UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

ARQUITECTURA DE ENSAMBLADORES 1

SECCIÓN “A”

Ing. OTTO ESCOBAR LEIVA

Aux. SUSEL RETANA

Segundo Semestre 2020

PRACTICA 1

Fecha: 21 de agosto del 2020.

Nombre	Carnet
Adrian Byron Ernesto Alvarado Alfaro	201700308
Carlos Alejandro Tenes Mejía	201700317
Ludwing Gabriel Paz Hernández	201700521
Byron Antonio Orellana Alburez	201700733
Jackeline Alexandra Benitez Benitez	201709166

INDICE

Arduino

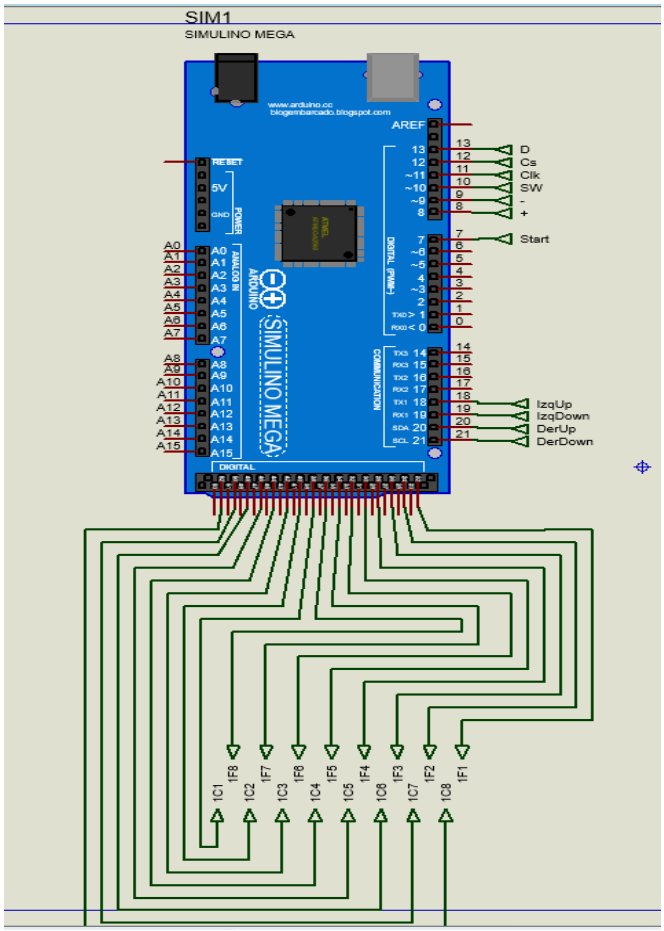

Imagen 1 Conexión Arduino

En la Imagen 1 se muestra la conexión del Arduino Mega con sus pines de entrada y salida. Para manejar la matriz sin controlador se utilizaron los pines digitales del 22 al 52, utilizando únicamente los pines pares con modo output. Del pin 22 al 36 se utilizaron para manejar las filas de la matriz 2, mientras que del pin 38 al 52 se utilizaron para manejar las columnas de la matriz 2. De igual modo los pines del 11 al 13 se utilizaron en modo output para conectarlo con el MAX7219.

Los pines del 7 al 10 se utilizaron en modo input para leer las entradas de los botones de Start, cambio de velocidad, cambio de direcciones y los botones que manejan los pads de los jugadores.

MAX 7219

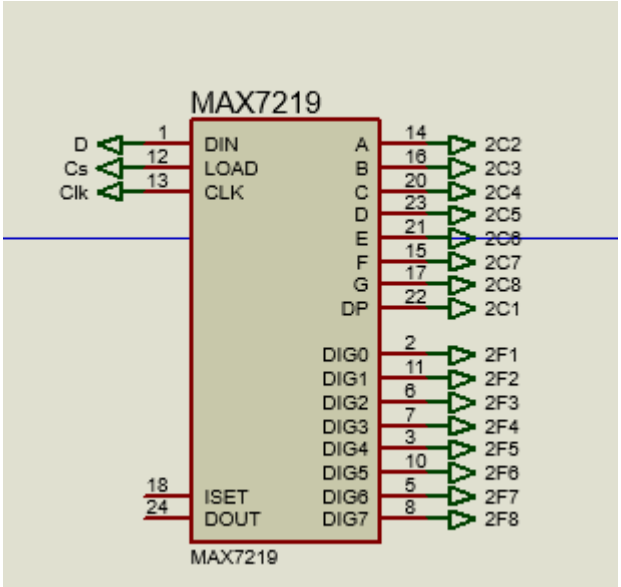

Imagen 2 Controlador Max 7219

En la Imagen 2, se muestra las conexiones correspondientes en el integrado de controlador Max7219. Siendo los pines 1, 12 y 13 entradas desde el Arduino. Los pines 14, 16, 20,23,21,15,17 y 22 se utilizan para manejar las columnas de la matriz 1, mientras que los pines 2, 11, 6, 7, 3, 10, 5 y 8 se utilizan para manejar las filas de la matriz 1.

VISUALIZACION DE PANTALLA

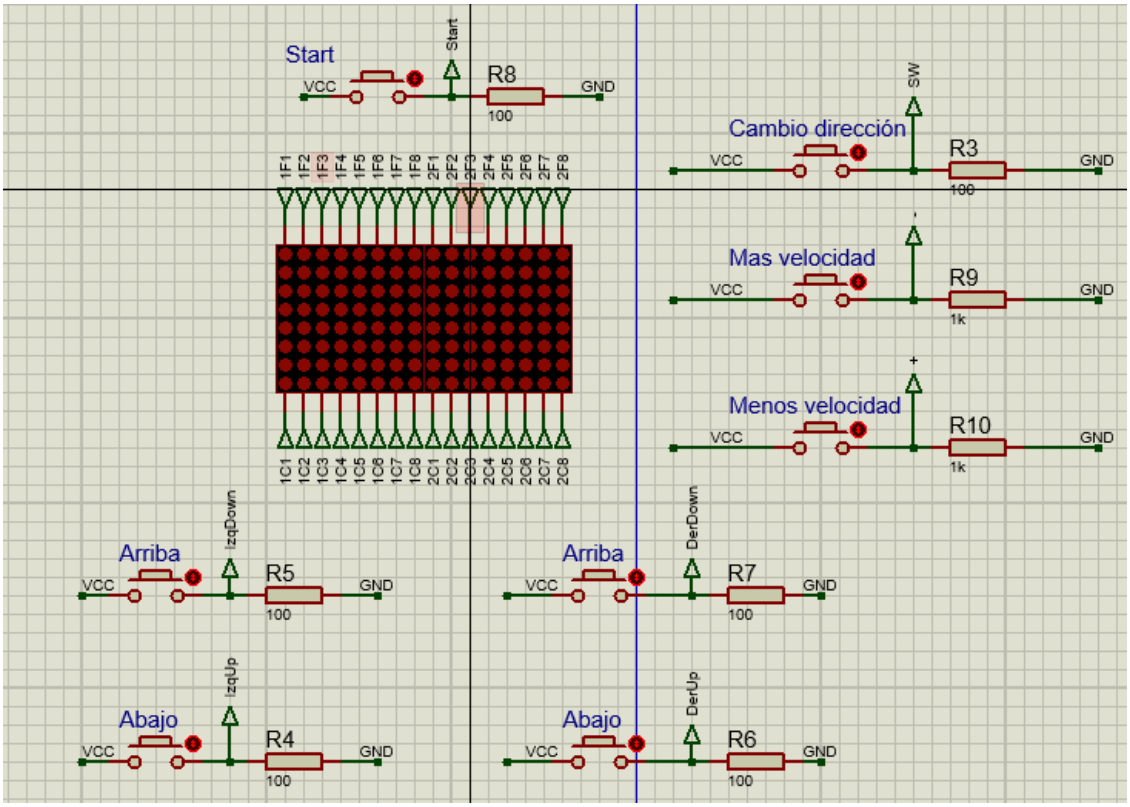

Imagen 3 Visualización de Pantalla

En la imagen 3, se muestran ambas matrices y sus respectivos botones para uso del juego, una estará conectada al max7219 y la otra estará conectada directamente al Arduino mega. A continuación, podremos visualizar que tendremos botones alrededor de nuestra matriz, los cuales tendrán distintas funciones dentro de la ejecución del juego, o bien dentro de nuestro letrero. A continuación, se describe la función de cada botón:

Botón Start: Su funcionalidad es poder cambiar del mensaje mostrado dentro de las matrices a el juego de ping pong, dejando pulsado el mismo por 3 segundos para poder realizar el cambio. Este a su vez cumple con otra funcionalidad una vez inicializado el juego este será el botón de pausa, el cual el marcador actual reemplazando el juego de ping pong, este volverá a la normalidad cuando se vuelva a presionar el botón start para poder seguir en donde se dejó pausado el juego.

Botón cambio de dirección: Su funcionalidad es realizar el cambio de dirección del mensaje. Este mensaje se mostrará al inicio de derecha a izquierda y cuando se presione este botón cambiará de sentido de izquierda a derecha.

Botón Mas velocidad: Su funcionalidad es incrementar la velocidad en la que se muestra el mensaje.

Botón Menos velocidad: Su funcionalidad es decrementar la velocidad en la que se muestra el mensaje.

Botón arriba: Su funcionalidad es mover el pad hacia arriba.

Botón abajo: Su funcionalidad es mover el pad hacia abajo.

ESTRUCTURACION DEL CODIGO

Bloque Loop

```
//Verifica el boton Start para entrar al juego o a pausa,luego se dirige a juego o a letrero dependiendo de Banderita
void loop() {
  verificarBotonStart();
  switch (banderita) {
    case true:
      jueguito();
      break;
    case false:
      Pantalla();
      break;
  }
}
```

Bloque Mensaje

```
//VARIABLES DE MENSAJE DE ENTRADA
bool tablero[8][16];                //representación de matriz
byte letrero[num][8][16] = {ASTERISCO, T, P, UNO, LINE, G, R, U, P, O, TRES, LINE,  S, E, C, C, I, O, N, B, ASTERISCO};                //Mensaje a mostrar
int sumador;                //variable que representa la velocidad media
int corrimiento;                //variable que acumula el desplazamiento
int retraso;                //variable que guarda la disminución de velocidad
int rapidez;                //variable que guarda el aumento de velocidad
int Cont = 0;                //variable que representa el número de byte del mensaje
int contFila = 0;                //variable que cuenta el número de filas
bool derizq = 1;                //variable que fija la direccion, hacia la derecha
int pinDir = 10;                //Pin de cambio de direccion
int pinStart = 7;                //Pin de Start que genera cambio entre juego y letrero
//FIN VARIABLES DE MENSAJE DE ENTRADA
```

Aquí se declararon todas las variables utilizadas en el mensaje que se muestra, entre ellos algunos pines que se utilizaron, la cadena de los bytes del mensaje, y otras variables usadas en la funcionalidad

Bloque de Juego

```
//VARIABLES DE JUEGO
LedControl inputEntrada = LedControl(13, 11, 12, 1);                //Objeto para controlar la matriz a través del MAX7219
byte pnts[5][8][16] = {CERO, UNO, DOS, TRES, CUATRO};                //Mensaje del marcador
int pinIzqUp = 18, pinIzqDown = 19;                //Pines pad izquierda arriba abajo
int pinDerUp = 20, pinDerDown = 21;                //Pines pad derecha arriba abajo
unsigned long Ptiempo = 0;                //Variable que define el tiempo anterior START
```

```
unsigned long Ctiempo = 0;                  //Variable que define el tiempo actual START
unsigned long Ttiempo = 0;                  //Variable que define el tiempo total START
unsigned long P1tiempo = 0;                 //Variable que define el tiempo anterior SALIR
unsigned long C1tiempo = 0;                 //Variable que define el tiempo actual SALIR
unsigned long T1tiempo = 0;                 //Variable que define el tiempo total SALIR
int retardoJuego = 2;                       //Delay del juego
bool banderita;                             //bandera que indica el estado: mensaje o juego
int velPelotita = 5;                        //Velocidad de la pelota
int dirPelota = velPelotita;                //Variable que mueve la pelota
int padIzq = 0;                             //Variable que define el pad izquierdo
int padDer = 0;                             //Variable que define el pad derecha
int x = 4;                                  //Posicion de pelota en x
int y = 1;                                  //Posicion de pelota en y
int dirx = 0;                               //Direccion en x
int diry = 0;                               //Direccion en y
int puntosIzq = 0;                          //Marcador izquierda
int puntosDer = 0;                          //Marcador derecha
bool banda = 0;                             //Bandera de pausa
//FIN VARIABLES DE JUEGO
```

Aquí se declaran las variables utilizadas en el juego, como también la declaración de los pines que se utilizan, el objeto de que se declaro para el MAX7219, las variables utilizadas con el tiempo relacionadas con la función millis().

```
//Sirve para verificar el tiempo de presionado del botón para acceder al juego o no.
void verificarBotonStart() {
  if (digitalRead(7) == HIGH) {
    Ctiempo = millis();
    while (digitalRead(7) == HIGH) {
      Ptiempo = millis();
      digitalRead(7);
    }
    Ttiempo = Ptiempo - Ctiempo;
    if (Ttiempo >= 3000) {          //Si es mayor a 3 segundos entra al juego
      banderita = 1;
      initGame();
    }else{                          //Sino entra al mensaje
      Pantalla();
    }
  }
}
```

Con la utilización de la función millis() se hace el conteo de tiempo, desde el momento que se presiona el boton de inicio hasta el momento que se suelta, y se resta ambos tiempos para obtener el tiempo que se mantuvo presionado, si es mayor entra al juego y sino al mensaje

```
//Sirve para verificar el tiempo de presionado del botón para acceder a la pausa.
void verificarBotonStart2() {     //verifica si son 3 sec para salir
```

```
  if (digitalRead(7) == HIGH) {
    C1tiempo = millis();
    while (digitalRead(7) == HIGH) {
      P1tiempo = millis();
      digitalRead(7);
    }
    T1tiempo = P1tiempo - C1tiempo;
    if (T1tiempo >= 300) {          //si es mayor a 3 sale
      banderita = 0;
    } else {                        //sino solo quita la pausa
      banderita = 1;
    }
  }
}
```

Con la utilización de la función millis() se hace el conteo de tiempo, desde el momento que se presiona el boton de inicio hasta el momento que se suelta, y se resta ambos tiempos para obtener el tiempo que se mantuvo presionado, si es mayor entra a la pausa y sino solo vuelve al juego

```
//Controlador del juego e indicador decisiones del pong
void jueguito() {
  //primero limpiamos matriz
  puntosDer = puntosIzq = 0;
  borrador();
  Imprimir();
  while (banderita) {
    if (digitalRead(7) == HIGH && banda == 0) {
      banda = 1;
      delay(2);
    }else if (banda && digitalRead(7) == HIGH){
      banda = 0;
      delay(2);
    }
    verificarBotonStart2();                 //verifica si en algun momento se ha pulsado 3 sec
    if (puntosDer == 4 || puntosIzq == 4){
      mostrarPuntosTablero();
      delay(300);
      banderita = 0;
      break;
    }else if (banda){
      mostrarPuntosTableroSinDel();
    }else{
      repetirJuego();
    }
  }
}
```

En este método se imprime la matriz, primero se limpia, los puntos de ambos jugadores se inicializan en cero y se imprime el mensaje, posterior se valida si se ha presionado o no. Si este ha sido presionado se hace la verificación estando en el mensaje y después de verifica si se ha presionado durante el juego.

```
//Metodo que tienen las instrucciones de direccionamiento para el juego
void repetirJuego() {
  borrador();
  if (dirPelota == 0) {
    pelotita();
    dirPelota = velPelotita;
  } else {
    --dirPelota;
  }
  movJugadorIzq();
  movJugadorDer();
  tablero[padIzq][0] = tablero[padIzq + 1][0] = tablero[padIzq + 2][0] = 1;
  tablero[padDer][15] = tablero[padDer + 1][15] = tablero[padDer + 2][15] = 1;
  tablero[y][x] = 1;
  Imprimir();
  delay(retardoJuego);
}
```

En este método hace el direccionamiento de la pelota, se llaman los métodos de movimiento de ambos jugadores, y se validan los limites de la matriz para que la pelota no se salga o se vaya de largo.

```
//Metodo que cambio los bytes de las letras para mostrarlas
void Pivote() {
  if (++contFila == 8) {
    Cont = ++Cont % num;
    contFila = 0;
  }
}
```

Método, donde se hace un recorrido de los bytes existentes declarados en el mensaje para que se cambien en el mensaje según el mod de la cantidad de bytes.

```
//Metodo que maneja la velocidad y el direccionamiento del mensaje
void Pantalla() {
  int aumentaVelocidad = digitalRead(8);
  int disminuyeVelocidad = digitalRead(9);
  if (aumentaVelocidad)sumador = sumador + 2;
  if (disminuyeVelocidad)sumador = sumador - 2;
  switch (sumador) {
    case 0:
      rapidez = 0;
      break;
    case 2:
      rapidez = 2;
```

```
    break;
   case 4:
    rapidez = 4;
    break;
   case 6:
    rapidez = 6;
    break;
   case 8:
    rapidez = 8;
    break;
  }

  if (retraso >= rapidez) {
   Mensaje(Cont);  //Mensaje donde se manda como parametro el contador para hacer el cambio de bytes
  }
  else {
   retraso++;
  }
  Imprimir();
  int bandera = digitalRead(10);
  if (bandera) {
   if (derizq == 1) derizq = 0;
   else derizq = 1;
  }
}
```

Sección de código que controla la velocidad, según sea el caso. Si se presiona el boton de aumentar velocidad aumenta el contador a 6 y 8 y si se presiona el de disminuir baja a 2 y 0.

```
//metodo que muestra y desplaza el mensaje
void Mensaje(int num1)  { //el parametro num1 es el contador que recibe mas arriba y num1 es el primer corchete del letrero definido mas arriba que es numero de byte
  for (int fila = 0; fila < 8; fila++) {
   int limite = 0;
   for (int columna = 0; columna < 16; columna++) {
    if (columna + corrimiento < 16 ) {
     tablero[fila][columna] = letrero[num1][fila][columna + corrimiento];
    }else{
     tablero[fila][columna] = letrero[num1][fila][limite++];
    }
   }
  }

  if (derizq) {
   if (corrimiento < 16) corrimiento++;
   else corrimiento = 0;
  }else{
```

```
   if (corrimiento > 0) corrimiento--;
   else corrimiento = 16;
  }
  retraso = 0;
}
```

Método en donde se hace el corrimiento del mensaje, donde se valida si es menor a 16. Si lo es se le suma a la columna el corrimiento. Sino se le va sumando el limite.

```
//Metodo que muestra la matriz
void Imprimir() {
  Pivote();

  for (int x = 0; x < 8; x++) {
    for (int y = 8; y < 16; y++) {

      if (tablero[x][y] == 1)
      {
        inputEntrada.setLed(0, x, y - 8, true);
      }
      else
      {
        inputEntrada.setLed(0, x, y - 8, false);
      }
    }
  }
```

Sección de código que controla la velocidad, según sea el caso. Si se presiona el boton de aumentar velocidad aumenta el contador a 6 y 8 y si se presiona el de disminuir baja a 2 y 0.

```
//Metodo que muestra la matriz
void Imprimir() {
  Pivote();

  for (int x = 0; x < 8; x++) {
    for (int y = 8; y < 16; y++) {
      if (tablero[x][y] == 1){
        inputEntrada.setLed(0, x, y - 8, true);
      }else{
        inputEntrada.setLed(0, x, y - 8, false);
      }
    }
  }
  for (int x = 0; x < 8; x++) {
    for (int y = 0; y < 8; y++) {
      int par = y * 2;
      if (tablero[x][y] == 1)  digitalWrite(38 + par , HIGH);
      else digitalWrite(38 + par, LOW);
```

```
    }
    int par2 = x * 2;
    digitalWrite(22 + par2, LOW);
    delay(2);
    digitalWrite(22 + par2, HIGH);
  }

  inputEntrada.clearDisplay(0);
  for (int serial = 38; serial < 53; serial = serial + 2){
    digitalWrite(serial, LOW);
  }
}


//Metodo para el movimiento de jugador izquierdo
void movJugadorIzq() {
  int lector = digitalRead(pinIzqUp);
  if (lector) {
    if (padIzq != 5) {

      padIzq++;
    }
  }
  int lector1 = digitalRead(pinIzqDown);
  if (lector1) {
    if (padIzq != 0) {
      padIzq--;
    }
  }
}

//Metodo para el movimiento de jugador derecho
void movJugadorDer() {
  int lector = digitalRead(pinDerUp);
  if (lector) {
    if (padDer != 5) {
      padDer++;
    }
  }
  int lector1 = digitalRead(pinDerDown);
  if (lector1) {
    if (padDer != 0) {
      padDer--;
    }
  }
}

//Metodo que sirve para limpiar la matriz
void borrador() {
```

```
  for (int i = 0; i < 8; ++i)
  {
    for (int j = 0; j < 16; ++j)
    {
      tablero[i][j] = 0;
    }
  }
}

//Metodo para cambiar la direccion de la pelota
void pelotita() {
  if (dirx == 0)
  {
    x++;
  } else {
    x--;
  }

  if (diry == 0)
  {
    y++;
  } else {
    y--;
  }
  verificador();
  metaDerecha();
  metaIzquierda();
}

//Metodo que verifica los limites de la matriz
void verificador() {
  if (y == 7) {
    diry = 1;
  }
  if (x == 15) {
    dirx = 1;
  }
  if (y == 0) {
    diry = 0;
  }
  if (x == 0) {
    dirx = 0;
  }
}

//Puntajes derecho, izquierdo y mostrar tablero en pantalla
void metaDerecha() {
  if (x == 15 && !(y == padDer || y == padDer + 1 || y == padDer + 2)) {
    ++puntosIzq;
```

```
    mostrarPuntosTablero();
  }
}

void metaIzquierda() {
  if (x == 0 && !(y == padIzq || y == padIzq + 1 || y == padDer + 2)) {
    ++puntosDer;
    mostrarPuntosTablero();
  }
}

void mostrarPuntosTablero() {
  borrador();
  for (int i = 0; i < 8; ++i){
    for (int j = 0; j < 8; ++j){
      tablero[i][j] = pnts[puntosIzq][i][j];
    }
  }

  for (int i = 0; i < 8; ++i){
    for (int j = 8; j < 16; ++j){
      tablero[i][j] = pnts[puntosDer][i][j - 8];
    }
  }

  int temporal = 25;
  while (true) {
    Imprimir();
    --temporal;
    if (temporal == 0) {
      break;
    }
  }
}
void mostrarPuntosTableroSinDel() {
  borrador();
  for (int i = 0; i < 8; ++i){
    for (int j = 0; j < 8; ++j){
      tablero[i][j] = pnts[puntosIzq][i][j];
    }
  }
  for (int i = 0; i < 8; ++i){
    for (int j = 8; j < 16; ++j){
      tablero[i][j] = pnts[puntosDer][i][j - 8];
    }
  }
  Imprimir();
}
```