Informa2 S.A.S – Parcial 1

Angie Tatiana Solano Rodriguez

Universidad de Antioquia

Faculta de Ingeniería

Ingeniería de Telecomunicaciones

Medellín, Antioquia, Colombia

2023

Tabla de contenido

Análisis y alternativas de solución al problema

Descripción del problema

Se nos menciona un sistema compuesto de 64 leds que, al finalizar el desarrollo del proyecto, nos permita mostrar diferentes patrones según lo ingresado por el usuario, aparte de esto debemos tener en cuenta que tenemos un límite de pines digitales de Arduino que podemos utilizar, es decir que necesitamos hallar alguna alternativa que nos permita reducir la cantidad de pines usados.

Análisis del problema.

Requerimientos o restricciones.

1. Límite de pines digitales: El sistema este compuesto por 64 leds, por lo tanto, para ser conectados en el Arduino de Tinkercad, superaríamos el límite interpuesto que es de 7 pines digitales.
2. Control independiente de los leds: Debido a que hay ejercicios donde se deben mostrar diferentes patrones, es necesario tener en cuenta que necesitaríamos el control independiente de los leds.
3. Facilidad de entendimiento y uso del sistema: Al tener conocimiento básico en el uso de la plataforma y en el dispositivo que utilizamos allí mismo, es importante tener lo más posible el desarrollo del proyecto.
4. Limitaciones Tinkercad: Consulta información respecto a ejercicios o proyectos similares, podemos observar que existen diversas maneras de solucionarlo, pero al ser una plataforma de simulación presenta limitaciones en componentes.

Aparte de ello, podemos realizar un tipo de boceto o plano de como pensamos hacer las conexiones en la plataforma, para una mayor facilidad al momento de hacer la conexión en el simulador o para desarrollar los diversos problemas presentados.

Alternativas de solución

Al consultar o buscar alternativas para la solución del problema inicial, hallamos estas diferentes opciones:

Multiplexores

Este permite seleccionar una de sus varias entradas de datos y llevar esto a la única salida. Aparte de ello, este permite la reducción del cableado o en su conclusión la disminución en el uso de pines digitales en el Arduino.

Integrado 74HC595

Este también nos permite realizar la conexión de los diferentes LEDS del sistema que necesitas realizar, este tiene un desplazamiento de 8 bits con salida paralela, investigando permite controlar múltiples LEDS y tienes ventajas como:

1. Ahorros de pines de control
2. Control independiente
3. Capacidad de cadena, es decir podemos conectar varios integrados

El uso de este nos permitiría simplificar y optimizar el control múltiple de Leds en los problemas establecidos.

Programación en Arduino

Deberemos desarrollar un programa que nos permita controlar tanto el Arduino como el integrador en el control de los 64 Leds, teniendo presente que tendremos que realizar diferentes patrones luminosos.

Desarrollo del algoritmo

Esquema de tareas

Diseño algoritmo de control

1. Definición de los requisitos: Hemos desarrollado esta parte en el inicio del informe, pudimos identificar que necesitamos mostrar patrones por lo tanto es importante el control independiente de los Leds, sin superar el uso de pines digitales acordados en Arduino.
2. Selección de Integrado: Usaremos el integrado 74HC595 que nos permitirá la conexión de los diferentes Leds sin superar el límite de pines acordados, aparte de ello nos permitirá el control de estos mismos.
3. Planificación de conexiones: Realizaremos un boceto de cómo podemos realizar las conexiones entre el Arduino, Integrado y la matriz de Leds.

Programación en Arduino

4. Inicialización de los pines: Debemos de configurar los pines digitales utilizados.
5. Configuración de una matriz: Crear una estructura de datos que nos represente la matriz 8x8 de Leds, podríamos crear una matriz bidimensional.
6. Programación del bucle principal: La implementación que nos permita encender y apagar los diferentes Leds de acuerdo con los patrones indicados o deseados.

Pruebas

7. Verificar las conexiones: validar que todas las conexiones entre los componentes estén correctas.
8. Prueba de funcionalidad: correr o ejecutar el código para verificar que todos los Leds enciendan y apaguen según lo que programemos.
9. Identificación de los errores en las pruebas: Aquí visualizaremos esos errores que no nos permitan el desarrollo o el funcionamiento correcto de la solución del problema.

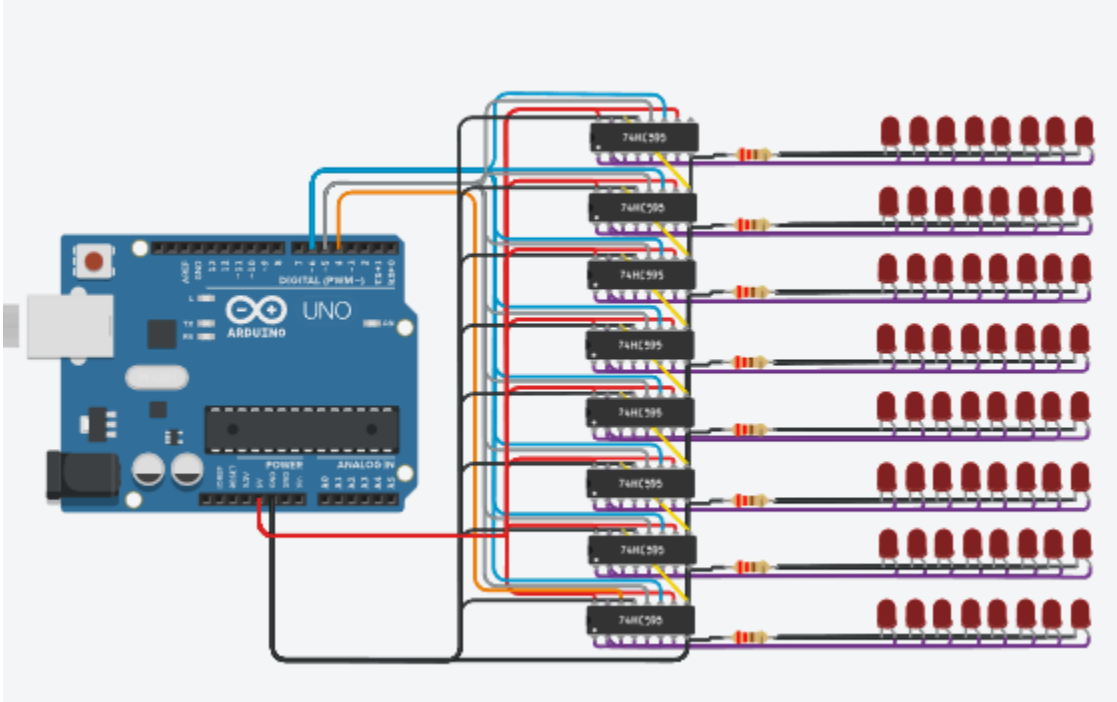
Documentación

10. Llevar comentado el desarrollo del código para un mayor entendimiento: Nos facilitaríamos el proceso de desarrollo del algoritmo o código teniendo comentado lo que vamos

realizando de esta manera podemos recordar de manera fácil el porqué de lo que utilizamos.

Algoritmos desarrollados

Conexión final



Código para el encendido de los leds (Verificación)

```
void verificacion(){
  for(int i=0; i<8; i++){
                    digitalWrite(latchPin, LOW);
                    shiftOut(dataPin, clockPin, LSBFIRST,B11111111);
    digitalWrite(latchPin, HIGH);
  }
  delay(1000);
  for(int i=0; i<8; i++){
                    digitalWrite(latchPin, LOW);
                    shiftOut(dataPin, clockPin, LSBFIRST,B00000000);
                    digitalWrite(latchPin, HIGH);
  }
  delay(2000);
}
```

Función imagen

En esta función se debe de pedirle al usuario que ingrese el patrón deseado, para esto, lo analice de la siguiente forma:

1. Todos los patrones tienen un sistema 8x8, ya que nuestro sistema está compuesto de esa manera.
2. Podemos solicitarle al usuario que ingrese los datos de cada fila hasta completar 8, claramente cada fila ingresada debe de tener 8 datos, entre 0 y 1, donde 0 significa ausencia de luz y 1 datos o luz de la imagen.

```
void imagen(){
  for (int fila=0; fila<8; fila++){
    Serial.print("Fila " + String(fila+1)+": ");
      while(!Serial.available());
      String bitChar = Serial.readStringUntil('\n');
      patron[fila]= byte(strtol(bitChar.c_str(), NULL, 2));

    Serial.println();
  }
}
```

Patrones

Patrón 1 – rombo

Este patrón como los siguientes siguen una secuencia, en este la primera fila enciende los 2 les del centro y por cada fila que baje, van encendiendo dos más a cada lado de los leds centrales, hasta encender los ocho leds, esto se ve a la mitad de las filas, en la posición 4. Luego se repite el ciclo, pero desde 8 leds hasta llegar a los dos leds centrales. Entonces se hizo el siguiente código:

```
void patron1(){
  int p1=3; int p2=4;
    for (int i = 0; i < 8; ++i) {

        String tempo="00000000";
        if(i<4){
            for (int k=p1; k <=p2; ++k) {
                tempo[k]='1';
            }
```

```
            patron[i]=byte(strtol(tempo.c_str(), NULL, 2));
            p1-=1; p2+=1;
        }else{
            p1+=1; p2-=1;
            for (int k=p1; k <=p2; ++k) {
                tempo[k]='1';
            }
            patron[i]=byte(strtol(tempo.c_str(), NULL, 2));
        }
    }
}
```

Patrón 2 – X

Para hallar la solución a este patrón definido, se buscó un tipo de secuencia o cuales eran los pasos para que se pudiera crear el patrón utilizando algún tipo de ciclo, arreglo entre otros tipos de elementos.

De acuerdo con lo anterior, se halló que los leds del sistema encienden, si el número de led de la fila coincidía con el número del led de la columna, por lo tanto, se desarrolló el siguiente código:

```
void patronX(byte m[8]){
    for(int fila=0; fila<8; fila++){
        String tempo;
        for(int columna=0; columna<8;columna++){
            if((fila==columna) || (fila==7-columna)){
                tempo+="1";
            }else{
                tempo+="0";
            }
        }
        patron[fila]=byte(strtol(tempo.c_str(), NULL, 2));
    }
}
```

Patrón 3

```
void patron3(){
  int contador=0;
    for (int i = 0; i < 8; ++i) {
        String tempo="00000000";
        if((i==2 || i==3) || (i==6 || i==7)){
            for(int n=1; n<8;n++){
                if(contador==2 || contador==5) ;
                else{
                    tempo[n]='1';
                }
```

```
                contador++;
            }
        }else{
            for(int n=0; n<8;n++){
                if(contador==2 || contador==5);
                else{
                    tempo[n]='1';
                }
                contador++;
            }
        }
        contador=0;
        patron[i]=byte(strtol(tempo.c_str(), NULL, 2));
    }
}
```

Patrón 4 - flecha.

Para la solución de este patrón, observe que son cuatro los leds que encienden en cada fila, la primera fila enciende los leds desde la posición 0, bajando por fila, la posición se va corriendo un paso hacia la derecha. Entonces se llegó al siguiente código:

```
void flecha(){
    int p1=0; int p2=3;
    for (int i = 0; i < 8; ++i) {

        String tempo="00000000";
        if(i<4){
            for (int k=p1; k <=p2; ++k) {
                tempo[k]='1';
            }
            patron[i]=byte(strtol(tempo.c_str(), NULL, 2));
            p1+=1; p2+=1;
        }else{
            p1-=1; p2-=1;
            for (int k=p1; k <=p2; ++k) {
                tempo[k]='1';
            }
            patron[i]=byte(strtol(tempo.c_str(), NULL, 2));
        }
    }
}
```

Publik

Esta función reúne los ejercicios planteados anteriormente, agregando algunas cosas como el tiempo de apagado y encendido que debe ser digitado por el usuario, para ello se utilizo código de las funciones ya descritas para codificar esta función.

Partes

Principal

```
void publik(){
                    if (Serial.available()) {
    char opcion = Serial.read();
    switch (opcion) {
      case 'a':
        verificarLeds();
        break;
      case 'b':
        imagenPrueba();
        break;
      case 'c':
        mostrarAleatorio();
        break;
    }
  }
}
```

Verificación de Leds

```
void verificarLeds(){
  int nSecuencias;
  int timeOn;
  int timeOff;
  Serial.println("Ingrese el numero de secuencias: ");
  while (!Serial.available());
  nSecuencias=Serial.parseInt();
  Serial.println("Ingrese el tiempo de encendido en milisegundos: ");
  while (!Serial.available());
  timeOn=Serial.parseInt();
  Serial.println("Ingrese el tiempo de apagado en milisegundos: ");
  while (!Serial.available());
  timeOff=Serial.parseInt();
  for(int s=0; s<nSecuencias; s++){
    for(int i=0; i<8; i++){
        digitalWrite(latchPin, LOW);
        shiftOut(dataPin, clockPin, LSBFIRST,B11111111);
        digitalWrite(latchPin, HIGH);
    }
    delay(timeOn);
    for(int i=0; i<8; i++){
        digitalWrite(latchPin, LOW);
        shiftOut(dataPin, clockPin, LSBFIRST,B00000000);
        digitalWrite(latchPin, HIGH);
    }
    delay(timeOff);
  }
}
```

Mostrar patrones de prueba

```
      void imagenPrueba(){
        for (int fila=0; fila<8; fila++){
          Serial.print("Fila " + String(fila+1)+": ");
            while(!Serial.available());
            String bitChar = Serial.readStringUntil('\n');
            patron[fila]= byte(strtol(bitChar.c_str(), NULL, 2));

          Serial.println();
        }
        int timeOn, timeOff;
        Serial.println("Ingrese el tiempo de encendido en milisegundos: ");
        while (!Serial.available());
        timeOn=Serial.parseInt();
        Serial.println("Ingrese el tiempo de apagado en milisegundos: ");
        while (!Serial.available());
        timeOff=Serial.parseInt();

        digitalWrite(latchPin, LOW);
        for (int fila=0; fila<8;fila++){
            shiftOut(dataPin, clockPin, LSBFIRST, patron[fila]);
          }
        digitalWrite(latchPin, HIGH);
        delay(timeOn);

        digitalWrite(latchPin, LOW);
        for (int fila=0; fila<8;fila++){
            shiftOut(dataPin, clockPin, LSBFIRST, B00000000);
          }
        digitalWrite(latchPin, HIGH);
        delay(timeOff);
      }
```

Mostrar patrones aleatorios

```
      void mostrarAleatorio(){
        int tiempoRetardo;
        Serial.println("Ingrese el tiempo de retardo entre patrones en
milisegundos:");
        while (!Serial.available());
        tiempoRetardo = Serial.parseInt();

        for (int i = 0; i < 4; i++) {
          switch (i) {
            case 0:
              patron1();
              break;
            case 1:
              patron2();
              break;
            case 2:
              patron3();
              break;
            case 3:
              patron4();
              break;
```

```
    }
    mostrar(patron, tiempoRetardo);
  }
  for(int a=0; a<8; a++){
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, LSBFIRST,B00000000);
    digitalWrite(latchPin, HIGH);
  }
}
```

Problemas de desarrollo

Conexiones - Errores

Se inicio el desarrollo de las conexiones, teniendo en cuenta 8 integrados, 64 resistencias y 64 leds, aquí abajo un ejemplo de las conexiones iniciales.

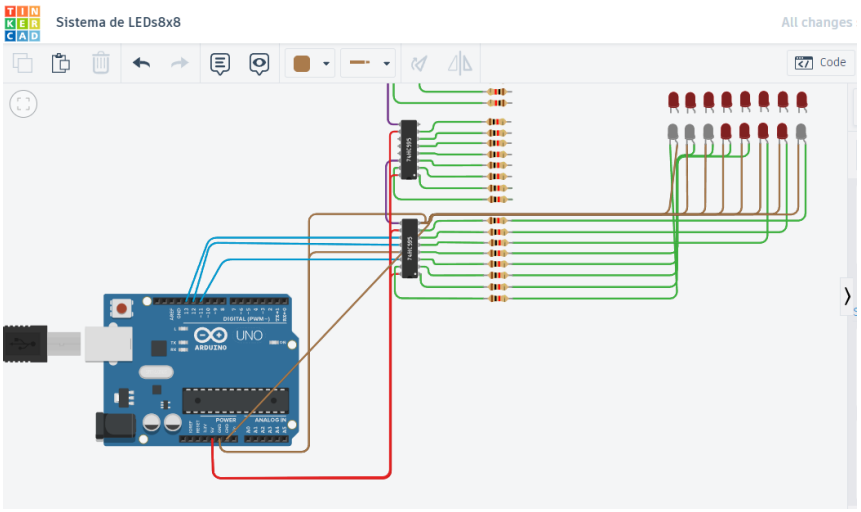


Ilustración 1. Inicio de conexiones

Usé el integrado 74HC595 para poder controlar los diferentes Leds, entonces se realizaron las conexiones necesarias, según consultas en la web, tuve en cuenta algunas imágenes como referencia.

Las siguientes imágenes son las usadas como apoyo.

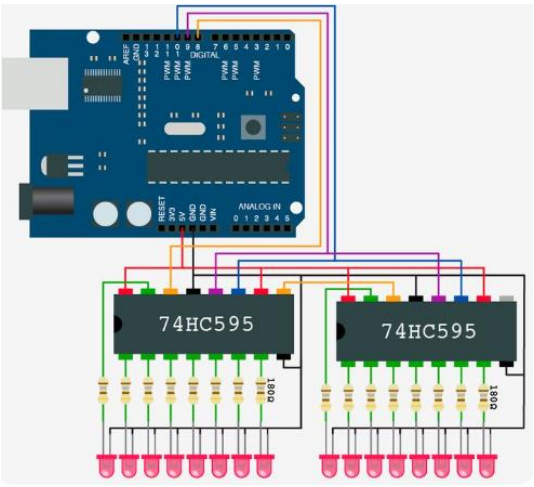


Ilustración 2. Conexión múltiple Integrado

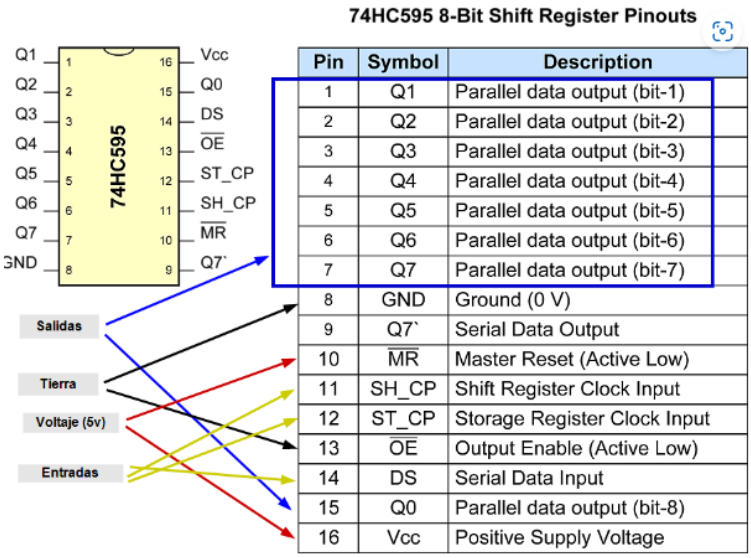


Ilustración 3. Componentes de Integrado

Luego de hacer estas conexiones se programó el código, pero al ejecutarlo para el encendido de los Leds, este corre correctamente, pero los Leds no enciende, este es uno de los primeros errores o dificultades halladas en el transcurso de la solución al caso.

Uno de los principales motivos de este error es el conocimiento previo de los componentes utilizados, ya que al no saber cómo funcionan, me guie por consultas externas.

Aparte de ello, se hizo la conexión de leds directo al Arduino para validar si funcionaba y estos encendieron correctamente, por lo cual mi error es *la conexión entre el Arduino, Integrado y los leds.*

Finalmente, luego de validar y consultar se llegó a la conexión correcta y acertada para el funcionamiento de los leds, que es el uso de cuatro integrados y dos resistencias por cada integrado ya que la conexión mostrada anteriormente no funciono y eran muchos componentes para usar, hay que tener en cuenta que el uso de solo cuatro integrados puede ser un problema para los patrones siguientes en el caso.

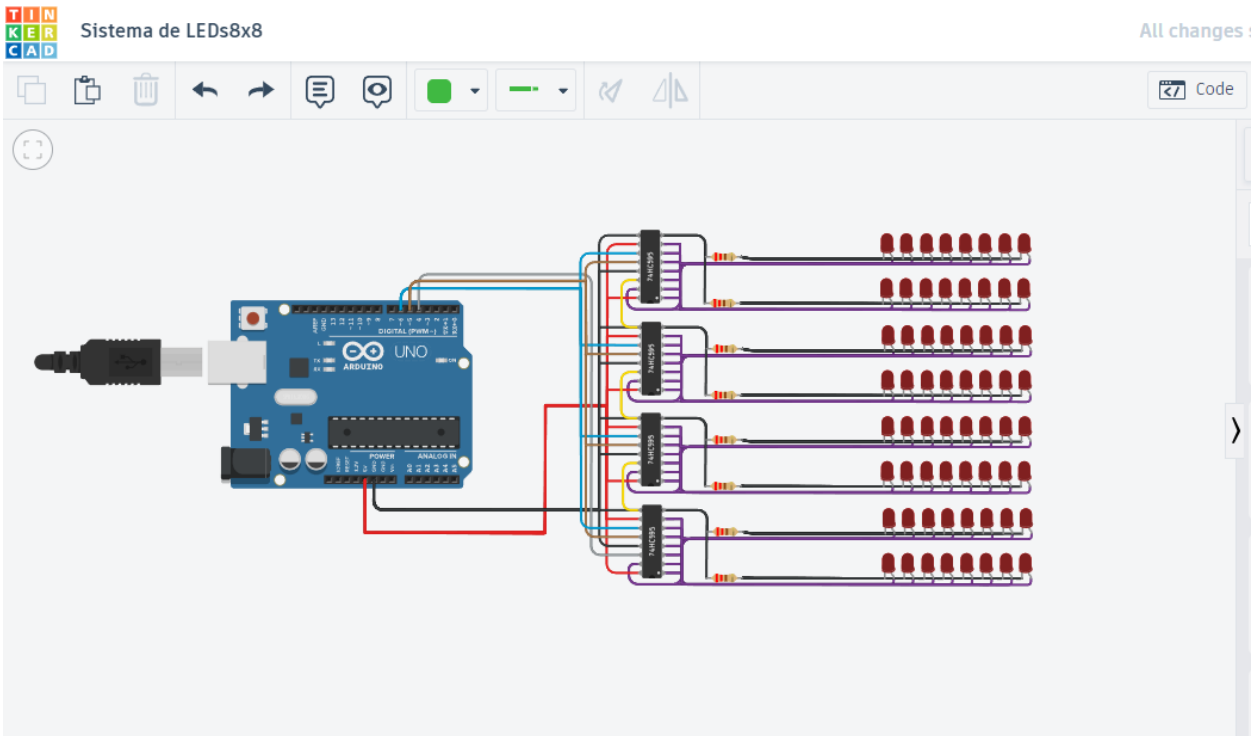


Ilustración 4. Conexión-Pre final.I

Podemos observar, que tanto las conexiones como el desarrollo visual es mucho mejor que al inicio del desarrollo, aparte de ello se hizo las correcciones necesarias como *la conexión de la resistencia al pin GND del integrado y luego cátodos a la resistencia, ya que inicialmente se tenía la conexión de resistencia a pines de salida del integrado.*

Errores en código de programación

Encendido

El siguiente código enciende los Leds, pero no todos a la vez, los enciende haciendo un tipo de intercalado entre ellos y lo que se busca es el encendido de todos a la vez.

```
#include <SPI.h>

const int latchPin=5;
const int clockPin=6;
const int dataPin=4;
byte patron[8][8]={0};

void setup(){

  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}

Void loop(){
  //Encendido de Leds
  digitalWrite(latchPin,LOW);
  shiftOute(dataPin, clockPin, MSBFIRST, B11111111);
  digitalWrite(latchPin, HIGH);

  delay(1000);

  //Apagado de Leds
  digitalWrite(latchPin,LOW);
  shiftOute(dataPin, clockPin, MSBFIRST, B00000000);
  digitalWrite(latchPin, HIGH);

  delay(1000);
}
```

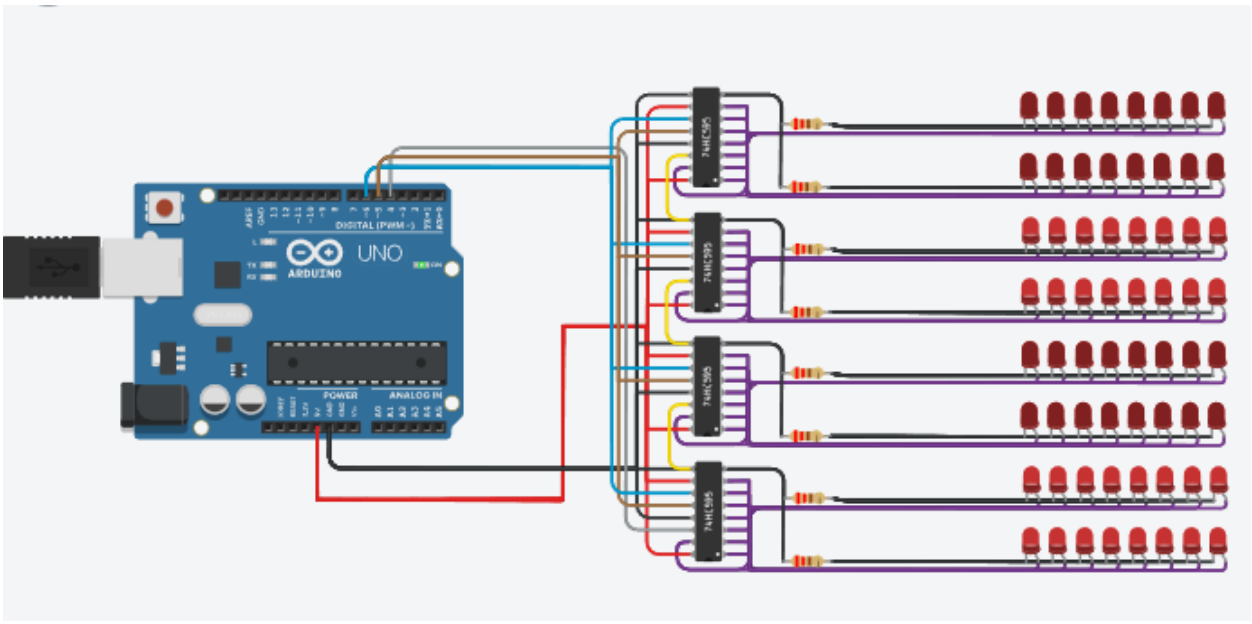
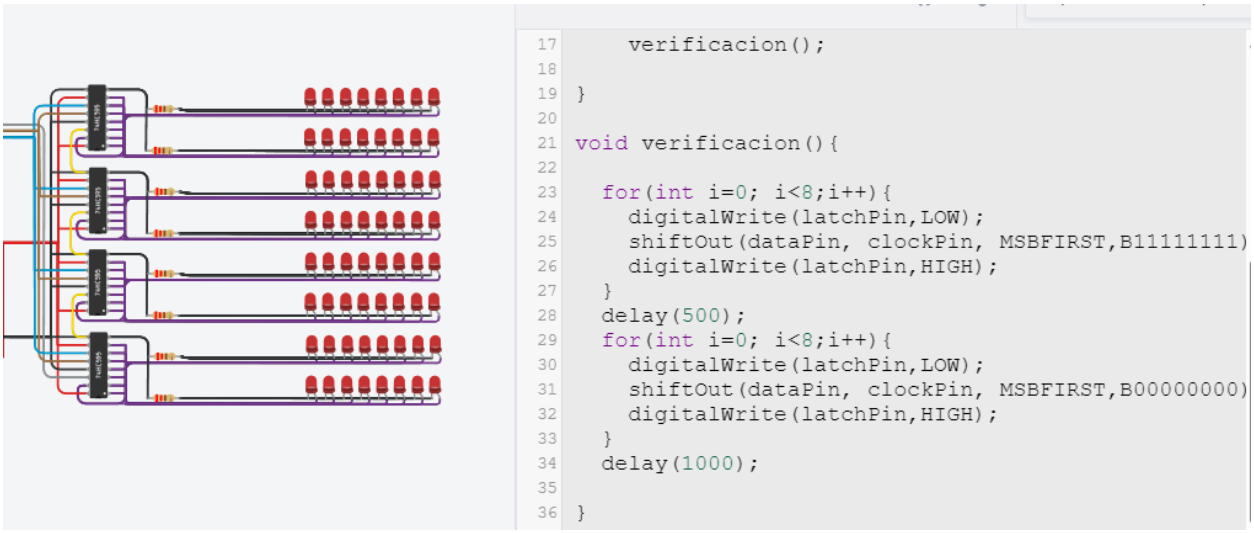


Ilustración 5. Encendido de Leds - Intercalado

Luego de validar el código de la función verificación, se concluye el uso de un ciclo for para la solución del encendido intercalado, funcionando correctamente el encendido y apagado a la vez.



```
    verificacion();

}

void verificacion(){

  for(int i=0; i<8;i++){
    digitalWrite(latchPin,LOW);
    shiftOut(dataPin, clockPin, MSBFIRST,B11111111);
    digitalWrite(latchPin,HIGH);
  }
  delay(500);
  for(int i=0; i<8;i++){
    digitalWrite(latchPin,LOW);
    shiftOut(dataPin, clockPin, MSBFIRST,B00000000);
    digitalWrite(latchPin,HIGH);
  }
  delay(1000);

}
```

```
    verificacion();

}

void verificacion(){

  for(int i=0; i<8;i++){
    digitalWrite(latchPin,LOW);
    shiftOut(dataPin, clockPin, MSBFIRST,B11111111);
    digitalWrite(latchPin,HIGH);
  }
  delay(500);
  for(int i=0; i<8;i++){
    digitalWrite(latchPin,LOW);
    shiftOut(dataPin, clockPin, MSBFIRST,B00000000);
    digitalWrite(latchPin,HIGH);
  }
  delay(1000);

}
```

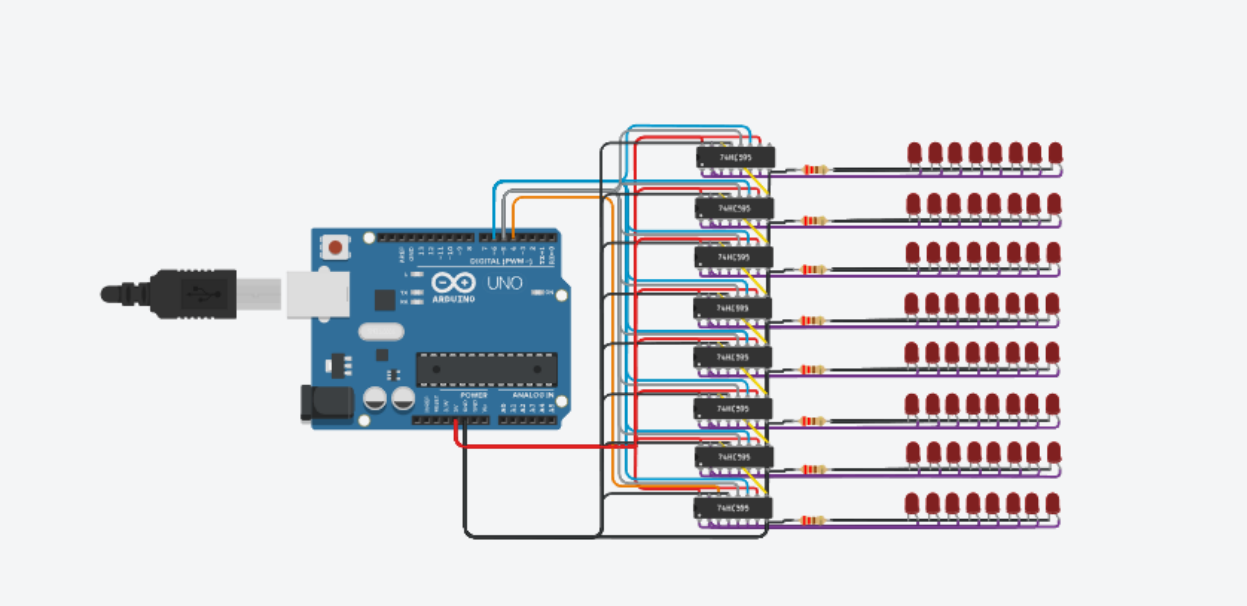
Patrón X

En el desarrollo de la función de patrones, se inició intentando programar el código para el patrón X, pero al correr este, encendieron los leds, pero al haber solo 4 integrados el patrón no se hizo correctamente.

```
#include <SPI.h>

const int dataPin = 4;  // Pin de datos conectado a
const int clockPin = 6; // Pin de reloj conectado a
const int latchPin = 5; // Pin de almacenamiento co

// Define una secuencia de bits para el patrón X
byte patronX[] = {
  B10000001,
  B01000010,
  B00100100,
  B00011000,
  B00011000,
  B00100100,
  B01000010,
  B10000001
};

void setup() {
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
```

Por lo tanto, es necesario, agregar otros cuatro integrados para poder que el patrón se muestre correctamente.

De acuerdo con lo anterior, se agregaron otros 4 integrados.



Al correr el código con la creación de secuencia utilizando ciclos for, se visualiza que se muestra el patrón, pero los demás leds, enciende con una luz tenue cuando el tiempo de encendido es bajo.

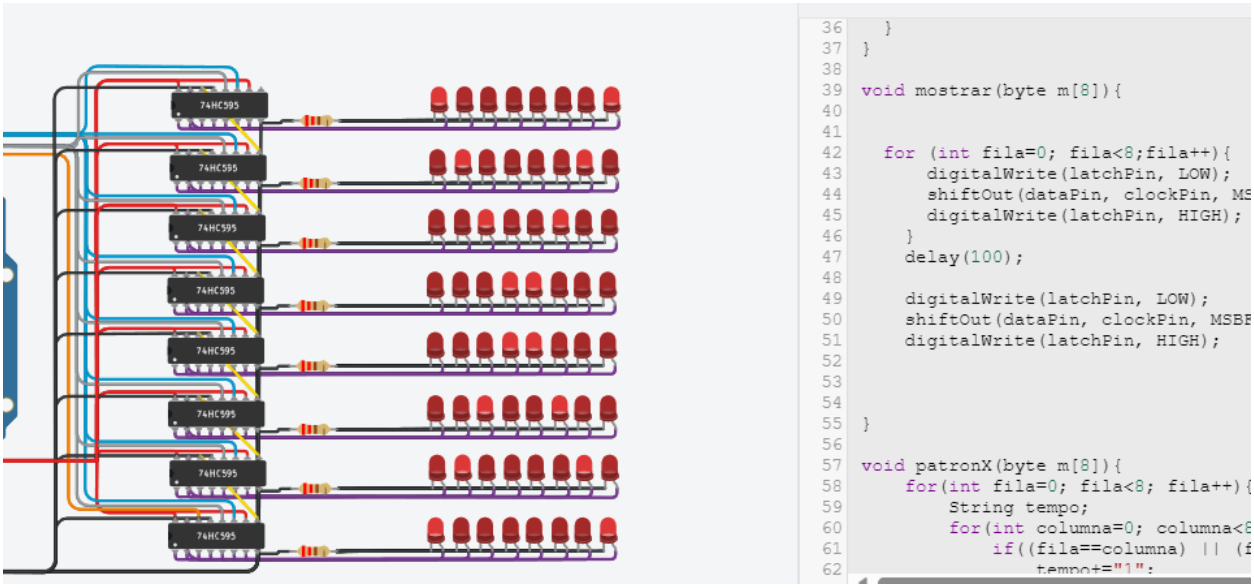


Ilustración 6. Prueba patrón X

Validando los códigos hechos, se halló que este error se debe a la función de mostrar, en la parte de *digitalWrite*, ya que estos se encuentran dentro

Referencias

Cómo funciona el registro de turnos 74HC595 y su interfaz con Arduino. (s.f.). Obtenido de Descubriendo Arduino: https://descubrearduino.com/74hc595/

Damian, J. (21 de 05 de 2021). *Cómo funciona el 74HC595 Shift Register y su interfaz con Arduino*. Obtenido de Electro geek: https://www.electrogeekshop.com/como-funciona-el-74hc595-shift-register-y-su-interfaz-con-arduino/

How to use 74HC595 Shift Register with Arduino? (14 de 09 de 2018). Obtenido de Electronics Hub: https://www.electronicshub.org/74hc595-shift-register-with-arduino/#:~:text=First%2C%20connect%20the%20Serial%20Input%20Pin%20of%2074HC595,to%20the%208%20output%20pins%20of%2074HC595%20IC