

USO DE IA PARA CONSULTAR SOBRE
EL ESAMBLADO Y PROGRAMACIÓN DE COMPONENTES PARTE #1

Equipo SJT RoboClub

Se le consultó a la IA (ChatGPT) sobre el armado y programación del auto autónomo, especificándole los componentes a usar y que función tendría cada uno dentro del proyecto.

Se nos facilitó un armado y una programación base dejando en claro que debíamos configurar algunos componentes para el buen funcionamiento de nuestro proyecto.

Al leer el diagrama de flujo y la programación nos dimos cuenta de que la lógica para el funcionamiento del auto estaba incompleta así que seguimos mejorando las consultas.

Se consultó sobre la alimentación del proyecto ya que contará con dispositivos como los servomotores y lidar en constante movimiento, con referencia a la idea de alimentar la placa de arduino Atmega separada del controlador de motor L298N y asegurar su correcta conexión y funcionamiento.

No se lo consultó sobre la idea del proyecto ni los componentes a utilizar.

CONSULTA A LA IA:

Necesito que me muestres detalladamente los pines de alimentación y comunicación para ensamblar los componentes que te voy a mencionar para crear un auto autónomo que esquiva bloques de color rojos y verdes en una cuadrado de 3 metros por tres metros,

Los componentes serán los siguientes:

Te mostraré la función de cada uno para que me muestres:

- a) Como conectar cada componente paso a paso
- b) La programación según las funciones a realizar cada uno de los componentes.
- c) Un diagrama de flujo gráfico

1. Arduino ATmega 2560: para montar los componentes, si necesito algo más me lo puedes sugerir.
2. Lidar WayPonDEV FHL-LD19: en un sensor de distancia 2D de 360, con este dispositivo deseo que el auto logre dar tres vueltas dentro del cuadro de 3 metros cuadrados sin chocar con los pilotes que encuentre en su camino ni los límites del cuadro los cuales son bordes borde de madera de 100mm de altura de color negro.
3. Una cámara Pixy2 Smart Vision Sensor: la cual identificará unos paralelepípedo rectangulares con dimensiones de 50x50x100 que encontrará en su camino y evitar colisionar con paralelepípedo de madera de color verde RGB (68, 214, 44), donde, El auto no cruza el radio al conducir desde el lado derecho del pilar verde y otro rojo RGB (238, 39, 55) donde, El auto no cruzará el radio al conducir desde el lado derecho del pilar rojo,
4. Motor de eje pasante: este motor logrará que el auto se despase hacia delante y hacia atrás.
5. Servomotor: este dispositivo va a transmitir la dirección del conductor al eje de dirección, que moverá las ruedas delanteras hacia la izquierda y hacia la derecha.
6. Controlador de motor L298N: este dispositivo controlará el Motor de eje pasante.
7. Kit de dos servomotores para Pixy2 - Soporte de cámara robótica de doble eje para que se mueva la cámara de arriba hacia abajo y de derecha a izquierda para buscar el paralelepípedo rectangular de colores verdes y rojos.
8. servomotor que va a transmiten la dirección del conductor al eje de dirección, para la dirección asistida de las llantas delanteras que mueve las ruedas delanteras hacia la izquierda y hacia la derecha, para evitar el paralelepípedo rectangular de colores verdes y rojos.

Para la programación me gustaría tomes en cuenta lo siguiente: luego de dar tres vueltas en dentro del cuadro de 3 metros cuadrados, el auto deberá estacionarse en estacionamiento de un paralelepípedo rectangular con dimensiones de 200x20x100 mm, de color magenta / RGB (255, 0, 255) que debe ser detectado por la cámara pixy2

ChatGPT

Memoria actualizada

Para ensamblar y programar tu auto autónomo, vamos a seguir los pasos detallados para conectar cada componente y escribir el código necesario para controlar sus funciones. Además, se proporcionará un diagrama de flujo gráfico.

Componentes y Conexiones

1. Arduino ATmega 2560

Este será el controlador principal al cual se conectarán todos los demás componentes.

2. Lidar WayPonDEV FHL-LD19

- **Alimentación:** VCC a 5V en el Arduino, GND a GND en el Arduino.
- **Comunicación:** RX a TX1 (19) en el Arduino, TX a RX1 (18) en el Arduino.

3. Pixy2 Smart Vision Sensor

- **Alimentación:** VCC a 5V en el Arduino, GND a GND en el Arduino.
- **Comunicación:** Utiliza el cable dedicado Pixy2 para I2C o SPI (según prefieras), por ejemplo, SDA a SDA (20) y SCL a SCL (21) en el Arduino.

4. Motor de eje pasante

- **Conexión:** A los terminales de salida del controlador de motor L298N.

5. Servomotor para dirección de las ruedas delanteras

- **Alimentación:** VCC a 5V en el Arduino, GND a GND en el Arduino.
- **Señal:** Cable de señal al pin digital 9 en el Arduino.

6. Controlador de motor L298N

- **Alimentación:** VCC a 12V (fuente de alimentación externa), GND a GND de la fuente y el Arduino.
- **Control:** IN1 e IN2 al pin 5 y 6 del Arduino, ENA al pin 3 del Arduino.

7. Kit de dos servomotores para Pixy2

- **Alimentación:** VCC a 5V en el Arduino, GND a GND en el Arduino.
- **Señal:** Servomotor horizontal al pin digital 10 en el Arduino y servomotor vertical al pin digital 11 en el Arduino.

Conexión de los componentes

Aquí tienes un esquema de conexión básica:

Programación

Vamos a realizar la programación por partes:

Configuración del Lidar

```
cpp
Copiar código
#include <SoftwareSerial.h>

SoftwareSerial lidarSerial(18, 19); // RX, TX

void setup() {
  Serial.begin(9600);
  lidarSerial.begin(115200);
}

void loop() {
  // Código para leer datos del Lidar y procesarlos
}
```

Configuración de la Pixy2

```
cpp
Copiar código
#include <Pixy2.h>

Pixy2 pixy;

void setup() {
  pixy.init();
}

void loop() {
  pixy.ccc.getBlocks();
  if (pixy.ccc.numBlocks) {
    // Procesar bloques detectados por Pixy2
  }
}
```

Control del Motor

```
cpp
Copiar código
const int motorIn1 = 5;
const int motorIn2 = 6;
const int motorEn = 3;

void setup() {
  pinMode(motorIn1, OUTPUT);
  pinMode(motorIn2, OUTPUT);
  pinMode(motorEn, OUTPUT);
}
```

```

}

void adelante() {
    digitalWrite(motorIn1, HIGH);
    digitalWrite(motorIn2, LOW);
    analogWrite(motorEn, 255); // Velocidad máxima
}

void atras() {
    digitalWrite(motorIn1, LOW);
    digitalWrite(motorIn2, HIGH);
    analogWrite(motorEn, 255); // Velocidad máxima
}

```

Control del Servomotor para dirección

```

cpp
Copiar código
#include <Servo.h>

Servo servoDireccion;

void setup() {
    servoDireccion.attach(9);
}

void loop() {
    // Código para controlar la dirección
}

```

Control del Kit de Servomotores para Pixy2

```

cpp
Copiar código
#include <Servo.h>

Servo servoH;
Servo servoV;

void setup() {
    servoH.attach(10);
    servoV.attach(11);
}

void loop() {
    // Código para controlar los servomotores del Pixy2
}

```

Diagrama de flujo

1. Iniciar sistemas (Lidar, Pixy2, motores, servos).
2. Leer datos del Lidar y Pixy2.
3. Procesar datos para detectar obstáculos y bloques.
4. Controlar motores y servos para esquivar obstáculos y seguir el recorrido.
5. Repetir el proceso hasta completar tres vueltas.
6. Al completar las vueltas, buscar el bloque de estacionamiento.
7. Estacionarse en el lugar indicado.

Programación Completa

```
cpp
Copiar código
#include <SoftwareSerial.h>
#include <Pixy2.h>
#include <Servo.h>

SoftwareSerial lidarSerial(18, 19); // RX, TX
Pixy2 pixy;
Servo servoDireccion;
Servo servoH;
Servo servoV;

const int motorIn1 = 5;
const int motorIn2 = 6;
const int motorEn = 3;

void setup() {
    Serial.begin(9600);
    lidarSerial.begin(115200);
    pixy.init();
    servoDireccion.attach(9);
    servoH.attach(10);
    servoV.attach(11);

    pinMode(motorIn1, OUTPUT);
    pinMode(motorIn2, OUTPUT);
    pinMode(motorEn, OUTPUT);
}

void adelante() {
    digitalWrite(motorIn1, HIGH);
    digitalWrite(motorIn2, LOW);
    analogWrite(motorEn, 255); // Velocidad máxima
}

void atras() {
    digitalWrite(motorIn1, LOW);
    digitalWrite(motorIn2, HIGH);
    analogWrite(motorEn, 255); // Velocidad máxima
}

void loop() {
    // Leer y procesar datos del Lidar
    if (lidarSerial.available()) {
        // Código para leer y procesar datos del Lidar
    }

    // Leer y procesar datos del Pixy2
    pixy.ccc.getBlocks();
    if (pixy.ccc.numBlocks) {
        for (int i = 0; i < pixy.ccc.numBlocks; i++) {
            // Procesar bloques detectados por Pixy2
        }
    }
}
```

```
// Controlar motores y servos para esquivar obstáculos y seguir el recorrido
// ...

// Código para completar tres vueltas
// ...

// Código para buscar y estacionarse en el bloque de color magenta
// ...
}
```

Este código es una base que necesitará ajustes y adiciones específicas según los detalles del proyecto, pero cubre las conexiones y la programación básica para cada componente.

4o

CONSULTA: MUESTRAME mostraré paso a paso cómo conectar cada componente, cómo programar cada uno y un diagrama me facilitas un diagrama de flujo gráfico.

ChatGPT

Memoria actualizada

Conexiones y Programación

Para ensamblar y programar el auto autónomo, te mostraré paso a paso cómo conectar cada componente, cómo programarlos y proporcionaré un diagrama de flujo gráfico.

Conexión de los Componentes

1. Arduino ATmega 2560

Este será el controlador principal al cual se conectarán todos los demás componentes.

2. Lidar WayPonDEV FHL-LD19

- **Alimentación:**
 - VCC a 5V en el Arduino.
 - GND a GND en el Arduino.
- **Comunicación:**
 - RX a TX1 (pin 18) en el Arduino.
 - TX a RX1 (pin 19) en el Arduino.

3. Pixy2 Smart Vision Sensor

- **Alimentación:**
 - VCC a 5V en el Arduino.
 - GND a GND en el Arduino.
- **Comunicación:**
 - SDA a SDA (pin 20) en el Arduino.
 - SCL a SCL (pin 21) en el Arduino.

4. Motor de eje pasante

- **Conexión:**

- A los terminales de salida del controlador de motor L298N.

5. Servomotor para dirección de las ruedas delanteras

- **Alimentación:**
 - VCC a 5V en el Arduino.
 - GND a GND en el Arduino.
- **Señal:**
 - Cable de señal al pin digital 9 en el Arduino.

6. Controlador de motor L298N

- **Alimentación:**
 - VCC a 12V (de la batería externa).
 - GND a GND de la batería y el Arduino.
- **Control:**
 - IN1 al pin 5 del Arduino.
 - IN2 al pin 6 del Arduino.
 - ENA al pin 3 del Arduino.

7. Kit de dos servomotores para Pixy2

- **Alimentación:**
 - VCC a 5V en el Arduino.
 - GND a GND en el Arduino.
- **Señal:**
 - Servomotor horizontal al pin digital 10 en el Arduino.
 - Servomotor vertical al pin digital 11 en el Arduino.

8. Batería (7.4V)

- **Alimentación:**
 - Conectar al controlador de motor L298N y al Arduino a través del conector de alimentación adecuado.

Diagrama de Conexión

Programación

Vamos a realizar la programación por partes:

Configuración del Lidar

```
cpp
Copiar código
#include <SoftwareSerial.h>

SoftwareSerial lidarSerial(18, 19); // RX, TX

void setup() {
  Serial.begin(9600);
  lidarSerial.begin(115200);
}

void loop() {
  // Código para leer datos del Lidar y procesarlos
}
```

Configuración de la Pixy2

```
cpp
Copiar código
#include <Pixy2.h>

Pixy2 pixy;

void setup() {
    pixy.init();
}

void loop() {
    pixy.ccc.getBlocks();
    if (pixy.ccc.numBlocks) {
        // Procesar bloques detectados por Pixy2
    }
}
```

Control del Motor

```
cpp
Copiar código
const int motorIn1 = 5;
const int motorIn2 = 6;
const int motorEn = 3;

void setup() {
    pinMode(motorIn1, OUTPUT);
    pinMode(motorIn2, OUTPUT);
    pinMode(motorEn, OUTPUT);
}

void adelante() {
    digitalWrite(motorIn1, HIGH);
    digitalWrite(motorIn2, LOW);
    analogWrite(motorEn, 255); // Velocidad máxima
}

void atras() {
    digitalWrite(motorIn1, LOW);
    digitalWrite(motorIn2, HIGH);
    analogWrite(motorEn, 255); // Velocidad máxima
}
```

Control del Servomotor para dirección

```
cpp
Copiar código
#include <Servo.h>

Servo servoDireccion;

void setup() {
    servoDireccion.attach(9);
}

void loop() {
    // Código para controlar la dirección
}
```


Control del Kit de Servomotores para Pixy2

```
cpp
Copiar código
#include <Servo.h>

Servo servoH;
Servo servoV;

void setup() {
  servoH.attach(10);
  servoV.attach(11);
}

void loop() {
  // Código para controlar los servomotores del Pixy2
}
```

Diagrama de Flujo

1. Iniciar sistemas (Lidar, Pixy2, motores, servos).
2. Leer datos del Lidar y Pixy2.
3. Procesar datos para detectar obstáculos y bloques.
4. Controlar motores y servos para esquivar obstáculos y seguir el recorrido.
5. Repetir el proceso hasta completar tres vueltas.
6. Al completar las vueltas, buscar el bloque de estacionamiento.
7. Estacionarse en el lugar indicado.

Programación Completa

```
cpp
Copiar código
#include <SoftwareSerial.h>
#include <Pixy2.h>
#include <Servo.h>

SoftwareSerial lidarSerial(18, 19); // RX, TX
Pixy2 pixy;
Servo servoDireccion;
Servo servoH;
Servo servoV;

const int motorIn1 = 5;
const int motorIn2 = 6;
const int motorEn = 3;

void setup() {
  Serial.begin(9600);
  lidarSerial.begin(115200);
  pixy.init();
  servoDireccion.attach(9);
  servoH.attach(10);
  servoV.attach(11);

  pinMode(motorIn1, OUTPUT);
  pinMode(motorIn2, OUTPUT);
  pinMode(motorEn, OUTPUT);
}
```

```

void adelante() {
    digitalWrite(motorIn1, HIGH);
    digitalWrite(motorIn2, LOW);
    analogWrite(motorEn, 255); // Velocidad máxima
}

void atras() {
    digitalWrite(motorIn1, LOW);
    digitalWrite(motorIn2, HIGH);
    analogWrite(motorEn, 255); // Velocidad máxima
}

void loop() {
    // Leer y procesar datos del Lidar
    if (lidarSerial.available()) {
        // Código para leer y procesar datos del Lidar
    }

    // Leer y procesar datos del Pixy2
    pixy.ccc.getBlocks();
    if (pixy.ccc.numBlocks) {
        for (int i = 0; i < pixy.ccc.numBlocks; i++) {
            // Procesar bloques detectados por Pixy2
        }
    }

    // Controlar motores y servos para esquivar obstáculos y seguir el
    recorrido
    // ...

    // Código para completar tres vueltas
    // ...

    // Código para buscar y estacionarse en el bloque de color magenta
    // ...
}

```

Este código es una base que necesitará ajustes y adiciones específicas según los detalles del proyecto, pero cubre las conexiones y la programación básica para cada componente.

4o

CONSULTA: una consulta los GND del Arduino mega serán suficiente?

ChatGPT

El Arduino Mega 2560 tiene múltiples pines GND disponibles, lo que debería ser suficiente para conectar los componentes mencionados. Sin embargo, es importante asegurar una buena conexión a tierra (GND) para evitar problemas de ruido eléctrico o inestabilidad en el sistema.

En proyectos más complejos, es posible utilizar una barra de distribución de GND (también conocida como bus de GND) para conectar múltiples dispositivos a una única conexión a tierra. Esto garantiza que todos los componentes tengan una referencia de tierra común y reduce el riesgo de problemas eléctricos.

Estrategia para Conexiones de GND

1. Utilizar Múltiples Pines GND:

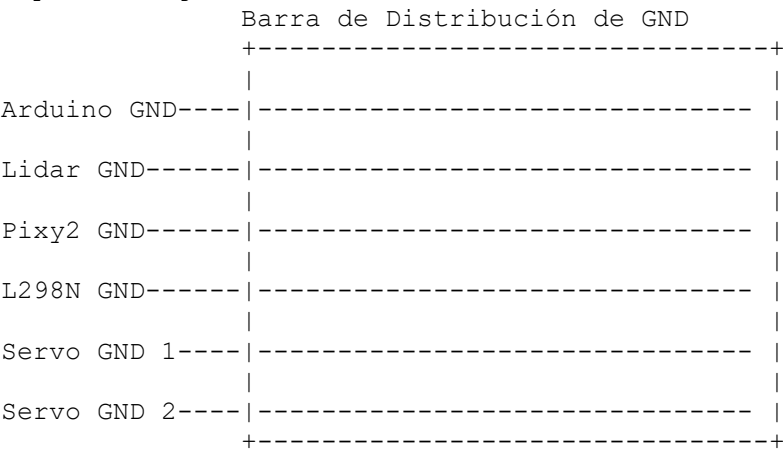
- El Arduino Mega 2560 tiene varios pines GND. Puedes distribuir las conexiones de tierra entre estos pines para balancear la carga.
-
2. **Uso de Barra de Distribución de GND:**
 - Utiliza una barra de distribución de GND para conectar todos los componentes a una única referencia de tierra. La barra se conectará a uno de los pines GND del Arduino.
 3. **Verificar las Conexiones:**
 - Asegúrate de que todas las conexiones de GND estén bien aseguradas y sin interferencias.

Conexiones de GND en tu Proyecto

1. **Lidar WayPonDEV FHL-LD19:** Conectar GND a uno de los pines GND del Arduino.
2. **Pixy2 Smart Vision Sensor:** Conectar GND a otro pin GND del Arduino.
3. **Motor de eje pasante y Controlador L298N:** Conectar GND del controlador L298N a la barra de distribución de GND y luego conectar la barra de distribución al Arduino.
4. **Servomotores (Dirección y Pixy2):** Conectar GND de cada servomotor a la barra de distribución de GND.
5. **Batería (7.4V):** Conectar GND de la batería a la barra de distribución de GND.

Ejemplo de Conexión de Barra de Distribución de GND

plaintext
Copiar código



Conclusión

El Arduino Mega 2560 debería ser capaz de manejar las conexiones GND necesarias para tu proyecto. Sin embargo, para mejorar la estabilidad y reducir el ruido eléctrico, es recomendable usar una barra de distribución de GND. Esto garantizará que todos los componentes tengan una conexión de tierra común y adecuada.

4o

CONSULTA: me puedes sugerir la programación para Controlar motores y servos para esquivar obstáculos y seguir el recorrido // ... // Código para completar tres vueltas // ... // Código para buscar y estacionarse en el bloque de color magenta

ChatGPT

Claro, puedo sugerirte un esquema de programación para controlar los motores y servos para que tu auto esquive obstáculos, complete tres vueltas dentro de un área de 3x3 metros, y luego se estacione en un bloque de color magenta. La estructura del código será modular para facilitar la comprensión y el mantenimiento.

Estructura General del Código

El código se dividirá en las siguientes secciones:

1. Inicialización de los componentes.
2. Funciones para controlar motores y servos.
3. Funciones para leer datos de los sensores.
4. Lógica de navegación para esquivar obstáculos.
5. Contador de vueltas.
6. Estacionamiento en el bloque magenta.

Inicialización de los Componentes

```
cpp
Copiar código
#include <SoftwareSerial.h>
#include <Pixy2.h>
#include <Servo.h>

// Definir pines
const int motorIn1 = 5;
const int motorIn2 = 6;
const int motorEn = 3;
const int servoDireccionPin = 9;
const int servoHPin = 10;
const int servoVPin = 11;

SoftwareSerial lidarSerial(18, 19); // RX, TX
Pixy2 pixy;
Servo servoDireccion;
Servo servoH;
Servo servoV;

int vueltasCompletadas = 0;

void setup() {
  Serial.begin(9600);
  lidarSerial.begin(115200);
  pixy.init();
  servoDireccion.attach(servoDireccionPin);
  servoH.attach(servoHPin);
  servoV.attach(servoVPin);

  pinMode(motorIn1, OUTPUT);
```

```

pinMode(motorIn2, OUTPUT);
pinMode(motorEn, OUTPUT);
}

void loop() {
  if (vueltasCompletadas < 3) {
    // Leer sensores y esquivar obstáculos
    leerSensoresYESquivar();
    // Contar vueltas
    contarVueltas();
  } else {
    // Buscar y estacionarse en el bloque magenta
    buscarYEstacionarse();
  }
}

```

Funciones para Controlar Motores y Servos

```

cpp
Copiar código
void adelante() {
  digitalWrite(motorIn1, HIGH);
  digitalWrite(motorIn2, LOW);
  analogWrite(motorEn, 255); // Velocidad máxima
}

void atrás() {
  digitalWrite(motorIn1, LOW);
  digitalWrite(motorIn2, HIGH);
  analogWrite(motorEn, 255); // Velocidad máxima
}

void parar() {
  digitalWrite(motorIn1, LOW);
  digitalWrite(motorIn2, LOW);
  analogWrite(motorEn, 0); // Parar
}

void girarIzquierda() {
  servoDireccion.write(45); // Ajustar el valor según sea necesario
}

void girarDerecha() {
  servoDireccion.write(135); // Ajustar el valor según sea necesario
}

void centrarDireccion() {
  servoDireccion.write(90); // Posición central
}

```

Funciones para Leer Datos de los Sensores

```

cpp
Copiar código
void leerSensoresYESquivar() {
  // Leer datos del Lidar
  if (lidarSerial.available()) {
    // Código para leer y procesar datos del Lidar
    // Por ejemplo, si detecta un obstáculo a menos de 30 cm, esquivar
    int distancia = leerDistanciaLidar();
    if (distancia < 30) {
      esquivarObstaculo();
    }
  }
}

```

```

// Leer datos del Pixy2
pixy.ccc.getBlocks();
if (pixy.ccc.numBlocks) {
    for (int I = 0; I < pixy.ccc.numBlocks; i++) {
        // Procesar bloques detectados por Pixy2
        if (pixy.ccc.blocks[i].m_signature == 1) { // Supongamos que 1 es verde
y 2 es rojo
            esquivarBloque();
        } else if (pixy.ccc.blocks[i].m_signature == 2) {
            esquivarBloque();
        }
    }
}
}

int leerDistanciaLidar() {
    // Función para leer y retornar la distancia del Lidar
    // Suponiendo que el Lidar envía datos en formato adecuado
    int distancia = 0;
    // Código para leer y procesar la distancia
    return distancia;
}

void esquivarObstaculo() {
    parar();
    girarDerecha();
    adelante();
    delay(1000); // Ajustar tiempo según sea necesario
    centrarDireccion();
}

void esquivarBloque() {
    parar();
    girarIzquierda();
    adelante();
    delay(1000); // Ajustar tiempo según sea necesario
    centrarDireccion();
}

```

Contador de Vueltas

```

cpp
Copiar código
void contarVueltas() {
    // Suponiendo que se utiliza algún sensor o condición para determinar una
vuelta completa
    // Aquí se debe implementar la lógica para contar vueltas
    // Por ejemplo, usando un sensor en el punto de inicio
    static int posicionInicial = leerPosicionInicial();
    int posicionActual = leerPosicionActual();

    if (posicionActual == posicionInicial) {
        vueltasCompletadas++;
    }
}

int leerPosicionInicial() {
    // Código para leer y retornar la posición inicial
    int ack tur = 0;
    // Código para leer y procesar la posición
    return ack tur;
}

```

```
}
```

```
int leerPosicionActual() {  
    // Código para leer y retornar la posición actual  
    int ack tur = 0;  
    // Código para leer y procesar la posición  
    return ack tur;  
}
```

Buscar y Estacionarse en el Bloque Magenta

```
cpp  
Copiar código  
void buscarYEstacionarse() {  
    while (true) {  
        pixy.ccc.getBlocks();  
        if (pixy.ccc.numBlocks) {  
            for (int I = 0; I < pixy.ccc.numBlocks; i++) {  
                if (pixy.ccc.blocks[i].m_signature == 3) { // Supongamos que 3 es  
magenta  
                    parar();  
                    estacionar();  
                    return;  
                }  
            }  
        }  
        adelante();  
    }  
}  
  
void estacionar() {  
    parar();  
    girarIzquierda();  
    adelante();  
    delay(1000); // Ajustar tiempo según sea necesario  
    centrarDireccion();  
    parar();  
}
```

Diagrama de Flujo

1. Iniciar sistemas (Lidar, Pixy2, motores, servos).
2. Leer datos del Lidar y Pixy2.
3. Procesar datos para detectar obstáculos y bloques.
4. Controlar motores y servos para esquivar obstáculos y seguir el recorrido.
5. Repetir el proceso hasta completar tres vueltas.
6. Al completar las vueltas, buscar el bloque de estacionamiento.
7. Estacionarse en el lugar indicado.

Este esquema de programación cubre las funciones básicas para esquivar obstáculos, completar vueltas y estacionarse. Ajustes específicos y refinamientos se deberán hacer según los resultados de las pruebas en el entorno real.

4°

CONSULTA: cómo sería la conexión si alimento la placa de Arduino Atmega 2560 con un Jack de alimentación con batería de 9 voltios y alimento por otro lado el Controlador de motor L298N con la batería de 7,2 ~ 8,4 V (batería de litio de 7,4 V)

ChatGPT

Alimentar el Arduino Mega 2560 y el controlador de motor L298N con fuentes de energía separadas es una opción viable y puede ayudar a distribuir la carga eléctrica. A continuación, te explico cómo realizar las conexiones:

Conexión de la Alimentación

1. Arduino Mega 2560 con Jack de Alimentación de 9V

- **Conexión del Jack de Alimentación:**
 - Conecta el positivo (+) de la batería de 9V al pin central del jack de alimentación del Arduino Mega 2560.
 - Conecta el negativo (-) de la batería de 9V al pin exterior del jack de alimentación del Arduino Mega 2560.

2. Controlador de Motor L298N con Batería de 7.4V

- **Conexión del L298N:**
 - Conecta el positivo (+) de la batería de 7.4V al terminal de 12V del L298N.
 - Conecta el negativo (-) de la batería de 7.4V al terminal GND del L298N.

Conexión de GND Común

Para asegurar que ambas fuentes de alimentación tengan una referencia de tierra común, conecta el GND del Arduino Mega 2560 al GND del L298N.

Esquema de Conexiones

1. Alimentación del Arduino Mega 2560

- **Jack de Alimentación:**
 - Pin central del jack de alimentación del Arduino -> Positivo (+) de la batería de 9V.
 - Pin exterior del jack de alimentación del Arduino -> Negativo (-) de la batería de 9V.

2. Alimentación del L298N

- **Terminal de 12V del L298N:**
 - Terminal 12V del L298N -> Positivo (+) de la batería de 7.4V.
 - Terminal GND del L298N -> Negativo (-) de la batería de 7.4V.

4. Conexión de GND Común

- **GND Común:**
 - GND del Arduino Mega 2560 -> GND del L298N.

Conexión de los Motores y Servomotores

Los motores y servomotores se conectan como se describe a continuación:

Controlador de Motor L298N

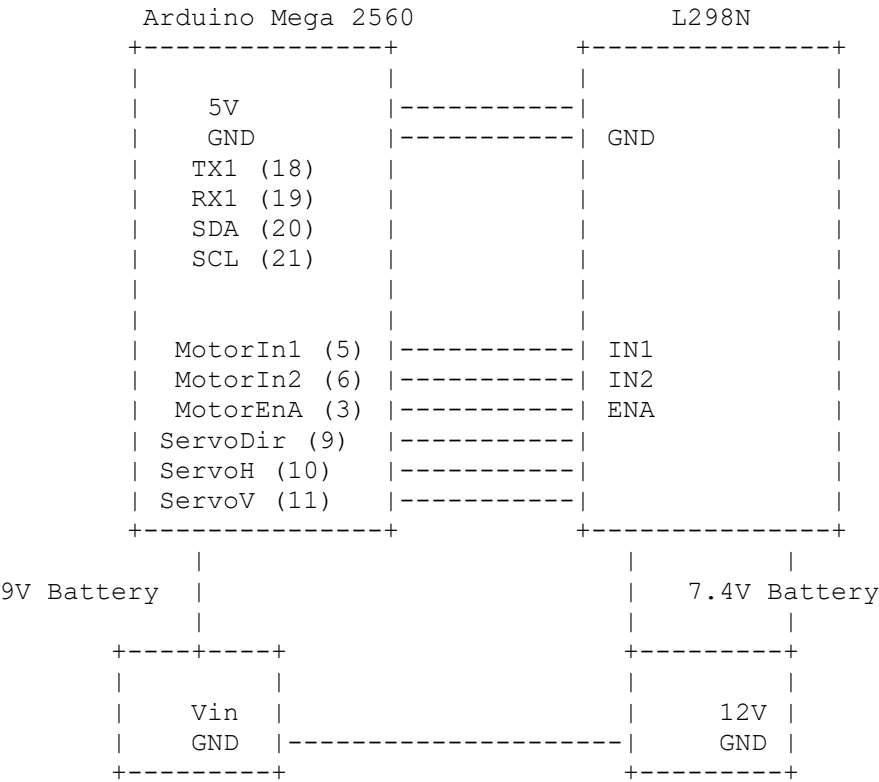
- **Motor de eje pasante:**
 - Conecta los terminales del motor a los pines de salida del L298N (OUT1 y OUT2).
- **Control del motor:**
 - IN1 del L298N -> Pin 5 del Arduino.
 - IN2 del L298N -> Pin 6 del Arduino.
 - ENA del L298N -> Pin 3 del Arduino.

Servomotores

- **Servomotor para dirección:**
 - VCC del servomotor -> 5V del Arduino.
 - GND del servomotor -> GND del Arduino.
 - Señal del servomotor -> Pin 9 del Arduino.
- **Kit de servomotores para Pixy2:**
 - VCC del servomotor horizontal -> 5V del Arduino.
 - GND del servomotor horizontal -> GND del Arduino.
 - Señal del servomotor horizontal -> Pin 10 del Arduino.
 - VCC del servomotor vertical -> 5V del Arduino.
 - GND del servomotor vertical -> GND del Arduino.
 - Señal del servomotor vertical -> Pin 11 del Arduino.

Esquema de Conexión Completo

plaintext
Copiar código



Notas Adicionales

1. **Asegurar Buenas Conexiones:** Verifica todas las conexiones para asegurar que estén firmes y sin cortocircuitos.
2. **Capacidad de las Baterías:** Asegúrate de que las baterías tengan suficiente capacidad (mAh) para alimentar todos los componentes durante el tiempo necesario.
3. **Monitoreo de Voltaje:** Considera agregar un monitor de voltaje para cada batería para asegurarte de que no se descarguen demasiado durante la operación.

Con esta configuración, tendrás una fuente de alimentación dedicada para el Arduino y otra para el controlador de motor, con una referencia de tierra común para evitar problemas de comunicación y estabilidad.

1/2
4°

CONSULTA: La batería de 9 voltios tiene un adaptador de jack

ChatGPT

Perfecto, si la batería de 9V tiene un adaptador de jack, esto simplificará la conexión al Arduino Mega 2560. Aquí te detallo las conexiones necesarias para alimentar tanto el Arduino como el controlador de motor L298N, junto con la conexión de GND común.

Conexión de la Alimentación

1. Arduino Mega 2560 con Jack de Alimentación de 9V

- **Conexión del Jack de Alimentación:**
 - Conecta el adaptador de jack de la batería de 9V al puerto de alimentación del Arduino Mega 2560.

2. Controlador de Motor L298N con Batería de 7.4V

- **Conexión del L298N:**
 - Conecta el positivo (+) de la batería de 7.4V al terminal de 12V del L298N.
 - Conecta el negativo (-) de la batería de 7.4V al terminal GND del L298N.

Conexión de GND Común

Para asegurar que ambas fuentes de alimentación tengan una referencia de tierra común, conecta el GND del Arduino Mega 2560 al GND del L298N.

Esquema de Conexiones

1. Alimentación del Arduino Mega 2560

- **Jack de Alimentación:**
 - Conecta el adaptador de jack de la batería de 9V al puerto de alimentación del Arduino Mega 2560.

2. Alimentación del L298N

- **Terminal de 12V del L298N:**
 - Terminal 12V del L298N -> Positivo (+) de la batería de 7.4V.
 - Terminal GND del L298N -> Negativo (-) de la batería de 7.4V.

3. Conexión de GND Común

- **GND Común:**
 - GND del Arduino Mega 2560 -> GND del L298N.

Conexión de los Motores y Servomotores

Los motores y servomotores se conectan como se describe a continuación:

Controlador de Motor L298N

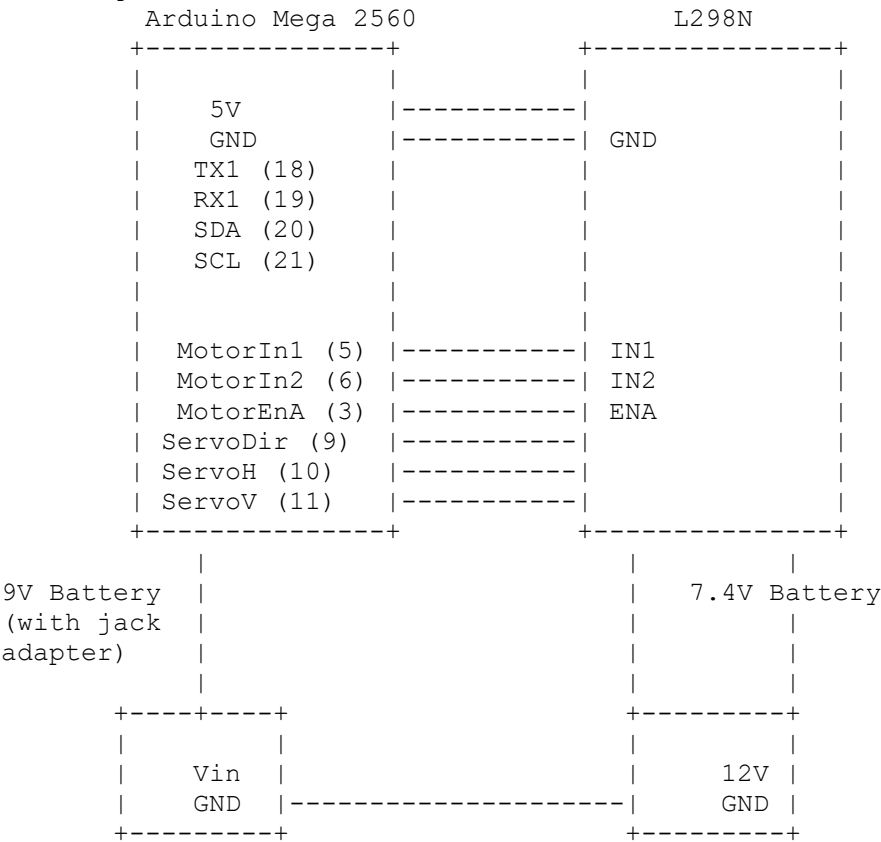
- **Motor de eje pasante:**
 - Conecta los terminales del motor a los pines de salida del L298N (OUT1 y OUT2).
- **Control del motor:**
 - IN1 del L298N -> Pin 5 del Arduino.
 - IN2 del L298N -> Pin 6 del Arduino.
 - ENA del L298N -> Pin 3 del Arduino.

Servomotores

- **Servomotor para dirección:**
 - VCC del servomotor -> 5V del Arduino.
 - GND del servomotor -> GND del Arduino.
 - Señal del servomotor -> Pin 9 del Arduino.
- **Kit de servomotores para Pixy2:**
 - VCC del servomotor horizontal -> 5V del Arduino.
 - GND del servomotor horizontal -> GND del Arduino.
 - Señal del servomotor horizontal -> Pin 10 del Arduino.
 - VCC del servomotor vertical -> 5V del Arduino.
 - GND del servomotor vertical -> GND del Arduino.
 - Señal del servomotor vertical -> Pin 11 del Arduino.

Esquema de Conexión Completo

plaintext
Copiar código



Notas Adicionales

1. **Asegurar Buenas Conexiones:** Verifica todas las conexiones para asegurar que estén firmes y sin cortocircuitos.
2. **Capacidad de las Baterías:** Asegúrate de que las baterías tengan suficiente capacidad (mAh) para alimentar todos los componentes durante el tiempo necesario.
3. **Monitoreo de Voltaje:** Considera agregar un monitor de voltaje para cada batería para asegurarte de que no se descarguen demasiado durante la operación.

Con esta configuración, tendrás una fuente de alimentación dedicada para el Arduino y otra para el controlador de motor, con una referencia de tierra común para evitar problemas de comunicación y estabilidad.

ChatGPT

Sí, debes conectar los GND de ambas baterías para establecer una referencia de tierra común. Esto es crucial para asegurar una comunicación correcta y evitar problemas de ruido eléctrico entre los componentes que usan diferentes fuentes de alimentación.

Aquí te dejo el esquema actualizado y un paso a paso para las conexiones:

Esquema de Conexiones

1. Alimentación del Arduino Mega 2560

- **Jack de Alimentación:**
 - Conecta el adaptador de jack de la batería de 9V al puerto de alimentación del Arduino Mega 2560.

2. Alimentación del L298N

- **Terminal de 12V del L298N:**
 - Terminal 12V del L298N -> Positivo (+) de la batería de 7.4V.
 - Terminal GND del L298N -> Negativo (-) de la batería de 7.4V.

3. Conexión de GND Común

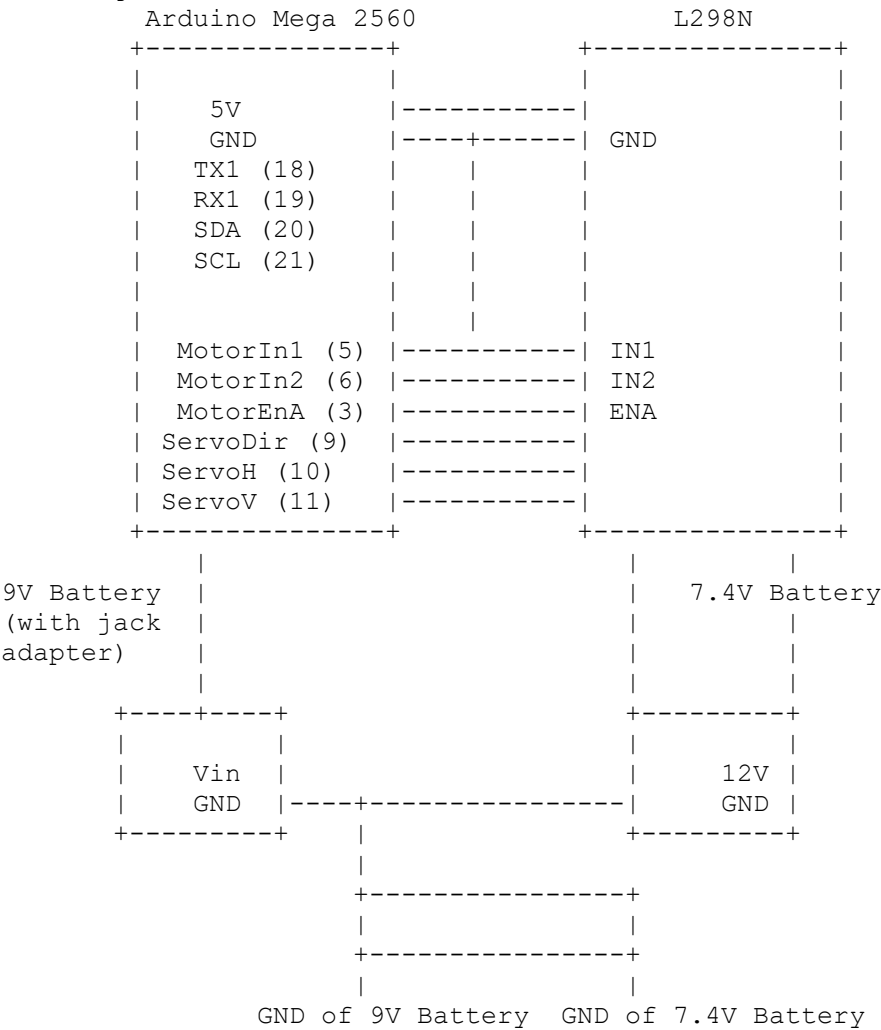
- **GND Común:**
 - Conecta el GND del Arduino Mega 2560 al GND del L298N.
 - Conecta el GND de la batería de 9V al GND de la batería de 7.4V.

Paso a Paso para Conexiones

1. **Conectar el Jack de Alimentación del Arduino:**
 - Inserta el adaptador de jack de la batería de 9V en el puerto de alimentación del Arduino Mega 2560.
2. **Conectar la Alimentación del L298N:**
 - Conecta el terminal 12V del L298N al positivo (+) de la batería de 7.4V.
 - Conecta el terminal GND del L298N al negativo (-) de la batería de 7.4V.
3. **Establecer GND Común:**
 - Conecta el GND del Arduino Mega 2560 al GND del L298N usando un cable.
 - Conecta el GND de la batería de 9V al GND de la batería de 7.4V usando otro cable.

Esquema de Conexión Completo

plaintext
Copiar código



Notas Adicionales

1. **Asegurar Buenas Conexiones:** Verifica todas las conexiones para asegurar que estén firmes y sin cortocircuitos.
2. **Capacidad de las Baterías:** Asegúrate de que las baterías tengan suficiente capacidad (mAh) para alimentar todos los componentes durante el tiempo necesario.
3. **Monitoreo de Voltaje:** Considera agregar un monitor de voltaje para cada batería para asegurarte de que no se descarguen demasiado durante la operación.

Con esta configuración, tendrás una fuente de alimentación dedicada para el Arduino y otra para el controlador de motor, con una referencia de tierra común para evitar problemas de comunicación y estabilidad.

CONSULTA: puedo hacer un puente, muéstrame un esquema

ChatGPT

Sí, puedes hacer un puente para conectar los GND de ambas baterías. Esto se hace fácilmente mediante un cable adicional que conecta los GND de ambas fuentes de alimentación. A continuación, te explico cómo realizar la conexión y te proporciono un esquema actualizado.

Conexión de la Alimentación

1. Arduino Mega 2560 con Jack de Alimentación de 9V

- **Conexión del Jack de Alimentación:**
 - Conecta el adaptador de jack de la batería de 9V al puerto de alimentación del Arduino Mega 2560.

2. Controlador de Motor L298N con Batería de 7.4V

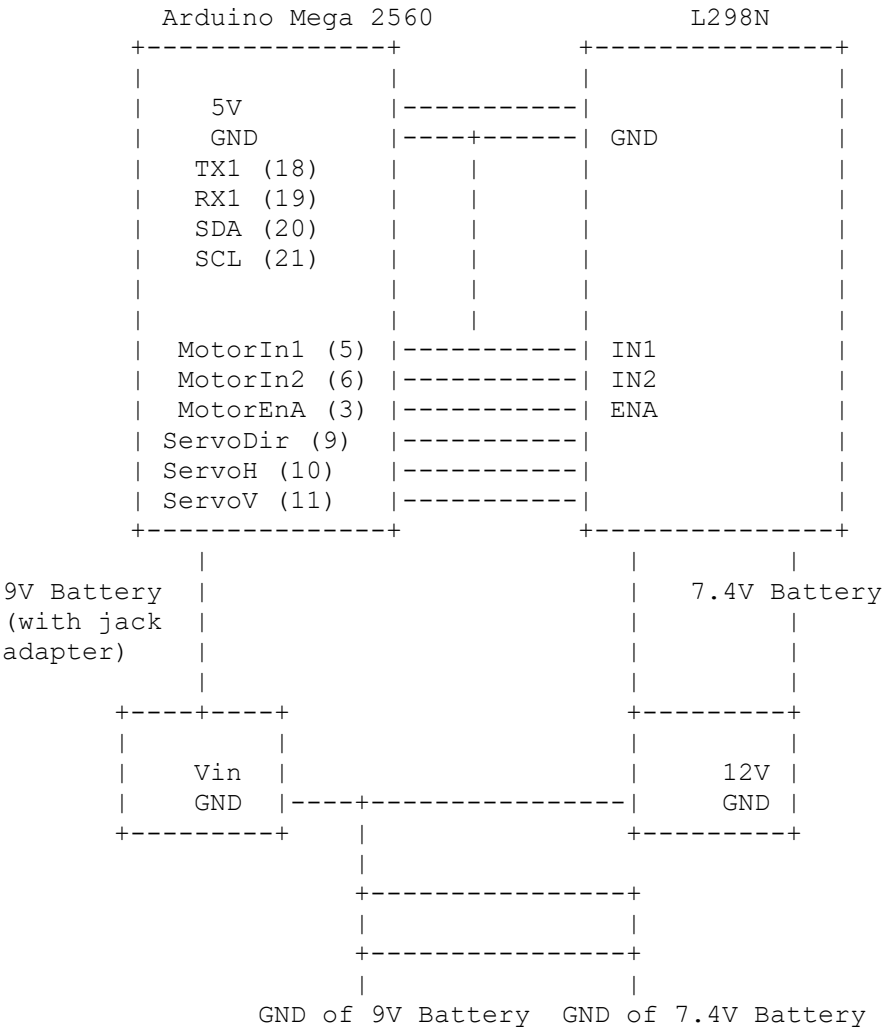
- **Conexión del L298N:**
 - Conecta el positivo (+) de la batería de 7.4V al terminal de 12V del L298N.
 - Conecta el negativo (-) de la batería de 7.4V al terminal GND del L298N.

3. Conexión de GND Común (Puente)

- **GND Común:**
 - Conecta el GND del Arduino Mega 2560 al GND del L298N.
 - Conecta el GND de la batería de 9V al GND de la batería de 7.4V utilizando un cable de puente.

Esquema de Conexiones

plaintext
Copiar código



Paso a Paso para Conexiones

1. **Conectar el Jack de Alimentación del Arduino:**
 - Inserta el adaptador de jack de la batería de 9V en el puerto de alimentación del Arduino Mega 2560.
2. **Conectar la Alimentación del L298N:**
 - Conecta el terminal 12V del L298N al positivo (+) de la batería de 7.4V.
 - Conecta el terminal GND del L298N al negativo (-) de la batería de 7.4V.
3. **Establecer GND Común (Punto):**
 - Conecta el GND del Arduino Mega 2560 al GND del L298N usando un cable.
 - Conecta el GND de la batería de 9V al GND de la batería de 7.4V utilizando un cable adicional para hacer el punto.

Notas Adicionales

1. **Asegurar Buenas Conexiones:** Verifica todas las conexiones para asegurar que estén firmes y sin cortocircuitos.
2. **Capacidad de las Baterías:** Asegúrate de que las baterías tengan suficiente capacidad (mAh) para alimentar todos los componentes durante el tiempo necesario.
3. **Monitoreo de Voltaje:** Considera agregar un monitor de voltaje para cada batería para asegurarte de que no se descarguen demasiado durante la operación.

Con esta configuración, tendrás una fuente de alimentación dedicada para el Arduino y otra para el controlador de motor, con una referencia de tierra común para evitar problemas de comunicación y estabilidad.

4o

CONSULTA: necesito que LOS BLOQUES sean detectados por la cámara pixy2

ChatGPT

Conexiones y Esquema

1. Arduino Mega 2560

- **Función:** Controlador principal del proyecto.

2. Lidar WayPonDEV FHL-LD19

- **Función:** Sensor de distancia 2D de 360° para evitar obstáculos y seguir el recorrido.
- **Pines de conexión:**
 - VCC -> 5V en el Arduino Mega 2560
 - GND -> GND en el Arduino Mega 2560
 - TX -> RX1 (pin 19) en el Arduino Mega 2560
 - RX -> TX1 (pin 18) en el Arduino Mega 2560

3. Cámara Pixy2 Smart Vision Sensor

- **Función:** Identificación de bloques de colores y ayuda en el estacionamiento.
- **Pines de conexión:**
 - VCC -> 5V en el Arduino Mega 2560
 - GND -> GND en el Arduino Mega 2560
 - SDA -> SDA (pin 20) en el Arduino Mega 2560
 - SCL -> SCL (pin 21) en el Arduino Mega 2560

4. Motor de eje pasante

- **Función:** Movimiento del vehículo hacia adelante y hacia atrás.
- **Pines de conexión:**
 - Motor conectado al controlador de motor L298N (OUT1 y OUT2)

5. Servomotor de dirección

- **Función:** Control de la dirección del vehículo.
- **Pines de conexión:**
 - VCC -> 5V en el Arduino Mega 2560
 - GND -> GND en el Arduino Mega 2560
 - Señal -> Pin 9 en el Arduino Mega 2560

6. Controlador de motor L298N

- **Función:** Control del motor de eje pasante.
- **Pines de conexión:**
 - IN1 -> Pin 5 en el Arduino Mega 2560
 - IN2 -> Pin 6 en el Arduino Mega 2560
 - ENA -> Pin 3 en el Arduino Mega 2560
 - 12V -> Positivo (+) de la batería de 7.4V
 - GND -> GND de la batería de 7.4V y GND del Arduino Mega 2560

7. Kit de dos servomotores para Pixy2

- **Función:** Movimiento de la cámara Pixy2 en dos ejes.
- **Pines de conexión:**
 - Servo Horizontal:
 - VCC -> 5V en el Arduino Mega 2560
 - GND -> GND en el Arduino Mega 2560
 - Señal -> Pin 10 en el Arduino Mega 2560
 - Servo Vertical:
 - VCC -> 5V en el Arduino Mega 2560
 - GND -> GND en el Arduino Mega 2560
 - Señal -> Pin 11 en el Arduino Mega 2560

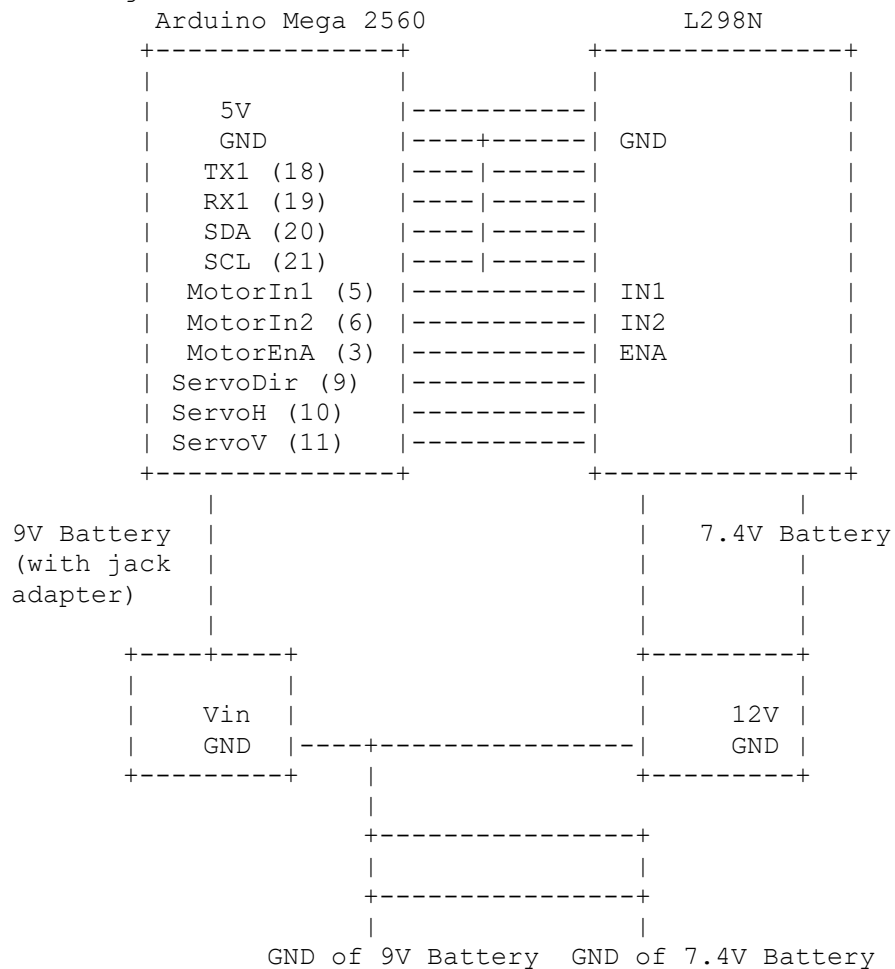
8. Baterías

- **Función:** Alimentación del sistema.
- **Arduino Mega 2560:**
 - Alimentado por una batería de 9V mediante un adaptador jack de alimentación.
- **Controlador de motor L298N:**
 - Alimentado por una batería de 7.4V.

Esquema de Conexiones

plaintext

Copiar código



Programación

Conexiones Iniciales y Definiciones

```
cpp
Copiar código
#include <Wire.h>
#include <Servo.h>

Servo servoDir;
Servo servoH;
Servo servoV;

const int motorIn1 = 5;
const int motorIn2 = 6;
const int motorEnA = 3;
const int servoDirPin = 9;
const int servoHPin = 10;
const int servoVPin = 11;

void setup() {
  // Inicializar serial
  Serial.begin(115200);

  // Configurar pines del motor
  pinMode(motorIn1, OUTPUT);
  pinMode(motorIn2, OUTPUT);
  pinMode(motorEnA, OUTPUT);
  analogWrite(motorEnA, 255); // Velocidad máxima

  // Configurar servomotores
  servoDir.attach(servoDirPin);
  servoH.attach(servoHPin);
  servoV.attach(servoVPin);

  // Inicializar comunicación I2C
  Wire.begin();

  // Inicializar Pixy2
  // Inicializar Lidar
}

void loop() {
  // Leer datos de la Pixy2
  // Leer datos del Lidar
  // Procesar datos y tomar decisiones

  // Ejemplo: evitar obstáculos
  evitarObstaculos();

  // Completar tres vueltas
  completarVueltas();

  // Buscar y estacionarse en el bloque de color magenta
  estacionarseEnMagenta();
}

void evitarObstaculos() {
  // Lógica para evitar obstáculos usando Lidar y Pixy2
}
```

```
void completarVueltas() {  
    // Lógica para completar tres vueltas  
}  
  
void estacionarseEnMagenta() {  
    // Lógica para buscar y estacionarse en el bloque de color magenta  
}
```

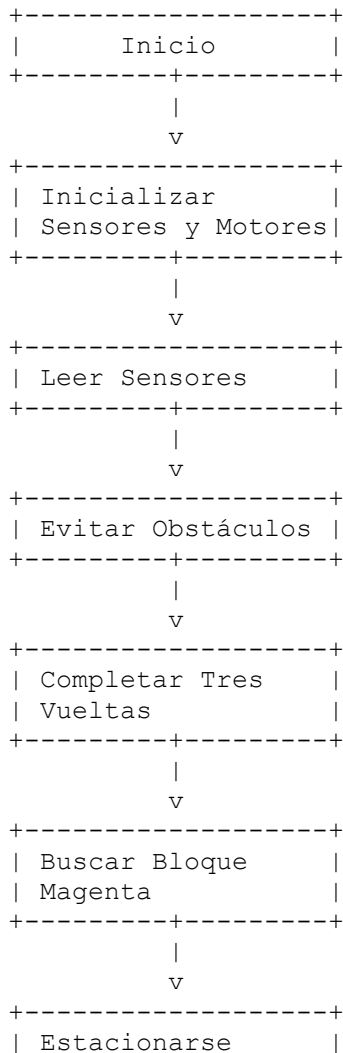
Diagrama de Flujo

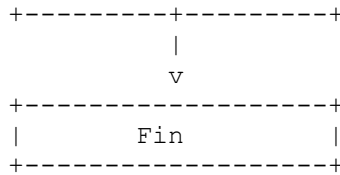
1. **Inicio**
2. **Inicializar Sensores y Motores**
3. **Leer Sensores**
4. **Evitar Obstáculos**
 - Si hay un obstáculo, ajustar dirección y velocidad.
5. **Completar Tres Vueltas**
 - Si se completaron tres vueltas, proceder al siguiente paso.
6. **Buscar Bloque Magenta**
 - Si se encuentra el bloque magenta, proceder al estacionamiento.
7. **Estacionarse**
8. **Fin**

Diagrama de Flujo Gráfico

plaintext

Copiar código





Con este detalle, tienes el esquema de conexiones, una base para la programación, y un diagrama de flujo que te guiará en la implementación del proyecto.

4o

me puedes redactar todo lo consultado en una sola parte
ChatGPT

Proyecto: Auto Autónomo que Esquiva Obstáculos y Estaciona

Componentes:

1. **Arduino ATmega 2560:** Controlador principal del proyecto.
2. **Lidar WayPonDEV FHL-LD19:** Sensor de distancia 2D de 360° para evitar obstáculos y seguir el recorrido.
3. **Cámara Pixy2 Smart Vision Sensor:** Identificación de bloques de colores y ayuda en el estacionamiento.
4. **Motor de eje pasante:** Movimiento del vehículo hacia adelante y hacia atrás.
5. **Servomotor:** Control de la dirección del vehículo.
6. **Controlador de motor L298N:** Control del motor de eje pasante.
7. **Kit de dos servomotores para Pixy2:** Movimiento de la cámara Pixy2 en dos ejes.
8. **Batería:** Alimentación del sistema.
 - o Alimentar el Arduino Mega 2560 con un adaptador jack de alimentación con una batería de 9V.
 - o Alimentar el Controlador de motor L298N con la batería de 7.4V.

Conexiones

Arduino Mega 2560

- **Función:** Controlador principal del proyecto.

Lidar WayPonDEV FHL-LD19

- **Pines de conexión:**
 - o VCC -> 5V en el Arduino Mega 2560
 - o GND -> GND en el Arduino Mega 2560
 - o TX -> RX1 (pin 19) en el Arduino Mega 2560
 - o RX -> TX1 (pin 18) en el Arduino Mega 2560

Cámara Pixy2 Smart Vision Sensor

- **Pines de conexión:**
 - o VCC -> 5V en el Arduino Mega 2560
 - o GND -> GND en el Arduino Mega 2560
 - o SDA -> SDA (pin 20) en el Arduino Mega 2560
 - o SCL -> SCL (pin 21) en el Arduino Mega 2560

Motor de eje pasante

- **Pines de conexión:**
 - o Motor conectado al controlador de motor L298N (OUT1 y OUT2)

Servomotor de dirección

- **Pines de conexión:**
 - VCC -> 5V en el Arduino Mega 2560
 - GND -> GND en el Arduino Mega 2560
 - Señal -> Pin 9 en el Arduino Mega 2560

Controlador de motor L298N

- **Pines de conexión:**
 - IN1 -> Pin 5 en el Arduino Mega 2560
 - IN2 -> Pin 6 en el Arduino Mega 2560
 - ENA -> Pin 3 en el Arduino Mega 2560
 - 12V -> Positivo (+) de la batería de 7.4V
 - GND -> GND de la batería de 7.4V y GND del Arduino Mega 2560

Kit de dos servomotores para Pixy2

- **Pines de conexión:**
 - Servo Horizontal:
 - VCC -> 5V en el Arduino Mega 2560
 - GND -> GND en el Arduino Mega 2560
 - Señal -> Pin 10 en el Arduino Mega 2560
 - Servo Vertical:
 - VCC -> 5V en el Arduino Mega 2560
 - GND -> GND en el Arduino Mega 2560
 - Señal -> Pin 11 en el Arduino Mega 2560

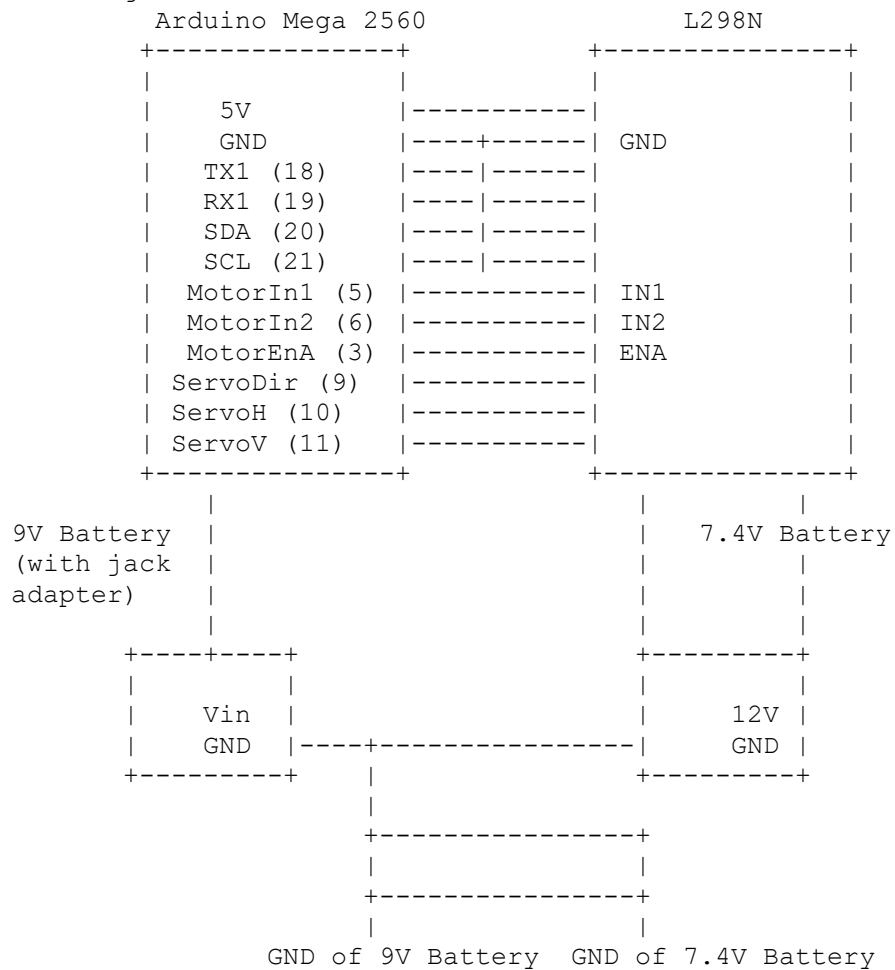
Baterías

- **Arduino Mega 2560:**
 - Alimentado por una batería de 9V mediante un adaptador jack de alimentación.
- **Controlador de motor L298N:**
 - Alimentado por una batería de 7.4V.

Esquema de Conexiones

plaintext

Copiar código



Programación

Conexiones Iniciales y Definiciones

```
cpp
Copiar código
#include <Wire.h>
#include <Servo.h>

Servo servoDir;
Servo servoH;
Servo servoV;

const int motorIn1 = 5;
const int motorIn2 = 6;
const int motorEnA = 3;
const int servoDirPin = 9;
const int servoHPin = 10;
const int servoVPin = 11;

void setup() {
    // Inicializar serial
    Serial.begin(115200);

    // Configurar pines del motor
    pinMode(motorIn1, OUTPUT);
    pinMode(motorIn2, OUTPUT);
    pinMode(motorEnA, OUTPUT);
    analogWrite(motorEnA, 255); // Velocidad máxima

    // Configurar servomotores
    servoDir.attach(servoDirPin);
    servoH.attach(servoHPin);
    servoV.attach(servoVPin);

    // Inicializar comunicación I2C
    Wire.begin();

    // Inicializar Pixy2
    // Inicializar Lidar
}

void loop() {
    // Leer datos de la Pixy2
    // Leer datos del Lidar
    // Procesar datos y tomar decisiones

    // Ejemplo: evitar obstáculos
    evitarObstaculos();

    // Completar tres vueltas
    completarVueltas();

    // Buscar y estacionarse en el bloque de color magenta
    estacionarseEnMagenta();
}

void evitarObstaculos() {
    // Lógica para evitar obstáculos usando Lidar y Pixy2
}

void completarVueltas() {
```

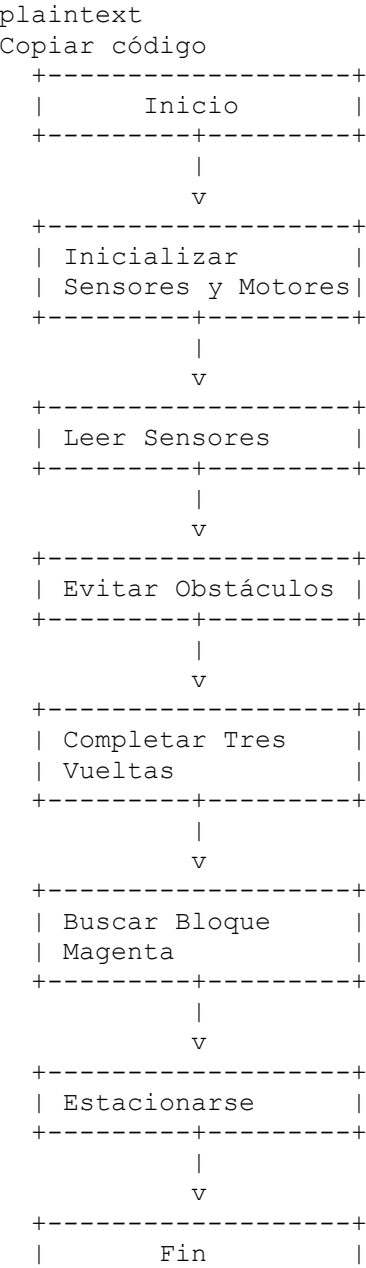
```
// Lógica para completar tres vueltas
}

void estacionarseEnMagenta() {
    // Lógica para buscar y estacionarse en el bloque de color magenta
}
```

Diagrama de Flujo

- 1. Inicio
- 2. Inicializar Sensores y Motores
- 3. Leer Sensores
- 4. Evitar Obstáculos
 - o Si hay un obstáculo, ajustar dirección y velocidad.
- 5. Completar Tres Vueltas
 - o Si se completaron tres vueltas, proceder al siguiente paso.
- 6. Buscar Bloque Magenta
 - o Si se encuentra el bloque magenta, proceder al estacionamiento.
- 7. Estacionarse
- 8. Fin

Diagrama de Flujo Gráfico

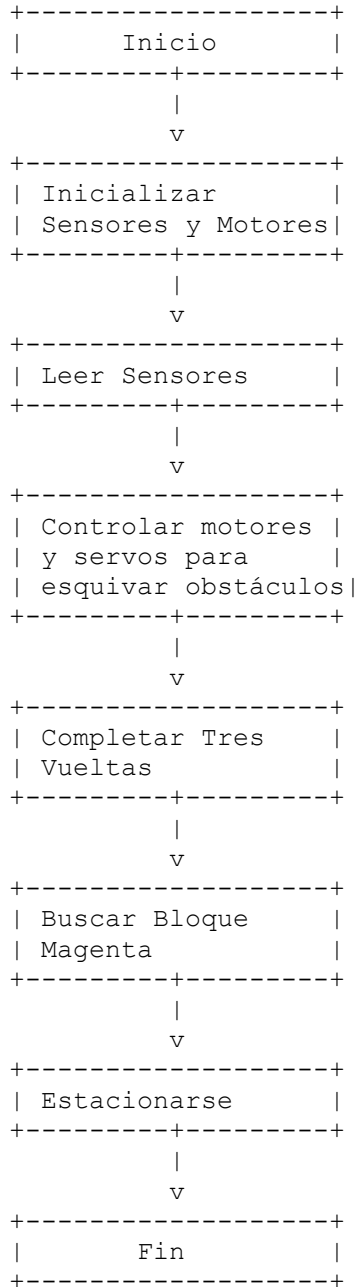


Con este detalle, tienes el esquema de conexiones, una base para la programación, y un diagrama de flujo que te guiará en la implementación del proyecto.

Diagrama de Flujo

plaintext

Copiar código



Este esquema y la programación te guiarán para ensamblar tu proyecto de auto autónomo que cumple con los requisitos de esquivar obstáculos, completar tres vueltas dentro de un cuadrado de 3 metros y estacionarse en un bloque de color magenta.

CONCLUSIÓN:

Hemos concluido que el uso de esta herramienta hasta este punto nos ha ahorrado tiempo y sacado de dudas con algunos de los procesos que deseábamos implementar en el proyecto, pero se evidencia que se necesita tener conocimientos previos de lo que se consulta, como programación y electrónica al menos básica ya que se muestran errores o si no se le da la información necesaria puede que no se consiga lo deseado al consultar a la IA.

Nos gustaría destacar que las consultas que se hicieron surgían de errores encontrados por nuestra parte en base a lo que necesitábamos como respuesta.