**Informe Parcial Informática II**

**Andres Felipe Agudelo Zapata**

**Planteamiento del problema:**

El objetivo es crear una matriz de leds de 8x8 que pueda mostrar diferentes patrones. Para esto se debe hacer uso de un Arduino UNO y el desarrollar en lenguaje C++ los diversos problemas planteados con el uso de los conceptos aprendidos en clase e investigación propia.

.

**Análisis de los problemas:**

* **Análisis Problema 1:**

Se debe de implementar un circuito en tinkercard que nos permite controlar 64 leds de la forma más eficientemente posible y con menor uso de pines digitales posibles, siendo además capaces de controlar estos leds y poder decidir sobre el estado de cada uno de ellos. Se decidió implementar una matriz de ánodo común conectando las filas a los 5v y las columnas a tierra (GND). Para el código se generará una matriz 8\*8 en C++ y se le asignará un puntero a su posición (0,0), que permita recorrerla por dos ciclos *for.*

* **Análisis Problema 2:**

Se debe de implementar una función que de la orden a los 64 leds de prenderse a la misma vez y de apagarse también en sincronía esto con el fin de poder verificar que todos los leds estén funcionando correctamente.

En el tinkercard se pude lograr esto mediante el integrado 74HC595, que al recibir una entrada en binario divide esta secuencia de 8 bits en 8 salidas con la que cada una va a una fila, nótese que al ingresar el numero 255 (en binario: 1111111) esta manda una señal de 1 a cada fila, permitiendo prender los 64 leds de la matriz.

* **Análisis Problema 3:** Para darle la libertad al usuario de imprimir lo que desee, vamos a preguntarle al usuario sobre el estado de cada led es decir sobre si desea que este encendido o apagado. Y al final terminara por imprimir la imagen deseada
* **Análisis Problema 4:**

a). Para generar este patrón se nos ocurre el generar dos ciclos anidados con 2 variables controladoras, el contador de espacios y el contador de leds, en un inicio el contador de leds va aumentando de dos en dos mientras el contador de espacios disminuye también de dos en dos, esto hasta que el contador de espacios es 0 y el contador de leds sea 8 en cuyo caso se observa que la segunda mitad de la matriz corresponde a un reflejo de la primera mitad, siendo la cuarta fila igual a la tercera, la quinta igual a la tercera y así mismo hasta llegar a la fila ocho que es igual a la primera fila.

b). Para generar este patrón es necesario dar corriente a las columnas que cumplan que se encuentran en la misma posición de la fila evaluada y en las que cumplan la ecuación (8-j) en la que j es la posición de la fila evaluada iniciando desde j=0.

c). Este patrón nos muestra un ciclo de cuadrados 2x2 que dejan un espacio en blanco entre cada cuadrado, para facilitar la programación se tomó que la tercer y cuarta fila de cuadrados son una combinación de la primera y de la segunda, en el que la segunda fila (o cuarta) esta corrida una porción a la derecha respecto a la primera(tercera). Para dibujar este patrón es necesario aparte de los dos ciclos “*for*” que recorren la matriz el de diversos condicionales que evalúen si la fila es par, (esto debido a que las filas impares son simplemente una copia de su fila pasada) y también es necesario el utilizar una variable que nos permita evaluar si se debe dibujar en esa celda o si por el contrario se debe de dejar en blanco.

d). Al observar el problema se nos ocurrieron dos opciones diferentes, la primera era viendo como los leds encendidos iban en aumento mientras los leds apagados iban disminuyendo hasta que al llegar a la mitad se invierte el orden y los que venían en incremento pasaban a disminuir y viceversa más esta opción se vio como practica al tener en cuenta la segunda solución, la cual corresponde al observar como los leds encendidos parecieran desplazarse una posición entre cada fila, hasta llegar a la mitad en la que la segunda mitad refleja la primera mitad,

* **Análisis Problema 5**

El problema 5 no es más que el diseño de un menú en el monitor serial del Arduino que nos permita invocar las funciones desarrolladas con anterioridad siendo la parte A la invocación del problema 2, la función B la invocación del problema 3, y la función C la invocación de las 4 funciones del problema 4.

**Esquema de tareas**

Este programa cuenta con muchas tareas que han de ser desarrolladas mas las de mayor importancia son 4 que consisten en:

A).

Desarrollar una función que permita verificar el funcionamiento de los leds al proporcionarle una cantidad de repeticiones y un intervalo de tiempo, el programa ha de encender los leds por el tiempo digitado por el usuario, esto ha de ser repetido una cantidad de veces que también la digita el usuario.

B).

Permitirle al usuario imprimir cualquier imagen en la matriz de leds, esto se pude lograr al preguntarle al usuario el estado de cada fila de la matriz al preguntarle al usuario un numero entre el 0 y el 255, esto mediante la conversión de enteros a binarios haciendo que el 0 sea que no se enciende ningún led de la fila, y el 255 que se enciendan todos los leds. Además, también se deben de usar las funciones que permitan que el usuario ingrese un tiempo en el que se imprime la imagen y la cantidad de veces que quiere repetir esto.

C).

Imprimir una secuencia de los 4 patrones que se dan en el documento del parcial de forma ordenada, además de que también se debe de tener en cuenta el tiempo y cantidad de repeticiones como en las dos tareas anteriores.

D).

La última tarea importante es la de generar un menú que permita invocar las tareas anteriores par que el usuario pude decidir a cuál de las tres quiere acceder.

**Algoritmos implementados:**

Las funciones utilizadas en el desarrollo del programa fueron:

void \*verificacion(){

/\* Esta función tipo void invoca las funciones time() y la función Cantsecuencia() para conseguir el tiempo y la cantidad de repeticiones que el usuario desee para luego generar un arreglo de enteros con el numero cero que en binario es : 00000000, esto ya que esta función manda este arreglo al segundo integrado 74HC595 que controla los cátodos en el un cero significa que se prende y un uno que se apaga por lo que al mandar un arreglo con 8 ceros los 64 leds se prenden, y esto se observa al invocar la función printmatrizbasica() que se encarga de desarrollar esto en la matriz \*/

unsigned int cantsecuencia;

unsigned long t=time();

cantsecuencia=Cantsecuencia();

int bin;

int \*apuntador= new int[8];

for(int fil = 0;fil < 8; fil++){

bin=0b00000000;

apuntador[fil]=bin;

}

printmatrizbasica(t,cantsecuencia,apuntador);

}

void \*imagen(){

/\* Esta función tipo void invoca las funciones time() y la función Cantsecuencia() para conseguir el tiempo y la cantidad de repeticiones que el usuario desee para luego invocar la función convertirbin y agregar lo que retorna a un arreglo para luego mandar ese arreglo a la función printmatrizbasica() junto con el tiempo y la cantidad de repeticiones\*/

unsigned int cantsecuencia;

unsigned long t=time();

cantsecuencia=Cantsecuencia();

int n;

int \*apuntador= new int[8];

for (int i=0;i<8;i++){

n=convertirbin();

apuntador[i]=n;

}

printmatrizbasica(t,cantsecuencia,apuntador);

}

int convertirbin(){

/\*

La función pregunta al usuario por un numero entre 0 y 255, para después retornar el número que se ingresó restándole 255 y multiplicándolo por menos uno, esto es lo mismo que generar la negación en binarios de ese numero es decir cambia los unos por ceros y viceversa

\*/

int n;

Serial.print("Ingrese un numero entre 0 y 255");

Serial.print("\n");

while(Serial.available() == 0){}

n = Serial.parseInt();

if((n >= 0) && (n <= 255)) {

n=n-255;

n=n\*-1;

return n;

}else{

Serial.println("Valor incorrecto");

}

}

unsigned long time(){

/\* la función pregunta por una cantidad de segundos en la que se generar el cambio de estado de los leds de la matriz después de verificar que si se aun dato valido se multiplica el número por 1000 para pasar de milisegundos a segundos y se retorna dicho valor\*/

unsigned int t;

while(true) {

Serial.println("Ingrese cuantos segundos hasta el cambio de estado");

while(Serial.available() == 0){}

t = Serial.parseInt();

if(t > 0) {

Serial.print("El valor ingresado fue de: ");

Serial.println(t);

break;

}

else {

Serial.println("No se puede ingresar tiempo negativo intente de nuevo");

}

}

return t\*1000;

}

unsigned int Cantsecuencia(){

/\* la función pregunta por la cantidad de repeticiones que generara el programa. Después de verificar que si se da un dato valido se retorna dicho valor\*/

unsigned int cc;

while(true) {

Serial.println("Ingrese cuantas repeticiones hay que realizar: ");

while(Serial.available() == 0){}

cc = Serial.parseInt();

if(cc > 0) {

break;

}

else {

Serial.println("No se puede ingresar una cant de repecticiones negativas");

}

}

return cc;

}

void printmatrizbasica(unsigned long tiempo,unsigned int Cantsecuencia,int \*columna){

/\*

La función recibe un valor de tiempo, de repeticiones y la dirección de memoria del primer valor de un array de enteros.

Inmediatamente se carga la función se genera una variable llamada checkpoint que equivale a la cantidad de milisegundos desde que se conecta el Arduino, en un ciclo while infinito se evalúa si el tiempo actual (en milisegundos) menos el checkpoint es mayor al tiempo ingresado en la función si esto es cierto la variable state cambia su valor.

Si state es positiva entonces se le manda a los leds la señal que se enciendan en cascada con los valores que hay en el array de enteros que señala el puntero llamado columna, de lo contrario la función apaga todos los leds.

Para cerrar el programa se evalúa si la variable auxiliar cc (que cuneta la cantidad de cambios de valores de state) cumple la desigualdad cc>=((2\*Cantsecuencia)-1). Si se cumple esto entonces ya el programa se ha repetido la cantidad de veces que digito el usuario por lo que se apaga

\*/

unsigned long long checkpoint=millis();

int fila[] = {128,64,32,16,8, 4, 2,1},cc=0;

bool state= true;

while(true){

if((millis()-checkpoint)>tiempo){

checkpoint=millis();

state=!state;

cc=cc+1;

}if(cc>=((2\*Cantsecuencia)-1))

{break;}

if(state){

for(int i = 0; i < 8; i++) {

shiftOut(SER,RCLX,LSBFIRST,\*(columna+i));

shiftOut(SER,RCLX,MSBFIRST,fila[i]);

digitalWrite(SRLK,HIGH);

digitalWrite(SRLK,LOW);

}}else{

for(int i = 0; i < 8; i++) {

shiftOut(SER,RCLX,LSBFIRST,255);

shiftOut(SER,RCLX,MSBFIRST,fila[i]);

digitalWrite(SRLK,HIGH);

digitalWrite(SRLK,LOW);

}}

}

}

int \*patron1(){

int Contadorespacio=3,bin,contadorleds=2,l=1;

int \*apuntador= new int[8];

for (int col=0;col<4;col++){

int contadorparcialleds=0,contparcialespacios=0;

bin=0b11111111;

for (int fil=0;fil<8;fil++){

if(contparcialespacios<Contadorespacio){

contparcialespacios=contparcialespacios+1;

}else if(contadorparcialleds<contadorleds){

bitWrite(bin, fil, 0);

contadorparcialleds=contadorparcialleds+1;

if(contadorparcialleds==contadorleds)

contparcialespacios=0;

}

}

Contadorespacio=Contadorespacio-1;

contadorleds=contadorleds+2;

apuntador[col]=bin;

}

for(int col=4;col<8;col++){

apuntador[col]=apuntador[col-l];

l=l+2;

}

return apuntador;

}

int \*patron3(){

int v=0,bin;

int \*apuntador= new int[8];

for (int col=0;col<8;col++){

bin = 0b11111111;

for (int fil=7;fil>=0;fil--){

if(col%2==0){

if(fil==7){

if(v==1)

v=0;

}

if(v!=2){

bitWrite(bin, fil, 0);

v=v+1;

}else{

v=0;

}

}else{

if(fil==7){

if(v==2)

v=0;

else

v=2;

}

if(v!=2){

bitWrite(bin, fil, 0);

v=v+1;

}else{

v=0;

}

}

}

apuntador[col]=bin;

}

return apuntador;

}

int \*patron4(){

int l=1,bin;

int \*apuntador= new int[8];

char arre[8]={1,1,1,1,0,0,0,0};

for(int fil = 0;fil < 4; fil++){

bin=0b00000000;

for(int col = 0;col < 8; col++){

bitWrite(bin, col, arre[col]);

}for(int k=0;k<8;k++){

arre[k]=arre[k+1];

}apuntador[fil]=bin;

arre[7]=1;

}

for(int fil=4;fil<8;fil++){

for (int col=0;col<8;col++){

apuntador[fil]=apuntador[fil-l];

}l=l+2;

}

return apuntador;

}

int \*patron2(){

int bin;

int \*apuntador= new int[8];

for(int fil = 0;fil < 8; fil++){

bin=0b11111111;

for(int col = 0;col < 8; col++){

if(fil==col||fil+col==7){

bitWrite(bin, col, 0);

}

}

apuntador[fil]=bin;

}

return apuntador;

}

void imprimirConTiempo(unsigned long tiempo,int \*columna){

/\*

Esta función recibe un valor de tiempo y un apuntador que señala un array de enteros que representan los diferentes valores que deben de tomar los leds.

La función se encarga de imprimir el patrón dado por la columna por medio de una cascada esto hasta que el tiempo de ejecución del programa supera al tiempo dado por la función en cuyo caso se cierra la función.

\*/

unsigned long long checkpoint=millis();

int fila[] = {128,64,32,16,8, 4, 2,1};

bool state=true;

while(true){

if((millis()-checkpoint)>tiempo){

break;}

if(state){

for(int i = 0; i < 8; i++) {

shiftOut(SER,RCLX,LSBFIRST,\*(columna+i));

shiftOut(SER,RCLX,MSBFIRST,fila[i]);

digitalWrite(SRLK,HIGH);

digitalWrite(SRLK,LOW);

}

}}

}

void secuencia(){

/\*

Esta función de tipo void invoca las funciones tiempo() y Cantsecuencia() , además de asignarle a 4 diferentes variables lo que retornaran las funciones patron1,patron2, patron3 y patron4. Luego genera un ciclo for mientras que el i que controla el for sea menor que la cantidad de secuencias digitadas por el usuario y dentro del for invoca a la función imprimirConTiempo() dando el valor de tiempo que dio el usuario y los retornos de las diversas funciones de patrones.

\*/

unsigned int cantsecuencia;

unsigned long t=time();

cantsecuencia=Cantsecuencia();

int \*patro1=patron1(),\*patro2=patron2(),\*patro3=patron3(),\*patro4=patron4();

for (int i=0;i<cantsecuencia;i++){

imprimirConTiempo(t,patro1);

imprimirConTiempo(t,patro2);

imprimirConTiempo(t,patro3);

imprimirConTiempo(t,patro4);

}

}

void publik(){

/\*

Esta función funciona como un menú en el que el usuario decide cual de las tareas decide realizar esta vez

\*/

Serial.println(mensaje0);

while(Serial.available() == 0){}

int opcion = Serial.parseInt();

switch(opcion) {

case 1:{

verificacion();

break;}

case 2:{

imagen();

break;}

case 3:{

secuencia();

break;}

default:{

Serial.println("Valor incorrecto, intente otra vez");}

}}

**Problemas de desarrollo:**

La mayoría de los problemas durante el trascurso del desarrollo del aplicativo se han debido que al tener que comunicar el software con el hardware se han presentado errores por lo que tanto el software como el hardware han tenido que ser adaptados para poder llegar a una solución. En el uso de esto una función básica de Arduino que me ha ayudado mucho ha sido la de bitWrite(); que nos permite cambiar el estado de un bit especifico en una cadena de estos, una habilidad muy útil a la hora de generar los valores que se led debe de dar a las columnas para generar los diferentes patrones.

Pero más allá de problemas de código lo que más se complicó en el desarrollo del funcionamiento fue el entender la naturaleza de diversas herramientas como el monitor serial ya que un error bastante problemático fue el que invoque una función que usaba el monitor serial por fuera del loop por lo que el monitor serial no funciono, después de experimentar e investigar descubrí que al invocar esa función por dentro del loop todo funcionaba bien, algo parecido suceddio con el integrado 74HC595 al que se tuvo que investigar profundamente para poder usarlo de la mejor manera posible.

**Evolución de la solución:**

La solución que se ha aplicado ha variado demasiado en el trascurso del desarrollo, lo primero que se realizo fue el desarrollo de las formulas para los patrones en el lenguaje de c++ por medio de matrices, mas esta solución no era la más adecuada a la hora del uso del Arduino por lo que se decidió por cambiarla a un arreglo de 8 enteros en la que al convertirse en un entero de 8 dígitos estos señalan que leds se prenden y cuales no en cada fila, mientras se tiene otro arreglo de leds con los valores de 20, 21, 22, 23,…, 27, 28. Para poder generar un efecto de cascada que imprime cada fila a una velocidad tan rápida que es imperceptible al ojo humano.