

Universidad de San Carlos  
Facultad de Ingeniería  
Escuela de Ciencias y Sistemas  
Arquitectura de Computadores y Ensambladores 1  
Sección N

GRUPO:

**#10**



**Integrantes:**

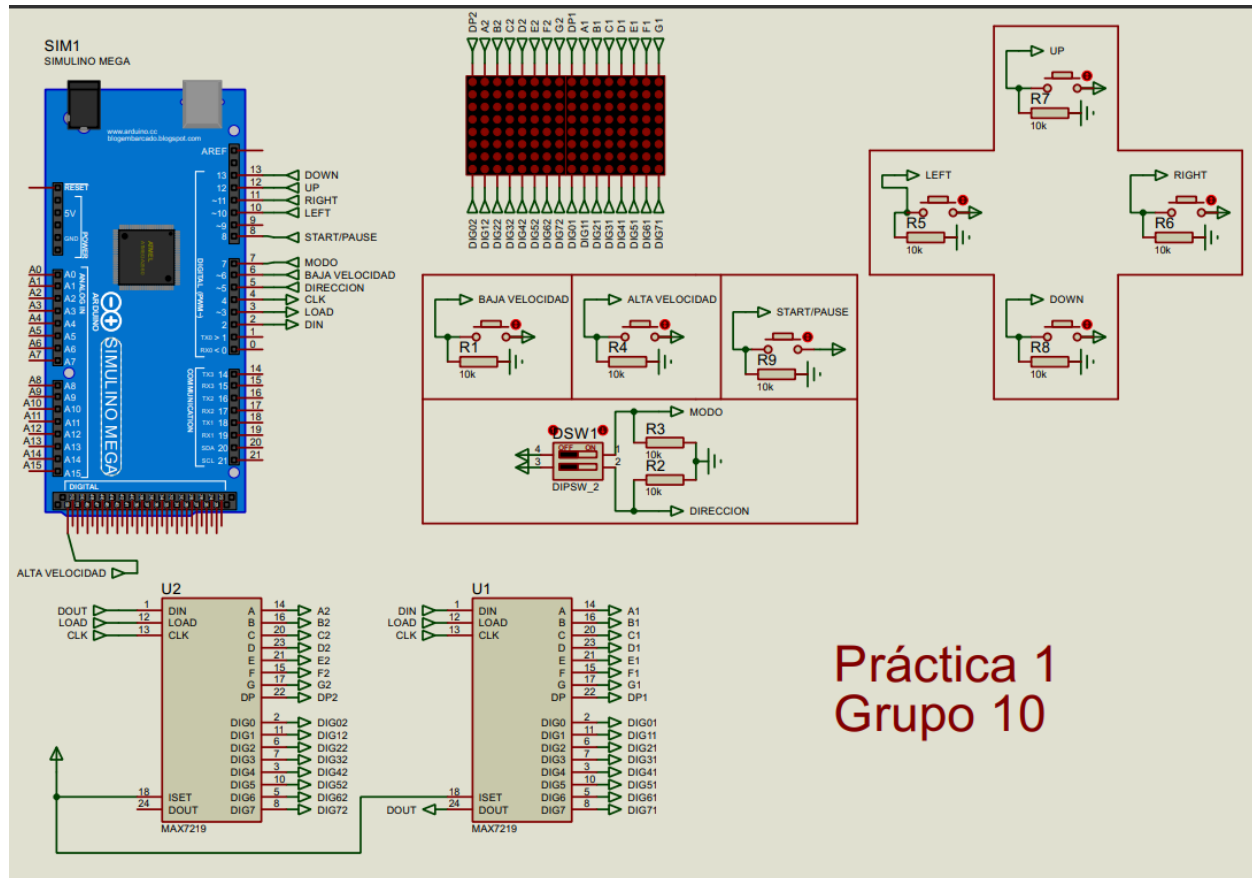
<b>Carné:</b>	<b>Nombre:</b>
201020252	Sergio Ariel Ramírez Castro
201800469	José Alejandro Lorenty Herrera
201800555	César Alejandro Sosa Enríquez
201801366	Josue Guillermo Orellana Cifuentes
201801397	Cristian Francisco Meoño Canel

**Fecha de entrega: 10 de junio de 2021**

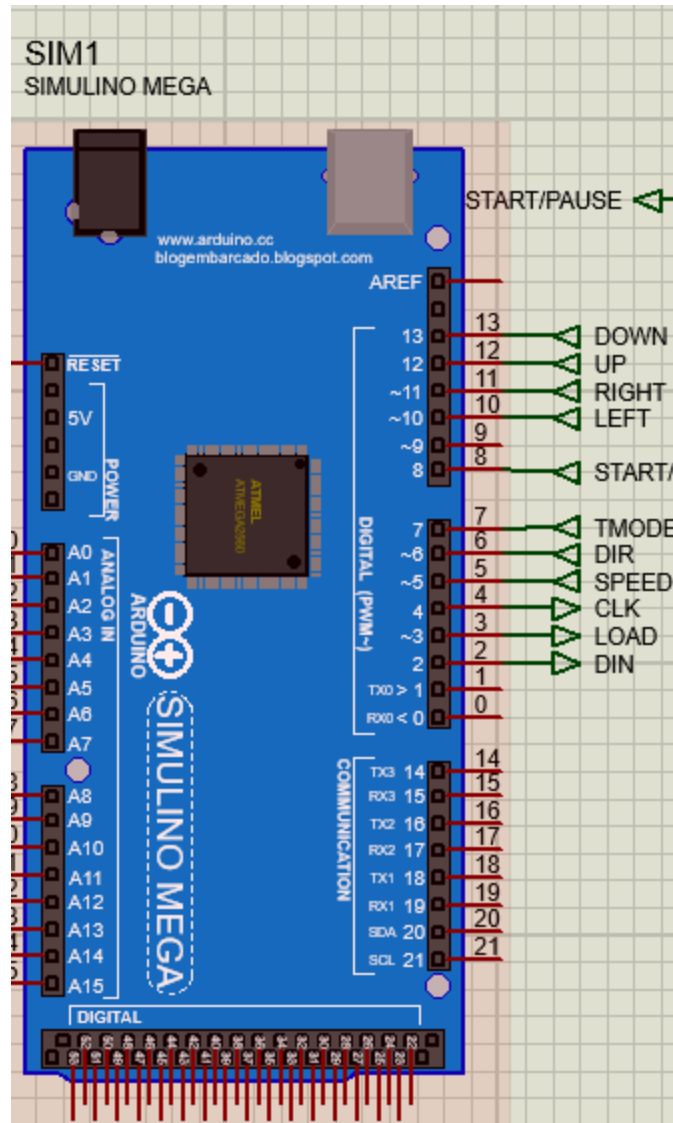
## **Introducción**

En el mundo de los componentes y la electrónica se tienen varias combinaciones las cuales pueden generar distintas opciones para crear y desarrollar actividades de aprendizaje como laborales, en el área de controladores y dispositivos electrónicos se tiene el dispositivo y herramienta conocida como ARDUINO, este dispositivo más su desarrollo de programación basado en C, crea la oportunidad de desarrollar mecánicas de interacción y desarrollo para que se pueda crear desde un letrero animado hasta un juego simple pero con fundamentos lógicos y sistémicos.

## Circuitos desarrollados:

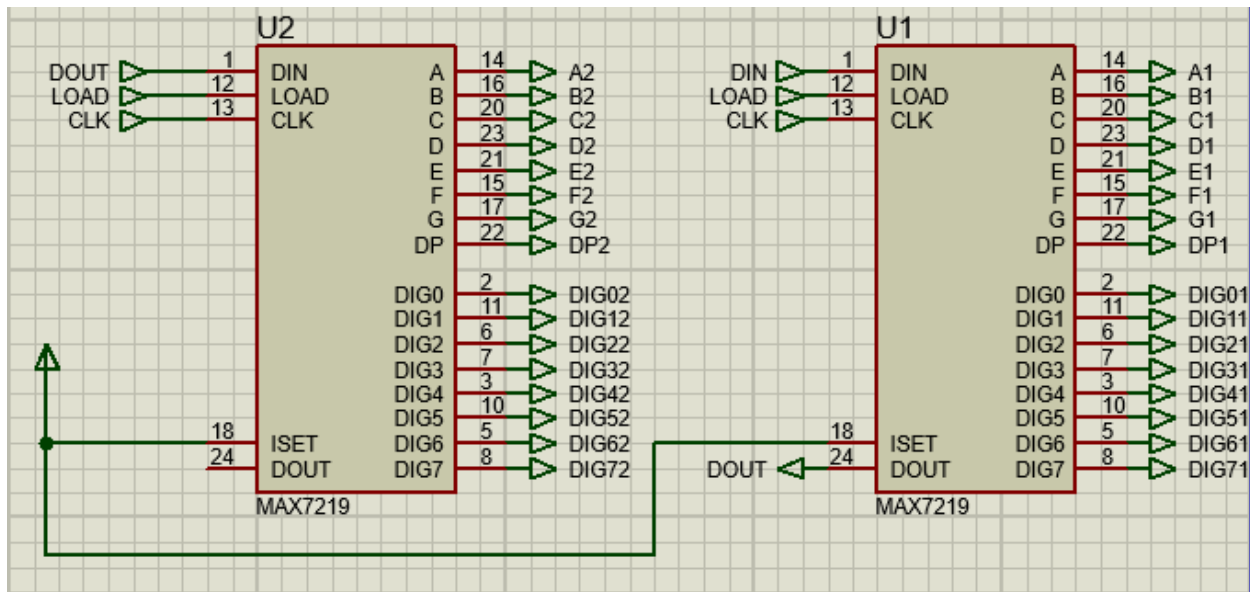


Circuito general de toda la aplicación, se puede apreciar ambos controladores de matriz que se tienen conectados de manera secuencial, debido a que las librerías implementadas desarrollan o controlan dichos dispositivos de esta manera, todo está conectado con pines debido a que se determina con mayor orden las conexiones del circuito.

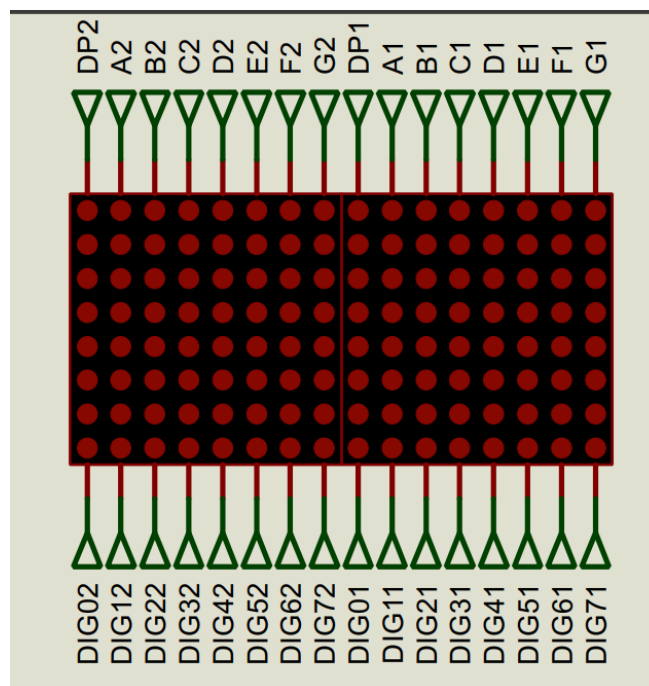


En el ARDUINO, SIMULINO MEGA 250, se tienen 3 pines de salida los cuales son las entradas de carga, reloj y la información de entrada de los controladores MAX72, estos además tienen la función de controlar las 2 matrices de 8x8x leds.

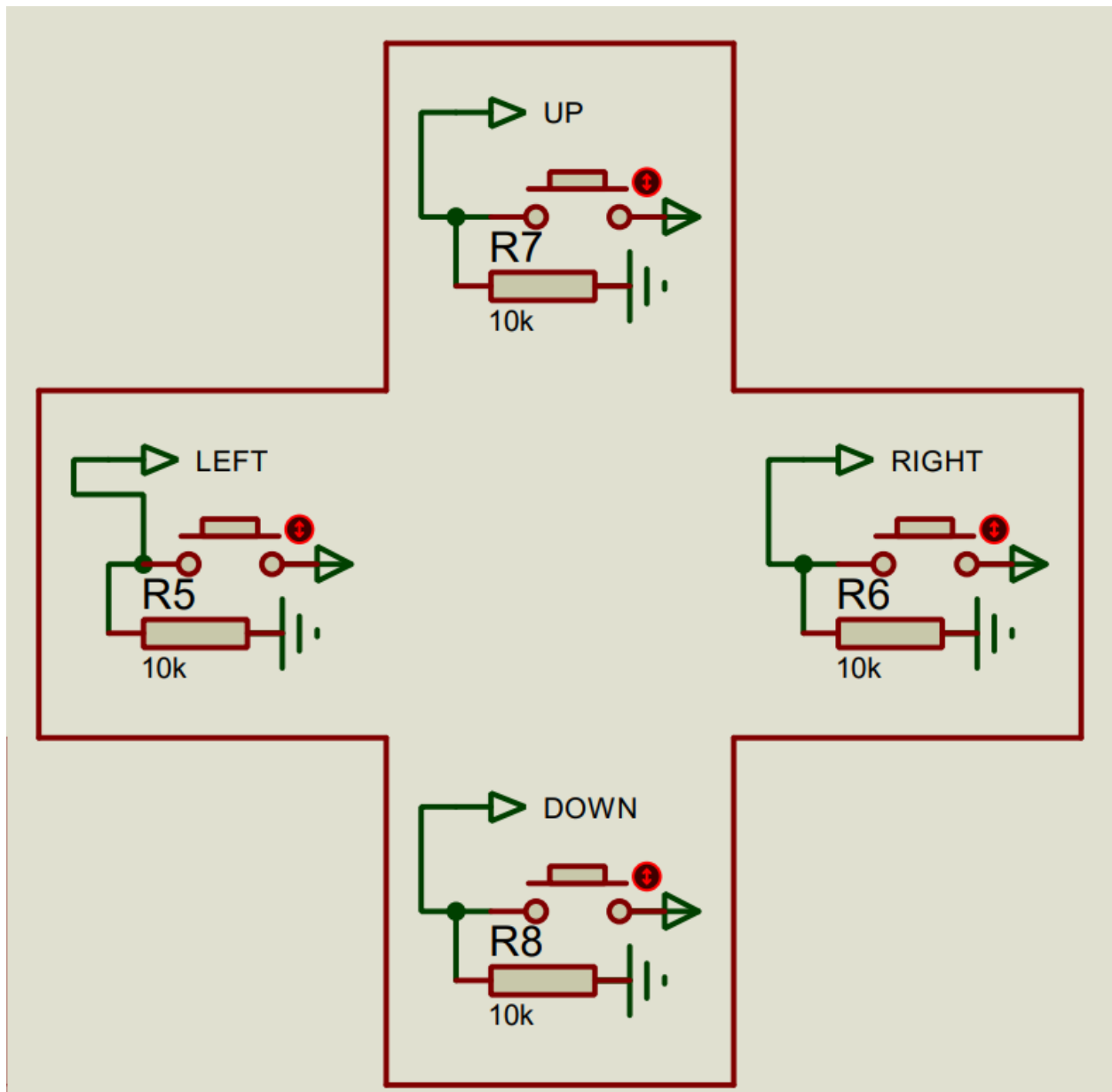
En el caso de demás pines, todos son entradas de lectura para el ARDUINO, todos estos dan pulsos de ceros y unos el cual este dispositivo tiene la capacidad de entender y con lógica de programación entrar y generar programas.



Estos dispositivos facilitan el control de la matriz, creando componentes además de implementar librerías para la manipulación de las matrices de leds.



Las pantallas de 8x8 se toman los pines de arriba como las columnas y los pines de abajo como las filas, esto es importante para determinar cómo y cuándo encendran los leds de la matriz, además los 0 y 1 lógicos de cada matriz.



Los botones son los cuales envían la señal alta o baja para que pueda trabajar ARDUINO y generar los cambios necesarios en el proyecto o entrada de variables.

### Código Utilizado:

```
#include <MD_Parola.h>
#include <MD_MAX72xx.h>
#include <SPI.h>

#include "LedControl.h"
..
```

Las librerías utilizadas son para la generación del texto de todas las funciones requeridas además de la facilitación para utilizar las matrices con sus respectivos controladores.

```
#define HARDWARE_TYPE MD_M/
//const int MAX_DEVICES = 2;
//const int DIN = 2;
//const int LOAD = 3;
//const int CLK = 4;
const int speedB = 5;
const int dirB = 6;
const int tText = 7;
const int startB = 8;
const String individualText
//BOTONES PARA JUEGO
const int leftB = 11;
const int rightB = 10;
const int upB = 12;
const int downB = 13;
```

La mayoría de los pines están reservados con sus respectivos nombres, esto para facilitar su utilización y búsqueda.

```

void MAINTEXT() {
  if(changeText == 1){
    displayMatrix.displayText("TP1 - GRUPO 3 - SECCION A", PA_LEFT, 100, 0, PA_SCROLL_RIGHT, PA_SCROLL_RIGHT);
    changeText = 0;
  }
  if(digitalRead(dirB)==HIGH){
    displayMatrix.setTextEffect(PA_SCROLL_LEFT, PA_SCROLL_LEFT); //texto de izquierda a derecha
    int slide_scroll_speed = map(speedLR, 1023, 0, 400,15);
    displayMatrix.setSpeed(slide_scroll_speed);
  }else{
    displayMatrix.setTextEffect(PA_SCROLL_RIGHT, PA_SCROLL_RIGHT); //texto de la derecha hacia la izquierda
    int slide_scroll_speed = map(speedLR, 1023, 0, 400,15);
    displayMatrix.setSpeed(slide_scroll_speed);
  }
  if(displayMatrix.displayAnimate()){
    displayMatrix.displayReset();
  }
}

```

El método "MAINTEXT" es el cual genera el texto y le agrega el efecto o cambio de efecto para que salga el texto en la pantalla, se crea una instancia del objeto de la matriz y se pone que desea lo que imprima.

```

//MÉTODO PARA LETRAS INDIVIDUALES
void individualLetter(String wordDisplay, int timeToWait){
  changeText = 1;
  displayMatrix.setTextAlignment(PA_CENTER);
  int longText = wordDisplay.length();
  for(int i=0; i<longText;i++){
    char letter = wordDisplay[i];
    displayMatrix.print(String(wordDisplay[i])+" "+String(wordDisplay[i]));
    delay(timeToWait);
    if(digitalRead(tText)== LOW){
      displayMatrix.displayClear();
      break;
    }
  }
}

```

Además, se tiene un método que genera las letras individualmente en las pantallas led.

```

//MÉTODO PARA AUMENTAR LA VELOCIDAD
void setupSpeed() {
  if(digitalRead(speedB)== LOW){
    speedLR = 150;
    waitTime = 350;
  }else{
    speedLR = 50;
    waitTime = 150;
  }
}

```

En un método se tiene la lectura de un botón para modificar la velocidad de la aplicación.



```

if(!gameover){
  draw();
  elapsedTime += currentTime - previousTime;
  if(elapsedTime > 500){
    move();
    eat();
    checkGameOver();
    elapsedTime = 0;
    readInput = true;
  }
  if(readInput){
    if(digitalRead(rightB) && !lastInput) { direction = (direction + 1) % 4; readInput = false; }
    if(digitalRead(leftB) && !lastInput) { direction = (4 + direction-1) % 4; readInput = false; }
  }
  lastInput = digitalRead(rightB) || digitalRead(leftB);
}else{
  if(gameover2 != 1){
    String gameOverText = "GAME OVER SCORE: "+String(score);
    individualLetterG(gameOverText,waitTime);
    gameOver2 = 1;
  }
  if(currentTime % 800 > 400)
    draw();
  gameover -= currentTime - previousTime;
  if(gameover <= 0)
    reset();
}

```

Este método tiene las funciones del juego lo cual genera que se inicie y empiece el contador y la animación en las pantallas led.

```

..
void setup() {

  Serial.begin(9600);
  //PINES
  pinMode(speedB, INPUT);
  pinMode(dirB, INPUT);
  pinMode(tText, INPUT);
  pinMode(startB, INPUT);
  pinMode(leftB, INPUT);
  pinMode(rightB, INPUT);
  pinMode(upB, INPUT);
  pinMode(downB, INPUT);

  //INICIALIZAR VARIABLES
  changeText = 1;
  //INICIALIZAR EL OBJETO MATRIZ
  displayMatrix.begin();
  //INTENSIDAD DE NUESTRA MATRIZ
  displayMatrix.setIntensity(5);
  //LIMPIAR TODO EL DISPLAY
  displayMatrix.displayClear();
}

```

El SETUP de nuestra aplicación solo se inicializan las variables y los pines para iniciar.



```

// .....
void loop() {
  setupSpeed();
  if(digitalRead(startB) == HIGH) {
    gameMode();
  } else {
    textMode();
  }
}
}

```

Y El LOOP solo tiene la selección de modo dependiendo del usuario.

Además del SKETCH se tiene un archivo .h el cual genera funcionalidad con el juego y la conexión entre la matriz:

 juego.h	2/12/2021 6:12 PM	H File	4 KB
 sketch_practica1	2/12/2021 5:50 PM	INO File	6 KB

En este se detalla más a profundidad la funcionalidad del juego.

```

/*DEFINICIONES PARA EL JUEGO "SNAKE"*/
#define largoSerpiente 24
/*DIMENSIONES DE LA MATRIZ DE JUEGO*/
#define altoZona 9
#define anchoZona 16
/*TIEMPO PARA REINICIAR EL JUEGO*/
#define tiempoFinJuego 3000
/*CONTROLES PARA EL JUEGO*/
#define arriba 0
#define derecha 1
#define abajo 2
#define izquierda 3

/*CONSTANTES PARA LOS OBJETOS MATRICES (INCLUIDA MD_Parola MATRIX)*/
const int MAX_DEVICES = 2;
const int DIN = 2;
const int LOAD = 3;
const int CLK = 4;

/*MATRIZ PARA EL JUEGO*/
LedControl matrizJuego = LedControl(DIN, CLK, LOAD, MAX_DEVICES);

```

Se definen las variables o constantes de área de la matriz como sus columnas y filas además de los pines los cuales están conectados a los controladores para tener el control. Además de las variables para el juego de los movimientos de la serpiente; esto para saber en qué dirección se mueve la serpiente y poder generar la jugabilidad.

```
typedef struct p {
    int x;
    int y;
} Coordenada;
```

Se creó un “STRUCT” para generar coordenadas en formato “X, Y” esto para facilitar el mapeado de todos los objetos que tengan que salir en la matriz, desde la comida hasta el cuerpo completo de la serpiente. El funcionamiento de las matrices Led es como un mapa de coordenadas fila columna, es por ello por lo que facilita la implementación.

```
int cabeza;
int cola;
/*LONGITUD DE LA SERPIENTE*/
int largoCuerpo;
/*DIRECCION HACIA DONDE SE DII
int direccion;
/*OVJETO PARA APARECER COMIDA
Coordenada comida;
/*FIN DE JUEGO Y REINCIAR EL J
int finJuego = 1;
/*ENTRAR EN EL TEXTO DE FIN DI
int finJuego2 = 1;
/*PUNTAJE DE LA PARTIDA*/
int puntaje;
/*VARIABLES PARA EL TIEMPO DE
int tiempoTranscurrido;
unsigned long tiempoAnterior;
/*VARIABLES PARA LA LECTURA DE
int ultimoMovimiento;
bool leerMovimiento;
/*VELOCIDAD DEL JUEGO*/
int velocidadJuego;
//-----
```

Algunas variables del juego para las funciones como la dirección del movimiento, el tamaño de la serpiente, variables de cambio para verificar los estados de juego, el puntaje del jugador, tiempos para las pausas y el parpadeo de los leds, los movimientos registrados para la dirección de la serpiente, las coordenadas del cuerpo de la serpiente y la comida.

```

void colocarPixel(int fila, int columna) {
    if (columna < 8 ) {
        matrizJuego.setLed(0, fila, columna, true);
    }
    else {
        matrizJuego.setLed(1, fila, columna - 8, true);
    }
}

```

Este método utilizado para prender el led en el cual se ubica la validación de la posición de la serpiente o de la comida, como son dos matrices conectadas se requiere que cuando cruce de una a otra el cambio de columna sea coherente por ello se utiliza una condición dependiendo de ella ya se sabrá en que matriz está dicho led.

```

bool validarComida() {
    /*NO SE ACEPTAN NUMEROS NEGATIVOS*/
    if (comida.x < 0 || comida.y < 0) {
        return false;
    }
    /*NO SE ACEPTA QUE APAREZCAN ENCIMA DE LA SERPIENTE*/
    for (int i = cola; i <= (cabeza > cola ? cabeza : largoSerpiente - 1); i++) {
        if (cuerpo[i].x == comida.x && cuerpo[i].y == comida.y) {
            return false;
        }
    }
    if (cabeza < cola) {
        for (int i = 0; i <= cabeza; i++) {
            if (cuerpo[i].x == comida.x && cuerpo[i].y == comida.y) {
                return false;
            }
        }
    }
}

```

---

La comida debe generarse aleatoriamente pero no debe ser en coordenadas fuera de la matriz ni encima de la serpiente en movimiento, esto referencia a que generaría problemas de jugabilidad si alguno de los dos casos anteriores.

```

void verificarJuego() {
    for (int i = cola; i <= (cabeza > cola ? cabeza - 1 : largoSerpiente - 1); i++) {
        if (cuerpo[cabeza].x == cuerpo[i].x && cuerpo[cabeza].y == cuerpo[i].y) {
            finJuego2 = 0;
            finJuego = tiempoFinJuego;
            return;
        }
    }
    if (cabeza < cola) {
        for (int i = 0; i < cabeza; i++) {
            if (cuerpo[cabeza].x == cuerpo[i].x && cuerpo[cabeza].y == cuerpo[i].y) {
                finJuego2 = 0;
                finJuego = tiempoFinJuego;
                return;
            }
        }
    }
}

```

Para verificar que la serpiente no choque contra si misma se tienen 2 ciclos para verificar en cada movimiento si esta no esta encima de sí misma, así validar el fin de juego y cambiar de modo.

```

void aparecerComida() {
    while (!validarComida()) {
        /*LIMITES DE LA ZONA SON 15 PARA LA ANCHURA (0-15) Y 7 P
        comida = { random(altoZona - 1), random(anchoZona) };
    }
}

```

El método genera de manera automática la comida, siempre validando con el método anterior que no sea una posición invalida.

```

void dibujar() {
    /*DIBUJAR EL CUERPO DE LA SERPIENTE*/
    for (int i = cola; i <= (cabeza > cola ? cabeza : largoSerpiente - 1); i++) {
        colocarPixel(cuerpo[i].x, cuerpo[i].y);
        delay(5);
    }
    if (cabeza < cola) {
        for (int i = 0; i <= cabeza; i++) {
            colocarPixel(cuerpo[i].x, cuerpo[i].y);
            delay(5);
        }
    }
    /*REINICIAR MATRIZ 0*/
    matrizJuego.shutdown(0, false);
    matrizJuego.setIntensity(0, 15);
    matrizJuego.clearDisplay(0);
    /*REINICIAR MATRIZ 1*/
    matrizJuego.shutdown(1, false);
    matrizJuego.setIntensity(1, 15);
    matrizJuego.clearDisplay(1);
    /*DIBUJAR LA COMIDA EN LA MATRIZ*/
    colocarPixel(comida.x, comida.y);
    delay(5);
}

```

El método dibujar utiliza la colocación de pixel, para poder prender los leds de la comida y del cuerpo de la serpiente, luego se reinicia la matriz para establecer los leds que si debieran estar encendidos.

```

void comer() {
    /*SE VALIDA QUE LA CABEZA CRUCE POR ENCIMA DE LA COMIDA*/
    if (cuerpo[cabeza].x == comida.x && cuerpo[cabeza].y == comida.y) {
        if (largoCuerpo < largoSerpiente) {
            /*SINO SE HA ALCANZADO EL LARGO MAXIMO, SE AUMENTA*/
            largoCuerpo++;
            cola--;
            if (cola < 0) cola = largoSerpiente - 1;
        }
        velocidadJuego = velocidadJuego - 20;
        if (velocidadJuego <= 75) {
            velocidadJuego = 75;
        }
        /*SE AUMENTA EL PUNTAJE*/
        puntaje++;
        /*COMIDA INVALIDA PARA GENERAR UNA POSICION AL AZAR*/
        comida = { -1, -1 };
        /*APARECE NUEVA COMIDA*/
        aparecerComida();
    }
}

```

El método de comer genera un aumento de tamaño en la serpiente y un mayor puntaje, a su vez aumenta la velocidad de juego para que sea el nivel una interacción dinámica.