Informe de Parcial I

Sistema de generación y recepción de datos encriptados

Diego Alejandro Osorio Jiménez
Sergio Manrique Martínez
María del Mar Arbeláez

UNIVERSIDAD DE ANTIOQUIA

1 8 0 3

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

Índice

1. Introducción

En el siguiente proyecto se busca implementar un sistema de encriptación utilizado en una sucursal bancaria mediante conexiones desde un computador de origen A con información cifrada, hacia otro computador B en la cual debe de llegar desencriptada, para dar paso a un modelo de toma de decisiones.

2. Marco Teórico

En esta sección se pondrán los resultados de las investigaciones y análisis al respecto del proyecto y de sus componentes.

2.1. Análisis y diseño del proyecto

Para la solución de este problema, se va a dividir el proyecto en fases para que su implementación no sea de manera abrupta, que es algo también que se muestra en los ejemplos proporcionados por el profesor Augusto Salazar.

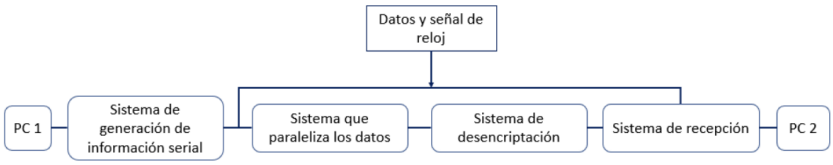

Figura 1: Ejemplo proporcionado

Así se planea dividir en las siguientes partes:

2.1.1. Computador Emisor

En este se generan o se reciben los datos de ocho bits que se terminan enviando a un arduino.

2.1.2. Arduino Receptor

El primer computador envía una señal con datos generados aleatoriamente a un Arduino que se encarga de leerlos y organizarlos, posteriormente lo manda a un sistema receptor que el cual los transforma de manera paralela.

2.1.3. Desencriptación

La información paralelizada se manda de a byte a un sistema de desencriptación, información que se envía al Arduino receptor.

2.1.4. Computador Receptor

Lee la información desencriptada y la muestra de forma serial.

2.2. Planeación del desarrollo del proyecto

En la siguiente, se muestra cómo se planea dar solución al proyecto día a día:

Día	Planeación
1	Organizar y planear el informe de análisis.
2	Comprender y desarrollar la especificación Nro1.
3	Empezar el montaje del circuito y ejecución de pruebas.
4	Diseño del algoritmo e implementación de las primeras funciones.
5	Se continúa el desarrollo del algoritmo.
6	Finaliza el desarrollo y se organiza el proyecto.
7	Desarrollo del video e informe de implementación.

Cuadro 1: Planeación del desarrollo del proyecto

2.3. Documentación y retroalimentación del circuito integrado 74HC595

El IC 74HC595 es un circuito integrado de registro de turnos que puede recibir datos en serie a través del pin de entrada SER y controlar 8 pines de salida en paralelo (Salidas Q0-Q7). Esto puede ser muy útil para ahorrar pines en nuestros microcontroladores ya que con solo 3 pines podríamos manejar 8 estados de corriente diferentes. Por ejemplo, si necesitáramos iluminar 8 leds no seria necesario utilizar 8 puertos del microcontrolador, con solo 3 pines podríamos dar instrucciones al 74HC595 y dependiendo de estas obtener 8 salidas paralelas para controlar el estado de nuestros 8 leds respectivamente. En nuestro proyecto usaremos el circuito integrado para paralelizar los datos emitidos por el arduino transmisor.

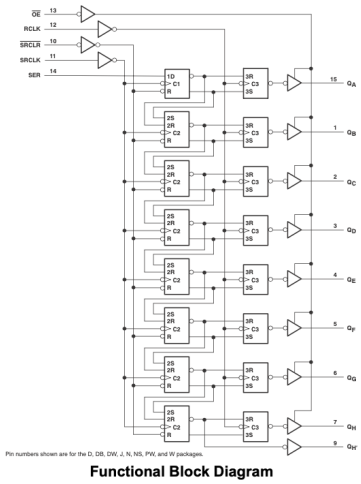

Figura 2: Functional Block Diagram

En el diagrama anterior podemos ver el funcionamiento interno del IC, el reloj de registro de desplazamiento (SRCLK) va corriendo cada bit a la siguiente salida Q(i+1) mientras no se detecte ningun flanco de subida en el reloj de registro de salida (RCLK). Una vez se detecte un flanco de subida en el RCLK si el pin de activación de salida esta en low se reflejara el estado correspondiente de cada salida Qi.

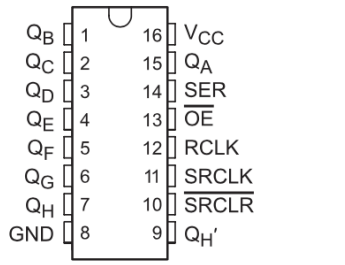

Figura 3: Pin Configuration Diagram

En el diagrama anterior vemos la configuración de pines del circuito integrado donde las conexiones que empiezan con Q son salidas, SER es la entrada digital, OE es el output enable (Que debería estar conectado a tierra, para que las salidas estén disponibles), RCLK es el reloj de registro de salida que permite guardar en el circuito integrado, SRCLK que es el reloj que se encarga de correr los datos para permitir la entrada de uno nuevo, SRCLR sería el que hace un clear o borra los datos para dar paso a otro set de datos (Suele estar conectado

al positivo) y, GND y Vcc serían tierra y power, respectivamente.

3. Implementaciones

Estos son los pasos dados por la guía para que la implementación del proyecto no sea abrupta, sino gradual, en términos de complejidad.

3.1. Implementación 1

Consiste en investigar e informarse al respecto del circuito integrado 74HC595 (Véase 2.3), para así poder utilizarlo en situaciones con y sin arduino.
Se hicieron 4 intentos, dos por cada situación. Se pueden ver todos en la siguiente figura (Figura 4):

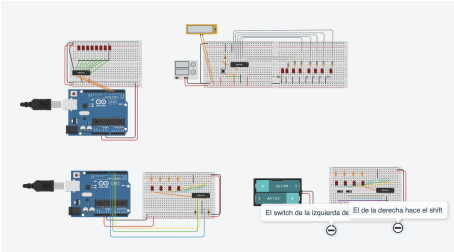

Figura 4: Circuitos de todos los intentos

3.1.1. Sin arduino

En primer lugar se trató de utilizar 8 leds y un solo pulsador para pasar la información a los leds. Como se puede ver en la siguiente figura (Figura 5):
Después se simplifica a 4 leds y 2 slide switchs. Donde uno de los switchs se

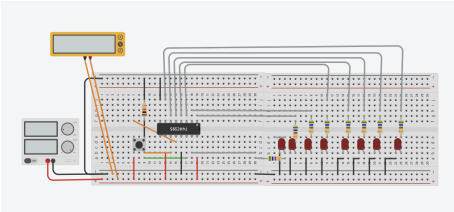

Figura 5: Sin Arduino Intento 1

encarga de dar la información del input (0 o 1, prendido o apagado), y el otro se

encarga de hacer el shift y permitir la entrada. De la siguiente manera (Figura 6):

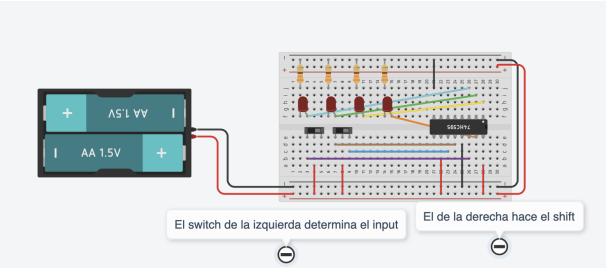

Figura 6: Intento correcto 2

3.1.2. Con arduino

Para el uso de arduino también se hicieron dos intentos:
En el primero se trata de hacer las conexiones y falta el código(Figura 7):
Para el segundo intento se hacen las conexiones correctas (Se minimiza a 4 leds)

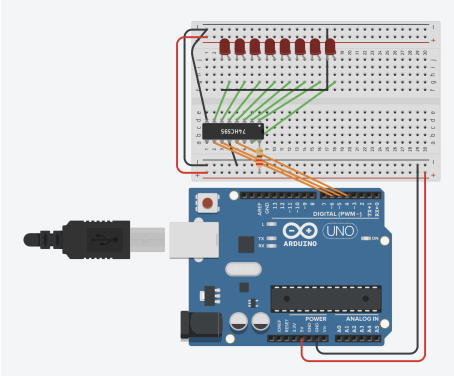

Figura 7: Intento 1 con Arduino

y se le escribe el código para pasar un número menor de 16 a una representación binaria en los leds. ((Figura 8))
En el código se ve que se definen los pines del input, el shift y el output clock, y que para hacer una impresión, se cierra la entrada del clock, se utiliza la función shiftOut para ir pasando los bits de un numero en binario (que se ingresa como entero), al circuito, al final se envía una señal a reloj de output para que se pase todo.

```
// C++ code
```

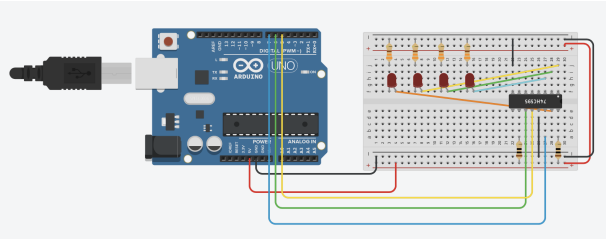

Figura 8: Intento correcto con Arduino

```
//
#define ORCLK 5
#define SRCLK 6
#define num 7

void setup()
{
  pinMode(num, OUTPUT);
  pinMode(ORCLK, OUTPUT);
  pinMode(SRCLK, OUTPUT);
}

void loop()
{
  int number=12; //Menor de 16
  digitalWrite(ORCLK,LOW);
  shiftOut(num, SRCLK, MSBFIRST, number);
  digitalWrite(ORCLK,HIGH);
}
```

3.2. Implementación 2

La segunda parte de la implementación se hace la conexión entre dos Arduinos, mediante el uso de los puertos seriales. Las conexiones se dan de forma sencilla siguiendo la figura (Figura 9):

Para esto, se crearían dos códigos, uno para el arduino emisor y otro para el receptor. Para el emisor, el pin que se utiliza sería definido como output y se esperaría un tiempo entre cada pulso. Sería así:

```
// C++ code
//
const int LED_BUILTIN1= 3;

void setup()
```

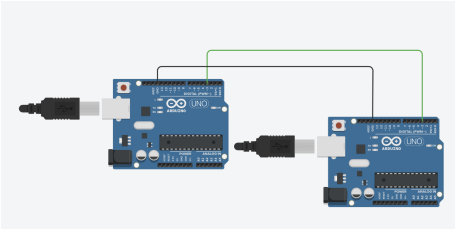

Figura 9: Conexiones del Arduino

```
{
  Serial.begin(9600);
  pinMode(LED_BUILTIN1, OUTPUT);
}

void loop()
{
  digitalWrite(LED_BUILTIN1, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  //digitalWrite(LED_BUILTIN1, LOW);
 // delay(1000); // Wait for 1000 millisecond(s)
}
```

Para el arduino receptor, su pin digital tendría que estar definido como input y se inicializa como 0, para que no corrompa o cambie los datos que van entrando, como en el Arduino Emisor, se hace una pausa del mismo tiempo entre cada lectura.

```
// C++ code
//

const int LED_BUILTIN2 =3;
int val=0;

void setup()
{
  Serial.begin(9600);
  pinMode(LED_BUILTIN2, INPUT);
  digitalWrite (LED_BUILTIN2, LOW);

}

void loop()
{
  delay(1000); // Wait for 1000 millisecond(s)
  val = digitalRead(LED_BUILTIN2);
```

```
  Serial.println(val);
}
```

Para la conexión entre los dos arduinos utilizando el circuito integrado 74HC595, del arduino emisor se paralelizan los datos enviados en el circuito, como se había explicado en la implementación 1, se conectan con resistencias de 560 ohmios. En el arduino receptor se reciben los datos una vez mediante los puertos 2-10, y se transforman en int de vuelta mediante un for. Así, se pasan los mismos valores entre los dos.
El circuito se ve así: (Figura 10)

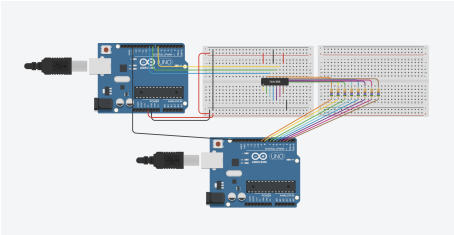

Figura 10: Conexion de Arduinos 74HC595

El código del emisor sería parecido al visto en la implementación 1, con la adición de que se ejecuta solo una vez utilizando aux y utilizando la función ledLineWrite() que se reutilizará para cuando se implemente con el uso de arreglos:

```
    //Arduino Emisor
// C++ code
//
#define ORCLK 8
#define SRCLK 7
#define num 9
int aux1=0;
void setup()
{
  Serial.begin(9600);
  pinMode(num, OUTPUT);
  pinMode(ORCLK, OUTPUT);
  pinMode(SRCLK, OUTPUT);
}

void loop()
{
  int number=108; //Menor de 255
  if(aux1==0){
    ledLineWrite(number);
```

```
    aux1=1;}
}

void ledLineWrite(int Sf1)
{
   shiftOut(num, SRCLK, MSBFIRST, Sf1);
   digitalWrite(ORCLK, HIGH);
   digitalWrite(ORCLK, LOW);
}
```

Para el arduino receptor, se utilizan los pines de 2 a 9 para recibir información, y para revisar que solo se haga el print del número una vez, se utiliza la variable aux, en este código se guarda el valor de cada pin en un arreglo de enteros, que se convierte en un int que guarda el numero enviado desde el emisor.

```
    C++ code
// Arduino Receptor
int arr[8]={0};
int aux=0;
void setup()
{
  Serial.begin(9600);
  for (int i=2;i<10;i++){
    pinMode(i,INPUT);
  }
}

void loop()
{
  delay (1000);
  for (int i=2;i<10;i++){
   arr[i-2]=digitalRead(i);
  }
  if(aux == 0){
  int number=0;
  for(int j=0;j<8;j++){
   number=number*2+arr[j];
  }
  Serial.print(number);
  aux=1;}
}
```

3.3. Implementación 3

Se le hacen leves cambios a los códigos de los arduinos y se añade un arreglo con números aleatorios al arduino emisor para ver las posibles soluciones y se empieza a trabajar en el sistema de desencriptación.

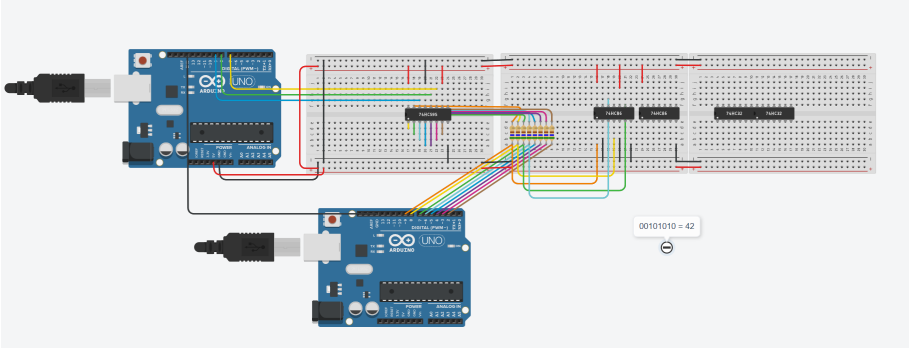

Figura 11: Sistema de desencriptación

```
// C++ code
    // Arduino Emisor

#define ORCLK 8
#define SRCLK 7
#define num 9
#include <stdlib.h>
#include <time.h>

int aux1=0;
void setup()
{
  Serial.begin(9600);
  pinMode(num, OUTPUT);
  pinMode(ORCLK, OUTPUT);
  pinMode(SRCLK, OUTPUT);
  randomSeed(analogRead(A2));
  }

void loop()
{
  if(aux1==0){
  int size=8; //int size=rand()%20;
  //unsigned char array[size]={0};
```

```
  unsigned char array []={42,42,42,42,42,42,42,42};
  //El array de arriba se puede utilizar para pruebas
  //for(int i=0;i<size;i++) //Creacion del arreglo
  //   array[i]=random(256);
  for (int i=0;i<size;i++){ //Envio arreglo a 74hc595
    ledLineWrite(array[i]);
    delay(100);
  }
    aux1=1;}
}

void ledLineWrite(unsigned char Sf1)
  //Solo se permite la entrada de numeros menores de 255
{
   shiftOut(num, SRCLK, MSBFIRST, Sf1);
   digitalWrite(ORCLK, HIGH);
   digitalWrite(ORCLK, LOW);
}
-------------- Codigo Arduino receptor -----------------
// C++ code
    // Arduino Receptor

char arr[8]={0};
int aux=0;
void setup()
{
  Serial.begin(9600);
  for (int i=2;i<10;i++) //Se especifican los pines de 2 a 9 como input
    pinMode(i,INPUT);
}
void loop()
{
  //delay (1000);
  for (int i=2;i<10;i++){
   arr[i-2]=digitalRead(i);
  }
  int number=0;
  for(int j=0;j<8;j++){
   number=number*2+arr[j];
  }
  Serial.print(number);
  Serial.print(' ');
  //aux=1;
  delay(100);

}
```

Versión final

Se adjunta el resultado del circuito de desencriptación, se corrigen los errores que ocurrieron a medida que se fue desarrollaron el código y se empieza a ensamblar los requerimientos de la implementación 4.

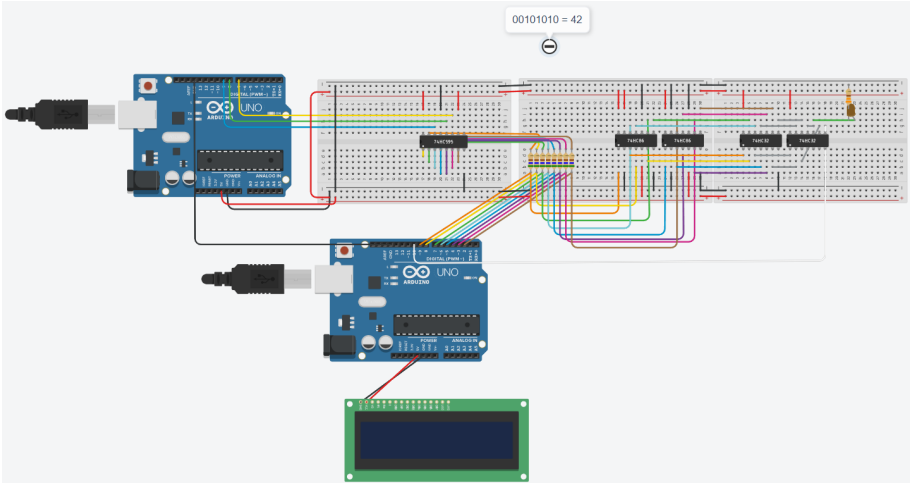

Figura 12: Término de implementación 3

```
    //Arduino Emisor
// C++ code
//
#define ORCLK 8
#define SRCLK 7
#define num 9
#include <stdlib.h>
#include <time.h>

int aux1=0;
void setup()
{
  Serial.begin(9600);
  pinMode(num, OUTPUT);
  pinMode(ORCLK, OUTPUT);
  pinMode(SRCLK, OUTPUT);
  randomSeed(analogRead(A2));

}
```

```
void loop()
{
  if(aux1==0){
  int size=18; //int size=rand()%20;
  //unsigned char array[size]={0};
  unsigned char array[]={42,2,52,38,42,64,1,2,4,9,9,42,12,1,42,1,2,4};
  //El array de arriba se puede utilizar para pruebas
  //for(int i=0;i<size;i++) //Creacion del arreglo
  //   array[i]=random(256);
  for (int i=0;i<size;i++){ //Envio arreglo a 74hc595
    ledLineWrite(array[i]);

  }
    aux1=1;}
}

void ledLineWrite(unsigned char Sf1)
  //Solo se permite la entrada de números menores de 255
{

   shiftOut(num, SRCLK, MSBFIRST, Sf1);
   digitalWrite(ORCLK, HIGH);
   digitalWrite(ORCLK, LOW);
   delay(100);
}

-------------- Codigo Arduino receptor -----------------

// C++ code
// Arduino Receptor
#define clave 10
char arr[8]={0};
int aux=0, key=0, c=0;
void setup()
{
  Serial.begin(9600);
  for (int i=2;i<10;i++) //Se especifican los pines de 2 a 9 como input
    pinMode(i,INPUT);
  pinMode(clave,INPUT);
}

void loop()
{
  delay (100);
  /*for (int j=2;j<10;j++){ //creacion de arreglo 0 y 1
    arr[j-2]=digitalRead(j);}
```

```
  int number=0;
  for(int j=0;j<8;j++) //Pasar de arreglo a int
   number=number*2+arr[j];
  key=digitalRead(clave);
  if (key==0)
    Serial.print('.');
  Serial.print(number);
  Serial.print(' ');*/
  key=digitalRead(clave);
  if (key==0)
    c++;
  if (c==2){
  for (int i=0;i<3;i++){
    delay(100);
    for (int j=2;j<10;j++){ //creacion de arreglo 0 y 1
      arr[j-2]=digitalRead(j);}
  int number=0;
  for(int j=0;j<8;j++) //Pasar de arreglo a int
   number=number*2+arr[j];
  Serial.print(number);
  Serial.print(' ');
  c=0;
  }
  }
}
```

3.4. Implementación 4

Desarrollo del código y funcionamiento de la pantalla lcd.

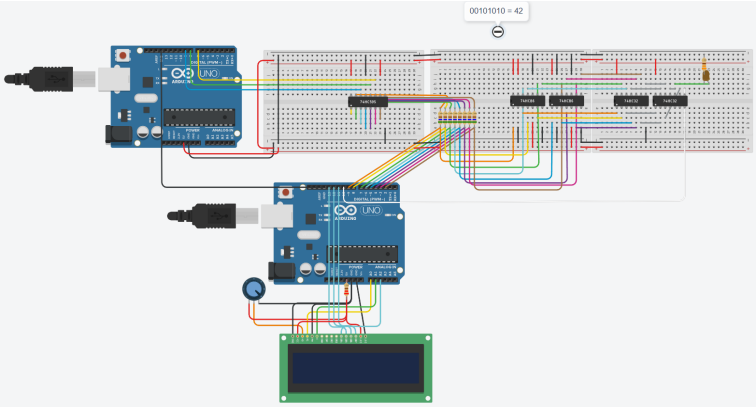

Figura 13: Término de implementación 3

```
    //Arduino Emisor
// C++ code
//
#define ORCLK 8
#define SRCLK 7
#define num 9

int aux1=0;

void setup()
{
  Serial.begin(9600);
  pinMode(num, OUTPUT);
  pinMode(ORCLK, OUTPUT);
  pinMode(SRCLK, OUTPUT);
  randomSeed(analogRead(A2));

}

void loop()
{
  if(aux1==0){
  delay(100);
  //Manera manual, se puede utilizar para pruebas NO PUEDE TERMINAR EN 42
  unsigned char array[]={12,245,43,43,42,2,52,38,42,110,115,65,4,9,9,42,12,
```

```
  1,42,65,65,65,41,42,42,111,112,103};
  int size=sizeof(array)/sizeof(*(array));
  //Manera random
  //El array de arriba se puede utilizar para pruebas
  //int size=random(200);
  //unsigned char array[size]={0};
  //for(int i=0;i<size;i++) //Creacion del arreglo
  //   array[i]=random(256);
  for (int i=0;i<size;i++){ //Envio arreglo a 74hc595
    ledLineWrite(array[i]);

  }
    aux1=1;}
}

void ledLineWrite(unsigned char Sf1)
  //Solo se permite la entrada de numeros menores de 255
{

   shiftOut(num, SRCLK, MSBFIRST, Sf1);
   digitalWrite(ORCLK, HIGH);
   digitalWrite(ORCLK, LOW);
   delay(100);
}

-------------- Codigo Arduino receptor -----------------

#include <LiquidCrystal.h>

// C++ code
// Arduino Receptor
#define clave 10
char arr[8]={0};
int aux=0, key=0, c=0;
LiquidCrystal lcd(A0,A1,A2,13,12,11);
int pos=0;
char* results=new char[3];
void setup()
{
  lcd.begin(16, 2);
  Serial.begin(9600);
  for (int i=2;i<10;i++) //Se especifican los pines de 2 a 9 como input
    pinMode(i,INPUT);
  pinMode(clave,INPUT);
}
```

```
void loop()
{
  delay (100);
  key=digitalRead(clave);
  if (key==0)
    c++;
  if (c==2){
  for (int i=0;i<3;i++){
    delay(100);
    for (int j=2;j<10;j++){ //creacion de arreglo 0 y 1
      arr[j-2]=digitalRead(j);}
  int number=0;
  for(int j=0;j<8;j++) //Pasar de arreglo a int
   number=number*2+arr[j];
  *(results+i)=char(number);
  Serial.print(number);
  Serial.print(' ');
  c=0;
  }
  lcd.clear();
  lcd.setCursor(0,0);
  for (int i=0;i<3;i++){
   lcd.print(int(*(results+i)));
    lcd.print(' ');
  }
  lcd.setCursor(0,1);
  for (int i=0;i<3;i++){
   lcd.print(*(results+i));
   lcd.print(' ');
  }
  }
}
```

Referencias