Parcial 1: Informática II

Análisis y diseño.

JOSE MIGUEL GOMEZ MONSALVE

ERIKA DAYANA LEÓN QUIROGA

DAVID AGUDELO OCHOA

Despartamento de Ingeniería Electrónica y Telecomunicaciones
Universidad de Antioquia
Medellín
Febrero 2022

Índice

1. Integrado 74HC595.

El integrago 74HC595 hace parte de la familia de dispositivos SNx4HC59, la cual contienen un registro de desplazamiento de 8 bits de entrada en serie y salida en paralelo, el registro de almacenamiento tiene salidas de 3 estados paralelos. Se proporcionan relojes separados para el registro de desplazamiento y el de almacenamiento. El registro de desplazamiento tiene una entrada de anulación directa (SRCLR), una entrada en serie (SER) y salidas en serie para la conexión en cascada. Tienen una amplia corriente de funcionamiento de 2 V a 6 V, y las salidas de 3 estados de alta corriente pueden controlar hasta 15 cargas LSTTL. Los dispositivos tienen un bajo consumo consumo de 80-A (máximo) ICC.

1.1. Características del integrado.

- Entrada serial, salida paralela, o salida serial que permite la conexión en cascada de varios integrados.
- Registro de desplazamiento de 8 bits que alimenta a un registro de almacenamiento.
- Entradas de reloj separadas para el registro de desplazamiento y el de almacenamiento con activación por flanco de subida.

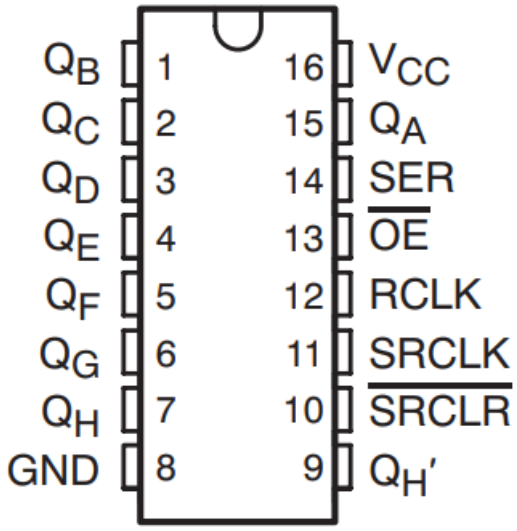

Figura 1: Integrado 74HC595.

Configuración de pines.

- Los pines de Q_B(pin 1) a Q_H(pin 7), añadiendo Q_A(pin 15) representan las salidas del integrado.
- V_{CC}(pin 16) es la alimentación y GND(pin 8) se conecta a tierra.
- $Q_{H'}$(pin 9) se utiliza para conectar otro integrado 74HC595 y generar un efecto de cascada.
- El pin 14 o SER es el pin donde se envían los datos.
- $\bar{OE}$(pin 13) llamado Output Enable, habilita las salidas y se activa con un nivel bajo, por lo cual, para que siempre esté activo se conecta a GND.

- El RCLK (pin 12) es el reloj del registro de almacenamiento y se utiliza para actualizar los datos a los pines de salida.
- El SRCLK (pin 11) es el reloj que sincroniza la carga de datos.
- El $SR\bar{C}LR$ (pin 11) llamado Shift Register Clear, reestablece el registro de desplazamiento.

1.2. Funcionamiento.

El objetivo principal es pasar el número dado de un formato serial a uno parelelo. Para explicar el funcionamiento del integrado tomaremos un número cualquiera de 8 bits, este número se irá guardando bit por bit en cada uno de los cuadros que se pueden ver en la Figura 2, también podemos ver en esta misma figura que la entrada de los datos es en serie (uno por uno), y la salida de ellos es en paralelo (8 bits).

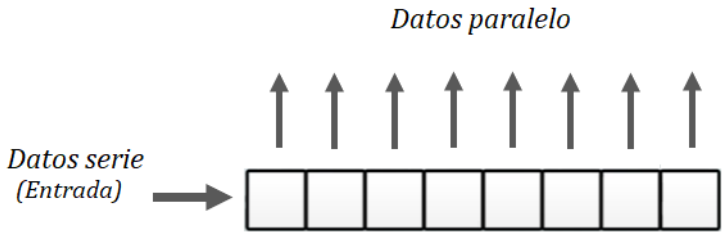

Figura 2: Representación del funcionamiento del circuito integrado 74HC595.

Para que la toma de los datos sea exitosa es necesario un reloj que por medio de pulsos, controlará en qué momento ingresa al integrado el bit presente en la entrada. Tomaremos de ejemplo la representación binaria del número 49, la cual es 00110001.

El primer paso para transformar la información que se encuentra en serie a paralelo es realizar el desplazamiento de los bits dentro del integrado iniciando por el bit más significativo (MSB por sus siglas en inglés), en nuestro ejemplo es un cero, que se encuentra presente en la entrada y que en el primer pulso del rejol ingresa a la primera posición del registro de desplazamiento. Figura 3

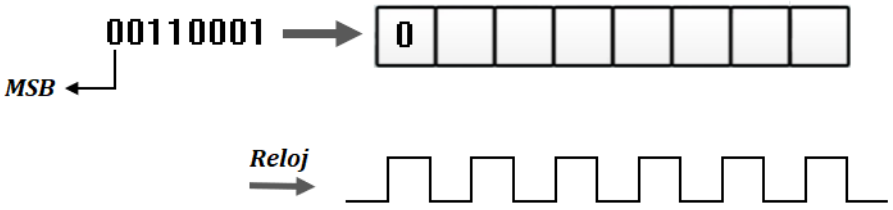

Figura 3: Entrada del MSB al integrado.

En el próximo pulso del reloj el MSB, ya dentro del integrado, se correrá una posición a la derecha en el registro de desplazamiento, mientras que el número a la derecha del MSB, en la entrada, se posicionará en la primera posición del registro de desplazamiento.

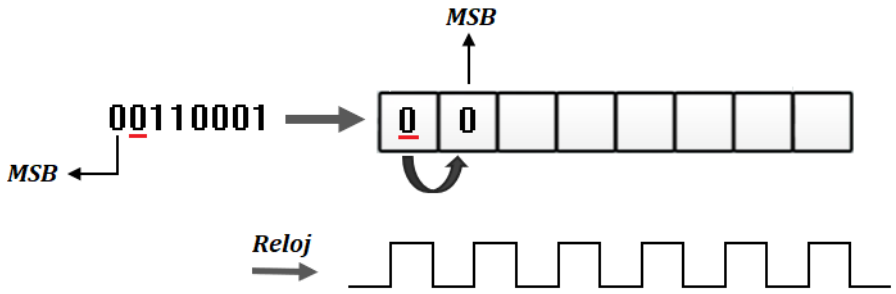

Figura 4: Desplazamiento de los bits dentro del integrado.

Este proceso se repetirá hasta que se ingrese al registro de desplazamiento el último bit del número (Figura 6).

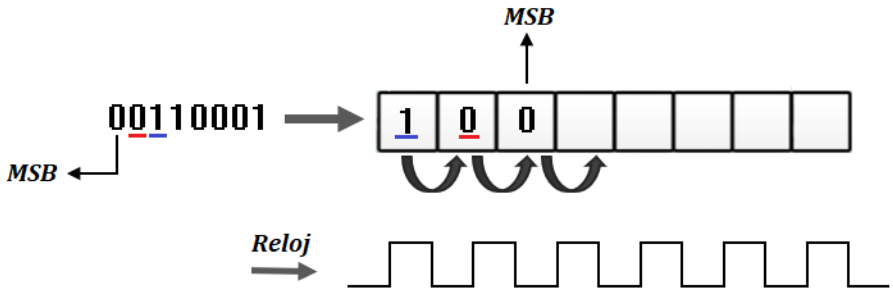

Figura 5: Desplazamiento de los bits dentro del integrado.

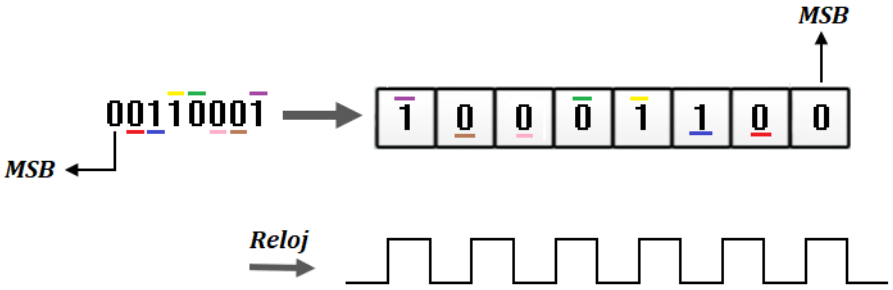

Figura 6: Registro de desplazamiento lleno. Se utilizaron colores encima o abajo de los bits para identificar fácilmente cuál es cuál en el registro de desplazamiento.

Para que las salidas no vayan cambiando mientras que el registro de desplazamiento se llena, el integrado hace uso del registro de almacenamiento y el RCLK (Rejol del registro de almacenamiento). Figura 7

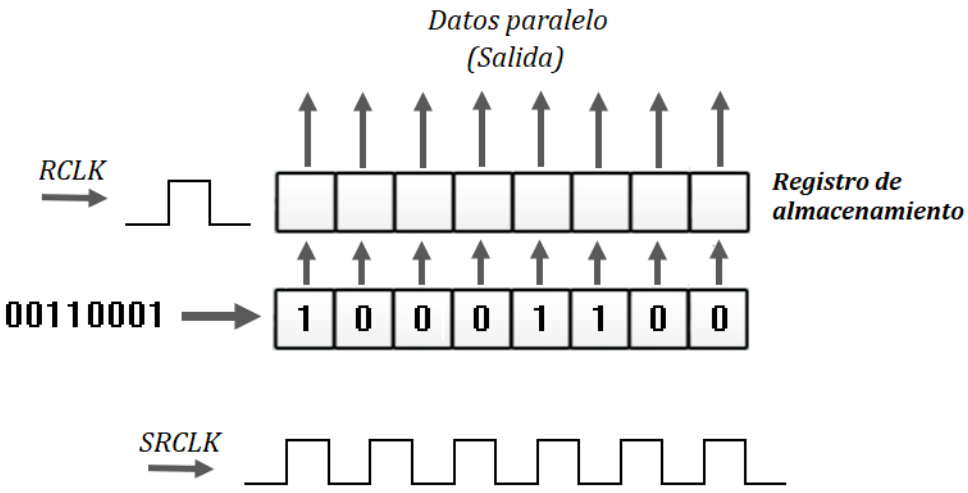

Figura 7: Representación del registro de almacenamiento.

Una vez se finalice la carga de datos en el registro de desplazamiento, con un único pulso del RCLK se cargan todos los datos del registro de desplazamiento al registro de almacenamiento y se muestran en la salida. De esta forma se alcanzará el objetivo principal, transformar los datos de entrada en serie a datos de salida paralelos.

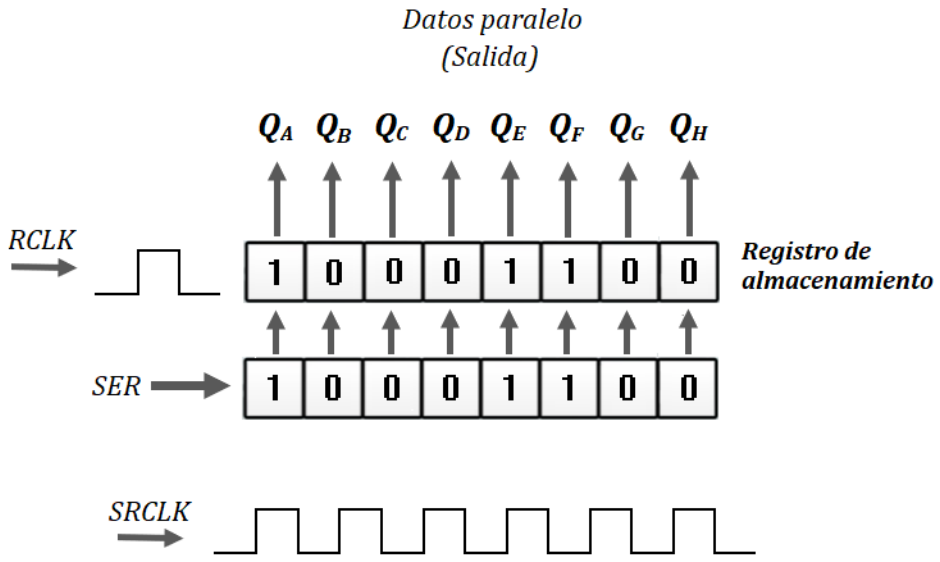

Figura 8: Representación del registro de almacenamiento.

1.3. Aplicaciones.

- Network switches
- Power infrastructure
- LED displays
- Servers

1.4. Uso del Circuito Integrado 74HC595.

- Con pulsadores:

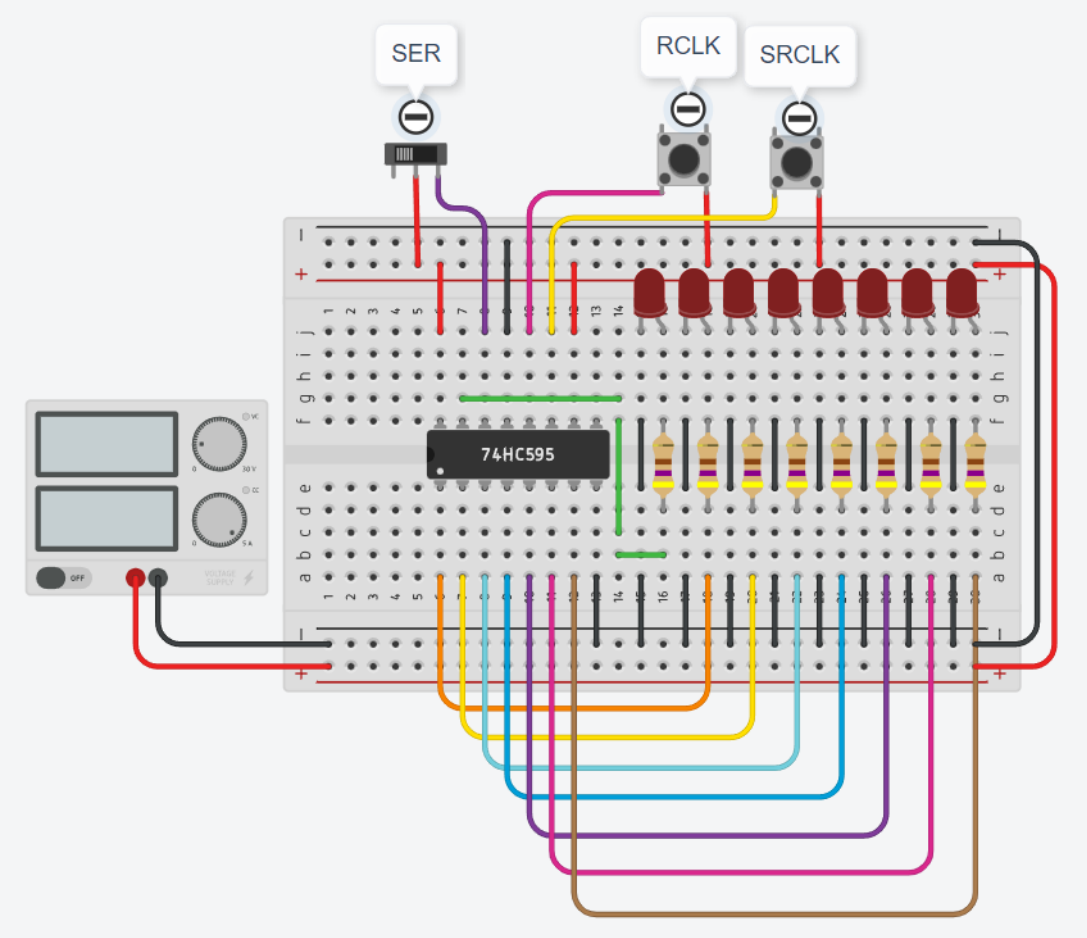

Figura 9: Circuito con integrado 74HC595 haciendo uso de pulsadores.

La simulación se puede probar utilizando el siguiente link: https://www.tinkercad.com/things/iBjAIisXge3

2. Comunicación entre Arduinos

2.1. Intento número 1

A continuación se presenta el código y montaje de arduinos que se usaron para probar el tema de comunicación entre dos arduinos. Cabe aclarar que la implementación de estos se dio antes de las

indicaciones y especificaciones de la clase del día 17 de febrero, por lo que no reflejan el futuro diseño a implementar, el cual contará con una gran participación del integrado 74HC595 y se sospecha que también del protocolo I2C para la señal de reloj.

2.1.1. arduino emisor

montaje:

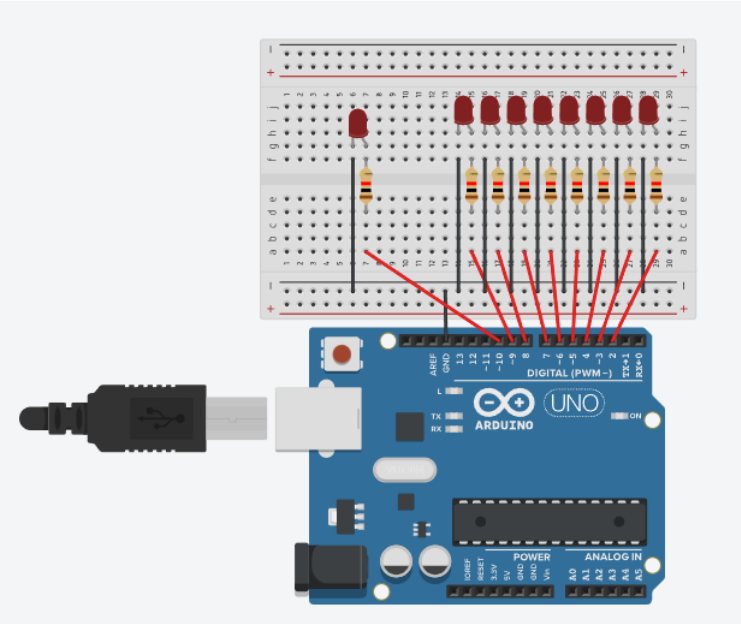

Figura 10: Prueba de emisión de bits y una señal de reloj por medio de 9 leds.

código:

```
#define tiempo 1000
byte codigo[8]={1,2,3,4,5,6,7,8};//A byte stores an 8-bit unsigned
    number, from 0 to 255.
//----------------------------------------------------------------
//prototipo de las funciones

//----------------------------------------------------------------
//setup
void setup()
{
  for(int i=2;i<11;i++)
  {
    pinMode(i, OUTPUT);
  }

}
//----------------------------------------------------------------
//loop
void loop()
{
  for(int i=0;i<8;i++)
  {
    digitalWrite(10, LOW);
```

```
23      delay(25); // Wait for 1000 millisecond(s)
24      for(int j=0;j<8;j++)
25      {
26        digitalWrite(j+2, bitRead(codigo[i], j));
27      }
28      delay(tiempo-25);
29      digitalWrite(10, HIGH);
30      delay(tiempo);
31    }
32
33
34  }
35  //--------------------------------------------------------------------
36  //cuerpo de las funciones
```

2.1.2. arduino emisor y receptor

montaje:

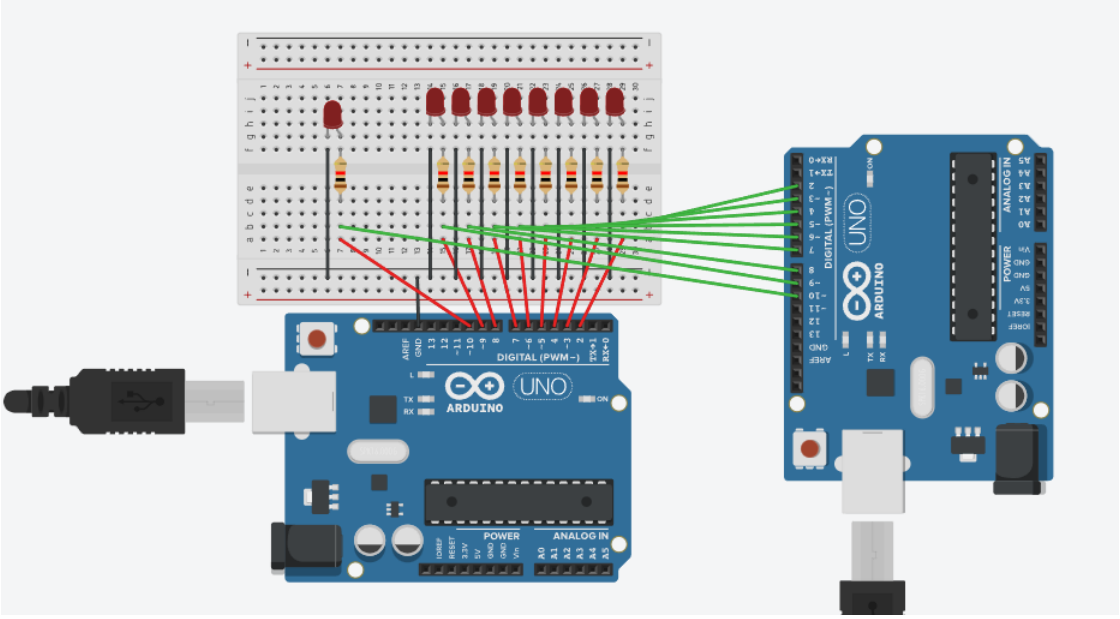

Figura 11: Prueba de emisión de bits y una señal de reloj por medio de 9 leds y su recepción a otro arduino.

código arduino emisor:

```
//directivas de preprocesamiento
#define tiempo 250
byte codigo[8]={2,1,3,4,5,6,7,8};//A byte stores an 8-bit unsigned
    number, from 0 to 255.
//---------------------------------------------------------------
//prototipo de las funciones

//---------------------------------------------------------------
//setup
void setup()
{
  for(int i=2;i<11;i++)
  {
    pinMode(i, OUTPUT);
  }

}
//---------------------------------------------------------------
//loop
void loop()
{
  for(int i=0;i<9;i++)
  {
    digitalWrite(10, LOW);
    delay(50); // Wait for 1000 millisecond(s)
    for(int j=0;j<8;j++)
```

```
    {
      digitalWrite(j+2, bitRead(codigo[i], j));
    }
    delay(tiempo-50);
    digitalWrite(10, HIGH);
    delay(tiempo);
  }


}
//-----------------------------------------------------------------
//cuerpo de las funciones
```

código arduino receptor:

```
// C++ code
//
int entero=0;
bool primerCero=false;
void setup()
{
  Serial.begin(9600);
  for(int i=2;i<11;i++)
  {
    pinMode(i, INPUT);
  }
}

void loop()
{
  if(primerCero==true)
  {
    for(int i=0;i<8;i++)
    {
      entero=entero+((int)digitalRead(i+2))*pow(2,i);
    }
    Serial.println(entero);
    entero=0;
    primerCero=false;
  }
  if(primerCero==false)
  {
    while(digitalRead(10)==true)
    {
      primerCero=true;
    }
  }

}
```

2.2. Intento número 2

A continuación se presenta un modelo más cercano a lo que se planea hacer para la comunicación de la señal de reloj y los datos. Para ello usaremos el protocolo I2C.

montaje:

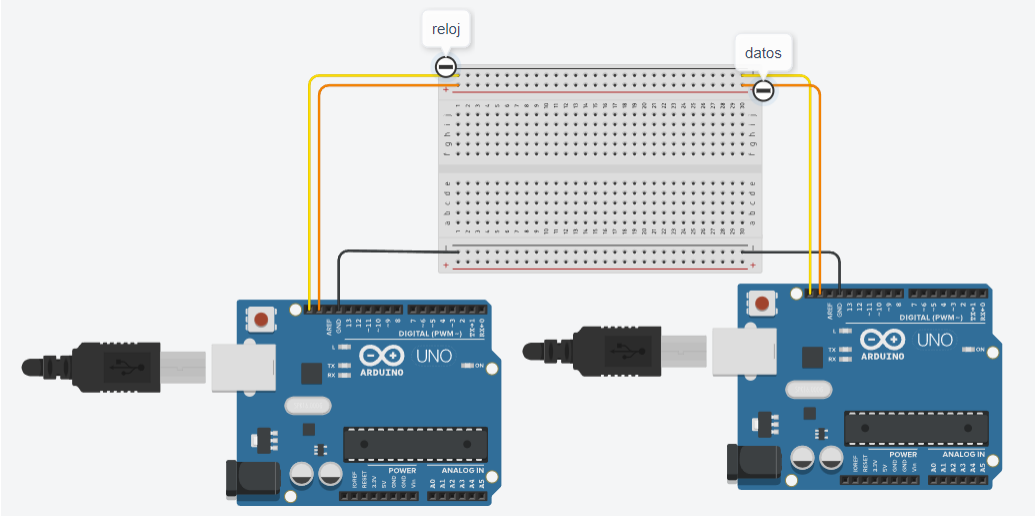

Figura 12: Comunicación mediante el protocolo I2C.

Código arduino emisor:

```
#include <Wire.h>

// C++ code
//----------------------------------------------------------------------
//----------------------------------------------------------------------

byte pin[] = {9, 10, 11, 12, 13};
byte estado = 0;
#define retardo 2000
int ValorSensor = 0;

//----------------------------------------------------------------------
//----------------------------------------------------------------------

void setup()
{
  Wire.begin();
  pinMode(LED_BUILTIN, OUTPUT);
}

//----------------------------------------------------------------------
//----------------------------------------------------------------------

void loop()
{
  Wire.beginTransmission(1); // Transmite al Esclavo 1
  Wire.write(estado);
  Wire.endTransmission();

  digitalWrite(LED_BUILTIN, estado);

  delay(retardo);
```

```
142
143   if (estado == 0)
144   {
145     estado = 1;
146   }
147   else
148   {
149     estado = 0;
150   }
151 }
```

Código arduino receptor:

```
153 #include <Wire.h>
154
155 void llegaDato(int howMany);
156 // C++ code
157 //
158 void setup()
159 {
160   pinMode(13, OUTPUT);     // Pines en modo salida
161   Wire.begin(1); // Unimos este dispositivo al bus I2C con direcci n 2
          (Esclavo 2)
162   Wire.onReceive(llegaDato);
163   pinMode(LED_BUILTIN, OUTPUT);
164 }
165
166 void loop()
167 {
168   delay(30); // Wait for 1000 millisecond(s)
169 }
170
171 void llegaDato(int howMany){
172   int estado = 0;
173   if (Wire.available() == 1) // Si hay un byte disponible
174   {
175     estado = Wire.read();
176   }
177   digitalWrite(13,estado);   // Activamos/desactivamos salida depende
        del Maestro
178 }
```

2.3. Intento número 3

Tras la sesión de clase del día 25 de febrero, el profesor nos indicó que no había necesidad de recurrir a protocolos como el I2C, por lo que se procedió a hacer una implementación mediante el uso de puertos digitales únicamente.
A continuación se presenta primeramente el montaje del uso del circuito integrado 74HC595 con pulsadores, pero esta vez reemplazando estos últimos con un arduino, para probar así el proceso de emisión del arduino y de recepción del integrado en conjunto.
Montaje:

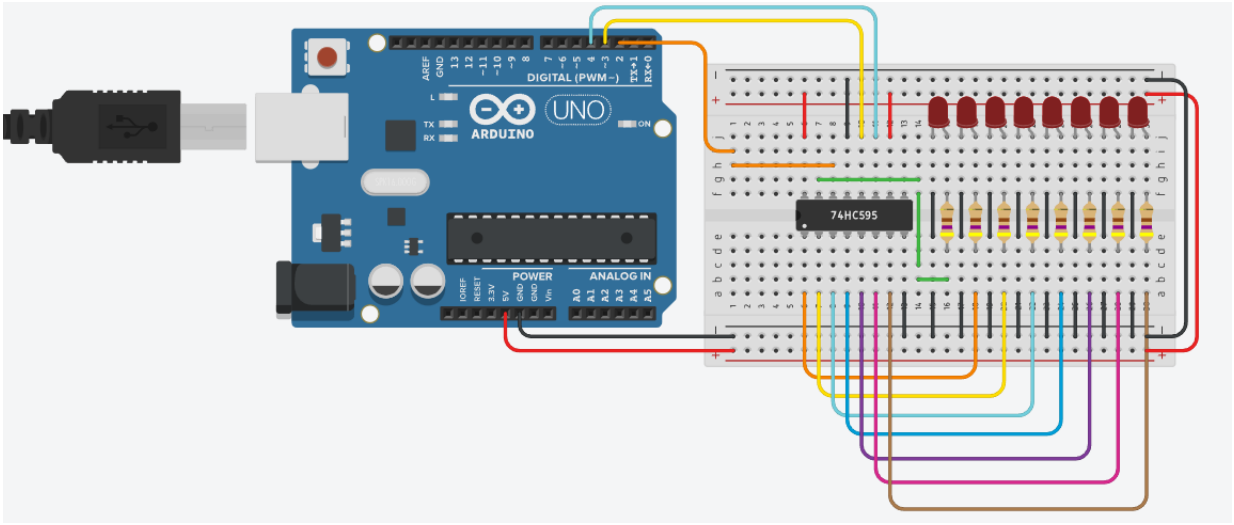

Figura 13: Circuito con integrado 74HC595 haciendo uso de un arduino.

Código:

```
//directivas de preprocesamiento
#define tiempo 2000
#define SER 2
#define RCLK 3
#define SRCLK 4


byte codigo[8]={10,255,0,49,50,6,7,8};//A byte stores an 8-bit unsigned
    number, from 0 to 255.
//-------------------------------------------------------------------
//prototipo de las funciones

//-------------------------------------------------------------------
//setup
void setup()
{
  for(int i=2;i<5;i++)
  {
    pinMode(i, OUTPUT);
  }

}
//-------------------------------------------------------------------
//loop
void loop()
{
  for(int i=0;i<8;i++)
  {
    //delay(25);
    for(int j=0;j<8;j++)
    {
      digitalWrite(SRCLK, LOW);
      digitalWrite(SER, bitRead(codigo[i], j));
```

```
33        digitalWrite(SRCLK, HIGH);
34      }
35      //delay(tiempo-25);
36      delay(tiempo);
37      digitalWrite(RCLK, HIGH);
38      digitalWrite(RCLK, LOW);
39
40    }
41
42
43
44  }
45  //-----------------------------------------------------------------
46  //cuerpo de las funciones
```

3. Solución propuesta.

Se plantea una solución en 3 ciclos. **Planeación, Ejecución, Verificación**
Paso 1 (Planeacion):
Se analiza el problema propuesto y se deducen los componentes a utilizar, adicional se busca el datasheet y se analiza el funcionamiento del integrado 74HC595 para ir bosquejando un circuito.
Paso 2 (Ejecución):
En esta fase ya se empezó a integrar los componentes al proyecto, luego de tener los componentes establecidos se procede con el aporte de ideas para la ejecuccion de los problemas planteados, de donde surgen las siguientes ideas:

El arreglo de datos a procesar se ingresará al código del arduino directamente. Se conectarán los dos arduinos, transmisor y receptor, el transmisor pasará los datos al integrado 74HC595 para que pasen de datos en serie a paralelo, estos datos paralelos serán los que entren a las compuertas lógicas para comprobar si son datos de interés o no. El transmisor también pasará los datos al arduino receptor y dependiendo el resultado en el sistema de desencriptación mostrará o no los datos en la pantalla LCD.

El sistema de desencriptación se modelará haciendo uso de integrados construidos con compuertas lógicas (xor, or, and, nand, xnor, nor, not) depende de la necesidad. Las compuertas con mayor posibilidad de ser usadas son la XOR, que según su tabla de verdad su salida es 0 cuando sus entradas son iguales y 1 cuando son diferentes. Otra compuerta sería la NOT, para negar las anteriores salidas y que me quede 1 cuando las entradas son iguales y cero cuando son diferentes. Por último se haría uso de compuertas AND para verificar que los bits de entrada sean iguales a la clave que me indica cuales son los datos reales a mostrar en pantalla. Este sistema de desencriptación, con un true/false, le indicará al arduino receptor la información que debe clasificar como el mensaje real enviado por el arduino emisor. Haciendo uso de dos puertos digitales, por uno ingresará el true o false (confirmación del mensaje real), y por el otro el byte correspondiente al mensaje real. Finalmente se mostrará la información por medio de un lcd (liquid crystal display)

Paso 3 (Verificación):
Ya por ultimo se ejecutará el programa verificando que todo este en orden y haciendo las llamadas pruebas de escritorio, con el fin de garantizar el funcionamiento optimo del programa, también se hizo una verificacion de componentes conectados.

4. Cibergrafía

https://www.electrogeekshop.com/como-funciona-el-74hc595-shift-register-y-su-interfaz-con-arduino/
https://github.com/trihedral/ArduinoLatexListing

https://www.youtube.com/watch?v=9gfZnNPBlVg