UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1
SEGUNDO SEMESTRE 2020
ING. OTTO ESCOBAR
AUXILIAR SECCIÓN A: HERBERTH ARGUETA
AUXILIAR SECCIÓN B: SUSEL RETANA

PRACTICA 2 GRUPO 18

Esdras Benjamín Cotto Revolorio
- **200412823**

Steven Aaron Sis Hernández
- **201706357**

Miguel Angel Solis Yantuche
- **201700543**

Jesús Alejandro Mansilla Villatoro
- **201709396**

Eli Samuel Mazariegos Ramirez
- **201709426**

INTRODUCCIÓN

El uso de la tecnología se expandido por todo el mundo debido a que facilita la creación de nuevas soluciones, simular los proyectos trae consigo ventajas de poder trabajar a distancia, poder probar los proyectos, sin gastar en componentes físicos.

Un buen diseño, incluye que sea ordenado, fácil de entender y sobre todo funcional, esto facilita la comprensión entre los colaboradores del proyecto. El desarrollo de la tecnología para mejorar herramientas o facilitar procesos es de vital importancia para el desarrollo de la humanidad.

El uso de motores, display y diversos componentes facilitará la creación de proyectos cada vez más complejos, se desarrollará un carrito sigue líneas el cual tiene como objetivo utilizar las herramientas de Arduino y simulación de componentes electrónicos.

OBJETIVO GENERAL:

- Que el estudiante adquiera, aplique e interactúe con el microcontrolador Arduino.

OBJETIVOS ESPECÍFICOS:

- Comprender el funcionamiento de las entradas y salidas, tanto digitales como análogas del
- microcontrolador Arduino.
- Comprender el funcionamiento de los dispositivos electromecánicos (motores DC, stepper, servomotores, etc.).
- Aplicar el uso de sensores de color.
- Comprender la configuración de las matrices de luces LED para visualizar texto alfanumérico y percepciones de movimiento aparente.
- Aplicar el lenguaje C para las estructuras de control en microcontroladores.

CONTENIDO

CARRITO SEGUIDOR DE LÍNEAS

Descripción:

Carro sigue línea compuesta por los siguientes componentes:

1. 2 motores de tipo Stepper y dos motores DC
2. 1 Servomotor
3. Sensores Infrarrojos
4. Sensor Ultrasónico
5. Matriz Led con el integrado MAX2719

FUNCIONES DE CARRITO SIGUE LÍNEAS

El carro seguidor de línea negra contara con distintas funciones:

- Debe poder permanecer en una pista negra sin salirse de ella - Debe poder esquivar obstáculos que encuentre en su camino.
- Debe poder ir trazando su camino en una matriz led con el integrado MAX7219
- Debe poder identificar objetos sobre la pista, los cuales activaran el servomotor simulando que el carro está barriendo dicho objeto.

ESTRUCTURA CARRO TORTUGA:

Pala la simulación en Turtle Virtual:

- 2 Ruedas conformadas por motores de tipo DC
- Sensor Ultrasónico para esquivar obstáculos
- Sensores de color para el control de los componentes

COMPONENTES SOBRE EL PANEL DE PROTEUS:

Los siguientes componentes y sus respectivas funciones solamente serán simuladas en el panel de proteus, no son necesarias su simulación en la ventana de Turtle Virtual:

Matriz Led: Mostrando los movimientos en tiempo real del carro, más adelante se explica con detalle.

Barredora: estará compuesta por el servomotor, debe dar 3 vueltas de 360° y media vuelta (180°) simulando la barrida de objetos, más adelante se explica con detalle.

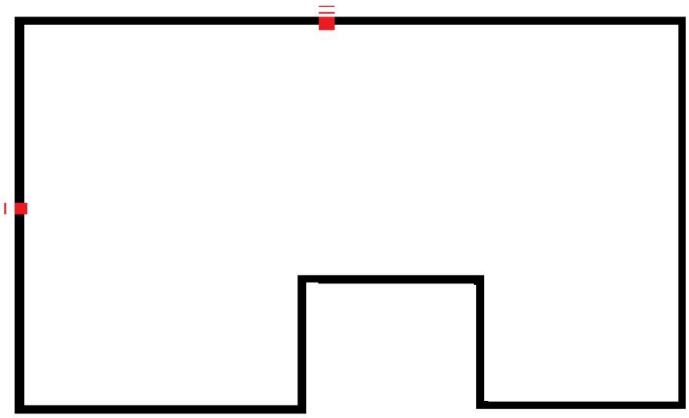
Motores Stepper: estos motores simularan la función de las llantas, deben simular los movimientos que hacen los motores DC, al mismo tiempo y la misma cantidad de movimiento, pero en el panel de proteus.

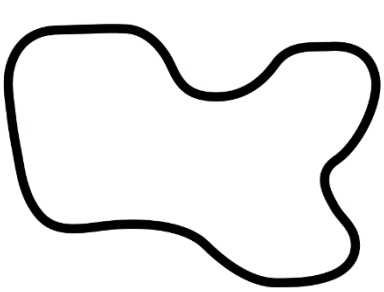
PISTAS

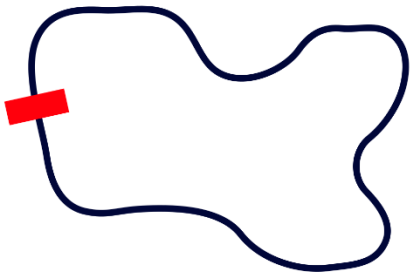
Las pistas que el carrito seguirá se muestran a continuación. Con y sin obstáculos.

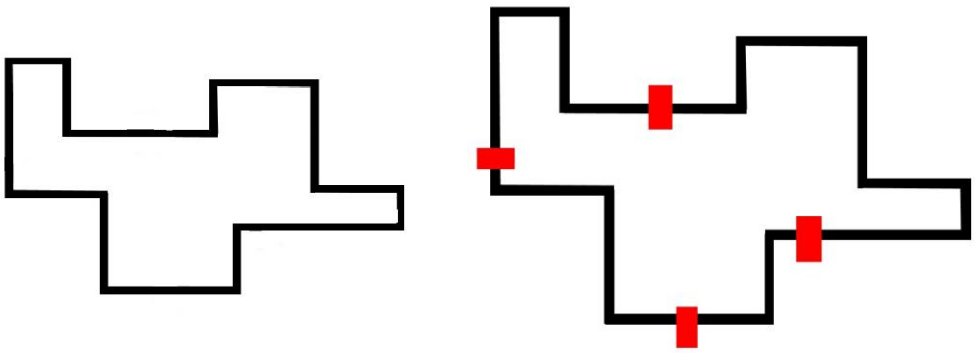
PISTA DE PRUEBA

Esta es la pista de prueba que se utiliza, para comprobar las funcionalidades que debe ejecutar el carrito seguidor de líneas.





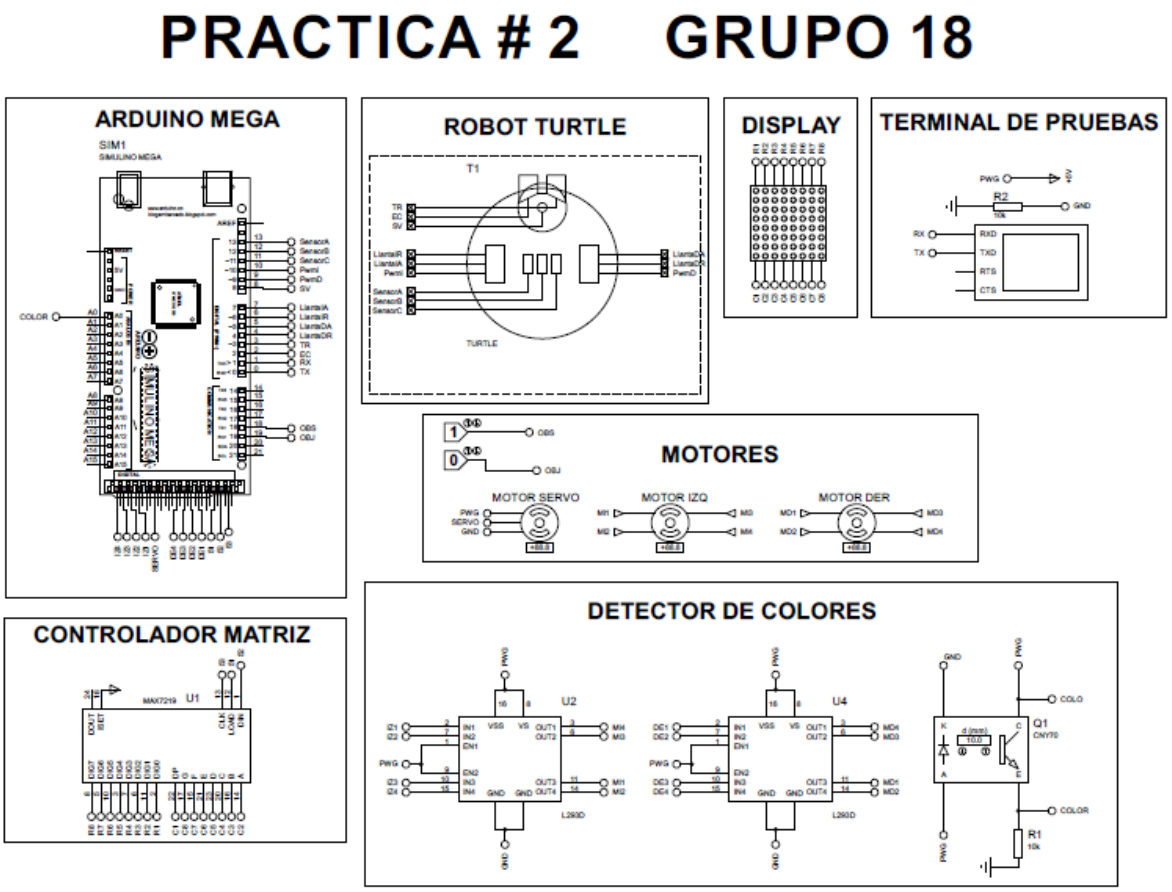




CIRCUITO

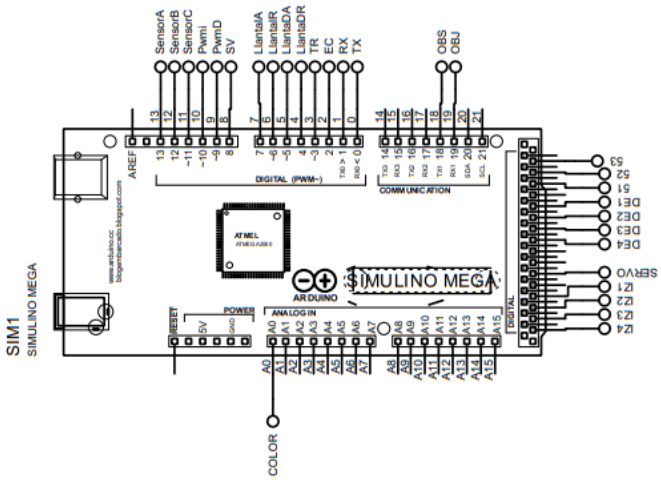
El circuito diseñado en Proteus, se muestra a continuación, todos los elementos utilizados. En esta primera imagen se muestra como esta distribuido todos los componentes dentro del proyecto Proteus.

Más adelante se mostrarán los circuitos individuales para poderse apreciar con más detenimiento la configuración con la que cuenta.



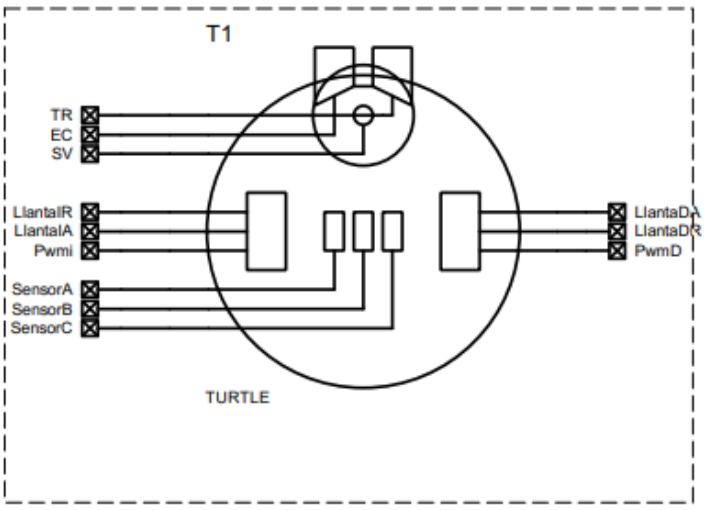
SIMULADOR DE ARDUINO MEGA

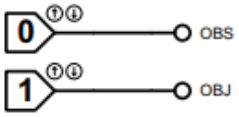
Se muestran todas las configuraciones de las conexiones utilizadas para realizar el proyecto, para esto se utilizó simulino para emular el funcionamiento de un Arduino mega, funciona como una librería integrada dentro de Proteus.



DIAGRAMAS

Se muestra turtle para simular el funcionamiento, se muestra el esquema principal de cómo están distribuidos los componentes.

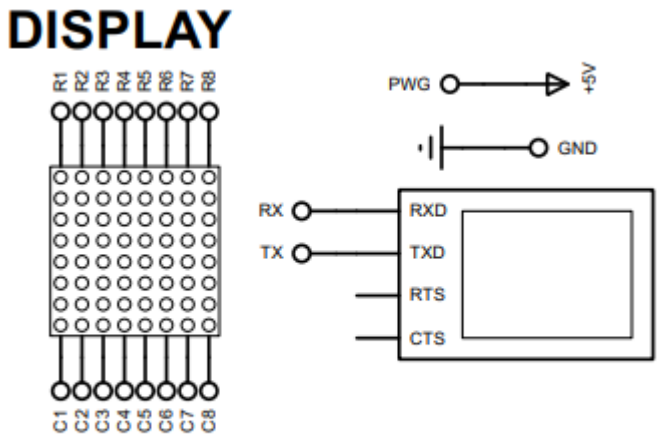




DISPLAY

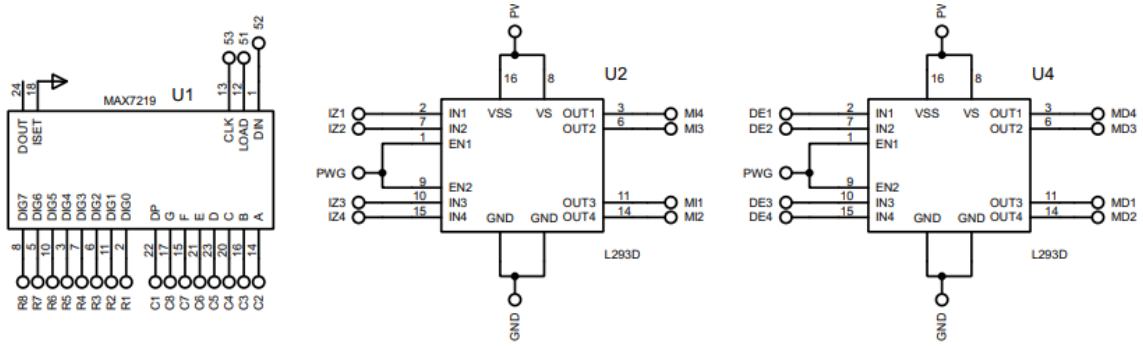
Los displays utilizados y su configuración se muestran a continuación. Esta mostrara un punto parpadeante indicando en que coordenada se encontró el obstáculo.

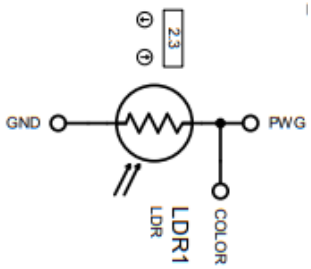
Para su control se utiliza un Max2719.



MAX7219 Y OTROS

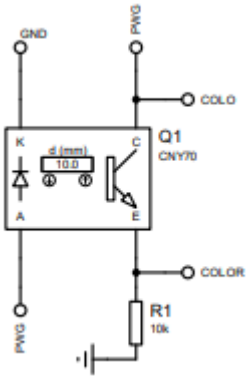
Configuración del integrado Max7219 y otros componentes utilizados en la practica





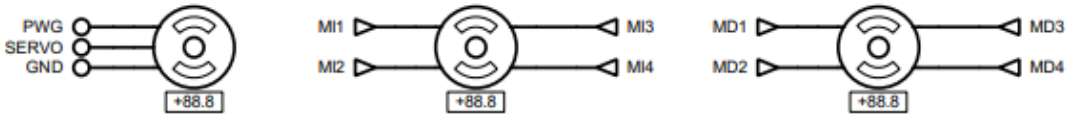
DETECTOR DE COLOR

El detector de color utilizado es el siguiente cny70 y a continuación se muestran la configuración de sus conexiones.



MOTORES

Se muestran la simulación de los 2 motores y servomotor



CÓDIGO ARDUINO

En el siguiente código comentado para facilitar la lectura (comentarios de color verde) se muestran todas las funciones que tiene codificado la práctica.

Código Arduino

```
#include <LedControl.h>
#include <Servo.h>
#include <Servo.h>
#include <Stepper.h>
#include <NewPing.h>

//Definir sensor de sonido
  #define Trigger 3
  #define Echo 2
//Fin de sensor de sonido

//Sensor de color
  #define COLOR A0
//Fin sensor de color
```

```
//Definir Llantas
  #define LlantaIA 7
  #define LlantaIR 6
  #define LlantaDA 5
  #define LlantaDR 4
//Fin define llantas

//Definir Sensores
  #define Si 13
  #define Sc 12
  #define Sd 11
//Fin Define Sensores

//Definir variaciones
  #define V1 18
  #define V2 19
//Fin Define variaciones

//Definir stepper
  Stepper STPI(20, 47, 49, 51, 53);
  int velSTPI=256;
  Stepper STPD(20, 31, 33, 35, 37);
  int velSTPD=256;
//Fin de stepper

//Definir servo proteus/turtle
  Servo servo;
  Servo Tservo;
//Fin servo
```

DEFINICIÓN DE PINES

Código Arduino

```
//Sensor sonido
  NewPing sonar(Trigger, Echo, 200);
//Sensor sonido

//Variables
  int SA,SB,SC;
  int Velocidad=0;
  int Estabilizador=0;
  int Graduador,Referencia;
  unsigned long Tiempo;
  int Centraliza = 0;
```

```
  int Direccion = 0;
//Fin Variables

//Velocidad Pwm
  #define PwmI 10
  #define PwmD 9
//Fin Velocidad Pwm

//Otras variables
  int Distancia;
  int ps=0;
  long t,d;
//Fin otras variables

// LedControl(DATA, CLK, CS, NoDEVICE)
LedControl matrix = LedControl(25, 23, 27, 1);
```

DEFINICIÓN DE MATRIZ

Código Arduino

```
//MATRIZ
int mat_clear[8][8] =   {
  {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}
};
```

CONFIGURACIÓN INICIAL

Código Arduino

```
void setup() {
  //Llamas Como Salidas
    pinMode(LlantaIA, OUTPUT);
    pinMode(LlantaIR, OUTPUT);
    pinMode(LlantaDA, OUTPUT);
    pinMode(LlantaDR, OUTPUT);
  //Fin Llantas

  //Configuracion Sensores
```

```
    pinMode(Si, INPUT);
    pinMode(Sc, INPUT);
    pinMode(Sd, INPUT);
  //Fin Configuracion Sensores

  //Pwm Configuracion
    pinMode(PwmI, OUTPUT);
    pinMode(PwmD, OUTPUT);
  //Fin configuracion pwm

  //Configuracion sensor de sonido
    pinMode(Trigger, OUTPUT);
    pinMode(Echo, INPUT);
    pinMode(COLOR, INPUT);
    digitalWrite(Trigger, LOW);
  //Fin Configuracion sensor de sonido

  //Configuracion servo
    servo.attach(43,1100,2000);
    servo.write(0);
    Tservo.attach(8);
  //Fin servo

  //Velocidad de stepper proteus
    STPI.setSpeed(300);
    STPD.setSpeed(300);


  //Velicidad Comuncacion Serial
    Serial.begin(9600);
  //Fin Velocidad Serial

  matrix.shutdown(0, false);
  matrix.setIntensity(0, 10);
  matrix.clearDisplay(0);

}
```

MOSTRAR RECORRIDO POR COORDENADAS DEL CARRITO, ATREVES DE UNA MATRIZ 8X8

Código Arduino

```
/*  COMIENZA PINTAR RECORRIDO DE CARRITO EN MATRIZ 8X8*/
/*STRUC CARRO*/
struct Carro{
    int x = 3;
```

```
    int y = 3;
};
Carro carrito;
int tab_principal[8][8];
int dir_x = 1;
int dir_y = 1;
int contador_pasos = 0;
int mov_x = 1, mov_y = 0;
int grados_m = 0;

void pintar_matriz(int caracter[][8]) {
  for (int i = 0; i < 8; i++) {
    for (int j = 0; j < 8; j++) {
      if (caracter[i][j] == 1) {
        // TODO: matrix.setLed(number, row, column, state);
        matrix.setLed(0, i, j, HIGH);
      }
    }
    delay(2);
  }
}
```

ACTUALIZAR COORDENADAS

Código Arduino

```
void actualizar_tablero(){
  for(int i = 0; i < 8; i++){
    for(int j = 0; j < 8; j++){
      tab_principal[i][j] = 0;
      if(carrito.x == j && carrito.y == i){
        tab_principal[i][j] = 1;
      }
    }
  }
  pintar_matriz(tab_principal);
}
```

CAMBIAR ORIENTACIÓN

Código Arduino

```
int cont_loop = 0;
void cambiar_orientacion(){
  if(mov_x == 1){
    mov_x = 0;
    mov_y = 1;
```

```
    //rotar_y();
  }else{
    mov_x = 1;
    mov_y = 0;
    //rotar_x();
  }
      if(cont_loop < 4){
        cont_loop++;
    }else{
      matrix.clearDisplay(0);
      cont_loop = 0;
    }


}
```

ROTAR EN X

Código Arduino

```
void rotar_x(){
  dir_x *= -1;
}
void rotar_y(){
  dir_y *= -1;
}
void moveren_x(){
  if(carrito.x + dir_x < -1 || carrito.x + dir_x > 7){
    dir_x *= -1;
  }else{
     carrito.x += dir_x;
  }
}
```

MOVER EN Y

Código Arduino

```
void moveren_y(){
  if(carrito.y + dir_y < -1 || carrito.y + dir_y > 7){
    dir_y *= -1;
  }else{
     carrito.y+= dir_y;
  }
}
void mover_carrito(int grados){
     if(contador_pasos < 90){
        contador_pasos++;
     }else{
        if(grados > 30){
           cambiar_orientacion();
           grados_m = 0;
        }else{
           if(mov_x == 1){
              moveren_x();
           }else if(mov_y == 1){
              moveren_y();
           }
           contador_pasos = 0;
        }
     }
}
```

LOOP GENERAL

Código Arduino

```
int cm=0;
void loop() {
  todoTurtle();
  //todoMotores();

  if(digitalRead(V1) == HIGH || digitalRead(V2) == HIGH){
    if(cm < 20){
      cm++;
    }else{
      todoSonido();
      cm = 0;
    }
  }else{
    todoMotores();
  }
```

CONFIGURACIÓN DE SONIDO

Código Arduino

```
void todoSonido(){

  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);          //Enviamos un pulso de 10s
  digitalWrite(Trigger, LOW);

  int uS = sonar.ping_median();
  uS = uS / US_ROUNDTRIP_CM;
  // Imprimir la distancia medida a la consola serial
  Serial.print("Distancia: ");
  // Calcular la distancia con base en una constante
  Serial.print(uS);
  Serial.println("cm");

  //Código para evitar obstaculo
  if(uS > 0 && uS <= 8) {

    int col = analogRead(COLOR);
    if(digitalRead(V2) == HIGH){
      Serial.print("Color: ");
      Serial.println(col);
      if(col >= 36 && col <= 42){
        mediaVuelta();
```

```
        simuladorServo();
      }
    }else if(digitalRead(V1) == HIGH){
      mediaVuelta();
    }
  }
}
```

MEDIA VUELTA

Código Arduino

```
void mediaVuelta(){
  analogWrite(PwmI,105);
  analogWrite(PwmD,105);
  digitalWrite(LlantaIA, HIGH);
  digitalWrite(LlantaIR, LOW);
  digitalWrite(LlantaDA, LOW);
  digitalWrite(LlantaDR, HIGH);

  delay(2000);

  analogWrite(PwmI,0);
  analogWrite(PwmD,0);
  digitalWrite(LlantaIA, LOW);
  digitalWrite(LlantaIR, LOW);
  digitalWrite(LlantaDA, LOW);
  digitalWrite(LlantaDR, LOW);

  delay(500);
}
```

SIMULADOR DE SERVOMOTOR

Código Arduino

```
//Simulador de servo motor
void simuladorServo(){
  ps++;
  servo.write(180);
  delay(5000);
  servo.write(0);
  delay(5000);
```

```
}
```

ACTIVAR MOTORES

Código Arduino

```
//Movimiento motores en proteus
void todoMotores(){
  //Estabilizada
  if(SA==HIGH && SB==HIGH && SC==HIGH){
    //delay(70);
    if(Estabilizador==1){
       //Movimiento de stepper izq
       STPI.step(256);
       //Movimiento de stepper der
       STPD.step(-256);

       //delay(700);
    }
    if(Estabilizador==2){
       //Movimiento de stepper izq
       STPI.step(-256);
       //Movimiento de stepper der
       STPD.step(256);

       //delay(700);
    }
  }
  //Estabilizada

  if(SA==LOW && SB==LOW && SC == LOW){
    if(Estabilizador==1){
      //Movimiento de stepper izq
      STPI.step(-256);
      //Movimiento de stepper der
      STPD.step(-256);

    }
    if(Estabilizador==2){
      //Movimiento de stepper izq
      STPI.step(256);
      //Movimiento de stepper der
      STPD.step(256);
    }
  }
```

```
//Sin sensor

//SensorCentral
if(SA==LOW && SB==HIGH && SC==LOW){
  //Movimiento de stepper izq
  STPI.step(256);
  //Movimiento de stepper der
  STPD.step(256);

  //delay(300);
}
//FinSensorCentral

//SensorCentralIzquierda
if(SA==HIGH && SB==HIGH && SC==LOW){
  //Movimiento de stepper izq
  STPI.step(128);
  //Movimiento de stepper der
  STPD.step(256);

  //delay(100);

  //Movimiento de stepper izq
  STPI.step(128);
  //Movimiento de stepper der
  STPD.step(256);
}
//FinSensorCentralIzquierda


//SensorCentralDerecha
if(SA==LOW && SB==HIGH && SC==HIGH){
  //Movimiento de stepper izq
  STPI.step(256);
  //Movimiento de stepper der
  STPD.step(128);

  //delay(100);

  //Movimiento de stepper izq
  STPI.step(256);
  //Movimiento de stepper der
  STPD.step(128);
}
//FinSensorCentralDerecha
```

```
  //SensorIzquierda
  if(SA==HIGH && SB==LOW && SC==LOW){
     //Movimiento de stepper izq
     STPI.step(64);
     //Movimiento de stepper der
     STPD.step(512);
  }
  //FinSensorIzquierda

  //SensorDerecha
  else if(SA==LOW && SB==LOW && SC==HIGH){
     //Movimiento de stepper izq
     STPI.step(512);
     //Movimiento de stepper der
     STPD.step(64);
  }
  //SensorDerecha

}
```

CONFIGURACIÓN TURTLE

Código Arduino

```
void todoTurtle(){
  //LecturaSensores
    actualizar_tablero();
    SA = digitalRead(Si);// izquierda
    SB = digitalRead(Sc); // centro
    SC = digitalRead(Sd); // derecha
  //FinLecturaSensores

  //Estabilizada
  if(SA==HIGH && SB==HIGH && SC==HIGH){
    //delay(70);
    mover_carrito(0);
    if(Estabilizador==1){
       analogWrite(PwmI,80);
       analogWrite(PwmD,85);

       digitalWrite(LlantaIA, HIGH);
       digitalWrite(LlantaIR, LOW);
```

```
      digitalWrite(LlantaDA, LOW);
      digitalWrite(LlantaDR, HIGH);

      Referencia = 0;
      Graduador = 0;
      delay(700);

      //Movimiento de stepper izq
      //STPI.step(256);
      //Movimiento de stepper der
      //STPD.step(-256);
   }
   if(Estabilizador==2){
      analogWrite(PwmI,85);
      analogWrite(PwmD,80);

      digitalWrite(LlantaIA, LOW);
      digitalWrite(LlantaIR, HIGH);

      digitalWrite(LlantaDA, HIGH);
      digitalWrite(LlantaDR, LOW);

      Referencia = 0;
      Graduador = 0;
      delay(700);

      //Movimiento de stepper izq
      //STPI.step(-256);
      //Movimiento de stepper der
      //STPD.step(256);
   }
  }

  if(SA==LOW && SB==LOW && SC == LOW){/// izq = off, centro = iff, der = o
ff
   if(Estabilizador==1){
     analogWrite(PwmI, 65);
     analogWrite(PwmD, 50);

     digitalWrite(LlantaIA, LOW);
     digitalWrite(LlantaIR, HIGH);

     digitalWrite(LlantaDA, LOW);
     digitalWrite(LlantaDR, HIGH);
```

```
    Referencia=0;
    Graduador =0;
    delay(15);

    //Movimiento de stepper izq
    //STPI.step(-256);
    //Movimiento de stepper der
    //STPD.step(-256);
  }
  if(Estabilizador==2){
    analogWrite(PwmI, 50);
    analogWrite(PwmD, 65);

    //FinVelocidad
    digitalWrite(LlantaIA, LOW);
    digitalWrite(LlantaIR, HIGH);

    digitalWrite(LlantaDA, LOW);
    digitalWrite(LlantaDR, HIGH);
    Referencia = 0;
    Graduador = 0;
    delay(15);

    //Movimiento de stepper izq
    //STPI.step(256);
    //Movimiento de stepper der
    //STPD.step(256);
  }
}

//SensorCentral
if(SA==LOW && SB==HIGH && SC==LOW){
  //ConfiguracionVelacidad
    mover_carrito(0);

    analogWrite(PwmI,100);
    analogWrite(PwmD,100);

    digitalWrite(LlantaIA,HIGH);
    digitalWrite(LlantaIR,LOW);

    digitalWrite(LlantaDA,HIGH);
    digitalWrite(LlantaDR,LOW);

    Referencia = 0;
```

```
    Graduador = 0;
    Estabilizador = 0;
    Direccion = 0;

    //Movimiento de stepper izq
    //STPI.step(256);
    //Movimiento de stepper der
    //STPD.step(256);
}
//FinSensorCentral

//SensorCentralIzquierda
if(SA==HIGH && SB==HIGH && SC==LOW){
  mover_carrito(grados_m++);

  analogWrite(PwmI,30);
  analogWrite(PwmD,200+Graduador);

  digitalWrite(LlantaIA, HIGH);
  digitalWrite(LlantaIR, LOW);

  digitalWrite(LlantaDA, HIGH);
  digitalWrite(LlantaDR, LOW);

  delay(100);

  //Movimiento de stepper izq
  //STPI.step(128);
  //Movimiento de stepper der
  //STPD.step(256);

  //ConfiguracionVelocidad
  analogWrite(PwmI, 50);
  analogWrite(PwmD, 100+Graduador);

  digitalWrite(LlantaIA, HIGH);
  digitalWrite(LlantaIR, LOW);

  digitalWrite(LlantaDA, HIGH);
  digitalWrite(LlantaDR, LOW);

  Estabilizador = 1;

  //Movimiento de stepper izq
  //STPI.step(128);
```

```
  //Movimiento de stepper der
  //STPD.step(256);
}
//FinSensorCentralIzquierda

//SensorCentralDerecha
if(SA==LOW && SB==HIGH && SC==HIGH){
    mover_carrito(grados_m++);

    analogWrite(PwmI, 200+Graduador);
    analogWrite(PwmD, 30);

    digitalWrite(LlantaIA,HIGH);
    digitalWrite(LlantaIR,LOW);

    digitalWrite(LlantaDA, HIGH);
    digitalWrite(LlantaDR, LOW);

    Referencia = 0;
    delay(100);
    //Movimiento de stepper izq
    //STPI.step(256);
    //Movimiento de stepper der
    //STPD.step(128);

    //ConfiguracionVelocidad
    analogWrite(PwmI, 100+Graduador);
    analogWrite(PwmD, 50+Graduador);

    digitalWrite(LlantaIA, HIGH);
    digitalWrite(LlantaIR, LOW);

    digitalWrite(LlantaDA, HIGH);
    digitalWrite(LlantaDR, LOW);

    Referencia = 0;
    Estabilizador = 2;

    //Movimiento de stepper izq
    //STPI.step(256);
    //Movimiento de stepper der
    //STPD.step(128);
}
//FinSensorCentralDerecha
```

```
  //SensorIzquierda
  if(SA==HIGH && SB==LOW && SC==LOW){
      mover_carrito(grados_m++);
      //ConfiguracionVelocidad
      analogWrite(PwmI, 40);
      analogWrite(PwmD, 145+Graduador);

      digitalWrite(LlantaIA, LOW);
      digitalWrite(LlantaIR, HIGH);

      digitalWrite(LlantaDA,HIGH);
      digitalWrite(LlantaDR,LOW);

      Estabilizador = 1;

      //Movimiento de stepper izq
      //STPI.step(256);
      //Movimiento de stepper der
      //STPD.step(128);
  }
  //FinSensorIzquierda
  //SensorDerecha
  if(SA==LOW && SB==LOW && SC==HIGH){
    mover_carrito(grados_m++);

    analogWrite(PwmI, 145+Graduador);
    analogWrite(PwmD, 40);

    digitalWrite(LlantaIA, HIGH);
    digitalWrite(LlantaIR, LOW);

    digitalWrite(LlantaDA, LOW);
    digitalWrite(LlantaDR, HIGH);

    Estabilizador = 2; //Izuierda

    //Movimiento de stepper izq
    //STPI.step(512);
    //Movimiento de stepper der
    //STPD.step(64);
  }
}
```