

A

Summer Training Report

Submitted

In partial fulfillment

For the award of the degree

Bachelor of Technology

**In the Department of Electronics & Communication
Engineering**

(with specialization in Computer Science & Engineering)



Submitted To

Mr. Gaurav Singh Rathore

Submitted By

Joydip Dutta

17EREEEC006

B.Tech 4th year

Department of Electronics and Communication Engineering

Rajasthan Institute of Engineering and Technology, Jaipur

Rajasthan Technical University, Kota

12 January - 2021

Candidate's Declaration

I hereby declare that the courses, which are being presented in the report, in partial fulfillment of the award of Degree of “Bachelor of Technology” in Dept. of Electronics And Communication Engineering with specialization in Computer Science Engineering, and submitted to the Department of Computer Science and Engineering, Rajasthan Institute of Engineering and Technology, Rajasthan Technical University is a record of my investigations carried under the guidance of, Department of Computer Science and Engineering, Rajasthan Institute of Engineering and Technology.

I have not submitted the matter presented in this Dissertation anywhere for the award of any other Degree.

(Joydip Dutta)

Electronics and Communication Engineering

Enrolment No.: 17EREEC006

Rajasthan Institute of Engineering and Technology Jaipur Rajasthan

ACKNOWLEDGEMENT

I take this opportunity to express my gratitude to all those people who have been directly or indirectly helped me to complete this course.

I pay thanks to my mentor of this training Dr. Peter Dalmaris who has given guidance and a light to me by providing this exciting course on Udemy. His versatile knowledge about “Arduino And Electronics” has eased me in the projects which I have made during the span of the Course.

I acknowledge here about debt to those who contributed significantly to one more step. I take full responsibility for any remaining sins of omission and commission.

Joydip Dutta

B.Tech IV year

(Electronics and Communication Engineering)

ABSTRACT

This is a document for an introductory course in Arduino. This document is not a comprehensive introduction or a reference manual. Instead, it focuses on the specific features of ARDUINO that are useful for embedded engineers. Other than a few essential tools and parts that you can learn about in the first section of the course (which is free to watch) I only ask that you come with an appetite to learn and a willingness to work hard. The course features a large variety of parts, but you don't need (and you shouldn't) get them all to begin learning. In fact, getting all of these parts too early is not a good idea! Watch the free lecture to understand why in the first section of this course.

Apart from a basic understanding of the Arduino, a student of this course must be willing to work hard. You can't learn serious skills without serious work. There are a lot of courses out there that promise you a fun and easy learning experience. This is not one of them. I promise you a lot of hard work ahead. The emphasis here is “learning by doing”.

CONTENTS

SECTION 1: INTRODUCTION TO THE ARDUINO	1
1.1-Overview of the Arduino Uno Board	1
1.2-Installation of Arduino IDE	4
Step 1 – Selection of the development board.	4
Step 2 – Download Arduino IDE Software.	5
Step 3 – Power up your board.	5
Step 4 – Launch Arduino IDE.	5
Step 5 – Open your first project.	5
Step 6 – Select your Arduino board.	7
Step 7 – Select your serial port.	7
Step 8 – Upload the program to your board.	8
1.3-Program Structure of Arduino	9
Void setup () {}	10
Void Loop () {}	10
SECTION 2: SENSORS, ACTUATORS, AND DISPLAYS	11
2.1-Sensors	11
2.2-Actuators	11
2.3-Displays	11
2.4-List of the Components (Sensors, Actuators, And Displays)	12
SECTION 3: INTERFACING OF SENSORS WITH ARDUINO UNO	13
3.1-Interfacing of Temperature and Humidity Sensor (DHT22)	13
3.1.1-Technical Specifications:	13
3.1.2-How to Use DHT22 Sensor module:	14
3.1.3-Circuit Diagram (Schematics):	15
3.1.4-Sketches of Temperature Sensor Modules	15
3.2-Environmental Sensor (BMP180)	15
3.2.1-Technical Specifications:	16

3.2.2 How to Use MP180 Module?	16
3.2.3-Circuit Diagram (Schematics):	17
3.2.4-Sketches of Environment Sensor Module	17
3.3-Motion Sensor and Accelerometer (MPU6050)	18
3.3.1-Technical Specifications:	18
3.3.2-How to Use MPU6050 Module?	19
3.3.3-Circuit Diagram (Schematics):	19
3.3.4-Sketches of MPU6050 Module	20
3.4-Ultrasonic Distance Sensor (HC-SR04)	20
3.4.1-Technical Specifications:	20
3.4.2 How to Use HC-SR04 Module?	21
3.4.3-Circuit Diagram (Schematics):	22
3.4.4-Sketches of UltraSonic Sensors Modules	22
3.5-Passive Infrared Sensor (PIR)	22
3.5.1-Technical Specifications:	23
3.5.2-How to Use PIR Module?	23
3.5.3-Circuit Diagram (Schematics):	24
3.5.4-Sketches of PIR Sensor Modules	24
3.6-Color Sensor Module (TCS320)	25
3.6.1-Technical Specifications	25
3.6.2-How to Use TCS3200 Color Sensor Module?	26
3.6.3-Circuit Diagram (Schematics)	27
3.6.4-Sketches of Color Sensor Modules	27
3.7-Gas Sensor Module (MQ135)	27
Alternative MQ Gas sensors	28
3.7.1-Technical Specifications	29
3.7.2-How to Use MQ135 Gas Sensor Module?	29
3.7.3-Circuit Diagram (Schematics)	31

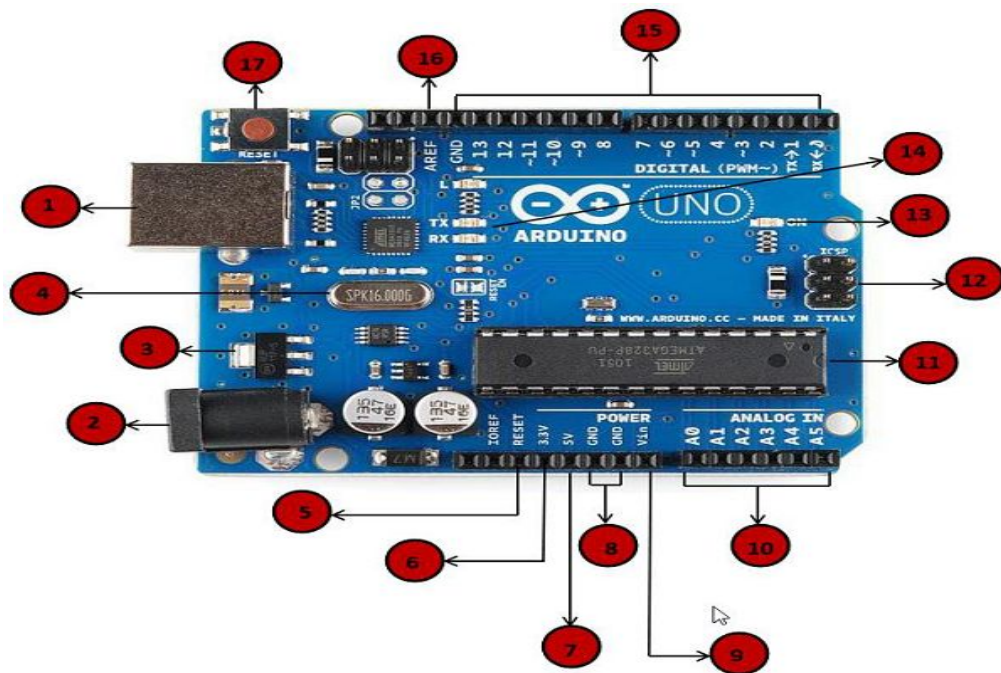
3.7.4-Sketches of Gas Sensor Modules	31
SECTION 4: INTERFACING OF DISPLAYS WITH ARDUINO UNO	32
4.1-Liquid Crystal Display (LCD) Module (Size: 2x16)	32
4.1.1-Pin Description:	32
4.1.2-Circuit Diagram (Schematics)	34
4.1.3-How to display Data On LCD (Custom Characters And Regular Characters)?	34
4.1.4-Sketches of LCD Modules	36
4.2-Seven Segment Displays (Single Digit)	36
4.2.1-Pin Description	36
4.2.2-Circuit Diagram (Schematics)	37
4.2.3-How to Program a Single Digit Seven Segment Display	37
4.2.4-Sketches of Seven Segment Display Modules	38
4.3-An 8x8 LED matrix display	38
4.3.1-Pin Description	38
4.3.2-Circuit Diagram (Schematics)	39
4.3.3-How to Program DotMatrix Display?	39
4.3.4-Sketches of DotMatrixDisplay Modules	40
SECTION 5: INTERFACING OF ACTUATORS WITH ARDUINO UNO	41
5.1- DC Motors And Arduino Uno:	41
5.1.1-Pin Description:	41
5.1.2-Circuit Diagram (Schematics):	42
5.1.3-How to control a pair of DC motors with L283D Driver ?	42
5.1.3-Sketch:	42
5.2- Servo Motors And Arduino Uno:	43
5.2.1-Circuit Diagram (Schematics):	43
5.2.2-How to connect a Servo motor with Arduino Uno ?	43
5.2.2-Sketch:	43
LITERATURE REFERENCES	44

SECTION 1: INTRODUCTION TO THE ARDUINO





Arduino is a prototype platform (open-source) based on easy-to-use hardware and software. It consists of a circuit board, which can be programmed (referred to as a microcontroller), and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer sketch to the physical board. Various kinds of Arduino boards are available depending on the different microcontrollers used. However, all Arduino boards have one thing in common: they are programmed through the Arduino IDE. The differences are based on the number of inputs and outputs (the number of sensors, LEDs, and buttons you can use on a single board), speed, operating voltage, form factor, etc. Some boards are designed to be embedded and have no programming interface (hardware), which you would need to buy separately. Some can run directly from a 3.7V battery, others need at least 5V.

1.1-Overview of the Arduino Uno Board

Arduino UNO board is the most popular board in the Arduino board family. Also, it is the best board to get started with electronics and coding. Some boards look a bit different from the one given below, but most Arduinos have the majority of these components in common.



	<p>Power USB</p> <p>Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).</p>
	<p>Power (Barrel Jack)</p> <p>Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).</p>
3	<p>Voltage Regulator</p> <p>The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.</p>
4	<p>Crystal Oscillator</p> <p>The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.</p>
5,17	<p>Arduino Reset</p> <p>You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labeled RESET (5).</p>

	<p>Pins (3.3, 5, GND, Vin)</p> <ul style="list-style-type: none"> • 3.3V (6) – Supply 3.3 output volt • 5V (7) – Supply 5 output volt • Most of the components used with the Arduino board work fine with 3.3 volts and 5 volts. • GND (8)(Ground) – There are several GND pins on the Arduino, any of which can be used to ground your circuit. • Vin (9) – This pin also can be used to power the Arduino board from an external power source, like an AC mains power supply.
	<p>Analog pins</p> <p>The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.</p>
	<p>Main microcontroller</p> <p>Each Arduino board has its microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the datasheet.</p>
	<p>ICSP pin</p> <p>Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. You are slaving the output device to the master of the SPI bus.</p>

	<p>Power LED indicator</p> <p>This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.</p>
14	<p>TX and RX LEDs</p> <p>On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speeds while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.</p>
15	<p>Digital I/O</p> <p>The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM.</p>
16	<p>AREF</p> <p>AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.</p>

1.2-Installation of Arduino IDE

In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

Step 1 – Selection of the development board.

First, you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or

Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.



Step 2 – Download Arduino IDE Software.

You can get different versions of Arduino IDE from the [Download Page](#) on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux).

Step 3 – Power up your board.

The Arduino Uno, Mega, Duemilanove, and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 – Launch Arduino IDE.

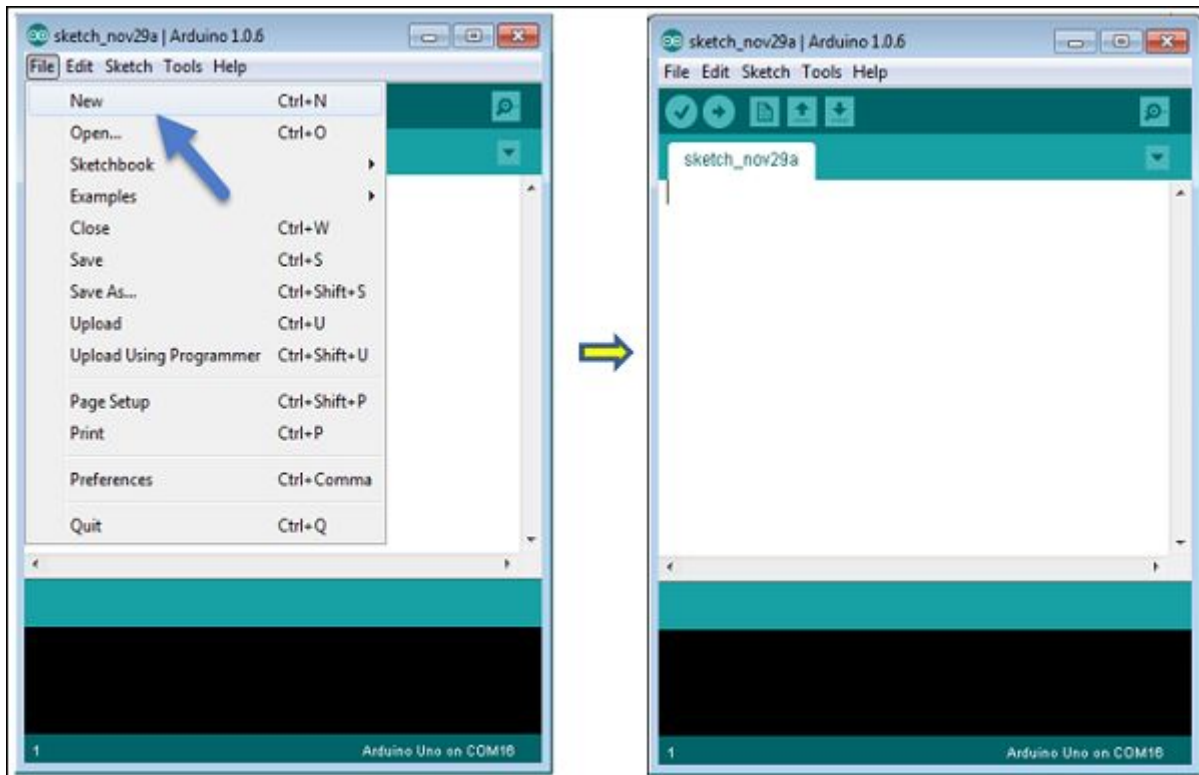
After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

Step 5 – Open your first project.

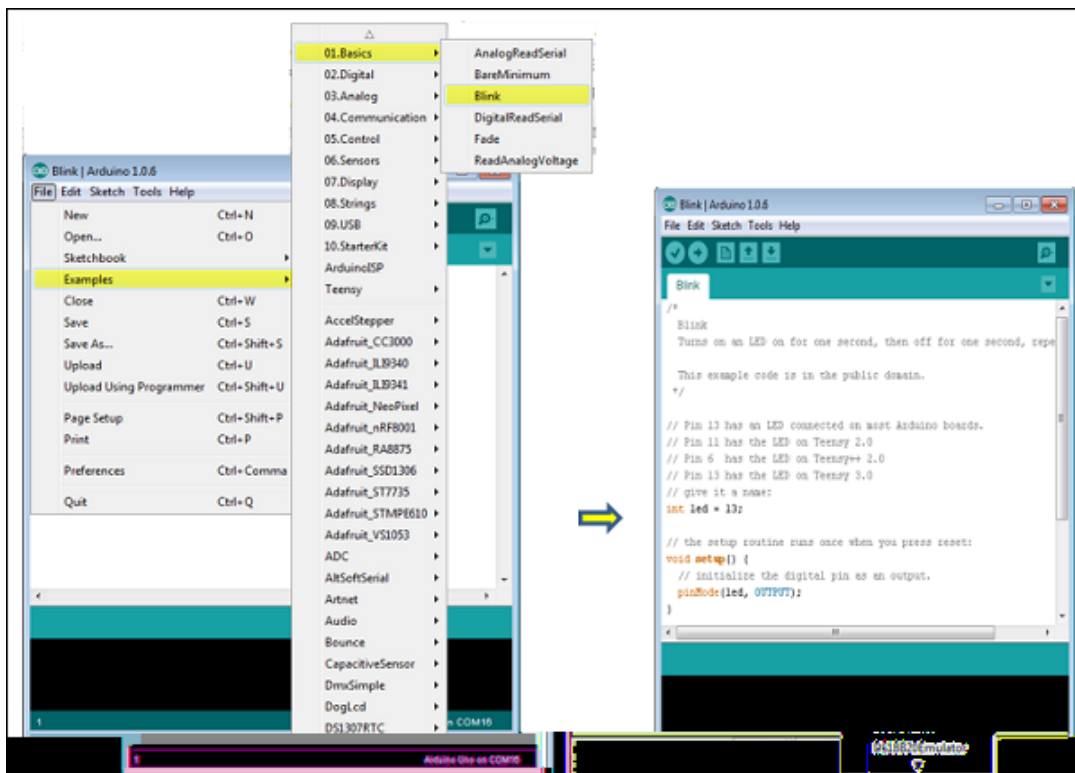
Once the software starts, you have two options –

- Create a new project.
- Open an existing project example.

To create a new project, select File → New.



To open an existing project example, select File → Example → Basics → Blink.

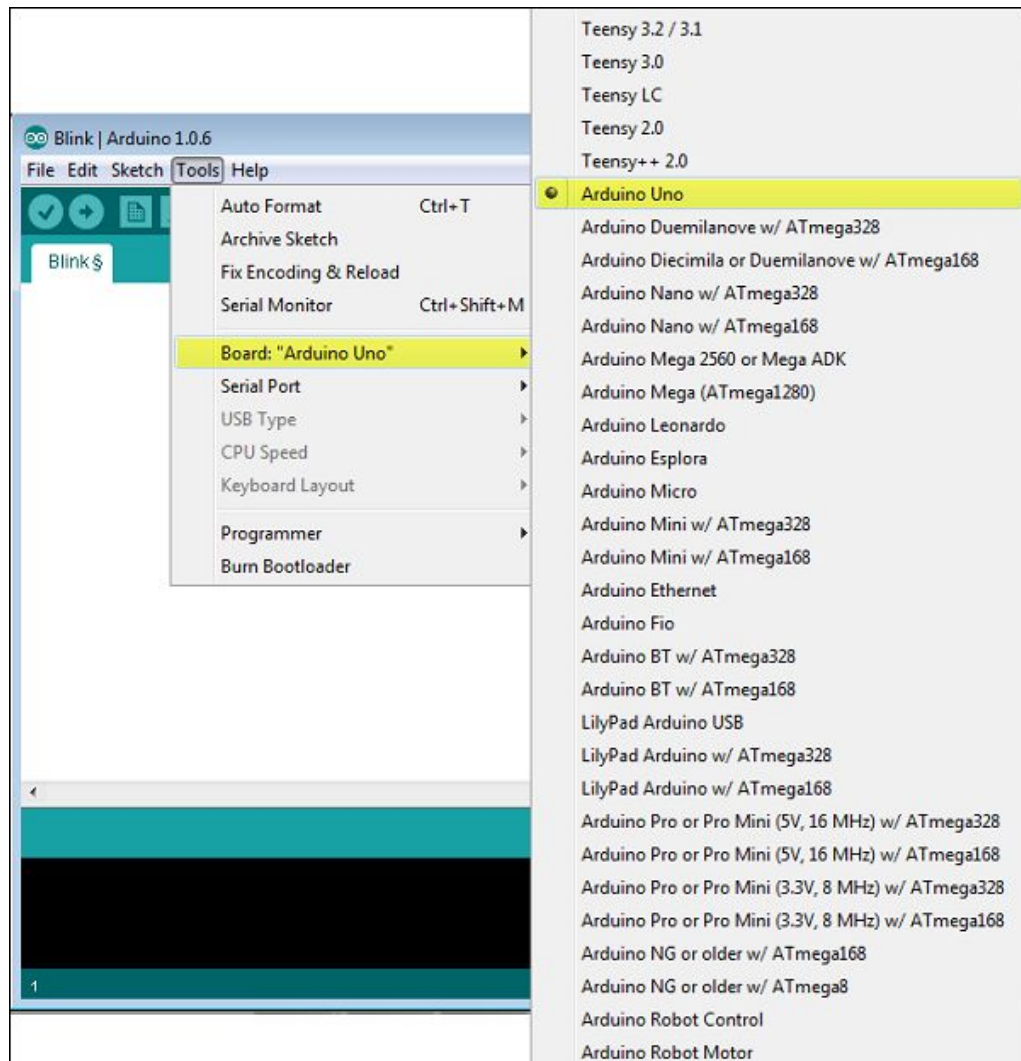


Here, we are selecting just one of the examples with the name Blink. It turns the LED on and off with some time delay. You can select any other example from the list.

Step 6 – Select your Arduino board.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

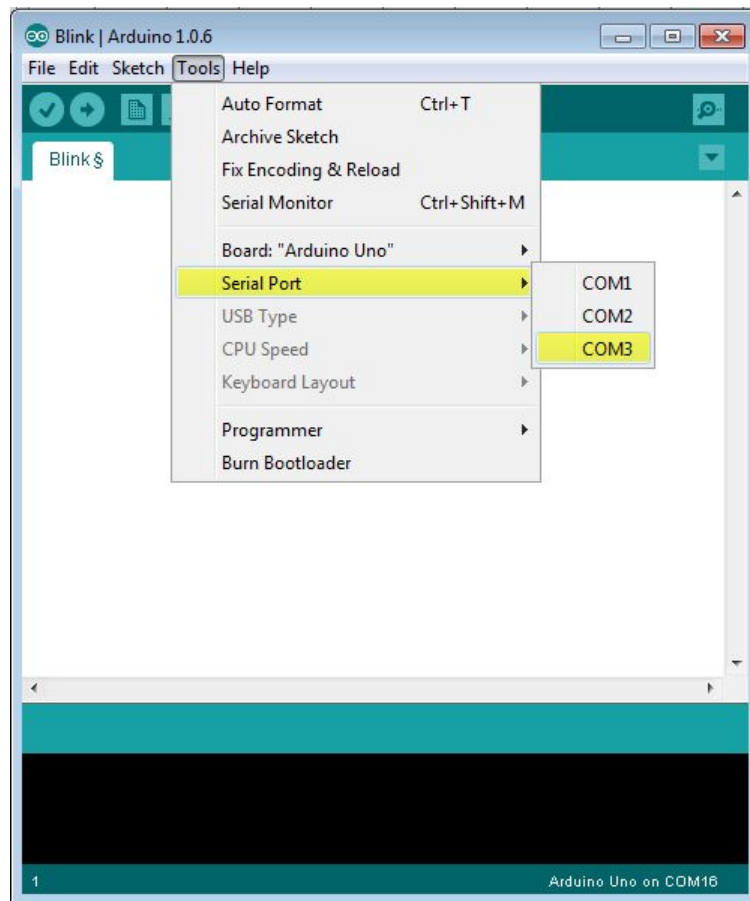
Go to Tools → Board and select your board.



Here, we have selected the Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

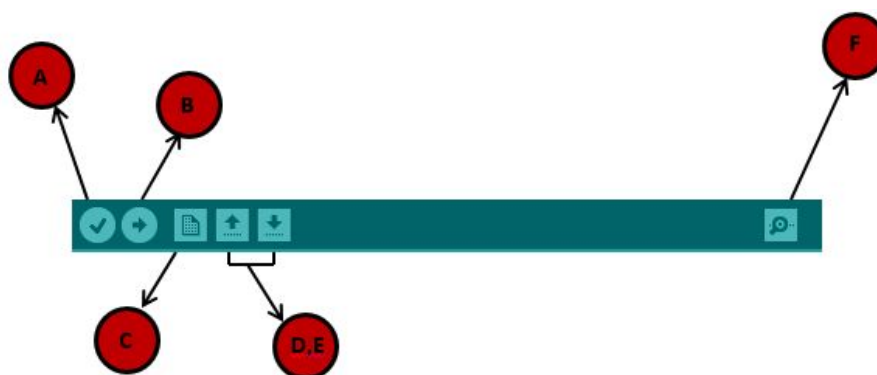
Step 7 – Select your serial port.

Select the serial device of the Arduino board. Go to Tools → Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.



Step 8 – Upload the program to your board.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



- A – Used to check if there is any compilation error.
- B – Used to upload a program to the Arduino board.
- C – Shortcut used to create a new sketch.
- D – Used to directly open one of the example sketches.

E – Used to save your sketch.

F – Serial monitor is used to receiving serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Note – If you have an Arduino Mini, NG, or other board, you need to press the reset button physically on the board, immediately before clicking the upload button on the Arduino Software.

1.3-Program Structure of Arduino

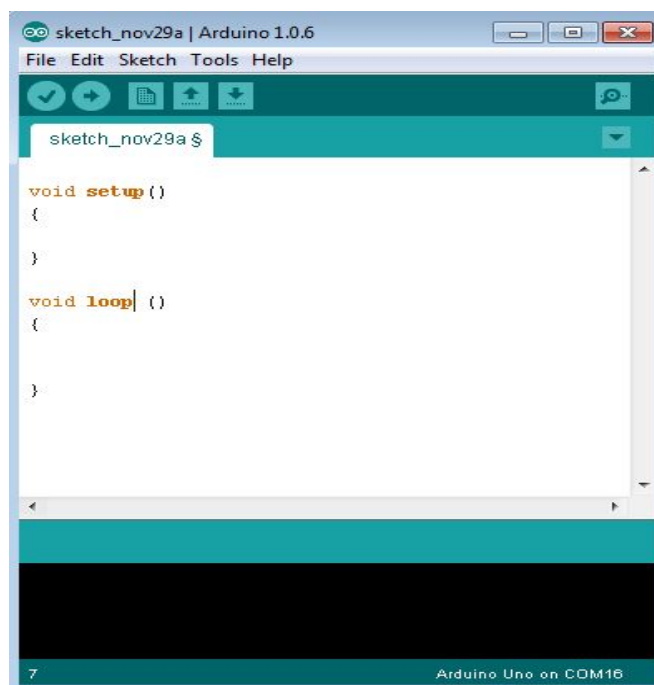
The Arduino software is open-source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL.

Sketch – The first new terminology is the Arduino program called “sketch”.

Arduino programs can be divided into three main parts: Structure, Values (variables and constants), and Functions. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

Let us start with the Structure. Software structure consists of two main functions –

- Setup() function
- Loop() function



Void setup () {}

- PURPOSE – The setup() function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power-up or reset of the Arduino board.
- INPUT – -
- OUTPUT – -
- RETURN – -

Void Loop () {}

- PURPOSE – After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.
- INPUT – -
- OUTPUT – -
- RETURN – -

SECTION 2: SENSORS, ACTUATORS, AND DISPLAYS

2.1-Sensors

A sensor detects (senses) changes in the ambient conditions or the state of another device or a system, and forwards or processes this information in a certain manner. They perform some input functions by sensing or feeling the physical changes in characteristics of a system in response to stimuli. For example, heat is converted to electrical signals in a temperature sensor, or atmospheric pressure is converted to electrical signals in a barometer.

“A device which detects or measures a physical property and records, indicates, or otherwise responds to it”.

- Oxford Dictionary

2.2-Actuators

An actuator is a component of a machine or system that moves or controls the mechanism or the system. An actuator is a mechanism by which a control system acts upon an environment. An actuator requires a control signal and a source of energy. Upon receiving a control signal, the actuator responds by converting the energy into mechanical motion. The control system can be simple (a fixed mechanical or electronic system), software-based (e.g. a printer driver, robot control system), a human, or any other input.

2.3-Displays

A display device is an output device for the presentation of information in a visual or tactile form (the latter used for example in tactile electronic displays for blind people). When the input information that is supplied has an electrical signal the display is called an electronic display. Here, I have discussed the interfacing of some common display boards with Arduino Uno.

2.4-List of the Components (Sensors, Actuators, And Displays)

A list of the sensors and actuators are given below. We will interface these sensors, actuators and displays with Arduino Uno through-out this report.

Sl. No.	COMPONENTS	Type
01	Temperature and Humidity Sensor (DHT22, TMP36, MCP9808, Thermistor)	Sensor
02	Environment Sensor (BMP180, BME280)	Sensor
03	Motion Sensor & Accelerometer (ADXL335, MPU6050)	Sensor
04	Ultrasonic Distance Sensor	Sensor
05	Passive Infrared Sensor (PIR Sensor)	Sensor
06	RGB Color Sensor (TCS3200, TCS34725 [ADA1334])	Sensor
07	Gas or Smoke Sensor (MQ135)	Sensor
08	Liquid Crystal Display (LCD) Module (Size: 2x16)	Display
09	Seven Segment Displays	Display
10	An 8x8 LED matrix display	Display
11	DC Motors	Actuator
12	Servo motors	Actuator

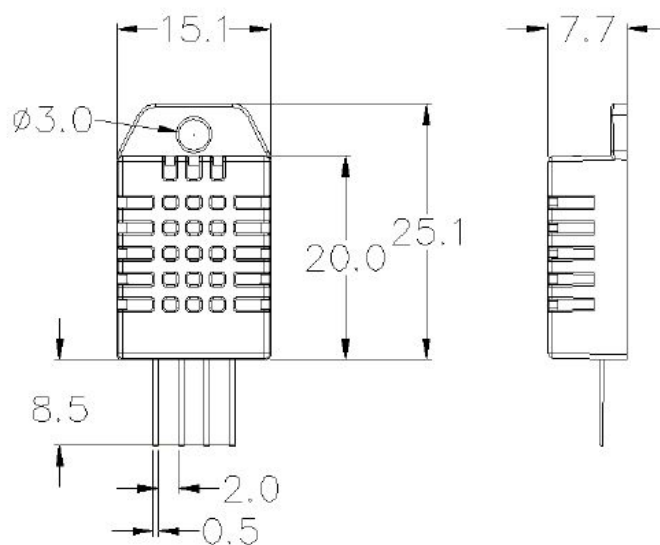
SECTION 3: INTERFACING OF SENSORS WITH ARDUINO UNO

In this section, we will interface the sensors (that I have mentioned in the list of components so far) with Arduino Uno. I have divided it into four parts for each sensor. i.e. Technical Specifications, How to Use the Sensor, Circuit Diagram (Schematics), and sketch. I have mentioned my GITHUB REPOSITORY Link in the box below. It contains all of the sketch of this section. You need to clone this repository before you go through this section.

Github Link: <https://github.com/joydipdutta001/ArduinoStepByStep>

3.1-Interfacing of Temperature and Humidity Sensor (DHT22)

DHT22 is playing an important role in our environment to measure temperature and moisture. It is a low cost easy to use small sensor. This sensor is used at different weather stations to measure temperature and ratio of moisture in the air, in this way they tell about temperature or rain prediction. It is a temperature and humidity measure sensing device. It is easily used but it needs a specific time for an operation. This sensor measures moisture content and temperature. This sensor is easily connected with other microcontrollers.



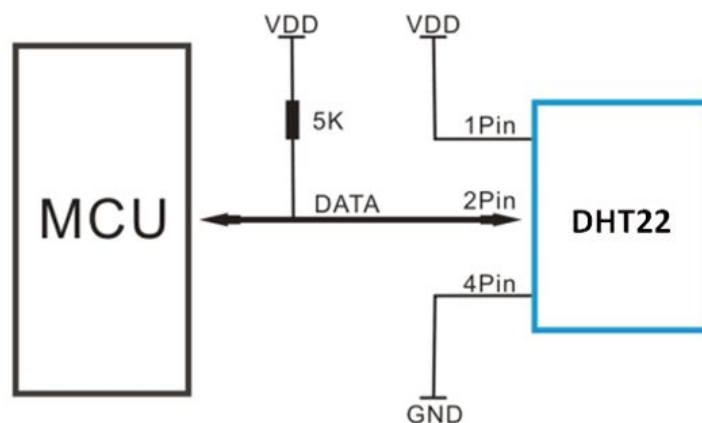
3.1.1-Technical Specifications:

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)

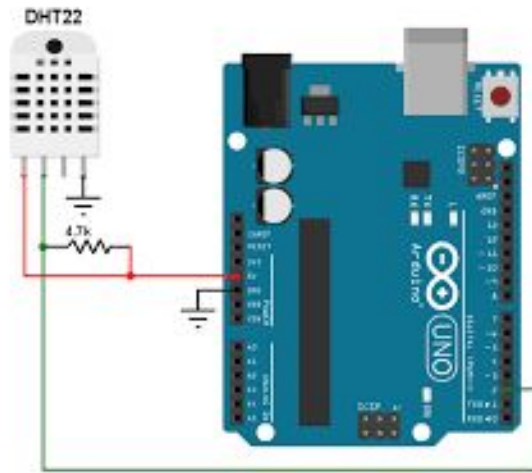
- Output: Serial data
- Temperature Range: -40°C to 80°C
- Humidity Range: 0% to 100%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: $\pm 0.5^{\circ}\text{C}$ and $\pm 1\%$

3.1.2-How to Use DHT22 Sensor module:

The DHT22 sensor is factory calibrated and outputs serial data, and hence it is super easy to set it up. The connection diagram for this sensor is shown below. As you can see in the figure the data pin is connected to an I/O pin of the Arduino and a 5K pull-up resistor is used (You can use the resistors in between 4k-5k ohm). This data pin outputs the value of both temperature and humidity as serial data. If you are trying to interface DHT22 with Arduino Uno then there are ready-made libraries for it which will give you a quick start.



3.1.3-Circuit Diagram (Schematics):



Pin	Wiring to Arduino Uno
1st pin- VCC	5V
2nd pin- DataOUT	Digital pin 2
3rd pin	Not need to connect
4th pin- GND	GND

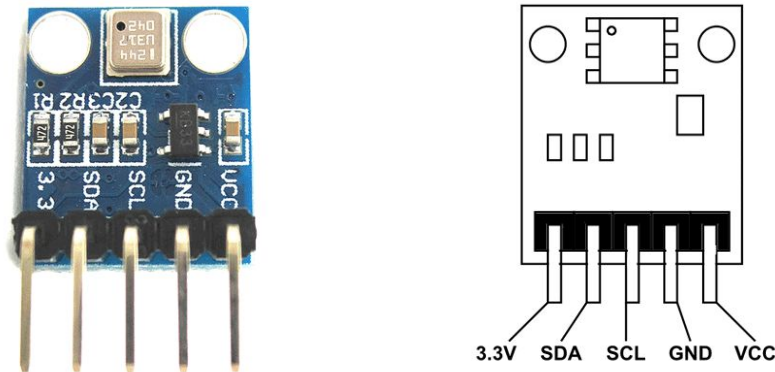
3.1.4-Sketches of Temperature Sensor Modules

Sketch: You can explore the sketches under the “3_1_Temperature_Sensor” folder in the GitHub Repository that I have mentioned in the introduction part of the SECTION 3.

3.2-Environmental Sensor (BMP180)

The BMP180 barometric sensor (model GY-68) is the one in the following figure (front and back view). It is a very small module with 1mm x 1.1mm (0.039in x 0.043in).It

measures the absolute pressure of the air around it. It has a measuring range from 300hPa to 1100hPa with an accuracy down to 0.02hPa. It can also measure altitude and temperature. The BMP180 barometric sensor communicates via an I2C interface. This means that it communicates with the Arduino using just 2 pins.



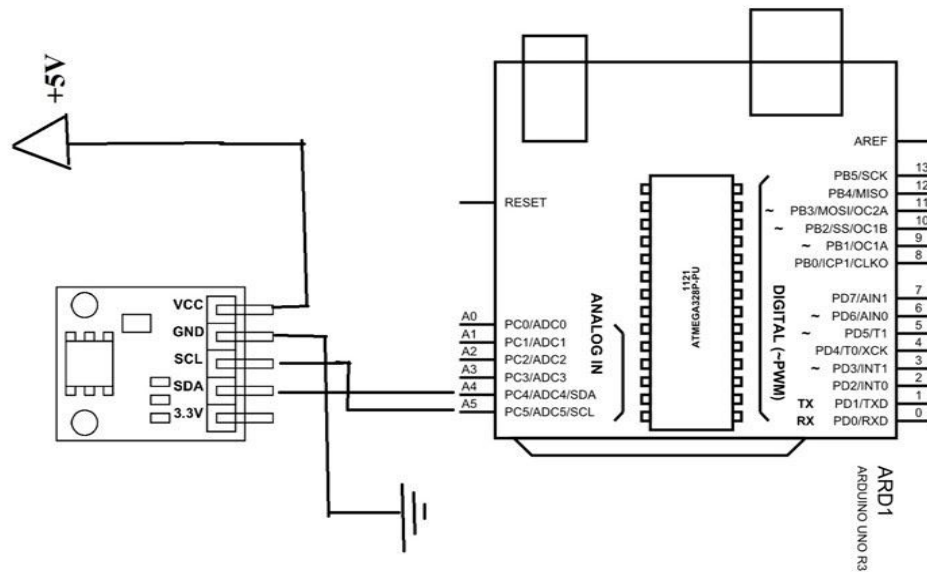
3.2.1-Technical Specifications:

- Operating voltage of BMP180: 1.3V – 3.6V
- An input voltage of BMP180MODULE: 3.3V to 5.5V
- Peak current: 1000uA
- Consumes 0.1uA standby
- Maximum voltage at SDA , SCL : VCC + 0.3V
- Operating temperature: -40°C to +80°C

3.2.2 How to Use MP180 Module?

There are only two pins present to communicate with the module. And this communication is the I2C interface. The data is sent to the module or received from the module through the I2C interface. So we have to get the information about barometric pressure and temperature through this interface. A typical circuit diagram of BMP180 with ARDUINO is shown below. Although connecting to the BMP180 sensor module is easy communicating with it is not easy. The data exchange between the controller of ARDUINO and the module is complex. Usually, to send information to the module or receive the information from it we have to follow the protocol. This protocol is a sequence of steps to be followed without an error. These steps are complex to follow for starters. So using libraries that are pre-written for the module is ideal. Using libraries makes communication easy. All you need to do is download these libraries and call them in programs. Once the header file is included, the ARDUINO follows the protocol automatically and decodes the required data. Once this data is available we can perform functions of desire.

3.2.3-Circuit Diagram (Schematics):



Pin	Wiring to Arduino Uno
Vin	5V
GND	GND
SCL	A5
SDA	A4

3.2.4-Sketches of Environment Sensor Module

To control the BMP180 barometric sensor, you need to install the SFE_BMP180 Library.

- **Installing the SFE_BMP180 Library**

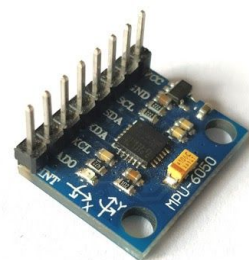
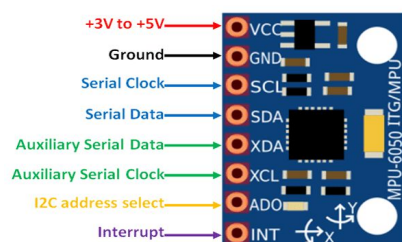
1. click [here to download the SFE_BMP180 library](#). You should have a .zip folder in your Downloads folder
2. Unzip the .zip folder and you should get the BMP180_Breakout_Arduino_Library-master folder
3. Rename your folder from BMP180_Breakout_Arduino_Library-master to BMP180_Breakout_Arduino_Library
4. Move the BMP180_Breakout_Arduino_Library folder to your Arduino IDE installation libraries folder
5. Finally, re-open your Arduino IDE Uploading the sketch

Go to File “3_2_Environment_Sensor\Sketches\SFE_BMP180_example”. This example is very well commented on and explained how the sensor reads the pressure, temperature, and computes the altitude. Don’t upload the sketch now, you need to set the altitude first. Before uploading the sketch, you need to set up your current altitude. Go to elevationmap.net, insert your address, and check your altitude’s location. Set your altitude in the sketch. The place where you should write your altitude is commented.

Sketch: You can explore the sketches under the “3_2_Environment_Sensor” folder in the GitHub Repository that I have mentioned in the introduction part of the SECTION 3.

3.3-Motion Sensor and Accelerometer (MPU6050)

The MPU-60X0 is the world’s first integrated 6-axis motion tracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP) all in a small 4x4x0.9mm package. With its dedicated I2C sensor bus, it directly accepts inputs from an external 3-axis compass to provide a complete 9-axis MotionFusion™ output.



3.3.1-Technical Specifications:

- MEMS 3-axis accelerometer and 3-axis gyroscope values combined
- Power Supply: 3-5V
- Communication: I2C protocol
- Built-in 16-bit ADC provides high accuracy
- Built-in DMP provides high computational power
- In-built temperature sensor

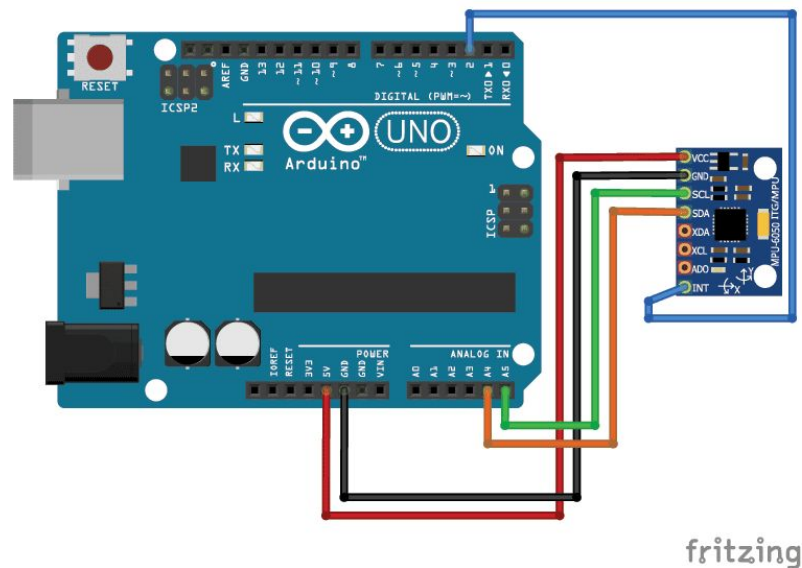
3.3.2-How to Use MPU6050 Module?

It is very easy to **interface the MPU6050 with Arduino**, thanks to the library developed by Jeff Rowberg. You can download the library from [Jeff Rowberg MPU6050 Library](#) for Arduino. Once you have added this library to your Arduino IDE, follow the below schematics to establish an I2C connection between your **Arduino and MPU6050**. The library provides two example programs, which can be found at File -> Examples -> MPU6050. In these two examples, one will give raw values while the other will give optimized values using the DMP. The following data values can be obtained using this example program.

- Quaternion Components [w, x, y, z]
- Euler angles
- Yaw, Pitch, Roll
- Real-world Acceleration
- World frame acceleration
- Teapot invent sense Values

Out of all these data, the Yaw, Pitch, Roll us commonly used. However, the library is capable of performing more than that and can be used for different purposes. Once the program is uploaded, open the serial monitor and set it to 115200 baud rate and you should see the data being printed on the screen.

3.3.3-Circuit Diagram (Schematics):



Pin	Wiring to Arduino Uno
Vin	5V

GND	GND
SCL	A5
SDA	A4
INT	2

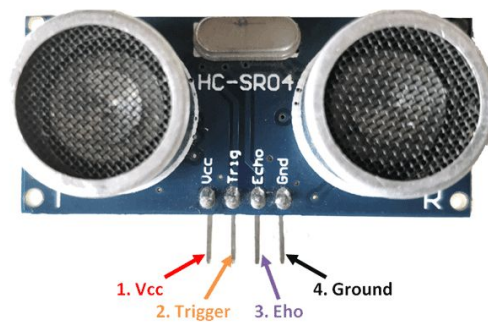
3.3.4-Sketches of MPU6050 Module

Sketch: You can explore the sketches under the “3_3_MPU6050_Sensor” folder in the GitHub Repository that I have mentioned in the introduction part of the SECTION 3.

3.4-Ultrasonic Distance Sensor (HC-SR04)

The HC-SR04 ultrasonic sensor uses SONAR to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1” to 13 feet.

The operation is not affected by sunlight or black material, although acoustically, soft materials like cloth can be difficult to detect. It comes complete with an ultrasonic transmitter and a receiver module.



3.4.1-Technical Specifications:

- Operating voltage: +5V
- Theoretical Measuring Distance: 2cm to 450cm

- Practical Measuring Distance: 2cm to 80cm
- Accuracy: 3mm
- Measuring angle covered: <15°
- Operating Current: <15mA
- Operating Frequency: 40Hz

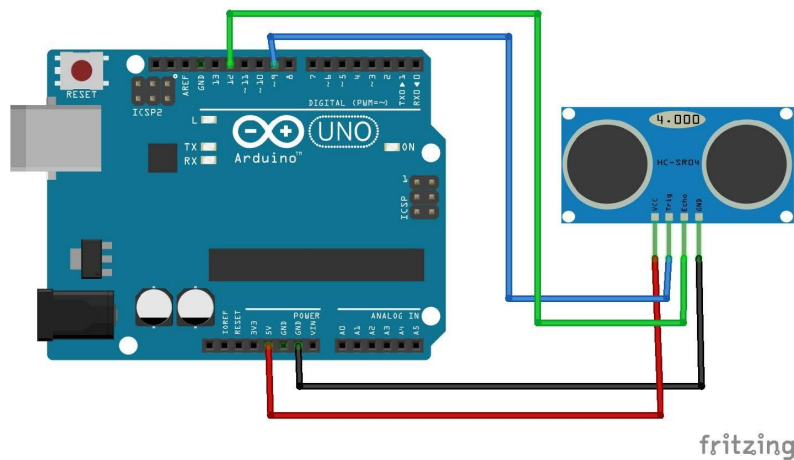
3.4.2 How to Use HC-SR04 Module?

As shown above, the HC-SR04 Ultrasonic (US) sensor is a 4 pin module, whose pin names are Vcc, Trigger, Echo, and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that is

$$\text{Distance} = \text{Speed} \times \text{Time}$$

The HC-SR04 distance sensor is commonly used with both microcontroller and microprocessor platforms like Arduino, ARM, PIC, Raspberry Pie, etc. The following guide is universal since it has to be followed irrespective of the type of computational device used. Power the Sensor using a regulated +5V through the Vcc and Ground pins of the sensor. The current consumed by the sensor is less than 15mA and hence can be directly powered by the onboard 5V pins (If available). The Trigger and the Echo pins are both I/O pins and hence they can be connected to the I/O pins of the microcontroller. To start the measurement, the trigger pin has to be made high for 10uS and then turned off. This action will trigger an ultrasonic wave at a frequency of 40Hz from the transmitter and the receiver will wait for the wave to return. Once the wave is returned after it gets reflected by any object the Echo pin goes high for a particular amount of time which will be equal to the time taken for the wave to return to the sensor. The amount of time during which the Echo pin stays high is measured by the MCU/MPU as it gives the information about the time taken for the wave to return to the Sensor. Using this information the distance is measured as explained in the above heading.

3.4.3-Circuit Diagram (Schematics):



Pin	Wiring to Arduino Uno
VCC	5V
Trig	9
Echo	12
GND	GND

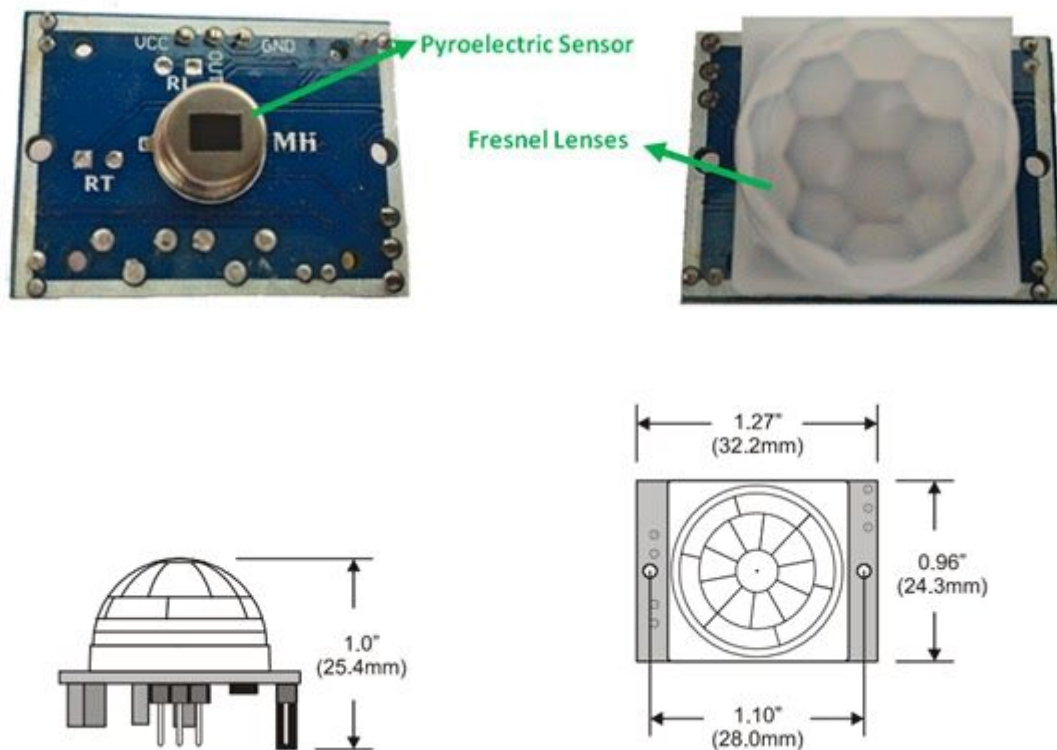
3.4.4-Sketches of UltraSonic Sensors Modules

Sketch: You can explore the sketches under the “3_4_UltraSonic_Sensor” folder in the GitHub Repository that I have mentioned in the introduction part of the SECTION 3.

3.5-Passive Infrared Sensor (PIR)

PIR sensors allow you to sense motion. They are used to detect whether a human has moved in or out of the sensor’s range. They are commonly found in appliances and gadgets used at home or for businesses. They are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors. PIRs are made of pyroelectric sensors, a round metal can with a rectangular crystal in the center, which can detect levels of infrared radiation. Everything emits low-level radiation, and the hotter something is, the more radiation is emitted. The sensor in a motion detector is split into two halves. This is to detect motion (change) and not average IR levels. The two halves are connected so that they cancel out each

other. If one-half sees more or less IR radiation than the other, the output will swing high or low.



3.5.1-Technical Specifications:

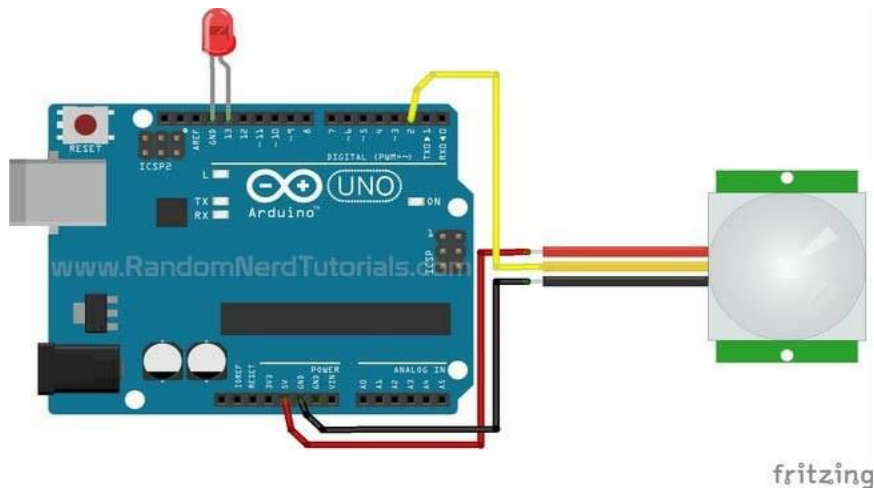
- Wide range on input voltage varying from 4.V to 12V (+5V recommended)
- The output voltage is High/Low (3.3V TTL)
- Can distinguish between object movement and human movement
- Has to operate modes - Repeatab(H) and Non- Repeatab(H)
- It can cover a distance of about 120° and 7 meters
- Low power consumption of 65mA
- Operating temperature from -20° to +80° Celsius

3.5.2-How to Use PIR Module?

The PIR sensor stands for Passive Infrared sensor. It is a low-cost sensor that can detect the presence of Human beings or animals. This sensor has three output pins Vcc, Output, and Ground as shown in the pin diagram above. Since the output pin is 3.3V TTL logic it can be used with any platforms like Arduino, Raspberry, PIC, ARM, 8051, etc. The module can be powered from voltage 4.5V to 20V but typically 5V is used. Once the module is powered, allow the module to calibrate itself for a few minutes, 2 minutes is a well-settled time. Then observe the output on the output pin. Before we analyze the output we need to

know that there are two operating modes in this sensor such as Repeatable(H) and Non-Repeatable(L) and mode. The Repeatable mode is the default mode. The output of the sensor can be set by shorting any two pins on the left of the module as shown below. You can also notice two orange color potentiometers that can be used to set the sensitivity and time which will be explained further below.

3.5.3-Circuit Diagram (Schematics):



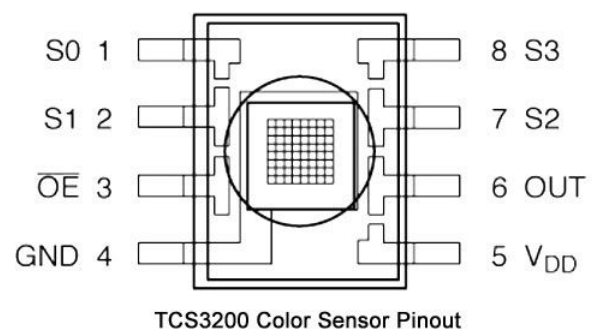
Pin	Wiring to Arduino Uno
VCC	5V
OUT	2
GND	GND

3.5.4-Sketches of PIR Sensor Modules

Sketch: You can explore the sketches under the “3_5_PIR_Sensor” folder in the GitHub Repository that I have mentioned in the introduction part of the SECTION 3.

3.6-Color Sensor Module (TCS320)

This Arduino compatible TCS3200 color sensor module consists of a TAOS TCS3200 RGB sensor chip and 4 white LEDs. The main part of the module is the TCS3200 chip which is a Color Light-to-Frequency Converter. The white LEDs are used for providing proper lighting for the sensor to detect the object's color correctly. This chip can sense a wide variety of colors and it gives the output in the form of corresponding frequency. This module can be used for making color sorting robots, test strip reading, color-matching tests, etc. The **TCS3200 chip** consists of an 8 x 8 array of photodiodes. Each photodiode has either a red, green, or blue filter, or no filter. The filters of each color are distributed evenly throughout the array to eliminate location bias among the colors. Internal circuits include an oscillator which produces a square-wave output whose frequency is proportional to the intensity of the chosen color.



3.6.1-Technical Specifications

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

3.6.2-How to Use TCS3200 Color Sensor Module?

TCS3200 color sensor module can be used to detect the colors with the help of a microcontroller. The microcontroller is measuring the output frequency from the 6th pin. To determine the color of an object, we've to measure the frequency from the 6th pin when each filter is activated. Set both S2 and S3 to LOW, measure the frequency. Now we get the intensity of the RED component in the object. Set S2 to LOW and S3 to HIGH in order to get the intensity of the BLUE component in the object. Set both S2 and S3 to HIGH and get the intensity of the GREEN component in the object. Compare the frequencies of the three components to get the actual color of the object.

Note: In Arduino, we can use the 'pulse' command to get the frequency variations.

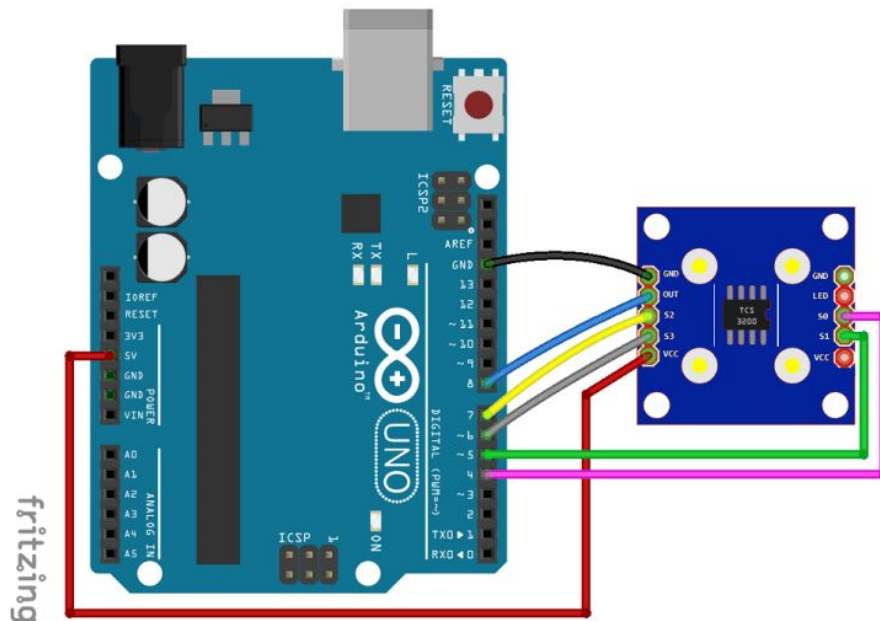
```
e.g.:  
digitalWrite(S2, LOW);  
digitalWrite(S3, LOW);    //Activating photodiode with red  
filter  
red = pulseIn(outpin, LOW);
```

Here we get the value corresponding to the red color component of the object color. Similarly, we've to activate each photodiode by changing the S2 and S3 states and read the corresponding values of green and blue color components of the object color. For a Red object, we get an approximate value of red=16, green=53, and blue=45. This may vary from ambient light and experiment setup. For good results, it's better to cover the object and sensor from ambient light.

Programming Logic

- First set the input pins as input and output pins as output. No need to use analog pins.
- Set S0 and S1 to high or low to set desired frequency scaling.
- In the loop, activate each filter by setting S2 and S3 to HIGH or LOW and measure frequency 'fo' from the 6th pin to get corresponding color intensity. Compare the frequencies of each color to determine the color of the object.

3.6.3-Circuit Diagram (Schematics)



Pin	Wiring to Arduino Uno
S0	Digital Pin 4
S1	Digital Pin 5
VCC	5V
S3	Digital Pin 6
S4	Digital Pin 7
OUT	Digital Pin 8

3.6.4-Sketches of Color Sensor Modules

Sketch: You can explore the sketches under the “3_6_RGBColor_Sensor” folder in the GitHub Repository that I have mentioned in the introduction part of the SECTION 3.

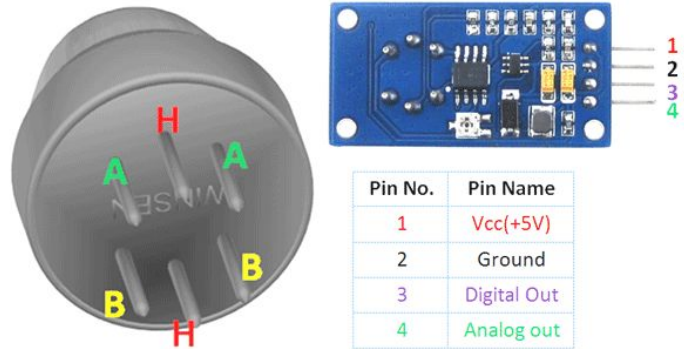
3.7-Gas Sensor Module (MQ135)

When it comes to measuring or detecting a particular Gas the MQ series Gas sensors are the most inexpensive and commonly used ones. MQ135 is available as a module or as just the sensor alone. If you are trying to only detect (not measuring PPM) the presence of gas then you can buy it as a module since it comes with an op-amp comparator and a digital output pin. But if you plan to measure the PPM of a gas it is recommended buying the sensor

alone without a module. The MQ-135 Gas sensors are used in air quality control equipment and are suitable for detecting or measuring NH₃, NO_x, Alcohol, Benzene, Smoke, CO₂. The MQ-135 sensor module comes with a Digital Pin which makes this sensor operate even without a microcontroller and that comes in handy when you are only trying to detect one particular gas. If you need to measure the gases in PPM the analog pin needs to be used. The analog pin is TTL driven and works on 5V and so can be used with the most common microcontrollers. If you are looking for a sensor to detect or measure common air quality gases such as CO₂, Smoke, NH₃, NO_x, alcohol, Benzene then this sensor might be the right choice for you.

Alternative MQ Gas sensors

Sensor Name	Gas to Measure
MQ-2	Methane, Butane, LPG, Smoke
MQ-3	Alcohol, Ethanol, Smoke
MQ-4	Methane, CNG Gas
MQ-5	Natural gas, LPG
MQ-6	LPG, butane
MQ-7	Carbon Monoxide
MQ-8	Hydrogen Gas
MQ-9	Carbon Monoxide, flammable gasses
MQ131	Ozone
MQ135	Air Quality
MQ136	Hydrogen Sulphide gas
MQ137	Ammonia
MQ138	Benzene, Toluene, Alcohol, Propane, Formaldehyde gas, Hydrogen
MQ214	Methane, Natural Gas
MQ216	Natural gas, Coal Gas
MQ303A	Alcohol, Ethanol, smoke
MQ306A	LPG, butane
MQ307A	Carbon Monoxide
MQ309A	Carbon Monoxide, flammable gas



3.7.1-Technical Specifications

- Wide detecting scope
- Fast response and High sensitivity
- Stable and long life
- Operating Voltage is +5V
- Detect/Measure NH₃, NO_x, alcohol, Benzene, smoke, CO₂, etc.
- Analog output voltage: 0V to 5V
- Digital output voltage: 0V or 5V (TTL Logic)
- Preheat duration 20 seconds
- Can be used as a digital or analog sensor

3.7.2-How to Use MQ135 Gas Sensor Module?

A. MQ135 Sensor to detect Gases:

You can either use the digital pin or the analog pin to do this. Simply power the module with 5V and you should notice the power LED on the module to glow and when no gas is detected the output LED will remain turned off meaning the digital output pin will be 0V. Remember that these sensors have to be kept on for pre-heating time (mentioned in the features above) before you can work with it. Now, introduce the sensor to the gas you want to detect and you should see the output LED to go high along with the digital pin, if not use the potentiometer until the output gets high. Now every time your sensor gets introduced to this gas at this particular concentration the digital pin will go high (5V) else will remain low (0V). You can also use the analog pin to achieve the same thing. Read the analog values (0-5V) using a microcontroller, this value will be directly proportional to the concentration of the gas to which the sensor detects. You can experiment with these values and check how the sensor reacts to different concentrations of gas and develop your program accordingly.

B. MQ135 Sensor to measure PPM:

MQ-135 gas sensor applies SnO₂ which has higher resistance in the clear air as a gas-sensing material. When there is an increase in polluting gases, the resistance of the gas

sensor decreases along with that. To measure PPM using the MQ-135 sensor we need to look into the (R_s/R_o) v/s PPM graph taken from the MQ135 datasheet. The above figure shows the typical sensitivity characteristics of the MQ-135 for several gases. in their: Temp: 20, Humidity: 65%, O₂ concentration 21%, RL=20k Ω ,

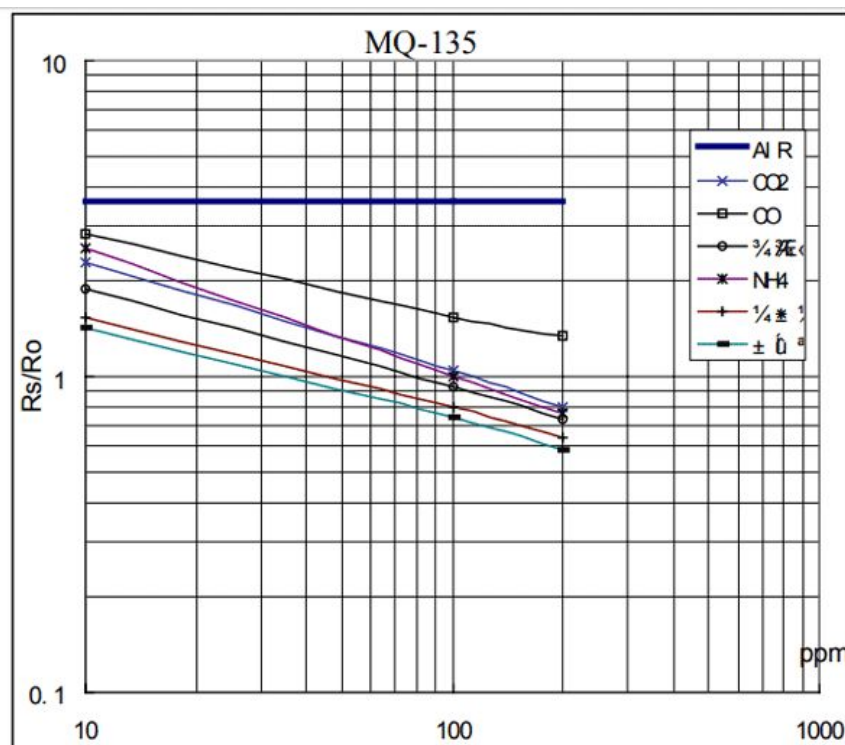
R_o : sensor resistance at 100ppm of NH₃ in the clean air.

R_s : sensor resistance at various concentrations of gases.

The value of R_o is the value of resistance in fresh air (or the air with we are comparing) and the value of R_s is the value of resistance in Gas concentration. First, you should calibrate the sensor by finding the values of R_o in the fresh air and then use that value to find R_s using the below formula:

Resistance of sensor(R_s): $R_s = (V_c/V_{RL} - 1) \times R_L$

Once we calculate R_s and R_o we can find the ratio and then using the graph shown above we can calculate the equivalent value of PPM for that particular gas.



3.7.3-Circuit Diagram (Schematics)



Pin	Wiring to Arduino Uno
A0	Analog pin A0
VCC	5V
GND	GND

3.7.4-Sketches of Gas Sensor Modules

Sketch: You can explore the sketches under the “3_7_Gas_Sensors” folder in the GitHub Repository that I have mentioned in the introduction part of the SECTION 3.

SECTION 4: INTERFACING OF DISPLAYS WITH ARDUINO UNO

The Arduino Board has a wide variety of compatible displays that you can use in your electronic projects. In most projects, it's very useful to give the user some sort of feedback from the Arduino. Whether it's a sensor reading, an "OK" message or to create an interface to interact with your Arduino Board. In this section you will interface some common display modules and graphics screens with Arduino Uno. Before you go through this section, clone the GITHUB repository that I have shared in the box below for the Sketches and Datasheets of the Display Modules.

GitHub: <https://github.com/joydipdutta001/ArduinoStepByStep>

4.1-Liquid Crystal Display (LCD) Module (Size: 2x16)

Liquid Crystal Display (LCD) is widely used in various electronic applications. It is commonly used in various systems to show different status and parameters. LCD16x2 has 2 lines with 16 characters in each line. Each character is made up of a 5x8 (column x row) pixel matrix.




4.1.1-Pin Description:

First two pins of LCD16x2 are used for ground and supply (+5 V).

→ 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.



LCD 16x2

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: Command 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

EW

➔ **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.

4.1.2-Circuit Diagram (Schematics)

4.1.3-How to display Data On LCD (Custom Characters And Regular Characters)?

While you will finish the wiring connections according to the figure-4.1.4, visit the link below and copy the sketch into your Arduino IDE or If you have already clone the github repository in your system then open the folder named as “4_1_LCD_MODULES”. Now you need to install a library named as “LiquidCrystal” through the library manager under the toolbar. Now you can upload the sketch in your Arduino Uno Board. There are some predefined functions available in the library that you have installed. I have used some functions in the sketch. Let’s see how they works.

Functions Used in the Sketch:

LiquidCrystal object_name(rs,rw,en,d0,d1,d2,d3,d4,d5,d6,d7)

LiquidCrystal object_name(rs,rw,en,d4,d5,d6,d7)

- This function defines an object named object_name of the class LiquidCrystal.
- rs, rw and en are the pin numbers of the Arduino board that are connected to rs, rw and en of LCD.

- d0, d1, d2, d3, d4, d5, d6 and d7 are the pin numbers of the Arduino board that are connected to data pins D1, D2, D3, D4, D5, D6 and D7 of LCD.
- Example, `LiquidCrystal lcd(13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3)`. This makes use of LCD in 8-bit mode.
- Example, `LiquidCrystal lcd(13, 12, 11, 6, 5, 4, 3)`. This makes use of LCD in 4-bit mode.

`lcd.begin(cols,rows)`

- This function is used to define the number of rows and columns the LCD has and to initialize the LCD.
- Needs to be called before calling other functions, once the object is defined using the function in point 1.
- Example, for 16x2 LCD we write `lcd.begin(16,2)`. `lcd` is the name of the object of the class `LiquidCrystal`. 16 is the number of columns and 2 is the number of rows.

`lcd.setCursor(col,row)`

- This function positions the cursor of the LCD to a location specified by the row and column parameters.
- `col` is the column number at which the cursor should be at (0 for column 1, 4 for column 5 and so on).
- `row` is the row number at which the cursor should be at (0 for row 1, 1 for row 2).
- Example, for setting the cursor at the 5th column in the 2nd row, `lcd.setCursor(4,1)`. `lcd` is the name of the object of the class `LiquidCrystal`.

`lcd.createChar(num,data)`

- This function is used to create a new custom character for use on the LCD.
- `num` is the CGRAM location (0 to 7) at which the custom character is to be stored.
- `data` is array of eight bytes which represent the custom character.
- Custom character can be of 5x8 pixels only.
- Each custom character is specified by an array of eight bytes, one for each row. The five least significant bits of each byte determine the pixels in that row.
- To display a custom character on the screen, `write()` function needs to be used. CGRAM location number (0 to 7) of the custom character which is to be displayed on LCD is passed as an argument to the write function.

- Note : When referencing custom character "0", you need to cast it as a byte, otherwise the compiler throws an error.

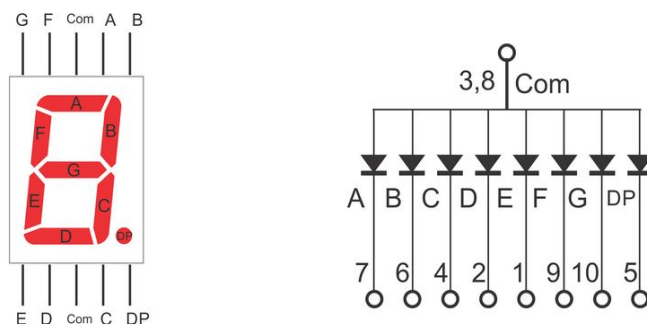
4.1.4-Sketches of LCD Modules

Sketch: You can explore the sketches under the “4_1_LiquidCrystal_Display” folder in the GitHub Repository that I have mentioned in the introduction part of the SECTION 4.

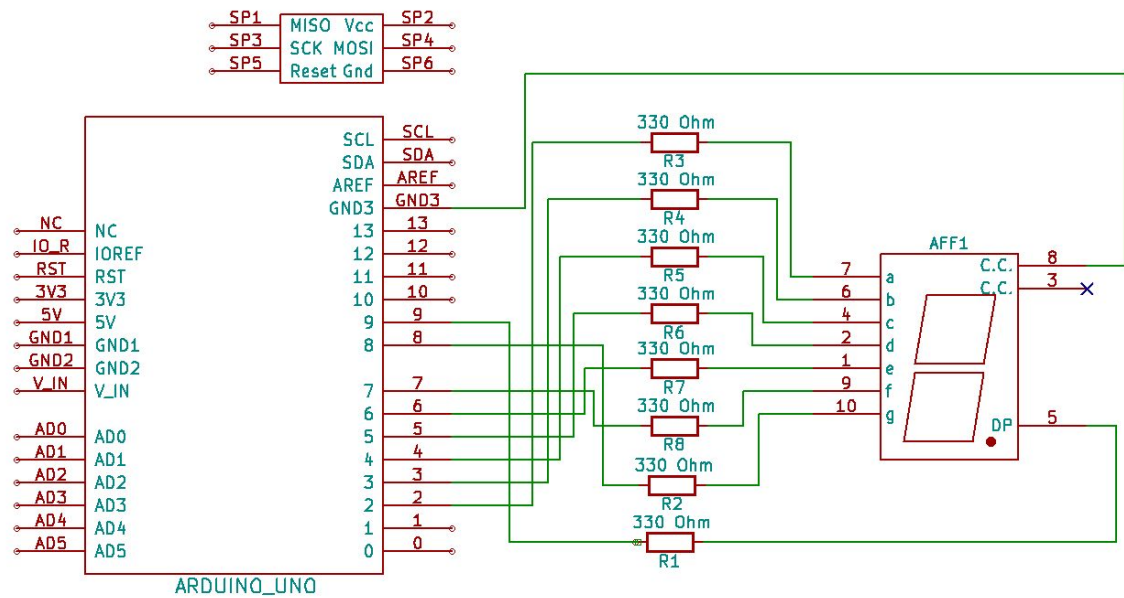
4.2-Seven Segment Displays (Single Digit)

Seven segment displays are used in many embedded system and industrial applications where the range of outputs to be shown is known beforehand. Basic 1 digit seven segment display can show numbers from 0-9 and a few characters. Seven segment displays are of different types; especially they differ in the number of digits/character it can display. Basically a seven segment display is a single unit, which can display only 1 digit or 1 character. More digits are displayed by multiplexing single unit seven segment displays together to form 2 digit display, 3 digit display or 4 digit 7 segment display. We are going to interface a single digit seven segment display without a library to arduino uno. Final result would be displaying 0-9 on the display. Let's identify the pinout and circuit diagram of the display.

4.2.1-Pin Description



4.2.2-Circuit Diagram (Schematics)



4.2.3-How to Program a Single Digit Seven Segment Display

As you can see there are 10 pins in total. You may notice two pins named com, as shown in the circuit diagram all the anodes (+ pins) of the LEDs are connected to these two pins. We call these 2 pins as common anodes and such displays are called Common Anode 7 segment displays. There are some seven segment displays which have common cathodes instead of common anodes. The only difference for common cathode displays is all the cathodes (- pins) are connected together and they are known as Common Cathode 7 segment displays. Apart from these 2 com pins, there are 8 other pins named A,B,C,D,E,F,G and DP. As you can see in the figure, these pins are cathodes (- pins) of the led segments of common anode display (in the case of common cathode display these pins will be anodes) To turn on each segment (LED), we have to supply appropriate voltage to it. Lets just say we want to show the number “3” on the display. To display ‘3’ properly, we need to turn ON A,B,C,D & G segment LED’s (refer the figures). since anodes are common, we supply positive voltage to common anode pins and connect GND (ground) to the segment pins we want to turn ON. If you wire like this on a breadboard. the 7 segment display will show ‘3’ on its display. But hard wiring is not practical, as we need to change the wiring all the time we want to change digit/character to be displayed. We use micro controllers like 8051 or AVR or even Arduino to solve this problem. We can manipulate the 7 segment wiring (the process of turning ON and OFF led segments) using software inside the micro controller. In our tutorial of interfacing with arduino, we achieve this by writing specific lines of arduino sketch to turn led segments ON and OFF.

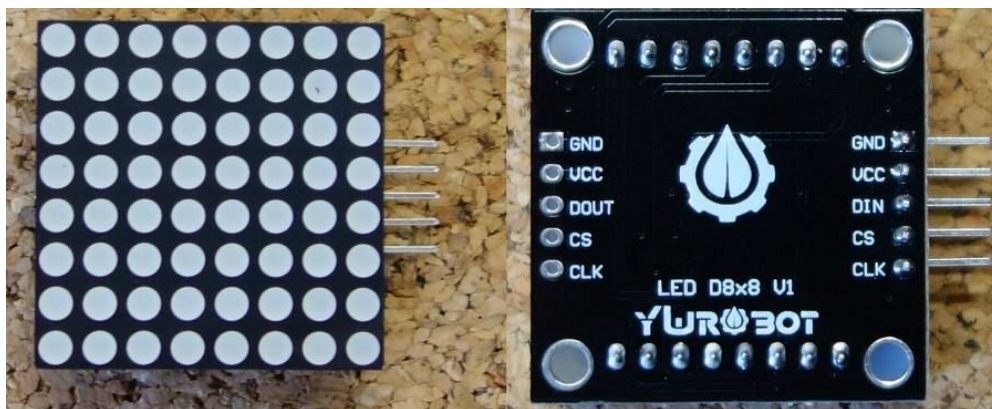
4.2.4-Sketches of Seven Segment Display Modules

Now, you can go through the sketch from the below mentioned github link or If you have already clone the repository in your system then open the folder named as “4_2_1_Seven_Segment_Single_Digit”.Then upload the sketch into you Arduino Uno Board. I have attached some other examples also. You can try those by just simply following the comments that I have provided in each sketch.

Sketch: You can explore the sketches under the “4_2_SevenSegment_Display” folder in the GitHub Repository that I have mentioned in the introduction part of the SECTION 4.

4.3-An 8x8 LED matrix display

The dot matrix that we’re going to use here, is an 8×8 matrix which means that it has 8 columns and 8 rows, so it contains a total of 64 LEDs. The MAX7219 chip makes it easier to control the dot matrix, by just using 3 digital pins of the Arduino board. I think the best option is to buy the dot matrix with the MAX7219 chip as a module, it will simplify the wiring. You can control more than one matrix at a time. For that you just need to connect them to each other, as they have pins in both sides to extend the dot matrix.



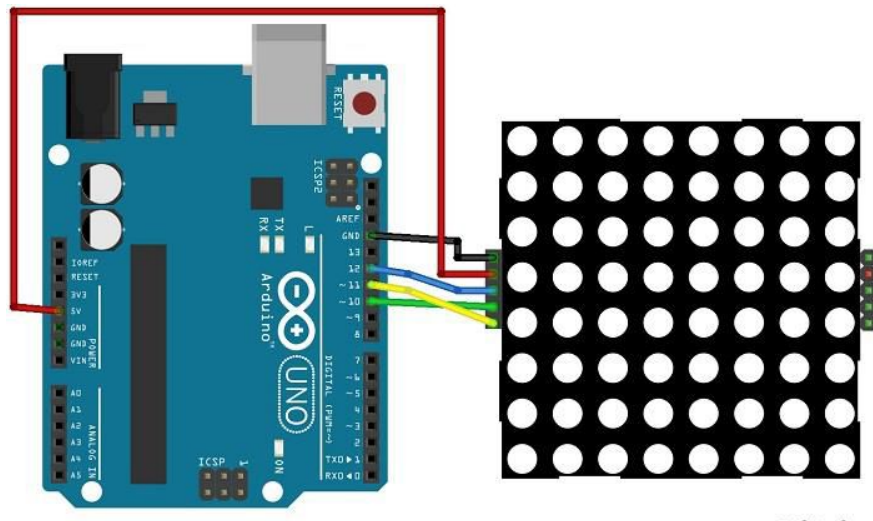
4.3.1-Pin Description

You only need to connect 5 pins from the dot matrix to your Arduino board. The wiring is straightforward:

Dot Matrix Pin	Wiring to Arduino Uno
GND	GND
VCC	5V
DIN	Digital Pin 12

CLK	Digital Pin 11
CS	Digital Pin 10

4.3.2-Circuit Diagram (Schematics)



4.3.3-How to Program DotMatrix Display?

To make it easier to control the dot matrix, you need to download and install in your Arduino IDE the LedControl library. To install the library follow these steps:

- ➔ 1. Open the folder named as “4_3_DotMatrix”. You will find a library folder.
- ➔ 2. Search for a folder named as “LedControl” under the library folder.
- ➔ 3. Move the LedControl folder to your Arduino IDE installation libraries folder.
- ➔ 4. Finally, re-open your Arduino IDE.

The easiest way to display something on the dot matrix is by using the functions `setLed()`, `setRow()` or `setColumn()`. These functions allow you to control one single led, one row or one column at a time.

Here’s the parameters for each function:

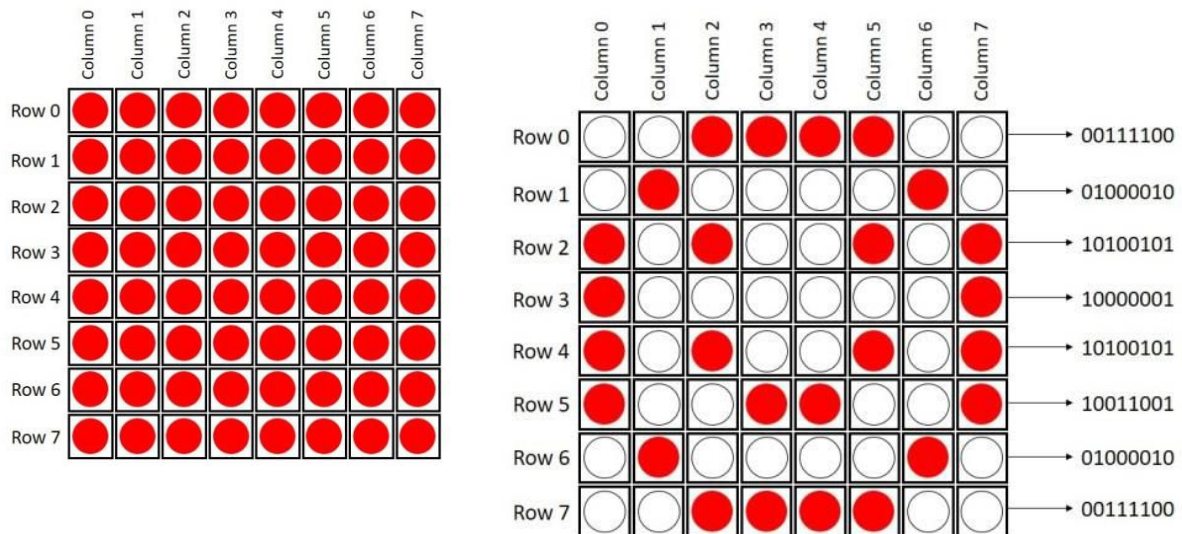
- ➔ **setLed(addr, row, col, state)**
 - ◆ **addr** is the address of your matrix, for example, if you have just 1 matrix, the addr will be zero.
 - ◆ **row** is the row where the LED is located
 - ◆ **col** is the column where the LED is located
 - ◆ **state**

- It's true or 1 if you want to turn the led on
- It's false or 0 if you want to switch it off

➔ **setRow(addr, row, value)**

➔ **setCol(addr, column, value)**

As previously stated, this matrix has 8 columns and 8 rows. Each one is indexed from 0 to 7. Here's a figure for better understanding:



If you want to display something in the matrix, you just need to know if in a determined row or column, the LEDs that are on or off. For example, if you want to display a happy face, you can follow the above figure.

4.3.4-Sketches of DotMatrixDisplay Modules

Sketch: You can explore the sketches under the “4_3_DotMatrix_Display” folder in the GitHub Repository that I have mentioned in the introduction part of the SECTION 4.

SECTION 5: INTERFACING OF ACTUATORS WITH ARDUINO UNO

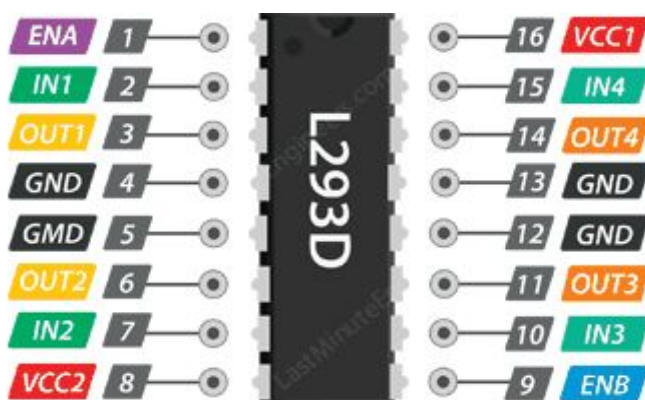
In the strict meaning, actuator is a device that converts energy in movement. It could be a valve or a motor. But I prefer not to be so strict, I consider an actuator anything that can convert electric energy in an output, for example a display, a led, a loudspeaker. Before you go through this section, clone the GITHUB repository that I have shared in the box below for the Sketches and Datasheets of this section.

GitHub: <https://github.com/joydipdutta001/ArduinoStepByStep>

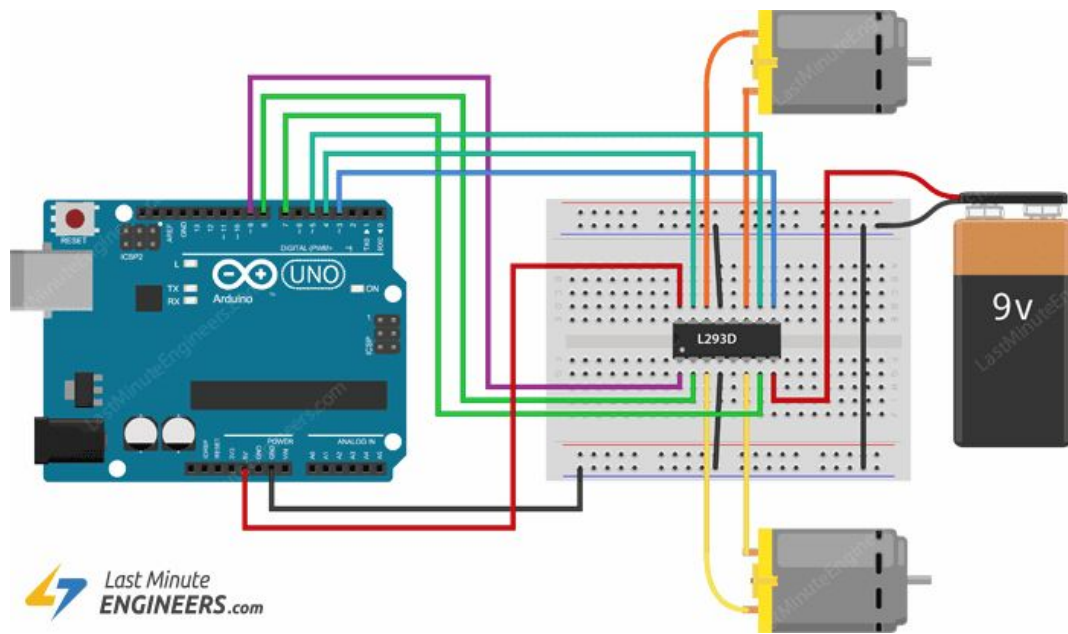
5.1- DC Motors And Arduino Uno:

DC motor converts electrical energy in the form of Direct Current into mechanical energy in the form of rotational motion of the motor shaft. The DC motor speed can be controlled by applying varying DC voltage; whereas the direction of rotation of the motor can be changed by reversing the direction of current through it. For applying varying voltage, we can make use of PWM technique. For reversing the current, we can make use of H-Bridge circuit or motor driver ICs that employ the H-Bridge technique. Here I am using L293D Motor Driver which is used as a H-Bridge technique. The L293D is a dual-channel H-Bridge motor driver capable of driving a pair of DC motors or one stepper motor. That means it can individually drive up to two motors making it ideal for building two-wheel robot platforms.

5.1.1-Pin Description:



5.1.2-Circuit Diagram (Schematics):



5.1.3-How to control a pair of DC motors with L283D Driver ?

Let's start by connecting the power supply to the motors. In our experiment we are using DC Gearbox Motors(also known as 'TT' motors) that are usually found in two-wheel-drive robots. They are rated for 3 to 9V. So, we will connect an external 9V power supply to the Vcc2 pin. Next, we need to supply 5 Volts for the L293D's logic circuitry. Connect Vcc1 pin to 5V output on Arduino. Make sure you common all the grounds in the circuit.

Now, the input and enable pins(ENA, IN1, IN2, IN3, IN4 and ENB) of the L293D IC are connected to six Arduino digital output pins(9, 8, 7, 5, 4 and 3). Note that the Arduino output pins 9 and 3 are both PWM-enabled.

Finally, connect one motor to across OUT1 & OUT2 and the other motor across OUT3 & OUT4. You can interchange your motor's connections, technically, there is no right or wrong way.

When you're done you should have something that looks similar to the illustration shown above figure.

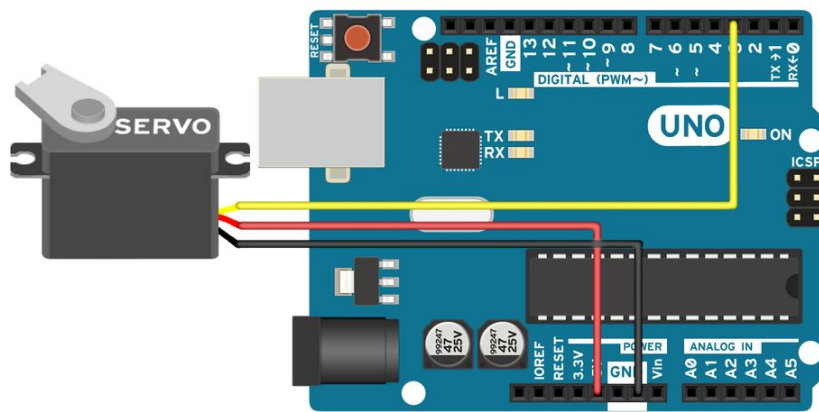
5.1.3-Sketch:

Sketch: You can explore the sketches under the “5_1_DCMotors_Actuator” folder in the GitHub Repository that I have mentioned in the introduction part of the SECTION 5.

5.2- Servo Motors And Arduino Uno:

Servo motors are great devices that can turn to a specified position. Usually, they have a servo arm that can turn 180 degrees. Using the Arduino, we can tell a servo to go to a specified position and it will go there. As simple as that! Servo motors were first used in the Remote Control (RC) world, usually to control the steering of RC cars or the flaps on a RC plane. With time, they found their uses in robotics, automation, and of course, the Arduino world. Here we will see how to connect a servo motor and then how to turn it to different positions.

5.2.1-Circuit Diagram (Schematics):



5.2.2-How to connect a Servo motor with Arduino Uno ?

Following are the steps to connect a servo motor to the Arduino:

1. The servo motor has a female connector with three pins. The darkest or even black one is usually the ground. Connect this to the Arduino GND.
2. Connect the power cable that in all standards should be red to 5V on the Arduino.
3. Connect the remaining line on the servo connector to a digital pin on the Arduino.

5.2.2-Sketch:

Sketch: You can explore the sketches under the “5_2_ServoMotors_Actuator” folder in the GitHub Repository that I have mentioned in the introduction part of the SECTION 5.

LITERATURE REFERENCES

1. github repository. [<https://github.com/joydipdutta001/ArduinoStepByStep>]
2. [Arduino Step by Step Getting Serious](https://www.udemy.com/course/arduino-sbs-getting-serious/) by Dr. Peter Dalmaris on Udemy.
[<https://www.udemy.com/course/arduino-sbs-getting-serious/>]