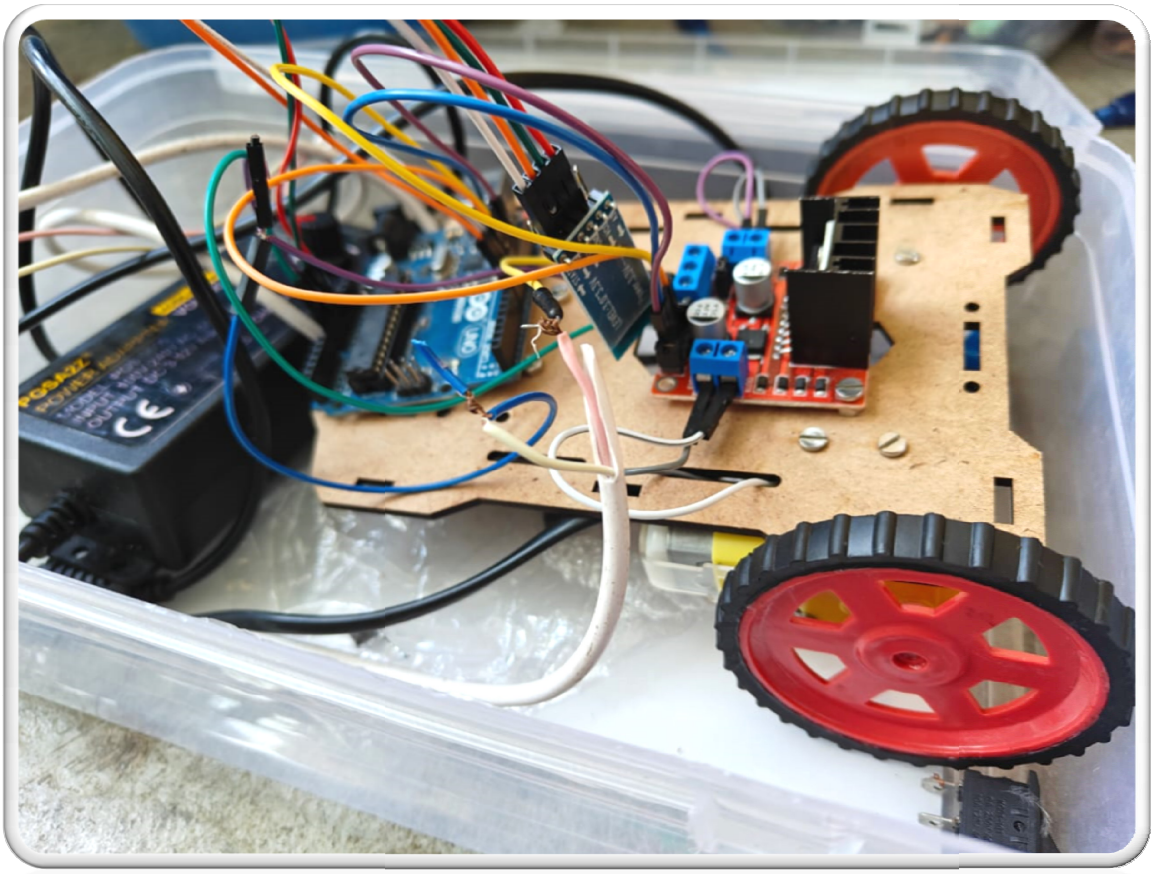


Robot CAR Design Assemble and Coding Development

Chapter 1: Introduction to RC Car Robotics

Overview:
This chapter introduces the basics of RC car construction and its applications in robotics. Students will learn the core components required to build a functional RC car, such as the Arduino Uno board and the motor driver module. We'll also touch on fundamental electronics and how coding enables remote control of the car's movement. students will learn how to design and build their own RC (Remote Controlled) car using fundamental electronic components and coding techniques. The project offers a hands-on approach to understanding how mechanical components, motor drivers, and coding come together to create functional, real-world robotic systems.

Learning Objectives:

- **Arduino Uno Board**: The microcontroller used to control the car.
- **L298N Motor Driver Module**: A key component that helps regulate the motor's speed and direction.
- **Motors**: To power the movement of the RC car.
- **Wheels**, **Battery Pack**, and **Wiring**: To complete the physical structure of the car.
- **Arduino IDE**: The software needed to write and upload code to the Arduino board.

How It Works

The Arduino board acts as the brain of the RC car, receiving commands and controlling the motor driver module to operate the motors. By connecting input pins from the Arduino to the motor driver, students can program the car to move forward, reverse, and make turns. This chapter will guide you through the wiring process and coding essentials to bring your RC car to life.

Circuit Design and Arduino Coding

A major part of the project involves setting up the motor driver and connecting it to the Arduino board. The motor's pins must be connected to the correct Arduino output pins to control the RC car's motion efficiently. Detailed instructions will be provided to ensure each student can complete the circuit successfully. Arduino code will be introduced step by step to make the car operational.

Chapter 2: Required Components

Arduino Uno Board
The Arduino Uno board serves as the main controller, sending signals to the motors and receiving inputs from a remote source.

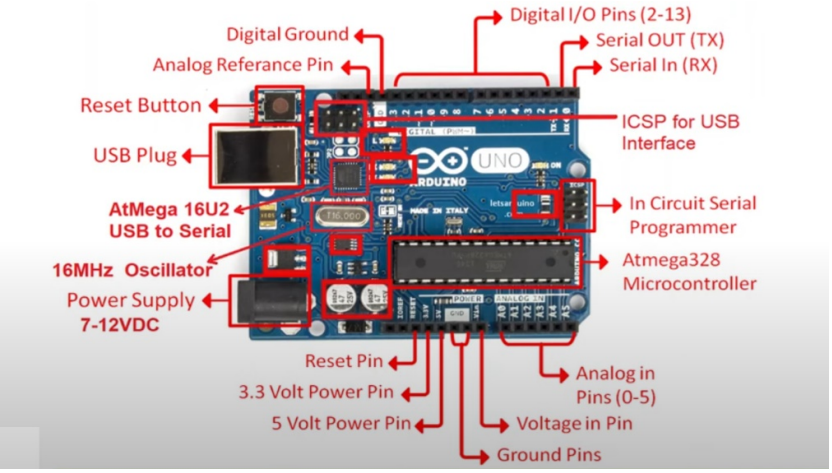
L298N Motor Driver Module
This module allows you to control the speed and direction of the motors. It acts as a bridge between the Arduino and the motors, amplifying the current to the levels required by the motors.

Motors
Two DC motors will drive the car, powered by a battery pack and controlled by the motor driver module.

Wheels, Battery Pack, and Chassis
The physical structure of the car, including wheels for movement and a chassis to hold the components together.

Wires and Connectors
Used to establish electrical connections between the components.

Chapter 3: Pin Configuration and Features of Arduino



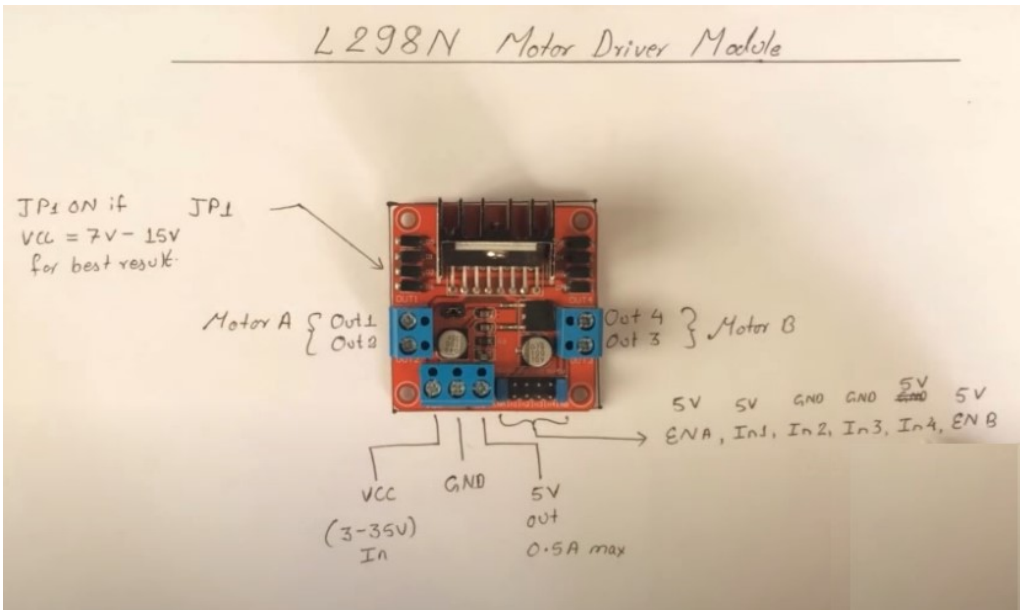
Arduino Pins

- **Digital Pins**: Pins used to control various modules, including motor driver input pins. For example, pins 8 and 9 can be connected to the input pins of the motor driver.
- **Analog Pins**: May be used for sensor input or other additional modules, but in this project, we will mainly focus on digital control.

Power and Ground Pins

- The Arduino requires a power source to function, which can be provided through USB or an external battery pack.

Chapter 4: Understanding the L298N Motor Driver Module



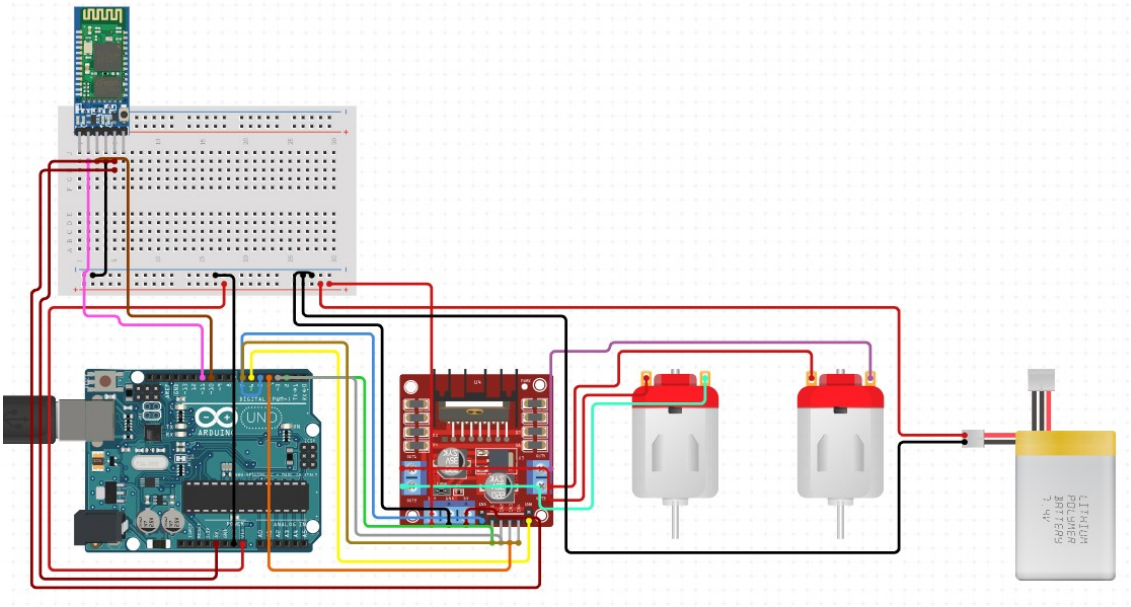
The **L298N Motor Driver** is essential for controlling the direction and speed of the motors.
Key Features:

- **Input Pins (IN1 and IN2)**: Used to receive signals from the Arduino, determining the motor's direction.
- **Enable Pins (ENA and ENB)**: These control the motor speed via PWM (Pulse Width Modulation).

Pin Configuration:

- Connect the **ENA** pin to a PWM pin on the Arduino to control motor speed.
- Connect **IN1** and **IN2** to digital output pins on the Arduino to control motor direction.

Chapter 5: Circuit Design



This chapter provides a step-by-step guide for assembling the RC car's circuit.
Steps:

1. **Connect the Arduino to the Motor Driver**:
    - Connect digital pin 2, pin 3, pin 4, pin 7 on the Arduino to input pin IN1,IN2, IN3, IN4 on the motor driver.
    - Connect analogpin 5, pin 6 to input pin ENA, ENB respectively.
2. **Connect Motors to the Motor Driver**:
    - Attach the first motor's positive and negative terminals to M1 on the motor driver.
    - Attach the second motor's positive and negative terminals to M1 on the motor driver.
3. **Power the Arduino and Motor Driver**:
    - Connect a 9V battery pack to the L298N Module.
    - Connect a 5v to power the Arduino module and bluetooth
    - Ensure proper power is supplied to the motor driver module.
4. **Connect Bluetooth Module (discuss in chapter 7 in details)**
    - Connect digital pin 10, pin 11 on the Arduino to Bluetooth TXD, RXD.
5. **Finalize the Wiring**:
    - Double-check connections to avoid any short circuits or misconfigurations.

Chapter 6: Arduino Coding for RC Car Control

Introduction to Arduino Programming:
In this chapter, you will write code to control the RC car's movement. The car will be able to move forward, backward, and turn based on the signals sent to the motors.

Sample Code

```
 #include<SoftwareSerial.h>

//Bluetooth for Signal Action
SoftwareSerialBluetooth(11, 10);//rxd=>tx, txd=rx
char Data;

//Motor Pin L298N DC
//--------------------

//Input Pin Signal
constint in1=2;
constint in2=3;
constint in3=4;
constint in4=7;

/**
* Jumper Enable A and Enabled B for  Speed Controller
* Value=> 0%=0 , 25%=64, 50%=127, 75%=191, 100% = 255
*/
constintena=5; //enabled A
//const int enb=6; //enabled B

voidsetup(){

  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
  pinMode(in3,OUTPUT);
  pinMode(in4,OUTPUT);
  Serial.begin(9600);//USB communication and serial key id
  Bluetooth.begin(9600);
}

voidloop(){
      if(Bluetooth.available()){
        Data=Bluetooth.read();
        if(Data==('L')){
            //LEFT WHEEL
            digitalWrite(in4, HIGH);
            digitalWrite(in3, LOW);
```

```
            digitalWrite(in2, LOW);
            digitalWrite(in1, LOW);
        }
        elseif(Data == 'R'){
            //RIGHT WHEEL
            digitalWrite(in4, LOW);
            digitalWrite(in3, LOW);
            digitalWrite(in2, HIGH);
            digitalWrite(in1, LOW);

        }
        elseif(Data == 'F'){
          //FORWARD WHEEL
          digitalWrite(in4, LOW);
          digitalWrite(in3, HIGH);
          digitalWrite(in2, LOW);
          digitalWrite(in1, HIGH);
        }
        elseif(Data == 'B'){
          //BACKWARD WHEEL
          digitalWrite(in4, HIGH);
          digitalWrite(in3, LOW);
          digitalWrite(in2, HIGH);
          digitalWrite(in1, LOW);
        }
        elseif(Data==('S')){
          //STOP WHEEL
          digitalWrite(in4, LOW);
          digitalWrite(in3, LOW);
          digitalWrite(in2, LOW);
          digitalWrite(in1, LOW);
        }
        elseif(Data=='T'){
          analogWrite(ena,255); //Turbo
          //analogWrite(enb,255); Turbo
        }
        else{
          //STOP WHEEL
          digitalWrite(in4, LOW);
          digitalWrite(in3, LOW);
          digitalWrite(in2, LOW);
          digitalWrite(in1, LOW);
        }
      }
}
```

Chapter 7: Advanced Control with Remote Inputs

In this chapter, we'll introduce how to control the RC car using wireless inputs, such as an infrared remote or Bluetooth module. This section will involve adding sensors and receivers to the Arduino to capture and respond to remote signals.

Bluetooth Module Setup:

- Connect a Bluetooth module to the Arduino.
- Modify the code to respond to Bluetooth commands for controlling the car remotely.

Bluetooth User Interface

- Install Bluetooth app (bluetoothControl.apk)in your smartphone android

Chapter 7: Testing and Troubleshooting

Once you have assembled the car and uploaded the code, it's time to test your project.
Key Testing Steps:

1. Power on the Arduino and check if the motors respond to the uploaded program.
2. Use a multimeter to ensure proper voltage is being supplied to the motor driver module.
3. If the car does not move as expected, check the wiring and review the code to identify potential errors.

Common Issues:

- **Motor Not Responding**: Check power supply and wiring.
- **Erratic Movement**: Ensure the code is correctly sending signals to the motor driver.

Chapter 9: Final Assembly and Customization

After building and testing your RC car, you can customize it.

- **Add Sensors**: Attach additional sensors to make the car autonomous.
- **Upgrade the Body**: Design a custom chassis or enhance the wheels for different terrains.

Chapter 10: Conclusion and Future Projects

This chapter concludes with insights into how the knowledge gained from building an RC car can be applied to larger robotics and automation projects. Suggestions for further reading and projects include line-following robots, obstacle avoidance, and home automation systems using Arduino.

***~~~ ***