Design of a remote-controlled mount for IOTA's magnet undulator

Gabriel Adrian Mihu
mihu.gabriel.adrian@gmail.com

About IOTA

The Integrable Optics Test Accelerator (IOTA) is a storage ring for advanced beam physics research currently being built and commissioned at Fermilab. It will operate with protons and electrons using injectors with momenta of 70 and 150 MeV/c, respectively. The research program includes the study of nonlinear focusing integrable optical beam lattices based on special magnets and electron lenses, beam dynamics of space-charge effects and their compensation, optical stochastic cooling, and several other experiments.

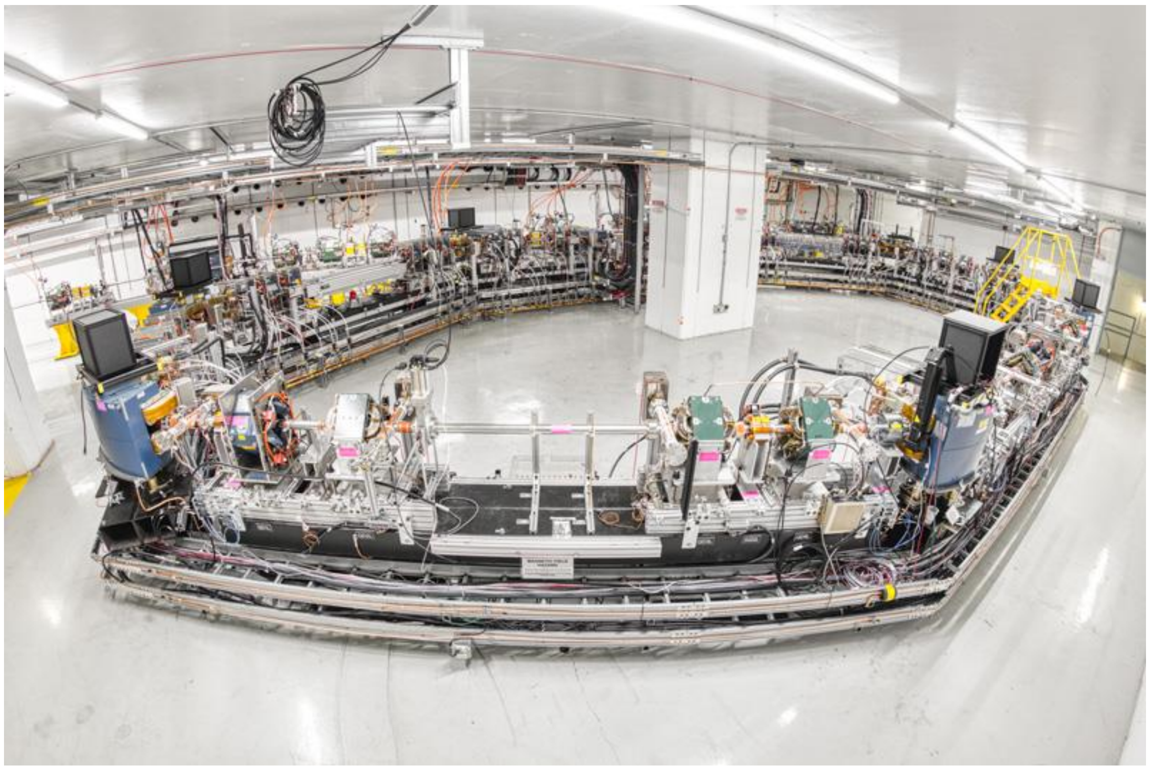


Figure 1: IOTA's ring

Electrons can be stored at energies of 100-150 MeV and the IOTA ring can be used to trap and store a single electron to study its emission properties. The main result was obtained during the first run in October 31, 2018. The scientists recorded a single electron traveling through the 40-meter-circumference IOTA ring. It circulated for four minutes, corresponding to about 2 billion turns. Figure 2 shows one of the photo-multiplier’s photon counting rates from an IOTA dipole magnet.

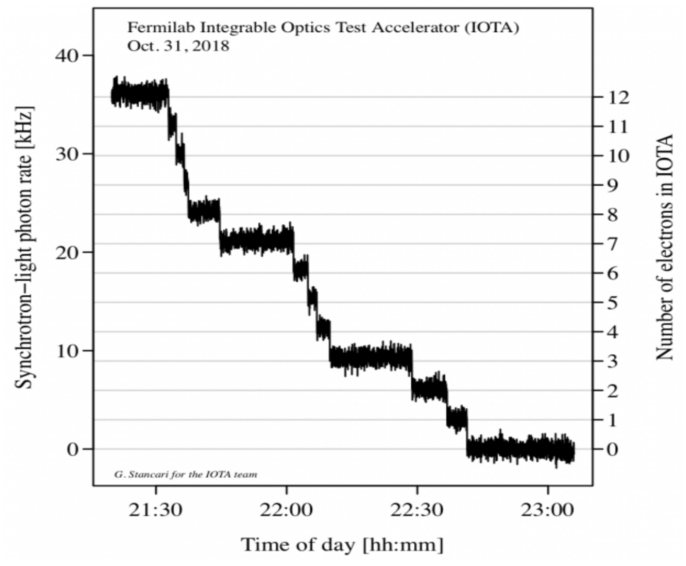


Figure 2: A measured photo-multiplier signal from a synchrotron radiation monitor after the bend magnet. One can clearly see finite jumps in the average proton count rate level as the number of trapped electrons becomes small, until a single electron is left in the IOTA storage ring.

In the previous plot you can see the Synchrotron-light photon rate in the left y-axis compared to the number of electrons in IOTA in the right y-axis function of time. Each step corresponds to an electron died due to collision with the residual gas in the vacuum chamber. This result paves the way for fundamental studies of the quantum properties of electrons, and of the photons they emit, as they are stored in the ring.

About my project

The SLAC undulator is essential for studies of synchrotron radiation produced by a single electron in IOTA. Since IOTA is meant for different experiments, the undulator is not always needed, so it must be pushed and pulled back depending upon the nature of the experiment. Moving the undulator manually result in significant time loss about 30 minutes for the controlled access, and about 30 minutes to achieve beam injector in IOTA. Because the operation is done manually is not easy to ensure a good repeatability of the undulator position. That's why IOTA's team decided to set-up a completely remote-controlled mount.

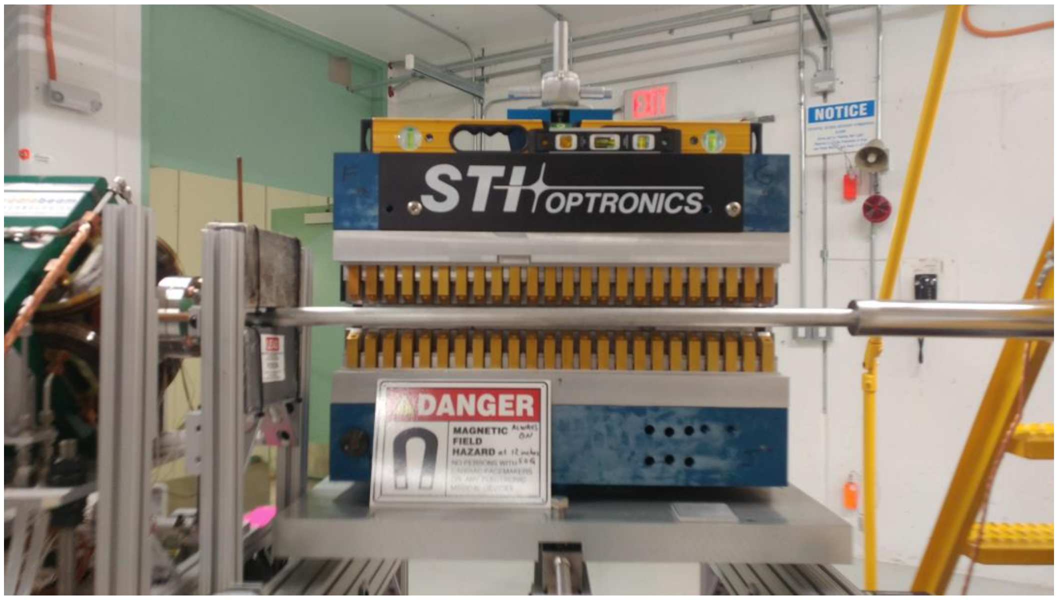


Figure 3: Undulator

The undulator is composed by a series of twenty-four permanent magnet with a special period of 55mm. The weight is about 763lb ≈ 364kg.

System specification

The main request is that the undulator must be finally aligned with the beam pipe with a resolution below 1mm, and the and the angle formed by the undulator with the pipe (yaw) must be equal to zero.

As first request is needed a very precise position sensor for the second one mechanical engineers' team must design new rails that keeps the undulator aligned with the pipe.

For safety reasons the undulator never has to touch the beam pipe, because it can break of eventually modify the geometry. That's why is needed a mechanical limit and micro-switches that can stop the motion once the undulator has reached the maximum displacement allowed.

The positioning system hat to be completely controlled from the main control room. The communication between the control system and the control room is done through ethernet network, using the MODBUS protocol, in order to easily integrate the system in the ACNET console.

System block diagram

The approach used to describe the system is top-down.

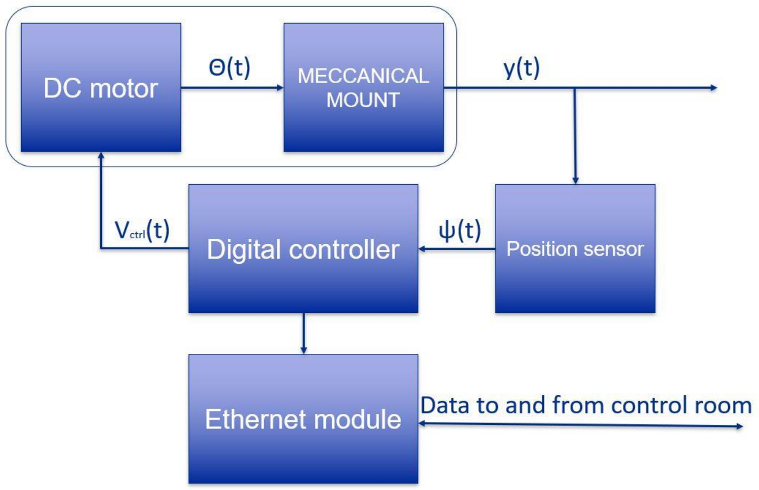


Figure 4: System block diagram

The positioning system is composed mainly by a DC motor that generates an angle displacement θ(t) function of the voltage applied to its armature. The angle displacement is the input for the mechanical mount that converts it in a linear displacement y(t). A position sensor reads and converts y(t) in a voltage signal ψ(t) that is used by the digital controller to generates the voltage V_{ctrl}(t) as input for the DC motor.
Digital controller, also, communicates real time, through an ethernet module, with the main control room. Each block of the previous figure corresponds to a physical object that will be described.

Electromechanical mount

To ensure the safety requirements, the most suitable and less expensive solution is a car jack. It is composed by a DC motor and a scissor jack. The scissor mechanism stops the motion once it is completely closed.

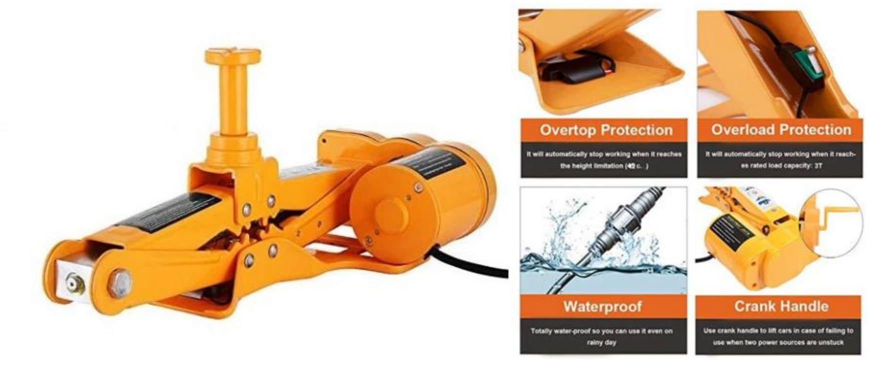

Overtop Protection
It will automatically stop working when it reaches the height limitation (42 c. .)
Overload Protection
It will automatically stop working when it reaches rated load capacity: 3T
Waterproof
Totally water-proof so you can use it even on rainy day
Crank Handle
Use crank handle to lift cars in case of failing to use when two power sources are unstuck

Figure 5: Car jack

The car jack found on amazon.com costs about 80$, can reach a maximum displacement of 42 cm and lift maximum 3ton. The supply needed is 12V and absorb 15A.

The displacement needed for the undulator is 25cm, and the force needed to move is less than its weight force F_w, actually the minimum needed is the static friction force which is $F_p = \mu_s F_w$ usually $\mu_s < 1$.

In case of motor failure, there is a crank handle that allows the specialized operator to move the undulator. If the maximum displacement is reached, the DC motor won't break because of the safety switch that will stop the current flow through the armature.

There is also an overload protection, but for this application is not necessary.

Position sensor: Linear potentiometer Novotechnik LWH0500

The position sensor for this application is a linear potentiometer. The sensibility of the potentiometer must be higher than 1mm, so the potentiometer must have at list 250 steps.

The potentiometer chosen has a resolution of 0.01mm, an intrinsic resistance of 5kΩ, IP54 protection class. It costs 353$, but the cost is justified by its specifications.

Figure 6: Linear potentiometer

Digital controller: Arduino Mega 2560 rev3

Figure 7: Arduino Mega 2560 board top and bottom view

The **Arduino Mega 2560** is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs connected to a 10 bits ADC converter, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery.

The Mega 2560 board is compatible with most shields designed for the Uno. The cost is 39$. A simple microcontroller would be cheaper, but it requires a lot of time to program and debug. Arduino's programming language is user friendly and the community provides many open source libraries to manage DC motors, PID controller and provides a more flexible and general solution.

ADC converter considerations

The resolution Q of the ADC is equal to the LSB voltage. The voltage resolution of an ADC is equal to its overall voltage measurement range divided by the number of intervals:

$$Q = \frac{V_{FSR}}{2^N}$$

Considering V_{FSR}= 5V and N=10 bits, Arduino's ADC has a resolution Q= 4,8 mV. Since the maximum extension of the potentiometer is 250 mm, and the specification requires at list 1 mm of sensibility. The value chosen for the application is 0,1mm. In the voltage range 0-5 V there are *2500 steps*, that means a minimum resolution Q=2mV. **Arduino's ADC is not suitable for this application.**

ADS1115 ADC ΣΔ converter

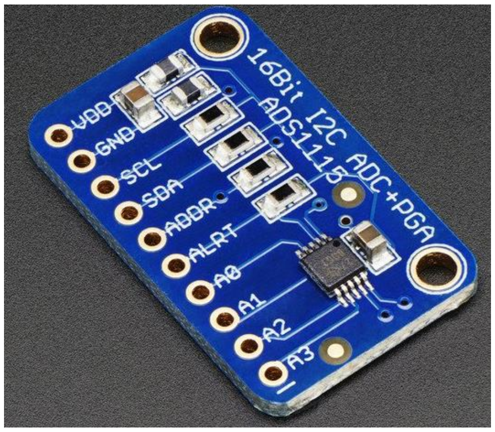


Figure 8: ADC board

The ADS1115 provides 16-bit precision at 860 samples/second over I2C. The chip can be configured as 4 single-ended input channels, or two differential channels. As a nice bonus, it even includes a programmable gain amplifier, up to x16, to help boost up smaller single/differential signals to the full range. We like this ADC because it can run from 2V to 5V power/logic, can measure a large range of signals and its super easy to use. It is a great general purpose 16-bit converter. The chip is small, so it comes on a breakout board with ferrites to keep the AVDD and AGND quiet. Interfacing is done via I2C protocol. The address can be changed to one of four options so you can have up to 4 ADS1115's connected on a single 2-wire I2C bus for 16 single ended inputs.

This ADC has a resolution of Q=76,29uV (<< 2mV), that means the minimum displacement measured is 0,0381 mm.

Ethernet module: Arduino ethernet shield 2

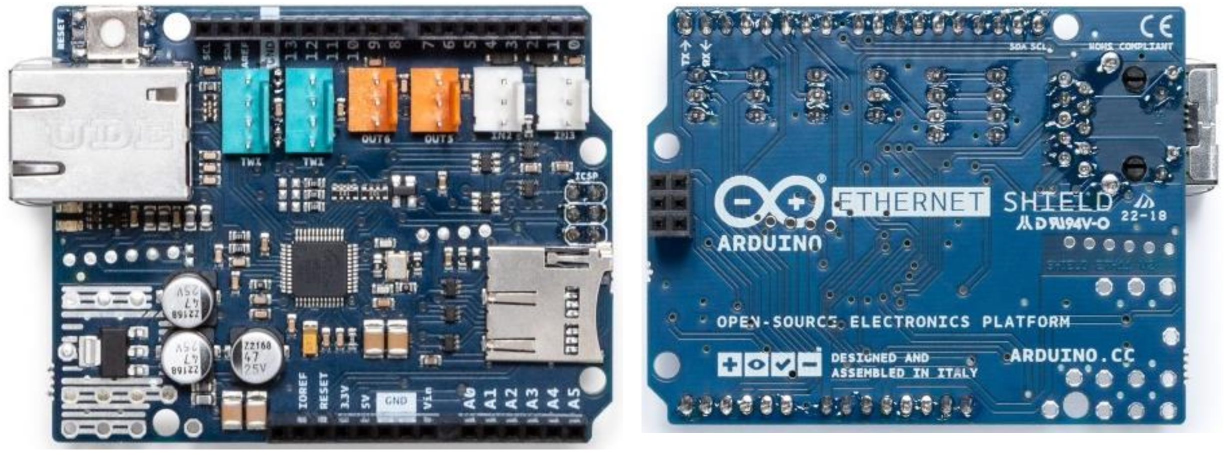

ETHERNET SHIELD
ARDUINO
OPEN-SOURCE ELECTRONICS PLATFORM
DESIGNED AND ASSEMBLED IN ITALY
ARDUINO.CC

Figure 9: Ethernet shield 2 top and bottom view

The Arduino Ethernet Shield 2 allows an Arduino Board to connect to the internet. It is based on the Wiznet W5500 Ethernet chip. The Wiznet W5500 provides a network (IP) stack capable of both TCP and UDP. It supports up to eight simultaneous socket connections. Use the Ethernet library to write sketches that connect to the Internet using the Shield. The Ethernet Shield 2 connects to an Arduino Board using long wire-wrap headers extending through the Shield. This keeps the pin layout intact and allows another Shield to be stacked on top of it. The Ethernet Shield 2 has a standard RJ-45 connection, with an integrated line transformer and Power over Ethernet enabled. There is an onboard micro-SD card slot, which can be used to store files for serving over the network. It is compatible with the Arduino Uno and Mega (using the Ethernet library). The onboard micro-SD card reader is accessible through the SD Library. The Shield also includes a reset controller, to ensure that the W5500 Ethernet module is properly reset on power-up. The current Shield supports a Power over Ethernet (PoE) module designed to extract power from a conventional twisted pair Category 5 Ethernet cable.

Analytical model of the electro-mechanical jack

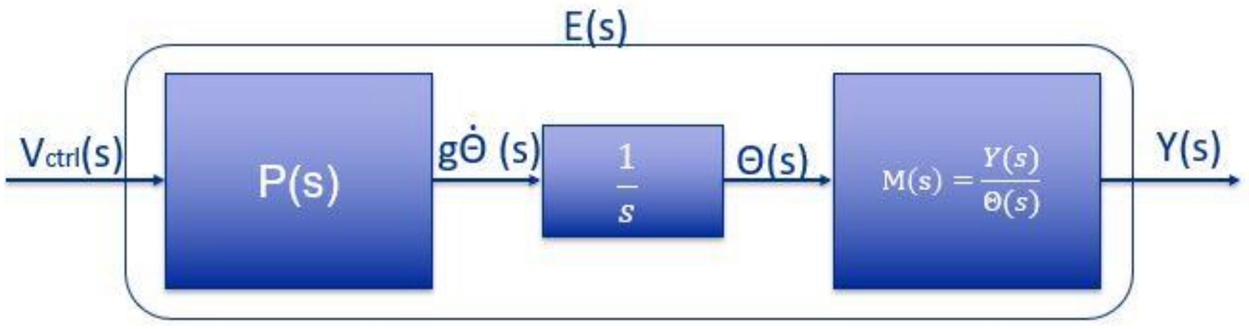


Figure 10:Block diagram of the electro-mechanical mount

The transfer function that describes the relationship between the y displacement and the voltage applied to the DC motor is

$$E(s) = \frac{Y(s)}{V_{ctrl}(s)} = \frac{gP(s)M(s)}{s}$$

P(s) is the transfer function of the DC motor, a relation between the voltage applied to the armature and the angular speed $\dot{\Theta}$ (t), after an integration (1/s in Laplace domain) is possible to obtain the angular position θ (t). The last transfer function describes the y displacement function of the angular position. The DC motor has also a gearbox, the gain of the gearbox is called g, it allows to increase the torque force, decreasing the angular speed.

DC motor model

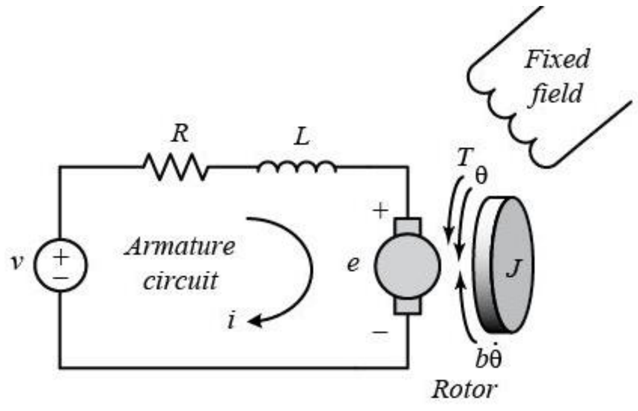


Figure 11: Physical model of the DC motor

The main parameters of the motor are:

- J: moment of inertia of the rotor
- b: motor viscous friction constant
- Ke: electromotive force constant
- Kt: motor torque constant
- R: electric resistance of the armature
- L: electric inductance of the armature
- e: back emf
- T: torque force

The torque force is proportional to the current flowing in the armature of the motor $T = K_t i$, the back emf is proportional to the angular speed $e = K_e \dot{\theta}$. In SI units the motor torque and emf constant are equal, K_t= K_e= K. The governing equations based on Newton's second law and Kirchhoff's law are:

$$J\ddot{\theta} + b\dot{\theta} = Ki$$

$$L\frac{di}{dt} + Ri = V - K\dot{\theta}$$

After applying the Laplace transform:

$$s(Js + b)\Theta(s) = KI(s)$$

$$(Ls + R)I(s) = V(s) - K\ \Theta(s)$$

Combining the two equations it's possible to write in a closed form the TF of the DC motor.

$$\boldsymbol{P(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{K}{(Js + b)(Ls + R) + K^2}}$$

Jack model

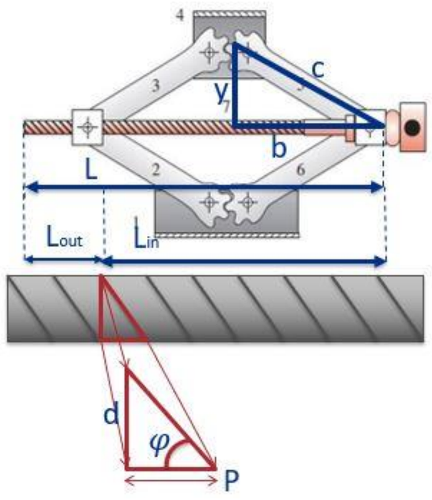


Figure 12: Car jack model

The main parameters are:

- C: lift arm
- y: half height
- b: half length of the inside threated bar
- L: length of the threated bar
- Lin: inside length of the threated bar
- Lout: outside length of the threated bar
- D: diameter of the threated bar
- P: pitch of the thread
- φ:pitch angle of the thread
- n: number of revolutions

Applying the Pythagorean theorem at the triangle y-c-b:

$$y^2 = c^2 - b^2 = c^2 - \frac{1}{4}\left(L - n2\pi\frac{d}{\tan(\varphi)}\right)^2$$

$$n = \frac{\theta(t)}{2\pi} \quad P = \frac{d}{\tan(\varphi)}$$

$$y^2 = c^2 - \frac{1}{4}(L - \theta(t)P)^2 = c^2 - \frac{1}{4}L^2 + \frac{1}{4}[\theta(t)P]^2 + \frac{1}{2}L\,\theta(t)\mathrm{P} \approx c^2 - \frac{1}{4}L^2 + \frac{1}{2}L\,\theta(t)\mathrm{P}$$

$$\boldsymbol{y(t) = \sqrt{c^2 - \frac{1}{4}L^2 + \frac{1}{2}L\,\theta(t)P}}$$

The previous equation is not linear. The car jack is not moldable as an LTI system, so the entire system E(s) is not LTI (**Linear** Time-Invariant), a classic PID controller won't guarantee the stability and the desired response.

PID basic concepts

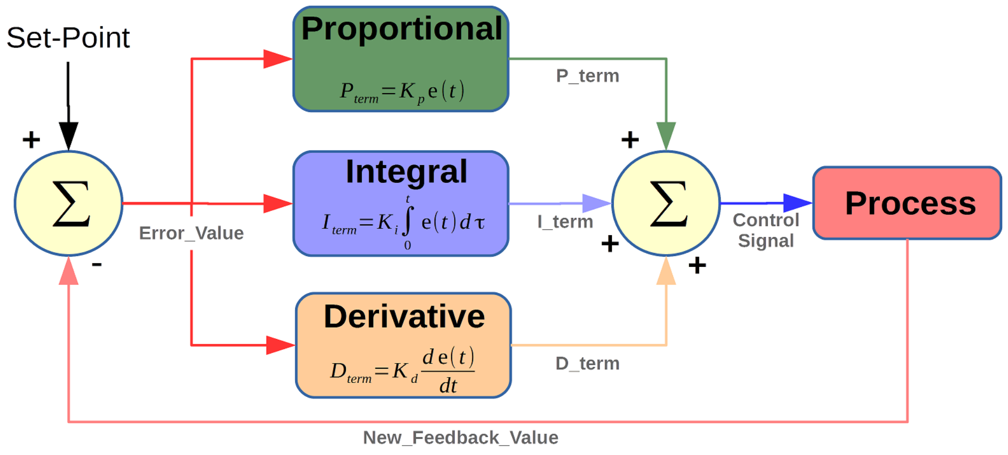


Figure 13: PID block diagram

The distinguishing feature of the PID controller is the ability to use the three control terms of proportional, integral and derivative influence on the controller output to apply accurate and optimal control. The block diagram shows the principles of how these terms are generated and applied. It shows a PID controller, which continuously calculates an error value e(t) as the difference between a desired setpoint SP and a measured process variable PV, and applies a correction based on proportional, integral, and derivative terms. The controller attempts to minimize the error over time by adjustment of a control variable to a new value determined by a weighted sum of the control terms.

In this model:

- Term P is proportional to the current value of the SP − PV error e(t). For example, if the error is large and positive, the control output will be proportionately large and positive, considering the gain factor "K". Using proportional control alone will result in an error between the setpoint and the actual process value, because it requires an error to generate the proportional response. If there is no error, there is no corrective response.
- Term I accounts for past values of the SP − PV error and integrates them over time to produce the I term. For example, if there is a residual SP − PV error after the application of proportional

control, the integral term seeks to eliminate the residual error by adding a control effect due to the historic cumulative value of the error. When the error is eliminated, the integral term will cease to grow. This will result in the proportional effect diminishing as the error decreases, but this is compensated for by the growing integral effect.

- Term D is a best estimate of the future trend of the SP − PV error, based on its current rate of change. It is sometimes called "anticipatory control", as it is effectively seeking to reduce the effect of the SP − PV error by exerting a control influence generated by the rate of error change. The more rapid the change, the greater the controlling or dampening effect.

Because the system is not linear the best approach for the system control is the Gain Scheduling

Gain scheduling PID

Gain scheduling is a PID enhancement that facilitates the control of a process with gains and time constants that vary according to the current value of the process variable. A gain scheduler runs in the controller’s microprocessor and monitors the process variable to determine when the process has entered a new operating range. It then updates the controller with a predetermined set of tuning parameters designed to optimize the closed-loop performance in that range. Gain scheduling is particularly appropriate for processes that speed up or slow down as the process variable rises and falls. It also works if the process becomes sensitive to the controller’s efforts as the process variable changes. A gain scheduler provides the best of both worlds. It allows the controller to be tuned for any number of operating ranges so that an optimal set of tuning parameters can be downloaded into the controller depending on the current value of the process variable. Unfortunately, that’s a lot of work, if done by hand. A special PID library available for Arduino provide an automatic PID tuner. The control engineer implementing the gain schedule must first determine how the full span of the process variable should be partitioned into distinct operating ranges that adequately represent all the possible variations in the process’s behavior. The implementer would then have to operate the process within each range and tune the controller for optimal closed-loop performance each time, then load the resulting sets of tuning parameters into the gain schedule to be retrieved by the controller whenever the process variable enters the operating range that corresponds to each set.

DAQ system

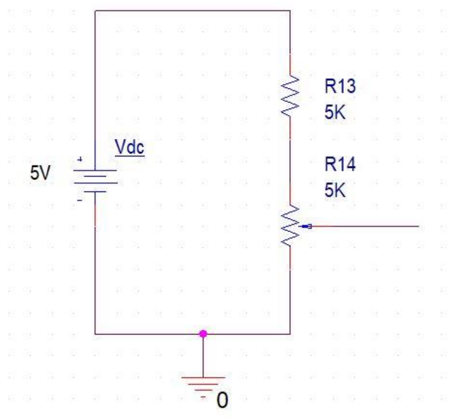


Figure 14: Linear potentiometer

The position transducer converts the y(t) position in an analog voltage signal Ψ(t). In order to perform a good acquisition, the signal must be amplified and filtered. The amplification is needed because the dynamic of the signal is between 0 and 2.5 V and ADC dynamics is 0-5V, the gain is Av=2. The filtering process is necessary in order to cut-off 60Hz, parasitic coupled signal and noise.

Amplifier

The most suitable architecture for this application is the instrumental amplifier. The IA is a type of differential amplifier that has been outfitted with input buffer amplifiers, which eliminate the need for input impedance matching and thus make the amplifier particularly suitable for use in measurement and test equipment. Additional characteristics include very low DC offset, low drift, low noise, very high open-loop gain, very high common-mode rejection ratio, and very high input impedances. Instrumentation amplifiers are used where great accuracy and stability of the circuit both short and long-term are required.

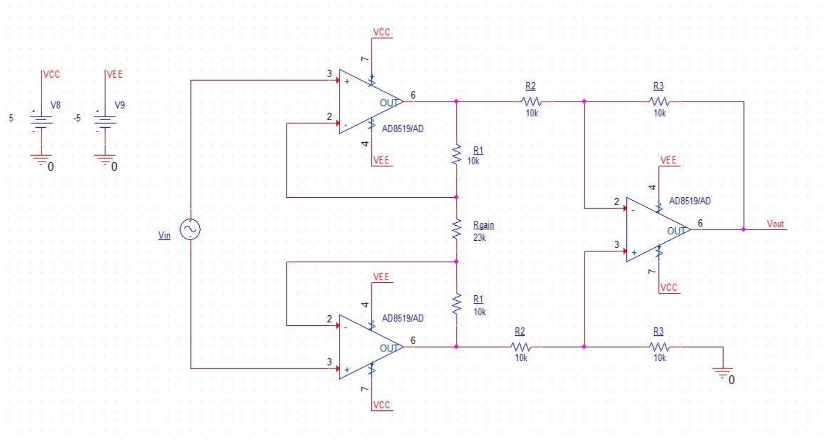


Figure 15: Instrumental amplifier schematic

The gain of the amplifier, is equal to:

$$A_v = \frac{V_{out}}{V_{in}} = \left(1 + \frac{2R_1}{R_{gain}}\right)\frac{R_3}{R_2} = 2$$

Low pass filter

The specifications of the filter are:

- Gain: 0dB
- $f_{c(-3dB)}$ =10 Hz
- $A_{60Hz} = -40dB$

The order of the filter is three, a cascade of a second order Sallen-Key low pass filter and a buffered RC filter.

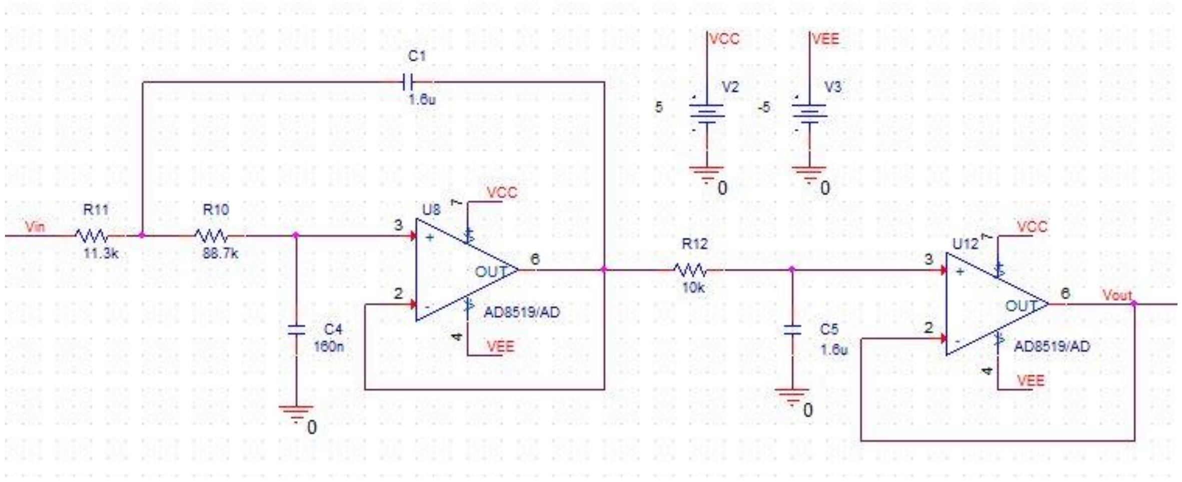


Figure 16: Third order LP filter schematic

The frequency response of the filter is:

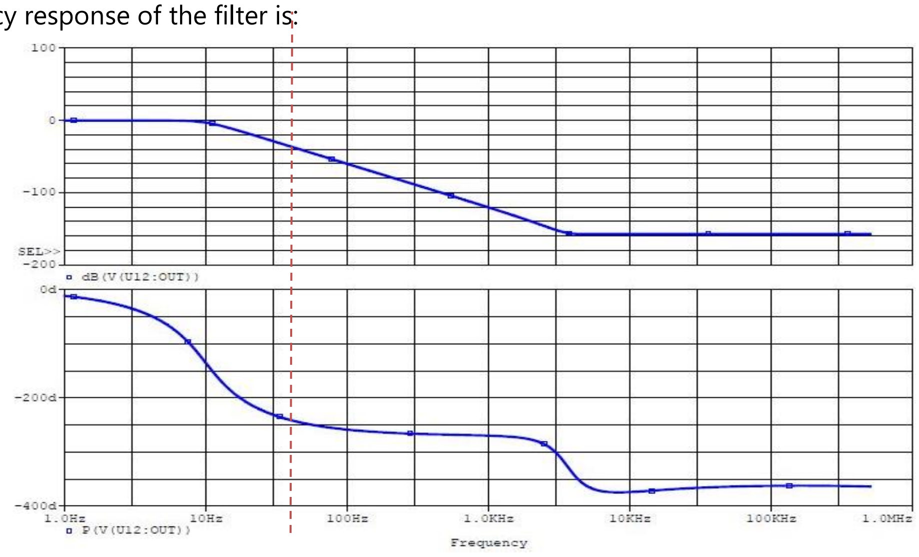


Figure 17: Bode diagram of the filter

The filter fits perfectly the specification

DAQ circuit - final simulations

After putting together the two parts the schematic is the following:

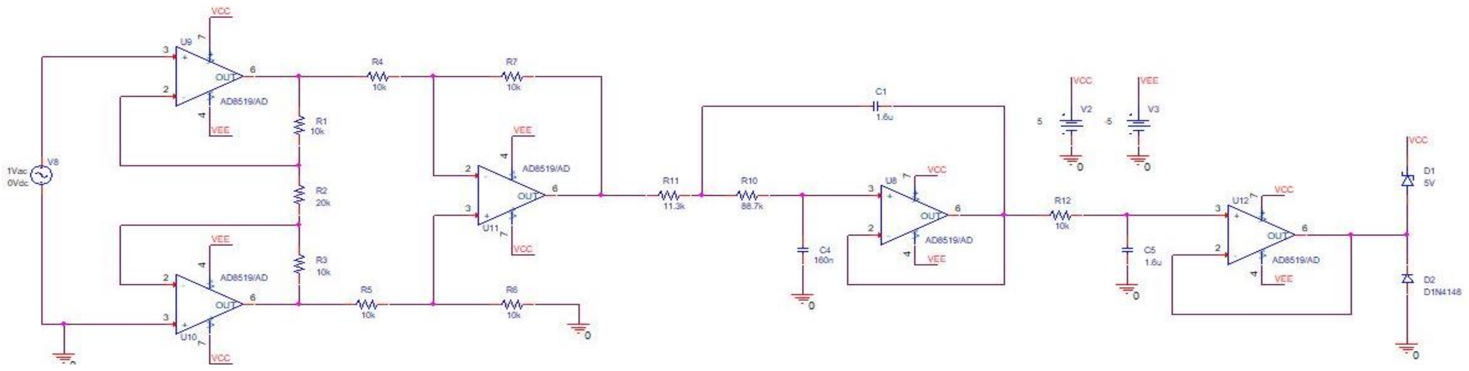


Figure 18: DAQ schematic - single channel

The final two diodes, one Schottky, one classic PN diode, serves as protection for the ADC converter. If for some reasons the voltage increases over Vcc+Vs, where Vs is the direct voltage bias of a Schottky diode, it will be clamped at Vcc+Vs. If the voltage would decrease below -Vd, it will be clamped at -Vd.

Because the signal read from the linear potentiometer change very slow in time, it is necessary to have a DC simulation of the circuit, to better understand the behavior of the circuit and the limits of the input and output dynamics.

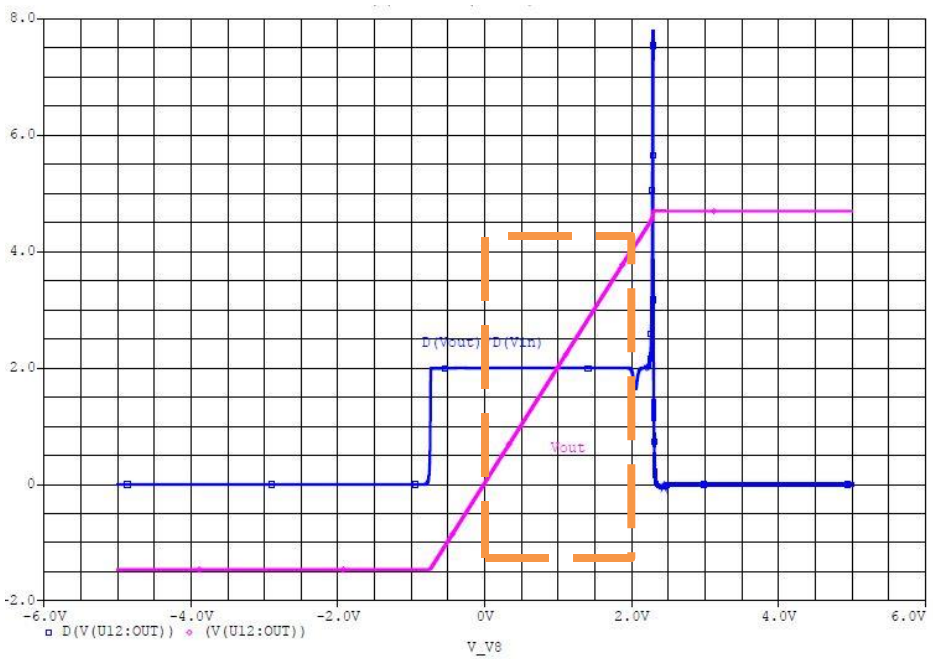


Figure 19: DAQ DC simulation

The previous plot shows two curves. The pink one is the relation between input and output, the blue one is the first derivative of the pink. In the range 0-2,5 V the pink curve shows a liner relation, the pendency of is the gain. If derived, it's possible to notice that the real linear range is 0-2V (orange) and the gain is constant and equal to 2.

The frequency response of the entire circuit is:

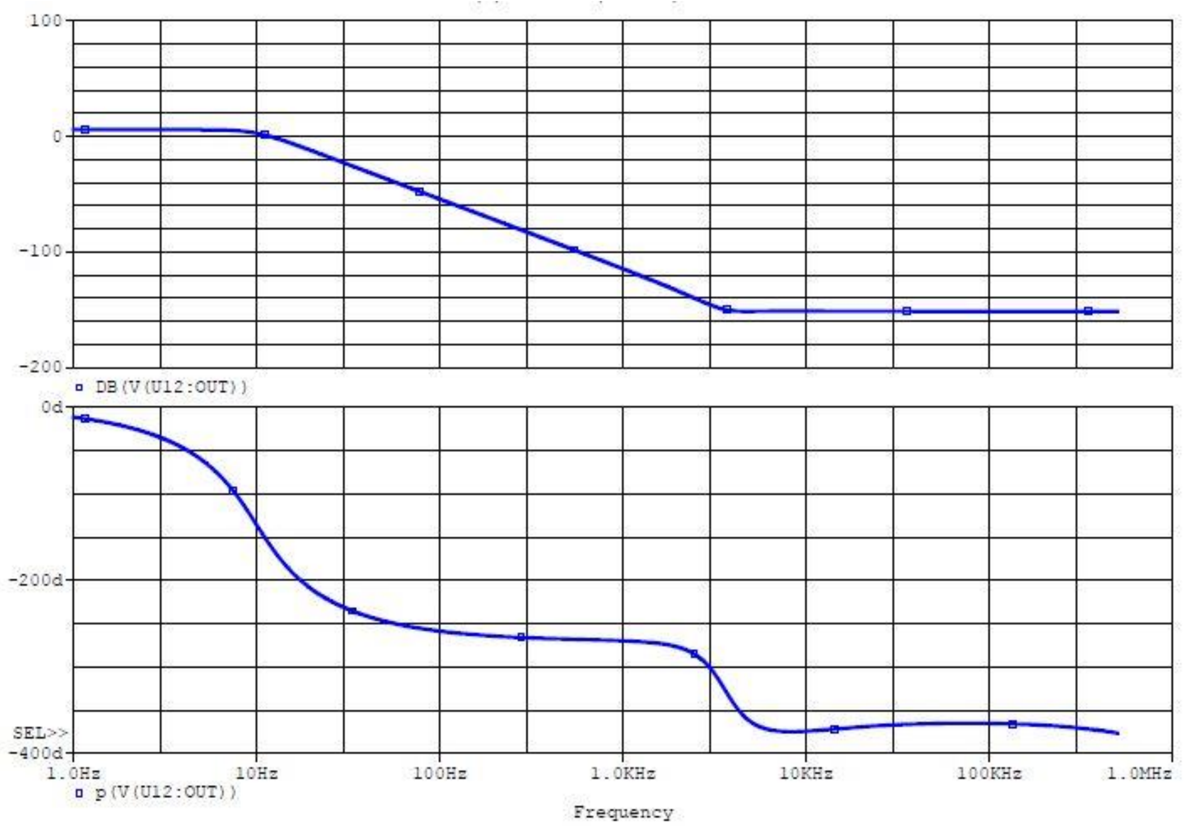


Figure 20: Frequency response of the DAQ

The frequency response is like the previous, but the amplitude is shifted by the gain. Because there are two linear potentiometers, the DAQ must have 2 similar channels.

DAQ schematic and PCB

The schematic is attached at the end of the report.

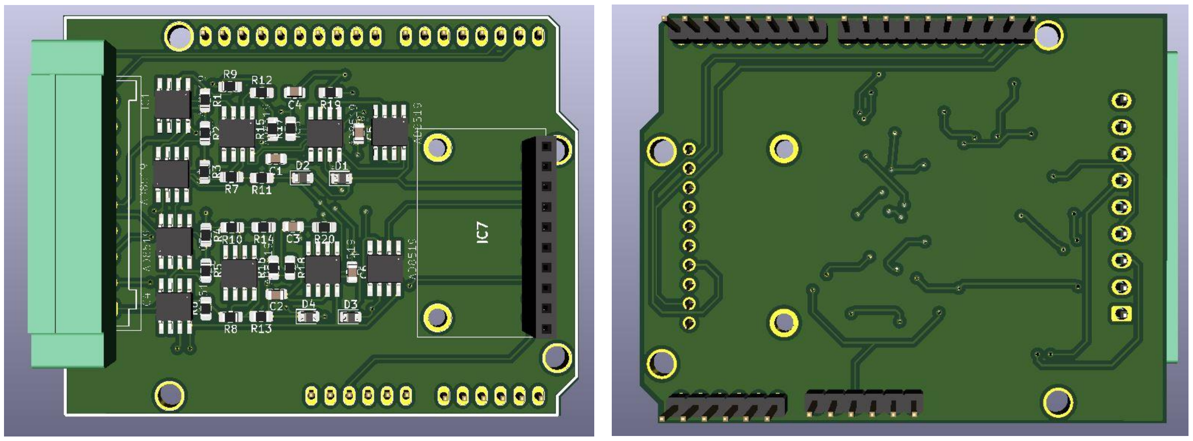
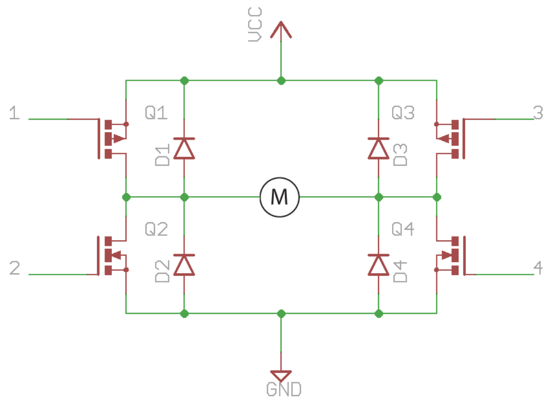


Figure 21: DAQ PCB - top and bottom view

The shape of the PCB is compatible with Arduino Uno, that means it's possible to put it on top of the Arduino Mega, it's what the community use to call a Shield. The black connector is used to connect the ADC to the board.

DC motor driver

The DC motor direction spin and speed are controlled trough an H bridge, that can provide a maximum current of 15 A



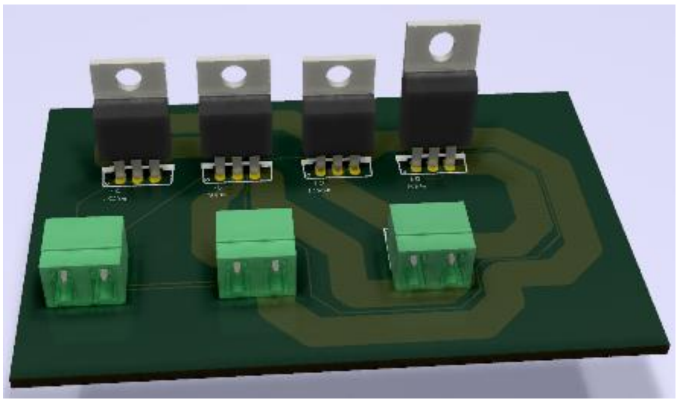


Figure 22: H bridge

The H-bridge arrangement is generally used to reverse the polarity/direction of the motor, but can also be used to 'brake' the motor, where the motor comes to a sudden stop, as the motor's terminals are shorted, or to let the motor 'free run' to a stop, as the motor is effectively disconnected from the circuit.

Arduino sw

This is the code wrote for Arduino. It should be improved and modified. This code is for a single channel DAQ, it must be modified in order to include a second channel.

```
#include <Wire.h> //library needed for I2C comunication
#include <Adafruit_ADS1015.h> //ADC library
#include <DC_Motor.h> // DC motor library
#include <PID_v1.h>
#include <SPI.h>
#include <Ethernet.h>

#define MotFwd 2 //forward direction
#define MotBwd 3 //backward direction
#define IntSwS  8 // interrupt ()
#define IntSwE  7//interrupt()

double Setpoint=0, Input=0, Output=0; //Define Variables we'll be connecting to
float pos_val; //position value
float alpha = 1; //proportional constant betwen the ADC_val read and the position
float pos=0;
int ADC_val;
const byte pin1=8;
const byte pin2=9;

DC_Motor motor(MotFwd,MotBwd); //DC motor connector(H bridge input)

Adafruit_ADS1115 ads1115(0x48); //assign the addres at the ADC

PID myPID(&Input, &Output, &Setpoint, 2, 5, 1, DIRECT); //Specify the links and initial tuning parameters

byte mac[] = {0xA8, 0x61, 0x0A, 0xAE, 0x5D, 0x03 };
IPAddress ip(192, 168, 1, 177);
IPAddress myDns(192, 168, 1, 1);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 0, 0);
```

```
EthernetServer server(23);
bool alreadyConnected = false;

void setup()
{
  pinMode(MotFwd, OUTPUT);
  pinMode(MotBwd, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(pin1),Safe1,RISING);
  attachInterrupt(digitalPinToInterrupt(pin2),Safe2,FALLING);
  Serial.begin(9600);
  Serial.print("Hello!");
  Serial.println("I'm getting differential reading from AIN0 (P) and AIN1 (N)");
  Serial.println("ADC Range: +/- 5V (1bit = 152.5uV");
  Serial.println("please insert the desired position");
  Ethernet.begin(mac, ip, myDns, gateway, subnet); // initialize the ethernet device
  if (Ethernet.hardwareStatus() == EthernetNoHardware) {
    Serial.println("Ethernet shield was not found.  Sorry, can't run without hardware. :(");
    while (true) {
      delay(1); // do nothing, no point running without Ethernet hardware
    }
  }
  if (Ethernet.linkStatus() == LinkOFF) {
    Serial.println("Ethernet cable is not connected.");
  }
  // start listening for clients
  server.begin();

  Serial.print("Chat server address:");
  Serial.println(Ethernet.localIP());

  ads1115.begin(); //intialize ads115
  //turn the PID on
  myPID.SetMode(AUTOMATIC);
   myPID.SetSampleTime(1);//refresh rate of pid controller
   myPID.SetOutputLimits(-125,125);//set the max pwm to move the motor
```

```
}

void loop()
{

  // wait for a new client:
  EthernetClient client = server.available();

  // when the client sends the first byte, say hello:
  if (client) {
    if (!alreadyConnected) {
      // clear out the input buffer:
      client.flush();
      Serial.println("We have a new client");
      client.println("Hello, client!");
      alreadyConnected = true;
    }

    if (client.available() > 0) {
      // read the bytes incoming from the client:
      char SetPoint = client.read();
      Setpoint = double(SetPoint);
    }
  }
    int16_t ADC_val; //16 bit integer local variable
    ADC_val=ads1115.readADC_Differential_0_1();// ads1115.readADC_Differential_0_1(); //ADC value
reading
    Serial.println(ADC_val);
    pos_val = alpha * float(ADC_val); //COMPUTATION AND FILTERING SECTION
    Input = double(pos_val);
    myPID.Compute();
    pos=map(Output,-125, 125, 0, 125);
    if (Output > 0.1) {
      digitalWrite(MotFwd,pos);
    }
    else if (Output < -0.1) {
      digitalWrite(MotBwd,pos);
    }
    else{
      digitalWrite(7,HIGH);
    }
}

  void Safe1(){
    digitalWrite(MotBwd,1);
    digitalWrite(MotFwd,0);
  }
  void Safe2(){
    digitalWrite(MotBwd,0);
    digitalWrite(MotFwd,0);
  }
```