Arduino-project 007 – Multitasking

Jun 1, 2018 | Example code, Guides | 0 comments

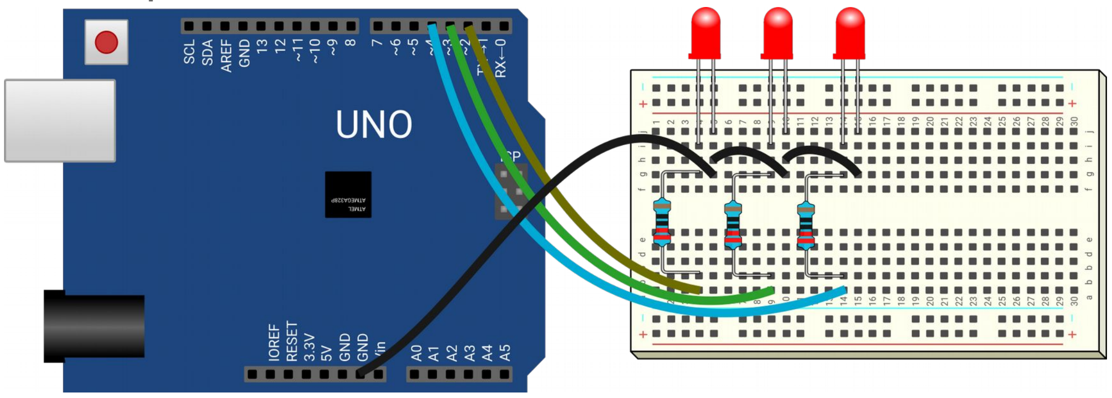
For simple projects it's no problem to do only one thing at a time and waiting until it is done before doing the next thing, but for more advanced projects you sometimes need to do several things at the same time, so called multitasking. You may for example want to read in data from a radio reciever, drive a stepper motor, flash a LED and show status text on a display, all at the same time. There are several ways to accomplish this, but the method we show here is very popular because it is relatively simple to understand and program and gives good results. The method is in fact inspired by how professional industrial systems work and is therefore very well tested. The basic idea is to make sure that all tasks that the Arduino board should perform are kept small and as fast as possible to execute and that you never actively wait for anything. It is therefore vital to never use the `delay()`-function in your code as this will completely block any other task from being run. Instead you let the processor continuously compare timer values and perform the different tasks whenever the appropriate amount of time has passed.

How is it done?

Simply put, you have a series of `if`-statements that compares time values and let the processor run through them over and over again as fast as possible. When a timer value exceeds a preset threshold the associated function is called and executed and then the control is returned to the main program with the If-statements. The threshold is then set to the next point in time when the function should be run and the main program continues its loop.

The circuit

In our example we want three LEDs to blink at different speeds. This may at first seem simple, but if we want them to blink independently of each other we need to use some sort of multitasking. Start by building the circuit in the diagram below. Then enter the code from below into Arduinos development environment and upload it to the Arduino board.



Description of the code

In the code we set up three separate functions, `blink_1()`, `blink_2()` and `blink_3()` that we will run in parallel. You can of course use this code as basis for other projects and switch the functions for your own or add more. We also set up a number of variables to keep track of how often each function should run and to keep track of the state of each LED (on or off). The main program resides in `loop()` as usual and this code will run as fast as possible, thousands of times per second. The loop begins with getting the current time with the `millis()` function. This function returns the number of milliseconds that have elapsed since the Arduino board was powered on, or since the last reset. After this follows three separate `if`-statements where we compare the current time with the different timer values for the blink functions. The first time all these values will be 0 meaning that all three `if`-statements will be true and the three blink-functions will be run. When each blink function has finished running we set the timer values to be current time plus the number milliseconds we want to wait until we run it the next time.

The next time the loop runs `current_time` will have increased slightly, exactly how much depends on how long it took to run the three blink-functions, but not so much that `current_time` is greater than any of the timer variables. Therefore all `if`-statements will be false and won't run their associated function. This will now repeat over and over again until `current_time`has increased so much that it is greater than one of the timer values and the associated function is run and its timer value is again increased. Most of the time the processor will do nothing but running around the main loop and comparing timer values, but occasionally one of the blink functions will run. It's important to remember that even though this method makes it seem as if the processor do multiple things at the same time it will never do more than once thing at a time. It's just so very fast that it seems to use it's doing them concurrently. This is what makes it so important that all functions are fast and to never use `delay()` as this will bind up the processor and prevent the other functions from running.