# PARCIAL 1
# INFORMATICA II

**K. Lopez,** ***estudiante Ing.Telecomunicaciones.***
**A. Salazar ,** ***Profesor de teoría.***

*Departamento de Ingeniería Electrónica y Telecomunicaciones- Universidad de Antioquia*

Informe realizado el 17 de septiembre del presente año, informe que corresponde al primer parcial del curso de INFORMATICA II, cuya entrega final es el día 25 de septiembre.

**Abstract:** El objetivo principal de este documento es dar a conocer un poco la solución que se propone al problema que fue establecido en este primer parcial del curso, el cual consiste en crear un circuito capaz de cumplir ciertas funcionalidades con el manejo de una matriz de 64 LEDS

## 1. Introducción

El presente documento tiene como fin ilustrar la solución propuesta al primer parcial del curso de informática 2, el cual consiste en elaboración de un circuito utilizando el software tinkercad que tenga la capacidad de controlar una matriz de 64 LEDS , y finalmente pueda cumplir ciertas especificaciones y funcionalidades pedidas. Cabe resaltar que la solución propuesta se desarrolla utilizando el lenguaje de c++, para lograr este objetivo es sumamente importante entender el funcionamiento del integrado 74HC595, así como tener claro algunos conceptos vistos en las sesiones sobre el lenguaje [c++] y el manejo del simulador online tinkercad.

## 2. Materiales utilizados en el montaje del circuito

- Arduino uno
- Protoboard
- Integrado 74HC595 X8
- LED X64 - Resistencia 100Ω
- Cable UTP o cables tipo jumpers para las conexiones

## 3. Desarrollo del proyecto

El objetivo principal del proyecto es crear un circuito que tenga la capacidad de controlar 64 LEDS, y de esta manera cumplir ciertas funciones(especificaciones). Antes de intentar buscar una solución al problema, se profundizo en el funcionamiento del integrado 74HC595, este se utiliza comúnmente como un registro de desplazamiento de 8 bits con salida en serie.
Después de tener claro el funcionamiento del integrado, se elaboro inicialmente un circuito el cual usaba dos integrados de este tipo para controlar los 64 leds y usaba 6 pines digitales del arduino, uno de los problemas que se tuvo con la implementación de este circuito fue que no fue fácil controlar cada LED de forma individual, así se debía de buscar una mejor solución.
La gran pregunta era como controlar fácilmente cada led mediante el uso del integrado, luego de investigar un poco, se encuentra que un uso común de este integrado cuando se desea tomar

control de varios leds, es usar varios de estos en serie (cascada), es decir la salida del uno es la entrada del otro, por lo que se opto por crear un nuevo circuito que utilizara exactamente 8 de estos integrados y 3 pines del arduino.
Luego de hacer el montaje del circuito en tinkercad del nuevo diseño, la pregunta que se presento fue de como utilizar (programar) correctamente los 8 integrados para que cada uno de ellos realizara el desplazamiento de un byte correctamente.
Una herramienta que nos facilito este trabajo es la función shifOut() la cual recibe 4 parámetros, los datos del pin SER (entrada serial), pin del ciclo del reloj de desplazamiento, y una palabra reservada LSBFIRST o MSBFIRST, en este caso se utilizo la primera pues se quería hacer un desplazamiento comenzando por el bit menos significativo, el ultimo parámetro de la función es la secuencia en binario que se desea obtener a la salida de los led, 1 representa el estado del led en alto [encendido] y 0 el estado del led en bajo [apagado].
Como el circuito esta constituido por 8 integrados cuando se quiere por ejemplo obtener un patrón en los leds es necesario utilizar la función shiftOut() la misma cantidad de veces y pasarle en cada una las diferentes representaciones en binario (leds altos y apagados) que se desea en cada fila de la matriz de 64 leds, usando el principio anterior se crea la función verificación teniendo en cuenta que por cada fila de la matriz se llama una vez dicha función.
Luego se creo una función llamada imagen la cual tiene como principal objetivo establecer un patrón (ingresado por el usuario) en los leds recibe 8 parámetros todos enteros, en donde cada numero entero representa un patrón correspondiente a una fila en especifico, estos patrones se pueden representar en números enteros que están en el intervalo de [0-255], para ello fue de gran ayuda la función anterior shiftOut() pues simplemente era saber cual era el patroncito correspondiente de cada fila de la matriz y utilizar dicha función de manera análoga.
También en esta primera entrega se logro implementar una función encargada de mostrar en la matriz de 64 LEDS los cuatro patrones en pantalla que están preseleccionados, en ella se crea un arreglo de enteros en donde cada elemento representa un patroncito correspondiente a una fila, así el tamaño del arreglo sera de 8*4=32, para colocar cada patrón completo en la matriz de LEDS se usara la función imagen descrita anteriormente, con el objetivo de ir poniendo paulatinamente cada patrón a través de sus correspondientes patroncitos de sus filas.

Cabe resaltar que para cumplir con el objetivo de mostrar un patrón que el usuario quiere también se implemento algunas otras funciones como: la función encargada de tomar los datos por el puerto serial y guardarlos en un arreglo. Otra función fue la encargada de convertir un patroncito correspondiente a una fila en binario a su representación en decimal. Creemos que estas funciones serán de gran ayuda para cumplir con el desarrollo del proyecto.

# 4. Algunas imágenes del código implementado hasta el momento

```
/*
pinData:Entrada serial
pinLatch:Reloj de registro de salida
pinClock:Reloj de registro de desplazamiento
*/

//Pines utilzados del arduino:
int pinData  = 6;
int pinLatch = 7;
int pinClock = 8;

int patrones;
float tiempo;
float tiempo1;
int *ptrp=&patrones;
float *ptrt=&tiempo;

/*Función verificacion: para encender todos los leds*/
void verificacion(){

   shiftOut(pinData, pinClock, LSBFIRST, 255);
   shiftOut(pinData, pinClock, LSBFIRST, 255);
   shiftOut(pinData, pinClock, LSBFIRST, 255);
   shiftOut(pinData, pinClock, LSBFIRST, 255);
   shiftOut(pinData, pinClock, LSBFIRST, 255);
   shiftOut(pinData, pinClock, LSBFIRST, 255);
   shiftOut(pinData, pinClock, LSBFIRST, 255);
   shiftOut(pinData, pinClock, LSBFIRST, 255);
  //se activa el reloj para el registro de salida en todos los integrados
   digitalWrite(pinLatch, HIGH);
   digitalWrite(pinLatch, LOW);
}
```

Fig. 1. función verificación

```
/* Funcion imagen: Para establecer un determinado patron en la matriz de leds, para ello
cada fila se compone de un patroncito cuya representacion es un numero entero,se establece
el patroncito en cada fila para finalmente obtener el patron completo*/

void imagen(int led1,int led2,int led3,int led4,int led5,int led6,int led7,int led8){

  //debemos de tener cuidado con el orden del ultimo argumento que recibe la funcion shifOut
  //para poder poner el patroncito de cada fila en el orden correcto.

   shiftOut(pinData, pinClock, LSBFIRST, led8);
   shiftOut(pinData, pinClock, LSBFIRST, led7);
   shiftOut(pinData, pinClock, LSBFIRST, led6);
   shiftOut(pinData, pinClock, LSBFIRST, led5);
   shiftOut(pinData, pinClock, LSBFIRST, led4);
   shiftOut(pinData, pinClock, LSBFIRST, led3);
   shiftOut(pinData, pinClock, LSBFIRST, led2);
   shiftOut(pinData, pinClock, LSBFIRST, led1);
  //se activa el reloj para el registro de salida en todos los integrados
   digitalWrite(pinLatch, HIGH);
   digitalWrite(pinLatch, LOW);
}
```

Fig. 2. función imagen

```
void patrones_preseleccionados(){
  //el usuario elegi el tiempo entre patrones
    Serial.print("tiempo entre patrones: ");
    char *tiempo_patrones=new char[1];
    while(true){
      if (Serial.available()){
        while(Serial.available()>0)
        {
         delay(50);
         *tiempo_patrones=Serial.read();
        }
        break;
      }
    }
    Serial.print(*tiempo_patrones);

        // guardamos en un arreglo los patroncitos correspondientes a las filas de todos los 4 patrones.
        int patroncitosFilas []={24,60,126,255,255,126,60,24,129,66,36,24,24,36,66,129,219,219,109,109,219,219,109
          ,109,240,120,60,30,30,60,120,240};
        while(true){
            for(int k=0;k<(4*8)-1;k=k+8){
                  //colocamos un patron completo
                  imagen(patroncitosFilas[k],patroncitosFilas[k+1],patroncitosFilas[k+2],patroncitosFilas[k+3],patroncitosFilas[k+4],
                         patroncitosFilas[k+5],patroncitosFilas[k+6],patroncitosFilas[k+7]);
                  //Esperamos un tiempo para poner el siguiente
                  delay(atoi(tiempo_patrones)*1000);
             }
        }
}
```

Fig. 3. función para los patrones preseleccionados

```
//Función que permite convertir un arreglo que contiene un número binario a un entero decimal.
int BinToInt (char ing[8]){

  int i = 0;  //Variable para el contador
  int n = 0;  //Variable para calcular el resultado

  while ( ing[i] == '0' || ing[i] == '1' ) {
    if ( ing[i] == '0' )
      n <<= 1;
    else {
      n ^= 1;
      n <<= 1;
    }
  ++i;
  }
  n >>= 1;
  return(n);

}
```

Fig. 4. Implementación de una función que nos puede ser útil