ROBOT VACUUM CLEANER WITH ULTRASONIC SENSOR AND BLUETOOTH BASED ON ARDUINO UNO

[1]Fiki Mayasari, [2]Hironimus Leong
[1,2]Program Studi Teknik Informatika Fakultas Ilmu Komputer,
Universitas Katolik Soegijapranata
[2]marlon.leong@unika.ac.id

ABSTRACT

The development of technology today has increased in various fields of human life. The combination of hardware and software can produce complex systems that certainly save production and maintenance costs. In the world of electronics, the emergence of increasingly sophisticated sensors can help humans create a tool that can be used in everyday life for heavy and light work so as to facilitate human work. A robot vacuum cleaner equipped with a Bluetooth controller and supported by an Arduino-based ultrasonic sensor as a rangefinder is a solution for cleaning hard-to-reach parts of the room, saving labor and time. The robot vacuum cleaner can be controlled in two modes, manual and automatic modes, which are identified by a smartphone connected to the robot vacuum cleaner. The robot vacuum cleaner can move forward, backward, left and right. The test results show that the robot vacuum cleaner works well.

Keywords: vacuum cleaner 1, robot 2, Arduino 3

INTRODUCTION

1.1. Background

The development of technology today has increased in various aspects of human life. The combination of hardware and software can produce complex systems that will definitely save production and maintenance costs. The emergence of increasingly complex sensors in the electronic world can help humans create a tool that can be used in everyday life, both heavy and light work, and make human work easier. This development is supported by increasingly sophisticated hardware and software.

Robots are very much needed in today's life, where all human work is required to be more efficient and effective. Especially for housewives, where activities at home are very timeconsuming and the work done is never finished. Arduino-based floor cleaning robot is a robot that can clean the floor efficiently so as to facilitate human work[1], where the robot will move along the line automatically to transport garbage on the floor. Designing a robot that is able to move left and right and can move heavy items using a microcontroller that has been connected to an android smartphone connected via Bluetooth to drive the system on the robot[2]. Robot vacuum cleaner made automatically designed and implemented with Maze and PID methods to map the room being cleaned [Dian Tresnawan, Meidi, 2015]. The operation of this robot vacuum cleaner

starts with mapping the area to be cleaned. In his research, the area to be cleaned is 200cm x 200cm and is divided into 8 columns and 8 rows, each column is divided into 8 indices, and the row is divided into 8 indices and entered into the main robot movement program. In addition, the robot moves according to the planned mapping and returns to its original position quickly after emptying through all rows and columns. And according to the experiment, the robot can run according to the rules with 100% success rate.

1.2. Problem Formulation

Based on the explanation of the background, the following problem formulation:

1. How does the robot vacuum cleaner work in the process of vacuuming dust or small particles around the room?
2. How to make a robot vacuum cleaner using Arduino Uno?
3. How does the ultrasonic sensor work on the robot vacuum cleaner system to determine whether there are obstacles around the room?

1.3 Scope

Making a robot vacuum cleaner using ultrasonic sensors, Bluetooth, IR sensors based on Arduino uno. The ultrasonic sensor works as a monitor for the presence or absence of obstacles when the robot vacuum cleaner works to clean the floor, and the movement can be adjusted via a Bluetooth connection that is connected to a smartphone. As power, the robot vacuum cleaner only uses DC current and is also equipped with a push button to turn on the robot vacuum cleaner. The robot vacuum cleaner here can only clean dust and small particles on the floor.

1.4 Objective

The purpose of making this robot vacuum cleaner is to simplify housework, especially in cleaning the floor, to save time. The robot uses ultrasonic sensors and Bluetooth. The ultrasonic sensor works to determine whether or not there are obstacles that exist when the robot cleans the floor. And the movement is directed by Bluetooth which has been connected to a programmed smartphone.

LITERATURE STUDY

This chapter reviews the literature on robots and vacuum cleaners, the theory of support systems, both hardware and software, and several other supporting theories.

A robot is a set of mechanical devices that can perform physical tasks either under human supervision and control or using a predetermined program (artificial intelligence). Robots are usually used for heavy, dangerous, repetitive, and dirty work. In general, most industrial robots are used in manufacturing. Other uses of robots are toxic waste cleanup, underwater and space exploration, mining, search and rescue, and mine exploration. Recently, robots are conquering the consumer market in the entertainment sector and as household helpers such as vacuum cleaners and lawn mowers. A vacuum cleaner is a device that uses an air pump to create a vacuum that sucks up dust and dirt, usually from the floor. Most carpeted homes in developing countries have a vacuum cleaner to clean them. Dirt is collected and removed with filters or cyclone systems. Robot vacuum cleaners are one example of a very useful application of robotics in the home. These robots are designed to clean floors automatically and can be set to perform such tasks on a scheduled or on-demand basis. Arduino Uno is one of the very popular and easy-to-use microcontroller boards to control robot vacuum cleaners.

Vacuum cleaners are currently manually operated and operated with human assistance, so users cannot perform other tasks. Based on these problems, researchers are encouraged to develop a Vacuum Cleaner Robot that can move randomly and also navigate through existing obstacles. The behaviors used are Wandering, Obstacle Avoidance, and stop. Vacuum cleaner robot automatically moves randomly and can avoid obstacles by using PID method.

The PID controller consists of three components, namely proportional components, integral components, and derivative components. The three components complement each other, so that weaknesses in one component can be corrected by other components.

The robot performs each action based on the input value received from the ultrasonic sensor. Input is an input parameter received consisting of ultrasonic sensors and input is carried out by the process section using a microcontroller. Information from the distance sensor is the distance between the distance sensor and the obstacle, and also as input information for the movement of the automatic robot vacuum cleaner.

Through testing the robot vacuum cleaner can move automatically and move randomly and can avoid existing obstacles with the PID method. The PID method on the robot vacuum cleaner aims to make the robot move straight and stable. However, the robot's behavior cannot run randomly because the robot does not use a mapping system and for its cleanliness in cleaning existing dirt, the robot can clean existing dirt but is not perfect in the process of vacuuming the dirt [3].

One of the robot mechanisms uses an automatic control system based on Arduino Mega 2560Pro. The control system is connected to a DC motor to control the right and left rotation

direction of the two side brushes, while the sweeping rod will automatically move by itself after the robot is turned on with the bottom switch. Knowing the load capacity and torque of the DC motor sweeping system on the floor cleaning robot. In testing the sweeping system of the floor cleaning robot, measurements and calculations of dust sweeping performance were made at 1000 rpm and 1200 rpm. Based on the results of sweeping performance testing, it is known that a sweeper with a speed of 1200 rpm is better used for sweeping than a speed of 1000 rpm [4].

Automatic floor cleaning robot / tool to make it easier for housewives to clean the floor. in making a robot, of course, cannot be separated from the electrical system. Therefore, to get the electrical system as needed, it is necessary to do the design. The design is the manufacture of regulators, DC motor installation, motor driver placement, ultrasonic sensor installation, button settings and appearance on the LCD. Arduino UNO is the main basis, so an Arduino UNO module is needed as the basic brain. This hardware design is carried out to realize the creation of an automatic floor cleaner that is simple and can be operated using an ultrasonic sensor. From the results of the design, the robot can move by passing through existing obstacles but when turning it is less than perfect because the torque of the DC motor used is not strong enough to move. For the rotation of the brush also causes the robot to be less stable in moving [5].

RVC was first introduced in research (Iwuoha Chigozie Williams 2019) as a hoover or sweeper, a device that uses a centrifugal fan that creates a partial vacuum to suck up dust and dirt, usually from floors, and other surfaces such as upholstery and curtains. The act of sucking up dust and dirt is seen as suction. This suction determines how well the RVC cleans the room. First, the centrifugal fan is mathematically modeled through a DC motor and then simulated through MATLAB SIMULINK to show that Newton's second law of rotation can be used to configure the fan speed. Secondly, the air flow rate is modeled to be constant for perfect suction. This research is an improvement on modeling RVC suction with mathematical models and MATLAB SIMULINK models, but there is still room for further improvement [6].

In today's technology, especially in the field of robotics and communication technology is growing very rapidly, a lot of robots and communication tools are created with advanced technology. The implementation of these two elements is to create a robot vacuum cleaner using an omni wheel that can move in any direction without making complicated movements and is controlled using an accelerometer sensor android smartphone sensor via Bluetooth RC controller application found on the Playstore so that robotic moves following the movement of the next smartphone Connecting media between smartphones and robots using bluetooth HC - 05. With accelerometer tilt sensitivity on a robot of 5 ° reliable enough to maneuver to remove dust or dirt with a small back and forth movement method. Robot omni wheel vacuum cleaner can move in various directions and can move efficiently [7].

When designing this prototype, a simple robot vacuum cleaner is made which is implemented through the data collection method by observing the vacuum cleaner system and interviewing vacuum cleaner users, as well as searching for Arduino and Android literature.

After that, the System Analysis and Program Design methods can be connected to the Arduino and Android microcontrollers. After the two processes above, the prototyping tool process can be carried out. The main feature of this prototype is that it can vacuum or other materials and can be controlled via Bluetooth from the Android application program installed on the cellphone. After the prototype method, the black box text method can be implemented, where the results of testing the performance of the prototype starting from the robot's movement speed and the required energy consumption, as well as the results of hardware specifications and functions can be obtained that can be provided by the software installed on the Android application, as well as the function of the cleaning tool and the benefits of the functions that can be provided to the user of the tool [8].

By using a microcontroller with an Arduino platform, a dust cleaning robot can be made at a lower price and with reliable functions. Arduino can be used to create a robot that will automatically remove dust from where the robot will not hit the wall, and the robot is also equipped with two brooms at the bottom. The purpose of the brushes is to remove dust from the floor and the dust is picked up by a suction fan mounted below the robot. In addition, the robot can also be controlled manually with an Android smartphone, thus allowing the user to freely direct the robot to places that the robot may not have traveled [9].

Design and implementation of a mobile robot dust cleaner based on AT89S52 microcontroller that uses four ultrasonic sensors and EMS 1A dual H-bridge motor driver. The results of the research are in the form of a mobile robot system capable of cleaning dust per column by cleaning from bottom to top to top and turning to the right with the condition that the right wheel is stationary and the left wheel rotates 18 ° and from the top, the robot goes down and turns towards the left while the right wheel rotates 18 ° and the left wheel is stationary. The conclusion of the research is the success rate of ultrasound measurement with 88%. The robot can measure a minimum distance of two cm to 320 cm and was tested in a room of 1 m^2 [10].

Based on some literature searches, researchers want to design a robot vacuum cleaner with ultrasonic sensors and Bluetooth as a means of connecting with a smartphone as a vacuum cleaner control tool and for the brain of a vacuum cleaner robot with Arduino Uno. The production of the planned robot vacuum cleaner should facilitate room cleaning, especially in hard-to-reach places, which can save time and energy. The robot vacuum cleaner can move automatically without human control and can move according to instructions.

RESEARCH METHODOLOGY

3.1. Research Methodology

The steps taken to create a vacuum cleaner design as a tool in cleaning the room that makes it easier for users to clean the side of the room that is difficult to reach if done manually using a broom such as in the corner of the room, under the table, and under the bed. In addition to making it easier, the robot vacuum cleaner can also save time and energy.

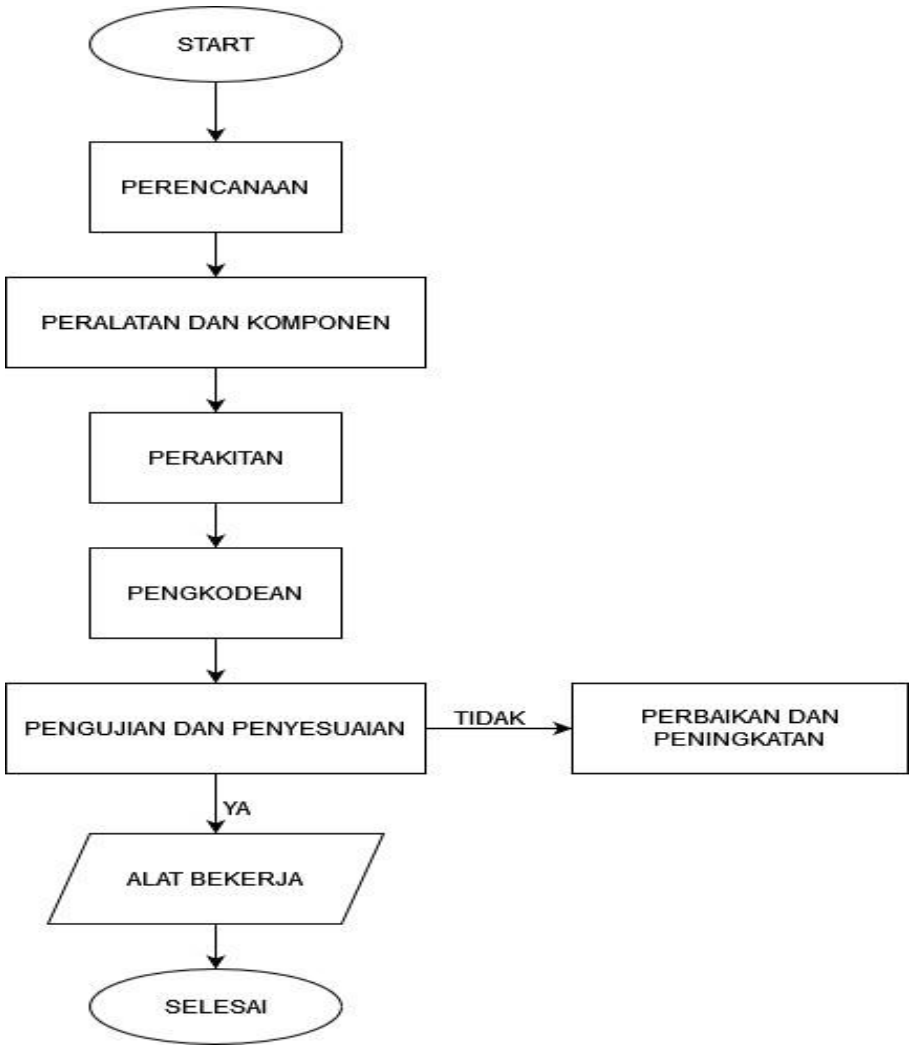

Figure 3. 1 Research Methodology Explanation

of the steps in the diagram:

1. Start

2. Planning: This stage involves determining the goals and specifications of the robot vacuum cleaner and creating a sketch or design.

3. Equipment and components: Equipment and components needed to assemble the robot vacuum cleaner such as Arduino Uno, DC motor, ultrasonic sensor, IR sensor, battery, drive wheel, HC-05 bluetooth.

4. Assembly: Assemble the robot vacuum cleaner frame, install the drive motor on the wheel, install the proximity sensor and infrared sensor, and install the cleaning device.

5. Coding: Writing and uploading program code to the Arduino Uno to manage robot navigation, obstacle detection, and the use of motors and cleaning devices.

6. Testing and adjustment: testing the robot vacuum cleaner to ensure all components and functions are working properly. Testing robot navigation, motor performance, sensors, cleaning devices.

7. Repair and upgrade: repair or upgrade the robot vacuum cleaner based on the test results.

8. The tool works: the robot works well

9. Completed

3.2. Dataset Collection

At this stage, data collection is carried out which is used to design a robot vacuum cleaner as a tool to assist housewives in cleaning each room and to save time and energy. Data required before designing:

1. Identification of complaints and needs.

From the results of designing a robot vacuum cleaner, it is hoped that it can help and meet the needs of cleaning a dirty room efficiently and save time.

2. Technical specs that are considered in designing a robot vacuum cleaner, such as the ability to suck dust, the sensors needed, battery power, and the design of the robot itself. In the mechanical design itself, it can be reviewed from the wheels, brushes, vacuum cleaners, and dust containers as a container for dirt that has been successfully sucked.

3. Review of the navigation system. Review of the obstacle recognition system in the room with the help of ultrasonic sensors and IR sensors.

3.3. Design

3.3.1. Circuit Design

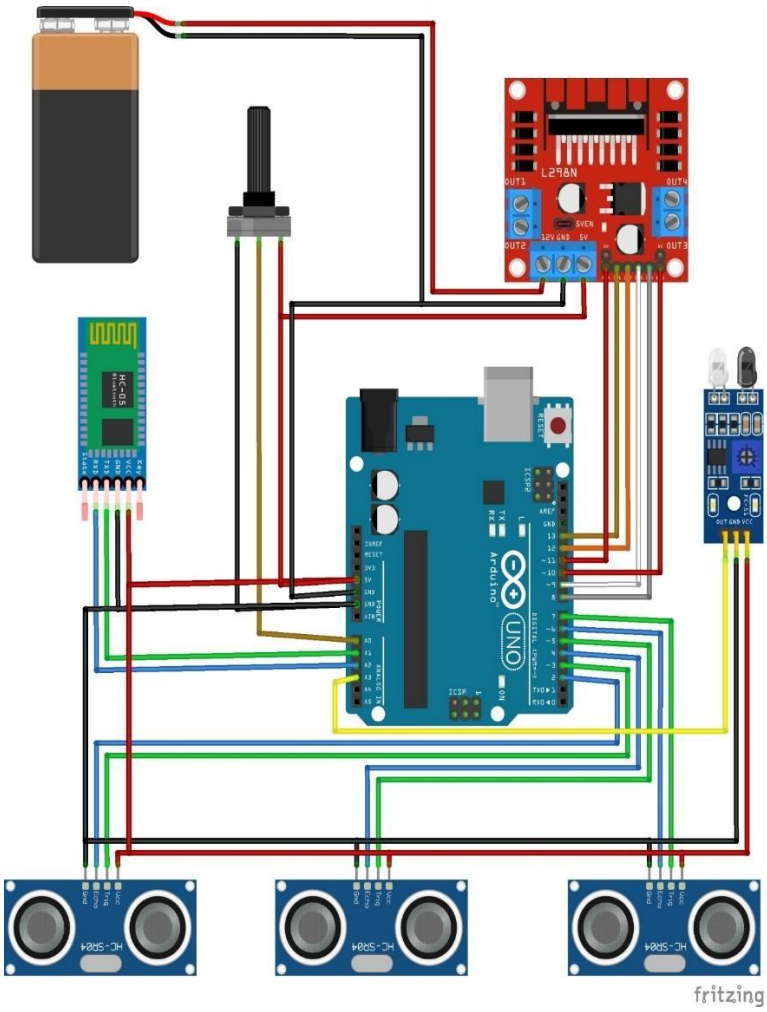

Figure 3. 2 Sensor Part Connection

Circuit design in the sensor connection section and Arduino Uno in making a robot vacuum cleaner. There are three ultrasonic sensors, IR sensor, Bluetooth module, motor driver, battery, motor driver, and potentiometer. All of these components are assembled on the Arduino Uno. Arduino will work as a brain that receives input from sensors, processes, and provides output on the movement of the robot vacuum cleaner.

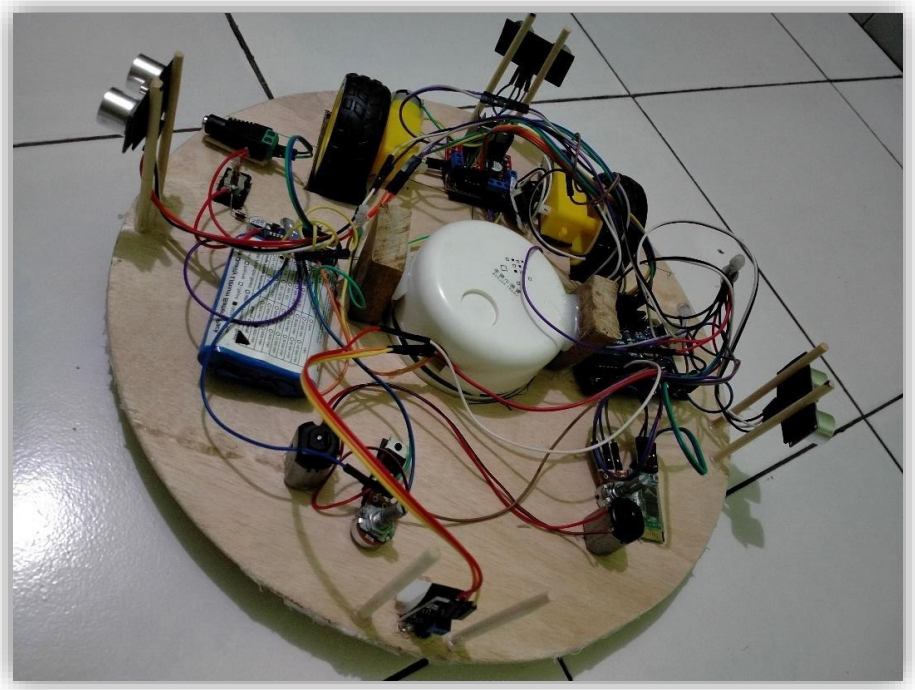

Figure 3. 3 Robot Vacuum Cleaner Circuit Design

Figure 3.3 shows the implementation view of the robot circuit after assembly. The placement of three ultrasonic sensors on the left, right, and back sides. The IR sensor is placed at the front, and the center is a portable vacuum cleaner. The motor driver is placed at the back adjacent to the rear ultrasonic sensor and wheels. Bluetooth on the right side. The division of the layout of ultrasonic sensors and IR sensors in order to read obstacles from various positions, thus minimizing collisions during automatic mode and assisting movement in manual mode.

3.3.2. Mechanical Design Diagram

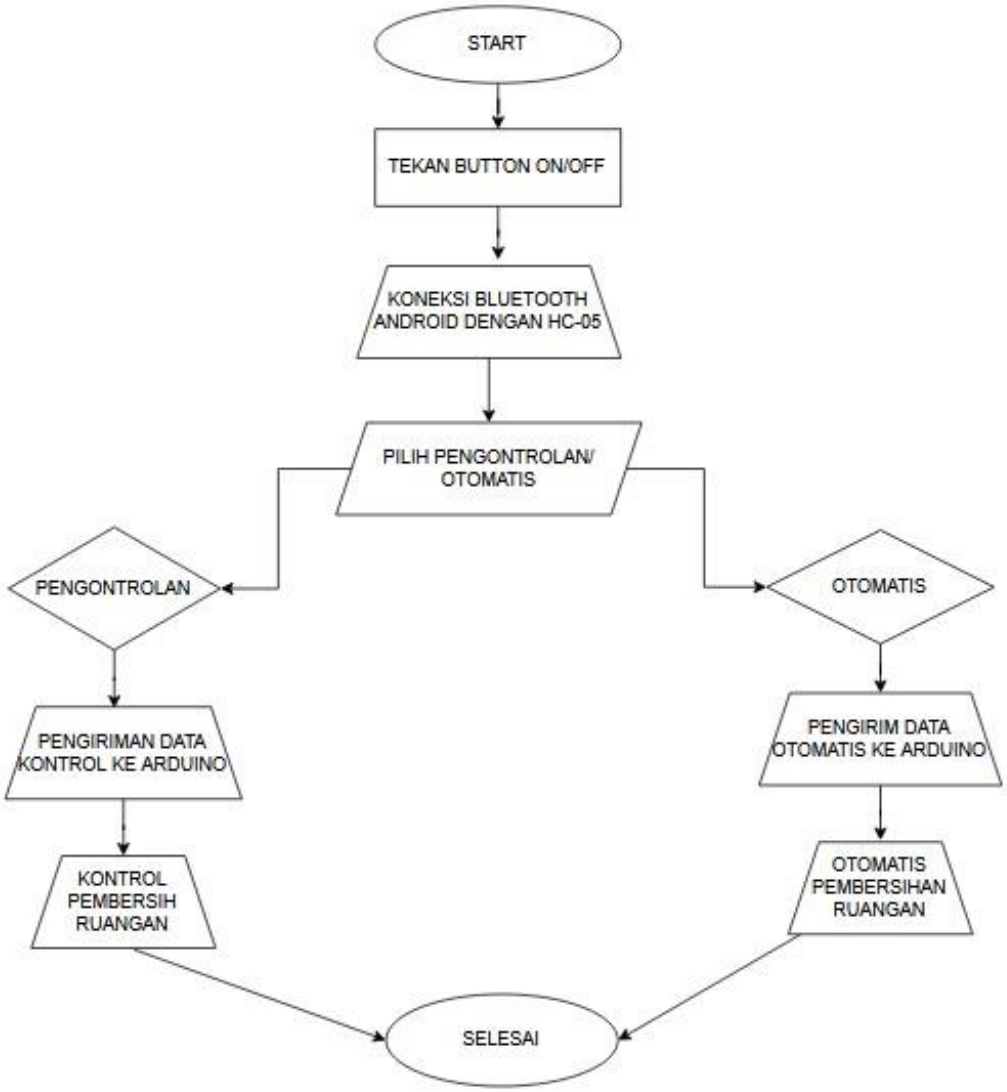

Figure 3. 4 Mechanical Design Diagram

Figure 3.4 explains the mechanism of how the robot vacuum cleaner works. First press the ON/OFF button at the bottom of the robot, then activate Bluetooth on Android and connect with the robot via the Bluetooth RC Controller application, after successfully connecting select the mode to run it. If you choose automatic mode, press the car image. The robot will work automatically. For manual mode, just press the available navigation and the robot can move according to the direction.

IMPLEMENTATION AND RESULTS

4.1. Experiment Setup

This research uses several components to create a robot vacuum cleaner design. The components used to design the robot vacuum cleaner are as follows :

1. Arduino Uno : as an electronic control device, whose job is to read inputs with a microcontroller, which then converts them into outputs.
2. Ultrasonic Sensor : a sensor that measures the distance between the sensor and an obstacle. Navigation devices use sensors to avoid obstacles and determine the distance to obstacles.
3. HC-05 Bluetooth Module : Used for wireless communication and remote control of the robot using a connected smartphone.
4. Infrared sensor : The infrared sensor detects obstacles when the infrared light is blocked by an obstacle.
5. Remote control battery : The source of power to drive the robot Starter cable: a conductor used to connect electrical circuits.
6. Jumper Cable : a conductor that serves to connect electrical circuits.Castor Wheel: Castor wheels are used to provide movement and mobility capabilities to the robot vacuum cleaner.
7. DC Motor : The DC motor on the robot vacuum cleaner provides enough traction to produce a strong suction force.
8. Portable Vacuum Cleaner : Vacuum cleaner and small dirt
9. IRF Z44 Motor Speed Controller : IRF Z44 Motor Speed Controller is used to control the rotation speed of the DC motor. By adjusting the voltage applied to the motor, users can speed up or slow down the motor rotation speed as needed.
10. L298 Motor Driver : Controls the direction and speed of rotation of the DC motor.
11. Smartphone : As a command communication sending device to Bluetooth on the robot vacuum cleaner.

4.2. Implementation

Defines a constant for the ultrasonic sensor. Declares variables to store distance measurements from the ultrasonic sensor and defines the minimum avoidance distance.

```
1.      //====================Ultrasonic====================
2.      #include <NewPing.h>
3.      #define SONAR_NUM 3 // Number of sensors.
4.      #define MAX_DISTANCE 200 // Maximum distance (in cm) to ping.
```

```
5.       NewPing sonar[SONAR_NUM] = { // Sensor object array.
6.       NewPing(3, 2, MAX_DISTANCE), // Each sensor's trigger pin, echo pin,
         and max distance to ping.
7.       NewPing(5, 4, MAX_DISTANCE),
8.       NewPing(6, 7, MAX_DISTANCE)
9.       };
   1. long duration, distance, RightSensor, FrontSensor, LeftSensor;
      //variables to store distance data
10.      int avoidDistance = 15; //the minimum distance for the robot to
         avoid obstacles in front of the sensor
11.      int a=0; //additional variables for logic
12.      boolean automatic = false; //additional variable for logic
```

Line 3 defines the number of connected sensors, line 4 sets the maximum distance (in centimeters) to ping. Any distance beyond this value will be considered invalid, line 5 creates an array of NewPing objects to represent the sensors. In this case, there are three sensors, and each sensor is defined with a trigger pin, echo pin, and max distance to ping.

Here are the details of the sensor configuration:

- Sensor 1: The trigger pin is connected to pin 3, the echo pin is connected to pin 2, and the maximum distance to ping is set to `MAX_DISTANCE`.
- Sensor 2: The trigger pin is connected to pin 5, the echo pin is connected to pin 4, and the maximum distance for ping is set to `MAX_DISTANCE`.
- Sensor 3: The trigger pin is connected to pin 6, the echo pin is connected to pin 7, and the maximum distance to ping is set to `MAX_DISTANCE`.

Lines 10 to 12 define additional variables for storing distance data and controlling program logic.

Lines 13 to 19 define variables and pins for the IR sensor and set up SoftwareSerial communication for the Bluetooth module.

```
   #define irPin A3 //declare the input pin for the IR sensor
   14. //=====================Bluetooth=====================
   15. #include <SoftwareSerial.h> //to call the library
   16. #define rxPin 15 //declare pin for communication with bluetooth module
       17. #define txPin 16 / / pin declaration for communication with
       bluetooth module
18. SoftwareSerial mySerial = SoftwareSerial(rxPin, txPin); / / call the
    serial software function
19. char inputValue; //variable to store data from communication with the
    bluetooth module
```

Line 16 `#define rxPin 15` defines the constant `rxPin` and assigns the value `15`. This constant represents the pin used to receive data from the Bluetooth module. Line 17 `#define txPin 16` defines the constant `txPin` and assigns the value `16`. This constant represents the pin used to transmit data to the Bluetooth module.

Lines 20 to 28 define the pins for the DC motor and potentiometer to control the motor speed.

```
20. #define IN1 13 // declaration of IN1 pin
21. #define IN2 12/// declaration of IN2 pin
22. #define IN3 9 // IN3 pin declaration
23. #define IN4 8 // declaration of IN4 pin
24. #define ENA 11 // IN3 pin declaration
25. #define ENB 10 // declaration of IN4 pin
26. int speedMotor = 200; //variable to store the motor speed, taken from
    the result of the potentiometer setting
27. //=====================DC Motor=====================
28. #define potentiometer A0 // IN4 pin declaration
```

Line 24 `#define ENA 11` defines the constant `ENA` and assigns the value `11`. This constant represents the pin used to control the speed of the first motor connected to the H-bridge driver. Line 25 `#define ENB 10` defines the constant `ENB` and assigns it the value `10`. This constant represents the pin used to control the speed of the second motor connected to the Hbridge driver. Line 26 int speedMotor = 200` declares the variable `speedMotor` and initializes it with the value `200`. Line 28 `#define potentiometer A0` defines the constant `potentiometer` and assigns the value `A0`.

Lines 29 to 48 in the `setup()` function, initialize serial communication, set the pin mode, and configure the motor speed.

```
29. unsigned long previousMillis = 0; // will store last time LED was updated
30. const long interval = 1000; // interval at which to blink (milliseconds)
31. void setup() {
32. Serial.begin(9600); // start serial communication with the computer via
    baud rate 9600
33. // ir
34. pinMode(irPin, INPUT); //set the pin to INPUT
35. // Bluetooth
36. pinMode(rxPin, INPUT); / set the pin to INPUT
37. pinMode(txPin, OUTPUT); / / set the pin to OUTPUT
38. mySerial.begin(9600); / / start serial communication with bluetooth
    module via baud rate 9600
39. // DC motor
40. pinMode(IN1, OUTPUT); / / set the pin to OUTPUT 41. pinMode(IN2, OUTPUT);
    / / set the pin to OUTPUT
42. pinMode(IN3, OUTPUT); / / set the pin to OUTPUT
43. pinMode(IN4, OUTPUT); / set the pin to OUTPUT 44. pinMode(ENA,
    OUTPUT); / set the pin to OUTPUT
45.      pinMode(ENB, OUTPUT); / set the pin to OUTPUT
46.      analogWrite(ENA, speedMotor); / set the left motor speed based on
         the potentiometer setting
47.      analogWrite(ENB, speedMotor); / set the speed of the right motor
         based on the potentiometer setting
48.      }
```

Line 29 declares the unsigned long variable `previousMillis` and initializes it with the value `0`. This variable will be used to store the previous time the LED was updated or an action was performed. Line 30 declares the constant variable `interval` and assigns it the value `1000`. This constant represents the interval at which an action or task will be performed, in this case, blinking the LED. The interval is specified in milliseconds. Line 34 sets the pin assigned to the `irPin` variable as the input pin. Line 36 sets the pin assigned to the `rxPin` variable as an input

pin. Line 37 sets the pin assigned to the `txPin` variable as the output pin. Lines 46-47 set the speed.

Lines 49 to 69 of the `loop()` function, which runs iteratively, reads ultrasonic sensor data, controls the robot based on input from Bluetooth or manual control, and transmits sensor data via serial and Bluetooth.

```
49.      void loop() {
50.      unsigned long currentMillis = millis();
51.      if (currentMillis - previousMillis >= interval) {
52.      previousMillis = currentMillis;
53.      Serial.print(LeftSensor);
54.      Serial.print(" - ");
55.      Serial.print(FrontSensor);
56.      Serial.print(" - ");
57.      Serial.println(RightSensor);
58.      mySerial.print(LeftSensor);
59.      mySerial.print("  -  "); 60.  mySerial.print(FrontSensor);  61.
         mySerial.print(" - ");
mySerial.println(RightSensor);
62.      }
63.      speedMotor = map(analogRead(A0), 0, 1023, 100, 255); //determine
         motor speed based on potentiometer setting
64.      if(mySerial.available() > 0) { //receive data from bluetooth module
         inputValue = mySerial.read(); // read data and store data in
         variable if(inputValue == 'W'){ //when receiving 'W' command then
         automatic mode works automatic = true;
Serial.println("automatic");
}else if(inputValue == 'w'){ //if it receives the command 'w' then manual
mode works automatic = false; stopMotor();
Serial.println("manual");
}
65.      }
66.      if(automatic == true){ runAutomatic();
Serial.println("runAutomatic");
67.      }else{
if(inputValue == 'F'){ //if it receives the command 'F' then the robot goes
forward forward();
}else if(inputValue == 'B'){ //if it receives the command 'B' then the robot
goes backwards backward();
}else if(inputValue == 'R'){ //if it receives the command 'R' then the robot
goes right
turnRight();
}else if(inputValue == 'L'){ //if it receives the command 'L' then the robot
goes left
turnLeft();
}else if(inputValue == 'S'){ //if it receives the command 'S' then the robot
stops
stopMotor();
}
68.      }
69.      }
```

Line 51 `if (currentMillis - previousMillis >= interval)` checks whether the elapsed time since the last execution of this code block is greater than or equal to the specified `interval`. Line 52 `previousMillis = currentMillis; ` updates the `previousMillis` variable to store the current time as the new reference point for the execution of the next code block. Line 63 of the code reads the analog value from pin `A0` using `analogRead()` and maps it to the range of 100 to 255 to determine the motor speed based on the potentiometer setting. The mapped value is stored in the variable `motor speed`. Lines 64 to 65 explain if there is data available to be read from the Bluetooth module (`mySerial.available()` > 0), the code will read the data and store it in the `inputValue` variable. If the received value is 'W', the `automatic` flag is set to true, indicating that the robot should operate in automatic mode. If the received value is 'w', the `automatic` flag is set to false, indicating manual mode. In either case, the appropriate action is taken and a debug message is printed to the serial monitor.

Lines 66 to 68 explain if the `automatic` flag is true, the `runAutomatic()' function is called to execute the automatic behavior, and a debug message is printed to the serial monitor. If the `automatic` flag is false, the code checks the received `inputValue` and performs the appropriate action.

Lines 70 to 78 function from `takeUltrasonicData()`, which measures the distance using the ultrasonic sensor and stores the value in a variable.

```
70.      void takeUltrasonicData(){
71.      LeftSensor = sonar[1].ping_cm(); //measure the left sensor obstacle
         distance 72. delay(30);
73.  FrontSensor = sonar[0].ping_cm(); //measuring right sensor obstacle
distance 74. delay(30);
75.  RightSensor = sonar[2].ping_cm(); //measuring front sensor obstacle
distance 76. delay(30);
77.      // display the measurement results on the serial monitor
78.      }
```

Line 71 `LeftSensor = sonar[1].ping_cm();` measures the distance to the obstacle using the ultrasonic sensor connected to the second element (`sonar[1]`) of the `sonar` array. The measured distance in centimeters is stored in the `LeftSensor` variable. Line 73 `FrontSensor = sonar[0].ping_cm();` measures the distance to the obstacle using the ultrasonic sensor connected to the first element (`sonar[0]`) of the `sonar` array. The measured distance in centimeters is stored in the `FrontSensor` variable. Line 75 `RightSensor = sonar[2].ping_cm();` measures the distance to the obstacle using the ultrasonic sensor connected to the third element (`sonar[2]`) of the `sonar` array. The measured distance in centimeters is stored in the variable `RightSensor`.

Lines 79 to 88 function to control the robot movement (stop, go forward, backward, turn right, turn left).

```
79.      void stopMotor(){
80.      // analogWrite(ENA, speedMotor); //set the left motor speed based
         on the potentiometer setting
81.      // analogWrite(ENB, speedMotor); //set the right motor speed based
         on the potentiometer setting
82.      Serial.println("stopMotor");
```

```
83.     // the logic needed to command the driver module to control the
        motor
84.     digitalWrite(IN1, HIGH);
85.     digitalWrite(IN2, HIGH); 86. digitalWrite(IN3, HIGH);
87.     digitalWrite(IN4, HIGH);
88.     }
```

Lines 89 through 127 function to run the robot in automatic mode, using sensor data to make decisions and control movement.

```
89.     analogWrite(ENA, speedMotor); //set the speed of the left motor
        based on the potentiometer setting
90.     analogWrite(ENB, speedMotor); / set the speed of the right motor
        based on the potentiometer setting
91.     Serial.println("Forward");
92.     // logic needed to command the driver module to control the motor
93.     digitalWrite(IN1, LOW);  94. digitalWrite(IN2, HIGH);
95.     digitalWrite(IN3, HIGH);
96.     digitalWrite(IN4, LOW);
97.     }
98.     void backward(){
99.     analogWrite(ENA, speedMotor); //set the left motor speed based on
        the potentiometer setting
100.    analogWrite(ENB, speedMotor); //set the right motor speed based on
        the potentiometer setting
101.    Serial.println("backward");
102.    // the logic needed to command the driver module to control the
        motor
103.    digitalWrite(IN1, HIGH);
104.    digitalWrite(IN2, LOW);
105.    digitalWrite(IN3, LOW);
106.    digitalWrite(IN4, HIGH);
107.    }
108.    void turnRight(){
109.    analogWrite(ENA, 255); //set the left motor speed based on the
        potentiometer setting
110.    analogWrite(ENB, 255); //set the right motor speed based on the
        potentiometer setting
111.    Serial.println("turnRight");
112.    // the logic needed to command the driver module to control the
        motor
113.    digitalWrite(IN1, LOW);
114.    digitalWrite(IN2, HIGH);
115.    digitalWrite(IN3, LOW);
116.    digitalWrite(IN4, HIGH);
117.    }
118.    void turnLeft(){
119.    analogWrite(ENA, 255); //set the left motor speed based on the
        potentiometer setting
120.    analogWrite(ENB, 255); //set the right motor speed based on the
        potentiometer setting
121.    Serial.println("turnLeft");
122.    // logic needed to command the driver module to control the motor
123.    digitalWrite(IN1, HIGH);
124.    digitalWrite(IN2, LOW);
```

```
125.     digitalWrite(IN3, HIGH);
126.     digitalWrite(IN4, LOW);
127.     }
```

Lines 128 to 145 of the `runAutomatic() function, which implements the logic for automatic mode based on sensor data.

```
128.     void runAutomatic(){
129.     int s = digitalRead(irPin); //check if the IR sensor detects the
         floor
130.     takeUltrasonicData(); //call the distance measurement function
131.     if(s==HIGH) //if floor not detected, vacuum reverse
132.     {
Serial.print("IR: ");
Serial.println(s);
backward();
delay(1000); a=1;
}
133.     // if the floor is detected and the front sensor has no obstacles,
         the vacuum goes forward
134.     if ((a==0)&&(s==LOW)&&(LeftSensor <= avoidDistance && FrontSensor >
         avoidDistance && RightSensor <= avoidDistance) ||
(a==0)&&(s==LOW)&&(LeftSensor > avoidDistance && FrontSensor > avoidDistance
&& RightSensor > avoidDistance))
135.     {
1. forward();
136.     }
137.     // If the floor is detected and the right sensor has no obstacles,
         vacuum to the right.
138.     if ((a=1)&&(s==LOW)||(s==LOW)&&(LeftSensor <= avoidDistance &&
FrontSensor       <=       avoidDistance       &&       RightSensor       >
avoidDistance)||(s==LOW)&&(LeftSensor <= avoidDistance && FrontSensor <=
avoidDistance && RightSensor > avoidDistance)||(s==LOW)&& (LeftSensor <=
avoidDistance   &&   FrontSensor   >   avoidDistance   &&   RightSensor   >
avoidDistance)||(LeftSensor <= avoidDistance && FrontSensor > avoidDistance
&& RightSensor > avoidDistance))
139.     {
turnRight();
delay(100);
a=0; 140. }
141.     // if the floor is detected and the left sensor has no obstacles,
         then vacuum to the left
142.     if  ((s==LOW)&&(LeftSensor  >  avoidDistance  &&  FrontSensor  <=
         avoidDistance  &&  RightSensor  <=  avoidDistance)  ||(s==LOW)&&
         (LeftSensor  >  avoidDistance  &&  FrontSensor  >  avoidDistance  &&
         RightSensor  <=  avoidDistance)  ||(s==LOW)&&  (LeftSensor  >
         avoidDistance  &&  FrontSensor  <=  avoidDistance  &&  RightSensor  >
         avoidDistance) )
143.     {
1. turnLeft();
144.     }
145.     }
```

Line 129 `int s = digitalRead(irPin); ` reads the digital value from the IR sensor pin (`irPin`) to check if the floor is detected. The value is stored in the variable `s`. Line 130

`takeUltrasonicData();` calls the `takeUltrasonicData()` function to measure the distance using ultrasonic sensors and updates the values of `LeftSensor`, `FrontSensor`, and `RightSensor`. Lines 131-132, if the floor is not detected (`s` is `HIGH`), the vacuum robot moves backward (`backward()`) for 1 second and sets `a` to 1. This indicates that the robot is in the backward state.

Lines 134 - 136 if the floor is detected (`s` is `LOW`) and there are no obstacles ahead (`SensorFront` is greater than `avoidDistance`), the robot vacuum moves forward (`forward()`). Lines 139 - 140 if the floor is detected and there is an obstacle on the right side (condition with `SensorRight` and `SensorLeft`), the vacuum robot turns right (`turnRight()`) and sets `a` back to 0. Lines 142 - 143 if the floor is detected and there is an obstacle on the left side (condition with `SensorLeft` and `SensorRight`), the vacuum robot turns left (`turnLeft()`).

4.3. Results

The results of the analysis after design and testing on the robot vacuum cleaner. In the initial condition, the robot is ON and the battery is fully charged. The robot is able to move freely with smartphone control that has been connected via Bluetooth. the robot can move manually and automatically. Robot testing is carried out on a flat plane with obstacles in the room. Testing the movement forward, backward, to the right, and to the left is able to run normally even though it is not 100% perfect. For vacuum cleaners in vacuuming dust and smallsized dirt can be lifted. For power using RC batteries, because when using a battery with a voltage of 12V only lasts a short time and is unable to move the vacuum cleaner. The ultrasonic sensor that functions as a distance detection device with obstacles is considered less than perfect, because it is less accurate and precise in detecting and the speed in detecting is still a little slow, so that when it cannot detect obstacles there is a collision that results in the robot not moving.

4.4. Discussion

The results of designing a robot vacuum cleaner using ultrasonic sensors and Bluetooth are that the robot is able to move forward, backward, left, and right. However, the robot is only able to move on a flat plane, if the surface is uneven, it interferes with the movement of the wheels. Vacuum cleaner works well, able to suck up dust and small dirt. Overall it works well, it's just that when the ultrasonic sensor in detecting obstacles is less accurate, it causes the robot to collide with obstacles and the robot will stand still

CONCLUSION

The results of testing the overall system of robot vacuum cleaner using ultrasonic sensors and Bluetooth based on Arduino Uno can be given some conclusions as follows:

1. Vacuum cleaners that function to clean small dust and dirt can do their job according to what is desired by using smartphone control.

2. Bluetooth which is used as a connecting medium between the robot and the smartphone can function properly.

3. The robot can move in all directions in a field that has a flat surface to facilitate movement and avoid slippage on the wheels.

4. The power used in the robot is able to run the vacuum cleaner and last long enough. Here the author uses RC Batteries

The suggestions that can be given for the development of a robot vacuum cleaner using Ultrasonic Sensors and Bluetooth based on Arduino Uno are to use Lidar for automatic movement, so that the results obtained when measuring distance and detecting obstacles can be accurate and precise. And the high speed of Lidar can detect and map in real-time.

REFERENCES

[1] M. D. Faraby, M. Akil, A. Fitriati, and I. Isminarti, "Rancang Bangun Robot Pembersih Lantai Berbasis Arduino," *JTT (Jurnal Teknol. Terpadu)*, vol. 5, no. 1, p. 70, 2017. https://jurnal.poltekba.ac.id/index.php/jtt/article/view/214

[2] A. Saefullah, D. Immaniar, and R. A. Juliansah, "Sistem Kontrol Robot Pemindah Barang Menggunakan," *Ccit*, vol. 8, no. 2, pp. 1–12, 2015. https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=web&cd=&ved=0CAIQw7AJahcKEwj4mZnXroiAAxUAAAAAHQAAAAAQAg&url=https%3A%2F%2Fejournal.raharja.ac.id%2Findex.php%2Fccit%2Farticle%2Fdownload%2F314%2F255%2F&psig=AOvVaw14S9f-neYwg6sWfhKXMwH2&ust=1689223603923495&opi=89978449

[3] A. Trianes and A. Yulianto, "Implementasi Behavior Based Control Dan Pid Pada Robot Vacuum Cleaner," *J. Sains dan Inform.*, vol. 1, no. 2, pp. 72–77, 2015. https://jsi.politala.ac.id/index.php/JSI/article/view/29/28

[4] P. Heru, S. P. Sutisna, and E. Sutoyo, "Rancang Bangun Sistem Penyapu Pada Robot Pembersih Lantai," *Mekanika*, vol. 1, no. 2, pp. 1–14, 2020. https://ejournal.unugha.ac.id/index.php/me/article/view/388

[5] U. K. Yuliza, "ROBOT PEMBERSIH LANTAI BERBASIS ARDUINO UNO DENGAN SENSOR Abstrak Perkembangan Ilmu pengetahuan dan teknologi saat ini sangatlah pesat , terutama di bidang teknologi elektronika mempengaruhi kehidupan masyarakat untuk melangkah lebih maju , praktis dan si," *J. Teknol. Elektro, Univ. Mercu Buana*, vol. 6, no. 3, pp. 136–143, 2015. https://www.neliti.com/publications/143244/robot-pembersih-lantai-berbasis-arduinouno-dengan-sensor-ultrasonik

[6] I. Chigozie Williams, "Mathematical Modeling of a Robot Vacuum Cleaner Suction with Matlab Simulink," *Int. J. Res. Innov. Appl. Sci.* |, vol. IV, no. X, pp. 2454–6194, 2019. https://www.semanticscholar.org/paper/Mathematical-Modeling-of-a-Robot-VacuumCleaner-Williams/fb16c4a843e85441f1af920f783c03e52f802ee8

[7] F. L. Ivan Suwanda, Elang Derdian M., "Rancang Bangun Robot Omni Wheel Penyedot Debu Menggunakan Sensor Accelerometer Berbasis Mikrokontroler ATMega16," *J. Tek. Elektro Univ. Tanjungpura*, vol. 2, no. 1, 2014. https://jurnal.untan.ac.id/index.php/jteuntan/article/view/8355

[8] S. Ardhi, M. Kom, and S. M. T. Hari, "Perancangan dan Pembuatan Prototipe Alat Pembersih Lantai dengan Kendali dari Jaringan Bluetooth," *Semin. Int. dan Konf. Nas.*

IDEC, pp. 3–04, 2016. https://www.researchgate.net/profile/Setya-Ardhi/publication/319256288_PERANCANGAN_DAN_PEMBUATAN_PROTOTIPE_ALAT_PEMBERSIH_LANTAI_DENGAN_KENDALI_DARI_JARINGAN_BLUETOOTH/links/599e57490f7e9b892bb41186/PERANCANGAN-DANPEMBUATAN-PROTOTIPE-ALAT-PEMBERSIH-LANTAI-DENGAN-KENDALIDARI-JARINGAN-BLUETOOTH.pdf

[9] A. Z. Hasibuan and M. S. Asih, "Rancang Bangun Robot Vacum Cleaner Berbasis Mikrokontroler dengan Pengendali Smartphone Android," *InfoTekJar (Jurnal Nas. Inform. dan Teknol. Jaringan)*, vol. 4, no. 1, pp. 116–120, 2019. https://core.ac.uk/download/pdf/304226736.pdf

[10] B Satria, H Wijaya, and R Susanto, "Robot pembersih debu otomatis," *J. Tek.*, no. 9, pp. 15–22, 2. http://docplayer.info/38040287-Robot-pembersih-debu-otomatisabstract.html