Arduino Software Reset

On our quest to find an easy way to program an arduino wirelessly over bluetooth, we investigated the possibility to reset the arduino through software only, without the need to add or change hardware to reset the arduino. Here I will describe the necessary changes to do so:

1. Reset Function

To reset the arduino from the software, we force a software reset using the AVR watchdog. A detailed description of the watchdog and using it in an arduino project can be found here: http://tushev.org/articles/electronics/48-arduino-and-watchdog-timer. For our case, we included this in the following function:

```
#include <avr/wdt.h>

void reset()
{
  Serial.println("Device will reset in 1 second ...");
  wdt_disable();
  wdt_enable(WDTO_1S);
  while (1) {}
}
```

If you want to change the delay, use another constant instead of WDTO_1S.

However, this alone is not sufficient as using the watchdog will cause problems with most bootloaders since they don't disable the WDT which could cause the bootloader to be resetted during program upload. Other, newer, bootloaders on the other hand do disable the watchdog, but they only enable program upload if the reset was triggered by an external event such as pressing the reset button. So in order to be able to upload a program after resetting the arduino with the watchdog we need to adapt the bootloader and burn it on the arduino.

2. Bootloader Changes

The default bootloader installed on most arduinos (except the UNO and the Mini) is the ATmegaBOOT_168 which can be found in the arduino distribution in '$ARDUINO_PATH/hardware/arduino/bootloaders/atmega', where $ARDUINO_PATH is the location that you installed the arduino IDE.

The first thing to do before we can use the bootloader is to make it compile. A detailed description can be found here http://n0m1.com/2012/04/01/how-to-compiling-the-arduino-bootloader/.

Next we need to enable the watchdog modifications so that the bootloader disables the watchdog when it starts. To do that, open the Makefile and search for the target of the board you are using. E.g. for an Arduino Mini Pro 5V 16 MHz this would be the target atmega328. Then add '-DWATCHDOG_MODS' as an additional flag.

E.g.:
atmega328: CFLAGS += '-DMAX_TIME_COUNT=F_CPU>>4' '-DNUM_LED_FLASHES=3' '-DBAUD_RATE=57600' '-DWATCHDOG_MODS'

NB: You can also define how often the LED should flash if the board is reset to show that the bootloader is waiting for a program upload (default is 1, I changed it to 3)

Last but not least we need to let the bootloader wait for a program upload once the watchdog resets the arduino (and not only if the reset button is pressed)

To do so open ATmegaBOOT_168.c and change the line

```
if (! (ch &  _BV(EXTRF))) // if its a not an external reset...
```

to

```
if (! (ch &  (_BV(EXTRF) | _BV(WDRF)))) // if its a not an external reset...
```

WDRF is the flag being set by the watchdog to indicate that the arduino was reset by the watchdog.

Save your changes, then open a terminal and navigate to the folder atmega bootloader folder. Here call make together with the target that you modified (the target for your board). In my example:

```
make atmega328
```

this will create one *.hex and one *.lst file. (in my case ATmegaBOOT_168_atmega328.hex and .lst)

3. Burn Bootloader on Arduino

Now the bootloader is ready to be burned on the arduino. To do so you need an AVR ISP. Since I didn't have one handy, but I had an UNO instead, I made my UNO into an ISP. The tutorial to set up the UNO as isp can be found here http://arduino.cc/en/Tutorial/ArduinoISP. Since my target was an arduino Mini Pro 5V I used this source https://forum.sparkfun.com/viewtopic.php?f=32&t=27960 instead to connect the UNO with the mini pro. Because my mini pro was 5V I connected the VCC Pin of the Pro Mini with the 5V Pin of to UNO and selected the board Arduino Pro Mini 5V w/ ATmega328. Otherwise the procedure is exactly the same. (Note in the newest version of Arduino you have to select the Programmer (Arduino as ISP) before pressing Burn Bootloader under Tools > Programmer > Arduino as ISP)

If everything went alright, the IDE should notify you once the bootloader was burned successfully.

Now your arduino is ready to go and will listen for a program upload after a software reset. In order to combine that with the wireless programming, follow my tutorial at https://github.com/eggerdo/arduino_blue.