CODES FOR ELECTRIC CAR

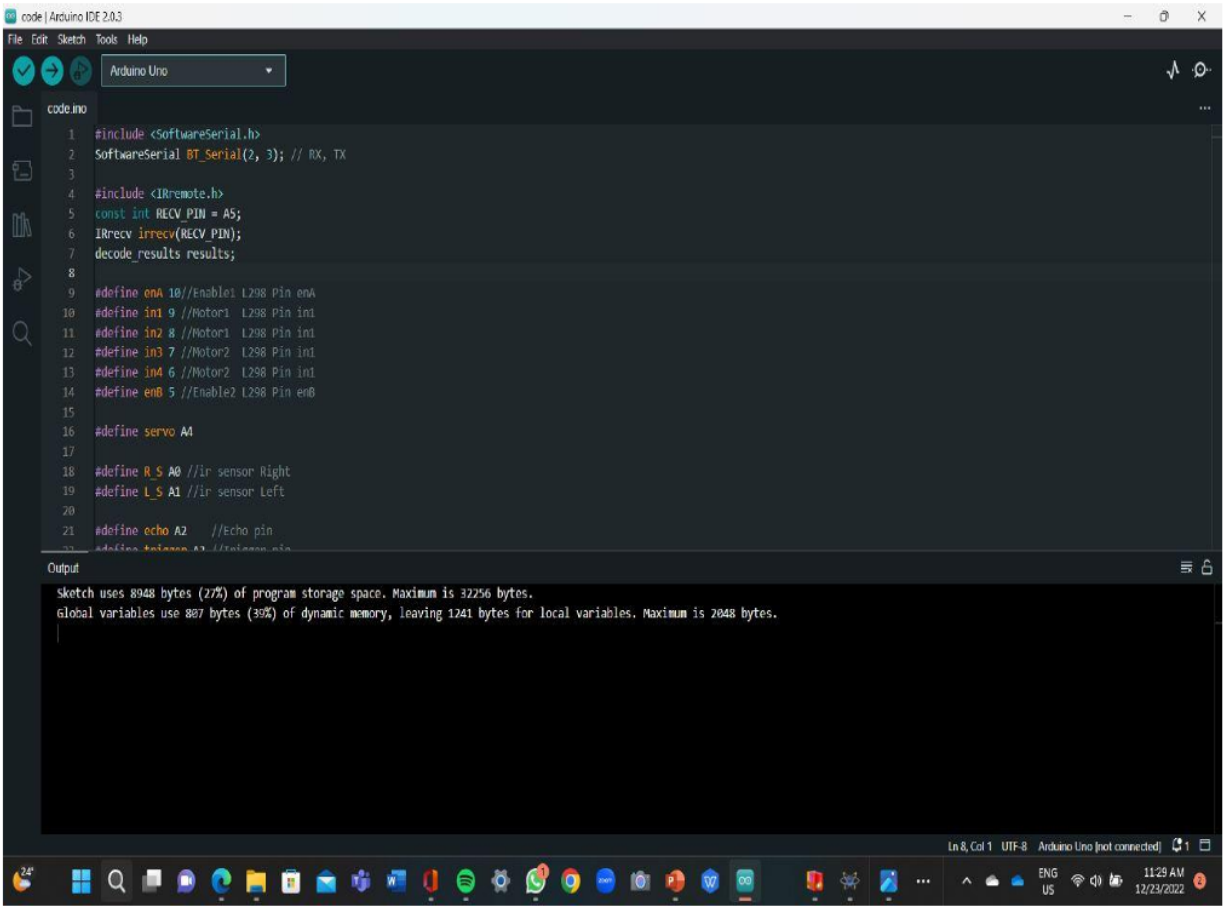
```
#include <SoftwareSerial.h>
SoftwareSerial BT_Serial(2, 3); // RX, TX

#include <IRremote.h>
const int RECV_PIN = A5;
IRrecv irrecv(RECV_PIN);
decode_results results;

#define enA 10//Enable1 L298 Pin enA
#define in1 9 //Motor1  L298 Pin in1
#define in2 8 //Motor1  L298 Pin in1
#define in3 7 //Motor2  L298 Pin in1
#define in4 6 //Motor2  L298 Pin in1
#define enB 5 //Enable2 L298 Pin enB

#define servo A4
#define R_S A0 //ir sensor Right
#define L_S A1 //ir sensor Left
#define echo A2    //Echo pin
#define trigger A3 //Trigger pin

int distance_L, distance_F = 30, distance_R;
long distance;
int set = 20;
int bt_ir_data; // variable to receive data from the serial port and IRremote
int Speed = 130;
int mode=0;
int IR_data;

void setup(){ // put your setup code here, to run once

pinMode(R_S, INPUT); // declare if sensor as input
pinMode(L_S, INPUT); // declare ir sensor as input

pinMode(echo, INPUT );// declare ultrasonic sensor Echo pin as input
pinMode(trigger, OUTPUT); // declare ultrasonic sensor Trigger pin as Output

pinMode(enA, OUTPUT); // declare as output for L298 Pin enA
pinMode(in1, OUTPUT); // declare as output for L298 Pin in1
pinMode(in2, OUTPUT); // declare as output for L298 Pin in2
pinMode(in3, OUTPUT); // declare as output for L298 Pin in3
pinMode(in4, OUTPUT); // declare as output for L298 Pin in4
pinMode(enB, OUTPUT); // declare as output for L298 Pin enB

irrecv.enableIRIn(); // Start the receiver
irrecv.blink13(true);
Serial.begin(9600); // start serial communication at 9600bps
BT_Serial.begin(9600);

pinMode(servo, OUTPUT);
```

```
 for (int angle = 70; angle <= 140; angle += 5)  {
   servoPulse(servo, angle);  }
 for (int angle = 140; angle >= 0; angle -= 5)  {
   servoPulse(servo, angle);  }

 for (int angle = 0; angle <= 70; angle += 5)  {
   servoPulse(servo, angle);  }
delay(500);
}

void loop(){

if(BT_Serial.available() > 0){  //if some date is sent, reads it and saves in state
bt_ir_data = BT_Serial.read();
Serial.println(bt_ir_data);
if(bt_ir_data > 20){Speed = bt_ir_data;}
}

if (irrecv.decode(&results))
{
Serial.println(results.value,HEX);
bt_ir_data = IRremote_data();
Serial.println(bt_ir_data);
irrecv.resume(); // Receive the next value
delay(100);
}

if(bt_ir_data == 8){mode=0; Stop();}    //Manual Android Application and IR Remote Control Command
else if(bt_ir_data == 9){mode=1; Speed=130;} //Auto Line Follower Command
else if(bt_ir_data ==10){mode=2; Speed=255;} //Auto Obstacle Avoiding Command analogWrite(enA, Speed); // Write The Duty Cycle 0 to 255 Enable Pin A for Motor1 Speed analogWrite(enB, Speed); // Write The Duty Cycle 0 to 255 Enable Pin B for Motor2 Speed
if(mode==0)
{
//================================================================================
//                    Key Control Command
//================================================================================
if(bt_ir_data == 1){forword(); }  // if the bt_data is '1' the DC motor will go forward
else if(bt_ir_data == 2){backword();}  // if the bt_data is '2' the motor will Reverse
else if(bt_ir_data == 3){turnLeft();}  // if the bt_data is '3' the motor will turn left
else if(bt_ir_data == 4){turnRight();} // if the bt_data is '4' the motor will turn right
else if(bt_ir_data == 5){Stop();
}
// if the bt_data '5' the motor will Stop
//================================================================================
//                    Voice Control Command
//================================================================================
else if(bt_ir_data == 6){turnLeft();  delay(400);  bt_ir_data = 5;
}
else if(bt_ir_data == 7){turnRight(); delay(400);  bt_ir_data = 5;}
}

if(mode==1){
//================================================================================
//                    Line Follower Control
//================================================================================
```

```
if((digitalRead(R_S) == 0)&&(digitalRead(L_S) == 0)){forword();}  //if Right Sensor and Left Sensor are at White color
then it will call forword function
if((digitalRead(R_S) == 1)&&(digitalRead(L_S) == 0)){turnRight();}//if Right Sensor is Black and Left Sensor is White then
it will call turn Right function
if((digitalRead(R_S) == 0)&&(digitalRead(L_S) == 1)){turnLeft();} //if Right Sensor is White and Left Sensor is Black then
it will call turn Left function
if((digitalRead(R_S) == 1)&&(digitalRead(L_S) == 1)){Stop();}     //if Right Sensor and Left Sensor are at Black color then
it will call Stop function
}

if(mode==2){
//================================================================================
//                       Obstacle Avoiding Control
//================================================================================
 distance_F = Ultrasonic_read();
 Serial.print("S=");Serial.println(distance_F);
  if (distance_F > set){forword();}
    else{Check_side();}
}

delay(10);
}

long IRremote_data()
{
     if(results.value==0xFF02FD){IR_data=1;}
else if(results.value==0xFF9867){IR_data=2;}
else if(results.value==0xFFE01F){IR_data=3;}
else if(results.value==0xFF906F){IR_data=4;}
else if(results.value==0xFF629D || results.value==0xFFA857){IR_data=5;}
else if(results.value==0xFF30CF){IR_data=8;}
else if(results.value==0xFF18E7){IR_data=9;}
else if(results.value==0xFF7A85){IR_data=10;}
return IR_data;
}

void servoPulse (int pin, int angle){
int pwm = (angle*11) + 500;      // Convert angle to microseconds
 digitalWrite(pin, HIGH);
 delayMicroseconds(pwm);
 digitalWrite(pin, LOW);
 delay(50);                 // Refresh cycle of servo
}


//**********************Ultrasonic_read*****************************
long Ultrasonic_read(){
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  distance = pulseIn (echo, HIGH);
  return distance / 29 / 2;
}

void compareDistance(){
```

```
      if (distance_L > distance_R){
  turnLeft();
delay(350);
  }
  else if (distance_R > distance_L){
  turnRight();
  delay(350);
  }
  else{
  backword();
  delay(300);
  turnRight();
  delay(600);
  }
}

void Check_side(){
    Stop();
    delay(100);
 for (int angle = 70; angle <= 140; angle += 5)  {
   servoPulse(servo, angle);  }
    delay(300);
    distance_L = Ultrasonic_read();
    delay(100);
  for (int angle = 140; angle >= 0; angle -= 5)  {
   servoPulse(servo, angle);  }
    delay(500);
distance_R = Ultrasonic_read();
    delay(100);
 for (int angle = 0; angle <= 70; angle += 5)  {
   servoPulse(servo, angle);  }
    delay(300);
    compareDistance();
}

void forword(){  //forword
digitalWrite(in1, HIGH); //Right Motor forword Pin
digitalWrite(in2, LOW);  //Right Motor backword Pin
digitalWrite(in3, LOW);  //Left Motor backword Pin
digitalWrite(in4, HIGH); //Left Motor forword Pin
}

void backword(){ //backword
digitalWrite(in1, LOW);  //Right Motor forword Pin
digitalWrite(in2, HIGH); //Right Motor backword Pin
digitalWrite(in3, HIGH); //Left Motor backword Pin
digitalWrite(in4, LOW);  //Left Motor forword Pin
}

void turnRight(){ //turnRight
digitalWrite(in1, LOW);  //Right Motor forword Pin
digitalWrite(in2, HIGH); //Right Motor backword Pin
digitalWrite(in3, LOW);  //Left Motor backword Pin
digitalWrite(in4, HIGH); //Left Motor forword Pin
}
```

```
void turnLeft(){ //turnLeft
digitalWrite(in1, HIGH); //Right Motor forword Pin
digitalWrite(in2, LOW);  //Right Motor backword Pin
digitalWrite(in3, HIGH); //Left Motor backword Pin
digitalWrite(in4, LOW);  //Left Motor forword Pin
}

void Stop(){ //stop
digitalWrite(in1, LOW); //Right Motor forword Pin
digitalWrite(in2, LOW); //Right Motor backword Pin
digitalWrite(in3, LOW); //Left Motor backword Pin
digitalWrite(in4, LOW); //Left Motor forword Pin
}
```



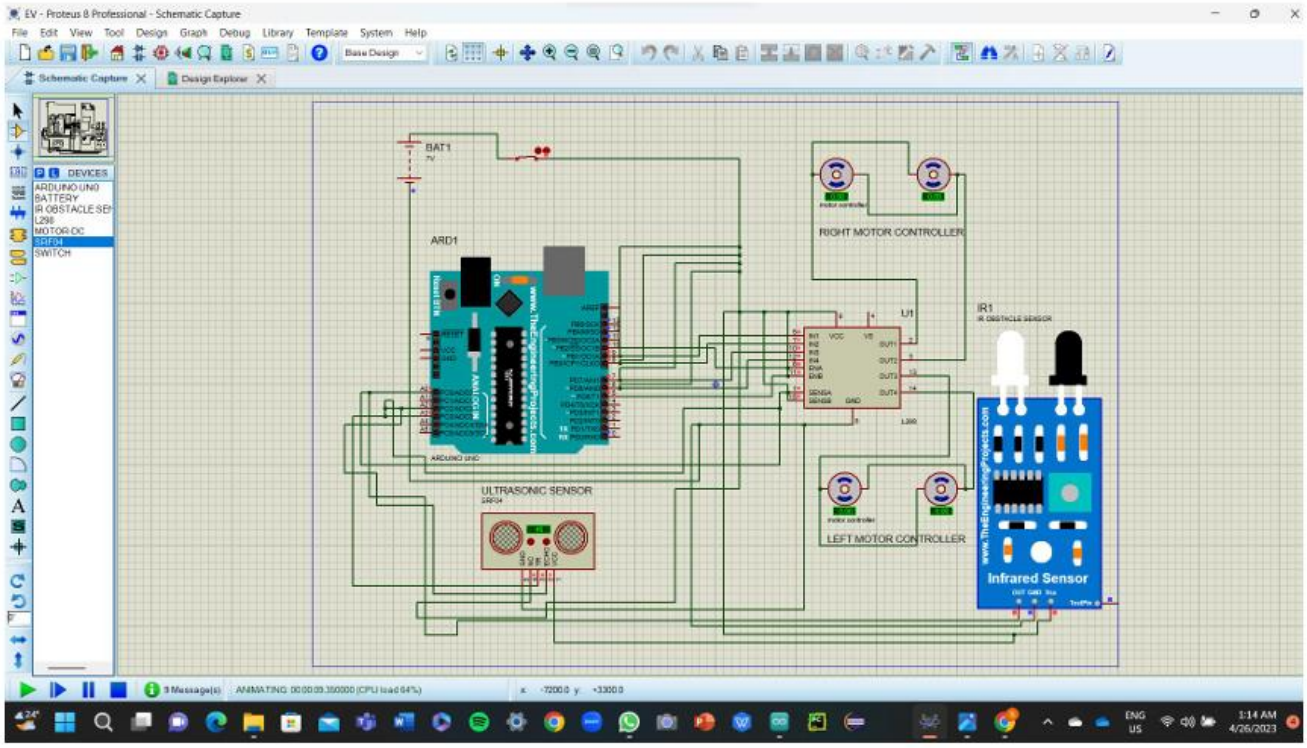
CODING OUTPUT

SIMULATION CODE:

```
#include <avr/io.h>
#include <util/delay.h>
#define TRIG_PIN PD0
#define ECHO_PIN PD1
#define IR_PIN PD2
#define IN1 PB0
```

```
#define IN2 PB1
#define IN3 PB2
#define IN4 PB3
int main(void) {
// Set up ports
DDRB = 0xFF;    // Set PB0 to PB3 as outputs
DDRD &= ~(1 << ECHO_PIN); // Set PD1 as input
DDRD |= (1 << TRIG_PIN); // Set PD0 as output
DDRD &= ~(1 << IR_PIN); // Set PD2 as input

// Set up timer1
TCCR1B |= (1 << CS11);  // Set prescaler to 8
TCNT1 = 0;  // Set timer1 counter to 0

while (1) {
// Send ultrasonic pulse
PORTD |= (1 << TRIG_PIN);
_delay_us(10);
PORTD &= ~(1 << TRIG_PIN);

// Wait for echo
while (!(PIND & (1 << ECHO_PIN)));
TCNT1 = 0;
while (PIND & (1 << ECHO_PIN));
uint16_t pulse_duration = TCNT1 / 2;

// Calculate distance in cm
uint16_t distance = pulse_duration / 29;

// Check IR sensor
uint8_t ir_sensor_value = PIND & (1 << IR_PIN);

// Move robot based on distance and IR sensor value
if (distance > 10 && ir_sensor_value) {
// Move forward
PORTB |= (1 << IN1);
PORTB &= ~(1 << IN2);
PORTB |= (1 << IN3);
PORTB &= ~(1 << IN4);
} else if (distance <= 10 && ir_sensor_value) {
// Turn left
PORTB &= ~(1 << IN1);
PORTB &= ~(1 << IN2);
PORTB |= (1 << IN3);
PORTB &= ~(1 << IN4);
_delay_ms(500);
// Move forward
PORTB |= (1 << IN1);
PORTB &= ~(1 << IN2);
```

```
PORTB |= (1 << IN3);
PORTB &= ~(1 << IN4);
} else if (distance > 10 && !ir_sensor_value) {
// Turn right
PORTB |= (1 << IN1);
PORTB &= ~(1 << IN2);
PORTB &= ~(1 << IN3);
PORTB &= ~(1 << IN4);
_delay_ms(500);
// Move forward
PORTB |= (1 << IN1);
PORTB &= ~(1 << IN2);
PORTB |= (1 << IN3);
PORTB &= ~(1 << IN4);
} else {
// Stop
PORTB &= ~(1 << IN1);
PORTB &= ~(1 << IN2);
PORTB &= ~(1 << IN3);
PORTB &= ~(1 << IN4);
}
}
return 0;
}
```



SIMULATION OUTPUT