Instituto Politécnico Nacional

Escuela Superior de Cómputo

Proyecto de Semáforo con Arduino

Sistemas en chip
Profesora: Ana Luz Barrales Lopez

Luciano Hernández Jonathan
Rodriguez Morales Iñaki

9 de octubre de 2024

Contents

1 Introducción

La automatización de sistemas de control es un campo fundamental en la ingeniería electrónica y de sistemas, donde la interacción entre hardware y software juega un papel crucial. En esta práctica, se desarrolló un sistema de semáforo utilizando un microcontrolador Arduino, que permite gestionar de manera eficiente el flujo de tráfico y la seguridad de los peatones. Este proyecto no solo tiene aplicaciones en entornos urbanos, sino que también representa una oportunidad para entender los principios de la programación, el diseño de circuitos y la integración de dispositivos electrónicos.

El objetivo principal de esta práctica fue construir un semáforo funcional que simule el comportamiento típico de un semáforo real, con LEDs que representan las luces verde, ámbar y roja. Además, se incorporó un botón de presión que permite interrumpir el ciclo de funcionamiento normal y activar un modo de emergencia, en el cual el semáforo se pone en rojo por un tiempo prolongado. Este enfoque no solo mejora la seguridad en la intersección, sino que también ofrece un entorno de aprendizaje práctico y dinámico para explorar conceptos de programación y diseño de sistemas embebidos.

2 Desarrollo de la Práctica

2.1 Materiales

Los materiales utilizados en la práctica son los siguientes:

- Arduino Uno
- Tres LEDs (verde, ámbar, rojo)
- Resistencias de 330 ohms
- Botón (push button)
- Protoboard
- Cables de conexión

2.2 Configuración inicial de pines

Primero, se definieron los pines de salida para los LEDs y el pin de entrada para el botón. Esto se realizó en la función `setup()`, que se ejecuta una sola vez al inicio del programa. A continuación, se muestra el fragmento de código correspondiente:

```
void setup() {
    pinMode(LED_VERDE, OUTPUT);
    pinMode(LED_AMBAR, OUTPUT);
    pinMode(LED_ROJO, OUTPUT);
    pinMode(BOTON, INPUT);
}
```

Listing 1: Configuración de pines en el setup()

En este código se declaran los pines de los LEDs como salidas (`OUTPUT`) y el botón como entrada (`INPUT`). Esta configuración permite controlar el encendido y apagado de los LEDs y leer el estado del botón.

2.3 Lógica del semáforo

El ciclo del semáforo se controla mediante un bucle `loop()`, que se repite indefinidamente. El semáforo pasa por las fases de verde, ámbar y rojo, con la duración específica para cada LED. El siguiente fragmento de código muestra cómo se implementó esta lógica:

```
void loop() {
    digitalWrite(LED_VERDE, HIGH);
    delay(3000); // Verde encendido por 3 segundos
    digitalWrite(LED_VERDE, LOW);

    digitalWrite(LED_AMBAR, HIGH);
    delay(1000); //  mbar  encendido por 1 segundo
    digitalWrite(LED_AMBAR, LOW);

    digitalWrite(LED_ROJO, HIGH);
    delay(3000); // Rojo encendido por 3 segundos
    digitalWrite(LED_ROJO, LOW);
}
```

Listing 2: Lógica del ciclo de semáforo

En este fragmento, se utiliza la función `digitalWrite()` para encender o apagar los LEDs, y la función `delay()` para controlar el tiempo que cada LED permanece encendido. El LED verde se enciende por 3 segundos, el ámbar por 1 segundo, y el rojo por 3 segundos .

2.4 Interrupción del ciclo con el botón

Para añadir la funcionalidad del botón, se implementó una condición que verifica si el botón está presionado. Cuando se detecta que el botón está activado, el ciclo de semáforo se interrumpe y el LED rojo se enciende por 4 segundos. El código para esta funcionalidad es el siguiente:

```
if (digitalRead(BOTON) == HIGH) {
    digitalWrite(LED_ROJO, HIGH);
    delay(4000); // Rojo encendido por 4 segundos
    digitalWrite(LED_ROJO, LOW);
}
```

Listing 3: Lógica del botón para interrumpir el ciclo

En este caso, `digitalRead()` se utiliza para leer el estado del botón. Si el botón está presionado, se enciende el LED rojo por 4 segundos .

2.5 Integración del código

La integración de todo el código se basa en un ciclo continuo (`loop()`) que controla el flujo de trabajo del semáforo. El ciclo básico del semáforo es gestionado a través de las funciones `digitalWrite()` y `delay()`, que establecen los tiempos durante los cuales los LEDs están encendidos. Por otro lado, el botón actúa como una interrupción externa que modifica el comportamiento del semáforo cuando es presionado, activando el LED rojo independientemente del estado actual del ciclo.

```
// Definici n de pines
const int ledVerde = 8;     // Pin del LED verde
const int ledAmbar = 9;      // Pin del LED  mbar
```

```
const int ledRojo = 10;      // Pin del LED rojo
const int boton = 7;         // Pin del bot n

// Variables de tiempo
unsigned long tiempoAnterior = 0;
unsigned long intervaloVerde = 3000;  // 3 segundos en milisegundos
unsigned long intervaloAmbar = 1000;  // 1 segundo en milisegundos
unsigned long intervaloRojo = 3000;   // 3 segundos en milisegundos
unsigned long intervaloRojoBoton = 4000; // 4 segundos en milisegundos

// Variables para el control del estado del sem foro
int estadoActual = 0;         // 0: Verde, 1:  mbar , 2: Rojo
bool botonPresionado = false;
unsigned long tiempoInicioBoton = 0;

void setup() {
  // Configuraci n de los pines como salidas
  pinMode(ledVerde, OUTPUT);
  pinMode(ledAmbar, OUTPUT);
  pinMode(ledRojo, OUTPUT);
  pinMode(boton, INPUT_PULLUP); // Bot n con resistencia pull-up
   interna

  // Inicializaci n de LEDs (apagados al inicio)
  digitalWrite(ledVerde, LOW);
  digitalWrite(ledAmbar, LOW);
  digitalWrite(ledRojo, LOW);
}

void loop() {
  // Verificar si el bot n fue presionado
  if (digitalRead(boton) == LOW && !botonPresionado) {
    botonPresionado = true;  // Marcamos que el bot n fue presionado
    tiempoInicioBoton = millis();  // Guardamos el tiempo de inicio
  }

  if (botonPresionado) {
    // Si el bot n fue presionado, el LED rojo se enciende durante 4
   segundos
    digitalWrite(ledVerde, LOW);
    digitalWrite(ledAmbar, LOW);
    digitalWrite(ledRojo, HIGH);

    // Verificar si han pasado los 4 segundos
    if (millis() - tiempoInicioBoton >= intervaloRojoBoton) {
      botonPresionado = false;  // Termina el modo del bot n y vuelve
   al ciclo normal
      tiempoAnterior = millis(); // Reiniciamos el tiempo del sem foro
    normal
      estadoActual = 0;          // Reiniciamos al estado verde
    }
  } else {
    // Control normal del sem foro (si el bot n no fue presionado)
    unsigned long tiempoActual = millis();

    switch (estadoActual) {
      case 0:  // Estado verde
        digitalWrite(ledVerde, HIGH);
```

```
58          digitalWrite(ledAmbar, LOW);
59          digitalWrite(ledRojo, LOW);
60          if (tiempoActual - tiempoAnterior >= intervaloVerde) {
61            estadoActual = 1;  // Cambiamos al estado  mbar
62            tiempoAnterior = tiempoActual;
63          }
64          break;
65
66        case 1:  // Estado  mbar
67          digitalWrite(ledVerde, LOW);
68          digitalWrite(ledAmbar, HIGH);
69          digitalWrite(ledRojo, LOW);
70          if (tiempoActual - tiempoAnterior >= intervaloAmbar) {
71            estadoActual = 2;  // Cambiamos al estado rojo
72            tiempoAnterior = tiempoActual;
73          }
74          break;
75
76        case 2:  // Estado rojo
77          digitalWrite(ledVerde, LOW);
78          digitalWrite(ledAmbar, LOW);
79          digitalWrite(ledRojo, HIGH);
80          if (tiempoActual - tiempoAnterior >= intervaloRojo) {
81            estadoActual = 0;  // Regresamos al estado verde
82            tiempoAnterior = tiempoActual;
83          }
84          break;
85      }
86    }
87  }
```

Listing 4: Código integrado

La estructura completa del código permite que el sistema sea flexible y manejable. Al estar basado en condiciones lógicas sencillas (`if-else`), el programa es capaz de monitorear continuamente el estado del botón mientras mantiene el ciclo del semáforo funcionando de manera ininterrumpida.

Este tipo de arquitectura de control se puede clasificar como un *loop polling* debido a que el programa verifica continuamente el estado del botón en cada iteración del bucle principal .

El desarrollo de la práctica comenzó con la conexión de los componentes al Arduino y la configuración de los pines en el código. El diagrama del circuito se muestra a continuación:

2.6 Explicación de las conexiones

El circuito fue conectado de la siguiente manera:

- **LED Verde, Ámbar y Rojo**: Cada uno de los LEDs está conectado a una salida digital del Arduino (pines 2, 3 y 4) a través de una resistencia de 220 ohms para limitar la corriente y evitar dañar los LEDs.

- **Botón**: El botón fue conectado al pin digital 7 del Arduino. Utilizamos un *pull-down resistor* para asegurarnos de que el valor leído sea LOW cuando el botón no esté presionado. Esto evita lecturas falsas debidas a ruido eléctrico.

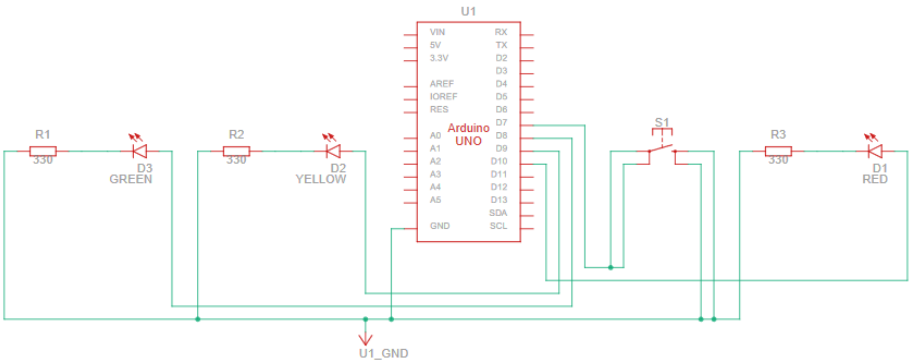

Figure 1: Diagrama del circuito del semáforo con Arduino.

2.7 Simulación

La simulación del circuito fue realizada en la plataforma **Tinkercad**, un entorno de simulación en línea que permite construir y ejecutar circuitos electrónicos de manera virtual. En la simulación, el ciclo del semáforo funciona exactamente como en la implementación física. Los LEDs se encienden en la secuencia adecuada, y al presionar el botón, el semáforo cambia inmediatamente a rojo por 4 segundos, deteniendo el ciclo temporalmente.

Tinkercad permite ejecutar el código directamente en el simulador y observar el comportamiento de los componentes electrónicos conectados. A continuación, se presentan dos imágenes tomadas durante la práctica real, donde se muestra el funcionamiento del semáforo:

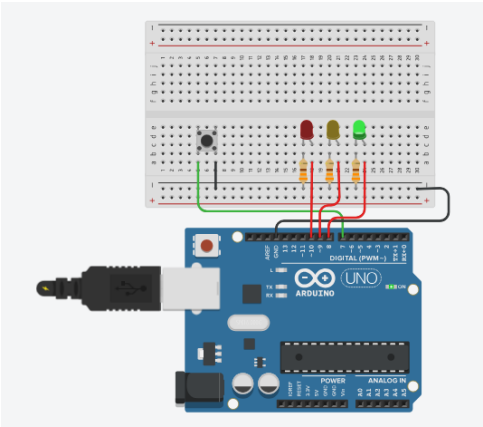

Figure 2: Funcionamiento normal del semáforo.

En esta primera imagen, se observa el ciclo regular del semáforo, con el LED verde encendido.

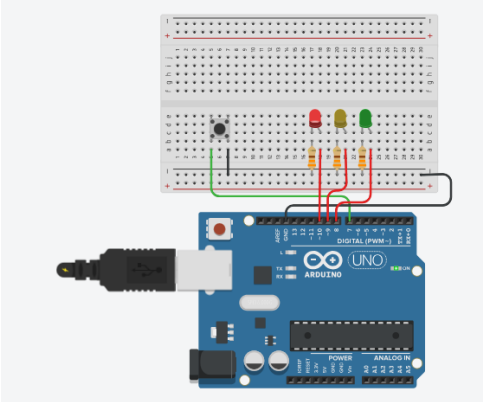

Figure 3: Semáforo en rojo cuando se presiona el botón.

En la segunda imagen, el botón ha sido presionado, activando el LED rojo por 4 segundos.

2.8 Implementación

2.8.1 Implementación del semaforo

El sistema de semáforo se compone de tres LEDs: uno verde, uno ámbar y uno rojo, cada uno conectado a una salida digital del microcontrolador Arduino. La funcionalidad del semáforo simula el ciclo de un semáforo real, donde cada color indica un estado específico del tráfico. La luz verde se activa durante 30 segundos, permitiendo el paso de vehículos; luego, la luz ámbar se enciende durante 30 segundo, advirtiendo a los conductores que deben detenerse; finalmente, la luz roja se activa durante 30 segundos, deteniendo el tráfico. Este ciclo se repite continuamente, gestionando de manera eficiente el flujo vehicular.

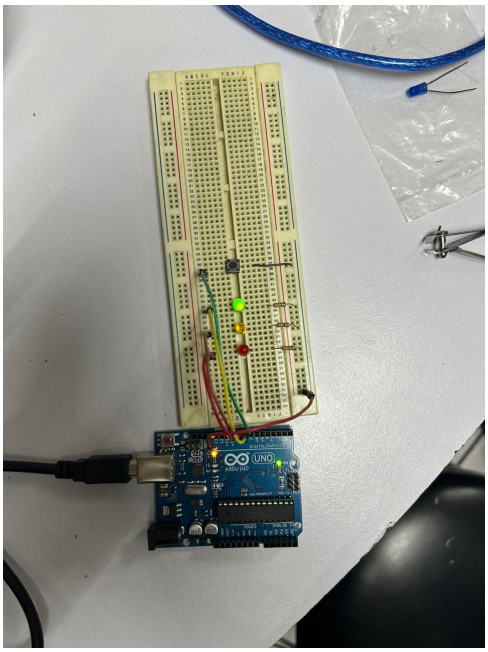

Figure 4: Funcionamiento habitual del semáforo

Cada LED está conectado a un resistor limitador de corriente para evitar daños al componente. La lógica de programación del Arduino gestiona el tiempo que cada LED debe estar encendido, utilizando la función delay() para controlar los intervalos. El diagrama del circuito muestra cómo se conectan los LEDs a los pines digitales del Arduino, y cómo la alimentación es proporcionada adecuadamente a cada componente para asegurar su funcionamiento.

2.8.2 Funcionamiento del Botón

Además de los LEDs, se incorpora un botón de presión conectado a un pin digital del Arduino, que permite interrumpir el ciclo normal del semáforo. Cuando el botón es presionado, se activa una función que pone inmediatamente el semáforo en estado rojo durante 40 segundos, independientemente del ciclo actual.

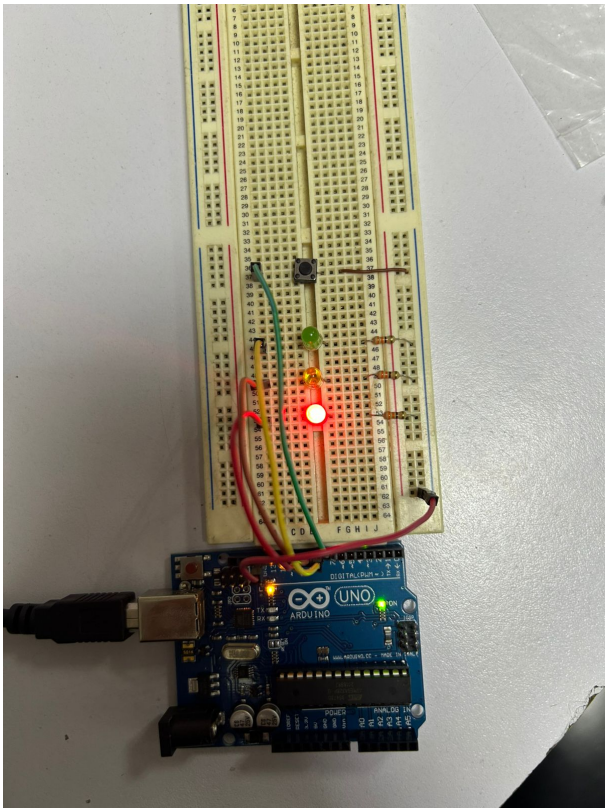

Figure 5: Funcionamiento con boton presionado

El botón está conectado de manera que, al ser presionado, se envía una señal al Arduino para cambiar el estado del semáforo. La programación incluye un manejo de interrupciones, permitiendo que la presión del botón sea detectada en cualquier momento, lo que garantiza que el semáforo responda rápidamente a la acción del usuario. Este aspecto de la implementación resalta la capacidad del sistema para adaptarse a situaciones imprevistas y mejorar la seguridad en la intersección.

3 Conclusiones

3.1 Luciano Hernández Jonathan

Este proyecto permitió dominar el uso de entradas y salidas digitales mediante la plataforma Arduino, especialmente en la configuración de componentes como LEDs y botones. A través de la programación en lenguaje arduino, fue posible implementar un ciclo de semáforo básico que, al integrar el botón como interrupción, añade mayor flexibilidad y control sobre el flujo del sistema. Este conocimiento es aplicable en proyectos de control secuencial y automatización en sistemas electrónicos reales.

3.2 Rodriguez Morales Iñaki

El desarrollo y la implementación del sistema de semáforo utilizando Arduino han demostrado ser una experiencia enriquecedora en la comprensión de la programación de microcontroladores y el manejo de dispositivos electrónicos. A través de este proyecto, se exploraron conceptos fundamentales como la gestión de entradas y salidas digitales, así como el diseño de un ciclo de control secuencial que simula el funcionamiento real de un semáforo.

References

- J. Boxall, Arduino Workshop: A Hands-On Introduction with 65 Projects, San Francisco, CA, USA: No Starch Press, 2013.
- M. Banzi and M. Shiloh, Getting Started with Arduino, 3rd ed., Sebastopol, CA, USA: O'Reilly Media, 2014.
- F. Adams, "Arduino Traffic Light Project with Push Button Control," Circuit Digest, Jun. 15, 2018. [Online]. Available: https://circuitdigest.com/microcontroller-projects/arduino-traffic-light-project-with-push-button-control. [Accessed: 08-Oct-2024].
- P. J. Hamill, Practical Electronics for Inventors, 4th ed., New York, NY, USA: McGraw-Hill, 2016.
- Autodesk, "Simulating Circuits with Tinkercad," Autodesk Tinkercad. [Online]. Available: https://www.tinkercad.com/. [Accessed: 08-Oct-2024].