
	<b>UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO</b> <b>FACULTAD DE INGENIERÍA</b> <b>Microprocesadores y Microcontroladores</b>	Semestre: 2021-1	
	<b>Prof. Dr. Saúl De La Rosa Nieves</b>	Página 1 de 27	

### TAREA-EXAMEN 3

<b>Título:</b>	<b>Sistema de monitoreo</b>
<b>Fecha:</b>	17-12-2020
<b>Preparado por:</b>	Fiel Muñoz Teresa Elpidia
<b>Evaluación:</b>	

## 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.

### 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).

## II. Requerimientos del proyecto

### Requerimientos de hardware

1. Como unidad de procesamiento 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.

6. La visualización de los niveles de monitoreo y el estado del sistema se presentarán en un “Display de 7 segmentos” de 4 dígitos como el que se muestra en la 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.

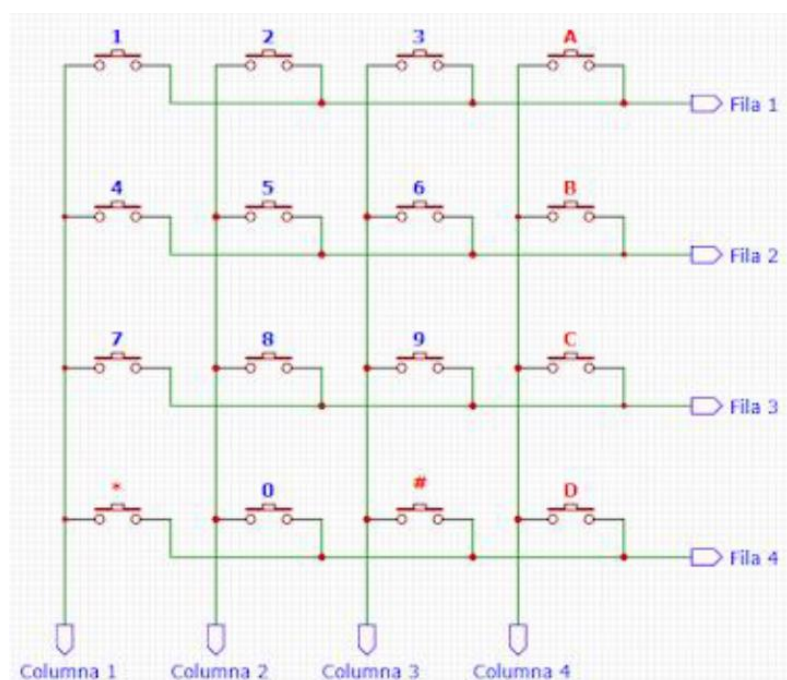
### Requerimientos de funcionamiento



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 mostrara 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.

## III. Marco Teórico

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.

### Teclado Matricial



	<b>UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO</b> <b>FACULTAD DE INGENIERÍA</b> <b>Microprocesadores y Microcontroladores</b>	Semestre: 2021-1	
	<b>Prof. Dr. Saúl De La Rosa Nieves</b>	Página 3 de 27	

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.

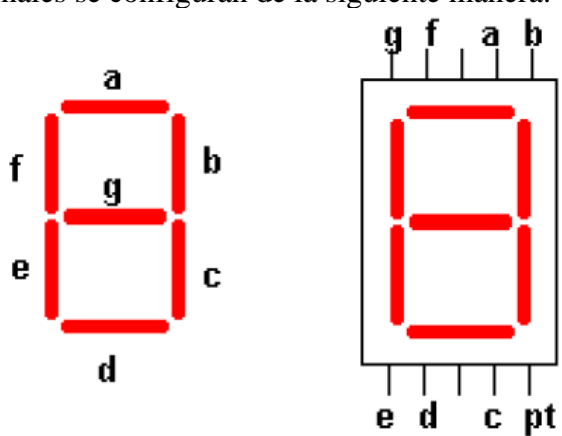
Siguiendo el diagrama se puede ver con esta coincidencia a que tecla corresponden los valores presionados. Una vez conseguido este valor se puede decodificar a 7 segmentos o a hexadecimal según sea el caso.

### 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:



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

## LM35

El LM35 es un sensor de temperatura de buenas prestaciones a un bajo precio. Posee un rango de trabajo desde  $-55^{\circ}\text{C}$  hasta  $150^{\circ}\text{C}$ . 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  $0.5^{\circ}\text{C}$ .

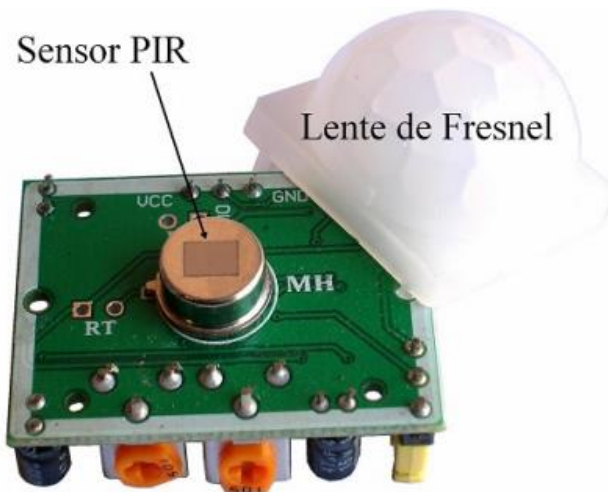
Es un sensor muy popular por su fácil uso y variadas aplicaciones. No necesita de ningún circuito adicional para ser usado. Se alimenta directamente con una fuente de 5V y entrega una salida analógica entre 0V a 1.5V. Este voltaje analógico puede ser leído por el ADC de un microcontrolador como PIC o 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  $110^{\circ}$  y, adicionalmente, concentra la energía en la superficie de detección del sensor PIR, permitiendo una mayor sensibilidad del dispositivo.

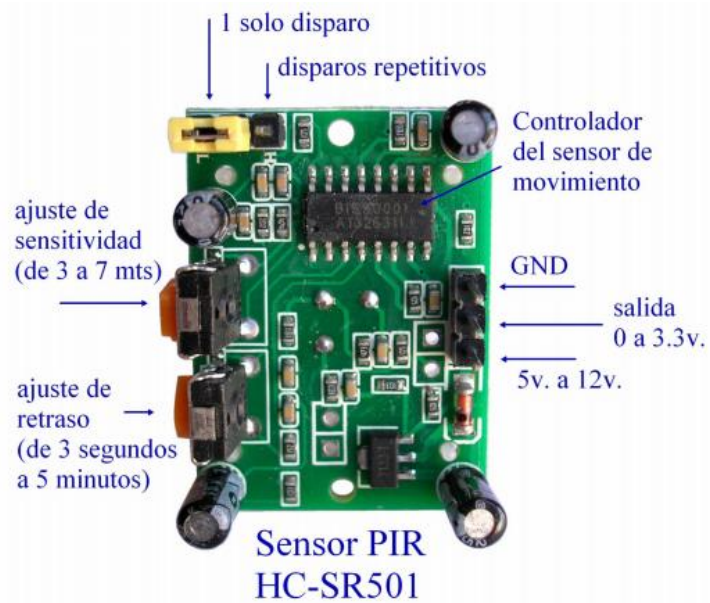


El sensor PIR infrarrojo: En los sensores de movimiento, el sensor PIR consta en realidad de 2 elementos detectores separados, siendo la señal diferencial entre ambos la que permite activar la alarma de movimiento. En el caso del HC-SR501, la señal generada por el sensor ingresa al circuito integrado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

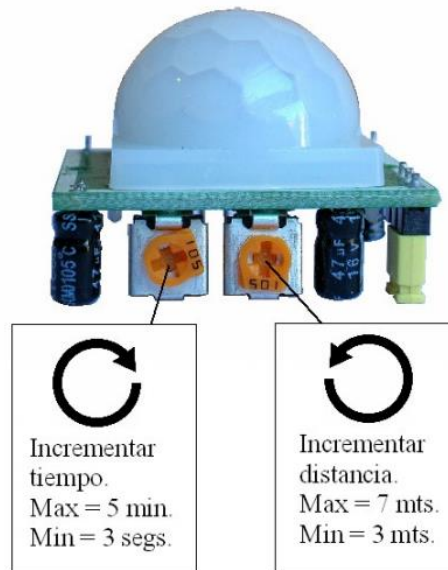
Las funciones y ajustes complementarios del sensor de movimiento son:

- 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.

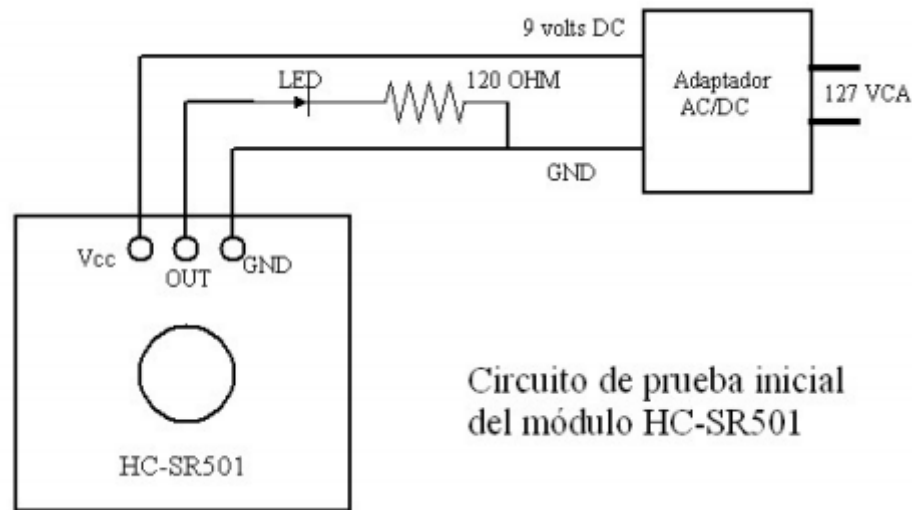
El módulo PIR modelo HC-SR501 es de bajo costo, pequeño, e incorpora la tecnología más reciente en sensores de movimiento. El sensor utiliza 2 potenciómetros y un jumper que permiten modificar sus parámetros y adaptarlo a las necesidades de la aplicación: sensibilidad de detección, tiempo de activación,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



*Ajustes y configuración del sensor:*

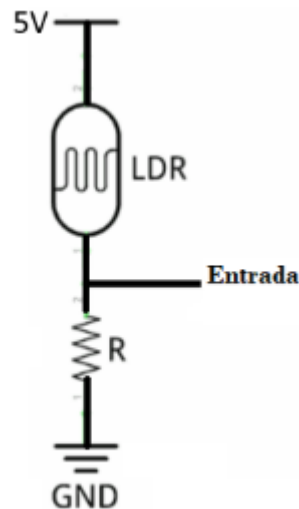


*Conexiones al ADC*



### 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

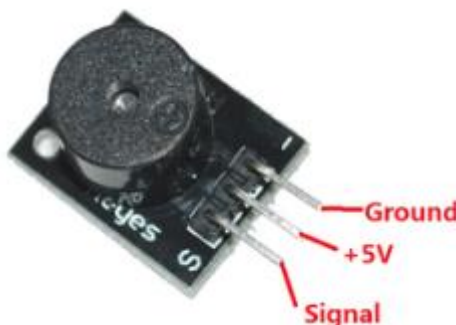
Módulo zumbador piezoeléctrico pasivo Arduino Keyes KY-006, puede producir una gama de tonos de sonido dependiendo de la frecuencia de entrada.

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.



	<b>UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO</b> <b>FACULTAD DE INGENIERÍA</b> <b>Microprocesadores y Microcontroladores</b>	Semestre: 2021-1	
	<b>Prof. Dr. Saúl De La Rosa Nieves</b>	<b>Grupo 3</b>	



#### 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.

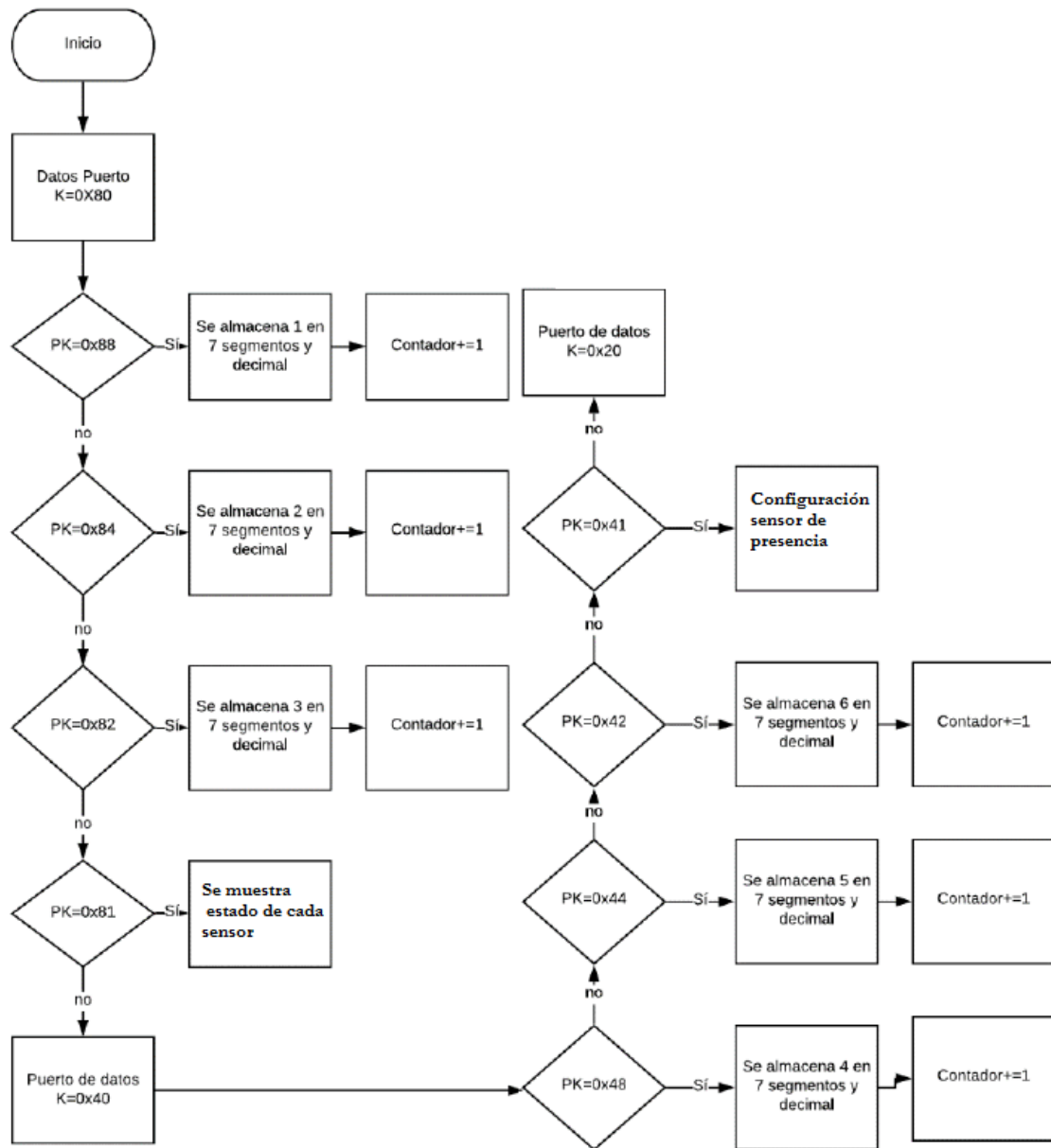
Tecla #.- Es la tecla utilizada para que el buzzer deje de sonar y se reestablezca el sistema.

Tecla \*.- Es la tecla utilizada para que se ingresen los datos en la memoria del programa, tanto para frecuencias como para valores.

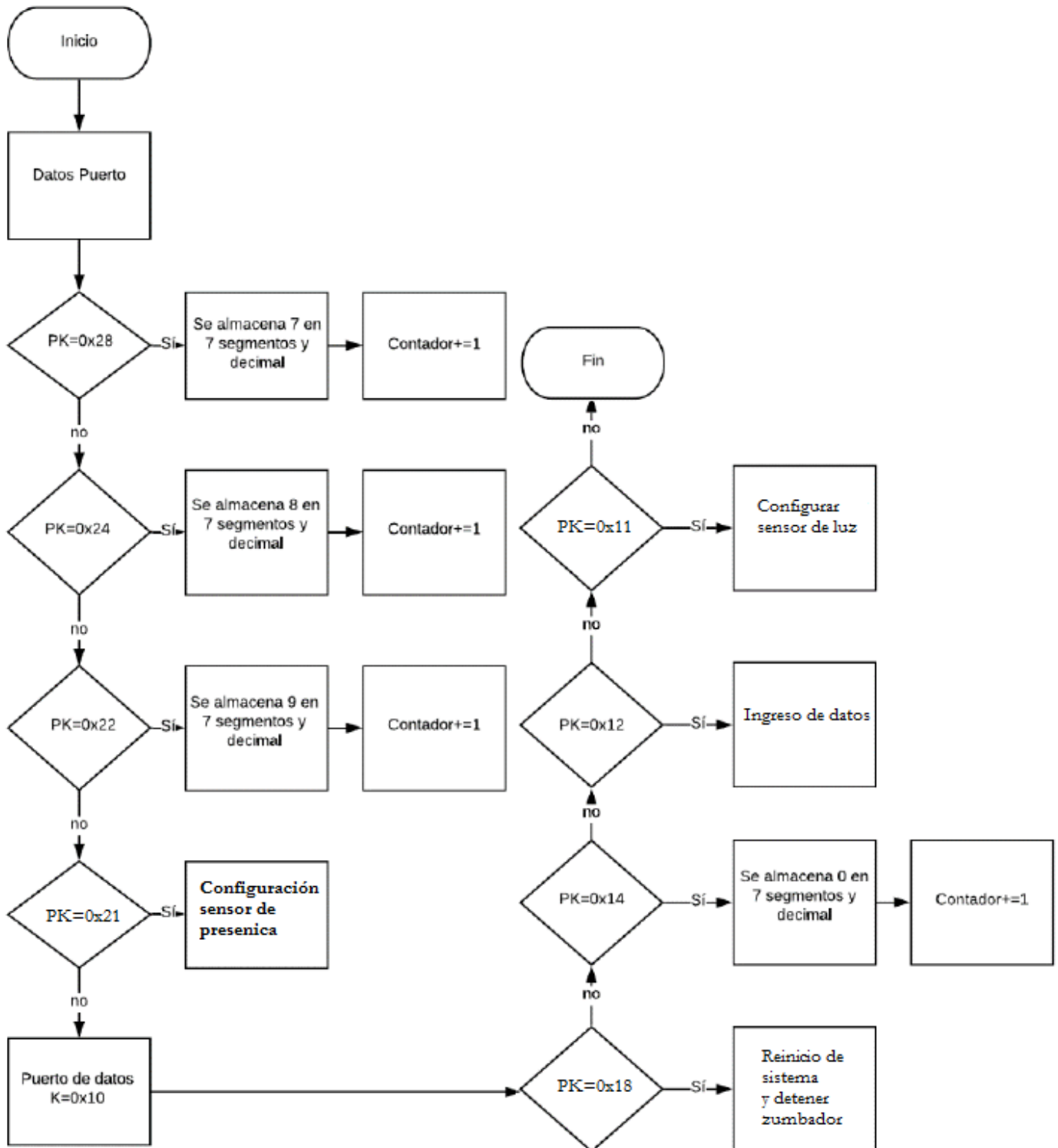
#### Diagrama de Flujo

*Decodificación teclado matricial primeras dos columnas:*

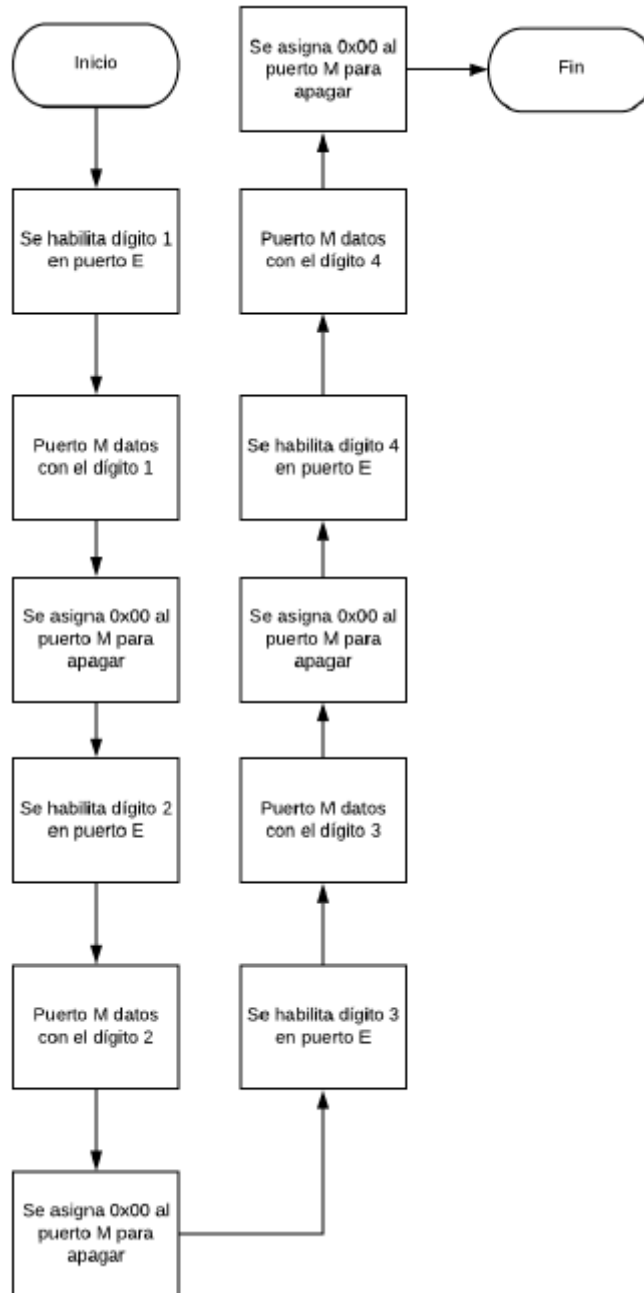




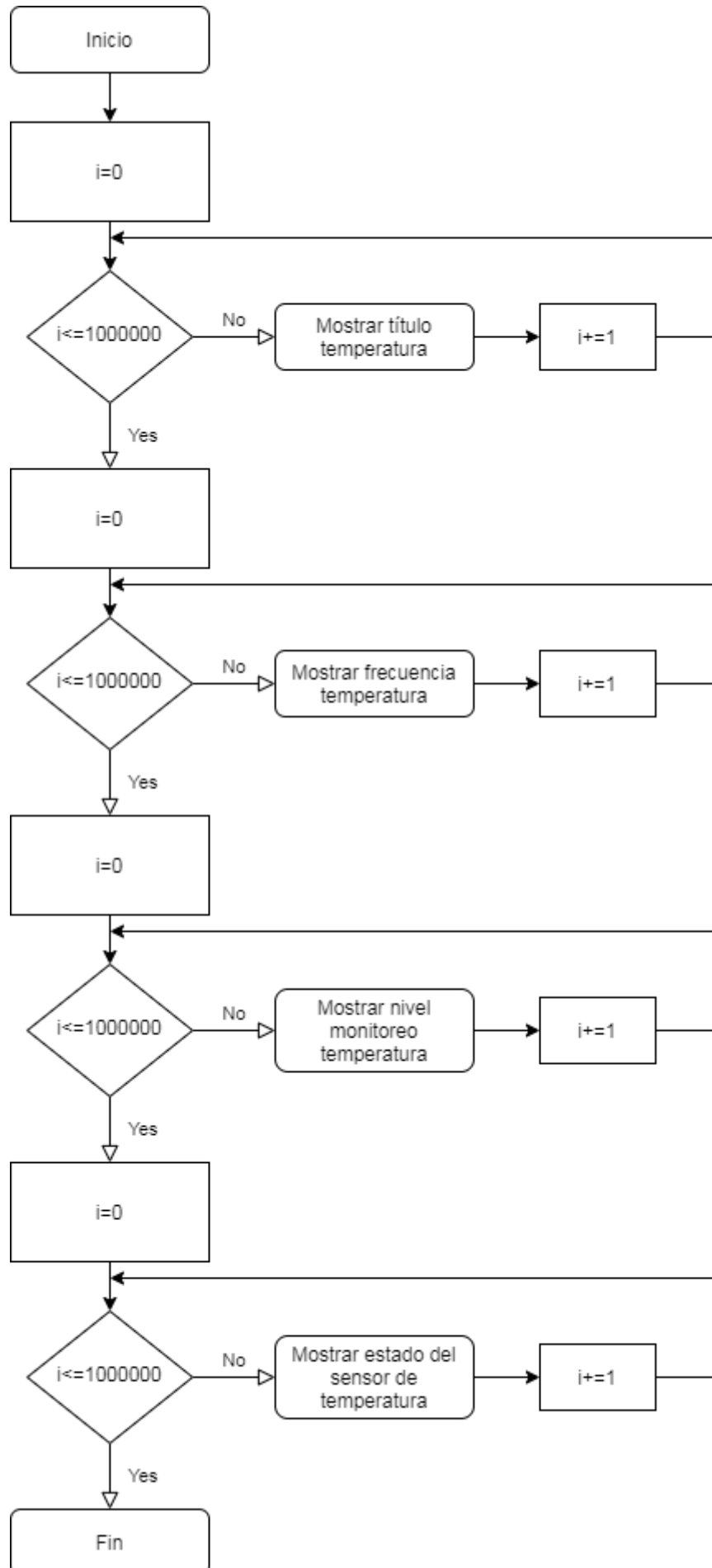
*Decodificación teclado matricial últimas dos columnas:*



*Despliegue del display 7 segmentos*

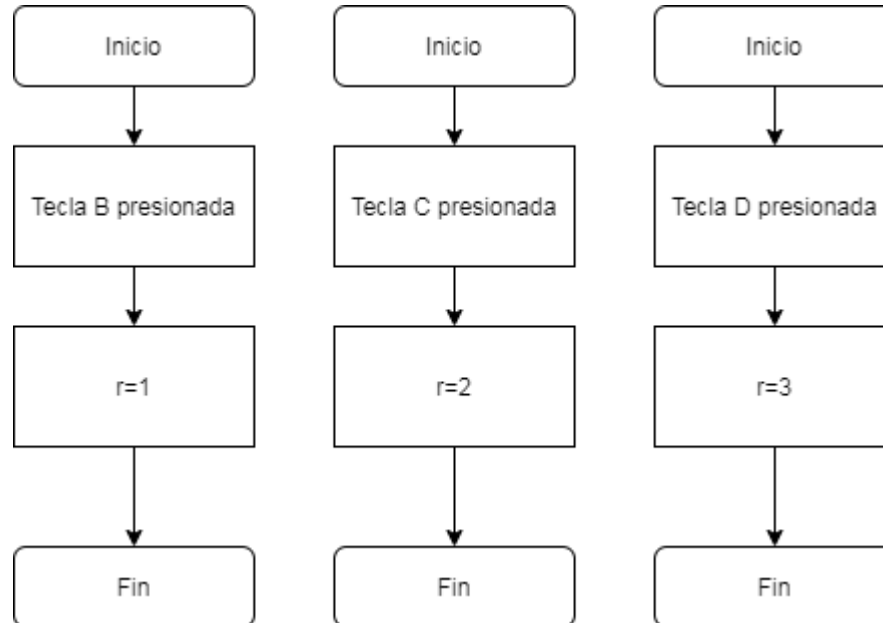


*Muestra de datos (Tecla A)*

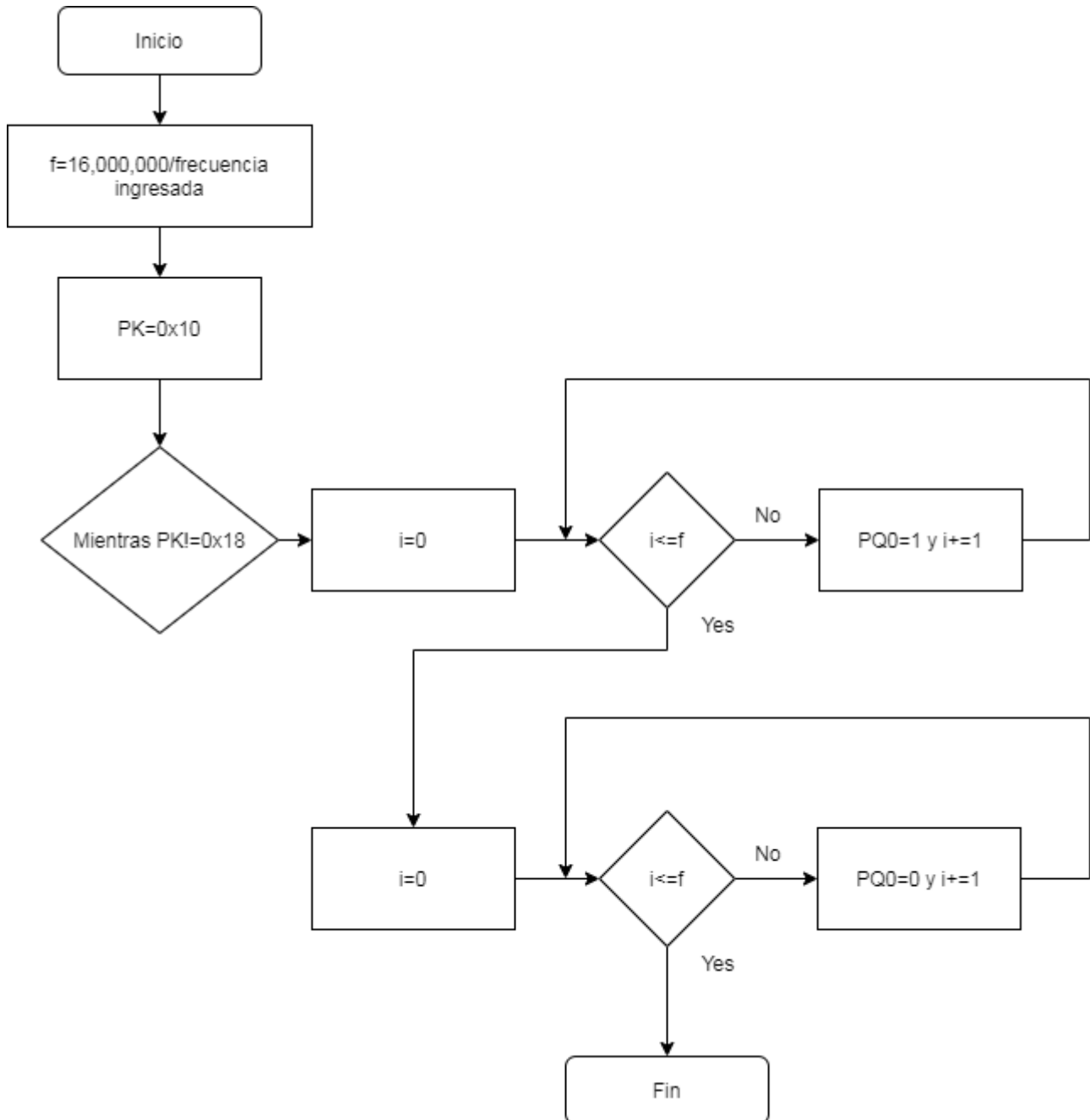




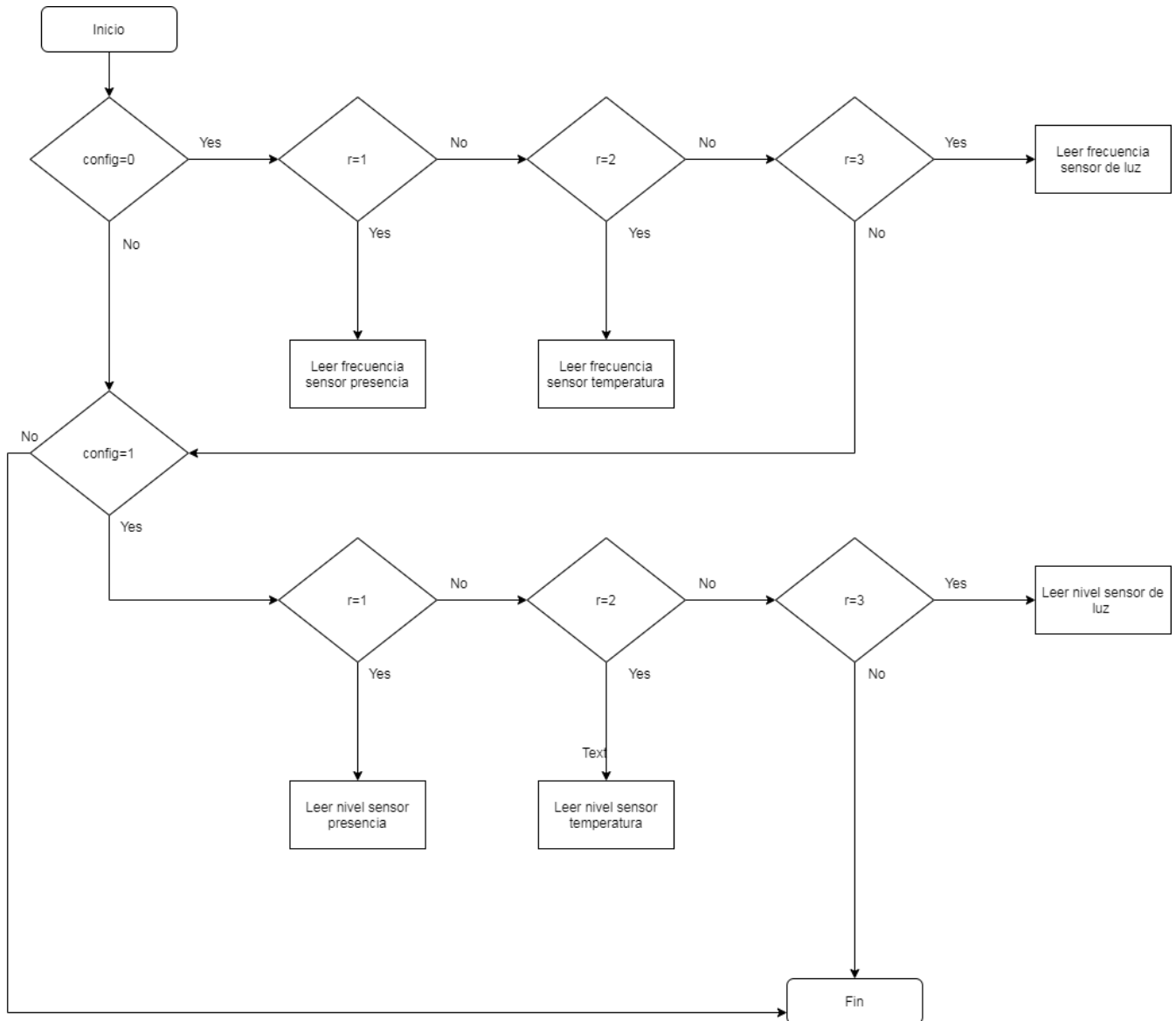
***Ingreso de configuración (Teclas B,C,D)***



***Tecla #***



Tecla \*



## Código

### Código para teclado matricial y configuraciones:

```
/* TAREA-EXAMEN NO.3
 * Realizado por: Fiel Muñoz Teresa Elpidia
 */
```

```
//-----LIBRERIAS DEL PROGRAMA-----
```

```
#include <stdbool.h>
#include <stdint.h>
#include <math.h>
volatile uint32_t Count= 0;
volatile uint32_t Termino= 0;
```

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

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

```
#define GPIO_PORTK_DATA_R (*(volatile unsigned long *)0x400613FC) // Registro de Datos Puerto K
#define GPIO_PORTK_DIR_R (*(volatile unsigned long *)0x40061400) // Registro de Dirección Puerto K
#define GPIO_PORTK_DEN_R (*(volatile unsigned long *)0x4006151C) // Registro de Habilitación Puerto K
```







```
r=0; //Inicio de configuración
config=0; //Inicio de configuración

while(1)
{
    valor=digreal[3]+digreal[2]*10+digreal[1]*100+digreal[0]*1000; //Calcula dígito total con dígitos ingresados
    /*Fila 1*/

    GPIO_PORTK_DATA_R=0x80;          // Se multiplexa la primera fila del teclado

    if(GPIO_PORTK_DATA_R==0x88)      // Presiona tecla 1
    {
        if(contdig==4)              // Si ya se ingresaron cuatro dígitos del display se reinicia
        {
            contdig=0x00;           // Cuenta reiniciada
        }
        if(contdig<=3){              // Si la cuenta es menor a 3 se ingresa el dato
            digitos[contdig]=0x30;    // Valor del display 7 segmentos
            digreal[contdig]=1;       // El dígito real es 1
            contdig+=1; // Se incrementa la cuenta
            for(i=0;i<=0x100000;i++){ // Valor de espera
            }
        }
    }
    else if(GPIO_PORTK_DATA_R==0x84) // Valor 2 leído
    {
        if(contdig==4) // Se reinician los displays
        {
            contdig=0x00; //El contador se reinicia
        }
        if(contdig<=3){ // Si la cuenta es menor a 3 se ingresa el dato
            digitos[contdig]=0x60; // Valor del display 7 segmentos
            digreal[contdig]=2; // El dígito real es 2
            contdig+=1; // Se incrementa la cuenta
            for(i=0;i<=0x100000;i++){ // Valor de espera
            }
        }
    }
    else if(GPIO_PORTK_DATA_R==0x82) // Valor 3 del teclado
    {
        if(contdig==4) // Si la cuenta es 4 del display
        {
            contdig=0x00; // Se reinicia el valor de cuenta
        }
        if(contdig<=3){ // Si la cuenta es tres o menor
            digitos[contdig]=0x79; // Valor del display 7 segmentos
            digreal[contdig]=3; // El dígito real es 3
            contdig+=1; // Se incrementa la cuenta
            for(i=0;i<=0x100000;i++){ // Valor de espera
            }
        }
    }
    else if(GPIO_PORTK_DATA_R==0x81) // Tecla A presionada
    {
        /*Presencia*/

        for(i=0;i<=1000000;i++)
        {
            GPIO_PORTC_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_PORTC_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_PORTC_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_PORTC_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_PORTM_DATA_R=0xFE; // Habilita dígito 4
    GPIO_PORTM_DATA_R=fmuestrap[3]; //Enciende dígito 4
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xFD; // Habilita dígito 3
    GPIO_PORTM_DATA_R=fmuestrap[2]; //Enciende dígito 3
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xFB; // Habilita dígito 2
    GPIO_PORTM_DATA_R=fmuestrap[1]; //Enciende dígito 2
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xF7; // Habilita dígito 1
    GPIO_PORTM_DATA_R=fmuestrap[0]; //Enciende dígito 1
    GPIO_PORTM_DATA_R=0x00;
}
for(i=0; i<=1000000; i++)
{
    GPIO_PORTM_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_PORTM_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_PORTM_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_PORTM_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_PORTM_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_PORTM_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_PORTM_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_PORTM_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_PORTM_DATA_R=0xFE; // Habilita dígito 4
    GPIO_PORTM_DATA_R=fmuestrat[3]; //Enciende dígito 4
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xFD; // Habilita dígito 3
    GPIO_PORTM_DATA_R=fmuestrat[2]; //Enciende dígito 3
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xFB; // Habilita dígito 2
    GPIO_PORTM_DATA_R=fmuestrat[1]; //Enciende dígito 2
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xF7; // Habilita dígito 1
    GPIO_PORTM_DATA_R=fmuestrat[0]; //Enciende dígito 1
    GPIO_PORTM_DATA_R=0x00;
}
for(i=0; i<=1000000; i++)
{
    GPIO_PORTM_DATA_R=0xFE; // Habilita dígito 4
    GPIO_PORTM_DATA_R=vmuestrat[3]; //Enciende dígito 4
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xFD; // Habilita dígito 3
    GPIO_PORTM_DATA_R=vmuestrat[2]; //Enciende dígito 3
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xFB; // Habilita dígito 2
```



```
GPIO_PORTM_DATA_R=vmuestrat[1];//Enciende dígito 2
GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
GPIO_PORTM_DATA_R=0xF7; // Habilita dígito 1
GPIO_PORTM_DATA_R=vmuestrat[0];//Enciende dígito 1
GPIO_PORTM_DATA_R=0x00;
}
for(i=0;i<=1000000;i++)
{
    GPIO_PORTM_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_PORTM_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_PORTM_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_PORTM_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_PORTM_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_PORTM_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_PORTM_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_PORTM_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_PORTM_DATA_R=0xFE; // Habilita dígito 4
    GPIO_PORTM_DATA_R=fmuestrat[3]; //Enciende dígito 4
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xFD; // Habilita dígito 3
    GPIO_PORTM_DATA_R=fmuestrat[2]; //Enciende dígito 3
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xFB; // Habilita dígito 2
    GPIO_PORTM_DATA_R=fmuestrat[1]; //Enciende dígito 2
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xF7; // Habilita dígito 1
    GPIO_PORTM_DATA_R=fmuestrat[0]; //Enciende dígito 1
    GPIO_PORTM_DATA_R=0x00;
}
for(i=0;i<=1000000;i++)
{
    GPIO_PORTM_DATA_R=0xFE; // Habilita dígito 4
    GPIO_PORTM_DATA_R=vmuestrat[3]; //Enciende dígito 4
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xFD; // Habilita dígito 3
    GPIO_PORTM_DATA_R=vmuestrat[2]; //Enciende dígito 3
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xFB; // Habilita dígito 2
    GPIO_PORTM_DATA_R=vmuestrat[1]; //Enciende dígito 2
    GPIO_PORTM_DATA_R=0x00; // Apaga los 7 segmentos
    GPIO_PORTM_DATA_R=0xF7; // Habilita dígito 1
    GPIO_PORTM_DATA_R=vmuestrat[0]; //Enciende dígito 1
    GPIO_PORTM_DATA_R=0x00;
}
for(i=0;i<=1000000;i++)
{

```



```
GPIO_PORTC_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_PORTC_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_PORTC_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_PORTC_DATA_R=0xF7; // Habilita dígito 1
GPIO_PORTM_DATA_R=0x00; // Enciende dígito 1
GPIO_PORTM_DATA_R=0x00;
}
frec=valor*10;
frecuencia(frec);
}

/* Fila 2 */

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

if(GPIO_PORTK_DATA_R==0x48) // Valor 4 presionado
{
    if(contdig==4) // Si ya se registraron todos los dígitos el conteo reinicia
    {
        contdig=0x00; // Se reinicia
    }
    if(contdig<=3){ // Si aún se pueden ingresar dígitos
        digitos[contdig]=0x33; // Valor 4 en 7 seg
        digreal[contdig]=4; // Valor 4 en decimal
        contdig+=1; // Se incrementa el conteo en 1
        for(i=0;i<=0x100000;i++){ // Ciclo de espera
        }
    }
}
else if(GPIO_PORTK_DATA_R==0x44) // Tecla 5 presionada
{
    if(contdig==4) // Si ya se ingresaron todos los dígitos
    {
        contdig=0x00; // El valor se reinicia
    }
    if(contdig<=3){ // Si aún se pueden ingresar datos
        digitos[contdig]=0x5B; // Valor 7 segmentos de 5
        digreal[contdig]=5; // Valor real 5
        contdig+=1; // Se incrementa el conteo
        for(i=0;i<=0x100000;i++){ // Ciclo de espera
        }
    }
}
else if(GPIO_PORTK_DATA_R==0x42) // Valor 6 presionado
{
    if(contdig==4) // Si ya se ingresaron todos los dígitos
    {
        contdig=0x00; // Se reinicia el valor
    }
    if(contdig<=3){ // Si aún se pueden ingresar dígitos
        digitos[contdig]=0x5F; // Se ingresa el valor 5 7segm
        digreal[contdig]=6; // Se ingresa el valor 6 decimal
        contdig+=1; // Se incrementa en 1 el contador
        for(i=0;i<=0x100000;i++){ // Ciclo de espera
        }
    }
}
else if(GPIO_PORTK_DATA_R==0x41) // Se configuró tecla B como # por fallas en mi teclado
{
    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 reincia 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) // Si se presiona tecla C
{
    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 reincia 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)
```







```
{
    frequen[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
}
else if(r==2)
{
    frequen[1]=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
}
else if(r==3)
{
    frequen[2]=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
}
}
else if(config==1)
{
    if(r==1)
    {
        values[0]=valor; //Valor igual al valor ingresado
        vmuestrap[0]=digitos[0]; //Guarda valor en frecuencia de sensor
        vmuestrap[1]=digitos[1]; //Guarda valor en frecuencia de sensor
        vmuestrap[2]=digitos[2]; //Guarda valor en frecuencia de sensor
        vmuestrap[3]=digitos[3]; //Guarda valor en frecuencia de sensor
    }
    else if(r==2)
    {
        values[1]=valor; //Valor igual al valor ingresado
        vmuestrap[0]=digitos[0]; //Guarda valor en frecuencia de sensor
        vmuestrap[1]=digitos[1]; //Guarda valor en frecuencia de sensor
        vmuestrap[2]=digitos[2]; //Guarda valor en frecuencia de sensor
        vmuestrap[3]=digitos[3]; //Guarda valor en frecuencia de sensor
    }
    else if(r==3)
    {
        values[2]=valor; //Valor igual al valor ingresado
        vmuestrap[0]=digitos[0]; //Guarda valor en frecuencia de sensor
        vmuestrap[1]=digitos[1]; //Guarda valor en frecuencia de sensor
        vmuestrap[2]=digitos[2]; //Guarda valor en frecuencia de sensor
        vmuestrap[3]=digitos[3]; //Guarda valor en frecuencia de sensor
    }
}
config+=1; //Se suma configuración
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) // Se presiona D
{
    r=3; //Sensor de Luz
    for(i=0; i<=0x100000; i++){ // Ciclo de espera
}

/* Muestra de datos */

GPIO_PORTA_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_PORTA_DATA_R=0xFD; // Habilita dígito 3
```



	<p align="center"><b>UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO</b>  <b>FACULTAD DE INGENIERÍA</b>  <b>Microprocesadores y Microcontroladores</b></p>	<p>Semestre: 2021-1</p>	 <p align="center"><b>INGENIERÍA ELÉCTRICA ELECTRÓNICA</b></p>
	<p><b>Prof. Dr. Saúl De La Rosa Nieves</b></p>	<p>Página 23 de 27</p>	

```

GPIO_PORTM_DATA_R=digitos[2];//Enciende dígito 3
GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
GPIO_PORTM_DATA_R=0xFB;// Habilita dígito 2
GPIO_PORTM_DATA_R=digitos[1];//Enciende dígito 2
GPIO_PORTM_DATA_R=0x00;// Apaga los 7 segmentos
GPIO_PORTM_DATA_R=0xF7;// Habilita dígito 1
GPIO_PORTM_DATA_R=digitos[0];//Enciende dígito 1
GPIO_PORTM_DATA_R=0x00;
}
}
void frecuencia(int frecuenciar)//Función frecuencia
{
    int f;
    f=1600000/frecuenciar;//Convierte frecuencia en ciclos de reloj
    GPIO_PORTK_DATA_R=0x10;//Monitorea valor de la tecla # ingresada
    while(GPIO_PORTK_DATA_R!=0x18)//Si se teclea #
    {
        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
        }
    }
    /*digitos[0]=0x00;
    digitos[1]=0x00;
    digitos[2]=0x00;
    digitos[3]=0x00;*/
}

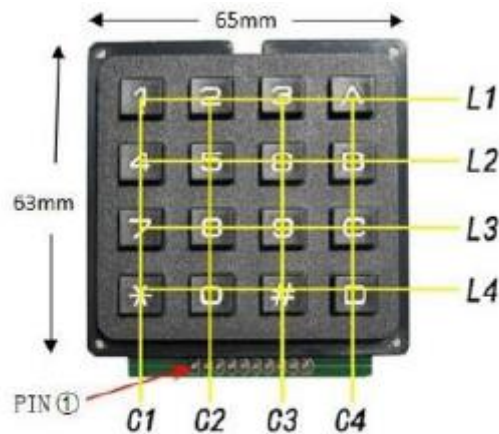
```

## V. Construcción

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.

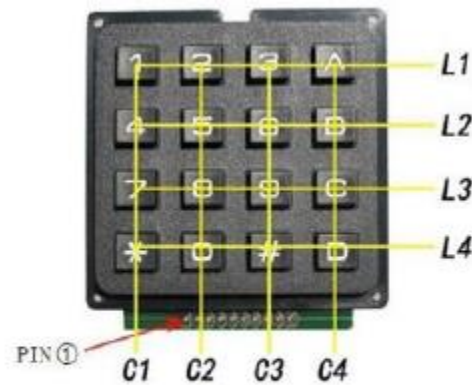
### *Conexión al teclado matricial*

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.



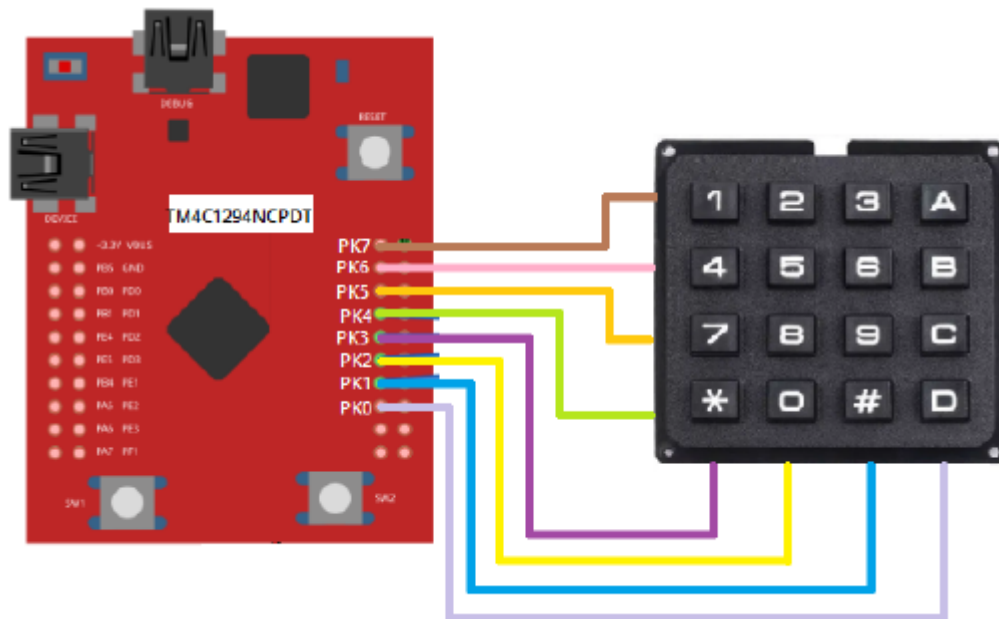
PIN	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
	NC	C1	C2	C3	C4	L1	L2	L3	L4	NC

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



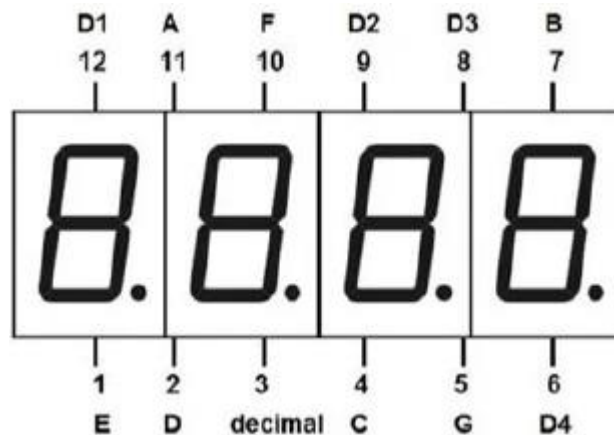
PIN	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
	NC	PK3	PK2	PK1	PK0	PK4	PK7	PK6	PK5	NC

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

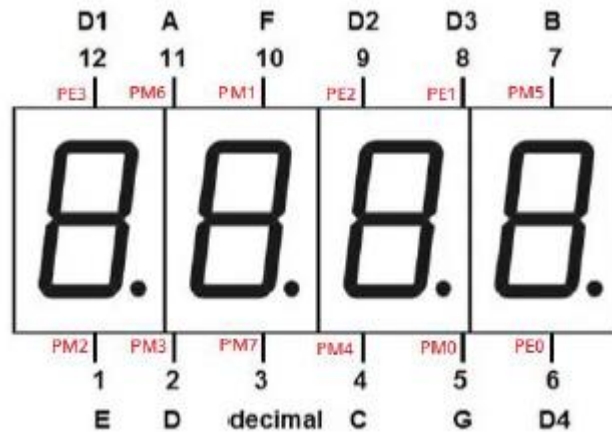


### Conexión al display 7 segmentos

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

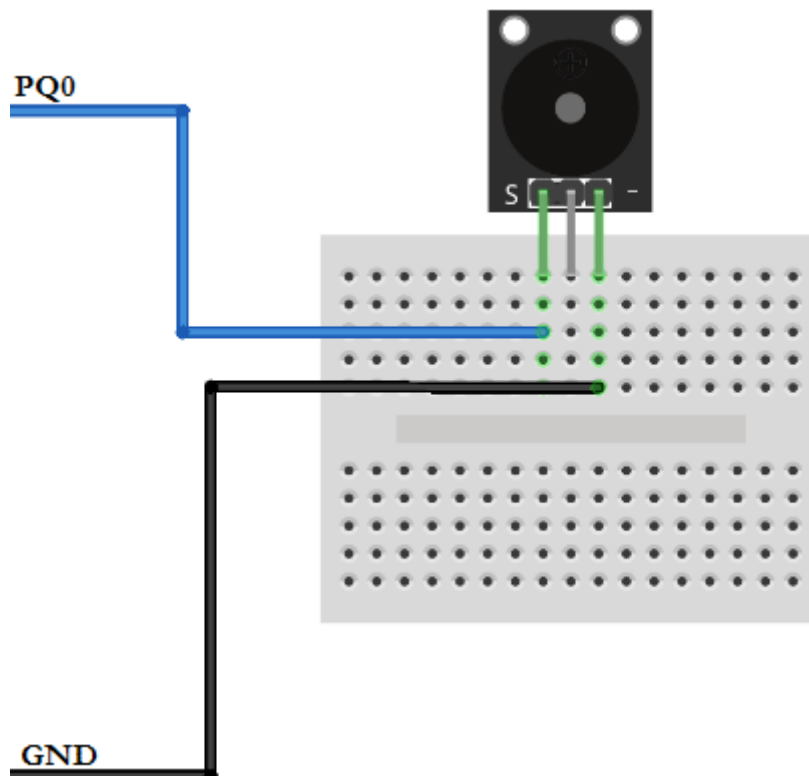


Para realizar las conexiones utilicé el puerto E para la habilitación de los dígitos y el puerto M para los segmentos por lo que la conexión realizada al microcontrolador fue la siguiente.





### *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.



## **VI. Resultados y – Conclusiones**

Esta tarea tuvo como finalidad la programación de un sistema de monitoreo, realizado en programación en C la cual nos ayuda a interactuar más fácilmente con el microcontrolador y aprender más temas sobre él, en este caso lo complicado del trabajo fue utilizar el ADC y las interrupciones, por lo que fue una buena decisión el dejar esto en C dado que si fuese en ensamblador se utilizarían códigos mucho más largos, en general este proyecto sirve para aprender a usar el microcontrolador y dominar algunas de las herramientas más poderosas que tiene como lo son las interrupciones. Leer y escribir en el

	<b>UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO</b> <b>FACULTAD DE INGENIERÍA</b> <b>Microprocesadores y Microcontroladores</b>		Semestre: 2021-1	 <b>INGENIERÍA ELÉCTRICA ELECTRÓNICA</b>
	<b>Prof. Dr. Saúl De La Rosa Nieves</b>	<b>Grupo 3</b>	Página 27 de 27	

microcontrolador mediante puertos de entrada y salida también fue necesario por lo que se abordaron gran parte de los temas requeridos para la materia.

## VII. Bibliografía

- <https://puntoflotante.net/MANUAL-DEL-USUARIO-SENSOR-DE-MOVIMIENTO-PIR-HC-SR501.pdf>
- <https://www.ti.com/lit/ds/symlink/lm35.pdf>