Non-blocking Virtual Delay timer for the Arduino

Details
Last Updated: 11 February 2018

Contents[Hide]

Introduction

The standard Arduino delay() function blocks the Arduino, that is not always allowed. The standard delay has limitations too; it is not possible to use multiple delays at the same time. So I decided to developed a VirtualDelay library which has many advantages:

Advantages of the VirtualDelay library

- The delay is virtual, during the delay, the code execution is continued
- We can use multiple delays sequentially in a loop.
- We can use multiple delays simultaneously and independent of each other.
- The delay time can set in micro-seconds or milli-seconds.
- No hardware timers are used
- The library is timer-rollover safe

VirtualDelay library download

You can download the library from GitHub (https://github.com/avandalen/VirtualDelay).

Questions

Please post any questions at the Arduino forum (http://forum.arduino.cc/index.php?topic=428666.0).

Notes

- The VirtualDelay must always run inside a loop which is executed continuously.
- In contrast with the standard delay function, we need a start() and an elapsed() function.
- The delay time value can be between 0 and 2147483647 (2^31-1).

Simple blinking LED sketch with the VirtualDelay library

```
#include <Arduino.h>
#include "avdweb_VirtualDelay.h"

const byte ledPin = 13;
bool b;
VirtualDelay singleDelay; // default = millis

void setup()
{ pinMode(ledPin, OUTPUT);
}

void loop()
{ singleDelay.start(400); // calls while running are ignored
  if(singleDelay.elapsed()) digitalWrite(ledPin, b=!b); // blink the onbo
}
```

Micro seconds VirtualDelay example

```
#include "avdweb_VirtualDelay.h"

VirtualDelay delay_us(micros);

void setup()
{ Serial.begin(9600);
}

void loop()
{ DO_ONCE ( delay_us.start(1000000)) // 1s
  if(delay_us.elapsed()) Serial.print(micros());
}
```

Standard Arduino blocking delay

Here a standard blinking LED sketch. Open the serial port and you can see that printing goes very slowly.

```
#include <Arduino.h>
#include <Streaming.h>

const byte ledPin = 13;
int i;

void setup()
{ pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{ digitalWrite(ledPin, 1); // this comes after the 700ms delay
  delay(100);
  digitalWrite(ledPin, 0); // this comes after the 100ms delay
  delay(700);
  Serial << " " << i++; // since the cpu is being blocked, printing goes
}
```

Deadlock

With a sequence of VirtualDelays, each delay will wait on the foregoing by which the sequence can't start. This is a so-called deadlock. To break the deadlock, one of the delays has to start one-time. To do so, I have built a macro DO_ONCE. This is carried out only once in a loop, see the following example:

Using 3 VirtualDelays in sequence

Note: You can use the virtual delay to generate continuously timing signals like the example below. But this is not the most appropriate way. You better use a timer and link it to an interrupt service routine.

We need three separate VirtualDelay instances: delay1, delay2 and delay3. The line with the macro DO_ONCE ensures that the sequence is started. You may use any instance e.g. delay2.start(0).

```
#include <Arduino.h>
#include <Streaming.h>
#include "avdweb_VirtualDelay.h"

VirtualDelay delay1, delay2, delay3;

void setup()
{ Serial.begin(9600);
  Serial << "\ntestSequence";
}

void loop()
{ if(delay1.elapsed()) // this sequence has a deadlock
  { Serial << "\ndelay1 200ms " << millis();
    delay2.start(100);
  }
  if(delay2.elapsed())
  { Serial << "\ndelay2 100ms " << millis();
    delay3.start(400);
  }
  if(delay3.elapsed())
  { Serial << "\ndelay3 400ms " << millis();
    delay1.start(200);
  }
  DO_ONCE(delay1.start(200)); // breaks the deadlock, you can start with
}
```

Here is the serial output:

```
delay1 200ms 200
delay2 100ms 300
delay3 400ms 700
delay1 200ms 900
delay2 100ms 1000
delay3 400ms 1400
delay1 200ms 1600
```

Using multiple delays at the same time

In this example, we use 6 VirtualDelays at the same time, this is not possible with the standard Arduino delay function. It is also showed how we can use the macro DO_ONCE multiple times in a loop.

```
#include <Arduino.h>
#include <Streaming.h>
#include "avdweb_VirtualDelay.h"

void setup()
{ Serial.begin(9600);
}

void loop()
{ static VirtualDelay delay1, delay2, delay3, delay4, delay5, delay6;
  DO_ONCE
  ( Serial << "\nDO_ONCE 1";
    delay1.start(200); // start sequence delay1 delay2 delay3
    delay4.start(550); // start one-shot delay4
    delay5.start(1250); // start one-shot delay5
  )
  if(delay4.elapsed()) Serial << "\nONE-SHOT 550ms          " << millis()
  if(delay5.elapsed()) Serial << "\nONE-SHOT 1250ms         " << millis()

  if(millis()>2250) DO_ONCE(Serial << "\nDO_ONCE 2 2250ms        " << mil

  delay6.start(750);
  if(delay6.elapsed()) Serial << "\n  Repeat delay6 750ms   " << millis()

  if(delay1.elapsed()) // sequence with deadlock
  { Serial << "\nsequence delay1 200ms   " << millis();
    delay2.start(100);
  }
  if(delay2.elapsed())
  { Serial << "\nsequence delay2 100ms   " << millis();
    delay3.start(400);
  }
  if(delay3.elapsed())
  { Serial << "\nsequence delay3 400ms   " << millis();
    delay1.start(200);
  }
}
```

Here is the serial output:

```
DO_ONCE 1
sequence delay1 200ms   200
sequence delay2 100ms   300
ONE-SHOT 550ms          550
sequence delay3 400ms   700
  Repeat delay6 750ms   750
sequence delay1 200ms   900
sequence delay2 100ms   1000
ONE-SHOT 1250ms         1250
sequence delay3 400ms   1400
  Repeat delay6 750ms   1500
sequence delay1 200ms   1600
sequence delay2 100ms   1700
sequence delay3 400ms   2100
  Repeat delay6 750ms   2250
DO_ONCE 2 2250ms        2251
sequence delay1 200ms   2300
sequence delay2 100ms   2400
sequence delay3 400ms   2800
```

One-shot example

To create a one-shot, start() may only be called once during the loop, to do so, the macro DO_ONCE is used.

```
#include <Arduino.h>
#include <Streaming.h>
#include "avdweb_VirtualDelay.h"

VirtualDelay delay1;

void setup()
{ Serial.begin(9600);
  Serial << "\ntestOneShot 2s\n";
}

void loop()
{ DO_ONCE(delay1.start(2000)) // do only one time in the loop
  if(delay1.elapsed()) Serial << millis() << "ms" ;
}
```

The library has to be timer-rollover proof

The 32 bit millis() and micros() timers counts from 0, 1, 2 to 4294967295 (2^32-1) and than rollover to 0 again. The millis() timer is going to roll over after roughly 49.7 days, the micros() rolls over every 71.6 minutes. We have to take care about this rollover, because it may happen that we start the virtual-delay just before the rollover time. After the rollover the timestamps will be messed up.

The time compare part in the VirtualDelay library is tricky, it compares 2 timestamps like this:

`if(millis() >= timeOut)`

However, this won't work, we have to use:

`if((millis()-timeOut)>=0)`

Believe it or not, in this situation, both lines are not identical! You can read all about this in the detailed story from Edgar Bonet (https://arduino.stackexchange.com/questions/12587/how-can-i-handle-the-millis-rollover/12588#12588).

History

The VirtualDelay library looks quite simple, but the development has cost me a lot of time.

The first library from 10-1-2016 didn't use a start function. The library seemed to work well but after a year it turned out that there were situations in which it failed. The reason was that I had used a simple LED test, which caused that a serious shortcoming was not discovered.

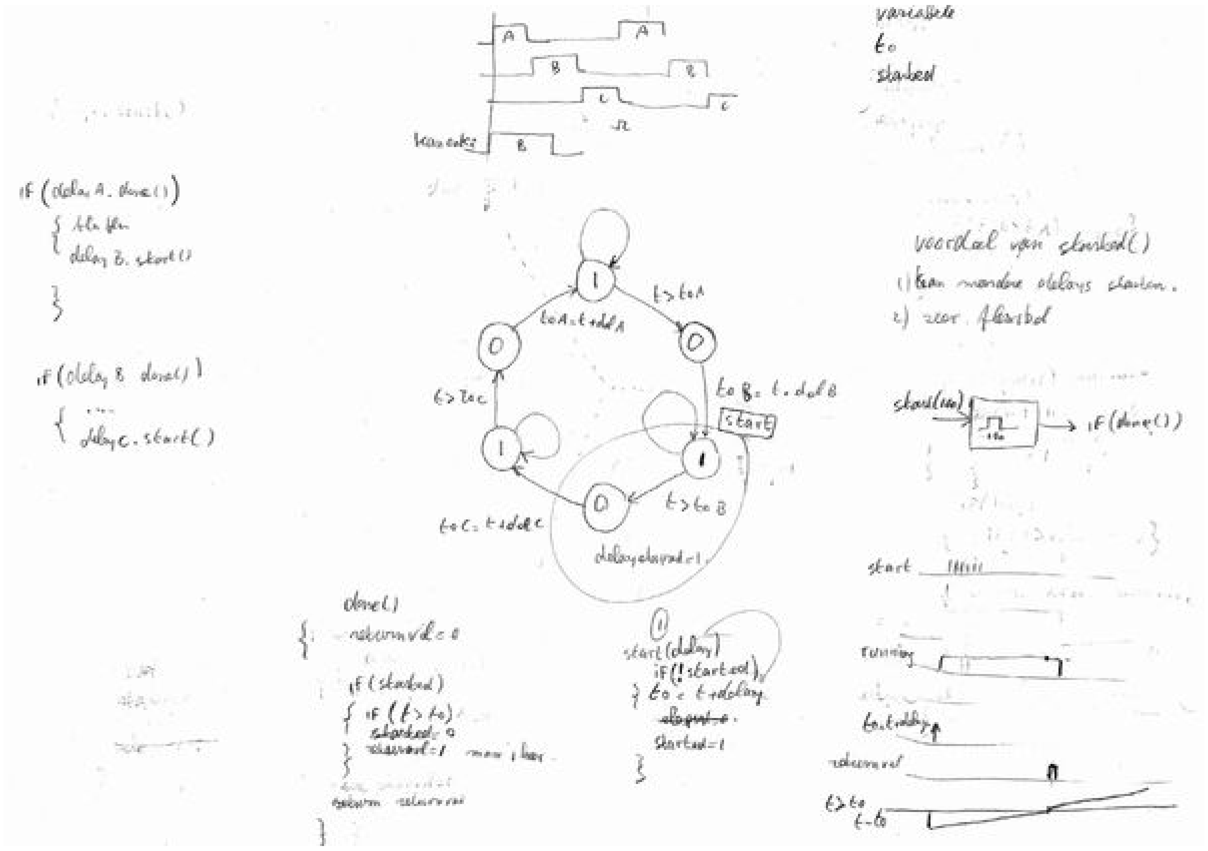
I had to make new library from scratch. This new library seemed to work well, also with a sequence of two delays.

Later tests showed that there were problems with three and more delays sequentially. This again required a completely different approach. But now it turned out that there were problems with using multiple delays simultaneously and again a complete new library had to be made.

Finally, I came to the conclusion that to solve all problems, it was necessary to use a start function and a macro DO_ONCE.

Future expansions

Extra functions may be implemented like restart, pause, resume, stop, reset. Who wants to expand the library on github?


VirtualDelay-development (/Article_files/Arduino/Virtual-delay/VirtualDelay-development.jpg)

Simple microseconds VirtualDelay

Do you have any comments? Please let me know (http://www.avdweb.nl/contact.html).