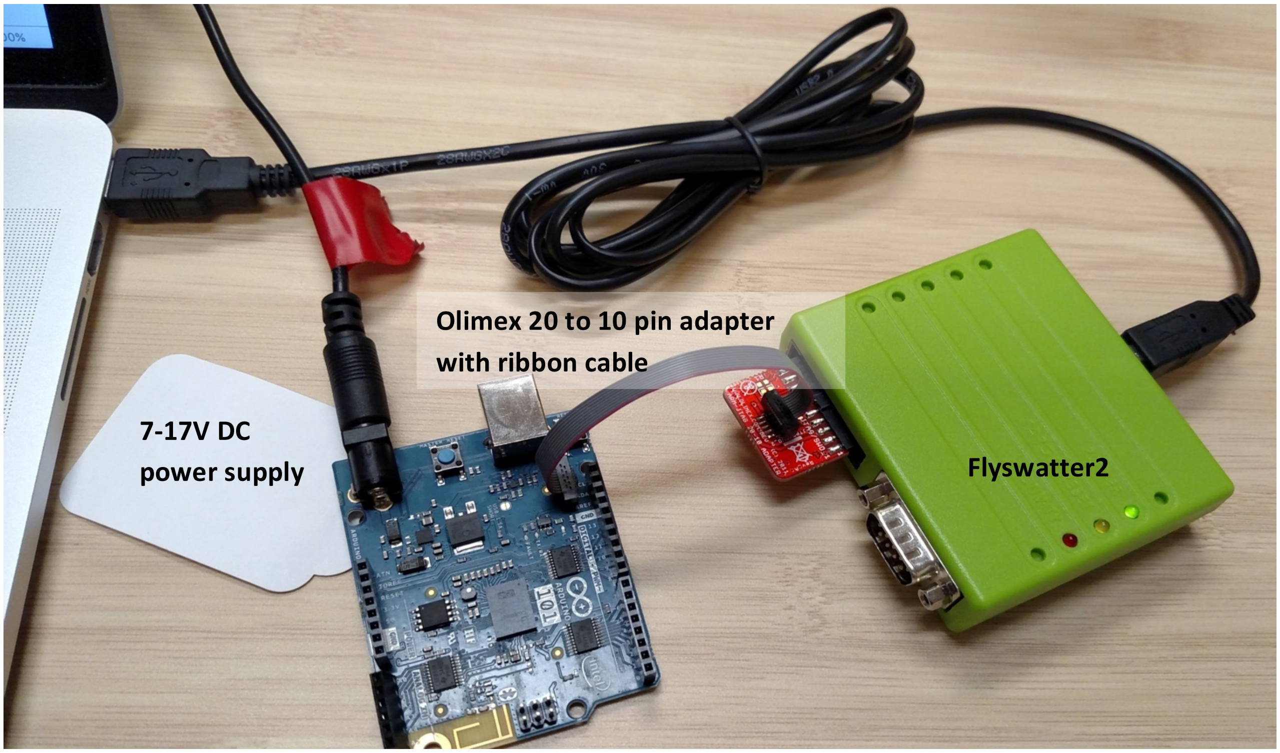
V1.1 2017-02-02

Arduino 101* Firmware Update

Table of Contents

Firmware update using JTAG through Flyswatter2

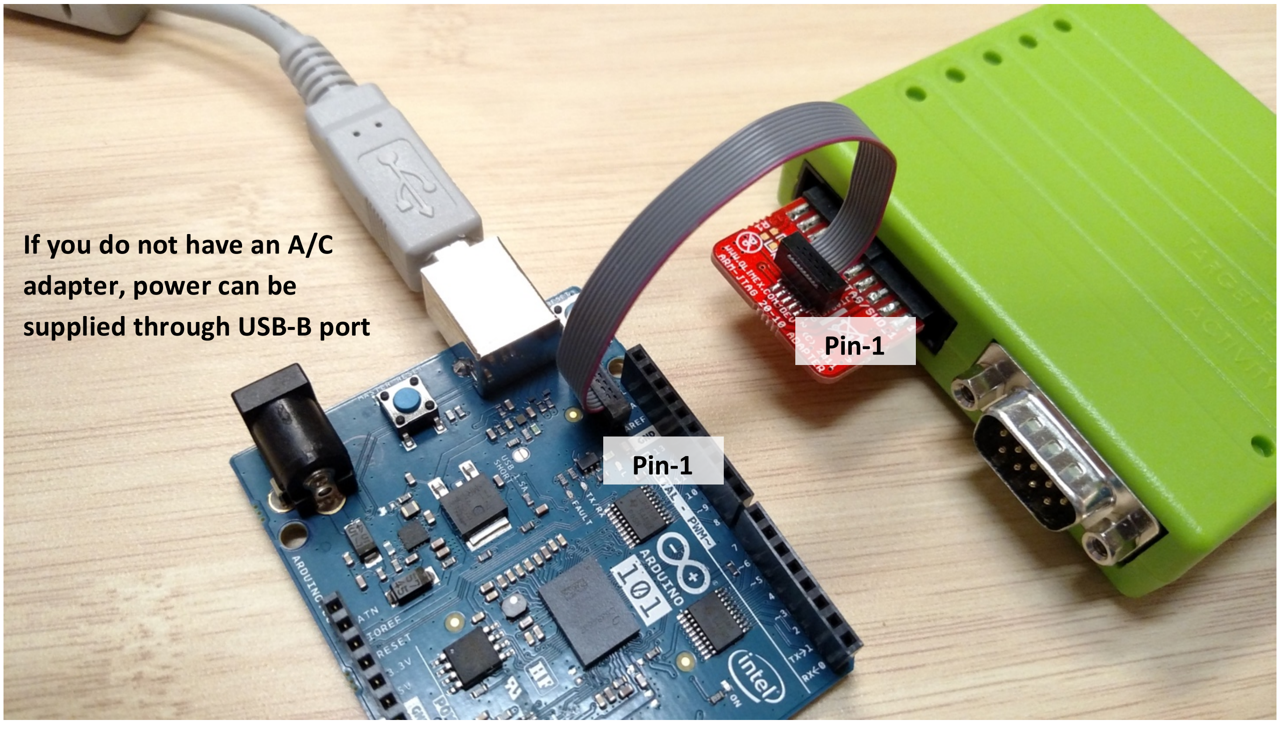
1. Connect all wires as shown in the picture:



Components list:

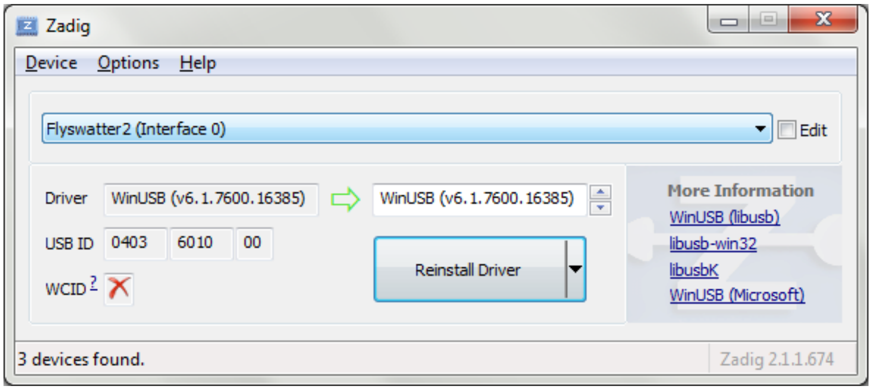
a. Windows/Linux/macOS computer
b. Flyswatter2 JTAG probe with USB-B cable
c. Olimex* 20 to 10 pin adapter with ribbon cable
d. Arduino 101 board with 7-17V DC power supply (or USB-B cable)

Make sure to attach ribbon cable correctly, as shown on the photo below:


If you do not have an A/C adapter, power can be supplied through USB-B port
Pin-1
Pin-1

2. Install Flyswatter2 probe drivers:

2.1. Windows



Flyswatter2 (Interface 0)

Driver WinUSB (v6.1.7600.16385) WinUSB (v6.1.7600.16385)

USB ID 0403 6010 00

WCID ?

3 devices found.

Zadig 2.1.1.674

a. Plug in Flyswatter2 probe to the host

b. Download and extract the latest firmware release

c. Go to `bin\` directory, run `zadig_2.1.1.exe`.

d. Options > List all devices.

e. Select your probe (Flyswatter2), pick WinUSB and hit Reinstall Driver; do it for Interface 0 and Interface 1.

f. Close Zadig and **REPLUG THE PROBE**

2.2. Linux

By default, non-root users will not have access to the JTAG pods connected via USB. You must grant write access to the proper `/dev/bus/usb` entry every time a device is connected to be able to run OpenOCD using a non-root account. The process can be automated by adding an udev rule. Simply create a text file in the rules directory:

```
$ sudo vim /etc/udev/rules.d/99-openocd.rules
```

The IDs depend on the JTAG device. For example, for the Flyswatter2 and the Olimex-ARM-USB-OCD-H, the rules file must have the following content:

```
SUBSYSTEM=="usb", ATTR{idVendor}=="0403", ATTR{idProduct}=="6010", MODE="0666"
SUBSYSTEM=="usb", ATTR{idVendor}=="15ba", ATTR{idProduct}=="002b", MODE="0666"
```

(See `drivers/rules.d/99-openocd.rules`)

You can also try flashing with `sudo`.

2.3 macOS

Download and install FTDI-D2XX driver: http://www.ftdichip.com/Drivers/D2XX.htm

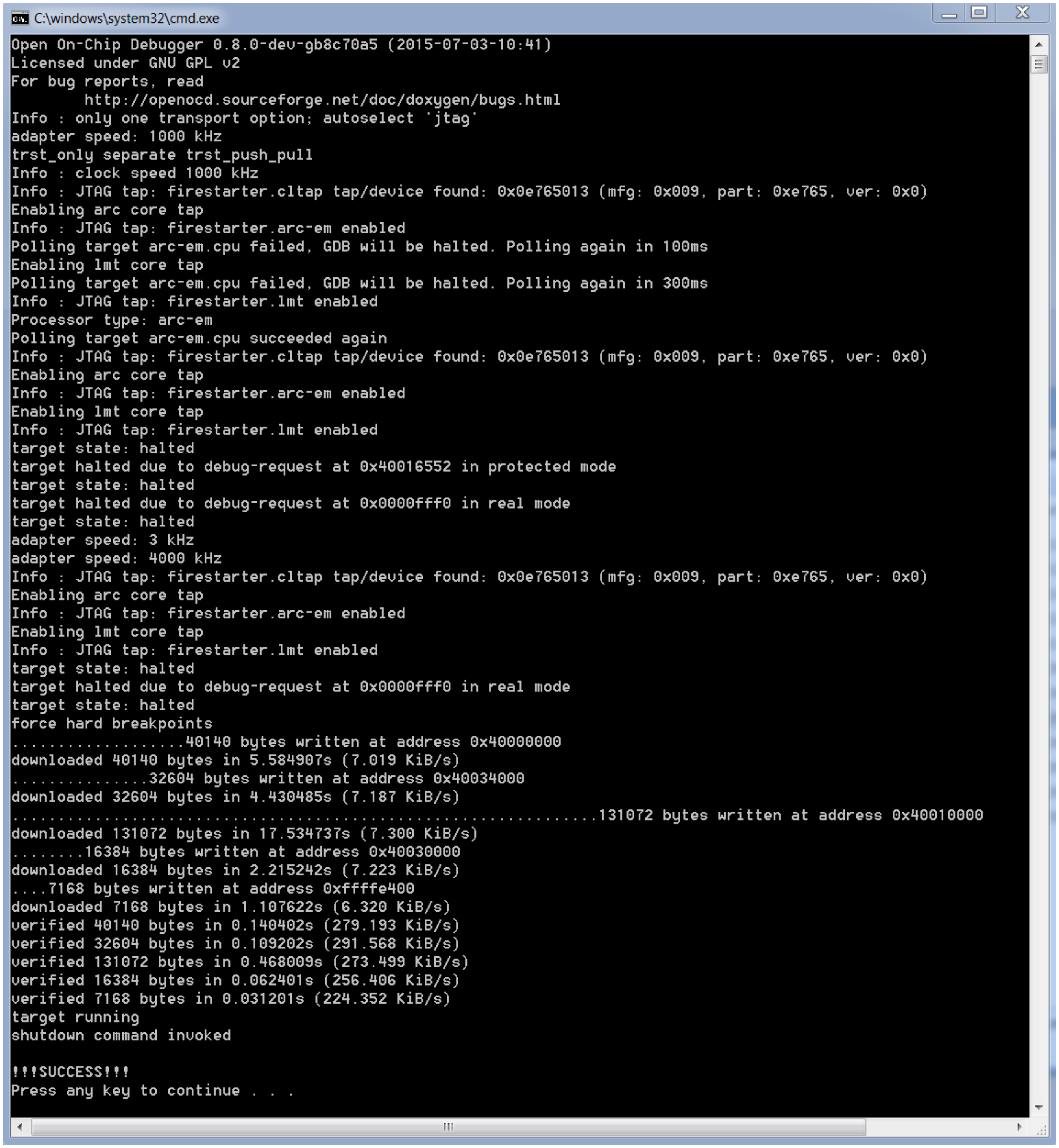
You may need to unload FTDI-VCP driver. See the included `OSX-FTDI-Fix.sh` and run it with `sudo` prior to flashing.

3. Flash the firmware

In the extracted flash pack directory, run the flashing script:

- **Windows:** Execute (double-click) **flash_jtag.bat** in order to flash production image.
- **Linux & macOS:** Run **flash_jtag.sh** in order to flash production image

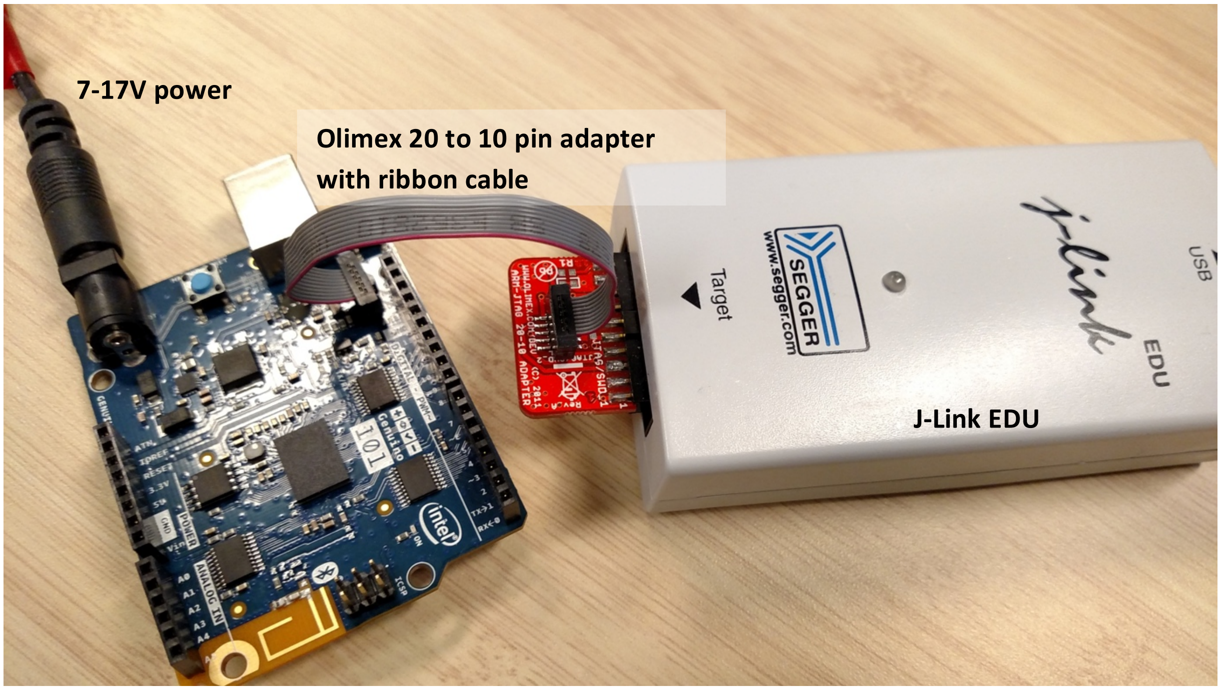
Flashing takes about a minutes, below is what a successful flash looks like:



```
Open On-Chip Debugger 0.8.0-dev-gb8c70a5 (2015-07-03-10:41)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.sourceforge.net/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'jtag'
adapter speed: 1000 kHz
trst_only separate trst_push_pull
Info : clock speed 1000 kHz
Info : JTAG tap: firestarter.cltap tap/device found: 0x0e765013 (mfg: 0x009, part: 0xe765, ver: 0x0)
Enabling arc core tap
Info : JTAG tap: firestarter.arc-em enabled
Polling target arc-em.cpu failed, GDB will be halted. Polling again in 100ms
Enabling lmt core tap
Polling target arc-em.cpu failed, GDB will be halted. Polling again in 300ms
Info : JTAG tap: firestarter.lmt enabled
Processor type: arc-em
Polling target arc-em.cpu succeeded again
Info : JTAG tap: firestarter.cltap tap/device found: 0x0e765013 (mfg: 0x009, part: 0xe765, ver: 0x0)
Enabling arc core tap
Info : JTAG tap: firestarter.arc-em enabled
Enabling lmt core tap
Info : JTAG tap: firestarter.lmt enabled
target state: halted
target halted due to debug-request at 0x40016552 in protected mode
target state: halted
target halted due to debug-request at 0x0000fff0 in real mode
target state: halted
adapter speed: 3 kHz
adapter speed: 4000 kHz
Info : JTAG tap: firestarter.cltap tap/device found: 0x0e765013 (mfg: 0x009, part: 0xe765, ver: 0x0)
Enabling arc core tap
Info : JTAG tap: firestarter.arc-em enabled
Enabling lmt core tap
Info : JTAG tap: firestarter.lmt enabled
target state: halted
target halted due to debug-request at 0x0000fff0 in real mode
target state: halted
force hard breakpoints
...................40140 bytes written at address 0x40000000
downloaded 40140 bytes in 5.584907s (7.019 KiB/s)
...............32604 bytes written at address 0x40034000
downloaded 32604 bytes in 4.430485s (7.187 KiB/s)
................................................................131072 bytes written at address 0x40010000
downloaded 131072 bytes in 17.534737s (7.300 KiB/s)
........16384 bytes written at address 0x40030000
downloaded 16384 bytes in 2.215242s (7.223 KiB/s)
....7168 bytes written at address 0xffffe400
downloaded 7168 bytes in 1.107622s (6.320 KiB/s)
verified 40140 bytes in 0.140402s (279.193 KiB/s)
verified 32604 bytes in 0.109202s (291.568 KiB/s)
verified 131072 bytes in 0.468009s (273.499 KiB/s)
verified 16384 bytes in 0.062401s (256.406 KiB/s)
verified 7168 bytes in 0.031201s (224.352 KiB/s)
target running
shutdown command invoked

!!!SUCCESS!!!
Press any key to continue . . .
```

Firmware update using JTAG through J-Link EDU

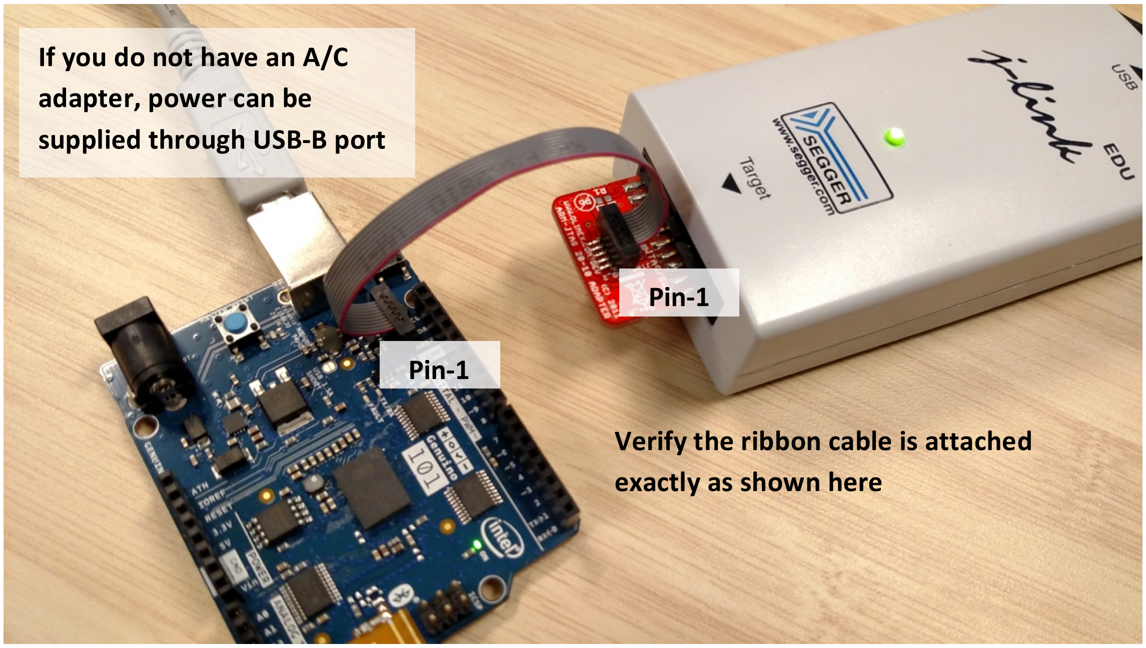
1. Connect all wires as shown on the picture:



Components list:

a. Windows/Linux/macOS computer
b. J-Link EDU probe with USB cable
c. Olimex 20 to 10 pin adapter with ribbon cable
d. Arduino 101 board with 7-17V DC power supply (or USB-B cable)

Make sure to attach ribbon cable correctly, as shown in the photo below:

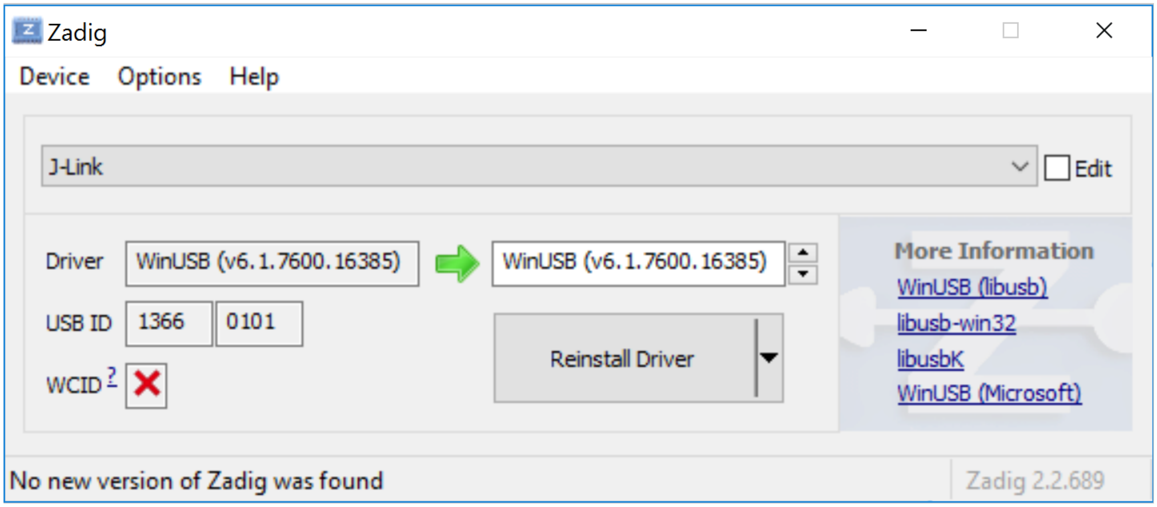


If you do not have an A/C adapter, power can be supplied through USB-B port

Verify the ribbon cable is attached exactly as shown here

2. Install J-Link EDU probe drivers:

2.1 Windows



J-Link

Driver WinUSB (v6.1.7600.16385) → WinUSB (v6.1.7600.16385)

USB ID 1366 0101

WCID ☒

No new version of Zadig was found

a. Plug in J-Link EDU probe to the host
b. Download and extract the latest firmware release
c. Go to `bin\` directory, run `zadig_2.1.1.exe`
d. Options > List all devices.
e. Select your probe (J-Link), pick WinUSB and hit Reinstall Driver
f. Close Zadig and **REPLUG THE PROBE**

2.2 Linux

By default, non-root users will not have access to the JTAG pods connected via USB. You must grant write access to the proper `/dev/bus/usb` entry every time a device is connected to be able to run OpenOCD using a non-root account. The process can be automated by adding an udev rule. Copy the `rules` file from `drivers` directory:

```
$ sudo cp drivers/rules.d/99-jlink.rules /etc/udev/rules.d/99-jlink.rules
```

You can also try flashing with `sudo`.

2.3 macOS

Download and install FTDI-D2XX driver: http://www.ftdichip.com/Drivers/D2XX.htm

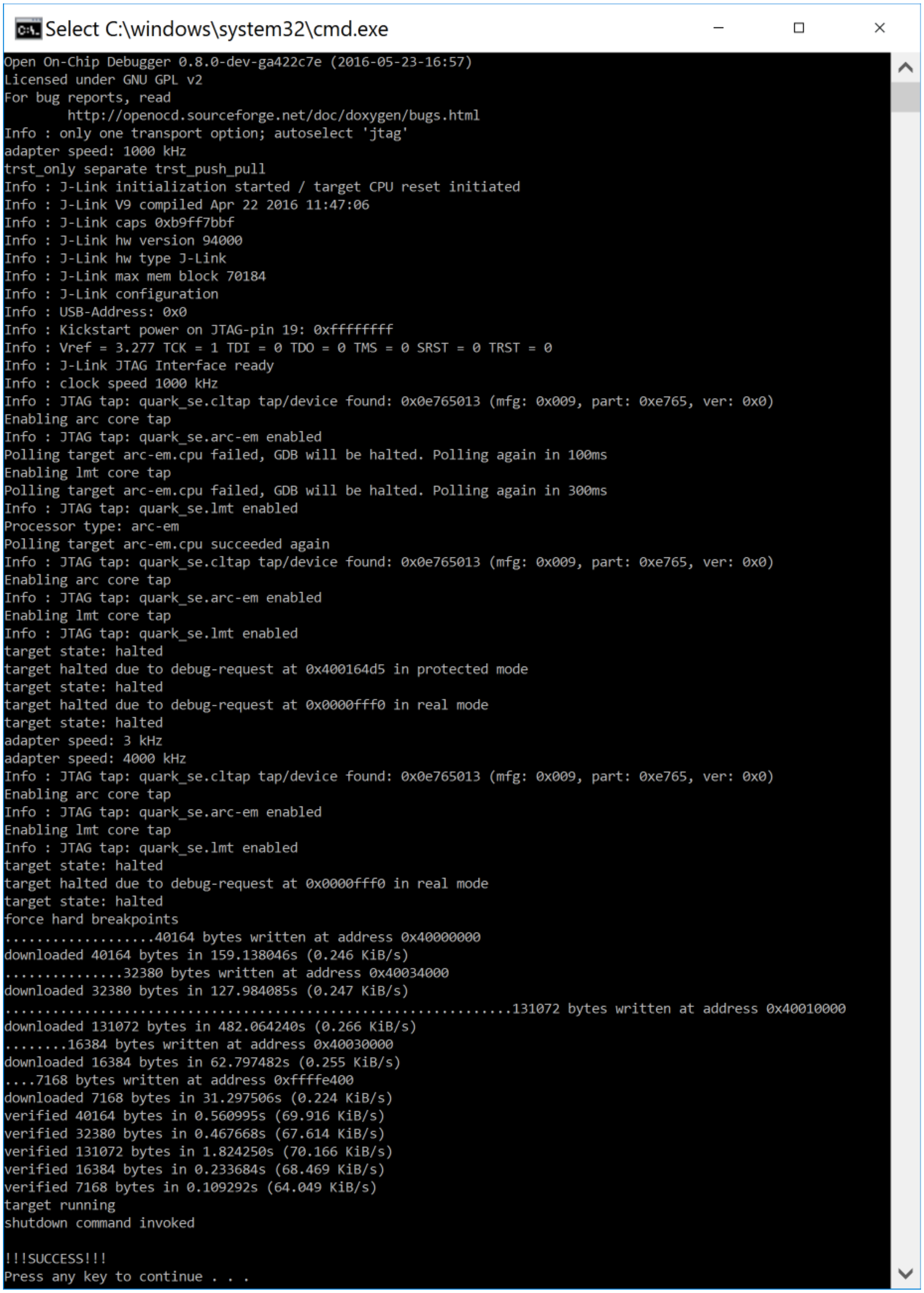
You may need to unload FTDI-VCP driver. See the included `OSX-FTDI-Fix.sh` and run it with `sudo` prior to flashing.

3 Flash the firmware

In the extracted flash pack directory, run the flashing script:

- **Windows:** Execute (double-click) **flash_jlink.bat** in order to flash production image
- **Linux & macOS:** Run **flash_jlink.sh** in order to flash production image

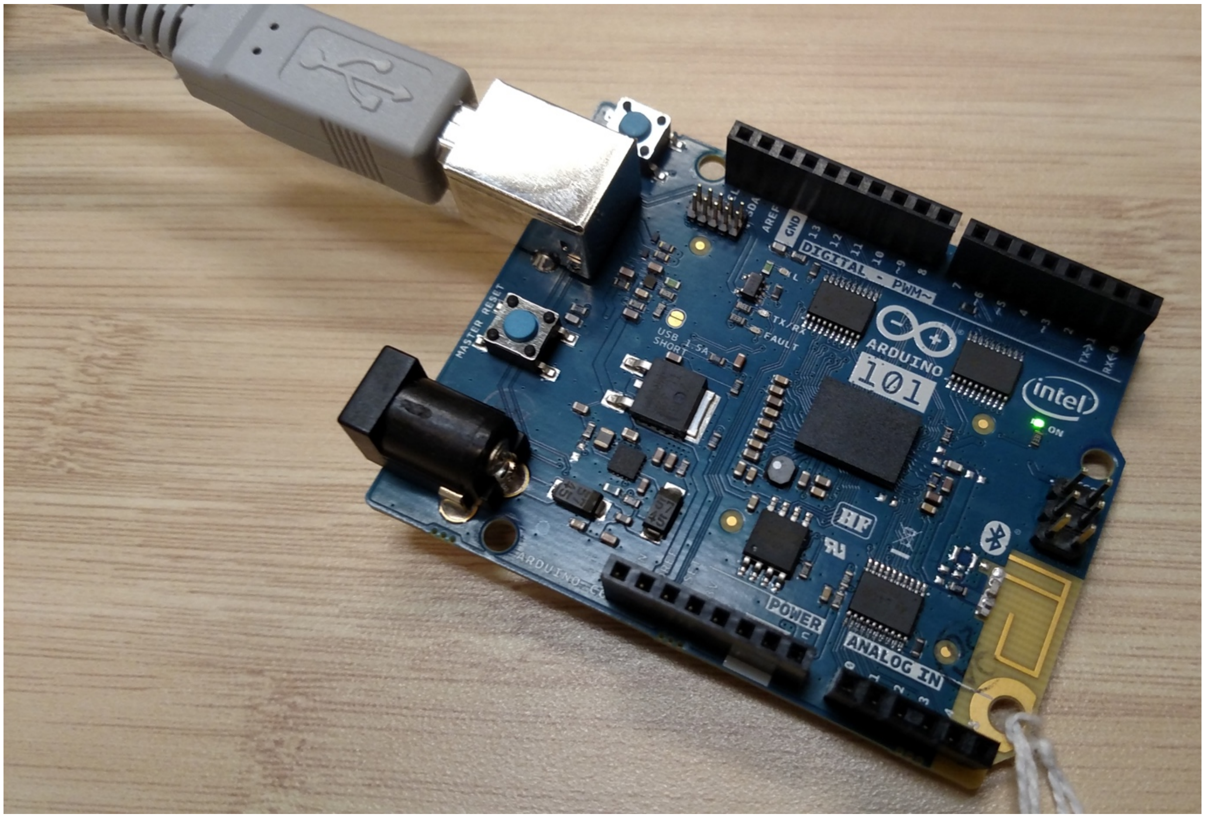
Flashing takes about 20 minutes, below is what a successful flash looks like:



```
Open On-Chip Debugger 0.8.0-dev-ga422c7e (2016-05-23-16:57)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.sourceforge.net/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'jtag'
adapter speed: 1000 kHz
trst_only separate trst_push_pull
Info : J-Link initialization started / target CPU reset initiated
Info : J-Link V9 compiled Apr 22 2016 11:47:06
Info : J-Link caps 0xb9ff7bbf
Info : J-Link hw version 94000
Info : J-Link hw type J-Link
Info : J-Link max mem block 70184
Info : J-Link configuration
Info : USB-Address: 0x0
Info : Kickstart power on JTAG-pin 19: 0xffffffff
Info : Vref = 3.277 TCK = 1 TDI = 0 TDO = 0 TMS = 0 SRST = 0 TRST = 0
Info : J-Link JTAG Interface ready
Info : clock speed 1000 kHz
Info : JTAG tap: quark_se.cltap tap/device found: 0x0e765013 (mfg: 0x009, part: 0xe765, ver: 0x0)
Enabling arc core tap
Info : JTAG tap: quark_se.arc-em enabled
Polling target arc-em.cpu failed, GDB will be halted. Polling again in 100ms
Enabling lmt core tap
Polling target arc-em.cpu failed, GDB will be halted. Polling again in 300ms
Info : JTAG tap: quark_se.lmt enabled
Processor type: arc-em
Polling target arc-em.cpu succeeded again
Info : JTAG tap: quark_se.cltap tap/device found: 0x0e765013 (mfg: 0x009, part: 0xe765, ver: 0x0)
Enabling arc core tap
Info : JTAG tap: quark_se.arc-em enabled
Enabling lmt core tap
Info : JTAG tap: quark_se.lmt enabled
target state: halted
target halted due to debug-request at 0x400164d5 in protected mode
target state: halted
target halted due to debug-request at 0x0000fff0 in real mode
target state: halted
adapter speed: 3 kHz
adapter speed: 4000 kHz
Info : JTAG tap: quark_se.cltap tap/device found: 0x0e765013 (mfg: 0x009, part: 0xe765, ver: 0x0)
Enabling arc core tap
Info : JTAG tap: quark_se.arc-em enabled
Enabling lmt core tap
Info : JTAG tap: quark_se.lmt enabled
target state: halted
target halted due to debug-request at 0x0000fff0 in real mode
target state: halted
force hard breakpoints
...................40164 bytes written at address 0x40000000
downloaded 40164 bytes in 159.138046s (0.246 KiB/s)
...............32380 bytes written at address 0x40034000
downloaded 32380 bytes in 127.984085s (0.247 KiB/s)
................................................................131072 bytes written at address 0x40010000
downloaded 131072 bytes in 482.064240s (0.266 KiB/s)
........16384 bytes written at address 0x40030000
downloaded 16384 bytes in 62.797482s (0.255 KiB/s)
....7168 bytes written at address 0xffffe400
downloaded 7168 bytes in 31.297506s (0.224 KiB/s)
verified 40164 bytes in 0.560995s (69.916 KiB/s)
verified 32380 bytes in 0.467668s (67.614 KiB/s)
verified 131072 bytes in 1.824250s (70.166 KiB/s)
verified 16384 bytes in 0.233684s (68.469 KiB/s)
verified 7168 bytes in 0.109292s (64.049 KiB/s)
target running
shutdown command invoked

!!!SUCCESS!!!
Press any key to continue . . .
```

Firmware update using USB

1. Connect USB cable to board as shown



Components listing:

a. Windows/Linux/macOS computer
b. USB cable
c. Arduino 101 board
d. 7-17V DC power supply (or USB-B cable)

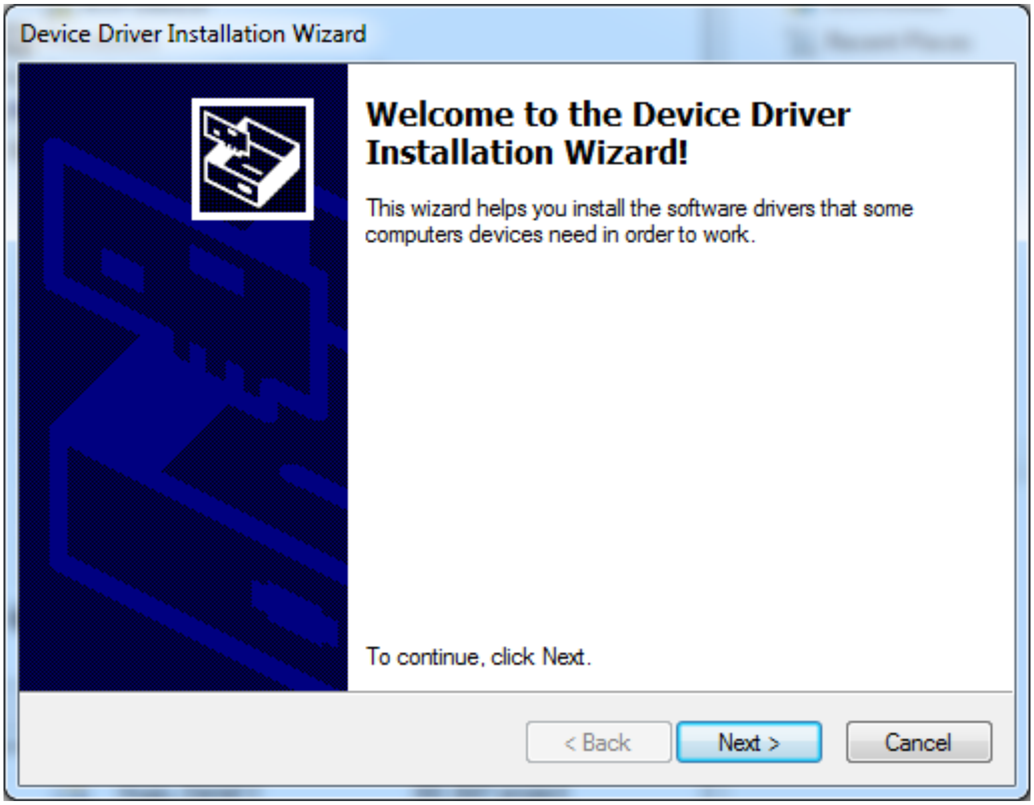
2. Install drivers

2.1. Windows

a. Download and extract the latest firmware release

b. In the extracted flash pack directory, go to `drivers\Windows\` directory and run

- `dpinst-amd64.exe` on 64-bit Windows
- `dpinst-x86.exe` on 32-bit Windows



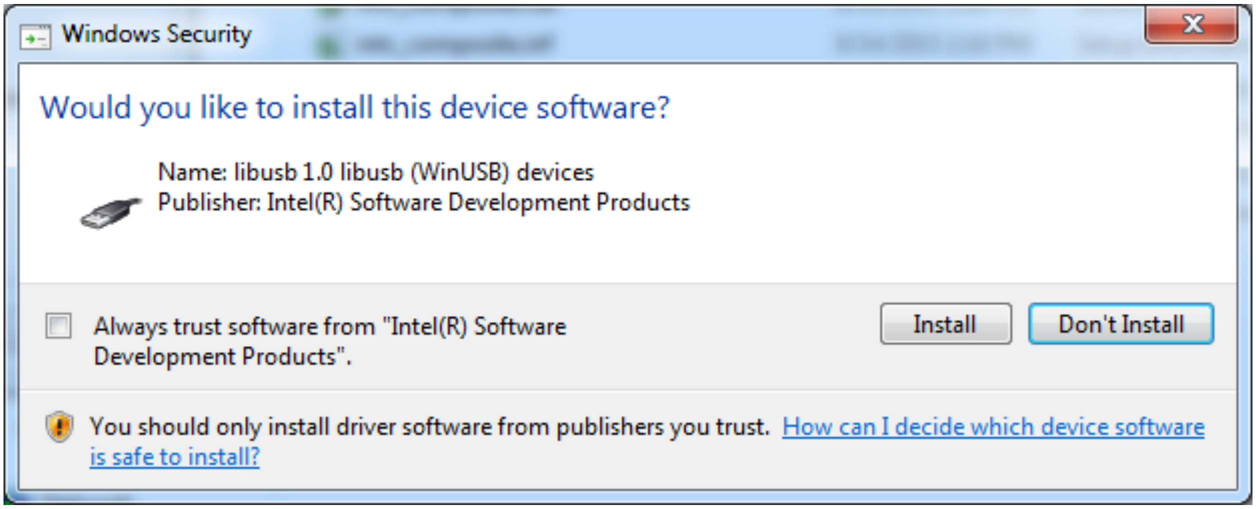
Device Driver Installation Wizard

Welcome to the Device Driver Installation Wizard!

This wizard helps you install the software drivers that some computers devices need in order to work.

To continue, click Next.

c. Click **Next**



Windows Security

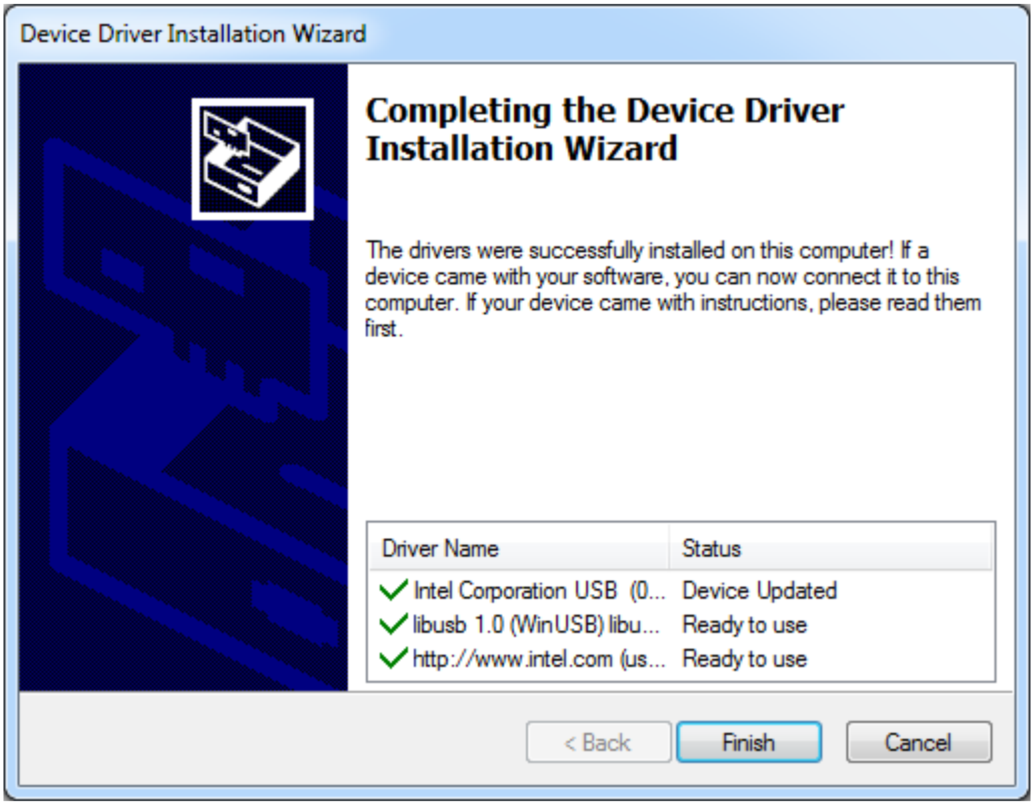
Would you like to install this device software?

Name: libusb 1.0 libusb (WinUSB) devices
Publisher: Intel(R) Software Development Products

☐ Always trust software from "Intel(R) Software Development Products".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

d. Click **Install**



Device Driver Installation Wizard

Completing the Device Driver Installation Wizard

The drivers were successfully installed on this computer! If a device came with your software, you can now connect it to this computer. If your device came with instructions, please read them first.

Driver Name	Status
✓ Intel Corporation USB (0...	Device Updated
✓ libusb 1.0 (WinUSB) libu...	Ready to use
✓ http://www.intel.com (us...	Ready to use

e. Click **Finish**

2.2. Linux

The DFU device can be set up by editing a text file in the rules directory.

Enter **ONE RULE PER LINE**. Newline characters are not allowed.

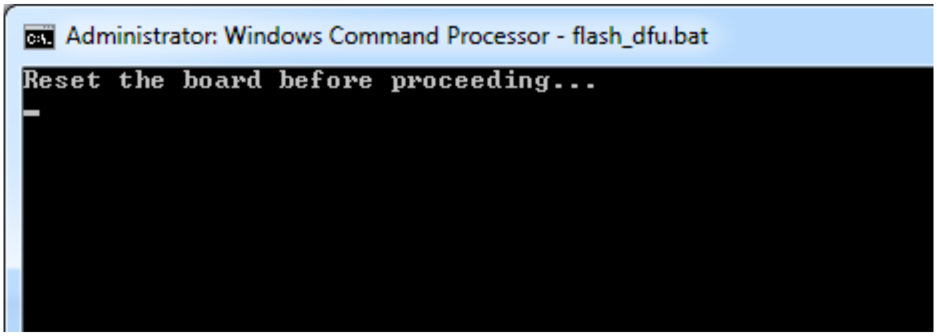
```
sudo vi /etc/udev/rules.d/99-dfu.rules

# Arduino 101 in DFU Mode
SUBSYSTEM=="tty", ENV{ID_REVISION}=="8087", ENV{ID_MODEL_ID}=="0ab6", MODE="0666", ENV{ID_MM_DEVICE_IGNORE}="1", ENV{ID_MM_CANDIDATE}="0"
SUBSYSTEM=="usb", ATTR{idVendor}=="8087", ATTR{idProduct}=="0aba", MODE="666", ENV{ID_MM_DEVICE_IGNORE}="1"
```

(See `drivers/rules.d/99-dfu.rules`)

3. Flash the firmware

In the extracted flash pack directory, run the flashing script:

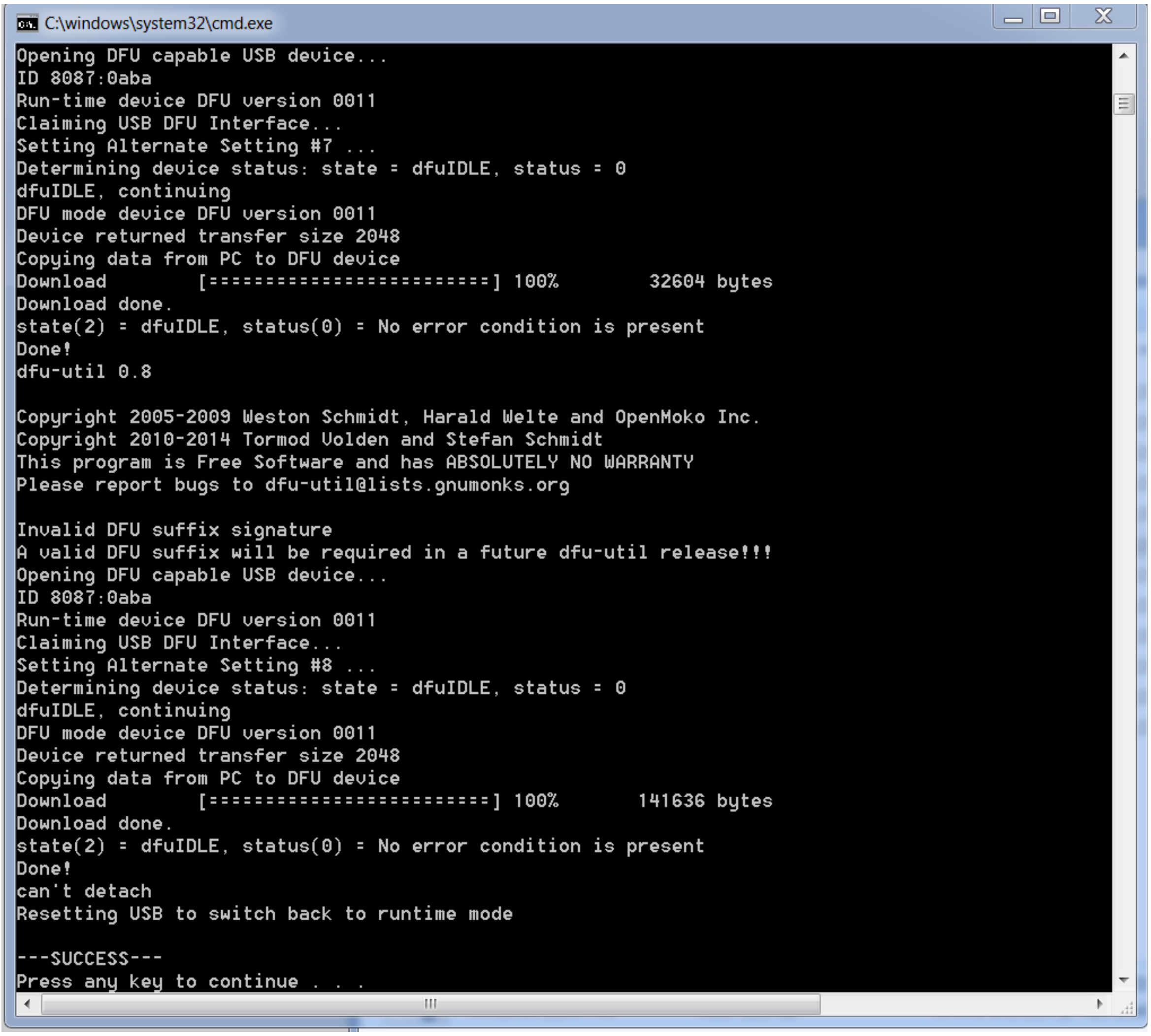
- **Windows:** Execute (double-click) **flash_dfu.bat** in order to flash production image
- **Linux & macOS:** Run **flash_dfu.sh** in order to flash production image



```
Reset the board before proceeding...
_
```

3.1 Press the **reset** button on the board to start the flash process

Flashing takes about a minute, below is what a successful DFU flash looks like:



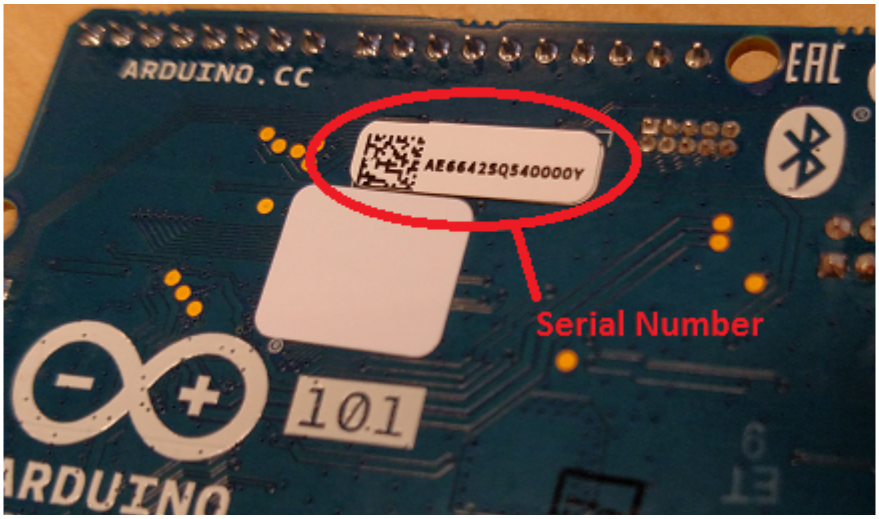
```
Opening DFU capable USB device...
ID 8087:0aba
Run-time device DFU version 0011
Claiming USB DFU Interface...
Setting Alternate Setting #7 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 0011
Device returned transfer size 2048
Copying data from PC to DFU device
Download        [=========================] 100%        32604 bytes
Download done.
state(2) = dfuIDLE, status(0) = No error condition is present
Done!
dfu-util 0.8

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2014 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to dfu-util@lists.gnumonks.org

Invalid DFU suffix signature
A valid DFU suffix will be required in a future dfu-util release!!!
Opening DFU capable USB device...
ID 8087:0aba
Run-time device DFU version 0011
Claiming USB DFU Interface...
Setting Alternate Setting #8 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 0011
Device returned transfer size 2048
Copying data from PC to DFU device
Download        [=========================] 100%       141636 bytes
Download done.
state(2) = dfuIDLE, status(0) = No error condition is present
Done!
can't detach
Resetting USB to switch back to runtime mode

---SUCCESS---
Press any key to continue . . .
```

Appendix

1. Flashing multiple Arduino101 board simultaneously

By default, users can flash to only one board at a time. If more than one DFU device is visible to the host system, an error will occur. When flashing firmware to multiple boards at the same time, specify a serial number.

a. Get the board serial number:



b. Specify the serial number when running the flash_dfu script:

   i. **Linux**

      e.g.,

      ```
      flash_dfu.sh AE642SQ34000Y
      ```

   ii. **Windows**

      e.g.,

      ```
      flash_dfu.bat AE642SQ34000Y
      ```