Govt Polytecnic Srinagar Garhwal

REPORT OF

MAJOR PROJECT WORK

ON

Product Filteration Using Arduino

Submitted by

RAHUL KUMAR (22014090022)

VISHAL SINGH (23014092005)

SAKSHI BHATT (22014090028)

ANUJ NEGI(22014090008)

in the partial fulfillment of the requirements for the award of the diploma

in

ELECTRONICS ENGINEERING

Under the supervision of

Mrs. Seema Rawat (HOD,ELEX. Department)

DEPARTMENT OF

ELECTRONICS ENGINEERING

GOVT. POLYTECHNIC SRINAGAR GARHWAL UTTARAKHAND

Winter Semester 2024-25

CERTIFICATE

This is to certify that the project is a record of the work done by Rahul Kumar,Vishal Singh,Sakshi Bhatt,Anuj Negi in partial fulfillment of the requirements for the award of the Diploma in Electronics Engineering of the Government Polytechnic Srinagar (Garhwal) affiliated to Uttarakhand Board of Technical Education, (Roorkee). During the year 2024-25, under the supervision of Mrs. Seema Rawat department of Electronics Engineering.

Mrs. Seema Rawat

Mrs. Shilpi kannojia

Mr.Prabhakar Singh

Mr.Prashant Dobhal

Project Viva Held on……

DECLARATION

We hereby declare that this submission is our work and that to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or the university or other institute of higher learning, except where due acknowledgement has been made in the text.

RAHUL KUMAR (22014090022)
VISHAL SINGH (2301902005)
SAKSHI BHATT (22014090028)
ANUJ NEGI(22014090008)

ACKNOWLEDGEMENT

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crowns all efforts with success. We are grateful to our All faculties with project guide Mrs.Seema Rawat for the guidance, inspiration and constructive suggestions that helpful us in the preparation of this project.

I also thank my colleagues who have helped me in successful completion of the project.

ABSTRACTION

The student management system, a digital guardian of information, stands as a central hub for streamlining the academic journey. It abstracts away the burden of paperwork and manual processes, transforming data into a well-organized tapestry. At its core, it holds student details, weaving intricate threads of courses, grades, and attendance. Teachers find solace in streamlined lesson planning and grade recording, while students navigate their academic path with ease, accessing resources and monitoring progress. Communication flourishes under its watchful gaze, connecting parents, teachers, and administrators in a seamless flow of updates and announcements. This digital tapestry, meticulously woven, becomes an abstraction of academic life, offering efficiency, organization, and accessibility at every thread

TABLE OF CONTENT

INTRODUCTION AND USE OF PROJECT IN INDUSTRY OR IN SOCITY:

The use of color-based product filtration with Arduino in industrial applications combines cost-effective hardware and automation to filter or sort items based on color. This project leverages Arduino boards, color sensors, and actuators to create a scalable, efficient, and customizable solution for industries requiring sorting or quality control based on color attributes.

Below are some key industrial applications:

- **Food and Agriculture Industry**
- **Recycling and Waste Management**
- **Textile and Apparel Industry**
- **Pharmaceutical Industry**
- **Automotive Industry**
- **Packaging Industry**

Core Components for Arduino-Based Projects:

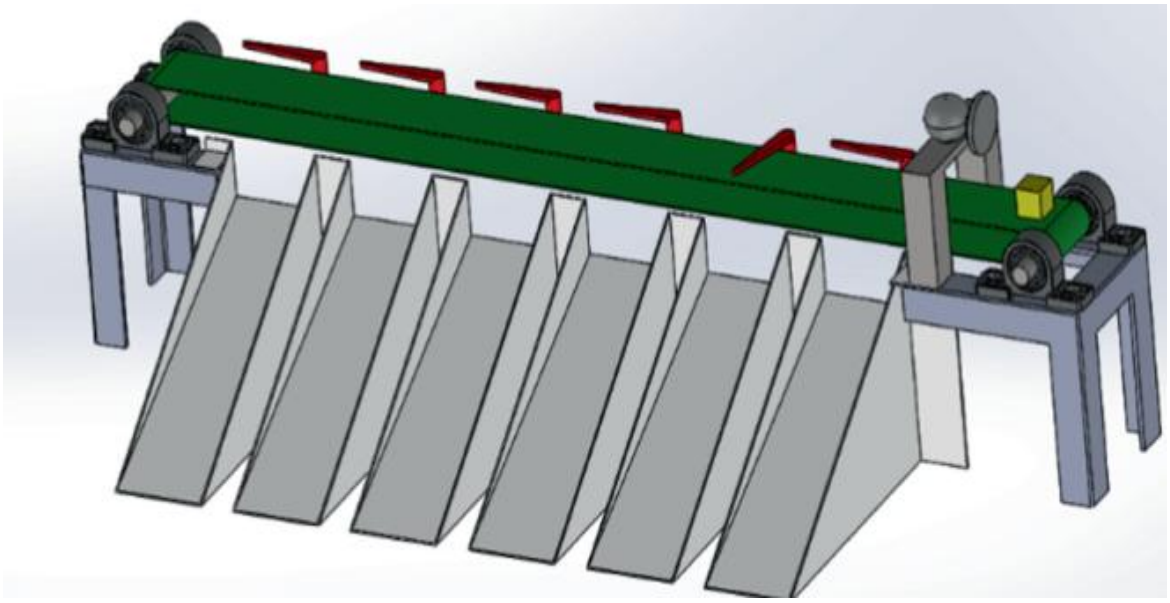
1. **Arduino Board**: Acts as the central controller (e.g., Arduino Uno or Mega).
2. **Color Sensor**: TCS34725, TCS230, TCS3200, APDS-9960, or similar sensors detect object colors.
3. **Actuators**: Servo motors, DC gear motors, stepper motors, or pneumatic arms to sort items.
4. **Conveyor System**: Moves products past the sensor.
5. **LCD with I2C Protocol**: Displays real-time product counts and system status.
6. **HM-10 BLE Module**: Provides wireless communication for sending start pulses to the pallet carrier system.
7. **Power Supply**: Provides energy to Arduino and actuators.
8. **Software**: Arduino IDE and libraries for programming and calibration.

Workflow:

1. **Detection**: The object passes under the color sensor.
2. **Processing**: Arduino interprets color data and matches it to predefined criteria.
3. **Action**: Based on the color match, actuators sort the object into the appropriate bin or area.
4. **Counting and Display**: The LCD module shows the count of products sorted for easy monitoring.
5. **Communication and Transport**: The HM-10 BLE module sends a start pulse signal to the pallet carrier, which then picks the sorted product and places it in its designated space.

Benefits in Industry:

- **Cost-Effectiveness**: Arduino-based systems are affordable and scalable.
- **Customization**: Easy to program and adapt for specific needs.
- **Automation**: Reduces manual labor and increases throughput.
- **Accuracy**: Consistently identifies and sorts based on predefined color parameters.
- **Real-Time Monitoring**: LCD with I2C ensures that product counts and system performance are easily visible.
- **Wireless Operation**: BLE communication minimizes wiring complexity and improves system flexibility.



COMPONENT DETAILS

1).Arduino Mega 2560 :

Arduino Mega 2560 is a development electronic board based on the Atmega2560 microcontroller.

This board is a good match for projects that require more GPIO pins and memory space because it carries 16 analog pins and 54 digital I/O pins out of which 15 pins are used for PWM output.

The board comes with a DC power jack to power up this unit and you can also turn on the board using VIN pin on the board. The unit also supports a USB interface where a USB cable is used to connect the board with the computer. The unit also supports the ICSP header which is used to program the board without disconnecting it from the main circuitry.

Two voltage regulators are included on the board through which you can regulate the voltage as you like better.

Arduino Mega 2560 is programmed using Arduino IDE (Integrated Development Environment) software that is the official software introduced by Arduino.cc

The ATmega2560 controller on the board comes with 256 KB of flash memory used for storing code (out of which 8 KB is used for the Bootloader), while the SRAM is 8 KB of SRAM and EEPROM is 4 KB of EEPROM.

Specification of ArduIno Mega 2560:

Technical Specifications	Description
Microcontroller	ATmega2560 microcontroller
Clock Speed	16 MHz
Architecture	AVR RISC (Reduced Instruction Set Computer) architecture
EEPROM	4 KB
Flash Memory	256KB
Operating Voltage	5V
Watchdog Timer	For system reset in case of failure
SRAM	8 KB
USART	Four UARTs for serial communication
Input/Output Pins	54 digital pins, and 16 analog inputs
Timers	6 timers (2 eight-bit and 4 sixteen-bit)
PWM Frequency	490 Hz

Arduino Mega 2560 Pin Description:

In this section, we'll cover the pin description of each pin incorporated on the board.

Digital I/O Pins: There are total of 54 digital I/O pins available on the board which can be used to connect the board with external components.

PWM: 15 pins are used for PWM which is a process used to control the speed of the motor or brightness of the LED.

LED: This is the built-in LED connected to pin 13. When 5V is provided to this pin, it will turn ON the LED while ground or zero V will turn it OFF.

Analogue Pins: There are 16 analogue pins incorporated on the board marked as A0 to A15. These pins can measure voltage from ground to 5V and each pin is a 10-bit resolution pin.

GND: This board carries 5 ground pins which are used for projects where more than one ground is required.

External Interrupts: Six pins are reserved for generating external interrupts. Those are pin number 0, 3, 18, 19, 20 & 21.

Reset: This is the reset pin of the board. This pin is useful when your code gets stuck in the middle of the running program, pressing this pin will reset the code compiled into the board.

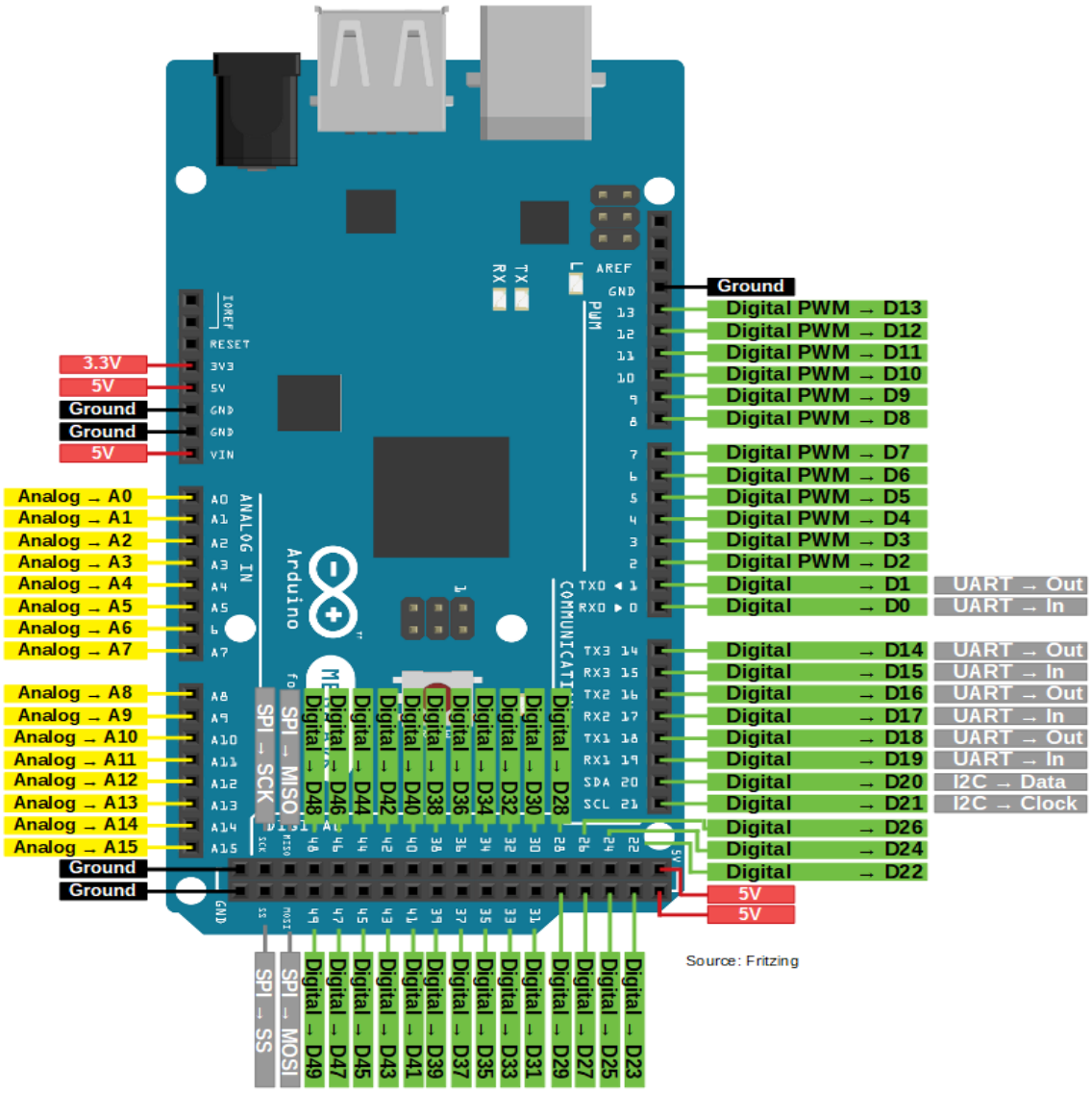
Vin: This is the input voltage of the board which ranges from 6V to 12V, however, recommended input voltage ranges from 7V to 12V.

AREF: This is the analogue reference voltage that is a reference voltage for the analogue inputs.

USART Communication: The board comes with USART serial communication where two pins TX and RX are used for the transmission and receiving of serial data.

SPI Communication: The device supports SPI (serial peripheral interface) communication which allows the transmission of data between the controller and other peripheral devices.

I2C Communication: The unit supports the I2C serial communication protocol where two pins 20 & 21 are reserved for this communication. The 20 is an SDA pin which is a serial data line used for holding the data and the 21 is an SCL pin which is a serial clock line used employed for offering data synchronization between the devices



Processor used in Arduino Mega:- ATmega 2560

The ATMEGA2560-16AU is an AVR 8-bit high-performance low-power microcontroller from ATMEL Corporation. The ATMEGA2560-16AU is a RISC (Reduced Instruction Set Computer) architecture microcontroller with 256kB of flash memory. The microcontroller IC also houses 4kB of EEPROM, 8kB of internal SRAM and 86 GPIO lines. The device also features three flexible timers/counter, serial USART, SPI port, 10-bit ADC and five programmable power saving modes. The operating voltage range is 1.8V to 5.5V. The A in the suffix signifies that the microcontroller comes in a TQFP package and the U signifies that the device has an “industrial” temperature range

Pin Number	Pin Name	Description
10, 31, 61, 80	V_{CC}	IC Supply pins
11, 32, 62, 81, 99	GND	IC ground reference pins
98	A_{REF}	Reference supply for ADC
100	A_{VCC}	Supply pin for analog peripherals
33, 34	XTAL	Crystal oscillator pins
30	$\overline{RESET}$	Reset pin, active low
2 - 9	PE0 – PE7	GPIO Port E pins
12 – 18, 27	PH0 – PH6, PH7	GPIO Port H pins
19 - 26	PB0 – PB7	GPIO Port B pins
28 – 29, 51 – 52, 70,1	PG3 – PG4, PG0 – PG1, PG2, PG5	GPIO Port G pins
35 - 42	PL0 – PL7	GPIO Port L pins
43 - 50	PD0 – PD7	GPIO Port D pins
53 - 60	PC0 – PC7	GPIO Port C pins
63 – 69, 79	PJ0 – PJ6, PJ7	GPIO Port J pins
71 - 78	PA7 – PA0	GPIO Port A pins
82 - 89	PK7 – PK0	GPIO Port K pins
90 - 97	PF7 – PF0	GPIO Port F pins

Pin Configuration

The pin description of the ATMEGA2560-16AU microcontroller is given in the following table:

Pin name	Pin description
Vcc, GND	Digital supply voltage and ground pins
PA7:0	Port A pins. Bi-directional I/O port with internal pull up resistors.
PB7:0	Port B pins. Bi-directional I/O port with internal pull up resistors.
PC5:0	Port C pins. Bi-directional I/O port with internal pull up resistors.

PD7:0	Port D pins. Bi-directional I/O port with internal pull up resistors.
PE7:0	Port E pins. Bi-directional I/O port with internal pull up resistors.
PF7:0	Analog input port for the ADC.
PH/J/K/L7:0	Port H/J/K/L pins. Bi-directional I/O port with internal pull up resistors.
AVcc	Supply voltage for ADC.
Aref	Analog reference pin for ADC.

Features of ATmega 2560:-

- 8-bit microcontroller
- RISC architecture
- Powerful instruction set
- Fully static operation
- 16 MIPS throughput at 16MHz
- On-chip 2 cycle multiplier
- 4kB EEPROM
- 8kB built-in SRAM
- 256kB flash memory
- Optional boot code section
- Firmware lock security feature
- Real time counter with separate oscillator
- 12 x PWM channels
- 16-channel 10-bit ADC
- 4 x Programmable serial USART
- SPI interface

- On-chip analog comparator
- Internal calibrated oscillator
- Support for power down modes

How to Program Arduino Mega 2560:

Arduino Mega 2560 can be programmed using Arduino IDE software which is an official Arduino software used to program all Arduino boards. This software is used for writing, compiling, and uploading the code into the Arduino board.

This unit comes with a USB interface so a USB cable can be used to connect the device with the computer through which you can transfer sketch (Arduino program is called a sketch) to the board. Moreover, this software is open-source which means it is free to use and anyone can use this software to allow the board to work as per the number of instructions you send from this software to the Arduino board.

The Arduino IDE (Integrated Development Environment) is a software application used to write, compile, and upload code to Arduino boards. It is a cross-platform tool, meaning it works on Windows, macOS, and Linux, and is based on the Processing IDE. The Arduino IDE is designed to be user-friendly, especially for beginners, and provides a simplified way to program Arduino boards using the C and C++ languages.

Here's a more detailed look at the Arduino IDE:

Key Features:

- **Text Editor:**

A text editor where you write your Arduino code, called a sketch, using the C/C++ language.

- **Compiler:**

Converts your code into a machine-readable format that the Arduino board can understand.

- **Uploader:**

Transfers the compiled code to the Arduino board via USB.

- **Serial Monitor:**

Allows you to communicate with your Arduino board through a serial interface, sending and receiving text data.

- **Library Manager:**

Enables you to install and manage third-party libraries that add functionality to your Arduino code.

- **Board Manager:**

Allows you to install and manage different Arduino boards and their configurations.

- **Example Sketches:**

A collection of pre-written code snippets that demonstrate various features of the Arduino platform.

How to Use:

1. **Install the IDE:** Download the Arduino IDE from the official website and install it on your computer.
2. **Connect your Arduino board:** Plug your Arduino board into your computer via USB.
3. **Open the IDE:** Launch the Arduino IDE application.
4. **Write your sketch:** Type your code into the text editor.
5. **Verify your code:** Use the "Verify" button to check for any syntax errors in your code.
6. **Upload your code:** Use the "Upload" button to transfer the code to your Arduino board.

7. **Use the Serial Monitor:** Open the Serial Monitor to interact with your Arduino board and see the output of your code.

In essence, the Arduino IDE provides a complete environment for programming Arduino boards, making it easier to write, compile, and upload your code, and to interact with your board using the Serial Monitor.

- Getting Started with Arduino

Applications:

This board can work as a stand-alone project or can be integrated with other Arduino boards, Arduino shields, and raspberry pi boards. This unit is recommended for electronic projects that require more memory space and GPIO pins.

The following are the applications of this Arduino mega board.

1. Embedded Systems: Arduino mega plays a pivotal role in embedded systems, serving as the brain behind countless electronic devices and appliances.

2. Medical Equipment: Its reliability and performance make it a preferred choice for medical devices, ensuring precise functionality in critical applications.

3. Home Automation: Arduino mega facilitates the automation of household tasks, enhancing convenience and efficiency in managing home appliances and systems.

4. Automotive Devices: In the automotive industry, Arduino mega contributes to the operation of various electronic components, ranging from engine control units to entertainment systems.

5. Industrial Automation: Its robust capabilities make it well-suited for industrial automation applications, enabling seamless control and monitoring of manufacturing processes.

6. Security Systems: Arduino mega forms the backbone of security systems, providing essential functionalities for surveillance cameras, access control systems, and alarm systems.

7. Temperature-Controlled Devices: Its ability to precisely control temperature makes it ideal for temperature-sensitive applications, such as climate control systems and incubators.

8. Motor Control Systems: Arduino mega facilitates efficient control of motors in diverse applications, from robotics to industrial machinery.

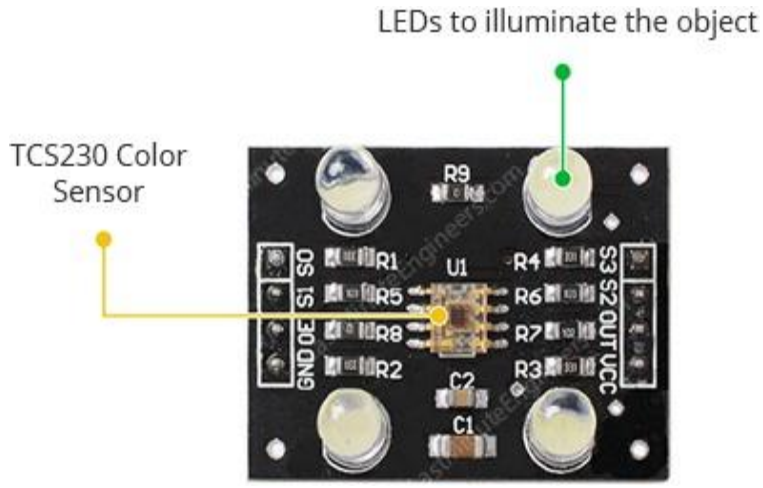
9. Digital Signal Processing: Its processing power enables it to handle digital signal processing tasks efficiently, making it valuable in audio processing, image processing, and communication systems.

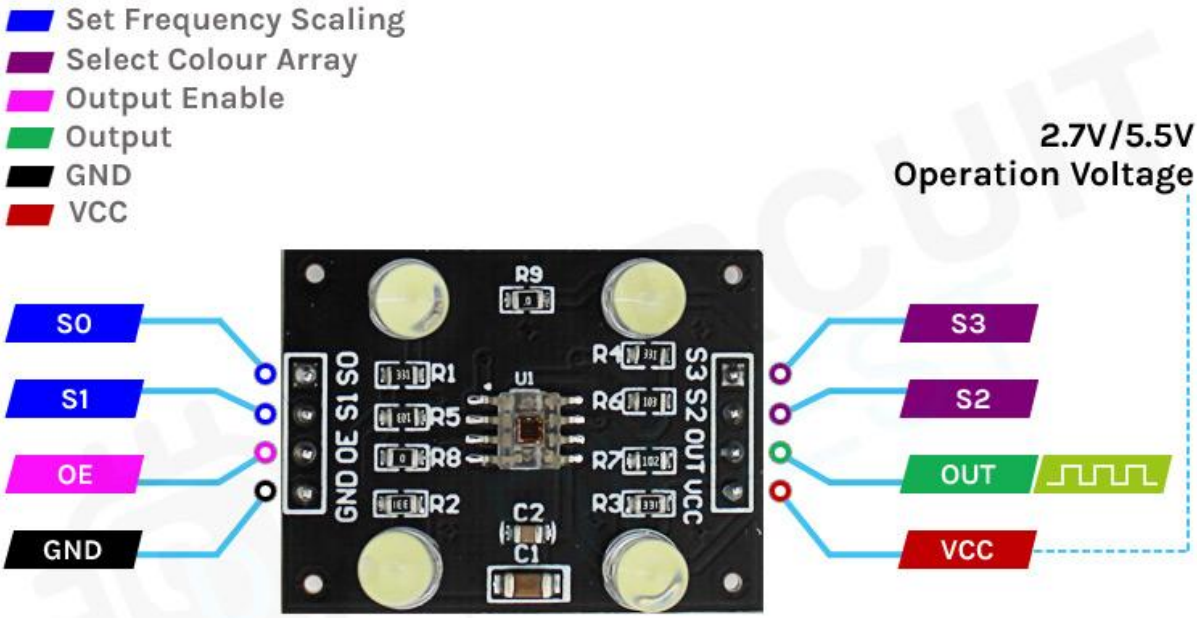
10. Peripheral Interface Systems: Arduino mega interfaces seamlessly with various peripherals, expanding its capabilities in interfacing with sensors, displays, and communication modules.

2).Color Sensor (TCS230/TCS3200):

Color sensors provide more reliable solutions to complex automation challenges. They are used in various industries including the food and beverage, automotive and manufacturing industries for purposes such as detecting material, detecting color marks on parts, verifying steps in the manufacturing process and so on.

The TCS230 color sensor (also branded as the TCS3200) is quite popular, inexpensive and easy to use.



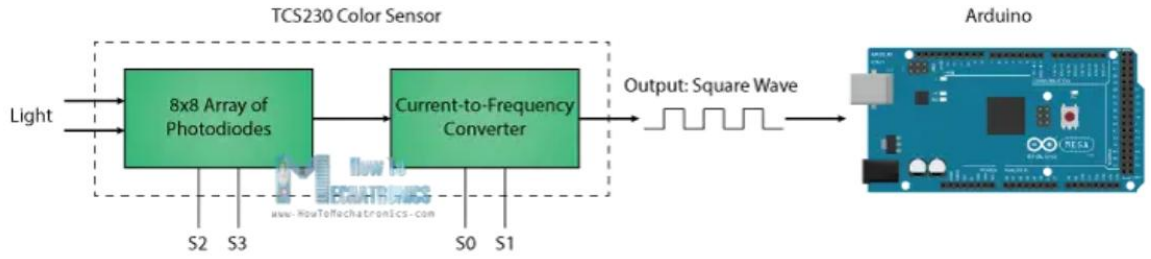


Specification of Color Sensor:

- Input voltage: (2.7V to 5.5V)
- Interface: Digital TTL
- High-resolution conversion of light intensity to frequency
- Programmable colour and full-scale output frequency
- No need of ADC(Can be directly connected to the digital pins of the microcontroller)
- Power down feature
- Working temperature: -40oC to 85oC
- Size: 28.4x28.4mm(1.12x1.12")

How TCS230 / TCS3200 Color Sensor Works:

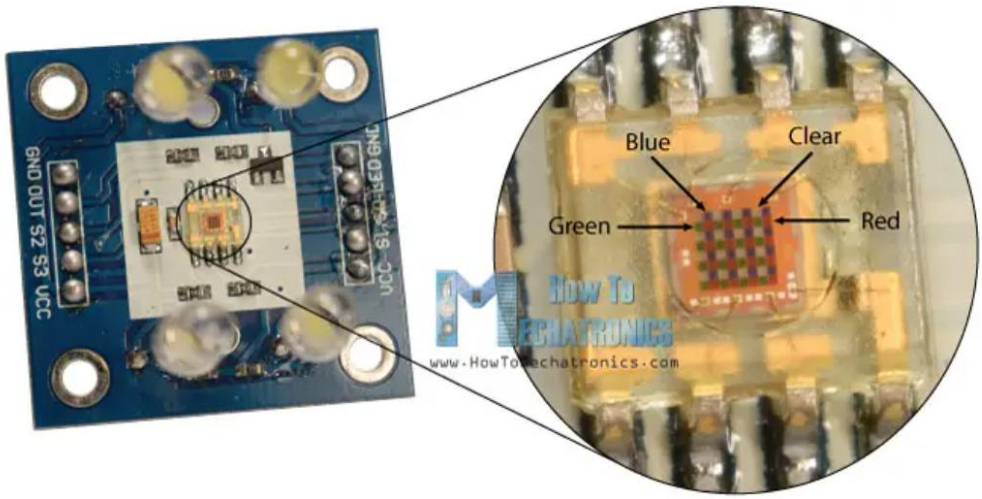
- The TCS230 senses color light with the help of an 8 x 8 array of photodiodes. Then using a Current-to-Frequency Converter the readings from the photodiodes are converted into a square wave with a frequency directly proportional to the light intensity. Finally, using the Arduino Board we can read the square wave output and get the results for the color.



- If we take a closer look at the sensor we can see how it detects various colors. The photodiodes have three different color filters. Sixteen of them have red filters, another 16 have green filters, another 16 have blue filters and the other 16 photodiodes are clear with no filters.

  Each 16 photodiodes are connected in parallel, so using the two control pins S2 and S3 we can select which of them will be read. So for example, if

we want to detect red color, we can just use the 16 red filtered photodiodes by setting the two pins to low logic level according to the table.

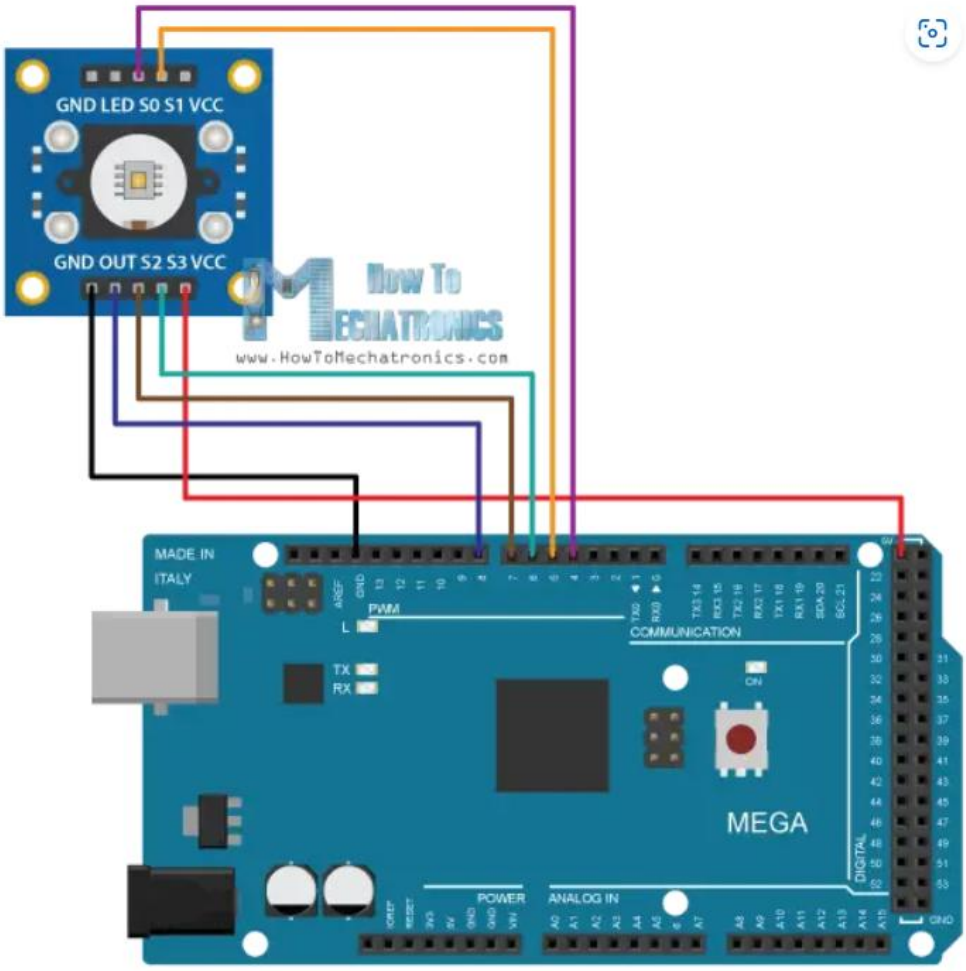


S0	S1	Output Frequency Scaling
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

S2	S3	Photodiode Type
L	L	Red
L	H	Blue
H	L	Clear (no filter)
H	H	Green

The sensor has two more control pins, S0 and S1 which are used for scaling the output frequency. The frequency can be scaled to three different preset values of 100 %, 20 % or 2%. This frequency-scaling function allows the output of the sensor to be optimized for various frequency counters or microcontrollers.

Now we are ready to move on and connect the TCS230 sensor to the Arduino board. Here's the circuit schematics.

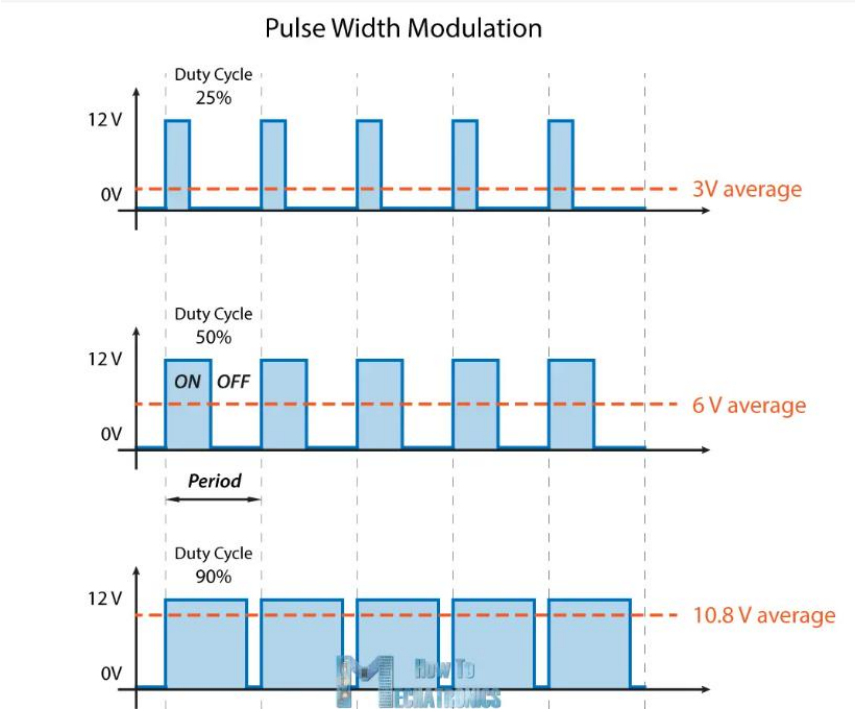

GND LED S0 S1 VCC
GND OUT S2 S3 VCC
How To
MECHATRONICS
www.HowToMechatronics.com
MADE IN
ITALY
PWM
L
TX
RX
COMMUNICATION
ON
MEGA
POWER
ANALOG IN
DIGITAL
GND

3).Control of Motors using L298N Motor Driver:

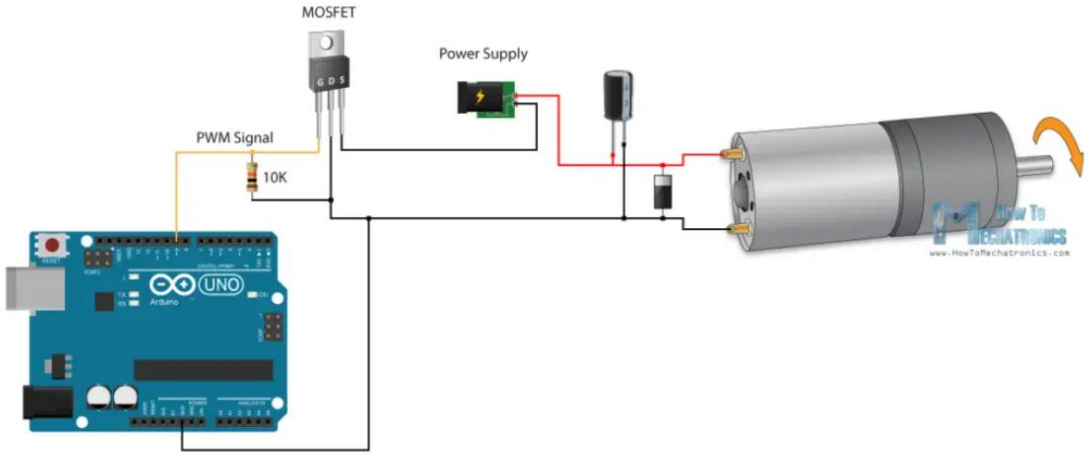
Motor speed control by PWM pin:-

We can control the speed of the DC motor by simply controlling the input voltage to the motor and the most common method of doing that is by using PWM signal.

- PWM DC Motor Control
- PWM, or pulse width modulation is a technique which allows us to adjust the average value of the voltage that's going to the electronic device by turning on and off the power at a fast rate. The average voltage depends on the duty cycle, or the amount of time the signal is ON versus the amount of time the signal is OFF in a single period of time.



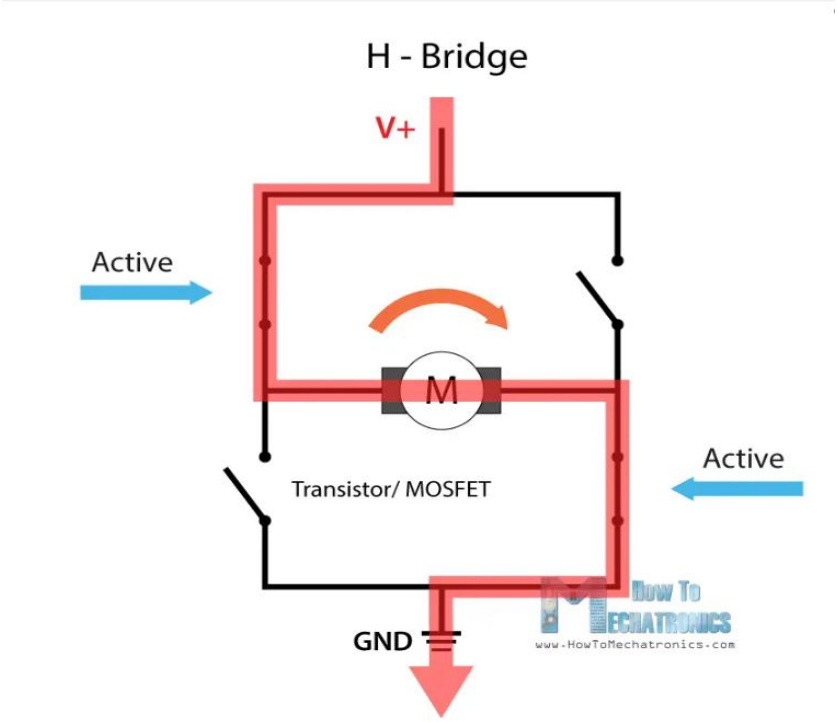
- So depending on the size of the motor, we can simply connect an Arduino PWM output to the base of transistor or the gate of a MOSFET and control the speed of the motor by controlling the PWM output. The low power Arduino PWM signal switches on and off the gate at the MOSFET through which the high power motor is driven.



Note: Arduino GND and the motor power supply GND should be connected together.

H-Bridge DC Motor Control

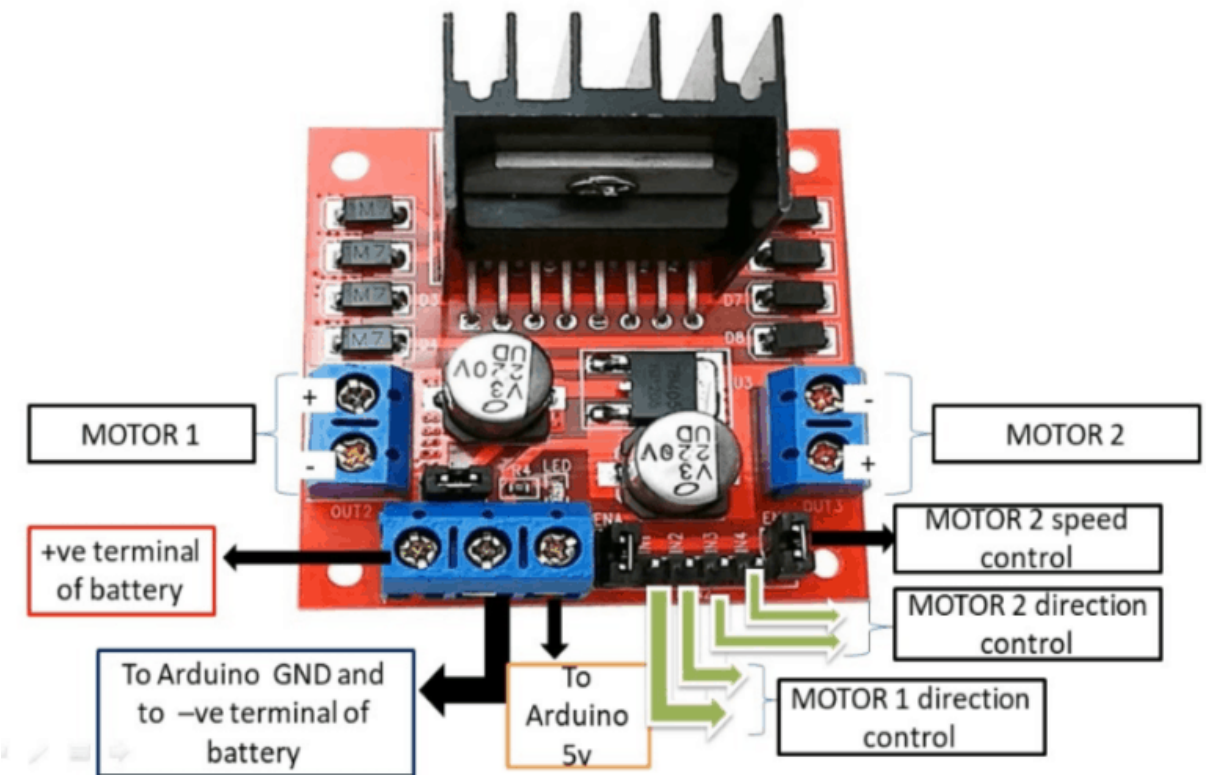
- On the other hand, for controlling the rotation direction, we just need to inverse the direction of the current flow through the motor, and the most common method of doing that is by using an H-Bridge. An H-Bridge circuit contains four switching elements, transistors or MOSFETs, with the motor at the center forming an H-like configuration. By activating two particular switches at the same time we can change the direction of the current flow, thus change the rotation direction of the motor.



- So if we combine these two methods, the PWM and the H-Bridge, we can have a complete control over the DC motor. There are many DC motor drivers that have these features and the L298N is one of them.

There are many DC motor drivers that have these features and the L298N is one of them.

This **L298N Motor Driver Module** is a high power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator. **L298N Module** can control up to 4 DC motors, or 2 DC motors with directional and speed control.

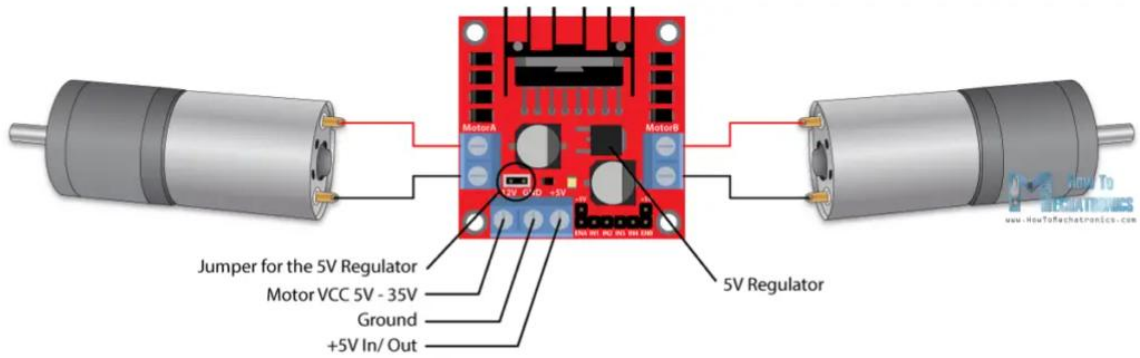


Specification of L298N Motor Driver:

- Driver Model: L298N 2A
- Driver Chip: Double H Bridge L298N

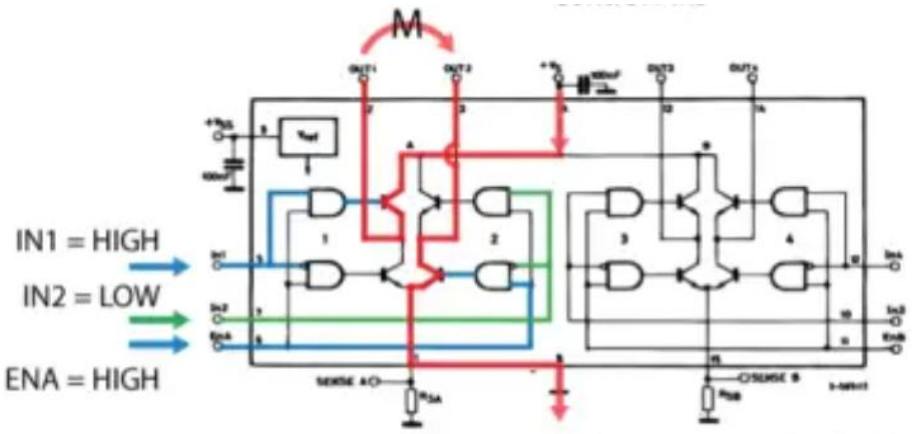
- Motor Supply Voltage (Maximum): 46V
- Motor Supply Current (Maximum): 2A
- Logic Voltage: 5V
- Driver Voltage: 5-35V
- Driver Current:2A
- Logical Current:0-36mA
- Maximum Power (W): 25W
- Current Sense for each motor
- Heatsink for better performance
- Power-On LED indicator

The module has two screw terminal blocks for the motor A and B, and another screw terminal block for the Ground pin, the VCC for motor and a 5V pin which can either be an input or output.

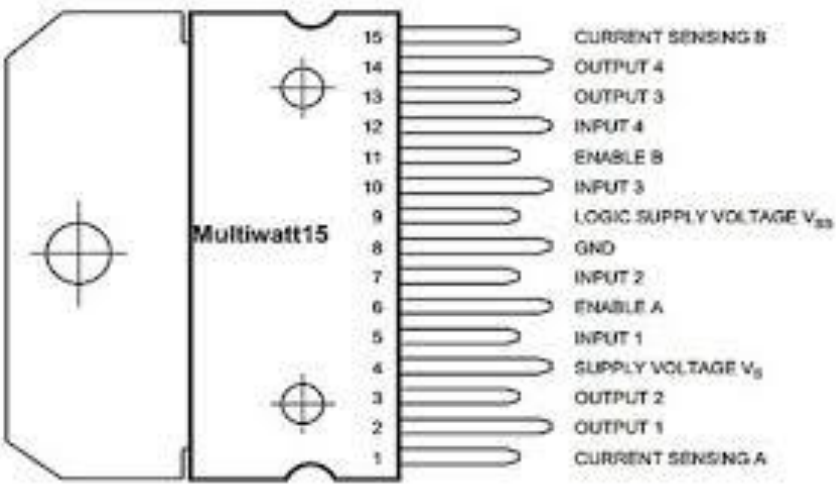


This depends on the voltage used at the motors VCC. The module have an onboard 5V regulator which is either enabled or disabled using a jumper. If the motor supply voltage is up to 12V we can enable the 5V regulator and the 5V pin can be used as output, for example for powering our Arduino board. But if the motor voltage is greater than 12V we must disconnect the jumper because those voltages will cause damage to the onboard 5V regulator. In this case the 5V pin will be used as input as we need connect it to a 5V power supply in order the IC to work properly.

We can note here that this IC makes a voltage drop of about 2V. So for example, if we use a 12V power supply, the voltage at motors terminals will be about 10V, which means that we won't be able to get the maximum speed out of our 12V DC motor.



L298N Block Diagram



Next are the logic control inputs. The Enable A and Enable B pins are used for enabling and controlling the speed of the motor. If a jumper is present on this pin, the motor will be enabled and work at maximum speed, and if we remove the jumper we can connect a PWM input to this pin and in that way control the speed of the motor. If we connect this pin to a Ground the motor will be disable. Next, the Input 1 and Input 2 pins are used for controlling the rotation direction of the motor A, and the inputs 3 and 4 for the motor B. Using these

pins we actually control the switches of the H-Bridge inside the L298N IC. If input 1 is LOW and input 2 is HIGH the motor will move forward, and vice versa, if input 1 is HIGH and input 2 is LOW the motor will move backward. In case both inputs are same, either LOW or HIGH the motor will stop. The same applies for the inputs 3 and 4 and the motor B.

4).12v DC Gear Motor:

DC Motors :- Two main types of DC motors exist: brushed and brushless, with permanent magnet DC (PMDC) motors also being a widely used motor type for simpler, cost-effective solutions. Brushed motors employ brushes and a commutator, whereas brushless motors use electronic controllers for smoother operation, reduced maintenance, and higher efficiency.

This efficiency typically ranges between 70% and 90%, a reflection of these motors' ability to maximize power output while minimizing energy loss.

Understanding these principles allows us to appreciate how DC motors effectively balance torque and efficiency, making them indispensable in various technical applications.

What is a Gearmotor?

A gearmotor, also known as a geared motor, is essentially a combination of a motor and a gearbox. Its design integrates these components to optimize performance and increase versatility in various applications. Gearmotors are often categorized based on the type of gearbox and motor they employ—most common motor types are DC or AC motors—each gearbox serving unique purposes and functions.

To fully appreciate the workings of a gearmotor, it is essential to understand its primary components:

1. **Motor:** The motor generates rotational motion. In most gearmotors, either a brushed or brushless DC motor is used. Brushed motors are known for their simplicity and cost-effectiveness, while brushless motors excel in efficiency and longevity.
2. **Gearbox:** This system of gears is responsible for altering the output torque and speed of the motor. Depending on the gear ratio employed, the gearbox can significantly enhance torque output while concurrently reducing the rotational speed. This adaptation makes gearmotors suitable for various tasks where high torque and controlled speed are vital.

3. **Mounting and Accessories:** Gearmotors are often equipped with various accessories, such as encoders for position feedback and mounting brackets that facilitate easy integration into machinery.

Basics of DC Gearmotors:

DC gearmotors are engineered to deliver high torque at reduced speeds, making them ideal for industrial applications requiring precise motor control, such as robotics and automation. By integrating a gearbox, these motors effectively transform the motor's speed and torque, enabling controlled motion and enhanced performance in demanding scenarios.

DC gear motors typically achieve an operational efficiency between 70% and 90%, contingent on their design and load conditions. This efficiency minimizes power loss, ensuring the motors' reliability and longevity.

DC Gearmotor Key Parameters:

When selecting a gear motor, we must consider key parameters: speed, measured in revolutions per minute (RPM); torque, expressed in pound-inches (lb-in); and efficiency (%). These factors must align with the application's

requirements to optimize motor control and achieve desired operational outcomes.

Working Principle:

Understanding the working principle of DC gear motors requires delving into the electromagnetic induction process that drives them. When direct current is applied to the motor, it generates a magnetic field within the stator. This magnetic field interacts with the rotor, creating a rotational motion through electromagnetic forces. The motor's rotation and the output shaft are the primary mechanisms that transform electrical energy into mechanical energy.

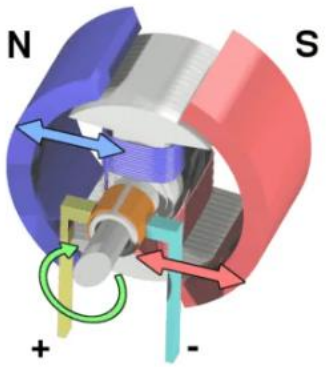


Figure 1 - Inside the DC Motor


Figure 2 - DC Motor

Incorporating a gearbox into this system effectively modifies the motor's output characteristics. The gearbox reduces the speed of the motor while simultaneously increasing its torque. This alteration is essential for applications demanding precise control and high torque at lower speeds. The gear mechanism achieves this by altering the gear ratio, thereby tailoring the speed and torque to specific requirements.

The interplay between speed and torque is vital in the operation of DC gear motors. By controlling these parameters, we can adapt the motor's

performance to the unique demands of various applications, such as robotics and automation.

The efficiency of this process typically ranges from 70-90%, making these motors both practical and versatile for challenging environments. Understanding these principles equips us to harness the full potential of DC gear motors effectively.

Key Motor Characteristics:

Speed and Torque:

Building on the relationship between speed and torque, let's explore the key characteristics that define a DC gear motor's performance. A fundamental aspect is the relationship between speed and torque, quantified in RPM and foot-pounds or kg-cm, respectively. These metrics are essential for evaluating motor performance, as they determine the motor's capacity to handle resistance and maintain desired operational velocities.

Gear Ratio / Gear Reduction:

In a DC small gear motor, gear reduction plays a significant role. By adjusting the gear ratio, such as 48:1, we can effectively reduce speed while amplifying torque. This balance is necessary for applications demanding precision and control, ensuring the motor delivers ideal performance under various load conditions.

Voltage and Current:

Voltage is another important parameter; DC motors typically operate at rated voltages to achieve peak efficiency. Operating below this threshold can diminish power output, while exceeding it may lead to overheating and potential burnout. The rated speed, achieved at this voltage, directly

influences application performance, dictating how well a motor can meet specific demands.

Understanding stall current and running current under no load is crucial for evaluating energy consumption and efficiency. Stall current marks the maximum power draw when stopped, contrasting with the lower running current during operation.

Types of DC Motors

DC motors, a staple in various engineering applications, come in several types, each tailored for specific needs and operational environments, including dc brushed motors that use conductive brushes for electrical commutation.

- Brushed Motor
- Brushless Motor (BLDC)
- Planetary Gear Motor
- Spur Gear Motor
- Stepper Motor
- Coreless & Coreless Brushless
- Servo

Brushed DC Motor for Cost-Effective Power:

Brushed DC motors use brushes and commutators to deliver straightforward, affordable power. While they do require maintenance due to brush wear, their simplicity makes them a dependable choice in applications where budget constraints and ease of operation take priority.

Brushless DC Motor (BLDC Motor) for High Efficiency

By removing brushes from the design, BLDC motors achieve superior efficiency and extended service life. Their low-maintenance nature and

precise control capabilities make them particularly suited for demanding applications that prioritize longevity and accuracy.

Planetary Gear DC Motor for Compact High Torque:

When DC motors are combined with compact planetary gearboxes, they deliver high torque output within space-limited environments. This integration is ideal for sophisticated automation tasks, where reliable torque is required without sacrificing a small footprint.

Spur Gear DC Motor for Reliable Motion:

Spur gear DC motors, sometimes referred to as parallel shaft motors, feature a straightforward gear arrangement that translates motor rotation into reliable motion. By pairing DC motors with spur gears, these setups offer balanced performance for general applications requiring moderate torque and consistent speed.

Stepper Motor for Precision Positioning:

Stepper motors move in discrete steps, making them the go-to choice for projects involving accurate positioning and repeatable movements. Their inherent design allows for controlled, incremental motion without the need for additional feedback systems.

Coreless DC Motor & Coreless Brushless DC Motor for Enhanced Performance:

Coreless motors eliminate the traditional iron core, reducing overall weight and inertia. This design yields smooth operation and rapid acceleration, with coreless brushless variants providing even greater efficiency and durability.

Servo Motor for Accurate Control:

Servo motors employ feedback mechanisms to deliver precise positional control, making them indispensable where accuracy and responsiveness are paramount. Shaft gear motors, a type of servo motor, are commonly used in robotics and automation, since they adjust output based on real-time input for consistently accurate performance.

Benefits of using a DC Gear Motor

DC gear motors can provide several unique benefits, making this type of motor better suited to many applications.

Some benefits of a gear motor include:

- **Increased torque:** The reduction mechanism increases the torque output, allowing the combined unit to power loads that require more torque than the motor alone can produce. For higher torque applications, you want to use metal gears, as opposed to plastic.
- **Reduced speed:** The gear reducer also lowers the rotational speed of the motor's output shaft, which is beneficial in applications where high speeds are neither required nor desirable.
- **More efficient power transfer:** Especially with planetary gearmotors, the gears inside the gearbox or speed reducer provide power efficiently.
- **Compact size and design:** Combined power solutions are designed to be space-saving, both inline and right angle options are available. You can get a small DC gear motor, which is beneficial if space is limited.
- **Greater flexibility:** The ability to adjust the gear ratio to provide the desired operating conditions allows for more control over the torque and speed output, which is helpful in various types of applications.

Selecting the Right Motor:

Selecting the right motor is vital for enhancing performance and efficiency in any application. In the motor selection process, we must carefully analyze speed and torque requirements to guarantee compatibility with the application's load demands. Determining these parameters allows us to match the motor's capabilities with the required performance criteria, guaranteeing peak functionality.

Evaluating Voltage and Current Ratings:

Equally important is evaluating the motor's voltage and current ratings to match the power supply. A motor operating efficiently within its rated specifications minimizes energy waste and enhances reliability. This alignment guarantees the motor's continuous operation without risking performance degradation or failure due to mismatched power parameters.

Dimensions and Configuration:

Physical dimensions and mounting configurations are significant considerations as well. We need to confirm the motor fits within the spatial constraints of our design. Customization options such as gear ratios and shaft configurations should be assessed to tailor the motor to our specific application needs, maximizing performance and efficiency.

Environmental Factors:

Finally, understanding environmental factors like noise levels and temperature ranges is vital. These considerations affect the motor's durability and suitability for intended operating conditions, guaranteeing longevity and consistent performance.

Common Applications:

When choosing the right motor, understanding its common applications can aid in making informed decisions that enhance system performance.

Robotics and Automation:

DC gear motors excel in scenarios demanding high torque and precise control, making them indispensable across various industries. In robotics, robots utilize these motors to drive wheels and actuators, providing the requisite torque and speed control necessary for intricate maneuvers and tasks. Their ability to deliver consistent performance in compact designs is essential for automated systems.

Automotive Industry:

In the automotive sector, dc gear motors are vital for applications like window lifts, seat adjusters, and windshield wipers, where controlled motion is paramount. Their reliability guarantees seamless operation, contributing to overall vehicle functionality.

Industrial Machinery:

Industrial machinery also benefits from these motors, particularly in conveyor belt systems, where they facilitate efficient material handling and transport with minimal power consumption.

Consumer Electronics:

Consumer electronics, including automated toys, power tools, home automation assistants, and small home appliances frequently utilize small gear motors to achieve reliable motion. Their compact and efficient design allows for enhanced user experience without compromising performance.

Medical Industry:

Additionally, in the medical field, medical equipment and devices such as precision surgical tools and powered wheelchairs rely on these motors for their ability to deliver precise control and quiet operation, assuring safety and reliability in critical applications.

Motor used in project :

A **12V DC gear motor** is a type of direct current (DC) electric motor that uses a gearbox to reduce the motor's speed and increase its torque output. The motor operates on a 12V DC power supply and is commonly used in various applications where moderate power, torque, and controlled speed are needed. The gear mechanism allows the motor to provide more force (torque) at a slower rotational speed, which is ideal for tasks like driving wheels in robotics, lifting mechanisms, or other machinery that requires precise motion control.(Recommended to be used with DC Motor Driver 20A or Dual DC Motor Driver 20A)

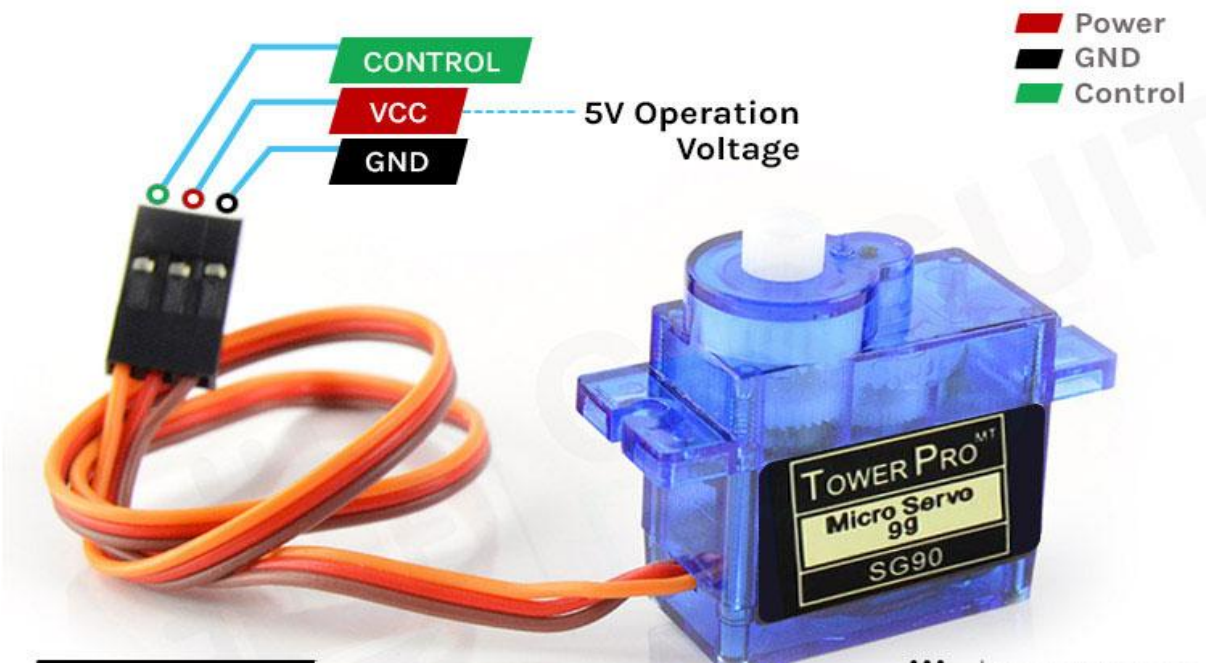
Specification of 12v Gear Motor:

- Power Supply: 12V DC
- RPM: 150
- Rated Torque: 4.7 kg-cm
- 6mm Dia shaft with M3 thread hole

- Gearbox diameter: 37 mm
- Motor Diameter: 28.5 mm
- Length without shaft: 63 mm
- Shaft length: 30mm
- Weight: 180gm
- No-load current = 800 mA, Load current = upto 7.5 A(Max)

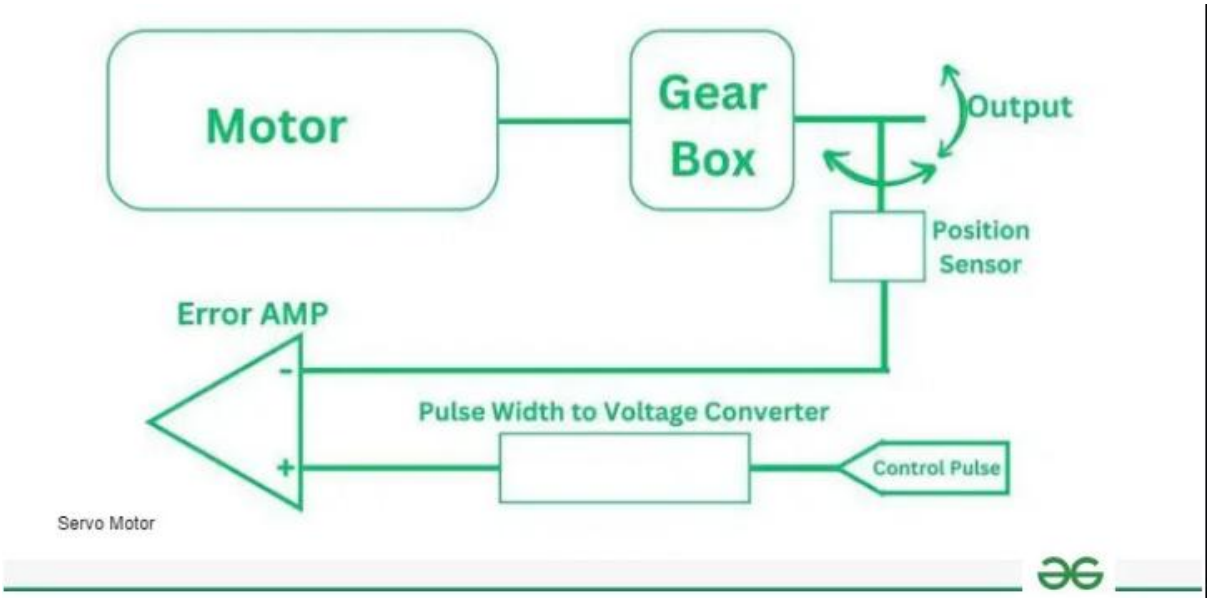
5).Servo Motor:

A **servo motor** is a type of motor used in various control systems where precise control of angular position, velocity, and acceleration is required. Servo motors are widely used in robotics, automation systems, CNC machinery, and radio-controlled (RC) vehicles, among other applications. Unlike regular motors, which rotate continuously, servo motors are designed to rotate to specific positions based on the control signal they receive.



What is a Servo Motor?

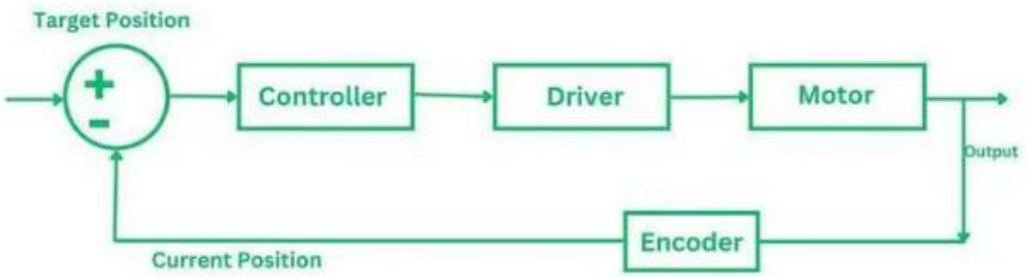
A servo motor is a rotary actuator that enables accurate control of angular position. It comprises a motor, a feedback system, and a controller. The feedback system constantly monitors the motor's actual position and adjusts it to match the desired position. The controller interprets the difference between the actual and desired positions, sending signals to the motor to correct any variations.



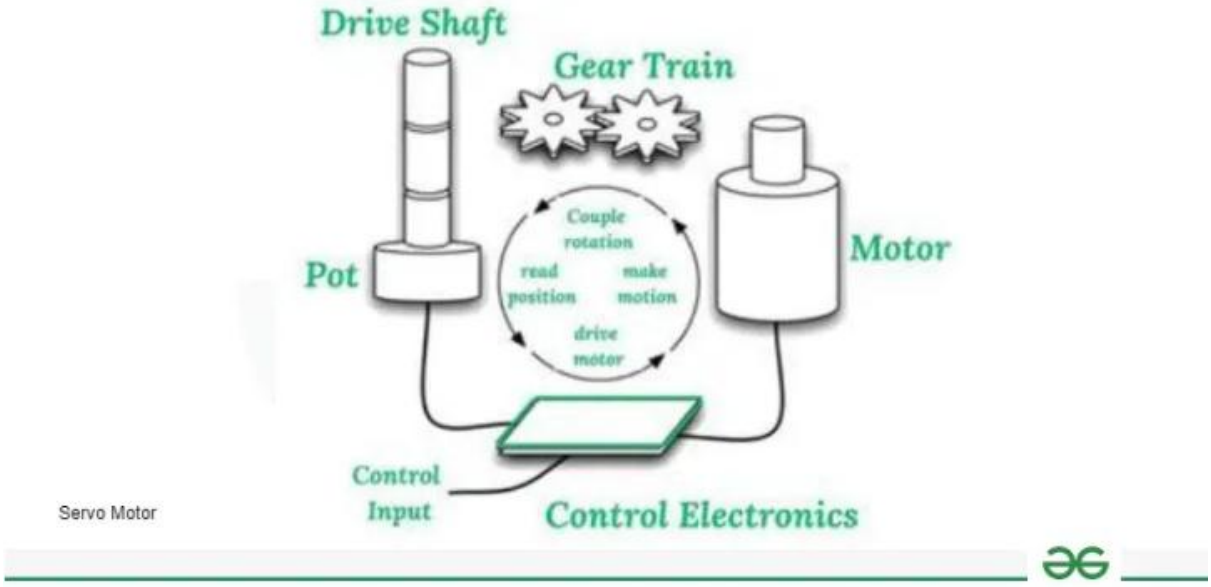
Construction of Servo Motor:

The construction of a servo motor involves many key components that are used to enable precise control of angular position. A servo motor is similar to a regular motor, but it has more additional parts to facilitate position control. These essential components include sensors, gears, and a circuit. The motor is guided by a controller, such as Arduino or STM. In industrial applications, AC servo motors utilize an encoder as a position sensor, while DC servo motors employ a potentiometer for this purpose.

A DC servo motor is assembled by combining a DC motor with various components like a gearbox, controller, and potentiometer. On the other hand, an AC servo motor uses an induction motor, complemented by gears and encoders for precise control.



Components of Servo Motor:



- **Rotor and Stator:** The core of a servo motor consists of two main parts: the rotor (the moving part) and the stator (the stationary part). The rotor is typically connected to the output shaft, responsible for generating motion.
- **Feedback Device:** Incorporated within the servo motor is a feedback device, often in the form of an encoder or resolver. This device constantly monitors the actual position of the rotor and provides this information to the controller.
- **Controller:** The controller is the most important part of the servo motor system. It interprets the feedback from the encoder and compares it to

the desired position. If there's any difference, the controller calculates the necessary adjustment.

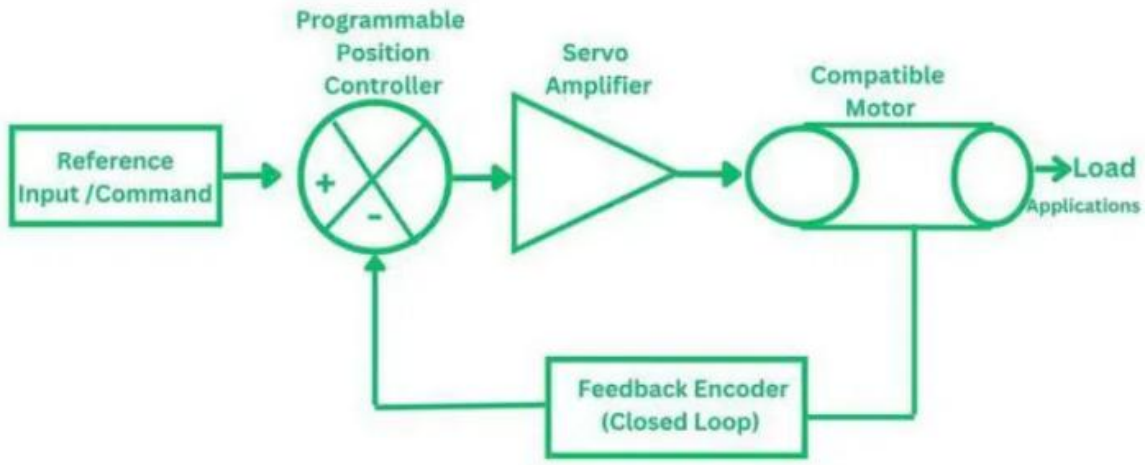
- **Control Input:** The servo motor receives control input, usually in the form of electrical pulses. The controller utilizes this input to determine how much and in which direction the motor should move.
- **Power Supply:** The motor requires a power supply, typically in the form of direct current (DC) or alternating current (AC), depending on the motor type.
- **Gear Train (optional):** In some servo motors, especially those used in robotics, a gear train may be included to amplify the torque or adjust the speed of the output shaft.

Working of Servo Motor:

A servo motor works in a simple way and is easy to understand. Usually, a servo motor has a system called closed-loop control. This system includes a comparator and a feedback path. It's like a setup that constantly checks and adjusts the motor to keep it in the right place. The comparator is an important part of the servo motor. It carefully checks where the motor is right now and compares it to where it's supposed to be. If there's a difference, it signals that there's an error, telling the motor to make the necessary adjustments to get to the correct position.

The block diagram below shows the components of a standard servo motor control system:

Servo motors are commonly controlled using a method called Pulse Width Modulation (PWM). This technique requires the transmission of an electrical signal containing pulses of different lengths to the motor. These pulses have a width that varies between 1 to 2 milliseconds, and they are sent repeatedly at a rate of 50 times per second to the servo motor. The adjustment of the pulse width serves as a means to effectively control the position of the rotating shaft in the servo motor. In simpler terms, changing the duration of these pulses guides the motor in achieving the desired position for its rotating shaft.

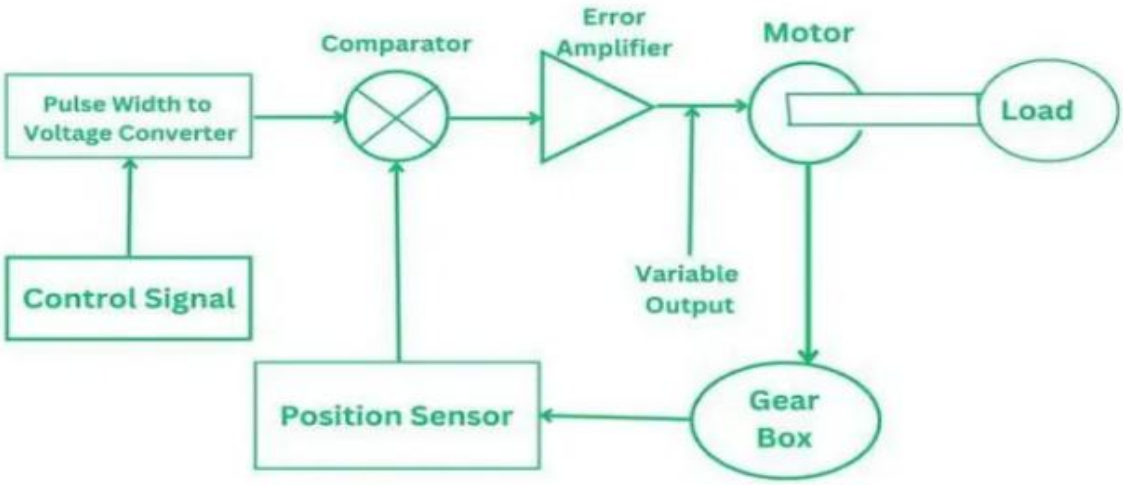


Types of Servo Motor

Servo motors are broadly classified into two categories depending on their power source:

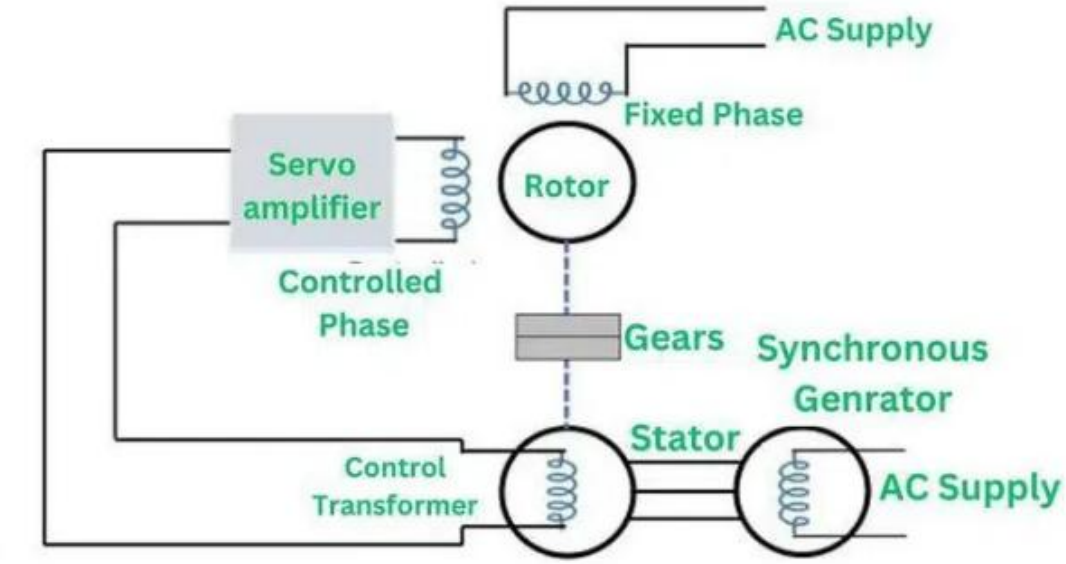
1).DC Servo Motor:

A DC servo motor consists of essential components like a DC motor, position sensor, gear assembly, and control circuit. This motor allows precise control of speed and position. To set the desired output, a DC reference voltage is determined using a potentiometer, pulse converter, or timers. In digital control, microprocessors generate PWM pulses for accuracy. Feedback, obtained through a potentiometer, guides an error amplifier, ensuring precise motor positioning. The amplifier compares current and desired positions, generating an error voltage that powers the motor until the error is zero, facilitating accurate rotation.



2).AC Servo Motor:

AC servo motors are a specific type of servomotor that converts AC electricity into precise mechanical movements, focusing on accurate angular velocity. Essentially, these motors are two-phase induction motors, featuring specific design distinctions. They produce mechanical power ranging from a few watts to several hundred watts, operating within a frequency range of 50 to 400 Hz. What sets them apart is their utilization of a closed-loop control system, employing encoders to monitor speed and position. This feature makes these motors exceptionally adept at precision and control, distinguishing them from others lacking such an advanced feedback system.



Characteristics of Servo Motor:

Servo Motors features several key characteristics that make them an excellent choice for applications that require precise control and accuracy. Here are some key characteristics of servo motors

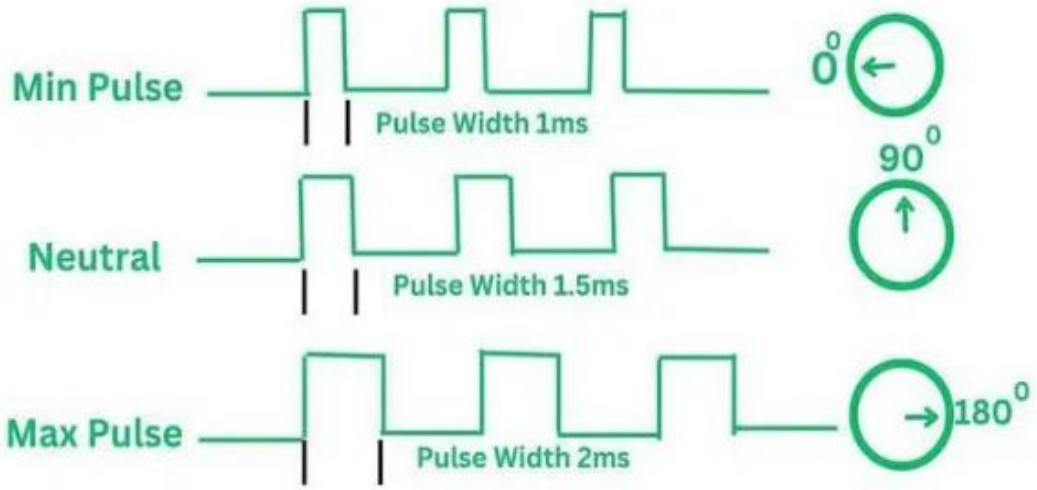
- **High Precision:** Servo motors provide precise control over position, speed, and torque. This precision is achieved through the use of feedback devices such as encoders, which continuously monitor the motor's actual position and provide feedback to the controller.

- **Fast Response Time:** Servo motors have an impressive response time, allowing them to quickly adjust their speed and position based on changing input signals.
- **High Torque:** Another characteristic of servo motors is their ability to deliver high torque even at low speeds. The high torque output of servo motors ensures that they can handle heavy loads and perform tasks with precision.
- **Closed-Loop Control:** Servo motors operate in a closed-loop control system, which means that they continuously receive feedback about their actual position and adjust their performance accordingly.
- **Wide Speed Range:** Servo motors offer a wide speed range, allowing them to operate at both high and low speeds without compromising performance.
- **Low Inertia:** Low rotor inertia enables quick acceleration and deceleration, contributing to the motor's dynamic performance.

How To Control a Servo Motor?

Servo motors are operated by transmitting an electrical pulse, known as pulse width modulation (PWM), through a control wire. This pulse has a variable width and consists of a minimum and maximum value, along with a repetition rate. The servo motor typically has a limited range of movement, usually 180° in total, with a capability of turning 90° in either direction from its neutral position. The neutral position is the point where the servo can rotate equally in both clockwise and counter-clockwise directions.

The position of the servo motor's shaft is determined by the duration of the PWM pulse sent through the control wire. The motor anticipates receiving a pulse every 20 milliseconds (ms), and the length of the pulse dictates how far the motor will turn. For instance, a 1.5ms pulse will position the motor at 90°. Pulses shorter than 1.5ms move the motor counter-clockwise toward the 0° position, while pulses longer than 1.5ms cause the servo to turn clockwise toward the 180° position.



Once directed to a specific position, these servos will maintain and resist any external force attempting to displace them. The extent of force a servo can withstand is referred to as its torque rating. It's essential to note that servos don't sustain their position indefinitely; a repetition of the position pulse is required to instruct the servo to stay in the designated position.

Interfacing Servo Motors with Microcontrollers:

In order to control servo motors, they need to be interfaced with microcontrollers, which act as the brain of the system. There are several methods to interface servo motors with microcontrollers. One common approach is to use pulse width modulation (PWM). PWM works by varying the width of the pulse signal to control the position of the servo motor. The microcontroller generates the PWM signal, which is then sent to the servo motor. By changing the pulse width, the microcontroller can control the angle at which the servo motor rotates. To interface a servo motor with a microcontroller, you will need to connect the servo motor to the appropriate pins of the microcontroller.

Servo Motors have three wires:

- Power
- Ground
- Signal

The power and ground wires are connected to a power source, while the signal wire is connected to a PWM pin on the microcontroller. Once the hardware connections are made, you can start programming the microcontroller to control the servo motor. This involves writing code that generates the PWM signal with the desired pulse width. The microcontroller will continuously send the PWM signal to the servo motor, causing it to rotate to the desired angle. It is important to note that different servo motors may have different operating characteristics, such as the range of angles they can rotate or the speed at which they can move. Therefore, it is essential to consult the datasheet or specifications of the servo motor to ensure proper interfacing and control.

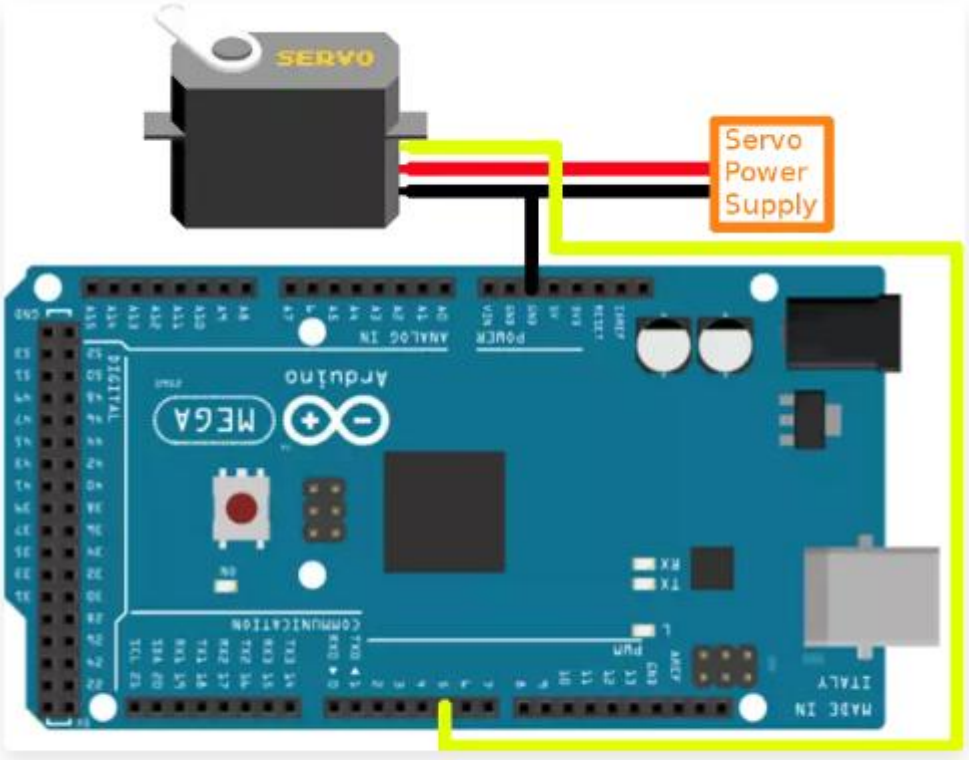


Figure 1 - Arduino Mega 2560 with Servo Motor

Applications of Servo Motors

Servo Motors have a wide range of applications across industries. Let's explore some of the common applications of servo motors:

- **Robotics:** Servo motors are used in robot arms, grippers, and joints to achieve accurate positioning and smooth motion. This enables robots to perform tasks with precision, such as assembly, welding, and material handling.
- **CNC Machines:** Servo motors are extensively used in Computer Numerical Control (CNC) machines. They control the movement of the cutting tools, ensuring precise and consistent machining operations. The servo motors enable high-speed positioning and accurate control over the cutting process, resulting in superior quality and productivity.
- **Industrial Automation:** They are used in conveyors, packaging machines, printing presses, and other automated equipment. The precise control offered by servo motors ensures efficient and reliable operation, improving productivity and reducing downtime.
- **Aerospace and Defense:** Servo motors are used in aircraft control surfaces, missile guidance systems, and unmanned aerial vehicles (UAVs). The high accuracy and responsiveness of servo motors enable accurate control of flight surfaces and guidance mechanisms, ensuring safe and reliable operation.
- **Electronics:** Servos are commonly used in electronic devices such as cameras, where they facilitate autofocus and image stabilization. They are also found in consumer electronics like DVD players and home automation systems.
- **Renewable Energy:** Servo motors are used in solar tracking systems to adjust the position of solar panels, optimizing their orientation to the sun for increased energy capture.

Specification of Servo Motor:

1. Operating Voltage

- Common values: **4.8V – 6V** for small hobby servos
- Industrial servos: **24V, 48V, or higher**

2. Torque

- Unit: **kg·cm** (kilogram-centimeter) or **N·m** (Newton-meter)
- Example: **4.5 kg·cm** @ **6V** (can hold 4.5 kg at 1 cm distance from shaft)

3. Speed

- Unit: **sec/60°** (time to rotate 60 degrees)
- Example: **0.1 sec/60°** @ **6V** (faster at higher voltages)

4. Angle of Rotation

- Typically **0° to 180°** for hobby servos
- Special servos: **360° continuous** or **limited custom angles**

5. Motor Type

- Brushed DC Motor (common in basic servos)
- Brushless DC Motor (for precision and long life)

6. Control Signal

- **Pulse Width Modulation (PWM)**
  - Pulse width: **1 ms to 2 ms** typically
  - Frame rate: **20 ms (50 Hz)**

7. Dimensions and Weight

- Depends on the model:
  - Micro servo: **~12g, small dimensions**
  - Standard servo: **~40g, ~40mm x 20mm x 40mm**

8. Gearing

- Plastic gears (cheap, light-duty)
- Metal gears (heavy-duty, durable)

9. Feedback Mechanism

- Potentiometer (basic models)
- Optical or magnetic encoders (high-end servos)

10. Bearing Type

- Bushings (cheaper)
- Ball bearings (better performance)

11. Stall Current

- Important for sizing the power supply.
- Can range from **500mA to 2A** or more in hobby servos.

12. Applications

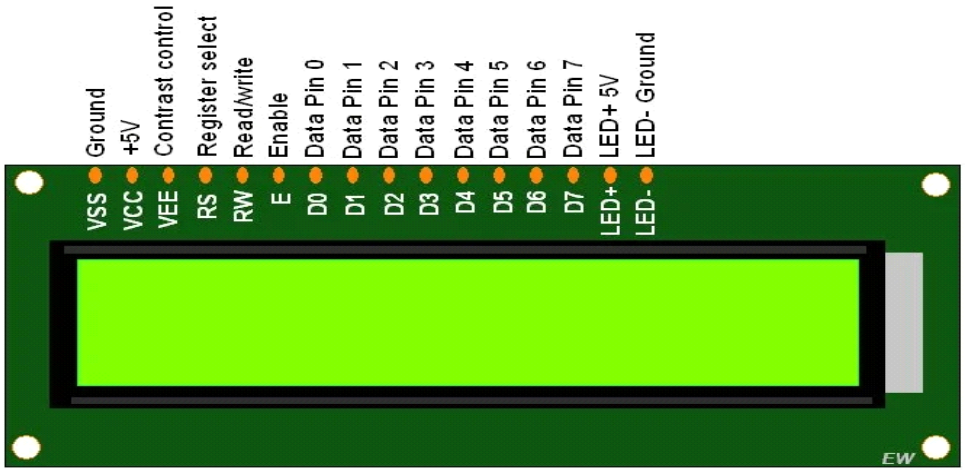
- Robotics
- Remote Control (RC) vehicles
- CNC machinery
- Automation systems

6).LCD Display:

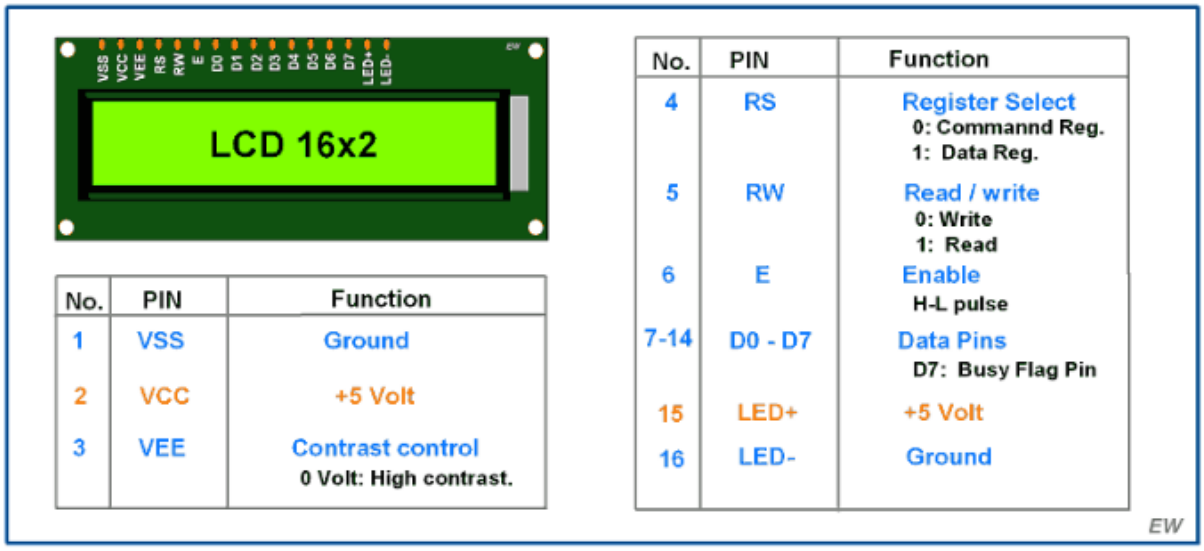
LCD **(Liquid Crystal Display)** is a type of flat panel display which uses liquid crystals in its primary form of operation. LEDs have a large and varying set of use cases for consumers and businesses, as they can be commonly found in smartphones, televisions, computer monitors and instrument panels.

LCDs were a big leap in terms of the technology they replaced, which include light-emitting diode (LED) and gas-plasma displays. LCDs allowed displays to be much thinner than cathode ray tube (CRT) technology. LCDs consume much less power than LED and gas-display displays because they work on the principle of blocking light rather than emitting it. Where an LED emits light, the liquid crystals in an LCD produces an image using a backlight.

As LCDs have replaced older display technologies, LCDs have begun being replaced by new display technologies such as OLEDs.



LCD 16x2 Pin Description:



No.	PIN	Function
1	VSS	Ground
2	VCC	+5 Volt
3	VEE	Contrast control 0 Volt: High contrast.

No.	PIN	Function
4	RS	Register Select 0: Commannd Reg. 1: Data Reg.
5	RW	Read / write 0: Write 1: Read
6	E	Enable H-L pulse
7-14	D0 - D7	Data Pins D7: Busy Flag Pin
15	LED+	+5 Volt
16	LED-	Ground

Pin 3 - VEE pin

This pin is used for adjusting the contrast of the display. Voltage on this pin defines contrast on display, lower the voltage, higher the contrast. We can connect 4.7 k pot for contrast adjustment or simply connect this pin to ground to get maximum contrast.

Pin 4 –RS: Register Select pin

- **RS = 0**: Data on the D0 to D7 pins is considered as a command**.**

- **RS = 1:** Data on the D0 to D7 pins is considered as data to display on LCD16x2.

Pin 5 – RW: Read / Write pin

- **RW = 0:** Write data to the LCD
- **RW = 1:** Read data from the LCD

Pin 6 –E: Enable

This pin is used to latch the data present on the data pins D0 to D7. High to low pulse with a minimum width of 450 ns is required to latch the data to the display.

Pins 7:14 - DATA pins D0 to D7

Data pins are used to send data/command to the LCD16x2 as parallel 8 data bits.

Pin 15:16 - LED + and LED -

Liquid Crystal Displays don't have their own light like seven segment displays. Therefore, the module has a backlight LED. Supply to this LED is provided through these pins.

Specification of LCD16x2:

1. Display Type: Alphanumeric character display
2. Character Format: 5x8 dots matrix format
3. Display Size: 16 characters x 2 lines
4. Display Color: Blue or Green
5. Backlight: LED backlight
6. Voltage Supply: 5V DC
7. Operating Temperature: -20°C to +70°C
8. Interface: 4-bit or 8-bit mode
9. Dimension: 84.0 x 44.0 x 13.0 mm

LCD 16x2 Commands:

While interfacing an LCD16x2 with any microcontroller, firstly we need to initialize the LCD. For that, we need to send some commands. Similarly, to clear the display or for changing the position we need to send commands. So basically, we can say that LCD16x2 is controlled by using commands.

Commonly Used LCD16x2 Commands

Code (HEX)	Command to LCD	Execution Time
0x01	Clear the display screen	1.64ms
0x06	Shift the cursor right (e.g. data gets written in an incrementing order, left to right)	40 us
0x0C	Display on, cursor off	40 us
0x0E	Display on, cursor blinking	40 us
0x80	Force the cursor to the beginning of the 1st line	40 us
0xC0	Force the cursor to the beginning of the 2nd line	40 us
0x10	Shift cursor position to the left	40 us
0x14	Shift cursor position to the right	40 us
0x18	Shift entire display to the left	40 us
0x1C	Shift entire display to the right	40 us

0x38	2 lines, 5x8 matrix, 8-bit mode	40 us
0x28	2 lines, 5x8 matrix,4-bit mode	40 us
0x30	1 line, 8-bit mode	40us
0x20	1 line, 4-bit mode	40us

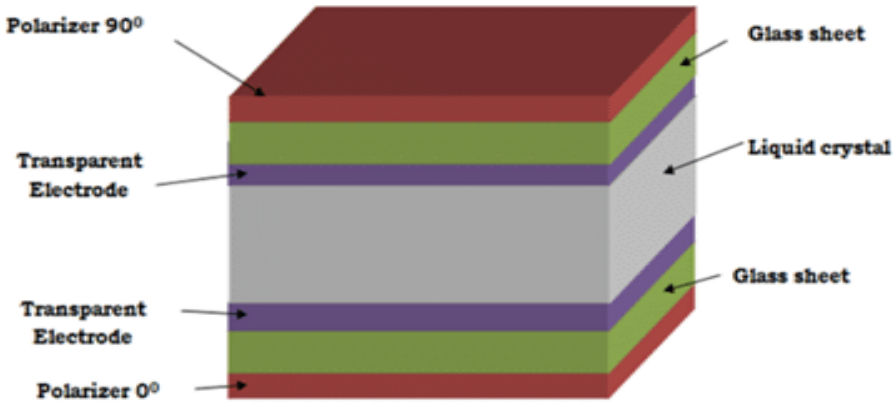
Now, while printing a character on LCD16x2, we need to send the ASCII code of that character to LCD16x2. Suppose, we want to print a character 'H' on the LCD, then we should send 0x48 (ASCII code of 'H') data to the LCD16x2. The LCD16x2 has its own controller, which does the printing job on the LCD16x2.

Construction of Liquid Crystal Display:

The LCD is constructed from two pieces of polarized glass. There are two electrodes used: a positive electrode and a negative electrode.

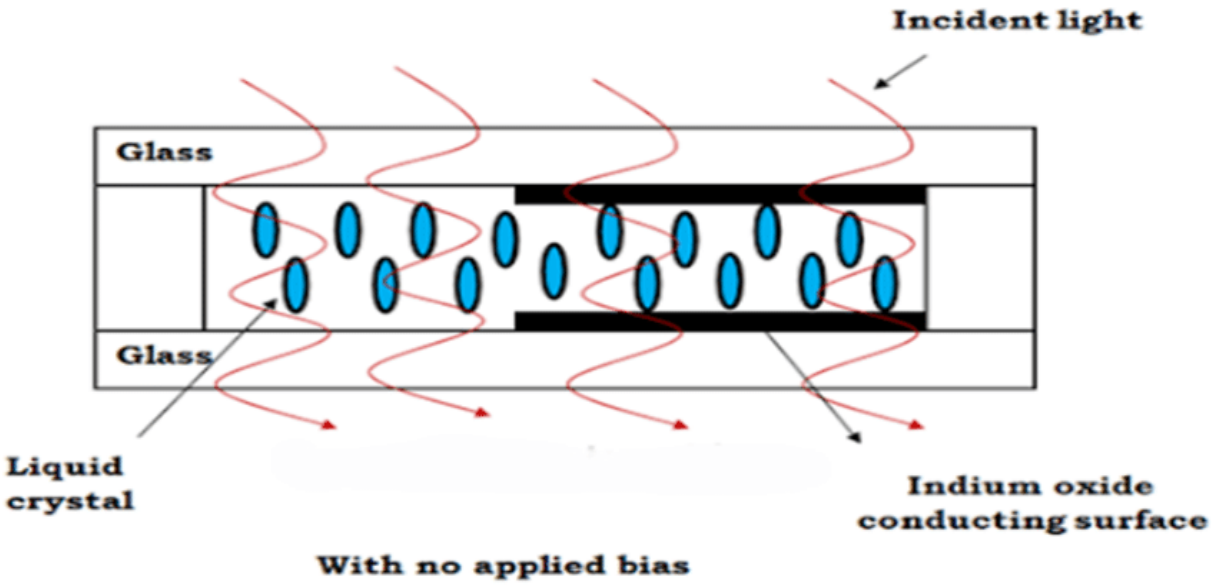
External voltage is applied to the LCD to LCD using these electrodes and it is made up of indium-tin-oxide. A 10–20 µm liquid crystal layer is sandwiched between two sheets of glass.

By altering the polarization, light can be transmitted through or prevented.

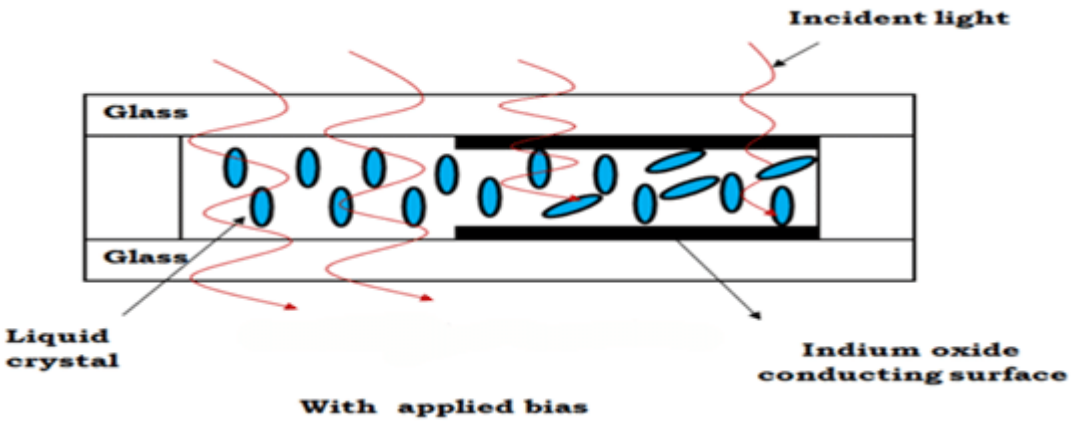


Working Principle:

The basic working principle of LCD is obstruction of light. It cannot generate light by itself. Thus, an external light source is required. When external light moves from one polarizer to the next, a liquid crystal receives an external supply, and the polarized light aligns itself to form an image on the screen.



The transparent layer on each side of the sealed thick layer of liquid crystal is the indium oxide conducting surface. The molecular arrangement is unaffected in the absence of any external bias.



The molecular arrangement changes when an external bias occurs, making one area appear dark and the other area appear clea

How LCDs work:

A display is made up of millions of pixels. The quality of a display commonly refers to the number of pixels; for example, a 4K display is made up of 3840 x2160 or 4096x2160 pixels. A pixel is made up of three subpixels; a red, blue and green—commonly called RGB. When the subpixels in a pixel change color combinations, a different color can be produced. With all the pixels on a display working together, the display can make millions of different colors. When the pixels are rapidly switched on and off, a picture is created.

The way a pixel is controlled is different in each type of display; CRT, LED, LCD and newer types of displays all control pixels differently. In short, LCDs are lit by a backlight, and pixels are switched on and off electronically while using liquid crystals to rotate polarized light. A polarizing glass filter is placed in front and behind all the pixels, the front filter is placed at 90 degrees. In between both filters are the liquid crystals, which can be electronically switched on and off.

LCDs are made with either a passive matrix or an active matrix display grid. The active matrix LCD is also known as a thin film transistor (TFT) display. The passive matrix LCD has a grid of conductors with pixels located at each intersection in the grid. A current is sent across two conductors on the grid to control the light for any pixel. An active matrix has a transistor located at each pixel intersection, requiring less current to control the luminance of a pixel. For this reason, the current in an active matrix display can be switched on and off more frequently, improving the screen refresh time.

Some passive matrix LCD's have dual scanning, meaning that they scan the grid twice with current in the same time that it took for one scan in the original technology. However, active matrix is still a superior technology out of the two.

7).I2C Module/Protocol:

I2C stands for Inter-Integrated Circuit. It is a bus interface connection protocol incorporated into devices for serial communication. It was originally designed by Philips Semiconductor in 1982. Recently, it is a widely used protocol for short-distance communication. It is also known as Two Wired Interface(TWI).

Working of I2C Communication Protocol:

It uses only 2 bi-directional open-drain lines for data communication called SDA and SCL. Both these lines are pulled high.

Serial Data (SDA) : Transfer of data takes place through this pin.

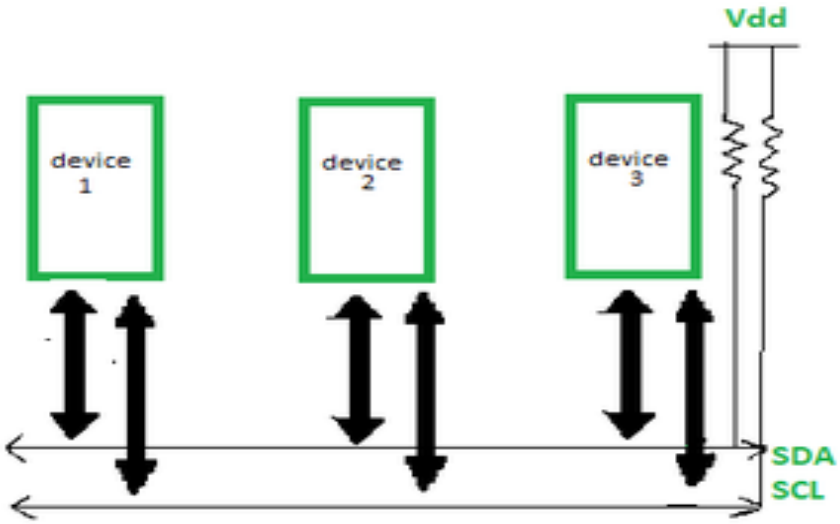
Serial Clock (SCL) : It carries the clock signal.

I2C operates in 2 modes

Master mode

Slave mode

Each data bit transferred on SDA line is synchronized by a high to the low pulse of each clock on the SCL line.



According to I2C protocols, the data line can not change when the clock line is high, it can change only when the clock line is low. The 2 lines are open drain, hence a pull-up resistor is required so that the lines are high since the devices on the I2C bus are active low. The data is transmitted in the form of packets which comprises 9 bits. The sequence of these bits are –

Start Condition: 1 bit

Slave Address: 8 bit

Acknowledge: 1 bit

Steps of I2C Data Transmission?

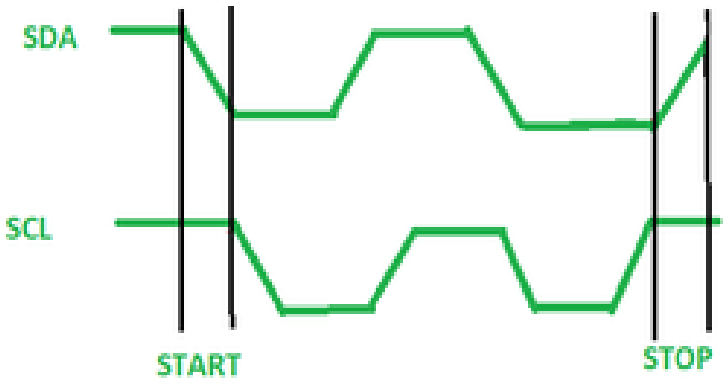
Here are the steps of I2C (Inter-Integrated Circuit) data transmission

- **Start Condition:** The master device sends a start condition by pulling the SDA line low while the SCL line is high. This signals that a transmission is about to begin.
- **Addressing the Slave:** The master sends the 7-bit address of the slave device it wants to communicate with, followed by a read/write bit. The read/write bit indicates whether it wants to read from or write to the slave.
- **Acknowledge Bit (ACK):** The addressed slave device responds by pulling the SDA line low during the next clock pulse (SCL). This confirms that the slave is ready to communicate.

- **Data Transmission:** The master or slave (depending on the read/write operation) sends data in 8-bit chunks. After each byte, an ACK is sent to confirm that the data has been received successfully.
- **Stop Condition**: When the transmission is complete, the master sends a stop condition by releasing the SDA line to high while the SCL line is high. This signals that the communication session has ended.

What is Start and Stop Conditions ?

START and STOP can be generated by keeping the SCL line high and changing the level of SDA. To generate START condition the SDA is changed from high to low while keeping the SCL high. To generate STOP condition SDA goes from low to high while keeping the SCL high, as shown in the figure below.



start and stop condition

Start and Stop Condition

What is Repeated Start Condition?

Between each start and stop condition pair, the bus is considered as busy and no master can take control of the bus. If the master tries to initiate a new transfer and does not want to release the bus before starting the new transfer, it issues a new START condition. It is called a REPEATED START condition.

Read/Write Bit:

A high Read/Write bit indicates that the master is sending the data to the slave, whereas a low Read/Write bit indicates that the master is receiving data from the slave.

ACK/NACK Bit:

After every data frame, follows an ACK/NACK bit. If the data frame is received successfully then ACK bit is sent to the sender by the receiver.

Addressing:

The address frame is the first frame after the start bit. The address of the slave with which the master wants to communicate is sent by the master to every slave connected with it. The slave then compares its own address with this address and sends ACK.

I2C Packet Format:

In the I2C communication protocol, the data is transmitted in the form of packets. These packets are 9 bits long, out of which the first 8 bits are put in SDA line and the 9th bit is reserved for ACK/NACK i.e. Acknowledge or Not Acknowledge by the receiver.

START condition plus address packet plus one more data packet plus STOP condition collectively form a complete Data transfer.

Features of I2C Communication Protocol:

- **Half-duplex Communication Protocol –** Bi-directional communication is possible but not simultaneously.
- **Synchronous Communication –** The data is transferred in the form of frames or blocks.
- Can be configured in a multi-master configuration.
- **Clock Stretching –** The clock is stretched when the slave device is not ready to accept more data by holding the SCL line low, hence disabling

the master to raise the clock line. Master will not be able to raise the clock line because the wires are AND wired and wait until the slave releases the SCL line to show it is ready to transfer next bit.

- **Arbitration –** I2C protocol supports multi-master bus system but more than one bus can not be used simultaneously. The SDA and SCL are monitored by the masters. If the SDA is found high when it was supposed to be low it will be inferred that another master is active and hence it stops the transfer of data.
- **Serial transmission –** I2C uses serial transmission for transmission of data.
- Used for low-speed communication.

Advantages of I2C Communication Protocol

- Can be configured in multi-master mode.
- Complexity is reduced because it uses only 2 bi-directional lines (unlike SPI Communication).
- Cost-efficient.
- It uses ACK/NACK feature due to which it has improved error handling capabilities.
- Fewer Wires: Only two wires are needed, making it easier to set up.
- Multiple Devices: You can connect many devices to the same bus.
- Simple Communication: It’s relatively easy to program and use.

Why we use I2C Module for connection of LCD with Arduiuno Mega:

When connecting an LCD display (like a 16x2 LCD) directly to an Arduino, you normally need at least 6 to 10 digital pins (for RS, E, D4-D7, plus maybe RW and backlight control).

Problem:

- It uses too many pins.
- Wiring becomes messy and hard to debug.

Benefits of using I2C with LCD:

1).Saves Arduino pins:

- Only 2 pins needed (SDA and SCL).

2).Simpler wiring:

- Cleaner, fewer jumper cables.

3).Allows multiple devices:

- You can connect multiple I2C devices (sensors, displays) on the same two wirs (addressed by different I2C addresses).

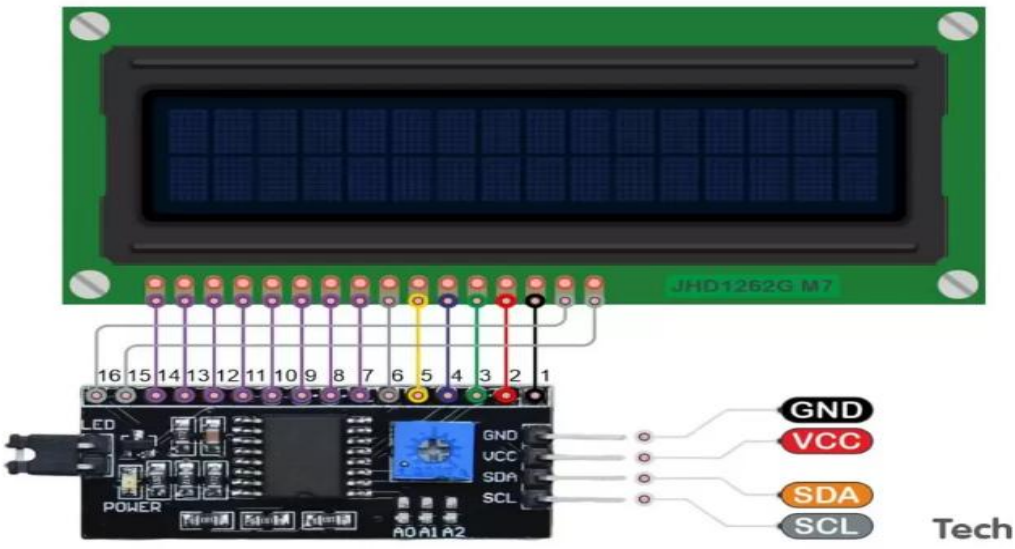
4).Easier programming:

- Libraries like **LiquidCrystal_I2C.h** make coding simpler.

5).More reliable:

- Fewer chances of wrong connections or noisy signals compared to parallel wiring.

8).I2C With LCD:



GND is a ground pin and should be connected to the ground of Arduino.

VCC supplies power to the module and the LCD. Connect it to the 5V output of the Arduino or a separate power supply.

SDA is a Serial Data pin. This line is used for both transmit and receive. Connect to the SDA pin on the Arduino.

SCL is a Serial Clock pin. This is a timing signal supplied by the Bus Master device. Connect to the SCL pin on the Arduino.

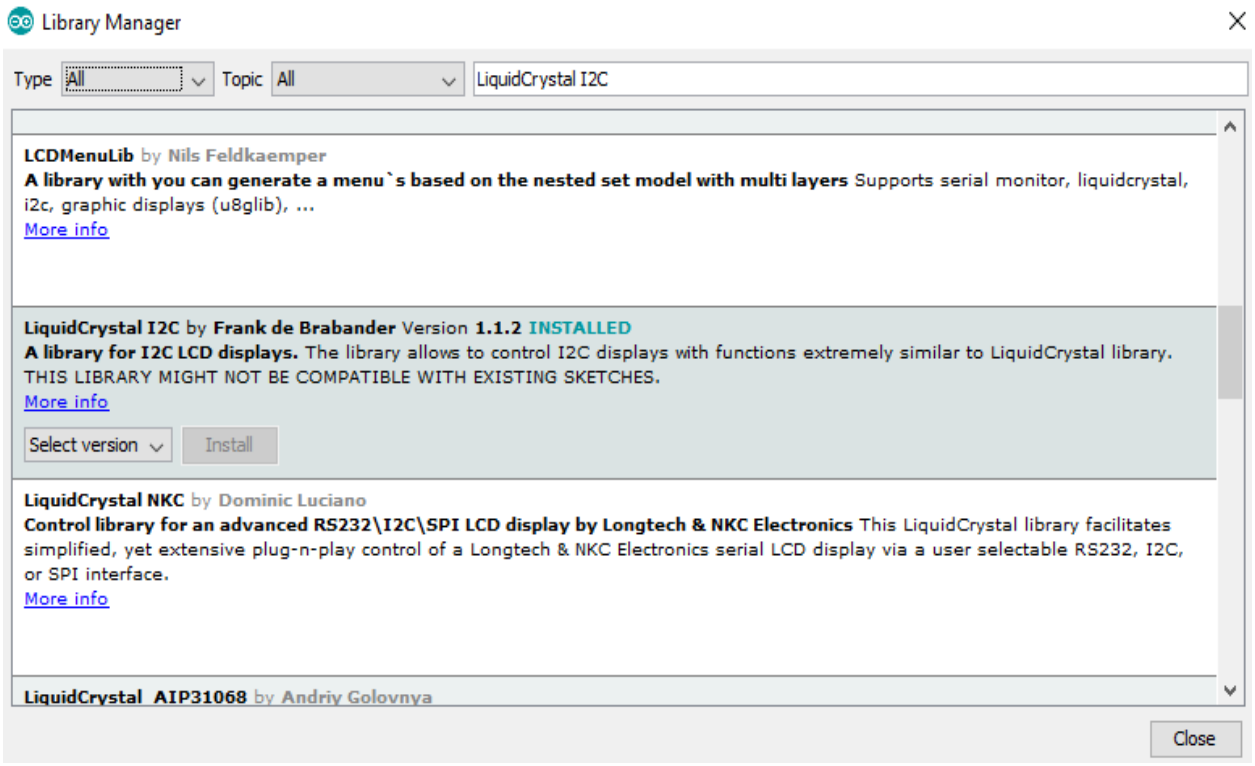
9).Interfacing I2C LCD With Arduino:

I2C LCD can be connected to the Arduino directly with SDA pin to SDA pin and SCL pin to SCL pin as per the below circuit diagram. I2C LCD requires additional library to be installed. The next step is to connect the LCD to the address of the device using the following code. Those steps are explained in detail below.

Steps to Interface LCD display with Arduino:

Step 1: Install the library for LCD display in Arduino IDE.

- Open Arduino IDE and navigate to **Tools>Library Manager.**
- Search for **"LiquidCrystal I2C"** and install the **"LiquidCrystal I2C"** library in the Arduino IDE.



Step 2: Import **“LiquidCrystal_I2C.h”** header file in the code.

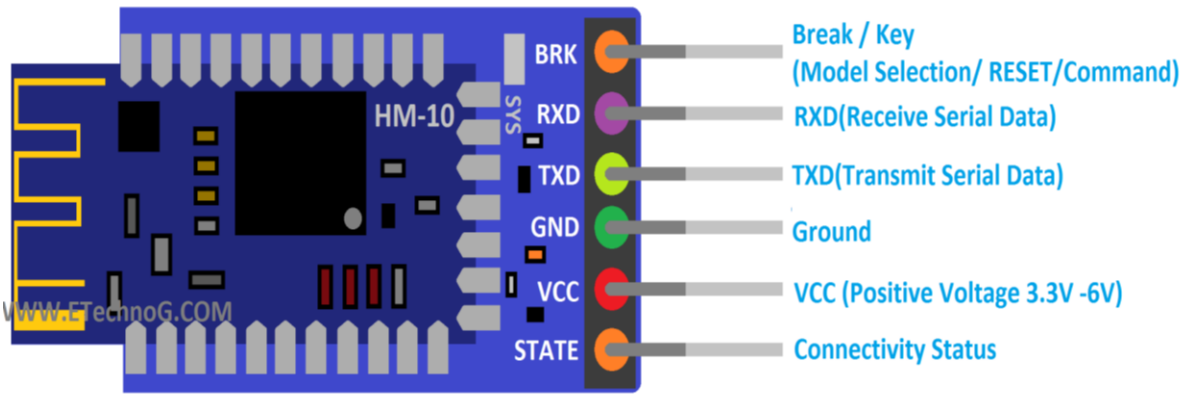
- Define header file in the code **” #include <LiquidCrystal_I2C.h> “.**

Step 3: Connect display device to Arduino.

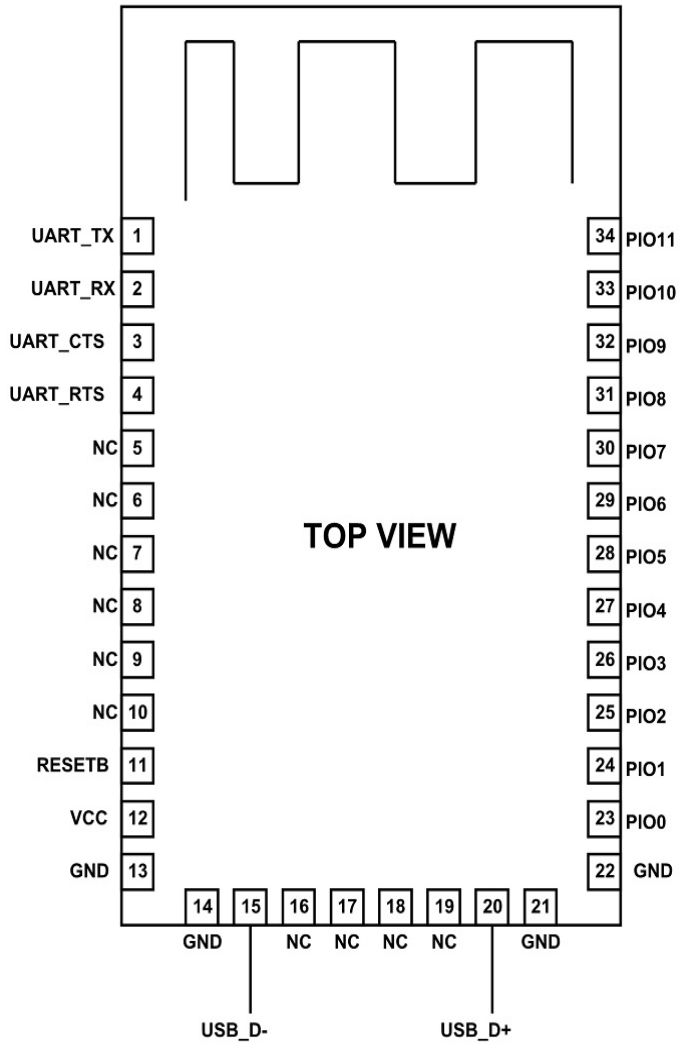
- Connect the **SDA** pin of an LCD display to the **SDA** pin of the Arduino.
- Connect the **SCL** pin of an LCD display to the **SCL** of the Arduino.
- Connect **VCC** to **5V** pin
- Connect **GND** to **GND** pin.

10).Bluetooth Module(HM-10):

The **HM-10 Bluetooth module** is a low-power Bluetooth 4.0 (BLE) module based on the **TI CC2541** chip. It's widely used in wireless communication projects and supports communication with both iOS and Android devices. The HM-10 operates over UART and can be easily integrated with microcontrollers like Arduino. It's popular due to its **low power consumption**, **long range**, and **compatibility with BLE devices**.



Pinout of HM-10 Module:



1). **UART_TX** – UART transmit (serial data output).

2). **UART_RX** – UART receive (serial data input).

3). **UART_CTS** – UART Clear To Send (flow control, optional).

4). **UART_RTS** – UART Request To Send (flow control, optional).

5). **NC** – Not connected.

6). **NC** – Not connected.

7). **NC** – Not connected.

8). **NC** – Not connected.

9). **NC** – Not connected.

10). **NC** – Not connected.

11). **RESETB** – Reset input, active LOW. Pulling LOW resets the module.

12). **VCC** – Power supply input (3.3V).

13). **GND** – Ground.

14). **GND** – Ground.

15).**USB_D-** – USB Data Minus (for USB communication if supported; normally unused).

16). **NC** – Not connected.

17). **NC** – Not connected.

18). **NC** – Not connected.

19). **NC** – Not connected.

20). **NC** – Not connected.

21). **GND** – Ground.

22). **GND** – Ground.

23). **PIO0** – Programmable Input/Output pin 0.

24). **PIO1** – Programmable Input/Output pin 1.

25). ***PIO2*** – *Programmable Input/Output pin 2.*

26). **PIO3** – Programmable Input/Output pin 3.

27). **PIO4** – Programmable Input/Output pin 4.

28). **PIO5** – Programmable Input/Output pin 5.

29). **PIO6** – Programmable Input/Output pin 6.

30). **PIO7** – Programmable Input/Output pin 7.

31). **PIO8** – Programmable Input/Output pin 8.

32). **PIO9** – Programmable Input/Output pin 9.

33). **PIO10** – Programmable Input/Output pin 10.

34). **PIO11** – Programmable Input/Output pin 11.

Working Principle of a Bluetooth Module:

The Bluetooth module usually works by:

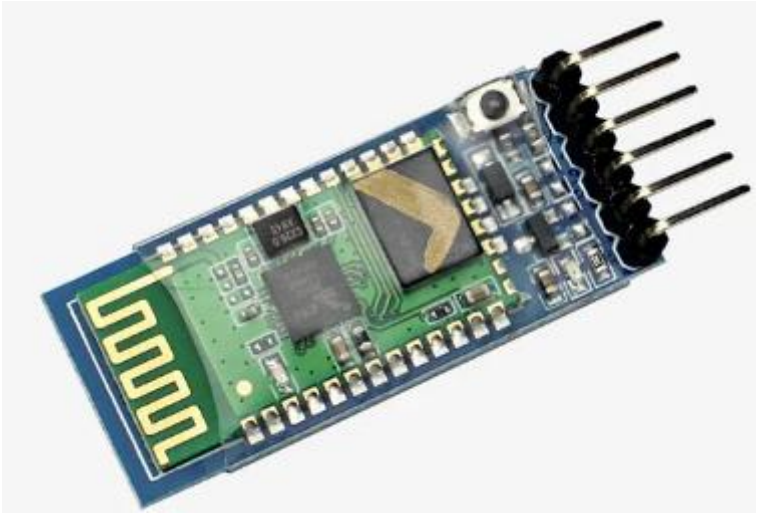
1. **Initialization:**
    - When powered, the module becomes visible (discoverable) to nearby Bluetooth devices.
2. **Pairing:**
    - A secure connection is established with another device using a PIN/password.
3. **Data Communication:**
    - Most Bluetooth modules use **UART (Serial Communication)**.
    - The microcontroller communicates with the module using **TX** (transmit) and **RX** (receive) pins.
    - The Bluetooth module wirelessly transmits this serial data to another paired device (phone, computer, etc.).
4. **Modes:**
    - **Master Mode:** Initiates connection.
    - **Slave Mode:** Waits for a connection.

Types of Bluetooth Modules:

1).HC-05 Bluetooth Module (Classic Bluetooth):

The HC-05 is a Bluetooth-to-Serial-Bridge module that allows wireless communications between two microcontrollers or between a microcontroller

and a smartphone, laptop, or desktop PC with Bluetooth capability. It's perfect for directly replacing a wired asynchronous serial interface!



Key Features:

- Supports both **Master** and **Slave** modes.
- Works on **Bluetooth V2.0 + EDR**.
- Easy to switch between modes using **AT commands**.
- Default PIN: **1234** or **0000**.
- Range: **~10 meters**.

Use Case:

- Two microcontrollers talking to each other wirelessly.
- Mobile app control projects.

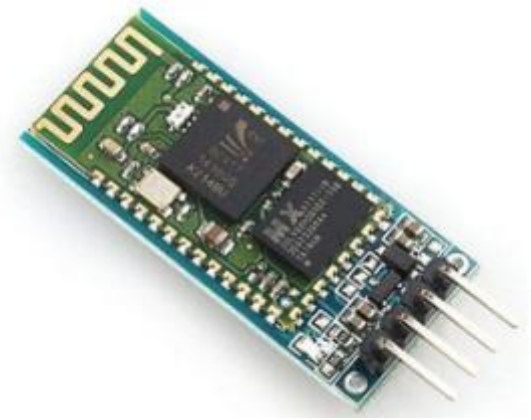
Specifications:

- Operating Voltage: **3.3V** (but often has onboard 5V tolerance)
- Communication: UART (serial)
- Baud Rate: 9600 bps (default, can change)
- Frequency: 2.4 GHz ISM Band
- Current: 30-50 mA during transmission

2). HC-06 Bluetooth Module (Classic Bluetooth):

HM-06 is a **Bluetooth module** designed for establishing short range wireless data communication between two microcontrollers or systems. The module works on **Bluetooth 2.0 communication protocol** and it can only act as a slave device. This is cheapest method for wireless data transmission and more flexible compared to other methods and it even can transmit files at speed up to 2.1Mb/s.

HC-06 uses frequency hopping spread spectrum technique (**FHSS**) to avoid interference with other devices and to have full duplex transmission. The device works on the frequency range from 2.402 GHz to 2.480GHz.



Key Features:

- Supports **only Slave** mode.
- Slightly cheaper and simpler than HC-05.
- No Master-Slave switch; always ready to be connected.
- Default PIN: **1234** or **0000**.

Use Case:

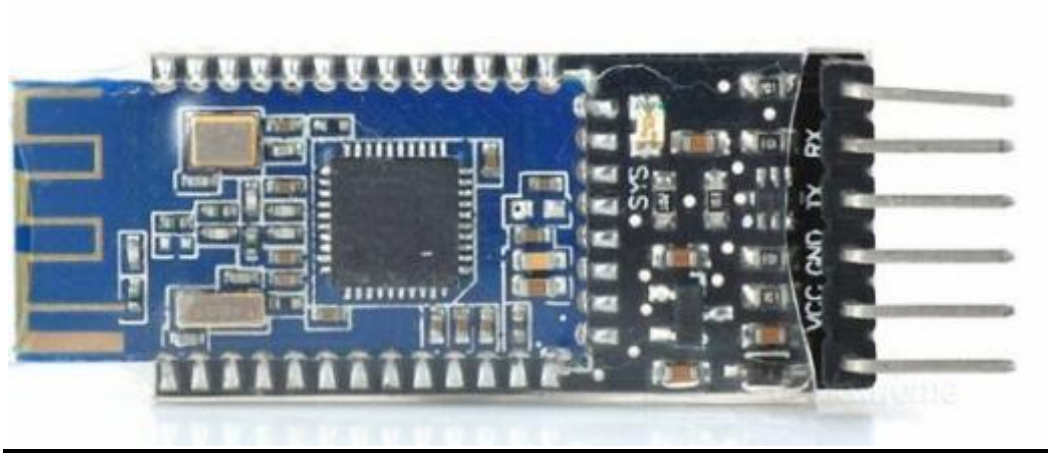
- Ideal for simple mobile-to-device communication.
- Arduino controlling small home automation setups.

Specifications:

- Voltage: **3.3V – 5V**
- Serial Interface: UART
- Range: 10 meters
- Frequency: 2.4 GHz ISM Band

3). HM-10 Bluetooth Module (BLE – Bluetooth Low Energy):

The **HM-10 Bluetooth module** is a low-power Bluetooth 4.0 (BLE) module based on the **TI CC2541** chip. It's widely used in wireless communication projects and supports communication with both iOS and Android devices. The HM-10 operates over UART and can be easily integrated with microcontrollers like Arduino. It's popular due to its **low power consumption**, **long range**, and **compatibility with BLE devices**.



Key Features:

- Supports **Bluetooth 4.0 / BLE** standard.
- Can act as **Master** or **Slave**.
- Very **low power consumption** (ideal for battery-powered devices).
- Used for communicating with modern smartphones (Android, iPhone).

Use Case:

- IoT applications (low energy, sensors).
- Wearables (fitness trackers, smartwatches).

Specifications:

- Voltage: **3.3V – 6V**
- Communication: UART (serial)
- Range: 50 meters (depending on version)
- Frequency: 2.4 GHz ISM Band
- Current: Ultra-low (few mA in operation, microamps in sleep)

4). HC-08 Bluetooth Module (BLE – Bluetooth Low Energy):

The HC-08 module allows to transmit and receive data through the serial port of the micro controller wirelessly. It belongs version 4.0 of Bluetooth devices, which has the same characteristics as its predecessors, but with the difference that this device is low energy consumption. It also adds the feature of being compatible with Apple brand products.

Key Features:

- Works only in **Slave mode**.
- **Simpler BLE** compared to HM-10.
- Compatible with iPhone and Android devices.

Use Case:

- Simple BLE communication with mobile apps.
- Sensor data sending (e.g., temperature, humidity).

Specifications:

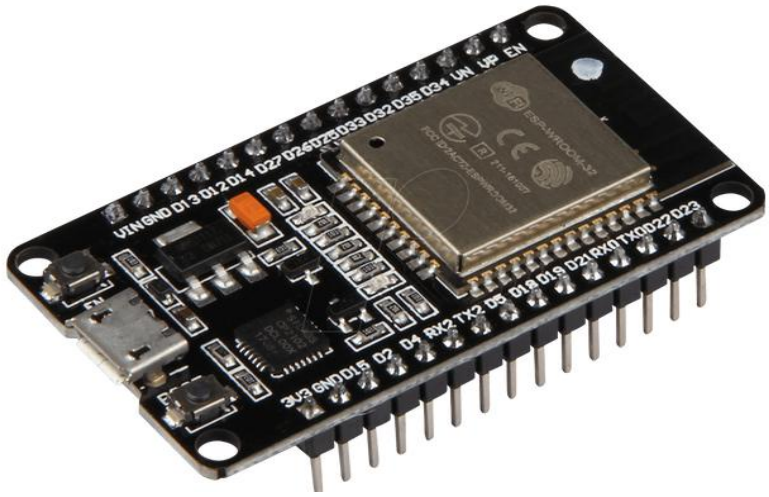
- Voltage: **3.3V – 5V**
- Communication: UART
- Range: 80 meters (open space)
- Current: Very low power

5). ESP32 Bluetooth (Integrated Classic + BLE):

ESP32 has on-chip Bluetooth and BLE (Bluetooth Low Energy). In this guide, we will see the Bluetooth part.

ESP32 Bluetooth is also referred as classic Bluetooth.

Using Bluetooth is very much simple on ESP32 with **BluetoothSerial** Library with Arduino IDE.



Key Features:

- ESP32 is a **microcontroller with WiFi + Bluetooth** both built-in.
- Supports both **Classic Bluetooth** and **BLE**.

- High processing power: dual-core processor.
- Can run **Arduino programs** directly.

Use Case:

- Smart IoT devices (sending data over WiFi and Bluetooth).
- Advanced projects (Voice control, Real-time monitoring).

Specifications:

- Voltage: **3.3V**
- Range: 10-100 meters
- Connectivity: WiFi 802.11 b/g/n, Bluetooth v4.2
- Powerful: Up to 520 KB RAM, integrated sensors.

General Specifications of Bluetooth Modules

Feature	Range
Operating Voltage	3.3V – 5V
Communication	UART (Serial TX, RX)
Frequency	2.4 GHz ISM Band
Data Rate	9600 bps (default), up to 115200 bps
Range	10 meters (typical)
Power Consumption	Low (especially BLE modules)

Applications of Bluetooth Modules

Home Automation

- Controlling lights, fans, locks using smartphone apps.

Wireless Data Logging

- Sending temperature, humidity, and other sensor data to a mobile app.

Wireless Robotics

- Controlling robots, drones, and cars wirelessly.

Health Monitoring

- Wearable health devices like fitness bands (BLE modules).

Industrial Automation

- Machine-to-machine wireless communication in factories.

IoT (Internet of Things)

- Bluetooth-enabled smart devices like alarms, weather stations, etc.

Smart Home

- Appliances like smart thermostats, doorbells, and speakers.

Gaming Devices

- Wireless controllers, VR headsets.

Why we use HM-10 Bluetooth Module Over HC-05 Module:

1. Power Consumption

- The HM-10 BLE module **consumes much less power** than the HC-05.
- BLE is specially designed for devices that operate on small batteries for **months or even years**.

- HC-05, using Classic Bluetooth, consumes **more current** because it maintains an active connection constantly (which drains the battery faster).

2. Compatibility with Smartphones (especially iPhones)

- BLE (used by HM-10) is **natively supported** by both **Android and iOS (iPhone)**.
- Classic Bluetooth (used by HC-05) works well with **Android phones** but **not easily with iPhones** because Apple restricts Classic Bluetooth in many apps.
- If you want to connect your Arduino to an iPhone app, **you must use BLE** (HM-10 or similar).

3. Data Transmission Method

- **HC-05** is designed for **continuous streaming** of large data (like audio, music).
- **HM-10** is designed for **short, quick bursts** of small data packets (like sending a temperature reading or a button press).
- BLE reduces energy usage by sending small data, then immediately disconnecting.

4. Range and Signal Strength (RSSI)

- RSSI stands for **Received Signal Strength Indicator**.
- It tells **how strong the Bluetooth signal is** between devices.
- Both HC-05 and HM-10 provide an **RSSI value**, but:
    - In HM-10 (BLE), RSSI is commonly used to **estimate distance** or **signal quality** because BLE applications often involve dynamic devices (moving wearables, trackers).
- **Higher RSSI** (closer to 0 dBm) = **Stronger signal**.

- **Lower RSSI** (more negative) = **Weaker signal**.

Typical RSSI examples:

- -40 dBm → Very strong signal (close)
- -70 dBm → Medium signal (some distance)
- -90 dBm → Weak signal (far away)

BLE devices like HM-10 often use RSSI for **proximity detection** (example: smart keys, asset tracking).

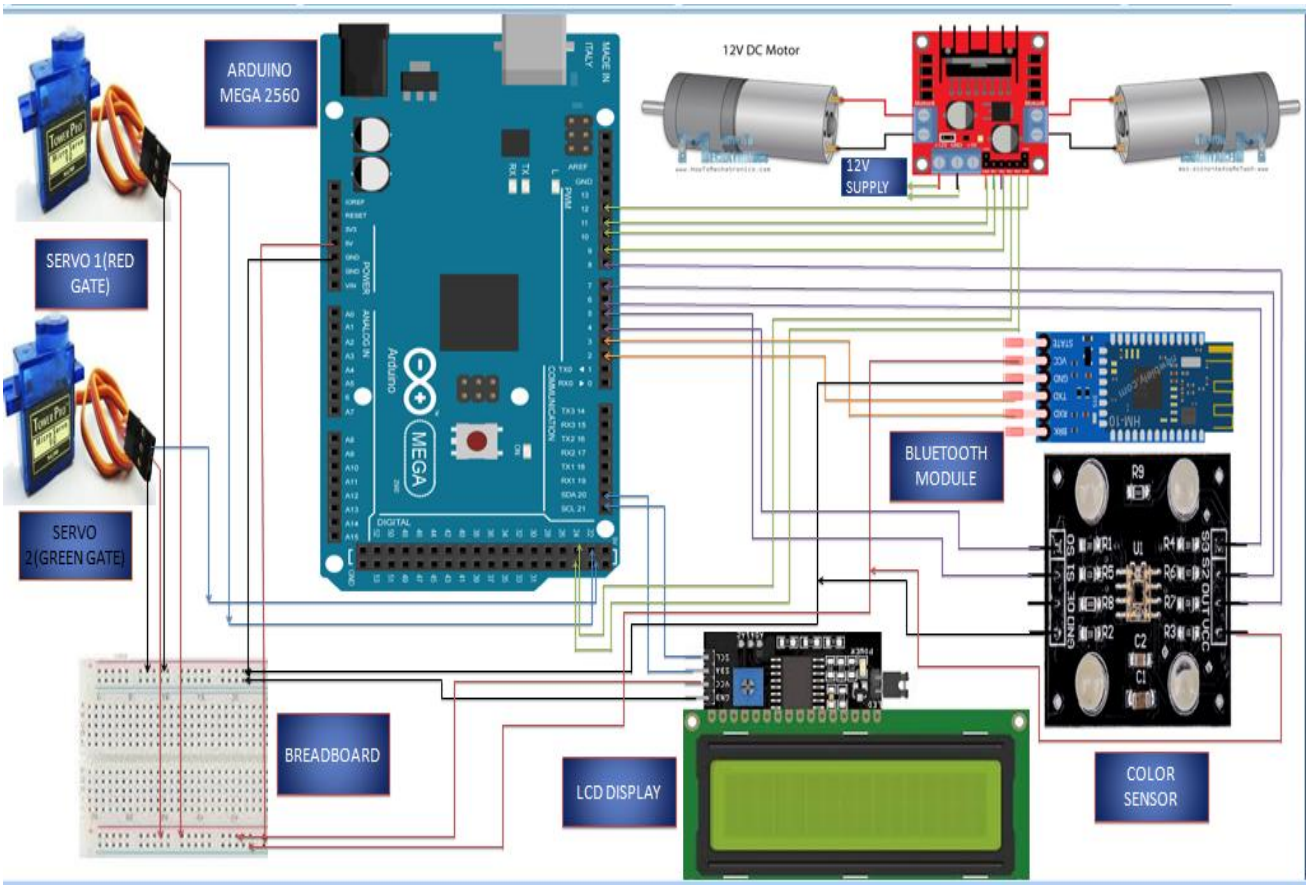
5. Speed and Data Rate

- **HC-05** can transmit data faster because it supports continuous streaming.
- **HM-10** has a lower transmission rate because BLE is optimized for small, quick messages, not streaming.

But for most Arduino projects like:

- Sending sensor values
- Turning motors on/off
- Controlling LEDs
  You don't need high data rates — so HM-10 is enough and better for power-saving.

Circuit Design



Circuit Explaination:

This is a **project based on Arduino Mega 2560**, combining multiple devices:

- **Two Servo Motors** (to open/close gates — Red and Green)
- **DC Motors** (controlled through Motor Driver Module)
- **Bluetooth Module** (for wireless control)
- **Color Sensor** (to detect object color)
- **LCD Display with I2C Adapter** (to show messages)
- **Breadboard** (for power and ground distribution)

1. **Arduino Mega 2560**

- **Main microcontroller board** that connects and controls all modules.

- It has **more input/output pins** compared to Arduino Uno, suitable for this big project.

2. Servo Motors (Red Gate and Green Gate)

- **Servo 1** and **Servo 2** are small TowerPro SG90 servos.
- Connected to **digital pins** of the Arduino (through Breadboard for power sharing).
- They are probably used to **open and close gates** based on some conditions (like detected color or Bluetooth command).

3. DC Motors and L298N Motor Driver Module

- Two **12V DC motors** are controlled by an **L298N Dual Motor Driver**.
- The motor driver acts as a **bridge** — it receives control signals from Arduino and switches higher power (12V) to the DC motors.
- Motors could be for moving a platform, conveyor, or robot wheels.

4. Bluetooth Module (HM-10)

- Connected to **TX/RX (serial communication)** of Arduino.
- Allows **wireless communication** with a smartphone or computer.
- You can send commands through an app to:
    - Open/close gates (servo control)
    - Start/stop motors
    - Display something on the LCD
- It’s labeled "Bluetooth Module" in the diagram.

5. Color Sensor (TCS3200)

- This sensor detects the **color** of objects.
- It sends color information to Arduino.
- Arduino could then decide actions:
    - For example, if color = RED, open Red Gate (Servo 1).

    - If color = GREEN, open Green Gate (Servo 2).
- TCS3200 color sensors give **frequency output** proportional to detected color.

6. LCD Display with I2C Adapter

- A **16x2 LCD** is used for showing messages or statuses.
- It uses an **I2C adapter** (backpack module) — which **reduces wiring** from 12 pins to just 4 pins (VCC, GND, SDA, SCL).
- Connected to Arduino's **SDA/SCL pins** (A4, A5 on Uno — different on Mega).

7. Breadboard

- Breadboard is used for **power and ground distribution**.
- 5V and GND from Arduino are distributed to Servo motors and other components through this.

ARDIUNO CODE:

```
#include <SoftwareSerial.h>

SoftwareSerial BT05(2, 3); // TX = 3, RX = 2 (opposite of Receiver)

#include <Wire.h>

#include <LiquidCrystal_I2C.h>

#include <Servo.h>

// LCD Setup (I2C Address 0x27, 16x2 Display)

LiquidCrystal_I2C lcd(0x27, 16, 2);

// Color Sensor Pins

#define S0 4

#define S1 5
```

```
#define S2 6

#define S3 7

#define SENSOR_OUT 8

// L298N Motor Driver Pins (Motor A - Conveyor)

#define IN1 9

#define IN2 10

#define ENA 11  // PWM for speed control

// L298N Motor Driver Pins (Motor B - Extra DC Motor after Green)

#define IN3 24

#define IN4 25

#define ENB 12  // PWM

// Servo Motor Pins

#define SERVO_RED_PIN 22

#define SERVO_GREEN_PIN 23

#define relay_pin 26

// Servo Objects

Servo servoRed;

Servo servoGreen;

// Counters for Boxes

int redCount = 0;

int greenCount = 0;

int blueCount = 0;
```

```
String previousColor = "None";  // Previous color tracking
void setup() {
    Serial.begin(9600);
    BT05.begin(9600); // Bluetooth module
    Serial.println("Sender Ready");
    // Initialize LCD
    lcd.init();
    lcd.backlight();
    // Initialize Servo Motors
    servoRed.attach(SERVO_RED_PIN);
    servoGreen.attach(SERVO_GREEN_PIN);
    servoRed.write(0);
    servoGreen.write(0);
    pinMode(26, OUTPUT);
    digitalWrite(26, HIGH);
    // Set Color Sensor Pins
    pinMode(S0, OUTPUT);
    pinMode(S1, OUTPUT);
    pinMode(S2, OUTPUT);
    pinMode(S3, OUTPUT);
    pinMode(SENSOR_OUT, INPUT);
    // Motor A Pins
```

```
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(ENA, OUTPUT);
    // Motor B Pins
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);
    pinMode(ENB, OUTPUT);
    // Set Frequency Scaling to 20%
    digitalWrite(S0, HIGH);
    digitalWrite(S1, LOW);
    Serial.println("System Ready...");
    lcd.setCursor(0, 0);
    lcd.print("System Ready...");
}
void conveyorStart() {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(ENA, 200);
}
void conveyorStop() {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
```

```
    analogWrite(ENA, 0);
}
void motorBStart() {
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    analogWrite(ENB, 200);
}
void motorBStop() {
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    analogWrite(ENB, 0);
}
String detectColor() {
    int redFrequency, greenFrequency, blueFrequency;
    delay(300);
    digitalWrite(S2, LOW); digitalWrite(S3, LOW);
    redFrequency = pulseIn(SENSOR_OUT, LOW);
    delay(50);
    digitalWrite(S2, HIGH); digitalWrite(S3, HIGH);
    greenFrequency = pulseIn(SENSOR_OUT, LOW);
    delay(50);
```

```
    digitalWrite(S2, LOW); digitalWrite(S3, HIGH);
    blueFrequency = pulseIn(SENSOR_OUT, LOW);
    delay(50);
    Serial.print("Red: "); Serial.print(redFrequency);
    Serial.print(" Green: "); Serial.print(greenFrequency);
    Serial.print(" Blue: "); Serial.println(blueFrequency);
    if (redFrequency < 100 && greenFrequency < 100 && blueFrequency <
100) {
        Serial.println("No Object Detected");
        previousColor = "None";
        return "None";
    }

    if (redFrequency < greenFrequency && redFrequency < blueFrequency) {
        return "Red";
    } else if (greenFrequency < redFrequency && greenFrequency <
blueFrequency) {
        return "Green";
    } else {
        return "Blue";
    }
}
```

```
void openGate(Servo &servo) {
    servo.write(90);       // Open gate
    delay(3000);           // Shorter time for box to pass
    servo.write(0);        // Close gate
}
void loop() {
    String color = detectColor();
    if (color == "None") {
        conveyorStop();
        previousColor = "None";
        return;
    }
    if (color == previousColor) {
        conveyorStart();
        if (color == "Red") {
            Serial.println("Red Box Confirmed");
            openGate(servoRed);
            redCount++;
        }
        else if (color == "Green") {
            Serial.println("Green Box Confirmed");
            openGate(servoGreen);
```

```
            greenCount++;
            motorBStart();
            Serial.println("Extra Motor Started");
            delay(5000);  // Run for 5 seconds
            motorBStop();
            Serial.println("Extra Motor Stopped");
        }
        else {
            Serial.println("Blue Box Confirmed");
            blueCount++;
        }

        previousColor = "None";
    }
    else {
        previousColor = color;
        Serial.print("Detected (waiting): "); Serial.println(color);
        delay(100);
        return;
    }

    // LCD Update
```

```
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Red:"); lcd.print(redCount);
lcd.setCursor(8, 0);
lcd.print("Green:"); lcd.print(greenCount);
lcd.setCursor(0, 1);
lcd.print("Blue:"); lcd.print(blueCount);
// Show current detected color short form
lcd.setCursor(13, 1);
lcd.print("Clr:");
lcd.print(color[0]);  // Show first letter: R, G, B
if (redCount >= 1 && greenCount >= 1 ) {
digitalWrite(26, LOW);
BT05.println("Hello from sender");
Serial.println("Sent message to receiver");
delay(7000);
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Reset Done");
   Serial.println("Auto Reset Counts!");
   redCount = 0;
   greenCount = 0;
```

```
        blueCount = 0;
        delay(2000);
        lcd.clear();
    }
    delay(2000);
}
```

TESTING AND TROUBLESHOOTING OF CIRCUIT

1. Power Supply Check:

- **Check Power to Arduino MEGA:** Ensure the Arduino is powered either via USB or external 9V/12V source.
- **Check Power to Components:** Use a multimeter to confirm that:
- The **breadboard power rails** are properly powered (typically 5V and GND).
- The **motor driver** is getting 12V supply from its input.
- **Servo motors** are connected to 5V and GND properly.

2. Component Connectivity Check:

A).Wiring Inspection:

- Visually confirm each connection to pins as per the diagram.
- Verify **servo wires** (signal, VCC, GND) are not reversed.
- Ensure proper orientation of modules like **Bluetooth** and **color sensor**.

3. Test Individual Components:

A).Servo Motors:

- Use a basic servo sweep test sketch to see if each one moves correctly.

- If not moving, check signal wire continuity and power level.

B).DC Motors & Motor Driver:

- Upload a basic sketch to rotate motors forward and backward.
- Check that the **motor driver inputs** (IN1, IN2, etc.) are correctly linked to the Arduino pins.

C).Bluetooth Module:

- Connect to it via a mobile app (like Arduino Bluetooth Controller).
- Send simple commands and check for proper TX-RX communication.

D).Color Sensor (TCS3200 or similar):

- Run a test sketch that prints out RGB values or frequency.
- Confirm that the S0–S4 and OUT pins are connected correctly.

E).LCD Display:

- Run the Hello World sketch from the LiquidCrystal library.
- Adjust the **potentiometer** to set contrast if the display is blank.

4. Software Debugging:

- Use the **Serial Monitor**:
- Add Serial.println() statements in key parts of your code.
- Monitor sensor values, control flow, and component states.

A).Check Libraries:

- Make sure all required libraries (e.g., Servo, LiquidCrystal, etc.) are installed.

REFERENCES

1).Electronics Circuit

2).Google Websites:

- https://forum.arduino.cc/t/controlling-a-12v-motor-2-servo-motors-and-2-ir-sensors-with-an-arduino/548239
- https://forum.arduino.cc/t/conveyor-belt-arduino/191891
- https://forum.arduino.cc/t/please-help-with-my-conveyor-belt-color-sensor-project-tcs-3200/587984

3).Youtube

- https://youtu.be/8qOU7Uejr74?si=0kffMR4xImtNngmM
- https://youtu.be/9_DpSb6gBFY?si=kxWfH0LYxm5LYFJ7

4).Chatgpt.

5). quartz components

ESTIMATION

S.No	Componnent Name	Quantity	Price
1.	12V Battery	2(6v Each)	800
2.	Arduino Mega 2560	1	1860
3.	12V DC gear motor	2	466
4.	Color Sensor	1	500
5.	L298N Motor Driver Module	1	285
6.	Motor Speed Controller	1	403
7.	Servo Motors	2	314
8.	Glue Gun sticks	2	60
9.	Screws	40	40
10.	Fevibond	1	50
11.	Jumper Wire	60	200
12.	I2C Module	1	177
13.	LCD Display	1	College inventery
14.	Bluetooth Module(HM-10)	4	964
15.	Tape	2	120

Total=6240