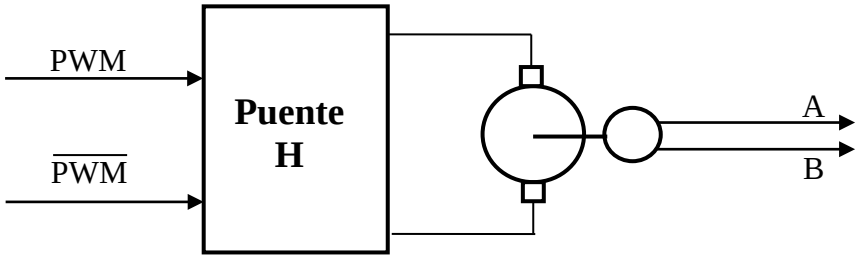
Practica 8: Sistemas muestreados. Control de velocidad de un motor.

1 Objetivos

- Implementación de un controlador de velocidad de un motor de corriente continua. Utilización de un encoder, generación de PWM y programación de un regulador.

2 Descripción sistema

Se desea realizar el control de velocidad de un motor de corriente continua. El motor está equipado con un encoder incremental.



3 Trabajo a realizar

Se dispone de un módulo PWM ya implementado, pwm.h y pwm.c. El módulo produce la onda PWM y su negada por las patillas P6.2 y P6.3.

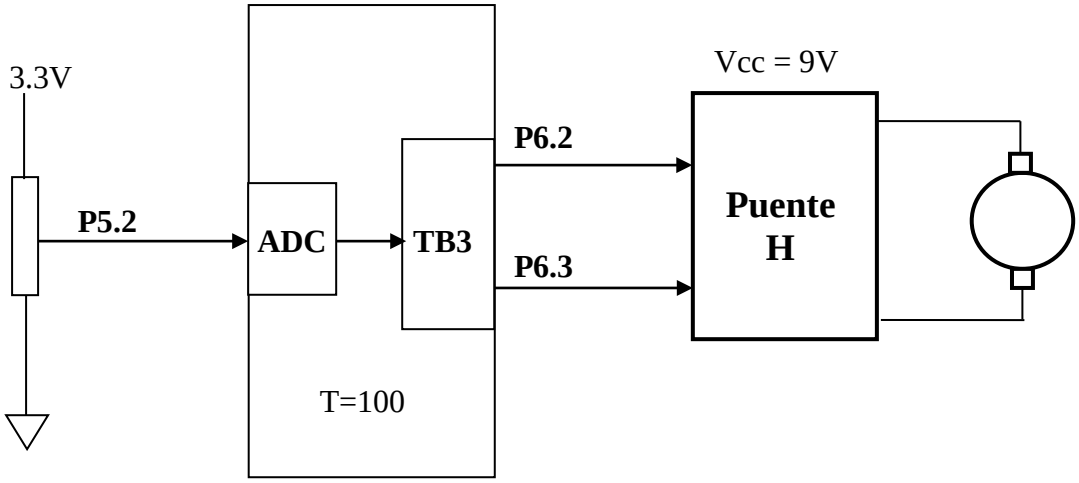
Se dispone de un módulo ENCODER ya implementado, encoder.h y encoder.c. El módulo lee las señales A y B del encoder por las patillas P6.0 y P6.1.

Se dispone de un módulo SERVOS parcialmente implementado, servos.h y servos.c. El módulo calcula la velocidad angular del motor en función de las medidas del encoder y del tiempo.

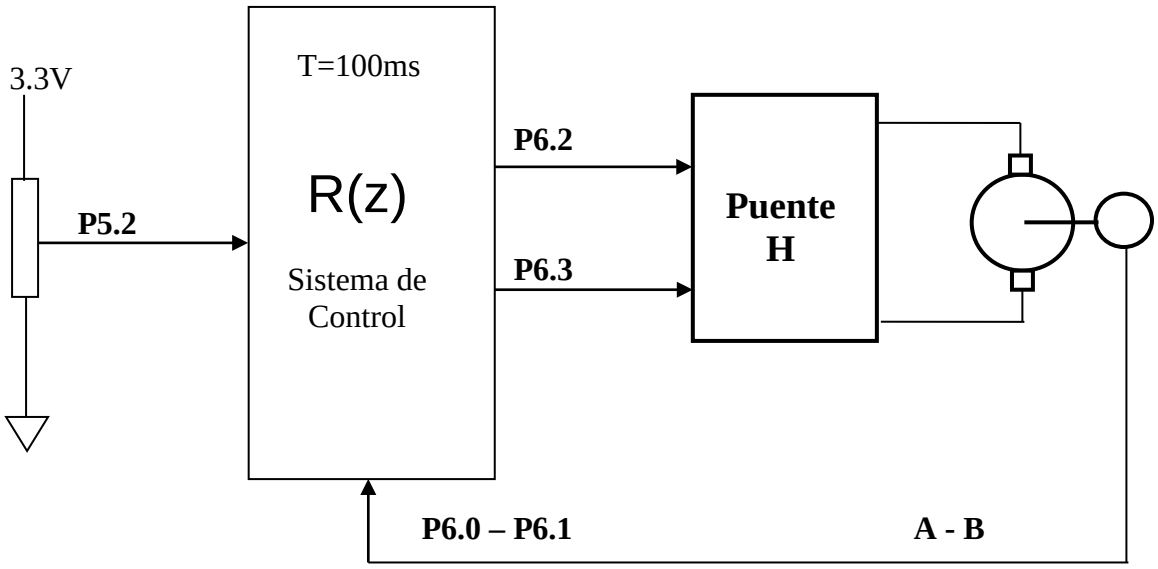
También se dispone de un módulo de conversión AD que lee las entradas analógicas por las patillas P5.2 y P5.3.

Estúdiense estos módulos para su correcta utilización.

3.1 Accionar el motor mediante PWM



3.2 Implementar control con regulador y T=100ms



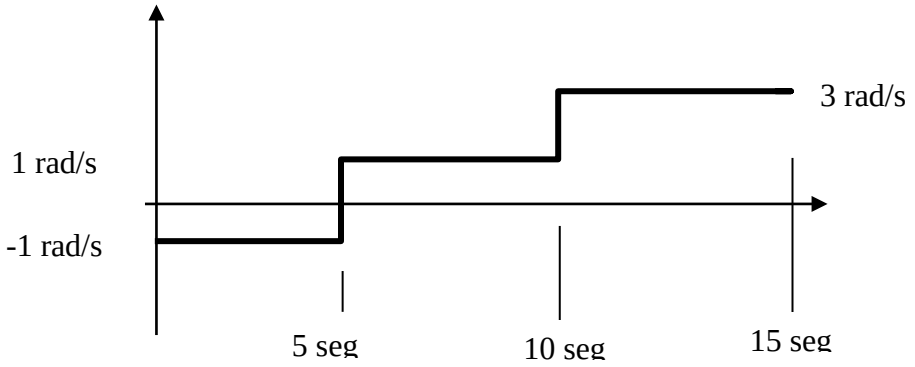
$$R(s) = \frac{U(s)}{E(s)} = 2.86\frac{1+0.1s}{s}$$

La acción U se expresa en voltios y el error E en rad/s. E = Wref – W

La lectura del potenciómetro hay que transformarla de forma lineal en una referencia entre -4 rad/s y 4 rad/s.

El cálculo de la velocidad basado en el encoder es en rad/s.

3.3 Eliminar el potenciómetro y generar la siguiente consigna por programa



Almacenar 15s de muestras de velocidad en un buffer y comprobar que el comportamiento del motor controlado es correcto.

3.4 Determinación del tiempo de cómputo

Se quiere determinar cuál es el tiempo de cómputo del control, esto es cuánto tiempo cuesta realizar cada iteración del bucle. Para ello utilizar una salida digital del μC.

4 Programas

Se dan los siguientes módulos ya implementados. Se dispone de los ficheros de cabecera que contienen la especificación y de su implementación en C.

4.1 ad.h

```
void Init_AD (void) ;
unsigned int Read_Value_Int_1 (void) ;
unsigned int Read_Value_Int_2 (void) ;
```

4.2 pwm.h

```
void Init_PWM (void) ;
void Set_Value_10b (unsigned int value) ;
```

4.3 clock.h

```
#define PERIODIC 1
#define ONE_SHOT 0

typedef unsigned char Timer_id ;
typedef unsigned char Timer_type ;

void Init_Clock (void) ;
unsigned int Get_Time (void) ;
void delay_until(unsigned int T);
Timer_id Set_Timer (unsigned int Ticks, Timer_type tt, void (*p)(void)) ;
char Time_Out (Timer_id n) ;
void Remove_Timer (Timer_id n) ;
```

4.4 encoder.h

```
void Init_Encoder (void) ;
unsigned int Get_Counter (void) ;
```

4.5 servos.h

```
void Init_Servos (unsigned int T) ; // sampling period in ms
float velocity (void) ;  // returns rad/s
void action (float U) ;  // action in Volts

float R (float Wref, float W) ;
// Returns the action in volts
//   Wref : angular velocity reference in rad/s
//   W : actual angular velocity rad/s
```