



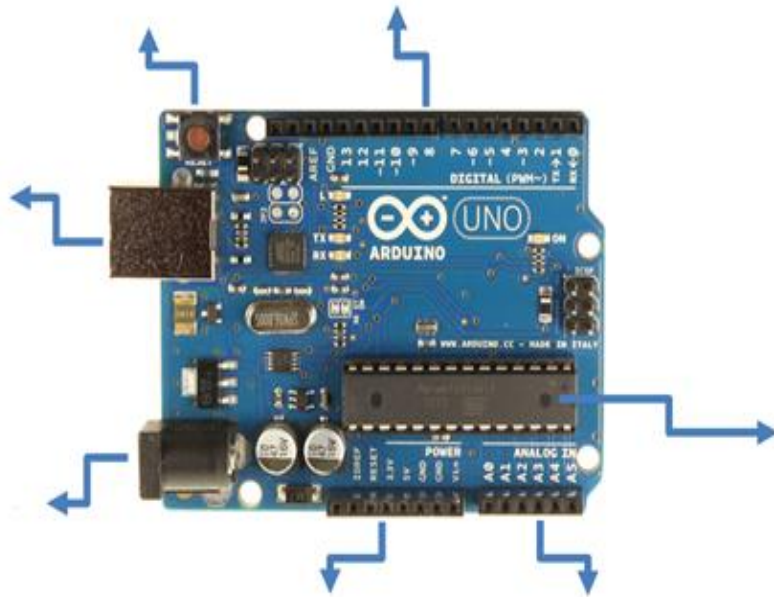
INTRODUCTION TO **Arduino** Hardware & Software Installation



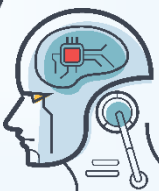
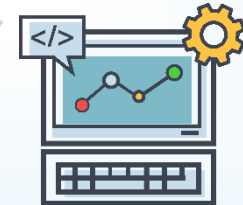
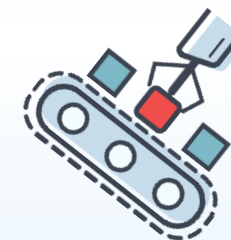
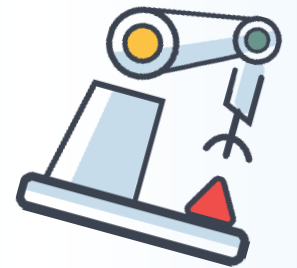
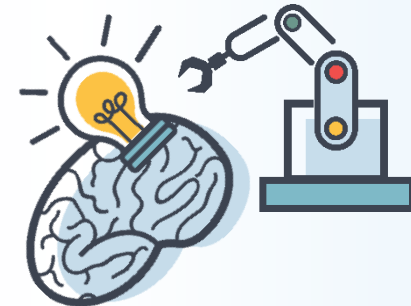


Activity Puzzle Time

Match the names given in the box to the Arduino.



- 1) Digital pins
- 2) Analog Pins
- 3) Power Connector
- 4) Power Pins
- 5) ATmega328 microcontroller
- 6) USB
- 7) Reset Button

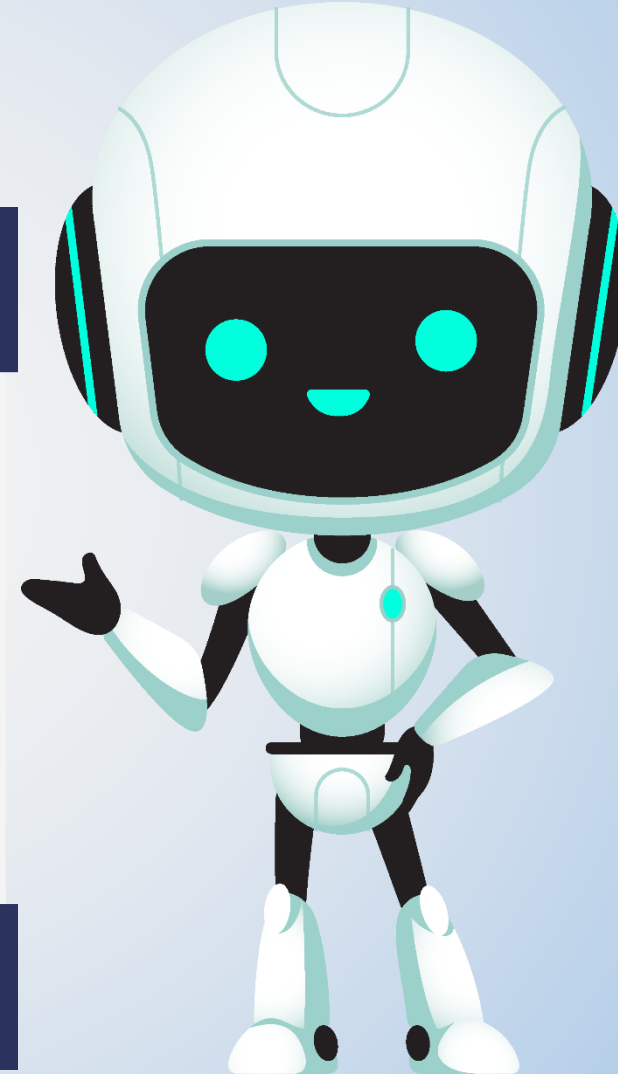




Scan the QR code to cross check the answers to the puzzle.



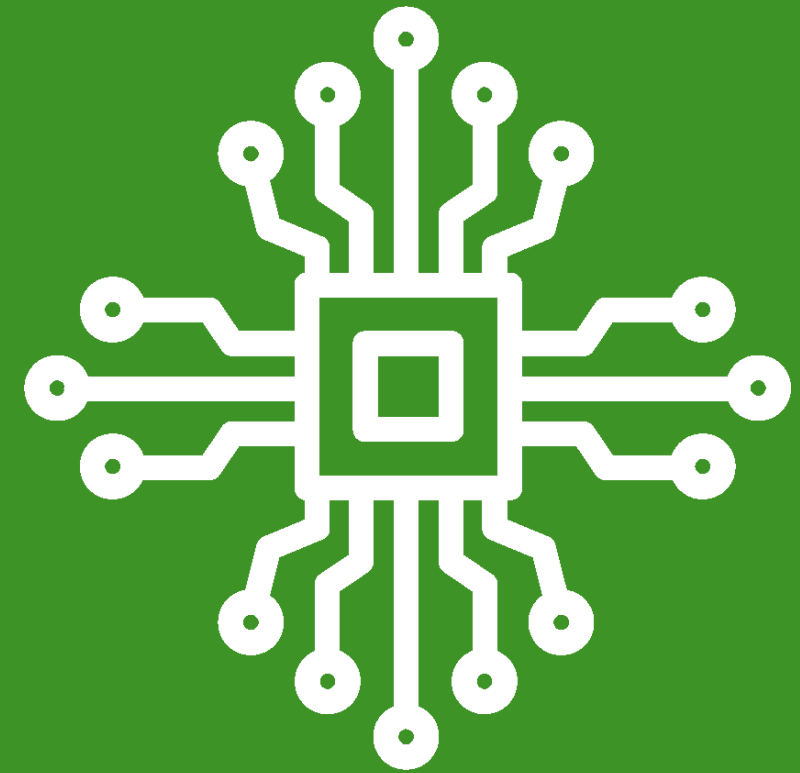
Activity Solution





What is Arduino

Arduino is an open-source prototyping platform. It is used by tinkerers, hobbyists, and makers to design and develop solutions to solve real world problems. It consists of both a physical programmable circuit board and a software, or IDE (Integrated Development Environment), where you can write and upload the computer code to the physical board.



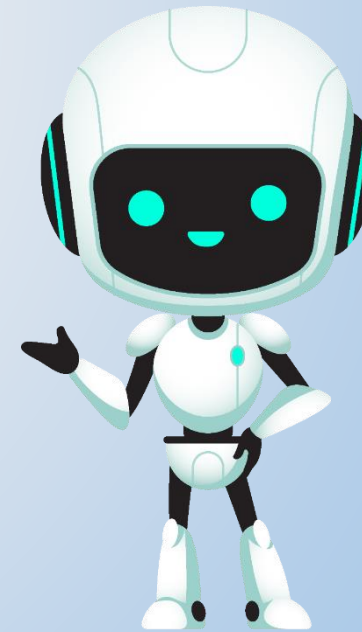
Arduino IDE Installation





Download Arduino IDE

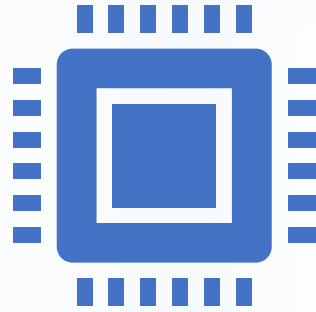
Tutorial Video Arduino IDE





Worksheet Time

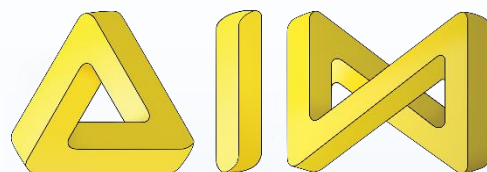




In 2005, building upon the work of Hernando Barragán (creator of Wiring), Massimo Banzi and David Cuartielles created Arduino, an easy-to-use programmable device for interactive art design projects, at the Interaction Design Institute Ivrea in Ivrea, Italy



SBC has microprocessor(s), memory, input/output (I/O) and other features required of a functional computer



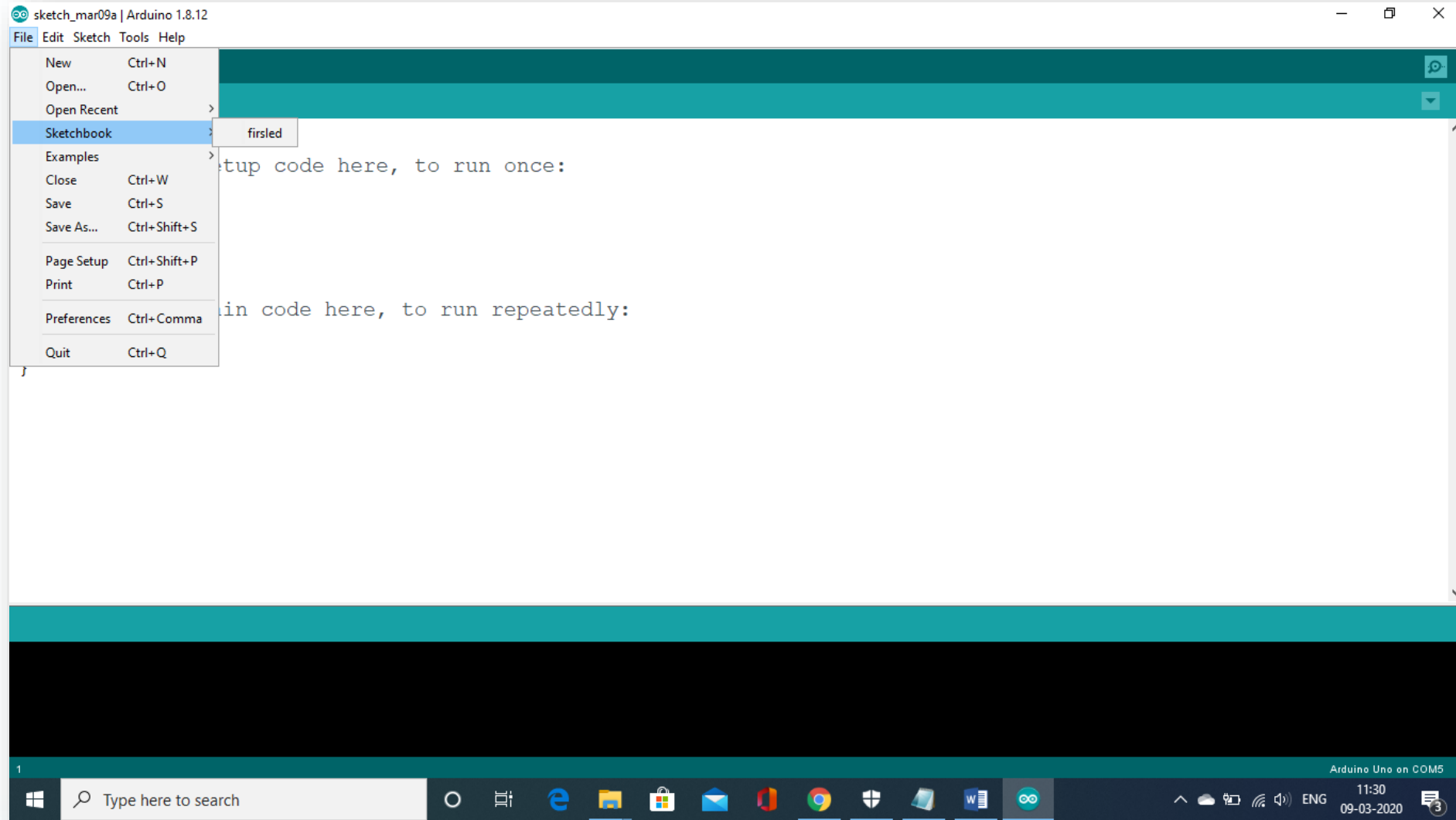
ATAL INNOVATION MISSION

Arduino

Interface



Arduino Interface

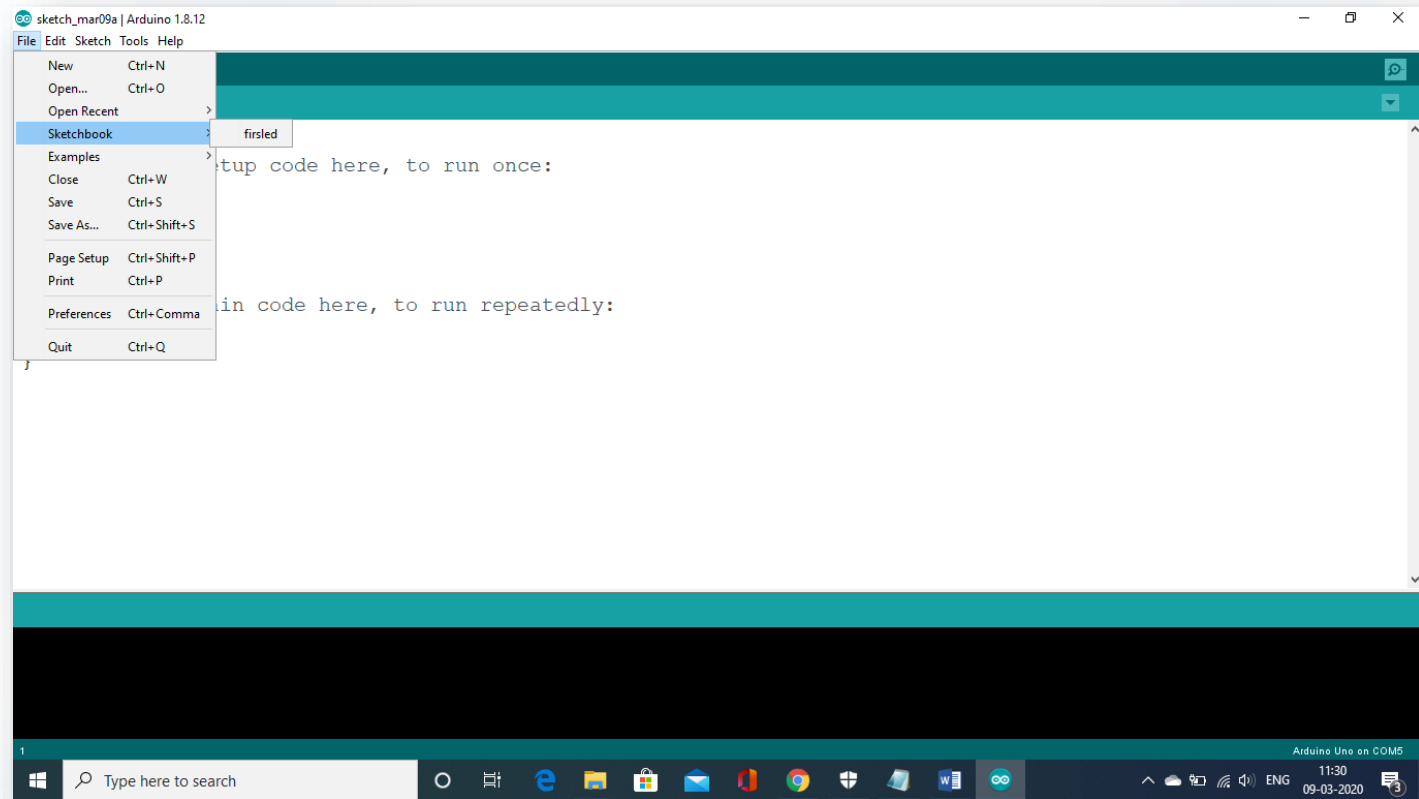


Introduction to IDE Options

Arduino IDE is an interface where you can write, compile, upload, debug and see the output in a serial monitor.

File

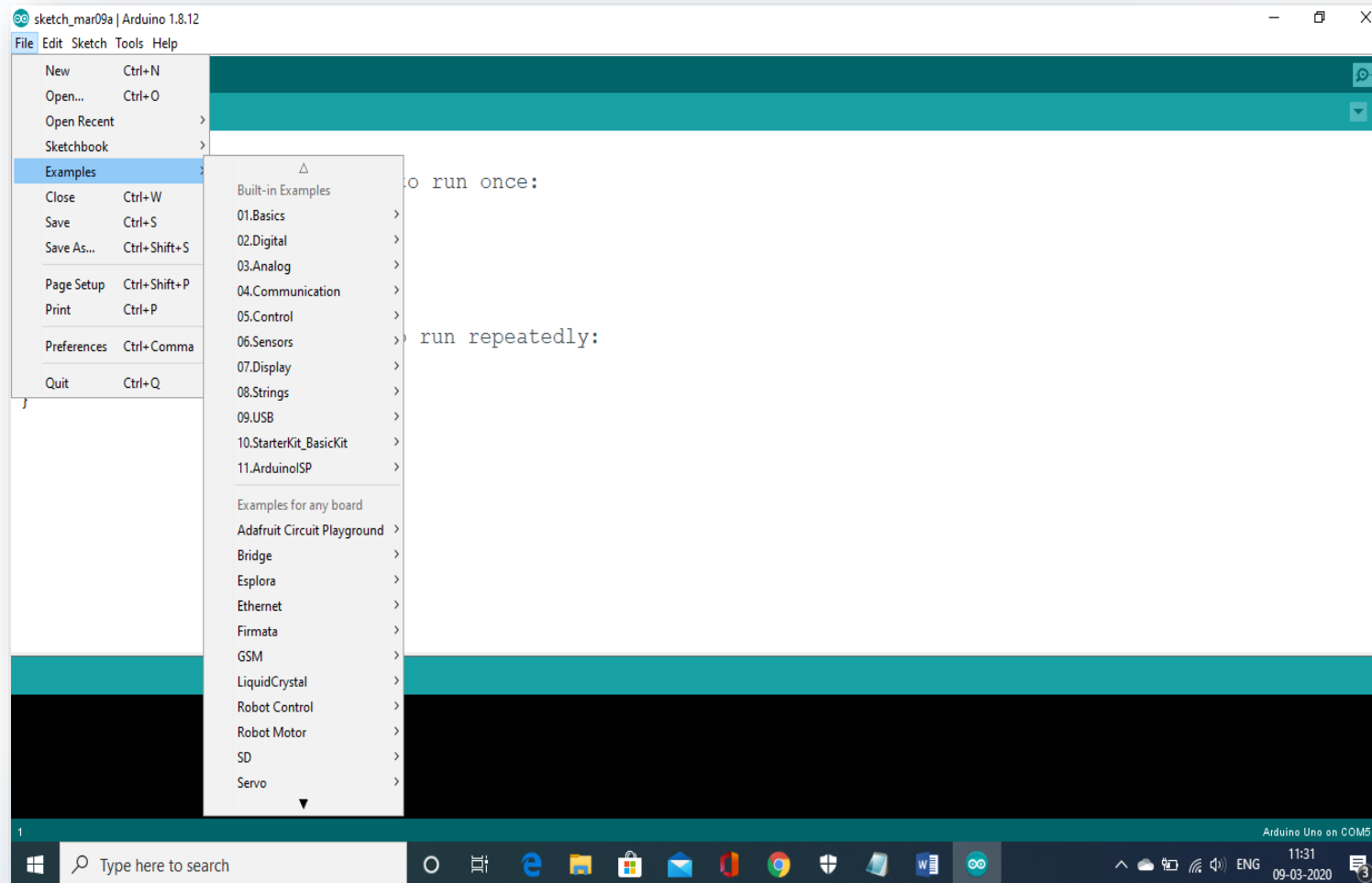
Sketchbook: Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instantly.



File

Examples:

Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.



Sketch

Verify/Compile: Checks your sketch for errors by compiling it.

Upload: Compiles and loads the file onto the board.

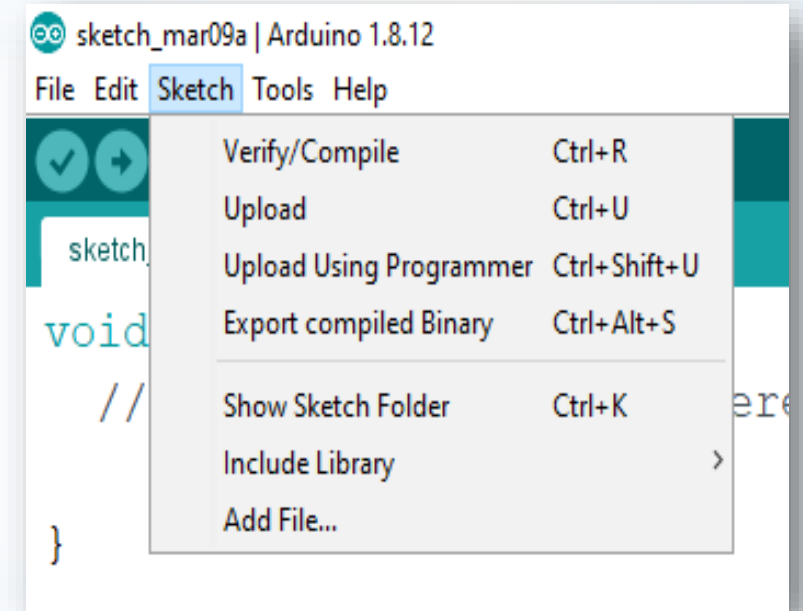
Upload Using Programmer: This will overwrite the bootloader on the board. It allows you to use the full capacity of the Flash memory for your sketch. (Rarely used).

Export Compiled Binary: Saves a (.hex) file that may be kept as archive or sent to the board using other tools.

Show Sketch Folder: Opens the current sketch folder.

Include Library: Adds a library to your sketch by inserting #include statements at the start of your code. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files that you have downloaded through other sources.

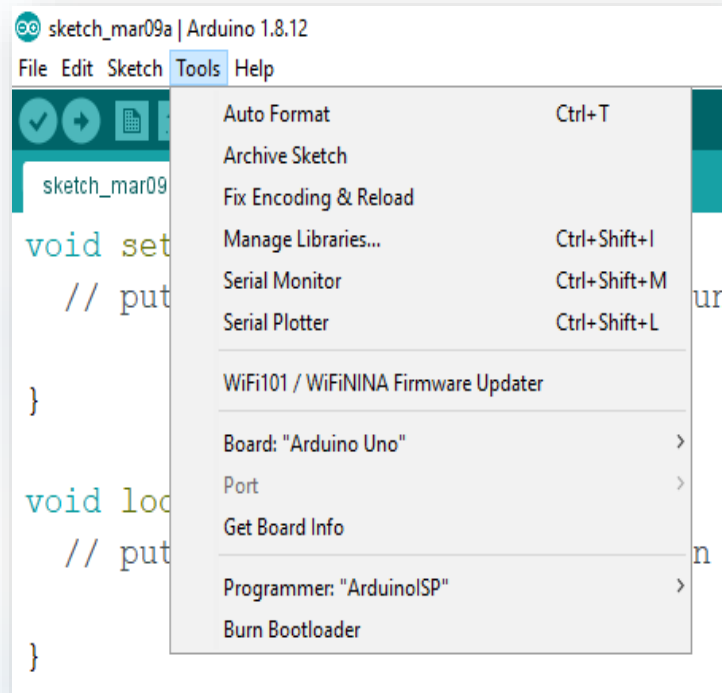
Add File: Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window



Auto Format: This formats your code visually: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.

Archive Sketch: Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

Fix Encoding & Reload: Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.



Tools

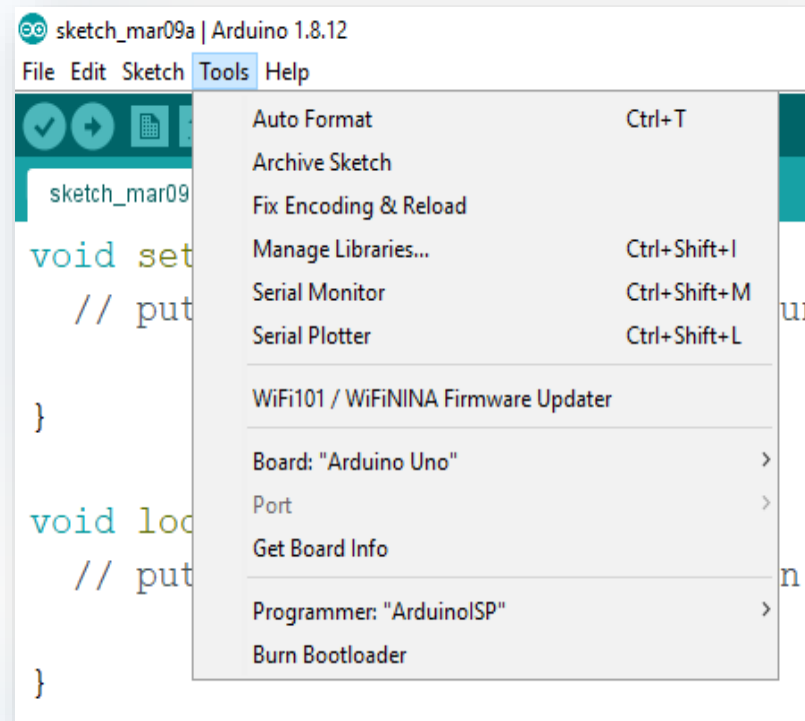
Serial Monitor: Opens the serial monitor window and acts as a separate terminal to display the output data transmitted from the sensors.

Board: To select the board that you're using.

Port: This menu contains all the serial devices (real or virtual) on your machine.

Programmer: For selecting a hardware programmer when programming a board or chip and not using the on-board USB-serial connection. (Rarely used).

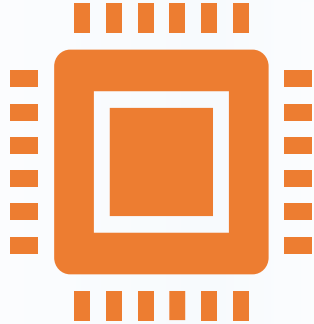
Burn Bootloader: This allows you to burn a bootloader onto the microcontroller on an Arduino board. (Rarely used).



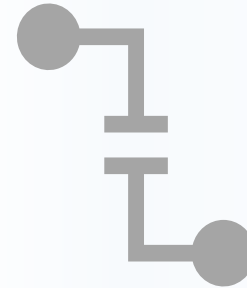


Worksheet Time

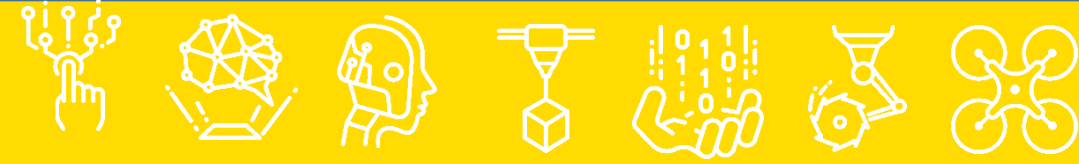




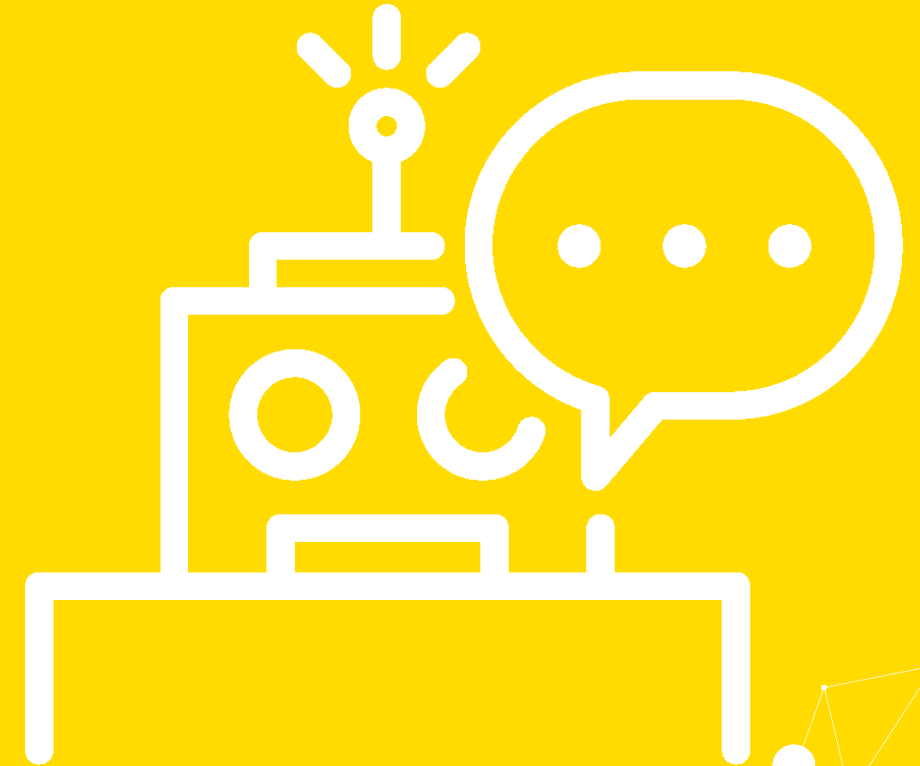
Arduino Uno uses a different USB chip which makes installation of the Arduino software lot easier.



The first widely distributed Arduino board, the Diecimila, was released in 2007.



Blink a LED using Arduino

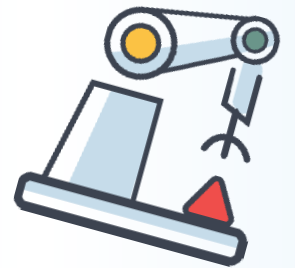
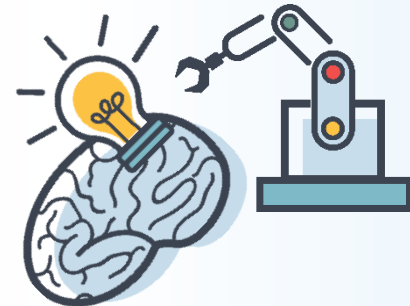




Activity

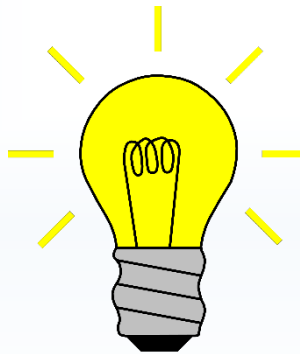
Blink the inbuilt LED on Arduino.

- Double click on Arduino IDE to open the Arduino programming interface.
- Click on File → Examples → Basics → Blink. (To open sample blink programme for Inbuilt LED)
- Click on Tools → Board. (Select the Arduino UNO board from the list)
- Click on Tools → Port. (Select the COM port connected to the Computer)
- Click on Tools → Programmer. (Select Arduino ISP from the list)
- Click on Sketch → Verify/Compile. (To check whether the programme is free from errors)
- Click on Sketch → Upload. (To upload the programme in Arduino)



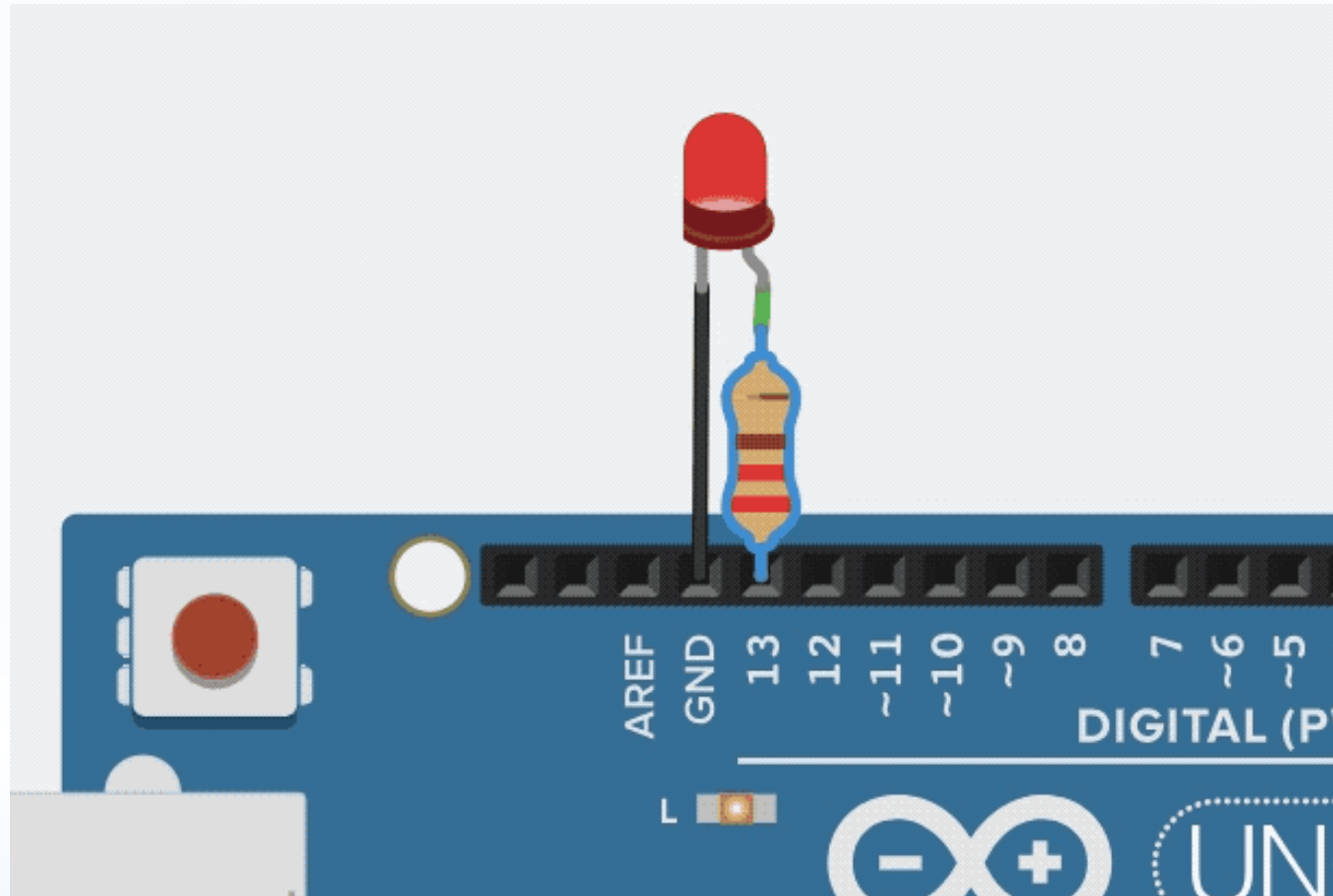
ANALYSIS

Using the example code present in Arduino, it is possible to blink the In-built LED without any connection and through the code itself.



Arduino LED Blinking

Now connect the LED to PIN no. 13 as shown in the diagram & Try again using the code used for blinking an Inbuilt LED.



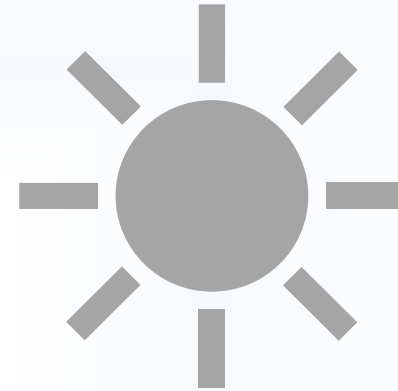


Worksheet Time





An incandescent lamp converts about 9-10 percent of the energy fed to it into light, whereas LEDs convert nearly 100 percent of the energy they consume as light.



LEDs contain no mercury— and at least 95 percent of an LED is recyclable.

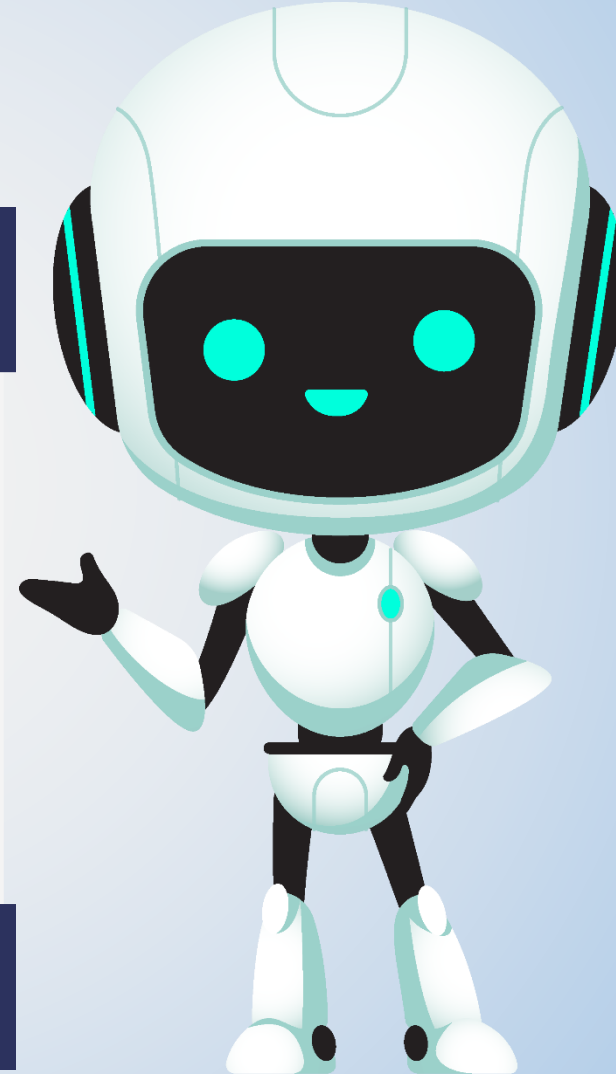
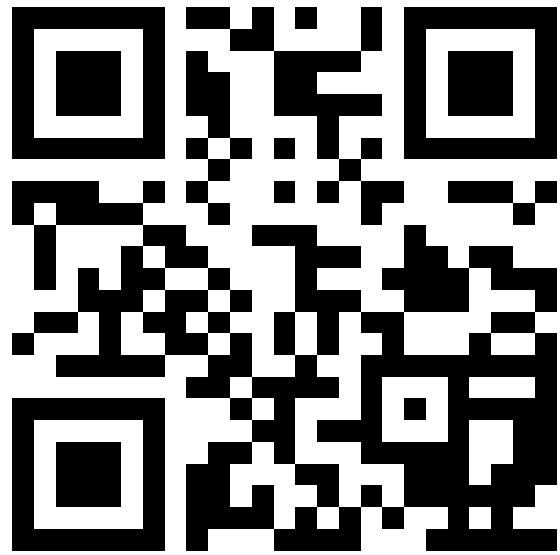


Coding & Basics



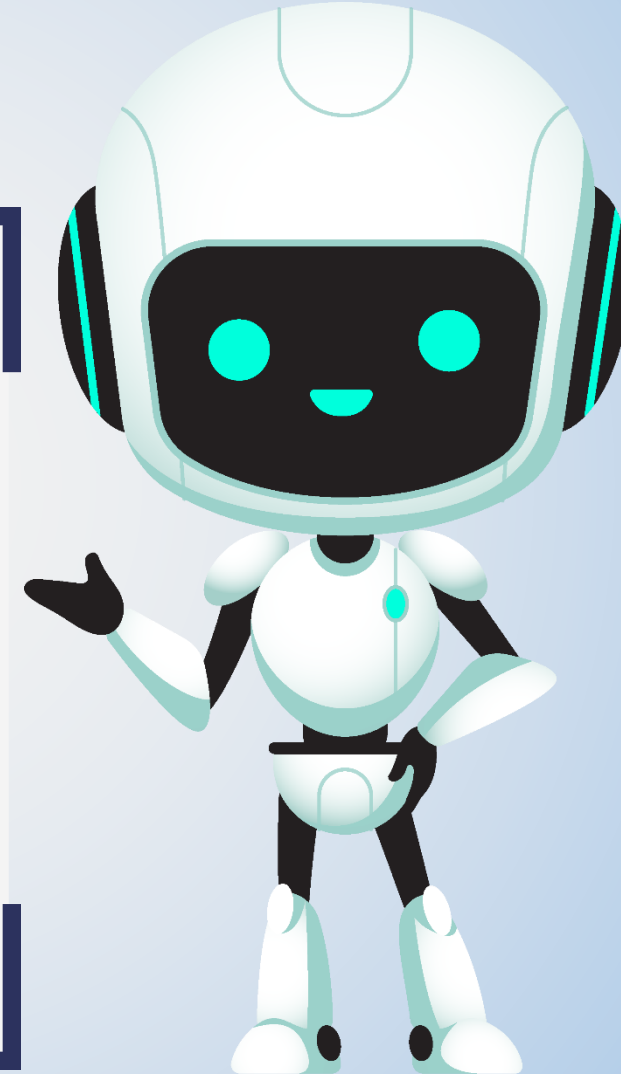


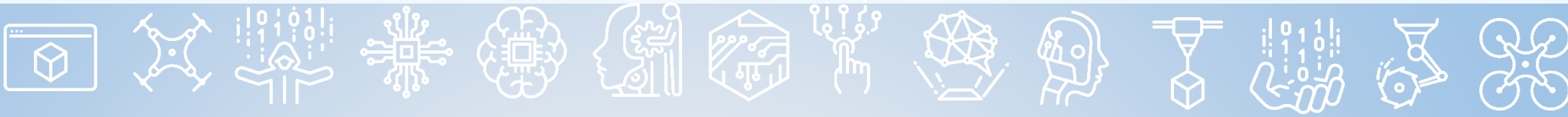
Video on Explanation of Basics of coding in Arduino IDE



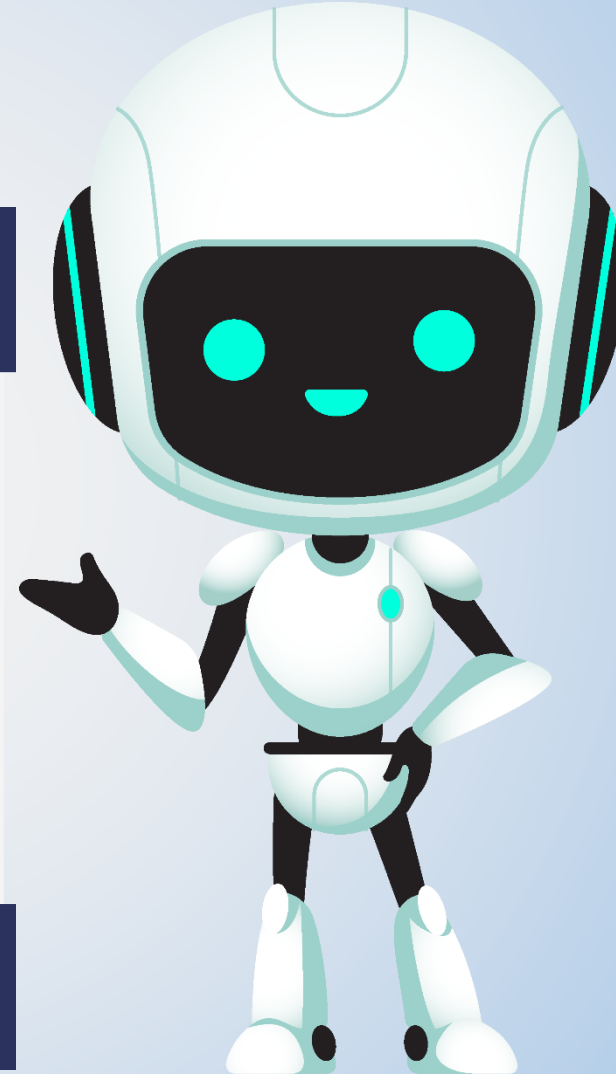
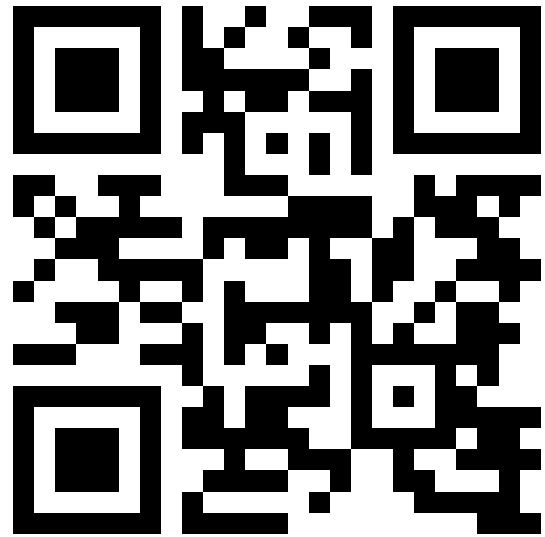


Video on explanation of basics of coding in Arduino IDE





**Video on explanation of program flow by demonstrating
example codes present in Arduino IDE**



Sample Code

```
// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin = 13;      // the number of the LED pin
// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

We use some system define function that helps to operate the Arduino (Arduino IDE uses a case sensitive language).

1. `const int buttonPin = 2;` to declare a constant that won't change. Pin number where push button is attached
2. `const int ledPin = 13;` Pin number where LED is attached
3. `int buttonState = 0;` Variable to read Push button status.

```
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}
```

`pinMode()` function determines how the pins will operate. In bracket write details of pin no. & whether it is an INPUT or OUTPUT.

4. `digitalRead(buttonPin);` - digitalRead, Reads the value from the pin
- 5.

```
void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
  // check if the pushbutton is pressed.
```

If the Boolean expression evaluates to be true, then the if block will be executed, otherwise, the else block will be executed.

```
  //If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Explanation

We use some system define a function that helps to operate the Arduino (Arduino IDE uses a case sensitive language).

1. `pinMode (13, OUTPUT);` or `pinMode (13, INPUT);`

`pinMode ()` function determines how the pins will operate. In bracket put important information like pin no. & whether it an INPUT or OUTPUT.

2. `digitalWrite(13, HIGH);` or `analogWrite(13, 25);`

The `analogWrite` is mainly used to update the status of analog pins and is also used to map the analog values on the PWM (Pulse Width Modulation) pins. Whereas the digital is used for digital Sensors or Actuators.

3. `analogRead(13);` or `digitalRead(13);`

The `analogRead` or `digitalRead`, Read the value from the pin, if a sensor is digital then we will use “`digitalRead`” or if a sensor is analog then we will use “`analogRead`”.

4. If or Else statement.

If the Boolean expression evaluates to true, then the if block will be executed, otherwise, the else block will be executed. Have a look at the syntax.

```
if(boolean_expression) {  
    /* statement(s) will execute if the Boolean expression is true */  
} else {  
    /* statement(s) will execute if the Boolean expression is false */  
}
```

* Note: In the end of statement we need to put terminator at the end denoted by “;”.

Operators in programming language.

Sl.No	Name of operator	Symbol	Syntax
1	Addition	+	$a + b$
2	Subtraction	-	$a - b$
3	Multiplication	*	$a * b$
4	Division	/	a / b
5	Increment Prefix	++	++a
6	Increment postfix	++	a++
7	Decrement prefix	--	--a
8	Decrement postfix	--	a--
9	Basic assignment	=	a=b
10	Equal to	==	a==b
11	Not equal to	!=	a!=b
12	Greater than	>	a>b
13	Less than	<	a<b
14	Greater than or equal to	>=	a>=b
15	Less than or equal to	<=	a<=b



Any Doubt's in Programming





Worksheet Time





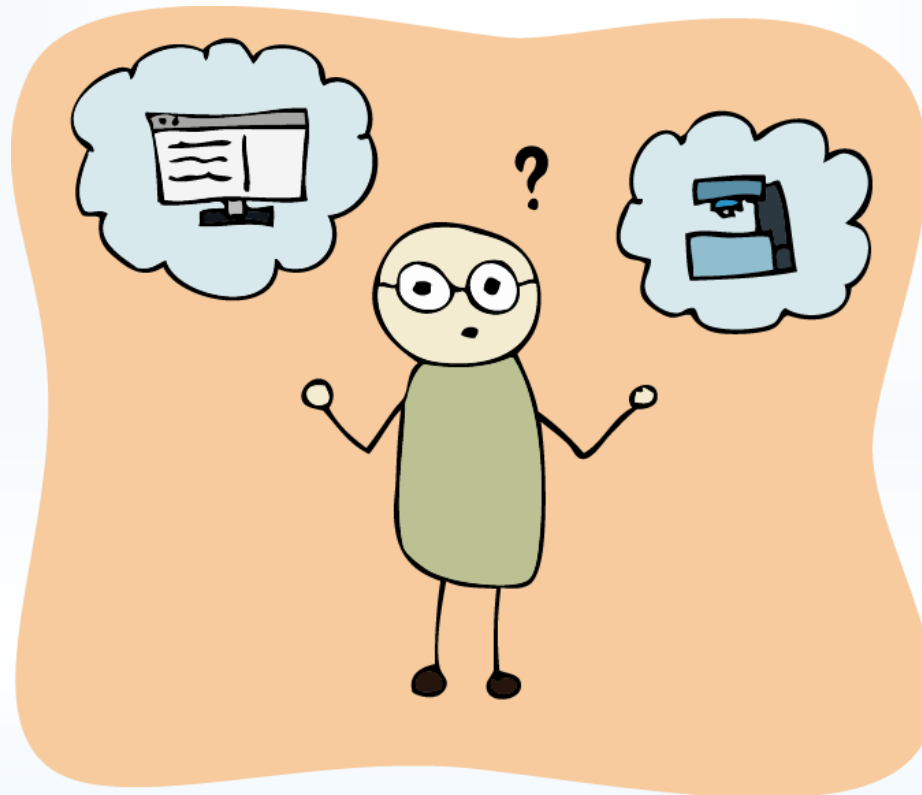
Arduino

Troubleshooting



Let's Troubleshoot Arduino Connections and code

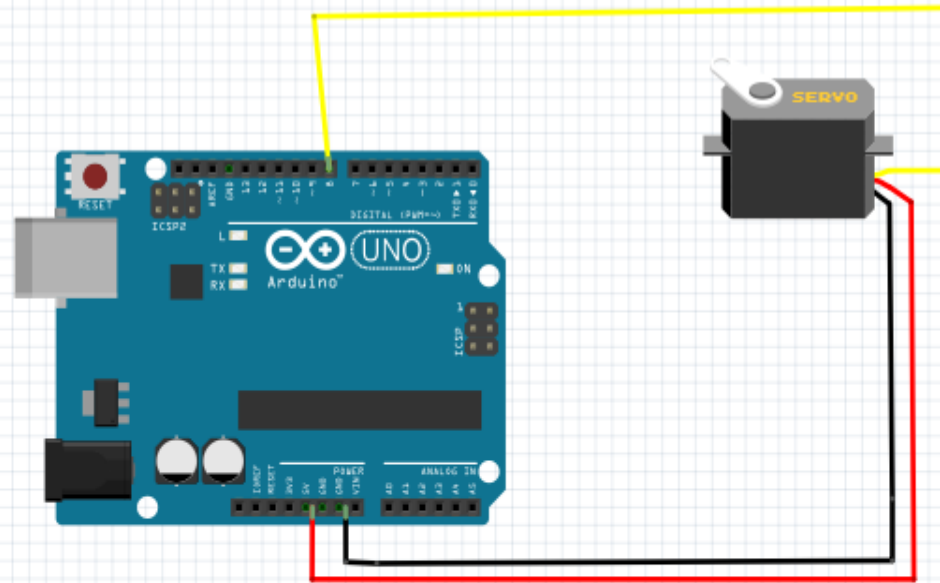
Try to find out the possible errors that can be present in the image shown in next slide and try to correct the code.



Find the Errors and troubleshoot!



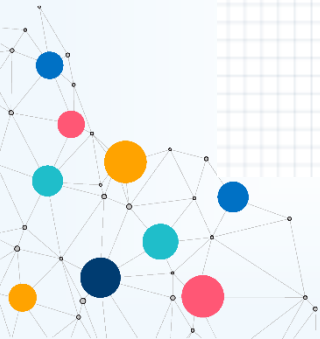
Connect the circuit as shown and write the code mentioned in IDE



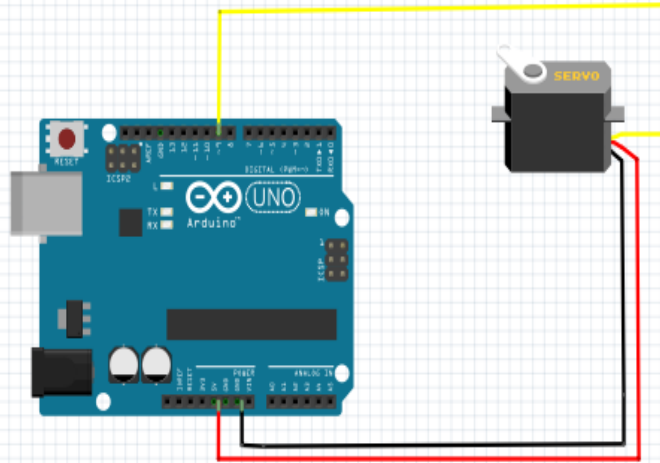
Yellow wire-----Signal pin
Red wire-----+5V
Black Wire-----GND(Ground)

Code:

```
#include <Servo.h>
int servoPin = 9;
Servo servo;
int angle = 0; // servo position in degrees
void setup()
{
  servo.attach(servoPin)
}
void loop()
{
  // scan from 0 to 180 degrees
  for(angle = 0; angle < 180; angle++)
  {
    servo.write(angle);
    Delay(15);
  }
  // now scan back from 180 to 0 degrees
  for(Angle = 180; Angle > 0; Angle--)
  {
    servo.write(angle);
    Delay(15);
  }
}
```



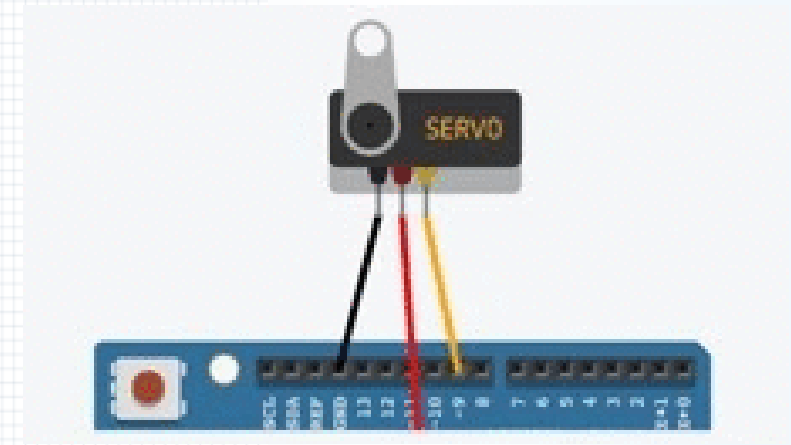
Answer!



Yellow Wire-----Signal pin
Red Wire-----+5v
Black Wire-----Ground

Code:

```
#include <Servo.h>
int servoPin = 9;
Servo servo;
int angle = 0; // servo position in degrees
void setup()
{
  servo.attach(servoPin);
}
void loop()
{
  // scan from 0 to 180 degrees
  for(angle = 0; angle < 180; angle++)
  {
    servo.write(angle);
    delay(15);
  }
  // now scan back from 180 to 0 degrees
  for(angle = 180; angle > 0; angle--)
  {
    servo.write(angle);
    delay(15);
  }
}
```



Instructions for correcting the code and connections

- Connect the yellow (signal pin) to digital pin 9.
- In void setup() function there is a semicolon missing for the command. Correct command is `servo.attach(servoPin);`
- In void loop() function, delay function is written as `delay` and not `Delay`. The correct code follows: `delay(15);`
- In void loop() function, in for loop function the variable declaration is corrected as `angle` and not `Angle`. The correct code follows: `for(angle=180;angle>0;angle--)`
- Correcting the following codes will result in successful compile without any errors and will result in uploading of programme to the board without any errors.

General Errors while Compiling

SEMI COLON ERROR: This type of error is generated when we forget to put a semicolon at the end of the instruction.

VARIABLE NAME ERROR: A variable is used to save the data which has a name, value and type.

- We should keep in mind not to assign a variable name starting with a number
- No spacing in between the variable name
- The variable name once assigned has to be used the same throughout the programme.

BRACKET MISSING ERROR: This type of error generates whenever either opening bracket or closing bracket missing in the programme.

SERIAL PORT NOT FOUND ERROR: This error is generated whenever we upload the programme without selecting the port number.



General Errors while Compiling



BOARD ERROR: This error is generated whenever a wrong board is selected for the programme. To solve this error, select the proper board before uploading the programme.

OTHER ERRORS: If a programme is written for Arduino Mega and the digital input pins are assigned more than 13, then the following code cannot run on Arduino UNO as it has less number of digital pins as compared to Arduino Mega.

WRONG CONNECTION ERROR: If a programme is written for led and output pin is assigned to digital pin 10 but the connection is made for digital pin 11 then the led will function as per the programme.





Worksheet Time





Once your computer is connected with Arduino via USB cable, it remembers the port assigned. The next time you connect to the same port it will automatically get connected to the same port number. You don't have to again follow the process of adding the port.



