ERAGON FLOOR CLEANING ROBOT

A Project Report submitted to

SRIMAD ANDAVAN ARTS & SCIENCE COLLEGE (AUTONOMOUS)

Affiliated to BHARATHIDASAN UNIVERSITY

in partial fulfillment of the requirement for the award of the Degree of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

Submitted By

G. NARAYANAN

(U19CS0067)

R. NAVANEETHA KRISHNAN

(U19CS0068)

Under the Guidance of

MR. M. KUMARESAN, MCA., M.Phil.,

ASSISTANT PROFESSOR

PG AND RESEARCH DEPARTMENT OF COMPUTER SCIENCE

SRIMAD ANDAVAN ARTS & SCIENCE COLLEGE (AUTONOMOUS)

(Affiliated with Bharathidasan University)

Nationally Re-accredited by NAAC

An ISO 9001:2015 Certified Institution

No 7, Nelson Road, Tiruvanaikoil,

Tiruchirappalli - 620 005.

JUNE 2022

Srimad Andavan Arts & Science College (Autonomous)

(Affiliated with Bharathidasan University)

Nationally Re-accredited by NAAC

An ISO 9001:2015 Certified Institution

No 7, Nelson Road, Tiruvanaikoil,

Tiruchirappalli - 620 005.

MR. M. KUMARESAN, MCA., M.Phil.,

ASSISTANT PROFESSOR

<u>CERTIFICATE</u>

This is to certify that the Project Report appellation entitled "ERAGON FLOOR CLEANING ROBOT" was submitted to Srimad Andavan Arts & Science College (Autonomous), Tiruchirappalli, Affiliated to Bharathidasan University, Tiruchirappalli in partial fulfillment of the requirement for the award of the degree of Bachelor of Science in Computer Science for the academic Year 2021-2022, is a bonafide record of the project work done by **G.NARAYANAN (REG. U19CS0067)** under my guidance and supervision. The project has previously not formed the basis for the award of any degree to the candidate.

SIGNATURE OF THE GUIDE **SIGNATURE OF THE HOD**

SIGNATURE OF THE EXAMINER

Date: 07.06.2022

DECLARATION

I declare that the project work entitled " **ERAGON FLOOR CLEANING ROBOT** " submitted to Srimad Andavan Arts and Science College (Autonomous), Tiruchirappalli affiliated to Bharathidasan University, Tiruchirappalli in partial fulfillment of the requirement for the award of the degree of Bachelor of Science in Computer Science, for the academic year 2021 -2022, is the result of the original work carried out by me under the guidance and supervision of **Mr M.KUMARESAN ASST PROFESSOR** PG and Research Department of Computer Science, Srimad Andavan Arts and Science College (Autonomous), Thiruchirappalli.

Date : 07.06.2022

Place : Trichy -5.

Signature of the Candidate

(G.NARAYANAN)

VIVA VOCE

The Viva-Voce Examination of this Project Work is held on 07.06.2022 at Srimad Andavan Arts and Science College (Autonomous), Tiruchirappalli-5

Internal Examiner	**External Examiner**

ACKNOWLEDGEMENT

First, I am very thankful to the Almighty for showering the Blessings throughout my life. I would like to express my heartfelt thanks to my beloved parents who have sacrificed a lot for my future. This support was the foundation stone for my efforts.

I acknowledge my sincere thanks to **Shri Ammangi V.BALAJI**, Secretary & Correspondent, **Dr.N.RAMANUJAM Advisor, Dr.M.PITCHAIMANI.,** Principal, **Dr.MEERA PARTHASARATHY**, Vice Principal **, Dr.S.R.SATHYANARAYANAN,**Vice Principal, **Dr.S.MADHU,** Dean of Basic Sciences,**Dr.V.UPENDRAN.,** Head-PG and Research Department of Computer Science, **Dr.N.KARTHIKEYAN,** Co-ordinator, PG and Research Department of Computer Science, for giving me the opportunity to do my Under Graduate Degree course in this renowned institution and providing facilities to carry out the project work.

I am grateful to **MR. M. KUMARESAN, MCA., M.Phil.,**PG and Research Department of Computer Science, the internal guide for timely suggestions and constant encouragement and support that led to the accomplishment of the project.

I extend my thanks to all the teaching and non-teaching staff members and all lab assistants of the Computer Science Department, who helped me either directly or indirectly to complete this project.

I wish to express my heartfelt thanks to my friends for their useful tips that helped me to complete this project successfully.

ERAGON FLOOR CLEANING ROBOT USING EMBEDDED C

ABSTRACT

Households of today are becoming smarter and more automated. Home automation delivers convenience and creates more time for people. Domestic robots are entering the homes and people's daily lives, but it is yet a relatively new and immature market.

However, growth is predicted, and the adoption of domestic robots is evolving. Several robotic vacuum cleaners are available on the market but only a few implement wet cleaning of floors. The purpose of this project is to design and implement an Eragon Floor Cleaning Robot Autonomous and Manual via Phone Application.

Eragon Floor Cleaning Robot is designed to make the cleaning process become easier rather than using a manual Vacuum and Mop.

The main objective of this project is to design and implement a vacuum and Mop robot prototype by using Arduino Uno, Arduino Motor Shield, Motor Drive L29 3D,Ultrasonic Sensor, and Bluetooth and to achieve the goal of this project.

Vacuum and Mop Robot will have several criteria that are user-friendly. The main objective of this Robot is to automatically find the dust and tiny particle and vacuum it and mop that area.

CONTENT

1. INTRODUCTION

We are going do Floor cleaning robot using Arduino board and I kept a name Eragon Floor Cleaning robot for a floor cleaning robot. This robot can vacuum all tiny particle dust and it mop the floor cleanly. This robot has water in container it sprinkles water and mop the floor This robot has ultrasonic sensor from that sensor it identifies everything.

Eragon is a smart phone-controlled robot that cleans your house's floor! The rotating mops on the front of the robot along with a foam roller (used to paint walls, not here) at the back can do the job perfectly. There's also a water pump and water reservoir which can be switched on when required to throw water on the floor and make the mops moist for a proper clean. The foam roller is movable, which means you can lift it when not in use. I've also added speed controls for the driver motors.

The project uses Bluetooth communication via an HC-05 Bluetooth module to send the commands to the most used micro controller- Arduino UNO. The robot is powered on a 12V lead acid battery, the ideal voltage for all motors used here... The driver motor pair are 100rpm ones while for the mops I've used 75rpm plastic ones.

The best part is that the mops used were home-made, from old CDs and rags and they clean just perfectly. This is a smaller version so might not be suitable for a large area. There can be tons of other features added, like making it completely autonomous, which I couldn’t be due to shortage of time.

It took me around 4-5 days to complete this thing, simultaneously it vacuume at the back side.

MODULES:

Arduinoboard, Bluetooth Module, ultrasonic sensor ,Geared motors,motor driver board,Standard servo motor.12V Sealed lead acid rechargeable battery,12V Water pump,NPN power transistors,Male-male/ female-female/ male-female jumper wires.

2 REQUIREMENT SPECIFICATION

2.1 HARDWARE REQUIREMENT

SYSTEM	:	HP X86(Laptop)
HARD DISK	:	1tb + 256 gb SSD
SCREEN	:	Liquid Crystal Display
MOUSE	:	Touchpad
RAM	:	Installed 8GB
KEYBOARD	:	110 keys enhanced.

Other hardware lists are

- Arduino Uno
- Bluetooth Module -HC-05
- Ultrasonic Sensor
- Geared motors
- Motor driver board
- Standard servo motor
- 12V Sealed lead acid rechargeable battery
- 12V Water pump
- NPN power transistors

2.2 SOFTWARE REQUIREMENTS

OPERATING SYSTEM	:	WINDOWS 11 PRO
LANGUAGE	:	EMBEDDED C
MOBILE APPLICATION	:	Arduino Bluetooth control

2.3 DATASHEET

1.ARDUINO UNO DATASHEET

Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

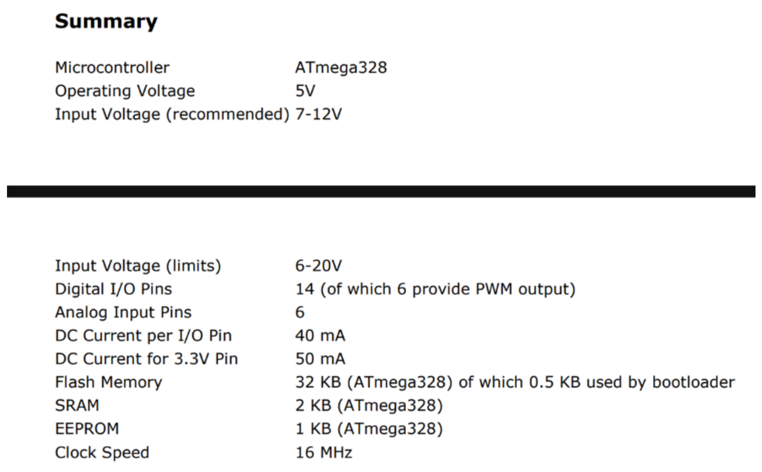
• 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In the uture, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.

• Stronger RESET circuit.

• Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions

Summary



Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery.

The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts.

If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library. There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and ATmega328 ports. The mapping for the Atmega8, 168, and 328 is identical.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX).

An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed.

However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details.

For SPI communication, use the SPI library.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board). For details. The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer.

It communicates using the original STK500 protocol (reference, C header files). You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware. The ATmega16U2/8U2 i loaded with a DFU bootloader, which can be activated by:

• On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.

• On Rev2 or later boards: there is a resistor that pulls the 8U2/16U2 HWB line to the ground, making it easier to put into DFU mode.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

2.BLUETOOTH-HC05 DATASHEET

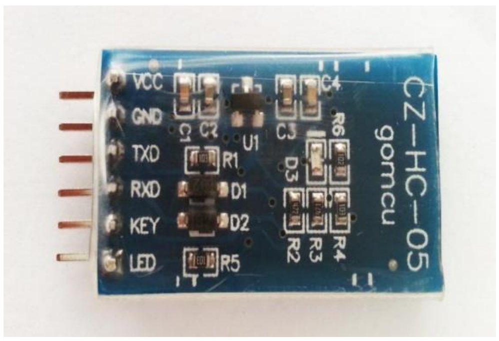
Introduction

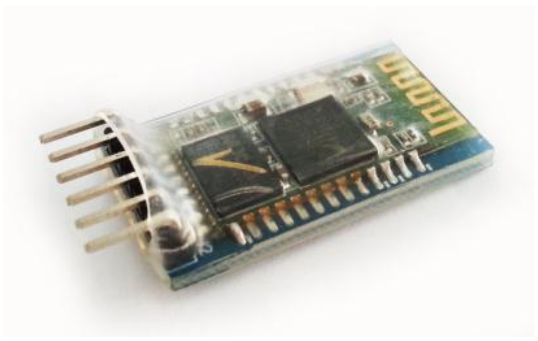
HC-05 Bluetooth Module is an easy-to-use Bluetooth SPP (Serial Port Protocol) module, designed for a transparent wireless serial connection setup. Its communication is via serial communication which makes an easy way to interface with controller or PC.

HC-05 Bluetooth module provides switching mode between master and slave mode which means it able to use neither receiving nor transmitting data. Specification:

- Model: HC-05
- Input Voltage: DC 5V
- Communication Method: Serial Communication
- Master and slave mode can be switched .

Pin Definition





The HC-05 Bluetooth Module Defaults Using AT Commands. Diagram below shows the hardware connection between HC-05 Bluetooth Module and Arduino UNO.
Besides Arduino, it may interface with any microcontroller such as PIC and etc.

- VCC ◊ Arduino 5V
- GND ◊ Arduino GND
- TXD ◊ Arduino Pin RX
- RDX ◊ Arduino Pin TX
- KEY ◊ Connect to the air for communication mode

After completing hardware and source code installation on Arduino UNO, the next step is setting up PC site. In order to communicate with Arduino UNO, a Bluetooth device is needed as well on PC site for data transfer between Arduino UNO and PC via Bluetooth devices.

HC-05 Bluetooth Module USB Plug In Bluetooth Device Arduino UNO PC or Computer Data Transfer via Bluetooth devices In order to use Bluetooth device in PC site,
Bluetooth Device Driver is needed to install in PC. If your USB Plug in Bluetooth device does not provides providestallation, you may download this PC Site Bluetooth Software and install it.

Plug in your Bluetooth device to PC during installation and restart PC after installation.
After setting up the Arduino UNO and PC site, now we proceed to next session which will show communication between Arduino UNO and PC through Bluetooth devices via communication mode.

3.MOTOR SHIELD DATASHEET

Features:

• 2 connections for 5V 'hobby' servos connected to the Arduino's high-resolution dedicated timer - no jitter!

• Up to 4 bi-directional DC motors with individual 8-bit speed selection (so, about 0.5% resolution)

• Up to 2 stepper motors (unipolar or bipolar) with single coil, double coil, interleaved or micro-stepping.

- 4 H-Bridges: L293D chipset provides 0.6A per bridge (1.2A peak) with thermal shutdown protection, 4.5V to12V
- Pull down resistors keep motors disabled during power-up
- Big terminal block connectors to easily hook up wires (10-22AWG) and power
- Arduino reset button brought up top
- 2-pin terminal block to connect external power, for separate logic/motor supplies
- Tested compatible with Mega, UNO & Duemilanove
- Dimensions: 69mm x 53mm x 14.3mm (2.7in x 2.1in x 0.6in)

The L293D is a dedicated module to fit in Arduino UNO R3 Board, and Arduino MEGA. It is actually a motor driver shield that has full featured Arduino Shield can be used to drive 2 to 6 DC motor and 4 wire Stepper motor and it has 2 set of pins to drive a SERVO.

DC Motors and Stepper Motors and switching power transistor. To simplify to used as two bridges on each pair of channels and equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included.

The device is suitable for use in switching applications at frequencies up to 5kHz. The L293D is assembled in a 16 lead plastic package which has 4 centre pins connected together and used for heat sinking.

The L293D is assembled in a 20 lead surface mount which has 8 centre pins connected together and used for heat shrinking.

- Control 2 Servos.
- Logic Control Voltage VSS: 4.5 ~ 5.5 V
- Motor Supply Voltage VSS: 15v
- Drive operating current IO: 1.2A
- 8 Stage Serial Shift Registers Wiring a DC Motor

4.12V WATER PUMP DATASHEET

Specifications:

Working voltage: 6-12V DC

Load operating current: 0.5-0.7A

Maximum flow: 1-3L/Min

Maximum head: 3 m

Maximum suction range: 2 m

Temperature: ≤80℃

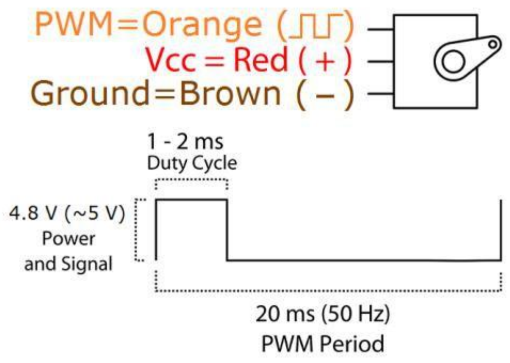
Service life: 2500H

Inlet pipe diameter: 6mm

Size: 86 x 43mm

5.SERVO MOTOR DATASHEET

31002-MD SG90 MINI SERVO MOTOR Mini servo for robotics & modeling. Nylon gears for smooth quite operation. Rated: Op. Voltage 3.5-6VDC, Stall Torque: 1.2kg/cm @ 4.8V, 1.6kg/cm @ 6V, Speed: 300Deg./Sec (60/.12sec.), Dead Bandwidth: 5us. Pulse Cycle: 20mS -90deg (Max "Left") ~1mS 0deg (Center) ~1.5mS +90deg (Max "Right") ~2mS Futaba/JR 0.1 pitch connector on 9in. cable. Includes 3 control Arms and screws L: 23mm W: 12mm O/A T: 29mm WT: .02



6.12V RECHARGEABLE BATTERY DATASHEET

- High Service Life : 3000 cycles and more
- Deep discharge allowed up to 100 %
- Ultra safe Lithium Iron Phosphate chemistry (no thermal run-away, no fire or explosion risks)
- Embedded BMS (Battery Management System) : improve lifespan AND secure the battery
- No Lead, nearth piles of earth, no acid, no degassing
- Calendar life > 10 years
- Excellent temperature robustness (-20 °C up to +60 °C)
- Flexible deployment: up to 10 packs in parallel and 4 in serial
- Constant power during discharge (very low internal resistance)

• Very low Peukert's losses (energy efficiency >98 %)

• Very low self-discharge

7.100 RPM GEARED MOTORS DATASHEET

Metal DC Geared Motor - 12V 100RPM 42kg.cm

INTRODUCTION

This is a metal DC geared motor, 100% pure copper coils, high-density molecular layer, 50:1 metal reducer, small size, large torque. The maximum torque could arrive 42 kg.cm, stable and durable!

SPECIFICATION

Rated voltage: 12 V Gear reduction ratio: 50:1 D output shaft diameter: 6 mm No-load speed: 100 RPM @ 12 v No-load current: 0.17 A Rated speed: 93 RPM @ 12 v Current rating: 0.68 A Rated torque: 7 kg.cm Stall torque: 42 kg.cm Stall current: 2.19 A Power: 5W Weight: 210 g

- Rated Voltage: 6.0 V
- Motor Speed: 15000 RPM
- Gear Reduction Ratio: 210:1
- Reducer Length: 9.0 mm
- No-Load Speed: 75 rpm@6v
- No-Load Current: 60 mA
- Rated Torque: 1.4 kg.cm
- Rated Speed: 42 rpm@6V
- Current Rating: 170 mA

- Instant Torque:

8.ULTRASONIC SENSOR DATASHEET

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work: (1) Using IO trigger for at least 10us high level signal, (2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back. (3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning. Test distance = (high level time×velocity of sound (340M/S) / 2, λ

Wire connecting direct as following:

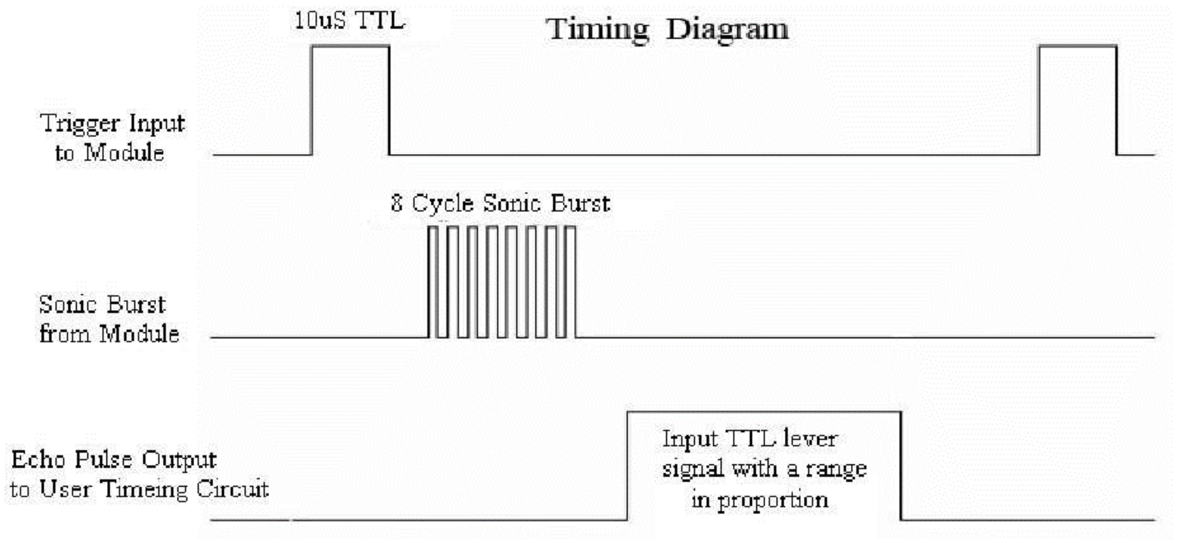
- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Supply

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo.

The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal.

Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.


10uS TTL
Timing Diagram
Trigger Input to Module
8 Cycle Sonic Burst
Sonic Burst from Module
Echo Pulse Output to User Timeing Circuit
Input TTL lever signal with a range in proportion

Attention:

 The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.

 When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

9.6V MOTOR DATASHEET

DC motor 6/9V

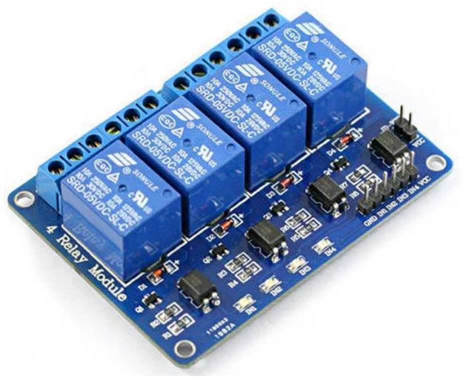
The 6V 250mA Brushed DC Motor is a standard '130 size' DC hobby motor. It comes with a wider operating range than most toy motors: from 4.5 to 9VDC instead of 1.5-4.5V. This range makes them perfect for controlling with an Adafruit Motor Shield, or with an Arduino where you are more likely to have 5 or 9V available than a high current 3V setting.

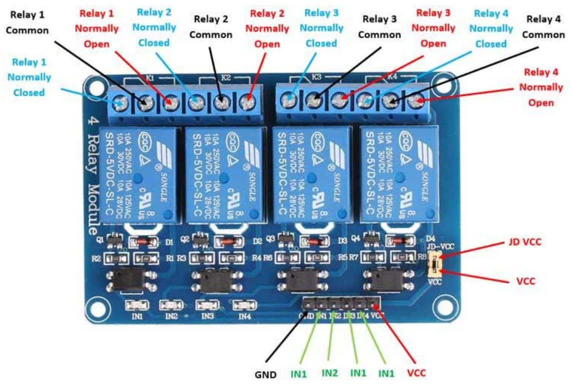
It will fit in most electronics that already have 130-size motors installed and there's two breadboard-friendly wires soldered on already for fast prototyping.

Specifications:

• Operating temperature: -10°C ~ +60°C • Rated voltage: 6.0VDC
• Rated load: 10 g*cm
• No-load current: 70 mA (max)
• Loaded current: 250 mA (max)
• Loaded speed: 4500 ±1500 rpm
• Starting voltage: 2.0
• Starting torque: 20 g*cm
• No-load speed: 9100 ±1800 rpm
• Stall current: 500mA max
• Body size: 27.5mm x 20mm x 15mm
• Shaft size: 8mm x 2mm (diameter)
• Weight: 17.5g

10.5V Four-Channel Relay Module





The **four-channel relay module** contains four 5V relays and the associated switching and isolating components, which makes interfacing with a microcontroller or sensor easy with minimum components and connections. The contacts on each relay are specified for 250VAC and 30VDC and 10A in each case, as marked on the body of the relays.

Four-Channel Relay Module Pinout

Pin Number	Pin Name	Description
1	GND	Ground reference for the module
2	IN1	Input to activate relay 1
3	IN2	Input to activate relay 2
4	IN3	Input to activate relay 3
5	IN4	Input to activate relay 4

6	V_{CC}	Power supply for the relay module
7	V_{CC}	Power supply selection jumper
8	JD-V_{CC}	Alternate power pin for the relay module

Four-Channel Relay Module Specifications

- Supply voltage – 3.75V to 6V
- Trigger current – 5mA
- Current when the relay is active - ~70mA (single), ~300mA (all four)
- Relay maximum contact voltage – 250VAC, 30VDC
- Relay maximum current – 10A

2.4 SOFTWARE DESCRIPTION

FRONT END SOFTWARE:

Arduino Bluetooth Control

Arduino Bluetooth Control is an application that allows you to control your arduino board (and similar boards) via Bluetooth, and so to create awesome and fully customized projects, with the new features available within the app.
The settings section allows you to adapt the application to your needs, through a very simple and intuitive interface.

The application also smartly remembers your bluetooth module and tries to connect automatically to the latest one you have used, so you won't have to select it every time you use it.

You can also use the application on your wearable device if you have any.

1.Metrics tool

This tool was optimized to receive data via the println() function of arduino, which allows special processing of the data received, like in the "Metrics" tool. It allows you to receive only numbers and fix alarms to get notified about the variations of the value received.Once the alarm triggered, a stop button shows up, allowing you to stop it.Besides you can activate the shaking mode, that will allow you to send data simply by shaking your phone.

2.Arrow keys

This tool provides direction buttons that can fully customized with the data to send, and the sensitivity, which allows to send continuously data to the board by maintainning long press on them.

3.Terminal

This tool is just a classic terminal that receives and sends data to the board, displayed with the timestamp corresponding to each action.

4.Buttons and slider

In portrait orientation, this tool provides 6 buttons fully customized, that will allow you to send specific data when pressed. When you rotate your device, a slider view shows up , to which you can set the range of the data to be sent.

5.Accelerometer

This tool permits you to interpret the gesture commands of your phone, and send the corresponding data to your board, and so , your phone can be the steering wheel of your robot. You can of course set the sensitivity of it through the settings interface.

6. Voice Control

Have you ever dreamed of talking to you robots ? well now your dream is becoming true ! With Arduino Bluetooth Control, you can customize your own vocal commands and use them to control all your microcontroller-based boards !

If you encounter any problem with the application,or need some specific feature to control you board, we will be pleased to help you out with it !

We also provides app customization service if you which to have your own custom bluetooth control app, tailored to your needs.

Version
Varies with device
Updated on
08-May-2021
Requires Android
Varies with device
Downloads
100,000+ downloads
Content rating
Rated for 3+ Learn more

Data safety

See details

Released on

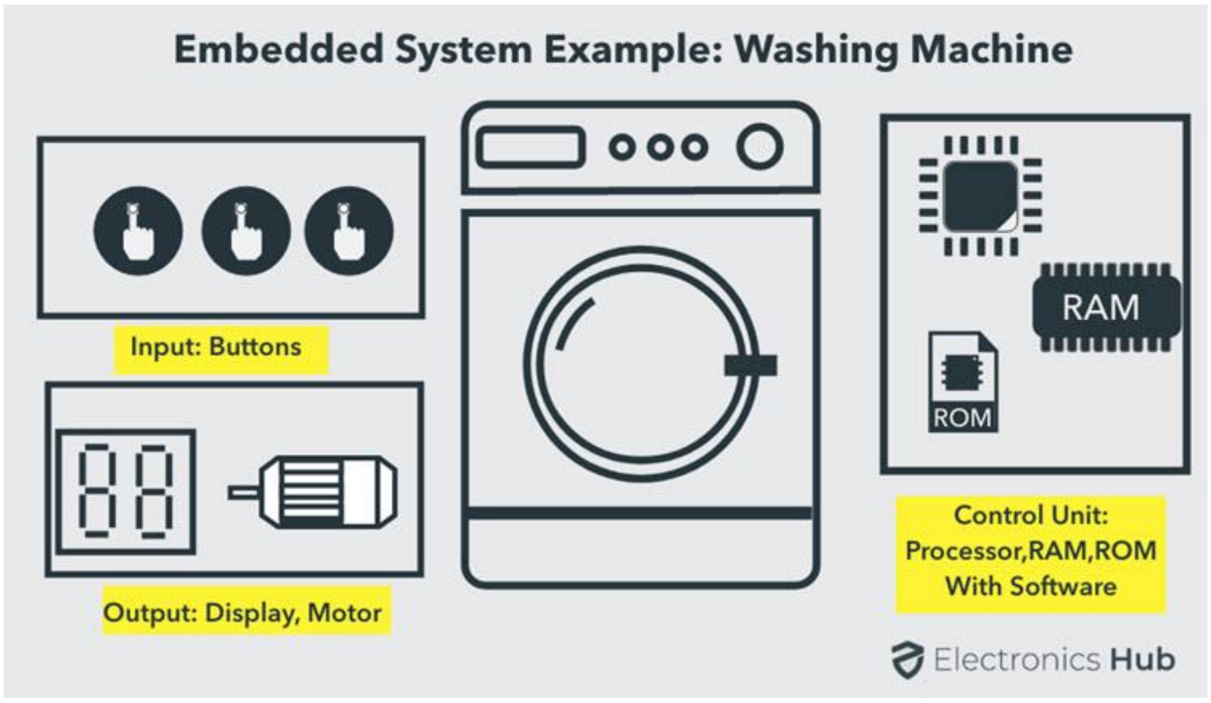
02-Aug-2015

BACK END SOFTWARE

What is an Embedded System?

An Embedded System can be best described as a system which has both the hardware and software and is designed to do a specific task. A good example for an Embedded System, which many households have, is a Washing Machine.

We use washing machines almost daily but wouldn't get the idea that it is an embedded system consisting of a Processor (and other hardware as well) and software.


Embedded System Example: Washing Machine
Input: Buttons
Output: Display, Motor
RAM
ROM
Control Unit: Processor,RAM,ROM With Software
Electronics Hub

It takes some inputs from the user like wash cycle, type of clothes, extra soaking and rinsing, spin rpm, etc., performs the necessary actions as per the instructions and finishes washing and drying the clothes. If no new instructions are given for the next wash, then the washing machines repeats the same set of tasks as the previous wash.

Embedded Systems can not only be stand-alone devices like Washing Machines but also be a part of a much larger system. An example for this is a Car. A modern day Car has several individual embedded systems that perform their specific tasks with the aim of making a smooth and safe journey.

Some of the embedded systems in a Car are Anti-lock Braking System (ABS), Temperature Monitoring System, Automatic Climate Control, Tire Pressure Monitoring System, Engine Oil Level Monitor, etc.

Introduction to Embedded C Programming Language

Before going in to the details of Embedded C Programming Language and basics of Embedded C Program, we will first talk about the C Programming Language.

The C Programming Language, developed by Dennis Ritchie in the late 60’s and early 70’s, is the most popular and widely used programming language. The C Programming Language provided low level memory access using an uncomplicated compiler (a software that converts programs to machine code) and achieved efficient mapping to machine instructions.

The C Programming Language became so popular that it is used in a wide range of applications ranging from Embedded Systems to Super Computers.

Embedded C Programming Language, which is widely used in the development of Embedded Systems, is an extension of C Program Language. The Embedded C Programming Language uses the same syntax and semantics of the C Programming Language like main function, declaration of datatypes, defining variables, loops, functions, statements, etc.

The extension in Embedded C from standard C Programming Language include I/O Hardware Addressing, fixed point arithmetic operations, accessing address spaces, etc.

Difference between C and Embedded C

There is actually not much difference between C and Embedded C apart from few extensions and the operating environment. Both C and Embedded C are ISO Standards that have almost same syntax, datatypes, functions, etc.

Embedded C is basically an extension to the Standard C Programming Language with additional features like Addressing I/O, multiple memory addressing and fixed-point arithmetic, etc.

C Programming Language is generally used for developing desktop applications, whereas Embedded C is used in the development of Microcontroller based applications

Features of Embedded C

1. It is only the extension of C language and mothing more.
2. It has source code format that depends upon the kind of microcontroller or microprocessor that have been being used.
3. Through embedded c high level optimization can done.
4. It is used in microprocessor or microcontroller applications.
5. It has limited resources for used, mean the embedded system have only memory location.
6. In embedded c constraints runs on real time and output is not available at operating system.
7. It only supports the adequate processor or controller.
8. In embedded c only pre-define program can run.

9. It requires a compiler of embedded c, which have the compatibility with all the embedded system resources.
10. Some of the examples of embedded system c application are digital camera, DVD and digital TV etc.
11. The major advantage of embedded c is its coding speed and size is very simple and easy to understand.

Advantages of embedded c

1. it is easier to understand.
2. It performs the same task all the time so there is no need of any hardware changing such as extra memory or space for storage.
3. It preforms only one task at one time mean it purposed the dedicated task
4. Hardware cost of embedded c systems are usually so much low.
5. Embedded applications are very suitable for industrial purposes.

Disadvantages of embedded c

1. It performs only one task at same time so many can’t perform the multi task at same time.
2. If we change the program then must be need to change the hardware.
3. It only supports the hardware system.
4. It also have issue of scalability mean it can’t easily have scaled up as scope change or demand.
5. It have a limitation such as limited memory or computer compatibility.

REQUIRED COMPONENTS

The following parts and tools are required for building this Eragon Floor cleaning Robot:

[FOR MOP PURPOSE]

Parts:

- 2x 100rpm Geared motors
- 2x Wheels for motors
- 2x 75rpm Plastic geared motors w/ wheels
- 1x Arduino UNO
- 1x HC-05/06 Bluetooth module
- 1x L293D motor driver board
- 1x 12V Sealed lead acid rechargeable battery
- 1x 12V Water pump
- 2x TIP31C/TIP122 NPN power transistors
- 2x Old CDs
- A paint roller w/ shaft
- Wiping cloth/ napkin/ old rags
- A needle and thread
- Vinyl tubing
- Rainbow wire
- Male-male/ female-female/ male-female jumper wires
- Male/female headers
- Perforated Cardboard
- Nuts and Screws
- Plywood base
- A 600ml plastic bottle

Tools:

- Soldering iron w/ solder
- Hot glue gun w/ glue sticks
- Drill
- Pliers
- Wire cutter/stripper

- Paper cutter
- Tape
- Safety equipment while working (Important!)

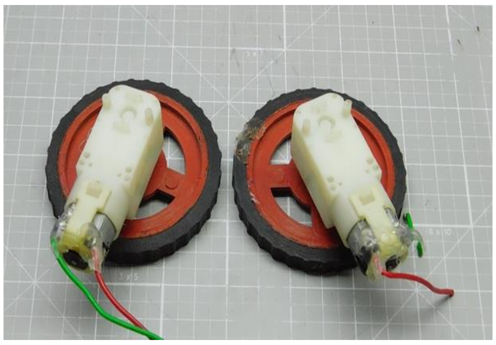
The following parts and tools are required for building this Eragon Floor cleaning Robot:

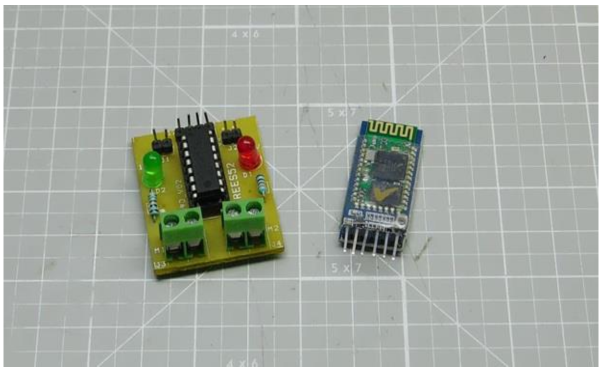
[VACCUME PURPOSE]

- Gear motor
- wheels
- 12v AC to DC Power supply
- Arduino Uno
- Arduino motor shield
- Servo motor
- Ultra sonic sensor
- 6v motor

Tools:

- Soldiers with wire
- Wires
- Cardboard
- Glue gun
- Glue sticks
- Favicol
- Super glue
- Cardboard cutter
- Pen knife cutter
- Black insulation tape
- Long Wooden stick





PLYWOOD, B

REDMI NOTE 10 PRO
SHOT BY NAREN
26/04/2022 18:02

2.5 PROJECT DESCRIPTION

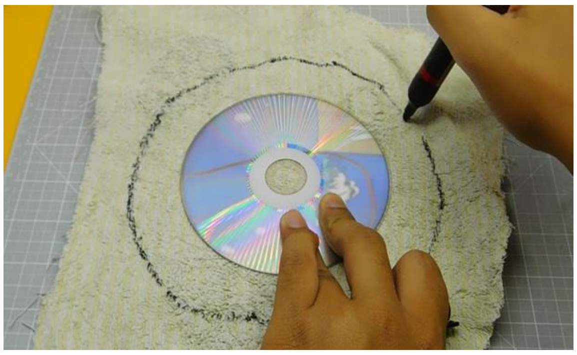
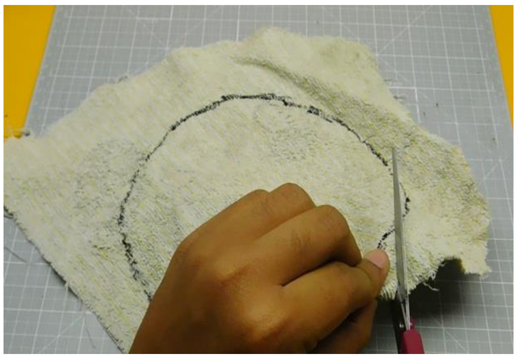
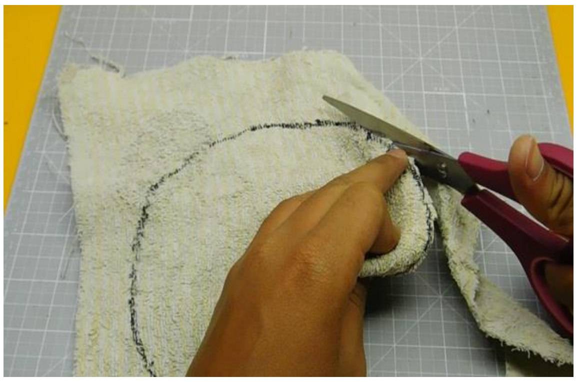
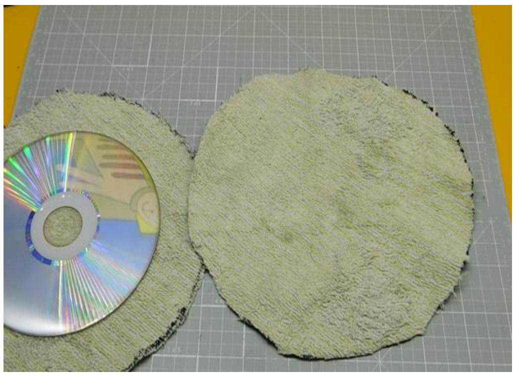
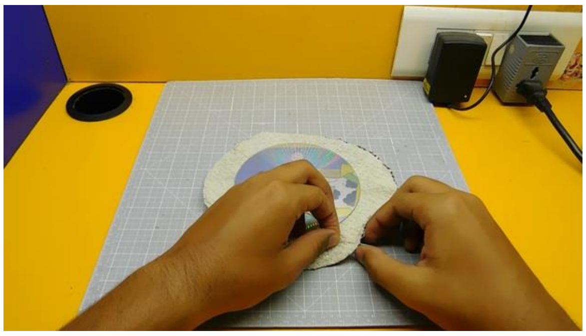
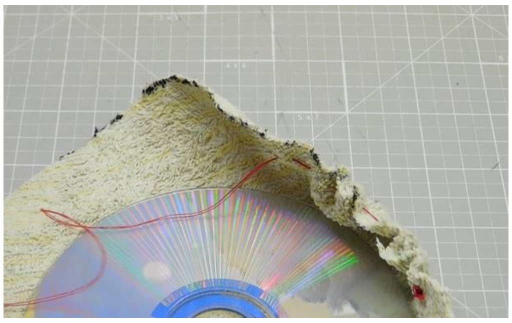
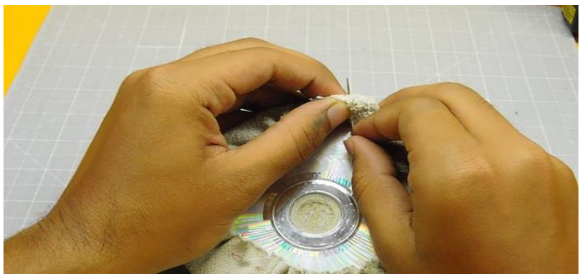
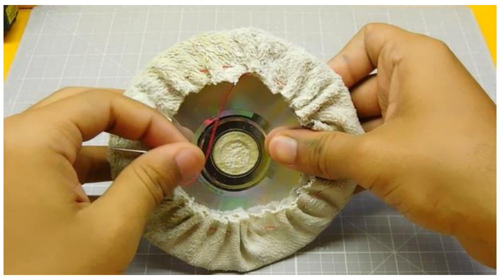
Get the Base Ready:

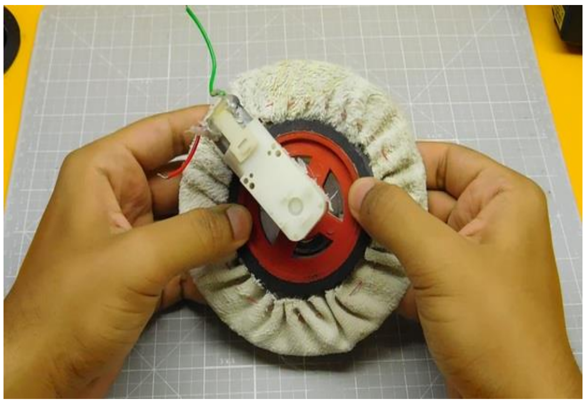
The rst step is to prepare the base on which the parts will be placed. For this, rst get a piece of plywood cut, measuring 12x9 inches. The base wasn't looking great at all, so to make it look a bit attractive, I painted it in white and orange stripes with black at the borders with acrylic paints.After this, drill two holes each at the back for both the motor clamps. Make proper measurements such that both should be parallel to each other.

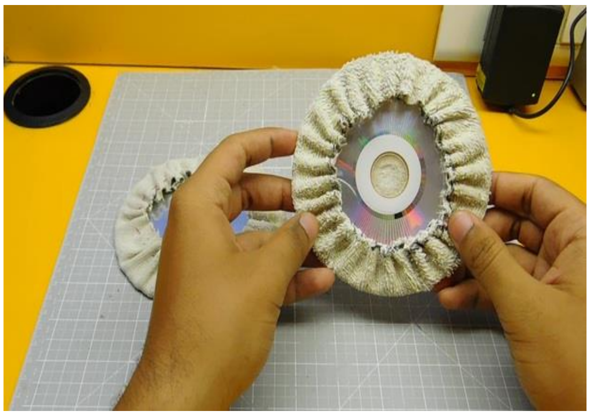
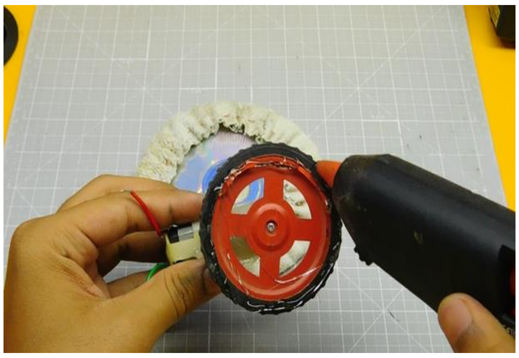
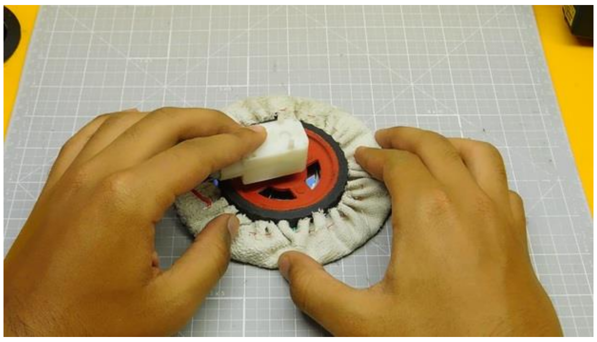
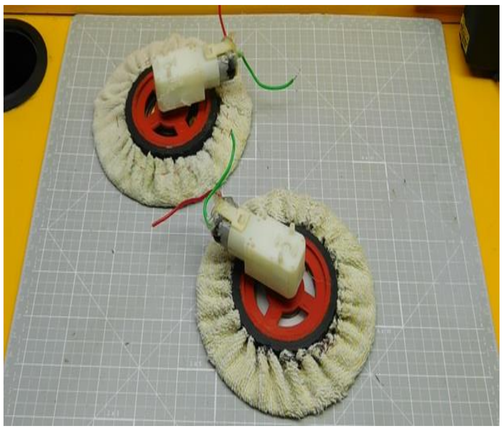
Fix them in place using some screws then attach the motors to the clamps.Make the Floor Moppers and Attach Them to the Base:I. To make cheap DIY circulars Orr moppers, you can use old compact disks along with a piece of cloth.First,, mark a circle on the cloth which should be bigger than the CD. Cut it using a pair of scissors.Take a needle and thread and start sewing and making folds to the cloth in such a way that it surrounds the entire CD.

(something I am very bad at)Refer to the images above or have a look at this video, or maybe this one.After you're done with all the folds, make 2-3 knots at the end and cut the left over thread.Repeat the same for the second CD.Attach wheels to both the plastic geared motors. Hot glue the wheels to the CDs.The moppers have to be placed in the front. Hot glue the motors in such a way that the cloth stays away from each other and the motors are at an equal distance from the sides.

Make sure the cloth properly touches the floor



The Water Supply Mechanism

This consists of a 12V water pump which carries the water from the reservoir and spills it near the mops on the floor. First mark, drill and x your pump in place.Take a 600ml empty plastic bottle and cut it into half using a paper cutter. Use the lower half and place it on the robot base using some hot glue.Take two pieces of rubber tubing. One will be connected
to the inlet of pump to take water from the reservoir from the pump and the second one will be used to take water from the pump to the floor. Adding straws to the outlet will be done later.The pump can be switched on/o via your smartphone just like other controls.

Connections

Now this is always the typical part. You must be accurate. For making it a bit easier, I always use jumper wires which can be swapped or removed any time.Before that, drill some holes and x your Arduino in place using some screws. Start by connecting the geared motors to the driver board. Solder some wire to the motor terminals and then connect them to the screw terminals of the driver circuit. The rest of the pins have to be connecting as per the following:

Signal 1 ---- D6 on Arduino

Signal 2 ---- D9 on Arduino

Signal 3 ---- D10 on Arduino

Signal 4 ---- D11 on Arduino

+5V ---- +5V on Arduino

Gnd ---- Gnd on Arduino

+12V (motors will move at this voltage) ---- to be connected to battery later

Next comes the bluetooth module. Connections are:

Vcc ---- +5V on Arduino

Gnd ---- Gnd on Arduino

Rx ---- Tx on Arduino

Tx ---- Rx on Arduino

Add a voltage divider to signal pins if you're afraid that the signal pins on arduino might burn.

The two mop motors have to be connected in parallel such that the left one runs anticlockwise and the right one turns
clockwise when seen from the front. Use heat shrink tubes to keep the connections safe. Solder the motor wires to the
transistor circuit as per the schematic given above. Similarly connect the water pump wires as well.
We will be supplying the 12V from the battery directly to the transistor circuit and then this 12V will go to the Vcc of
arduino and the motor driver circuit.
Connect the base of transistor two, controlling the mops to D5 on arduino and transistor one, controlling the pump to D4
on arduino. The common ground wire from all the motors has to be connected to the Gnd on arduino.
What remains now is the servo motor. The connections are:
Vcc ---- +5V on Arduino
Gnd ---- Gnd on Arduino
Signal ---- D3 on Arduino

Finishing

Keep all the circuit boards, wires, tubings in place with hot glue. It should look neat, the wires shouldn't entangle and the connections shouldn't break, which can be irritating.
Next, take two straws and cut them about 7cm in length. Crush and squeeze one end of both into the outlet pipe of the water pump. Bend both of them in opposite directions and glue them in place such that water ows from both the straws and falls just a little ahead of the rotating mop (look at the pictures). Don't forget to put some tape where the straws are connected so the water doesn't leak.

Upload the Code

Remove the Rxx and Txx cableless from Arduino before uploading!!

Connect the board to a pc and program it with the code given below. You can make necessary changes.

Make sure you set the correct COM port and Board under Tools.

After it's done, replace the Rx and Tx wires. You'll need to remove them everytime you upload the code.

Attach the Battery

Remove the Rxx and Txx cableless from Arduino before uploading!!

Connect the board to a pc and program it with the code given below. You can make necessary changes.

Make sure you set the correct COM port and Board under Tools.

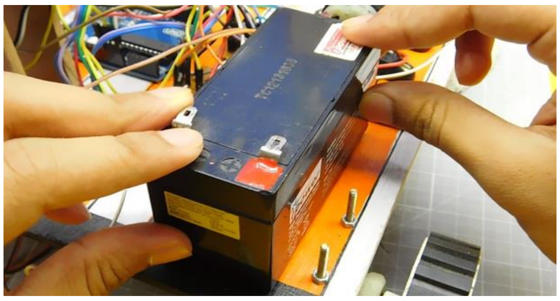
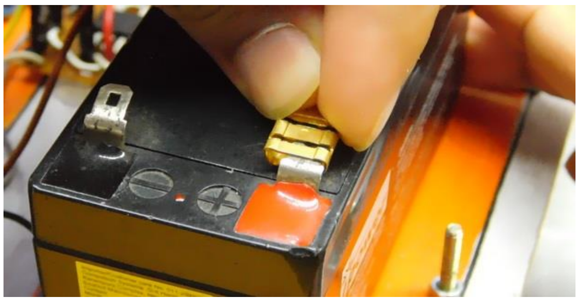
After it's done, replace the Rx and Tx wires. You'll need to remove them everytime you upload the code.

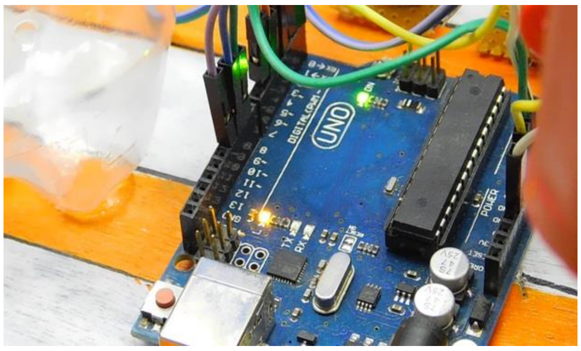
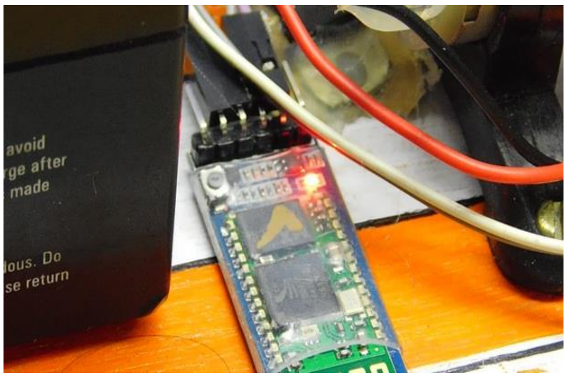
The voltage should be 12V. I would recommend a single lead acid battery or 3x Li-ion 18650, 3.7V each connected in series.

Use some double sided tape or hot glue to keep the battery pack in place. +Ve terminal will go to the transistor circuit from where it will go to Vcc on arduino and to the motor driver circuit. -Ve terminal will be connected to the common Gnd. Use proper battery connectors.

You can even add an On/O main switch.

If the power LED on the Arduino glows, it's all good. Immediately remove Arduino once if LEDs dim rapidly and recheck all the connections. Do not use a very high voltage else the regulator on the board may overheat.

Configure the App and Connect

Go to the google play store and get this app called 'Bluetooth Serial Controller' which lets you set your own control buttons and commands.

After opening the app, click on 'settings' and then 'visibility' Turn o visibility for buttons 5, 9, 12 as we won't be needing them.Next, go on the 'names' icon so set the display names for each button. Make them short, 3-4 letters. For example, look at the names I set above.

Now under the 'commands' option, set the following commands (without quotes) for each button (they are case sensitive):

Button 1 (FWD): 'F'

Button 2 (BCK): 'B'

Button 3 (LFT): 'L'

Button 4 (RGT): 'R'

Button 6 (MPON): 'M'

Button 7 (MPOF): 'm'

Button 8 (PMP): 'P'

Button 10 (RUP): 'U'

Button 11 (RDWN): 'u'

Button 13 (S1): '1'

Button 14 (S2): '2'

Button 15 (S3): '3'

Button 16 (S4): '4'

Under the 'stop commands' section in 'commands' itself, you need to set the following stop commands ONLY for the
buttons mentioned below:

Button 1: 'S'

Button 2: 'S'

Button 3: 'S'

Button 4: 'S'

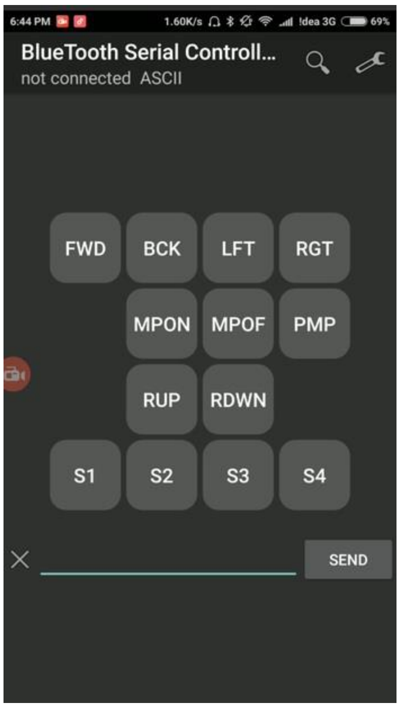
Button 8: 'p'

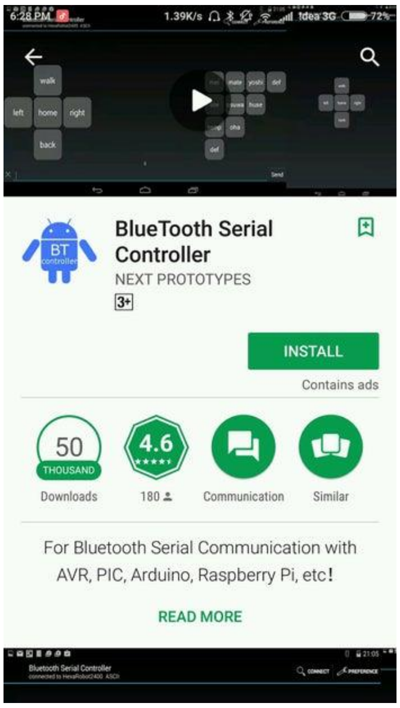
This means that example if button 2 isn't pressed, the command 'S' will be sent which will stop the robot.

To connect the robot, rst pair up the bluetooth module named 'HC-05' or other. Password will be '0000' or '1234'

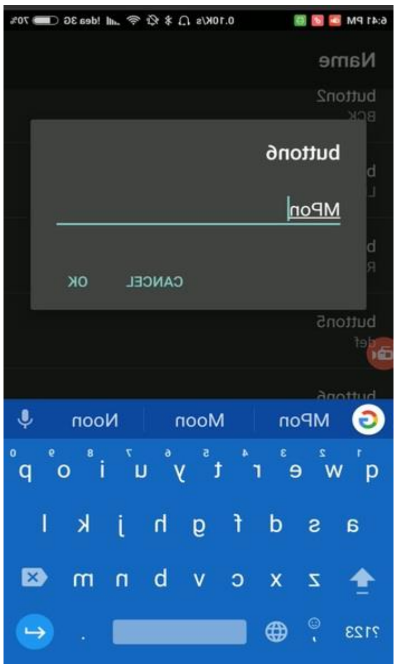
Then connect the paired up module via the app.

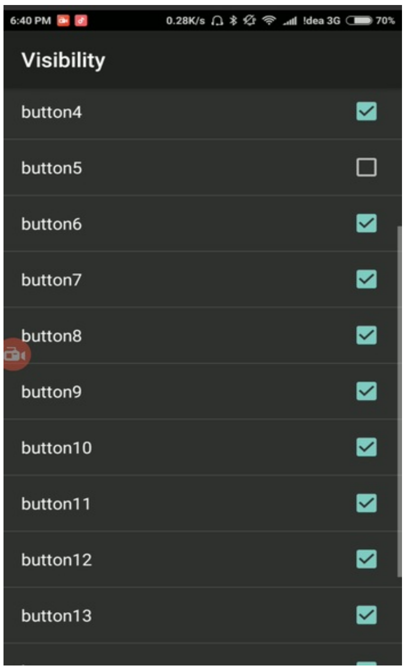
Press and check all the buttons on the app one by one.





For Bluetooth Serial Communication with AVR, PIC, Arduino, Raspberry Pi, etc!



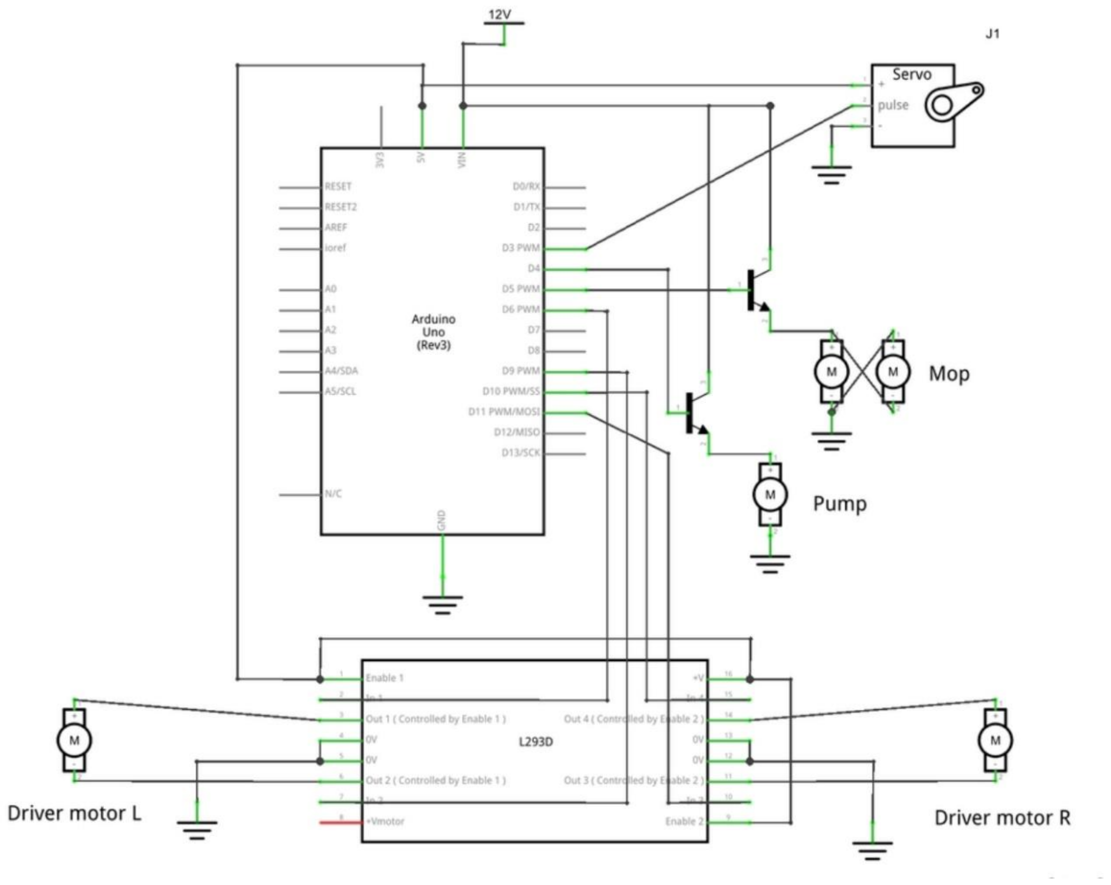


Visibility

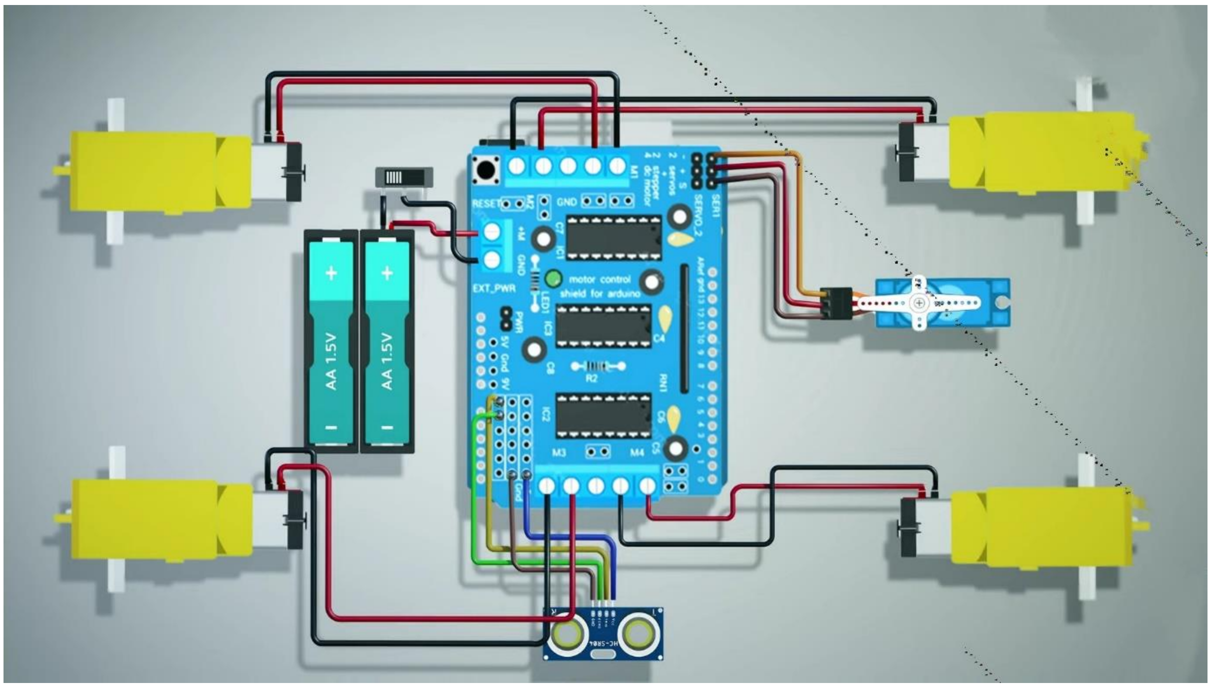
- button4 ✓
- button5 ☐
- button6 ✓
- button7 ✓
- button8 ✓
- button9 ✓
- button10 ✓
- button11 ✓
- button12 ✓
- button13 ✓

3 DESIGN

3.1 DIAGRAM FOR MOP



3.2 DIAGRAM FOR VACCUME



4. IMPLEMENTATION

4.1 SOURCE CODE

```
#include <AFMotor.h>

#include <NewPing.h>

#include <Servo.h>

#define TRIG_PIN A0

#define ECHO_PIN A1

#define MAX_DISTANCE 200

#define MAX_SPEED 170 // sets speed of DC  motors

#define MAX_SPEED_OFFSET 20



int r_motor_n = 10; //PWM control Right Motor +

int r_motor_p = 11; //PWM control Right Motor -

int l_motor_p = 9; //PWM control Left Motor -

int l_motor_n = 6; //PWM control Left Motor +

int pump = 4;

int mop = 5;

int vac=8;

int speedy = 160;

int incomingByte = 0; // for incoming serial data
```

```
NewPing sonar(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);

AF_DCMotor motor1(1, MOTOR12_1KHZ);
AF_DCMotor motor2(2, MOTOR12_1KHZ);
AF_DCMotor motor3(3, MOTOR34_1KHZ);
AF_DCMotor motor4(4, MOTOR34_1KHZ);
Servo myservo;

boolean goesForward=false;
int distance = 100;
int speedSet = 0;


void setup()
{
pinMode(r_motor_n, OUTPUT); //Set control pins to be outputs
pinMode(r_motor_p, OUTPUT);
pinMode(l_motor_p, OUTPUT);
pinMode(l_motor_n, OUTPUT);

pinMode(pump, OUTPUT);

pinMode(mop, OUTPUT);

digitalWrite(r_motor_n, LOW); //set both motors off for start-up

digitalWrite(r_motor_p, LOW);

digitalWrite(l_motor_p, LOW);

digitalWrite(l_motor_n, LOW);

digitalWrite(pump, LOW);

digitalWrite(mop, LOW);

digitalWrite(vac,LOW);

  myservo.attach(10);

  myservo.write(115);

  delay(2000);

  distance = readPing();

  delay(100);

  distance = readPing();

  delay(100);

  distance = readPing();

  delay(100);

  distance = readPing();
```

```
 delay(100);

 Serial.begin(9600);

}




void loop()

{


if (Serial.available() > 0)

{

incomingByte = Serial.read();

}


switch(incomingByte)

{


case 'S': // control to stop the robot

digitalWrite(r_motor_n, LOW);

digitalWrite(r_motor_p, LOW);
digitalWrite(l_motor_p, LOW);

digitalWrite(l_motor_n, LOW);

Serial.println("Stop");

incomingByte='*';

break;


case 'R': //control for right

analogWrite(r_motor_n, speedy);

digitalWrite(r_motor_p, LOW);

analogWrite(l_motor_p, speedy);

digitalWrite(l_motor_n, LOW);

Serial.println("right");

incomingByte='*';

break;



case 'L': //control for left

digitalWrite(r_motor_n, LOW);

analogWrite(r_motor_p, speedy);

digitalWrite(l_motor_p, LOW);
```

```
analogWrite(l_motor_n, speedy);

Serial.println("left");

incomingByte='*';

break;



case 'F': //control for forward

analogWrite(r_motor_n, speedy);

digitalWrite(r_motor_p, LOW);

digitalWrite(l_motor_p, LOW);

analogWrite(l_motor_n, speedy);

Serial.println("forward");

incomingByte='*';

break;



case 'B': //control for backward

digitalWrite(r_motor_n, LOW);

analogWrite(r_motor_p, speedy);

analogWrite(l_motor_p, speedy);
```

```
digitalWrite(l_motor_n, LOW);

Serial.println("back");

incomingByte='*';

break;



case 'P': // pump on

digitalWrite(pump, HIGH);

Serial.println("pump on");

incomingByte='*';

break;


case 'p': // pump off

digitalWrite(pump, LOW);

Serial.println("pump off");

incomingByte='*';

break;


case 'M':

digitalWrite(mop, HIGH); // mopper on
```

```
Serial.println("mopper on");

incomingByte='*';

break;



case 'm':

digitalWrite(mop, LOW); // mopper off

Serial.println("mopper off");

incomingByte='*';

break;



case 'V':

digitalWrite(vac,HIGH);

Serial.println("vaccum on");

incomingByte='*';

break;



case 'v':

digitalWrite(vac,LOW);

Serial.println("vaccum off");

incomingByte='*';
```

```
break;


case '1':

speedy = 155;

Serial.println("speed= 10");

incomingByte='*';

break;


case '2':

speedy = 185;

Serial.println("speed= 25");

incomingByte='*';

break;



case '4':

speedy = 255;

Serial.println("speed= 100");

incomingByte='*';

break;
```

```
delay(5000);
 }
int distanceR = 0;
 int distanceL =  0;
 delay(40);

 if(distance<=15)
 {
  moveStop();
  delay(100);
  moveBackward();
  delay(300);
  moveStop();
  delay(200);
  distanceR = lookRight();
  delay(200);
  distanceL = lookLeft();
  delay(200);
  if(distanceR>=distanceL)

  {

    Turn  Right();

    moveStop();

  }else

  {

    turnLeft();

    moveStop();

  }

 }else

 {

  moveForward();

 }

 distance = readPing();

}



int lookRight()

{

    myservo.write(50);

    delay(500);
    int distance = readPing();

    delay(100);

    myservo.write(115);

    return distance;

}



int lookLeft()

{

    myservo.write(170);

    delay(500);

    int distance = readPing();

    delay(100);

    myservo.write(115);

    return distance;

    delay(100);

}



int readPing() {

  delay(70);

  int cm = sonar.ping_cm();
  if(cm==0)

  {

    cm = 250;

  }

  return cm;

}



void moveStop() {

  motor1.run(RELEASE);

  motor2.run(RELEASE);

  motor3.run(RELEASE);

  motor4.run(RELEASE);

  }



void moveForward() {



 if(!goesForward)

  {

    goesForward=true;

    motor1.run(FORWARD);
    motor2.run(FORWARD);

    motor3.run(FORWARD);

    motor4.run(FORWARD);

   for (speedSet = 0; speedSet < MAX_SPEED; speedSet +=2) // slowly bring the speed up to
avoid loading down the batteries too quickly

   {

    motor1.setSpeed(speedSet);

    motor2.setSpeed(speedSet);

    motor3.setSpeed(speedSet);

    motor4.setSpeed(speedSet);

    delay(5);

   }

  }

}


void moveBackward() {

    goesForward=false;

    motor1.run(BACKWARD);

    motor2.run(BACKWARD);

    motor3.run(BACKWARD);

    motor4.run(BACKWARD);
  for (speedSet = 0; speedSet < MAX_SPEED; speedSet +=2) // slowly bring the speed up to
avoid loading down the batteries too quickly

  {

    motor1.setSpeed(speedSet);

    motor2.setSpeed(speedSet);

    motor3.setSpeed(speedSet);

    motor4.setSpeed(speedSet);

    delay(5);

  }

}


void turnRight() {

  motor1.run(FORWARD);

  motor2.run(FORWARD);

  motor3.run(BACKWARD);

  motor4.run(BACKWARD);

  delay(500);

  motor1.run(FORWARD);

  motor2.run(FORWARD);

  motor3.run(FORWARD);

  motor4.run(FORWARD);
```

```
}


void turnLeft() {
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
  delay(500);
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
```

4.2 SCREENSHOTS

5.TESTING

Embedded Testing

Embedded Testing is a testing process for checking functional and non-functional attributes of both software and hardware in an embedded system and ensuring that the final product is defect free. The main purpose of Embedded testing is to verify and validate whether the final product of embedded hardware and software fulfill the requirements of the client or not. Embedded Software testing checks and ensure the concerned software is of good quality and complies with all the requirements it should meet. Embedded software testing is an excellent approach to guarantee security in critical applications like medical equipment, railways, aviation, vehicle industry, etc. Strict and careful testing is crucial to grant software certification.

How to perform Embedded Software Testing

In general, you test for four reasons:

- To find bugs in software
- Helps to reduce risk to both users and the company
- Cut down development and maintenance costs
- To improve performance

In Embedded Testing, the following activities are performed:

1. The software is provided with some inputs.

2. A Piece of the software is executed.

3. The software state is observed, and the outputs are checked for expected properties like whether the output matches the expected outcome, conformance to the requirements and absence of system crashes.

Embedded Software Testing Types

Fundamentally, there are five levels of testing that can be applied to embedded software

Software Unit Testing

The unit module is either a function or class. Unit Testing is performed by the development team, primarily the developer and is usually carried out in a peer-review model. Based on the specification of the module test cases are developed.

Integration Testing

Integration testing can be classified into two segments:

1. Software integration testing
2. Software/hardware integration testing.

In the end, the interaction of the hardware domain and software components are tested. This can incorporate examining the interaction between built-in peripheral devices and software.

Embedded software development has a unique characteristic which focuses on the actual environment, in which the software is run, is generally created in parallel with the software. This causes inconvenience for testing since comprehensive testing cannot be performed in a simulated condition.

System Unit Testing

Now the module to be tested is a full framework that consists of complete software code additionally all real-time operating system (RTOS) and platform-related pieces such as interrupts, tasking mechanisms, communications and so on. The Point of Control protocol is not anymore a call to a function or a method invocation, but rather a message sent/got utilizing the RTOS message queues.

System resources are observed to evaluate the system's ability to support embedded system execution. For this aspect, gray-box testing is the favored testing method. Depending on the organization, system unit testing is either the duty of the developer or a dedicated system integration team.

System Integration Testing

The module to be tested begins from a set of components within a single node. The Points of Control and Observations (PCOs) are a mix of network related communication protocols and RTOS, such as network messages and RTOS events. Additionally to a component, a Virtual Tester can likewise play the role of a node.

System Validation Testing

The module to be tested is a subsystem with a complete implementation or the complete embedded system. The objective of this final test is to meet external entity functional requirements. Note that an external entity either be a person, or a device in a telecom network, or both.

Challenges: Embedded Software Testing

Some of the challenges that one can face during Embedded software testing:

Hardware Dependency

Hardware dependency is among the main difficulties faced during embedded software testing because of limited access to hardware. However, Emulators and Simulators may not precisely represent the behavior of the actual device and could give a wrong sense of system performance and application's usability.

Open Source Software

The majority of the embedded software components are open source in nature, not created in-house and absence of complete test available for it. There is a wide range of test combinations and resulting scenarios.

Software vs. Hardware Defects

Another aspect is when software is being developed for a freshly created hardware, during this process high ratio of hardware defects can be identified. The found defect is just not limited to software. It may be related to hardware also.

Reproducible Defects

Defects are harder to reproduce/recreate in the case of the embedded system. That enforces the embedded testing procedure to value every defect occurrence substantially higher than in a standard case, other than to gather as much data as could sensibly be required to alter the system to find the foundation of the defect.

Continuous Software Updates

Embedded systems require regular software updates like the kernel upgrade, security fixes, different device drivers, etc. Constraints identified with the software updates influence makes bug identification difficult. Additionally, it increases the significance of build and deployment procedure.

6. CONCLUSION

A cheaper and user-friendly Vacuum Cleaner robot can be developed with different modes of control (Manual) using an Arduino Board with more electronic functionality. Battery monitoring, self-charging, lighter body weight and setting motor on/off time manually are the future scope of this project.

7. BIBLIOGRAPHY

ONLINE REFERENCE

www.arduino.com for embedded purpose

https://www.instructables.com/CleanSweep-the-Floor-Cleaning-Robot/