Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

O-Level (M4-R5.1)

Internet of Things and its Applications by

Ms.Maffi

IOT Instructor

Zed-King Group of Institutions Kaithal

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

O-Level (M4-R5.1)

Internet of Things and its Applications

Video Tutorial

O-Level M4-R5.1 Full Video Playlist
Available on YouTube Channel **Zed-King Computer Institute Kaithal**

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

IOT Complete Crash Course on Zed-King App

(100 % Pass Guarantee)

For free PDF Notes

Our Telegram Channel: t.me/boost/Zedking_Institute

IOT M4-R5.1

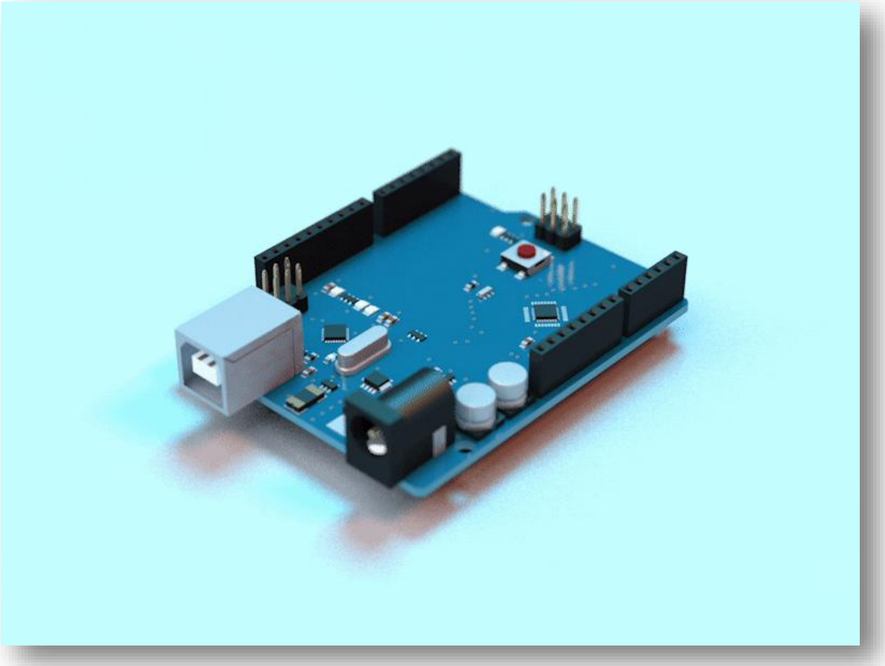
Chapter 4 -
Building of IOT Applications

Class-1

❖ What is Arduino?

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- Arduino is an open-source electronics platform based on easy-to-use hardware and software. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.
- Arduino was created in the year 2005 by two Italian engineers **David Cuartielles** and **Massimo Banzi**.
- "**Uno**" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0.



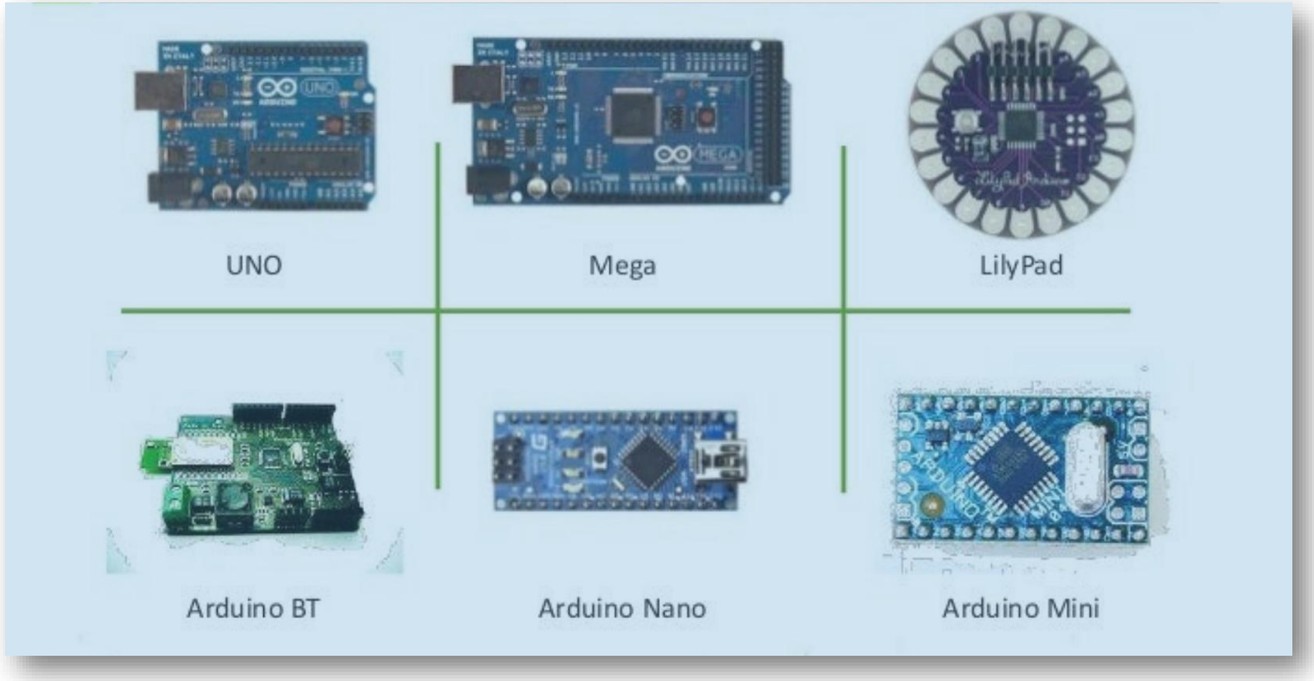
YouTube @zedkinginstitute Subscribe

❖ Types of Arduino Boards

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

There are 8 types of Arduino boards:

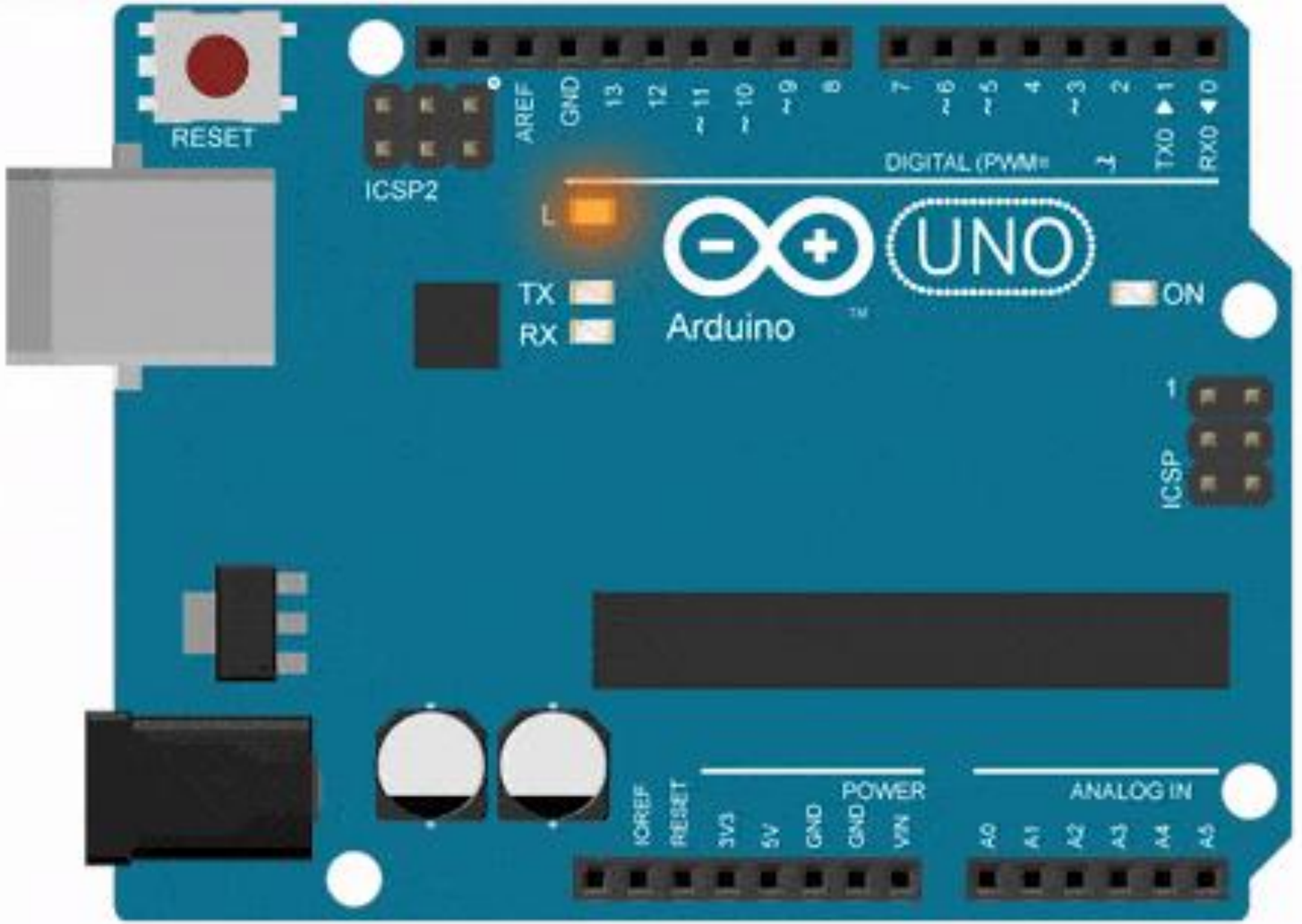
1. Arduino UNO
2. Arduino Mega’
3. Arduino Leonardo
4. Arduino Micro
5. Lilypad Arduino Board
6. Arduino Red Board
7. Arduino Shields
8. Arduino Zero



YouTube @zedkinginstitute Subscribe

❖ Arduino UNO Hardware

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified



YouTube @zedkinginstitute
Subscribe

❖ Component Explanation

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **ATmega328 Microcontroller-** It is a single chip Microcontroller of the ATmel family. The processor code inside it is of 8-bit. It combines **Memory (SRAM, EEPROM, and Flash), Analog to Digital Converter, SPI serial ports, I/O lines, registers, timer, external and internal interrupts, and oscillator.**
- **ICSP pin** - The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.
- **Power LED Indicator-** The ON status of LED shows the power is activated. When the power is OFF, the LED will not light up.
- **Digital I/O pins-** The digital pins have the value HIGH or LOW. The pins numbered from D0 to D13 are digital pins.
- **TX and RX LED's-** The successful flow of data is represented by the lighting of these LED's.

YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **AREF-** The Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.
- **Reset button-** It is used to add a Reset button to the connection.
- **USB-** It allows the board to connect to the computer. It is essential for the programming of the Arduino UNO board.
- **Crystal Oscillator-** The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.
- **Voltage Regulator-** The voltage regulator converts the input voltage to 5V.
- **GND-** Ground pins. The ground pin acts as a pin with zero voltage.
- **Vin-** It is the input voltage.
- **Analog Pins-** The pins numbered from A0 to A5 are analog pins. It can also act as GPIO (General Purpose Input Output) pins.

YouTube @zedkinginstitute Subscribe

❖ Hardware Key Points

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Microcontroller	ATmega328P (8-bit)(pico- power)
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output) (0-13)(0-RX)(1-TX)
PWM Digital I/O Pins	6 (3,5,6,9,10,11)
Analog Input Pins	6 (A0-A5)
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-2

❖ Arduino IDE (Software)

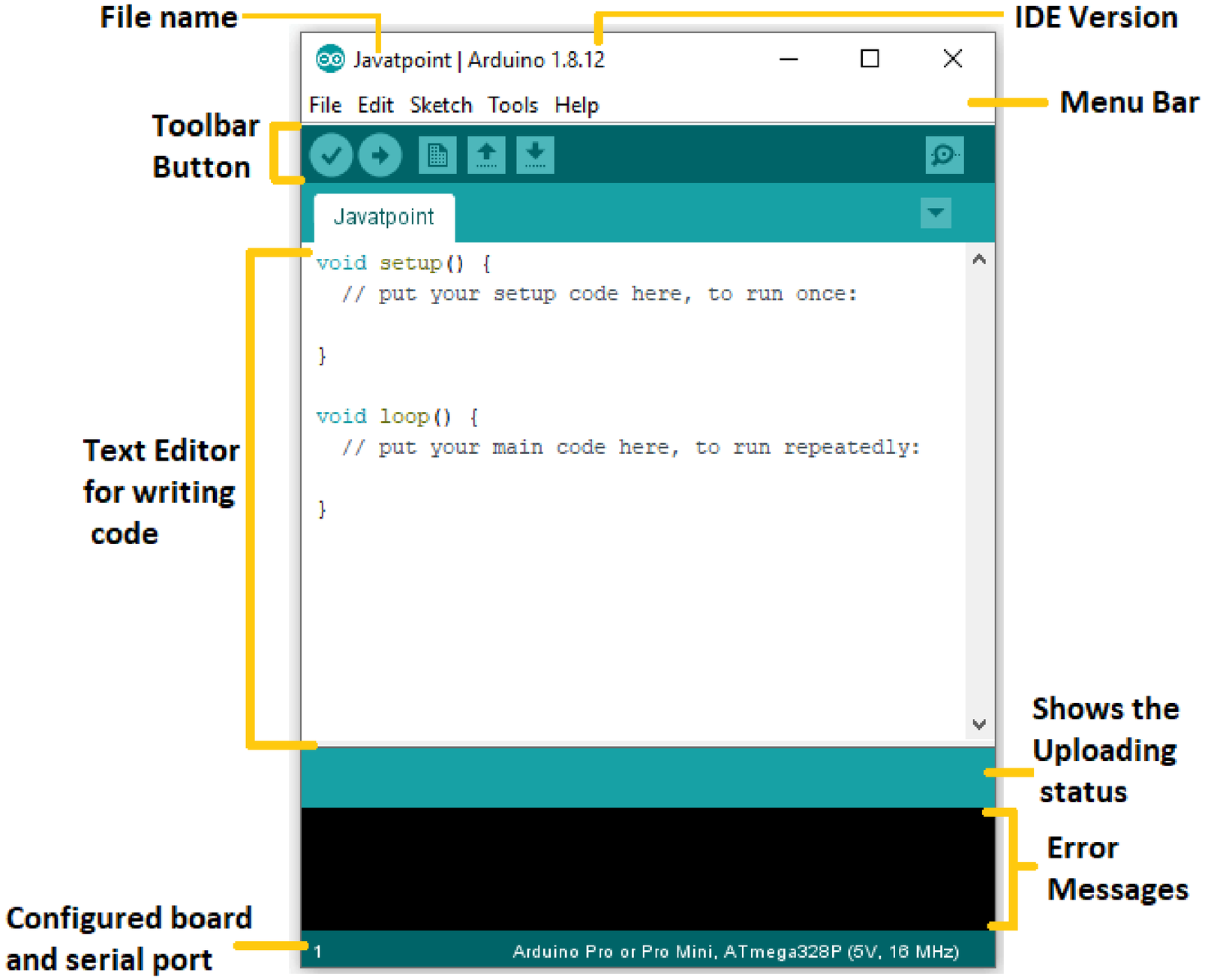
Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. Here, IDE stands for Integrated Development Environment.It supports the programming languages C and C++.
- The program or code written in the Arduino IDE is often called as sketch.
- We need to connect the Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino‘



YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified



YouTube @zedkinginstitute
Subscribe

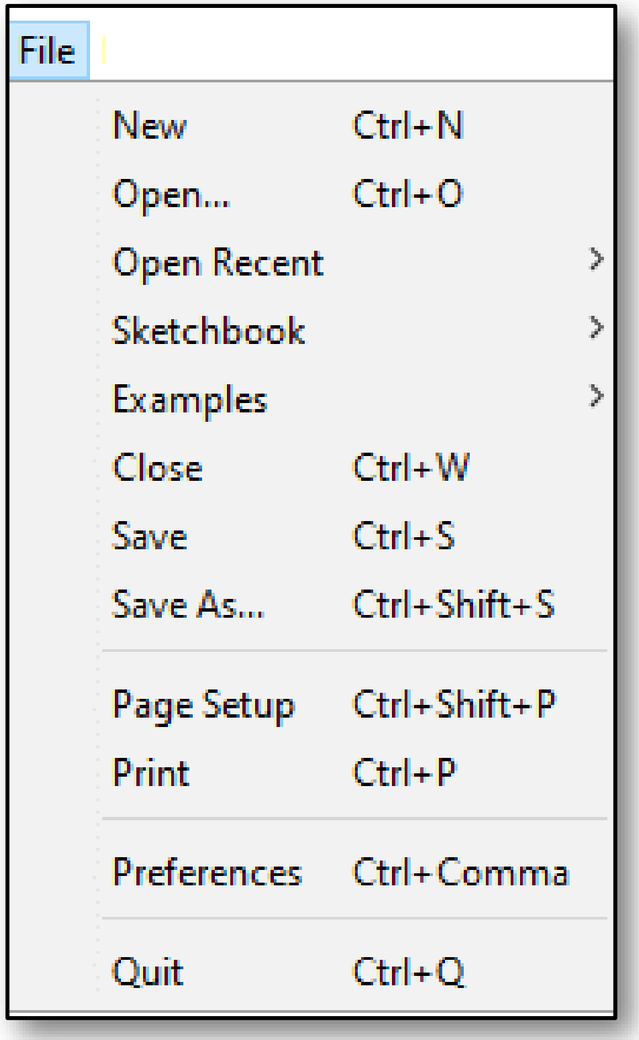
❖ Menu Bar

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- There are **File**, **Edit**, **Sketch**, **Tools**, **Help** menus present in this bar.

➢ File Menu:

When you click on the File option in the Menu bar, you will observe the following list of options:



- **New**: The New button is used to open the new window.
- **Open:** This option is used for opening an already existing file.
- **Open Recent**: This button is used to display the sketch files that were opened recently.
- **Sketchbook:** This option is used to store the current sketches created in the Arduino IDE software by creating an instance in a new editor.

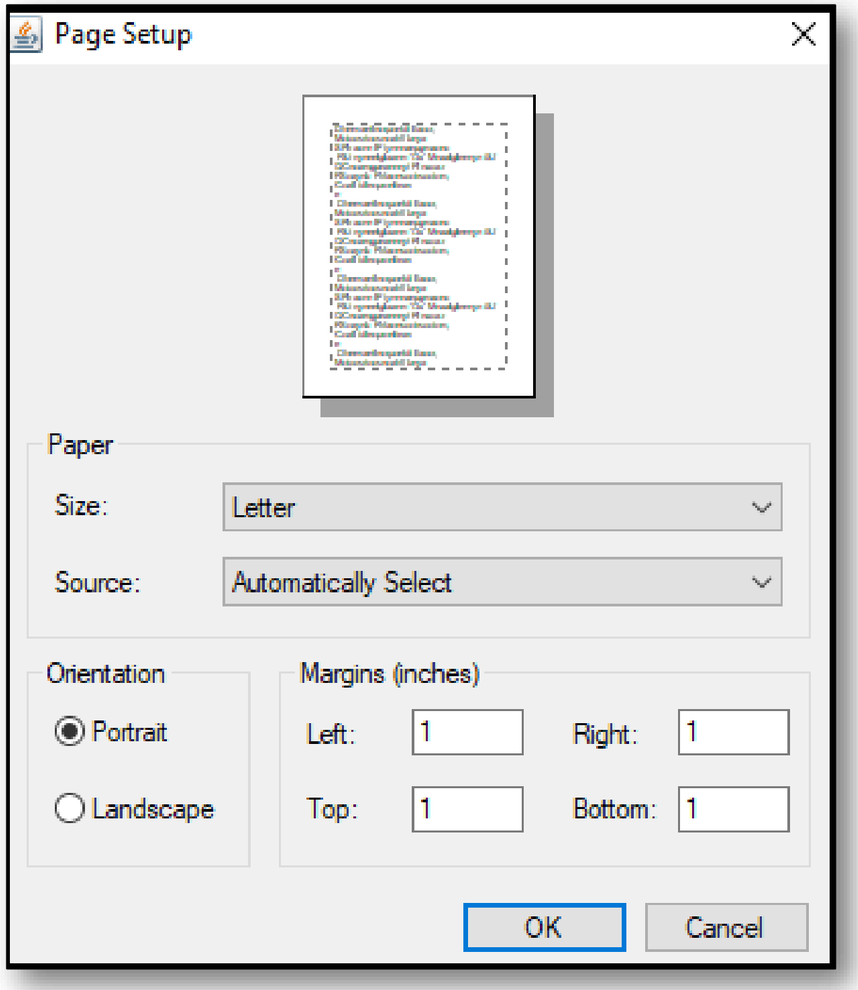
YouTube @zedkinginstitute Subscribe

- **Examples:** This is a useful feature in arduino since it is used for looking at some small projects for basic understanding.
- **Close**: This button is used to close the window(tab) when clicked.
- **Save**: When you have made changes to your current sketch file and want to save the changes then you can use the save option.
- **Save As**: If you want to save the file with a new name and at a different location, then we can use this button and write the new name in the box.
- **Page Setup**: Page Setup is used for setting up the page options by setting certain properties like page margins, orientation, and size for printing.

YouTube @zedkinginstitute Subscribe

Here is what the **'Page Setup'** looks like:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

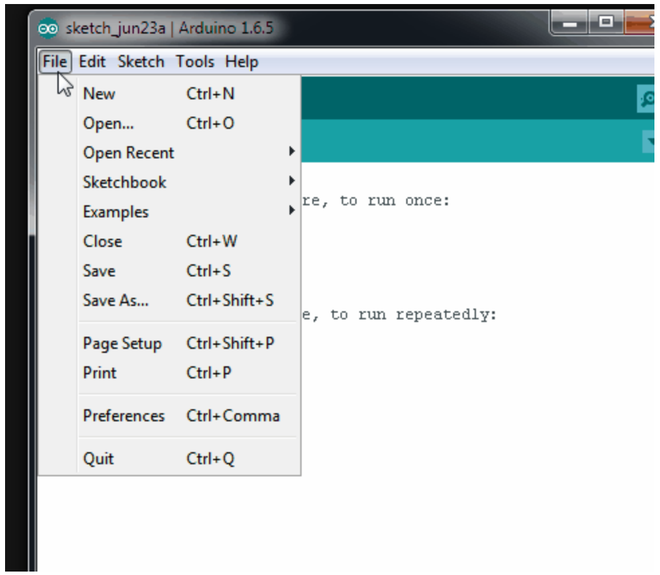


YouTube @zedkinginstitute Subscribe

Here is what the '**Page Setup**' looks like:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **Print**: Once you have selected the options in 'Page Setup', you can go ahead and print the original hardcopy using this button.
- **Preferences**: It is used to set preferences by customizing the settings of the Arduino IDE.
- **Quit**: As the name suggests, the button is used to exit the program and close all the IDE windows that are currently open.

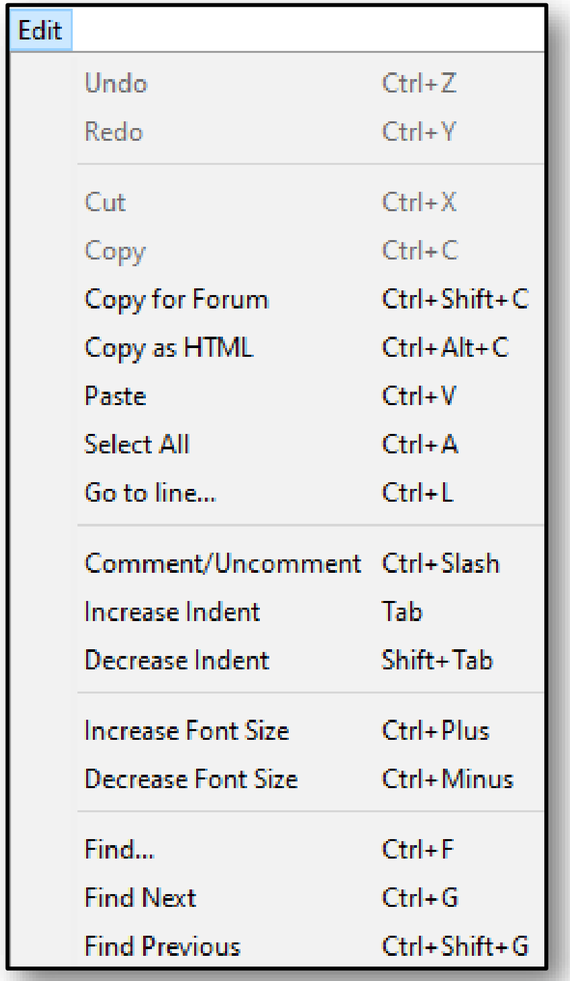


YouTube @zedkinginstitute Subscribe

➢ Edit Menu:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **Undo:** Incase you want to reverse the last action you performed then you can use the Undo button which will undo the action you made.
- **Redo**: Incase you want to redo the last action you performed then you can use the Redo button which will redo the action you made.
- **Cut**: Incase we want to remove a specific piece of text from the written code, you can use the cut option. After selecting the cut option, the text will be saved on the clipboard temporarily and you can further paste it to a new location.



YouTube @zedkinginstitute Subscribe

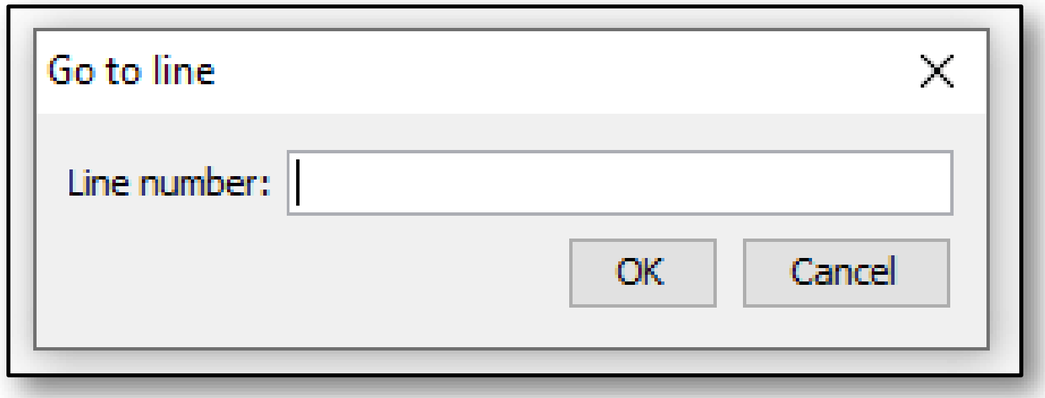
- **Copy**: Incase we want to duplicate a specific piece of text from the written code, you can use the copy option. After selecting the copy option, the text will be saved on the clipboard temporarily and you can further paste it to a new location.

- **Copy for Forum**: The 'Copy for Forum' button is used to copy the selected text to the clipboard, which is also suitable for posting to the forum.

- **Copy as HTML**: It is used to copy the selected text as HTML to the clipboard. It is desirable for embedding in web pages.

YouTube @zedkinginstitute Subscribe

- **Paste**: Incase you want to paste a specific piece of text from the written code, you can use the paste option. The text will be pasted from the clipboard to a new location.
- **Select All**: It is used to select all the text from the sketch at once.
- **Go to line**: It is a shortcut to move the cursor to the specific line location. Here is the dialog box.



YouTube @zedkinginstitute Subscribe

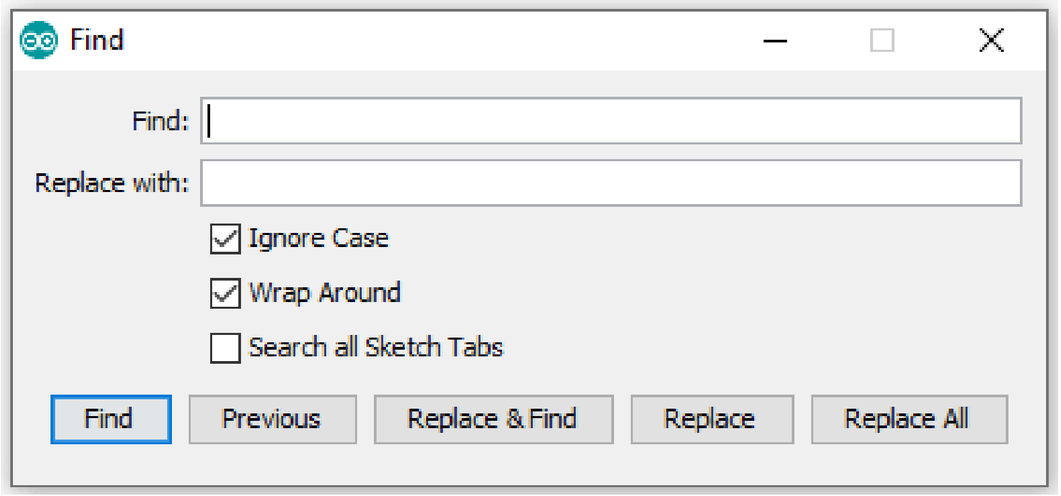
Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **Comment/Decomment**
  The Comment/ Decomment button is used to put or remove the comment mark (**//**) at the beginning of the specified line.
- **Increase Indent**
  It is used to add the space at the starting of the specified line. The spacing moves the text towards the right.
- **Decrease Indent**
  It is used to subtract or remove the space at the starting of the specified line. The spacing moves the text towards the left.
- **Increase Font Size**
  It increases the font size of the written text.

YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **Decrease Font Size**
  It decreases the font size of the written text.
- **Find...**
  It is used to find the specified text. We can also replace the text. It highlights the text in the sketch.



- **Find Next**
  It highlights the next word, which has specified in the
- **Find...**' window. If there is no such word, it will not show any highlighted text.

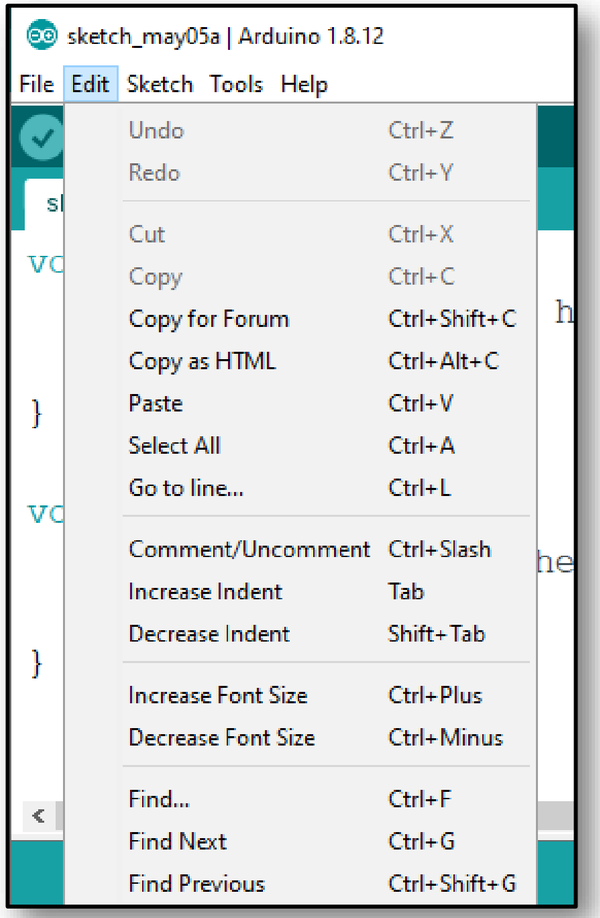
YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **Find Previous:**

It highlights the previous word, which has specified in the **Find...'** window. If there is no such word, it will not show any highlighted text.



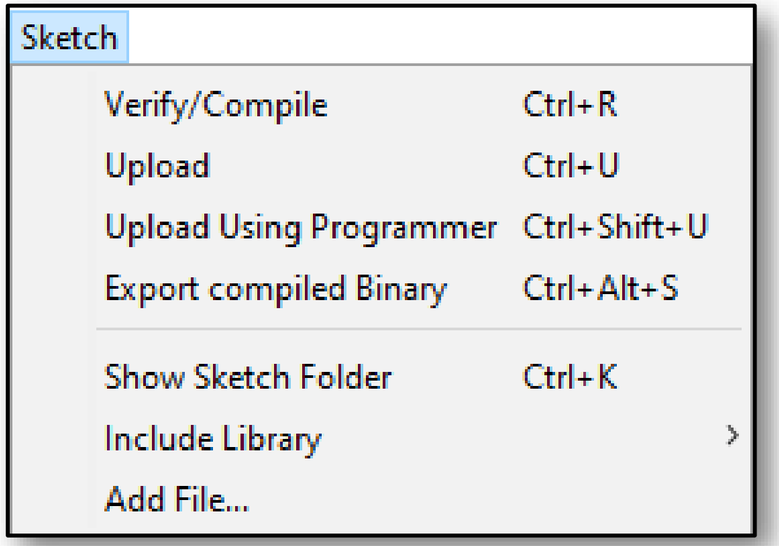


YouTube @zedkinginstitute Subscribe

➢ Sketch Menu:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

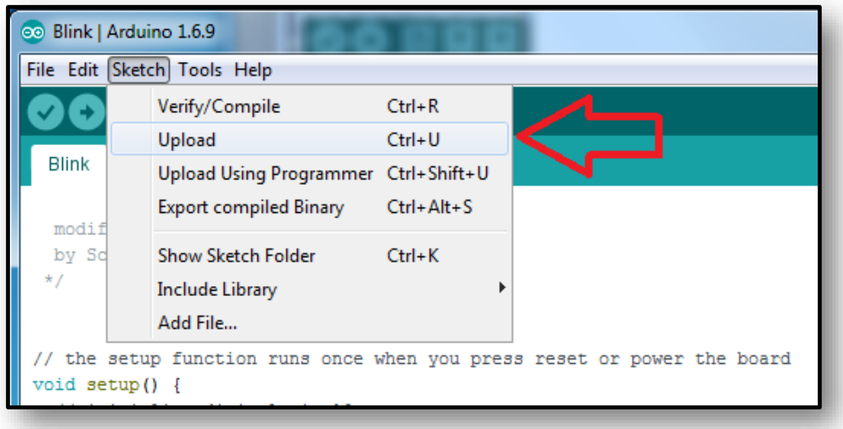
- **Verify/Compile:** It will check for the errors in the code while compiling. The memory in the console area is also reported by the IDE.
- **Upload:** The Upload button is used to configure the code to the specified board through the port.
- **Upload Using Programmer:**It is used to override the Bootloader that is present on the board. We can utilize the full capacity of the Flash memory using the '**Upload Using Programmer**' option.
- **Export compiled Binary:**It allows saving a **.hex** file and can be kept archived. Using other tools, .hex file can also be sent to the board.



YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **Show Sketch Folder:**
  It opens the folder of the current code written or sketch.
- **Include Library:**
  Include Library includes various Arduino libraries. The libraries are inserted into our code at the beginning of the code starting with the #. We can also import the libraries from **.zip** file.
- **Add File:**
  The Add File... button is used to add the created file in a new tab on the existing file.



YouTube @zedkinginstitute Subscribe

➢ Tools Menu:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **Auto Format:**

  The Auto Format button is used to format the written code. For example, lining the open and closed curly brackets in the code.

- **Archive Sketch:**

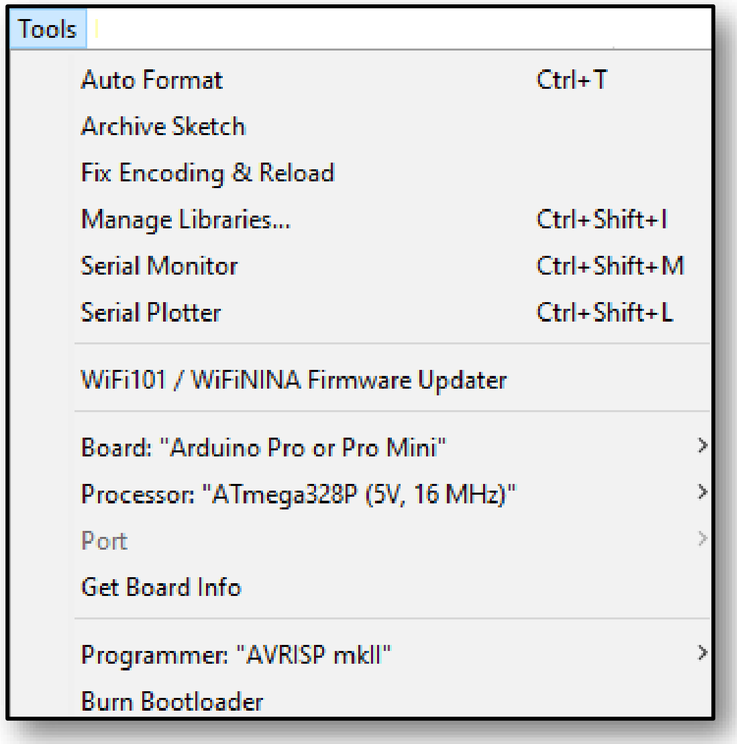
  The copy of the current sketch or code is archived in the .zip format.

- **Fix Encoding and Reload:**

  This button is used to fix the inconsistency between the operating system char maps and editor char map encoding.

- **Manage Libraries...**

  It shows the updated list of all the installed libraries. We can also use this option to install a new library into the Arduino IDE.



YouTube @zedkinginstitute Subscribe

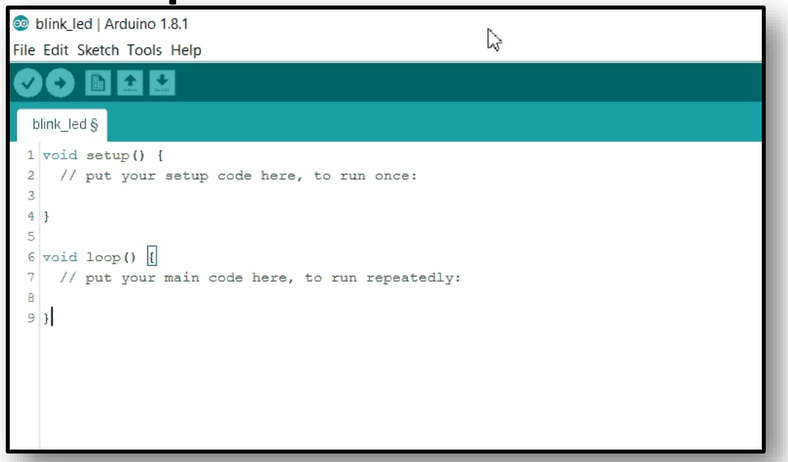
Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **Serial Monitor:**
  It allows the exchange of data with the connected board on the port.
- **Serial Plotter:** The Serial Plotter button is used to display the serial data in a plot. It comes preinstalled in the Arduino IDE.
- **WiFi101/WiFiNINA Firmware Updater:** It is used to check and update the Wi-Fi Firmware of the connected board.
- **Board:** We are required to select the board from the list of boards. The selected board must be similar to the board connected to the computer.
- **Processor:** It displays the processor according to the selected board. It refreshes every time during the selection of the board.
- **Port:** It consists of the virtual and real serial devices present on our machine.

YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **Get Board Info:**
  It gives the information about the selected board.
- **Programmer:** We need to select the hardware programmer while programming the board. It is required when we are not using the onboard USB serial connection. It is also required during the burning of the Bootloader.
- **Burn Bootloader:** The Bootloader is present on the board onto the microcontroller. The option is useful when we have purchased the microcontroller without the bootloader.
  Before burning the bootloader, we need to make sure about the correct selected board and port.

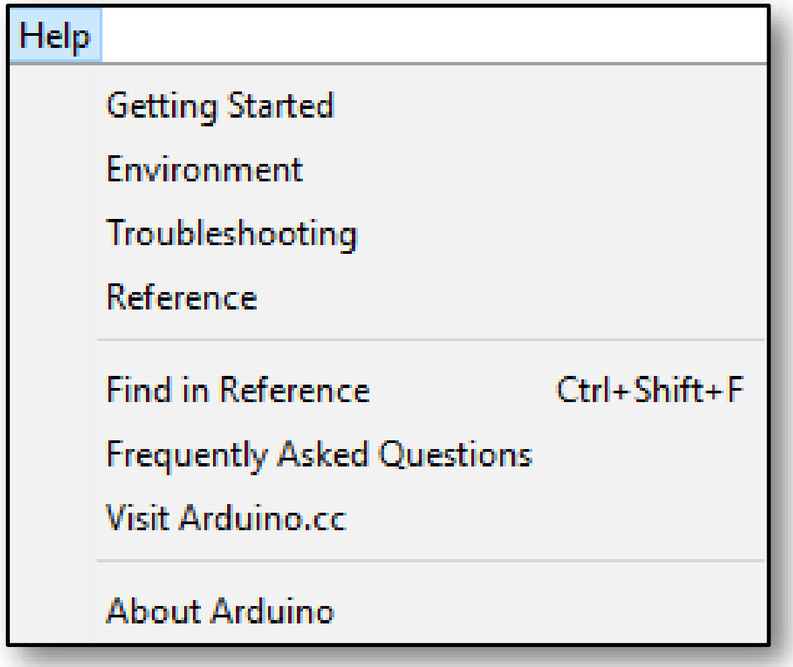


YouTube @zedkinginstitute Subscribe

➢ Help Menu:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

The Help section includes several documents that are easy to access, which comes along with the Arduino IDE. It consists of the number of options such as Getting Started, Environment, Troubleshooting, Reference, etc. We can also consider the image shown above, which includes all the options under the Help section.

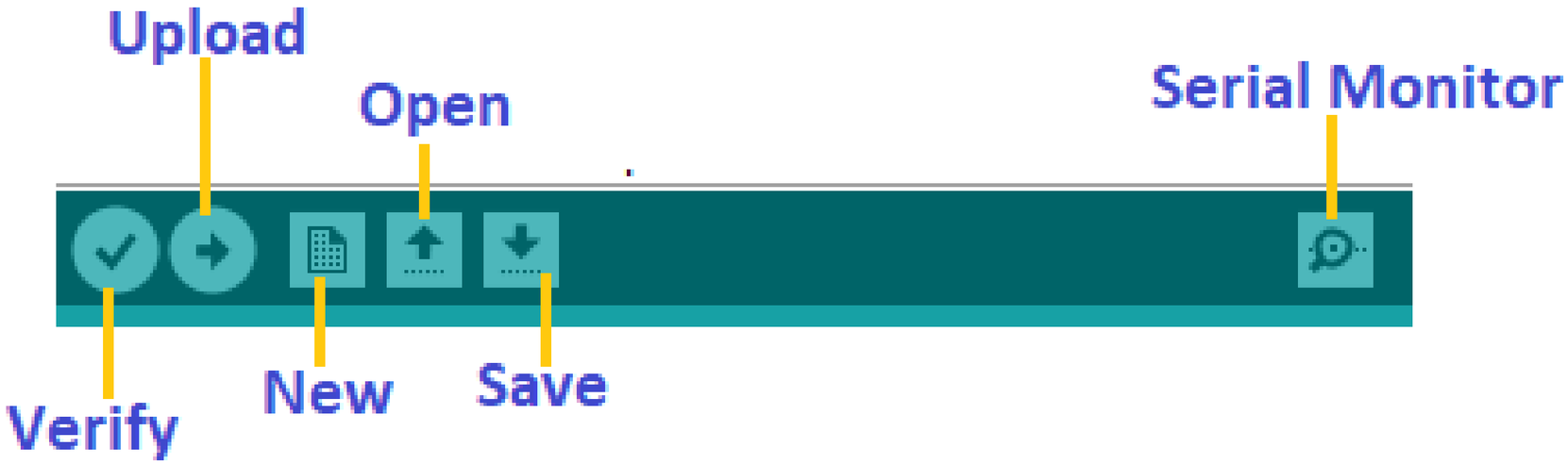


YouTube @zedkinginstitute
Subscribe

❖ Toolbar Button

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- The icons displayed on the toolbar are **New**, **Open**, **Save**, **Upload**, and **Verify**.



➢ **Upload:**

The Upload button compiles and runs our code written on the screen. Before uploading the sketch, we need to make sure that the correct board and ports are selected. If the uploading is failed, it will display the message in the **error** window.

YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **Open**

The Open button is used to open the already created file. The selected file will be opened in the current window.

- **Save**

The save button is used to save the current sketch or code.

- **New**

It is used to create a new sketch or opens a new window.

- **Verify**

The Verify button is used to check the compilation error of the sketch or the written code.

- **Serial Monitor** 

The serial monitor button is present on the right corner of the toolbar. It opens the serial monitor.

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

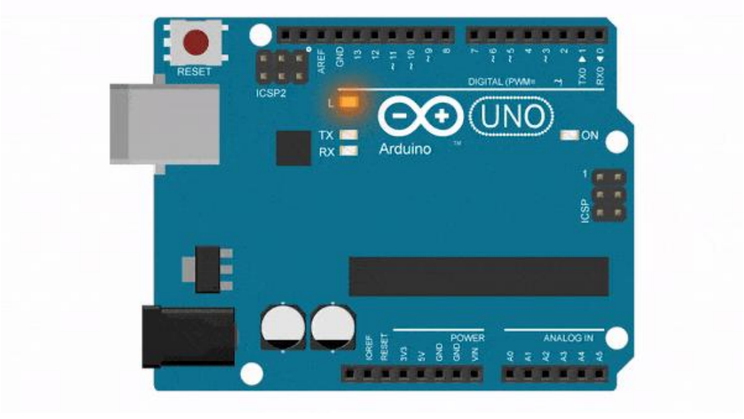
Chapter 4 -
Building of IOT Applications

Class-3

❖ Arduino Coding Basics

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- We have already discussed the popular Arduino Boards, Arduino IDEs, and Installation process of the Arduino software.
- We learned that Arduino IDE (Integrated Development Environment) allows us to draw the sketch and upload it to the various Arduino boards using code.
- The code is written in a simple programming language similar to **C** and **C++**.
- The initial step to start with Arduino is the IDE download and installation.
- Let's discuss the basics to start with Arduino programming.

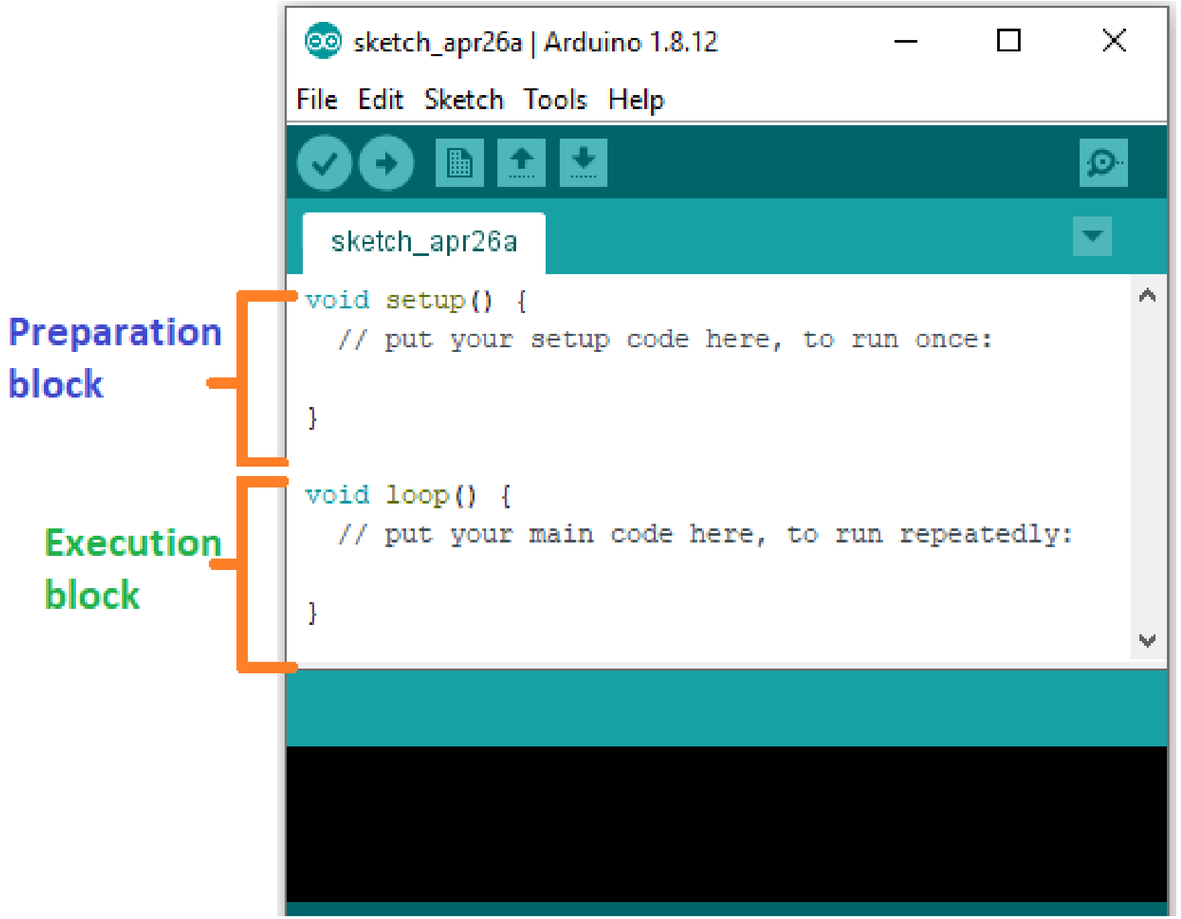


YouTube @zedkinginstitute Subscribe

❖ Coding Screen

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- The coding screen is divided into two blocks. The **setup** is considered as the preparation block, while the **loop** is considered as the execution block. It is shown below:



YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **What is Setup?**

It contains an initial part of the code to be executed. The pin modes, libraries, variables, etc., are initialized in the setup section. It is executed only once during the uploading of the program and after reset or power up of the Arduino board.

- **What is Loop?**

The loop contains statements that are executed repeatedly. The section of code inside the curly brackets is repeated depending on the value of variables.

YouTube @zedkinginstitute Subscribe

❖ Brackets

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- There are two types of brackets used in the Arduino coding, which are listed below:

Parentheses ( ) **Curly Brackets { }**

❑ Parentheses ( )

When writing a function in IDE, the parentheses brackets are used to include the argument parameters, such as methods, functions, or code statements. In addition to this, the bracket is also used for defining the precedence order while dealing with mathematical equations. These brackets are represented by '( )'.

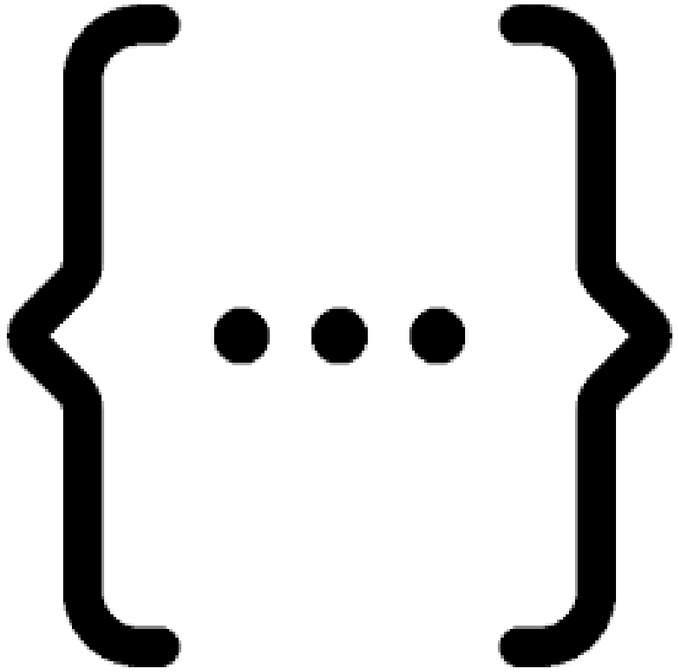
YouTube @zedkinginstitute Subscribe

❑ Curly Brackets { }

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Curly brackets are used to open and close all the statements in the functions or out of the functions. Note that a closed curly bracket always follows the open curly bracket in the code for proper layout. These brackets are denoted by '{ }'.

- **Open curly bracket- ' { '**
- **Closed curly bracket - ' } '**

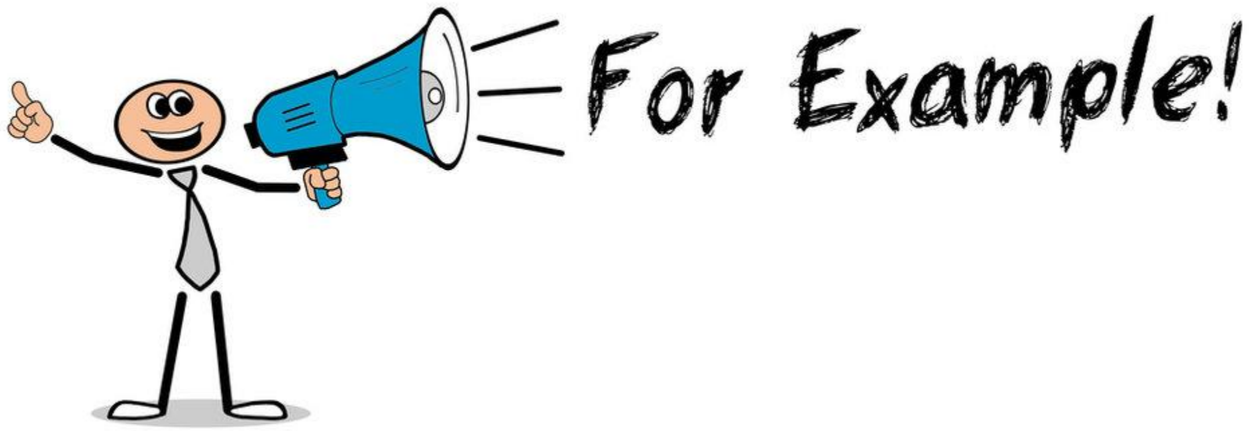


YouTube @zedkinginstitute Subscribe

❖ Statements within Curly Brackets

Zed-King GROUP OF INSTITUTIONS ISO 9001 : 2015 Certified

- The set of statements in the setup and loop blocks are enclosed with the curly brackets. We can write multiple statements depending on the coding requirements for a particular project.




```
void setup ( )
{
Coding statement 1;
Coding statement 2;
.
.
.
Coding statement n;
}
void loop ( )
{
Coding statement 1;
Coding statement 2;
.
.
.
Coding statement n;
}
```

YouTube @zedkinginstitute Subscribe

❖ Line Comment

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

There are two types of line comments, which are listed below:

- Single line comment
- Multi-line comment

// Single line comment

The text that is written after the two forward slashes are considered as a single line comment. The compiler ignores the code written after the two forward slashes. The comment will not be displayed in the output. Such text is specified for a better understanding of the code or for the explanation of any code statement.

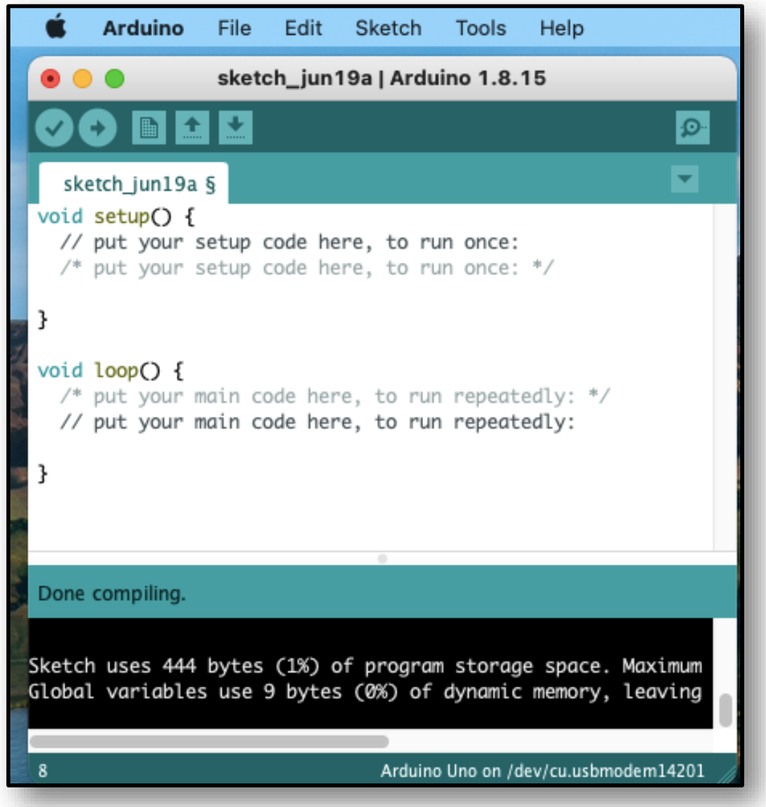
The // (two forward slashes) are also used to ignore some extra lines of code without deleting it.

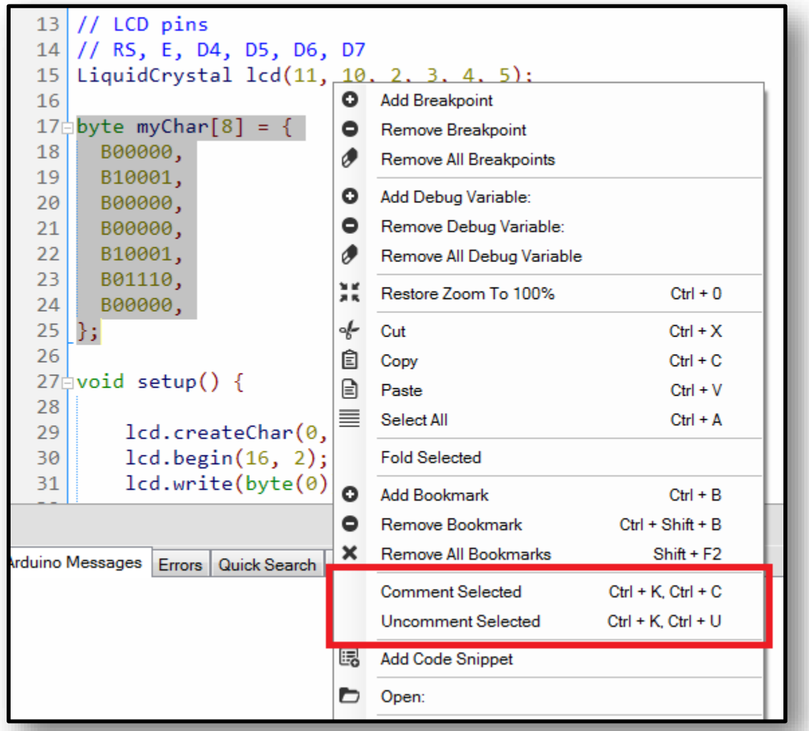
YouTube @zedkinginstitute Subscribe

/ * Multi - line comment */

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

The Multi-line comment is written to group the information for clear understanding. It starts with the single forward slash and an asterisk symbol (/ *). It also ends with the / *. It is commonly used to write the larger text. It is a comment, which is also ignored by the compiler.





YouTube @zedkinginstitute
Subscribe

❖ Arduino Serial

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **Serial Communication**

The serial communication is a simple scheme that uses the **UART** (Universal Asynchronous Receiver/Transmitter) on the Microcontroller. It uses,

- **5V for logic 1 (high)**
- **0V for logic 0 (low)**

For a 3.3V board, it uses

- **3V for logic 1 (high)**
- **0V for logic 0 (low)**

Every message sent on the UART is in the form of 8 bits or 1 byte, where **1 byte = 8 bits.**

YouTube @zedkinginstitute Subscribe

❖ Serial.begin()

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

The Serial.begin( ) sets the baud rate for serial data communication.
The **baud rate** signifies the data rate in bits per second.

The default baud rate in Arduino is **9600 bps (bits per second)**. We can specify other baud rates as well, such as 4800, 14400, 38400, 28800, etc.

The Serial.begin( ) is declared in two formats, which are shown below:
begin(speed)
begin(speed, config) **Where,**

- **serial:** It signifies the serial port object.
- **speed:** It signifies the baud rate or bps (bits per second) rate. It allows **long** data types.
- **config:** It sets the stop, parity, and data bits.

YouTube @zedkinginstitute Subscribe

❖ Example1.

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

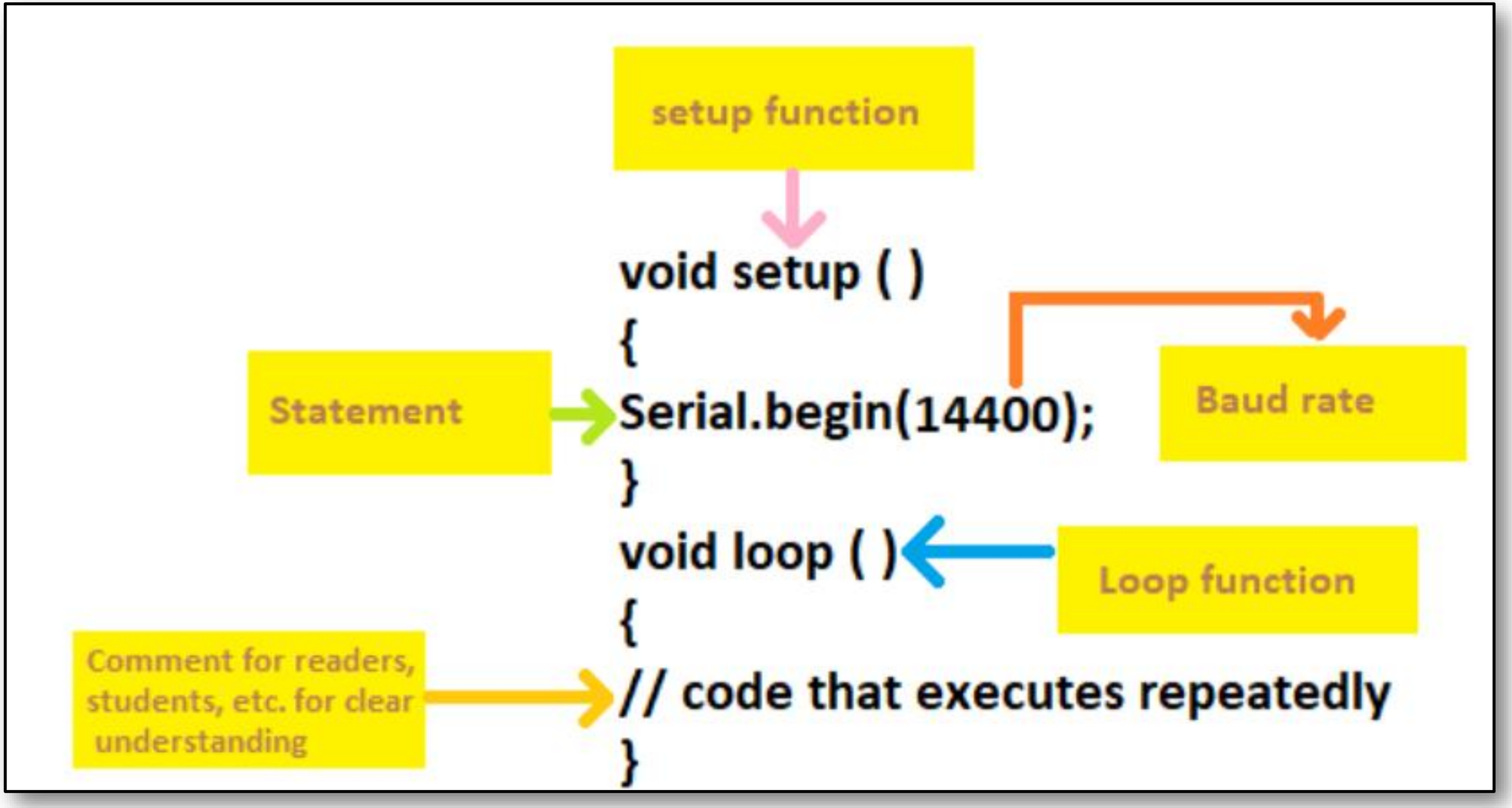
```
void setup ( )
{
Serial.begin(4800);
}
void loop ( )
{
}
```

The serial.begin (4800) open the serial port and set the bits per rate to 4800. The messages in Arduino are interchanged with the serial monitor at a rate of 4800 bits per second.

YouTube @zedkinginstitute Subscribe

❖ Example2.

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified



YouTube @zedkinginstitute Subscribe

❖ Serial.print()

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

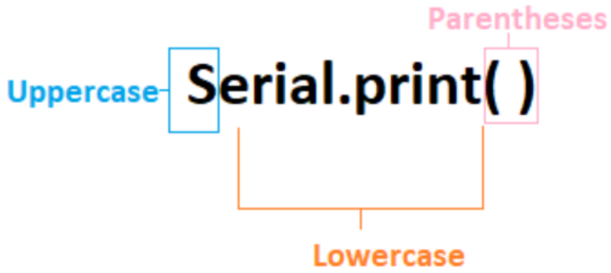
The serial.print ( ) in Arduino prints the data to the serial port. The printed data is stored in the ASCII (American Standard Code for Information Interchange) format, which is a human-readable text.

Each digit of a number is printed using the ASCII characters.

The printed data will be visible in the **serial monitor**, which is present on the right corner on the toolbar.

The Serial.print( ) is declared in two formats, which are shown below:

print(value)
print(value, format)



Note: In Serial.print( ), S must be written in uppercase.

YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Where,

- **serial:** It signifies the serial port object.
- **print:** The print ( ) returns the specified number of bytes written.
- **value:** It signifies the value to print, which includes any data type value.
- **format:** It consists of number base, such as **OCT** (Octal), **BIN** (Binary), **HEX** (Hexadecimal), etc. for the integral data types. It also specifies the number of decimal places.

❖ Serial.print(value)

The serial.print ( ) accepts the number using the ASCII character per digit and value upto two decimal places for floating point numbers.

YouTube @zedkinginstitute Subscribe

Example 1:

```
Serial.print(15.452732)
```

Output:

15.45

It sends bytes to the printer as a single character. In Arduino, the strings and characters using the Serial.print( ) are sent as it is.

Example 2:

```
Serial.print("Hello Arduino")
```

Output:

"Hello Arduino"

Zed-King GROUP OF INSTITUTIONS ISO 9001 : 2015 Certified

YouTube @zedkinginstitute Subscribe

❖ Serial.print(value, format)

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

It specifies the base format and gives the output according to the specified format. It includes the formats Octal -OCT (base 8), Binary-BIN (base 2), Decimal-DEC (base 10), and Hexadecimal-HEX (base 16).
Let's understand by few examples.

```
Serial.print(25, BIN)
```

Output:

11001

```
Serial.print(58, HEX)
```

Output:

3A

```
Serial.print(58, OCT)
```

YouTube @zedkinginstitute Subscribe

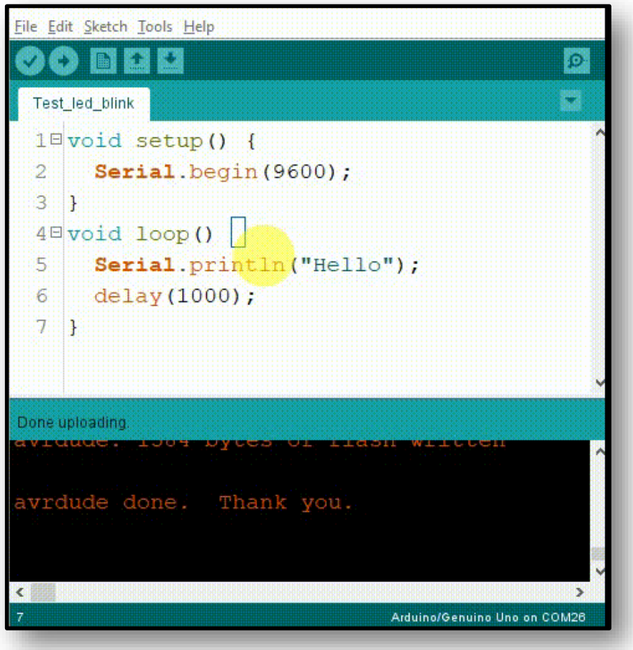
❖ Serial.println()

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

The Serial.println ( ) means print line, which sends the string followed by the carriage return ('\r' or ASCII 13) and newline ('\n' or ASCII 10) characters. It has a similar effect as pressing the Enter or Return key on the keyboard when typing with the Text Editor.

The Serial.println( ) is also declared in two formats, which are shown below:

- println(value)
- println(value, format)



YouTube @zedkinginstitute Subscribe

❖ Difference b/w Serial.print() & Serial.println()

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

The Serial.println ( ) means print line, which sends the string followed by the carriage return ('\r' or ASCII 13) and newline ('\n' or ASCII 10) characters. It has a similar effect as pressing the Enter or Return key on the keyboard when typing with the Text Editor.

The Serial.println( ) is also declared in two formats, which are shown below:

- println(value)
- println(value, format)

YouTube @zedkinginstitute Subscribe

🔷 Task 1: Using `Serial.print()` and `Serial.println()`

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Question:

Write a program that prints the following on the Serial Monitor:

- `"Welcome to Arduino"` using `Serial.println()`
- `"Temperature: "` using `Serial.print()`
- A value (e.g., 25) using `Serial.print()`
- `" °C"` using `Serial.println()`

Expected Output:

```vbnet
Welcome to Arduino
Temperature: 25 °C
```

YouTube @zedkinginstitute Subscribe

🔷 Task 2: Printing a Number in Decimal, Hexadecimal, and Octal

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Question:

Declare an integer `num = 100`. Print this number in:

- Decimal format
- Hexadecimal format
- Octal format

Each on a new line using `Serial.println()` and format specifiers (`DEC`, `HEX`, `OCT`).

Expected Output:

```
Decimal: 100
Hexadecimal: 64
Octal: 144
```

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-3

❖ Data types in C (used in Arduino)

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

They define what kind of data a variable holds and how much memory it uses.

Data type	Size(bytes)	Range
char	1	-128 to 127
unsigned char	1	0 to 255
short	2	-32,768 to 32,767
unsigned short	2	0 to 65535
int	4	-2147483648 to +2147483647
unsigned int	4	0 to 4294967295
long	4	-2147483648 to +2147483647
Unsinged long	4	0 to 4294967295
float	4	-3.4e-38 to +3.4e-38
double	8	1.7 e-308 to 1.7 e+308
long double	8	1.7 e-308 to 1.7 e+308
bool	1 bit	
void	-	-
wchar_t	2 or 4	1 wide character

YouTube @zedkinginstitute Subscribe

❖ Integer Data Type

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **INTEGER:** The integer data type in C is used to store the integer numbers (any number including positive, negative and zero without decimal part). Octal values, hexadecimal values, and decimal values can be stored in int data type in C.
- The keyword used to define integers is **int**

Range: -2,147,483,648 to 2,147,483,647
Size: 4 bytes
Format Specifier: %d

YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

The integer data type can also be used as:

- **unsigned int:** Unsigned int data type in C is used to store the data values from zero to positive numbers but it can't store negative values like signed int.
- **short int:** It is lesser in size than the int by 2 bytes so can only store values from -32,768 to 32,767.
- **long int:** Larger version of the int datatype so can store values greater than int.
- **unsigned short int:** Similar in relationship with short int as unsigned int with int.

Note: *The size of an integer data type is compiler-dependent. We can use sizeof operator to check the actual size of any data type.*

YouTube @zedkinginstitute Subscribe

❖ Example of Integer Data Type:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```c
// C program to print Integer data types.
#include <stdio.h>

int main()
{
    // Integer value with positive data.
    int a = 9;

    // integer value with negative data.
    int b = -9;

    // U or u is Used for Unsigned int in C.
    int c = 89U;

    // L or l is used for long int in C.
    long int d = 99998L;

    printf("Integer value with positive data: %d\n", a);
    printf("Integer value with negative data: %d\n", b);
    printf("Integer value with an unsigned int data: %u\n",
           c);
    printf("Integer value with an long int data: %ld", d);

    return 0;
}
```

Output

```
Integer value with positive data: 9
Integer value with negative data: -9
Integer value with an unsigned int data: 89
Integer value with an long int data: 99998
```

YouTube @zedkinginstitute Subscribe

❖ Example of Integer Data Type:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```c
// C program to print Integer data types.
#include <stdio.h>

int main()
{
    // Integer value with positive data.
    int a = 9;

    // integer value with negative data.
    int b = -9;

    // U or u is Used for Unsigned int in C.
    int c = 89U;

    // L or l is used for long int in C.
    long int d = 99998L;

    printf("Integer value with positive data: %d\n", a);
    printf("Integer value with negative data: %d\n", b);
    printf("Integer value with an unsigned int data: %u\n",
           c);
    printf("Integer value with an long int data: %ld", d);

    return 0;
}
```

Output

```
Integer value with positive data: 9
Integer value with negative data: -9
Integer value with an unsigned int data: 89
Integer value with an long int data: 99998
```

YouTube @zedkinginstitute Subscribe

❖ Float

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

FLOATING POINT: In C programming float data type is used to store floating-point values. Float in C is used to store decimal and exponential values. It is used to store decimal numbers (numbers with floating point values) with single precision.

- The keyword used to define float variables is **float**

Range: 1.2E-38 to 3.4E+38
Size: 4 bytes
Format Specifier: %f

YouTube @zedkinginstitute Subscribe

❖ Example of Float Data Type:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```c
// C Program to demonstrate use
// of Floating types
#include <stdio.h>

int main()
{
    float a = 9.0f;
    float b = 2.5f;

    // 2x10^-4
    float c = 2E-4f;
    printf("%f\n", a);
    printf("%f\n", b);
    printf("%f", c);

    return 0;
}
```

Output

```
9.000000

2.500000

0.000200
```

YouTube @zedkinginstitute Subscribe

❖ Double

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **DOUBLE**: These are floating point numbers, both positive and negative, which have a higher precision than float variables.
- The keyword used to define double variables is **double**

Range: 1.7E-308 to 1.7E+308
Size: 8 bytes
Format Specifier: %lf

```c
// C Program to demonstrate
// use of double data type
#include <stdio.h>

int main()
{
    double a = 123123123.00;
    double b = 12.293123;
    double c = 2312312312.123123;

    printf("%lf\n", a);

    printf("%lf\n", b);

    printf("%lf", c);

    return 0;
}
```

Output

```
123123123.000000
12.293123
2312312312.123123
```

YouTube @zedkinginstitute Subscribe

❖ Char

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **Character**: Character data type allows its variable to store only a single character. The size of the character is 1 byte.
- The keyword used to define character variables is **char.**

Range: (-128 to 127) or (0 to 255)
Size: 1 byte
Format Specifier: %c

```
Value of a: a
Value of a after increment is: b
Value of c: c
```

```c
// C program to print Integer data types.
#include <stdio.h>

int main()
{
    char a = 'a';
    char c;

    printf("Value of a: %c\n", a);

    a++;
    printf("Value of a after increment is: %c\n", a);

    // c is assigned ASCII values
    // which corresponds to the
    // character 'c'
    // a-->97 b-->98 c-->99
    // here c will be printed
    c = 99;

    printf("Value of c: %c", c);

    return 0;
}
```

YouTube @zedkinginstitute Subscribe

❖void data type

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- The void data type in C is used to specify that no value is present. It does not provide a result value to its caller. It has no values and no operations. It is used to represent nothing. Void is used in multiple ways as function return type, function arguments as void, and pointers to void.

Syntax:

```
// function return type void
void exit(int check);
// Function without any parameter can accept void.
int print(void);
// memory allocation function which
// returns a pointer to void.
void *malloc (size_t size);
```

YouTube @zedkinginstitute Subscribe

❖Size of Data Types in C

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- The size of the data types in C is dependent on the size of the architecture, so we cannot define the universal size of the data types. For that, the C language provides the **sizeof()** operator to check the size of the data types.

```
void setup() {
 Serial.begin(115200);
 Serial.print(sizeof(23));

}

void loop() {


}
```

YouTube @zedkinginstitute Subscribe

❖ What is ASCII Value of a Character in C

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- American Standard Code for Information Interchange (ASCII) is a character encoding standard that assigns a unique numerical value to all characters including special symbols. In C programming, the ASCII value of the character is stored instead of the character itself.
- Each character or special character is represented by some ASCII code.
- Each ASCII code occupies 7 bits in memory.
- Each character variable is assigned an ASCII value ranging from 0 to 127

Input: *char ch = 'A';*
Output: *ASCII value of A is 65*
Explanation: *The character 'A' has an ASCII value of 65.*

Input: *char ch = 'z';*
Output: *ASCII value of z is 122*
Explanation: *The character 'z' has an ASCII value of 122.*

YouTube @zedkinginstitute Subscribe

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified



YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

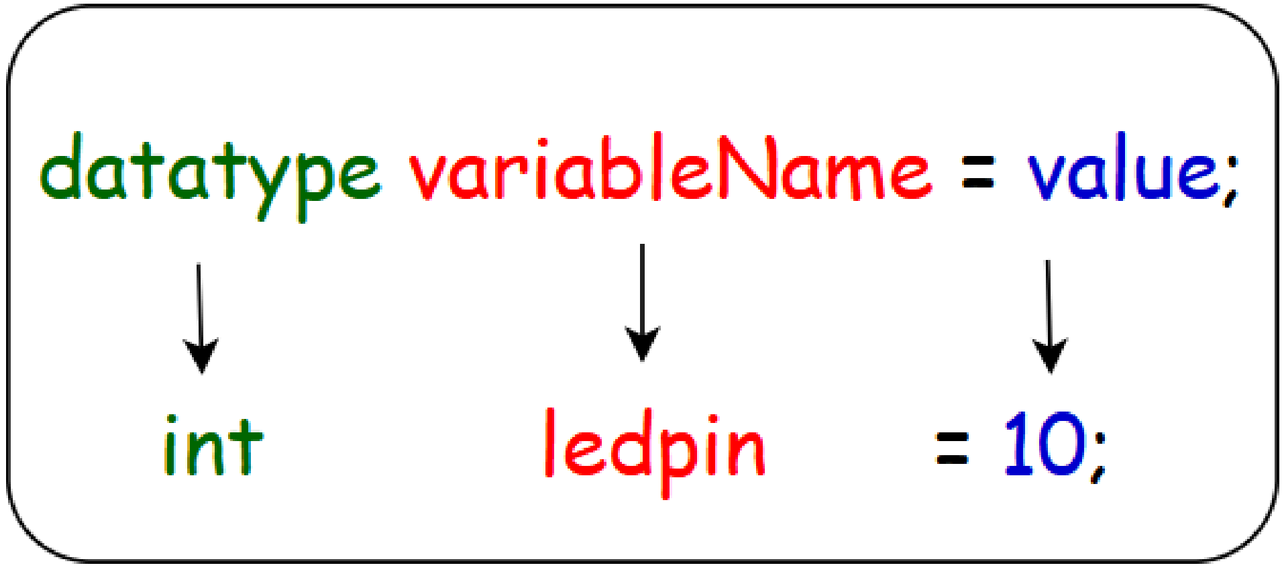
Chapter 4 -
Building of IOT Applications

Class-4

❖ Variables

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Variables are defined as named containers that are used to store data. They are used to hold various data, such as numbers, characters, and even more complex types like arrays or structures. The basic syntax for declaring a variable in Arduino is:



Syntax for declaring a variable

YouTube @zedkinginstitute Subscribe

❖Rules of Variables

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- Identifiers in C **must begin with a character or underscore**, and may be followed by any combination of characters, underscores, or the digits 0-9.

summary	**exit_flag**	**i**
Jerry7	**Number_of_moves**	**_id**

- You should ensure that you use **meaningful (but short) names** for your identifiers. The reasons for this are to make the program easier to read and self-documenting. Example:

distance = speed * time;

- Some users choose to adopt the **convention** that variable names are all lower case while symbolic names for constants are all upper case.
- **Keywords** are reserved identifiers that have strict meaning to the C compiler. C only has 29 keywords. Example keywords are:

if, else, char, int, while

YouTube @zedkinginstitute Subscribe

❖ Variables Scope

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

In Arduino programming, variables can have different scopes based on where in the code they are declared.

- Inside a function or a block, which is called **local variables**.
- In the definition of function parameters, which is called **formal parameters**.
- Outside of all functions, which is called **global variables**.

YouTube @zedkinginstitute Subscribe

❖ Local Variables

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

The variables that are defined within a function or block of code are termed as local variables. They are only accessible within the specified block of code.

```
Void setup () {

}

Void loop () {
   int x , y ;
   int z ; Local variable declaration
   x = 0;
   y = 0; actual initialization
   z = 10;
}
```

YouTube @zedkinginstitute Subscribe

❖ Global Variables

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Global Variables are declared outside of all functions, typically at the beginning of the sketch. They have a global scope, meaning they can be accessed from any part of the code, including inside functions.

```
Int T , S ;
float c = 0 ; Global variable declaration

Void setup () {

}

Void loop () {
   int x , y ;
   int z ; Local variable declaration
   x = 0;
   y = 0; actual initialization
   z = 10;
}
```

YouTube @zedkinginstitute Subscribe

❖ Formal parameters:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Formal parameters are variables declared in the function's definition and act as placeholders for values that are passed to the function when it's called.

```
1  void printNumber(int num) {
2      // num is a formal parameter
3      Serial.println(num);
4  }
```

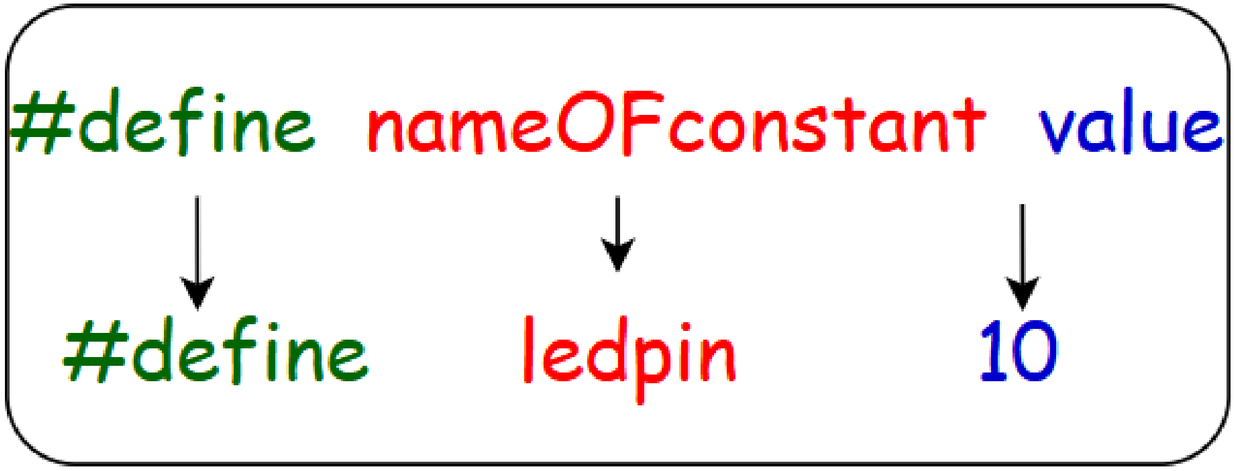
YouTube @zedkinginstitute Subscribe

❖ Constants

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Constants are like variables, but their value remains constant throughout the program's execution. They are used to represent fixed values that should not be altered during the program's runtime. The constants defined in the Arduino program do not occupy any program memory.

These constants are introduced with the **#define** keyword.



Syntax for declaring a variable

YouTube @zedkinginstitute Subscribe

❖ const keyword

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Arduino allows you to define constants using the **const** keyword. These constants are used to represent fixed values and are typically used for things like pin numbers or calibration factors.

```
1   const int ledPin = 13;
2   const float pi = 3.14159;
```

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-5

❖What is Operator?

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- An operator is a symbol that instructs the compiler to perform a specific mathematical or logical operation on operands



YouTube @zedkinginstitute Subscribe

❖ Assignment Operators

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

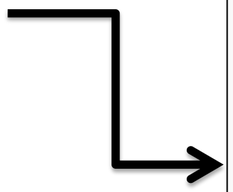
Operator	Description	Example	Result
=	Assigns value to a variable.	x = 10	x = 10
+=	Adds right operand to the left operand and assigns the result to the left operand.	x += 3	x = 13
-=	Subtracts right operand from the left operand and assigns the result to the left operand.	x -= 3	x = 7
*=	Multiplies left operand by the right operand and assigns the result to the left operand.	x *= 3	x = 30
/=	Divides left operand by the right operand and assigns the result to the left operand.	x /= 3	x = 3.3333
//=	Performs floor division on the left operand by the right operand and assigns the result to the left operand.	x //= 3	x = 3
%=	Takes the modulus of the left operand by the right operand and assigns the result to the left operand.	x %= 3	x = 1
**=	Raises the left operand to the power of the right operand and assigns the result to the left operand.	x **= 3	x = 1000

YouTube @zedkinginstitute Subscribe

❖ Example

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  int x = 10;
  Serial.println("Initial value of x: " + String(x));

  x += 5;
 Serial.println("After x += 5: " + String(x));

  x -= 3;
 Serial.println("After x -= 3: " + String(x));

  x *= 2;
 Serial.println("After x *= 2: " + String(x));

  x /= 4;
 Serial.println("After x /= 4: " + String(x));

  x %= 3;
 Serial.println("After x %= 3: " + String(x));
}

void loop() {  }
```



```
Initial value of x: 10
After x += 5: 15
After x -= 3: 12
After x *= 2: 24
After x /= 4: 6
After x %= 3: 0
```

YouTube @zedkinginstitute Subscribe

❖ Arithmetic Operator

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

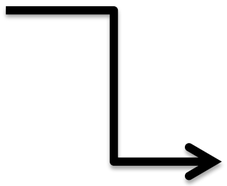
Operator	Description	Example	Result
+	Addition: Adds two operands.	5 + 3	8
-	Subtraction: Subtracts second operand from the first.	5 - 3	2
*	Multiplication: Multiplies two operands.	5 * 3	15
/	Division: Divides first operand by second. Returns float.	5 / 3	1.6667
//	Floor Division: Divides first operand by second, rounds down to nearest integer.	5 // 3	1
%	Modulus: Returns the remainder of the division.	5 % 3	2
**	Exponentiation: Raises the first operand to the power of the second.	5 ** 3	125

YouTube @zedkinginstitute Subscribe

❖ Example

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  int a = 20;
  int b = 6;

  Serial.println("Arithmetic Operations:");

  int sum = a + b;
  Serial.println("a + b = " + String(sum));

  int difference = a - b;
  Serial.println("a - b = " + String(difference));

  int product = a * b;
  Serial.println("a * b = " + String(product));

  int quotient = a / b;
  Serial.println("a / b = " + String(quotient));

  int remainder = a % b;
  Serial.println("a % b = " + String(remainder));
}

void loop() { }
```



```
Arithmetic Operations:
a + b = 26
a - b = 14
a * b = 120
a / b = 3
a % b = 2
```

YouTube @zedkinginstitute Subscribe

❖ Comparison Operator

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

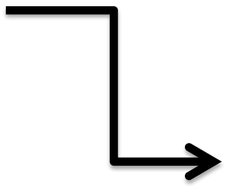
Operator	Description	Example	Result
==	Equal to: Checks if two operands are equal.	5 == 5	True
!=	Not equal to: Checks if two operands are not equal.	5 != 3	True
>	Greater than: Checks if left operand is greater than right.	5 > 3	True
<	Less than: Checks if left operand is less than right.	5 < 3	False
>=	Greater than or equal to: Checks if left operand is greater than or equal to the right.	5 >= 3	True
<=	Less than or equal to: Checks if left operand is less than or equal to right.	5 <= 3	False

YouTube @zedkinginstitute Subscribe

❖ Example

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  int a = 10;
  int b = 15;

  Serial.println("Comparison Operations:");

  Serial.print("a == b: ");
  Serial.println(a == b); // false -> 0

  Serial.print("a != b: ");
  Serial.println(a != b); // true  -> 1

  Serial.print("a > b: ");
  Serial.println(a > b);  // false -> 0

  Serial.print("a < b: ");
  Serial.println(a < b);  // true  -> 1

  Serial.print("a >= b: ");
  Serial.println(a >= b); // false -> 0

  Serial.print("a <= b: ");
  Serial.println(a <= b); // true  -> 1
}

void loop() {   }
```



```
Comparison Operations:
a == b: 0
a != b: 1
a > b: 0
a < b: 1
a >= b: 0
a <= b: 1
```

YouTube @zedkinginstitute Subscribe

🔷 Task 1: Assignment Operator

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Question:

Write a program that assigns the value `25` to a variable `a` and prints it to the Serial Monitor.

Expected Output:

```
CSS

Value of a: 25
```

YouTube @zedkinginstitute Subscribe

🔷 Task 2: Arithmetic Operators

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Question:

Declare two integers `x = 10` and `y = 5`. Calculate and display the results of the following operations on the Serial Monitor:

- Addition
- Subtraction
- Multiplication
- Division
- Modulo

Expected Output:

```matlab
x + y = 15
x - y = 5
x * y = 50
x / y = 2
x % y = 0
```

YouTube @zedkinginstitute Subscribe

🔷 Task 3: Comparison Operators

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Question:

Write a program that compares two numbers `m = 12` and `n = 15` using all comparison operators (`==`, `!=`, `<`, `>`, `<=`, `>=`) and prints the results (true/false or 1/0) to the Serial Monitor.

Expected Output:

```
m == n: 0
m != n: 1
m < n: 1
m > n: 0
m <= n: 1
m >= n: 0
```

YouTube @zedkinginstitute Subscribe

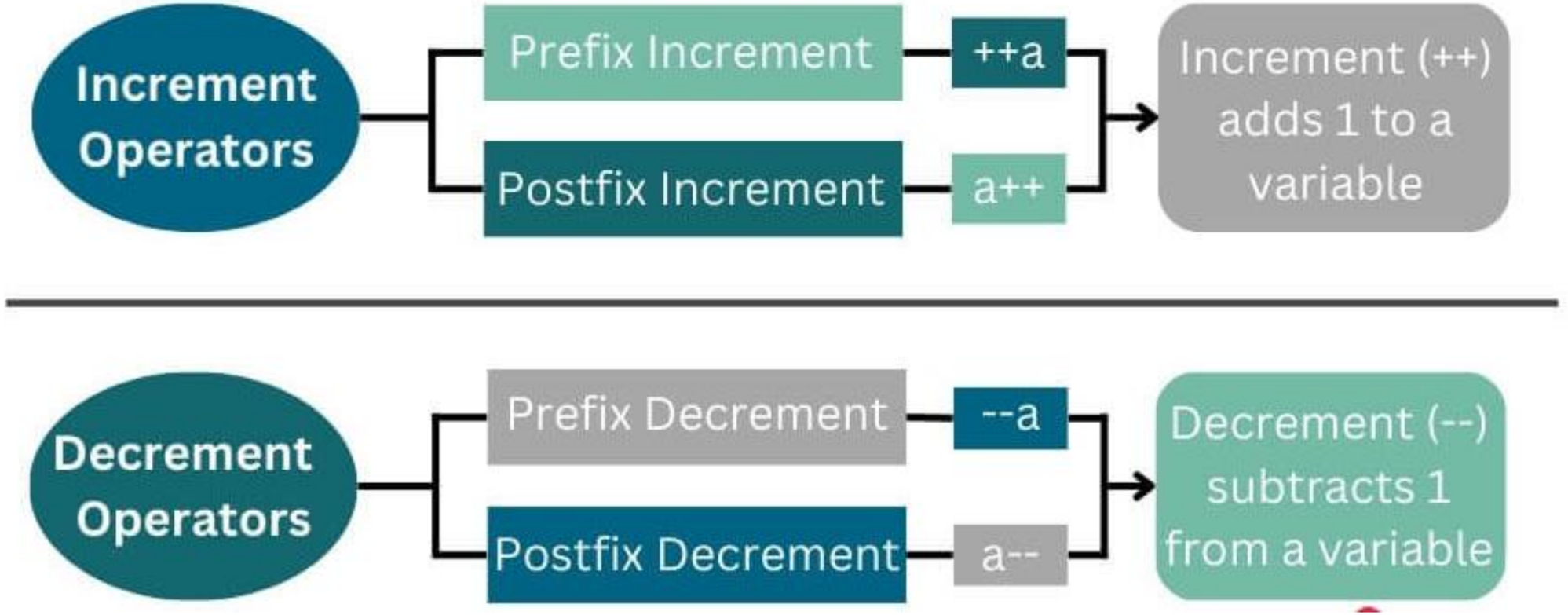
IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-6

❖ Increment/Decrement Operator

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

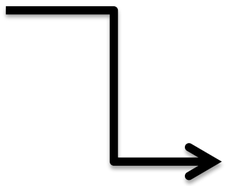


YouTube @zedkinginstitute Subscribe

❖ Example

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  int x = 5;

  Serial.println("Increment and Decrement Operations:");

  Serial.print("Initial value of x: ");
  Serial.println(x);

  Serial.print("Post-increment (x++): ");
  Serial.println(x++);  // prints 5, then x becomes 6

  Serial.print("Value after post-increment: ");
  Serial.println(x);    // prints 6

  Serial.print("Pre-increment (++x): ");
  Serial.println(++x);  // x becomes 7, then prints 7

  Serial.print("Post-decrement (x--): ");
  Serial.println(x--);  // prints 7, then x becomes 6

  Serial.print("Value after post-decrement: ");
  Serial.println(x);    // prints 6

  Serial.print("Pre-decrement (--x): ");
  Serial.println(--x);  // x becomes 5, then prints 5
}

void loop() {}
```



```
Increment and Decrement Operations:
Initial value of x: 5
Post-increment (x++): 5
Value after post-increment: 6
Pre-increment (++x): 7
Post-decrement (x--): 7
Value after post-decrement: 6
Pre-decrement (--x): 5
```

YouTube @zedkinginstitute Subscribe

❖ Logical Operator

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

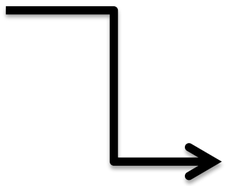
Operator	Description	Example	Result
and	Returns True if both operands are True.	True and False	False
or	Returns True if at least one operand is True.	True or False	True
not	Reverses the Boolean value.	not True	False

YouTube @zedkinginstitute Subscribe

❖ Example

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  int a = 10;
  int b = 20;
  Serial.println("Logical Operators Demonstration:");

  // Logical AND (&&)
  Serial.print("(a > 5 && b > 15): ");
  Serial.println((a > 5 && b > 15));  // true && true -> 1

  Serial.print("(a > 15 && b > 15): ");
  Serial.println((a > 15 && b > 15)); // false && true -> 0

  // Logical OR (||)
  Serial.print("(a > 15 || b > 15): ");
  Serial.println((a > 15 || b > 15)); // false || true -> 1

  Serial.print("(a > 15 || b < 10): ");
  Serial.println((a > 15 || b < 10)); // false || false -> 0

  // Logical NOT (!)
  Serial.print("!(a > 15): ");
  Serial.println(!(a > 15)); // !(false) -> true -> 1

  Serial.print("!(b == 20): ");
  Serial.println(!(b == 20)); // !(true) -> false -> 0
}

void loop() {}
```



```
Logical Operators Demonstration:
(a > 5 && b > 15): 1
(a > 15 && b > 15): 0
(a > 15 || b > 15): 1
(a > 15 || b < 10): 0
!(a > 15): 1
!(b == 20): 0
```

YouTube @zedkinginstitute Subscribe

Increment/Decrement Operator Questions

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

1. What will be the output of this expression?

```
int a = 5; int b = a++; Serial.println(b);
```

2. Predict the output:

```
int x = 3; int y = ++x + x++; Serial.println(y);
```

3. What are the values of `a`, `b`, and `c`?

```
int a = 2, b = 4, c = 6; a++; b--; ++c; Serial.println(a); Serial.println(b);
Serial.println(c);
```

YouTube @zedkinginstitute Subscribe

Logical Operator Questions

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

4. What will be printed?

```
int a = 1, b = 0; Serial.println(a && b);
```

5. Predict the result:

```
int x = 0, y = 5; int result = x++ && ++y; Serial.println(result);
Serial.println(x); Serial.println(y);
```

6. What is the output of the following?

```
bool flag = false; flag = !flag; Serial.println(flag);
```

YouTube @zedkinginstitute Subscribe

Bitwise Operator Questions

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

7. What is the output of this operation?

```
int a = 6, b = 3; Serial.println(a & b);
```

8. Predict the values of `x` and `y` after swapping using XOR:

```
int x = 4, y = 5; x = x ^ y; y = x ^ y; x = x ^ y; Serial.println(x);
Serial.println(y);
```

9. What will be printed?

```
int val = 5; Serial.println(~val);
```

10. What is the result of these bit shifts?

```
int num = 8; Serial.println(num << 1); Serial.println(num >> 1);
```

YouTube @zedkinginstitute Subscribe

❖ Bitwise Operator

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

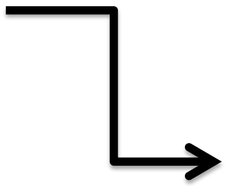
Operator	Description	Example	Result
&	Bitwise AND: Sets each bit to 1 if both bits are 1.	5 & 3	1
|	Bitwise OR: Sets each bit to 1 if one of the bits is 1.	5 | 3	7
^	Bitwise XOR: Sets each bit to 1 if only one of the bits is 1.	5 ^ 3	6
~	Bitwise NOT: Inverts all the bits.	~5	-6
<<	Left shift: Shifts the bits of the first operand to the left by the number of positions specified by the second operand.	5 << 1	10
>>	Right shift: Shifts the bits of the first operand to the right by the number of positions specified by the second operand.	5 >> 1	2

YouTube @zedkinginstitute Subscribe

❖ Example

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  int a = 10;
  int b = 20;
  Serial.println("Logical Operators Demonstration:");

  // Logical AND (&&)
  Serial.print("(a > 5 && b > 15): ");
  Serial.println((a > 5 && b > 15));  // true && true -> 1
  Serial.print("(a > 15 && b > 15): ");
  Serial.println((a > 15 && b > 15)); // false && true -> 0

  // Logical OR (||)
  Serial.print("(a > 15 || b > 15): ");
  Serial.println((a > 15 || b > 15)); // false || true -> 1
  Serial.print("(a > 15 || b < 10): ");
  Serial.println((a > 15 || b < 10)); // false || false -> 0

  // Logical NOT (!)
  Serial.print("!(a > 15): ");
  Serial.println(!(a > 15)); // !(false) -> true -> 1
  Serial.print("!(b == 20): ");
  Serial.println(!(b == 20)); // !(true) -> false -> 0
}

void loop() {}
```



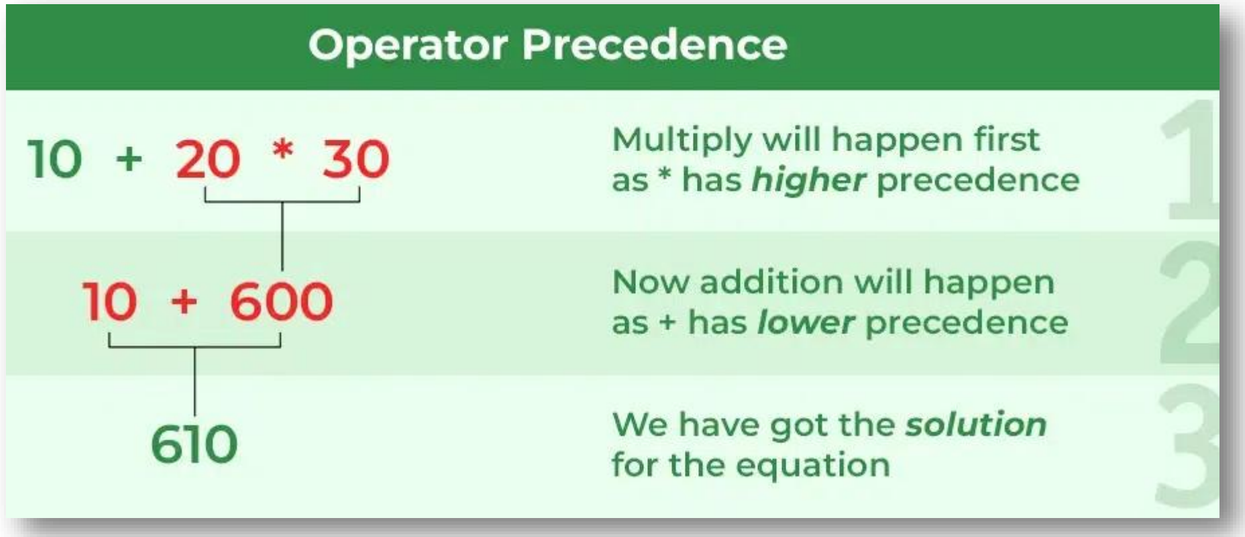
```
Increment and Decrement Operations:
Initial value of x: 5
Post-increment (x++): 5
Value after post-increment: 6
Pre-increment (++x): 7
Post-decrement (x--): 7
Value after post-decrement: 6
Pre-decrement (--x): 5
```

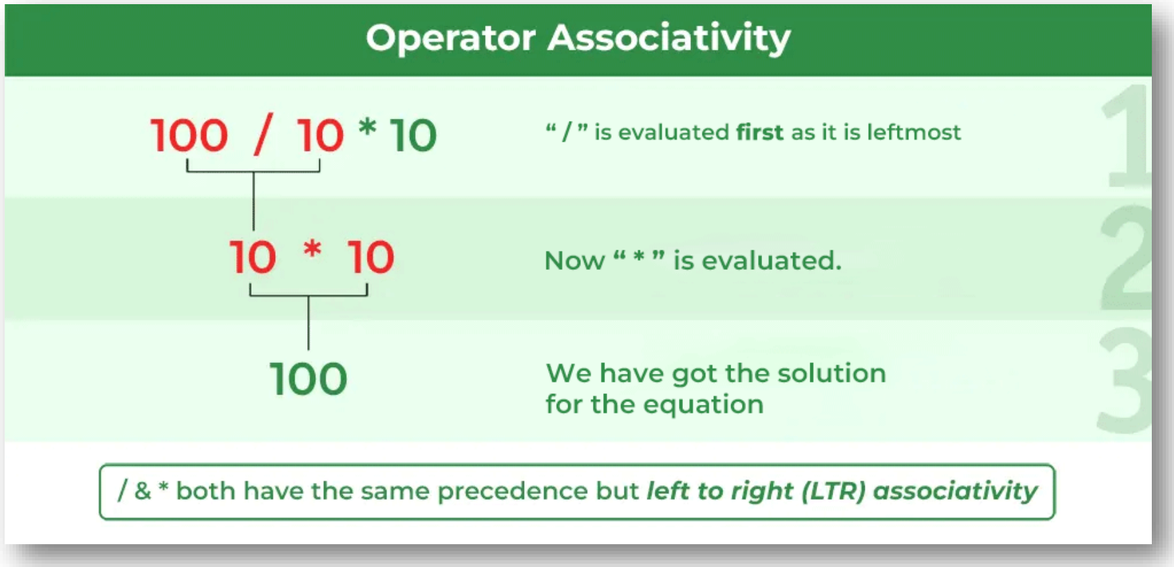
YouTube @zedkinginstitute Subscribe

❖ Precedence & Associativity of Operators

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- **Operator precedence** determines which operation is performed first in an expression with more than one operator with different precedence.
- When two operators of the same precedence appear in an expression. Associativity can be either from **Left to Right** or **Right to Left**.





YouTube @zedkinginstitute Subscribe

❖ Operators Precedence

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Precedence Table

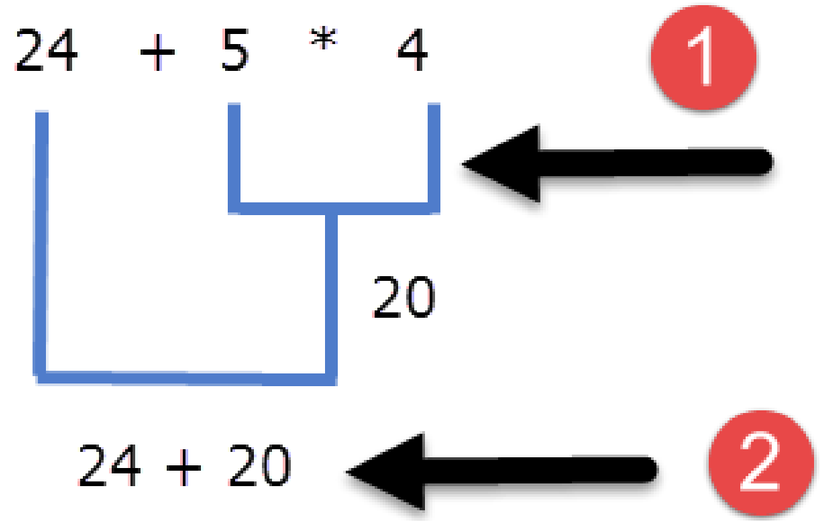
Description	Operator	Associativity
Function expression	( )	Left to Right
Array Expression	[ ]	
Unary Minus	-	Right to Left
Increment/Decrement	++ --	
One's compliment	~	
Negation	!	
Size in bytes	sizeof	
Type cast	(type)	
Multiplication	*	Left to Right
Division	/	
Modulus	%	
Addition	+	
Subtraction	-	
Left shift	<<	Left to Right
Right shift	>>	Left to Right
Less than	<	Left to Right
Less than or equal to	<=	
Greater than	>	
Greater than or equal to	>=	

All operators have same precedence

All operators have same precedence

All operators have same precedence

All operators have same precedence



Ans : 44

YouTube @zedkinginstitute Subscribe

❖ Example

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

This is how the following expression is evaluated:



```
1 + 2 * 3 - 4
1 + 6 - 4
7 - 4
3
```

The programmer can **use parentheses to override the hierarchy** and force a desired order of evaluation. **Expressions enclosed in parentheses are evaluated first**. For example:

```
(1 + 2) * (3 - 4)
3 * -1
-3
```

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-7

❖ Control Statements

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

A control statement in programming is a command that changes the normal sequential execution of code. It allows a program to make decisions, perform actions repeatedly, or jump to other parts of the code, based on specific conditions or logic.

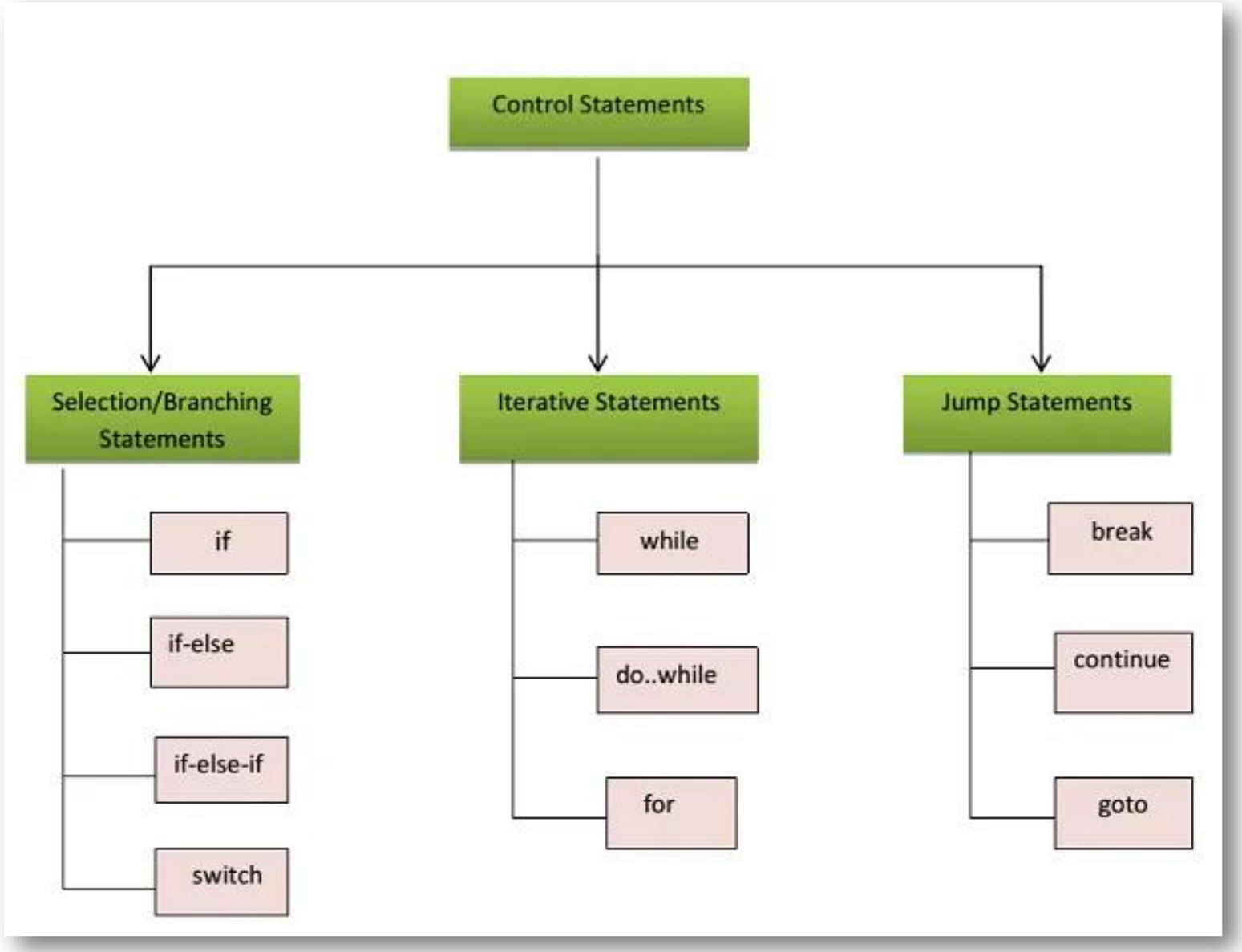
🎯 Purpose of Control Statements:

- To make decisions (if, switch)
- To repeat a block of code (for, while)
- To skip or stop certain parts (break, continue, return)
- To organize complex logic in a clean and readable way

YouTube @zedkinginstitute Subscribe

❖ Types of Control Statements

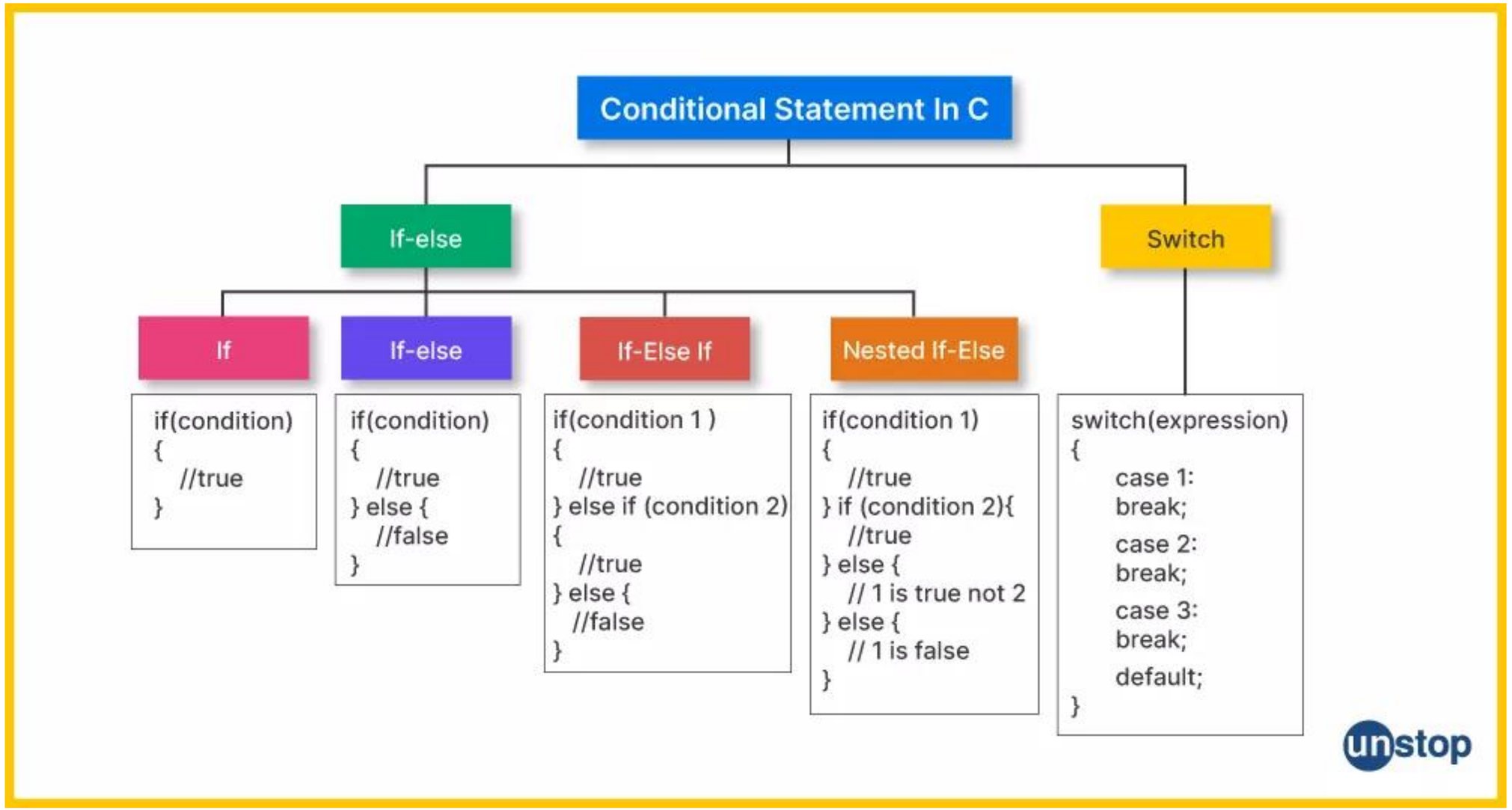
Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified



YouTube @zedkinginstitute Subscribe

❖ Conditional Statements

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified



YouTube @zedkinginstitute Subscribe

❖ If Statements

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  int number = 7;

  Serial.println("Checking number using if statement...");

  if (number > 0) {
    Serial.println("The number is positive.");
  }
}

void loop() {
  // Nothing here
}
```

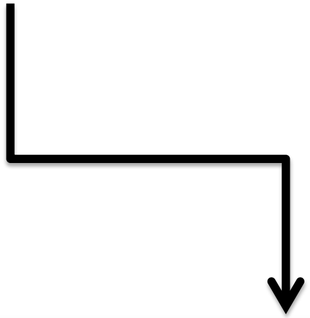
```
Checking number using if statement...
The number is positive.
```

YouTube @zedkinginstitute Subscribe

❖ If-else Statements

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  int number = -3;

  Serial.println("Checking number using if-else statement...");

  if (number >= 0) {
    Serial.println("The number is positive.");
  } else {
    Serial.println("The number is negative.");
  }
}

void loop() {
  // Nothing here
}
```



```
Checking number using if-else statement...
The number is negative.
```

YouTube @zedkinginstitute Subscribe

❖ If-else if Statements

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  int number = 0;

  Serial.println("Checking number using if-else if-else statement...");

  if (number > 0) {
    Serial.println("The number is positive.");
  } else if (number < 0) {
    Serial.println("The number is negative.");
  } else {
    Serial.println("The number is zero.");
  }
}

void loop() {
  // Nothing here
}
```

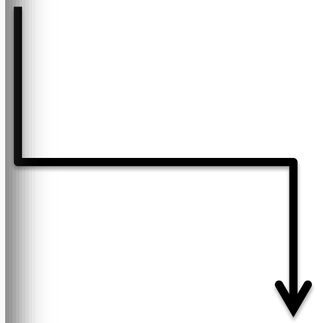
```
Checking number using if-else if-else statement...
The number is zero.
```

YouTube @zedkinginstitute Subscribe

❖ Nested if-else Statements

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  int number = 6;

  Serial.println("Checking number using nested if-else...");

  if (number > 0) {
    Serial.println("The number is positive.");

    if (number % 2 == 0) {
      Serial.println("It is even.");
    } else {
      Serial.println("It is odd.");
    }

  } else {
    Serial.println("The number is not positive.");
  }
}

void loop() {
  // Nothing here
}
```



```
Checking number using nested if-else...
The number is positive.
It is even.
```

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

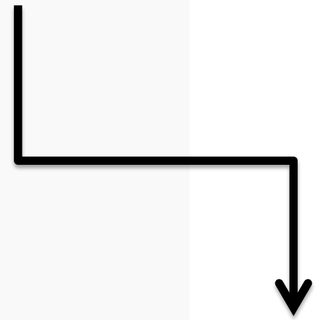
Chapter 4 -
Building of IOT Applications

Class-11

Switch Statements

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  int day = 3;

  Serial.println("Checking day using switch statement...");

  switch (day) {
    case 1:
      Serial.println("Monday");
      break;
    case 2:
      Serial.println("Tuesday");
      break;
    case 3:
      Serial.println("Wednesday");
      break;
    case 4:
      Serial.println("Thursday");
      break;
    case 5:
      Serial.println("Friday");
      break;
    case 6:
      Serial.println("Saturday");
      break;
    case 7:
      Serial.println("Sunday");
      break;
    default:
      Serial.println("Invalid day number.");
  }
}

void loop() {
  // Nothing here
}
```

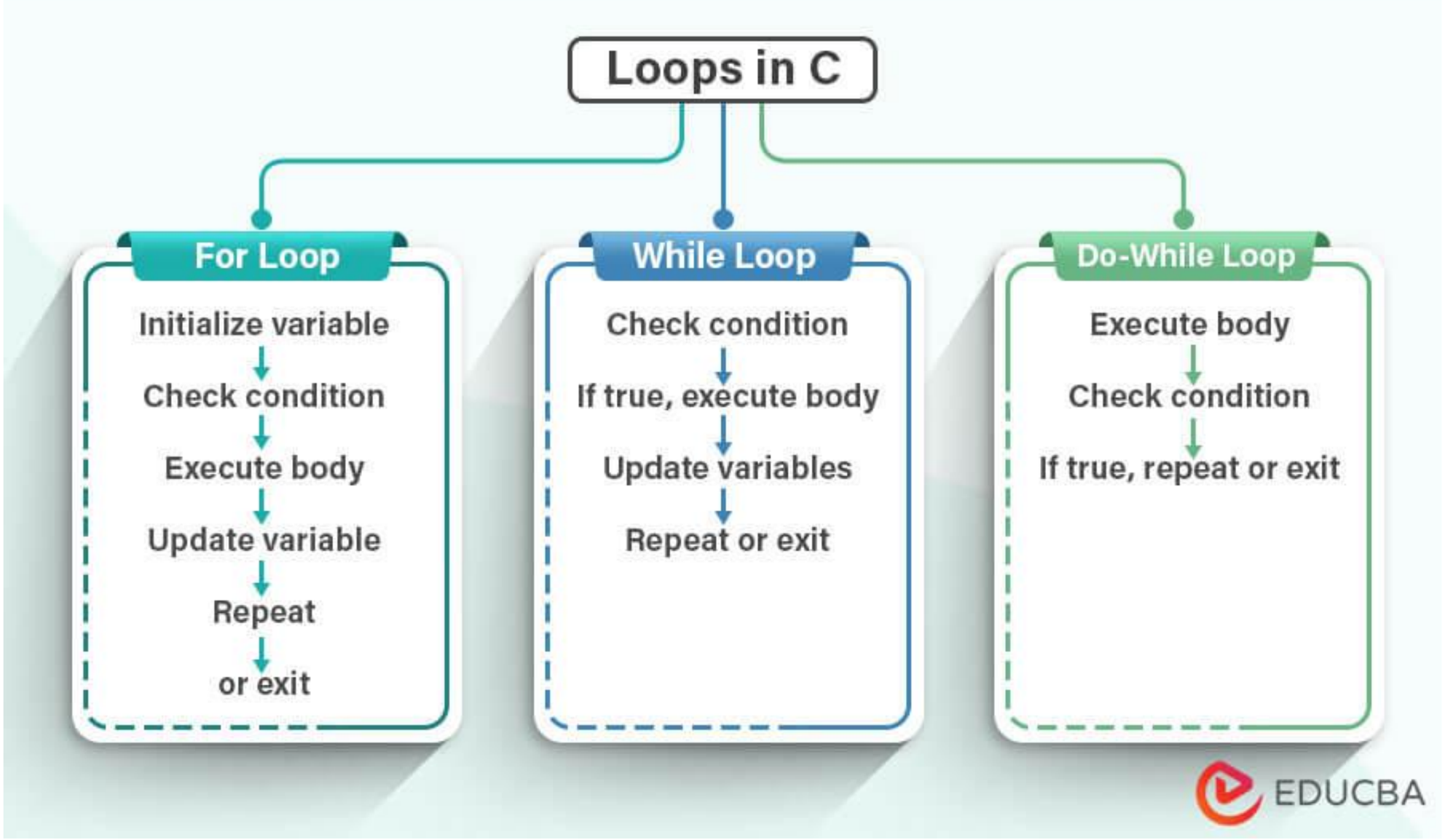


```
Checking day using switch statement...
Wednesday
```

YouTube @zedkinginstitute Subscribe

Iterative Statements

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified



YouTube @zedkinginstitute Subscribe

For Loop

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

When you know exactly how many times you want to loop through a block of code, use the `for` loop instead of a `while` loop:

Syntax

```
for (expression 1; expression 2; expression 3) {
  // code block to be executed
}
```

Expression 1 is executed (one time) before the execution of the code block.

Expression 2 defines the condition for executing the code block.

Expression 3 is executed (every time) after the code block has been executed.

The example below will print the numbers 0 to 4:

YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  Serial.println("Printing numbers from 1 to 5 using for loop:");

  for (int i = 1; i <= 5; i++) {
    Serial.println(i);
  }
}

void loop() {
  // Nothing here
}
```




```
Printing numbers from 1 to 5 using for loop:
1
2
3
4
5
```

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-12

While Loop

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

The `while` loop loops through a block of code as long as a specified condition is `true`:

Syntax

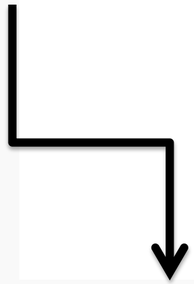
```
while (condition) {
  // code block to be executed
}
```

YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  int i = 1;

  Serial.println("Printing numbers from 1 to 5 using while loop:");

  while (i <= 5) {
    Serial.println(i);
    i++; // Increment to avoid infinite loop
  }
}

void loop() {
  // Nothing here
}
```





```
Printing numbers from 1 to 5 using while loop:
1
2
3
4
5
```

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-13

The Do/While Loop

ed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

The `do/while` loop is a variant of the `while` loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
do {
  // code block to be executed
}
while (condition);
```

YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600); // Start serial communication

  int i = 1;

  Serial.println("Printing numbers from 1 to 5 using do...while loop:");

  do {
    Serial.println(i);
    i++; // Increment to move toward loop exit
  } while (i <= 5);
}

void loop() {
  // Nothing here
}
```




```
Printing numbers from 1 to 5 using do...while loop:
1
2
3
4
5
```

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-14

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

What Are Jump Statements In C?

Break Statement	break;
Continue Statement	continue;
Goto Statement	goto label;
Return Statement	return expression;

YouTube @zedkinginstitute Subscribe

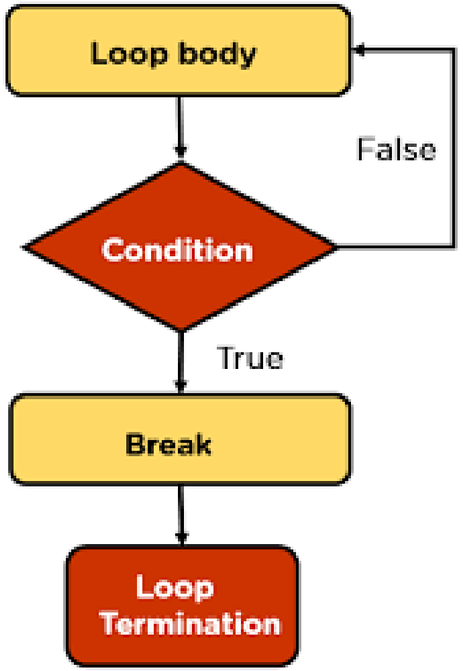
Break

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

You have already seen the `break` statement used in an earlier chapter of this tutorial. It was used to "jump out" of a `switch` statement.

The `break` statement can also be used to jump out of a **loop**.

This example jumps out of the **for loop** when `i` is equal to 4:




```
void setup() {
  Serial.begin(9600);

  for (int i = 1; i <= 10; i++) {
    if (i == 4) {
      break;  // Exit loop when i is 4
    }
    Serial.println(i);
  }
}

void loop() {
  // Nothing here
}
```

```
1
2
3
```

YouTube @zedkinginstitute Subscribe

Continue

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

The `continue` statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
void setup() {
  Serial.begin(9600);

  for (int i = 1; i <= 5; i++) {
    if (i == 3) {
      continue;  // Skip the rest of the loop when i is 3
    }
    Serial.println(i);
  }
}

void loop() {
  // Nothing here
}
```

```
1
2
4
5
```

YouTube @zedkinginstitute Subscribe

❖ Functions

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times.

There are two required functions in an Arduino sketch or a program i.e. setup () and loop(). Other functions must be created outside the brackets of these two functions.

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-15

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Functions

Predefined

- Library functions
- declarations in header files
- body in .dll files

User Defined

User customized functions, to reduce complexity of big programs.

YouTube @zedkinginstitute Subscribe

The most common syntax to define a function is –

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

RETURN TYPE :
is the type of the value returned by the function Can be any C data type

Function name :
is the identifier by which the function can be called

argument :
Parameters passed to function , any C data type

```
Return type  function name  ( argument1  , argument2  ,...)
{
   Statements
}
```

Statements or function body

YouTube @zedkinginstitute Subscribe

Function Declaration

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

A function is declared outside any other functions, above or below the loop function.

We can declare the function in two different ways –

The first way is just writing the part of the function called **a function prototype** above the loop function, which consists of –

- Function return type
- Function name
- Function argument type, no need to write the argument name

Function prototype must be followed by a semicolon (;).

YouTube @zedkinginstitute Subscribe

The following example shows the demonstration of the function declaration using the first method.

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Example

```
int sum_func (int x, int y) // function declaration {
   int z = 0;
   z = x+y ;
   return z; // return the value
}

void setup () {
   Statements // group of statements
}

Void loop () {
   int result = 0 ;
   result = Sum_func (5,6) ; // function call
}
```

YouTube @zedkinginstitute Subscribe

The second part, which is called the function definition or declaration, must be declared below the loop function, which consists of –

- Function return type
- Function name
- Function argument type, here you must add the argument name
- The function body (statements inside the function executing when the function is called)

YouTube @zedkinginstitute Subscribe

The following example demonstrates the declaration of function using the second method.

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Example

```
int sum_func (int , int ) ; // function prototype

void setup () {
   Statements // group of statements
}

Void loop () {
   int result = 0 ;
   result = Sum_func (5,6) ; // function call
}

int sum_func (int x, int y) // function declaration {
   int z = 0;
   z = x+y ;
   return z; // return the value
}
```

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-16

❖ Pre-defined Functions

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Pre-defined functions in Arduino C, also known as built-in functions, are functions readily available for use without the need for external library inclusion, although some require specific libraries. These functions are categorized based on their purpose, such as digital I/O, analog I/O, math, time, and communication.

YouTube @zedkinginstitute Subscribe

❖ Time Functions

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

S.No.	Function & Description
1	**delay () function** The way the **delay()** function works is pretty simple. It accepts a single integer (or number) argument. This number represents the time (measured in milliseconds).
2	**delayMicroseconds () function** The **delayMicroseconds()** function accepts a single integer (or number) argument. There are a thousand microseconds in a millisecond, and a million microseconds in a second.
3	**millis () function** This function is used to return the number of milliseconds at the time, the Arduino board begins running the current program.
4	**micros () function** The micros() function returns the number of microseconds from the time, the Arduino board begins running the current program. This number overflows i.e. goes back to zero after approximately 70 minutes.

YouTube @zedkinginstitute Subscribe

❖ Digital Input/ Output Functions

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

1. pinMode()

The pinMode() function is used to configure a specific pin to behave either as an input or an output.

Syntax

```
Void setup () {
   pinMode (pin , mode);
}
```

- **pin** – the number of the pin whose mode you wish to set
- **mode** – INPUT, OUTPUT, or INPUT_PULLUP.

YouTube @zedkinginstitute Subscribe

2. digitalWrite()

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- digitalWrite() is an **output function** in Arduino.
- It is used to **set a digital pin to HIGH or LOW**.
- Typically used to **turn devices ON or OFF**, such as LEDs, relays, or buzzers.

Syntax:

```
Void loop() {
   digitalWrite (pin ,value);
}
```

- **pin** – the number of the pin whose mode you wish to set
- **value** – HIGH, or LOW.

YouTube @zedkinginstitute Subscribe

Example

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
int LED = 6; // LED connected to pin 6

void setup () {
    pinMode(LED, OUTPUT); // set the digital pin as output
}

void setup () {
    digitalWrite(LED,HIGH); // turn on led
    delay(500); // delay for 500 ms
    digitalWrite(LED,LOW); // turn off led
    delay(500); // delay for 500 ms
}
```

YouTube @zedkinginstitute Subscribe

3. digitalRead()

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- digitalRead() is an **input function** used to **read digital values** (either HIGH or LOW) from a **digital pin**.
- It detects whether the pin is receiving **5V (HIGH)** or **0V (LOW)**.
- Commonly used to read input from devices like **buttons, sensors, or switches**

Syntax:

```
int value = digitalRead(pin);
```

- `pin` : the digital pin number you're reading from.
- Returns: `HIGH` or `LOW`

YouTube @zedkinginstitute Subscribe

Example

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
int ledPin = 13;  // LED connected to digital pin 13
int inPin = 7;    // pushbutton connected to digital pin 7
int val = 0;      // variable to store the read value

void setup() {
  pinMode(ledPin, OUTPUT);  // sets the digital pin 13 as output
  pinMode(inPin, INPUT);    // sets the digital pin 7 as input
}

void loop() {
  val = digitalRead(inPin);   // read the input pin
  digitalWrite(ledPin, val);  // sets the LED to the button's value
}
```

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-17

❖ Analog Input/ Output Functions

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

1. analogWrite()

- analogWrite() is an **output function** used to **send analog-like signals** (PWM) to a pin.
- It is used to **control brightness**, **motor speed**, etc.
- It does **not** generate true analog output—**it's actually PWM (Pulse Width Modulation).**

Syntax:

```
analogWrite(pin, value);
```

- `pin` : a **PWM-capable pin** (marked with `~` on most Arduino boards).
- `value` : a number between **0 (always off)** and **255 (always on)**.

YouTube @zedkinginstitute Subscribe

2. analogRead()

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- analogRead() is an **input function** used to read the **analog voltage** from a pin.
- It converts the analog signal (0 to 5V) into a **digital value from 0 to 1023**.
- Commonly used with sensors like **temperature sensors, light sensors, and potentiometers**.

Syntax:

```
analogRead(pin);
```

- **pin** – the number of the analog input pin to read from (0 to 5 on most boards, 0 to 7 on the Mini and Nano, 0 to 15 on the Mega)

YouTube @zedkinginstitute Subscribe

Example

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
int ledPin = 9;      // LED connected to digital pin 9
   int analogPin = A3;   // potentiometer connected to analog pin 3 (A3)
   int val = 0;          // variable to store the read value

   void setup() {
     pinMode(ledPin, OUTPUT);  // sets the pin as output
   }

   void loop() {
     val = analogRead(analogPin);  // read the input pin
     analogWrite(ledPin, val / 4); // analogRead values go from 0 to 1023, analogWrite
values from 0 to 255
   }
```

YouTube @zedkinginstitute Subscribe

2. analogReference()

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

- analogReference() sets the **reference voltage** used by analogRead().
- This voltage determines the **maximum value (1023)** for analog readings.
- By default, Arduino uses **5V (or 3.3V on some boards)** as the reference.

Syntax:

```
analogReference(type);
```

Reference Type	Description
`DEFAULT`	Uses the default reference voltage (5V on Uno).
`INTERNAL`	Uses the internal 1.1V reference voltage (Uno only).
`EXTERNAL`	Uses an external voltage applied to the AREF pin (up to 5V max).

> ⚠ **Important:** If using `EXTERNAL`, you must **connect a voltage to the AREF pin**.
> Never call `analogReference(EXTERNAL)` **without** connecting an external voltage — it can damage your board.

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-18

❖ Character Functions

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Sr. No.	Function Name	Description
1	`isAlpha(char c)`	Returns **true** if `c` is a **letter** (A–Z or a–z).
2	`isAlphaNumeric(char c)`	Returns **true** if `c` is a **letter or number** (A–Z, a–z, 0–9).
3	`isDigit(char c)`	Returns **true** if `c` is a **digit** (0–9).
4	`isControl(char c)`	Returns **true** if `c` is a **control character** (e.g., newline `\n`).
5	`isGraph(char c)`	Returns **true** if `c` has a visible representation (excluding space).
6	`isLowerCase(char c)`	Returns **true** if `c` is a **lowercase** letter (a–z).

YouTube @zedkinginstitute Subscribe

7	`isUpperCase(char c)`	Returns **true** if `c` is an **uppercase** letter (A–Z).
8	`isPrintable(char c)`	Returns **true** if `c` is a **printable character** (including space).
9	`isPunct(char c)`	Returns **true** if `c` is a **punctuation mark** (e.g., `.`, `,`, `?`).
10	`isSpace(char c)`	Returns **true** if `c` is a **whitespace** character (e.g., space, tab).
11	`isHexadecimalDigit(char c)`	Returns **true** if `c` is a valid **hex digit** (0–9, A–F, a–f).
12	`toLowerCase(char c)`	Converts character `c` to **lowercase** (if it's uppercase).
13	`toUpperCase(char c)`	Converts character `c` to **uppercase** (if it's lowercase).

YouTube @zedkinginstitute Subscribe

Example:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600);

  char ch1 = 'A';
  char ch2 = 'g';
  char ch3 = '5';
  char ch4 = ' ';
  char ch5 = '\n';
  char ch6 = '.';
  char ch7 = 'F';

  Serial.println("Character Function Examples:");

  Serial.print("1. isAlpha('A'): ");
  Serial.println(isAlpha(ch1));  // 1
```

```
  Serial.print("2. isAlphaNumeric('5'): ");
  Serial.println(isAlphaNumeric(ch3));  // 1

  Serial.print("3. isDigit('5'): ");
  Serial.println(isDigit(ch3));  // 1

  Serial.print("4. isControl('\\n'): ");
  Serial.println(isControl(ch5));  // 1

  Serial.print("5. isGraph('A'): ");
  Serial.println(isGraph(ch1));  // 1

  Serial.print("6. isLowerCase('g'): ");
  Serial.println(isLowerCase(ch2));  // 1
```

YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
Serial.print("2. isAlphaNumeric('5'): ");
Serial.println(isAlphaNumeric(ch3));  // 1

Serial.print("3. isDigit('5'): ");
Serial.println(isDigit(ch3));  // 1

Serial.print("4. isControl('\\n'): ");
Serial.println(isControl(ch5));  // 1

Serial.print("5. isGraph('A'): ");
Serial.println(isGraph(ch1));  // 1

Serial.print("6. isLowerCase('g'): ");
Serial.println(isLowerCase(ch2));  // 1
```

```
Serial.print("7. isUpperCase('A'): ");
Serial.println(isUpperCase(ch1));  // 1

Serial.print("8. isPrintable(' '): ");
Serial.println(isPrintable(ch4));  // 1

Serial.print("9. isPunct('.'): ");
Serial.println(isPunct(ch6));  // 1

Serial.print("10. isSpace(' '): ");
Serial.println(isSpace(ch4));  // 1

Serial.print("11. isHexadecimalDigit('F'): ");
Serial.println(isHexadecimalDigit(ch7));  // 1
```

YouTube @zedkinginstitute Subscribe

Output:

```
Character Function Examples:
1. isAlpha('A'): 1
2. isAlphaNumeric('5'): 1
3. isDigit('5'): 1
4. isControl('\n'): 1
5. isGraph('A'): 1
6. isLowerCase('g'): 1
7. isUpperCase('A'): 1
8. isPrintable(' '): 1
9. isPunct('.'): 1
10. isSpace(' '): 1
11. isHexadecimalDigit('F'): 1
```

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-19

Arduino Math Functions (with Description)

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Sr. No.	Function Name	Description
1	`abs(x)`	Returns the **absolute** (positive) value of `x`.
2	`constrain(x, a, b)`	Limits `x` to the range between `a` and `b`.
3	`map(x, inMin, inMax, outMin, outMax)`	Re-maps `x` from one range to another.
4	`max(a, b)`	Returns the **larger** of `a` and `b`.
5	`min(a, b)`	Returns the **smaller** of `a` and `b`.

YouTube @zedkinginstitute Subscribe

6	`pow(base, exponent)`	Returns base raised to the power of exponent.
7	`sqrt(x)`	Returns the **square root** of `x`.
8	`sq(x)`	Returns the square of `x` (same as `x * x`).
9	`round(x)`	Rounds a float to the **nearest integer**.
10	`ceil(x)`	Rounds a float **up** to the nearest integer.
11	`floor(x)`	Rounds a float **down** to the nearest integer.
12	`fmod(x, y)` (in C++)	Returns the **remainder** of x divided by y (not in standard Arduino lib).

YouTube @zedkinginstitute Subscribe

Example:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600);

  int a = -15;
  int b = 10;
  int c = 150;
  float f1 = 3.6;
  float f2 = 4.4;

  Serial.println("Math Function Examples:");

  Serial.print("1. abs(-15) = ");
  Serial.println(abs(a)); // 15
  Serial.println(abs(a)); // 15

  Serial.print("2. constrain(150, 0, 100) = ");
  Serial.println(constrain(c, 0, 100)); // 100

  Serial.print("3. map(50, 0, 100, 0, 255) = ");
  Serial.println(map(50, 0, 100, 0, 255)); // 127

  Serial.print("4. max(10, -15) = ");
  Serial.println(max(b, a)); // 10

  Serial.print("5. min(10, -15) = ");
  Serial.println(min(b, a)); // -15

  Serial.print("6. pow(2, 3) = ");
  Serial.println(pow(2, 3)); // 8.0
```

YouTube @zedkinginstitute Subscribe

```
  Serial.print("7. sqrt(49) = ");
  Serial.println(sqrt(49)); // 7.0

  Serial.print("8. sq(5) = ");
  Serial.println(sq(5)); // 25

  Serial.print("9. round(3.6) = ");
  Serial.println(round(f1)); // 4

  Serial.print("10. ceil(4.4) = ");
  Serial.println(ceil(f2)); // 5

  Serial.print("11. floor(4.4) = ");
  Serial.println(floor(f2)); // 4
}
```

Output:

```
Math Function Examples:
1. abs(-15) = 15
2. constrain(150, 0, 100) = 100
3. map(50, 0, 100, 0, 255) = 127
4. max(10, -15) = 10
5. min(10, -15) = -15
6. pow(2, 3) = 8.00
7. sqrt(49) = 7.00
8. sq(5) = 25
9. round(3.6) = 4
10. ceil(4.4) = 5
11. floor(4.4) = 4
```

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class- 20

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

📋 Trigonometric Functions in Arduino

Sr. No.	Function	Description	Syntax	Example Input (Degrees)	Example Output
1	`sin()`	Returns the sine of an angle in radians	`sin(x)`	`sin(45°)`	≈ `0.707`
2	`cos()`	Returns the cosine of an angle in radians	`cos(x)`	`cos(45°)`	≈ `0.707`
3	`tan()`	Returns the tangent of an angle in radians	`tan(x)`	`tan(45°)`	≈ `1.000`

> ⚠️ **Note:** All inputs to these functions **must be in radians.**

YouTube @zedkinginstitute Subscribe

Example:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600);

  float degree = 45.0;
  float radian = degree * (PI / 180.0);  // Convert degrees to radians

  Serial.println("Trigonometric Functions (45 degrees):");

  Serial.print("sin(45°) = ");
  Serial.println(sin(radian));

  Serial.print("cos(45°) = ");
  Serial.println(cos(radian));

  Serial.print("tan(45°) = ");
  Serial.println(tan(radian));
}
```

Output

```
Trigonometric Functions (45 degrees):
sin(45°) = 0.7071
cos(45°) = 0.7071
tan(45°) = 1.0000
```

YouTube @zedkinginstitute Subscribe

Random Functions in Arduino

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Sr. No.	Function	Description
1	`random(max)`	Returns a random number from **0** to **(max - 1)**
2	`random(min, max)`	Returns a random number from **min** to **(max - 1)**
3	`randomSeed(seed)`	Sets the starting point (seed) for random number generation (optional)

> 🔹 `randomSeed()` is used to initialize the random generator (e.g., using `analogRead(A0)` for more randomness).
> 🔹 If `randomSeed()` is not used, results may be the same every time the Arduino restarts.

YouTube @zedkinginstitute Subscribe

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
  Serial.begin(9600);

  // Initialize random seed (use unconnected analog pin for randomness)
  randomSeed(analogRead(0));

  Serial.println("Random Number Examples:");

  // Random number from 0 to 99
  long rand1 = random(100);
  Serial.print("random(100): ");
  Serial.println(rand1);

  // Random number from 10 to 49
  long rand2 = random(10, 50);
  Serial.print("random(10, 50): ");
  Serial.println(rand2);
}

void loop() {
  // Nothing here
}
```

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-21

❖ Arrays

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

An **array** groups multiple values into one variable. Each value can be accessed by its **index number**, starting from **0**.

🧠 Syntax:

```cpp
datatype arrayName[arraySize];
```

YouTube @zedkinginstitute Subscribe

🛠️ 2. Array Declaration Methods

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Method Type	Syntax Example	Notes
Fixed size, initialized	`int nums[5] = {10, 20, 30, 40, 50};`	Declares size and sets values
Size inferred	`int nums[] = {10, 20, 30};`	Size is auto-calculated from number of elements
Fixed size, not initialized	`int nums[3];`	Declares an array with garbage/default values
Initialize later	`nums[0] = 100;`	Values assigned after declaration

YouTube @zedkinginstitute Subscribe

✅ 3. Simple Examples of Each Type

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

🔷 Fixed Size with Initialization

```cpp
int nums[3] = {1, 2, 3};  // Creates array of 3 integers
```

🔷 Auto Size (Let Compiler Count)

```cpp
int pins[] = {2, 4, 6, 8};  // Automatically size = 4
```

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-22

🔷 Declaration Only (Uninitialized)

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

cpp

```cpp
float temps[5];  // Creates array of 5 floats, values undefined
```

Later:

cpp

```cpp
temps[0] = 23.5;
temps[1] = 25.2;
```

YouTube @zedkinginstitute Subscribe

Character Array (String)

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

cpp

```cpp
char name[] = "Arduino";  // Character array including null '\0'
```

Or manually:

cpp

```cpp
char name[8] = {'A', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
```

YouTube @zedkinginstitute Subscribe

✅ Example: Integer Array

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```cpp
void setup() {
  Serial.begin(9600);

  int numbers[5] = {10, 20, 30, 40, 50};

  Serial.println("Array Elements:");
  for (int i = 0; i < 5; i++) {
    Serial.print("numbers[");
    Serial.print(i);
    Serial.print("] = ");
    Serial.println(numbers[i]);
  }
}

void loop() {
  // Nothing here
}
```

Output:

```
Array Elements:
numbers[0] = 10
numbers[1] = 20
numbers[2] = 30
numbers[3] = 40
numbers[4] = 50
```

YouTube @zedkinginstitute Subscribe

📘 Types of Arrays:

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Type	Example
Integer	`int nums[3] = {1, 2, 3};`
Float	`float temps[4] = {23.5, 24.0, 22.8, 25.1};`
Char	`char letters[4] = {'A', 'B', 'C', 'D'};`
String	`String names[2] = {"Ali", "Zara"};`

🧠 Notes:

- Indexing starts at **0**.
- Access elements like: `arrayName[index]`.
- Arrays can be **modified** just like regular variables: `numbers[2] = 99;`

YouTube @zedkinginstitute Subscribe

IOT M4-R5.1

Chapter 4 -
Building of IOT Applications

Class-23

❖ Array - Strings

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Strings are used to store text. They can be used to display text on an LCD or in the Arduino IDE Serial Monitor window. Strings are also useful for storing the user input. For example, the characters that a user types on a keypad connected to the Arduino.

There are two types of strings in Arduino programming –

- Arrays of characters, which are the same as the strings used in C programming.
- The Arduino String, which lets us use a string object in a sketch.

YouTube @zedkinginstitute Subscribe

String Character Arrays

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

The first type of string that we will learn is the string that is a series of characters of the type **char**. In the previous chapter, we learned what an array is; a consecutive series of the same type of variable stored in memory. A string is an array of char variables.

A string is a special array that has one extra element at the end of the string, which always has the value of 0 (zero). This is known as a "null terminated string".

String Character Array Example

This example will show how to make a string and print it to the serial monitor window.

YouTube @zedkinginstitute Subscribe

Example

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

```
void setup() {
   char my_str[6]; // an array big enough for a 5 character string
   Serial.begin(9600);
   my_str[0] = 'H'; // the string consists of 5 characters
   my_str[1] = 'e';
   my_str[2] = 'l';
   my_str[3] = 'l';
   my_str[4] = 'o';
   my_str[5] = 0; // 6th array element is a null terminator
   Serial.println(my_str);
}

void loop() {

}
```

YouTube @zedkinginstitute Subscribe

This same example can be written in a more convenient way as shown below –

Example

```
void setup() {
   char my_str[] = "Hello";
   Serial.begin(9600);
   Serial.println(my_str);
}

void loop() {

}
```

YouTube @zedkinginstitute Subscribe

Manipulating String Arrays

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

We can alter a string array within a sketch as shown in the following sketch.

Example

```
void setup() {
   char like[] = "I like coffee and cake"; // create a string
   Serial.begin(9600);
   // (1) print the string
   Serial.println(like);
   // (2) delete part of the string
   like[13] = 0;
   Serial.println(like);
   // (3) substitute a word into the string
   like[13] = ' '; // replace the null terminator with a space
   like[18] = 't'; // insert the new word
   like[19] = 'e';
   like[20] = 'a';
   like[21] = 0; // terminate the string
   Serial.println(like);
}

void loop() {
```

Result

```
I like coffee and cake
I like coffee
I like coffee and tea
```

YouTube @zedkinginstitute Subscribe

Our Social Media Platforms



