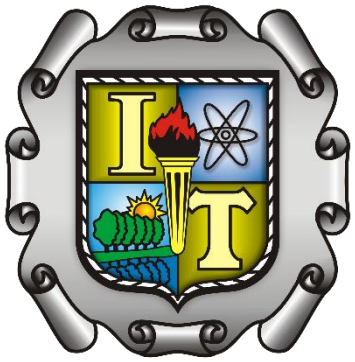


INSTITUTO TECNOLOGICO DE SALTILLO

REPORTE DE PRACTICA 1: CONTADOR BINARIO

Presenta:

Kevin Alejandro Fuentes Martínez

Orlando Ávila Cespedes

Eduardo Ernesto Castillo Cordero

Rogelio Pérez Guevara

Materia:
Sistemas Programables

Profesor:

ING. Mona Peña Luis Javier

04/10/2024

Índice

Objetivo general

- Programar un contador binario en Arduino y utilizar LEDs para mostrar los estados de los bits.

- Comprender cómo Arduino puede ser utilizado en la interacción con dispositivos electrónicos para realizar funciones básicas como el conteo binario.

Marco teórico

En esta práctica se ha implementado un contador binario utilizando Arduino como controlador y LEDs para la representación visual de los bits. Esta implementación permite profundizar en la programación de microcontroladores y la interacción con componentes electrónicos a través del lenguaje de programación de Arduino.

El contador binario es un circuito secuencial que, mediante el uso de señales digitales, puede representar números en su forma binaria. Arduino facilita la creación de este tipo de sistemas digitales, permitiendo el control de múltiples salidas, como los LEDs, que se encienden y apagan en función de los bits correspondientes al número binario.

Desarrollo de la práctica

Materiales y Equipos

- 1 Arduino UNO
- 4 LEDs
- Resistencias de 330 Ω para los LEDs y de 220 Ω para el display
- Protoboard y cables de conexión
- Fuente de alimentación USB para el Arduino

Procedimiento

Programación del Arduino

Se desarrolló un código para controlar los LEDs, donde cada uno representa un bit del contador binario, mostrando la secuencia binaria de los números del 0 al 15. Además, se agregó un display de 7 segmentos para mostrar los números en formato decimal.

Código:

```
// Pines para los LEDs
int indicatorPin = A0;     // Pin 8 para el indicador de encendido
int leds[] = {A1, A2, A3, A4}; // Definimos un array para los pines de los
LEDs
int delayt = 500;           // Velocidad de los LEDs
```

```
bool indicatorState = false; // Estado del pin 8 para hacer que parpadee

// Pines para los segmentos del display
int a = 12;
int b = 8;
int c = 4;
int d = 6;
int e = 7;
int f = 11;
int g = 3;

// Pines para los dígitos del display
int dg1 = 2;
int dg2 = 9;
int dg3 = 10;
int dg4 = 13;

// Velocidad de multiplexación
int vel = 5; // Velocidad del display para que no parpadee

// Unidades y decenas para el número en el display
int dec, uni;

// Segmentos (a-g) para los dígitos del display
byte segmentos[7] = {a, b, c, d, e, f, g};
byte orden_digitos[4] = {dg1, dg2, dg3, dg4};

// Configuración para los dígitos (0-9) en un display de 7 segmentos
byte siete_segmentos_digitos[10][7] = {
  { 1, 1, 1, 1, 1, 1, 0 },  // = 0
  { 0, 1, 1, 0, 0, 0, 0 },  // = 1
  { 1, 1, 0, 1, 1, 0, 1 },  // = 2
  { 1, 1, 1, 1, 0, 0, 1 },  // = 3
  { 0, 1, 1, 0, 0, 1, 1 },  // = 4
  { 1, 0, 1, 1, 0, 1, 1 },  // = 5
  { 1, 0, 1, 1, 1, 1, 1 },  // = 6
  { 1, 1, 1, 0, 0, 0, 0 },  // = 7
  { 1, 1, 1, 1, 1, 1, 1 },  // = 8
  { 1, 1, 1, 0, 0, 1, 1 }   // = 9
};

void setup() {
  // Configuramos el pin para el indicador de encendido
  pinMode(indicatorPin, OUTPUT);
```

```
  // Encendemos el pin indicador durante 2 segundos
  digitalWrite(indicatorPin, HIGH);
  delay(2000);

  // Configuramos los pines de los LEDs como salidas
  for (int i = 0; i < 4; i++) {
    pinMode(leds[i], OUTPUT);
  }

  // Inicializamos los pines del display como salidas
  for (int i = 2; i <= 13; i++) {
    pinMode(i, OUTPUT);
  }
}

void loop() {
  // Recorremos los números del 0 al 15
  for (int i = 0; i < 16; i++) {
    // Configuramos los LEDs para representar el número actual
    for (int j = 0; j < 4; j++) {
      int state = (i >> j) & 1; // Extraemos el bit correspondiente
      digitalWrite(leds[j], state == 1 ? HIGH : LOW); // Convertimos 1 y 0 a
HIGH y LOW
    }

    // Calculamos las unidades y decenas para el display
    uni = i % 10; // Unidad (0-9)
    dec = i / 10; // Decenas (0 o 1 para el rango 0-15)

    // Mostramos el número en el display
    EscribeDigito(2, dec);  // Decena
    delay(vel);
    EscribeDigito(1, uni);  // Unidad
    delay(vel);

    // Mil y Cien siempre deben mostrar 0 sin parpadeos
    //EscribeDigito(3, 0);    // Centenas
    //delay(vel);
    //EscribeDigito(4, 0);    // Millares
    //delay(vel);

    // Alternamos el estado del pin indicador para que parpadee
    indicatorState = !indicatorState;
    digitalWrite(indicatorPin, indicatorState ? HIGH : LOW);
```

```
    // Esperamos el tiempo definido para los LEDs (500 ms)
    delay(delayt);
  }
}

// Función para escribir un número en el dígito correspondiente
void EscribeDigito(byte digito, byte numero) {
  // Limpiamos los segmentos antes de escribir
  LimpiarSegmentos();

  // Activamos el dígito correspondiente
  ActivarDigito(digito);

  // Escribimos el número en el dígito activo
  for (byte i = 0; i < 7; i++) {
    digitalWrite(segmentos[i], siete_segmentos_digitos[numero][i]);
  }
}

// Función para activar un solo dígito a la vez
void ActivarDigito(int x) {
  // Desactivar todos los dígitos
  digitalWrite(dg1, HIGH);
  digitalWrite(dg2, HIGH);
  digitalWrite(dg3, HIGH);
  digitalWrite(dg4, HIGH);

  // Activar el dígito especificado
  switch (x) {
    case 1:
      digitalWrite(dg1, LOW);
      break;
    case 2:
      digitalWrite(dg2, LOW);
      break;
    case 3:
      digitalWrite(dg3, LOW);
      break;
    default:
      digitalWrite(dg4, LOW);
      break;
  }
}

// Función para limpiar los segmentos antes de escribir un nuevo número
```

```
void LimpiarSegmentos() {
  for (byte i = 0; i < 7; i++) {
    digitalWrite(segmentos[i], LOW);
  }
}
```

2.- Montaje del Circuito:

- Conectar los cuatro LEDs a los pines A1, A2, A3 y A4 del Arduino, utilizando resistencias de 330 Ω en serie con cada LED.

- Alimentar el Arduino a través del puerto USB.

3.- Ejecución

Al cargar el código en el Arduino, los LEDs representan la secuencia binaria de los números del 0 al 15. Cada LED representa un bit del número en binario, y el encendido o apagado de los LEDs cambia según el valor del contador.

Explicaciones del código

Contador Binario

El contador binario se implementa utilizando cuatro LEDs conectados a los pines A1, A2, A3, y A4 de la placa Arduino. El objetivo de este contador es mostrar la representación binaria de los números del 0 al 15, donde cada LED corresponde a un bit del número en cuestión.

Para lograrlo, se utiliza un bucle for que recorre los valores del 0 al 15. Dentro de este bucle, el código calcula el estado de cada LED usando una operación de desplazamiento de bits (i >> j) y una máscara (& 1) para extraer los valores binarios de cada bit del número actual. Por ejemplo, si el número es 5, la salida binaria será 0101, lo que hará que los LEDs en las posiciones 1 y 3 se enciendan mientras que los LEDs en las posiciones 2 y 4 permanezcan apagados.

Cada estado de los LEDs se actualiza usando la función digitalWrite, que establece cada LED como HIGH (encendido) o LOW (apagado) según el valor del bit correspondiente. Al hacer esto repetidamente, se visualiza un conteo binario completo, incrementando en cada iteración del bucle.

Además, el contador está sincronizado con un indicador de estado (indicatorPin), que parpadea en cada ciclo para indicar que el proceso está corriendo correctamente. La velocidad de este parpadeo se controla mediante la variable delayt, que introduce una pausa de 500 milisegundos entre cada incremento del contador binario.

Display

Como podrá verse en el código, agregamos un display para que los números que nos dé el contador binario se puedan mostrar del 1 al 9.

El display utilizado es de 7 segmentos, lo que permite representar números del 0 al 9 encendiendo combinaciones específicas de sus segmentos. Cada segmento se controla mediante pines conectados a la placa Arduino, asignados a los segmentos a, b, c, d, e, f y g. Además, el display cuenta con cuatro dígitos (dg1, dg2, dg3, y dg4), aunque en este proyecto nos enfocamos en utilizar solo dos de ellos para mostrar unidades y decenas.

El proceso para mostrar un número se basa en la multiplexación. Primero, se calcula la unidad (uni) y la decena (dec) a partir del número binario generado por los LEDs. Luego, se utiliza la función EscribeDigito para activar el dígito correspondiente (ya sea unidad o decena) y encender los segmentos correctos para representar el número. Este proceso ocurre a una velocidad controlada por la variable vel, lo que permite que el display no parpadee y muestre de manera estable los números.

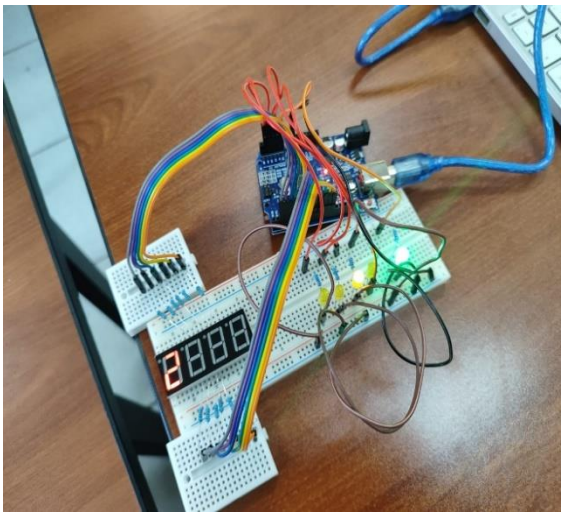
Finalmente, la función LimpiarSegmentos se encarga de apagar todos los segmentos antes de escribir el siguiente número, garantizando que no queden rastros de números anteriores encendidos en el display

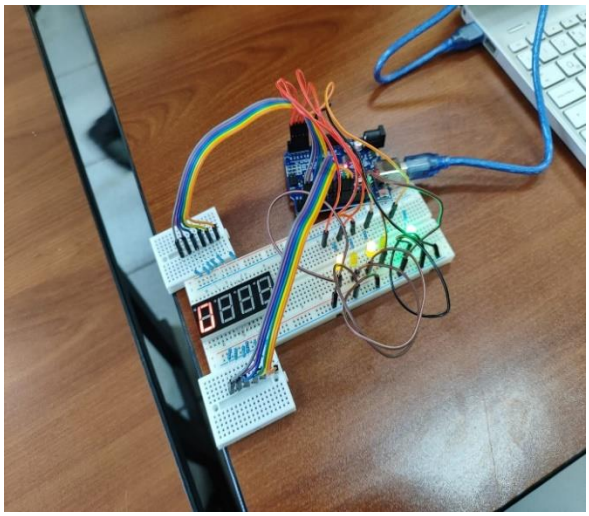
Resultados

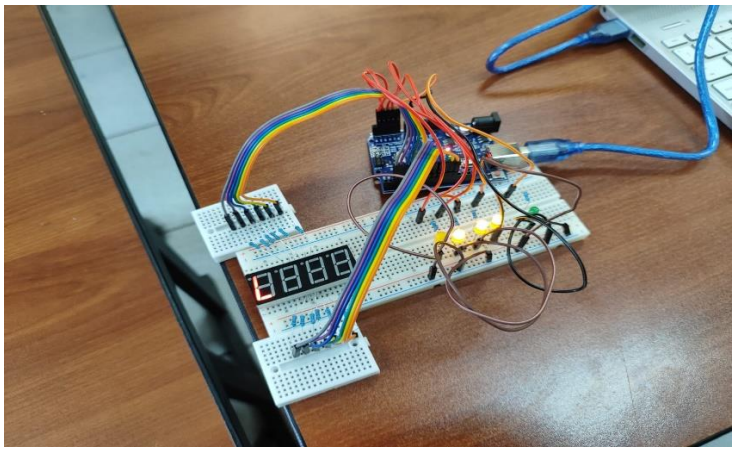
El siguiente cuadro muestra la secuencia de encendido de los LEDs, que corresponde a los números en binario del 0 al 15:

Estado Binario (LEDs)	Valor Decimal
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Imágenes del circuito







Conclusiones

El contador binario se implementó con éxito, mostrando de manera visual en los LEDs la secuencia binaria de los números del 0 al 15. Esta práctica permitió reforzar los conocimientos de binarios y su representación en un sistema digital, así como la programación y control de hardware utilizando Arduino.