Informe de Análisis

Sistema de comunicación encripatado

John Edisson Chamorro Coral

Daniela Alvarez Bernal

Despartamento de Ingeniería Electrónica y
Telecomunicaciones
Universidad de Antioquia
Medellín
Febrero 21 de 2022

Índice

1. Resúmen

El tratamiento de la información bancaria requiere de varias capas de seguridad aplicada desde todos los frentes. El canal de comunicación es uno de los puntos críticos respecto a la seguridad, pues es más vulnerables al estar expuesto sin seguridad constante. Es por ello que en este documento se procede a realizar una solución de seguridad, con la cual, por medio de microcontroladores Arduino Uno R3, se manipulará la información transmitida desde un punto y aplicando algoritmos de encriptación se brindara un nivel de protección y seguridad a dicha información hasta llegar al receptor.

2. Introducción

La información se ha vuelto uno de los objetos de mayor valor en la actualidad, pues con estos se puede realizar grandes tareas en pro del crecimiento investigativo, empresarial, entre otras áreas; Pero a su vez la vulnerabilidad de esta puede poner en conflictos a muchas personas e incluso a naciones. Es por ello que se ha incrementado la investigación de nuevas tecnologías que permitan proteger la información cuando sea requerido. Hay que tener en cuenta que la comunicación se puede transmitir por muchos medios [1], y por tanto tiene muchos frentes de ataque por lo que se hace necesario enfocarse en cada uno de ellos y brindar una solución a la problemática de la vulnerabilidad de la información. A pesar de que en la actualidad hay muchas herramientas que buscan la protección de datos, para el caso donde la comunicación se realizara a través de medio cableado, lo mejor para seguridad de la información es usar encriptación de datos. Es decir se modificará la información de tal forma que solo pueda ser entendida o cifrada por medio de un algoritmo especializado y único dentro del protocolo de comunicación que se estableció [2].

3. Objetivos Generales

Como objetivo general se requiere generar un canal de comunicación entre dos microcontroladores Arduino Uno R3, en donde uno de ellos se encargara de enviar información encriptada en serie y esta se paralizará para proceder a su desencriptación; El resultado de desencriptar será enviado en serie hacia el otro microcontrolador receptor quien desplegará en pantalla la información. La comunicación debe brindar un nivel de seguridad y fiabilidad de los datos, además de tener un comportamiento eficaz y ser un sistema escalable.

4. Objetivos Específicos

- Analizar y documentar el funcionamiento del integrado 54HC595 y cómo este puede ayudar en la transmisión de datos en paralelo.

- Analizar y documentar el funcionamiento de los puertos digitales del microcontrolador Arduino Uno R.
- Analizar y documentar como generar una señal de reloj por medio de Arduino Uno R3.
- Analizar y documentar cómo transmitir información serial por medio de un puerto digital de Arduino Uno R3.

5. Análisis del problema

Se requiere un sistema de comunicación síncrono, en donde se envíe una trama de datos de forma serial, y en el recorrido pase por un sistema el cual convertirá los datos en una trama paralela. Esa trama paralela será encriptada y enviada al sistema de desencriptación, el cual procederá a realizar la selección de los datos correctos y se procesará para serializarlos y enviarlos al receptor.

Bajo lo anterior el sistema se podría descomponer en los siguientes subsistemas:

- Un subsistema de transmisión de datos de manera serial en tramas de 8 bits.
- Un subsistema de Reloj para protocolizar la transmisión y recepción de datos.
- Un subsistema conversor de datos seriales a datos paralelos en tramas de 8 bits.
- Un subsistema de desencriptación de datos con entrada en paralelo y salida en serie.
- Un subsistema de recepción de datos de manera serial síncrono con el sistema de Reloj.
- La clave del sistema de desencriptación será cuando se ingrese el número 73.
- Los datos validos o a guardar serán el promedio de los últimos cinco números antes de que ingrese la clave.

Las consideraciones a tener en cuenta para la solución de este problema, son:

- El ambiente de trabajo será virtual.
- Dentro de las herramientas de trabajo se tiene dos Arduino Uno R3 (transmisor y receptor), un dispositivo Integrado 54HC595, cables y placas.

- La trama de datos a enviar estará constituida por los datos a transmitir, y un dato bandera el cual indicará al sistema de desencriptación, que se debe realizar el proceso de toma de datos para su proceso de serialización.
- La información tratada será realizada bajo bloques de 8 bits.
- La serialización de datos y el sistema de reloj se realizará por medio de dos puertos digitales de Arduino Uno R3 transmisor.
- Para el procesamiento de los datos en paralelo se emplearan 8 puertos digitales del Arduino receptor.
- Lo desencriptado se mostrara por consola como medio de representación visual.

5.1. ¿Cómo funciona el Integrado 74HC595?

Circuito integrado:

Un circuito integrado en una estructura pequeña de material semiconductor [9], normalmente de silicio, sobre la que se fabrican circuitos electrónicos. En cuanto a las funciones integradas los circuitos se clasifican en dos grandes grupos:

- Circuitos integrados analógico: No tratan con señales digitales sino analógicas, es decir que función abordando señales continuas variables en lugar de señales discretas, la electrónica digital es discreta por el hecho de que cada señal solo puede tener dos valores. En cambio, en la electrónica analógica cada señal es de rango variable.

- Circuitos integrados digitales o circuitos lógicos: Este tipo de circuito integrado tienen dos niveles definidos: 1 y 0, lo que implica que funcionan en matemáticas binarias en las que 1 significa encendido y 0 apagado.
  Estos circuitos emplean componentes encapsulados, los cuales pueden albergar puertas lógicas o circuitos lógicos más complejos. Los Circuitos Lógicos están compuestos por compuerta lógicas como la compuerta AND (Y), compuerta OR(O),compuerta NOT(NO) y otras combinaciones más complejas de los circuitos antes mencionados. Las compuertas lógicas tienen una única salida, aunque pueden tener una o más entradas. Los componentes que emplean estos circuitos están estandarizados, para que haya una compatibilidad entre fabricantes, de forma que las características más importantes sean comunes. De forma global los componentes lógicos se engloban dentro de una de las dos familias siguientes: TTL: diseñada para una altavelocidad. CMOS: diseñada para un bajo consumo, Esta familia requiere un manejo especial ya que la electricidad estática del cuerpo humano podría dañarlos al tocar sus terminales. Actualmente dentro de estas dos familias se han creado otras, que intentan conseguir lo mejor de ambas.

Cada circuito integrado tiene cierto número de pines o terminales. Es muy importante saber dónde va conectado cada terminal, ya que si se conecta en forma errada sepuede dañar fácilmente. Los pines están numerados en el sentido contrario a las manecillas del reloj en forma de U. Los circuitos integrados vienen en configuraciones de 8, 14, 16, 18, 20, 24, 40 y 64 pines.

Circuito integrado 74HC595:

Es un registro de desplazamiento, es decir un circuito digital secuencial que cuenta con entrada en serie (bit a bit) y salida en paralelo de 8 bits. Esto quiere decir que las salidas en un instante dado no dependen única y exclusivamente de las entradas en dicho instante, sino que dependen también de las entradas que han ocurrido con anterioridad. En este tipo de registros con salida en paralelo se dispone de la salida de cada flip-flop por lo que una vez almacenados los datos cada bit se representa en su respectiva salida. De esta manera todos los bits de salida estarán disponibles al mismo tiempo.

Este circuito consistente en una serie de biestables o flip-flop (circuito que tiene dos estados estables y puede almacenar información.) del tipo D (biestable de datos o seguidor. Tienen una única entrada D), conectados en cascada y un latch (circuito biestable que puede almacenar un bit de información) tipo D (solo tiene una entrada D) de 8 bits. El registro de desplazamiento proporciona datos paralelos al latch y también cuenta con una salida en serie.

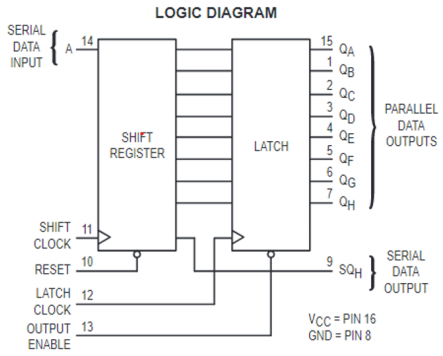

Figura 1: Diagrama Logico Circuito Integrado 74HC595

La principal diferencia entre el latch y el flip-flop es que un latch cambia de estado de inmediato, según sus señales de excitación de entrada, es decir que verifica la entrada continuamente y cambia la salida cuando hay un cambio en la entrada. [3] Mientras que un flip-flop espera la seña de su reloj antes de cambiar de estado. [4]

Descripción de los pines:

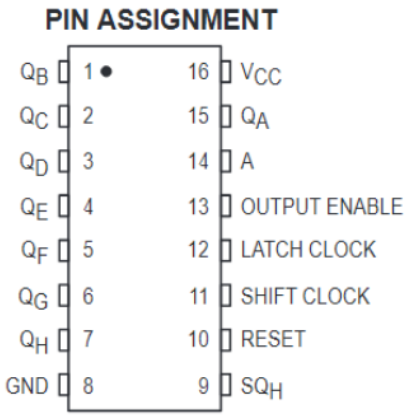

Figura 2: Diagrama de asignacion de pines de Integrado 74HC595

Inputs:

- A(Pin 14): Entrada de datos en serie. Los datos en este pin se desplazan al registro de desplazamiento en serie de 8 bits.

Control inputs:

- Shift Clock(Pin 11): Entrada de reloj de registro de desplazamiento. Una transición de bajo a alto en esta entrada hace que los datos en el pin de entrada en serie se desplacen al registro de desplazamiento de 8 bits.
- Reset (Pin 10): Alto-bajo asincrónico, entrada de restablecimiento de registro de desplazamiento. Un nivel bajo en este pin restablece solo la parte del registro de desplazamiento de este dispositivo. El latch de 8 bits no se ve afectado.
- Latch Clock (Pin 12): Entrada de reloj del almacenamiento latch. Una transición de bajo a alto en esta entrada bloquea los datos del registro de desplazamiento. Funciona para cargar los datos del pin 14 de datos, al momento que se activa con el estado lógico "1"
- Output Enable (Pin 13): Habilitación de salida activa-baja. Un nivel bajo en esta entrada permite que los datos de los latches se presenten en las salidas. Un valor alto en esta entrada obliga a las salidas (QA-QH) a entrar en el estado de alta impedancia (Resistencia de un circuito al flujo de una corriente eléctrica alterna). La salida serial no se ve afectada por esta unidad de control.

Outputs:

- QA – QH (Pins 15, 1, 2, 3, 4, 5, 6, 7): Salidas no invertidas (no son contrarias a su entrada) de 3 estados (un estado de bajo nivel (0), un estado de alto nivel (1), un estado de altaimpedanciao estado flotante (Z)) de los latches.
- SQH (Pin 9): Salida de datos en serie no invertida. Esta es la salida de la octava etapa del registro de desplazamiento de 8 bits. Esta salida no tiene capacidad de tres estados. [5]

Especificaciones y características:

- Rango de voltaje de alimentación: – 0.5 Vcc a + 6 Vcc.
- Voltaje de entrada: 0 Vcc a Vcc.
- Voltaje de salida: 0 Vcc a Vcc.
- Consumo de corriente: 80 A max.
- Frecuencia de reloj máxima opera dependiendo el VCC:
    - 2 Vcc la frecuencia es 4.8 ns.

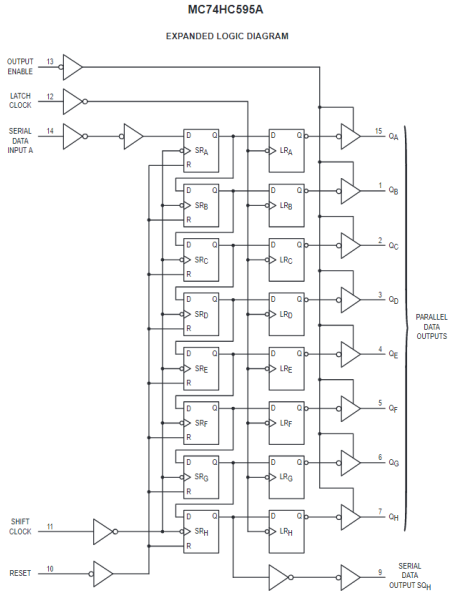

Figura 3: Diagrama digital del Circuito Integrado 74HC595

- 4.5 Vcc la frecuencia es 24 ns.
- 6 Vcc la frecuencia es 28 ns.

- Familia lógica: CMOS.
- Número de bits: 8 bits.
- Número de entradas: 3.

5.1.1. Aplicación de Integrado 74HC595 empleando switches

En este ejemplo mostraremos el funcionamiento del circuito por medio de 8 leds conectados a las ocho salidas del circuito. El objetivo es encender el número de leds que queramos de forma paralela.

Para esto utilizamos:

- 2 placas de prueba.
- 8 Leds.
- 1 fuente suministro de energía.
- 2 Switches.

- 11 Resistencias, de las cuales 8 se usaron para proteccion de los led y tuvieron un valor de 220 ohm, dos resistencias de 1000 ohm para realizar un divisor de tension y una resistencia de 500ohm para proteccion de suministrar de tension el integrado.
- 1 Integrado 74HC595.

El pin número 13 nos permite habilitar o deshabilitar todas las salidas, como queremos que todas las salidas estén habilitadas este pin siempre estará en cero voltios. El pin 10 estará recibiendo 5 voltios constantemente ya que este pin se activa con nivel bajo y resetea el registro de desplazamiento. El pin 8 es la tierra y el pin 16 es Vcc. El pin 14 es el pin de entrada de datos en serie y el estado que ingresemos por este pin se va a ver reflejado en los pines de salida, como nuestro objetivo es encender los leds este pin siempre va a estar en alto.

Cuando ingresamos pulsos positivos por el pin 11 que es el shift clock por medio de un switche le decimos al circuito que lea la entrada y cuando ingresamos un pulso positivo por medio de un switche a el pin 12, que es le latch clock, le indicamos al registro de almacenamiento que tome los datos del de desplazamiento y los exponga al exterior iluminando así tantos leds como pulsos positivos hayamos enviado al shift clock antes de activar el latch clock.

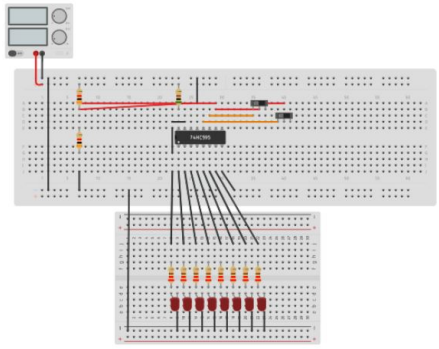

Figura 4: Esquema de uso de integrado 74HC595 con switches

5.1.2. Aplicación de Integrado 74HC595 empleando Arduino Uno R3

En este ejemplo vamos a usar el circuito con un Arduino. El objetivo fue encender ocho leds en orden uno por uno.

Para ello fue necesario implementar los siguientes elementos:

- 2 placas de pruebas.
- 8 leds.
- 1 placa Arduino uno R3.
- 17 resistencias.
- 1 integrado 74HC595.

El funcionamiento para dar un uso simple al circuito integrado, es alimentar con tensión alta, y de manera constante, el reset para que el integrado no me borre el registro de datos almacenados; la entrada serial tendrá una tensión alta y constante ya que, para simular entrada de datos, se usó la entrada de reloj de desplazamiento, que es el pin 11 del integrado, es decir por medio de una función de código, se procederá a alimentar el puerto digital del Arduino que está conectado al pin 11 para que este esté en constante cambio de estado HIGH a LOW y viceversa. Esto con el fin de permitir ingresar el dato que halla en el puerto serial y a su vez desplegarlo hacia el registro de desplazamiento. Ahora bien, para poder desplegar hacia los leds, fue necesario usar otro puerto digital de Arduino conectado al pin 12, que es la entrada del reloj de registro de salida, y de la misma manera que con el reloj de desplazamiento, se alimentó el reloj de Latch.

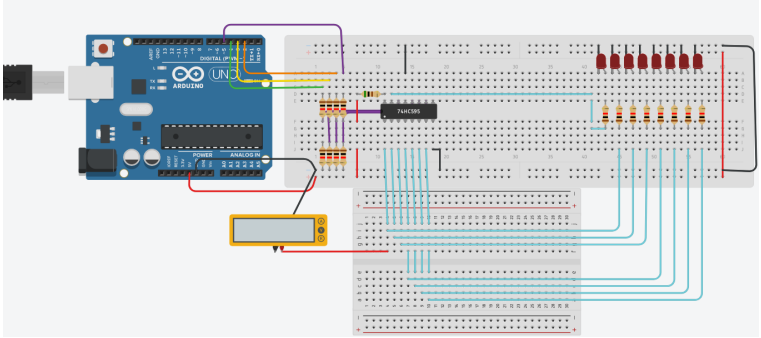

Figura 5: Esquema de uso de integrado 74HC595 con Arduino Uno R3.

5.2. Cómo funciona el bloque que paraleliza los datos

La paralelización de los datos se puede realizar por medio del integrado 74HC595, ya que como se observó en la documentación de este integrado, es posible convertir datos en serie a datos en paralelo con un buen control de los relojes internos.

Específicamente, al usar el integrado 74HC595, se tendría que enviar los datos, de forma serial, al pin 14 y al indicarle por medio del clock de desplazamiento (pin 11), estos datos se van almacenando en el shift register; como bien se sabe el circuito integrado solo tiene la capacidad de paralelizar hasta 8 bits al tiempo. Ahora bien para desplegar los bits almacenados es necesario mandar tensión hacia el pin 12, que es el latch clock, para indicarle al registro de almacenamiento que tome los datos almacenados y los exponga al exterior de manera paralela.

Básicamente este sería la manera se emplearía el uso del circuito integrado para paralelizar los datos que se envíen de manera serial desde Arduino. . [6]

5.3. ¿Qué es y cómo funciona Arduino?

Arduino es una placa electrónica de hardware libre, que cuenta con múltiples periféricos, los cuales interactúan como transmisores o receptores de información permitiendo manipular variables físicas o virtuales, a través de un microcontrolador que para este caso es ATmega328P [7]. Las variables se manipulan por medio del microcontrolador que a su vez es configurado por medio de instrucciones en lenguaje de programación de alto nivel [8].

Debido a su enorme flexibilidad y carácter libre y abierto, este dispositivo permite realizar casi cualquier cosa como por ejemplo relojes, básculas, robots, entre otras cosas.

Figura 6: Placa Arduino Uno R3

5.4. ¿Cómo funcionan los puertos digitales de Arduino?

Los pines en Arduino son adaptados para trabajar como entrada o salida de datos, pero para que Arduino reconozca el puerto digital como entrada o salida debe ser configurado a través de código. Los pines configurados como entrada están en estado de alta impedancia [9], lo que quiere decir que se requiere muy poca corriente para poder realizar una lectura de la variable asociada a dicho puerto; el problema asociado a esa alta impedancia es que si ese puerto no tiene conectado nada, registra el ruido electrónico del entorno o se acopla capacitivamente al estado de un pin cercano [9]. Alguno de los puertos digitales de la placa permite realizar transmisión de PWM con una frecuencia de 490Hz y con una tensión de aproximadamente 2.5V [10]. Ahora bien el puerto digital configurado como salida tiene una baja impedancia lo que permite suministrar hasta 40 mA hacia otro circuito externo Tabla 1. Básicamente su funcionamiento es a través de corriente o tensión, interactuando con las variables o circuitos como si fueran switches. Es decir el pin se activa con un valor de tensión de 3V pero no se activa con un valor de tensión menor al valor antes mencionado; respecto a un puerto como salida, actúa con un nivel alto con una tensión de 5 V (0 3.3V para placas alimentadas a esta tensión), y con un valor bajo con 0V.

5.5. ¿Cómo generar una señal de reloj en Arduino?

Recordando que Arduino tiene en su estructura 6 puertos digitales Tabla 1, que me permiten realizar transmisión en PWM, es fácil deducir que se podría realizar una señal de reloj por medio de estos puertos. Bastaría con configurar el ancho de pulso requerido para el reloj; Teniendo en cuenta que el puerto permite enviar, en este formato PWM, valores entre 0 y 255 haciendo alusión al porcentaje que el valor alto de la señal permanecerá dentro del periodo de transmisión, es decir que si requiero un ancho de pulso del 50Cabe resaltar que para cada placa de Arduino, la frecuencia de transmisión de datos por medio de estos puertos digitales varía; Para el caso de Arduino Uno R3 la frecuencia de 490Hz [10], es decir que para una señal de reloj por medio de esos puertos, se tendría una frecuencia estándar. Si dentro de la solución de salida de reloj se desea obtener una frecuencia diferente a la estándar, se deberá generar una función codificada para que el microcontrolador pueda interactuar con la salida del puerto digital enviando valores altos y bajos con el retraso asociado a la frecuencia deseada.

5.6. ¿Cómo transmitir datos de manera serial por Arduino?

Aunque algunos puertos digitales de Arduino generan salida PWM, no son funcionales para enviar información, por tanto se debe desarrollar una implementación de instrucciones dentro del microcontrolador para que este, por medio de un puerto digital, permita enviar información en trama de 8 bits.

Recordando que los puertos digitales se comportan como switches, es posible enviar tramas binarias con solo controlar la salida digital correspondiente al dato a enviar y enviando el valor asociado a este.

Para lograr lo anterior se deben tener en cuenta las siguientes consideraciones:

- Se define el puerto digital para la transmisión de datos serial.
- La entrada de datos numéricos se realizará por consola. Cabe resaltar que la lectura de datos por consola se realiza tomando de un dato a la vez.
- Los datos numéricos ingresados por consola serán almacenados en una variable String para su posterior manipulación.
- Para el tratamiento de los datos ingresados por consola y su futura conversión a binario y despliegue se requiere de tres funciones que son, conversión de entero a representación en String de binario de 4 bits denominada intToBinary, integración de binarios para formar un binario de 8 bits denominada binary, y el despliegue o envío de binarios por un puerto digital denominada sendSerial.
- Inicialmente el dato numérico que se toma por consola, será almacenado en una variable String, y este será enviado a la función binary; Dentro de la función binary se toma la representación en decimal del dato numérico que está almacenado en la variable String y se envía, por separado, hacia la función intToBinary para que este realice la comparación correspondiente y me reenvié la representación binaria, en String, del número ingresado. Finalmente esas representaciones binarias de los números tratados son almacenadas en una sola variable creando así un binario de 8 bits el cual representa el dato numérico que se envió por consola.
- Como último paso se tiene reenviar al consolidado de 8bits hacia la función sendSerial, el cual se encarga de coger cada uno de los bits, convertirlos de tipo String a tipo entero y ese valor enviarlo por el puerto digital.

5.6.1. Inclusion de código de comunicacion serial simple entre dos Arduinos Uno R3

Listing 1: Código Arduino Uno R3 como transmisor serial

```
int serialSalida=3;

String dato;

void setup()
{
        Serial.begin(9600);
```

```
    pinMode(serialSalida , OUTPUT);

}

void loop() {

  if(Serial.available()>0){

    dato=Serial.read();
    sendSerial(binary(dato));

  }
}
//-- funcion que envia trama serial de 8 bits
void sendSerial(String serialData){

  for(int i=0;i<serialData.length();i++){

    int out=String(serialData.charAt(i)).toInt();
    delay(1000);
    digitalWrite(serialSalida ,out);

  }
}

//--funcion que retorna el binario 8 bits del valor ascci del numero
String binary(String dato)
{
  String res;
  int n=0;

  for(int i=0;i<dato.length();i++){

    n=String(dato.charAt(i)).toInt();
    res+=intToBinary(n);

  }

   return res;
}

//--funcion que convierte un int a string correspondiente a su binario

String intToBinary(int n){
  String binary;
  switch(n){
```

```
    case 0:
        binary="0000";
        break;
    case 1:
        binary="0001";
        break;
    case 2:
        binary="0010";
        break;
    case 3:
        binary="0011";
        break;
    case 4:
        binary="0100";
        break;
    case 5:
        binary="0101";
        break;
    case 6:
        binary="0110";
        break;
    case 7:
        binary="0111";
        break;
    case 8:
        binary="1000";
        break;
    case 9:
        binary="1001";
        break;
          }
  return binary;
}
```

[language=C++, caption=C digo Arduino Uno R3 como receptor serial]

```
int serialEntrada=4;


void setup()
{
        Serial.begin(9600);
        pinMode(serialEntrada, INPUT);

}
```

```
void loop() {

        int entrada=digitalRead(serialEntrada);
        Serial.print(entrada);
        delay(1000);

}
```

6. Marco Experimental

Bajo la anterior investigación y documentación, se pretende crear un sistema de comunicación en donde se envía información serial, y para el proceso de manipulación y desencriptación de datos, los datos se transformaran de manera paralela, con el fin de generar un grado de seguridad. Finalmente los datos desencriptados pasaran de un modelo paralelo a un modelo serial para finalmente llegar al receptor como información legible en pantalla. Para lograr diseñar el canal de comunicación deseado, se tuvieron en cuenta las siguientes consideraciones:

- La comunicación se realizará entre dos placas de Arduino y dentro del proceso de manipulación de datos, se empleará el integrado 74HC595 el cual se encargará de paralelizar los datos en tramas de 8 bits.
- La alimentación del circuito Integrado 74HC595 se realizará desde un puerto correspondiente en Arduino el cual se encarga de suministrar energía, en este mismo sentido las tierras del integrado y de las dos placas de Arduino tendrán el mismo punto de referencia.
- Para tener una sincronización de datos entre las dos placas, se procederá a implementar un puerto digital en Arduino transmisor para enviar pulsos de reloj y en la otra placa para recibir dicho pulso de reloj.
- Los datos a enviar y recibir se realizara por consola. El Arduino transmisión solo será responsable de enviar la trama binaria de manera serial. El Arduino receptor se encargara de manipular los datos en paralelo y a su vez convertir dicha información a serial para posteriormente transformar ese binario en un dato entendible y proceder a realizar su desencriptación.
- La desencriptación se realizará cuando se reciba el dato numérico correspondiente al valor 73, en donde se le define al sistema que debe tomar los anteriores cinco (5) valores y realizar su promedio. Finalmente su resultado es el dato valido a guardar.

6.1. ¿Cómo se realizó la transmisión de datos numéricos?

Los datos entran por el monitor en serie al Arduino 1, los números pueden ir separados por cualquier carácter ascii representado decimalmente por un

número menor al 48. Para la serialización de los datos, manipulamos la representación decimal ASCII del carácter que se lee desde consola, cada digito tendrá su representación binaria de 4 bits y al final se concatenará para formar una representación total de 8 bits. Por ejemplo, si ingresamos 1 por el monitor serial queda almacenado como 49, luego lo descomponemos en dígitos y cada dígito lo pasamos a binario para posteriormente concatenarlos; así, la representación en binario del 49 seria 01001001.

Antes de enviar los datos al integrado sincronizamos ambos Arduinos enviando pulsos de reloj a través del puerto 4 que estará conectado a la otra placa de Arduino. Después enviamos el número binario vía serial al integrado 74HC595 por medio de pulsos de reloj, se almacena en el shift register y posteriormente se despliega de forma paralela hacia sus 8 salidas que estarán conectadas al Arduino 2 a través de 8 pines digitales.

6.2. ¿Cómo se realizó la recepción y manipulación de datos numéricos?

Para la recepción de datos se emplearon, en total, 9 pines digitales en formato de entrada; ocho pines digitales (del puerto 2 hasta el 9), recibirán los bits que el integrado estará desplegando de manera paralela y el otro puerto (puerto 10), estará conectado con la otra placa de Arduino para realizar una sincronización en la transmisión/recepción de datos.

Como dentro del proceso de desencriptación se trabajará con los últimos cinco datos enviados antes de entrar el dato clave, se declara un vector de tipo entero y dinámico (numerosGuardados), para que vaya guardando datos a medida que vayan llegando. En cuanto el control de posiciones a la hora de almacenar los datos numéricos se procede a trabajar con una variable entero (index) que va a ir incrementando a medida que se procesen los datos recepcionados para su posterior almacenamiento.

Para ir almacenando los bits en paralelo se verifica por medio del pin 10 si se ha enviado una trama de datos seriales, de ser así se procede a esperar que se despliegue en todos los puertos de salida del integrado 74HC595 los bits correspondientes y posteriormente se guarda dicha información en una variable de tipo String (binario) y a su vez esta información se va concatenando en otro variable del mismo tipo (binarioTotal). Esta última será la encargada de almacenar toda la información binaria para el dato numérico correspondiente a uno, dos o tres dígitos.

Como se desconoce, en medio del proceso de recepción, si el dato enviado corresponde a un numero de uno, dos o tres dígitos, se implementó dos contadores, uno encargado de llevar un registro de cuantas veces se recepciona un dato (contador), y el otro el que va almacenar el número de dígitos del dato ingresado (contadorByte), ahora bien para tener un control exacto de dígitos se

creó un condicional el cual se encargó de verificar si el dato recibido corresponde a un dato numérico o carácter. Cuando el dato recibido es un carácter, que puede ser una coma o espacio dentro de lo que separa la lista de datos numéricos, se procede a almacenar en la variable ContadorByte el valor que lleva en el momento la variable contador llegando al valor de dígitos correctos del dato ingresado, a su vez se procede a consolidar todo los datos binarios en la variable binarioTotal para proceder a su siguiente tratamiento y por último se resetea la variable contador y binario para proceder con el nuevo ciclo de recepción.

Cuando un dato numérico ya fue completamente almacenado en la variable binarioTotal, se procede, por medio de otros condicionales y la variable contadorByte, a verificar de cuantos dígitos es ya que para cada caso se convierte a dato numérico de manera diferente, pero en todos los tres casos se realiza el almacenamiento del dato dentro del vector numerosGuardados, se incrementa la variable index y finalmente se resetea las variables binarioTotal y contadorByte para el nuevo ciclo de recepción.

Cuando el dato recibido es de un digito, se implementa la función byteToInt la cual recibe por parámetro una variable de tipo String, que para este caso es la variable binarioTotal, donde se encuentra toda la información binaria del dato recibo. Dentro de esta función se procede a dividir los ocho bits en dos grupos de 4 bits, para que por medio de la función denominada binaryToInt, convierta y retorne a un número entero lo correspondiente a la representación ingresada. El fin de este proceso es darle nuevamente forma al dato numérico hasta llegar a su representación decimal ASCII, por tanto es necesario que el primer número convertido y el que representa el primer digito de dicho decimal sea multiplicado por 10 para poder sumar el otro número convertido y finalmente llegar al valor deseado. Por último se procede a restar al decimal ASCII que se obtuvo, el valor 48 (representación decimal ASCII del número cero), para lograr finalmente el dato numérico que se recepcionó.

Cuando se recepciona un dato numérico de dos dígitos, se verifica si el dato recibido obtiene el valor definido como clave o no. Cuando lo recibido no corresponde a la clave se procede convertir esa representación binaria a el dato numérico por medio de la función twoDigits; twoDigits se encarga de dividir en dos grupos de a un byte para posteriormente emplear la función byteToInt la cual se encarga finalmente de entregar el valor numérico asociado a esa representación binaria. Finalmente para consolidar los números recibidos por medio de byteToInt en un solo dato numérico de dos dígitos, se multiplico, el representativo del primer digito, por 10 y se sumó el otro digito.

Cuando se recibe la clave se emplea la función datoValido, la cual se encarga de llamar la función promedio que me retornar el promedio de los últimos cinco datos numéricos recepcionados antes de que ingresara la clave, y a su vez este dato es convertido a String para ser retornado y almacenado.

Ya como último recurso, cuando se recepciona un dato numérico de tres dígitos, se emplea la función threeDigits que se encarga de dividir en tres grupos de 8 bits la trama binaria que representan dichos dígitos y se emplea nuevamente la función byteToInt.

7. Conclusiones

Con la placa de Arduino es posible realizar un sistema de comunicacion serial con un alto grado de calidad en cuanto a la transmision y recepcion de informacion.

A pesar de que Arduino ofrese multiples herramientas por medio de sus puertos, tambien tiene sus limitaciones, pero en contraparte está el saber diseñar instrucciones que permitan crear mecanismos para buscar la solucion a un problema en especifico.

Una de las grandes limitantes de Arduino es su memoria, por tanto es ideal tener conciencia sobre el manejo de recursos a la hora de diseñar las instrucciones de código.

Es importante tener en cuenta las frecuencias de trabajo de los puertos tanto del integrado como de Arduino, ya que con esto se puede ejecutar las tareas de transmision y recepcion de datos de manera que no se pierda informacion en el transcurso de dichas acciones.

Dentro de las instrucciones de codigo en Arduino tambien es importante tener en cuenta si se va a trabajar con retrasos pues cada instrucción consume un tiempo de ejecución y esto hace que posiblemente haya desincronización por lo que la información se tergiversa. En esta sección .

Referencias

[1] wikipedia. Medios de transmisión. [Online]. Available: https://sites.google.com/site/investigacionesitlm/home/1-3-medios-de-transmision

[2] xakata. Encriptar: qué es, para qué sirve y cómo cifrar tus archivos. [Online]. Available: https://www.xataka.com/basics/encriptar-que-sirve-como-cifrar-tus-archivos

[3] EcuRed. ¿cuál es la diferencia entre latch y flip flop? [Online]. Available: https://es.strephonsays.com/what-is-the-difference-between-latch-and-flip-flop

[4] V. P. Nelson. ¿cuál es la diferencia entre latch y flip flop? [Online]. Available: http://mumoaldigitales1.blogspot.com/2010/11/diferencia-entre-los-tipos-de-flip-flop.html

[5] ONSemiconductor. 8-bit serial-input/serial or parallel-output shift register with latched3-state outputs. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/12198/ONSEMI/74HC595.html

[6] B. Ar. Arduino desde cero en español - capítulo 70 - 74hc595 registro de desplazamiento (shift register). [Online]. Available: https://www.youtube.com/watch?v=LFqIA3ZvZE8t=640s

[7] Arduino. ¿qué es arduino? [Online]. Available: https://arduino.cl/que-es-arduino

[8] xakata. Qué es arduino, cómo funciona y qué puedes hacer con uno. [Online]. Available: https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer

[9] Arduino. Digital pins. [Online]. Available: https://docs.arduino.cc/learn/microcontrollers/digital-pins

[10] Hetpro. Arduino pwm con arduino uno. [Online]. Available: https://hetpro-store.com/TUTORIALES/arduino-pwm/