
	UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores	Semestre: 2021-1	
	Prof. Dr. Saúl De La Rosa Nieves	Página 1 de 52	

TAREA-EXAMEN 3	
Título:	Sistema de monitoreo con motor a pasos y reloj en tiempo real
Fecha:	08-02-20
Preparado por:	Fiel Muñoz Teresa Elpidia
Evaluación:	

I. Planteamiento del proyecto

Objetivo:

Diseñe un sistema de monitoreo y alarma que detecte la presencia de una persona, identifique una temperatura ambiente peligrosa y la ausencia de luz. A este proyecto agregue un sistema de control de motor a pasos y una lectura de la fecha en un display de 7 segmentos con el módulo bluetooth HC06 en una aplicación, utilizando el 28BYJ y el DS1307 respectivamente.

Descripción del proyecto:

El sistema deberá ser reconfigurable por medio de un teclado matricial y un “Display de 7 segmentos” de 4 dígitos, mediante los cuales se podrá ajustar los niveles que disparen las señales de alarma (nivel de temperatura, luz y proximidad de la persona).

Aunado a esto, el control del motor y la forma de accionar el sistema se deben hacer mediante una aplicación siguiendo una transmisión bluetooth, el sistema recibirá hasta tres ángulos consecutivos de tres dígitos cada uno con dirección incluida.

II. Requerimientos del proyecto

Para la construcción del proyecto se presentaron los siguientes requerimientos:

Requerimientos de Hardware

1. Como unidad de procesamiento de la caja registradora utilice el microcontrolador TM4C1294NCPDT.
2. El monitoreo de la temperatura se debe realizar con un sensor que proporcione una señal analógica (p. ej. LM35) la cual debe ser digitalizada por el convertidor AD del microcontrolador.
3. El monitoreo de la intensidad de luz se debe realizar con un arreglo que contenga una fotorresistencia y la señal debe ser digitalizada por el convertidor AD del microcontrolador.
4. El monitoreo de presencia se puede realizar con un sensor PIR (p. ej. HC-SR505) y debe atenderse con una interrupción.
5. Teclado matricial como el que se muestra en la Figura 1 para configurar los niveles de monitoreo.



Figura 1.

6. La visualización de los valores que se ingresen y el resultado de la operación se presentarán en un “Display de 7 segmentos” de 4 dígitos como el que se muestra en la Figura 2.

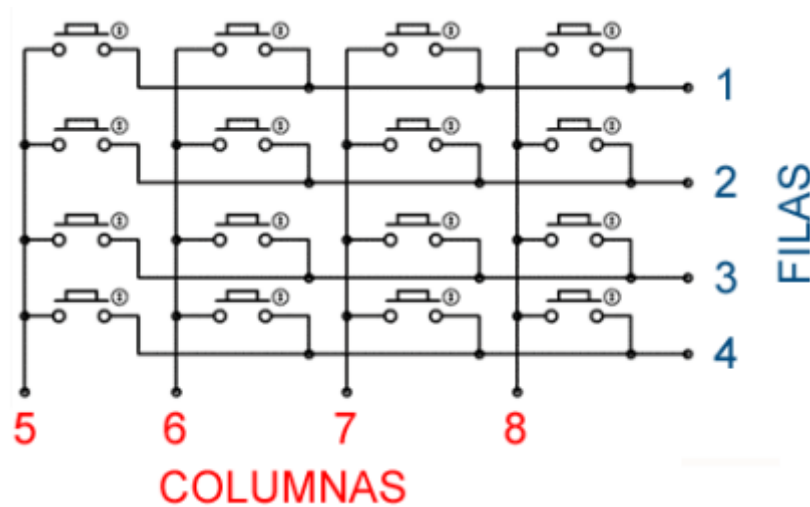


Figura 2.

7. Utilizar un Zumbador Buzzer Pasivo (p. ej. KY-006) para emitir señales sonoras de alarma, una frecuencia para luminosidad, otra para temperatura y otra para presencia.

Requerimientos de funcionamiento

1. Al encender el sistema se podrán configurar los niveles de monitoreo con el teclado matricial. El teclado matricial deberá ser atendido por el microcontrolador mediante interrupciones.
2. Los pasos de configuración y los datos que se ingresen se deberá mostrar en el “Display”
3. Ante la ocurrencia de algún evento de alarma se deberá de emitir la señal sonora correspondiente y en el “Display” se mostrará el motivo de alarma.
4. La alarma se deberá de desactivar con el ingreso de un comando especial por el teclado y el sistema deberá reiniciar su operación normal.
5. El control del motor se realizará con el módulo bluetooth del celular para controlar el motor de pasos en tres ángulos y reiniciarlo.
6. Integrar el RTC para leer la fecha en el display de 4 dígitos y 7 segmentos cuando se reciba un comando bluetooth desde el celular.

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 de los sensores utilizados ya que algunos se utilizan como variables digitales y otras como variables analógicas.

Decodificación de un teclado matricial

Para la decodificación del teclado matricial se debe saber que la lectura de un teclado matricial se realiza por filas y columnas, donde cada tecla presionada representa la unión de una fila con una columna.

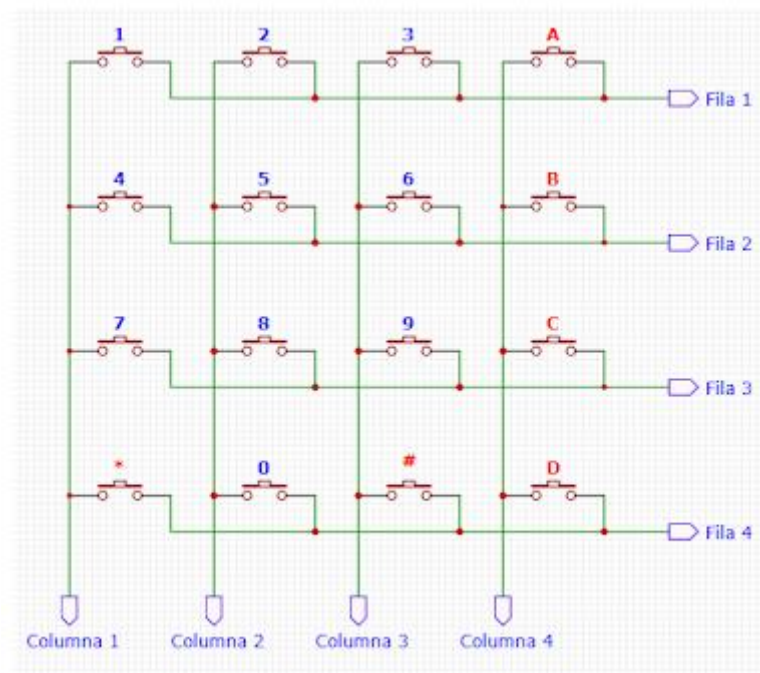


Figura 3. Estructura de un teclado matricial

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:

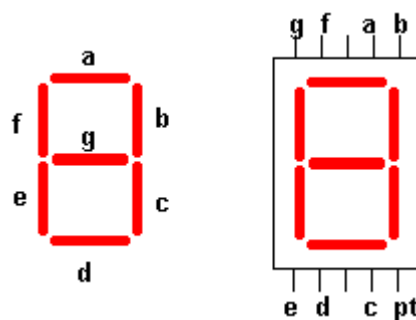


Figura 4. Terminales del display 7 segmentos

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

El LM35 es un sensor de temperatura de buenas prestaciones a un bajo precio. Posee un rango de trabajo desde -55°C hasta 150°C . Su salida es de tipo analógica y lineal con una pendiente de $10\text{mV}/^{\circ}\text{C}$. El sensor es calibrado de fábrica a una precisión de 0.5°C .

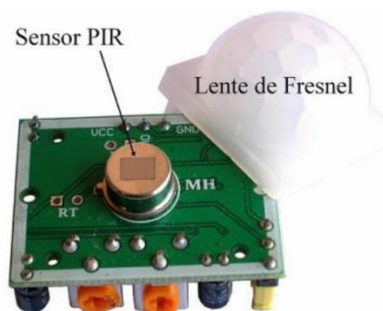
Es un sensor muy popular por su fácil uso y variadas aplicaciones. No necesita de ningún circuito adicional para ser usado. Se alimenta directamente con una fuente de 5V y entrega una salida analógica entre 0V a 1.5V. Este voltaje analógico puede ser leído por el ADC de un microcontrolador como PIC o Arduino. Entre sus aplicaciones podemos encontrar termómetros, termostatos, sistemas de monitoreo y más.

HC-SR505

Principios de funcionamiento:

La radiación infrarroja: Todos los seres vivos e incluso los objetos, emiten radiación electromagnética infrarroja, debido a la temperatura a la que se encuentran. A mayor temperatura, la radiación aumenta. Esta característica ha dado lugar al diseño de sensores de infrarrojo pasivos, en una longitud de onda alrededor de los 9.4 micrones, los cuales permiten la detección de movimiento, típicamente de seres humanos ó animales. Estos sensores son conocidos como PIR, y toman su nombre de ‘Pyroelectric Infrared’ ó ‘Passive Infrared’.

El lente de Fresnel: El lente de Fresnel es un encapsulado semiesférico hecho de polietileno de alta densidad cuyo objetivo es permitir el paso de la radiación infrarroja en el rango de los 8 y 14 micrones. El lente detecta radiación en un ángulo con apertura de 110° y, adicionalmente, concentra la energía en la superficie de detección del sensor PIR, permitiendo una mayor sensibilidad del dispositivo.



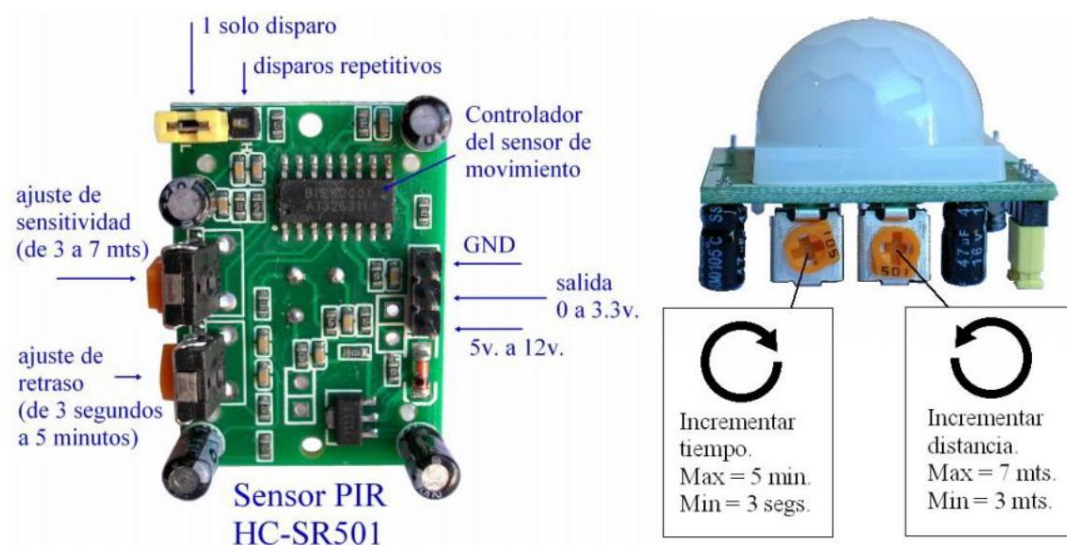
El sensor PIR infrarrojo: En los sensores de movimiento, el sensor PIR consta en realidad de 2 elementos detectores separados, siendo la señal diferencial entre ambos la que permite activar la alarma de movimiento. En el caso del HC-SR501, la señal generada por el sensor ingresa al circuito integrado BISS0001, el cual contiene amplificadores operacionales e interfaces electrónicas adicionales.

Las funciones y ajustes complementarios del sensor de movimiento son:

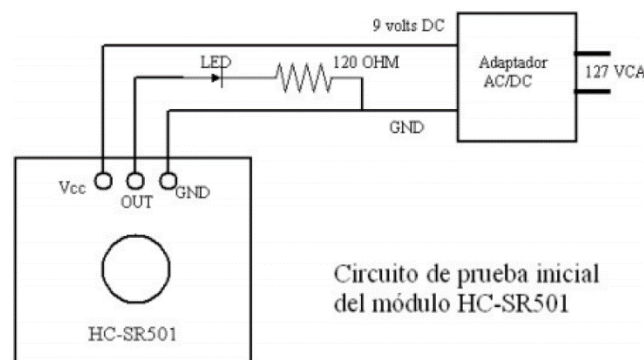
- Ajuste de parámetros: mediante 2 potenciómetros, el usuario puede modificar tanto la sensibilidad como la distancia de detección del PIR.

- Detección automática de luz (esta función no está disponible al adquirir el sensor de fábrica): por medio de una foto resistencia CdS (Sulfuro de Cadmio), se deshabilita la operación del sensor en caso que exista suficiente luz visible en el área. Esta función es utilizada en caso de sensores que enciendan lámparas en lugares poco iluminados durante la noche, y especialmente en corredores ó escaleras.

El módulo PIR modelo HC-SR501 es de bajo costo, pequeño, e incorpora la tecnología más reciente en sensores de movimiento. El sensor utiliza 2 potenciómetros y un jumper que permiten modificar sus parámetros y adaptarlo a las necesidades de la aplicación: sensibilidad de detección, tiempo de activación, y respuesta ante detecciones repetitivas.

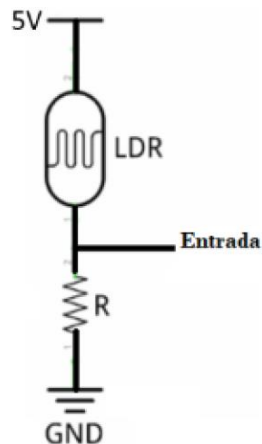


Conexión al ADC



Sensor de Luz

Para el sensor de luz se utilizó únicamente una resistencia, el cual da mayor tensión en la entrada si la luz es mayor y viceversa.



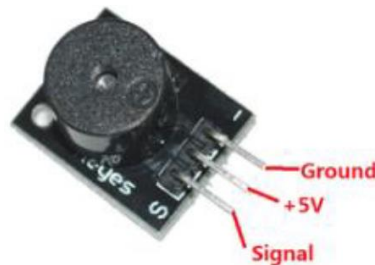
Mayor luz -> Mayor tensión

La resistencia utilizada para la fotoresistencia que conseguí es de 10k, puede variar según la resistencia, lo importante fue obtener valores de salida de 0-5V para la lectura del ADC.

Buzzer Pasivo

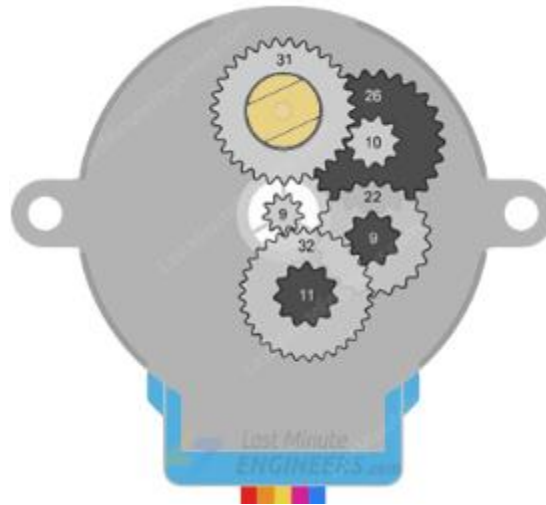
Módulo zumbador piezoeléctrico pasivo Arduino Keyes KY-006, puede producir una gama de tonos de sonido dependiendo de la frecuencia de entrada.

El módulo de zumbador KY-006 consiste en un zumbador piezoeléctrico pasivo, puede generar tonos entre 1.5 a 2.5 kHz al encenderlo y apagarlo a diferentes frecuencias, ya sea mediante retardos o PWM. Este zumbador se utiliza comúnmente para generar alarmas sonoras.



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:

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

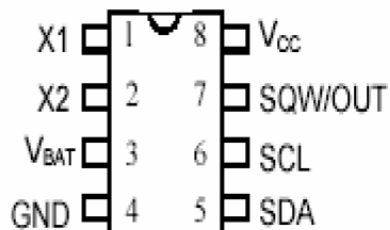
En este caso yo solo utilicé una velocidad constante de 40s por revolución, pero esto puede ajustarse en el preescalador del timer.

DS1307

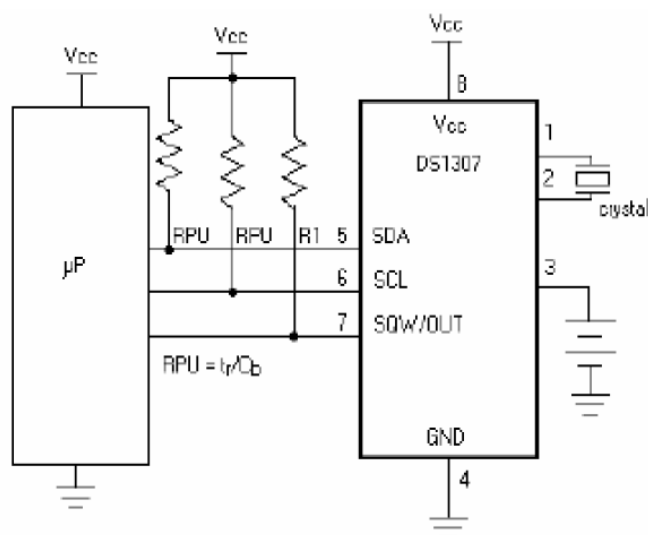
El DS1307 es un RTC cuya interfaz con el microcontrolador se realiza utilizando el bus I2C (Inter-Integrated Circuit), desarrollado por Phillips Semiconductors. El DS1307 es un poderoso reloj calendario en BCD, cuyas características más destacadas son las siguientes:

- Reloj de tiempo real que cuenta los segundos, los minutos, las horas, la fecha, el mes, el día de la semana, y el año, con compensación de años bisiestos, válido hasta el año 2100
- Formato de 12 Horas con indicador AM/PM ó de 24 horas
- Protocolo I2C
- 56 bytes de RAM no volátil, para almacenamiento de datos

- Señal de onda cuadrada programable
- Circuitos internos de respaldo para la alimentación automático
- Bajo consumo de potencia: menor a 500nA en modo respaldo, a 25 grados Centígrados
- Sólo 8 pines



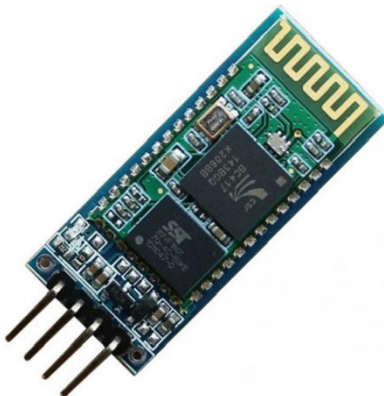
Conexiones típicas a un microprocesador



HC-06

Este módulo te permite agregar conectividad inalámbrica a través de una interfaz serial TTL entre Microcontroladores (PIC, Arduino) y otros dispositivos como PC, laptops o tu Smartphone. El módulo Bluetooth HC-06 viene configurado de fábrica para trabajar como esclavo, es decir, preparado para escuchar peticiones de conexión. Sus características son:

- Voltaje de Operación: 3.3 V / 5 V.
- Corriente de Operación: < 40 mA
- Corriente modo sleep: < 1 mA
- Chip: BC417143
- Alcance 10 metros



IV. Diseño



La primera parte realizada fue toda la decodificación del teclado matricial con el display 7 segmentos, en esta etapa lo que realicé fue leer valores numéricos en todo momento y la posibilidad de configurar cada sensor con teclas diferentes. En este caso mis configuraciones son:

- Tecla A.- Con esta tecla mostramos todo el estado del sistema de alarmas, muestra cada sensor con su nombre y te dice que nivel de frecuencia se configuró, así como cuál es el valor del nivel de monitoreo en el caso del sensor de temperatura y del sensor de presencia. Finalmente muestra si está apagada la alarma. Dado que al mostrar la alarma la única forma de callarla es reiniciando el sistema siempre el estado de los sensores será apagado.
- Tecla B.- Configura el sensor de presencia, solicita frecuencia de buzzer.
- Tecla C.- Configura el sensor de temperatura, solicita frecuencia de buzzer y valor de monitoreo.
- Tecla D.- Configura el sensor de luz, solicita frecuencia de buzzer y valor de monitoreo.
- Tecla #- Es la tecla utilizada para que el buzzer deje de sonar y se reestablezca el sistema.
- Tecla *.- Es la tecla utilizada para que se ingresen los datos en la memoria del programa, tanto para frecuencias como para valores.

La segunda parte del sistema diseñado engloba el uso del motor a pasos y del RTC, ambos utilizan sus propios puertos, pero sus protocolos para funcionamiento se incorporan en el mismo código fuente, el sistema RTC tiene su propio display aparte por si se requiere leer los grados simultáneamente a leer la fecha, y dado que también se muestra el día, mes, año y fecha se implementó una conexión de 14 displays de 7 segmentos.

El control principal de ambos periféricos se hará mediante una aplicación bluetooth, sin embargo, se incorporó también un control por push-buttons para cada acción programada, esto en caso de que cuando se quiera monitorear el sistema por alguna razón no se cuente con el celular a la mano.

A su vez, este sistema está diseñado para poder ingresar datos desde dos fuentes, el teclado matricial y la aplicación bluetooth, sin embargo, este ingreso de datos solo será útil para los sensores en cuanto a niveles de

	UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores	Semestre: 2021-1	
	Prof. Dr. Saúl De La Rosa Nieves	Página 12 de 52	

monitoreo, es decir, los periféricos de ingreso de datos solo tienen el común el teclado numérico, los botones y las letras del teclado son exclusivas de su periférico.

El motor a pasos está diseñado para recibir hasta tres grados consecutivos y realizarlos mientras se muestran los grados en el display, en ese sentido se le agregaron dos espacios más por si se llegara a necesitar hasta cinco comandos consecutivos, se pueden configurar los grados de hasta tres dígitos enteros en sentido horario u antihorario.

Y el RTC tiene un botón para mostrar o no los datos, este ingreso funciona de manera tal que una vez que se presione el botón mostrará la fecha hasta que este se presione de nuevo.

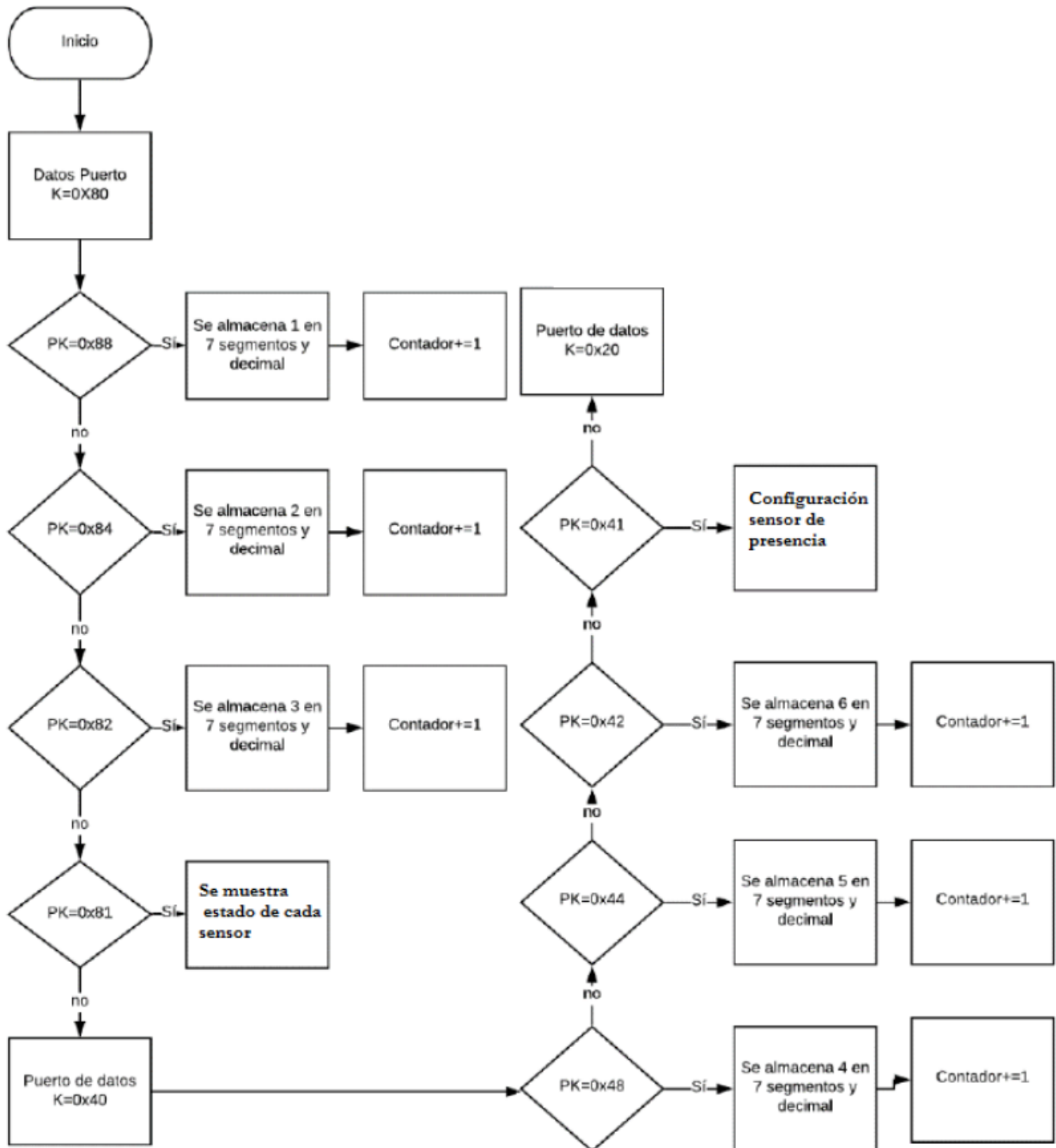
V. Diagrama de flujo y código comentado (en caso de VHDL o código de programa).

Diagramas de flujo

SISTEMA DE MONITOREO

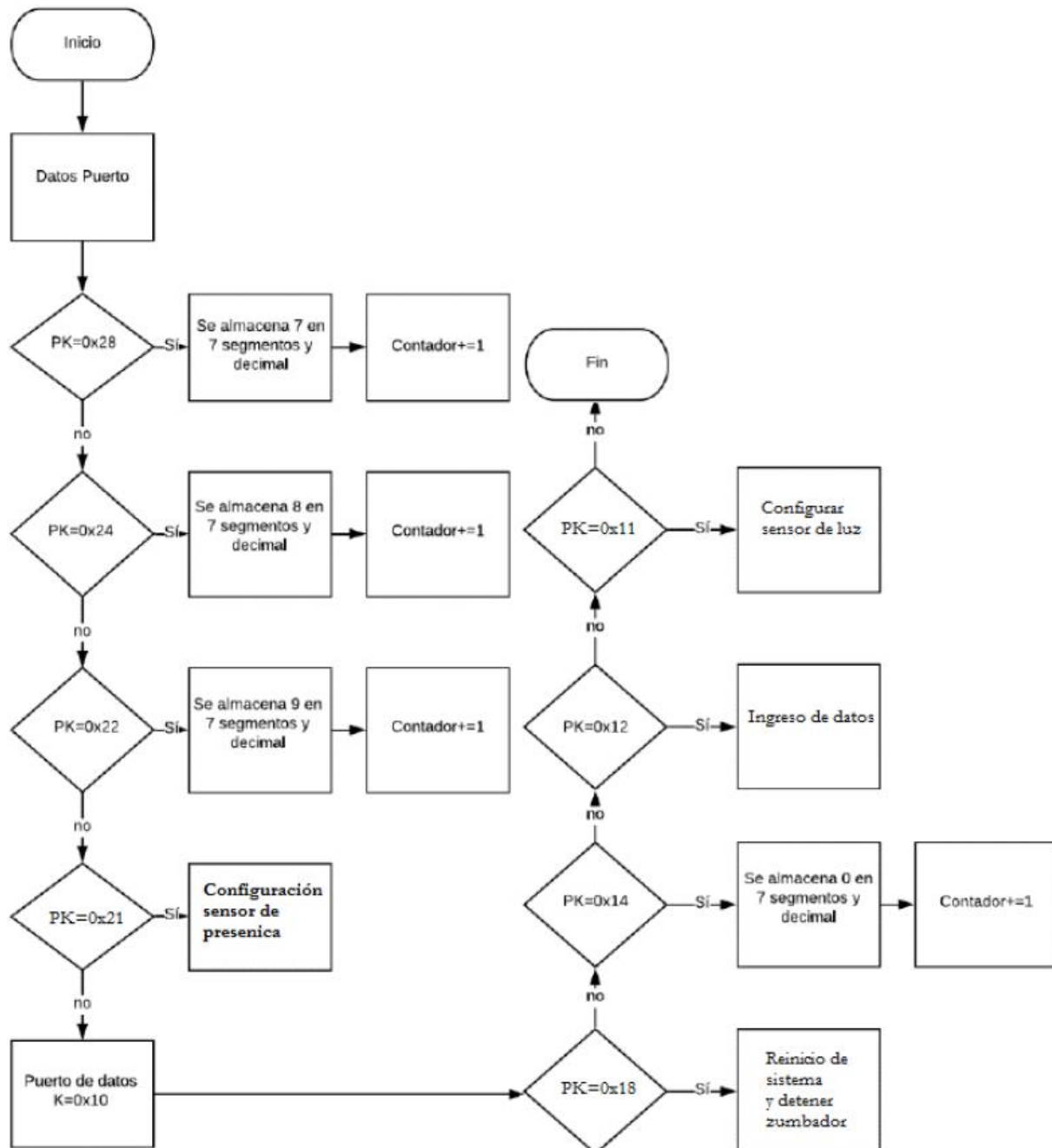


Decodificación teclado matricial primeras dos columnas:



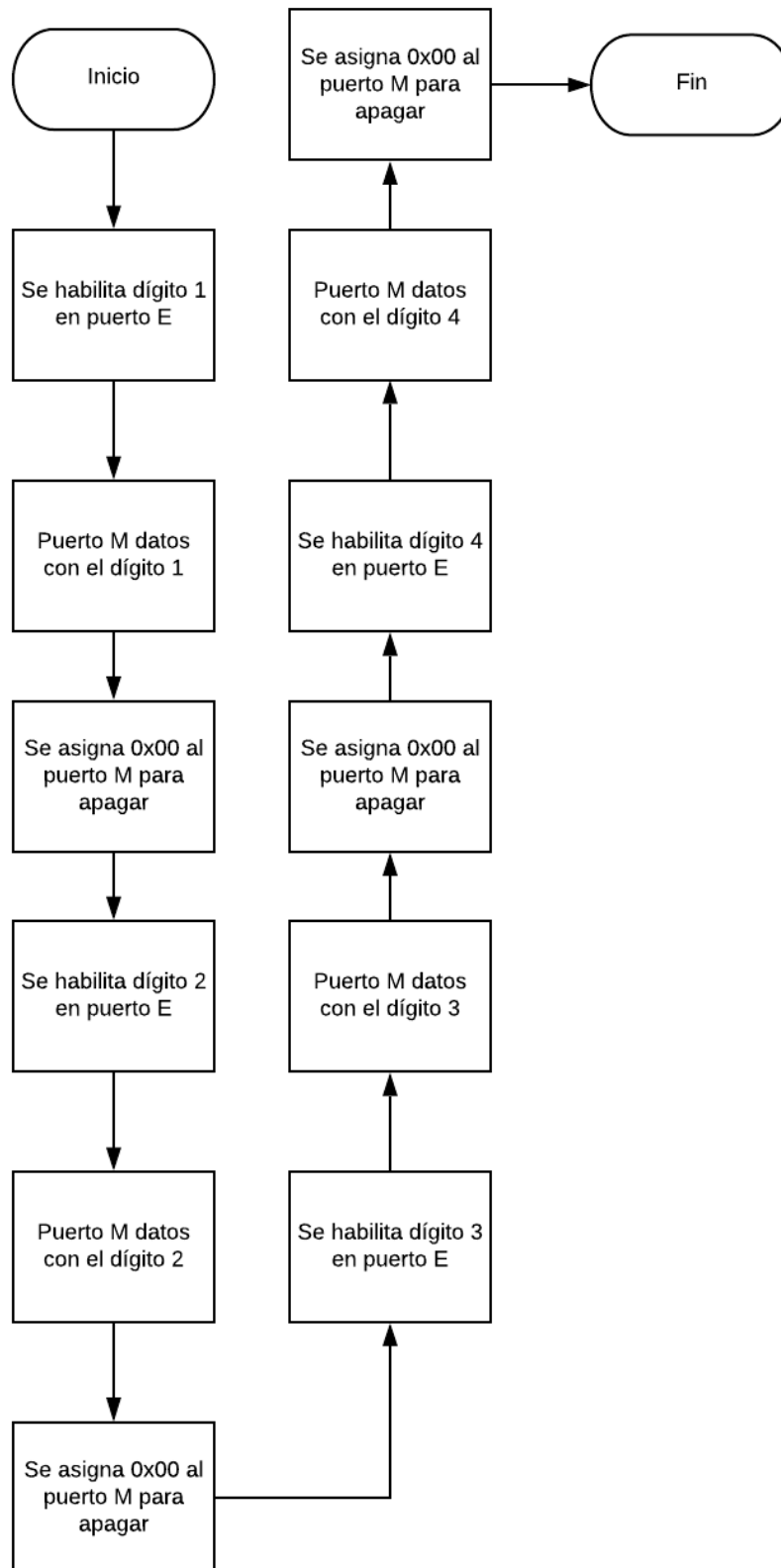


Decodificación teclado matricial últimas dos columnas:



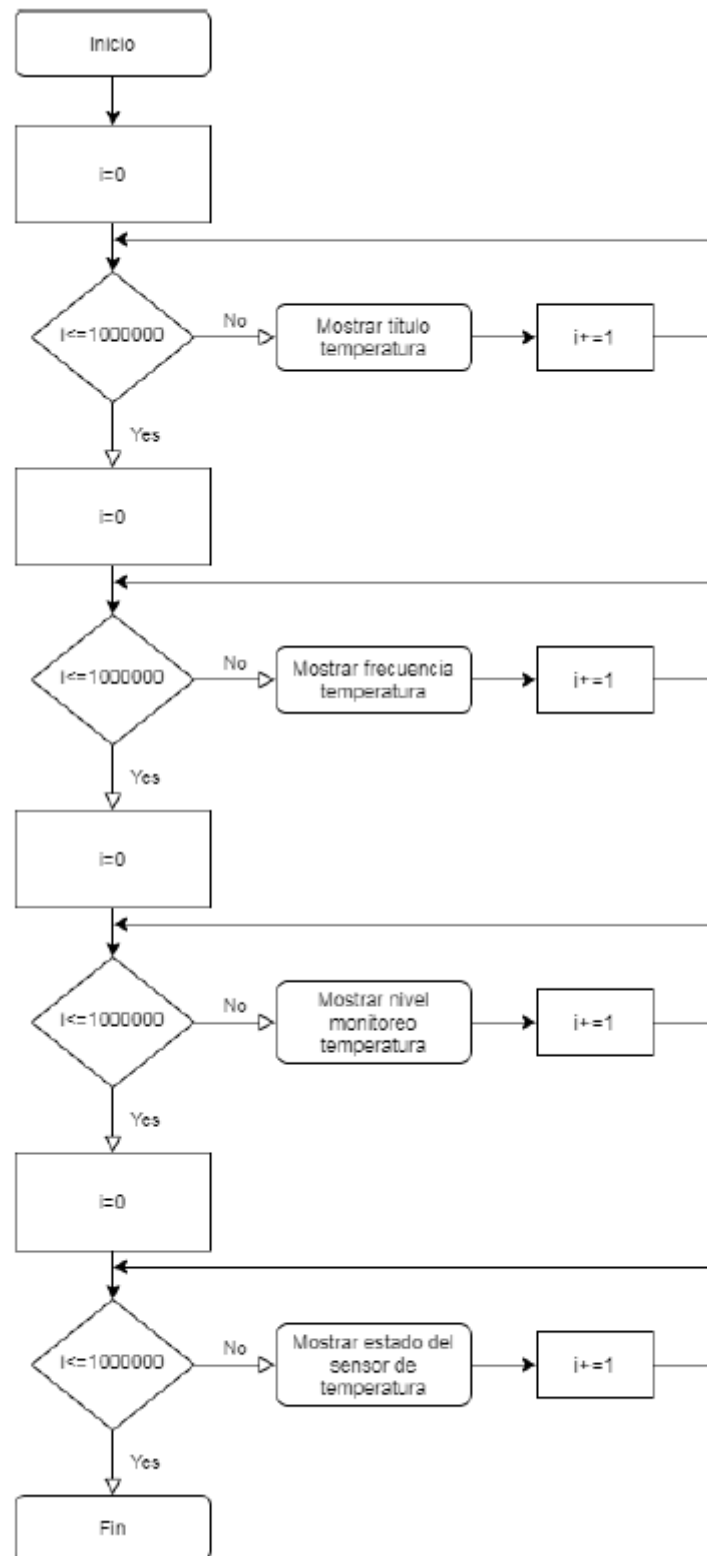


Despliegue del display 7 segmentos



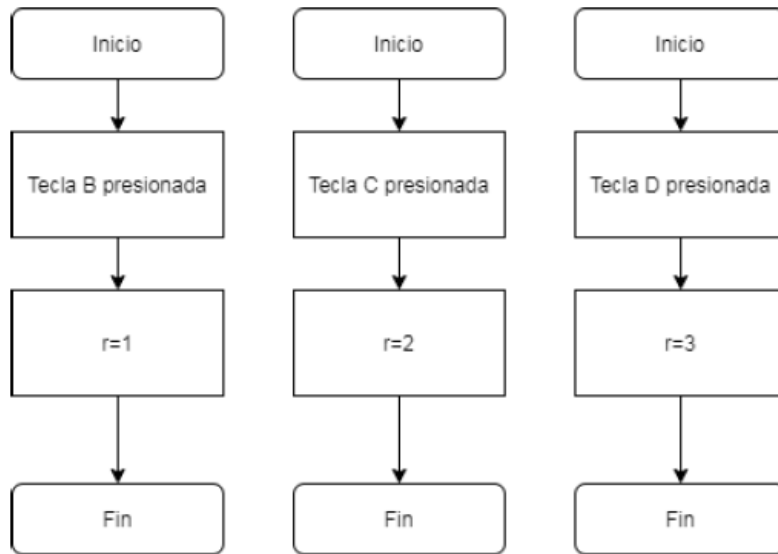


Muestra de datos (Tecla A)

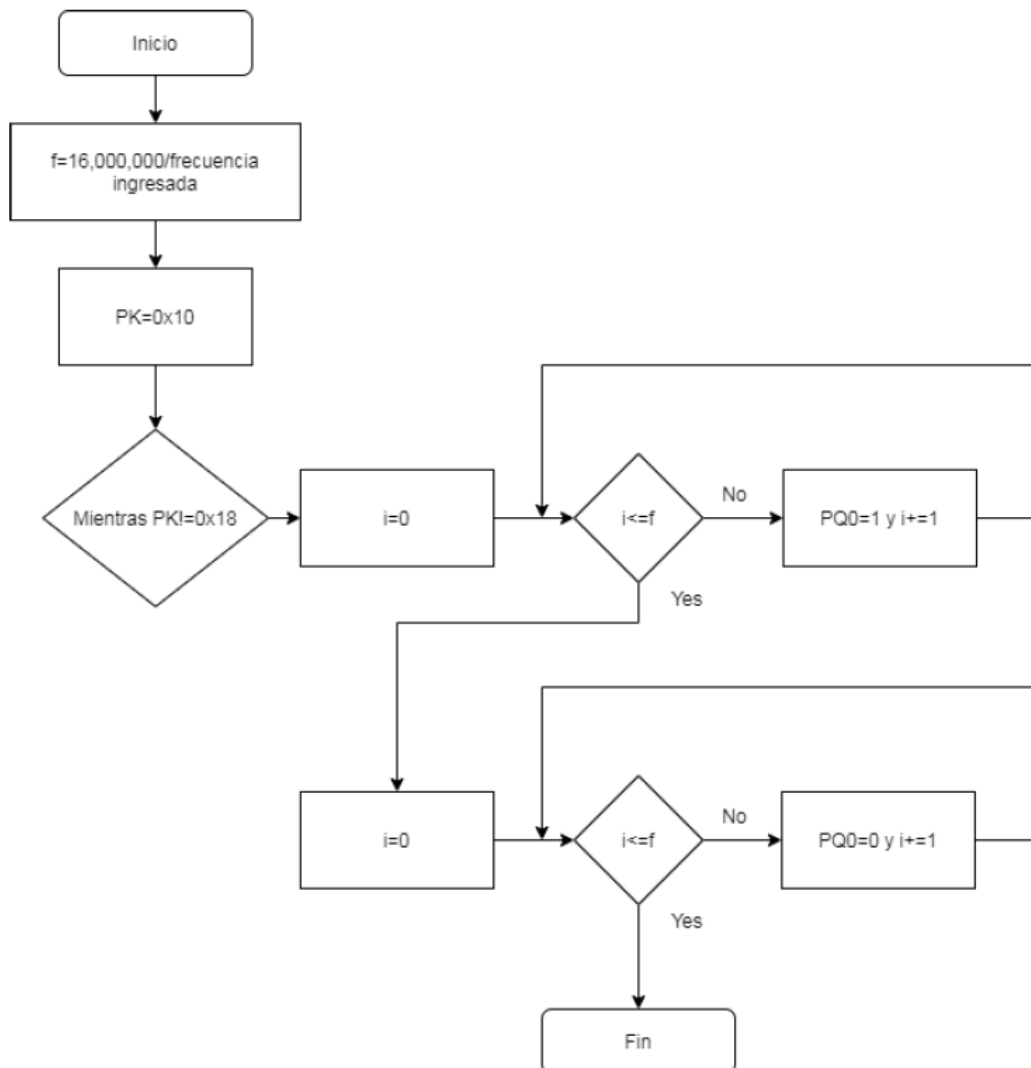




Ingreso de configuración (Teclas B,C,D)

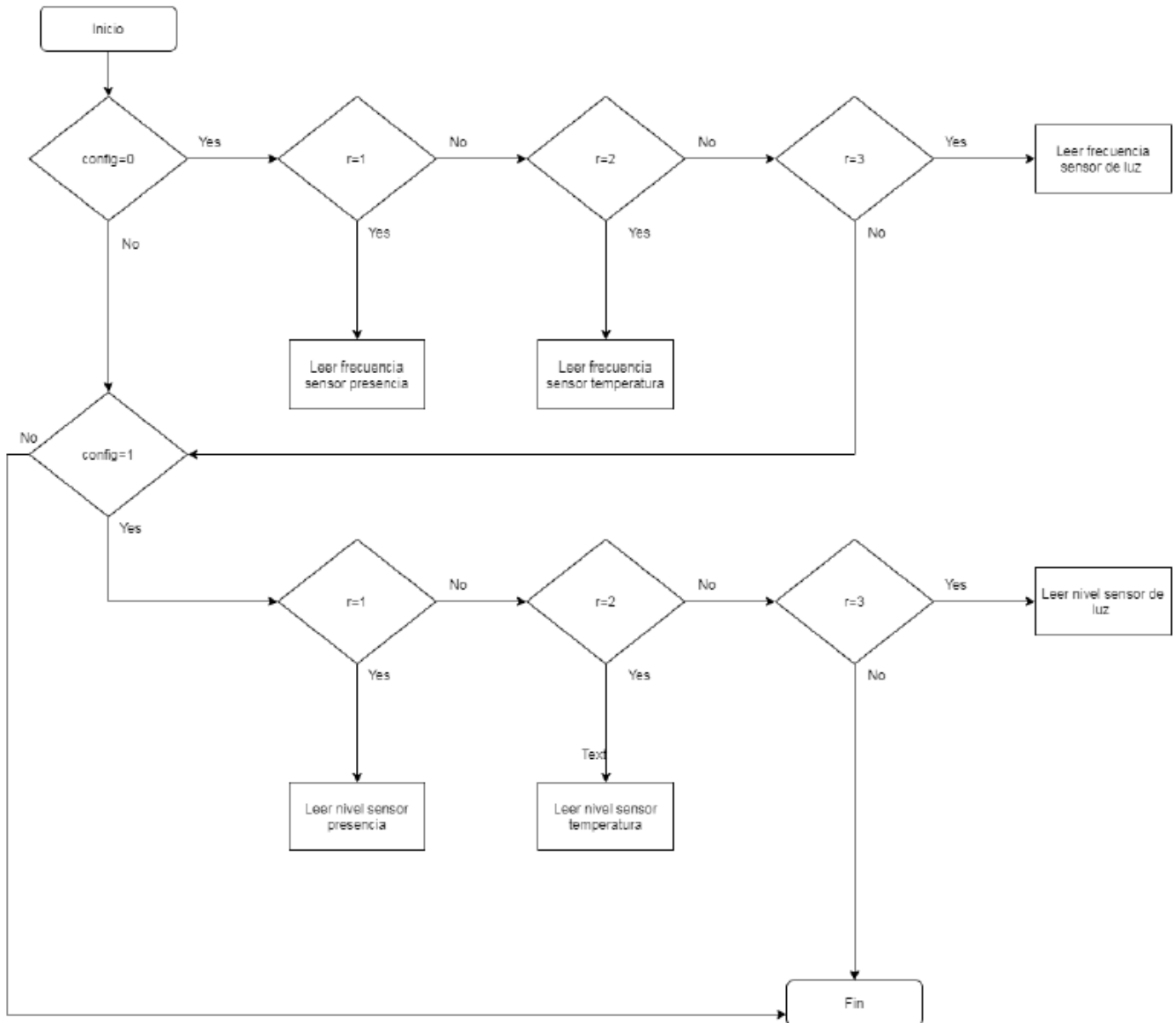


Tecla #





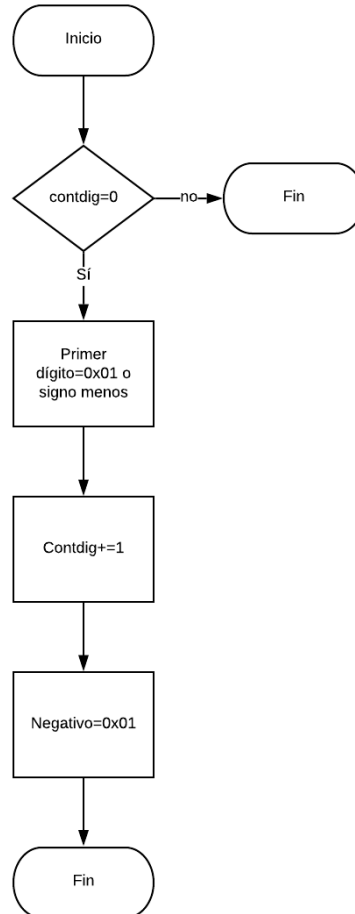
Tecla *



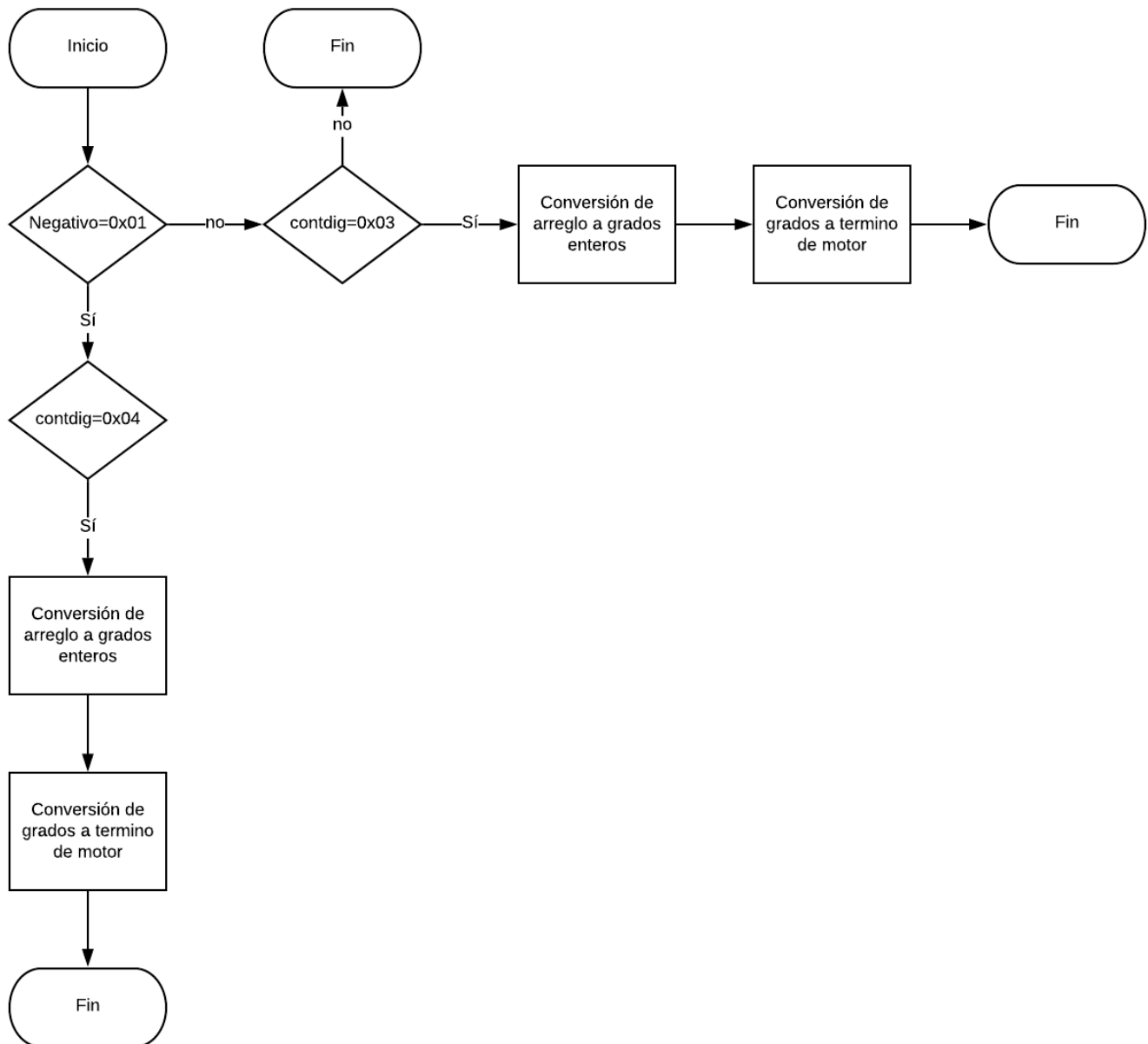


MOTOR A PASOS

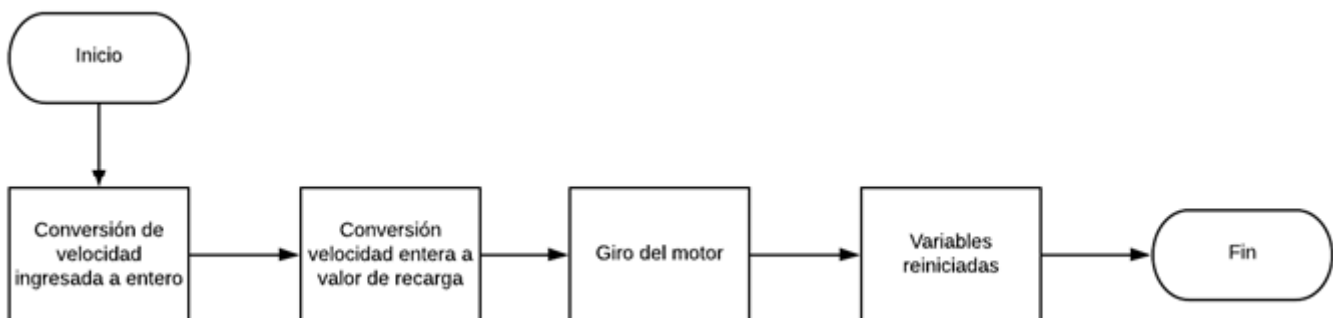
Sentido del giro (horario o antihorario)



Ingreso del ángulo



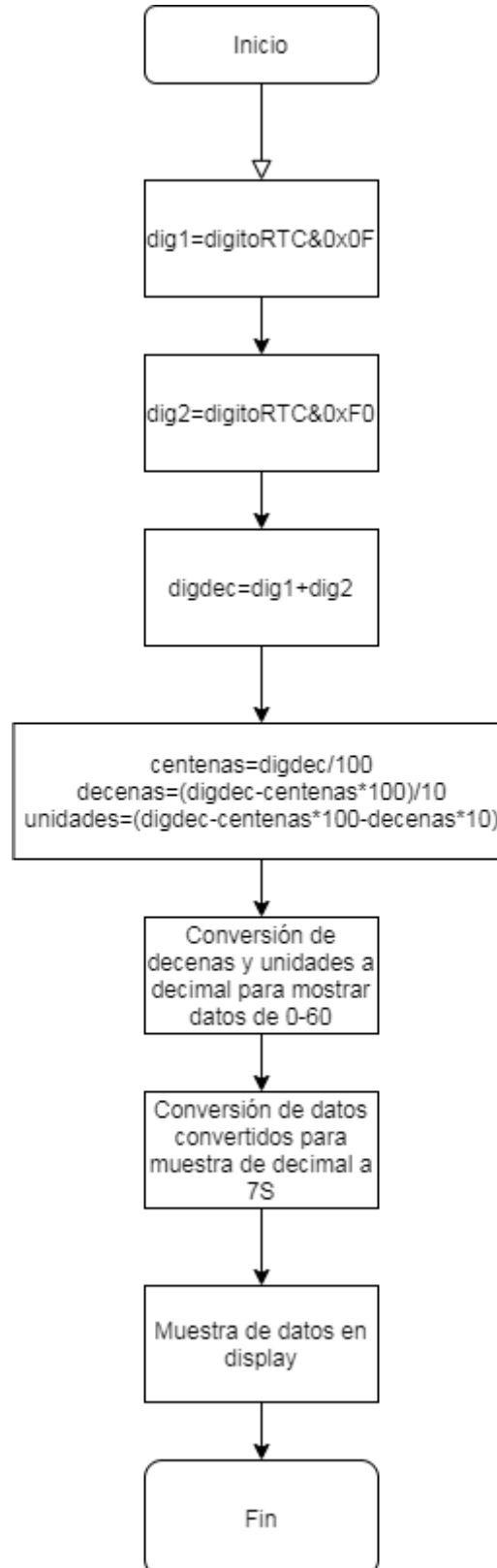
Ejecución de Giro del Motor







RELOJ EN TIEMPO REAL

Conversión de datos del RTC a 7s



	<p align="center">UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores</p>	<p>Semestre: 2021-1</p>	
	<p align="center">Prof. Dr. Saúl De La Rosa Nieves</p>	<p>Página 22 de 52</p>	

Código comentado

```

/*Tarea Examen 4: Motor a pasos y reloj en tiempo real
 * Código creado por: Teresa Fiel Muñoz
 * Microprocesadores y Microcontroladores 2021-1
 */

//-----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-----

//-----ADC-----
#define GPIO_PORTE_AHB_DIR_R      (*((volatile uint32_t *)0x4005C400))
#define GPIO_PORTE_AHB_AFSEL_R    (*((volatile uint32_t *)0x4005C420)) //Registro para habilitar funciones alternativas
del de GPIO (p.770)
#define GPIO_PORTE_AHB_DEN_R      (*((volatile uint32_t *)0x4005C51C)) //Registro para habilitar funciones digital del
de GPIO (p.781)
#define GPIO_PORTE_AHB_AMSEL_R    (*((volatile uint32_t *)0x4005C528))
#define SYSCTL_RCGCADC_R          (*((volatile uint32_t *)0x400FE638)) // Registro para habilitar el reloj al ADC(p.
396)
#define SYSCTL_PRADC_R            (*((volatile uint32_t *)0x400FEA38)) // Registro para verificar si el ADC esta listo
(p.515)
#define ADC0_PC_R                 (*((volatile uint32_t *)0x40038FC4)) // Registro para configurar tasa de muestreo
(p.1159)
#define ADC0_SS PRI_R             (*((volatile uint32_t *)0x40038020)) // Registro para configurar la prioridad del
secuenciador (p.1099)
#define ADC0_ACTSS_R              (*((volatile uint32_t *)0x40038000)) // Registro para controlar la activación del
secuenciador (p. 1076)
#define ADC0_EMUX_R              (*((volatile uint32_t *)0x40038014)) // Registro para seleccionar el evento (trigger)
que inicia el muestreo en cada secuenciador (p.1091)
#define ADC0_SSEMUX2_R            (*((volatile uint32_t *)0x40038098)) // Registro que selecciona entre las entradas
AIN[19:16] o AIN[15:0] (p.1137)
#define ADC0_SSMUX2_R            (*((volatile uint32_t *)0x40038080)) // Registro para configurar la entrada analógica
para el Secuenciador 2 (p.1128)
#define ADC0_SSCTL2_R            (*((volatile uint32_t *)0x40038084)) // Registro que configura la muestra ejecutada
con el Secuenciador 2 (p.1142)
#define ADC0_IM_R                (*((volatile uint32_t *)0x40038008)) // Registro que controla la mascara de
interrupciones en secuenciadores (p. 1081)
#define ADC0_ACTSS_R             (*((volatile uint32_t *)0x40038000)) // Registro que controla la activación de los
secuenciadores (p.1077)
#define ADC0_ISC_R               (*((volatile uint32_t *)0x4003800C)) //Registro de estatus y para borrar las
condiciones de interrupción del secuenciador (p. 1084)
#define ADC0_PSSI_R              (*((volatile uint32_t *)0x40038028)) //Registro que permite al software iniciar el
muestreo en los secuenciadores (p. 1102)
#define ADC0_RIS_R               (*((volatile uint32_t *)0x40038004)) //Registro muestra el estado de la señal de
interrupción de cada secuenciador (p.1079)
#define ADC0_SSIF02_R            (*((volatile uint32_t *)0x40038088)) //Registro que contiene los resultados de
conversión de las muestras recogidas con el secuenciador (p. 1118)



#define ADC0_SAC_R               (*((volatile uint32_t *)0x40038030)) //Registro de control de muestras a promediar (p.
1105)
#define ADC0_CTL_R               (*((volatile uint32_t *)0x40038038))
#define ADC0_SSOP2_R             (*((volatile uint32_t *)0x40038090))
#define ADC0_SSTSH2_R            (*((volatile uint32_t *)0x4003809C))

#define SYSCTL_PLLFREQ0_R         (*((volatile uint32_t *)0x400FE160)) //Registro para configurar el PLL
#define SYSCTL_PLLSTAT_R         (*((volatile uint32_t *)0x400FE168)) //Registro muestra el estado de encendido del PLL
#define SYSCTL_PLLFREQ0_PLLPWR  0x00800000 // Valor para encender el PLL

//-----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

```


	<p>UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores</p>		Semestre: 2021-1	 <p>INGENIERÍA ELÉCTRICA ELECTRÓNICA</p>
	<p>Prof. Dr. Saúl De La Rosa Nieves</p>	<p>Grupo 3</p>	Página 23 de 52	

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

```
#define GPIO_PORTM_DATA_R      (*((volatile unsigned long *)0x40063FC)) // 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_PORTH_DATA_R      (*((volatile unsigned long *)0x4005F3FC)) // Registro de Datos Puerto H
#define GPIO_PORTH_DIR_R      (*((volatile unsigned long *)0x4005F400)) // Registro de Dirección Puerto H
#define GPIO_PORTH_DEN_R      (*((volatile unsigned long *)0x4005F51C)) // Registro de Habilitación Puerto H
#define GPIO_PORTH_PDR_R      (*((volatile unsigned long *)0x4005F510)) // Registro de Pull-Up Puerto H
```

//-----ENTRADAS MOTOR A PASOS-----

```
#define GPIO_PORTP_DATA_R      (*((volatile unsigned long *)0x400653FC)) // Registro de Datos Puerto P
#define GPIO_PORTP_DIR_R      (*((volatile unsigned long *)0x40065400)) // Registro de Dirección Puerto P
#define GPIO_PORTP_DEN_R      (*((volatile unsigned long *)0x4006551C)) // Registro de Habilitación Puerto P
#define GPIO_PORTP_PDR_R      (*((volatile unsigned long *)0x40065514)) // Registro de Pull-Down Puerto P
```

//-----BUZZER PASIVO-----

```
#define GPIO_PORTQ_DATA_R      (*((volatile unsigned long *)0x400663FC)) // Registro de Datos Puerto Q
#define GPIO_PORTQ_DIR_R      (*((volatile unsigned long *)0x40066400)) // Registro de Dirección Puerto Q
#define GPIO_PORTQ_DEN_R      (*((volatile unsigned long *)0x4006651C)) // Registro de Habilitación Puerto Q
#define GPIO_PORTQ_PDR_R      (*((volatile unsigned long *)0x40066514)) // Registro de Pull-Down Puerto Q
```

//-----SENSOR DE PRESENCIA-----

```
#define GPIO_PORTA_DIR_R      (*((volatile uint32_t *)0x40058400)) // Registro de Dirección PA
#define GPIO_PORTA_DEN_R      (*((volatile uint32_t *)0x4005851C)) // Registro de Habilitación PA
#define GPIO_PORTA_PDR_R      (*((volatile uint32_t *)0x40058510)) // Registro de Pull-Up PA
#define GPIO_PORTA_DATA_R      (*((volatile uint32_t *)0x40058004)) // Registro de Datos A
#define GPIO_PORTA_IS_R      (*((volatile uint32_t *)0x40058404)) // Registro de configuración de detección de nivel o flanco
#define GPIO_PORTA_IBE_R      (*((volatile uint32_t *)0x40058408)) // Registro de configuración de interrupción por ambos flancos
#define GPIO_PORTA_IEV_R      (*((volatile uint32_t *)0x4005840C)) // Registro de configuración de interrupción por un flanco
#define GPIO_PORTA_ICR_R      (*((volatile uint32_t *)0x4005841C)) // Registro de limpieza de interrupción de flanco en PA
#define GPIO_PORTA_IM_R      (*((volatile uint32_t *)0x40058410)) // Registro de máscara de interrupción PA (p.764)
#define NVIC_EN0_R      (*((volatile uint32_t *)0xE000E100)) // Registro de habilitación de interrupción PA
#define NVIC_PRI0_R      (*((volatile uint32_t *)0xE000E400)) // Registro de prioridad de interrupción
```

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

```
#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_PDR_R      (*((volatile unsigned long *)0x4005C510)) // Registro de Pull-Up Puerto E
```

```
#define GPIO_PORTD_DATA_R      (*((volatile unsigned long *)0x4005B3FC)) // Registro de Datos Puerto D
#define GPIO_PORTD_DIR_R      (*((volatile unsigned long *)0x4005B400)) // Registro de Dirección Puerto D
#define GPIO_PORTD_DEN_R      (*((volatile unsigned long *)0x4005B51C)) // Registro de Habilitación Puerto D
#define GPIO_PORTD_PDR_R      (*((volatile unsigned long *)0x4005B510)) // Registro de Pull-Up Puerto D
```

```
#define GPIO_PORTF_DATA_R      (*((volatile unsigned long *)0x4005D3FC)) // Registro de Datos Puerto F
#define GPIO_PORTF_DIR_R      (*((volatile unsigned long *)0x4005D400)) // Registro de Dirección Puerto F
#define GPIO_PORTF_DEN_R      (*((volatile unsigned long *)0x4005D51C)) // Registro de Habilitación Puerto F
#define GPIO_PORTF_PDR_R      (*((volatile unsigned long *)0x4005D510)) // Registro de Pull-Up Puerto F
```

```
#define GPIO_PORTG_DATA_R      (*((volatile unsigned long *)0x4005E3FC)) // Registro de Datos Puerto G
#define GPIO_PORTG_DIR_R      (*((volatile unsigned long *)0x4005E400)) // Registro de Dirección Puerto G
#define GPIO_PORTG_DEN_R      (*((volatile unsigned long *)0x4005E51C)) // Registro de Habilitación Puerto G
#define GPIO_PORTG_PDR_R      (*((volatile unsigned long *)0x4005E510)) // Registro de Pull-Up Puerto G
```

```
#define GPIO_PORTN_DATA_R      (*((volatile unsigned long *)0x400643FC)) // Registro de Datos Puerto G
#define GPIO_PORTN_DIR_R      (*((volatile unsigned long *)0x40064400)) // Registro de Dirección Puerto G
#define GPIO_PORTN_DEN_R      (*((volatile unsigned long *)0x4006451C)) // Registro de Habilitación Puerto G
#define GPIO_PORTN_PDR_R      (*((volatile unsigned long *)0x40064510)) // Registro de Pull-Up Puerto G
```



```
//REGISTROS DE RELOJ
#define SYSCTL_RCGCGPIO_R      (*((volatile uint32_t *)0x400FE608)) //Reloj del puerto
#define SYSCTL_RCGCI2C_R      (*((volatile uint32_t *)0x400FE620)) //Reloj de I2C
#define SYSCTL_PRGPIO_R      (*((volatile uint32_t *)0x400FEA08)) //Bandera de "Peripheral Ready"

//REGISTROS DEL PUERTO B
#define GPIO_PORTB_DATA_R      (*((volatile uint32_t *)0x400593FC)) //Para los datos del puerto
#define GPIO_PORTB_DIR_R      (*((volatile uint32_t *)0x40059400)) //Para seleccionar función
#define GPIO_PORTB_AFSEL_R      (*((volatile uint32_t *)0x40059420)) //Para seleccionar función alterna
#define GPIO_PORTB_ODR_R      (*((volatile uint32_t *)0x4005950C)) //Para activar el Open Drain
#define GPIO_PORTB_DEN_R      (*((volatile uint32_t *)0x4005951C)) //Para activar función digital
#define GPIO_PORTB_PCTL_R      (*((volatile uint32_t *)0x4005952C)) //Para el control del puerto

//REGISTROS DEL MÓDULO I2C
#define I2C0_MSA_R      (*((volatile uint32_t *)0x40020000)) //I2C Master Slave Address
#define I2C0_MCS_R      (*((volatile uint32_t *)0x40020004)) //I2C Master Control Status
#define I2C0_MDR_R      (*((volatile uint32_t *)0x40020008)) //I2C Master Data Register
#define I2C0_MTPR_R      (*((volatile uint32_t *)0x4002000C)) //I2C Master Time Period
#define I2C0_MCR_R      (*((volatile uint32_t *)0x40020020)) //I2C Master Congirutation Register

/*
El registro I2C Master Control/Status (I2C_MCS_R) tiene:
-Modo READ-ONLY DATUS: los 7 bits menos significativos son:
    7:Clock Time Out Error    6:BUS BUSY        5:IDLE
    4:Arbitration Lost        3:DataAck          2:AdrAck
    1>Error                   0:CONTROLLER BUSY

-Modo WRITE-ONLY CONTROL_ Los 6 bits menos significativos son:
    6:BURST        5:QuickCommand    4:High Speed Enable
    3:ACK          2:STOP             1:START
    0:RUN

*/
#define I2C_MCS_ACK 0x00000008 //Transmitter Acknowledge Enable
#define I2C_MCS_DATAACK 0x00000008 // Data Acknowledge Enable
#define I2C_MCS_ADRACK 0x00000004 // Acknowledge Address
#define I2C_MCS_STOP 0x00000004 // Generate STOP
#define I2C_MCS_START 0x00000002 // Generate START
#define I2C_MCS_ERROR 0x00000002 // Error
#define I2C_MCS_RUN 0x00000001 // I2C Master Enable
#define MAXRETRIES 5 // number of receive attempts before giving up

/**Direcciones del DS1307

int AdreDS1307 =0x068;///Dirección del RTC DS1307
int AdreSec= 0x00;
int AdreMin=0x01;
int decenasd[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int unidadesd[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int segmentosd[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int segmentosu[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int dig1=0x00;
int dig2=0x00;
int digdec=0x00;
int dec1=0x00;
int dec2=0x00;
int dec3=0x00;

/*El cálculo del Time Period Register (TPR) se especifica en la página 1284
Asumiendo un reloj de 16 MHz y un modo de operación estándar (100 kbps):
*/
int TPR = 7;

// Variables para manejar los valores del RTC
uint8_t segundos, minutos, horas, dia, fecha, mes, anio;
uint8_t error;
uint32_t i;

/** Función que inicializa los relojes, el GPIO y el I2C0 **
void I2C_Init(void){
    //CONFIGURACIÓN DE LOS RELOJ
```



```
SYSTCTL_RCGCI2C_R |= 0x0001; // Activamos el reloj de I2C0 [I2C9 I2C8 I2C7 ... I2C0]<--Mapa de RCGCI2C
SYSTCTL_RCGCGPIO_R |= 0x7EFF; // Activamos el reloj GPIO_PORTB mientras se activa el reloj de I2C0
while((SYSTCTL_PRGPIO_R&0X7EB2) == 0){}; //Espero a que se active el reloj del puerto B

//CONFIGURACIÓN DE LOS GPIOs
/*Acorde con la tabla "Signals by function" de la p. 1808:
  el PIN 2 del puerto B (PB2) es el I2C0SCL del I2C0, y
  el PIN 3 del puerto B (PB3) es el I2C0SDA del I2C0
*/
GPIO_PORTB_AFSEL_R |= 0x0C; // Activo la función alterna del PB2 y PB3
GPIO_PORTB_ODR_R |= 0x08; // Activo el OPEN DRAIN para el PB3, ya que el PB2 ya tiene uno por preconfig.
GPIO_PORTB_DIR_R |= 0x0C; //Activo al PB2 y al PB3 como OUTPUT
GPIO_PORTB_DEN_R |= 0x0C; //Activo la función digital de PB3 y PB2
GPIO_PORTE_DIR_R |= 0x0F; // Salidas de PE0-PE3
GPIO_PORTD_DIR_R |= 0x0F; // Salidas de PD0-PD3
GPIO_PORTF_DIR_R |= 0x0E; // Salidas de PF1-PF3
GPIO_PORTG_DIR_R |= 0x03; // Salidas de PG0-PG1
GPIO_PORTN_DIR_R |= 0x10; // Salidas de PN4
GPIO_PORTE_DEN_R |= 0x0F; // Habilita PE0-PE3
GPIO_PORTD_DEN_R |= 0x0F; // Habilita PD0-PD3
GPIO_PORTF_DEN_R |= 0x0E; // Habilita PF1-PF3
GPIO_PORTG_DEN_R |= 0x03; // Habilita PG0-PG1
GPIO_PORTN_DEN_R |= 0x10; // Habilita PN4
GPIO_PORTE_PUR_R |= 0x0F; // Habilita resistencias de pull-up para dígitos de habilitación 7 segmentos
GPIO_PORTD_PUR_R |= 0x0F; // Habilita resistencias de pull-up para dígitos de habilitación 7 segmentos
GPIO_PORTF_PUR_R |= 0x0E; // Habilita resistencias de pull-up para dígitos de habilitación 7 segmentos
GPIO_PORTG_PUR_R |= 0x03; // Habilita resistencias de pull-up para dígitos de habilitación 7 segmentos
GPIO_PORTN_PUR_R |= 0x10; // Habilita resistencias de pull-up para dígitos de habilitación 7 segmentos
```

```
/*
Así como el registro AFSEL indica que se ejecutará una función externa, en el registro PCTL
debemos indicar qué función alterna se realizará acorde con la tabla 26-5 de la p.1808 e indicarlo
en el correspondiente PCMN (uno por cada bit del puerto) del registro PCTL
*/
GPIO_PORTB_PCTL_R|=0x00002200;
```

```
//CONFIGURACIÓN DEL MODULO I2C0
I2C0_MCR_R = 0x00000010; // Habilitar función MASTER para el I2C0
I2C0_MTPR_R = TPR; // Se establece una velocidad estándar de 100kbps
```

```
}
```

```
// ** Función esperar **
int esperar(){
    while(I2C0_MCS_R&0x00000001){}; //Espero a que la transmisión acabe
    if(I2C0_MCS_R&0x00000002==1){ //¿Hubo error?
        error=1;
        return error;
    };
    return 0;
}
```



```
}
```

```
/** Función para configurar al esclavo RTC DS1307 **
```

```
void CargarFecha(){
    /*
    Programar: Jueves 31 de octubre del 2019, a las 9:40:00 pm
```

El mapa de memoria del DS1207 es el siguiente:

DIRECCIÓN	FUNCIÓN	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
00h	Segundos	0	0	0	0	0	0	0	0
01h	Minutos	0	0	0	0	0	0	0	0
02h	Horas	0	0	0	1	0	0	0	0
03h	Día	0	0	1	1	0	0	0	1
04h	Fecha	0	0	0	0	0	1	0	0
05h	Mes	0	0	0	1	0	0	0	0
06h	Año	0	0	0	1	1	0	0	1
07h	Control	0	0	0	0	0	0	1	1
08h-3Fh	RAM 56x8								

	UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores	Semestre: 2021-1	 INGENIERÍA ELÉCTRICA ELECTRÓNICA
	Prof. Dr. Saúl De La Rosa Nieves Grupo 3	Página 26 de 52	

```

*/
//Por lo tanto

int segundos=0x00, minutos=0x40, horas=0x09, dia=0x05, fecha=0x05, mes=0x02, anio=0x21;
while(I2C0_MCS_R&0x00000001){}; // wait for I2C ready
//Para transmitir
I2C0_MSA_R=(AdreDS1307<<1)&0xFE; //Cargo la dirección del DS1307 e indico "SEND", es decir, el Slave va a recibir
I2C0_MDR_R=AdreSec&0xFF; //Envio la Subdirección( dirección del registro interno "segundos") al DS1307
I2C0_MCS_R=(I2C_MCS_RUN|I2C_MCS_START); // Condición de START y corro
esperar();
for(i=0;i<300;i++){ //Delay

I2C0_MDR_R=segundos; //Envio el valor de "segundos"
I2C0_MCS_R=(I2C_MCS_RUN);
esperar();
for(i=0;i<300;i++){ //Delay

I2C0_MDR_R=minutos; //Envio el valor de "minutos"
I2C0_MCS_R=(I2C_MCS_RUN); //Inicio la transmisión 1
esperar();
for(i=0;i<300;i++){ //Delay

I2C0_MDR_R=horas; //Envio el valor de "horas"
I2C0_MCS_R=(I2C_MCS_RUN); //Inicio la transmisión 2
esperar();
for(i=0;i<300;i++){ //Delay

I2C0_MDR_R=dia; //Envio el valor de "dia"
I2C0_MCS_R=(I2C_MCS_RUN); //Inicio la transmisión 4
esperar();
for(i=0;i<300;i++){ //Delay

I2C0_MDR_R=fecha; //Envio el valor de "fecha"
I2C0_MCS_R=(I2C_MCS_RUN); //Inicio la transmisión 5
esperar();
for(i=0;i<300;i++){ //Delay

I2C0_MDR_R=mes; //Envio el valor de "mes"
I2C0_MCS_R=(I2C_MCS_RUN); //Inicio la transmisión 6
esperar();
for(i=0;i<300;i++){ //Delay

I2C0_MDR_R= anio; //Envio el valor de "año"
I2C0_MCS_R=(I2C_MCS_STOP|I2C_MCS_RUN); //Inicio la ultima transmisión y STOP
esperar();
for(i=0;i<300;i++){ //Delay
}

void leerFecha(){
while(I2C0_MCS_R&0x00000001){}; // wait for I2C ready

//Para actualizar registro para iniciar la lectura
I2C0_MSA_R=(AdreDS1307<<1)&0xFE; //Cargo la dirección del DS1307 e indico "SEND", es decir, el Slave va a
recibir
I2C0_MDR_R=AdreSec&0xFF; //Envio la Subdirección( dirección del registro interno "segundos") al DS1307
I2C0_MCS_R=(I2C_MCS_START|I2C_MCS_RUN); // Condición de START, y corro
esperar();
for(i=0;i<300;i++){ //Delay

//Para recibir información
I2C0_MSA_R=(AdreDS1307<<1)&0xFE; //La dirección del DS1307 en el Master Slave Address
I2C0_MSA_R|=0x01; //Indico "RECIEVE", es decir, el Slave va a transmitir
I2C0_MCS_R=(I2C_MCS_START|I2C_MCS_RUN|I2C_MCS_ACK); // Condición de RESTART, corro, ack
esperar();
for(i=0;i<300;i++){ //Delay
segundos=(I2C0_MDR_R&0xFF); //El Master lee lo que envia el DS1307

I2C0_MCS_R=(I2C_MCS_RUN|I2C_MCS_ACK); // corro, ack
esperar();
for(i=0;i<300;i++){ //Delay
minutos=(I2C0_MDR_R&0xFF); //El Master lee lo que envía el DS1307

```



```
I2C0_MCS_R=(I2C_MCS_RUN|I2C_MCS_ACK); //corro, ack
esperar();
for(i=0;i<300;i++){ //Delay
horas=(I2C0_MDR_R&0xFF); //El Master lee lo que envía el DS1307

I2C0_MCS_R=(I2C_MCS_RUN|I2C_MCS_ACK); // corro, ack
esperar();
for(i=0;i<300;i++){ //Delay
dia=(I2C0_MDR_R&0xFF); //El Master lee lo que envía el DS1307

I2C0_MCS_R=(I2C_MCS_RUN|I2C_MCS_ACK); // corro, ack
esperar();
for(i=0;i<300;i++){ //Delay
fecha=(I2C0_MDR_R&0xFF); //El Master lee lo que envía el DS1307

I2C0_MCS_R=(I2C_MCS_RUN|I2C_MCS_ACK); // corro, ack
esperar();
for(i=0;i<300;i++){ //Delay
mes=(I2C0_MDR_R&0xFF); //El Master lee lo que envía el DS1307

I2C0_MCS_R=(I2C_MCS_RUN|I2C_MCS_ACK); // corro, ack
esperar();
for(i=0;i<300;i++){ //Delay
anio=(I2C0_MDR_R&0xFF); //El Master lee lo que envía el DS1307

I2C0_MCS_R=(I2C_MCS_STOP|I2C_MCS_RUN); //corro, alto
}

//*** PROGRAMA PRINCIPAL ****

int frec,valor,r,config,TEMP,LUZ; // Datos enteros
int muestrack=0;
int alarma[]={0,0,0,0};
int setupp=0;
int setupt=0;
int setupl=0;
int frequen[]={0,0,0,0}; // Datos enteros
int fmuestrap[]={0,0,0,0}; // Datos enteros
int fmuestrat[]={0,0,0,0}; // Datos enteros
int fmuestral[]={0,0,0,0}; // Datos enteros
int vmuestrap[]={0,0,0,0}; // Datos enteros
int vmuestrat[]={0,0,0,0}; // Datos enteros
int vmuestral[]={0,0,0,0}; // Datos enteros
int values[]={0,0,0}; // Datos enteros

int graditos,muestras,p=0,teclaA=0; // Variables de grados en pasos y los
valores registrados de muestras
int digmuestra[]={0,0,0,0}; // Dígitos de muestra del teclado
int negativo=0x00; // Valor entero para saber si es o no
negativo el giro
int vueltas,s=0,x=0;
int negativor[5];
int graditosr[5];
int muestra1[4];
int muestra2[4];
int muestra3[4];
int muestra4[4];
int muestra5[4];
int ejecutar=0;
int ingangulo=0;
int angulonegativo=0;
int puertoP=0;

// 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(vueltas<=1){}
else if(Termino==graditosr[x] && s<=vueltas-2)
{
    x+=1;
    s+=1;
    Termino=0;
    if(x==1)
    {
        digmuestra[0]=muestra2[0];
        digmuestra[1]=muestra2[1];
        digmuestra[2]=muestra2[2];
        digmuestra[3]=muestra2[3];
    }
    else if(x==2)
    {
        digmuestra[0]=muestra3[0];
        digmuestra[1]=muestra3[1];
        digmuestra[2]=muestra3[2];
        digmuestra[3]=muestra3[3];
    }
    else if(x==3)
    {
        digmuestra[0]=muestra4[0];
        digmuestra[1]=muestra4[1];
        digmuestra[2]=muestra4[2];
        digmuestra[3]=muestra4[3];
    }
    else if(x==4)
    {
        digmuestra[0]=muestra5[0];
        digmuestra[1]=muestra5[1];
        digmuestra[2]=muestra5[2];
        digmuestra[3]=muestra5[3];
    }
}
if(Termino==graditosr[x])// 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
    graditosr[0]=0;
    graditosr[1]=0;
    graditosr[2]=0;
    graditosr[3]=0;
    graditosr[4]=0;
}
if (Termino < graditosr[x]) // 32 * 64 = 2048
{

    if(negativor[x]==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;
    }
}
}
}

#define SYSCTL_RCGC2_GPION      0x00007EFF // bit de estado del reloj
#define SYSCTL_RCGC2_R          (*(volatile unsigned long *)0x400FE608) // Registro de Habilitación de Reloj de Puertos

volatile uint32_t Flancosdebajada = 0;
void EdgeCounter_Init(void){
    SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPION; //(a) activa el reloj para puerto L
    Flancosdebajada = 0; // (b) inicializa el contador
    GPIO_PORTA_DIR_R &= ~0x10; // (c) PA4 dirección entrada - boton SW1
    GPIO_PORTA_DEN_R |= 0x10; // PA4 se habilita
    GPIO_PORTA_PUR_R |= 0x10; // habilita weak pull-up on PL1
    GPIO_PORTA_IS_R &= ~0x10; // (d) PA4 es sensible por flanco (p.761)
    GPIO_PORTA_IBE_R &= ~0x10; // PA4 no es sensible a dos flancos (p. 762)
    GPIO_PORTA_IEV_R &= ~0x10; // PA4 detecta eventos de flanco de bajada (p.763)
    GPIO_PORTA_ICR_R = 0x10; // (e) limpia la bandera 0 (p.769)
    GPIO_PORTA_IM_R |= 0x10; // (f) Se desenmascara la interrupcion PA4 y se envia al controlador de interrupciones (p.764)
    NVIC_PRI0_R = (NVIC_PRI0_R&0x00FFFFFF)|0x00000000; // (g) prioridad 0 (p. 159)
    NVIC_EN0_R = 1<<(0-32); // (h) habilita la interrupción 0 en NVIC (p. 154)
}

void GPIOPortA_Handler(void)
{
    GPIO_PORTA_ICR_R = 0x01; // bandera0 de confirmación
    Flancosdebajada = Flancosdebajada + 1;
    /*if(setupp==1)
    {
        alarma[0]=0x67;
        alarma[1]=0x05;
        alarma[2]=0x4F;
        alarma[3]=0x5B;
```


	<div>UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO</div> <div>FACULTAD DE INGENIERÍA</div> <div>Microprocesadores y Microcontroladores</div>		Semestre: 2021-1	 <div>INGENIERÍA</div> <div>ELÉCTRICA ELECTRÓNICA</div>
	Prof. Dr. Saúl De La Rosa Nieves	Grupo 3	Página 30 de 52	

```

    frecuencia(frequen[0]);
    //setupp=0; //Puede o no quitarse
  }*/
  for(i=0;i<=10000;i++){
  }

//-----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    0x00007EFF // bit de estado del reloj puerto E,L,M,P,H

main(void)
{
    I2C_Init(); //Función que inicializa los relojes, el GPIO y el I2C0

    //Inicializo Slave
    while(I2C0_MCS_R&0x00000001){}; // espera que el I2C esté listo

    //Para transmitir

    CargarFecha(); // Función para configurar al esclavo RTC DS1307
    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_PORTQ_DIR_R |= 0x01; // Salida PQ0
    GPIO_PORTK_DIR_R |= 0xF0; // Salidas PK4-PK7 y Entradas PK0-PK3
    GPIO_PORTM_DIR_R |= 0xFF; // Salidas de PM0-PM7
    GPIO_PORTH_DIR_R |= 0x0F; // Salidas de PH0-PH3
    GPIO_PORTP_DIR_R |= 0x00; // Entradas PP0-PP3
    GPIO_PORTQ_DEN_R |= 0x01; // Habilita PKQ0
    GPIO_PORTK_DEN_R |= 0xFF; // Habilita PK0-PK7
    GPIO_PORTM_DEN_R |= 0xFF; // Habilita PM0-PM7
    GPIO_PORTH_DEN_R |= 0x0F; // Habilita PH0-PH3
    GPIO_PORTP_DEN_R |= 0x0F; // Habilita PP0-PP3
    GPIO_PORTQ_PDR_R |= 0x01; // Habilita resistencias de pull-down para BUZZER PASIVO
    GPIO_PORTK_PDR_R |= 0x0F; // Habilita resistencias de pull-down para salidas del teclado matricial
    GPIO_PORTP_PDR_R |= 0x0F; // Habilita resistencias de pull-down para botones del motor
    GPIO_PORTM_PDR_R |= 0xFF; // Habilita resistencias de pull-down para salidas del display 7 segmentos
    GPIO_PORTH_PDR_R |= 0x0F; // Habilita resistencias de pull-up para dígitos de habilitación 7 segmentos

    r=0; //Inicio de configuración
    config=0; //Inicio de configuración
    GPIO_PORTE_AHB_DIR_R = 0x0F; // 2) PE5-4 entradas (analógica)
    GPIO_PORTE_AHB_AFSEL_R |= 0x30; // 3) Habilita Función Alterna en PE5-4 (p. 770)
    GPIO_PORTE_AHB_DEN_R &= ~0x30; // 4) Deshabilita Función Digital en PE5-4 (p. 781)
    GPIO_PORTE_AHB_AMSEL_R |= 0x30; // 5) Habilita Función Analógica de PE5-4 (p. 786)

    SYSCTL_RCGCADC_R |= 0x01; // 6) Habilita reloj para ADC0(p. 396)

    while((SYSCTL_PRADC_R & 0x01) == 0); // Se espera a que el reloj se estabilice

    ADC0_PC_R = 0x01; // 7)Configura para 125Ksamp/s (p.1159)
    ADC0_SSRI_R = 0x0123; // 8)SS3 con la más alta prioridad

    ADC0_ACTSS_R &= ~0x0004; // 9) Deshabilita SS2 antes de cambiar configuración de registros (p. 1076)

```



```
ADC0_EMUX_R = 0x0000;    // 10) Se configura SS2 para disparar muestreo por software (default) (p.1091)
ADC0_SAC_R = 0x0;        // 11) Se configura para no tener sobremuestreo por hardware(default)(p. 1105)
ADC0_CTL_R = 0x0;        //12) Se configura con referencias internas (default VDDA and GNDA) (p. 1107)
ADC0_SSOP2_R = 0x0000;   // 13) Se configure para salvar los resultados del ADC en FIFO (default)(p. 1134)
ADC0_SSTSH2_R = 0x000;   // 14) Se configure el ADC para un periodo de 4 S&H (default) (p. 1134)

ADC0_SSMUX2_R = 0x0089;   // 15) Se configura entradas 1°muestra=AIN9 2°muestra=AIN8 (p.1128)
ADC0_SSEMUX2_R &= ~0x0011; // 16) Canales del SS2 para 1°muestra y 2°muestra en AIN(15:0) (p.1137)
ADC0_SSCTL2_R = 0x0060;   // 17) SI: AIN, Habilidad de INR2; No:muestra diferencial (p.1142)
ADC0_IM_R &= ~0x0010;     // 18) Deshabilita interrupción SS2 (p. 1081)
ADC0_ACTSS_R |= 0x0004;   // 19) Habilita SS2 (p. 1076)

SYSCTL_PLLFREQ0_R |= SYSCTL_PLLFREQ0_PLLPWR; // encender PLL
while((SYSCTL_PLLSTAT_R&0x01)==0);           // espera a que el PLL fije su frecuencia
SYSCTL_PLLFREQ0_R &= ~SYSCTL_PLLFREQ0_PLLPWR; // apagar PLL

ADC0_ISC_R = 0x0004; // Se recomienda Limpiar la bandera RIS del ADC0

// data[0] es el segundo resultado (ADC8 (PE5)) 0 to 10 Sensor de LUZ
// data[1] es el primer resultado (ADC9 (PE4)) 0 to 10 Sensor de TEMPERATURA

GPIO_PORTD_DATA_R=0xFF; //Apaga todos los puertos de segmentos para que no se traslape su muestra
GPIO_PORTF_DATA_R=0xFF; //Apaga todos los puertos de segmentos para que no se traslape su muestra
GPIO_PORTG_DATA_R=0xFF; //Apaga todos los puertos de segmentos para que no se traslape su muestra
GPIO_PORTN_DATA_R=0xFF; //Apaga todos los puertos de segmentos para que no se traslape su muestra
GPIO_PORTE_DATA_R=0xFF; //Apaga todos los puertos de segmentos para que no se traslape su muestra

while(1)
{
    leerFecha();
    valor=digreal[3]+digreal[2]*10+digreal[1]*100+digreal[0]*1000; //Calcula dígito total con dígitos ingresados
    puertoP=GPIO_PORTP_DATA_R;
    /*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<=0x1000000;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]=0x60; // Valor del display 7 segmentos
            digreal[contdig]=2; //El dígito real es 2
            contdig+=1; // Se incrementa la cuenta
            for(i=0;i<=0x1000000;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; //El contador se reinicia
        }
        if(contdig<=3){ // Si la cuenta es menor a 3 se ingresa el dato
            digitos[contdig]=0x90; // Valor del display 7 segmentos
            digreal[contdig]=3; //El dígito real es 3
            contdig+=1; // Se incrementa la cuenta
            for(i=0;i<=0x1000000;i++){ // Valor de espera
            }
        }
    }
    else if(GPIO_PORTK_DATA_R==0x80) // Valor 0 del teclado
    {
        if(contdig==4) // Si la cuenta es 4 del display
        {
            contdig=0x00; //El contador se reinicia
        }
        if(contdig<=3){ // Si la cuenta es menor a 3 se ingresa el dato
            digitos[contdig]=0x00; // Valor del display 7 segmentos
            digreal[contdig]=0; //El dígito real es 0
            contdig+=1; // Se incrementa la cuenta
            for(i=0;i<=0x1000000;i++){ // Valor de espera
            }
        }
    }
}
```



```
    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
{
    /*Presencia*/

    for(i=0;i<=1000000;i++)
    {
        GPIO_PORTH_DATA_R=0xFE;// Habilita dígito 4
        GPIO_PORTM_DATA_R=0x5B;//Enciende dígito 4 S
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFD;// Habilita dígito 3
        GPIO_PORTM_DATA_R=0x4F;//Enciende dígito 3 E
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFB;// Habilita dígito 2
        GPIO_PORTM_DATA_R=0x05;//Enciende dígito 2 R
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xF7;// Habilita dígito 1
        GPIO_PORTM_DATA_R=0x67;//Enciende dígito 1 P
        GPIO_PORTM_DATA_R=0x00;
    }
    for(i=0;i<=1000000;i++)
    {
        GPIO_PORTH_DATA_R=0xFE;// Habilita dígito 4
        GPIO_PORTM_DATA_R=0xF7;//Enciende dígito 4 F
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFD;// Habilita dígito 3
        GPIO_PORTM_DATA_R=0xF7;//Enciende dígito 3 F
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFB;// Habilita dígito 2
        GPIO_PORTM_DATA_R=0x07;//Enciende dígito 2 O
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xF7;// Habilita dígito 1
        GPIO_PORTM_DATA_R=0x00;//Enciende dígito 1
        GPIO_PORTM_DATA_R=0x00;
    }
}

    /*Temperatura*/

    for(i=0;i<=1000000;i++)
    {
        GPIO_PORTH_DATA_R=0xFE;// Habilita dígito 4
        GPIO_PORTM_DATA_R=0x67;//Enciende dígito 4 P
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFD;// Habilita dígito 3
        GPIO_PORTM_DATA_R=0x37;//Enciende dígito 3 M
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFB;// Habilita dígito 2
        GPIO_PORTM_DATA_R=0x4F;//Enciende dígito 2 E
```



```
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
GPIO_PORTM_DATA_R=0x0F; // Enciende dígito 1 T
GPIO_PORTM_DATA_R=0x00;
}
for(i=0; i<=1000000; i++)
{
    GPIO_PORTH_DATA_R=0xFE; // Habilita dígito 4
    GPIO_PORTM_DATA_R=fmustrat[3]; // Enciende dígito 4
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xFD; // Habilita dígito 3
    GPIO_PORTM_DATA_R=fmustrat[2]; // Enciende dígito 3
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xFB; // Habilita dígito 2
    GPIO_PORTM_DATA_R=fmustrat[1]; // Enciende dígito 2
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
    GPIO_PORTM_DATA_R=fmustrat[0]; // Enciende dígito 1
    GPIO_PORTM_DATA_R=0x00;
}
for(i=0; i<=1000000; i++)
{
    GPIO_PORTH_DATA_R=0xFE; // Habilita dígito 4
    GPIO_PORTM_DATA_R=vmustrat[3]; // Enciende dígito 4
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xFD; // Habilita dígito 3
    GPIO_PORTM_DATA_R=vmustrat[2]; // Enciende dígito 3
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xFB; // Habilita dígito 2
    GPIO_PORTM_DATA_R=vmustrat[1]; // Enciende dígito 2
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
    GPIO_PORTM_DATA_R=vmustrat[0]; // Enciende dígito 1
    GPIO_PORTM_DATA_R=0x00;
}
for(i=0; i<=1000000; i++)
{
    GPIO_PORTH_DATA_R=0xFE; // Habilita dígito 4
    GPIO_PORTM_DATA_R=0x47; // Enciende dígito 4 F
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xFD; // Habilita dígito 3
    GPIO_PORTM_DATA_R=0x47; // Enciende dígito 3 F
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xFB; // Habilita dígito 2
    GPIO_PORTM_DATA_R=0x7E; // Enciende dígito 2 0
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
    GPIO_PORTM_DATA_R=0x00; // Enciende dígito 1
    GPIO_PORTM_DATA_R=0x00;
}

/*Luz*/

for(i=0; i<=1000000; i++)
{
    GPIO_PORTH_DATA_R=0xFE; // Habilita dígito 4
    GPIO_PORTM_DATA_R=0x6D; // Enciende dígito 4 S
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xFB; // Habilita dígito 3
    GPIO_PORTM_DATA_R=0x0E; // Enciende dígito 3 U
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xFD; // Habilita dígito 2
    GPIO_PORTM_DATA_R=0x3E; // Enciende dígito 2 L
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
    GPIO_PORTM_DATA_R=0x00; // Enciende dígito 1 0
    GPIO_PORTM_DATA_R=0x00;
}
for(i=0; i<=1000000; i++)
{
    GPIO_PORTH_DATA_R=0xFE; // Habilita dígito 4
```



```
GPIO_PORTM_DATA_R=fmuestral[3];//Enciende dígito 4
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTH_DATA_R=0xFD; // Habilita dígito 3
GPIO_PORTM_DATA_R=fmuestral[2];//Enciende dígito 3
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTH_DATA_R=0xFB; // Habilita dígito 2
GPIO_PORTM_DATA_R=fmuestral[1];//Enciende dígito 2
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
GPIO_PORTM_DATA_R=fmuestral[0];//Enciende dígito 1
GPIO_PORTM_DATA_R=0x00;
}
for(i=0;i<=1000000;i++)
{
    GPIO_PORTH_DATA_R=0xFE; // Habilita dígito 4
    GPIO_PORTM_DATA_R=vmuestral[3];//Enciende dígito 4
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xFD; // Habilita dígito 3
    GPIO_PORTM_DATA_R=vmuestral[2];//Enciende dígito 3
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xFB; // Habilita dígito 2
    GPIO_PORTM_DATA_R=vmuestral[1];//Enciende dígito 2
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
    GPIO_PORTM_DATA_R=vmuestral[0];//Enciende dígito 1
    GPIO_PORTM_DATA_R=0x00;
}
for(i=0;i<=1000000;i++)
{
    GPIO_PORTH_DATA_R=0xFE; // Habilita dígito 4
    GPIO_PORTM_DATA_R=0x47; //Enciende dígito 4 F
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xFD; // Habilita dígito 3
    GPIO_PORTM_DATA_R=0x47; //Enciende dígito 3 F
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xFB; // Habilita dígito 2
    GPIO_PORTM_DATA_R=0x7E; //Enciende dígito 2 0
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
    GPIO_PORTM_DATA_R=0x00; //Enciende dígito 1
    GPIO_PORTM_DATA_R=0x00;
}
}

/* 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 aun 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
{
    for(i=0;i<=1000000;i++)
    {
        GPIO_PORTH_DATA_R=0xFE; // Habilita dígito 4
        GPIO_PORTM_DATA_R=0x5B; // Enciende dígito 4 S
        GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFD; // Habilita dígito 3
        GPIO_PORTM_DATA_R=0x4F; // Enciende dígito 3 E
        GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFB; // Habilita dígito 2
        GPIO_PORTM_DATA_R=0x05; // Enciende dígito 2 R
        GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
        GPIO_PORTM_DATA_R=0x67; // Enciende dígito 1 P
        GPIO_PORTM_DATA_R=0x00;
    }
    r=1; // Presencia
    for(i=0;i<=0x100000;i++){ // Espera
    }
}

/* 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
        }
    }
}
else if(GPIO_PORTK_DATA_R==0x21)//Tecla C presionada
{
    for(i=0;i<=1000000;i++)
    {
        GPIO_PORTH_DATA_R=0xFE;// Habilita dígito 4
        GPIO_PORTM_DATA_R=0x67;//Enciende dígito 4 P
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFD;// Habilita dígito 3
        GPIO_PORTM_DATA_R=0x37;//Enciende dígito 3 M
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFB;// Habilita dígito 2
        GPIO_PORTM_DATA_R=0x4F;//Enciende dígito 2 E
        GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xF7;// Habilita dígito 1
        GPIO_PORTM_DATA_R=0x0F;//Enciende dígito 1 T
        GPIO_PORTM_DATA_R=0x00;
    }
    r=2; // Temperatura
    for(i=0;i<=0x100000;i++){//Aguanta
    }

    /* 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 7segm
            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(config==0)
        {
            if(r==1)
            {
                for(i=0;i<=1000000;i++)
                {
                    GPIO_PORTH_DATA_R=0xFE;// Habilita dígito 4
                    GPIO_PORTM_DATA_R=0x47;//Enciende dígito 4 F
                    GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
                    GPIO_PORTH_DATA_R=0xFD;// Habilita dígito 3
                    GPIO_PORTM_DATA_R=0x00;//Enciende dígito 3 F
                    GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
                    GPIO_PORTH_DATA_R=0xFB;// Habilita dígito 2
                    GPIO_PORTM_DATA_R=0x00;//Enciende dígito 2 O
                    GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
                    GPIO_PORTH_DATA_R=0xF7;// Habilita dígito 1
                    GPIO_PORTM_DATA_R=0x00;//Enciende dígito 1
                    GPIO_PORTM_DATA_R=0x00;
```




```
}
frecuen[0]=valor; //Frecuencia igual al valor ingresado
fmuestrap[0]=digitos[0]; //Guarda valor en frecuencia de sensor
fmuestrap[1]=digitos[1]; //Guarda valor en frecuencia de sensor
fmuestrap[2]=digitos[2]; //Guarda valor en frecuencia de sensor
fmuestrap[3]=digitos[3]; //Guarda valor en frecuencia de sensor
setup=1;
}
else if(r==2)
{
    for(i=0;i<=1000000;i++)
    {
        GPIO_PORTH_DATA_R=0xFE; // Habilita dígito 4
        GPIO_PORTM_DATA_R=0x47; //Enciende dígito 4 F
        GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFD; // Habilita dígito 3
        GPIO_PORTM_DATA_R=0x00; //Enciende dígito 3 F
        GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFB; // Habilita dígito 2
        GPIO_PORTM_DATA_R=0x00; //Enciende dígito 2 0
        GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
        GPIO_PORTM_DATA_R=0x00; //Enciende dígito 1
        GPIO_PORTM_DATA_R=0x00;

    }
    frecuen[1]=valor; //Frecuencia igual al valor ingresado
    fmuestrat[0]=digitos[0]; //Guarda valor en frecuencia de sensor
    fmuestrat[1]=digitos[1]; //Guarda valor en frecuencia de sensor
    fmuestrat[2]=digitos[2]; //Guarda valor en frecuencia de sensor
    fmuestrat[3]=digitos[3]; //Guarda valor en frecuencia de sensor
}
else if(r==3)
{
    for(i=0;i<=1000000;i++)
    {
        GPIO_PORTH_DATA_R=0xFE; // Habilita dígito 4
        GPIO_PORTM_DATA_R=0x47; //Enciende dígito 4 F
        GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFD; // Habilita dígito 3
        GPIO_PORTM_DATA_R=0x00; //Enciende dígito 3 F
        GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFB; // Habilita dígito 2
        GPIO_PORTM_DATA_R=0x00; //Enciende dígito 2 0
        GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
        GPIO_PORTM_DATA_R=0x00; //Enciende dígito 1
        GPIO_PORTM_DATA_R=0x00;

    }
    frecuen[2]=valor; //Frecuencia igual al valor ingresado
    fmuestral[0]=digitos[0]; //Guarda valor en frecuencia de sensor
    fmuestral[1]=digitos[1]; //Guarda valor en frecuencia de sensor
    fmuestral[2]=digitos[2]; //Guarda valor en frecuencia de sensor
    fmuestral[3]=digitos[3]; //Guarda valor en frecuencia de sensor
}
}
else if(config==1)
{
    if(r==2) //TEMPERATURA
    {
        for(i=0;i<=1000000;i++)
        {
            GPIO_PORTH_DATA_R=0xFE; // Habilita dígito 4
            GPIO_PORTM_DATA_R=0x3E; //Enciende dígito 4 S
            GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
            GPIO_PORTH_DATA_R=0xFB; // Habilita dígito 3
            GPIO_PORTM_DATA_R=0x00; //Enciende dígito 3 U
            GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
            GPIO_PORTH_DATA_R=0xFD; // Habilita dígito 2
            GPIO_PORTM_DATA_R=0x00; //Enciende dígito 2 L
            GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
            GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
```



```
        GPIO_PORTM_DATA_R=0x00; //Enciende dígito 1 0
        GPIO_PORTM_DATA_R=0x00;
    }
    values[1]=valor; //Valor igual al valor ingresado
    vmuestrat[0]=digitos[0]; //Guarda valor en frecuencia de sensor
    vmuestrat[1]=digitos[1]; //Guarda valor en frecuencia de sensor
    vmuestrat[2]=digitos[2]; //Guarda valor en frecuencia de sensor
    vmuestrat[3]=digitos[3]; //Guarda valor en frecuencia de sensor
    setup1=1;
}
else if(r==3) //LUZ
{
    for(i=0;i<=1000000;i++)
    {
        GPIO_PORTH_DATA_R=0xFE; //Habilita dígito 4
        GPIO_PORTM_DATA_R=0x3E; //Enciende dígito 4 S
        GPIO_PORTM_DATA_R=0x00; //Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFB; //Habilita dígito 3
        GPIO_PORTM_DATA_R=0x00; //Enciende dígito 3 U
        GPIO_PORTM_DATA_R=0x00; //Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFD; //Habilita dígito 2
        GPIO_PORTM_DATA_R=0x00; //Enciende dígito 2 L
        GPIO_PORTM_DATA_R=0x00; //Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xF7; //Habilita dígito 1
        GPIO_PORTM_DATA_R=0x00; //Enciende dígito 1 0
        GPIO_PORTM_DATA_R=0x00;
    }
    values[2]=valor; //Valor igual al valor ingresado
    vmuestrat[0]=digitos[0]; //Guarda valor en frecuencia de sensor
    vmuestrat[1]=digitos[1]; //Guarda valor en frecuencia de sensor
    vmuestrat[2]=digitos[2]; //Guarda valor en frecuencia de sensor
    vmuestrat[3]=digitos[3]; //Guarda valor en frecuencia de sensor
    setup1=1;
}
}
config+=1; //Se suma configuración
if(r==1)
{
    if(config==1)
    {
        config=0;
        r=0;
    }
}
if(config==2)
{
    config=0; //Si es igual a 2 r se reinicia
    r=0; //R también se reinicia
}
for(i=0;i<=0x100000;i++){ //Ciclo de espera
}
else if(GPIO_PORTK_DATA_R==0x11) //Tecla D presionada
{
    for(i=0;i<=1000000;i++)
    {
        GPIO_PORTH_DATA_R=0xFE; //Habilita dígito 4
        GPIO_PORTM_DATA_R=0x6D; //Enciende dígito 4 S
        GPIO_PORTM_DATA_R=0x00; //Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFB; //Habilita dígito 3
        GPIO_PORTM_DATA_R=0x0E; //Enciende dígito 3 U
        GPIO_PORTM_DATA_R=0x00; //Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFD; //Habilita dígito 2
        GPIO_PORTM_DATA_R=0x3E; //Enciende dígito 2 L
        GPIO_PORTM_DATA_R=0x00; //Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xF7; //Habilita dígito 1
        GPIO_PORTM_DATA_R=0x00; //Enciende dígito 1 0
        GPIO_PORTM_DATA_R=0x00;
    }
    r=3; //Sensor de Luz
    for(i=0;i<=0x100000;i++){ //Ciclo de espera
}
}
```



```
/* Muestra de datos */
```

```
GPIO_PORTH_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_PORTH_DATA_R=0xFD; // Habilita dígito 3
GPIO_PORTM_DATA_R=digitos[2]; // Enciende dígito 3
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTH_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_PORTH_DATA_R=0xF7; // Habilita dígito 1
GPIO_PORTM_DATA_R=digitos[0]; // Enciende dígito 1
GPIO_PORTM_DATA_R=0x00;
```

```
/*Lectura ADC*/
```

```
uint32_t data[0];
ADC0_PSSI_R = 0x0004; // Inicia conversión del SS2
while((ADC0_RIS_R&0x04)==0); // Espera a que SS2 termine conversión (polling)
data[1] = (ADC0_SSIF02_R&0xFFF); // se lee el primer resultado TEMPERATURA
data[0] = (ADC0_SSIF02_R&0xFFF); // se lee el segundo resultado LUZ
ADC0_ISC_R = 0x0004; // Limpia la bandera RIS del ADC0
```

```
LUZ=data[0]; // Luz se almacena en data0
TEMP=data[1]; // Temp se almacena en data1
if(setup1==1)
{
    if(LUZ>=values[2])
    {
        alarma[0]=0x00;
        alarma[1]=0x0E;
        alarma[2]=0x3E;
        alarma[3]=0x6D;
        frecuencia(frequen[2]);
        //setup1=0; // Puede o no quitarse
    }
}
if(setup1==1)
{
    if(TEMP>=values[1])
    {
        alarma[0]=0x0F;
        alarma[1]=0x4F;
        alarma[2]=0x37;
        alarma[3]=0x67;
        frecuencia(frequen[1]);
        //setup1=0; // Puede o no quitarse
    }
}
```

```
if(muestras==1)
{
    if(negativo==1) // Si el valor negativo es real
    {
        GPIO_PORTH_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_PORTH_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_PORTH_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_PORTH_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_PORTH_DATA_R=0xFE; // Habilita dígito 4
GPIO_PORTM_DATA_R=dig muestra[3]; // Enciende dígito 4
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTH_DATA_R=0xFD; // Habilita dígito 3
GPIO_PORTM_DATA_R=dig muestra[2]; // Enciende dígito 3
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTH_DATA_R=0xFB; // Habilita dígito 2
GPIO_PORTM_DATA_R=dig muestra[1]; // Enciende dígito 2
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
GPIO_PORTM_DATA_R=dig muestra[0]; // Enciende dígito 1
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    }
}
if(GPIO_PORTP_DATA_R==0x01 || angulonegativo==1)
{
    if(contdig==0){ // Solo se acepta el signo si se ingresa al inicio
        digitos[contdig]=0x01; // Se iguala al signo - en 7seg
        contdig++; // Se incrementa el valor en 1
        negativo=0x01; // El valor negativo se hace 1
        for(i=0; i<=0x100000; i++){ // Ciclo de espera
        }
    }
}
if(GPIO_PORTP_DATA_R==0x02 || ingangulo==1)
{
    for(i=0; i<=0x100000; i++){ // Se espera un rato
        teclaA++;
    }

    /* 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
        {
            p=1;
            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
{
    p=1;
    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
}
if(p==1)
{
    if(teclaA==0x01)
    {
        graditosr[0]=graditos; // Guarda grados
        negativor[0]=negativo; // Guarda negativo
    }
}
```



```
muestra1[0]=digmuestra[0];
muestra1[1]=digmuestra[1];
muestra1[2]=digmuestra[2];
muestra1[3]=digmuestra[3];
negativo=0;
}else if(teclaA==0x02)
{
    graditosr[1]=graditos;//Guarda grados
    negativor[1]=negativo;//Guarda negativo
    muestra2[0]=digmuestra[0];
    muestra2[1]=digmuestra[1];
    muestra2[2]=digmuestra[2];
    muestra2[3]=digmuestra[3];
    negativo=0;
}else if(teclaA==0x03)
{
    graditosr[2]=graditos;//Guarda grados
    negativor[2]=negativo;//Guarda negativo
    muestra3[0]=digmuestra[0];
    muestra3[1]=digmuestra[1];
    muestra3[2]=digmuestra[2];
    muestra3[3]=digmuestra[3];
    negativo=0;
}else if(teclaA==0x04)
{
    graditosr[3]=graditos;//Guarda grados
    negativor[3]=negativo;//Guarda negativo
    muestra4[0]=digmuestra[0];
    muestra4[1]=digmuestra[1];
    muestra4[2]=digmuestra[2];
    muestra4[3]=digmuestra[3];
    negativo=0;
}else if(teclaA==0x05)
{
    graditosr[4]=graditos;//Guarda grados
    negativor[4]=negativo;//Guarda negativo
    muestra5[0]=digmuestra[0];
    muestra5[1]=digmuestra[1];
    muestra5[2]=digmuestra[2];
    muestra5[3]=digmuestra[3];
    negativo=0;
}else{}
}
}
if(GPIO_PORTP_DATA_R==0x04 || ejecutar==1)
{
    x=0;
    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=40;// Se convierte la velocidad a dígito real
        velocidadr=trunc(velocidad*1600000/2057); // Conversión de velocidad a RPM
        //habilita PORTN, PORTF, PORTL
        SYSCTL_RCGCGPIO_R |= 0x7EB0; // 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)
        TIMER3_ICR_R= 0x00000001 ; //LIMPIA POSIBLE BANDERA PENDIENTE DE TIMER3 (p.1002)
        TIMER3_IMR_R |= 0x00000001; //ACTIVA INTERRUPCION 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 = 0xF3; // Habilita PN0-PN1
GPIO_PORTN_DEN_R = 0xF3; // PN0-PN1 como salida

// habilita PF0 y PF4 como salida digital para monitoreo del programa
//
GPIO_PORTF_AHB_DIR_R = 0x1F; // PF0 y PF4
GPIO_PORTF_AHB_DEN_R = 0x1F; // 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
vueltas=teclaA;
teclaA=0;
s=0;
digmuestra[0]=muestra1[0];
digmuestra[1]=muestra1[1];
digmuestra[2]=muestra1[2];
digmuestra[3]=muestra1[3];
}
if(GPIO_PORTP_DATA_R==0x08)//Si se presiona PQ3
{
    for(i=0;i<=100;i++){
        if(muestrack==0)
        {
            muestrack=1; // Iguala a 1 para que se muestre
        }
        else
        {
            muestrack=0; // Iguala a 0 y ya no se muestra
        }
    }
    if(muestrack==1)
    {
        conversionD(); // Convierte para que se muestre
    }
}
}

void frecuencia(int frecuenciar) // Función frecuencia
{
    int f;
    f=16000000/frecuenciar; // Convierte frecuencia en ciclos de reloj
    GPIO_PORTK_DATA_R=0x10; // Monitorea valor de la tecla # ingresada
    for(i=0;i<=1000000;i++)
    {
        GPIO_PORTH_DATA_R=0xFE; // Habilita dígito 4
        GPIO_PORTM_DATA_R=alarma[3]; // Enciende dígito 4
        GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFD; // Habilita dígito 3
        GPIO_PORTM_DATA_R=alarma[2]; // Enciende dígito 3
        GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xFB; // Habilita dígito 2
        GPIO_PORTM_DATA_R=alarma[1]; // Enciende dígito 2
        GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
        GPIO_PORTH_DATA_R=0xF7; // Habilita dígito 1
        GPIO_PORTM_DATA_R=alarma[0]; // Enciende dígito 1
        GPIO_PORTM_DATA_R=0x00;
    }
    while(GPIO_PORTK_DATA_R!=0x18) // Si se teclaea #
    {
        for(i=0;i<=f;i++) // Ciclo encendido
```



```
{
    GPIO_PORTQ_DATA_R=0X01;//Enciende buzzer
}
for(i=0;i<=f;i++)//Ciclo apagado
{
    GPIO_PORTQ_DATA_R=0X00;//Apaga buzzer
}
}

void conversionD()//Función que convierte lectura de 12C a conversión en 7S para 14 dígitos
{
    int digitoRTC;
    int sevens;
    int sevenar;
    int j=0;
    int k=0;
    int n=0;

    for(j=0;j<=6;j++)
    {
        if(j==0)
        {
            segundos=convertidorsin(segundos);//Convierte dato
            digitoRTC=segundos;//Iguala variable
        }
        if(j==1)
        {
            minutos=convertidorsin(minutos);//Convierte dato
            digitoRTC=minutos;//Iguala variable
        }
        if(j==2)
        {
            horas=convertidorsin(horas);//Convierte dato
            digitoRTC=horas;//Iguala variable
        }
        if(j==3)
        {
            dia=convertidorsin(dia);//Convierte dato
            digitoRTC=dia;//Iguala variable
        }
        if(j==4)
        {
            digitoRTC=fecha;//Iguala variable, como es menor a 9 no se necesita convertir
        }
        if(j==5)
        {
            mes=convertidorsin(mes);//Convierte dato
            digitoRTC=mes;//Iguala variable
        }
        if(j==6)
        {
            anio=convertidorsin(anio);//Convierte dato
            digitoRTC=anio;//Iguala variable
        }

        dig1=digitoRTC&0x0F;//Obtiene digito1
        dig2=digitoRTC&0xF0;//Obtiene digito2
        digdec=dig2+dig1;//Conversión a decimal
        dec3=digdec/100;//Obtiene unidades
        dec2=(digdec-dec3*100)/10;//Obtiene decenas
        dec1=(digdec-dec3*100-dec2*10);//Obtiene centenas
        unidadesd[n]=dec1;//Almacena unidades
        decenasd[n]=dec2;//Almacena decenas
        n=n+1;//Incremento
    }
    for(j=0;j<=13;j++)//For para ir llenando los arreglos
    {
        for(k=0;k<=1;k++)
        {
            if(k==0)
```




```
{
    sevens=decenasd[j];//Almacena en el arreglo para ser mostrado
}
if(k==1)
{
    sevens=unidadesd[j];//Almacena en el arreglo para ser mostrado
}
if(sevens==1)
{
    sevenar=0x30;//Almacena en el arreglo para ser mostrado
}
if(sevens==2)
{
    sevenar=0x6D;//Almacena en el arreglo para ser mostrado
}
if(sevens==3)
{
    sevenar=0x79;//Almacena en el arreglo para ser mostrado
}
if(sevens==4)
{
    sevenar=0x33;//Almacena en el arreglo para ser mostrado
}
if(sevens==5)
{
    sevenar=0x5B;//Almacena en el arreglo para ser mostrado
}
if(sevens==6)
{
    sevenar=0x5F;//Almacena en el arreglo para ser mostrado
}
if(sevens==7)
{
    sevenar=0x70;//Almacena en el arreglo para ser mostrado
}
if(sevens==8)
{
    sevenar=0x7F;//Almacena en el arreglo para ser mostrado
}
if(sevens==9)
{
    sevenar=0x7B;//Almacena en el arreglo para ser mostrado
}
if(sevens==0)
{
    sevenar=0x7E;//Almacena en el arreglo para ser mostrado
}
if(k==0)
{
    segmentosd[j]=sevenar;//Almacena en el arreglo para ser mostrado
}
if(k==1)
{
    segmentosu[j]=sevenar;//Almacena en el arreglo para ser mostrado
}
}
}
for(i=0;i<=100;i++)
{
    GPIO_PORTH_DATA_R=0xFF;//Apaga puerto para que se muestre solo el que se requiere
    GPIO_PORTD_DATA_R=0xFF;//Apaga puerto para que se muestre solo el que se requiere
    GPIO_PORTF_DATA_R=0xFF;//Apaga puerto para que se muestre solo el que se requiere
    GPIO_PORTG_DATA_R=0xFF;//Apaga puerto para que se muestre solo el que se requiere
    GPIO_PORTN_DATA_R=0xFF;//Apaga puerto para que se muestre solo el que se requiere

    GPIO_PORTE_DATA_R=0xFE;//Habilita dígito 4
    GPIO_PORTM_DATA_R=segmentosu[0];//Enciende dígito 4
    GPIO_PORTM_DATA_R=0x00;//Apaga los 7 segmentos
    GPIO_PORTE_DATA_R=0xFD;//Habilita dígito 3
    GPIO_PORTM_DATA_R=segmentosd[0];//Enciende dígito 3
    GPIO_PORTM_DATA_R=0x00;//Apaga los 7 segmentos
```




```
GPIO_PORTE_DATA_R=0xFB; // Habilita dígito 2
GPIO_PORTM_DATA_R=segmentosu[1]; //Enciende dígito 2
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTE_DATA_R=0xF7; // Habilita dígito 1
GPIO_PORTM_DATA_R=segmentosd[1]; //Enciende dígito 1
GPIO_PORTM_DATA_R=0x00;
GPIO_PORTE_DATA_R=0xFF; // Habilita dígito 0

GPIO_PORTD_DATA_R=0xFE; // Habilita dígito 4
GPIO_PORTM_DATA_R=segmentosu[2]; //Enciende dígito 4
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTD_DATA_R=0xFD; // Habilita dígito 3
GPIO_PORTM_DATA_R=segmentosd[2]; //Enciende dígito 3
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTD_DATA_R=0xFB; // Habilita dígito 2
GPIO_PORTM_DATA_R=segmentosu[3]; //Enciende dígito 2
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTD_DATA_R=0xF7; // Habilita dígito 1
GPIO_PORTM_DATA_R=segmentosd[3]; //Enciende dígito 1
GPIO_PORTM_DATA_R=0x00;
GPIO_PORTD_DATA_R=0xFF; //Apaga puerto para que se muestre solo el que se requiere

GPIO_PORTF_DATA_R=0xFD; // Habilita dígito 3
GPIO_PORTM_DATA_R=segmentosu[4]; //Enciende dígito 3
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTF_DATA_R=0xFB; // Habilita dígito 2
GPIO_PORTM_DATA_R=segmentosd[4]; //Enciende dígito 2
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTF_DATA_R=0xF7; // Habilita dígito 1
GPIO_PORTM_DATA_R=segmentosu[5]; //Enciende dígito 1
GPIO_PORTM_DATA_R=0x00;
GPIO_PORTF_DATA_R=0xFF; //Apaga puerto para que se muestre solo el que se requiere

GPIO_PORTG_DATA_R=0xFE; // Habilita dígito 4
GPIO_PORTM_DATA_R=segmentosd[5]; //Enciende dígito 4
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTG_DATA_R=0xFD; // Habilita dígito 3
GPIO_PORTM_DATA_R=segmentosu[6]; //Enciende dígito 3
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTG_DATA_R=0xFF;

GPIO_PORTN_DATA_R=0xEF; // Habilita dígito 4
GPIO_PORTM_DATA_R=segmentosd[6]; //Enciende dígito 4
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTG_DATA_R=0xFF; //Apaga puerto para que se muestre solo el que se requiere
```

```
}
}

int convertidorsin(int digitc) //Función que convierte el ciclo de 0-90 a 0-60
{
    if(digitc<=9)
    {
        digitc=digitc; //Conversión de datos
    }
    else if(digitc<=25)
    {
        digitc=digitc-6; //Conversión de datos
    }
    else if(digitc<=41)
    {
        digitc=digitc-12; //Conversión de datos
    }
    else if(digitc<=57)
    {
        digitc=digitc-18; //Conversión de datos
    }
    else if(digitc<=73)
    {
        digitc=digitc-24; //Conversión de datos
    }
}
```

```

else if(digitc<=90)
{
    digitc=digitc-30;//Conversión de datos
}
return digitc;
}

```

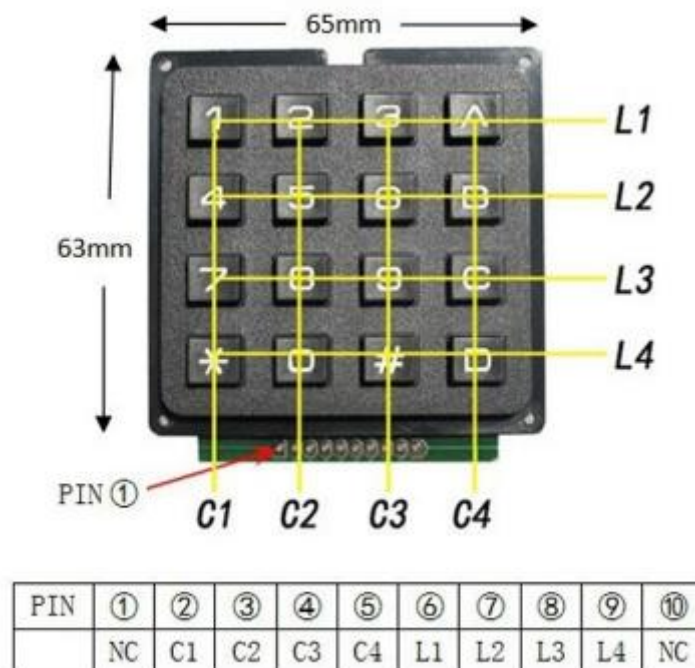
VI. Construcción

Sistema de monitoreo

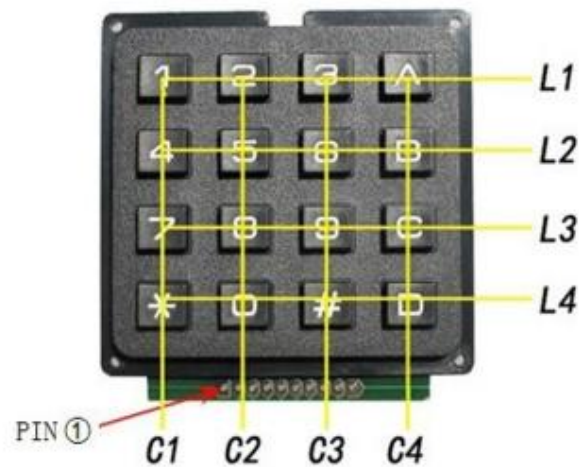
Para este proyecto se utilizaron 4 puertos del microcontrolador los cuales fueron el puerto M, K, Q y E. Se utilizó un teclado matricial de 4x4 con 4 entradas y 4 salidas, un display 7 segmentos con 6 salidas para leds y 4 salidas para habilitación de dígitos y por último tres sensores para medir temperatura, presencia y luz y un buzzer pasivo de 2 a 4.5kHz.

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.

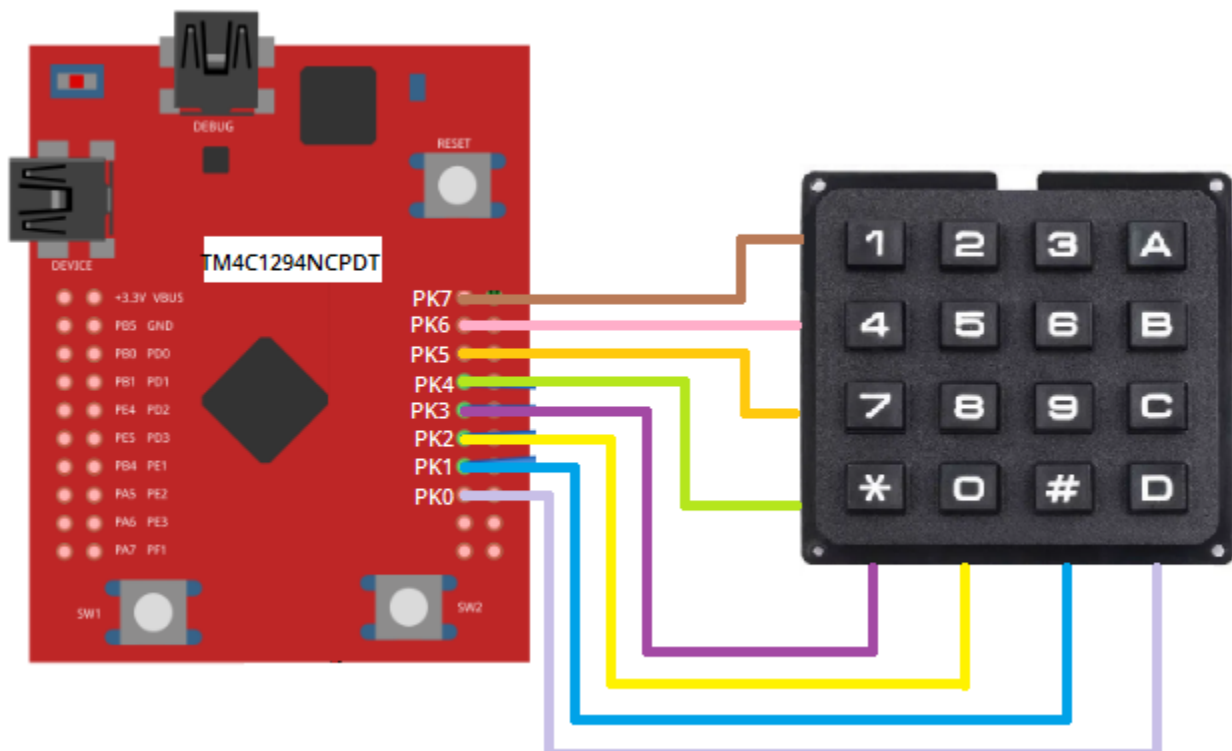


Ya que se conocen como están configurados los pines del teclado matricial procedemos a poner el puerto de cada pin como corresponde, sabiendo que se conectaran 8 de ellos.



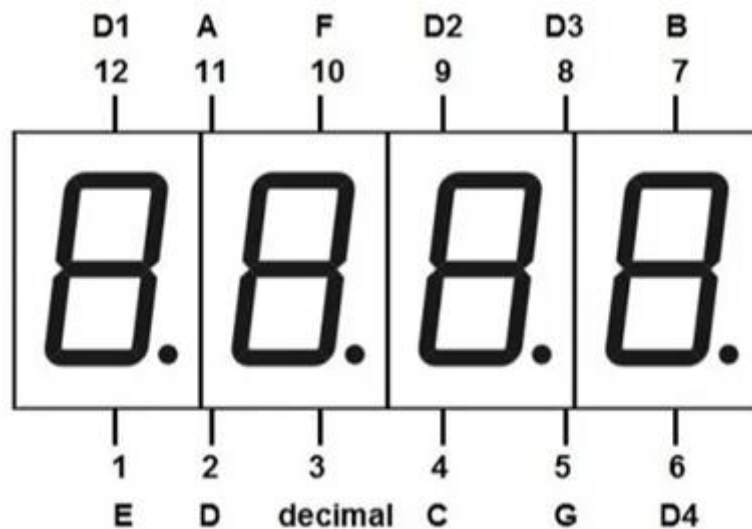
PIN	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
	NC	PK3	PK2	PK1	PK0	PK4	PK7	PK6	PK5	NC

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.

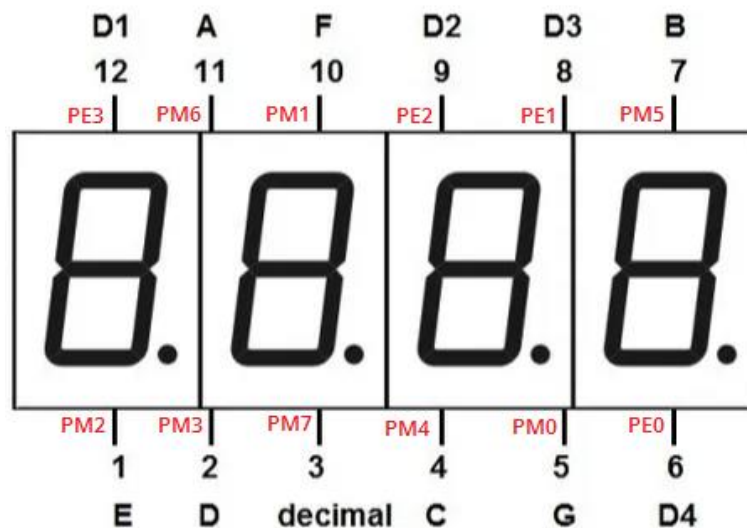


Conexión al display 7 segmentos

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:

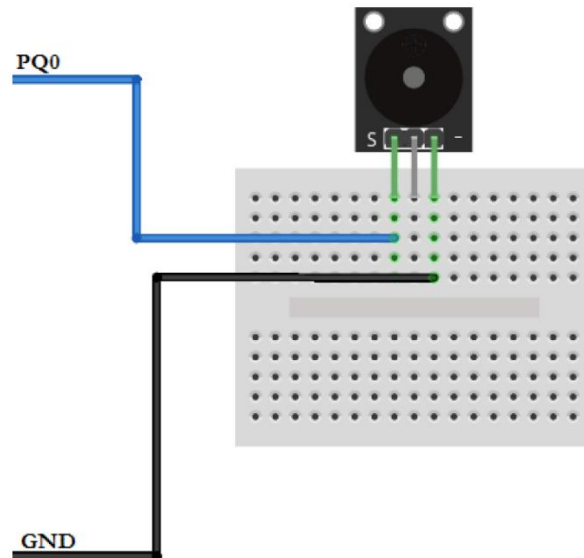


Para realizar las conexiones utilicé el puerto E para la habilitación de los dígitos y el puerto M para los segmentos por lo que la conexión realizada al microcontrolador fue la siguiente.



Conexiones de sensores y buzzer pasivo

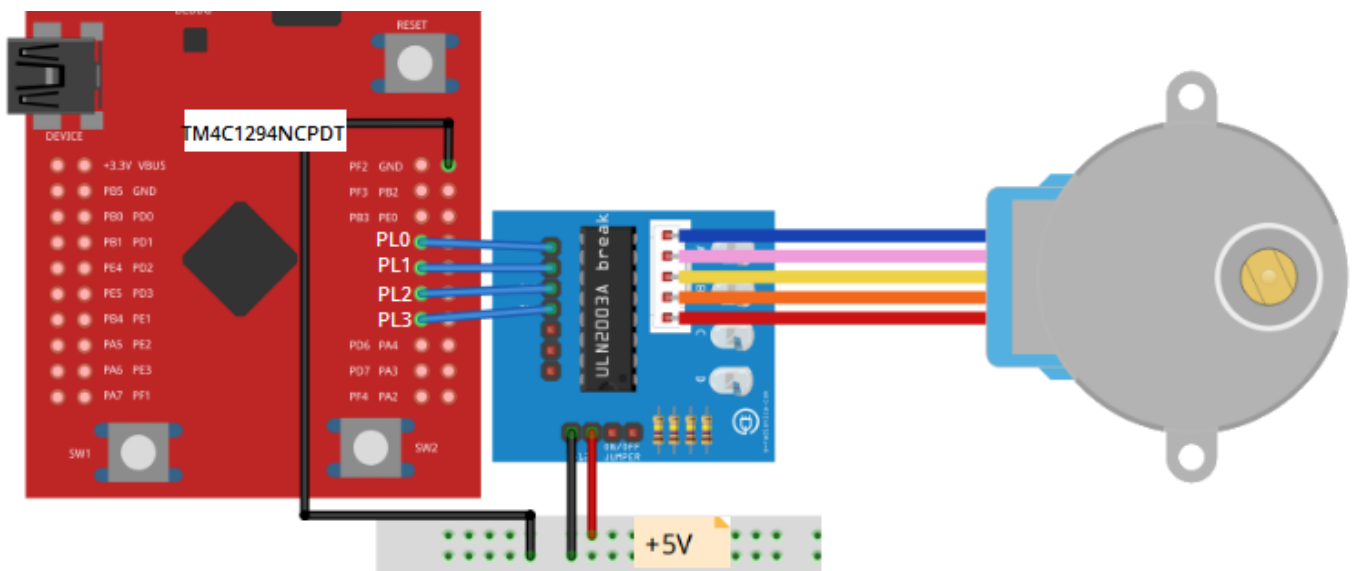
Para el Buzzer pasivo se utilizó el puerto PQ0, PQ1 sensor de luz y PQ2 sensor de temperatura, para la entrada del sensor de presencia se utilizó el puerto L.



Motor a pasos

Conexión al motor

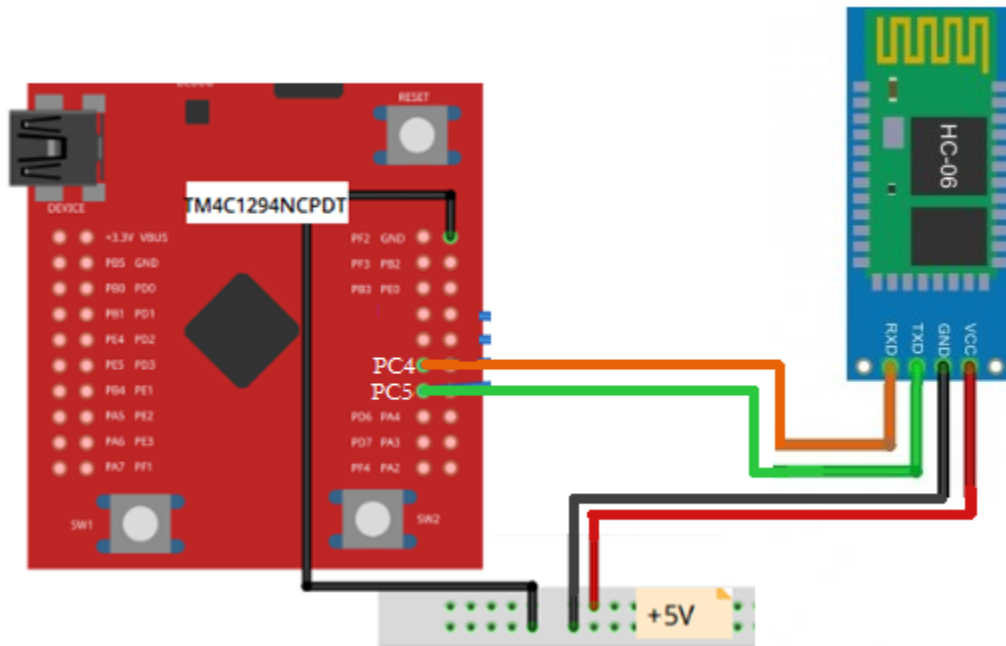
Para conectar el motor se habilitó el puerto L con 4 salidas las cuales son de PL0 a PL3, se realizaron las siguientes conexiones:



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.

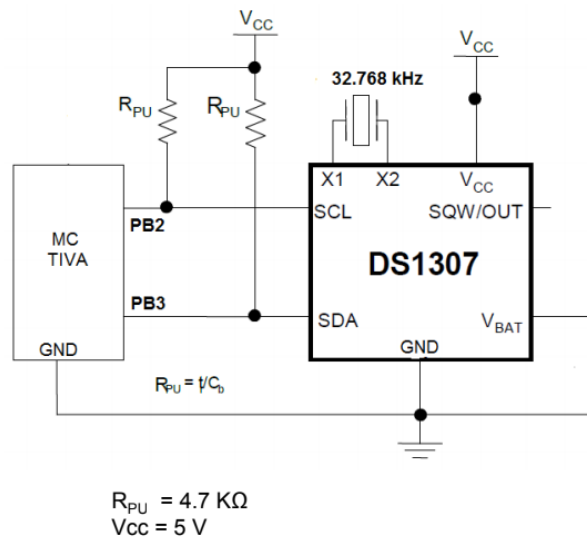
Módulo Bluetooth

Para el módulo bluetooth ocupamos el puerto C, las conexiones se realizaron así:



Reloj en Tiempo Real (RTC)

Para el reloj en tiempo real y su muestra en display 7 segmentos se utilizaron los puerdos D,E,F,G,N y B, además del puerto M para los 7 segmentos de todos los displays del proyecto, pero eso se replica con la parte de construcción del sistema de monitoreo, solo hay que conectar todos los segmentos en uno mismo y lo que variamos es el dígito que se activa. Las conexiones realizadas son:



Los dígitos de habilitación del display 7 segmentos de cada parte son:

Variable	Puerto para decenas	Puerto para unidades
Segundos	PE0	PE1

Minutos	PE2	PE3
Horas	PD0	PD1
Año	PG1	PN4
Mes	PF3	PG0
Día	PG1	PN4
Fecha de la semana	PD2	PD3

Puertos para ingreso de datos RTC y Motor a pasos

Ya para finalizar, la parte final fue ponerle botones extras al proyecto para que también tenga un control por push-buttons, el puerto utilizado para esto es el P, y sus funciones son las siguientes:

PP0 - Signo negativo del ángulo

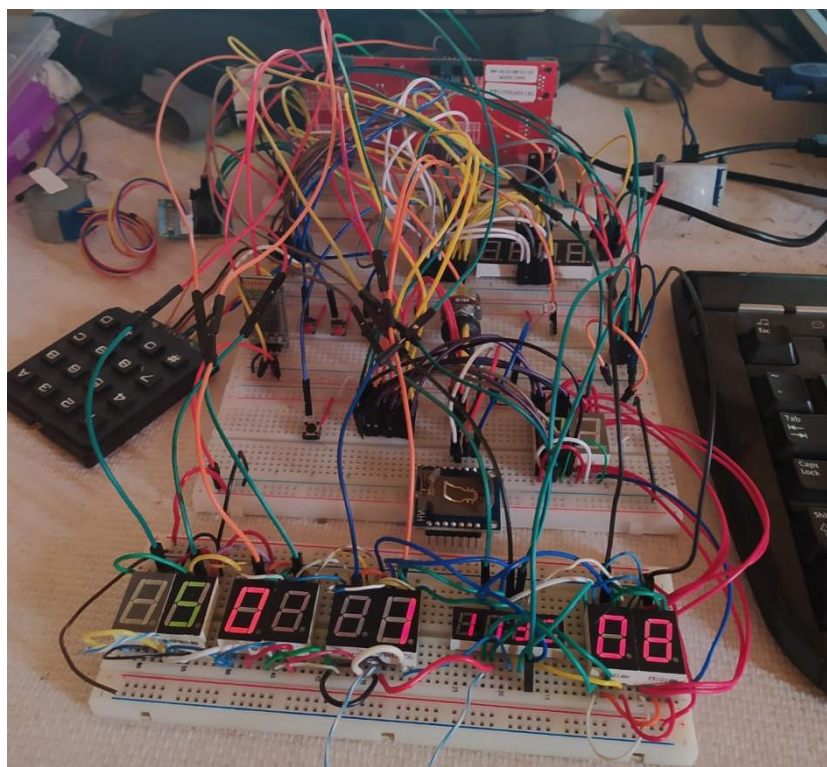
PP1 - Ingreso del ángulo



PP2 - Giro del motor

PP3 - Muestra del reloj

VII. Resultados y – Conclusiones

El ensamblado final del proyecto se muestra a continuación:



	UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Microprocesadores y Microcontroladores	Semestre: 2021-1	
	Prof. Dr. Saúl De La Rosa Nieves	Grupo 3 Página 52 de 52	

La conclusión de este proyecto es que se trata del culmen de todo lo aprendido en el semestre, pese a que la mayoría de lo que se programó corresponde a la parte de periféricos, nada de esto hubiera sido posible sin la aplicación de los conocimientos adquiridos al inicio del semestre y también sin lo aprendido en la materia de diseño digital. La parte más compleja que considero fue el hecho de juntar todo en un solo código, primero realicé el funcionamiento de todo individualmente y todo lo fui juntando en un solo código fuente.

El proyecto considero yo que es exitoso y que también posee algunas cosas extras a las solicitadas, sin embargo, es claro que puede ser optimizado ya que el resultado final supera las 1000 líneas de código y tiene algunos problemas cuando pruebas cada cosa sin seguir el proceso esperado del usuario, en pocas palabras, le falta ser más robusto.

Otro problema que encuentro es que el hecho de realizar un multiplexaje y desarrollo tan largo hay que pulsar lentamente las teclas o no se tomarán en cuenta los datos ingresados, sobre todo con la muestra de la fecha. Y por último, el error que también encuentro es que la muestra de datos para todos los sensores y también la parte de la alarma consumen todo el recurso del procesador, lo que hace que se detenga el reloj unos instantes. Para cambiar esto hay que introducir interrupciones a esos procesos, no se implementó por falta de tiempo pero es lo que se debe seguir para que el proyecto sea excelente.