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. Introducción.

En el siguiente informe se verán los pasos realizados para la solución de un problema en específico, este problema es la desencriptación de un mensaje. El mensaje encriptado llegará en forma de arreglo y, mediante una clave, el programa identificará el mensaje real. Para la solución del problema se investigó el uso del integrado 74HC595, el cual nos permitirá paralelizar los datos que llegarán desde el arduino emisor, y de esta forma minimizar el uso de puertos seriales en el arduino receptor, además de permitirnos comprobar la llegada de la clave por medio del llamado "Bloque de desencriptacion", el cual está implementado con compuertas lógicas. El arduino receptor será el encargado de mostrar por medio de una pantalla LCD el mensaje desencriptado.

2. 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-uA (máximo) ICC.

2.1. 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.

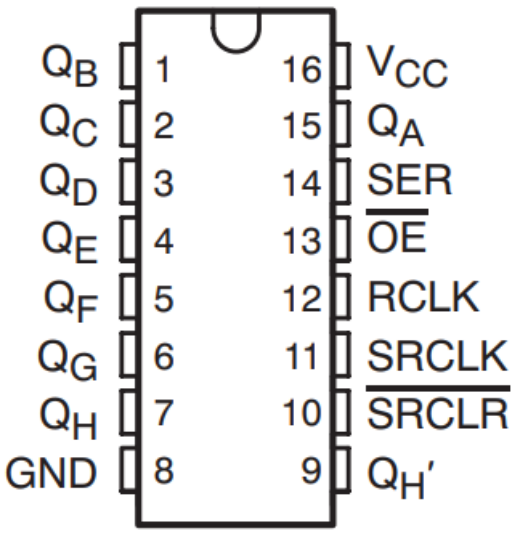

Figura 1: Integrado 74HC595.

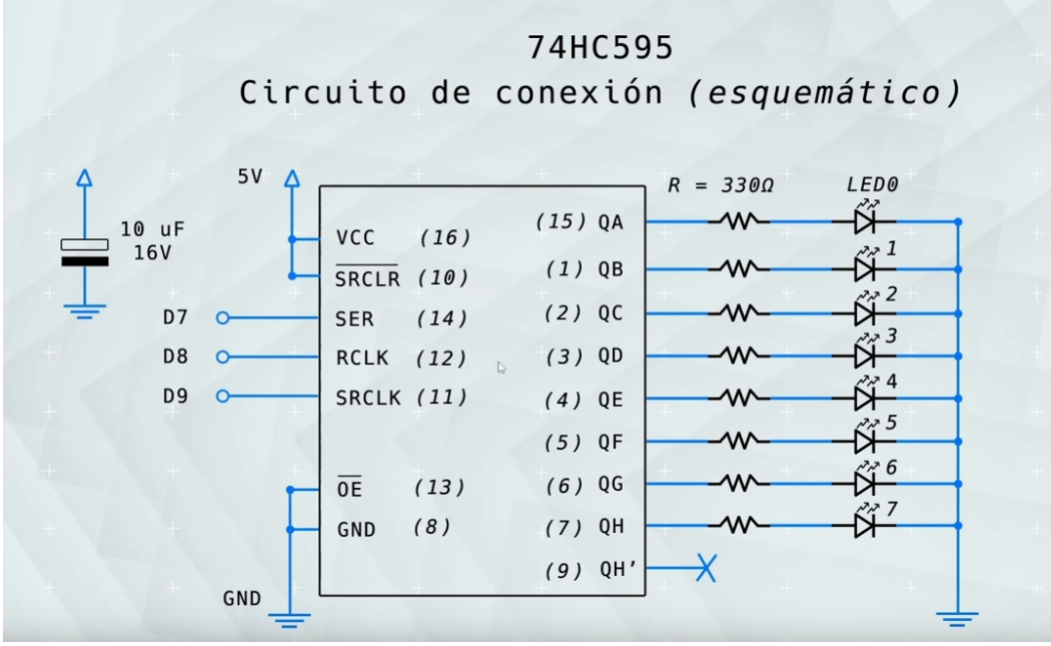

Figura 2: Circuito de conexión esquemático 74HC595.

Número pin	**Nombre del pin**	**Descripción**
1,2,3,4,5,6,7	Pines de salida (de Q1 a Q7)	Salidas del integrado
8	GND	Tierra
9	(P7) Salida en serie	Conexion de más de un 74hc595 en cascada
10	(MR) Restablecimiento maestro	Restablece todas las salidas como bajas.
11	(SH_CP) Reloj	Reloj que sincroniza la carga de datos.
12	(ST_CP) Pestillo	Se utiliza para actualizar los datos a los pines de salida.
13	(OE) Activación de salida	Se utiliza para desactivar las salidas.
14	(DS) Datos en serie	Este es el pin al que se envían los datos
15	(P0) salida	El primer pasador de salida.
16	Vcc	Este pin alimenta el IC

2.2. 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 3, 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).

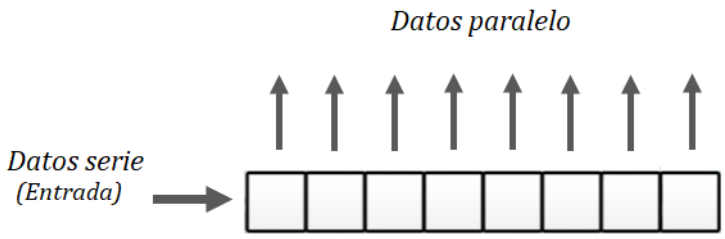

Figura 3: Representación del funcionamiento del circuito integrado 74HC595.

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 4

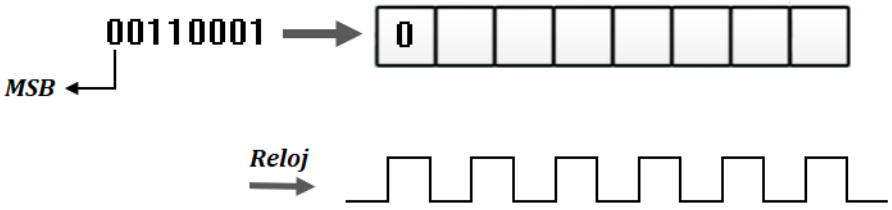

Figura 4: Entrada del MSB al integrado.

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.

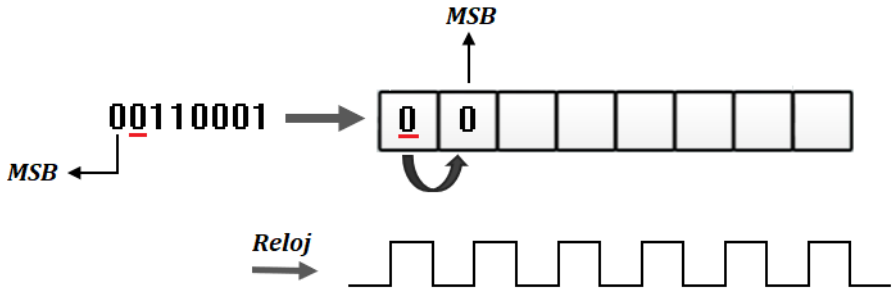

Figura 5: Desplazamiento de los bits dentro del integrado.

Este proceso se repetirá hasta que se ingrese al registro de desplazamiento el último bit del número (Figura 7).

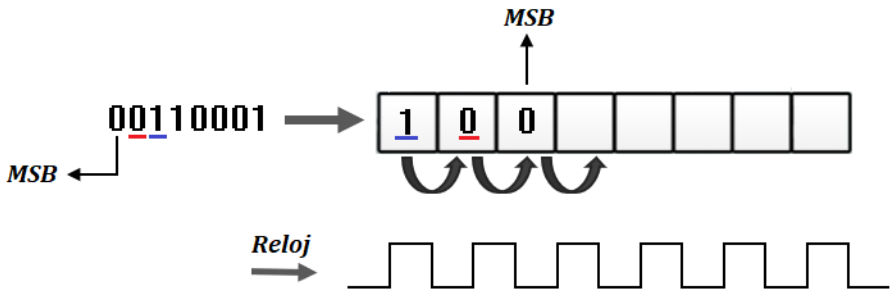

Figura 6: Desplazamiento de los bits dentro del integrado.

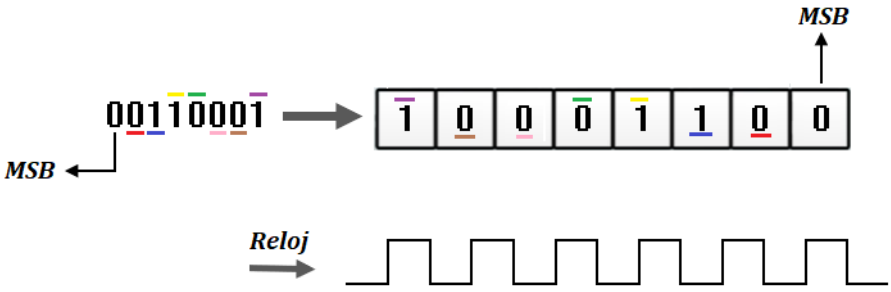

Figura 7: Registro de desplazamiento lleno. Se utilizaron colores encima o abajo de los bits para identificar fácilmente cuál es cuál en el registro de desplazamiento.

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 8

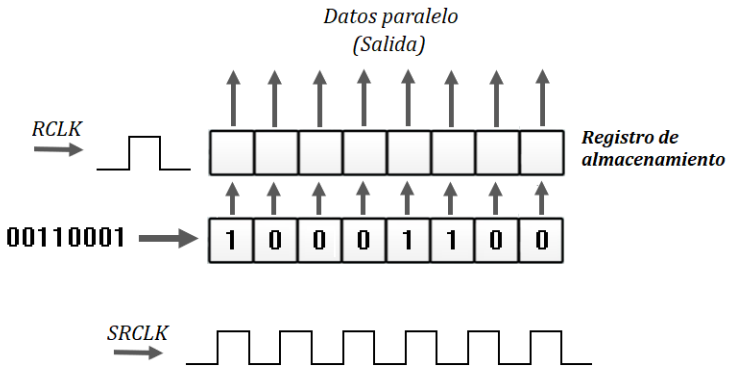

Figura 8: Representación del registro de almacenamiento.

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.

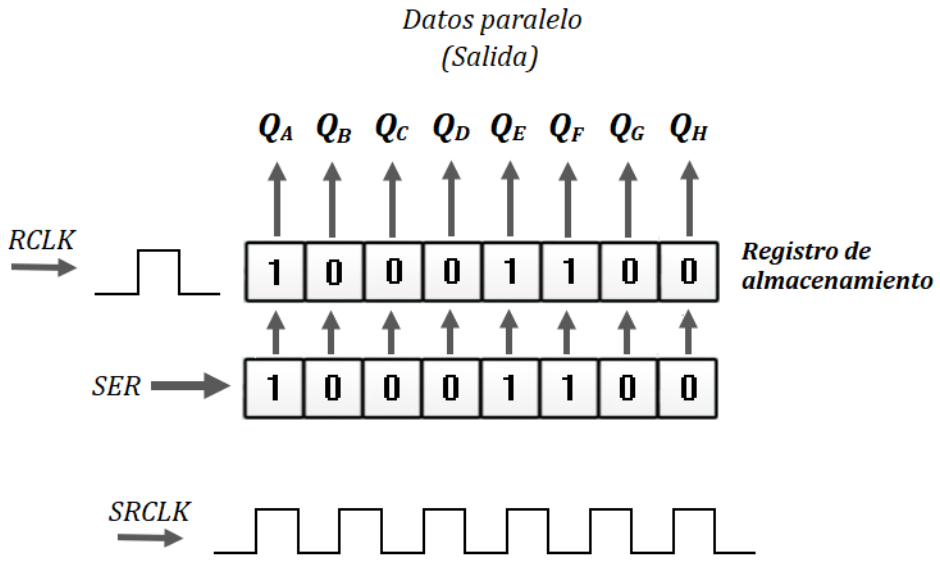

Figura 9: Representación del registro de almacenamiento.

2.3. Aplicaciones.

- Network switches
- Power infrastructure
- LED displays
- Servers

2.4. Uso del Circuito Integrado 74HC595.

- Con pulsadores:

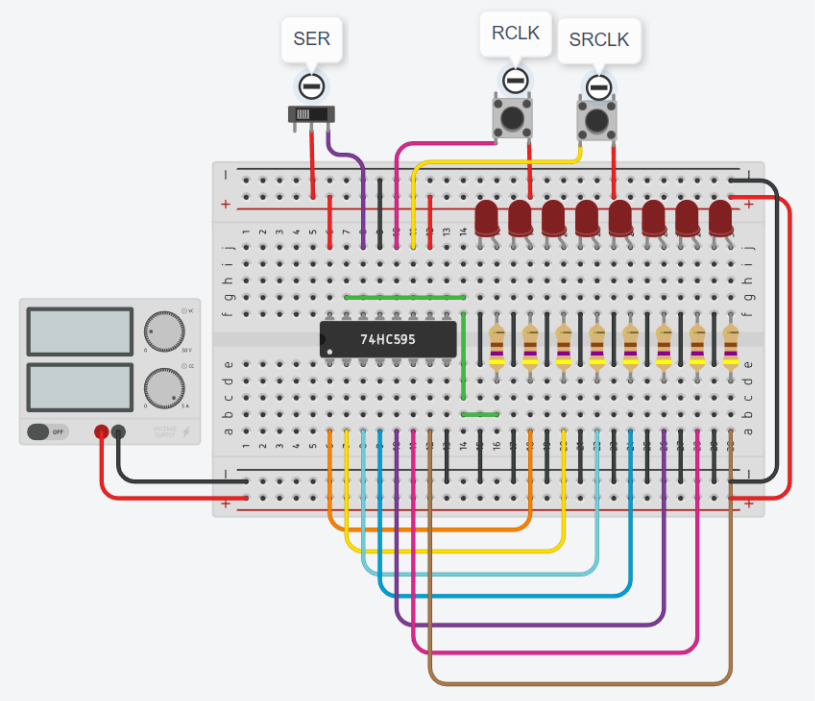

Figura 10: Circuito con integrado 74HC595 haciendo uso de pulsadores.

El montaje del circuito utilizando el 74HC595 realizado con pulsadores utiliza los conceptos vistos en la sección en donde se explica el funcionamiento del integrado. Se utiliza un interruptor deslizante para la entrada serial (SER), y dos pulsadores o switches que harán el trabajo de los relojes (SRCLK y RCLK). Si el interruptor deslizante está posicionado a la izquierda significa que entra un cero, si está posicionado a la derecha, es un uno. Para realizar la entrada de los datos al integrado, se desliza el interruptor SER hacia el lado del dato que necesito ingresar, después de esto se hace la activación del reloj del registro de desplazamiento (SRCLK) oprimiendo el switch con este nombre una vez, de esta forma se subirá el primer dato. Para el segundo dato se realizá el mismo procedimiento: se desliza el interruptor con el nombre SER hacia el lado del dato que requiere ingresar (izquierda cero, derecha uno) y posterior a esto se oprime una vez el interruptor llamado SRCLK. Este proceso se realiza hasta que ingrese los ocho datos que puede almacenar el integrado 74HC595, cuando termine de ingresar los datos se procede a oprimir el interruptor llamado RCLK, el cuál mandará los datos almacenados en el integrado hacia las salidas, por lo que el número ingresado se verá reflejado en los LEDs incluidos en el circuito.

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

- Con arduino:

Se define el uso de dos arduinos, montados con sus respectivas placa base, arduino emisor el cual será el encargado de transmitir la información y arduino receptor quien nos indicará al momento de recibir la información desencriptada. El uso de la función "bintodec"la cual esta incluida en el codigo, fue clave, ya que nos permite convertir un binario a un decimal y de esta forma será la manera en la que transmitiremos los datos entre los arduinos.
En el integrado se tiene un registro de 8 bits de dezplazamiento por cada salida, toma una

entrada en serie a través de un solo pin y lo convierte en salida paralela de 8 bits reduciendo así eficazmente el número de pines de interfaz entre un Microcontrolador y sus dispositivosde salida En el registro de desplazamiento tenemos una entrada en serie y datos en paralelos (salida), con cada pulso se comienza el progreso o ingreso de los datos en serie SER, reloj (SRCLK).
Los tres pines del registro de turnos que se necesitan para conectarse a Arduino son los pines 11 (la entrada del reloj), 12 (la entrada del reloj del registro de almacenamiento o simplemente la entrada del pestillo) y 14 (laentradade datos).

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.

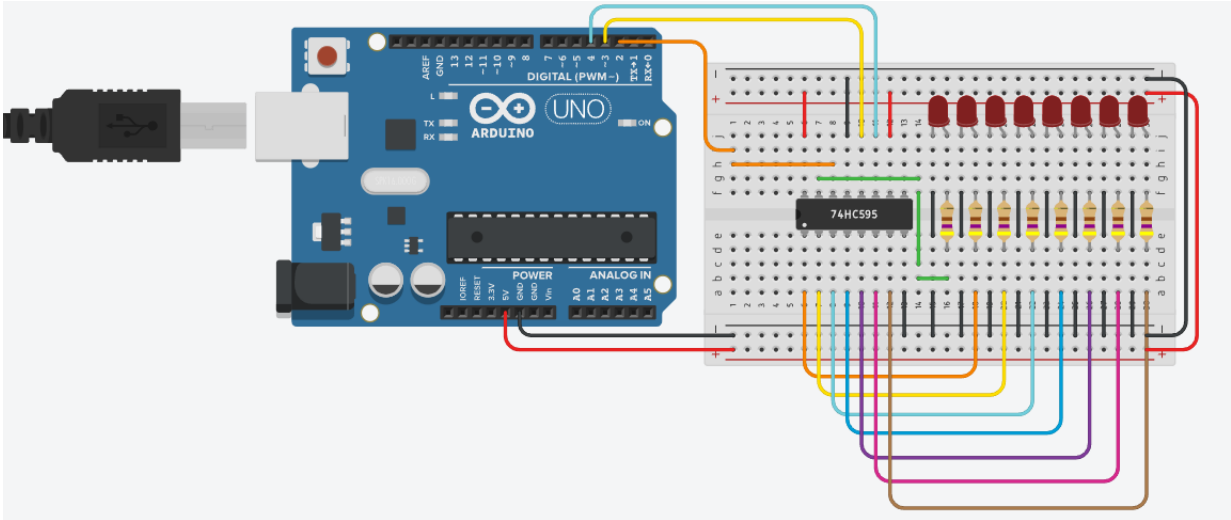

Figura 11: Circuito con integrado 74HC595 haciendo uso de un arduino.

```
//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);
  }

}
//-----------------------------------------------------------------
```

```
23 //loop
24 void loop()
25 {
26   for(int i=0;i<8;i++)
27   {
28     //delay(25);
29     for(int j=0;j<8;j++)
30     {
31       digitalWrite(SRCLK, LOW);
32       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. Comunicación entre Arduinos

3.1. Prueba 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.

3.1.1. Arduino emisor.

código:

```
1 #define tiempo 1000
2 byte codigo[8]={1,2,3,4,5,6,7,8};//A byte stores an 8-bit unsigned
     number, from 0 to 255.
3 //---------------------------------------------------------------
4 //prototipo de las funciones
5
6 //---------------------------------------------------------------
7 //setup
8 void setup()
9 {
10   for(int i=2;i<11;i++)
11   {
12     pinMode(i, OUTPUT);
```

```
  }

}
//--------------------------------------------------------------------
//loop
void loop()
{
  for(int i=0;i<8;i++)
  {
    digitalWrite(10, LOW);
    delay(25); // Wait for 1000 millisecond(s)
    for(int j=0;j<8;j++)
    {
      digitalWrite(j+2, bitRead(codigo[i], j));
    }
    delay(tiempo-25);
    digitalWrite(10, HIGH);
    delay(tiempo);
  }


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

montaje:

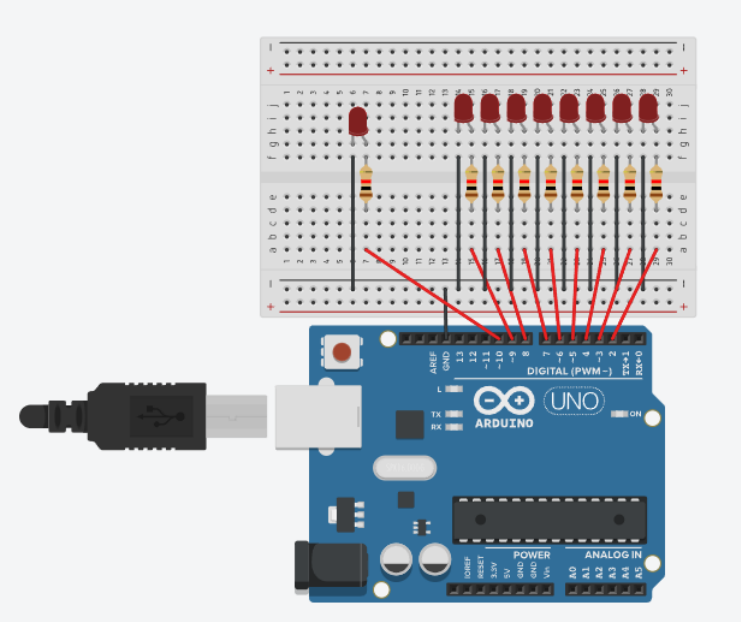

Figura 12: Prueba de emisión de bits y una señal de reloj por medio de 9 leds.

3.1.2. Conexión entre arduino emisor y receptor

montaje:

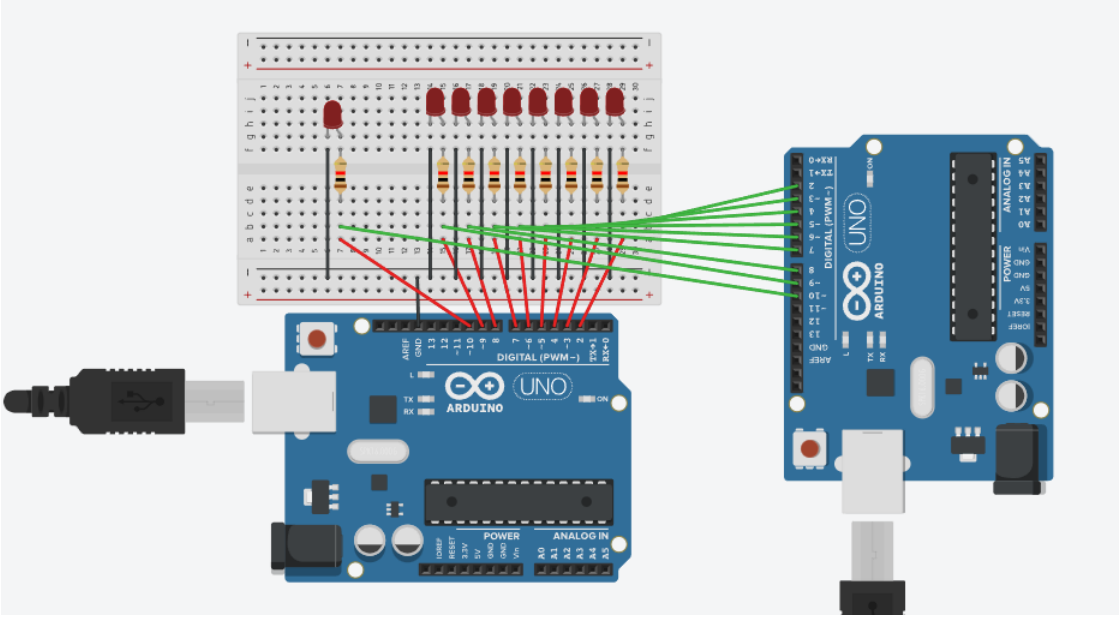

Figura 13: Prueba de emisión de bits y una señal de reloj por medio de 9 leds y su recepción a otro arduino.

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;
    }
  }

}
```

3.2. Prueba 2: Protocolo I2C

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:

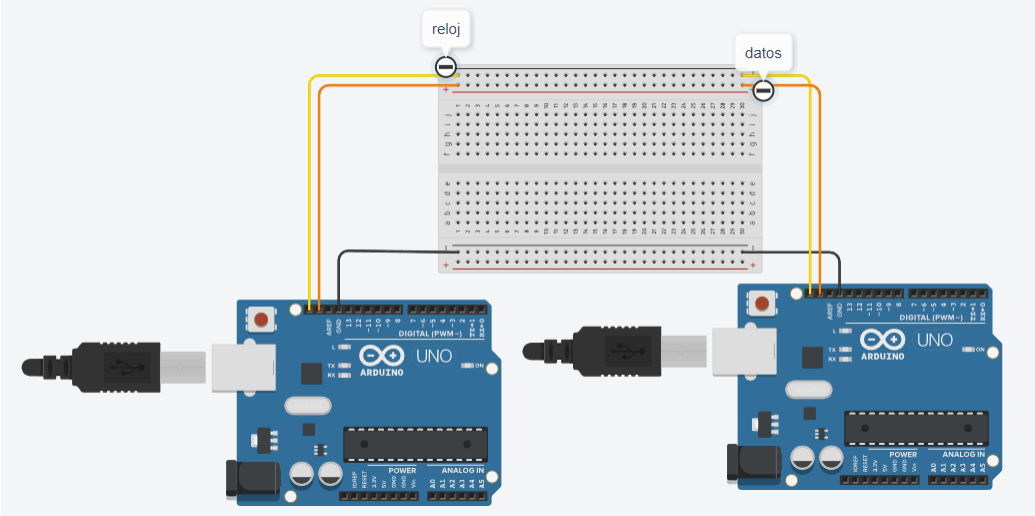

Figura 14: Comunicación mediante el protocolo I2C.

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 }
```

3.3. Prueba 3: Conexión entre arduinos usando puertos digitales

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, para probar así el proceso de emisión del arduino generador , y el proceso de recepción del segundo arduino, a la vez que verificamos con el integrado.
Montaje:

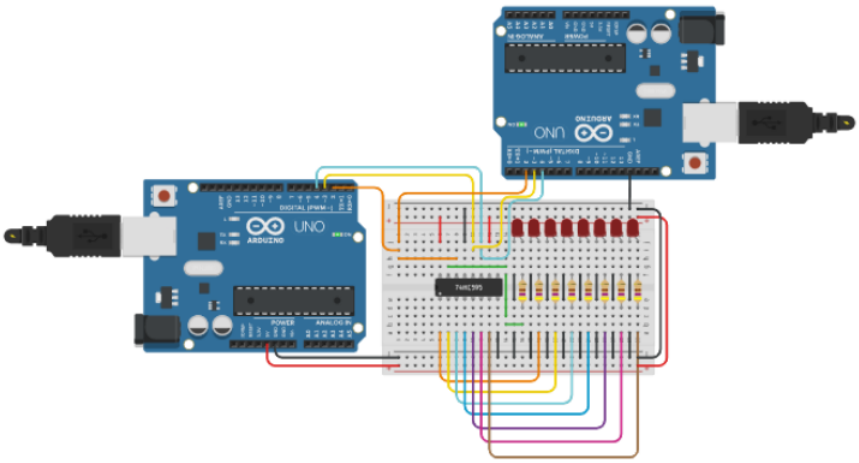

Figura 15: Comunicación entre arduinos, verificando con la ayuda de 8 leds y el integrado.

Lastimosamente, no se pudo codificar de forma correcta, por lo que el proceso de recepción falló.
Código emisor:

```
//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));
      digitalWrite(SRCLK, HIGH);
    }
    //delay(tiempo-25);
    delay(tiempo);
    digitalWrite(RCLK, HIGH);
    digitalWrite(RCLK, LOW);

  }



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

Código receptor:

```
#define SER 2
#define RCLK 3
#define SRCLK 4

void setup()
{
  Serial.begin(9600);
  for(int i=2;i<5;i++)
  {
    pinMode(i, INPUT);
  }

}

void loop()
{
  if(digitalRead(RCLK))
  {
    if(digitalRead(SRCLK))
    {
      for(int i=0;i<8;i++)
      {
        Serial.print(digitalRead(2));
      }
      Serial.println();
    }
  }
}
```

3.3.1. Conexión entre arduinos usando puertos digitales.

Tras varias horas de prueba se consigue una comunicación exitosa entre ambos arduinos, no solo mostrando los números en formato binario, sino también en décimal. (el montaje es el mismo del intento número 3).

Código emisor:

```
//directivas de preprocesamiento
#define tiempo 1000
#define SER 2
#define RCLK 3
#define SRCLK 4


byte codigo[8]={1,2,255,0,50,6,7,8};//A byte stores an 8-bit unsigned
    number, from 0 to 255.
int tamanoArray=sizeof(codigo)/sizeof(byte);
//-----------------------------------------------------------------
//prototipo de las funciones

//-----------------------------------------------------------------
//setup
void setup()
{
  for(int i=2;i<5;i++)
  {
    pinMode(i, OUTPUT);
    digitalWrite(i, LOW);
  }
}
//-----------------------------------------------------------------
//loop
void loop()
{
  for(int i=0;i<tamanoArray;i++)
  {
    for(int j=0;j<8;j++)
    {
      digitalWrite(SER, bitRead(codigo[i], j));
      digitalWrite(SRCLK, HIGH);
      delay(100);
      digitalWrite(SRCLK, LOW);

    }
    delay(tiempo);
    digitalWrite(RCLK, HIGH);
    digitalWrite(RCLK, LOW);

  }



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

Código receptor:

```
#define SER 2
#define RCLK 3
#define SRCLK 4
```

```
4
5  bool leerBit=true;
6  byte binario[8];
7
8  byte bintodec(byte* bin);
9  byte dos_exp (int exp);
10
11 void setup()
12 {
13   Serial.begin(9600);
14   for(int i=2;i<5;i++)
15   {
16     pinMode(i, INPUT);
17   }
18 }
19
20 void loop()
21 {
22
23   if(digitalRead(SRCLK))
24   {
25     for(int i=0;i<8;i++)
26     {
27       while(!digitalRead(SRCLK))
28       {
29
30       }
31       binario[i]=digitalRead(SER);
32       while(digitalRead(SRCLK))
33       {
34
35       }
36     }
37     delay(1000);
38     for(int i=7;i>=0;i--)
39     {
40       Serial.print(binario[i]);
41     }
42     Serial.println();
43     Serial.println(bintodec(binario));
44   }
45 }
46 //---------------------------------------------------------------
47 //funciones
48 byte bintodec(byte * bin){ // recibe un binario y retorna un decimal en
       tipo entero
49     byte dec = 0;
50     for(int i=7; i>=0 ; i--){
51         if (bin[i]>=1){
52                 dec = dec + (dos_exp(i));
53         }
54     }
55     return dec;
56 }
```

```
57
58 byte dos_exp (int exp){
59     int acu = 1;
60     for (int i=0 ; i<exp ; i++){
61         acu = acu*2;
62     }
63     return acu;
64 }
```

Salida por monitor serial:

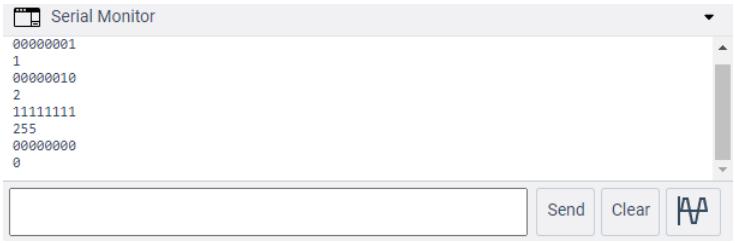

Figura 16: Salida monitor serial mostrando la recepción tanto en binario como en decimal.

3.3.2. Conexión entre arduinos: Pantalla LCD

Una vez funciona la comunicación entre arduinos, se procede a traspasar el codigo, de modo que el despliegue de información pase del monitor serial, como venía mostrándose, a un display lcd.

Montaje:

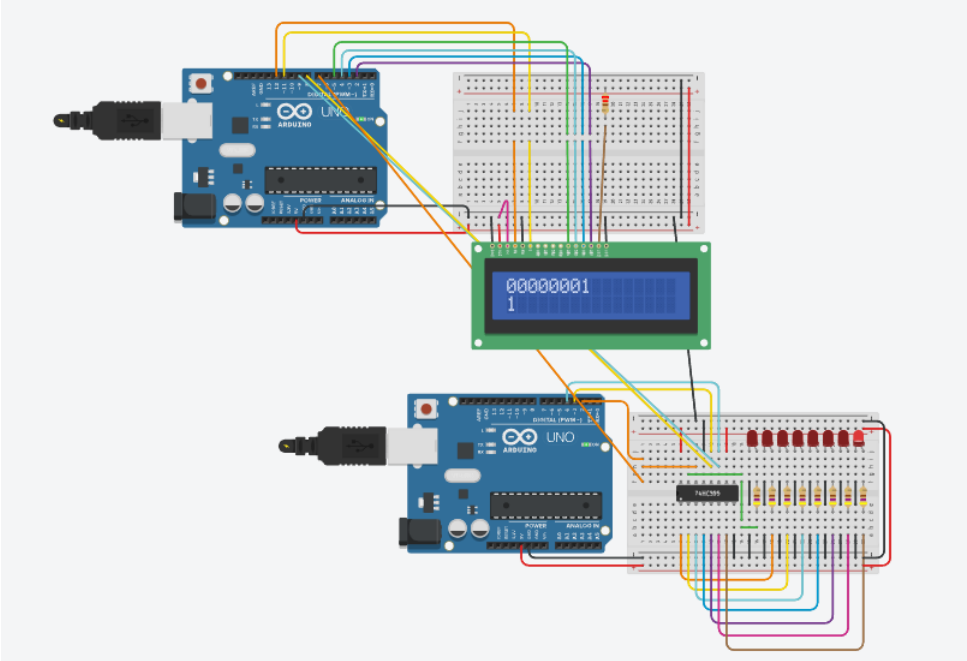

Figura 17: Conexión arduinos con lcd.

Código emisor:

```
//directivas de preprocesamiento
#define tiempo 1000
#define SER 2
#define RCLK 3
#define SRCLK 4


byte codigo[8]={1,2,255,0,50,6,7,8};//A byte stores an 8-bit unsigned
    number, from 0 to 255.
int tamanoArray=sizeof(codigo)/sizeof(byte);
//-----------------------------------------------------------------
//prototipo de las funciones

//-----------------------------------------------------------------
//setup
void setup()
{
  for(int i=2;i<5;i++)
  {
    pinMode(i, OUTPUT);
    digitalWrite(i, LOW);
  }
}
//-----------------------------------------------------------------
//loop
void loop()
{
  for(int i=0;i<tamanoArray;i++)
  {
    for(int j=0;j<8;j++)
    {
      digitalWrite(SER, bitRead(codigo[i], j));
      digitalWrite(SRCLK, HIGH);
      delay(100);
      digitalWrite(SRCLK, LOW);

    }
    delay(tiempo);
    digitalWrite(RCLK, HIGH);
    digitalWrite(RCLK, LOW);

  }



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

Código receptor:

```
// include the library code:
#include <LiquidCrystal.h>

```

```
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

#define SER 7
#define RCLK 8
#define SRCLK 9

byte bintodec(byte * bin);
byte binario[8];

byte dos_exp (int exp);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);

  for(int i=7;i<10;i++)
  {
    pinMode(i, INPUT);
  }
}

void loop() {

  if(digitalRead(SRCLK))
  {
    for(int i=0;i<8;i++)
    {
      while(!digitalRead(SRCLK))
      {

      }
      //Serial.print(digitalRead(2));
      binario[i]=digitalRead(SER);
      while(digitalRead(SRCLK))
      {

      }
    }
    delay(1000);
    lcd.clear();
    for(int i=7;i>=0;i--)
    {
      lcd.print(binario[i]);
    }
    lcd.setCursor(0, 1);
    lcd.print(bintodec(binario));


  }
  // print the number of seconds since reset:
  //lcd.print(millis() / 1000);
}

```

```
58
59 //funciones
60 byte bintodec(byte * bin){ // recibe un binario y retorna un decimal en
       tipo entero
61     byte dec = 0;
62     for(int i=7; i>=0 ; i--){
63         if (bin[i]>=1){
64                 dec = dec + (dos_exp(i));
65         }
66     }
67     return dec;
68 }
69
70 byte dos_exp (int exp){
71     int acu = 1;
72     for (int i=0 ; i<exp ; i++){
73         acu = acu*2;
74     }
75     return acu;
76 }
```

4. Bloque de desencriptación.

La tarea principal pedida en este proyecto es la desencriptación de cierta información en forma de arreglo que va a llegar para ser procesada, este proceso de desencriptación se quiere realizar por medio de hardware para así aliviar un poco la carga del programa que realiza la recepción de los datos y la conexión entre los arduinos. Se hará uso de compuertas lógicas como XOR, AND y NOT, según sea el caso. La salida de este bloque de desencriptación será un uno (true) o un cero (false), lo que nos dirá si el número ingresado a las compuertas lógicas es la clave que estamos esperando.

4.1. Prueba 1: Integrado 74HC08 y 74HC21

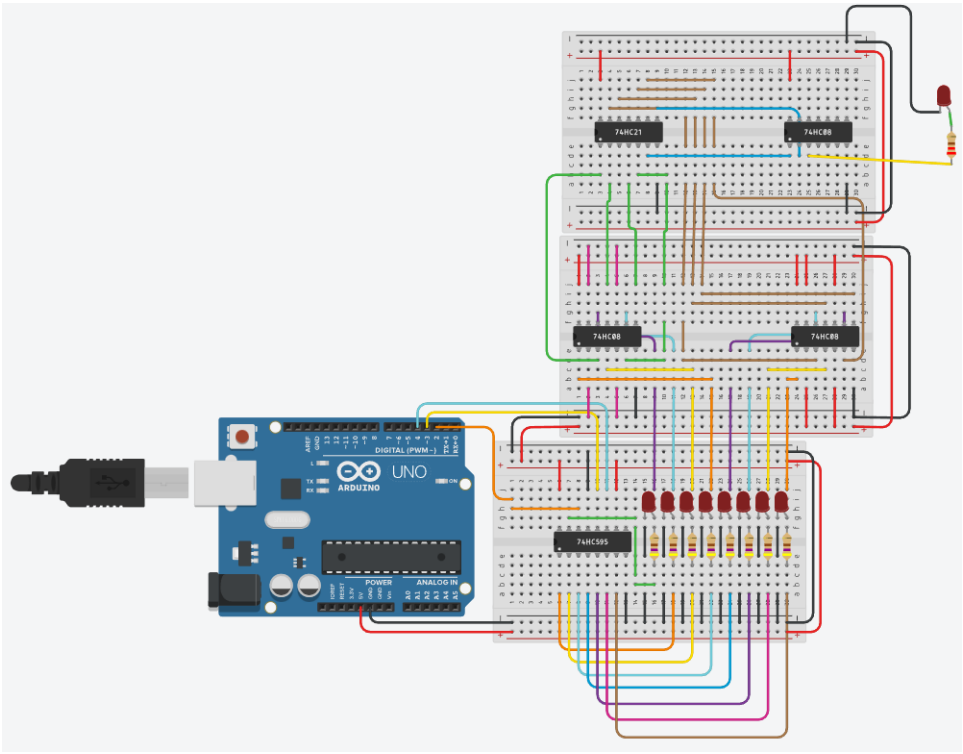

Figura 18: Bloque de desencriptación hecho con compuertas AND.

Para comprobar el funcionamiento del sistema de desencriptación se tomó la clave asignada al grupo y se ingresó en el arreglo de tipo byte anteriormente mostrado en los códigos de conexión entre arduino. Se llegó a la conclusión, gracias a la tabla de verdad de la compuerta AND, que esta no era la indicada para la comparación de números que se requiere para que el sistema de desencriptación realice un trabajo exitoso.

Es necesario el uso de compuertas lógicas XOR que estén conectadas con compuertas lógicas NOT, las cuales nos dirán si el número binario que sale por el integrado 74HC595 de forma paralela es nuestra clave de desencriptación.

4.2. Prueba 2: Integrado 74HC86(XOR), 74HC04(NOT) 74HC21(AND4INPUTS), 74HC08(AND)

Tras descubrir que la implementación de solo compuertas AND no era adecuada, se dibujó el circuito de compuertas lógicas para así saber claramente qué compuertas eran necesarias. El dibujo se presenta a continuación:

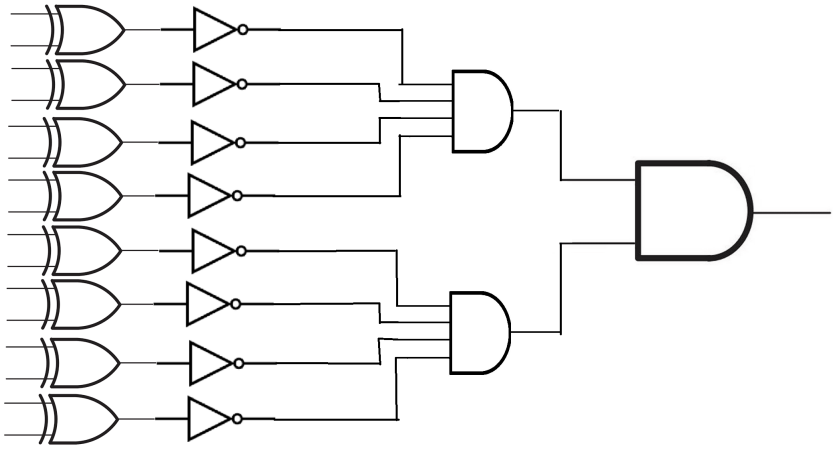

Figura 19: Conexión de las compuertas necesarias para la comparación de números.

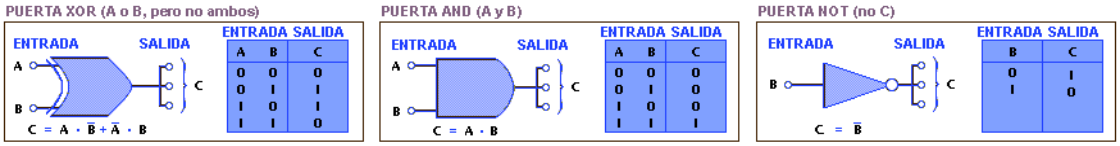

Figura 20: Tablas de verdad de las compuertas XOR, AND y NOT.

Las compuertas que recibirán la información bit a bit desde el circuito integrado 74HC6595 son las XOR, las cuales teniendo en cuenta su tabla de verdad Figura 20 nos arrojarán 1 cuando los datos sean diferentes y 0 cuando sean iguales, es por esto que a la salida de estas compuertas se hará necesario el uso de compuertas negadoras, para que cuando nuestros datos sean iguales obtengamos un uno y cuando sean diferentes obtengamos un cero. Estas salidas que obtenemos de las compuertas negadoras pasarán a las compuertas AND de cuatro entradas, esto porque necesitaremos saber si todas las salidas de las compuertas son uno, lo que confirmaría que el número ingresado a las compuertas XOR (integrado 74HC86) es la clave que estamos buscando. Por último, obtendremos dos salidas de las compuertas descritas anteriormente, estas dos salidas deberán entrar en otra AND de dos entradas la que nos mostrará uno (true) cuando sus dos datos de entrada sean 1, y cero (false) en cualquier otro caso.

Después del análisis anterior se procedió al montaje con las compuertas en Tinkercad. Como podemos ver, se tomaron las salidas del integrado 74HC595 para ser comparadas en las compuertas XOR con la clave que necesitamos encontrar. La clave se ingresa en forma de binario con ayuda de GND (0) y Vcc (1), y se compara con las salidas del integrado 74HC595. Figura 21

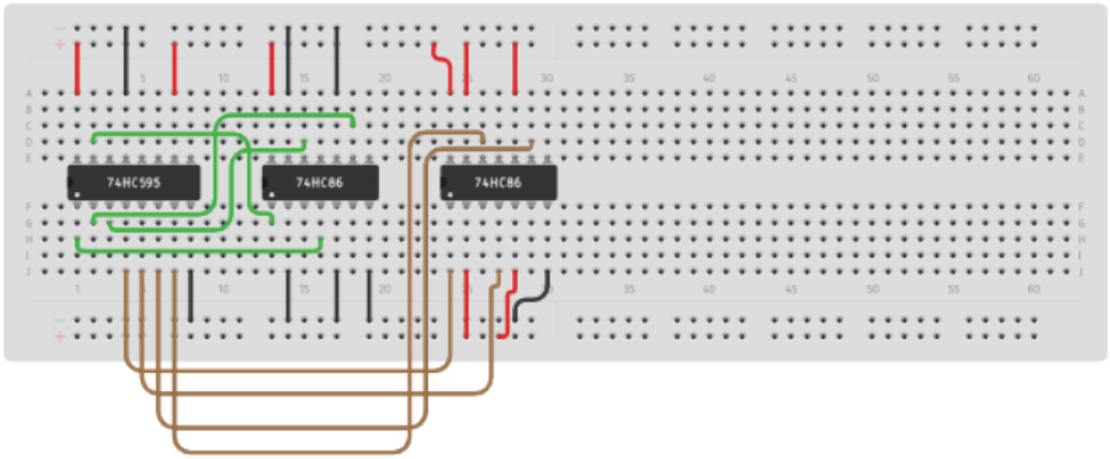

Figura 21: Inicio del montaje de desencriptación.

Las salidas de las compuertas XOR pasan a los integrados 74HC04 para ser negadas, esto por la explicación que se dio anteriormente, y luego entran a las compuertas AND de cuatro entradas (74HC21) para comprobar que todas la comparaciones hayan sido verdaderas (entradas igual uno). Por último, se comprobará que las salidas del integrado 74HC21 sean verdaderas (uno), y de esta forma sabremos si el número ingresado a los integrados es la clave que estamos buscando.

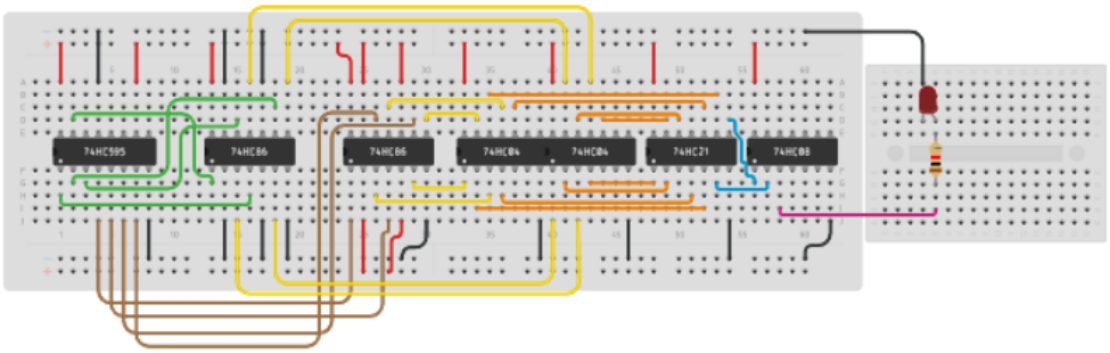

Figura 22: Montaje de desencriptación finalizado.

La verificación del funcionamiento de este sistema de comparación construido a partir de compuertas lógicas se realizó con dos pulsadores, un switch y una fuente de voltaje de 5v:

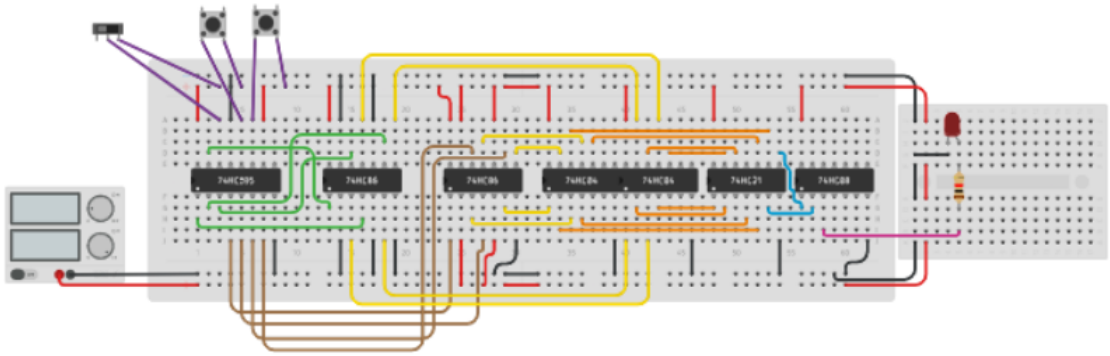

Figura 23: Montaje de desencriptación finalizado con pulsadores y fuente.

Afortunadamente, el circuito respondió tal como se esperaba, dando por finalizada la tarea de construir, en este caso, un desencriptador exitoso.

5. Implementación sistema completo.

A continuación se presenta la integración del bloque de desencriptación y la comunicación entre arduinos:

5.1. Implementación sistema completo con salida en monitor serial

Inicialmente se optó por implementarlo mostrando la salida por medio del monitor serial, para así poder confirmar su funcionamiento sin el riesgo que el error se encuentre en la conexión de los cables al lcd.

montaje:

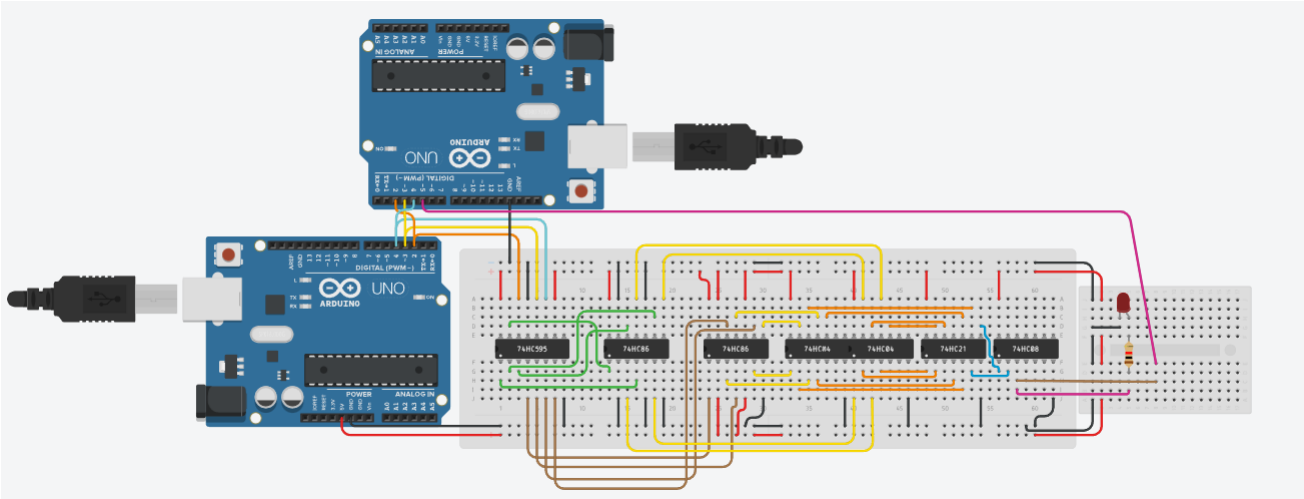

Figura 24: Integración del bloque de desencriptación, mostrando la información mediante el monitor serial

Código emisor:

```
1 //directivas de preprocesamiento
2 #define TIEMPO 1000
3 #define SER 2
```

```
#define RCLK 3
#define SRCLK 4


byte codigo[30];
//A byte stores an 8-bit unsigned number, from 0 to 255.
int tamanoArray=sizeof(codigo)/sizeof(byte);
//-----------------------------------------------------------------
//prototipo de las funciones

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

  //ensayo
  for(int i=0;i<30;i++)
  {
    codigo[i]=i+14;
  }
}
//-----------------------------------------------------------------
//loop
void loop()
{
  for(int i=0;i<tamanoArray;i++)
  {
    for(int j=0;j<8;j++)
    {
      digitalWrite(SER, bitRead(codigo[i], j));
      digitalWrite(SRCLK, HIGH);
      delay(100);
      digitalWrite(SRCLK, LOW);

    }
    delay(TIEMPO);
    digitalWrite(RCLK, HIGH);
    digitalWrite(RCLK, LOW);

  }



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

Código receptor:

```
#define SER 2
```

```
#define RCLK 3
#define SRCLK 4
#define TIEMPO 1000
#define CLAVE 5

//byte codigo[]={1,2,255,0,15,6,7,8};
//byte codigo[]={15,15,255,0,15,15,7,8};
byte binario[8];

byte bintodec(byte* bin);
byte dos_exp (int exp);

bool contar=false;
int posicion=-1;

void setup()
{
  Serial.begin(9600);
  for(int i=2;i<6;i++)
  {
    pinMode(i, INPUT);
  }
}

void loop()
{

  if(digitalRead(SRCLK))
  {
    for(int i=0;i<8;i++)
    {
      while(!digitalRead(SRCLK))
      {

      }
      binario[i]=digitalRead(SER);
      while(digitalRead(SRCLK))
      {

      }
    }

    delay(TIEMPO);
    for(int i=7;i>=0;i--)
    {
      Serial.print(binario[i]);
    }
    Serial.println();
    Serial.println(bintodec(binario));

    if(digitalRead(CLAVE))
    {
      contar=true;
      Serial.println("se leyo la clave");
```

```
56      }
57
58      if(contar)
59      {
60        posicion++;
61        Serial.print("posicion=␣");
62        Serial.println(posicion);
63      }
64      if(posicion==3)
65      {
66        Serial.println("hola");
67        contar=false;
68        posicion=-1;
69      }
70    }
71  }
72  //---------------------------------------------------------------
73  //funciones
74  byte bintodec(byte * bin){ // recibe un binario y retorna un decimal en
        tipo entero
75      byte dec = 0;
76      for(int i=7; i>=0 ; i--){
77          if (bin[i]>=1){
78                  dec = dec + (dos_exp(i));
79          }
80      }
81      return dec;
82  }
83
84  byte dos_exp (int exp){
85      int acu = 1;
86      for (int i=0 ; i<exp ; i++){
87          acu = acu*2;
88      }
89      return acu;
90  }
```

5.2. Implementación sistema completo con salida en monitor serial y LCD

Una vez se confirmó que no había ningún error, se procedió a implementarlo tanto en monitor serial como en el display lcd, para así poder verificar que la salida del serial coincida con la mostrada en el lcd.

montaje:

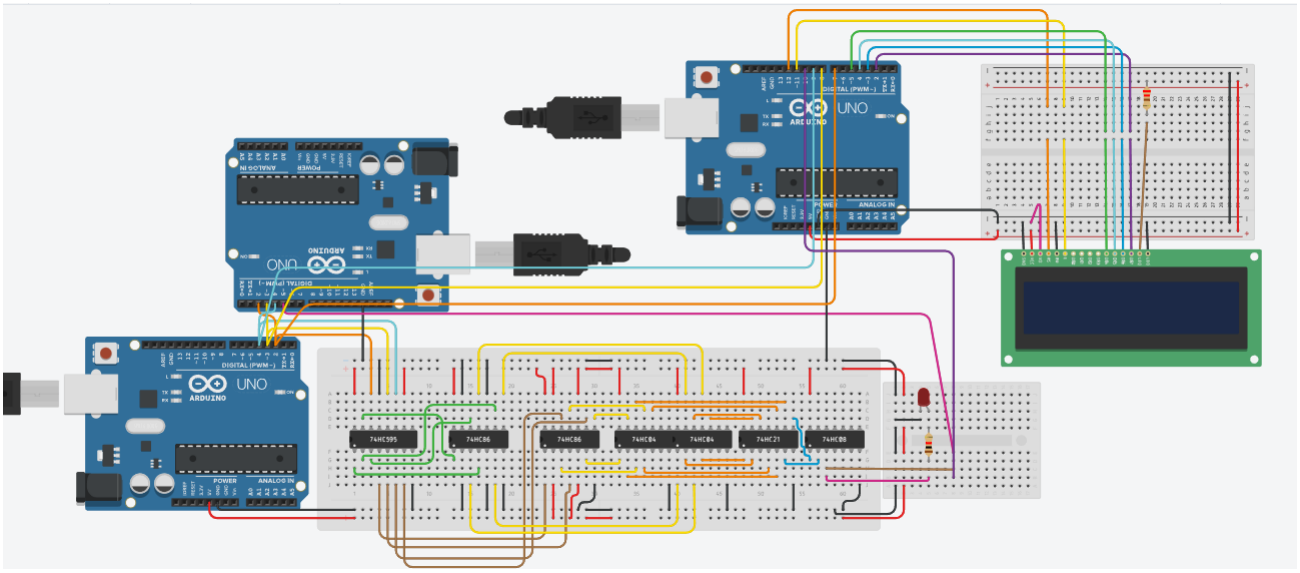

Figura 25: Integración del bloque de desencriptación, mostrando la información mediante el monitor serial y el lcd

Código emisor:

```
//directivas de preprocesamiento
#define TIEMPO 1000
#define SER 2
#define RCLK 3
#define SRCLK 4


byte codigo[30];
//A byte stores an 8-bit unsigned number, from 0 to 255.
int tamanoArray=sizeof(codigo)/sizeof(byte);
//-----------------------------------------------------------------
//prototipo de las funciones

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

  //ensayo
```

```
  for(int i=0;i<30;i++)
  {
    codigo[i]=i+14;
  }
}
//--------------------------------------------------------------------
//loop
void loop()
{
  for(int i=0;i<tamanoArray;i++)
  {
    for(int j=0;j<8;j++)
    {
      digitalWrite(SER, bitRead(codigo[i], j));
      digitalWrite(SRCLK, HIGH);
      delay(100);
      digitalWrite(SRCLK, LOW);

    }
    delay(TIEMPO);
    digitalWrite(RCLK, HIGH);
    digitalWrite(RCLK, LOW);

  }



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

Código receptor serial:

```
#define SER 2
#define RCLK 3
#define SRCLK 4
#define TIEMPO 1000
#define CLAVE 5

//byte codigo[]={1,2,255,0,15,6,7,8};
//byte codigo[]={15,15,255,0,15,15,7,8};
byte binario[8];

byte bintodec(byte* bin);
byte dos_exp (int exp);

bool contar=false;
int posicion=-1;

void setup()
{
  Serial.begin(9600);
  for(int i=2;i<6;i++)
  {
    pinMode(i, INPUT);
```

```
  }
}

void loop()
{

  if(digitalRead(SRCLK))
  {
    for(int i=0;i<8;i++)
    {
      while(!digitalRead(SRCLK))
      {

      }
      binario[i]=digitalRead(SER);
      while(digitalRead(SRCLK))
      {

      }
    }

    delay(TIEMPO);
    for(int i=7;i>=0;i--)
    {
      Serial.print(binario[i]);
    }
    Serial.println();
    Serial.println(bintodec(binario));

    if(digitalRead(CLAVE))
    {
      contar=true;
      Serial.println("se leyo la clave");
    }

    if(contar)
    {
      posicion++;
      Serial.print("posicion= ");
      Serial.println(posicion);
    }
    if(posicion==3)
    {
      Serial.println("hola");
      contar=false;
      posicion=-1;
    }
  }
}
//--------------------------------------------------------------
//funciones
byte bintodec(byte * bin){ // recibe un binario y retorna un decimal en
    tipo entero
    byte dec = 0;
```

```
    for(int i=7; i>=0 ; i--){
        if (bin[i]>=1){
                dec = dec + (dos_exp(i));
        }
    }
    return dec;
}

byte dos_exp (int exp){
    int acu = 1;
    for (int i=0 ; i<exp ; i++){
        acu = acu*2;
    }
    return acu;
}
```

Código receptor lcd:

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

#define SER 7
#define RCLK 8
#define SRCLK 9
#define CLAVE 10
#define TIEMPO 1000

byte bintodec(byte * bin);
byte binario[8];

byte dos_exp (int exp);

bool contar=false;
int posicion=-1;

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);

  for(int i=7;i<11;i++)
  {
    pinMode(i, INPUT);
  }
}

void loop() {

  if(digitalRead(SRCLK))
  {
    for(int i=0;i<8;i++)
    {
      while(!digitalRead(SRCLK))
```

```
      {

      }
      //Serial.print(digitalRead(2));
      binario[i]=digitalRead(SER);
      while(digitalRead(SRCLK))
      {

      }
    }
    delay(TIEMPO);
    lcd.clear();
    for(int i=7;i>=0;i--)
    {
      lcd.print(binario[i]);
    }
    lcd.setCursor(0, 1);
    lcd.print(bintodec(binario));

  if(digitalRead(CLAVE))
    {
      contar=true;
      lcd.print("se leyo la clave");
    }

    if(contar)
    {
      posicion++;
      lcd.print("posicion= ");
      lcd.print(posicion);
    }
    if(posicion==3)
    {
      lcd.print("hola");
      contar=false;
      posicion=-1;
    }

  }

}


//funciones
byte bintodec(byte * bin){ // recibe un binario y retorna un decimal en
    tipo entero
    byte dec = 0;
    for(int i=7; i>=0 ; i--){
        if (bin[i]>=1){
                dec = dec + (dos_exp(i));
        }
    }
    return dec;
}
```

```
91
92 byte dos_exp (int exp){
93     int acu = 1;
94     for (int i=0 ; i<exp ; i++){
95         acu = acu*2;
96     }
97     return acu;
98 }
```

El sistema se comportó tal como se esperaba tanto en el monitor serial como en el display LCD.

5.3. Implementación sistema completo con salida en display LCD(SISTEMA FINALIZADO)

Ya que el montaje del display funcionó correctamente, se procedió a eliminar el arduino encargado del monitor serial. Y para que el proyecto tenga un mejor acabado, se procedió a segmentar el código, y comentarlo de una forma más estructurada, a su vez, eliminando todo lo no necesario para el funcionamiento correcto del sistema:

montaje:

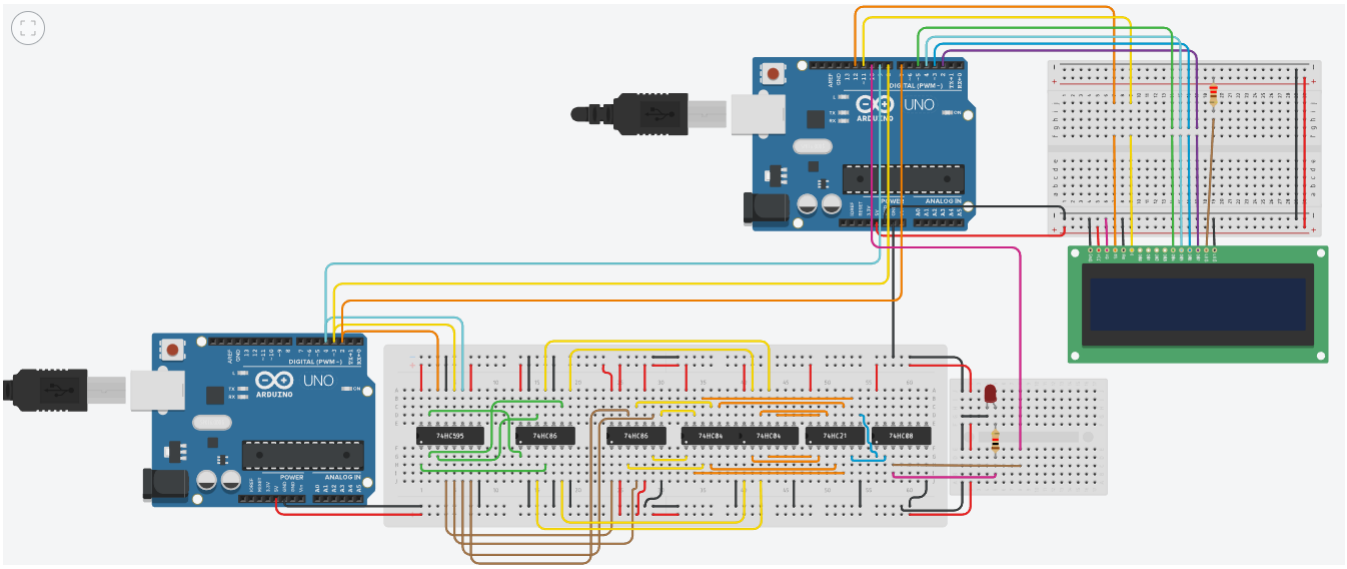

Figura 26: Integración del bloque de desencriptación, mostrando la información mediante el display lcd.

Código emisor:

```
1  //DIRECTIVAS DE PREPROCESAMIENTO
2
3  #define TIEMPO 500
4  #define SER 2
5  #define RCLK 3
6  #define SRCLK 4
7  #define TAMANO 60
8
9  //-----------------------------------------------------------------
10 //-----------------------------------------------------------------
11 //ARREGLOS DE PRUEBA (descomente uno solo a la vez):
12
```

```
13 byte codigo[]={15,15,255,0,15,15,7,8,15,9,2,15};
14 //byte codigo[]={1,2,255,0,15,6,7,8};
15
16 //En caso de querer crear un arreglo mediante un ciclo for:
17 //byte codigo[TAMANO];
18 //1.Cambie el tamano del arreglo en la directiva de
19 //preprocesamiento llamada TAMANO en la linea: 7.
20 //2.Quite los comentarios del for ubicado en la linea: 45
21 //y modifique como funcionar  este para llenar el arreglo.
22
23 // Por  qu  el tipo byte?
24 //A byte stores an 8-bit unsigned number, from 0 to 255.
25
26 //-----------------------------------------------------------------
27 //-----------------------------------------------------------------
28 //VARIABLES:
29
30 int tamanoArray=sizeof(codigo)/sizeof(byte);
31
32 //-----------------------------------------------------------------
33 //-----------------------------------------------------------------
34 //setup
35
36 void setup()
37 {
38   //configuramos los puertos de salida y los inicializamos en 0
39   for(int i=2;i<5;i++)
40   {
41     pinMode(i, OUTPUT);
42     digitalWrite(i, LOW);
43   }
44
45   //ARREGLO MODIFICABLE:
46   /*for(int i=0;i<TAMANO;i++)
47   {
48     if(i<=30)
49     {
50       codigo[i]=i+14;
51     }
52     else if(i>32)
53     {
54       codigo[i]=i+100;
55     }
56     else
57     {
58       codigo[i]=15;
59     }
60   }*/
61 }
62
63 //-----------------------------------------------------------------
64 //-----------------------------------------------------------------
65 //loop
66
```

```
67 void loop()
68 {
69   //recorremos cada elemento
70   for(int i=0;i<tamanoArray;i++)
71   {
72     //obtenemos los 8 bits
73     for(int j=0;j<8;j++)
74     {
75       digitalWrite(SER, bitRead(codigo[i], j));//j=0 menos signi
76       digitalWrite(SRCLK, HIGH);
77       delay(100);
78       digitalWrite(SRCLK, LOW);
79     }
80     delay(TIEMPO);
81     digitalWrite(RCLK, HIGH);
82     digitalWrite(RCLK, LOW);
83   }
84 }
```

Código receptor:

```
1  //LIBRERIAS Y SU INICIALIZACION:
2
3  #include <LiquidCrystal.h>
4  // initialize the library with the numbers of the interface pins
5  LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
6
7  //-------------------------------------------------------------------
8  //-------------------------------------------------------------------
9  //DIRECTIVAS DE PREPROCESAMIENTO:
10
11 #define SER 7
12 #define RCLK 8
13 #define SRCLK 9
14 #define CLAVE 10
15 #define TIEMPO 500
16
17 //-------------------------------------------------------------------
18 //-------------------------------------------------------------------
19 //PROTOTIPO DE LAS FUNCIONES:
20
21 byte bintodec(byte * bin);
22 byte dos_exp (int exp);
23
24 //-------------------------------------------------------------------
25 //-------------------------------------------------------------------
26 //VARIABLES Y ARREGLOS:
27
28 bool contar=false;
29 int posicion=0;
30 byte binario[8];
31
32 //-------------------------------------------------------------------
33 //-------------------------------------------------------------------
34 //setup
```

```
35
36 void setup()
37 {
38   // set up the LCD's number of columns and rows:
39   lcd.begin(16, 2);
40   //configuramos los puertos de entrada
41   for(int i=7;i<11;i++)
42   {
43     pinMode(i, INPUT);
44   }
45 }
46
47 //-----------------------------------------------------------------
48 //-----------------------------------------------------------------
49 //loop
50
51 void loop()
52 {
53   //si se envia un digito
54   if(digitalRead(SRCLK))
55   {
56     //verifica si se recibe la clave
57     if(digitalRead(CLAVE))
58     {
59       contar=true;
60     }
61     //guarda los 8 bits recibidos
62     for(int i=0;i<8;i++)
63     {
64       while(!digitalRead(SRCLK))
65       {
66
67       }
68       binario[i]=digitalRead(SER);
69       while(digitalRead(SRCLK))
70       {
71
72       }
73     }
74     delay(TIEMPO);
75   //si recibio la clave, comienza a contar
76     if(contar)
77     {
78       posicion++;
79     }
80     //si van 15 posiciones luego de recibir la clave
81     if(posicion==15)
82     {
83       lcd.clear();
84       lcd.print(bintodec(binario));//muestra el numero secreto
85       //lcd.print("-");
86       contar=false;
87       posicion=0;
88     }
```

```
  }
}

//-------------------------------------------------------------
//-------------------------------------------------------------
//FUNCIONES:

// recibe un binario y retorna un decimal en tipo entero
byte bintodec(byte * bin){
    byte dec = 0;
    for(int i=7; i>=0 ; i--){
        if (bin[i]>=1){
                dec = dec + (dos_exp(i));
        }
    }
    return dec;
}

byte dos_exp (int exp){
    int acu = 1;
    for (int i=0 ; i<exp ; i++){
        acu = acu*2;
    }
    return acu;
}
```

El sistema completo funciona correctamente y puede ser probado haciendo uso del siguiente link:
https://www.tinkercad.com/things/6DJhzwBksef

6. Conclusiones

- El uso del integrado 74HC595 fue clave para la solución del problema, pues nos permitió paralelizar los datos que llegaban en serie, de esta forma se minimizó el uso de puertos digitales en el arduino receptor y se ingresó el número a desencriptar al bloque de desencriptación para ser comparado con la clave asignada.
- Para lograr una conexión entre arduinos exitosa se necesitó comprender el uso del circuito integrado 74HC595, ya que la sincronización entre el arduino emisor y receptor se logró gracias al SRLCK (Reloj de registro de desplazamiento), y al RCLK (Reloj de registro de salida).

7. 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