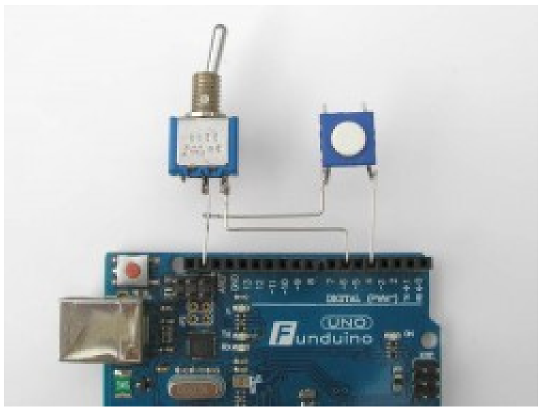
Simple Arduino library for debouncing switches and buttons

Contents[Hide]

1. Switch library


Arduino library for debouncing switches and buttons

Switches and push buttons need debouncing. It is straightforward to do this with software, using a library. The advantages of the Switch library are:

- External pull-up resistors are not required.
- Supports also **long press** and **double clicks**.
- Minimal used program space: 506 bytes.

2. Connecting switches to the Arduino

There is nothing needed beyond the switches, just connect the switches between the ground and a digital pin:

2.1. Switch between GND and digital pin

This is technically speaking the best solution and is taken by the library as default. An external pull-up resistor is not needed but allowed.

2.2. Switch between 5V and digital pin

For switches connected to the Arduino power supply, the settings are: polarity = HIGH and pinmode = INPUT, which disables the internal pull-up resistor. Note that we need external pull-down resistors of about 10k here.

3. Using the Switch library

All switched have to be polled individually to update the status. Each switch-status has its own get function:

- pushed()
  Use only for push buttons. It returns "true" if a button was pushed after the `poll()` instruction was executed.
- released()
  It returns "true" if a push button was released, this will however rarely be used.
- on()
  Use only for toggle switches. It returns "true" as long as the switch is in the "on" position. The polarity of the switch in the "on" position has to be filled in correctly. There is no `off()` function, this is simply `!on()`.
- longPress()
  It returns "true" if a push button is pressed longer than 500ms.
- doubleClick()
  It returns "true" if a push button is double clicked within 250ms.

```
#if ARDUINO >= 100
  #include "Arduino.h"
#else
  #include "WProgram.h"
#endif

#include <Streaming.h>
#include "Switch.h"

const byte toggleSwitchpin = 3; // (right button)
const byte buttonGNDpin = 4; // (left button)
const byte ButtonVCCpin = 6;
const byte Button10mspin = 8;
int i;

Switch buttonGND = Switch(buttonGNDpin); // button to GND, use internal 20K pullup resistor
Switch toggleSwitch = Switch(toggleSwitchpin);
Switch buttonVCC = Switch(ButtonVCCpin, INPUT, HIGH); // button to VCC, 10k pull-down resistor, no intern
al pull-up resistor, HIGH polarity
Switch button10ms = Switch(Button10mspin, INPUT_PULLUP, LOW, 1); // debounceTime 1ms
```

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

void loop()
{ buttonGND.poll();
  if(buttonGND.switched()) Serial << "switched ";
  if(buttonGND.pushed()) Serial << "pushed " << ++i << " ";
  if(buttonGND.released()) Serial << "released\n";

  if(toggleSwitch.poll()) Serial << toggleSwitch.on() << endl;
  if(toggleSwitch.longPress()) Serial << "longPress1 ";
  if(toggleSwitch.longPress()) Serial << "longPress2\n";
  if(toggleSwitch.doubleClick()) Serial << "doubleClick1 ";
  if(toggleSwitch.doubleClick()) Serial << "doubleClick2\n";
}
```

4. Notes

- The poll instruction must be called at least 50 times per second.
- The button states are saved till the next poll. Then all previous button states will be cleared, so the buttons must be read every poll interval.
- Reading a button several times within a poll interval gives the same value, so reading doesn't clear the button state.

5. How to use successive button events

With the Switch library, you can use all button events at the same time with the same button. For example pushed(), released(), doubleClick() and longPress(). To see how several button events can be used together, run Windows Explorer: a single click selects a file and a double click opens it.

5.1. Button pushed event followed by a longPress event

A long-press generates first a pushed event and after 500ms the longPress event. This is not a shortcoming but a logical consequence. We can, of course, not always wait 500ms to see if a push might be a long push.

5.2. Button pushed event followed by a doubleClick event

The same happens with doubleClick, which also generates two pushed() events. When doubleClick is used, ignore the second pushed() result or don't call pushed(). When doubleClick is not needed, simply don't call doubleClick().

6. Switch library software

For the latest software version you can ask me.
The debounce time is **50ms**, this is a safe value, it doesn't hurt the reaction time, and will handle even bad switches. Copy the code below into a Switch.cpp and Switch.h file and place the two files into an Arduino library folder like this: \libraries\Switch.

c

6.1. Switch.cpp

The debounce timing diagram is included in the Switch.cpp file.

```
/*
Switch
Copyright (C) 2012  Albert van Dalen http://www.
This program is free software: you can redistrib
as published by the Free Software Foundation, ei
This program is distributed in the hope that it
of MERCHANTABILITY or FITNESS FOR A PARTICULAR P

Version 20-4-2013
_debounceDelay=50
Version 22-5-2013
Added longPress, doubleClick


                        _______________________
                       |                       |
 input                 |                       |
                  _____|                       |

 poll                  ^
 switchedTime          ^
 debounceDelay                    <-------------
 switched
 newlevel
                              __________________
 level



                            ____________________
                           |
 input                     |
                   ________|

 longPressDelay             <----------->

 doubleClickDelay           <-------------------
                                          ______
 longPressLatch    ______________________|
                                          _
 _longPress        ______________________|¯|____

_doubleClick       _____________________________

*/

#if ARDUINO >= 100
  #include "Arduino.h"
#else
  #include "WProgram.h"
#endif
#include "Switch.h"

// level(0)
```

```
Switch::Switch(byte _pin, byte PinMode, bool pol
pin(_pin), polarity(polarity), debounceDelay(deb
{ pinMode(pin, PinMode);
  _switchedTime = millis();
  level = digitalRead(pin);
}

bool Switch::poll()
{ _longPress = _doubleClick = false;
  bool newlevel = digitalRead(pin);

  if(!longPressLatch)
  { _longPress = on() && ((long)(millis() - push
    longPressLatch = _longPress; // will be rese
  }

  if((newlevel != level) & (millis() - _switched
  { _switchedTime = millis();
    level = newlevel;
    _switched = 1;
    longPressLatch = false;

    if(pushed())
    { _doubleClick = (long)(millis() - pushedTim
      pushedTime = millis();
    }
    return _switched;
  }
  return _switched = 0;
}

bool Switch::switched()
{ return _switched;
}

bool Switch::on()
{ return !(level^polarity);
}

bool Switch::pushed()
{ return _switched && !(level^polarity);
}

bool Switch::released()
{ return _switched && (level^polarity);
}

bool Switch::longPress()
{ return _longPress;
}

bool Switch::doubleClick()
{ return _doubleClick;
}
```

c

Switch.h

```
/*
Switch
Copyright (C) 2012  Albert van Dalen http://www.a
This program is free software: you can redistribu
as published by the Free Software Foundation, eit
This program is distributed in the hope that it w
of MERCHANTABILITY or FITNESS FOR A PARTICULAR PU
*/

#ifndef SWITCH_H
#define SWITCH_H

class Switch
{
public:
  Switch(byte _pin, byte PinMode=INPUT_PULLUP, bo
  bool poll(); // Returns 1 if switched
  bool switched(); // will be refreshed by poll()
  bool on();
  bool pushed(); // will be refreshed by poll()
  bool released(); // will be refreshed by poll()
  bool longPress(); // will be refreshed by poll(
  bool doubleClick(); // will be refreshed by pol

  unsigned long _switchedTime, pushedTime;

protected:
  const byte pin;
  const int debounceDelay, longPressDelay, double
  const bool polarity;
  bool level, _switched, _longPress, longPressLat
};

#endif
```

7. Using an interrupt service routine for polling the buttons

Polling buttons has a high priority, slow functions such as Serial.print() may disrupt the timing. See here how to use an ISR for polling the buttons:

```
#include <Arduino.h>
#include "Switch.h"
#include <FrequencyTimer2.h>

Switch speedUpBtn(1);
Switch speedDownBtn(2);
Switch buttonLeft(3);
Switch buttonRight(4);
```

```

void setup(void)
{ Serial.begin(9600);
  FrequencyTimer2::setPeriod(1000);
  FrequencyTimer2::setOnOverflow(timer2ISR);
}

void loop(void)
{ printAll(); // run slow functions in loop()
}

void timer2ISR()
{ pollAll(); // polling buttons has priority
  buttonActions();
}
```

8. Other switch debouncing libraries

For the Arduino, there are several libraries available:

- Button library from Carlyn Maw
  Minimal used program space: 740 bytes.
  It has an extensive interface.
- phi_interfaces library from John Liu
  Minimal used program space: 976 bytes.
  This is a very sophisticated library for handling a whole bunch of input devices, but for just a few buttons, the class is rather big.
- Bounce library
  Minimal used program space: 550 bytes.
  This class is made unnecessarily complex by a rebounce function which doesn't belong here.
- Button library from Alexander Brevig
  This library had previously no debounce but it has been improved and now it has also debounce.

So actually, there's plenty of choice, but I wanted a very small and simple class with a straightforward interface. So I build the Switch class.

9. Links

- A Guide to Debouncing

Disclaimer
Contact

Do you have any comments? Please let me know.