SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF MCA

19CAE725 – Internet of Things

I YEAR II SEM

UNIT4 –Ardunio IDE

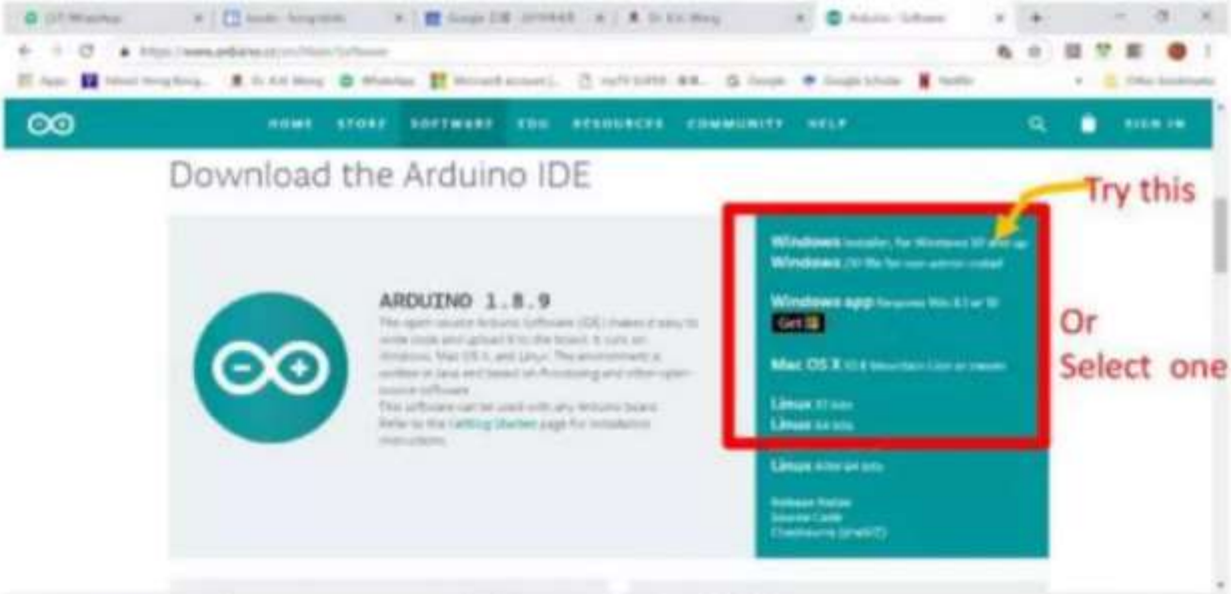
- Arduino, natively, supports a language that we call the Arduino Programming Language, or Arduino Language.
- This language is based upon the Wiring development platform, which in turn is based upon Processing

Introduction to Arduino

- Arduino is a computation tool for sensing and controlling signals
- It is more convenient and cost effective than using a personal computer PC.
- It's an open-source system in terms of hardware and software.
- You can download the Integrated Development Environment (IDE) for your own OS from http://arduino.cc/en/Main/Software
    - Follow the instruction to install the IDE

Installation of the Arduino IDE

- Download Interactive Development Environment DE from https://www.arduino.cc/en/Main/Software



Download the Arduino IDE

ARDUINO 1.8.9

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board. Refer to the Getting Started page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits

Linux ARM 64 bits

Release Notes
Source Code
Checksums (sha512)

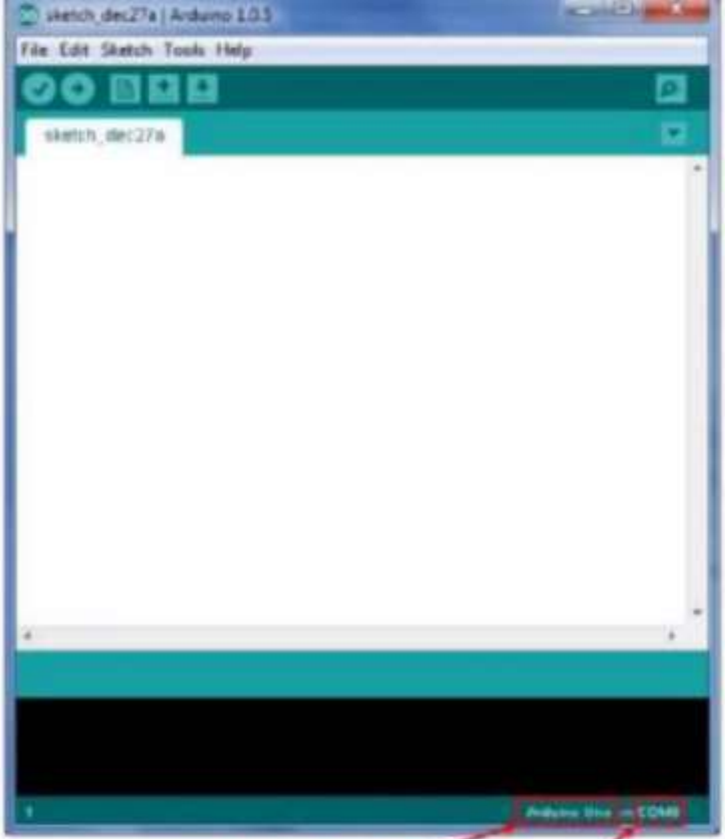
Try this

Or
Select one of these

Start to use the Arduino IDE

- To start Arduino IDE, click *Start Menu → All Programs → Arduino*

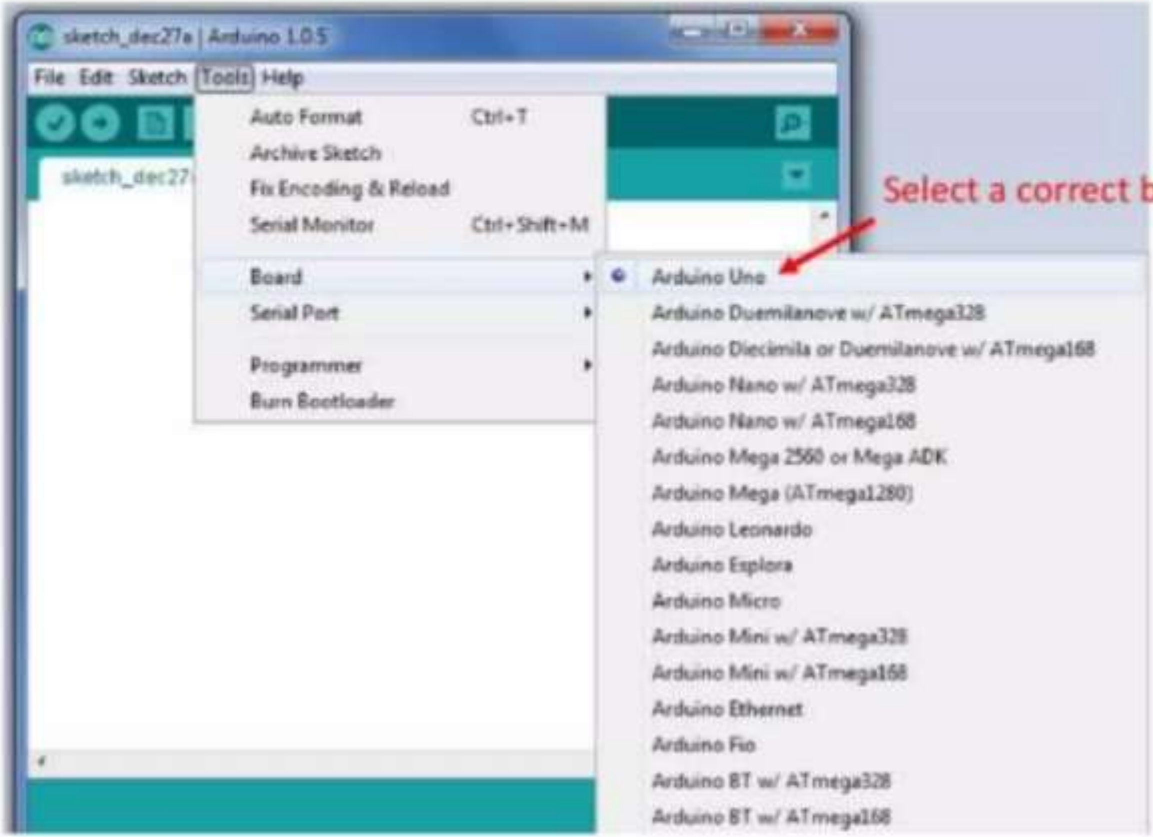
- Make sure the board model (Arduino Uno) and connected port (depends on your PC) are correct



Your board Model

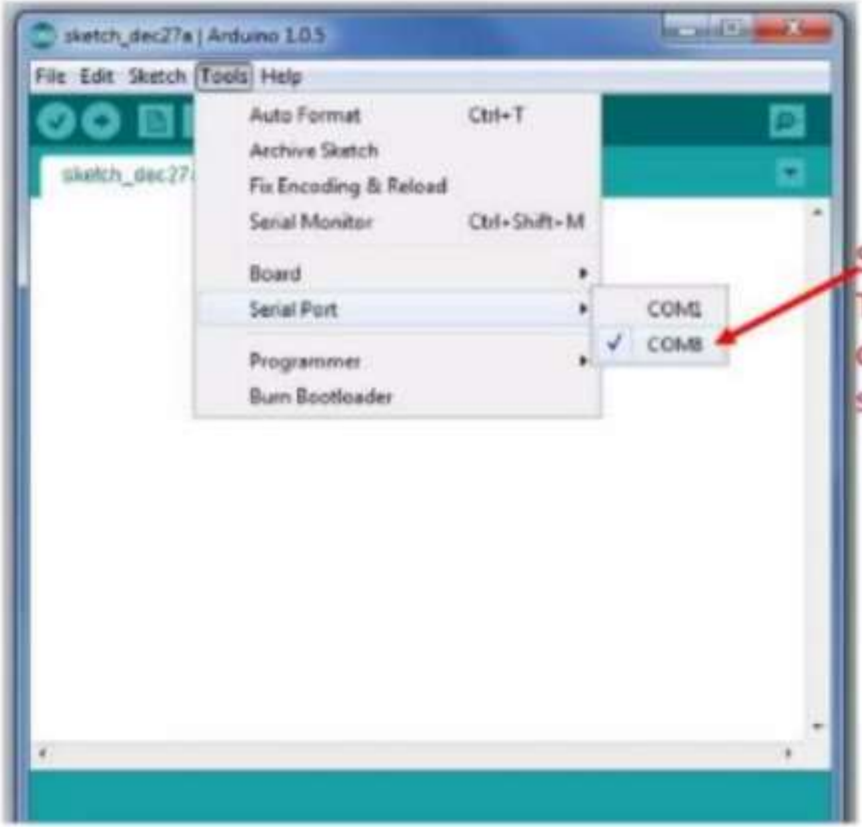
The port that your board connected to

Select Board


sketch_dec27a | Arduino 1.0.5
File
Edit
Sketch
Tools
Help
sketch_dec27a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168

Select a correct board

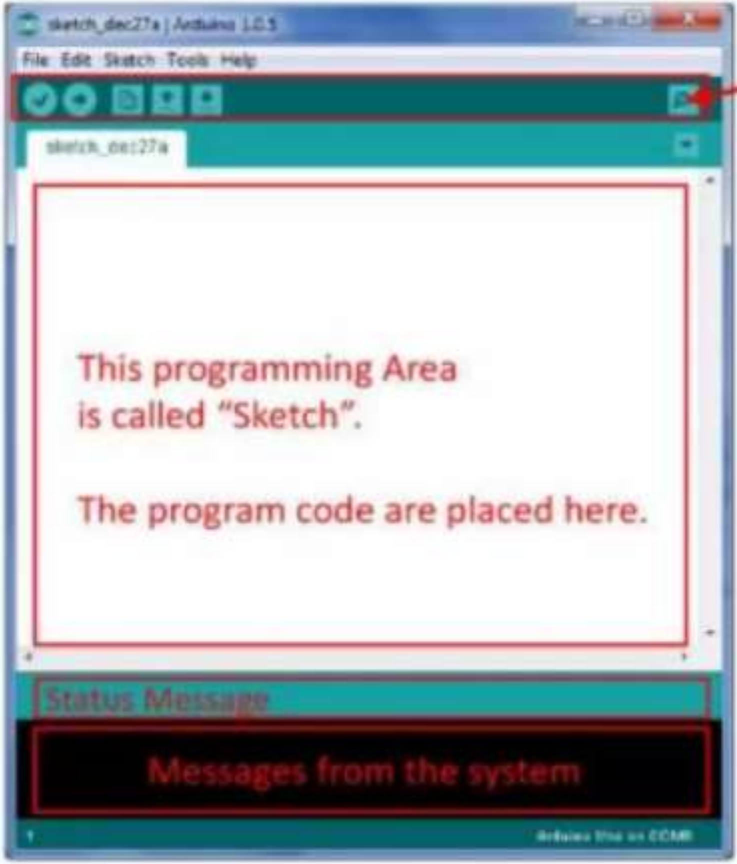
Select Port



Select a correct port. The actual number depends on your system.

Arduino IDE (integrated development environment)

Tool Bar



Serial monitor,
Can use this to issue commands to the board, and read outputs from the board.

This programming Area
is called "Sketch".

The program code are placed here.

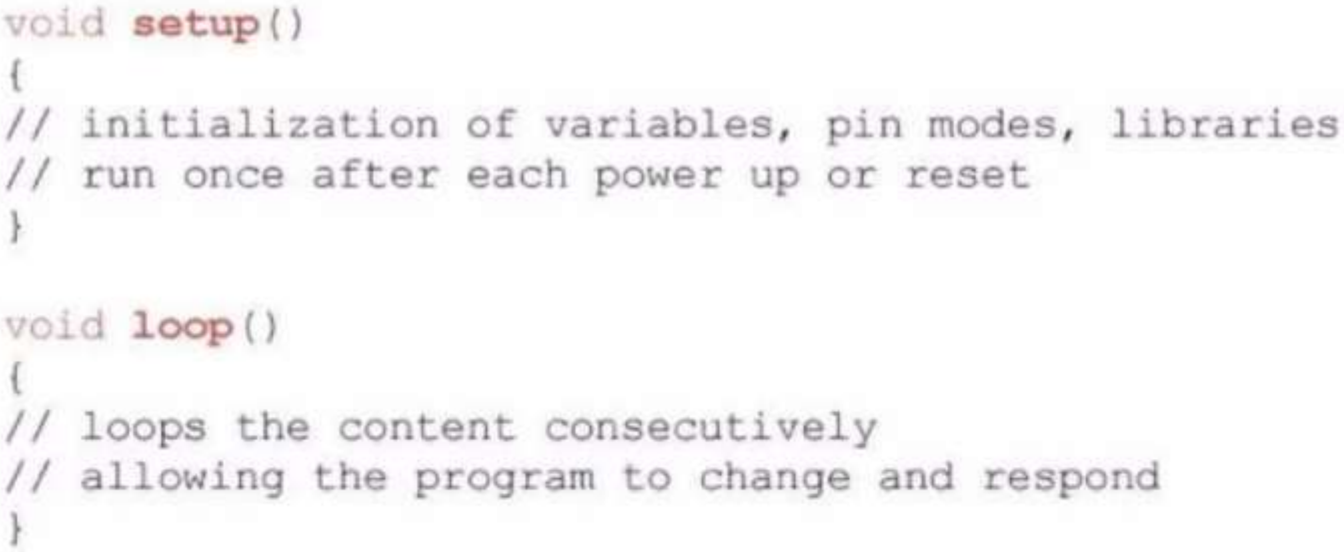
Messages from the system

Toolbar

- Verify
  - Checks code for errors
- Upload
  - Compiles and uploads code to the Arduino I/O board
- New
  - Creates a new sketch
- Open
  - Open sketch
- Save
  - Save sketch
- Serial Monitor
  - Display serial data being sent from the Arduino board

Arduino Code

To run a program in Arduino, your sketch should contain two methods



```
void setup()
{
// initialization of variables, pin modes, libraries
// run once after each power up or reset
}

void loop()
{
// loops the content consecutively
// allowing the program to change and respond
}
```

Basic software functions

- Hardware related
  - pinMode(), setup the functions of hardware pins
  - digitalWrite(), set a pin to a digital level : ‘1’ or ‘0’
  - digitalRead(), read the digital level of a pin: ‘1’ or ‘0’
  - delay()
- Software related
  - If-then-else
  - For
  - Switch-case

System setup procedures

- (Step 1) Setup the direction of the pins:
  - using pinMode(),
- (Step 2) Then you can set a pin to : HIGH or LOW
  - (Step 2a) digitalWrite(), //set pin to : HIGH '1' or LOW '0'
  - or
  - (step 2b) digitalRead(), //read state of pin: HIGH '1' or LOW '0'

Basic Function (step1) – pinMode()

- pinMode() is used to configure the specified pin to behave either as an input or output, or input_pullup
- Syntax

Pin =0,..,13, or A0,A1,..,A5 for Digital I/O, or

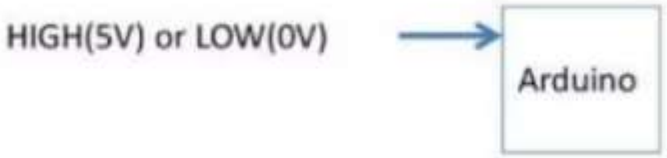
Write comment for you to read

```
pinMode(pin, mode) // comment
```

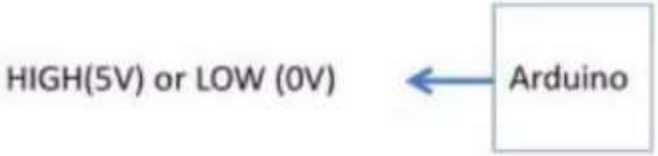
  - *pin*: the index number of the pin whose mode you wish to set
  - *mode*: INPUT, OUTPUT, INPUT_PULLUP
  - Example:
    - `pinMode(1, OUTPUT)//setup pin1 =digital out`
    - `pinMode(3, INPUT)//setup pin3 =digital in`
    - `pinMode(A3, INPUT)//setup A3 for digital in`
    - `pinMode(A3, OUTPUT)//setup A3 for digital out`
    - `If no PinMode applied to A0->A5, they are analog_in by default.`

Meaning of INPUT, OUTPUT, INPUT_PULLUP

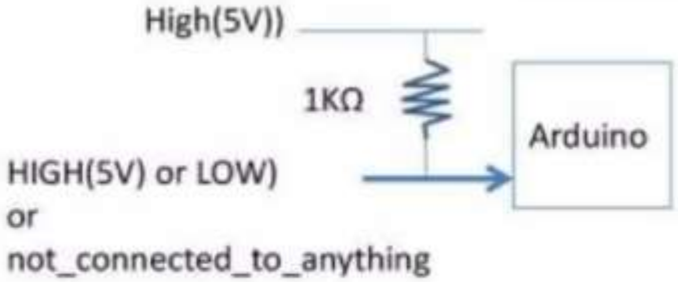
- INPUT:



- OUTPUT:



- INPUT_PULLUP: When the pin is not connect to anything, it is HIGH



Basic Function(step2a) – digitalWrite()

- digitalWrite() is used to write a <u>HIGH</u> or a <u>LOW</u> value to a digital pin

- Syntax `digitalWrite(pin, value) // comment`

  - *pin*: the number of the pin whose value you wish to set
  - *value*: HIGH (5 V) or LOW (Ground)
  - Example:
    - `digitalWrite(pin, value) // comment`
    - `E.g`
    - `digitalWrite(1, HIGH)//set pin1 to HIGH`

Basic Function(step2b) – digitalRead()

- digitalWrite() is used to read the value from a specified digital pin, either <u>HIGH</u> or <u>LOW</u>

```
digitalRead(pin)
```

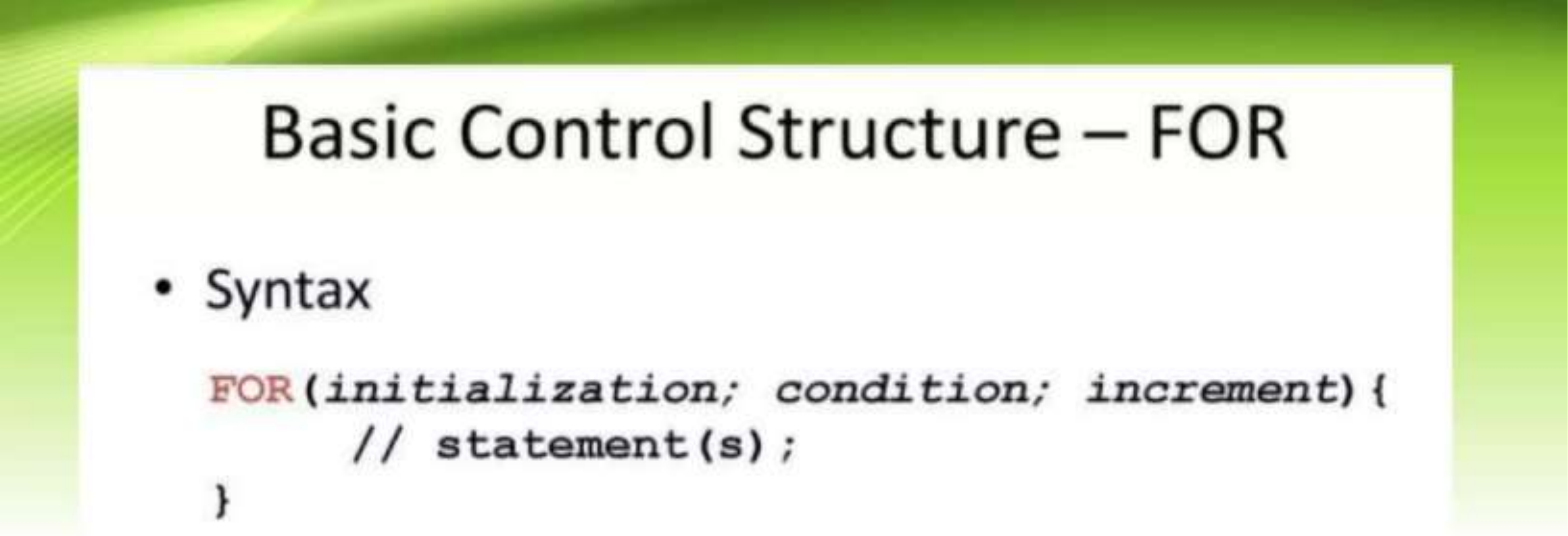
- Syntax
  - *pin*: the number of the pin whose mode you want to read (integer)
  - Example:
    - `digitalRead(pin)// read the state of the`
    - `                  // it can be "HIGH" or "LOW"`

Some other basic Function – delay()

- delay() is used to pause the program for the amount of time (in milliseconds)

```
delay(ms)
```

- Syntax
  - *ms*: the number of milliseconds to pause (unsigned long)



Basic Control Structure – FOR

- Syntax

```
FOR(initialization; condition; increment){
      // statement(s);
}
```

UNIT - II

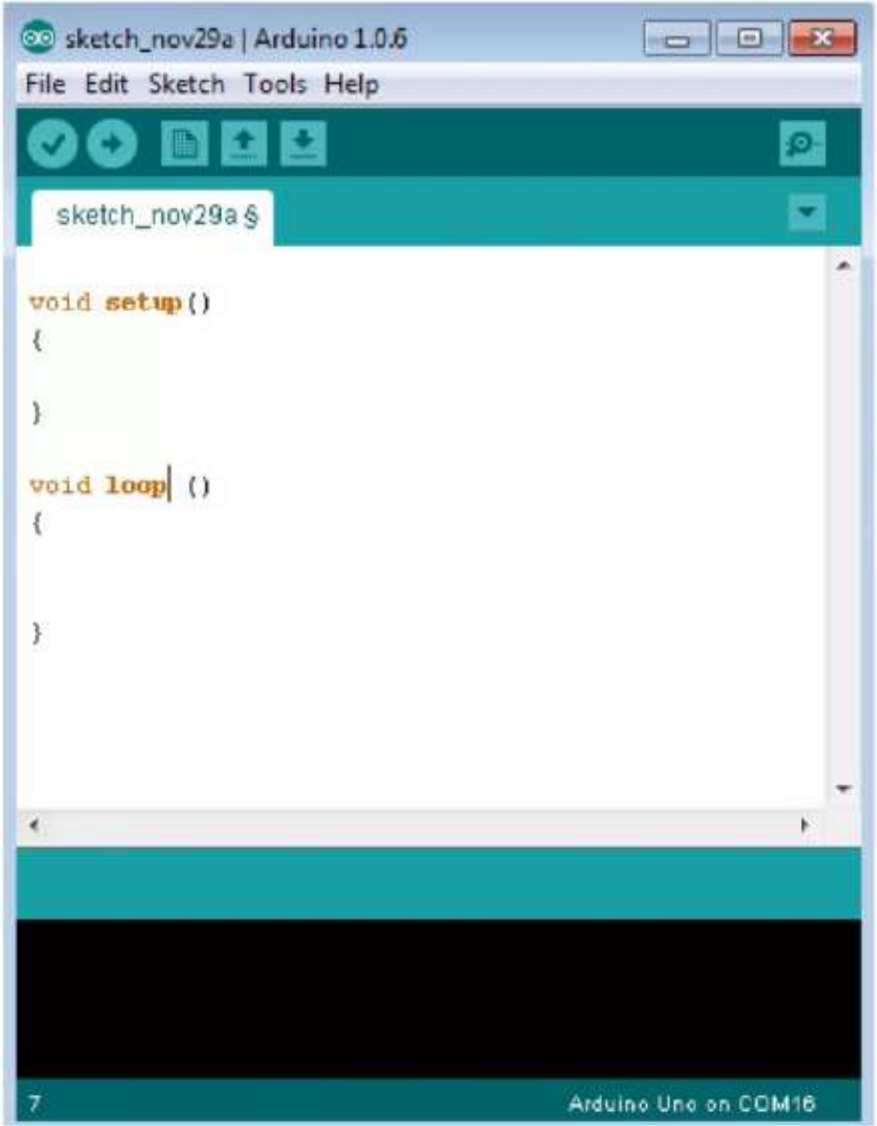
Programming with Arduino

Program Structure

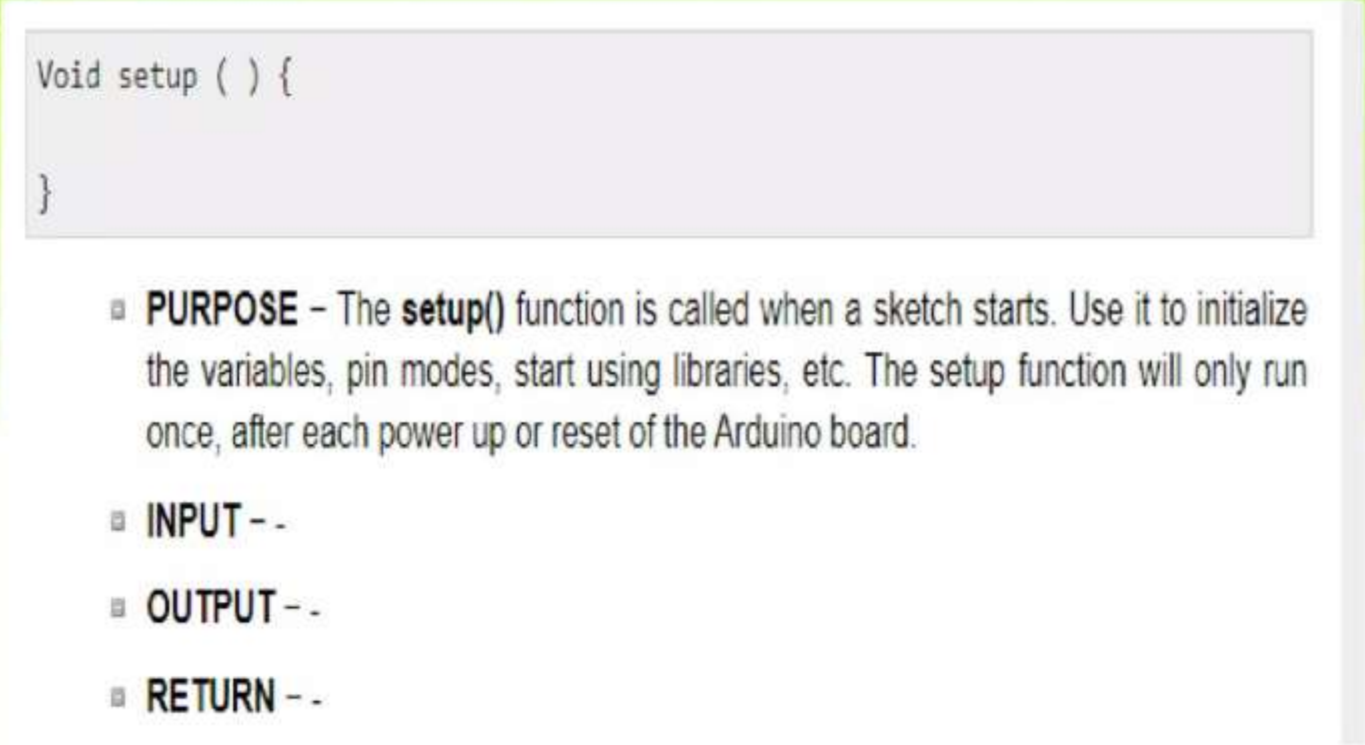
- Structure
- Arduino programs can be divided in three main parts: Structure, Values (variables and constants), and Functions. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

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

- Setup( ) function
- Loop( ) function

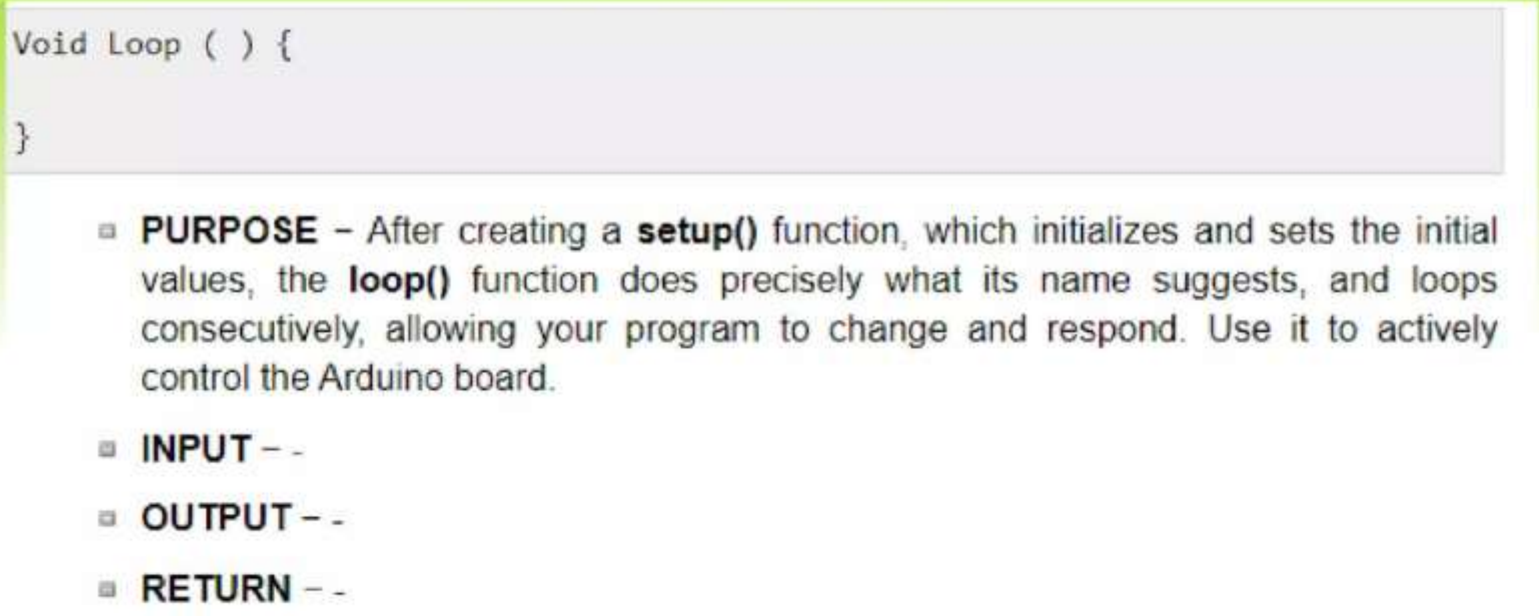


```
void setup()
{


}

void loop ()
{



}
```



```
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** – -