Instituto Politécnico Nacional

Escuela Superior de Cómputo

Departamento de Ciencias e Ingeniería de la Computación

Sistemas en Chip

Práctica 19

"Interrupciones externas"

Profesor: Fernando Aguilar Sánchez

Grupo: 6CM1

Equipo 3

Alumnos:

Ocampo Téllez Rodolfo

Patlani Mauricio Adriana

Ruvalcaba Flores Martha Catalina

Sandoval Hernández Eduardo

Fecha de entrega: 15/01/2023

Objetivo

Al término de la sesión, los integrantes del equipo contaran con la habilidad para manejar las interrupciones del Microcontrolador.

Introducción Teórica

El microcontrolador ATMEGA8535 fabricante ATMEL. En la figura 1 se muestra el microcontrolador ATMEGA8535 la cual maneja datos de 8 bits es decir su bus de datos de 8 bits. Aunque este contiene tres registros los cuales son el *x*, *y* y *z*, los cuales manejan datos de 16 bits.

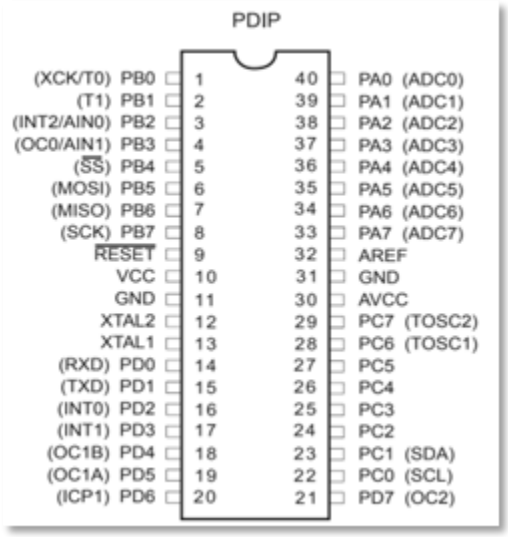

Figura 1. Configuración de pines ATmega8535

El microcontrolador utiliza una arquitectura cerrada, es decir, aquel que es inmodificable por los programadores ajenos a la compañía propietaria del código fuente. Por lo tanto, a este sistema no se le pueden colocar dispositivos periféricos, solo se usa el hardware de la compañía propietaria ya que los dispositivos ajenos a dicha compañía no son compatibles.

El microcontrolador ATMEGA8535 utiliza un encapsulado DIP-40, común en la construcción de circuitos integrados que consiste en un bloque con dos hileras paralelas de pines, observar la figura 2.

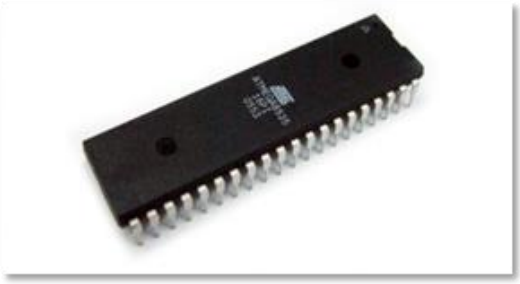

Figura 2. Microcontrolador atmega853.

La comunicación interna del microcontrolador se categoriza en 4 puertos, en la figura 1 se puede analizar la etiquetación de los puertos PA0 al PA7, PB0 al PB7, PC0 al PC7 y PD0 al PD7. Cabe recordar que maneja datos de 8 bits.

El microcontrolador se alimenta de las terminales 10 y 11 como lo muestra la figura 1, los cuales son el VCC (5 V y una tolerancia de ± 0.5V) y GND. Sin embargo, el convertidor se alimenta de forma externa en la terminal 31 y 32, los cuales son GND y AREF.

Posee un oscilador interno de 1MHz, sin embargo, como es un oscilador RC, es susceptible a variar la frecuencia con respecto a las variaciones de temperatura. Por otro lado, puede conectarse un oscilador de 0 a 8 MHz o de 0 a 16 MHz.

Display Cátodo Común

El display de 7 segmentos es un dispositivo opto-electrónico que permite visualizar números del 0 al 9 y o algunas letras. El dispositivo cuenta con 7 leds, uno por cada segmento, a cada uno de estos se le asigna una letra que va de la "a" a la "g" y se encuentran organizados como indica la figura 3.

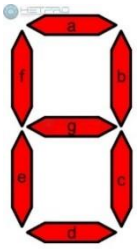

Figura 3. Disposición de los leds en un display de 7 segmentos.

Existen dos tipos de displays de 7 segmentos, de ánodo y cátodo comunes cuya diferencia se encuentra en la forma en que van conectados como lo indica la figura 4.

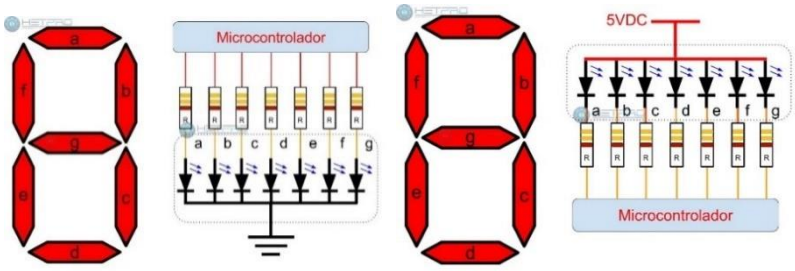

Figura 4. Conexiones en display de 7 segmentos.

Es importante mencionar que los display de 7 segmentos, dado que están construidos con diodos LED, requieren colocar una resistencia para limitar la corriente. Dicha resistencia depende de la corriente que se quiera suministrar al LED así como de la caída de voltaje. Para calcular la resistencia usamos la Ley de Ohm.

Flanco de subida y bajada

En una señal digital, se denomina flanco a la transición del nivel bajo al alto (flanco de subida) o del nivel alto al bajo (flanco de bajada). En este caso lo que interesa a los efectos de determinar que existe un requerimiento de interrupción es la existencia de un cambio de nivel bajo ("0") a alto ("1"), es decir un flanco, en la entrada INT. Es decir que no importa el valor absoluto actual de la entrada INT, lo que se tiene en cuenta es si existió un cambio de 0 a 1. Si hubo un flanco y aún no ha sido invocada la rutina de atención, la CPU lo hace y da por cumplido el pedido.

Interrupciones

Una interrupción consiste en un mecanismo que provoca la alteración del orden lógico de ejecución de instrucciones como respuesta a un evento externo, generado por el hardware de entrada/salida en forma asincrónica al programa que está siendo ejecutado y fuera de su control. Los controladores de E/S disponen de una señal de salida (habitualmente denominada INT ó INTR ó IRQ) que toma el valor "1" cuando el controlador interrumpe. Normalmente esa salida es el reflejo hardware de un bit del registro de ESTADO. Es decir que el nivel actual (0 ó 1) de la señal INT del controlador de E/S puede leerse en un bit de dicho registro. Este bit de "pedido de interrupción pendiente" y su correspondiente reflejo en el hardware permanece en nivel alto hasta que la rutina de atención de la interrupción realice alguna operación sobre los registros del controlador de E/S que satisfaga el pedido, con lo cual el controlador procederá a pasar ese bit (y su salida INT) a 0. Cuál es esa acción dependerá del controlador y, posiblemente, de la condición que provocó el pedido. Algunos pedidos de interrupción se satisfacen con la lectura del registro de estado, otros requieren de la lectura de algún registro adicional, otras requieren de la escritura de un registro, etc.

Materiales y Equipo empleado

- CodeVision AVR
- AVR Studio 4
- Microcontrolador ATmega 8535
- 1 Display cátodo común
- 2 Push Button
- 1 Resistor de 1KΩ

Desarrollo Experimental

1.- Diseñe un programa que cuando haya un flanco de subida en INT0 se incremente el conteo del display que podrá contar de 0 a 9, y que cuando haya un nivel lógico de 0 en INT1 se decremente el valor del display.

Número Display	Combinaciones								Valor Hexadecimal
	.	g	f	e	d	c	b	a	
0	0	0	1	1	1	1	1	1	0x3F
1	0	0	0	0	0	1	1	0	0x06
2	0	1	0	1	1	0	1	1	0x5B
3	0	1	0	0	1	1	1	1	0x4F
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6D
6	0	1	1	1	1	1	0	1	0x7C
7	0	0	0	0	0	1	1	1	0x07
8	0	1	1	1	1	1	1	1	0x7F
9	0	1	1	0	1	1	1	1	0x6F

Tabla 1. Valores hexadecimales correspondientes al display.

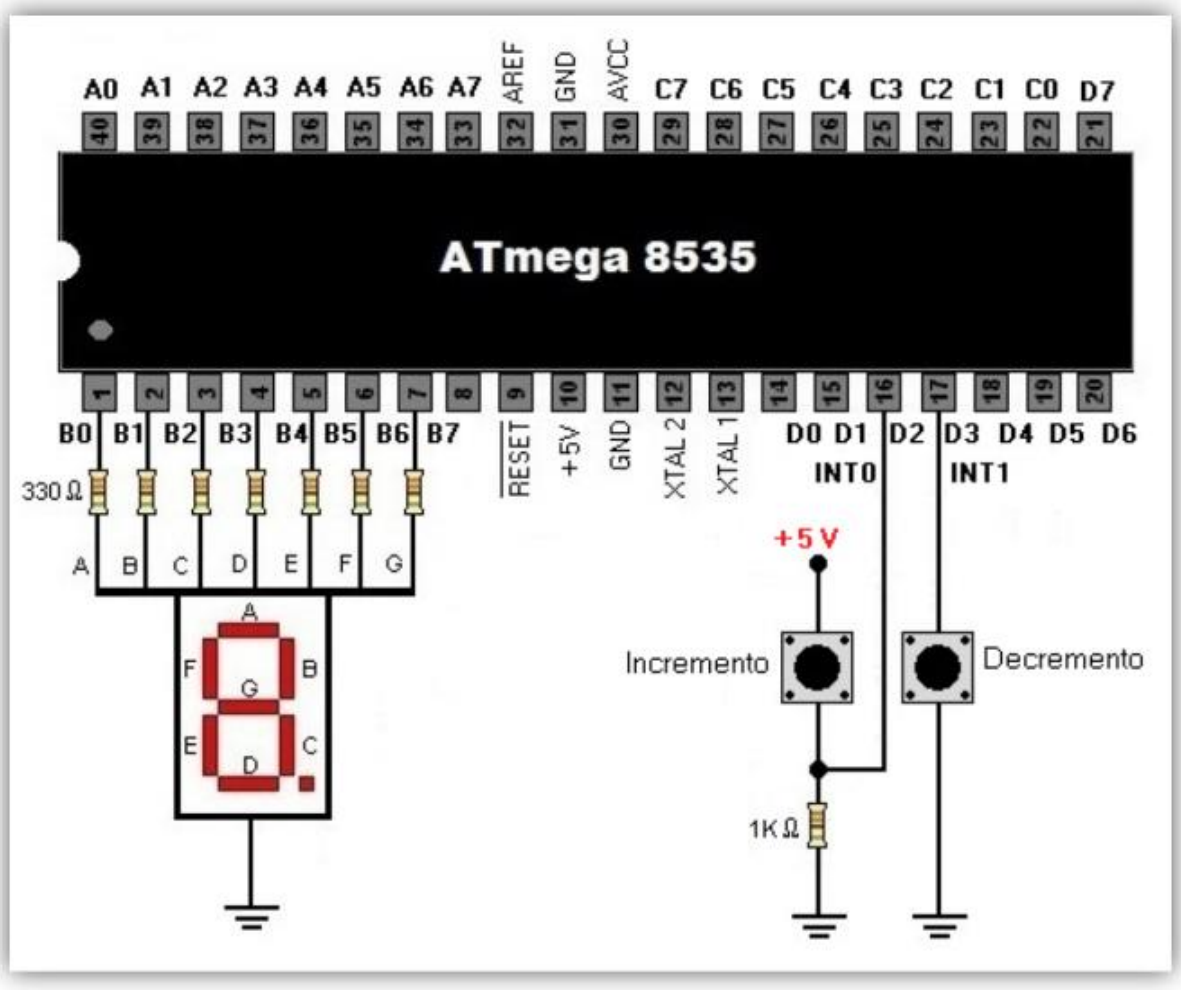

Figura 5. Circuito para las interrupciones INT0 y INT1.

Estructura del programa

Código de la configuración de los periféricos utilizados y código del programa principal en C, proporcionados por el IDE de CodeVision.

```
##include <mega8535.h>

#include <delay.h>
char numero = 0;

// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
    numero++;
    if (numero == 10) numero = 0;
    delay_ms(50);

}
```

```
// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
    if (numero == 0) numero = 9;
    numero--;
    delay_ms(50);
}

const char tabla7segmentos[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F,
0x6F};

void main(void)
{
// Input/Output Ports initialization
// Port A initialization
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1)
| (0<<DDA0);
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2)
| (0<<PORTA1) | (0<<PORTA0);

// Port B initialization
DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) | (1<<DDB1)
| (1<<DDB0);
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2)
| (0<<PORTB1) | (0<<PORTB0);

// Port C initialization
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1)
| (0<<DDC0);
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2)
| (0<<PORTC1) | (0<<PORTC0);
// Port D initialization
DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) | (0<<DDD1)
| (0<<DDD0);
PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) | (1<<PORTD2)
| (1<<PORTD1) | (1<<PORTD0);
…
…
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) |
(0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Rising Edge
```

```
// INT1: On
// INT1 Mode: Rising Edge
// INT2: Off
GICR|=(1<<INT1) | (1<<INT0) | (0<<INT2);
MCUCR=(1<<ISC11) | (1<<ISC10) | (1<<ISC01) | (1<<ISC00);
MCUCSR=(0<<ISC2);
GIFR=(1<<INTF1) | (1<<INTF0) | (0<<INTF2);
…
…
// Globally enable interrupts
#asm("sei")

while (1)
      {
        PORTB = tabla7segmentos[numero];
      }
}
```

Simulaciones

La simulación se realizó en el software Proteus Design Suite, previamente cargado el programa en formato ".hex" obtenido del software CodeVision AVR, el cual es un software para el microcontrolador Microchip AVR y sus contrapartes XMEGA.

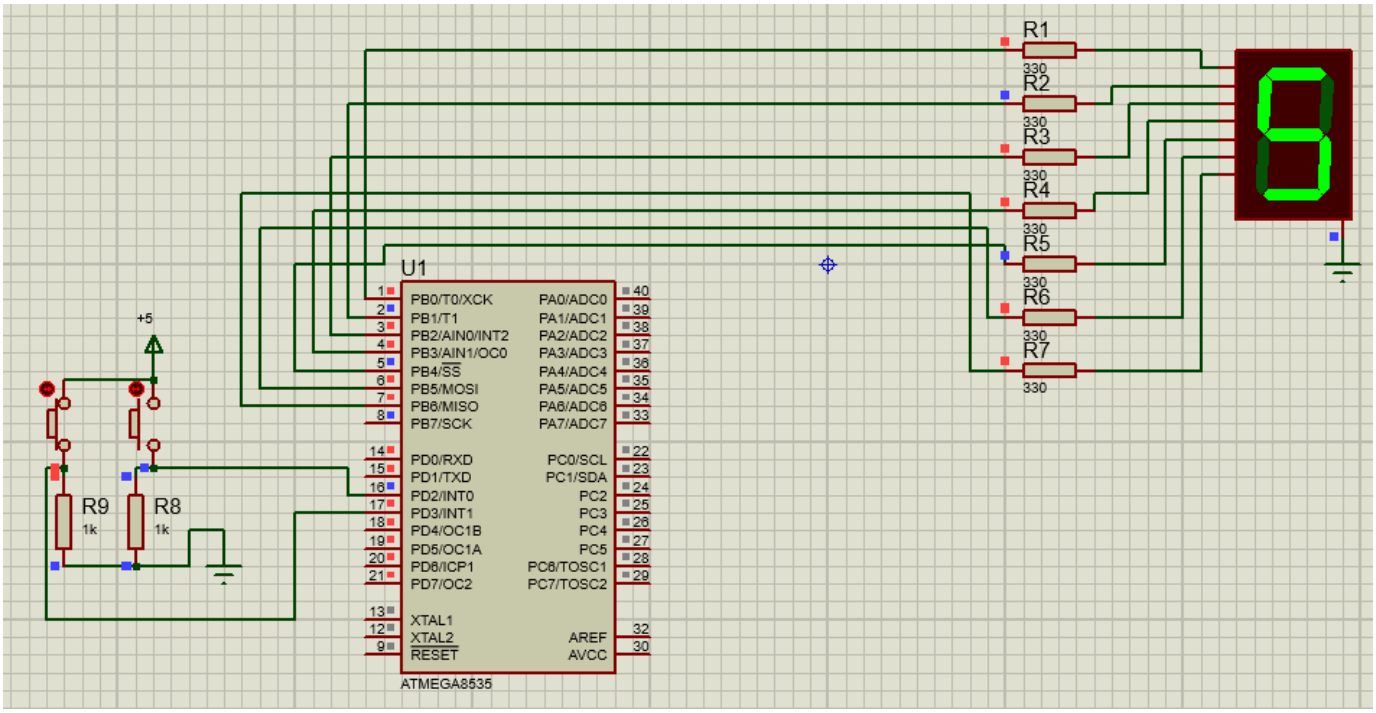

Figura 6. Simulación de la práctica 19.

Fotografía del circuito armado

A continuación, se muestra la figura 7 la evidencia sobre la realización y prueba de la practica 19.

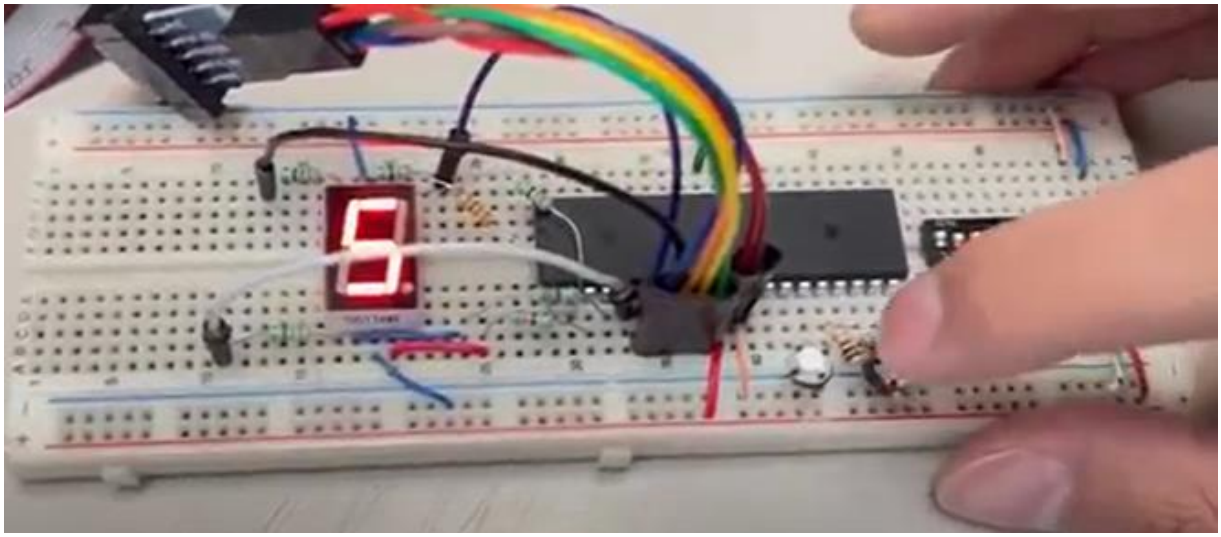

Figura 7. Circuito armado y funcionamiento de la práctica 19.

Observaciones y conclusiones Individuales

Ocampo Téllez Rodolfo

En esta práctica se exploró el manejo de interrupciones con el que cuenta el microcontrolador Atmega 8535, y precisamente en esto es en lo que se tuvieron varias complicaciones puesto que siguiendo las instrucciones originales en las que se manejaban dos distintos flancos dependiendo de cual caso de interrupción se diera (INT0 o INT1) solo permitía funcionar de forma correcta uno de los dos botones, y en el otro se mostraban números aleatorios no correspondientes a un incremento o decremento de 1 unidad. Se analizaron posibles causas del problema y con ello se probó incrementando el delay al pensar que se debía a un paso muy rápido de los números en el display, también se reconectó el circuito por si el problema era que alguna conexión estuviera errónea, sin embargo, ninguna resolvió el problema. La práctica funciono correctamente únicamente después de aplicar solo un flanco a ambas interrupciones, dicha solución fue explicada por el profesor.

Patlani Mauricio Adriana

Esta práctica fue de las que más conocimientos de otras prácticas se utilizan por el uso de varios tipos de componentes electronicos. En esta práctica se implementa el manejo de las interrupciones y manejos de flancos de bajada y de subida (INT0 y/o INT1) que se manejan desde la codificación mas que desde los componentes electrónicos como lo son los push bottons. En esta parte se complicó ya que un boton funcionaba bien y el otro mostraba numeors aleatorios lo que hizo que se implementara la aplicación de un solo flanco y asi solucionar el problema

Ruvalcaba Flores Martha Catalina

Para el desarrollo de la práctica se utilizaron los conocimientos adquiridos en las practicas anteriores, puesto que se hace uso del display de 7 segmentos, la activación por flancos y la aplicación de los push-buttons. Entonces lo nuevo por aplicar sería las interrupciones, estos se configuran desde la creación del programa, ya que las interrupciones externas son uno de los recursos internos de los microcontroladores, si bien las interrupciones externas sirvieron para detectar un estado lógico o un cambio de estado a través del monitoreo de los botones para ascender y descender el conteo. Sin embargo, el equipo tuvo complicaciones al momento de probarlo en la práctica, ya que cada interrupción externa estaba con un estado de flanco, provocando que solo un botón funcionara correctamente y el otro botón provocara números aleatorios. Tal situación fue explicada por el profesor, comentando que las interrupciones externas pueden configurarse para detectar un nivel bajo de voltaje o una transición, ya sea por un flanco de subida o de bajada. Con excepción de INT2, que sólo puede activarse por flancos. Por lo que el profesor recomendó que solo se aplicara un flanco, ya sea de subida o de bajada. Finalmente, el equipo programó las interrupciones externas con flancos de bajada, logrando exitosamente el funcionamiento de la practica 19.

Sandoval Hernández Eduardo

Al comienzo parecía que esta práctica sería la más sencilla de todo el curso, sin embargo, se complicó debido a que el conteo ascendente no funcionaba de la manera esperada, al final la razón de esto fue que las interrupciones no funcionan adecuadamente con los flancos de bajada, por tanto, tuvimos que realizar ambas interrupciones con flancos de subida y solo de esa manera se solucionó el problema.

Bibliografía

- Atmel Corporation. (2006). 8-bit Microcontroller with 8K Bytes In-System Programmable Flash. [Online]. Disponible en: https://drive.google.com/file/d/1acpQaDlsyLHr3w3ReSgrlXRbshZ5Z-lb/view

- F. Cereijo. (2004, Diciembre 28). Microcontroladores PIC. [Online]. Disponible en: https://drive.google.com/file/d/1mAydVlkZYqtYXhFQ0Tnr_UjZyjiwIp90/view

- HetPro. (2019). Display 7 Segmentos ánodo y cátodo común. [En línea]. Disponible en https://hetpro-store.com/TUTORIALES/display-7-segmentos-anodo-catodo-comun/

- G. Panton. (2017). Código BCD. [En línea]. Disponible en: https://arquiconsamuel.blogspot.com/2017/08/codigo-bcd.html

- Microprocesadores. Edu.Uy. [En linea]. Diciembre 02, 2022, en: https://www.fing.edu.uy/tecnoinf/mvd/cursos/arqcomp/material/teo/arq-teo08.pdf

- Interrupciones Externas. Controlesdigitales.com. [En linea]. Diciembre 02, 2022, en: http://controlesdigitales.com/Libro_Felipe_Santiago/04_Cap_4_5_6_7.pdf