Universidad de San Carlos de Guatemala
Faculta de Ingeniería
Escuela de Ciencias y Sistemas
Arquitectura de Computadores y Ensambladores 1
Ing. Otto Escobar
Aux. David González,
Aux. Andhy Solís

PRACTICA No. 1

Juego Pong y Visualización de Texto en Matrices LED de 8x8
(Simulación Microcontrolador Arduino en Proteus)

GRUPO No. 13

Carné	*Nombre*
201800712	*Jairo Sebastian Ramírez Palacios*
201643762	*José Francisco Santos Salazar*
201700857	*Daniel Arturo Alfaro Gaitan*
201700471	*Sohany Ayleen López Salazar*

MANUAL TECNICO
PRÁCTICA 1

Descripción de la Practica

- Que se adquieran conocimientos, se aplique e interactúe con el microcontrolador Arduino.

Objetivo de la Práctica

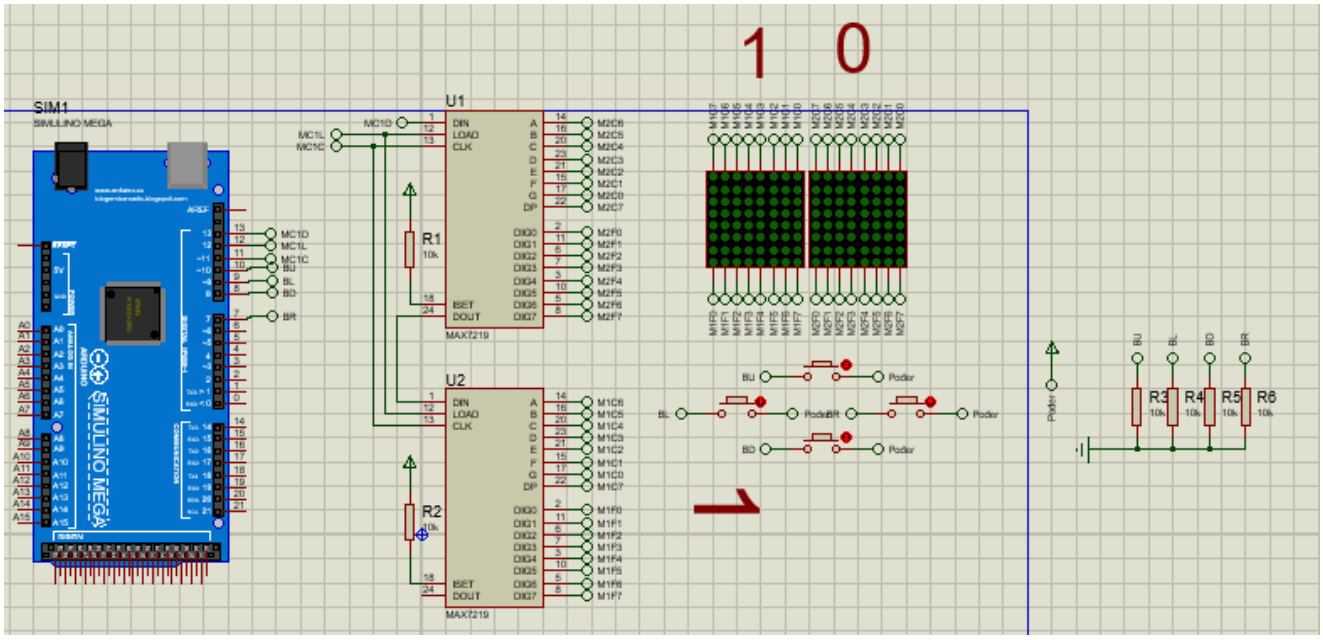
- Comprender el funcionamiento de las entradas y salidas, tanto digitales como análogas del microcontrolador Arduino.
- Comprender la configuración de las matrices de luces LED.
- Conocer las funciones básicas de salida serial.
- Aplicar el lenguaje C para estructuras de control en Arduino.

Componentes Utilizados

1. Simulino Mega
2. Controlador MAX7219
3. Matrices Led 8x8 Green
4. Resistor 10K
5. Botones
6. Software Proteus v8.5
7. Software Arduino IDE

Diseño de Simulación Proteus

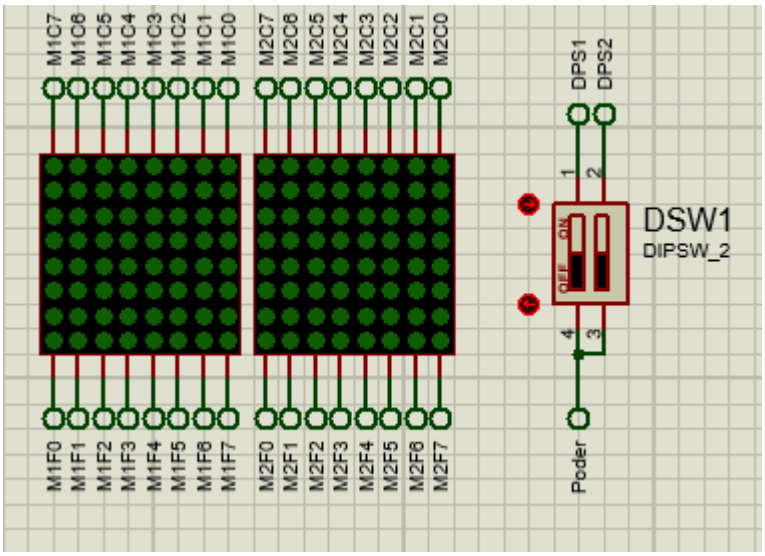
En la siguiente figura se muestra los componentes que se han incluido en el proyecto de Proteus para la simulación.



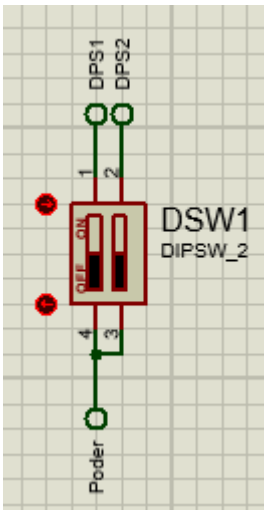
Código Arduino

Para la configuración del juego y de el letrero se realizó un programa en Arduino IDE, el cual contiene el código que fue cargado a Simulino Mega en Proteus para la ejecución de la Practica. Dicho código se encuentra en la sección de anexos.

Letrero Matrices 8x8



Las matrices muestran el texto ***TP1-GRUPO 13-SECCION A*;** el texto se desplaza hacia la derecha por ambos LEDS. Dependiendo de la configuración de los botones esto cambiará.



El botón de movimiento tiene la siguiente función:

Entrada 1	Función
0	Mostrar mensaje en movimiento.
1	Mostrar mensaje letra por letra sin movimiento.

Para el botón de dirección:

Entrada 2	Función
0	Desplazar mensaje de izquierda a derecha
1	Desplazar mensaje de derecha a izquierda

Con los botones de velocidad puede configurarse la rapidez con la que se muestra la frase en letrero , el botón de velocidad baja disminuye la misma y el botón de velocidad alta la aumenta, en todo caso ninguno de los botones se haya pulsado, el letrero se mostrara con una velocidad preestablecida.

Código Arduino del Letrero

Las frases o números que se muestran en las matrices estan almacenadas arrays tipo byte, cada letra o numero esta almacenada en 8 espacios de dicho array, los utilizados fueron:

Numeros:

```
byte letrero2[] {
 //O
 B00000000,
 B01111110,
 B11111111,
 B10000001,
 B10000001,
 B10000001,
 B01111110,
 B00000000,

 //1
 B00000000,
 B00100000,
 B01000000,
 B10000000,
 B11111111,
 B11111111,
 B00000000,
 B00000000,

 //2
 B00000000,
 B00000000,
 B00100111,
 B01001111,
 B10001001,
 B10010001,
 B01100001,
 B00000000,

 //3
 B00000000,
 B01000010,
 B10000001,
 B10010001,
 B11111111,
 B01101110,
 B00000000,
 B00000000,

 //4
 B00000000,
 B11111000,
 B00001000,
```

```
B00001000,
B00001000,
B11111111,
B11111111,
B00000000,

//5
B00000000,
B11110001,
B11110001,
B10010001,
B10010001,
B10011111,
B10001110,
B00000000,

//6
B00000000,
B01111110,
B11111111,
B10001001,
B10001001,
B10001001,
B10000110,
B00000000,

//7
B00000000,
B10000000,
B10000000,
B10000000,
B10000000,
B11111111,
B11111111,
B00000000,

//8
B00000000,
B01100110,
B11111111,
B10011001,
B10011001,
B10011001,
B01100110,
B00000000,
```

```
 //9
 B00000000,
 B01110001,
 B10001001,
 B10001001,
 B10001001,
 B11111111,
 B01111110,
 B00000000,
};
```

Frase *TP1-GRUPO13-SECCION A*

```
byte frase[]{
 // *
 B00000000,
 B00010000,
 B01010100,
 B00111000,
 B11111110,
 B00111000,
 B01010100,
 B00010000,

 // T
 B10000000,
 B10000000,
 B10000000,
 B11111111,
 B11111111,
 B10000000,
 B10000000,
 B10000000,

 //P
 B00000000,
 B11111111,
 B11111111,
 B10011000,
 B10011000,
 B10011000,
 B01100000,
 B00000000,

 // 1
 B00000000,
 B00100000,
```

```
B01000000,
B10000000,
B11111111,
B11111111,
B00000000,
B00000000,

//GUION
B00000000,
B00000000,
B00011000,
B00011000,
B00011000,
B00011000,
B00000000,
B00000000,

//G
B00000000,
B11111111,
B11111111,
B10000001,
B10001001,
B10001001,
B10001111,
B00000000,

//R
B00000000,
B00000000,
B11111111,
B11111111,
B10011000,
B10010100,
B01100011,
B00000000,

//U
B00000000,
B11111110,
B11111111,
B00000001,
B00000001,
B00000001,
B11111110,
B00000000,
```

```
//P
B00000000,
B11111111,
B11111111,
B10011000,
B10011000,
B10011000,
B01100000,
B00000000,

//O
B00000000,
B01111110,
B11111111,
B10000001,
B10000001,
B10000001,
B01111110,
B00000000,

//Espacio
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,

//1
B00000000,
B00100000,
B01000000,
B10000000,
B11111111,
B11111111,
B00000000,
B00000000,

//3
B00000000,
B01000010,
B10000001,
B10010001,
B10010001,
```

```
B01101110,
B00000000,
B00000000,

//GUION
B00000000,
B00000000,
B00011000,
B00011000,
B00011000,
B00011000,
B00000000,
B00000000,

//S
B00000000,
B01100001,
B11110001,
B10010001,
B10010001,
B10011111,
B10001110,
B00000000,

//E
B00000000,
B11111111,
B11111111,
B10011001,
B10011001,
B10000001,
B10000001,
B00000000,

//C
B00000000,
B11111111,
B11111111,
B10000001,
B10000001,
B10000001,
B10000001,
B00000000,
```

```
//C
B00000000,
B11111111,
B11111111,
B10000001,
B10000001,
B10000001,
B10000001,
B00000000,

//I
B00000000,
B10000001,
B10000001,
B11111111,
B11111111,
B10000001,
B10000001,
B00000000,

//O
B00000000,
B01111110,
B11111111,
B10000001,
B10000001,
B10000001,
B01111110,
B00000000,

//N
B00000000,
B11111111,
B11000000,
B00110000,
B00001100,
B00000011,
B11111111,
B00000000,

//Espacio
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,
```

```
  B00000000,
  B00000000,
  B00000000,

  //A
  B00000000,
  B01111111,
  B11001000,
  B11001000,
  B11001000,
  B11001000,
  B01111111,
  B00000000,

  //*
  B00000000,
  B00010000,
  B01010100,
  B00111000,
  B11111110,
  B00111000,
  B01010100,
  B00010000
};
```

Frase Game Over:

```
byte gameover[]{
  // G
  B00000000,
  B11111111,
  B11111111,
  B10000001,
  B10001001,
  B10001001,
  B10001111,
  B00000000,

  // A
  B00000000,
  B01111111,
  B11001000,
  B11001000,
  B11001000,
  B11001000,
  B01111111,
```

```
B00000000,

// M
B11111111,
B01000000,
B00100000,
B00010000,
B00100000,
B01000000,
B11111111,
B00000000,

// E
B00000000,
B11111111,
B11111111,
B10011001,
B10011001,
B10000001,
B10000001,
B00000000,

// ESPACIO
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,

// O
B00000000,
B01111110,
B11111111,
B10000001,
B10000001,
B10000001,
B01111110,
B00000000,

// V
B00000000,
B11000000,
B00110000,
```

```
  B00001100,
  B00000011,
  B00001100,
  B00110000,
  B11000000,

  // E
  B00000000,
  B11111111,
  B11111111,
  B10011001,
  B10011001,
  B10000001,
  B10000001,
  B00000000,

  // R
  B00000000,
  B00000000,
  B11111111,
  B11111111,
  B10011000,
  B10010100,
  B01100011,
  B00000000
};
```

Para el letrero en movimiento se usó el siguiente método; este tiene como parámetros el numero de matriz en la que se mostrará, el array que se mostrará, el tamaño del mismo y la dirección en la que se moverá, se usó un condicional if para la dirección y dependiendo de este se recorrió el array haciendo uso de un ciclo for:

```
void dinamicFrame1(int dem,byte arrayI[], int t, boolean spean){
   // (demora, arrayByte, sizeArrayByte, direccion)
   if(spean){
     for (int j = 0; j < t; j++)
     {
       // matriz 0 <- [0-7]
       for (int i = 0; i < 8; i++)
       {
         if(j-i == -1){
           for (int k = j,p = 7; k >= 0 && p >= 0; k--,p--)
           {
             lc.setColumn(0,p,arrayI[k]);
           }
```

```
    }
  }

  // matriz 1 <- [8-15]
  for (int i = 0; i < 8; i++)
  {
    if(j-i == 7){
      // matriz 0
      // 8
      for (int k = j,p = 7; p >= 0; k--,p--)
      {
        lc.setColumn(0,p,arrayI[k]);
      }
      // 8-8=0
      for (int k = j-8,p = 7; k >= 0 && p >= 0; k--,p--)
      {
        lc.setColumn(1,p,arrayI[k]);
      }
    }
  }

  // matriz 1 <- 16<
  if( j >= 16){
    // matriz 0
    //17
    for (int k = j, p = 7; p >= 0; k--,p--)
    {
      lc.setColumn(0,p,arrayI[k]);
    }
    // matriz 1
    // 17-7 = 10
    for (int k = j-8,p = 7; k >= 0 && p >= 0; k--,p--)
    {
      lc.setColumn(1,p,arrayI[k]);
    }
  }

  if(j == t-1){
    lc.clearDisplay(0);     // blanquea matriz
    lc.clearDisplay(1);     // blanquea matriz
    delay(10);
  } else {
    delay(dem);
  }
}
```

```
   }else {
    // modulador para envio de parametro de posicion de matriz
    int v = 0, a = 0;
    for (int j = t-1; j >= 0; j--){
     if(v < 8){
      for (int k = 0; k <= v; k++){
       lc.setColumn(1,k,arrayI[j+k]);
      }
     }else{
      for (int k = 0; k <= 7; k++){
       lc.setColumn(1,k,arrayI[j+k]);
      }
      // matriz 0
      if(a < 8){
       for (int k = 0; k <= a; k++){
        lc.setColumn(0,k,arrayI[j+k+8]);
       }
      }else{
       for (int k = 0; k <= 7; k++){
        lc.setColumn(0,k,arrayI[j+k+8]);
       }
      }
      a += 1;
     }
     v += 1;
     delay(dem);
     if(j == 0){
      lc.clearDisplay(0);    // blanquea matriz
      lc.clearDisplay(1);    // blanquea matriz
      delay(10);
     }
    }
   }
}
```

Para las letras sin movimiento se usó un método con los mismos parámetros que el método anterior, sin embargo los ciclos for de este método cambian de letra cuando el numero de iteración % 8 es igual a 0, el letrero sin movimiento se muestra en las dos matrices.

```
void staticFrame1(int dem,byte arrayI[], int t, boolean spean){
 bool chan = true;
 if(spean){
  for (int j = 0; j <= t; j++) // <<---
  {
```

```
    if(j == 0){
      dinamicArray1[j]= arrayI[j];
    }else if (j%8 != 0){
      dinamicArray1[j%8] = arrayI[j];
    }else {
      if(chan){
        for (int k = 0; k < 8; k++)     // bucle itera 8 veces por el array
        {
          lc.setColumn(0,k,dinamicArray1[k]);
        }
        memcpy(dinamicArray2,dinamicArray1,sizeof(dinamicArray1));
        chan = false;
        dinamicArray1[0]= arrayI[j];
      }else{
        for (int k = 0; k < 8; k++)     // bucle itera 8 veces por el array
        {
          lc.setColumn(0,k,dinamicArray1[k]);
          lc.setColumn(1,k,dinamicArray2[k]);
        }
        memcpy(dinamicArray2,dinamicArray1,sizeof(dinamicArray1));
        dinamicArray1[0]= arrayI[j];
      }
      if(j == t){
        delay(800);
        lc.clearDisplay(0);     // blanquea matriz
        lc.clearDisplay(1);     // blanquea matriz
      } else {
        delay(dem);
      }
    }
    // COLOCAR CONDICIONES DE CAMBIO DE VELOCIDAD
  }
}else {
  for (int j = t; j > 0; j--) // iteracion de array bit --->>
  {
    if (j%8 != 0){
      dinamicArray1[j%8-1] = arrayI[j-1];
      if (j%8 == 1){
        if(chan){
          for (int k = 0; k < 8; k++)     // bucle itera 8 veces por el array
          {
            lc.setColumn(1,k,dinamicArray1[k]);
          }
          memcpy(dinamicArray2,dinamicArray1,sizeof(dinamicArray1));
          chan = false;
        }else{
```

```
          for (int k = 0; k < 8; k++)     // bucle itera 8 veces por el array
          {
            lc.setColumn(1,k,dinamicArray1[k]);
            lc.setColumn(0,k,dinamicArray2[k]);
          }
          memcpy(dinamicArray2,dinamicArray1,sizeof(dinamicArray1));
        }

        if(j == 1){
          delay(800);
          lc.clearDisplay(0);     // blanquea matriz
          lc.clearDisplay(1);     // blanquea matriz
        } else {
          delay(dem);
        }
       }
     }else {
       dinamicArray1[7]= arrayl[j-1];
     }
     // COLOCAR CONDICIONES DE CAMBIO DE VELOCIDAD
   }
  }
}
```

El punteo se muestra en las dos matrices, dicho punteo se muestra en movimiento, para poder mostrar este número se separó el mismo en unidad, decena y centena con el siguiente método que tiene como parámetro int punteo:

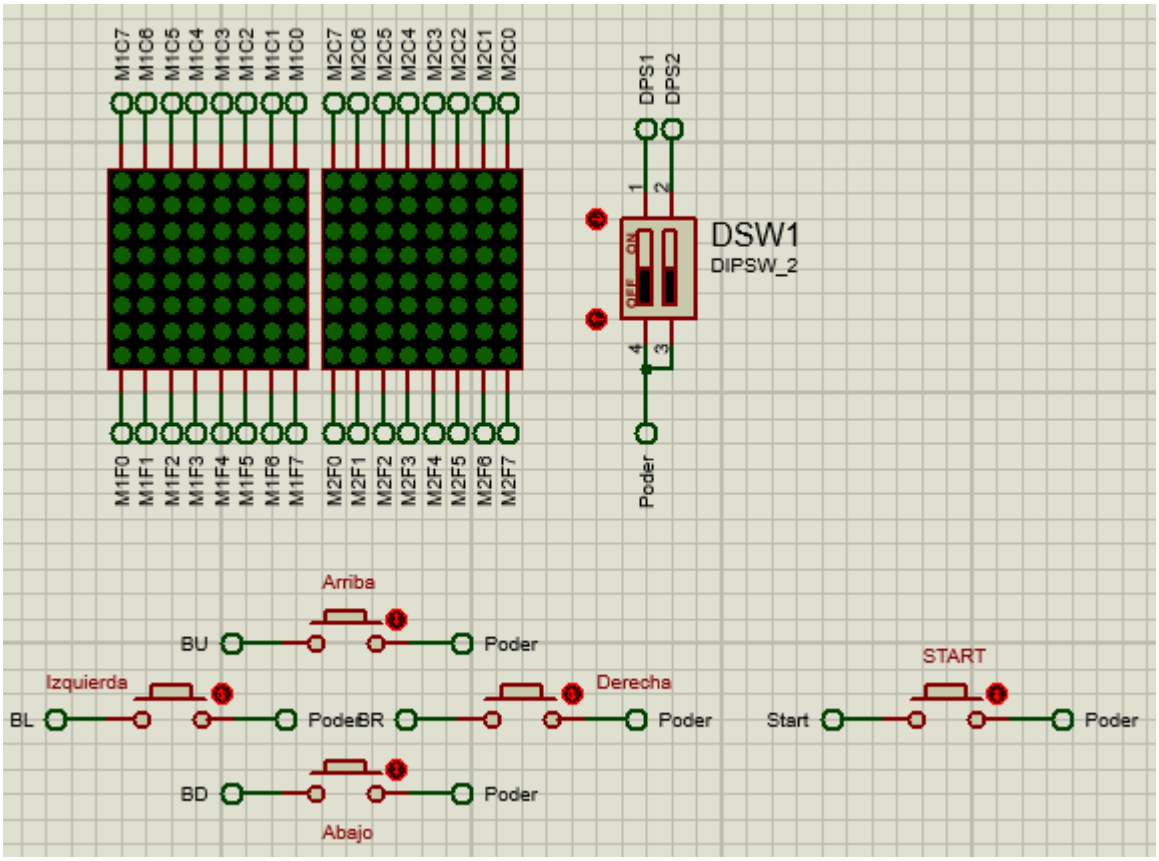
```
void punteo(int puntos){
  for(int i = 0; i< 24;i++){ matrixTemp[i] = pts[i];}
  int centena = puntos / 100;
  int decena = (puntos - centena * 100) / 10;
  int unidad = (puntos - centena * 100 - decena * 10);
  mostrar(centena,24);
  mostrar(decena,32);
  mostrar(unidad,40);
  dinamicFrame1(400,matrixTemp,sizeof(matrixTemp),true);
}
```

El siguiente método recibe como parámetro la unidad decena o centena, y el índice el cual indica en que posición en la que se almacenará dicho número en un array temporal que se llenara con el punteo del juego, y después de esto en el método anterior hará uso del método dinamicFrame1 para mostrar “Puntos” + punteo en las matrices:

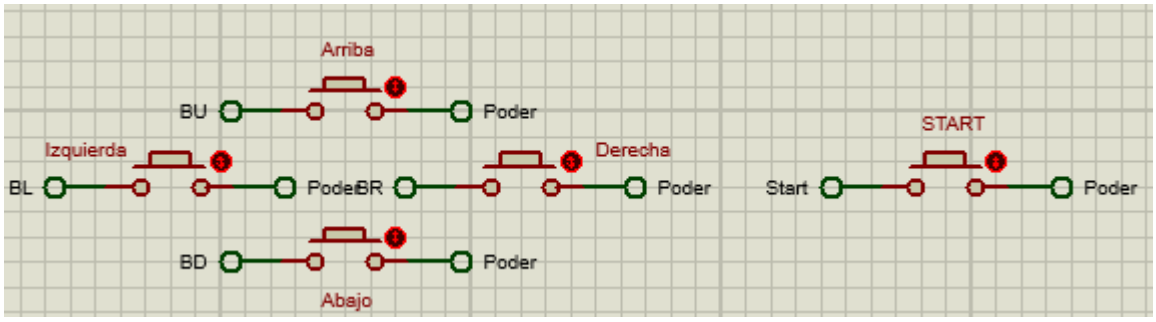
```
void mostrar(int numero,int indice){
  if(numero == 0){
    for(int i = 0; i<8;i++){
      matrixTemp[i+indice] = letrero2[i];
    }
  }else if(numero == 1){
    for(int i = 0; i<8;i++){
      matrixTemp[i+indice] = letrero2[i+8];
    }
  }else if(numero == 2){
    for(int i = 0; i<8;i++){
       matrixTemp[i+indice] = letrero2[i+16];
    }
  }
  else if(numero == 3){
    for(int i = 0; i<8;i++){
      matrixTemp[i+indice] = letrero2[i+24];
    }
  }
  else if(numero == 4){
    for(int i = 0; i<8;i++){
      matrixTemp[i+indice] = letrero2[i+32];
    }
  }
  else if(numero == 5){
    for(int i = 0; i<8;i++){
      matrixTemp[i+indice] = letrero2[i+40];
    }
  }
  else if(numero == 6){
    for(int i = 0; i<8;i++){
      matrixTemp[i+indice] = letrero2[i+48];
    }
  }
  else if(numero == 7){
    for(int i = 0; i<8;i++){
      matrixTemp[i+indice] = letrero2[i+56];
    }
  }
  else if(numero == 8){
    for(int i = 0; i<8;i++){
      matrixTemp[i+indice] = letrero2[i+64];
    }
  }
```

```
  else if(numero == 9){
    for(int i = 0; i<8;i++){
      matrixTemp[i+indice] = letrero2[i+72];
    }
  }
}
```

Funcionamiento del juego



Controles de dirección y start para el juego:



Código Arduino del juego

La serpiente se conforma por un array de 120 de tipo “Cuerpo”. Que es un struct, el cual cuenta con posición X y Y, un bool para saber si esa parte del cuerpo debe ser visible y tangible y una dirección.

```
struct Cuerpo{
   int X;
   int Y;
   bool Activo;
   int Direccion; //2 arriba, 1 izquierda, 0 abajo, 3 derecha
};
```

Las bolitas que se comerá la serpiente, son otro struct, el cual solo contará con posición X y Y.

```
struct Bolita{
   int X;
   int Y;
};
```

El juego snake está programado en la clase llamada “CuerpoSerpiente”.
El constructor configura el cuerpo de la serpiente, para que al principio solo los primeros 2 puntos del array sean tangibles. Y se les asigna una dirección inicial, que dependerá de donde se genere su posición inicial.
Su posición X y Y, son generados aleatoriamente con el método ColumnaInicio() y FilaInicio().

```
CuerpoSerpiente(){
        //Posicion inicial
        int Col = ColumnaInicio();
        int fil = FilaInicio();
      for (int n=0; n<120; n++){
        if (n==0 || n==1){
          cuerpo[n].Activo = true;
        } else {
           cuerpo[n].Activo = false;
        }
        cuerpo[n].Y = fil;
        if (Col == 15){
           cuerpo[n].X = Col + n;
           cuerpo[n].Direccion = 1;
        } else {
           cuerpo[n].X = Col - n;
           cuerpo[n].Direccion = 3;
        }
      }
      GenerarBolita();
    }
```

ColumnaInicio(), nos dará una ubicación binaria, eligiendo uno de los dos extremos, derecho o izquierdo. La función random(160), devuelve un entero, el cual con modulador a 2, se averigua si es divisible entre 2 o no, para así obtener un resultado binario con más probabilidades de cambio.

```
int ColumnaInicio(){
       srand(time(NULL));
       int inicioColumna = random(160);
       if (inicioColumna%2 == 0){
```

```
            //inicia izquierda
            return (0);
        } else {
            //inicia derecha
            return (15);
        }
}
```

FilaInicio(), funciona similar al de ColumnaInicio(), pero este solo devolverá un número entero del 0 al 7, para elegir en que fila empezará la serpiente al iniciar el juego.

```
int FilaInicio(){
    srand(time(NULL));
    int inicioFila = random(8);
    return inicioFila;
 }
```

FLUJO DEL JUEGO.

Para que el juego inicie, se necesita usar la función FlujoJuego, que devuelve un entero, el cual es la puntuación obtenida al finalizar este, ya sea por colisión o por mantener presionado el botón de start por más de 3 segundos.

Antes de iniciar, se solicita al jugador que ingrese la velocidad que desea, siendo 1 la más baja y 4 la más alta. Esta se selecciona con los botones arriba y abajo, para aumentar y disminuir la velocidad respectivamente. Para confirmar la selección se presiona el botón derecha.

Cuando el juego ha empezado, la serpiente se desplazará a la dirección que tenía al crearse, y en cada movimiento, detectará si hubo una colisión con alguna pared o consigo misma. En caso de no haber colisión, verificará si se comió alguna pelotita que encuentre por el camino, si se come una pelota, aumentará su puntuación y tamaño en 1. Cada punto servirá más adelante para aumentar la velocidad.

La función CambioJuegoMensaje(), reconocerá si el botón de start es presionado, en caso de ser así, detectará si se presiona por más de 3 segundos que provocará que salgamos del modo juego al modo mensaje. Si se suelta antes de cumplir este tiempo, el juego entrará en pausa, mostrando el punteo acumulado hasta el momento. Para salir del modo pausa, se detecta si se presiona el botón de start nuevamente, en caso de mantener el botón de pausa por más de 3 segundos, también nos sacará del modo juego y nos pasa al modo mensaje.

Entrada(Serpiente), detecta si se presiona alguno de los siguientes botones: Arriba, abajo, izquierda y derecha. Dependiendo de cual se presione, este devolverá un número entero 0 para abajo, 1 para izquierda, 2 para arriba y 3 para derecha. Este valor será trasladado a desplazamiento. Si no se detecta algún botón, el valor que se devuelve será el mismo que ya tenía la cabeza de la serpiente.

Desplazamiento lo que hace es que cada parte del cuerpo desde la cola hasta la cabeza adquiera la posición y dirección de su miembro más al frente. Hasta llegar a la cabeza, la cual adquirirá una posición hacia la dirección que tenga este en sus parámetros.

La velocidad deberá variar según la que se haya seleccionado más 5 veces la cantidad de puntos que tenga. Reduciendo el delay entre desplazamientos.

```
int FlujoJuego(CuerpoSerpiente Serpiente, LedControl lc){
        //Solicitud de velocidad para el juego
        int FracSpeed=0;
         FracSpeed = velocidadSerpiente(arriba, abajo, derecha, lc);
```

```
    //Juego iniciado
    while (true){
       Serpiente.Desplazamiento();
       if (Serpiente.Colision()){
          return (Serpiente.Puntos);
       }
       Serpiente.VerificarPelota();
       Serpiente.DibujarSerpiente(lc);
       int pausa = CambioJuegoMensaje();
       if (pausa == 1){
        bool modoPausa = true;
        while(modoPausa){
         //MostrarMensaje
         delay(400);
         pausa = CambioJuegoMensaje();
         if (pausa == 1){
           modoPausa = false;
         } else if (pausa == 2){
           return (Serpiente.Puntos);
         }
        }
       } else if (pausa == 2){
        return (Serpiente.Puntos);
       }
       int entrada = Entrada(Serpiente);
       Serpiente.CambiarDireccion(entrada);
       int velocidad = 1000/FracSpeed -5*Serpiente.Puntos;
       delay(velocidad);
       entrada = Entrada(Serpiente);
    }
}
```