



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 MECHANICAL ENGINEERING 19MEB204 IoT FOR PRODUCTION SYSTEM

TOPIC – Arduino



Arduino

What is Arduino ?

- Open source Electronic Platform



based on



Open Source Software

Closed source software



- Microsoft Office
- Source Code will not be available for everyone
- Available only for the creator and they can only able to modify the code

Open Source Software

- Linux, Arduino IDE
- Source code available to everyone
- Anyone can able to view, copy, learn and modify the code

Open source Hardware

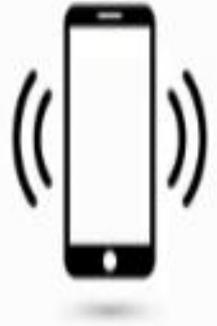


Open Source Hardware

- Schematic will be available to everyone
- Eg: Arduino Boards
NodeMCU

Closed Source Hardware

- Schematic and other details will not be available
- Eg: Mobile Motherboard
Laptop Motherboard



What do Arduino do?

Environment



read data
send command

Arduino

- light intensity
- temperature
- wind flow
- finger on button
- twitter message

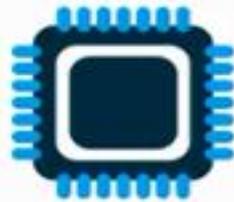


- activating a motor
- turning on a tubelight
- publishing something online



How Arduino do all this?

- each Arduino board has a Microcontroller
- by sending a set of instruction to the microcontroller on the Arduino board

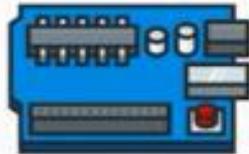


We can do lot of amazing projects





Open Source Hardware



- Circuit board/
Arduino board
- can be programmed
- also referred to as a
Microcontroller or brain



Open Source Software

- ready-made software
- Arduino IDE

Integrated Development Environment

Write
Compile
Upload

- Computer code to
Physical board





Arduino

- The Arduino is an **open-source electronics platform** based on easy-to-use hardware and software used to build electronics projects. All Arduino boards have **one thing in common which is a microcontroller**. A microcontroller is basically a really small computer.
- With the Arduino, you can **design and build devices that can interact with your surroundings**.
- The Arduino boards are basically a tool for controlling electronics. They are able to read inputs with their onboard microcontroller (eg. Light on a sensor, an object near a sensor) and turn it into an output (Drive a motor, ring an alarm, turning on an LED, display information on an LCD).
- With the Arduino, makers and electricians can easily prototype their products and make their ideas come to life.



Difference between Arduino and Raspberry Pi

- Before going into detail about the Arduino, some of you may be confused with the **Arduino and a Single Board Computer (SBC)** that is based on a **microprocessor** like the Raspberry Pi. Let's clear the confusion now by comparing the Arduino and one of the most popular SBC the Raspberry Pi.



What are their main differences?

- An **Arduino** is based on a **microcontroller** which is a simple easy to use computer designed for beginners to run **1 program at a time, over and over again**.
- A **Raspberry Pi** is a Single board computer based on a **microprocessor** that acts as a general-purpose computer. It is able to run **various operating systems like Windows and Linux**. The Raspberry Pi is able to **run multiple programs** and is more complicated to use compared to the Arduino.



How do I pick which one to get?

- If you want a **simple easy-to-use board to handle simple repetitive tasks** like reading the weather, opening a door, driving a simple robot, turning on an LED, etc. an **Arduino would be perfect**.
- However, if you want a fully operating computer that is able to **run more complicated functions** and the ability to run multiple tasks, an SBC like a Raspberry Pi 4 **would be perfect for you**.



Why use the Arduino

- There are many electronic boards out there, why use the Arduino board? Well, there are many reasons that make this microcontroller special.
Advantages of using the Arduino includes:
- Arduino **simplifies microcontrollers for beginners**
- Besides the main microcontroller chip, **a microcontroller will require many different parts for it to work.** What Arduino did is that they took all the essential components of a microcontroller and design it in a way that is **very simple to operate of a piece of PCB.** This makes the Arduino boards welcoming to all beginners!
- Furthermore, with Arduino easy to use IDE software for beginners, the Arduino are easier to learn to program as it uses a simplified version of C++ compared to other programming software. Because of this, the Arduino is commonly cited as the pathway for everyone who is looking to learn about microcontrollers. With it being optimised for users of all levels, even advanced users are taking advantage of the Arduino IDE as well!



- In addition, the Arduino community is very big and many users and organizations are all using it. There are wide varieties of tutorials and projects available online that are pre-coded for you to learn and build using the Arduino allowing beginners to get started very easily.



- **Cheap**

- Whenever you are buying something, you will always look at the cost first. The Arduino are very accessible and cost-effective!
- You can get an official complete Arduino UNO Rev3 at only \$24.95 or our very own Seeeduno V4.2 which is an Arduino-compatible board, that is based on ATmega328P MCU (same as Arduino UNO) at \$6.90 only!
- Without breaking your wallet, you can easily get an Arduino for yourself to play with!



- **Cross-Platform**

- Arduino IDE is also **cross-platform** which means you can run it on **Windows, Macintosh OSX and also Linux operating systems** compared to other microcontroller systems which can only run Windows.



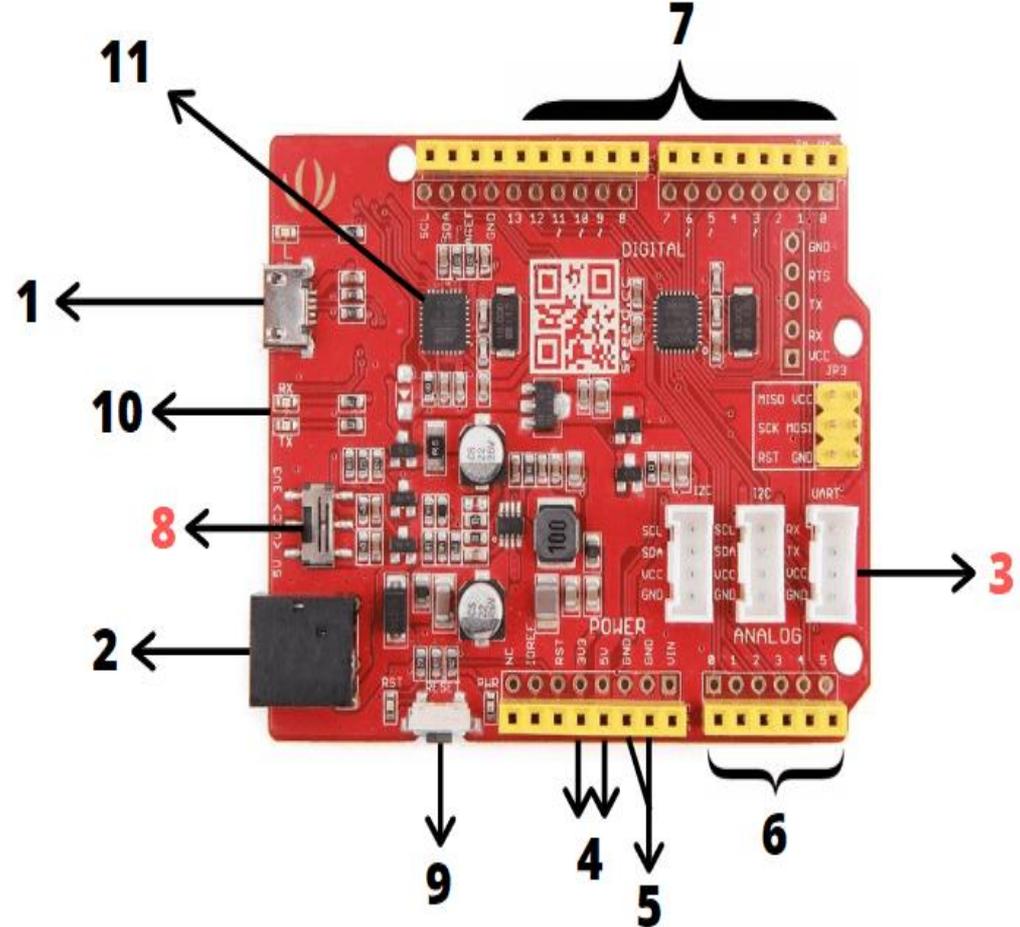
- **Wide Variety**

- The Arduino has many variations for you to choose from to allow you to pick one that suits your project the most!
- Having space constraints? You can get yourself an Arduino Nano that is only 43.18 mm by 18.54 mm! Require more memory space and processing power? You can get yourself an Arduino Mega!
- We will talk more about all the different types of Arduino's and their differences later on!



What makes up an Arduino Board?

- The physical hardware of Arduino is the board itself. However, when it comes to Arduino boards, there are many varieties with different functionalities.
- Today we will be looking at our **Seeeduino V4.2** which has the same functions as one of the most popular Arduino board the **Arduino UNO**. Most Arduino boards will have these various common components that we are going to list:



Different Components in Arduino UNOⁱ

- USB Jack
- Power Jack
- Voltage Regulator
- Crystal Oscillator
- Reset Button
- Power Pins
- Analog Input Pins
- Digital Pins
- TX and RX Pins
- Power On LED
- LED attached to digital pin 13
- AREF
- ICSP Pins
- Main Microcontroller



1 – USB Input

USB Port is used to **connect the board to your PC** for **programming and for powering up** the Arduino board.

This USB connection is important as it will be through this port where you will **upload your code onto your Arduino board.**

USB Jack

- Used to upload programs from computer to Arduino board
- Also used to power up the Arduino board





2 – DC Input

The DC power jack allows your Arduino board to be **powered from a wall adapter** so that you can supply more power to your project if needed.

Power Jack

- Used to Power Arduino
- Input Voltage Range
7V to 12V

Ways of Connection

- Rechargeable Batteries
- Disposable Batteries
- Wall - adapters
- Solar Panel





- **3 – Grove Connectors**
- These Grove connectors can **only be found on our Seeeduino boards.**
- SeeedStudio has a variety of sensors/devices that can make use of this I2C or UART connection.
- With our Grove Connectors, **you can easily plug-in modules** to use with the Arduino without any soldering or jumper systems.

Voltage Regulator



- Used to control & stabilize the voltage given to the arduino board
- Used by the Processor and other elements



Crystal Oscillator



- Helps Arduino in dealing with timing requirement
- Used to calculate time
- Frequency - **16MHz**





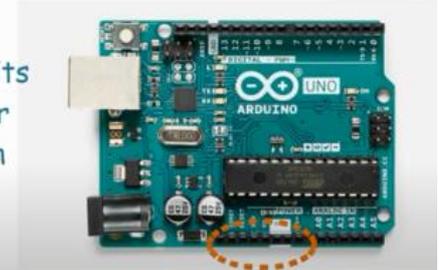
4 – 3.3V and 5V Pins

As the name specifies, the 3.3V and 5V pins supply volts of power to your modules.

The 3.3V pin supplies 3.3 volts of power while the 5V pin supplies 5 volts of power.

Power Pins

- 3.3V - supply 3.3 output volt
- 5V - supply 5 output volt
- GND - used to ground our circuits
- Vin - can also be used to power the Arduino board from an external power source





- **5 – GND pins**
- With this GND (Ground) pin, they are used to ground your circuit.
- GND means this **pin is at zero voltage** with respect to the power supply and ground plane of the circuit board



6 – Analog Pins

The analog pins allows the Arduino to **read signals from an analog sensor** like a light sensor and **convert it into a digital value**.

Even though the main function of the analog pins for most Arduino users is to read analog sensors, the analog pins also have all the functionality of general-purpose input/output (GPIO) pins.

Analog Input Pins

- has six analog input pins (A0 - A5)
- used to read the signal from an analog sensor



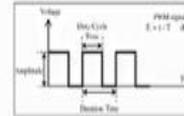


7 – Digital Pins

On the Seeeduno or Arduino UNO, the digital pins are on pin 0 to 13. They allow the Arduino to read digital inputs like a button being pushed and digital output like turning on an LED.

Digital Pins

- has 14 digital pins - 0 to 13
- 6 pins provide PWM (Pulse Width Modulation) output
- when set as I/P: these pins can read voltage





- **8 – System Power Switch**

- This system power switch can only be found on our Seeeduino boards.
- This slide switch is used to change the logic level and operating voltage of the board to either 5V or 3.3V which is useful as if you want to save power, you can set it to 3.3V.



9 – Reset button

This reset button allows you to **reset the board and restart any code** uploaded on your Arduino board. Once pressed, the reset pin will be temporarily connected to the ground.

This reset button is very useful for your projects in the event your code **does not repeat itself but you wish to test it multiple times**.

This button is conveniently placed on the side to allow you to reset the Seeeduno board even when a shield is placed on top. This is not the case in other Arduino boards where the button is placed on top making it hard to access.

Reset Button

- Used to Start our program from the beginning
- Reset Pin





10 – RX/TX Indicator

Also known as **Transmit and receive indicator**, the TX and RX LED indicators are connected to TX and RX of the USB-to-UART chip.

They work automatically and lets you know when the board is sending or receiving data respectively like when you are uploading a program onto your Arduino board.

TX and RX Pins

- TX - Transmit | RX - Receive
- Arduino uses these pins to communicate with other electronics via serial communication
- Avoid using these pins for other tasks other than serial communication unless you're running out of pins





- **11 – Microcontroller**

- On the Seeeduno V4.2 and Arduino UNO, they are based on the microcontroller: **ATmega328P**
- This is the primary chip which acts as the **brain of your Arduino board**.
- They allow you to program your Arduino in order for it to be able to execute commands and decisions based on the code.
- You will have to know which type of microcontroller your board is using before loading a new program from the Arduino Software.
- Even though the microcontroller on the Arduino boards is different, their difference is not big. The only difference that you may notice is the different amounts of onboard memory.
- With your Arduino board, you definitely can't do anything much with it. This is where Arduino sensors and shields come in:

Main Microcontroller

- each Arduino board has its own microcontroller - brain of the board
- usually of the ATMEL  **MICROCHIP** company
- we must know what IC our board has before loading up a new program from the Arduino IDE
- ATMEGA 328P



Power ON LED

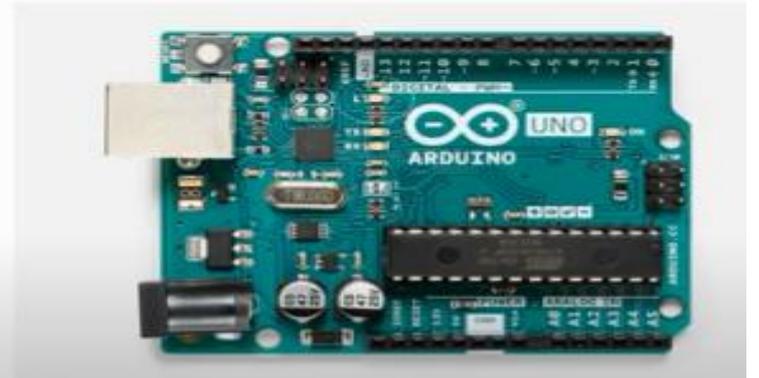
To indicate us Arduino board
is powered by correctly



LED Attached to digital pin 13



useful for an easy debugging of
the Arduino sketches



AREF

- stands for Analog Reference



- it is sometimes, used to set an external reference voltage (between 0 to 5V) as the upper limit for the analog input pins





Arduino Sensors and Shields

Arduino Shields

- Arduino shields are pre-built circuit boards that are easily plugged on top of your Arduino headers to extend its capabilities.
- With the Arduino, adding Bluetooth, wifi connectivity, GPS, motor driver can be difficult if you are new to the Arduino. With shields, you can avoid all the trouble and easily plug in a shield on the back of your Arduino.
- Similar to sensors, shields have various functionalities from wifi connectivity, ethernet, driving and controlling motors, camera, storage, touch screen, E-Ink Display, and many more!
- Interested? You can check out all our Arduino Shields here! Here is an example of one of them.

Arduino Shields





- Grove Base Shield V2.0 for Arduino
- The Grove Base Shield provides a simple way to connect your Arduino Boards to Grove modules.
- With the 16 on-board Grove connectors, you can be assured that you always have enough ports for your different Grove modules. There is also an RST button, and a green LED to indicate its power status. Being an Arduino shield, it allows for plug-and-play connection to the Arduino Uno R3 and several other Arduino boards.



- Arduino Sensors
- With a few lines of code on your Arduino, you can play around and control a wide variety of sensors and build awesome projects. Our sensors can measure light, ultrasonic distance, moisture, temperature, humidity, gas, pressure, motion, sound, touch, and many more! Whatever you can think of, our Grove sensors can sense it! Not to mention all our Arduino sensor modules are compatible with our Grove system, perfect for beginners.





About the Arduino IDE

- After knowing the hardware of Arduino, **you will require software** and programming to make your Arduino come to life and allow it to interact with various sensors and shields.
To program your Arduino, you will require the **Arduino IDE software**.
- Arduino IDE makes it easy for you to write code and upload it on your Arduino board. This program is **cross-platform which means it is able to run on Windows, Mac OS X, and Linux compared to other microcontroller systems which can only run Windows**.
- This software can be used with any Arduino board like our Seeeduino V4.2, Arduino UNO, etc. The environment is written in Java and based on processing and other open-source software.
- This program uses a **simplified version of C++** with syntax highlighting and other features which makes it easier to learn to program which is perfect for beginners to learn programming and coding!
- After you finish writing your code, you can then easily load your code on your Arduino IDE with a USB cable with a click of a button.



Types of Arduino Available





Types of Arduino Available

- Arduino Uno Rev3
- The Arduino Uno is the ideal board for getting started with electronics, through fun and engaging hands-on projects. This board is your entry to the unique Arduino experience: great for learning the basics of how sensors and actuators work, and an essential tool for your rapid prototyping needs.
- Arduino Uno Rev3 is also the most used and documented board in the Arduino family. There are many tutorials and projects available online with instructions for you to get started.
- It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, 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 an AC-to-DC adapter or battery to get started.



- Seeeduino V4.2
- Seeeduino V4.2 is based on the Arduino UNO bootloader. Our Seeeduino V4.2 is basically a much cheaper Arduino UNO with more functions!
- It has the same hardware and functions with a few additional features only found on our Seeeduino V4.2 like:
 - Switch to choose the system supply voltage, 3.3V or 5V, which is very useful if you want to set the system to 3.3V to save power.
 - Three on-board Grove interface allows your board to easily connect to our Grove modules. (Will explain more about Grove in our Sensors and Shields sector!)
 - Use of a DC-to-DC converter instead of an LDO (Low DropOut regulator), for enhanced efficiency



- Seeeduo Nano
- Want a smaller Arduino UNO or Seeeduo V4.2 for your project needs? Then the Seeeduo Nano would be perfect for you!
- The Seeeduo Nano is a compact board similar to the Seeeduo V4.2/Arduino UNO, and it is fully compatible with Arduino Nano on pinout and sizes.
- Standing at **43.18 mm × 18.54 mm** and less than a quarter size compared to the Seeeduo V4.2 the Seeeduo Nano size and also reliability allows them to be easily integrated into many projects like wearables, mini robots and many more!
- In addition, the Seeeduo Nano features 1 on-board Grove interface which allows your board to easily connect to our Grove modules.



- Seeeduino Mega(ATmega2560)
- Want a bigger, better and mega Arduino? The Seeeduino Mega definitely fits in that category.
- Seeeduino Mega is a powerful micro-controller derived from Arduino Mega. It features **ATmega2560 processor** which brings a large number of I/O pins.
 - It features as much as **70 digital I/O, 16 analog inputs, 14 PWM, and 4 hardware serial ports**
- Compared to Arduino Mega, we shrunk the volume of Arduino Mega by at least 30% and made it 100% compatible with Seeed Shield products.
- With this board, it is very suitable for projects which require many digital inputs or outputs like LEDs, buttons, etc.
-



```
sketch_may07a | Arduino 1.8.17 (Windows Store 1.8.42.0)
File Edit Sketch Tools Help

sketch_may07a
void setup() {
  // put your setup code here, to run once.
}

void loop() {
  // put your main code here, to run repeatedly.
}
```

1 -> Program name and Version



sketch_may07a | Arduino 1.8.13 (Windows Store 1.8.42.0)
File Edit Sketch Tools Help

```
sketch_may07a  
void setup() {  
  // put your setup code here, to run once:  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
}
```

1 2 3
4
5 6 7

- 1 -> Program name and Version
- 2 -> Menu Bar
- 3 -> Tool Bar
- 4 -> Text Editor
- 5 -> Message area/ Status area
- 6 -> Text Console
- 7 -> Connection area



The screenshot shows the Arduino IDE interface. The title bar reads "sketch_may07a | Arduino 1.8.13 (Windows Store 1.8.42.0)". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for "New", "Open", "Save", "Verify", and "Upload". The sketch editor contains the following code:

```
sketch_may07a
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Tool Bar: - frequent commands

- Verify - checks our code for errors & compile it
- Upload - compiles our code & upload it to the Arduino board
- New - create a new sketch
- Open - used to open already saved programs
- Save - saves our programs



sketch_may07a | Arduino 1.8.12 (Windows Store 1.8.42.0)

File Edit Sketch Tools Help

```
sketch_may07a
void setup() {
  // put your setup code here, to run once:
}

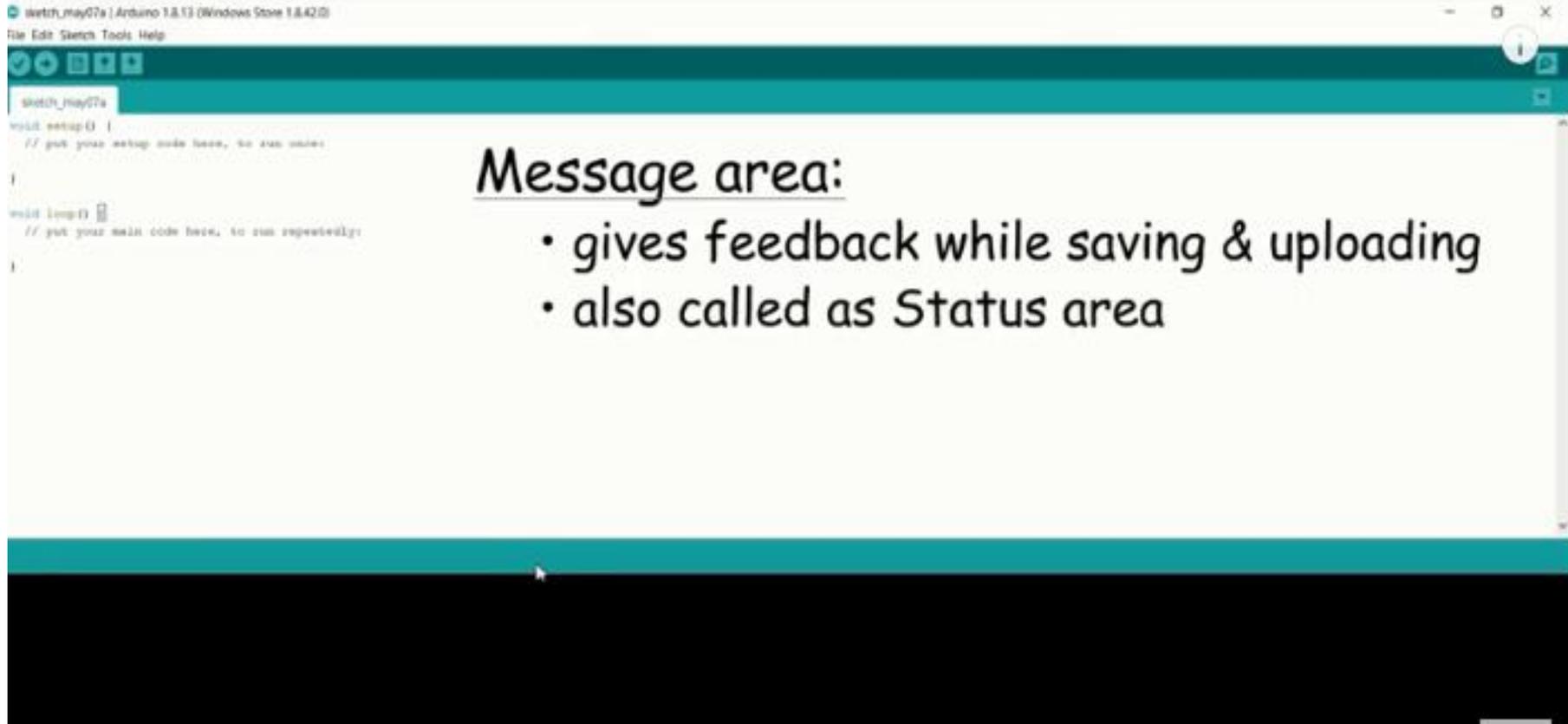
void loop() {
  // put your main code here, to run repeatedly:
}

```

Text Editor:

- Used to write our code
- Features
 - Cutting/Pasting
 - Searching/Replacing text

File extension - **.ion**
{program_name}.ion



Message area:

- gives feedback while saving & uploading
- also called as Status area



sketch_may07a | Arduino 1.8.13 (Windows Store 1.8.42.0)

File Edit Sketch Tools Help



```
sketch_may07a
void setup() {
  // put your setup code here, to run once:
}

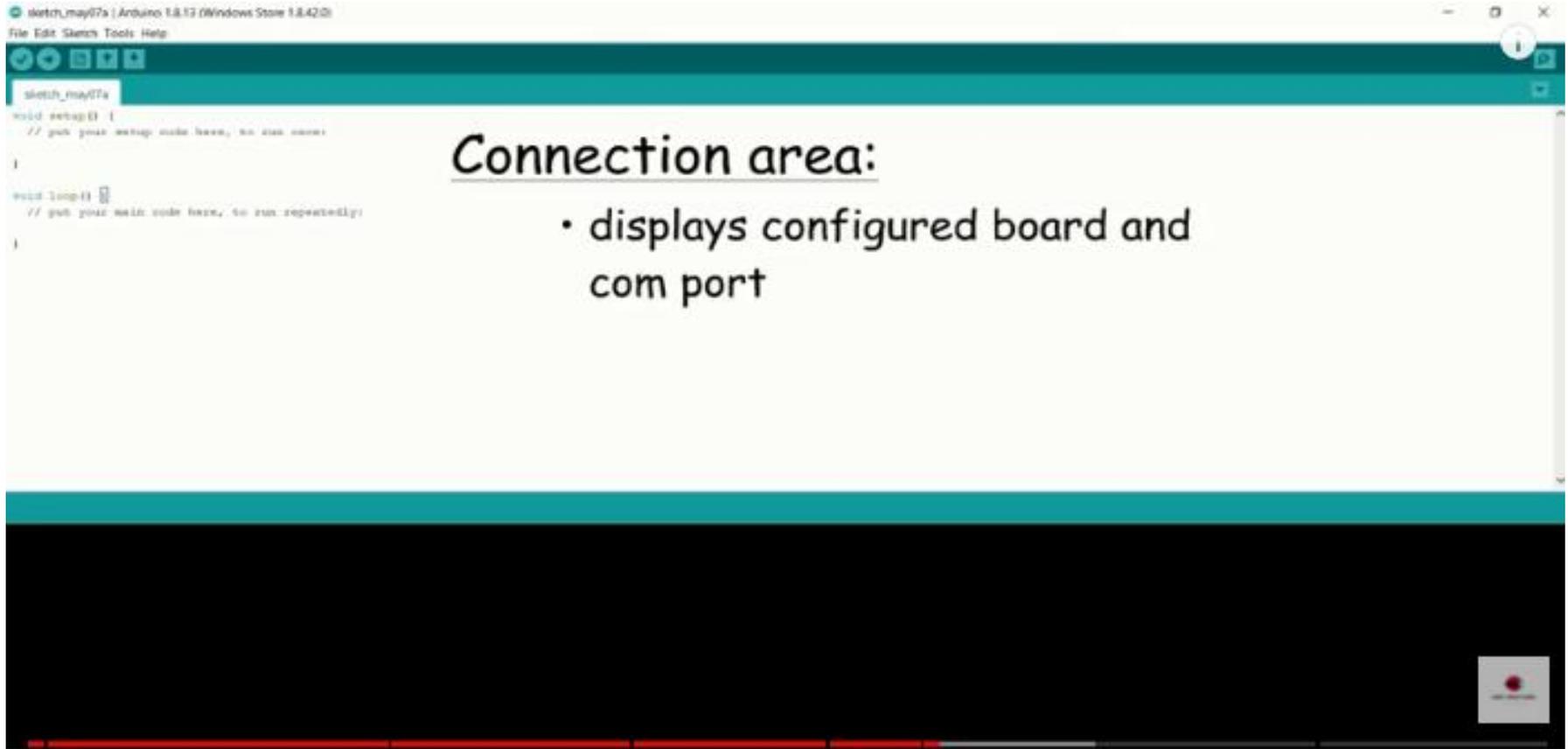
void loop() {
  // put your main code here, to run repeatedly:
}

//
```

Concole (Debug window):

- displays text output by the Arduino IDE
- includes
 - complete error message
 - other information







Serial Monitor:

- displays the serial message send by the Arduino board
- used to send data to the board



- Arduino Software - Open Source
- Source code released under - Free Software Foundation
- C/C++ library with AVR Compiler



Sketch - program written using Arduino Software (IDE)



Arduino Program divided into three main parts:

- Structure
- Values - (variables & constants)
- Functions



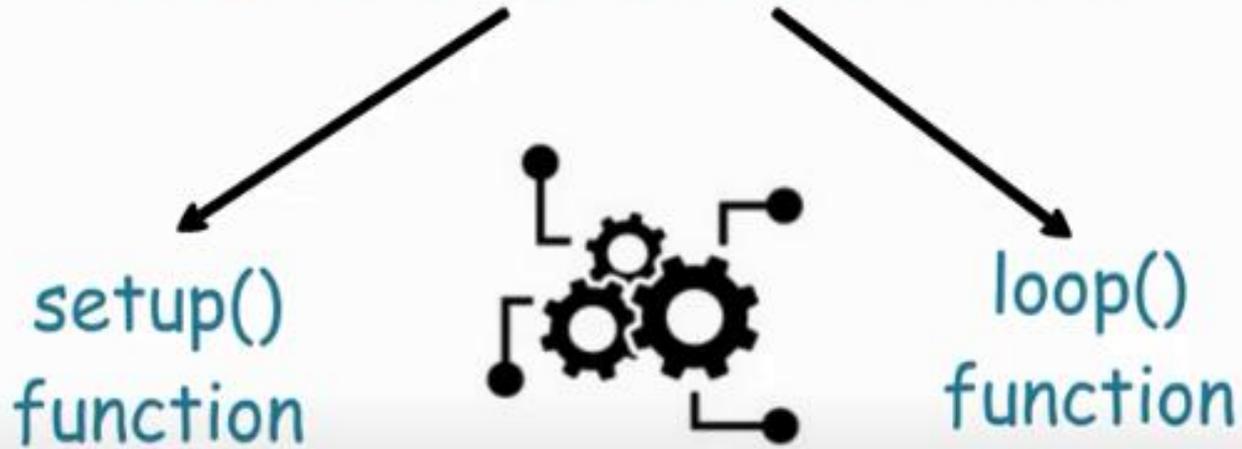
Arduino Software Structure

setup()
function

loop()
function



Arduino Software Structure



Function - group of statements that together perform a task
- can be called multiple times by a program





void setup():

- execute when a program starts
- used to define(initialize) the **variables**, **pin modes**, etc.,
- will only run once after each "**power up**" or "**reset**" of the Arduino board



void loop():

- After creating a setup() function, which initializes and set the initial values, we go for **void loop()**
- **it loops consecutively**
- runs over & over again forever
- used to actively control the Arduino board



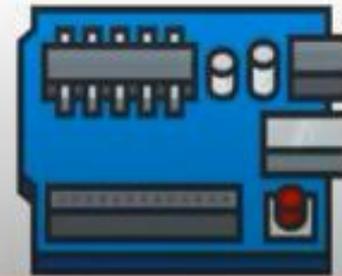
Program structure:

- void setup()

used to define initial
parameters
&
run only once

- void loop()

runs over & over again
forever





Digital I/O Functions

- `pinMode()`
- `digitalWrite()`
- `digitalRead()`



Digital I/O Functions

pinMode()

- used to configures the specific digital pin to behave as an "input" or "output"

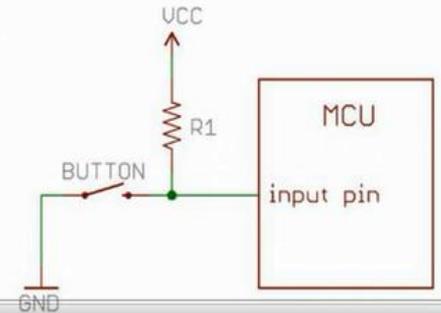
• syntax: pinMode(pin , mode)

-> pin : Arduino pin numbers

-> mode : INPUT, OUTPUT, INPUT_PULLUP

↓
read

↓
write

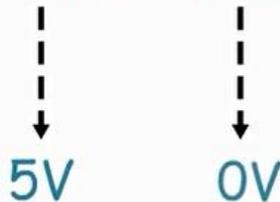




Digital I/O Functions

digitalWrite()

- used to write a **HIGH** or **LOW** value to a digital pin
- syntax: digitalWrite(pin , value)
 - > pin : Arduino pin numbers
 - > value : **HIGH** or **LOW**





Digital I/O Functions

digitalRead()

- used to read the value from a specified digital pin
- syntax: `digitalRead(pin)`
 - > pin : the Arduino pin number we want to read
- Return: HIGH or LOW

Digital - 5V or 0V



Digital I/O Functions

Example:

```
int val = 0 ;
void setup()
{
  pinMode(13, OUTPUT); //configuring pin 13 as Output
  pinMode(7, INPUT);   //configuring pin 7 as Input
}
void loop()
{
  val = digitalRead(7); // reading value from pin 7
  digitalWrite(13,HIGH); // writing High in pin 13
```



Analog I/O Functions

- `analogRead()`
- `analogReference()`
- `analogWrite()`



Analog I/O Functions

analogRead()

- used to read values from the specific analog pins
- Input Value Range - 0 to 5V
- syntax: `analogRead(pin)`
 - > pin : name of the analog pin (A0 to A5)
- Return: 0 to 1023 (for 10 bits ADC)



Discussed
in the last
lecture

- analog input Voltage (V_{in})
- analog Reference Voltage (V_{ref})



Analog I/O Functions

analogReference()

- used to configure the reference voltage used for analog input
- default analog reference - 5V
- syntax: `analogReference(type)`

-> type : DEFAULT --> 5V

INTERNAL --> 1.1V

EXTERNAL --> the voltage applied to the AREF pin



Reference must be less than or equal
to Operating Voltage(5V)

Input analog voltage must be less than
or equal to reference voltage



Analog I/O Functions

analogWrite()

- used to write a analog Value to a pin with the help of PWM waves
- PWM - can be used to light at varying brightness or to drive a motor at various speed
- generate Rectangular wave at PWM pins



UNO:

PWM pins: 3, 5, 6, 9, 10, 11 --> in digital pins

Frequency: 490 Hz

pin 5 & 6 --> 980 Hz



Timing Functions

- delay()
- millis()



Timing Functions

delay()

- used to pause the program for the amount of time
- Input parameter - in milliseconds
- syntax: delay(ms)
 - > ms - the number of milliseconds to pause
- For Example,
 - delay(1000) -> 1 second Pause
 - delay(500) -> 0.5 second Pause





Timing Functions

millis()

- used to find the number of milliseconds passed since the Arduino board began running the current program
- This number will overflow (go back to zero) after approximately 50 days

- syntax: time = millis()
- Return: number of milliseconds passed since the program started



THANKS!