23CSE201-Procedural Programming using **C**

Semester-**3**(2024-2028)

Section-**H**

Project review **1**

Title: Autonomous Robot Navigation System

Team Members: Team8

Devarasetty Kuldeep	-	CB.SC.U4CSE24712
Pasumarthy Abhinav	-	CB.SC.U4CSE24738
Rudraraju Ethish	-	CB.SC.U4CSE24761
Namburi Vivek	-	CB.SC.U4CSE24762

Problem Statement:

In contemporary robotics, autonomous driving and remote control are two fundamental features that allow robots to perform well in different environments. Most applications, including warehouse management, delivery vehicles, and inspection units, demand a robot drive through pre-designed courses, evade obstacles in real time, and also permit manual control when desired. Implementing all three features in one low-cost system is both an application challenge and an educational issue.

This project involves designing and developing a small mobile robot that can move in three modes:

1. Obstacle Avoidance Mode – The robot employs an servo-mounted ultrasonic distance sensor (scanning left and right) to sense obstacles on its path and automatically steer around them to avoid collisions.

2. Path Following Mode – The robot employs infrared (IR) line sensors to sense and follow a path marked on the ground with little deviation.

3. Manual Control Mode – The robot can be manually controlled from a smartphone by means of Bluetooth Low Energy (BLE) communication using ArduinoBLE library, enabling the user to drive it directly.

The primary goal of this project is to put these functions into practice utilizing C programming on an Arduino UNO R4 microcontroller, incorporating basic programming principles like variables, conditionals, loops, arrays, functions, and interrupts. The system will be constructed entirely out of low-cost, easily accessible components, with effective code structure permitting smooth mode shifting and stable operation.

This combined methodology not only illustrates practical embedded system uses in the real world but also offers a solid learning experience on how to integrate sensor data processing, motor control code, and wireless communication into a one-working robot.

Hardware Requirements

Core Components:

- Arduino UNO R4 WiFi – Main microcontroller board with built-in Wi-Fi & Bluetooth.
- L298N Motor Driver Module – Controls the direction and speed of DC motors.
- TT N20 6V Gear Motors with Encoders (2 pcs) – For driving and tracking movement.
- Servo Motor – To move the Ultrasonic Sensor equipped on it.
- 65mm Robot Wheels – Compatible with the motors for movement.

Sensors:

- HC-SR04 Ultrasonic Sensor – For obstacle detection and avoidance.
- TCRT5000 IR Line Tracking Sensors (3 pcs) – For detecting and following a path/line.

Power System:

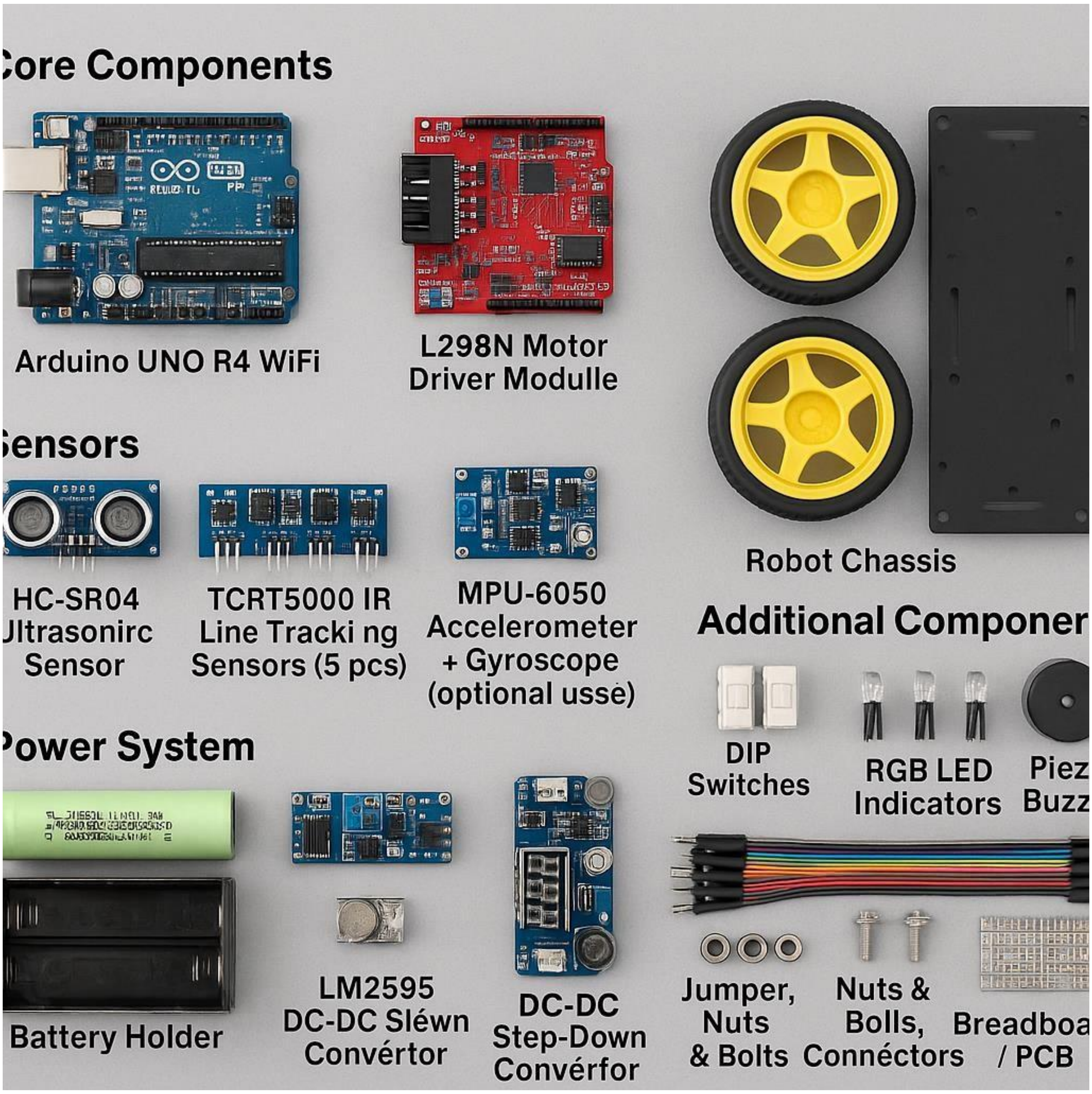
- 6 × 18650 Li-ion Batteries (Panasonic NCR18650GA 3300mAh) – Main power source.
- 2S BMS (Battery Management System) Module – Protects the batteries from overcharge/discharge.
- Battery Holder (6-Cell) – To mount and connect the Li-ion batteries.

Additional Components:

- Jumper Wires, Nuts & Bolts, Connectors – For assembly and wiring.
- Breadboard / PCB - For neat wiring.
- Robot Chassis – Holds all components securely.
- Servo Motor Bracket Set.

Software Requirements:

- Arduino IDE (Free) – For writing and uploading C code to Arduino UNO R4.
- User Interface(HTML Page) - A HTML page that sends commands through BLE to the Arduino UNO R4 WiFi to process


Core Components
Arduino UNO R4 WiFi
L298N Motor Driver Modulle
Robot Chassis
Sensors
HC-SR04 Ultrasonirc Sensor
TCRT5000 IR Line Tracki ng Sensors (5 pcs)
MPU-6050 Accelerometer + Gyroscope (optional ussė)
Additional Componer
DIP Switches
RGB LED Indicators
Piez Buzz
Power System
Battery Holder
LM2595 DC-DC Sléwn Convértor
DC-DC Step-Down Convérfor
Jumper, Nuts & Bolts
Nuts & Bolls, Connéctors
Breadboa / PCB

Architecture:

1. Input Layer (Sensors & User Control)

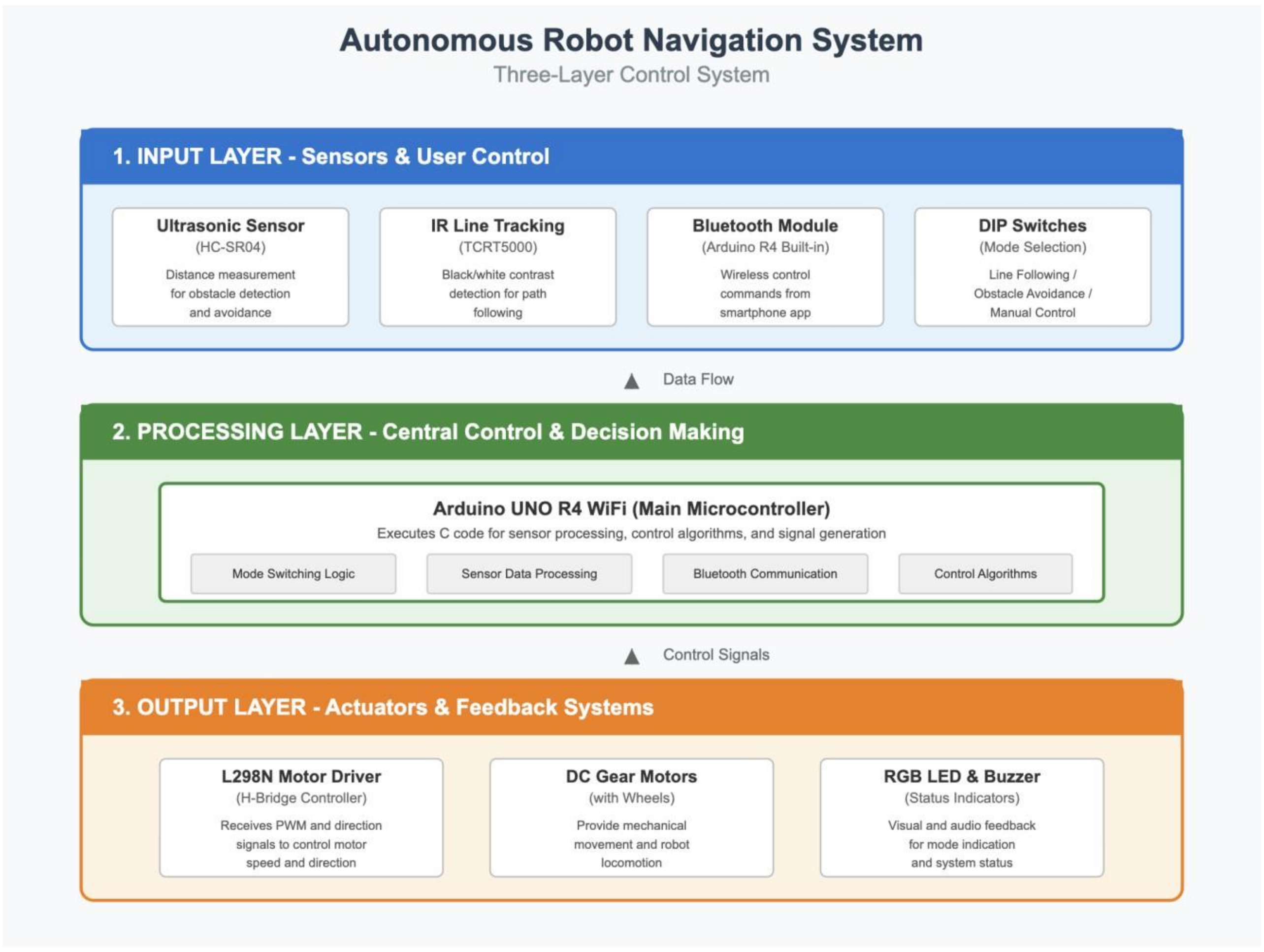
- Ultrasonic Sensor (HC-SR04) - Measures distance to detect obstacles.
- IR Line Tracking Sensors (TCRT5000) - Detect black/white contrast to follow a path.
- Bluetooth Module (built into Arduino UNO R4) – Receives manual control commands from a smartphone app.
- Takes input commands from user through BLE from the HTML page.

2. Processing Layer (Arduino UNO R4 WiFi)

- Executes C code to process sensor data, apply logic, and send control signals.
- Switches between modes based the input command from the user.
- Processes Bluetooth input for manual mode.
- Runs obstacle detection and line-following algorithms in autonomous modes.

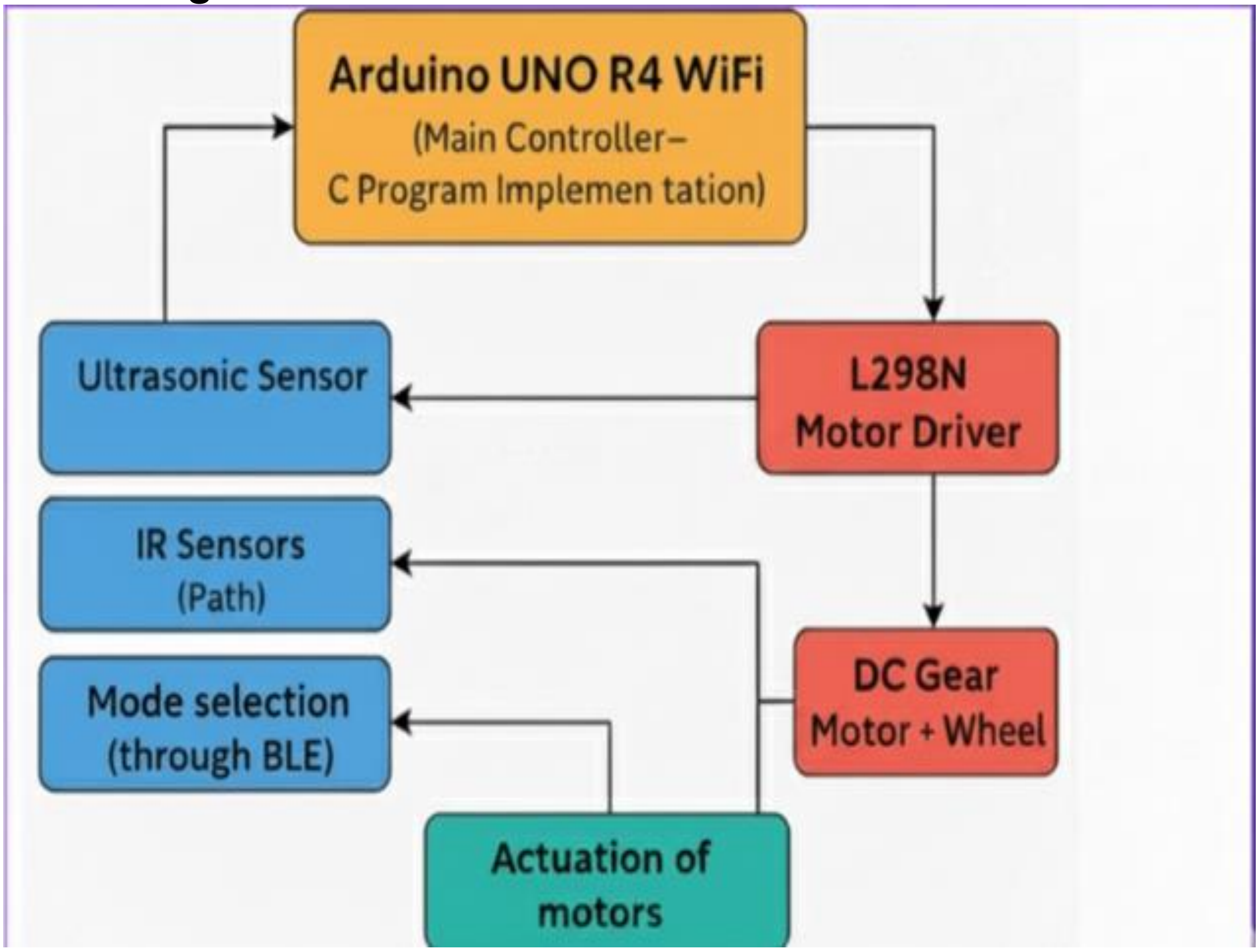
3. Output Layer (Actuators & Indicators)

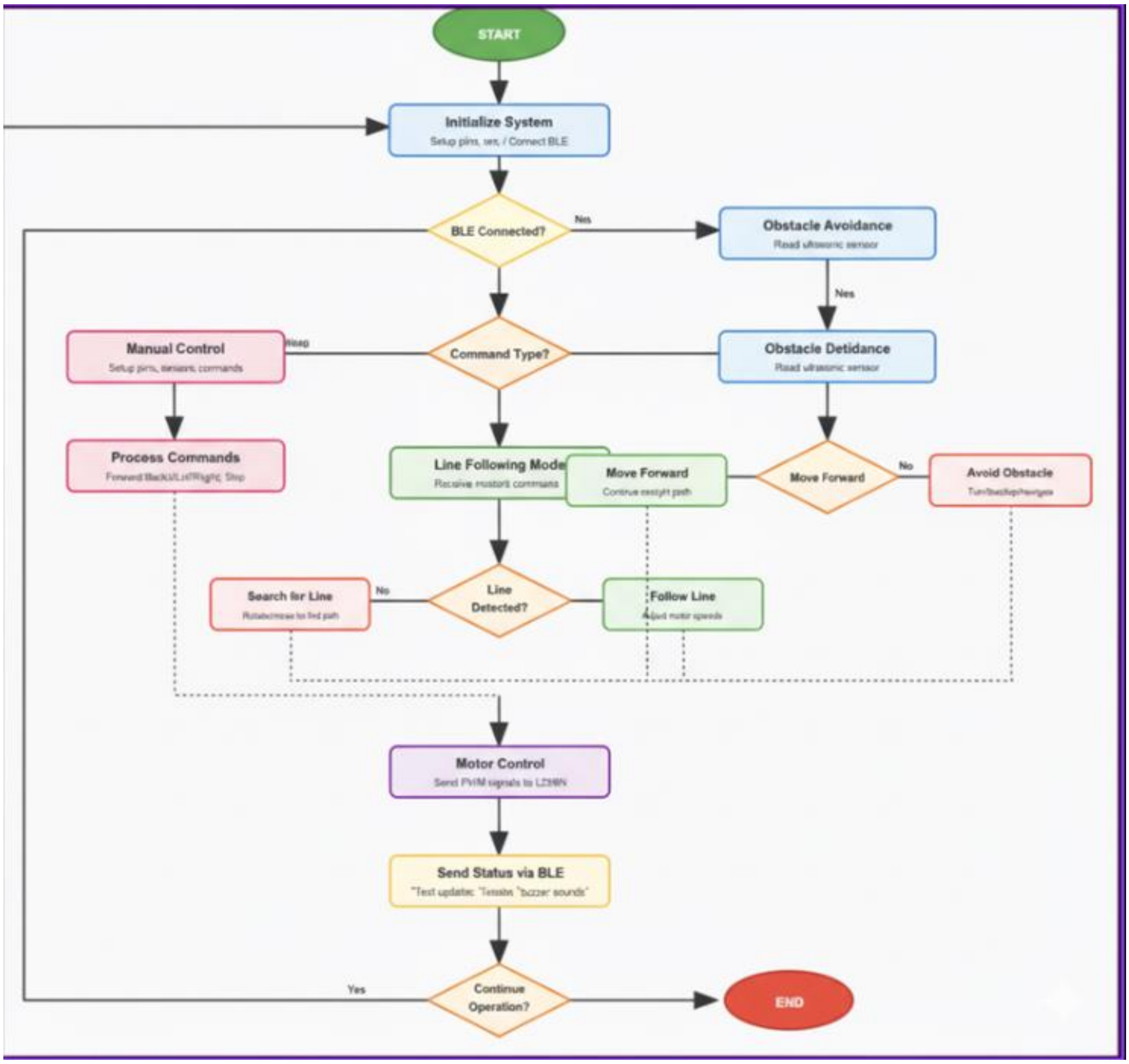
- L298N Motor Driver – Receives PWM and direction signals from Arduino to control motors.
- DC Gear Motors with Wheels - Provide movement.



Autonomous Robot Navigation System
Three-Layer Control System
1. INPUT LAYER - Sensors & User Control
Ultrasonic Sensor
(HC-SR04)
Distance measurement for obstacle detection and avoidance
IR Line Tracking
(TCRT5000)
Black/white contrast detection for path following
Bluetooth Module
(Arduino R4 Built-in)
Wireless control commands from smartphone app
DIP Switches
(Mode Selection)
Line Following / Obstacle Avoidance / Manual Control
Data Flow
2. PROCESSING LAYER - Central Control & Decision Making
Arduino UNO R4 WiFi (Main Microcontroller)
Executes C code for sensor processing, control algorithms, and signal generation
Mode Switching Logic
Sensor Data Processing
Bluetooth Communication
Control Algorithms
Control Signals
3. OUTPUT LAYER - Actuators & Feedback Systems
L298N Motor Driver
(H-Bridge Controller)
Receives PWM and direction signals to control motor speed and direction
DC Gear Motors
(with Wheels)
Provide mechanical movement and robot locomotion
RGB LED & Buzzer
(Status Indicators)
Visual and audio feedback for mode indication and system status

Flow Diagram:




START
Initialize System
Setup pins, sen, / Connect BLE
BLE Connected?
Nes
Obstacle Avoidance
Read ultrasonic sensor
Nes
Command Type?
Manual Control
Setup pins, sensors, commands
Obstacle Detidance
Read ultrasonic sensor
Process Commands
Forward/Back/Left/Right, Stop
Line Following Mode
Receive motors commands
Move Forward
Continue straight path
Move Forward
No
Avoid Obstacle
Turn/back/up/navigate
Search for Line
Rotate/move to find path
No
Line Detected?
Follow Line
Adjust motor speeds
Motor Control
Send PWM signals to L298N
Send Status via BLE
"Text updates: "Tensile "buzzer sounds"
Yes
Continue Operation?
END

C Programming Concepts + Design Principles:

1. Basic Structure & Flow

- **Program Entry** → main() function (Arduino: setup() + loop()).
- **Sequential, Conditional, Iterative flow** → if/else, loops, switch-case.

2. Data & Variables

- **Primitive Types** → int, float, char, unsigned, bool.
- **Constants & Macros** → const, #define for fixed values.
- **Enumerations (enum)** → State representation (e.g., FORWARD, STOP).

3. Functions & Modularity :

- **Function Decomposition → Break big tasks into smaller reusable functions.**
- **Parameters → Pass by value or by pointer (reference-like behavior).**
- **Return values → Use to get computed results from functions.**
- **Modular Programming** → Separate .c (implementation) and .h (declarations) files.

4. Encapsulation (C Style)

- No classes, but **achieved using struct + functions** in a .c file with internal variables as static so they're hidden from outside.

5. Structures & Data Grouping

- struct to bundle related variables (like object fields).
- Can combine with pointers for dynamic data handling.
- Additionally, the system uses dedicated structs (MotorPins, UltrasonicSensor, LineTrackingSensors, RobotConfig, RobotState) to organize hardware mapping, robot parameters, and state management in a modular way.

6. Abstraction

- Hide hardware details using functions (e.g., moveForward() instead of writing raw pin operations everywhere).
- Improves code readability & maintenance.

7. Pointers & Memory Management

- **Pointer basics** (int *p).
- **Pointer arithmetic** for arrays and buffers.
- **Dynamic allocation** with malloc()/free() if needed (less common in Arduino due to limited memory).

8. Arrays & Strings

- Store multiple readings or pin numbers.
- char[] for text messages (instead of Java's String).

9. Bitwise Operations

- Used for port manipulation, sensor data masking, and motor control.
- Operators: &, |, ^, ~, <<, >>.

10. Code Reusability

- **Libraries** for motors, sensors, and WiFi.
- Encourages DRY (Don't Repeat Yourself) principle.

11. Event Handling (Interrupts)

- attachInterrupt() to respond to hardware events instantly (e.g., stop when obstacle detected).

12. State Machines

- Maintain robot's mode: idle, moving forward, turning, avoiding obstacle.
- Implemented using enums + switch-case.

13. Defensive Programming

- Input validation (check sensor readings before use).
- Error handling (return error codes from functions).

14. Documentation & Readability

- Proper naming conventions.
- Comments (// and /* */).
- Indentation and consistent style.

Expected Output of the Robot Car Project:

1. Power-Up & Initialization

- On powering the robot, it initializes all sensors, motors, and serial communication.

2. Normal Operation

- **Movement**:
    - Moves forward until an obstacle is detected.
    - If obstacle detected within threshold distance (e.g., 10 cm):
    - Robot stops, then turns to avoid the obstacle, then continues forward.
- **IR Tracking Sensors**:
    - Follows a line if line-following mode is enabled.

3. Safety Features

- Fuse prevents damage from current overload.
- Battery protection board stops overcharging/discharging.

4. Final Demonstration

Behavior When fully coded

and assembled:

- Robot autonomously navigates avoiding obstacles.
- Can switch between **obstacle avoidance mode** and **line-following mode and manual mode.**
  (via user commands).
- Smooth motor control with adjustable speed.

- All sensors and actuators respond in real-time.

References:

1. C Programming Fundamentals

- Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language (2nd ed.)*. Prentice Hall.
- GeeksforGeeks – C Programming Language Tutorials:
  https://www.geeksforgeeks.org/c-programming-language

2. Embedded C & Arduino Programming

- Arduino Official Documentation:
  https://docs.arduino.cc
- Barrett, S. F., & Pack, D. J. (2020). *Arduino Microcontroller Processing for Everyone!*. Morgan & Claypool.
- Programming Embedded Systems in C and C++ – Michael Barr:
  https://barrgroup.com/embedded-systems/how-to/embedded-c-programming

3. Hardware Datasheets

- **Arduino UNO R4 WiFi** (Renesas RA4M1 / ESP32-S3):
  https://docs.arduino.cc/hardware/uno-r4-wifi
- **HC-SR04 Ultrasonic Sensor** Datasheet:
  https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
- **TCRT5000 IR Sensor** Datasheet:
  https://www.vishay.com/docs/83760/tcrt5000.pdf
- **L298N Motor Driver** Datasheet:
  https://www.st.com/resource/en/datasheet/l298.pdf
- **LM2596S Step-Down Converter** Datasheet:
  https://www.ti.com/lit/ds/symlink/lm2596.pdf