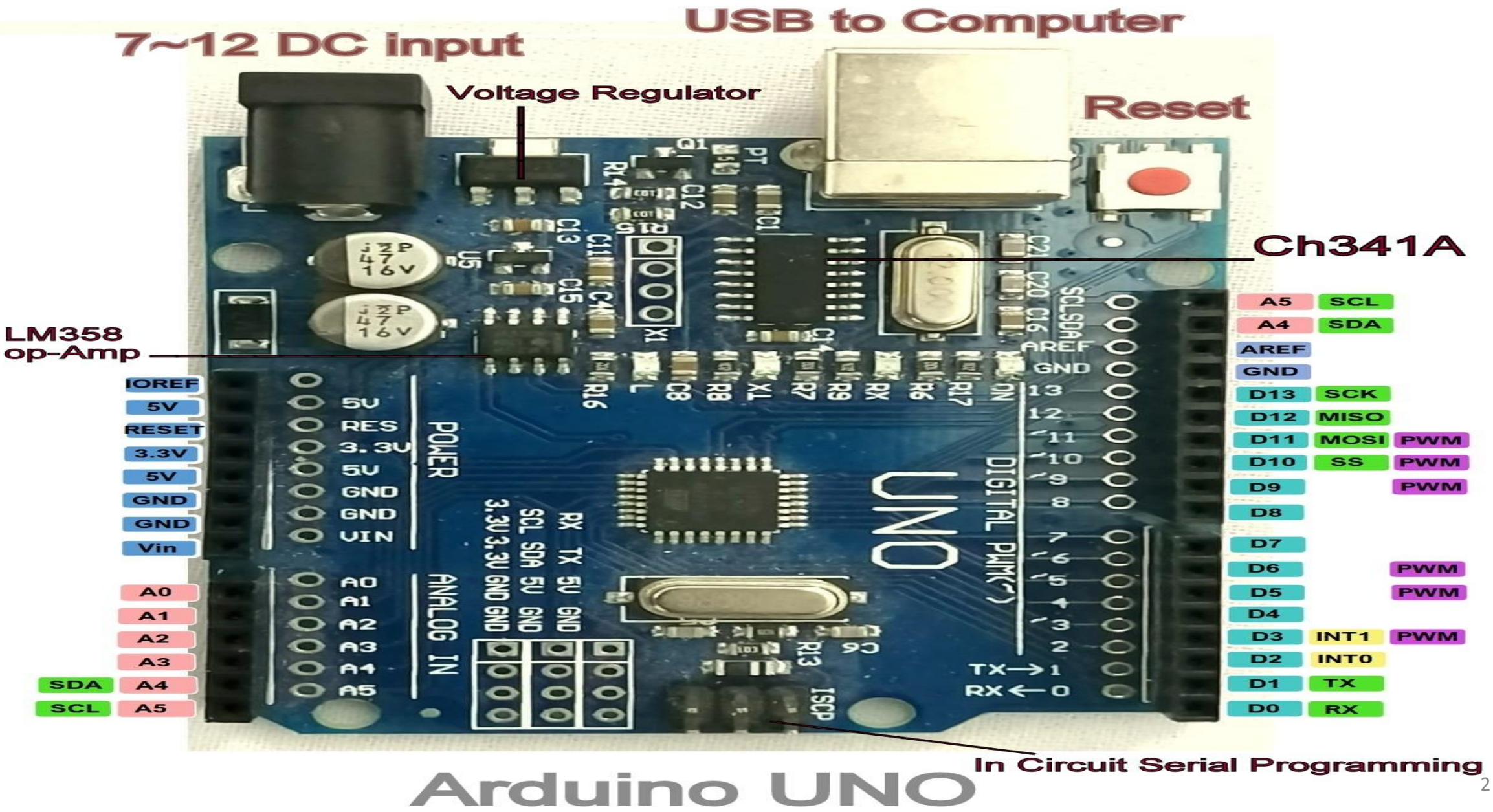

Unit 4:
Programming with the Arduino

Arduino Platform Boards Anatomy

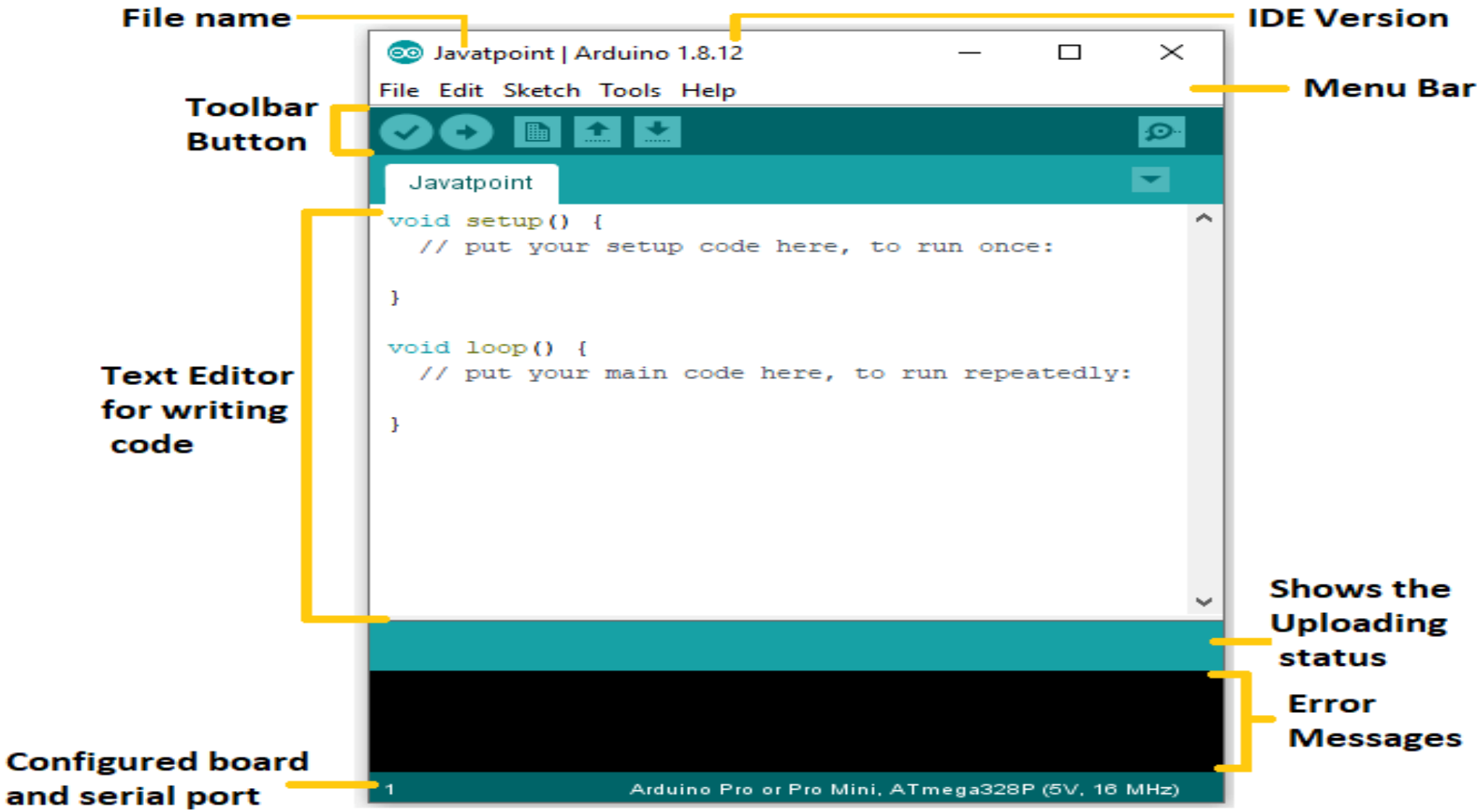
Explanation



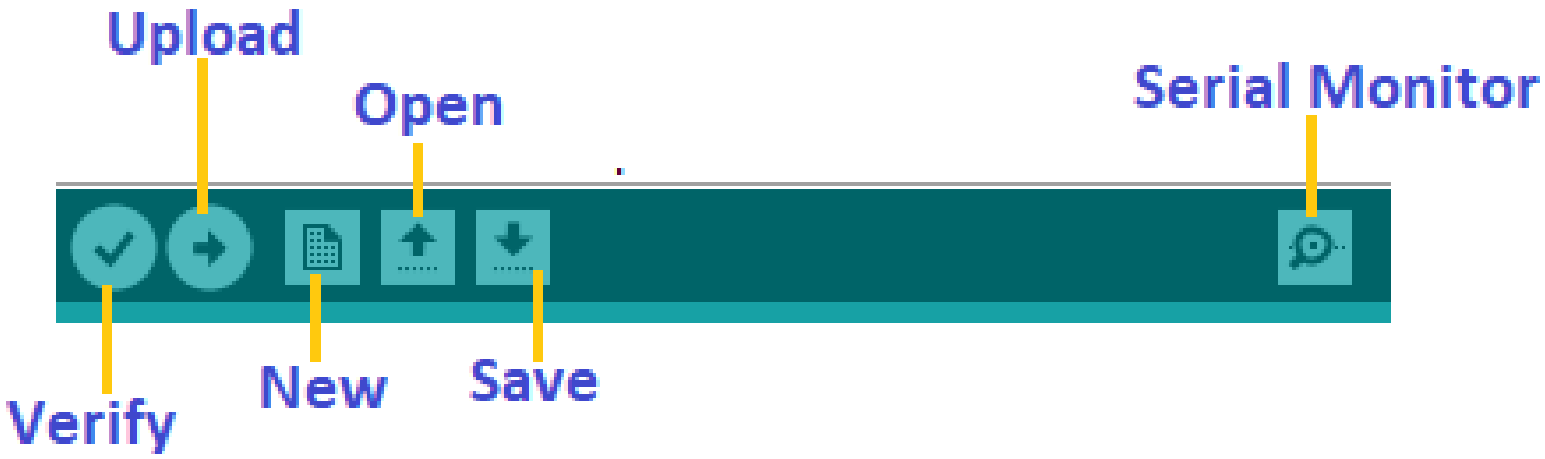
Arduino Platform Boards Anatomy

- **Microcontroller:** The brain of the board, responsible for processing and executing instructions. (ATmega328P)
- **Operating Voltage:** The voltage at which the board operates (5V logic level).
- **Input Voltage (recommended):** The safe voltage range to power the board externally (7-12V).
- **Digital I/O Pins:** Pins used for digital input and output operations, including PWM (Pulse Width Modulation) functionality. (14 (including 6 PWM outputs))
- **Analog Input Pins:** Pins that read varying voltage levels for sensors and other analog devices. (6 Pins)
- **Clock Speed:** Determines the processing speed of the microcontroller (16 MHz).
- **Flash Memory:** Stores the program/code that runs on the board (32 KB).
- **Communication:** Supports USB, UART, I2C, and SPI for data transfer and device interfacing.
- **Programming Interface:** Uses a USB Type-B connection to upload code via the Arduino IDE.

Arduino IDE


File name
IDE Version
Javatpoint | Arduino 1.8.12
File Edit Sketch Tools Help
Menu Bar
Toolbar Button
Javatpoint
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
}
Text Editor for writing code
Shows the Uploading status
Error Messages
Configured board and serial port
1
Arduino Pro or Pro Mini, ATmega328P (5V, 16 MHz)

Toolbar Buttons



Key components

Menu Bar:

- **File:** For creating new sketches, opening existing ones, and managing files.
- **Edit:** Standard editing functions like cut, copy, paste, and find/replace.
- **Sketch:** Contains options related to compiling, uploading, and verifying the code.
- **Tools:** For selecting the board, port, and configuring other settings.
- **Help:** Accesses documentation, tutorials, and support resources.

Key components

Toolbar:

- **Verify/Compile:** Checks the code for errors and generates the machine code.
- **Upload:** Sends the compiled code to the Arduino board.
- **New:** Creates a new sketch.
- **Open:** Opens an existing sketch.
- **Save:** Saves the current sketch.

Key components

Code Editor:

- The main area where you write your Arduino code.
- Supports syntax highlighting and auto-completion.

Key components

Message Area:

• Displays messages from the compiler, such as errors and warnings.

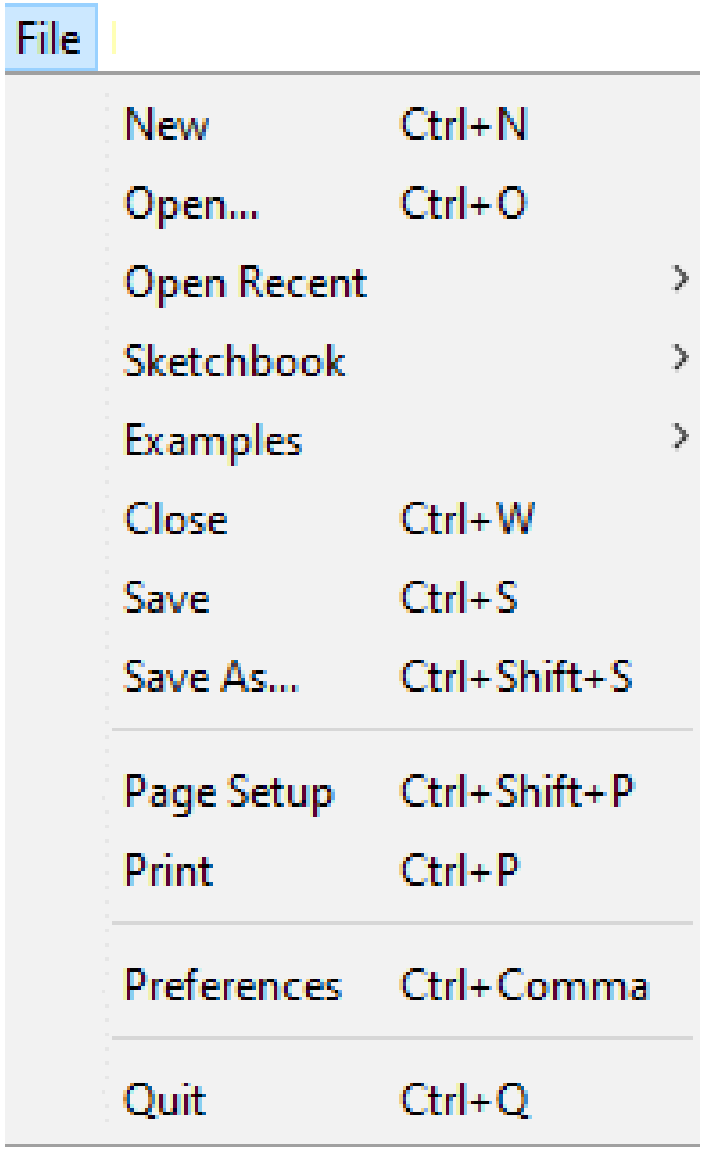
•Also shows output from the serial monitor.

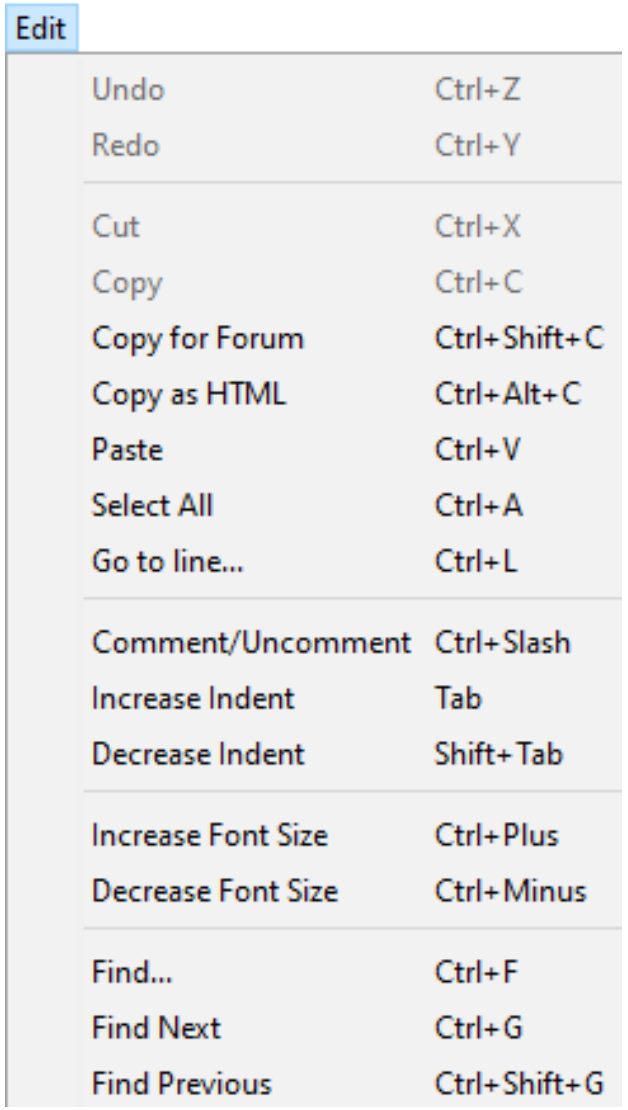
Serial Monitor:

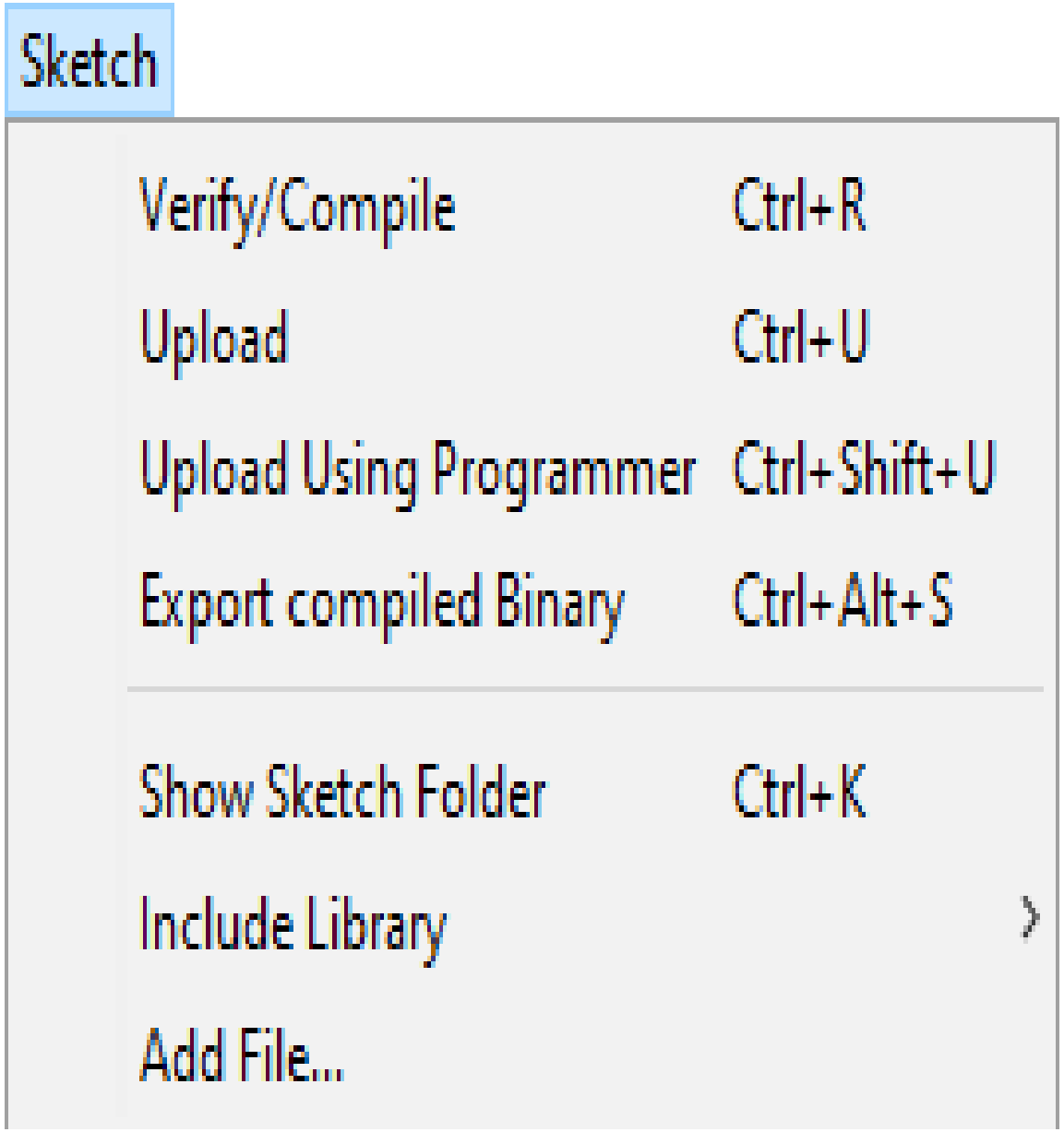
•Used to send and receive data from the Arduino board serially.

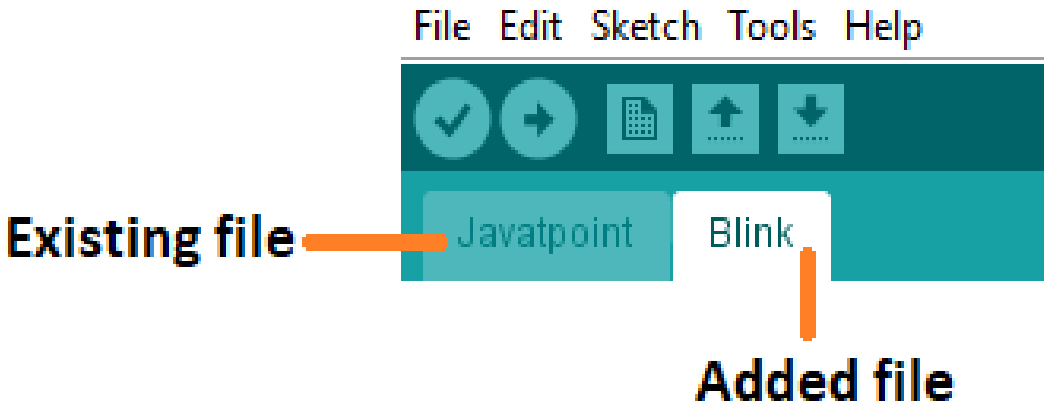
- **How it Works:**

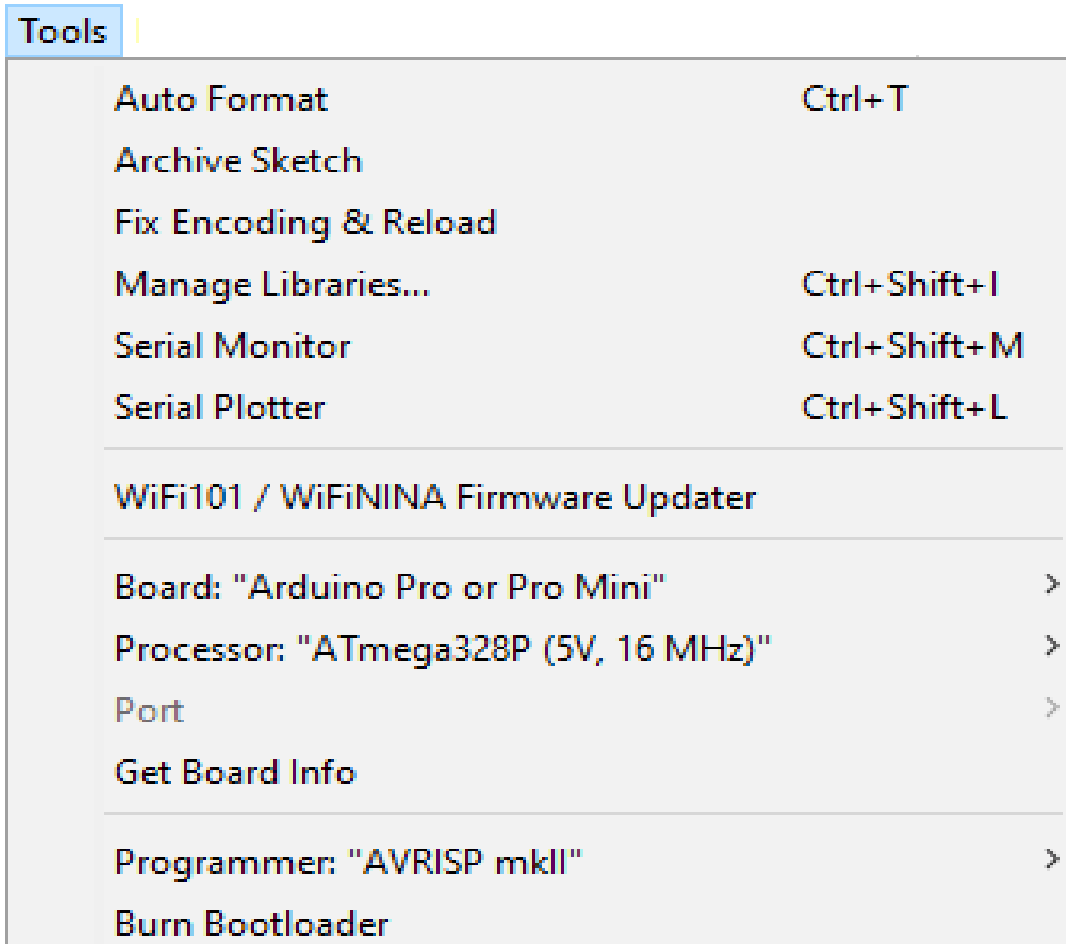
1. **Write Code:** You write your Arduino code in the code editor using the Arduino language (based on C/C++).
2. **Verify/Compile:** When you click "Verify," the IDE checks your code for syntax errors and compiles it into machine code that the Arduino board can understand.
3. **Upload:** Clicking "Upload" sends the compiled code to the Arduino board via the USB connection.
4. **Run Code:** The Arduino board executes the uploaded code, controlling the connected hardware.
5. **Monitor:** The serial monitor allows you to interact with the running code by sending and receiving data.

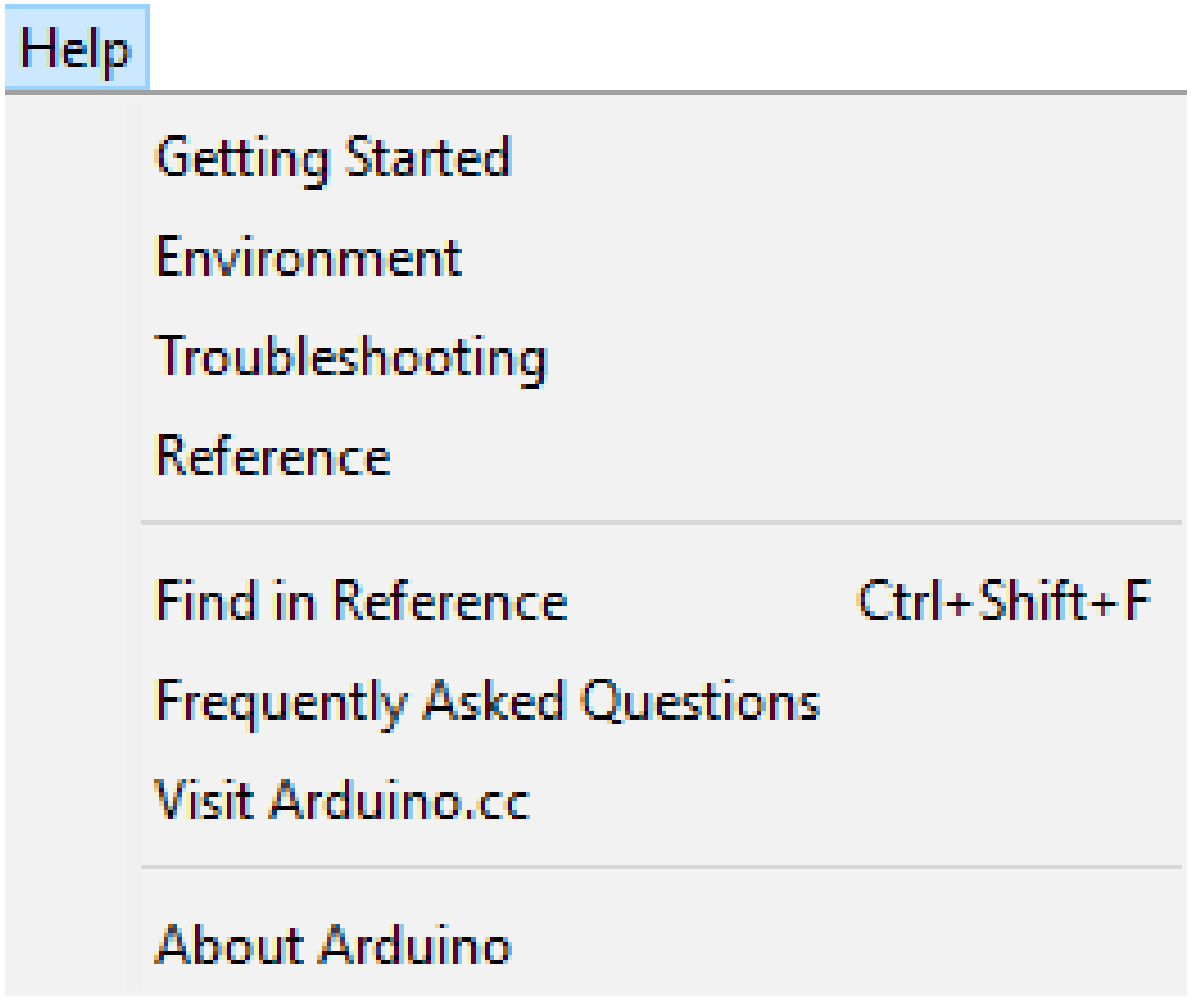

File
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q


Edit
Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Copy for Forum Ctrl+Shift+C
Copy as HTML Ctrl+Alt+C
Paste Ctrl+V
Select All Ctrl+A
Go to line... Ctrl+L
Comment/Uncomment Ctrl+Slash
Increase Indent Tab
Decrease Indent Shift+Tab
Increase Font Size Ctrl+Plus
Decrease Font Size Ctrl+Minus
Find... Ctrl+F
Find Next Ctrl+G
Find Previous Ctrl+Shift+G


Sketch
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...


File Edit Sketch Tools Help
Existing file
Javatpoint
Blink
Added file


Tools
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Pro or Pro Mini"
Processor: "ATmega328P (5V, 16 MHz)"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader


Help
Getting Started
Environment
Troubleshooting
Reference
Find in Reference Ctrl+Shift+F
Frequently Asked Questions
Visit Arduino.cc
About Arduino

Arduino Integrated Development Environment (IDE)

Menus with all commands

Buttons for common commands

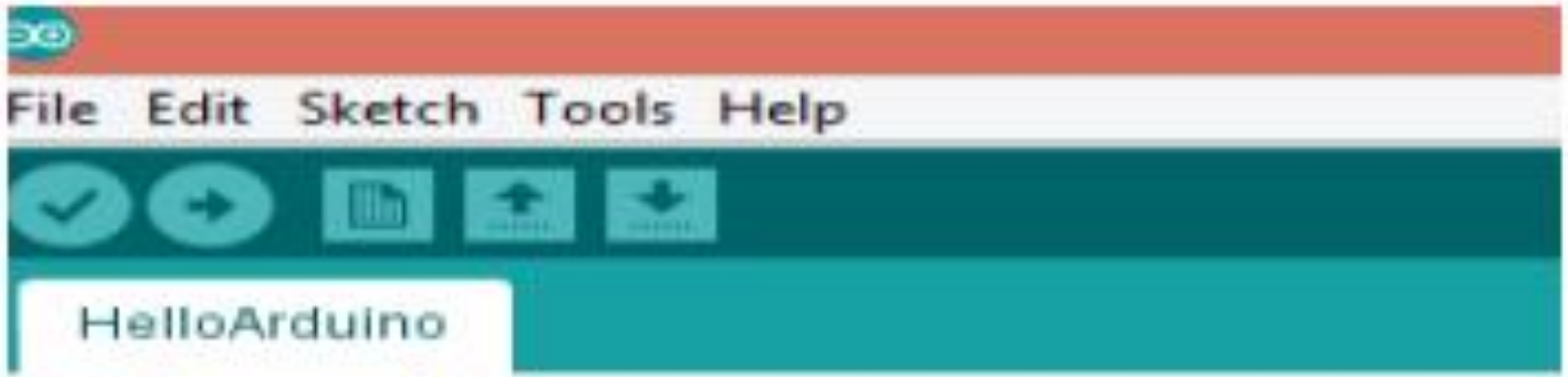
Text editor for writing code

Message area

ARDUINO IDE OVERVIEW:

- Program coded in Arduino IDE is called a **SKETCH**
- 1. To create a new sketch File -> New
- To open an existing sketch File -> open
- There are some basic ready-to-use sketches available in the EXAMPLES section: File -> Examples -> select any program
- 2. Verify: Checks the code for compilation errors
- 3. Upload: Uploads the final code to the controller board
- 4. New: Creates a new blank sketch with basic structure
- 5. Open: Opens an existing sketch
- 6. Save: Saves the current sketch

SKETCH STRUCTURE




```
void setup() {

  Serial.begin(9600);

}

void loop() {

  Serial.println("Hello Arduino!");


}
```

- **A sketch can be divided into two parts:**
- Setup ()
- Loop()
- The function **setup**() is the point where the code starts, just like the main() function in C and C++ . I/O Variables, pin modes are initialized in the Setup() function.
- **Loop**() function, as the name suggests, iterates the specified task in the program.

Arduino Function

- **Input/Output Functions:**
- The arduino pins can be configured to act as input or output pins using the pinMode() function.
- Void setup ()
- {
- pinMode (pin , mode);
- }

- Pin- pin number on the Arduino board , Mode- INPUT/OUTPUT

Cont…

- **digitalWrite()** : Writes a HIGH or LOW value to a digital pin.
- **analogRead**() : Reads from the analog input pin i.e., voltage applied across the pin.
- Character functions such as isdigit(), isalpha(), isalnum(), isxdigit(), islower(), isupper(), isspace() return 1(true) or 0(false).
- Delay() function is one of the most common time manipulation function used to provide a delay of specified time. It accepts integer value (time in miliseconds).

Library

- The Arduino environment can be extended through the use of libraries, just like most programming platforms. Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data.
- To use a library in a sketch, select it from **Sketch > Import Library**. A number of libraries come installed with the IDE, but you can also download or create your own.

Official Arduino Libraries

- **Robotics**
- Libraries for controlling servo and stepper motors.
- Servo - for controlling servo motors.
- Stepper - for controlling stepper motors.
- **Communication**
- Libraries for using the SPI, I2C and UART protocols.
- Wire - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.
- SoftwareSerial - for serial communication on any digital pins.

- **Connectivity**
- Libraries to access radio modules on different IoT boards (Wi-Fi, Bluetooth, LoRa, GSM, NB-IoT, Sigfox).
- ArduinoIoTCloud - This library allows to connect to the Arduino IoT Cloud service. .
- ArduinoBLE - library to use the Bluetooth Low Energy on a selection of boards.
- Ethernet - for connecting to the Internet via Ethernet.
- GSM - for connecting to a GSM/GRPS network with the GSM shield.
- WiFi - library for the WiFi shield, for Internet connections via Wi-Fi.
- WiFiNINA - library for boards with a Wi-Fi NINA module, for Internet connections via Wi-Fi.

➢**Memory**

- Libraries for memory management and data storage.
- EEPROM - reading and writing to "permanent" storage.
- SD - for reading and writing SD cards.

➢**Display**

- Libraries for controlling different displays.
- LiquidCrystal - for controlling liquid crystal displays (LCDs).
- TFT - for drawing text , images, and shapes on the Arduino TFT screen.

Programming the arduino for IoT

- To control an LED using an Arduino microcontroller and make it blink, the procedure involves the following steps:

Procedure:

Connect the LED to the Arduino:

- Insert the longer leg (anode) of the LED into a digital I/O pin on the Arduino (for example, pin 12).
- Insert the shorter leg (cathode) of the LED into the GND (ground) pin on the Arduino.

Write the Arduino code: In the Arduino IDE, open a new sketch (program).Define the pin connected to the LED as an output in the setup() function. In the loop() function, turn the LED on, wait for a specific amount of time, then turn the LED off and wait again. This cycle will repeat.

Example Code:

```
int led=12;
void setup()
{
pinMode(led, OUTPUT); // set the pin mode
}
void loop()
{
digitalWrite(led, HIGH); // Turn on the LED
delay(1000);
digitalWrite(led, LOW); //Turn off the LED
delay(1000);
}
```

- Set the pin mode as output which is connected to the led, pin 12 in this case.
- Use digitalWrite() function to set the output as HIGH and LOW.
- Delay() function is used to specify the delay between HIGH-LOW transition of the output.

Programming the arduino for IoT(Cont…)

Upload the Code:

- Connect your Arduino to your computer using a USB cable.
- Select the correct board and port in the Arduino IDE.
- Upload the code to the Arduino.

Observe the Blink: After uploading, the LED connected to pin 12 should blink on and off with a 1-second interval, as per the code.

THANK YOU