Switch debouncing in Arduino refers to the technique of handling the mechanical "bouncing" that happens when a physical switch is pressed or released. Without debouncing, an Arduino might read multiple presses instead of one, because the switch contacts can open and close several times quickly before settling.

There are two common methods to debounce a switch in Arduino: **software debouncing** and **hardware debouncing**.

**1. Software Debouncing**

In software debouncing, you can ignore the switch state changes that happen too quickly. The basic idea is to wait a small amount of time after detecting a change, then check the state again to ensure the bounce has finished.

Here’s a simple example using software debouncing:

const int buttonPin = 2; // Pin connected to the button

const int ledPin = 13; // Pin connected to an LED

int buttonState = LOW; // Variable to store the button state

int lastButtonState = LOW; // Variable to store the last button state

unsigned long lastDebounceTime = 0; // Timestamp for the last debounce time

unsigned long debounceDelay = 50; // Debounce time in milliseconds

void setup() {

pinMode(buttonPin, INPUT);

pinMode(ledPin, OUTPUT);

}

void loop() {

// Read the button state

int reading = digitalRead(buttonPin);

// Check if the button state has changed

if (reading != lastButtonState) {

lastDebounceTime = millis(); // Reset debounce timer

}

// Only update the button state if the debounce time has passed

if ((millis() - lastDebounceTime) > debounceDelay) {

// Update the button state

if (reading != buttonState) {

buttonState = reading;

// If the button is pressed, toggle the LED

if (buttonState == HIGH) {

digitalWrite(ledPin, !digitalRead(ledPin));

}

}

}

// Store the current reading as the last button state for the next loop

lastButtonState = reading;

}

**Explanation:**

* **debounceDelay**: This sets how long the system waits before it considers the switch stable. In this example, it’s 50 milliseconds.
* **millis()**: Tracks the current time in milliseconds. This is used to compare the time elapsed since the last button state change.
* **State check logic**: If the button’s state stays the same for more than 50 ms, the code assumes the switch is stable.