Universidad de San Carlos de Guatemala

Facultad de Ingeniería, Arquitectura de computadoras y ensambladores 1

Sección N, aux. Ronald Marín

**Práctica 1**

Carlos Eduardo Soto Marroquín 201902502

Carlos Javier Martinez Polanco 201709282

Jefferson Gamaliel Molina Barrios 201945242

Gladys Leticia Ajuchan Vicente 201807389

Douglas Alexander Soch Catalán 201807032

# Manual Técnico

## Visualización general del circuito:

Gráfico

Descripción generada automáticamente

## Visualización del Arduino y pines utilizados:

Gráfico

Descripción generada automáticamente con confianza media

## Listado de pines utilizados:

* Digitales (PWM):

Estos pines van conectados a la terminar serial, permiten el este tipo de comunicación con el Arduino.

* + 0: rx
  + 1: tx
* Analógicos:

Estos pines son conectados a los potenciómetros para controlarlos por medio del Arduino.

* + A0: MOV\_VELOCIDAD
  + A1: MOV\_VIDAS
* Digitales (I/O):

Estos pines son conectados a los dispositivos de entradas para el Arduino y salidas del Arduino a otros dispositivos.

* + 47 – 53
  + 22 – 37

## Componentes del sistema:

* Arduino UNO (simulino UNO en proteus)
* 2 potenciómetros
* 4 Push buttons.
* 5 resistencias de 10k
* 2 matriz de led de 8x8
* 1 controlador para matriz de led MAX7219

## Librerias para Arduino:

* LedControl

## Librerias para Proteus:

* Simulino

## Estructuras:

// Texto de inicio

const short LONGITUD\_TEXTO = 24;

bool cadena[8][LONGITUD\_TEXTO] = {

{ 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0 },

{ 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0 },

{ 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0 },

{ 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0 },

{ 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0 },

{ 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0 },

{ 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0 },

{ 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0 },

};

bool tablero[8][16] = {

{ 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 },

{ 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 }

};

bool menu\_principal[8][16] = {

{ 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0 },

{ 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0 },

{ 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0 },

{ 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0 },

{ 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0 },

{ 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0 },

{ 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0 },

{ 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0 }

};

bool numeros[10][8][4] = {

{

{ 0, 0, 0, 0 },

{ 1, 1, 1, 1 },

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 },

{ 1, 1, 1, 1 }

},

{

{ 0, 0, 0, 0 },

{ 0, 1, 1, 0 },

{ 0, 1, 1, 0 },

{ 1, 1, 1, 0 },

{ 0, 1, 1, 0 },

{ 0, 1, 1, 0 },

{ 0, 1, 1, 0 },

{ 1, 1, 1, 1 }

},

{

{ 0, 0, 0, 0 },

{ 1, 1, 1, 1 },

{ 0, 0, 0, 1 },

{ 0, 0, 0, 1 },

{ 1, 1, 1, 1 },

{ 1, 0, 0, 0 },

{ 1, 0, 0, 0 },

{ 1, 1, 1, 1 }

},

{

{ 0, 0, 0, 0 },

{ 1, 1, 1, 1 },

{ 0, 0, 0, 1 },

{ 0, 0, 0, 1 },

{ 1, 1, 1, 1 },

{ 0, 0, 0, 1 },

{ 0, 0, 0, 1 },

{ 1, 1, 1, 1 }

},

{

{ 0, 0, 0, 0 },

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 },

{ 1, 1, 1, 1 },

{ 0, 0, 0, 1 },

{ 0, 0, 0, 1 },

{ 0, 0, 0, 1 }

},

{

{ 0, 0, 0, 0 },

{ 1, 1, 1, 1 },

{ 1, 0, 0, 0 },

{ 1, 0, 0, 0 },

{ 1, 1, 1, 1 },

{ 0, 0, 0, 1 },

{ 0, 0, 0, 1 },

{ 1, 1, 1, 1 }

},

{

{ 0, 0, 0, 0 },

{ 1, 1, 1, 1 },

{ 1, 0, 0, 0 },

{ 1, 0, 0, 0 },

{ 1, 1, 1, 1 },

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 },

{ 1, 1, 1, 1 }

},

{

{ 0, 0, 0, 0 },

{ 1, 1, 1, 1 },

{ 0, 0, 0, 1 },

{ 0, 0, 0, 1 },

{ 0, 1, 1, 1 },

{ 0, 0, 0, 1 },

{ 0, 0, 0, 1 },

{ 0, 0, 0, 1 }

},

{

{ 0, 0, 0, 0 },

{ 1, 1, 1, 1 },

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 },

{ 1, 1, 1, 1 },

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 },

{ 1, 1, 1, 1 }

},

{

{ 0, 0, 0, 0 },

{ 1, 1, 1, 1 },

{ 1, 0, 0, 1 },

{ 1, 0, 0, 1 },

{ 1, 1, 1, 1 },

{ 0, 0, 0, 1 },

{ 0, 0, 0, 1 },

{ 0, 0, 0, 1 }

}

};

bool caracter\_nivel[8][8] = {

{ 0, 0, 0, 0, 0, 0, 0, 0 },

{ 0, 1, 0, 0, 0, 0, 1, 0 },

{ 0, 1, 1, 0, 0, 0, 1, 0 },

{ 0, 1, 0, 1, 0, 0, 1, 0 },

{ 0, 1, 0, 0, 1, 0, 1, 0 },

{ 0, 1, 0, 0, 0, 1, 1, 0 },

{ 0, 1, 0, 0, 0, 0, 1, 0 },

{ 0, 0, 0, 0, 0, 0, 0, 0 }

};

struct Bomba {

short x = -1;

short y = -1;

};

Bomba bombas[16];

bool caracter\_vidas[8][8]={

{ 0, 1, 1, 0, 0, 1, 1, 0 },

{ 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1 },

{ 1, 1, 1, 1, 1, 1, 1, 1 },

{ 0, 1, 1, 1, 1, 1, 1, 0 },

{ 0, 0, 1, 1, 1, 1, 0, 0 },

{ 0, 0, 0, 1, 1, 0, 0, 0 }

};

struct Digitos {

int digito1;

int digito2;

};

## Manejo de pines matriz sin controlador:

#include <Arduino.h>

#include "matriz\_sin\_driver.h"

// Pines que se utilizan en el arduino para manejar la matriz sin controlador

short columnas[] = { 37, 36, 35, 34, 33, 32, 31, 30 };

short filas[] = { 29, 28, 27, 26, 25, 24, 23, 22 };

void inicializarMatrizSinDriver() {

//Se definen las salidas

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

pinMode(filas[i], OUTPUT);

pinMode(columnas[i], OUTPUT);

}

}

void seleccionarFila(int fila) {

if (fila == 0) digitalWrite(filas[0], LOW);

else digitalWrite(filas[0], HIGH);

if (fila == 1) digitalWrite(filas[1], LOW);

else digitalWrite(filas[1], HIGH);

if (fila == 2) digitalWrite(filas[2], LOW);

else digitalWrite(filas[2], HIGH);

if (fila == 3) digitalWrite(filas[3], LOW);

else digitalWrite(filas[3], HIGH);

if (fila == 4) digitalWrite(filas[4], LOW);

else digitalWrite(filas[4], HIGH);

if (fila == 5) digitalWrite(filas[5], LOW);

else digitalWrite(filas[5], HIGH);

if (fila == 6) digitalWrite(filas[6], LOW);

else digitalWrite(filas[6], HIGH);

if (fila == 7) digitalWrite(filas[7], LOW);

else digitalWrite(filas[7], HIGH);

}

void setearEstadoEnColumna(int columna, bool estado) {

if (columna == 0) digitalWrite(columnas[0], estado);

if (columna == 1) digitalWrite(columnas[1], estado);

if (columna == 2) digitalWrite(columnas[2], estado);

if (columna == 3) digitalWrite(columnas[3], estado);

if (columna == 4) digitalWrite(columnas[4], estado);

if (columna == 5) digitalWrite(columnas[5], estado);

if (columna == 6) digitalWrite(columnas[6], estado);

if (columna == 7) digitalWrite(columnas[7], estado);

}

## Codigo principal del sistema:

#include <LedControl.h>

#include "estructuras.h"

#include "matriz\_sin\_driver.h"

// Mensaje

int offset = 0; // Indica la posicion actual del movimiento del mensaje

bool msg\_hacia\_izquierda = false; // Indica la direccion de recorrido que dara el mensaje inicial

int potenciometro = 0; // Indica la velocidad del movimiento del mensaje

// Pines generales

short pinPotenciometroVelocidad = A0; // Pin para manejar el potenciometro que representa la velocidad del mensaje y del avion

short pinPotenciometroVidas = A1; // Pin para manejar el potenciometro que representa la cantidad de vidas que se tendra en una partida nueva

short pinDisparo = 50; // Pin para manejar el boton de disparo

short pinDerecha = 49; // Pin para manejar el boton con movimiento hacia la derecha

short pinIzquierda = 48; // Pin para manejar el boton con movimiento hacia la izquierda

short pinIni = 47; // Pin para manejar el boton de inicio

LedControl matriz\_driver = LedControl(51, 53, 52, 1); // Matriz con driver

// Juego

short vidas = 3; // Indica la cantidad de vidas que tiene en el juego actual

short nivel = 1; // Indica el nivel actual de juego

short edificios\_a\_destruir = 0; // Indica la cantidad de edificios que hay que destruir en el nivel actual de juego

short edificios\_destruidos = 0; // Indica la cantidad de edificios que el usuario a destruido desde que ha iniciado el juego

short pos\_x\_avion = 0; // Indica la posicion del avion en el eje x de la matriz

short pos\_y\_avion = -3; // Indica la posicion del avion en el eje y de la matriz

bool avion\_hacia\_izquierda = false; // Indica la direccion de movimiento del avion

String estado\_app = "MENSAJE"; // Indica el estado actual en el que esta el programa; Estos pueden ser: MENSAJE, MENU, JUGAR, PAUSA, ESTADISTICA, CONFIGURACION

short estadisticas[5] = { 0, 0, 0, 0, 0 }; // Sirve para almacenar los 5 mejores puntajes

void setup() {

inicializarMatrizSinDriver(); // Inicializando la matriz sin driver

Serial.begin(9600); // Inicializando la comunicacion serial

// Inicializando la matriz con driver

matriz\_driver.shutdown(0, false);

matriz\_driver.setIntensity(0, 8);

matriz\_driver.clearDisplay(0);

}

long t0 = millis();

long t1 = millis();

void loop() {

if (estado\_app == "MENSAJE") {

// Definiendo la velocidad de recorrido del mensaje

t1 = millis();

if (t1 - t0 >= 200) {

offset++;

t0 = millis();

}

// Imprimiendo el mensaje en ambas matrices

imprimirMensajeMatrizSinDriver();

imprimirMensajeMatrizConDriver();

} else if (estado\_app == "MENU") {

imprimirMenuPrincipal();

} else if (estado\_app == "JUGAR") {

// Muestra el nivel y redibuja los edificios en el tablero en el caso que no halla mas edificios por destruir

if (edificios\_a\_destruir == 0) {

// Obtiene el tiempo actual en milisegundos

unsigned long tiempoInicio = millis();

// Muestra el mensaje del nivel durante 2 segundos

while (millis() - tiempoInicio < 2000) {

imprimirMensajeNivel();

}

// Dibuja los edificios la matriz 'tablero'

dibujarEdificios();

}

// Definiendo la velocidad de recorrido del avion

t1 = millis();

if (t1 - t0 >= 200) {

offset++;

t0 = millis();

movimientoBomba();

movimientoAvion();

}

// Dibujando el avion

dibujarAvion();

dibujarBomba();

imprimirTablero();

}else if (estado\_app == "PAUSA"){

imprimirMenuPausa(vidas);

}

// potenciometro = map(analogRead(A0), 0, 1024, 200, 800);

// Escuchando los botones

botonK();

botonDerecho();

botonIzquierdo();

botonDisparo();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\* JUEGO \*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Imprime todos los estados de la variable 'tablero'

// en la matriz sin driver y en la matriz con driver

void imprimirTablero() {

// Matriz sin driver

for (int i = 0; i < 9; i++) {

seleccionarFila(i);

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

setearEstadoEnColumna(j, tablero[i][j]);

}

delay(1);

}

// Matriz con driver

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

for (int j = 8; j < 16; j++) {

matriz\_driver.setLed(0, i, 15 - j, tablero[i][j]);

}

}

}

// Dibuja el avion en el tablero

void dibujarAvion() {

if (avion\_hacia\_izquierda) {

// Imprime la parte alta del avion

if (pos\_y\_avion >= 0 && pos\_y\_avion < 16) {

tablero[pos\_x\_avion][pos\_y\_avion] = 1;

}

// Imprime la parte baja del avion

for (int i = 0; i < 3; i++) {

if (pos\_y\_avion - i >= 0 && pos\_y\_avion - i < 16) {

tablero[pos\_x\_avion + 1][pos\_y\_avion - i] = 1;

}

}

// Si el avion colisiona contra un edificio se sube 2 unidades hacia arriba

if (pos\_y\_avion - 3 >= 0 && pos\_y\_avion - 3 < 16 && tablero[pos\_x\_avion + 1][pos\_y\_avion - 3] == 1) {

vidas--;

if (vidas > 0) {

movimientoAvion();

pos\_x\_avion = pos\_x\_avion - 3;

} else {

reiniciarJuego();

estado\_app = "MENSAJE";

}

}

// Si el avion llega a la parte final del tablero se baja un nivel y se reaparece del lado contrario

if (pos\_y\_avion < 0) {

pos\_y\_avion = 18;

pos\_x\_avion++;

}

} else {

// Imprime la parte alta del avion

if (pos\_y\_avion >= 0 && pos\_y\_avion < 16) {

tablero[pos\_x\_avion][pos\_y\_avion] = 1;

}

// Imprime la parte baja del avion

for (int i = 0; i < 3; i++) {

if (pos\_y\_avion + i >= 0 && pos\_y\_avion + i < 16) {

tablero[pos\_x\_avion + 1][pos\_y\_avion + i] = 1;

}

}

// Si el avion colisiona contra un edificio se sube 2 unidades hacia arriba

if (pos\_y\_avion + 3 >= 0 && pos\_y\_avion + 3 < 16 && tablero[pos\_x\_avion + 1][pos\_y\_avion + 3] == 1) {

vidas--;

if (vidas > 0) {

movimientoAvion();

pos\_x\_avion = pos\_x\_avion - 3;

} else {

reiniciarJuego();

estado\_app = "MENSAJE";

}

}

// Si el avion llega a la parte final del tablero se baja un nivel y se reaparece del lado contrario

if (pos\_y\_avion > 16) {

pos\_y\_avion = -3;

pos\_x\_avion++;

}

}

}

// Desplaza el avion en el tablero

void movimientoAvion() {

if (avion\_hacia\_izquierda) {

if (pos\_y\_avion >= 0 && pos\_y\_avion < 16) {

tablero[pos\_x\_avion][pos\_y\_avion] = 0;

}

for (int i = 0; i < 3; i++) {

if (pos\_y\_avion - i >= 0 && pos\_y\_avion - i < 16) {

tablero[pos\_x\_avion + 1][pos\_y\_avion - i] = 0;

}

}

pos\_y\_avion--;

} else {

if (pos\_y\_avion >= 0 && pos\_y\_avion < 16) {

tablero[pos\_x\_avion][pos\_y\_avion] = 0;

}

for (int i = 0; i < 3; i++) {

if (pos\_y\_avion + i >= 0) {

tablero[pos\_x\_avion + 1][pos\_y\_avion + i] = 0;

}

}

pos\_y\_avion++;

}

}

// Dibuja la bomba en el tablero

void dibujarBomba() {

for (int i = 0; i < 16; i++) {

if (bombas[i].x != -1 && bombas[i].y != -1) {

tablero[bombas[i].x + 1][bombas[i].y] = 1;

}

}

}

// Desplaza la bomba en el tablero

void movimientoBomba() {

for (int i = 0; i < 16; i++) {

if (bombas[i].x != -1 && bombas[i].y != -1) {

// Colosiona contra un edificio

if (tablero[bombas[i].x + 2][bombas[i].y] == 1) {

// Se acumula el edificio destruido

edificios\_destruidos++;

// Se agrega la cantidad de edificios que se ha destruido durante el juego en las estadisticas

for (int i = 0; i < sizeof(estadisticas) / sizeof(estadisticas[0]); i++) {

if (edificios\_destruidos > estadisticas[i]) {

estadisticas[i] = edificios\_destruidos;

break;

}

}

// En el caso que halla destruido 5 edificios se le agregara una vida extra

if (edificios\_destruidos % 5 == 0) {

vidas++;

}

// Se borra el edificio

for (int j = bombas[i].x + 1; j < 8; j++) {

tablero[j][bombas[i].y] = 0;

}

// Se borra la bala

tablero[bombas[i].x + 1][bombas[i].y] = 0;

bombas[i].x = -1;

bombas[i].y = -1;

// Se disminuye la cantidad de edificios a destruir

edificios\_a\_destruir--;

// Se pasa al siguiente nivel una vez halla destruido todos los edificios y en el caso que sobrepase el nivel 10,

// se va a la pantalla principal finalizando asi el juego

if (edificios\_a\_destruir == 0) {

nivel++;

if (nivel > 10) {

reiniciarJuego();

estado\_app = "MENSAJE";

} else {

movimientoAvion();

pos\_y\_avion = avion\_hacia\_izquierda ? 18 : -3;

pos\_x\_avion = 0;

}

}

}

// Colisiona con la parte baja de la matriz

else if (bombas[i].x >= 6) {

tablero[bombas[i].x + 1][bombas[i].y] = 0;

bombas[i].x = -1;

bombas[i].y = -1;

}

// Movimiento de la bomba

else {

tablero[bombas[i].x + 1][bombas[i].y] = 0;

bombas[i].x = bombas[i].x + 1;

}

}

}

}

// Dibuja los edificios que hay segun el nivel actual del juego

void dibujarEdificios() {

short indice = 0;

short largo = 0;

short cantidad\_edificios = nivel + 2;

edificios\_a\_destruir = cantidad\_edificios;

while (cantidad\_edificios > 0) {

indice = random(16);

largo = random(1, 5);

if (!tablero[7][indice]) {

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

tablero[7 - i][indice] = 1;

}

cantidad\_edificios--;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\* BOTONES \*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Reconoce el boton ini presionado

unsigned long tiempo\_boton\_presionado;

void botonK() {

int btnK = digitalRead(pinIni);

if (btnK == HIGH) {

if (tiempo\_boton\_presionado == 0) {

tiempo\_boton\_presionado = millis();

}

} else {

if (tiempo\_boton\_presionado != 0) {

unsigned long lapso\_tiempo = millis() - tiempo\_boton\_presionado;

tiempo\_boton\_presionado = 0;

// Entra al juego

if (estado\_app == "MENSAJE" && lapso\_tiempo >= 2000) {

estado\_app = "MENU";

}

// Entra a la pausa del juego

else if (estado\_app == "JUGAR" && lapso\_tiempo >= 2000) {

estado\_app = "PAUSA";

}

// Entra a la pausa del juego

else if (estado\_app == "PAUSA" && lapso\_tiempo >= 2000) {

estado\_app = "JUGAR";

}

// Regresar al menu principal

else if (estado\_app == "PAUSA" && (lapso\_tiempo >= 2000 && lapso\_tiempo < 3000)) {

estado\_app = "MENU";

}

}

}

}

// Reconoce el boton de disparo presionado

bool estado\_boton\_dis = false;

bool ultimo\_estado\_boton\_dis = false;

unsigned long ultimo\_tiempo\_rebote\_boton\_dis = 0;

const unsigned long delay\_rebote\_boton\_dis = 50;

void botonDisparo() {

int btnDisparo = digitalRead(pinDisparo);

if (btnDisparo != ultimo\_estado\_boton\_dis) {

ultimo\_tiempo\_rebote\_boton\_dis = millis();

}

if ((millis() - ultimo\_tiempo\_rebote\_boton\_dis) > delay\_rebote\_boton\_dis) {

if (btnDisparo != estado\_boton\_dis) {

estado\_boton\_dis = btnDisparo;

if (estado\_boton\_dis == LOW) {

if (estado\_app == "MENU") {

estado\_app = "ESTADISTICA";

} else if (estado\_app == "JUGAR") {

for (int i = 0; i < sizeof(bombas) / sizeof(bombas[0]); i++) {

if (bombas[i].x == -1 && bombas[i].y == -1) {

if (avion\_hacia\_izquierda) {

bombas[i].y = pos\_y\_avion - 1;

} else {

bombas[i].y = pos\_y\_avion + 1;

}

bombas[i].x = pos\_x\_avion + 1;

break;

}

}

}

}

}

}

ultimo\_estado\_boton\_dis = btnDisparo;

}

// Reconoce el boton derecho presionado

bool estado\_boton\_der = false;

bool ultimo\_estado\_boton\_der = false;

unsigned long ultimo\_tiempo\_rebote\_boton\_der = 0;

const unsigned long delay\_rebote\_boton\_der = 50;

void botonDerecho() {

int btnDerecha = digitalRead(pinDerecha);

if (btnDerecha != ultimo\_estado\_boton\_der) {

ultimo\_tiempo\_rebote\_boton\_der = millis();

}

if ((millis() - ultimo\_tiempo\_rebote\_boton\_der) > delay\_rebote\_boton\_der) {

if (btnDerecha != estado\_boton\_der) {

estado\_boton\_der = btnDerecha;

if (estado\_boton\_der == LOW) {

if (estado\_app == "MENSAJE") {

msg\_hacia\_izquierda = false;

} else if (estado\_app == "MENU") {

estado\_app = "CONFIGURACION";

} else if (estado\_app == "JUGAR") {

movimientoAvion();

avion\_hacia\_izquierda = false;

}

}

}

}

ultimo\_estado\_boton\_der = btnDerecha;

}

// Reconoce el boton izquierdo presionado

bool estado\_boton\_izq = false;

bool ultimo\_estado\_boton\_izq = false;

unsigned long ultimo\_tiempo\_rebote\_boton\_izq = 0;

const unsigned long delay\_rebote\_boton\_izq = 50;

void botonIzquierdo() {

int btnIzquierda = digitalRead(pinIzquierda);

if (btnIzquierda != ultimo\_estado\_boton\_izq) {

ultimo\_tiempo\_rebote\_boton\_izq = millis();

}

if ((millis() - ultimo\_tiempo\_rebote\_boton\_izq) > delay\_rebote\_boton\_izq) {

if (btnIzquierda != estado\_boton\_izq) {

estado\_boton\_izq = btnIzquierda;

if (estado\_boton\_izq == LOW) {

if (estado\_app == "MENSAJE") {

msg\_hacia\_izquierda = true;

} else if (estado\_app == "MENU") {

estado\_app = "JUGAR";

} else if (estado\_app == "JUGAR") {

movimientoAvion();

avion\_hacia\_izquierda = true;

}

}

}

}

ultimo\_estado\_boton\_izq = btnIzquierda;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\* NIVEL \*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void imprimirMensajeNivel() {

// Matriz sin driver

for (int i = 0; i < 9; i++) {

seleccionarFila(i);

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

setearEstadoEnColumna(j, caracter\_nivel[i][j]);

}

delay(1);

}

// Matriz con driver - Primer digito

int primer\_digito = nivel / 10;

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

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

matriz\_driver.setLed(0, fila, 7 - columna, numeros[primer\_digito][fila][columna]);

}

}

// Matriz con driver - Segundo digito

int segundo\_digito = nivel - primer\_digito \* 10;

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

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

matriz\_driver.setLed(0, fila, 3 - columna, numeros[segundo\_digito][fila][columna]);

}

}

}

void reiniciarJuego() {

nivel = 1;

edificios\_a\_destruir = 0;

pos\_x\_avion = 0;

pos\_y\_avion = -3;

edificios\_destruidos = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void imprimirMenuPrincipal() {

// Matriz sin driver

for (int i = 0; i < 9; i++) {

seleccionarFila(i);

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

setearEstadoEnColumna(j, menu\_principal[i][j]);

}

delay(1);

}

// Matriz con driver

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

for (int j = 8; j < 16; j++) {

matriz\_driver.setLed(0, i, 15 - j, menu\_principal[i][j]);

}

}

}

/\*\*\*\*\*\*IMPRIMIR MENU PAUSA\*\*\*\*\*\*/

void imprimirMenuPausa(int vidas){

Digitos digitos = obtenerDigitos(vidas);

int digito1 = digitos.digito1;

int digito2 = digitos.digito2;

Serial.println(digito1);

Serial.println(digito2);

for (int i = 0; i < 9; i++) {

seleccionarFila(i);

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

setearEstadoEnColumna(j, caracter\_vidas[i][j]);

}

delay(1);

}

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

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

matriz\_driver.setLed(0, fila, 7 - columna, numeros[digito1][fila][columna]);

}

}

// Matriz con driver - Segundo digito

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

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

matriz\_driver.setLed(0, fila, 3 - columna, numeros[digito2][fila][columna]);

}

}

}

Digitos obtenerDigitos(int numero) {

Digitos digitos;

digitos.digito1 = numero / 10;

digitos.digito2 = numero % 10;

return digitos;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\* CADENA \*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void imprimirMensajeMatrizSinDriver() {

for (int i = 0; i < 9; i++) {

seleccionarFila(i);

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

offset = offset >= LONGITUD\_TEXTO ? 0 : offset;

if (msg\_hacia\_izquierda) {

setearEstadoEnColumna(j, cadena[i][(j + offset) % LONGITUD\_TEXTO]);

} else {

setearEstadoEnColumna(j, cadena[i][(j - offset < 0 ? LONGITUD\_TEXTO + j - offset : j - offset)]);

}

}

delay(1);

}

}

void imprimirMensajeMatrizConDriver() {

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

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

if (msg\_hacia\_izquierda) {

short offset\_aux = 8 + offset + j;

matriz\_driver.setLed(0, i, 7 - j, cadena[i][offset\_aux >= LONGITUD\_TEXTO ? offset\_aux - LONGITUD\_TEXTO : offset\_aux]);

} else {

short offset\_aux = 8 - offset + j;

matriz\_driver.setLed(0, i, 7 - j, cadena[i][offset\_aux < 0 ? LONGITUD\_TEXTO + offset\_aux : offset\_aux]);

}

}

}

}

# Manual de Usuario

## Panel de control del juego:

Gráfico

Descripción generada automáticamente