DESIGN AND DEMONSTRATION OF AN INERTIAL-STABILIZED SINGLE-AXIS HEADLAMP FOR MOBILE SYSTEMS

by

Thomas Avery Allsup

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy at
The University of Texas at Arlington
December, 2023

Arlington, Texas

ACKNOWLEDGEMENTS

The author wishes to thank Dr. Robert L. Woods for his many years of guidance and support. The author would like to thank the entire staff of UTA MAE department who all greatly assisted a returning aged student. A special thank you to Dr. Kamesh Subbarao for assisting in an advanced Matlab programming task. Further, the author would like to recognize and thank Drs. Raul Fernandez, David Hullender, David Hunn, and David Wetz for serving on my committee.

November 30, 2023

DEDICATION

To my family, both blood and found, I dedicate this dissertation work. I am thankful for all their continued support. This was a real team effort.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES, PROGRAM CODES, AND LISTS

ABSTRACT

DESIGN AND DEMONSTRATION OF AN INERTIAL-STABILIZED
SINGLE-AXIS HEADLAMP FOR MOBILE SYSTEMS

Thomas Avery Allsup, Ph.D.

The University of Texas at Arlington, 2023

Supervising Professor: Dr. Robert L. Woods

The introduction of adaptive headlamps has added left and right movement of the headlamps to enhance the vision of the driver during a turn at night. Adaptive headlamps also allow up and down motion of headlamps in either two discrete positions with low and high beams based on vehicle speed and oncoming traffic lights or using the vehicle suspension measurement to adjust headlamp beam angles in any angle between low and high. Using the vehicle suspension ignores the roadway influence on where the headlamps should be positioned to illuminate the roadway for the driver. This dissertation develops governing equations and provides an electro-mechanical breadboard design for a single-axis servo system that provides inertial stabilization of headlamp angles such that the headlights are approximately always at the point in space with respect to inertial space. Not only does this keep the headlights level to optimally illuminate the road ahead for the driver, but it also prevents the headlights from shining in the eyes of the oncoming traffic. The breadboard design includes selection of the angular positioning sensor, mechanical embodiment, electrical circuitry and control program. Included simulation and demonstration of the breadboard design provides the foundation for future prototyping of vehicle systems.

CHAPTER 1

INTRODUCTION

Adaptive headlamp systems started 100 years ago with left and right angular headlamp movements corresponding to the mechanical linkage attached to the steering wheel and have now progressed to electronic sensing oncoming traffic's headlamps to control high and low beams based on vehicle speed.

European countries currently allow vertical headlamp leveling that use sensors on passenger vehicle suspensions to determine the angular pitch position (see Figure 1) [1]. These current headlamp systems only partially reference the pitch of the chassis and do not account for the roadway pitch [2]. United States has been slow to adopt vertical headlamp leveling except for low and high beams but it is currently allowed. If the vehicle is traveling along a roadway with little or no roadway angular pitch rate change then the headlamp attached to the chassis is acceptable. The change of the angular pitch rate of the roadway is negligible when the car is traveling on a flat roadway or is driving up or down a long straight incline. This situation is not ideal for roadways with changes in pitch, think short bumps, small hills, or train tracks, since the headlamps will follow the vehicle path which moves the lights up and down.

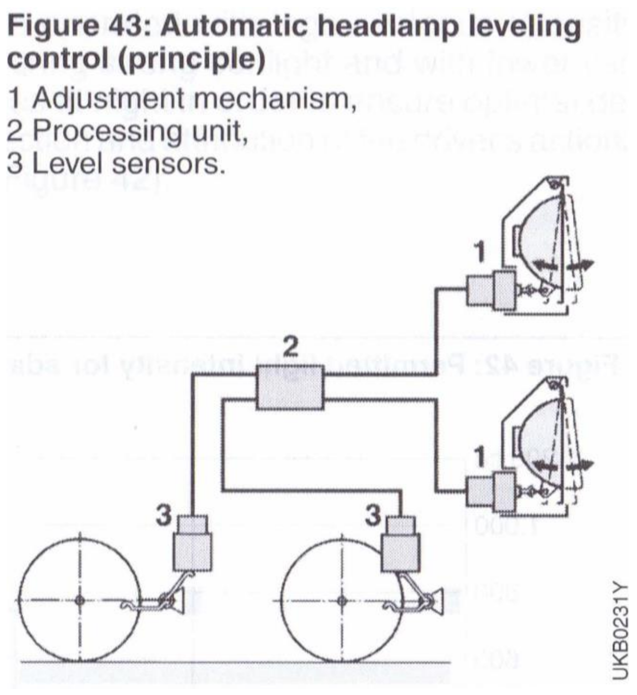


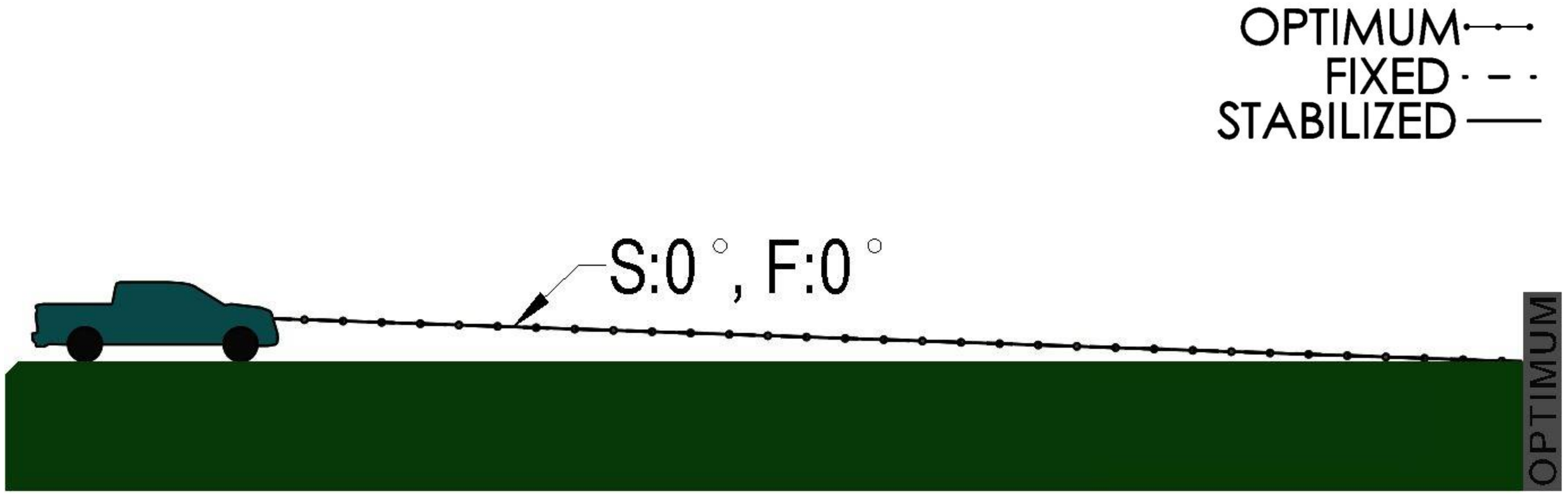
Figure 1 Automatic Headlamp Leveling Control [1]

This improvement of headlamp position can be further improved if an element of inertial stabilization is added to the pitch of the chassis headlamp leveling when the angular pitch of the road is variable. To further illustrate this point, an animation was created in Onshape to explain of concept of leveling versus inertial stabilization (see Appendix 1). Onshape is a cloud-based 3D parametric CAD program that provides design and drafting tools for mechanical designers. This animation shows that the headlamp following the vehicle path is not optimum for visibility.

Another method of conveying the optimum headlamp angle is to place a known vehicle traveling at a constant velocity on various roadway surfaces and display the optimum headlamp angle for braking distance, the fixed headlamp angle, and the inertially stabilized headlamp angle. Figure 2 shows various headlamp angle scenarios for a sample vehicle dimension and driving parameters described in Appendix 8. The dotted line shows the optimum beam angle to show light at the current braking distance at 45 MPH if possible. This point along the roadway is shown in the figures with the vertical bar indicator and the word “OPTIMUM”. This bar is not along the chord length of the road but located at the horizontal braking distance from front of the vehicle because the models are easier to create and the chord length is within 0.3% of the horizontal distance. There are two scenarios where the curvature of the road makes illumination at the optimum distance impossible. The dashed line indicates the headlamp fixed to the vehicle that follows only the pitch of the vehicle. The solid line is the inertially stabilized headlamp angle. Table 1 summarizes the various scenarios. The two difference columns each show the angle between either the stabilized or the fixed headlamp angle and the optimum angle. The minimum absolute difference would be the best angle choice. It is clear from these eleven examples that the inertially stabilized headlamp is often the minimal value but in the four scenarios where the inertially stabilized headlamp wasn't the best choice it was off by less than a degree.

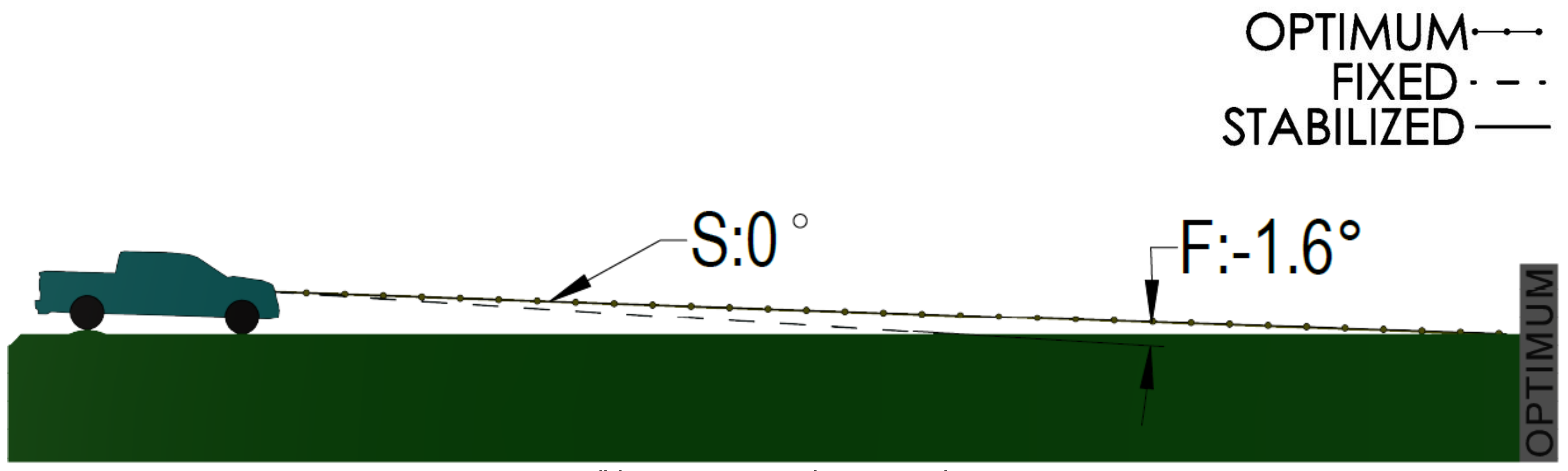
Table 1 Summary of Figure 2 Headlamp Angle Scenarios at 45 MPH

Figure	**Description**	**Best Angle Choice**	**Stabilized Angle Difference (Degrees)**	**Fixed Angle Difference (Degrees)**
(a)	Level Ground	Same	0	0
(b)	Bump Under Rear Tire	Stabilized	0	-1.60
(c)	Bump Under Front Tire	Stabilized	0	+1.60
(d)	Slight Decline	Fixed	+0.85	+0.72
(e)	Large Decline	Fixed	+4.63	+3.84
(f)	Slight Rise	Fixed	-1.00	-0.84
(g)	Large Rise	Fixed	-5.07	-4.30
(h)	Slight Crest	Stabilized	+0.44	+2.96
(i)	Large Crest	Stabilized	-2.86	+10.52
(j)	Slight Dip	Stabilized	-0.42	-3.00
(k)	Large Dip	Stabilized	-2.12	-16.17

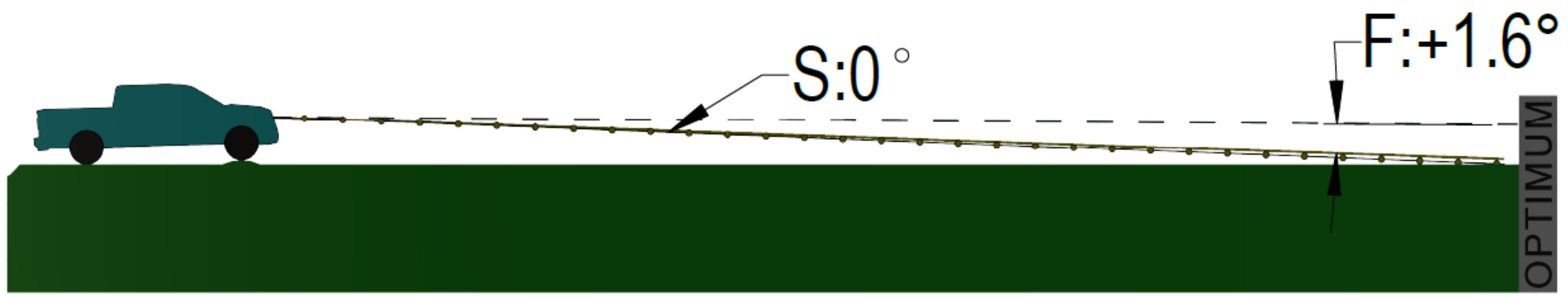


(a) Level Ground

Figure 2 Headlamp Angle Scenarios at 45 MPH

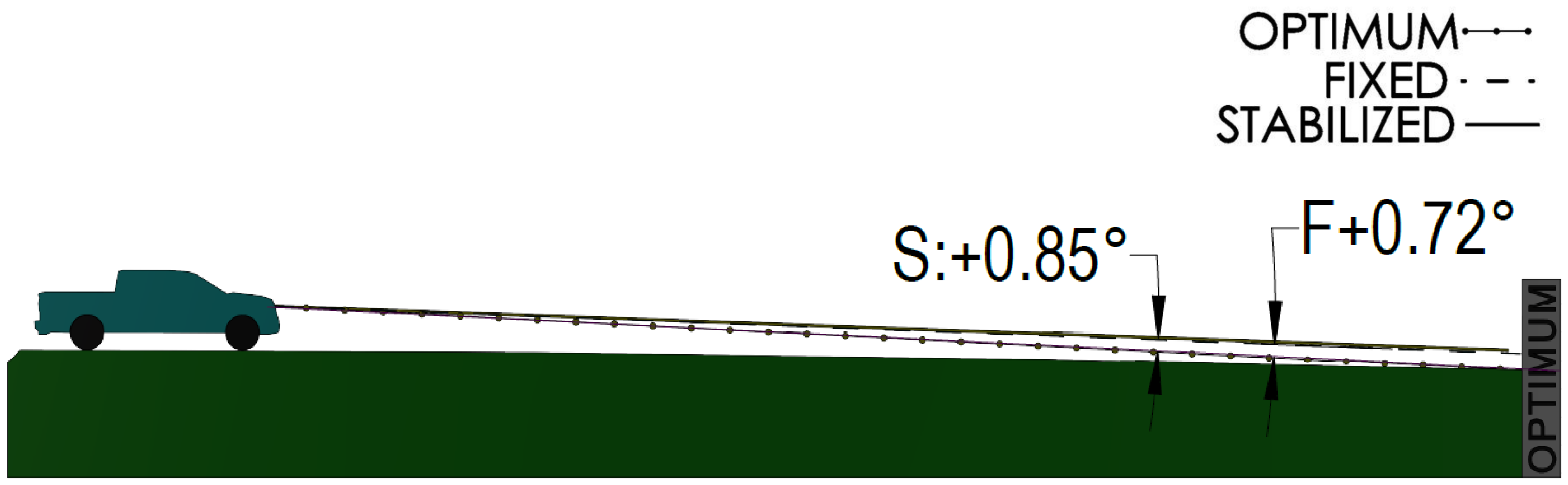


(b) Bump Under Rear Tire

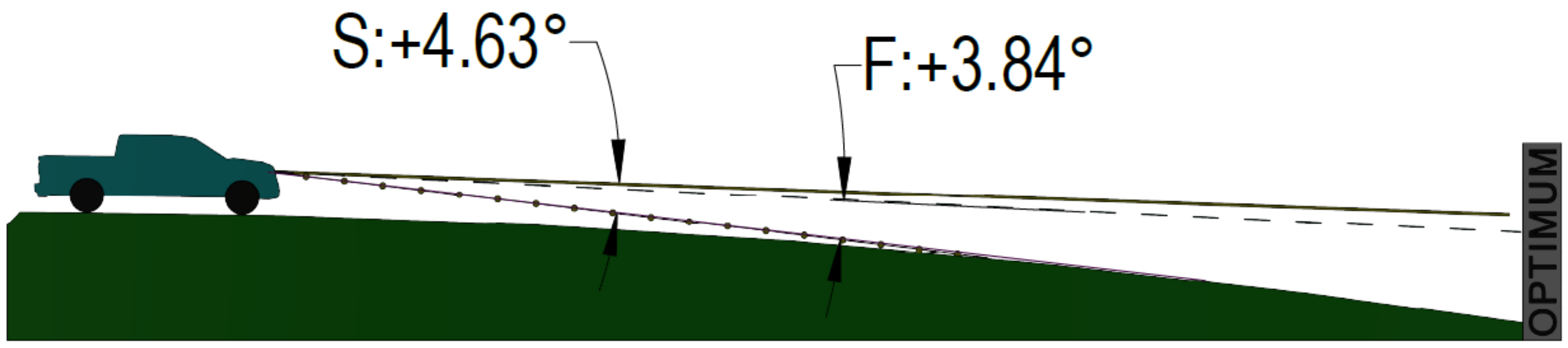


(c) Bump Under Front Tire

Figure 2 Headlamp Angle Scenarios at 45 MPH (Continued)

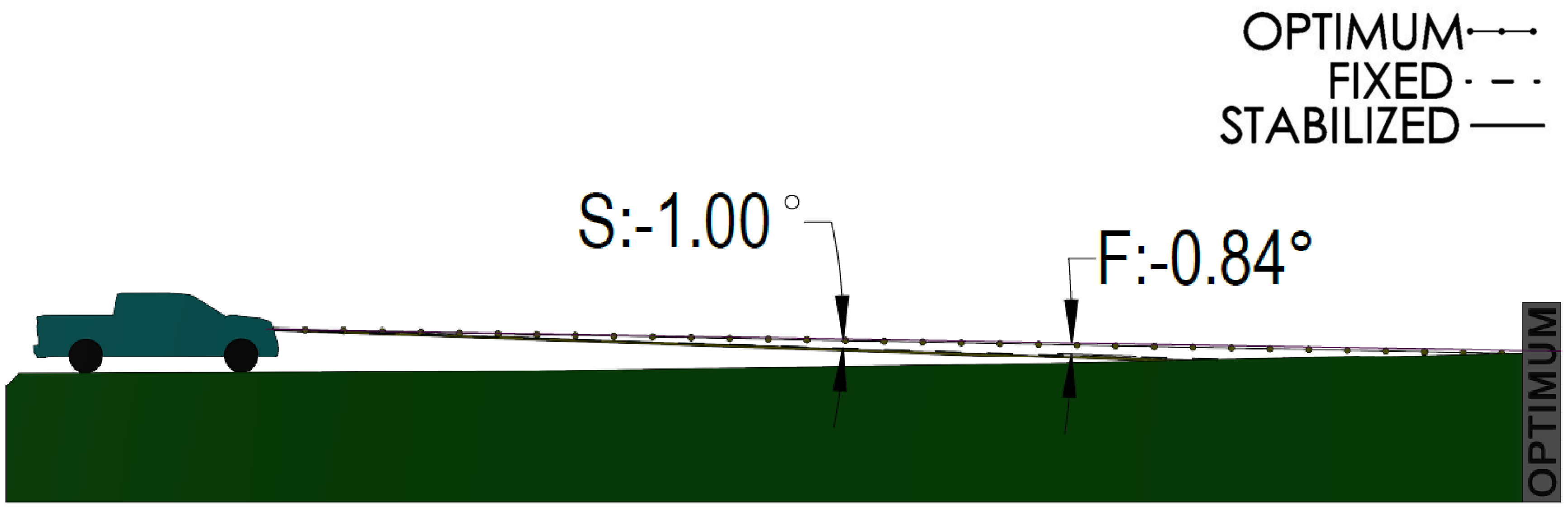


(d) Slight Decline

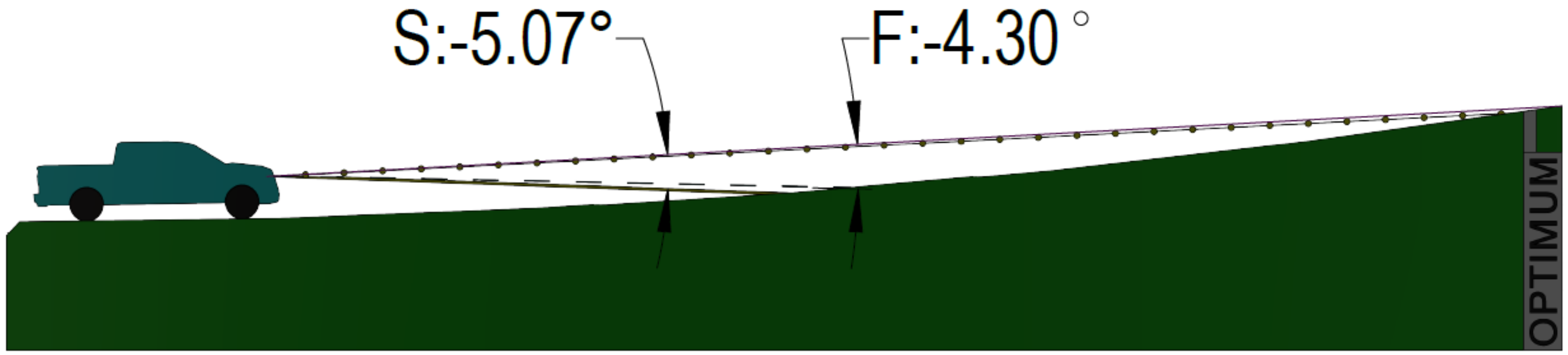


(e) Large Decline

Figure 2 Headlamp Angle Scenarios at 45 MPH (Continued)

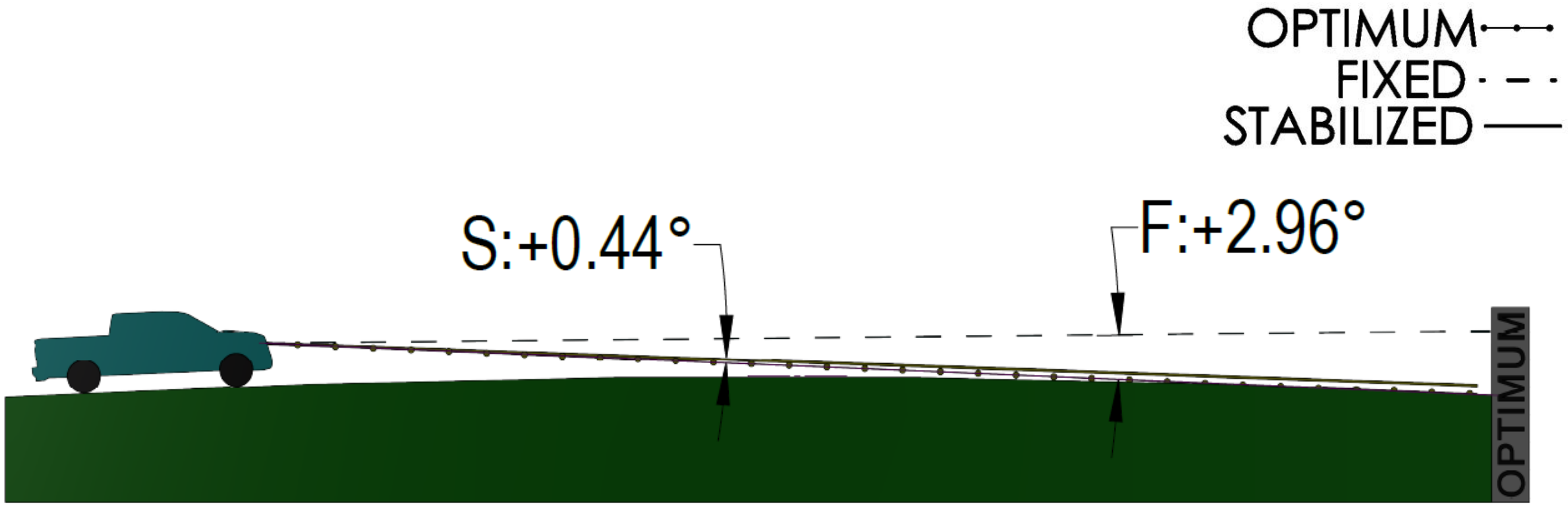


(f) Slight Rise

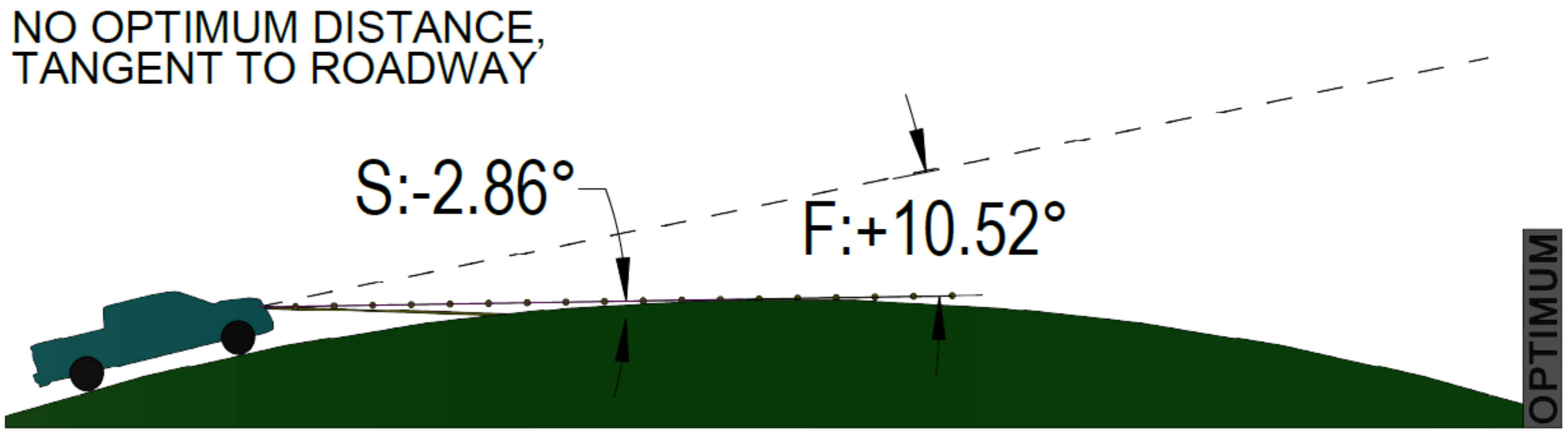


(g) Large Rise

Figure 2 Headlamp Angle Scenarios at 45 MPH (Continued)

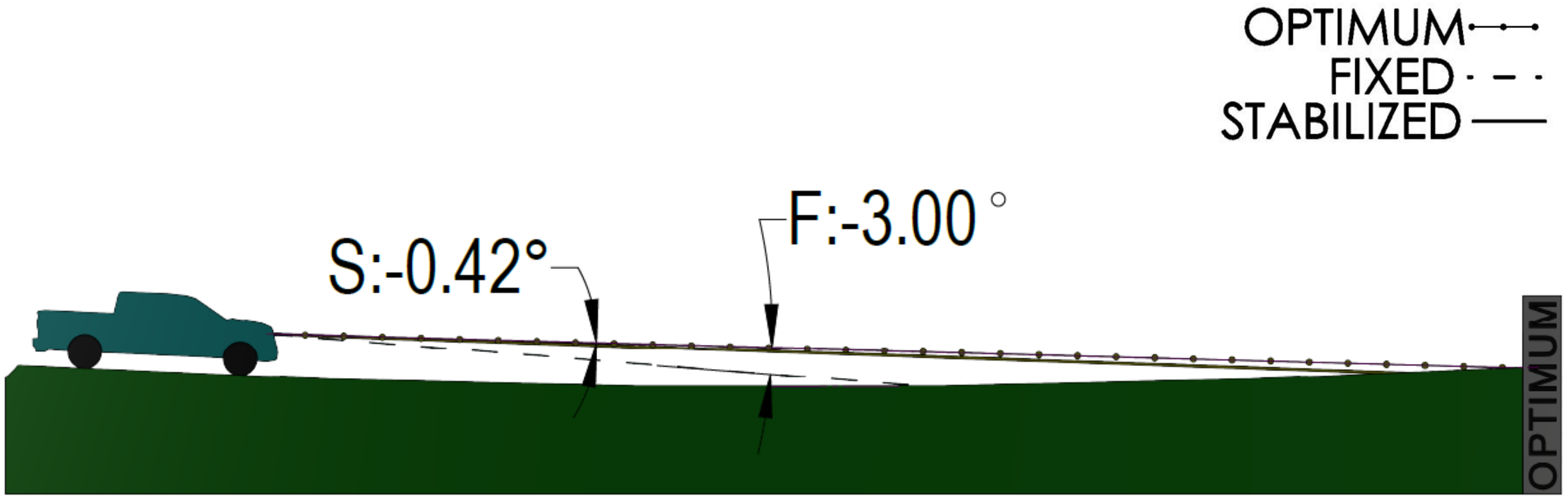


(h) Slight Crest

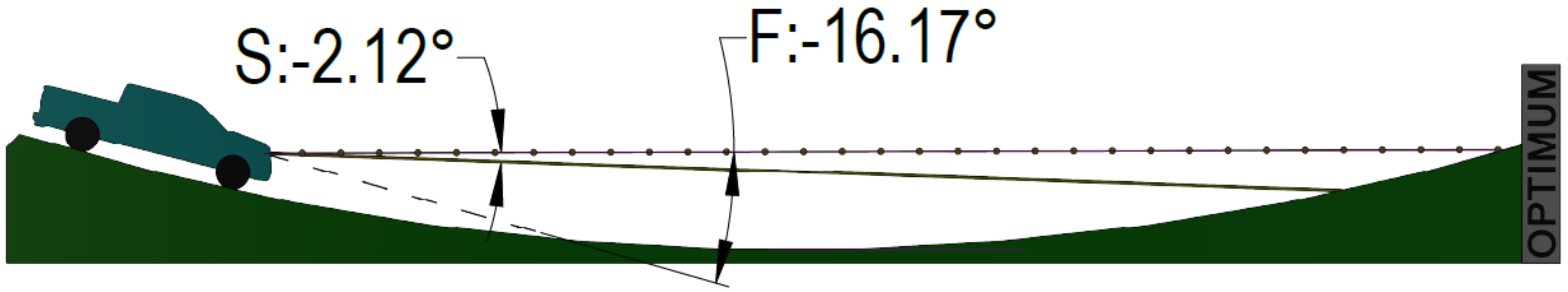


(i) Large Crest

Figure 2 Headlamp Angle Scenarios at 45 MPH (Continued)



(j) Slight Dip



(k) Large Dip

Figure 2 Headlamp Angle Scenarios at 45 MPH (Continued)

The purpose of headlamps is to avoid oncoming obstacles at night. You can avoid these obstacles by turning but the primary method is to stop the vehicle. The braking distance is a function of several factors including the reaction time of the driver, the roadway conditions, the speed of the vehicle, and the braking system employed. The angle of the headlamps can decrease the reaction time of the driver by allowing the obstacle to be observed sooner. This is known colloquially as “don’t overdrive your headlights”. Appendix 8 provides a dimensional analysis of this braking distance and the resulting optimum angle of the headlamps. The need for longer braking distances at higher speeds is the reason that highway speed vehicles have low and high beam settings [3].

A combination of headlamp leveling, headlamp inertial stabilization, and vehicle speed will therefore provide the optimum headlamp angular pitch positions based on all these conditions dynamically. This improved headlamp angle will improve driver safety while reducing the blinding or “dazzling” of oncoming vehicle drivers. The vision of the driver upcoming path based on your speed using a range of beam angles instead of just two settings.

Finally, adding leveling and inertial stabilization of the headlamps will make up for the loss of traditional headlamp adjustments during vehicle inspections that has occurred in the past.

CHAPTER 2

RESEARCH

The research for headlamp leveling and inertial stabilization has very little published works on vehicle inertial stabilization [1][2][3].

Three commercially available angular rate sensors were explored. The InvenSense MPU-6050 was easy to connect to electrically but only provided angular rates not position and with noise that was comparable to the signal [4]. The Analog Devices ADXRS800 was also easy to connect to electrically but also only provided angular rate but with a little less noise [5]. Attempts to understand and filter this noise only continued to expose these as flawed sensors. Finally, the Bosch BNO055 was investigated (see Figure 3) as it provides angular pitch positions with almost zero electrical noise along with pitch rate [6]. The Bosch BNO055 has nine internal single-axis sensors so it is capable of providing an Euler's angle output which is an inertial positional angle.

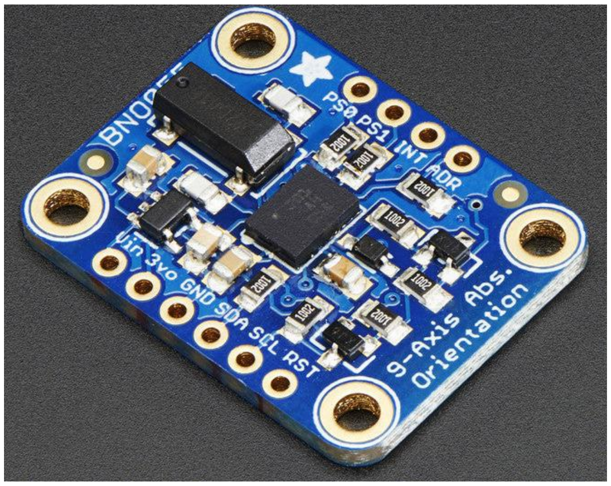


Figure 3 Bosch BNO055 Nine Axis Sensor [6]

After the comprehensive exam but before the completion of the final draft of this dissertation, a team of Korean engineers from Konkuk University and Hyundai published an IEEE research paper describing an inertial sensor-based headlamp angle using road to vehicle pitch [7]. The team used the MPU-6050 inertial sensor with a Kalman filter along with a vehicle wheel speed sensor. The angular acceleration is integrated to calculate the angular pitch. The vehicle linear acceleration and deceleration are also integrated along with the noise which creates a steadily increasing pitch angle. The team uses the vehicle wheel speed sensor to determine when the vehicle is stopped so the integration sum can be reset to zero. The system was simulated and prototyped successfully in a city scenario. They did not test the device over a long-haul trip. The authors described future work that would remove the vehicle wheel speed sensor.

Three adaptive headlamps were dissected that provided left and right headlamp positioning. It was obvious from the design of the motor drivers for all three headlamp that it would easily be changed to provide up and down servo control as well as left and right because the connection was a three dimensionally ball joint. These ball joints allowed the motor servo to move the headlamp left and right in either the low or high beam position. Photographs from one of the headlamp dissections is shown in Appendix 3.

A stationary fixture based on a sine plate fixture for accurate static angle measurements was created (see Figure 4). A series of gauge blocks, shown in the picture around the sine plate, were inserted in the sine plate fixture to achieve very precise angles. The angular pitch sensor is mounted on a machined white plastic base and placed on top of the sine plate fixture. The angular pitch sensor can be rotated to achieve negative angles or positive angles. The picture shows the Raspberry Pi housing in the background that is recording the information.

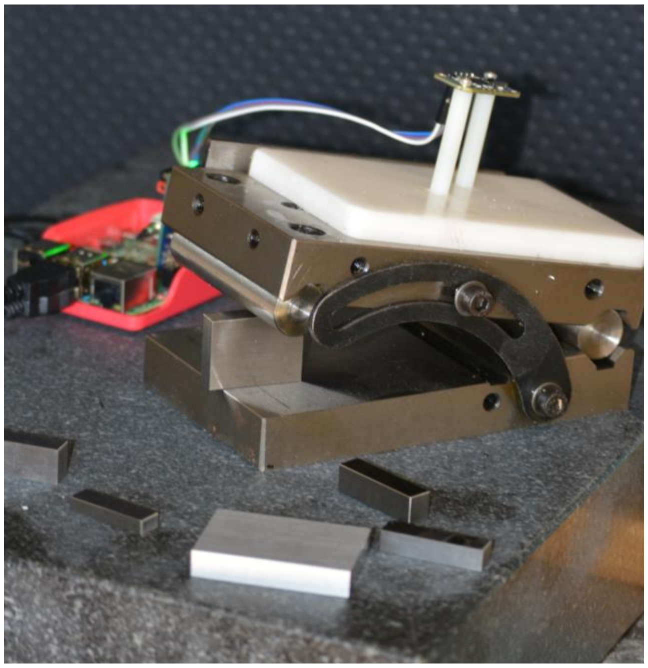
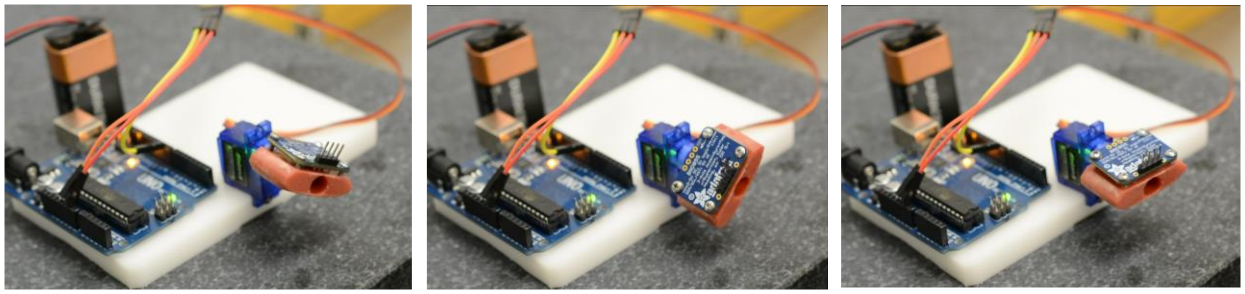


Figure 4 Static Sine Plate Angle Measurement

A slow speed sine wave rotating fixture to determine the noise and accuracy of the sensor was also fabricated (see Figure 5). The picture shows the Arduino controller that drives a small Remote Control vehicle DC servo. The Arduino and servo are both mounted to a white plastic base. The servo has a 3D printed plate that allows for mounting the angular pitch sensor. The wires connecting to the Raspberry Pi data collection are not shown for clarity.



(a) Rotating CCW (b) Rotating CW (c) Middle Position

Figure 5 Dynamic Triangular Wave Angle Measurement

A sample set of actual test data at increasing frequencies using a triangular pattern shows very little noise (see Figure 6). This figure does show the data sampling rate insufficient to capture the extremes of the amplitude at very fast speeds.

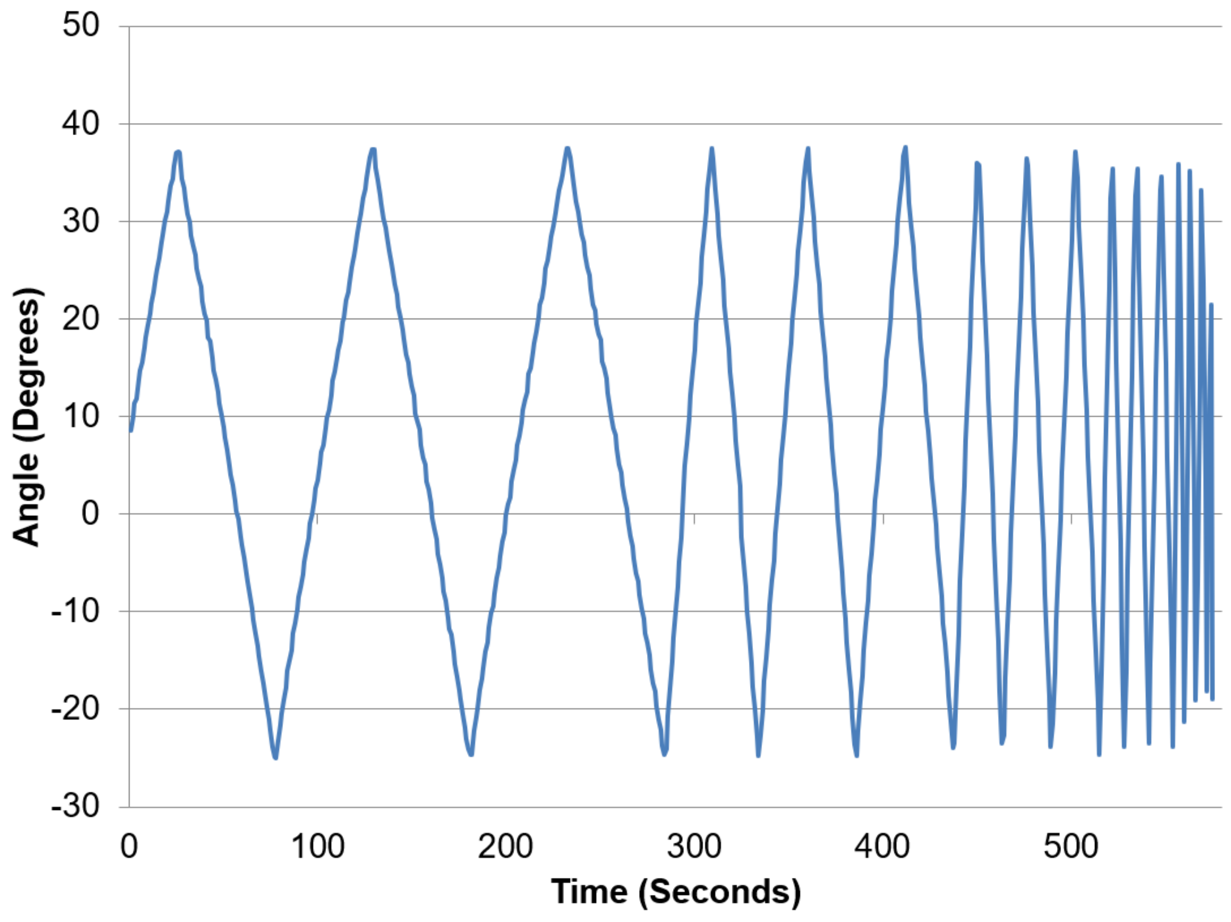
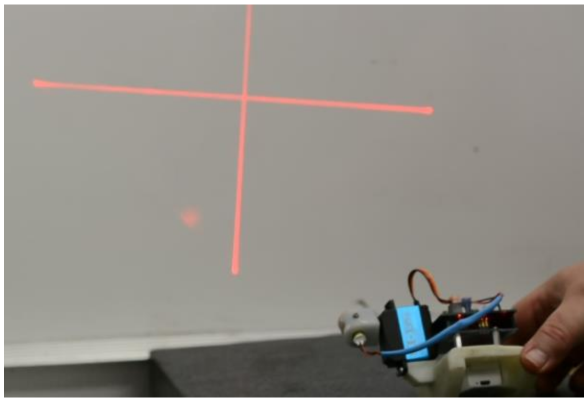
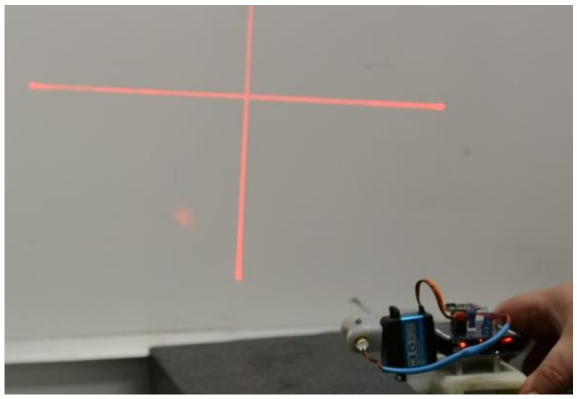


Figure 6 Sample Angle Measurement (Actual Data from Bosch Sensor)

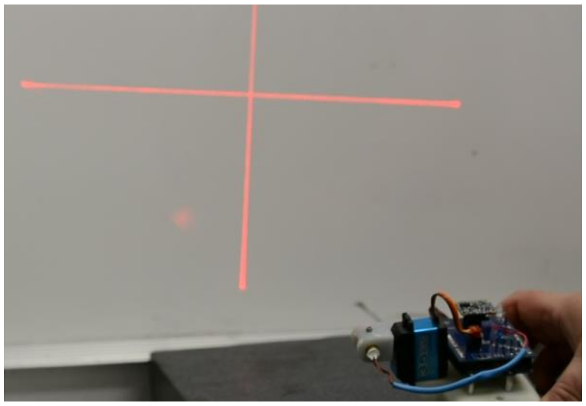
This research also included creating a custom 3D printed manually activated rocking device that demonstrated the proof-of-concept. A slow speed Arduino-based custom PCB driving a remote-control servo with a laser cross hair was first developed that evolved to a faster Raspberry Pi custom PCB driving a more responsive remote-control servo. These custom-designed and fabricated proof-of-concept models based on Arduino and Pi based are shown in Figure 7 and appear in Appendix 2. Figure 7 are screen shots from a video. These screen shots show the laser cross hair pointing to the same location in space as the base is rocked forward and backward.



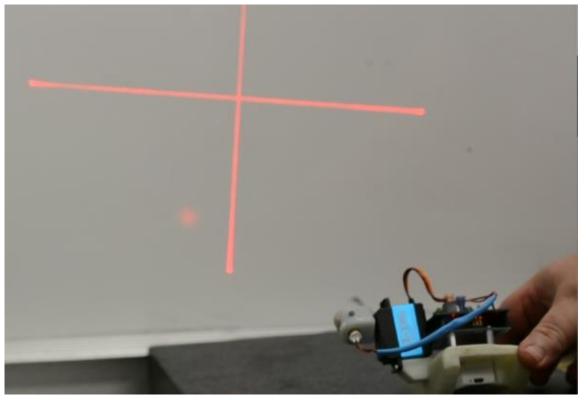
(a) Rocking forward



(b) Horizontal



(c) Rocking backward



(d) Rocking forward again

Figure 7 Proof-of-Concept Model Demonstration

The final research portion was designing a “coffee cup holder” structure with a Raspberry Pi and a custom PCB to record any of the nine-axis of information as shown in Figure 8 (details can be found in Appendix 4). The data collection device has thumb screws that can take up the space in different vehicle coffee cup holders. The Microswitch mounted on the side is used to start and stop the data collection. The angular rate sensor is shown mounted on the end of the Raspberry Pi plastic enclosure.

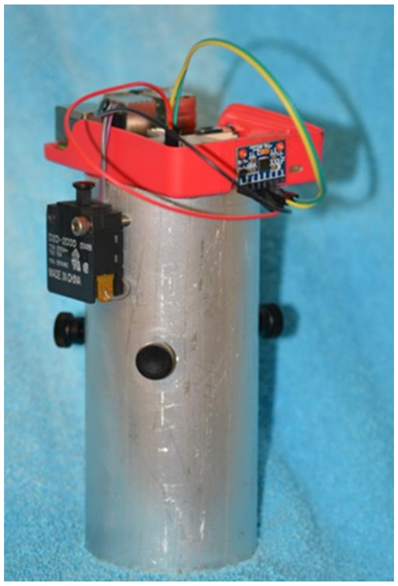


Figure 8 Vehicle Data Measurement Device

Angular position and rates from four vehicles driving the same roadway path were collected that included train tracks and a hill including several sudden stops and starts. The vehicles included a 2012 Ford Escape (see Figure 9), 2022 Ford Escape, 2017 Fiat 500e, and 2013 Ford C-Max.   Figure 9 shows the magnitude of angular changes is within 25 degrees and interestingly there is very little flat roadway in this sample.

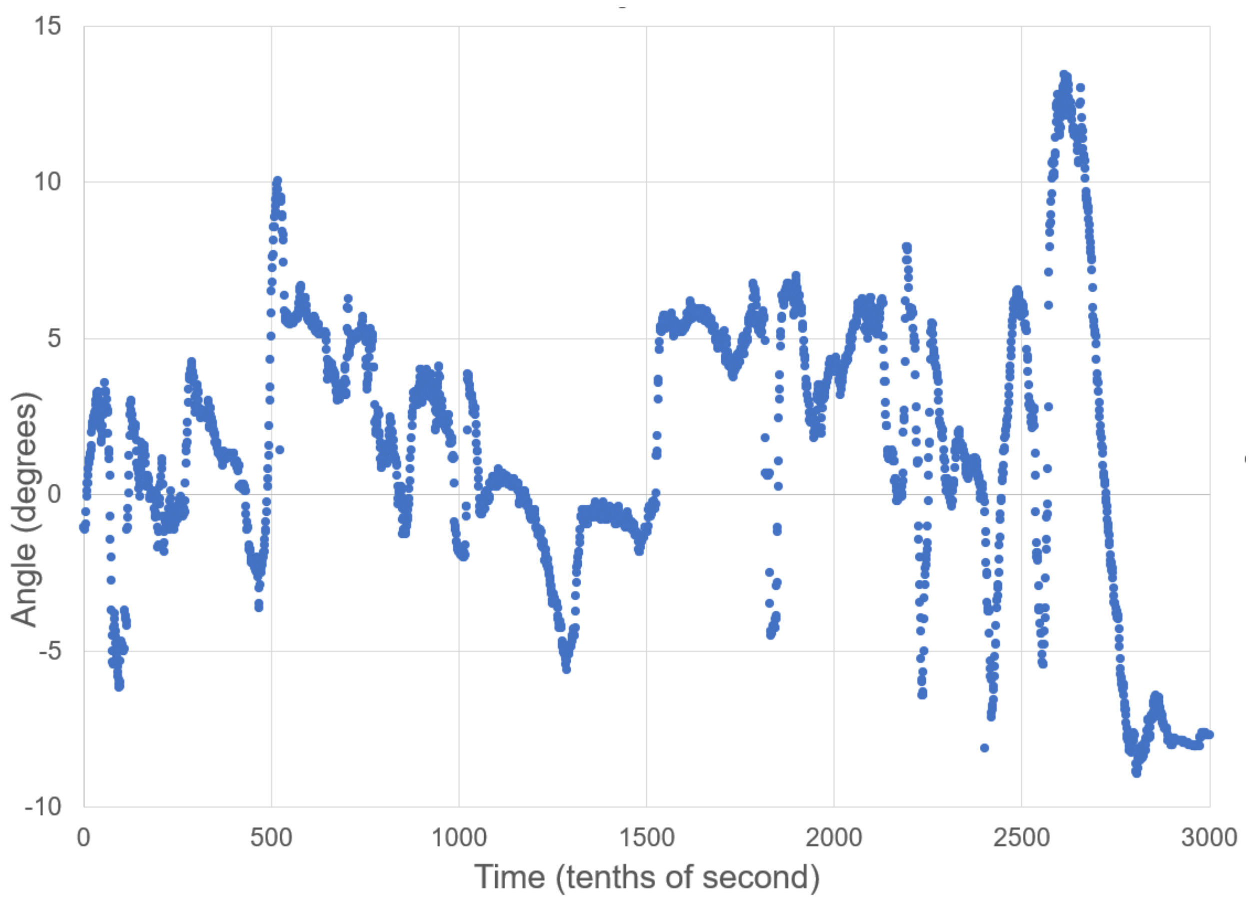


Figure 9 Sample Drive for 2012 Ford Escape Pitch Angle

CHAPTER 3

DERIVATION OF SYSTEM EQUATIONS

The first step in deriving the system Equations for the inertial headlight servo is to define the angles used. Figure 10 shows the five pitch direction angles of interest. The yaw and roll angles can be ignored for this analysis.

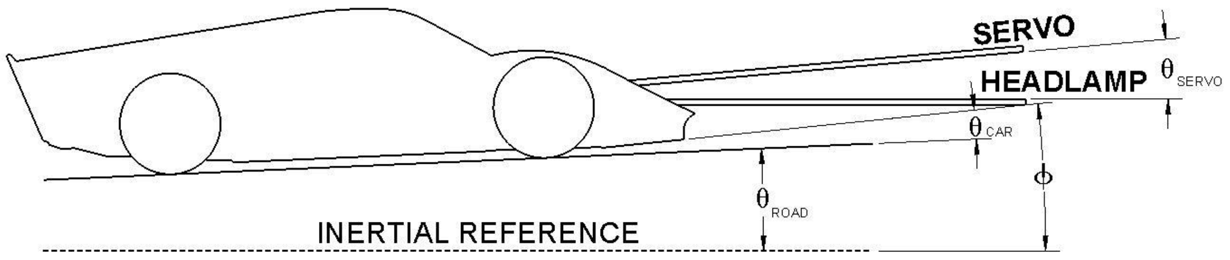


Figure 10 Angle Definitions

These five angles are measured from the inertial reference or an axis fixed to the car and their interactions are shown in Equations 1, 2, and 3.

$$\phi = \boldsymbol{\theta}_{\mathbf{ROAD}} + \theta_{\mathbf{CAR}} \qquad [1]$$

$$\boldsymbol{\theta}_{\mathbf{LIGHT}} = \phi - \boldsymbol{\theta}_{\mathbf{SERVO}} + \boldsymbol{\theta}_{\mathbf{OFFSET}} \qquad [2]$$

$$\boldsymbol{\theta}_{\mathbf{OFFSET}} = \boldsymbol{\theta}_{\mathbf{f}} + \boldsymbol{\theta}_{\mathbf{ALGORITHM}} \qquad [3]$$

where:

ϕ = Angle of headlamp with respect to inertial reference in degrees

θ_{ROAD} = Angle of road with respect to inertial reference in degrees

θ_{CAR} = Angle of car with respect to the road in degrees

θ_{SERVO} = Angle of headlight servo with respect to the car in degrees

θ_{LIGHT} = Angle of headlight with respect to inertial reference in degrees (also known as the inertial point angle)

θ_{OFFSET} = Offset angle of headlight from the inertial reference in degrees

θ_{f} = Fixed starting angle of headlight in degrees (see Appendix 8)

$\theta_{ALGORITHM}$ = Angle correction in degrees

The algorithm angle is currently assumed to be zero for the purposes of this research. In the future, this angle will be defined as either adding to or subtracting from the fixed angle based on driving conditions, vehicle speed, pitch angle changes, pitch angle and perhaps many other parameters.

The block diagram for the system is a traditional feedback system with the servo mechanism located in the feedback loop the components interactions indicated with multiport effort and flow graphics as shown in Figure 11[8].   The input, output, and feedback angles were defined previously in Figure 10.

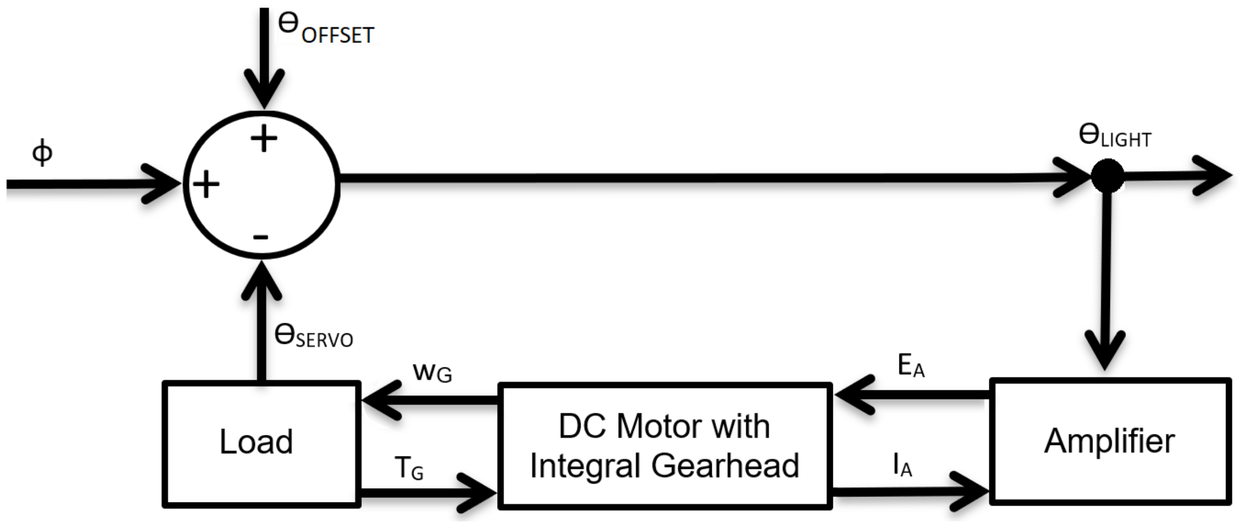


Figure 11 Multiport Diagram

The H-Bridge Driver Amplifier is governed by Equation 4:

$$E_A = G_A\theta_{LIGHT} - R_A I_A \qquad [4]$$

The DC Motor with Integral Gearhead is governed by Equations 5 and 6:

$$\omega_G = K_V E_A \qquad [5]$$

$$I_A = K_T T_G \qquad [6]$$

The load angle and required torque are shown in Equations 7 and 8:

$$\theta_{SERVO} = \frac{1}{D}\omega_G \qquad [7]$$

$$T_G = \frac{2\pi}{360°} J\dot{\omega}_G \qquad [8]$$

where:

θ_{SERVO} = Angle of headlight servo with respect to the car in degrees

θ_{LIGHT} = Angle of headlight with respect to inertial reference in degrees

E_A = Voltage to H-Bridge Driver in Volts

G_A = Amplifier gain in Volts per degree

R_A = Resistance of the H-Bridge Driver in Ohms

I_A = Current through H-Bridge Driver in Amperes

K_V = Motor velocity constant in degrees per Volt - sec (see Appendix 7 for unit conversions)

K_T = Motor torque constant in Amp per ounce-inch (see Appendix 7 for unit conversions)

ω_G = Gearhead rotary speed in degrees per second

T_G = Torque out of gearhead in ounce-inches

D = Differential operator is one per second

J = Rotary inertia of the load in ounce – inch – second^2 (see Appendix 7 for unit conversions)

Substituting Equation 4 into Equation 5 yields Equation 9:

$$\omega_G = K_V(G_A\theta_{LIGHT} - R_A I_A) \qquad [9]$$

Substituting Equation 6 into Equation 9 and simplifying yields Equation 10:

$$\omega_G = K_V G_A \theta_{LIGHT} - K_V R_A K_T T_G \qquad [10]$$

Substituting Equation 8 into Equation 10 yields Equation 11:

$$\omega_G = K_V G_A \theta_{LIGHT} - \frac{2\pi}{360°} K_V K_T R_A J \dot{\omega}_G \qquad [11]$$

Solving Equation 11 in terms of $\boldsymbol{\omega_G}$ yields Equation 12:

$$(K_V K_T R_A J\, D + 1)\omega_G = K_V G_A \theta_{LIGHT} \qquad [12]$$

Substituting Equation 2 into Equation 12 yields Equation 13:

$$(K_V K_T R_A J\, D^2 + D)\theta_{SERVO} = K_V G_A(\phi - \theta_{SERVO}) \qquad [13]$$

Simplifying Equation 13 yields Equation 14:

$$(K_V K_T R_A J\, D^2 + D + K_V G_A)\theta_{SERVO} = K_V G_A \phi \qquad [14]$$

Solving Equation 14 for ϕ yields Equation 15:

$$(\frac{K_V K_T R_A J}{K_V G_A} D^2 + \frac{D}{K_V G_A} + 1)\theta_{SERVO} = \phi \qquad [15]$$

For a second order system, the natural frequency, ω_N, and damping ratio, ζ, can be determined from the coefficients of the governing Equation 15 for Equation 16:

$$(\frac{1}{{\omega_N}^2} D^2 + \frac{2\zeta}{\omega_N} D + 1)\theta_{SERVO} = \phi \qquad [16]$$

Therefore, from Equation 16, the natural frequency (rad/s) is found in Equation 17:

$$\omega_N = \sqrt{\frac{G_A}{K_T R_A J}} \qquad [17]$$

And also from Equation 16, the damping ratio (unitless) is found in Equation 18:

$$\zeta = \frac{\omega_N}{2K_V G_A} \qquad [18]$$

CHAPTER 4

DESIGN EMBODIMENT

The breadboard design is described in Appendix 5. The DC motor parameters can be found in Appendix 7. For the breadboard device described in Appendix 5 and Appendix 7, the parameters are:

G_A = 120 Volts / degrees

K_T = 174.8 Amp / oz-in

K_V = 22.5 deg / V-s

R_A = 22.2 Ohms

J = 0.0016 oz-in-s^2

These parameters in Equation 17 yields a natural frequency, ω_N, is 33.3 rads/s (5.3 Hz) and from Equation 18 yields a damping ratio, ζ, of 0.06. The very low damping ratio makes the system oscillate [8][9]. The damping ratio for the simulation is less than ideal but it can be mitigated with a proportional-derivative controller with a low pass filter so that input noise is not amplified. These factors lead to a more appropriate design embodiment that would be a production design with a natural frequency target of 62.8 rad/s (10 Hz) and a damping ratio, ζ, closer to 0.7 [13][14].

For Matlab simulation purposes we will need the state equations in the form of matrix which can be found in Equations 19 and 20:

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -\omega_N{}^2 & -2\zeta\omega_N \end{bmatrix} x + \begin{bmatrix} 0 \\ \omega_N{}^2 \end{bmatrix} u \qquad [19]$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x + [0]u \qquad [20]$$

The linear second-order program code is now ready to put into Matlab [10][11][12]. The input file for this simulation can be the actual data collected from the device described in Appendix 4 or by the created roadway data described in Appendix 6. Code 2 is the Matlab program used to simulate the system.

Code 2 Matlab Simulation Code

```
clear; close all; clc;
%---------------------
dt   = 0.01;
time = 0:dt:20.30-dt; n = length(time);
% Read Excel File
uinp = readmatrix('R2-2.csv');
%Production
Wn = 62.8;
Z = .7;
%
X0 = [0; 0]; % Initial conditions at time t = 0 s
Ys = zeros(n,length(X0)); Ys(1,:) = X0';
for ct = 1:n-1
    [T, Y] = ode45(@eqn, [time(ct) time(ct+1)], X0, [], Wn, Z, uinp(ct,1));
    Ys(ct+1,:) = Y(end,:);
    X0         = Y(end,:)';
end
%
figure(1);
  plot(time,uinp,'r--', 'linewidth',2, 'DisplayName', 'Phi (Roadway Input)');
xlabel('Time (s)'); ylabel('Angle (deg)');
  hold on
  plot(time,Ys(:,1),'b', 'linewidth',1, 'DisplayName', 'Servo Output Angle');
  hold on
  plot(time, (Ys(:,1)-uinp),'k:', 'linewidth',2, 'DisplayName', 'Inertial
Pointing Angle');
  legend ('show');
  xlim ([0 20.30]);
  exportgraphics(gcf,"HighwayGravelResults.png",'resolution',300);
function [xd] = eqn(t,X,Wn,Z,u)
% simulates, xddot + 2*Z*Wn*xdot + Wn^2*x = u (Linear System)
x1 = X(1,1);
x2 = X(2,1);
xd(1,1) = x2;
xd(2,1) = -(Wn^2)*x1 - 2*Z*Wn*x2 + (Wn^2)*u;
end
```

Figure 13 shows the random highway with gravel roadway and is based on a power spectral density function [11]. This roadway represents elevation changes across a distance of roadway at a particular speed. This roadway was created for a vehicle moving 45 MPH (20.11 m/s).

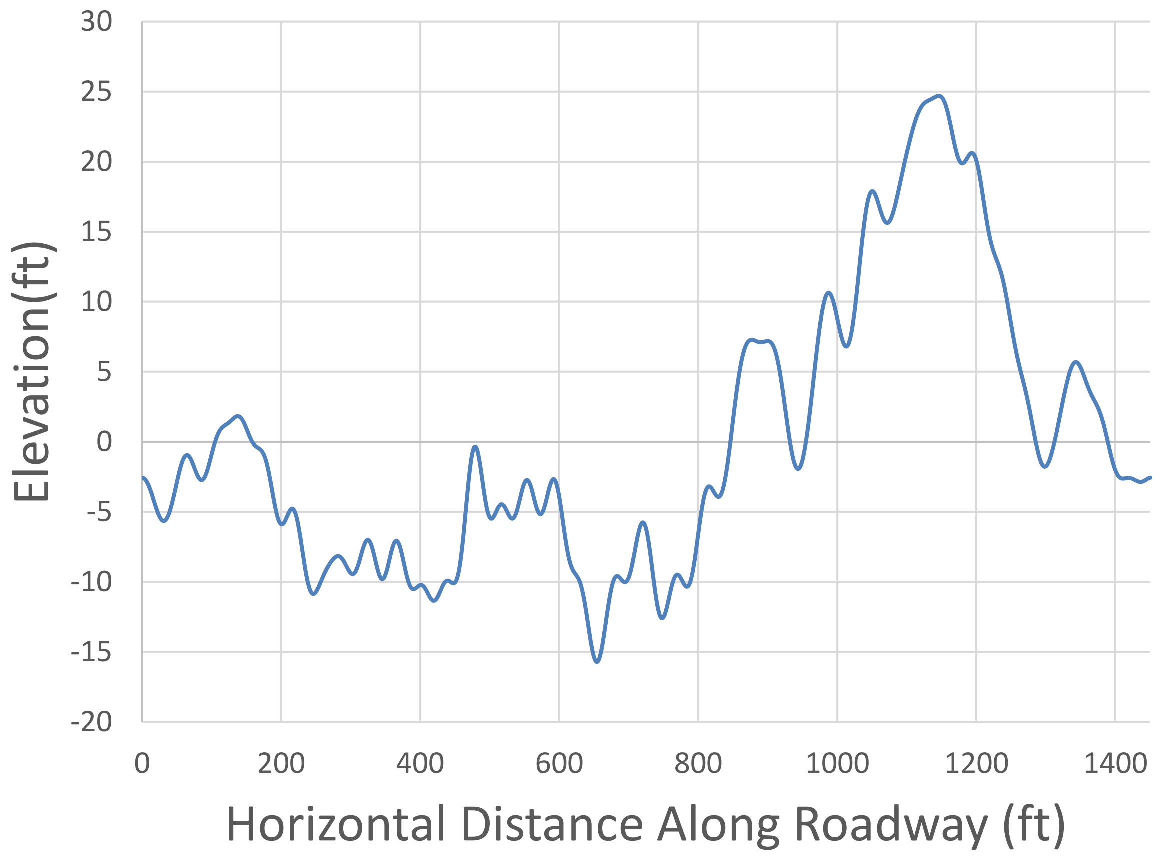


Figure 12 Roadway Elevation Based On PSD

Figure 13 shows the same roadway as Figure 12 but uses as 145-inch wheel base vehicle (see Appendix 8). The pitch angle for the road at each point uses the arc tangent of the difference of the elevations at the front and rear wheels over the length of the vehicle wheelbase.   This array of pitch angles will be the input file for angle ϕ used in the simulation.

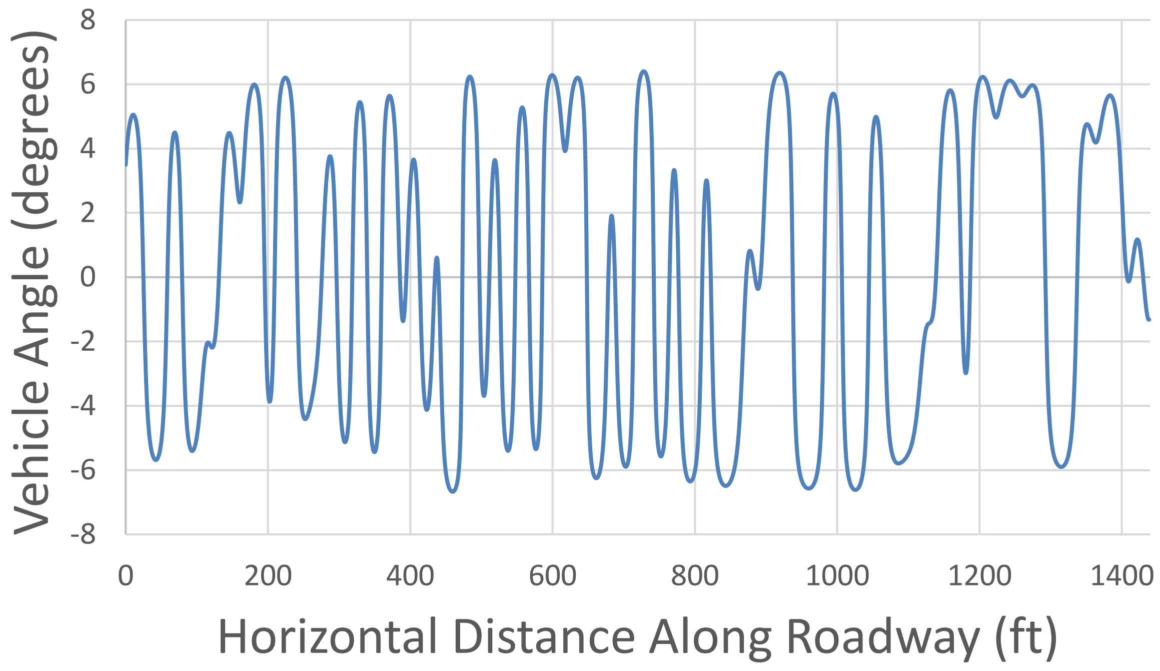


Figure 13 Pitch Angle of Vehicle Based on PSD Generated Roadway

Using the input angles from Figure 13 in the Matlab Code 2, creates the target production system simulation response shown in Figure 14 (a). This simulation is for the entire range of roadway which is 20.3 seconds long (note the Matlab code `xlim ([0 20.30]);`). The curves in Figure 14 are changes in angles relative to a reference angle of zero. Since the design objective is for the changes in the servo angle to cancel the changes in the vehicle pitch angle, in a perfect design, the change in the servo angle would be exactly equal but opposite in sign to the change in the vehicle pitch angle; this difference is shown with a black dotted line.

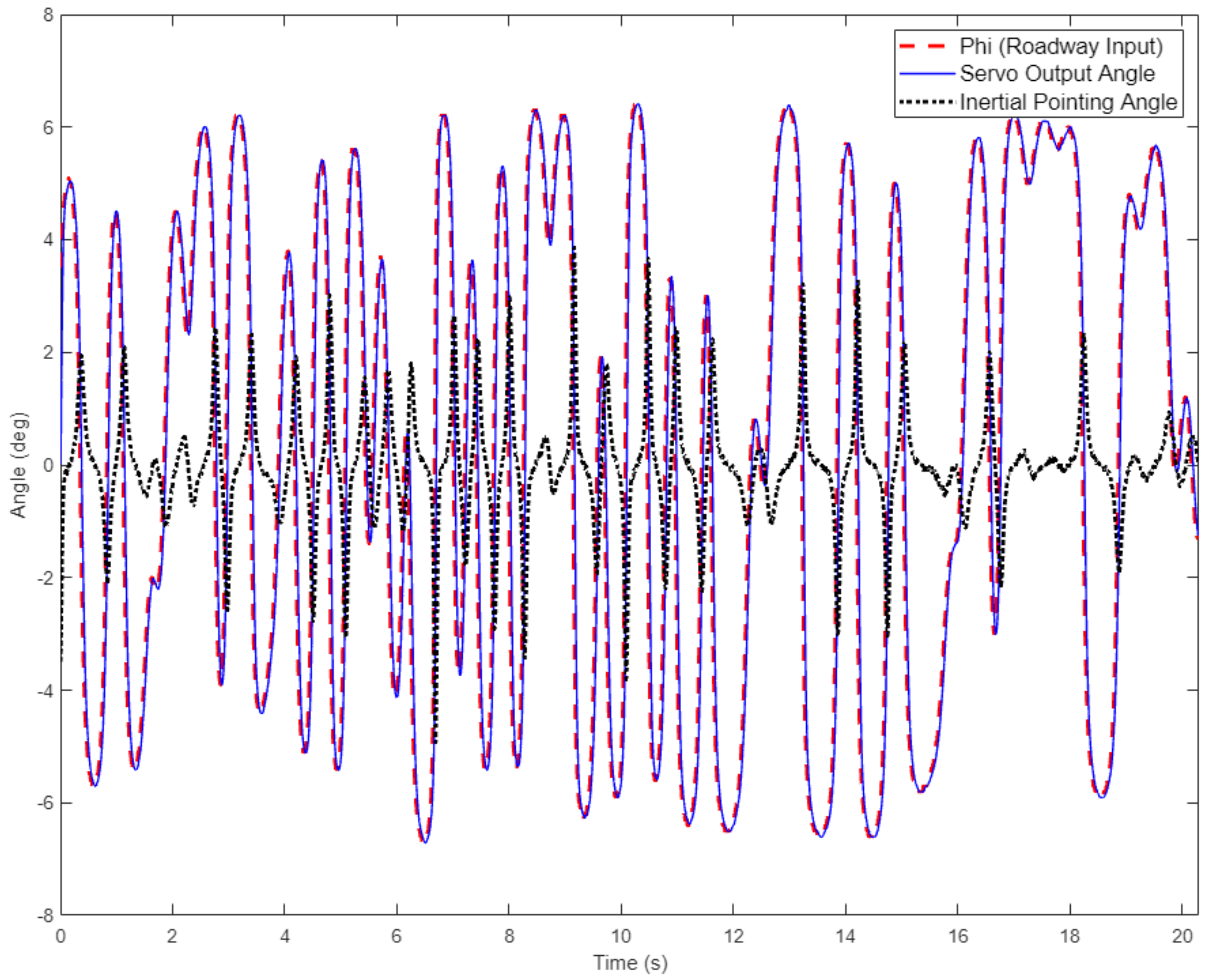


Figure 14 (a) Simulation of Production System

For clarity, one small portion of the response of the system is shown in Figure 14(b). This simulation is for a two second section of the roadway starting at 16.6 seconds and ending at 18.6 seconds (note the Matlab code line was changed to `xlim ([16.6 18.6]);`). As shown in Figure 14(b), the magnitude in the change in the servo angle is almost exactly equal to the change in the vehicle pitch angle but with a slight phase delay. The slight phase delay results in the changes in the light beam angle not being zero but definitely smaller than the changes in the vehicle pitch angle. Hence, a production design objective would be to increase the response time of the servo system to minimize the delay and thus, reduce the changes in the light beam angle (inertial pointing angle) to as small as possible.

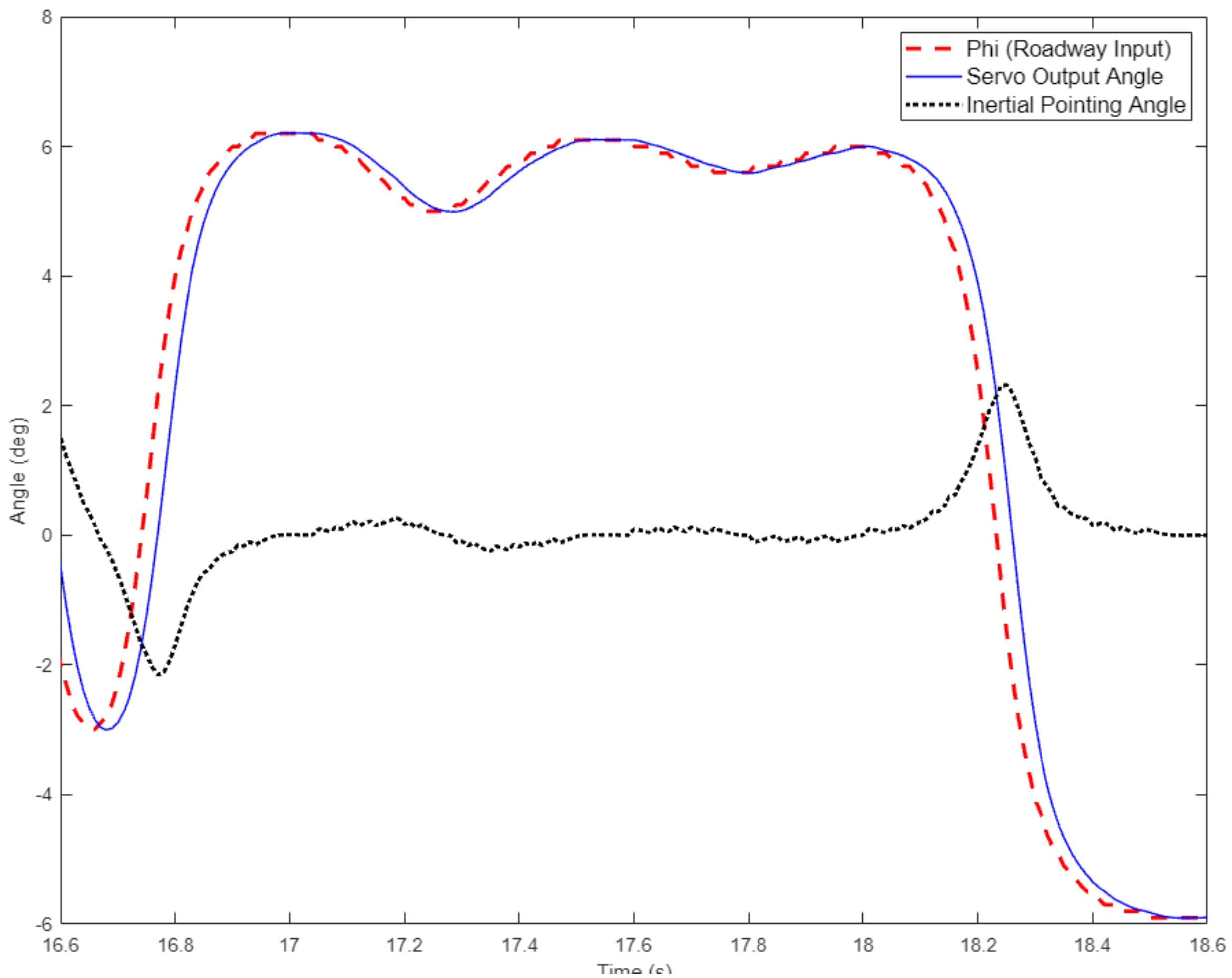


Figure 14 (b) Portion of Simulation of Production System

CHAPTER 5

CONCLUSION

The main question of this dissertation was the improvement of driver safety based on adding inertially stabilized headlamp. The simulation, proof-of-concept, and breadboard demonstration all show that a stable headlamp position is better for a moving vehicle than a headlamp tied physically to the vehicle chassis.

As the research progressed for this dissertation there was a familiar three step process used. Step one was to investigate prior art in the area of interest through reading books and internet searches. Step two was to design a real or virtual engineering apparatus to measure or demonstrate the area of interest. Finally, step three was to reflect on the data collected and provide a summary to document the engineering device and the data.

Other applications for this technology include other moving vehicles. Inertially stabilized lights and cameras are commonplace for manned and unmanned aerial vehicles. However, inertially stabilized marine vehicle lighting might improve safety over rough water conditions and could certainly reduce motion sickness symptoms in passengers because one of the mitigating methods of “sea sickness” is concentrating on the horizon which is not visible at night as well as improve pilot vision.

This dissertation contribution to the literature is a unique combination of existing several engineering concepts in a new and novel solution to vehicle headlamp pitch angle control. In addition to this dissertation, the author and supervising professor have prepared a provisional patent application.

This dissertation provides the solution for establishing inertially stabilized headlamp angles. Continuing work in this subject should add digital logic components to an electronic control unit (ECU) for the headlamps. The future headlamp system ECU would take into consideration the weather (rain, snow, or fog), oncoming traffic headlamps, vehicle speed, and roadway angular pitch rate change to provide an optimum headlamp angle over the entire driving experience. There would also be continuing work to completely retrofit an adaptive headlamp with pitch angle control motor.

REFERENCES

[1] Bosch Automotive Handbook, Wiley, Hoboken, N.J. , Wiley , John Wiley, 2022, pgs 1337-1373.

[2] Nilsson, P. Master of Science Thesis in Automatic Control (2016) Automatic Headlight Levelling Using Inertial Measurements (Unpublished Master of Science Thesis in Automatic Control). Department of Electrical Engineering, Linköping University, Sweden.

[3] Žaludová, L., The Headlamp Illuminance in Front of a Vehicle, Lambert Academic Publishing, Saarbrücken, Germany, 2012.

[4] MPU-600 and MPU-6050 Product Specification Revision 3.4, Sunnyvale, CA, InvenSense inc., 2013.

[5] ADXRS800 Datasheet for High Performance, SPI Digital Output, Angular Rate Sensor, Norwood, MA, Analog Devices, 2015.

[6] BNO055 Intelligent 9-axis Absolute Orientation Sensor, Reutilingen, Germany, Bosch Sensortec GmbH, 2021.

[7] Kim, C. Seok, J., Kim, S., Lee, K., Kim, J. Moon, I, Kang, J, and Jo, K., Embedded Inertial Sensor-Based Road to Vehicle Pitch Estimation for Automatic Headlamp Leveling, IEEE Vehicular Technology Society, 2003.

[8] Woods, R.L. and Lawrence, K.L., Modeling and Simulation of Dynamic Systems, Prentice Hall, Eaglewood Cliffs, NJ, 1997, pgs. 203-211.

[9] D’Azzo, J.J. and Houpis,C.H., Feedback Control System Analysis and Synthesis, McGraw-Hill, New York, NY, 1960, pgs 111-129.

[10] D’Azzo, J.J. and Houpis,C.H., Linear Control System Analysis and Design, McGraw-Hill, New York, NY, 1988, pgs. 743-770.

[11] Hullender, D., Dynamic Systems Modeling and Simulation: Theory and Examples, 24th Edition, Unpublished course notes, 2023, pgs. 2:27-2:30, 8:361

[12] Hossain, E., MATLAB and Simulink Crash Course for Engineers, Springer, Cham, Switzerland, 2022, pgs. 19-45.

[13] Kuo, B.C., Digital Control Systems, Holt, Rinehart and Winston, Chicago, IL, 1980, pgs. 303-338.

[14] Nasar,S.A. and Unnewehr,L.E., Electromechanics and Electric Machines – Second Edition, Wiley, New York, NY, 1983, pgs 157-219.

[15] Bollinger, J.G. and Duffie, N.A. , Computer Control of Machines and Processes, Addison-Wesley, Reading, MA, 1988, pgs 272-273.

APPENDIX 1

ANIMATION OF HEADLAMP PROBLEM AND SOLUTION

An Onshape CAD model was created and animated to show both the headlamp problem and the inertial-stabilized headlamp solution in a single video. The animation required several custom 3D parts to be created including a track, a vehicle, a light source, and a slider bar. These parts can be seen in Figure 15.

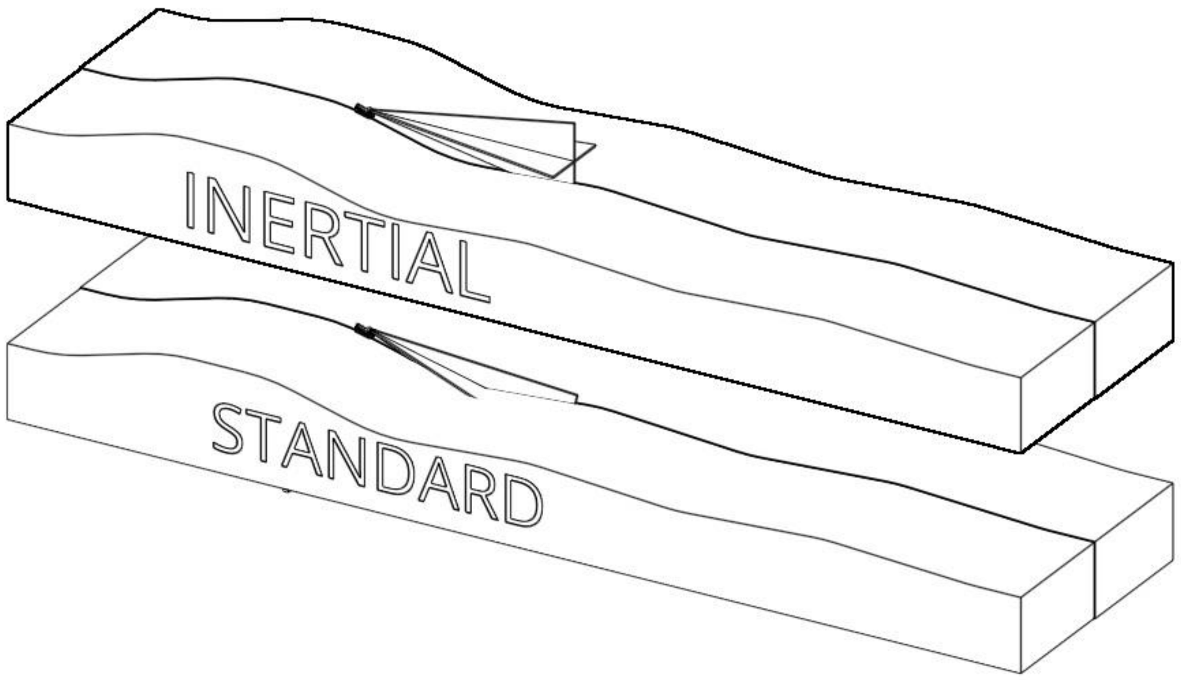


Figure 15 CAD Model for Animation

The track, part name “Land”, is a 3D model of a cross section of a road with various bumps and hills based on a single sketch with fillets to create a contiguous and continuous path. The entire track surface sketch is parameter driven so it can be changed to show how a vehicle would move over a slightly exaggerated but reasonable surface. There are two configurations of the track with the only difference being a text label on the side stating “STANDARD” and “INERTIAL”.

The vehicle chosen was a motorcycle model, part name “Motorcycle”, was available online in the public domain.   A motorcycle was chosen to show a singular headlamp but the process could be repeated with a two-headlamp vehicle such as a car or track. The motorcycle headlamp was modified to be a sphere so the model of the light beams could be mated to a spherical surface as in a ball and joint connection.

The slightly transparent light beams were modeled as a physical 3D part, part name “Headlamp”, and has two configurations with one being a solid cone of light and the one with cross hairs.   The solid cone of light better simulates an actual headlamp light source but the cross-hairs more accurately show the movement of the light beam in the animations.

The slide bar, part name “Slider”, is a simple small rectangular block.   The slider was used to allow both vehicles to move simultaneously without moving the cursor in the visible area and to allow for a single animation parameter to be set for the position of the slider bar.

The assembly of the 3D parts, assembly named “Comparison”, has two tracks arrayed vertically.   Each track has a vehicle mated to their respective track surfaces.   The front and rear wheels are both mated tangentially to the track surface so the motorcycle will rock up and down as it moves along the track.   The track is three dimensional but the motorcycle is limited to travel down the midplane of the track.   Both vehicles are also mated tangentially to the single slider bar.   The slider bar is also constrained to move along the flat bottom of one of the tracks.   The vehicle on one track will have the light beam source constrained to be aligned to the vehicle as in current vehicles.   This track and assembly are identified with a label of “STANDARD” on the side.   The vehicle on the other track will have the light source constrained to the flat horizon to simulate inertial stabilization.   The track and assembly are identified with a label of “INERTIAL” on the side.

The Windows 10 video capture routine is activated by touching WIN+ALT+R simultaneously.   Either by using the parameter animation or the user mouse movement, the slide bar moving back and forth simulates the movement of the vehicles back and forth along the track path. The captured video animation shows the difference in the current method of mounting headlamps and the inertially-stabilized headlamp. Two animations were created for clarity. Animation one was a straight on side view of the two tracks. Animation two was an anisometric view of the two tracks. These animations are available for viewing along with the CAD models on the Onshape website.

APPENDIX 2

PROOF-OF-CONCEPT MODEL

A proof-of-concept model was designed to demonstrate how an angular position sensor could be used with a microcontroller and a laser cross-hair generator on a small manually operated rocking platform. The proof-of-concept model is shown in Figure 16.

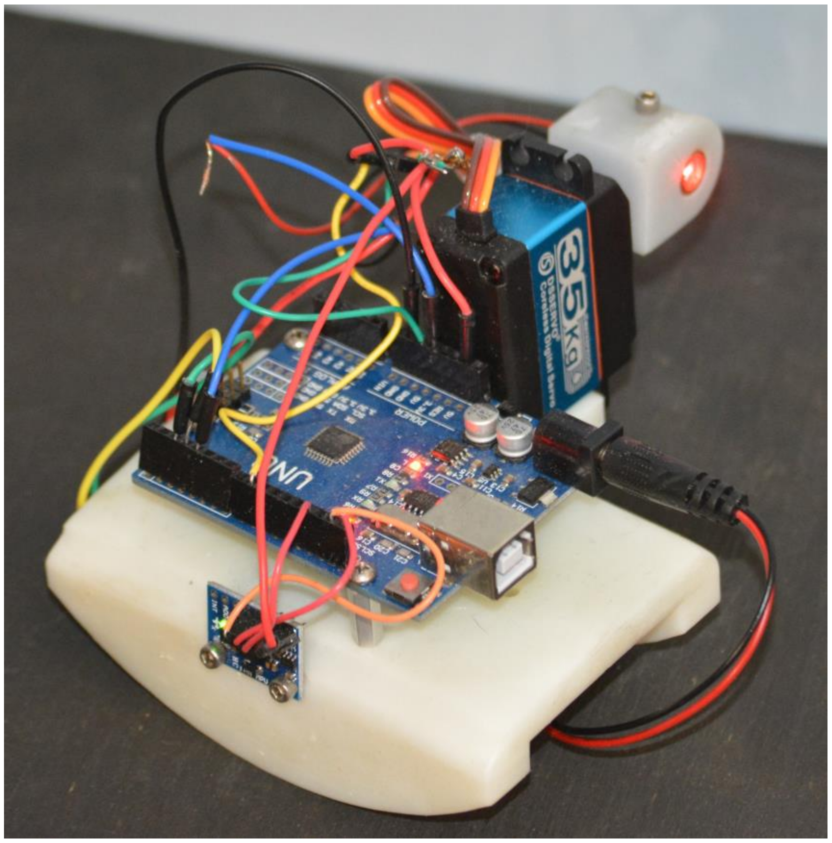


Figure 16 Proof-of-Concept Model

The proof-of-concept has several off-the-shelf components and two custom 3D printed parts. The larger custom designed part is the rocking platform that allows the user to "rock" the system back and forth like a rocking chair. The rocking platform provides a nine-volt battery holder underneath in the center of the mass. On the top of the rocking platform is a mount for an Arduino Uno microcontroller. On one side of the rocking platform is a mount for the angular rate sensor module which measures two angles of motion, part number BNO055. On the opposite side of the rocking platform is an angular motion servo used primarily in remote-control vehicles, part number DS323SG. Finally, the rocking platform has a switch at the “front” of the rocking platform. One switch activates the inertial stabilization and one switch controls the cross-hair laser generator. A detailed engineering drawing of the rocking platform is shown in Figure 17.

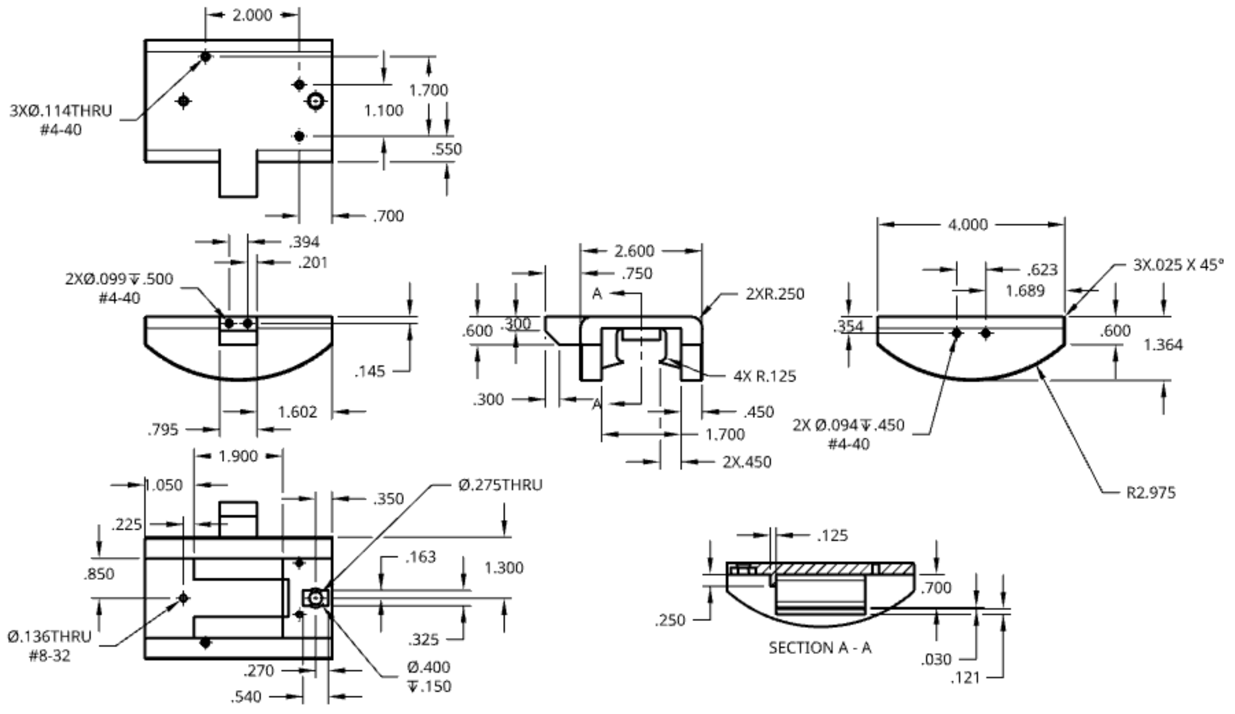


Figure 17 Rocking Platform

There is a smaller custom deigned part that attaches to the output shaft of the servo and provides a mount for a cross-hair laser generator. A detailed engineering drawing of the laser mount is shown in 18.

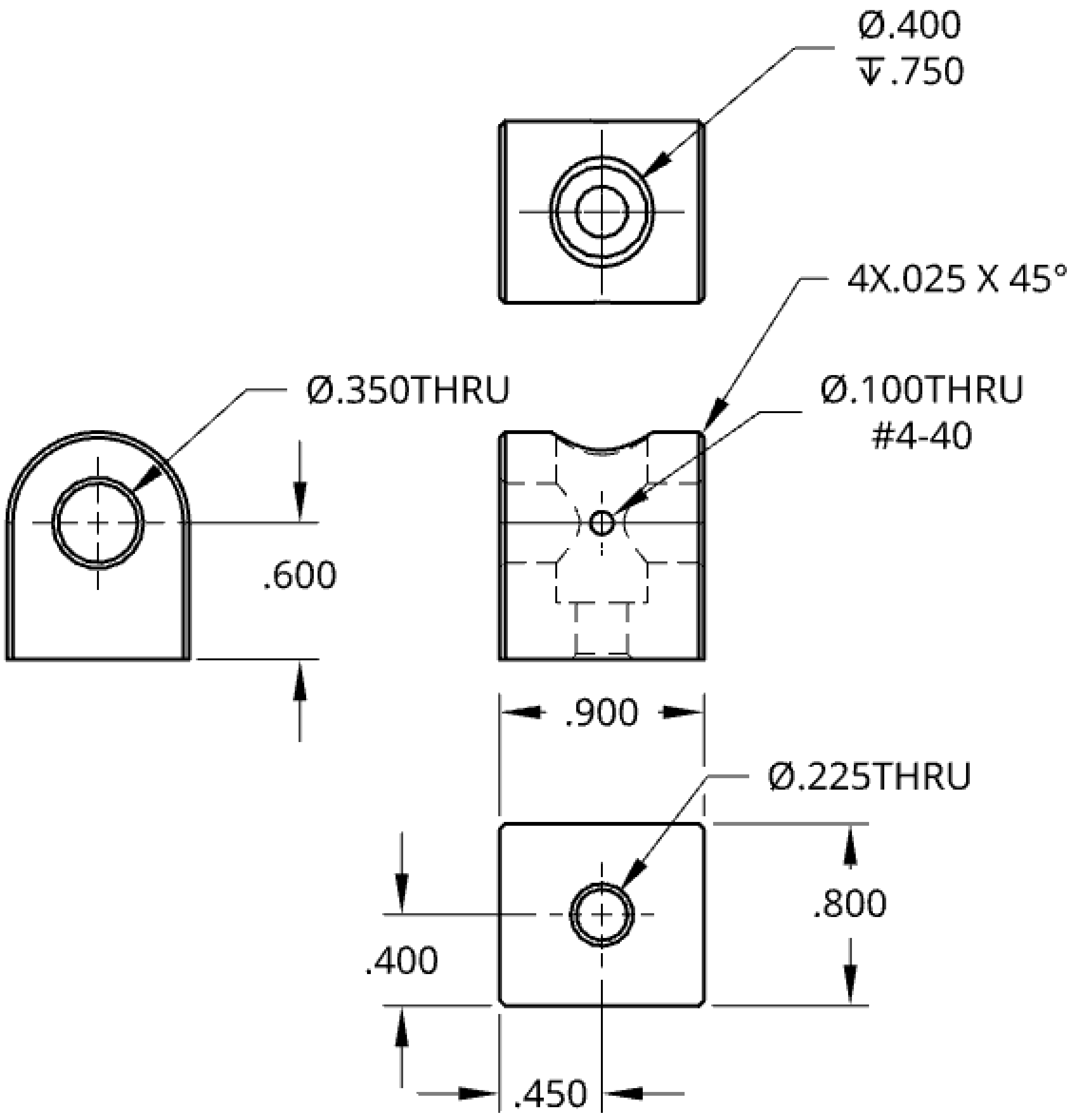


Figure 18 Laser Mount

The Arduino Uno has a voltage regulator that takes the nine volts from the battery and produces five volts for the Arduino controller, the angular rate sensor module, and the servo. The Arduino Uno has headers so the proof-of-concept module can be spaced wired without a custom printed wiring board. The schematic for the circuit is shown in Figure 19.

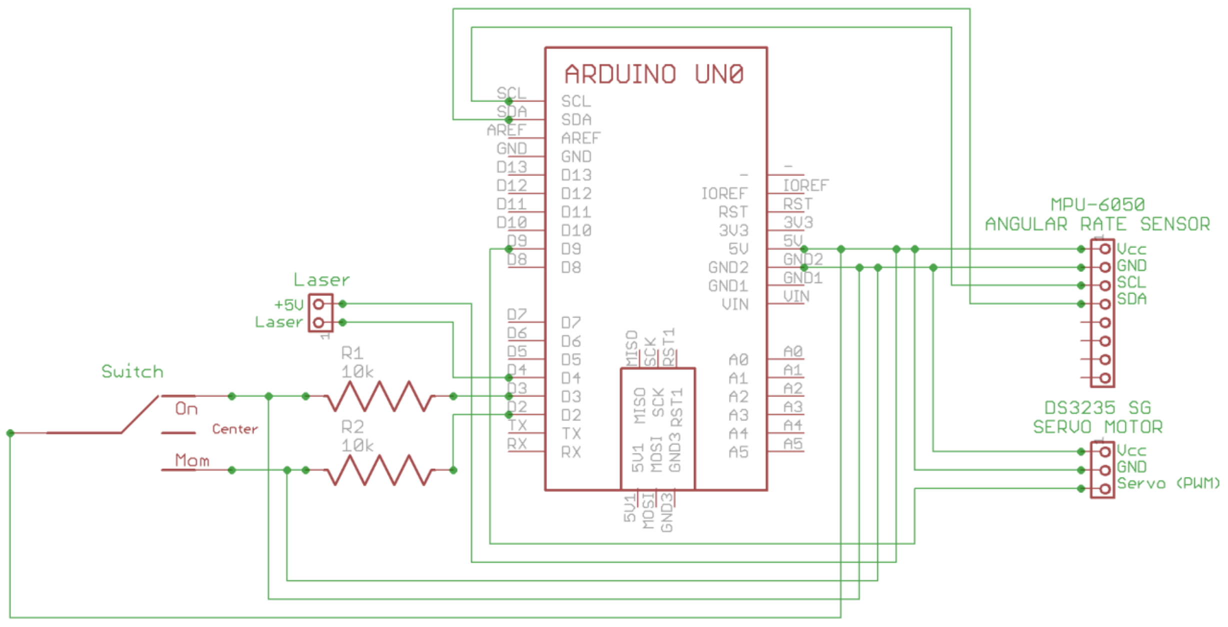


Figure 19 Proof-of-Concept Schematic

The Arduino Uno code runs in a single loop where the angular rate sensor is read. If the inertial switch is not activated then nothing happens on that loop. If the inertial switch is activated then the servo is driven opposite the angular sensor reading thus maintaining the position of the laser cross hairs. This code can be found in Code 3.

Code 3 Proof-of-Concept Arduino Code

```
//
// Written: Thomas Allsup
// Revision Log
// 02/24/2022 Started
// 03/03/2022 Updated servo
// 06/28/2022 Added comments and lo/high beam
// 09/08/2022 Added laser output (removed lo-high beam)
// 01/02/2023 Added comments and corrected laser on / off
// Comments
// Takes inputs from angular rate sensor MPU-6050 through I2C
// Sends correction output to servo
// Momentary switch turns on and off laser
// On / Off switch turns on and off inertial stabilization
//
#include <Wire.h> //library allows communication with I2C / TWI devices
#include <math.h> //library includes mathematical functions
#include <Servo.h>
// UNO Pinout - Wiring
//                                        SCL - GPS SCL Pin 3
//                                        SDA - GPS SDA Pin 4
//                                        AREF
//                                        GND - GPS GND Pin 2
// NC                                     D13
// 5V - Servo P2/Switch/GPS P1 / Laser    D12
// Res                                    D11
// 3.3V                                   D10
// 5V                                     D9 - Servo Pin 3
// GND - Servo Pin 1                      D8
// GND - Pull down resistors to D2/D3     D7
// Vin                                    D6
// A0                                     D5
// A1                                     D4 Laser On/Off
// A2                                     D3 Switch Pos 2 - Mom
// A3                                     D2 Switch Pos 1 - On/Off
// A4                                     D1 TX
// A5                                     D0 RX
const int MPU=0x68; //I2C address of the MPU-6050
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ; //16-bit integers
int AcXcal,AcYcal,AcZcal,GyXcal,GyYcal,GyZcal,tcal; //calibration variables
double t,tx,tf,pitch,roll;
Servo myservo;  // create servo object to control a servo
int pos = 0;
int ButtonOnOff = 3; // turns inertial stabilization on and off
int ButtonMom = 2; // turns laser on and off
int Laser = 4;
int LaserState=0;
void setup()
{
    Wire.begin(); //initiate wire library and I2C
    Wire.beginTransmission(MPU); //begin transmission to I2C slave device
    Wire.write(0x6B); // PWR_MGMT_1 register
    Wire.write(0); // set to zero (wakes up the MPU-6050)
    Wire.endTransmission(true); //ends transmission to I2C slave device
    Serial.begin(9600); //serial communication at 9600 bauds
    myservo.attach(9); // servo can only attach to pins 9 or 10
    pinMode(ButtonOnOff,INPUT) ; // Switch Pos 1
    pinMode(ButtonMom,INPUT) ; // Switch pos 2
    pinMode(Laser,OUTPUT) ; // Laser On / Off
    digitalWrite(Laser,HIGH); // Laser off
}
void loop()
{
  if (digitalRead(ButtonMom)==HIGH){
    LaserState = 1 - LaserState; // toggle laser staate
    if (LaserState==LOW) {
      digitalWrite(Laser, HIGH);
    } else {
      digitalWrite(Laser,LOW);
    }
    delay(500);
    }
  if (digitalRead(ButtonOnOff)==HIGH){
    Wire.beginTransmission(MPU); //begin transmission to I2C slave device
    Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
    Wire.endTransmission(false); //restarts transmission to I2C slave device
    Wire.requestFrom(MPU,14,true); //request 14 registers in total
    //Acceleration data correction
    AcXcal = -950;
    AcYcal = -300;
    AcZcal = 0;
    //Temperature correction
    tcal = -1600;
    //Gyro correction
    GyXcal = 480;
    GyYcal = 170;
    GyZcal = 210;
    //read accelerometer data
    AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) 0x3C
(ACCEL_XOUT_L)
    AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) 0x3E
(ACCEL_YOUT_L)
    AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) 0x40
(ACCEL_ZOUT_L)
    //read temperature data
    Tmp=Wire.read()<<8|Wire.read(); // 0x41 (TEMP_OUT_H) 0x42 (TEMP_OUT_L)
    //read gyroscope data
    GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) 0x44 (GYRO_XOUT_L)
    GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) 0x46 (GYRO_YOUT_L)
    GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) 0x48 (GYRO_ZOUT_L)
    //temperature calculation
    //tx = Tmp + tcal;
    //t = tx/340 + 36.53; //Equation for temperature in degrees C from
datasheet
    //tf = (t * 9/5) + 32; //fahrenheit
    //get pitch/roll
    getAngle(AcX,AcY,AcZ);
    //printing values to serial port (screen)
    // Serial.print(" Roll = "); Serial.println(roll); //used for debug
    pos=90+roll; // angle of servo is 90 degrees from zero of PCB
    myservo.write(pos);
    delay(1);
  }
}

//function to convert accelerometer values into pitch and roll
void getAngle(int Ax,int Ay,int Az)
{
    double x = Ax;
    double y = Ay;
    double z = Az;
    // pitch = atan(x/sqrt((y*y) + (z*z))); //pitch calculation
    roll = atan(y/sqrt((x*x) + (z*z))); //roll calculation
    //converting radians into degrees
    // pitch = pitch * (180.0/3.14);
    roll = roll * (180.0/3.14) ;
}
```

APPENDIX 3

DISASSEMBLY OF AN ADAPTIVE HEADLAMP

A defective Toyota adaptive headlamp was acquired for disassembly to determine how it could be modified to accept an inertial-stabilized mechanism.

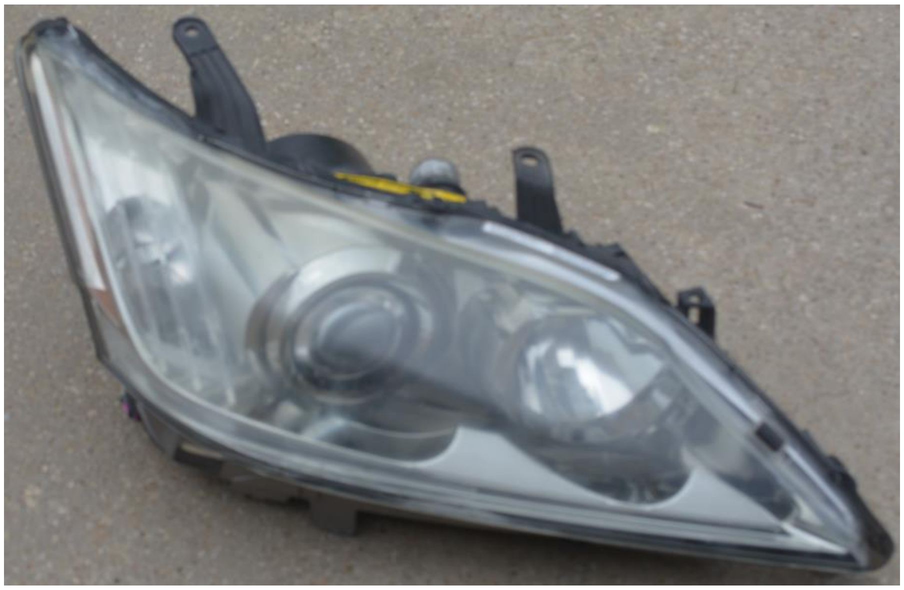


Figure 20 Disassembled Headlight

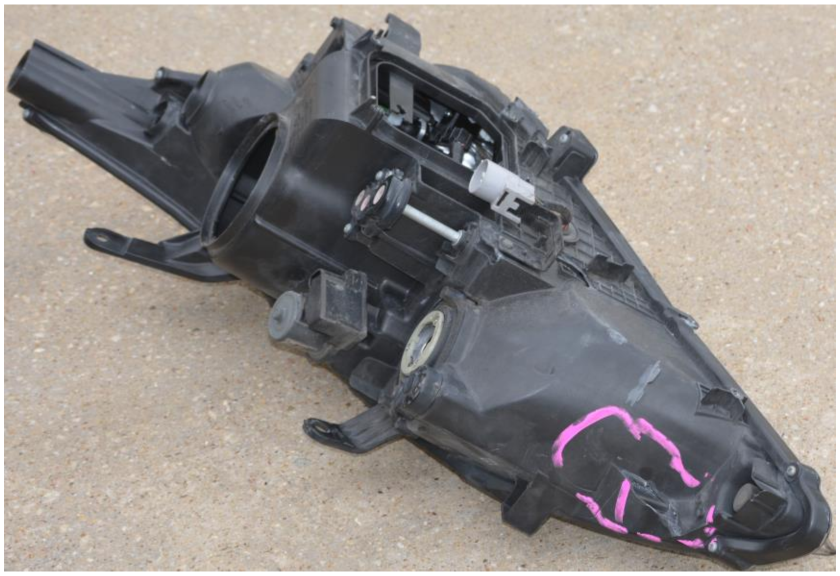


Figure 21 Disassembled Headlight Backside

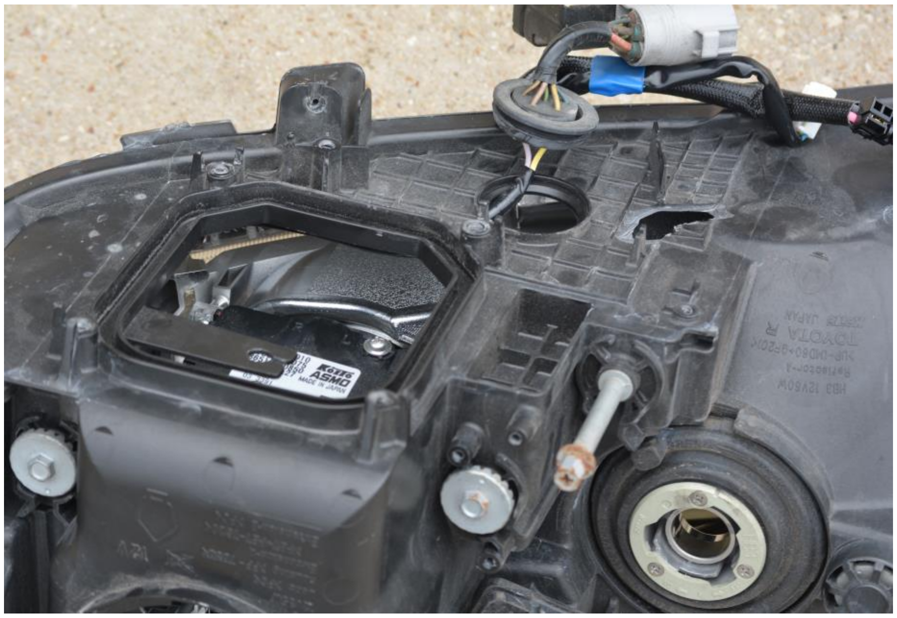


Figure 22 Disassembled Headlight with Linear Actuator Removed

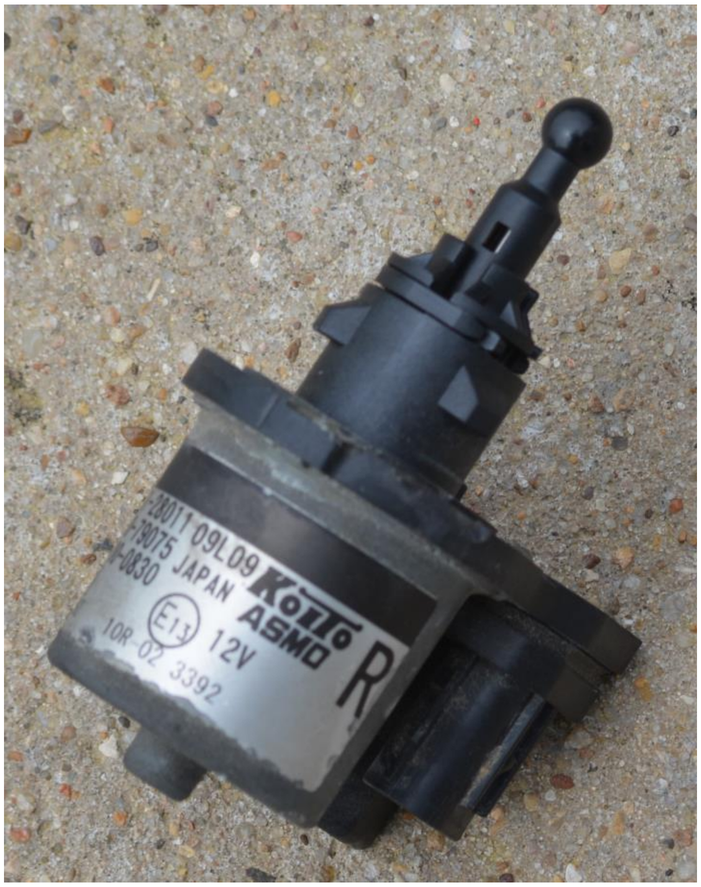


Figure 23 Linear Actuator

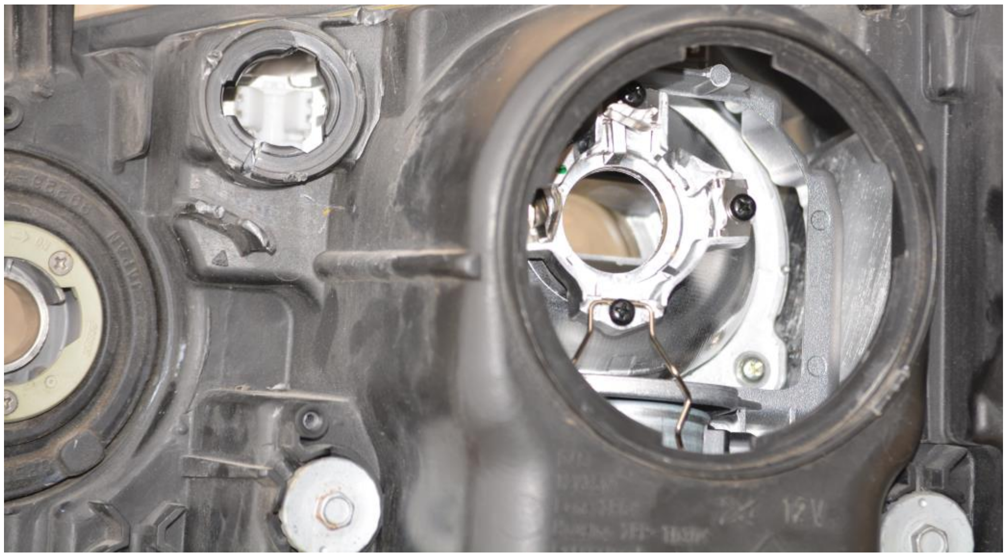


Figure 24 Disassembled Headlight with Back of Light Shown

APPENDIX 4

VEHICLE MOVEMENT DATA ACQUISITION DEVICE

To determine the bandwidth required to compensate for a vehicle's headlamp angular movement, a custom data acquisition device was required. A Raspberry Pi 4 computer was chosen as a data logging device since it could be operated in headless mode (without keyboard or display) and could easily store data in readily accessible format. Initial space wired components functioned but were not robust so a custom "Pi Hat" was designed and fabricated to eliminate as many wires as possible. The "General Purpose Input / Output" (GPIO) pins of the Raspberry Pi are routinely accessed with custom daughter printed wiring boards known as "hats" since they "top off the projects". This completed data acquisition device is shown in Figure 25.

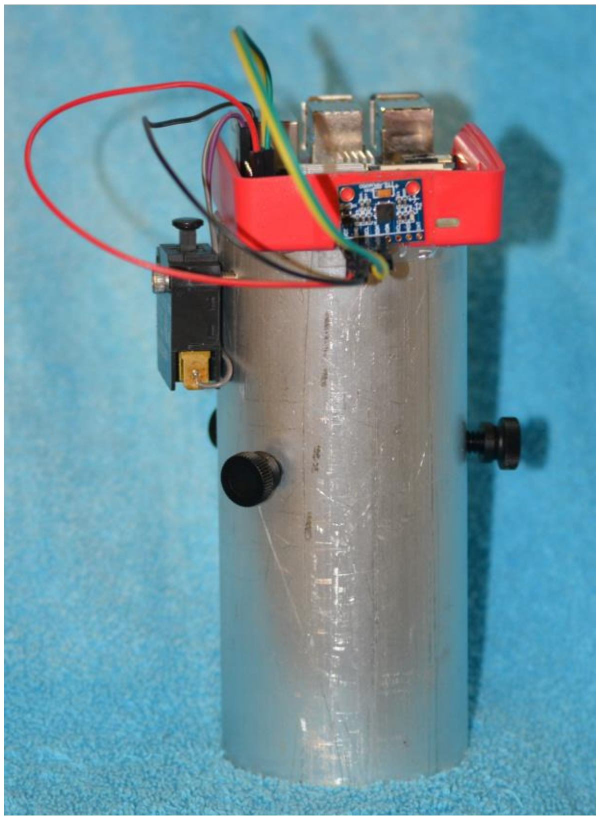


Figure 25 Vehicle Data Acquisition Device

The data acquisition device needed to be placed in several different vehicles. The device also needed to be as stable as possible so as not to introduce any additional movement to the measurement. To ensure this stability, a large heavy thick-walled aluminum custom tube was used as the base. A series of tapped screw holes were machined around the base so that the plastic thumb screws could be moved in and out to take secure to the base to the cup holder in each vehicle. On top of the tube were two tapped screw holes to mount the custom Raspberry Pi enclosure. This base is shown in Figure 26.

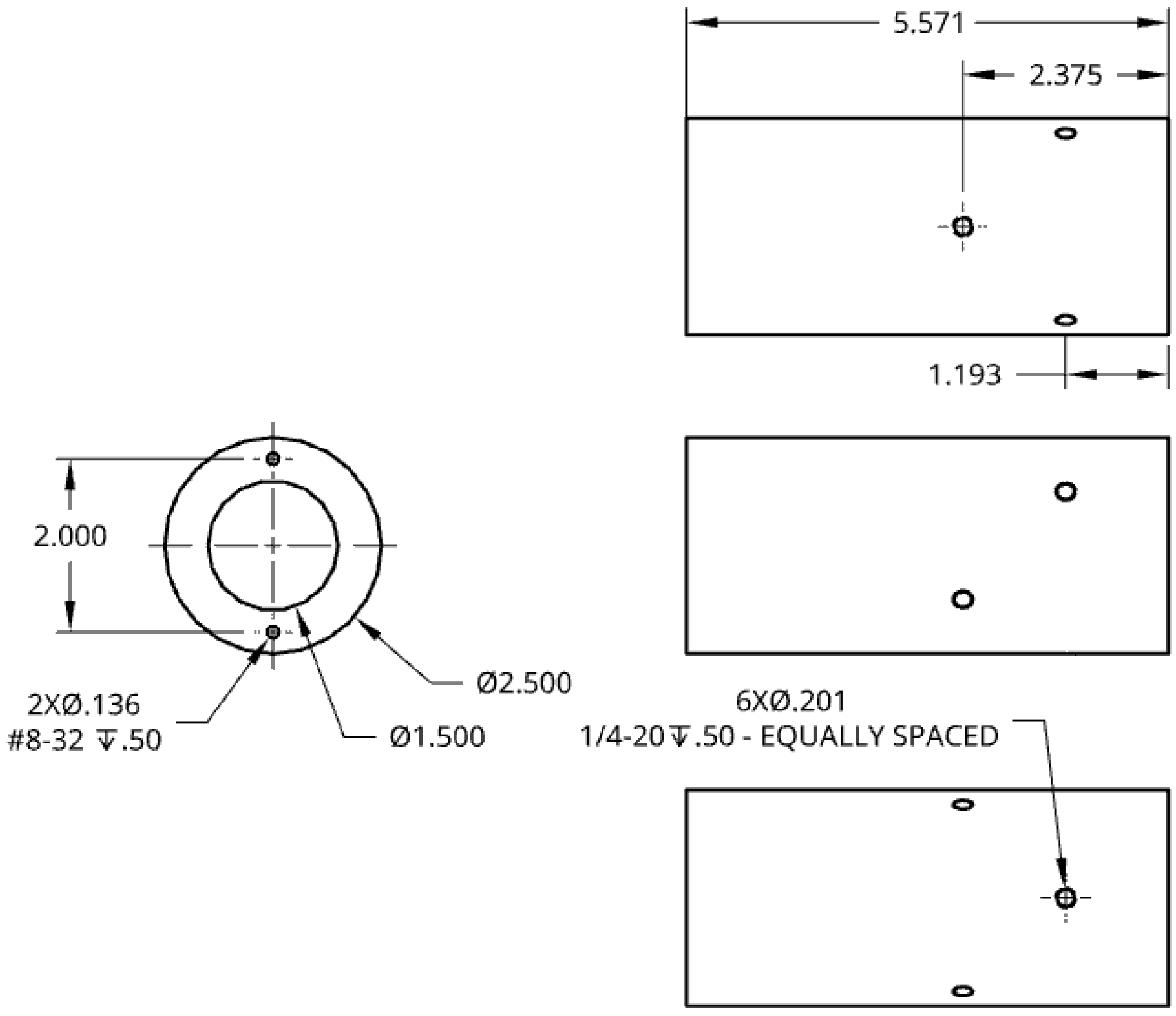


Figure 26 Machined Aluminum Base Drawing

There are several off-the-shelf Raspberry Pi enclosures but many do not have two features required for this application. First, the enclosure needed to mount to the end of the tube. This mounting configuration might have been as easy as drilling and countersinking holes into an existing enclosure. Second, the enclosure needed to support a vertical Pi Hat board with the GPS and switch. Based on an already designed two-piece Raspberry Pi enclosure for a camera project, the unique GPIO Pi hat opening was added. The new custom design was then 3D printed as is shown in Figure 27.

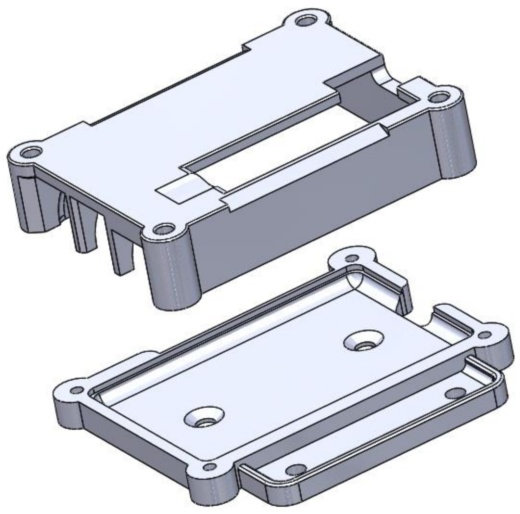


Figure 27 Custom Raspberry Pi Housing

The custom Raspberry Pi Hat was needed to allow the device to function without a keyboard, mouse, or display. The keyboard and mouse were replaced with a single momentary switch. The display was replaced with a series of colored LEDs. A small buzzer was added to add some audible feedback during switch pushes. The simple switch, LEDs, and buzzer have the added benefit of less distraction to the driver during data logging. The hat also mounted the GPS module that collected the angular rate data. The schematic of the custom Pi Hat is shown in Figure 28 and the component side of the printed wiring board is shown in Figure 29. Both the schematic and board layout were done in Eagle PCB.

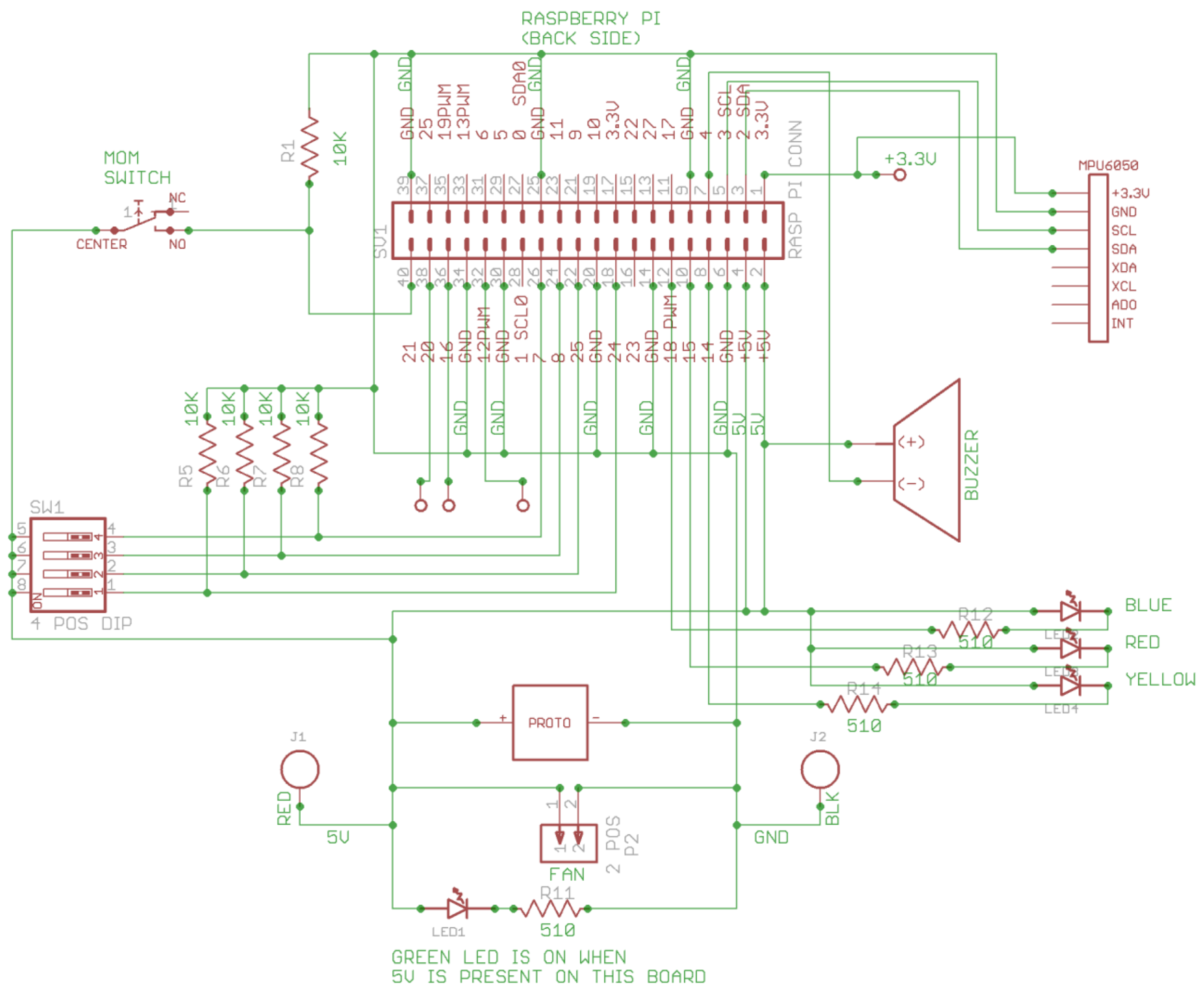


Figure 28 Custom Raspberry Pi Hat Schematic

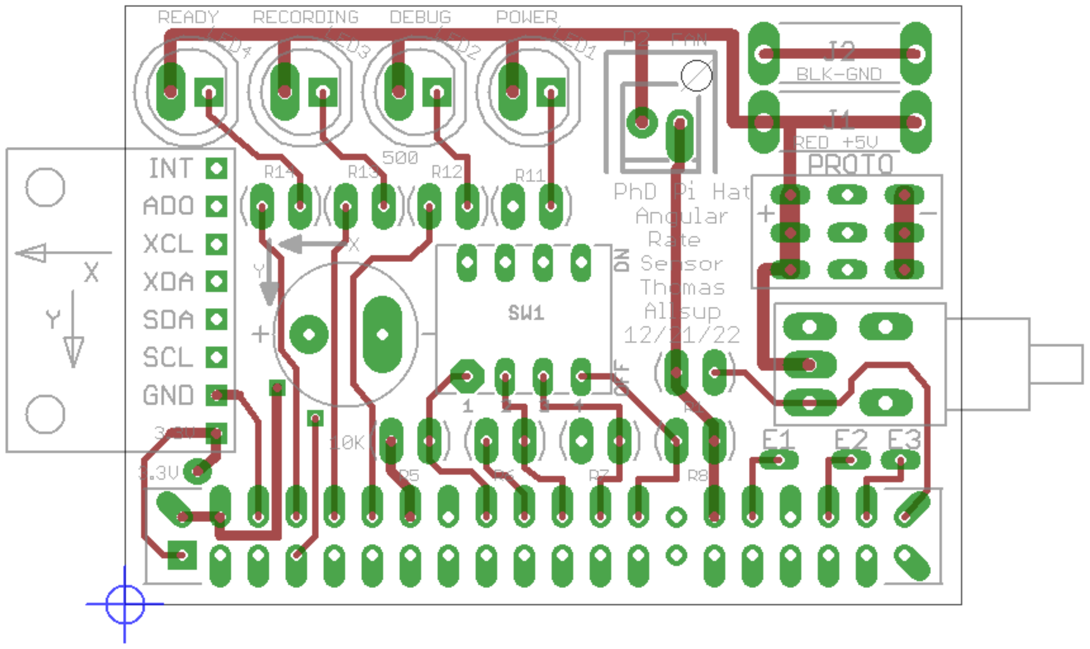


Figure 29 Custom Raspberry Pi Printed Wiring Board

There were five other areas of the hat that were not used or not populated during this experiment but allow for future use of the hat for other applications:

1. Two tip jack connectors for +5 Volts and Ground to assist debugging.
2. Future 5V fan connector if the device was going to be used in a hot environment.
3. Four Dual Inline Package (DIP) switches were designed to allow for future configuration changes even in "headless" operation.
4. Three pins terminated in pads to allow for additional future connections.
5. Prototype area for adding circuitry for future connections.

The custom data logging program was written in Python. The program automatically starts after the Raspberry Pi is powered on which takes about thirty seconds to boot. Once running, the yellow LED indicates the device is ready to record data. The momentary switch is pushed and the red LED indicates the device is recording data. Once the data is complete, the momentary switch is pressed again and the yellow LED indicates the device is ready to records again. The listing of the program is presented in Code 4.

Code 4 Raspberry Pi Python Code

```
#!/usr/bin/env python
# AngularDataLogger.py
# Thomas Allsup 1/5/2023
# Designed for "headless" operation without keyboard, mouse, or display
# This program uses a custom PWB hat that has a GPS module, beeper,
# four LEDs, and switch
# This program reads in y axis angular data and saves to a datalog file
# The green LED on the main hat and the small red LED on the GPS
# are always on
# Yellow LED lights up when waiting for switch
# Red LED lights up when recording
# Blue LED is for future use
# Yellow, Red, and Blue LED's briefly light up prior to each
# waiting for switch state

from mpu6050 import mpu6050
import time
from datetime import datetime
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM) # sets up how to number pins on Raspberry Pi

# Pin Definitions
BUZZER=4
BLUE=18
YELLOW=14
RED=15
BUTTON=21
DIP1=24 # Not populated on this PWB
DIP2=25 # Not populated on this PWB
DIP3=8 # Not populated on this PWB
DIP4=7 # Not populated on this PWB

GPIO.setwarnings(False)
GPIO.setup(BUTTON,GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(BUZZER,GPIO.OUT)
GPIO.setup(BLUE,GPIO.OUT)
GPIO.setup(RED,GPIO.OUT)
GPIO.setup(YELLOW, GPIO.OUT)
GPIO.setup(DIP1,GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(DIP2,GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(DIP3,GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(DIP4,GPIO.IN, pull_up_down = GPIO.PUD_UP)
sensor = mpu6050(0x68)
while 1==True:
    # Flash all LEDs
    GPIO.output(BUZZER, False)
    GPIO.output(BLUE, False)
    GPIO.output(RED, False)
    GPIO.output(YELLOW, False)
    time.sleep(.5)
    GPIO.output(BUZZER, True)
    GPIO.output(BLUE, True)
    GPIO.output(RED, True)
    GPIO.output(YELLOW, True)
    time.sleep(.5)
    GPIO.output(YELLOW, False) # turn on the Yellow Ready LED
    print("Waiting for Switch")
    while GPIO.input(BUTTON) == False:
        time.sleep(.01)
    GPIO.output(YELLOW, True) # turn off the Yellow Ready LED
    GPIO.output(RED, False) # turn on the Red Ready LED
    GPIO.output(BUZZER, False)
    nowfile = datetime.now()
    file=open("/home/pi/Desktop/log_number.dat","r")
    datalogcount=int(file.read()) # reads in the next datalog
                                  # number from a data file
    file.close()
    datalogcount=datalogcount+1
    file=open("/home/pi/Desktop/log_number.dat","w")
    file.write(str(datalogcount)+"\n") # writes the next datalog
                                       # number back to data file
    file.close()
    file=open("/home/pi/Desktop/datalog"+str(datalogcount)+".txt","w")
    print("Starting Datalog "+str(datalogcount))
    file.write(str(nowfile)+"\n")
    time.sleep(.5)
    while GPIO.input(BUTTON) == True:
        time.sleep(.1)
    while GPIO.input(BUTTON) == False:
        accelerometer_data = sensor.get_accel_data()
        now = datetime.now()
        #print (accelerometer_data)
        a=str(now)+">""{0:.3f}".format(accelerometer_data["y"])
        file.write(a+"\n")
        print (a)
    GPIO.output(BUZZER, True)
    file.close()
    print("File closed")
    GPIO.output(BUZZER, False)
    GPIO.output(RED, True) # turn off RED recording LED
    time.sleep(.5)
    GPIO.output(BUZZER, True)
    while GPIO.input(BUTTON) == True:
        time.sleep(.1)
```

The data log filename is a concatenation of the words “Datalog” and a string of numbers. The data log numbers are an incremented counter that is saved in a separate file called “log_number.dat” so that filenames are never repeated even if the Raspberry Pi computer is rebooted. The data log file is a text file that has a single header line that indicates the date and time that the file was created. Each entry has the time that the data was collected followed by a “>” followed by the angular data. A short example of a data log file is shown in List 5.

List 5 Example Raspberry Pi Program Data Log File

```
2022-10-18 17:13:24.583664
2022-10-18 17:13:25.089593>-10.005
2022-10-18 17:13:25.092873>-10.125
2022-10-18 17:13:25.096060>-10.092
2022-10-18 17:13:25.099277>-10.101
2022-10-18 17:13:25.102563>-10.127
2022-10-18 17:13:25.105839>-9.967
2022-10-18 17:13:25.109117>-10.185
```

APPENDIX 5

DESIGN AND CONSTRUCTION OF A HEADLAMP SIMULATOR

The demonstration of the inertially-stabilized headlamp was simulated with a combined physical electrical and mechanical breadboard as shown in Figure 30. The demonstrator simulator uses a Raspberry Pi with a custom PCB that receives angular position data from the Bosch sensor and the potentiometer and outputs the motor controls through an off-the-shelf full bridge motor driver as well as driving the laser cross-hair.

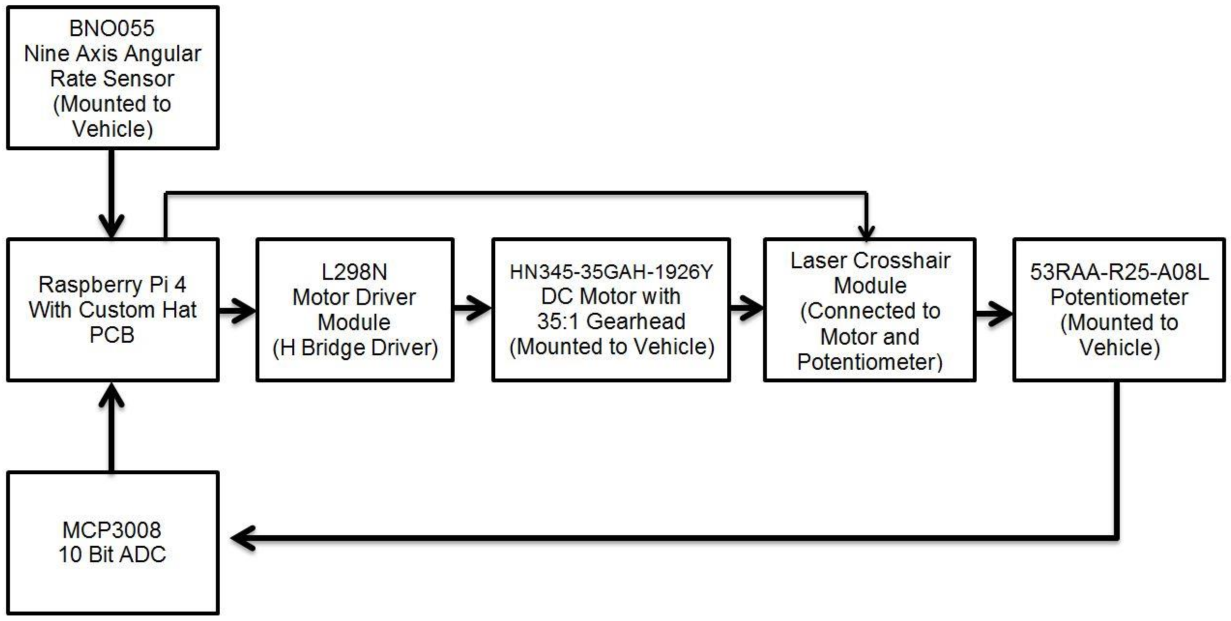

BNO055
Nine Axis Angular Rate Sensor (Mounted to Vehicle)
Raspberry Pi 4 With Custom Hat PCB
L298N Motor Driver Module (H Bridge Driver)
HN345-35GAH-1926Y DC Motor with 35:1 Gearhead (Mounted to Vehicle)
Laser Crosshair Module (Connected to Motor and Potentiometer)
53RAA-R25-A08L Potentiometer (Mounted to Vehicle)
MCP3008 10 Bit ADC

Figure 30 Block Diagram of Breadboard

The mechanical portion of the system is shown as a CAD model in Figure 31 and a photograph in Figure 32. The mechanical portion has a custom designed 3D printed base that provides linear and angular alignment between the DC gearmotor and potentiometer as is shown in Figure 33. Features to polarize the potentiometer and motor are integral to the single piece base. The base also includes features so the device can be mounted to a vehicle. The cross-hair laser mount must secure three features; the motor output shaft, the cross-hair laser module, and the potentiometer shaft. Because the two shafts are rigidly attached to the base, a custom 3D printed "clam shell" mounting component was required. The custom part was designed so that the same component formed both halves of the laser mount as shown in Figure 34. The part used the same "V" shape to capture both 6mm shafts of the gearmotor and the potentiometer and ensure there was no misalignment. A larger "V" shape captures the laser cross-hair module. The clamshells had hexagonal recesses to capture hex nuts on one side and screws on the opposite side to "squeeze" the clamshell together.

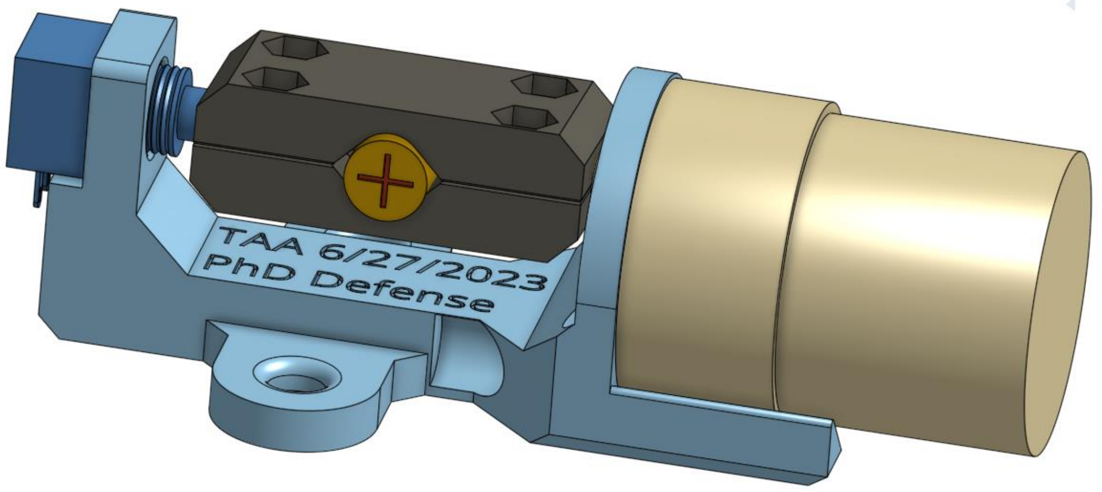


Figure 31 Mechanical Design of Breadboard

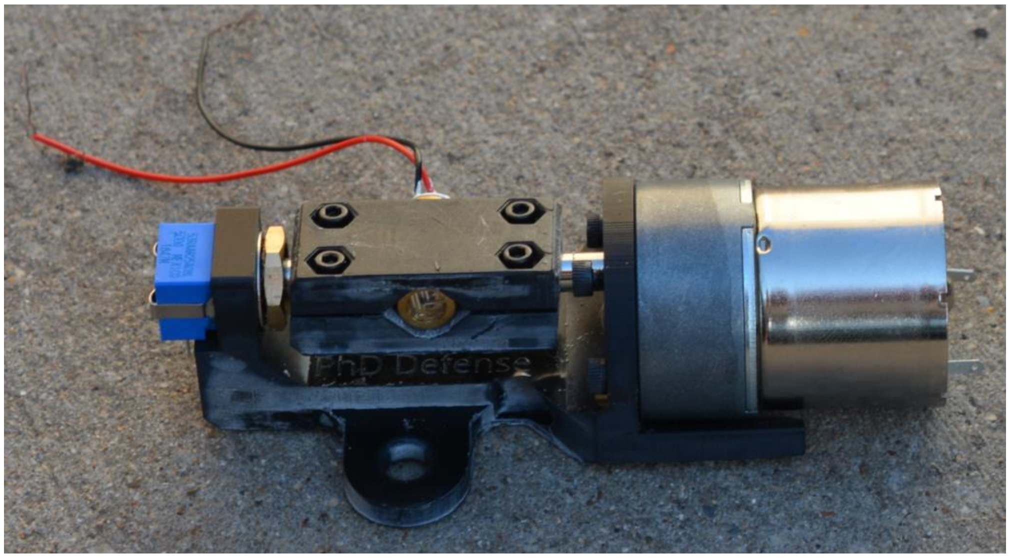


Figure 32 System Photograph

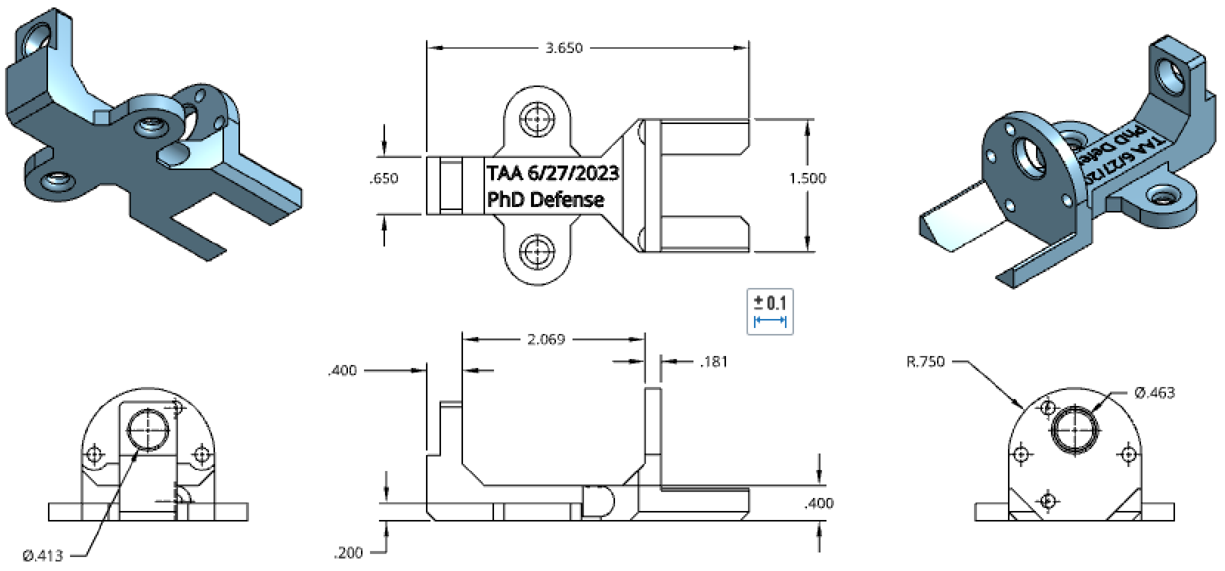


Figure 33 Mechanical Breadboard Base

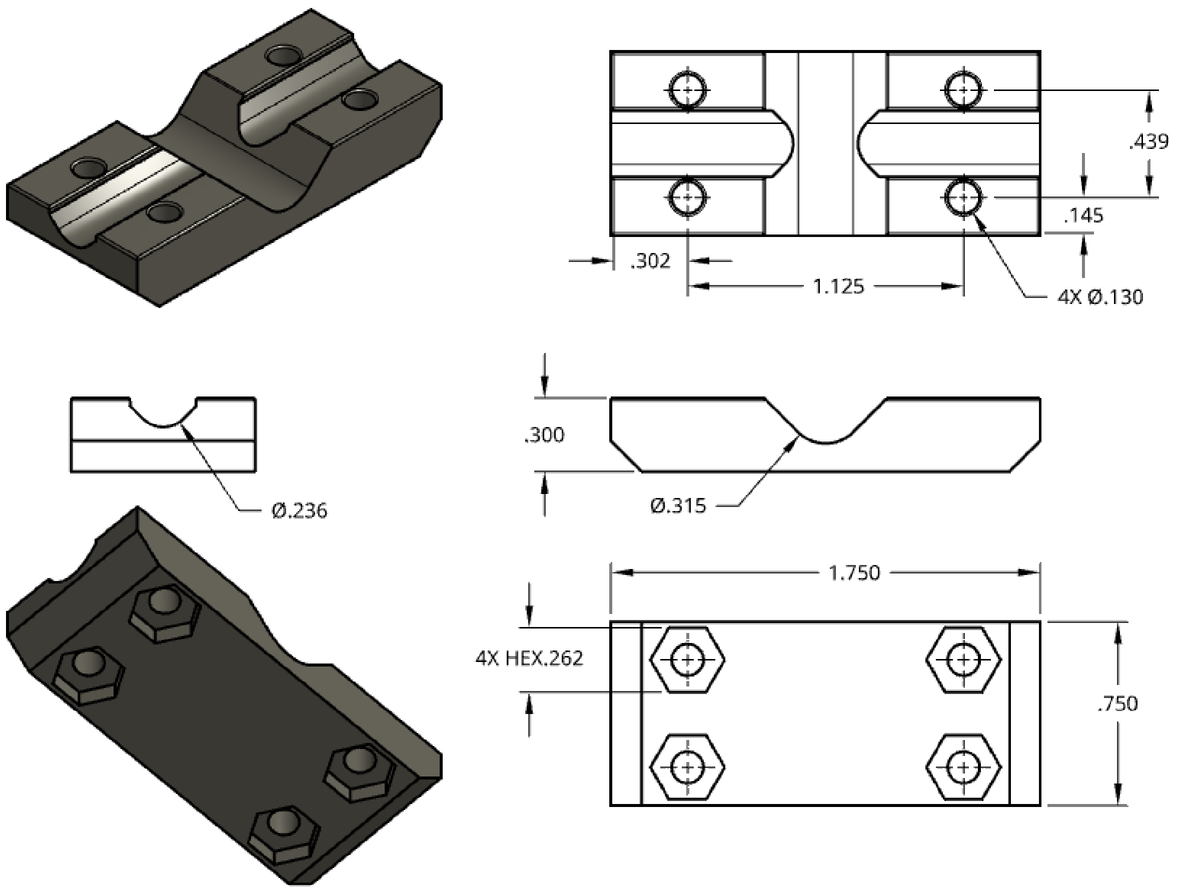
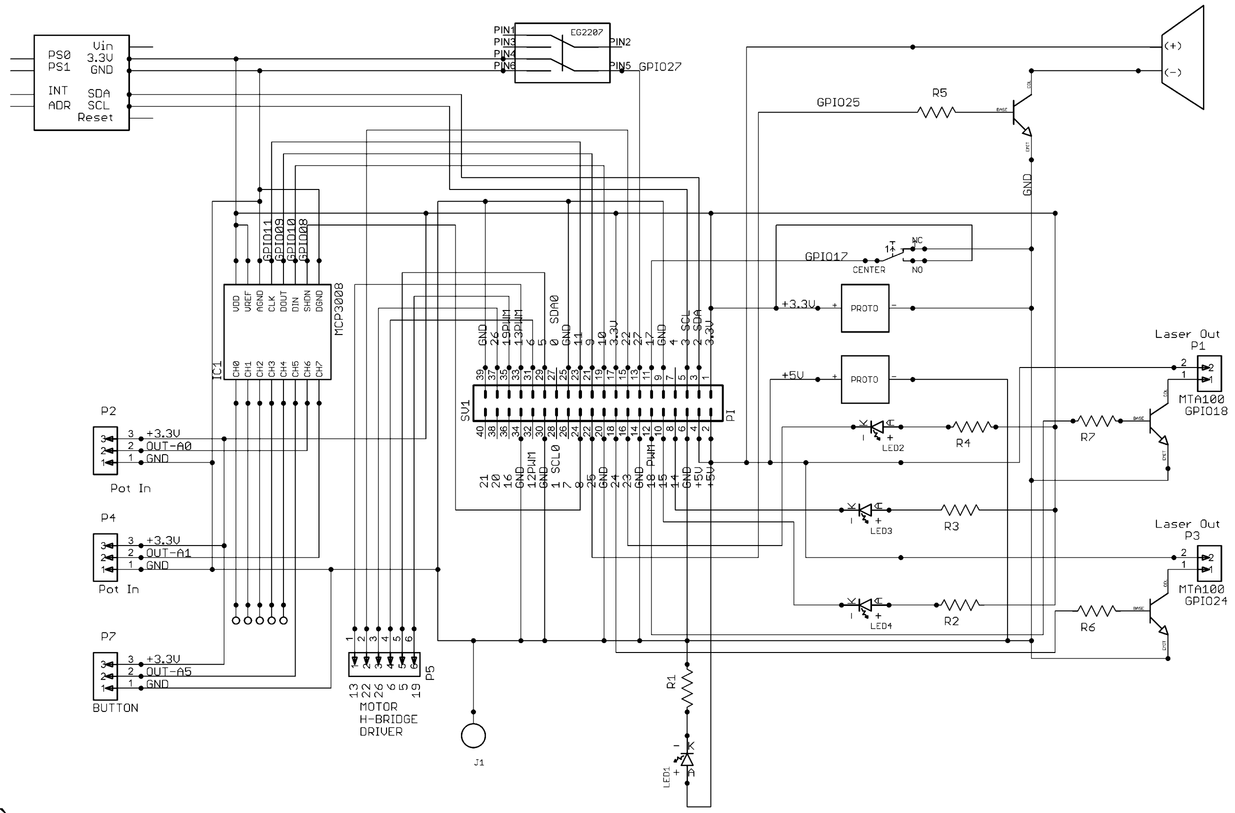


Figure 34 Breadboard Laser Mount Clamshell Half

The electrical portion of the simulator requires a Raspberry Pi 3 or 4 with a custom Hat PCB and motor drive module. The custom Hat PCB attaches to the Raspberry Pi GPIO pins as shown in the schematic in Figure 35 and the top side silkscreen in Figure 36. The Pi Hat has a buzzer to alert the user if used while driving and an integral LED to indicate power status along with three programmable LEDs. There is a slider switch and a push momentary button that can control any operation. The Pi Hat has connections out to the L298N Motor Driver Module (H Bridge Driver) that is a full bridge motor driver which allows up to two DC motors to be driven with a separate 12 Volt power supply such as a car battery. Appendix 7 explores the DC motor and driver parameters available for the proof-of-concept model. The Pi Hat has two separate 5V laser driver connections. The Pi Hat also has two separate potentiometer connections that feed into an eight channel Analog to Digital module. There is a future optional six button input connector that connects to the ADC. Finally, the Pi Hat also has a connection for the Bosch Angular Rate Sensor BNO055 module as well.

The use of the rotary potentiometer for angular feedback has a major benefit. The potentiometer provides an absolute encoder so the system knows the angle at all times including at start-up. If a relative encoder was used for feedback there would need to be limit or home switches and the motor would need to exercise the system every time power was removed from the controller [15].



`

Figure 35 Breadboard Custom PCB Schematic

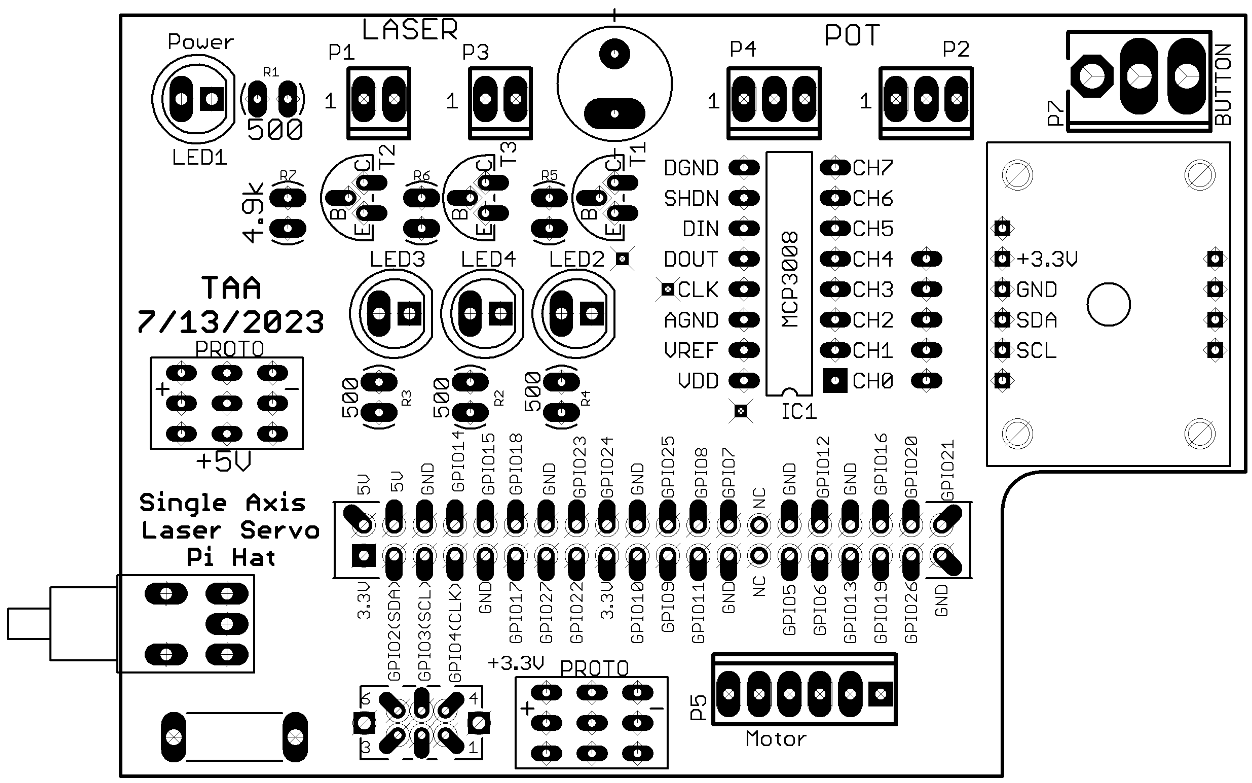


Figure 36 Breadboard Custom PCB Layout

PCB Pinout Definitions

The following pins are connected to the Raspberry Pi:GPIO18 Laser P1

GPIO24 Laser P3	GPIO10 ADC-DIN	GPIO22 Motor A Dir
GPIO08 ADC - Shutdown	GPIO11 ADC-Clock	GPIO26 Motor A DIr
GPIO09 ADC-DOUT	GPIO13 Motor A En (PWM)	GPIO18 Motor B En (PWM)
GPIO06 Motor B Dir	GPIO25 Beeper	GPIO17 Push Button
GPIO05 Motor B Dir	GPIO27 Slider Switch	

The following inputs are connected to the A2D Converter:

A0 Pot P2 A1 Pot P4 A2 External Buttons (Optional) A3-A7 Solder Pads

The Python Graphical User Interface uses a simple TKInter screen, shown in Figure 37, that allows for virtual button inputs for functionality and text output for variables for the motor, potentiometer, and other parameters.   The program code is shown in Code 6.

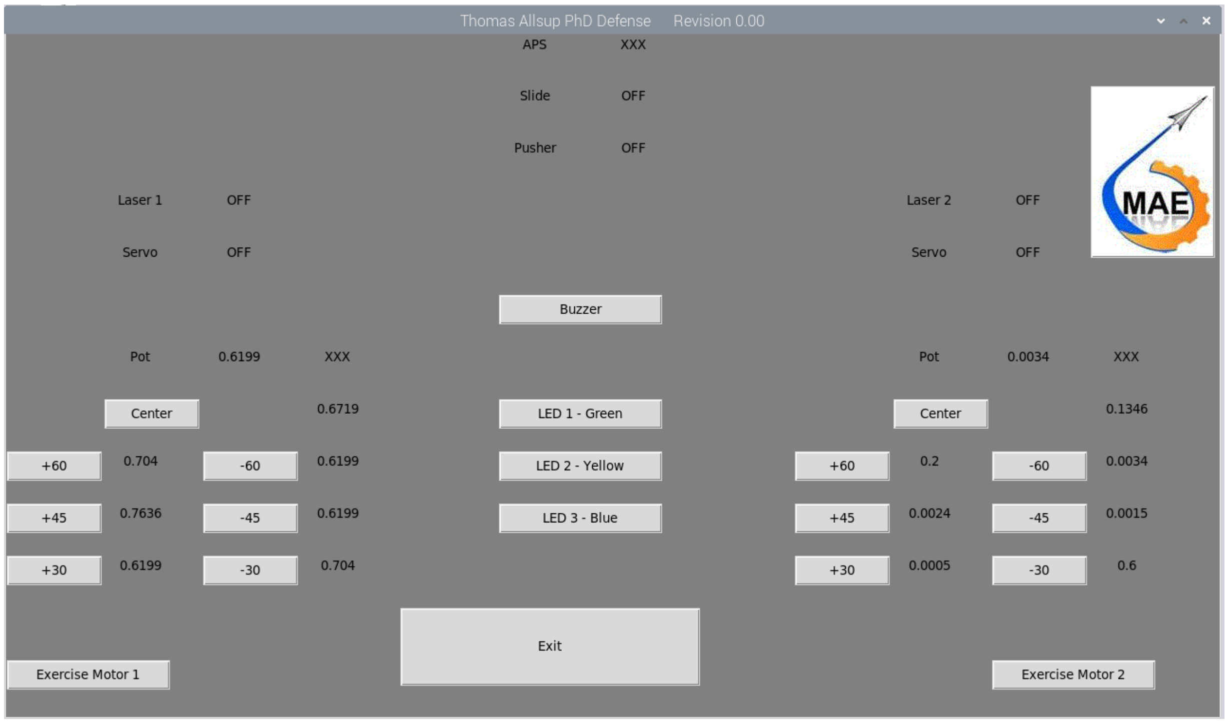


Figure 37 Headlamp Simulator Graphical User Interface

Code 6 Headlamp Simulator Python Program Code

```
# !/usr/bin/python3
from tkinter import *
import tkinter.ttk as ttk
import time
from datetime import datetime
from gpiozero import MCP3008
import RPi.GPIO as GPIO
import numpy as np

# ****************** CONSTANTS *********************
SWRevision="0.00"
# resolution is for Raspberry Pi
RESOLUTION="1274x680+3+1" # +1,+1 is top left corner
#RESOLUTION="800x600+1+1"
LineDist=int(680/13)
LINE01 = LineDist*0+1
LINE02 = LineDist*1
LINE03 = LineDist*2
```

```
LINE04 = LineDist*3
LINE05 = LineDist*4
LINE06 = LineDist*5
LINE07 = LineDist*6
LINE08 = LineDist*7
LINE09 = LineDist*8
LINE10 = LineDist*9
LINE11 = LineDist*10
LINE12 = LineDist*11
LINE13 = LineDist*12
CharWidth=8
ColDist = int(1274/(12*CharWidth))
COL01 = ColDist*0*CharWidth+1
COL02 = ColDist*1*CharWidth
COL03 = ColDist*2*CharWidth
COL04 = ColDist*3*CharWidth
COL05 = ColDist*4*CharWidth
COL06 = ColDist*5*CharWidth
COL07 = ColDist*6*CharWidth
COL08 = ColDist*7*CharWidth
COL09 = ColDist*8*CharWidth
COL10 = ColDist*9*CharWidth
COL11 = ColDist*10*CharWidth
COL12 = ColDist*11*CharWidth
BUTTON_WIDTH_SINGLE = int(ColDist*.6)
BUTTON_WIDTH_DOUBLE = int(BUTTON_WIDTH_SINGLE*1.9)
BUTTON_WIDTH_THIRD = int(BUTTON_WIDTH_SINGLE*3.6)
BUTTON_WIDTH_HALF = int(BUTTON_WIDTH_SINGLE*5.5)
BUTTON_WIDTH_FULL = int(BUTTON_WIDTH_SINGLE*11)
BUTTON_HEIGHT = 1
BUTTON_HEIGHT_DOUBLE = 4
ENTRY_WIDTH_SINGLE = ColDist
ENTRY_WIDTH_THIRD = ColDist
ENTRY_WIDTH_HALF = ColDist
ENTRY_WIDTH_FULL = ColDist
#GPIO
inA1 = 22
inA2 = 26
enA = 13
inB1 = 5
inB2 = 6
enB =18
Buzzer=25
Laser1=18
Laser2=24
Switch_Slider = 27
Switch_Button = 17
LED1 = 14
LED2 = 15
LED3 = 23
temp1=1
GPIO.setmode(GPIO.BCM)
GPIO.setup(Buzzer,GPIO.OUT)
GPIO.output(Buzzer, GPIO.LOW)
GPIO.setup(LED1,GPIO.OUT)
GPIO.output(LED1, GPIO.HIGH)
GPIO.setup(LED2,GPIO.OUT)
GPIO.output(LED2, GPIO.HIGH)
GPIO.setup(LED3,GPIO.OUT)
GPIO.output(LED3, GPIO.HIGH)
GPIO.setup(Laser1,GPIO.OUT)
GPIO.output(Laser1, GPIO.HIGH)
GPIO.setup(Laser2,GPIO.OUT)
GPIO.output(Laser2, GPIO.HIGH)
GPIO.setup(inA1,GPIO.OUT)
GPIO.setup(inA2,GPIO.OUT)
GPIO.setup(enA,GPIO.OUT)
GPIO.output(inA1,GPIO.LOW)
GPIO.output(inA2,GPIO.HIGH)
p=GPIO.PWM(enA,50)
GPIO.setup(inB1,GPIO.OUT)
GPIO.setup(inB2,GPIO.OUT)
GPIO.setup(enB,GPIO.OUT)
GPIO.output(inB1,GPIO.LOW)
GPIO.output(inB2,GPIO.HIGH)
GPIO.output(enB,GPIO.HIGH)
GPIO.setup(Switch_Slider,GPIO.IN)
GPIO.setup(Switch_Button,GPIO.IN)
pot_cen=[.5,.2,.8,.3,.7,.4,.6,.5,.2,.8,.3,.7,.4,.6]
# *************************************************
frame = Tk()
frame.title("Thomas Allsup PhD Defense      Revision "+SWRevision)
frame.geometry(RESOLUTION)
frame.config(width=1280, height=720, bg="gray")
frame.resizable(width=False, height=False)
def ComBuzzer():
    GPIO.output(Buzzer,1)
    time.sleep(.25)
    GPIO.output(Buzzer,0)
    time.sleep(.15)
def ComExit():
    GPIO.cleanup()
    exit("Exited Program")
def ComLED1():
    GPIO.output(LED1,GPIO.LOW)
    time.sleep(.5)
    GPIO.output(LED1,GPIO.HIGH)
def ComLED2():
    GPIO.output(LED2,GPIO.LOW)
    time.sleep(.5)
    GPIO.output(LED2,GPIO.HIGH)
def ComLED3():
    GPIO.output(LED3,GPIO.LOW)
    time.sleep(.5)
    GPIO.output(LED3,GPIO.HIGH)
def ComPot1P60():
    pot1v=MCP3008(7)
    pot1vv=pot1v.value
    pot1s=str(int(pot1vv*10000+.5)/10000)
    pot_cen[1]=pot1s
    LowerTitle10 = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL02,y = LINE09)
def ComPot1P45():
    pot1v=MCP3008(7)
    pot1vv=pot1v.value
    pot1s=str(int(pot1vv*10000+.5)/10000)
    pot_cen[2]=pot1s
    LowerTitle10 = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL02,y = LINE10)
def ComPot1P30():
    pot1v=MCP3008(7)
    pot1vv=pot1v.value
    pot1s=str(int(pot1vv*10000+.5)/10000)
    pot_cen[3]=pot1s
    LowerTitle10 = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL02,y = LINE11)
def ComPot1M60():
    pot1v=MCP3008(7)
    pot1vv=pot1v.value
    pot1s=str(int(pot1vv*10000+.5)/10000)
    pot_cen[4]=pot1s
    LowerTitle10 = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL04,y = LINE09)
def ComPot1M45():
    pot1v=MCP3008(7)
    pot1vv=pot1v.value
    pot1s=str(int(pot1vv*10000+.5)/10000)
    pot_cen[5]=pot1s
    LowerTitle10 = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL04,y = LINE10)
def ComPot1M30():
    pot1v=MCP3008(7)
    pot1vv=pot1v.value
    pot1s=str(int(pot1vv*10000+.5)/10000)
    pot_cen[6]=pot1s
    LowerTitle10 = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL04,y = LINE11)
def ComPot2P60():
    pot1v=MCP3008(6)
    pot1vv=pot1v.value
    pot1s=str(int(pot1vv*10000+.5)/10000)
    pot_cen[8]=pot1s
    LowerTitle10 = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL10,y = LINE09)
def ComPot2P45():
    pot1v=MCP3008(6)
    pot1vv=pot1v.value
    pot1s=str(int(pot1vv*10000+.5)/10000)
    pot_cen[9]=pot1s
    LowerTitle10 = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL10,y = LINE10)
def ComPot2P30():
    pot1v=MCP3008(6)
    pot1vv=pot1v.value
    pot1s=str(int(pot1vv*10000+.5)/10000)
    pot_cen[10]=pot1s
    LowerTitle10 = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL10,y = LINE11)
def ComPot2M60():
    pot1v=MCP3008(6)
    pot1vv=pot1v.value
    pot1s=str(int(pot1vv*10000+.5)/10000)
    pot_cen[11]=pot1s
    LowerTitle10 = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL12,y = LINE09)
def ComPot2M45():
    pot1v=MCP3008(6)
    pot1vv=pot1v.value
    pot1s=str(int(pot1vv*10000+.5)/10000)
    pot_cen[12]=pot1s
    LowerTitle10 = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL12,y = LINE10)
def ComPot2M30():
    pot1v=MCP3008(6)
    pot1vv=pot1v.value
    pot1s=str(int(pot1vv*10000+.5)/10000)
    pot_cen[13]=pot1s
    LowerTitle10 = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL12,y = LINE11)
def ComCenter1():
    Avg_Cen1=(pot_cen[1]+pot_cen[4])/2
    Avg_Cen2=(pot_cen[2]+pot_cen[5])/2
    Avg_Cen3=(pot_cen[3]+pot_cen[6])/2
    Avg_Cen=(Avg_Cen1+Avg_Cen2+Avg_Cen3)/3
    Avg_Cen_Str=str(int(Avg_Cen*10000+.5)/10000)
    LowerTitle16 = Label(frame, text = Avg_Cen_Str,
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL04,y = LINE08)
    pot_cen[0]=Avg_Cen_Str
    file=open("PotCenters.bin","wb")
    np.save(file,pot_cen)
    file.close()
def ComCenter2():
    Avg_Cen1=(pot_cen[8]+pot_cen[11])/2
    Avg_Cen2=(pot_cen[9]+pot_cen[12])/2
    Avg_Cen3=(pot_cen[10]+pot_cen[13])/2
    Avg_Cen=(Avg_Cen1+Avg_Cen2+Avg_Cen3)/3
    Avg_Cen_Str=str(int(Avg_Cen*10000+.5)/10000)
    LowerTitle16 = Label(frame, text = Avg_Cen_Str,
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL12,y = LINE08)
    pot_cen[7]=Avg_Cen_Str
    file=open("PotCenters.bin","wb")
    np.save(file,pot_cen)
    file.close()
def UnknownCommand():
    pass
def gui():
    Bit1=GPIO.input(Switch_Slider)
    Bit2=GPIO.input(Switch_Button)
    LowerTitle1A = Label(frame, text = "XXX", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL07,y = LINE01)
    if (Bit1==0):
        YesNo1="OFF"
    else:
        YesNo1="ON "
    if (Bit2==0):
        YesNo2="OFF"
    else:
        YesNo2="ON "
    LowerTitle2A = Label(frame, text = YesNo1, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL07,y = LINE02)
    LowerTitle3A = Label(frame, text = YesNo2, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL07,y = LINE03)
    pot1v=MCP3008(7)
    pot1vv=pot1v.value
    pot1s=str(int(pot1vv*10000+.5)/10000)
    Pot1Title = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL03,y = LINE07)
    pot2v=MCP3008(6)
    pot2vv=pot2v.value
    pot2s=str(int(pot2vv*10000+.5)/10000)
    Pot2Title = Label(frame, text = pot2s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL11,y = LINE07)
    pot1ang=(pot_cen[0]-pot1vv)*268.3443
    pot2ang=(pot_cen[7]-pot2vv)*268.3443
    pot1s=str(int(pot1ang*10+.5)/10)
    pot2s=str(int(pot2ang*10+.5)/10)
    LowerTitle19 = Label(frame, text = pot1s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL04,y = LINE07)
    LowerTitle29 = Label(frame, text = pot2s, width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL12,y = LINE07)
def updater():
    gui()
    frame.after(1000, updater)
LowerTitle1 = Label(frame, text = "APS", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL06,y = LINE01)
LowerTitle2 = Label(frame, text = "Slide", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL06,y = LINE02)
LowerTitle3 = Label(frame, text = "Pusher", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL06,y = LINE03)
LowerTitle4 = Label(frame, text = "Laser 1", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL02,y = LINE04)
LowerTitle5 = Label(frame, text = "Servo", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL02,y = LINE05)
LowerTitle6 = Label(frame, text = "Laser 2", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL10,y = LINE04)
LowerTitle7 = Label(frame, text = "Servo", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL10,y = LINE05)
LowerTitle4A = Label(frame, text = "OFF", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL03,y = LINE04)
LowerTitle5A = Label(frame, text = "OFF", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL03,y = LINE05)
LowerTitle6A = Label(frame, text = "OFF", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL11,y = LINE04)
LowerTitle7A = Label(frame, text = "OFF", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL11,y = LINE05)
LowerTitle17 = Label(frame, text = "Pot", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL02,y = LINE07)
LowerTitle27 = Label(frame, text = "Pot", width=BUTTON_WIDTH_SINGLE,
bg="gray").place(x = COL10,y = LINE07)
BUTTON10=Button(frame,text="+60",command=ComPot1P60,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL01,y=LINE09)
BUTTON11=Button(frame,text="+45",command=ComPot1P45,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL01,y=LINE10)
BUTTON12=Button(frame,text="+30",command=ComPot1P30,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL01,y=LINE11)
```

```
BUTTON13=Button(frame,text="-60",command=ComPot1M60,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL03,y=LINE09)
BUTTON14=Button(frame,text="-45",command=ComPot1M45,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL03,y=LINE10)
BUTTON15=Button(frame,text="-30",command=ComPot1M30,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL03,y=LINE11)
BUTTON16=Button(frame,text="Center",command=ComCenter1,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL02,y=LINE08)
BUTTON17=Button(frame,text="LED 1 - Green",command=ComLED1,
width=BUTTON_WIDTH_DOUBLE,height=BUTTON_HEIGHT).place(x=COL06,y=LINE08)
BUTTON18=Button(frame,text="LED 2 - Yellow",command=ComLED2,
width=BUTTON_WIDTH_DOUBLE,height=BUTTON_HEIGHT).place(x=COL06,y=LINE09)
BUTTON19=Button(frame,text="LED 3 - Blue",command=ComLED3,
width=BUTTON_WIDTH_DOUBLE,height=BUTTON_HEIGHT).place(x=COL06,y=LINE10)
BUTTON20=Button(frame,text="+60",command=ComPot2P60,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL09,y=LINE09)
BUTTON21=Button(frame,text="+45",command=ComPot2P45,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL09,y=LINE10)
BUTTON22=Button(frame,text="+30",command=ComPot2P30,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL09,y=LINE11)
BUTTON23=Button(frame,text="-60",command=ComPot2M60,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL11,y=LINE09)
BUTTON24=Button(frame,text="-45",command=ComPot2M45,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL11,y=LINE10)
BUTTON25=Button(frame,text="-30",command=ComPot2M30,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL11,y=LINE11)
BUTTON26=Button(frame,text="Center",command=ComCenter2,
width=BUTTON_WIDTH_SINGLE,height=BUTTON_HEIGHT).place(x=COL10,y=LINE08)
BUTTON27=Button(frame,text="Exercise Motor 1",command=ComBuzzer,
width=BUTTON_WIDTH_DOUBLE,height=BUTTON_HEIGHT).place(x=COL01,y=LINE13)
BUTTON28=Button(frame,text="Exercise Motor 2",command=ComBuzzer,
width=BUTTON_WIDTH_DOUBLE,height=BUTTON_HEIGHT).place(x=COL11,y=LINE13)
BUZZER_BUTTON=Button(frame,text="Buzzer",command=ComBuzzer,
width=BUTTON_WIDTH_DOUBLE,height=BUTTON_HEIGHT).place(x=COL06,y=LINE06)
FrameExit=Button(frame,text="Exit",command=ComExit,width=BUTTON_WIDTH_THIRD,h
eight=BUTTON_HEIGHT_DOUBLE).place(x=COL05,y=LINE12)
file=open("PotCenters.bin","rb")
pot_cen = np.load(file)
file.close()
LowerTitle16 = Label(frame, text = str(pot_cen[0]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL04,y = LINE08)
LowerTitle10 = Label(frame, text = str(pot_cen[1]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL02,y = LINE09)
LowerTitle11 = Label(frame, text = str(pot_cen[2]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL02,y = LINE10)
LowerTitle12 = Label(frame, text = str(pot_cen[3]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL02,y = LINE11)
LowerTitle13 = Label(frame, text = str(pot_cen[4]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL04,y = LINE09)
LowerTitle14 = Label(frame, text = str(pot_cen[5]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL04,y = LINE10)
LowerTitle15 = Label(frame, text = str(pot_cen[6]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL04,y = LINE11)
LowerTitle26 = Label(frame, text = str(pot_cen[7]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL12,y = LINE08)
LowerTitle20 = Label(frame, text = str(pot_cen[8]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL10,y = LINE09)
```

```
LowerTitle21 = Label(frame, text = str(pot_cen[9]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL10,y = LINE10)
LowerTitle22 = Label(frame, text = str(pot_cen[10]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL10,y = LINE11)
LowerTitle23 = Label(frame, text = str(pot_cen[11]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL12,y = LINE09)
LowerTitle24 = Label(frame, text = str(pot_cen[12]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL12,y = LINE10)
LowerTitle25 = Label(frame, text = str(pot_cen[13]),
width=BUTTON_WIDTH_SINGLE, bg="gray").place(x = COL12,y = LINE11)
gui()
logo=PhotoImage(file="MAE_Logo.gif")
HelpScreen=Button(frame,relief=RAISED,image=logo,command=UnknownCommand,
width=125, height=175).place(x=COL12,y=LINE02)
updater()
frame.mainloop()
```

APPENDIX 6

ROADWAY DATA CREATOR

Despite collecting actual roadway data from various vehicles using the device described in Appendix 4, it was desired to produce a specific data file that described an idealized roadway instead of depending upon finding a series of actual roadways.

Drawing the desired roadway in CAD such as shown in Appendix 1 for the simulator was the natural first step but it proved difficult to divide the roadway into equal length line segments and export a comma separated data file. An attempt was made to export the 3D data into a 2D cad system, adding equal distance data points, exporting a DXF and then converting this to 2D data file. Automating any portion of this method proved difficult or impossible so the method was abandoned.

The second attempt was to work inside completely in Excel®. It was difficult to make the gradual pitch changes with a constant radius and ending pitch as desired. Changes in one portion of the roadway were not propagated through the end of the data. This method was not intuitive nor effective but did drive the third method that proved successful.

The third and final method was to create a Python program, shown in Code 8, that takes in a short datafile that describes the roadway and quickly produces a comma separate value text file. Changing the input file allows for the output to be updated quickly. A sample roadway file with several segments is shown in List 7 and a plot of the output data which has 6000 data points is shown in Figure 38.

A future development would be to create a graphical user interface to enter segment information and have the program create the roadway dynamically.

List 7 Sample Roadway Input File

```
Roadway Test    1              6
275             0.000025       .005
600             -0.000015      -.006
750             0.00075        -.003
400             -0.0001        -.005
300             0.00002        0.003
400             -0.00003       0
600             0.0001         0.004
200             -0.0001        -0.002
300             -0.0002        -0.006
200             -0.0003        -0.002
100             0.0001         0
275             -0.00015       0
```

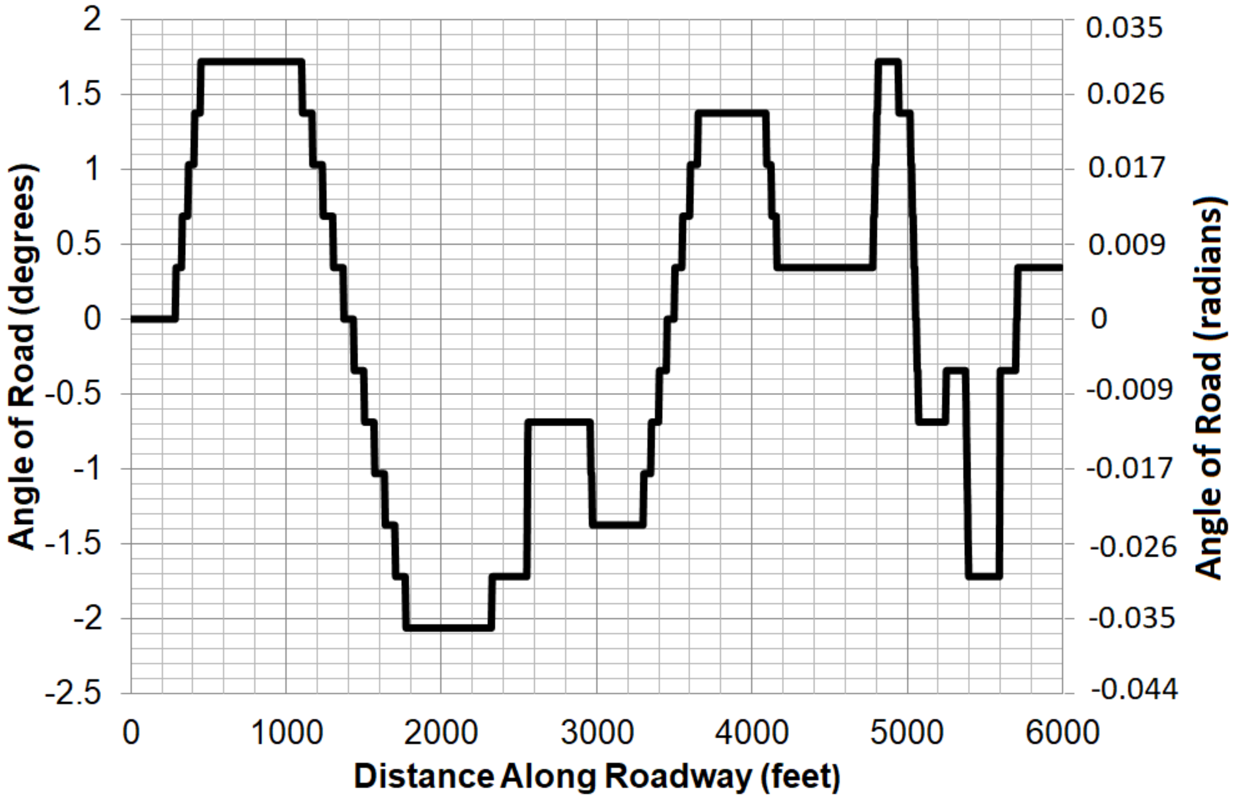


Figure 38 Sample Angle of Road Plot

Code 8 Roadway Data Creator Python Code

```
# Thomas Allsup 6/21/2023
#
# Reads in a tab separated file that describes a roadway
# changing slope and creates a CSV of the road with equal distances.
#
# Input file starts with header line.
#     File name, spacing between each x position, and
#        length of car (not used)
#     Starting slope is always 0
#   Each line (after header) has three parameters:
#     How long do you do you continue current road slope (line)?
#       How much do you want to change slope per step after straight?
#       What is the new road slope after the curvature?
#                 (where slope stops changing)
#
# There is no error checking on the input file.
#
# The output file is a true Comma Separated Value file and has distance
# along road (x), elevation (y), slope and change in slope.  The distance
# between each x is uniform.
#
import csv
with open('C:\Roadway.txt') as csv_file:
    f=open('C:\Roadway_Out.csv', 'w')
    csv_reader = csv.reader(csv_file, delimiter='\t')
    line_count = 0
    slope=0
    distance=0
    height=0
    for row in csv_reader:
        if line_count == 0:
            print('Road file: {}     Spacing:{}       Vehicle Length :
{}'.format(row[0], row[1],row[2]))
            f.write("Road file: "+row[0]+"   Spacing: "+row[1]+"   Vehicle
Length: "+row[2]+"\n")
            f.write("X, Height, Slope, Slope Change\n")
            spacing = float(row[1])
            line_count += 1
            slope=0
        else:
            print(f'Length {row[0]} curvature {row[1]} until {row[2]}.')
            line_count += 1
            straight=float(row[0])
# calculate the number of steps for straight
            straightcount = int(straight/spacing)
            for i in range(0, straightcount,1):
                distance = distance + spacing
                height = int(1000*(height + slope)+.5)/1000
                #road.append((distance,height, slope, 0))
                print(distance,height,slope,"0")
                f.write(str(distance)+","+str(height)+","+str(slope)+",0\n")
                curve=float(row[2])
                delta=float(row[1])
# calculate the number of steps for curve
curvecount = int ((curve-slope)/delta)
```

```
            for i in range(0, curvecount,1):
                distance = distance + spacing
                slope = slope + delta
                slopeout = int(1000*(slope)+.5)/1000
                height = int(1000*(height + slope)+.5)/1000
f.write(str(distance)+","+str(height)+","+str(slopeout)+","+str(delta)+"\n")
            slope = float(row[2])
    print(f'Processed {line_count} lines.')
    f.close()
```

APPENDIX 7

MOTOR UNIT CONVERSIONS

The DC gearmotor was purchased with a motor test report, as seen in Figure 39. The units included in this datasheet were in kg-cm so they were converted to English system units for the purpose of this dissertation. Several other needed parameters needed to be extracted to provide inputs to the simulation.

The gear ratio of the gearhead was listed in the filename and the part number as thirty-five to one. This gear ratio was confirmed by other documentation and experiments. Therefore, G_G is equal to 35. It should be noted that this means the motor turns 35 degrees as the output only turns one degree. This analysis treats the DC motor and the integral gearhead as a single unit so the gear ratio is irrelevant for the calculations.

The motor velocity constant, K_V, is the no-load speed divided by the motor voltage. The text states the no-load speed is 44 RPM but the table states 45 RPM so with a 12V power supply, the K_V is 3.67 to 3.75 RPM / V or 22.0 to 22.5 deg / V s.

The motor torque constant, K_T, is the current divided by the stall torque (listed at locked rotor on data sheet) is shown in Table 9. The value of K_T is 174.8 A / oz in.

Table 9 Motor Torque Conversions at 12VDC

Scenario	RPM	Current, A	Torque, Kg-cm	Torque, oz-in
No Load	45	0.046	0.000	0.000
Locked Rotor (Stalled)	0	0.845	10.000	139.0

The rotary inertia of the load at the motor, J, was determined by measuring the actual weight of the fully assembled laser mount as 0.672 ounces and using the measured linear dimensions of width of the load, 0.750 inches, and length of the load, 0.600 inches. The moment of inertia of the load, using Equation 21 is 0.0016 oz-in-s^2.

$$J = \frac{m\,(w^2 + \,l^2)}{12g} \qquad [21]$$

Where:

J is the moment of inertia at the load in oz-in sec^2

m is mass of the rotary load in ounces

w is width of load in inches

h is height of load in inches

g is gravity as 32.2 ft/s^2

The amplifier used on the proof-of-concept was an L298N Motor Driver Module (Full H Bridge Driver). Part of the data sheet is shown in Figure 40. Technically this motor driver is only used to drive the DC motor in both directions.

The gain of the L298N H Bridge Driver by itself is unity since it is a pure H Bridge but there are Field Effect Transistors in each leg of the system which are operating at 12 Volts therefore there is an amplifier gain of at least 12 Volt per tenth of a degree of motor rotation or G_A is 120 V / deg.

The resistance of the L298N H Bridge Driver depends upon two things. First, the resistance of any leg of the H Bridge is dependent upon he current running through the device. The Total Voltage Drop, V_{CEsat}, ranges from 3.2V to 4.9V at 1A and 2A respectively. Using Ohm's Law at 12 Volts operating voltage, the resistance of any leg of the circuit is 11.1 Ohms. Second, the operation of the H Bridge is to reverse the direction of the motor so the electrical circuit must always pass through two legs therefore the R_A=22.2 Ohms.

祥能精機股份有限公司

馬達特性測試結果 (Motor Test Report)

客戶名稱 (Customer):景行	馬達溫度 (Temperature):
資料檔名 (Filename):35G019090	無載轉速 (No-Load R.P.M):44
馬達型號 (Motor No.):HN345-35GAH-1926Y	無載電流 (No-Load Current):
額定電壓 (Voltage):DC12V	啟動轉矩 (Started Torque):
額定輸出 (Pout Rated):	測試日期 (Test Date):03-03-2016

	轉 速 R.P.M.	電 流 Amps	轉 矩 Kgcm	效 率 Eff	輸出功率 Wout	輸入功率 Win
無載狀態 (No-Load)	45	0.046	0.038	0.0317	0.0176	0.554
堵住狀態 (Locked Rotor)	0	0.845	10.000	0.0000	0.0000	10.170
最大扭力 (Max-Torque)	0	0.841	10.000	0.0000	0.0000	10.114
最大效率 (Max-Efficiency)	36	0.173	2.183	0.3884	0.8071	2.078
最大輸出功率 (Max-Pout)	24	0.386	5.391	0.2856	1.3287	4.652

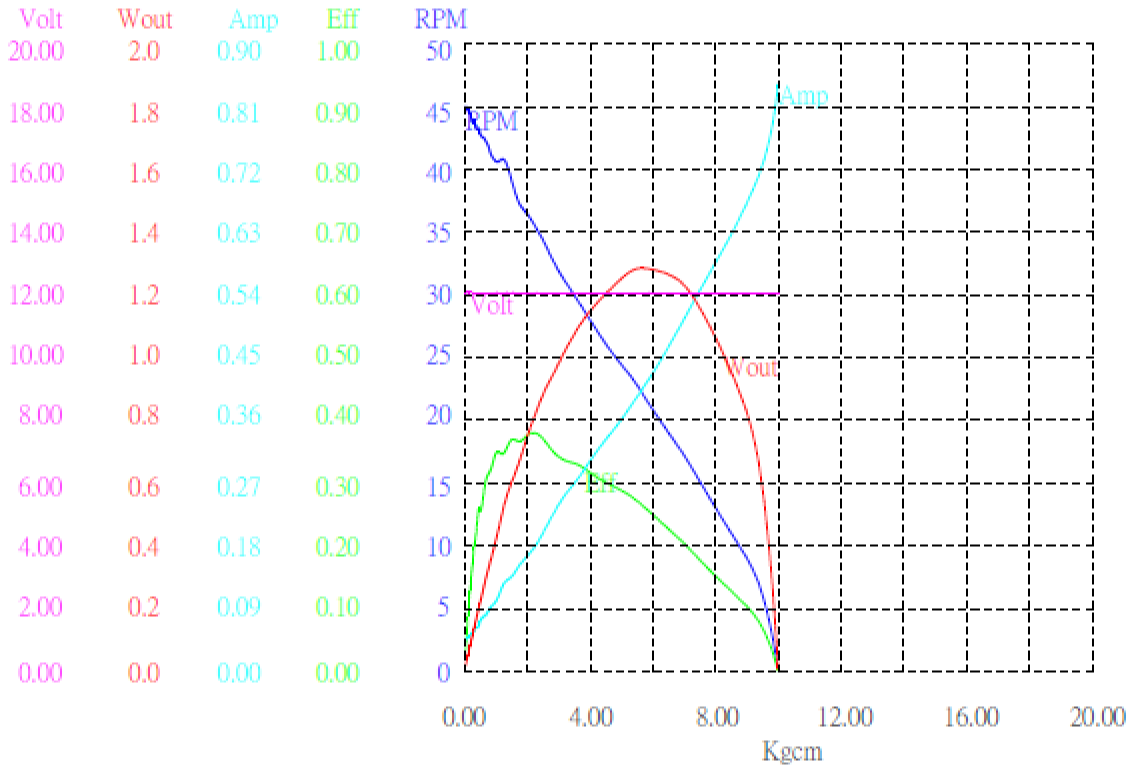
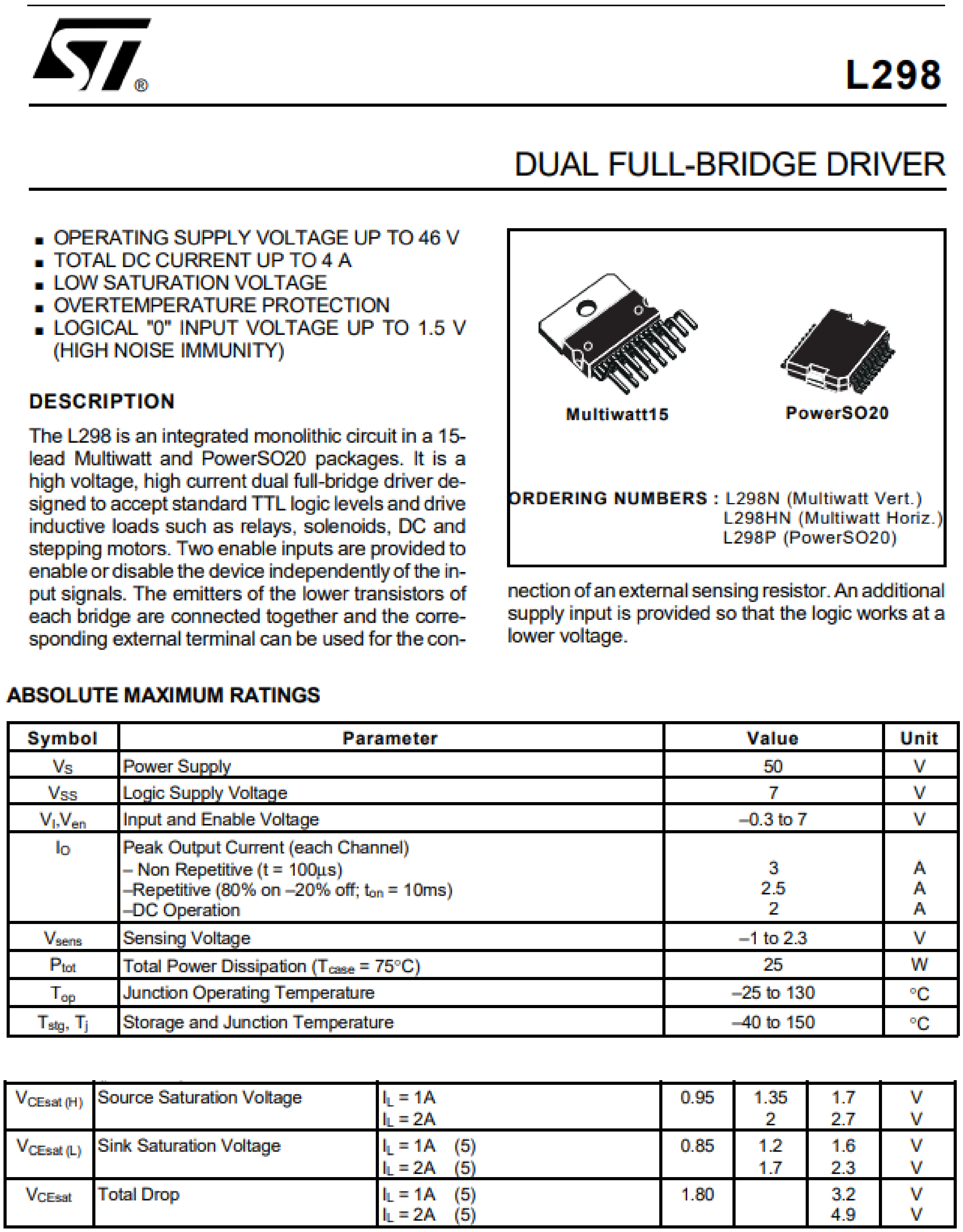


Figure 39 Motor Test Data



L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

Multiwatt15

PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100μs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

$V_{CEsat\ (H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A	0.95	1.35 2	1.7 2.7	V V
$V_{CEsat\ (L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)	1.80		3.2 4.9	V V

Figure 40 L298N Motor Driver Specification (Partial)

APPENDIX 8

HEADLAMP FIXED ANGLE OFFSET CALCULATIONS

The ideal angle of the headlamp is based on the height of the headlamp and the braking distance of the vehicle at the current linear speed. Figure 41 shows these two parameters as well as the fixed angle offset on flat level ground.

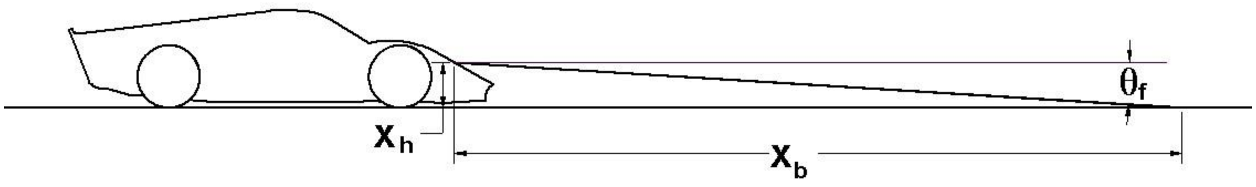


Figure 41. Ideal Headlamp Fixed Angle Definition

The laws of motion state the change in velocity is equal to the deceleration multiplied by time as shown in Equation 22.

$$\frac{5280}{3600}(V_c - V_f) = a_d g\, t_b \qquad [22]$$

Where:

V_c is the current vehicle speed in miles per hour

V_f is the final speed of the vehicle after braking in miles per hour

a_d is the unitless braking deceleration in terms of g's

g is gravity in feet per second squared

t_b is the time to brake in seconds

Knowing the final speed of the vehicle would be zero at full stop, Equation 22 can be solved for the time to brake as shown in Equation 23.

$$t_b = \frac{5280\, V_c}{3600\, a_d g} \qquad [23]$$

The laws of motion also state the braking distance can be calculated using the current speed multiplied by the braking time minus half the acceleration multiplied by the square of the braking time as shown in Equation 24.

$$x_b = \frac{5280}{3600} V_c \mathrm{t}_b - \frac{1}{2}\, a_d g\, {t_b}^2 \qquad [24]$$

Where:

x_b is the braking distance in feet

Substituting Equation 24 into Equation 24 and simplifying yields Equation 25:

$$x_b = 1.075556\, \frac{{V_c}^2}{a_d g} \qquad [25]$$

From Figure 41, the fixed angle offset is the arc tangent of the vertical height of the headlamp over the braking distance which is shown in Equation 26.

$$\theta_\mathrm{f} = \tan^{-1}\left(\frac{x_h}{12\, x_b}\right) \qquad [26]$$

The vehicle shown in Figure 42 is traveling at 45 miles per hour on flat level ground. Assuming the braking can occur at 0.7g, then Equation 25 yields a braking distance of 96.6 feet. The headlamp center is 40 inches vertically off the ground therefore from Equation 26, the headlamp optimum fixed angle is 1.98 degrees downward.

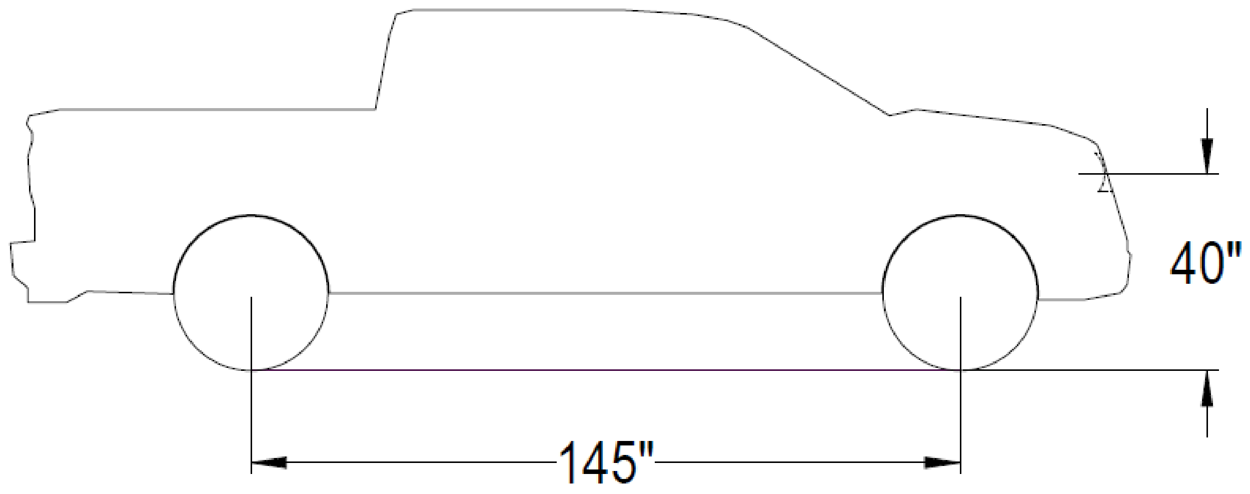


Figure 42 Headlamp Height

It should be noted that several factors affect this fixed headlamp angle. As the speed of the vehicle increases, you must see further in front of the vehicle so this angle decreases. The effective braking deceleration also affects this angle so worn tires or in slick roadways can reduce the deceleration factor thus increasing the braking distance. Additionally, if the vehicle loading front to back is changed then the front of the vehicle will angle upward or downward with respect to the rear of the vehicle then this ideal fixed angle is also affected.