Floor Cleaning Bot

Design Lab Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Karan Markandey and Prakhar Gupta
(210102042 and 210108075)

under the guidance of

Dr. Manish Bhatt

to the

DEPARTMENT OF ELECTRONICS AND ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

Abstract

A floor cleaning robot is a device that combines electronic and mechanical control systems to enable automated cleaning of floors. This system is programmed to clean large areas of flooring in both homes and offices. The robot is designed to reach almost every corner of a room and is small in size for better manoeuvrability. It can be controlled using an Android phone that has wifi technology. The robot is built around an Arduino microcontroller, which is supported by communication modules like NodeMCU, motors, and brushes to ensure it works efficiently. It is powered by a 12V rechargeable battery. The robot cleans by continuously moving its cleaning brush on the floor surface, with mopping taking place at the same time as the robot starts the cleaning process. Using a floor cleaning robot can lead to savings in labour costs over time. One of the key advantages of this home cleaning robot is its cost-effectiveness and its ability to operate independently without the need for human intervention.

Keywords- *Autonomous vehicle, sweeping, moping, ultrasonic sensor, water sprayer, NodeMCU.*

Contents

Chapter 1

Introduction

1.1 Problem Statement

To develop an autonomous floor cleaning bot that works on both automatic as well as manual mode.

1.2 Overview

Recently, the interest in robotic cleaners has grown significantly, finding applications in homes, hotels, offices, and more. These robots combine mechanical and electrical components for automated sweeping and mopping, offering greater convenience than traditional vacuums. They operate autonomously, reducing the need for manual intervention. Equipped with an automatic water sprinkler, they handle mopping without requiring a wet cloth. The robot employs three motors for movement and water pump operation, controlled by a motor driver circuit. Obstacle detection is managed by three ultrasonic sensors. The LM293D IC is used to drive the wheel motor. The core of the robot is an Arduino UNO microcontroller, receiving control signals via a NodeMCU communication module from an Android phone.

1.3 Motivation

The primary motivation behind the development of floor cleaning robots is to enhance efficiency and convenience in cleaning tasks. These robots, capable of both dry and wet cleaning, have revolutionized the way cleaning is done by automating the entire process, thereby eliminating the need for human intervention. This not only reduces the operational burden but also saves significant amounts of time and energy. Furthermore, the safety aspect is a crucial factor, as these robots can perform cleaning tasks in potentially hazardous environments, reducing the risk of injuries. Lastly, the use of such robots is a step towards embracing technological advancements and promoting smart living solutions. They represent a significant stride in the field of home automation, contributing to the creation of smarter and more efficient homes.

Chapter 2

Literature Survey

Historically, a variety of products, including sawdust and tea leaves, were used to wipe up spills and debris. Even with the development of increasingly sophisticated cleaning instruments like mops and brooms, manual work was still needed.

By mechanically vacuuming up dust particles, the vacuum cleaner's innovation helped lessen the work needed to clean. But human supervision and care were still needed for this.

Later, in an effort to further automate the cleaning process, room cleaning robots with random cleaning algorithms started to appear. Nevertheless, these devices frequently fell short of offering whole cleaning coverage and need user oversight.

Then, a variety of platform-specific smart vacuum cleaners were created, but they lacked a specific cleaning algorithm and were not very user-friendly. Although some wirelessly networked smart vacuum cleaners were developed, obstacle detection proved to be a problem for them.

The present requirement is to create a robot that can simultaneously mop and vacuum, providing flawless cleaning. Most cleaning robots on the market today still need to be manually activated.

This project's primary goals are to design and build a robot with an Arduino Mega, ultrasonic sensors, an LCD display, and other parts. The robot will also be controlled by the user via NodeMCU. The principal objective is to develop a vacuum and mopping robot that can navigate by using data from limit switches and ultrasonic sensors, at a cost that is reasonable and appropriate for usage by Indian enterprises and consumers.

A cleaning robot is a basic autonomous robot that cleans a designated area using software and algorithms that have been preprogrammed. Such a robot's primary goal is to minimize time-consuming human engagement throughout the cleaning process.

With the help of an Arduino Mega, ultrasonic sensors, an LCD display, and other parts, this project seeks to design and build a robot that can be controlled wirelessly via a NodeMCU. While there are a lot of cleaning and mopping robots on the market, only a small number are reasonably priced. Furthermore, fewer robots are equipped with both mopping and cleaning capabilities.

Techniques	Advantages	Disadvantages
Sawdust	Was used to remove water or any liquid that spilled on the floor	Running electricity bill. Depending on the vacuum cleaner model uses hundreds to thousands watts of electricity
Broomsticks and mops	Removes all particles from a hard floor on the first pass makes cleaning more efficient	It requires manpower.
The emergence of room cleaning robot	Vacuum under furniture and beds	The emergence of room cleaning robot with random cleaning algorithm occurred but the system should monitored by the user and they failed to produce complete cleaning
Smart vacuum cleaner	They are more convenient to use because they can vacuum on their own.	Smart vacuum cleaner using wireless network developed but it is much complex on obstacle detection.

Table 2.1 Literature Survey

Chapter 3

Our Proposal

We present the design of a low-cost Internet of Things (IoT)-based autonomous cleaning robot for cleaning rooms and halls. Our design will offer two distinct modes of operation: a manual mode and an automatic mode, with an Arduino UNO serving as the core microcontroller.

3.1 Manual Mode

For the manual mode, our design leverages WiFi technology to enable easy control of the robot. We will incorporate the NodeMCU ESP8266 module and interface its control with the Blynk IoT app, allowing users to remotely operate the cleaning robot. The robot will periodically apply the floor cleaner solution to ensure thorough and efficient cleaning of the designated areas on instruction of the user.

3.2 Automatic Mode

In the automatic mode, our design will work on the continuous feedback provided by the HC-SR05 ultrasonic sensor to navigate an obstruction-free path. The robot will periodically apply the floor cleaner solution to ensure thorough and efficient cleaning of the designated areas.

The integration of the Arduino UNO, the NodeMCU ESP8266 module, and the HC-SR05 ultrasonic sensor allows our autonomous cleaning robot to switch seamlessly between the manual and automatic modes, catering to the user's preference and the specific cleaning requirements of the environment. This innovative design, along with the use of readily available and cost-effective components, aims to make autonomous cleaning more accessible and practical for a wide range of applications, from residential spaces to commercial establishments.

3.3 Features

Here is a rephrased and expanded version of the features:

- **LCD Display for User Assistance**: We have integrated an LCD display into our design to provide the user with clear and intuitive guidance during the operation of the cleaning robot. This display will showcase key information, such as the current mode of operation, cleaning progress, and any relevant alerts or status updates.

- **High-Capacity LiPo Batteries for Extended Runtime**: To ensure prolonged cleaning sessions, our design incorporates high-capacity Lithium-Polymer (LiPo) batteries. These batteries offer superior energy density and runtime compared to traditional battery technologies, allowing the autonomous cleaning robot to operate for extended durations without the need for frequent recharging.

- **Dedicated Cleanser Reservoir with Low-Level Indicator**: Our design features a dedicated reservoir to store the floor cleaning solution. This reservoir is equipped with a low-level indicator, which will notify the user when the cleaning solution needs to be refilled. This feature ensures effective and consistent cleaning by maintaining the appropriate amount of cleaning solution throughout the operation.

- **"Find My Cleaner" Function**: In the event that the autonomous cleaning robot becomes misplaced or lost, our design includes a "Find My Cleaner" feature. Upon user command, a loud buzzer will be activated, allowing the user to easily locate and retrieve the robot from its current position.

- **Low Battery Buzzer**: To prevent the cleaning robot from unexpectedly shutting down due to a depleted battery, our design incorporates a low battery buzzer. This feature will emit a distinct audible alert when the battery level drops below a predefined threshold, prompting the user to recharge the robot's batteries in a timely manner.

The integration of these advanced features, including the LCD display, high-capacity LiPo batteries, dedicated cleanser reservoir, "Find My Cleaner" functionality, and low battery buzzer, enhances the overall user experience and ensures the reliable and efficient operation of our autonomous cleaning robot.

3.4 Model Diagram

The flowchart represents the operational modes of a programmed cleaning robot controlled by a latching switch. When the switch is in the 'ON' position, the robot operates in an automatic mode, performing its tasks independently. However, when the switch is 'OFF', the robot switches to a manual mode, requiring instructions from the Blynk IoT app to carry out its functions. This highlights the robot's ability to operate both autonomously and under direct control.

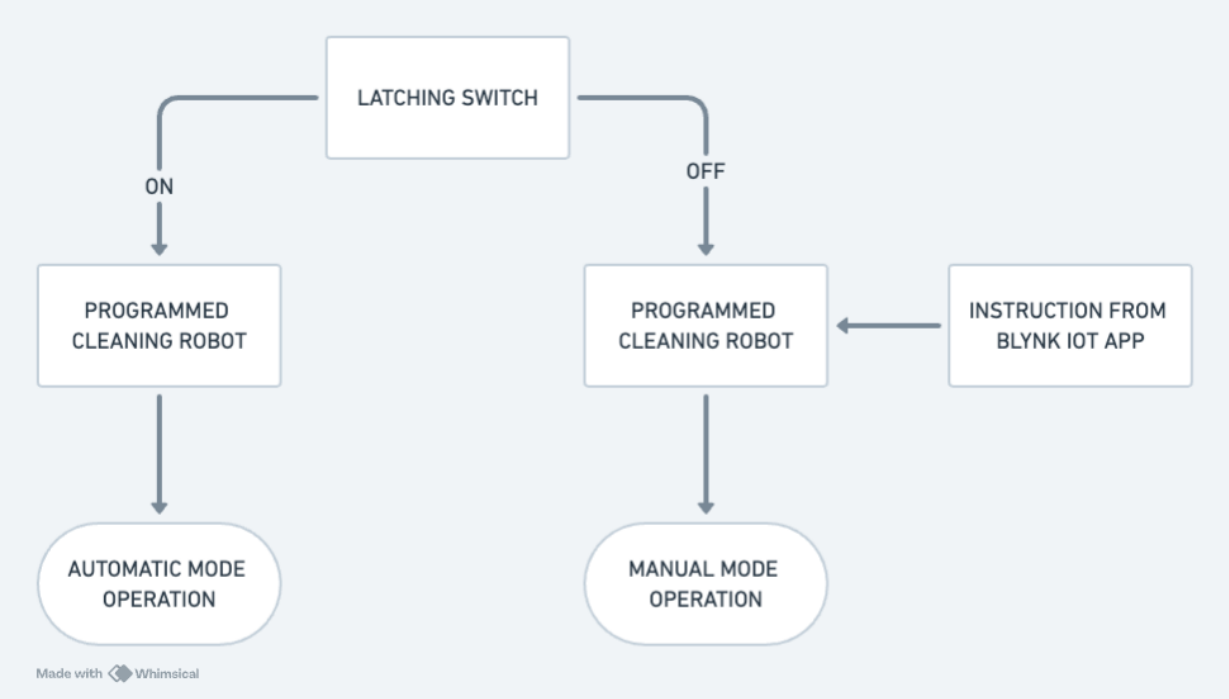

Fig. 3.1 Proposed Model Diagram

Chapter 4

Components

4.1 Arduino UNO

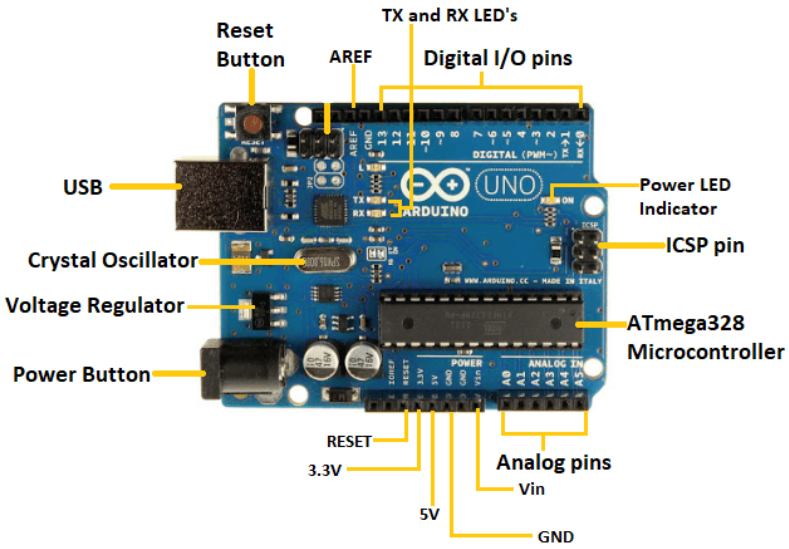

Fig. 4.1 Arduino UNO

Constructed around the ATmega328P microprocessor, the Arduino UNO is renowned for being incredibly user-friendly, particularly in contrast to other boards such as the Arduino Mega. This board has many features, such as shields, extra circuitry, and digital and analog input/output (I/O) pins. Six analog pin inputs, fourteen digital pins, a USB connector for interface, a power jack for power supply, and an ICSP (In-Circuit Serial Programming) header for programming are the features that define the Arduino UNO. An Integrated Development Environment (IDE) that works with both online and offline platforms is used to program the Arduino UNO.

Components of Arduino Uno :

- **ATmega328 Microcontroller** - An 8-bit microcontroller from the Atmel family is called the ATmega328P. On a single chip, it combines memory (SRAM, EEPROM, Flash), an ADC, SPI serial ports, I/O lines, registers, a timer, and interrupts. An oscillator for clock signals is also included.
- **ICSP pin** - The Arduino board's firmware can be programmed by the user through the In-Circuit Serial Programming pin.
- **Power LED Indicator** - The LED's ON state indicates that the power is on. The LED won't turn on when the power is off.
- **Digital I/O pins** - The digital pins are either HIGH or LOW in value. Digital pins have numbers ranging from D0 to D13.
- **TX and RX LED's** - The illumination of these LEDs signifies the effective data transfer.
- **AREF** - The Arduino UNO board receives a reference voltage from the external power source via the Analog Reference (AREF) pin.
- **Reset button** - It's employed to give the connection a Reset button.
- **USB**- It permits the board and computer to be connected. It is necessary in order to program the Arduino UNO board.
- **Crystal Oscillator** - The Arduino UNO is a powerful board because of the 16MHz frequency of the Crystal oscillator.
- **Voltage Regulator**- The input voltage is changed to 5V by the voltage regulator.
- **GND** - pins for the ground. The ground pin functions as a zero-voltage pin.
- **Vin** - It is the input voltage.
- **Analog Pins** - Analog pins have numbers ranging from A0 to A5. Reading the analog sensor utilized in the connection is the purpose of the analog pins. Additionally, it can function as GPIO (General Purpose Input Output) pins.

The technical specifications of the Arduino UNO are listed below:

1. There are 20 Input/Output pins present on the Arduino UNO board. These 20 pis include 6 PWM pins, 6 analog pins, and 8 digital I/O pins.
2. The PWM pins are Pulse Width Modulation capable pins.
3. The crystal oscillator present in Arduino UNO comes with a frequency of 16MHz.
4. It also has a Arduino integrated WiFi module. Such Arduino UNO board is based on the Integrated WiFi ESP8266 Module and ATmega328P microcontroller.
5. The input voltage of the UNO board varies from 7V to 20V.

6. Arduino UNO automatically draws power from the external power supply. It can also draw power from the USB

4.2 NodeMCU

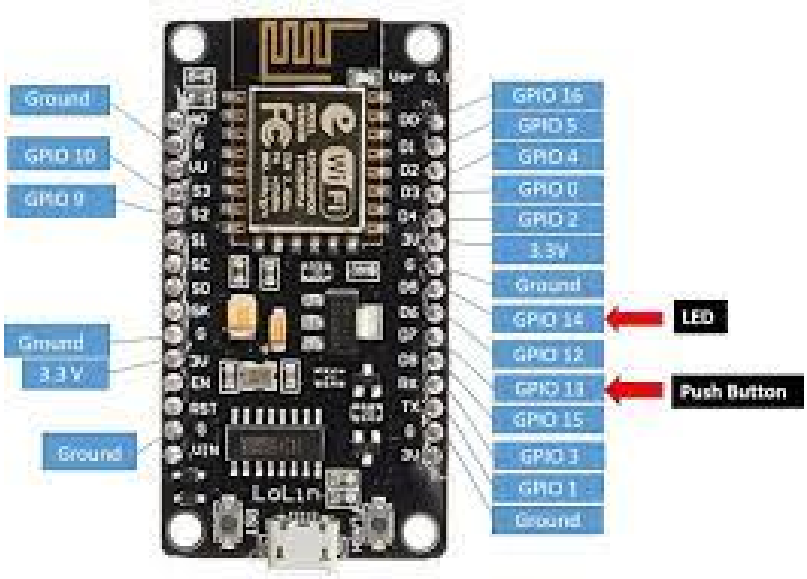

Fig. 4.2 NodeMCU

The NodeMCU, standing for Node MicroController Unit, is an open-source platform for both software and hardware development. It's centered around a cost-effective System-on-a-Chip (SoC) known as the ESP8266. This chip, a product of Espressif Systems, encompasses key computer components such as a CPU, RAM, WiFi networking capabilities, and even a contemporary operating system and SDK. This comprehensive feature set makes it a preferred choice for various Internet of Things (IoT) projects.

While the ESP8266 chip is packed with features, it can be challenging to access and utilize. Basic tasks like powering it on or sending a keystroke to the onboard "computer" require soldering wires with the correct analog voltage to its pins. Programming it involves dealing with low-level machine instructions that the chip hardware can interpret. While this level of integration is manageable when the ESP8266 is used as an embedded controller chip in mass-produced electronics, it poses a significant challenge for hobbyists, hackers, or students looking to use it in their personal IoT projects.

What about the Arduino? The Arduino initiative has developed an open-source hardware blueprint and software SDK for their adaptable IoT controller. The Arduino hardware, much like the NodeMCU, is a microcontroller board equipped with a USB connector, LED indicators, and conventional data pins. It also establishes standard interfaces for communication with sensors or other boards. However, in contrast to the NodeMCU, the Arduino board can accommodate various types of CPU chips (usually an ARM or Intel x86 chip),

memory chips, and a range of programming environments. There's even a reference design for the ESP8266 chip within the Arduino framework. The versatility of Arduino, however, leads to considerable differences among products from different manufacturers. For instance, WiFi capabilities are absent in most Arduino boards, and some even replace the USB port with a serial data port.

- **Power Pins** - There are four power pins. VIN pin and three 3.3V pins.

    1. **VIN**:Direct supply to the NodeMCU/ESP8266 and its peripherals can be achieved via VIN. The NodeMCU module's inbuilt regulator controls the power sent to the VIN pin; alternatively, you can supply 5V regulated power to the pin.
    2. **3.3V**: The onboard voltage regulator's 3.3V pins are its output, and you can use them to power external components.

- **GND** denotes the ground pins of ESP8266.

- **I2C Pins**: I2C sensors and peripherals are connected using I2C pins. There is support for both I2C Master and I2C Slave. Programmatic implementation of I2C interface capability is possible, with a maximum clock frequency of 100 kHz. It should be mentioned that the I2C clock frequency needs to be higher than the slave device's slowest clock frequency.

- **GPIO Pins** -The 17 GPIO pins of the NodeMCU/ESP8266 can be dynamically allocated to various functions, including I2C, I2S, UART, PWM, IR remote control, LED light, and button. Every digitally enabled GPIO has the option of being set to high impedance, internal pull-up, or pull-down. It can also be configured as an input and set to either edge-trigger or level-trigger, which will cause CPU interruptions.

- **ADC Channel** - An embedded 10-bit accuracy SAR ADC is present in the NodeMCU. ADC can be utilized to implement the two tasks. Testing the input voltage of the TOUT pin and the power supply voltage of the VDD3P3 pin. They cannot, however, be put into practice simultaneously.

- **UART Pins**- With two UART ports (UART0 and UART1), the NodeMCU/ESP8266 can communicate at up to 4.5 Mbps and support asynchronous protocols (RS232 and RS485). The TXD0, RXD0, RST0, and CTS0 pins of UART0 can be used for interface. But since UART1 (TXD1 pin) can only transfer data, log printing is typically done with it.

- **SPI Pins** - Two SPIs (SPI and HSPI) in slave and master modes are available on the NodeMCU/ESP8266. The following general-purpose SPI capabilities are also supported by these SPIs: Four timing modes for the transfer of SPI format Both the split 80 MHz clocks and up to 80 MHz 64-Byte FIFO maximum

- **SDIO Pins**- Secure Digital cards can be directly interfaced with NodeMCU/ESP8266 thanks to its Secure Digital Input/Output Interface (SDIO) capability. Supported SDIO versions are 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0.

- **PWM Pins** - There are four Pulse Width Modulation (PWM) channels on the board. Programmatically implemented PWM output can be utilized to drive LEDs and digital motors. The adjustable PWM frequency range is between 1000 s and 10000 s (100 Hz and 1 kHz).

- **Control Pins**: To operate the NodeMCU/ESP8266, utilize the control pins. These pins are the WAKE pin, the reset pin (RST), and the chip enable pin (EN).

    1. **EN**:When the EN pin is pulled HIGH, the ESP8266 device is activated. Pulling LOW uses the least amount of power for the chip.
    2. **RST**: The ESP8266 chip can be reset using the RST pin.
    3. **WAKE**: The chip can be woken from deep sleep via the wake pin.

4.3 Ultrasonic Sensors

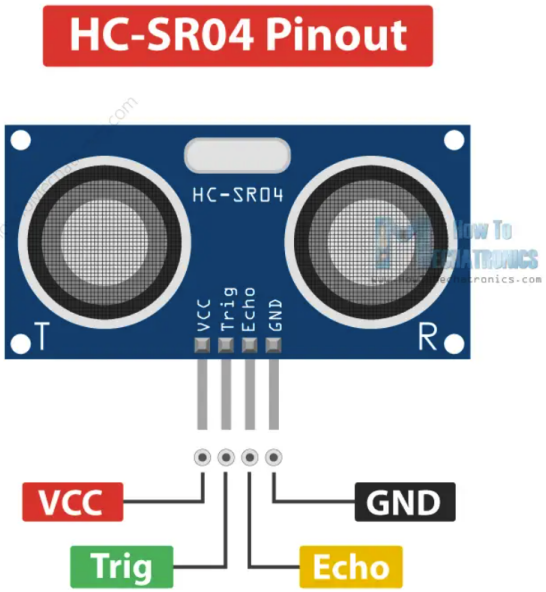

Fig. 4.3 Ultrasonic Sensor Pins

The HC-SR04 is an affordable and easy to measure distance which has a range from 2cm to 400cm.

It works by emitting ultrasonic sound waves and then detecting their echoes when they bounce off an object. This method is similar to how sonar works, which is used by submarines to navigate underwater.

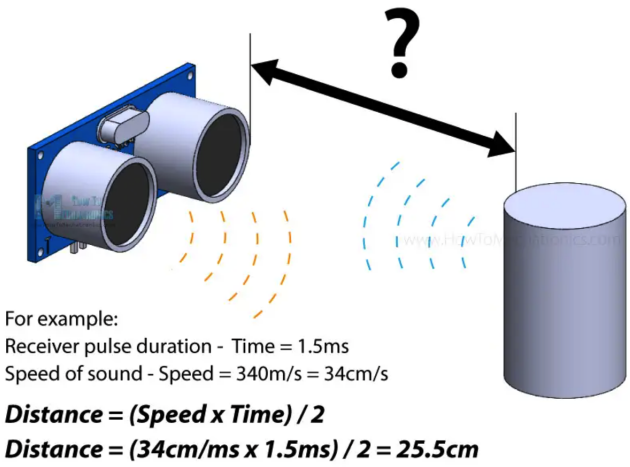

Fig. 4.4 Ultrasonic Sensor Working

4.3.1 Working

The transmitter emits a burst of high-frequency sound, typically at 40,000 Hz (inaudible to humans). If an object is in the path of this sound wave, the wave will bounce back towards the sensor.
The echo pin goes high after the sound wave is emitted, indicating it's listening for the echo. If no echo is received within a set timeout (typically around 38 milliseconds), the echo pin returns to a low state, indicating there's no object within the sensor's range. If an echo is received before the timeout, the time the echo pin stays high corresponds to the round-trip travel time of the sound wave, allowing for distance calculation.
If there is no object or reflected pulse, the Echo pin will time-out after 38ms and get back to low state.
If we receive a reflected pulse, the Echo pin will go down sooner than those 38ms. According to the amount of time the Echo pin was HIGH, we can determine the distance the sound wave travelled, thus the distance from the sensor to the object.
For that purpose we are using the following basic formula for calculating distance:
Distance = Speed x Time

We have knowledge of both the velocity and duration parameters. The duration corresponds to the period during which the Echo pin remains in the HIGH state, while the velocity is the speed of sound, approximately 340m/s. An important step in our calculation is to halve the final result. This is necessary because we are determining the time it takes for the sound wave to reach the object and return.

4.4 Relay Module

A power relay module is an electrical switch that is operated by an electromagnet. The electromagnet is activated by a separate low-power signal, often from a microcontroller.

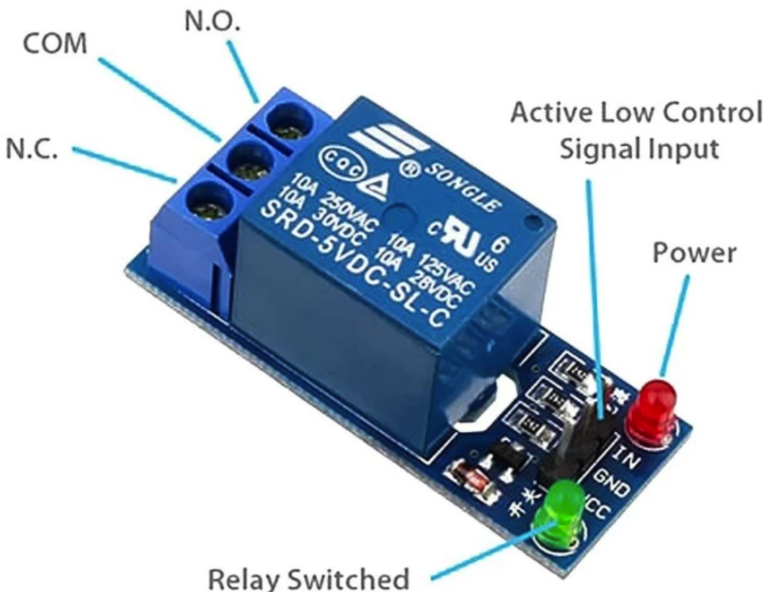

Fig. 4.5 Relay Module

When activated, the electromagnet pulls to either open or close an electrical circuit.

The basic structure of a simple relay includes a wire coil wrapped around a soft iron core or solenoid, an iron yoke that provides a low reluctance path for magnetic flux, a movable iron armature, and one or more sets of electrical contacts. When current flows through the coil, it generates a magnetic field that activates the armature. This movement of the armature makes or breaks the connection between the moving and fixed contacts.

When the relay is de-energized, a spring or gravity returns the armature to its relaxed position, opening or closing the contacts as needed. Power relays are designed to switch circuits quickly and efficiently, making them useful in a variety of applications.

4.5 Blynk

Blynk is a platform designed for the Internet of Things. It allows users to control hardware remotely, display sensor data, store data, visualize it, and perform various other functions. The Blynk platform has three main components:

- **Blynk App**: This mobile app allows users to create custom interfaces for their projects using various widgets provided by Blynk.
- **Blynk Server**: This component is responsible for all the communication between the smartphone and the hardware. For the project, users need to install a private Blynk

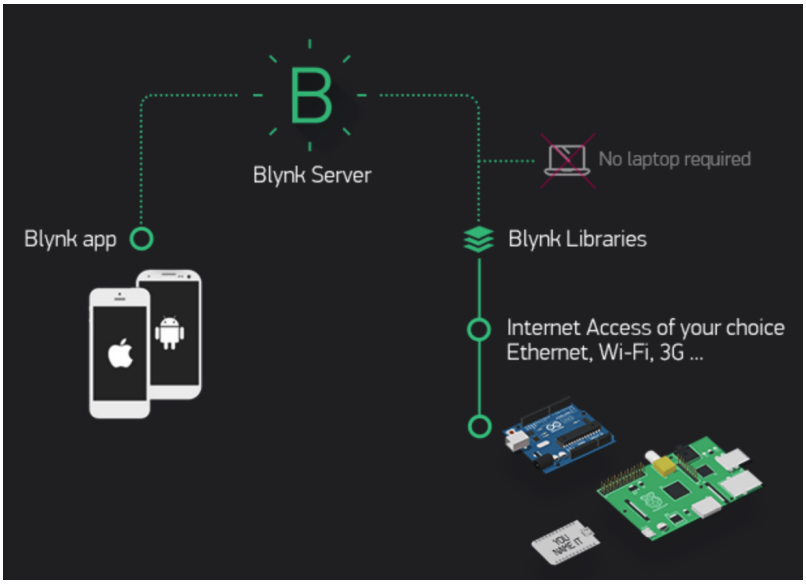

Fig. 4.6 Blynk

server on their local computer (Windows, macOS, Raspberry Pi, etc.). The Blynk server is open-source and can handle thousands of devices, even on a Raspberry Pi.

- **Blynk Libraries**: These libraries, available for a variety of popular hardware platforms, enable communication with the Blynk server and process all incoming and outgoing commands.

4.6 DC Motor

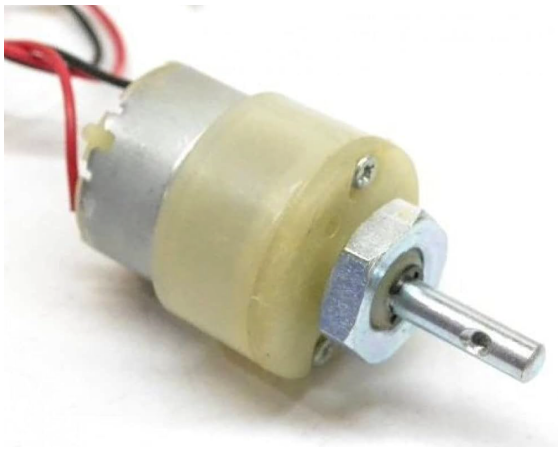

Fig. 4.7 DC Motor

A DC motor is a type of rotary electric machine that converts electrical energy into mechanical energy. To periodically change the direction of current flow in part of the

motor, most DC motors include some internal mechanisms, either electromechanical or digital.
The majority of DC motors produce rotary motion, where the motor shaft rotates. However, a linear motor directly produces force and motion in a straight line, rather than rotational motion.

4.7 7805 Voltage Regulator

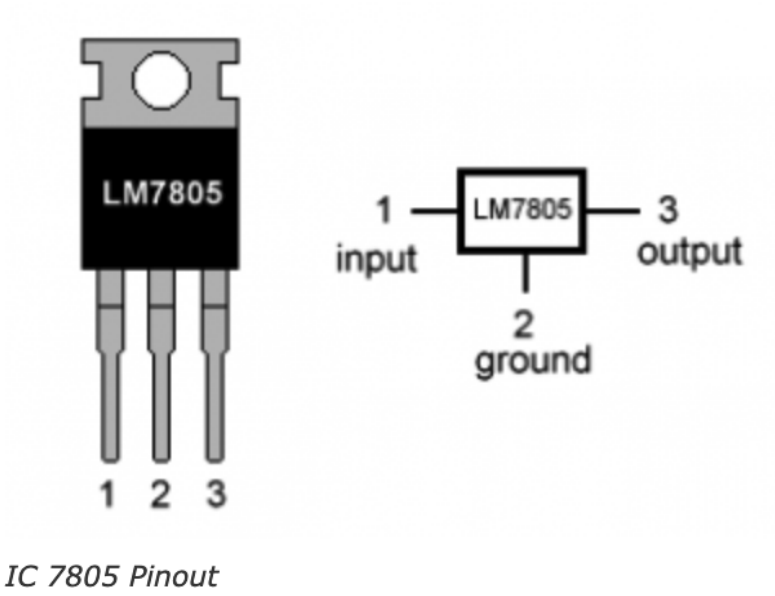

Fig. 4.8 7805 Voltage Regulator

Voltage sources in a circuit may have fluctuations, resulting in an inconsistent voltage output. A voltage regulator IC helps maintain the output voltage at a constant value, even with these input fluctuations.

The 7805 voltage regulator, part of the 78xx series of fixed linear voltage regulators, is a popular integrated circuit (IC) used to address this issue. The "xx" in "78xx" indicates the specific output voltage provided by the regulator - in the case of the 7805, it provides a regulated +5 volt output. The 7805 IC also includes provisions to add a heat sink, as needed to dissipate excess heat.

4.7.1 7805 Voltage Regulator IC Specifications

- Minimum Input voltage is 7V
- Maximum Input Voltage is 35V
- Current rating Ic = 1A

- Maximum Output Voltage VMax=5.2V
- Minimum Output Voltage VMin=4.8V

4.8 Diaphragm Pump

Fig. 4.9 Diaphragm Pump

The R385 Pump is a high-quality diaphragm pump that can be used to dispense water. This pump has a working life of approximately 2,000 hours and can be used to dispense heated water up to 80C (176F).

The R385 Pump is well-suited for various projects such as garden automation, fountains, or waterfalls. It can be used to move water from one container to another without needing to submerge the pump in the water itself.

Overall, the R385 Pump is a durable and versatile option for applications that require reliable water movement and dispensing, while being able to handle heated water as needed.

4.8.1 Specification of R385 DC water Pump

- Model: R385

- Material: ABS , Silicone and Metal
- Shape: Round
- Voltage 6 Vot when: power is 6 Watt / H
- Outlet diameter: Inner: 6 mm, Outer: 9 mm

4.8.2 Features of R385 Water Pump

- Voltage Range: 6 - 12 Volts (recommended 9V-1A ; 12V-1A)
- Power: 6 watt/H
- Flow Rate: 1 - 3 Liter/min
- Maximum Suction range: 2 meter
- Maximum Head range: 3 meter
- Life Span: 2000 Hours
- Max Water Temperature: 80

4.9 LiPo Battery

Fig. 4.10 LiPo Battery

A lithium-polymer battery, or LiPo, is a type of rechargeable battery. True LiPo batteries use a solid polymer material as the electrolyte, along with lithium as one of the electrodes.

However, commercially available LiPo batteries are actually hybrid designs. They use either a gel polymer or a liquid electrolyte, contained within a pouch format. These are more accurately termed lithium-ion polymer batteries, rather than true LiPo batteries.
The key distinction is the use of a solid polymer electrolyte in true LiPo batteries, versus the gel or liquid electrolytes found in the commercially available lithium-ion polymer battery products.

4.10 L298N Motor Driver Module

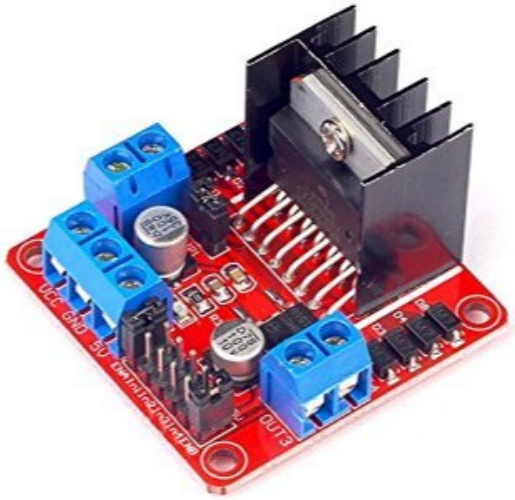

Fig. 4.11 L298N Motor Driver

The L298N Motor Driver Module is a high-power motor driver module used for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator.

The L298N Module can control up to 4 DC motors, or 2 DC motors with directional and speed control. The module includes the L298 Motor Driver IC, 78M05 Voltage Regulator, resistors, a capacitor, a Power LED, and a 5V jumper.

The 78M05 voltage regulator will only be enabled when the 5V jumper is in place. When the power supply is less than or equal to 12V, the internal circuitry will be powered by the

voltage regulator, and the 5V pin can be used as an output to power a microcontroller. However, if the power supply is greater than 12V, the jumper should not be placed, and a separate 5V supply should be provided through the 5V terminal to power the internal circuitry.

The module also includes ENA and ENB pins for controlling the speed of Motor A and Motor B, respectively. Additionally, IN1 IN2, and IN3 IN4 are the direction control pins for Motor A and Motor B.

4.11 LCD Display

LCD displays are very popular and widely used in many electronics projects because they are great for displaying simple information, such as sensor data, while being very affordable.

A standard 16-pin LCD display has the following pin functions:

- **Ground (GND)**: The first pin from left to right, used to connect to ground.
- **VCC**: The second pin, connected to the 5V pin on the Arduino board.
- **Vo**: The contrast adjustment pin, where a potentiometer can be connected to control the display contrast.
- **Register Select (RS)**: This pin selects whether we are sending commands or data to the LCD. When RS is low (0V), we are sending commands, such as setting the cursor position, clearing the display, or turning the display off. When RS is high (5V), we are sending data, such as characters to be displayed.
- **Read/Write (R/W)**: This pin selects the mode of operation, either read or write. In most cases, we only need the write mode, as we are sending data and commands to the LCD.
- **Enable (E)**: This pin enables the writing of data to the LCD's registers. The 8 data pins (D0-D7) are used to send the 8-bit data when writing to the registers. The remaining 8 pins (D0-D7) are used to send the 8-bit data to the LCD when writing commands or characters to be displayed.

Chapter 5

Theory

5.1 Circuit Diagram and its explanation

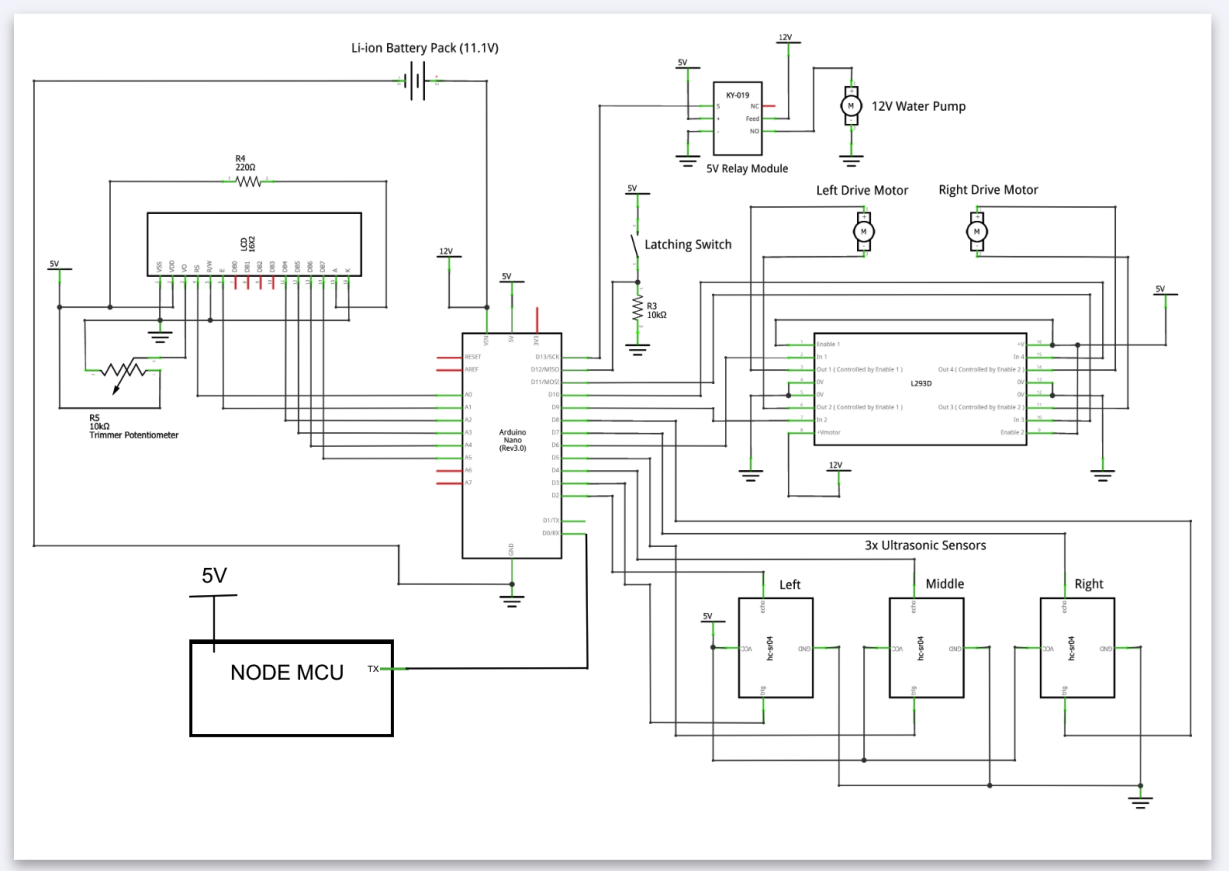

Fig. 5.1 Circuit Diagram

TThe Arduino UNO is the main microcontroller that drives our self-cleaning robot. We have decided to use a high-capacity 4500 mAh 3-cell Lithium-Polymer (LiPo) battery as the main power source, which has a combined output of 11.1V. Long-term operation and reliable power transmission to the cleaning robot's different components are guaranteed by

this sturdy battery.

We have combined an LCD display with the Arduino UNO to help the user when they are using it. This display improves the user experience overall by offering real-time information and direction.

We have integrated a relay module to allow for the regulated application of the floor cleaning solution. By establishing electrical isolation between the high-voltage diaphragm water pump and the low-voltage control circuit, this module guarantees dependable and safe operation.

Two geared DC motors operating at 100 rpm provide the robot with motion, and they are controlled by an L293D motor driver. The cleaning robot can move across the intended surfaces at the required speed and torque thanks to these strong motors.

In order to facilitate obstacle-free navigation in the automated mode, three HC-SR05 ultrasonic sensors have been positioned at the robot's front, left, and right sides. These sensors keep an eye on the surroundings all the time, which enables the robot to navigate effectively and safely.

Toggling between the automatic and manual modes of operation is accomplished via a latching switch. The Blynk IoT app, which connects via Wi-Fi to the onboard NodeMCU ESP8266 module, allows the user to operate the cleaning robot manually. Then, via wired serial communication, the NodeMCU transmits the user's commands to the Arduino UNO, allowing the robot to react appropriately.

The Arduino UNO, LiPo battery, LCD display, relay module, geared DC motors, ultrasonic sensors, and NodeMCU ESP8266 are just a few of the diverse parts that come together to form an all-encompassing and adaptable autonomous cleaning robot that can operate in both manual and automatic modes to meet the needs of the environment and the user.

5.2 Code

5.2.1 Code Explantion

The provided Arduino code is for a robot that uses ultrasonic sensors to navigate in both manual and automatic modes. It uses the NewPing library to interface with three ultrasonic sensors (left, middle, and right) to measure distances. The motor control is achieved through an L293D motor driver connected to four pins. Additionally, it has a button for manual control switching and a pump connected to pin 13, likely for water pumping or similar functions.

In the setup() function, pin modes are set, and initial states are defined. The LCD is initialized to display "Welcome!", and serial communication is started at 9600 baud rate.

The robot has several movement functions (moveForward(), moveBackward(), turnLeft(), turnRight(), and stop()) that control the motor direction using the motor driver

pins. These functions are called based on the received serial data in the manualMode() function when the robot is in manual mode.

In the loop() function, the robot checks for serial data. If data is available, it switches to manual mode, and if not, it operates in automatic mode. In automatic mode (automaticMode()), the robot reads distances from the sensors and decides its movement based on the readings. If obstacles are detected, it either moves backward and turns right or left to avoid obstacles. If no obstacles are detected, it moves forward.

The readSensorL(), readSensorM(), and readSensor() functions read distances from the left, middle, and right sensors, respectively. These distances are then used in the automaticMode() to make navigation decisions.

Overall, the robot can be controlled manually using serial input or operate autonomously by avoiding obstacles detected by the ultrasonic sensors.

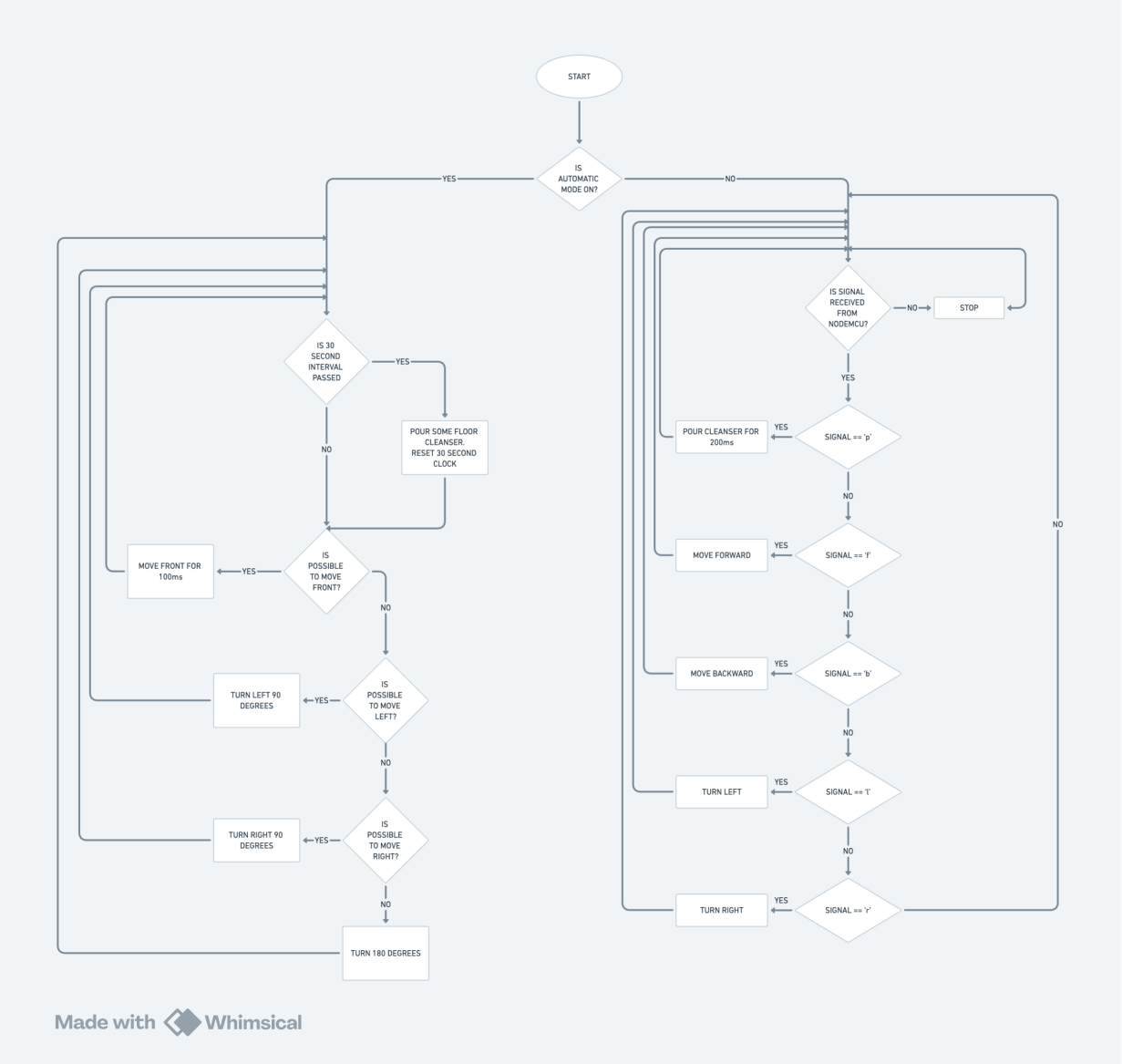

Fig. 5.2 flowchart

```
#include <NewPing.h>    //import libraries
#include <LiquidCrystal.h>

char incomingByte; // for incoming serial data
char prevByte ='\0'; // for incoming serial data

const int echo_L = 2;   //initialize pin numbers
const int trig_L = 3;
const int echo_M = 4;
const int trig_M = 5;
const int echo_R = 7;
const int trig_R = 8;

const int in1 =6;
const int in2 =9;
const int in3 =11;
const int in4 =10;

const int button = 12;
const int pump = 13;

int motor_speed = 255;    //speed of the motor can be set between 125 (minimum) and 255 (maximum)
int max_distance = 200;   //max distance of ultrasonic sensors is set to 200cm
int distance_L = 0;
int distance_M = 0;
int distance_R = 0;

NewPing sonar_L(trig_L, echo_L, max_distance);    //initialize all the 3 sensors
NewPing sonar_M(trig_M, echo_M, max_distance);
NewPing sonar_R(trig_R, echo_R, max_distance);
LiquidCrystal lcd(A0, A1, A2, A3, A4, A5);    //initialize LCD

void setup() {
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
  pinMode(in3,OUTPUT);
```

```
void setup() {
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
  pinMode(in3,OUTPUT);
  pinMode(in4,OUTPUT);

  pinMode(button, INPUT);
  pinMode(pump, OUTPUT);
  digitalWrite(L1, LOW);
  digitalWrite(L2, LOW);
  digitalWrite(R1, LOW);
  digitalWrite(R2, LOW);
  digitalWrite(pump, LOW);
  lcd.begin(16, 2);
  lcd.print("Welcome!");
  Serial.begin(9600);   //begin serial communication via bluetooth at 9600 baud rate
  delay(2000);
}

void stop(){
  digitalWrite(in1,LOW);
  digitalWrite(in2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in4,LOW);
  Serial.println("Stopped");
}
void moveForward(){
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in3,HIGH);
  digitalWrite(in4,LOW);
  Serial.println("Moving Forward");
}

```

```
67  void moveBackward(){
68    digitalWrite(in1,LOW);
69    digitalWrite(in2,HIGH);
70    digitalWrite(in3,LOW);
71    digitalWrite(in4,HIGH);
72    Serial.println("Moving Backward");
73  }
74  void turnLeft(){
75    digitalWrite(in1,LOW);
76    digitalWrite(in2,HIGH);
77    digitalWrite(in3,HIGH);
78    digitalWrite(in4,LOW);
79    Serial.println("Turning Left");
80  }
81
82  void turnRight(){
83    digitalWrite(in1,HIGH);
84    digitalWrite(in2,LOW);
85    digitalWrite(in3,LOW);
86    digitalWrite(in4,HIGH);
87    Serial.println("Turn Right");
88  }
89
90  void moveVehicle(char c){
91    if(c=='f'){
92      moveForward();
93    }else if(c == 'b'){
94      moveBackward();
95    }else if(c == 'r'){
96      turnRight();
97    }else if(c=='l'){
98      turnLeft();
99    }
100   return ;
```

```
void loop() {
  // send data only when you receive data:
  // manual
  if (Serial.available() > 0) {
    // read the incoming byte:
    lcd.clear();                    //manual mode
    lcd.print("Manual Mode");
    manualMode();
  }
  else{
    lcd.clear();                    //manual mode
    lcd.print("Automatic Mode");
    automaticMode();
  }
}

void manualMode(){
  incomingByte = Serial.read();
    if(incomingByte == 'f' || incomingByte == 'b' || incomingByte == 'l' || incomingByte == 'r'   )
      if(prevByte == incomingByte){
        prevByte ='\0';
        stop();
        //Serial.println("s");
      }else{
        //Serial.println(incomingByte);
        moveVehicle(incomingByte);
        prevByte =incomingByte;
      }
    }
    else{
      stop();
    }
}

```

```
void automaticMode()
{
  distance_L = readSensor_L();    //read distance from all the 3 sensors
  distance_M = readSensor_M();
  distance_R = readSensor_R();
  lcd.clear();    //print distance on LCD
  lcd.print("L=");
  lcd.print(distance_L);
  lcd.print("cm ");
  lcd.print("M=");
  lcd.print(distance_M);
  lcd.print("cm");
  lcd.setCursor(0, 1);
  lcd.print("R=");
  lcd.print(distance_R);
  lcd.print("cm");

  if(distance_M <= 20)    //if middle sensor distance is less than 20cm
  {
    if(distance_R > distance_L)   //check if there is place at right or left
    {
      if((distance_R <= 20) && (distance_L <= 20))    //if there is no place on both sides
      {
        moveStop();
        delay(200);
        moveBackward();   //move back
        delay(2000);
      }
      else
      {
        moveBackward();  //move back then turn right
        delay(500);
        moveRight();
        delay(2000);
```

```
    else
    if(distance_R < distance_L)
    {
      if((distance_R <= 20) && (distance_L <= 20))
      {
        moveStop();   //move back
        delay(200);
        moveBackward();
        delay(2000);
      }
      else
      {
        moveBackward();   //move back then turn left
        delay(500);
        moveLeft();
        delay(2000);
      }
    }
  }

  else
  if(distance_R <= 15)    //if right sensor distance is less than 20cm
  {
    moveLeft();   //turn left
    delay(500);
  }
  else
  if(distance_L <= 15)    //if left sensor distance is less than 20cm
  {
    moveRight();    //turn right
    delay(500);
  }
```

```
  else
  {
    moveForward();    //in all other cases keep on moving forward
  }
}

int readSensor_L()    //read distance in centimeters from left sensor
{
  delay(70);
  int cm_L = sonar_L.ping_cm();
  if(cm_L==0)
  {
    cm_L = 250;
  }
  return cm_L;
}

int readSensor_M()    //read distance in centimeters from left sensor
{
  delay(70);
  int cm_M = sonar_M.ping_cm();
  if(cm_M==0)
  {
    cm_M = 250;
  }
  return cm_M;
}

int readSensor_R()    //read distance in centimeters from left sensor
{
  delay(70);
  int cm_R = sonar_R.ping_cm();
  if(cm_R==0)
  {
    cm_R = 250;
```

```
int readSensor_R()    //read distance in centimeters from left sensor
{
  delay(70);
  int cm_R = sonar_R.ping_cm();
  if(cm_R==0)
  {
    cm_R = 250;
  }
  return cm_R;
}
```

Fig. 5.3 Arduino Code

```
#define BLYNK_TEMPLATE_NAME "NodeMCU"
#define BLYNK_AUTH_TOKEN "b9qFIrrnjhptGJ3bS4vaPBQ3NtwtJp6g"
#define BLYNK_TEMPLATE_ID "TMPL32r-xTRFd"

// Comment this out to disable prints and save space
#define BLYNK_PRINT Serial

//hardWare Specifics
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = BLYNK_AUTH_TOKEN;

// Your WiFi credentials.
char ssid[] =  "Prakhar";
char pass[] =  "prakhar@1210";


bool fetch_blynk_state = true;  //true or false
#define wifiLed   LED_BUILTIN   //D0int
int wifiFlag = 0;

// the virtual pins
#define VPIN_SWITCH_1    V0
#define VPIN_SWITCH_2    V1
#define VPIN_SWITCH_3    V2
#define VPIN_SWITCH_4    V3

BlynkTimer timer;


BLYNK_WRITE(VPIN_SWITCH_1)
{
  Serial.write('f');
}
```

```
BLYNK_WRITE(VPIN_SWITCH_2)
{
  Serial.write('b');
}
BLYNK_WRITE(VPIN_SWITCH_3)
{
  Serial.write('l');
}
BLYNK_WRITE(VPIN_SWITCH_4)
{
  Serial.write('r');
}

BLYNK_CONNECTED() {
  // Request the latest state from the server
  if (fetch_blynk_state){
    Blynk.syncVirtual(VPIN_SWITCH_1);
    Blynk.syncVirtual(VPIN_SWITCH_2);
    Blynk.syncVirtual(VPIN_SWITCH_3);
    Blynk.syncVirtual(VPIN_SWITCH_4);
  }
}

void checkBlynkStatus() { // called every 2 seconds by SimpleTimer

  bool isconnected = Blynk.connected();
  if (isconnected == false) {
    Serial.println("Blynk Not Connected");
  }
  if (isconnected == true) {
    if (!fetch_blynk_state){
    }

    wifiFlag = 0;
  }
}

```

```
void checkBlynkStatus() { // called every 2 seconds by SimpleTimer

  bool isconnected = Blynk.connected();
  if (isconnected == false) {
    Serial.println("Blynk Not Connected");
  }
  if (isconnected == true) {
    if (!fetch_blynk_state){
    }

    wifiFlag = 0;
  }
}

void setup()
{
      // Debug console
      Serial.begin(9600);

      pinMode(wifiLed, OUTPUT);

      Blynk.begin(auth, ssid, pass);

      timer.setInterval(2000L, checkBlynkStatus);
      delay(1000);

}

void loop()
{
  Blynk.run();
  timer.run();

}
```

Fig. 5.4 NodeMCU code

Chapter 6

Results

This project's autonomous cleaning robot provides an adaptable and affordable indoor cleaning solution. The robot's basic microprocessor, the Arduino UNO, allows it to smoothly combine a number of components, including as geared DC motors, LCD screens, high-capacity LiPo batteries, relay modules, and ultrasonic sensors. Users have the option to give the robot direct control or let it travel on its own thanks to the combination of manual and automatic modes, as well as a speed mode feature. Reliable functioning and an improved user experience are guaranteed by advanced features including the low battery buzzer, "Find My Cleaner" functionality, and a specialized cleaner reservoir. With this all-encompassing design, which serves a variety of home and business applications, autonomous cleaning is intended to become more widely available and useful.

Chapter 7

Future Scope

There is a lot of room for improvement and new uses for the autonomous cleaning robot design. By incorporating sophisticated navigation techniques, like SLAM, the robot would be able to map its surroundings and create effective cleaning routes. Improving cleaning performance on a range of surfaces could be accomplished by using water-based solutions and specialized instruments. Adding scheduling and multi-room cleaning capabilities would make the automated system more complete. Developing remote monitoring capabilities and linking the robot to smart home platforms would improve user ease of use and upkeep. Further investigating eco-friendly cleaning agents and adaptable sensors would increase efficacy and sustainability. Future developments have the potential to completely transform mundane cleaning duties by enabling smart, approachable cleaning solutions to be widely available in both household and commercial contexts.

Chapter 8

Conclusion

Our inexpensive, Internet of Things (IoT)-based autonomous cleaning robot's design demonstrates a thorough and creative approach to tackling the difficulties of effective and user-friendly cleaning in a range of indoor spaces. We have developed a strong and dependable platform that easily combines a wide range of components, including as high-capacity LiPo batteries, LCD screens, relay modules, geared DC motors, and ultrasonic sensors, by using the Arduino UNO as the central microcontroller.

The NodeMCU ESP8266 module and the Blynk IoT app enable the combination of manual and automatic modes of operation, giving users the choice to operate the cleaning robot manually or let it navigate on its own. This allows the user to customize the cleaning process to meet their individual needs and preferences. To further increase the robot's versatility, our design also includes a speed setting that allows users to alternate between a low and fast operating speed depending on what has to be cleaned.

Advanced features like the "Find My Cleaner" feature, the low battery buzzer, and the dedicated cleanser reservoir with low-level indicators improve the user experience overall and guarantee the cleaning robot's dependable and effective performance even in difficult environments.

Our goal is to increase the accessibility and practicality of autonomous cleaning by creating a feature-rich and affordable solution that can be used in a variety of settings, including homes and businesses. If this cleaning robot concept is successfully implemented, it will revolutionize everyday cleaning duties and enhance consumers' overall quality of life by contributing to the increased adoption of IoT-enabled smart home and office technology.

Chapter 9

References

[1]. G Tuangzhi Dai and Tiequn Chen "Design on measurement and control system of cleaning robot based on sensor array detection", IEEE International conference on control automation Guangzhou, May 30 to June 1, 2007, China.
[2]. Youngkak Ma, seungwoo Kim, Dongik Oh and Youngwan Cho, "A study on the development of home mess- cleanup robot McBot", IEEE/ASME international conference on advanced mechatronics, July 2-5, 2008, Xian, China.
[3]. Zheng Zhao, Weihai Chen, Chen C.Y. Peter and Xingming Wu "A novel navigation system for indoor cleaning robot", IEEE International Conference on Robotics and Biomimetics (ROBIO), Dec. 3 to Dec. 7, 2016, Qingdao, China.
[4]. V.Prabakaran, M.Rajesh Elara, T.Pathmakumar and S. Nansai "HTetro: A Tetris inspired shape-shifting floor cleaning robot", IEEE International Conference on Robotics and Automation (ICRA), May 29 to June 3, 2017, Singapore