INTERNATIONAL JOURNAL OF CREATIVE RESEARCH AND STUDIES
www.ijcrs.org ISSN-0249-4655

THE MAKING OF A VOICE-COMMANDED AUTOMATIC VACUUM CLEANER OUT OF ARDUINO INTERFACE AND VOICE MODULE

Randy C. Nilles MAIE,[1,2,3,4] **Elijah James P. Sikat,**[1,2,3] **Marc Laurence G. Magnaye,**[1,2,3] **Kirsten Jeff C. Garces,**[1,2,3] **Jesse Noah A. Castillo,**[1,2,3] **Izz al-Din P. Ibrahim,**[1,2,3] **Clint Deelan T. Quitalig,**[1,2,3]

Philippine School Doha, Doha, Qatar, PO Box 19664,
Quality Assurance and Accreditation Department, PSD, Doha Qatar
Research Capstone Project, PSD, Doha, Qatar

ABSTRACT
Concerns regarding the health and welfare of people are the most important and pressing aspects of climate and environmental change. Compared to outside exposure, indoor air pollution presents detrimental effects on health, this study then aims to provide a solution through the creation of a Voice Commanded Automatic Vacuum Cleaner utilizing Arduino Uno R3 and a voice module as a main component. The quantitative experimental research method was used to determine the obstacle detection, maximum detectable range, and latency of vacuum cleaner. The results of this study revealed that the vacuum cleaner robot excelled in obstacle detection, achieving 100% success rates on walls and chair legs, and a 75% success rate on table legs with trials. It detected voice commands up to a range of 10 meters and efficiently responded with an average latency of 0.83 seconds. The sensor responsiveness in terms of obstacle detection achieved a 100% success rate on both the wall and chair leg obstacles, as well as a 75% success rate on the table leg obstacle. The study's findings demonstrated that the voice-commanded vacuum cleaner was effective in terms of obstacle detection, as well as the voice module's efficacy in terms of command maximum detectable range and the command latency between audible commands. As discussed, the Voice Commanded Automatic Vacuum Cleaner is beneficial to people as it is efficient, easy to use, has a big impact on the home automation market, and possibly encourages the widespread use of voice-activated appliances to make the vacuum cleaner more accessible to people with impairments or mobility challenges.

KEYWORDS: *Arduino Interface, Obstacle Detection, Voice Module*

1. INTRODUCTION

As science and technology progress, chip processing performance continues to improve, and speech recognition technology has made significant strides. There are several products available on the market that support speech recognition. Speech recognition and control systems with high reliability and recognition rates are critical for popularizing and promoting speech recognition technology (Wu et al., 2022).

This study focuses on making a Voice-Commanded Automatic Vacuum Cleaner out of the Arduino interface and voice module. A traditional manufactured automatic vacuum cleaner is a self-propelled floor cleaner that uses brushes, a spinning brush, or an air-driven turbine to remove dirt and debris from carpets and hard floors. Automatic vacuum cleaners use significantly less electricity per unit of time than their manual counterparts, which is why they are classified as “energy-saving” appliances. Even when used more frequently, robotics use less energy (Nicholls & Strangers, 2019). Robot vacuums are one example of everyday automation that might be used to research human-automation interactions.

The study's goal is to combine the features of a voice command with the automatic vacuum cleaner to create a voice-commanded automatic vacuum cleaner. Additionally, the study aims to improve the overall user experience by allowing for hands-free operation of the vacuum cleaner. The use of voice commands also makes the device more accessible for individuals with mobility issues or disabilities.

No component of climate and environmental change is more crucial or urgent than those that have an impact on people's health and well-being, especially the most vulnerable. Indoor and ambient air pollution claims more than seven million lives each year. The burning of biomass, the use of fossil fuels, and agriculture are the main causes of air pollution. No area of policy in the world routinely accounts for the possible costs and benefits of every action for public health, not even those that directly result in health-influencing externalities (Neira & Ramanathan, 2020).

The Gulf Cooperation Council (GCC) countries have been focusing on outdoor pollution, but few have addressed indoor air quality. The Qatar Environment and Energy Research Institute (QEERI) observed that Qatar experienced a 30% decrease in pollutant concentrations in Doha due to people staying indoors, however, Qatar experienced poor outdoor air quality in 2021, earning a US air quality index score of 111. Vulnerable groups should wear masks and close doors and windows to prevent contaminated air. Indoor air pollution poses a larger threat and may result in more harmful health impacts than outdoor exposure. Social and Economic Survey Research Institute (SESRI) observed that Ischemic heart disease accounts for 12% of all fatalities caused by household air pollution. The GCC region faces similar negative effects on human health, with unfavorable meteorological conditions forcing citizens to spend more time indoors. Urgent governmental intervention and control measures are needed, along with increased research on health consequences (Abusin et al., 2022)

Arduino is an open-source platform for building and programming electronics. It can receive and send information to most devices, as well as command the specific electronic device via the internet. To program the board, it makes use of an Arduino Uno circuit board and a software program called Simplified C++ (Badamasi, 2014). In a study utilizing an arduino-based remote and autonomous controlled robotic car with real time obstacle detection and avoidance, the Arduino Uno, which uses 14 pins, served as the device's brain. The most popular Arduino card is the Arduino UNO models. It can be easily programmed with Arduino libraries. Other microprocessors have a significant advantage in their ease of programming. Programming on the Arduino Uno requires the integrated development environment (IDE) (Yılmaz & Özyer, 2019). In another study developing a bluetooth based tracking system for tracking devices, Arduino UNO R3 was used as its main component a real-time device is needed such as the Arduino. The bluetooth based tracking system utilizing the Arduino UNO R3 is a low-power device with a 5V battery, making it highly portable. It is not only used to monitor items, but also to track them in real time (Adjei et al., 2020).

A voice module is a compact and simple speech recognition board that can be easily interfaced with Arduino. The voice module is speaker-dependent and can support up to 80 voice commands in total. In real-world applications of the intended systems for speech contact with the management system, a problem with additive sounds for the specific surroundings must be resolved. Voice management in smart homes is primarily favored by elderly or disabled neighbors. The entire project could be conceptually split into three parts: the creation of applications for speech recognition, programming and animation, and the execution of a communication interface (Vanus et al., 2015). Hence, the creation of a voice-commanded automatic vacuum cleaner through Arduino along with a compact voice module aids in accurately navigating surroundings and following trained voice commands to clean dust and dirt.

One of the better examples of robotic technology for usage in homes is the robot vacuum cleaner. The robot Hoover cleaner is a machine made to remove all debris from flat surfaces without the need for human assistance. They now have additional features including being lightweight, remote-controlled, autonomous, and taking up little space. The primary distinction between conventional and automatic vacuum cleaners is that the former move about to do cleaning tasks without the need for human assistance. In addition, hoover cleaner robots are far smaller and quieter than traditional ones Unfortunately, because of their expensive price, many people now cannot afford robot vacuum cleaners. These robots must be affordable and useful for them to become widely used. The majority of the market's low-cost vacuum cleaner robots employ simple switches as sensors to identify impediments which present a variety of functional inefficiencies, making it a less desirable alternative to a manual vacuum, and which present the argument that only the priciest robot vacuum cleaners are functionally useful (Eren & Doğan, 2022).

This study benefits the Qatari and Filipino communities, and future researchers. The outcomes of this study will help the Philippine School Doha community of students, teachers, and non-teaching staff by notifying them of the dangers of indoor air pollution. PSD students and staff must be aware of the health implications as Qatar residents. Additionally, this research informs the PSD community that the Arduino interface and voice module can be used to create environmentally friendly solutions like a voice-commanded automatic vacuum cleaner. This study could also assist Qatar by promoting improved strategies to prevent indoor dust pollution. As a result, Qatar will be less likely to have a household with breathable dirty air that is harmful to their health. Besides, because it employs the Arduino interface and Voice module to create a voice-commanded automatic vacuum cleaner, this research guarantees the population of Qatar a better and more sustainable standard of living. In rural parts of the Philippines, the Voice Commanded Automatic Vacuum Cleaner provides a cheap and reliable source of cleanliness. This study would also help Filipinos understand the dangers of breathing dirty air and the importance of doing so. Moreover, future researchers can use the results, data, and information presented in this study as references while doing their research on voice-commanded automatic vacuum cleaners. They can utilize this study to check the accuracy of other research in the field. As a result of this research, issues in their investigations will be reduced, as will the use of similar factors and procedures to create higher-quality systems.

2. STATEMENT OF PROBLEM

The objective of the study is to make a Voice-Commanded Automatic Vacuum Cleaner out of Arduino interface and voice module. Specifically, it aims to answer the following questions:

1) How accurate is the voice-commanded automatic vacuum cleaner in detecting obstacles in terms of sensor responsiveness?
2) What is the maximum detectable range of the voice module in response to a command at the following distances in meters? and
3) How long is the latency between the audible commands and the action of the vacuum cleaner in terms of seconds?

3. PURPOSES OF THE STUDY

The purpose of the study is to develop a Voice-Commanded Automatic Vacuum Cleaner using an Arduino Uno R3 and voice module, integrating the convenience and effectiveness of a voice command system with an Arduino interface that addresses indoor and ambient air pollution in households while improving on the automation of traditional automatic vacuum cleaners found in the market.

4. OBJECTIVES OF THE STUDY

The objective of the study is to create a low-cost Voice-Commanded Automatic Vacuum Cleaner using an Arduino interface and voice module that is responsive over long distances, with low-latency response, and high accuracy in obstacle detection in terms of sensor responsiveness. This study also assesses the practicality and usability of the development system considering factors such as the reliability and energy efficiency of the device. Hence the study also points out the differentiation of the Voice-Commanded Automatic Vacuum Cleaner's abilities compared to other factory made automatic vacuum cleaners. Additionally, The objective of the Voice-Commanded Automatic Vacuum Cleaner is to improve the indoor air quality of our homes as there is always dust that is being generated due to the amount of time being spent inside the house.

5. RESEARCH HYPOTHESES

H1: The Voice-Commanded Automatic Vacuum Cleaner can be developed with improved functionality and efficiency compared to traditional commercialized automated vacuum cleaners.

6. RESEARCH METHODOLOGY

6.1. Research Design

This study will utilize the experimental research design. Experimental study design provides data that has a strong causal validity. Causal validity is the accuracy of statements about the cause-and-effect relationship. Controlling variables helps to optimize internal validity, support causal inferences, and ensure trustworthy results. It also helps to isolate and adjust causal effects (Lewis, 2020). This formulation emphasizes that a quantitative dimension is present in work and is designed as a tool to assist in improving research designs (Corte & Aspers, 2019). The main framework for social science research is quantitative methods. It describes a collection of approaches, methods, and presumptions used to investigate numerical patterns to research psychological, social, and economic phenomena. Many numbers are collected during quantitative research. Creating knowledge and insight into the social world is the goal of quantitative research. To observe phenomena or events that have an impact on people, researchers utilize quantitative research (Coghlan & Brydon-Miller, 2014). In this study, the Arduino interface and voice module are the independent variables, and the voice-commanded automatic vacuum is the dependent variable.

6.2. Research Locale

The research study will be conducted at Philippine School Doha in Doha, State of Qatar, specifically in the Mesaimeer Area (Zone 56), Al Khulaifat Al Jadeeda Street (St. 1011), as the researchers are not only students of this school but will also facilities present in the school that will enable them to make their product.

6.3. Data Gathering Procedure

The procedure shows the step-by-step process that shows and instructs how to make a voice-commanded automatic vacuum cleaner out of an Arduino interface and voice module.

Ensuring the protection and maintaining safety

Wear personal protective equipment such as safety goggles, safety gloves, safety shoes, and a laboratory coat while performing the procedures below to avoid hazardous conditions.

Making the Enclosure Cutout

1. Using a marker, mark out a circle with a diameter of 28 cm on an illustration board.
2. Cut along the marked circle using a pair of sharp scissors.
3. Using a marker, mark out an 8 cm x 8 cm square at the center of the circle for the fan.
4. Cut along the marked square using a cutter.
5. Using a marker, mark out two 7 cm x 2.7 cm strips for the wheels on both the left and right sides of the 8 cm x 8 cm center square.
6. Cut along both marked strips using a cutter.
7. For the border of the 28 cm diameter circle base, mark out an 88 cm x 4.5 cm strip from an illustration board using a marker.
8. Cut along the marked strip using a cutter.
9. Using a marker, mark out two 1.5 cm diameter circles 1 cm away from each other on the left side of the strip and another pair of circles on the right side of the strip for the eyes of the ultrasonic sensor.
10. Cut along the marked circles using a cutter.
11. Using a marker, mark out a 2 cm x 2 cm square on the bottom midpoint of the eyes for the Type A to Type B cable port.
12. Cut out along the marked-out square using a cutter.
13. Using a cutter, make a small circular cut at the opposite side of the hole of the Type A to Type B cable port where the voice module port will be positioned.
14. For the border of the fan, mark out a 32 cm x 8 cm strip from an illustration board using a marker.
15. Cut along the marked-out strip using the cutter.
16. Fold the fan border strip into four even 8 cm spaces.
17. Using a marker, mark out another circle with a diameter of 28 cm on an illustration board to serve as the dust compartment.
18. Cut along the marked circle with a pair of sharp scissors.
19. Repeat steps 3-4 for the dust compartment circle.
20. Using a marker, mark out 4 identical 7 cm x 4 cm strips for the fan barrier
21. Cut along the marked strips using a cutter.
22. Fold all four strips in half horizontally.
23. For the border of the 28 cm diameter dust compartment circle, cut out a 99 cm x 10.5 cm strip from an illustration board using a cutter.
24. Using a marker, mark out a 5 cm x 4 cm rectangle on the left side of the bottom of the dust compartment strip and another one on the right side of the bottom strip for the body of the ultrasonic sensor.
25. Cut along the marked rectangles using a cutter.
26. Using a marker, mark out a 1 cm x 1 cm square on the bottom midpoint of the rectangles for the Type A to Type B cable port.
27. Cut along the marked-out square using a cutter.
28. Repeat steps 1-2 for the cover of the dust compartment.
29. Mark out an 8 cm x 3 cm strip for the handle of the cover of the dust compartment.
30. Cut along the marked strip using a cutter.
31. Using a marker, mark out a trapezoid with sides 12 cm, 9 cm, and 3 cm for the vent.
32. Cut along the marked-out trapezoid using a cutter.

Assembling the Enclosure Cutout

1. Using a glue gun, affix the 95 cm x 45 cm strip around the 28 cm diameter circle to act as the interior wiring walls.
2. Using a glue gun, affix the 80 mm x 80 mm DC Cooling Fan inside the 8 cm x 8 cm hole with its intake side facing upwards so that its base is held in place by the illustration board's bottom most part.
3. Using a glue gun, affix the quarter folded 32 cm x 3cm square strip around the 80 mm x 80 mm DC Cooling Fan.
4. Using a glue gun, insert and attach the eyes of the HC-SR04 Ultrasonic Module to the two 1.5 cm diameter circles 1 cm away from each other on the strip to serve as the vacuum's front eyes.
5. Repeat Step 4 for the other HC-SR04 Ultrasonic Module to serve as the vacuum's rear eyes.
6. Using a glue gun, attach the two 3.7V Battery 2 Position Battery Holders back to back with its battery cell sides facing outwards.
7. Using a glue gun, affix the two 3.7V Battery 2 Position Battery Holders to the chosen right side of the fan barrier.
8. Using a glue gun, secure the ZK-12KX V2Buck Converter diagonally to the right, adjacent to the 80 mm x 80 mm DC Cooling Fan at the selected front side.
9. Using a glue gun, secure the breadboard diagonally to the left, adjacent to the 80 mm x 80 mm DC Cooling Fan at the selected front side.
10. Using a glue gun, secure the Arduino Uno R3 diagonally to the left, adjacent to the 80 mm x 80 mm DC Cooling Fan at the selected back side.
11. Using a glue gun, secure the Voice Recognition Module diagonally to the right, adjacent to the 80 mm x 80 mm DC Cooling Fan at the selected back side.
12. Using a glue gun, secure the L298N Motor Driver in between the Voice Recognition Module and Arduino Uno behind the 80 mm x 80 mm DC Cooling Fan.
13. Using a glue gun, secure the 5V Single-Channel Relay in front of the breadboard.
14. Using a glue gun, affix the edges of the four trapezoids with sides 12 cm, 9 cm, and 3 cm, to each other to form a square funnel-shaped vent.
15. Using a glue gun, secure the smaller-sized apex of the vent to underside edges of the 80 mm x 80 mm DC Cooling Fan.
16. Attach the TT Motor Wheel to the shaft of the TT Motor- Metal Gear.
17. Repeat Step 16 for the other TT Motor Wheel and TT Motor- Metal Gear.
18. Using a glue gun, secure the TT Motor- Metal Gear to the underside of the 28 cm diameter circle base, positioning it beside and parallel to the 7 cm x 2.7 cm strip designated for the wheels so that the TT Motor Wheel passes through the strip.
19. Repeat Step 18 for the other TT Motor Wheel and TT Motor- Metal Gear.
20. Using a cutter, cut out three 16.5 cm support sticks from chopsticks.
21. Using a cutter, cut out an 8 cm support stick from a chopstick.
22. Using a glue gun, secure one of the 16.5 cm support sticks behind the vent.
23. Using a glue gun, secure the second 16.5 cm support stick on the component side of the vacuum, behind the 80 mm x 80 mm DC Cooling Fan, and between the TT Motor Wheels.
24. Using a cutter, cut out two small parallel holes on the two parallel sides of the vent beside each wheel.
25. Using a glue gun, secure the third 16.5 cm support stick through the two cut holes and below the TT Motor- Metal Gears.
26. Using a glue gun, affix the 28 cm diameter dust compartment circle along the midpoint of the 99 cm x 10.5 cm border.

27. Using a glue gun, secure the half-folded fan barrier diagonally adjacent to one corner of the 8 cm x 8 cm square at the center of the dust compartment circle.
28. Repeat Step 27 for the rest of the 3 half-folded fan barriers and 3 corners.
29. Using a glue gun, attach the 8 cm x 3 cm strip handle on the center of the 28 cm diameter dust compartment cover.

Making Connections

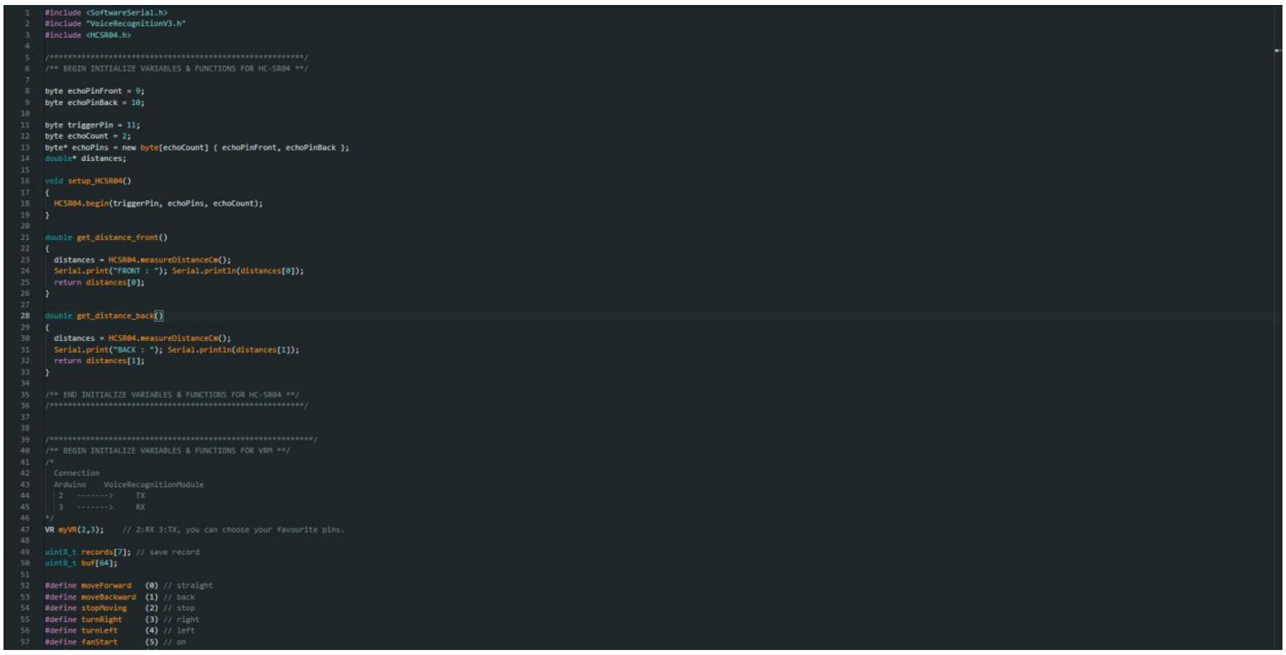
1. Connect the 14.8 Volt DC batteries to the ZK-12KX V2 Buck converter.
2. Connect the buck converter to the 5VDC relay, 12DC Cooling fan, DC Motor drive, Arduino UNO, and ultrasonic sensors with an (+)11.1VDC charge.
3. Connect the three 3.7V batteries to the Arduino UNO upper-left most ground pin.
4. Connect pin 13 of the Arduino to the 5VDC relay. This relay acts as a switch for the 12VDC cooling fan.
5. Connect pins 12, 5, and 4 of the Arduino to the DC motor driver.
6. Connect pin 11 of the Arduino to both ultrasonic sensors.
7. Connect pin 10 of the Arduino to the front ultrasonic sensor.
8. Connect pin 9 of the Arduino to the back ultrasonic sensor.
9. Connect pins 8, 7, and 6 of the Arduino to the DC Motor driver.
10. Connect pins 3 and 2 to the Voice recognition module.
11. Connect the 5V pin (Left) and the GRN pin below it to the Voice recognition module.
12. Connect the GRN above the VIN pin and the VIN pin itself to the Voice recognition module.
13. Connect the VIN to the DC Motor driver with a flow of 5VDC(+)
14. Connect the DC motor driver to both the TT motor engines.
15. Connect the 5VDC relay to the 12VDC Cooling fan.

Programming the Commands

1. Install the library of the L298N Motor Driver's variables and functions.
2. Install the library of the HC-SR04 ultrasonic sensor's variables and functions.
3. Configure the trigger and echo pins on the HC-SR04 sensor.
4. Define functions for establishing and measuring front and rear distances.
5. Install the library of the Voice recognition module’s variables and functions.
6. Define the pins for the VRM link and note down instructions.
7. Configure the fan control, movement, and load commands for the VRM.
8. Define functions for printing VRM information.
9. Set the L298N motor driver's variables and functions to their initial values.
10. Define pins for connection and setup functions of the L298N motor driver.
11. Set up the vacuum system's variables and functions upon startup.
12. Define safe distances and movement commands, as well as pins and variables, for the vacuum system.
13. Set up the vacuum system by defining the pin modes and default commands.
14. Define the functions and commands for the movement of the vacuum cleaner (forward, backward, stop, turn right, turn left)
15. Configure the functions for controlling vacuum movement in response to commands and distance.
16. Establish serial communication between Arduino and delay to enable other devices to start.
17. Execute the setup functions for the HC-SR04, VRM, L298N, and vacuum system.
18. Enter the main loop.
19. Utilize the VRM to recognize commands and carry out the appropriate actions.

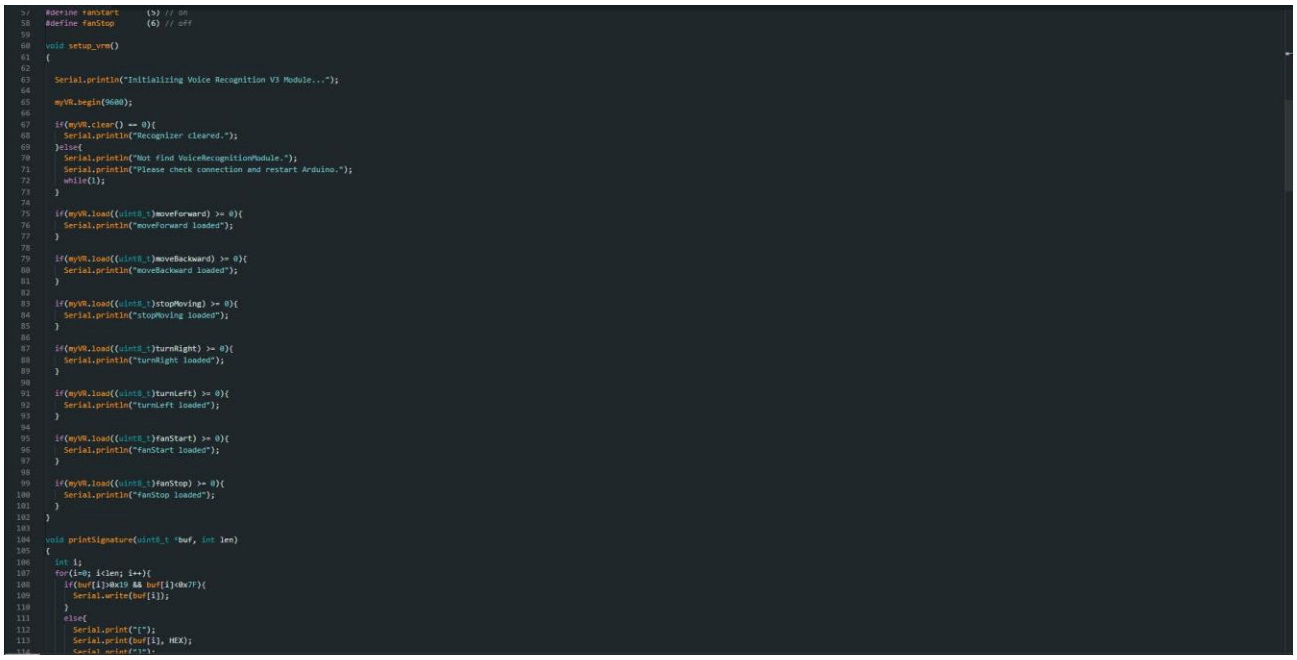
20. Print VRM data and process vacuum commands in response to detected actions.
21. Repeat the loop to constantly monitor and execute commands.

Figure 1: Entirety of Code made in Arduino IDE 2.2.1



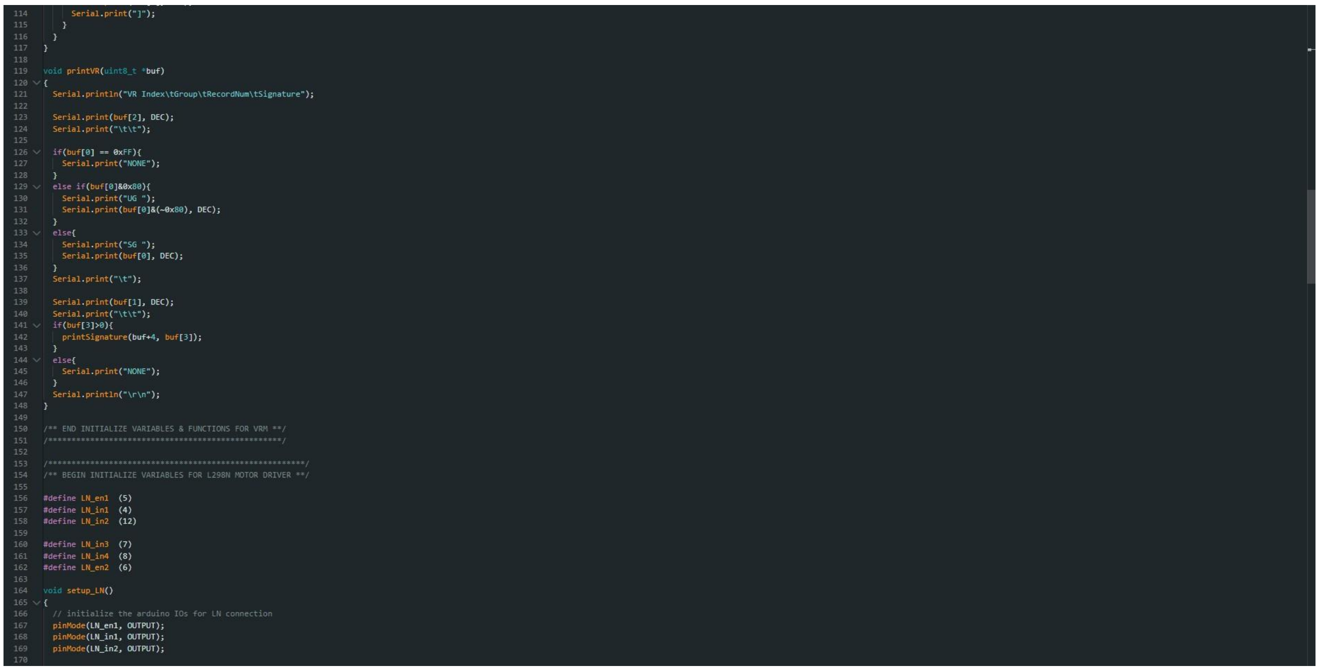
```
#include <SoftwareSerial.h>
#include "VoiceRecognitionV3.h"
#include <HCSR04.h>

/*********************************************************/
/** BEGIN INITIALIZE VARIABLES & FUNCTIONS FOR HC-SR04 **/

byte echoPinFront = 9;
byte echoPinBack = 10;

byte triggerPin = 11;
byte echoCount = 2;
byte* echoPins = new byte[echoCount] { echoPinFront, echoPinBack };
double* distances;

void setup_HCSR04()
{
  HCSR04.begin(triggerPin, echoPins, echoCount);
}

double get_distance_front()
{
  distances = HCSR04.measureDistanceCm();
  Serial.print("FRONT : "); Serial.println(distances[0]);
  return distances[0];
}

double get_distance_back()
{
  distances = HCSR04.measureDistanceCm();
  Serial.print("BACK : "); Serial.println(distances[1]);
  return distances[1];
}

/** END INITIALIZE VARIABLES & FUNCTIONS FOR HC-SR04 **/
/*********************************************************/


/*********************************************************/
/** BEGIN INITIALIZE VARIABLES & FUNCTIONS FOR VRM **/
/*
  Connection
  Arduino    VoiceRecognitionModule
   2   ------->     TX
   3   ------->     RX
*/
VR myVR(2,3);    // 2:RX 3:TX, you can choose your favourite pins.

uint8_t records[7]; // save record
uint8_t buf[64];

#define moveForward   (0) // straight
#define moveBackward  (1) // back
#define stopMoving    (2) // stop
#define turnRight     (3) // right
#define turnLeft      (4) // left
#define fanStart      (5) // on
```

1.1 **Code Lines 1-57**



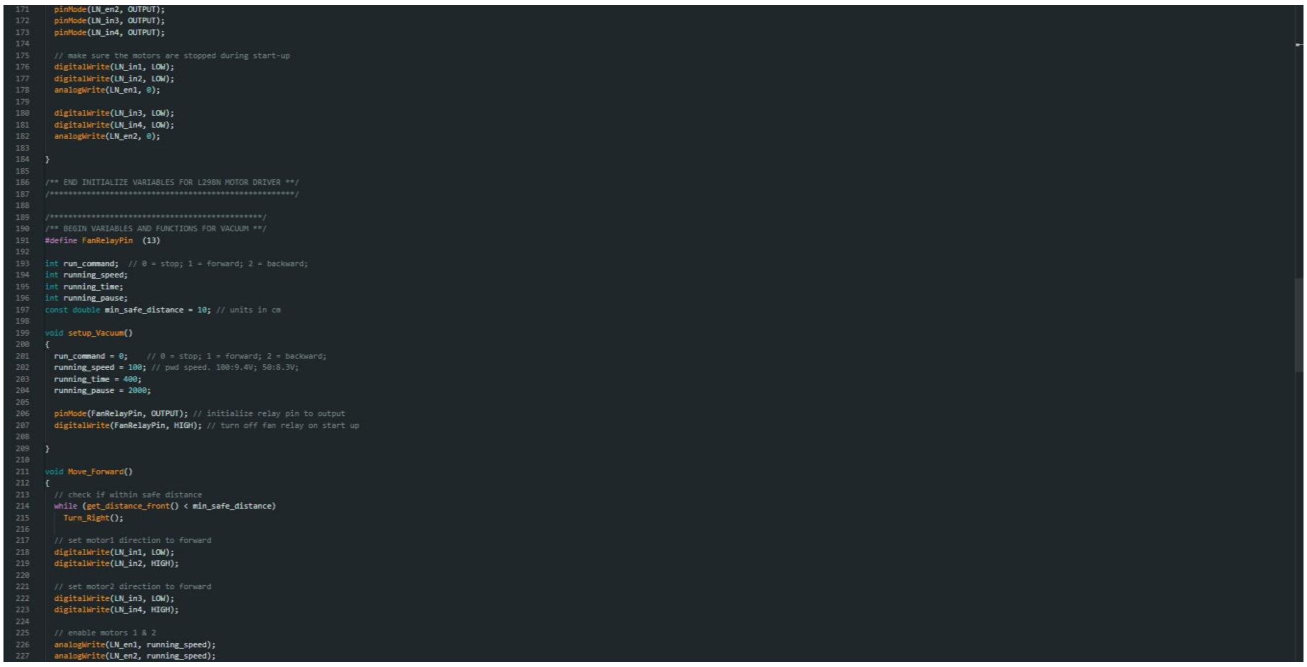
```
#define fanStart      (5) // on
#define fanStop       (6) // off

void setup_vrm()
{

  Serial.println("Initializing Voice Recognition V3 Module...");

  myVR.begin(9600);

  if(myVR.clear() == 0){
    Serial.println("Recognizer cleared.");
  }else{
    Serial.println("Not find VoiceRecognitionModule.");
    Serial.println("Please check connection and restart Arduino.");
    while(1);
  }

  if(myVR.load((uint8_t)moveForward) >= 0){
    Serial.println("moveForward loaded");
  }

  if(myVR.load((uint8_t)moveBackward) >= 0){
    Serial.println("moveBackward loaded");
  }

  if(myVR.load((uint8_t)stopMoving) >= 0){
    Serial.println("stopMoving loaded");
  }

  if(myVR.load((uint8_t)turnRight) >= 0){
    Serial.println("turnRight loaded");
  }

  if(myVR.load((uint8_t)turnLeft) >= 0){
    Serial.println("turnLeft loaded");
  }

  if(myVR.load((uint8_t)fanStart) >= 0){
    Serial.println("fanStart loaded");
  }

  if(myVR.load((uint8_t)fanStop) >= 0){
    Serial.println("fanStop loaded");
  }
}

void printSignature(uint8_t *buf, int len)
{
  int i;
  for(i=0; i<len; i++){
    if(buf[i]>0x19 && buf[i]<0x7F){
      Serial.write(buf[i]);
    }
    else{
      Serial.print("[");
      Serial.print(buf[i], HEX);
      Serial.print("]");
```

1.2 **Code Lines 58-113**



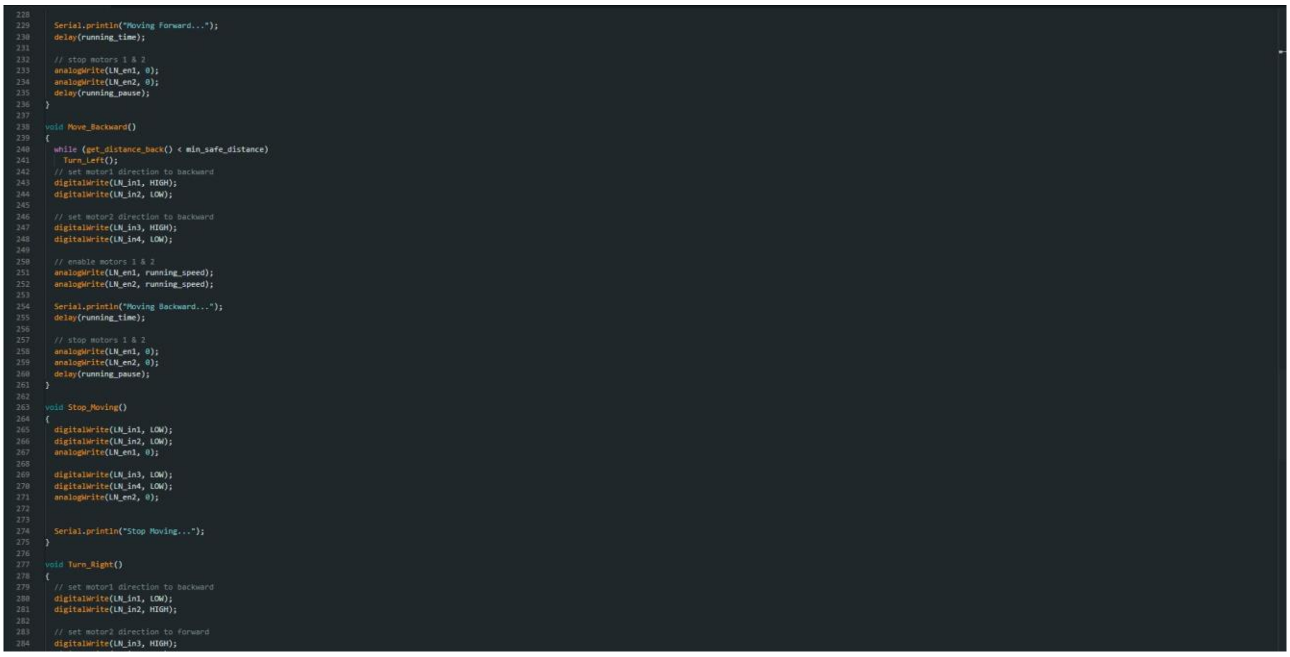
```
      Serial.print("]");
    }
  }
}

void printVR(uint8_t *buf)
{
  Serial.println("VR Index\tGroup\tRecordNum\tSignature");

  Serial.print(buf[2], DEC);
  Serial.print("\t\t");

  if(buf[0] == 0xFF){
    Serial.print("NONE");
  }
  else if(buf[0]&0x80){
    Serial.print("UG ");
    Serial.print(buf[0]&(~0x80), DEC);
  }
  else{
    Serial.print("SG ");
    Serial.print(buf[0], DEC);
  }
  Serial.print("\t");

  Serial.print(buf[1], DEC);
  Serial.print("\t\t");
  if(buf[3]>0){
    printSignature(buf+4, buf[3]);
  }
  else{
    Serial.print("NONE");
  }
  Serial.println("\r\n");
}

/** END INITIALIZE VARIABLES & FUNCTIONS FOR VRM **/
/****************************************************/

/******************************************************/
/** BEGIN INITIALIZE VARIABLES FOR L298N MOTOR DRIVER **/

#define LN_en1  (5)
#define LN_in1  (4)
#define LN_in2  (12)

#define LN_in3  (7)
#define LN_in4  (8)
#define LN_en2  (6)

void setup_LN()
{
  // initialize the arduino IOs for LN connection
  pinMode(LN_en1, OUTPUT);
  pinMode(LN_in1, OUTPUT);
  pinMode(LN_in2, OUTPUT);

```

1.3 Code Lines 114-170



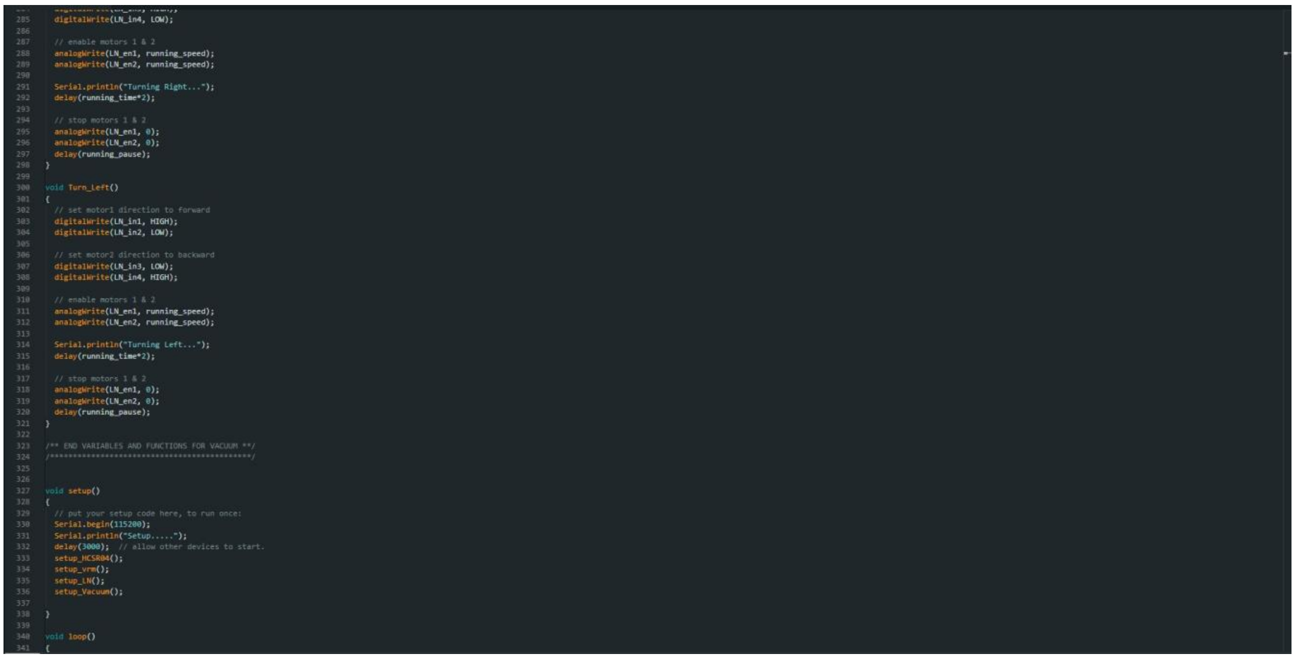
```
  pinMode(LN_en2, OUTPUT);
  pinMode(LN_in3, OUTPUT);
  pinMode(LN_in4, OUTPUT);

  // make sure the motors are stopped during start-up
  digitalWrite(LN_in1, LOW);
  digitalWrite(LN_in2, LOW);
  analogWrite(LN_en1, 0);

  digitalWrite(LN_in3, LOW);
  digitalWrite(LN_in4, LOW);
  analogWrite(LN_en2, 0);

}

/** END INITIALIZE VARIABLES FOR L298N MOTOR DRIVER **/
/******************************************************/

/*********************************************/
/** BEGIN VARIABLES AND FUNCTIONS FOR VACUUM **/
#define FanRelayPin  (13)

int run_command;  // 0 = stop; 1 = forward; 2 = backward;
int running_speed;
int running_time;
int running_pause;
const double min_safe_distance = 10; // units in cm

void setup_Vacuum()
{
  run_command = 0;    // 0 = stop; 1 = forward; 2 = backward;
  running_speed = 100; // pwd speed. 100:9.4V; 50:8.3V;
  running_time = 400;
  running_pause = 2000;

  pinMode(FanRelayPin, OUTPUT); // initialize relay pin to output
  digitalWrite(FanRelayPin, HIGH); // turn off fan relay on start up

}

void Move_Forward()
{
  // check if within safe distance
  while (get_distance_front() < min_safe_distance)
    Turn_Right();

  // set motor1 direction to forward
  digitalWrite(LN_in1, LOW);
  digitalWrite(LN_in2, HIGH);

  // set motor2 direction to forward
  digitalWrite(LN_in3, LOW);
  digitalWrite(LN_in4, HIGH);

  // enable motors 1 & 2
  analogWrite(LN_en1, running_speed);
  analogWrite(LN_en2, running_speed);
```

1.4 Code Lines 171-227



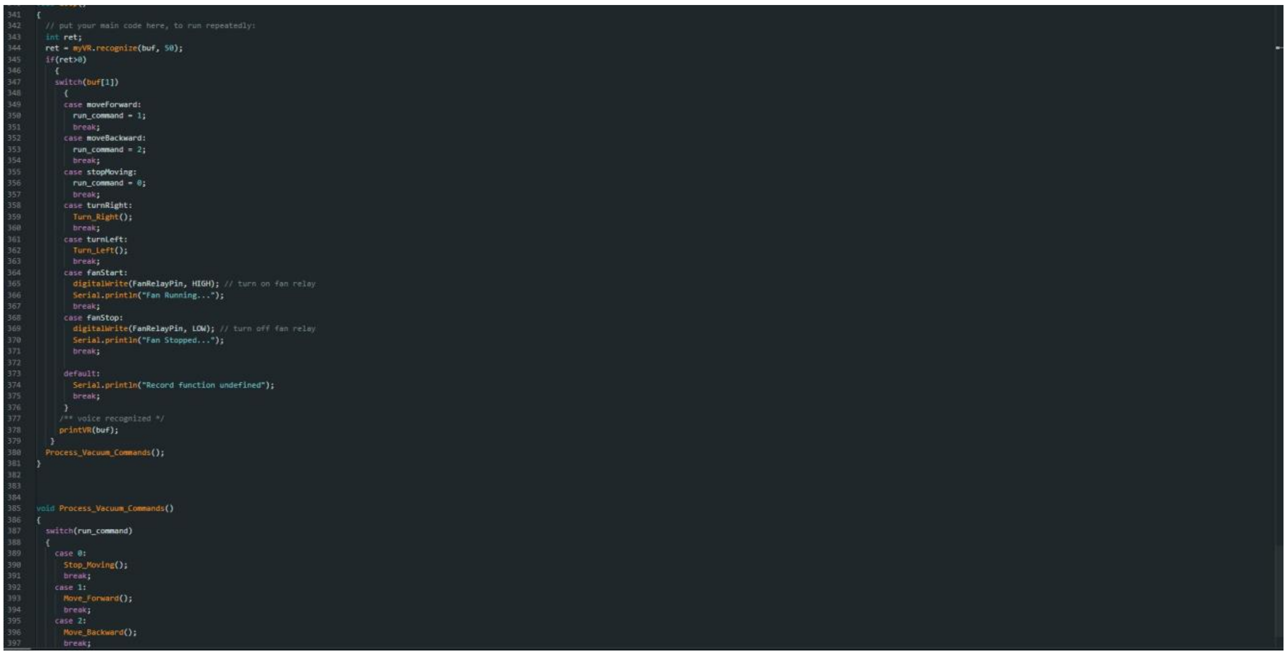
```
228
229     Serial.println("Moving Forward...");
230     delay(running_time);
231
232     // stop motors 1 & 2
233     analogWrite(LN_en1, 0);
234     analogWrite(LN_en2, 0);
235     delay(running_pause);
236   }
237
238   void Move_Backward()
239   {
240     while (get_distance_back() < min_safe_distance)
241       Turn_Left();
242     // set motor1 direction to backward
243     digitalWrite(LN_in1, HIGH);
244     digitalWrite(LN_in2, LOW);
245
246     // set motor2 direction to backward
247     digitalWrite(LN_in3, HIGH);
248     digitalWrite(LN_in4, LOW);
249
250     // enable motors 1 & 2
251     analogWrite(LN_en1, running_speed);
252     analogWrite(LN_en2, running_speed);
253
254     Serial.println("Moving Backward...");
255     delay(running_time);
256
257     // stop motors 1 & 2
258     analogWrite(LN_en1, 0);
259     analogWrite(LN_en2, 0);
260     delay(running_pause);
261   }
262
263   void Stop_Moving()
264   {
265     digitalWrite(LN_in1, LOW);
266     digitalWrite(LN_in2, LOW);
267     analogWrite(LN_en1, 0);
268
269     digitalWrite(LN_in3, LOW);
270     digitalWrite(LN_in4, LOW);
271     analogWrite(LN_en2, 0);
272
273
274     Serial.println("Stop Moving...");
275   }
276
277   void Turn_Right()
278   {
279     // set motor1 direction to backward
280     digitalWrite(LN_in1, LOW);
281     digitalWrite(LN_in2, HIGH);
282
283     // set motor2 direction to forward
284     digitalWrite(LN_in3, HIGH);
```

1.6 Code Lines 228-284



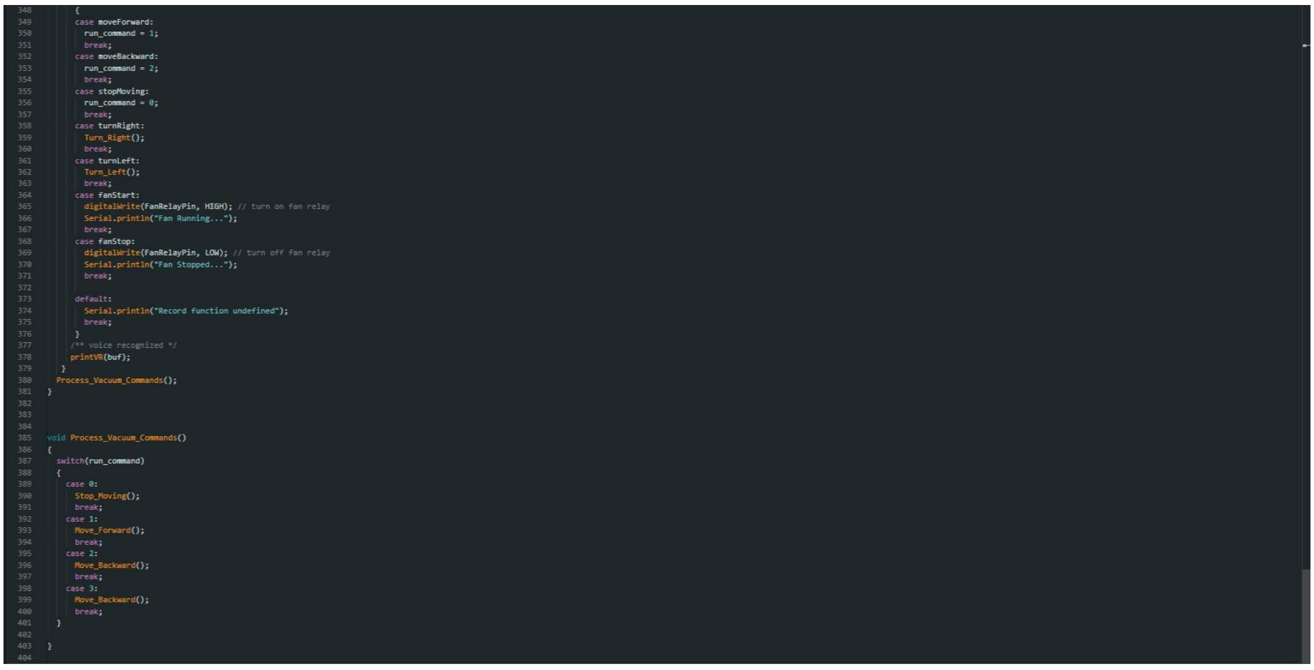
```
285     digitalWrite(LN_in4, LOW);
286
287     // enable motors 1 & 2
288     analogWrite(LN_en1, running_speed);
289     analogWrite(LN_en2, running_speed);
290
291     Serial.println("Turning Right...");
292     delay(running_time*2);
293
294     // stop motors 1 & 2
295     analogWrite(LN_en1, 0);
296     analogWrite(LN_en2, 0);
297     delay(running_pause);
298   }
299
300   void Turn_Left()
301   {
302     // set motor1 direction to forward
303     digitalWrite(LN_in1, HIGH);
304     digitalWrite(LN_in2, LOW);
305
306     // set motor2 direction to backward
307     digitalWrite(LN_in3, LOW);
308     digitalWrite(LN_in4, HIGH);
309
310     // enable motors 1 & 2
311     analogWrite(LN_en1, running_speed);
312     analogWrite(LN_en2, running_speed);
313
314     Serial.println("Turning Left...");
315     delay(running_time*2);
316
317     // stop motors 1 & 2
318     analogWrite(LN_en1, 0);
319     analogWrite(LN_en2, 0);
320     delay(running_pause);
321   }
322
323   /** END VARIABLES AND FUNCTIONS FOR VACUUM **/
324   /*********************************************/
325
326
327   void setup()
328   {
329     // put your setup code here, to run once:
330     Serial.begin(115200);
331     Serial.println("Setup.....");
332     delay(3000);  // allow other devices to start.
333     setup_HCSR04();
334     setup_vrm();
335     setup_LN();
336     setup_Vacuum();
337
338   }
339
340   void loop()
341   {
```

1.7 Code Lines 285-341



```
341  {
342    // put your main code here, to run repeatedly:
343    int ret;
344    ret = myVR.recognize(buf, 50);
345    if(ret>0)
346    {
347      switch(buf[1])
348      {
349        case moveForward:
350          run_command = 1;
351          break;
352        case moveBackward:
353          run_command = 2;
354          break;
355        case stopMoving:
356          run_command = 0;
357          break;
358        case turnRight:
359          Turn_Right();
360          break;
361        case turnLeft:
362          Turn_Left();
363          break;
364        case fanStart:
365          digitalWrite(FanRelayPin, HIGH); // turn on fan relay
366          Serial.println("Fan Running...");
367          break;
368        case fanStop:
369          digitalWrite(FanRelayPin, LOW); // turn off fan relay
370          Serial.println("Fan Stopped...");
371          break;
372
373        default:
374          Serial.println("Record function undefined");
375          break;
376      }
377      /** voice recognized */
378      printVR(buf);
379    }
380    Process_Vacuum_Commands();
381  }
382
383
384
385  void Process_Vacuum_Commands()
386  {
387    switch(run_command)
388    {
389      case 0:
390        Stop_Moving();
391        break;
392      case 1:
393        Move_Forward();
394        break;
395      case 2:
396        Move_Backward();
397        break;
```

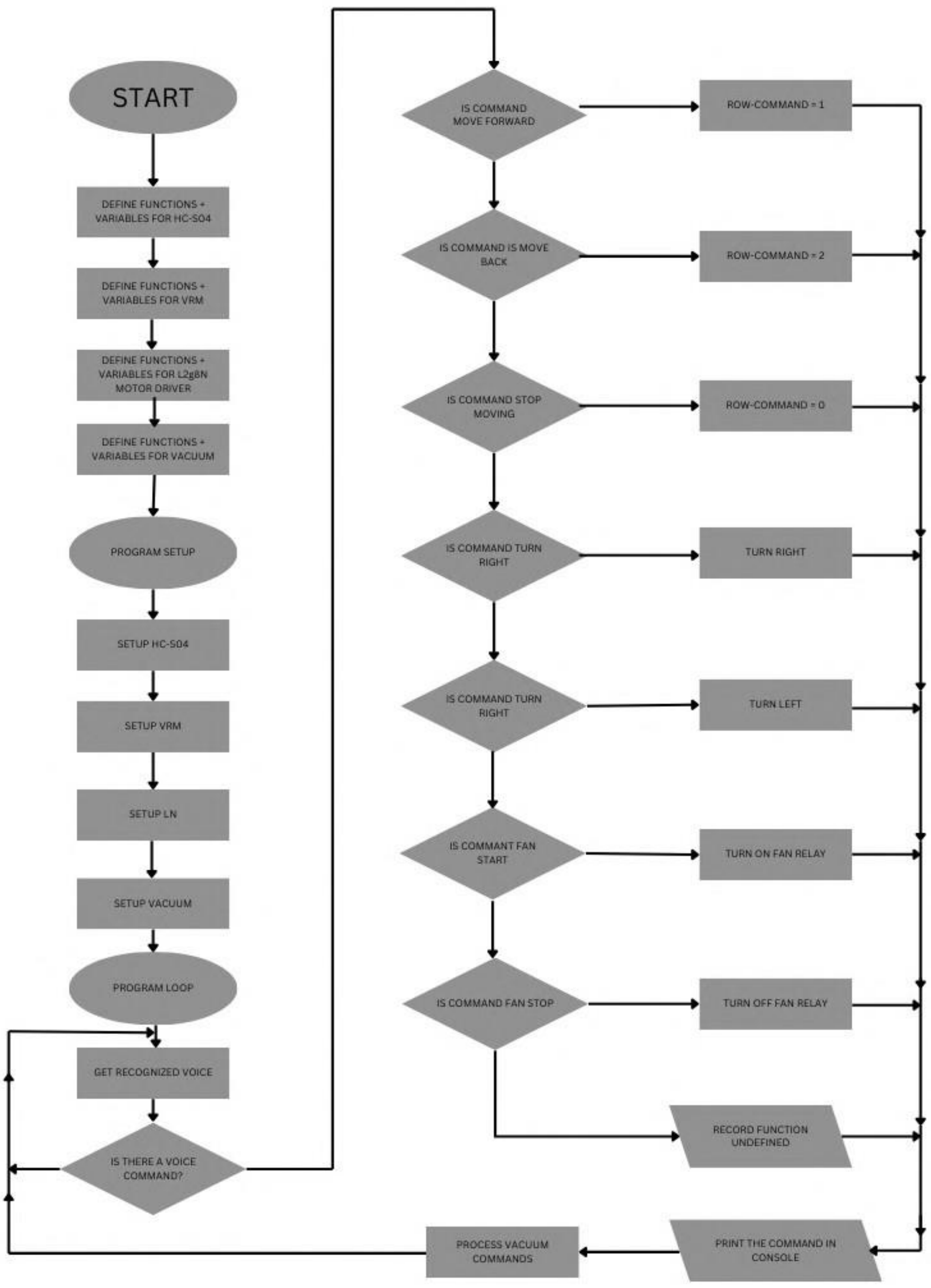
1.8 Code Lines 342-397



```
348      {
349        case moveForward:
350          run_command = 1;
351          break;
352        case moveBackward:
353          run_command = 2;
354          break;
355        case stopMoving:
356          run_command = 0;
357          break;
358        case turnRight:
359          Turn_Right();
360          break;
361        case turnLeft:
362          Turn_Left();
363          break;
364        case fanStart:
365          digitalWrite(FanRelayPin, HIGH); // turn on fan relay
366          Serial.println("Fan Running...");
367          break;
368        case fanStop:
369          digitalWrite(FanRelayPin, LOW); // turn off fan relay
370          Serial.println("Fan Stopped...");
371          break;
372
373        default:
374          Serial.println("Record function undefined");
375          break;
376      }
377      /** voice recognized */
378      printVR(buf);
379    }
380    Process_Vacuum_Commands();
381  }
382
383
384
385  void Process_Vacuum_Commands()
386  {
387    switch(run_command)
388    {
389      case 0:
390        Stop_Moving();
391        break;
392      case 1:
393        Move_Forward();
394        break;
395      case 2:
396        Move_Backward();
397        break;
398      case 3:
399        Move_Backward();
400        break;
401    }
402
403  }
404
```

1.9 Code Lines Number 348-404 (End)

Figure 2: Flowchart representation of the device process



7. FINDINGS

This part contains the results and interpretations of the data collected during the testing procedure in connection to the research topics.

7.1. Accuracy of the voice-commanded automatic vacuum cleaner in detecting obstacles in terms of sensor responsiveness

Table 1: Detection of Obstacles of the Vacuum Cleaner

Trial	**1**	**2**	**3**
Obstacle	Wall	Table Leg	Chair Leg
Test 1			
Test 2			

Test 3			
Test 4			
Accuracy	100%	75%	100%

Table 1 shows the accuracy of the voice-commanded automatic vacuum cleaner in detecting obstacles in terms of sensor responsiveness. The accuracy of the obstacle detection was determined by the receptiveness of the ultrasonic module. A light signal will appear before the command while turning off after the command. For accurate findings, four tests were undertaken with three distinct obstacles to overcome: a wall, a table leg, and a chair. The success percentage was calculated by dividing the total number of successful attempts by the total number of attempts and multiplying by 100. For the first obstacle, the wall, the voice-commanded automatic vacuum cleaner correctly recognized the obstacle in all four of the tests, indicating that the sensor response was 100% responsive to the wall. For the second obstacle, the table leg, the voice-commanded automatic vacuum cleaner correctly recognized the obstacle for Test 1, Test 2, and Test 3, but not Test 4, indicating that the sensor response was 75% responsive to the table leg. For the third obstacle, the chair leg, the voice-commanded automatic vacuum cleaner correctly recognized the obstacle in all four of the tests, indicating that the sensor response was 100% responsive to the wall.

As per the test findings, the sensor responsiveness in terms of obstacle detection achieved a 100% success rate on both the wall and chair leg obstacles, as well as a 75% success rate on the table leg obstacle. The results demonstrate that the voice-commanded automatic vacuum cleaner's sensor response was accurate in identifying obstacles. Furthermore, results from an identical study revealed that their sensor reactivity fluctuated somewhat less in percentage, with an obstacle detection rate of 81.4%. The study's findings indicate that a minimum sensor responsiveness accuracy of 81.4% is crucial for ensuring the effectiveness of a vacuum cleaner in navigating through dynamic environments (Lee et al., 2016).

7.2. Maximum detectable range of the voice module in response to a 60-decibel command at the following distances in meters

Table 2: Maximum Detectable Range in Meters

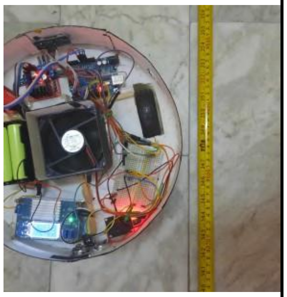
Trial	**1**	**2**	**3**
Photos			
Response	Response Observed	Response Observed	Response Observed
Range (in meters)	5 meters	7.5 meters	10 meters

Table 2 shows the three different trials in evaluating the Maximum Detectable Range in meters using a Tape Measure. In the first trial, the voice-commanded automatic vacuum was able to successfully detect a command at a range of 5 meters. In the second trial, the voice-commanded automatic vacuum was able to successfully detect a command at a range of 7.5 meters. In the last trial, the voice-commanded automatic vacuum was able to successfully detect a command at a range of 10 meters.

Interpreting the data, the average distance it took for the voice-commanded automatic vacuum cleaner to successfully recognize the command is 7.5 meters. It is shown in the study that the device can easily receive the command and perform it given a specific distance. In another study, their implemented module is more dependable and adaptable in controlling any load, with a wireless control coverage area of 10 meters. As a result, this initiative has the potential to revolutionize real-time voice-controlled home automation. (Nlerum, Amannah, 2020).

7.3. Latency between the audible commands and the action of the vacuum cleaner in terms of seconds

Table 3: Latency of Audible Commands and Action of the Vacuum Cleaner

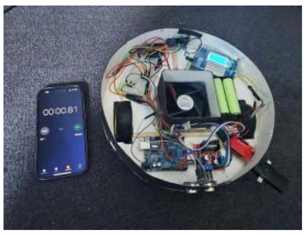
Trial	**1**	**2**	**3**
Photos			
Time (in seconds)	0.91 seconds	0.81 seconds	0.78 seconds

Table 3 shows the three different trials in assessing the latency of audible commands of the vacuum cleaner using a stopwatch. In the first trial, the voice module was able to detect the voice command and activate the vacuum cleaner with a delay of 0.91 seconds. In the second trial, the voice module was able to detect the voice command and activate the vacuum cleaner with a delay of 0.81 seconds. In the third trial, the voice module was able to detect the voice command and activate the vacuum cleaner with a delay of 0.78 seconds. It was inferred by the researchers that human reaction time was a factor in the variability of time for the trials.

Assessing the data, the average time it took for the voice module to recognize a voice command and activate the vacuum cleaner was 0.83 seconds. The studies demonstrate that the speech module can be detected quickly and with little latency. In another experiment, integrating the speech module into a physical prototype resulted in a significant improvement in voice recognition with no latency (Basyal, 2018). Furthermore, low inference latency is vital for providing a user-friendly experience, as delayed feedback has a negative influence on usability (Goyal, Garera, 2023).

8. DISCUSSIONS

A vacuum cleaner is designed to make a simple and more efficient user experience to do specific tasks. A vacuum cleaner gathers various forms of trash, dust, and allergens by airflow, which is often combined with brushing motion. It effectively collects and eliminates these substances from the air, preventing them from being re-circulated back into the living environment (Subramaniam et al., 2022). Through Arduino Uno R3, an essential component to many advancements as a user-friendly, open-source electronics platform, this study aimed to create a voice-commanded automatic vacuum cleaner out of Arduino interface and voice module that is cost-effective and environmentally sustainable for ease of use, preventing indoor air pollution and improving housekeeping. This research sought to assess the effectiveness of the voice-commanded vacuum cleaner in terms of suction power and obstacle detection. The effectiveness of the voice module was evaluated through the command's maximum detectable range and latency. As a result, this research is claimed to be effective as the product can easily detect obstacles, perform the command given a specific distance, and quickly perform the command. The study in Table 1 shows that detecting walls and chair legs was successful with an accuracy of 100%, and for table legs, it had an accuracy of 75%. Therefore, we can trust the voice-command automated vacuum cleaner so that it can easily detect items in its way. To reach a perfect device, it should have a stronger sensor and a brighter surrounding. The results in Table 2 show that the device can successfully detect the command given a specific distance of 5 meters, 7.5 meters, and 10 meters. As a result, the voice-command automatic vacuum cleaner can easily receive the command and perform it at varied distances. Providing a better voice module would greatly allow the device to perform commands at a farther distance. Lastly, the study shows in Table 3 that it can swiftly operate the said command with a delay of 0.91, 0.81, and 0.78 seconds. Therefore, the voice-command automatic vacuum cleaner can quickly respond to the given command while maintaining a stronger and consistent suction power. As stated earlier, getting a hand on a higher-quality voice module would greatly benefit not only the latency but also the command being performed at a distance. Moreover, usability is impacted by delayed feedback, therefore low inference latency is essential for a user-friendly experience (Goyal, Garera, 2023).

9. CONCLUSIONS

Based on the results, the Voice Commanded Automatic Vacuum Cleaner can detect obstacles and respond to voice commands at various distances at a high success rate. Additionally, the Voice Commanded Automatic Vacuum Cleaner responds and operates at very low latency rates after a voice command, although the vacuum cleaner may have little to no recognition of voices that were not used to calibrate the voice module. This research study can assist homes around the world in addressing the detrimental problem of indoor air pollution.

10. RECOMMENDATIONS

Students and future researchers are advised to make use of higher-grade material to improve longevity, durability, and weight considerations for optimal performance. The researchers advise the communities to add more essential features and a better dust compartment storage system to enhance the device's functionality and capabilities. Due to the automatic and voice commanded nature of the device with the voice module and Arduino Uno R3 as the main components, the device presents the heightened convenience in effectively mitigating the issue of indoor air pollution. Furthermore, future researchers may use this study as a guide in innovating projects that have similarities in functions or components. Future researchers may add and integrate a better vent system for the intake and exhaust system for the vacuum, as well as useful cleaning additions like spinning brushes. The researchers suggest utilizing a better quality microphone and voice module to ensure that more voice commands can be stored at a larger database, and voices could be more easily recognized consistently and accurately. The researchers also suggest for future researchers to use a more effective exhaust and intake fan system that has minimal noise and high suction power for better functionality and efficacy in cleaning.

REFERENCES

Abusin, S., Al-Thani, N., AL-Emadi, N., & Petcu, C. (2022). Health impact of indoor air pollution in the gulf region: a review. *World Journal of Advanced Research and Reviews*, 16(02), 49-57. https://doi.org/10.30574/wjarr.2022.16.2.1138

Adjei, H. A., Oduro-Gyimah, F. K., Shunhua, T., Agordzo, G. K., & Musariri, M. (2020). Developing a Bluetooth based tracking system for tracking devices using Arduino. 2020 5th International Conference on Computing, Communication and Security (ICCCS), 1–5. https://doi.org/10.1109/iccs49678.2020.9276884

Aspers, P., & Corte, U. (2019). What is qualitative in qualitative research? *Qual Sociol* 42, 139-160 https://doi.org/10.1007/s11133-019-9413-7

Badamasi, Y. (2014). The working principle of an arduino. *International Conference on Electronics, Computer and Computation (ICECCO)*, 1-4. https://doi.org/10.1109/ICECCO.2014.6997578

Basyal, L. (2023). *Voice recognition robot with real-time surveillance and automation*. arXiv.org. https://doi.org/10.48550/arXiv.2312.04072

Coghlan, D., Brydon-Miller, M. (2014). The sage encyclopedia of action research. *SAGE Publications Ltd,* 1-2. https://doi.org/10.4135/9781446294406

Eren, A. & Doğan, H. (2022). Design and implementation of a cost effective vacuum cleaner robot. *Turkish Journal of Engineering,* 6 (2), 166-177.

https://doi.org/10.31127/tuje.830282

Goyal, A., & Garera, N. (2023). Building accurate low latency ASR for streaming voice search in e-commerce. *ACL Anthology*. https://doi.org/10.18653/v1/2023.acl-industry.26

Lee, T.-J., Yi, D.-H., & Cho, D.-I. (2016, March 1). *A monocular vision sensor-based obstacle detection algorithm for Autonomous Robots*. MDPI. https://doi.org/10.3390/s16030311

Neira, M., & Ramanathan, V. (2020). Climate change, air Pollution, and the environment: the health argument. *Springer, Cham,* 93–103. https://doi.org/10.1007/978-3-030-31125-4_8

Nicholls, L., & Strengers, Y. (2019). Robotic vacuum cleaners save energy? raising cleanliness conventions and energy demand in australian households with smart home technologies. *Elsevier,* 50, 73-81. https://doi.org/10.1016/j.erss.2018.11.019

Vanus, J., Smolon, M., Martinek, R., Koziorek, J., Zidek, J., & Bilik, P. (2015). Testing of the voice communication in smart home care. *SpringerOpen,* 15, https://doi.org/10.1186/s13673-015-0035-0

Wu, D., Deng, B., & Zhuang, X. (2022). Design of speech recognition robot. *Digital library,* 221-227. https://dl.acm.org/doi/10.1145/3532213.3532246

Yılmaz, E., Özyer, S. (2019). Remote and Autonomous Controlled Robotic Car based on Arduino with Real Time Obstacle Detection and Avoidance. Universal Journal of Engineering Science, 7(1), 1 - 7. DOI: 10.13189/ujes.2019.070101