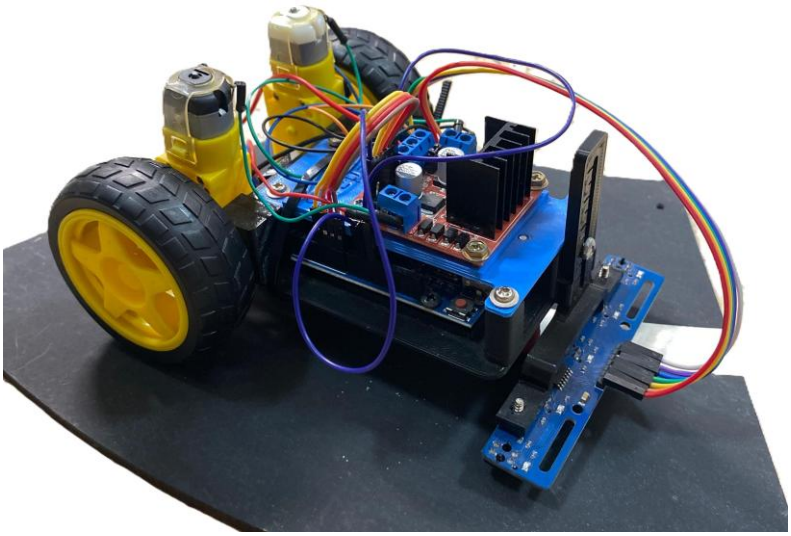


PROYECTO DE EXTENSION: ROBOT SIGUELINEAS

FACULTAD DE INGENIERIA, UNLP
CDR.FIUNLP@GMAIL.COM

¿QUE ES UN SIGUELINEAS?

El siguelíneas es un robot que detecta y sigue una línea blanca sobre un fondo negro utilizando sensores infrarrojos y lógica de control PID. El proyecto tiene como objetivo introducir conceptos de electrónica, programación y control lógico mediante una aplicación funcional y didáctica.

INTRODUCCION

¿QUE NECESITAMOS?

TABLA DE COMPONENTES

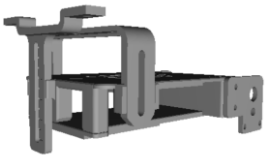
Componente	Cantidad	Descripción	Imagen
Placa Arduino UNO (o similar)	1	Es el “cerebro” de nuestro robot.	
Sensor infrarrojo digital (TCRT5000)	5	Estos sensores detectan la línea blanca, a través de emisión y recepción de luz infrarroja.	
Driver L298N (Puente H)	1	Controla los motores y el movimiento del robot.	
Motor de Corriente Continua con Ruedas	2	Encargado de darle tracción a las ruedas y poder mover el robot.	
Batería o fuente de alimentación	1	Nos dará la alimentación de tensión que necesitamos.	
Chasis impreso o armado	1	Es el chasis del robot, es decir, su estructura física.	
Cables – Precintos - Tornillos	Varios	Elementos necesarios para el armado del chasis.	

DIAGRAMA DE CONEXIONES

El objetivo de estos diagramas es modelar como seria el conexionado de los componentes individuales con nuestra placa Arduino UNO. La integración de todos los elementos podría llevar algunas reconexiones con respecto a las figuras para su correcto funcionamiento.

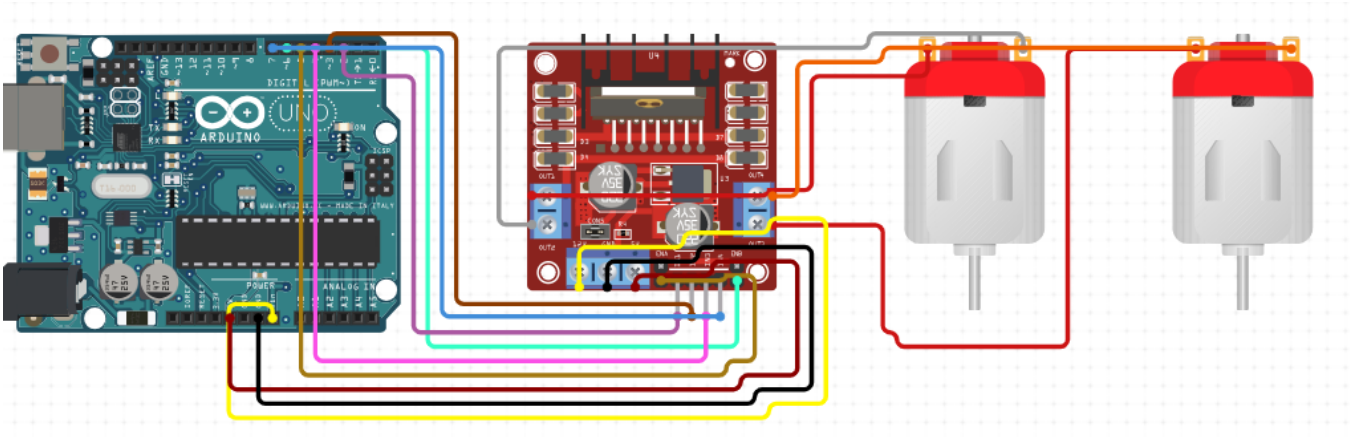


FIGURA 1: CONEXIÓN DEL PUENTE H CON ARDUINO

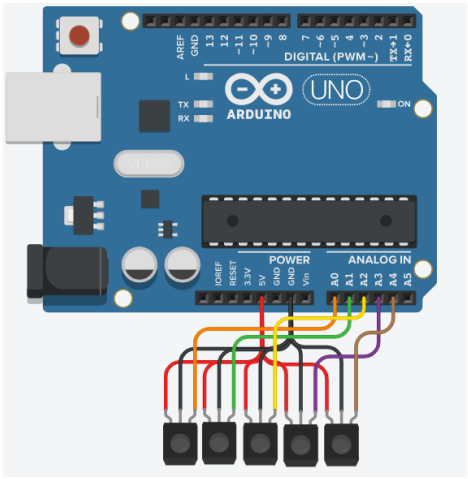


FIGURA 2: CONEXIÓN SENSORES INFRARROJOS CON EL ARDUINO

TABLA DE CONEXIONES CON EL ARDUINO

Pin Arduino	Nombre en el código	Componente conectado
A0	Sensores[0]	Sensor izquierdo
A1	Sensores[1]	Sensor centro – izquierda
A2	Sensores[2]	Sensor centro
A3	Sensores[3]	Sensor centro – derecha
A4	Sensores[4]	Sensor derecho
10	Motores[0]	Enable motor izquierdo
11	Motores[1]	Enable motor derecho
7	Motores[2]	IN1 del motor izquierdo
6	Motores[3]	IN2 del motor derecho
5	Motores[4]	IN3 del motor derecho
4	Motores[5]	IN4 del motor derecho

INSTRUCCIONES BASICAS DE ARMADO

Ahora, a modo practico, vamos a armar paso a paso nuestro robot.

1. Montar los sensores y el controlador en la parte superior e inferior del chasis.
2. Montar los motores al chasis y comprobar que giren libremente.
3. Conectar los motores al controlador L298N, verificando sus conexiones.
4. Conectar el Arduino con el L298N.
5. Integrar todos los componentes al chasis.
6. Alimentar el sistema con una batería adecuada.
7. Cargar el código en el Arduino.

CODIGO DEL ROBOT (BASICO)

Empezamos declarando las variables que utilizaremos para programar nuestro siguelíneas. Inicializamos dos arreglos que nos serán útiles para gestionar nuestras variables de entrada (sensores) y salida (motores), además, ingresamos algunos parámetros iniciales a las variables con las que manejaremos el PID.

```
const int sensores[5] = {A0, A1, A2, A3, A4}; // S0..S4 (Izq a Der)
const int motores[6] = {10, 11, 7, 6, 5, 4};  // ENABLE1, ENABLE2, IN1, IN2, IN3, IN4

// Variables globales del PID
int lastError = 0; // Ultima error registrado
float integral = 0; // Acumulador del termino integral
int lastKnownPosition = 2000; // Ultima posicion conocida (inicio en el centro)
```

Inicializamos nuestras variables declaradas anteriormente, de acuerdo con si son variables de entradas o variables de salida.

```
void setup() {
  Serial.begin(9600); // Inicia comunicacion para depurar

  // Configura los pines de los sensores y motores

  for (int i = 0; i < 5; i++) {
    pinMode(sensores[i], INPUT);
  }

  for (int j = 0; j < 6; j++) {
    pinMode(motores[j], OUTPUT);
  }
}
```

Con las variables declaradas e inicializadas, pasamos a la parte del loop “infinito” que seguirá nuestro robot. Allí se encuentra toda la lógica necesaria para el funcionamiento correcto del robot.

```
void loop() {
  // Lectura de sensores y cálculo de posición
  int sensorValues[5];
  int position = 0;
  int activeSensors = 0;

  for (int i = 0; i < 5; i++) {
    sensorValues[i] = digitalRead(sensores[i]);
    if (sensorValues[i] == HIGH) {
      position += i * 1000; // Configuro peso a los sensores (0,1000,2000,3000,4000)
      activeSensors++; // Cuento cantidad de sensores activos
    }
  }

  int linePosition;
  if (activeSensors > 0) {
    linePosition = position / activeSensors; // Promedio posicion
    lastKnownPosition = linePosition;
  } else {
    linePosition = lastKnownPosition; // Mantener última posición si se pierde la línea
  }
```

Nuestro robot utiliza PID para un funcionamiento mucho mas preciso y sin tantos errores, asociando muchos parámetros que nos servirán para este control.

```
// PID
int setPoint = 2000; // Centro
int error = linePosition - setPoint; // Error actual

// Constantes PID
float Kp = 0.12; // Ganancia proporcional
float Ki = 0.0005; // Ganancia integral
float Kd = 0.2; // Ganancia derivativa

integral += error;
integral = constrain(integral, -1000, 1000); // Acota el valor para proteger contra integral windup

int derivative = error - lastError; // Cambio de error respecto al anterior
float output = Kp * error + Ki * integral + Kd * derivative; // Cálculo de salida PID
lastError = error; // Actualiza el último error
```

Finalmente, con los cálculos ya realizados, procedemos a implementar el movimiento del robot utilizando los motores. Es importante agregar un delay(x) al final para lograr mayor estabilidad.

```
// Control de motores
int baseSpeed = 120; // Velocidad base de los motores (ajustable)
int maxSpeed = 255; // Velocidad máxima de los motores

int leftSpeed = baseSpeed + output; // Ajuste de velocidad izquierda
int rightSpeed = baseSpeed - output; // Ajuste de velocidad derecha

// Limita las velocidades para no exceder el PWM máximo
leftSpeed = constrain(leftSpeed, 0, maxSpeed);
rightSpeed = constrain(rightSpeed, 0, maxSpeed);

// Motor izquierdo
analogWrite(motores[0], leftSpeed);    // ENABLE1
digitalWrite(motores[2], HIGH);        // IN1 (sentido adelante activado)
digitalWrite(motores[3], LOW);         // IN2 (sentido trasero no activado)

// Motor derecho
analogWrite(motores[1], rightSpeed);   // ENABLE2
digitalWrite(motores[4], HIGH);        // IN3 (sentido adelante activado)
digitalWrite(motores[5], LOW);         // IN4 (sentido trasero no activado)
```

Opcionalmente, si queremos agregar un debug vía serial, implementaremos lo siguiente

```
// Debug por Serial
Serial.print("Error: ");
Serial.print(error);
Serial.print(" | Output: ");
Serial.print(output);
Serial.print(" | L: ");
Serial.print(leftSpeed);
Serial.print(" R: ");
Serial.println(rightSpeed);
```