MECH 450 SEC 150
INVERTED PENDULUM

Prepared for:

Dr. Amy Lehman
Department of Mechanical & Materials Engineering
University of Nebraska-Lincoln

Prepared by:

Jacob Gottberg, Sherevan Alhamy, Ricardo Jacome

Submitted 12/5/2018

1 Analytical Design Steps

System Layout

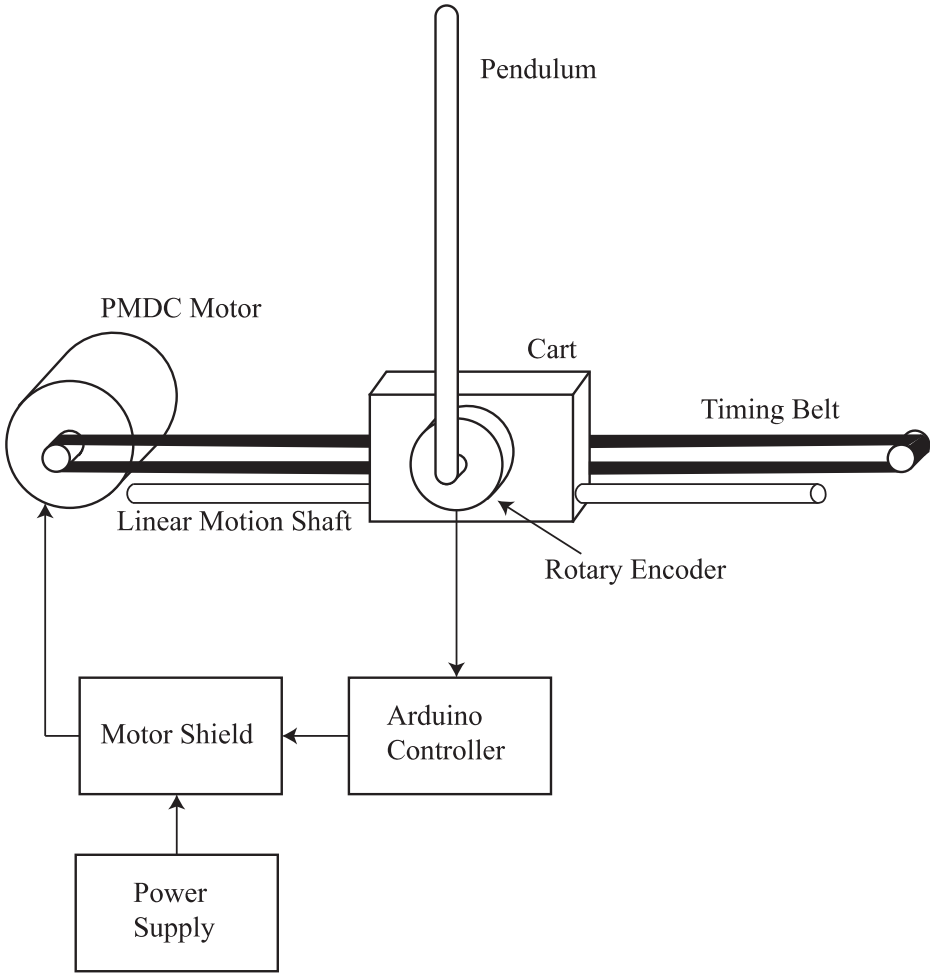


Figure 1: Detailed Layout

The detailed layout of the inverted pendulum controlled in this project was arranged shown (Fig. 1). As with many inverted pendulum control projects, it is constrained to one degree of freedom by limiting the translational motion of the cart to one axis (by the linear motion shaft) and limiting the rotational motion of the pendulum to one axis. It consisted of parts salvaged from a Hewlett-Packard inkjet printer (ca. 2000), coupled to a fabricated pendulum attachment assembly (described later in Sec. 2) and control electronics (described later in Sec. 3).

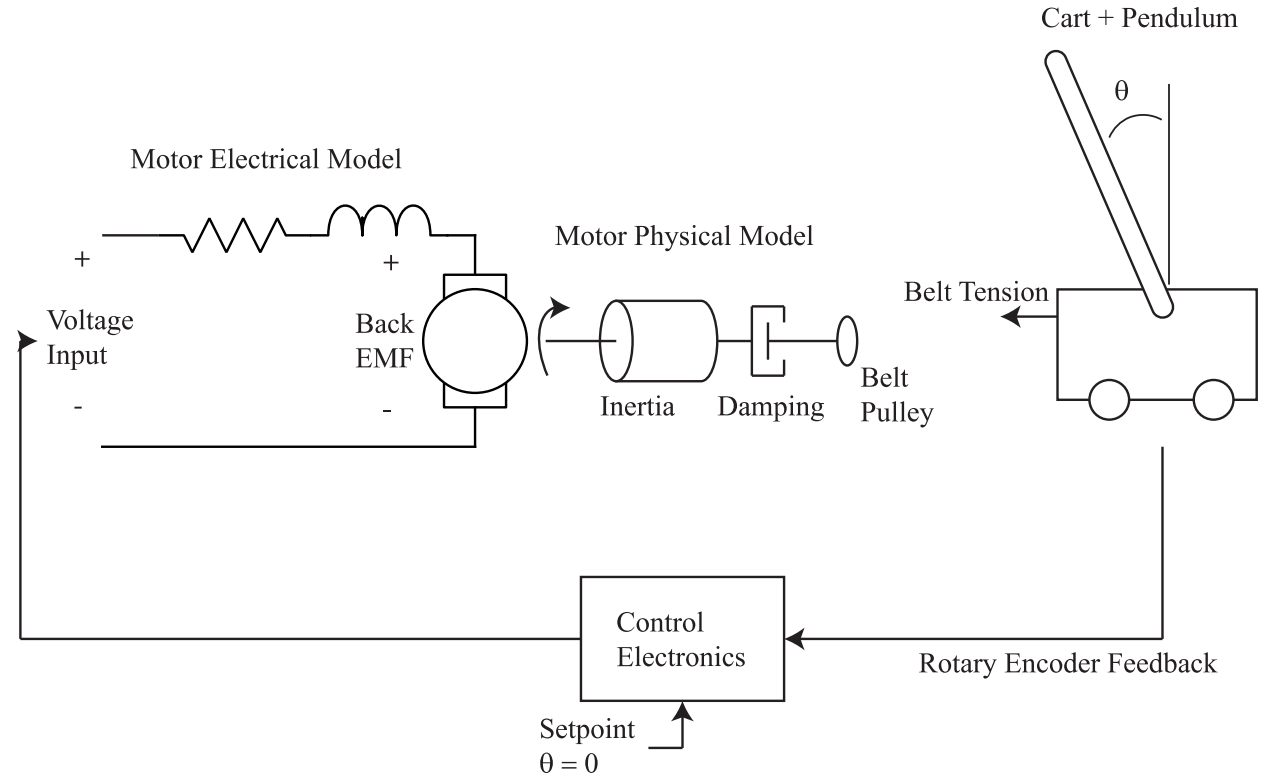


Figure 2: Schematic

The schematic of the dynamic control system was as shown in Fig. 2. The physical and electrical model of the motor will be discussed further in Sec. 1.3. The control electronics will be discussed in Sec. 3.

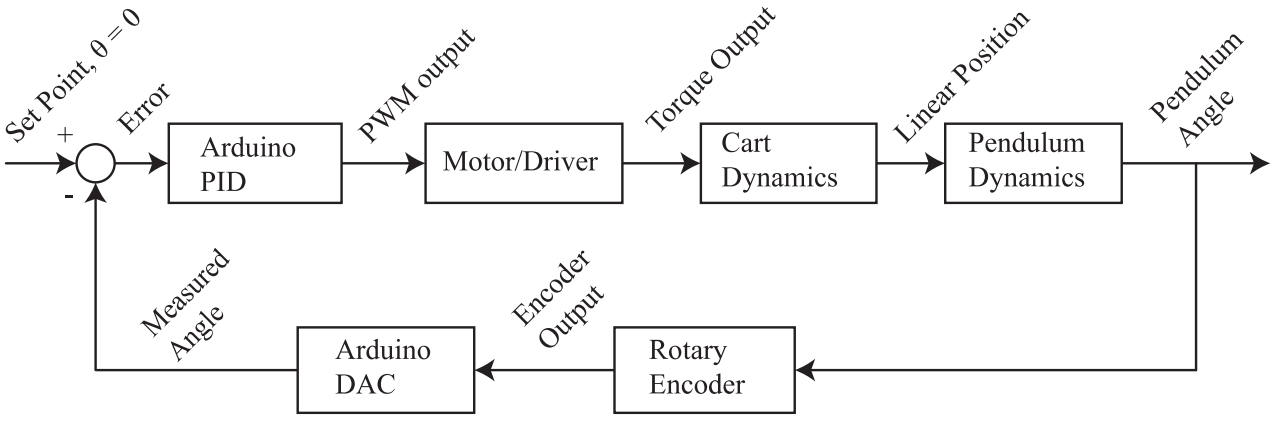


Figure 3: Functional Block Diagram

The functional block diagram of the dynamic control system is shown above (Fig. 3).

1.1 Modeling in the Frequency Domain

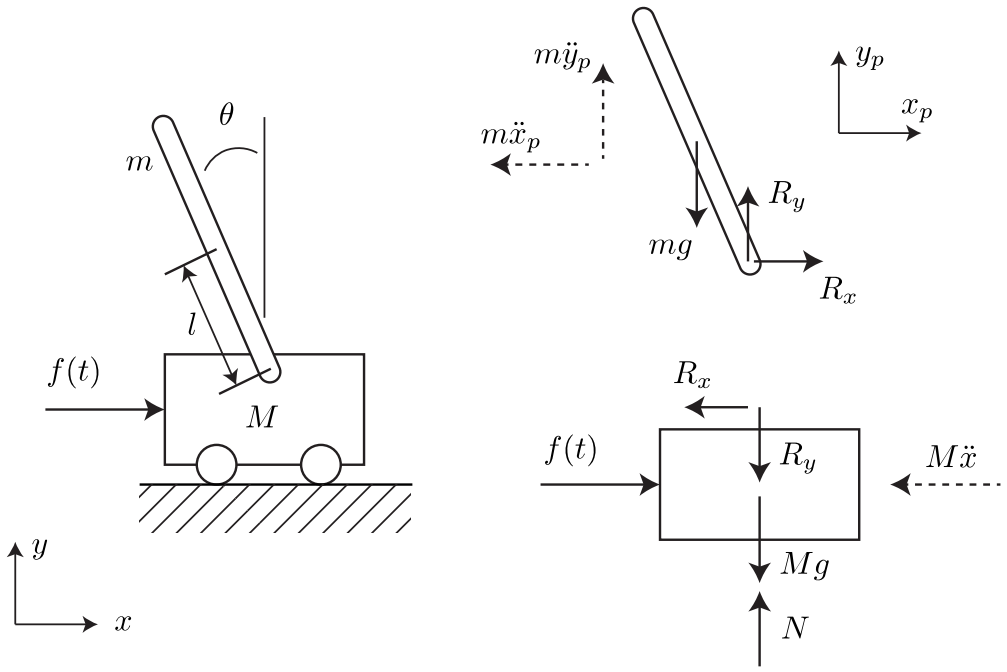


Figure 4: Translational Free Body Diagram

Pendulum position in x:

$$x_p = x - l\sin\theta$$

$$\dot{x}_p = \dot{x} - l\dot{\theta}\cos\theta$$

$$\ddot{x}_p = \ddot{x} + l\dot{\theta}^2\sin\theta - l\ddot{\theta}\cos\theta$$

Pendulum position in y:

$$y_p = -l\cos\theta$$

$$\dot{y}_p = l\dot{\theta}\sin\theta$$

$$\ddot{y}_p = l\dot{\theta}^2\cos\theta + l\ddot{\theta}\sin\theta$$

An important assumption for the system's math requires the correct assumption of relative translational motion directions of the cart and pendulum. This is necessary for getting the correct relative directions of inertia for the pendulum. Consider:

$$\dot{x}_p = \dot{x} - l\dot{\theta}\cos\theta$$

θ is assumed small and for reasonably sized l, $\dot{x}_p$ should share the sign of x, the position of the cart. Therefore, the pendulum will be **translating horizontally in the direction of the cart, and rotating counter-clockwise (i.e., falling or translating downward)** for the FBD shown.

First eq. of motion (ref. Fig. 4):

$$\sum_{cart} F_x: \qquad M\ddot{x} + R_x = f(t) \tag{1}$$

$$\sum_{pendulum} F_x: \qquad R_x = m\ddot{x}_p \tag{2}$$

Substitute $\ddot{x}_p$ in (2):

$$R_x = m(\ddot{x} + l\dot{\theta}^2 \sin\theta - l\ddot{\theta}\cos\theta)$$

Substitute R_x in (1):

$$(M+m)\ddot{x} + ml\dot{\theta}^2 \sin\theta - ml\ddot{\theta}\cos\theta = f(t)$$

Linearize equations by assuming small θ (i.e., $\sin\theta = \theta$ and $\cos\theta = 1$):

$$(M+m)\ddot{x} + ml\dot{\theta}^2\theta - ml\ddot{\theta} = f(t)$$

Higher orders of theta are ≈ 0

$$\boxed{(M+m)\ddot{x} - ml\ddot{\theta} = f(t)} \tag{i}$$

Second eq. of motion (ref. Fig. 4 and 5):

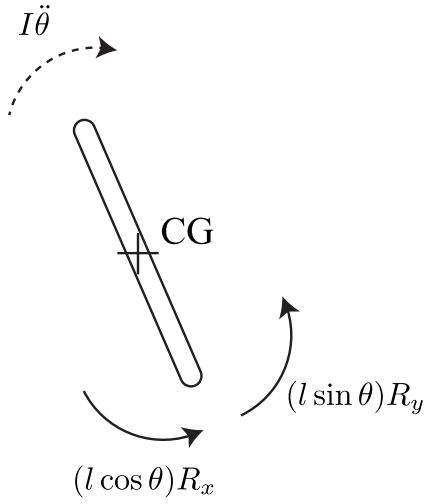


Figure 5: Rotational Free Body Diagram

$$\sum_{pendulum} M_{\text{CG}}: \qquad (l\cos\theta)R_x + (l\sin\theta)R_y = I\ddot{\theta} \tag{3}$$

$$\sum_{pendulum} F_y: \qquad R_y = -m\ddot{y}_p + mg \tag{4}$$

Substitute $\ddot{y}_p$ in (4):

$$R_y = m(-l\dot{\theta}^2\cos\theta - l\ddot{\theta}\sin\theta + g)$$

Sub. R_y in (3):

$$ml\cos\theta(\ddot{x} + l\dot{\theta}^2\sin\theta - l\ddot{\theta}\cos\theta) + ml\sin\theta(-l\dot{\theta}^2\cos\theta - l\ddot{\theta}\sin\theta + g) = I\ddot{\theta}$$

Linearize for small angle:

$$ml(\ddot{x} + l\dot{\theta}^2\theta - l\ddot{\theta}) + ml\theta(-l\dot{\theta}^2 - l\ddot{\theta}\theta + g) = I\ddot{\theta}$$

Canceling higher orders:

$$\boxed{ml\ddot{x} + mlg\theta = (I + ml^2)\ddot{\theta}} \tag{ii}$$

The mechanical system is modeled by the equations of motion (i) and (ii):

$$(M+m)\ddot{x} - ml\ddot{\theta} = f(t) \tag{i}$$

$$ml\ddot{x} + mlg\theta = (I+ml^2)\ddot{\theta} \tag{ii}$$

Applying Laplace transform, the model is represented in the frequency domain:

$$(M+m)s^2X - mls^2\theta = F(s)$$

$$mls^2X + mlg\theta = (I+ml^2)s^2\theta$$

This is represented in a block diagram:

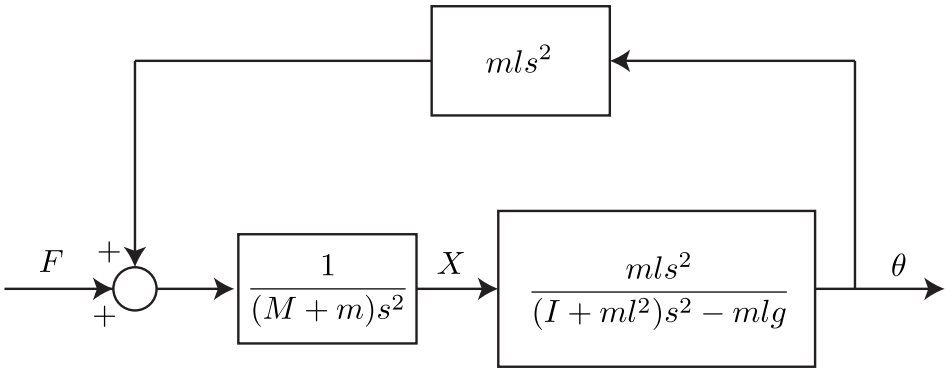


Figure 6: Mechanical Model Block Diagram

Which reduces to:

$$\boxed{G_{mech} = \frac{\theta}{F} = \frac{ml}{[(M+m)(I+ml^2) - m^2l^2]s^2 - (M+m)mlg}} \tag{iii}$$

1.2 Modeling in the Time Domain

The mechanical system is modeled in the time domain using state space representation (SSR). Using the state variables:

$$q_1 = x \qquad q_2 = \dot{x} \qquad q_3 = \theta \qquad q_4 = \dot{\theta}$$

The first state eq. proceeds simply:

$$\boxed{\dot{q}_1 = \dot{x}} \tag{s1}$$

The second state eq.:

$$\dot{q}_2 = \ddot{x}$$

From (i):

$$\ddot{x} = \frac{f + ml\ddot{\theta}}{M + m}$$

From (ii):

$$\ddot{\theta} = \frac{ml\ddot{x} + mlg\theta}{(I + ml^2)}$$

Sub. $\ddot{\theta}$ in $\ddot{x}$ and rearrange:

$$\boxed{\dot{q}_2 = \ddot{x} = \frac{(I + ml^2)}{(I + ml^2)(M + m) - m^2l^2} f + \frac{m^2l^2g}{(I + ml^2)(M + m) - m^2l^2}\theta} \tag{s2}$$

The third state eq. proceeds simply:

$$\boxed{\dot{q}_3 = \dot{\theta}} \tag{s3}$$

The fourth state eq.:

$$\dot{q}_4 = \ddot{\theta}$$

Recall from (ii):

$$\ddot{\theta} = \frac{ml\ddot{x} + mlg\theta}{(I + ml^2)}$$

Substitute (s2):

$$\boxed{\dot{q}_4 = \ddot{\theta} = \frac{ml}{(I + ml^2)(M + m) - m^2l^2} f + \left[\frac{m^3l^3g}{(I + ml^2)^2(M + m) - m^2l^2(I + ml^2)} + \frac{mlg}{(I + ml^2)}\right]\theta} \tag{s4}$$

Representing the state equations (s1, s2, s3, s4) in standard state space matrix form:

$$\boldsymbol{\dot{X} = AX + BU}$$

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{m^2l^2g}{(I+ml^2)(M+m)-m^2l^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{m^3l^3g}{(I+ml^2)^2(M+m)-m^2l^2(I+ml^2)} + \frac{mlg}{(I+ml^2)} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{(I+ml^2)}{(I+ml^2)(M+m)-m^2l^2} \\ 0 \\ \frac{ml}{(I+ml^2)(M+m)-m^2l^2} \end{bmatrix} f$$

$$\boldsymbol{Y = CX + DU}$$

$$\begin{bmatrix} x \\ \theta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} f$$

The advantage of modeling in the time domain is that multiple inputs/outputs can be observed, whereas the frequency domain is limited to single input single output (SISO) systems.

1.3 Motor Model

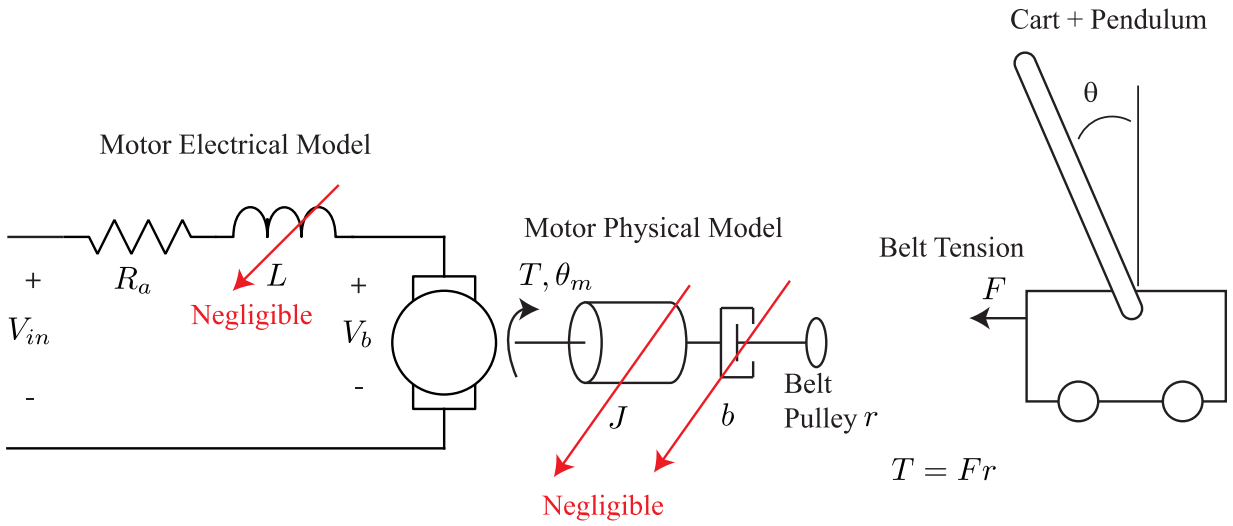


Figure 7: Motor Model With Simplification

Used in this project was a PMDC motor (data sheet found in appendix 1). The motor model is as shown (Fig. 7), where motor inductance, rotor inertia and damping are neglected for convenience. The input to the inverted pendulum cart system is a force to the cart, causing it to move left and right along the linear motion shaft to catch the pendulum as it falls. This force is generated by a torque from the motor and transmitted via a belt and pulley. The input to the mechanical system is this force. Thus, the output of the motor transfer function will be force.

To find the motor transfer function, apply Kirchoff's voltage law to the motor circuit:

$$v_{in} = iR_a + v_b \tag{m1}$$

Where the motor constants apply:

$$v_b = K_b\dot{\theta}_m$$

$$\tau = K_t i$$

The angular velocity of the motor has the following relation to cart position, x:

$$r\theta_m = x$$

$$r\dot{\theta}_m = \dot{x}$$

And motor torque, T, relates to belt tension, F, by:

$$\tau = fr$$

Simplifying (m1):

$$v_{in} = \frac{R_a r}{K_t} f + \frac{K_b}{r}\dot{x}$$

Rearrange and apply Laplace transform for a transfer function with output F:

$$\boxed{G_{motor} = \frac{F}{V_{in} - K_b sX/r} = \frac{K_t}{R_a r}} \tag{iv}$$

Overall System Block Diagram

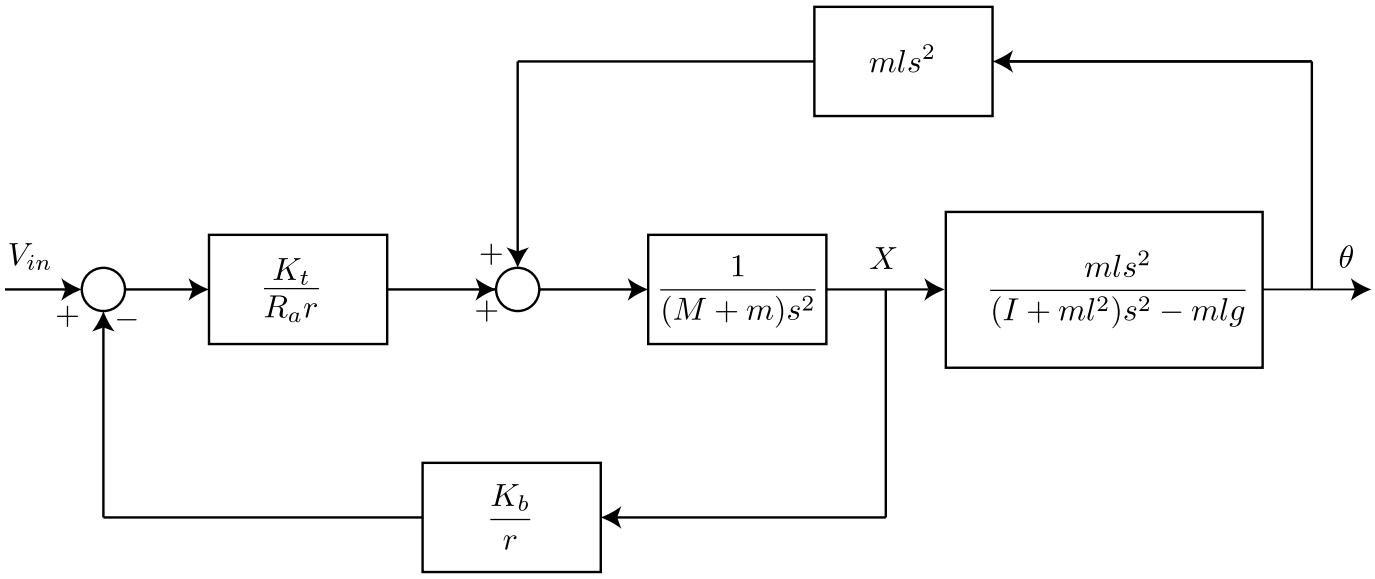


Figure 8: Overall System Block Diagram

The overall system block diagram is shown above (Fig. 8), coupling (iv) to (iii). **The input to the system is voltage, V_{in}, and the output is pendulum angle, θ.** This will be simplified using MATLAB's built in block diagram functionality. A numerical transfer function will be given, using the determined system parameters.

1.4 Determination of System Parameters

The system parameters used in this project were either estimated, measured or sourced from reference material. The table of system parameters used follows:

Table 1: System parameters

Parameter	Definition	Units	Value	Source
M	Mass of cart	kg	0.100	Estimated
m	Mass of pendulum	kg	0.076	Measured
l	Pendulum length	m	0.037	Measured
r	Radius of belt Pulley	m	0.00573	Measured
I	Rotational inertia of pendulum	kg m^2	8.67e-6	Measured
J_m	Rotational inertia of rotor	kg m^2	-	Neglected
J_p	Rotational inertia of belt pulley	kg m^2	-	Neglected
K_t	Motor torque constant	N m A^{-1}	0.05	Motor Data Sheet
K_b	Motor back EMF constant	V s rad^{-1}	0.05	Motor Data Sheet
R_a	Armature resistance	Ω	6.5	Motor Data Sheet
B_{db}	Drive belt damping	kg m^2 rad^{-1} s^{-1}	-	Neglected

Mass parameters were measured using a digital laboratory balance. The mass of the cart was estimated as it was difficult to remove from the linear motion rail. Length parameters were measured using a ruler and a digital caliper. Motor constants were taken from performance curves found in the motor's data sheet (appendix 1). Inertia was calculated for a rod about its center, $I = \frac{1}{12}ml^2$. Indicated parameters were neglected to simplify the system. This results in the numerical form of the open-loop transfer function, found using MATLAB's block diagram functionality:

$$\boxed{\text{OLTF} = \frac{2.265 \cdot 10^{-9}\ s}{7.16 \cdot 10^{-12}\ s^3 + 7.924 \cdot 10^{-10}\ s^2 - 2.914 \cdot 10^{-9}\ s - 1.1939 \cdot 10^{-7}}} \quad \text{(v)}$$

1.5 MATLAB Time Response Simulation

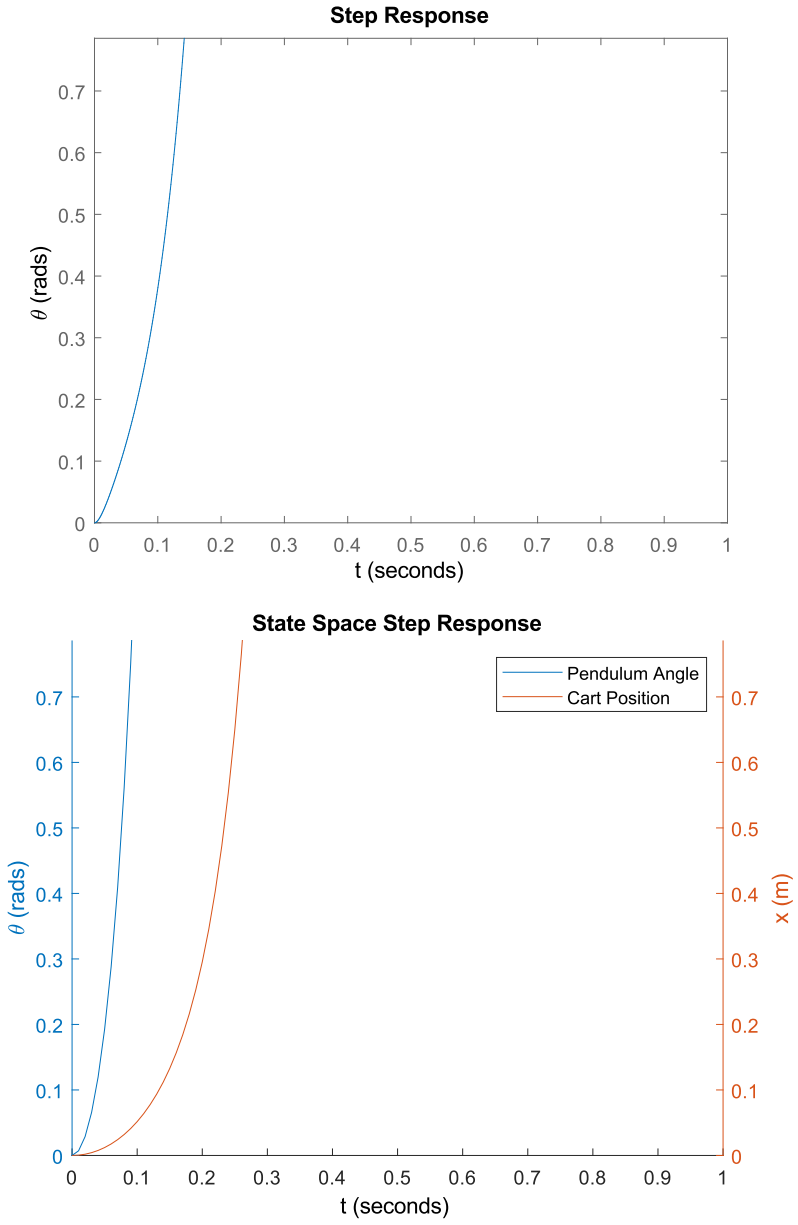


Figure 9: Time Response Simulation
(top) frequency domain, (bottom) state space

Shown is the time response characteristics of the uncompensated system (Fig. 9). The step response in state space is expected to be the same as the frequency domain. The slight discrepancy is likely due to rounding. For a step input, the pendulum simply falls over. Note that as the pendulum angle increases the validity of these response curves diminishes, since the small angle assumption was used extensively in crafting the system model. For instance, the cart position should appear more linear for a step response... but it does not in this simulation due to the small angle assumption being violated at large θ. MATLAB code used in this simulation is attached as an appendix (appendix 2).

1.6 MATLAB Root Locus and Compensation

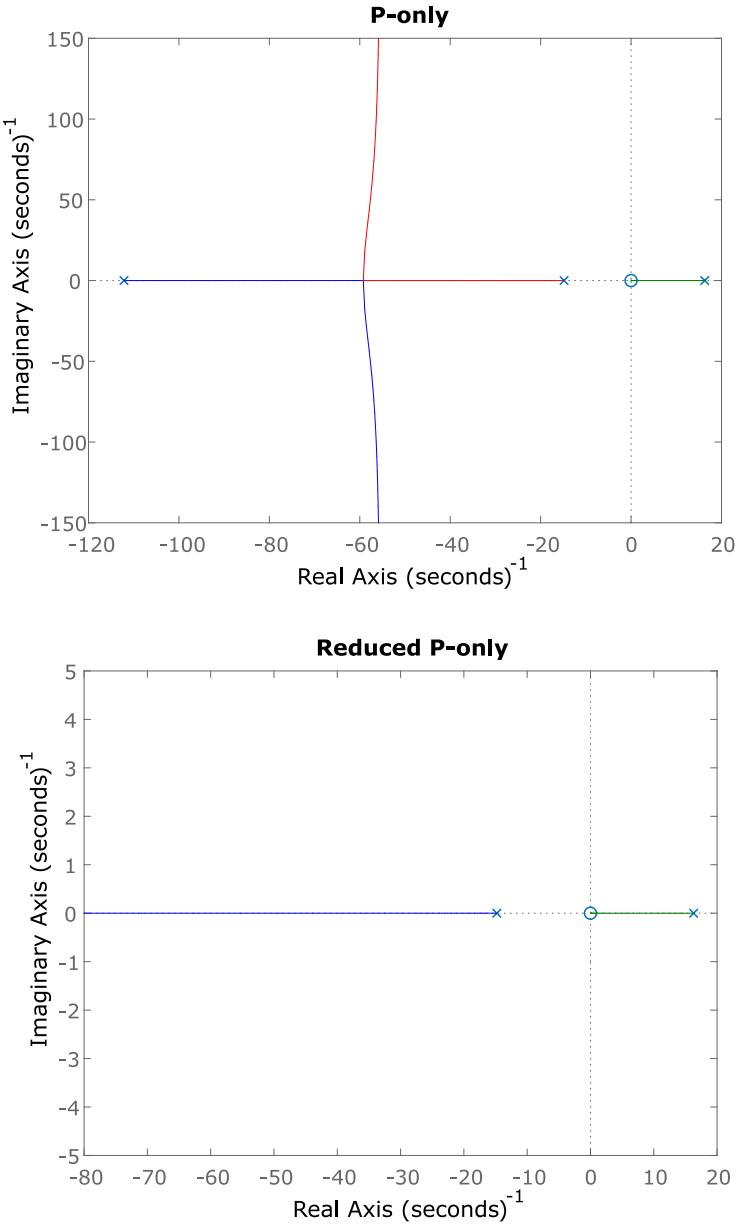


Figure 10: Uncompensated Root Locus
(top) root locus of the uncompensated system, (bottom) reduced root locus with leftmost pole removed

Considering the root locus only including proportional gain (no compensation), the system appears to be unstable. The branch in the right half plane (RHP) never enters the stable, left half plane (LHP). The system will approach marginal stability at infinite gain, but this is not a useful outcome. It does not appear that the system can be controlled with proportional only gain.

To simplify the root locus and further iterations, the furthermost pole in the LHP will be neglected (Fig. 10 bottom), since the poles closer to the origin will be dominant.

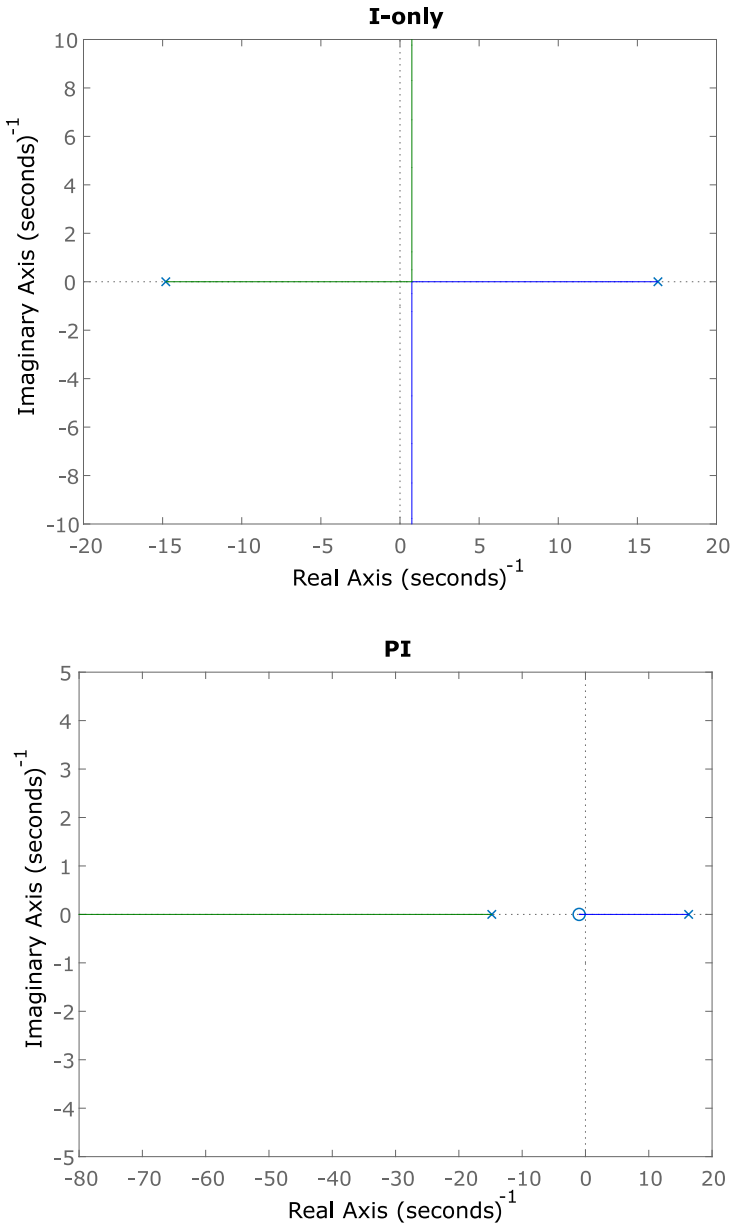


Figure 11: Compensated Root Locus
(top) root locus including integral only gain, (bottom) root locus with proportional and integral gain

Considering the integral-only root locus (Fig. 11 top), the system is unstable for all values of gain. Adding integral only gain cancels the zero at the origin, but cannot pull the root locus into the RHP.

Considering the proportional and integral (PI) gain root locus (Fig. 11 bottom), the zero at the origin is canceled, and a new zero is introduced in the RHP. This effectively pulls the rightmost branch into the LHP. Thus, for PI compensation, the system can be stabilized for higher values of gain. Although it can be stabilized with PI compensation, all the root loci lie on the real axis. This means that the system will always assume an overdamped response, which is not desirable. A proportional integral and derivative (PID) gain controller will be developed to achieve a faster response.

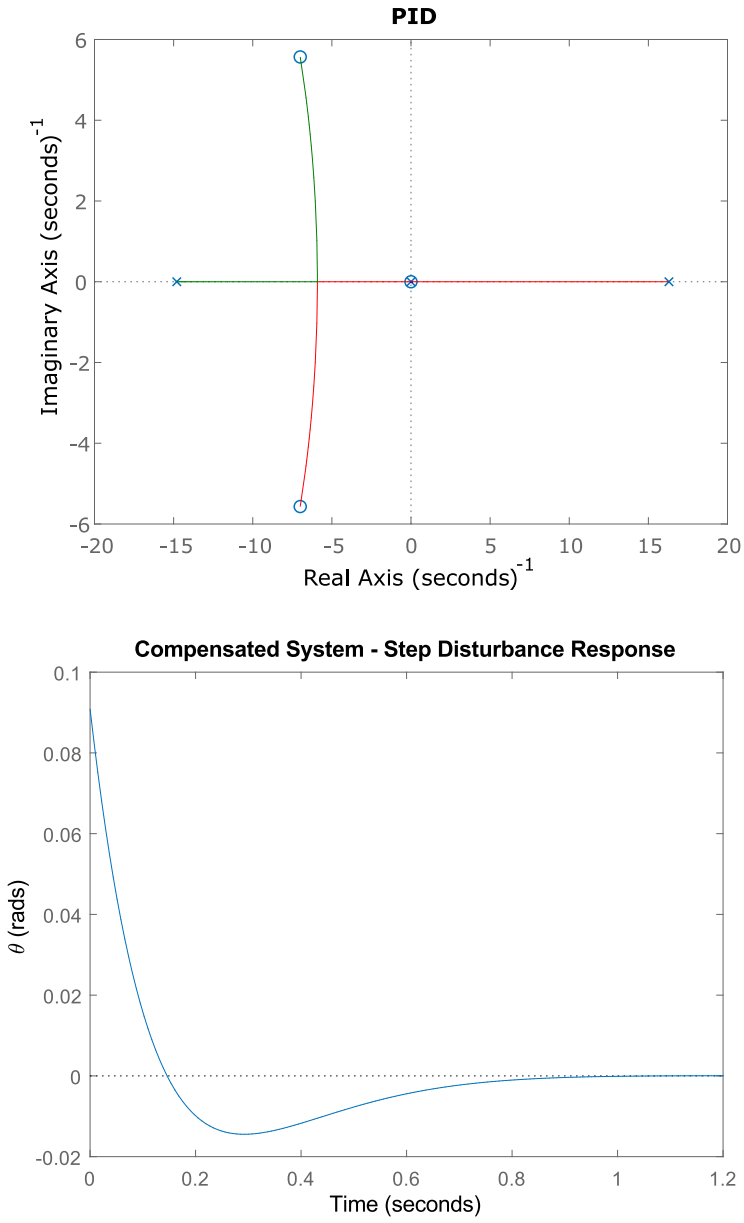


Figure 12: PID Compensation and Response
(top) root locus including PID gain, (bottom) step disturbance response

With a small amount of iteration, zeroes introduced in the PID root locus were able to be positioned such that the damping of the system was decreased. This was achieved by placing zeroes off the real axis (introducing imaginary component). The system under PID compensation is stable for a wider range of gains. Particular gains chosen in this simulation were:

$$K_p = 140 \qquad K_i = 800 \qquad K_d = 10$$

The response appears to be desirable, with the pendulum recovering in nearly half a second (Fig. 12 bottom). This response is stable, and with desirable time response characteristics. These gains will be considered a starting point for the implementation of the controller, but the actual values of gain that will stabilize the system in practice will depend heavily upon the limitations of the actual physical system.

2 Physical Design

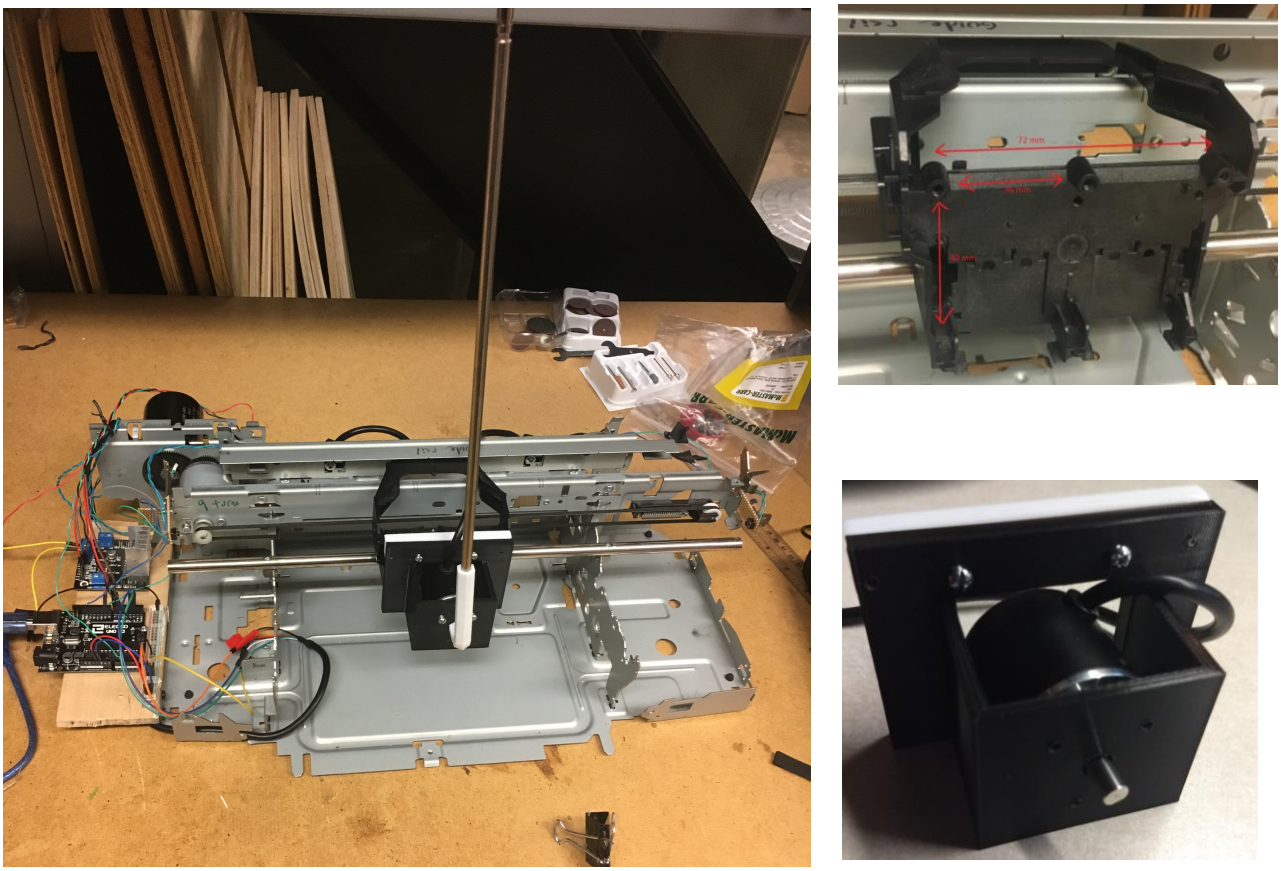


Figure 13: Hardware Configuration
(left) final physical design, (top right) ink carriage, (bottom right) 3D printed attachment

The phyiscal design is as shown in Fig. 13. The original linear motion shaft and ink carriage were salvaged from the printer, and a 3D printed part was designed to attach the pendulum and encoder to the carriage. An additional 3D printed part was used to attach the pendulum to the encoder shaft using interference fits. Care was taken in creating this physical system so that the overall weight (176g) was similar to the original weight of the carriage and ink cartridge assembly (200g). This ensured that the motor salvaged from the printer would be able to move the assembly easily. The parts used to build this system are tabulated below:

Table 2: Parts list

Part	Description	Source
Rotary Encoder	600 pulse/rev	Amazon
3D Printed Cart	PLA filament	Schematic Attached
Printer	HP Deskjet 5150	Provided
Hardware	M2 bolts	Salvaged from printer
PMDC Motor	Mitsumi M36N-2 Series	Salvaged from printer
Arduino	Uno R3 Clone	-
Motor Shield	L298N H-bridge	-

3 Implementation

Control Electronics

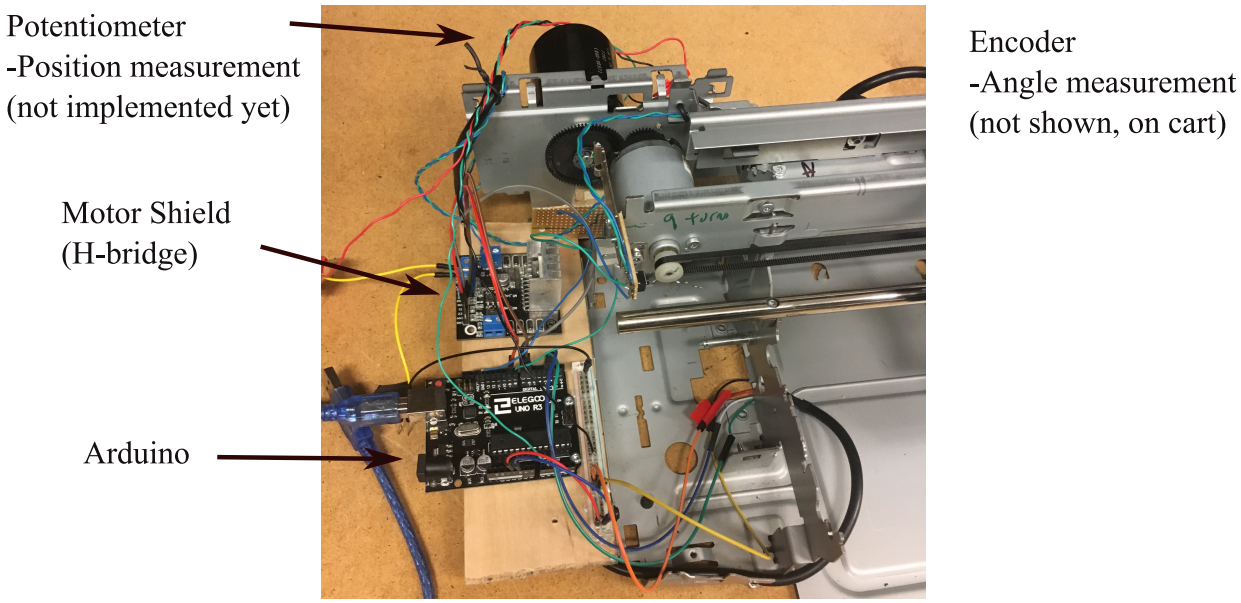


Figure 14: Hardware Configuration

Shown above is the control electronics used for the project. This included an Arduino, a H-bridge motor shield and a rotary encoder. A potentiometer was attached to the motor for position measurement. The hope was to implement a positional control in addition to angle, but time did not allow this. By using the Arduino, control software was able to be easily implemented into the project (Arduino code attached).

Software

Software used to control the pendulum utilized an existing PID library (source in attached code). Arduino code had to be written to couple the PID library to the physical design. This included a "calibration" routine, where the pendulum would begin in the down position (gravity equilibrium), and would be flipped vertical. The control program would start upon reaching vertical. The vertical position was the set point for the PID control.

Several values of gain were tested, and it appears the actual physical models had great limitations in comparison to the simulation. Derivative and integral gains much above 1 would cause rattling. This is likely due to backlash in the physical system not accounted for in simulation. The final PID values used to stabilize the system were:

$$\boxed{K_p = 20 \qquad K_i = 0.3 \qquad K_d = 0.1}$$

Importance of Encoder Interrupts

One of the difficulties faced in the control software was coupling the rotary encoder to the Arduino. The rotary encoder outputs angular position via two square wave signals. Their relative phase is what gives direction of the changing position. Due to the digital nature of the rotary encoder, it is vital to ensure that none of the square wave pulses are missed by the software. This would cause the zero position to drift, making it impossible to balance the pendulum. This was solved by using an interrupt routine, which interrupts the Arduino code in order to run the rotary encoder code, ensuring that none of the square wave pulses are missed. The encoder code in this project was written based on an example found online (source in attached code).

4 Conclusions

In closing, this project provided valuable hands-on experience building a simple control system. An inverted pendulum was successfully balanced using PID control. This project was an exercise in analysis, design and fabrication of a control system. Valuable experience in basic electronics was also gained through this project.

In the analysis, the inverted pendulum was successfully stabilized in simulation, but the gain values did not match the values used in practice. This is likely due to discrepancy between the model and the actual system. Several parameters were neglected in modeling for convenience. Backlash present in the actual system was not accounted for in the modeling.

Proposed Extensions

While the pendulum was balanced, it would tend to drift either to the left or the right. A proposed extension to this project would be to implement a method to center the pendulum. This was explored during the project, but time did not allow any effort come to fruition. An attempt was made to achieve this using a nested PID loop, which would be an interesting concept to explore further.

Additionally, since the simulated gain values were much different than the gain values used in practice, it would be interesting to explore further why this is the case. This would probably include experimentation to find some of the neglected parameters, like drive belt damping.

APPENDIX 1:
DC Motor Data Sheet

DC Mini-Motors
M36N-2 Series

Applications

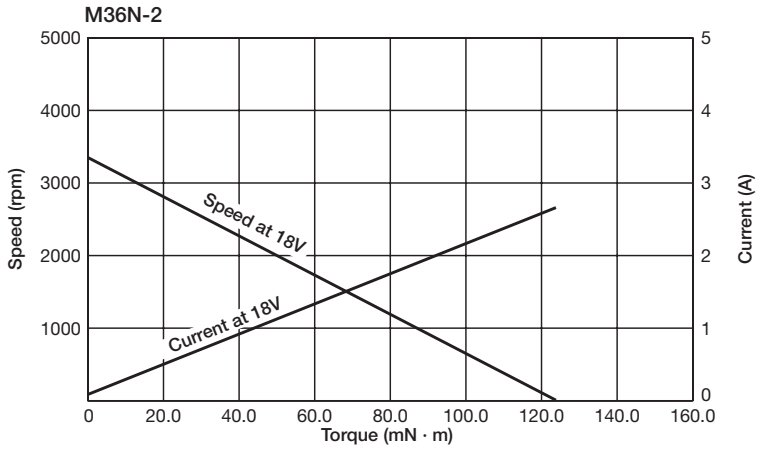
1. Printer
2. Power assistance

SPECIFICATIONS

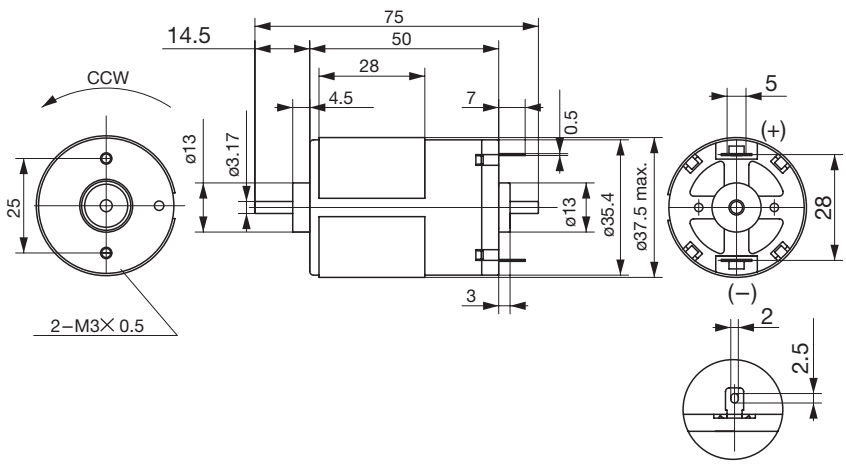
Items \ SPEC-NO	R14 7343
Rated Voltage (V)	18
Voltage Range (V)	6~18
Rated Load (mN·m)	15
No Load Speed (rpm)	3400
No Load Current (mA)	82.5
Starting Torque (mN·m)	122
Rotation	CW/CCW

*Encoders available as options.

CHARACTERISTICS



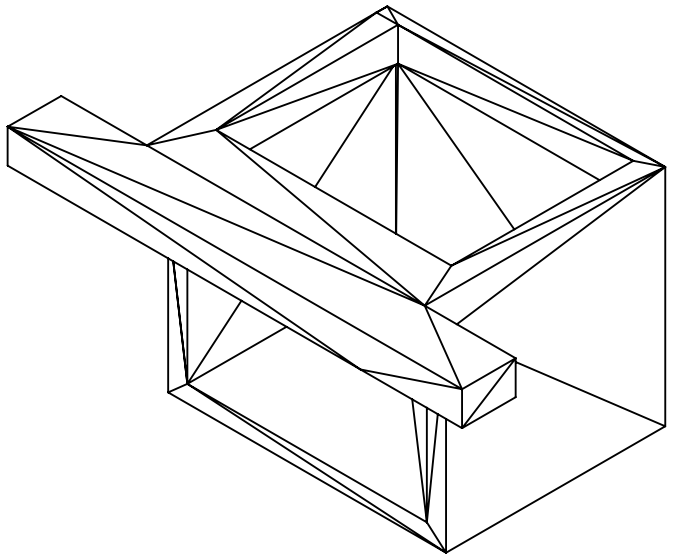
DIMENSIONS



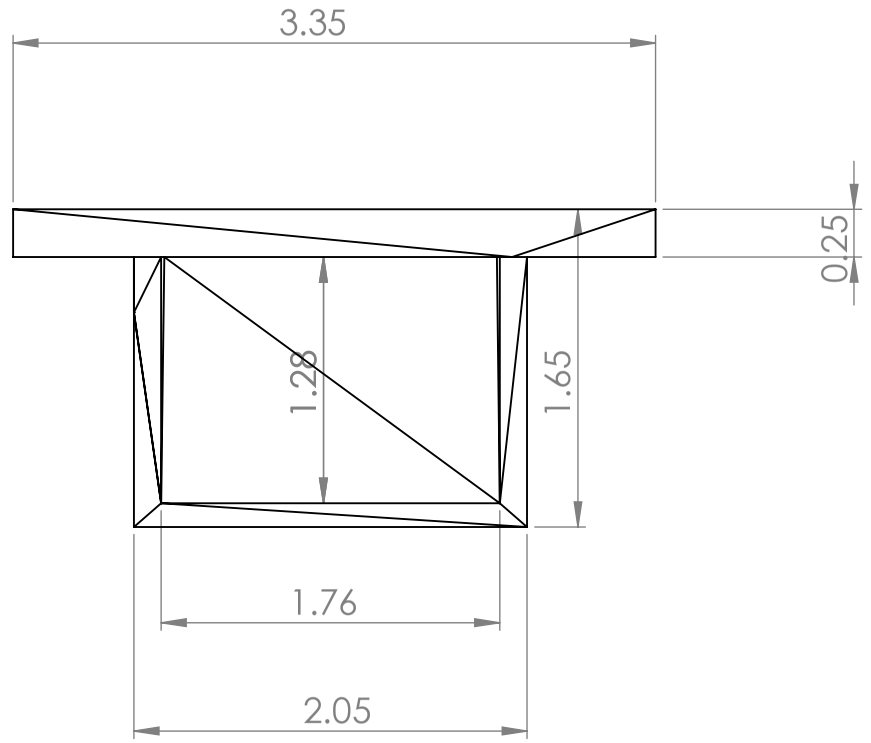
S=1/2

Unit : mm

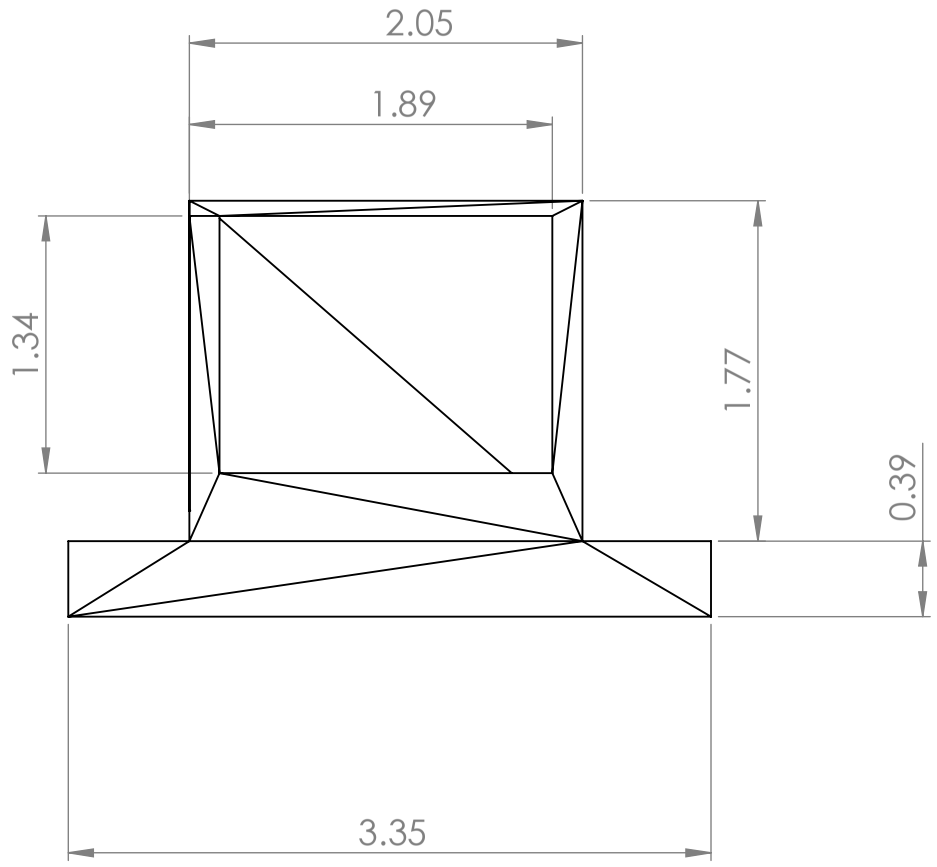
APPENDIX 2:
3D Printed Parts



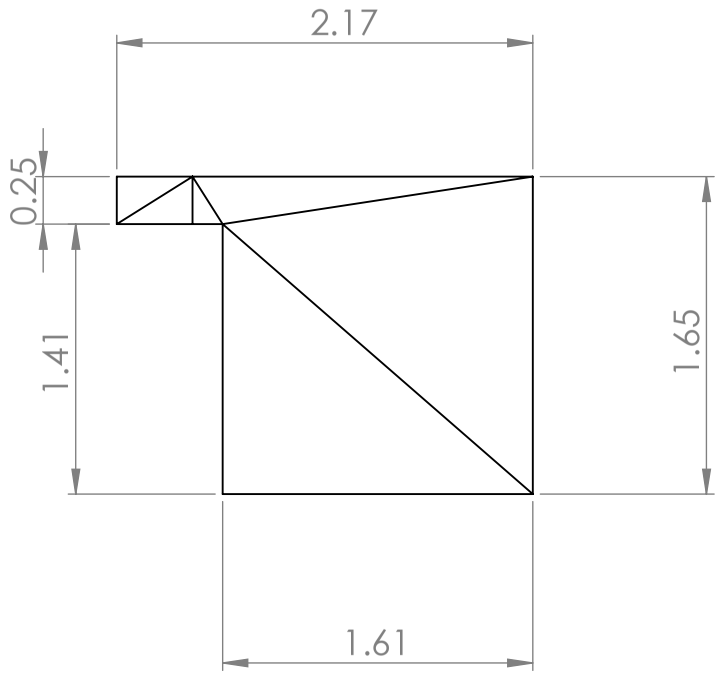
Isometric



Front View

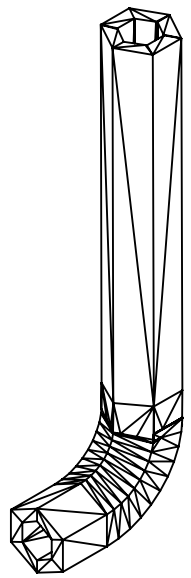


Top View

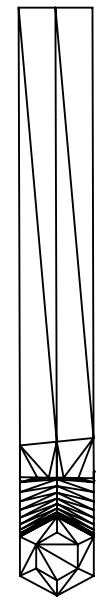


Side View

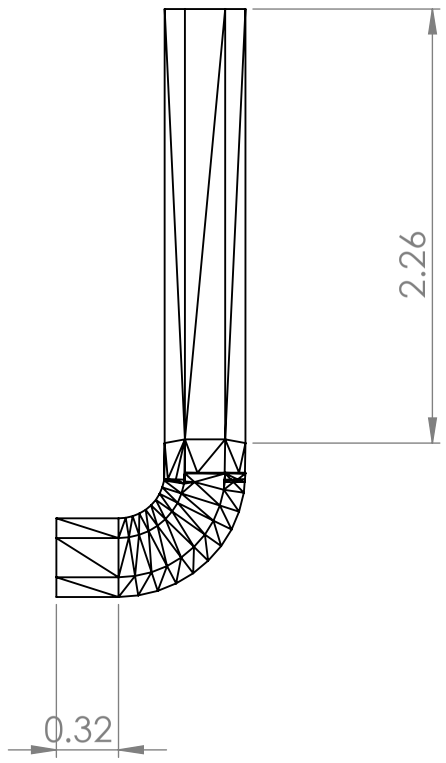
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE			TITLE:	
DRAWN							
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:		DWG NO. Cart	A3
				WEIGHT:		SCALE:1:1	SHEET 1 OF 1



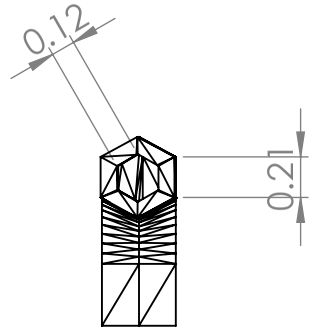
Isometric



Front View



Side View



Top View

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE			TITLE:	
DRAWN							
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:		DWG NO. Rod	A3
				WEIGHT:		SCALE:1:1	SHEET 1 OF 1

APPENDIX 3:
MATLAB Simulation Code

```
function [] = pen()
% MECH 450 SEC 150
% MATLAB simulation of inverted pendulum
% By: Jacob Gottberg, Sherevan Alhamy, Ricardo Jacome

close all;

syms s;

M = .1;
m = .076;
l = .037;
g = 9.81;
I = 8.67E-6;
Kt = .05;
Kb = .05;
Ra = 6.5;
r = .00573;

%blocks based on block diagram in report (fig. 8)
cartOne = tf( 1, [(M+m) 0 0] );

cartTwo = tf( [m*l 0 0 ], [1] );

pendulum = tf( [m*l 0 0], [(I + m*l^2) 0 -m*l*g] );

motorOne = tf( [Kt], [Ra*r] );

motorTwo = tf( [Kb 0], [r] );

G = feedback( cartOne*pendulum, cartTwo, +1 );


%TF is uncompensated, open loop transfer function
TF = feedback( motorOne*G, motorTwo/pendulum );

%freq. domain step
figure;
step(TF,1);
ylim([ 0 pi/4 ]);
xlim([0 1]);
xlabel('t');
ylabel('\theta (rads)');


%STATE SPACE:
A=[ 0 1 0 0 ;...
    0 0 m^2*l^2*g/((I+m*l^2)*(M+m)-m^2*l^2) 0 ;...
    0 0 0 1 ; 0 0 m^3*l^3*g/((I+m*l^2)^2*(M+m)-m^2*l^2*(I +
m*l^2))+m*l*g/(I+m*l^2) 0 ];
B=[ 0 ;...
    (I+m*l^2)/((I+m*l^2)*(M+m)-m^2*l^2) ;...
    0 ;...
    m*l/((I+m*l^2)*(M+m))-m^2*l^2 ];
```

```
C=[ 1 0 0 0 ; 0 0 1 0 ];
D=[  0 ;   0 ];


%time domain step
figure;
hold on;
t=0:0.01:1;
U = ones(size(t));
[Y,X]=lsim(A,B,C,D,U,t);
yyaxis left
plot(t,Y(: , 2));
ylim([ 0 pi/4 ]);
xlim([0 1]);
xlabel('t (seconds)');
ylabel('\theta (rads)');
yyaxis right
plot(t,Y(: , 1));
ylim([ 0 pi/4 ]);
ylabel('x (m)');
legend('Pendulum Angle', 'Cart Position');
title('State Space Step Response');
hold off;


% ROOT LOCUS:
%uncompensated
figure;
rlocus(TF);
title('P-only')

%reduce by removing far left pole
TF = zpk([0],[-14.8 16.3], 1);
figure;
rlocus(TF);
title('Reduced P-only');


%integral only
I = tf(1, [1 0]);
figure;
rlocus(tf(I*TF));
title('I-only');

%propotional-integral
PI = tf([1 1], [1 0]);
figure;
rlocus(tf(PI*TF));
title('PI')

%propotional-integral-derivative
PID = tf([10 140 800], [1 0]);
figure;
rlocus(PID*TF);
```

```
title('PID')

%step disturbance.
%NOTE controller in feedback since "setpoint" is 0, i.e., disturbance
%occurs after controller.
compTF = feedback(TF,PID);
figure;
impulse(compTF);
title('Compensated System - Step Disturbance Response');
ylabel('\theta (rads)');


end
```

APPENDIX 4:
Arduino Code

```
/*******************************************************
   PID Inverted Pendulum
   Jacob Gottberg, Sherevan Alhamy, Ricardo Jacome
 *******************************************************/

// PID Library source:  http://brettbeauregard.com/blog/
#include <PID_v1.h>

//encoder
#define PINA 2
#define PINB 3

//motorshield
#define INA 7
#define INB 6
#define EN 9

//stop button
#define STOP 8


boolean go = 0;
boolean calState = 1;
volatile boolean fired;
volatile boolean up;
volatile boolean stopState;

// Sets theta = 0 if started in down position.
double theta = 1200;

double angSetpoint = 0, posSetpoint, vout, angout;

// Tuning parameters
double angP = 25, angI = 0.3, angD = .1;


PID angPID(&theta, &vout, &angSetpoint, angP, angI, angD, DIRECT);


// Two nterrupt routines for encoder. One for each interrupt pin (2&3).
// This encoder code is written following the example by Nick Gammon
// https://forum.arduino.cc/index.php?topic=62026.0
void isr ()
{
  if (digitalRead (PINA))
    up = digitalRead (PINB);
  else
    up = !digitalRead (PINB);
  fired = true;

}

void isrTwo ()
{
```

```
 if (digitalRead (PINB))
   up = !digitalRead (PINA);
 else
   up = digitalRead (PINA);

 fired = true;
}


void setup()
{
  pinMode(STOP, INPUT_PULLUP);
  pinMode(EN, OUTPUT);
  pinMode(INA, OUTPUT);
  pinMode(INB, OUTPUT);
  pinMode (PINA, INPUT_PULLUP);     // enable pull-ups
  pinMode (PINB, INPUT_PULLUP);

  attachInterrupt (0, isr, CHANGE);   // interrupt 0 is pin 2, interrupt
1 is pin 3
  attachInterrupt (1, isrTwo, CHANGE);

  //turn the PID on
  angPID.SetMode(AUTOMATIC);
  angPID.SetOutputLimits(-255, 255);
  angPID.SetSampleTime(1.5);

  // Troubleshooting.
  //Serial.begin(9600);

}

void loop()
{
  // Button stop code. If the cart hits the bumper, it shuts off to
prevent stall.
  stopState = digitalRead(STOP);
  if ( stopState == LOW )
  {
    go = 0;
    vout = 0;
    analogWrite( EN, vout);
  }

  // Calibration code. Start in down position (gravity equilibrium),
rotate to vertical.
  if ( calState )
  {
    if ( theta == 0 )
    {
      calState = 0;
      go = 1;
    }
  }
```

```
  // Encoder code---takes result from interrupts and records theta.
  if ( fired )
  {

    if ( up )
      theta++;
    else
      theta--;
    fired = false;

  }


  // Cart code.
  if (go)
  {


    if (vout > 0 )
    {
      digitalWrite(INA, LOW);
      digitalWrite(INB, HIGH);
    }
    if (vout < 0 )
    {
      digitalWrite(INA, HIGH);
      digitalWrite(INB, LOW);
    }
    if (vout == 0 )
    {
      digitalWrite(INA, LOW);
      digitalWrite(INB, LOW);
    }

    angPID.Compute();

   // Troubleshooting
   //Serial.println(vout);

    analogWrite(EN, abs(vout));
  }

}
```