Universidad de San Carlos de Guatemala
Arquitectura de computadores y ensambladores 1
Escuela de Sistemas
Ing. Otto Rene Escobar Leiva
Aux Frederick Jonathan Faugier Pinto

PRACTICA 1:

Nombre: Carnet:
John Henry López Mijangos 201710392

Guatemala 06 de JUNIO del 2022

INTRODUCCIÓN

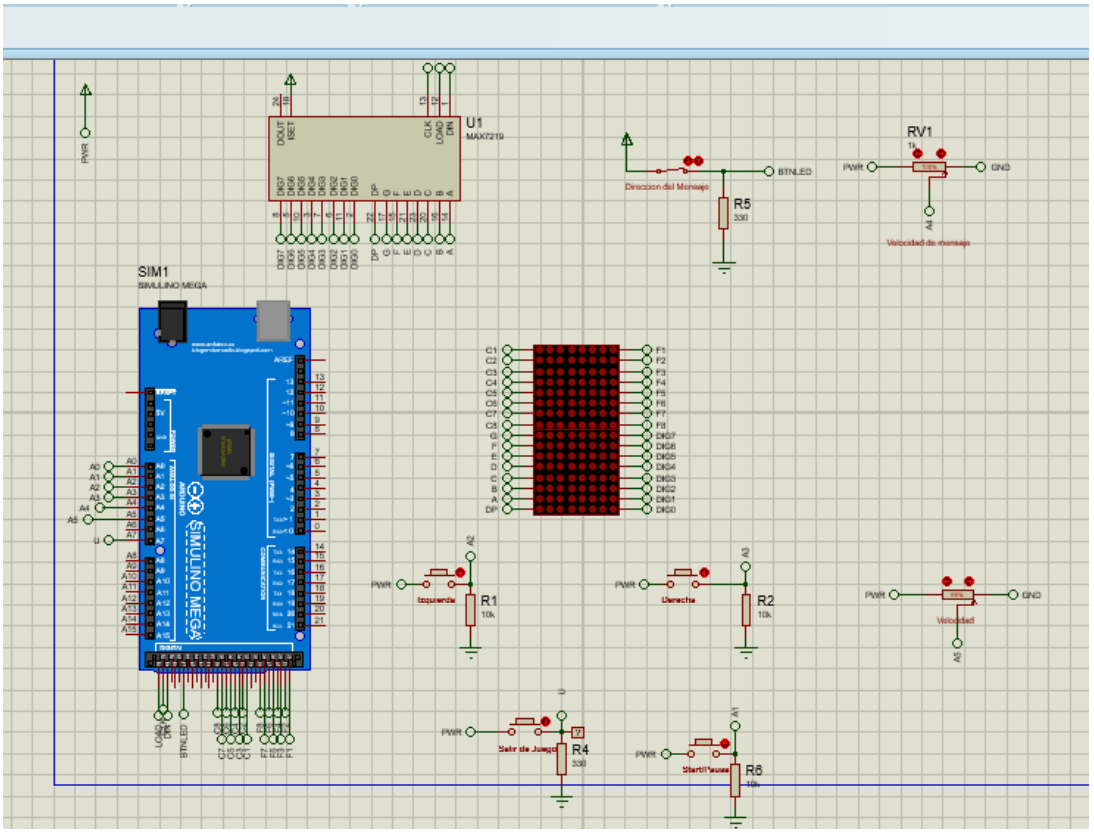
El microcontrolador arduino es una herramienta de suma importancia en el ámbito computacional pues es un referente para la manipulación de entradas y salidas de bits a través de este, ya sea en la graficación en matrices led así como la manipulación de datos entrantes al chip, esto conjugado con el integrado MAX7219 nos permitio personalizar de gran forma el comportamiento de un conjunto de leds presentes en dos matrices de 8x8, facilitando el encendido de led's en específico.

Este proyecto fue trabajado, como anteriormente se dijo, a través de dos matrices led 8x8, teniendo un juego que simula ser un carro que evita a sus enemigos, contabilizando los enemigos chocados y deteniendo el juego en caso de colisión del carro enemigo con el carro original utilizada, además fue agregado un mensaje de bienvenida, puntaje y una opción de pausa.

Finalizando el proyecto se pudo conocer y entender la enorme amplitud de métodos y posibilidades de creación que un microcontrolador como Arduino y un driver pueden dar.

CONTENIDO

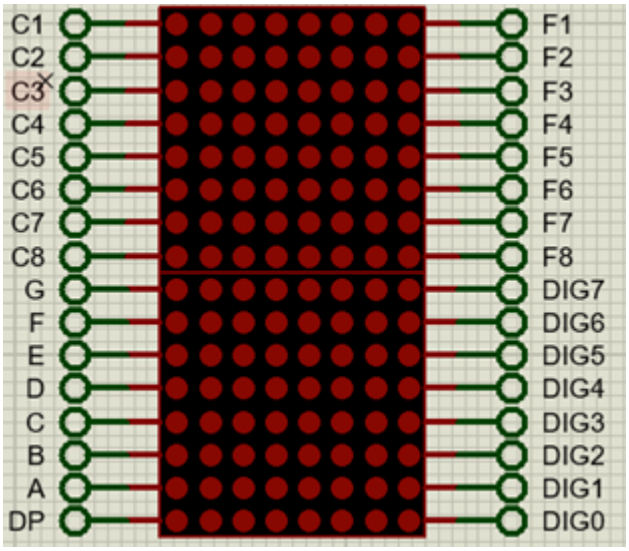
Diagramas de circuitos:



Explicación del Código Arduino

Mensaje

Es el mensaje que se muestra al arrancar el circuito y muestra un mensaje de arriba hacia abajo (en principio), el cual muestra la siguiente cadena: *PRACTICA1-201710392* en el siguiente arreglo de matrices:



En donde la matriz superior no hace uso de driver mientras que la inferior sí.
Los caracteres se irán mostrando en principio de arriba hacia abajo, pero por medio de un switch se puede hacer que cambien de dirección y se muestren de abajo hacia arriba; así como también se puede cambiar la velocidad de transición en las matrices por medio de un potenciómetro

Para poder realizar lo anterior en Arduino, se hizo lo siguiente;

Variables:

Se hizo uso de las siguientes variables globales:

```
int fil[] = {22, 23, 24, 25, 26, 27, 28, 29}; // Filas de matriz sin controlador
int col[] = {32, 33, 34, 35, 36, 37, 38, 39}; // Columnas de matriz sin controlador
int pot = A4; // Lee el valor del potenciometro para manejar la velocidad de transicion en las matrices
int switchPresionado = 47; // Lee el switch para controlar la direccion de transicion de los caracteres
int estado=0; // Para automata
int boton; // guarda el estado del switch (presionado o no)
int del;   // guarda el valor del potenciomatro para hacer que el texto se traslade rapido/lento (delay)
```

Setup:

Esto es lo primero que se ejecuta. Dentro de esta función se inicializa el controlador, se establece la intensidad que tendrá cada led de la matriz que controle este y se "limpia" la matriz que usará el controlador.

En el primer bucle se establecen cuáles serán los pines de Arduino que será de salida y que controlarán las filas y columnas de la matriz sin controlador por medio de la función pinMode().

Posteriormente, en el segundo bucle se apagan todos los leds de la matriz sin controlador por medio de la función digitalWrite().

Y por último, se establece que el pin de Arduino 47 funcionará como entrada para cuando se presione el switch, esto por medio de la función pinMode().

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

  //inicializamos driver
  lc.shutdown(0, false); //inicia apagado - dispositivo 1
  lc.setIntensity(0, 15); // Estableciendo la intesidad de los LED's en la matriz con controlador
  lc.clearDisplay(0); // Limpiando la matriz con controlador

  // Indicando pines de salida para la matriz sin controlador
  for(int i=0; i<8; i++){
    pinMode(fil[i], OUTPUT);
    pinMode(col[i], OUTPUT);
  }
  // Garantizando que se la matriz empiece apagada, para la matriz con controlador
  for(int i=0; i<8; i++){
    digitalWrite(fil[i], HIGH);
    digitalWrite(col[i], LOW);
  }

  pinMode(switchPresionado,INPUT);

  reiniciarJuego();
}
```

loop():

Dentro del loop se tienen las siguientes instrucciones:

```
boton = digitalRead(switchPresionado);
del = analogRead(pot);
if(boton==1){
  mostrarTexto_AbajoArriba(); // Muestra el texto de abajo hacia arriba
} else {
  mostrarTexto_ArribaAbajo(); // Muestra el texto de arriba hacia abajo
}
```

Donde *boton* almacena el estado del switch (si este esta con un valor lógico 1 o 0) por medio de la función digitalRead().
del es una variable de tipo int, la cual lee el valor actual del potenciómetro por medio de la función analogRead().
Posterior, se entra una estructura condicional, en la cual si el interruptor está con un valor lógico 0 (que es como se inicializó) el texto se mostrará de arriba hacia abajo, de lo contrario si el interruptor tiene un valor lógico 0, el texto se mostrará de abajo hacia arriba.

Funciones

mostrarTexto_ArribaAbajo()

Hace que la transición de los caracteres sea de arriba hacia abajo. Para muestra, una porción del código de este método.

```
464
465   if (estado == 0) {
466     //mostrarCaracter(ASTERISCO);
467
468     mostrarCaracterDriver(P1);
469     mostrarCaracter(ESPACIO);
470     delay(del);
471     LimpiarMatriz();
472     estado = 1; // reseteando los estad
473   } else if (estado == 1) {
474
475     mostrarCaracterDriver(P2);
476     mostrarCaracter(ESPACIO);
477     delay(del);
478     LimpiarMatriz();
479     estado = 2;
480   } else if (estado == 2) {
```

Como se puede ver, se maneja una variable de tipo int llamada *estado*, y esto es porque esta función es una máquina de estados finita, en la que, dependiendo del estado en el que se encuentre se comportará de una determinada manera.

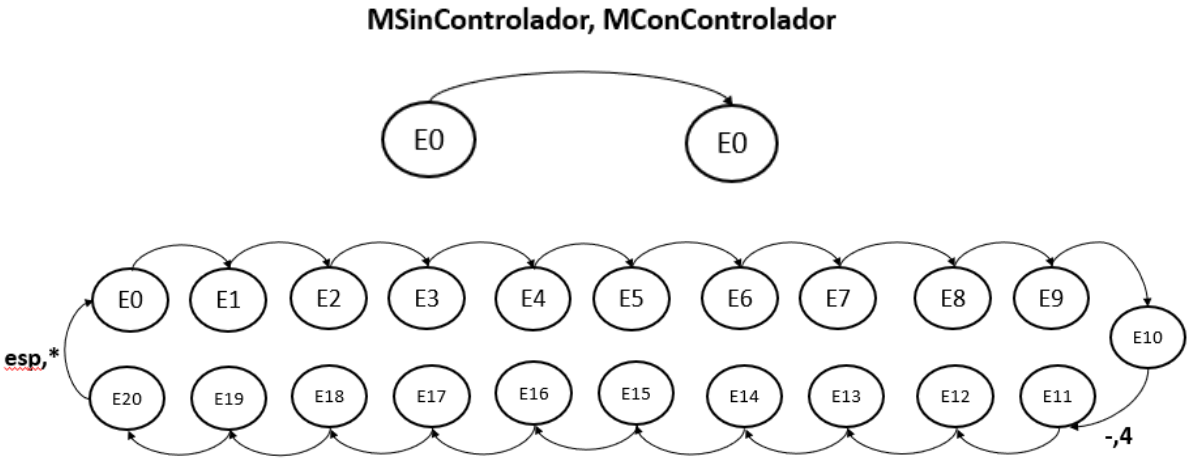
Lo anterior indica el funcionamiento que tendrán las matrices, en la matriz sin controlador (en este caso la superior) es en la que se mostrarán siempre el primer carácter mediante la función *mostrarCaracter*, la cual recibe por parámetros la letra que se quiere mostrar en la matriz sin controlador:

```
// Muestra un caracter el la matriz sin driver (la de arriba)
void mostrarCaracter(int letra[8][8]){
  int col_aux=7;
  for(int repe=0; repe<30; repe++){
    for(int i=0; i<8; i++){
      digitalWrite(col[i],HIGH); // Activando la fila
      for(int j=0; j<8; j++){
        if(letra[i][j]==1){
          digitalWrite(fil[col_aux], LOW); // Activando la columna
        }
        col_aux--;
      }
      col_aux=7;
      delay(2);
      digitalWrite(col[i],LOW); // Desactivando la fila
      for(int j=0; j<8; j++){
        digitalWrite(fil[j],HIGH); // Desactivando la columna
      }
    }
  }
}
```

La matriz con controlador (en este caso la inferior) es en la cual se muestra el carácter que fué mostrado en el estado inmediato anterior. La matriz se muestra usando la función *mostrarCaracterDriver*:

```
// Muestra un caracter el la matriz con driver (la de abajo)
void mostrarCaracterDriver(int letra[8][8]){
  int fil_aux=0;
  int col_aux=7;
  for(int i=0; i<8; i++){
    for(int j=0; j<8; j++){
      lc.setLed(0, fil_aux, col_aux, letra[i][j]);
      fil_aux++;
    }
    fil_aux=0;
    col_aux--;
  }
}
```

Se puede representar gráficamente el autómata usando la siguiente notación, donde: ***MSinControlador*** es la matriz sin controlador (la matriz superior) y ***MConControlador*** es la matriz con controlador (la matriz inferior).



mostrarTexto_AbajoArriba()

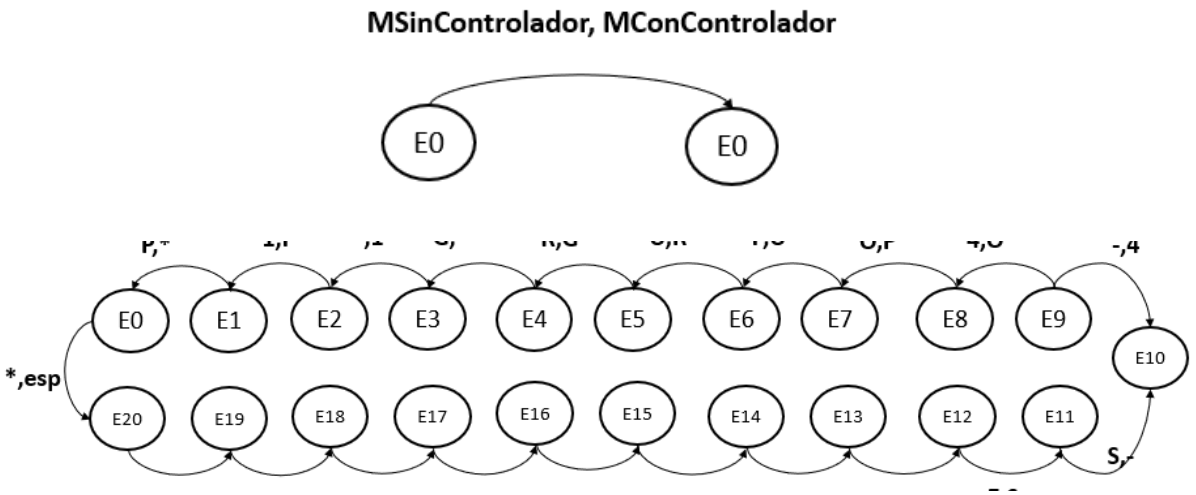
Hace que la transición de los caracteres sea de abajo hacia arriba. Para muestra, una porción del código de este método:

```
// Muestra el texto de abajo hacia arriba
void mostrarTexto_AbajoArriba(){

  if(estado==0){
    //mostrarCaracter(ASTERISCO);
    mostrarCaracterDriver(ESPACIO);
    mostrarCaracter(ASTERISCO);
    delay(del);
    LimpiarMatriz();
    estado=20; // "reseteando" los estados
  } else if(estado==1){
    mostrarCaracterDriver(ASTERISCO);
    mostrarCaracter(P);
    delay(del);
    LimpiarMatriz();
    estado=0;
  } else if(estado==2){
    mostrarCaracterDriver(P);
    mostrarCaracter(UNO);
    delay(del);
    LimpiarMatriz();
    estado=1;
  } else if(estado==3){
    mostrarCaracterDriver(UNO);
    mostrarCaracter(GUION);
    delay(del);
    LimpiarMatriz();
    estado=2;
```

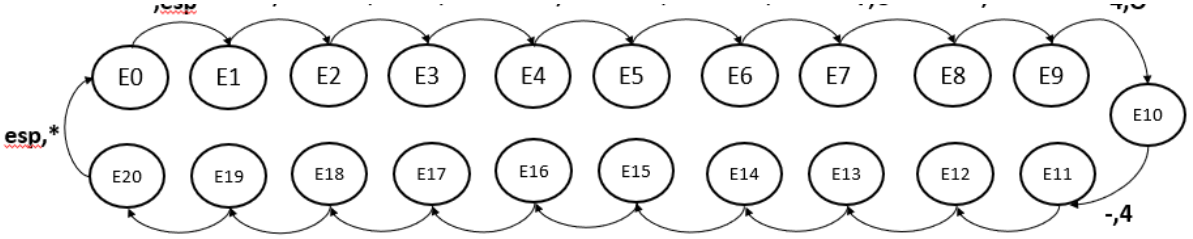
Las funciones *mostrarCaracter* y *mostrarCaracterDriver* también se utilizan para este autómata.

Se puede representar gráficamente el autómata usando la siguiente notación:

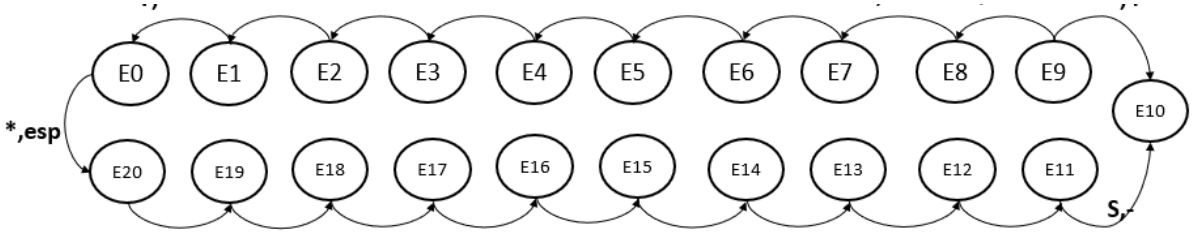


Comparación de los dos autómatas:

Transición de arriba hacia abajo



Transición de abajo hacia arriba



LimpiarMatriz(): Es función que tienen en común las dos funciones anteriores explicadas (*mostrarTexto_ArribaAbajo* y *mostrarTexto_AbajoArriba*) y que su función es apagar todos los leds de la matriz sin controlador (matriz superior).

Contador de puntos:

En este método se contemplan 2 variables, una para unidades y una para decenas, las dos del tipo int. sirven y se iteran cada vez que se le da una nave enemiga, aumentando las unidades cada vez que sucede esto, llegando a validarse cada vez que el contador de unidades es 9 y aumenta a uno las decenas y reinicia las unidades a cero.

```
,
if(pausa==true){
    unidad++;
    pausa=false;
     if(unidad==9){
       decena++;
       unidad=0;
       }
}
```

Impresión de driver y sin driver de contador

```
void imprimir_conteo(){
     for(int fila = 0; fila < 8; fila++){
      digitalWrite(filas[fila], HIGH);
      for(int columna = 0; columna < 8; columna++){
   if(decena==1){ molde[columna][fila] = m1[columna][fila];  }

     if(decena==2){  molde[columna][fila] = m2[columna][fila];  }

     if(decena==3){  molde[columna][fila] = m3[columna][fila];
     if(decena==4){ molde[columna][fila] = m4[columna][fila];  }

     if(decena==5){   molde[columna][fila] = m5[columna][fila];

     if(decena==6){ molde[columna][fila] = m6[columna][fila]; }

     if(decena==7){ molde[columna][fila] = m7[columna][fila];  }

     if(decena==8){ molde[columna][fila] = m8[columna][fila];  }

     if(decena==9){  molde[columna][fila] = m9[columna][fila];  }
```

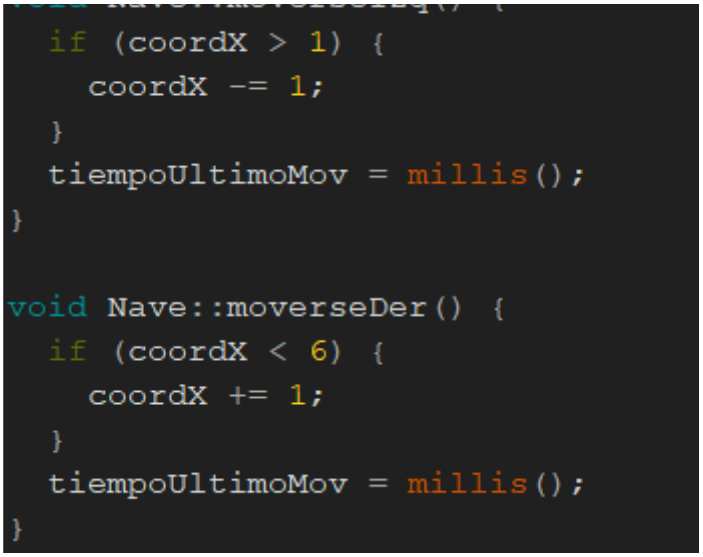
```
void imprimir_sindriver_conteo(){
  for(int columna = 0; columna < 8; columna++){
      //digitalWrite(filas[columna],HIGH);
      for(int fila=0; fila < 8; fila++){
      if(molde[columna][fila] == 1){digitalWrite(columnas[fila],LOW);}}
      delay(4);
      digitalWrite(filas[columna],LOW);
      for(int j = 0; j < 8;j++){ digitalWrite(columnas[j],HIGH); }}
}


void imprimir_conteo(){
     for(int fila = 0; fila < 8; fila++){
      digitalWrite(filas[fila], HIGH);
```

Juego:

El juego empezará dibujando la nave en la matriz de abajo, y los enemigos se dibujaran en la matriz de arriba, en un intervalo de tiempo que calibramos con el potenciómetro.

Movimiento de la carro principal: A través de condicionales como lo son el que esté presionado el botón de movimiento a la izquierda o derecha y los métodos puedeMovIzq() o puedeMovDer() (estas condicionales devuelven un booleano) y con los métodos moverseIzq() y moverseDer() se le restará o sumará un uno al valor actual de la coordenada x de la nave, esto solo si se cumplen las condiciones anteriormente dichas.



```
  if (coordX > 1) {
    coordX -= 1;
  }
  tiempoUltimoMov = millis();
}

void Nave::moverseDer() {
  if (coordX < 6) {
    coordX += 1;
  }
  tiempoUltimoMov = millis();
}
```

Creación de carros enemigo: Creamos un enemigo mediante el método crear Enemigo(), creamos un valor entero random para la coordenada x, verificamos que en esa coordenada no haya un enemigo con el método enemigosOcupanEspacio(), mandando el id del enemigo. Si no hay enemigos ocupando el espacio se cambial el valor false a true para indicar que ahora hay un enemigo.

```
void crearEnemigo() {
  Enemigo enemigo;
  enemigo.coordX = random(6);
  for (int idEnemigo = 0; idEnemigo < MAX_NUMERO_DE_ENEMIGOS; idEnemigo++) {
    if(!enemigosOcupanEspacio[idEnemigo]){
      if(idEnemigo != 0){
        if(enemigosVivos[idEnemigo-1].coordY < 12){
          enemigosOcupanEspacio[idEnemigo] = true;
          enemigosVivos[idEnemigo] = enemigo;
        }
      }else {
        enemigosOcupanEspacio[idEnemigo] = true;
        enemigosVivos[idEnemigo] = enemigo;
      }
      break;
    }
  }
}
```

Movimiento de carro enemigo: Los enemigos se mueven dependiendo del valor que se lea en el potenciometro, primero verificamos si se puede mover con el método puedeMoverse(), este método devuelve un valor True o false, si es true se le suma 2 a la coordenada en Y para observar el movimiento descendente.

Destrucción de carro principal: Para esta parte se utilizó el método verificarEnemigoTocaNave(), para saber si el enemigo está chocando con la nave principal, primero recorremos la matriz para saber los id's de las naves luego utilizamos el método estaTocandoCoordenadas() le mandamos la posición x y y, si devuelve un true se acaba el juego.

```
void verificarEnemigoTocaNave() {
  for (int idEnemigo = 0; idEnemigo < MAX_NUMERO_DE_ENEMIGOS; idEnemigo++) {
    Enemigo enemigo = enemigosVivos[idEnemigo];
    bool tocaDer = nave.estaTocandoCoordenadas(enemigo.coordX-1, enemigo.coordY+1);
    bool tocaIzq = nave.estaTocandoCoordenadas(enemigo.coordX+1, enemigo.coordY+1);
    bool tocaFrente = nave.estaTocandoCoordenadas(enemigo.coordX, enemigo.coordY);
    if (enemigosOcupanEspacio[idEnemigo] && (tocaDer || tocaIzq || tocaFrente)) {
      reiniciarJuego();
    }
  }
}
```

EQUIPO UTILIZADO

- Computadora dual core.
- Windows 10
- Proteus 8.10
- Arduino IDE
- Librería Simulino
- Librería LedControl
- Memoria RAM de 4 gb.
- Driver para matriz led 8x8 (MAX7219)
- Potenciómetro
- 2 matrices Led 8x8.
- Resistencias.
- Botones
- Switch

CONCLUSIONES

- El microcontrolador Arduino permite dependiendo su versión una manipulación de diversas entradas y salidas a través de su programación con el IDE correspondiente.
- El Driver MAX7219 permite a través de la librería LedControl una manipulación mas sencilla del encendido de luces, permitiendo enviar filas enteras para graficar una por una.
- Los potenciómetros pueden ser una herramienta muy eficaz para ser un regulador de la velocidad al tomarse como parámetro para los delays.
- Los delays permiten visualizar de mejor forma los cambios, sin embargo alentan al sistema, siendo mejor usar los datos del tiempo a través del metodo millis() en su lugar.

APENDICE

```
#include <LedControl.h>
#include "letras.h"
#include "numeros.h"

//// MOdificado

LedControl lc = LedControl(51, 52, 53, 1);
const int button3Pin = 3;
long unsigned int contabilidad=0;
int contando=0;
unsigned long tiempo;
unsigned long tiempo2;
unsigned long tiempo3;
 int button3State = 0;
int fil[] = {22, 23, 24, 25, 26, 27, 28, 29}; // Filas
int col[] = {32, 33, 34, 35, 36, 37, 38, 39}; // Columnas
int pot = A4; // Lee el potenciometro
int pot2 = A5; // Lee el potenciometro para el juego
int switchPresionado = 47; // Lee el switch
int estado = 0;
int boton; // guarda el estado del switch (presionado o no)
int del;   // guarda el valor del potenciomatro para hacer que el texto se traslade rapido/lento
(delay)
// Fin variables para texto ------------------------------------------------------------
bool juegopausa = false;
bool pausa = false;
int unidad = 1;
int decena = 1;
int columnas[] = {22, 23, 24, 25, 26, 27, 28, 29};
int filas[] = {32, 33, 34, 35, 36, 37, 38, 39};

int molde[][8] = {
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0}
};

int btnDisparo = A0;
int btnIzq = A2;
```

```
int btnDer = A3;
int btnStart = A1;
bool btnIzqPres;
bool btnDerPres;
bool btnDisparoPres;
bool GAME_OVER = true;
byte cuadriculaLimpia[16] = {};
byte cuadriculaActual[16] = {};
const int MAX_NUMERO_DE_PROYECTILES = 25;
const int TODOS_LOS_COHETES_HAN_SIDO_USADOS = -1;
const int MAX_NUMERO_DE_ENEMIGOS = 55;
const int TODOS_LOS_ENEMIGOS_HAN_SIDO_USADOS = -1;


// NAVE ----------------------------------------------------------------------------
class Nave {
  public:
    int coordX, velMov, velDisparo, score[2]; // [0]: Decenas, [1]; Unidades

    bool puedeMovIzq();
    bool puedeMovDer();
    //bool puedeDisparar();

    bool estaTocandoCoordenadas(int, int); // x, y

    void moverseIzq();
    void moverseDer();
    //void disparar();

  private:
    long unsigned int tiempoUltimoMov;
    //long unsigned int tiempoUltimoDisparo;
};

bool Nave::puedeMovIzq() {
  return (coordX > 1 && (millis() - tiempoUltimoMov) > velMov);
}

bool Nave::puedeMovDer() {
  return (coordX < 6 && (millis() - tiempoUltimoMov) > velMov);
}



bool Nave::estaTocandoCoordenadas(int x, int y) {
  bool estaTocandoLadoDer = ((x == coordX + 1) && (y == 3));
  bool estaTocandoLadoIzq = ((x == coordX - 1) && (y == 3));
  bool estaTocandoLadoFrontal = ((x == coordX) && (y == 3));
```

```
  return (estaTocandoLadoDer || estaTocandoLadoIzq  || estaTocandoLadoFrontal);
}

void Nave::moverseIzq() {
  if (coordX > 1) {
    coordX -= 1;
  }
  tiempoUltimoMov = millis();
}

void Nave::moverseDer() {
  if (coordX < 6) {
    coordX += 1;
  }
  tiempoUltimoMov = millis();
}



Nave nave;


// ENEMIGO ------------------------------------------------------------------------------
class Enemigo {
  public:
    int coordX, coordY = 15, velMov = 2000; //Debe ser en una funcion

    bool puedeMoverse();
    void moverse();
    //void actvel();

  private:
    long unsigned int tiempoUltimoMov = 0;
};

bool Enemigo::puedeMoverse() {
  return (millis() - tiempoUltimoMov) > velMov;
}

Enemigo enemigosVivos[MAX_NUMERO_DE_ENEMIGOS] = {};
bool enemigosOcupanEspacio[MAX_NUMERO_DE_ENEMIGOS] = {};

void Enemigo::moverse() {
  coordY--;
  tiempoUltimoMov = millis();
}

void crearEnemigo() {
```

```
  Enemigo enemigo;
  enemigo.coordX = random(6);
  for (int idEnemigo = 0; idEnemigo < MAX_NUMERO_DE_ENEMIGOS; idEnemigo++) {
    if (!enemigosOcupanEspacio[idEnemigo]) {
      if (idEnemigo != 0) {
        if (enemigosVivos[idEnemigo - 1].coordY < 12) {
//          if (millis()-contabilidad>=5000) {
//            enemigosOcupanEspacio[idEnemigo] = true;
//            contabilidad=0;
//            contabilidad = millis();
//            enemigosVivos[idEnemigo] = enemigo;
//          }
          enemigosOcupanEspacio[idEnemigo] = false;
          enemigosVivos[idEnemigo] = enemigo;
        }
      } else {
        enemigosOcupanEspacio[idEnemigo] = true;
        enemigosVivos[idEnemigo] = enemigo;
      }
      break;
    }
  }
}

void actualizarVelEnemigo() {
  for (int idEnemigo = 0; idEnemigo < MAX_NUMERO_DE_ENEMIGOS; idEnemigo++) {
    if (enemigosOcupanEspacio[idEnemigo]) {
      enemigosVivos[idEnemigo].velMov = analogRead(pot2) * 4;
    }
  }
}

// JUEGO ---------------------------------------------------------------------------------
void dibujarCuadricula(byte * cuadricula) {
  int contadorM2 = 7;
  for (int i = 0; i < 16; i++) {
    if (i < 8) {
      digitalWrite(i + 32, HIGH);
      for (int j = 7; j >= 0; j--) {
        int a = cuadricula[i] >> j;
        bool valor = a & 1;
        if (valor) {
          digitalWrite(j + 22, LOW);
        } else {
          digitalWrite(j + 22, HIGH);
        }
      }
      delay(1);
```

```
      digitalWrite(i + 32, LOW);
      for (int j = 22; j < 30; j++) {
        digitalWrite(j, HIGH);
      }
    } else {
      lc.setColumn(0, contadorM2, cuadricula[i]);
      contadorM2--;
    }
  }
}

void dibujarNave(int coord) {
  cuadriculaActual[12] = cuadriculaActual[12] | (B01000000 >> coord - 1);
  cuadriculaActual[13] = (B11100000 >> coord - 1);
  cuadriculaActual[14] = (B01000000 >> coord - 1);
  cuadriculaActual[15] = (B11100000 >> coord - 1);
  // este es de 3 B11100000
  // este es de 1 B01000000
}



void dibujarEnemigos() {
  for (int idEnemigo = 0; idEnemigo < MAX_NUMERO_DE_ENEMIGOS; idEnemigo++) {
    Enemigo enemigo = enemigosVivos[idEnemigo];
    if (enemigosOcupanEspacio[idEnemigo]) {
      if (enemigo.coordY == 15) {
        cuadriculaActual[0] = cuadriculaActual[0] | (B01000000 >> enemigo.coordX - 1);
        contando=0;
      } else {
        if (enemigo.coordY == 14) {
          cuadriculaActual[0] = cuadriculaActual[0] | (B01000000 >> enemigo.coordX - 1);
          cuadriculaActual[1] = cuadriculaActual[1] | (B11100000 >> enemigo.coordX - 1);
        } else {
          if (enemigo.coordY == -1) {
            cuadriculaActual[14] = cuadriculaActual[14] | (B11100000 >> enemigo.coordX - 1);
            cuadriculaActual[15] = cuadriculaActual[15] | (B01000000 >> enemigo.coordX - 1);
          } else {
            if (enemigo.coordY == -2) {
              cuadriculaActual[15] = cuadriculaActual[15] | (B11100000 >> enemigo.coordX - 1); //
es tope del tablero---
              contando++;
              if(contando==1){
                unidad++;
                if(unidad==10){
                decena++;
                unidad=0;
              }
```

```
          }


        } else {
          cuadriculaActual[13 - enemigo.coordY] = cuadriculaActual[13 - enemigo.coordY] |
(B11100000 >> enemigo.coordX - 1);
          cuadriculaActual[14 - enemigo.coordY] = cuadriculaActual[14 - enemigo.coordY] |
(B01000000 >> enemigo.coordX - 1);
          cuadriculaActual[15 - enemigo.coordY] = cuadriculaActual[15 - enemigo.coordY] |
(B11100000 >> enemigo.coordX - 1);
        }
       }
      }
     }
     if (enemigo.puedeMoverse()) {
      enemigosVivos[idEnemigo].moverse();
      if (enemigosVivos[idEnemigo].coordY + 2 < 0) {
       enemigosOcupanEspacio[idEnemigo] = false;
      }
     }
    }
   }
}

bool estaPresionado(int idBtn) {
  int estadoBtn = digitalRead(idBtn);
  return (estadoBtn == HIGH);
}

void verificarEnemigoTocaNave() {
  for (int idEnemigo = 0; idEnemigo < MAX_NUMERO_DE_ENEMIGOS; idEnemigo++) {
    Enemigo enemigo = enemigosVivos[idEnemigo];
    bool tocaDer = nave.estaTocandoCoordenadas(enemigo.coordX - 1, enemigo.coordY +
1);
    bool tocaIzq = nave.estaTocandoCoordenadas(enemigo.coordX + 1, enemigo.coordY +
1);
    bool tocaFrente = nave.estaTocandoCoordenadas(enemigo.coordX, enemigo.coordY);
    if (enemigosOcupanEspacio[idEnemigo] && (tocaDer || tocaIzq || tocaFrente)) {
      if(decena==0){conteosindriver(CERO); }
  else if(decena==1){conteosindriver(UNO); }
  else if(decena==2){conteosindriver(DOS); }
  delay(100);
  if(unidad==0){conteocondriver(CERO); }
  else if(unidad==1){conteocondriver(UNO); }
  else if(unidad==2){conteocondriver(DOS); }
  else if(unidad==3){conteocondriver(TRES); }
  else if(unidad==4){conteocondriver(CUATRO); }
  else if(unidad==5){conteocondriver(CINCO); }
```

```
  else if(unidad==6){conteocondriver(SEIS); }
  else if(unidad==7){conteocondriver(SIETE); }
  else if(unidad==8){conteocondriver(OCHO); }
  else if(unidad==9){conteocondriver(NUEVE); }
  delay(6000);
     reiniciarJuego();
    }
  }
}



void reiniciarJuego() {
  GAME_OVER = true;
  dibujarCuadricula(cuadriculaLimpia);
  nave.coordX = 3;
  nave.velMov = 50;
  nave.velDisparo = 200;
  unidad = 0;
  decena = 0;
  // Enemigos
  for (int idEnemigo = 0; idEnemigo < MAX_NUMERO_DE_ENEMIGOS; idEnemigo++) {
    enemigosOcupanEspacio[idEnemigo] = false;
  }
}

// SETUP ----------------------------------------------------------------------------
void setup() {
  Serial.begin(9600);

  //inicializamos driver
  lc.shutdown(0, false); //inicia apagado - dispositivo 1
  lc.setIntensity(0, 15);
  lc.clearDisplay(0);

  // Indicando pines de salida
  for (int i = 0; i < 8; i++) {
    pinMode(fil[i], OUTPUT);
    pinMode(col[i], OUTPUT);
  }
  // Garantizando que se la matriz empiece apagada
  for (int i = 0; i < 8; i++) {
    digitalWrite(fil[i], HIGH);
    digitalWrite(col[i], LOW);
  }
  pinMode(button3Pin, INPUT);
  pinMode(switchPresionado, INPUT);
```

```
  reiniciarJuego();
}

void loop() {
    //LimpiarMatriz();
 // imprimir_conteo();
   //   imprimir_sindriver_conteo();
  //Serial.println(analogRead(btnStart));
//
  if (!GAME_OVER) {
    // PAUSA

    if (juegopausa==true) {
  if(decena==0){conteosindriver(CERO); }
  else if(decena==1){conteosindriver(UNO); }
  else if(decena==2){conteosindriver(DOS); }

  if(unidad==0){conteocondriver(CERO); }
  else if(unidad==1){conteocondriver(UNO); }
  else if(unidad==2){conteocondriver(DOS); }
  else if(unidad==3){conteocondriver(TRES); }
  else if(unidad==4){conteocondriver(CUATRO); }
  else if(unidad==5){conteocondriver(CINCO); }
  else if(unidad==6){conteocondriver(SEIS); }
  else if(unidad==7){conteocondriver(SIETE); }
  else if(unidad==8){conteocondriver(OCHO); }
  else if(unidad==9){conteocondriver(NUEVE); }
    if (estaPresionado(btnStart)) {
      juegopausa=false;
      delay(800);

    }


    } else {
      if (nave.puedeMovIzq()) {
        if (!estaPresionado(btnIzq) ) {
          btnIzqPres = true;
        }
        if (estaPresionado(btnIzq) && btnIzqPres) {
          nave.moverseIzq();
          btnIzqPres = false;
        }
      }
      if (nave.puedeMovDer()) {
        if (!estaPresionado(btnDer) ) {
          btnDerPres = true;
        }
```

```
      if (estaPresionado(btnDer) && btnDerPres) {
        nave.moverseDer();
        btnDerPres = false;
      }
    }
//    if (nave.puedeDisparar()) {
//      if (!estaPresionado(btnDisparo) ) {
//        btnDisparoPres = true;
//      }
//      if (estaPresionado(btnDisparo) && btnDisparoPres) {
//        nave.disparar();
//        btnDisparoPres = false;
//      }
//    }

    crearEnemigo();
    actualizarVelEnemigo();

    memcpy(cuadriculaActual, cuadriculaLimpia, 16);

    dibujarNave(nave.coordX);
    dibujarEnemigos();
    dibujarCuadricula(cuadriculaActual);
    verificarEnemigoTocaNave();
    if(decena==1 && unidad==2){

      GAME_OVER=true;
      reiniciarJuego();
      win();
    }

    if (estaPresionado(btnStart)) {
juegopausa=true;



     }
   }
  } else {



   if (estaPresionado(btnStart)) {
  tiempo=millis();

while(analogRead(btnStart)==1) {
   tiempo2=millis();
  }
```

```
tiempo3=tiempo2-tiempo;

  if (tiempo3 >= 4000){ //comprueba el estado del boton Codigo para diferenciar pulsacion
corta ->>   pulsacion larga ->>
 GAME_OVER = false;
 delay(500);
}
```

```
    }else {
        if (tiempo3 >= 4000){ //comprueba el estado del boton Codigo para diferenciar
pulsacion corta ->>   pulsacion larga ->>
 GAME_OVER = false;
}
      boton = digitalRead(switchPresionado);
      del = analogRead(pot);
      if (boton == 1) {
        AS();
      } else {
        DES();
      }
    }
    delay(250);

  }
}
```

```
void AS() {

  if (estado == 0) {
    //mostrarCaracter(ASTERISCO);
```

```
    mostrarCaracterDriver(P1);
    mostrarCaracter(ESPACIO);
    delay(del);
    LimpiarMatriz();
    estado = 1; // reseteando los estados (esto para que cuando el texto termine, se vuelva a
mostrar el mismo texto desde el inicio)
  } else if (estado == 1) {

    mostrarCaracterDriver(P2);
    mostrarCaracter(ESPACIO);
    delay(del);
    LimpiarMatriz();
    estado = 2;
  } else if (estado == 2) {

    mostrarCaracterDriver(PR);
    mostrarCaracter(P1);
    delay(del);
    LimpiarMatriz();
    estado = 3;
  }
  else if (estado == 3) {

    mostrarCaracterDriver(R1);
    mostrarCaracter(P2);
    delay(del);
    LimpiarMatriz();
    estado = 4;
  }

  else if (estado == 4) {

    mostrarCaracterDriver(RA);
    mostrarCaracter(PR);
    delay(del);
    LimpiarMatriz();
    estado = 5;
  }
else if (estado == 5) {

    mostrarCaracterDriver(AA1);
    mostrarCaracter(R1);
    delay(del);
    LimpiarMatriz();
    estado = 6;
  }
  else if (estado == 6) {
```

```
    mostrarCaracterDriver(AC);
    mostrarCaracter(RA);
    delay(del);
    LimpiarMatriz();
    estado = 7;
  }

  else if (estado == 7) {

    mostrarCaracterDriver(C1);
    mostrarCaracter(AA1);
    delay(del);
    LimpiarMatriz();
    estado = 8;
  }
  else if (estado == 8) {

    mostrarCaracterDriver(CT);
    mostrarCaracter(AC);
    delay(del);
    LimpiarMatriz();
    estado = 9;
  }

else if (estado == 9) {

    mostrarCaracterDriver(T1);
    mostrarCaracter(C1);
    delay(del);
    LimpiarMatriz();
    estado = 10;
  }


  else if (estado == 10) {

    mostrarCaracterDriver(TI);
    mostrarCaracter(CT);
    delay(del);
    LimpiarMatriz();
    estado = 11;
  }

else if (estado == 11) {

    mostrarCaracterDriver(I1);
    mostrarCaracter(T1);
    delay(del);
```

```
    LimpiarMatriz();
    estado = 12;
  }


else if (estado == 12) {

    mostrarCaracterDriver(IC);
    mostrarCaracter(TI);
    delay(del);
    LimpiarMatriz();
    estado = 13;
  }


  else if (estado == 13) {

    mostrarCaracterDriver(C1);
    mostrarCaracter(I1);
    delay(del);
    LimpiarMatriz();
    estado = 14;
  }

  else if (estado == 14) {

    mostrarCaracterDriver(CA);
    mostrarCaracter(IC);
    delay(del);
    LimpiarMatriz();
    estado = 15;
  }
  else if (estado == 15) {

    mostrarCaracterDriver(AA1);
    mostrarCaracter(C1);
    delay(del);
    LimpiarMatriz();
    estado = 16;
  }

else if (estado == 16) {

    mostrarCaracterDriver(AUNO);
    mostrarCaracter(CA);
    delay(del);
    LimpiarMatriz();
    estado = 17;
```

```
  }

else if (estado == 17) {

    mostrarCaracterDriver(UNO);
    mostrarCaracter(AA1);
    delay(del);
    LimpiarMatriz();
    estado = 18;
  }

else if (estado == 18) {

    mostrarCaracterDriver(UNODIA);
    mostrarCaracter(AUNO);
    delay(del);
    LimpiarMatriz();
    estado = 19;
  }

else if (estado == 19) {
    mostrarCaracterDriver(DIA);
    mostrarCaracter(UNO);
    delay(del);
    LimpiarMatriz();
    estado = 20;
  }

else if (estado == 20) {//7777777777777777777777777777777777777777777777
    mostrarCaracterDriver(DOS);
    mostrarCaracter(DIA);
    delay(del);
    LimpiarMatriz();
    estado = 21;
  }

else if (estado == 21) {
    mostrarCaracterDriver(DOSCERO);
    mostrarCaracter(CABEZADOS);
    delay(del);
    LimpiarMatriz();
    estado = 22;
  }

  else if (estado == 22) {
    mostrarCaracterDriver(CERO);
    mostrarCaracter(DOS);
    delay(del);
```

```
    LimpiarMatriz();
    estado = 23;
  }

  else if (estado == 23) {
    mostrarCaracterDriver(CERO1);
    mostrarCaracter(DOSCERO);
    delay(del);
    LimpiarMatriz();
    estado = 24;
  }

  else if (estado == 24) {
    mostrarCaracterDriver(UNO);
    mostrarCaracter(DOS);
    delay(del);
    LimpiarMatriz();
    estado = 25;
  }

  else if (estado == 25) {
    mostrarCaracterDriver(UNO7);
    mostrarCaracter(CERO1);
    delay(del);
    LimpiarMatriz();
    estado = 26;
  }

  else if (estado == 26) {
    mostrarCaracterDriver(SIETE);
    mostrarCaracter(UNO);
    delay(del);
    LimpiarMatriz();
    estado = 27;
  }

  else if (estado == 27) {
    mostrarCaracterDriver(SIETE1);
    mostrarCaracter(UNO7);
    delay(del);
    LimpiarMatriz();
    estado = 28;
  }
///AQUI
 else if (estado == 28) {
    mostrarCaracterDriver(UNO);
    mostrarCaracter(SIETE);
    delay(del);
```

```
    LimpiarMatriz();
    estado = 29;
  }

else if (estado == 29) {
    mostrarCaracterDriver(UNOCERO);
    mostrarCaracter(SIETE1);
    delay(del);
    LimpiarMatriz();
    estado = 30;
  }

  else if (estado == 30) {
    mostrarCaracterDriver(CERO);
    mostrarCaracter(UNO);
    delay(del);
    LimpiarMatriz();
    estado = 31;
  }

  else if (estado == 31) {
    mostrarCaracterDriver(CERO2);
    mostrarCaracter(UNOCERO);
    delay(del);
    LimpiarMatriz();
    estado = 32;
  }


  else if (estado == 32) {
    mostrarCaracterDriver(TRES);
    mostrarCaracter(CERO);
    delay(del);
    LimpiarMatriz();
    estado = 33;
  }


  else if (estado == 33) {
    mostrarCaracterDriver(TRES9);
    mostrarCaracter(CERO2);
    delay(del);
    LimpiarMatriz();
    estado = 34;
  }

  else if (estado == 34) {
    mostrarCaracterDriver(NUEVE);
```

```
    mostrarCaracter(TRES);
    delay(del);
    LimpiarMatriz();
    estado = 35;
  }


  else if (estado == 35) {
    mostrarCaracterDriver(NUEVE2);
    mostrarCaracter(TRES9);
    delay(del);
    LimpiarMatriz();
    estado = 36;
  }


  else if (estado == 36) {
    mostrarCaracterDriver(DOS);
    mostrarCaracter(NUEVE);
    delay(del);
    LimpiarMatriz();
    estado = 0;
  }



}



void DES() {

  if (estado == 36) {
    mostrarCaracterDriver(ESPACIO);
    mostrarCaracter(DOS);
    delay(del);
    LimpiarMatriz();
    estado = 35;
  }
if (estado == 35) {
    mostrarCaracterDriver(FINALDOS);
    mostrarCaracter(NUEVE2);
    delay(del);
    LimpiarMatriz();
    estado = 34;
  }

  else if (estado == 34) {
```

```
    mostrarCaracterDriver(DOS);
    mostrarCaracter(NUEVE);
    delay(del);
    LimpiarMatriz();
    estado = 33;
  }

else if (estado == 33) {
    mostrarCaracterDriver(NUEVE2);
    mostrarCaracter(TRES9);
    delay(del);
    LimpiarMatriz();
    estado = 32;
  }

else if (estado == 32) {
    mostrarCaracterDriver(NUEVE);
    mostrarCaracter(TRES);
    delay(del);
    LimpiarMatriz();
    estado = 31;
  }////////////////////////////////////
else if (estado == 31) {
    mostrarCaracterDriver(TRES9);
    mostrarCaracter(CERO2);
    delay(del);
    LimpiarMatriz();
    estado = 30;
  }

  else if (estado == 30) {
    mostrarCaracterDriver(TRES);
    mostrarCaracter(CERO);
    delay(del);
    LimpiarMatriz();
    estado = 29;
  }

  else if (estado == 29) {
    mostrarCaracterDriver(CERO2);
    mostrarCaracter(UNOCERO);
    delay(del);
    LimpiarMatriz();
    estado = 28;
  }

  else if (estado == 28) {
    mostrarCaracterDriver(CERO);
```

```
    mostrarCaracter(UNO);
    delay(del);
    LimpiarMatriz();
    estado = 27;
  }

  else if (estado == 27) {
    mostrarCaracterDriver(UNOCERO);
    mostrarCaracter(SIETE1);
    delay(del);
    LimpiarMatriz();
    estado = 26;
  }

  else if (estado == 26) {
    mostrarCaracterDriver(UNO);
    mostrarCaracter(SIETE);
    delay(del);
    LimpiarMatriz();
    estado = 25;
  }

else if (estado == 26) {
    mostrarCaracterDriver(SIETE1);
    mostrarCaracter(UNO7);
    delay(del);
    LimpiarMatriz();
    estado = 25;
  }

else if (estado == 25) {
    mostrarCaracterDriver(SIETE);
    mostrarCaracter(UNO);
    delay(del);
    LimpiarMatriz();
    estado = 24;
  }

  else if (estado == 24) {
    mostrarCaracterDriver(UNO7);
    mostrarCaracter(CERO1);
    delay(del);
    LimpiarMatriz();
    estado = 23;
  }

  else if (estado == 23) {
    mostrarCaracterDriver(UNO);
```

```
  mostrarCaracter(CERO);
  delay(del);
  LimpiarMatriz();
  estado = 22;
}

else if (estado == 22) {
  mostrarCaracterDriver(CERO1);
  mostrarCaracter(DOSCERO);
  delay(del);
  LimpiarMatriz();
  estado = 21;
}

else if (estado == 21) {
  mostrarCaracterDriver(CERO);
  mostrarCaracter(DOS);
  delay(del);
  LimpiarMatriz();
  estado = 20;
}

else if (estado == 20) {
  mostrarCaracterDriver(DOS);
  mostrarCaracter(DIA);
  delay(del);
  LimpiarMatriz();
  estado = 19;
}

else if (estado == 19) {
  mostrarCaracterDriver(CABEZADOS);
  mostrarCaracter(UNODIA);
  delay(del);
  LimpiarMatriz();
  estado = 18;
}

else if (estado == 18) {
  mostrarCaracterDriver(DIA);
  mostrarCaracter(UNO);
  delay(del);
  LimpiarMatriz();
  estado = 17;
}

else if (estado == 17) {
  mostrarCaracterDriver(UNODIA);
```

```
    mostrarCaracter(AUNO);
    delay(del);
    LimpiarMatriz();
    estado = 16;
  }

  else if (estado == 16) {
    mostrarCaracterDriver(UNO);
    mostrarCaracter(AA1);
    delay(del);
    LimpiarMatriz();
    estado = 15;
  }

else if (estado == 15) {
    mostrarCaracterDriver(AUNO);
    mostrarCaracter(CA);
    delay(del);
    LimpiarMatriz();
    estado = 14;
  }

  else if (estado == 14) {
    mostrarCaracterDriver(AA1);
    mostrarCaracter(C1);
    delay(del);
    LimpiarMatriz();
    estado = 13;
  }

  else if (estado == 13) {
    mostrarCaracterDriver(CA);
    mostrarCaracter(IC);
    delay(del);
    LimpiarMatriz();
    estado = 12;
  }

  else if (estado == 12) {
    mostrarCaracterDriver(C1);
    mostrarCaracter(I1);
    delay(del);
    LimpiarMatriz();
    estado = 11;
  }

  else if (estado == 11) {
    mostrarCaracterDriver(IC);
```

```
    mostrarCaracter(TI);
    delay(del);
    LimpiarMatriz();
    estado = 10;
  }

  else if (estado == 10) {
    mostrarCaracterDriver(I1);
    mostrarCaracter(T1);
    delay(del);
    LimpiarMatriz();
    estado =9;
  }

  else if (estado == 9) {
    mostrarCaracterDriver(TI);
    mostrarCaracter(CT);
    delay(del);
    LimpiarMatriz();
    estado = 8;
  }

  else if (estado == 8) {
    mostrarCaracterDriver(T1);
    mostrarCaracter(C1);
    delay(del);
    LimpiarMatriz();
    estado = 7;
  }

  else if (estado == 7) {
    mostrarCaracterDriver(CT);
    mostrarCaracter(AC);
    delay(del);
    LimpiarMatriz();
    estado = 6;
  }

  else if (estado == 6) {
    mostrarCaracterDriver(C1);
    mostrarCaracter(AA1);
    delay(del);
    LimpiarMatriz();
    estado = 5;
  }

else if (estado == 5) {
    mostrarCaracterDriver(AC);
```

```
  mostrarCaracter(RA);
  delay(del);
  LimpiarMatriz();
  estado = 4;
}

else if (estado == 4) {
  mostrarCaracterDriver(AA1);
  mostrarCaracter(R1);
  delay(del);
  LimpiarMatriz();
  estado = 3;
}

else if (estado == 3) {
  mostrarCaracterDriver(RA);
  mostrarCaracter(PR);
  delay(del);
  LimpiarMatriz();
  estado = 2;
}

else if (estado == 2) {
  mostrarCaracterDriver(R1);
  mostrarCaracter(P2);
  delay(del);
  LimpiarMatriz();
  estado = 1;
}

else if (estado == 1) {
  mostrarCaracterDriver(PR);
  mostrarCaracter(P1);
  delay(del);
  LimpiarMatriz();
  estado = 0;
}

else if (estado == 0) {
  mostrarCaracterDriver(P2);
  mostrarCaracter(ESPACIO);
  delay(del);
  LimpiarMatriz();
  estado = 36;
}

}
```

```
void LimpiarMatriz() {
  for (int i = 0; i < 8; i++) {
    digitalWrite(fil[i], HIGH);
    digitalWrite(col[i], LOW);
  }
}
```

```
void mostrarCaracter(int letras[8][8]) {
  //int fil_aux=0;
  int col_aux = 7;
  for (int repe = 0; repe < 30; repe++) {
    for (int i = 0; i < 8; i++) {
      //digitalWrite(fil[i],LOW);
      digitalWrite(col[i], HIGH);
      for (int j = 0; j < 8; j++) {
        if (letras[i][j] == 1) {
          //digitalWrite(fil[j], HIGH);
          digitalWrite(fil[col_aux], LOW);
        }
        col_aux--;
      }
      col_aux = 7;
      delay(2);
      //digitalWrite(fil[i],HIGH);
      digitalWrite(col[i], LOW);
      for (int j = 0; j < 8; j++) {
        //digitalWrite(col[j],LOW);
        digitalWrite(fil[j], HIGH);
      }
    }
  }
}
```

```
void mostrarCaracterDriver(int letras[8][8]) {
  int fil_aux = 0;
  int col_aux = 7;
```

```
  for (int i = 0; i < 8; i++) {
    for (int j = 0; j < 8; j++) {
      lc.setLed(0, fil_aux, col_aux, letras[i][j]);
      fil_aux++;
    }
    fil_aux = 0;
    col_aux--;
  }
}



void conteosindriver(int letras[8][8]) {
  //int fil_aux=0;
  int col_aux = 7;
  for (int repe = 0; repe < 30; repe++) {
    for (int i = 0; i < 8; i++) {
      //digitalWrite(fil[i],LOW);
      digitalWrite(col[i], HIGH);
      for (int j = 0; j < 8; j++) {
        if (letras[i][j] == 1) {
          //digitalWrite(fil[j], HIGH);
          digitalWrite(fil[col_aux], LOW);
        }
        col_aux--;
      }
      col_aux = 7;
      delay(1);
      //digitalWrite(fil[i],HIGH);
      digitalWrite(col[i], LOW);
      for (int j = 0; j < 8; j++) {
        //digitalWrite(col[j],LOW);
        digitalWrite(fil[j], HIGH);
      }
    }
  }
}


void conteocondriver(int letras[8][8]) {
  int fil_aux = 0;
  int col_aux = 7;
  for (int i = 0; i < 8; i++) {
    for (int j = 0; j < 8; j++) {
      lc.setLed(0, fil_aux, col_aux, letras[i][j]);
      fil_aux++;
    }
    fil_aux = 0;
```

```
    col_aux--;
  }
}




void win() {
LimpiarMatriz();

    mostrarCaracterDriver(W1);
    mostrarCaracter(ESPACIO);
    delay(800);
    LimpiarMatriz();

    mostrarCaracterDriver(I1);
    mostrarCaracter(W1);
    delay(800);
    LimpiarMatriz();

    mostrarCaracterDriver(N1);
    mostrarCaracter(I1);
    delay(800);
    LimpiarMatriz();

    mostrarCaracterDriver(ESPACIO);
    mostrarCaracter(N1);
    delay(800);
    LimpiarMatriz();

    mostrarCaracterDriver(ESPACIO);
    mostrarCaracter(ESPACIO);
    delay(800);
    LimpiarMatriz();



}
```