# Voice Controlled Smart Car

**Group 4: VoxRovers**

Aditya Abhilash, 22EC01059

N. Jay Roshan Devankar, 22EC01005

Atharva Tol, 22EC01003

Nihar Ranjan Jena, 22EC01001

## Project summary

This project aimed to build a compact robotic car that can be controlled entirely through voice commands while still maintaining safe navigation using obstacle detection. The idea was to create a simple and intuitive system where the user could say “forward,” “back,” “left,” “right,” or “stop,” and the car would respond instantly—without needing any manual controls. To achieve this, the design combined three key parts: the voice recognition module, the motor control system, and an ultrasonic sensor setup that keeps the robot from hitting obstacles. All of this was managed using an Arduino Nano.

The core functionality worked well in the final prototype. The Voice Recognition Module V3 responded accurately once it was trained properly, and it continued to perform reliably even in moderately noisy surroundings. Bluetooth support through the HC-05 module helped during testing and also acted as a backup control option. The ultrasonic sensor, mounted on a servo motor, scanned the area in front of the car and allowed the system to stop immediately if something came too close. The Arduino coordinated the voice commands and sensor data, making sure that the robot always gave priority to safety over movement.

There were, however, some parts of the plan that could not be finished. The continuous left-to-right servo scanning sometimes introduced small delays, especially when the car was moving at higher speeds. This limited the maximum speed the robot could safely achieve. Training the voice module for multiple users also reduced its accuracy, so the final design works best when trained for a single speaker. The idea of adding an autonomous navigation mode was explored but had to be dropped due to time limitations.

One of the unique aspects of this project is the way it combines voice control with an automatic safety override. Unlike ordinary Bluetooth-controlled cars, it will stop itself if it detects danger—even if the user tells it to move forward. This makes the system particularly suitable for assistive technologies, where safety is crucial. The design is also modular and low-cost, making it easy to upgrade later with more advanced features such as AI-based speech recognition or camera-based navigation.

Overall, this project has potential applications in voice-controlled wheelchairs, educational robotics, home service robots, and interactive learning kits. The project highlights how simple electronic components and sensors can be brought together to create practical, easy-to-use robotic systems.

## Circuit/ System Schematic

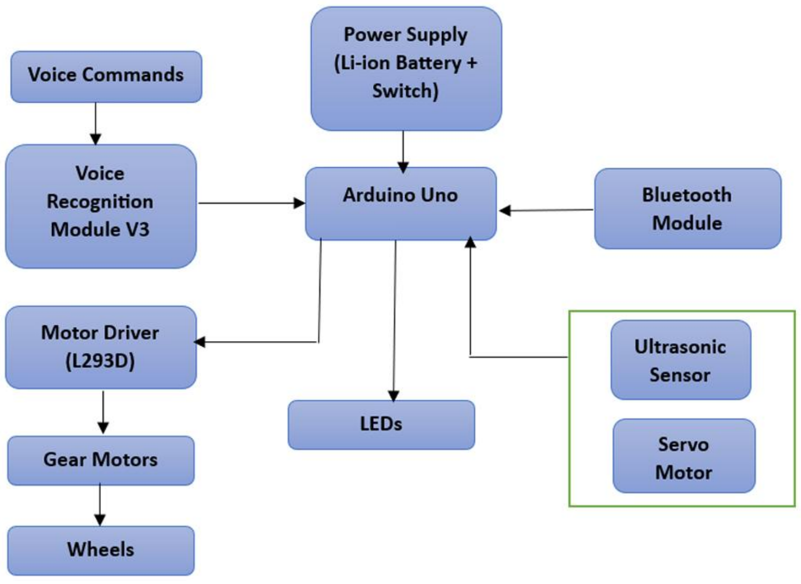

Figure 1: Block diagram for circuit

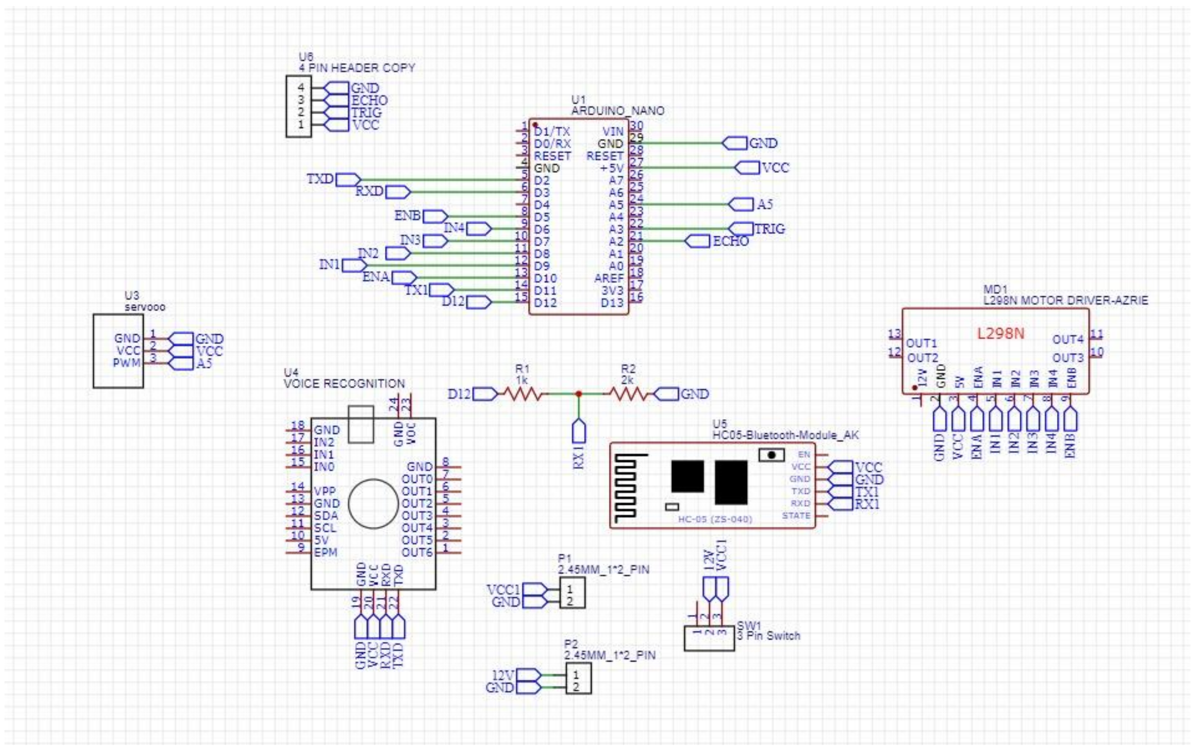

Figure 2: Circuit diagram

## PCB Layout

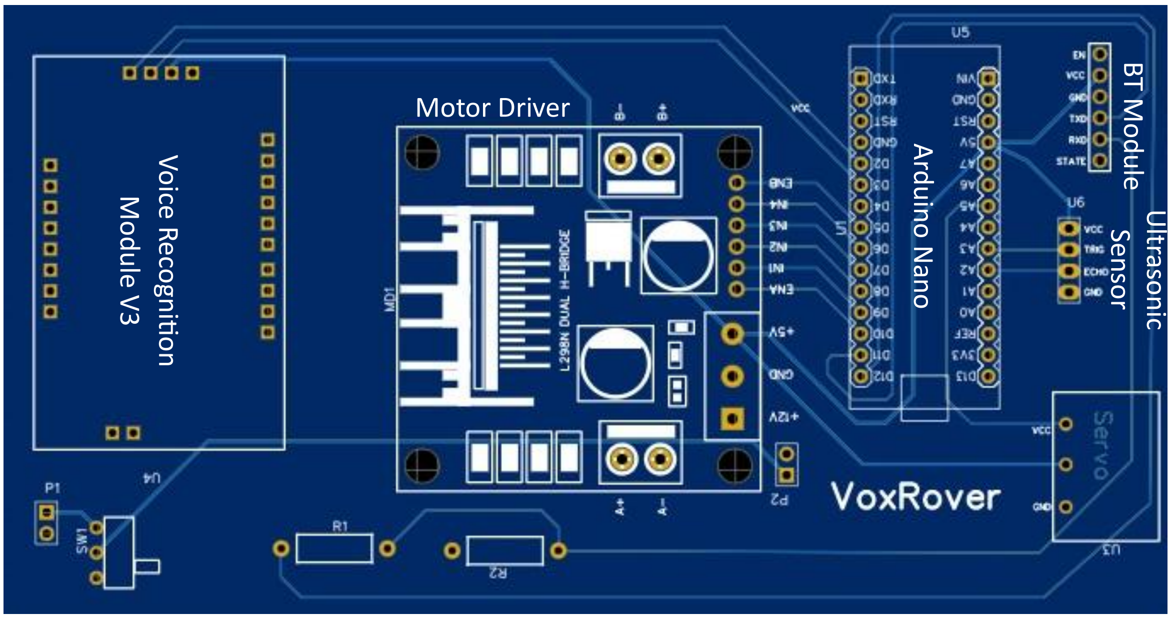

Figure 3: 2D view of PCB Layout

## Assembled System

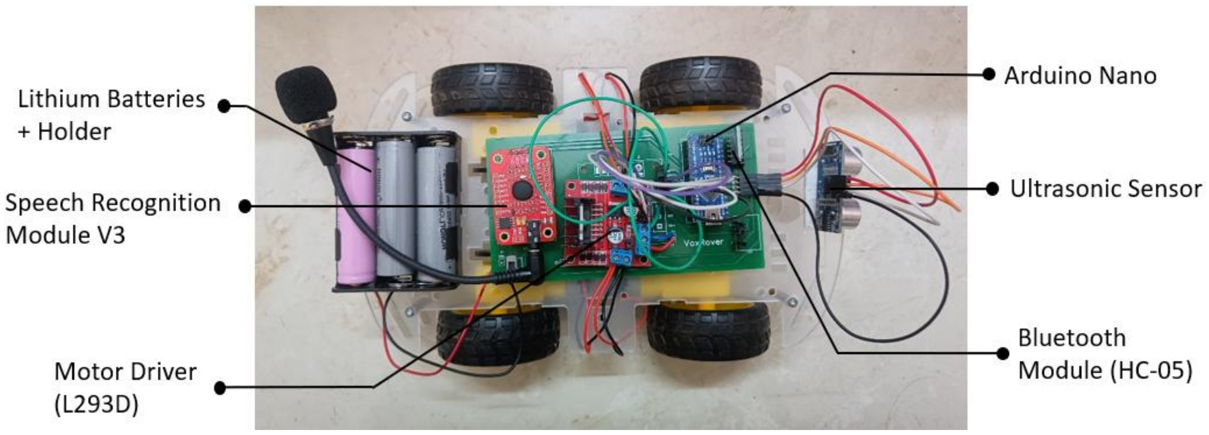

Figure 4: Assembled system on PCB

## Challenges and Troubleshooting

Based on the text provided from your report, here is the point-wise breakdown of the challenges faced and the troubleshooting steps taken:

### 1. Unequal Motor Speeds

**The Problem**:

We observed that the motors rotated at different speeds even when provided with the exact same PWM values. Initially, we suspected a mechanical mismatch in the motors, but individual testing revealed the root cause was defective or unstable Arduino PWM pins delivering inconsistent signals.

**Troubleshooting:**

i. We tested motors individually to rule out mechanical faults.
ii. We shifted the motor driver connections to more reliable pins on the Arduino.
iii. We modified the code to avoid Timer-conflicting PWM outputs, which stabilized the motion significantly.

### 2. Voice Recognition Accuracy in Noise

**The Problem:**

The Voice Recognition Module V3 struggled to detect commands in environments with slight background noise, such as classrooms, fans, or motor vibrations. The system often failed to register a command unless the user repeated it multiple times.

**Troubleshooting:**

i. We retrained the module using cleaner voice samples.
ii. We increased the time gap between training repetitions to ensure distinct command registration.
iii. We added small delays in the code to prevent signal interference, though clear speech from a fixed distance remains necessary.

### 3. Servo Jitter and False Detection

**The Problem**:

During continuous scanning, the servo motor tended to jitter, causing the ultrasonic sensor to generate inconsistent distance readings. This instability led to false obstacle detections, triggering the safety stop function unnecessarily.

**Troubleshooting:**

i. We slowed down the scanning range of the servo to reduce mechanical stress.
ii. We added short delays between sensor readings to allow the servo to settle before measuring distance, making detection more stable.

### 4. Hardware Instability (Ground Loops & Voltage)

**The Problem:**

The system faced typical hardware issues, including loose jumper wires on the breadboard, battery voltage drops that reduced motor torque, and ground loops that caused unstable sensor data.

**Troubleshooting:**

i. We implemented proper wiring techniques to secure loose connections.
ii. We isolated the motor power lines from the sensor logic to prevent voltage fluctuations.
iii. We migrated the circuit from a breadboard to a PCB, which significantly reduced electrical noise.

**5. Microcontroller Processing Overload**

**The Problem:**

A planned feature to combine voice control with partial autonomous navigation could not be implemented because the real-time processing load was too high for the Arduino Nano. Processing voice commands, reading sensors, and controlling motors simultaneously left insufficient processing headroom.

**Troubleshooting:**

Due to hardware limitations, the autonomous navigation feature was removed from the final prototype to ensure the reliability of the core voice control and safety stop features.

## Conclusions and Future Scope

This project provided significant technical learning in embedded systems, motor control, sensor calibration, servo dynamics, PCB design, and real-time programming. Key takeaways include the importance of noise isolation in mixed-signal circuits, the challenges of real-time task synchronization, and optimizing embedded code for low-power microcontrollers.

Future improvements include migrating to a more powerful microcontroller (e.g., ESP32), adding AI-based speech models, integrating camera-based navigation, improving PCB power routing, and enabling multi-speaker or multilingual voice support. The system can evolve into a robust platform for assistive robotics or smart autonomous devices.

## References

[1] Instructables.com, "Introduction to Voice Recognition With Elechouse V3," *Instructables*, n.d. https://www.instructables.com/Introduction-to-Voice-Recognition-With-Elechouse-V/.

[2] Texas Instruments, "L293x Quadruple Half-H Drivers," Datasheet SLRS008D, Jan. 2016. https://cdn-shop.adafruit.com/datasheets/l293d.pdf.

[3] ITead Studio, "HC-05 Bluetooth to Serial Port Module Datasheet," *Faranux*, Dec. 2016. https://www.faranux.com/wp-content/uploads/2016/12/Datasheet.pdf.

[4] Arduino, "Arduino Nano (A000005) Datasheet," *Arduino Documentation*, Mar. 2025. https://docs.arduino.cc/resources/datasheets/A000005-datasheet.pdf.