CHAPTER -1
INTRODUCTION TO PROJECT

1.1 OBJECTIVE

Technology makes cleaning more intelligent and accessible. Future energy sources will become saturated and run out. Instead of using nonrenewable energies, consider solar power. Today, practically every field uses solar energy. Cleaning is one household task that never becomes obsolete and welcomes new technology. Floors are cleaned with broomsticks, vacuum cleaners, and advanced robot cleaners like Roomba.

Middle-age vacuum cleaners and even advanced robotic cleaners are too expensive for low and middle-class consumers. Traditional vacuum cleaners reduce the amount of human energy needed to clean floors, but the user must remain behind the machine to direct the suction pipe to dusty areas. These vacuum cleaners are likewise plugins, meaning they can only be used while plugged in. Solar energy is used to charge the battery, which powers the driving circuit.

Automatic floor cleaner is a compact robotics system which provides floor cleaning service in room and big offices reducing human labor. Basically as a robot it eliminates human error and provide cleaning activity with much more efficiency. If we clean the floor manually then there is a possibility that the operator will leave some portion of the floor. Also due to manual labor involved this is time consuming and irritating to clean the floor.

Also in big offices floor area is very huge and the people involved there for cleaning purpose cannot clean it much more . This is where the robot comes as an advantage. Also the robot is small and compact in size So we can carry it and place it wherever we can on the house. Also in industries the robot is very cost effective as compared to manual labor involved. The flexibility, time saving and efficiency make the robot a clean choice for cleaning the floor.

1.2 INTRODUCTION

Cleaning is important work approximate every place . Sometimes this is easy and sometimes difficult . Sometimes we assigned people for purpose of cleaning and pay money and sometimes cleaning is required in areas where presence of living being dangerous so we cannot assigned living being in every place . Some places are so that have a large floor areas in that place for cleaning purpose we need more than one person so we required some technique to compensate this problems . In advancement of science a robot come in light but in operate by a personnel . To avoid this limitations of personnel we require more technologies.

Cleaning is important work approximate every place . Sometimes this is easy and sometimes difficult . Sometimes we assigned people for purpose of cleaning and pay money and sometimes cleaning is required in areas where presence of living being dangerous so we cannot assigned living being in every place . Some places are so that have a large floor areas in that place for cleaning purpose we need more than one person so we required some technique to compensate this problems . In advancement of science a robot come in light but in operate by a personnel . To avoid this limitations of personnel we require more technologies.

Automation is a great solution of this problem . So we make an autonomous floor cleaning robot .Ultrasonic sensor is the most important component for autonomous floor cleaning robot because ultrasonic sensor works as eyes of robot . Ultrasonic sensor useful for turning of robot by sensing the obstacle or wall . Sensing distance range set by programming . In this range robot sense the obstacle and turn back .Cleaning is Important work inexact each spot. At times this is simple and once in a while troublesome.

At times we allocated individuals for reason for cleaning and pay cash and once in a while cleaning is needed in regions where presence of living being hazardous so we can't relegate living being in each spot. A few spots are so that have a huge floor territory in that place for cleaning reason we need more than one individual so we required some method to repay these issues. In headway of science a robot come in light however it works by a faculty. To keep away from this limit of faculty we require more innovations. Computerization is an extraordinary arrangement of this issue. So, we make a self-governing floor cleaning robot that worked by web of things and Arduino programming. Families of today are getting more astute and furthermore more mechanized. Home robotization conveys accommodation and makes more opportunity for individuals. Homegrown robots are entering the homes and individuals' everyday lives, yet it is yet a moderately new and juvenile market. Vacuum Robot will have a few measures that are easy to use.

1.2.1 EMBEDDED SYSTEMS

Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have real time performance constraints that must be met, for reason such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

An embedded system is not always a separate block - very often it is physically built-in to the device it is controlling. The software written for embedded systems is often called firmware, and is stored in read-only memory or flash convector chips rather than a disk drive. It often runs with limited computer hardware resources: small or no keyboard, screen, and little memory.

will be in high demand. Unfortunately, there are few adorable environments available for development and classroom use, so students often do not learn about these technologies during hands-on lab exercises. The communication mediums were twisted pair, optical fiber, infrared, and generally wireless radio.

1.3 PROBLEM STATEMENT

Traditional vacuum cleaners rely on grid electricity, which contributes to carbon emissions and increases energy costs for users. Additionally, these cleaners often lack smart features, leading to inefficient cleaning and user inconvenience. To address these challenges, there's a pressing need to develop a smart vacuum cleaner powered by solar energy. This cleaner should integrate advanced navigation systems, efficient cleaning algorithms, and user-friendly controls to optimize cleaning performance while reducing environmental impact and operating costs. However, designing such a product requires overcoming technical, design, and usability challenges, making it essential to formulate a comprehensive problem statement to guide the development process effectively.

1.4 EXISTING SYSTEM

Before the advent of solar-powered vacuum cleaners, conventional smart vacuum systems relied primarily on electric power from traditional grid sources. These systems typically consisted of autonomous or remotely controlled vacuum cleaners equipped with advanced sensors and navigation technology to efficiently clean indoor spaces. They operated by consuming electricity from wall outlets, often requiring periodic recharging or docking at a charging station.

While these electric-powered smart vacuum cleaners provided convenience and automation in household cleaning tasks, they were limited by their dependence on grid electricity, which could contribute to energy costs and environmental impact. However, with the emergence of solar-powered vacuum cleaners, there arose a new paradigm of eco-friendly cleaning solutions that offered greater sustainability and reduced reliance on conventional energy sources.

1.5 PROPOSED SYSTEM

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, a growth is predicted and the adoption of domestic robots is evolving.

The purpose of this project is to design and implement a Vacuum Robot Autonomous .autonomous Robot is designed to make cleaning process become easier rather than by using manual vacuum. The main objective of this project is to design and implement AUTONOMOUS ROBOT using obstacle sensor prototype .

Technology makes cleaning more intelligent and accessible. Future energy sources will become saturated and run out. Instead of using nonrenewable energies, consider solar power. Today, practically every field uses solar energy. Cleaning is one household task that never becomes obsolete and welcomes new technology.

Floors are cleaned with broomsticks, vacuum cleaners, and advanced robot cleaners like Roomba. Middle-age vacuum cleaners and even advanced robotic cleaners are too expensive for low and middle-class consumers.

Traditional vacuum cleaners reduce the amount of human energy needed to clean floors, but the user must remain behind the machine to direct the suction pipe to dusty areas. These vacuum cleaners are likewise plugins, meaning they can only be used while plugged in. Solar energy is used to charge the battery, which powers the driving circuit. This cleaner uses Arduino-Uno and Motor driver L293D. This designed solar floor cleaner is driven autonomously with sensor communication by recognizing obstructions and avoiding them.

Another Bluetooth module lets the user steer the cleaner to any desired area. This module accepts commands and drives the model. This household and outdoor cleaner provide easy and rapid cleaning. It avoids regular vacuum cleaners' 'plugin and use' method by self-moving and cleaning concurrently. Thus, automated solar floor cleaners have efficient cleaning benefits and uses.

1.6 SYSTEM ARCHITECTURE

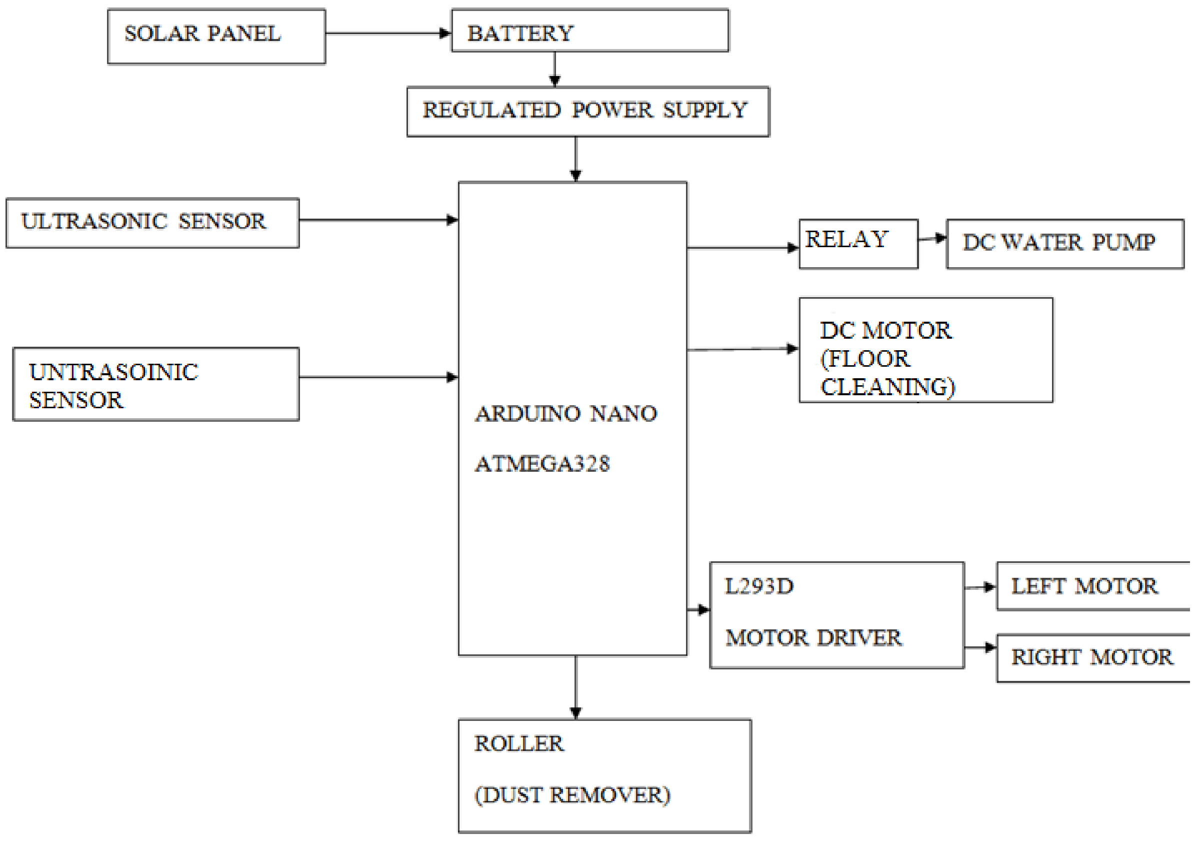


Fig 1.1 : Block diagram of the system

1.7 REQUIREMENTS:

Requirements for a smart vacuum cleaner utilizing solar panels would encompass various aspects to ensure efficiency, sustainability, and user-friendliness. Here are some key requirements:

- **Solar Panels Integration**:

  Incorporate high-efficiency solar panels into the design of the vacuum cleaner to efficiently capture solar energy. A system, typically using batteries, is needed to store excess solar energy for times when sunlight is not available, ensuring continuous operation. Install solar panels on the vacuum cleaner to collect sunlight for power.
- **Energy Storage System**:

  Implement a reliable energy storage system, such as lithium-ion batteries, to store excess solar energy for continuous operation, especially during low-light conditions or when the vacuum cleaner is in shaded areas. Include batteries to store extra solar energy for use when sunlight is unavailable.

- **Smart Charging Management**:

  Develop intelligent charging management systems to optimize the utilization of solar energy and battery storage, ensuring efficient charging and prolonged battery life. This involves technologies to optimize how the vacuum cleaner charges using solar energy, ensuring efficient charging and prolonging battery life.

- **Efficient Cleaning Mechanism**:

  Design a powerful and efficient cleaning mechanism that effectively removes dust, dirt, and debris from various surfaces, including carpets, hardwood floors, and outdoor spaces. The vacuum cleaner should be designed to effectively clean different surfaces, such as carpets and hardwood floors, as well as outdoor spaces, removing dust and debris efficiently.

- **Navigation and Mapping**:

  Incorporate smart navigation and mapping technology, such as sensors and cameras, to enable the vacuum cleaner to navigate obstacles, avoid collisions, and efficiently cover cleaning areas. Sensors and cameras help the vacuum cleaner navigate around obstacles and efficiently cover cleaning areas without human intervention.

  - **Autonomous Operation**:

    Enable autonomous operation with features like scheduling, automatic charging, and self-emptying capabilities, allowing the vacuum cleaner to perform cleaning tasks without constant user intervention.

- **Connectivity and Control**:

  Integrate connectivity features, such as Wi-Fi or Bluetooth, to enable remote control, monitoring, and firmware updates via smartphone apps or voice assistants Users should be able to control the vacuum cleaner remotely using smartphone apps or voice commands, and it should also receive updates over the internet..

- **Energy Efficiency Optimization**:

  Implement energy-efficient components and algorithms to minimize power consumption and maximize the utilization of solar energy, thereby enhancing overall sustainability. Components and algorithms should be designed to minimize energy consumption and make the best use of solar power available.

- **Durability and Weather Resistance**:

Ensure durability and weather resistance to withstand outdoor environments and varying weather conditions, allowing the smart vacuum cleaner to operate reliably in both indoor and outdoor settings. The vacuum cleaner needs to be sturdy and weatherproof to withstand outdoor conditions and continue operating reliably.

- **User Interface and Feedback**:

Provide an intuitive user interface with feedback mechanisms, such as LED indicators or mobile notifications, to inform users about the cleaning progress, battery status, and any potential issues. The vacuum cleaner should provide clear information to users about its status, progress, and any issues it encounters during operation.

CHAPTER 2

LITERATURE SURVEY OF THE PROJECT WORK

- **Literature Survey 1:**

"Autonomous Solar-Powered Robot Vacuum Cleaner" by Smith et al. (2018).This study presents the design and development of an autonomous vacuum cleaner powered by solar energy. The authors discuss the integration of solar panels into the device, including placement and efficiency considerations. They also evaluate the performance of the solar-powered vacuum cleaner in real-world conditions, assessing factors such as battery life, cleaning efficiency, and sunlight exposure.

- **Literature Survey 2:**

"Solar-Powered Autonomous Vacuum Cleaners: A Review" by Johnson et al. (2020)Johnson et al. provide a comprehensive review of the existing literature on solar-powered autonomous vacuum cleaners. They analyze various research efforts and commercial products in this field, discussing the technological advancements, energy efficiency, and environmental benefits associated with solar-powered cleaning devices. The review also addresses challenges such as battery storage capacity, sunlight exposure optimization, and cost-effectiveness.

- **Literature Survey 3:**

"Energy Management Strategies for Solar-Powered Autonomous Vacuum Cleaners" by Chenetal. (2019). This paper focuses on energy management strategies specifically tailored for solar-powered autonomous vacuum cleaners. The authors propose and evaluate different approaches for optimizing energy usage, including dynamic scheduling algorithms, battery charging optimization, and power management techniques. The study aims to improve the overall efficiency and autonomy of solar-powered cleaning devices by maximizing energy harvested from solar panel.

CHAPTER 3

PROJECT METHODOLOGY AND ANALYSIS

In this system ,the robot clean the total floor ,if obstacle detected by the ultrasonic sensor it avoid the obstacle move beside of it ,infront this set the dust cleaner and on the bot have been arranged the water pump, when the bot start parallelly dust cleaner and water pump will starts then floor cleaning scrubber will rotates under the bot.

3.1 WORKING PROCEDURE:

The smart vacuum cleaner project integrates a range of essential hardware components to ensure efficient and autonomous operation. These components include transformers, rectifiers, filters, capacitors, voltage regulators, IC 7805, Arduino Nano, ultrasonic sensors, transducers, L293D motor driver, solar panel, battery, and a DC motor.

The heart of the system lies in its ability to harness solar energy, achieved through the incorporation of a solar panel. This panel is connected to a voltage regulator to maintain a stable power supply to the system. Additionally, a charging circuit is implemented to store surplus solar energy in the battery, ensuring uninterrupted operation even during periods of low sunlight.

The functionality of the smart vacuum cleaner is greatly enhanced by its sensing capabilities. Ultrasonic sensors are strategically placed to detect obstacles and map out the cleaning path. These sensors provide crucial input data to the microcontroller, enabling intelligent decision-making during operation.

Furthermore, transducers are utilized for feedback and communication purposes, facilitating interaction with users or external devices.Motor control is achieved through the integration of a DC motor, which is managed by the L293D motor driver. This driver allows for precise control of motor speed and direction, essential for effective vacuuming and maneuvering around obstacles.

The microcontroller, typically an Arduino Nano or Arduino with ATmega328, serves as the brain of the system. It is programmed using the Arduino IDE to handle sensor data processing, motor control algorithms, and communication protocols.One of the standout features of the smart vacuum cleaner is its smart functionality. This includes advanced capabilities such as obstacle avoidance, automatic recharging when the battery level is low, and the ability to schedule cleaning tasks.

These features are made possible by sophisticated algorithms implemented in the microcontroller firmware.To enhance user experience and convenience, the smart vacuum cleaner can be controlled remotely via Bluetooth connectivity. This allows users to initiate cleaning cycles, monitor status, and adjust settings using a mobile device. Overall, meticulous testing and calibration procedures ensure that the system operates reliably and efficiently under various conditions, making it a valuable addition to any household or commercial space.

3.2 FLOWCHART:

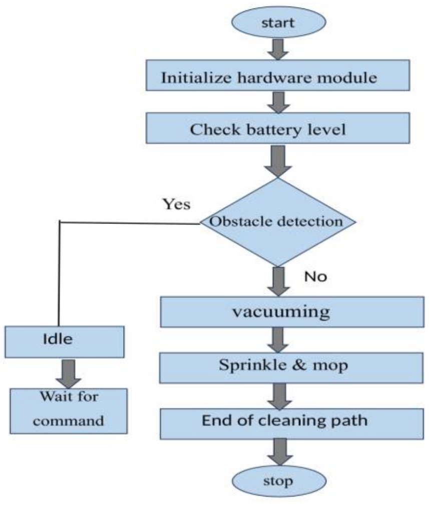


Fig3.1: Flowchart

The cleaning robot operation begins with the initialization of its hardware module, where all components are powered on and sensors are activated. Following this, the robot checks its battery level to ensure it has sufficient charge to complete the cleaning task. If the battery is low, the robot proceeds to charge itself before continuing. Once the battery check is complete, the robot engages in obstacle detection, scanning its environment for any obstructions in its cleaning path. If obstacles are detected, the robot navigates around them to avoid collisions.

Subsequently, the robot commences the cleaning process by activating its vacuuming mechanism, which collects dirt and debris from the floor surface. Once vacuuming is complete, the robot proceeds to sprinkle cleaning solution and mop the floor, ensuring a thorough cleaning. As it progresses along its cleaning path, the robot continually monitors for the end of the cleaning area. Once the entire area has been covered, the robot stops its operation, ready to return to its charging dock if necessary. This systematic approach ensures efficient and effective cleaning performance by the robot, providing a hassle-free solution for maintaining cleanliness in indoor environments.

After completing its cleaning cycle, the robot stands ready for its next task, its sensors still alert to any potential obstacles or hazards in its surroundings. If the cleaning path has been entirely covered and no further tasks are required, the robot remains idle until prompted to start another cleaning session or to return to its charging station. This systematic workflow ensures that the cleaning robot operates smoothly and autonomously, efficiently maintaining cleanliness without the need for constant human intervention. With its ability to navigate obstacles, monitor battery levels, and adapt to different surfaces, the cleaning robot offers a convenient and effective solution for automated cleaning in various indoor environments, from homes to commercial spaces.

In addition to its core cleaning functions, the robot can be equipped with smart features such as scheduling capabilities, allowing users to set specific cleaning times according to their preferences. Moreover, advanced models may integrate with smart home systems, enabling remote control and monitoring via smartphone apps or voice assistants.

As technology continues to evolve, these cleaning robots are becoming increasingly sophisticated, with improved navigation algorithms and enhanced cleaning capabilities, ensuring they remain at the forefront of automated cleaning solutions. With their efficiency, convenience, and ability to adapt to diverse environments, cleaning robots represent a significant advancement in home and facility maintenance, promising a future where household chores are effortlessly managed with the touch of a button.

CHAPTER 4
IMPLEMENTATION

4.1 HARDWARE DESCRIPTION

- **POWER SUPPLY**

All digital circuits require regulated power supply. In this article we are going to learn how to get a regulated positive supply from the mains supply.

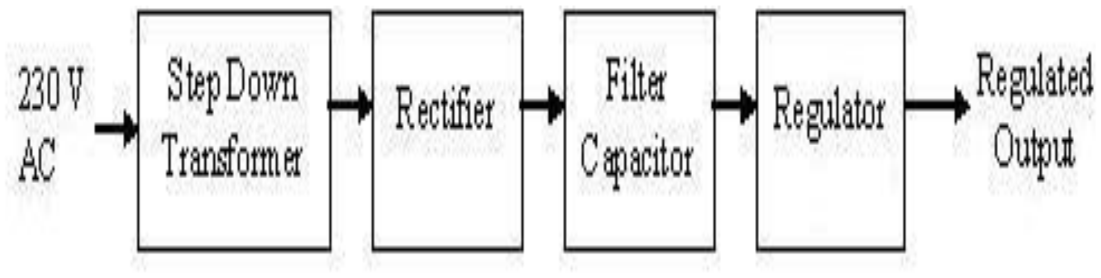


Fig4.1: Basic block diagram of a fixed regulated power supply.

- **TRANSFORMER**

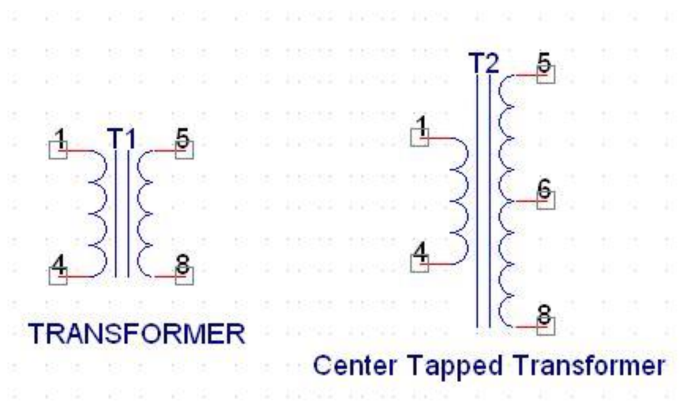


Fig 4.2:Transformer

A transformer consists of two coils also called as “WINDINGS” namely PRIMARY & SECONDARY.

They are linked together through inductively coupled electrical conductors also called as CORE. A changing current in the primary causes a change in the Magnetic Field in the core & this in turn induces an alternating voltage in the secondary coil. If load is applied to the secondary then an alternating current will flow through the load. If we consider an ideal condition then all the energy from the primary circuit will be transferred to the secondary circuit through the magnetic field.

So $I_pV_p = I_sV_s$ $P_{primary} = P_{secondary}$

A rectifier is a device that

converts an AC signal into DC signal. For rectification purpose we use a diode, a diode is a device that allows current to pass only in one direction i.e. when the anode of the diode is positive with respect to the cathode also called as forward biased condition & blocks current in the reversed biased condition.

Rectifier can be classified as follows:

1. Half Wave rectifier.

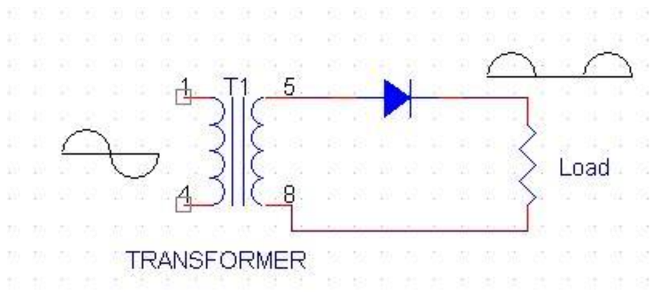


Fig 4.3: Half wave rectifier

This is the simplest type of rectifier as you can see in the diagram a half wave rectifier consists of only one diode. When an AC signal is applied to it during the positive half cycle the diode is forward biased & current flows through it. But during the negative half cycle diode is reverse biased & no current flows through it. Since only one half of the input reaches the output, it is very inefficient to be used in power supplies

2. Full wave rectifier.

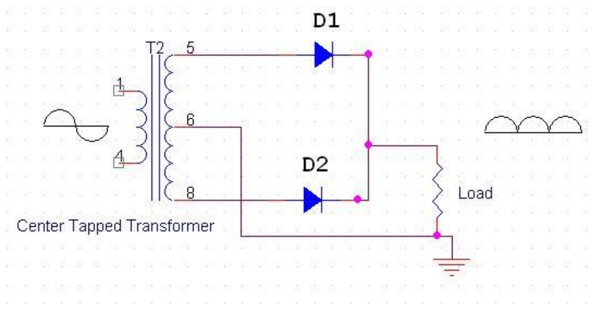


Fig 4.4: Full wave rectifier

Half wave rectifier is quite simple but it is very inefficient, for greater efficiency we would like to use both the half cycles of the AC signal. This can be achieved by using a center tapped transformer i.e. we would have to double the size of secondary winding & provide connection to the center. So during the positive half cycle diode D1 conducts & D2 is in reverse biased condition. During the negative half cycle diode D2 conducts & D1 is reverse biased. Thus we get both the half cycles across the load.

One of the disadvantages of Full Wave Rectifier design is the necessity of using a center tapped transformer, thus increasing the size & cost of the circuit. This can be avoided by using the Full Wave Bridge Rectifier

- **Bridge Rectifier**

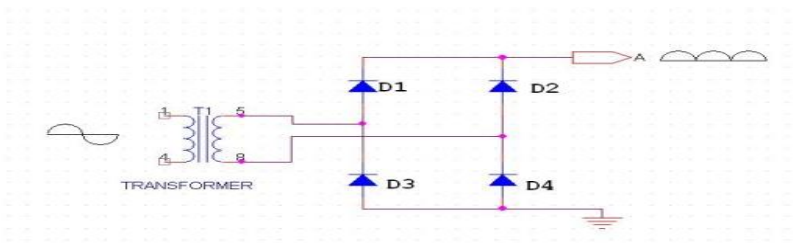


Fig 4.5 Bridge rectifier

As the name suggests it converts the full wave i.e. both the positive & the negative half cycle into DC thus it is much more efficient than Half Wave Rectifier & that too without using a center tapped transformer thus much more cost effective than Full Wave Rectifier.

Full Bridge Wave Rectifier consists of four diodes namely D1, D2, D3 and D4. During the positive half cycle diodes D1 & D4 conduct whereas in the negative half cycle diodes D2 & D3 conduct thus the diodes keep switching the transformer connections so we get positive half cycles in the output.

- **FILTER CAPACITOR**

Even though half wave & full wave rectifier give DC output, none of them provides a constant output voltage. For this we require to smoothen the waveform received from the rectifier. This can be done by using a capacitor at the output of the rectifier this capacitor is also called as “FILTER CAPACITOR” or “SMOOTHING CAPACITOR” or “RESERVOIR CAPACITOR”. Even after using this capacitor a small amount of ripple will remain.
We place the Filter Capacitor at the output of the rectifier the capacitor will charge to the peak voltage during each half cycle then will discharge its stored energy slowly through the load while the rectified voltage drops to zero, thus trying to keep the voltage as constant as possible.

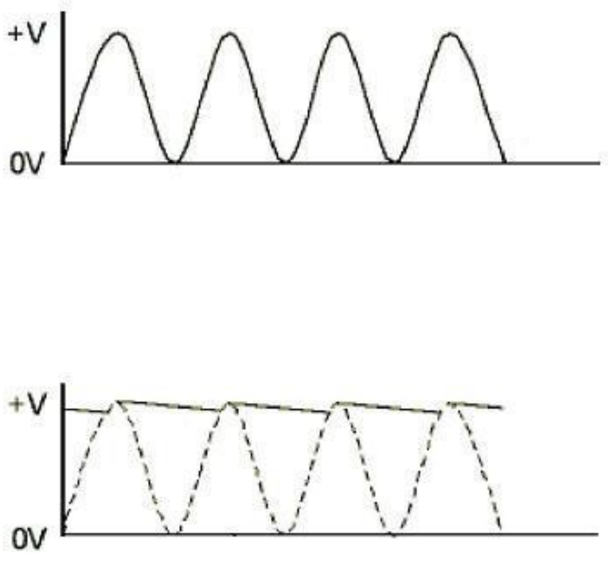


FIG 4.6 : Positive And Negative Half Cycle

If we go on increasing the value of the filter capacitor then the Ripple will decrease.
But then the costing will increase. The value of the Filter capacitor depends on
the current consumed by the circuit, the frequency of the waveform & the accepted ripple.

$$C = \frac{V_r F}{I}$$

Where,

Vr= accepted ripple voltage.(should not be more than 10% of the voltage)

I= current consumed by the circuit in Amperes.

F= frequency of the waveform. A half wave rectifier has only one peak in one cycle so F=25hz
Whereas a full wave rectifier has Two peaks in one cycle so F=100hz.

- **VOLTAGE REGULATOR**

A Voltage regulator is a device which converts varying input voltage into a constant regulated output voltage. Voltage regulator can be of two types

1) Linear Voltage Regulator

Also called as Resistive Voltage regulator because they dissipate the excessive voltage resistively as heat.

2) Switching Regulators.

They regulate the output voltage by switching the Current ON/OFF very rapidly. Since their output is either ON or OFF it dissipates very low power thus achieving higher efficiency as compared to linear voltage regulators. But they are more complex & generate high noise due to their switching action. For low level of output power switching regulators tend to be costly but for higher output wattage they are much cheaper than linear regulators.

The most commonly available Linear Positive Voltage Regulators are the 78XX series where the XX indicates the output voltage. And 79XX series is for Negative Voltage Regulators.

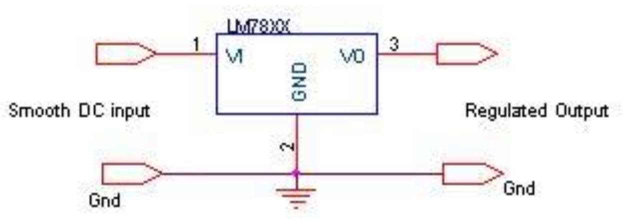


Fig 4.7 Voltage Regulator

After filtering the rectifier output the signal is given to a voltage regulator. The maximum input voltage that can be applied at the input is 35V.Normally there is a 2-3 Volts drop across the regulator so the input voltage should be at least 2-3 Volts higher than the output voltage. If the input voltage gets below the Vmin of the regulator due to the ripple voltage or due to any other reason the voltage regulator will not be able to produce the correct regulated voltage.

Circuit diagram:

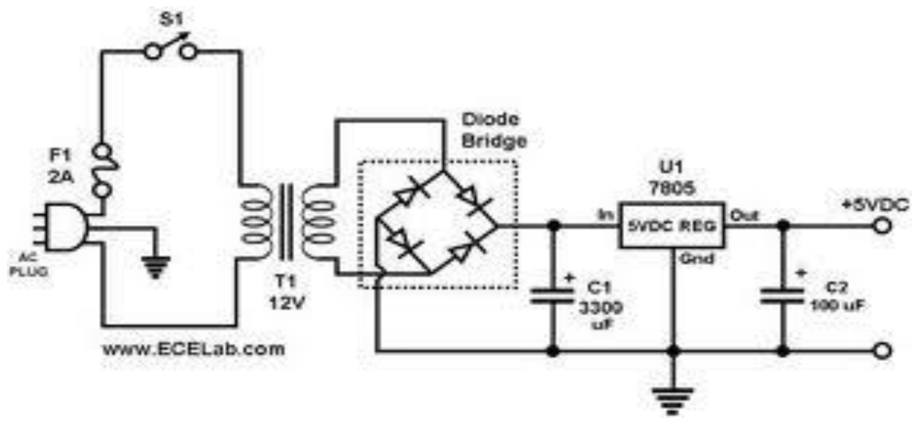


Fig 4.8. Circuit Diagram of power supply

- **IC 7805:**

7805 is an integrated three-terminal positive fixed linear voltage regulator. It supports an input voltage of 10 volts to 35 volts and output voltage of 5 volts. It has a current rating of 1 amp although lower current models are available. Its output voltage is fixed at 5.0V. The 7805 also has a built-in current limiter as a safety feature. 7805 is manufactured by many companies, including National Semiconductors and Fairchild Semiconductors.

The 7805 will automatically reduce output current if it gets too hot.The last two digits represent the voltage; for instance, the 7812 is a 12-volt regulator. The 78xx series of regulators is designed to work in complement with the 79xx series of negative voltage regulators in systems that provide both positive and negative regulated voltages, since the 78xx series can't regulate negative voltages in such a system.

The 7805 & 78 is one of the most common and well-known of the 78xx series regulators, as it's small component count and medium-power regulated 5V make it useful for powering TTL devices.

SPECIFICATIONS	**IC 7805**
V_{out}	5V

V_{ein} - V_{out} Difference	5V - 20V
Operation Ambient Temp	0 - 125°C
Output I_{max}	1A

Table 4.1: specifications of IC7805

4..1.1ARDUINO NANO (Micro controller)

- **Introduction to the Arduino NANO Board**

The Arduino Nano, as the name suggests is a compact, complete and bread-board friendly microcontroller board. The Nano board weighs around 7 grams with dimensions of 4.5 cms to 1.8 cms (L to B). This article discusses about the technical specs most importantly the pinout and functions of each and every pin in the Arduino Nano board.

Arduino Nano has similar functionalities as Arduino Duemilanove but with a different package. The Nano is inbuilt with the ATmega328P microcontroller, same as the Arduino UNO. The main difference between them is that the UNO board is presented in PDIP (Plastic Dual-In-line Package) form with 30 pins and Nano is available in TQFP (plastic quad flat pack) with 32 pins. The extra 2 pins of Arduino Nano serve for the ADC functionalities, while UNO has 6 ADC ports but Nano has 8 ADC ports. The Nano board doesn't have a DC power jack as other Arduino boards, but instead has a mini-USB port. This port is used for both programming and serial monitoring. The fascinating feature in Nano is that it will choose the strongest power source with its potential difference, and the power source selecting jumper is invalid

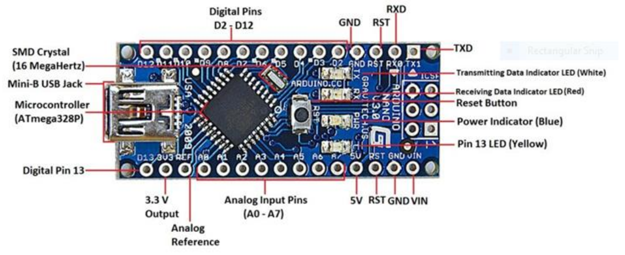


Fig 4.9 Arduino nano

- **Arduino Nano – Specification**

Arduino Nano	Specifications
Analog I/O Pins	8
Architecture	AVR
Clock Speed	16 MHz
DC Current per I/O Pins	40 milliAmps
Digital I/O Pins	22
EEPROM	1 KB
Flash Memory	32 KB of which 2 KB used by Bootloader
Input Voltage	(7-12) Volts
Microcontroller	ATmega328P
Operating Voltage	5 Volts
PCB Size	18 x 45 mm
Power Consumption	19 milliAmps
PWM Output	6
SRAM	2KB
Weight	7 gms

- **Pin diagram**

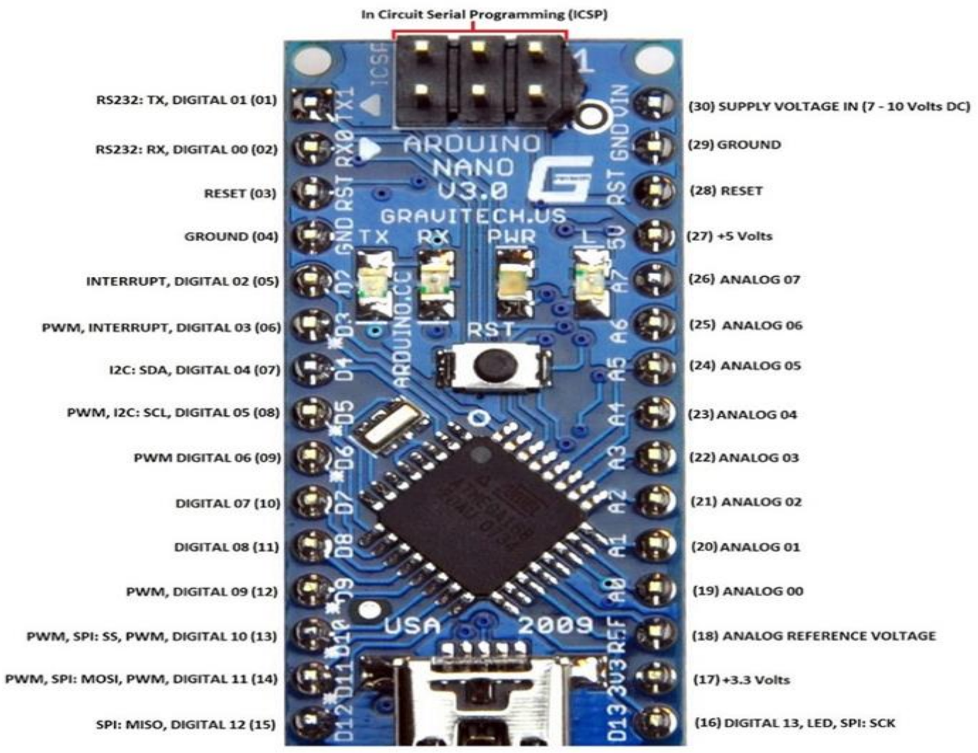

In Circuit Serial Programming (ICSP)
RS232: TX, DIGITAL 01 (01)
RS232: RX, DIGITAL 00 (02)
RESET (03)
GROUND (04)
INTERRUPT, DIGITAL 02 (05)
PWM, INTERRUPT, DIGITAL 03 (06)
I2C: SDA, DIGITAL 04 (07)
PWM, I2C: SCL, DIGITAL 05 (08)
PWM DIGITAL 06 (09)
DIGITAL 07 (10)
DIGITAL 08 (11)
PWM, DIGITAL 09 (12)
PWM, SPI: SS, PWM, DIGITAL 10 (13)
PWM, SPI: MOSI, PWM, DIGITAL 11 (14)
SPI: MISO, DIGITAL 12 (15)
(30) SUPPLY VOLTAGE IN (7 - 10 Volts DC)
(29) GROUND
(28) RESET
(27) +5 Volts
(26) ANALOG 07
(25) ANALOG 06
(24) ANALOG 05
(23) ANALOG 04
(22) ANALOG 03
(21) ANALOG 02
(20) ANALOG 01
(19) ANALOG 00
(18) ANALOG REFERENCE VOLTAGE
(17) +3.3 Volts
(16) DIGITAL 13, LED, SPI: SCK

Fig 4.10 Pin Configuration of Atmega328

Pin Description

Arduino Nan0 – Pin Description

Pins 1 to 30

Arduino Nano Pin	Pin Name	Type	Function
1	D1/TX	I/O	Digital I/O Pin Serial TX Pin
2	D0/RX	I/O	Digital I/O Pin Serial RX Pin

Arduino Nano Pin	Pin Name	Type	Function
3	RESET	Input	Reset (Active Low)
4	GND	Power	Supply Ground
5	D2	I/O	Digital I/O Pin
6	D3	I/O	Digital I/O Pin
7	D4	I/O	Digital I/O Pin
8	D5	I/O	Digital I/O Pin
9	D6	I/O	Digital I/O Pin
10	D7	I/O	Digital I/O Pin
11	D8	I/O	Digital I/O Pin
12	D9	I/O	Digital I/O Pin
13	D10	I/O	Digital I/O Pin
14	D11	I/O	Digital I/O Pin
15	D12	I/O	Digital I/O Pin

Arduino Nano Pin	**Pin Name**	**Type**	**Function**
16	D13	I/O	Digital I/O Pin
17	3V3	Output	+3.3V Output (from FTDI)
18	AREF	Input	ADC reference
19	A0	Input	Analog Input Channel 0
20	A1	Input	Analog Input Channel 1
21	A2	Input	Analog Input Channel 2
22	A3	Input	Analog Input Channel 3
23	A4	Input	Analog Input Channel 4
24	A5	Input	Analog Input Channel 5
25	A6	Input	Analog Input Channel 6
26	A7	Input	Analog Input Channel 7
27	+5V	Output or Input	+5V Output (From On-board Regulator) or +5V (Input from External Power Supply

Arduino Nano Pin	**Pin Name**	**Type**	**Function**
28	RESET	Input	Reset (Active Low)
29	GND	Power	Supply Ground
30	VIN	Power	Supply voltage

- **ICSP Pins**

Arduino Nano ICSP Pin Name	**Type**	**Function**
MISO	Input or Output	Master In Slave Out
Vcc	Output	Supply Voltage
SCK	Output	Clock from Master to Slave
MOSI	Output or Input	Master Out Slave In
RST	Input	Reset (Active Low)
5 GND	Power	Supply Ground

5.1.7.5 Arduino Nano Digital Pins

```
Pins - 1, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16
```

As mentioned earlier, Arduino Nano has 14 digital I/O pins that can be used either as digital input or output. The pins work with 5V voltage as maximum, i.e., digital high is 5V and digital low is 0V. Each pin can provide or receive a current of 20mA, and has a pull-up resistance of about 20-50k ohms. Each of the 14 digital pins on the Nano pinout can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions.

Other than the digital input and output functions, the digital pins have some additional functionality as well.

5.1.7.5 Serial Communication Pins

```
Pins - 1, 2



1 - RX and 2 - TX
```

These two pins RX- receive and TX- transmit are used for TTL serial data communication. The pins RX and TX are connected to the corresponding pins of the USB-to-TTL Serial chip.

5.1.7.6 PWM Pins

```
Pins - 6, 8, 9, 12, 13, and 14
```

Each of these digital pins provide a Pulse Width Modulation signal of 8-bit resolution. The PWM signal can be generated using analog Write () function.

5.1.7.7 External Interrupts

```
Pins - 5, 6
```

When we need to provide an external interrupt to other processor or controller we can make use of these pins. These pins can be used to enable interrupts INT0 and INT1 respectively by using the attachInterrupt () function. These pins can be used to trigger three types of interrupts such as interrupt on a low value, a rising or falling edge interrupt and a change in value interrupt.

5.1.7.8 SPI Pins

```
Pins - 13, 14, 15, and 16
```

When you don’t want the data to be transmitted asynchronously you can use these Serial Peripheral Interface pins. These pins support synchronous communication with SCK as the synchronizing clock. Even though the hardware has this feature, the Arduino software doesn’t have this by default. So you have to include a library called SPI Library for using this feature..

5.1.7.9 Nano Analog Arduino Pins

```
Pins - 18, 19, 20, 21, 22, 23, 24, 25, and 26
```

As mentioned earlier UNO got 6 analog input pins but Arduino Nano has 8 analog inputs (19 to 26), marked A0 through A7. This means you can connect *8 channel analog sensor inputs for processing. Each of these analog pins has a inbuilt ADC of resolution of 1024 bits (so it will give 1024 values). By default, the pins are measured from ground to 5V. If you want the reference voltage to be 0V to 3.3V, we can give 3.3V to AREF pin (18th Pin) by using the analogReference () function.

Similar to digital pins in Nano, analog pins also got some other functions as well.

5.1.7.9 I2C

```
Pins 23, 24 as A4 and A5
```

Since SPI communication also has its disadvantages such as 4 essential pins and limited within a device. For long distance communication we use the I2C protocol. I2C supports multi master and multi slave with only two wires. One for clock (SCL) and another for data (SDA). For using this I2C feature we need to import a library called Wire library.

5.1.7.10 ICSP

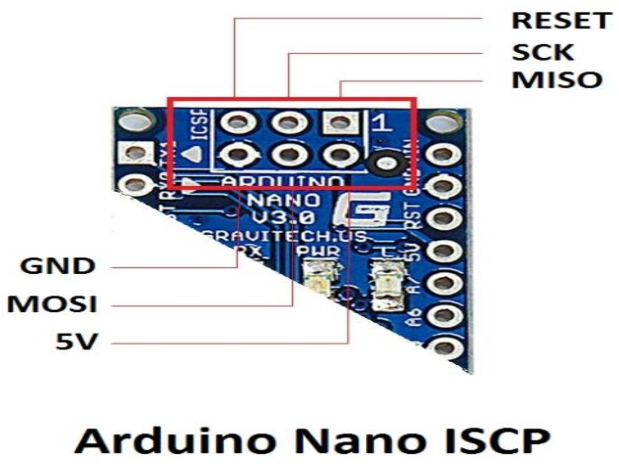


Fig 4.11 Arduino Nano Iscp

ICSP stands for *In Circuit Serial Programming*, which represents one of the several methods available for programming Arduino boards. Ordinarily, an Arduino bootloader program is used to program an Arduino board, but if the bootloader is missing or damaged, ICSP can be used instead. ICSP can be used to restore a missing or damaged bootloader.

Each ICSP pin usually is cross-connected to another Arduino pin with the same name or function. For example, MISO on Nano's ICSP header is connected to MISO / digital pin 12 (Pin 15); MOSI on the ISCP header is connected to MOSI / digital pin 11 (Pin 16); and so forth. Note, MISO, MOSI, and SCK pins taken together make up most of an SPI interface.

We can use one Arduino to program another Arduino using this ICSP.

Arduino as ISP	**ATMega328**
Vcc/5V	Vcc
GND	GND
MOSI/D11	D11
MISO/D12	D12
SCK/D13	D13
D10	Reset

5.1.7.11 Features

- 1.8-5.5V operating range
- Up to 20MHz
- Part: ATMEGA328P-AU
- 32kB Flash program memory
- 1kB EEPROM
- 2kB Internal SRAM
- 2 8-bit Timer/Counters
- 16-bit Timer/Counter
- RTC with separate oscillator

- Master/Slave SPI interface
- 2-wire (I2C) interface
- Watchdog timer
- 23 IO lines
- Data retention: 20 years at 85C/ 100 years at 25C
- Digital I/O Pins are 14 (out of which 6 provide PWM output)
- Analog Input Pins are 6.
- DC Current per I/O is 40 mA
- DC Current for 3.3V Pin is 50mA

5.1.7.12 AVR CPU Core

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

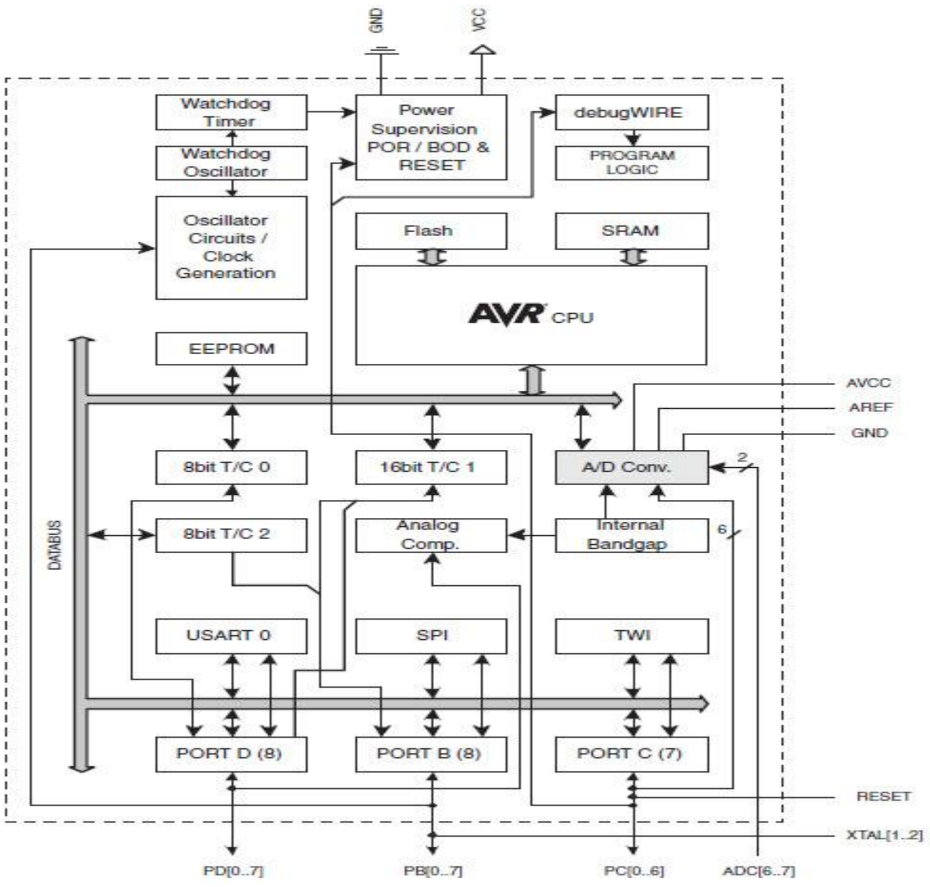
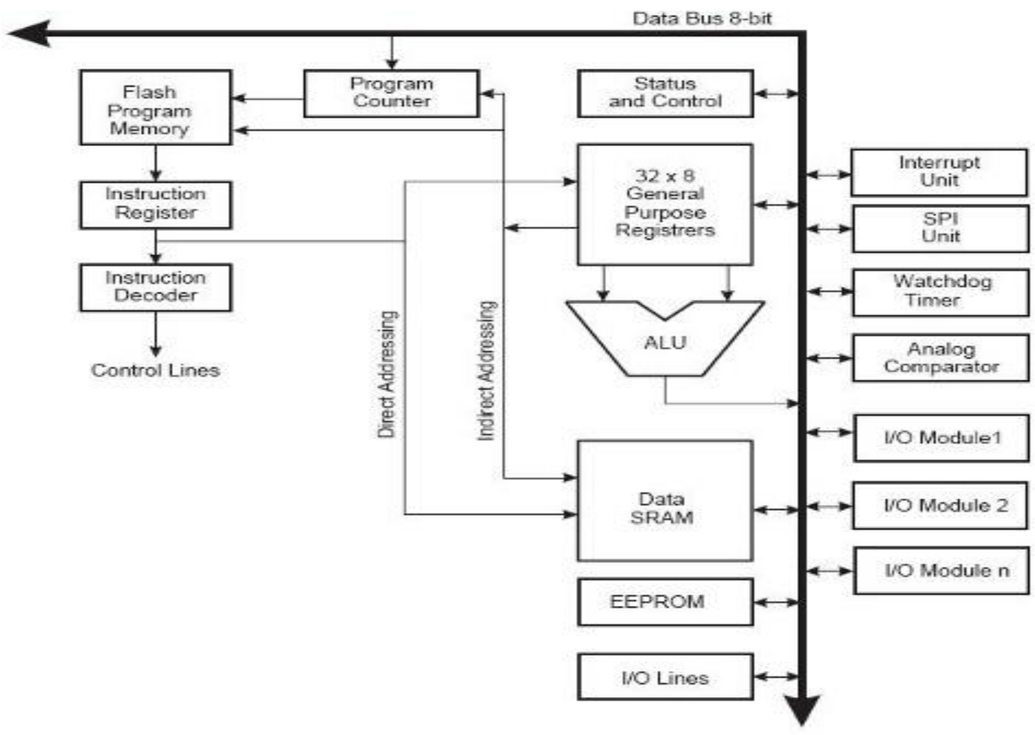


Figure 4.12 AVR Block Diagram



.

Fig 4.13: AVR core architecture

Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

- **Arduino with ATmega328**

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 a 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 versionR2) programmed as a USB-to-serial converter.

➢ Pin out: 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 future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino. Due that operates 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, see the index of Arduino boards.

- **Arduino Characteristics**

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:

1. **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.
- **IOREF.** This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

- **Memory:**

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

2. Serial 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 Software Serial 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. For SPI communication, use the SPI library.

• Current Transformer:

Voltage Divider Network:

In electronics, a **voltage divider** (also known as a **potential divider**) is a simple linear circuit that produces an output voltage (V_{out}) that is a fraction of its input voltage (V_{in}). **Voltage division** refers to the partitioning of a voltage among the components of the divider.

The formula governing a voltage divider is similar to that for a current divider, but the ratio describing voltage division places the selected impedance in the numerator, unlike current division where it is the unselected components that enter the numerator.

A simple example of a voltage divider consists of two resistors in series or a potentiometer. It is commonly used to create a reference voltage, and may also be used as a signal attenuator at low frequencies.

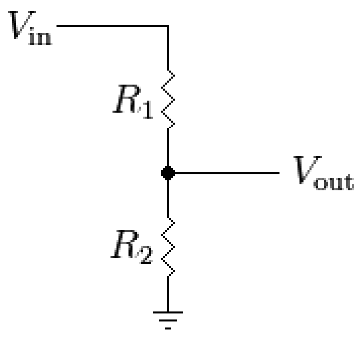


Fig. 4.14 Simple resistive voltage divider

A resistive divider is a special case where both impedances, Z_1 and Z_2, are purely resistive .

Substituting $Z_1 = R_1$ and $Z_2 = R_2$ into the previous expression gives:

$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in}$$

As in the general case, R_1 and R_2 may be any combination of series/parallel resistors.

- **MOS voltage divider:**

We can use the enhanced MOSFET as a nonlinear resistor while its gate and drain are connected.

Examples:

Resistive divider

As a simple example, if $R_1 = R_2$ then

$$V_{out} = \frac{1}{2} \cdot V_{in}$$

As a more specific and/or practical example, if V_{out}=6V and V_{in}=9V (both commonly used voltages), then :

$$\frac{V_{\text{out}}}{V_{\text{in}}} = \frac{R_2}{R_1 + R_2} = \frac{6}{9} = \frac{2}{3}$$

And by solving using algebra, R_2 must be twice the value of R_1.

To solve for R1:

$$R_1 = \frac{R_2 \cdot V_{\text{in}}}{V_{\text{out}}} - R_2$$

To solve for R2:

$$R_2 = \frac{R_1}{\left(\frac{V_{\text{in}}}{V_{\text{out}}} - 1\right)}$$

Any ratio between 0 and 1 is possible. That is, using resistors alone it is not possible to either reverse the voltage or increase V_{out} above V_{in}

- **Design:**

Like any other transformer, a current transformer has a primary winding, a magnetic core, and a secondary winding. The alternating current flowing in the primary produces a magnetic field in the core, which then induces current flow in the secondary winding circuit. A primary objective of current transformer design is to ensure that the primary and secondary circuits are efficiently coupled, so that the secondary current bears an accurate relationship to the primary current.

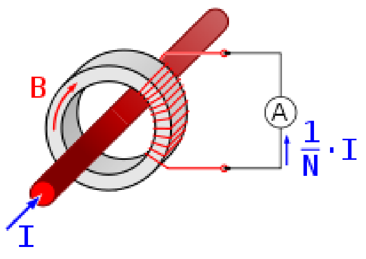


Fig. 4.15 Current Transformer Principle

The most common design of CT consists of a length of wire wrapped many times around a silicon steel ring passed over the circuit being measured. The CT's primary circuit therefore consists of a single 'turn' of conductor, with a secondary of many hundreds of turns. The primary winding may be a permanent part of the current transformer, with a heavy copper bar to carry current through the magnetic core. Window-type current transformers are also common, which can have circuit cables run through the middle of an opening in the core to provide a single-turn primary winding. When conductors passing through a CT are not centered in the circular (or oval) opening, slight inaccuracies may occur. Current transformers used in metering equipment for three-phase 400 ampere electricity supply

Shapes and sizes can vary depending on the end user or switchgear manufacturer. Typical examples of low voltage single ratio metering current transformers are either ring type or plastic molded case. High-voltage current transformers are mounted on porcelain bushings to insulate them from ground. Some CT configurations slip around the bushing of a high-voltage transformer or circuit breaker, which automatically centers the conductor inside the CT window.

The primary circuit is largely unaffected by the insertion of the CT. The rated secondary current is commonly standardized at 1 or 5 amperes. For example, a 4000:5 CT would provide an output current of 5 amperes when the primary was passing 4000 amperes. The secondary winding can be single ratio or multi ratio, with five taps being common for multi ratio CTs. The load, or burden, of the CT should be of low resistance. If the voltage time integral area is higher than the core's design rating, the core goes into saturation towards the end of each cycle, distorting the waveform and affecting accuracy.

- **Safety precautions:**

Care must be taken that the secondary of a current transformer is not disconnected from its load while current is flowing in the primary, as the transformer secondary will attempt to continue driving current across the effectively infinite impedance. This will produce a high voltage across the open secondary (into the range of several kilovolts in some cases), which may cause arcing. The high voltage produced will compromise operator and equipment safety and permanently affect the accuracy of the transformer.

- **Accuracy:**

  The accuracy of a CT is directly related to a number of factors including:

- Burden
- Burden class/saturation class
- Rating factor
- Load
- External electromagnetic fields
- Temperature and
- Physical configuration.
- The selected tap, for multi-ratio CT's

- **Rating factor:**

Rating factor is a factor by which the nominal full load current of a CT can be multiplied to determine its absolute maximum measurable primary current. Conversely, the minimum primary current a CT can accurately measure is "light load," or 10% of the nominal current (there are, however, special CTs designed to measure accurately currents as small as 2% of the nominal current). The rating factor of a CT is largely dependent upon ambient temperature. Most CTs have rating factors for 35 degrees Celsius and 55 degrees Celsius. It is important to be mindful of ambient temperatures and resultant rating factors when CTs are installed inside pad-mounted transformers or poorly ventilated mechanical rooms. Recently, manufacturers have been moving towards lower nominal primary currents with greater rating factors. This is made possible by the development of more efficient ferrites and their corresponding hysteresis curves. This is a distinct advantage over previous CTs because it increases their range of accuracy, since the CTs are most accurate between their rated current and rating factor.

- **Special designs:**

Specially constructed wideband current transformers are also used (usually with an oscilloscope) to measure waveforms of high frequency or pulsed currents within pulsed power systems. One type of specially constructed wideband transformer provides a voltage output that is proportional to the measured current. Another type (called a Rogowski coil) requires an external integrator in order to provide a voltage output that is proportional to the measured current. Unlike CTs used for power circuitry, wideband CTs are rated in output volts per ampere of primary current.

- **Voltage Divider Network:**

In electronics, a **voltage divider** (also known as a **potential divider**) is a simple linear circuit that produces an output voltage (V_{out}) that is a fraction of its input voltage (V_{in}). **Voltage division** refers to the partitioning of a voltage among the components of the divider.

The formula governing a voltage divider is similar to that for a current divider, but the ratio describing voltage division places the selected impedance in the numerator, unlike current division where it is the unselected components that enter the numerator.

A simple example of a voltage divider consists of two resistors in series or a potentiometer. It is commonly used to create a reference voltage, and may also be used as a signal attenuator at low frequencies.

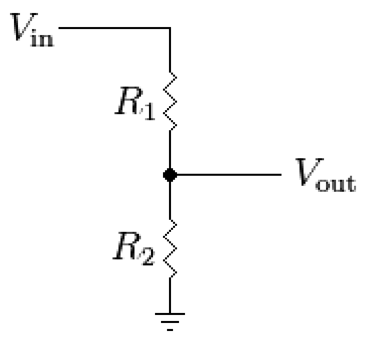


Fig 4.16 Simple resistive voltage divider

A resistive divider is a special case where both impedances, Z_1 and Z_2, are purely resistive .

Substituting $Z_1 = R_1$ and $Z_2 = R_2$ into the previous expression gives:

$$V_{\text{out}} = \frac{R_2}{R_1 + R_2} \cdot V_{\text{in}}$$

As in the general case, R_1 and R_2 may be any combination of series/parallel resistors.

- **MOS voltage divider:**

We can use the enhanced MOSFET as a nonlinear resistor while its gate and drain are connected.

3. **Examples:**

Resistive divider

As a simple example, if $R_1 = R_2$ then

$$V_{\text{out}} = \frac{1}{2} \cdot V_{\text{in}}$$

As a more specific and/or practical example, if V_{out}=6V and V_{in}=9V (both commonly used voltages), then:

$$\frac{V_{\text{out}}}{V_{\text{in}}} = \frac{R_2}{R_1 + R_2} = \frac{6}{9} = \frac{2}{3}$$

And by solving using algebra, R_2 must be twice the value of R_1.

To solve for R1:

$$R_1 = \frac{R_2 \cdot V_{\text{in}}}{V_{\text{out}}} - R_2$$

To solve for R2:

$$R_2 = \frac{R_1}{\left(\frac{V_{\text{in}}}{V_{\text{out}}} - 1\right)}$$

Any ratio between 0 and 1 is possible. That is, using resistors alone it is not possible to either reverse the voltage or increase V_{out} above V_{in}

4.1.ULTRASONIC SENSOR

Ultrasonic sensors are industrial control devices that use sound waves above 20,000 Hz, beyond the range of human hearing, to measure and calculate distance from the sensor to a specified target object.

4.1.8.1 Features of ultrasonic sensors:

- Devices with TEACH-IN functionality for fast and simple installation
- ULTRA 3000 software for improved adaptation of sensors to applications
- Adjustable sensitivity to the sound beam width for optimized adjustment of the sensor characteristics according to the application
- Temperature compensation - compensates for sound velocity due to varying air temperatures
- Synchronization input to prevent cross-talk interference when sensors are mounted within close proximity of each other
- Sensors with digital and/ or analog outputs

4.1.8.2 Description:

Ultrasonic sensors use electrical energy and a ceramic transducer to emit and receive mechanical energy in the form of sound waves. Sound waves are essentially pressure waves that travel through solids, liquids and gases and can be used in industrial applications to measure distance or detect the presence or absence of targets

Ultrasonic sensors (also known as tranceivers when they both send and receive) work on a principle similar to radar or sonar which evaluate attributes of a target by interpreting the echoes from radio or sound waves respectively. Ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor. Sensors calculate the time interval between sending the signal and receiving the echo to determine the distance to an object.

This technology can be used for measuring: wind speed and direction (anemometer), fullness of a tank, and speed through air or water. For measuring speed or direction a device uses multiple detectors and calculates the speed from the relative distances to particulates in the air or water. To measure the amount of liquid in a tank, the sensor measures the distance to the surface of the fluid. Further applications include: humidifiers, sonar, medical ultrasonography, burglar alarms, and non-destructive testing.

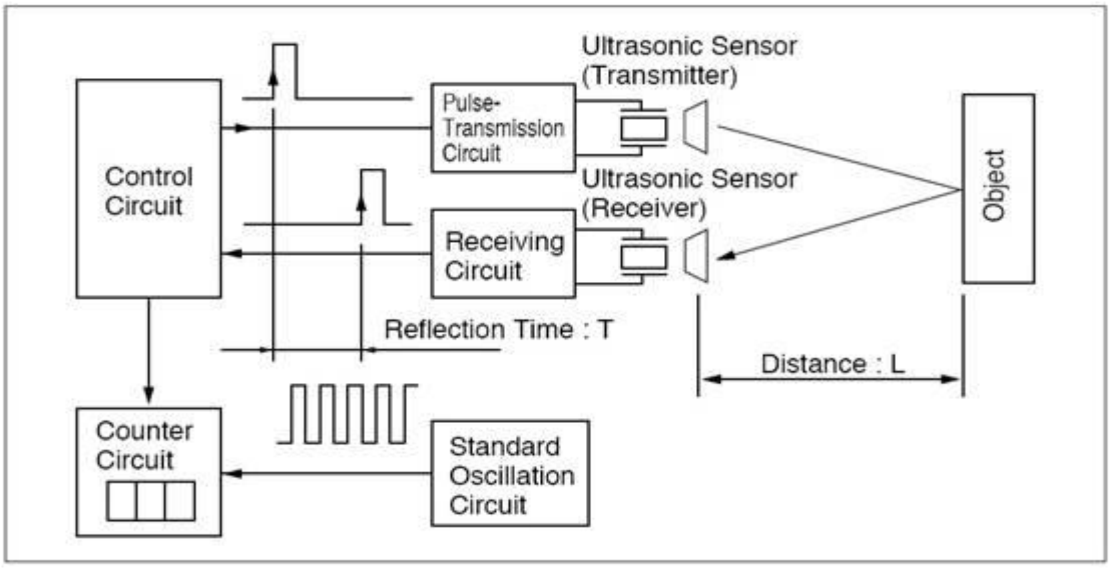

Ultrasonic Sensor (Transmitter)
Pulse-Transmission Circuit
Control Circuit
Ultrasonic Sensor (Receiver)
Receiving Circuit
Object
Reflection Time : T
Distance : L
Counter Circuit
Standard Oscillation Circuit

Fig 4.17: Block Diagram of Ultrasonic Sensor

Systems typically use a transducer which generates sound waves in the ultrasonic range, above 20,000 hertz, by turning electrical energy into sound, then upon receiving the echo turn the sound waves into electrical energy which can be measured and displayed.

The technology is limited by the shapes of surfaces and the density or consistency of the material. For example foam on the surface of a fluid in a tank could distort a reading.

o Transducers

Sound field of a non focusing 4MHz ultrasonic transducer with a near field length of N=67mm in water. The plot shows the sound pressure at a logarithmic dbscale.Sound pressure field of the same ultrasonic transducer (4MHz, N=67mm) with the transducer surface having a spherical curvature with the curvature radius R=30mm

An ultrasonic transducer is a device that converts energy into ultrasound, or sound waves above the normal range of human hearing. While technically a dog whistle is an ultrasonic transducer that converts mechanical energy in the form of air pressure into ultrasonic sound waves, the term is more apt to be used to refer to piezoelectric transducers that convert electrical energy into sound. Piezoelectric crystals have the property of changing size when a voltage is applied, thus applying an alternating current (AC) across them causes them to oscillate at very high frequencies, thus producing very high frequency sound waves.

The location at which a transducer focuses the sound, can be determined by the active transducer area and shape, the ultrasound frequency and the sound velocity of the propagation medium.

The example shows the sound fields of an unfocused and a focusing ultrasonic transducer in water.

Working

The sensor has a ceramic transducer that vibrates when electrical energy is applied to it. The vibrations compress and expand air molecules in waves from the sensor face to a target object. A transducer both transmits and receives sound. The ultrasonic sensor will measure distance by emitting a sound wave and then "listening" for a set period of time, allowing for the return echo of the sound wave bouncing off the target, before retransmitting.

Microcontroller and the ultrasonic transducer module HC-SR04 forms the basis of this circuit. The ultrasonic module sends a signal to the object, then picks up its echo and outputs a wave form whose time period is proportional to the distance. The microcontroller accepts this signal, performs necessary processing and displays the corresponding distance on the 3 digit seven segment display. This circuit finds a lot of application in projects like automotive parking sensors, obstacle warning systems, terrain monitoring robots, industrial distance measurements etc.

It has a resolution of 0.3cm and the ranging distance is from 2cm to 500cm. It operates from a 5V DC supply and the standby current is less than 2mA. The module transmits an ultrasonic signal, picks up its echo, measures the time elapsed between the two events and outputs a waveform whose high time is modulated by the measured time which is proportional to the distance.

The supporting circuits fabricated on the module makes it almost stand alone and what the programmer need to do is to send a trigger signal to it for initiating transmission and receive the echo signal from it for distance calculation.

The HR-SR04 has four pins namely Vcc, Trigger, Echo, GND and they are explained in detail below.

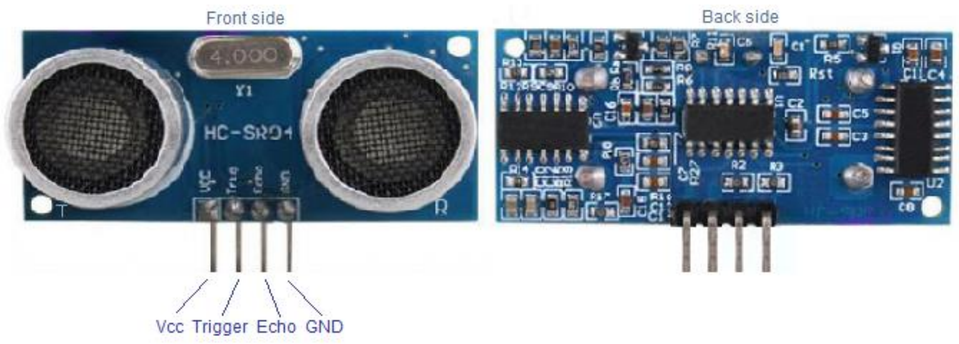


Fig 4.18: Pin Description and View of Ultrasonic Sensor

1) **VCC** : 5V DC supply voltage is connected to this pin.

2) **Trigger**: The trigger signal for starting the transmission is given to this pin. The trigger signal must be a pulse with 10uS high time. When the module receives a valid trigger signal it issues 8 pulses of 40KHz ultrasonic sound from the transmitter. The echo of this sound is picked by the receiver.

3) **Echo**: At this pin, the module outputs a waveform with high time proportional to the distance.

4) **GND**: Ground is connected to this pin.

The transmitter part of the circuit is build around IC1(NE 555).The IC1 is wired as an astable multi vibrator operating at 40KHz.The output of IC1 is amplifier the complementary pair of transistors (Q1 & Q2) and transmitted by the ultrasonic transmitter K1.The push button switch S1 is used the activate the transmitter.

The receiver uses an ultrasonic sensor transducer (K2) to sense the ultrasonic signals. When an ultrasonic signal is falling on the sensor, it produces a proportional voltage signal at its output. This weak signal is amplified by the two stage amplifier circuit comprising of transistors Q3 and Q4.The output of the amplifier is rectified by the diodes D3 & D4.The rectified signal is given to the inverting input of the opamp which is wired as a comparator. Whenever there is an ultrasonic signal falling on the receiver, the output of the comparator activates the transistors Q5 & Q6 to drive the relay. In this way the load connected via the relay can be switched. The diode D5 is used as a free-wheeling diode.

Detectors

Since piezoelectric crystal generate a voltage when force is applied to them, the same crystal can be used as an ultrasonic detector. Some systems use separate transmitter and receiver components while others combine both in a single piezoelectric transceiver.

Alternative methods for creating and detecting ultrasound include magnetostriction and capacitive actuation.

Application

Ultrasonic sensors use sound waves rather than light, making them ideal for stable detection of uneven surfaces, liquids, clear objects, and objects in dirty environments. These sensors work well for applications that require precise measurements between stationary and moving objects.

Ultrasonic sensor provides a very low-cost and easy method of distance measurement. This sensor is perfect for any number of applications that require you to perform measurements between moving or stationary objects. Naturally, robotics applications are very popular but it is also find in product which is useful in security systems or as an infrared replacement if so desired.

Use in medicine

Medical ultrasonic transducers (probes) come in a variety of different shapes and sizes for use in making pictures of different parts of the body. The transducer may be passed over the surface of the body or inserted into an body opening such as the rectum or vagina. Clinicians who perform ultrasound-guided procedures often use a probe positioning system to hold the ultrasonic transducer.

Use in industry

Ultrasonic sensors are used to detect the presence of targets and to measure the distance to targets in many automated factories and process plants. Sensors with an on or off digital output are available for detecting the presence of objects, and sensors with an analog output which varies proportionally to the sensor to target separation distance are commercially available.

Other types of transducers are used in commercially available ultrasonic cleaning devices. An ultrasonic transducer is affixed to a stainless steel pan which is filled with a solvent (frequently water or isopropanol) and a square wave is applied to it, imparting vibrational energy on the liquid.

Application In Industries

- Measurement of dynamically changing diameters
- Measurement of dynamically changing distances
- Measurement of dynamically changing heights
- Measurement of dynamically changing depths
- Counting number of units.

4.1.3 L293D DRIVER CIRCUIT

L293D IC generally comes as a standard 16-pin DIP (dual-in line package). This motor driver IC can simultaneously control two small motors in either direction; forward and reverse with just 4 microcontroller pins (if you do not use enable pins).

5.1.10.1 Some of the features (and drawbacks) of this IC are:

1. Output current capability is limited to 600mA per channel with peak output current limited to 1.2A (non-repetitive). This means you cannot drive bigger motors with this IC. However, most small motors used in hobby robotics should work. If you are unsure whether the IC can handle a particular motor, connect the IC to its circuit and run the motor with your finger on the IC. If it gets really hot, then beware... Also note the words "non-repetitive"; if the current output repeatedly reaches 1.2A, it might destroy the drive transistors.
2. Supply voltage can be as large as 36 Volts. This means you do not have to worry much about voltage regulation.
3. L293D has an enable facility which helps you enable the IC output pins. If an enable pin is set to logic high, then state of the inputs match the state of the outputs. If you pull this low, then the outputs will be turned off regardless of the input states
4. The datasheet also mentions an "over temperature protection" built into the IC. This means an internal sensor senses its internal temperature and stops driving the motors if the temperature crosses a set point
5. Another major feature of **L293D** is its internal clamp diodes. This flyback diode helps protect the driver IC from voltage spikes that occur when the motor coil is turned on and off (mostly when turned off)
6. The logical low in the IC is set to 1.5V. This means the pin is set high only if the voltage across the pin crosses 1.5V which makes it suitable for use in high frequency applications like switching applications (upto 5KHz)
7. Lastly, this integrated circuit not only drives DC motors, but can also be used to drive relay solenoids, stepper motors etc.

5.1.10.2 Pin Diagram

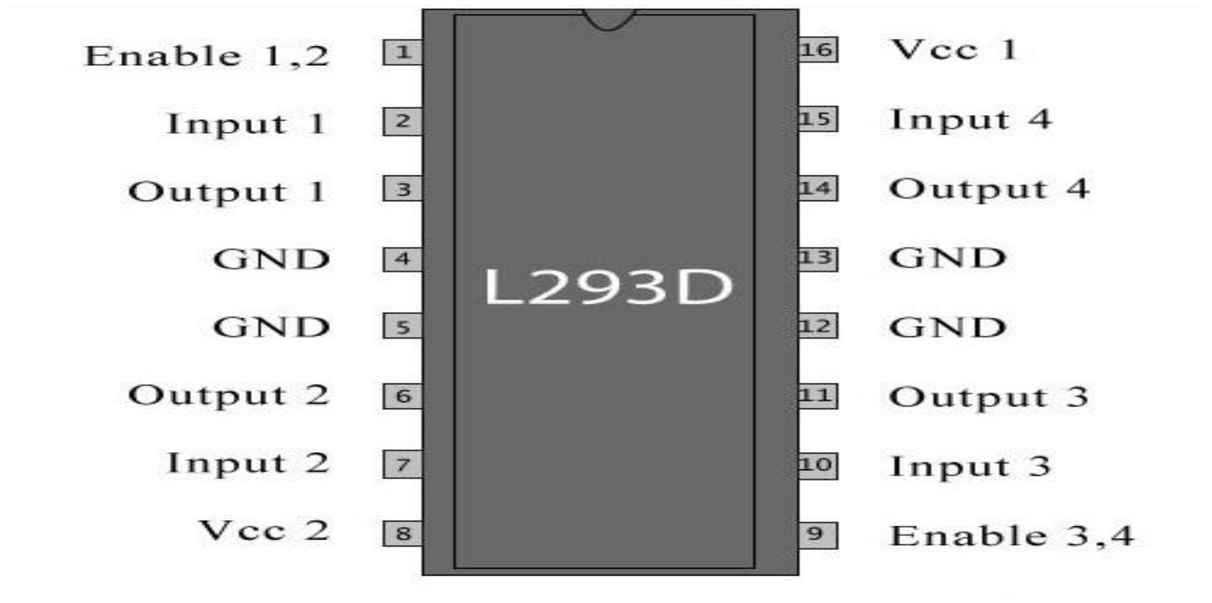


Fig 4.19 pin diagram of L293D

Pin Description

Pin No	**Function**	**Name**
1	Enable pin for Motor 1; active high	Enable 1,2
2	Input 1 for Motor 1	Input 1
3	Output 1 for Motor 1	Output 1
4	Ground (0V)	Ground
5	Ground (0V)	Ground
6	Output 2 for Motor 1	Output 2
7	Input 2 for Motor 1	Input 2
8	Supply voltage for Motors; 9-12V (up to 36V)	Vcc_2
9	Enable pin for Motor 2; active high	Enable 3,4
10	Input 1 for Motor 1	Input 3
11	Output 1 for Motor 1	Output 3

12	Ground (0V)	Ground
13	Ground (0V)	Ground
14	Output 2 for Motor 1	Output 4
15	Input2 for Motor 1	Input 4
16	Supply voltage; 5V (up to 36V)	Vcc $_1$

Table 4.2 : Pin Description of L293D

5.1.10.3 Working of L293D

The 4 input pins for this l293d, pin 2,7 on the left and pin 15 ,10 on the right as shown on the pin diagram. Left input pins will regulate the rotation of motor connected across left side and right input for motor on the right hand side. The motors are rotated on the basis of the inputs provided across the input pins as LOGIC 0 or LOGIC 1.

In simple you need to provide Logic 0 or 1 across the input pins for rotating the motor.

4.1.4 DC MOTOR

A DC Motor in simple words is a device that converts direct current (electrical energy) into mechanical energy. It’s of vital importance for the industry today.

- **Gear DC Motor**

Geared DC motors can be defined as an extension of DC motor. A geared DC Motor has a gear assembly attached to the motor. The speed of motor is counted in terms of rotations of the shaft per minute and is termed as RPM .The gear assembly helps in increasing the torque and reducing the speed. Using the correct combination of gears in a gear motor, its speed can be reduced to any desirable figure. This concept where gears reduce the speed of the vehicle but increase its torque is known as gear reduction. This Insight will explore all the minor and major details that make the gear head and hence the working of geared DC motor.

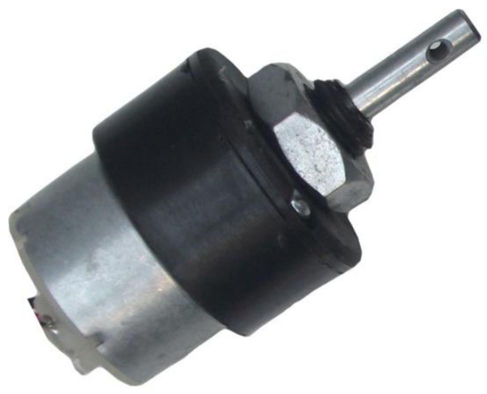


Fig 4.20: DC Motor

❖ WORKING

The DC motor works over a fair range of voltage. The higher the input voltage more is the RPM (rotations per minute) of the motor. For example, if the motor works in the range of 6-12V, it will have the least RPM at 6V and maximum at 12 V. The speed of a DC motor can be controlled by changing the voltage applied to the armature or by changing the field current. The introduction of variable resistance in the armature circuit or field circuit allowed speed control. Modern DC motors are often controlled by power electronics systems called DC drives.

❖ Usage

The DC motor or Direct Current Motor to give it its full title, is the most commonly used actuator for producing continuous movement and whose speed of rotation can easily be controlled, making them ideal for use in applications were speed control, servo type control, and/or positioning is required. Our gear motors are used for various applications in the areas of medicine and healthcare, automobile systems, drive systems, or as a variety of positioning or industrial / consumer actuators.

4.1.5 SOLAR PANEL

Renewable energy is critical to our fight against climate change. We simply must shift our world to a low-carbon economy and away from oil and coal.

Experts agree we need a substantial reduction in CO2 over the next 40-50 years and this means we need renewable energy to replace fossil fuels now.

Presently, most of the energy what we are using is from non renewable sources like petrol, coal etc.., which are very limited.

Apart from the non-renewable energy sources, renewable energy sources like wind energy, solar energy etc.., are used then we can save non-renewable sources for longer time.

A solar cell (also called photovoltaic cell) is a solid state device that converts the energy of sunlight directly into electricity by the photovoltaic effect.

Assemblies of cells are used to make solar modules, also known as solar panels.

The energy generated from these solar modules, referred to as solar power, is an example of solar energy.

Figure 4.21 : Solar panels

Advantages They have no moving parts and hence require little maintenance and work quite satisfactorily without any focusing device.

It does not cause any environmental pollution like the fossil fuels and nuclear power.

Solar cells last a longer time and have low running costs

Low power consumption.

Conservation of energy.

Utilization of free available source of energy from sun

Storage of energy into rechargeable battery.

Stored energy is used for grass cutter.

Motor automation.

High efficiency can be achieved with relay switch.

By using this project we can save more power. That is we can reduce the wastage of power.

Here no need of man. The circuit itself checks the presence of vehicle and also checks weather it is day or night time. Once we switch on the circuit it automatically performs all this actions without manual operation.

At night time also whenever vehicle comes at that time only light brightness increases after few seconds it will come to normal position that is decreases light brightness.

It is the most advantage of this project. For these all reasons in future this project may be used in all streets to save power.

4.1..6 BATTERY

A battery is a device consisting of one or more electrochemical cells that convert stored chemical energy into electrical energy. Batteries are another way to produce electricity. They are smaller and more safe. Batteries have one end that is positive and one end that is negative. For batteries to work, you need to make sure you put them in the right way. Batteries have become a common power source for many household and industrial applications.

There are two types of batteries: primary batteries (disposable batteries), which are designed to be used once and discarded, and secondary batteries (rechargeable batteries), which are designed to be recharged and used multiple times. Batteries come in many sizes, from miniature cells used to power hearing aids and wristwatches to battery banks the size of rooms that provide standby power for telephone exchanges and computer data centers

Lead Acid Battery

Lead battery is most commonly used in PV systems due to low cost and easily available everywhere in the world. These batteries are available in both sealed and wet cell batteries. Lead acid batteries have high reliability due to their capability to withstand overcharge, over discharge & shock. The batteries have excellent charge acceptance, low self-discharge and large electrolyte volume. Lead acid batteries Are tested using Computer Aided Design. These applications of these batteries are used in UPS Systems and Inverter and have the skill to perform under dangerous conditions.

.

Fig 4.22: Lead acid battery

4.2SOFTWARE DESCRIPTION:

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

WRITING SKETCHES

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.

Verify

Checks your code for errors compiling it.

Upload

Compiles your code and uploads it to the configured board. See uploading below for details.

Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

New Creates a new sketch.

Open Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File | Sketchbookmenu instead.

Save Saves your sketch.

Serial Monitor Opens the serial monitor.

Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

File

- *New* Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.
- *Open* Allows to load a sketch file browsing through the computer drives and folders.
- *Open Recent* Provides a short list of the most recent sketches, ready to be opened.
- *Sketchbook* Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.
- *Examples* Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.
- *Close* Closes the instance of the Arduino Software from which it is clicked.
- *Save* Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as.." window.
- *Saveas...* Allows to save the current sketch with a different name.
- *PageSetup* It shows the Page Setup window for printing.
- *Print*
  Sends the current sketch to the printer according to the settings defined in Page Setup.
- *Preferences*
  Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.
- *Quit*
  Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

EDIT

- *Undo/Redo*

  Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.
- *Cut*

  Removes the selected text from the editor and places it into the clipboard.
- *Copy*

  Duplicates the selected text in the editor and places it into the clipboard.
- *Copy for Forum* Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.
- *Copy as HTML* Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.
- *Paste*

  Puts the contents of the clipboard at the cursor position, in the editor.
- *Select All* Selects and highlights the whole content of the editor.
- *Comment/Uncomment*

  Puts or removes the // comment marker at the beginning of each selected line.
- *Increase/Decrease Indent* :Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.
- *Find*

  Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.
- *Find Next*: Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.
- *Find Previous* : Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

SKETCH

- *Verify/Compile*

  Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.
- *Upload*

  Compiles and loads the binary file onto the configured board through the configured Port.
- *Upload Using Programmer*

This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a *Tools -> Burn Bootloader* command mus t be executed.

- *Export Compiled Binary*
  Saves a .hex file that may be kept as archive or sent to the board using other tools.
- *Show Sketch Folder*
  Opens the current sketch folder.
- *Include Library*
  Adds a library to your sketch by inserting #include statements at the start of your code. For more details, seelibraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.
- *Add File...*
  Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side o the toolbar.

TOOLS

- *Auto Format*
  This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.
- *Archive Sketch*
  Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.
- *Fix Encoding & Reload*
  Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.
- *Serial Monitor*
  Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.

- *Board*

  Select the board that you're using. See below for descriptions of the various boards.
- *Port*

  This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.
- *Programmer*

  For selecting a harware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.
- *Burn Bootloader*

  The items in this menu allow you to burn a bootloader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the Boards menu before burning the bootloader on the target board. This command also set the right fuses.

HELP

Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.

- *Find in Reference*

  This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

SKETCHBOOK

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

Tabs, Multiple Files, and Compilation

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

UPLOADING

Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or/dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx ,/dev/ttyUSBx or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

LIBRARIES

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #includestatements from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

To write your own library.

THIRD-PARTY HARDWARE

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, bootloaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "arduino" as the sub-directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory.

For details on creating packages for third-party hardware, see the Arduino IDE 1.5 3rd party Hardware specification.

LANGUAGE SUPPORT

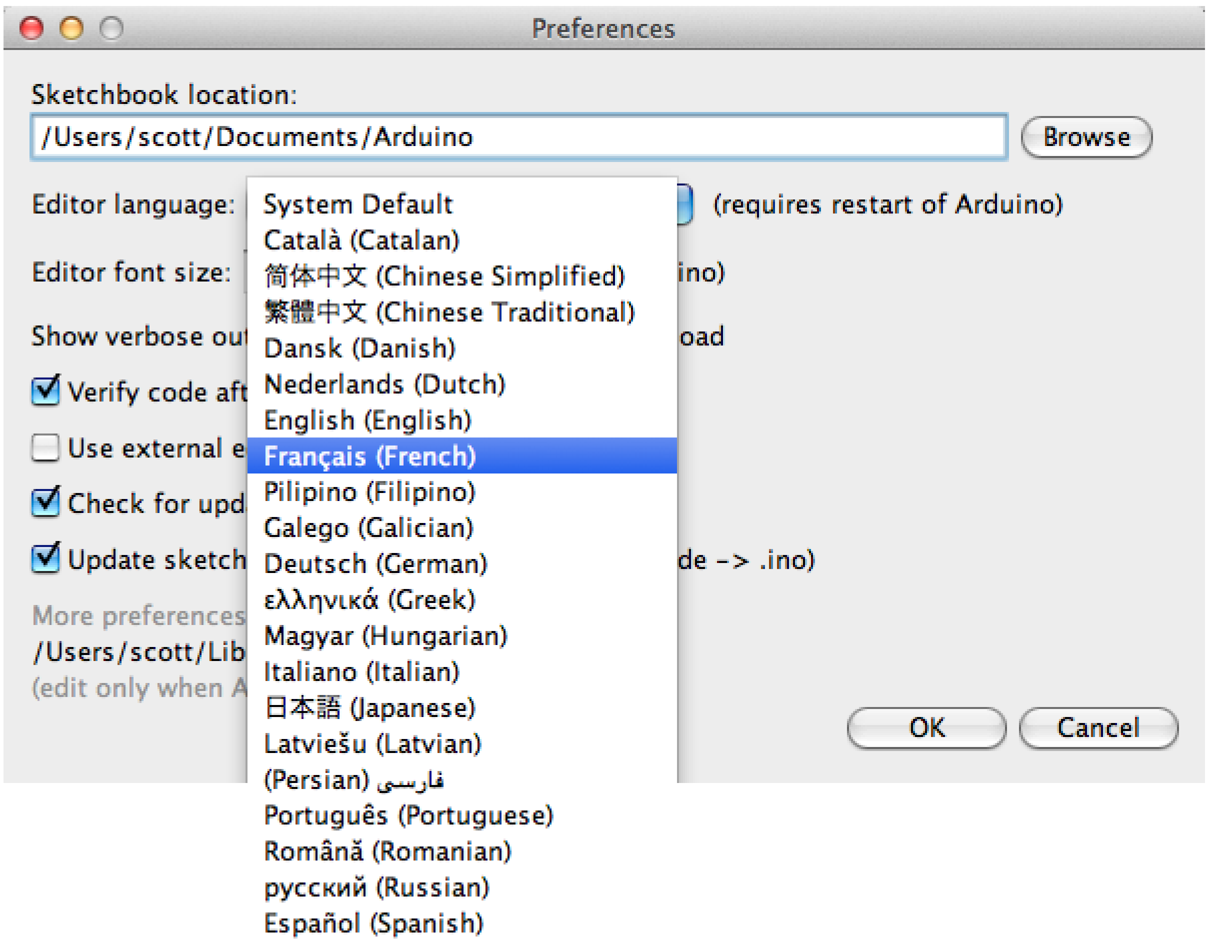


Figure 4.23 : Language support

Since version 1.0.1 , the Arduino Software (IDE) has been translated into 30+ different languages. By default, the IDE loads in the language selected by your operating system. (Note: on Windows and possibly Linux, this is determined by the locale setting which controls currency and date formats, not by the language the operating system is displayed in.)

If you would like to change the language manually, start the Arduino Software (IDE) and open the Preferences window. Next to the Editor Language there is a dropdown menu of currently supported languages. Select your preferred language from the menu, and restart the software to use the selected language. If your operating system language is not supported, the Arduino Software (IDE) will default to English.

You can return the software to its default setting of selecting its language based on your operating system by selecting System Default from the Editor Language drop-down. This setting will take effect when you restart the Arduino Software (IDE). Similarly, after changing your operating system's settings, you must restart the Arduino Software (IDE) to update it to the new default language.

BOARDS

The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets and the file and fuse settings used by the burn bootloader command. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection you'll want to check it before burning the bootloader. You can find a comparison table between the various boards here.

Arduino Software (IDE) includes the built in support for the boards in the following list, all based on the AVR Core. The Boards Manager included in the standard installation allows to add support for the growing number of new boards based on different cores like Arduino Due, Arduino Zero, Edison, Galileo and so on.

- *Arduino Yùn*
  An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino/Genuino Uno*
  An ATmega328 running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino Diecimila or Duemilanove w/ ATmega168*
  An ATmega168 running at 16 MHz with auto-reset.
- *Arduino Nano w/ ATmega328*
  An ATmega328 running at 16 MHz with auto-reset. Has eight analog inputs.
- *Arduino/Genuino Mega 2560*
  An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- *Arduino Mega*
  An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.

- *Arduino Mega ADK*

  An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- *Arduino Leonardo*

  An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino/Genuino Micro*

  An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino Esplora*

  An ATmega32u4 running at 16 MHz with auto-reset.
- *Arduino Mini w/ ATmega328*

  An ATmega328 running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino Ethernet*

  Equivalent to Arduino UNO with an Ethernet shield: An ATmega328 running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino Fio*

  An ATmega328 running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ATmega328, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino BT w/ ATmega328*

  ATmega328 running at 16 MHz. The bootloader burned (4 KB) includes codes to initialize the on-board bluetooth module, 6 Analog In, 14 Digital I/O and 6 PWM..
- *LilyPad Arduino USB*

  An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4 PWM.
- *LilyPad Arduino*

  An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328*

  An ATmega328 running at 16 MHz with auto-reset. Equivalent to Arduino Duemilanove or Nano w/ ATmega328; 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino NG or older w/ ATmega168*

  An ATmega168 running at 16 MHz *without* auto-reset. Compilation and upload is equivalent to Arduino Diecimila or Duemilanove w/ ATmega168, but the bootloader burned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 Analog In, 14 Digital I/O and 6 PWM.

- *Arduino Robot Control*

  An ATmega328 running at 16 MHz with auto-reset.
- *Arduino Robot MOTOR*

  An ATmega328 running at 16 MHz with auto-reset.

  An ATtiny85 running at 8 MHz with auto-reset, 1 Analog In, 3 Digital I/O and 2 PWM.

Chapter 5

RESULTS

PROJECT OUTPUT RESULTS

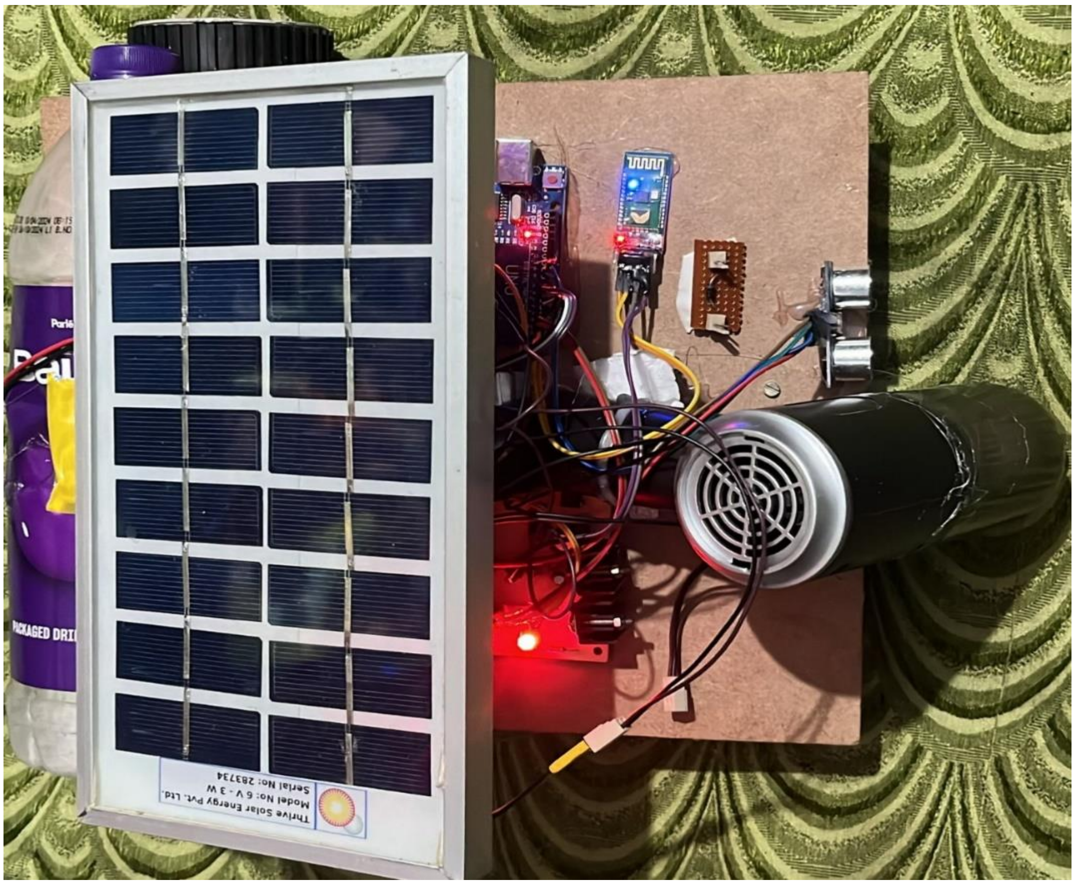


Figure5.1: when the ciruit is ON

The above figure shows the connections of the system . when the kit is in ON condition . the energy stored in the solar panel is used to clean the floor and it can be operated through remote control

CHAPTER 6

TESTING AND DEBUGGING

TESTING:

Testing a smart vacuum cleaner equipped with solar panels involves a comprehensive evaluation of its various functionalities and performance metrics.

The efficiency of the solar panels is rigorously assessed, examining their ability to convert sunlight into electrical energy under different lighting conditions and angles.Concurrently, battery performance undergoes scrutiny, encompassing capacity, charging speed, and overall reliability, including the effectiveness of the charging management system.

Furthermore, the vacuum cleaner's cleaning prowess is evaluated across diverse surfaces, such as carpets, hardwood floors, and outdoor areas, ensuring thorough cleaning coverage and optimal suction power. Navigation and mapping capabilities are put to the test to verify the vacuum's ability to navigate complex spaces, avoid obstacles, and execute efficient cleaning routes.

Autonomous operation is scrutinized for its scheduling accuracy, ability to return to the charging station, and seamless resumption of cleaning tasks post-recharging. Connectivity features, including remote control via smartphone apps or voice commands, are evaluated for responsiveness and reliability, along with the reception of over-the-air firmware updates.

Energy efficiency is closely monitored to gauge optimization of energy usage and prioritization of solar power. Durability and weather resistance are assessed through exposure to various environmental conditions, ensuring the vacuum's resilience to outdoor elements without compromising performance or safety.

The user interface undergoes usability testing to confirm clarity, intuitiveness, and effectiveness in providing real-time feedback on cleaning progress, battery status, and operational issues.

Lastly, safety features are meticulously evaluated to guarantee compliance with safety standards and regulations, including emergency stop mechanisms, obstacle detection sensors, and temperature monitoring systems to prevent accidents and ensure safe operation.

Debugging:

Embedded debugging may be performed at different levels, depending on the facilities available. From simplest to most sophisticate they can be roughly grouped into the following areas:

- Interactive resident debugging, using the simple shell provided by the embedded operating system (e.g. Forth and Basic)
- External debugging using logging or serial port output to trace operation using either a monitor in flash or using a debug server like the Remedy Debugger which even works for heterogeneous multi core systems.
- An in-circuit debugger (ICD), a hardware device that connects to the microprocessor via a JTAG or Nexus interface. This allows the operation of the microprocessor to be controlled externally, but is typically restricted to specific debugging capabilities in the processor.
- An in-circuit emulator replaces the microprocessor with a simulated equivalent, providing full control over all aspects of the microprocessor.
- A complete emulator provides a simulation of all aspects of the hardware, allowing all of it to be controlled and modified and allowing debugging on a normal PC.
- Unless restricted to external debugging, the programmer can typically load and run software through the tools, view the code running in the processor, and start or stop its operation. The view of the code may be as assembly code or source-code.

Because an embedded system is often composed of a wide variety of elements, the debugging strategy may vary. For instance, debugging a software(and microprocessor) centric embedded system is different from debugging an embedded system where most of the processing is performed by peripherals (DSP, FPGA, co-processor). An increasing number of embedded systems today use more than one single processor core.

CHAPTER 7
ADVANTAGES

- **Environmental Sustainability:**

  Perhaps the most significant advantage is their contribution to environmental sustainability. By harnessing solar energy, these cleaners reduce reliance on grid electricity, which is often generated from non-renewable sources. This helps decrease carbon emissions and environmental footprint, promoting cleaner air and a healthier planet.

- **Cost Savings:**

  Solar energy is essentially free once the initial investment in solar panels is made. Over time, users can enjoy significant savings on energy bills since they are no longer reliant on grid electricity to power their vacuum cleaners.

- **Energy Independence**:

  Solar-powered vacuum cleaners provide energy independence, especially in remote or off-grid locations where access to traditional power sources may be limited or unreliable. Users can rely on solar energy to power their cleaners, regardless of grid availability.

- **Extended Runtime**:

  Solar-powered cleaners often feature rechargeable battery systems that store excess solar energy for use during periods of low sunlight or at night. This extends the runtime of the vacuum cleaner, ensuring continuous operation even when solar energy is not available.

- **Eco-Friendly Image**:

  Using a solar-powered vacuum cleaner reflects a commitment to sustainability and environmental responsibility. This can enhance the eco-friendly image of individuals, businesses, or organizations, contributing to a positive reputation and social responsibility

DISADVANTAGE

- **Size and Weight**:

Solar panels add bulk and weight to the vacuum cleaner, which can affect portability and maneuverability, particularly for robotic or handheld models. Additionally, larger solar panels may be required to generate sufficient power, further increasing the size and weight of the device.

- **Market Availability:**

  Solar-powered vacuum cleaners may have limited availability compared to traditional electric-powered models, particularly in certain geographic regions or niche markets. As solar technology becomes more widespread and affordable, however, the availability of solar-powered cleaning solutions is expected to increase over time.

- **Maintenance Complexity**:

  Solar-powered vacuum cleaners may require additional maintenance compared to electric-powered models due to the integration of solar panels and battery systems. Regular cleaning and inspection of solar panels, as well as maintenance of battery health, are essential to ensure optimal performance and longevity.

- **Weather Sensitivity:**

  Solar panels and electrical components exposed to outdoor environments are susceptible to damage from weather conditions such as rain, snow, hail, and extreme temperatures. Proper weatherproofing and durability measures are necessary to protect the cleaner from environmental elements, adding complexity and cost to the design.

- **Dependence on Sunlight**:

  Solar-powered vacuum cleaners rely on sunlight to generate electricity, which means they may not be suitable for areas with limited sunlight or during cloudy days. In regions with long periods of overcast weather or high latitudes with shorter daylight hours, the efficiency of solar panels may be reduced, affecting the cleaner's performance.

APPLICATIONS

- **Residential Use:**

  Solar-powered vacuum cleaners are ideal for residential households looking to reduce energy costs and environmental impact. They can efficiently clean indoor spaces such as living rooms, bedrooms, kitchens, and bathrooms, providing a convenient and eco-friendly solution for routine cleaning tasks.
- **Outdoor Cleaning**:

  Solar-powered vacuum cleaners equipped with robust weatherproofing and durable construction are suitable for outdoor cleaning applications. They can be used to clean outdoor spaces such as patios, decks, sidewalks, parking lots, and recreational areas, providing a convenient and eco-friendly alternative to gas-powered or electric outdoor vacuum cleaners.
- **Commercial Buildings**:

  Solar-powered vacuum cleaners can be used in commercial buildings such as offices, hotels, hospitals, and retail stores to maintain cleanliness while reducing energy expenses. They can navigate large indoor areas efficiently, providing a sustainable cleaning solution for high-traffic environments.
- **Educational Institutions:**

  Solar-powered vacuum cleaners can be used in schools, colleges, and universities to promote sustainability and environmental awareness. They can serve as educational tools to teach students about renewable energy technology and the importance of energy conservation in everyday life.
- **Disaster Relief:**

  Solar-powered vacuum cleaners can be used in disaster relief efforts to clean up debris and debris from natural disasters such as hurricanes, earthquakes, and floods. Their ability to operate independently of grid infrastructure makes them valuable tools for rapid response and cleanup operations in areas without access to electricity.

FUTURE SCOPE

The future of smart vacuum cleaners incorporating solar panels holds immense promise, poised to revolutionize household cleaning with sustainable and efficient solutions. Anticipated advancements span various fronts, starting with enhancements in solar panel efficiency, ensuring optimal energy conversion from sunlight to power the vacuum cleaners. Concurrently, innovations in battery technology will bolster energy storage capabilities, enabling extended runtimes even during periods of low sunlight or at night. Moreover, the integration of advanced AI and machine learning algorithms will elevate cleaning performance by enabling the vacuum cleaners to adapt to their surroundings, optimize energy usage, and learn user preferences over time. These smart devices are expected to offer multifunctional capabilities beyond mere floor cleaning, with attachments for tasks such as mopping and air purification. Connectivity will be seamless through IoT integration, enabling remote control and monitoring via voice commands or mobile apps. Environmental sensing and mapping technologies will further enhance navigation and obstacle avoidance, ensuring precise and efficient cleaning. Modular designs will empower users to customize their vacuum cleaners according to specific needs, fostering flexibility and scalability. Sustainability will be a central focus, with eco-friendly materials and circular economy principles guiding product design and production. As these advancements converge, future solar-powered smart vacuum cleaners will offer users an unparalleled combination of efficiency, convenience, and environmental responsibility, transforming the way we clean our homes and workplaces.

CONCLUSIONS

This research facilitates efficient floor cleaning with sweeping operations. This robot works in Automatic modes. This proposed work provides the hurdle detection in case of any obstacle that comes in its way. The obsacle detection range is 30cm. If there is hurdle in the way of robot, it changes its direction. It reduces the labor cost and saves time also and provides efficient cleaning. In automatic mode, the robot operates autonomously. The operations such as changing the path in case of hurdle are performed automatically.

CHAPTER 7

REFERENCES

-] MUSALLAMI, S. A. R. (2021). A Smart Solar Powered Vacuum Cleaner: Solar Powered Vacuum Cleaner. Journal of Student Research.

- Khadka, N., Bista, A., Adhikari, B., Shrestha, A., & Bista, D. (2020, March). Smart solar photovoltaic panel cleaning system. In IOP Conference Series: Earth and Environmental Science (Vol. 463, No. 1, p. 012121). IOP Publishing

.

➢ Nado, M., & Taggu, A. (2020). Intelligent Automated Smart Solar Panel Using Internet of Things. In Proceedings of the 5th International Conference on Computers & Management Skills (ICCM 2019)| North Eastern Regional Institute of Science & Technology (NERIST), Nirjuli, Arunachal Pradesh, India.

➢ Manasa, M., Vidyashree, T. S., Bindushree, V., Rao, S., & Gowra, P. S. (2021). Smart vacuum cleaner. Global Transitions Proceedings, *2*(2), 553-558.

➢ Supriya, P. T., & Prakash, R. (2023). Solar Panel Cleaning System Using Smart Hybrid Embedded Systems with IoT Assistance. Electric Power Components and Systems, 1-12.

APPENDIX

SOURCE CODE:

```
char S;
#define m11 4
#define m12 5
#define m21 6
#define m22 7
#define TRIGGER 9
#define ECHO    10
#define buzzer 11
#define relay 2
#define relay1 3
void beep()
{
   digitalWrite(buzzer, HIGH);
   delay(1000);
    digitalWrite(buzzer, LOW);
    delay(1000);
}


void setup() {
  // put your setup code here, to run once:
 pinMode(TRIGGER, OUTPUT);
  pinMode(ECHO, INPUT);
pinMode (m11,OUTPUT);
pinMode (m12,OUTPUT);
pinMode (m21,OUTPUT);
pinMode (m22,OUTPUT);
pinMode (relay,OUTPUT);
pinMode (relay1,OUTPUT);
Serial.begin(9600);
 Serial.println("power up");
 delay(2000);

}

void loop() {

 long duration, d;
  digitalWrite(TRIGGER, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER, LOW);
  duration = pulseIn(ECHO, HIGH);
  d = (duration/2) / 29.1;//The speed of sound is: 343m/s = 0.0343 cm/uS = 1/29.1 cm/uS
  Serial.println(d);
 if(d < 40)
{

digitalWrite(m12,LOW);
digitalWrite(m11,LOW);
digitalWrite(m22,LOW);
digitalWrite(m21,LOW);
delay(900);

}
if(Serial.available())
  {
    S=Serial.read();
    Serial.println(S);
  }
 if(S=='f')
 {
digitalWrite(m11,HIGH);
digitalWrite(m12,LOW);
digitalWrite(m21,HIGH);
digitalWrite(m22,LOW);
delay(10);
Serial.println("ROBOT MOVING FORWARD");
delay(300);
}

else  if(S=='b')
{
digitalWrite(m11,LOW);
digitalWrite(m12,HIGH);
digitalWrite(m21,LOW);
digitalWrite(m22,HIGH);
delay(10);

Serial.println("ROBOT MOVING BACKWORD");

 }

else  if(S=='l')
{

digitalWrite(m11,HIGH);
digitalWrite(m12,LOW);
digitalWrite(m21,LOW);
digitalWrite(m22,HIGH);
delay(3000);

digitalWrite(m12,LOW);
digitalWrite(m11,LOW);
digitalWrite(m21,LOW);
digitalWrite(m22,LOW);
Serial.println("ROBOT MOVING LEFT");
```

```
 }

else  if(S=='r')
{
digitalWrite(m11,LOW);
digitalWrite(m12,HIGH);
digitalWrite(m21,HIGH);
digitalWrite(m22,LOW);

delay(3000);

digitalWrite(m12,LOW);
digitalWrite(m11,LOW);
digitalWrite(m21,LOW);
digitalWrite(m22,LOW);
Serial.println("ROBOT MOVING RIGHT");

 }

 else  if(S=='s')
 {
digitalWrite(m12,LOW);
digitalWrite(m11,LOW);
digitalWrite(m22,LOW);
digitalWrite(m21,LOW);
delay(10);
Serial.println("ROBOT MOVING STOPED");

 }
  else  if(S=='p')
 {
digitalWrite(relay,HIGH);

delay(10);
```

```
Serial.println("floor cleaning");
 }
  else  if(S=='q')
 {
digitalWrite(relay,LOW);

delay(100);

 }
 else  if(S=='r')
 {
digitalWrite(relay1,HIGH);
delay(100);
digitalWrite(relay1,LOW);
delay(100);
 }

}
char S;
#define m11 4
#define m12 5
#define m21 6
#define m22 7
#define TRIGGER 9
#define ECHO    10
#define buzzer 11
#define relay 2
#define relay1 3
void beep()
{
   digitalWrite(buzzer, HIGH);
   delay(1000);
    digitalWrite(buzzer, LOW);
    delay(1000);
```

```
}



void setup() {
  // put your setup code here, to run once:
 pinMode(TRIGGER, OUTPUT);
  pinMode(ECHO, INPUT);
pinMode (m11,OUTPUT);
pinMode (m12,OUTPUT);
pinMode (m21,OUTPUT);
pinMode (m22,OUTPUT);
pinMode (relay,OUTPUT);
pinMode (relay1,OUTPUT);
Serial.begin(9600);
 Serial.println("power up");
 delay(2000);

}

void loop() {

 long duration, d;
  digitalWrite(TRIGGER, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER, LOW);
  duration = pulseIn(ECHO, HIGH);
  d = (duration/2) / 29.1;//The speed of sound is: 343m/s = 0.0343 cm/uS = 1/29.1 cm/uS
  Serial.println(d);
 if(d < 40)
{

digitalWrite(m12,LOW);
digitalWrite(m11,LOW);
digitalWrite(m22,LOW);
digitalWrite(m21,LOW);
delay(900);

}
if(Serial.available())
  {
    S=Serial.read();
    Serial.println(S);
  }
 if(S=='f')
 {
digitalWrite(m11,HIGH);
digitalWrite(m12,LOW);
digitalWrite(m21,HIGH);
digitalWrite(m22,LOW);
delay(10);
Serial.println("ROBOT MOVING FORWARD");
delay(300);
}

else  if(S=='b')
{
digitalWrite(m11,LOW);
digitalWrite(m12,HIGH);
digitalWrite(m21,LOW);
digitalWrite(m22,HIGH);
delay(10);

Serial.println("ROBOT MOVING BACKWORD");

 }
```

```
else  if(S=='l')
{

digitalWrite(m11,HIGH);
digitalWrite(m12,LOW);
digitalWrite(m21,LOW);
digitalWrite(m22,HIGH);
delay(3000);

digitalWrite(m12,LOW);
digitalWrite(m11,LOW);
digitalWrite(m21,LOW);
digitalWrite(m22,LOW);
Serial.println("ROBOT MOVING LEFT");

 }

else  if(S=='r')
{
digitalWrite(m11,LOW);
digitalWrite(m12,HIGH);
digitalWrite(m21,HIGH);
digitalWrite(m22,LOW);

delay(3000);

digitalWrite(m12,LOW);
digitalWrite(m11,LOW);
digitalWrite(m21,LOW);
digitalWrite(m22,LOW);
Serial.println("ROBOT MOVING RIGHT");

 }
 else  if(S=='s')
 {
digitalWrite(m12,LOW);
digitalWrite(m11,LOW);
digitalWrite(m22,LOW);
digitalWrite(m21,LOW);
delay(10);
Serial.println("ROBOT MOVING STOPED");

 }
  else  if(S=='p')
 {
digitalWrite(relay,HIGH);

delay(10);
Serial.println("floor cleaning");
 }
  else  if(S=='q')
 {
digitalWrite(relay,LOW);

delay(100);

 }
 else  if(S=='r')
 {
digitalWrite(relay1,HIGH);
delay(100);
digitalWrite(relay1,LOW);
delay(100);
 }
```