# Unit – 2 : Getting Started with Arduino

## Introduction to Arduino

- **Arduino** is an open-source electronics platform based on easy-to-use hardware and software. It is ideal for creating interactive projects by connecting sensors and actuators and programming logic using the Arduino IDE.

**Purpose:**

- Designed for **students**, **hobbyists**, and **professionals** to build **digital devices** and **interactive objects**.
- Suitable for both **beginners** and advanced users due to its simplicity and versatility.

**Key Features:**

- **Microcontroller-based** board (e.g., ATmega328P on the Arduino Uno).
- **Arduino IDE** supports coding in a simplified **C/C++-like language**.
- Provides **easy access to digital and analog I/O pins** for sensor/actuator connections.

**How Arduino Works?**

- It reads **inputs** such as light from a sensor, a finger on a button, or a Twitter message.
- It then **processes** the input according to your code (program).
- Finally, it can **control outputs** such as turning on an LED, starting a motor, or sending data to the internet.

**Programming:**

- Written using **Arduino IDE** which is compatible with Windows, macOS, and Linux.
- Code is uploaded via a USB cable using a **bootloader** already installed on the board.

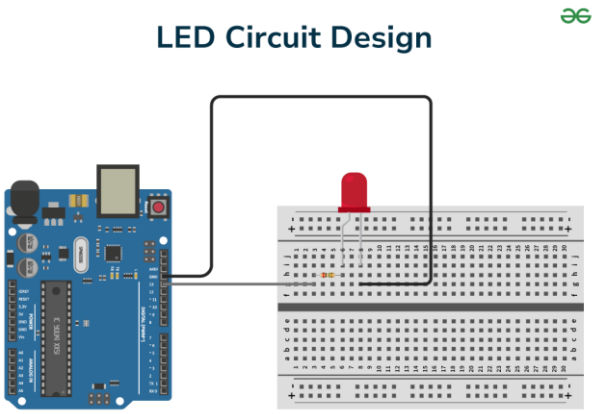
**Common Applications:**

- **LED blinking and lighting control**
- **Sensor monitoring** (e.g., temperature, humidity, motion)
- **Robotics** (e.g., motor control, obstacle avoidance)
- **Home automation** (e.g., smart lights, smart thermostats)
- **IoT projects** (e.g., weather stations, remote monitoring)

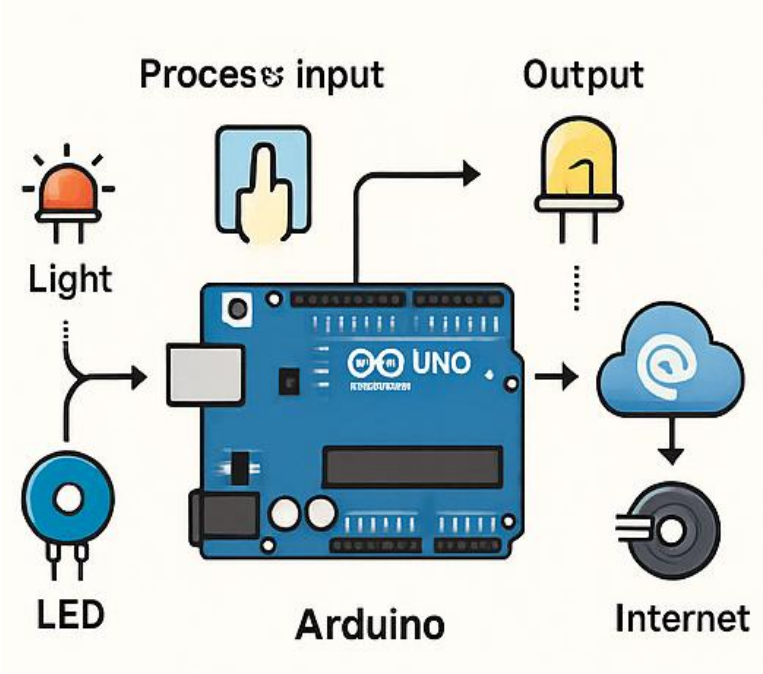
# Unit – 2 : Getting Started with Arduino

**Circuit Basics:**

Consider this basic LED circuit




- Consider the circuit shown above, the LED is connected to pins on the Arduino using some resistors to limit the amount of current flowing through the LED. The reference level is set using the ground pin.
- During the high or the 'ON' state, the circuit connections will be complete, and current will flow through the circuit components as programmed in the microcontroller. In this state, the LED will glow. When the circuit is turned 'OFF', the pins are set to low and the LED becomes dim due to no current flow.



# Unit – 2 : Getting Started with Arduino

## Arduino Pin Configuration and Architecture

**Pin Configuration (Arduino Uno):**

Different Arduino are designed to serve different purposes but some basic components are needed in every Arduino. Note that Arduino Uno is the most used board and is the most common choice for different users. Let us study the internal structure of the Arduino Uno model.

- **Processor:** 16 MHz ATmega16U2
- **Flash memory:** 32KB
- **Ram:** 2KB
- **Voltage Needed:** 5V
- **Input Voltage:** 7-12V
- **Analog input pins:** 6
- **Number of digital I/O:** 14 with 6 of them being PWM pins

**Components of Arduino:**

Let's study the basic components of Arduino:

**Breadboard:** Breadboards are used to provide a base for setting up the connecting components together. If you look at a breadboard, is a plastic block made up of holes that are left for making connections using wires. The internal of the breadboard consists of different connections that are hidden. They are generally used for smaller circuits.

**LEDs(Light Emitting Diode):** LEDs are small devices that illuminate when a small voltage is supplied to them. These LEDs come in a variety of colors ranging from red to green. These LEDs can be used for testing minimum voltages since they don't burn for a long period.

**Photo Resistor:** Many Arduino contain a photo resistor that is used for measuring changes in light using Arduino.

**Tactile Switch:** This switch resembles a button that is used to open or close a circuit like any other switch. When the button is turned on, the voltage of Arduino increases from 0V

to +5V. This voltage change acts as a trigger for the Arduino and an Arduino detects this change momentarily since the switch is then turned off when the button is released.

**Microcontroller:** An Arduino consists of a microcontroller that controls the whole functioning of the Arduino by generating an apt output corresponding to the input code. Depending on the type of Arduino board you are using, you can select a microcontroller that fits well.

**Resistors**: Resistors are used to resist the flowing electricity. Resistors are often bought to set a limit to the current flowing in the circuit thereby protecting the components of the circuit from getting burnt due to excessive current.

**Jumper Wires:** Jumper wires are thin wires covered with a plastic covering for insulation. These wires are used for connecting different components in the breadboard.

| Pin Type | Label/Range | Description |
|---|---|---|
| **Digital I/O** | D0 to D13 | Used for digital input/output. Can read or write HIGH/LOW. |
| **PWM Output** | D3, D5, D6, D9, D10, D11 | Support PWM via analogWrite() for dimming LEDs, motor control. |
| **Analog Inputs** | A0 to A5 | Read analog voltages (0–5V), returns 0–1023 using 10-bit ADC. |
| **Power Pins** | 3.3V, 5V, GND, VIN | Provide power or accept external power. |
| **AREF** | AREF | Analog reference voltage for ADC. |
| **Reset** | RESET | Resets microcontroller. Triggered by button or external pin. |
| **Communication** | TX (D1), RX (D0) | UART serial communication (TX = Transmit, RX = Receive). |

**Key Internal Blocks of Arduino Uno:**

- **CPU Core**: 8-bit AVR microcontroller (ATmega328P)
- **Timers**: 2 × 8-bit and 1 × 16-bit — used for delays, PWM, counters.

- **Serial Communication Interfaces**:
  - **UART** (TX/RX)
  - **SPI** (D10–D13)
  - **I2C** (A4–SDA, A5–SCL)
- **ADC**: 6-channel, 10-bit Analog to Digital Converter
- **Oscillator**: 16 MHz crystal oscillator for accurate timing

**Detailed Pin Roles:**

**Digital Pins (D0–D13)**

- Used for digital HIGH/LOW input or output
- D0 & D1 are reserved for serial communication (RX, TX)

**PWM Pins (~D3, D5, D6, D9, D10, D11)**

- Use analogWrite() for PWM output
- Applications: LED brightness, servo motors, fan speed

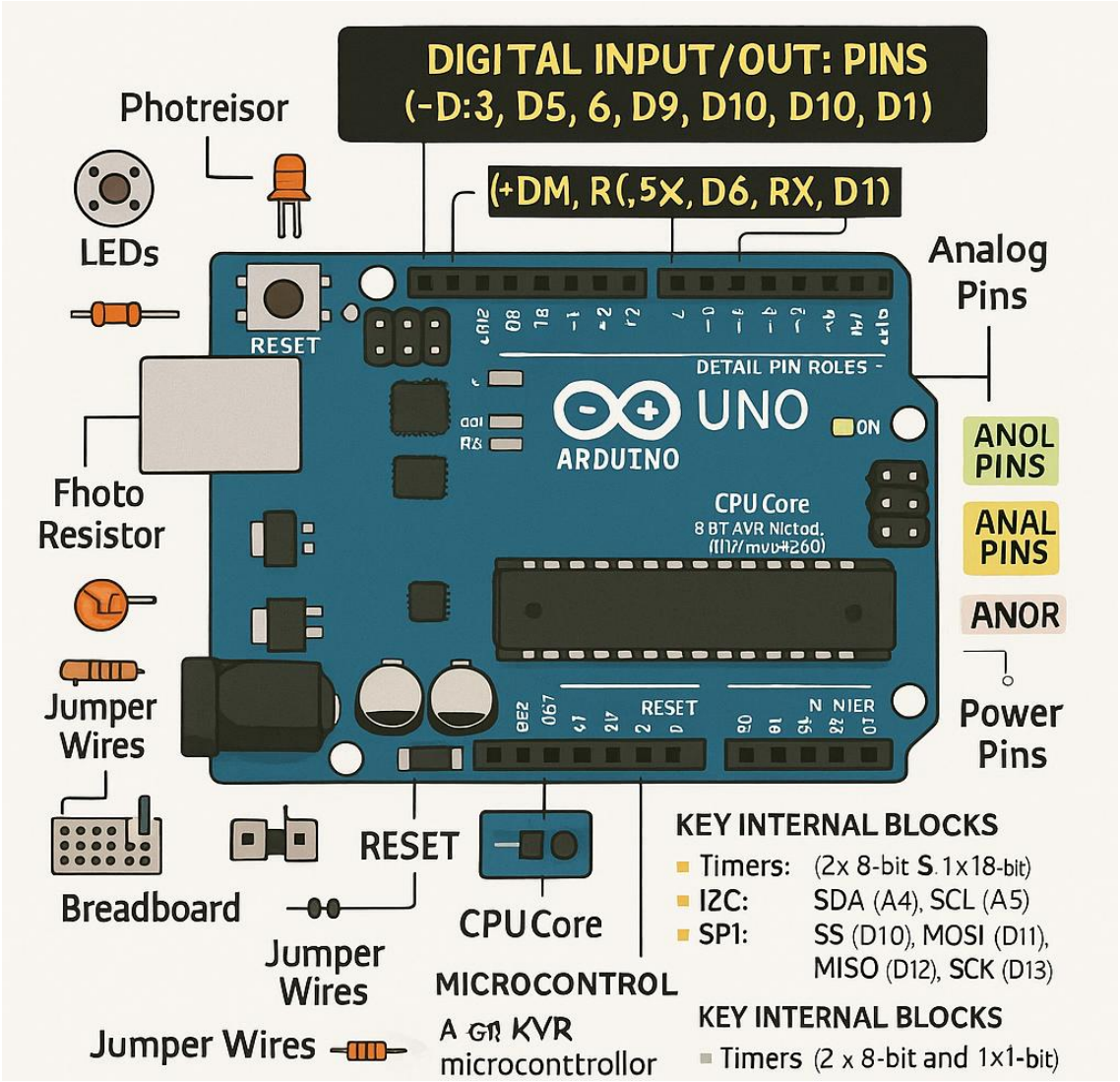
**Analog Pins (A0–A5)**

- Input analog voltages (0–5V)
- Useful for sensors (temperature, LDR, potentiometers)

**Power Pins**

| Pin | Function |
|---|---|
| **5V** | Provides 5V regulated power to external components |
| **3.3V** | Supplies 3.3V power (less current available) |
| **GND** | Ground pin |
| **VIN** | Voltage input (7–12V) when powering externally |

# Unit – 2 : Getting Started with Arduino



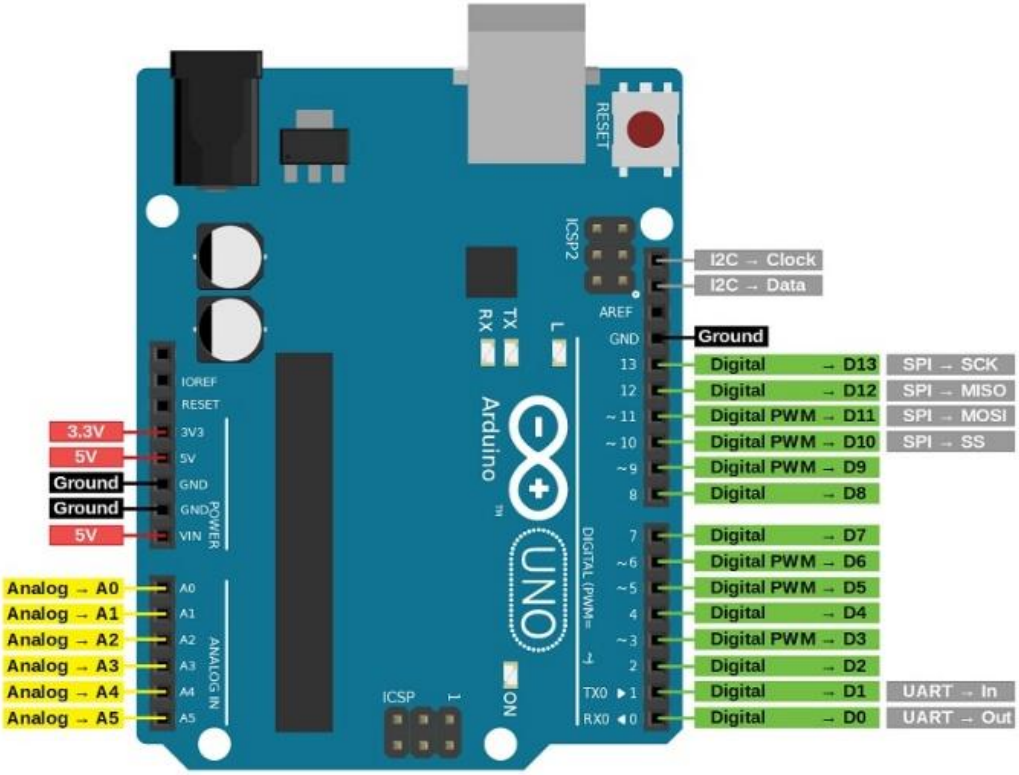

**Communication Pins:**

- **UART**:
  - TX (D1): Transmits data
  - RX (D0): Receives data
- **I2C**:
  - SDA (A4): Data line
  - SCL (A5): Clock line
- **SPI**:
  - SS (D10): Slave Select
  - MOSI (D11): Master Out Slave In
  - MISO (D12): Master In Slave Out
  - SCK (D13): Serial Clock

# Unit – 2 : Getting Started with Arduino

**AREF Pin:**

- Use with analogReference() in code
- External reference voltage for analog input conversion




**Arduino Architecture (Based on Arduino Uno):**

The **Arduino Uno** is an open-source microcontroller board built around the **ATmega328P** chip. It includes hardware components and software support for embedded development.

**Microcontroller: ATmega328P**

- **CPU**: 8-bit AVR RISC processor
- **Clock Speed**: 16 MHz
- **Flash Memory**: 32 KB (for program storage)
- **SRAM**: 2 KB (for runtime data)
- **EEPROM**: 1 KB (non-volatile storage)

# Unit – 2 : Getting Started with Arduino

**Digital I/O Pins (D0–D13):**

- Used for general digital input/output
- 6 pins support **PWM** (D3, D5, D6, D9, D10, D11)

**Analog Input Pins (A0–A5):**

- Connected to a **10-bit ADC** (Analog to Digital Converter)
- Reads analog voltage (0–5V), returns 0–1023

**Power Supply Section:**

- Can be powered via:
  - **USB** (5V)
  - **Vin** (7–12V external adapter)
- Contains a **voltage regulator** to convert input to stable 5V or 3.3V

**Timers:**

- **Timer0 and Timer2**: 8-bit
- **Timer1**: 16-bit
- Used for timekeeping (delay(), millis()), PWM, and interrupts

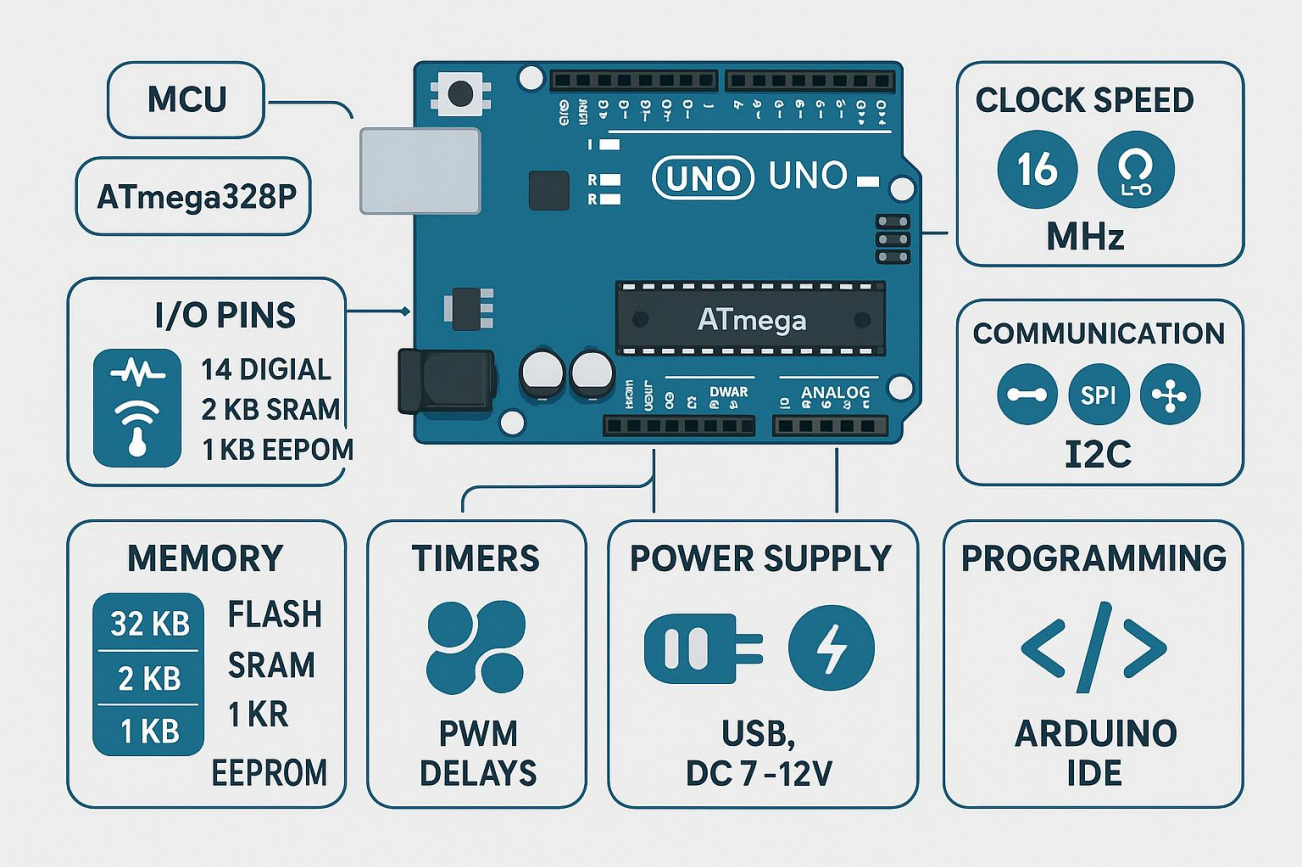
**Communication Interfaces:**

| Protocol | Pins Used | Purpose |
|---|---|---|
| **UART** | D0 (RX), D1 (TX) | Serial communication with PC |
| **SPI** | D10 (SS), D11 (MOSI), D12 (MISO), D13 (SCK) | For SPI-based sensors and modules |
| **I2C** | A4 (SDA), A5 (SCL) | For communicating with I2C devices |

# Unit – 2 : Getting Started with Arduino

**Other Components:**

- **USB Interface**: For uploading code and serial communication
- **Reset Circuit**: Manually or automatically restarts the board
- **Crystal Oscillator (16 MHz)**: Provides clock to the MCU
- **Voltage Regulator**: Maintains 5V or 3.3V levels for safe operation




**Summary:**

| Component | Description |
|---|---|
| **MCU** | ATmega328P – brain of Arduino Uno |
| **Clock Speed** | 16 MHz |
| **Memory** | Flash (32 KB), SRAM (2 KB), EEPROM (1 KB) |
| **I/O Pins** | 14 Digital, 6 Analog |
| **Communication** | UART, SPI, I2C |
| **Timers** | 3 Timers (used in PWM, delays, interrupts) |
| **Power Supply** | USB or DC (7–12V), onboard regulators |
| **Programming** | Arduino IDE with simplified C++ |

# Unit – 2 : Getting Started with Arduino

## Arduino Uno: Device and Platform Features

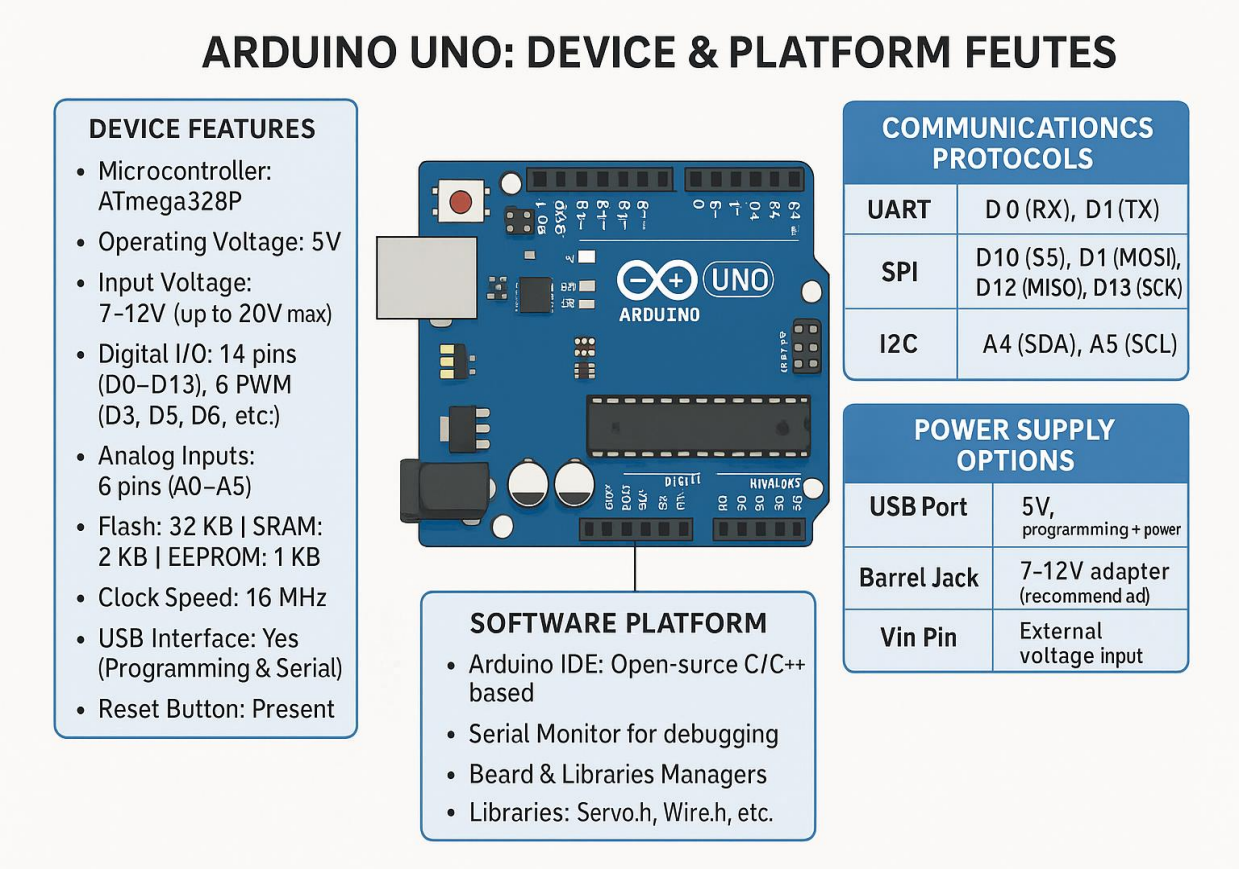
**Device Features (Arduino Uno Board):**

The **Arduino Uno** is a widely used development board based on the **ATmega328P** microcontroller, suitable for IoT, robotics, and embedded systems.

**General Specifications:**

| Feature | Specification |
|---|---|
| **Microcontroller** | ATmega328P |
| **Operating Voltage** | 5V |
| **Input Voltage** | 7–12V (recommended), up to 20V (absolute max) |
| **Digital I/O Pins** | 14 (6 support PWM) |
| **Analog Input Pins** | 6 (A0 to A5) |
| **Flash Memory** | 32 KB (0.5 KB used by bootloader) |
| **SRAM** | 2 KB |
| **EEPROM** | 1 KB |
| **Clock Speed** | 16 MHz |
| **USB Interface** | Yes – for programming and serial communication |
| **Reset Button** | Resets the program execution |

# Unit – 2 : Getting Started with Arduino




**Platform Features:**

Arduino combines hardware with a powerful and beginner-friendly **software ecosystem**.

**Arduino IDE (Integrated Development Environment):**

- Free, open-source coding platform
- Code in simplified **C/C++**
- Supports uploading via USB
- Comes with:
    - Built-in **examples**
    - **Serial Monitor** for real-time debugging
    - **Board Manager** for adding new boards

# Unit – 2 : Getting Started with Arduino

**Arduino Libraries:**

- Ready-to-use code for sensors, displays, actuators
- Examples:
    - Servo.h – for controlling servos
    - Wire.h – I2C communication
    - DHT.h – temperature/humidity sensors
- Installable from **Library Manager**

**Board Compatibility:**

- Supports many boards: **Uno, Mega, Nano, Due**
- Add custom boards via Board Manager

**Communication Capabilities:**

Arduino Uno supports three major communication protocols:

| Protocol | Use Case | Pins |
|---|---|---|
| **UART** | Serial monitor, serial devices | D0 (RX), D1 (TX) |
| **SPI** | High-speed modules (e.g. SD card) | D10 (SS), D11 (MOSI), D12 (MISO), D13 (SCK) |
| **I2C** | Multi-device, 2-wire protocol | A4 (SDA), A5 (SCL) |

**Power Supply Options:**

Arduino Uno can be powered in three ways:

| Method | Details |
|---|---|
| **USB Cable** | 5V input and data transmission to IDE |
| **Barrel Jack** | 7–12V via adapter (recommended) |
| **Vin Pin** | Direct external voltage input (regulated onboard) |

# Unit – 2 : Getting Started with Arduino

## Concept of Digital and Analog Ports – Arduino

**Overview:**

Arduino boards come with **Digital** and **Analog** pins, used to interface with a wide range of external electronic components such as LEDs, sensors, motors, and buttons.

**Digital Ports:**

**Definition:**

Digital pins can **read or write binary values**:

- **HIGH (1)** = 5V
- **LOW (0)** = 0V

Used for interacting with digital devices like buttons, relays, and LEDs.

**Key Functions:**

- pinMode(pin, INPUT/OUTPUT)
- digitalRead(pin)
- digitalWrite(pin, HIGH/LOW)

**Common Use Cases:**

| Application | Function |
|---|---|
| **LED Control** | Turn LEDs ON or OFF |
| **Button Input** | Detect press or release (HIGH/LOW) |
| **Relay Switching** | Activate appliances |
| **Digital Modules** | Receive/send digital signals (e.g., IR sensor) |

# Unit – 2 : Getting Started with Arduino

**Analog Ports:**

**Definition:**

Analog pins (A0 to A5 on Uno) **read varying voltages** between 0V and 5V. These voltages are converted to digital values using a 10-bit **ADC** (Analog-to-Digital Converter).

**Voltage to Digital Mapping:**

- Voltage Range: 0V to 5V
- Digital Value Range: 0 to 1023

**Key Function:**

- analogRead(pin)

Analog pins only **read** values. To simulate analog output, Arduino uses **PWM** via analogWrite() on specific digital pins.

**Common Use Cases:**

| Application | Function |
|---|---|
| **Potentiometer** | Measure turning position |
| **Temperature Sensor** | Read varying voltage based on temperature (e.g., LM35) |
| **Light Sensor (LDR)** | Measure light intensity |
| **Joystick Module** | Read X/Y movement axes |

# Unit – 2 : Getting Started with Arduino

## Arduino Interfacing Board

**Definition:**

The **Arduino Interfacing Board** refers to external modules or shields that connect to an Arduino to **enhance its capabilities**. These include **relay boards, sensor shields, motor drivers**, displays, and communication modules that help the Arduino interact with the physical world.

These boards:

- Provide **plug-and-play** expansion
- Simplify **wiring and coding**
- Enable communication with external **devices or networks**

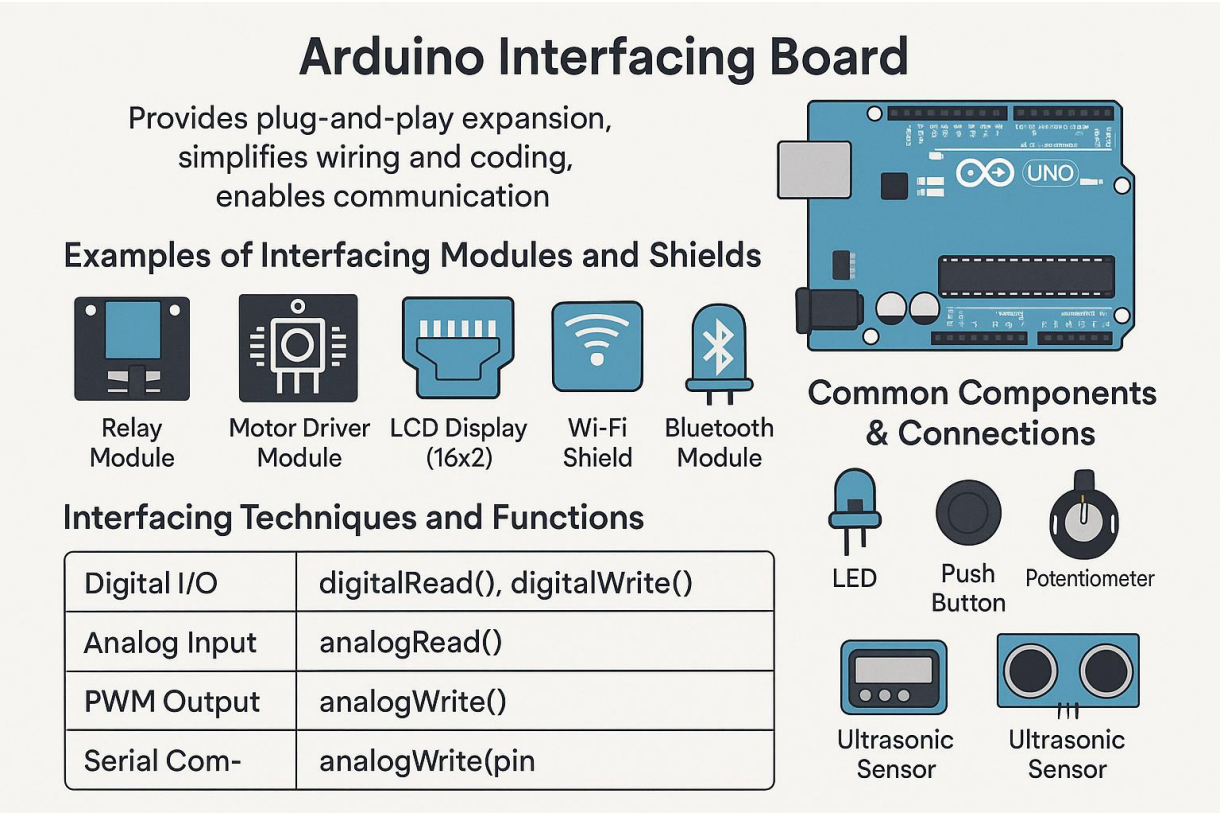
**Examples of Interfacing Modules and Shields:**

| Module/Shield | Purpose |
|---|---|
| **Relay Module** | Control high-voltage AC appliances like fans, lamps |
| **Motor Driver Module** | Drive DC motors, stepper motors, or servos (e.g., L298N, L293D) |
| **Sensor Shield** | Easily connect multiple sensors to Arduino pins |
| **LCD Display (16x2)** | Show sensor readings (e.g., temperature, humidity) |
| **Wi-Fi Shield (ESP8266)** | Enable wireless communication via internet |
| **Ethernet Shield** | Connect Arduino to a LAN for network communication |
| **Bluetooth Module (HC-05)** | Wireless data transfer with phones, PCs, or other microcontrollers |

# Unit – 2 : Getting Started with Arduino

**Interfacing Techniques and Functions**

| Technique | Arduino Function | Purpose |
|---|---|---|
| **Digital I/O** | digitalRead(), digitalWrite() | Read button presses or control LEDs, relays |
| **Analog Input** | analogRead() | Read values from analog sensors (e.g., potentiometer) |
| **PWM Output** | analogWrite() | Dim LEDs, control motor speeds, servo position |
| **Serial Communication** | Serial.begin(), Serial.print() | Communicate with PC, sensors (e.g., Bluetooth) |
| **I2C Communication** | Wire.h library | Connect to devices like LCDs, RTCs using 2 wires |
| **SPI Communication** | SPI.h library | High-speed communication with SD cards, sensors |



# Unit – 2 : Getting Started with Arduino

**Arduino Interfacing Board Examples:**

Arduino's pins allow **easy interfacing** with sensors and actuators.

**Common Components & Connections:**

| Component | Interface Type | Purpose |
|---|---|---|
| LED | Digital Output | Visual feedback or status light |
| Push Button | Digital Input | User input detection |
| Potentiometer | Analog Input | Adjustable voltage source |
| Ultrasonic Sensor | Digital | Distance measurement |
| DHT11 Sensor | Digital | Temperature and humidity sensing |
| Servo Motor | PWM | Controlled angular motion |

**Communication Protocols:**

| Protocol | Use Case | Pins Involved |
|---|---|---|
| **UART** | Serial Monitor, Serial Comm | D0 (RX), D1 (TX) |
| **SPI** | High-speed device communication | D10 (SS), D11 (MOSI), D12 (MISO), D13 (SCK) |
| **I2C** | 2-wire communication for sensors | A4 (SDA), A5 (SCL) |

**Benefits of Using Interfacing Boards:**

- Simplifies wiring of complex circuits
- Reduces development time
- Makes prototyping **modular and reusable**
- Enables Arduino to connect with a wide **variety of hardware**
- Promotes better **organization and debugging**

Arduino interfacing boards are **essential** for building real-world IoT and embedded systems. They bridge the gap between **software code and physical components**, allowing your Arduino to **sense, process, and act** in its environment.