Closed-Loop Position Control of a Pendulum System

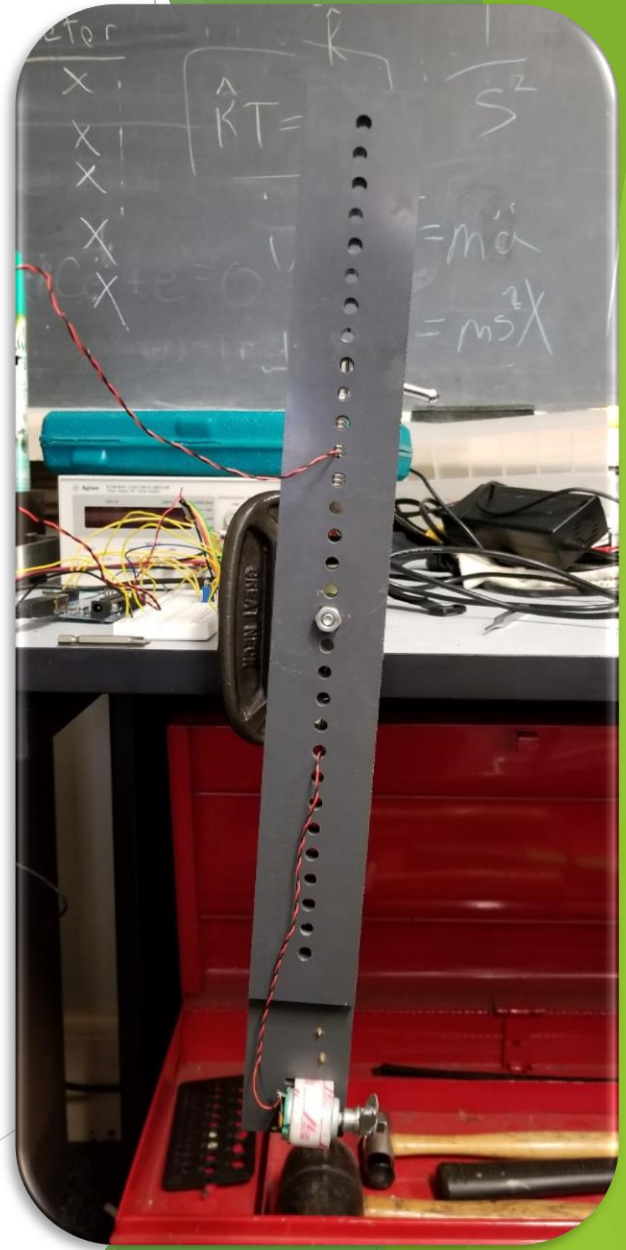
Research Project: June Kwon

Professor: Bor-Chin Chang

Outline

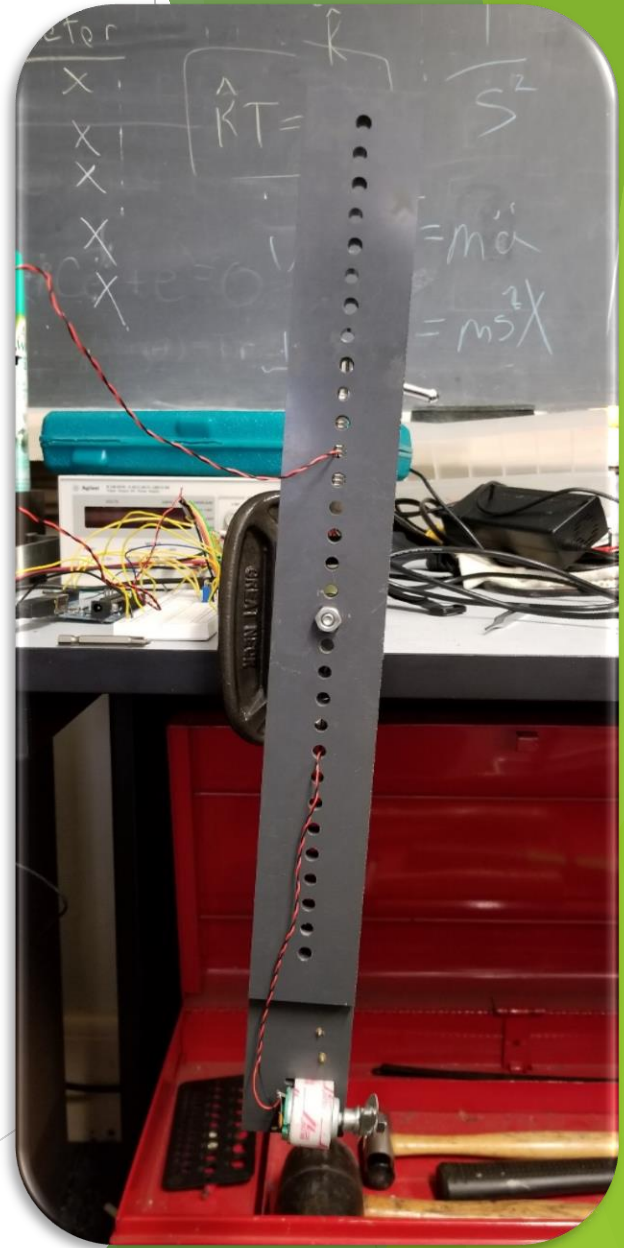
Background

- This pendulum control system was designed in the past, and was used as one of the Lab projects for MEM351: Dynamics Systems Laboratory I.



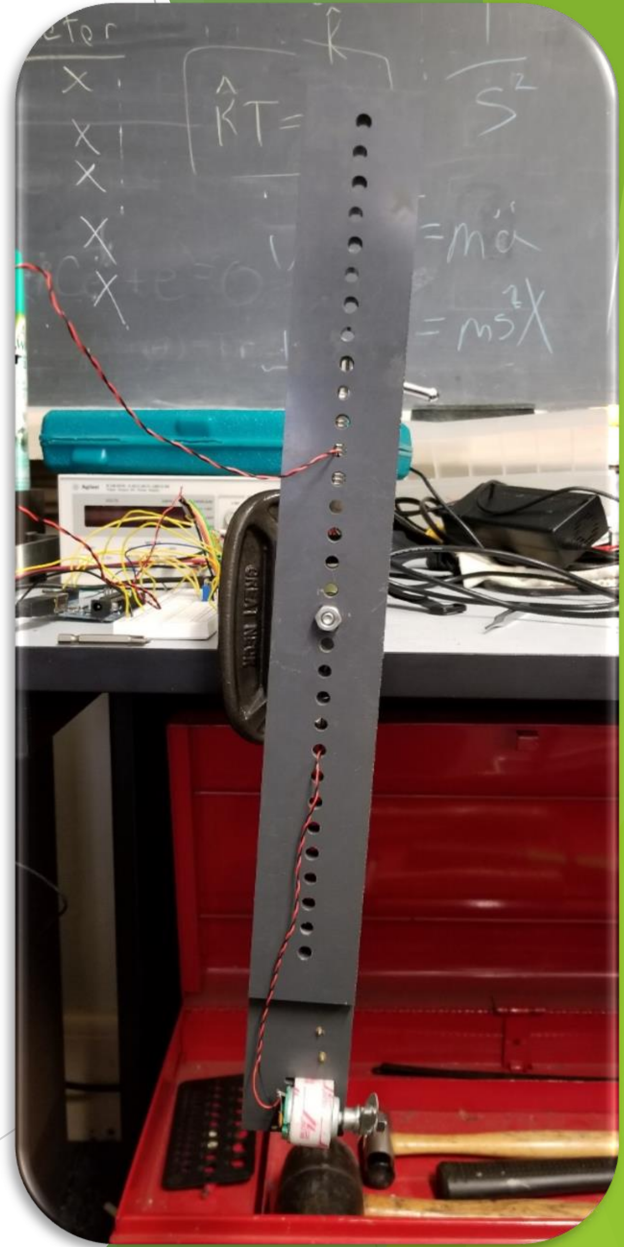
Background

- ***So what's the difference now?***
- ***Back then, the lab used a PID controller with LabVIEW.***
- ***Problem was that the equipment for the project was overly expensive and heavy.***



Background

- Moreover, designing the PID controller using LabVIEW was a tough task for students.
- Thus, the project was closed 2 years ago.



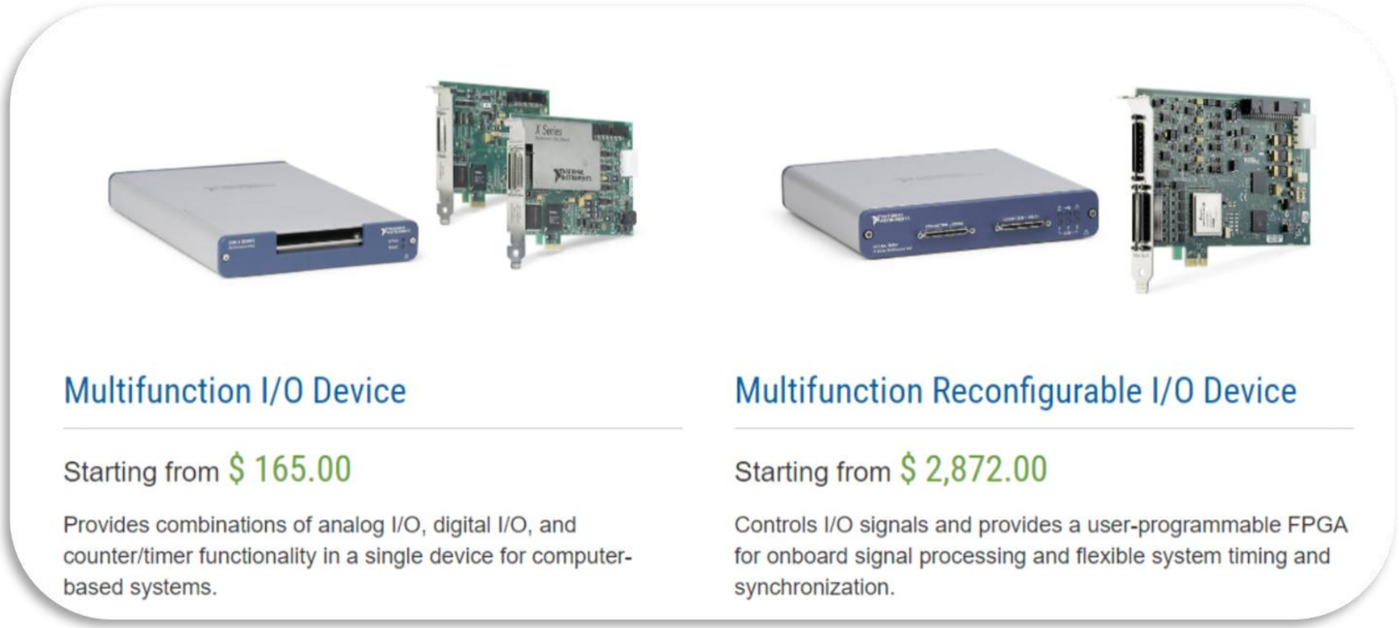
Objective

- ***The main objective of this project was to design a closed-loop pendulum control system with light equipment, less programming, and affordable price.***

- ***It was expected that students are able to purchase the kit and design a controller on their own to apply the knowledge they learned from MEM255: Intro to Control and MEM355: Control Design.***

Objective

- ***Thus, to make the design more affordable, instead of using LabVIEW and NI-DAQ device, "Arduino" was chosen for the project.***



Multifunction I/O Device

Starting from $ 165.00

Provides combinations of analog I/O, digital I/O, and counter/timer functionality in a single device for computer-based systems.

Multifunction Reconfigurable I/O Device

Starting from $ 2,872.00

Controls I/O signals and provides a user-programmable FPGA for onboard signal processing and flexible system timing and synchronization.

NI Product

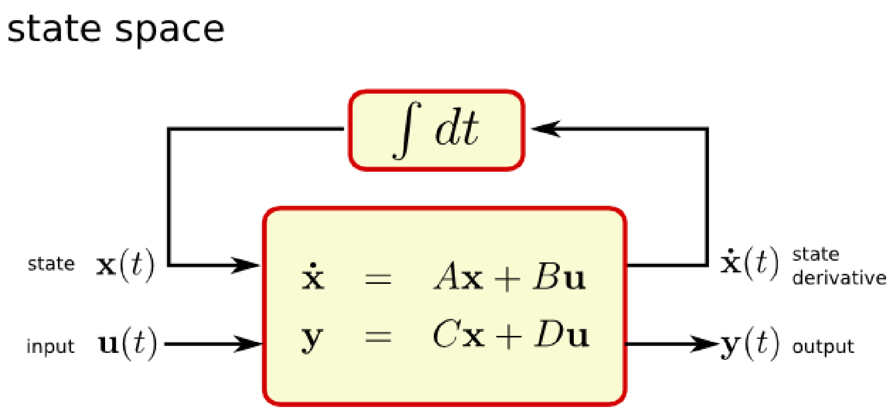
$22.00

Arduino Uno Rev3

Arduino

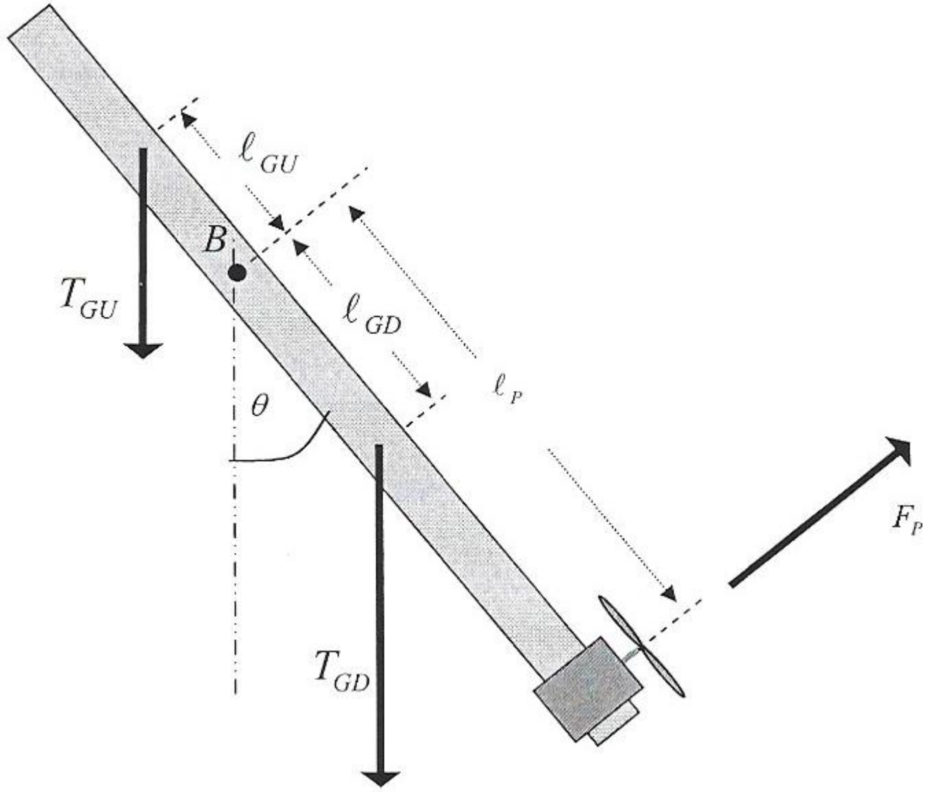
Objective

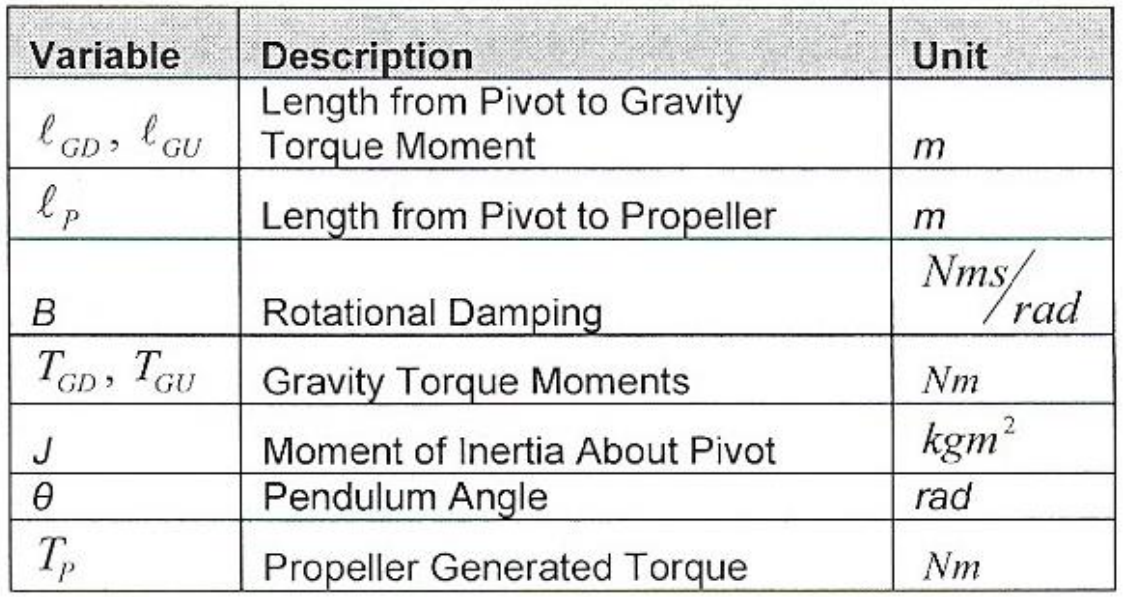
- **Also, instead of using "PID controller", the "State Feedback Controller" was chosen for the project.**


state space
$\int dt$
state $\mathbf{x}(t)$
input $\mathbf{u}(t)$
$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$
$\mathbf{y} = C\mathbf{x} + D\mathbf{u}$
$\dot{\mathbf{x}}(t)$ state derivative
$\mathbf{y}(t)$ output

Problem

- ***The pendulum system can be converted to the following diagram.***





Variable	Description	Unit
ℓ_{GD}, ℓ_{GU}	Length from Pivot to Gravity Torque Moment	m
ℓ_P	Length from Pivot to Propeller	m
B	Rotational Damping	Nms/rad
T_{GD}, T_{GU}	Gravity Torque Moments	Nm
J	Moment of Inertia About Pivot	kgm^2
θ	Pendulum Angle	rad
T_P	Propeller Generated Torque	Nm

[1] MEM351 Dynamic Systems Laboratory I by the Department of Mechanical Engineering, Drexel University. B.C. Chang and Mishah U. Salman (2015). MEM351 Lab II Manual

Problem

- *A simple pendulum is governed by,*

$$J\ddot{\theta} + B\dot{\theta} + T_G sin(\theta) = T_P$$

- *Where T_G is the net gravity torque moment about the pivot, and T_P is the propeller generated torque.*

- *Thus, linearizing the system, and rearranging for torque-voltage relationship yields…*

Problem

- *A transfer function that governs the system.*

$$\frac{\theta(s)}{V(s)} = \frac{\frac{\alpha l_p K_t}{J_{MP} R J}}{(s^2 + \frac{B}{J}s + \frac{T_G}{J})(s + \frac{1}{J_{MP}}(B_{MP} + \frac{K_t K_b}{R}))}$$

- *However, even with many assumption taken to arrive at this model. There are still many parameters that need to be identified!*

Approach to Solve Problem

- ***However, if is often beneficial to forgo the lengthy process by approximating the 2nd Order system!***

$$\boxed{\frac{Y(s)}{U(s)} = \frac{\alpha\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}}$$

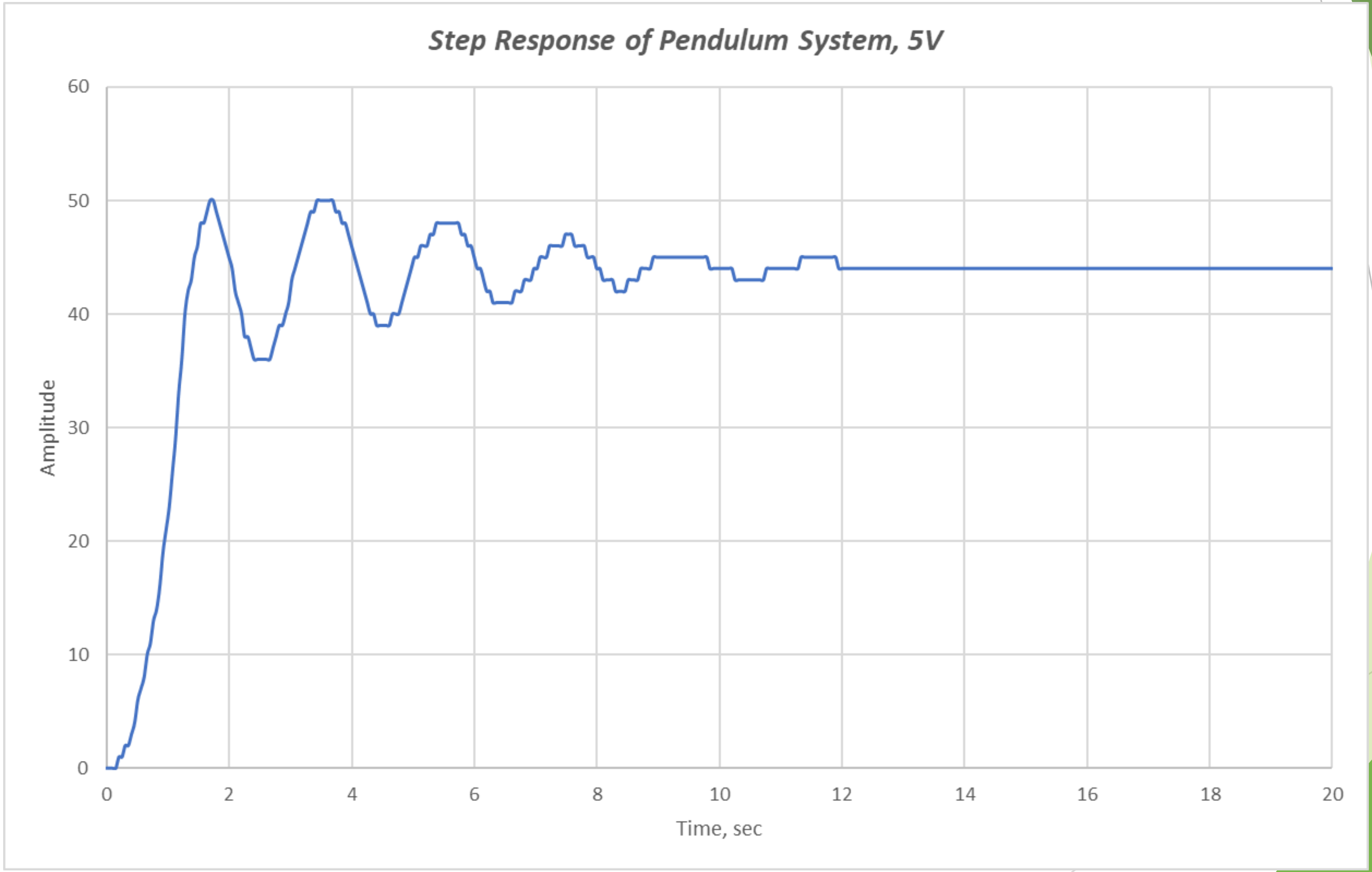
Where,

$$\zeta = \frac{-ln(OS/100)}{\sqrt{\pi^2 + ln^2(OS/100)}} \qquad T_P = \frac{\pi}{\omega_n\sqrt{1 - \zeta^2}}$$

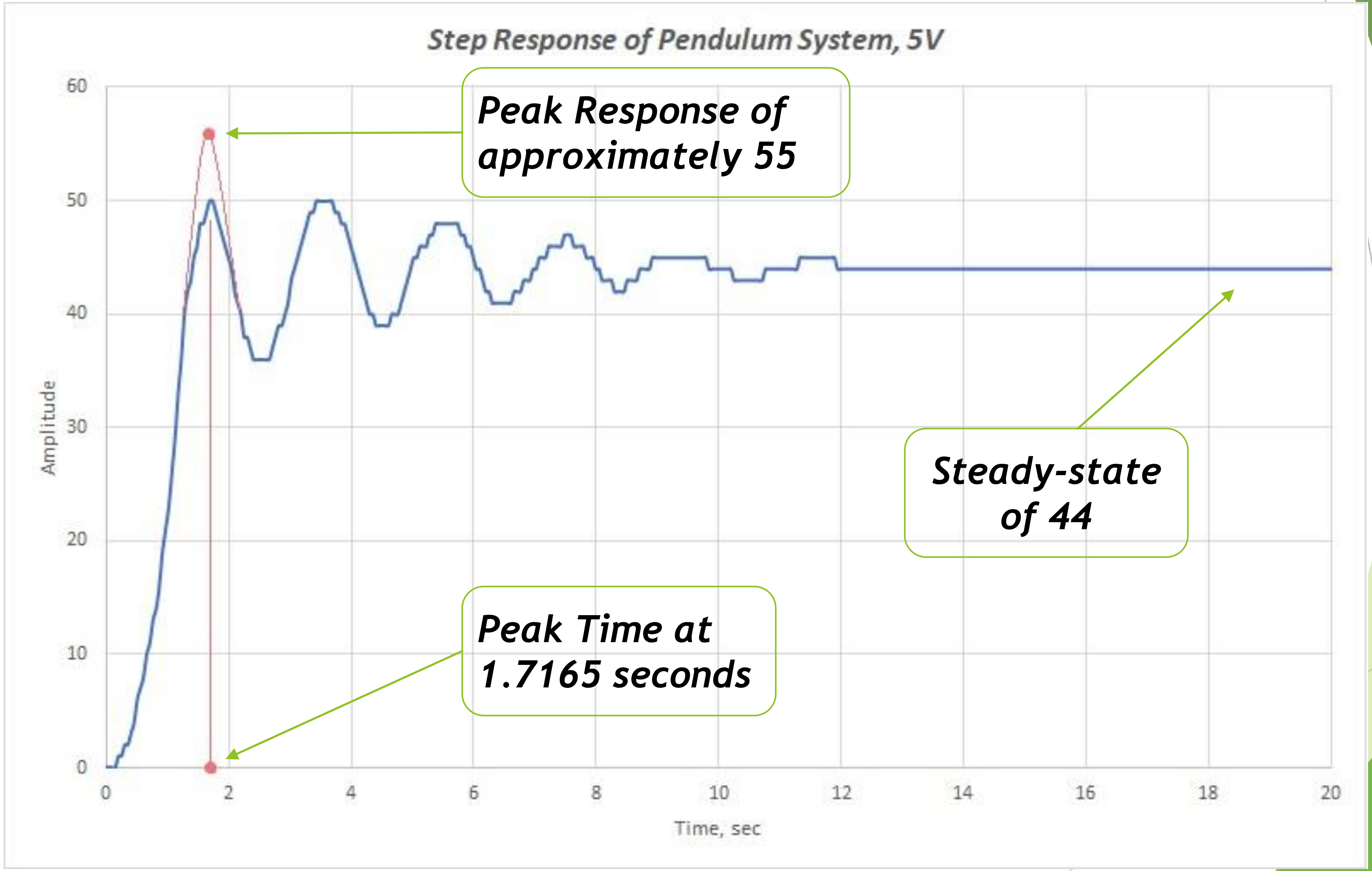
Approach to Solve Problem

- ***OK, in order to use the approximated 2nd Order Model, a response of the pendulum due to a step input is required!***

Approach to Solve Problem



Approach to Solve Problem



Approach to Solve Problem

$$OverShoot = \frac{55-44}{44} = 25\%$$

$$\downarrow$$

$$\zeta = \frac{-ln(25/100)}{\sqrt{\pi^2 + ln^2(25/100)}} = \boxed{0.4037}$$

$$\downarrow$$

$$\omega_n = \frac{\pi}{(1.7165)\sqrt{1-\zeta^2}} = \boxed{2.00\ rad/s}$$

Approach to Solve Problem

- ***Thus, The Approximated 2nd Order Transfer Function was found!***

$$G(s) = \frac{Y(s)}{U(s)} = \frac{\alpha\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

$$= \frac{44 \cdot (2.00)^2}{s^2 \ + \ (2 \cdot 0.4037 \cdot 2.00)s \ + \ (2.00)^2}$$

$$\boxed{= \frac{176.10}{s^2 \ + \ 1.62s \ + \ 4.00}}$$

Approach to Solve Problem

- ***Now, let's design a State-Feedback Controller!***
- ***First, converting the transfer function to state-space representation…***

$$G(s) = \frac{176.10}{s^2 + 1.62s + 4.00}$$

$$\Updownarrow$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -4 & -1.62 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} 176.10 & 0 \end{bmatrix} x$$

Approach to Solve Problem

- **According to the feedback control law,**

$$u = -Kx, where\ K = [K_0\ K_1].$$

- **Thus,**

$$\dot{x} = Ax + Bu = Ax - BKx = (A - BK)x$$

- **We can form a closed A matrix,**

$$A_{CL} = (A - BK).$$

Approach to Solve Problem

- ***Thus,***

$$A_{CL} = \begin{bmatrix} 0 & 1 \\ -4 & -1.62 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} K_0 & K_1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 \\ -4 - K_0 & -1.62 - K_1 \end{bmatrix}$$

- ***Taking the characteristic equation…***

$$det(sI - A_{CL}) = s^2 + (1.62 + K_1)s + (4 + K_0)$$

Approach to Solve Problem

- ***Now, the desired performance for the pendulum system needs to be defined.***

- ***Since the damping ratio of the pendulum was quite low at 0.4037, the desired damping ratio was chosen to be 0.95 to reduce the oscillation behavior of the pendulum.***

- ***Also the desired natural frequency was chosen to be 4 rad/s to reduce the oscillation behavior of the pendulum.***

Approach to Solve Problem

- **Thus, taking $\zeta = 0.95$ and $\omega_n = 4\, rad/s$, the desired characteristic equation is,**

$$s^2 + 7.6s + 16 = 0$$

- **Comparing it with closed A matrix, A_{CL}, the K gain can be found as**

$$s^2 + (1.62 + K_1)s + (4 + K_0) = 0$$

$$\boxed{K = \begin{bmatrix} 12.0 & 5.98 \end{bmatrix}}$$

Approach to Solve Problem

- ***Thus,*** $A_{CL} = (A - BK)$ ***is…***

$$\boxed{A_{CL} = \begin{bmatrix} 0 & 1 \\ -16 & -7.6 \end{bmatrix}}$$

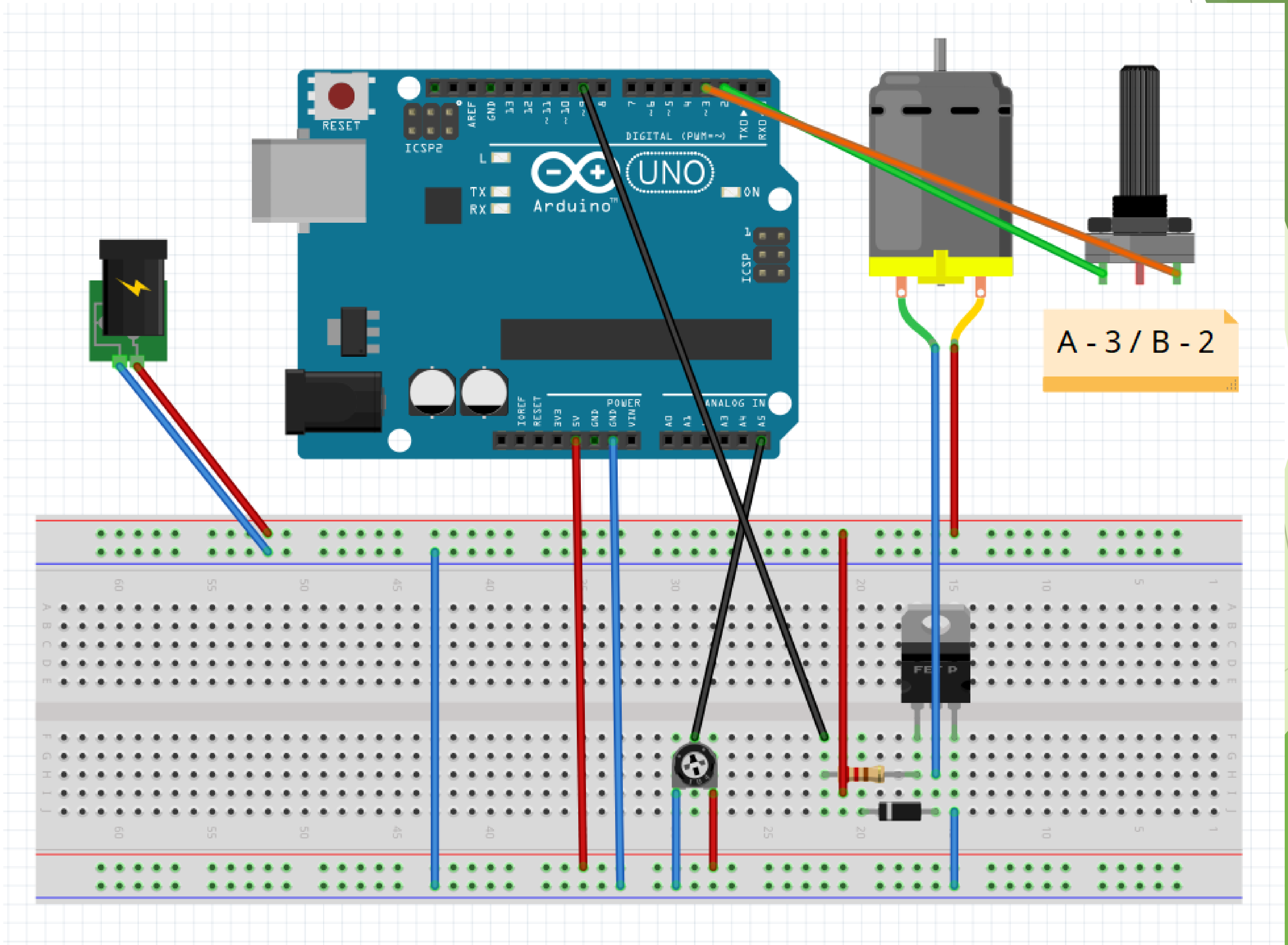
- ***Now! The state-feedback controller was successfully made. Let's implement this to the hardware.***

- ***Circuit Design…***

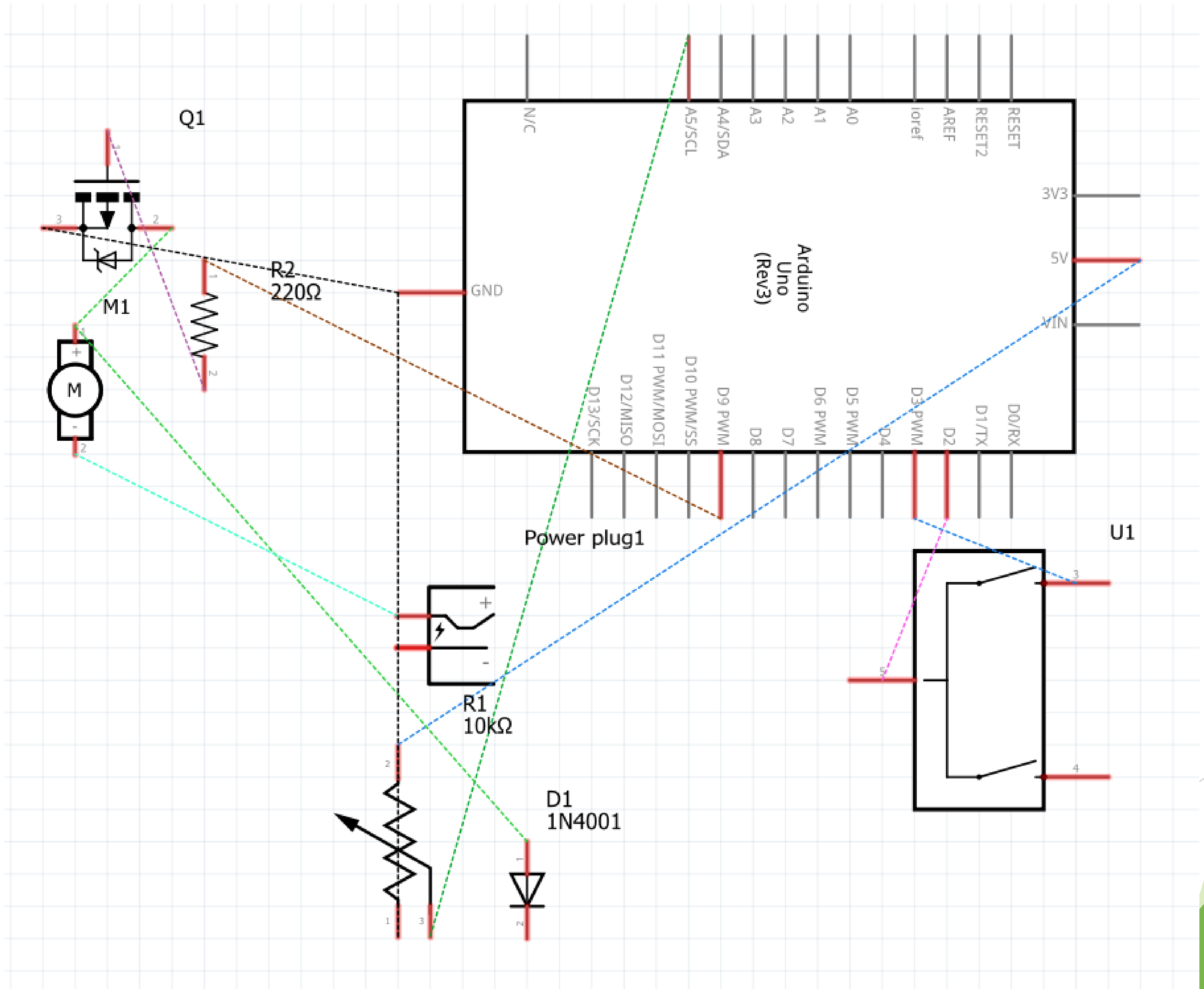
Approach to Solve Problem

- ***Circuit Design***
- ***To operate the Arduino, a electrical circuit must be designed.***
- ***What do we want for Arduino?***
- ***1. Motor Control by adjusting power input.***
- ***2. Data Acquisition to determine the power input.***

Approach to Solve Problem



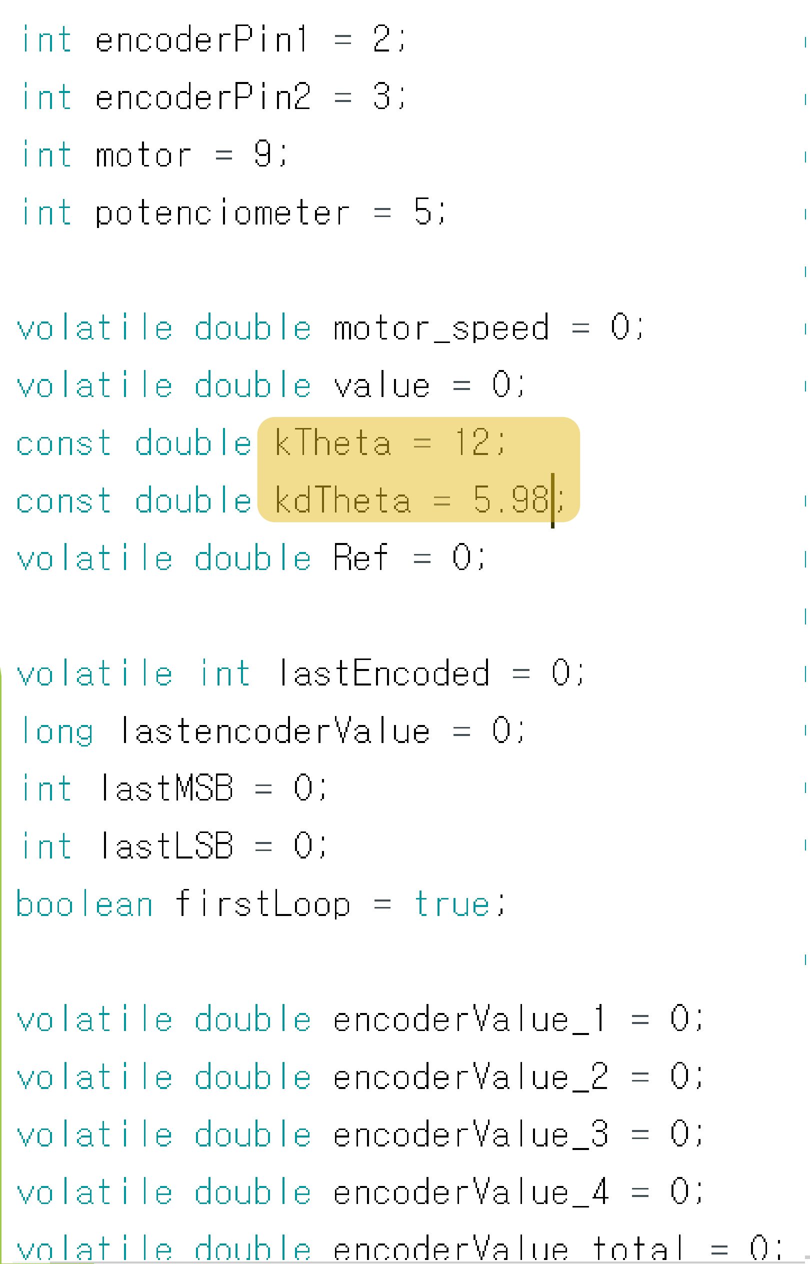
Approach to Solve Problem



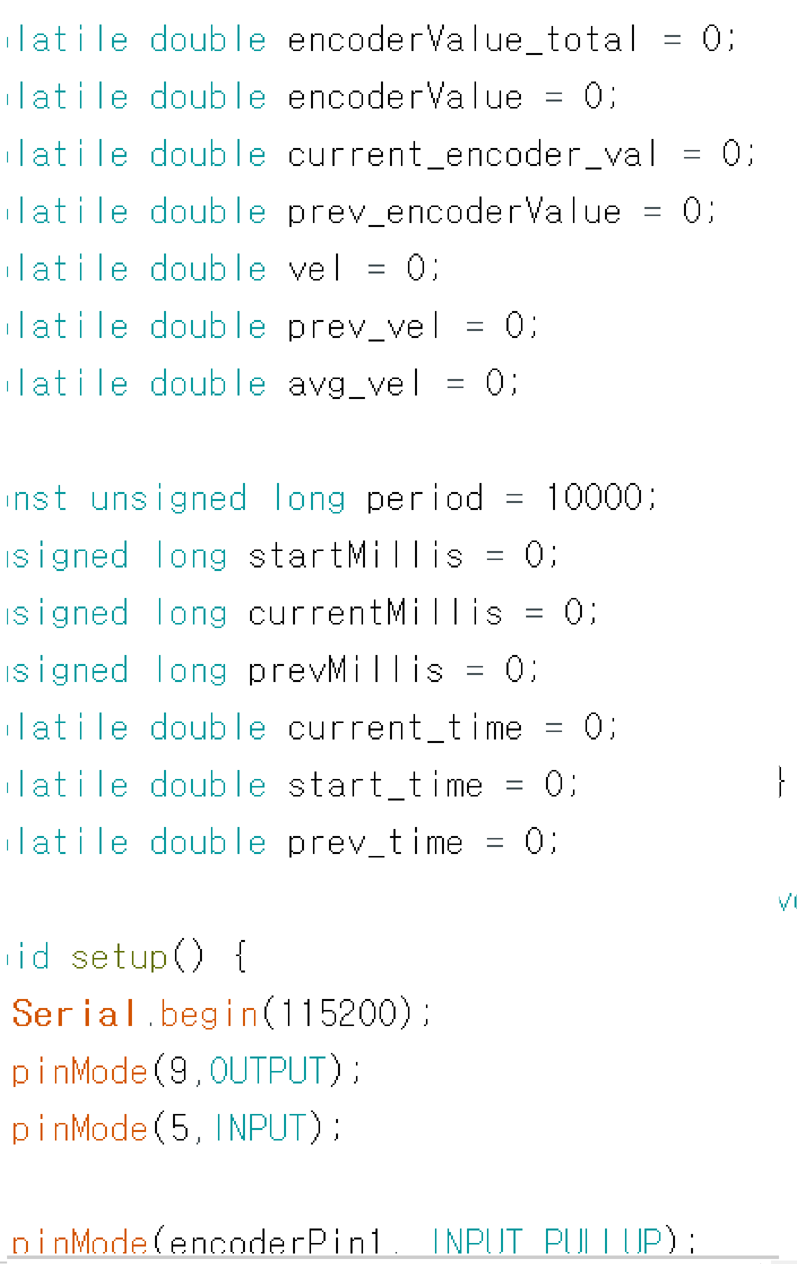
Approach to Solve Problem

- ***Ok, once the circuit is set up. Now is the time to code the Arduino.***

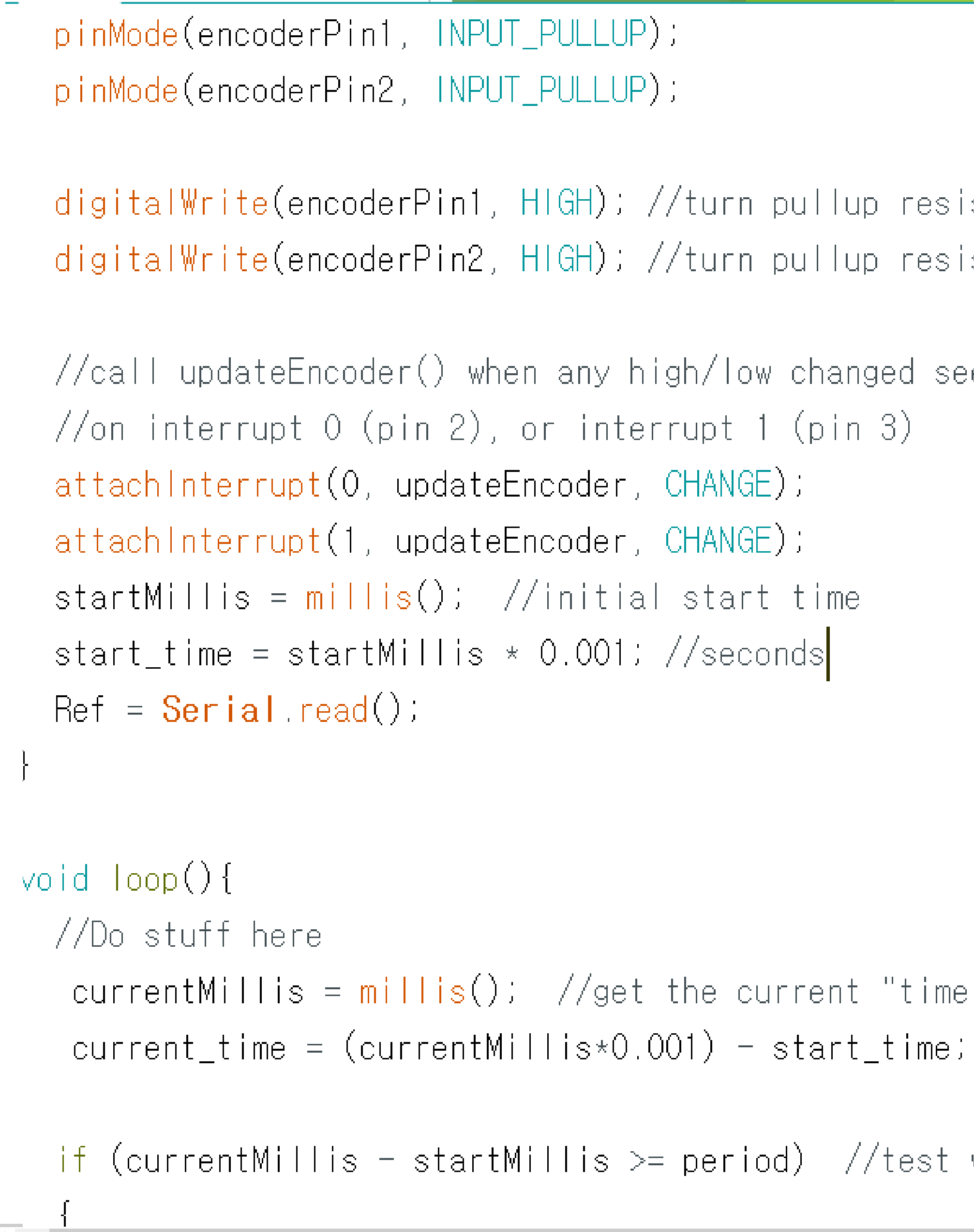
Approach to Solve Problem



```
int encoderPin1 = 2;
int encoderPin2 = 3;
int motor = 9;
int potenciometer = 5;

volatile double motor_speed = 0;
volatile double value = 0;
const double kTheta = 12;
const double kdTheta = 5.98;
volatile double Ref = 0;

volatile int lastEncoded = 0;
long lastencoderValue = 0;
int lastMSB = 0;
int lastLSB = 0;
boolean firstLoop = true;

volatile double encoderValue_1 = 0;
volatile double encoderValue_2 = 0;
volatile double encoderValue_3 = 0;
volatile double encoderValue_4 = 0;
volatile double encoderValue_total = 0;
```

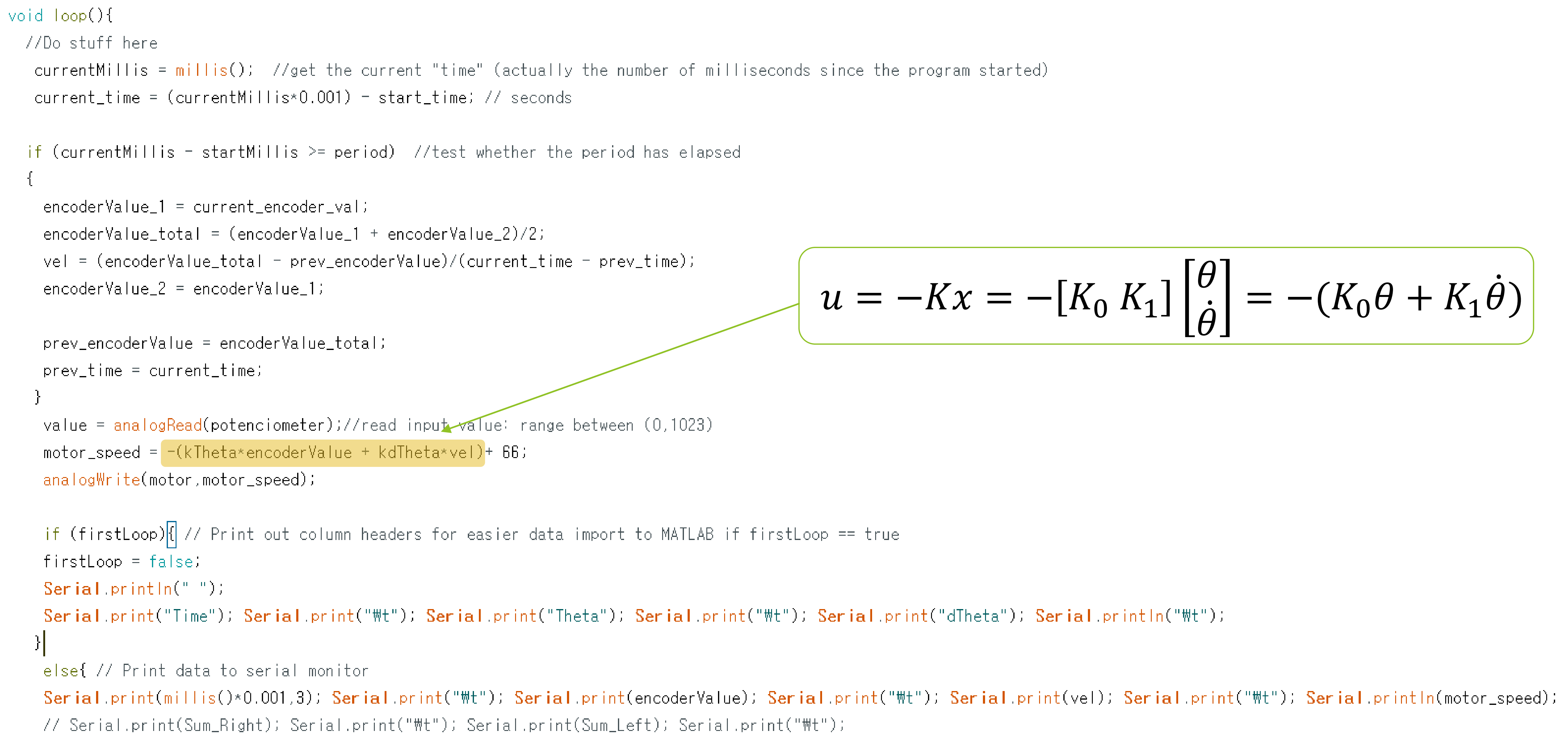


```
latile double encoderValue_total = 0;
latile double encoderValue = 0;
latile double current_encoder_val = 0;
latile double prev_encoderValue = 0;
latile double vel = 0;
latile double prev_vel = 0;
latile double avg_vel = 0;

nst unsigned long period = 10000;
signed long startMillis = 0;
signed long currentMillis = 0;
signed long prevMillis = 0;
latile double current_time = 0;
latile double start_time = 0;
latile double prev_time = 0;

id setup() {
Serial.begin(115200);
pinMode(9,OUTPUT);
pinMode(5,INPUT);

pinMode(encoderPin1, INPUT_PULLUP);
```



```
  pinMode(encoderPin1, INPUT_PULLUP);
  pinMode(encoderPin2, INPUT_PULLUP);

  digitalWrite(encoderPin1, HIGH); //turn pullup resi
  digitalWrite(encoderPin2, HIGH); //turn pullup resi

  //call updateEncoder() when any high/low changed se
  //on interrupt 0 (pin 2), or interrupt 1 (pin 3)
  attachInterrupt(0, updateEncoder, CHANGE);
  attachInterrupt(1, updateEncoder, CHANGE);
  startMillis = millis();  //initial start time
  start_time = startMillis * 0.001; //seconds
  Ref = Serial.read();
}

void loop(){
  //Do stuff here
   currentMillis = millis();  //get the current "time
   current_time = (currentMillis*0.001) - start_time;

  if (currentMillis - startMillis >= period)  //test
  {
```

Approach to Solve Problem



```
void loop(){
  //Do stuff here
   currentMillis = millis();  //get the current "time" (actually the number of milliseconds since the program started)
   current_time = (currentMillis*0.001) - start_time; // seconds

  if (currentMillis - startMillis >= period)  //test whether the period has elapsed
  {
    encoderValue_1 = current_encoder_val;
    encoderValue_total = (encoderValue_1 + encoderValue_2)/2;
    vel = (encoderValue_total - prev_encoderValue)/(current_time - prev_time);
    encoderValue_2 = encoderValue_1;

    prev_encoderValue = encoderValue_total;
    prev_time = current_time;
   }
    value = analogRead(potenciometer);//read input value: range between (0,1023)
    motor_speed = -(kTheta*encoderValue + kdTheta*vel)+ 66;
    analogWrite(motor,motor_speed);

    if (firstLoop){ // Print out column headers for easier data import to MATLAB if firstLoop == true
    firstLoop = false;
    Serial.println(" ");
    Serial.print("Time"); Serial.print("₩t"); Serial.print("Theta"); Serial.print("₩t"); Serial.print("dTheta"); Serial.println("₩t");
   }
    else{ // Print data to serial monitor
    Serial.print(millis()*0.001,3); Serial.print("₩t"); Serial.print(encoderValue); Serial.print("₩t"); Serial.print(vel); Serial.print("₩t"); Serial.println(motor_speed);
    // Serial.print(Sum_Right); Serial.print("₩t"); Serial.print(Sum_Left); Serial.print("₩t");
```

$$u = -Kx = -[K_0\ K_1]\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} = -(K_0\theta + K_1\dot{\theta})$$

Approach to Solve Problem

```
  }
   Serial.flush(); // Force data transmission to complete before continuing
   //delay(1000); //just here to slow down the output, and show it will work  even during a delay
}

void updateEncoder(){
  int MSB = digitalRead(encoderPin1); //MSB = most significant bit
  int LSB = digitalRead(encoderPin2); //LSB = least significant bit

  int encoded = (MSB << 1) |LSB; //converting the 2 pin value to single number
  int sum  = (lastEncoded << 2) | encoded; //adding it to the previous encoded value

  if(sum == 0b1101 || sum == 0b0100 || sum == 0b0010 || sum == 0b1011) encoderValue ++;
  if(sum == 0b1110 || sum == 0b0111 || sum == 0b0001 || sum == 0b1000) encoderValue --;

  lastEncoded = encoded; //store this value for next time
  current_encoder_val = encoderValue;
}
```

Approach to Solve Problem

- ***Circuit and Codes are complete.***
- ***What else?***
- ***I have to make sure I get the "accurate" state data for angular displacement, θ, and angular velocity, $\dot{\theta}$.***
- ***The encoder can read the angular displacement data, but what about angular velocity?***

Approach to Solve Problem

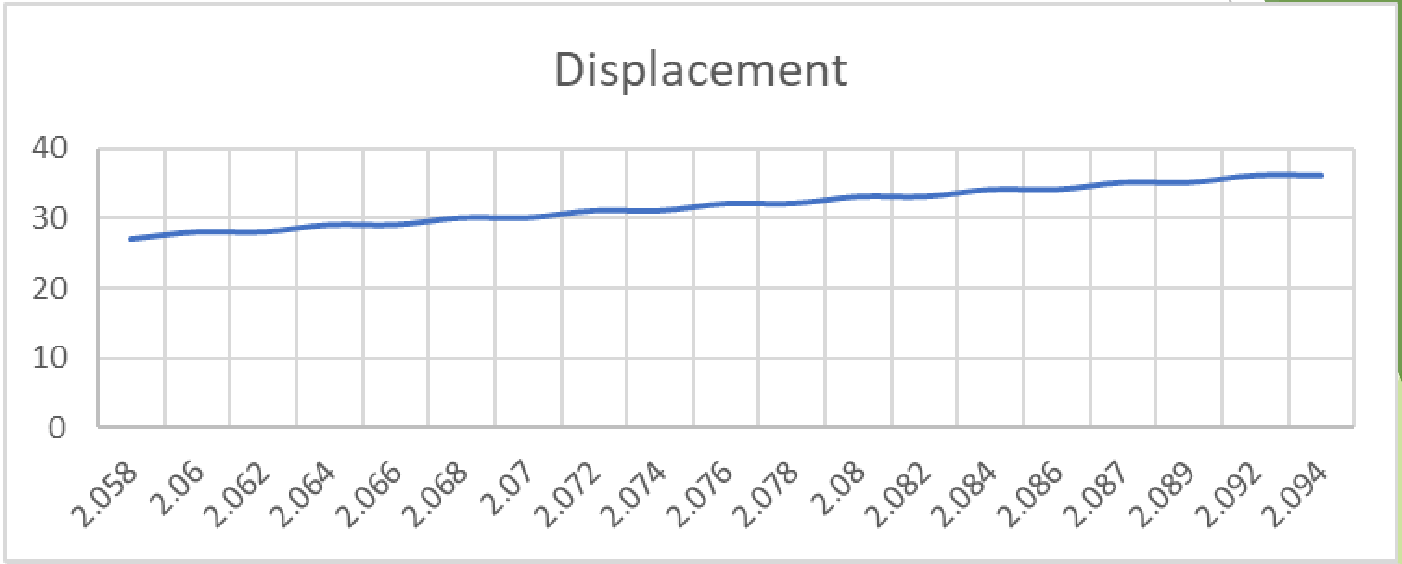
- ***Angular Velocity was computed based on following formula…***

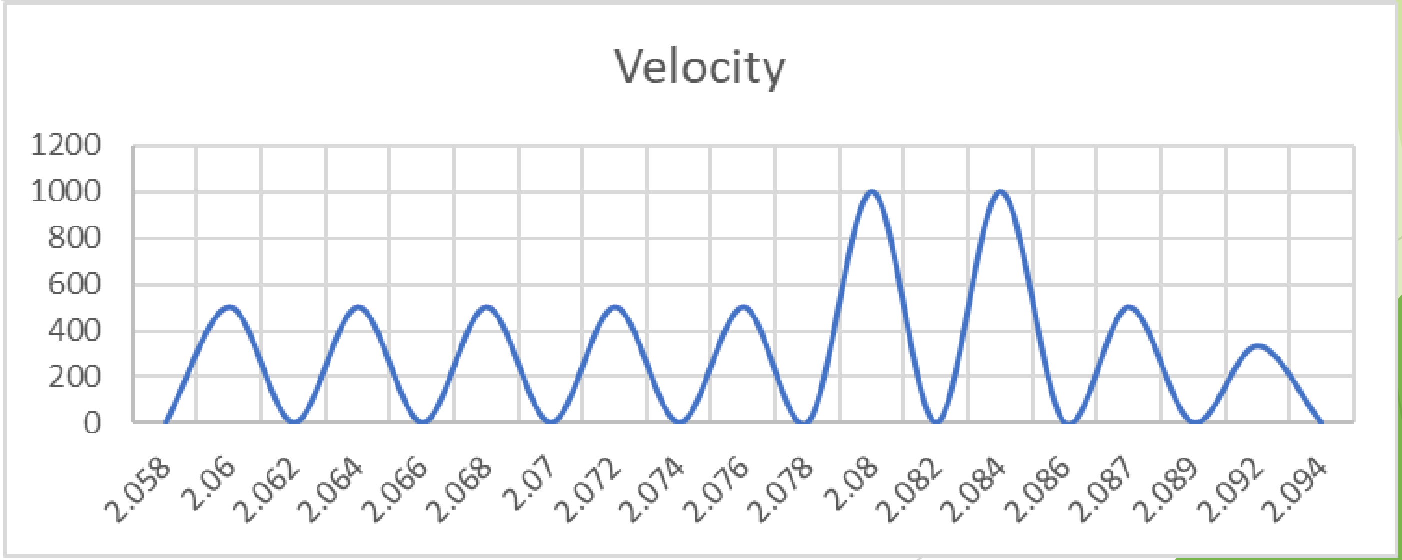
$$\boxed{\dot{\theta} = \frac{\theta_2 - \theta_1}{t_2 - t_1}}$$

- ***OK let's see the result…***

Approach to Solve Problem

Time	Displacement	Velocity
2.058	27	0
2.06	28	499.98
2.062	28	0
2.064	29	499.98
2.066	29	0
2.068	30	499.98
2.07	30	0
2.072	31	500.04
2.074	31	0
2.076	32	499.98
2.078	32	0
2.08	33	1000.07
2.082	33	0
2.084	34	999.83
2.086	34	0
2.087	35	499.98
2.089	35	0
2.092	36	333.33
2.094	36	0





Approach to Solve Problem

- ***The encoder reads the angular displacement well. However, the angular velocity is very inaccurate…***

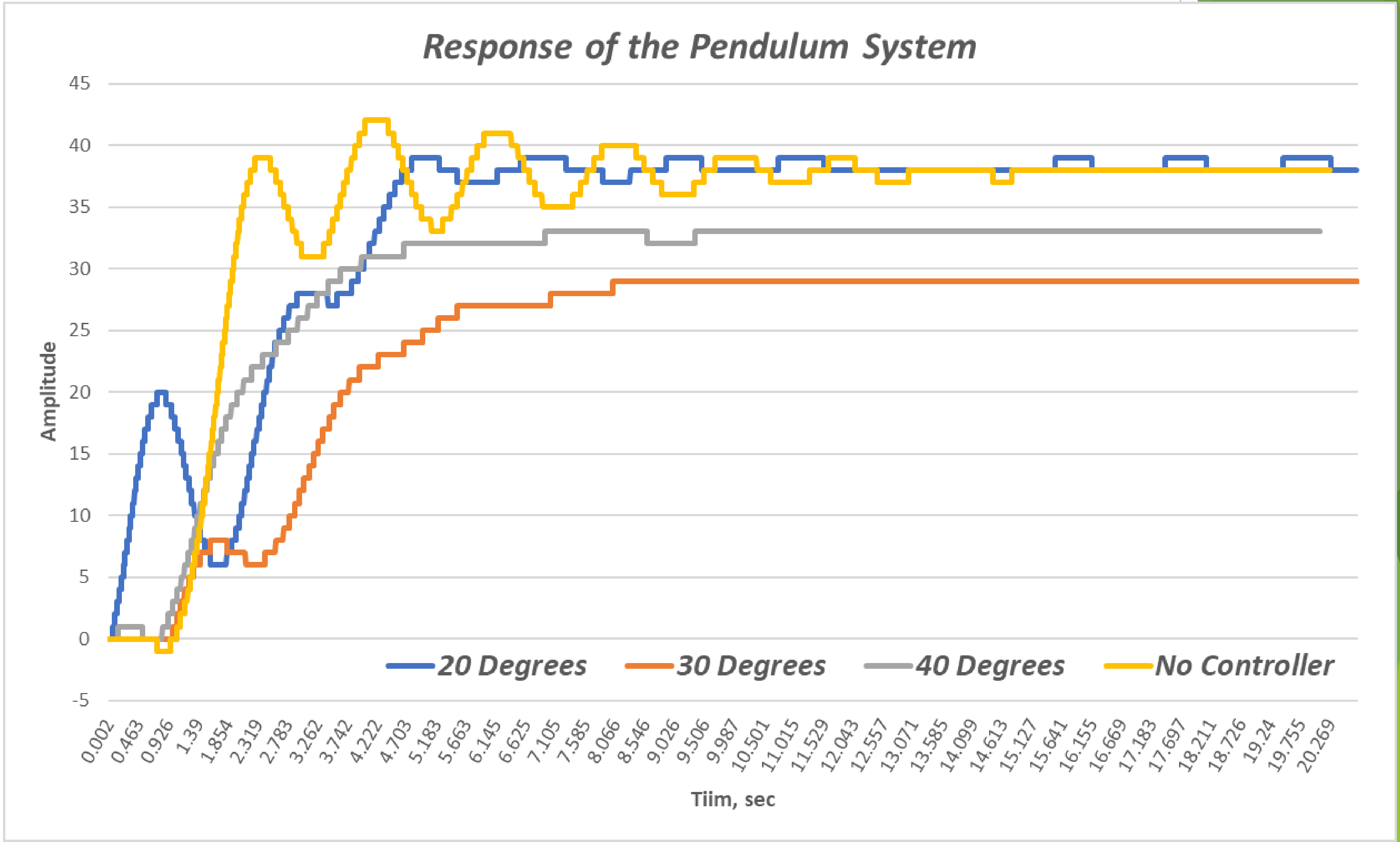
- ***This is where I face the problem.***

- ***Possible Solution: Moving Average, Various Filter Design, and Observer Controller.***

Approach to Solve Problem

- ***Since there was not enough time, the inaccuracy of the angular velocity was not fixed yet.***
- ***Next person can continue working on the pendulum control design.***
- ***However, let's still see the result with inaccurate velocity data to see if the pendulum is controlled.***

Result: kTheta = -2.04; kdTheta = 1.605;



Conclusion

- *The pendulum system was sufficiently controlled by the designed state-feedback controller with the Arduino Program.*
- *Although there need to be more research in improving the performance of the system, the project can be sufficiently used for individual student to practice his or her knowledge gained from the Control Classes.*
- *Overall, the project was successful.*

THANKS!

Reference

[1] MEM351 Dynamic Systems Laboratory I by the Department of Mechanical Engineering, Drexel University. B.C. Chang and Mishah U. Salman (2015). MEM351 Lab II Manual