

**Universidad Mondragón México**

**IM 8vo Semestre**

**Materia: Microcontroladores**

**Investigación: Prácticas PIC**

**Fecha: 19/03/2020**

**Alumnos:**

**José Andrés Aguilar Corona**

**Daniel Castellanos Martínez**

**Yael Iván López Rivera**

**Alberto Morales Linares**

Contenido

Introducción………………………………………………………………………………...……….3

[Desarrollo 4](#_Toc32772347)

[Resultados 7](#_Toc32772348)

[Conclusiones 8](#_Toc32772349)

[Referencias 8](#_Toc32772350)

# 

# Introducción

La programación se le conoce como el conjunto de instrucciones consecutivas y ordenadas que llevan a cabo indicaciones o instrucciones a seguir por la computadora con el objetivo de ejecutar una actividad o tarea, en este caso el propósito de estas prácticas es trabajar con motores a pasos, convertidor ADC, display LCD, motor DC, y servomotores. Para el desarrollo de estas prácticas se hará su respectivo código en el programa MPLAB y se hará la simulación con Proteus.

Un convertidor ADC es un dispositivo electrónico con un sistema que transforma señales analógicas en señales digitales mediante los procesos de muestreo, la cuantificación y la codificación. En la siguiente imagen 1.1 podemos observar dicho proceso.

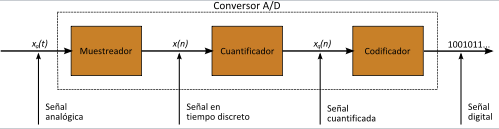


Imagen 1 Diagrama de bloques ADC

Como podemos observar en la imagen 1.1, el proceso comienza con el muestreador, que consiste en tomar muestras periódicas de la amplitud de onda. La velocidad con que se toma esta muestra, es decir, el número de muestras por segundo es lo que se conoce como frecuencia de muestreo. Después en la parte del cuantificador, se refiere a medir el nivel de voltaje de cada una de las muestras y esta consiste en asignar un margen de valor de una señal analizada a un único nivel de salida. Por último, el codificador consiste en traducir los valores obtenidos durante la cuantificación al código binario.

El microcontrolador utilizado tiene un ADC de 10 bits y sólo puede muestrear hasta 5V por lo que una medición del ADC de 1024 equivale a 5V medidos.

Una pantalla LCD es un módulo de pantalla electrónica que utiliza cristal líquido para producir una imagen visible. La pantalla LCD de 16x2 es un módulo muy básico. El 16x2 se traduce en una pantalla de 16 caracteres por línea en 2 de esas líneas. En esta pantalla, cada carácter se muestra en una matriz de 5x7 píxeles.

Esta pantalla se comunica de forma paralela con 8 pines de datos y se puede configurar para sólo usar 4. A través de estos pines podemos mandar comandos y datos para elegir que se va a mostrar en la pantalla

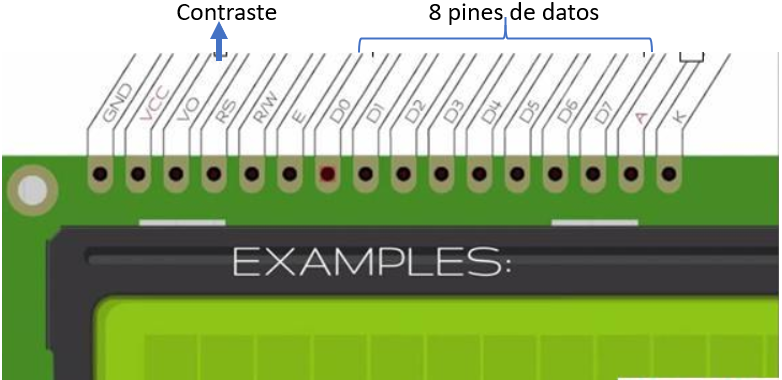


Imagen 2 Pines LCD

El motor DC, es un componente que convierte la energía eléctrica en mecánica, provocando un movimiento rotatorio y se compone principalmente de dos partes, un estator que da soporte mecánico al aparato y se encuentran los polos, que pueden ser de imanes permanentes o devanados con hilo de cobre sobre núcleo de hierro. La otra parte es el rotor es generalmente de forma cilíndrica, también devanado y con núcleo, al que llega la corriente mediante dos escobillas. En la imagen 1.3 podemos observas dichas partes mencionadas.



Imagen 3 Diseño de un motor

El puente H es un circuito electrónico que permite a un motor eléctrico DC girar en ambos sentidos. Se puede construir con 4 transistores o switches

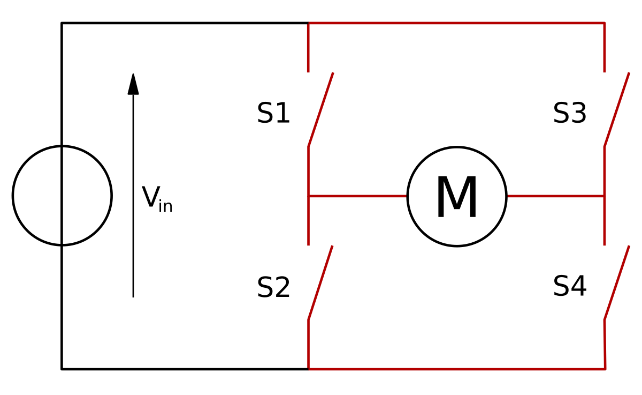


Imagen 4 Puente H

El servomotor es un [motor eléctrico](https://www.areatecnologia.com/EL%20MOTOR%20ELECTRICO.htm) al que podemos controlar tanto la velocidad, como la posición del eje que gira, es decir, que al invertir la polaridad de este, cambia su sentido de giro al moverse el eje unos grados estipulados. Este se compone de un sistema mixto de piezas electromecánicas y electrónicas. El motor y un conjunto de engranajes es la parte electromecánica y la parte electrónica es el circuito de control. En la imagen 1.4 podemos encontrar los componentes que forman a un servomotor.

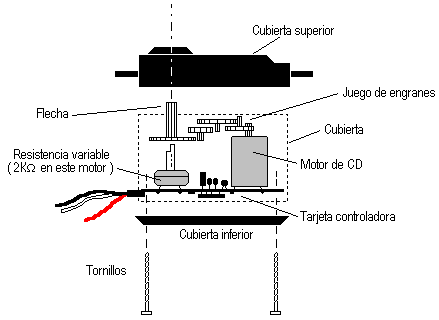


Imagen 5 Construcción servomotor

El servomotor se puede controlar mandando una señal en su pin sig. De periodo de 20ms y la señal en alto se debe mantener .5ms para obtener un ángulo de 0 grados y 2.5 para obtener un ángulo de 180 grados.

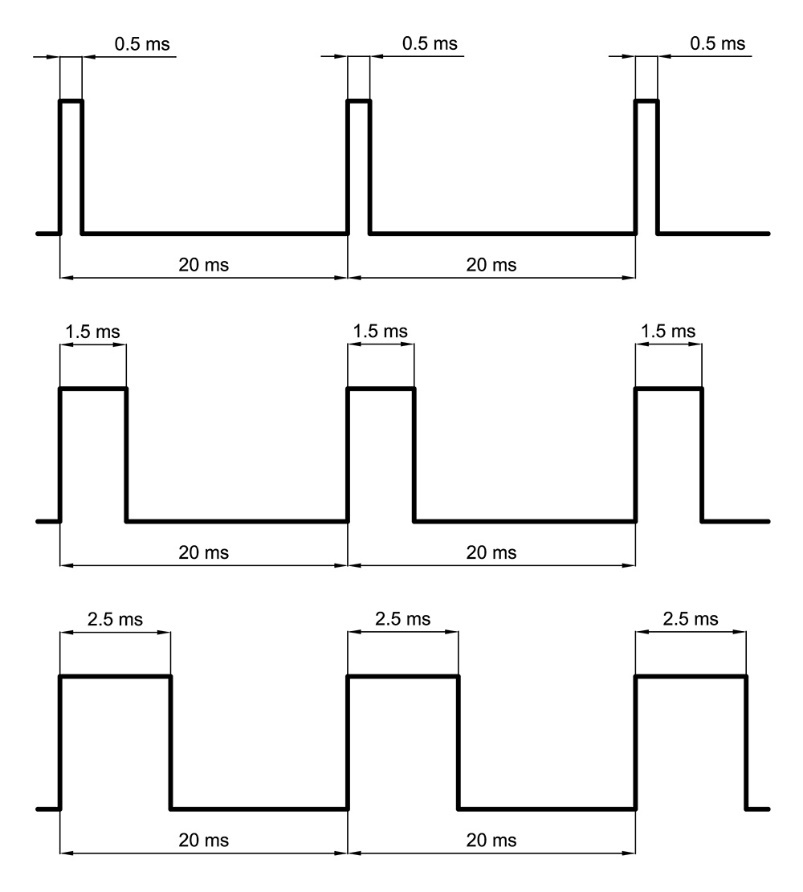


Imagen 6 PWM Servomotor

El motor a pasos es un dispositivo electromecánico que convierte una serie de pulsos eléctricos en desplazamientos angulares, lo que significa que es capaz de girar una cantidad de grados dependiendo de sus entradas de control. Ahora hablando de su funcionamiento se basa en un estator construido por varios bobinados en un material ferromagnético y un rotor que puede girar libremente en el estator. Un ejemplo del motor a pasos y una conexión clásica es el de la imagen 1.5 que veremos a continuación.

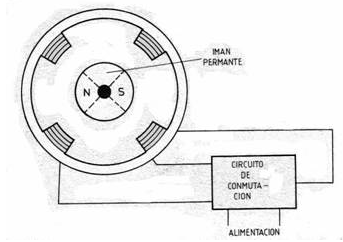


Imagen 7 Motor a pasos

# Desarrollo

El desarrollo de estas prácticas consiste en el control de los motores sobre un código proporcionado visto en clase para ver su comportamiento y el manejo de estos, de igual manera se trabajará con el conversor ADC y el display LCD para hacer una interacción de su funcionamiento. Todo esto, con el fin de analizar todas las funciones utilizadas con relación al código creado y su comportamiento con lo programado. Todo el código, fotos, video y simulaciones de estas prácticas estarán en el siguiente repositorio de github <https://github.com/Daniel289728/ClaseMicros>.

**ADC**

**Pseudocódigo**:

Analizando el código deducimos que el pseudocódigo para el conversor ADC:

1. Inicializar LCD, módulo ADC y configurar LEDs
2. Leer ADC
3. Imprimir lectura en LCD
4. Actualizar los LEDs

A continuación, se presenta el código desarrollado en MPLAB

**Código para practica con ADC**:

#include "Alteri.h" /\* librería especifica de tarjeta de desarrollo

\* usada \*/

#include "xlcd.h" /\* librería para LCD \*/

#include <stdio.h>

#include <stdint.h>

char strResultado[5]; /\* C-Lique Sting para almacenar el resultado de la

\* conversión \*/

/\* Prototipo de funciones \*/

void initADC(void); /\* Función para inicializar el ADC \*/

void initLCD(void); /\* Función para inicializar la LCD \*/

void initLED(void); /\* Función para inicializar LED \*/

uint16\_t conversionADC(void); /\* Función que lee el ADC \*/

void ChecarLEDs(uint16\_t res); /\* Función para cambiar LEDs \*/

void main(void)

{

initLCD();  
 initADC();  
 initLED();

while(1)

{  
 uint16\_t resultado = conversionADC(); /\* Leer el ADC \*/

sprintf(strResultado,"%04u",resultado); /\* Poner la lectura de

\* ADC en el string \*/

WriteCmdXLCD(NEXT\_LINE + 5);

putsXLCD(strResultado); /\* Poner el string en la LCD \*/

ChecarLEDs(resultado); /\* Actualizar los LEDs \*/

delay\_ms(200);

}  
}

void initLED(void)

{

TRISA = 0;  
 TRISB = 0;  
 LATA = 0;  
 LATB = 0;  
}

void initADC(void) /\* función que inicializa el módulo ADC del PIC \*/

{

ADCON0 = 0b00000001; /\* Habilitar ADC (bit 0) Empezar la conducción

\* (bit 1) \*/

ADCON1 = 0b00001110; /\* Elegir AN0 (bits 5-2) \*/

ADCON2 = 0b10111100; /\* Declarar RA0 como pin analógico (bits 0-3)\*/  
}

void initLCD(void) /\*función que inicializa la pantalla LCD \*/

{

OpenXLCD(FOUR\_BIT & LINES\_5X7); /\* Iniciamos LCD.- \*/

WriteCmdXLCD(SHIFT\_DISP\_LEFT); /\* Nos aseguramos incremento de

dirección, display fijo \*/

WriteCmdXLCD(DON); /\* Encendemos LCD \*/

WriteCmdXLCD(BLINK\_OFF); /\* Apagamos parpadeo de cursor \*/

putrsXLCD("Convertidor AD:");  
}

uint16\_t conversionADC(void)/\* Función que realiza la conversión

Analógica - Digital \*/

{

unsigned int resultado;  
 ADCON0bits.GO\_DONE = 1;/\* Inicia la conversión\*/

while (ADCON0bits.GO\_DONE == 1); /\* Espero a que se complete \*/

resultado = (ADRESH<<8) + ADRESL; /\*Formatea el resultado a 16 bits\*/

return resultado; /\* Devuelve resultado \*/  
}

void ChecarLEDs(uint16\_t res)/\* La función es mas cara los bits del

resultado y prende los LEDs \*/

{

LATAbits.LA1 = (res&(1<<9)) == 0 ? 0 : 1;  
 LATAbits.LA2 = (res&(1<<8)) == 0 ? 0 : 1;  
 LATAbits.LA3 = (res&(1<<7)) == 0 ? 0 : 1;  
 LATAbits.LA4 = (res&(1<<6)) == 0 ? 0 : 1;  
 LATAbits.LA5 = (res&(1<<5)) == 0 ? 0 : 1;  
 LATBbits.LB0 = (res&(1<<4)) == 0 ? 0 : 1;  
 LATBbits.LB1 = (res&(1<<3)) == 0 ? 0 : 1;  
 LATBbits.LB2 = (res&(1<<2)) == 0 ? 0 : 1;  
 LATBbits.LB3 = (res&(1<<1)) == 0 ? 0 : 1;

LATBbits.LB4 = (res&(1<<0)) == 0 ? 0 : 1;  
}  
void DelayFor18TCY(void){  
 \_delay(240);  
}  
void DelayPORXLCD(void){  
 delay\_ms(15);  
}  
void DelayXLCD(void){  
 delay\_ms(5);  
}

**Resultados**

Podemos ver como tanto en la simulación como en físico la pantalla da el resultado de la lectura del ADC de un potenciómetro y los LEDs muestran el valor en binario.

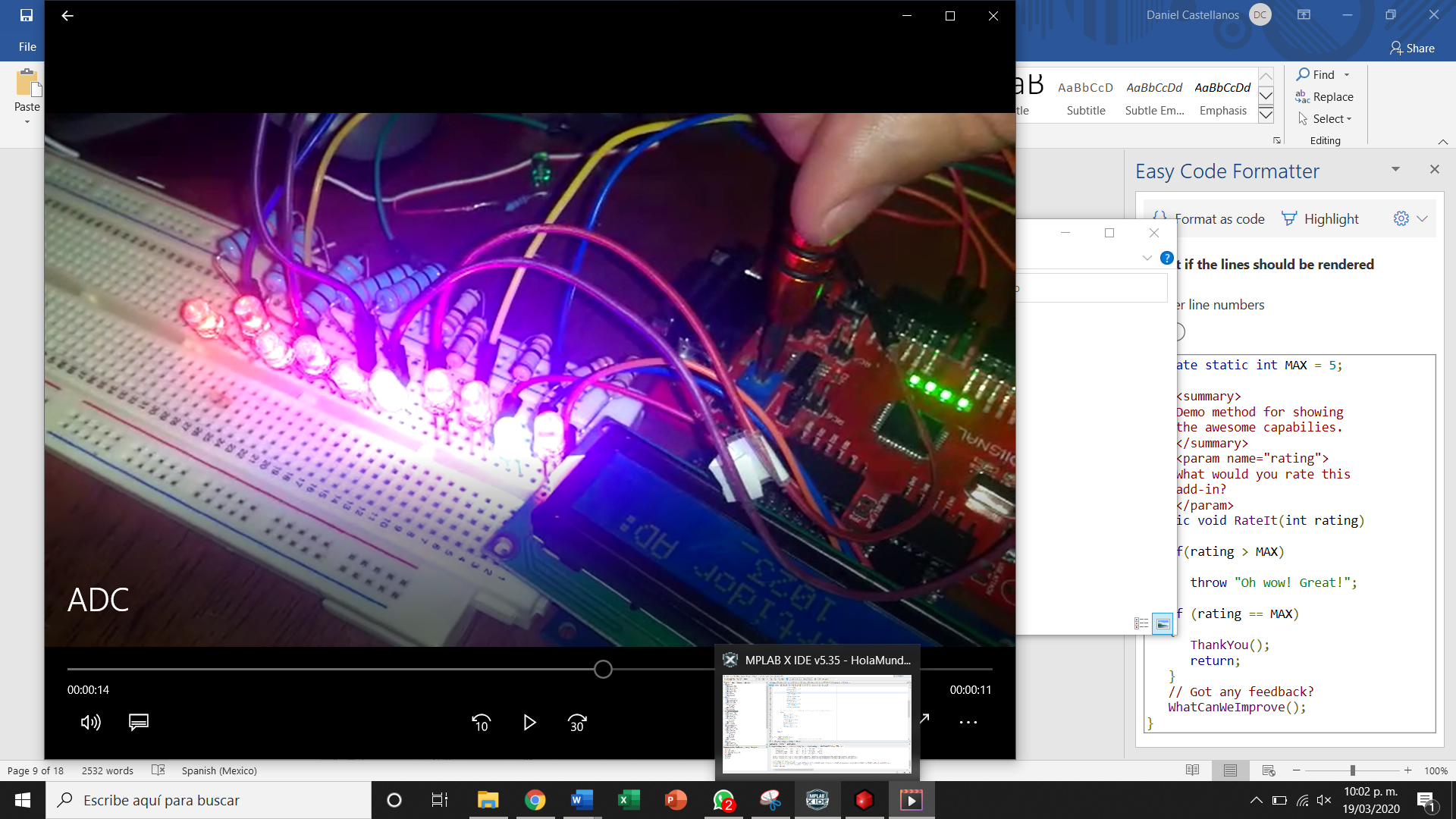


Imagen 8 ADC Físico

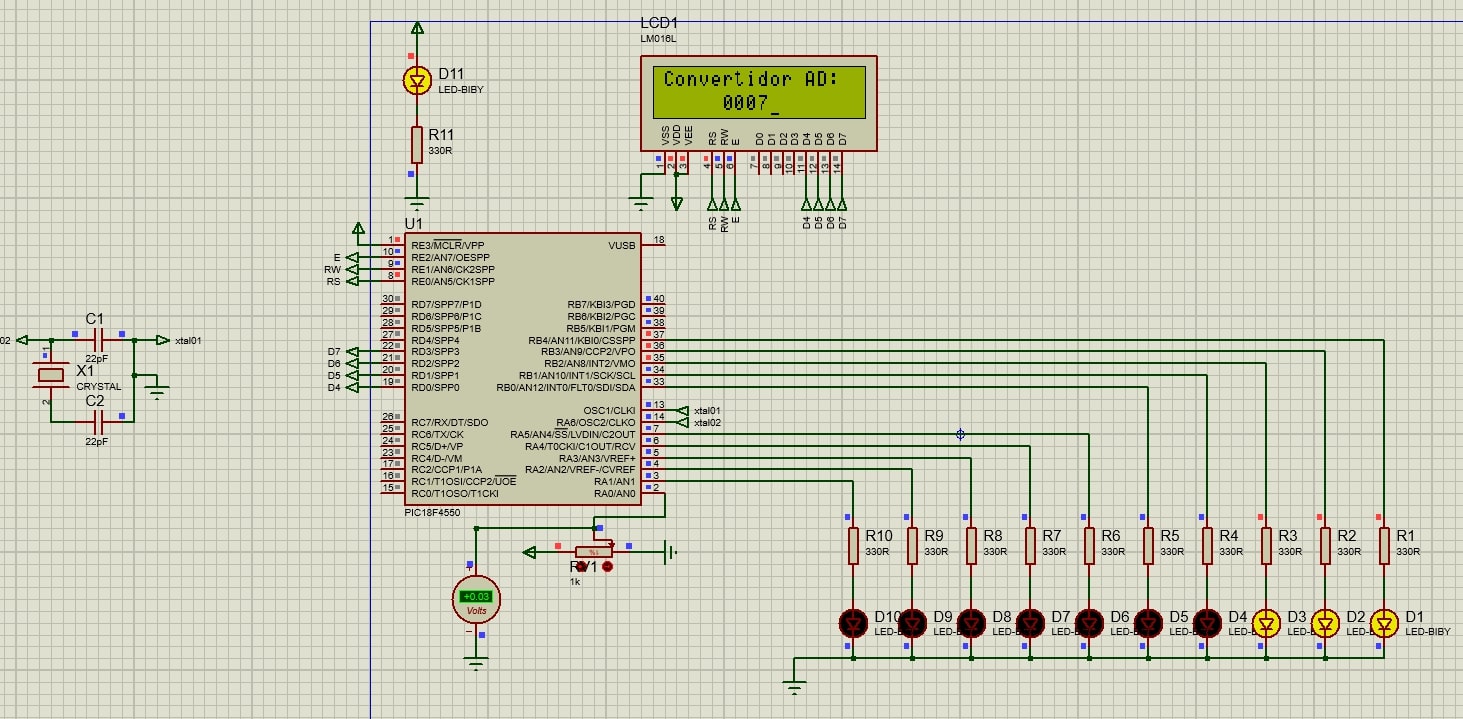


Imagen 9 ADC Simulación

**LCD**

**Pseudocódigo**

1. Inicializar LCD
2. Mandar caracteres

**Código para practica con LCD:**

#include "Alteri.h"/\*Libreria espcífica de tarjeta de desarrollo usada \*/

#include "xlcd.h"/\* Libreria para LCD \*/

void inicializacionLCD(void);/\*Función para inicializar LCD \*/

void main (void)

{

inicializacionLCD();

putrsXLCD(" HOLA MUNDO");

WriteCmdXLCD(NEXT\_LINE);

putrsXLCD(" ADIOS MUNDO");

while(1);

}

void inicializacionLCD(void)

{

OpenXLCD(FOUR\_BIT & LINES\_5X7); /\* Iniciamos LCD.-\*/

WriteCmdXLCD(SHIFT\_DISP\_LEFT); /\* Nos aseguramos incremento de

direccion, display fijo\*/

WriteCmdXLCD(DON); /\* Encendemos LCD\*/

WriteCmdXLCD(BLINK\_OFF); /\* Apagamos parpadeo de cursor\*/

}

void DelayFor18TCY(void){

\_delay(240);

}

void DelayPORXLCD(void){

\_delay\_ms(15);

}

void DelayXLCD(void){

\_delay\_ms(5);

}

**Resultados**

Podemos ver que la pantalla muestra los caracteres que mandamos desde el programa

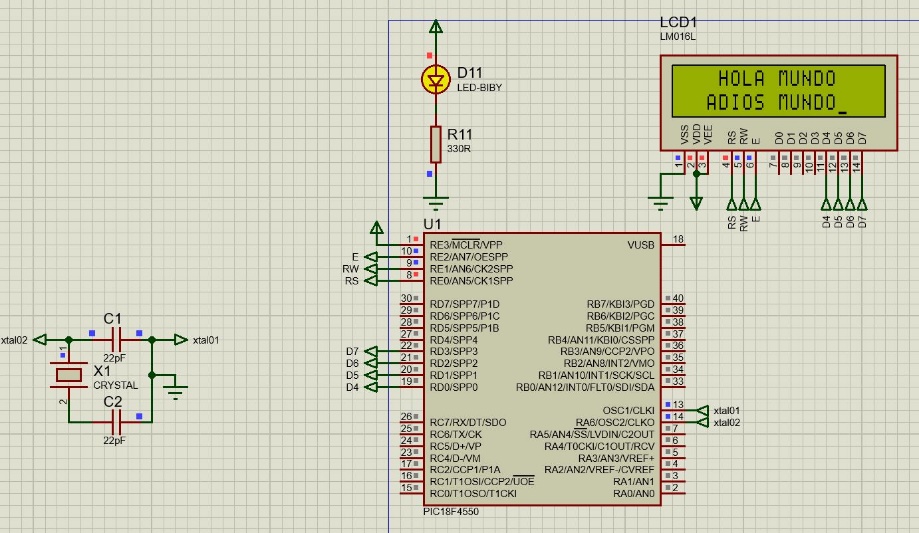


Imagen 10 LCD Simulación

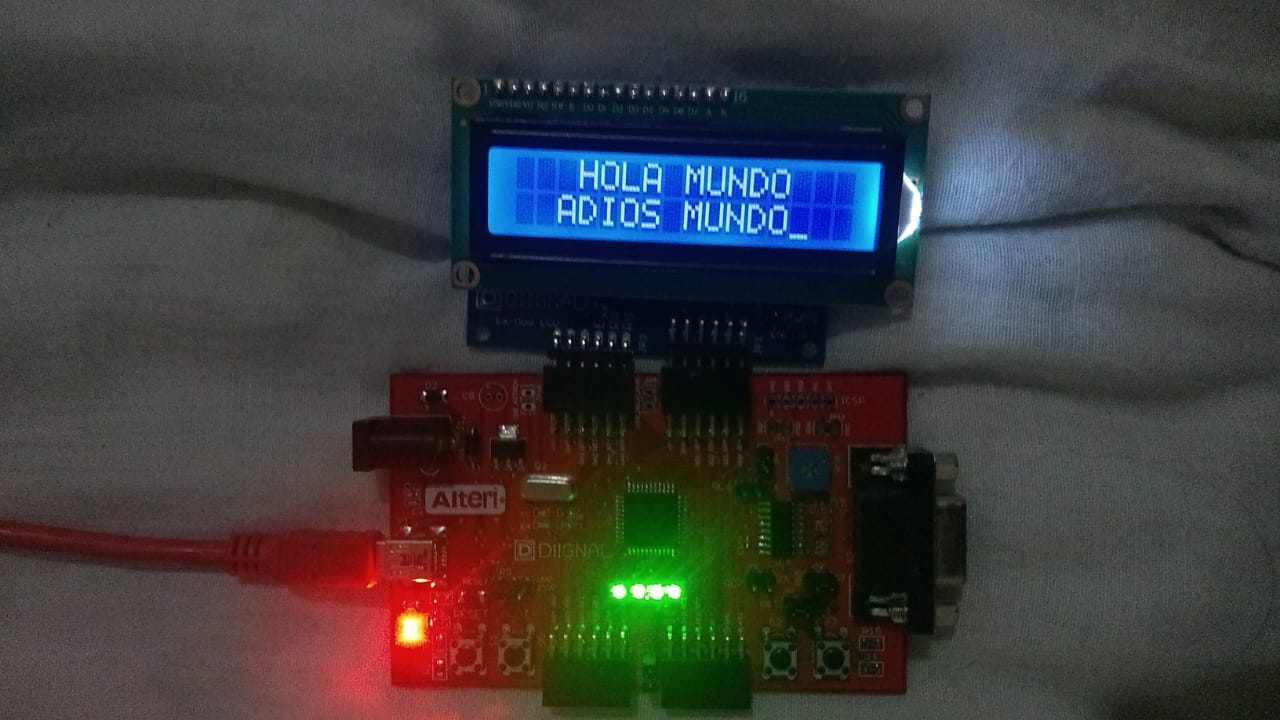


Imagen 11 LCD Físico

**Servomotor**

**Pseudocódigo**

1. Configurar Timer para emitir interrupción cada medio milisegundo
2. Cuando la interrupción se active
   1. Vaciar timer
   2. Aumentar contador
   3. Si el contador llega a 40(20 ms)
      1. Vaciar contador
      2. Mandar alto a señal de servo
   4. Si el contador llega a cierto valor predeterminado (0 para 0 grados 5 para 180) mandar bajo a señal de servo
   5. Volver a setear tiempo para interrupción
3. Botón uno pone valor predeterminado en 5
4. Botón dos pone valor predeterminado en 0

**Código para practica con Servomotor:**

#include <xc.h>

#include "Alteri.h"/\*Libreria espcífica de tarjeta de desarrollo usada \*/

#include<pic18.h>

#define servo LATBbits.LB2

#define boton1 PORTBbits.RB0

#define boton2 PORTBbits.RB1

#define \_XTAL\_FREQ 8000000

int count=0, value;

void main()

{

TRISB |= 0x03;

GIE = 1; /\* Habilitar interrupciones globales\*/

TMR0IE = 1; /\*Habilitar interrupción del timer 0\*/

T0CON = 0XC3; /\* Preescaler \*/

TMR0 = 0X64; /\* interrupción cada .5ms \*/

while(1)

{

if(boton1 && value != 5){/\*180 grados\*/

value = 5;

}

else if(boton2 && value != 0){/\*0 grados\*/

value = 0;

}

}

}

\_\_interrupt() void ISR(void)

{

if(T0IF)

{

TMR0IF = 0; /\* Vaciar timer \*/

count++; /\* Contador \*/

if(count==40) /\* Suma para obtener 20ms \*/

{

count = 0;

servo = 1; /\* Servo high \*/

}

if(count==value)

{

servo = 0;

}

TMR0=0x64; /\* Volver a setear tiempo al timer \*/

T0IF = 0;

}

}

**Resultados**

Podemos ver como el servomotor gira de 0 a 180 grados al picar los dos botones tanto en la simulación como en físico

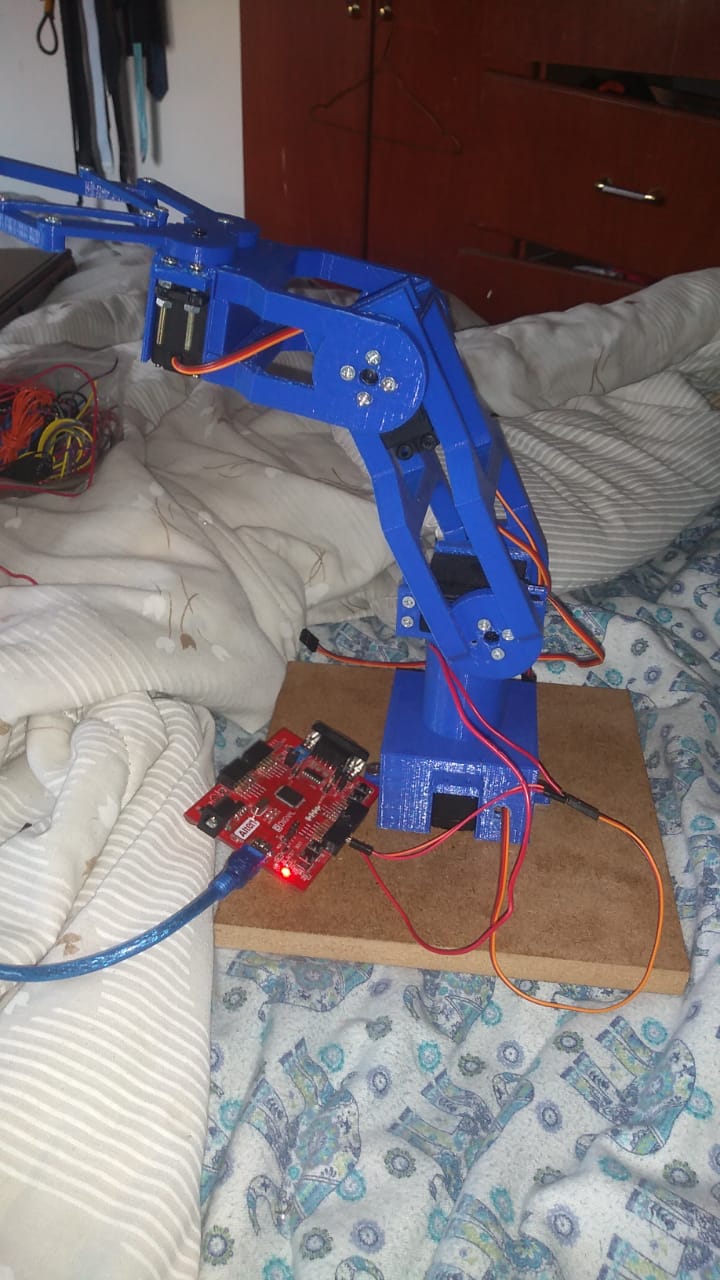
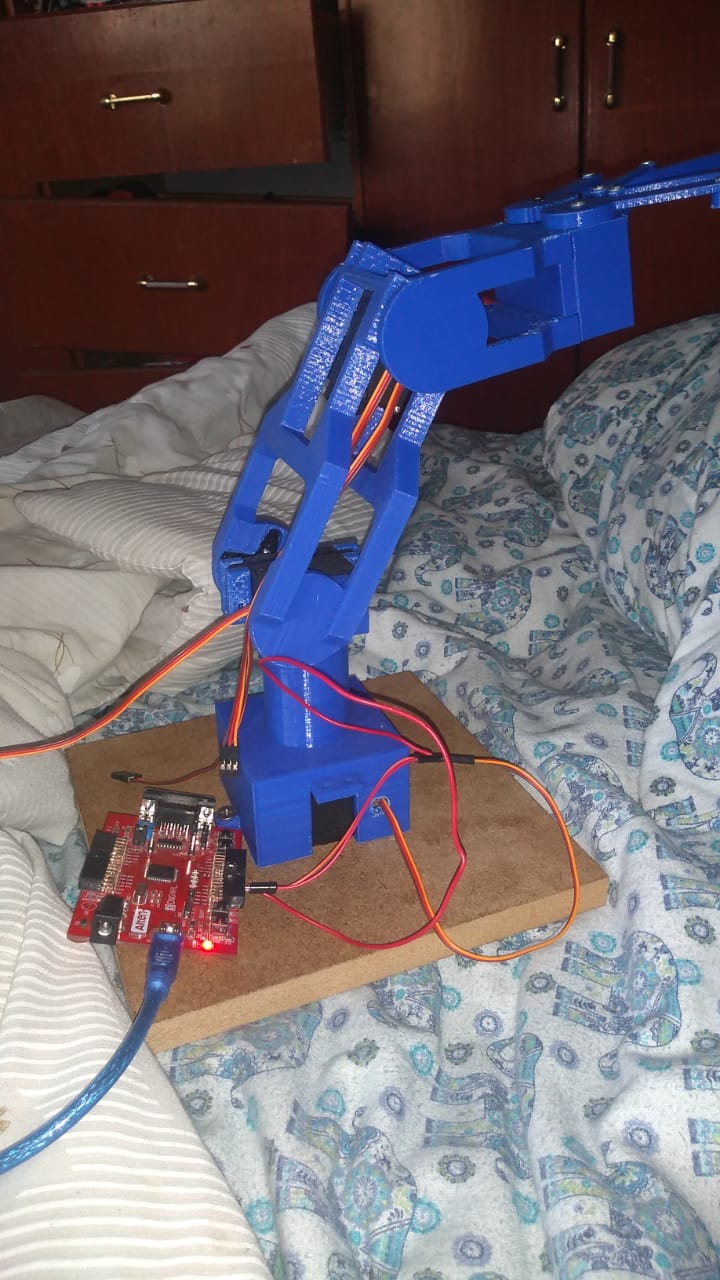


Imagen 12 Servomotor físico

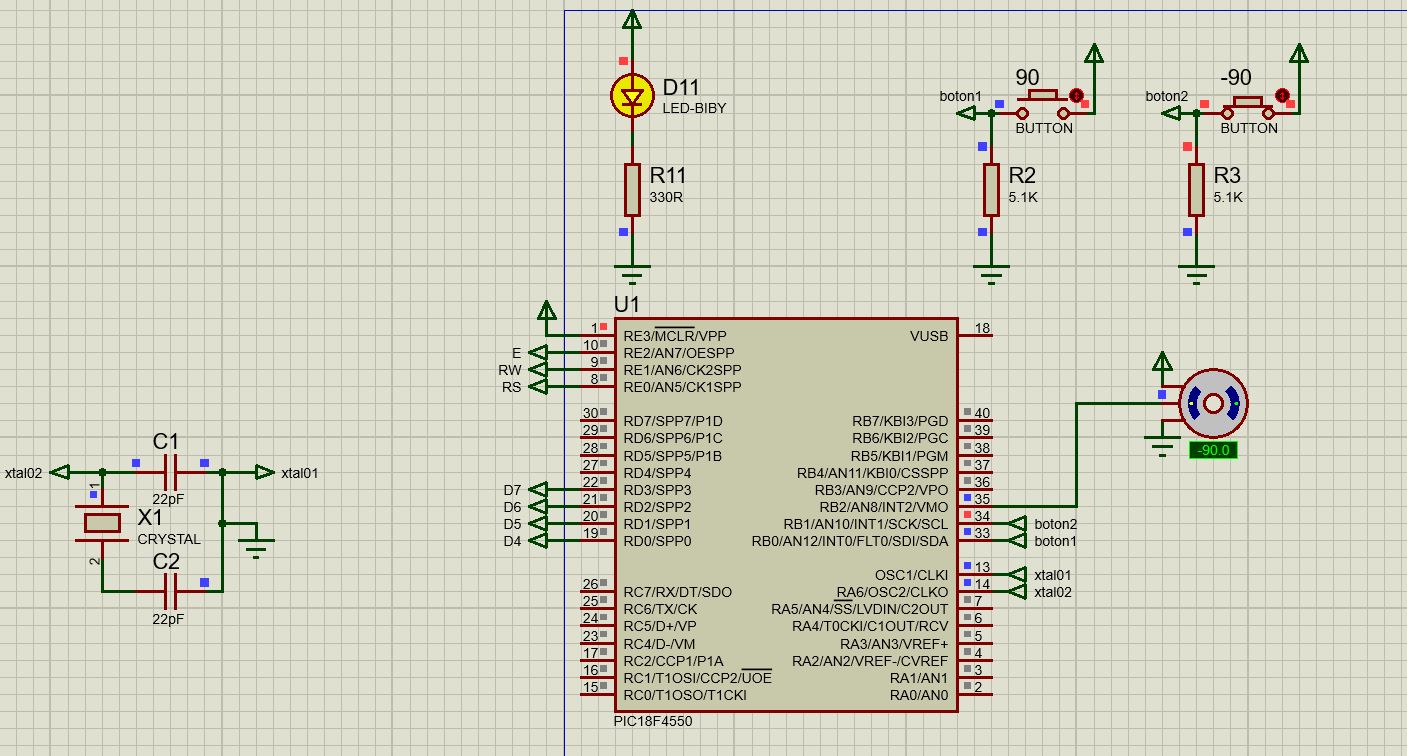
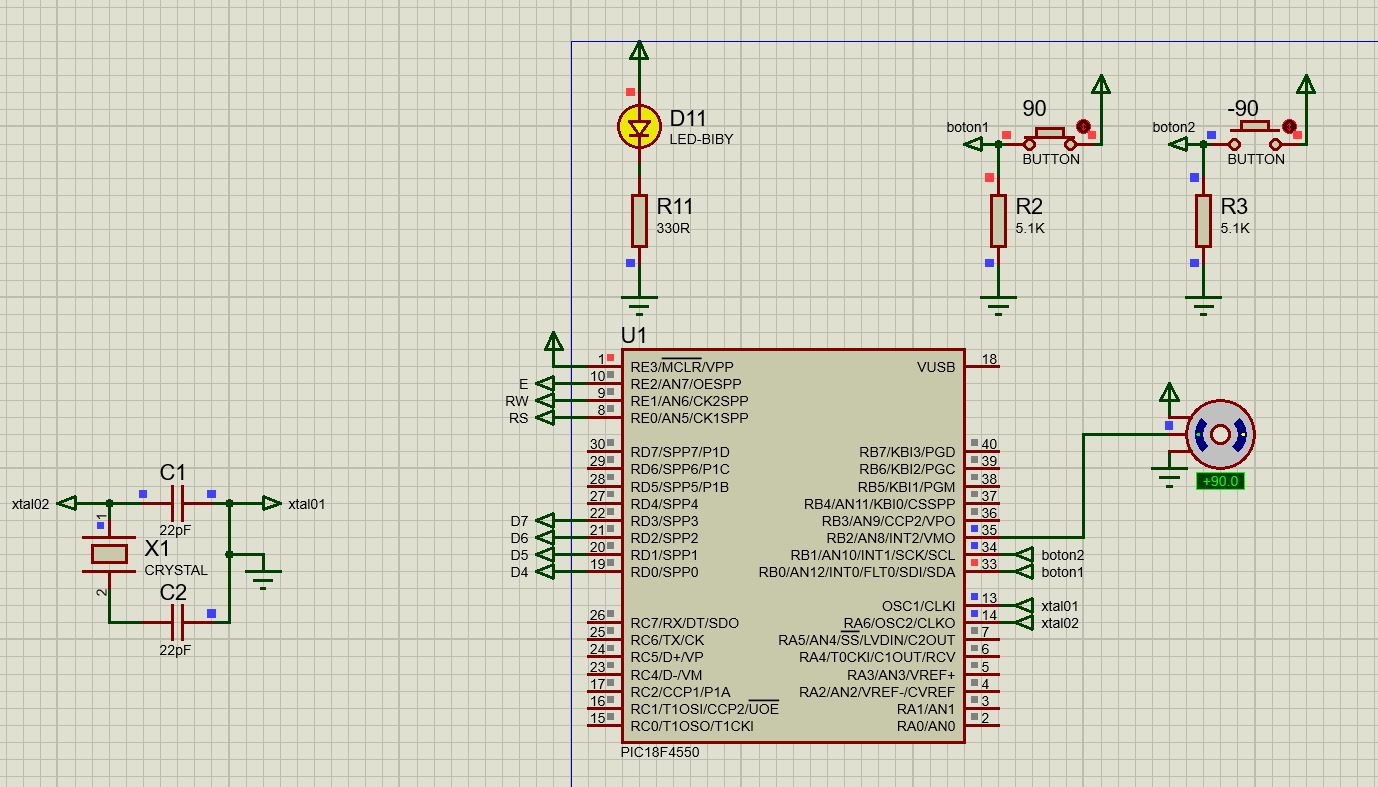


Imagen 13Servomotor Simulación

**Motor a pasos**

**Pseudocódigo**

1. Inicializar pines de motor
2. Hacer secuencia de de pines para girar motor a pasos

**Código para practica con Motor a pasos:**

#include <xc.h>

#include <stdint.h>

#include "Alteri.h"/\* Libreria espcífica de tarjeta de desarrollo usada \*/

#define period 100

void StepDelay(uint32\_t val);/\* Pseudo delay para tiempo entre señales a

motor\*/

void InterruptInit(void); /\* Inicializacion de interrupción \*/

uint8\_t value = 0;

void main(void)

{

OSCCON = 0x72;/\* Reloj interno a 8MHz\*/

InterruptInit();

TRISD = 0;

TRISA = 0;

while(1)

{

if(value){/\*Rotar el motor a pasos en el sentido del reloj a

medio paso\*/

LATD = 0x09;

StepDelay(period);

LATD = 0x08;

StepDelay(period);

LATD = 0x0C;

StepDelay(period);

LATD = 0x04;

StepDelay(period);

LATD = 0x06;

StepDelay(period);

LATD = 0x02;

StepDelay(period);

LATD = 0x03;

StepDelay(period);

LATD = 0x01;

StepDelay(period);

}

/\* Rotar el motor en contra del sentido del reloj a paso completo \*/

else{

LATD = 0x09;

StepDelay(period);

LATD = 0x03;

StepDelay(period);

LATD = 0x06;

StepDelay(period);

LATD = 0x0C;

StepDelay(period);

}

}

return;

}

void InterruptInit(void){

INTCONbits.INT0E = 1; //Habilita interrupcion por RB0

INTCON2bits.INTEDG0 = 1; //int por flanco de subida

INTCONbits.INT0F = 0; //reset a bandera de int

INTCONbits.GIE = 1; //switch para habilitar interrupciones

}

void StepDelay(uint32\_t val)

{

uint32\_t i,j;

for(i=0;i<val;i++)

{

for(j=0;j<165;j++);

}

}

\_\_interrupt() void ISR(void)

{

if(INTCONbits.INT0IF==1)

{

if(value == 0){

value = 1;

}

else if(value == 1){

value = 0;

}

LATAbits.LA3 = value;

INTCONbits.INT0IF=0; /\*vaciar bandera de interrupción\*/

}

}

**Resulados**

Podemos ver como el motor gira para un lado con medios pasos y cuando se presiona el botón gira al sentido contrario y con pasos completos



Imagen 14 Motor a pasos Físico

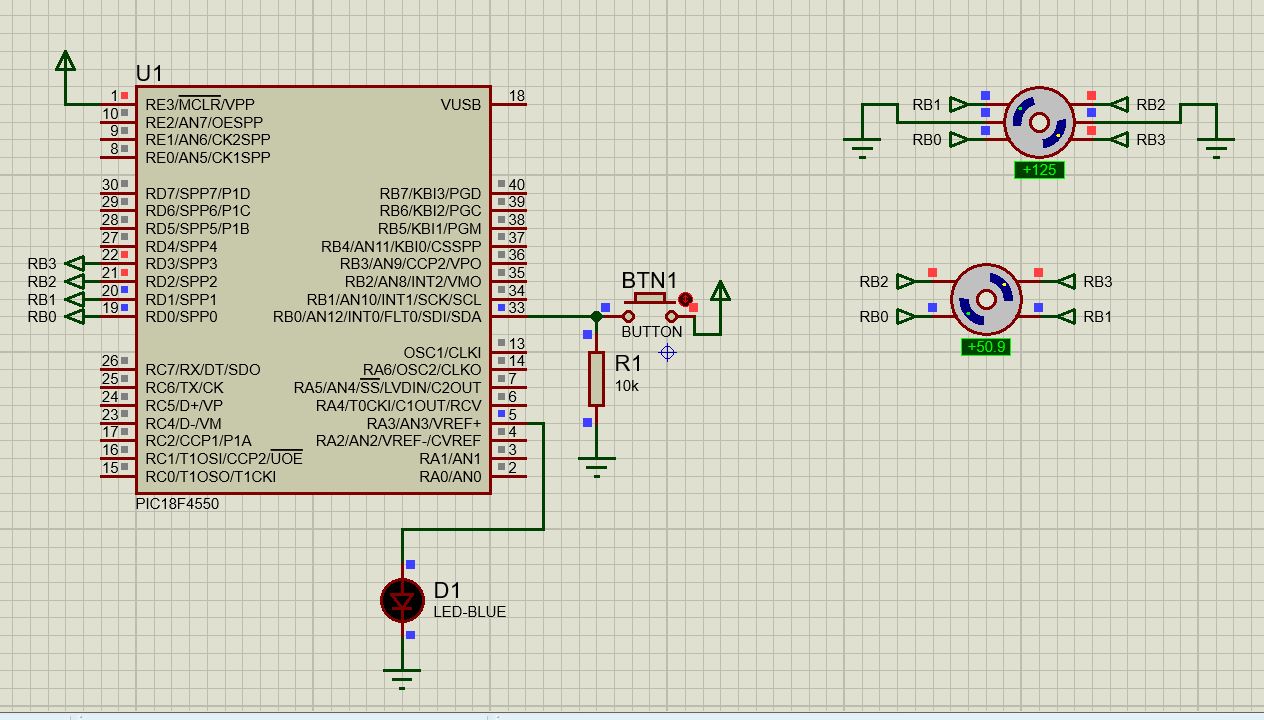


Imagen 15 Motor a pasos Simulación

**Motor DC**

**Pseudocódigo**

1. Pin 1 alto y pin 2 bajo
2. Si botón es presionado cambiar sentido de motor con pin 1 bajo y pin 2 alto

**Código para practica con Motor DC:**

#include <xc.h>

#include <stdint.h>

#include "Alteri.h"/\* Libreria espcífica de tarjeta de desarrollo usada\*/

void InterruptInit(void);/\* Función para inicializar la interrupcion en

RB0 \*/

uint8\_t value = 0;

void main(void)

{

OSCCON = 0x72;/\* Reloj interno como 8MHz\*/

InterruptInit();

TRISBbits.RB1 = 0;

TRISBbits.RB2 = 0;

while(1)

{

if(value)

{

LATBbits.LB1 = 1;

LATBbits.LB2 = 0;

}

else

{

LATBbits.LB1 = 0;

LATBbits.LB2 = 1;

}

}

return;

}

void InterruptInit(void)

{

INTCONbits.INT0E = 1; //Habilita interrupcion por RB0

INTCON2bits.INTEDG0 = 1; //int por flanco de subida

INTCONbits.INT0F = 0; //reset a bandera de int

INTCONbits.GIE = 1; //switch para habilitar interrupciones

}

\_\_interrupt() void ISR(void)

{

if(INTCONbits.INT0IF==1)

{

if(value == 0){/\*Cambiar el valor de value\*/

value = 1;

}

else if(value == 1){

value = 0;

}

INTCONbits.INT0IF=0; /\* Vaciar la bandera de interrupción \*/

}

}

# **Resultados**

Podemos ver como el motor gira para un lado y cuando se presiona el botón gira hacia el otro lado

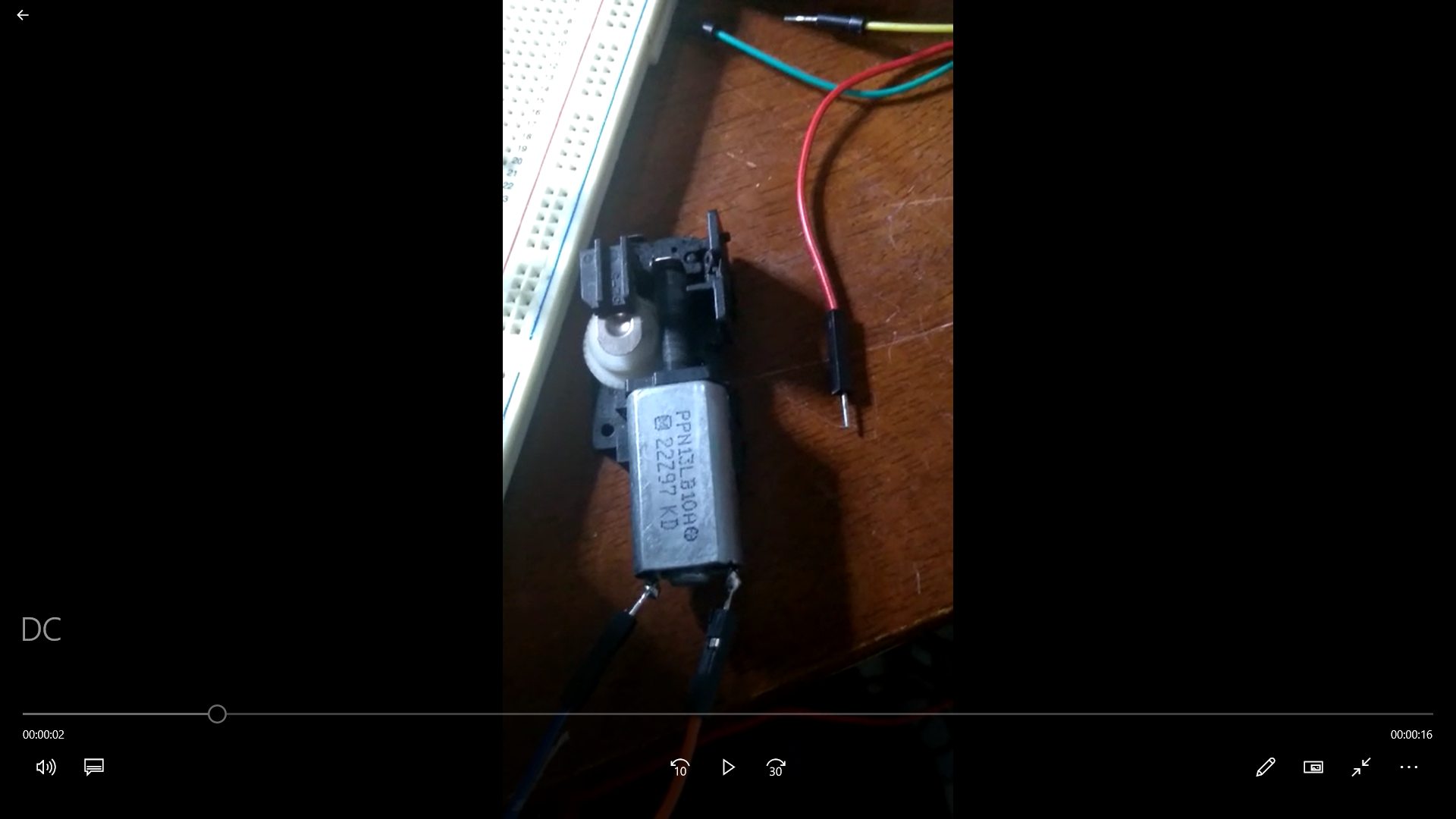


Imagen 16 DC Físico

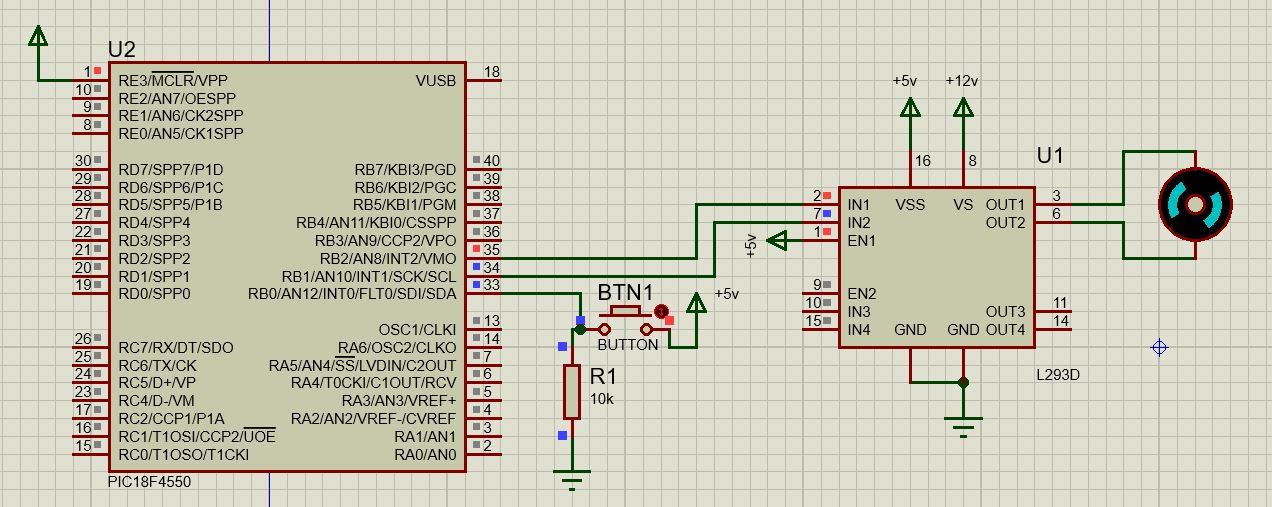


Imagen 17 DC Simulación

# Conclusiones

Conclusiones Daniel Castellanos: En estas practicas pude poner en práctica varias cosas que ya había hecho, pero al hacerlas en Arduino estaban muy simplificadas y al hacerlo en PIC pude aprender todas las bases que es lo que normalmente se pierde al hacer en Arduino y usar las librerías preestablecidas. Tuvimos problemas en la práctica de motor a pasos ya que no estábamos conectando bien los pines y no hacíamos correctamente la secuencia, pero al realizar varias pruebas pudimos lograrlo, al igual en la práctica de ADC no estábamos haciendo bien el enmascaramiento para mandar los LED con el valor binario y no salían correctamente, pero al analizarlo desde el punto de vista de bits pudimos hacerlo.

Conclusiones José Andrés Aguilar: En estas prácticas aprendí como programar y usar el PIC en lo particular la práctica que más se me complicó entender y desarrollar fue la del LCD debido a que no había trabajado con ellas antes. Pudimos realizar el código y las simulaciones con éxito, así como las pruebas en el vídeo anexado.

Conclusiones Yael López: Como conclusión podemos observar que en base a los resultados de nuestro pseudocódigo se trató de el control de motores en base a códigos, por lo cual logramos programar un ADC y un LCD satisfactoriamente, igualmente analizamos funciones de acuerdo con los programas y funciones que utilizamos. Con estas prácticas también logramos obtener resultados positivos con los programas y conocimiento que previamente no teníamos totalmente claro.

Conclusiones Alberto Morales: En conclusión, podemos observar el tipo de manejo o control que se le puede dar a los motores con la modificación del código, de igual manera con los convertidores de analógico a digital. Tal cual es el caso son el proyecto de brazo robótico que vamos armando, donde con base a nuestro código implementado, el servomotor podrá actuar según nuestras indicaciones.

# Referencias

330ohms. (16 de Marzo de 2016). *330OHMS*. Obtenido de https://blog.330ohms.com/2016/03/14/que-son-los-servomotores/

Anda, N. d. (18 de Ocutbre de 2018). *Factor*. Obtenido de https://www.factor.mx/portal/base-de-conocimiento/motor-dc/

Enrique, G. (28 de febrero de 2017). *Rinconingenieril*. Obtenido de https://www.rinconingenieril.es/conversor-analogico-a-digital-adc/

Faletti, E. (18 de octubre de 2016). *Monografias*. Obtenido de https://www.monografias.com/trabajos94/motores-paso-paso/motores-paso-paso.shtml

Mecafenix, F. (18 de Abril de 2017). *Ingenieria Mecafenix*. Obtenido de https://www.ingmecafenix.com/electricidad-industrial/servomotor/

Unicrom, E. (2016 de Mayo de 16). *Unicrom*. Obtenido de https://unicrom.com/lcd-display-de-cristal-liquido/