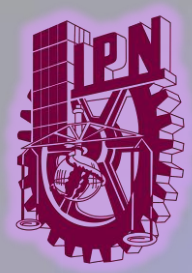

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Introducción a los microcontroladores

3CM16

Practica 01

Uso de puertos E/S

Equipo 7

Integrantes:

- Bocanegra Heziquio Yestlanezi
- Dominguez Durán Alan Axel
- Hernandez Mendez Oliver Manuel
- Martinez Cruz José Antonio

Profesor: Aguilar Sanchez Fernando

Contenido

Índice de imágenes

Objetivo

Al término de la sesión, los integrantes del equipo contaran con la habilidad de programar los puertos como entrada y salida del Microcontrolador ATmega8535 usando las herramientas “Code Vision AVR” y “AVR Studio 4”.

Introducción

Microcontrolador

Qué es un microcontrolador

Lon circuitos integrados programables que incorporan un procesador, memoria y periféricos en un solo chip. Estos dispositivos se utilizan en una amplia variedad de aplicaciones, desde electrodomésticos hasta sistemas de control de maquinaria [1].

Los microcontroladores suelen tener una arquitectura de procesador de 8, 16 o 32 bits, lo que significa que pueden manejar instrucciones de un tamaño determinado. También suelen incluir una cantidad limitada de memoria flash para almacenar el código del programa y RAM para almacenar los datos durante la ejecución del programa [2].

Además del procesador y la memoria, los microcontroladores también pueden incluir una amplia variedad de periféricos integrados, como convertidores analógico-digitales, interfaces de comunicación (como UART, SPI y I2C), temporizadores y puertos de entrada/salida para conectar sensores, actuadores y otros dispositivos [3].

Atmega8535

El microcontrolador ATmega8535 es un dispositivo de 8 bits fabricado por la empresa Microchip. Es parte de la familia de microcontroladores AVR y se utiliza comúnmente en proyectos electrónicos debido a su capacidad para controlar dispositivos externos [4].

Características

- Arquitectura de 8 bits RISC
- Frecuencia de reloj máxima de 16 MHz
- 8 KB de memoria flash programable
- 512 bytes de memoria EEPROM
- 512 bytes de memoria SRAM
- 32 pines de entrada/salidas programables
- Temporizador/Contador incorporado
- Convertidor analógico a digital de 10 bits
- Interfaz serial UART/SPI
- Interfaz de comunicación de 2 cables (TWI/I2C) [4].

CodeVisionAVR

CodeVision es un ambiente de desarrollo integrado (IDE) y un compilador de lenguaje C especialmente diseñado para microcontroladores AVR de la empresa Atmel/Microchip. El software incluye una amplia variedad de características y herramientas para facilitar el desarrollo de aplicaciones para microcontroladores AVR, como la programación de microcontroladores [5].

Material y equipo empleado

- ♥ CodeVision AVR
- ♥ AVR Studio 4
- ♥ Microcontrolador ATmega 8535
- ♥ 8 LED's
- ♥ 8 Resistores de 330 Ω a 1⁄4 W
- ♥ 1 Dip switch u ocho Push Botón

Desarrollo experimental

Realiza un programa para programar el Puerto B como entrada y escribir la información en el Puerto D activándolo como salida, recuerde activar las resistencias de Pull-up del puerto B para colocar solo el Dipswitch.

Como abrir un nuevo proyecto en CodeVisionAVR

Seguimos los siguientes pasos como se muestra a continuación en la imagen 1.

⇒ File

⇒ New

⇒ Project

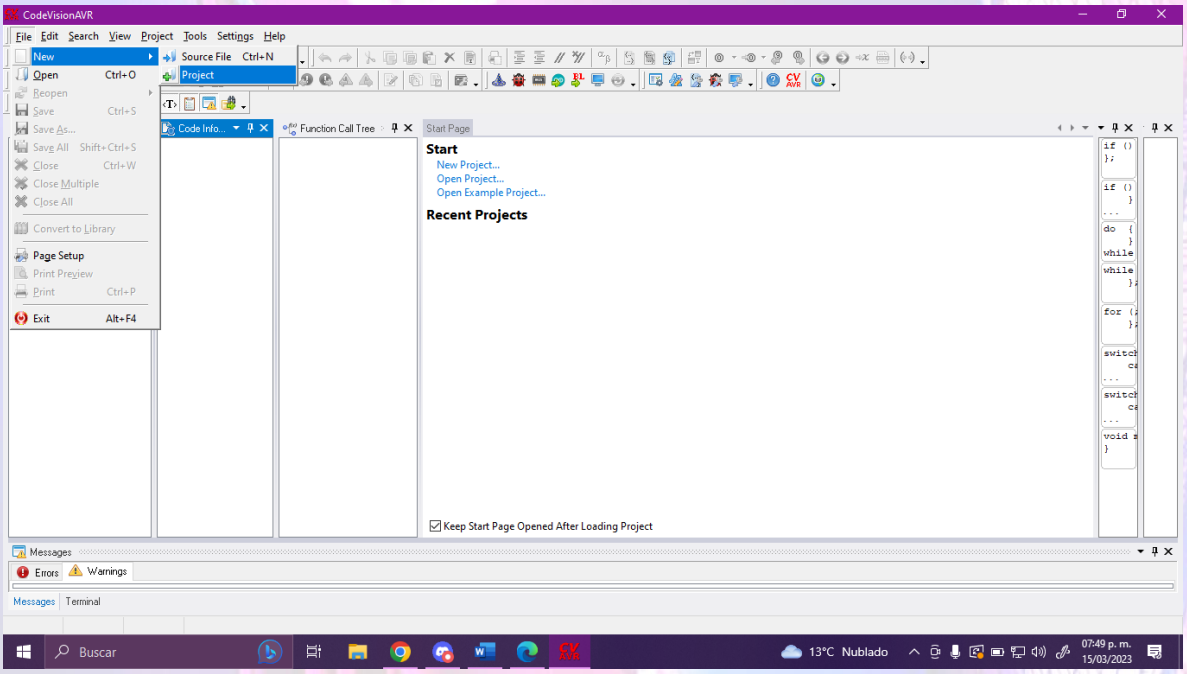

Imagen 1 CrearNuevoProyecto

A continuación le damos en “Yes” como se muestra en la imagen 2.

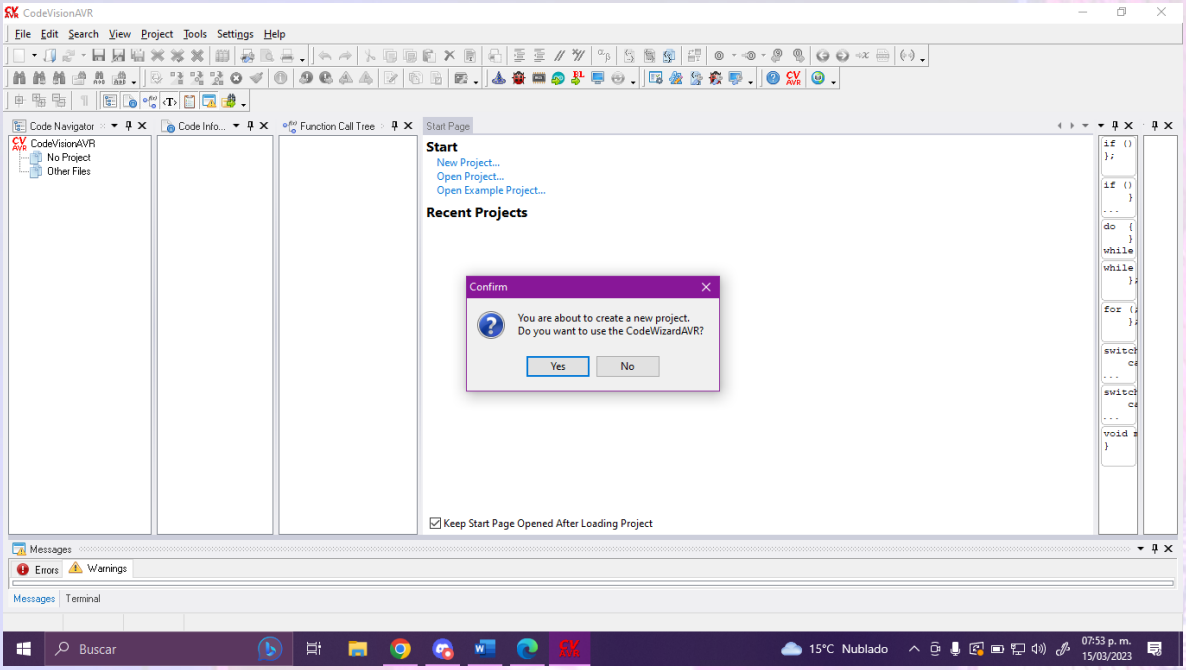

Imagen 2 CrearUnNuevoProyecto

A continuación le damos en “OK” como se muestra en la imagen 3.

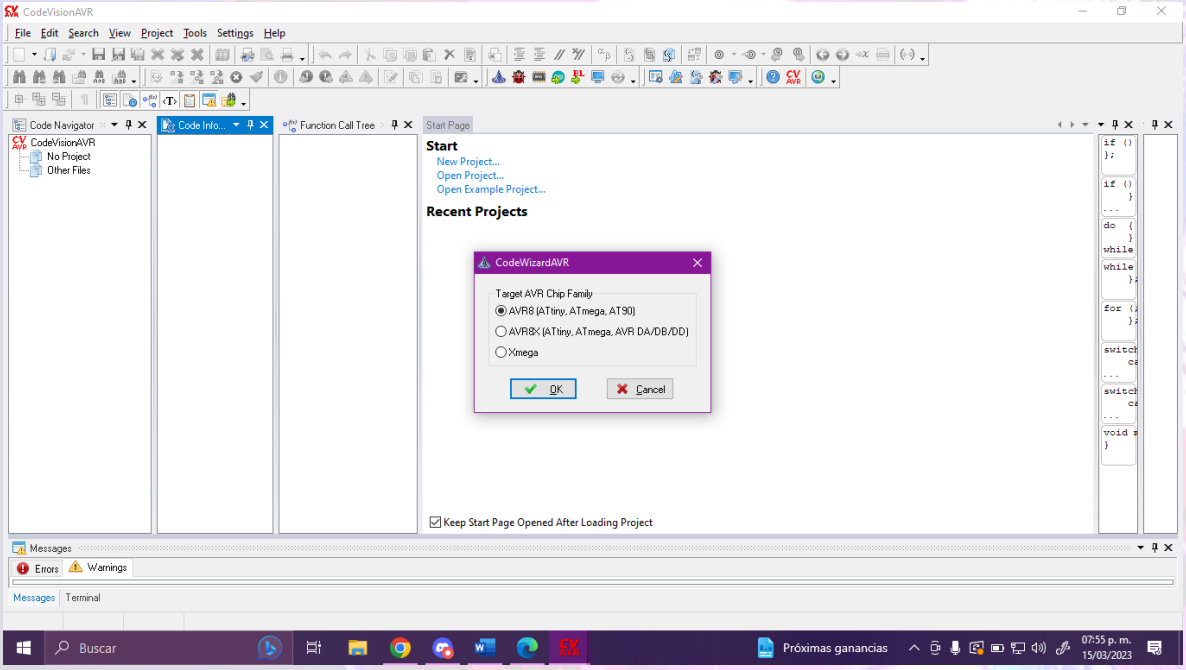

Imagen 3 CrearUnNuevoProyecto2

Seleccionamos el microcontrolador que vamos a utilizar, que en nuestro caso será el “Atmega8535” como se muestra en la imagen 4 y 5 .

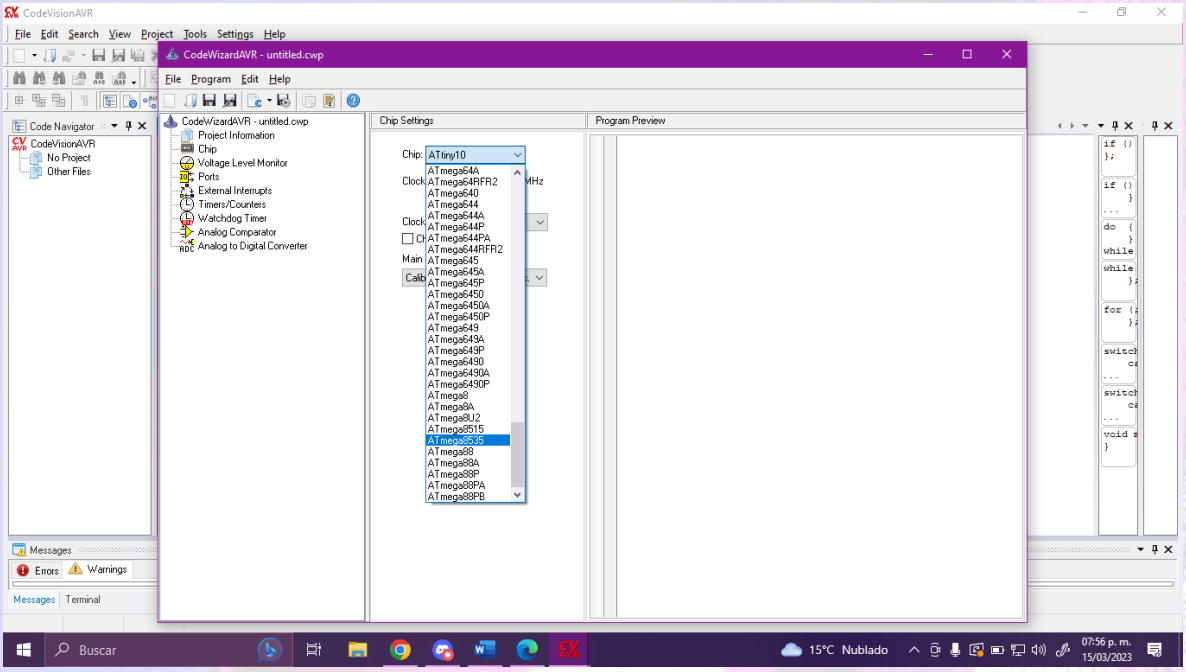

Imagen 4 Seleccionar microcontrolador

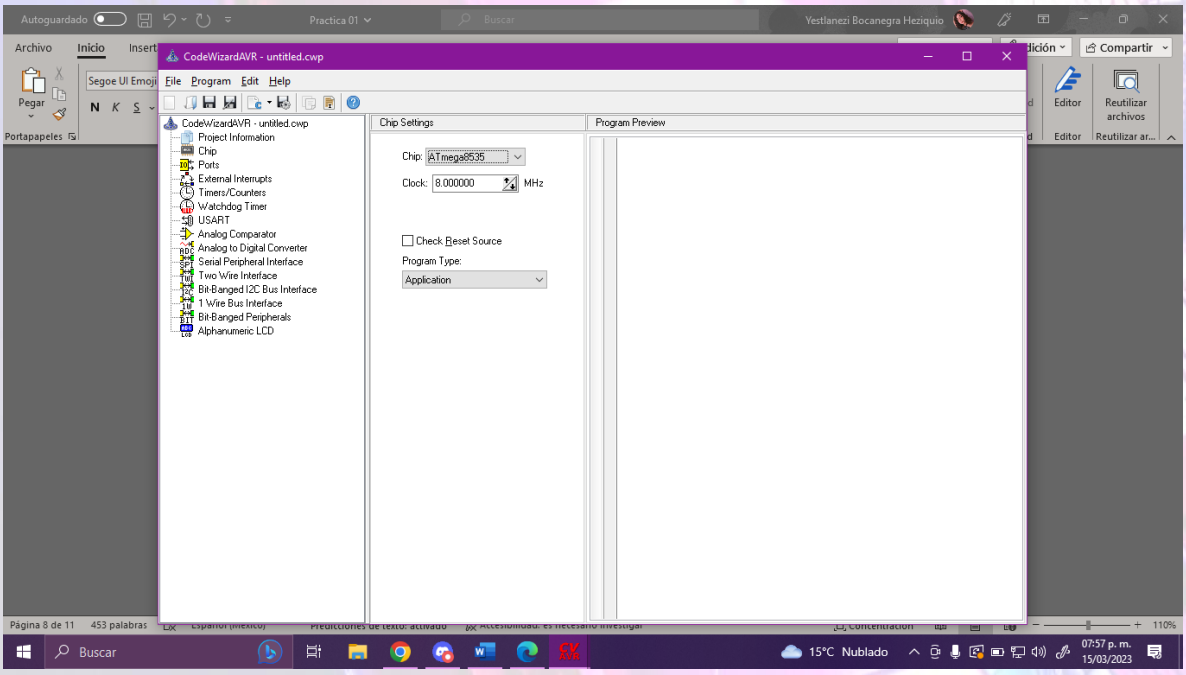

Imagen 5 Microcontrolador Seleccionado

A continuación, cambiamos la frecuencia a 1MHz como se muestra en la imagen 6.

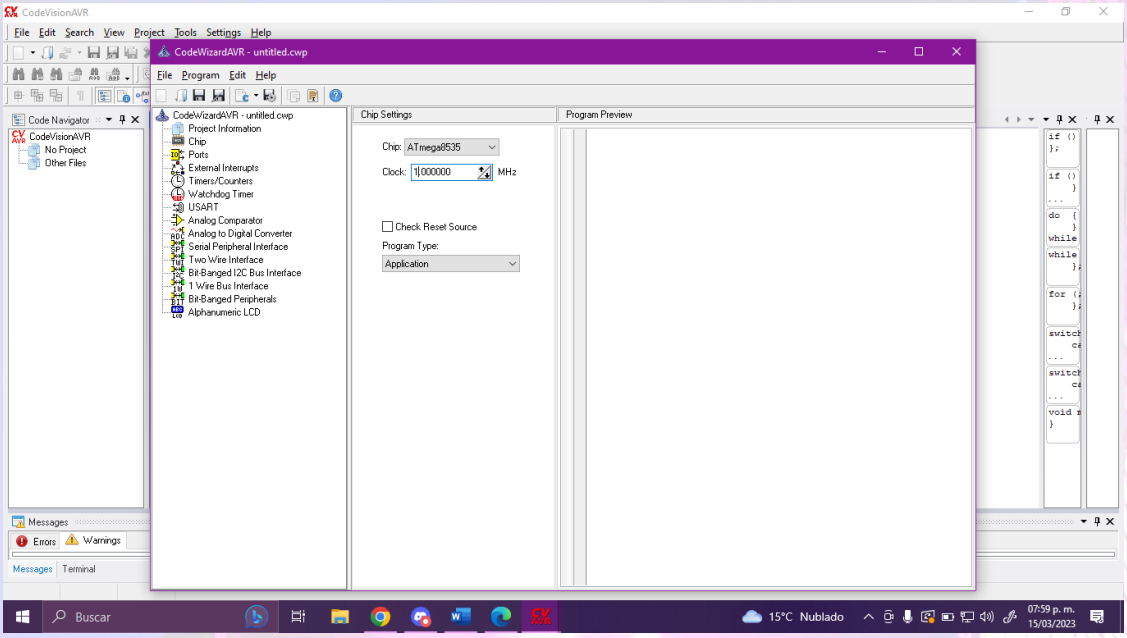

Imagen 6 Clock

A continuación, en el apartado de "Ports" cambiaremos el estado de puerto D como se muestra en la imagen 7.

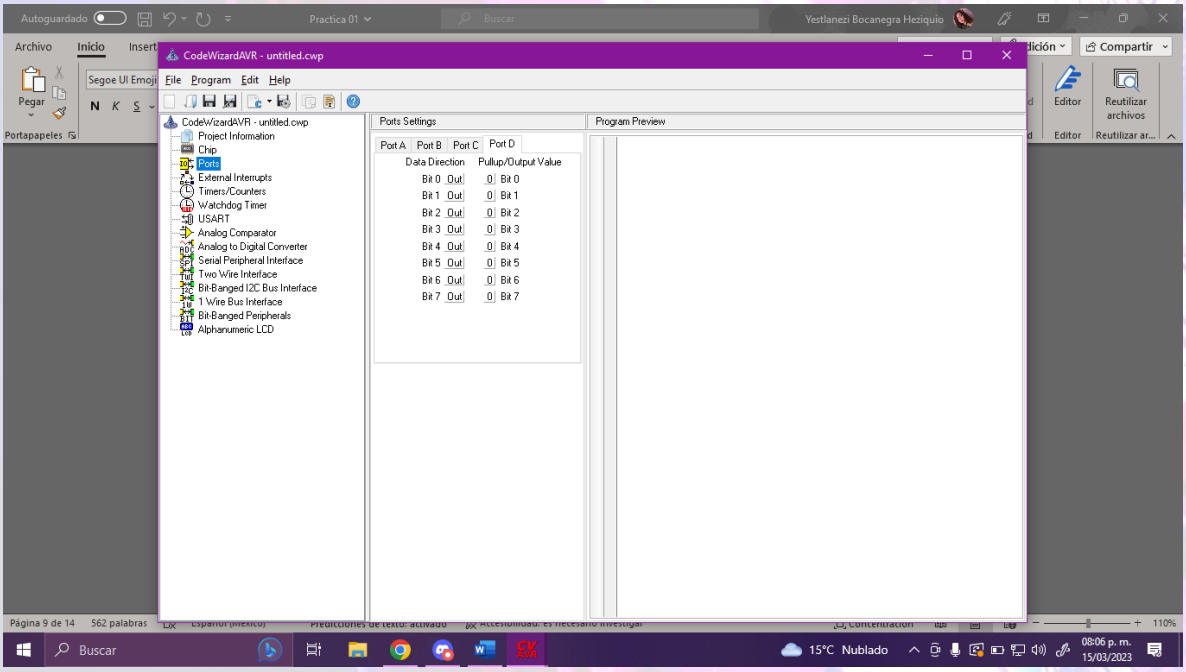

Imagen 7 Puerto D

Así como también cambiaremos los puertos en B como se muestra en la imagen 8.

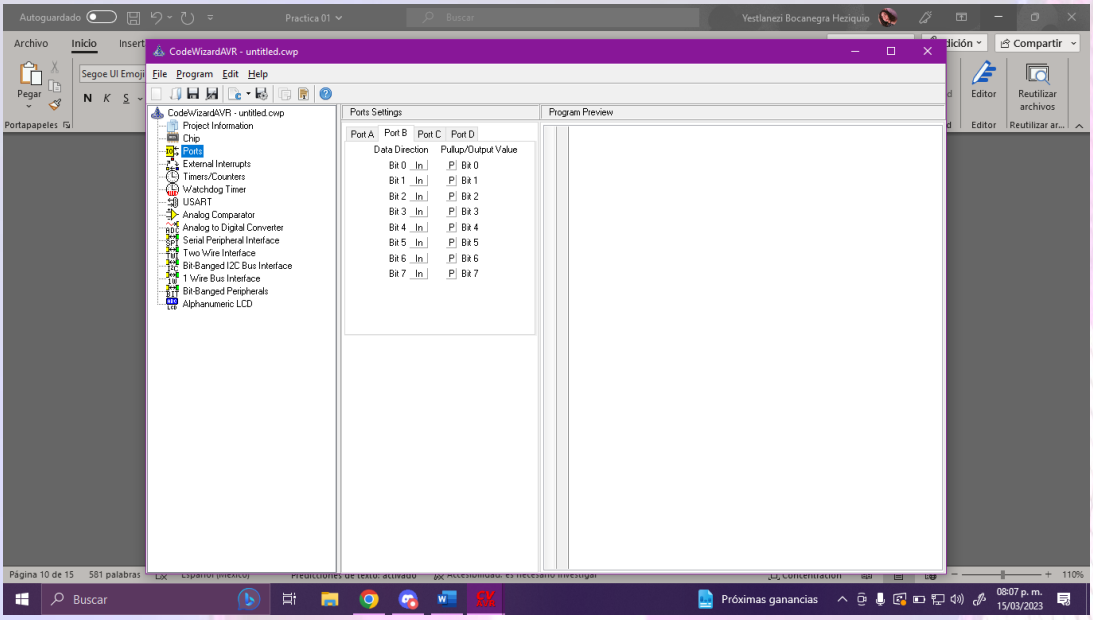

Imagen 8 Puerto B

A continuación, se debe guardar en el programa como se muestra en la imagen 9, el programa te pedía guardarlo 3 veces con el mismo nombre, en nuestro caso será "Practica 01" como se muestra en la imagen 10.

⇒ Program.

⇒ Generate, Save and Exit.

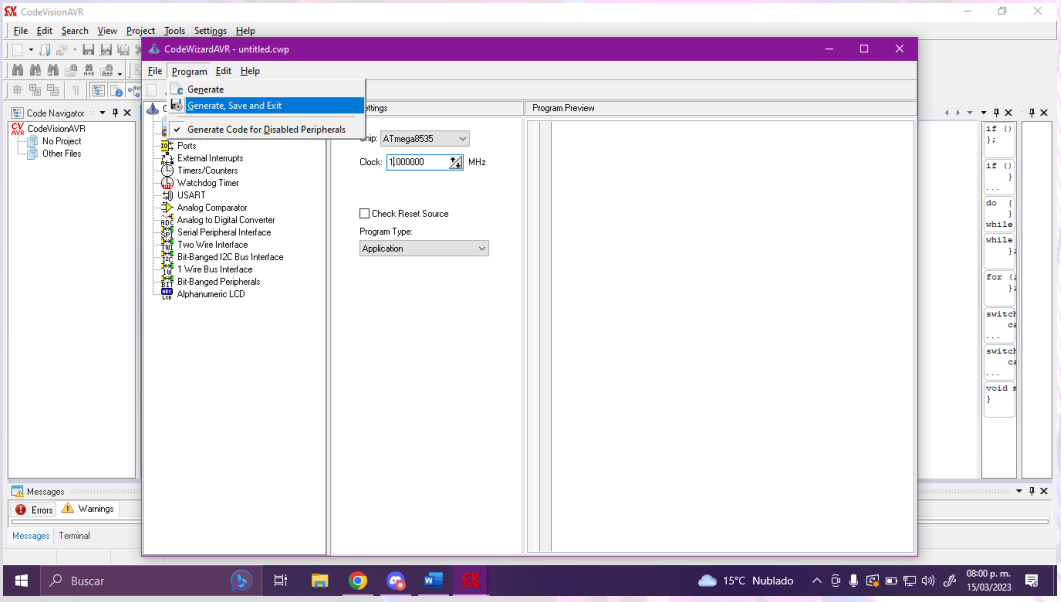

Imagen 9 Guardar proyecto "Practica 01"

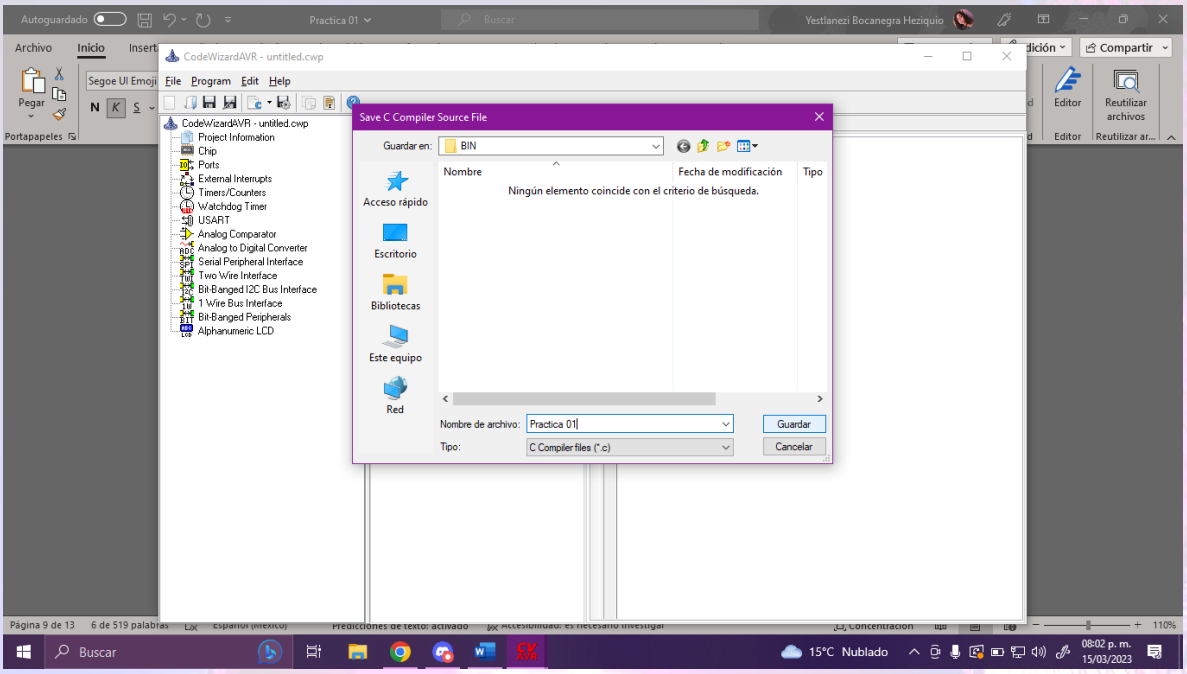

Imagen 10 Guardar 3 veces

A continuación, si los pasos fueron correctos, se genera el código como se muestra en la imagen 11.

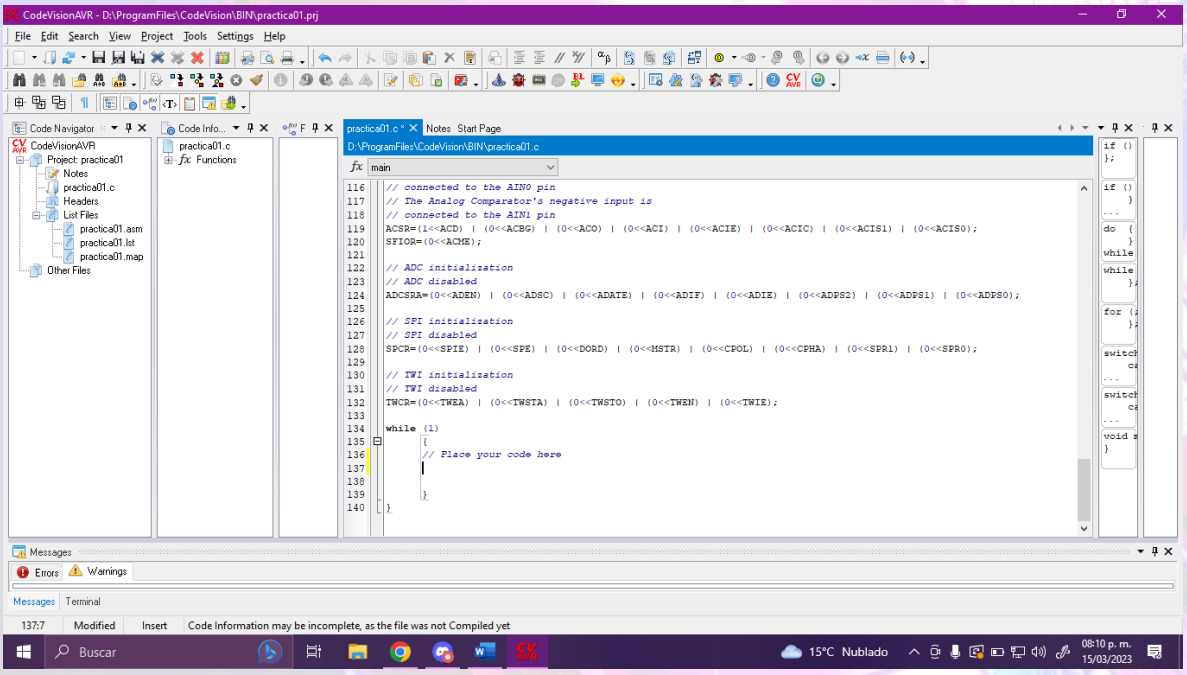

Imagen 11 Código generado

A continuación, modificaremos el código con el siguiente código:

```
while (1)
    {
    // Place your code here
        PORTD = PINB;
    };
}
```

A continuación, se muestra en la imagen 12.

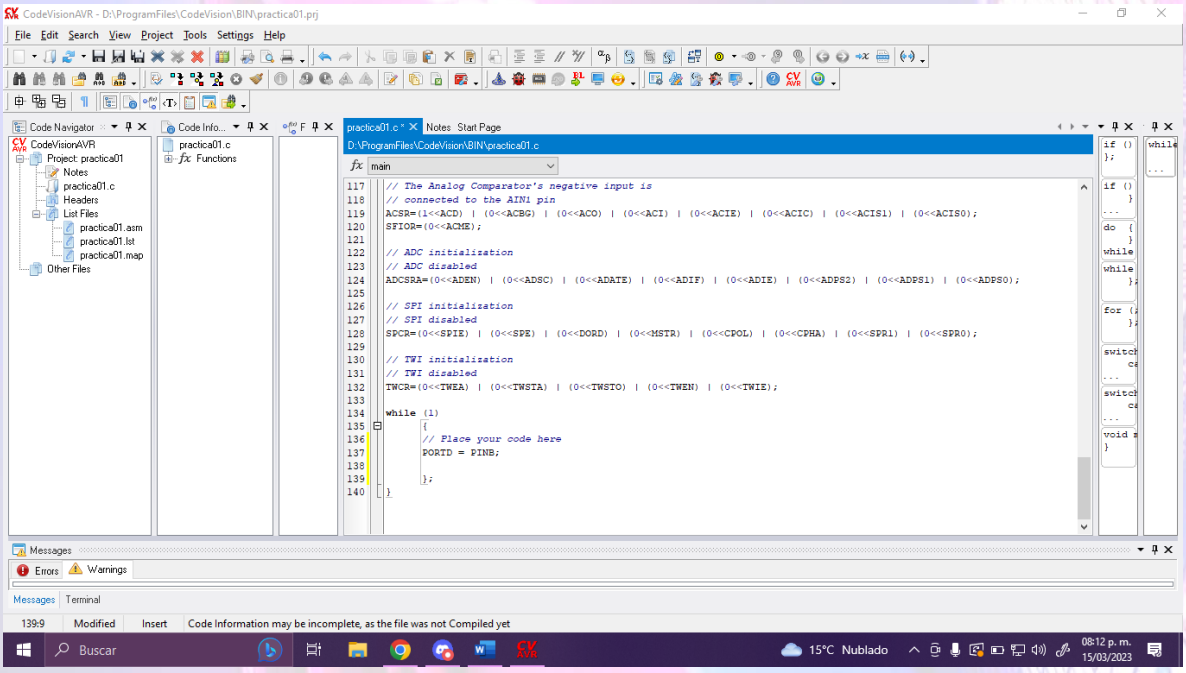

Imagen 12 Corrección de código

Circuito

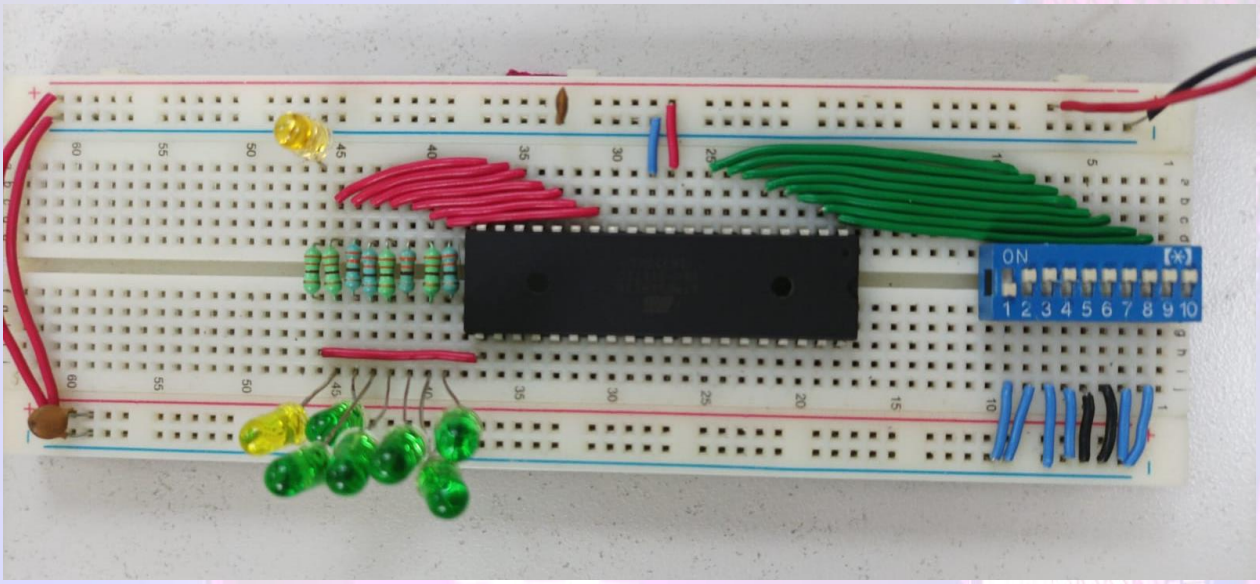

Imagen 13 Circuito armado

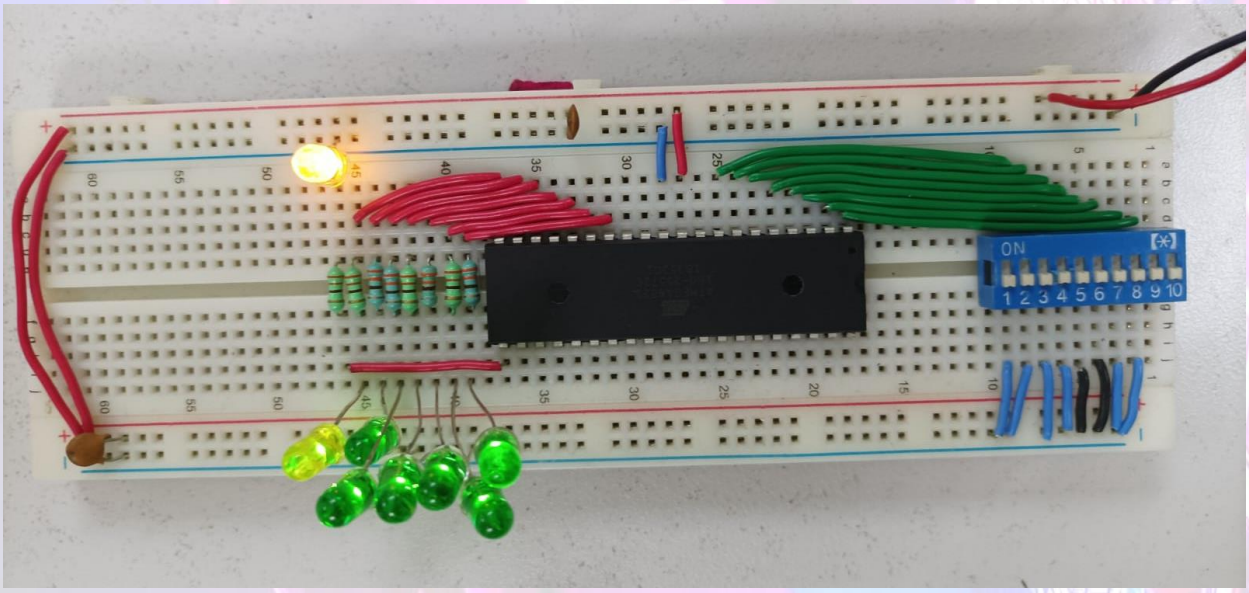

Imagen 14 Circuito funcionando

Al energizar nuestro circuito, podemos notar que todos los leds encienden.

Cuando vamos modificando los valores de entrada del switch, podemos notar como se apaga determinado led, dependiendo de la entrada que estamos modificando.

Estructura del programa

Código CodeVisionAVR

```
/*****************************************************
This program was created by the CodeWizardAVR V3.51
Automatic Program Generator
© Copyright 1998-2023 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 15/03/2023
Author  :
Company :
Comments:


Chip type               : ATmega8535
Program type            : Application
AVR Core Clock frequency: 1.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************************************************/

// I/O Registers definitions
#include <mega8535.h>

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) |
(0<<DDA1) | (0<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) |
(0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);

// Port B initialization
```

```
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) |
(0<<DDB1) | (0<<DDB0);
// State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
PORTB=(1<<PORTB7) | (1<<PORTB6) | (1<<PORTB5) | (1<<PORTB4) | (1<<PORTB3) |
(1<<PORTB2) | (1<<PORTB1) | (1<<PORTB0);

// Port C initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) |
(0<<DDC1) | (0<<DDC0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) |
(0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

// Port D initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out
Bit0=Out
DDRD=(1<<DDD7) | (1<<DDD6) | (1<<DDD5) | (1<<DDD4) | (1<<DDD3) | (1<<DDD2) |
(1<<DDD1) | (1<<DDD0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) |
(0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) |
(0<<CS01) | (0<<CS00);
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
```

```
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) |
(0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) |
(0<<CS11) | (0<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0<<AS2;
TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) |
(0<<CS21) | (0<<CS20);
TCNT2=0x00;
OCR2=0x00;

// 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: Off
// INT1: Off
// INT2: Off
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);

// USART initialization
// USART disabled
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) |
(0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
// connected to the AIN0 pin
```

```
// The Analog Comparator's negative input is
// connected to the AIN1 pin
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) |
(0<<ACIS1) | (0<<ACIS0);
SFIOR=(0<<ACME);

// ADC initialization
// ADC disabled
ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) |
(0<<ADPS2) | (0<<ADPS1) | (0<<ADPS0);

// SPI initialization
// SPI disabled
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) |
(0<<SPR1) | (0<<SPR0);

// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

while (1)
      {
      // Place your code here
      PORTD = PINB;

      };
}
```

Simulación

Proteus

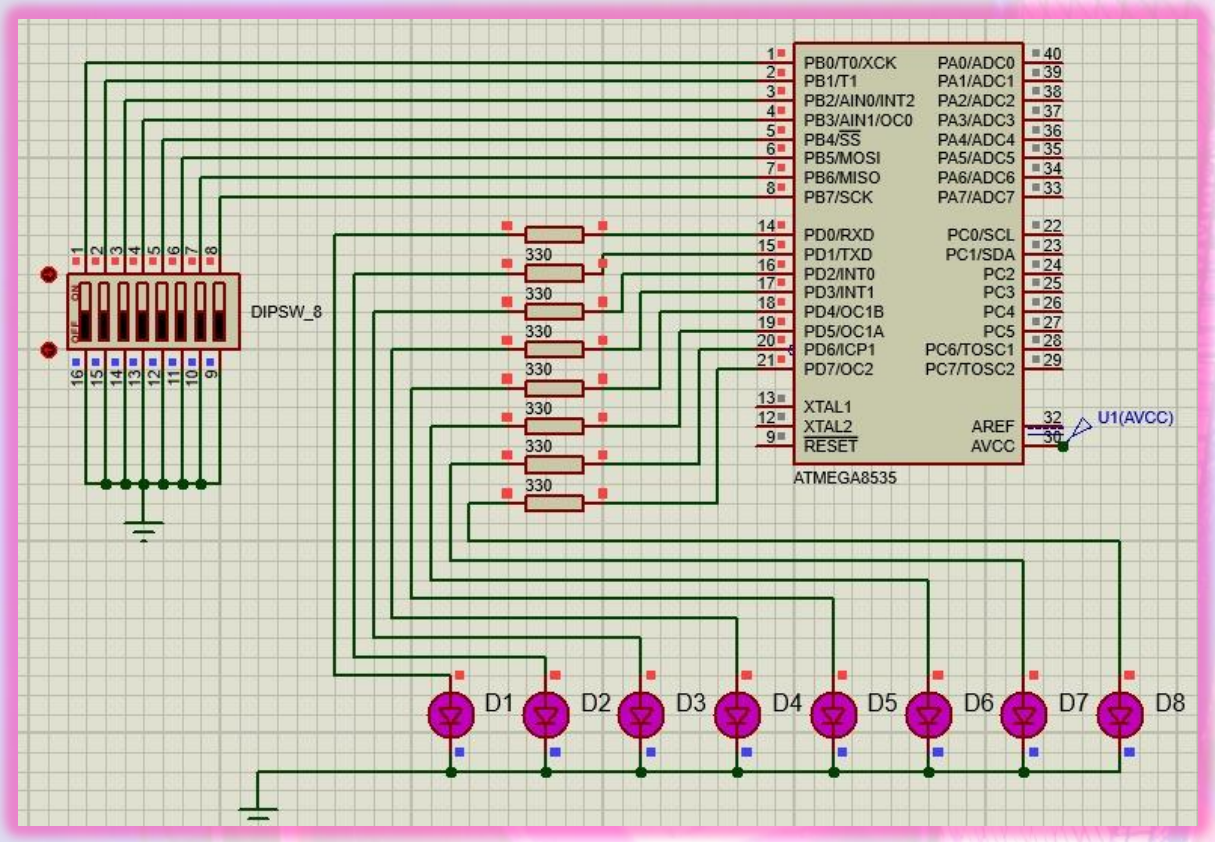

Imagen 15 Simulación del circuito funcionando en Proteus

Conclusiones

Bocanegra Heziquio Yestlanezi

Tuvimos complicaciones al momento de programar el microcontrolador, debido a que adquirimos el ATmega8535L, pudimos notar que todos los que teníamos este microcontrolador no podíamos programarlo, ya que aunque el circuito estaba bien, así como el programa, al momento de programarlo no tenía ningún error, pero cuando se intentaba encender no funcionaba, por lo que tuvimos que cambiar al microcontrolador ATmega8535 y efectivamente pudimos comprobar que el armado del circuito y el programa estaban bien, después de realizar este cambio no tuvimos mayor problema para realizar la práctica.

Dominguez Durán Alan Axel

Al finalizar esta práctica pudimos introducirnos al mundo de los microcontroladores, pudiendo diferenciarlo de un microprocesador, conocer el proceso de programación del mismo y además cablear un alambrado sencillo para demostrar su funcionamiento, sin duda es un buen preámbulo para realizar las prácticas siguientes y para poder familiarizarnos con la forma en que funciona.

Hernandez Mendez Oliver Manuel

A pesar de ser una práctica relativamente sencilla, funciona como una buena introducción para lo que viene posteriormente en el curso, el armado del circuito y la interacción con el programador me resultaron experiencias nuevas, pero interesantes, algo que considero importante fue que, en este caso, la selección del tipo de Microcontrolador si resultó una problemática ya que tuvimos problemas con él, sin embargo, la solución solo consistió en sustituirlo.

Gracias a esta práctica nos vamos a poder concentrar más en la parte a nivel de código en prácticas posteriores.

Martínez Cruz José Antonio

Con esta práctica nos introducimos a este nuevo apartado en la programación de micros y componentes para el uso de microcontroladores. Aunque surgieron diferentes inconvenientes en torno a la programación de nuestro atmega8535L, estos fueron soluciones con la adquisición de un atmega8535. Con esto logramos entender su comportamiento dentro del software de programación, su codificación y su instalación en la protoboard, permitiendo un flujo de trabajo más directo al momento de realizar las próximas prácticas.

Referencias

[1] Kim, H., & Park, J. (2021). A Design of Temperature Monitoring System Based on Microcontroller and Wireless Sensor Network for Smart Factory. IEEE Access, 9, 95625-95635.

[2] Nafea, M., & El-Sayed, A. (2020). An Embedded System for Real-Time Heart Rate Monitoring Using Microcontroller and Wireless Communication. IEEE Access, 8, 94731-94740.

[3] Maldonado, M., & Valderrama, C. (2019). Design and Implementation of a Microcontroller-Based System for Monitoring Carbon Monoxide Levels in Vehicles. IEEE Access, 7, 159280-159292.

[4] S. Bej, A. Kumar, and P. Kumar, "Design and Implementation of Automatic Irrigation System using Microcontroller," 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Kolkata, India, 2020, pp. 1-5. doi: 10.1109/ICIMIA48254.2020.9267553.

[5] CodeVisionAVR. (2021). CodeVisionAVR C Compiler. [Online]. Available: http://www.hpinfotech.ro/html/cvavr.htm [Accessed: Mar. 16, 2023].