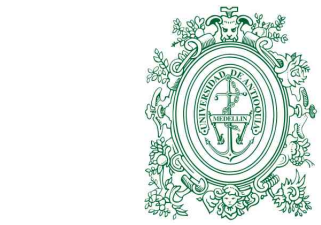

Universidad de Antioquia

INFORMATICA II

Informe de Implementación Parcial 1

Autores:
Ardila Pérez Valentina – 1017253532
Arroyo Estrada Jorge Sebastián – 1193482707

3 de marzo de 2022

Resumen:

Se comunicarán dos Arduinos por vía serial enviando una trama de datos encriptada, utilizando hardware para hacer validaciones y se mostrará en el Arduino receptor los datos desencriptados por medio de un LCD. Se usan compuertas lógicas y chip 74HC595.

Introducción:

En este trabajo se realizará una comunicación de dos Arduinos por vía serial, para realizar esta comunicación se hará uso de un chip llamado 74HC595 y varias compuertas que ayudan a comprobar la información, ya que los datos se encuentran encriptados y se deben desencriptar y así tomar decisiones, poder comunicar los dos Arduinos y mostrar la información correcta. Se describen un poco los problemas encontrados en el proceso del diseño como de los códigos y los conocimientos que se deben obtener para realizar esta comunicación con reglas previamente establecidas.

Conceptos:

- **Arduino:**
  Es una placa electrónica donde se pueden conectar periféricos de entrada y salida de un microcontrolador, es de código abierto, el dispositivo permite crear códigos de diferentes tipos de microcontroladores.
- **Puerto Serie:**
  El término serial se refiere a los datos enviados mediante un solo hilo. Los bits se envían uno detrás de otro como se muestra en el siguiente esquema figura **1** en el cual está pasando información desde el equipo A hacia el equipo B con una conexión en serie.
- **Puerto Paralelo:**
  Es un tipo de puerto que permite el intercambio de paquetes simultáneos (en ambos sentidos) de datos por medio de hilos o cables. La forma en que funciona este puerto es a nivel físico, este tiene un cable por cada bit que se envía, por eso se llama puerto paralelo.

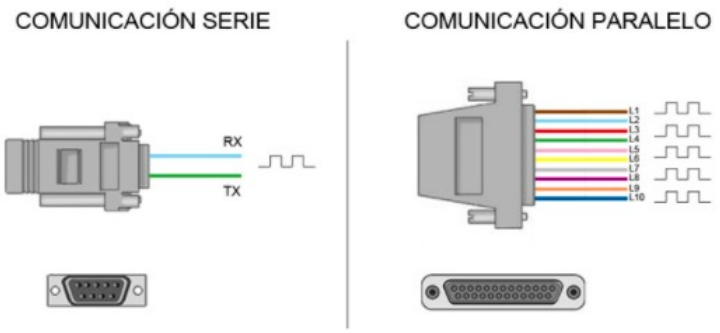

Figura 1: Comunicación serie y paralelo

- **74HC595:**

  Descripción: El 74HC595 es un chip de registro de desplazamiento de 8bit de entrada serial y salida serial y paralela que normalmente se utiliza como chip de expansión de los cuales se usan 3 pines de la MCU y este nos permite usar 8 pines del chip. Ver figura 2.

  Configuración de pines y funciones: 74HC595 ver cuadro 1.

  Características:

  - Maneja un voltaje mínimo de -0.6V a un máximo de +5 V.
  - Tiene 8 biestables para almacenar 8 bits. El pin de salida o patillas, Vcc y GND para la alimentación, y Q que son las de datos. El resto corresponden a señales de reloj/control.

Cuadro 1: Chip 74HC959

CHIP 74HC959		
PIN	**SÍMBOLO**	**DESCRIPCIÓN**
1	QB	Salida paralela 1
2	QC	Salida paralela 2
3	QD	Salida paralela 3
4	QE	Salida paralela 4
5	QF	Salida paralela 5
6	QG	Salida paralela 6
7	QH	Salida paralela 7
8	GND	Conexión a tierra
9	QH'	Conexión a otro chip 74HC595
10	SRCLR	Resetea el registro de desplazamiento (activa con nivel bajo)
11	SRCLK	Reloj de sincronización de los datos
12	RCLK	Reloj registro de almacenamiento
13	OE	Habilita las salidas (se activa con un nivel bajo)
14	SER	Ingreso de los datos serial
15	QA	Salida paralela 0
16	VCC	Alimentación del chip (5v)

- Las entradas la tiene en serie y la salida en paralelo, con una sola entrada, se pueden controlar a la vez esas 8 salidas. Solo necesitarás tres pines del microcontrolador usado para manejarlo. Esas son Latch, Clock y Data. Latch es el pin 13 en este caso. Clock puede estar en el 11 u otros, y el bit de datos es el 14.
- La señal de reloj alimenta el circuito, es el ritmo, los datos cambian el comportamiento del chip, pasa de LOW a HIGH , con esto se consigue es grabar la posición actual donde se encuentre el desplazamiento el valor ingresado por este pin de datos. Si repites esto 8 veces, entonces habrás grabado las 8 posiciones y tener un byte almacenado (QA-QH).
- Salida de datos de tres estados de salida paralela de 8 bits (tri-estado: nivel bajo, nivel alto y estado de alta impedancia).
- Con frecuencia de cambio de 100MHz.
- La salida en serie puede controlar el chip en cascada del siguiente nivel.

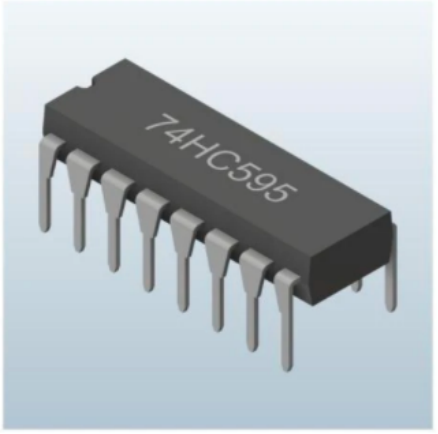

Figura 2: Chip 74HC595

- **Aplicaciones:**
  - Conmutadores de red
  - Infraestructura de energía
  - Pantallas LED
  - Servidores
- **Observación:**
  El circuito integrado 74HC595 es un registro de desplazamiento que cuenta con entrada en serie y salida en paralelo de 8 bits (SIPO). Se alimenta un registro de almacenamiento de tipo D (datos).

  El 74HC595 es de gran ayuda cuando se requiere ampliar la cantidad de salidas digitales ya que se puede conectar 8 LEDs con tan solo 3 pines del microcontrolador o Arduino.

Metodología:

Procedimos a leer, analizar y discutir el texto del desafío, y decidimos como grupo de trabajo que cada quien se instruya sobre los conocimientos necesarios para llevar a cabo el desarrollo del desafío, como lo es Arduino, c++, manejo de repositorios, chip 74HC595, puertas lógicas, con el fin de hacer uso de estos, aplicándolos al desarrollo. De forma grupal, nos guiamos de algunos ejemplos de Arduino que utilizan el chip 74HC595, con el fin de entender de una mejor manera el uso de este chip en la vida real.

En el primer Arduino (**figura 3**), entendimos a partir del código (**figuras 4** y 5) que el chip lo podemos usar para convertir información serial a paralela y entendimos la forma de desplazamiento de los datos, evidenciando esto con la iluminación de cada led.

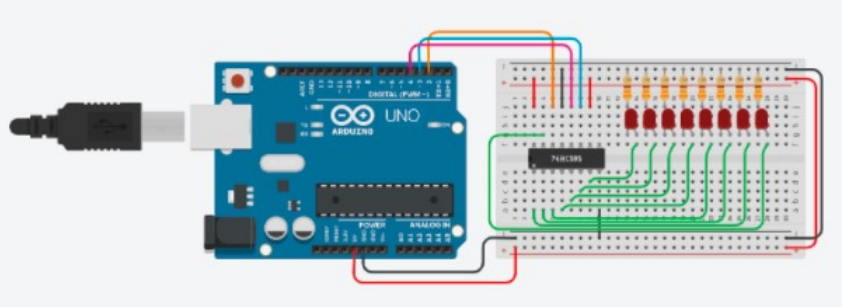

Figura 3: Ejemplo Arduino 1

```
#define pinDATA 2 //SER
#define pinCLOCK 3 //SRCLK
#define pinLATCH 4 //RCLK

void setup() {
  Serial.begin(9600);
  pinMode(pinDATA, OUTPUT);
  pinMode(pinCLOCK, OUTPUT);
  pinMode(pinLATCH, OUTPUT);
  digitalWrite(pinCLOCK,LOW);   //SE INICIA EN BAJO POR SI NO SE HA DECIDIDO EL
DATO QUE SE VA ENVIAR
  digitalWrite(pinLATCH,LOW);   //SE INICIA EN BAJO POR SI TIENE ALGO EN
MEMORIAL
}

void loop()
```

Figura 4: Primera parte del código ejemplo 1

```
{
 byte j = B00000001; // ES IGUAL A 1 EN BINARIO 00000001
  for(int i = 0; i < 8; i++)
  {
    escribir(j);
    j<<= 1;
    Serial.println(j);
    delay(500);
  }
}

//FUNCION ESCRIBIR EN CHIP LO QUE SE VA A MANDAR
void escribir(byte data)
{
  byte h = B10000000;  // h es ugual a 128 en binario 1000000
  for(int i = 0; i < 8; i++)
  {
    if(data == h){
      digitalWrite(pinDATA,HIGH);//DATA=1 // INDICAR QUE ES UN 1 LO QUE VA A
ENVIAR
    }
    else
    {
      digitalWrite(pinDATA,LOW);//DATA=0  // INDICAR QUE ES UN 0 LO QUE VA A
ENVIAR
    }

    h >>= 1;
    digitalWrite(pinCLOCK,HIGH);//CLOCK=1   se envía el valor que se eligió de pinDATA
    digitalWrite(pinCLOCK,LOW);//CLOCK=0
  }
  digitalWrite(pinLATCH,HIGH);//LATCH=1
  digitalWrite(pinLATCH,LOW);//LATCH=0
}
```

Figura 5: Segunda parte del código ejemplo 1

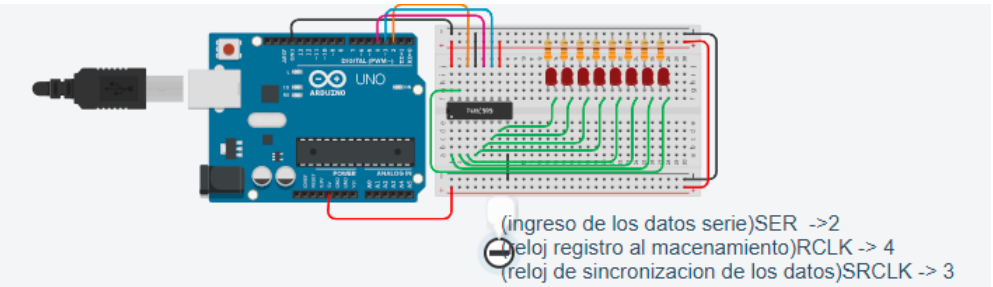

Figura 6: Ejemplo Arduino 1

En el segundo Arduino (**figura 6**), a partir del código (**figura 7**), que podemos controlar de manera manual la iluminación de cada led.

Después de estos ejemplos anteriores con el chip 74HC959 pasamos ahora a implementar un código que nos permita, convertir a binario cada uno de los datos entregados y así poderlos pasarlos al chip 74HC959 y mostrar por medio de leds el byte en paralelo. Avance del diseño en la figura 8.

Después de evidenciar que los leds encienden correctamente, pasamos ahora a comprobar ese byte con cada uno de los números primos entre 0 a 255 y así poder saber qué número es primo, para tomar decisiones de qué camino seguir cuando encuentre o no un número primo.

Después de esta comprobación tenemos dos caminos, uno, si es primo verificar si el siguiente número es 249 o dos, si no es primo, seguir verificando los próximos números de la trama, comprobando si son primos o no. Ver figura 9.

Realizando este procedimiento el Tinkercad se bloqueo completamente borrándolos y bloqueándonos todo el código y diseño que llevábamos, no se encontró solución un buscando ayuda y apoyadonos de los profesores, debimos empezar de nuevo.

Empezando de nuevo organizamos más el diseño y en una asesoría que tuvimos entendimos más como es el funcionamiento, también haciendo pruebas en Tinckerkad nos dimos cuenta que no podemos los dos interactuar en el código ya que este se bloquea y no guarda los cambios de uno o del otro, por esto decidimos que solo una persona escribía y la otra observaba y ayudaba con soluciones para los problemas que salían.

```
#define SER 2 //ingreso de los datos serial
#define RCLK 4 //reloj registro de almacenamiento
#define SRCLK 3 //reloj de sincronizacion de los datos

#define LED0 1
#define LED1 2
#define LED2 4
#define LED3 8
#define LED4 16
#define LED5 32
#define LED6 64
#define LED7 128

void setup() {
  pinMode(SER, OUTPUT);
  pinMode(RCLK, OUTPUT);
  pinMode(SRCLK, OUTPUT);
}

void loop()
{
  digitalWrite(RCLK, LOW);
  shiftOut(SER,SRCLK, MSBFIRST, LED0);
  digitalWrite(RCLK, HIGH);
  delay(500);
}
```

Figura 7: Primera parte del código ejemplo 2

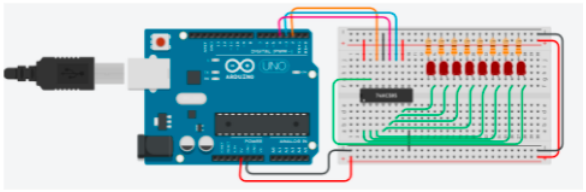

Figura 8: Primer diseño

Luego de tener ya el código identificando el primo y la bandera, pudimos avanzar dos posiciones y observar el valor real es este paso tuvimos varias problemas como, reutilizar variables en diferentes funciones que nos dañaba las funciones en donde esta variable se encontraba, también nos demoramos un poco en hacer avanzar las posiciones del string de los datos para no repetir valores y que los valores que éste leyera fuera el siguiente y no el mismo o uno más adelante o atrás.

Realizando la comunicación tuvimos que consultar como se realizaba y como era la conexión entre ellos, esto nos demoró un poco ya que no conocíamos como era el envío de datos por vía serial, pero cuando lo logramos y fue fácil identificar los 0 y los 1 y así poder mostrarlos en un LCD. Ver figura 10.

Análisis:

Se debe enviar de un Arduino a otro una trama de datos que se encuentra encriptada, se debe leer la trama, verificar qué datos son los reales y mostrarlos en otro Arduino. Para esta comunicación se utilizará un chip 74HC595.

Para identificar el 249 en el trama utilizamos dos chips, en el chip 1 entran los datos de la trama y en el chip 2 siempre muestra el 249, comprobamos si el dato de la trama es primo, guardamos en una variable booleana si lo es o no, luego comparamos los bits del chip 1 y el chip 2, esto por medio de dos XOR cuádruples, junto con dos OR y así al final tenemos un 1 o un 0 que nos indica si los bits son iguales o diferentes, si son iguales lo que hacemos es verificar si el dato anterior es primo, esto lo hacemos comparando una variable booleana que anteriormente asignamos, si es primo lo que hace el Arduino 1 es mandar un 1 a la terminal y estos datos de la terminal llegan al Arduino 2 por vía serial, si no es primo manda a la terminal un 0, este dato llega al Arduino 2 por vía serial, y continúa evaluando los números siguientes realizando este mismo proceso explicado.

Dos datos que llegan al Arduino 2 por vía serial, este verifica si es un 1 o un 0, si es un 0 lo que hace es continuar por el próximo valor, si en algún momento encuentra un 1 que es nuestro valor bandera, lo que hace es avanzar una posición, (esto lo hace por que nuestra regla de desencriptación se encuentra dos posiciones después de la bandera) y lee los siguientes 8 bits indiferentes si son 1 o 0 con el fin de obtener un byte, el byte es el número real que debemos mostrar en el LCD, para mostrarlo lo convertimos de nuevo a decimal y lo mostramos finalmente en el LCD.

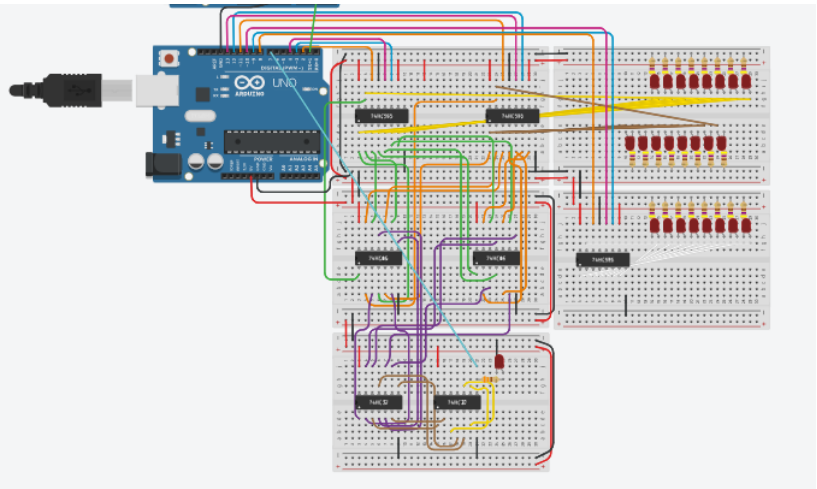

Figura 9: Segundo diseño

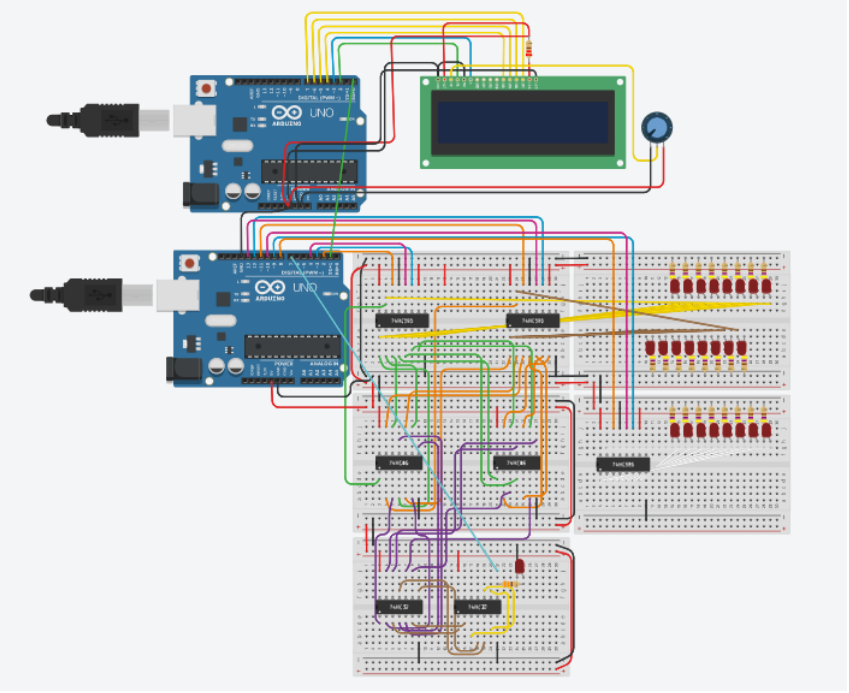

Figura 10: Tercer y ultimo diseño

Conclusiones:

Realizando este trabajo concluimos que la comunicación entre dos Arduinos es sencilla al momento de enviar ya los datos y recibirlos, en el proceso de encontrar los datos reales por medio de hardware se hace más largo que realizarlo por medio de software, pero realizando este trabajo por hardware utilizamos dispositivos como el 74HC595 que no conocíamos y aprendimos que por medio de hardware también se puede hacer validaciones que se hacen por software y este conjunto de dispositivos utilizados por hardware hacen que desarrollemos habilidades y destrezas en el uso de circuitos.

Referencias:

- https://www.ti.com/lit/ds/symlink/sn74hc595.pdf?ts=1644870203944&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FSN74HC595
- https://programarfacil.com/blog/74hc595-registro-de-desplazamiento-Arduino/
- https://www.hwlibre.com/74hc595/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+hwlibreweb+%28Hardware+libre%29
- https://www.youtube.com/watch?v=1ryCohAYV9Q
- https://www.youtube.com/watch?v=LFqIA3ZvZE8
- https://cdn.shopify.com/s/files/1/0069/0028/5529/products/74hc595-circuito-integrado-registro-de-diZ130095594XvZgrandeXpZ1XfZ61349119-429872737-1XsZ61349119xIM_grande.jpg?v=1558318879