**Arduino Timer**

**Description**

The library does not interfere with the built-in timers, it just uses 'millis()' in a crude type of scheduler to decide when something needs doing.

**Examples**

The Arduino 'delay' function is both a blessing and a curse. Its great for showing beginners how to make an LED flash. But as soon as you get more complex and start slowing down your 'loop' function you will run into problems.

***A classic example is turning a relay on for 10 minutes. The 'delay'-way looks like this:***

*int pin = 13;*

*void setup()*

*{*

*pinMode(13, OUTPUT);*

*digitalWrite(pin, HIGH);*

*delay(10 \* 60 \* 1000);*

*digitalWrite(pin, LOW);*

*}*

*void loop()*

*{*

*}*

*The disadvantage of the delay approach is that nothing else can go on while the 'delay' is happening. You cannot update a display, or check for key presses for example.*

***'Timer' library version looks like this:***

*#include "Timer.h"*

*Timer t;*

*int pin = 13;*

*void setup()*

*{*

*pinMode(pin, OUTPUT);*

*t.pulse(pin, 10 \* 60 \* 1000, HIGH); // 10 minutes*

*}*

*void loop()*

*{*

*t.update();*

*}*

The 'pulse' method takes arguments of a pin to change, the period to change it for and its initial state.

The call to t.update() will take a matter of microseconds to run, unless the appropriate period of time has passed.

***Lets look at another example that uses two timer events. One to flash an LED and another that reads A0 and displays the result in the Serial Monitor.***

*#include "Timer.h"*

*Timer t;*

*int pin = 13;*

*void setup()*

*{*

*Serial.begin(9600);*

*pinMode(pin, OUTPUT);*

*t.oscillate(pin, 100, LOW);*

*t.every(1000, takeReading);*

*}*

*void loop()*

*{*

*t.update();*

*}*

*void takeReading()*

*{*

*Serial.println(analogRead(0));*

*}*

The first thing to notice is that we are using a callback function called 'takeReading'. We connect it to the Timer using the 'every' command, which in this case, will call the function every second.

We have also attached another event to the timer using the method 'oscillate'. This will cause the LED to toggle state every 100 milliseconds.

***Each of the events has an integer ID associated with it, so that you can stop an event, as we do in this example below, which will write to the serial monitor every 2 seconds, flash the LED and after 5 seconds, stop the LED flashing fast, and flash it 5 times slowly.***

*#include "Timer.h"*

*Timer t;*

*int ledEvent;*

*void setup()*

*{*

*Serial.begin(9600);*

*int tickEvent = t.every(2000, doSomething);*

*Serial.print("2 second tick started id=");*

*Serial.println(tickEvent);*

*pinMode(13, OUTPUT);*

*ledEvent = t.oscillate(13, 50, HIGH);*

*Serial.print("LED event started id=");*

*Serial.println(ledEvent);*

*int afterEvent = t.after(10000, doAfter);*

*Serial.print("After event started id=");*

*Serial.println(afterEvent);*

*}*

*void loop()*

*{*

*t.update();*

*}*

*void doSomething()*

*{*

*Serial.print("2 second tick: millis()=");*

*Serial.println(millis());*

*}*

*void doAfter()*

*{*

*Serial.println("stop the led event");*

*t.stop(ledEvent);*

*t.oscillate(13, 500, HIGH, 5);*

*}*

***You can attach up to 10 events to a timer.***

**Installation**

1. Download <https://github.com/JChristensen/Timer/archive/v2.1.zip>
2. (In the Arduino IDE) Sketch > Include Library > Add .ZIP Library > select the downloaded file > Open

**Usage**

int every(long period, callback)

Run the 'callback' every 'period' milliseconds.

Returns the ID of the timer event.

int every(long period, callback, int repeatCount)

Run the 'callback' every 'period' milliseconds for a total of 'repeatCount' times. Returns the ID of the timer event.

int after(long duration, callback)

Run the 'callback' once after 'period' milliseconds. Returns the ID of the timer event.

int oscillate(int pin, long period, int startingValue)

Toggle the state of the digital output 'pin' every 'period' milliseconds. The pin's starting value is specified in 'startingValue', which should be HIGH or LOW. Returns the ID of the timer event.

int oscillate(int pin, long period, int startingValue, int repeatCount)

Toggle the state of the digital output 'pin' every 'period' milliseconds 'repeatCount' times. The pin's starting value is specified in 'startingValue', which should be HIGH or LOW.

Returns the ID of the timer event.

int pulse(int pin, long period, int startingValue)

Toggle the state of the digital output 'pin' just once after 'period' milliseconds. The pin's starting value is specified in 'startingValue', which should be HIGH or LOW. Returns the ID of the timer event.

int stop(int id)

Stop the timer event running. Returns the ID of the timer event.

int update()

Must be called from 'loop'. This will service all the events associated with the timer.