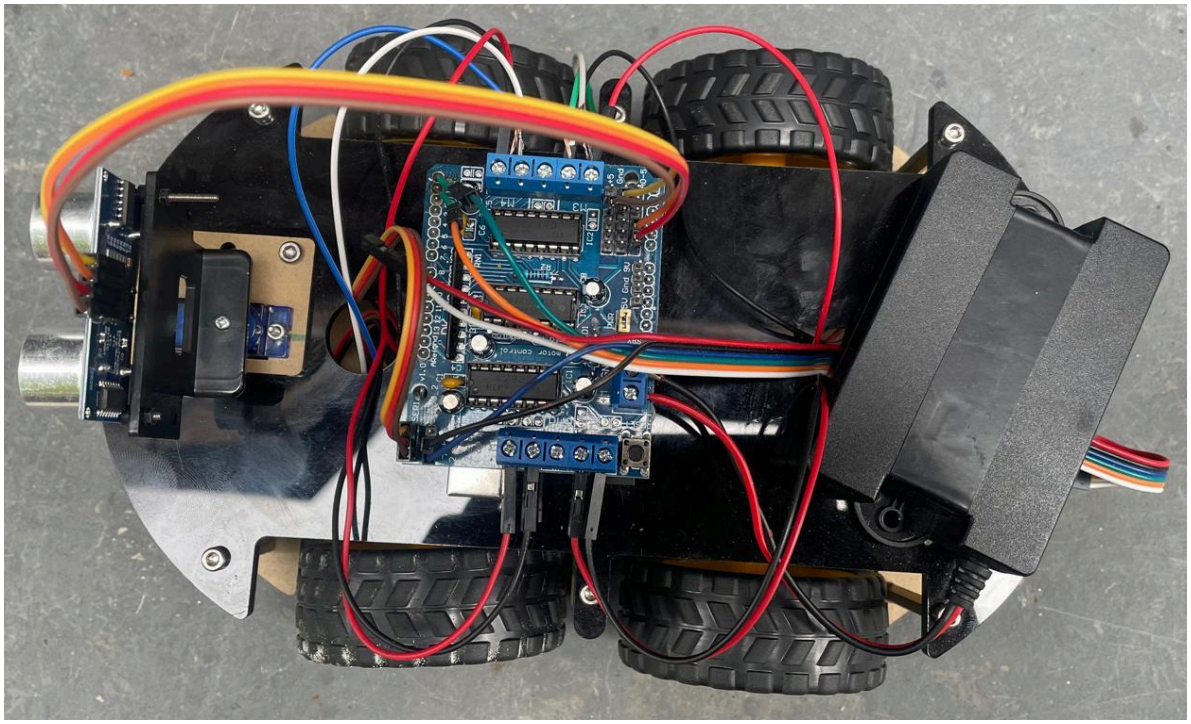
Project Title: Arduino Bluetooth/Serial Monitor Controlled Car



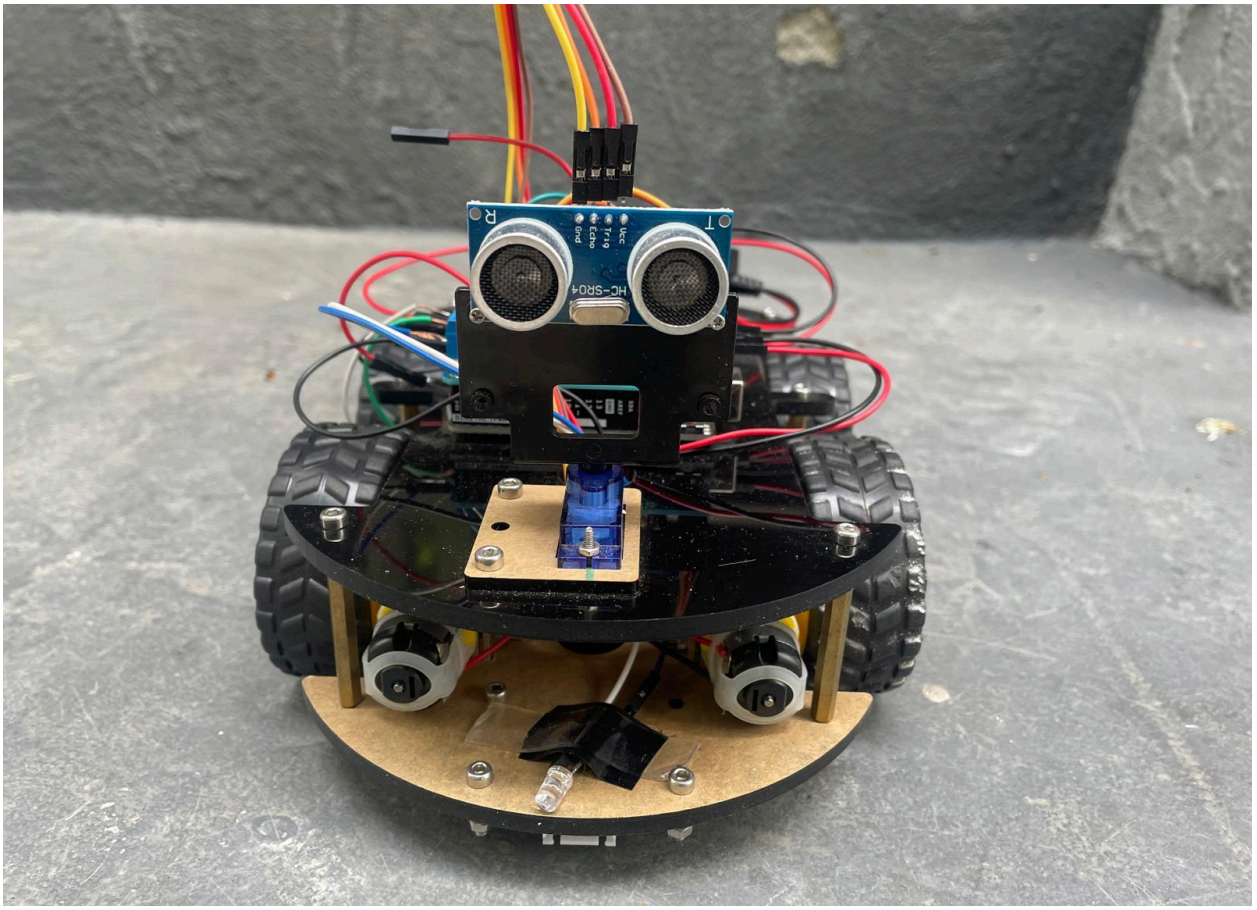
Author: B Hydara

Department: Dept. of Computer Eng. Tech.

Date: 8/07/2025

Table of contents Page

Arduino Bluetooth/Serial Monitor Controlled Car



Project Summary

This is an Arduino-based car project that is controlled through both a Bluetooth app (HC-05 module) and the Arduino IDE Serial Monitor. The car uses an L293D Motor Driver to control four DC motors for forward, backward, left, and right movement. Commands sent wirelessly through the Bluetooth app or manually through the Serial Monitor control the car's motion. The system also includes an ultrasonic sensor and a servo motor for automatic obstacle detection and avoidance, allowing the car to navigate safely around objects.

Background Information

Arduino Bluetooth-controlled cars have already been built by many students and others. Most projects use an Arduino UNO, an HC-05 Bluetooth module, and an L293D motor driver to control DC motors wirelessly from a smartphone.

The phone sends movement commands (forward, backward, left, right) through Bluetooth. The Arduino receives the data and sends signals to the motor driver, which powers the motors. Some have integrated L298N drivers, sensors, and cameras, but the system structure is mostly the same.

This project will follow the same basic idea from the start to the end

Project Management Plan

Tasks (and Sub-Tasks)	Resources (Person Responsible)	Timeline (Class Sessions)
	A	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1: Project Abstract / Executive Summary	A	
2. Background Research	A	
3. Pre-Design	A	
4. Design (Diagrams)	A	
5. Project Unit Test Programs	A	
6. Project Prototype (Final Code)	A	
7. Project Demo/Presentation/Report	A	

Color Code:
Work completed
Work remaining

The project management plan was followed successfully throughout the project. Most tasks were completed according to the original schedule, including research, design, coding, testing, and final integration. The original project idea remained the same.

One small modification was made during development. Serial Monitor control was added along with Bluetooth app control to make testing and troubleshooting easier.

Some issues such as motor wiring direction, Bluetooth communication, and motor power supply problems were encountered and fixed through rewiring, debugging, and using a dedicated battery supply.

Physical Design & diagrams

Major Components Include:

- **Arduino UNO**: The main microcontroller used to control the entire system.
- **L293D Motor Driver**: Controls the speed and direction of the four DC motors.
- **HC-05 Bluetooth Module**: Enables wireless communication bwn the car and a smartphone.
- **DC Motors**: Four motors used for forward, backward, left, and right movement.
- **Power Supply**: 9V battery and connections to the Arduino and motor driver.
- **Ultrasonic Sensor:** Used for automatic obstacle detection and avoidance.
- **Servo Motor:** Used to rotate the ultrasonic sensor to scan different directions for obstacles.
- **LED:** Used as a status indicator to show power or obstacle detection.
- **Jumper Wires**: Used to connect all electronic components together.

Bill of Materials

Component	Part Number	Quantity	Price ($)
Arduino UNO	A000066	1	22.00
L293D Motor Driver	L293D	1	8.00
HC-05 Bluetooth Module	HC-05	1	10.00
DC Motors	JG-37	4	3.00 each
LEDs	-	40	7.00
Battery Holder	-	1	3.00
Jumper Wires	-	20	5.00
Breadboard	-	1	5.00
Chassis	-	1	10.00
Wheels	-	4	5.00 each
Servo Motor	-	1	10
ultrasonic sensor	-	1	10
Total			**$105.00**

Construction

The physical design of this project consists of a four-wheel robotic car built using an acrylic chassis, four DC motors, Arduino UNO, L293D motor driver shield, ultrasonic sensor, servo motor, HC-05 Bluetooth module, LEDs, and an external battery pack.

Two DC motors are mounted on each side of the chassis to provide forward, backward, left, and right movement. The motors are controlled using the L293D motor driver shield mounted on top of the Arduino UNO.

The ultrasonic sensor is placed at the front of the car on a servo motor bracket. The servo motor rotates the sensor left and right to detect obstacles and help the car choose a clear path. The HC-05 Bluetooth module allows wireless control using a mobile app, while the Arduino IDE Serial Monitor can also be used for testing and movement control.

The car includes two LEDs: one front LED and one rear LED for visual indication. A rechargeable external battery pack with an ON/OFF switch is mounted on the upper platform to power the system.

The final design is compact, stable, and supports both manual control and automatic obstacle avoidance.

Problems Encountered During Construction

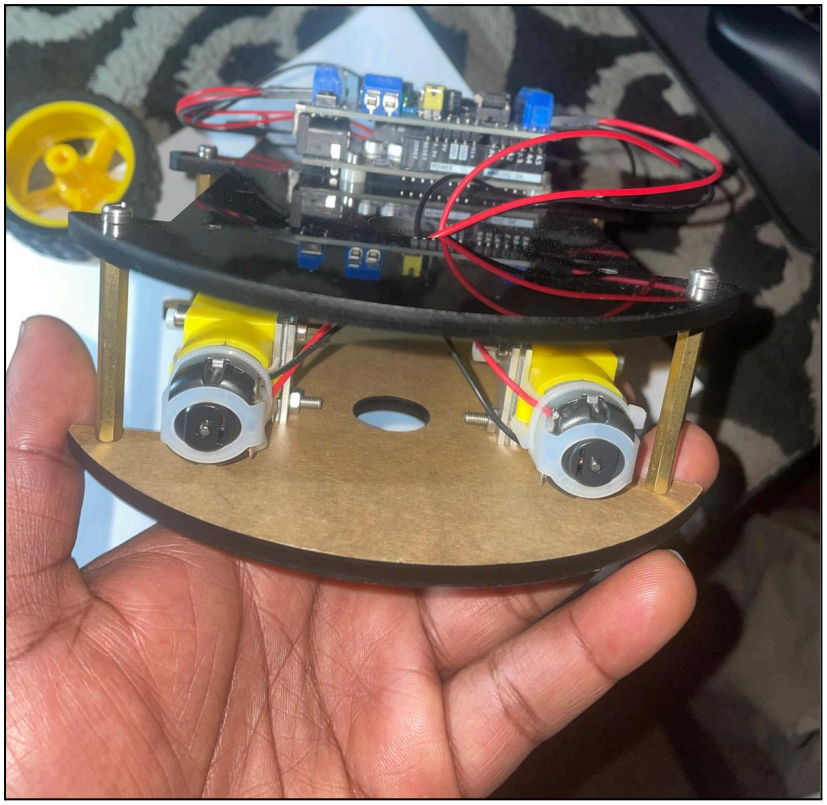
- Incorrect motor wiring caused wrong movement directions
- Motors did not receive enough power from the Arduino UNO
- Bluetooth module caused upload communication issues
- Limited chassis space made wiring difficult

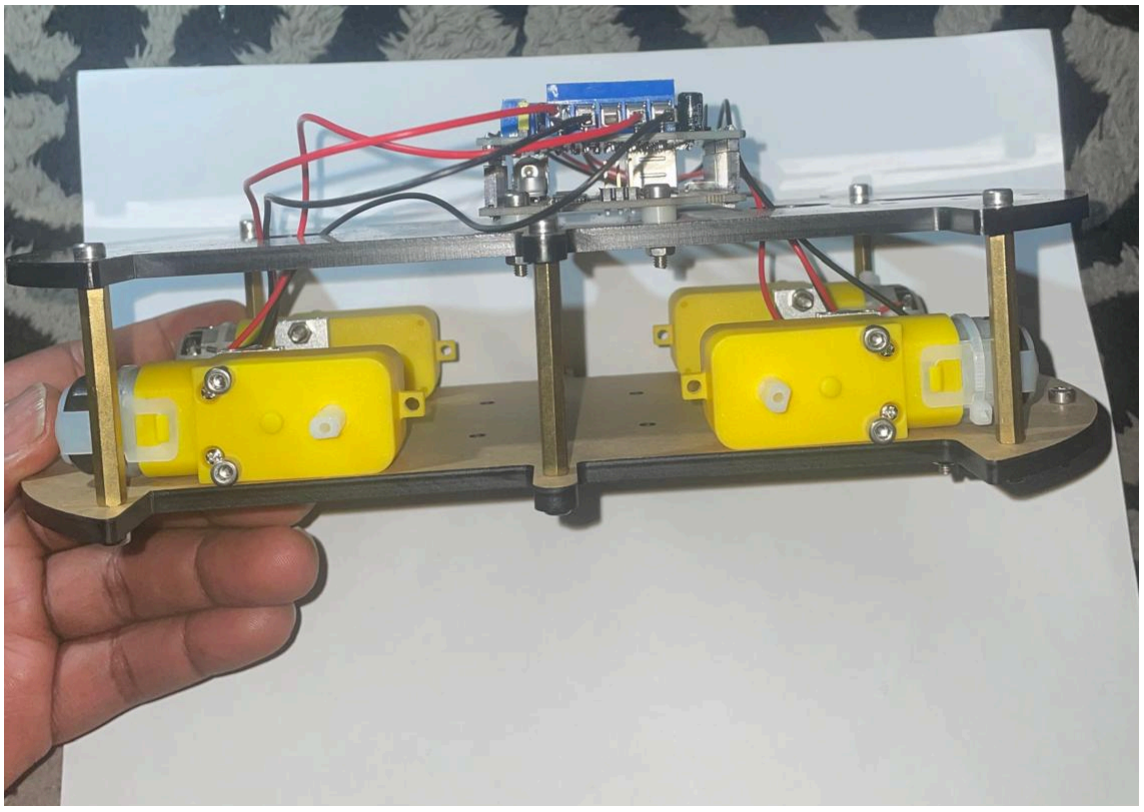
Troubleshooting Steps

- Tested and rewired motors individually until movement directions were correct
- Added a separate rechargeable battery pack for motor power
- Disconnected HC-05 during code uploads and reconnected afterward
- Used Serial Monitor testing to debug motors, sensors, and servo movement

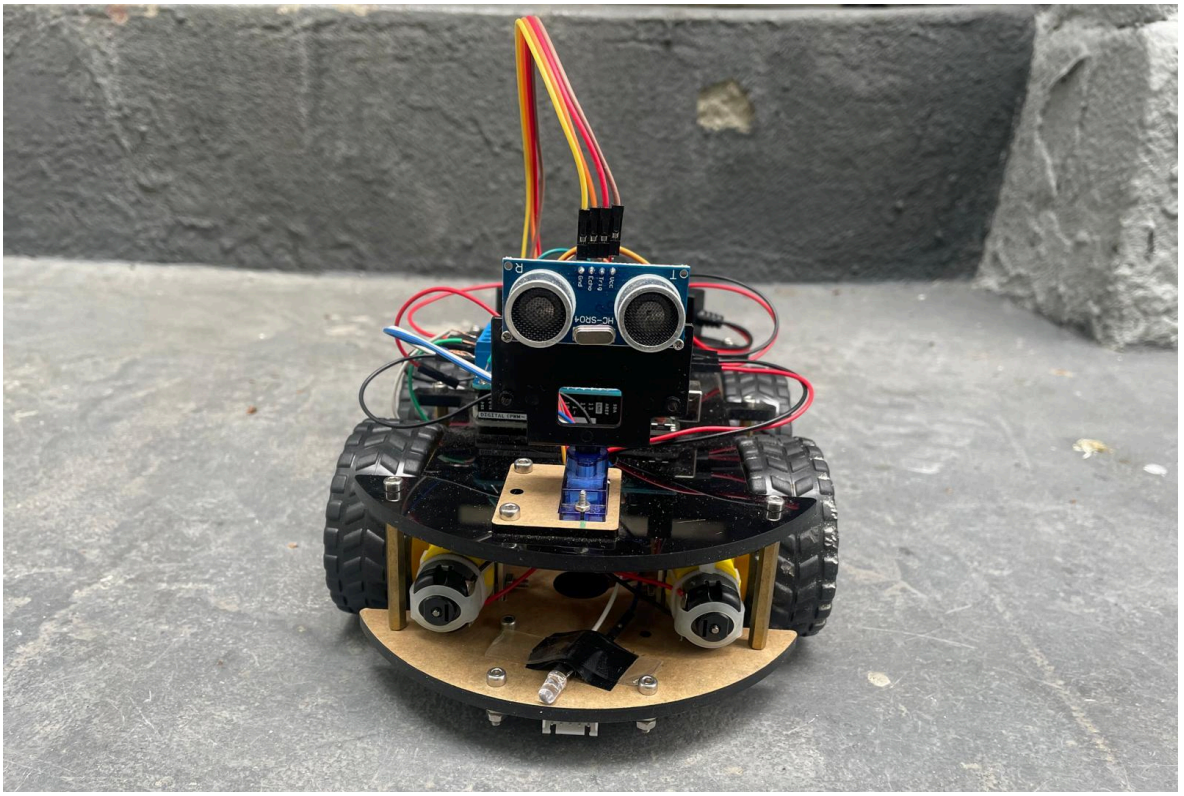
Pictures Included:

- **intermediate stages of the robotic car construction.**

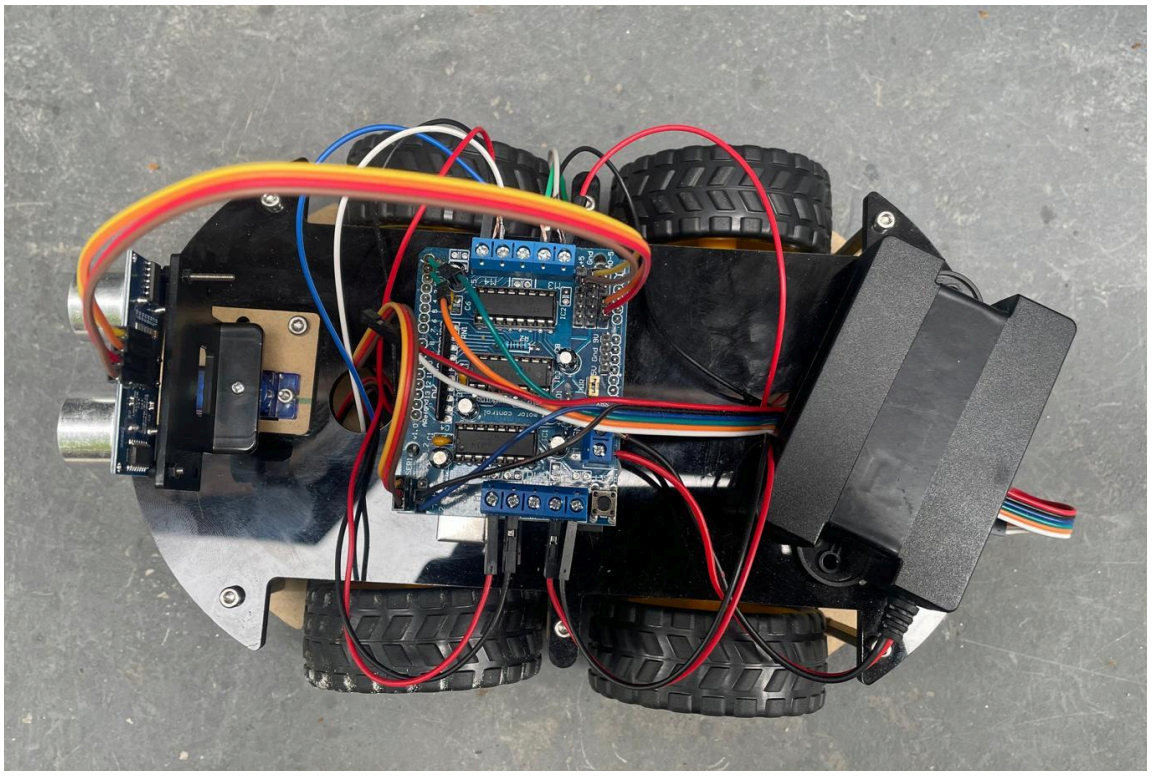




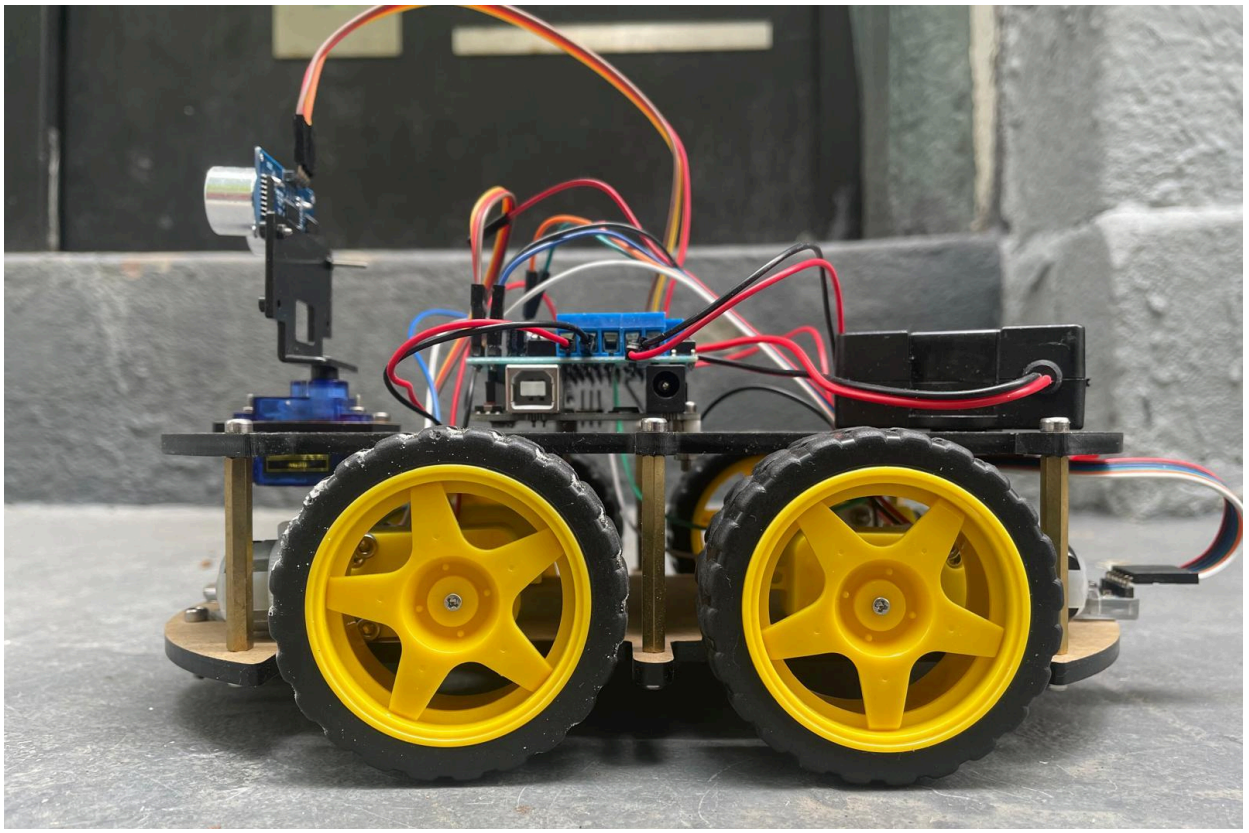
- **Front view of robotic car**



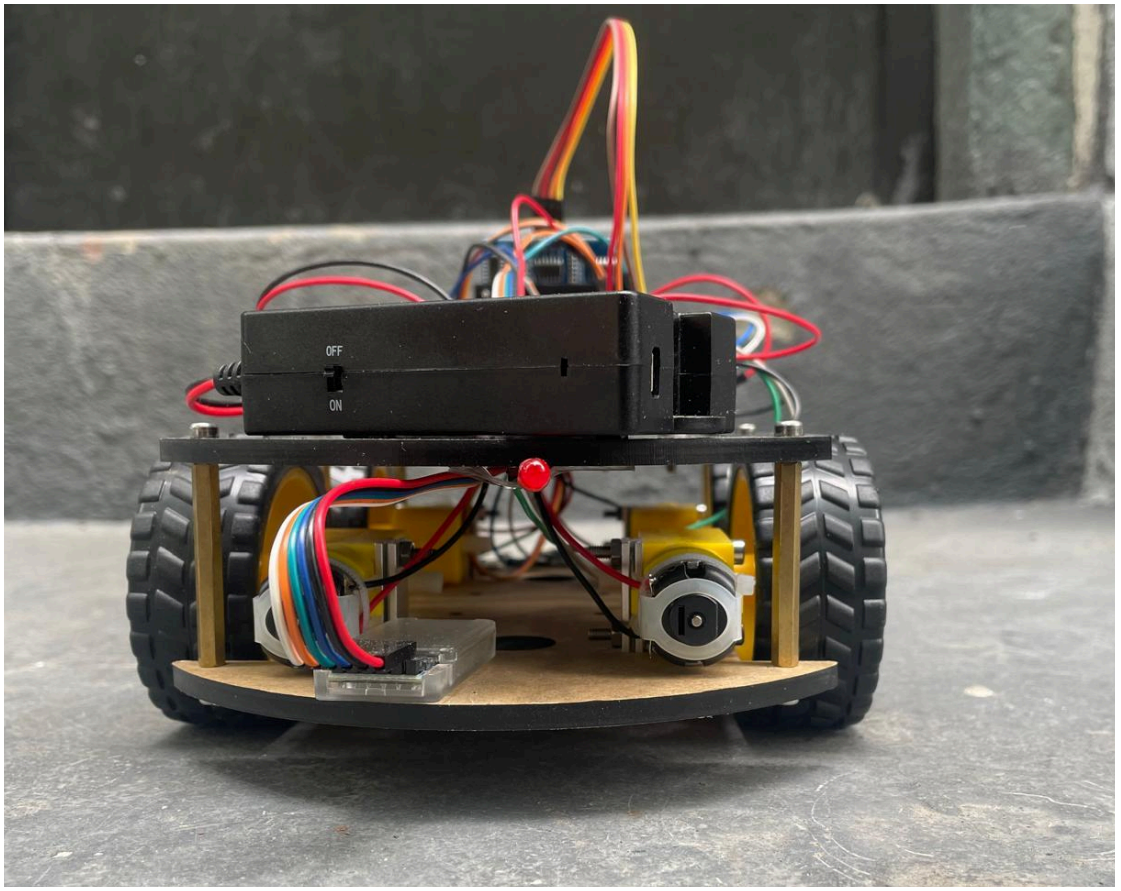
- **Top view of robotic car**



- **Side view of robotic car**

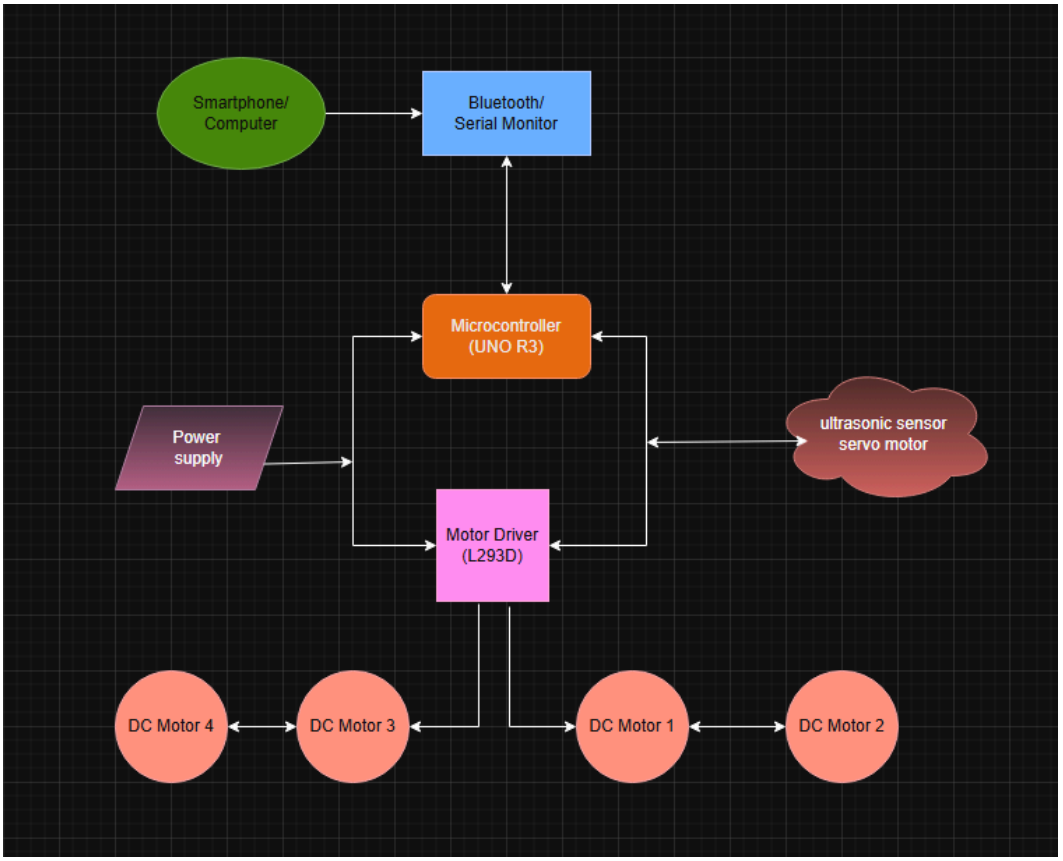


- **Back view of robotic car**

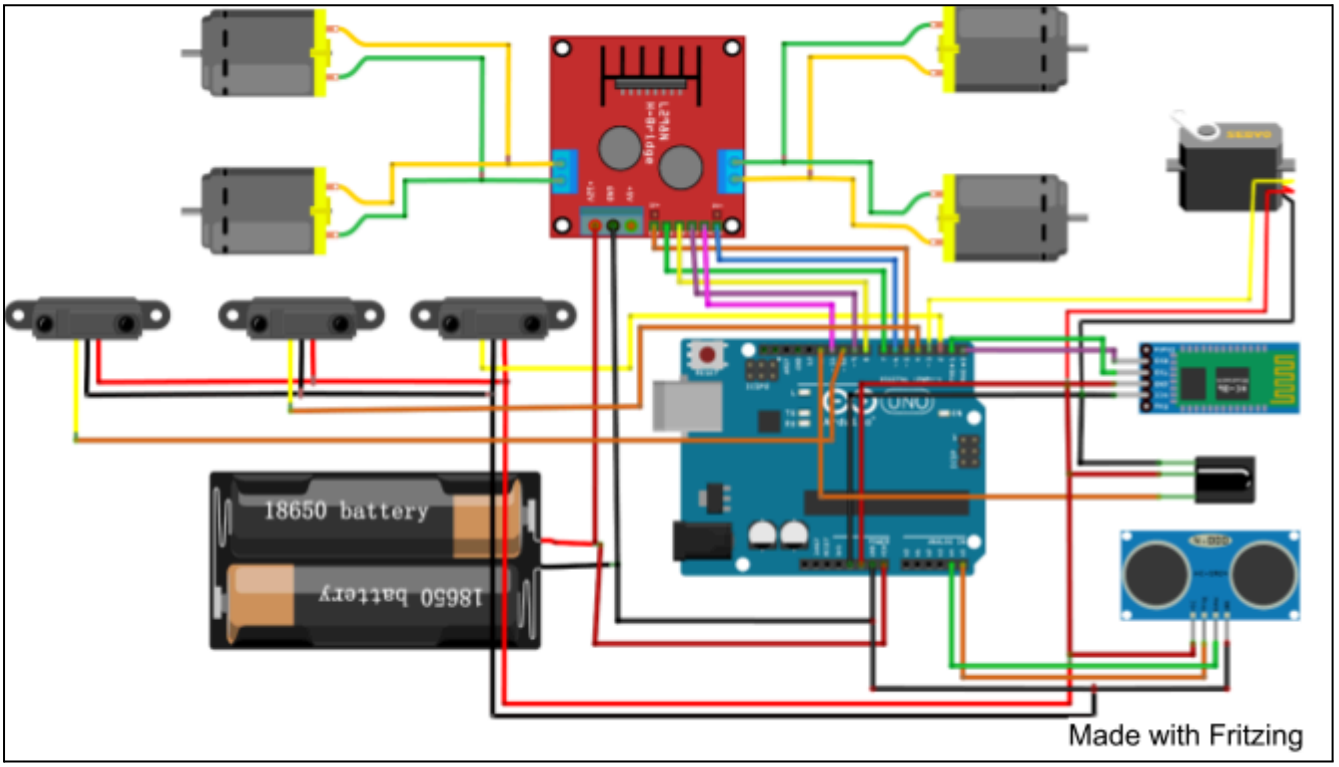


Electrical Design & diagrams

Block Diagram



Schematic Diagram (Made with Fritzing)



Description | Theory of Operation | Datasheets

HC-SR04 Ultrasonic Sensor:

The ultrasonic sensor works by sending out a short burst of high-frequency sound waves (ultrasonic waves) and then listening for the echo that comes back after hitting an object. The Arduino sends a trigger pulse, and the sensor emits 8 cycles of 40 kHz sound. The sensor then sets the echo pin HIGH and waits until the sound reflects back. When the echo is received, the pin goes LOW. The time the echo pin stays HIGH represents the travel time of the sound wave. The Arduino measures this time and calculates the distance using the speed of sound.

- Operating Voltage: 3.3v ~ 5V DC
- Operating Current: less than 2 mA (static)
- Operating Frequency: 40 kHz
- Minimum Distance: about 2 cm
- Maximum Distance: up to 450 cm (about 4.5 m)
- Detection Angle: about 15°
- Sound Pressure: 112 dB
- Connector: 4-pin header with 2.54mm pitch
- VCC: 3.3v ~ 5V
- TRIG: Triggering Input Pin. 10uS TTL Pulses
- ECHO: TTL Logic Output Pin. Proportional to distance
- GND: Ground Pin

Arduino UNO R3:

Pin	Function	Type	Description
1	NC	NC	Not connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog/GPIO	Analog input 0 / GPIO
10	A1	Analog/GPIO	Analog input 1 / GPIO
11	A2	Analog/GPIO	Analog input 2 / GPIO
12	A3	Analog/GPIO	Analog input 3 / GPIO
13	A4/SDA	Analog input/I2C	Analog input 4/I2C Data line

14	A5/SCL	Analog input/I2C	Analog input 5/I2C Clock line
-	-	DC Current per I/O Pin	20 mA
	-	Input Voltage (Nominal)	7–12V
-	-	EEPROM, Clock	1 KB, 16 MHz
-		Flash, RAM	32 KB, 2 KB

For more **Data Sheets Excerpts,** see the link below

UNO R3 | Arduino Documentation

PDF document

L293D Motor Driver:

Description

The L293 and L293D are quadruple high-current half-H driver ICs used to control motors and other inductive loads. They support bidirectional motor control with operating voltages from 4.5 V to 36 V. The L293 can deliver up to 1 A per channel, while the L293D supports up to 600 mA per channel and includes internal protection diodes.

Features

- Wide supply voltage range: 4.5 V to 36 V
- Separate input logic supply
- Internal ESD protection
- High-noise-immunity inputs
- Output current up to 1 A per channel (600 mA for L293D)
- Peak output current up to 2 A per channel (1.2 A for L293D)
- Output clamp diodes for transient suppression (L293D)

Applications

- Stepper motor drivers
- DC motor drivers
- Latching relay drivers

For more info, see this link

LED (Light Emitting Diode):

- Has two terminals:
  - Anode (+, long leg)
  - Cathode (−, short leg)
- Works only in one direction
- Requires a forward voltage to turn ON
- Produces light when current flows

Theory of operation:

When voltage is applied in the correct direction:

- Current flows from anode to cathode
- Electrons recombine inside the semiconductor
- Energy is released as light

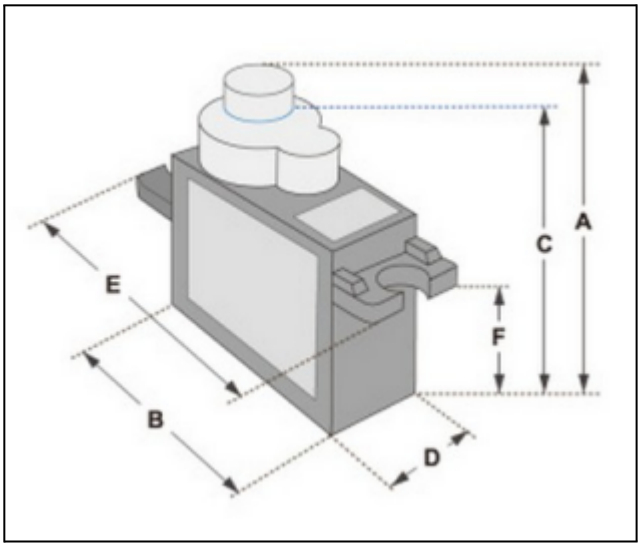
If connected backward, no current flows, so LED stays OFF.

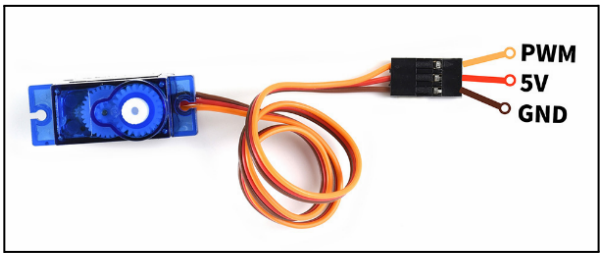
Practical tips:

- Always use a series resistor (220Ω–330Ω) to limit current
- Without a resistor to LED or Arduino pin can be damaged
- Arduino output pin gives about 5V, so resistor is needed to reduce current
- Typical safe current ≈ 20 mA or less

SG90 Servo Motor:

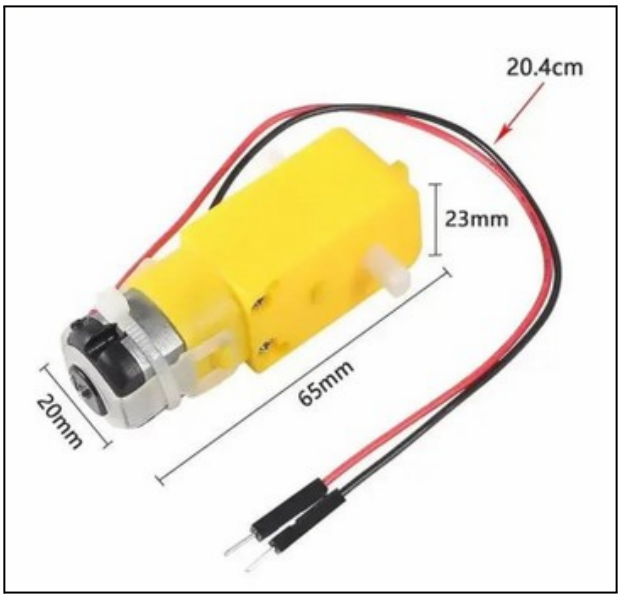
Dimensions & Specifications	Value
A (mm)	32
B (mm)	23
C (mm)	28.5
D (mm)	12
E (mm)	32
F (mm)	19.5
Speed (sec)	0.1
Torque (kg-cm)	2.5
Weight (g)	14.7
Voltage	4.8 – 6 V
Range	180°
Pulse Width	500-2400 µs





DC Motor: Small, low-power motors suitable for a lightweight car chassis.

Technical Details	Value
Rated Voltage	3–6 V
Continuous No-Load Current	150 mA ± 10%
Min. Operating Speed (3V)	90 ± 10% RPM
Min. Operating Speed (6V)	200 ± 10% RPM
Torque	0.15 Nm – 0.60 Nm
Stall Torque (6V)	0.8 kg.cm
Gear Ratio	1:48
Body Dimensions	70 × 22 × 18 mm
Wires Length	200 mm & 28 AWG
Weight	30.6 g
Product Weight	30.6 g / 1.1 oz



General Specification

Bearings	Sintered
Brushes	Carbon
Rotation	Bi-directional
Number of poles	3
Operating voltage	12 V to 30 V
Nominal resistance	124.5 Ω
Nominal inductance	106.7 mH
Typical detent torque	1 mNm
Suppression	Internal ring varistor
Weight (D2431-071200)	47 g

Shaft length options 'L1','L2'

D2431-071200	20.5mm, 52.6mm
D2431-071200S	11.5mm, 43.6mm

For more info, [see this link]

HC-05 Bluetooth module

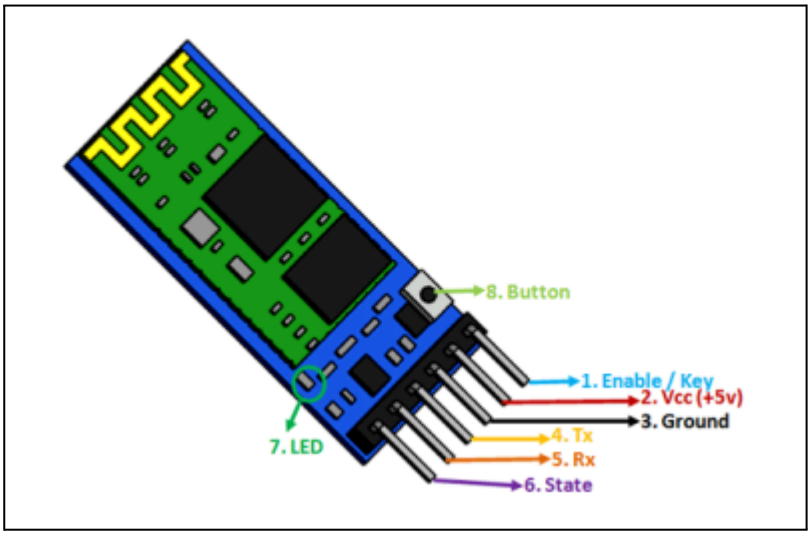
The HC-05 Bluetooth module has two modes: Data Mode for sending and receiving data, and AT Command Mode for changing settings like the device name and password. The mode is selected using the Key pin.

To connect the module, supply +5V power, connect the module's Rx pin to the microcontroller's Tx pin, and connect the module's Tx pin to the microcontroller's Rx pin.

If the Key pin is grounded during startup, the module enters AT Command Mode. If left unconnected, it enters Data Mode automatically. The device appears as "HC-05" and can be paired using the default password "1234."

Bluetooth (HC-05 module) and Arduino Connection

Arduino Uno	HC-05 Module
Pin 0 Rx	Tx
Pin 1 Tx	Rx
GND	GND
+5V	+5V



Pin Number	Pin Name	Description
1	Enable / Key	Used to toggle between Data Mode (set low) and AT Command Mode (set high). By default, it is in Data Mode.
2	Vcc	Powers the module. Connect to +5V supply voltage.
3	Ground	Ground pin of module, connected to system ground.

4	TX – Transmitter	Transmits serial data. Everything received via Bluetooth is given out by this pin as serial data.
5	RX – Receiver	Receives serial data. Every serial data given to this pin is broadcasted via Bluetooth.
6	State	Connected to onboard LED. Used as feedback to check if Bluetooth is working properly.
7	LED	Indicates the status of the module.
8	Button	Used to control the Key/Enable pin to toggle between Data Mode and Command Mode.

For more info, see this link

Data communication and network interface

UART (Universal Asynchronous Receiver Transmitter)

- UART is the internal communication circuit inside the Arduino microcontroller.
- It is used for serial communication between Arduino and PC.

Key specifications:

- Asynchronous communication (no clock signal needed)
- Uses two lines:
  - TX (Transmit)
  - RX (Receive)
- Full duplex communication (send and receive at the same time)
- Baud rate used in this lab: **9600 bits per second**

How it works:

- Data is sent one bit at a time
- Each data frame includes:
  - Start bit
  - Data bits (usually 8 bits)
  - Stop bit

USB (Universal Serial Bus)

- USB is the external communication interface between Arduino and PC.
- It connects using a USB cable.

Key features:

- Converts USB signals to UART signals
- Provides both **data communication and power supply**
- Allows Arduino IDE Serial Monitor to display data

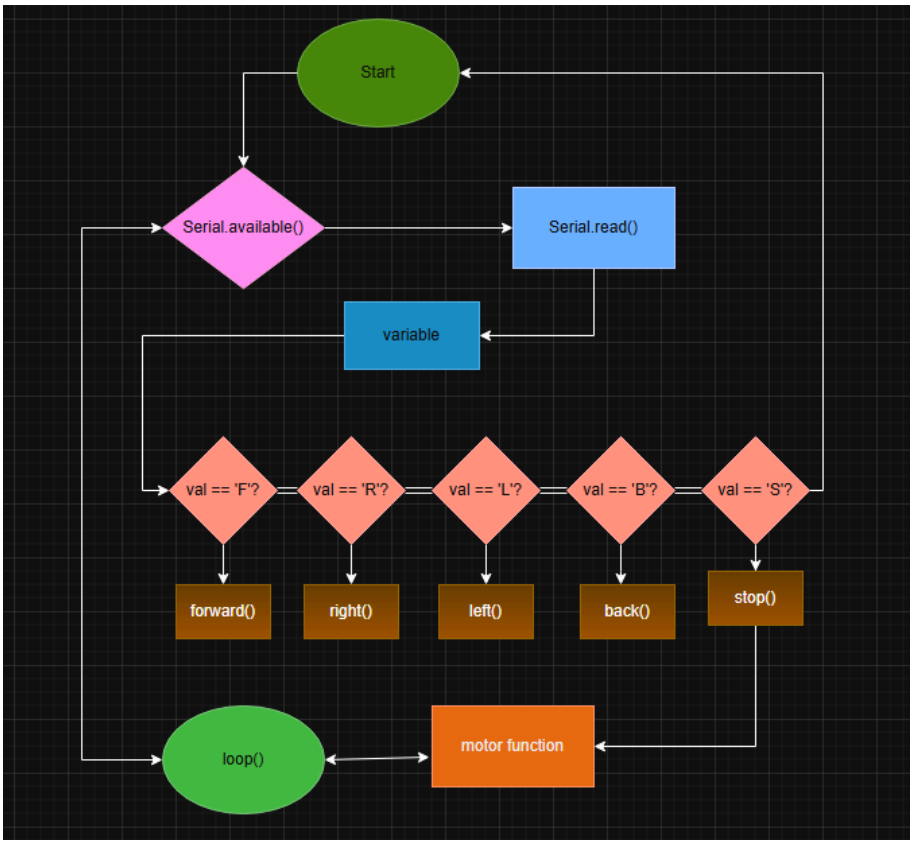
Power Supply System

The project uses a rechargeable external battery pack to provide dedicated power for the DC motors and motor driver shield. The Arduino UNO is powered separately through the USB connection during programming and testing. A common ground connection is shared between the Arduino UNO, motor driver shield, Bluetooth module, ultrasonic sensor, and servo motor to ensure proper operation of the entire system.

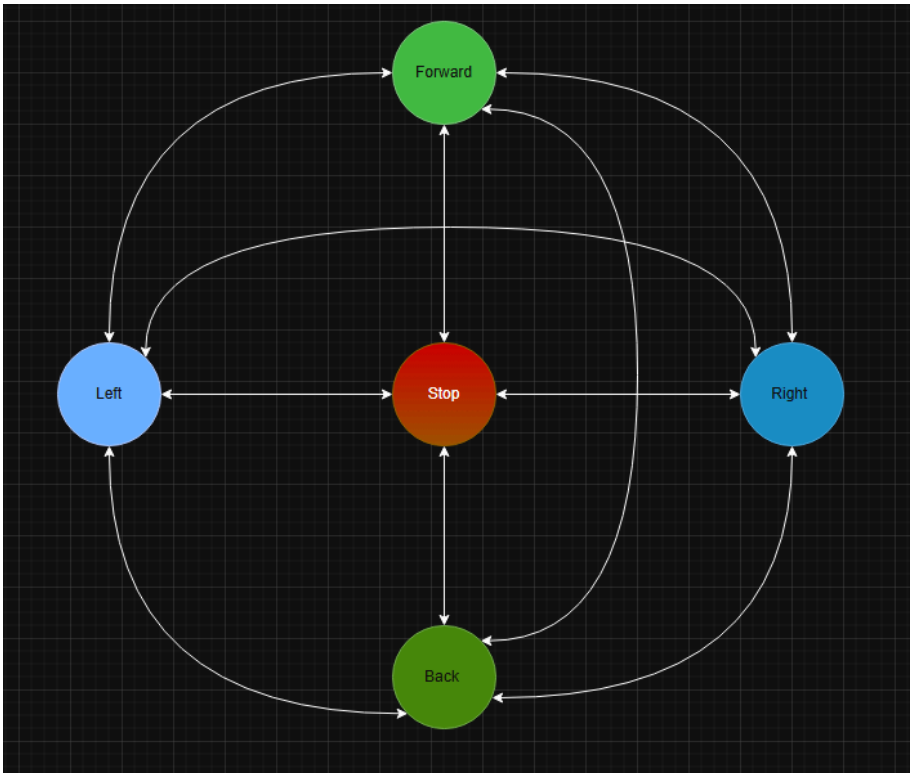
The rechargeable battery pack includes an ON/OFF switch for safe power control and provides enough current for stable motor movement and obstacle avoidance operation.

Software Design

Software flowchart



State-diagram



Software testing of project sub-systems (unit tests).

// Testing LEDs

```
#define BUTTON_PIN 2
#define LED_PIN 9

void setup() {
  pinMode(BUTTON_PIN, INPUT_PULLUP);  // button input
  pinMode(LED_PIN, OUTPUT);           // LED output
  Serial.begin(9600);

  Serial.println("Test Program for Digital Input/Output");
}

void loop() {
  int buttonState = digitalRead(BUTTON_PIN);

  if (buttonState == LOW) { // button pressed
    digitalWrite(LED_PIN, HIGH);
    Serial.println("Button Pressed... LED ON");
  } else {
    digitalWrite(LED_PIN, LOW);
    Serial.println("Button Released... LED OFF");
  }

  delay(500);
}
```

---

source code

```
/*
  Arduino Bluetooth/Serial Monitor Controlled Car

  This project controls a 4-wheel Arduino car using both
  a Bluetooth mobile app (HC-05) and the Arduino IDE Serial Monitor.

  The system uses:
  - L293D Motor Driver Shield
  - 4 DC Motors
  - HC-05 Bluetooth Module
  - Ultrasonic Sensor
  - Servo Motor
```

```
  Features:
  - Forward, backward, left, and right movement
  - Bluetooth and Serial Monitor control
  - Automatic obstacle detection and avoidance

Created by Balagie Hydara
You can watch the video here: https://youtu.be/1xQAdOWvVt4
*/


#include <AFMotor.h>
#include <NewPing.h>
#include <Servo.h>

AF_DCMotor motor1(1, MOTOR12_1KHZ);
AF_DCMotor motor2(2, MOTOR12_1KHZ);
AF_DCMotor motor3(3, MOTOR34_1KHZ);
AF_DCMotor motor4(4, MOTOR34_1KHZ);
Servo myservo;

int valSpeed = 255;

// Ultrasonic
#define TRIG_PIN A0
#define ECHO_PIN A1
#define MAX_DISTANCE 200
NewPing sonar(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);

int stopDistance = 20;

// movement state
char currentMode = 'S';

void setup() {
  Serial.begin(9600);

  myservo.attach(10);
  myservo.write(115);
  delay(500);

  SetSpeed(valSpeed);
  stopMotors();

  Serial.println("f=forward auto avoid, b=back, l=left, r=right, s=stop");
}

void loop() {
```

```
  // read command from Serial Monitor
  if (Serial.available() > 0) {
    char cmd = Serial.read();

    if (cmd == 'f' || cmd == 'F') {
      currentMode = 'F';
      Serial.println("Forward with auto obstacle avoidance");
    }
    else if (cmd == 'b' || cmd == 'B') {
      currentMode = 'B';
      moveBackward();
      Serial.println("Backward");
    }
    else if (cmd == 'l' || cmd == 'L') {
      currentMode = 'L';
      turnLeft();
      Serial.println("Left");
    }
    else if (cmd == 'r' || cmd == 'R') {
      currentMode = 'R';
      turnRight();
      Serial.println("Right");
    }
    else if (cmd == 's' || cmd == 'S') {
      currentMode = 'S';
      stopMotors();
      Serial.println("Stop");
    }
  }

  // automatic obstacle detection only when going forward
  if (currentMode == 'F') {
    int d = readDistance();

    if (d > stopDistance) {
      moveForward();
    } else {
      obstacleAvoid();
    }
  }
}

void SetSpeed(int val) {
  valSpeed = val;
  motor1.setSpeed(val);
  motor2.setSpeed(val);
  motor3.setSpeed(val);
```

```
  motor4.setSpeed(val);
}

void moveForward() {
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}

void moveBackward() {
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
}

void turnLeft() {
  motor1.run(FORWARD);
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(FORWARD);
}

void turnRight() {
  motor1.run(BACKWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(BACKWARD);
}

void stopMotors() {
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}

int readPingOnly() {
  delay(50);
  int cm = sonar.ping_cm();
  if (cm == 0) cm = 250;
  return cm;
}

int readDistance() {
  myservo.write(115);
```

```
  delay(200);
  return readPingOnly();
}

int lookLeft() {
  myservo.write(170);
  delay(400);
  int d = readPingOnly();
  myservo.write(115);
  delay(200);
  return d;
}

int lookRight() {
  myservo.write(50);
  delay(400);
  int d = readPingOnly();
  myservo.write(115);
  delay(200);
  return d;
}

void obstacleAvoid() {
  stopMotors();
  delay(200);

  moveBackward();
  delay(400);
  stopMotors();
  delay(200);

  int rightD = lookRight();
  int leftD  = lookLeft();

  if (rightD >= leftD) {
    turnRight();
    delay(500);
  } else {
    turnLeft();
    delay(500);
  }

  stopMotors();
  delay(200);
}
```

Software Sub-systems: Library Functions

Serial.begin()

- **Syntax:** Serial.begin(9600);
- **Parameters:** baud rate, such as 9600
- **Return value:** None
- **Description:** Starts serial communication for Bluetooth and Serial Monitor.

Serial.available()

- **Syntax:** Serial.available();
- **Parameters:** None
- **Return value:** Number of bytes available
- **Description:** Checks if a command was received.

Serial.read()

- **Syntax:** Serial.read();
- **Parameters:** None
- **Return value:** Incoming character
- **Description:** Reads the command from Bluetooth or Serial Monitor.

motor.setSpeed()

- **Syntax:** motor1.setSpeed(value);
- **Parameters:** speed value from 0–255
- **Return value:** None
- **Description:** Sets the DC motor speed.

motor.run()

- **Syntax:** motor1.run(FORWARD);
- **Parameters:** FORWARD, BACKWARD, or RELEASE
- **Return value:** None
- **Description:** Controls motor direction or stops the motor.

myservo.attach()

- **Syntax:** myservo.attach(pin);
- **Parameters:** servo signal pin
- **Return value:** None
- **Description:** Connects the servo motor to an Arduino pin.

myservo.write()

- **Syntax:** myservo.write(angle);
- **Parameters:** angle from 0–180
- **Return value:** None

- **Description:** Rotates the servo to a specific angle.

sonar.ping_cm()

- **Syntax:** sonar.ping_cm();
- **Parameters:** None
- **Return value:** Distance in centimeters
- **Description:** Measures distance using the ultrasonic sensor.

pinMode()

- **Syntax:** pinMode(pin, mode);
- **Parameters:** pin number and INPUT or OUTPUT
- **Return value:** None
- **Description:** Sets a pin as input or output.

digitalWrite()

- **Syntax:** digitalWrite(pin, value);
- **Parameters:** pin number and HIGH or LOW
- **Return value:** None
- **Description:** Turns LED or buzzer ON/OFF.

Conclusion & Reflection

This project successfully demonstrated the design and implementation of an Arduino-based Bluetooth controlled robotic car with automatic obstacle avoidance. The system was able to control four DC motors using both a Bluetooth mobile application and the Arduino IDE Serial Monitor. The ultrasonic sensor and servo motor successfully detected nearby obstacles and helped the car avoid collisions automatically.

Throughout the project, several hardware and software challenges were encountered, including motor wiring direction problems, Bluetooth communication issues, and motor power limitations. These problems were solved through testing, rewiring, debugging, and using a dedicated external battery supply.

Overall, the project improved understanding of embedded systems, motor control, Bluetooth communication, sensors, and troubleshooting techniques. It also provided practical experience in combining hardware and software into a complete computer-controlled system.

References

- Make DIY. (Jun 1, 2020). *How to make DIY Arduino Bluetooth control car at Home with Arduino UNO, L298N Motor Driver, HC-05* [Video]. YouTube. Retrieved from

  How to make DIY Arduino Bluetooth control car at Home with Arduino UNO, L298N Motor Driv…

- DIY Builder (Mar 23, 2019). *How To Make DIY Arduino Voice Controlled Car At Home* [Video]. YouTube. Retrieved from https://www.youtube.c
- Unknown Author. (n.d.). *Voice Control Car || Smartphone Control Robot Using Arduino* [Video]. YouTube. Retrieved from https://www.youtube.com/watch?v=_____

- Ramji Patel. (Oct 10, 2022). *Arduino Bluetooth Car: 12 Steps*. Retrieved from Arduino Bluetooth Car : 12 Steps (with Pictures) - Instructables

- SriTu Hobby. (Sep 5, 2020). *How to make a Bluetooth control 4WD car | Step by step instructions*. SriTu Hobby. Retrieved from https://srituhobby.com/how-to-make-a-bluetooth-control-4wd-car-step-by-step-instructions/