Topic: Using millis() for timing. A beginners guide (Read 137835 times)

prev_next=prev#new) - Next Topic (Https://forum.arduino.cc/index.php?topic=503368.0;prev_next=next#new)

UKHeliBob (https://forum.arduino.cc/index.php?action=profile;u=157795)

(https://forum.arduino.cc/index.php?action=profile;u=157795)

Brattain Member

Posts: 23,324

Karma: 1603 [add] (https://forum.arduino.cc/index.php?action=karma;sa=applaud;uid=157795;aaa1357219bf=cb0a4f448aa083e28b77d649682a8d9e)

May all of your blinks be without delay()

Using millis() for timing. A beginners guide (https://forum.arduino.cc/index.php?topic=503368.msg3433063#msg3433063)

Oct 02, 2017, 08:16 am (https://forum.arduino.cc/index.php?topic=503368.msg3433063#msg3433063)

It is not usually long before new Arduino users discover that although the delay() function is easy to use it has side effects, the main one of which is that its stops all activity on the Arduino until the delay is finished (not quite true, I know, but that is usually how the problem presents itself). When this occurs the new user is usually directed to the BlinkWithoutDelay example sketch
(BWoD) and/or the excellent thread entitled Several things at the same time. Several things at the same time (http://forum.arduino.cc/index.php?topic=223286.0)

Quite often this only serves to confuse the new user because they don't just want to blink an LED or can't get their head around doing more than one thing at (apparently) the same time. What they need is to understand the BWod principle before they can see how to apply it to their own situation.

The programs presented here overlap with those in that thread but I have put my own spin on using millis() and described the programs in my own way. Between the two you should have a clearer understanding of how to use millis() for non blocking timing. In this thread I will try to explain the principles of using millis() for timing and apply it to some common areas where questions arise. The principle is easy to describe but there are some gotchas along the way that you need to look out for.

To use millis() for timing you need to record the time at which an action took place to start the timing period and then to check at frequent intervals whether the required period has elapsed. If so, you presumably want to do something, otherwise why would you be timing ?

If the period has not yet elapsed then you can go on your way and possibly do other

things until the next check.

In the sample programs below there are some assumptions made as follows :

1. You know how to set the pinMode() of an input and output
2. Pins A1, A2 and A3 are used as digital inputs but any pin other than 0 and 1 can be used.
3. Input pins are defined as INPUT_PULLUP in pinMode() and that closing the associated switch takes the pin LOW
4. Pins 10, 11, 12 and 13 each have LEDs attached via a current limiting resistor to 5V, so taking the pin LOW turns on the LED
5. Pins 10 and 11 are capable of PWM output when required.
6. Variables used will be declared as globals for ease of use but purists may want to declare some of them locally
7. The programs in this thread have been written and tested on a Uno but will run on most/all Arduino boards

Let's get started

In order to use millis() for timing the program is going to need to know its current value, perhaps more than once in each time through loop(). Whilst it is possible to read the value each time it is required it is more convenient to read it once in each pass so that within the program its value can be used as many times as needed and that it is consistent. It is also convenient to do this at the start of loop() and you do it like this

Code: [Select]

```
currentMillis = millis();
```

Simple enough, but this line of code embodies a number of important ideas :

1. The variable must previously have been declared.
2. It is an unsigned long because that is what millis() returns.
3. It is important that it is unsigned (more on this later)
4. It has a descriptive name. Feel free to use your own name but this is the one that I will be using.

We are going to need at least 2 other variables in order to determine whether the required period has elapsed. We know the current value of millis(), but when did the timing period start and how long is the period ?

At the start of the program declare 3 global variables, as follows
Code: [Select]

```
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 1000;  //the value is a number of milliseconds, ie 1 second
```

Let's use those variables in a program that blinks an LED, not quite the same as the example in the IDE, but one that shows the use of millis() nevertheless. Remember the

principle, as laid out above "you need to record the time at which an action took place to start the timing period and then to check at frequent intervals whether the required period has elapsed." If so, you presumably want to do something, otherwise why would you be timing ?"

If the period has not yet elapsed then you can go on your way and possibly do other things until the next check. More of this later.

The program :

Code: [Select]

```
unsigned long startMillis;  //some global variables available anywhere in t
unsigned long currentMillis;
const unsigned long period = 1000;  //the value is a number of milliseconds
const byte ledPin = 13;    //using the built in LED

void setup()
{
  Serial.begin(115200);  //start Serial in case we need to print debugging
  pinMode(ledPin, OUTPUT);
```

Follow the code through and see how the current value of millis() is compared with the start time to determine whether the period has expired. If you don't understand how the state of the LED is changed don't worry but for information it reads the current state of the LED pin (HIGH or LOW) and writes the opposite state (HIGH or LOW) to it.

Probably the most important thing is to remember to save the start time when the LED state is changed, ie the start of the next timing period.

Copy the program into the IDE, upload it and watch in amazement as the LED blinks. Do not resist the temptation to change the blink period.

Now let's tackle what might be an elephant in the room. The program will sit there quite happily comparing the current time with the start time and turning the LED on and off if it is time to do so. After some time, actually just over 49 days, the value returned by millis() will get so large that it will not fit in an unsigned long variable and it will roll over to zero and start incrementing again.

I don't suppose that you will leave this program running for 49 days to see what happens (please feel free to do so), but suppose that you had used the principle for something more critical that simply must not do something silly after 49 and a bit days. Fear not. Using unsigned variables and subtraction for the elapsed period the comparison will work even if/when millis() rolls over to zero when the program is running. This is not the place to have a diversion into the reasons why this works but trust me, it does.

OK, that is blinking a single LED with a symmetrical on/off period done and dusted using the principle of testing the elapsed time since an event. What next ? Well, we could blink more than one LED with a different period, we could have different on/off periods and

even do a combination of both. The best way to do this is to use arrays of values but it would mean introducing the principles of using arrays and if you are reading this I suspect that it will overload you and be too large a diversion from the subject in hand. You could change the on/off periods by simply changing the value of the period variable when you change the state of the LED

So what next ?
How about applying the principle to changing the brightness of an LED instead of turning it on and off ?

In the next installment that is what we will do and after that we will make the program appear to do two things at once using the magic of millis()

Please do not send me PMs asking for help. Post in the forum then everyone will benefit from seeing the questions and answers.

UKHeliBob (https://forum.arduino.cc/index.php?action=profile;u=157795)

(https://forum.arduino.cc/index.php?action=profile;u=157795)

Brattain Member

Posts: 23,324

Karma: 1603 [add] (https://forum.arduino.cc/index.php?action=karma;sa=applaud;uid=157795;aaa1357219bf=cb0a4f448aa083e28b77d649682a8d9e)

May all of your blinks be without delay()

Re: Using millis() for timing. A beginners guide (https://forum.arduino.cc/index.php?topic=503368.msg3433065#msg3433065)
Oct 02, 2017, 08:16 am (https://forum.arduino.cc/index.php?topic=503368.msg3433065#msg3433065) #1

Part 2

We have seen how to turn an LED on and off at intervals which is fascinating for maybe a minute or two at most but how about using the same principle to change the brightness of an LED ?

I will use an LED on pin 10 in this example as that pin supports PWM output. The basis of the program will be the same as before except that when the period ends the brightness of the LED will be changed. The period will be much less than 1 second, unless you want to wait a long while, but otherwise the same principles will be used.

So here it is. Note how similar it is to the previous sketch.

Code: [Select]

```
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 10;  //the value is a number of milliseconds b
const byte ledPin = 10;    //using an LED on a PWM pin.
byte brightness = 0;       //initial brightness
byte increment = 1;        //amount to change the brightness at the end of

void setup()
{
```

In this program at the end of the timing period the brightness of the LED is changed rather than turning it on or off.

So what is all the fuss about ? That's the point, there is no fuss. You can use the BWoD principle for long or short periods and do anything you like at the end of the period (within reason), but to be honest you could have done of this using delay(). But suppose you wanted to do something else at the same time ? delay() could interfere with whatever else you want to do but using millis() and the BWoD principle you can execute code in loop() in between checking whether the period has expired and, of course, the other code can use millis() for timing too.

Let's look at a program that does what each of the previous two do but apparently at the same time. They won't actually be done at the same time, but because loop() repeats thousands of times per second it will appear that the program is doing two things at once.

Here it is

Code: [Select]

```
unsigned long blinkStartMillis;
unsigned long fadeStartMillis;
unsigned long currentMillis;
const unsigned long blinkPeriod = 1000;  //blink period
const unsigned long fadePeriod = 10;  //fade period
const byte blinkLedPin = 13;    //this LED will blink
const byte fadeLedPin = 10;    //this LED will fade
byte brightness = 0;  //initial brightness of LED
byte increment = 1;  //amount to change PWM value at each change
```

At first sight it my seem daunting but you have seen all of it before in the two previous programs. Changes have been made as follows :

1. Separate variables are needed for the start times of the two different periods
2. Separate variables are needed for the periods
3. The new variable have been given meaningful names which relate to the code they belong to
4. The currentMillis variable is shared between the two functions as it the same for both of them
5. The code for each set of timing has been moved into functions with descriptive names to make maintenance easier but it could equally all have been in loop().

Note that it would not have been possible to combine the two programs if delay() had been used. For one thing the one second delay in the blink program would have prevented the fade working.

OK, we can blink and fade LEDs at the "same" time but what if we want to do something completely different such as reading an input for a period of a few seconds and count how many times the button is pressed in that time ? You cannot do that using delay() because you cannot delay and read an input at the same time, but as the blink and fade program illustrates, using millis() for timing you can do two things so frequently that they appear to happen at the same time.

)FILE%3ACORE+PROFILE%3APUBLIC+PROFILE%3ACONTACT+OFFLINE&RESPONSE_TYPE=CODE)

In the next part that is what we will do.

Please do not send me PMs asking for help. Post in the forum then everyone will benefit from seeing the questions and answers.

UKHeliBob (https://forum.arduino.cc/index.php?action=profile;u=157795)

(https://forum.arduino.cc/index.php?action=profile;u=157795)

Brattain Member

Posts: 23,324

Karma: 1603 [add] (https://forum.arduino.cc/index.php?action=karma;sa=applaud;uid=157795;aaa1357219bf=cb0a4f448aa083e28b77d649682a8d9e)

May all of your blinks be without delay()

Re: Using millis() for timing. A beginners guide (https://forum.arduino.cc/index.php?topic=503368.msg3433066#msg3433066)

Oct 02, 2017, 08:17 am (https://forum.arduino.cc/index.php?topic=503368.msg3433066#msg3433066) #2

Part 3

Here we go then with counting button presses in a 5 second period

The program

Code: [Select]

```
unsigned long startMillis;
unsigned long currentMillis;
const unsigned long period = 5000;  //period during which button input  is
const byte buttonPin1 = A1;    //button on pin A1
byte currentButtonState;
byte previousButtonState;
int count = 0;
boolean counting)
```

So what is going on in this program ?

It starts as do the previous ones by declaring global variables. There are some new ones here because we want to detect when the button becomes pressed rather than when it is pressed. We are going to print a message on the screen when the time is up so need a way of preventing the message from being repeated, hence the boolean variable.

As usual we start by getting the current value of millis() then establishing whether the period has elapsed by the usual subtraction and compare calculation. But this time we need to read the button state if the period has not elapsed so the comparison with the period is reversed from greater than used in the previous examples, to less than. If the period has not elapsed we read the input. In order to determine whether the button has become pressed we save the previous state, read it again next time through loop() and compare the two.

If it has changed since last read and is now LOW we know that the button has become pressed. If so we increment the count and print it. If not we do nothing but go round loop() again. Once the period has elapsed the subtract and compare calculation will

)FILE%3ACORE+PROFILE%3APUBLIC+PROFILE%3ACONTACT+OFFLINE&RESPONSE_TYPE=CODE)

return false and we need to print a message. This is done in the else clause and the final message is printed only if the boolean variable is true. Once the message has been printed the boolean is set to false to prevent the final message being printed again even though the period has ended.

Upload it to your Arduino and try is. Does it do what it should ?
It is very likely that the count will actually be higher than the number of button presses because the switch contacts bounce and each bounce is counted. You may be lucky and have perfect switches that do not bounce but most of us will not be that lucky.

We need to increment the count only when the switch contacts remain closed for a few milliseconds so that we know they have stopped bouncing. Our friend millis() allows us to do that without blocking the program and hanging around waiting for the bouncing to stop.

The next chapter will show how millis() timing can be used to detect when a switch has stopped bouncing.

Please do not send me PMs asking for help. Post in the forum then everyone will benefit from seeing the questions and answers.

UKHeliBob (https://forum.arduino.cc/index.php?action=profile;u=157795)

(https://forum.arduino.cc/index.php?action=profile;u=157795)

Brattain Member

Posts: 23,324

Karma: 1603 [add] (https://forum.arduino.cc/index.php?action=karma;sa=applaud;uid=157795;aaa1357219bf=cb0a4f448aa083e28b77d649682a8d9e)

May all of your blinks be

Re: Using millis() for timing. A beginners guide (https://forum.arduino.cc/index.php?topic=503368.msg3433067#msg3433067)

Oct 02, 2017, 08:17 am (https://forum.arduino.cc/index.php?topic=503368.msg3433067#msg3433067) #3

Here is a program that will count button pressws in a 5 second period but in this version the button has to remain pressed for 20 milliseconds before the button press is counted. Can you guess what we use to time the 20 milliseconds ?

Code: [Select]

```
unsigned long periodStartMillis;
unsigned long currentMillis;
const unsigned long period = 5000;  //period during which button input is v
const byte buttonPin1 = A1;    //button on pin A1
byte currentButtonState;
byte previousButtonState;
int count = 0;
boolean printFinalMessage = true;
unsigned long debounceStartMillis;
```

The majority of this program is taken from the previous one but a new element has been introduced. When the change of button state is detected the value of millis() at that time is saved and a boolean variable is set to true indicating that debouncing is in progress.

Then each time through loop() the elapsed time since debouncing started is checked and if it expires and debouncing is in progress and the button is currently pressed we know that it has been pressed for at least the debouncing period, so we count the button press

without delay()

by incrementing the count and set the boolean to false to stop it happening again the next time through loop() even though the debouncing period has expired. It will be set to true again the next time a button press is detected to start the debouncing process again.

All of the button press detection and debouncing code above is wrapped in the test to see whether the 5 second timing period is still happening. As before we could not have used delay() for this as the 5 second delay() would have held the program up. We could conceivably used delay() for the debouncing as in this program doing nothing for 20 milliseconds would not make much difference, but it is all too easy to use delay() in circumstances where it is not appropriate, so it is better to consider using millis() from the very start. This is particularly so when programs are being tested in isolation to iron out bugs before combining them. This is good practice, but just because they work in isolation does not mean that they will work when combined, particularly when precise timing is involved.

Please do not send me PMs asking for help. Post in the forum then everyone will benefit from seeing the questions and answers.

Go Up Pages: [1]

PRINT (HTTPS://FORUM.ARDUINO.CC/INDEX.PHP?ACTION=PRINTPAGE;TOPIC=503368.0)

Jump to:
=> Programming Questions Go