**Ministerio de Educación**

**Instituto Profesional y Técnico**

**JEPTHA B. DUNCAN**

**Grupo:**

**Andrea Moreno**

**Cristian Villalba**

**Rafael Pelletier**

**Tema:**

**Robotica**

**Carrito**

**Año Lectivo**

**2024**

**Introduction**

Armar un carrito utilizando Arduino puede ser un proyecto emocionante y educativo. El procedimiento general para construir y programar un carrito incluye varias etapas clave: planificación, ensamblaje de hardware, y programación. A continuación, se presenta una introducción sobre cada uno de estos pasos y cómo se integran los códigos en Arduino.

**Procedimiento de Construcción del Carrito**

1. **Planificación y Diseño**

El primer paso en la creación del carrito es la planificación. Esto incluye definir las funcionalidades del carrito, seleccionar los componentes necesarios y diseñar un esquema de cómo se conectarán estos componentes. Entre los elementos básicos se encuentran el chasis, los motores, las ruedas, una placa Arduino, una batería y sensores opcionales.

1. **Recolección de Componentes:**

**Chasis**

El chasis es la estructura física del carrito. Sirve como soporte para todos los componentes y mantiene todo en su lugar de manera segura y estable.

* Importancia:

Proporciona la base sobre la cual se montan los motores, la placa Arduino y otros componentes.

Garantiza que el carrito mantenga su forma y funcionalidad durante el movimiento.

Permite la distribución equilibrada del peso, lo cual es crucial para la estabilidad y el rendimiento del carrito.

**Motor**

El Motor es un dispositivo que proporciona el movimiento al carrito. En la mayoría de los casos, se utilizan motores DC (corriente continua) o servomotores. La función del motor es convertir la energía eléctrica en movimiento mecánico, permitiendo que el carrito avance, retroceda o gire. El motor se conecta a las ruedas del carrito para transmitirles el movimiento.

**Placa Arduino**

La placa Arduino es el cerebro del carrito. Es una microcontroladora que se programa para gestionar las operaciones del carrito. Recibe entradas de los sensores y, basándose en el código programado, controla los motores y otros actuadores. Es la pieza central que coordina todas las funciones del carrito mediante el procesamiento de señales y ejecución de instrucciones.

**Controlador de Motor L293D**

El controlador de motor L293D es un puente H que permite controlar la dirección y velocidad de los motores DC. Debido a que los pines de Arduino no pueden proporcionar suficiente corriente para los motores, el L293D actúa como un intermediario. Recibe señales de baja corriente desde el Arduino y las amplifica para controlar los motores. Este controlador permite cambiar la polaridad de los motores, lo que es necesario para mover el carrito hacia adelante y hacia atrás.

**Batería**

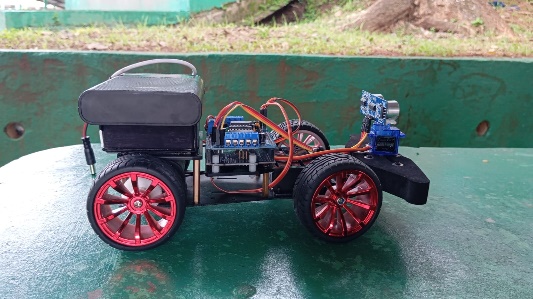
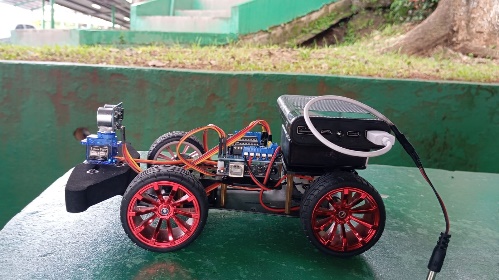
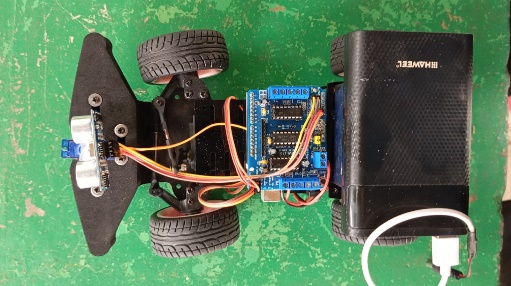
La batería proporciona la energía necesaria para que todos los componentes del carrito funcionen. Debe suministrar el voltaje y la corriente adecuados tanto para los motores como para la placa Arduino y otros componentes electrónicos. Una batería comúnmente usada en estos proyectos es una de 9V o un pack de baterías recargables, que ofrece una buena duración y rendimiento.

**Sensor**

Los sensores permiten que el carrito interactúe con su entorno. En este caso usamos el Sensor Ultrasónico.

* Sensore Ultrasónico: Utilizados para medir distancias y detectar obstáculos, permitiendo al carrito evitar colisiones.

**Fotos del Carrito**



**Programación en Arduino**

#include <AFMotor.h>

#include <Servo.h>

// Pines para el sensor ultrasónico

const int trigPin = A0;

const int echoPin = A1;

// Inicializar el motor y el servo

AF\_DCMotor motor(1); // Motor conectado al M1

Servo steeringServo;

// Constantes para la lógica del carro

const int distanceThreshold = 30; // Distancia mínima para cambiar dirección en cm

const int turnAngle = 45; // Ángulo de giro del servo motor en grados

const int initialAngle = 90; // Ángulo inicial del servo motor (centro)

void setup() {

Serial.begin(9600);

// Configurar los pines del sensor ultrasónico

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

// Inicializar el servo

steeringServo.attach(9); // Pin del servo

steeringServo.write(initialAngle); // Posición inicial del servo (centro)

// Inicializar el motor

motor.setSpeed(255); // Establecer la velocidad máxima del motor

motor.run(BACKWARD); // Avanzar el motor

}

void loop() {

// Medir distancia con el sensor ultrasónico

long duration, distance;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration / 2) / 29.1;

Serial.print("Distancia: ");

Serial.println(distance);

if (distance < distanceThreshold) {

// Si hay un obstáculo, detener el motor y girar el servo

motor.run(RELEASE); // Detener el motor

// Girar el servo motor para evitar el obstáculo

steeringServo.write(initialAngle + turnAngle); // Girar el servo

delay(500); // Esperar para el giro

// Regresar el servo motor a la posición inicial

steeringServo.write(initialAngle);

delay(500); // Esperar para estabilizar

// Reanudar el movimiento después de ajustar el servo

motor.run(BACKWARD); // Avanzar el motor

} else {

// Continuar avanzando hacia adelante

motor.run(BACKWARD); // Avanzar el motor

}sggggggggggggggggggggggggggggggggggggggggggggg

delay(100); // Pausa para estabilidad

}