pinMode:

In Arduino programming, the **pinMode()** function is used to configure a specific digital pin on the Arduino board as either an input or an output. This function sets the mode of operation for a particular pin, allowing you to control whether the pin will be used to read digital input signals from sensors, switches, etc., or to send digital output signals to control devices like LEDs, motors, and relays.

The syntax for the **pinMode()** function is as follows:

pinMode(pin, mode);

Where:

* **pin**: The number of the pin you want to configure. This is typically an integer value representing the physical pin number on the Arduino board.
* **mode**: The mode you want to set for the specified pin. This can be either **INPUT** or **OUTPUT**, represented by constants in Arduino.

Here's how you might use the **pinMode()** function:

|  |
| --- |
| int ledPin = 13; // Assuming you want to control an LED connected to pin 13  void setup() {  pinMode(ledPin, OUTPUT); // Configure pin 13 as an output  }  void loop() {  digitalWrite(ledPin, HIGH); // Turn on the LED  delay(1000); // Wait for 1 second  digitalWrite(ledPin, LOW); // Turn off the LED  delay(1000); // Wait for 1 second again  } |
|  |

In this example, the **pinMode()** function is used in the **setup()** function to configure pin 13 as an output. This allows you to use the **digitalWrite()** function to control the state of the LED connected to that pin. By setting the pin to **HIGH**, you turn on the LED, and by setting it to **LOW**, you turn it off.

Remember that not all pins on the Arduino board are capable of both input and output modes. Some pins have specific functionalities like analog input, PWM (Pulse Width Modulation) output, or communication (UART, SPI, I2C). Therefore, it's important to refer to the Arduino documentation or the board's pinout diagram to understand the capabilities of each pin.

Why we use digitalWrite why not anologWrite:

In the program you provided, **digitalWrite()** is used instead of **analogWrite()** because the pin being controlled (pin 13 in this case) is a digital pin and not a pin capable of PWM (Pulse Width Modulation) output.

Here's the breakdown:

1. **Digital Pins**: Digital pins on Arduino can be in one of two states: HIGH (5V) or LOW (0V). They are primarily used for simple on/off control of devices like LEDs, relays, and switches. **digitalWrite()** is used to set the state of a digital pin. When you use **digitalWrite(pin, HIGH)**, the pin is set to a HIGH voltage (ON state), and when you use **digitalWrite(pin, LOW)**, the pin is set to a LOW voltage (OFF state).
2. **Analog Pins**: Some pins on Arduino boards are capable of PWM output, which can simulate an analog voltage by rapidly switching between HIGH and LOW states with varying duty cycles. This is useful for smoothly controlling devices that require a range of voltages, like dimming an LED or controlling the speed of a motor. For these pins, you would use **analogWrite()**.

In the program you provided, pin 13 is being used as a digital output to control an LED. Since you're only turning the LED on (HIGH state) and off (LOW state), you use **digitalWrite()** because pin 13 is a standard digital pin, not one capable of generating PWM signals.

If you were working with an LED or device that requires variable brightness or speed control, and the pin you're using supports PWM (like pins 3, 5, 6, 9, 10, and 11 on most Arduino boards), then you would indeed use **analogWrite()** to achieve the desired effect