Raspberry Pi Pico - Non Blocking Timer (Neotimer)

Library footprint: approx 3kB

This library implements a non-blocking delay function to use in your program. Originally developed by me for Arduino and Parallax Propeller 2.

Why use Neotimer?

In the microcontroller world, delays are used extensively to control different aspects of timing control. This is usually implemented using functions such as time.sleep(). But when you call `time.sleep()`, the processor stops everything it is doing until this delay is completed. That is called a blocking delay, because it blocks the processor until it finishes.

There are many situations where this is undesirable and you need to implement a workaround.

This library provides a way to use time delays without blocking the processor, so it can do other things while the timer ends up. This is called a non-blocking delay timer.

Neotimer provides a set of basic functionalities to implement different ways of timing in a program. You can use the timer in the following ways:

A) Start-Stop-Restart Timer -

You can start, stop and restart the timer. Once started it will count towards the initialization time. You can check if it finished counting by calling `finished()`.

- `start()` will start the timer.
- `stop()` will stop the timer. It will also return the elapsed milliseconds since it was started.
- `finished()` will return True if the timer reached the initialization time.
- `restart()` will restart the timer.

```
myTimer = Neotimer(200) #<-------- Initializes a 200ms timer

if collision_detected:
   myTimer.start()     #<--------- Starts timer
   led.on()

if myTimer.finished():
   led.off()  #<--------- Called after 200ms
```

B) Periodic trigger (`repeat_execution()`)

The timer can be used to periodically trigger the execution of a block of code. The following example will toggle pin 25 every 500ms

```
led_pin = Pin(25,Pin.OUT)
myTimer = Neotimer(500)   #<---------------- Initializes a 500ms timer
```

```
while True:
    if(myTimer.repeat_execution()):
        led_pin.toggle()  #<---------------- Called every 500ms
```

C) Periodic trigger with count (`repeat_execution(count)`)

You can also trigger the execution of some code a specific amount of times. The following example will toggle pin 56 every 500ms, only 3 times. After 3 times, the timer will not repeat the code until a reset is issued. To reset the repetitions use `reset_repetitions()` .

```
led_pin = Pin(25,Pin.OUT)
button = Pin(2, Pin.IN)

myTimer = Neotimer(500)   #<---------------- Initializes a 500ms timer

while True:
    if(myTimer.repeat_execution(3)): #<--- Only repeat 3 times
        led_pin.toggle()  #<---------------- Called every 500ms
    if(button.value()):
        myTimer.reset_repetitions() #<--- Reset repetitions
```

D) Debouncer for signals

You can debounce a signal using debouce_signal. The debouncing period will be duration.

In this example, the button pin value signal will be debounced for 250 milliseconds:

```
button = Pin(2, Pin.IN)
presses = 0
myTimer = Neotimer(250) #<--------------- Initializes a 250ms timer

while True:
    if myTimer.debounce_signal(button.value()): #<----- button pressed signal
        presses += 1
        print(presses)
```

E) Hold signal

Neotimer can be used to detect a signal hold. For example holding down a button for 3 seconds:

```
BUTTON_A = Pin(20,Pin.IN)
```

```
led = Pin(25,Pin.OUT)

myTimer = Neotimer(3000) #<---- 3 seconds hold time

while True:
    if myTimer.hold_signal(BUTTON_A.value()):
        led.on()
    else:
        led.off()
```

F) Waiting

The following example will turn on the led for 1000ms each time the button is pressed

```
from machine import Pin
from neotimer import *

button = Pin(2, Pin.IN)
led = Pin(25,Pin.OUT)
led.off()

# Two timers, one for debouncing the button signal
# and the other for generating a pulse using waiting()
debouncer = Neotimer(200)
pulse = Neotimer(1000)

while True:
    if debouncer.debounce_signal(button.value()):
        pulse.start()

    if pulse.waiting():  #<---- Led on during pulse time
        led.on()
    else:
        led.off()
```

Author: Jose Rullan

Date: January 24, 2022