(An Autonomous Institute of Government of Maharashtra)

Government College of Engineering, Aurangabad

Robotics and Automation Innovation Society

"Arduino workshop for Beginners"

Workshop Handout

1] What is Arduino ?

Arduino is an open source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world. Arduino boards are available commercially in preassembled form, or as do-it-yourself kits.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (*shields*) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++.

The Arduino project started in 2003 as a program for students at the Interaction Design Institute Ivrea in Ivrea, Italy, aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

The name *Arduino* comes from a bar in Ivrea, Italy, where some of the founders of the project used to meet. The bar was named after Arduin of Ivrea, who was the margrave of the March of Ivrea and King of Italy from 1002 to 1014.

2] Different Arduino Boards

The original Arduino hardware was produced by the Italian company Smart Projects. Some Arduino-branded boards have been designed by the American companies SparkFun Electronics and Adafruit Industries. As of 2016, 17 versions of the Arduino hardware have been commercially produced.

3] Arduino Shields

Arduino and Arduino-compatible boards use printed circuit expansion boards called *shields*, which plug into the normally supplied Arduino pin headers. Shields can provide motor controls for 3D printing and other applications, Global Positioning System (GPS), Ethernet, liquid crystal display (LCD), or breadboarding (prototyping). Several shields can also be made do it yourself (DIY).

4] Arduino Uno

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Technical Specifics

- **Power**

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

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

Memory

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

Input and Output

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

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off. The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:
- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library. There are a couple of other pins on the board:
- AREF. Reference voltage for the analog inputs. Used with analogReference().

- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; For SPI communication, use the SPI library.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol.

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar.

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

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader).

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line.

USB Overcurrent Protection

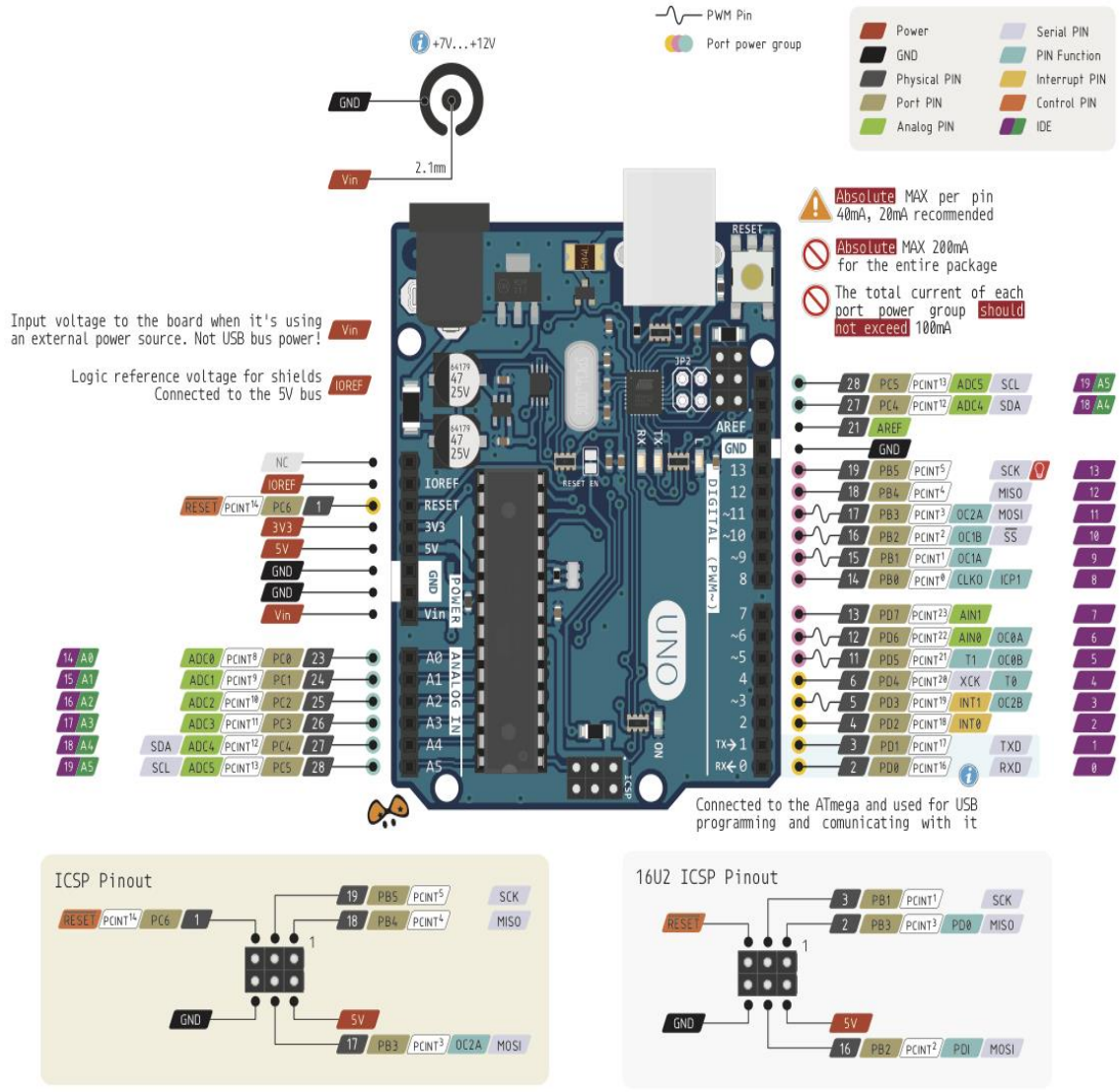
The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than

500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

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

5] Arduino Uno Pin Description



6] Basic Electronics

Voltage, **electric potential difference**, **electric pressure** or **electric tension** is the difference in electric potential energy between two points per unit electric charge. The voltage between two points is equal to the work done per unit of charge against a static electric field to move the test charge between two points. This is measured in units of *volts* (a joule per coulomb)

An **electric current** is a flow of electric charge. In electric circuits this charge is often carried by moving electrons in a wire. It can also be carried by ions in an electrolyte, or by both ions and electrons such as in an ionised gas (plasma).

The SI unit for measuring an electric current is the ampere, which is the flow of electric charge across a surface at the rate of one coulomb per second. Electric current is measured using a device called an ammeter.

The **electrical resistance** of an electrical conductor is a measure of the difficulty to pass an electric current through that conductor. The inverse quantity is **electrical conductance**, and is the ease with which an electric current passes. Electrical resistance shares some conceptual parallels with the notion of mechanical friction. The SI unit of electrical resistance is the ohm (Ω), while electrical conductance is measured in siemens (S).

7] Ohm's Law

Ohm's law states that the current through a conductor between two points is directly proportional to the voltage across the two points. Introducing the constant of proportionality, the resistance, one arrives at the usual mathematical equation that describes this relationship:

$$V \propto I$$
$$V = I \times R \quad \text{(R is proportionality constant)}$$

where *I* is the current through the conductor in units of amperes, *V* is the voltage measured *across* the conductor in units of volts, and *R* is the resistance of the conductor in units of ohms. More specifically, Ohm's law states that the *R* in this relation is constant, independent of the current.

8] Kirchoff’s Law

Kirchhoff's circuit laws are two equalities that deal with the current and potential difference (commonly known as voltage) in the lumped element model of electrical circuits. They were first described in 1845 by German physicist Gustav Kirchhoff. This generalized the work of Georg Ohm and preceded the work of Maxwell.

Kirchhoff’s Current Law

This law is also called Kirchhoff's first law, Kirchhoff's point rule, or Kirchhoff's junction rule (or nodal rule).

The principle of conservation of electric charge implies that:

- At any node (junction) in an electrical circuit, the sum of currents flowing into that node is equal to the sum of currents flowing out of that node or equivalently

- The algebraic sum of currents in a network of conductors meeting at a point is zero.

Kirchhoff’s Voltage Law

This law is also called Kirchhoff's second law, Kirchhoff's loop (or mesh) rule, and Kirchhoff's second rule.

The principle of conservation of energy implies that

- The directed sum of the electrical potential differences (voltage) around any closed network is zero, or:

- More simply, the sum of the emfs in any closed loop is equivalent to the sum of the potential drops in that loop, or:

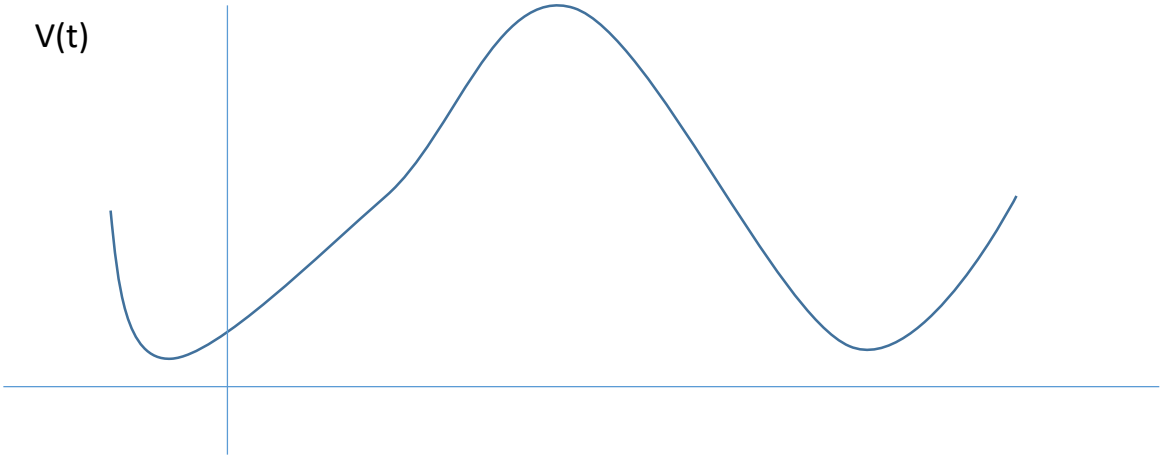
- The algebraic sum of the products of the resistances of the conductors and the currents in them in a closed loop is equal to the total emf available in that loop.

9] Types of Communication

1. **Analog Communcation**
2. **Digital Communication**

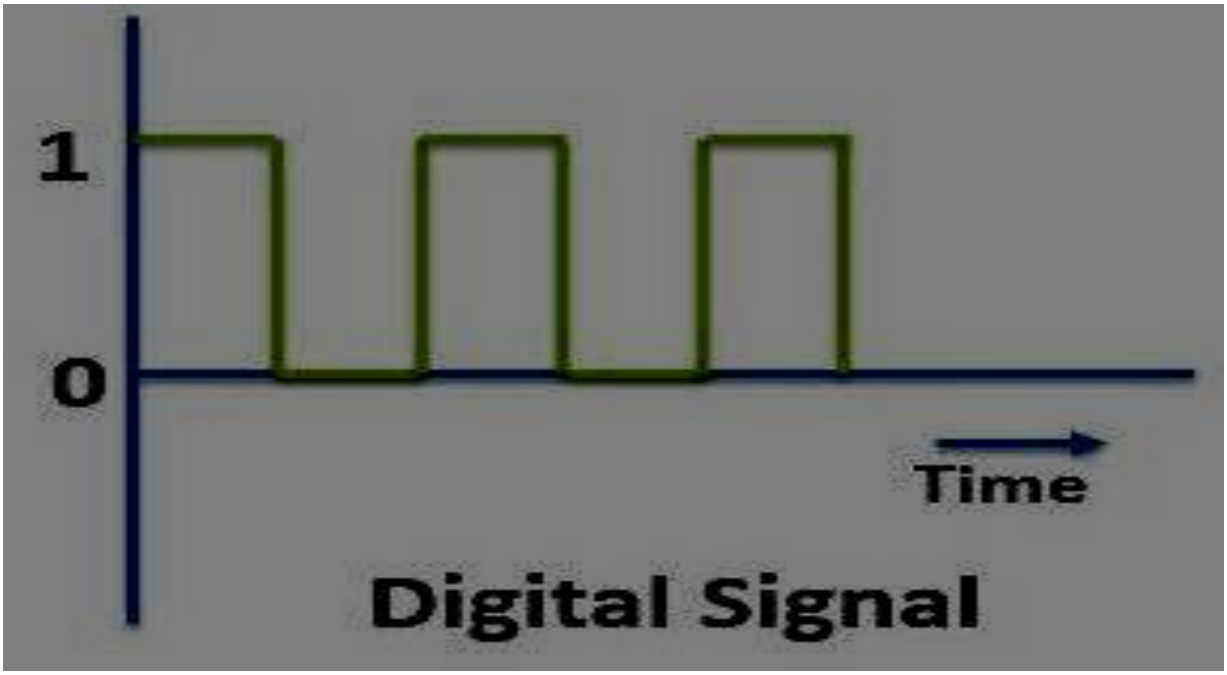
Analog System

- The system in which signals take any value from a given range, and each unique signal value represents different information.



Digital System

The system in which signal is constructed from a discrete set of values.



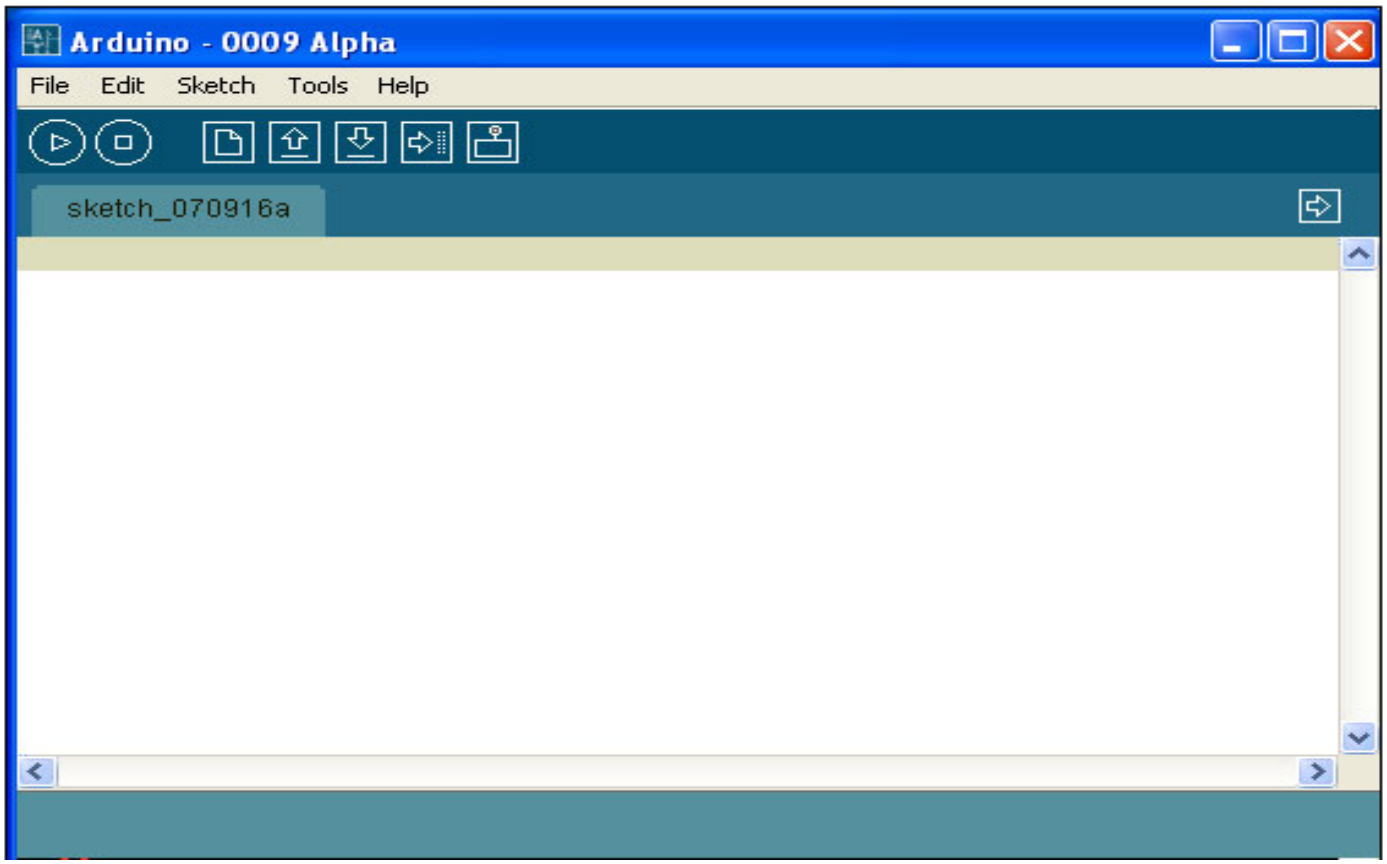
10] Arduino IDE

- The Arduino integrated development environment (IDE) is a cross-platform application written in Java, and derives from the IDE for the Processing programming language and the Wiring projects

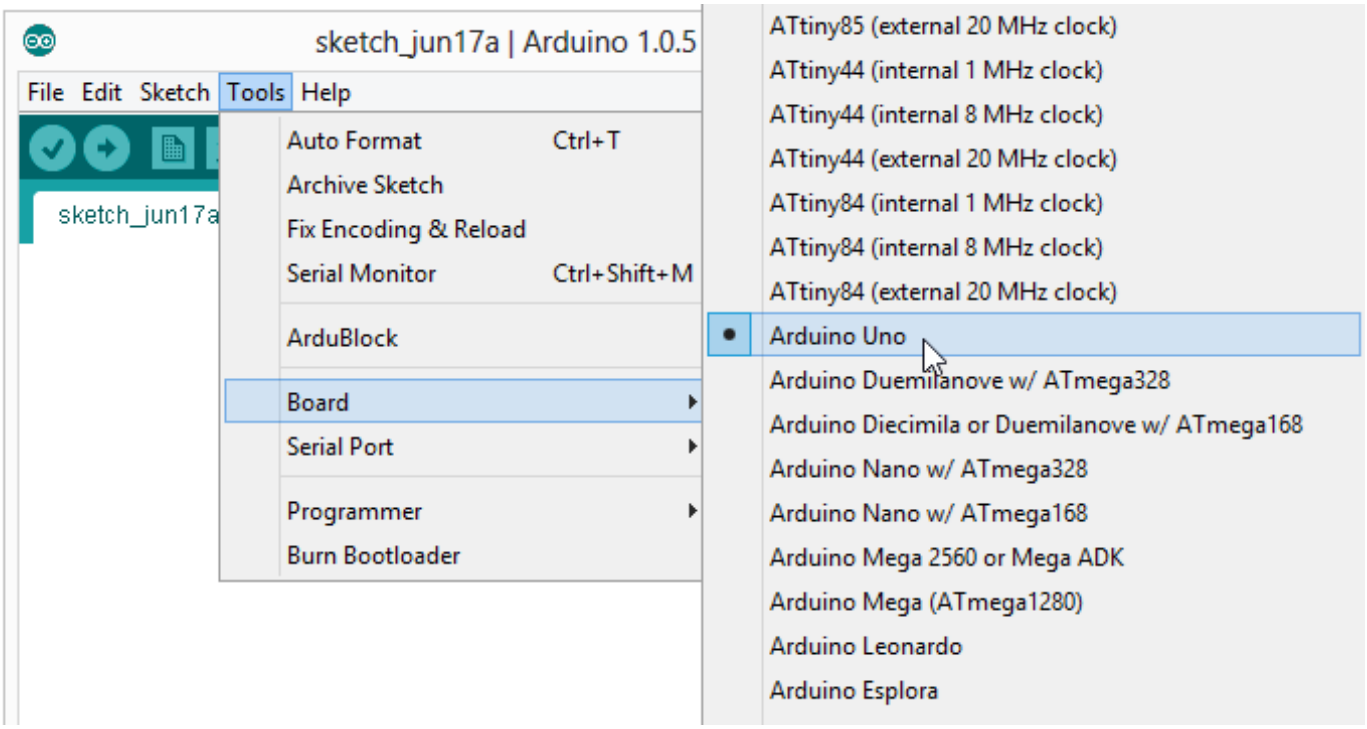
- It includes a code editor which is capable of compiling and uploading programs to the board with a single click.

- A program or code written for Arduino is called a "sketch“

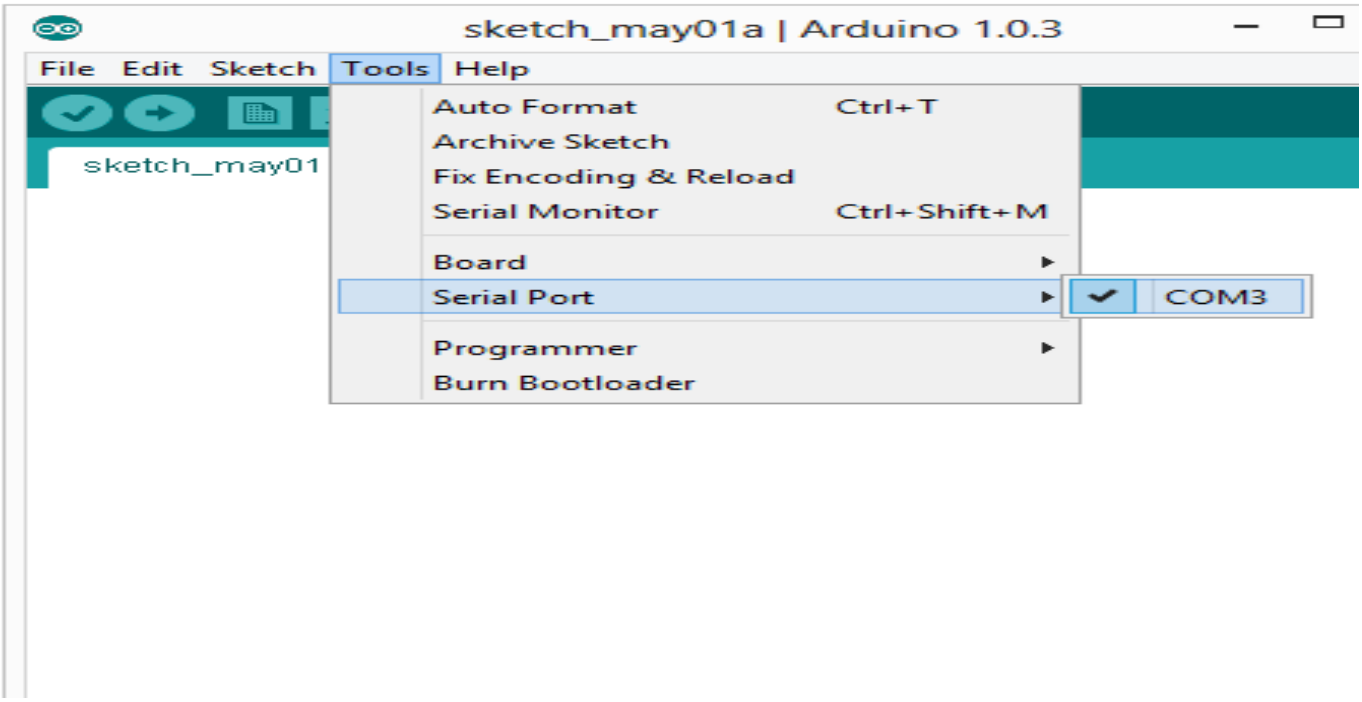
- Arduino programs are written in C or C++



Settings: Tools → Board



Settings: Tools → Serial Port



11] setup() and loop()

Two required functions/ methods / routines:

```
void setup(){
        // runs once
}

void loop(){
        // repeats
}
```

12] Menu items & the icons

File: Contains options to save, load, and print sketches; a thorough set of example sketches to open; as well as the Preferences submenu

Edit: Contains the usual copy, paste, and search functions common to any word processor

Sketch: Contains the function to verify your sketch before uploading to a board, and some sketch folder and import options

Tools: Contains a variety of functions as well as the commands to select the Arduino board type and USB port

Help: Contains links to various topics of interest and the version of the IDE

- ▶ **Verify :** Click this to check that the Arduino sketch is valid and doesn't contain any programming mistakes.
- ▶ **Upload :** Click this to verify and then upload your sketch to the Arduino board.

- **New :** Click this to open a new blank sketch in a new window.
- **Open :** Click this to open a saved sketch.
- **Save :** Click this to save the open sketch. If the sketch doesn't have a name, you will be prompted to create one.
- **Serial Monitor :** Click this to open a new window for use in sending and receiving data between your Arduino and the IDE.

13] Programming Basics

Comments

It is a programmer-readable explanation or annotation in the source code of a computer program

It is not processed by the Compiler

- Single line comment : example :- // a formal statement
- Multi-line comment : example :- /* a formal statement */

Constants

A quantity or parameter that does not change its value

- HIGH and LOW
- INPUT and OUTPUT

Curly Braces

Curly braces defines the beginning and ending of function blocks and statement blocks such as void loop() and for and if.

```
type function name()
{
  statements;
 }
```

Semicolon

A semicolon must be used to end a statements and to separate the elements of the program.

Example:- int a;

Datatypes and Variables

It is a kind of data item, that defines what values it can store and the operations that can be performed on it.

Arduino Data Types	Value Assigned	Value Ranges
boolean	8 Bit	True or False
byte	8 Bit	0 to 255
char	8 Bit	-127 to 128
unsigned char	8 Bit	0 to 255
word	16 Bit	0 to 65535
unsigned int	16 Bit	0 to 65535
int	16 Bit	-32768 to 32767
long	32 Bit	-2,147,483,648 to 2,147,483,647
float	32 Bit	-3.4028235E38 to 3.4028235E38

Variable declaration

- All variables have to be declared before they can be used.
- Declaring a variable means defining it's type as int, float, long etc.
- This only needs to be done once in the program but the value can be changed at any time using arithmetic or various assignment operators.
- A variable can be declared at the beginning of the program before void setup, locally inside the function, and sometimes within a statement block such as for loops.
- Example :- int led=13;

Comparison operators

Comparison of one variable or constant with another are often used to test whether given condition is true.

- X==Y // X is equal to Y
- X!=Y // X is not equal to Y
- X<Y // X is less than Y
- X>Y // X is greater than Y
- X<=Y // X is less than or equal to Y
- X>=Y // X is greater than or equal to Y

Logical operators

Logical operators are way to compare two expressions and return a TRUE or FALSE depending on the operator.

- Logical AND

  if(x>0 && x<5) // TRUE only if both expressions are true

- Logical OR

  if(x>0 || y>0) // TRUE only if either expression is true

- Logical NOT

  if(!x>0) // TRUE only if expression is false

Functions

A function is a block of code that has a name and a block of statements that are executed when function is called.

Custom functions can be used to perform repetitive task.

Functions are declared by first declaring function type like int, void.

```
Type function name( parameters) {
        statements;
}
```

14] Basic functions in Arduino

- **delay(milliseconds)** : pauses the program for the given amount of time.

  1 second = 1000 millisecond

- **pinMode(pin,INPUT / OUTPUT)** : Configures the specified pin to behave either as an input or an output
- **digitalWrite(pin,HIGH / LOW)** : writes a HIGH or LOW value to digital pin
- **digitalRead(pin)** : Reads the value from a specified digital pin, either HIGH or LOW
- **analogWrite(pin,value)** : Writes an analog value to a pin
- **analogRead(pin)** : Reads the value from the specified analog pin

  The value ranges from 0 – 1023

15] C Programming – Flow Control Statements

if()

- If statement tests whether certain condition has been reached or not.
- If the condition contained in the brackets() is TRUE the statements in side curly braces executes.
- If FALSE then the statements are skipped.

```
if ( condition ){
        // dosomething;
}
```

if..else…

If….. Else….. Is used where we have " either… or….." decisions to be made.

```
if( condition){
        do thing A;
}
else {
        dothing B;
}
```

16] C Programming – looping statements

while

- While loop loops continuously, infinitely until condition in the parenthesis or brackets becomes false.
- Something must change the tested variable inside parenthesis, or the while loop will never exit.

```
while(somevariable < 200) // tests condition whether less
                          // than 200 or not
{
    dosomething;          // executes statement
    somevariable++;       // increments variable by 1
}
```

for

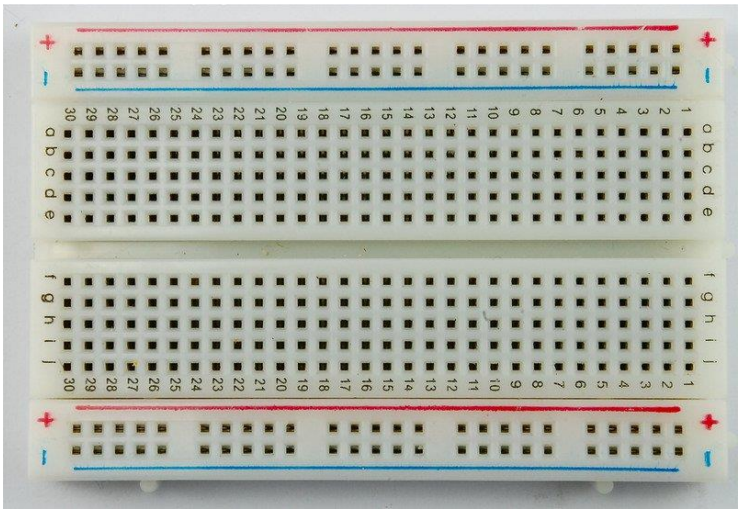
- The for statement is used to repeat the block of statements enclosed in a curly braces a specified number of times.
- There are three parts in the for loop separated by semicolons(;)

```
for(initialization; condition; expression)
{
        dosomething;
}
```

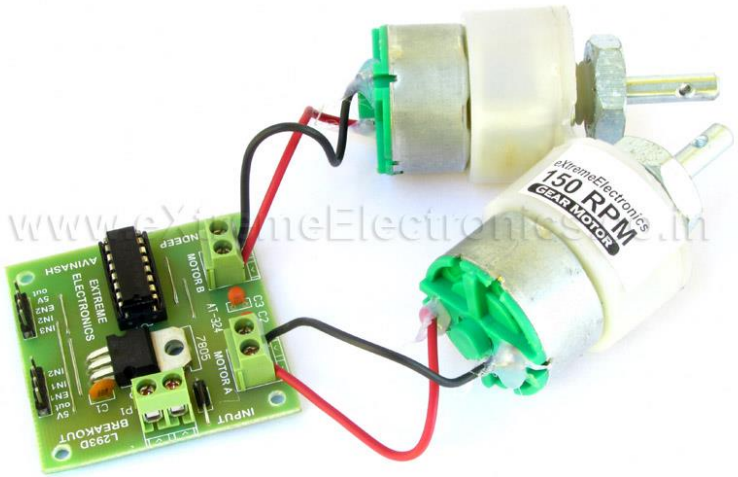
17] Breadboard

A breadboard is a construction base for prototyping of electronics. Originally it was literally a bread board, a polished piece of wood used for slicing bread. In the 1970s the solderless breadboard (AKA plugboard, a terminal array board) became available and nowadays the term "breadboard" is commonly used to refer to these.

Because the solderless breadboard does not require soldering, it is reusable. This makes it easy to use for creating temporary prototypes and experimenting with circuit design.



18] Need for Motor Driver



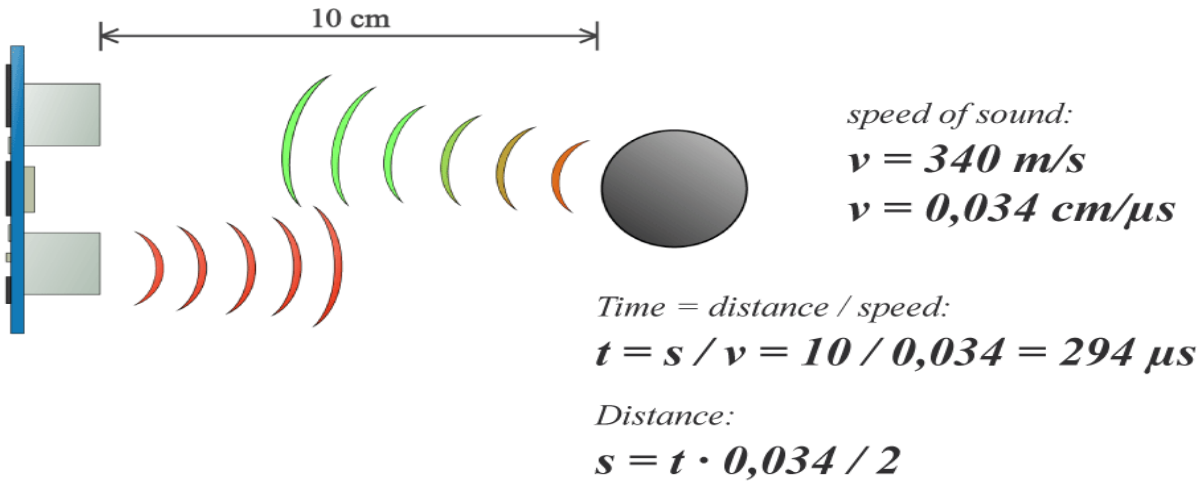
- Arduino provides 5V output
- But we need more voltage(power) to run the motors i.e. Arduino power is not sufficient for motor.
- Motor Driver acts as bridge in between this two power differences.

19] Ultrasonic Sensor

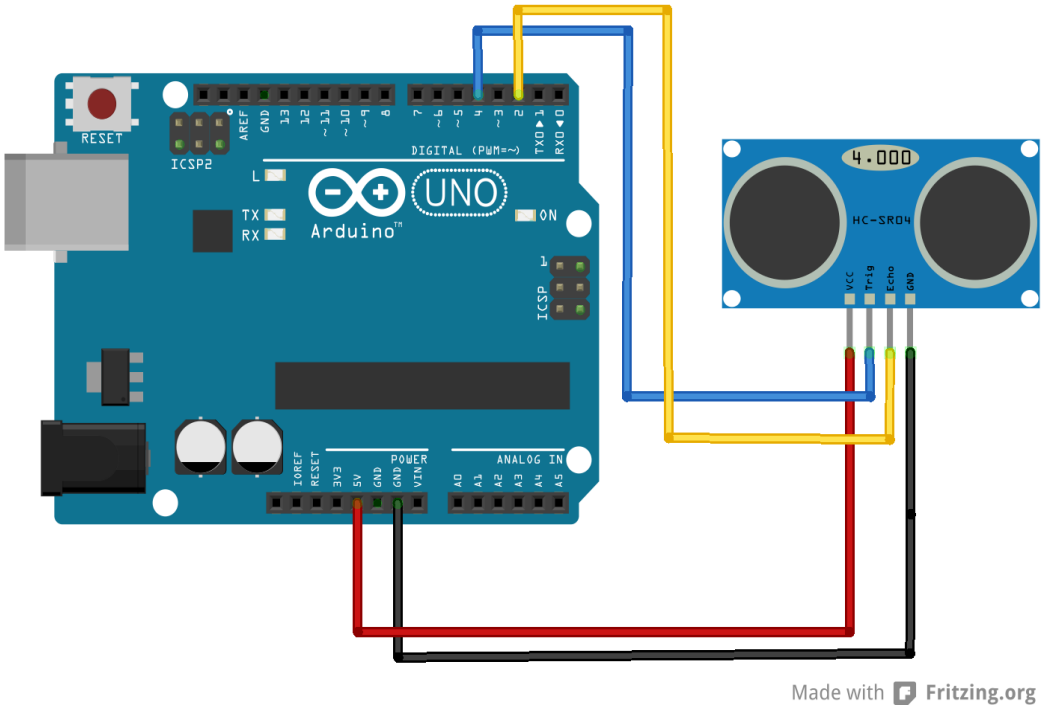
An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object.

Ultrasonic Sensor Working

1. Using IO trigger for at least 10us high level signal
2. The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back
3. If the signal is back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
4. Test distance = (high level time × velocity of sound (340M/S) / 2



Circuit Connections



20] Get Ultrasonic sensor value via Arduino

pulseIn

- Reads a pulse (either HIGH or LOW) on a pin.
- For example, if value is HIGH, pulseIn() waits for the pin to go HIGH, starts timing, then waits for the pin to go LOW and stops timing. Returns the length of the pulse in microseconds. Gives up and returns 0 if no pulse starts within a specified time out.
- pulseIn(pin, value)
- pin: the number of the pin on which you want to read the pulse.
- value: type of pulse to read: either HIGH or LOW

Algorithm

1. Apply 5v to trigger pin for 10 microseconds
2. Apply 0v to trigger pin for 2 microseconds.
3. Measure time period of echo pulse.
4. Distance is given by formula , D=time*340/(2*10000)
5. Observe in Serial monitor.

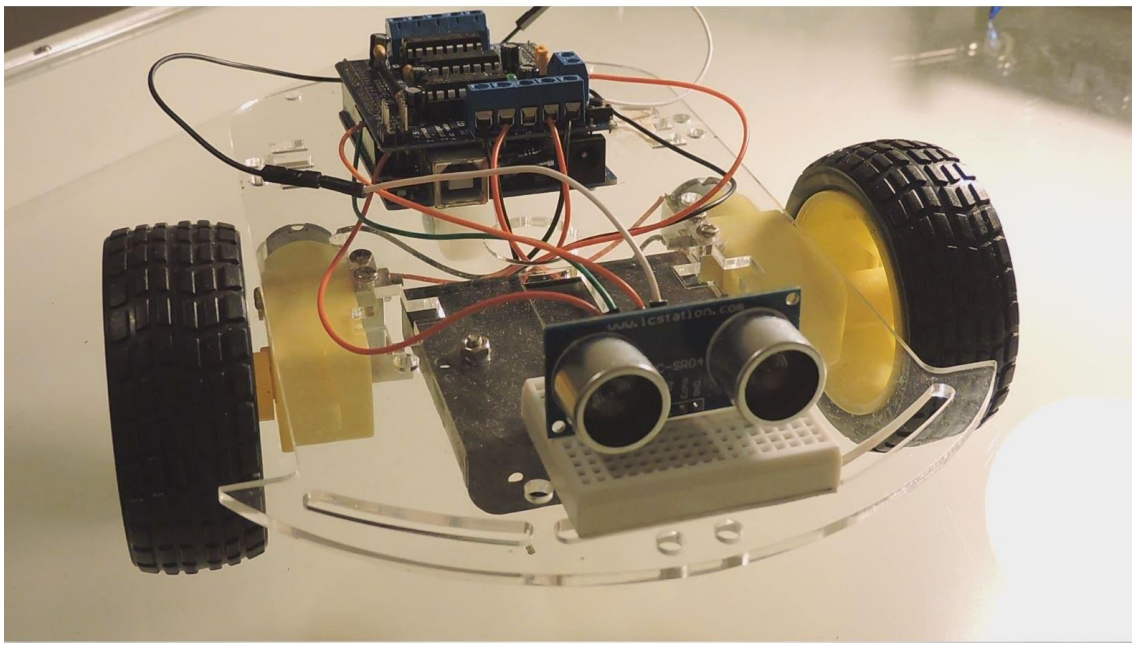
21] Obstacle Avoider Robot

- An Obstacle Avoiding Robot is a type of autonomous mobile robot that avoids collision with unexpected obstacles.

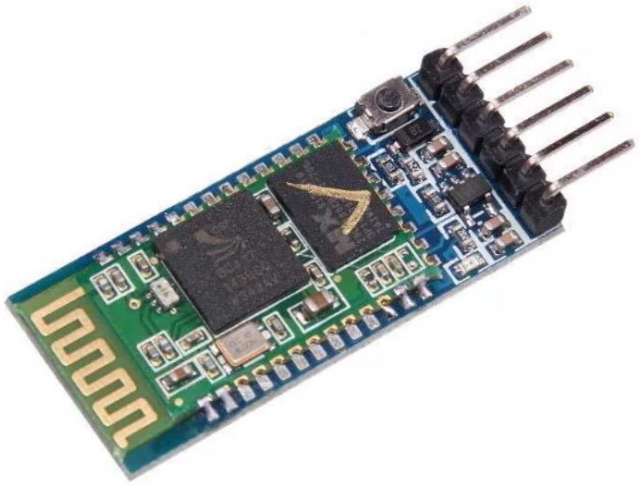
- When the robot is powered on, both the motors of the robot will run normally and the robot moves forward. During this time, the ultrasonic sensor continuously calculate the distance between the robot and the reflective surface.

- This information is processed by the Arduino. If the distance between the robot and the obstacle is less than 15cm, the left wheel motor is reversed in direction and the right wheel motor is operated normally.

- This will rotate the robot towards right. This rotation continues until the distance between the robot and any obstacle is greater than 15cm. The process continues forever and the robot keeps on moving without hitting any obstacle



22] Bluetooth module – HC 05



The HC-05 Bluetooth Module has 6pins.

They are as follows:

- ENABLE: When enable is pulled LOW, the module is disabled which means the module will not turn on and it fails to communicate . When enable is left open or connected to 3.3V, the module is enabled i.e. the module remains on and communication also takes place.
- Vcc: Supply Voltage 3.3V to 5V
- GND: Ground pin
- TXD & RXD: These two pins acts as an UART interface for communication
- STATE: It acts as a status indicator . When the module is not connected to / paired with any other bluetooth device , signal goes Low . At this low state , the led flashes continuously which denotes that the module is not paired with other device . When this module is connected to/paired with any other bluetooth device , the signal goes High . At this high state , the led blinks with a constant delay say for example 2s delay which indicates that the module is paired.
- BUTTON SWITCH: This is used to switch the module into AT command mode . To enable AT command mode , press the button switch for a second . With the help of AT commands , the user can change the parameters of this module but only when the module is

not paired with any other BT device . If the module is connected to any other bluetooth device, it starts to communicate with that device and fails to work in AT command mode.

Default Settings

The default settings for new modules are

Name = HC-05

Password = 1234

Baud rate in communication mode = 9600

Baud rate in AT/Command mode = 38400

Connections For AT mode Configuration

Sr No.	Arduino pin	HC-05 pin
1.	5V	Vcc
2.	GND	GND
3.	2	TXD
4.	3	RXD

Configuration steps

Step1: Upload the above AT mode program to Arduino.

Step 2: Power ON the arduino and press & hold the button on the module.

Step 3: Remove the VCC pin of the module and reconnect it.

Step 4: Release the button on the module

Step5: Notice the led on the module if it blinks once in a second then the module has enter the AT mode if not then repeat the above steps.

AT Commands

Sr no.	AT Commands	Module response	Description
1	AT	OK	If the connection is correct then “OK” is received. Or if ERROR (0) then give the same command.
2	AT+NAME?	< whatever the name of the module>	Using this command the name of the module can be known.
3	AT+NAME=<new name>	Ok----success FAIL----failure	The command is used to change the name of the module.
4	AT+ADDR?	< Bluetooth address>	To get the address of the Bluetooth
5	AT+ROLE=<Parameter>	OK	Parameter: 0--- Slave role 1--- Master role 2--- loop slave role
6	AT+ROLE?	+ROLE:<Parameter>	Give the role of the module.
7	AT+BIND=<address of other module>	OK	The module connects only to module of the given address.
8	AT+BIND?	+BIND:<address>	Gives the address of the module to which it is bind.
8	AT+UART=<Param1>, <param2>,<param3>	OK	Param1: Baud rate Param2: Stop bit Param3: parity bit
10	AT+UART?	+UART:<param1>, <param2>, <param3>	Receives the above parameter of the module
11	AT+RESET	OK	Resets the module.

Configure two Bluetooth modules to establish communication betn Laptops

Set the module in AT mode by the steps mentioned above while the arduino is connected to the computer.

1. Open the serial monitor of arduino
2. Set the baud rate to 38400 of the serial monitor.
3. Send the following AT commands
    a. AT+NAME=(set your name)
    b. AT+ROLE=0
    c. AT+UART=9600,0,0
    d. AT+ADDR?
        1. Note down the address of the module
            i. (e.g. 12: 34: 56: ab: cd: ef)
        2. Share the Address and name with partner Group.
    e. AT+BIND=<address of another partner group>

Now both the modules are configured. The module will connect only to the respective binded address module and vice versa.

To check connect VCC and ground of both the module i.e. give supply to the module and observe the LED of both if the LEDs of both the module blink twice in one second the two modules are successfully paired to each other.