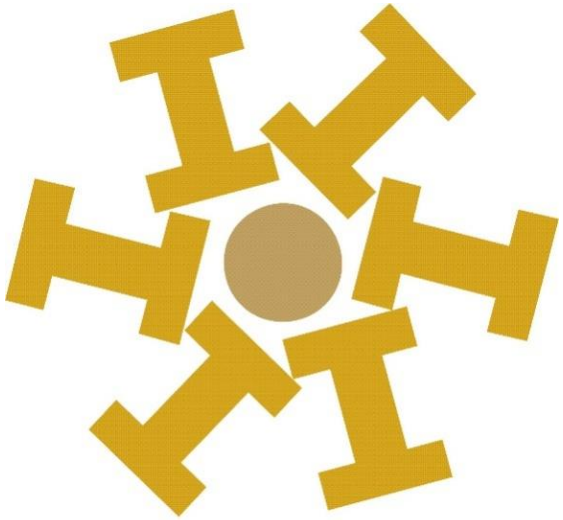
SISTEMAS PARA CONTROL

TRABAJO PRÁCTICO Nº 5

PARCIAL 2

Grupo 3
Diamantino, Mauro – Registro 24391
Marín, Carlos – Registro 25437
Quintana, Mario – Registro 24397



Facultad de Ingeniería
Universidad Nacional de San Juan

Contenido

1-Objetivos 1

2- Laboratorio – Uso del Ultrasonido como sensor 1

2.1-Propuesta teórica con esquema de control y esquema eléctrico 1

2.2-Funcionamiento del sistema 4

2.3- Conclusiones sobre el laboratorio 6

3 - Desarrollo e innovación 7

4- Apéndice – Código implementado para el Laboratorio 7

1-Objetivos

- Utilización de un sensor de medición de distancia (ultrasonido)
- Implementar un software de comunicación, procesamiento y control sobre una plataforma de hardware basada en microcontrolador.
- Innovación y diseño de un sensor y/o control propuesto.

2- Laboratorio – Uso del Ultrasonido como sensor

2.1-Propuesta teórica con esquema de control y esquema eléctrico

Se armó el siguiente esquema de control:

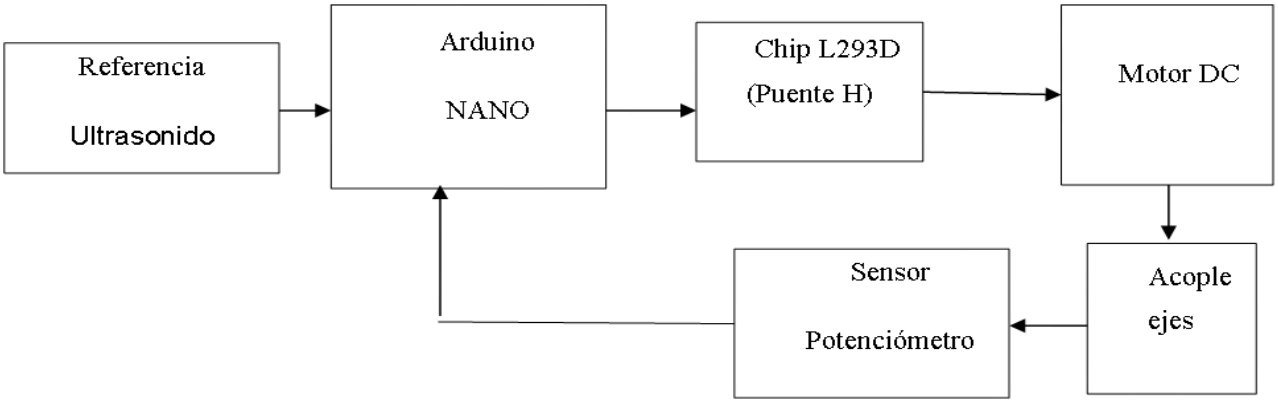


Ilustración 1

Se implementó en Arduino un lazo de control PID modificado, con un filtro pasa bajo en cascada con la acción derivativa, para controlar la posición de un motor de corriente continua. Las características de este motor C9050-6003 son: voltaje 18v, impedancia 11Ω y velocidad angular de 8800 revolución/minutos. Como referencia de posición se usó un ultrasonido modelo HC-SR04 y para censar la posición angular se usó un potenciómetro lineal de 100k Ω. En resumen, el sistema controla la posición angular del eje del potenciómetro sensor (indirectamente la posición del eje del motor) en función de la distancia a la que se encuentre un obstáculo frente al ultrasonido.

Para este controlador, se usó un periodo de muestreo del lazo de control de 1ms, y transmitiendo una muestra cada 100ms desde Arduino a la PC. Cabe aclarar que, el ultrasonido puede muestrearse como máximo 1 vez cada 60 ms, por lo que se muestreó la referencia cada 70 ms dando un margen a lo recomendado.

La planta de este sistema, consistió en estructura metálica sobre la cual se montó el motor de corriente continua, el cual estaba acoplado al sensor (potenciómetro) a través de un juego de engranajes y polea que funcionó como reducción, lo cual se observa en la Ilustración 2. Con este montaje se evitó sobrecargar el motor, para que el consumo de corriente sea menor a lo que soporta el integrado puente H.

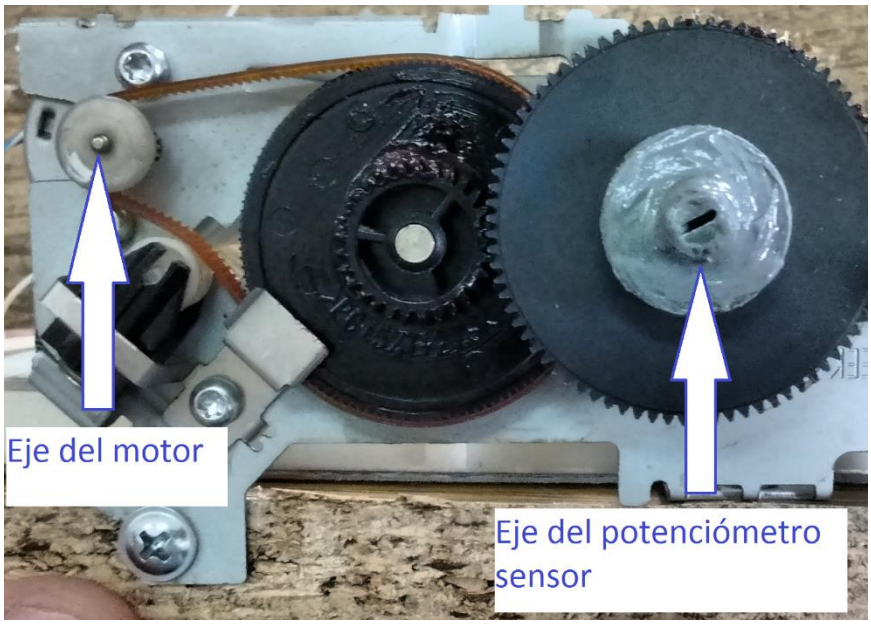


Ilustración 2

El sensor (potenciómetro) tiene un mapeo tal que las lecturas de su pin central, a través de una entrada analógica, se llevan del rango 0 a 5volts al rango de 0 a 300°. Este mapeo se observa en la siguiente figura.

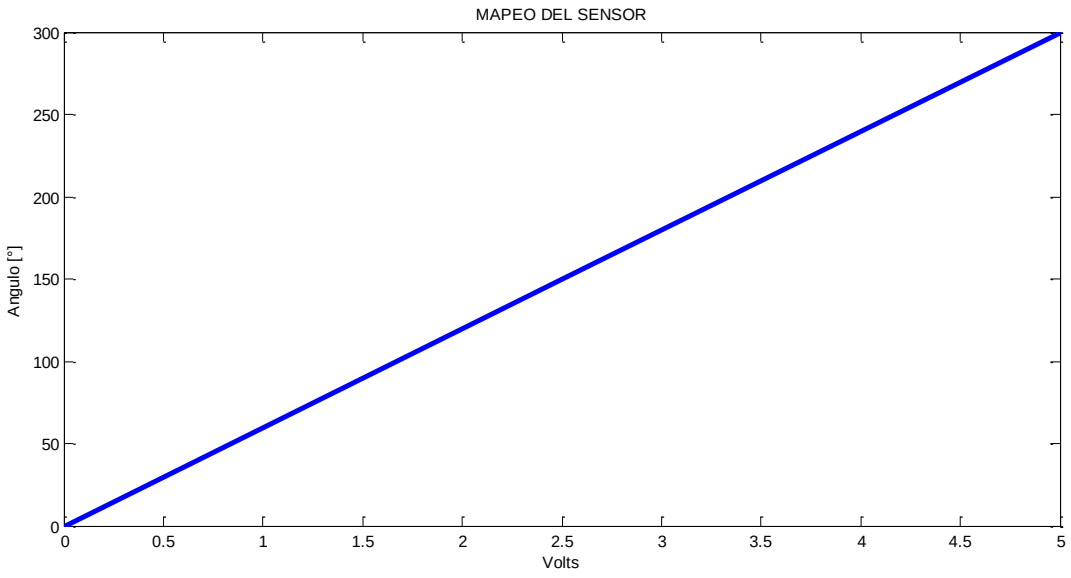


Ilustración 3

Por otro lado el ultrasonido (referencia) puede medir con un error de 3mm en un rango de 3 a 400cm. Por razones de practicidad para las pruebas realizadas, se acotaron las mediciones del ultrasonido entre 5 y 50 cm, quedando el mapeo a grados como indica la siguiente figura.

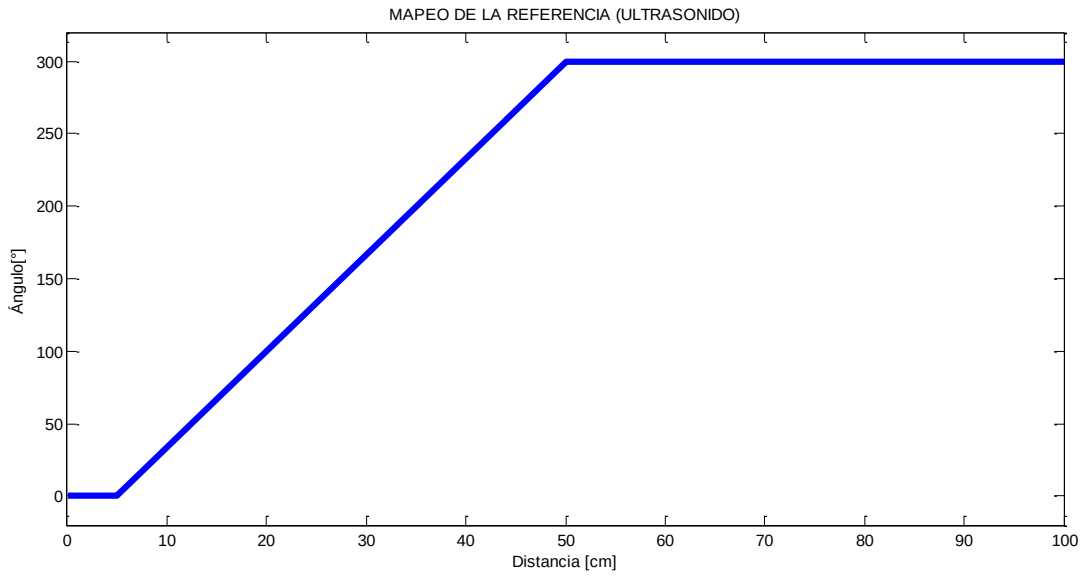


Ilustración 4

A través del código cargado al Arduino, se compararon la señal de referencia y la realimentación con lo que se determinó la señal de error, entrada del controlador PID. Una vez obtenida la señal de acción de control, se la adecuó teniendo en cuenta la zona muerta del motor y la tensión máxima disponible.

Para controlar el giro del motor se excitó, usando PWM, las entradas al puente H. Si la acción de control es de valor positiva, entonces el motor debe girar en sentido horario, esto se logra enviando PWM (con un ciclo útil proporcional al valor de la acción de control) a través del pin 5 del arduino (conectado al pin 15 del punte H) y colocando en 0v el pin 6 del Arduino (conectado al pin 10 del puente H). En cambio, si la acción de control es de valor negativo, entonces el motor debe girar en sentido anti horario, esto se logra enviando PWM (proporcional al valor de la acción de control) a través del pin 6 del arduino y colocando en 0v el pin 5 del arduino. Cabe aclarar, que la habilitación de los drivers utilizados del puente H está siempre a 5v. Por último, para el caso de que la acción de control esté en 0v, tanto el pin 5 como el 6 se colocaron a 0v.

Toda esta descripción se puede observar en la Ilustración 5:

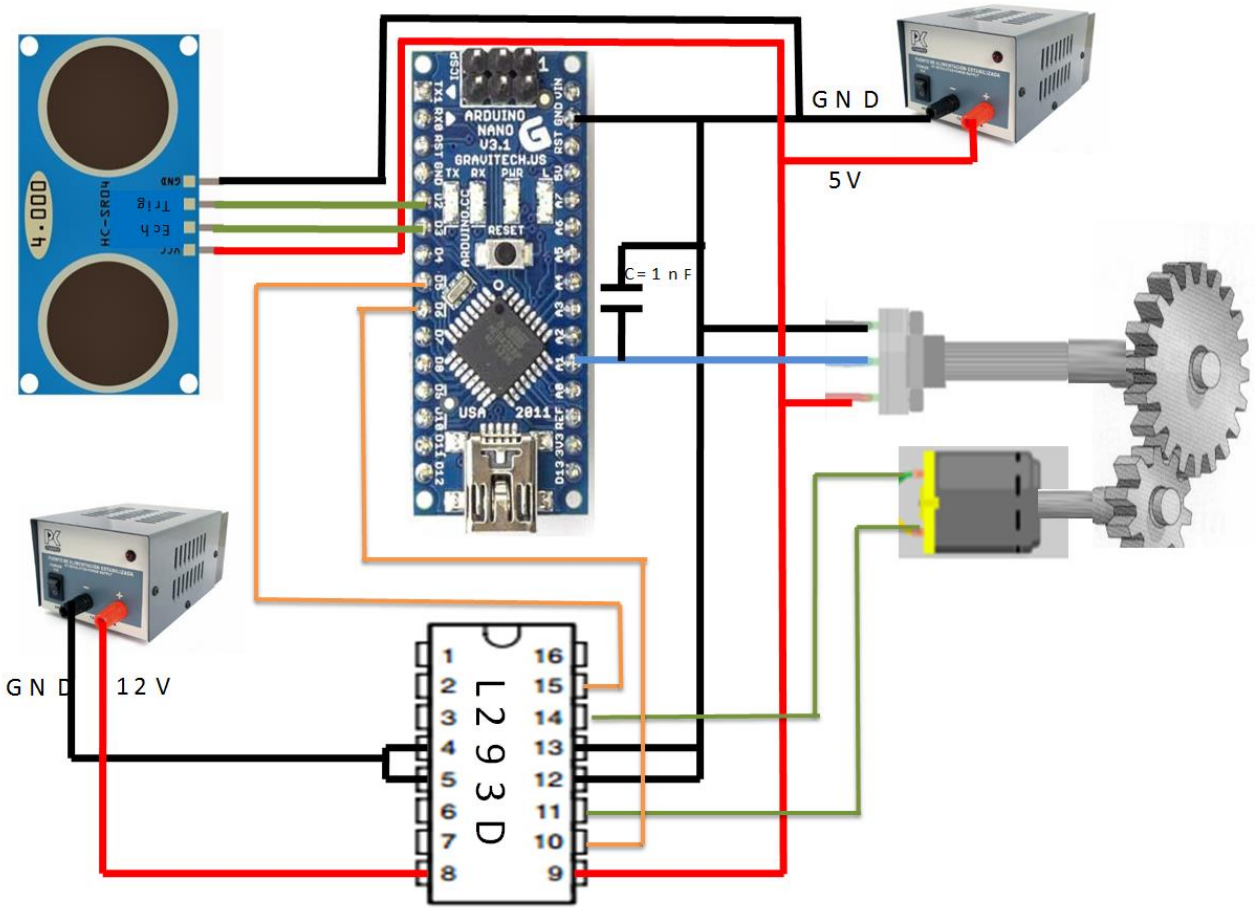


Ilustración 5

2.2-Funcionamiento del sistema

El sistema posee un controlador PID modificado, con un filtro pasa bajos de primer orden en cascada con la acción derivativa:

$$u(k) = Kp * e(k) + i(k) + d(k)$$

Donde:

$$i(k) = i(k-1) + Kp * Ki * Ts\left(\frac{e(k) + e(k-1)}{2}\right)$$

$$d(k) = \left(\frac{Td}{Ts + Td}\right) * [d(k-1) - Kp * (y(k) - y(k-1))]$$

Siendo: Kp, Ki y Td, las constantes proporcional, integrativa y derivativa respectivamente, y además Ts es el tiempo de muestreo.

Los parámetros del controlador para esta planta fueron ajustados en la práctica 1, por lo que no se fue necesario hacer este ajuste nuevamente, entonces los parámetros con los que se trabajó fueron: Kp=0.1, Td=0.05 y Ki=0.5. Cabe destacar que se colocó un capacitor de 1nF a la entrada que lee el potenciómetro sensor, para suavizar las oscilaciones de esta medición.

En las siguientes ilustraciones se observa el funcionamiento del sistema, en las cuales se ve que responde de forma óptima en todo el rango de control. Además, es importante recordar que cuando la acción de

control es positiva, el motor gira en sentido anti horario y cuando la acción de control es negativa, el motor gira en sentido horario.

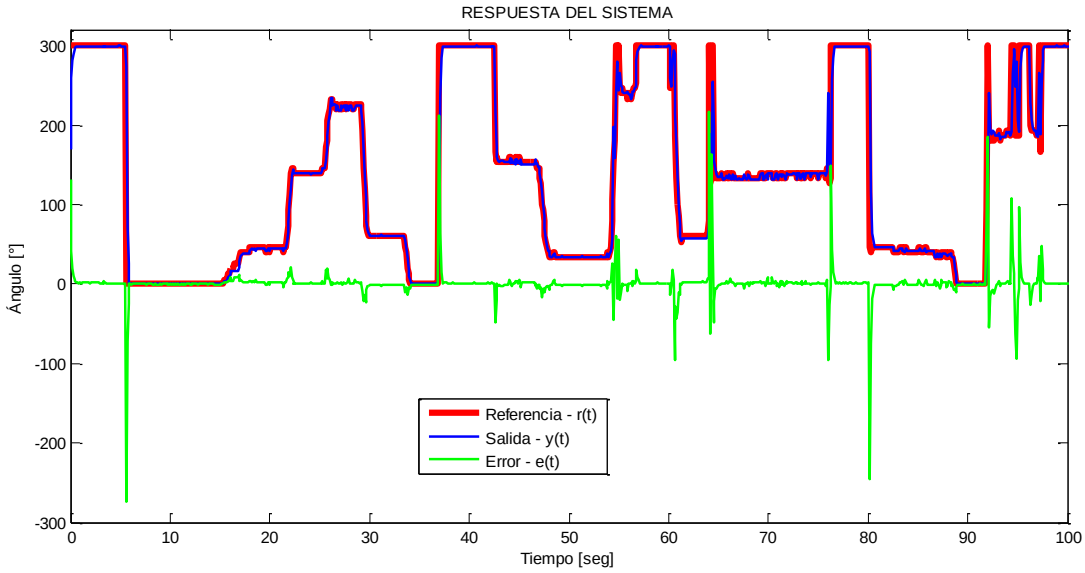


Ilustración 6

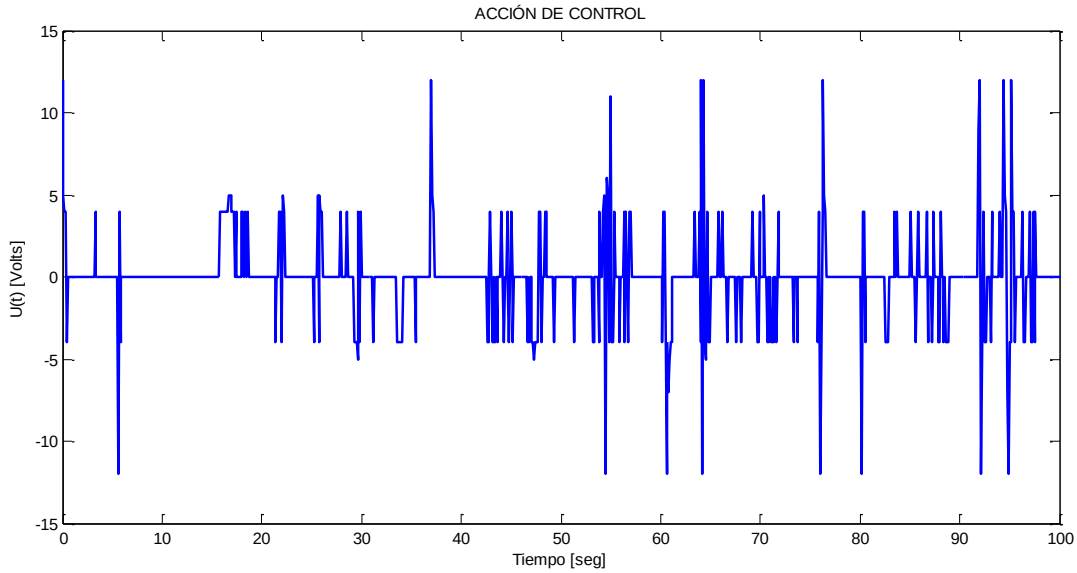


Ilustración 7

En una segunda prueba, se verificó la capacidad de regulación del sistema. Para esto, se mantuvo la referencia en tres puntos fijos 300º, 0º y 130º durante tres periodos te tiempo distintos, a su vez se forzó manualmente al eje del motor para cambiarlo de posición. Por lo que podemos apreciar en la Ilustración 9, cuando se producen las perturbaciones, el error es distinto de cero y la acción de control sigue las variaciones del error. Sin embargo se observa que, aunque el error sea contante como en la perturbación 4, la acción de control tiende a aumentar, esto es debido a la parte integrativa del controlador que va acumulando el error. Lo mencionado se refleja en las siguientes ilustraciones.

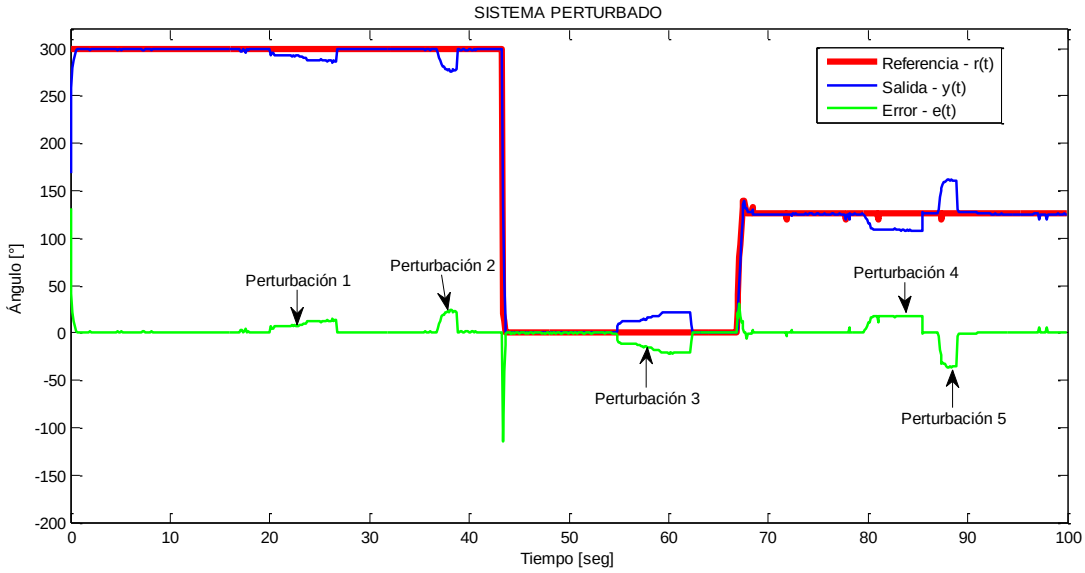


Ilustración 8

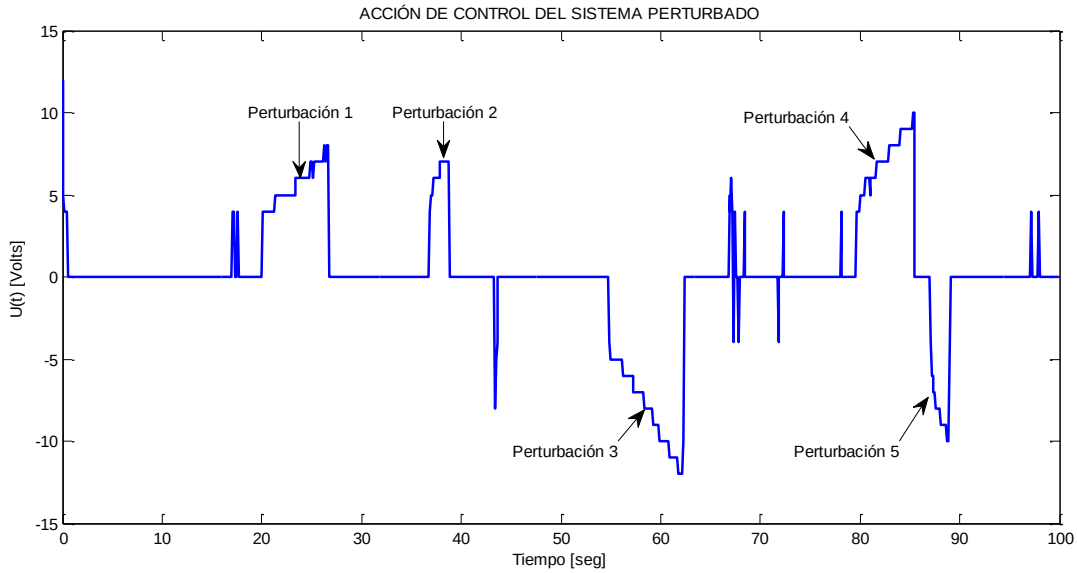


Ilustración 9

2.3- Conclusiones sobre el laboratorio

Esta experiencia permitió entender el funcionamiento del sensor ultrasonido, que junto a las funciones de Arduino nos encontramos con una implementación sencilla. El ultrasonido, es un sensor muy útil para medir distancias a un bajo costo, pero su gran desventaja es su amplio ángulo de medida de 15º.

Cabe destacar, que por el requerimiento de procesamiento del computador, se tomó una de estas muestras cada 100mS a través de Simulink para su graficación.

3 - Desarrollo e innovación

4- Apéndice – Código implementado para el Laboratorio

```
//////////////////////////LIBRERIAS////////////////////////////////
#include<TimerOne.h>

//////////////////////////CONSTANTES////////////////////////////////
//Motor
#define RANGO 300  //Angulo de giro del potenciómetro sensor
#define ZONA_MUERTA 4.0
#define TOLERANCIA 2
#define TENSION_MAX 12 //Nuestro motor es de 18V, 11 Ohm y 8800RPM
#define Ts 0.001  //Periodo de muestreo de 1 ms

//Ultrasonido
#define Trig 2
#define Echo 3

//////////////////////////VARIABLES////////////////////////////////
//Motor
int sensor;
int hora = 5;//Salidas PWM para el motor
intantihora = 6;//Salidas PWM para el motor
float y, r, u, y1,u1;//señal de salida,referencia, acción de control,  y1(k)=y(k-1)
                //u1 solo utiliza para graficar la accion de control
floatKp, Ki, Kd, Td; //Parametros PID + Td<1
float integral, integral1; //integral1 es la acumulacion de los errores de parte integral
float e, e1, d1, d; //error, termino derivativo  e1=e(k-1)  d1(k)=d(k-1)
bytepwm_motor;
bytecontador_tx = 0;
boolbandera_control = false; //activa tarea lazo de control
boolbandera_tx = 0; // activa tarea transmisión de datos
char buffer[20];

//Ultasonido
long distancia;
long tiempo;
```

```
boolbandera_ref = false;//Activa la toma de una referencia.
                    //La referencia es lenta comparada con la planta
                    //Una referencia cada 60ms maximo
                    //De lo contrario el Ultrasonido arroja valores erroneos
bytecontador_ref = 0;

////////////////////////////SETUP////////////////////////////////
void setup()
{
//Inicializa comunicación serie a 115200 bits por segundo
Serial.begin(115200);

//Motor
  //////CONFIGURA PWM
  //*   - Base frequencies:
  // *      o The base frequency for pins 3, 9, 10, and 11 is 31250 Hz.
  // *      o The base frequency for pins 5 and 6 is 62500 Hz.
  //   o The divisors available on pins 5, 6, 9 and 10 are: 1, 8, 64,
  // *        256, and 1024.
  // *      o The divisors available on pins 3 and 11 are: 1, 8, 32, 64,
// *        128, 256, and 1024.
setPwmFrequency(5, 8);
setPwmFrequency(6, 8);

  //Configura para parametros del controlador
Td= 0.05;
Kp = 0.1;
  Ki= 0.5;
  integral1 = 0.0;
  e1 = 0.0;
  d1 = 0.0;

pinMode(hora, OUTPUT); //Salidas PWM para el motor
pinMode(antihora, OUTPUT);//Salidas PWM para el motor

Timer1.initialize(1000); // configura un timer de 1 ms
Timer1.attachInterrupt(timerIsr ); // asocia una la interrupción de un timer a una rutina de servicio de interrupción
```

```
//Ultrasonido
pinMode(Trig, OUTPUT); /*Configuración del pin 9 como salida: para el pulso ultrasónico*/
pinMode(Echo, INPUT); /*Configuración del pin 8 como entrada: tiempo del rebote del ultrasonido*/

}

////////////////////////////////LOOP/////////////////////////////
voidloop()
{
//Ultrasonido
if (bandera_ref == 1){ //Muestreo referencia
digitalWrite(Trig,LOW); /* Por cuestión de estabilización del sensor*/
delayMicroseconds(5);
digitalWrite(Trig, HIGH); /* envío del pulso ultrasónico*/
delayMicroseconds(10);
digitalWrite(Trig, LOW);

tiempo=pulseIn(Echo, HIGH); /* Función para medir la longitud del pulso entrante. Mide el tiempo que transcurrido entre el envío
del pulso ultrasónico y cuando el sensor recibe el rebote, es decir: desde que el pin 8 empieza a recibir el rebote, HIGH, hasta que
deja de hacerlo, LOW, la longitud del pulso entrante*/
distancia= int(0.017*tiempo); /*fórmula para calcular la distancia obteniendo un valor entero*/
   /*Monitorización en centímetros por el monitor serial*/
if(distancia > 50){
distancia=50;
   }
if(distancia < 5){
distancia = 5;
   }
bandera_ref = 0;
  }

if (bandera_control == 1) // Lazo de control
  {
   //Conversion a grados de referencia
   r = (distancia - 5) * RANGO / 45;
```

```
    //Lee tensión en potenciometro (SENSOR) a través de un canal del conversor A/D: [0,1023]
sensor = analogRead(A1);
    //valor actual en grados
    y = sensor * (RANGO / 1023.0);

    //Calculo de error
e = r - y;

    //Calculo de integral con una aproximacion trapezoidal
integral = integral1 + (Kp*Ki*Ts*(e + e1)/2);
    //Cálculo de término derivativo
    d = (Td/(Ts+Td)) * (d1 - Kp * (y - y1));
    //Controlador PID
    u = Kp * e + integral + d;

//Motor
    //Signo positivo o negativo (giro horario o anti-antihorario) o alta impedancia del puente H
if (abs(e) < TOLERANCIA) {
analogWrite(hora, 0); // Detengo el motor
analogWrite(antihora, 0);
integral = 0.0;
      u = 0.0;
      u1=u;
    } else {
if (u >= 0) { //GIRO EN SENTIDO HORARIO
       u = u + ZONA_MUERTA;
if (u > TENSION_MAX) {
       u = TENSION_MAX;
       }
       //mapeo de tension [volt] a PWM de 8 bits
       u1=u;
pwm_motor = (byte)((float)u * (255.0 / TENSION_MAX));//establece el duty (porcentaje de tiempo en "1") de la señal PWM
analogWrite(hora, pwm_motor);
analogWrite(antihora, 0);
     } else {  //GIRO EN SENTIDO ANTIHORARIO
       u = abs(u) + ZONA_MUERTA;
```

```
if (u > TENSION_MAX) {
        u = TENSION_MAX;
        }
        u1=-u;
        //mapeo de tension [volt] a PWM de 8 bits
pwm_motor = (byte)((float)u * (255.0 / TENSION_MAX));//establece el duty (porcentaje de tiempo en "1") de la señal PWM
analogWrite(hora, 0);
analogWrite(antihora, pwm_motor);
      }
    }

    //Actualizacion de variables
    e1 = e;
    d1 = d;
    integral1 = integral;
    y1 = y;

bandera_control = 0;
  }

if (bandera_tx == 1){
sprintf(buffer, "%d,%d,%d",(int) r,(int) y,(int)u1);
Serial.println(buffer);
bandera_tx = 0;
}

}

//rutina (función) llamada cada interrupción del timer (1 ms)
voidtimerIsr()
{
bandera_control = 1; //activa tarea lazo de control cada 1 ms
contador_ref++;
contador_tx++;
if (contador_ref>= 70){//Se muestrea la entrada cada 70ms
bandera_ref=1;
contador_ref = 0;
```

```
  }
if (contador_tx>= 100){
bandera_tx=1;
contador_tx = 0;
  }
}

voidsetPwmFrequency(int pin, int divisor) {
byte mode;
if(pin == 5 || pin == 6 || pin == 9 || pin == 10) {
switch(divisor) {
case 1: mode = 0x01; break;
case 8: mode = 0x02; break;
case 64: mode = 0x03; break;
case 256: mode = 0x04; break;
case 1024: mode = 0x05; break;
default: return;
    }
if(pin == 5 || pin == 6) {
      TCCR0B = TCCR0B & 0b11111000 | mode;
    } else {
      TCCR1B = TCCR1B & 0b11111000 | mode;
    }
  } else if(pin == 3 || pin == 11) {
switch(divisor) {
case 1: mode = 0x01; break;
case 8: mode = 0x02; break;
case 32: mode = 0x03; break;
case 64: mode = 0x04; break;
case 128: mode = 0x05; break;
case 256: mode = 0x06; break;
case 1024: mode = 0x7; break;
default: return;
    }
   TCCR2B = TCCR2B & 0b11111000 | mode;
}
}
```