
	UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores	Semestre: 2020-2	
	Prof. Dr. Saúl De La Rosa Nieves	Página 1 de 25	

TAREA-EXAMEN 4	
Título:	Práctica motor a pasos
Fecha:	19-05-2020
Preparado por:	Fiel Muñoz Teresa Elpidia
Evaluación:	

I. Planteamiento del proyecto

- Objetivo:

El objetivo del proyecto fue realizar un controlador de motor a pasos 28BYJ con un teclado matricial y un display 7 segmentos.



- Descripción del proyecto:

Para la implementación del proyecto se utilizará la entrada con un teclado matricial y se mostrará el ángulo que se ingresa en un display 7 segmentos. Se podrá programar el sentido del giro siendo positivo el sentido horario y negativo el antihorario.

II. Requerimientos del proyecto

Requerimientos de Hardware

1. Como unidad de procesamiento utilice el microcontrolador TM4C1294NCPDT.
2. El Ingreso de la secuencia de ángulos de movimiento del motor de pasos se debe de ingresar con un teclado matricial como el que se muestra en la Figura 1.
3. La visualización de los ángulos que se ingresen y el despliegue del ángulo que se esté se presentarán en un “Display de 7 segmentos” de 4 dígitos como el que se muestra en la Figura 2.

	UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores	Semestre: 2020-2	
	Prof. Dr. Saúl De La Rosa Nieves	Página 2 de 25	

4. El sentido de giro del motor se presentará en el primer dígito del “Display de 7 segmentos” de 4 dígitos.

Requerimientos de Software

1. Se debe utilizar el microcontrolador TM1294NCPDT.
2. Para la programación se utilizará lenguaje C.

Requerimientos de Funcionamiento

1. Al encender el sistema se podrán ingresar el ángulo de giro que se desee con un valor de hasta 3 dígitos enteros signados.
2. El signo negativo del ángulo (sentido contrario a las manecillas del reloj) se asignará mediante la tecla * y se mostrara en el primer dígito del Display.
3. Cada ángulo que se ingrese se deberá mostrar en el “Display”
4. Después de ingresar el valor del ángulo se seleccionara la letra “A” y después se podrá ingresar el siguiente ángulo y así sucesivamente.
5. Para terminar el ingreso de ángulos deberá presionar la tecla # y como siguiente paso el sistema esperara el ingreso por el teclado de la velocidad del motor dada en segundos por revolución. Es decir si se ingresa el numero 100 entonces una revolución del motor se ejecutara en 100 segundos.
6. Después del ingreso del tiempo de revolución se deberá presionar nuevamente la tecla # y se ejecutara el movimiento programado y el valor del ángulo que se esté ejecutando se mostrara en el “Display”.

III. Marco teórico (antecedentes necesarios para el diseño)

Para el diseño de este proyecto se necesitaron conocimientos previos sobre el teclado matricial y su funcionamiento, así como del display 7 segmentos sea ánodo y cátodo común. Finalmente también se abordará el tema del motor a pasos brevemente para justificar porque lo programe así en el código fuente.

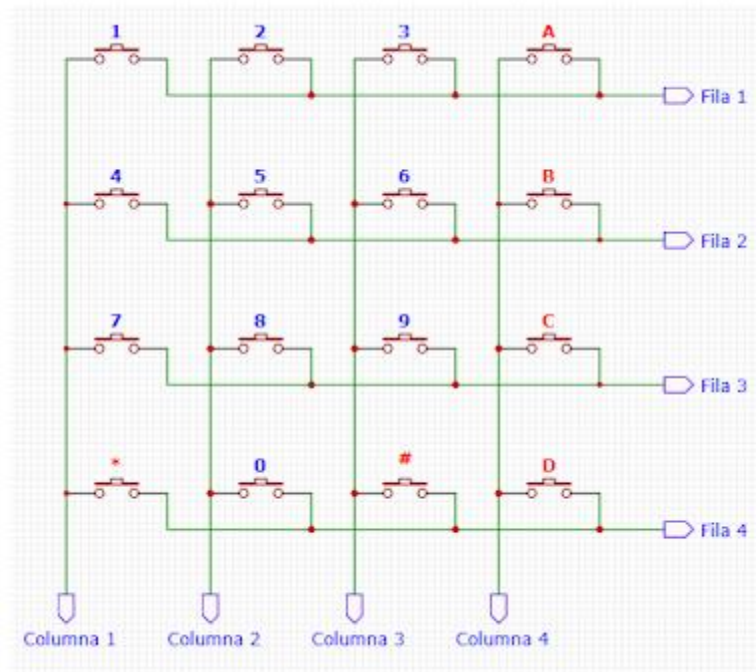


Figura 3. Estructura de un teclado matricial

Lo que se realiza es un barrido en las filas habilitando una por una para que en todo momento se estén habilitando. Dado que el ciclo de reloj es corto, se puede presionar una tecla y en el instante que la fila se habilite con el valor habrá una coincidencia fila-columna.

Siguiendo el diagrama se puede ver con esta coincidencia a que tecla corresponden los valores presionados. Una vez conseguido este valor se puede decodificar a 7 segmentos o a hexadecimal según sea el caso.

Decodificación de un display 7 segmentos

Para la decodificación de un display 7 segmentos primero se debe comprender la diferencia entre un display cátodo y ánodo común.

El display cátodo común posee un valor 1 común para cada terminal del display por lo que se necesita mandar un cero para el bit de habilitación o aterrizar a tierra el pin común para poder encender con un 1 cada uno de los 7 segmentos del display. A diferencia del display cátodo común, el ánodo común tiene un valor 0 común para cada terminal del display, por lo que se necesita mandar un 1 lógico al bit de habilitación para mostrar un dígito o bien mandar el pin común a un 1 lógico y encender con un 0 cada uno de los 7 segmentos del display.

Ahora bien, una vez que se identifica qué tipo de display se posee, se procede a identificar que led representa cada valor, las terminales se configuran de la siguiente manera:

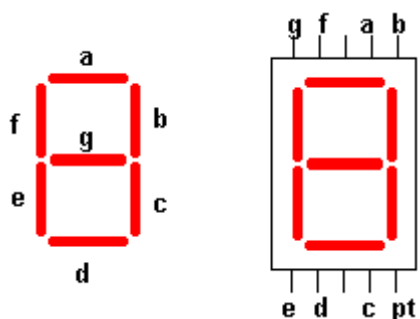


Figura 4. Terminales del display 7 segmentos

Cuando las terminales se han determinado, cada número tendrá una habilitación distinta dependiendo de qué leds deben prenderse y en qué configuración. De esta manera se determinan los dígitos para el display ánodo y cátodo común.

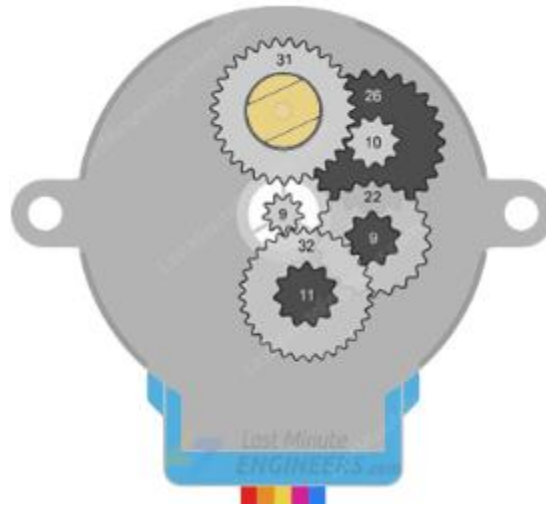
		Catodo Comun							
		Numero	A	B	C	D	E	F	G
Enable	0	1	1	1	1	1	1	1	0
0	1	0	1	1	0	0	0	0	0
0	2	1	1	0	1	1	0	1	1
0	3	1	1	1	1	0	0	1	1
0	4	0	1	1	0	0	1	1	1
0	5	1	0	1	1	0	1	1	1
0	6	1	0	1	1	1	1	1	1
0	7	1	1	1	0	0	0	0	0
0	8	1	1	1	1	1	1	1	1
0	9	1	1	1	1	0	1	1	1

		Anodo Comun							
		Numero	A	B	C	D	E	F	G
Enable	0	0	0	0	0	0	0	0	1
1	1	1	0	0	1	1	1	1	1
1	2	0	0	1	0	0	1	0	0
1	3	0	0	0	0	1	1	0	0
1	4	1	0	0	1	1	0	0	0
1	5	0	1	0	0	1	0	0	0
1	6	0	1	0	0	0	0	0	0
1	7	0	0	0	1	1	1	1	1
1	8	0	0	0	0	0	0	0	0
1	9	0	0	0	0	1	0	0	0

Figura 5. Tabla de números para el display 7 segmentos

Motor a pasos

El motor a pasos es un 28BYJ el cual tiene un voltaje de operación de 5V de DC. Para manipularlo se prefiere usar una fuente externa al microcontrolador y por lo tanto se utilizó un eliminador de 5V a 100mA. En la programación del motor se tomaron en cuenta los pasos a seguir para completar ciertos grados y la conversión de recarga para ajustar la velocidad en RPM.



Para los grados se utilizó la siguiente formula:

$$Pasos = \frac{grados}{11.2} * 64$$

Dado que la velocidad se da respecto al tiempo en el que se da una revolución, para calcular el valor de recarga necesito saber cuántos pasos involucran una revolución.

Sustituí el valor de 360 grados en la fórmula anterior obteniendo:

$$Pasos = \frac{360}{11.2} * 64 = 2057$$



Ahora para obtener el valor de recarga dependiendo de la velocidad en segundos para una revolución efectué la siguiente operación:

$$Recarga = \frac{Velocidad * 16,000,000}{2057}$$

IV. Diseño

Para el diseño del proyecto se tomaron en cuenta varias etapas. La primera fue la programación de lectura en el teclado matricial y su decodificación en el display 7 segmentos. La segunda fue la programación de las teclas A y B del teclado para transformar el sentido de giro y los grados, así como la velocidad en variables numéricas operables. Y el tercero fue programar la operación del motor para que realizara el ángulo solicitado con su sentido y velocidad correspondientes.

Primera Etapa

	UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores	Semestre: 2020-2	
	Prof. Dr. Saúl De La Rosa Nieves	Página 6 de 25	

Para la primera etapa se habilitaron los puertos K, M y E. El primero lo use para el teclado matricial habilitando 4 bits de entrada y 4 de salida. El puerto M se ocupó para los 7 segmentos del display y el puerto E para la habilitación de los dígitos a ingresar.

Se realizó un barrido por las filas del teclado y se igualaba cada tecla presionada con su valor correspondiente en 7 segmentos o hexadecimal.

Posteriormente se desplegaban los datos habilitando uno por uno cada bit del puerto E para mostrar dígito por dígito los valores y que se muestren en todo momento durante su ingreso.



Segunda Etapa

Para la programación de la segunda etapa se tomaron en cuenta varios procesos. El primero fue la asignación del giro del motor y como se implementó en el teclado. El segundo fue la tecla de ingreso del ángulo la cual fue la letra A y el tercero fue el ingreso de la velocidad con la tecla B.

Para el primer proceso se configuró la tecla sólo para ingresar y ser válida cuando se ingresa al inicio de todos los dígitos, si se cumple eso el dígito de las decenas va a ser el segundo del teclado en lugar del primero por lo que el conteo de dígitos se incrementa, a su vez se programó una variable entera la cual indica con un 1 que el valor es negativo y con un 0 que es positivo. Para los valores positivos la cuenta empieza desde cero y solo se registran 3 dígitos, aunque si se llegara a ingresar un valor incorrecto y se quiere corregir se ingresan 4 dígitos y se vuelven a registrar porque el contador cuando llega a 4 se reinicia.

El segundo proceso involucra la conversión de los dígitos en el arreglo a un valor decimal. En este caso la primera etapa está diseñada para que el ingreso de datos se almacene en decimal y siete segmentos de manera paralela, por lo que lo único que se hace es multiplicar cada valor del arreglo por 100, 10 y 1 según sea el caso y sumar esos valores para obtener los grados en decimal. Posterior a ello se convierten los grados a pasos con la fórmula del marco teórico, posteriormente los dígitos que se muestran en el display 7 segmentos se vuelven 0 para ingresar la velocidad a continuación.

El tercer proceso involucra la programación de la tecla B la cual al ser ingresada por vez primera incrementa un contador llamado ingresos en una unidad. Si se realiza esto al ingresar de nuevo un valor en el display este se va a poder registrar como velocidad, por lo que al volver a apretar la tecla este valor registrado como arreglo en decimal y siete segmentos se podrá transformar en un solo valor entero para después ser

	UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores	Semestre: 2020-2	
	Prof. Dr. Saúl De La Rosa Nieves	Página 7 de 25	

convertido en el valor de recarga para el motor y posteriormente ejecutar ese ángulo con el sentido y la velocidad en el motor.

Tercera Etapa

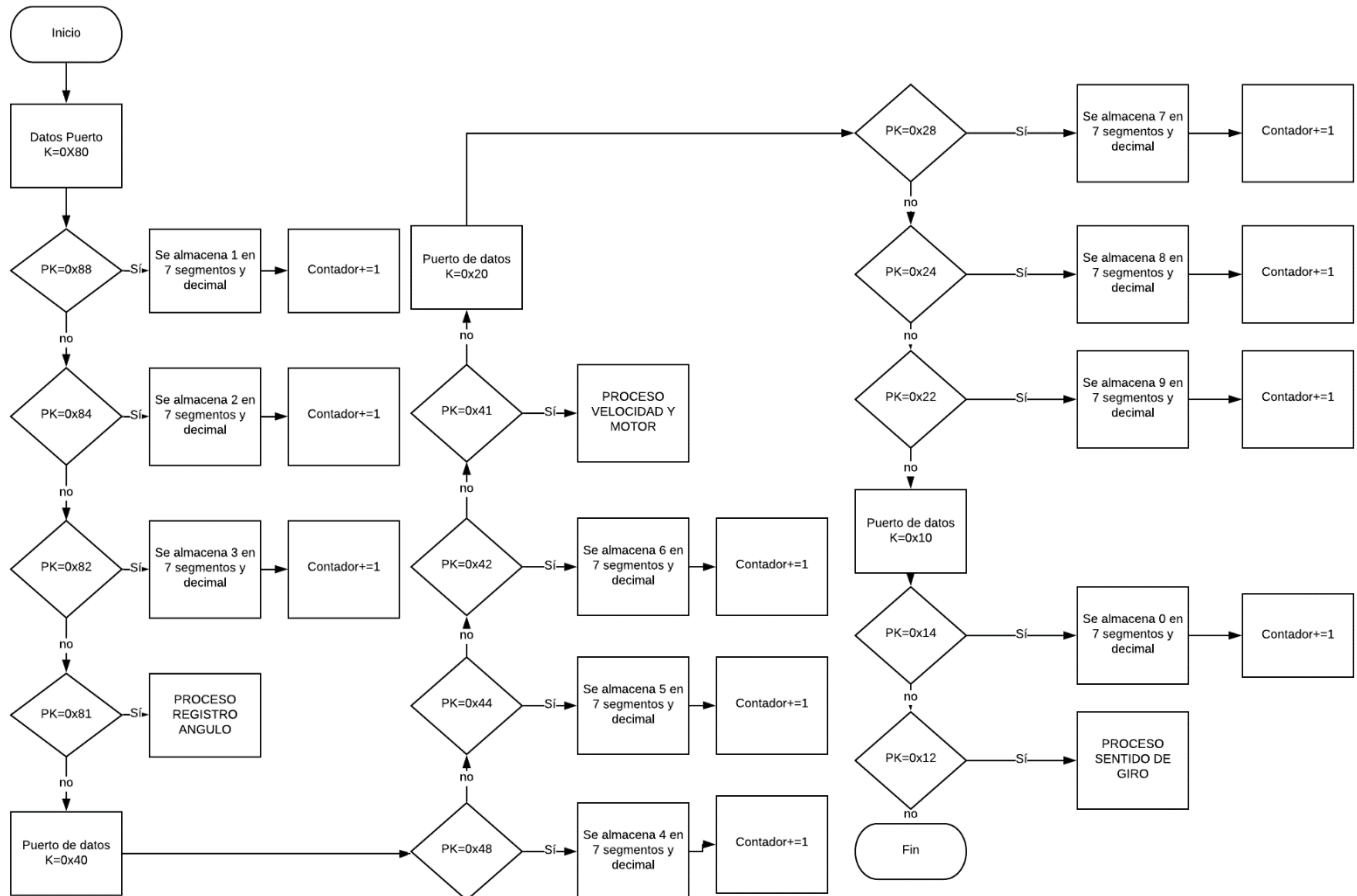
Para la tercera etapa ya se tienen los pasos del motor y el valor de recarga, por lo que el último paso es mandar el motor con un sentido de giro dependiendo de la variable negativo. Si es 0 el motor tiene sus switch case ordenados en manera descendente y si por el contrario es 1 los valores se registran de manera ascendente. Para poder mostrar el valor de los grados antes de realizar el borrado de los dígitos que registran los 7 segmentos se igualan estos a un arreglo que sirve solo para la muestra durante la ejecución por lo que estos valores se muestran debajo del código de muestra común con la condición de que una variable a la que llamé muestra valga 1 para que sólo se muestren estos dígitos cuando el motor esté en operación.

V. Diagrama de flujo y código comentado (en caso de VHDL o código de programa)

Diagrama de flujo

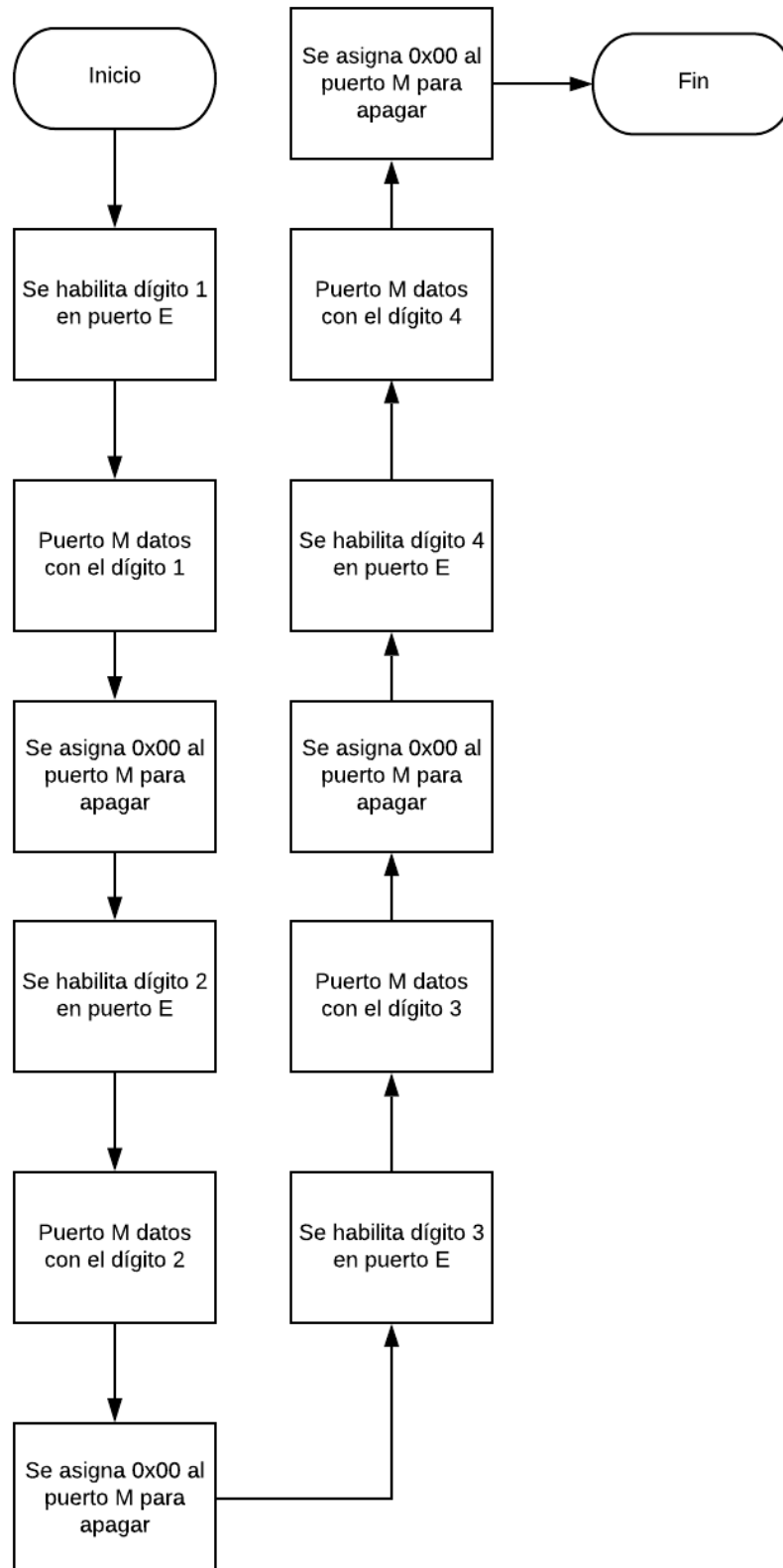


1. Decodificación del teclado matricial



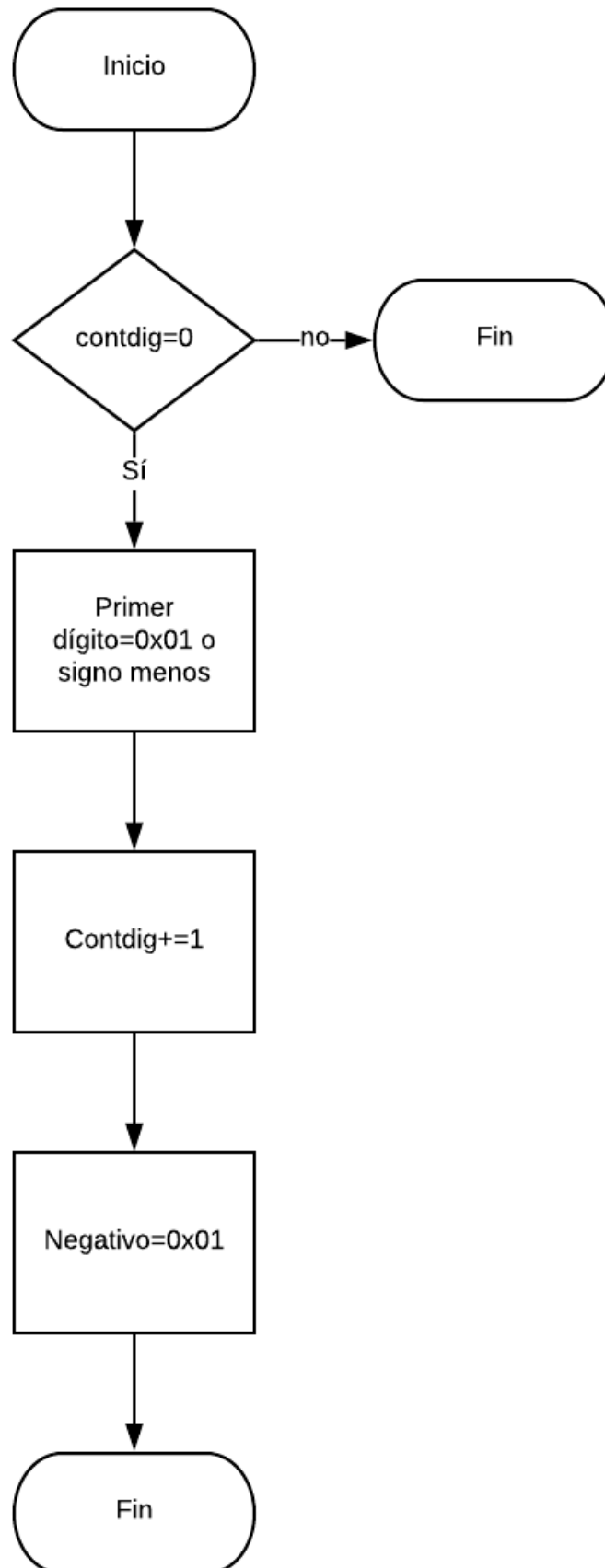


2. Despliegue del display 7 segmentos



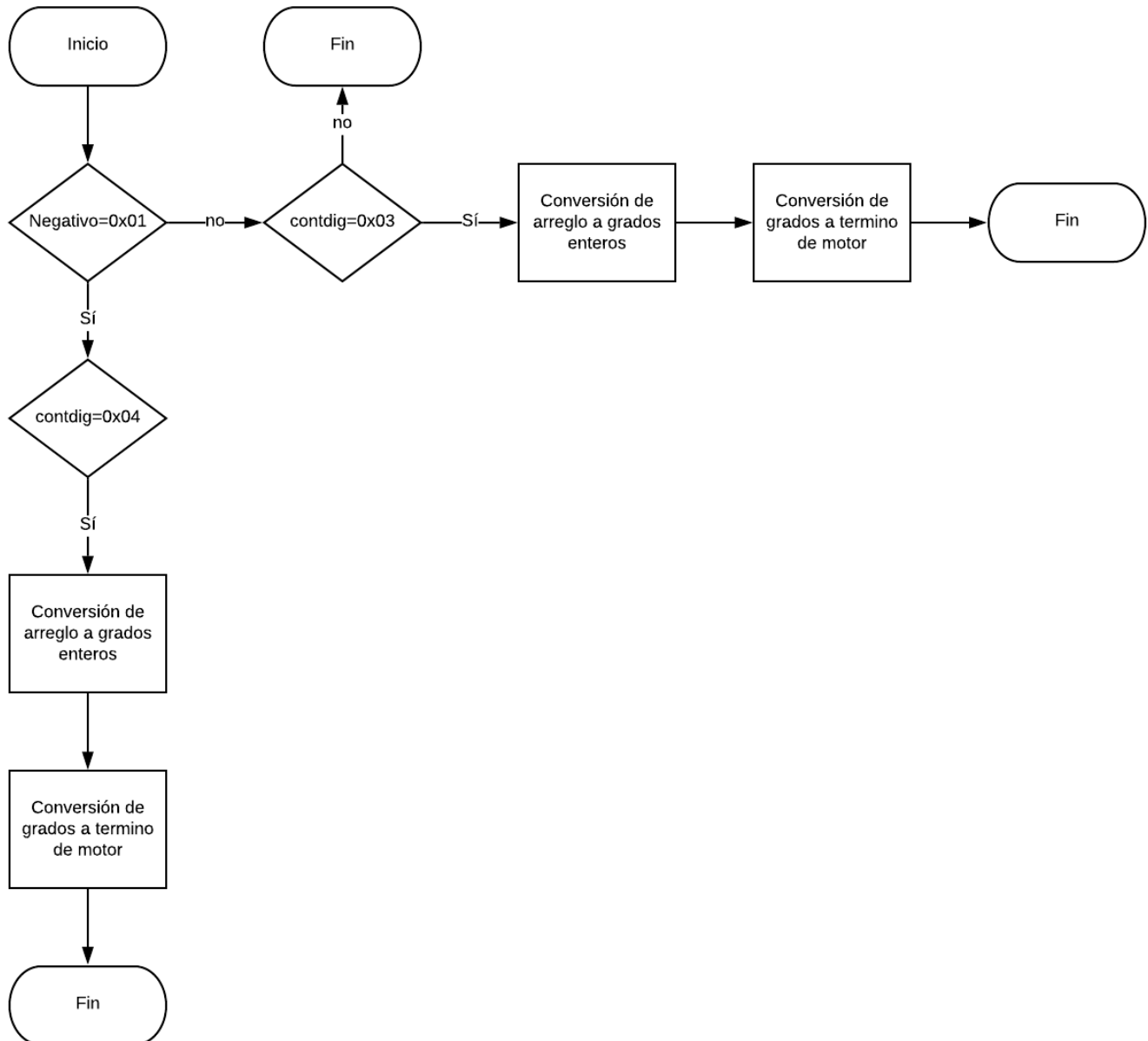


3. Sentido de giro (botón *)



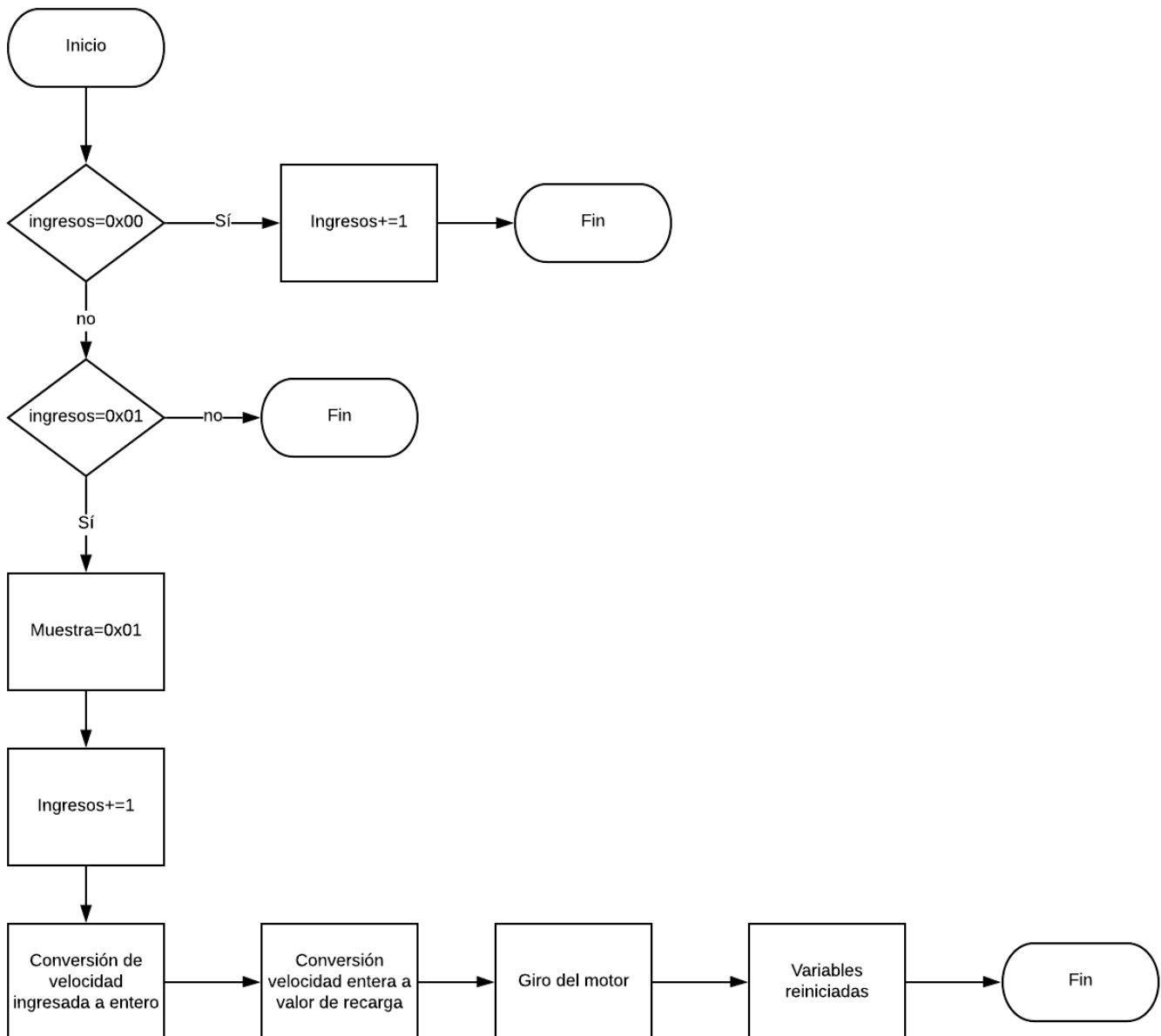




4. Ingreso del ángulo (botón A)





5. Ingreso de la velocidad y ejecución del motor (botón B)



	<p align="center">UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores</p>	<p>Semestre: 2020-2</p>	
	<p align="center">Prof. Dr. Saúl De La Rosa Nieves Grupo 3</p>	<p>Página 13 de 25</p>	

Código

```

/* TAREA-EXAMEN NO.4
 * Realizado por: Fiel Muñoz Teresa Elpidia
 */

//-----LIBRERIAS DEL PROGRAMA-----
#include <stdbool.h>
#include <stdint.h>
#include <math.h>
#include "inc/tm4c1294ncpdt.h"
volatile uint32_t Count= 0;
volatile uint32_t Termino= 0;

//-----DEFINICIÓN DE VARIABLES-----
-----

//-----TECLADO MATRICIAL-----

#define GPIO_PORTK_DATA_R      (*((volatile unsigned long *)0x400613FC)) // Registro de Datos
Puerto K
#define GPIO_PORTK_DIR_R      (*((volatile unsigned long *)0x40061400)) // Registro de
Dirección Puerto K
#define GPIO_PORTK_DEN_R      (*((volatile unsigned long *)0x4006151C)) // Registro de
Habilitación Puerto K
#define GPIO_PORTK_PDR_R      (*((volatile unsigned long *)0x40061514)) // Registro de Pull-
Down Puerto K

//-----DISPLAY 7 SEGMENTOS-----



#define GPIO_PORTM_DATA_R      (*((volatile unsigned long *)0x400633FC)) // Registro de Datos
Puerto M
#define GPIO_PORTM_DIR_R      (*((volatile unsigned long *)0x40063400)) // Registro de
Dirección Puerto M
#define GPIO_PORTM_DEN_R      (*((volatile unsigned long *)0x4006351C)) // Registro de
Habilitación Puerto M
#define GPIO_PORTM_PDR_R      (*((volatile unsigned long *)0x40063514)) // Registro de Pull-
Down Puerto M

//-----HAB. DISPLAY 7 SEGMENTOS-----

#define GPIO_PORTE_DATA_R      (*((volatile unsigned long *)0x4005C3FC)) // Registro de Datos
Puerto E
#define GPIO_PORTE_DIR_R      (*((volatile unsigned long *)0x4005C400)) // Registro de
Dirección Puerto E
#define GPIO_PORTE_DEN_R      (*((volatile unsigned long *)0x4005C51C)) // Registro de
Habilitación Puerto E
#define GPIO_PORTE_PUR_R      (*((volatile unsigned long *)0x4005C510)) // Registro de Pull-
Up Puerto E

int graditos,muestras; // Variables de
grados en pasos y los valores registrados de muestras
int digmuestra[]={0,0,0,0}; // Dígitos de muestra
del teclado

```

	UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores		Semestre: 2020-2	 INGENIERÍA ELÉCTRICA ELECTRÓNICA
	Prof. Dr. Saúl De La Rosa Nieves	Grupo 3	Página 14 de 25	

```

int negativo=0x00; // Valor entero para
saber si es o no negativo el giro
int a,b,c,d;

// RUTINA DE SERVICIO DE INTERRUPCIÓN
void Timer03AIntHandler(void)
{
    //LIMPIA BANDERA
    TIMER3_ICR_R= 0X00000001 ; //LIMPIA BANDERA DE TIMER3
    Termino = Termino + 1; // Se suma 1 a termino a manera de ciclo

    if(Termino==graditos)// Si se han completado los pasos del motor
    {
        negativo=0x00; //Se reinicia el valor negativo a positivo por default
        muestras=0x00; // La muestra de los grados en el programa se reinicia
    }
    if (Termino < graditos) // 32 * 64 = 2048
    {

        if(negativo==0)// Switch case para el caso de giro horario
        {

            Count = Count + 0x01;// Se incrementa la cuenta

            switch (Count&0x0F) {
                case 0x04:
                    GPIO_PORTL_DATA_R=0x09; // A,B
                    GPIO_PORTN_DATA_R = 0x03; // A,B
                    GPIO_PORTF_AHB_DATA_R = 0x00; //
                    Count=0x00;// Se reinicia la cuenta
                    break;
                case 0x03:
                    GPIO_PORTL_DATA_R=0x0C; // A',B
                    GPIO_PORTN_DATA_R = 0x01; // B
                    GPIO_PORTF_AHB_DATA_R = 0x10; // A'
                    break;
                case 0x02:
                    GPIO_PORTL_DATA_R=0x06; // A', B'
                    GPIO_PORTN_DATA_R = 0x00; //
                    GPIO_PORTF_AHB_DATA_R = 0x11; //A', B'
                    break;
                case 0x01:
                    GPIO_PORTL_DATA_R=0x03; // A, B'
                    GPIO_PORTN_DATA_R = 0x02; // A
                    GPIO_PORTF_AHB_DATA_R = 0x01; // B'
                    break;
            }
        }
        else// Caso de giro antihorario
        {

            Count = Count + 0x01;//Se incrementa la cuenta en 1

            switch (Count&0x0F) {
                case 0x01:
                    GPIO_PORTL_DATA_R=0x09; // A,B

```



```
        GPIO_PORTN_DATA_R = 0x03; // A,B
        GPIO_PORTF_AHB_DATA_R = 0x00; //
        break;
    case 0x02:
        GPIO_PORTL_DATA_R=0x0C; // A',B
        GPIO_PORTN_DATA_R = 0x01; // B
        GPIO_PORTF_AHB_DATA_R = 0x10; // A'
        break;
    case 0x03:
        GPIO_PORTL_DATA_R=0x06; // A', B'
        GPIO_PORTN_DATA_R = 0x00; //
        GPIO_PORTF_AHB_DATA_R = 0x11; //A', B'
        break;
    case 0x04:
        GPIO_PORTL_DATA_R=0x03; // A, B'
        GPIO_PORTN_DATA_R = 0x02; // A
        GPIO_PORTF_AHB_DATA_R = 0x01; // B'
        Count=0x00;
        break;
    }
}
}

//-----RELOJ INTERNO-----

#define SYSCTL_RCGC2_R      (*((volatile unsigned long *)0x400FE608)) // Registro de
Habilitación de Reloj de Puertos
#define SYSCTL_PRGPIO_R      (*((volatile unsigned long *)0x400FEA08)) // Registro de
estatus de Reloj de Puerto

// Definición de constantes para operaciones
#define SYSCTL_RCGC2_GPION      0x00001E30 // bit de estado del reloj

main(void)
{
    int i,grados,velocidadr; // Datos de los grados en valor entero y el contador
    int velocidad=0x00; // El valor entero de la velocidad
    int contdig=0x00; // La cuenta que verifica que dígito se registra
    int ingresos=0x00; // El valor de ingreso para la velocidad con tecla #
    int digitos[]={0x00,0x00,0x00,0x00}; // Los dígitos reales que se registran en 7seg
    int digreal[]={0,0,0,0}; // Arreglo donde se almacenan los dig. reales
    int digvelocidad[4]; // Arreglo de dígitos de velocidad
    SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPION; // Activa el reloj

    while ((SYSCTL_PRGPIO_R & 0X1000) == 0){}; // reloj listo?

    GPIO_PORTK_DIR_R |= 0xF0; // Salidas PK4-PK7 y Entradas PK0-PK3
    GPIO_PORTM_DIR_R |= 0xFF; // Salidas de PM0-PM7
    GPIO_PORTE_DIR_R |= 0x0F; // Salidas de PE0-PE3
    GPIO_PORTK_DEN_R |= 0xFF; // Habilita PK0-PK7
    GPIO_PORTM_DEN_R |= 0xFF; // Habilita PM0-PM7
    GPIO_PORTE_DEN_R |= 0x0F; // Habilita PE0-PE3
```



```
GPIO_PORTK_PDR_R |= 0x0F;    // Habilita resistencias de pull-down para salidas del teclado  
matricial  
GPIO_PORTM_PDR_R |= 0xFF;    // Habilita resistencias de pull-down para salidas del display  
7 segmentos  
GPIO_PORTE_PDR_R |= 0x0F;    // Habilita resistencias de pull-up para dígitos de  
habilitación 7 segmentos  
  
while(1)  
{  
    /*Fila 1*/  
  
    GPIO_PORTK_DATA_R=0x80;    // Se multiplexa la primera fila del teclado  
  
    if(GPIO_PORTK_DATA_R==0x88)    // Presiona tecla 1  
    {  
        if(contdig==4)    // Si ya se ingresaron cuatro dígitos del display se  
reinicia  
        {  
            contdig=0x00;    // Cuenta reiniciada  
        }  
        if(contdig<=3){    // Si la cuenta es menor a 3 se ingresa el dato  
            digitos[contdig]=0x30;    // Valor del display 7 segmentos  
            digreal[contdig]=1;    // El dígito real es 1  
            contdig+=1;    // Se incrementa la cuenta  
            for(i=0;i<=0x100000;i++){// Valor de espera  
            }  
        }  
    }  
    else if(GPIO_PORTK_DATA_R==0x84)// Valor 2 leído  
    {  
        if(contdig==4)// Se reinician los displays  
        {  
            contdig=0x00;//El contador se reinicia  
        }  
        if(contdig<=3){// Si la cuenta es menor a 3 se ingresa el dato  
            digitos[contdig]=0x6D;// Valor del display 7 segmentos  
            digreal[contdig]=2;    //El dígito real es 2  
            contdig+=1;// Se incrementa la cuenta  
            for(i=0;i<=0x100000;i++){// Valor de espera  
            }  
        }  
    }  
    else if(GPIO_PORTK_DATA_R==0x82) // Valor 3 del teclado  
    {  
        if(contdig==4)// Si la cuenta es 4 del display  
        {  
            contdig=0x00;// Se reinicia el valor de cuenta  
        }  
        if(contdig<=3){// Si la cuenta es tres o menor  
            digitos[contdig]=0x79;// Valor del display 7 segmentos  
            digreal[contdig]=3;// El dígito real es 3  
            contdig+=1;    // Se incrementa la cuenta  
            for(i=0;i<=0x100000;i++){ // Valor de espera  
            }  
        }  
    }  
    else if(GPIO_PORTK_DATA_R==0x81) // Tecla A presionada  
    {  
        for(i=0;i<=0x100000;i++){// Se espera un rato
```




/* Obtención de grados en valor entero*/

```
if(negativo==1)//Si los grados que se registrarán son negativos
{
    if(contdig==4)// Si ya se llenó el display se registraron 4 dígitos
    {
        grados=digreal[1]*100+digreal[2]*10+digreal[3];//Conversión a decimal
        graditos=trunc(grados/11.2*64);//Se obtienen esos grados en pasos con
        truncamiento
        digmuestra[0]=digitos[0]; // Se transfieren los datos del 7seg
        digmuestra[1]=digitos[1]; // a otro arreglo para mostrarlo en lo que gira
        el motor
        digmuestra[2]=digitos[2]; // El dígito 3
        digmuestra[3]=digitos[3]; // Y el dígito 4
        for(i=0;i<=3;i++) // Se reinician los dígitos decimales y 7 seg
        {
            digitos[i]=0x00; // Se reinicia 7 seg
            digreal[i]=0x00; // Se reinicia dígitos en decimal
        }
        contdig=0x00; // El conteo se reinicia
    }
}
else if(contdig==3) // Si el valor es positivo la cuenta llega a 3
{
    grados=digreal[0]*100+digreal[1]*10+digreal[2]; // Se convierte a decimal
    graditos=trunc(grados/11.2*64); // Se convierten los grados en pasos
    digmuestra[0]=digitos[0]; // Se guardan los dígitos en 7seg para ser mostrados
    digmuestra[1]=digitos[1]; // dígito 1
    digmuestra[2]=digitos[2]; // dígito 2
    digmuestra[3]=digitos[3]; // dígito 3
    for(i=0;i<=3;i++) // Ciclo de reinicio de arreglos
    {
        digitos[i]=0x00; // Se reinician los dígitos en 7seg
        digreal[i]=0x00; // Se reinician los dígitos en decimal
    }
    contdig=0x00; // Se reinicia el contador de dígitos
}
}
```

/* Fila 2 */

GPIO_PORTK_DATA_R=0x40; // Se multiplexa la segunda fila

```
if(GPIO_PORTK_DATA_R==0x48) // Valor 4 presionado
{
    if(contdig==4) // Si ya se registraron todos los dígitos el conteo reinicia
    {
        contdig=0x00; // Se reinicia
    }
    if(contdig<=3){ // Si aún se pueden ingresar dígitos
        digitos[contdig]=0x33; // Valor 4 en 7 seg
        digreal[contdig]=4; // Valor 4 en decimal
        contdig+=1; // Se incrementa el conteo en 1
        for(i=0;i<=0x100000;i++){ // Ciclo de espera
        }
    }
}
else if(GPIO_PORTK_DATA_R==0x44) // Tecla 5 presionada
```



```
{
    if(contdig==4) // Si ya se ingresaron todos los dígitos
    {
        contdig=0x00; // El valor se reinicia
    }
    if(contdig<=3){ // Si aún se pueden ingresar datos
        digitos[contdig]=0x5B; // Valor 7 segmentos de 5
        digreal[contdig]=5; // Valor real 5
        contdig+=1; // Se incrementa el conteo
        for(i=0;i<=0x100000;i++){ // Ciclo de espera
        }
    }
}
else if(GPIO_PORTK_DATA_R==0x42) // Valor 6 presionado
{
    if(contdig==4) // Si ya se ingresaron todos los dígitos
    {
        contdig=0x00; // Se reinicia el valor
    }
    if(contdig<=3){ // Si aún se pueden ingresar dígitos
        digitos[contdig]=0x5F; // Se ingresa el valor 5 7segm
        digreal[contdig]=6; // Se ingresa el valor 6 decimal
        contdig+=1; // Se incrementa en 1 el contador
        for(i=0;i<=0x100000;i++){ // Ciclo de espera
        }
    }
}
else if(GPIO_PORTK_DATA_R==0x41) // Se configuró tecla B como # por fallas en mi
teclado
{
    if(ingresos==0) // Si se teclea una vez se prepara para recibir los datos
    {
        ingresos+=1; // Se incrementa ingresos para saber que ya se tecleo una vez
        for(i=0;i<=0x100000;i++){ // Ciclo de espera
        }
    }
    else if(ingresos==1) // Si ya se tecleo una vez se realizará el giro
    {
        muestras=0x01; // La muestra se activa para mostrar el display mientras gira
        ingresos+=1; // Los ingresos se vuelven a incrementar
        for(i=0;i<=0x100000;i++){ // Ciclo de espera
        }
        velocidad=digreal[0]*100+digreal[1]*10+digreal[2]; // Se convierte la velocidad
a dígito real
        velocidadr=trunc(velocidad*16000000/2057); // Conversión de velocidad a RPM
        //habilita PORTN, PORTF, PORTL
        SYSCTL_RCGCGPIO_R |= 0x1E30; // RELOJ PARA EL PUERTO F, L y N
        SYSCTL_RCGCTIMER_R |= 0x08; //RELOJ Y HABILITA TIMER 3 (p.380)

        //retardo para que el reloj alcance el PORTN Y TIMER 3
        while ((SYSCTL_PRGPIO_R & 0x1000) == 0){}; // reloj listo?
        TIMER3_CTL_R=0x00000000; //DESHABILITA TIMER 3 PARA CONFIGURAR (p.986)
        TIMER3_CFG_R= 0x00000000; //CONFIGURA TIMER DE 32 BITS (p. 976)
        //TIMER3_TAMR_R= 0x00000002; //CONFIGURAR PARA MODO PERIODICO CUENTA HACIA
ABAJO (p. 977)
        TIMER3_TAMR_R= 0x00000012; //CONFIGURAR PARA MODO PERIODICO CUENTA HACIA
ARRIBA (p. 977)
        TIMER3_TAILR_R= velocidadr; // VALOR DE RECARGA (p.1004)
        //
        TIMER3_TAILR_R= 0x0004E200; // VALOR DE RECARGA (p.1004)
        TIMER3_TAPR_R= 0x00; // PRESCALADOR DE TIMER A, SOLO PARA MODOS DE 16 BITS
(p.1008)
```



(p.1002)

```
TIMER3_ICR_R = 0x00000001 ; //LIMPIA POSIBLE BANDERA PENDIENTE DE TIMER3

TIMER3_IMR_R |= 0x00000001; //ACTIVA INTRRRUPCION DE TIMEOUT (p.993)
NVIC_EN1_R = 1<<(35-32); //HABILITA LA INTERRUPCION 35 (TIMER3 A)
TIMER3_CTL_R |= 0x00000001; //HABILITA TIMER 3 (p.986)

// habilita al Puerto L como salida digital para control de motor
// PL0,...,PL3 como salidas hacia el ULN2003 (A,A',B,B')
GPIO_PORTL_DIR_R = 0x0F; // Habilita PL0-PL3
GPIO_PORTL_DEN_R = 0x0F; // Habilita PL0-PL3 como salida
GPIO_PORTL_DATA_R = 0x09; // Valor de dato

// habilita PN0 y PN1 como salida digital para monitoreo del programa
//
GPIO_PORTN_DIR_R = 0x03; // Habilita PN0-PN1
GPIO_PORTN_DEN_R = 0x03; // PN0-PN1 como salida

// habilita PF0 y PF4 como salida digital para monitoreo del programa
//
GPIO_PORTF_AHB_DIR_R = 0x11; // PF0 y PF4
GPIO_PORTF_AHB_DEN_R = 0x11; // PF0 y PF4 como salida

digitos[0]=0x00; // Se reinician todos los dígitos en 7seg
digitos[1]=0x00; //dig1
digitos[2]=0x00; //dig2
digitos[3]=0x00; //dig3
contdig=0x00; // El conteo se reinicia
ingresos=0x00; // Los ingresos se reinician
Termino=0x00; // El termino del motor se reinicia
}
}

/* Fila 3 */

GPIO_PORTK_DATA_R=0x20; // Se multiplexa la fila 3

if(GPIO_PORTK_DATA_R==0x28)// Valor 7 leído
{
    if(contdig==4)// Si ya no se pueden ingresar más dígitos
    {
        contdig=0x00; // Se reinicia el conteo
    }
    if(contdig<=3){// Si aún se pueden ingresar dígitos
        digitos[contdig]=0x70; // Se ingresan los valores en 7seg
        digreal[contdig]=7; // Se ingresan los valores decimales
        contdig+=1; // Se incrementa el contador
        for(i=0;i<=0x100000;i++){// Ciclo de espera
        }
    }
}
else if(GPIO_PORTK_DATA_R==0x24)// Valor 8 leído
{
    if(contdig==4)// Si ya no se ingresan más datos
    {
        contdig=0x00; // Se reinicia el conteo
    }
    if(contdig<=3){// Si aún se pueden ingresar más datos
        digitos[contdig]=0x7F; // Valor 8 en 7seg
    }
}
```



```
    digreal[contdig]=8; // Valor 8 en decimal
    contdig+=1; // Se incrementa el conteo
    for(i=0; i<=0x100000; i++){ // Ciclo de espera
    }
}
else if(GPIO_PORTK_DATA_R==0x22) // Tecla 9 presionada
{
    if(contdig==4) // Si ya no pueden ingresarse datos
    {
        contdig=0x00; // Se reinicia el conteo
    }
    if(contdig<=3){ // Si aún se pueden ingresar datos
        digitos[contdig]=0x7B; // Valor 9 7seg
        digreal[contdig]=9; // Valor 9 decimal
        contdig+=1; // Se incrementa el conteo en 1
        for(i=0; i<=0x100000; i++){ // Ciclo de espera
        }
    }
}



/* Fila 4 */

GPIO_PORTK_DATA_R=0x10; // Multiplexación fila 4

if(GPIO_PORTK_DATA_R==0x18) // Si se teclea #
{
    // No se hizo nada porque se implemento en la tecla B
}
else if(GPIO_PORTK_DATA_R==0x14) // Si se teclea 0
{
    if(contdig==4) // Si ya no se pueden ingresar datos
    {
        contdig=0x00; // Se reinicia el conteo
    }
    if(contdig<=3){ // Si aún se pueden ingresar datos
        digitos[contdig]=0x7E; // Valor 0 en 7seg
        digreal[contdig]=0; // Valor 0 decimal
        contdig+=1; // Se incrementa el conteo
        for(i=0; i<=0x100000; i++){ // Ciclo de espera
        }
    }
}
else if(GPIO_PORTK_DATA_R==0x12) // Se presiona *
{
    if(contdig==0){ // Solo se acepta el signo si se ingresa al inicio
        digitos[contdig]=0x01; // Se iguala al signo - en 7seg
        contdig+=1; // Se incrementa el valor en 1
        negativo=0x01; // El valor negativo se hace 1
        for(i=0; i<=0x100000; i++){ // Ciclo de espera
        }
    }
}

/* Muestra de datos */

GPIO_PORTC_DATA_R=0xFE; // Habilita dígito 4
GPIO_PORTM_DATA_R=digitos[3]; // Enciende dígito 4
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTC_DATA_R=0xFD; // Habilita dígito 3
GPIO_PORTM_DATA_R=digitos[2]; // Enciende dígito 3
```

	<p align="center">UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores</p>	<p>Semestre: 2020-2</p>	 <p align="center">INGENIERÍA ELECTRICA ELECTRONICA</p>
	<p>Prof. Dr. Saúl De La Rosa Nieves</p>	<p>Página 21 de 25</p>	

```

GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
GPIO_PORTM_DATA_R=0xFB;// Habilita dígito 2
GPIO_PORTM_DATA_R=digitos[1];//Enciende dígito 2
GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
GPIO_PORTM_DATA_R=0xF7;// Habilita dígito 1
GPIO_PORTM_DATA_R=digitos[0];//Enciende dígito 1
GPIO_PORTM_DATA_R=0x00;

if(muestras==1)
{
    if(negativo==1)// Si el valor negativo es real
    {
        GPIO_PORTM_DATA_R=0xFE;// Habilita dígito 4
        GPIO_PORTM_DATA_R=digmuestra[3];//Enciende dígito 4
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTM_DATA_R=0xFD;// Habilita dígito 3
        GPIO_PORTM_DATA_R=digmuestra[2];//Enciende dígito 3
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTM_DATA_R=0xFB;// Habilita dígito 2
        GPIO_PORTM_DATA_R=digmuestra[1];//Enciende dígito 2
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTM_DATA_R=0xF7;// Habilita dígito 1
        GPIO_PORTM_DATA_R=0x01;// Dígito 1 es el signo negativo
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
    }
    else
    {
        GPIO_PORTM_DATA_R=0xFE;// Habilita dígito 4
        GPIO_PORTM_DATA_R=digmuestra[3];//Enciende dígito 4
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTM_DATA_R=0xFD;// Habilita dígito 3
        GPIO_PORTM_DATA_R=digmuestra[2];//Enciende dígito 3
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTM_DATA_R=0xFB;// Habilita dígito 2
        GPIO_PORTM_DATA_R=digmuestra[1];//Enciende dígito 2
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTM_DATA_R=0xF7;// Habilita dígito 1
        GPIO_PORTM_DATA_R=digmuestra[0];//Enciende dígito 1
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
    }
}
}
}

```

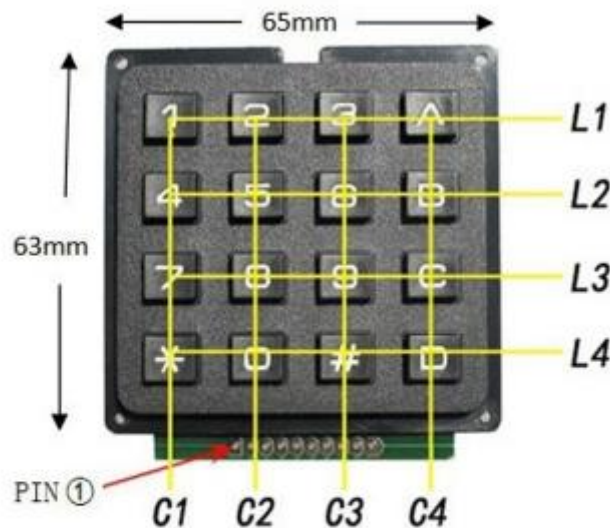
VI. Construcción

Para este proyecto se utilizaron 4 puertos del microcontrolador los cuales fueron el puerto M, K, L y E. Se utilizó un teclado matricial de 4x4 con 4 entradas y 4 salidas, un display 7 segmentos con 6 salidas para leds y 4 salidas para habilitación de dígitos y por último un motor a pasos con cuatro salidas. Además de esto se usó una fuente de 5 V externa para el motor.

Conexión al teclado matricial

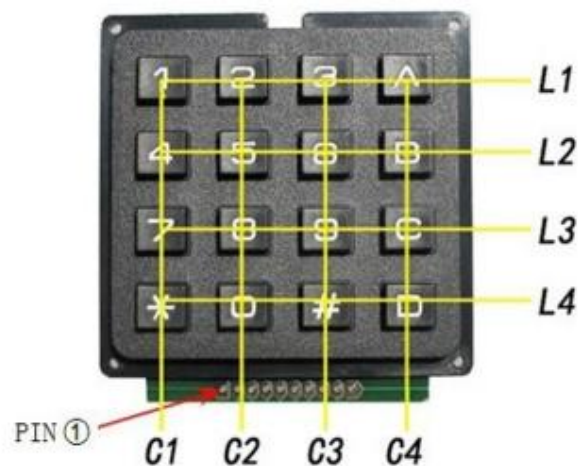


Para el teclado matricial se utiliza un solo puerto. En mi caso yo utilicé el teclado matricial que se muestra a continuación y que tiene esta configuración de pines para las entradas y salidas del teclado. Se configuran 4 entradas y 4 salidas, PK0-PK3 son salidas y PK4-PK7 son entradas.



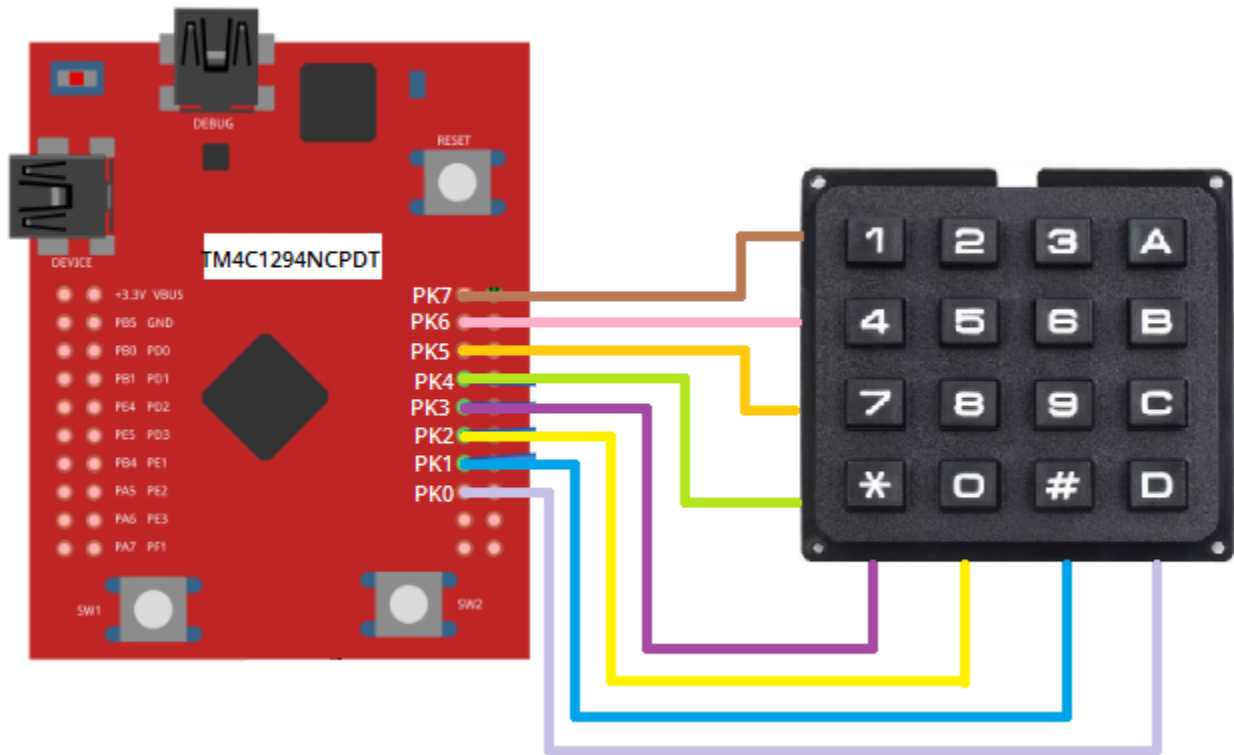
PIN	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
	NC	C1	C2	C3	C4	L1	L2	L3	L4	NC

Ya que se conocen como están configurados los pines del teclado matricial procedemos a poner el puerto de cada pin como corresponde, sabiendo que se conectarán 8 de ellos.



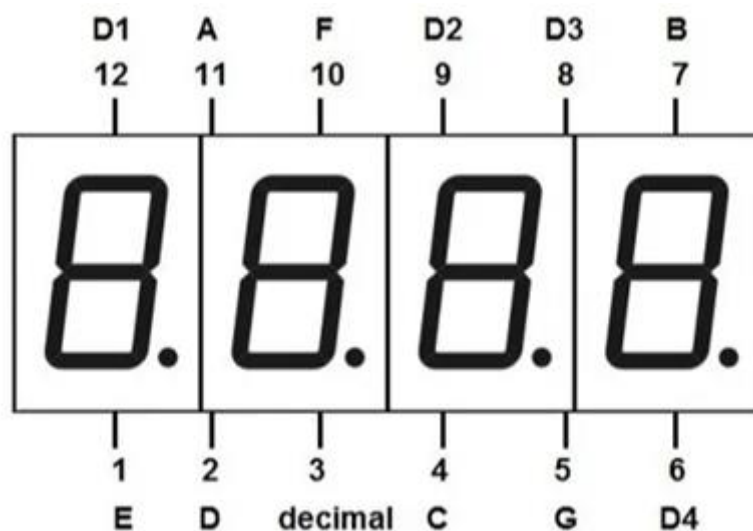
PIN	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
	NC	PK3	PK2	PK1	PK0	PK4	PK7	PK6	PK5	NC

Ahora se muestra una imagen de las conexiones al microcontrolador partiendo de las filas y columnas como se mostraron anteriormente, el teclado matricial ocupa el puerto K y las filas son las que se multiplexan para ir haciendo el barrido de datos y conseguir la lectura de las teclas.

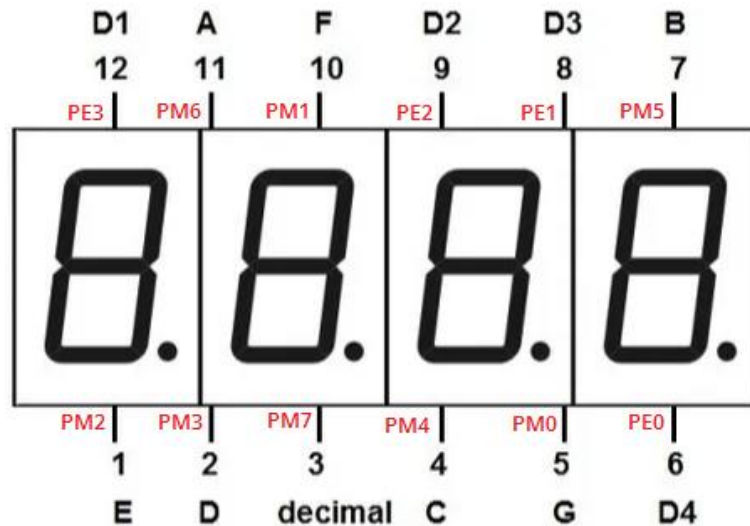


Conexión al display 7 segmentos

Para la conexión del display 7 segmentos se utilizó el puerto M y el puerto E, un total de 12 pines conectados, 6 y 4 respectivamente. En este caso el display 7 segmentos tiene la siguiente configuración:

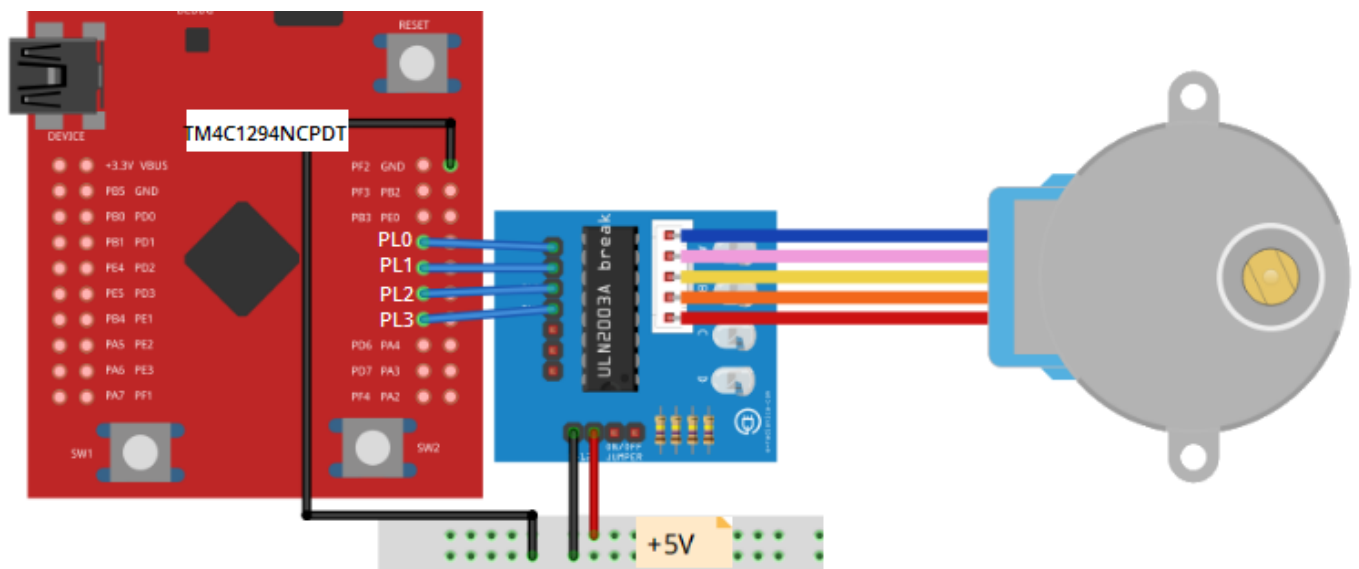


Para realizar las conexiones utilicé el puerto E para la habilitación de los dígitos y el puerto M para los segmentos por lo que la conexión realizada al microcontrolador fue la siguiente.





Conexión al motor

Para conectar el motor se habilitó el puerto L con 4 salidas las cuales son de PL0 a PL3, se realizaron las siguientes conexiones:



Una vez alambrados los 4 puertos se puede cargar el programa y conseguir el correcto funcionamiento del proyecto. Se necesita como se muestra el diagrama una fuente de 5V de DC para el motor.

VII. Resultados y – Conclusiones

	UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores	Semestre: 2020-2	
	Prof. Dr. Saúl De La Rosa Nieves	Grupo 3 Página 25 de 25	

Este proyecto ayudó a poder programar en C con más facilidad y practicar como controlar el microcontrolador usando este lenguaje. En general se consiguió el objetivo del proyecto el cual fue controlar un ángulo dado con sentido y velocidad dados. Se tiene una forma de ingreso de datos que es el teclado y dos de salida de este los cuales fueron el display 7 segmentos y el motor a pasos.

En cuanto a resultados se obtuvo un código capaz de dar sentido y velocidad, así como los grados de giro de un motor dentro de sus limitantes de operación ya que si pides una velocidad muy alta el motor no será capaz de ejecutarlo, aunque el teclado si lea el valor. Cabe mencionar que el código que se implementó no es tan robusto, por lo que, aunque fueron programados ciertos casos donde un error no rompe el código, hay situaciones en las que el código se rompe y esta es si le ingresas una velocidad demasiado lenta o demasiado rápida. Fuera de eso el programa funciona y fue una buena práctica para la comprensión del motor a pasos y del microcontrolador en general.