Implementación control PID en Seguidor solar

Para controlar el seguidor solar ya tenemos el Angulo de la hora actual, derivado de los cálculos previamente expuestos, ahora es necesario hacer que el seguidor solar se posiciones en el Angulo para ello se requieres transmitirle esa información al motor y posicione correctamente.

Para lograr este objetivo se requiere la implementación un **control PID** (Proporcional Integral y Derivativo) que se encargara que el Angulo se mantenga constante y un **sistema conversor** que transmite los valores a la etapa de potencia que controla el motor (puente H)

El funcionamiento del PID depende de 2 variable de entrada

* Angulo deseado (Angulo de orientación solar)
* Angulo real (Angulo en el que el panel solar esta actualmente)

Estas variables son analógicas y pueden tomar valores que van desde 0 a 5 voltios

Para tal fin se hace uso del módulo analógico que los PIC integran, este modulo convierte el voltaje en una secuencia binaria, de 8 o 10 bits según se configure el módulo.

Según sea la cantidad de bits a utilizar, será mayor o menor la resolución de la lectura y por ende la precisión

Si utilizamos 8 bits solo podemos realizar 254 valores entre 0 a 5 voltios

En cambio 10 bits nos permiten 1024 valores entre 0 a 5 voltios

En este caso se usan 10 bits permitiendo mayor resolución y por ende mejor precisión.

Un inconveniente de usar este módulo, los valores almacenados en formato de bits, no son imprácticos al realizar todas las operaciones del PID que opera utiliza valores de tipo Float. Por lo tanto, se procede a convertir los bits en valores Float.

Para resolver este inconveniente se realizó la conversión de formato binario a Float al realizando una operación y obtener el voltaje y almacenarlo en el formato necesario utilizando la siguiente formula.

################################definicion de PID

################################# ecuaciones PID

* e=error Actual
* k=número de muestra
* t=periodo de muestreo
* kp=Ganancia Proporcional
* ki=Ganancia integral
* kd=Ganancia derivativa
* u=resultado

################################definicion de etapa de potencia

#include <18F4550.h>

#device ADC=10

#fuses XT,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT,INTRC\_IO

#use delay(clock=4000000)

#INCLUDE <LCD.C>

#USE STANDARD\_IO(A)

#USE STANDARD\_IO(B)

#USE STANDARD\_IO(C)

#USE STANDARD\_IO(D)

#USE STANDARD\_IO(E)

//configuraciones entradas/salidas

//entradas

   #define PIN\_SetPoint 0    //[AN0]Setpoint      set\_adc\_channel(0);

   #define PIN\_PocitionReal 1 //[AN1]posicion real set\_adc\_channel(1);

//salidas

   //control puente H

      //comando de control de giro del motor

            //set\_pwm1\_duty() [RC2]

            //set\_pwm1\_duty() [RC3]

//variables tipo bits

unsigned int16 bits\_SetPoint=0;

unsigned int16 bits\_PocitionReal=0;

unsigned int16 bits\_difer=0;

unsigned int16 bits\_RES=0;

//variable control PWM

int16 duty=0;

int Timer2,Poscaler;

//variable control giro motor

int cambio, giro=1;//giro 1 si es negativos los valores

//#########     PID

float Var\_SetPoint=0;

float Var\_PocitionReal=0;

float Var\_error\_0=0;

float Var\_error\_1=0;

float Var\_error\_2=0;

float a,b,c;               //variables temporales

float t=1;                 //periodo de muestreo

int k=0;                   //contador de muestras

float Var\_res\_0;           //Salidas

float Var\_res\_1;           //Salidas

//#########    ganancias

float kp=1;                //proporcional

float kd=0.2;              //Derivativa

float ki=0.2;              //integral

vid main() {

   Timer2=249;    //Se carga Timer 2 con 249

   Poscaler=1;  //Preescaler solo puede tomar valores de: 1

   //Configuración de Timer 2 para establecer frec. PWM a 1kHz

   setup\_timer\_2(t2\_div\_by\_4,Timer2,Poscaler);

   setup\_ccp1(ccp\_pwm);      //Configurar modulo CCP1 en modo PWM

   setup\_ccp2(ccp\_pwm);      //Configurar modulo CCP2 en modo PWM

   //Configurar ADC configuración

   //de los puertos que funcionan en analógico

   setup\_adc\_ports(allánalo);

   setup\_adc(adc\_clock\_internal);

   LCD\_INIT();       //Inicializa el LCD

   LCD\_PUTC("\f");   //Borrar el contenido del LCD

   while(true){

      //leer Setpoint

      set\_adc\_channel(PIN\_SetPoint);//selecciona valore deseado setpoint

      delay\_us(100);

      bits\_SetPoint=read\_adc();

      //leer posición real

      set\_adc\_channel(PIN\_PocitionReal);  //seleciona la posicion real

      delay\_us(100);

      bits\_PocitionReal=read\_adc();       //leer

      //cálculos

      //convierte los bits en voltaje

      Var\_SetPoint      = (5.0\*bits\_SetPoint)/1024.0;

      Var\_PocitionReal  = (5.0\*bits\_PocitionReal)/1024.0;

      bits\_difer=bits\_PocitionReal-bits\_SetPoint;

      //diferencia de voltaje

      c=Var\_PocitionReal-Var\_SetPoint;

      //PID cálculos

      Var\_error\_2=Var\_error\_1;

      Var\_error\_2=Var\_error\_0;

      Var\_error\_0=Var\_PocitionReal-Var\_SetPoint;

      //calculo

            a=Var\_error\_0\*( kp + (ki\*t) )+( kd/t );

            b=Var\_error\_1\*( (-2\*(kd/t) - kp) );

            c=Var\_error\_2\*( (kd/t) +Var\_res\_1 );

            Var\_res\_1=Var\_res\_0;

            Var\_res\_0=a+b+c;

         //conversión para PWM

         if(Var\_res\_0<0){//control de giro

            Var\_res\_0=Var\_res\_0\*-1;

            giro=2;

         }else{giro=1;}

         if(Var\_res\_0>=4.90){//control contra desbordamiento

            Var\_res\_0=4.99;

         }

         bits\_RES      = (1024\*Var\_res\_0) /5;

         //incremento en contador

            k=k+1;

      //interfaz

      LCD\_GOTOXY(1,1);

      printf(lcd\_putc," %4Lu",bits\_difer); lcd\_putc("b");

      printf(lcd\_putc," %f",Var\_error\_0); lcd\_putc("V");

      LCD\_GOTOXY(1,2);

      printf(lcd\_putc," %4Lu",bits\_RES);  lcd\_putc("b");

      printf(lcd\_putc," %f",Var\_res\_0);   lcd\_putc("V");

      delay\_ms(100);

      LCD\_PUTC("\f"); //Borrar el contenido del LCD

      // EL SISGUIENTE CODIGO SE ENCARGA DE CONVERTIR

      // el valor de salida del pid en valores validos

      // para el puente h de dos hilos el sentido

      // de giro se definirá por el signo ->0<+

      if(Var\_PocitionReal<Var\_SetPoint){//valores positivos

         c=Var\_SetPoint-Var\_PocitionReal;

         set\_pwm1\_duty(bits\_RES);   //write salida de señal PWM

         output\_high(PIN\_B1); //led indicador de giro

      }else{

         output\_low(PIN\_B1);

         set\_pwm1\_duty(0);

      }

      if(Var\_PocitionReal>Var\_SetPoint){  //valores negativos

         c=Var\_PocitionReal-Var\_SetPoint;

         set\_pwm2\_duty(bits\_RES);         //write salida de señal PWM

         output\_high(PIN\_B2);             //led indicador de giro

      }else{

         output\_low(PIN\_B2);

         set\_pwm2\_duty(0);

      } //si el sentido de giro no corresponde apaga el led y PWM

      if(Var\_SetPoint==Var\_PocitionReal){

         set\_pwm2\_duty(0);

         set\_pwm1\_duty(0);

      }

   }

}