# Autonomous Line following Vehicle

*Note: A university prototyping project by team 4

1st Hammad Karamat
*Electronic Engineering*
*Hochschule Hamm-Lippstadt*
Lippstadt, Germany
Hammad.karamat@stud.hshl.de

2nd Ismail Hossain
*Electronic Engineering*
*Hochschule Hamm-Lippstadt*
Lippstadt, Germany
ismail.hossain@stud.hshl.de

3rd Abu Sayem
*Electronic Engineering*
*Hochschule Hamm-Lippstadt*
Lippstadt, Germany
abu.sayem@stud.hshl.de

4th Subir Balo
*Electronic Engineering*
*Hochschule Hamm-Lippstadt*
Lippstadt, Germany
subir.Balo@stud.hshl.de

***Abstract*—In this paper we will discuss an autonomous robot car with multiple uses that has been designed and developed using Arduino. The autonomous car comes with motor driver that is what accelerates it and other features such as the ultrasonic sensor, infrared (IR) sensors, and a color sensor in order to enable it avoid obstacles, follow tracks and capture colors. In relation to the movements of the robot, the motor driver allows it to be moved in various directions namely forward, backward, left or right. To ensure that this device navigates its environment correctly without colliding with objects on its path as well as any other obstacles, ultrasonic sensor measures distance. These IR sensors are applied in line following so that the robot can autonomously follow a given path. A color sensor also helps identify specific colors for decision making by the robot concerning object colors around it. Being cost effective, efficiency and scalability solutions for education and prototyping have been achieved through Arduino programming which brings together all these sensors and actuators. Furthermore, this paper goes into hardware setup details including software algorithms alongside experimental results showing how effectively complex tasks can be performed by an autonomous robot car under a wide range of conditions. Developers will benefit from this paper because of its usability when one needs to build different navigation and sensing techniques that are self-reliant.**

## I. Introduction

Autonomous robotics is a relatively young intensively developing branch of engineering, computer science, and artificial intelligence fields, which implies the creation of equipment capable of performing work without any human intervention. This kind of robots has the capacity to transform sectors including production, supply chain, healthcare, and education in terms of productivity, accuracy, and security thus, an autonomous robot car becomes an effective educational objective and a basis for constructing more intricate forms of robotics.

This paper describes the development and construction of a self-operated robot car with features of a multi functional car with an Arduino control unit. The project integrates several key components: motor driver, ultrasonic sensor, infrared (IR) sensors, and a colour sensor. All contribute to the fact that the robot is able to move around, avoid obstacles, follow bi-colored lines (black and white) lines as well as identify the color of the objects.

Motor driver unit controls the movement of the robot which has the options of forward, backward, and right or left turns. Distance reader is controlled by ultrasonic sensor which gives distances, and they allow the robot to react to the obstacles in real time. The IR sensors are used for line following which makes the robot to have an ability to follow certain paths without the need of any external control. Also there is the colour sensor and this locates a particular colour, and therefore, helps the robot in making more evolved decisions vis a vis the colour of the objects on its path.

The paper will also discuss the future possibilities of this project which includes integrating Internet of things (IOT) to achieve solutions for modern problems,

Therefore, the main goal of this work is to design an open-sourced, low cost and multi-functional system that could be used as an educational and prototyping tool in autonomous robotics. The connections of a number of individual sensors and actuators with Arduino programming provides a fully fledged teaching along with experimentation facility to the students and hobbyists. Also, this compensation allows for fine-tuning of different algorithms and technologies associated with autonomous navigation and sensor fusion on this platform.

## II. Structure

This paper is structured as follows:

- **Section III** describes the items of the hardware and their Corresponding Diagrams.
- **Section IV** with regard to software implementation at the control algorithms and the signal conditioning for the sensors.
- **Section V**, Using CAD and other 3D designing methods to create 3D printed object to complete the prototype frame.
- **Section VI** concludes with the summary of the findings and improvements for the robot in the future.
- In the last section, **Appendix** provides with extra workings of coding and the full form of Code for the prototype

With this work, we would like to advance the discourse on robotics by showing an example of an efficient autonomous robot car and stressing on the numerous uses in these sort of systems.

## III. Hardware and their corresponding Diagrams

### A. Hardware

The prototype combines several electronic components, each serving specific functions and processing distinct properties. Here is a summary of the components:

- Arduino Uno WiFi Rev 2.
- 2 Infrared Sensors
- Ultrasonic Sensor (HCSR04)
- Color Sensor (TCS3200
- Motor Driver (L298N)
- 2 DC Motors.
- Battery

The description of each component is as followed:

- Arduino Uno WiFi Rev 2
  For this prototype, the Arduino Uno wifi rev 2 is the prototype's primary microcontroller; it is responsible for data acquisition of the sensors and the execution of operations involving the application of the actuators to initiate the right functioning for navigation. This development board integrates a microchip from Atmel AVR family which is called **ATmega4809** that has 8-bit RISC CPU core and is fulfilled with 6KB RAM, 48KB flash memory. The board includes total of 14 digital GPIO pins among which 6 come with PWM output and 6 of which are the analogue input pins, A0 to A5. The USB connection is used to establish the connection channel between PC and the device.

Fig. 1. Arduino [3]

- Infrared Sensors
  IR sensors are used at the front side of the car with the help of which it locates and follows the line. At the bottom of the automobile, there are IR sensors that illuminate the exterior through the emission of infrared light. Subsequently, through observing two states, good and bad, they are capable of perceiving the reflection of the light. The location of the car can then be ascertained by the IR sensor either on the line or not on the line. It mostly remains in the queue due to these sensors. Digital pins 7 and 8 of the microcontroller are interfaced with the signal pins of the IR sensors concerned.

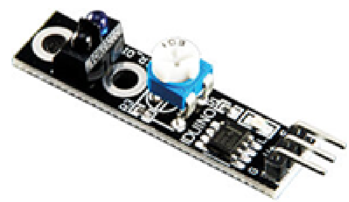

Fig. 2. Infrared Sensors [4]

- Ultrasonic Sensor (HCSR04)
  An ultrasonic sensor has been installed to help the car feel the environment in front of the car to avoid obstruction. It produces sound waves and measures the time interval for the waves reflected by the object to get back to the device. This is made possible by the use of the ultrasonic sensor close to the car; thus, the distance measured in terms of; the overall travel time will enable the identification of an obstruction. As for the connections of the sensor, its Trigger and Eco pins are assigned to pins 2 and 3 of the board.

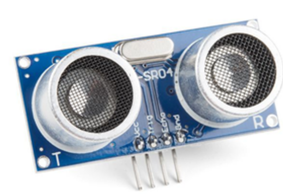

Fig. 3. Ultrasonic Sensor (HCSR04) [5]

- Color Sensor (TCS3200)
  To measure the color of the obstacles on the track a color sensor (TCS3200=color sensor) was employed. Depending on the color of the car, it would knock the object out of the track or bypass the object and go around it. The TCS3200 has white LEDs as the light source used to illuminate the surface of the object whose color needs to be measured. Their reflectance or the amount of light that bounces back from the object is determined. The converter produces a frequency proportional to the intensity which the microcontroller employs to estimate the object color.


Fig. 4. Color Sensor (TCS3200) [15]

Fig. 6. DC Motor [7]

- Motor Driver (L298N)
  A motor driver, L298N, also applied a for the purpose of both controlling and powering our vehicle's geared motors. A driver IC is needed to drive the Motors. After getting signals from the Arduino Uno, the driver IC works as a switch. Provided high is the signal, the driver IC drives the switch, hence providing the motor with the required voltage to make it rotate. With the L298N motor driver, there exist two enable inputs that allow a device to be enabled on or off when needed.

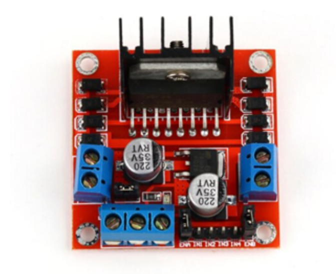

Fig. 5. Motor Driver (L298N) [6]

- DC Motors.
  It is also integrated with 2 DC Motor in the front side for the incorporation of the car wheels. The Modelcraft RB 35 gearbox with a motor can apply a considerable rotational force with its torque that is at 1:30. It may also bring down speed at the exact rate that the torque is and, therefore, when torque advances, the speed of the motor will also decrease at a proportionate rate, hence giving a well-balanced and efficient performance.

- Battery
  The car runs on a polymer Li-Ion battery that can be recharged. This specific battery gives the Arduino board, the motors, and the sensors the power they require. [8]


Fig. 7. Battery [8]

## B. Diagrams

- Block Diagram The block diagram indicates the

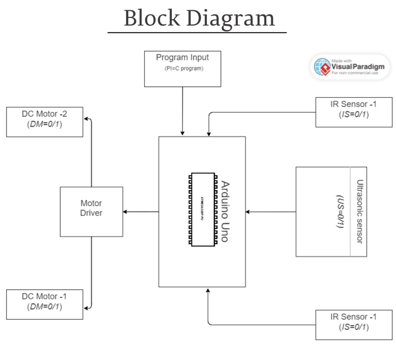


Fig. 8. Block diagram

architecture for a system of an autonomous robot car, indicating the main components, controlled using an Arduino Uno microcontroller board (section III, hardware). At the very center is an Arduino Uno, representing the central system for control. Using C language as program Input, making it take a number of inputs from the different sensors and drive outputs to control the motors. On the right-hand side of the diagram, there are two different sensors attached to the Arduino Uno board: an Ultrasonic sensor and two IR sensors, denoted as IR Sensor -1 and IR Sensor -2. This ultrasonic sensor gives the distance measurement from the nearby obstacles; hence, it helps the robot navigate without collision. IR sensors detect lines or obstacles on the path and generate binary signals to the Arduino so that line following or obstacle avoidance functionality could be possible. On the left is one of the motor drivers connected into the Arduino Uno, which drives two DC motors. These are what give motion to the robot as forward, backwards, and turning. The motor driver interprets control signals from the Arduino and adjusts accordingly the speed and direction of the motors. Basically, the block diagram shows a system integrated with IR and ultrasonic sensor inputs, processed by the Arduino Uno controlling a motor driver for the control of the robot car. In clear terms, this setup will thus make the robot car trace its way around the environment in an automatic way, following lines or avoiding obstacles with instructions pre-programmed on the Arduino.

- State Machine Diagram The state machine diagram (9) shows the runtime logic of an Arduino-controlled lane-identifying vehicle with integrated color detection capabilities. The system runs in three distinct states, ”Waiting”, ”Lane Follow”, and ”Collision Detected”. When in the ”Waiting” state, the vehicle stays until both left and right sensors detect the lane markers, after which transition to the ”Lane Follow” state occurs.

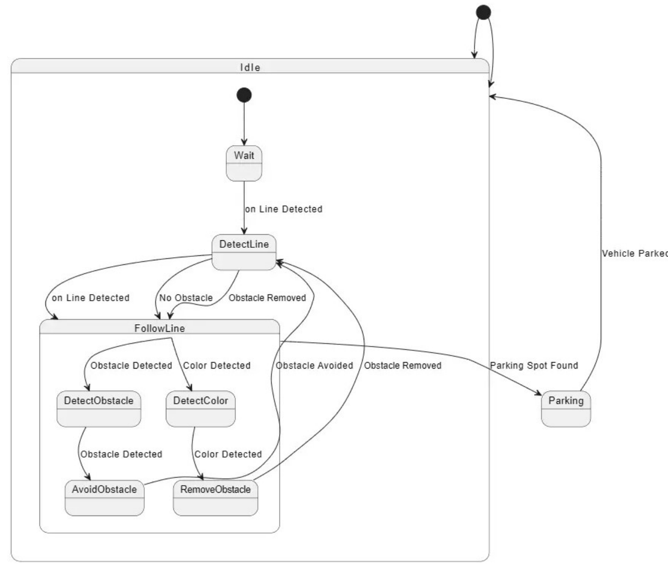


Fig. 9. State Machine Diagram

At this stage, the vehicle follows the lane with motor control based on continuous infrared sensor feedback. A transition back to ”Waiting” will be made when either sensor loses track of the lane resulting in bringing the car back towards the lane. The state Collision Detected is activated in case of detecting an obstacle that is too near, distance is less than 20, while seeing a red color simultaneously, ColorSensor() == 0, this is defined in he functions in sections of Arduino code In this state, the vehicle performs avoidance actions, for example, turning left or right, then goes back to its route once the obstacle no longer poses difficulties for the vehicle’s movements. It is used in this way for behavior in a structured context to keep the vehicle running autonomously, following the line, due to dynamic reactions to environmental conditions, thus promoting its efficiency and safety within specific practical applications.

- Activity Diagram
  The activity diagram (10) illustrates the decision-making process involving the vehicle when it is on the track. The process initiates right from when the IR sensors start sensing that the black line is present or is being chased. When both Sensors are in contact with the line, the car moves forward. Therefore, in case only the left IR sensor has detected this line, the vehicle moves to the left. On the other hand, if only the right IR sensor detects the line then it turns right.
  In any case of failure to detect both the IR sensors help the ultrasonic sensors to detect the obstacles. When an obstacle is identified, the vehicle steers by 45 degrees towards left avoiding the obstacle. The vehicle moves forward when there are no obstacles found in the path of the vehicle or the path is clear.

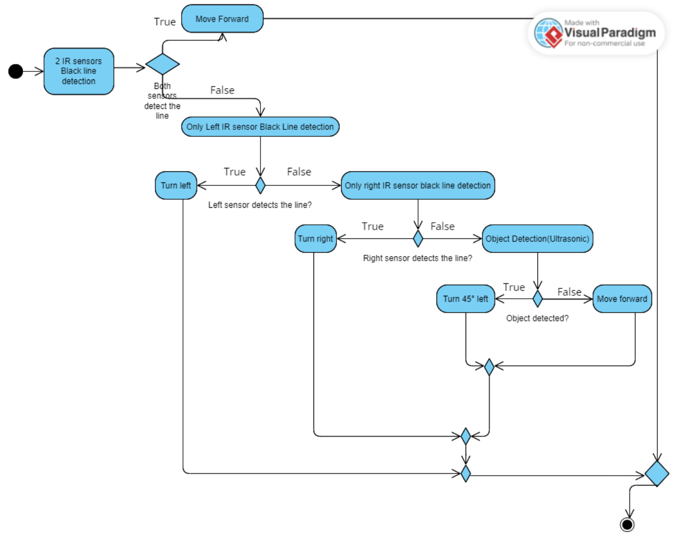


Fig. 10. Activity Diagram

It enables the car to stay on course; this decision-making loop can plot its own path around an object in its path with help from real-time input from sensors. They include its ability for track and obstacle detection on the track that will make the required autonomous movement safely.
It is the diagram of vision for the logical sequences of sensors and the control mechanism of the vehicle to make it an autonomous car.

- Requirement Diagram

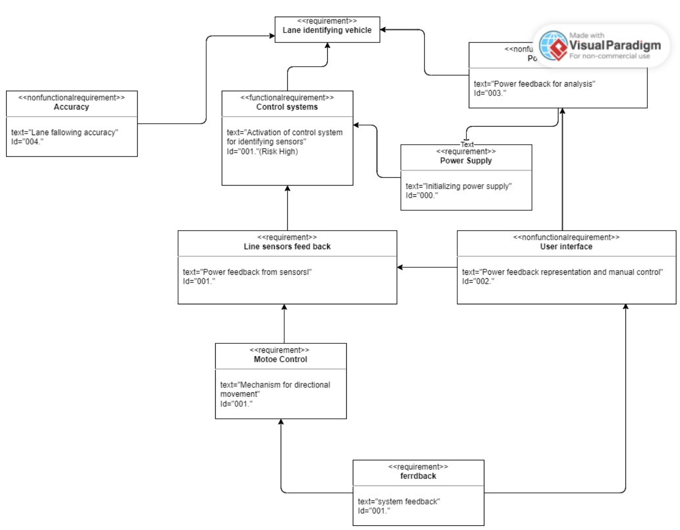


Fig. 11. Requirement Diagram

The requirement diagram (11) indicates detailed requirements with respect to a vehicular autonomous vehicle system that identifies lanes. It has a primary requirement at the center labeled as Lane identifying vehicle( autonomous vehicle) anchoring many functional and non-functional features. Relating to one of the categories, it incorporates the functionality of the control systems' requirements, featuring the activation of control systems for identifying sensors with a risk level noted to be high. Other critically important functional requirements will include the power supply, involving the initialization of the vehicle's power supply. The line sensors will prove critical in providing feedback since it implies giving the critical environmental data to the system. Another will be the motor control, defining mechanisms for movement of the vehicle in the directional aspects so as to ensure a following of lanes with precision. All the way to the end, there will be a continuous feedback of systems to upgrade the status of the vehicle to ensure accuracy in its operation. On the other hand, the nonfunctional requirements would be concerned with the performance of the system and how the user will relate to it. The accuracy required, especially lane following, is that the vehicle hold its path correctly. One critical requirement is that of power efficiency and requires power feedback for analysis to address how best the system will suitably use energy. Finally, there is a user interface requirement intended to represent feedback and allow manual control. This shall ensure that there is good presentation of the outputs from this system to the users, as well as allowing manual override if need be. The detailed breakdown of the requirements followed clearly reiterates the complexity and retroactions between different system components required to be in place for a dependable lane-identifying autonomous vehicle to become realized.

## IV. Arduino Code

The complete code can be found in A. **Motor Driver**

The motor driver facilitates precise control over the movement of the car. It utilizes digital output pins (in1Pin, in2Pin, in3Pin, in4Pin) for controlling the direction of rotation and PWM pins (enA, enB) for adjusting the speed of motors A and B.

```c
// Define pins for motor driver
const int in1Pin = 11;
const int in2Pin = 10;
const int in3Pin = 13;
const int in4Pin = 12;
const int enA = 5;
const int enB = 6;
```

Listing 1. C Language

The code explains the following:

- in2Pin (Pin 10): Assigned to the second input signal of the motor driver.
- in3Pin (Pin 13): Assigned to the third input signal of the motor driver.
- in4Pin (Pin 12): Assigned to the fourth input signal of the motor driver.
- enA (Pin 5): Assigned to allow for motor A and control its speed via a PWM signal.

- enB (Pin 6): Assigned to allow for motor B and control its speed via a PWM Signal

**Ultrasonic Sensor** An ultrasonic sensor is employed for distance measurement and obstacle detection. It consists of a trigger pin (trigPin) to emit ultrasonic signals and an echo pin (echoPin) to receive and calculate distances based on signal reflections.

```c
// Define pins for ultrasonic sensor
const int trigPin = 2;
const int echoPin = 3;
```

Listing 2. C Language

The ultrasonic sensor's trigger pin is connected to the micro-controller's digital pin 2 and the Echo pin is connected to the digital pin 3

```c
void measureDistance() {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance = duration * 0.034 / 2;
}
```

Listing 3. C Language

Here in the code,

- digitalWrite(trigPin, LOW);: Ensures the trigger pin is set to LOW.
- delayMicroseconds(2);: Waits for 2 microseconds.
- digitalWrite(trigPin, HIGH);: Sets the trigger pin to HIGH.
- delayMicroseconds(10);: Waits for 10 microseconds.
- digitalWrite(trigPin, LOW);: Sets the trigger pin back to LOW.
- duration = pulseIn(echoPin, HIGH);: Measures the pulse duration on the echo pin.
- distance = duration * 0.034 / 2;: Calculates the distance using the speed of sound.

**IR Sensors**

Infrared (IR) sensors are utilized for line detection and navigation. Digital input pins (leftsensorPin, rightsensorPin) detect variations in reflected IR light, enabling the car to follow predefined paths.

```c
// Define pins for IR sensors
const int leftsensorPin = 8;
const int rightsensorPin = 7;
```

Listing 4. C Language

**Color Sensor**

A color sensor enhances the car's functionality by detecting and identifying colors of encountered objects. Control pins (s0, s1, s2, s3) configure the sensor's operational mode, while an output pin (out) provides RGB values for color analysis

```c
#define s0 A0       // Analog pins for color sensor
    control
#define s1 A1
#define s2 A2
#define s3 A3
#define out A4
```

Listing 5. C Language

**Functions and Operation**
**Motor Control Functions**

These functions regulate the movement of the car, including forward motion, turning, and stopping.

```c
void forward() {
    digitalWrite(in1Pin, LOW);
    digitalWrite(in2Pin, HIGH);
    digitalWrite(in3Pin, LOW);
    digitalWrite(in4Pin, HIGH);
}
```

Listing 6. C Language

The forward function is used to set the motor driver to move the motors in a forward direction. Where,

- digitalWrite(in1Pin, LOW);: Sets the first control pin of the first motor to
- LOW, ensuring that one side of the motor is turned off.
- digitalWrite(in2Pin, HIGH);: Sets the second control pin of the first
- motor to HIGH, ensureing that the other side of the motor rotate in the forward direction.
- digitalWrite(in3Pin, LOW);: Sets the first control pin of the second motor to LOW, turning off the other side of the motor-
- digitalWrite(in4Pin, HIGH);: Sets the second control pin of the second
- motor to HIGH, allowing the other motor to rotate forward.

```c
void left() {
    analogWrite(enA, motorSpeed);
    analogWrite(enB, motorSpeed);
    digitalWrite(in1Pin, LOW);
    digitalWrite(in2Pin, HIGH);
    digitalWrite(in3Pin, HIGH);
    digitalWrite(in4Pin, LOW);
}
```

Listing 7. C Language

Here the code explains,

- analogWrite(enA, motorSpeed);: Sets the speed of motor A by sending a PWM signal to enA based on the motorSpeed.
- analogWrite(enB, motorSpeed);: Sets the speed of motor B by sending a PWM signal to enB based on the motorSpeed.
- digitalWrite(in1Pin, LOW);: Sets the first control pin of the first motor to LOW.
- digitalWrite(in2Pin, HIGH);: Sets the second control pin of the first motor to HIGH.
- digitalWrite(in3Pin, HIGH);: Sets the first control pin of the second motor to HIGH.
- digitalWrite(in4Pin, LOW);: Sets the second control pin of the second motor to LOW.

```c

void right() {
```

```c
    analogWrite(enA, motorSpeed);
    analogWrite(enB, motorSpeed);
    digitalWrite(in1Pin, HIGH);
    digitalWrite(in2Pin, LOW);
    digitalWrite(in3Pin, LOW);
    digitalWrite(in4Pin, HIGH);
}
```

Listing 8. C Language

The Right function also does the opposite of the Left to help the car move on the right side

```c

void turn() {
    analogWrite(enA, 0);
    analogWrite(enB, 100);
    digitalWrite(in1Pin, LOW);
    digitalWrite(in2Pin, HIGH);
    digitalWrite(in3Pin, HIGH);
    digitalWrite(in4Pin, LOW);
}
```

Listing 9. C Language

The function is used when the car has come back to the line and the line is not detected by the IR sensors. It causes the car to rotate on its axis until the IR sensors detect the line again.

It performs the following steps:

- Sets the speed of motor A to 0 using the analogWrite function on enA, effectively stopping motor A.
- Sets the speed of motor B to 100 using the analogWrite function on enB.
- Sets in1Pin to LOW to control the direction of the first motor.
- Sets in2Pin to HIGH to control the direction of the first motor.
- Sets in3Pin to HIGH to control the direction of the second motor.
- Sets in4Pin to LOW to control the direction of the second motor

**Color Sensor Functions**

These functions interact with the color sensor to capture RGB values and analyze colors of encountered objects.

```c
void GetColors() {
    digitalWrite(s2, LOW);
    digitalWrite(s3, LOW);
    Red = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
    delay(20);
    digitalWrite(s3, HIGH);
    Blue = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
    delay(20);
    digitalWrite(s2, HIGH);
    Green = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
    delay(20);
}

int ColorSensor(){
   if (Red <= 15 && Green <= 15 && Blue <= 15) {   // If the values are low, it's likely the white color (all the colors are present)
    Serial.println("White");

  } else if (Red < Blue && Red <= Green && Red < 23) {  // If Red value is the lowest one and smaller than 23, it's likely Red
    Serial.println("Red");
    return 0;
  } else if (Blue < Green && Blue < Red && Blue < 20) {  // Same thing for Blue
    Serial.println("Blue");
    return 1;
  } else {
    Serial.println("Unknown");  // If the color is not recognized, you can add more conditions as needed
  }

  delay(1000);  // 2s delay, you can modify if you want
}
```

Listing 10. C Language

The GetColors() function configures the sensor to read different colors and uses the pulseIn function to measure the duration of the pulse width for each color channel.

s2 and s3 are control pins for the color sensor that select which color filter to use. The out pin is the sensor's output that gives a pulse width corresponding to the intensity of the color. The pulseIn function reads the duration of the pulse (in microseconds) on the out pin. The pulse duration is inversely proportional to the intensity of the color. A small delay is added after reading each color to ensure the sensor stabilizes before the next reading. The ColorSensor function compares the pulse widths of Red, Green, and Blue. It returns an integer corresponding to the detected color: 1 for Red unknown for Green 0 for Blue Unknown (when the color cannot be clearly determined)

**Setup and Initialization**

Initialising Pins

```c
void setup() {
    // Initialize serial communication for debugging
    Serial.begin(9600);

    // Initialize motor driver pins
    pinMode(in1Pin, OUTPUT);
    pinMode(in2Pin, OUTPUT);
    pinMode(in3Pin, OUTPUT);
    pinMode(in4Pin, OUTPUT);
    pinMode(enA, OUTPUT);
    pinMode(enB, OUTPUT);

    // Initialize ultrasonic sensor pins
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);

    // Initialize IR sensor pins
    pinMode(leftsensorPin, INPUT);
    pinMode(rightsensorPin, INPUT);

    // Initialize color sensor pins
    pinMode(s0, OUTPUT);
    pinMode(s1, OUTPUT);
    pinMode(s2, OUTPUT);
    pinMode(s3, OUTPUT);
    pinMode(out, INPUT);

    // Set initial motor speed
    analogWrite(enA, motorSpeed);
```

```
   analogWrite(enB, motorSpeed);
}
```

Listing 11. C Language

The setup function is called once during the program's initialization phase. It is in charge of configuring the pin modes for the motor driver, ultrasonic sensor, and infrared sensors. By selecting the right pin modes (input or output), the controller guarantees that signals are appropriately received or delivered. In addition, the setup function uses the analogWrite function to specify the initial speeds for the car's motors.

These initial speed settings can be changed to meet the car's individual requirements.

Motor Driver Pins Initialization:

- pinMode(in1Pin, OUTPUT);: Sets in1Pin as an output pin.
- pinMode(in2Pin, OUTPUT);: Sets in2Pin as an output pin.
- pinMode(in3Pin, OUTPUT);: Sets in3Pin as an output pin.
- pinMode(in4Pin, OUTPUT);: Sets in4Pin as an output pin.
- pinMode(enA, OUTPUT);: Sets enA as an output pin.
- pinMode(enB, OUTPUT);: Sets enB as an output pin. Ultrasonic Sensor Pins Initialization:
- pinMode(trigPin, OUTPUT);: Sets trigPin as an output pin for the ultrasonic
- sensor.
- pinMode(echoPin, INPUT);: Sets echoPin as an input pin for the ultrasonic sensor.

IR Sensor Pins Initialization:

- pinMode(leftSensorPin, INPUT);: Sets leftSensorPin as an input pin for the left IR sensor.
- pinMode(rightSensorPin, INPUT);: Sets rightSensorPin as an input pin for the right IR sensor.

**Main Control Loop**

A color sensor enhances the car's functionality by detecting and identifying colors of encountered objects. Control pins (s0, s1, s2, s3) configure the sensor's operational mode, while an output pin (out) provides RGB values for color analysis

```
 // Read IR sensor input
 int leftsensorValue = digitalRead(leftsensorPin);
 int rightsensorValue = digitalRead(rightsensorPin);
analogWrite(enA, motorSpeed);
 analogWrite(enB,motorSpeed );
 if (leftsensorValue == 1 && rightsensorValue == 1) {
 forward();
 } else if (leftsensorValue == 0 && rightsensorValue == 1) {
 right();
 } else if (leftsensorValue == 1 && rightsensorValue == 0) {
 left();

 }
 //color sensor
   GetColors();  // Execute the GetColors function to get the value of each RGB color

 // Ultrasonic sensor code

 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(7);
 digitalWrite(trigPin, LOW);
 duration = pulseIn(echoPin, HIGH);
 distance = duration/34.2;
 if(distance==0){
 distance=100;
 }


 if(distance < 20 && ColorSensor() == 0 )
 {
      Serial.println(distance);

 left();
 delay(900);
 forward();
 delay(1400);
 right();
 delay(1200);
 forwardU();
 leftsensorValue = digitalRead(leftsensorPin);
 rightsensorValue = digitalRead(rightsensorPin);

while(leftsensorValue == 1 && rightsensorValue == 1 ){
 leftsensorValue = digitalRead(leftsensorPin);
rightsensorValue = digitalRead(rightsensorPin);
}
 stop();
 delay(2000);
 forwardU();
 delay(100);
 rightsensorValue = digitalRead(rightsensorPin);
 turn();
 while(rightsensorValue == 1){
 rightsensorValue = digitalRead(rightsensorPin);
}

}

delay(500);
}
```

Listing 12. C Language

This code begins by reading two infrared (IR) sensors by means of the function digitalRead. Besides reading HIGH (1) or LOW (0), the reading was then stored for the left and right sensor in leftsensorValue and rightsensorValue, respectively. The speed of the motors was set with the analogWrite function and was controlled by the motorSpeed variable that fed enA and enB pins with a proper PWM signal for the motors.

Control logic for the motors, based on IR sensor inputs, is to move forward only if the line is detected by both sensors: rightsensorValue == 1  leftsensorValue == 1; hence, move forward. Else, it should turn to the right if only the right sensing element indicates that it has found a line: if rightsensorValue == 1 leftsensorValue == 0, then turnright. Conversely, if a line is detected on only the left sensor, that means leftsensorValue is 1, while the rightsensorValue is 0. Then, the robot turns to the left by invoking the left() function.

Finally, the current RGB color values are read from a color sensor using a GetColors() function Call to get the current RGB color values from the Color Sensor done later to be used to implement the line following logic. After this, distance measurement to any obstacle was done using the ultrasonic sensor. A low level is given out on trigger pin for a short period and then brought high in order to transmit an ultrasonic pulse. Then the pulseIn() function is used to determine the echo received in the echo pin's duration and the duration divided by 34.2. This is achieved through a division of the speed of sound and two. If not detecting any object, distance == 0, then it has to set the distance to 100 cm.

If the measured distance is less than 20 cms and the color detected is red, ColorSensor() == 0, then the robot will start to give a number of evasive maneuvers against the obstacle; it will just print the distance, turn left for 900 ms, move forward for 1400 ms, turn right for 1200 ms, and keep moving. In this sequence, the IR sensor values are checked again so as to make sure this way is free. It then stops and waits for 2 s, moves forward again for 100 ms, rechecks the right sensor, and does a turn until the right sensor does not give a TRUE value for the line detection. After these actions, the robot has a pause of 500 ms and the loop is repeated. This logic ensures that the robot can navigate by following a path and being warned of obstacles based on sensor inputs and programmed behavior.

## V. Designing

### A. Laser cutting

For this prototype lase cutting technique on plywood was to used to create the chassis 12 of the vehicle. Two similar sheets were cut with same dimension, first being the chassis with 1 central hole to place the ball bearing as we were using front wheel drive to maneuver the vehicle. Similarly screw points were later created by hand drill to create mounted hole for brackets

The chassis is with the dimensions as followed.

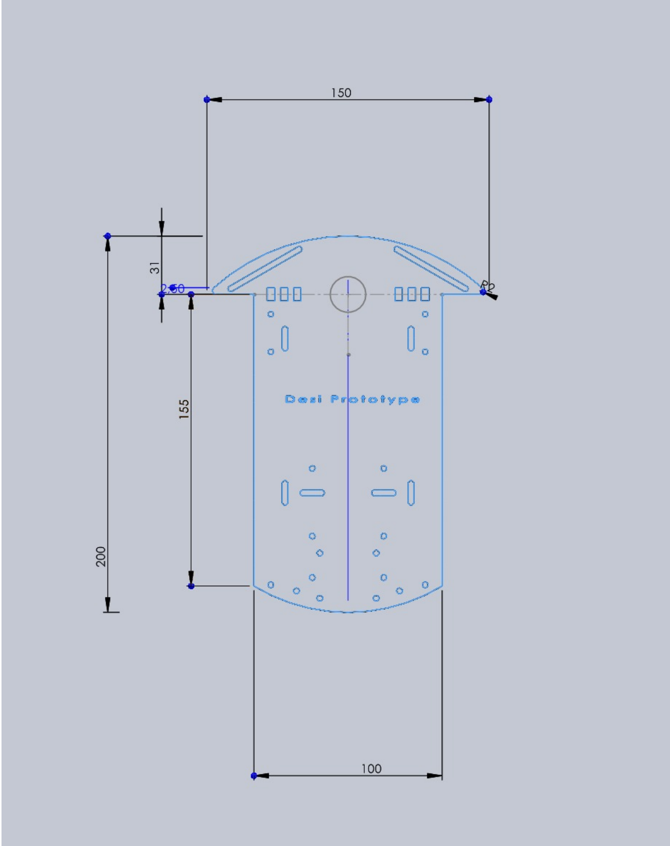


Fig. 12. Chassis

Fig. 13. UltraSonic mount

### B. Ultra sonic Mount

An ultra sonic mount (Fig. 13) was 3D printed and designed on solid works with .stl file format. This prototype uses this mount to stabilise the ultrasonic sensor to receive a more precise reading. This mount also enable us to join a servo motor with the ultrasonic sensor so the ultrasonic sensor can be used with 180 degrees rotation so it can detect objects left and the right side of the car.

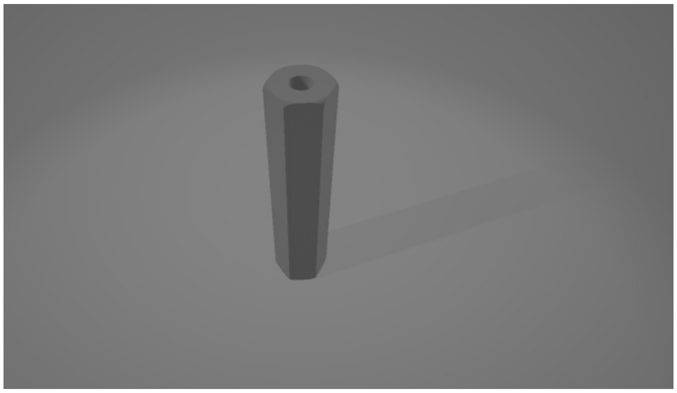

Fig. 14. Coupling Nut

### *C. Couping nut*

A coupling nut was also designed and 3D with the same techniques as the ultrasound mount. The coupling nut was used to stabilize and connect both the upper plywood sheet and the chassis together for stable and precise drive, This technique also helped lower the vehicles center of gravity which provides higher stability allowing the vehicle to go faster. [1]

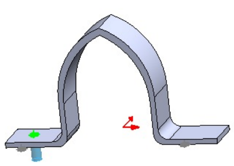

Fig. 15. Motor Clamp

### *D. Motor mounting Clamp*

This figure shows a 3D motor mounting clamp that is used to mount DC motors to the chassis of the car for more stable connection and torque production for the drive of the car.

### *E. Complete CAD model*

The Following figures show the complete CAD Model of the main prototype. The color sensor module has been sourced from online library. [1]

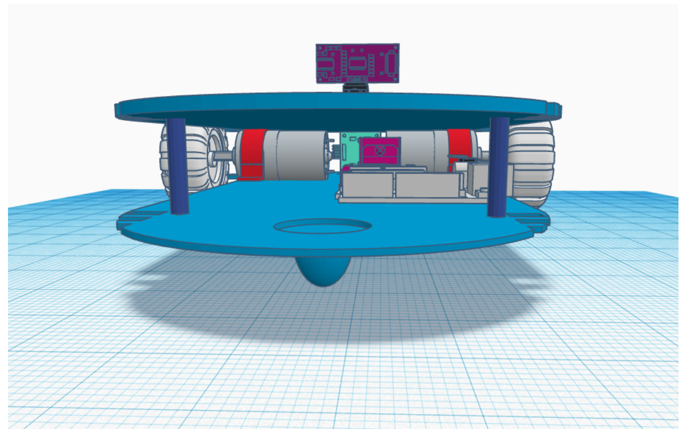

Fig. 16. Back View

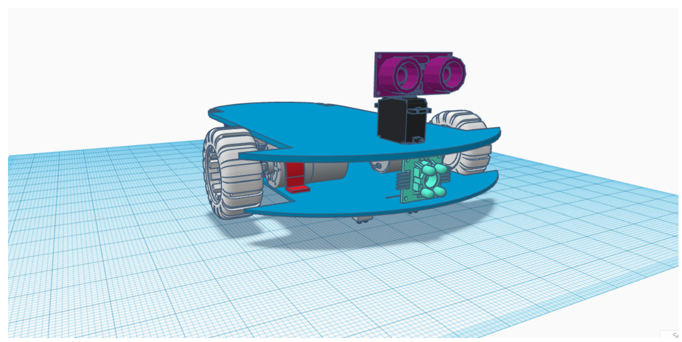

Fig. 17. Left Side View

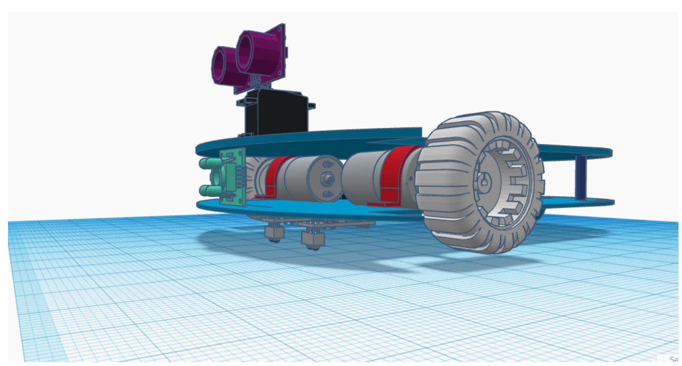

Fig. 18. Right side view

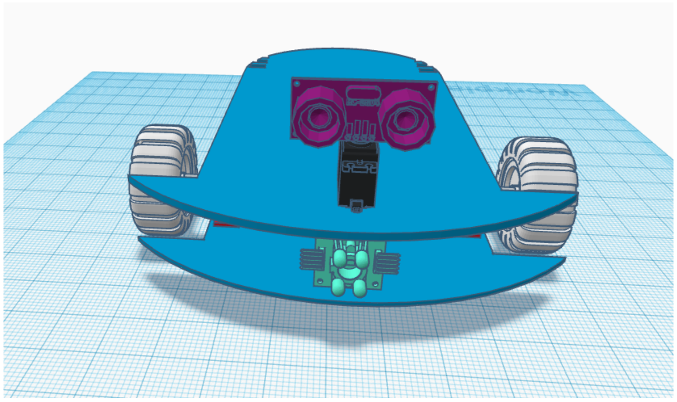

Fig. 19. Top Front VIew

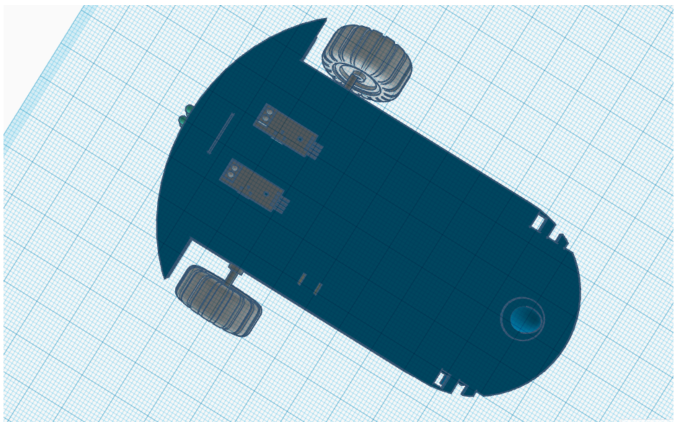

Fig. 20. Under View

## VI. Future implementations and Conclusion

With the extreme challenges of this world humans require new ideas and admiration to conquer the new rising problems. This paper also provides a solution for a futuristic problem.
The Arduino code for websocket handshake, with open-source web-socket handshake library [websocket], provides a robust framework for creating versatile robotic systems that help in a wide scope of applications in automation, IoT, and robotics. This idea integrates capabilities of the line following prototype's autonomous navigation and web sockets together. It identifies colors with a very high degree of accuracy using an RGB sensor, and thus can be used in object sorting and quality control in various manufacturing-related industries. It also provides support for WebSocket communication that makes real-time interaction and control over the internet possible for remote monitoring and operation of the robotic system. This code has not only practical applications in industry—in manufacturing and smart automation at home—but also turns out to be useful for teaching students and hobbyists programming, integration of sensors, and design of robots. This would be very conducive to research and development in furthering robotics technology, exploring sensor technology innovations, and autonomous navigation algorithms. At large, this code means having versatile toolsets ready to drive progress into automation and robotics across several domains; one can use this technology for industrial automation projects or educational robotics projects.
You can find a trial code fully integrated with the main code of the prototype presented in this paper in the **appendix** (A) section in the end of this paper. With a webpage, Using HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) and Javascript, representing user interface for inputs from the user and outputs for the user.

GitHub Repository [9]

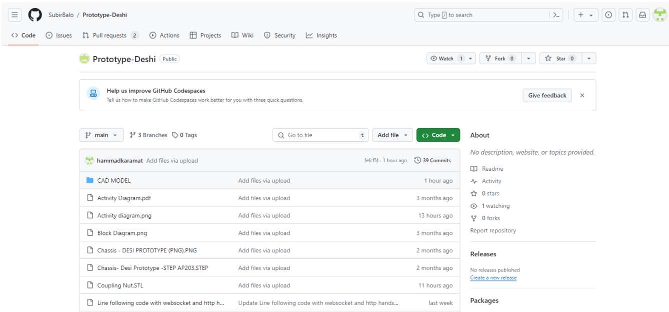


Fig. 21. https://github.com/SubirBalo/Prototype-Deshi

### A. *Finalised Prototype*

The following figures show the finalised prototype that compliments this paper.

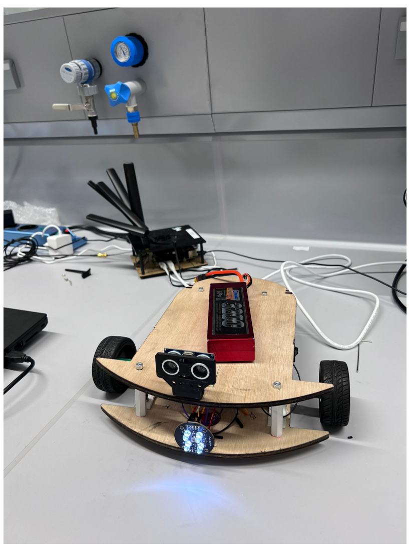

Fig. 22. Top View

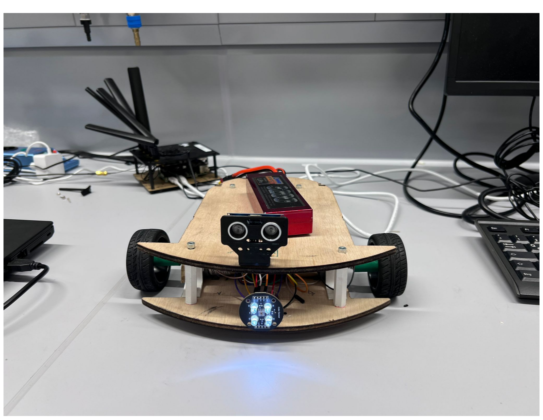

Fig. 23. Front View

## VII. Authors and Affiliations

As the proud authors of prototyping paper we present our ideas and hard work as combined force with equal and dedicated intentions. The hard work of all authors must be equally taken into consideration. Team record of this project is equal division of work to all participating members. SysMl, Code, designing and all other work tasks have been equally fulfilled by all members.

## Acknowledgment

We would like to express deepest gratitude to Professor Henkler and Professor Wahrmann for valuable guidance, imperative feedback, and continuous support in regard to this research. Their expertise, coupled with encouragement, was vital in the shaping of this work.

We would also like to extend my gratitude to Hochschule Hamm-Lippstadt for providing resources and facilities—without which this research could not have been completed. The academic atmosphere and teamwork opportunities have deeply enriched my experience in research.

This research could not have been possible if not for the support and contribution from these individuals and institutions. Their effort and help were indeed invaluable.

## VIII. AFFIDAVIT

**I hereby certify that I have written this paper independently without the help of third parties and without using any sources or aids other than those indicated. I have indicated all passages in the paper that are taken from printed works or sources from the Internet, either in wording or in meaning, by citing the sources. This also applies to all illustrations. The submitted work has not been the subject of any other examination procedure, neither in its entirety nor in essential parts. I am aware that plagiarism is serious academic misconduct that will be reported to the examination board and will result in sanctions. Furthermore, I assure that the electronic version of the paperr corresponds to the printed version.**

**Hammad Karamat**
**01.07.2024**

**I hereby certify that I have written this paper independently without the help of third parties and without using any sources or aids other than those indicated. I have indicated all passages in the paper that are taken from printed works or sources from the Internet, either in wording or in meaning, by citing the sources. This also applies to all illustrations. The submitted work has not been the subject of any other examination procedure, neither in its entirety nor in essential parts. I am aware that plagiarism is serious academic misconduct that will be reported to the examination board and will result in sanctions. Furthermore, I assure that the electronic version of the paperr corresponds to the printed version.**

**Ismail Hossain**
**01.07.2024**

**I hereby certify that I have written this paper independently without the help of third parties and without using any sources or aids other than those indicated. I have indicated all passages in the paper that are taken from printed works or sources from the Internet, either in wording or in meaning, by citing the sources. This also applies to all illustrations. The submitted work has not been the subject of any other examination procedure, neither in its entirety nor in essential parts. I am aware that plagiarism is serious academic misconduct that will be reported to the examination board and will result in sanctions. Furthermore, I assure that the electronic version of the paperr corresponds to the printed version.**

**Abu Sayem**
**01.07.2024**

**I hereby certify that I have written this paper independently without the help of third parties and without using any sources or aids other than those indicated. I have indicated all passages in the paper that are taken from printed works or sources from the Internet, either in wording or in meaning, by citing the sources. This also applies to all illustrations. The submitted work has not been the subject of any other examination procedure, neither in its entirety nor in essential parts. I am aware that plagiarism is serious academic misconduct that will be reported to the examination board and will result in sanctions. Furthermore, I assure that the electronic version of the paperr corresponds to the printed version.**

**Subir Balo**
**01.07.2024**

# REFERENCES


[1] https://www.thingiverse.com/thing:6287118 *TCS 3200 Color Sensor Module* by LukiDesign is licensed under the Creative Commons - Attribution - Share Alike license.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] https://circuit.rocks/products/arduino-uno-wifi-esp8266

[4] https://quartzcomponents.com/products/ir-line-follower-sensor-module

[5] https://www.antratek.de/hc-sr04-ultrasonic-sonar-distance-sensor

[6] https://forum.arduino.cc/t/motor-driver-l298n-motors-dont-get-voltage/952073

[7] https://www.botnroll.com/en/dc-motor/2799-rb-35-1-30-gearbox-motor.html

[8] https://www.delongbattery.com/3-7V-5000mAh-Lipo-Battery-Lithium-Polymer-Battery-Cell-5560100-pd49032705.html

[9] https://github.com/SubirBalo/Prototype-Deshi

[10] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[11] K. Elissa, “Title of paper if known,” unpublished.

[12] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[13] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[14] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[15] https://joy-it.net/en/products/SEN-Color

[websocket] Author Markus Sattler Website https://github.com/Links2004/arduinoWebSockets Category Communication License LGPL 2.1 Library Type Contributed Architectures Any use 2.x.x for ESP and 1.3 for AVR


# APPENDIX

This appendix shows all sorts of code workings.

### Line following with Websocket integration

```
/* WebSocket_WiFiRev2
 * Deriviative work from several of the builtin examples.
 * Markus Sattler's websockets library takes up 150% of the memory.
 */

String GUID = "258EAFA5-E914-47DA-95CA-C5AB0DC85B11";

#include <SPI.h>
#include <WiFiNINA.h>
#include <Arduino_LSM6DS3.h>
#include "cencode_inc.h"
#include "libsha1.h"
#include "webpage.h"

#include "arduino_secrets.h"
///////please enter your sensitive data in the Secret tab/arduino_secrets.h
char ssid[] = "Hammad";        // your network SSID (name)
char pass[] = "paklhr123";    // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0;                // your network key index number (needed only for WEP)

int led =  LED_BUILTIN;
int status = WL_IDLE_STATUS;
WiFiServer server(80);
WiFiServer wsserver(8080);
WiFiClient wsclient;

/**
 * base64_encode
 * @param data uint8_t *
 * @param length size_t
 * @return base64 encoded String
 */
String base64_encode(uint8_t * data, size_t length) {
    size_t size   = ((length * 1.6f) + 1);
    char * buffer = (char *)malloc(size);
    if(buffer) {
        base64_encodestate _state;
        base64_init_encodestate(&_state);
        int len = base64_encode_block((const char *) &data[0], length, &buffer[0], &_state);
        len     = base64_encode_blockend((buffer + len), &_state);

        String base64 = String(buffer);
        free(buffer);
        return base64;
    }
    return String("-FAIL-");
}

void setup() {
  //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  Serial.println("Access Point Web Server");

```

```
  pinMode(led, OUTPUT);      // set the LED pin mode

  // check for the WiFi module:
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    // don't continue
    while (true);
  }

  String fv = WiFi.firmwareVersion();
  if (fv < WIFI_FIRMWARE_LATEST_VERSION) {
    Serial.println("Please upgrade the firmware");
  }

  // by default the local IP address will be 192.168.4.1
  // you can override it with the following:
   WiFi.config(IPAddress(10, 0, 0, 1));

  // print the network name (SSID);
  Serial.print("Creating access point named: ");
  Serial.println(ssid);

  // Create open network. Change this line if you want to create an WEP network:
  #if IS_ACCESS_POINT
  status = WiFi.beginAP(ssid, pass);
  if (status != WL_AP_LISTENING) {
    Serial.println("Creating access point failed");
    // don't continue
    while (true);
  }
  #else
  status = WiFi.begin(ssid, pass);
  while(status != WL_CONNECTED) {
      delay(100);
      status = WiFi.begin(ssid, pass);
  }
  #endif


  // wait 3 seconds for connection:
  delay(3000);

  IMU.begin();

  // start the web server on port 80
  server.begin();
  wsserver.begin();
  wsclient.stop();

  // you're connected now, so print out the status
  printWiFiStatus();

  WiFiDrv::pinMode(25, OUTPUT);
  WiFiDrv::pinMode(26, OUTPUT);
  WiFiDrv::pinMode(27, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
}

void check_wifi_status()
{
   // compare the previous status to the current status
  if (status != WiFi.status()) {
    // it has changed update the variable
    status = WiFi.status();

    if (status == WL_AP_CONNECTED) {
      // a device has connected to the AP
      Serial.println("Device connected to AP");
    } else {
      // a device has disconnected from the AP, and we are back in listening mode
      Serial.println("Device disconnected from AP");
    }
  }
}

void check_web_request()
{
  WiFiClient client = server.available();   // listen for incoming clients

  if (client) {                             // if you get a client,
    Serial.println("new client");           // print a message out the serial port
    String request = "";                    // make a String to hold incoming data from the client
    if (client.connected()) {               // loop while the client's connected
      while (client.available()) {          // if there's bytes to read from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        request += c;                       // add it to current
      }
    }
    if (request.startsWith("GET / HTTP/1.1")) {
        // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
        // and a content-type so the client knows what's coming, then a blank line:
        client.println("HTTP/1.1 200 OK");
        client.println("Content-type:text/html");
        client.println();
        //client.println("<html><head><script>var connection = new WebSocket('ws://'+location.hostname+':8080/');connection.onopen = function () {  connection.send('Connect ' + new Date()); }; connection.onerror = function (error) { console.log('WebSocket Error ', error);}; connection.onmessage = function (e) {  console.log('Server: ', e.data);};function sendRGB() { var r = parseInt(document.getElementById('r').value).toString(16);  var g = parseInt(document.getElementById('g').value).toString(16);  var b = parseInt(document.getElementById('b').value).toString(16);  if(r.length < 2) { r = '0' + r; }   if(g.length < 2) { g = '0' + g; }   if(b.length < 2) { b = '0' + b; }   var rgb = '#'+r+g+b;    console.log('RGB: ' + rgb); connection.send(rgb); }</script></head><body>LED Control:<br/><br/>R: <input id=\"r\" type=\"range\" min=\"0\" max=\"255\" step=\"1\" oninput=\"sendRGB();\" /><br/>G: <input id=\"g\" type=\"range\" min=\"0\" max=\"255\" step=\"1\" oninput=\"sendRGB();\" /><br/>B: <input id=\"b\" type=\"range\" min=\"0\" max=\"255\" step=\"1\" oninput=\"sendRGB();\" /><br/></body></html>");
        client.println(webpage);
        // The HTTP response ends with another blank line:
        client.println();
    } else {
        client.println("HTTP/1.1 404 Not Found");
        client.println();
    }
    // close the connection:
    client.stop();
    Serial.println("client disconnected");
  }
}
```

```

void handshake()
{
  size_t matchpos = 0;
  bool nonce_active = false;
  String nonce = "";
  String Sec_WebSocket_Key = "Sec-WebSocket-Key: ";
  Serial.println("new WS client");          // print a message out the serial port
  while (wsclient.available()) {
    char c = wsclient.read();
    if (nonce_active) {
      if (c != '\r' && c != '\n') {
        nonce += c;
      } else {
        nonce_active = false;
      }
    }
    if (c == Sec_WebSocket_Key[matchpos]) {
      matchpos++;
      if (matchpos == Sec_WebSocket_Key.length()) {
        nonce_active = true;
      }
    } else {
      matchpos = 0;
    }
  }

  if (nonce.length() > 0) {
    uint8_t sha1HashBin[20] = { 0 };
    String clientKey = nonce;
    clientKey += GUID;

    SHA1_CTX ctx;
    SHA1Init(&ctx);
    SHA1Update(&ctx, (const unsigned char*)clientKey.c_str(), clientKey.length());
    SHA1Final(&sha1HashBin[0], &ctx);

    String key = base64_encode(sha1HashBin, 20);
    key.trim();

    Serial.print("Nonce: \"");
    Serial.print(nonce);
    Serial.print("\" -> \"");
    Serial.print(key);
    Serial.println("\"");

    wsclient.print("HTTP/1.1 101 Web Socket Protocol Handshake\r\n");
    wsclient.print("Upgrade: websocket\r\n");
    wsclient.print("Connection: Upgrade\r\n");
    wsclient.print("Sec-WebSocket-Accept: ");
    wsclient.print(key);
    wsclient.print("\r\n\r\n");
  }
}

enum {
  WS_FIN         = 0x80,
  WS_FIN_SHIFT   = 0x07,
  WS_FR_OP_TXT   = 0x01,
  WS_FR_OP_BIN   = 0x02,
  WS_FR_OP_CLOSE = 0x08,
  WS_FR_OP_PING  = 0x09, // 1001
  WS_FR_OP_PONG  = 0x0A, // 1010
  WS_FR_OP_UNSUPPORTED = 0x0F,
};

uint8_t fromhex(char c)
{
  if ('0' <= c && c <= '9') { return c - '0'; }
  if ('A' <= c && c <= 'F') { return c - 'A' + 10; }
  if ('a' <= c && c <= 'f') { return c - 'a' + 10; }
  return 0;
}

uint8_t R;
uint8_t G;
uint8_t B;
uint8_t LED_value;
float ax = 0.0f;
float ay = 0.0f;
float az = 0.0f;
float gx = 0.0f;
float gy = 0.0f;
float gz = 0.0f;
float temp = 0.0f;
float AX, AY, AZ, GX, GY, GZ, TEMP;

void ws_send()
{
  char c;
  c = WS_FIN | WS_FR_OP_BIN;
  wsclient.write(c);
  c = 32;
  wsclient.write(c);
  wsclient.write(R);
  wsclient.write(G);
  wsclient.write(B);
  wsclient.write(LED_value);
  wsclient.write((const char*)&AX, 4);
  wsclient.write((const char*)&AY, 4);
  wsclient.write((const char*)&AZ, 4);
  wsclient.write((const char*)&GX, 4);
  wsclient.write((const char*)&GY, 4);
  wsclient.write((const char*)&GZ, 4);
  wsclient.write((const char*)&TEMP, 4);
}

void check_ws_request()
{
  if (wsclient.connected()) {

    int total = 0;
    while (wsclient.available()) {
      char c = wsclient.read();
      total++;
      char n;
      char op = c & 0x0F;
      Serial.print(c, HEX);
      Serial.print(" ");
      switch(op)
      {
        case WS_FR_OP_TXT: Serial.println("TXT"); break;
        case WS_FR_OP_BIN: Serial.println("BIN"); break;
        case WS_FR_OP_CLOSE: Serial.println("CLOSE"); break;
        case WS_FR_OP_PING: Serial.println("PING"); break;
        case WS_FR_OP_PONG: Serial.println("PONG"); break;
        case WS_FR_OP_UNSUPPORTED: Serial.println("UNSUPPORTED"); break;
      }

      n = wsclient.read();
      bool is_mask = (0x80 & n) != 0;
      n = n & 0x7F;
      total++;
      Serial.println(n, DEC);
      char mask[4];
      mask[0] = wsclient.read();
      mask[1] = wsclient.read();
      mask[2] = wsclient.read();
      mask[3] = wsclient.read();
```

```
      String cmd = "";
      while(n > 0) {
        for (char i = 0; i < 4 && n > 0; i++, n--) {
          c = wsclient.read() ^ mask[i];
          cmd += c;
          total++;
        }
      }

      const char* s = cmd.c_str();
      LED_value = fromhex(s[0]);
      R = fromhex(s[2]) << 4 | fromhex(s[3]);
      G = fromhex(s[4]) << 4 | fromhex(s[5]);
      B = fromhex(s[6]) << 4 | fromhex(s[7]);
      WiFiDrv::analogWrite(25, R);
      WiFiDrv::analogWrite(26, G);
      WiFiDrv::analogWrite(27, B);
      digitalWrite(LED_BUILTIN, LED_value);

      if (wsclient.available()) { readIMU(); continue; }

      ws_send();

    }
  } else {
    wsclient = wsserver.available();
    if (!wsclient.connected()) { wsclient.stop(); return; }
    handshake();
  }

}


int read_count = 0;

void readIMU()
{
  float Ax, Ay, Az;
  float Gx, Gy, Gz;
  float Temp;
  if (IMU.accelerationAvailable() && IMU.gyroscopeAvailable() && IMU.temperatureAvailable()) {

    IMU.readAcceleration(Ax, Ay, Az);
    IMU.readGyroscope(Gx, Gy, Gz);
    IMU.readTemperature(Temp);

    ax += Ax;
    ay += Ay;
    az += Az;
    gx += Gx;
    gy += Gy;
    gz += Gz;
    temp += Temp;
    read_count++;
  }

  if (read_count == 100) {
    AX = ax / 100.0f;
    AY = ay / 100.0f;
    AZ = az / 100.0f;
    GX = gx / 100.0f;
    GY = gy / 100.0f;
    GZ = gz / 100.0f;
    TEMP = temp / 100.0f;
    ws_send();
    ax = 0.0f;
    ay = 0.0f;
    az = 0.0f;
    gx = 0.0f;
    gy = 0.0f;
    gz = 0.0f;
    temp = 0.0f;
    read_count = 0;
  }
}

void loop()
{

  if (!wsclient.connected()) {
      check_wifi_status();
      check_web_request();
  } else {
      readIMU();
  }
  check_ws_request();
}

void printWiFiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print where to go in a browser:
  Serial.print("To see this page in action, open a browser to http://");
  Serial.println(ip);

}
```

Listing 13. C Language

### Line following with Colorsensor and UltraSonic Sensor

```
// Define pins for motor driver
const int in1Pin = 11;
const int in2Pin = 10;
const int in3Pin = 13;
const int in4Pin = 12;
const int enA = 5;
const int enB = 6;
// Define pins for ultrasonic sensor
const int trigPin = 2;
const int echoPin = 3;
// Define pins for IR sensor
const int leftsensorPin = 8;
const int rightsensorPin = 7;
// Define variables for ultrasonic sensor
long duration;
int distance;
int motorSpeed =110;
#define s0 A0       // Module pins wiring to analog pins
#define s1 A1
#define s2 A2
#define s3 A3
#define out A4
int Red = 0, Blue = 0, Green = 0;  // RGB values

void forward() {
 digitalWrite(in1Pin, LOW);
 digitalWrite(in2Pin, HIGH);
 digitalWrite(in3Pin, LOW);
 digitalWrite(in4Pin, HIGH);
}
void forwardU() {
 analogWrite(enA, motorSpeed);
 analogWrite(enB,motorSpeed );
 digitalWrite(in1Pin, LOW);
 digitalWrite(in2Pin, HIGH);
 digitalWrite(in3Pin, LOW);
 digitalWrite(in4Pin, HIGH);
}
void left() {
 analogWrite(enA, motorSpeed);
 analogWrite(enB,motorSpeed );
 digitalWrite(in1Pin, LOW);
 digitalWrite(in2Pin, HIGH);
 digitalWrite(in3Pin, HIGH);
 digitalWrite(in4Pin, LOW);
}
void stop() {
 digitalWrite(in1Pin, LOW);
 digitalWrite(in2Pin, LOW);
 digitalWrite(in3Pin, LOW);
 digitalWrite(in4Pin, LOW);
}
void right() {
 analogWrite(enA, motorSpeed);
 analogWrite(enB,motorSpeed );
 digitalWrite(in1Pin, HIGH);
 digitalWrite(in2Pin, LOW);
 digitalWrite(in3Pin, LOW);
 digitalWrite(in4Pin, HIGH);
}
void turn() {
 analogWrite(enA, 0);
 analogWrite(enB,100 );
 digitalWrite(in1Pin, LOW);
 digitalWrite(in2Pin, LOW);
 digitalWrite(in3Pin, LOW);
 digitalWrite(in4Pin, HIGH);
}
void GetColors()
{
  digitalWrite(s2, LOW);  // S2/S3 levels define which set of photodiodes we are using: LOW/LOW is for RED, LOW/HIGH is for Blue, and HIGH/HIGH is for Green
  digitalWrite(s3, LOW);
  Red = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);  // Measure the duration until "out" goes LOW, then measure until it goes HIGH again
  delay(20);
  digitalWrite(s3, HIGH);  // Select the other color (set of photodiodes) and measure the value
  Blue = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  delay(20);
  digitalWrite(s2, HIGH);
  Green = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
  delay(20);
}
int ColorSensor(){
   if (Red <= 15 && Green <= 15 && Blue <= 15) {  // If the values are low, it's likely the white color (all the colors are present)
    Serial.println("White");

  } else if (Red < Blue && Red <= Green && Red < 23) {  // If Red value is the lowest one and smaller than 23, it's likely Red
    Serial.println("Red");
    return 0;
  } else if (Blue < Green && Blue < Red && Blue < 20) {  // Same thing for Blue
    Serial.println("Blue");
    return 1;
  } else {
    Serial.println("Unknown");  // If the color is not recognized, you can add more conditions as needed
  }

  delay(1000);  // 2s delay, you can modify if you want
}
void setup() {
  Serial.begin(9600);
 // Initialize motor driver pins
 pinMode(in1Pin, OUTPUT);
 pinMode(in2Pin, OUTPUT);
 pinMode(in3Pin, OUTPUT);
 pinMode(in4Pin, OUTPUT);
 pinMode(enA, OUTPUT);
 pinMode(enB, OUTPUT);
 // Initialize ultrasonic sensor pins
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 // Initialize IR sensor pin
 pinMode(leftsensorPin, INPUT);
 pinMode(rightsensorPin, INPUT);
 // Set initial speed for both motors
 analogWrite(enA, motorSpeed);
 analogWrite(enB,motorSpeed );
    pinMode(s0, OUTPUT);    // Pin modes
   pinMode(s1, OUTPUT);
   pinMode(s2, OUTPUT);
   pinMode(s3, OUTPUT);
   pinMode(out, INPUT);

   Serial.begin(9600);   // Initialize the serial monitor baud rate

   digitalWrite(s0, HIGH); // Setting S0/S1 to HIGH/HIGH levels means the output frequency scaling is at 100%
   digitalWrite(s1, HIGH); // LOW/LOW is off, HIGH/
```

```
    LOW is 20%, and LOW/HIGH is 2%
}
void loop() {

 // Read IR sensor input
 int leftsensorValue = digitalRead(leftsensorPin);
 int rightsensorValue = digitalRead(rightsensorPin);
analogWrite(enA, motorSpeed);
 analogWrite(enB,motorSpeed );
 if (leftsensorValue == 1 && rightsensorValue == 1) {
 forward();
 } else if (leftsensorValue == 0 && rightsensorValue == 1) {
 right();
 } else if (leftsensorValue == 1 && rightsensorValue == 0) {
 left();

 }
 //color sensor
   GetColors();  // Execute the GetColors function to get the value of each RGB color

 // Ultrasonic sensor code

 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(7);
 digitalWrite(trigPin, LOW);
 duration = pulseIn(echoPin, HIGH);
 distance = duration/34.2;
 if(distance==0){
 distance=100;
 }


 if(distance < 20 && ColorSensor() == 0 )
 {
      Serial.println(distance);

 left();
 delay(900);
 forward();
 delay(1400);
 right();
 delay(1200);
 forwardU();
 leftsensorValue = digitalRead(leftsensorPin);
 rightsensorValue = digitalRead(rightsensorPin);

while(leftsensorValue == 1 && rightsensorValue == 1 ){
 leftsensorValue = digitalRead(leftsensorPin);
rightsensorValue = digitalRead(rightsensorPin);
}
 stop();
 delay(2000);
 forwardU();
 delay(100);
 rightsensorValue = digitalRead(rightsensorPin);
 turn();
 while(rightsensorValue == 1){
 rightsensorValue = digitalRead(rightsensorPin);
}

}

delay(500);
}
```

Listing 14. C Language