Electronics Workshop (EC-106)- Project report

Dual-Control Smart Car

Made by: Sarthak Sharma (24/A14/003)

Date of Submission - 24th April, 2025

Submitted to - Mr. Pankaj Dahiya

ACKNOWLEDGEMENT

I would like to express my special thanks to my teacher Mr. Pankaj Dahiya for providing us with this golden opportunity to work on the project whose topic is “Dual-Control Smart Car”, which led us to do a lot of research and learn about the model as well as presentation of the project report.

Further, I would like to thank my parents and friends for their invaluable support. I would like to extend my gratitude towards the teachers of my department who helped me with this project and provided me with unending assistance and guidance .

Thank You

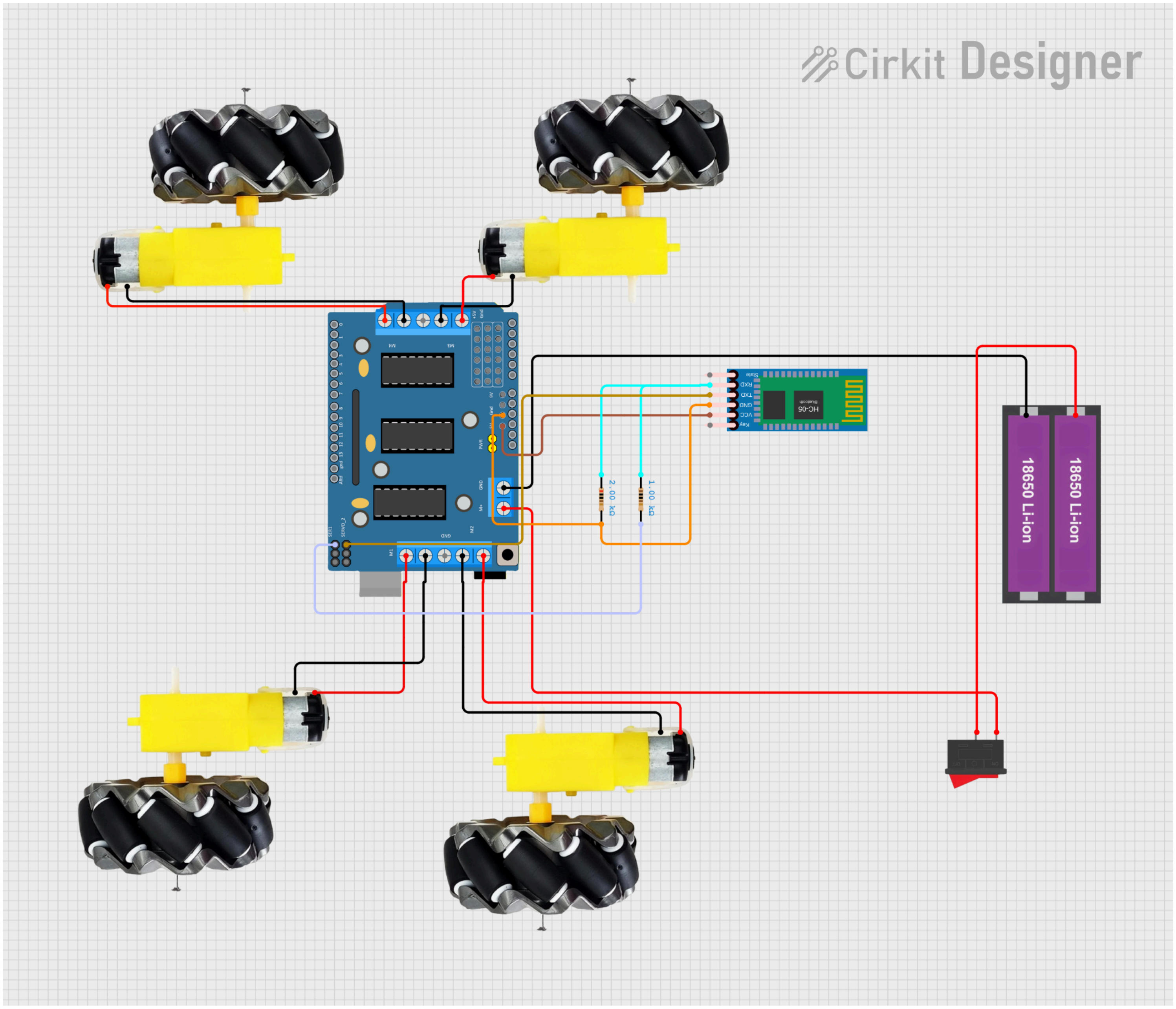
Table of Contents

TITLE	Page No.
Introduction	1
Circuit Diagram	2
Procedure	3
Arduino Code	4-6
Python Code	7-8
Working - Dual Control System - Movement control - Gesture Recognition Logic - Power Management - Operating Logic Summary	9-10
Applications	11-12
Relevance in Today's World	12-14
References	14

INTRODUCTION

The Dual Control Smart Car project presents a versatile and interactive robotic system designed for both manual and gesture-based control, utilizing Arduino as the core control unit. With increasing demand for intuitive and accessible robotic platforms, this project aims to integrate traditional RC-based Bluetooth control with modern computer vision techniques using OpenCV for hand gesture recognition. The smart car can be maneuvered using a smartphone app or through simple hand gestures detected via a camera, offering a dual-mode operation for enhanced user experience. This system demonstrates the potential for hybrid control mechanisms in robotics, paving the way for applications in smart mobility, assistive technologies, and interactive learning environments.

Circuit Diagram



PROCEDURE

1. **Chassis & Assembly**
   Used a **ready-made 4-wheel robot chassis** with pre-mounted BO motors. Mounted the **Arduino UNO** and **L293D Motor Shield** on top using screws or double-sided tape.

2. **Motor & Shield Connections**
   Connected the motors to M1–M4 terminals on the motor shield. The shield handles both direction and speed control for all motors.

3. **Bluetooth Module Setup**
   Connected **HC-05 Bluetooth module** to the Arduino (via the motor shield) for wireless communication with a mobile app.

4. **Gesture Control (OpenCV)**
   Set up a webcam connected to a laptop or Raspberry Pi. Used a **Python + OpenCV script** to detect hand gestures and send commands to Arduino via **serial communication** using PySerial.

5. **Power Supply**
   Powered the car using **two 3.7V Li-ion batteries** in series (7.4V total). Connected to the motor shield's external power terminal with an ON/OFF switch for control.

6. **Arduino Programming**
   Uploaded code to interpret both Bluetooth and gesture inputs and convert them into directional movement commands for the motors.

7. **Testing & Calibration**
   Verified movement via mobile app and hand gestures. Tuned gesture sensitivity and motor response for smooth operation.

ARDUINO CODE

```
#include <AFMotor.h>
#include <SoftwareSerial.h>

SoftwareSerial HC05(9, 10); // RX, TX

//initial motors pin
AF_DCMotor motor1(1);
AF_DCMotor motor2(2);
AF_DCMotor motor3(3);
AF_DCMotor motor4(4);

char command;

void setup()
{
  HC05.begin(9600);  //Set the baud rate to your Bluetooth module.
}

void loop() {
  if (HC05.available() > 0) {
    command = HC05.read();

    Stop(); //initialize with motors stoped

    switch (command) {
      case 'F':
        forward();
        break;
      case 'B':
        back();
        break;
    case 'L':
        left();
        break;
      case 'R':
        right();
        break;
    }
  }
}

void forward()
{
  motor1.setSpeed(255); //Define maximum velocity
  motor1.run(FORWARD);  //rotate the motor clockwise
  motor2.setSpeed(255); //Define maximum velocity
  motor2.run(FORWARD);  //rotate the motor clockwise
  motor3.setSpeed(255); //Define maximum velocity
  motor3.run(FORWARD);  //rotate the motor clockwise
  motor4.setSpeed(255); //Define maximum velocity
  motor4.run(FORWARD);  //rotate the motor clockwise
}

void back()
{
  motor1.setSpeed(255); //Define maximum velocity
  motor1.run(BACKWARD); //rotate the motor anti-clockwise
  motor2.setSpeed(255); //Define maximum velocity
  motor2.run(BACKWARD); //rotate the motor anti-clockwise
  motor3.setSpeed(255); //Define maximum velocity
  motor3.run(BACKWARD); //rotate the motor anti-clockwise
  motor4.setSpeed(255); //Define maximum velocity
  motor4.run(BACKWARD); //rotate the motor anti-clockwise
}

void left()
```

```
{
  motor1.setSpeed(255); //Define maximum velocity
  motor1.run(BACKWARD); //rotate the motor clockwise
  motor2.setSpeed(255); //Define lower velocity
  motor2.run(BACKWARD); //rotate the motor clockwise
  motor3.setSpeed(255); //Define maximum velocity
  motor3.run(FORWARD);  //rotate the motor clockwise
  motor4.setSpeed(255); // Define lower velocity
  motor4.run(FORWARD);  //rotate the motor clockwise
}

void right()
{
  motor1.setSpeed(255); //Define lower velocity
  motor1.run(FORWARD);  //rotate the motor clockwise
  motor2.setSpeed(255); //Define maximum velocity
  motor2.run(FORWARD);  //rotate the motor clockwise
  motor3.setSpeed(255); //Define lower velocity
  motor3.run(BACKWARD); //rotate the motor clockwise
  motor4.setSpeed(255); //Define maximum velocity
  motor4.run(BACKWARD); //rotate the motor clockwise
}

void Stop()
{
  motor1.setSpeed(0);  //Define minimum velocity
  motor1.run(RELEASE); //stop the motor when release the button
  motor2.setSpeed(0);  //Define minimum velocity
  motor2.run(RELEASE); //rotate the motor clockwise
  motor3.setSpeed(0);  //Define minimum velocity
  motor3.run(RELEASE); //stop the motor when release the button
  motor4.setSpeed(0);  //Define minimum velocity
  motor4.run(RELEASE); //stop the motor when release the button
}
```

Python Code

```python
import cv2
from cvzone.HandTrackingModule import HandDetector
import serial
import time

bluetooth_port = 'COM7'
baud_rate = 9600

# Try connecting to the Bluetooth serial port
try:
    bt = serial.Serial(bluetooth_port, baud_rate, timeout=1)
    time.sleep(2)  # Let it connect
    print(f"✅ Connected to Bluetooth on {bluetooth_port}")
except Exception as e:
    print(f"❌ Failed to connect to Bluetooth: {e}")
    bt = None

# Initialize camera and hand detector
cap = cv2.VideoCapture(0)
detector = HandDetector(detectionCon=0.5, maxHands=1)

while True:
    ret, frame = cap.read()
    if not ret or frame is None:
        print("⚠️ Frame capture failed. Skipping...")
        continue

    # Flip/rotate as needed
    frame = cv2.rotate(frame, cv2.ROTATE_180)
    frame = cv2.flip(frame, -1)

    hands, frame = detector.findHands(frame)
    output = "S"  # Default gesture (Stop)
    if hands:
        hand = hands[0]
        fingers = detector.fingersUp(hand)
        count = fingers.count(1)

        if count == 1:
            output = "F"
        elif count == 2:
            output = "B"
        elif count == 3:
            output = "L"
        elif count == 4:
            output = "R"
        elif count == 0 or count == 5:
            output = "S"
    else:
        output = "S"

    # Send output every frame
    print(f"➡️ Sending: {output}")
    if bt:
        bt.write(output.encode())

    cv2.imshow("Hand Tracking", frame)
    if cv2.waitKey(1) & 0xFF == 27:  # ESC to quit
        break

# Cleanup
cap.release()
cv2.destroyAllWindows()
if bt:
    bt.close()
    print("🔌 Bluetooth connection closed.")
```

Working of Dual Control Smart Car

1. Dual Control System

a. Bluetooth-Based RC Control (Manual Mode):

- Controlled via an Android smartphone using a Bluetooth RC car app.
- The app sends directional commands (Forward, Backward, Left, Right, Stop) via Bluetooth (HC-05 module).
- The Arduino receives these commands through the serial interface and maps them to motor actions.
- This allows real-time, user-directed operation of the car.

b. Gesture-Based Control (Autonomous Mode):

- A webcam captures live video input, processed using a Python script with OpenCV.
- Hand gestures are recognized using contour detection and finger counting.
- Each gesture corresponds to a directional command (e.g., one finger = forward, two fingers = backward, etc.).
- The gesture data is sent over serial (via USB or Bluetooth) to the Arduino using **PySerial**.
- Arduino interprets the commands and controls the motors accordingly.

The car can switch seamlessly between both modes, depending on user preference or context.

---

2. Movement Control

- **Four BO motors** (two on each side) are connected to the **L293D Motor Shield** mounted on top of the Arduino UNO.
- Motor channels M1 to M4 control the left and right wheel pairs individually.
- Direction and speed are controlled using logic signals and PWM sent from the Arduino.
- Basic motion logic includes:
  - **Forward:** All motors rotate in the same direction
  - **Backward:** All motors rotate in the reverse direction
  - **Left Turn:** Right-side motors move forward, left-side motors stop or reverse
  - **Right Turn:** Left-side motors move forward, right-side motors stop or reverse
  - **Stop:** All motors stopped by setting LOW or zero PWM

---

3. Gesture Recognition Logic (Python + OpenCV)

- The Python script continuously captures frames and applies:

    - Background subtraction
    - Contour detection
    - Convex hull + finger count analysis
- Based on the number of extended fingers:
    - **1 Finger:** Move Forward
    - **2 Fingers:** Move Backward
    - **3 Fingers:** Turn Left
    - **4 Fingers:** Turn Right
    - **Fist/0 Fingers:** Stop
- Commands are sent via serial to the Arduino to trigger corresponding movements in real-time.

---

4. Power Management

- The smart car is powered by **two 3.7V Li-ion 18650 batteries** connected in series (7.4V total).
- Power is supplied through the **L293D motor shield's external terminal**.
- A toggle switch allows the user to turn the system on/off easily.
- The shield's onboard regulator ensures voltage is safely delivered to both Arduino and motors.
- All components share a common GND for stability.

---

5. Operating Logic Summary

- The system is designed to:
    - Accept commands from either Bluetooth or gesture recognition input
    - Translate these into motion commands through the Arduino
    - Drive the car in the specified direction using the L293D shield
- The hybrid control system showcases how real-time interaction between physical devices and computer vision can result in smooth, dual-mode robotic control.

APPLICATIONS

Industrial Automation:

- **Warehouses:** Remote inspection of inventory aisles, reducing need for human navigation in tight or hazardous areas.
- **Manufacturing Plants:** Equipment monitoring and material transport in confined or dynamic factory environments.

Education & Learning:

- **STEM Education:** Ideal for teaching robotics, Arduino programming, and computer vision concepts through hands-on learning.
- **Workshops & Competitions:** Used in robotics contests and training programs to demonstrate hybrid control systems.

Home Automation & Assistance:

- **Smart Homes:** Gesture-based control for mobility applications within homes, useful for people with limited mobility.
- **Elderly Assistance:** Can be adapted for carrying small items or providing interaction via simple gesture controls.

Surveillance & Inspection:

- **Restricted Access Areas:** Can be deployed in tight or hazardous locations, remotely controlled via app or gestures.
- **Campus or Office Patrol:** Lightweight security patrols in closed or semi-open spaces during off-hours.

Entertainment & Hobbyist Projects:

- **DIY Robotics Projects:** Popular among hobbyists looking to explore RC and gesture control integration.
- **Interactive Demonstrations:** Used in exhibitions, tech fairs, and science shows to engage audiences with real-time control.

Prototyping & Research:

- **Human-Machine Interaction Studies:** Useful for studying intuitive control methods, especially in gesture recognition research.
- **Smart Mobility Testing:** Base platform for testing autonomous navigation, obstacle avoidance, and control switching systems.

Accessibility Solutions:

- **Customizable Interfaces:** Can be tailored to offer gesture-based control for users with physical impairments, improving device accessibility.
- **Rehabilitation Tools:** Useful in therapy environments to encourage movement and interaction through gamified gesture-based tasks.

RELEVANCE IN TODAY'S WORLD

1. Bridging the Gap Between Manual and Autonomous Control

In a rapidly advancing technological world, the ability to control machines both remotely and through intuitive human interaction is increasingly important. With smart devices becoming integral to daily life, systems that can seamlessly switch between manual and gesture-based operation are in high demand. This dual control smart car project showcases how hybrid control methods can:

- Enhance user interaction and adaptability
- Cater to different user needs and environmental scenarios
- Offer an engaging way to explore human-machine interfaces

2. Smart Mobility for the Future

With urbanization and automation on the rise, small-scale robotic mobility platforms are becoming essential for various applications such as surveillance, remote inspection, and delivery. The dual control model offers:

- Remote accessibility through Bluetooth for real-time control
- Hands-free operation via gesture control for scenarios requiring minimal physical contact
- Compatibility with AI-driven systems for future automation

3. Inclusive and Accessible Technology

This project supports the growing need for technologies that are accessible and customizable. Gesture control is particularly useful in:

- Assisting people with physical impairments

- Reducing the dependency on touch-based systems in sensitive environments (labs, hospitals)
- Creating educational tools that promote inclusivity and interaction

4. 24/7 Operation with Flexible Control

Unlike traditional RC vehicles, the dual control model provides flexibility in how it is used—either manually or through intelligent gesture detection, allowing for:

- Operation in restricted or hard-to-reach spaces
- Control during low-visibility conditions when phones may not be ideal
- Continuous monitoring with reduced physical engagement

5. Scalable and Adaptable Framework

The design is open-source and modular, allowing easy customization based on user requirements:

- Can be scaled up or down for different mobility tasks
- Sensor suite and control methods can be upgraded
- Camera-based gesture control can be trained for custom gestures
- Compatible with additional modules like obstacle avoidance or line following

6. Cost-Effective and Educational

Built using readily available components such as:

- Arduino microcontroller
- Bluetooth module
- Standard motor driver and chassis
- Webcam and basic Python/OpenCV setup

Despite its simplicity, it offers high utility and learning potential:

- Ideal for student projects, prototypes, and hobbyists
- Easy to repair, modify, and improve
- Low-cost platform to explore advanced robotics concepts

7. Future Development Potential

This project lays a strong foundation for:

- Autonomous navigation with obstacle avoidance
- Integration with IoT and cloud control systems
- Enhanced gesture recognition using AI/ML models
- Voice command integration for multi-modal control

- Use in swarm robotics or coordinated fleet systems

This smart car project is a prime example of how accessible hardware and software can be combined to develop powerful, adaptable, and interactive systems. Its relevance spans education, automation, and even assistive technology, making it a timely and impactful innovation in today's evolving tech landscape.

References

►Saad innovative ideas – youtube channel

►Wikipedia (articles related to bluetooth controlled cars)

►Researchgate website – research papers on:

1. Motor Drivers
2. Uses of BO Motors
3. Bluetooth Signal Transmission