

Microcontroladores

Semestre: 2021-2

Profesor: Kalun José Lau Gan

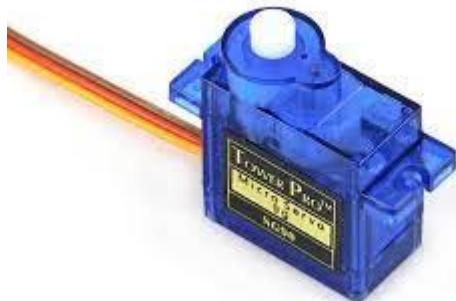
Semana 11: Manejo de un hobby servo

Preguntas previas:

- (levantar y dejar la mano alzada mientras dure la intervención)
- Mi servo gira la vuelta completa. ¿Es normal?
 - No, los servos no giran la vuelta completa, solo 180°

El servo

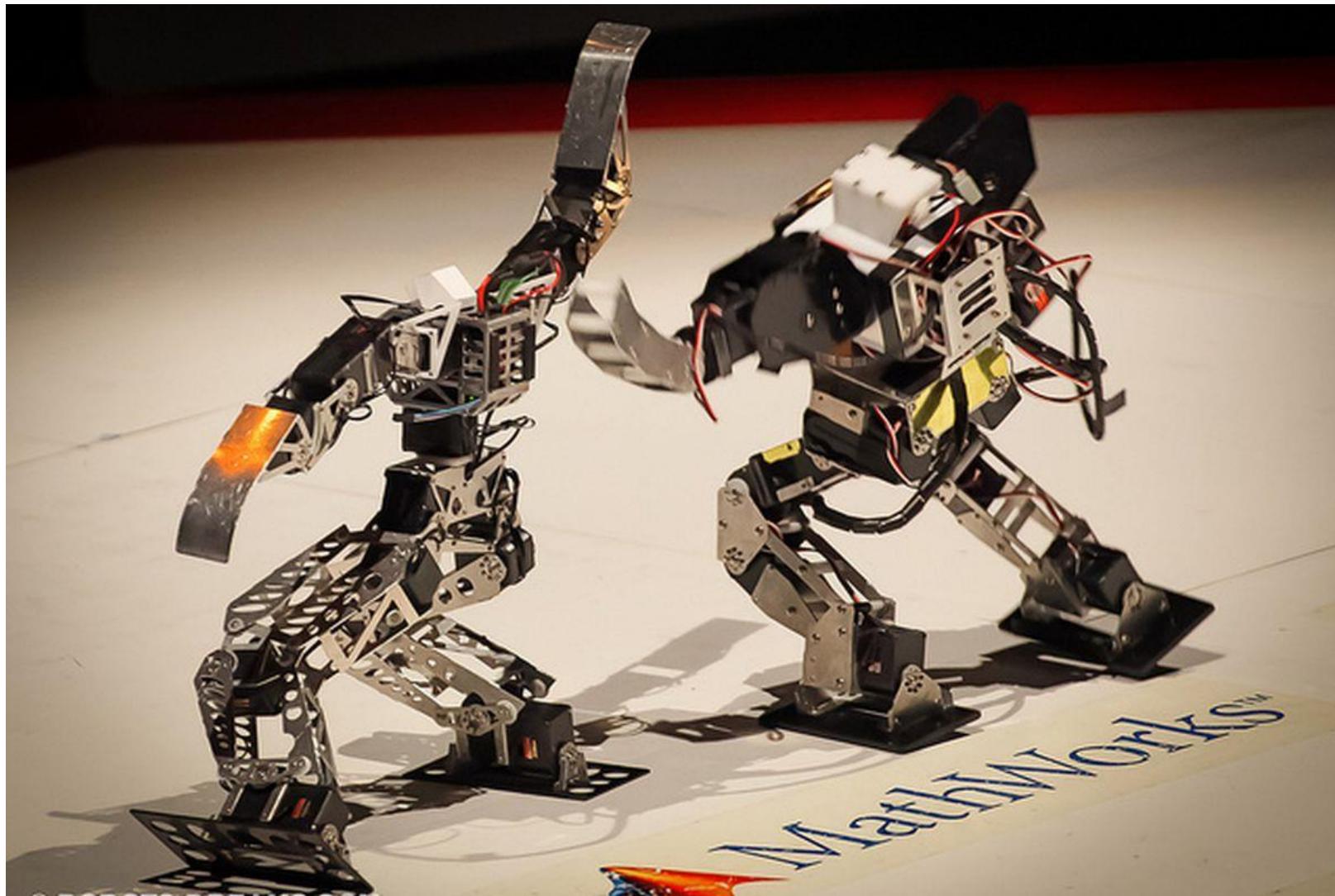
- Empleado comúnmente en hobby para radiocontrol de vehículos terrestres, aéreos y acuáticos, para el control de posición (ángulo)
 - Acelerador, freno, dirección vehicular, alerones (flaps), timón, robótica.
- Son clasificados por: tamaño, torque, velocidad, precisión, tamaño



Servos en el hobby profesional



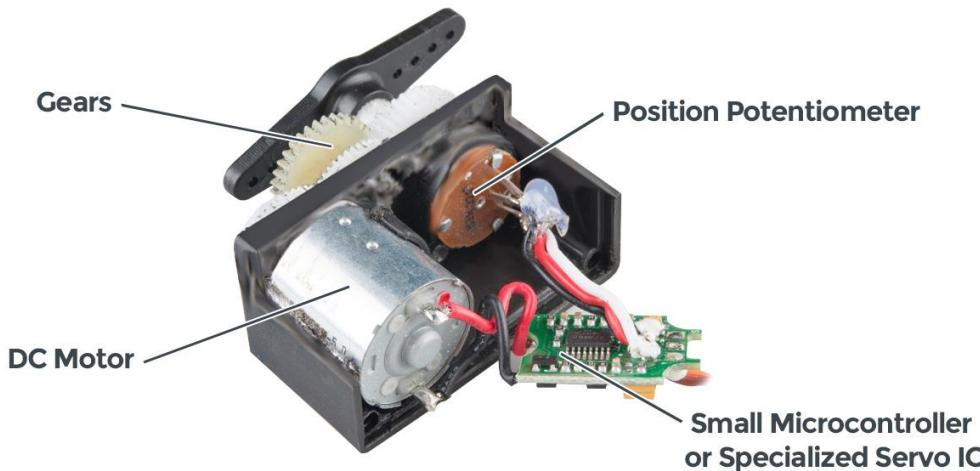
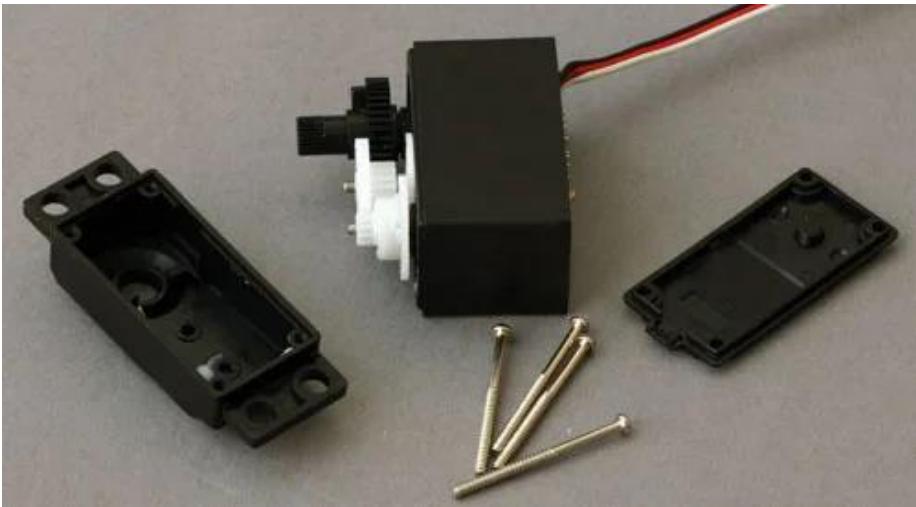
Servos en el hobby profesional





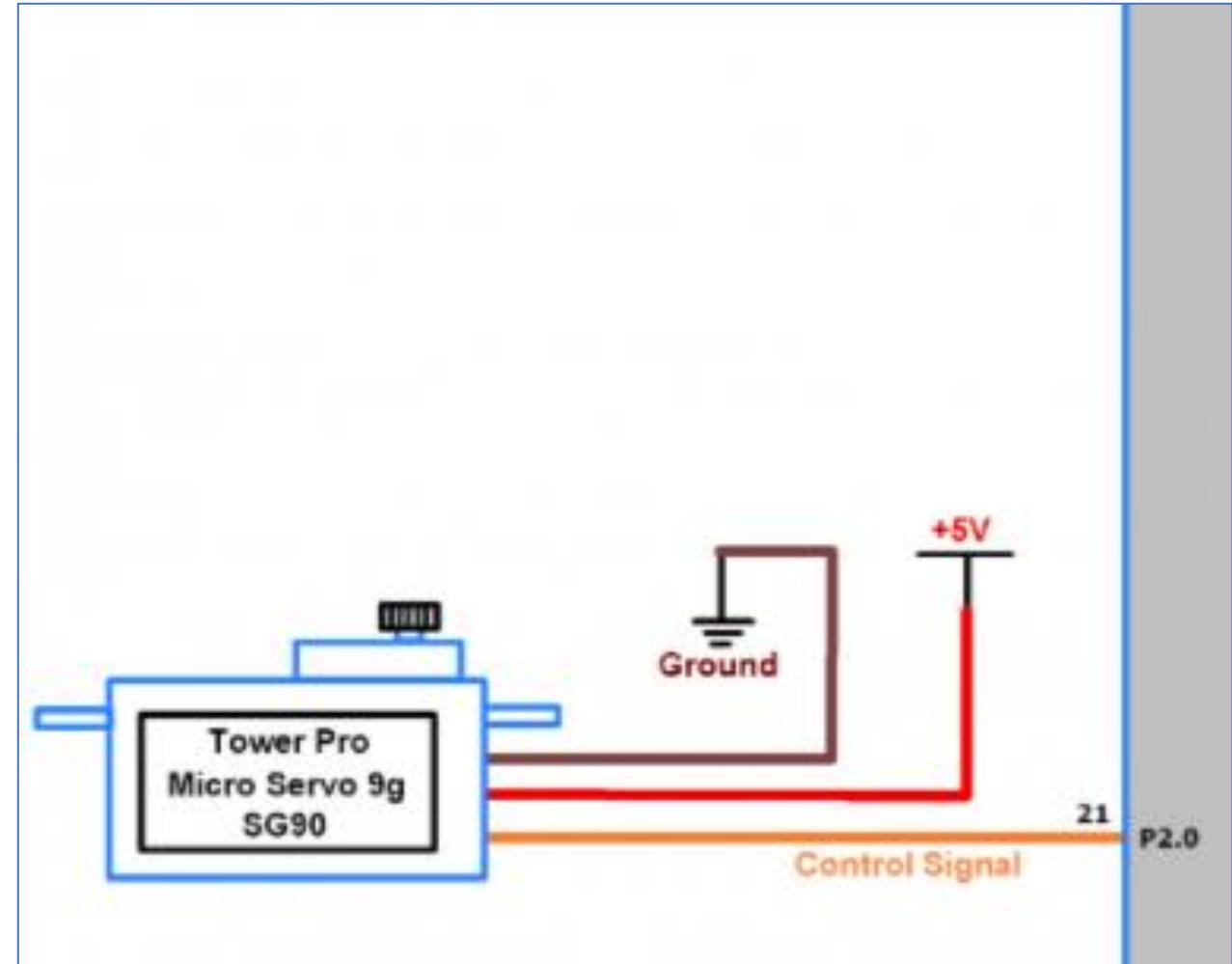
El servo

- Internamente posee un motor realimentado con un potenciómetro y mecanismo de reducción.



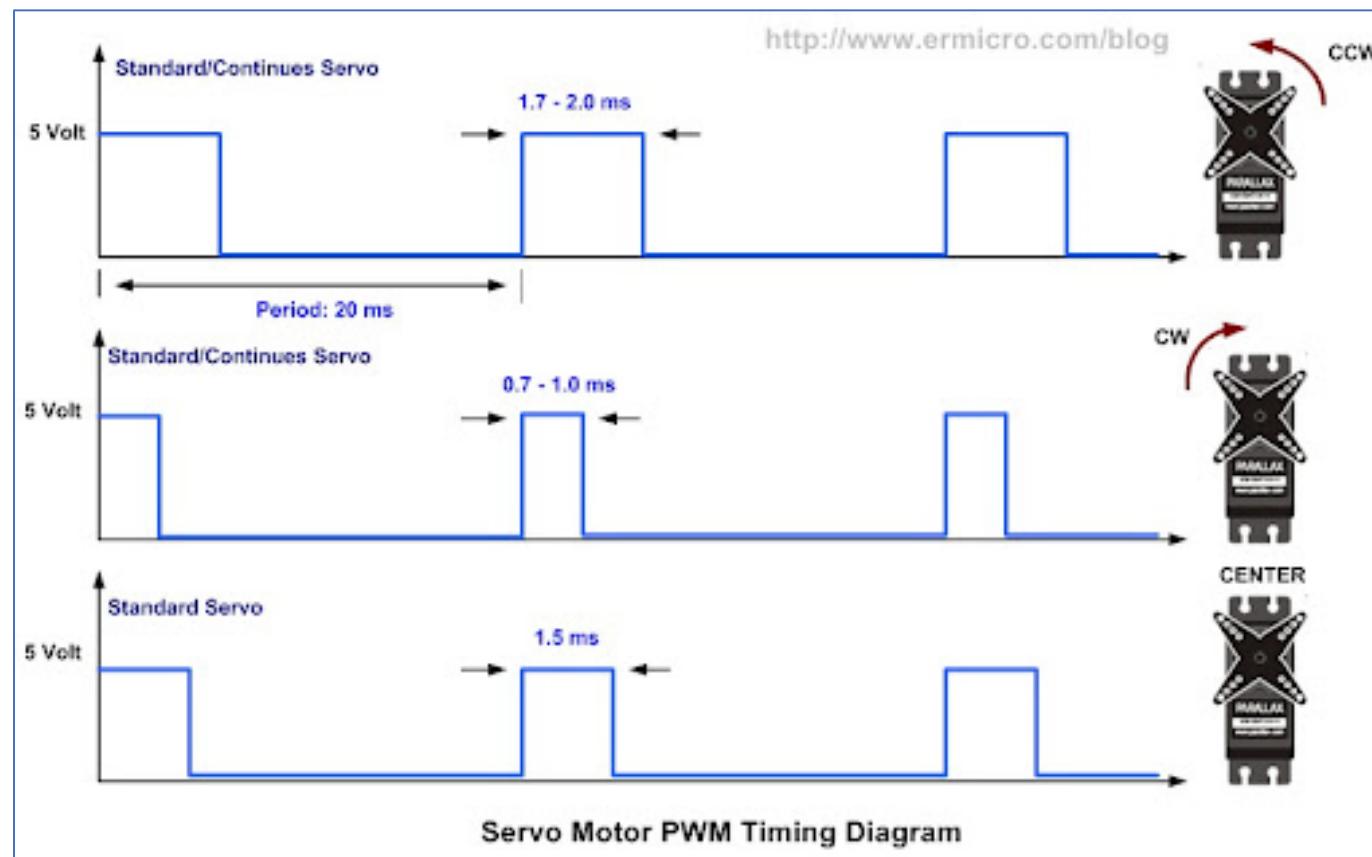
El servo

- Conexión con el microcontrolador:

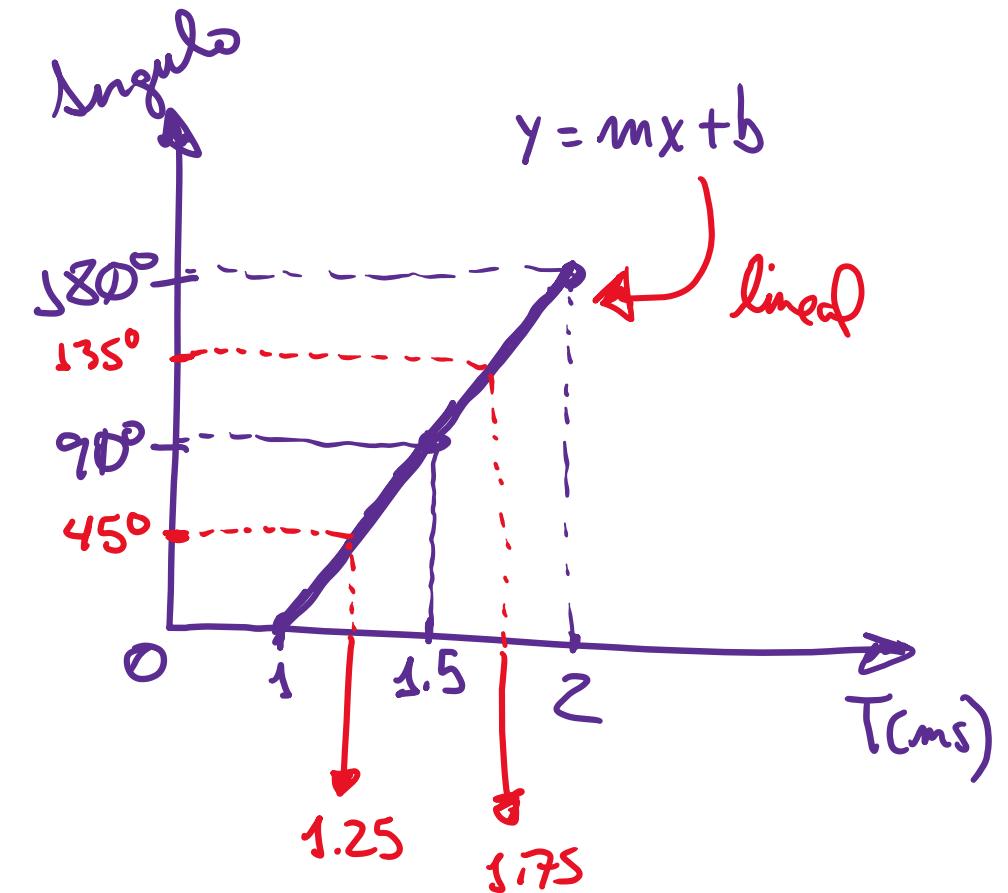


El servo

- Modo de funcionamiento:
 - Tren de pulsos de period 20ms
 - El ancho del pulso (1.0ms – 2.0ms)positivo determinará la posición del eje del servo

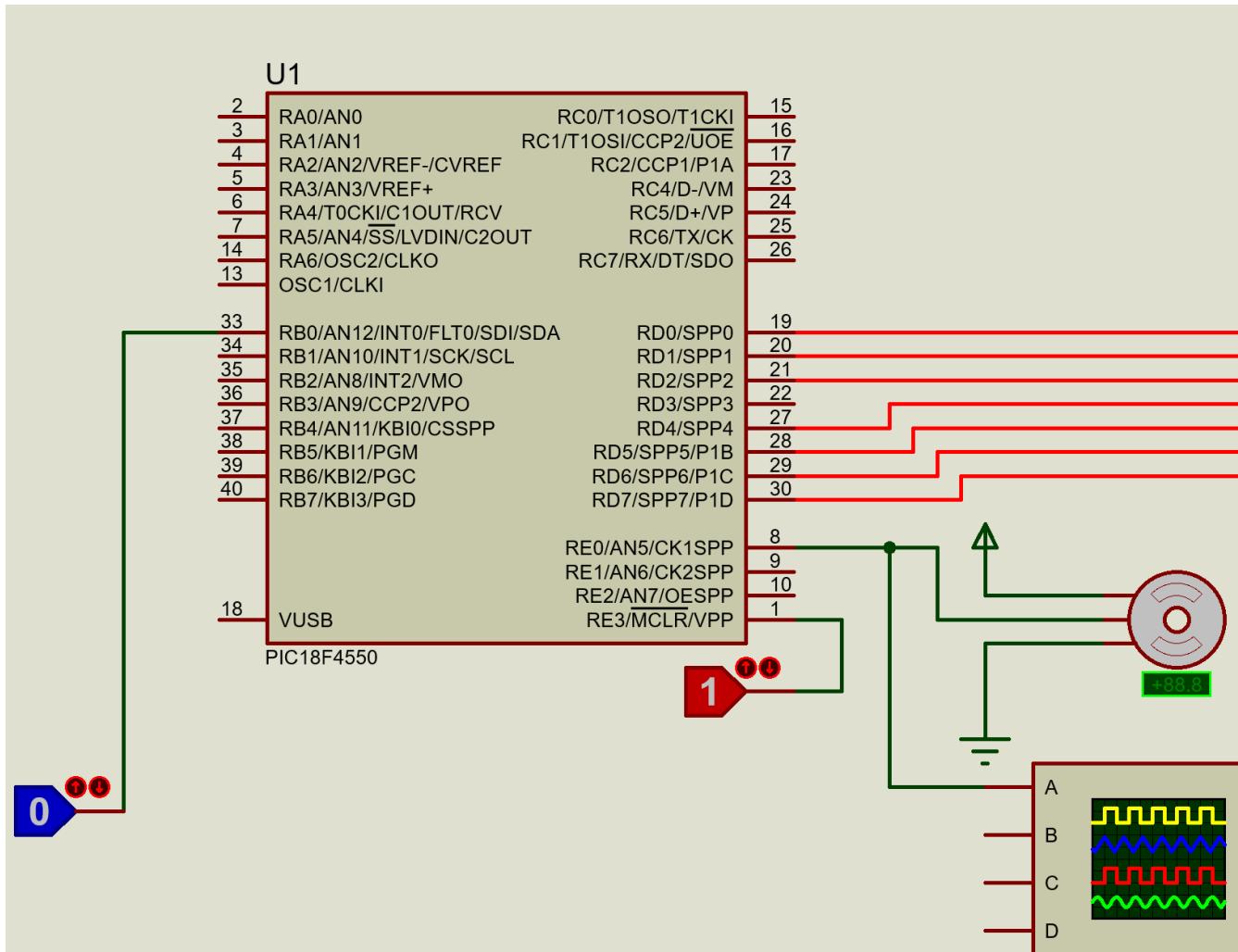


Relación TON vs ángulo del servo:



Circuito de prueba

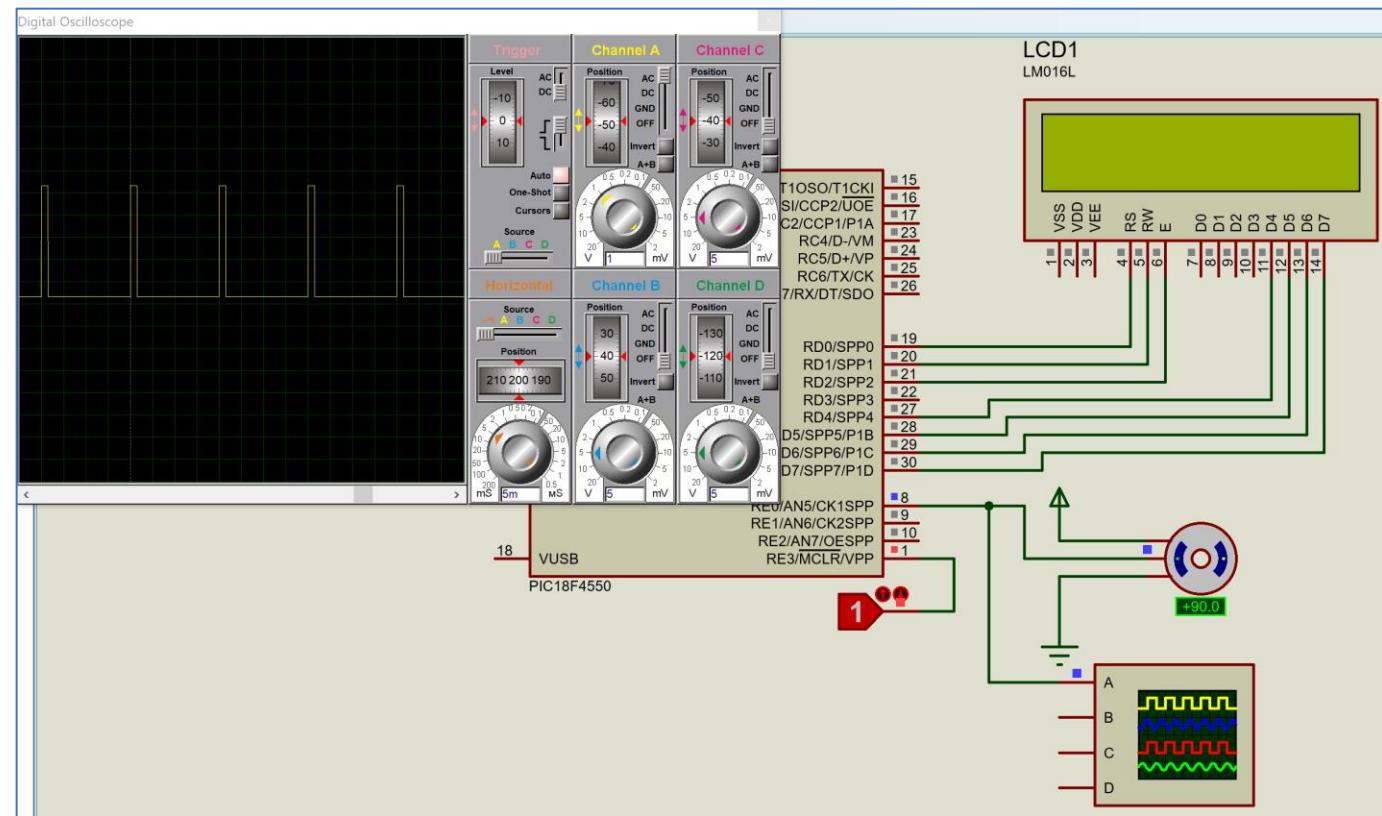
- En Proteus se usará el Motor – PWM Servo



Prueba inicial: Servo a posición media (TON=1.5ms)

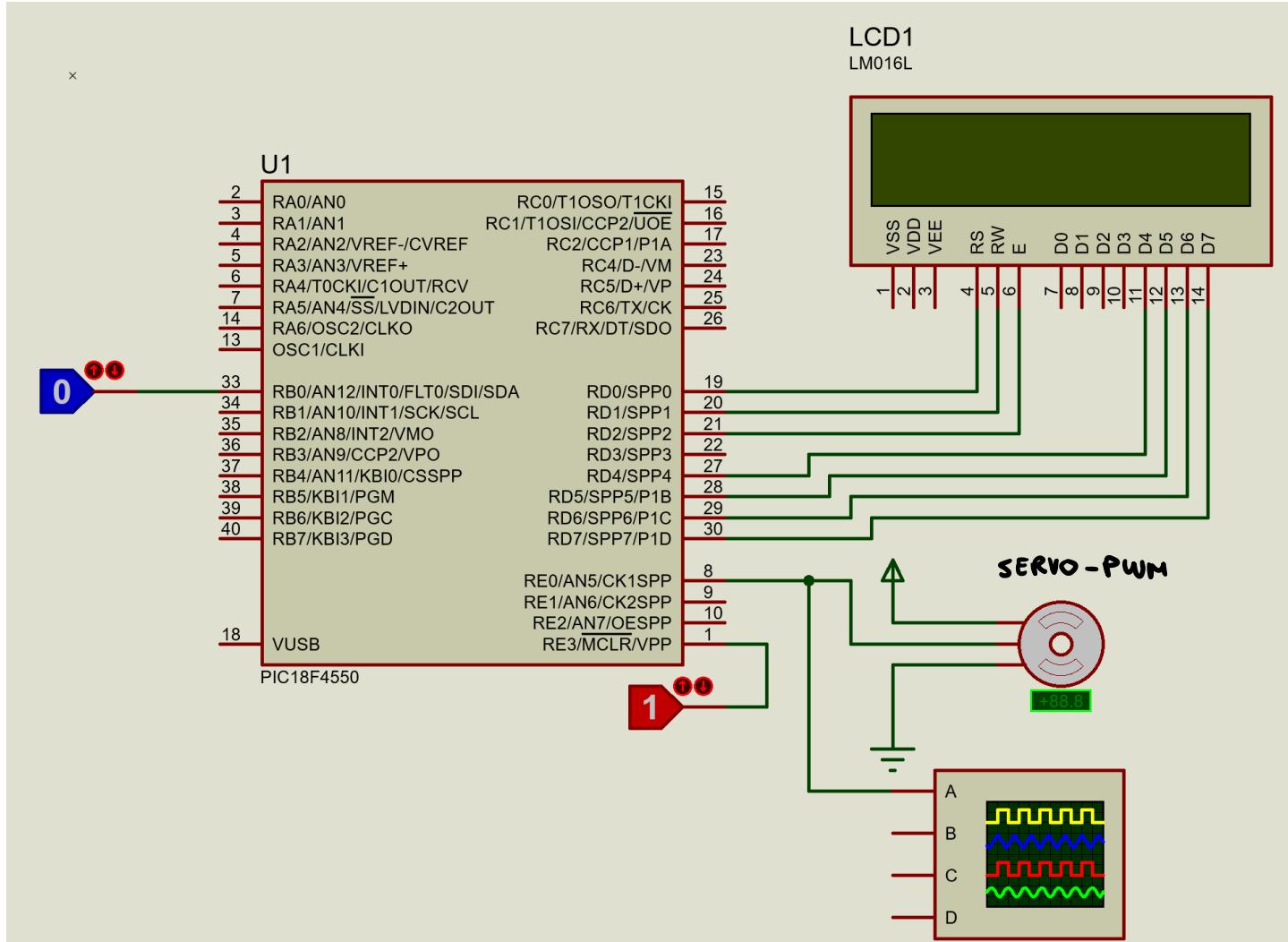
- Se empleará `_delay_us()` para la generación de TON y TOFF de la señal que va al servo

```
1 #include "cabecera.h"
2
3 #include <xc.h>
4
5 #define _XTAL_FREQ 48000000UL
6
7 void configuro(void) {
8     ADCON1 = 0x0F;           //Todos en digital
9     TRISEbits.RE0 = 0;       //RE0 como salida
10
11 void main(void) {
12     configuro();
13     while(1){
14         LAT Ebis. LE0 = 1;
15         _delay_us(1500);
16         LAT Ebis. LE0 = 0;
17         _delay_us(18500);
18     }
}
```



Añadiendo el LCD al ejemplo:

- Se empleará el LCD para visualizar el ángulo actual del servo.



Código ejemplo con LCD y entrada RB0

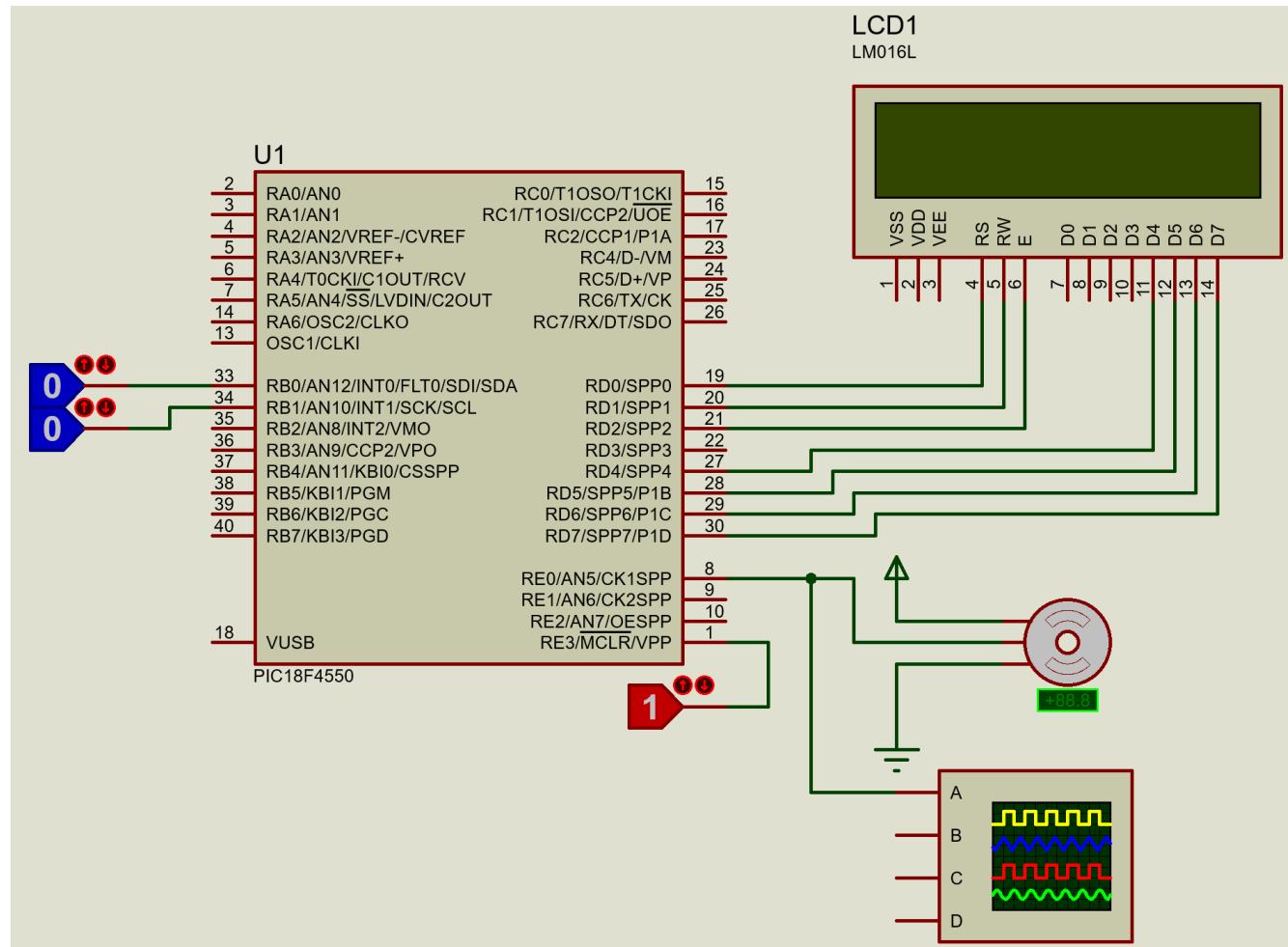
- La entrada RB0 servirá para seleccionar entre el ángulo 45° y 135° del eje del servo.

```
1 #include "cabecera.h"
2 #include "LCD.h"
3 #include <xc.h>
4 #define _XTAL_FREQ 48000000UL
5
6 void configuracion(void){
7     ADCON1 = 0x0F;          //Puertos
8     TRISEbits.RE0 = 0;      //RE0 com
9 }
10
11 void lcd_init(void){
12     TRISD = 0x00;
13     __delay_ms(15);
14     LCD_CONFIG();
15     __delay_ms(15);
16     BORRAR_LCD();
17     CURSOR_HOME();
18     CURSOR_ONOFF(OFF);
19 }
```

```
21 void main(void) {
22     configuracion();
23     lcd_init();
24     POS_CURSOR(1,0);
25     ESCRIBE_MENSAJE("Servomecanismo",14);
26
27     while(1) {
28         if(PORTBbits.RB0 == 1){
29             LATEbits.LE0 = 1;
30             __delay_us(1750);
31             LATEbits.LE0 = 0;
32             __delay_us(18250);
33             POS_CURSOR(2,0);
34             ESCRIBE_MENSAJE("Angulo:135",10);
35             ENVIA_CHAR(0xDF);
36         }
37         else{
38             LATEbits.LE0 = 1;
39             __delay_us(1250);
40             LATEbits.LE0 = 0;
41             __delay_us(18750);
42             POS_CURSOR(2,0);
43             ESCRIBE_MENSAJE("Angulo: 45",10);
44             ENVIA_CHAR(0xDF);
45         }
46     }
47 }
```

Ejemplo: Dos entradas para seleccionar 4 ángulos en el servo

- Se empleará la sentencia switch-case en el programa en XC8 para la selección del ángulo que va a tener el eje del servo.

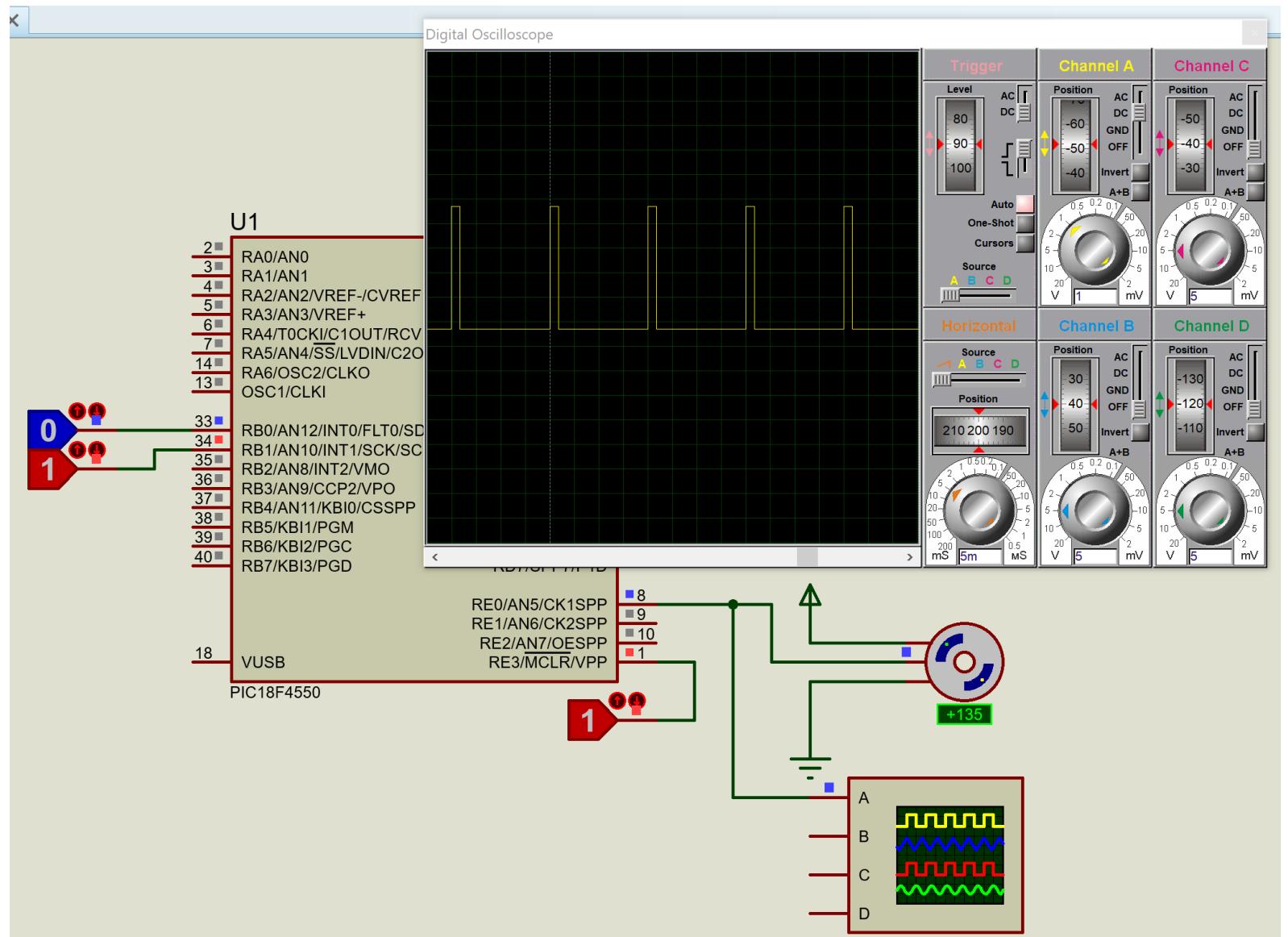


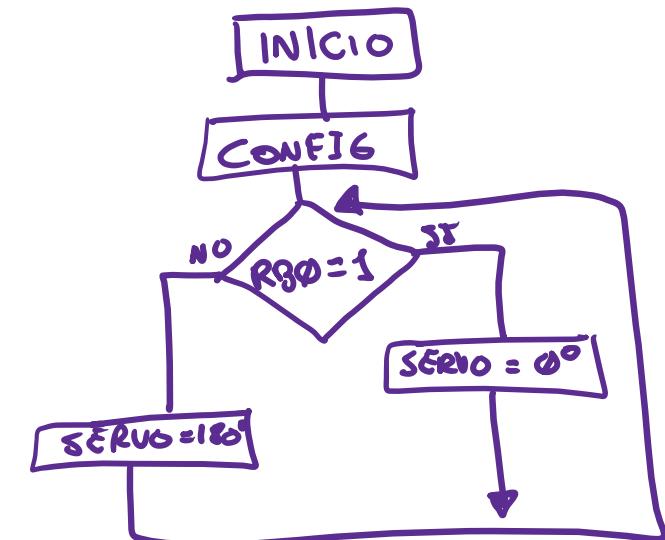
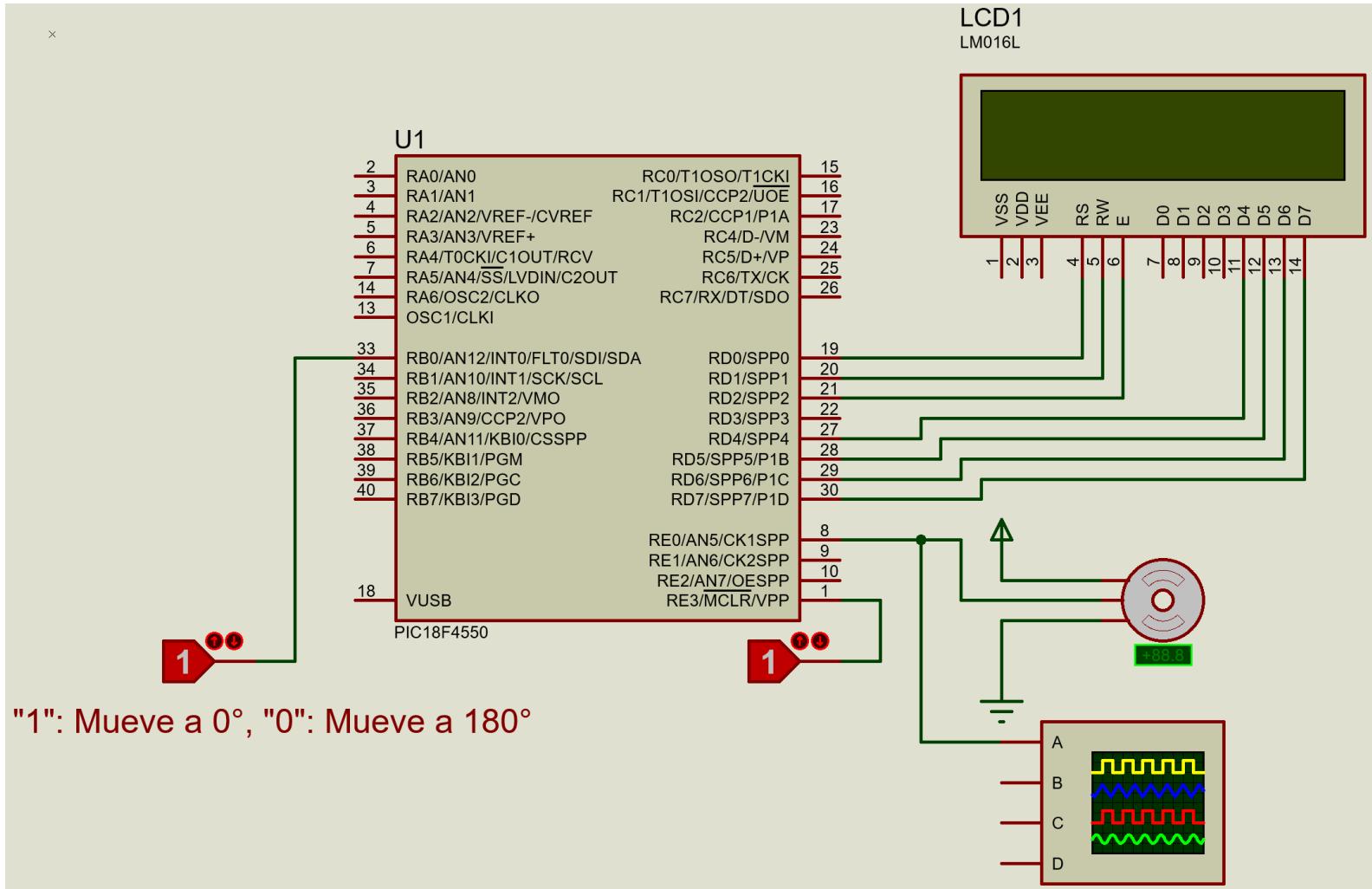
Ejemplo: Dos entradas para seleccionar 4 ángulos en el servo

```
1  #include "cabecera.h"
2  #include <xc.h>
3  #define _XTAL_FREQ 48000000UL
4
5  void configuro(void){
6      ADCON1 = 0x0F;          //Todos en digital
7      TRISEbits.RE0 = 0;     //RE0 como salida
8 }
```

```
10 void main(void) {
11     unsigned char entrada = 0;
12     configuro();
13     while(1){
14         entrada = PORTB & 0x03;
15         switch(entrada){
16             case 0:
17                 LATEbits.LE0 = 1;
18                 __delay_us(1000);
19                 LATEbits.LE0 = 0;
20                 __delay_us(19000);
21                 break;
22             case 1:
23                 LATEbits.LE0 = 1;
24                 __delay_us(1250);
25                 LATEbits.LE0 = 0;
26                 __delay_us(18750);
27                 break;
28             case 2:
29                 LATEbits.LE0 = 1;
30                 __delay_us(1750);
31                 LATEbits.LE0 = 0;
32                 __delay_us(18250);
33                 break;
34             case 3:
35                 LATEbits.LE0 = 1;
36                 __delay_us(2000);
37                 LATEbits.LE0 = 0;
38                 __delay_us(18000);
39                 break;
40         default:
41             break;
42     }
43 }
44 }
```

Simulación



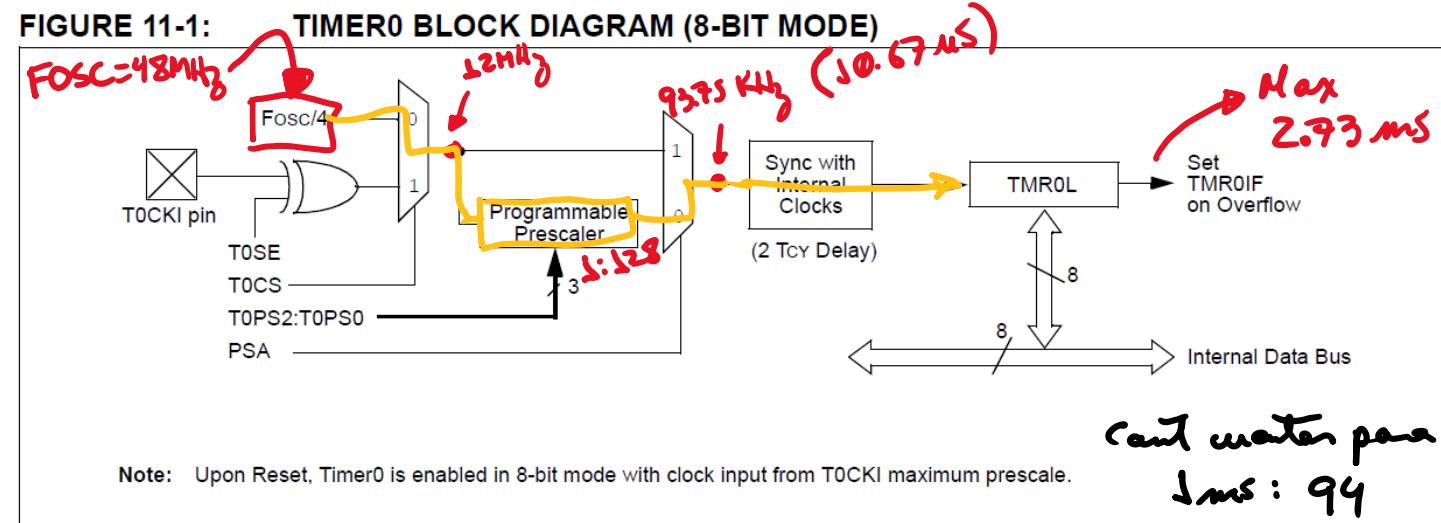
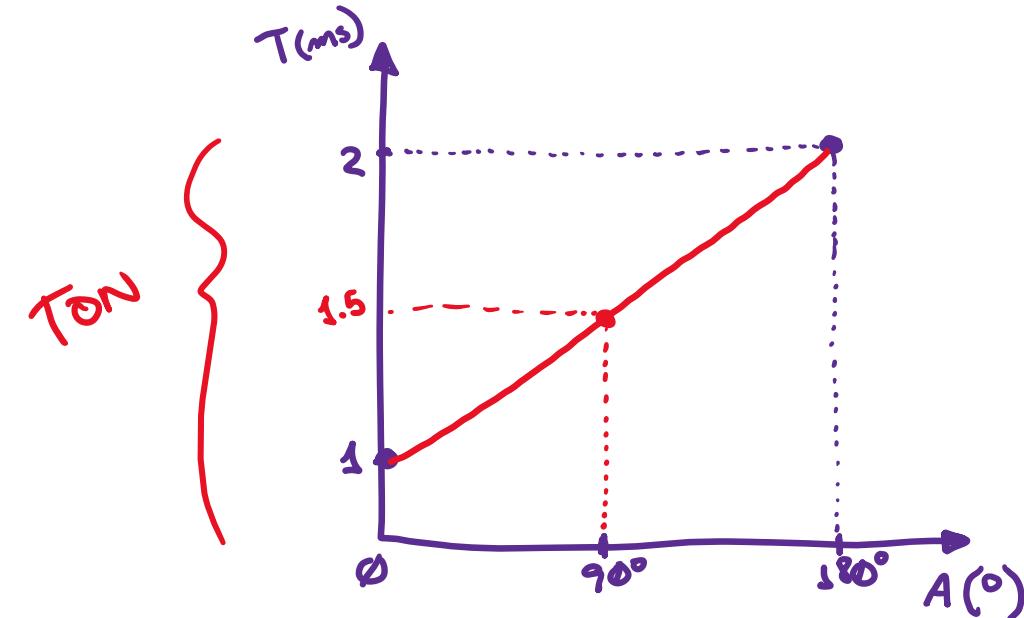


$$\text{Servo } 180^\circ: \\ \overline{\text{TON}} = 2\text{ms}$$

$$\text{Servo } 0^\circ: \\ \text{TON} = 1\text{ms}$$

Trabajando con el Timer0 para manipular el servo

- El Timer0 debe temporizar 1 á 2ms en el TON y 18 á 20ms en TOFF



Cuenta inicial para 1ms: 162
2ms: 188
2ms: 68

Configuración de Timer0 (T0CON)

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	TMR0ON: Timer0 On/Off Control bit 1 = Enables Timer0 0 = Stops Timer0
bit 6	T08BIT: Timer0 8-Bit/16-Bit Control bit 1 = Timer0 is configured as an 8-bit timer/counter 0 = Timer0 is configured as a 16-bit timer/counter
bit 5	T0CS: Timer0 Clock Source Select bit 1 = Transition on T0CKI pin 0 = Internal instruction cycle clock (CLKO)
bit 4	T0SE: Timer0 Source Edge Select bit 1 = Increment on high-to-low transition on T0CKI pin 0 = Increment on low-to-high transition on T0CKI pin
bit 3	PSA: Timer0 Prescaler Assignment bit 1 = Timer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler. 0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.
bit 2-0	T0PS2:T0PS0: Timer0 Prescaler Select bits 111 = 1:256 Prescale value 110 = 1:128 Prescale value 101 = 1:64 Prescale value 100 = 1:32 Prescale value 011 = 1:16 Prescale value 010 = 1:8 Prescale value 001 = 1:4 Prescale value 000 = 1:2 Prescale value

T0CON: 0XC6

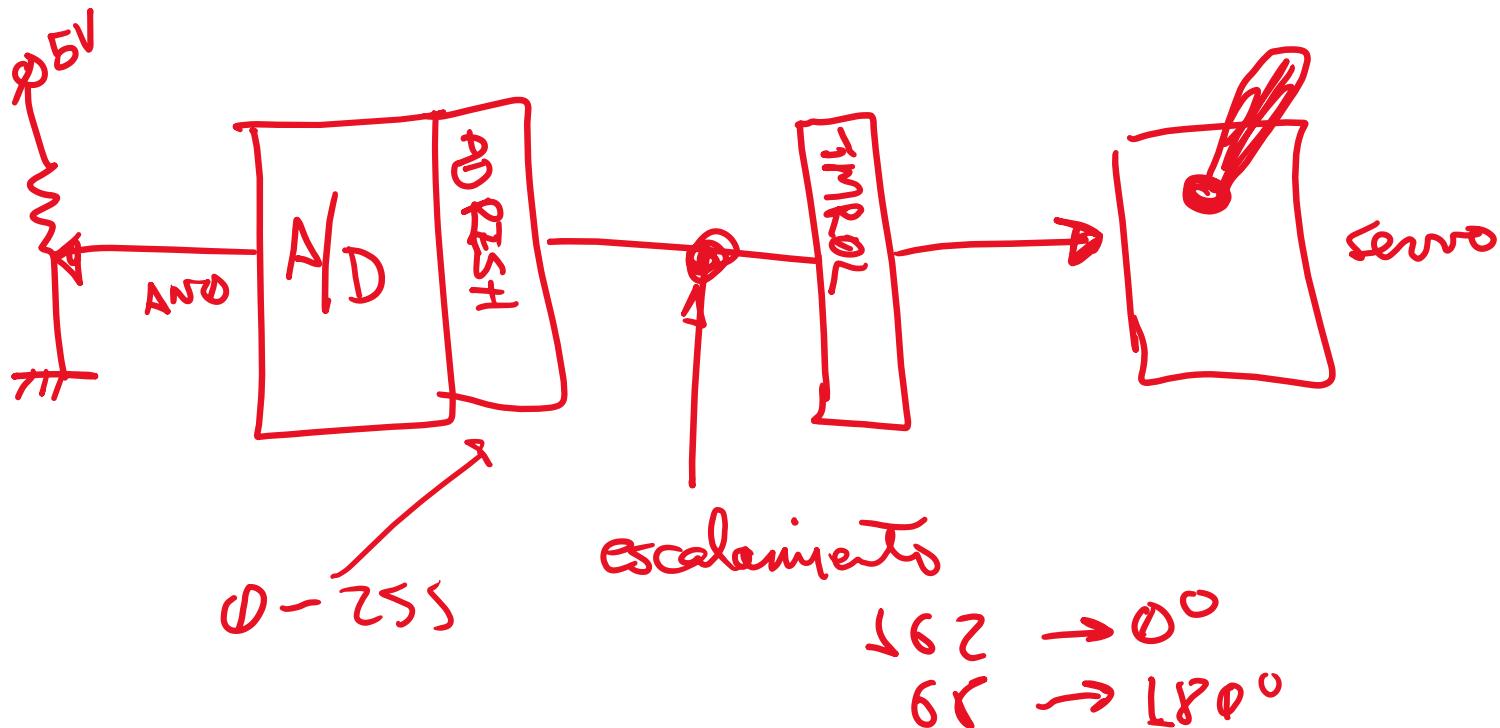
Código en XC8

- Empleando Timer0 para temporizar TON
- TOFF se mantiene con `__delay_us()`, pendiente de cambio para ser usado con Timer0 también.

```
1  #include "cabecera.h"
2  #include <xc.h>
3  #define _XTAL_FREQ 48000000UL
4
5  void configuracion(void){
6      ADCON1 = 0x0F;          //Todos los I/O como digitales
7      TRISEbits.RE0 = 0;     //RE0 como salida
8      T0CON = 0xC6;         //TMR0 ON, PSC 1:128, FOSC/4
9 }
10
11 void main(void) {
12     configuracion();
13     while(1){
14         if(PORTBbits.RB0 == 1){
15             LATEbits.LE0 = 1;
16             TMR0L = 162;
17             while(INTCONbits.TMR0IF == 0);
18             LATEbits.LE0 = 0;
19             __delay_ms(20);
20             INTCONbits.TMR0IF = 0;
21         }
22         else{
23             LATEbits.LE0 = 1;
24             TMR0L = 68;
25             while(INTCONbits.TMR0IF == 0);
26             LATEbits.LE0 = 0;
27             __delay_ms(20);
28             INTCONbits.TMR0IF = 0;
29         }
30     }
31 }
```

Propuesta: Enlace A/D con Timer0 para controlar la posición del servo a través de un potenciómetro en AN-0

- Tener en cuenta la resolución tanto del A/D como del Timer0, asimismo el proceso de escalamiento entre el valor obtenido de AN.0 a ser trasladado a la cuenta inicial de Timer0



Empleando el Timer 3 para manipular el servo 0° y 180°

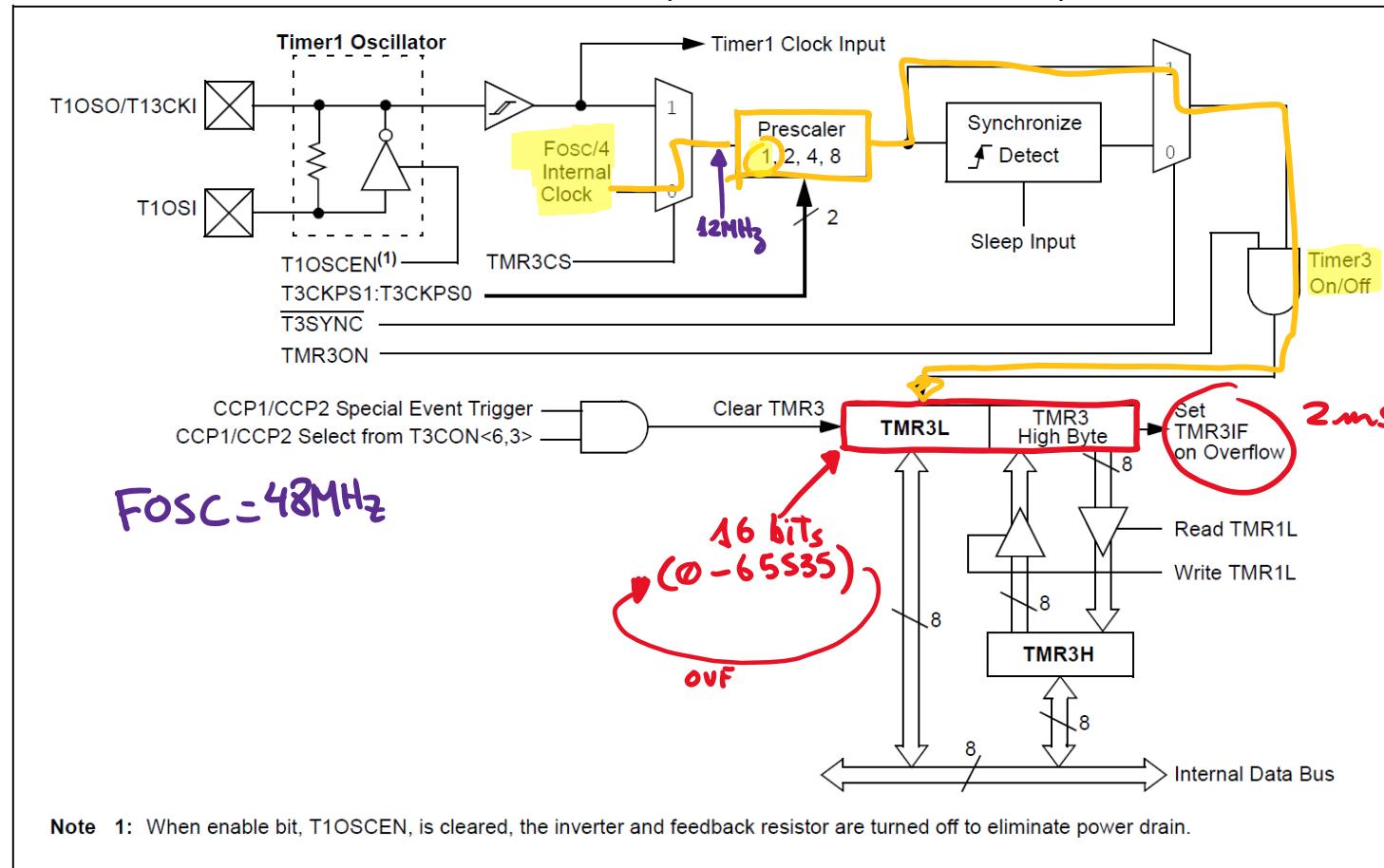
- Periodo de temporización:

- TON entre 1ms y 2ms
- TOFF entre 18ms y 19ms

✓ pendiente

Temporización máxima:
5.46 ms

FIGURE 14-2: TIMER3 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)



Para 2ms:

$$5.46 \text{ ms} - 65536$$

$$2 \text{ ms} - X$$

$$X = \frac{65536}{5.461333\dots}$$

$$X = 24000$$

↑
la cantidad de
cuentos que TMR3
debe de realizar

⇒ Cuenta inicial:

$$41536$$

↑
TMR3N:TMR3L

para 2ms.

Empleando el Timer 3 para manipular el servo

- Periodo de temporización:
 - TON entre 1ms y 2ms
 - TOFF entre 18ms y 19ms
- ✓ pendiente ↗

Pare 1ms:

$$x = \frac{65536(1)}{5.4613373\ldots}$$

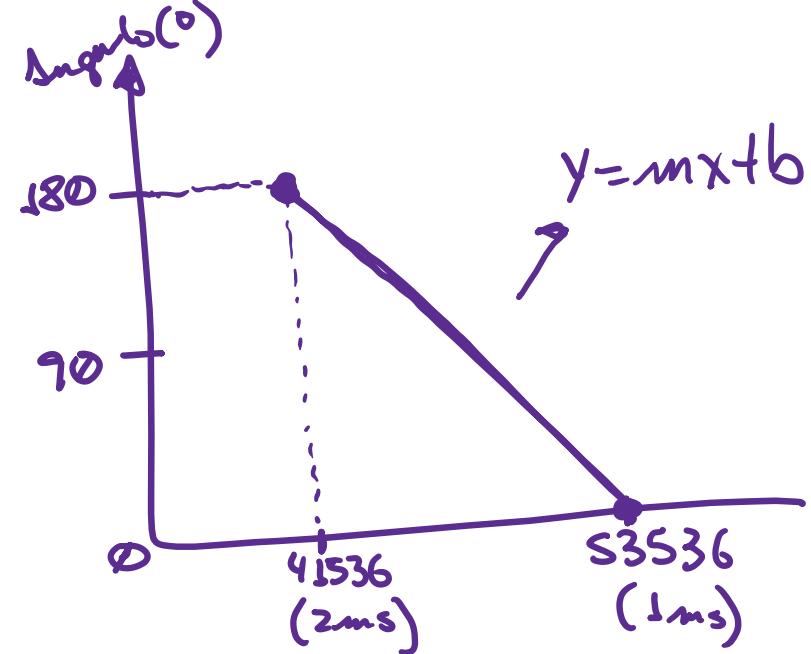
$$x = 12000$$

⇒ cuenta inicial:

$$\begin{array}{r} 65536 - \\ 12000 \\ \hline 53536 \end{array}$$

↑
TMR3H:TMR3L Pare 1ms

Sigulo vs cuenta inicial TMR3



Configuración de T3CON

REGISTER 14-1: T3CON: TIMER3 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON
bit 7							bit 0

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7

RD16: 16-Bit Read/Write Mode Enable bit

- 1 = Enables register read/write of Timer3 in one 16-bit operation
- 0 = Enables register read/write of Timer3 in two 8-bit operations

bit 6, 3

T3CCP2:T3CCP1: Timer3 and Timer1 to CCPx Enable bits

- 1x = Timer3 is the capture/compare clock source for both CCP modules
- 01 = Timer3 is the capture/compare clock source for CCP2;
Timer1 is the capture/compare clock source for CCP1
- 00 = Timer1 is the capture/compare clock source for both CCP modules

bit 5-4

T3CKPS1:T3CKPS0: Timer3 Input Clock Prescale Select bits

- 11 = 1:8 Prescale value
- 10 = 1:4 Prescale value
- 01 = 1:2 Prescale value
- 00 = 1:1 Prescale value

bit 2

T3SYNC: Timer3 External Clock Input Synchronization Control bit

(Not usable if the device clock comes from Timer1/Timer3.)

When TMR3CS = 1:

- 1 = Do not synchronize external clock input
- 0 = Synchronize external clock input

When TMR3CS = 0:

This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1

TMR3CS: Timer3 Clock Source Select bit

- 1 = External clock input from Timer1 oscillator or T13CKI (on the rising edge after the first falling edge)
- 0 = Internal clock (Fosc/4)

bit 0

TMR3ON: Timer3 On bit

- 1 = Enables Timer3
- 0 = Stops Timer3

T3CON = 0x01

Código en XC8:

- Se reemplazó el `_delay_us()` de TON con Timer3
- Pendiente: hacer el mismo procedimiento para TOFF

```
1  #include "cabecera.h"
2  #include "LCD.h"
3  #include <xc.h>
4  #define _XTAL_FREQ 48000000UL
5
6  void configuracion(void){
7      ADCON1 = 0x0F;          //Puertos E/S como digitales
8      TRISEbits.RE0 = 0;     //RE0 como salida
9      T3CON = 0x01;          //Configuracion Timer3
10 }
11
12 void lcd_init(void){
13     TRISD = 0x00;
14     __delay_ms(15);
15     LCD_CONFIG();
16     __delay_ms(15);
17     BORRAR_LCD();
18     CURSOR_HOME();
19     CURSOR_ONOFF(OFF);
20 }
```

```
22 void main(void) {
23     configuracion();
24     lcd_init();
25     POS_CURSOR(1,0);
26     ESCRIBE_MENSAJE("Servomecanismo",14);
27     while(1){
28         if(PORTBbits.RB0 == 1){
29             LATEbits.LE0 = 1;
30             TMR3H = 0xA2;
31             TMR3L = 0x40;           //Angulo 180° Cuenta inicial 41536
32             while(PIR2bits.TMR3IF == 0);
33             LATEbits.LE0 = 0;
34             __delay_us(18250);
35             POS_CURSOR(2,0);
36             ESCRIBE_MENSAJE("Angulo:180",10);
37             ENVIA_CHAR(0xDF);
38             PIR2bits.TMR3IF = 0;
39         }
40         else{
41             LATEbits.LE0 = 1;
42             TMR3H = 0xD1;
43             TMR3L = 0x20;           //Angulo 0° Cuenta inicial 53536
44             while(PIR2bits.TMR3IF == 0);
45             LATEbits.LE0 = 0;
46             __delay_us(18750);
47             POS_CURSOR(2,0);
48             ESCRIBE_MENSAJE("Angulo: 0",10);
49             ENVIA_CHAR(0xDF);
50             PIR2bits.TMR3IF = 0;
51         }
52     }
53 }
54 }
```

Cuestionario:

- Ya que se ha analizado el Timer0 y el Timer3 como fuente de tiempo para obtener los periodos de un servo. Es posible manipular dos servos, uno con el Timer0 y otro con el Timer3 junto con el manejo adecuado de las interrupciones.

Fin de la sesión

- Destinar una hora el sábado y una hora el domingo para repasar el curso.