Universidad La Salle

Seminario en Tecnología I

Informe

Proyecto Semáforo

Proyecto Nº 1

Integrantes:

Karlo Pacha Curimayhua

Octavo Semestre - Ingeniería de Software

15 de abril del 2024

1. Marco Teórico

En este proyecto específico, se están construyendo dos semáforos, los cuales deben estar sincronizados, es decir, no deben tener el mismo color a la vez. Para este proyecto se utilizaron LEDs, resistencias, fotorresistencias, cables puente, un Arduino, una pantalla LCD y un protoboard.

Los **LEDs** son dispositivos semiconductores que emiten luz cuando se les aplica una corriente eléctrica. Son ampliamente utilizados en una variedad de aplicaciones debido a su eficiencia energética, vida útil prolongada y capacidad para emitir luz en una gama de colores **[1]**. En este proyecto se usan para las luces del semáforo.

Las **resistencias** son componentes pasivos que limitan la cantidad de corriente que fluye a través de un circuito **[2]**. En este proyecto, las resistencias se utilizan para limitar la corriente que fluye a través de los LEDs, protegiéndolos de recibir demasiada corriente.

Las **fotoresistencias** son componentes pasivos que disminuyen su resistencia a medida que aumenta la luminosidad (luz) en su superficie sensible **[3]**. En este proyecto, las fotoresistencias se usan como si fuese una resistencia dada la falta de resistencias, estas se usaron para proteger uno de los LEDs y a la pantalla LCD.

Los **cables puente** son cables eléctricos con pines conectores en cada extremo, que se utilizan para conectar dos puntos en un circuito sin soldadura **[4]**. En este proyecto, los cables puente se utilizan para conectar el Arduino con el protoboard.

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo (software), diseñada para facilitar el uso de la electrónica en proyectos multidisciplinarios **[5]**. En este proyecto se usó el Arduino UNO; el Arduino se utiliza como el cerebro del semáforo, controlando cuándo se encienden y apagan los LEDs, y qué texto muestra la pantalla LCD.

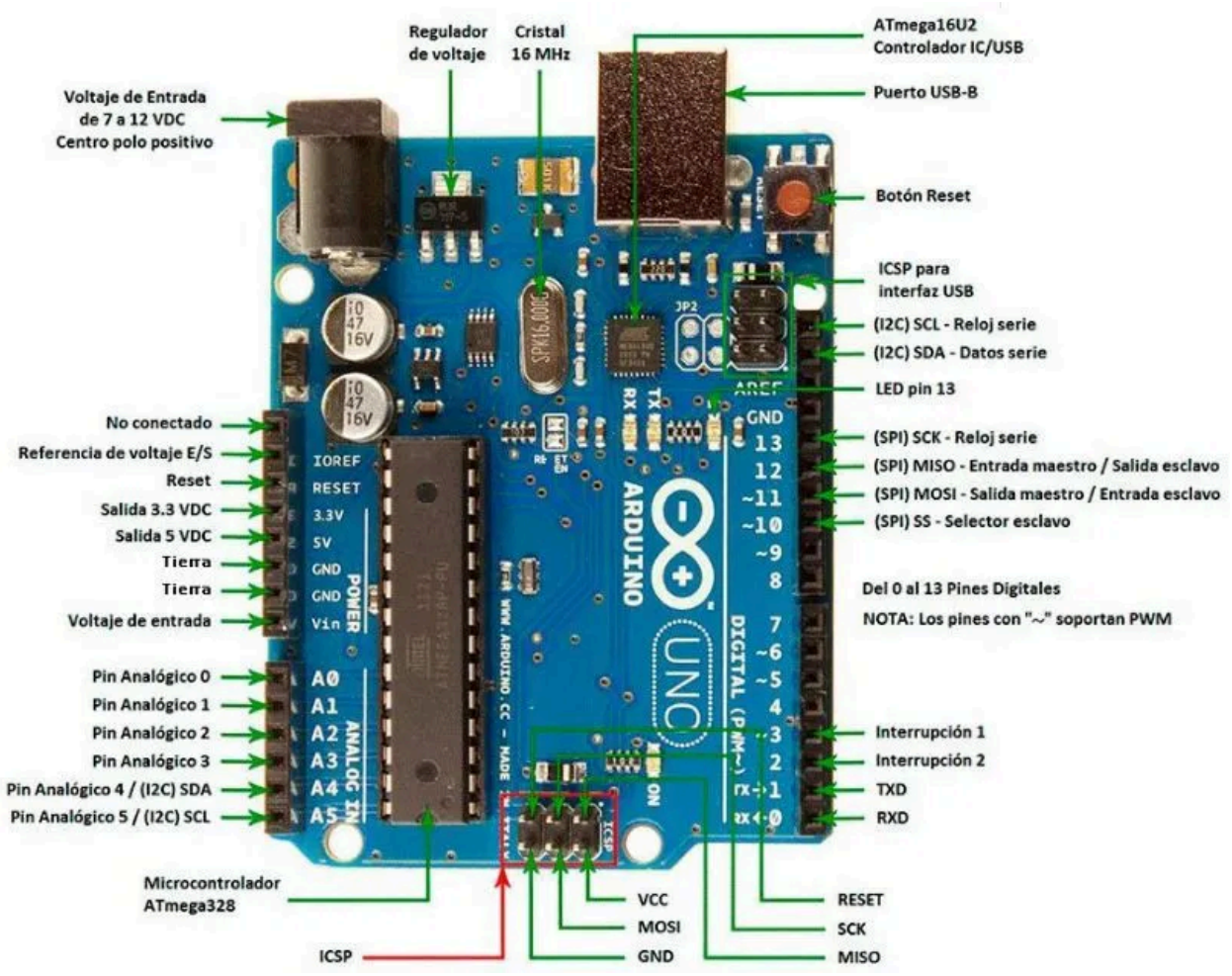


Imagen 1: Partes del Arduino UNO **[9]**.

Una **pantalla LCD** es un tipo de pantalla que utiliza la luz modulada por cristales líquidos para producir imágenes **[6]**. En este proyecto, la pantalla LCD se usa para mostrar el color actual de cada semáforo.

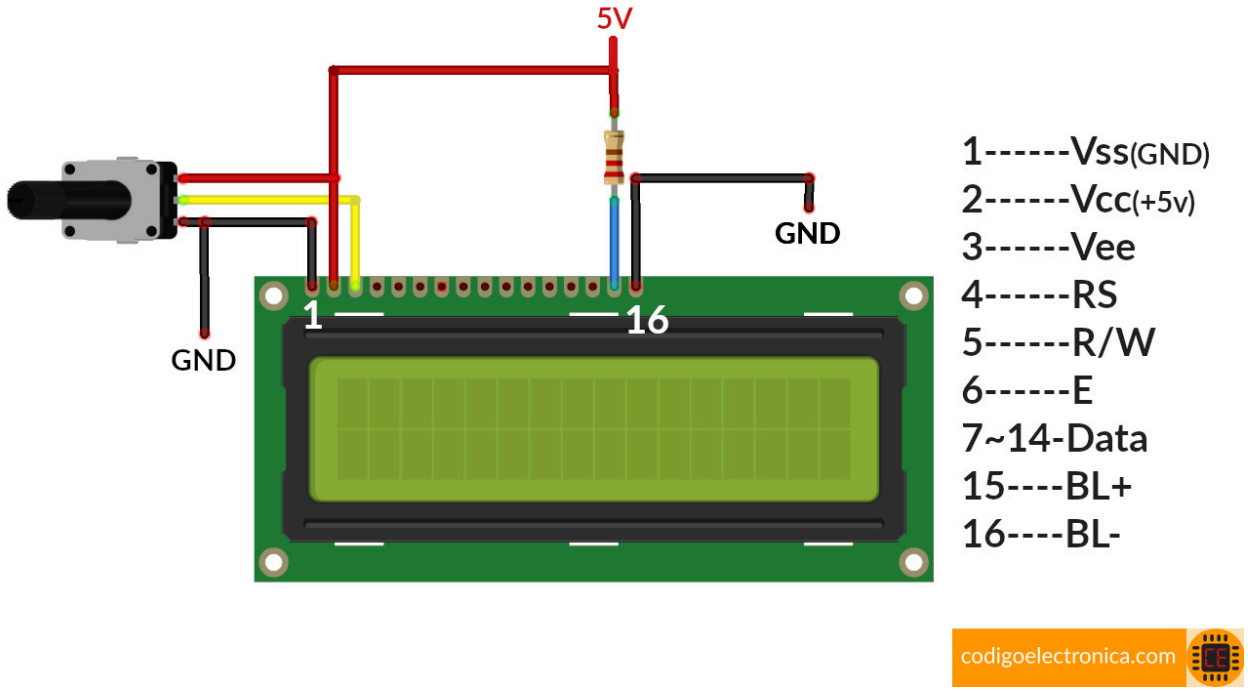


Imagen 2: Partes de la pantalla LCD 2x16 en Arduino **[10]**.

El término 2x16 hace referencia a las 2 filas de texto que puede soportar, y a los 16 caracteres que puede tener por fila **[10]**.

Un **protoboard** es una placa de pruebas que se utiliza para construir y probar circuitos electrónicos de manera temporal antes de soldarlos de forma permanente **[7]**. En este proyecto, el protoboard se utiliza para conectar todos los componentes del semáforo.

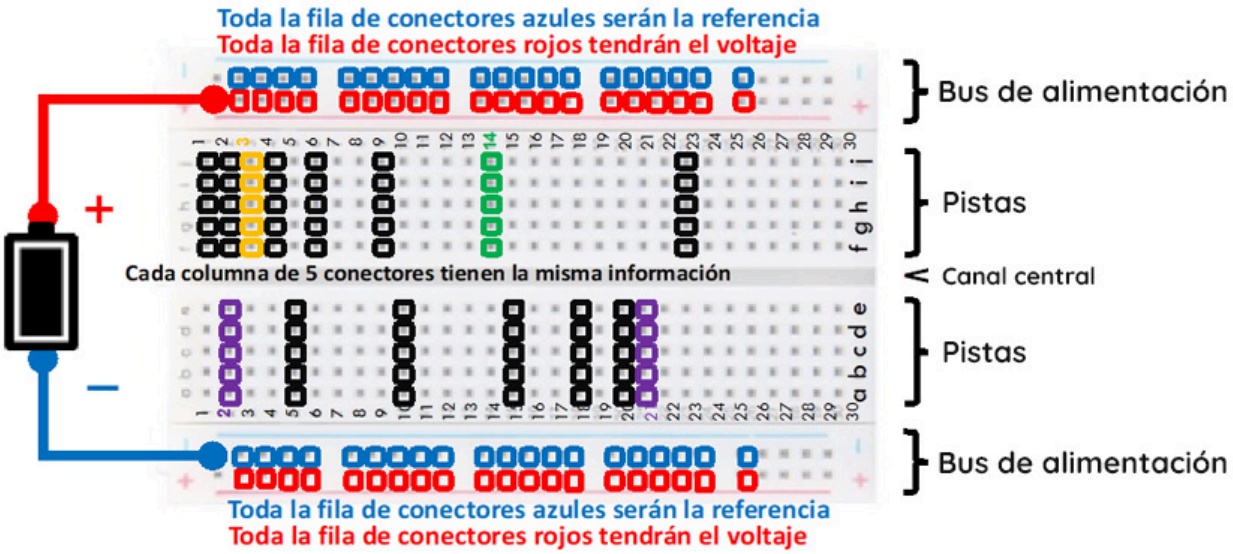


Imagen 2: Partes de un protoboard **[8]**.

Los LEDs, las resistencias y las fotorresistencias están conectados al protoboard y los cables puente conectan el protoboard al Arduino, a la pantalla LCD al Arduino y al protoboard. Los LEDs actúan como las luces del semáforo, cambiando de color en respuesta a la programación del Arduino. Las resistencias y las fotorresistencias se utilizan para limitar la corriente a los LEDs y evitar que se quemen, lo mismo con la pantalla LCD, la cual muestra el color actual del semáforo con letras. El Arduino, programado a través de su entorno de desarrollo, controla el comportamiento de los semáforos y la pantalla LCD.

2. Implementación del proyecto

2.1. Primera versión

Como requerimiento inicial, solo se tuvo en cuenta la implementación un semáforo con Arduino. Tampoco se especificó la secuencia de las luces, el tiempo de duración de cada estado del semáforo. Con estos requerimientos se optó por hacer un semáforo *normal*.

Dado el poco conocimiento en electrónica que posee el equipo, se decidió hacer el diseño en la plataforma Tinkercad, el cual da a disposición un entorno de simulación para el uso de Arduino y otros componentes electrónicos. En el entorno se usaron los componentes físicos disponibles como:

- 3 LEDs
- 3 resistencias
- 4 cables puente
- 1 Arduino
- 1 Protoboard

Se empezó colocando los LEDs en las pistas o nodos del protoboard, pero sin conectarlas a los buses. Las resistencias se colocaron en los nodos, pero conectándolas también con los buses negativos.

Los cables puente se usaron para conectar los nodos en la columna positiva en la que se encuentran los LEDs, y los pines digitales de Arduino. Se usaron los pines 13, 12 y 11. Además, se conectó el GND de Arduino con el bus negativo del protoboard.

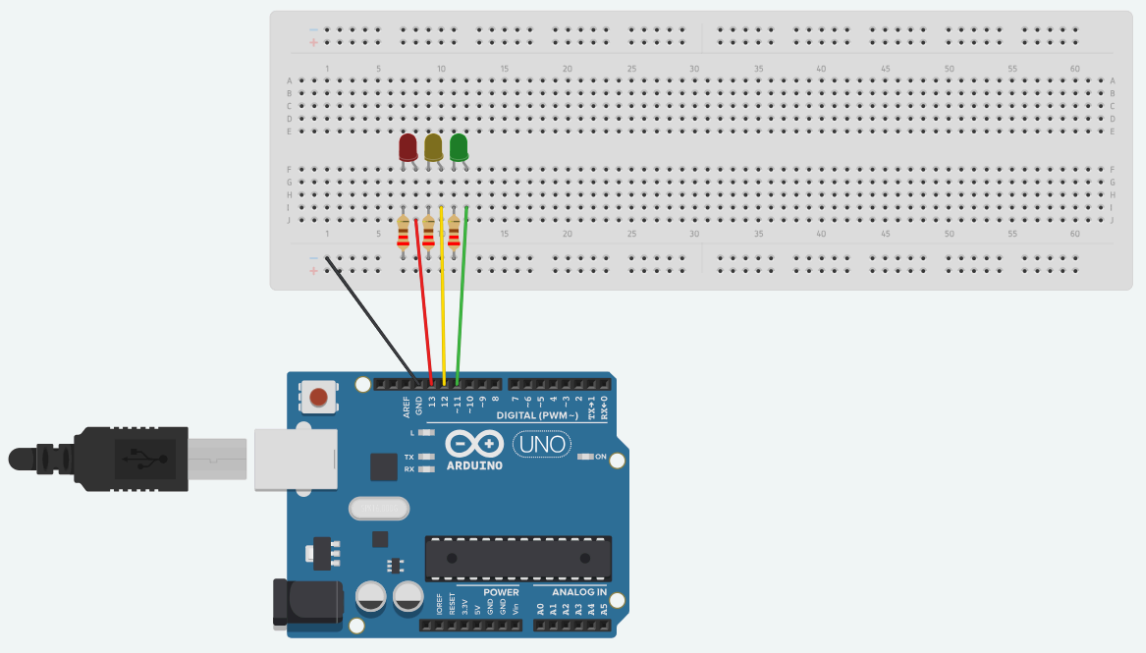


Imagen 3: Diseño de la primera versión del semáforo en Tinkercad.

2.2. Segunda Versión

En esta segunda versión se tuvieron los requerimientos completos, la secuencia se especificó, aunque el tiempo de duración de los estados del semáforo quedaron a libre especificación.

Se decidió seguir el diseño en la plataforma Tinkercad y se usaron los siguientes componentes:

- 6 LEDs
- 6 resistencias, en la implementación real se usó una fotorresistencia como sustituto de una resistencia.
- 7 cables puente
- 1 Arduino
- 1 Protoboard

Se empezó colocando más LEDs, resistencias y una fotorresistencia (en la implementación física), igual que en la versión 1, pero en un conjunto de nodos más separados. Se añadieron más cables puente, se usaron los pines 13, 12, 11, 7, 6, y 5. Además, se mantuvo conectado el GND de Arduino con el bus negativo del protoboard.

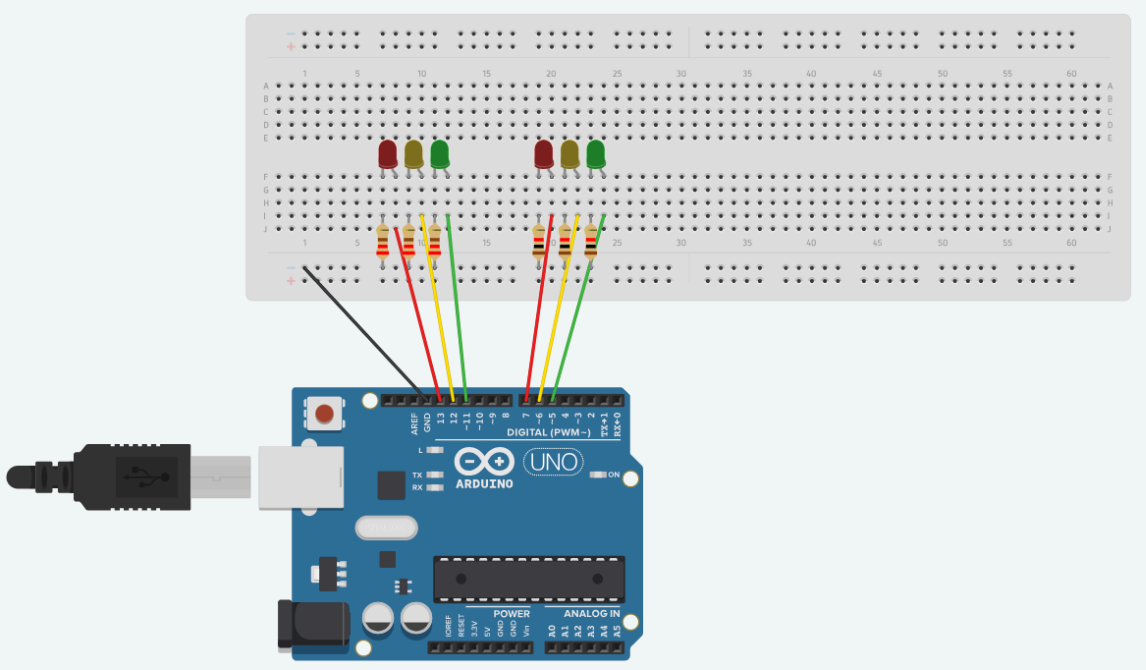


Imagen 4: Diseño de la segunda versión del semáforo en Tinkercad.

2.3. Tercera versión

Para esta versión se decidió añadir al diseño e implementación el uso de una pantalla LCD 2x16. Para lograr esto se siguió el tutorial "*Simulando un LCD en Tinkercad con Arduino*".

Se siguió usando Tinkercad y se usaron los siguientes componentes:

- 6 LEDs
- 7 resistencias, en la implementación real se usaron 2 fotorresistencias como sustituto de 2 resistencias.
- 20 cables puente
- 1 Arduino
- 1 Protoboard
- 1 pantalla LCD 2x16

Se mantuvo el diseño e implementación de la versión 2. Se colocó una fotorresistencia (en la implementación física) para el pin *BLA* o *LED* (encargado de la alimentación de la luz de fondo) **[10]**; este pin, junto al pin *VDD* o *VCC* (alimentación del LCD), se alimentó conectándolo al bus positivo. El bus positivo se conectó con un cable puente a la salida de 5v del Arduino.

Del pin *D7* al *D4* se conectaron al los pines 7 al 4 respectivamente. El pin *E* (sincronización de lectura de datos) y el *RS* (selector entre comando y datos), se conectaron al pin 3 y 2 respectivamente.

Para el *GND* (negativo), *VO* (regulador del contraste), *RW* (lectura y escritura de datos) y *BLK* (*GND* de la luz de fondo), se usó un cable puente conectado al bus negativo que ya se estaba usando en el diseño de la versión 2.

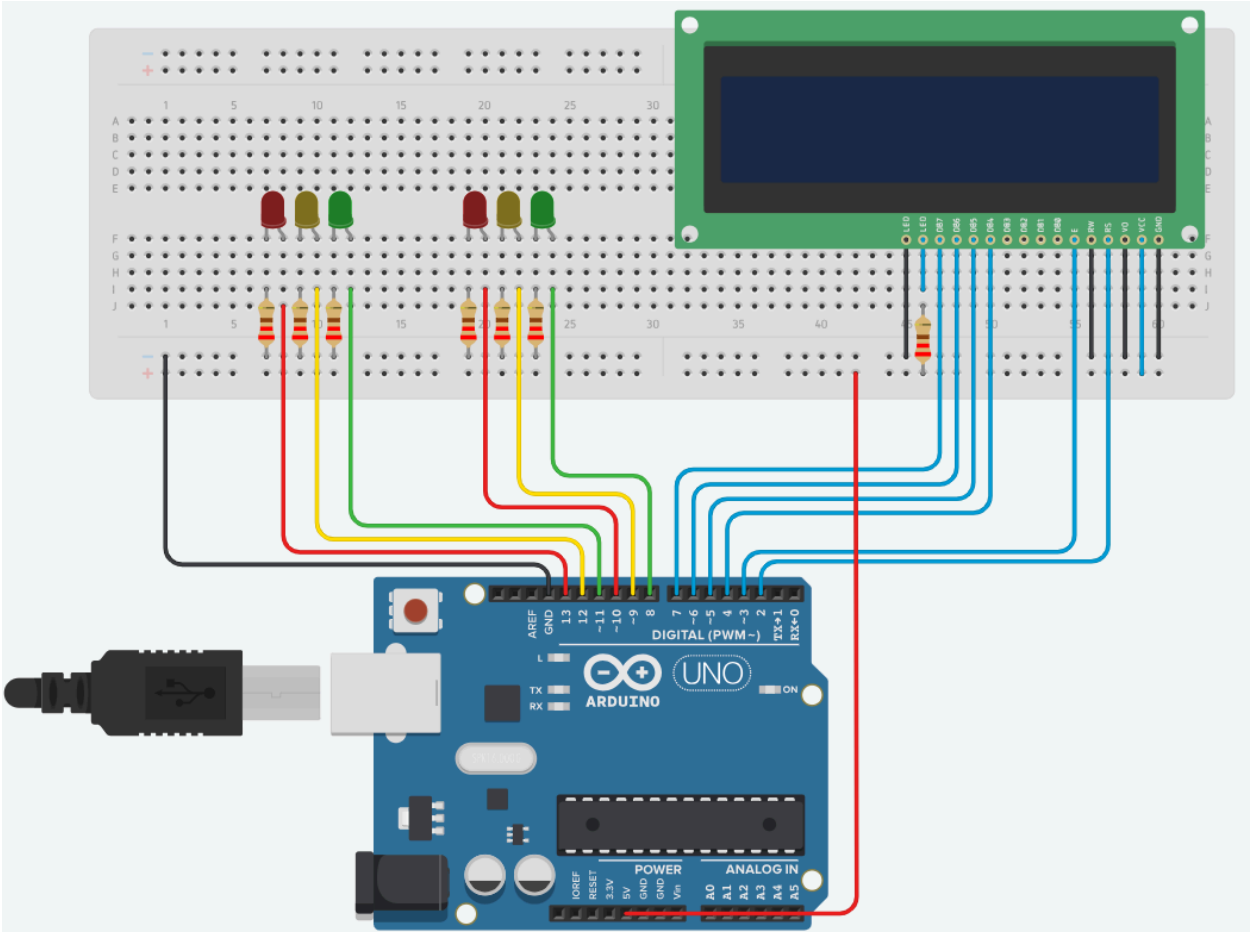
- 

Imagen 5: Diseño de la tercera versión del semáforo en Tinkercad.

3. Programación

Por motivos de comodidad y una mejor comprensión del código se decidió trabajar en Visual Studio Code con una extensión de Arduino, aunque para la compilación y envío del código al Arduino se usó el IDE oficial de Arduino.

3.1. Primera versión

```
int red  = 13;
int yellow = 12;
int green = 11;

int const SECOND = 1000;
```

Las variables red, yellow y green hacen referencia a los pines del Arduino que usa cada color. La variable constante SECOND se usa para el delay o tiempo que estará prendido cada luz del semáforo.

```
void setup()
{
  pinMode(red, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(green, OUTPUT);
}
```

Se usa la función ponMode nativa de Arduino para indicar los pines que se usarán.

```
void loop()
{
  digitalWrite(red, HIGH);
  delay(7.5*SECOND);
  digitalWrite(red, LOW);

  digitalWrite(yellow, HIGH);
  delay(2*SECOND);
  digitalWrite(yellow, LOW);

  digitalWrite(green, HIGH);
  delay(5*SECOND);
  digitalWrite(green, LOW);
}
```

La funcioes digitalWrite y delay, nativas de Arduino se usan para prender o apagar los LEDs de lsemáforo, y para establecer un tiempo de pausa respectivamente. Para el color rojo se establecieron 7.5 segundos de espera, para el verde 5 segundos, y para el amarillo solo 2 segundos.

3.2. Segunda versión

```
#define SECOND 1000
```

Se define una constante llamada SECOND que representa un segundo en milisegundos.

```
class Semaphore
{
  private:
    int RedPin, GreenPin, YellowPin , State, Id;
    unsigned long LastTime;
```

Esta clase representa un semáforo. Cada objeto de esta clase tiene cinco variables privadas: RedPin, YellowPin, GreenPin, State, Id, y LastTime. Estas variables representan los pines de los LEDs rojo, amarillo y verde, el estado actual del semáforo, el identificador del semáforo y la última vez que el estado del semáforo fue actualizado, respectivamente.

```
    void UpdateLights()
    {
      digitalWrite(RedPin, State == 0 ? HIGH : LOW);
      digitalWrite(GreenPin, State == 1 ? HIGH : LOW);
      digitalWrite(YellowPin, State == 2 ? HIGH : LOW);
      Print();
    }
```

Este método privado se utiliza para actualizar el estado de los LEDs basado en el estado actual del semáforo.

```
  public:
    Semaphore(int id, int redPin, int yellowPin, int greenPin, int
initialState)
    {
      Id = id;
      RedPin = redPin;
      YellowPin = yellowPin;
      GreenPin = greenPin;
      State = initialState;
      LastTime = millis();

      pinMode(RedPin, OUTPUT);
      pinMode(YellowPin, OUTPUT);
      pinMode(GreenPin, OUTPUT);

      UpdateLights();
    }
```

Este método constructor público se utiliza para inicializar un objeto de la clase Semaphore. Configura los pines de los LEDs como salidas y actualiza las luces.

```
    void UpdateState()
    {
      unsigned long currentTime = millis();

      if(
          (State == 0 && currentTime - LastTime >= 7*SECOND) ||
          (State == 1 && currentTime - LastTime >= 5*SECOND) ||
          (State == 2 && currentTime - LastTime >= 2*SECOND)
        )
      {
        State = (State + 1) % 3;
        LastTime = currentTime;
        UpdateLights();
      }
    }
};
```

Este método público se utiliza para actualizar el estado del semáforo basado en el tiempo transcurrido desde la última actualización. Además, convoca al Método UpdateLights si se ya ha transcurrido el tiempo especificado para que el semáforo cambie de color (5 segundos para el rojo y verde, y 2 segundos para el amarillo).

```
Semaphore* Semaphores[2];
```

Este es un array de punteros a objetos de la clase Semaphore. Se utiliza para almacenar los 2 semáforos que serán controlados por el programa.

```
void setup()
{
  StartingLCD();

  Semaphores[0] = new Semaphore(0, 13, 12, 11, 0);
  Semaphores[1] = new Semaphore(1, 10, 9, 8, 1);
}
```

Se utiliza la funcción setUp nativa de Arduino para inicializar los semáforos, se les pasa los parámetros. En el caso del Semaphore 1, utilizara los pines 13, 12 y 11 para los LEDs, y comenzará con el LED rojo prendido. En el caso del Semaphore 2, utilizará los pines 7,6 y 5 para los LEDs, y comenzará con el LED verde prendido.

```
void loop()
{
  Semaphores[0]->UpdateState();
  Semaphores[1]->UpdateState();
}
```

Esta función se ejecuta continuamente después de que la función setup() ha terminado. Se utiliza para actualizar el estado de cada semáforo cada segundo.

3.3. Tercera versión

```
#include <LiquidCrystal.h>

#define SECOND 1000

LiquidCrystal lcd(2,3,4,5,6,7);

class Semaphore
{
  private:
      .
      .
      .
```

Se incluye la biblioteca LiquidCrystal, que permite controlar pantallas LCD. Luego se crea un objeto lcd de la clase LiquidCrystal. Los números representan los pines a los que está conectada la pantalla LCD.

```
    void Print()
    {
      char buffer[16];

      switch (State) {
        case 0:
          sprintf(buffer, "Sem %d: RED   ", Id);
```

```
          break;
        case 1:
          sprintf(buffer, "Sem %d: GREEN ", Id);
          break;
        case 2:
          sprintf(buffer, "Sem %d: YELLOW", Id);
          break;
        default:
          sprintf(buffer, "Sem %d: ERROR ", Id);
          break;
      }

      if(Id == 0) lcd.setCursor(0, 0);
      else lcd.setCursor(0, 1);

      Serial.println(buffer);
      lcd.print(buffer);
    }
```

Este método privado en la clase Semaphore se utiliza para imprimir el estado actual del semáforo en la pantalla LCD y en el monitor serial. Utiliza la función sprintf() para formatear una cadena que representa el estado del semáforo, y luego imprime esa cadena en la posición correcta de la pantalla LCD.

```
    void UpdateLights()
      .
      .
      .
      Print();
    }
```

En la función UpdateLights se añade al final una invocación a la función Print descrita anteriormente. Esto para que actualice la pantalla LCD.

```
void StartingLCD()
{
  lcd.begin(16, 2);

  lcd.setCursor(0, 0);
```

```
  lcd.print("Semaphore");

  lcd.setCursor(0, 1);
  lcd.print("Starting");
  delay(0.25*SECOND);

  for(int i = 0; i < 5; i++)
  {
    lcd.setCursor(0, 0);
    lcd.print("Semaphore");
    lcd.setCursor(0, 1);
    lcd.print("Starting.  ");
    delay(0.25*SECOND);

    lcd.setCursor(0, 0);
    lcd.print("Semaphore");
    lcd.setCursor(0, 1);
    lcd.print("Starting.. ");
    delay(0.25*SECOND);

    lcd.setCursor(0, 0);
    lcd.print("Semaphore");
    lcd.setCursor(0, 1);
    lcd.print("Starting...");
    delay(0.25*SECOND);

    lcd.clear();
  }

  lcd.clear();
}
```

La función se utiliza para inicializar la pantalla LCD. Primero, configura la pantalla LCD para que tenga 16 columnas y 2 filas con lcd.begin(16, 2);. Luego, imprime un mensaje de “Semaphore Starting” en la pantalla LCD, con una animación de puntos que aparecen y desaparecen.

```
Semaphore* Semaphores[2];
```

```
void setup()
{
  StartingLCD();
      .
      .
      .
```

En la función setup(), ahora se llama a la función StartingLCD() para inicializar la pantalla LCD antes de crear los objetos Semaphore.

4. Pruebas elaboradas y Datos obtenidos en las pruebas

4.1. Primera versión

En la primera versión se obtuvieron resultados satisfactorios tanto en Tinkercad como en la implementación física. Para probarlo, se cambió el tiempo de espera para cada foco LED, y aún así respondió de forma satisfactoria.

4.2. Segunda versión

En esta actualización, se intentó usar la base de la prima versión, pero no era apta para solucionar el problema, por ello se tuvo que crear clases en primera instancia dado a que debían estar sincronizados, y con las clases se tendría algo más ordenado. Para sincronizar en cierta medida a ambos emáforos se usó la función millis, con la cual se pudo obtener el tiempo en el que el estado actual (color de semaforo) estuvo prendido, y así se hizo el cálculo para cambiar entre estados.

En Tinkercad los resultados fueron confusos, si bien se respetaba la secuencia, al pareer la función millis no funcionaba del todo bien, ya que a pesar de usar código en duro para los tiempos, este entorno duplicaba o triplicada la duración del tiempo de cada estado del semáforo. Al llevarlo a la implementación física, se notó que si respetaba la duración de tiempos de cada estado. Dado esto, se decidió implementar el código en Visual Studio Code y ejecutar y subir el código con el IDE de Arduino.

4.3. Tercera versión

Se probó inicialmente con texto en código duro en la función setUp, todo funcionaba con normalidad ne cuando a código. Pero los problemas con el hardware comenzaron, la pantalla LCD parecía desconectarse del software con el más leve mínimo movimiento, para lo cual era necesario reiniciarlos o desconectarlo y volverlo a conectar de la PC.

Para probar el funcionamiento de la función millis, se empezó a mostrar el currenTime sustrayendo el valor de LastTime, inicialmente se obtuvieron resultados positivos, pero pasó los mismo que al inicio de esta actualización, la pantalla LCD se desconectaba del programa de Arduino y no mastaba nada o a veces mostraba carácteres en chino.

Después otrs pruebas insignificantes, se optó po sólo mostrar texto y dejarlo en la 3 sub-versión de esta segunda actualización. Inicialmente se muestra una pequeña animación de “Starting…”, para luewgo mostrar los colores en texto en el que se encuentra cada semáforo.

El problema con Tinkercard continuó, respetaba la secuencia, pero no los tiempos de espera de cada estado.

4.3.1. Videos

- https://drive.google.com/file/d/1R33c2jS0Gi7WC0yhQq-Fo8hpvBXNdGhY/view?usp=drive_link
- https://drive.google.com/file/d/1Qu7HXNbPoHrGUufskv1QAaSobgkd7uYg/view?usp=drive_link
- https://drive.google.com/file/d/1QnTC0b0pyvJPx5Yq5i5GLMvLfFJbDRXt/view?usp=drive_link

4.4. Tinkercad

4.4.1. Primera versión:
https://www.tinkercad.com/things/6Sm2mcx0cgQ-semaphore

4.4.2. Segunda versión:
https://www.tinkercad.com/things/kVShHSuFksn-semaphore2

4.4.3. Tercera versión:
https://www.tinkercad.com/things/lhYf7xkGQKW-semaphore3

5. Conclusiones y Recomendaciones

Con este proyecto se pudo evidenciar el enorme potencial de Arduino, con simples códigos se pueden hacer cosas en minutos que de forma convencional con un PLC u otro componente electrónico. Su flexibilidad permite que se adhieran o se remuevan funcionalidades o componentes electrónicos, lo cual es clave para la exploración y desarrollo de aplicaciones de hardware-software, o incluso Internet de las Cosas.

Si bien Arduino es capaz de ejecutar métodos de seguridad para no sufrir una sobrecarga de energía y tampoco los componentes conectados, es importante documentarse un poco para no sufrir accidentes. En cuanto a la programación, es importante optimizar el código, hacer uso de métodos debug o entornos simulados como Tinkercad para no tener problemas con un posible mal diseño en el hardware.

6. Bibliografía

[1] *“A complete overview of LEDs: from theory to applications,” PLAY Embedded, Abr. 14, 2023. https://www.playembedded.org/blog/complete-overview-leds/ (accedido Abr. 15, 2024).*

[2] *“Theory Overview,” Engineering LibreTexts, Jun. 23, 2020. https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Electronics/Laboratory_Ma*

nual_-_DC_Electrical_Circuit_Analysis_%28Fiore%29/03%3A_Resistor_Color_Code/3.1%3A_Theory_Overview (accedido Abr. 15, 2024).

[3] *"Photoresistor," Wikipedia, May 31, 2020. https://en.wikipedia.org/wiki/Photoresistor (accedido Abr. 15, 2024).*

[4] *Wiltronics, "What Are Jumper Wires: Know by Colour, Types and Uses," Wiltronics, Abr. 14, 2022. https://www.wiltronics.com.au/wiltronics-knowledge-base/what-are-jumper-wires/ (accedido Abr. 15, 2024).*

[5] *Arduino, "What Is Arduino?," Arduino.cc, Feb. 05, 2018. https://www.arduino.cc/en/guide/introduction (accedido Abr. 15, 2024).*

[6] *Wikipedia Contributors, "Liquid-crystal display," Wikipedia, Feb. 14, 2019. https://en.wikipedia.org/wiki/Liquid-crystal_display (accedido Abr. 15, 2024).*

[7] *"Breadboard," Wikipedia, Jun. 08, 2020. https://en.wikipedia.org/wiki/Breadboard (accedido Abr. 15, 2024).*

[8] *"Tableta protoboard," Portal Académico del CCH. https://portalacademico.cch.unam.mx/cibernetica1/implementacion-de-circuitos-logicos/tableta-protoboard (accedido Abr. 15, 2024).*

[9] *"Arduino UNO | Arduino.cl - Compra tu Arduino en Línea." https://arduino.cl/producto/arduino-uno/ (accedido Abr. 15, 2024).*

[10] *O. M. F. Alzate, "Arduino lcd 2x16," http://codigoelectronica.com, Oct. 08, 2015. http://codigoelectronica.com/blog/arduino-lcd-2x16 (accedido Abr. 15, 2024).*

[11] *"Simulando un LCD en Tinkercad con Arduino," www.youtube.com. https://www.youtube.com/watch?v=6QzIVh6upyQ (accedido Abr. 13, 2024).*