MINI PROJECT
INDUSTRIAL DRIVES

Bluetooth Controlled Vacuum Cleaner

Report

Submitted by

TEAM DETAILS	NAME	ROLL.NO.	DEPARTMENT
Team Member 1	Siddhant Ajay Mahalank	H030	B.Tech Mechatronics
Team Member 2	Kayzin Teshter Mirza	H033	B.Tech Mechatronics
Team Member 3	Mannchet Singh Bhaad	H051	B.Tech Mechatronics

SVKM'S NMIMS Deemed to be UNIVERSITY

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING

By Assistant Professor Nirmal Thakur

Table of contents

List figures

CHAPTER 1
Introduction

A vacuum cleaner that runs on even maneuvers around obstructions. To offer remote control and monitoring features, an Arduino-based vacuum cleaner can also be linked to other gadgets and networks, like home automation an Arduino microcontroller is known as an Arduino-based vacuum cleaner. The vacuum cleaners' motors, sensors, and other parts are all controlled by programming on the Arduino board. As a result, the vacuum cleaner can be highly customized and controlled during the cleaning process. It can be set up to clean particular areas, adjust suction power, and a system or smartphone. Because of this, the vacuum cleaner doubles as a useful cleaning tool and an entertaining and instructive project for makers and robotics enthusiasts.

1.1 Vacuum Cleaner

Often called a vacuum or hoover, a vacuum cleaner is a device that uses suction to remove debris from surfaces such as carpets, couches, curtains, and other items. Usually, it is powered by electricity. The dirt is collected using a cyclone or a dust bag for later disposal. There are many different types and sizes of vacuum cleaners used in both homes and businesses. They include small battery-powered hand-held vacuum cleaners, wheeled canister models for home use, domestic central vacuum cleaners, massive stationary industrial machines that can hold hundreds of liters of dirt before being emptied, and self-propelled vacuum trucks for cleanup of significant spills or removal of contaminated soil. Specialized shop vacuums can be used to remove liquids and solid objects from the air. The performance of a vacuum cleaner can be measured by several parameters:

1. Airflow, in litters per second [l/s] or cubic feet per minute (CFM or ft^3/min)
2. Airspeed, in meters per second [m/s] or miles per hour [mph]
3. Suction, vacuum, or water lift, in pascals [Pa] or inches of water.

CHAPTER 2
Project Description

2.1 Block diagram

The Arduino Uno is the central component of the system. It communicates with the Bluetooth module to receive commands from the user. It also communicates with the motor driver to control the DC motors. The motor driver powers the DC motors that drive the wheels and propeller. The battery provides power to all of the components. The user can use a smartphone or other Bluetooth device to control the vacuum cleaner's movement and suction power. The vacuum cleaner will continue to clean until it is stopped by the user or until the battery runs out.

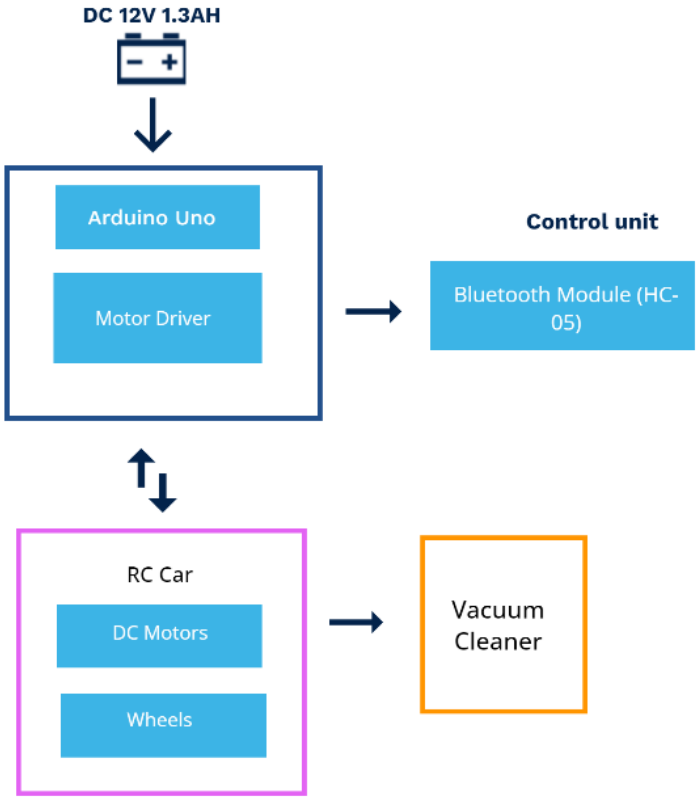

Fig 1: Block diagram

2.2 Circuit diagram

The remote-control Vacuum Cleaner's circuit design is shown in the picture () below. The brains of the apparatus are Arduino. It transports data from the Bluetooth Module to the Arduino, which then sends it to the motor and every other component in the machine. The DC motor will get the information from the motor driver, which will take it from the Arduino. When the sensor detects an obstruction in the machine's path, it will cause the bot to halt. The Bluetooth module will establish a connection with another device (such as a phone, laptop, etc.), which will transmit data and input to the computer automatically.

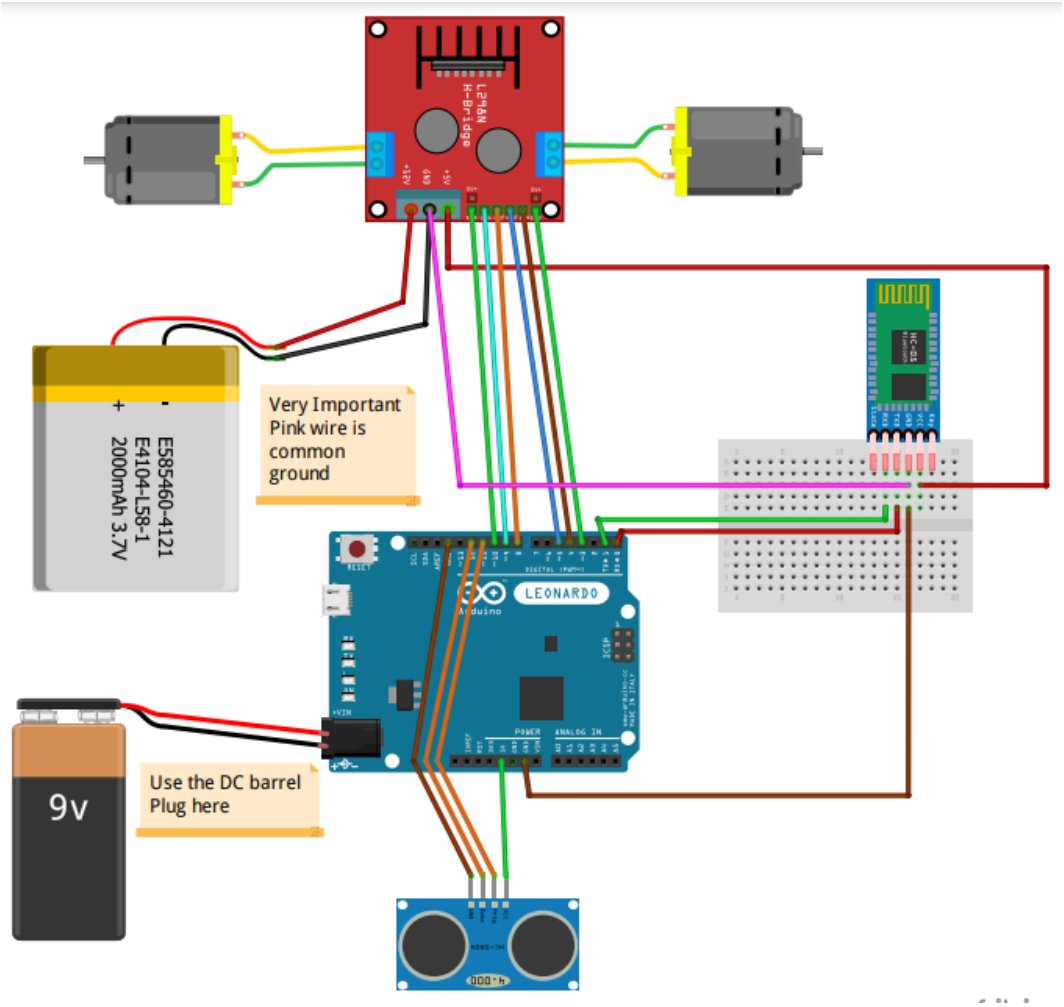

Fig 2: Circuit Diagram

2.3 Components used

i. Arduino Uno(R3)
ii. 12V Li-ion Battery
iii. Bluetooth Module (HC-05)
iv. Motor Driver (L298N H-Bridge)
v. 3 DC Motors
vi. Wires
vii. Breadboard
viii. Wheels
ix. Propeller
x. Plastic Bottle
xi. Chassis

2.4 Flowchart

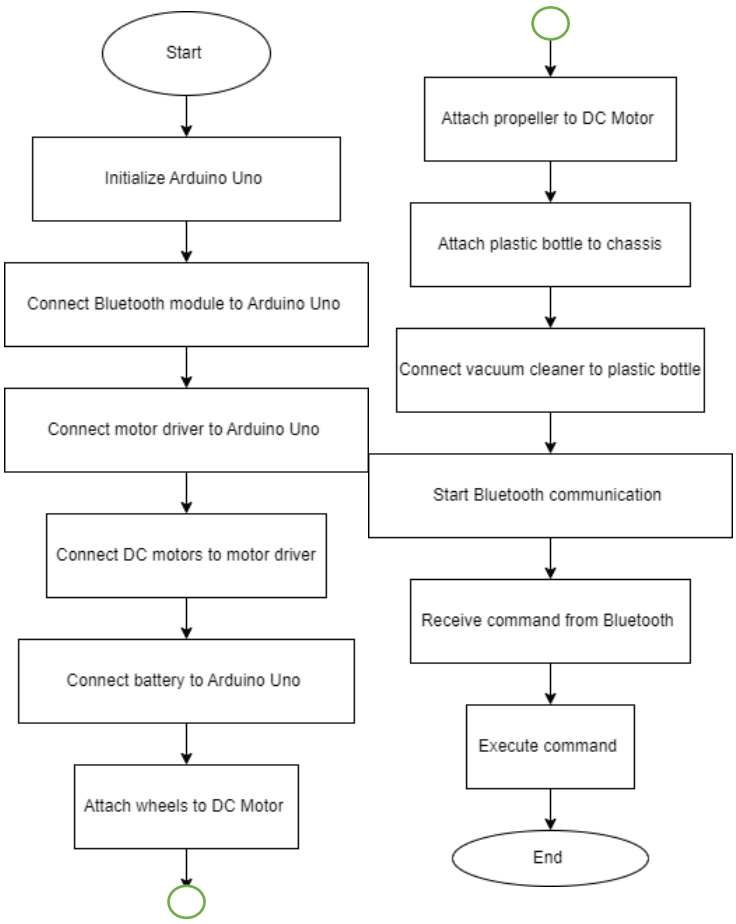

Fig 3: Flow chart

The vacuum cleaner is attached to the plastic bottle, the Arduino Uno is initialized, the battery is linked, the motor driver and Bluetooth module are connected, and the DC motors are connected to the motor driver. The Arduino Uno initiates Bluetooth connectivity and gets orders from the Bluetooth device. The commands are then carried out by the Arduino Uno, possibly involving the vacuum cleaner or DC motors being started or stopped. The process is repeated until the user ends the software or the battery runs out.

2.5 Program

The following code allows the Arduino-based robot to receive commands through Bluetooth, move in different directions, adjust its speed, and measure distances using an ultrasonic sensor. The specific actions depend on the characters received over the Bluetooth connection and the measured distance from the ultrasonic sensor.

```
//Define for Bluetooth:
char incoming_value = 0;
//Define pins for Motor:
int ENA = 10; //Enable Pin of the
Right Motor (must be PWM)
int IN1 = 9;  //Control Pin
int IN2 = 8;
int ENB = 3; //Enable Pin of the Left
Motor (must be PWM)
int IN3 = 5;
int IN4 = 4;
//Speed of the Motors
int speed = 0;
//Define for Ultrasonic:
int trigPin = 11;
int echoPin = 12;
void setup()
{
  //for Bluetooth
  Serial.begin(9600);
  //for Motor
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  //for Ultrasonic
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  //Use analogWrite to set initial
motor speed
  setSpeed(speed);
}
void loop()
{
  if (Serial.available() > 0)
  {
    delay(10);
    incoming_value = Serial.read();

    if (UltraSonic() >= 10)
    {
      {
        switch (incoming_value)
        {
          case 'F': //Right
            right();
            break;
          case 'B': //Left
            left();
            break;
          case 'L': //Back
            backward();
            break;
          case 'R': //Forward
            forward();
            break;
          case 'S': //Stop
            stop();
            break;
          case '0':
            speed = 0;
            setSpeed(speed);
            break;
          case '1':
            speed = 100;
            setSpeed(speed);
            break;
          case '2':
            speed = 120;
            setSpeed(speed);
            break;
          case '3':
            speed = 130;
            setSpeed(speed);
            break;
          case '4':
            speed = 140;
            setSpeed(speed);
            break;
          case '5':
            speed = 150;
            setSpeed(speed);
            break;
          case '6':
            speed = 160;
            setSpeed(speed);
            break;
          case '7':
            speed = 170;
            setSpeed(speed);
            break;
          case '8':
            speed = 200;
            setSpeed(speed);
            break;
          case '9':
            speed = 220;
            setSpeed(speed);
            break;
          case 'q':
            speed = 255;
            setSpeed(speed);
            break;
          default:
            Serial.println("Wrong
choice");
            break;
        } // End of switch
      }
    }
    else
    {
      switch (incoming_value)
      {
        case 'B': //Backward
          backward();
          break;
        case 'S': //Stop
          stop();
          break;
        default:
          Serial.println("Wrong
choice");
          break;
      }
    }
  }
}
void left()
{
  //Turn LEFT
  //motor A Backward
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  //motor B Forward
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  Serial.println("LEFT");
  delay(40);
}
void right()
{
  //Turn RIGHT
  //motor A Forward
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);

  //motor B Backward
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("RIGHT");
  delay(40);
}
void forward(){
digitalWrite(IN1, HIGH);
digitalWrite(IN2, LOW);
digitalWrite(IN3, HIGH);
digitalWrite(IN4, LOW);
Serial.println("FORWARD");
}
void backward()
{
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("BACKWARD");
}
void stop()
{
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  Serial.println("STOP");
}
void setSpeed(int speed)
{
  analogWrite(ENA, speed);
  analogWrite(ENB, speed);
  Serial.print("Speed: ");
  Serial.println(speed);
}
float UltraSonic()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Measure the response from the
HC-SR04 Echo Pin
  float duration = pulseIn(echoPin,
HIGH);
  // Determine distance from duration
  // Use 343 metres per second as
speed of sound
  float distance = (duration / 2) *
0.0343;

  if (distance > 400)
  {
    distance = 400;
  }
  else if (distance < 2)
  {
    distance = 2;
  }
  Serial.println(distance);
  return distance;
}
```

CHAPTER 3

Results

3.1 Prototype Images

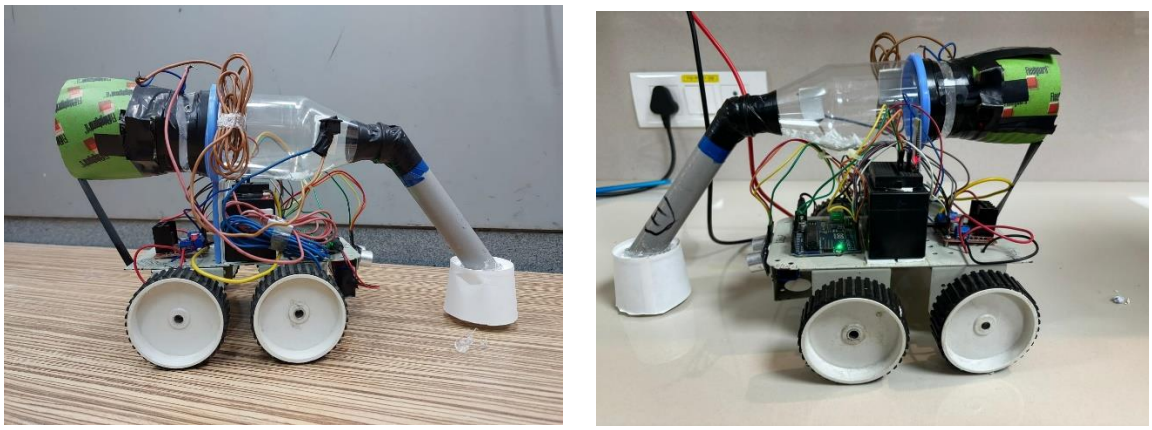

Fig 4: Prototype images

3.2 Control display

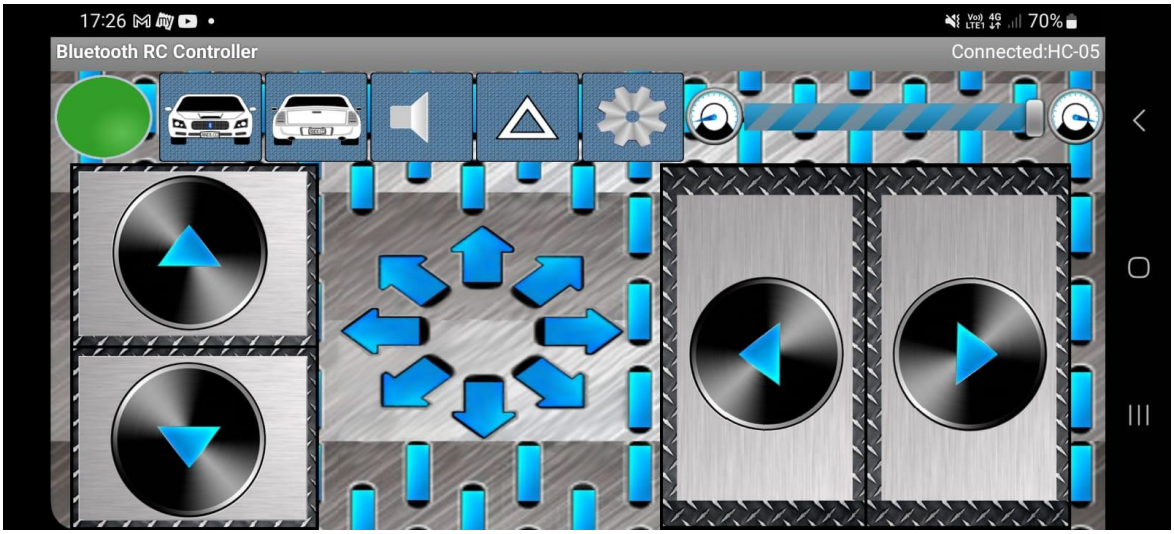

Fig 5: Control display

3.3 Advantages and Limitations

The result of using an Arduino-based vacuum cleaner is that it is less costly compared to regular remote-controlled vacuum cleaners, and it can do the cleaning of the surroundings effectively. The following are some benefits of Arduino-based smart vacuum cleaners.

- Cost-effective: Arduino-based smart vacuum cleaners are relatively inexpensive compared to commercially available vacuum cleaners, making them a cost-effective option for those on a budget.
- Customizable: Arduino provides a platform for developers to customize their smart vacuum cleaners, allowing them to add features that meet their specific needs.
- User-friendly: Arduino's programming environment is user-friendly, making it easier for non-programmers to create their smart vacuum cleaner.
- Scalability: The modular design of Arduino boards allows users to easily upgrade or add new features to their smart vacuum cleaner as needed.
- Flexibility: With an Arduino-based smart vacuum cleaner, users can choose to use different sensors, motors, and other components, allowing for greater flexibility in terms of design and functionality.
- Remote Control: The smart vacuum cleaner can be remotely controlled through a smartphone or tablet, allowing users to operate it from anywhere in their home.
- Real-time monitoring: The smart vacuum cleaner can be equipped with sensors to monitor its environment in real-time, providing data on the level of dirt and dust in the environment.
- Efficient cleaning: An Arduino-based smart vacuum cleaner can be programmed to optimize its cleaning patterns, making it more efficient in terms of cleaning time and energy consumption.

CHAPTER 4

Conclusion and future scope

A remote-control vacuum cleaner may provide insightful real-world experience, stimulating learning possibilities, and creative experimentation. We may learn more about mechanical systems, suction and air flow concepts, and basic engineering ideas by taking on this project. We may cultivate a feeling of creativity and problem-solving abilities via modification and experimentation.

Furthermore, by utilizing recycled materials and energy-efficient components, this project may function as a platform for the promotion of sustainability and eco-friendly behaviors. The initiative also promotes a greater awareness of technology and its uses in daily life and offers a useful means of supporting sustainable living habits.

All things considered, the remote-controlled vacuum cleaner is a wonderful example of learning, experimentation, and creativity. It inspires people to explore the fields of engineering and innovation while simultaneously improving their local community.

The future of robotic vacuum cleaners is promising, with ongoing technological advancements and innovations expected to drive significant developments in the field. Some key areas where the future scope of robotic vacuum cleaners is anticipated to expand include:

1) Artificial Intelligence and Machine Learning: Robotic vacuum cleaners will be able to learn from user habits, adapt to different environments more effectively, and optimize cleaning routines for increased efficiency through the integration of advanced artificial intelligence and machine learning algorithms.
2) Smart Home Integration: Future robotic vacuum cleaners should be able to easily interact with smart home systems, giving consumers remote control over them using smartphones or other smart devices. They could even be able to employ virtual assistants to give voice-activated orders.
3) Enhanced Navigation and Mapping: Robotic vacuum cleaners will likely incorporate advanced mapping and navigation technologies, such as simultaneous localization and mapping (SLAM), to create more accurate and efficient cleaning paths, as well as to better navigate complex environments.
4) Autonomous Maintenance and Upkeep: Robotic vacuum cleaners of the future might be equipped with the ability to perform self-maintenance tasks, such as emptying their dustbins or performing basic repairs, further reducing the need for human intervention.

Overall, the future scope of robotic vacuum cleaners is oriented toward creating more intelligent, adaptable, and user-friendly devices that can seamlessly integrate into modern smart homes, providing users with enhanced cleaning capabilities and convenience.

References

1) S. Harish Bala, K. P. Prajith, B. Siddharth, B. N. Prashanth, DESIGN AND ANALYSIS OF AUTOMATIC ROBOTIC VACUUM CLEANER, International Journal of Advanced Research in Engineering and Technology (IJARET) Volume 11, Issue 11, Year: 2020

2) Manreet Kaur, Preeti Abrol, Design and Development of Floor Cleaner Robot (Automatic and Manual), International Journal of Computer Applications (0975 – 8887) Volume 97– No.19, Year: 2014