Project Title: Actively-Controlled Stabilization Platform for Automobile Applications

Team Members: Mohammad Sami Al Chllk
               Jacob Kryder
               Graehm Sack
               John Scheurich

Advisor: Dr. Hossein Oloomi

Date: December 16, 2013
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index of Figures</td>
<td>2</td>
</tr>
<tr>
<td>Index of Tables</td>
<td>3</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>4</td>
</tr>
<tr>
<td>Abstract/Summary</td>
<td>5</td>
</tr>
<tr>
<td>Section I – Conceptual Design</td>
<td>6</td>
</tr>
<tr>
<td>Section II – Prototype Fabrication</td>
<td>16</td>
</tr>
<tr>
<td>Section III – Prototype Testing</td>
<td>23</td>
</tr>
<tr>
<td>Section IV – Evaluation and Recommendations</td>
<td>46</td>
</tr>
<tr>
<td>Section V - Conclusions</td>
<td>48</td>
</tr>
<tr>
<td>References</td>
<td>50</td>
</tr>
<tr>
<td>Appendix 1 – Project Schedule</td>
<td>51</td>
</tr>
<tr>
<td>Appendix 2 – Testing Procedures</td>
<td>60</td>
</tr>
<tr>
<td>Appendix 3 – Microcontroller Programming Code</td>
<td>62</td>
</tr>
<tr>
<td>Appendix 4 – MATLAB Scripts</td>
<td>67</td>
</tr>
</tbody>
</table>
Index of Figures

Figure 1: Conceptual diagram of a gimbal table................................................................. 10
Figure 2: Simplified circuit diagram of a DC motor attached to the platform. ................. 11
Figure 3: Simulink block diagram of the open loop motor system................................. 12
Figure 4: Modeling of the multi-feedback loop system.................................................... 13
Figure 5: Angular position calculation of the platform after the initialization sequence. .... 13
Figure 6: Finished prototype......................................................................................... 14
Figure 7: Modified Motor bracket for the X-axis ............................................................ 17
Figure 8: Bracket used for mounting the Y-axis motor.................................................... 18
Figure 9: The Y-axis motor mount.................................................................................. 19
Figure 10: Circuit Diagram ............................................................................................. 20
Figure 11: Angular position using the angle gauge.......................................................... 25
Figure 12: Determination of the accelerometer gain and bias in the x axis ...................... 27
Figure 13: Determination of gyroscope gain and bias ..................................................... 28
Figure 14: Step response in the Y-axis CCW acceleration .............................................. 29
Figure 15: Gain calculated from accelerometer in the Y-axis CCW direction..................... 29
Figure 16: Step response in the Y-axis CW acceleration ................................................. 30
Figure 17: Accelerometer gain in the Y-axis CW direction............................................. 30
Figure 18: Acceleration in the X-axis CCW direction...................................................... 31
Figure 19: Step response in the X-axis CW direction...................................................... 31
Figure 20: Gain calculation of the Accelerometer in the X-axis CW direction................... 32
Figure 21: Complementary filter response tracking....................................................... 33
Figure 22: Model of the Plant (Motor) .......................................................................... 34
Figure 23: Full System with Controller ......................................................................... 35
Figure 24: Impulse Response of X Axis......................................................................... 36
Figure 25: Error on X Axis Produced by an Impulse ...................................................... 37
Figure 26: Impulse Response of Y Axis......................................................................... 38
Figure 27: Error on X Axis Produced by an Impulse ...................................................... 39
Figure 28: Step Response of X Axis .............................................................................. 40
Figure 29: Error on X Axis Produced by a Step ................................................................ 41
Figure 30: Step Response of Y Axis .............................................................................. 42
Figure 31: Error on Y Axis Produced by a Step ................................................................ 43
Figure 32: Optimized Simulink Model of the Full System ............................................. 44
Figure 33: Optimized Simulation of Impulse Response ................................................... 44
Figure 34: Optimized Simulation Step Response ............................................................. 45
Index of Tables

Table 1: Component Testing .................................................................................................................. 24
Table 2: Performance Requirements ................................................................................................. 25
Table 3: Range of Motion .................................................................................................................. 26
Table 4: Measurements Taken to Determine Gain of the Motor Driver ........................................... 34
Table 5: Constants Used for Simulation ............................................................................................ 34
Table 6: Controller Gain Constants ................................................................................................... 35
Acknowledgements

We would like to thank Kerry Scheurich for fabrication feasibility analysis and support. His technical knowledge and skill in aluminum fabrication and machining, along with his donated time, were critical to the success of the development of the prototype fabrication.

We would also like to thank Gary Traxler for use of his machine shop which allowed us to fabricate the prototype in a timely and cost-effective manner.

We would also like to thank Dr. Hossein Oloomi for his technical support and assistance without which this project would not have been possible.
Abstract/Summary

During the operation of a vehicle, pitching and rolling cause surfaces in the interior of the vehicle to deviate from level. This can cause driver inattention when an object within the vehicle, such as a beverage, needs to be stabilized and level. Automatic control systems are used to provide a desired output from a system during changing operating conditions without human intervention. A common application of control systems is stabilization of a system throughout a variety of changing environmental conditions.

By employing an automatic control system, a platform can be kept level during movement of the vehicle. This can reduce driver inattention during demanding driving conditions. This stabilized platform must be powered by the auto's electrical system, be lightweight and have a small footprint so as to allow integration into the interior of an automobile.

This report covers the design process beginning with the implementation of the detailed conceptual design selected in Report #1. In particular, this report covers the fabrication process and software development, calibration and testing as well as evaluation and recommendations for improvement.
Section I

Conceptual Design
To address the need for an actively-controlled stabilized platform within a vehicle occupant compartment, a detailed design process determined the best conceptual design is a gimbal table utilizing PMDC motors for actuation and relying on sensor feedback from an accelerometer and a MEMS gyroscope. A detailed analysis of the conceptual design process can be found in Report #1. This document addresses the implementation and refinement of the prototype. This section contains a summary of the amended problem statement, requirements and specifications, an overview of the subsystems, and the mathematical model.

Requirements and Specifications

To ensure a stabilization system with sufficient performance under the conditions that exist in the intended application, the following requirements and specifications are set forth.

- **System Response**
  - The control system shall maintain a level surface with a maximum of 10% overshoot when subjected to a step input.
  - The control system used must be able to perform corrections of 30˚ within 0.25 seconds.
  - The steady state tracking error of the control system shall be no more than 5%.

- **Range of Motion**
  The stabilized platform will have a minimum 60˚ range of motion on each rotational axis.

- **Power Consumption**
  The maximum current draw of the system must be less than 15 amperes.

- **Device Physical Characteristics**
  - To ensure the device does not encroach on occupant space, the footprint shall be no more than 500 in³.
  - The mass of the device shall be no more than 5 lbs.

- **Load Capacity**
  To accommodate different types and sizes of objects, the platform shall handle an object weight up to 2.0 pounds. This accommodates either a digital camera such as the Nikon - D5100 weighing 1.2 lbs. or a 24 oz. beverage weighing approximately 1.7 lbs.

After carefully considering the proposed specifications for our stabilization system, the different subsystems necessary for a stabilization system to operate were identified and investigated to determine the most suitable options. These systems were identified as follows: power supply, frame and platform, sensor, control system and drive system. The power supply must be able to deliver power to the required subsystems either directly from the vehicle’s electrical system or from a rechargeable battery source. The frame and platform contains the entirety of the device and provides the surface which must be stabilized. The sensors determine the state of the system and each sends a signal that describes that state. The control system interprets the signals sent by the sensors and initiates an appropriate response, which also induces a
feedback signal based on both sensors’ response. The drive system receives a signal from the controller and responds accordingly by bringing the platform back to a stable position (i.e. level). Neglecting any of these subsystems renders the device incapable of fulfilling its need.

**Control Subsystem:**

The control system is the computational center of the device. It must be able to detect when the device is operating properly and correct that operation when it is not. The control system receives a sensory signal and based on that signal prompts a response from the drive system. That response is then re-sensed and fed back through the control system which creates a feedback loop. This endless loop is what allows the stabilization device to constantly maintain a level surface. As in most electrical systems there is a defining choice to make as to how a system will send and receive signals, analogously or digitally.

A digital control system for this application requires the sensory input and driving output to be converted to a digital signal. Some drive systems operate specifically on a pulse width modulation (PWM) scheme, which is a digital communication scheme. The digital approach allows for simple changes to be made throughout the design process through the programming of the digital control device source code.

The digital system was selected primarily as it is a relatively cheap option where reasonably-sized microcontrollers range from $20-$70. Additionally, with a digital system there is a wealth of open source code which can be adapted, modified, and re-created to generate a working digital control device. This provides additional savings in cost and manpower resources.

In order for the digital control system to work properly with any analog components it must be interfaced accordingly. This requires additional signal processing. These additional steps introduce an increased delay in the system response. This delay may cause detrimental device behavior and must be accounted for when digital control schemes are being considered. In particular, the data transmission rates and processing times are critical to the appropriate response signal being sent from the controller. Code optimization is a key to limiting the negative implications of the processing delay.

**Drive Subsystem:**

For the platform to be stable and also maintain a level surface, a method of controlled rotation must be developed. The drive system of the platform receives signal commands from the control system to rotate the surface along the two axes in order to maintain level orientation. Many different methods of providing rotational motion were considered and the DC gear motor was chosen.

DC gear motors are widely available with operational voltages between 3 V and 24 V. Depending on the power and speed required, a motor that will produce satisfactory performance is commercially available and a stock item. The PWM frequency used as a control signal is configurable and based on duty cycle opposed to pulse width in time. This allows for a greater resolution of control signal and a faster response. Many commercially available motor drivers
have a current sense feedback that expands the number of possible configurations of the control system.

The impact of choosing DC gear motors is that the physical size of gear motors is typically larger than servo motors. This results in an increased footprint of the system which may adversely affect the performance of the system particularly in this application. Motor control hardware must also be purchased for the application which results in added cost to the design and added complexity in the source code.

**Sensor Subsystem:**

A major component of our device is the sensing system. In order for the control system to determine the necessary corrections, there needs to be a constant stream of data indicating the relative position and predictors of future position such as velocity and acceleration. The sensors selected are the digital gyroscope and accelerometer.

Gyroscopes are effective in providing the necessary sensory feedback in the form of angular velocity. The cost is relatively low and the resolution available is adequate for this project. However, there are some drawbacks. Drift occurs over time as the integration error compounds when determining the platform’s position. The sensor cannot determine the global level and also changes in the levelness of the platform.

For this reason, an accelerometer is also used to provide feedback to the control system. Accelerometers provide a low-cost solution to the sensory system requirements and demonstrate flexibility in being either analog or digital models. Accelerometers can determine the global level of the platform at start-up. Some design considerations for using accelerometers are the resolution sensitivity particularly when gravity readings are important and the adverse effect of high level of noise created when the vehicle is in motion.

**MEMS Gyroscope**

The gyroscope measures the angular velocity about a given axis. Micro-electro-mechanical systems (MEMS) gyroscopes use the Coriolis Effect to measure the angular rate by utilizing a tuning fork configuration. Two masses oscillate and move constantly in opposite directions. When angular velocity is applied, the Coriolis force on each mass also acts in opposite directions, which result in a change of capacitance. This differential value in capacitance is proportional to the angular velocity and is then converted into output voltage for analog gyroscopes or LSBs for digital gyroscopes. When linear acceleration is applied to two masses, they move in the same direction. Therefore, there will be no capacitance difference detected. The gyroscope will output zero-rate level of voltage or LSBs, which shows that the MEMS gyroscopes are not sensitive to linear acceleration such as tilt, shock, or vibration. Supply voltage can be between 3-5 V with the output is a low voltage signal either analog or digital. Resolution is available from 2° to 25000°.
Accelerometer

Tilt or inclination sensors utilize an accelerometer to determine position through gravitational forces. Accelerometers are used to sense both static (e.g. gravity) and dynamic (e.g. sudden starts/stops) acceleration. One of the more widely used applications for accelerometers is tilt-sensing. Because they are affected by the acceleration of gravity, an accelerometer can be used to determine its orientation with respect to the Earth’s surface. A smaller full-scale range means a more sensitive output; therefore more precise reading is output from an accelerometer with a low full-scale range. A range of 1-2 g is appropriate for this application.

Accelerometers with an analog output will produce a voltage that is directly proportional to the sensed acceleration. At 0 g, the analog output will usually reside at about the middle of the supplied voltage (e.g. 1.65 V for a 3.3 V sensor).

Frame and Platform Subsystem:

The platform design is a gimbal table. Figure 1 below illustrates the interaction between components. The platform is attached by pins to the first frame, which itself is attached to a fixed second frame with pins. The first frame will allow the movement along the pitch axis, while the other will allow the movement along the roll axis.

![Figure 1: Conceptual diagram of a gimbal table](image)

This design minimizes the complication in the delivery of motor torque compared to other frame systems. This design also yields a slightly smaller footprint with easily fabricated parts. An additional design consideration is that the motors must not only provide the movement, but also be able to maintain the position of the platform and handle the stress created by the weight of the placed object on the platform. Motor interface with the frame requires more consideration with respect to space constraints and mechanical connection issues.

Control System Modeling

For permanent magnet direct current (PMDC) motors, control is done by adjusting the armature current. Motor speed is not controlled as the armature current is used to adjust the motor shaft’s position.

The simplified circuit representation of the PMDC motor and actuator system is shown in Figure 2 below where:

\[ v_a(t) = \text{armature voltage (V)} \]
\[ i_a(t) = \text{armature current (A)} \]
\[ R_a = \text{armature resistance (}\Omega) \]
\[ L_a = \text{armature inductance (H)} \]
\[ v_b(t) = \text{back emf (V)} \]
\[ T_m = \text{torque generated by the motor (N-m)} \]
\[ \omega_m = \text{angular velocity of the motor shaft (rad/s)} \]
\[ \theta_m = \text{position of the motor shaft (rad)} \]
\[ J_m = \text{moment of inertia of the rotor} \]
\[ B_m = \text{viscous friction coefficient of the motor} \]
\[ T'_m = \text{net torque generated by the motor (N-m)} \]
\[ T''_m = \text{torque of the motor after gear reduction (N-m)} \]
\[ T_l = \text{torque generated on the load (N-m)} \]
\[ \omega_l = \text{angular velocity of the load (rad/s)} \]
\[ \theta_l = \text{position of the load (rad)} \]
\[ J_l = \text{moment of inertia of the load} \]
\[ B_l = \text{viscous friction coefficient of the load} \]
\[ g_1 = \text{number of teeth of drive gear} \]
\[ g_2 = \text{number of teeth of driven gear} \]

**Figure 2:** Simplified circuit diagram of a DC motor attached to the platform.

From Figure 2 above, the plant transfer function can be derived:

\[
\frac{\theta_l(s)}{V_a(s)} = \frac{nK_t}{s[L_a J_{eq} s^2 + (L_a B_{eq} + R_a J_{eq}) s + R_a B_{eq} + K_t K_b]}
\]

(1)

From this equation, the motor block diagram can be generated as shown below in Figure 3. A reference voltage signal correlating to the angular velocity of the platform is input into the motor system and summed with the back emf \(v_b\) to generate the armature voltage to the motor. The resulting torque produced by the motor \(T_m\) is summed with the input torque of the load \(T_{load}\) to create the total torque \(T_e\) on the system. Using the transfer function found in Equation 1, this torque signal is then converted into the angular velocity of the motor shaft. Integrating and multiplying by the gear ratio \(n\) results in the output signal correlating to the platform position.
**Figure 3:** Simulink block diagram of the open loop motor system

Understanding the input and output variables of the plant consisting of the motor, gearing and platform, a block diagram of the entire system can be generated. The system consists of the plant, controller, sensors and an amplifier as shown in Figure 4 below. A reference signal for the initial angular position is input into the system and summed with the measured angular position of the platform to generate an error signal. The error signal is conditioned by the proportional-integral (PI) controller and then amplified. This voltage is then input to the motor creating the desired torque to level the system. The angular velocity of the shaft is output from the motor and amplified by the gear ratio to simulate the reading from the gyroscope sensor. The shaft position is determined by the accelerometer and combined with the gyroscope reading to generate the feedback signal. This value representing position of the platform is the fed back into the summer to produce the error signal.

With the closed loop control system established, a transfer function can be created to reflect the response of the controller. By considering an open circuit between the controller amplifier gain ($K_a$) and the feedback signal of the gyroscope ($K_g$), the equation for this transfer function can be created as shown below:

\[
T(s) = \frac{nKK_aK_g}{L_aJ_eqs^3 + (L_aB_{eq} + R_aL_eq)s^2 + (R_aB_{eq} + K_bK_t + nKK_aK_g) s + nKK_aK_g}
\]

(2)

where  
$K =$ system gain constant  
$K_a =$ accelerometer gain constant  
$K_g =$ gyro conversion constant

Based on hardware selection, the following variables are defined:

$K_0 = 0.000133 \text{ bits-s/ rad}$  
$K_i = 3.34 \text{ oz-in/ A}$  
$L_a = 1 \text{ mH}$  
$R_a = 1.17 \text{ Ω}$  
$K_b = 2.47 \text{ V/ krpm}$  
$K_m = 3.09 \text{ oz-in/ W}^{1/2}$
The closed loop system can be modeled in Simulink as shown in Figure 4 below.

**Figure 4: Modeling of the multi-feedback loop system**

**Sensor Data Processing**

During the initial start, the system begins by reading the accelerometer and the microcontroller integrates the reading to get the initial angle of the platform as shown in Figure 5 below. This angle becomes the reference angle for the system. At this point the controller begins the steady state process by reading the gyroscope and calculating any change in the angle of the platform. Then the microcontroller reads the accelerometer and calculates the angle of the platform. Next, the filtering algorithm calculates the estimated error and passes that value to the control equations to generate the appropriate PWM duty cycle. Based on this PWM signal, the necessary corrections to the X (roll) and the Y (pitch) axes are relayed to the motor. The microcontroller repeats these steps until the system is powered off.

**Figure 5: Angular position of the platform is calculated in a continuous loop process for each axis after the initialization sequence.**
**Mechanical Design**

Material selection consists of aluminum sheet stock for the platform and a combination of aluminum bar stock and aluminum angles for the frame system. The aluminum sheet was selected to ensure adequate rigidity and durability of the platform throughout the device’s life cycle.

![Finished prototype](image)

**Figure 6: Finished prototype**

As shown in Figure 6 above, the two sensors are mounted to the underside of the platform as close to the platform centroid as feasible. The gyroboard is stacked on top of the accelerometer. The controller is mounted directly to the aluminum angle base.

For the remainder of this report, all data relative to the outer frame is referenced as the x axis and the inner platform is considered to be the y axis.
Section II

Prototype Fabrication
Hardware and Frame Components:

The mechanical components of the prototype platform were fabricated according to the drawing packet contained the previous report. Minor issues such as missing dimensions and incomplete hardware data were resolved verbally with the fabricator. The entire fabrication process took 42 man-hours and incurred a cost of $12 in shop supplies. There was no cost for labor, since the fabricator donated his time to this project.

Several modifications were made to the detailed design previously put forth. These modifications were limited to the mounting methods of the motors, to allow for adjustment of in the location of the motors. The first change that was made was the motor that actuated produced movement on the Y-axis. Instead of the motor being mounted to the external part of the platform which would cause it to extend beyond the frame of the platform, it was moved to a position directly under the platform. This was necessary for a control algorithm that is symmetric about both axes. The face mount in the original detailed design was decided to be infeasible due to the difficulty in screw-hole location in the fabricated bracket. While it was possible to fabricate a bracket as such, this method does not allow for adjustment in the location of the motor. The method chosen for mounting of the motors was to use a U-bolt that would clamp the motor to a flat surface of a bracket. This allows for adjustment longitudinally and rotationally. First, 18 gauge steel was bent up in an L-shape as seen in Figure 7. This shape of bracket was used for the motor that controls the movement of the X-axis.

Figure 7: Modified Motor bracket for the X-axis

A similar bracket was bent up from the same material but in a U-shape for mounting of the Y-axis motor. Figure 8 is an illustration of the bracket fabricated for the Y-axis motor.
In initial tests of the platform, these motor brackets proved to have too much flex in them and would result in too much extraneous movement of the platform. It was then decided the brackets would have to be made of a stiffer material. These same brackets were fabricated out of 3/16” mild steel. These brackets proved to be stiff enough so that all flex in the motor mounting system was removed.

The next problem faced in fabrication was the mounting of the Y-axis motor. By mounting it to the bottom of the platform, the weight of the motor and bracket produced a torque around the X-axis. The resulted in an unbalanced load on the X-axis motor. This presented significant difficulties in obtaining any sort of a stable system. It was then determined that a mounting system would have to be created which the weight of the motor and bracket would have to be supported by the frame and not be bared by any part of the platform itself. Figure 9 shows the new Y-axis motor mount that is supported by the frame legs and rotates along the X-axis. These were the only modifications to the structure of the platform.
The components of the control system are the microcontroller, a motor driver to power the two PMDC motors and the two sensors, a gyroscope and an accelerometer. As shown in Figure 10 below, the system power is provided to the motor driver which in turn powers the controller. The sensors draw power from the microcontroller. Both sensors output directly to the microcontroller via the Analog In pins located by the processor chip. The microcontroller in turn transmits a PWM signal to the motor driver using the Digital pins located on the opposite side. The motor driver converts the input signal frequency to one used by the motors and relays the corresponding PWM signal to each motor. Connections between the sensors and controller are soldered while stackable headers and terminal blocks are used to connect the motor driver with the microprocessor.
**Figure 10:** Circuit Diagram shows the wiring connections from the microcontroller to the motor driver and sensors.

The accelerometer is mounted directly to the underside of the platform as close to center as possible. The gyroscope is stacked on it by using standoffs to mount the accelerometer and screws to secure the gyroscope.

**Software Development:**

The software development was broken into several different functions as outlined below. Additionally, different methods and algorithms were derived to improve the performance of the system throughout the testing period.

- LCD display:

The first taken approach with the programming process was to connect an LCD to the MCU to display the sensors outputs, which will help with the troubleshooting process. However, this method turned out to be unnecessary after discovering that the Arduino Programming Software (1.0.5) has a serial monitor that allows the programmer to display on it in real-time, which has been a huge bonus for the project, since there was too much information to be displayed which the LCD couldn't accommodate on its two displaying lines. Wiring the LCD and connecting it to the MCU and then programming the MCU for the LCD was a waste of time.
• Motor Driver:

Although the motor driver, which is also referred to as motor shield, can be used with any type of motors and interfaced with any microcontroller, it has made for Arduino applications which means that it comes with a lot of documentations that ease the process of integrating it with the project. The manufacture has a pre written code and heard file that can be easily integrated with the code and reduce the learning curve.

The use of this particular motor driver introduced some problems that will be discussed in the following section; however, the motor driver had more advantages over other commercial alternatives that excluded the idea of having substituted. A major advantage of using this motor was its ability to handle high currents up to 30 Amps compared to the 2-10 Amps commercial alternatives.

Most of the problems, one way or the other, was associated with motor driver.

  o Taking up all the pins:

The motor driver, or so called motor shield, supposed to shield the motor driver logical layer from the top layer in a way that preserve all the pins or at least most of them. However, it didn’t and furthermore it occupied the unused pins which aren’t being directly connected to driver for command. For example p.13 is one of the pins that is supposed to be free, however one of the motor led direction indicators was connected to it.

This was very inconvenient since it prevented us to connect anything else, like an LCD to display sensor data, a button to rest or to calibrate, or a knob which would have been greatly advantages into calibrating the PI controller constants rather than manually changing the values in the code and recompiling and uploading the code to the MCU.

  o Directional Un-linearites:

The motor driver provided a different motor speed on each axis and each direction for the same output value. This might be associated to the motors themselves rather than being caused by the motor driver. This problem was addressed by detecting this un-linearites and fixing it in the code by outputting the corrected value.

  o Voltage Regulator:

The motor driver on chip voltage regulator introduced a different DC offset on the accelerometer angular readings for three different power source cases(USB powered, power supply powered, both USB and PS powered, and in the car powered). This means different code for different power source since that it is a different DC offset and different accelerometer calibration.
• Accelerometer:

Programming the code for the analog accelerometer wasn’t much of a challenge, especially after mounting it to the bottom of the platform and displaying its output data, which gave away the relationship between the analog output voltage and the angular displacement.

• Gyroscope Calibration:

Luckily The Arduino Wire library supports I2C protocols which is the protocol that is used to get the data from the digital gyroscope’s buffer. The gyroscope can operate on different buffer transfer modes. For this project stream mode, which is the default mode, is used since it provides the most recent angular velocity rate. The data is provided as a signed 16 bit digital number for each of the axis. The digital number needed to be processed and converted into meaningful information, angular displacement in this case.

On the other hand, calibrating the gyroscope and eliminating the drift consumed a good amount of time. The datasheet oddly didn’t include the procedure of converting the digital number into angular displacement, or how to eliminate the drifting problem.

• Complimentary Filter:

Implementing the complimentary filter was straightforward process, however, varying its parameter to test the behavior for different values was time consuming with manually changing the values in the code and recompiling and uploading the code to the MCU.

• Control Algorithm:

  o Proportional:

At first a pure proportional was used to control the platform by simply having the output to the motors multiplied by the output of the complimentary filter (the error) as $\text{Output}_X(\text{Proportional} \times \text{Error})$;

This method didn’t lead to solid results since that it would have too much gain for high error values which causes a big overshoot, and a very small gain for small error values which isn’t enough to move the motors.

  o Conditional:

The use of if..else conditional statements improved the performance with the use of different proportion values for different intervals, which eliminated some of the previous discussed disadvantages.

  o Proportional Integral:

The PI controller proved to be a great solution for the problem. It had a fast response and better study state error correction capabilities when it is compared to the previous ones. However, it had an undesired behavior caused by the motor’s gear backlashing. It slowly oscillates around the 0 degrees and can’t come to a complete stop.
Section III

Prototype Testing
Testing Criteria and Procedures

In-depth testing is performed to ensure that the prototype successfully addresses the problem statement. Testing of the design is organized into two categories, component verification and project performance requirements. For each test the expected result must be determined. Therefore, test variables and validation criteria must be determined.

For component verification, the following tests must be performed as indicated in Table 1 below. Each major component is listed with the necessary tests, the corresponding output measurement and unit of measure. Function testing of the electrical devices ensures the components are free of defects. Additionally the two sensors are tested for calibration with the system after mounting. The prototype power supply is tested to verify it is within the design parameter of 9 to 16 V.

Table 1: Component Testing

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Response variable</th>
<th>Unit of Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>Determine rotation about pitch and roll axes</td>
<td>Angle</td>
<td>Degree</td>
</tr>
<tr>
<td>Microcontroller</td>
<td>Function test of input and output pins</td>
<td>Digital logic and PWM signal output</td>
<td></td>
</tr>
<tr>
<td>Accelerometer</td>
<td>Function test and calibration</td>
<td>Angular position</td>
<td>Degree</td>
</tr>
<tr>
<td>Gyroscope</td>
<td>Function test and calibration</td>
<td>Angular velocity</td>
<td>Degrees per second (dps)</td>
</tr>
<tr>
<td>DC Motor (2)</td>
<td>Confirm motor design parameters per manufacturer's specifications</td>
<td>Torque constant, armature resistance, armature inductance</td>
<td></td>
</tr>
<tr>
<td>Motor Driver</td>
<td>Function test of input and output pins</td>
<td>Digital logic and PWM signal output</td>
<td></td>
</tr>
<tr>
<td>Battery</td>
<td>Verify power supply range</td>
<td>Voltage</td>
<td>Volts (v)</td>
</tr>
</tbody>
</table>

Based on suitable components, the assembled prototype must meet the performance guidelines stated in the design phase of this project. Table 2 below lists the system requirements, the measured output variable, corresponding unit of measure and the acceptable value range necessary to validate the design solution. The system response overshoot and steady state tracking error requirements are not measured variables but calculated based on the measured response of the sensors.
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Response variable</th>
<th>Unit of Measure</th>
<th>Acceptable value range</th>
</tr>
</thead>
<tbody>
<tr>
<td>System response overshoot</td>
<td>Calculated from angle measurements</td>
<td>Unit-less</td>
<td>10% maximum</td>
</tr>
<tr>
<td>Correction time</td>
<td>Time</td>
<td>Seconds</td>
<td>250 ms or less for a 30° error correction</td>
</tr>
<tr>
<td>Steady state tracking error</td>
<td>Calculated from angle measurements</td>
<td>Unit-less</td>
<td>5% maximum</td>
</tr>
<tr>
<td>Range of motion</td>
<td>Angle</td>
<td>Degrees</td>
<td>30° deviation from reference angle</td>
</tr>
<tr>
<td>Power consumption</td>
<td>Current</td>
<td>Amperes</td>
<td>15 amperes maximum draw</td>
</tr>
<tr>
<td>Footprint size</td>
<td>Volume</td>
<td>Cubic inches</td>
<td>500 cubic inches maximum</td>
</tr>
<tr>
<td>Unit weight</td>
<td>Weight</td>
<td>Pounds</td>
<td>5 lbs. maximum</td>
</tr>
<tr>
<td>Load capacity</td>
<td>Weight</td>
<td>Pounds</td>
<td>2 lbs. minimum</td>
</tr>
</tbody>
</table>

**Physical Specifications**

The range of motion was measured using a Wixey Model WR300 Digital Angle Gauge as shown in Figure 11 below. A torpedo bubble level was used to determine the global level with the gauge being calibrated from this position. Moving the platform through the full range of operation for each axis yielded the relative range of motion.

![Determining the platform angular position using the angle gauge.](image-url)
Table 3: Range of Motion

<table>
<thead>
<tr>
<th>Axis of Measurement</th>
<th>Clockwise Angle</th>
<th>Counterclockwise Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>X - axis</td>
<td>30.1</td>
<td>31.2</td>
</tr>
<tr>
<td>Y - axis</td>
<td>31.3</td>
<td>31.3</td>
</tr>
</tbody>
</table>

Power consumption was calculated by measuring the maximum current drawn by the motor driver and the maximum voltage supplied to the motor driver. The maximum current was 3.5 Amps when the platform was given a step response. The non-vehicular power supply used in testing was rated for 12 V but actually provided 13.5 V under testing conditions. This indicates that the maximum power consumption will not exceed 45.25 W.

The overall size of the device is 8.25” x 8.25” x 8” high measured with a standard tape measure with a 1/16” resolution. This equates to a volume of 544.5 cubic inches.

The device was weighed on a Salter Brecknell Model 311 scale rated to 11 pounds with a resolution of 0.1 ounces. A total of three measurements were taken with the results being consistent within 0.1 ounces. The device was determined to weigh 6 pounds, 6.5 ounces.

Sensor Calibration

Upon mounting, the accelerometer was calibrated using a Wixey Model WR300 Digital Angle Gauge as shown in Figure 11 above. This level provides an accuracy of 0.1° when measured at 0°. The test procedure is outlined in Appendix 2 – Testing Procedures. The results of the X axis calibration are shown in Figure 12 below.
Figure 12: Determination of the accelerometer gain and bias in the x axis.

This indicates that the accelerometer reference voltage for the x-axis is 1.72 V. There is a linear increase as the platform is rotated to 30° and then stabilized for half a second. There is then a linear decrement as the platform returns to the global level and another linear decrement as the platform is rotated to its clockwise limit before returning to the global level. Linearizing the platform change in position indicates that the gain constant averages out to be 75.

Testing of the Y axis indicated the global level to be 1.63 V likely due to the chip’s center being closer to the platform center than in the X axis. With the voltage varying from 0 to 3.3 V, a precisely aligned accelerometer would read 1.65 V at global level. The gain constant for the Y axis is the same as the X axis.

The calibrated accelerometer angular data is then used to calibrate the gyroscope. Since the gyroscope returns digital values relative to the angular velocity of the platform, the data must be integrated over the time period to determine the change in angular position. The procedure for calibrating the gyroscope can be found in Appendix 2 – Testing Procedures and is based on a technique devised by Mark Looney of Analog Systems. [1]
Figure 13: Angular velocity of the platform as it moves from global level to maximum counterclockwise displacement, remains stationary for approximately 1 second, moves back to level and then to the maximum clockwise displacement before returning to global level.

As shown in Figure 13 above, the gyroscope has an initial bias of approximately 159 mdeg/s. This is considered drift as it will indicate the platform is moving over time when integrated. Subtracting the average drift allows for the integration during movement to be done using Reimann sums. Comparison of the integrated angle with the accelerometer data indicates the gain of the gyroscope is approximately 0.18 along both axes.

Initial Step Input Readings

To generate initial step input readings without damaging the motors or the linkage, the microcontroller was programmed to provide a step response in the form of a constant PWM signal of 47 to the motor. The platform was oriented at maximum angle before applying power to the motor driver. The measured response for each axis in both directions is detailed below in Figure 14 through Figure 20.
**Figure 14:** With the platform initially at -33°, movement begins at 0.8 seconds and reaches maximum extension at 1.2 seconds with an ending position of +44°.

**Figure 15:** The movement appears to be linear for the time period of 0.8 to 1.2 seconds with a gain of 2.8 V/s.
**Figure 16:** The platform traverses from 42° to -33° and appears linear from 0.6 to 0.8 seconds.

**Figure 17:** The linearization of the axial acceleration results in a gain of -3.4 V/s
**Figure 18:** Acceleration in the x-axis from -37° to +49° indicates a non-linear response likely due to the asymmetrical motor position.

**Figure 19:** Platform moves from 52° to -36° with one outlier data sample occurring at 1.7 seconds.
Figure 20: Linearizing the response from 1.4 to 1.8 seconds indicates a gain of -4.2 V/s.

Based on this data with Figure 18 in particular, the decision was made to relocate the X-axis motor to a more central location. This necessitated the need to revise the plant moment of inertia but allowed for a more symmetrical response to counterclockwise and clockwise error corrections made by the microcontroller.

Complementary Filter Design

The complementary filter design requires that the data from both sensors is incorporated using specific design ratio. Adjusting this ratio affects the sensitivity of platform to lateral movement against the settling time. The contribution of the accelerometer readings provide a quicker response to a step response but create significant noise to the system when there is axial movement. A ratio of 93:7 has been selected to meet the performance goals of this project.
Calculating Experimental Values

After completing the construction of the prototype it is necessary to recalculate values to better model the system. The accelerometer, gyroscope, and motor driver gain are primarily the components in question. The remaining values are based on the datasheet provided by the motor manufacturer and digital values calculated and contained within the microcontroller. Both the accelerometer and the gyroscope convert an angle into a voltage. The method used to determine the gain for each of these components is the same. These gains are found by taking output voltage measurements at two different angles and calculating the slope. After taking measurements we determined that at 44° and 33° the accelerometer produced output voltages of 2.2V and 1.17V respectively and the gyroscope produced output voltages of -1.37V and 0.01V. The calculations are as follows in Equations 3 and 4:

\[ K_{ac} = \frac{2.20 - 1.17}{44 - 33} = 0.01338 \left( \frac{Volts}{Degree} \right) \]

\[ K_g = \frac{-1.37 - 0.01}{44 - 33} = -0.178 \left( \frac{Volts}{Degree} \right) \]

To determine the gain of the motor driver a similar process was used. The output of the motor driver is based on the duty cycle output of the micro controller. Table 4 shows the values recorded at different duty cycles and the corresponding output voltage when the driver is fed by a 13.5 Volt 3.2 Amp power supply.
Table 4: Measurements Taken to Determine Gain of the Motor Driver

<table>
<thead>
<tr>
<th>Duty Cycle (%)</th>
<th>13</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>87.5</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage (V)</td>
<td>1.365</td>
<td>2.625</td>
<td>5.25</td>
<td>7.875</td>
<td>9.1875</td>
<td>10.5</td>
</tr>
</tbody>
</table>

After evaluating these values the gain was found to be 0.105. Now after having found the values for these components a software model can be developed and tested. This model will now accurately represent the prototype.

Developing the Simulink Model

Before the model is created in Simulink it is important to know all of the values necessary for its operation. Table 5 shows all of the experimental values from the calculations and the given values from the datasheet for the motor.

Table 5: Constants Used for Simulation

<table>
<thead>
<tr>
<th>Description</th>
<th>Torque Constant (ounce*inches/amp)</th>
<th>Gear Ratio</th>
<th>Armature Resistance (Ω)</th>
<th>Armature Inductance (H)</th>
<th>Inertia (ounce*inches)</th>
<th>Friction</th>
<th>Back EMF Voltage (Volts)</th>
<th>Amplifier Gain</th>
<th>Accelerometer Gain</th>
<th>Gyroscope Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Td</td>
<td>n</td>
<td>Ra</td>
<td>La</td>
<td>Jeq</td>
<td>Beq</td>
<td>2E-09</td>
<td>0.0025</td>
<td>0.105</td>
<td>0.0134</td>
</tr>
<tr>
<td>Value</td>
<td>3.34</td>
<td>30.9</td>
<td>1.17</td>
<td>0.001</td>
<td>0.0008</td>
<td>2E-09</td>
<td>0.0025</td>
<td>0.105</td>
<td>0.0134</td>
<td>0.178</td>
</tr>
</tbody>
</table>

The next step in developing the model is being able to model individual components. The plant in this particular control system consists of the motor and platform frame. It is known that the motor is given some input voltage and this will result in some angular velocity Ω, which will induce an angular displacement θ. The system defined by the motor is shown in Figure 22.

Figure 22: Model of the Plant (Motor)

Then the output θ is sensed by the accelerometer and gyroscope. The voltage values sent by these sensors are passed to the controller of the system. This controller includes the process
used by the microcontroller and the motor driver. The decided design for the controller is a PID topology. This means proportionality, integration, and derivative control algorithms will be used. There will be proportionality constants for both the accelerometer and the gyroscope. The accelerometer signal will be controlled with the integration part of the control algorithm and the gyroscope will be controlled with the derivative. The derivation for these values is done using the PID control block in Simulink. This software plug-in is placed in the system where the control is desired. It then scans the system and develops gains for the proportional, integral, and derivative constants. The full system with the controller is shown in Figure 23. The tuned system parameters are then shown in Table 6.

![Figure 23: Full System with Controller](image)

<table>
<thead>
<tr>
<th>$K_P$</th>
<th>$K_I$</th>
<th>$K_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.0289665190436555</td>
<td>-0.258654492473789</td>
<td>0.000422783661267403</td>
</tr>
</tbody>
</table>

With the system built simulations were run to determine the impulse and step responses of the system. Before making any changes to the system, to ensure the software model is representative of the prototype, physical measurements were recorded and plotted in MATLAB. The following plots show the impulse and step responses of the system as well as the error of the system and integration error measurements. Note that in several figures the accelerometer, gyroscope, and complimentary filter are plotted. The complimentary filter is the value which the microcontroller uses for its computations and that is what will be analyzed. Also, note that the plots for positive and negative axes for all of the following figures were produced by separate trials. They are shown together to depict similarities.
Figure 24: Impulse Response of X Axis

Figure 24 depicts the impulse response for both the positive and negative x axis. At time 1030 ms the impulse is introduced to the positive x axis. It is at time 1210 ms that the system returns to steady state, where the offset of the angle is less than 5% of the maximum offset of 30°, which is 1.5°. Also, the settling time is given by this time interval, which is 180 ms and is within the settling time parameter. At time 1210 ms the impulse is introduced to the negative x axis. It is at time 1399 ms that the system returns to steady state. The settling time for this response is 189 ms. On the positive axis graph note some of the problems that were dealt with in the design process. In this plot it is observed that the accelerometer signal contains high noise components. Also, the gyroscope contains a drift offset. Although in this example the filter output remains within the steady state range there is a significant amount of error introduced by these characteristics of the components.
Figure 25 depicts the error observed by the controller. The error is produced by an impulse introduced to the positive x axis at 1030 and negative x axis at 1210. The errors recorded represent both in the total proportional, derivative, and integral portions of the error and specifically the integrator error as it is the most prevalent.
Figure 26 depicts the impulse response for both the positive and negative y axis. At time 854 ms the impulse is introduced to the positive y axis. It is at time 1206 ms that the system returns to steady state. Also, the settling time is given by this time interval, which is 352 ms. At time 534 ms the impulse is introduced to the negative y axis. It is at time 854 ms that the system returns to steady state. The settling time for this response is 320 ms.
Figure 27 depicts the error observed by the controller. The error is produced by an impulse introduced to the positive y axis at 854 and negative y axis at 534.
Figure 28 depicts the step response for both the positive and negative x axis. At time 622 ms the step is introduced to the positive x axis. It is at time 1082 ms that the system returns to steady state. Also, the settling time is given by this time interval, which is 460 ms. At time 3203 ms the step is introduced to the negative x axis. It is at time 3617 ms that the system returns to steady state. The settling time for this response is 414 ms.
Figure 29: Error on X Axis Produced by a Step

Figure 29 depicts the error observed by the controller. The error is produced by a step introduced to the positive x axis at 622 and negative x axis at 3203.
Figure 30 depicts the step response for both the positive and negative y axis. At time 623 ms the step is introduced to the positive y axis. It is at time 1199 ms that the system returns to steady state. Also, the settling time is given by this time interval, which is 576 ms. At time 1272 ms the step is introduced to the negative y axis. It is at time 1879 ms that the system returns to steady state. The settling time for this response is 607 ms.
Figure 31: Error on Y Axis Produced by a Step

Figure 31 depicts the error observed by the controller. The error is produced by an impulse introduced to the positive y axis at 623 and negative y axis at 1272.

Now with physical data the simulation is run. After confirming that the simulation is accurately representing the output model the simulation can be used to optimize the control system parameters. Note that it is assumed in the software model that the positive and negative axes, as well as the x and y axes, do not differentiate from each other. Because of this there is one plot for the impulse and step responses which can be applied to each axis.

Figure 32 shows the updated controlled closed loop system. Figure 33 and Figure 34 depict the adjusted system responses that represent and will be observed in the physical prototype.
After optimizing the simulation model the transfer functions that governed the controller have been modified to be represented with a simple proportional gain for the accelerometer and gyroscope. Also, the integration variable is held constant.

The figure shown above depicts the response of the system due to an impulse. The overshoot here is 0.6543°. This overshoot meets the requirements previously set forth. The system also meets the steady state requirements as it reaches ±5% of steady state before 200 ms. Note that there is no steady state error with these control parameters for impulse responses.
The figure shown above depicts the response of the system due to a step. The overshoot here is 0.62°. Again the overshoot is acceptable as well as the settling time settling time, which in this case is 112.5 ms. The steady state error due to a step response is minimal (i.e. less than 10% of the steady state tolerance of 1.5°.)

After running additional tests on the prototype the optimization is confirmed. With these updated system gains the prototype now meets both the settling time and overshoot parameters. It is now appropriate to proceed with testing of the prototype in its real world application.

All plots have been produced in Matlab and Simulink. Code for physical data plots is attached in Appendix 4 – MATLAB Scripts.
Section IV

Evaluation and Recommendations
**Control Design**

The step response for the X-axis has an overshoot of 0.654° which is a 9.8% overshoot. The Y-axis response is slightly better with the angular overshoot being 0.62°. This response meets the specifications and indicates it can further refined with additional testing.

Response settling time for an impulse input is 101.3 ms while a step input yields 112.5 ms. This is within the 250 ms requirement.

Steady state error for a step input results in 0.0783° for an allowed 1.5°. This equates to a 5% steady state error which is the requirement.

**Physical Parameters**

**Weight**

The weight of the platform is 6 pounds 6.5 ounces which exceeds the performance specifications. Using a thinner platform surface of 3/16” instead of 1/4” as well as optimizing the material used in the frame may reduce the overall weight to within the desired range without compromising the structural integrity and robustness of the system.

**Footprint**

The footprint is 544.5 cubic inches which exceeds the performance specifications of 500 cubic inches. There are limitations on the height based on the motor size and location; therefore, any footprint reduction would impact the size of the platform. The outer frame width may be reduced pending structural analysis which would bring the footprint within the design requirements. This is a design consideration relative to the specific vehicle where the device would be incorporated.

**Range of motion**

The prototype meets the requirements for the total 60° range of motion for each axes. Additional range is unlikely due to the location of the motors and mounting bracket configuration.

**Load capacity**

The device easily handles the 2 pound load capacity. The load capacity can be increased or the current motors may be replaced with a lower HP motor.

**Power consumption**

The device draws a maximum of 3.5 Amps when the power source must supply 13.5 V. This is well below the 15 Amp requirement.
Section V

Conclusions
This prototype addresses the design requirements for range of motion, load capacity, and power consumption. The controller overshoot is within the requirements as is the settling time in the specifications. The steady state error is 5% as required which does not allow adjustment of the steady state response. The prototype failed to meet the weight and footprint requirements, however and this should be taken into consideration before moving forward. Reducing the amount of material in the structural subsystem is likely solution for the weight. It is recommended that further structural analysis be conducted to produce a more efficient and lighter-weight unit. The footprint size cannot be easily reduced and as such, market research should be conducted to determine whether a reduction in the platform size is necessary.

Furthermore, the application of this solution requires specific conditions for operational use. The power supply may vary from vehicle to vehicle and effects the performance due to requirements for the motor driver. In addition, the position which the platform is mounted may also vary. This would require a separate calibration for each vehicle. In order for this solution to be universally applicable these issues must be considered and corrected. In conclusion the final design operates with acceptable performance and solves the initial problem of stabilizing objects within a motor vehicle but a marketable solution will require more time and resources to produce.
References

## Appendix 1 – Project Schedule

<table>
<thead>
<tr>
<th>Action</th>
<th>Owner</th>
<th>Comments</th>
<th>September</th>
<th>October</th>
<th>November</th>
<th>December</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototype fabrication and implementation</td>
<td></td>
<td></td>
<td>2</td>
<td>9</td>
<td>16</td>
<td>23</td>
</tr>
<tr>
<td>Material acquisition</td>
<td>JS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fabrication and component mounting</td>
<td>JS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Live demonstration</td>
<td>SS</td>
<td>Due November 19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programming</td>
<td></td>
<td></td>
<td>14</td>
<td>21</td>
<td>28</td>
<td>4</td>
</tr>
<tr>
<td>functional code for sensors</td>
<td>SS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>functional code for actuators</td>
<td>SS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>functional code for controller</td>
<td>SS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>interfacing</td>
<td>SS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>implementation of complementary filter</td>
<td>SS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulation and analysis</td>
<td></td>
<td></td>
<td>7</td>
<td>14</td>
<td>21</td>
<td>28</td>
</tr>
<tr>
<td>Simulink model with revised motor parameters</td>
<td>JK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysis of working prototype</td>
<td>JK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td>11</td>
<td>18</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>Identify testing parameters</td>
<td>GS</td>
<td>Due September 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop test procedures</td>
<td>GS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component functionality testing</td>
<td>JS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor calibration</td>
<td>GS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing of working prototype</td>
<td>GS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Troubleshooting/optimization</td>
<td>SS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final Presentation</td>
<td></td>
<td></td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>First Draft</td>
<td>GS</td>
<td>Due December 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Due December 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix 2 – Testing Procedures

Accelerometer Calibration Procedure

1. Initialize the Arduino microprocessor to power sensor, read sensor, output the readings to the serial port and disable the motors.
2. Level the platform relative to the desired axis using a digital level.
3. Take continuous readings from the Arduino microprocessor for approximately 2 seconds.
4. Rotate platform counterclockwise to 30°. Verify angle with digital level.
5. Take continuous readings from the Arduino microprocessor for approximately 2 seconds.
6. Rotate platform clockwise to 0°. Verify angle with digital level.
7. Take continuous readings from the Arduino microprocessor for approximately 2 seconds.
8. Rotate platform clockwise to 30°. Verify angle with digital level.
9. Take continuous readings from the Arduino microprocessor for approximately 2 seconds.
10. Rotate platform counterclockwise to 0°. Verify angle with digital level.
11. Take continuous readings from the Arduino microprocessor for approximately 2 seconds.
12. Calculate the bias-offset-correction factor by averaging the values at 0°.
13. A linear plot in MATLAB is used to determine accelerometer gain correction.
14. Repeat process for the axis of rotation.

Gyroscope Calibration Procedure

1. Initialize the Arduino microprocessor to power sensor, read sensor, output the readings to the serial port and disable the motors.
2. Power the gyroscope on and allow it to reach thermal stability.
3. Continuously record the output throughout the remainder of the procedure.
4. Hold the gyroscope against the first stop position (30° clockwise) for 5 seconds.
5. After 5 seconds turn the gyroscope toward the second stop. Begin with a smooth motion that takes approximately 4 seconds to move the 60° span.
6. Hold the gyroscope against the second stop position for 5 seconds.
7. Rotate the gyro back to the first stop using a similar motion.
8. Hold the gyroscope against the first stop position for 5 seconds.
9. Turn the gyroscope off.
10. Calculate the bias-offset-correction factor by averaging the first 3 seconds of data. The bias correction will be the opposite polarity of this average.
11. Subtract the bias estimate from the time record. Then integrate output data from the 4-second time stamp to the 10-second time stamp (before the start of movement until after
the platform stops). The scale factor is calculated by dividing the known angle by the measured angle calculated from the integration.

12. Using the bias-corrected response from Step 2, integrate output data from the 12-second time stamp to the 19-second time stamp.
13. Average the results of steps 2 and 3 to calculate the scale-factor correction.
14. Repeat process for the other axis of rotation.

Step Response Testing

1. Initialize microcontroller with stabilization program.
2. Turn on power supply and allow platform to auto-level.
3. Place the test object on platform frame specific to the testing axis.
4. Measure the response using the output generated by the microcontroller serial port.

Power Consumption Testing Procedure

1. Initialize microcontroller with stabilization program.
2. Attach multimeter leads to the wires supplying power to the motor responsible for actuation in the tested axis.
3. Turn on power supply and allow platform to auto-level.
4. Place the test object on platform frame specific to the testing axis.
5. Measure the response using the output generated by the microcontroller serial port.
Appendix 3 – Microcontroller Programming Code

#include "DualVNH5019MotorShield.h"
// include the libraGRY code for the gyro:
#include <Wire.h>
#include <L3G.h>

DualVNH5019MotorShield md;
L3G gyro;

//SC (dps/LSB): sensitivity
//The gyro has SC = 8.75 (mdps/digit), and R0 = ±10 for FS = 250 dps
// 8.72 / 250 * 1023 = 35.68224
float R0=10;
float SC=35.68224;
float GX,GY,GZ=0;

float GXX,GYY,GZZ=0;
float GRRX=0;
float Xout,Yout=0;

// Variables for Accelerometer
float AX;
float AY;
float Time=0;
float QX=0;
float QX_OLD=0;
float QX_OLD_OLD=0;
float SPDX=0;
float QY=0;
float QY_OLD=0;
float QY_OLD_OLD=0;
float SPDY=0;
float YDIR=0;
float GX0 = 0;
float GY0 = 0;

float AngleX=0;
float AngleY=0;

float DGX=0;
float DGY=0;

// Motor Driver
//Pin map
float dt=0;
float a=0;
float b=0;

//PID Variable
float InputX, OutputX;
float dErrX=0;
float errSumX=0;
float lastErrX=0;
float kpX=9;
float kiX=0.025;
float kdX=0;

float InputY, OutputY;
float dErrY=0;
float errSumY=0;
float lastErrY=0;
float kpY=11;
float kiY=0.03;
float kdY=0;

void setup() {

pinMode(A0,INPUT);
pinMode(A1,INPUT);

//Serial.begin(115200);

md.init();
Wire.begin();

if (!gyro.init())
{
    while (1);
}

//startup the Gyro in the default mode
gyro.enableDefault();

//take 100 sample and get the average to the gyro offset and accelerometer offset to calibrate
for(int i=0; i<100; i++)
{
    gyro.read();
}
GX = (float)gyro.g.x;
GY = (float)gyro.g.y;
GX0=GX0+GX;
GY0=GY0+GY;
}
GX0=GX0/100;
GY0=GY0/100;

AX = analogRead(A3);
QX_OLD_OLD = (-0.3571*AX + 116.63);
AX = analogRead(A3);
QX_OLD = (-0.3571*AX + 116.63);
DGX=QX_OLD;
AY = analogRead(A2);
QY_OLD_OLD = (-0.3352*AY + 116.88);
AY = analogRead(A2);
QY_OLD = (-0.3352*AY + 116.88);
DGY=QY_OLD;
AngleX= QX_OLD;
AngleY= QY_OLD;
}

void loop() {
  //super-loop
  //Acc
  //Linear equation and subtracting the error angle (7)
  AX = analogRead(A3);
  QX = (-0.3571*AX + 116.63)-3;
  QX=(QX+QX_OLD+QX_OLD_OLD)/3;
  QX_OLD_OLD=QX_OLD;
  QX_OLD=QX;
  //Linear equation and subtracting the error angle (10)
  AY = analogRead(A2);
  QY = (-0.3352*AY + 116.88);
  QY=(QY+QY_OLD+QY_OLD_OLD)/3;
  QY_OLD_OLD=QY_OLD;
  QY_OLD=QY;
  //Gyro
  gyro.read();
  GX = (float)gyro.g.x;
  GY = (float)gyro.g.y;
//this need to be commented out
Time=Time+1;
//subtracting the zero rate from the last gyro rate reading and then converting the digital number by multiplying by the gyro degree per digit (0.000875 for +-250 sensitivity
GXX=0.000875*(GX-GX0);
GYY=0.000875*(GY-GY0);

// reman sum; dt=0.0519 sampling period
DGX = DGX + (0.01 * dt * GXX);
DGY = DGY + (0.01 * dt * GYY);

b=millis();
dt=b-a;

// Complementary Filter
//dt=0.067; note that 0.98 + 0.02 = 1
AngleX= (0.98*(AngleX+(0.01 * dt * GXX))) + ((0.02)*(QX));
AngleY= (0.98*(AngleY+(0.01 * dt * GYY))) + ((0.02)*(QY));

//PID Controller Code
errSumX += (AngleX * dt);
dErrX = (AngleX - lastErrX) / dt;
OutputX = kpX * AngleX + kiX * errSumX + kdX * dErrX;
lastErrX = AngleX;

errSumY += (AngleY * dt);
dErrY = (AngleY - lastErrY) / dt;
OutputY = kpY * AngleY + kiY * errSumY + kdY * dErrY;
lastErrY = AngleY;

//This is important to insure that the output value never exceeds the 400 limit
if(OutputX>399)
{
    OutputX=399;
    lastErrX=399;
    errSumX=0;
}
if(OutputX<-399)
{
    OutputX=-399;
    lastErrX=-399;
    errSumX=0;
}
if(OutputY>399)
{
    OutputY=399;
    lastErrY=399;
    errSumY=0;
}
if(OutputY<-399)
{
    OutputY=-399;
    lastErrY=-399;
    errSumY=0;
}

a=millis();

// Movement Code
// the md.setM1Speed(0-400) maps the to 255 by multiplying the value by (255/400) which is
// the mapping factor that the functions uses
// the mapped value in the range 0-255 which is % duty cycle

md.setM1Speed(OutputX);
md.setM2Speed(OutputY);
Appendix 4 – MATLAB Scripts

% Jacob Kryder
% ECE406 Senior Design Project
% Stabilization Platform

clear

% loads recorded movement data
movement_file='H:\Program Files\MATLAB\angle_data.txt'; % assigns load variable
movement_data=load(movement_file); % assigns total data array
time_M=movement_data(:,1); % time vector for position data
ax=movement_data(:,2); % accelerometer x axis position data
gx=movement_data(:,3); % gyroscope x axis position data
cx=movement_data(:,4); % complimentery filter x axis position data
ay=movement_data(:,5); % accelerometer y axis position data
gy=movement_data(:,6); % gyroscope y axis position data
cy=movement_data(:,7); % complimentery filter y axis position data

% plots movement data
figure(1)
subplot(2,1,1)
plot(time_M,ax,time_M,gx,time_M,cx)
title('Movement Data for X Axis');
ylabel('Offset Angle (degrees)');
xlabel('Time (ms)');
legend('Accelerometer','Gycroscope','Complimentery Filter');
subplot(2,1,2)
plot(time_M,ay,time_M,gy,time_M,cy)
title('Movement Data for Y Axis');
ylabel('Offset Angle (degrees)');
xlabel('Time (ms)');
legend('Accelerometer','Gycroscope','Complimentery Filter');

% loads recorded x axis impulses response data
x_impulse_response_file='H:\Program Files\MATLAB\x_impulse_response_data.txt'; % assigns load variable
x_impulse_response_data=load(x_impulse_response_file); % assigns total data array

time_IRX=x_impulse_response_data(:,1); % time vector for x axis impulse response
iaxp=x_impulse_response_data(:,2); % accelerometer positive x axis impulse response
igxp=x_impulse_response_data(:,3); % gyroscope positive x axis impulse response
icxp=x_impulse_response_data(:,4); % complimentery filter positive x axis impulse response
i\text{xn}=\text{x\_impulse\_response\_data(:,5)}; \quad \%\text{accelerometer negative x axis impulse response}
igx=x\_impulse\_response\_data(:,6); \quad \%\text{gyroscope negative x axis impulse response}
icx=x\_impulse\_response\_data(:,7); \quad \%\text{complimentery filter negative x axis impulse response}
ixp=x\_impulse\_response\_data(:,8); \quad \%\text{error on positive x axis impulse response}
imxp=max(ixpe); \quad \%\text{max error on positive x axis impulse response}
ixpie=x\_impulse\_response\_data(:,9); \quad \%\text{integral error on positive x axis impulse response}
imxpe=max(ixpe); \quad \%\text{max integral error on positive x axis impulse response}
ixpm=x\_impulse\_response\_data(:,10); \quad \%\text{positive x axis output to motor driver}
ixn=x\_impulse\_response\_data(:,11); \quad \%\text{error on negative x axis impulse response}
imxn=max(ixne); \quad \%\text{max error on negative x axis impulse response}
ixnie=x\_impulse\_response\_data(:,12); \quad \%\text{integral error on negative x axis impulse response}
imxn=max(ixnie); \quad \%\text{max integral error on negative x axis impulse response}
ixnmd=x\_impulse\_response\_data(:,13); \quad \%\text{negative x axis output to motor driver}

%for each of the following sections of code the variables are the same as in the the preceeding with the only changes being in the x or y axis and the impulse or step response.

%plots x axis impulse response
figure;
subplot(2,1,1)
plot(time\_IRX,i\text{xp},time\_IRX,igxp,time\_IRX,icxp)
title('Impulse Response of Positive X Axis');
ylabel('Offset Angle (degrees)');
xlabel('Time (ms)');
legend('Accelerometer','Gycroscope','Complimentery Filter');

subplot(2,1,2)
plot(time\_IRX,i\text{xn},time\_IRX,igxn,time\_IRX,icxn)
title('Impulse Response of Negative X Axis');
ylabel('Offset Angle (degrees)');
xlabel('Time (ms)');
legend('Accelerometer','Gycroscope','Complimentery Filter');

figure;
subplot(2,1,1)
plot(time\_IRX,(ix\text{p}e/im\text{xpe}),time\_IRX,(ix\text{p}ie/im\text{xpie}))
title('Error for Positive X Axis from Impulse');
ylabel('Error (%)');
xlabel('Time (ms)');
legend('Total Error','Integrator Error');

subplot(2,1,2)
plot(time\_IRX,(ix\text{ne}/im\text{xne}),time\_IRX,(ix\text{nie}/im\text{xnie}))
title('Error for Negative X Axis from Impulse');
ylabel('Error (%)');
xlabel('Time (ms)');
legend('Total Error','Integrator Error');
figure;
subplot(2,1,1)
plot(time_IRX,ixpmd)
title('Microprocessor Output to Motor Driver for Positive X Axis from Impulse Response');
ylabel('Output PWM (0-400)');
xlabel('Time (ms)');
subplot(2,1,2)
plot(time_IRX,ixnmd)
title('Microprocessor Output to Motor Driver for Negative X Axis from Impulse Response');
ylabel('Output PWM (0-400)');
xlabel('Time (ms)');

loads recorded y axis impulse response data
y_impulse_response_file='H:\Program Files\MATLAB\y_impulse_response_data.txt';
y_impulse_response_data=load(y_impulse_response_file);
time_IRY=y_impulse_response_data(:,1);
iayp=y_impulse_response_data(:,2);
igyp=y_impulse_response_data(:,3);
icyp=y_impulse_response_data(:,4);
iayn=y_impulse_response_data(:,5);
igyn=y_impulse_response_data(:,6);
icyn=y_impulse_response_data(:,7);
iype=y_impulse_response_data(:,8);
imype=max(iype);
iypie=y_impulse_response_data(:,9);
imypie=max(iypie);
iypmd=y_impulse_response_data(:,10);
iyne=y_impulse_response_data(:,11);
imyne=max(iyne);
iynie=y_impulse_response_data(:,12);
imynie=max(iynie);
iynmd=y_impulse_response_data(:,13);

plots y axis impulse response
figure;
subplot(2,1,1)
plot(time_IRY,iayp,time_IRY,igyp,time_IRY,icyp)
title('Impulse Response of Positive Y Axis');
ylabel('Offset Angle (degrees)');
xlabel('Time (ms)');
legend('Accelerometer','Gyroscope','Complimentery Filter');
subplot(2,1,2)
plot(time_IRY,iayn,time_IRY,igyn,time_IRY,icyn)
title('Impulse Response of Negative Y Axis');
ylabel('Offset Angle (degrees)');
xlabel('Time (ms)');
legend('Accelerometer','Gyroscope','Complimentery Filter');

figure;
subplot(2,1,1)
plot(time_IRY,iype/imype,time_IRY,(iypie/imypie))
title('Error for Positive Y Axis from Impulse');
ylabel('Error (%)');
xlabel('Time (ms)');
legend('Total Error','Integrator Error');
subplot(2,1,2)
plot(time_IRY,(iyne/imyne),time_IRY,(iynie/imynie))
title('Error for Negative Y Axis from Impulse');
ylabel('Error (%)');
xlabel('Time (ms)');
legend('Total Error','Integrator Error');

figure;
subplot(2,1,1)
plot(time_IRY,iypmd)
title('Microprocessor Output to Motor Driver for Positive Y Axis from Impulse Response');
ylabel('Output PWM (0-400)');
xlabel('Time (ms)');
subplot(2,1,2)
plot(time_IRY,iynmd)
title('Microprocessor Output to Motor Driver for Negative Y Axis from Impulse Response');
ylabel('Output PWM (0-400)');
xlabel('Time (ms)');

loads recorded x axis step response data
x_step_response_file='H:\Program Files\MATLAB\x_step_response_data.txt';
x_step_response_data=load(x_step_response_file);
time_SRX=x_step_response_data(:,1);
saxp=x_step_response_data(:,2);
saxn=x_step_response_data(:,5);
sxpe=x_step_response_data(:,8);
sxpmd=max(sxpe);
sxpie=max(sxpmd);
sxpmd=x_step_response_data(:,10);
sxne=max(sxpie);
sxnie=x_step_response_data(:,12);
smxne=max(sxnie);
sxned=x_step_response_data(:,13);

%plots x axis step response
figure;
subplot(2,1,1)
plot(time_SRX,saxp,time_SRX,sgxp,time_SRX,scxp)
title('Step Response of Positive X Axis');
ylabel('Offset Angle (degrees)');
xlabel('Time (ms)');
legend('Accelerometer','Gyroscope','Complimentery Filter');
subplot(2,1,2)
plot(time_SRX,saxn,time_SRX,sgxn,time_SRX,scxn)
title('Step Response of Negative X Axis');
ylabel('Offset Angle (degrees)');
xlabel('Time (ms)');
legend('Accelerometer','Gyroscope','Complimentery Filter');

figure;
subplot(2,1,1)
plot(time_SRX,(sxpe/smxpe),time_SRX,(sxpie/smxpie))
title('Error for Positive X Axis from Step Response');
ylabel('Error (%)');
xlabel('Time (ms)');
legend('Total Error','Integrator Error');
subplot(2,1,2)
plot(time_SRX,(sxne/smxne),time_SRX,(sxnie/smxnie))
title('Error for Negative X Axis from Step Response');
ylabel('Error (%)');
xlabel('Time (ms)');
legend('Total Error','Integrator Error');

figure;
subplot(2,1,1)
plot(time_SRX,sxpmd)
title('Microprocessor Output to Motor Driver for Positive X Axis from Step Response');
ylabel('Output PWM (0-400)');
xlabel('Time (ms)');
subplot(2,1,2)
plot(time_SRX,sxnmd)
title('Microprocessor Output to Motor Driver for Negative X Axis from Step Response');
ylabel('Output PWM (0-400)');
xlabel('Time (ms)');

%loads recorded y axis step response data
y_step_response_file='H:\Program Files\MATLAB\y_step_response_data.txt';
y_step_response_data=load(y_step_response_file);
time_SRY=y_step_response_data(:,1);
sayp=y_step_response_data(:,2);
sgyp=y_step_response_data(:,3);
scyp=y_step_response_data(:,4);
sayn=y_step_response_data(:,5);
sgyn=y_step_response_data(:,6);
scyn=y_step_response_data(:,7);
sype=y_step_response_data(:,8);
smype=max(sype);
sypie=y_step_response_data(:,9);
smypie=max(sypie);
sypmd=y_step_response_data(:,10);
syne=y_step_response_data(:,11);
smyne=max(syne);
synie=y_step_response_data(:,12);
smyne=max(synie);
synmd=y_step_response_data(:,13);

%plots y axis step response
figure;
subplot(2,1,1)
plot(time_SRY,sayp,time_SRY,sgyp,time_SRY,scyp)
title('Step Response of Positive Y Axis');
ylabel('Offset Angle (degrees)');
xlabel('Time (ms)');
legend('Accelerometer','Gyroscope','Complimentery Filter');
subplot(2,1,2)
plot(time_SRY,sayn,time_SRY,sgyn,time_SRY,scyn)
title('Step Response of Negative Y Axis');
ylabel('Offset Angle (degrees)');
xlabel('Time (ms)');
legend('Accelerometer','Gyroscope','Complimentery Filter');

figure;
subplot(2,1,1)
plot(time_SRY,(sype/smype),time_SRY,(sypie/smypie))
title('Error for Positive Y Axis from Step Response');
ylabel('Error (%)');
xlabel('Time (ms)');
legend('Total Error','Integrator Error');
subplot(2,1,2)
plot(time_SRY,(syne/smyne),time_SRY,(synie/smynie))
title('Error for Negative Y Axis from Step Response');
ylabel('Error (%)');
xlabel('Time (ms)');
legend('Total Error','Integrator Error');

figure;
subplot(2,1,1)
plot(time_SRY,sypmd)
title('Microprocessor Output to Motor Driver for Positive Y Axis from Step Response');
ylabel('Output PWM (0-400)');
xlabel('Time (ms)');
subplot(2,1,2)
plot(time_SRY,synmd)
title('Microprocessor Output to Motor Driver for Negative Y Axis from Step Response');
ylabel('Output PWM (0-400)');
xlabel('Time (ms)');