INFORME PARCIAL INFORMÁTICA 2

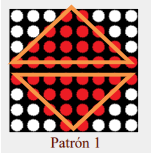
Participantes: David Alejandro Morón Acacio y Luisa María Bohórquez Ardila

1. ***Análisis del problema y consideraciones para la alternativa de solución propuesta***

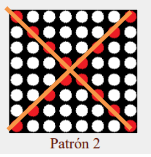
Se tiene como punto de partida el cómo representar una matriz en Arduino mediante el lenguaje de programación que maneja (el uso de arreglos, punteros, la memoria dinámica y la memoria estática para guardar y reproducir el comportamiento de los LEDs). Después de esa familiarización con el tipo de estructuras de control y de datos y de tener una fluidez con su manejo y sus definiciones al momento de usarlas (nótese que nos referimos a la parte de la codificación), resta el tener en consideración los componentes electrónicos a utilizar y concatenar (nótese que nos referimos a las características del hardware disponible).

1. ***Esquema donde describa las tareas que usted definió en el desarrollo de los algoritmos.***

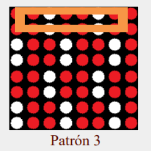
**Patrón 1:** Haremos un triángulo teniendo en cuenta las filas y luego replicaremos ese código de manera decreciente.



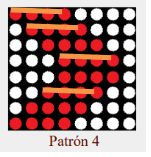
**Patrón 2:** Analizamos la figura por los extremos, a medida que va iterando la función vamos desplazando la posición de los leds encendidos unos a la izquierda(restando) y otros a la derecha(sumando).



**Patrón 3:** Haremos que cada dos renglones de la matriz se impriman, dado que son el mismo patrón, y a cada renglón se le encienden dos leds y se apaga uno.



**Patrón 4:** Encendemos 4 leds en el primer extremo y lo vamos desplazando a la derecha hasta la cuarta iteración, repetimos el patrón tal cual en la quinta y de la sexta a la octava desplazamos los leds encendidos a la izquierda



**Para encender toda la matriz,** vamos a recorrer con un arreglo cada variable asociada a un led se enciende.

Nótese que fueron nuestras primeras ideas sin tener el conocimiento suficiente del sistema sobre el que íbamos a trabajar. Luego de tener la claridad pertinente al respecto, concluimos que el único esquema que debemos definir es el de producir el efecto visual mediante un ciclo *for* al encender todas las filas y apagar cada led enviando un pulso a los leds mediante las columnas, y diseñar las figuras a partir de matrices, las cuales iterarán en el ciclo *for*, y sus datos entrarán por el serial y saldrán en paralelo gracias al 74HC595

1. ***Algoritmos implementados***

***Ideas iniciales:***

**Código para encender toda la matriz:**

|  |
| --- |
| char high='+', low='-';  char matriz[8][8];  for(int i=0; i<8; i++){  for(int j=0;j<8;j++){  matriz[i][j]= high;  }  }  //Imprimir la matriz  for(int i=0; i<8; i++){  for(int j=0;j<8;j++){  char posicion= matriz[i][j];  char espacio= ' ';  cout<< espacio<< posicion;  }  cout<<endl;  } |

**Código para el primer patrón:**

|  |
| --- |
| char high='+', low='-';  char matriz[8][8];  int cotainferior=3, cotasuperior=5;  //primera parte de la figura  for(int i=0; i<5; i++){  for(int j=0;j<8;j++){  if(j>=cotainferior and j<cotasuperior){  for(int control2=j; control2<cotasuperior;control2++){  matriz[i][j]=high;}  }else{  matriz[i][j]=low;  }  }  if(i<4){  cotainferior-=1;  cotasuperior+=1;  }    }  //segunda parte de la figura  cotainferior=1; cotasuperior=7;  for(int i=5; i<8; i++){  for(int j=0;j<8;j++){  if(j>=cotainferior and j<cotasuperior){  for(int control2=j; control2<cotasuperior;control2++){  matriz[i][j]=high;}  }else{  matriz[i][j]=low;  }  }  cotainferior+=1;  cotasuperior-=1;  }    //Imprimir la matriz  for(int i=0; i<8; i++){  for(int j=0;j<8;j++){  char posicion= matriz[i][j];  char espacio= ' ';  cout<< espacio<< posicion;  }  cout<<endl;  } |

**Código para el segundo patrón:**

|  |
| --- |
| char high='+', low='-';  char matriz[8][8];  int extremoIzquierdo=0, extremoDerecho=7;  //Figura  for(int i=0; i<8; i++){  for(int j=0;j<8;j++){  if(j==extremoIzquierdo){  matriz[i][j]=high;  }else if(j==extremoDerecho){  matriz[i][j]=high;  }else{  matriz[i][j]=low;  }  }  extremoIzquierdo+=1;  extremoDerecho-=1;  }  //Imprimir la matriz  for(int i=0; i<8; i++){  for(int j=0;j<8;j++){  char posicion= matriz[i][j];  char espacio= ' ';  cout<< espacio<< posicion;  }  cout<<endl;  } |

**Código para el tercer patrón:**

|  |
| --- |
| int main()  {  char high = '+', low='-';  char matriz[8][8];  int i = 0;  int sep1 = 0,sep2 = 2, sep3 = 3, sep4 = 5, sep5 = 6; //señalamos los leds apagados  //introducimos el ciclo for que se encarga de llenar la matriz  while(i >=0 && i<8){  //hacemos los ciclos que van a crear las filas  for(int a=0; a<2; a++){  for(int j=0; j<8; j++){  if(j==sep2 || j==sep4){  matriz[i][j]=low;  }else{  matriz[i][j]=high;  }  }  i += 1;  }  for(int a=0;a<2;a++){  for(int j=0; j<8; j++){  if(j==sep1 || j==sep3 || j==sep5){  matriz[i][j]=low;  }else{  matriz[i][j]=high;  }  }  i += 1;  }  }  //imprimimos la matriz  for(int i=0; i<8; i++){  for(int j=0;j<8;j++){  char posicion= matriz[i][j];  char espacio= ' ';  cout<< espacio<< posicion;  }  cout<<endl;  }  } |

**Código para el cuarto patrón:**

|  |
| --- |
| char high='+', low='-';  char matriz[8][8];  int cotainferior=0, cotasuperior=4;  //primera parte de la figura  for(int i=0; i<5; i++){  for(int j=0;j<8;j++){  if(j>=cotainferior and j<cotasuperior){  for(int control2=j; control2<cotasuperior;control2++){  matriz[i][j]=high;}  }else{  matriz[i][j]=low;  }  }  if(i<3){  cotainferior+=1;  cotasuperior+=1;  }  }  //segunda parte de la figura  cotainferior=2; cotasuperior=6;  for(int i=5; i<8; i++){  for(int j=0;j<8;j++){  if(j>=cotainferior and j<cotasuperior){  for(int control2=j; control2<cotasuperior;control2++){  matriz[i][j]=high;}  }else{  matriz[i][j]=low;  }  }  cotainferior-=1;  cotasuperior-=1;  }  //Imprimir la matriz  for(int i=0; i<8; i++){  for(int j=0;j<8;j++){  char posicion= matriz[i][j];  char espacio= ' ';  cout<< espacio<< posicion;  }  cout<<endl;  } |

***Ideas definitivas***

|  |
| --- |
| //Definir nuestro setup  void setup()  {  Serial.begin(9600);  // 74HC595  pinMode(data, OUTPUT); // data  pinMode(store, OUTPUT); // store  pinMode(shift, OUTPUT); // shift    }  //Algoritmo para los patrones(donde definimos con los componentes que se apagan y agregados el ciclo que maneja la cantidad de tiempo que va a imprimirse la figura y el ciclo que imprime la figura como tal)  void patron1()  {  int columna[8] = {B11100111,  B11000011,  B10000001,  B00000000,  B00000000,  B10000001,  B11000011,  B11100111};    while(tiempo\_transcurrido <= tiempo\_parámetro  {  for(int i = 0; i < 8; i++)  {  digitalWrite(store, LOW);  shiftOut(data, shift, LSBFIRST, fila[i]);  shiftOut(data, shift, LSBFIRST, columna[j]);  digitalWrite(store, HIGH);  j++;  }  j = 0;  } |

Siendo esta nuestra idea a implementar, sólo resta diseñar los patrones a generar y ponerlos en la secuencia que pide la función publik(), teniendo en cuenta los tiempos de apagado y encendido entre los intervalos (entre otras cosas que exige la naturaleza del código, como las validaciones de entrada y el procesamiento de los datos.

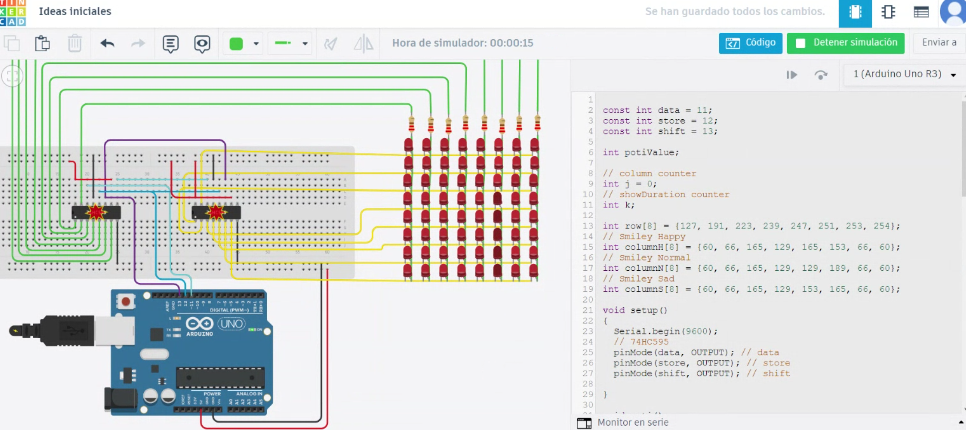
1. ***Problemas de desarrollo que afrontó***

La primera piedra con la que tropezamos durante el desarrollo del trabajo fue el aprender a manejar adecuadamente el sistema de Arduino y del integrado 74HC595. El tener en cuenta el funcionamiento de las entradas seriales y salidas en paralelo, el cómo recibir y tratar esa información para lograr el resultado deseado y el cómo extrapolar lo que ya sabíamos a esta aplicación, y los demás aspectos electrónicos involucrados (como programar los voltajes que soportan los leds).

En el procedimiento de ensamblaje, a nuestro primer intento, tanto los integrados como los leds sufrieron una sobrecarga y se quemaron. Luego aumentamos el valor de las resistencias aplicadas a cada led y los integrados no se quemaron ni los leds (a excepción de algunos debido a un led tenía mal puesta una conexión).

Luego de probar con detenimiento las propiedades y las características del integrado, nos dimos cuenta de que el generar los patrones solicitados era más sencillo y estaba más alejado de cualquier código que se nos hubiera ocurrido. Por tanto, tuvimos que adecuarnos rápidamente al tipo de dinámica que presentaba el sistema a diferencia de la dinámica que inocentemente estábamos planteando.

Acerca de esas mismas características, aprendimos cómo funcionan a nivel de efectos ópticos las salidas en paralelo de los 74HC595, y de inmediato pudimos visualizar el tipo de patrón que nos ofrecía esta herramienta. Después de comprender mejor cómo enviar y a dónde enviar cada dato mediante el serial, nos dedicamos a rediseñar los patrones (siendo impresiones estáticas que podían ir dentro de una función de manera que pudieran llamarse cuando fueran solicitadas por el usuario) y a pensar el modelo de funcionamiento de las entradas que el usuario debía ingresar para ver la imagen que él quisiera.



Nos enfrentamos al problema de orden de conexión entre las salidas (estaban encendiendo al revés y tuvimos que re-cablear el sistema), pero al final logramos el orden que buscábamos recurriendo al concepto base de paso de energía y conseguimos no quemar ninguna parte del cableado. Luego afrontamos los problemas de recepción de información a través del Arduino, y después de conseguir sortearlos, llegamos al resultado que planeábamos.

Después de lograr que las salidas funcionaran como lo esperado, nos dedicamos al problema de ingresar los valores para que la matriz encendiera como lo pidiera el usuario. Luego de mucho investigar el cómo el sistema recibe por el monitor un dato, vimos la manera de procesar un dato después de pedirlo al usuario (cómo hacer la conversión de variables tipo Char a Int y cómo hacer las validaciones, por ejemplo) para usarlo en el sistema y que se imprimiera como entrara. Hecho esto, nos dedicamos a encapsular el código en las funciones posibles y a reorganizar el diseño del código para hacer las pruebas finales y ultimar los detalles que hicieran falta para entregar.

Preguntas formuladas en el proceso:

1. ¿Qué ventajas tiene la memoria dinámica ante el stack?
2. ¿Para qué su usa el Git-ignore?
3. ¿Cuál es el formato de entrada de los datos para los LED’S?
4. ¿Cómo usar correctamente el integrado 74HC595?
5. ¿Cómo calcular la resistencia para que la matriz de leds genere el mayor brillo posible?
6. ¿Cómo pedirle al usuario que ingrese un dato?
7. ¿Cuáles son las distintas formas de estructurar un integrado? ¿En qué afecta esto a las salidas de datos?
8. ¿Cómo convertir el tiempo de duración de visualización que ingresa el usuario a las unidades que maneja el Arduino?
9. ¿Cómo podemos transponer una matriz en este lenguaje de programación?
10. ¿Cuál es la esencia óptica de la velocidad de la impresión de datos y cómo podemos aprovecharla?
11. ¿Cuál es la influencia del orden de conexión los ánodos y de los cátodos?
12. ¿Cómo funcionan los prints y los inputs en el monitor serial?

1. ***Evolución de la solución y consideraciones a tener en cuenta en la implementación.***

Vamos a utilizar un Arduino y un integrado electrónico conocido como 74HC595, que nos va a permitir generar datos en paralelo a partir de datos en serie. De esta forma, podemos generar ocho salidas con sólo tres pines digitales del Arduino. No obstante, dado el tamaño de la matriz, se debe recurrir a una definición electrónica de la información y también a la concatenación de otro 74HC595 al anterior (cosa que es posible por un lujo del mismo 74HC595). De esta manera, podríamos manejar prácticamente de forma simultánea a cada led en la matriz, facilitando los problemas de impresión consecutiva y de imagen a mostrar.

Entonces, mediante el serial se ingresan los primeros ocho bits que van a representar las columnas de la matriz (estos ocho bits saldrán por el segundo 74HC595), y sus salidas estarán conectadas a los ánodos de los leds en la matriz, y los siguientes ocho bits ingresados por el serial serán los que representan las filas de la matriz (estos ocho bits saldrán por el primer 74HC595) y sus salidas estarán conectadas a los cátodos de la matriz, de manera que si un valor es alto, toda la fila/columna se enciende, y si ambos valores son altos, toda la fila/columna se apaga. Por decirlo de alguna forma, se van a apagar todos los componentes que tengan ambos valores altos, y se encenderán todos los componentes que tengan sólo un valor alto.

Sin embargo, a manera de aclaración del concepto (ya que nos dimos cuenta de lo que la anterior propuesta no cubría nuestra necesidad y redefinimos la estrategia), lo que vamos a hacer es imprimir cada fila independientemente en la matriz (es decir, imprimiremos ocho estados de la matriz) iterando sobre un ciclo *for* para que la velocidad de la impresión del sistema haga lucir una imagen estática, cuando en realidad se está imprimiendo específicamente cada fila con su respectivo patrón (generado por la influencia de las columnas, que apagan los leds encendidos según se pida, ya que las filas siempre están encendidas).

Para la función de ingresar una imagen por el usuario logramos recibir de una forma cómoda para el usuario los bits que van a apagar/encender cada fila como él lo precise, pidiendo entonces ocho veces los datos para generar toda la matriz (pidiendo que ingrese la información con la convención de 0 para encendido y 1 para apagado). Después de eso, se guarda el arreglo que será enviado al integrado para generar la figura. Para verificar que la matriz de leds esté encendida, podemos generar una matriz (con base a punteros y memoria dinámica) que nos garantice que la matriz se encienda. Para los patrones, proporcionamos las formas de las figuras para imprimirlos en secuencia (nótese que se pueden generar con código, el cual dejamos evidenciado en la parte superior del documento, pero no los ingresamos al proyecto en pro de disminuir la carga que puede presentar la máquina). Posteriormente, organizamos los tiempos de los menús y la interfaz para el usuario, y nuestro proyecto queda listo para ser usado.

Enlace al proyecto en tinkercard:

<https://www.tinkercad.com/things/anEjLFuSUSK>