Universidad de San Carlos de Guatemala
Arquitectura de computadores y ensambladores 1
Escuela de Sistemas
Ing. Otto Rene Escobar Leiva
Aux Mario Pineda

Manual Técnico:

PRACTICA 2:

Nombre: **Carnet**:
Osmar Abdel Peña Santizo 201801619
Brandon Oswaldo Yax Campos 201800534
Gerson Sebastian Quintana Berganza 201908686
Wilson Kevin Javier Chávez Cabrera 201807428
John Henry López Mijangos 201710392

Guatemala, 24 de febrero del 2022

INTRODUCCIÓN

La comunicación I2C es una tecnología que nos permite comprender más a fondo la comunicación serial, comportándose esta con un mecanismo parecido a como una página web funciona con el Frontend y el Backend permitiendo dividir los trabajos en dos partes siendo estas: El arduino maestro y los arduinos esclavos, dando así un mayor alcance para la manipulación de bits e información de entrada y salida, En esta práctica dicha tecnología nos permite entender el alcance que un integrado arduino puede tener conjunto a otros unidos en un mismo sistema dividiendo la dificultad. La práctica consistió principalmente en un motor con un integrado ultrasónico pegado a este, mostrándose un mensaje en un lcd al inicio para posteriormente detectar obstáculos y disparar, si asi se desea, a estos gracias a el ultrasonido, dicho componente también nos permitió explorar la tecnología sensorial a través de la ecolocalización, detectando objetos a una distancia cercana permitiendo la detección de enemigos, el disparo de estos para posteriormente obtener los datos de estos para la generación de la media, mediana y moda.

Objetivos

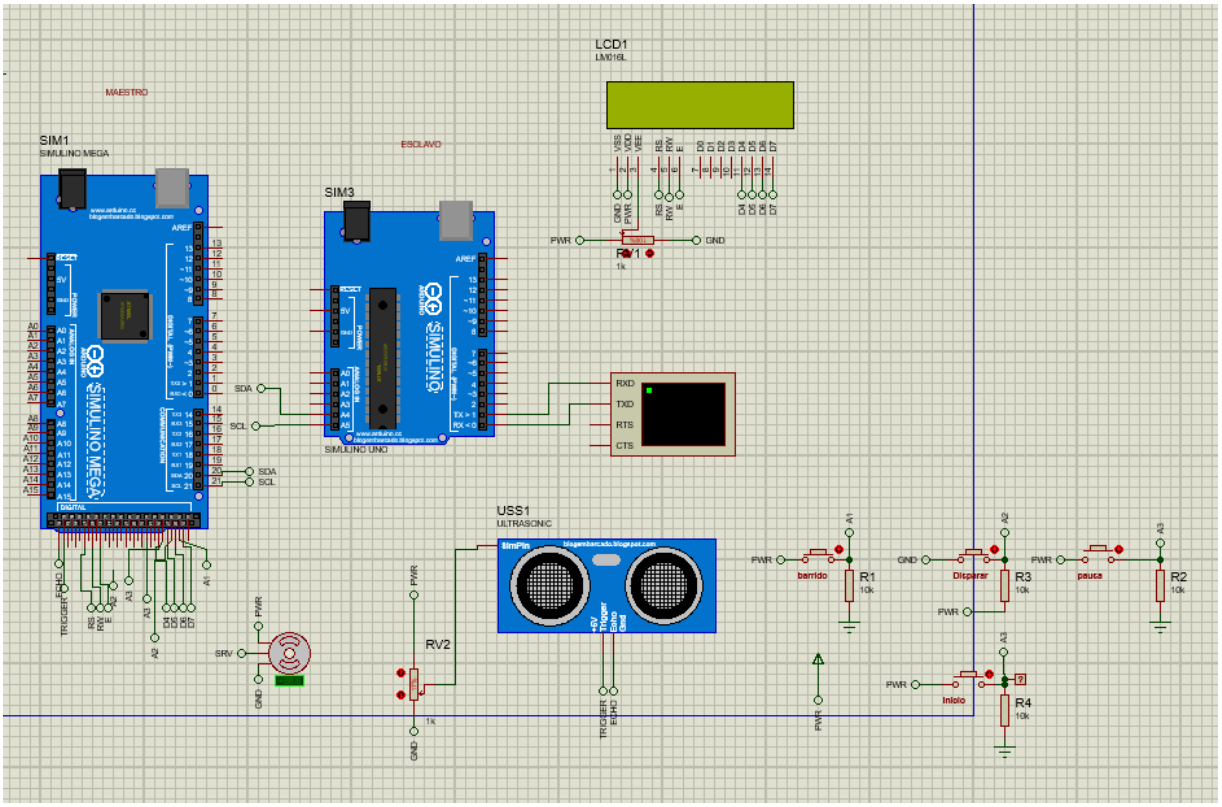
Objetivo General:

Que el estudiante adquiera, aplique e interactúe con el microcontrolador Arduino de forma física.

Objetivos Específicos:

- Aplicar el conocimiento adquirido en simulación.
- Aprender a comunicar arduinos por medio de I²C - Maestro - Esclavo
- Utilizar sensores físicos para la resolución de problemas.
- Aplicar el lenguaje C para estructuras de control en Arduino.

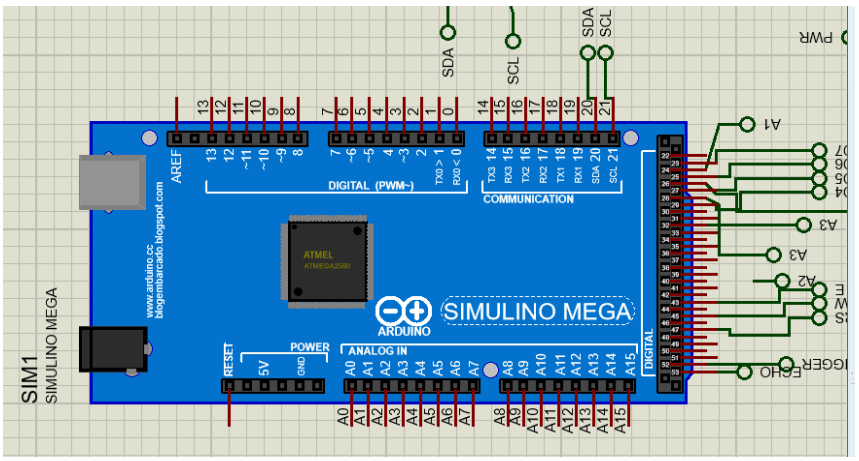
Diagrama del Circuito



Código de Arduino Para la configuración de la práctica se realizaron 2 programas en Arduino IDE, ambos utilizando la configuración del Simulino Mega en Proteus. Cada programa corresponde a 1 Simulino Mega (1 para el maestro y 1 para el esclavo). El código de ambos se encuentra en anexos.

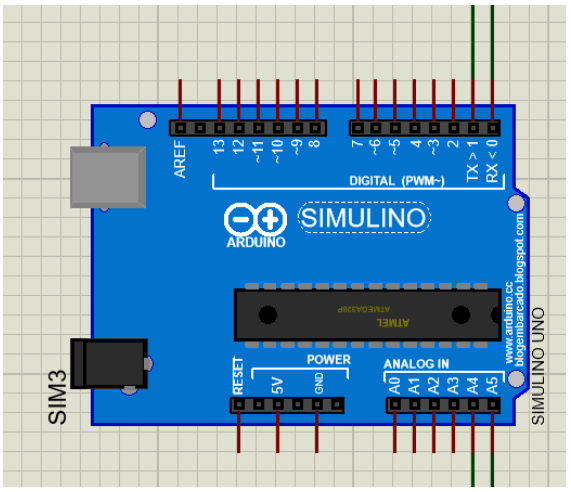
Arduino Mestro:

Encargado de analizar y dirigir las instrucciones que envía el usuario, recibe los pulsos en variables que se conectan y validan, recibe información del arduino esclavo para que se accione y valla a los métodos de ordenamiento de datos, métodos estadísticas y estados que activan ciertas funciones.



Arduino Esclavo:

Arduino encargado de recibir instrucciones que se generan desde el arduino maestro, para asi mover el motor, el sensor ultrasonido y lacer, que recolectan los datos y los envia al arduino maestro para su debida validación.



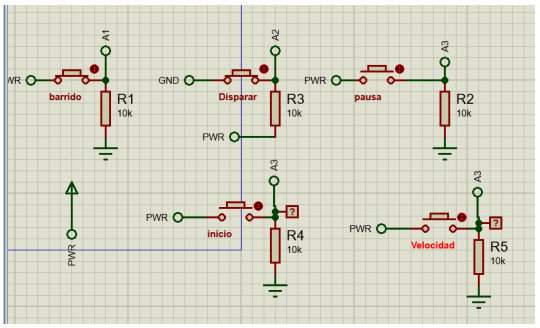
Pantalla LCD:

Encargada de mostrar el contenido visual que el usuario ve, por medio de una pantalla conformada de 16 columnas por 2 filas.

Explicación de Código Arduino

Botones:

Los botones son los que mandan un pulso a 1 por medio del arduino maestro, válida y se comunica por medio de i2c con el esclavo para transmitir el dato y entrar y activar si es escaneo, si es disparo o si es inicio.




```
int btn_barrido = 24;
int btn_ataque = 26;
int btn_pausa = 28;
bool btn_barrido_pres, btn_ataque_pres, btn_pausa_pres;
```

Por medio de la entrada que le llega al esclavo se valida si es 1, 2 o 3 y activa los estados que se validan para la opciones de barrido, disparar.

```
int x = Wire.read();     // re
if(x==0){
  ESTADO = 0;
}
else if (x == 1 ) {
  ESTADO = 1;
  if(contador >180){
    contador =0;
  }

} else if (x == 2) {
  ESTADO = 2;
}
else if (x == 3) {
  char ok = Wire.read();
```

mensaje inicial:

En este lugar se imprima como inicio el mensaje inicial correspondiente a: "😊GRUPO ## ACE1✔", esperando 10 segundos, para cambiar de opcion, se definen por medio de matrices de byte y la librería LiquidCrystal.

```
void mensajeInicial() {
  byte cara[] = {
    B00000,
    B00000,
    B01010,
    B10101,
    B00000,
    B10001,
    B01110,
    B00000
  };
  byte cheque[] = {
    B00000,
    B00000,
    B00000,
    B00001,
    B10010,
    B01100,
    B01000,
    B00000
  };
  lcd.createChar(0, cara);
  lcd.setCursor(0, 0);
  lcd.write(byte(0));
  lcd.setCursor(1, 0);
  lcd.print("GRUPO 04 ACE");
  lcd.createChar(1, cheque);
  lcd.setCursor(13, 0);
  lcd.write(byte(1));
}
```

Transmisión de maestro a esclavo :

Por medio de la librería Wire, transmite la información del arduino maestro al esclavo. I2C.

```
Wire.begin();
Wire.beginTransmission(2);
Wire.write(0);
Wire.endTransmission();
Wire.requestFrom(2, 1);
```

Transmisión de esclavo a maestro :

Por medio de la librería Wire, transmite la información del arduino esclavo al maestro. I2C.

```
char c = Wire.read();
if ((int)c == 1) {
  mensajeInicial();
  tiempo_msj = millis();
}
```

Va al método **datos** que lleva y valida los datos que se leen desde la comunicación i2c.

```
void datos() {
  Serial.println("datosw");
  if(ESTADO == 0){
    Wire.write(1);
  }else if(ESTADO == 1){
    barrido();
  }else if(ESTADO ==4){
    Serial.println("datos");
    Disparar();
  }
```

Método ubicado en el loop del maestro, del cual se está a la espera de recibir instrucciones, por medio de los botones que se presionen desde el circuito físico.

```
while (true) {
  // Instruccion barrido
  if (!estaPresionado(btn_barrido)) {

    btn_barrido_pres = true;
  }
  if (estaPresionado(btn_barrido) && btn_barrido_pres) {
    recibirDistancias();

    btn_barrido_pres = false;
  }
  // Instruccion ataque
  if (!estaPresionado(btn_ataque)) {
    btn_ataque_pres = true;
  }
  if (estaPresionado(btn_ataque) && btn_ataque_pres) {
    Serial.println("PRESIONASTE");
    Wire.beginTransmission(2);
    Wire.write(4);
    Wire.endTransmission();
    btn_ataque_pres = false;
```

Recibir distancias:

Tal y como indica el nombre, la función recibirDistancias() lo que hace es recibir cada una de las distancias que recibe del arduino esclavo. Esta función se ejecuta tantas veces dependiendo cuantos enemigos detecte; las almacena en dos listas (esto porque se escanean los enemigos dos veces). Posteriormente compara cada una de las posiciones del arreglo para asegurar que las dos listas sean iguales, y si lo son, se ordenan una de las dos listas. Además de lo anterior, esta función también recibe el grado en el que está cada uno de los enemigos y lo almacena en una lista llamada *grados.*

```
        Serial.println((int)c);
        distancia[contadorlista] = (int)c;

        if ((int)c < 25) {
          grados[contadorlista] = contador;
        }

        contadorlista++;
        delay(2000);
        lcd.clear();

        contador += 18;
      }


  Wire.beginTransmission(2);
  Wire.write(1);
  Wire.endTransmission();
  Wire.requestFrom(2, 1);

  char c = Wire.read();
  if ((int)c < 25 ) {

    lcd.clear();

    lcd.setCursor(0, 0);
    lcd.print("Detectado");
    lcd.setCursor(0, 1);
    lcd.print("D: " + String((int)c) + " A: " + contador);

  }
  Serial.println((int)c);
  distancia[contadorlista] = (int)c;
  if ((int)c < 25) {
    grados[contadorlista] = contador;
  }

  contadorlista++;
  delay(2000);
  lcd.clear();

  contador += 18;
}
```

```
      Wire.beginTransmission(2);
      Wire.write(1);
      Wire.endTransmission();
      Wire.requestFrom(2, 1);

      char c = Wire.read();
      if ((int)c < 25 ) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Detectado");
        lcd.setCursor(0, 1);
        lcd.print("D: " + String((int)c) + " A: " + contador);

      }
      Serial.println((int)c);
      distancia2[contadorlista] = (int)c;
      contadorlista++;
      delay(2000);
      lcd.clear();
      contador += 18;
    }
    if (compararPosiciones()) {
      Serial.println("Son iguales");
      ordenarLista();
    } else {
      Serial.println("No son iguales");
    }
```

Enviar Grados:

Inicia la transmisión con el arduino esclavo, luego envía un caracter (‘o’) para indicar que los enemigos se encuentran en la misma distancia en los dos barridos. Si son las mismas distancias iguales, empieza ahora sí la transmisión de todos grados. Estos grados no están ordenados de menor a mayor, si no que, en el momento que se ordenan las distancias de menor a mayor, también se cambia la correspondiente posición en el arreglo de distancias.

```
// Envia cada una de las posiciones del arreglo al esclavo
void enviarGrados() {
  int enemigosvivos=0;
  Wire.beginTransmission(2); // inciando la transmision de datos
  Wire.write(3);
  Wire.write('o'); // o -> ok, las lista está correcta
  for (int i = 0; i < 10; i++) {
    if (grados[i] != -1) {
      Wire.write(grados[i]);  // Enviando el valor de la posicion
      enemigosvivos++;
    }
  }
  Wire.write(200); // terminando la transmision de datos
  Wire.endTransmission();
}
```

Método de moda :

Este método recibe como parámetro una lista, que pasa por los procesos de leerla y guardar las repeticiones que cada elemento se repite, después lee la matriz auxiliar que tiene los elementos guardados de la repetición de cada elemento de la matriz original. Por último se busca la posición dentro de los arreglos para encontrar la posición de la moda.

```
void moda(int ma [] ){
 int longitud=10;
 int auxiliar[longitud];
   // int ma[10] = {1,2,3,2,20,7,90,83,1,0};
    for(int i=0; i <longitud;i++) {
        auxiliar[i]=0;
    }
    int posicion;
    int numero;
    //leo las veces que se repite cada numero.
    for(int i=0;i<longitud;i++) {
        numero = ma[i];
        posicion = i;
        for(int y=i;y<longitud;y++) {
            if(ma[y]==numero) auxiliar[posicion]++;
        }
    }
    int mayor=auxiliar[0];
    int posicionmayor = 0;
    for(int i=0;i<longitud;i++) {
        if(auxiliar[i]>mayor) {
```

Método de media :

Este método recibe como parámetro una lista, que pasa por el proceso de leerla y sumar todos sus elementos para luego dividirla dentro del tamaño del arreglo y asi retornar el valor de la media.

```
void media(int ma []){
     int suma = 0;
      int longitud=10;
    for(int i=0;i<longitud;i++) {
        suma+=ma[i];
    }
    float resultado=suma/longitud;
    lcd.setCursor(0, 1);
    lcd.print("X: " +String(resultado));
    //print.serial(resultado);

}
```

Método de mediana :

Este método recibe como parámetro una lista, que pasa por el proceso de leerla por medio de un ordenamiento que ordena los datos, para después encontrar la posición central o posiciones centrales del arreglo para así imprimirlas.

```
void mediana(int ma []){
int longitud=enemigoseliminados;
 int bandera=0;
 int numero=0;
    for(int i=longitud; i>0 && bandera==0;i--) {
        bandera=1;
        for(int y=0;y<i;y++) {
            if(ma[y]>ma[y+1]) {
                numero = ma[y];
               ma[y] = ma[y+1];
                ma[y+1]=numero;
                bandera=0;
            } }}
     if(longitud%2!=0) {
        lcd.setCursor(6, 0);
        lcd.print("Me: " +String(ma[longitud/2]));
    } else {
        lcd.setCursor(6, 0);
        lcd.print("Med: " +String((ma[longitud/2] + ma[(longitud/2)-1])/2));
    }
```

Mensaje de objetivos eliminados y vivos:

Este método manda una petición del maestro al esclavo por medio de IO2, por el cual se pide el dato de objetivos eliminados y los objetivos vivos, y se valida con una sentencia para validar e imprimir los resultados en la pantalla lcd.

```
long unsigned tiempoataque = millis();
enviarGrados();
delay(3000);
Wire.beginTransmission(2);
Wire.write(4);
Wire.endTransmission();
enemigoseliminados =0;
while (enemigosvivos > 0) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Vivos: " + String(enemigosvivos));
  lcd.setCursor(0, 1);
  lcd.print("Muertos: " + String(enemigoseliminados));
  delay(2000);
  enemigosvivos--;
  enemigoseliminados++;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Vivos: " + String(enemigosvivos));
  lcd.setCursor(0, 1);
  lcd.print("Muertos: " + String(enemigoseliminados));
```

Setup esclavo:

Parte que instancia la conexión i2c, peticiones que hace el maestro para detectar en el momento que se soliciten. Además instancia de pines de motor, láser, y servomotor.

```
void setup()
{
  Wire.begin(2);                // Este Esclavo es el número 2

  Wire.onReceive(requestEvent);
  Wire.onRequest(datos); // Cuando el Maestro le hace una petición,
  pinMode(TRIG, OUTPUT);
  pinMode(ECO, INPUT);                              // realiza el requestEvent
  Serial.begin(9600);
  servomec.attach(8);//Pin
  pinMode(LASER,OUTPUT);
  digitalWrite(LASER,LOW);
}
```

Escaneo de enemigos:

Esta función se ejecuta cuando se están detectando enemigos. Los que hace es mover el motor servo 18 grados, obtiene la duración del “rebote” que tarda de ir al enemigo y regresar, y con estos datos se ingresan a la fórmula, y se envían al arduino maestro

```
servomec.write(contador);

digitalWrite(TRIG, LOW);
delayMicroseconds(2);
digitalWrite(TRIG, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG, LOW);
DURACION = pulseIn(ECO, HIGH);

DISTANCIA = (DURACION * 0.034) / 2;

DISTANCIAC = (char)DISTANCIA;

Wire.write(DISTANCIAC);

delay(1000);
```

Disparar:

Esta función mueve el motor una cantidad de grados que indicará la lista *grados*, la cual contiene los grados que debe moverse para matar al enemigo más cercano. Luego, lo que hace es encender el láser, mantenerlo 1 segundo así y luego apagar el láser.

```
void Disparar() {
  for (int i = 0; i < 10 ; i++) {
    servomec.write(grados[i]);
    delay(1000);
    digitalWrite(LASER, HIGH);
    delay(1000);
    digitalWrite(LASER, LOW);
    delay(2000);
  }
  ESTADO = 0;
}
```

SolicitarEvento:

Cuando el arduino maestro envía datos al arduino esclavo se ejecuta esta función, la cual lo que hace es cambiar el estado del programa dependiendo del valor transmitido.

Cuando se transmite un 0, el programa pasa a un estado 0, el cual envía un 1 al arduino maestro para que muestre el mensaje inicial; si se transmite un 1, el programa pasa a un estado 1, que lo que hace es empezar con el barrido, es decir, a detectar enemigos y le va enviando cada una de las distancias el arduino maestro. En el estado 3, lo que hace es recibir las distancias ordenadas en el arduino maestro; también recibe los grados correspondientes a cada una de las distancias recibidas.

```
// Esto es lo que envía cuando le hace la petición.
void requestEvent()
{
  int x = Wire.read();
  if (x == 0) {
    ESTADO = 0;
  }
  else if (x == 1 ) {
    ESTADO = 1;
    if (contador > 180) {
      contador = 0;
    }
  } else if (x == 2) {
    ESTADO = 2;
  }
  else if (x == 3) {
    leerAngulos();
  }
  else if (x == 4) {
    ESTADO = 4;
  }
}
```

EQUIPO UTILIZADO

- 2 dispositivos arduino (uno Mega y el otro UNO).
- 1 sensor ultrasónico HC-SR04
- Laser rojo
- Botones Pantalla LCD 16x2
- Servo Motor de 180°
- Botones

CONCLUSIONES

1. La comunicación serial nos permite el ensanchamiento de la capacidad de un sistema para poder abarcar más tareas, usando aquí la comunicación I2C para este procedimiento.

2. El I2C es un protocolo de suma importancia para la comunicación del chip arduino en unión con otros, usando la librería Wire para usar dicho protocolo.

3. Los chips arduino en conjunto permiten la simulación de un procesador más o menos complejo dependiendo la cantidad de arduinos unidos en un mismo sistema.

4. Se debe de tener sumo cuidado si se usa la comunicación I2C en arduinos de tipos distintos, teniendo énfasis en colocar en el pin correcto los enlaces de un arduino a otro.

APÉNDICE

Código Arduino Maestro

```
#include <LiquidCrystal.h>
#include <Wire.h>

// LCD
int rs = 47, rw = 45, en = 43, d4 = 29, d5 = 27, d6 = 25, d7 = 23;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7); //Rs, en ,D3,D2,D1,D0
// Mensaje
long unsigned tiempo_msj;
// Variables
int contador_loop = 0;
int btn_barrido = 24;
int btn_ataque = 26;
int btn_pausa = 28;
int btn_reset = 30;
bool btn_barrido_pres, btn_ataque_pres, btn_pausa_pres, btn_reset_pres;


int distancia[10];
int distancia2[10];
int grados[10];
bool iguales = true;
int nPos = 0;
int enemigosvivos = 0;
int enemigoseliminados = 0;
bool estaenpausa = false;
long unsigned tiempopausa ;
```

```
void setup() {
  Serial.begin(9600);
  // LCD
  for (int i = 23; i < 30; i++) pinMode(i, OUTPUT);
  for (int i = 45; i < 48; i++) pinMode(i, OUTPUT);
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0, 0);
  // I2C - Mensaje Inicial (respuesta esclavo)
  Wire.begin();
  Wire.beginTransmission(2);
  Wire.write(0);
  Wire.endTransmission();
  Wire.requestFrom(2, 1);
  char c = Wire.read();
  if ((int)c == 1) {
    mensajeInicial();
    tiempo_msj = millis();
  }

  while (true) {
    if ((millis() - tiempo_msj) > 10000) {

      break;
    }
  }
}
```

```
void loop() {
  mensajeReinicioLoop();
  resetearParametros();
  esperarInstrucciones();
}

void mensajeInicial() {
  byte cara[] = {
    B00000,
    B00000,
    B01010,
    B10101,
    B00000,
    B10001,
    B01110,
    B00000
  };
  byte cheque[] = {
    B00000,
    B00000,
    B00000,
    B00001,
    B10010,
    B01100,
    B01000,
    B00000
  };
  lcd.createChar(0, cara);
  lcd.setCursor(0, 0);
  lcd.write(byte(0));
  lcd.setCursor(1, 0);
```

```
  lcd.print("GRUPO 04 ACE");
  lcd.createChar(1, cheque);
  lcd.setCursor(13, 0);
  lcd.write(byte(1));
}

void mensajeReinicioLoop() {
  lcd.clear();
  contador_loop++;
  lcd.setCursor(0, 0);
  lcd.print("Entrando al loop");
  lcd.setCursor(0, 1);
  lcd.print("Contador: ");
  lcd.print(contador_loop);
  delay(3000);

}

void resetearParametros() {
  // LCD
  lcd.clear();
  lcd.setCursor(0, 0);


  //Reseteo de listas
  for (int i = 0; i < 10; i++) {
    distancia[i] = 0;
    distancia2[i] = 0;
    grados[i] = -1;
  }
  iguales = true;
```

```
  enemigosvivos = 0;
  enemigoseliminados = 0;
}

void esperarInstrucciones() {
  while (true) {
    // Instruccion barrido
    if (!estaPresionado(btn_barrido)) {

      btn_barrido_pres = true;
    }
    if (estaPresionado(btn_barrido) && btn_barrido_pres) {
      recibirDistancias();

      btn_barrido_pres = false;
    }
    // Instruccion ataque
    if (!estaPresionado(btn_ataque)) {
      btn_ataque_pres = true;
    }
    if (estaPresionado(btn_ataque) && btn_ataque_pres) {
      long unsigned tiempoataque = millis();
      enviarGrados();
      delay(3000);
      Wire.beginTransmission(2);
      Wire.write(4);
      Wire.endTransmission();
      enemigoseliminados = 0;
      while (enemigosvivos > 0) {
        lcd.clear();
        lcd.setCursor(0, 0);
```

```
  lcd.print("Vivos: " + (String)enemigosvivos);
  lcd.setCursor(0, 1);
  lcd.print("Muertos: " + String(enemigoseliminados));
  delay(2000);
  enemigosvivos--;
  enemigoseliminados++;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Vivos: " + (String)enemigosvivos);
  lcd.setCursor(0, 1);
  lcd.print("Muertos: " + String(enemigoseliminados));
  delay(2000);

}
tiempoataque = millis() - tiempoataque;
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("T: " + String(tiempoataque / 1000));
for (int i = 0; i < 10; i++) {
  Serial.println(distancia[i]);

  Serial.println(distancia2[i]);
  Serial.println("----");
}
int prueba[10] = {22, 22, 22, 22, 22, 22, 22, 22, 22, 22};
mediana(distancia);
moda(distancia);
media(distancia);



delay(5000);
```

```
      btn_ataque_pres = false;
    }
    // Instruccion reset
    if (!estaPresionado(btn_reset)) {
      btn_reset_pres = true;
    }
    if (estaPresionado(btn_reset) && btn_reset_pres) {

      btn_reset_pres = false;
      break;
    }

    // Si no tiene instrucciones
    if (!estaPresionado(btn_barrido) && !estaPresionado(btn_ataque) &&
!estaPresionado(btn_reset) && !estaPresionado(btn_pausa) ) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Esperando");
      lcd.setCursor(0, 1);
      lcd.print("instruccion...");
    }
    delay(1000);
  }
}

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

```
void recibirDistancias() {
  int contador = 0;
  int contadorlista = 0;

  while (contador <= 180) {
    // Instruccion pausa
    if (!estaPresionado(btn_pausa)) {
      btn_pausa_pres = true;
    }
    if (estaPresionado(btn_pausa) && btn_pausa_pres) {
      estaenpausa = ! estaenpausa;
      btn_pausa_pres = false;
      tiempopausa = millis();
    }

    lcd.clear();
    while (estaenpausa) {
      lcd.setCursor(0, 0);
      lcd.write("En pausa");
      if ((millis() - tiempopausa) >= 5000) {

        estaenpausa = false;
        break;
      }
    }
    lcd.clear();

    Wire.beginTransmission(2);
    Wire.write(1);
    Wire.endTransmission();
    Wire.requestFrom(2, 1);
```

```
  char c = Wire.read();
  if ((int)c < 25 ) {

    lcd.clear();

    lcd.setCursor(0, 0);
    lcd.print("Detectado");
    lcd.setCursor(0, 1);
    lcd.print("D: " + String((int)c) + " A: " + contador);

  }
  Serial.println((int)c);
  distancia[contadorlista] = (int)c;
  if ((int)c < 25) {
    grados[contadorlista] = contador;
  }

  contadorlista++;
  delay(2000);
  lcd.clear();

  contador += 18;


}
contador = 0;
contadorlista = 0;
while (contador <= 180) {
  // Instruccion pausa
  if (!estaPresionado(btn_pausa)) {
```

```
  btn_pausa_pres = true;
}
if (estaPresionado(btn_pausa) && btn_pausa_pres) {
  estaenpausa = ! estaenpausa;
  btn_pausa_pres = false;
  tiempopausa = millis();
}



lcd.clear();
while (estaenpausa) {
  lcd.setCursor(0, 0);
  lcd.write("En pausa");
  if ((millis() - tiempopausa) >= 5000) {
    estaenpausa = false;
    break;
  }
}
lcd.clear();
Wire.beginTransmission(2);
Wire.write(1);
Wire.endTransmission();
Wire.requestFrom(2, 1);

char c = Wire.read();
if ((int)c < 25 ) {

  lcd.clear();

  lcd.setCursor(0, 0);
```

```
      lcd.print("Detectado");
      lcd.setCursor(0, 1);
      lcd.print("D: " + String((int)c) + " A: " + contador);

    }

    Serial.println((int)c);
    distancia2[contadorlista] = (int)c;
    contadorlista++;
    delay(2000);
    lcd.clear();
    contador += 18;

  }

  if (compararPosiciones()) {
    Serial.println("Son iguales");
    ordenarLista();
  } else {
    Serial.println("No son iguales");
  }


}


// ---------------------------------- INICIO AGREGADO
-----------------------------------------
// Comparando si cada una de las posiciones son iguales
bool compararPosiciones() {
  for (int i = 0; i < 10; i++) {
```

```
    int resultado = abs(distancia2 [i ] - distancia [ i]);
    if (resultado > 3) {
      return false;
    }
  }
  return true;
}

// Metodo de Burbuja para ordenar el arreglo
void ordenarLista() {
  int temp = 0;
  int temp2 = 0;
  for (int i = 1; i < 10; i++) {
    for (int j = 10 - 1; j >= i; j--) {
      if (distancia[j - 1] > distancia[j]) {
        temp = distancia[j - 1];
        distancia[j - 1] = distancia[j];
        distancia[j] = temp;
        temp2 = grados[j - 1];
        grados[j - 1] = grados[j];
        grados[j] = temp2;
      }
    }
  }
}


// Envia cada una de las posiciones del arreglo al esclavo
void enviarGrados() {
  enemigosvivos = 0;
  Wire.beginTransmission(2); // inciando la transmision de datos
```

```
  Wire.write(3);
  Wire.write('o'); // o -> ok, las lista está correcta
  for (int i = 0; i < 10; i++) {
    if (grados[i] != -1) {
      Wire.write(grados[i]);  // Enviando el valor de la posicion
      Serial.println(enemigosvivos);
      enemigosvivos++;
    }
  }
  Wire.write(200); // terminando la transmision de datos
  Wire.endTransmission();

}

 void mediana(int ma []) {
  int longitud = enemigoseliminados;
  int bandera = 0;
  int numero = 0;
  for (int i = longitud; i > 0 && bandera == 0; i--) {
    bandera = 1;
    for (int y = 0; y < i; y++) {
      if (ma[y] > ma[y + 1]) {
        numero = ma[y];
        ma[y] = ma[y + 1];
        ma[y + 1] = numero;
        bandera = 0;
      }
    }
  }
  if (longitud % 2 != 0) {
    //        print.serial(ma[longitud/2]);
```

```
    lcd.setCursor(6, 0);
    lcd.print("Me: " + String(ma[longitud / 2]));
    //printf("\nEL valor de la mediana es : %d",ma[longitud/2]);
  } else {
    //        printf("\nEL valor 1 de la mediana es : %d",ma[longitud/2]);
    //        printf("\nEL valor 2 de la mediana es : %d",ma[(longitud/2)-1]);
    lcd.setCursor(6, 0);
    lcd.print("Med: " + String((ma[longitud / 2] + ma[(longitud / 2) - 1]) / 2));
    //        print.serial(ma[longitud/2]);
    //        print.serial(ma[(longitud/2)-1]);
  }

}

void media(int ma []) {
  int suma = 0;
  int longitud = 10;
  for (int i = 0; i < longitud; i++) {
    suma += ma[i];
  }
  float resultado = suma / longitud;
  lcd.setCursor(0, 1);
  lcd.print("X: " + String(resultado));
  //print.serial(resultado);

}

void moda(int ma [] ) {
  int longitud = 10;
  int auxiliar[longitud];
  // int ma[10] = {1,2,3,2,20,7,90,83,1,0};
```

```
for (int i = 0; i < longitud; i++) {
  auxiliar[i] = 0;
}

int posicion;
int numero;
//leo las veces que se repite cada numero.
for (int i = 0; i < longitud; i++) {
  numero = ma[i];
  posicion = i;
  for (int y = i; y < longitud; y++) {
    if (ma[y] == numero) auxiliar[posicion]++;
  }
}
// VEO QUIEN ES EL MAYOR
int mayor = auxiliar[0];
int posicionmayor = 0;
for (int i = 0; i < longitud; i++) {
  if (auxiliar[i] > mayor) {
    posicionmayor = i;
    mayor = auxiliar[i];
  }

}
// Visualizar el elemento con mas frecuencia de aparicion


// print.serial(ma[posicionmayor]);
lcd.setCursor(9, 1);
lcd.print("Mo: " + String(ma[posicionmayor]));
```

}

Código Arduino Esclavo

```
#include <Wire.h>
#include <Servo.h>
int TRIG = 12;
int ECO = 11;
int DURACION;
int DISTANCIA;
int ESTADO = 0;
int LASER = 9;
char DISTANCIAC;
int grados[10];
int contador = 0;
//servo
Servo servomec;
void setup()
{
  Wire.begin(2);                // Este Esclavo es el número 2

  Wire.onReceive(requestEvent);
  Wire.onRequest(datos); // Cuando el Maestro le hace una petición,
  pinMode(TRIG, OUTPUT);
  pinMode(ECO, INPUT);                          // realiza el requestEvent
  Serial.begin(9600);
  servomec.attach(8);//Pin
  pinMode(LASER,OUTPUT);
  digitalWrite(LASER,LOW);
```

```
}

void loop()
{
  if(ESTADO == 4){

    Disparar();
  }
  delay(500);

}

void datos() {

  if(ESTADO == 0){
    Wire.write(1);
  }else if(ESTADO == 1){
    barrido();
  }

}

// Esto es lo que envía cuando le hace la petición.
void requestEvent()
{

  int x = Wire.read();    // receive byte as an integer
  if(x==0){
    ESTADO = 0;
  }
  else if (x == 1 ) {
```

```
    ESTADO = 1;
    if(contador >180){
      contador =0;
    }

  } else if (x == 2) {
    ESTADO = 2;
  }
  else if (x == 3) {
    char ok = Wire.read();

    if (ok == 'o') { // Los arreglos son iguales
      int pos = 0;
      x = Wire.read();
      while (x != 200) { // Mientras no se envie un 100, seguir leyendo lo que
envia el maestro
        grados[pos] = x; // Guardando lo enviado por el maestro
        pos++; // incrementando la posicion del arreglo
        x = Wire.read(); // Leyendo lo siguiente que envio el Maestro
      }
 Serial.println("Distancias obtenidas en el Esclavo: ");
      for (int i = 0; i < 10; i++) {
      Serial.print(grados[i]);
        //servomec.write(distancias[i]);
        delay(1000);
        Serial.print(", ");
      }
    }
    else {
//      Serial.println("Hay distancias diferentes");
    }
```

```
  }
  else if(x == 4){
//    Serial.println("requestEvent");
    ESTADO = 4;
  }

}



void barrido() {

    servomec.write(contador);

    digitalWrite(TRIG, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG, LOW);
    DURACION = pulseIn(ECO, HIGH);

    DISTANCIA = (DURACION * 0.034) / 2;

    DISTANCIAC = (char)DISTANCIA;

   Wire.write(DISTANCIAC);

    delay(1000);


//  for (int i = 0 ; i < 180 ; i += 18) {
```

```
//    servomec.write(i);
//    digitalWrite(TRIG, LOW);
//    delayMicroseconds(2);
//    digitalWrite(TRIG, HIGH);
//    delayMicroseconds(10);
//    digitalWrite(TRIG, LOW);
//    DURACION = pulseIn(ECO, HIGH);
//
//    DISTANCIA = (DURACION * 0.034) / 2;
//
//    DISTANCIAC = (char)DISTANCIA;
//
//    Serial.println(DISTANCIA);
//
//    Wire.write(DISTANCIAC);
//    delay(1000);
//  }
//  for (int i = 0 ; i < 180 ; i += 18) {
//    servomec.write(i);
//    digitalWrite(TRIG, LOW);
//    delayMicroseconds(2);
//    digitalWrite(TRIG, HIGH);
//    delayMicroseconds(10);
//    digitalWrite(TRIG, LOW);
//    DURACION = pulseIn(ECO, HIGH);
//
//    DISTANCIA = (DURACION * 0.034) / 2;
//
//    DISTANCIAC = (char)DISTANCIA;
//
//    Serial.println(DISTANCIA);
```

```
//    Wire.write(DISTANCIAC);
//    delay(1000);
//  }

  contador+=18;


   ESTADO = 0;

}

void Disparar() {
//      Serial.println("Metodo disparar");
      for (int i = 0; i <10 ; i++) {

        servomec.write(grados[i]);
        delay(1000);
        digitalWrite(LASER,HIGH);
        delay(1000);
        digitalWrite(LASER,LOW);
        delay(2000);
      }
      ESTADO = 0;
}
```