Instituto Tecnológico de Buenos Aires

22.85 Sistemas de Control

Trabajo Práctico Final

Rodriguez Turco, Martín	56629
Mechoulam, Alan	58438
Lambertucci, Guido Enrique	58009
Londero Bonaparte, Tomás Guillermo	58150

Profesores
Nasini, Victor Gustavo
Zujew, Cristian Alejo

Presentado: 23/02/21

Índice

1. Introducción

En el siguiente informe se presenta el análisis y desarrollo de un péndulo el cual se estabiliza a 45^o por encima de la recta horizontal. Este es accionado por un sistema de motor de continua más hélice. La retroalimentación se realiza por medio de un acelerómetro montado cerca del pivote del péndulo.

Cabe aclarar que al momento de realizar el trabajo, se plantearon ciertas condiciones, a modo de objetivo, entre las cuales se propuso utilizar elementos que se encontraran al alcance del grupo. Para ello se empleó un motor y hélice de un dron, un Arduino Nano, un acelerómetro MPU6050 y un modulo de puente H L298N.

2. Modelado Físico

Para el modelado físico se toma en cuenta al ventilador como un objeto puntual, el cual se puede mover únicamente en el versor tangencial dado que el mismo palo restringe el movimiento en los otros versores.

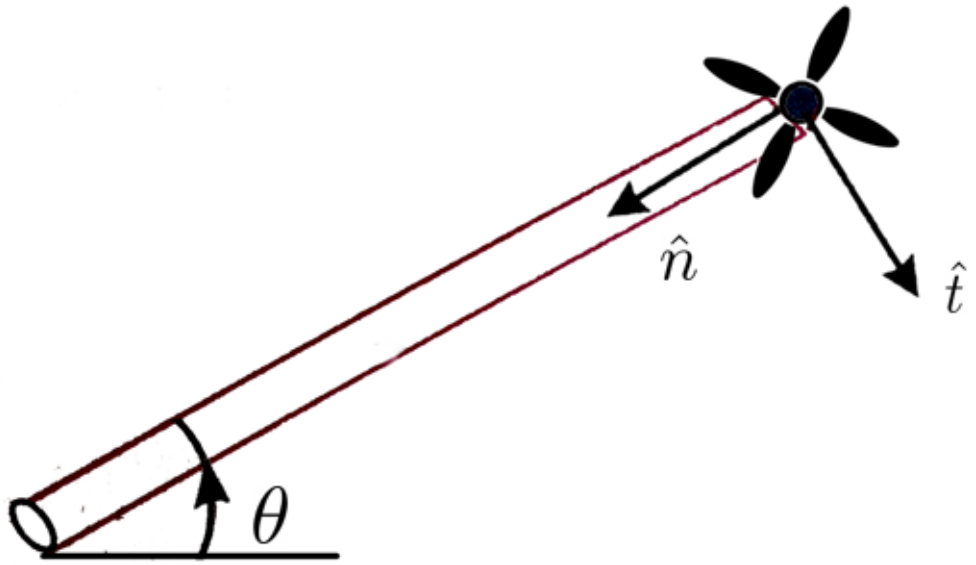

Figura 1: Diagrama de la planta.

Se plantea el versor tangencial:

$$\hat{t} : mg \cdot \cos(\theta) = \frac{\partial P}{\partial t} \tag{1}$$

$$mg \cdot \cos(\theta) = v \cdot \frac{\partial m}{\partial t} + m\frac{\partial v}{\partial t} \tag{2}$$

$$mg \cdot \cos(\theta) - v \cdot \frac{\partial m}{\partial t} = m \cdot a \tag{3}$$

$$mg \cdot \cos(\theta) - (v_t - v_f)^2 \cdot \frac{\rho C_d A}{2} = m \cdot a \tag{4}$$

Donde ρ es la densidad del fluido, en este caso del aire, C_d es el coeficiente de arrastre, A es el área por la cual atraviesa el flujo de aire, v_t es la velocidad tangencial del ventilador respecto de la mesa, y v_f es la velocidad del aire expulsado gracias la rotación de las aspas.
Además sabiendo las siguientes relaciones:

$$v_t = R \cdot \omega \qquad v_f = R_f \cdot \omega_f(t) \qquad \omega \cdot R = \dot{\theta} \qquad \alpha \cdot R = \ddot{\theta} \tag{5}$$

Donde R es el radio desde el pivote al ventilador, y R_f el radio de las aspas, ω_f es la función de giro del ventilador.

$$mg \cdot \cos(\theta) - \left(R \cdot \dot{\theta} - R_f \cdot \omega_f\right)^2 \cdot \frac{\rho C_d A}{2} = mR \cdot \ddot{\theta} \tag{6}$$

Aquí se llega a una expresión del sistema, en el cual debido a la elección de actuador y sensor, se obtiene ω_f el cual será modificada por el actuador, y θ el cual es medido por el sensor.
Si se toman las siguientes aproximaciones:

$$\left(R \cdot \dot{\theta} - R_f \cdot \omega_f\right)^2 \approx R \cdot \dot{\theta} - R_f \cdot \omega_f \qquad \cos(\theta) \approx \frac{\sqrt{2}}{2} \tag{7}$$

se puede aproximar de la siguiente manera.

$$\overset{\mathcal{L}}{\approx}$$

$$mg\frac{\sqrt{2}}{2}\delta(s) - (R \cdot s\theta(s) - R_f\omega_f(s)) \cdot \frac{\rho C_d A}{2} = m \cdot s^2\theta(s) \tag{8}$$

Aqui se puede apreciar que se puede estimar una transferencia lineal si se ignora el término gravitatorio, el cual tendrá 2 polos. Donde θ es la salida y ω_f es la entrada

3. Implementación

La arquitectura de control utilizada se observa en la Figura (2).

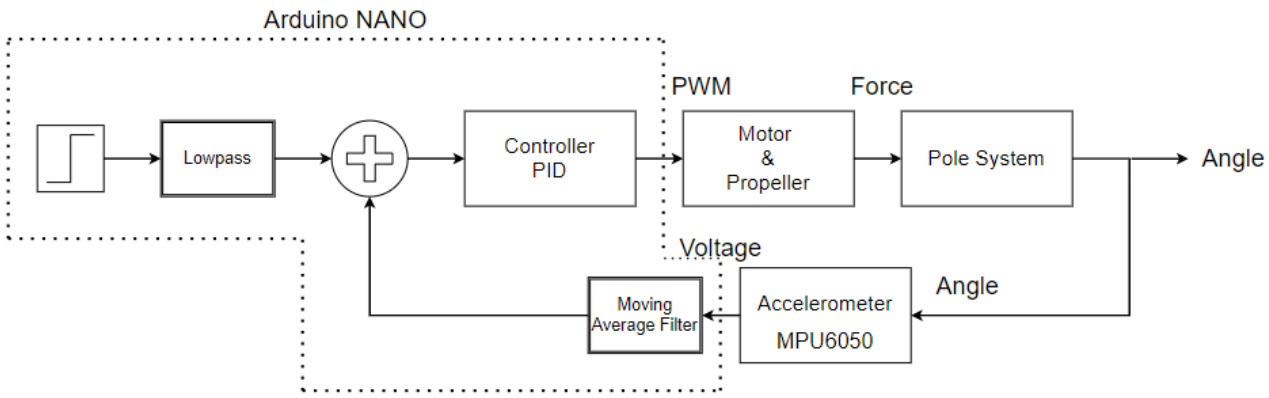

Figura 2: Diagrama del sistema de control.

Mientras que el hardware se conectó según la Figura (3).

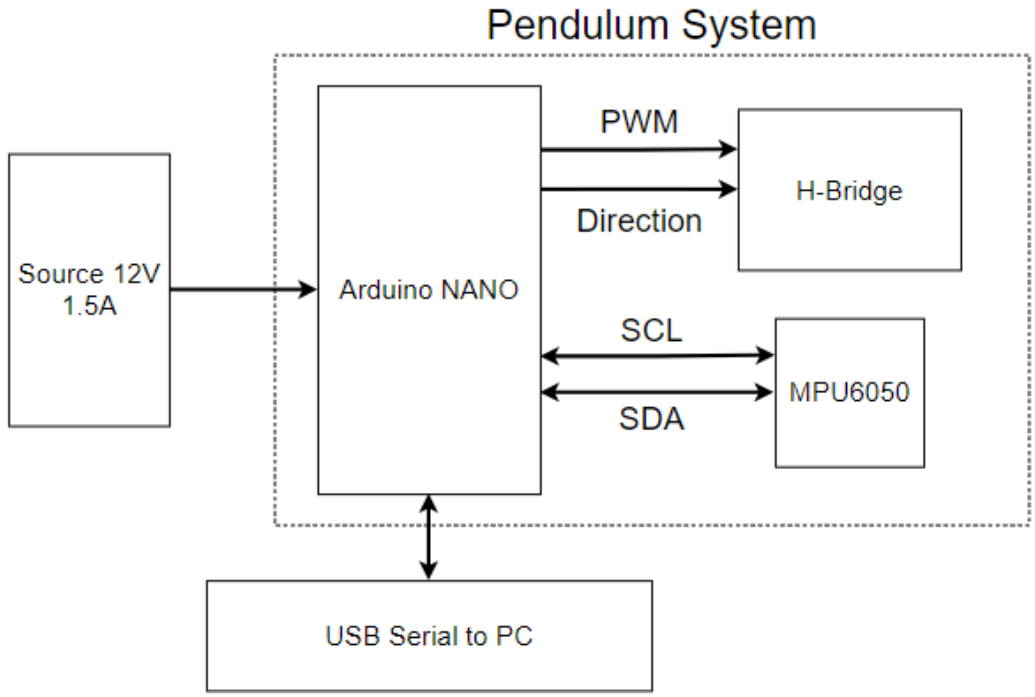

Figura 3: Diagrama del conexionado de hardware.

Como punto de partida, se midió la respuesta del sistema a lazo cerrado con baja ganancia, observando que este se estabilizaba en cierto punto, pero con un error permanente grande.

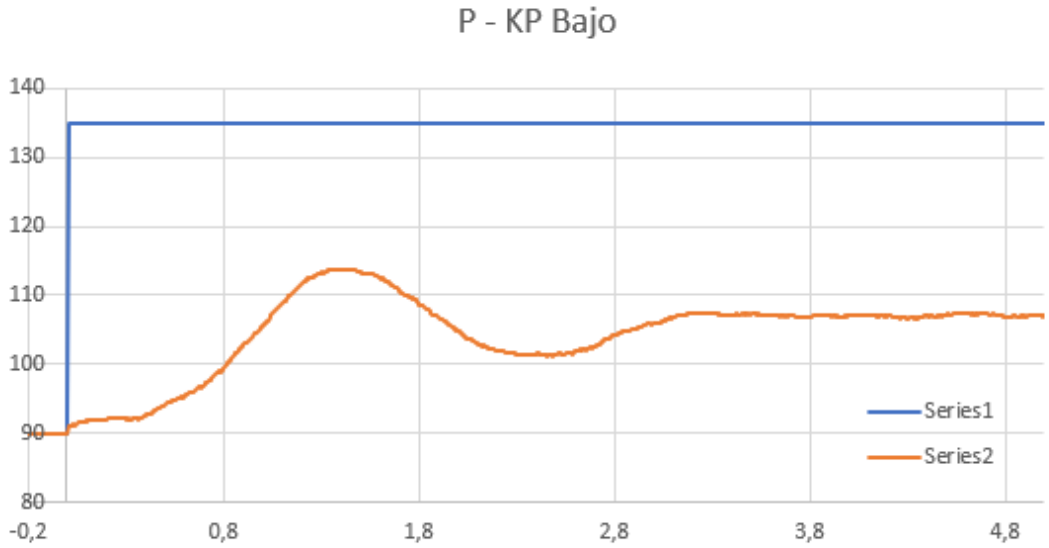

Figura 4: Respuesta al escalón del controlador proporcional. Setpoint en curva azul y ángulo en curva naranja.

Si bien este sistema se estabilizaba frente a una respuesta al escalón, frente a un disturbio se tornaba inestable como se observa en la Figura (5).

Figura 5: Respuesta a disturbio del controlador proporcional. Setpoint en curva azul y ángulo en curva naranja.

Por esto, se agregó control derivativo para estabilizar al sistema frente disturbios. Esto exageró significativamente el ruido del acelerómetro debido a que las vibraciones inducidas por el motor producían ruido que tras pasar por el derivador provocaban más vibraciones en el motor. Para reducir este efecto, se utilizó un filtro de media móvil. Con esto, el sistema se tornó estable frente a disturbios.

Figura 6: Respuesta a disturbio del controlador proporcional derivativo. Setpoint en curva azul y ángulo en curva naranja.

Sin embargo, el sistema aún poseía gran error permanente. Se agregó control integral con anti-windup para mitigar el error permanente quedando así la respuesta como se observa en la Figura (7).

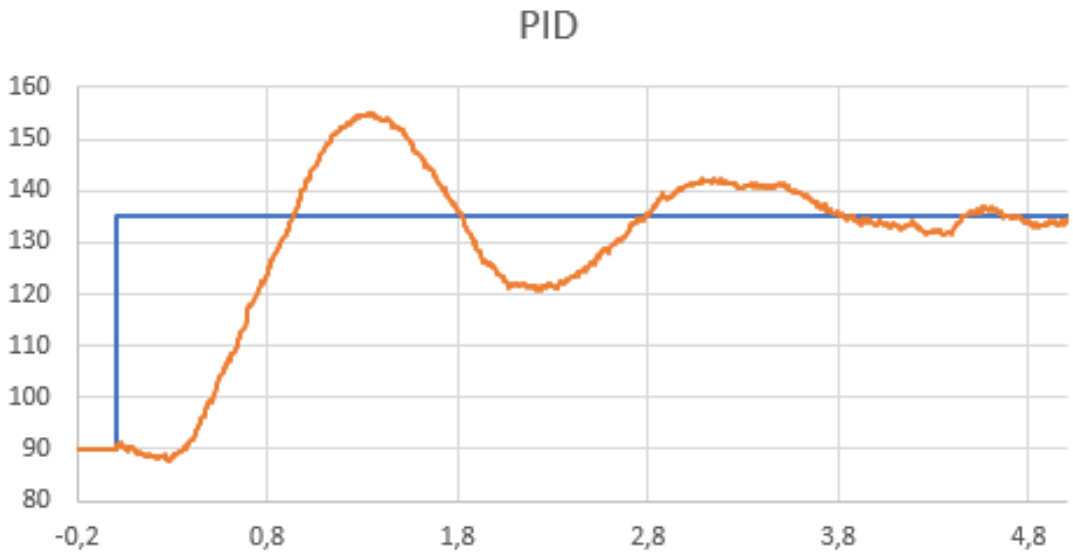

Figura 7: Respuesta al escalón del controlador PID. Setpoint en curva azul y ángulo en curva naranja.

Esta respuesta era demasiado oscilatoria, por lo que se colocó un prefiltro lo cual arregló este problema.

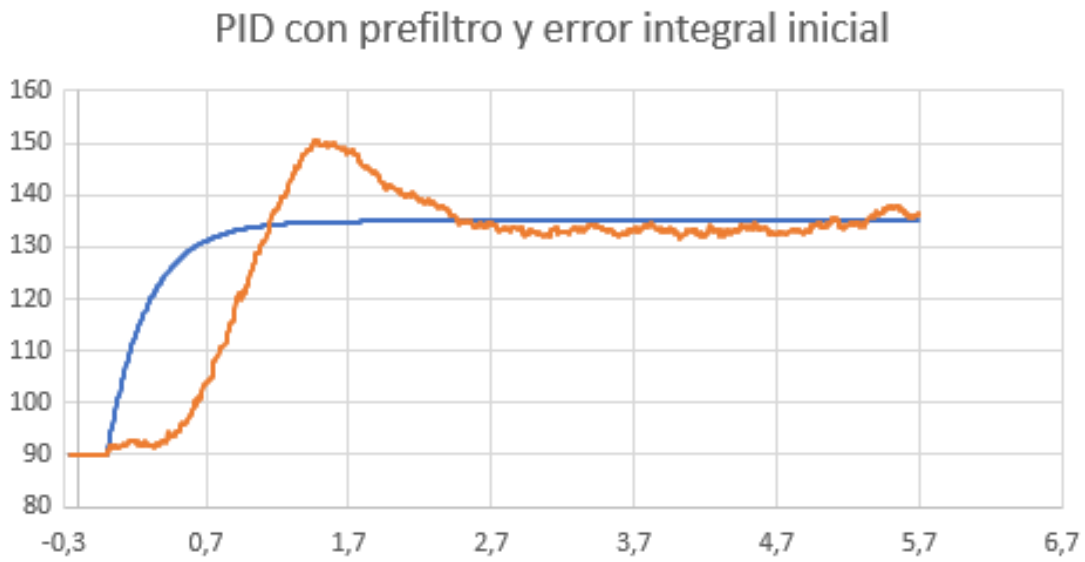

Figura 8: Respuesta al escalón del controlador PID con prefiltro. Setpoint en curva azul y ángulo en curva naranja.

Quedando así finalmente la respuesta frente a un disturbio como se observa en la Figura (9).

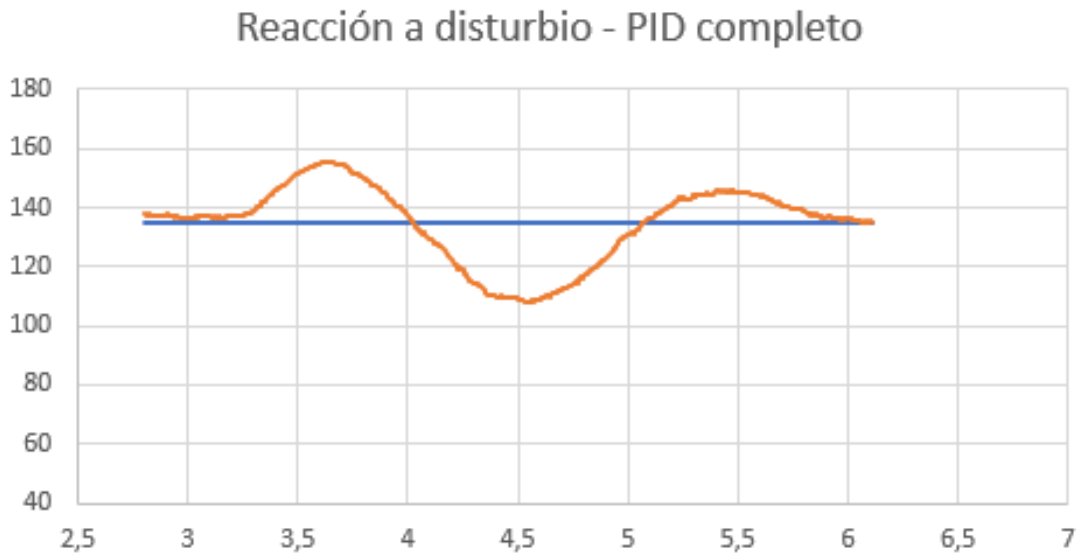

Figura 9: Respuesta a disturbio del controlador PID con prefiltro. Setpoint en curva azul y ángulo en curva naranja.

Finalmente, en las Figuras (10) y (11) se puede observar una comparación entre las respuestas al escalón y las respuestas a disturbios.

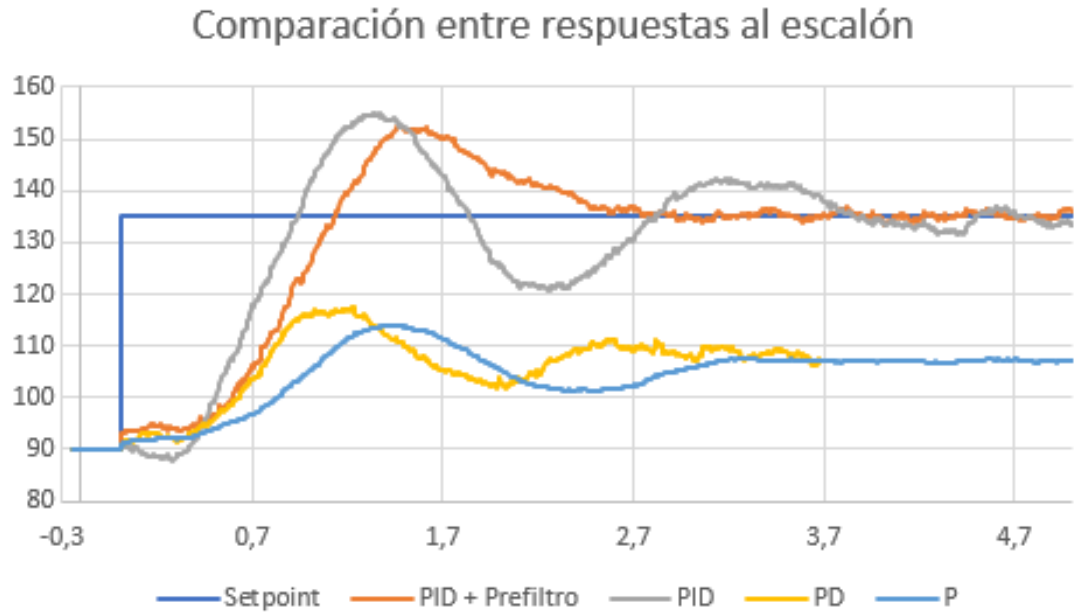

Figura 10: Comparación entre las respuestas al escalón de los distintos controladores.

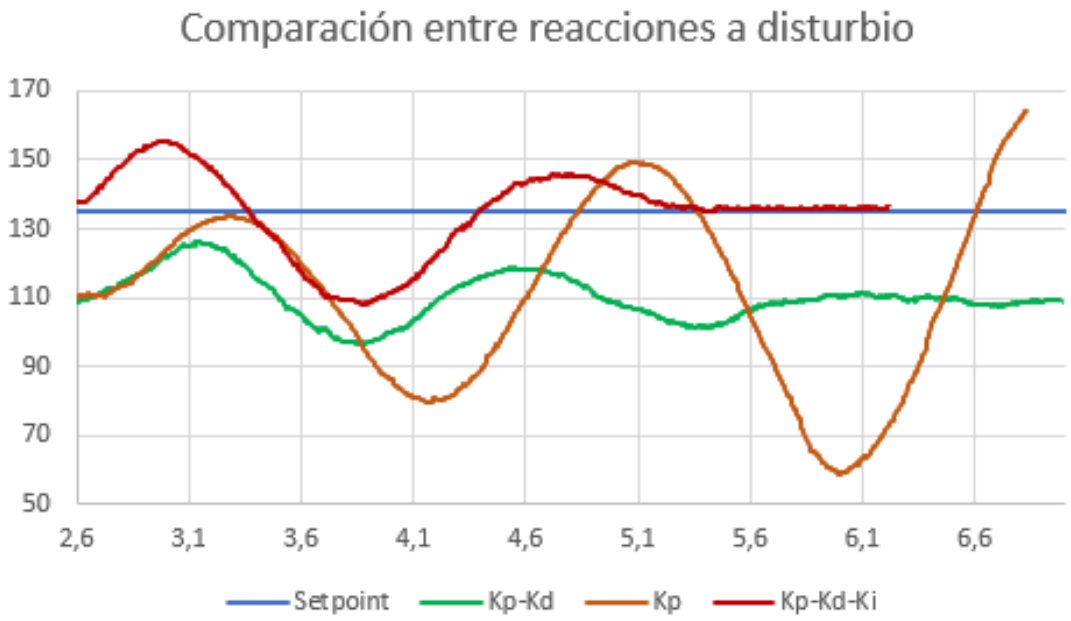

Figura 11: Comparación entre las respuestas a disturbio de los distintos controladores.

Finalmente, los valores de las sensibilidades quedaron $Kp = 0.55$, $Ki = 3 \cdot dt$ y $Kd = \frac{0.45}{dt}$ ajustados a frecuencia de sampleo genérica. Para el prefiltro se utilizó un filtro IIR de primer orden con frecuencia de corte de $65Hz$. El filtro de media móvil es de 40 samples.

4. Código

Luego se presenta el codigo implementador para realizar el control:

```
#define VALUE 80
#define SETPOINT 135
#define NPOLE 1
#define NZERO 1
#define ANGLE_OFFSET 1
#define SAMPLE_TIME_IN_MS 5
#define INITIAL_ANGLE 90

typedef double REAL;
REAL acoeff[]={-0.9816475744248808,1};
REAL bcoeff[]={1,1};
REAL gain=108.97742054932769;
REAL xv[]={45.0,45.0};
REAL yv[]={45.0,45.0};

REAL applyfilter(REAL v)
{
  int i;
  REAL out=0;
  for (i=0; i<NZERO; i++) {xv[i]=xv[i+1];}
  xv[NZERO] = v/gain;
  for (i=0; i<NPOLE; i++) {yv[i]=yv[i+1];}
  for (i=0; i<=NZERO; i++) {out+=xv[i]*bcoeff[i];}
  for (i=0; i<NPOLE; i++) {out-=yv[i]*acoeff[i];}
  yv[NPOLE]=out;
  return out;
}

```

```
29 double kp = 0.55;
30 double ki = 3/(SAMPLE_TIME_IN_MS/5);
31 double kd = 0.45*(SAMPLE_TIME_IN_MS/5);
32
33 HBRIDGE hb;
34 float input, output, setPoint;
35 bool clamped;
36 bool stop_bool;
37 unsigned long int prevTime, curTime, elapsedTime, initTime;
38 double totalTime;
39
40 double Setpoint, Input, Output;
41 PID myPID(&Input, &Output, &Setpoint, kp, ki, kd, DIRECT);
42
43 #define NUM_SAMPLES 40
44 double avg_buffer[NUM_SAMPLES];
45 int pointer;
46
47 void init_buffer(void){
48   unsigned int i;
49   for(i=0;i<NUM_SAMPLES;i++){
50     avg_buffer[i]=INITIAL_ANGLE;
51   }
52 }
53
54 void add_to_buffer(double angle)
55 {
56   avg_buffer[pointer %NUM_SAMPLES] = angle;
57   pointer++;
58 }
59
60 double get_filt_out(double angle)
61 {
62   add_to_buffer(angle);
63   double aux = 0;
64   for (int i = 0; i < NUM_SAMPLES; i++)
65   {
66     aux += avg_buffer[i];
67   }
68   return aux / (double)NUM_SAMPLES;
69 }
70
71 bool goingFoward;
72 void setup(void)
73 {
74
75   Setpoint = SETPOINT;
76
77   Serial.begin(115200);
78   init_buffer();
79   init_mpu();
80   hb.H_Bridge_Init(8, 7, 3); //Motor Plus, Minus and PWM
81   hb.H_Bridge_Set_Dir(H_FOWARD);
82   digitalWrite(13, HIGH);
83   stop_bool = false;
84   myPID.SetMode(AUTOMATIC); //AUTOMATIC);
85   myPID.SetOutputLimits(-VALUE, VALUE);
86   goingFoward = true;
87   pinMode(13, OUTPUT);
```

```
88    digitalWrite(13, LOW);
89    myPID.SetIntegralError(25);
90    prevTime = millis();
91    initTime = millis();
92  }
93
94  void loop(void)
95  {
96    static float angle = 0.0f;
97    curTime = millis();
98    elapsedTime = curTime - prevTime;
99    totalTime = (double)((curTime - initTime)/1000.0);
100   angle = (get_angle() * 180 / PI); //read angle in degrees
101   if(angle < 0){
102     angle += 360.0;
103   }
104   angle = get_filt_out(angle) + ANGLE_OFFSET;
105   Input = angle;
106   if(elapsedTime >= SAMPLE_TIME_IN_MS){
107     myPID.SetSampleTime(elapsedTime);
108     Setpoint = applyfilter(SETPOINT);
109     myPID.Compute();
110
111     uint8_t aux = (uint8_t)abs(Output);
112     if (Output > 0)
113     {
114       hb.H_Bridge_Set_Dir(H_FOWARD);
115       digitalWrite(13, HIGH);
116     }
117     else
118     {
119       hb.H_Bridge_Set_Dir(H_BACKWARD);
120       digitalWrite(13, LOW);
121       aux=aux*2;
122     }
123     hb.H_Bridge_Set_Pwm(aux);
124     prevTime = curTime;
125   }
126 }
```