**Smart City - C Project**

**1. Code to implement the line following robot(car), along with the BLYNK Application controlling it’s destinations.**

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

#include <BlynkSimpleEsp8266.h>

// Wi-Fi credentials

const char\* ssid = "David's Galaxy S20 FE 5G"; // Mobile hotspot SSID

const char\* password = "ownp4619"; // Mobile hotspot password

#define BLYNK\_TEMPLATE\_ID "TMPL3jaPXoxLy"

#define BLYNK\_TEMPLATE\_NAME "Smart Car Controller"

#define BLYNK\_AUTH\_TOKEN "Ns5z6zhJkvEhP\_EpXdPAAWxUJybi4KGz"

// Pin configuration

#define IR\_LEFT D6

#define IR\_RIGHT D5

#define IN1 D1

#define IN2 D2

#define IN3 D3

#define IN4 D4

#define TRIG\_PIN D7

#define ECHO\_PIN 3

#define EN\_A D0

#define EN\_B D8

// Variables

String trafficStatus = "RED";

long distanceThreshold = 20;

int speedPWM = 150;

bool startProgram = false;

// Blynk Virtual Pin handler for Start button

BLYNK\_WRITE(V0) {

startProgram = param.asInt();

if (startProgram) {

Serial.println("Program Started!");

} else {

stopMotors();

Serial.println("Program Stopped!");

}

}

void setup() {

Serial.begin(115200);

// Pin modes

pinMode(IR\_LEFT, INPUT);

pinMode(IR\_RIGHT, INPUT);

pinMode(IN1, OUTPUT);

pinMode(IN2, OUTPUT);

pinMode(IN3, OUTPUT);

pinMode(IN4, OUTPUT);

pinMode(TRIG\_PIN, OUTPUT);

pinMode(ECHO\_PIN, INPUT);

pinMode(EN\_A, OUTPUT);

pinMode(EN\_B, OUTPUT);

stopMotors();

setSpeed(speedPWM);

WiFi.begin(ssid, password);

Serial.println("Connecting to Wi-Fi...");

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.print(".");

}

Serial.println("\nConnected to Wi-Fi!");

Serial.println("IP Address: ");

Serial.println(WiFi.localIP());

Blynk.begin(auth, ssid, password);

}

void loop() {

Blynk.run();

if (startProgram) {

if (WiFi.status() == WL\_CONNECTED) {

WiFiClient client;

HTTPClient http;

http.begin(client, "http://192.168.24.174/");

int httpCode = http.GET();

if (httpCode == 200) {

trafficStatus = http.getString();

Serial.println("Traffic Status: " + trafficStatus);

} else {

Serial.println("Failed to get traffic status.");

}

http.end();

}

long distance = getDistance();

Serial.println("Distance: " + String(distance) + " cm");

if (trafficStatus == "GREEN" && distance > distanceThreshold) {

followLine();

} else if (distance <= distanceThreshold) {

stopMotors();

Serial.println("Obstacle detected!");

} else {

stopMotors();

}

delay(1000);

}

}

// Ultrasonic distance measurement

long getDistance() {

digitalWrite(TRIG\_PIN, LOW);

delayMicroseconds(2);

digitalWrite(TRIG\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIG\_PIN, LOW);

long duration = pulseIn(ECHO\_PIN, HIGH); // Use the RX pin to read echo

long distance = duration \* 0.034 / 2; // Convert to cm

return distance;

}

// Line-following logic

void followLine() {

int leftSensor = digitalRead(IR\_LEFT);

int rightSensor = digitalRead(IR\_RIGHT);

if (leftSensor == LOW && rightSensor == LOW) {

stopMotors();

} else if (leftSensor == LOW && rightSensor == HIGH) {

turnRight();

} else if (leftSensor == HIGH && rightSensor == LOW) {

turnLeft();

} else {

moveForward();

}

}

void moveForward() {

setSpeed(150);

digitalWrite(IN1, LOW);

digitalWrite(IN2, HIGH);

digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

}

void turnRight() {

setSpeed(250);

digitalWrite(IN1, LOW);

digitalWrite(IN2, HIGH);

digitalWrite(IN3, LOW);

digitalWrite(IN4, LOW);

}

void turnLeft() {

setSpeed(250);

digitalWrite(IN1, LOW);

digitalWrite(IN2, LOW);

digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

}

void stopMotors() {

setSpeed(150);

digitalWrite(IN1, LOW);

digitalWrite(IN2, LOW);

digitalWrite(IN3, LOW);

digitalWrite(IN4, LOW);

}

// Set motor speed using PWM

void setSpeed(int pwmValue) {

analogWrite(EN\_A, pwmValue);

analogWrite(EN\_B, pwmValue);

}

Explanation:-

**Traffic Light Code:**

#include <ESP8266WiFi.h>

#include <ESP8266WebServer.h>

#define GREEN\_LIGHT\_PIN D6

#define GREEN\_LIGHT\_PIN2 D7

#define RED\_LIGHT\_PIN D5

#define RED\_LIGHT\_PIN2 D3

#define echo1 D1

#define echo2 D2

#define INPUT\_PIN D8

const char\* ssid = "David's Galaxy S20 FE 5G"; // Mobile hotspot SSID

const char\* password = "ownp4619"; // Mobile hotspot password

ESP8266WebServer server(80); // HTTP server on port 80

String trafficStatus = "RED"; // Default status

unsigned long previousMillis = 0; // Stores the last time the light was toggled

const unsigned long greenDuration = 10000; // Green light duration (ms)

const unsigned long redDuration = 10000; // Red light duration (ms)

bool isGreen = false; // Tracks the current light state

void setup() {

Serial.begin(115200);

pinMode(GREEN\_LIGHT\_PIN, OUTPUT);

pinMode(RED\_LIGHT\_PIN, OUTPUT);

pinMode(GREEN\_LIGHT\_PIN2, OUTPUT);

pinMode(RED\_LIGHT\_PIN2, OUTPUT);

pinMode(echo1,INPUT);

pinMode(echo2,INPUT);

pinMode(INPUT\_PIN, INPUT);

// Connect to Wi-Fi

WiFi.begin(ssid, password);

Serial.println("Connecting to Wi-Fi...");

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.print(".");

}

Serial.println("\nConnected to Wi-Fi!");

Serial.println("IP Address: ");

Serial.println(WiFi.localIP());

// Configure HTTP server routes

server.on("/", HTTP\_GET, []() {

server.send(200, "text/plain", trafficStatus); // Send traffic light status

});

server.begin();

Serial.println("HTTP server started.");

}

void loop() {

int signal = digitalRead(INPUT\_PIN);

long duration = pulseIn(echo1, HIGH);

long distance = duration \* 0.034 / 2; // Convert to cm

Serial.println(distance);

long duration1 = pulseIn(echo2, HIGH);

long distance1 = duration1 \* 0.034 / 2; // Convert to cm

Serial.println(distance1);

unsigned long currentMillis = millis(); // Get current time in milliseconds

// Toggle traffic lights based on elapsed time

if(signal == HIGH){

digitalWrite(GREEN\_LIGHT\_PIN, HIGH);

digitalWrite(GREEN\_LIGHT\_PIN2, HIGH);

digitalWrite(RED\_LIGHT\_PIN, LOW);

digitalWrite(RED\_LIGHT\_PIN2, LOW);

}

else{

if(distance<50 && distance1> 50){

digitalWrite(GREEN\_LIGHT\_PIN, HIGH);

digitalWrite(GREEN\_LIGHT\_PIN2, HIGH);

digitalWrite(RED\_LIGHT\_PIN, LOW);

digitalWrite(RED\_LIGHT\_PIN2, LOW);

trafficStatus = "GREEN";

}

else if(distance>50 and distance1<50){

digitalWrite(GREEN\_LIGHT\_PIN, LOW);

digitalWrite(GREEN\_LIGHT\_PIN2, LOW);

digitalWrite(RED\_LIGHT\_PIN, HIGH);

digitalWrite(RED\_LIGHT\_PIN2, HIGH);

trafficStatus = "GREEN";

}

else{

if (isGreen && currentMillis - previousMillis >= greenDuration) {

// Switch to RED light

digitalWrite(GREEN\_LIGHT\_PIN, LOW);

digitalWrite(GREEN\_LIGHT\_PIN2, LOW);

digitalWrite(RED\_LIGHT\_PIN, HIGH);

digitalWrite(RED\_LIGHT\_PIN2, HIGH);

trafficStatus = "RED";

isGreen = false;

previousMillis = currentMillis; // Reset timer

} else if (!isGreen && currentMillis - previousMillis >= redDuration) {

// Switch to GREEN light

digitalWrite(GREEN\_LIGHT\_PIN, HIGH);

digitalWrite(GREEN\_LIGHT\_PIN2, HIGH);

digitalWrite(RED\_LIGHT\_PIN, LOW);

digitalWrite(RED\_LIGHT\_PIN2, LOW);

trafficStatus = "GREEN";

isGreen = true;

previousMillis = currentMillis; // Reset timer

}}

if(distance>50){

trafficStatus = "GREEN";

}}

// Handle HTTP requests

server.handleClient();

}

**RFID Code:**

#include <SPI.h>

#include <MFRC522.h>

// Pins for RFID module

#define RST\_PIN 5 // Reset pin

#define SS\_PIN 7 // SDA pin (Chip Select)

#define OUTPUT\_PIN A0 // Pin to send HIGH signal if RFID is authorized

// RFID module initialization

MFRC522 rfid(SS\_PIN, RST\_PIN);

// Define the UID of the authorized RFID card

byte authorizedUID[] = {0x4A, 0xA0, 0xB5, 0x15}; // Replace with your card's UID

const int UID\_LENGTH = 4; // Length of the UID

// Pins for Ultrasonic Sensor 1

const int trigPin1 = 9; // Trig pin for Sensor 1

const int echoPin1 = 10; // Echo pin for Sensor 1

// Pins for Ultrasonic Sensor 2

const int trigPin2 = 3; // Trig pin for Sensor 2

const int echoPin2 = 6; // Echo pin for Sensor 2

// Pins for LEDs

const int led1 = 8; // LED for Sensor 1

const int led2 = 4; // LED for Sensor 2

const int led3 = 2; // LED triggered after 25 seconds of close distance

unsigned long sensor2StartTime = 0; // Time when Sensor 2 first detects <40 cm

bool sensor2Triggered = false;

void setup() {

// Initialize Serial Monitor

Serial.begin(9600);

// Initialize RFID module

SPI.begin();

rfid.PCD\_Init();

pinMode(OUTPUT\_PIN, OUTPUT);

digitalWrite(OUTPUT\_PIN, LOW); // Ensure the output pin starts LOW

// Initialize Ultrasonic Sensor pins

pinMode(trigPin1, OUTPUT);

pinMode(echoPin1, INPUT);

pinMode(trigPin2, OUTPUT);

pinMode(echoPin2, INPUT);

// Initialize LED pins

pinMode(led1, OUTPUT);

pinMode(led2, OUTPUT);

pinMode(led3, OUTPUT);

digitalWrite(led1, LOW);

digitalWrite(led2, LOW);

digitalWrite(led3, LOW);

Serial.println("Place your RFID card near the reader...");

}

void loop() {

// RFID logic

handleRFID();

// Ultrasonic sensor logic

handleUltrasonicSensors();

delay(500); // Wait for 500ms before the next cycle

}

// Function to handle RFID logic

void handleRFID() {

// Look for a card

if (!rfid.PICC\_IsNewCardPresent() || !rfid.PICC\_ReadCardSerial()) {

return; // No card detected

}

// Print the UID of the detected card

Serial.print("Card UID: ");

for (byte i = 0; i < rfid.uid.size; i++) {

Serial.print(rfid.uid.uidByte[i], HEX);

Serial.print(" ");

}

Serial.println();

// Check if the detected UID matches the authorized UID

if (isAuthorizedCard()) {

Serial.println("Authorized Card Detected!");

digitalWrite(OUTPUT\_PIN, HIGH); // Send HIGH signal through A0

delay(10000); // Keep HIGH for 10 seconds

digitalWrite(OUTPUT\_PIN, LOW); // Set A0 back to LOW

} else {

Serial.println("Unauthorized Card!");

}

rfid.PICC\_HaltA(); // Halt the card

}

// Function to handle Ultrasonic Sensor logic

void handleUltrasonicSensors() {

// Measure distance for Sensor 1

float distance1 = measureDistance(trigPin1, echoPin1);

// Measure distance for Sensor 2

float distance2 = measureDistance(trigPin2, echoPin2);

// Control LED1 based on Sensor 1 distance

if (distance1 < 100) {

digitalWrite(led1, HIGH);

} else {

digitalWrite(led1, LOW);

}

// Control LED2 based on Sensor 2 distance

if (distance2 < 100) {

digitalWrite(led2, HIGH);

} else {

digitalWrite(led2, LOW);

}

// Control LED3 if Sensor 2 detects <40 cm for 25 seconds

if (distance2 < 40) {

if (!sensor2Triggered) {

sensor2StartTime = millis();

sensor2Triggered = true;

} else if (millis() - sensor2StartTime >= 25000) {

digitalWrite(led3, HIGH);

}

} else {

sensor2Triggered = false;

digitalWrite(led3, LOW);

}

// Print the distances to the Serial Monitor

Serial.print("Sensor 1 Distance: ");

Serial.print(distance1);

Serial.println(" cm");

Serial.print("Sensor 2 Distance: ");

Serial.print(distance2);

Serial.println(" cm");

}

// Function to measure distance using an ultrasonic sensor

float measureDistance(int trigPin, int echoPin) {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

long duration = pulseIn(echoPin, HIGH);

return (duration \* 0.0343) / 2; // Convert duration to distance in cm

}

// Function to check if the detected card matches the authorized UID

bool isAuthorizedCard() {

if (rfid.uid.size != UID\_LENGTH) {

return false; // UID length mismatch

}

for (byte i = 0; i < UID\_LENGTH; i++) {

if (rfid.uid.uidByte[i] != authorizedUID[i]) {

return false; // UID mismatch

}

}

return true; // UID matches

}

**Smart Dustbin**

#include <Servo.h>

Servo myServo;

const int trigPin = 8;

const int echoPin = 7;

const int ir = 13;

void setup() {

  myServo.attach(9);

  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);

  Serial.begin(9600);

  pinMode(ir,INPUT);

}

void loop() {

  int irread = digitalRead(ir);

  long duration, distance;

  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);

  distance = duration \* 0.034 / 2;

  Serial.print("Distance: ");

  Serial.print(distance);

  Serial.println(" cm");

  Serial.println(irread);

if(irread==1){

  if (distance > 0 && distance <= 10) {

    Serial.println("Object detected! Moving servo.");

    myServo.write(180);

    delay(1000);

  } else {

    Serial.println("No object detected.");

    myServo.write(0);

  }}

  else{

    myServo.write(0);

  }

  delay(500);

}