second simulation where ROGAR responded to a step input of 22.5 degrees. ROGAR only rotated this time, where the torques on both were equal in magnitude, but opposite in direction.

Figure 5.3: Angular simulation: States vs. Time
Figures 5.5 and 5.6 show the results from the third simulation where ROGAR responded to a step input of both 0.5 meters and 22.5 degrees. Using the conclusions from Equations 4.33 and 4.34, ROGAR was able to correct for linear and angular errors. Wheel torques differed in magnitude this time to account for both linear and angular errors.
Figure 5.5: Linear and angular simulation: States vs. Time

Figure 5.6: Linear and angular simulation: Outputs vs. Time
The MATLAB simulation shows that state space controller that was developed can control ROGAR effectively. Using the two controller approach, linear and angular, allows the controller to adjust for multiple errors simultaneously. The gains derived from the Linear Quadratic Regulator prove to provide a balance between the tracking ability of ROGAR and the effort that the wheels must exert. In all the simulated cases, the controller was able to quickly and accurately move ROGAR. Each test the device was brought to equilibrium in approximately one second with no visible overshoot.

5.2 Device Response to Step Input

Once the controller was simulated in MATLAB and proven to stabilize the system, the controller was implemented on ROGAR. A LabView program was developed to set a desired position from the start location using the encoders as the feedback for position, velocity, angle, and angular velocity; by removing the human from the system, only the response of the controller was tested. The device attempted to get to its set location after the desired location was set via LabView and power was applied to the wheels by releasing the emergency stop.

Two types of step inputs were used to test the linear and angular controllers: a small step to mimic normal use of the device, and a large step to mimic a worst-case scenario. In many cases, the controller on the device did not behave as it did in the simulation: In some cases this led to steady state error, where the ROGAR never reached the set location, in other cases, the device quickly reached the set location and oscillated about the set point. By varying the $\rho$ parameter, the controller gains could be adjusted so that ROGAR would have a short rise time, minimal overshoot, and no steady state error.

In Figures 5.7, 5.8, 5.9, and 5.10, the smoothed response of ROGAR is shown for the various step responses and various $\rho$. 
Figure 5.7: Device response to 0.5m step input

Figure 5.8: Device response to 1m step input
Figure 5.9: Device response to 22.5deg step input

Figure 5.10: Device response to 45deg step input
5.3 Device Versus Simulated Response to Step Input

To get an idea of how well the computer model matched the device’s response, the data from the simulation tests and device tests were plotted against each other, Figures 5.11 and 5.12.

Figure 5.11 shows the system’s linear response at varying values of $\rho$. At a high value of $\rho$ (1.00), the system was much slower to respond than the predicted response from the simulation. Additionally, the higher values of $\rho$ led to steady state error at equilibrium, approximately 6cm. For median values of $\rho$ (0.50-0.75), the device responded very similarly to the predicted behavior. Both tests showed a quick response with little to no overshoot or steady state error. Finally, at very low values of $\rho$ (0.25), the device had a faster response than the predicted, but it also experienced a significant amount of overshoot, approximately 13cm.

![Figure 5.11: Device and simulation response to 0.5m step input](image)

Figure 5.12 shows the system’s angular response to varying $\rho$. In this case, the system behaved much differently than that of the simulation’s prediction. Values of $\rho$ above 0.10 led to a substantial amount of steady state error; this error ranged from approximately 3
degrees to upwards of 15 degrees. Values of $\rho$ below 0.10 keep the system stable and tracking the reference, however in all cases the system responded slower than the predicted behavior. The errors between the angular simulation and device response are most likely due to an underestimate of ROGAR’s moment of inertia, neglecting viscous friction, or disregarding back EMF in the wheels.

Figure 5.12: Device and simulation response to 22.5deg step input

5.4 Device Response to User Input

To test ROGAR’s response to a user input, a Logitech C920 Webcam was mounted in the ceiling using extruded aluminum and the lab’s drop ceiling frame. The camera’s view can be seen in Figure 5.13, where the camera was set to record with 20% brightness, 20% contrast, and 100% color intensity to ease image processing. Videos, recorded at 30 frames per second with a resolution of 640x480, were shot of the user interacting with the device and saved to the computer.
MATLAB’s built-in image processing tools were then used to post-process each frame of the recorded video. The code used four colored circles printed on standard 8.5x11” paper to determine the location and orientation of the patient and the device, as seen in Figure 5.13. The two outer circles (Light Blue and Black) were attached to the device, while the two inner circles (Dark Blue and Green) were attached to the patient. After each circle was identified, the code calculated the centroid of the patient (the median distance between the Dark Blue and Green circles), the centroid of the device (the median distance between the Light Blue and Black circles), the error in distance between the two centroids, and the error in angle between the patient and the device. Figure 5.14 was created to show where MATLAB had identified the four colored circles (plotted in their respective color) and two centroids (plotted in red) within each frame; red lines connected the respective dots to identify the patient and the device. When processing an entire video, Figure 5.14 was not displayed to speed up processing time. Once the video processing was complete, MATLAB
saved a text file with the frame number, patient location, device location, patient angle, and device angle. The source code for the image processing can be found in Appendix D.

The text file from processing the overhead video was then imported into the simulation code. The patient’s position from the video was used as the input to the simulation. For both the device test and the simulation, the linear controller used a $\rho$ of 0.35 for the system’s stiffness. By importing the patient’s position data, the actual device response and the simulated device response could be directly compared to see how well the simulation matched reality. Figure 5.15 shows the device and the simulated device responses to a patient walking forward 1m.
Figure 5.15: Linear controller: Simulated response vs. device response with patient input

Figure 5.16 shows three errors. First, the device error is the error between the patient’s position and the device’s position. Second, the simulation error is the error between the patient’s position and the device’s simulated response. Finally is the prediction error, the error between the device’s actual response and the simulated response. It was found that both the device and simulation tracked well to the patient. ROGAR followed the patient with approximately an error of 15cm, while the simulation predicted an error of approximately 10cm. The simulation matches reality extremely well in that the simulation has an average error of 2.7cm and a max error of 7cm from the actual device’s response. Looking back at Section 2.2, the IR sensor has an average error of 5.9cm which contributes to the prediction error between the simulated response and the actual response. Other sources of error are the simulation assuming that the patient’s position is perfectly measured, that there is no noise in the incoming IR signal, the video was perfectly digitized, and the fact that system model used is relatively simplistic and assumes little about the internals of the motors.
Figure 5.16: Linear controller: Error between simulated response and device response

Next, the angular controller, using a $\rho$ of 0.1, was tested on ROGAR. Like the linear case, a video was recorded and processed to determine the patient’s angle and the device’s response. The patient’s angle was then imported into the MATLAB simulation, and a simulated response was generated. Figure 5.17 shows the device and the simulated device responses to a patient spinning a full 360 degree circle.

Figure 5.18 shows the three errors associated with the device and simulation: the device error, the simulation error, and the prediction error. The angular controller works, but it does not work as well as the linear controller.
Figure 5.17: Angular controller: Simulated response vs. device response with patient input

Figure 5.18: Angular controller: Error between simulated response and device response

The device lagged the patient an average of 15 degrees and had locations that it lagged the
patient as much as 30 degrees. The simulation predicted a much faster, tighter response with an average error of 6 degrees and a maximum of 28 degrees. The discrepancies in device responses between the simulation and reality lead to an average prediction error of 8 degrees. Unlike the linear case where the device and simulation errors followed the same shape, the angular case is much more sporadic. These inconsistencies are most likely due to measurement errors associated with the complementary filter. Though the filter proved to be very accurate during testing, the filter seemed to have interference problems when implemented on the device. Electromagnetic noise from computers and other electronic devices in the room would force the magnetometer to have jumps in readings as much as 90 degrees. This interference is most likely the largest source of error when tracking the orientation of the patient.

5.5 Conclusions

 Millions of American adults are going through physical therapy to improve their ability to walk. From geriatric patients with weakened coordination, traumatic brain injury patients, to patients with lower-limb amputations, these patients come from all walks of life. The Robot for Observational Gait Assessment and Rehabilitation (ROGAR) will aid physicians and therapists in their ability to quickly and accurately gather quantitative gait measurements. In this research, the goals discussed in section 1.3 were achieved. The contributions of this thesis are discussed in the following.

 Design and build a robotic platform. The current robotic platform works well for the ROGAR research. The frame is robust enough to support the patient’s weight without concern. The hardware has been laid out so that each piece is easily accessible, yet is condensed enough that the sides and front are open for future expansion to the project. The hardware that is currently used is sized well for the application: the wheels provide sufficient torque to adequately move ROGAR; the servos are capable of supplying the wheels’ required current; and the processor handles all the necessary computations at a high rate. Like the mechanical side, the electrical has been laid out so that changes and additions to the device can easily be made.
Develop and implement a method to track patient position and orientation. The device uses an infrared (IR) distance sensor to track the patient’s position forward or backward of ROGAR’s wheels’ center of rotation. This sensor is attached to a linear bearing which can be adjusted to the patient’s height. The sensor has an average error of 5.9cm associated with it. However, the error is the worst at either end of the usable range of the sensor, fortunately these areas of the range are rarely need to be used. The complementary filter used to track the patient’s orientation worked much better in the desktop trials than in the implementation on the device due to interference. With a complementary filter gain of 0.97, the filter had an average error of approximately 1 degree over a range of angular velocities during the benchtop tests. However, when the complementary filter was implemented on ROGAR, it was sensitive to changing magnetic fields in the room. It would be interesting to test the device in a more clinical setting to see if the complementary filter performed better in a larger space. The issues with the complementary filter most likely were the largest source of error when trying to control the orientation of the device relative to a patient.

Design and implement an optimal controller. An optimal state space controller was implemented using a linear quadratic regulator. For a step input, the simulation calculated that both the linear and angular controllers had a settling time around 1 second and no overshoot. The controller also adequately compensated the device when there were both linear and angular errors. The devices step response varied slightly from the simulations that were run. The linear controller was quick to respond in all cases. However for higher values of $\rho$, the device was left with some steady state error; the extreme lower values of $\rho$ forced the system to oscillate about equilibrium before settling. The angular controller did not perform as well as the linear. For high values of $\rho$, the device was left with a significant amount of steady state error, while extremely low values forced the system to equilibrium with no oscillations about equilibrium. When ROGAR was used with a patient input, the simulation showed a fast linear response that matched very closely to when the controller was run on the device. On average, the simulated device had an average of 10cm of error when the patient was walking compared to the actual device’s error of 15cm. The error of the actual device was very predictable based on the simulations calculations.
The angular controller did not simulate reality as well as the linear case. The simulation predicted an average error of 6 degrees between the patient and the device. In reality this error was closer to an average of 15 degrees for a full rotation of the patient. The errors between the simulation and the actual device most likely are due to the assumption of a perfect measurement in the simulation, whereas the complementary filter implemented on the device is susceptible to noise and interference. Overall, the optimal state space controller is able to actuate ROGAR relative to the patient’s movements.

5.6 Future Work

Several ideas relating to this research were discussed before and during the work on this thesis that fell outside the scope of the research goals. These include wireless communication, FPGA filtering, I2C communication, an adaptive controller, and sensors to track a patient’s gait.

The wireless communication has been started by an undergraduate in the lab. The wireless router has to share the same IP address as the cRIO, and the network shared variable library must be deployed from the LabView project to the cRIO. Currently, the cRIO is able to push variables to the wireless router and local network. These values are then accessible by any computer on the network that is calling them. The wireless communication would be ideal for ROGAR so that it can be a stand-alone device. Additionally, wireless communication would allow patient data to be stored live, therefore allowing physicians to track progress over multiple appointments.

The current robot does all signal filtering on the Real-Time side of the processor. This takes a significant amount of processing power and slows down other functions. After talking to a National Instruments representative, he suggested that the filtering be moved to the FPGA side of the processor. This would speed up the filtering and allow the Real-Time side to focus on the control aspect of the project.

Next, the magnetometer and gyroscope could be better integrated with ROGAR. The NI 9402 was purchased solely to integrate directly with the sensors. However, I2C is not yet well supported by LabView for their FPGA processors. Furthermore, the sensor unit that
the patient wears could be wirelessly integrated instead of having the BNC cables attaching the sensing unit to the cRIO. This could easily be implemented using the existing Arduino and two XBee radios.

An adaptive controller could be implemented to adjust the controller gains based on the force the patient is applying to the parallel bars on ROGAR. If a patient was relying on ROGAR to support some of their weight, more torque could be applied to the wheels so that the device operates the same as if no weight were applied.

Finally, sensors could be added to measure a patient’s gait – the ultimate goal of the ROGAR project. There are tons of possibilities to address this problem from IR sensors to cameras and video processing. It will be interesting to see this project progress once this portion of the research concludes.
Bibliography


Appendix A

LabView and Device References

A.1 National Instruments References

LabView Tutorials http://www.ni.com/tutorials/
NI Discussion Forums http://forums.ni.com/
NI Technical Support http://www.ni.com/support/
(866) 275-6964
NI Customer Service 1-800-531-5066
NI Support 1-800-433-3488
<table>
<thead>
<tr>
<th>Kevin Warner</th>
<th>11500 N. Mopac Expwy.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELP Engineer</td>
<td>Austin, TX 78759-3504</td>
</tr>
<tr>
<td>Applications Engineering</td>
<td>Office: (512) 683-0786</td>
</tr>
<tr>
<td><a href="mailto:kevin.wenner@ni.com">kevin.wenner@ni.com</a></td>
<td>Fax: (512)683-5678</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mark Eadie</th>
<th>2820 Audubon Village Dr. PMB#362</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Engineer &amp; District Sales Manager</td>
<td>Audubon, PA 19403</td>
</tr>
<tr>
<td>Philadelphia, Eastern, and Central PA</td>
<td>Office: (610) 783-0641</td>
</tr>
<tr>
<td><a href="mailto:mark.eadie@ni.com">mark.eadie@ni.com</a></td>
<td>Mobile: (734) 915-0181</td>
</tr>
<tr>
<td></td>
<td>Fax: (512)683-7924</td>
</tr>
</tbody>
</table>
A.2 Charging ROGAR

1. Turn off the main power to both the cRIO/sensors and the wheels, see Figure C.3 in Appendix C.

2. Switch the Charging Switches to **Charge** (Towards the front of the device).

3. Insert the banana plugs from the chargers to the charging ports, making sure to match the polarities (Red and Black).

4. Set the chargers to the correct setting (play: select/OK, +: up, -: down, stop: cancel)
   
   (a) Wheel batteries - Pb Charge, 1.5A, 12V(6P)
   
   (b) cRIO/Sensors Battery - Pb Charge, 0.5A, 12V(6P)

5. Hold the play button for approximately 3 seconds to start the charge cycle

6. After charging, toggle the Charging Switches back to the **On** position to re-wire the circuits.

NOTES:

- The charge cycle takes approximately 2 hours to complete, the chargers will beep when finished.

- The chargers will say *Connection Break* if the banana plugs have not been plugged in fully, the switches have not been toggled to the correct position, or there is a bad connection in the battery box.

- Do not leave the chargers unattended or left on overnight.
A.3 LabView Program Layout

At the completion of this portion of the ROGAR research, the LabView code is on build 4. These files are located in the Robotic Parallel Bars Folder. When starting LabView, make sure to open the LabView Project titled Robotic Parallel Bars - Build 4. This will link all of the necessary VIs to run the device. All of the codes are described in the VI properties, these descriptions can be view by using the LabView Context Help. The project is set up as follows:

File Location:
Robotic Parallel Bars/Code/LabView/Robotic Parallel Bars/4th Build

- My Computer
  - Text File Programs
    * File that run on the desktop and are able to write data to the computers hard drive (e.g. C:/)
    * A program must be running on the cRIO and be pushing global variables to the desktop. These global variables must be called on the desktop side.
    * These programs are used to create tab delimited *.txt files to collect data from the device. These text files are easily imported into Excel or MATLAB

- cRio9076-Parallel-Bars (130.203.244.68)
  - Top Level Programs
    * These are the programs that actually run the robot. They pull all the necessary VIs to read the sensors and control the device.

  - Pull Into Controller
    * These files provide data that can be directly used in the controller.
    * In most cases some level of processing is done at this level
    * All units are in degrees or meters.
- **Sub VI**
  * These files read the FPGA side and convert the data to semi-useful values.

- **Chassis (cRIO-9076)**
  * This is where the FPGA code is implemented (File: Sensor Read 2 (FPGA).vi).
  Its important that all the FPGA code be in *one* VI.
  * This is also where all IO pins are defined.
  * The FPGA code must be compiled before anything above this will run. To compile the code, simply run the FPGA code. This will bring up a window, select use local compiler. The compiling process will take upwards of a half hour, so plan accordingly.

- **Variables - network - RT(separate).lvlib**
  * This is where all global variables are define.
  * To access these variables on the wireless network, Right Click and select *Deploy All*.

**NOTES:**

- To have the project start when the cRIO boots, Right Click *Build Specifications* and select *Build All*. This will create a Real-Time Application, Right Click this and select *Run as startup*
A.4 Running the State Space Controller Program

1. Check that the E-Stop is pressed in.

2. Turn on the main power to the wheels and the sensor.

3. Click **Run** to state the LabView Code.

4. Click **Start Motors**.

5. Select whether the linear, angular, or both controllers will be used (Right Hand Side).

6. Press **Zero Heading** twice to bias the patient and robot headings. Make sure both the robot and patient heading are near zero before starting the robot.
7. Press *Encoder Reset* twice to bias the encoders.

8. Release the E-Stop.

**NOTES:**

- If one wheel seems stronger than the other, check the connections on the hub board. Otherwise, the batteries need charged.

- If either one of the headings are displaying a constant zero, the associated Arduino needs to be reset and the LabView program needs to be restarted.

- The large stop button in the middle of this front panel simply stops the LabView code.

- The gains associated with either controller can be modified by going to the block diagram and changing the respective Rho value (rho_x is for linear, rho_a is for angular)
Appendix B

Device Photos and Mechanical Drawings
B.1 Frame Pictures

Figure B.1: Frame front isometric
Figure B.2: Frame rear isometric
Figure B.3: Frame rear
Figure B.4: Frame side

Figure B.5: Left side hardware
Figure B.6: Right side hardware
B.2 Reference Graphics

Gear Ratio = 0.0481:1

Figure B.7: Gear ratio for hub motors
Figure B.8: Hub motor assembly

B.3 Shop Drawings
Appendix C

Electrical Schematics and Diagrams

Figure C.1: Hub board I/O layout
Figure C.2: cRIO Channel Layout

(a) NI 9401

<table>
<thead>
<tr>
<th>Ch</th>
<th>Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Encoder - Right A</td>
</tr>
<tr>
<td>1</td>
<td>Encoder - Right B</td>
</tr>
<tr>
<td>2</td>
<td>Encoder - Left A</td>
</tr>
<tr>
<td>3</td>
<td>Encoder - Left B</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

(b) NI 9402

<table>
<thead>
<tr>
<th>Ch</th>
<th>Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Patient Magnetometer</td>
</tr>
<tr>
<td>1</td>
<td>Robot Magnetometer</td>
</tr>
<tr>
<td>2</td>
<td>Patient Gyroscope</td>
</tr>
<tr>
<td>3</td>
<td>Robot Gyroscope</td>
</tr>
</tbody>
</table>

(c) NI 9263

<table>
<thead>
<tr>
<th>Ch</th>
<th>Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Wheel - Right</td>
</tr>
<tr>
<td>1</td>
<td>Wheel - Left</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

(d) NI 9205

<table>
<thead>
<tr>
<th>Ch</th>
<th>Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>IR 1</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Force 1</td>
</tr>
<tr>
<td>3</td>
<td>Force 2</td>
</tr>
<tr>
<td>4</td>
<td>Force 3</td>
</tr>
<tr>
<td>5</td>
<td>Force 4</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>
Figure C.3: Switch, battery, and charging port layout
Note:
- Belt Power: 9V Battery
- Parallel Bar Power: 12V connection from on-board battery

Robotic Parallel Bars
Arduino & Protoboard

Andrew Homich
Rev 2.0
1/1/2014
Page 4 of 4
Appendix D

MATLAB Code

Listing D.1: MATLAB simulation (step input) source code

```matlab
% Robotic Parallel Bars Model
%
% Andrew Homich (ajh5267@psu.edu)
% 1/10/14
%
% Program to calculate and simulate the state space controller

clear all;
close all;
cle;

% Measurements:
% IR Sensor
% Right Wheel Encoder
% Left Wheel Encoder
% Patient Mag
% Patient Gyro
% Robot Mag
% Robot Gyro

% System Properties
% [kg]
M = 42.25;
J = 12.7; % Moment of inertia for robot [kg*m^2]
L = 1.00/2; % 1/2 of Robot Inside Width [m]

r = 0.2032/2; % Wheel radius [m]
m = 4.8; % Wheel weight [kg]
I = (m*r^2)/2; % Wheel moment of inertia [kg*m^2]
```

---

Linear System

---

84
% State Variables:
    % X1: Robot position relative to patient
    % X2: Robot velocity relative to patient

% System Equations / Matrices
fprintf( ’Linear System Matrices ———— \n’ )

% State Matrices
Ax = [ 0 1; 0 0 ];
Bx = [ 0 1/(M*r) ];
Cx = [ 1 0 ];
Dx = [ 0 ];
syss = ss(Ax,Bx,Cx,Dx);

% Controllability Test Matrix — With Torque at Pivot as Input
Cstar = [ Bx Ax+Bx ];
rank_Cstar = rank(Cstar);

% Observability Test Matrix — Measuring ball velocity
Ostar = [ Cx; Cx*Ax ];
rank_Ostar = rank(Ostar);

% = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

% Angular System

% = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

% State Variables:
    % X3: Robot angle relative to patient
    % X4: Robot angular velocity relative to patient

% System Equations / Matrices
fprintf( ’Angular System Matrices ———— \n’ )

% Matrices
Aa = [ 0 1; 0 0 ];
Ba = [ 0 (I*L)/(J*r) ];
Ca = [ 1 1 ];
Da = [ 0 ];
syss = ss(Aa,Ba,Ca,Da);

% Controllability Test Matrix — With Torque at Pivot as Input
Cstar = [ Ba Aa+Ba ];
rank_Cstar = rank(Cstar);

% Observability Test Matrix — Measuring ball velocity
Ostar = [ Ca; Ca*Aa ];
rank_Ostar = rank(Ostar);

% = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

% Linear Quadratic Regulator
fprintf('---------- Linear Gains ---------- 
')
t_s = 1;
rho = 0.5;

% Using LQR to choose poles
% 1/"bad" position^2 , 1/"bad" velocity^2
% Distances are in meters [m]
% Velocities meters per sec [m/s]
Qx = diag([1/(t_s*(0.01^2)) 1/(t_s*(0.7^2))]);
% "bad" max torque of wheel (change weight down to make more aggressive)
% Torques are in Newton Meters [N.m]
Rx = rho*[1/(9^2)];
[Kx,Sx,ex] = lqr(Ax,Bx,Qx,Rx);
lambda_c = eig(Ax-Bx*Kx);
Kx

fprintf('---------- Angular Gains ---------- 
')
t_s = 1;
rho = .05;

% Using LQR to choose poles
% 1/"bad" angle^2 , 1/"bad" angular velocity^2
% Angles are in radians (Degrees converted to radians)
% Angular Velocities are in radians per sec (Degrees converted to radians)
Qa = diag([1/(t_s*(1*pi/180)^2) 1/(t_s*(45*pi/180)^2)]);
% "bad" max torque of wheel (change weight down to make more aggressive)
% Torques are in Newton Meters [N.m]
Ra = rho*[1/(9^2)];
[Ka,Sa,ea] = lqr(Aa,Ba,Qa,Ra);
lambda_c = eig(Aa-Ba*Ka);
Ka

%% Simulation
play_sim = 1;
if play_sim == 1
    add_disturbances = 0;
    add_noise = 0;
    simtime = 5;

    figure(1)
    clf

    % Setup
    num_state_vars = 4;         % Number of state variables
    dt = 0.01;                  % time step of the simulation
    endtime = simtime+dt;       % duration of the simulation

    % Initial conditions
    x0 = [1 0 45*pi/180 0];    %[Pos Vel Angle A.Vel]
    u = [0 0];                 %[RightWheel LeftWheel]
% Gains
G1 = -Kx(1); % Position
G2 = -Kx(2); % Velocity
G3 = -Ka(1); % Angle
G4 = -Ka(2); % Angular velocity

% Time Stuff
numsteps = ceil(endtime/dt) + 1; % Calculate number of iterations
t = 0:dt:(numsteps-1)*dt; % Define time matrix

% System Stuff
% Define state variable matrix
x = zeros(numsteps, num_state_vars);
% Define initial conditions in matrix (defined above)
x(1,:) = x0;

% Define output matrix
y = zeros(numsteps, 2);
y(1,1) = x(1,1); % Define initial output (position)
y(1,2) = x(1,3); % Define initial output (angle)

% Define x_dot matrix (state variable)
xdot = zeros(numsteps, num_state_vars);
% Define matrix to track ball velocity
ydot = zeros(numsteps, 1);

for i = 1:numsteps - 1
% Calculate input
ux(i) = G1*x(i,1)+G2*x(i,2);
ua(i) = G3*x(i,3)+G4*x(i,4);

uR(i) = 1/2*ux(i) + 1/2*ua(i);
uL(i) = 1/2*ux(i) - 1/2*ua(i);

% Create a disturbance
if add_disturbances == 1
    if i == round(numsteps*4/8)
        x(i,1) = -0.2;
        x(i,3) = 5*pi/180;
    end
end

% Calculate the x_dot's (equations of motion)
xdot(i,1) = x(i,2);
xdot(i,2) = 1/M*ux(i);
xdot(i,3) = x(i,4);
xdot(i,4) = 1*L/J*ua(i);

% Save the current states to next iteration
x(i+1,:) = xdot(i,:)*dt + x(i,:);

% Create Noise
if add_noise == 1
    noise = L/2*0.1*randn(1)/3;
end
y(i+1,:) = [x(i+1,1)+noise x(i+1,3)+noise];

else
    % Sensor perfectly measures linear position
    y(i+1,:) = [x(i+1,1) x(i+1,3)];
end

maxPlotTime = 3;

maxPlotTime = 3;

% Plot States vs Time
figure(1)
subplot(2,1,1)
plot(t(1:i),x(1:i,1),'-b')
axis([0 maxPlotTime -1.2 1.2])
ylabel('X Pos (m)')
grid on

subplot(2,1,2)
plot(t(1:i),x(1:i,3)*180/pi,'-b')
axis([0 maxPlotTime -5 60])
ylabel('X Vel (m/s)')
grid on

% Plot Position vs Time
figure(2)
subplot(2,1,1)
plot(t(1:i),x(1:i,1),'-b')
axis([0 maxPlotTime -1.2 1.2])
ylabel('Linear Error (m)')
xlabel('Time (s)')
grid on

subplot(2,1,2)
plot(t(1:i),x(1:i,3)*180/pi,'-b')
axis([0 maxPlotTime -60 60])
ylabel('Angular Error (Deg)')
xlabel('Time (s)')
set(gca,'YTick',[60 30 60])
grid on

% Plot Outputs vs Time
figure(3)

subplot(2,1,1)
plot(t(1:i),uR(1:i).*I,'-b')
axis([0 endtime-dt -15 15])
ylabel('Right Wheel Torque (N.m)')
xlabel('Time (s)')
grid on

subplot(2,1,2)
plot(t(1:i),uL(1:i).*I,'-b')
axis([0 endtime-dt -15 15])
ylabel('Left Wheel Torque (N.m)')
xlabel('Time (s)')
grid on

end
Listing D.2: MATLAB simulation (patient test data input) source code

```matlab
% Robotic Parallel Bars Model - Patient Test Data Input
%
% Andrew Homich (ajh5267@psu.edu)
% 1/10/14
%
% Program to calculate and simulate the state space controller

clear all;
clc;

% Measurements:
% IR Sensor
% Right Wheel Encoder
% Left Wheel Encoder
% Patient Mag
% Patient Gyro
% Robot Mag
% Robot Gyro

% System Properties
M = 42.25;        %[kg]
J = 12.7;         %[Moment of inertia for robot [kg*m2]]
L = 1.00/2;      %[1/2 of Robot Inside Width [m]]

r = 0.2032/2;    %[Wheel radius [m]]
m = 4.8;          %[Wheel weight [kg]]
I = (m*r^2)/2;   %[Wheel moment of inertia [kg.m2]]

% System Equations / Matricies
fprintf('Linear System Matricies 

% State Variables:
% X1: Robot position relative to patient
% X2: Robot velocity relative to patient

% System Equations / Matricies
sysx = ss(Ax, Bx, Cx, Dx);

% Controllability Test Matrix - With Torque at Pivot as Input
C_star = [Bx * Ax * Bx];
rank_C_star = rank(C_star);
```
% Observability Test Matrix – Measuring ball velocity
O_star = [Cx; Cx*A];
rank_O_star = rank(O_star);

% Angular System
% State Variables:
% X3: Robot angle relative to patient
% X4: Robot angular velocity relative to patient

% System Equations / Matricies
fprintf('------------- Angular System Matrices ----------- \n');
% Matricies
Aa = [0 1;
     0 0];
Ba = [0 (I(1*L)/(J*r)) ];
Ca = [1 1];
Da = [0];
sysa = ss(Aa,Ba,Ca,Da);

% Controllability Test Matrix – With Torque at Pivot as Input
C_star = [Ba Aa; Ba];
rank_C_star = rank(C_star);

% Observability Test Matrix – Measuring ball velocity
O_star = [Ca; Ca*Aa];
rank_O_star = rank(O_star);

% Linear Quadratic Regulator
fprintf('------------- Linear Gains ----------- \n');
t_si = 1;
rho = 0.35;%0.5
% Using LQR to choose poles
% 1/"bad" position"^2, 1/"bad" velocity"^2
% Distances are in meters [m]
% Velocities meters per sec [m/s]
Qx = diag([1/(t_si*(0.01^2)) 1/(t_si*(0.7^2))]);
% "bad" max torque of wheel (change weight down to make more aggressive)
% Torques are in Newton Meters [N.m]
Rx = rho*[1/(9^2)];
[Kx,Sx,ex] = lqr(Ax,Bx,Qx,Rx);
lambda_c = eig(Ax-Bx*Kx);
Kx

fprintf('------------- Angular Gains ----------- \n');
% Using LQR to choose poles
% 1/('bad' angle)^2 , 1/('bad' angular velocity)^2
% Angles are in radians (Degrees converted to radians)
% Angular Velocities are in radians per sec (Degrees converted to radians)
Qa = diag([(1/(t_s*(t_s+180)^2) 1/(t_s*(45*t_s+180)^2))]);
% "bad" max torque of wheel (change weight down to make more aggressive)
% Torques are in Newton Meters [N.m]
Ra = rho*[1/((9)^2)];
[Ka, Sa, ea] = lqr(Aa, Ba, Qa, Ra);
lambda_c = eig(Aa-Ba*Ka);

% Simulation
play_sim = 1;

% Setup
num_state_vars = 4;
% Number of state variables
dt = 0.033;
% time step of the simulation

% Initial conditions
x0_D = [0 0 0 pi/180 0];
%[Pos Vel Angle A.Vel] Device
x0_P = [0 0 0 pi/180 0];
%[Pos Vel Angle A.Vel] Patient
u = [0 0];
%[RightWheel LeftWheel]

% Gains
G1= -Kx(1);
% Position
G2= -Kx(2);
% Velocity
G3= -Ka(1);
% Angle
G4= -Ka(2);
% Angular velocity

% Time Stuff
numsteps=length(dev);
% Calculate number of iterations
t=0:dt:(numsteps-1)*dt;
% Define time matrix

% System Stuff
% Define state variable matrix
x_D=zeros(numsteps,num_state_vars);
x_P=zeros(numsteps,num_state_vars);
% Define initial conditions in matrix (defined above)
x_D(1,:)=x0_D;
x_P(1,:)=x0_P;
error = zeros(numsteps, 2);

y=zeros(numsteps, 2);
% Define output matrix
y(1,1)=x_D(1,1);
% Define initial output (position)
y(1,2)=x_D(1,3);
% Define initial output (angle)
% Define x_dot matrix (state variable)
xdot=zeros(numsteps,num_state_vars);

% Define matrix to track ball velocity
ydot=zeros(numsteps,1);

for i=2:1:numsteps-1
    % Patient's Movements
    x_P(i,1) = 0; % Position
    x_P(i,2) = (x_P(i,1)-x_P(i-1,1))/dt; % Velocity
    x_P(i,3) = dev(i,3); % Angle
    x_P(i,4) = (x_P(i,3)-x_P(i-1,3))/dt; % A Vel

    % Device Movements
    % Calculate input
    errorL = -(x_P(i,1)-x_D(i,1));
    errorA = -(x_P(i,3)-x_D(i,3));
    error(i,:) = [errorL errorA];

    ux(i) = G1*errorL+G2*x_D(i,2);
    ua(i) = G3*errorA+G4*x_D(i,4);

    uR(i) = 1/2*ux(i) + 1/2*ua(i);
    uL(i) = 1/2*ux(i) - 1/2*ua(i);

    % Calculate the x_dot's (equations of motion)
    xdot(i,1)= x_D(i,2);
    xdot(i,2)= 1/M*ux(i);
    xdot(i,3)= x_D(i,4);
    xdot(i,4)= 1/L/J*ua(i);

    % Save the current states to next iteration
    x_D(i+1,:) = xdot(i,:)*dt+x_D(i,:);
end

% Plot States vs Time
figure(1)
% subplot(2,2,1)
% plot(t(1:i),x(1:i,1),'-b')
% axis([0 maxPlotTime -0.2 1.2])
% ylabel('X Pos (m) ')
% grid on

% subplot(2,2,2)
% plot(t(1:i),x(1:i,2))
% axis([0 maxPlotTime -2 2])
% ylabel('X Vel (m/s) ')
% grid on

% subplot(2,2,3)
% plot(t(1:i),x(1:i,3).*180/pi)
% axis([0 maxPlotTime -5 60])
% ylabel('Angle (Deg) ')
% grid on
% subplot(2,2,4)
plot(t(1:i),x(1:i,4).*180/pi)
axis([0 maxPlotTime -180 180])
gtext('Ang Vel (Deg/s)')
xlabel('Time (s)')
grid on

% Plot Position vs Time
clf
figure(1)
subplot(2,1,1)
hold on
plot(t(1:i),x_P(1:i,1),'-b')
plot(t(1:i),x_P(1:i,2),'-r')
plot(t(1:i),x_D(1:i,1),'-c')
plot(t(1:i),x_D(1:i,2),'-m')
hold off
axis([0 maxPlotTime 1.2 1.2])
ylabel('Linear Error (m)')
xlabel('Time (s)')
grid on

subplot(2,1,2)
hold on
plot(t(1:i),x_P(1:i,3).*180/pi,'-b')
plot(t(1:i),x_P(1:i,4).*180/pi,'-r')
plot(t(1:i),x_D(1:i,3).*180/pi,'-c')
plot(t(1:i),x_D(1:i,4).*180/pi,'-m')
hold off
axis([0 maxPlotTime 60 60])
ylabel('Angular Error (Deg)')
xlabel('Time (s)')
set(gca,'YTick',[60:30:60])
grid on

% figure(3)
hold on
plot(t(1:i),error(1:i,1),'-b')
plot(t(1:i),error(1:i,2).*180/pi,'-m')
hold off
axis([0 maxPlotTime -1.2 1.2])
ylabel('Linear Error (m)')
xlabel('Time (s)')

% figure(2)
subplot(2,1,1)
plot(t(1:i),ux(1:i),'-b')
axis([0 8 -15 15])
ylabel('Right Wheel Torque (N.m)')
xlabel('Time (s)')
grid on

subplot(2,1,2)
plot(t(1:i),ua(1:i),'-b')
axis([0 8 -15 15])
axis([0 endtime-dt -15 15])
ylabel('Left Wheel Torque (N.m)')
xlabel('Time (s)')
grid on

[t(1:501)’ x(1:501,:') uR(1:501)’ uL(1:501)’]
end
data2write = [t’ x_D x_P];
dlmwrite('simData.txt',data2write,'delimiter','\t');
Listing D.3: MATLAB video processing code

```matlab
% Movie Processor
% Example from Color-Based Segmentation with Live Image Acquisition
% See: http://www.mathworks.com/products/image/description5.html

clear all;
close all;
clc;

% for countDown = 10:-1:1
% pause(1)
% countDown
% end

vidIn = VideoReader('C:\Users\JMoore Lab\Desktop\Robotic Parallel Bars\Experiments\Video Controller Test\Linear_rL035_HC.wmv')
nFrames = vidIn.NumberOfFrames
vidHeight = vidIn.Height;
vidWidth = vidIn.Width;
data = zeros(nFrames,9);

%Read one frame at a time.
for frameCount = 1:nFrames
    try
        frame = read(vidIn,frameCount);
    catch
        continue;
    end

    %Lens correction
    frame_c = lensdistort(frame,0.03,'bordertype','fit');

    %Thresholding the image on each color plane
    im = im2double(frame_c);
    [r c p] = size(im);
    % Extract individuals plane from RGB image
    imR = squeeze(im(:,:,1));
    imG = squeeze(im(:,:,2));
    imB = squeeze(im(:,:,3));

    %Thresholding on individual planes
    imBinaryR = im2bw(imR,graythresh(imR));
    imBinaryG = im2bw(imG,graythresh(imG));
    imBinaryB = im2bw(imB,graythresh(imB));
    imBinary = imcomplement(imBinaryR & imBinaryG & imBinaryB);

    %Clean up picture
    se = strel('disk',12);
    imErode = imerode(imBinary,se);
    se = strel('disk',6);
    imDilate = imdilate(imErode,se);
    imClear = imclearborder(imDilate);
    imClean = imClear;

    %Segmented gray-level image
    [labels,numLabels] = bwlabel(imClean);
```
%Initialize Matrices
rLabel = zeros(r,c);
gLabel = zeros(r,c);
bLabel = zeros(r,c);

%Get average color vector for each labeled region
for i = 1:numLabels
    rLabel(labels==i) = median(imR(labels==i));
    gLabel(labels==i) = median(imG(labels==i));
    bLabel(labels==i) = median(imB(labels==i));
end
imLabel = cat(3,rLabel,gLabel,bLabel);

%Find Circles
[centers, radii, metric] = imfindcircles(imBinary,[10 25]);
centersStrong = centers(1:4,:);
radiiStrong = radii(1:4);
metricStrong = metric(1:4);

%Identify Color
Black = [0 0];
Blue = [0 0];
Green = [0 0];
Cyan = [0 0];

%Identify circle colors
for i = 1:4
    R = imLabel(round(centersStrong(i,2)),round(centersStrong(i,1)),1);
    G = imLabel(round(centersStrong(i,2)),round(centersStrong(i,1)),2);
    B = imLabel(round(centersStrong(i,2)),round(centersStrong(i,1)),3);
    if (R<0.6 && G<0.6 && B<0.6) %Black
        Black = [centersStrong(i,1),centersStrong(i,2)];
    elseif(R<0.2 && G<0.5 && B>0.7) %Blue
        Blue = [centersStrong(i,1),centersStrong(i,2)];
    elseif(R<0.3 && G>0.5 && B>0.7) %Cyan
        Cyan = [centersStrong(i,1),centersStrong(i,2)];
    else %Green
        Green = [centersStrong(i,1),centersStrong(i,2)];
    end
end

%Calculate Linear Distance
deviceL = [mean([Black(1),Cyan(1)]),mean([Black(2),Cyan(2)])];
patientL = [mean([Green(1),Blue(1)]),mean([Green(2),Blue(2)])];
errorL = sqrt((deviceL(1)-patientL(1))^2 + (deviceL(2)-patientL(2))^2);

%Calculate Angles
%Define "up" as zero
deviceA = atan2(Black(1)-Cyan(1),Black(2)-Cyan(2))*-180/pi;
patientA = atan2(Green(1)-Blue(1),Green(2)-Blue(2))*-180/pi;
errorA = patientA - deviceA;

% clf
figure(1)
hold on
plot(Black(1), Black(2), 'ok')
plot(Cyan(1), Cyan(2), 'oc')
plot([Black(1) Cyan(1)], [Black(2) Cyan(2)], 'r')
plot(deviceL(1), deviceL(2), 'or')
plot(Green(1), Green(2), 'og')
plot(Blue(1), Blue(2), 'ob')
plot([Green(1) Blue(1)], [Green(2) Blue(2)], 'r')
plot(patientL(1), patientL(2), 'or')
hold off
axis([0 vidWidth 0 vidHeight])
xlabel('Pixels')
ylabel('Pixels')
drawnow

% Conversion: 9 Pixels/inch or 354.33 px/m
% Resolution: 0.1 inch or 3mm
newDataPt = [frameCount deviceL patientL errorL deviceA patientA errorA]
data(frameCount,:) = newDataPt;
catch err
    fprintf('CATCH')
end
end
dlmwrite('movData.txt', data, 'delimiter', '\t');
Appendix E

LabView Code

E.1 FPGA

1. Replace this Input Signal boolean with an FPGA I/O Node and select the input pin that you wish to use. This VI measures the high and low periods of a pulse width modulated signal for use on a NI-IMAQ I/O FPGA device.

- High Period is denoted with "H"
- Low Period is denoted with "L"

Read the Encoders
- Inputs: Pin for Ch A & Ch B of rotary encoder
- Black Magic from downloaded FPGA Encoder VI Block
- Outputs: Position, Velocity, and Direction

- Code was adapted from compactrio_motor_control_basics.pdf

Read the Right Encoder

- PID Clock
- Reset Right Encoder
- Right Position (Counts)
- Right Velocity (Counts/Interval)
- Right Accel (Counts/Interval^2)
- Right CW Direction

Read the Left Encoder

- PID Clock
- Reset Left Encoder
- Left Position (Counts)
- Left Velocity (Counts/Interval)
- Left Accel (Counts/Interval^2)
- Left CW Direction

This was in the example as part of the motor driver:
- Only used the part necessary to create the clock which is referenced in the encoder block

- Code was adapted from compactrio_motor_control_basics.pdf
Replace this Input Signal boolean with an FPGA I/O Node and select the input pin that you wish to use. This VI measures the high and low periods of a pulse width modulated signal for use on a NI-IMAQ I/O FPGA device.

The High Period is denoted with “H”
The Low Period is denoted with “L”

Read the Encoders
- Inputs: Pin for Ch A & Ch B of rotary encoder
- Black Magic from downloaded FPGA Encoder VI Block
- Outputs: Position, Velocity, and Direction
- Code was adapted from compactrio_motor_control_basics.pdf

- Only used the part necessary to create the clock which is referenced in the encoder block

This was in the example as part of the motor driver.
1. Replace this input signal boolean with an FPGA I/O Node and select the input pin that you wish to use. This VI measures the high and low periods of a pulse width modulated signal for use on a NI-IMAQ I/O FPGA device.

Figure E.1: FPGA code
E.2 Sub VIs

Call values from the FPGA code
- Patient and Robot values separated for ease of reading

Calculate the Angular Velocity
- Range of Gyro is \( \pm 250 \text{ deg/s} \)
  \[ = (\text{DutyCycle}-0.5)\times 500 \]

\[ \text{DutyCycle} = \frac{\text{Time On}}{\text{Total Time}} \]

\[ \text{Deg}_s = \left(\frac{\text{HighPeriod}}{\text{HighPeriod}+\text{LowPeriod}}-0.5\right)\times 500; \]

Converted values are saved to a global variable to be called in higher VIs
- Data is displayed on front panel

Figure E.2: IMU Read code
Figure E.3: Bias Readings code
Calculate the Duty Cycle
Duty Cycle = Time On / Total Time
Angle = DutyCycle*360 Deg

Values called from the FPGA code
- Patient and robot values separated for ease of reading

\[
\text{DutyCycle} = \frac{\text{HighPeriod}}{\text{HighPeriod} + \text{LowPeriod}};
\]
\[
\text{Angle} = (1 - \text{DutyCycle}) \times 360;
\]

\[
\text{Angle} = 1433 \times \text{DutyCycle}^4 - 1874 \times \text{DutyCycle}^3 + 155 \times \text{DutyCycle}^2 - 67 \times \text{DutyCycle} + 360;
\]

DutyCycle
Angle
LowPeriod
HighPeriod

DutyCycle_P
DutyCycle_R
output variable

Robot Mag Last (0-360 deg)
Patient Mag Last (0-360 deg)

Patient Mag Last (0-360 deg)
Robot Mag Last (0-360 deg)

Patient Rotations
Robot Rotations

Patient Heading (Continuous Deg)
Robot Heading (Continuous Deg)

Angle_Raw_P
Angle_Raw_R

-250
1
250
-1

Patient Rotations
Robot Rotations

Magnetic Field Interference Correction

Convert Mag signal to Degrees
Calculate Derivative
Save Current Value as Last Value
Increment Rotations

- If 0 to 360, rotation is negative ... -1
- If 360 to 0, rotation is positive ... +1

Get Heading
- Heading = current degrees + 360*rotations
- We don’t have to deal with the 0-360 jump anymore ... Huzzah!

error in (no error)
Refnum in error out 2
Refnum out

This should be +-250, but CH2, CH3 don’t seem to read as fast. This causes errors.
Calculate the Duty Cycle

Duty Cycle = Time On / Total Time

Angle  =  DutyCycle*360 Deg

Values called from the FPGA code
- Patient and robot values separated for ease of reading

\[
\text{float DutyCycle} = \frac{\text{HighPeriod}}{\text{HighPeriod} + \text{LowPeriod}}; \\
\text{Angle} = (1 - \text{DutyCycle}) \times 360; \\
\text{Angle} = 1433 \times \text{DutyCycle}^4 - 1874 \times \text{DutyCycle}^3 + 155 \times \text{DutyCycle}^2 - 67 \times \text{DutyCycle} + 360;
\]

DutyCycle
Angle
LowPeriod
HighPeriod

Figure E.4: Mag Read code
Complementary Filter

\[ \text{Old_Vel} \]

\[ \text{Vel} \]

\[ \text{Vel}_\text{Out} = \text{Vel}; \]

\[ \text{Accel}_\text{Out} = \text{Vel} - \text{Old_Vel}; \]

\[ \text{Integrate } 0.1 \text{ is time step} \]

\[ 0.97 \]

\[ 1 \]

Patient Filtered Heading (Continuous Deg)

Patient Gyro

Period

Patient Heading Offset

Patient Filtered A. Vel (Deg_s)

Patient Filtered A. Accel (Deg_s^2)

Robot Filtered Heading (Continuous Deg)

Robot Gyro

Robot Filtered A. Vel (Deg_s)

Robot Filtered A. Accel (Deg_s^2)

Robot Heading Offset

Robot Heading (Continuous Deg)
Integrate 0.1 is time step
Complementary Filter
0.97
Vel_Out = Vel;
Accel_Out = Vel - Old_Vel;
1
0.97

Figure E.5: Complementary Filter code
E.3 Component Blocks

Call values from the FPGA code
- Left and Right values are separated for ease of reading

FPGA Values are converted to meaningful data

- Data is also shown on front panel

Converted data is stored in global variables so that it can be called from other VIs.

Encoder Reset

Refnum in (no error)
Converted data is stored in global variables so that it can be called from other VIs.
- Data is also shown on front panel

---

Figure E.6: Encoder Read code
Calculate the Distance
- Range of IR is 20-120 cm
= 56.632*V^-1.023

Call values from the FPGA code
-Patient and Robot values separated for ease of reading

Converted values are filtered and sent to nodes on block

Figure E.7: IR Read code
Figure E.8: Wheel Write code
Figure E.9: Complementary Filter Package code
E.4 Controller

Figure E.10: State Space Controller code
Appendix F

Arduino Code

Listing F.1: Arduino code to convert I2C signals to PWM

```c
/* Sensor2PWM

Sensor2PWM is used to read in I2C signals and convert these to PWM signals
which are easier to read using the Labview FPGA platform.

Adapted by: Andrew Homich
The Pennsylvania State University
ajh5267 at psu dot edu

This code is based on two different example codes:
MPU6050_raw

I2Cdev device library code is placed under the MIT license
Copyright (c) 2011 Jeff Rowberg

Permission is hereby granted, free of charge, to any person obtaining a
copy of this software and associated documentation files (the "Software"),
to deal in the Software without restriction, including without limitation
the rights to use, copy, modify, merge, publish, distribute, sublicense,
and/or sell copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.

MAG3110 Breakout Example Code*/
```

114
The code reads the raw 16-bit x, y, and z values and prints them out. This sketch does not use the INT1 pin, nor does it poll for new data.

```c
// Arduino Wire library is required if I2Cdev I2CDEV ARDUINO_WIRE implementation
#include "Wire.h"

// I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files
// for both classes must be in the include path of your project
#include "I2Cdev.h"
#include "MPU6050.h"

// MPU6050 class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for InvenSense evaluation board)
// AD0 high = 0x69
MPU6050 accelgyro;

// 7-bit address for the MAG3110, doesn’t change
#define MAG_ADDR 0x0E

// Define variables
int16_t ax, ay, az;
int16_t gx, gy, gz;
int16_t mx, my, mz;

#define MAG_PIN 10
#define GYRO_PIN 11
#define MAG_PIN_Y 12
#define ACCEL_PIN 9

#define LED_PIN 13
#define PI 3.14

bool blinkState = false;

// Defined from calibration
#define ROBOT
#endif PATIENT

int mOffX = 730;
int mOffY = 80;
```
int mOffZ = 1650;
#endif

#define ROBOT
int mOffX = 1180;
int mOffY = -1370;
int mOffZ = 1380;
#endif

void setup() {
  // join I2C bus (I2Cdev library doesn’t do this automatically)
  Wire.begin();

  // initialize serial communication
  Serial.begin(9600);

  Serial.println("Initializing I2C devices...");
  // initialize MPU6050 device
  accelgyro.initialize();
  // turn the MAG3110 on
  config();

  // verify connection to MPU6050
  Serial.println("Testing device connections...");
  Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");

  // configure Arduino LED for
  pinMode(LED_PIN, OUTPUT);

  // configure PWM Pins
  pinMode(MAG_PIN, OUTPUT);
  pinMode(GYRO_PIN, OUTPUT);
}

void loop() {
  // read raw accel/gyro measurements from device
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  // read raw mag measurements from device
  mx = readx()+mOffX;
  my = ready()+mOffY;
  mz = readz()+mOffZ;

  // Calculate the angle from north
  #ifdef PATIENT
      float Angle = atan2((float)my,(float)mz);
    #endif

  #ifdef ROBOT
      float Angle = atan2((float)my,(float)mz);
    #endif

    // Convert all values to 0–255
```cpp
int axPWM = ((float)ax + 32768) / 65536 * 255; // (shift to 0−2^16)/2^16*255
int ayPWM = ((float)ay + 32768) / 65536 * 255;
int azPWM = ((float)az + 32768) / 65536 * 255;
int gxPWM = ((float)gx + 32768) / 65536 * 255;
int gyPWM = ((float)gy + 32768) / 65536 * 255;
int gzPWM = ((float)gz + 32768) / 65536 * 255;
int anglePWM = (Angle+PI)/(2*PI)*255; // (shift to 0−2PI)/range*255

// display tab-separated accel/gyro x/y/z values
Serial.print("a/g/m: \\
");
Serial.print(ax); Serial.print(" ");
Serial.print(ay); Serial.print(" ");
Serial.print(az); Serial.print(" ");
Serial.print(gx); Serial.print(" ");
Serial.print(gy); Serial.print(" ");
Serial.print(gz); Serial.print(" ");
Serial.print(mx); Serial.print(" ");
Serial.print(my); Serial.print(" ");
Serial.print(mz); Serial.print(" ");
Serial.println(Angle);

// display tab-separated accel/gyro x/y/z values
Serial.print("a/g/m: \\
");
Serial.print(axPWM); Serial.print(" ");
Serial.print(ayPWM); Serial.print(" ");
Serial.print(azPWM); Serial.print(" ");
Serial.print(gxPWM); Serial.print(" ");
Serial.print(gyPWM); Serial.print(" ");
Serial.print(gzPWM); Serial.print(" ");
Serial.print(anglePWM); Serial.println(" ");

// send out PWM signals
#if defined ROBOT
analogWrite(MAG_PIN, anglePWM);
analogWrite(GYRO_PIN, gyPWM);
#endif
#if defined PATIENT
analogWrite(MAG_PIN, anglePWM);
analogWrite(GYRO_PIN, gyPWM);
#endif

// blink LED to indicate activity
blinkState = !blinkState;
digitalWrite(LED_PIN, blinkState);
```
Listing F.2: Arduino code to read MAG3110

```c
void config(void)
{
  Wire.beginTransmission(MAG_ADDR); // transmit to device 0x0E
  Wire.write(0x11); // ctrl register2
  Wire.write(0x80); // send 0x80, enable auto resets
  Wire.endTransmission(); // stop transmitting
  delay(15);

  Wire.beginTransmission(MAG_ADDR); // transmit to device 0x0E
  Wire.write(0x10); // ctrl register1
  Wire.write(1); // send 0x01, active mode
  Wire.endTransmission(); // stop transmitting
}

int readx(void)
{
  int x1, xh; // define the MSB and LSB
  Wire.beginTransmission(MAG_ADDR); // transmit to device 0x0E
  Wire.write(0x01); // x MSB reg
  Wire.endTransmission(); // stop transmitting
  // needs at least 1.3us free time between start and stop
  delayMicroseconds(2);

  Wire.requestFrom(MAG_ADDR, 1); // request 1 byte
  while (Wire.available()) // slave may send less than requested
  {
    xh = Wire.read(); // receive the byte
  }
  // needs at least 1.3us free time between start and stop
  delayMicroseconds(2);

  Wire.beginTransmission(MAG_ADDR); // transmit to device 0x0E
  Wire.write(0x02); // x LSB reg
  Wire.endTransmission(); // stop transmitting
  // needs at least 1.3us free time between start and stop
  delayMicroseconds(2);

  Wire.requestFrom(MAG_ADDR, 1); // request 1 byte
  while (Wire.available()) // slave may send less than requested
  {
    x1 = Wire.read(); // receive the byte
  }

  int xout = (x1|(xh << 8)); // concatenate the MSB and LSB
  return xout;
}

int ready(void)
{
  int y1, yh; // define the MSB and LSB
```

void Wire::beginTransmission(MAG_ADDR) // transmit to device 0x0E
    
void Wire::write(0x03); // y MSB reg

void Wire::endTransmission() // stop transmitting
    // needs at least 1.3us free time between start and stop
delayMicroseconds(2);

void Wire::requestFrom(MAG_ADDR, 1); // request 1 byte
while (Wire.available()) // slave may send less than requested
    
    yh = Wire.read(); // receive the byte
    
    // needs at least 1.3us free time between start and stop
delayMicroseconds(2);

void Wire::beginTransmission(MAG_ADDR) // transmit to device 0x0E
    
void Wire::write(0x04); // y LSB reg

void Wire::endTransmission() // stop transmitting
    // needs at least 1.3us free time between start and stop
delayMicroseconds(2);

void Wire::requestFrom(MAG_ADDR, 1); // request 1 byte
while (Wire.available()) // slave may send less than requested
    
    yl = Wire.read(); // receive the byte
    
    int yout = (yl | (yh << 8)); // concatenate the MSB and LSB
    return yout;

void readz(void)
    
    int zl, zh; // define the MSB and LSB

void Wire::beginTransmission(MAG_ADDR) // transmit to device 0x0E
    
void Wire::write(0x05); // z MSB reg

void Wire::endTransmission() // stop transmitting
    // needs at least 1.3us free time between start and stop
delayMicroseconds(2);

void Wire::requestFrom(MAG_ADDR, 1); // request 1 byte
while (Wire.available()) // slave may send less than requested
    
    zh = Wire.read(); // receive the byte
    
    // needs at least 1.3us free time between start and stop
delayMicroseconds(2);

void Wire::beginTransmission(MAG_ADDR) // transmit to device 0x0E
    
void Wire::write(0x06); // z LSB reg

void Wire::endTransmission() // stop transmitting
    // needs at least 1.3us free time between start and stop
delayMicroseconds(2);
Wire.requestFrom(MAG_ADDR, 1); // request 1 byte
while(Wire.available()) // slave may send less than requested
{
  zl = Wire.read(); // receive the byte
}

int zout = (zl | (zh << 8)); // concatenate the MSB and LSB
return zout;

int getMaxMin(void)
{
  if(mx > mxM){mxM = mx;}
  if(mx < mxm){mxm = mx;}
  if(my > myM){myM = my;}
  if(my < mym){mym = my;}
  if(mz > mzM){mzM = mz;}
  if(mz < zmz){zmz = mz;}
  Serial.print(mxM); Serial.print("t");
  Serial.print(mxm); Serial.print("t");
  Serial.print(myM); Serial.print("t");
  Serial.print(mym); Serial.print("t");
  Serial.print(mzM); Serial.print("t");
  Serial.println(zmz);
Appendix G

Data Sheets
MPU-6000 and MPU-6050
Product Specification
Revision 3.4
5 Features

5.1 Gyroscope Features
The triple-axis MEMS gyroscope in the MPU-60X0 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec
- External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
- Integrated 16-bit ADCs enable simultaneous sampling of gyro
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.6mA
- Standby current: 5µA
- Factory calibrated sensitivity scale factor
- User self-test

5.2 Accelerometer Features
The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500µA
- Low power accelerometer mode current: 10µA at 1.25Hz, 20µA at 5Hz, 60µA at 20Hz, 110µA at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- High-G interrupt
- User self-test

5.3 Additional Features
The MPU-60X0 includes the following additional features:

- 9-Axis MotionFusion by the on-chip Digital Motion Processor (DMP)
- Auxiliary master I²C bus for reading data from external sensors (e.g., magnetometer)
- 3.9mA operating current when all 6 motion sensing axes and the DMP are enabled
- VDD supply voltage range of 2.375V-3.46V
- Flexible VLOGIC reference voltage supports multiple I²C interface voltages (MPU-6050 only)
- Smallest and thinnest QFN package for portable devices: 4x4x0.9mm
- Minimal cross-axis sensitivity between the accelerometer and gyroscope axes
- 1024 byte FIFO buffer reduces power consumption by allowing host processor to read the data in bursts and then go into a low-power mode as the MPU collects more data
- Digital-output temperature sensor
- User-programmable digital filters for gyroscope, accelerometer, and temp sensor
- 10,000 g shock tolerant
- 400kHz Fast Mode I²C for communicating with all registers
- 1MHz SPI serial interface for communicating with all registers (MPU-6000 only)
- 20MHz SPI serial interface for reading sensor and interrupt registers (MPU-6000 only)
• MEMS structure hermetically sealed and bonded at wafer level
• RoHS and Green compliant

5.4 MotionProcessing
• Internal Digital Motion Processing™ (DMP™) engine supports 3D MotionProcessing and gesture recognition algorithms
• The MPU-60X0 collects gyroscope and accelerometer data while synchronizing data sampling at a user defined rate. The total dataset obtained by the MPU-60X0 includes 3-Axis gyroscope data, 3-Axis accelerometer data, and temperature data. The MPU’s calculated output to the system processor can also include heading data from a digital 3-axis third party magnetometer.
• The FIFO buffers the complete data set, reducing timing requirements on the system processor by allowing the processor burst read the FIFO data. After burst reading the FIFO data, the system processor can save power by entering a low-power sleep mode while the MPU collects more data.
• Programmable interrupt supports features such as gesture recognition, panning, zooming, scrolling, tap detection, and shake detection
• Digitally-programmable low-pass filters
• Low-power pedometer functionality allows the host processor to sleep while the DMP maintains the step count.

5.5 Clocking
• On-chip timing generator ±1% frequency variation over full temperature range
• Optional external clock inputs of 32.768kHz or 19.2MHz
6 Electrical Characteristics

6.1 Gyroscope Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>CONDITIONS</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
<th>UNITS</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>GYROSCOPE SENSITIVITY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full-Scale Range</td>
<td>FS_SEL=0</td>
<td>±250</td>
<td></td>
<td></td>
<td>°/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FS_SEL=1</td>
<td>±500</td>
<td></td>
<td></td>
<td>°/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FS_SEL=2</td>
<td>±1000</td>
<td></td>
<td></td>
<td>°/s</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FS_SEL=3</td>
<td>±2000</td>
<td></td>
<td></td>
<td>°/s</td>
<td></td>
</tr>
<tr>
<td>Gyroscope ADC Word Length</td>
<td></td>
<td>16</td>
<td></td>
<td></td>
<td>bits</td>
<td></td>
</tr>
<tr>
<td>Sensitivity Scale Factor</td>
<td>FS_SEL=0</td>
<td>131</td>
<td></td>
<td></td>
<td>LSB/(°/s)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FS_SEL=1</td>
<td>65.5</td>
<td></td>
<td></td>
<td>LSB/(°/s)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FS_SEL=2</td>
<td>32.8</td>
<td></td>
<td></td>
<td>LSB/(°/s)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FS_SEL=3</td>
<td>16.4</td>
<td></td>
<td></td>
<td>LSB/(°/s)</td>
<td></td>
</tr>
<tr>
<td>Sensitivity Scale Factor Tolerance</td>
<td>25°C</td>
<td>-3</td>
<td></td>
<td>+3</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>Sensitivity Scale Factor Variation Over</td>
<td></td>
<td>±2</td>
<td></td>
<td></td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nonlinearity</td>
<td></td>
<td>0.2</td>
<td></td>
<td></td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>Cross-Axis Sensitivity</td>
<td></td>
<td>±2</td>
<td></td>
<td></td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>GYROSCOPE ZERO-RATE OUTPUT (ZRO)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial ZRO Tolerance</td>
<td></td>
<td>±20</td>
<td></td>
<td></td>
<td>°/s</td>
<td></td>
</tr>
<tr>
<td>ZRO Variation Over Temperature</td>
<td>-40°C to +85°C</td>
<td>±20</td>
<td></td>
<td></td>
<td>°/s</td>
<td></td>
</tr>
<tr>
<td>Power-Supply Sensitivity (1-10Hz)</td>
<td>Sine wave, 100mVpp; VDD=2.5V</td>
<td>0.2</td>
<td></td>
<td></td>
<td>°/s</td>
<td></td>
</tr>
<tr>
<td>Power-Supply Sensitivity (10 - 250Hz)</td>
<td>Sine wave, 100mVpp; VDD=2.5V</td>
<td>0.2</td>
<td></td>
<td></td>
<td>°/s</td>
<td></td>
</tr>
<tr>
<td>Power-Supply Sensitivity (250Hz - 100kHz)</td>
<td>Sine wave, 100mVpp; VDD=2.5V</td>
<td>4</td>
<td></td>
<td></td>
<td>°/s</td>
<td></td>
</tr>
<tr>
<td>Linear Acceleration Sensitivity</td>
<td>Static</td>
<td>0.1</td>
<td></td>
<td></td>
<td>°/s/g</td>
<td></td>
</tr>
<tr>
<td>SELF-TEST RESPONSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>Relative</td>
<td></td>
<td>-14</td>
<td></td>
<td>14</td>
<td>%</td>
<td>1</td>
</tr>
<tr>
<td>GYROSCOPE NOISE PERFORMANCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total RMS Noise</td>
<td>FS_SEL=0</td>
<td>0.05</td>
<td></td>
<td></td>
<td>°/s-rms</td>
<td></td>
</tr>
<tr>
<td>Low-frequency RMS noise</td>
<td>DLPFCFG=2 (100Hz)</td>
<td>0.033</td>
<td></td>
<td></td>
<td>°/s-rms</td>
<td></td>
</tr>
<tr>
<td>Rate Noise Spectral Density</td>
<td>Bandwidth 1Hz to10Hz</td>
<td>0.005</td>
<td></td>
<td></td>
<td>°/s/√Hz</td>
<td></td>
</tr>
<tr>
<td>GYROSCOPE MECHANICAL FREQUENCIES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X-Axis</td>
<td></td>
<td>30</td>
<td></td>
<td>33</td>
<td>36</td>
<td>kHz</td>
</tr>
<tr>
<td>Y-Axis</td>
<td></td>
<td>27</td>
<td></td>
<td>30</td>
<td>33</td>
<td>kHz</td>
</tr>
<tr>
<td>Z-Axis</td>
<td></td>
<td>24</td>
<td></td>
<td>27</td>
<td>30</td>
<td>kHz</td>
</tr>
<tr>
<td>LOW PASS FILTER RESPONSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmable Range</td>
<td></td>
<td>5</td>
<td></td>
<td>256</td>
<td>Hz</td>
<td></td>
</tr>
<tr>
<td>OUTPUT DATA RATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmable</td>
<td></td>
<td>4</td>
<td></td>
<td>8,000</td>
<td>Hz</td>
<td></td>
</tr>
<tr>
<td>GYROSCOPE START-UP TIME</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZRO Settling (from power-on)</td>
<td>DLPFCFG=0</td>
<td>30</td>
<td></td>
<td></td>
<td>ms</td>
<td></td>
</tr>
</tbody>
</table>

1. Please refer to the following document for further information on Self-Test: MPU-6000/MPU-6050 Register Map and Descriptions
6.2 Accelerometer Specifications
VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, $T_A = 25°C$

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>CONDITIONS</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
<th>UNITS</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCELEROMETER SENSITIVITY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full-Scale Range</td>
<td>AFS_SEL=0</td>
<td>±2</td>
<td></td>
<td></td>
<td>g</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AFS_SEL=1</td>
<td>±4</td>
<td></td>
<td></td>
<td>g</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AFS_SEL=2</td>
<td>±8</td>
<td></td>
<td></td>
<td>g</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AFS_SEL=3</td>
<td>±16</td>
<td></td>
<td></td>
<td>g</td>
<td></td>
</tr>
<tr>
<td>ADC Word Length</td>
<td>Output in two’s complement format</td>
<td>16</td>
<td></td>
<td></td>
<td>bits</td>
<td></td>
</tr>
<tr>
<td>Sensitivity Scale Factor</td>
<td>AFS_SEL=0</td>
<td>16,384</td>
<td></td>
<td></td>
<td>LSB/g</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AFS_SEL=1</td>
<td>8,192</td>
<td></td>
<td></td>
<td>LSB/g</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AFS_SEL=2</td>
<td>4,096</td>
<td></td>
<td></td>
<td>LSB/g</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AFS_SEL=3</td>
<td>2,048</td>
<td></td>
<td></td>
<td>LSB/g</td>
<td></td>
</tr>
<tr>
<td>Initial Calibration Tolerance</td>
<td></td>
<td>±3</td>
<td></td>
<td></td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>Sensitivity Change vs. Temperature</td>
<td>AFS_SEL=0, -40°C to +85°C</td>
<td>±0.02</td>
<td></td>
<td></td>
<td>%/°C</td>
<td></td>
</tr>
<tr>
<td>Nonlinearity</td>
<td>Best Fit Straight Line</td>
<td>0.5</td>
<td></td>
<td></td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>Cross-Axis Sensitivity</td>
<td></td>
<td>±2</td>
<td></td>
<td></td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>ZERO-G OUTPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial Calibration Tolerance</td>
<td>X and Y axes</td>
<td>±50</td>
<td></td>
<td></td>
<td>mg</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Z axis</td>
<td>±80</td>
<td></td>
<td></td>
<td>mg</td>
<td></td>
</tr>
<tr>
<td>Zero-G Level Change vs. Temperature</td>
<td>X and Y axes, 0°C to +70°C</td>
<td>±35</td>
<td></td>
<td></td>
<td>mg</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Z axis, 0°C to +70°C</td>
<td>±60</td>
<td></td>
<td></td>
<td>mg</td>
<td></td>
</tr>
<tr>
<td>SELF TEST RESPONSE</td>
<td>Change from factory trim</td>
<td>-14</td>
<td></td>
<td></td>
<td>%</td>
<td>2</td>
</tr>
<tr>
<td>NOISE PERFORMANCE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Spectral Density</td>
<td>@10Hz, AFS_SEL=0 &amp; ODR=1kHz</td>
<td>400</td>
<td></td>
<td></td>
<td>μg/√Hz</td>
<td></td>
</tr>
<tr>
<td>LOW PASS FILTER RESPONSE</td>
<td>Programmable Range</td>
<td>5</td>
<td></td>
<td></td>
<td>Hz</td>
<td></td>
</tr>
<tr>
<td>OUTPUT DATA RATE</td>
<td>Programmable Range</td>
<td>4</td>
<td></td>
<td>1,000</td>
<td>Hz</td>
<td></td>
</tr>
<tr>
<td>INTELLIGENCE FUNCTION INCREMENT</td>
<td></td>
<td>32</td>
<td></td>
<td></td>
<td>mg/LSB</td>
<td></td>
</tr>
</tbody>
</table>

1. Typical zero-g initial calibration tolerance value after MSL3 preconditioning
2. Please refer to the following document for further information on Self-Test: MPU-6000/MPU-6050 Register Map and Descriptions
Xtrinsic MAG3110 Three-Axis, Digital Magnetometer

Freescale’s MAG3110 is a small, low-power, digital 3-axis magnetometer.

The device can be used in conjunction with a 3-axis accelerometer to realize an orientation independent electronic compass that can provide accurate heading information. It features a standard I2C serial interface output and smart embedded functions.

The MAG3110 is capable of measuring magnetic fields with an output data rate (ODR) up to 80 Hz; these output data rates correspond to sample intervals from 12.5 ms to several seconds.

The MAG3110 is available in a plastic DFN package and it is guaranteed to operate over the extended temperature range of -40°C to +85°C.

Features
- 1.95 V to 3.6 V supply voltage (VDD)
- 1.62 V to VDD IO voltage (VDDIO)
- Ultra small 2 mm x 2 mm x 0.85 mm, 0.4 mm pitch, 10-pin package
- Full-scale range ±1000 µT
- Sensitivity of 0.10 µT
- Noise down to 0.25 µT rms
- Output Data Rates (ODR) up to 80 Hz
- 400 kHz Fast Mode compatible I2C interface
- Low-power, single-shot measurement mode
- RoHS compliant

Applications
- Electronic Compass (e-compass)
- Location-Based Services

Ruggedized Target markets
- Smartphones, personal navigation devices, robotics, UAVs, speed sensing, current sensing and wrist watches with embedded electronic compasses (e-compass) function.

Table 1. Ordering information

<table>
<thead>
<tr>
<th>Part number</th>
<th>I2C Address</th>
<th>Package description</th>
<th>Shipping</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAG3110FCR1</td>
<td>0xE</td>
<td>DFN-10</td>
<td>Tape and Reel (1000)</td>
</tr>
<tr>
<td>FXMS3110CDR1</td>
<td>0xF</td>
<td>DFN-10</td>
<td>Tape and Reel (1000)</td>
</tr>
</tbody>
</table>
## 2 Operating and Electrical Specifications

### 2.1 Operating characteristics

Table 3. Operating characteristics @ VDD = 2.4 V, VDDIO = 1.8 V, T = 25°C unless otherwise noted.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Test Conditions</th>
<th>Symbol</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-scale range</td>
<td></td>
<td>FS</td>
<td>±1000</td>
<td></td>
<td></td>
<td>µT</td>
</tr>
<tr>
<td>Output data range(^1)</td>
<td></td>
<td>-30000</td>
<td>+30000</td>
<td>LSB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensitivity</td>
<td></td>
<td>So</td>
<td>0.10</td>
<td></td>
<td></td>
<td>µT/LSB</td>
</tr>
<tr>
<td>Sensitivity change versus temp</td>
<td></td>
<td>Tcs</td>
<td>±0.1</td>
<td></td>
<td></td>
<td>%/°C</td>
</tr>
<tr>
<td>Zero-flux offset accuracy</td>
<td></td>
<td></td>
<td>±100</td>
<td></td>
<td></td>
<td>µT</td>
</tr>
<tr>
<td>Hysteresis(^2)</td>
<td></td>
<td></td>
<td>0.25</td>
<td>1</td>
<td></td>
<td>%</td>
</tr>
<tr>
<td>Non linearity Best fit straight line(^3)</td>
<td></td>
<td>NL</td>
<td>-1</td>
<td>±0.3</td>
<td>1</td>
<td>%FS</td>
</tr>
<tr>
<td>Magnetometer output noise</td>
<td>OS = 00(^4)</td>
<td>Noise</td>
<td>0.4</td>
<td></td>
<td></td>
<td>µT rms</td>
</tr>
<tr>
<td></td>
<td>OS = 01</td>
<td></td>
<td>0.35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OS = 10</td>
<td></td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OS = 11</td>
<td></td>
<td>0.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor die-to-package misalignemt error (yaw)</td>
<td>D2PE(_{yaw})</td>
<td>±0.36</td>
<td>±1.36</td>
<td></td>
<td>degrees</td>
<td></td>
</tr>
<tr>
<td>Operating temperature range</td>
<td>T(_{op})</td>
<td>±40</td>
<td></td>
<td></td>
<td>+85</td>
<td>°C</td>
</tr>
</tbody>
</table>

1. Output data range is the sum of ±10000 LSBs full-scale range, ±10000 LSBs user defined offset (provided that CTRL_REG2[RAW] = 0) and ±10000 zero-flux offset.
2. Hysteresis is measured from 0 µT to 1000 µT to 0 µT and from 0 µT to -1000 µT to 0 µT.
3. Best-fit straight line over the 0 to ±1000 µT full-scale range.
4. OS = Over Sampling Ratio.
2.2 Absolute maximum ratings

Stresses above those listed as "absolute maximum ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device under these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Table 4. Maximum ratings

<table>
<thead>
<tr>
<th>Rating</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage</td>
<td>VDD</td>
<td>-0.3 to +3.6</td>
<td>V</td>
</tr>
<tr>
<td>Input voltage on any control pin (SCL, SDA)</td>
<td>Vin</td>
<td>-0.3 to VDDIO + 0.3</td>
<td>V</td>
</tr>
<tr>
<td>Maximum applied magnetic/field</td>
<td>B MAX</td>
<td>100,000</td>
<td>µT</td>
</tr>
<tr>
<td>Operating temperature range</td>
<td>T op</td>
<td>-40 to +85</td>
<td>°C</td>
</tr>
<tr>
<td>Storage temperature range</td>
<td>T STG</td>
<td>-40 to +125</td>
<td>°C</td>
</tr>
</tbody>
</table>

Table 5. ESD and latchup protection characteristics

<table>
<thead>
<tr>
<th>Rating</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Body Model</td>
<td>HBM</td>
<td>±2000</td>
<td>V</td>
</tr>
<tr>
<td>Machine Model</td>
<td>MM</td>
<td>±200</td>
<td>V</td>
</tr>
<tr>
<td>Charge Device Model</td>
<td>CDM</td>
<td>±500</td>
<td>V</td>
</tr>
<tr>
<td>Latchup current at T = 85°C</td>
<td>I L,U</td>
<td>±100</td>
<td>mA</td>
</tr>
</tbody>
</table>

This device is sensitive to mechanical shock. Improper handling can cause permanent damage of the part or cause the part to otherwise fail.

This device is sensitive to ESD, improper handling can cause permanent damage to the part.
GP2Y0A02YK0F

Distance Measuring Sensor Unit
Measuring distance: 20 to 150 cm
Analog output type

Description
GP2Y0A02YK0F is a distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector), IRED (infrared emitting diode) and signal processing circuit. The variety of the reflectivity of the object, the environmental temperature and the operating duration are not influenced easily to the distance detection because of adopting the triangulation method. This device outputs the voltage corresponding to the detection distance. So this sensor can also be used as a proximity sensor.

Features
1. Distance measuring range: 20 to 150 cm
2. Analog output type
3. Package size: 29.5x13x21.6 mm
4. Consumption current:Typ. 33 mA
5. Supply voltage:4.5 to 5.5 V

Agency approvals/Compliance
1. Compliant with RoHS directive (2002/95/EC)

Applications
1. Touch-less switch
   (Sanitary equipment, Control of illumination, etc.)
2. Sensor for energy saving
   (ATM, Copier, Vending machine, Laptop computer, LCD monitor)
3. Amusement equipment
   (Robot, Arcade game machine)
### Block diagram

![Diagram of distance measuring IC](image)

- **Signal processing circuit**
- **Voltage regulator**
- **Oscillation circuit**
- **Output circuit**
- **LED drive circuit**

### Outline Dimensions

**(Unit : mm)**

- **Product mass**: approx. 4.8g

![Dimensions diagram](image)

- **Light detector side**
- **Light emitter side**

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Output terminal voltage</td>
<td>$V_O$</td>
</tr>
<tr>
<td>2. Ground</td>
<td>GND</td>
</tr>
<tr>
<td>3. Supply voltage</td>
<td>$V_{CC}$</td>
</tr>
</tbody>
</table>

**Stamp (Example)**

- **Model name**: SHARP GP2Y0A02YK0F
- **Production month**: Jan. to Sep.: 1 to 9
- **Oct.**: X, Nov.: Y, Dec.: Z
- **Production year**: Last digit of prod. year

**Notes**

1. Unspecified tolerances shall be ± 0.3 mm.
2. The connector is made by J.S.T.TRADING COMPANY,LTD. and its part number is S3B-PH.
3. The dimensions in parenthesis are shown for reference.
4. The dimension marked by "*" show a distance from/to the center of an internal optical slit.

*Product mass: approx. 4.8g*
### Absolute Maximum Ratings

(T<sub>a</sub>=25°C, V<sub>CC</sub>=5V)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Rating</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage</td>
<td>V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>-0.3 to +7 V</td>
<td></td>
</tr>
<tr>
<td>Output terminal voltage</td>
<td>V&lt;sub&gt;O&lt;/sub&gt;</td>
<td>-0.3 to V&lt;sub&gt;CC&lt;/sub&gt;+0.3 V</td>
<td></td>
</tr>
<tr>
<td>Operating temperature</td>
<td>T&lt;sub&gt;opr&lt;/sub&gt;</td>
<td>-10 to +60°C</td>
<td></td>
</tr>
<tr>
<td>Storage temperature</td>
<td>T&lt;sub&gt;stg&lt;/sub&gt;</td>
<td>-40 to +70°C</td>
<td></td>
</tr>
</tbody>
</table>

### Electro-optical Characteristics

(T<sub>a</sub>=25°C, V<sub>CC</sub>=5V)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>MIN.</th>
<th>TYP.</th>
<th>MAX.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average supply current</td>
<td>I&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>L=150cm (Note 1)</td>
<td>33</td>
<td>50</td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>Measuring distance range</td>
<td>ΔL</td>
<td>(Note 1)</td>
<td>20</td>
<td>150</td>
<td></td>
<td>cm</td>
</tr>
<tr>
<td>Output voltage</td>
<td>V&lt;sub&gt;O&lt;/sub&gt;</td>
<td>L=150cm (Note 1)</td>
<td>0.25</td>
<td>0.4</td>
<td>0.55</td>
<td>V</td>
</tr>
<tr>
<td>Output voltage differential</td>
<td>ΔV&lt;sub&gt;O&lt;/sub&gt;</td>
<td>Output voltage difference between L=20cm and L=150cm (Note 1)</td>
<td>1.8</td>
<td>2.05</td>
<td>2.3</td>
<td>V</td>
</tr>
</tbody>
</table>

* L : Distance to reflective object

Note 1 : Using reflective object : White paper (Made by Kodak Co., Ltd. gray cards R-27 white face, reflectance; 90%)

### Recommended operating conditions

(T<sub>a</sub>=25°C, V<sub>CC</sub>=5V)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>Rating</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage</td>
<td>V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>4.5 to 5.5 V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Description

The E4P miniature reflective encoder is designed to provide digital quadrature encoder feedback for high volume applications with limited space constraints. The E4P version utilizes an innovative, patented push-on codewheel which accepts shaft diameters of 2mm to .250”.

The E4P reflective encoder is the leader for high quantity OEM applications, but the E4 is the ideal choice when a set-screw codewheel encoder is required (see the E4 page).

The E4P miniature encoder base provides mounting holes for two #3-48, length 1/4” or two M2.5x.45mm, length 6mm screws on a .586” bolt circle. When mounting holes are not available, a pre-applied transfer adhesive (with peel-off backing) is available for “stick-on” mounting.

The encoder cover is easily snapped onto the base and is embossed with the connector pin-out.

The E4P reflective series encoder can be connected by using a (high retention 4-conductor snap-in polarized 1.25mm pitch) connector. Mating cables and connectors (see the Cables / Connectors web page) are not included, and are available separately.

Features

- Miniature size
- Push-on hub - spring loaded collet design
- Minimum shaft length of .375”
- Fits shaft diameters of .079” to .250”
- Accepts +/- .020” Axial shaft play
- Off-axis mounting tolerance of .010”
- 100 to 360 cycles per revolution (CPR)
- 400 to 1440 pulses per revolution (PPR)
- Single +5V supply
## Environmental

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vibration (5Hz to 2kHz)</td>
<td>20</td>
<td>G</td>
</tr>
<tr>
<td>Max. Relative Humidity</td>
<td>90</td>
<td>%</td>
</tr>
<tr>
<td>Storage Temperature</td>
<td>-40 to 100</td>
<td>°C</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>-20 to 100</td>
<td>°C</td>
</tr>
<tr>
<td>Electrostatic Discharge, Human Body Model</td>
<td>± 8</td>
<td>kV</td>
</tr>
<tr>
<td>Single-ended (S-option)</td>
<td>± 15</td>
<td></td>
</tr>
<tr>
<td>Differential (D-option)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Mechanical

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. Shaft Axial Play</td>
<td>± .020</td>
<td>in.</td>
</tr>
<tr>
<td>Max. Off-axis Mounting Tolerance</td>
<td>± .010</td>
<td>in.</td>
</tr>
<tr>
<td>Shaft to Mounting Surface Perpendicularity</td>
<td>90 ± 1</td>
<td>deg.</td>
</tr>
<tr>
<td>Max. Acceleration</td>
<td>250000</td>
<td>rad/sec²</td>
</tr>
<tr>
<td>Mounting Screw Size</td>
<td>#3-48 x 1/4&quot;</td>
<td></td>
</tr>
<tr>
<td>Metric (M-option base)</td>
<td>M2.5x.45mm, length 6mm</td>
<td></td>
</tr>
<tr>
<td>Screw Bolt Circle Diameter</td>
<td>.586 ± .002</td>
<td>in.</td>
</tr>
<tr>
<td>Axial Length of Codewheel</td>
<td>.270</td>
<td>in.</td>
</tr>
<tr>
<td>Required Shaft Length (1)</td>
<td>.375 to .395</td>
<td>in.</td>
</tr>
<tr>
<td>Mounting Screw Torque</td>
<td>2-3</td>
<td>in-lbs</td>
</tr>
</tbody>
</table>

Technical Bulletin TB1001 - Shaft and Bore Tolerances [Download](#)

(1) Includes axial play.

## Single-ended Electrical

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Units</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
<td>4.5</td>
<td>5.0</td>
<td>5.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Supply Current</td>
<td>21</td>
<td>27</td>
<td>mA</td>
<td>no load</td>
<td></td>
</tr>
<tr>
<td>Low-level Output</td>
<td>0.4</td>
<td>V</td>
<td>IOL = 6 mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High-level Output</td>
<td>2.4</td>
<td>V</td>
<td>IOH = -1 mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rise Time</td>
<td>500</td>
<td>ns</td>
<td>CL = 25 pF, RL = 2.7 kΩ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Specifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Units</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall Time</td>
<td></td>
<td>100</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### Differential Electrical

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Units</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
<td>4.5</td>
<td>5.0</td>
<td>5.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Supply Current</td>
<td>23</td>
<td>29</td>
<td></td>
<td>mA</td>
<td>no load</td>
</tr>
<tr>
<td>Single-Ended Output Voltage High</td>
<td></td>
<td>5.0</td>
<td></td>
<td>V</td>
<td>One TTL load</td>
</tr>
<tr>
<td>Single-Ended Output Voltage Low</td>
<td></td>
<td>0.4</td>
<td></td>
<td>V</td>
<td>One TTL load</td>
</tr>
<tr>
<td>Differential Output Voltage</td>
<td>3.0</td>
<td>3.8</td>
<td></td>
<td>V</td>
<td>RL = 100 ohm</td>
</tr>
<tr>
<td>Differential Output Rise/Fall Time</td>
<td></td>
<td></td>
<td>20</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

### Phase Relationship

- **Parameter**: Symmetry, S
  - **Typ.**: 180 ± 16
  - **Max.**: 180 ± 75
  - **Units**: electrical degrees

- **Parameter**: Quadrature Delay, Q
  - **Typ.**: 90 ± 10
  - **Max.**: 90 ± 60
  - **Units**: electrical degrees

(1) A leads B for clockwise shaft rotation, B leads A for counter clockwise shaft rotation viewed from the cover/label side of the encoder.

(2) Typical values represent the encoder performance at typical mounting alignment, whereas the maximum values represent the encoder performance across the range of recommended mounting tolerance.

### Pin-out

<table>
<thead>
<tr>
<th>4-pin Single-ended (1)</th>
<th>6-pin Differential (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin</td>
<td>Description</td>
</tr>
<tr>
<td>1</td>
<td>+5VDC power</td>
</tr>
<tr>
<td>2</td>
<td>A channel</td>
</tr>
<tr>
<td>3</td>
<td>Ground</td>
</tr>
</tbody>
</table>
(1) 4-pin single-ended mating connector is CON-MIC4  
(2) 6-pin differential mating connector is CON-MIC6

**Options**

**H-option (Hole In Cover)**

The H-option adds a hole in the cover for the shaft to pass through:

- For shaft diameters of 1.5mm to 1/8", a 0.170" hole is supplied.
- For shaft diameters of 5/32" to 1/4", a 0.295" hole is supplied.

**M-option (Metric Mounting Screws)**

Provides alternate metric M2.5x.45mm, length 6mm screws. When M-option is NOT specified the default is #3-48 x 1/4" screws.

**T-option (Transfer Adhesive)**

When mounting holes are not available, a pre-applied transfer adhesive (with peel-off backing) is available for "stick-on" mounting. Use the centering tool (above) to position the base. T-option specifies transfer adhesive.

Before installation, cleaning the mounting surface with alcohol is recommended to remove dust and oil.

**Accessories**

1. **Centering Tool**

   The centering tool is only included with the -3 packaging option. It has to be ordered separately for other package options.

   **Part #: MCTOOL - (Shaft Diameter)**  
   **Description:** This reusable tool provides a simple method for accurately centering the E4P base onto the shaft. A centering tool is highly recommended when using the T-option transfer adhesive.

2. **Spacer Tool**

   A spacer tool is included for all packaging options.

   **Part #: SPACER-E4P**  
   **Description:** This reusable tool is used to properly space the codewheel from the encoder.

**Assembly Instructions**

E4P Assembly Instructions - http://usdigital.com/assets/assembly/E4P%20Assembly%20Instructions.pdf
The 25A8 PWM servo drive is designed to drive brush type DC motors at a high switching frequency. A single red/green LED indicates operating status. The drive is fully protected against over-voltage, under voltage, over-current, over-heating and short-circuits across motor, ground and power leads. Furthermore, the drive can interface with digital controllers or be used stand-alone and requires only a single unregulated DC power supply. Loop gain, current limit, input gain and offset can be adjusted using 14-turn potentiometers. The offset adjusting potentiometer can also be used as an on-board input signal for testing purposes.

See Part Numbering Information on last page of datasheet for additional ordering options.
Information on Approvals and Compliances

- US and Canadian safety compliance with UL 508c, the industrial standard for power conversion electronics. UL registered under file number E140173. Note that machine components compliant with UL are considered UL registered as opposed to UL listed as would be the case for commercial products.

- Compliant with European CE for both the Class A EMC Directive 2004/108/EC on Electromagnetic Compatibility (specifically EN 61000-6-4:2007 and EN 61000-6-2:2005) and LVD requirements of directive 2006/95/EC (specifically EN 60204-1:2006), a low voltage directive to protect users from electrical shock.

- RoHS (Reduction of Hazardous Substances) is intended to prevent hazardous substances such as lead from being manufactured in electrical and electronic equipment.
## SPECIFICATIONS

### Power Specifications

<table>
<thead>
<tr>
<th>Description</th>
<th>Units</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC Supply Voltage Range</td>
<td>VDC</td>
<td>20 - 80</td>
</tr>
<tr>
<td>DC Bus Over Voltage Limit</td>
<td>VDC</td>
<td>86</td>
</tr>
<tr>
<td>Maximum Peak Output Current</td>
<td>A</td>
<td>25</td>
</tr>
<tr>
<td>Maximum Continuous Output Current</td>
<td>A</td>
<td>12.5</td>
</tr>
<tr>
<td>Maximum Continuous Output Power</td>
<td>W</td>
<td>950</td>
</tr>
<tr>
<td>Maximum Power Dissipation at Continuous Current</td>
<td>W</td>
<td>50</td>
</tr>
<tr>
<td>Minimum Load Inductance (Line-To-Line)</td>
<td>µH</td>
<td>200</td>
</tr>
<tr>
<td>Low Voltage Supply Outputs</td>
<td>VDC</td>
<td>±5</td>
</tr>
<tr>
<td>Switching Frequency</td>
<td>kHz</td>
<td>22</td>
</tr>
</tbody>
</table>

### Control Specifications

<table>
<thead>
<tr>
<th>Description</th>
<th>Units</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Sources</td>
<td>±10 V</td>
<td>±10 V</td>
</tr>
<tr>
<td>Feedback Supported</td>
<td>VDC</td>
<td>Position, Tachometer (±60 VDC)</td>
</tr>
<tr>
<td>Commutation Methods</td>
<td></td>
<td>Brush Type</td>
</tr>
<tr>
<td>Modes of Operation</td>
<td></td>
<td>Current, IR Compensation, Velocity, Voltage</td>
</tr>
<tr>
<td>Motors Supported</td>
<td></td>
<td>Single Phase (Brushed, Voice Coil, Inductive Load)</td>
</tr>
<tr>
<td>Hardware Protection</td>
<td></td>
<td>Over Current, Over Temperature, Over Voltage, Short Circuit (Phase-Phase &amp; Phase-Ground)</td>
</tr>
<tr>
<td>Primary I/O Logic Level</td>
<td></td>
<td>5V TTL</td>
</tr>
</tbody>
</table>

### Mechanical Specifications

<table>
<thead>
<tr>
<th>Description</th>
<th>Units</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agency Approvals</td>
<td></td>
<td>CE Class A (EMC), CE Class A (LVD), cUL, RoHS, UL</td>
</tr>
<tr>
<td>Size (H x W x D)</td>
<td>mm (in)</td>
<td>129.3 x 75.8 x 25.1 (5.1 x 3 x 1)</td>
</tr>
<tr>
<td>Weight</td>
<td>g (oz)</td>
<td>280 (9.9)</td>
</tr>
<tr>
<td>Heatsink (Base) Temperature Range</td>
<td>°C (°F)</td>
<td>0 - 65 (32 - 149)</td>
</tr>
<tr>
<td>Storage Temperature Range</td>
<td>°C (°F)</td>
<td>-40 - 85 (-40 - 185)</td>
</tr>
<tr>
<td>Form Factor</td>
<td></td>
<td>Panel Mount</td>
</tr>
<tr>
<td>P1 Connector</td>
<td></td>
<td>16-pin, 2.54 mm spaced, friction lock header</td>
</tr>
<tr>
<td>P2 Connector</td>
<td></td>
<td>5-port, 5.08 mm spaced, screw terminal</td>
</tr>
</tbody>
</table>

### Notes

1. Maximum duration of peak current is ~2 seconds. Peak RMS value must not exceed continuous current rating of the drive.
2. Lower inductance is acceptable for bus voltages well below maximum. Use external inductance to meet requirements.
3. Additional cooling and/or heatsink may be required to achieve rated performance.
### PIN FUNCTIONS

#### P1 - Signal Connector

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description / Notes</th>
<th>I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5V 3mA OUT</td>
<td>±5 V @ 3 mA low power supply for customer use. Short circuit protected. Reference ground common with signal ground.</td>
<td>O</td>
</tr>
<tr>
<td>2</td>
<td>SIGNAL GND</td>
<td>Differential Reference Input (±10 V Operating Range, ±15 V Maximum Input)</td>
<td>GND</td>
</tr>
<tr>
<td>3</td>
<td>-5V 3mA OUT</td>
<td>Negative Tachometer Input (Maximum ±60 V). Use signal ground for positive input.</td>
<td>O</td>
</tr>
<tr>
<td>4</td>
<td>+REF IN</td>
<td>Positive Tachometer Input and Signal Ground</td>
<td>I</td>
</tr>
<tr>
<td>5</td>
<td>-REF IN</td>
<td>Current Monitor. Analog output signal proportional to the actual current output.</td>
<td>I</td>
</tr>
<tr>
<td>6</td>
<td>+TACH / GND</td>
<td>Measures the command signal to the internal current-loop. This pin has a maximum output of ±7.25 V when the drive outputs maximum peak current. Measure relative to signal ground.</td>
<td>O</td>
</tr>
<tr>
<td>7</td>
<td>CONT CURRENT LIMIT</td>
<td>Can be used to reduce the factory-preset maximum continuous current limit without affecting the peak current limit by attaching an external current limiting resistor between this pin and signal ground. See pin details for resistor values.</td>
<td>I</td>
</tr>
<tr>
<td>8</td>
<td>INHIBIT IN</td>
<td>TTL level (+5 V) inhibit/enable input. Leave open to enable drive. Pull to ground to inhibit drive. Inhibit turns off all power devices.</td>
<td>I</td>
</tr>
<tr>
<td>9</td>
<td>FAULT OUT</td>
<td>TTL level (+5 V) output becomes high when power devices are disabled due to at least one of the following conditions: inhibit, output short circuit, over voltage, over temperature, power-up reset.</td>
<td>O</td>
</tr>
<tr>
<td>10</td>
<td>INHIBIT IN</td>
<td>Positive Direction Inhibit (Does Not Cause A Fault Condition)</td>
<td>I</td>
</tr>
<tr>
<td>11</td>
<td>NC</td>
<td>Not Connected (Reserved)</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>NC</td>
<td>Not Connected (Reserved)</td>
<td>-</td>
</tr>
</tbody>
</table>

#### P2 - Power Connector

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description / Notes</th>
<th>I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-MOT</td>
<td>Negative Motor Output</td>
<td>O</td>
</tr>
<tr>
<td>2</td>
<td>+MOT</td>
<td>Positive Motor Output</td>
<td>O</td>
</tr>
<tr>
<td>3</td>
<td>POWER GND</td>
<td>Power Ground (Common With Signal Ground)</td>
<td>PGND</td>
</tr>
<tr>
<td>4</td>
<td>POWER GND</td>
<td>DC Power Input</td>
<td>PGND</td>
</tr>
<tr>
<td>5</td>
<td>HIGH VOLTAGE</td>
<td>DC Power Input</td>
<td>I</td>
</tr>
</tbody>
</table>

#### Pin Details

**CONT CURRENT LIMIT (P1-10)**

This pin can be used to reduce the continuous current limit without affecting the peak current limit by connecting an external current limiting resistor between this pin and signal ground. See table below.

<table>
<thead>
<tr>
<th>Current Limit Resistor</th>
<th>15 kΩ</th>
<th>6.6 kΩ</th>
<th>3.4 kΩ</th>
<th>2.1 kΩ</th>
<th>1.2 kΩ</th>
<th>810Ω</th>
<th>500 Ω</th>
<th>250 Ω</th>
<th>0 kΩ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Current Limit</td>
<td>90%</td>
<td>80%</td>
<td>70%</td>
<td>60%</td>
<td>50%</td>
<td>40%</td>
<td>30%</td>
<td>20%</td>
<td>10%</td>
</tr>
</tbody>
</table>

Note: These values are secondary to the continuous/peak ratio set by the DIP switches.
HARDWARE SETTINGS

Switch Functions

<table>
<thead>
<tr>
<th>Switch</th>
<th>Description</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Voltage feedback. Mode dependent (see mode selection table below).</td>
<td>On</td>
</tr>
<tr>
<td>2</td>
<td>Current loop integral gain. Activates or deactivates integration. OFF by default.</td>
<td>Inactive</td>
</tr>
<tr>
<td>3</td>
<td>Outer loop integration. Activates or deactivates integration. ON, by default, for current mode and OFF for other modes.</td>
<td>Inactive</td>
</tr>
<tr>
<td>4</td>
<td>Test/Offset. Switches the function of the Test/Offset pot between an on-board command input for testing or a command offset adjustment. OFF by default.</td>
<td>Test</td>
</tr>
</tbody>
</table>

Mode Selection Table

<table>
<thead>
<tr>
<th>Mode</th>
<th>SW1</th>
<th>SW3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENT</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>VOLTAGE</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>IR COMPENSATION</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>TACHOMETER VELOCITY</td>
<td>OFF</td>
<td>OFF</td>
</tr>
</tbody>
</table>

Potentiometer Functions

<table>
<thead>
<tr>
<th>Potentiometer</th>
<th>Description</th>
<th>Turning CW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Loop gain adjustment for voltage/velocity modes. Turn this pot fully CCW in current mode.</td>
<td>Increases gain</td>
</tr>
<tr>
<td>2</td>
<td>Current limit. It adjusts both continuous and peak current limit while maintaining their ratio.</td>
<td>Increases limit</td>
</tr>
<tr>
<td>3</td>
<td>Reference gain. Adjusts the ratio between input signal and output variables (voltage, current, or velocity).</td>
<td>Increases gain</td>
</tr>
<tr>
<td>4</td>
<td>Offset / Test. Used to adjust any imbalance in the input signal or in the amplifier. Can also be used as an on-board signal source for testing purposes.</td>
<td>Adjusts offset in negative direction</td>
</tr>
</tbody>
</table>

Note: Potentiometers are approximately linear and have 12 active turns with 1 inactive turn on each end.
Through-hole Components

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C10*</td>
<td>Current Loop Integrator. Through-hole capacitor that can be added for more precise current loop tuning. See section below on Tuning with Through-hole components for more details.</td>
</tr>
<tr>
<td>C5*</td>
<td>Velocity Loop Integrator. Through-hole capacitor that can be added for more precise velocity loop tuning. See section below on Tuning with Through-hole components for more details.</td>
</tr>
<tr>
<td>R13*</td>
<td>Tachometer Input Scaling. Through-hole resistor that can be added to change the gain of the tachometer input. See section below on Tachometer Gain for more details.</td>
</tr>
<tr>
<td>R30*</td>
<td>Current Loop Proportional Gain. Through-hole resistor that can be added for more precise current loop tuning. See section below on Tuning with Through-hole components for more details.</td>
</tr>
<tr>
<td>R8*</td>
<td>IR Compensation Scaling. Through-hole resistor that must be added to configure the amplifier for IR Compensation mode. See section below on IR Compensation Notes for more details.</td>
</tr>
</tbody>
</table>

Tachometer Gain

Some applications may require an increase in the gain of the tachometer input signal. This occurrence will be most common in designs where the tachometer input has a low voltage to RPM scaling ratio. The drive offers a through-hole location listed in the above table where a resistor can be added to increase the tachometer gain. Use the drive's block diagram to determine an appropriate resistor value.

Tuning With Through-hole Components

In general, the drive will not need to be further tuned with through-hole components. However, for applications requiring more precise tuning than what is offered by the potentiometers and dipswitches, the drive can be manually modified with through-hole resistors and capacitors as denoted in the above table. By default, the through-hole locations are not populated when the drive is shipped. Before attempting to add through-hole components to the board, consult the section on loop tuning in the installation notes on the manufacturer’s website. Some general rules of thumb to follow when adding through-hole components are:

- A larger resistor value will increase the proportional gain, and therefore create a faster response time.
- A larger capacitor value will increase the integration time, and therefore create a slower response time.

Proper tuning using the through-hole components will require careful observation of the loop response on a digital oscilloscope to find the optimal through-hole component values for the specific application.

IR Compensation Notes

For applications that will use IR Compensation mode, a resistor must be added to the location named in the table above. The combination of the added resistor and correct dipswitch settings will configure the amplifier for IR Compensation mode. While in IR Compensation mode, the amplifier will adjust the duty cycle to compensate for changes in the output current. Consult the amplifier's functional block diagram and the manufacturer's website for more information.

†Note: Damage done to the drive while performing these modifications will void the warranty.
### MECHANICAL INFORMATION

#### P1 - Signal Connector

<table>
<thead>
<tr>
<th>Connector Information</th>
<th>Details</th>
<th>Mating Connector</th>
<th>Included with Drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-pin, 2.54 mm spaced, friction lock header</td>
<td>Molex: P/N 22-01-3167 (connector) and P/N 08-50-0114 (insert terminals)</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

![Signal Connector Diagram](diagram1.png)

#### P2 - Power Connector

<table>
<thead>
<tr>
<th>Connector Information</th>
<th>Details</th>
<th>Mating Connector</th>
<th>Included with Drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-port, 5.08 mm spaced, screw terminal</td>
<td>Not applicable</td>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

![Power Connector Diagram](diagram2.png)
**PART NUMBERING INFORMATION**

<table>
<thead>
<tr>
<th>Peak Current</th>
<th>Additional Options*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum peak current rating in Amps.</td>
<td>-ANP: Analog Position Loop</td>
</tr>
<tr>
<td>Peak Voltage</td>
<td>-INV: Inverted Inhibit</td>
</tr>
<tr>
<td>Peak voltage rating scaled 1:10 in Volts.</td>
<td>-QD: Quick Disconnect</td>
</tr>
<tr>
<td>Isolation Option**</td>
<td>-QDI: Quick Disconnect w/ Inverted Inhibit</td>
</tr>
<tr>
<td>I: Optical Isolation</td>
<td>Command</td>
</tr>
<tr>
<td></td>
<td>DD: PWM Command</td>
</tr>
<tr>
<td>Power Supply</td>
<td><strong>Power Supply</strong></td>
</tr>
<tr>
<td>:</td>
<td>- DC Power Supply</td>
</tr>
<tr>
<td>AC:</td>
<td>AC: AC Power Supply</td>
</tr>
</tbody>
</table>

* Options available for orders with sufficient volume. Contact ADVANCED Motion Controls for more information.

** Isolation comes standard on all AC supply drives and most DC supply drives 200V and above. Consult selection tables of the website or the drive datasheet block diagram to see if isolation is included.

ADVANCED Motion Controls analog series of servo drives are available in many configurations. Note that not all possible part number combinations are offered as standard drives. All models listed in the selection tables of the website are readily available, standard product offerings.

ADVANCED Motion Controls also has the capability to promptly develop and deliver specified products for OEMs with volume requests. Our Applications and Engineering Departments will work closely with your design team through all stages of development in order to provide the best servo drive solution for your system. Equipped with on-site manufacturing for quick-turn customs capabilities, ADVANCED Motion Controls utilizes our years of engineering and manufacturing expertise to decrease your costs and time-to-market while increasing system quality and reliability.

**Examples of Modifications and Customized Products**

- Integration of Drive into Motor Housing
- Mount OEM PCB onto Drive Without Cables
- Multi-axis Configuration for Compact System
- Custom PCB and Baseplate for Optimized Footprint
- RTV/Epoxy Components for High Vibration
- OEM Specified Connectors for Instant Compatibility
- OEM Specified Silkscreen for Custom Appearance
- Increased Thermal Limits for High Temp. Operation
- Integrate OEM Circuitry onto Drive PCB
- Custom Control Loop Tuned to Motor Characteristics
- Custom I/O Interface for System Compatibility
- Preset Switches and Pots to Reduce User Setup
- Optimized Switching Frequency
- Ramp Performance Command for Smooth Acceleration
- Remove Unused Features to Reduce OEM Cost
- Application Specific Current and Voltage Limits

Feel free to contact Applications Engineering for further information and details.

**Available Accessories**

ADVANCED Motion Controls offers a variety of accessories designed to facilitate drive integration into a servo system. Visit [www.a-m-c.com](http://www.a-m-c.com) to see which accessories will assist with your application design and implementation.
2.6 Modes of Operation

The family of analog drives offers a variety of different control methods. While some drives in the series are designed to operate solely in one mode, on other drives it is possible to select the control method by DIP switch settings (see “Potentiometer Function Details” on page 43 for more information). Consult the datasheet for the drive in use to see which modes are available for use.

The name of the mode refers to which servo loop is being closed in the drive, not the end-result of the application. For instance, a drive operating in Current (Torque) Mode may be used for a positioning application if the external controller is closing the position loop. Oftentimes, mode selection will be dependent on the requirements and capabilities of the controller being used with the drive as well as the end-result application.

2.6.1 Current (Torque) Mode

In Current (Torque) Mode, the input command voltage controls the output current. The drive will adjust the output duty cycle to maintain the commanded output current. This mode is used to control torque for rotary motors (force for linear motors), but the motor speed is not controlled. The output current can be monitored through an analog current monitor output pin. The voltage value read at the "Current Monitor Output" can be multiplied by a scaling factor found on the drive datasheet to determine the actual output current.

While in Current (Torque) Mode, the drive will maintain a commanded torque output to the motor based on the input reference command. Sudden changes in the motor load may cause the drive to be outputting a high torque command with little load resistance, causing the motor to spin rapidly. Therefore, Current (Torque) Mode is recommended for applications using a digital position controller to maintain system stability.

2.6.2 Duty Cycle (Open Loop) Mode

In Duty Cycle Mode, the input command voltage controls the output PWM duty cycle of the drive, indirectly controlling the output voltage. Note that any fluctuations of the DC supply voltage will affect the voltage output to the motor.

This mode is recommended as a method of controlling the motor velocity when precise velocity control is not critical to the application, and when actual velocity feedback is unavailable.
**Current Reference Output**  Measured relative to signal ground, the current reference provides an analog voltage output signal that is proportional to the command signal to the internal current loop. The drive does not need to be connected to a load to read the current reference output. The internal command current may differ from the actual drive output current due to certain conditions such as a small load, drive faults, undersized power supplies, inhibited drive, etc. The command to the internal current loop can be solved for by the following equation:

\[
I_{\text{command}} = V_{\text{current ref}} \cdot \frac{I_{\text{peak}}}{V_{\text{max}}}
\]

Where:
- \(I_{\text{command}}\) - command current to the internal current loop
- \(V_{\text{current ref}}\) - measured voltage at current reference pin
- \(I_{\text{peak}}\) - peak current value of the drive
- \(V_{\text{max}}\) - voltage corresponding to maximum internal current command value found on drive datasheet; on most drive models \(V_{\text{max}} = 7.45V\)

**Example Measurement**

The current reference pin on a drive with a peak current value of 12A and \(V_{\text{max}}\) of 7.45V is measured to be 2.63V. Following the above equation to solve for \(I_{\text{command}}\), the command current to the internal current loop would be 4.24A.

**Inhibit Input**  This pin provides a +5V TTL input that allows a user to enable/disable the drive by either connecting this pin to ground or by applying a +5VDC voltage level to this pin, referenced to signal ground. By default, the drive will be enabled if this pin is high, and disabled if this pin is low. This logic can be reversed, however, either through DIP switch setting or by removing a SMT jumper from the PCB (consult the drive datasheet to see which option is available; note that removal of the SMT jumper must be done by a person familiar with SMT soldering, and that the drive warranty will be voided if the drive is damaged). This will require all inhibit lines to be brought to ground to enable the drive. Most drives can also be ordered with inverted inhibit logic as well (-INV option). Some drive models allow the drive to be configured so the inhibit input does not trigger a drive fault state. Typically this is achieved by DIP switch setting. Consult the drive datasheet to see if this option is available.

**Directional Inhibits**

Some drives also include directional inhibit pins that disable motor motion in either the positive or negative direction, typically used for limit switches. These pins do not cause a drive fault condition. They will follow the same logic (either standard or inverted) as the main inhibit/enable input.

**Continuous Current Limit Pin**  The Continuous Current Limit pin can be used to reduce the factory-preset maximum continuous current limit without affecting the peak current limit of the drive by attaching an external resistor between this pin and signal ground. Values for resistors and the corresponding reduction in continuous current are given on the drive datasheet. This continuous current reduction comes secondary to any reductions made by DIP switch settings on the drive and the current limiting potentiometer.
4.1.2 Potentiometer Function Details

All potentiometers vary in resistance from 0 to 50 kohm, over 12 turns. An additional full turn that does not affect resistance is provided on either end, for a total of 14 turns. When the end of potentiometer travel is reached, it will click once for each additional turn. Consult the drive datasheet to see which potentiometers are included on a specific drive.

### TABLE 4.1 Potentiometer Function Details

<table>
<thead>
<tr>
<th>Potentiometer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop Gain Adjustment</td>
<td>This potentiometer must be set completely counter-clockwise in Current Mode. In Velocity, Voltage, or Duty Cycle Mode, this potentiometer adjusts the gain in the velocity forward position of the closed loop. Turning this potentiometer clockwise increases the gain. Start from the full counter-clockwise position, turn the potentiometer clockwise until the motor shaft oscillates, then back off one turn.</td>
</tr>
</tbody>
</table>
| Current Limit                 | This potentiometer adjusts the current limit of the drive. To adjust the current limit, first use any available DIP switches or external current limiting resistors to set the maximum current limits and ratios (consult drive datasheet to see which options are available). If further adjustment is necessary, use the following equation to determine the number of clockwise turns from the full counter-clockwise position necessary to set the desired current limit:  

\[
\text{# of turns (from full CCW)} = \left( \frac{I_{\text{system}}}{I_{\text{max}}} \right) \times 12 + 1
\]

- \(I_{\text{system}}\) = the desired current limit of the system (typically determined by motor current rating)
- \(I_{\text{max}}\) = maximum current capability of the drive; this value is determined after any external current limiting resistors have been used and/or any current scaling or current reduction DIP switches have been set. If no DIP switches or external resistors have been used, then \(I_{\text{max}}\) is the default maximum continuous current limit set by the drive hardware. See “Current Limiting Procedure” on page 46 for an example of how to use this potentiometer. |
| Reference Gain                | This potentiometer adjusts the ratio between the input signal and the output variable (voltage, current, velocity, or duty cycle). For a specific gain setting, turn this potentiometer fully counter-clockwise, and adjust the command input to 1V. Then turn clockwise while monitoring motor velocity or drive output voltage (depending on mode of operation) until the required output is obtained for the given 1V command. Turning this potentiometer counter-clockwise decreases the reference in gain, while setting this potentiometer in the fully clockwise position makes the whole range of drive output available. This potentiometer may be left in the fully clockwise position if a controller is used to close the velocity or position loops. |
| Test/Offset                   | This potentiometer acts as an internal command source for testing when the Test/Offset switch is in the ON position. If the Test/Offset switch is in the OFF position, then this potentiometer can be used to adjust a small amount of command offset in order to compensate for offsets that may be present in the servo system. Turning this potentiometer clockwise adjusts the offset in a negative direction relative to the +Ref input command. Before offset adjustments are made, the reference inputs must be grounded or commanded to 0 volts. |

**Test Points for Potentiometers** After the potentiometer adjustments have been completed, the resistance values can be measured for future adjustments or duplication on other servo drives of the same part number. Test points for potentiometer wipers are provided and are located at the foot of all four potentiometers. Resistance measurements are only to be used to duplicate drive settings, since some potentiometers have other resistors in series or parallel. Measure the resistance between the test point and the outer leg of the potentiometer or between the test point and an appropriate ground. See the block diagram on the drive datasheet to determine which ground should be used for each potentiometer.

**Notice** Before taking potentiometer resistance measurements, make sure that all potentiometers and DIP switches have been set to the desired settings, and that all I/O and Feedback cables have been removed from the drive, as these can affect resistance measurements.
24 Volt DC - 8” Red (Solid Rubber Tire) With Electrically Released Brake

Max Capacity: 400 Lbs. [181 kg]
Weight: 10.6 Lbs. [4.8 kg]
Sound Level: >74 db @ 6” [152 mm]

Performance Data

<table>
<thead>
<tr>
<th></th>
<th>Power</th>
<th>Torque</th>
<th>Force @ Wheel Dia.</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Volts DC</td>
<td>Amps</td>
<td>Watts in</td>
<td>Watts Out</td>
</tr>
<tr>
<td>No Load</td>
<td>24.07</td>
<td>0.804</td>
<td>19.37</td>
<td>10.67</td>
</tr>
<tr>
<td>Max Eff.</td>
<td>24.05</td>
<td>3.470</td>
<td>63.47</td>
<td>63.65</td>
</tr>
<tr>
<td>Max Power Out</td>
<td>23.93</td>
<td>10.870</td>
<td>260.30</td>
<td>173.90</td>
</tr>
<tr>
<td>(Continuous Duty)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lock Rotor</td>
<td>25.40</td>
<td>58.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brake</td>
<td>18-28</td>
<td>94.80</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Results may vary.

Dimensions

[Diagram of motor and brake specifications]
<table>
<thead>
<tr>
<th>项目(Description)</th>
<th>U</th>
<th>I</th>
<th>P1</th>
<th>M</th>
<th>n</th>
<th>P2</th>
<th>Eff</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(V)</td>
<td>(A)</td>
<td>(W)</td>
<td>(N.m)</td>
<td>(rpm)</td>
<td>(W)</td>
<td>(%)</td>
</tr>
<tr>
<td>空载点(No_Load)</td>
<td>24.06</td>
<td>0.931</td>
<td>22.42</td>
<td>0.44</td>
<td>174.5</td>
<td>8.03</td>
<td>35.8</td>
</tr>
<tr>
<td>最高效率点(Max_Eff)</td>
<td>23.94</td>
<td>5.861</td>
<td>140.3</td>
<td>6.30</td>
<td>155.6</td>
<td>102.6</td>
<td>73.1</td>
</tr>
<tr>
<td>最大输出功率点(Max_Pout)</td>
<td>23.82</td>
<td>11.60</td>
<td>276.4</td>
<td>13.27</td>
<td>135.9</td>
<td>188.8</td>
<td>68.2</td>
</tr>
<tr>
<td>最大转矩点(Max_Torque)</td>
<td>23.82</td>
<td>11.60</td>
<td>276.4</td>
<td>13.27</td>
<td>135.9</td>
<td>188.8</td>
<td>68.2</td>
</tr>
<tr>
<td>结束(End)</td>
<td>23.82</td>
<td>11.60</td>
<td>276.4</td>
<td>13.27</td>
<td>135.9</td>
<td>188.8</td>
<td>68.2</td>
</tr>
<tr>
<td>No.</td>
<td>U</td>
<td>I</td>
<td>P1</td>
<td>M (N.m)</td>
<td>n (rpm)</td>
<td>P2 (W)</td>
<td>EFF (%)</td>
</tr>
<tr>
<td>-----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>---------</td>
<td>---------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>1</td>
<td>24.06</td>
<td>0.931</td>
<td>22.42</td>
<td>0.44</td>
<td>174.5</td>
<td>8.03</td>
<td>35.8</td>
</tr>
<tr>
<td>2</td>
<td>24.06</td>
<td>1.003</td>
<td>24.14</td>
<td>0.47</td>
<td>174.3</td>
<td>8.57</td>
<td>35.5</td>
</tr>
<tr>
<td>3</td>
<td>24.06</td>
<td>1.083</td>
<td>26.06</td>
<td>0.57</td>
<td>174.0</td>
<td>10.38</td>
<td>39.8</td>
</tr>
<tr>
<td>4</td>
<td>24.05</td>
<td>1.246</td>
<td>29.98</td>
<td>0.72</td>
<td>173.8</td>
<td>13.10</td>
<td>43.6</td>
</tr>
<tr>
<td>5</td>
<td>24.05</td>
<td>1.440</td>
<td>34.64</td>
<td>0.95</td>
<td>173.0</td>
<td>17.20</td>
<td>49.6</td>
</tr>
<tr>
<td>6</td>
<td>24.05</td>
<td>1.690</td>
<td>40.65</td>
<td>1.29</td>
<td>171.8</td>
<td>23.20</td>
<td>57.0</td>
</tr>
<tr>
<td>7</td>
<td>24.05</td>
<td>2.095</td>
<td>50.38</td>
<td>1.76</td>
<td>170.6</td>
<td>31.43</td>
<td>62.3</td>
</tr>
<tr>
<td>8</td>
<td>24.04</td>
<td>2.630</td>
<td>63.25</td>
<td>2.37</td>
<td>168.6</td>
<td>41.83</td>
<td>66.1</td>
</tr>
<tr>
<td>9</td>
<td>24.00</td>
<td>3.260</td>
<td>78.28</td>
<td>3.12</td>
<td>165.9</td>
<td>54.19</td>
<td>69.2</td>
</tr>
<tr>
<td>10</td>
<td>23.97</td>
<td>4.015</td>
<td>96.26</td>
<td>4.05</td>
<td>162.6</td>
<td>68.94</td>
<td>71.6</td>
</tr>
<tr>
<td>11</td>
<td>23.94</td>
<td>4.865</td>
<td>116.5</td>
<td>5.08</td>
<td>159.9</td>
<td>85.04</td>
<td>72.9</td>
</tr>
<tr>
<td>12</td>
<td>23.94</td>
<td>5.861</td>
<td>140.3</td>
<td>6.30</td>
<td>155.6</td>
<td>102.6</td>
<td>73.1</td>
</tr>
<tr>
<td>13</td>
<td>23.93</td>
<td>6.921</td>
<td>165.6</td>
<td>7.54</td>
<td>152.3</td>
<td>120.2</td>
<td>72.5</td>
</tr>
<tr>
<td>14</td>
<td>23.86</td>
<td>8.023</td>
<td>191.5</td>
<td>8.95</td>
<td>148.7</td>
<td>139.3</td>
<td>72.7</td>
</tr>
<tr>
<td>15</td>
<td>23.84</td>
<td>9.148</td>
<td>218.1</td>
<td>10.34</td>
<td>144.2</td>
<td>156.1</td>
<td>71.5</td>
</tr>
<tr>
<td>16</td>
<td>23.83</td>
<td>10.36</td>
<td>247.0</td>
<td>11.83</td>
<td>140.0</td>
<td>173.4</td>
<td>70.1</td>
</tr>
<tr>
<td>17</td>
<td>23.82</td>
<td>11.60</td>
<td>276.4</td>
<td>13.27</td>
<td>135.9</td>
<td>188.8</td>
<td>68.2</td>
</tr>
</tbody>
</table>

*金电机*