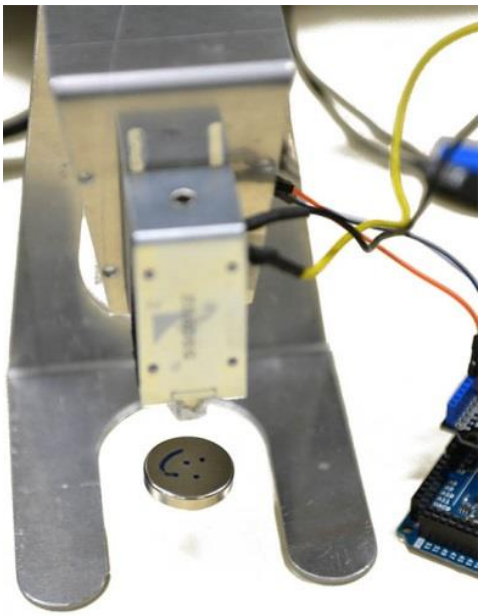
ELEC 3004/7312: Digital Linear Systems: Signals & Control!

Prac/Lab 4 LeviLab: Part II: Feedback Control

May 16, 2017 (by C. Reiger & S. Singh)

Pre-Lab



This laboratory considers system modelling and control as it applies to a levitating magnetic mass. Much like a magnetic bearing in turbomachinery, this suspends ferromagnetic material (or a weight to which a magnet has been attached) by means of an electromagnet whose current is controlled by the position of the mass.

Following on from LaviLab Part I (Lab 3), this lab focuses on the process of designing and tuning a discrete-time feedback controller, notably one using a PID control law.

Before starting this Lab, be sure to have completed the **Modelling and Simulation Exercises (of Lab 3)**, which include:

1. The continuous transfer function from I to x (i.e., $G(s) = X(s)$).
2. A candidate digital control design for the magnetic levitation device so that the closed-loop system has a rise time of less than 160 milliseconds ($t_r \leq 0.159s$), a settling time of less than 460 milliseconds ($t_s \leq 0.46s$), and overshoot ($M_p \leq 20\%$).
3. A plot of the step response of your design to an initial (unit) disturbance.

Laboratory Safety

Please Handle Carefully!

This laboratory involves Neodymium rare earth magnets. These magnets are very strong (some of these magnets have surface fields of 7000 Gauss and holding forces of 30 kilograms). Please note that they are:

- **Not a toy**
- A choking hazard. Do not swallow.
- A suffocation hazard
- A pinch hazard. These magnets can snap together very quickly. Keep fingers clear.
- A nickel coated sintered ceramic. **They are very fragile**.
- Combustible. Do not grind or cut.
- Can erase credit cards and other magnetic storage media. Keep >30 cm away.
- Should not be heated above room temperature ($T \ll 80°C$)
- **Not to be removed from the laboratory room**

This laboratory has a **strict safety policy**. Inappropriate handling is not only dangerous, but not fair.

Any violations (or perceived violations) of safety policy will result in **immediate removal from the room**

Laboratory Completion & Extra Credit Points

Laboratory Completion:
Please work **together** on the lab in groups of 2-3.

Extra credit points:

- +1 : Developing **your own** control strategy (PWM, etc.) and controller (including implementing it)
- +2 : Tuning the control code and answering lab questions
- + ½ : Successful demonstration of levitation
- + ½ : For **completely** returning the parts back in the kit/box

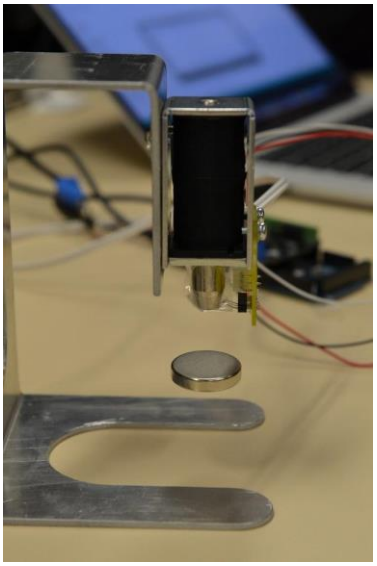
LeviLab II: Digital Feedback Control

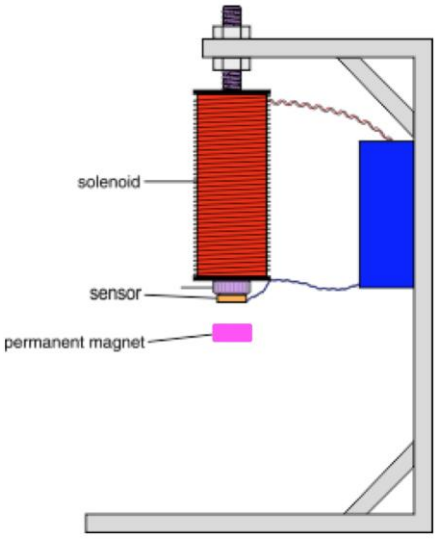
Equipment & Tools:
This is the same as LeviLab I. Please review the Lab 3 handout for more information and an assembly guide.

Goals of the Laboratory:

1. Modelling and Simulation of the LeviLab
2. Understanding the nature of linear digital feedback control laws
3. Implementation and tuning of control code for the LeviLab

Scenario:
Labs 3 & 4 work towards building a magnetic levitation system using a digital PID controller under feedback control. The scenario is outlined below:





Overview:
The equation of motion is: $m\frac{d^2x}{dt^2} = mg + f(x, I)$
where the force on the ball due to the electromagnet is given by $f(x, I)$. At equilibrium the magnet force balances the gravitational force.

Let I_0 represent the current at equilibrium, then we can linearize this system about $x = 0$ and $I = I_0$ to get:
$m\frac{d^2x}{dt^2} = k_1x + k_2i$

We know our model is characterised by the following equations:

Actual System: $m\frac{d^2x}{dt^2} = -mg + f(x, I)$

Linearised System: $m\frac{d^2x}{dt^2} = k_1x(t) + k_2i(t)$

Our Simplified Model can therefore be written as:

$G(s) = \frac{X(s)}{I(s)}$

We take the laplace transform of the linearised system:

$\mathcal{L}(m\frac{d^2x}{dt^2} = k_1x(t) + k_2i(t))$

$m(s^2X(s) - sx(0) - x'(0)) = k_1X(s) + k_2I(s)$

We know that x(0) = 0 and x'(0) = 0. Therefore:

$(ms^2 - k_1)X(s) = k_2I(s)$

$G(s) = \frac{X(s)}{I(s)} = \frac{k_2}{ms^2 - k_1} = \frac{k_2/m}{s^2 - k_1/m}$

Discretising with a 10kHz sampler, we take the ZOH:

$T = 1/10000 = 0.0001$ sec

$G(z) = (1 - z^{-1})\mathcal{Z}(\frac{G(s)}{s})$

As an example, taking m=0.05kg, k1=50N/m, k2=0.05N/A, we achieve the following:

$G(z) = (1 - z^{-1})\mathcal{Z}(\frac{1}{s(s^2 - 50)})$

Performing partial fraction expansion:

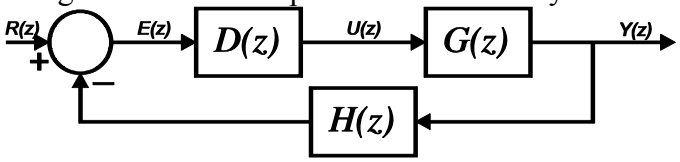
$\frac{1}{s(s-\sqrt{50})(s+\sqrt{50}} = \frac{-1/50}{s} + \frac{1/100}{s-\sqrt{50}} + \frac{1/100}{s+\sqrt{50}}$

$\therefore G(z) = (1 - z^{-1})(\frac{-1}{50}\frac{z}{z-1} + \frac{1}{100}\frac{z}{z-e^{\sqrt{50}T}} + \frac{1}{100}\frac{z}{z-e^{-\sqrt{50}T}})$

$G[z] = 5.002 * 10^{-5}\frac{(z+1)}{z^2 - 2.005z + 1}$

Empirical PID Control Tuning: Ziegler-Nichols Tuning

Let us start with a classic linear regulator closed-loop feedback control system



Its closed loop transfer function is

$$\frac{Y(z)}{R(z)} = \frac{D(z)G(z)}{1 + D(z)G(z)H(z)}$$

The goal is to determine the controller, $D(z)$. Recall that the definition of a PID control law is given by:

$$\mathrm{u}(t) = \mathrm{MV}(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{d}{dt}e(t)$$

This can be expressed as a digital controller in the digital domain as:

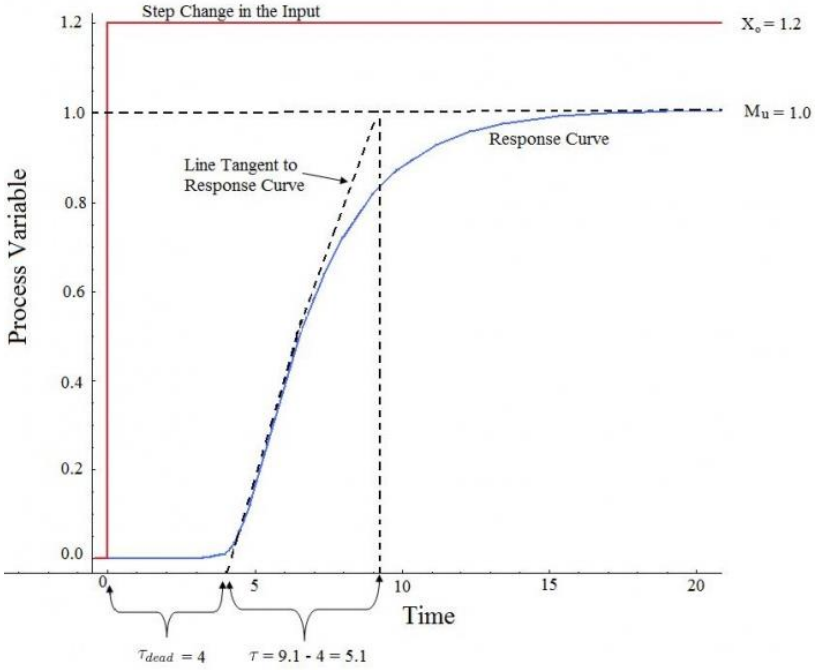
$$\frac{U(z)}{E(z)} = D(z) = K_p + K_i\left(\frac{Tz}{z-1}\right) + K_d\left(\frac{z-1}{Tz}\right)$$

This can be expressed as a difference equation by applying the backward difference rule, $e(kT) = \frac{1}{T}[e(kT) - e((k-1)T)]$. Simplifying this (and expressing $e(kT) = e(k)$) gives:

$$u(k) = \left[K_p + K_i T + \left(\frac{K_d}{T}\right)\right] \cdot e(k) - [K_d T] \cdot e(k-1) + [K_i] \cdot u(k-1)$$

One strategy for empirically tuning this is Ziegler-Nichols Tuning. As noted in Section 5.8 of Franklin, Powell, and Workman (2nd Ed.), this can be considered as a transient-response tuning method that uses the step response of the open-loop system to help estimate PID control gains. As shown in the graph below (from Michigan's Feedback Control & Matlab Wiki), we consider two parameters from the response:

- the time delay, **L** (also referred to as $\boldsymbol{\tau_{dead}}$)
- the reaction rate, **R** (also referred to by the rise time, $\boldsymbol{\tau}$, and ultimate steady-state value, $\boldsymbol{M_u}$)



From this we can calculate a system gain, K_o as

$$K_o = \frac{X_o}{M_u}\frac{\tau}{\tau_{dead}}$$

These values are then used with a gain estimation table to get initial values for the PID gains:

	K_p	T_i	T_d
P	K_o		
PI	$0.9K_o$	3.3L	
PID	$1.2K_o$	2L	0.5L

For more information on this and the closed-loop Ziegler-Nichols tuning method, see also the online tool at: **The University of Michigan's Feedback Control & Matlab Wiki.**

Modelling This in Matlab

Enter the following code into MatLab, you should see the same G(z) output to the window, verifying our above results.

```
m = 0.05; k1 = 50; k2 = 0.05;
plant = tf([k2],[m, 0, -k1]);
Gz = c2d(plant,0.0001, 'zoh')
```

Now inspect we can use root locus design tools on the system. Enter the following into MatLab:

```
rltool(Gz)
```

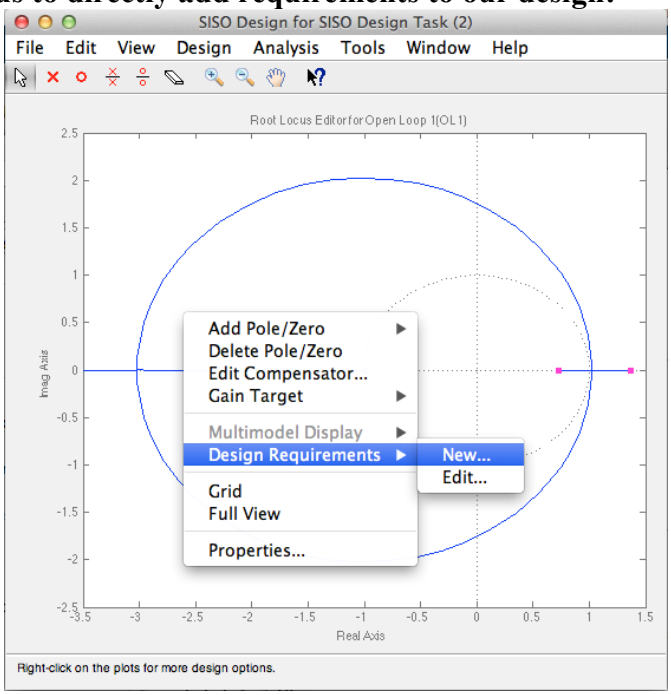
Recall the following design specifications:

$$t_r \leq 0.159s \rightarrow t_r \approx \frac{1.8}{w_0}$$

$$t_s \leq 0.46s \rightarrow t_s = \frac{4.6}{\zeta w_0}$$

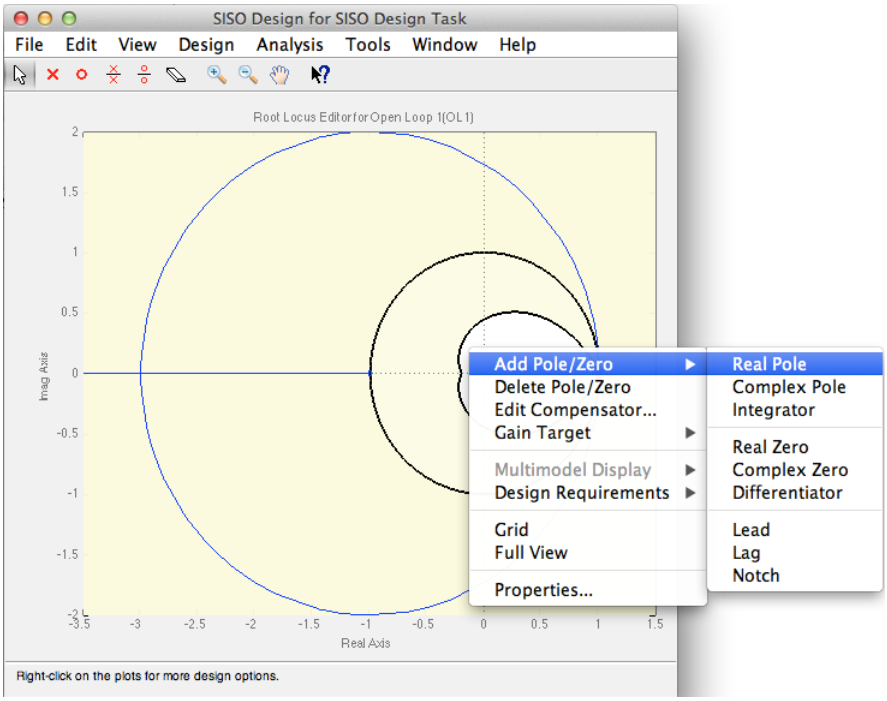
$$M_p \leq 20\% \rightarrow M_p \approx e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}} \rightarrow \zeta = \sqrt{\frac{ln(\frac{PO}{100\%})^2}{\pi^2 + ln(\frac{PO}{100\%})^2}}$$

MatLab's SISO tool allows us to directly add requirements to our design:



>>Add Settling time, Percent Overshoot and Natural Frequency constraints to the design.
Note you will have to calculate the natural frequency using the above equations.

PID / PD Control Design and Tuning (Using Pole/Zero methods):
We know that the total closed loop system must reside in the secant region as modeled above after adding constraints.



For simplicity, firstly design a PD controller, adding a single pole and single zero to the design. You may wish to look at the lecture notes for the discrete transfer function of a PD controller.

Adjust the gain K through sliding the purple 'handle' on the plot.

Finally, click Design-> Edit compensator. This allows you to find your design parameters.

PID Control Code Template

One handy step in extending the code template from Part I (Lab 3) is to modify it adjust the P, I, D parameters without having to reprogram the target board.

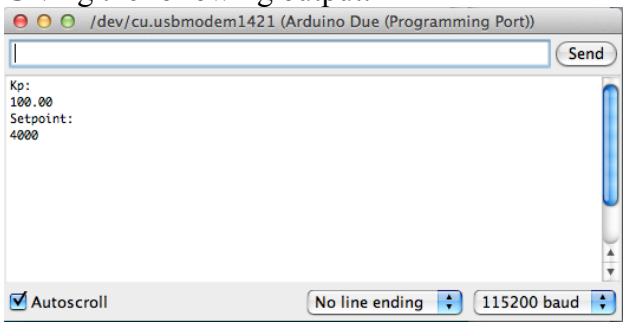
```
// ELEC3004 Levitation Lab
// PID Online Adjustment Method (Add to Main Loop)

// Serial Buffer
char strValue[20];
int v = 0;

// Add to Main Loop, Extend to adjust P,I,D,Setpoint.
#ifdef _SERIAL
  if(Serial.available())
  {
      char ch = Serial.read();
      if(v < 20 && isDigit(ch) ){
          strValue[v++] = ch; // add the ASCII character to the string;
      }
      else
      {
          // here when buffer full or on the first non digit
          strValue[v] = 0; // terminate the string with a 0
          v = 0;
          if(ch == 'p') { // proportional adjust e.g. "270p"
              Kp = atoi(strValue);
              Serial.println( "Kp: " );
              Serial.println( Kp );
          }
      }
  }
#endif
```

In the code provided you are able to tune parameters P, I, D, S corresponding to the PID tuning constants and the desired levitation set-point seen by the hall effect sensor.

E.g. **To adjust P to 100 and S to 4000**, enter the following into the serial monitor window:
100p
4000s
Giving the following output:



To print all parameters, press ‘a’.

As an additional exercise you may write an interrupt routine to achieve an accurate equal time delay in the controller without adjusting the loop delay manually. (See: http://playground.arduino.cc/Code/Timer)

Lab Activities

Activity 0:
Follow the instructions carefully as described in the section "LeviLab: Modelling & PID". Design your controller in MatLab.

Comment on the system stability before and after adding your controller.

Activity 1:
After having designed a controller in MatLab, apply this to your Arduino controller.

> *>>Note, parameters may not be the same as per MatLab due to scaling, etc. Please get a feel for what each parameter does through purposefully de-tuning the parameters in MatLab and observing the response to aid in correcting your real life response.*

Lab Questions

Question 1:
What is the furthest distance (mm) achievable keeping power supply Vs <16 volts.?

Question 2:
Use two magnets and join them such that the mass of the object is doubled. How does this affect the transfer function of the system? What parameters must be adjusted to regain control of the system ?

Question 3:
How does the PWM frequency effect the system ?

Question 4:
How does the PID loop frequency effect the system stability? Try lowering the loop frequency to 100 Hz.

Appendix I: Specifications

1. **Solenoid + Electromagnetic plunger**
   Jaycar SS0902
   Voltage: 12VDC, Current: 0.5A, DC Resistance: 24ohms, Rated Power: 6W,
   Pulling force: 110g-1.78Kg (continuous), total weight: 205g, Plunger weight: 39g.

2. **Hall Effect Sensor A1302 – See manual**

3. **Arduino Motor Shield (L298N) (SKU_DRI0009) – See manual**

4. **Neodymium Magnets – See manual**

Appendix II: PID Code Example

```
// ELEC3004 Levitation Lab
// Uses a digital PID controller
// Template by ChrisRieger(.com)

// PWM frequency
#define PWM_FREQ   25000
// Serial On?
#define _SERIAL
// Debug On?
#define _DEBUG
// Number of samples to average over
#define ANALOGDIV 5
// P-Gain input fine control
#define PGAININPUTDIV 1
// D-Gain input fine control
#define DGAININPUTDIV 1
// I-Gain input fine control
#define IGAININPUTDIV 1000

// On the Due, pins 6,7,8,9 are hardware PWM pins.
int pwmPin = 7; // PWM the DIR pin on the H-Bridge
int enablePin = 6; // Enable the PWM pin on the H-Bridge
int sensorPin = 0;    // For Analog Data IN
int sensorValue;
int i=0, j=0;

// Serial Buffer
char strValue[20];
int v = 0;

// PID
float actual_error = 0, error_previous = 0;
float P, I, D;
float Kp = 0, Ki = 0, Kd = 0;
int setpoint = 3000;

// Setup
void setup() {
  #ifdef _SERIAL
    Serial.begin(115200);
  #endif
  #ifdef _DEBUG
    pinMode(2, OUTPUT);
  #endif

  // Ensure 12 bit Read / Write
  analogReadResolution(12);
  analogWriteResolution(12);
  // Setup PWM output
  pinMode(enablePin, OUTPUT);
  digitalWrite(enablePin, HIGH);
  pinMode(pwmPin, OUTPUT);
  analogWrite(pwmPin, 2048); // 50% duty Cycle
  PWMC_ConfigureClocks(PWM_FREQ * PWM_MAX_DUTY_CYCLE , 0, VARIANT_MCK);
}

// PID Function
int Calculate_PID(long set_point, long measured_value)
{
  int temp;
  error_previous = actual_error;  // error_previous holds the previous error
  actual_error = set_point - measured_value;

  // PID
  P  = actual_error;    // Current error
  I += error_previous;  // Sum of previous errors
  D  = actual_error - error_previous;  // Difference with previous error

  temp = Kp*P + Ki*I + Kd*D;

  return temp;
}
```

```
// Main Loop
void loop() {
  sensorValue = 0; // reset
  for(j = 0; j < ANALOGDIV; j++){ // Average over j cycles
    sensorValue += analogRead(sensorPin);
  }

  int temp = 2048 + Calculate_PID( setpoint, sensorValue/ANALOGDIV);
  temp = (temp>4095) ? 4095 : temp; // Limits
  temp = (temp<0) ? 0 : temp; // Limits

  analogWrite(pwmPin, temp);

  delayMicroseconds( 100 ); // 0.1ms

  #ifdef _SERIAL
    if( Serial.available())
    {
      char ch = Serial.read();
      if(v < 20 && isDigit(ch) ){
        strValue[v++] = ch; // add the ASCII character to the string;
      }
      else
      {
        // here when buffer full or on the first non digit
        strValue[v] = 0; // terminate the string with a 0
        v = 0;
        if(ch == 'p') { // proportional adjust
          Kp = (float(atoi(strValue))/IGAININPUTDIV);
          Serial.println( "Kp: " );
          Serial.println( Kp );
        }
        if(ch == 'd') { // derivative adjust
          Kd = (float(atoi(strValue))/DGAININPUTDIV);
          Serial.println( "Kd: " );
          Serial.println( Kd );
        }
        if(ch == 'i') { // integral adjust
          Ki = (float(atoi(strValue))/IGAININPUTDIV);
          Serial.println( "Ki: " );
          Serial.println( Ki );
        }
        if(ch == 's') { // Setpoint adjust
          setpoint = atoi(strValue);
          Serial.println( "Setpoint: " );
          Serial.println( setpoint );
        }
        if(ch == 'a') { // Print Parameters
          Serial.println( "Kp: " );
          Serial.println( Kp );
          Serial.println( "Kd: " );
          Serial.println( Kd );
          Serial.println( "Ki: " );
          Serial.println( Ki );
          Serial.println( "Setpoint: " );
          Serial.println( setpoint );
        }
      }
    }

    //Serial.println( sensorValue/ANALOGDIV );
  #endif

  #ifdef _DEBUG
    // Allows us to know the PID loop speed through external pin.
    i = ~i;
    digitalWrite(2, i);
  #endif

  //while (0);  // Halt
}
```

END OF LAB 4

Experiment IV:

LeviLab: Part II: Feedback Control

Name: _____

Student ID: _____

Date: _____

Group Name/Members: _____

_____

_____

Part Completed

I. **Laboratory Pre-Lab:**

Developing your own control strategy (PWM, etc.) & controller, including implementing it

II. Tuning Parameters

How does P, I and D affect the response of this system?

III. Stable Magnetic Levitation Controller

What parameters of P, I and D did you calculate / use to achieve levitation?

How can this controller be made more robust?

IV. Kit Return

Please kindly properly disassemble and return the kit to its box