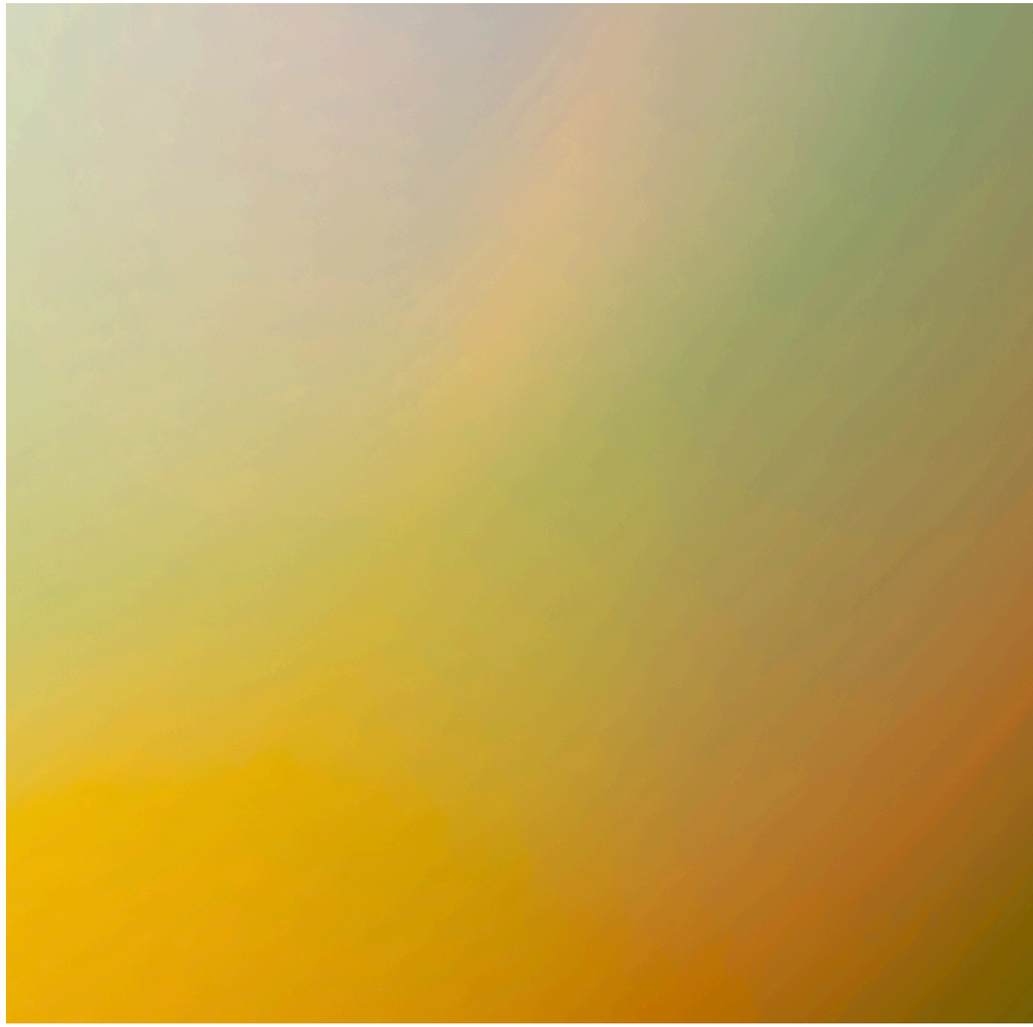
HSHL

PROTOTYPING

ELE-4



Muhammad Ali
Moeez Mufti

INTRODUCTION

Goal: Develop an autonomous robot car that can follow a line, detect obstacles and identify their color.

Key Features:

- Accurate line tracking using Infrared (IR) sensors
- Real-time obstacle detection with an ultrasonic sensor
- Color detection of obstacles using a color sensor
- Smart obstacle avoidance with a servo-mounted scanning sensor

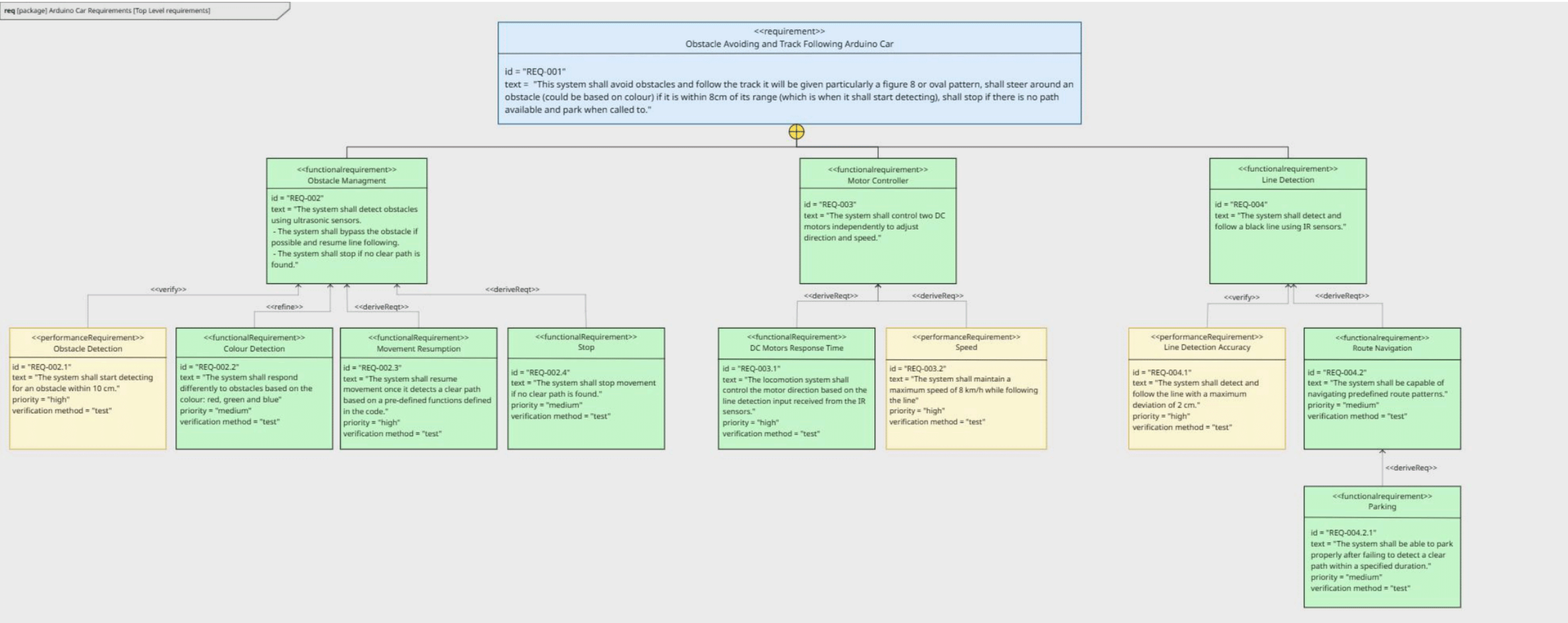
Technology:

- Controlled by an Arduino R4 Uno microcontroller
- Driven by DC motors & L298N motor driver module
- Integrated sensors: IR, Ultrasonic, Servo, and Color sensor

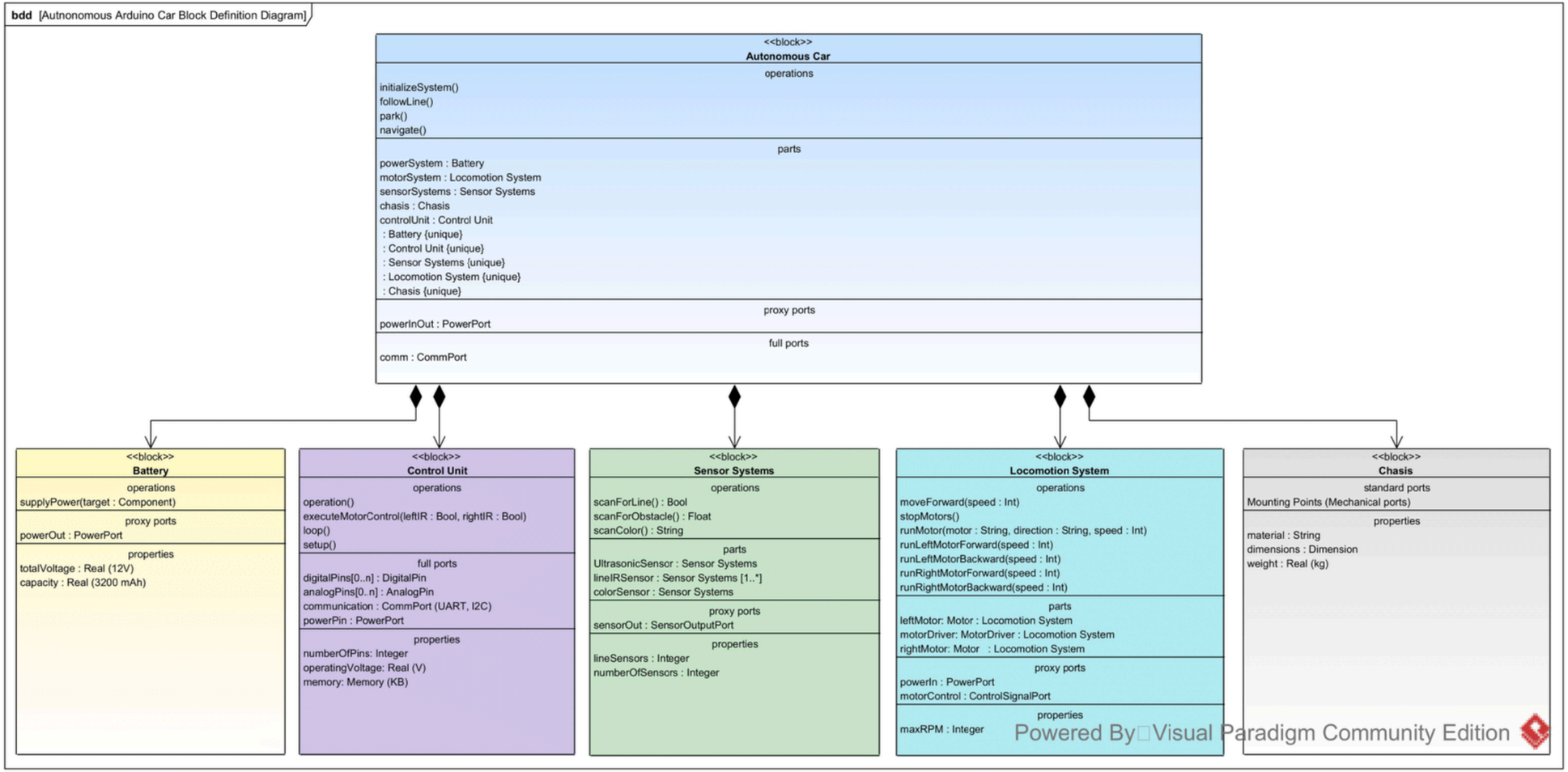
Purpose: Showcases principles of autonomous navigation, multi-sensor data fusion, and intelligent decision-making in embedded systems.

Applications: Robotics education, prototype for self-driving vehicle behavior, and groundwork for more complex autonomous robots.

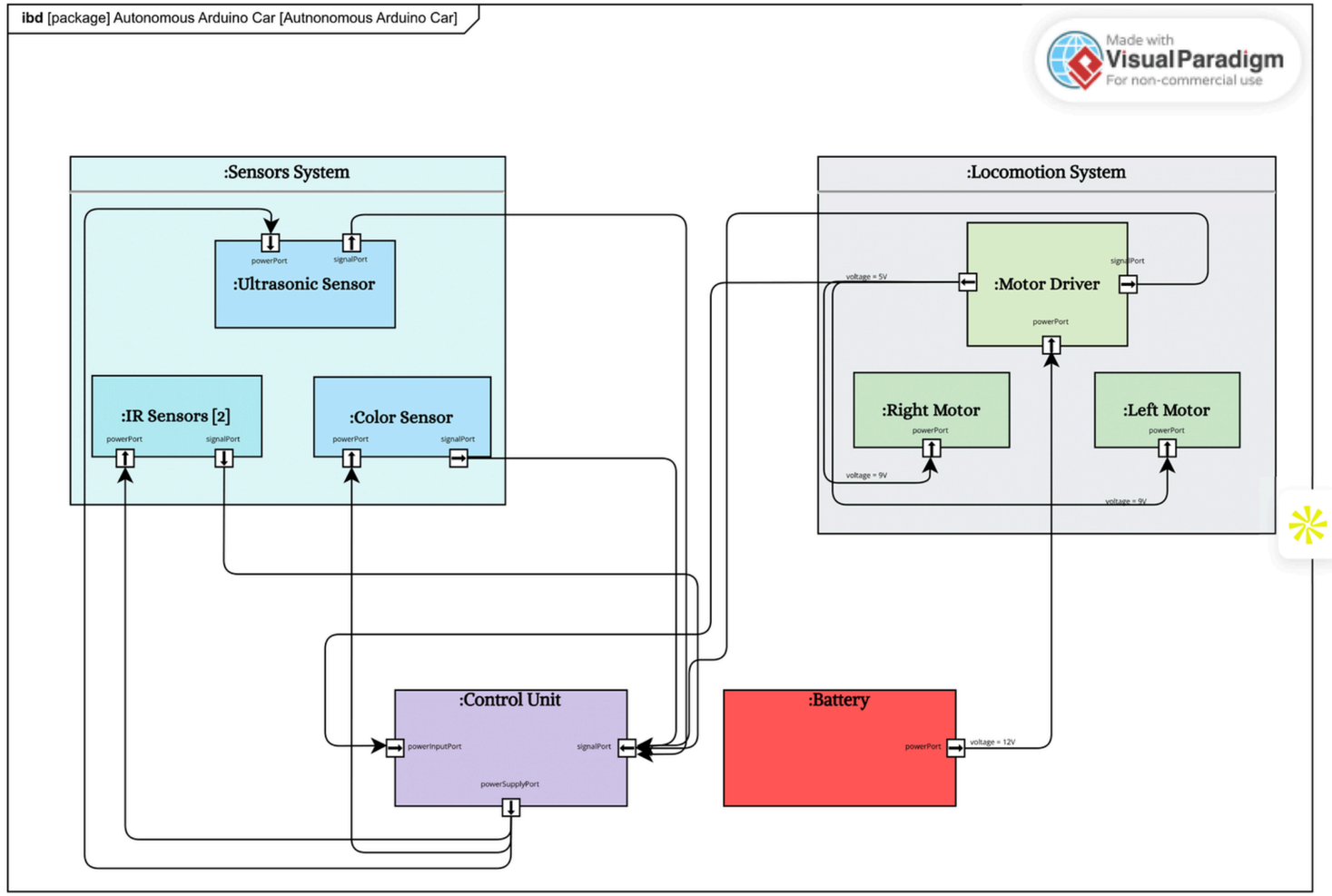
REQUIREMENTS DIAGRAM



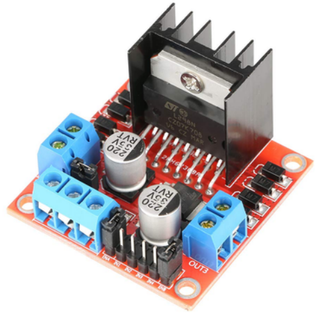
BLOCK DEFINITION DIAGRAM



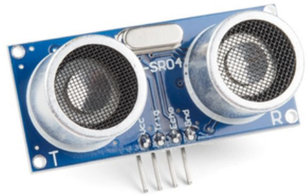
INTERNAL BLOCK DIAGRAM



COMPONENTS



L298N

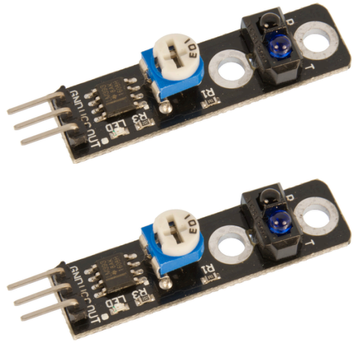


Ultrasonic Sensor

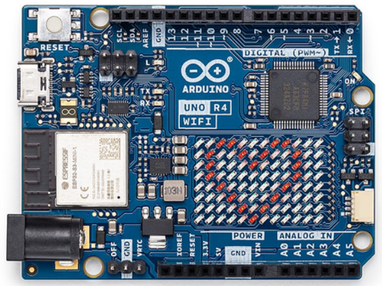
Color Sensor

Wheels & DC Motors

Servo Motor



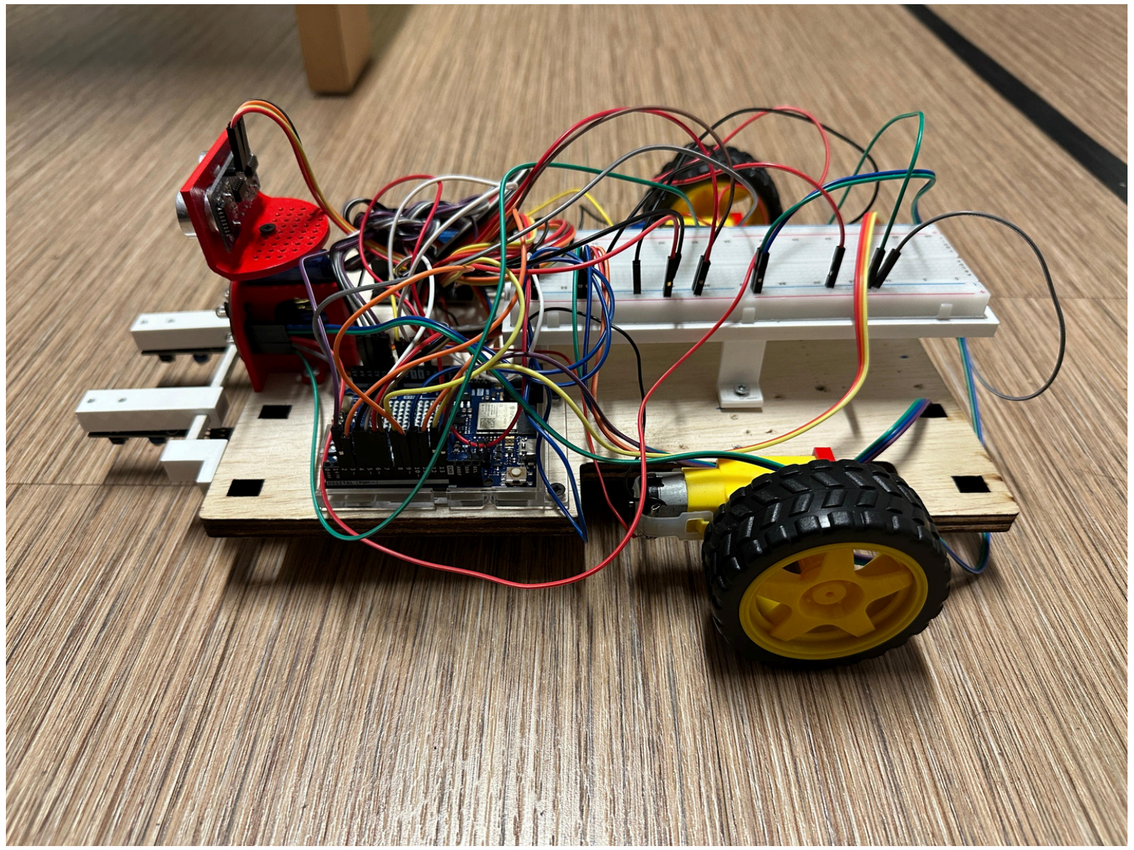
IR Sensors

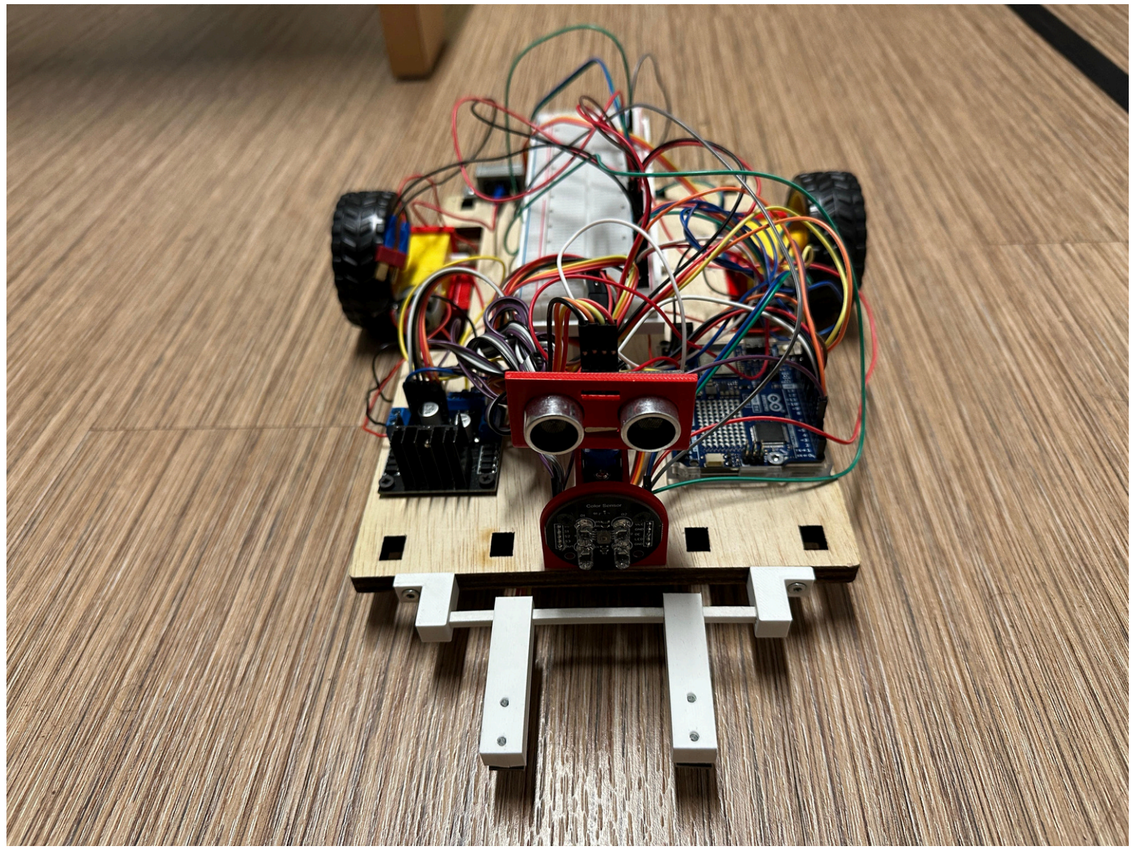


Arduino Uno R4

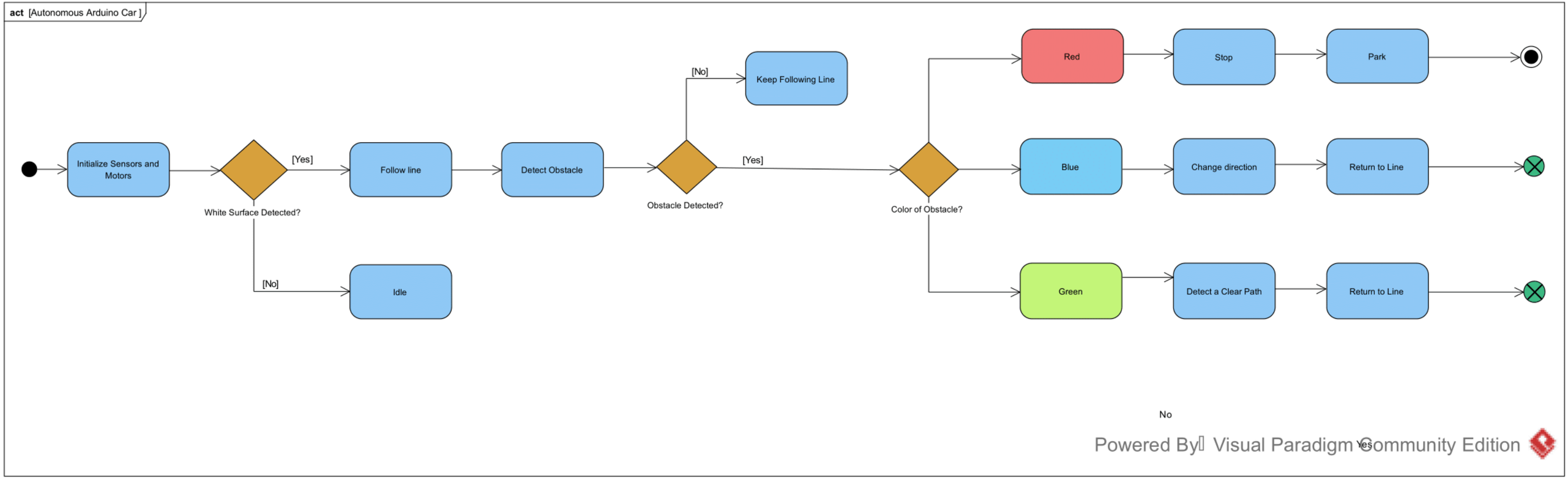
LiPo Battery

CAR ASSEMBLY

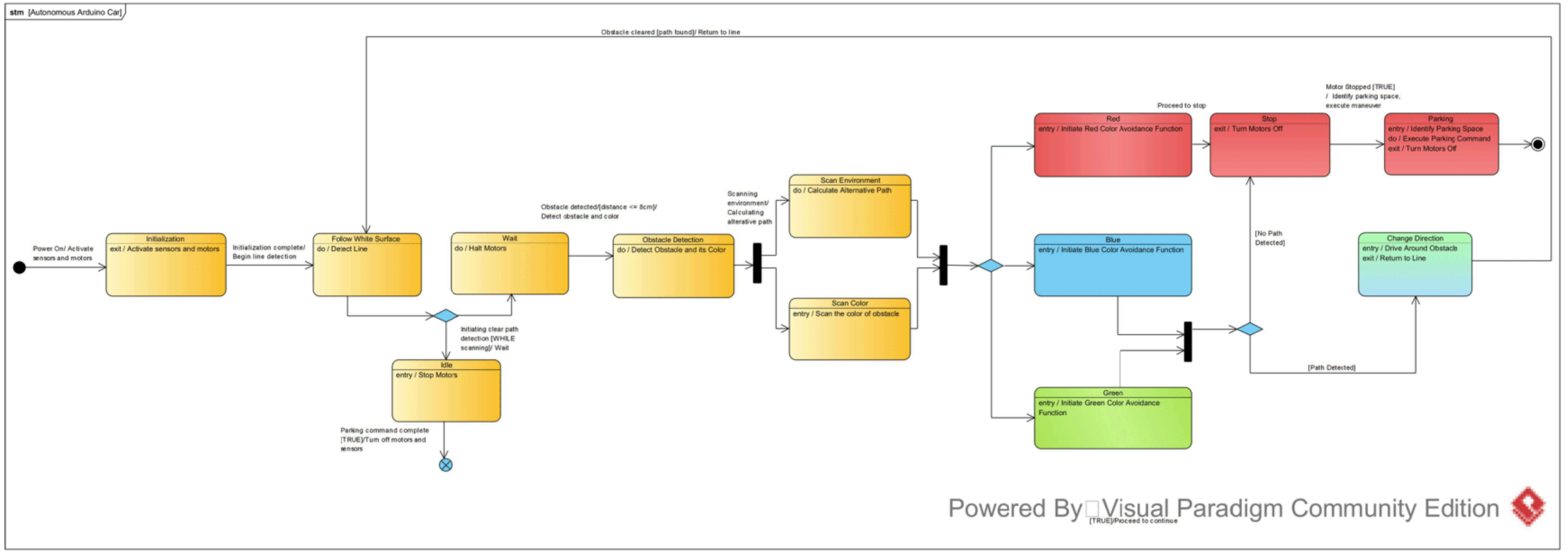




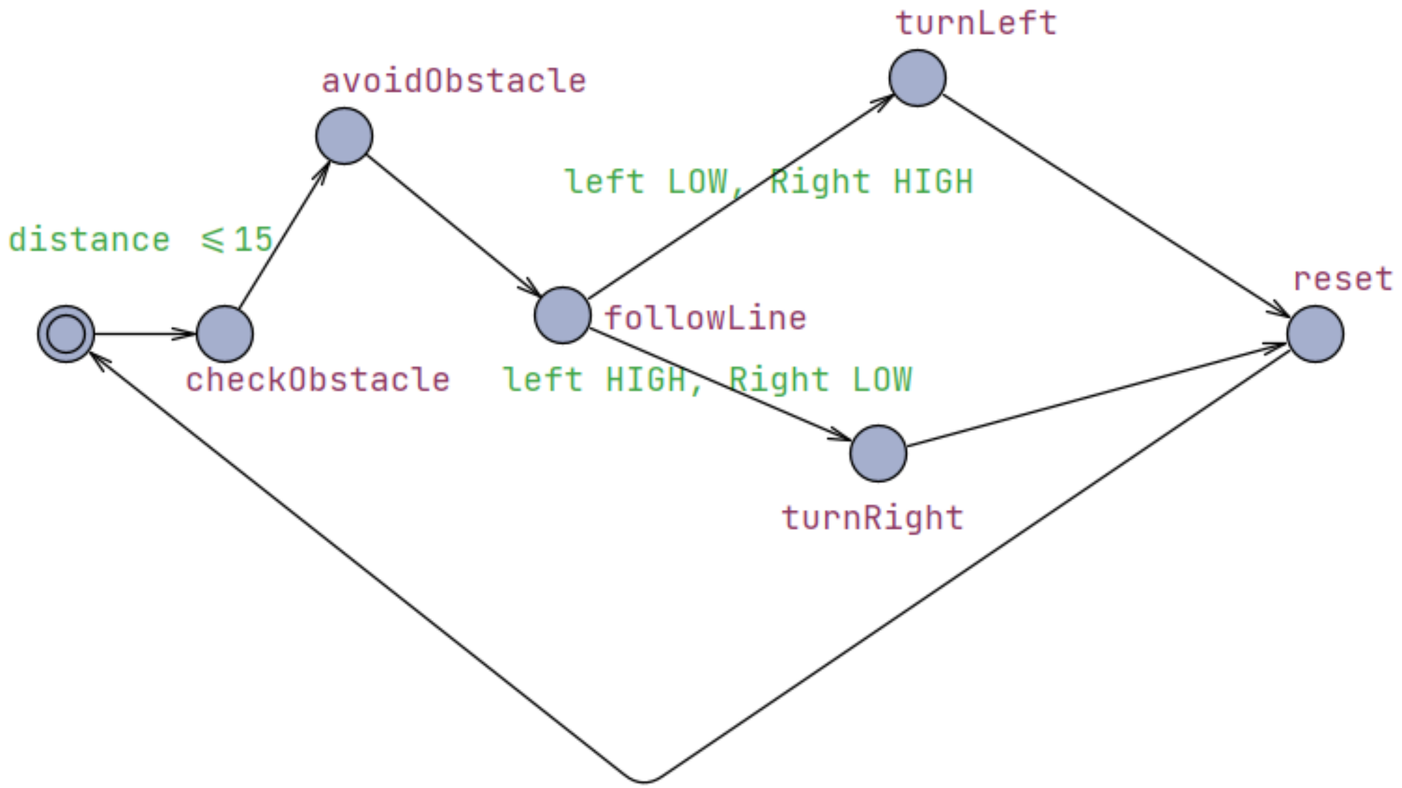
ACTIVITY DIAGRAM



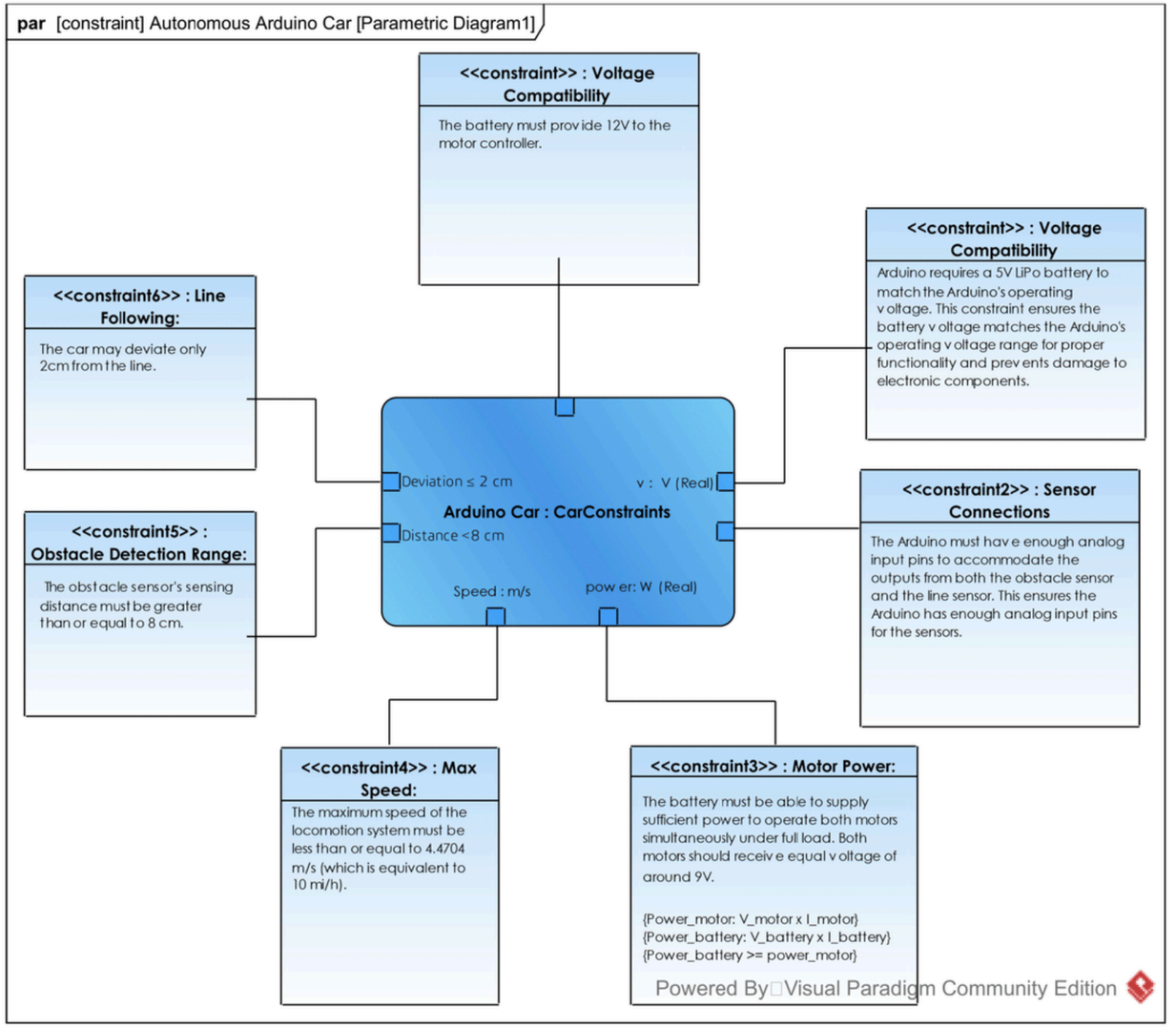
STATE CHART


stm [Autonomous Arduino Car]
Obstacle cleared [path found]/ Return to line
Power On/ Activate sensors and motors
Initialization
exit / Activate sensors and motors
Initialization complete/ Begin line detection
Follow White Surface
do / Deect Line
Wait
do / Halt Motors
Obstacle detected/[distance <= 8cm]/ Detect obstacle and color
Obstacle Detection
do / Detect Obstacle and its Color
Scanning environment/ Calculating alterative path
Scan Environment
do / Calculate Alternative Path
Scan Color
entry / Scan the color of obstacle
Initiating clear path detection [WHILE scanning]/ Wait
Idle
entry / Stop Motors
Parking command complete [TRUE]/Turn off motors and sensors
Proceed to stop
Red
entry / Initiate Red Color Avoidance Function
Stop
exit / Turn Motors Off
Motor Stopped [TRUE] / Identify parking space, execute maneuver
Parking
entry / Identify Parking Space
do / Execute Parking Command
exit / Turn Motors Off
Blue
entry / Initiate Blue Color Avoidance Function
[No Path Detected]
Change Direction
entry / Drive Around Obstacle
exit / Return to Line
[Path Detected]
Green
entry / Initiate Green Color Avoidance Function
[TRUE]/Proceed to continue
Powered By Visual Paradigm Community Edition

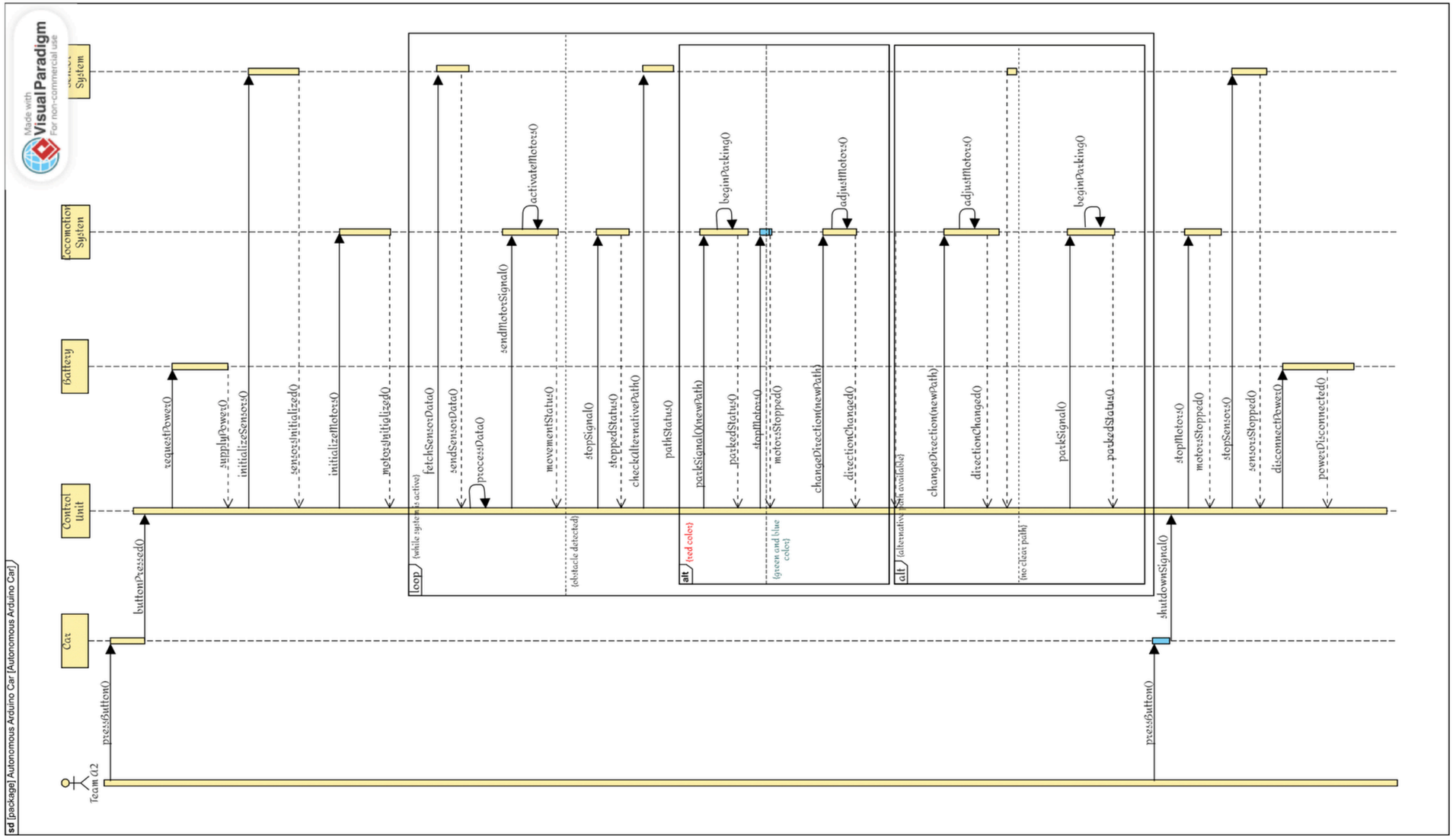
STATE MACHINE: UPPAAL



PARAMETRIC DIAGRAM



SEQUENCE DIAGRAM



CODE SOLUTION

```
/***************************************
 * Line Following + Obstacle Avoidance +
 * Color Detection
 * Group A2 - Prototyping
 * SS25 HSHL
 ***************************************/

// Pin Definitions

// Motor Definitions
#define ENA 10   // Right motor speed
#define IN1 4    // Right motor direction
#define IN2 5    // Right motor direction

#define ENB 11   // Left motor speed
#define IN3 6    // Left motor direction
#define IN4 7    // Left motor direction

// IR Sensors
#define IR_LEFT 12
#define IR_RIGHT 13

// Ultrasonic sensor
#define TRIG_PIN 9
#define ECHO_PIN 8

// TCS3200 Color Sensor pins
#define S0 2
#define S1 3
#define S2 A0
#define S3 A1
#define OUT A2
#define OE A3  // Output Enable (LOW = sensor ON)
#define LED A4 // LED control (HIGH = ON)

// Adjustable speed
int motorSpeed = 65; // 0-255

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

  // Motor pins
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENB, OUTPUT);
```

```
  // IR Sensors
  pinMode(IR_LEFT, INPUT);
  pinMode(IR_RIGHT, INPUT);

  // Ultrasonic
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  // TCS3200
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  pinMode(OE, OUTPUT);
  pinMode(LED, OUTPUT);
  pinMode(OUT, INPUT);

  // Configure TCS3200 frequency scaling
  digitalWrite(S0, HIGH);
  digitalWrite(S1, HIGH);
  digitalWrite(OE, LOW);    // Enable output
  digitalWrite(LED, HIGH);  // Turn on LEDs

  stopMotors();
}

void loop() {
  long distance = getDistance();

  if (distance < 15 && distance > 0) {
    stopMotors();
    delay(200);

    detectColor();
    avoidObstacle();
  } else {
    followLine();
  }
}
```

```
// Modular

// Line Following

void followLine() {
  int leftIR = digitalRead(IR_LEFT);
  int rightIR = digitalRead(IR_RIGHT);

  // Case 1: Both LOW -> Both motors forward
  if (leftIR == LOW && rightIR == LOW) {
    moveForward();
  }
  // Case 2: Both HIGH -> Both motors stopped
  else if (leftIR == HIGH && rightIR == HIGH) {
    stopMotors();
  }
  // Case 3: Mismatch - run motor on side with LOW sensor only
  else {
    if (leftIR == LOW && rightIR == HIGH) {
      runRightMotor();
      stopLeftMotor();
    }
    else if (leftIR == HIGH && rightIR == LOW) {
      runLeftMotor();
      stopRightMotor();
    }
  }
}

// Motor control

void moveForward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, motorSpeed);

  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENB, motorSpeed);
}

void stopMotors() {
  stopLeftMotor();
  stopRightMotor();
}

void runLeftMotor() {
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, motorSpeed);
}
```

```

void stopLeftMotor() {
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  analogWrite(ENB, 0);
}

void runRightMotor() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, motorSpeed);
}

void stopRightMotor() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  analogWrite(ENA, 0);
}

// Ultrasonic sensor

long getDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH, 20000);
  long distance = duration * 0.034 / 2;
  return distance;
}

// Color detection

void detectColor() {
  Serial.println("Detecting color...");

  int red = readColor(LOW, LOW);
  int green = readColor(HIGH, HIGH);
  int blue = readColor(LOW, HIGH);

  Serial.print("R: ");
  Serial.print(red);
  Serial.print(" G: ");
  Serial.print(green);
  Serial.print(" B: ");
  Serial.println(blue);
```

CODE SOLUTION

```
  if (red < green && red < blue) {
    Serial.println("Detected: RED");
  } else if (green < red && green < blue) {
    Serial.println("Detected: GREEN");
  } else if (blue < red && blue < green) {
    Serial.println("Detected: BLUE");
  } else {
    Serial.println("Unknown Color");
  }
}

int readColor(bool s2, bool s3) {
  digitalWrite(S2, s2);
  digitalWrite(S3, s3);
  delay(50);
  return pulseIn(OUT, LOW);
}

// Obstacle avoidance

void avoidObstacle() {
  Serial.println("Scanning for clear path...");

  // Turn left to check
  runLeftMotor();
  runRightMotorReverse();
  delay(500);
  stopMotors();
  delay(200);

  if (getDistance() > 15) {
    Serial.println("Path clear left");
    runLeftMotor();
    runRightMotorReverse();
    delay(500);
    return;
  }

  // Turn right to check
  runLeftMotorReverse();
  runRightMotor();
  delay(1000);
  stopMotors();
  delay(200);

  if (getDistance() > 15) {
    Serial.println("Path clear right");
    runLeftMotorReverse();
    runRightMotor();
    delay(500);
    return;
  }
```

```
  }

  // Stuck - stop
  stopMotors();
  Serial.println("No clear path found");
}

// Helper functions for reverse motors (used in avoidance)

void runLeftMotorReverse() {
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  analogWrite(ENB, motorSpeed);
}

void runRightMotorReverse() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  analogWrite(ENA, motorSpeed);
}
```

TEAM OVERVIEW

Muhammad Ali: 3D Modeling in AutoCAD, Vehicle Assembly, Initial Component Testing, Coding

Moeez Mufti: GitHub Moderation, UML Diagrams, Version Control, Component Testing, Vehicle Assembly