Using Intervals in Microcontrollers

Imagine you're juggling two things at once, like making a sandwich while also keeping an eye on the clock to catch your favorite TV show. You can't just stop making the sandwich to stare at the clock; you need to keep working and check the time every so often.

That's similar to how we use intervals in microcontrollers. Instead of pausing the entire program with delay(), we check for events at regular intervals. This allows the microcontroller to be more responsive and handle multiple tasks.

Here's how it works:

1. **Set an interval:** We decide how often we want to check for events (e.g., every 100 milliseconds).
2. **Use a timer:** The microcontroller has a built-in timer that keeps track of time.
3. **Check the timer:** In the main loop, we check if the timer has reached the set interval.
4. **Take action:** If the interval has elapsed, we perform the necessary actions (e.g., toggle an LED, read a sensor).
5. **Reset the timer:** We reset the timer to start counting again.

This way, the microcontroller can continue to perform other tasks while still keeping track of the interval.

Example:

Let's say you want to blink an LED every second. Instead of using delay(1000), you can use a timer to keep track of the time and toggle the LED state every second. This allows the microcontroller to check for other events (like button presses) while the LED is blinking.

Key Points:

- Intervals allow microcontrollers to be more responsive and handle multiple tasks.
- They are used to perform actions at regular intervals without pausing the entire program.
- Timers are used to keep track of time and trigger actions at specific intervals.

By understanding intervals, you can create more efficient and responsive microcontroller programs.

Arduino Programming - Interval Tutorial

Interval Task Scheduling (Arduino Crash Course 20)