OPENDROP V2.0 FIRMWARE BACKUP & RESTORE WITH AVRDUDE

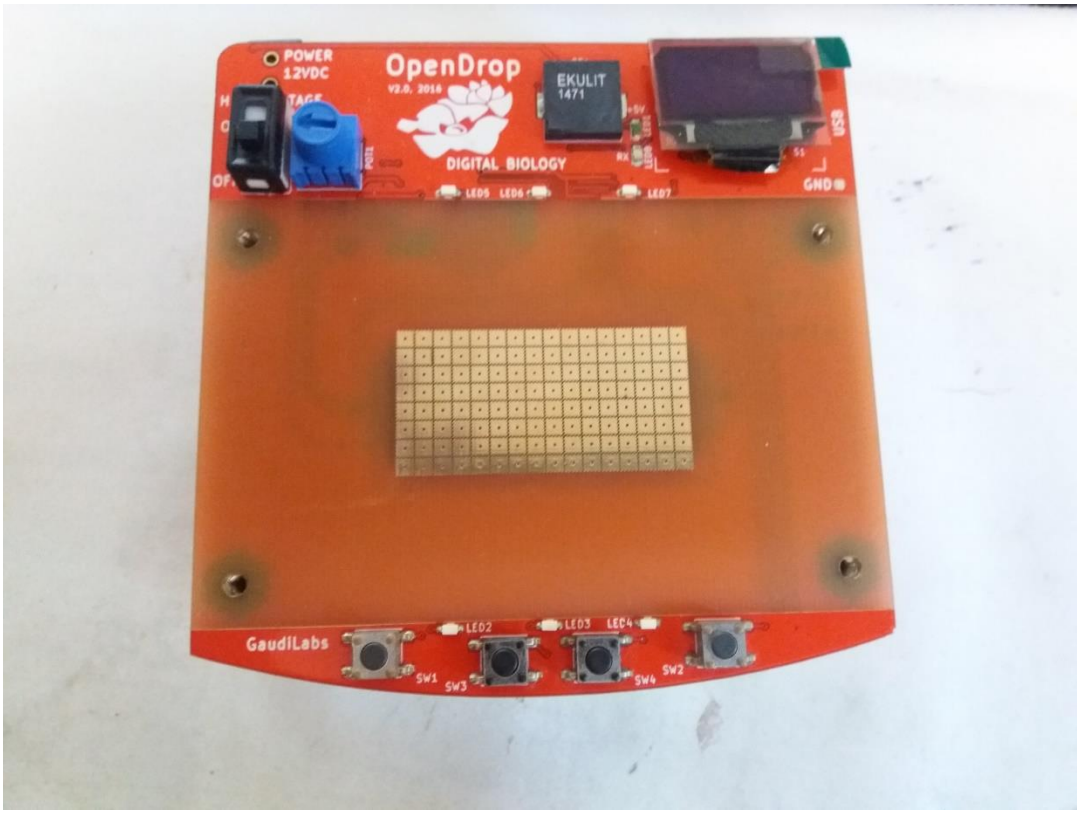
SHORT OVERVIEW

This document & blog post is for people who:

- want to get a basic understanding for avrdude usage with a Sparkfun Pro Micro (or compatible) board
- (ATmega32U4 5V, 16 MHz)
- get firmware files for the OpenDrop v2.0 (four push buttons) - the firmware is NOT compatible with newer OpenDrop versions (v2.1 with joystick) - please use the GitHub download to compile & upload the Firmware yourself.
- aren't afraid to get their hands dirty with command line stuff under Linux (and will bear the consequences - we are not liable for you breaking devices by following this guide! Use caution and check the commands - the difference between reading (r) and writing (w) is literally only one letter)

I had to repair a broken Arduino in one of our OpenDrop V2.0 units (with four push buttons). In the official GitHub repository (-> https://github.com/GaudiLabs/OpenDrop), the download will give you code for the OpenDrop v2.1 (with a joystick). Therefore I deemed the easiest solution would be to backup the firmware from another OpenDrop v2.0 device, and copy it to the new Arduino.

I used avrdude on a Raspberry Pi unit to back up the firmware, and write it onto the new Arduino. Backup requires several steps, and putting the Arduino into the proper state (also required for reading!).



PITFALLS I'VE ENCOUNTERED

- check that your USB cable includes wiring for data - the first cable I've tried will only power the Arduino, but it will not show up in the USB devices.
- Copy & pasting command line codes from most blogs might render some characters in a wrong way - consider using the PDF document I provide of this post, to ensure that the code is entered correctly

SET UP & BASIC UNDERSTANDING

The OpenDrop v2.0 is driven by an Arduino device, which is compatible with the Sparkfun Pro Micro board. (For instance Deekrobot Pro Micro)

- ATmega32U4 (5V, 16 MHz)
- integrated microUSB port for power & programming

avrdude is used to read and write firmware files & settings from / to the Arduino.

I use a Raspberry Pi, since we are a Raspberry Pi approved reseller & I feel comfortable with using the Linux command line over Windows.

INSTALLATION OF AVRDUDE:

```
sudo apt-get install avrdude
```

LOCATING THE PORT TO USE WITH AVRDUDE

Unplug the mains power adapter from the OpenDrop, if it had been plugged in.

Plug in the Arduino microUSB cable into the Arduino, and into the Raspberry Pi. The device will beep.

Ensure that the Arduino shows up as a USB device:

```
/home/pi# lsusb
```

Bus 001 Device 005: ID 2341:8037 Arduino SA
Bus 001 Device 004: ID 0424:7800 Standard Microsystems Corp.
Bus 001 Device 003: ID 0424:2514 Standard Microsystems Corp. USB 2.0 Hub
Bus 001 Device 002: ID 0424:2514 Standard Microsystems Corp. USB 2.0 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

If the Arduino SA entry does NOT show up, double-check your USB cable: some cables only transmit power, not data. This, by the way, was also the error why I could not use the Arduino IDE on Windows with this device - I was trying a wrong USB cable first!

Find the port, which the Sparkfun Pro Micro uses:

```
ls /dev/tty*
```

/dev/tty /dev/tty12 /dev/tty17 /dev/tty21 /dev/tty26 /dev/tty30 /dev/tty35 /dev/tty4 /dev/tty44
/dev/tty49 /dev/tty53 /dev/tty58 /dev/tty62 ***/dev/ttyACM0***
/dev/tty0 /dev/tty13 /dev/tty18 /dev/tty22 /dev/tty27 /dev/tty31 /dev/tty36 /dev/tty40 /dev/tty45
/dev/tty5 /dev/tty54 /dev/tty59 /dev/tty63 /dev/ttyAMA0
/dev/tty1 /dev/tty14 /dev/tty19 /dev/tty23 /dev/tty28 /dev/tty32 /dev/tty37 /dev/tty41 /dev/tty46
/dev/tty50 /dev/tty55 /dev/tty6 /dev/tty7 /dev/ttyprintk

/dev/tty10 /dev/tty15 /dev/tty2 /dev/tty24 /dev/tty29 /dev/tty33 /dev/tty38 /dev/tty42 /dev/tty47 /dev/tty51 /dev/tty56 /dev/tty60 /dev/tty8
/dev/tty11 /dev/tty16 /dev/tty20 /dev/tty25 /dev/tty3 /dev/tty34 /dev/tty39 /dev/tty43 /dev/tty48 /dev/tty52 /dev/tty57 /dev/tty61 /dev/tty9

You are looking for something like /dev/ttyACM0, or /dev/ttyUSB0

In my case it was /dev/ttyACM0

This port will only be present, when the device is plugged in. To ensure that you have the correct port, you can try listing with the OpenDrop being connected, and the OpenDrop being disconnected (microUSB -> USB on the Pi). Compare the outputs. I assume that it will be /dev/ttyACM0 for you as well.

CREATE DIRECTORY FOR BACKUP

We will become the superuser, create the directory for the backup files, and change into it:

```
sudo su

mkdir /home/pi/opendrop2.0_bak

cd /home/pi/opendrop2.0_bak
```

USING AVRDUDE

THE RESET BUTTON (IMPORTANT, DO NOT SKIP THIS PART)

For avrdude to work, the SparkFun Pro Micro must be in the right system state. In order to put the SparkFun Pro Micro into the right system state, you must press the reset button (labeled RST) quickly two times after each other. The device will beep, and the LED will start to blink.

You must execute the avrdude command immediately after doing this, as the device will go back into the normal state after a short timeout.

For every command which I executed, I pressed the RST button two times after each other. Probably the commands could be combined into a shell script; since I like to use caution as not to brick the devices by using the wrong command line, I stepped through the process manually.

If you do not use the RST button as I have outlined, you will get errors from avrdude like:

Connecting to programmer: .avrdude: butterfly_recv(): programmer is not responding

FIRMARE BACKUP: READING - OVERVIEW

We are going to back up

- flash
- eeprom
- hfuse
- lfuse
- efuse

Ref: https://www.hackster.io/rayburne/avr-firmware-duplicator (Careful, the avrdude settings in here are not correct for the OpenDrop / Sparkfun Pro Micro).

FIRMWARE BACKUP: THE COMMANDS

If not done yet, change into the directory you have prepared for the Firmware backup files, and become the super user (root). I am not sure if avrdude will work out of the box without root permissions - try it if you want.

Ensure that you use the right port (most probably /dev/ttyACM0). **Ensure that you have the letter "r" after the reading target**. r as in read; a letter w would use your destination file as source, to overwrite your Arduino Flash - without security prompt!

The letter r after the backup file name specifies the output type, it is important to get this right for the later restoration operation. If you are curious, read the manpage here: https://www.nongnu.org/avrdude/user-manual/avrdude_4.html

-p specifies the platform

-c the programmer

-P the port

-U specifies the operation (source/target:mode:target/source:file_mode)

Here, then, is the first command to use, and it's output:

avrdude -p atmega32u4 -c avr109 -P /dev/ttyACM0 -U flash:r:flash_backup_file_od2.0.hex:r

Connecting to programmer: .
Found programmer: Id = "CATERIN"; type = S
* Software Version = 1.0; No Hardware Version given.*
Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
* Device code: 0x44*

avrdude: AVR device initialized and ready to accept instructions

Reading | ## | 100% 0.00s

avrdude: Device signature = 0x1e9587 (probably m32u4)
avrdude: reading flash memory:

Reading | ## | 100% 0.64s

avrdude: writing output file "flash_backup_file_od2.0.hex"

avrdude: safemode: Fuses OK (E:CB, H:D8, L:FF)

avrdude done. Thank you.

Here's the second command. By the way, you can change the filenames (in this case the eeprom_backup_file_od2.0.hex bit):

avrdude -p atmega32u4 -c avr109 -P /dev/ttyACM0 -U eeprom:r:eeprom_backup_file_od2.0.hex:r

Connecting to programmer: .
Found programmer: Id = "CATERIN"; type = S
* Software Version = 1.0; No Hardware Version given.*

Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
Device code: 0x44

avrdude: AVR device initialized and ready to accept instructions

Reading | ## | 100% 0.00s

avrdude: Device signature = 0x1e9587 (probably m32u4)
avrdude: reading eeprom memory:

Reading | ## | 100% 1.76s

avrdude: writing output file "eeprom_backup_file_od2.0.hex"

avrdude: safemode: Fuses OK (E:CB, H:D8, L:FF)

avrdude done. Thank you.

Then we will backup three different kinds of fuses. Here's the first:

avrdude -p atmega32u4 -c avr109 -P /dev/ttyACM0 -U hfuse:r:hfuse_backup_file_od2.0.hex:r

Connecting to programmer: .
Found programmer: Id = "CATERIN"; type = S
Software Version = 1.0; No Hardware Version given.
Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
Device code: 0x44

avrdude: AVR device initialized and ready to accept instructions

Reading | ## | 100% 0.00s

avrdude: Device signature = 0x1e9587 (probably m32u4)
avrdude: reading hfuse memory:

Reading | ## | 100% 0.00s

avrdude: writing output file "hfuse_backup_file_od2.0.hex"

avrdude: safemode: Fuses OK (E:CB, H:D8, L:FF)

avrdude done. Thank you.

now the lfuse:

avrdude -p atmega32u4 -c avr109 -P /dev/ttyACM0 -U lfuse:r:lfuse_backup_file_od2.0.hex:r

Connecting to programmer: .
Found programmer: Id = "CATERIN"; type = S
Software Version = 1.0; No Hardware Version given.
Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
Device code: 0x44

avrdude: AVR device initialized and ready to accept instructions

Reading | ## | 100% 0.00s

avrdude: Device signature = 0x1e9587 (probably m32u4)
avrdude: reading lfuse memory:

Reading | ## | 100% 0.00s

avrdude: writing output file "lfuse_backup_file_od2.0.hex"

avrdude: safemode: Fuses OK (E:CB, H:D8, L:FF)

avrdude done. Thank you.

Now the last command, the efuse:

avrdude -p atmega32u4 -c avr109 -P /dev/ttyACM0 -U efuse:r:efuse_backup_file_od2.0.hex:r

Connecting to programmer: .
Found programmer: Id = "CATERIN"; type = S
Software Version = 1.0; No Hardware Version given.
Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
Device code: 0x44

avrdude: AVR device initialized and ready to accept instructions

Reading | ## | 100% 0.00s

avrdude: Device signature = 0x1e9587 (probably m32u4)
avrdude: reading efuse memory:

Reading | ## | 100% 0.00s

avrdude: writing output file "efuse_backup_file_od2.0.hex"

avrdude: safemode: Fuses OK (E:CB, H:D8, L:FF)

avrdude done. Thank you.

Remember that you need to press the RST button **twice** before each operation! It is **not** necessary to unplug the Arduino from the Raspberry Pi after executing the RST operation.

CREATING A ZIP FILE

you can create a ZIP file using the following command (replace the paths as appropriate):

sudo zip -r /home/pi/opendrop2.0_bak.zip /home/pi/opendrop2.0_bak

RESTORING THE FIRMWARE

If you need to restore the firmware, use the following commands. Again, don't forget to press the RST button twice before each command!

Run the commands from the correct directory as root. Ensure that you have the letter "w" for writing to the microcontroller (Arduino in this case).

Write the flash file:

avrdude -p atmega32u4 -c avr109 -P /dev/ttyACM0 -U flash:w:flash_backup_file_od2.0.hex

Connecting to programmer: .
Found programmer: Id = "CATERIN"; type = S
Software Version = 1.0; No Hardware Version given.
Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
Device code: 0x44

avrdude: AVR device initialized and ready to accept instructions

Reading | ## | 100% 0.00s

avrdude: Device signature = 0x1e9587 (probably m32u4)
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed
To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "flash_backup_file_od2.0.hex"
avrdude: writing flash (32604 bytes):

Writing | ## | 100% 2.98s

avrdude: 32604 bytes of flash written
avrdude: verifying flash memory against flash_backup_file_od2.0.hex:
avrdude: load data flash data from input file flash_backup_file_od2.0.hex:
avrdude: input file flash_backup_file_od2.0.hex contains 32604 bytes
avrdude: reading on-chip flash data:

Reading | ## | 100% 0.76s

avrdude: verifying ...
avrdude: verification error, first mismatch at byte 0x7000
0x55 != 0x5f
avrdude: verification error; content mismatch

avrdude: safemode: Fuses OK (E:CB, H:D8, L:FF)

avrdude done. Thank you.

Note: I tried to write several times, slightly varying the command. In each attempt I got the verification error; the firmware works fine, though - my superficial research showed it's related to some wiring of the microcontroller's periphery - it was inserted into the OpenDrop while I was doing the backup and restoration. I have decided that the results speak for themselves, and moved on to the restoration of the other parts. Your results might vary here. No guarantees given!

Restore the eeprom:

avrdude -p atmega32u4 -c avr109 -P /dev/ttyACM0 -U eeprom:w:eeprom_backup_file_od2.0.hex

Connecting to programmer: .
Found programmer: Id = "CATERIN"; type = S
Software Version = 1.0; No Hardware Version given.
Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
Device code: 0x44

avrdude: AVR device initialized and ready to accept instructions

Reading | ## | 100% 0.00s

avrdude: Device signature = 0x1e9587 (probably m32u4)
avrdude: reading input file "eeprom_backup_file_od2.0.hex"
avrdude: input file eeprom_backup_file_od2.0.hex auto detected as raw binary
avrdude: writing eeprom (1024 bytes):

Writing | ## | 100% 3.44s

avrdude: 1024 bytes of eeprom written
avrdude: verifying eeprom memory against eeprom_backup_file_od2.0.hex:
avrdude: load data eeprom data from input file eeprom_backup_file_od2.0.hex:
avrdude: input file eeprom_backup_file_od2.0.hex auto detected as raw binary
avrdude: input file eeprom_backup_file_od2.0.hex contains 1024 bytes
avrdude: reading on-chip eeprom data:

Reading | ## | 100% 1.76s

avrdude: verifying ...
avrdude: 1024 bytes of eeprom verified

avrdude: safemode: Fuses OK (E:CB, H:D8, L:FF)

avrdude done. Thank you.

Restore the fuses (Note: I've read something about some fuses being only resettable by a 12 V programmer. YMMV).

First hfuse:

avrdude -p atmega32u4 -c avr109 -P /dev/ttyACM0 -U hfuse:w:hfuse_backup_file_od2.0.hex

Connecting to programmer: .
Found programmer: Id = "CATERIN"; type = S
Software Version = 1.0; No Hardware Version given.
Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
Device code: 0x44

avrdude: AVR device initialized and ready to accept instructions

Reading | ## | 100% 0.00s

avrdude: Device signature = 0x1e9587 (probably m32u4)
avrdude: reading input file "hfuse_backup_file_od2.0.hex"
avrdude: input file hfuse_backup_file_od2.0.hex auto detected as raw binary
avrdude: writing hfuse (1 bytes):

*Writing | | 0% 0.00s ***failed;*
Writing | ## | 100% 0.00s

avrdude: 1 bytes of hfuse written
avrdude: verifying hfuse memory against hfuse_backup_file_od2.0.hex:
avrdude: load data hfuse data from input file hfuse_backup_file_od2.0.hex:
avrdude: input file hfuse_backup_file_od2.0.hex auto detected as raw binary
avrdude: input file hfuse_backup_file_od2.0.hex contains 1 bytes
avrdude: reading on-chip hfuse data:

Reading | ## | 100% 0.00s

avrdude: verifying ...
avrdude: 1 bytes of hfuse verified

avrdude: safemode: Fuses OK (E:CB, H:D8, L:FF)

avrdude done. Thank you.

now lfuse:

avrdude -p atmega32u4 -c avr109 -P /dev/ttyACM0 -U lfuse:w:lfuse_backup_file_od2.0.hex

Connecting to programmer: .
Found programmer: Id = "CATERIN"; type = S
Software Version = 1.0; No Hardware Version given.
Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
Device code: 0x44

avrdude: AVR device initialized and ready to accept instructions

Reading | ## | 100% 0.00s

avrdude: Device signature = 0x1e9587 (probably m32u4)
avrdude: reading input file "lfuse_backup_file_od2.0.hex"
avrdude: input file lfuse_backup_file_od2.0.hex auto detected as raw binary
avrdude: writing lfuse (1 bytes):

*Writing | | 0% 0.00s ***failed;*
Writing | ## | 100% 0.00s

avrdude: 1 bytes of lfuse written
avrdude: verifying lfuse memory against lfuse_backup_file_od2.0.hex:
avrdude: load data lfuse data from input file lfuse_backup_file_od2.0.hex:
avrdude: input file lfuse_backup_file_od2.0.hex auto detected as raw binary
avrdude: input file lfuse_backup_file_od2.0.hex contains 1 bytes
avrdude: reading on-chip lfuse data:

Reading | ## | 100% 0.00s

avrdude: verifying ...

avrdude: 1 bytes of lfuse verified

avrdude: safemode: Fuses OK (E:CB, H:D8, L:FF)

avrdude done. Thank you.

now efuse:

avrdude -p atmega32u4 -c avr109 -P /dev/ttyACM0 -U efuse:w:efuse_backup_file_od2.0.hex

Connecting to programmer: .
Found programmer: Id = "CATERIN"; type = S
* Software Version = 1.0; No Hardware Version given.*
Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
* Device code: 0x44*

avrdude: AVR device initialized and ready to accept instructions

Reading | ## | 100% 0.00s

avrdude: Device signature = 0x1e9587 (probably m32u4)
avrdude: reading input file "efuse_backup_file_od2.0.hex"
avrdude: input file efuse_backup_file_od2.0.hex auto detected as raw binary
avrdude: writing efuse (1 bytes):

*Writing | | 0% 0.00s ***failed;*
Writing | ## | 100% 0.00s

avrdude: 1 bytes of efuse written
avrdude: verifying efuse memory against efuse_backup_file_od2.0.hex:
avrdude: load data efuse data from input file efuse_backup_file_od2.0.hex:
avrdude: input file efuse_backup_file_od2.0.hex auto detected as raw binary
avrdude: input file efuse_backup_file_od2.0.hex contains 1 bytes
avrdude: reading on-chip efuse data:

Reading | ## | 100% 0.00s

avrdude: verifying ...
avrdude: 1 bytes of efuse verified

avrdude: safemode: Fuses OK (E:CB, H:D8, L:FF)

avrdude done. Thank you.

That's it - the firmware has been restored. You can unplug the OpenDrop / SparkFun Pro Micro from your Raspberry Pi now.

USEFUL LINKS & REFERENCES

- https://www.nongnu.org/avrdude/user-manual/avrdude_4.html
- https://askubuntu.com/questions/254835/how-can-i-know-the-usb-port
- https://forum.arduino.cc/index.php?topic=403201.0 (avrdude command for other target!)
- https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide/introduction
- https://www.hackster.io/rayburne/avr-firmware-duplicator (avrdude command for other target!)

- https://forum.arduino.cc/index.php?topic=453997.0 (concerning the error with flash verification)