Updating STMBL Firmware.

By John Dammeyer
Automation Artisans Inc.
www.autoartisans.com

Synopsis:

This document describes how to update software in the STMBL Servo Driver using a using a Raspberry Pi3 with a direct or WiFI internet connection, an HDMI monitor, a USB keyboard and a mouse. The monitor and keyboard/mouse were used so we could test run the servoterm software. Access to the Pi3 is also possible via puTTY with wireless and TightVNVC Viewer when tightvncserver is running on the Pi.

Setting up Communication Software:

Start by creating a directory 'stmbl' under ~ which is your home directory.

```
cd ~
mkdir stmbl
cd ~/stmbl
```

We want to be able to talk to the STMBL via USB and the STMBL monitor program called Servoterm.

```
sudo git clone https://github.com/STMBL/Servoterm-app
```

This creates a directory called

```
~/stmbl/Servoterm-app/Servoterm
```

Now we'll instruct Chrome to find the unpacked extension as per directions in the Andy Pugh Getting Started document. Once installed and running and you have the STMBL plugged into one of the USB ports you can click on Connect and if all is well you'll see

```
Connecting to /dev/ttyACM0
Connected
```

If you don't get the 'connected' or it connects and disconnects, then reboot the Pi3.

An apparent problem with Pi3 Jesse exists. The user entry line moves down with text until it vanishes off the end of the screen. This shrinks the plot area so it doesn't show the entire plot. Click on the Clear button to restore the line so you can see it. If the button row hasn't scrolled up off the screen. For some reason the slider on the side doesn't work. Right click and reload app. The Pi3 Stretch version doesn't appear to have this problem.

Installing the Cross Compiler for the ST Processors:

Next you need the ST cross compiler to make the project. Use this link to for instructions to install the tools. The ArmInArm Github page has instructions which are duplicated here. First make sure your system is up to date.

```
sudo apt-get update
```

The ‘upgrade’ isn’t normally needed if you’ve just created a MicroSD card from the latest image.

```
sudo apt-get upgrade
```

You'll want to install the following additional packages on Raspbian Jessie 2015-11-21 or newer. These packages provide support for USB and communications software in case you want serial port access to the ST processor using other terminal software like minicom.

```
sudo apt-get -y install minicom screen autoconf libusb-1.0-0-dev libtool libftdi-dev texinfo
```

Setup and install software for the ARMinARM board by cloning from github, and run setup.

```
sudo git clone https://github.com/ARMinARM/arminarm
```

This has created a directory called `~/stmbl/arminarm`

```
cd arminarm
sudo ./setup
```

Don't forget the sudo or the 'i' option to load the dfu_util won't work. When you run setup, you'll see a menu.

```
###############################################################################
#                              ARMinARM                                       #
###############################################################################

Essentials:
    0) Update Self
    1) Update/Install ARMinARM + GCC Toolchain
    2) Add /opt/arminarm* to PATH env (needs reboot)
    3) Disable serial port (required for ARMinARM board, needs reboot)
    4) Enable serial port (for booting RPI over serial port, default)

Fast start:
    10) Upload espruino.bin to ARMinARM board
    11) Upload elua.bin to ARMinARM board

Source code:
    a) Update/Install CMSIS_StdPeriph Examples
    b) Update/Install Espruino source code
    c) Update/Install esp-cli
    d) Update/Install eLua source code
    e) Update/Install libmaple
    f) Update/Install libopencm3-prj
    g) Update/Install OpenOCD
    h) Update/Install ST-Link
    i) Update/Install dfu-util
    j) Update/Install stm32flash

    q) Quit

Enter your choice:
```

You'll want to run the numeric options (1-2) to install the basic tools.

We need dfu-util so choose 'i' too.

Don't forget to do:

```
sudo reboot
```

Now back to our working directory.

```
cd ~/stmbl
```

First on the command line type

```
lsusb
```

Your Pi will be different but you should see the 0483:5740 VID:PID pair which shows your Pi3 is talking to the STMBL.

```
Bus 001 Device 082: ID 045e:00db Microsoft Corp. Natural Ergonomic Keyboard 4000 V1.0
Bus 001 Device 081: ID 04d9:048e Holtek Semiconductor, Inc. Optical Mouse
Bus 001 Device 080: ID 0483:5740 STMicroelectronics STM32F407
Bus 001 Device 079: ID 1a40:0101 Terminus Technology Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

After a change to enable the ST bootloader you would see 0483:df11 for the STM device. More on that later.

```
Bus 001 Device 081: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
```

You'd also see this on the F3 board if you short the boot jumper while powering up.

Downloading the latest version of STMBL code:

Now let's get the source code for the STMBL. First, back to our ~/stmbl directory.

```
cd ~/stmbl
git clone https://github.com/rene-dev/stmbl
```

then go into the new folder ~/stmbl/stmbl. For safety I always rename the folder if it's there to stmbl-old before I clone the new one. It always lets me easily go back to a previous version.

Programming the STMBL F3 and F4 directly using the USB connectors on each board:

An earlier "Makefile" included calls to the dfu-util program to directly access the ST embedded bootloader. That's because older software, as supplied in some units doesn't allow programming the F3 HV board through the F4 interface board. It's also the last resort in case somehow the F4 code was damaged and it's no longer possible to interact with it.

Inside the older `~/stmbl/stmbl/Makefile` is the following text near the end of the file.

```
# Flash the device
#
btburn: build showsize $(TARGET).dfu
	@tools/bootloader.py
	@sleep 1
	@sudo dfu-util -d 0483:df11 -a 0 -s 0x08010000:leave -D $(TARGET).dfu
```

The dfu-util in the line above should be preceded by sudo because the bootloader.py program resets the processor into bootloader mode by sending the string "bootloader" which changes the USB device from VID:PID 0483:5740 to 0483:df11 which is the STM loader DFU PID. Once that happens the dfu-util command cannot work with the port unless it's in super user mode.

If this information isn't there then there will be links to other script files that do this for you and you will be using the newer version. The linked makefiles will need to be modified. See Appendix 1 for the list of commands to the make utility.

Now that the USB device has the new VID:PID the dfu-util program can send down the file identified in the make file as $(TARGET) which is obj_app/stmbl.dfu

From the console window you could also type:

```
sudo dfu-util -d 0483:df11 -a 0 -s 0x08010000:leave -D obj_app/stmbl.dfu
```

But if you have added the sudo into the Makefile it's not needed.

First:

```
make btburn = flash the f4 firmware (without the f4 bootloader) with dfu-util
make boot_btburn = flash the f4 bootloader with dfu-util
make f3_btburn = flash the f3 firmware (without f3 bootloader) with dfu-util
make f3_boot_btburn = flash f3 bootloader with dfu-util
```

for all targets: if you replace btburn with flash (eg make f3_boot_flash) it will flash this target with stlink over swd instead of dfu-util

Old ---

Let's determine what revision of code you have by examining the Makefile.

```
make btburn
```

has been superceeded by

```
make boot_btburn
```

It will put the STMBL into boot loader mode after the compile is finished and update the firmware.

Next:

f3_btburn and f3_boot_btburn only work if you put the hv board into bootloader mode with the boot jumper (we removed usb support from the f3 firmware)

if the f3 bootloader and firmware are newer than 7.4.2018 you can update the f3 firmware from the f4 to do this:

flash current f4 firmware (this includes the f3 firmware) "make clean btburn"

power the hv board

in servoterm "hv_update" (repeat if you get something like "hv_update: FLASH_FAILED")

For the bottom HV board you will need to edit the "Makefile" in the stm32f303 folder to also have sudo in front of the dfu_util. Then plug in the USB cable into the bottom board. You'll have to remove the fan to do this.

Now short the boot jumper and power up the bottom board. I used a small screw driver to bridge the pads. Once it's powered up you can remove the short.

Next from the `~/stmbl/stmbl` folder

```
make f3_btburn
```

Depending on how fast the USB system reacts to the new 0483:df11 VID:PID combination you may have to do this twice before the USB port is properly detected. If you had a working system you know the F3 board is running the bootloader if the F4 board flashes a red error LED.

You may see a message that tells you the port is not found or already in bootoader mode. Once the code has been uploaded into the processor power, it down, move the USB cable back to the top board and cycle power. With the Pi3 Servoterm there may be issues when the port was disconnected. This may be due to the port VID:PID changing during boot loading. So far the only solution is to reboot the Pi3.

Programming the STMBL F3 through F4 directly using Servoterm:

At this point in the process on to upgrade the F4 and F3 modules isn't complete. It's possible to tell the F4 module to program the F3 if the F4 has the command to do this. More in the next revision.

Appendix 1:

```
#generate hal and command tables
tbl:
	@echo Generating tables
	@$(PYTHON) tools/create_hal_tbl.py . $(COMPS)
	@$(PYTHON) tools/create_config.py conf/template/* > src/conf_templates.c
	@$(PYTHON) tools/create_cmd.py $(SOURCES) > inc/commandslist.h

#build f4 bootloader
boot:
	$(MAKE) -f bootloader/Makefile

#flash f4 bootloader using stlink
boot_flash: boot
	$(MAKE) -f bootloader/Makefile flash

#flash f4 bootloader using df-util
boot_btburn: boot
	$(MAKE) -f bootloader/Makefile btburn


#build f3 bootloader
f3_boot:
	$(MAKE) -f f3_boot/Makefile

#flash f3 bootloader using stlink
f3_boot_flash:
	$(MAKE) -f f3_boot/Makefile flash

#flash f3 bootloader using df-util
f3_boot_btburn:
	$(MAKE) -f f3_boot/Makefile btburn
```

```
#build f3 firmware
f3:
	$(MAKE) -f stm32f303/Makefile

#flash f3 firmware using stlink
f3_flash:
	$(MAKE) -f stm32f303/Makefile flash

#flash f3 firmware using df-util
f3_btburn:
	$(MAKE) -f stm32f303/Makefile btburn


#generate f3 firmware object from f3 bin
hv_firmware.o:
	$(MAKE) -f stm32f303/Makefile all

#build f103 firmware for V3 hardware
f1:
	$(MAKE) -f stm32f103/Makefile

#flash f103 firmware for V3 hardware using stlink
f1_flash: boot
	$(MAKE) -f stm32f103/Makefile flash

deploy: f3_boot f3 boot build binall

binall:
	cat obj_boot/blboot.bin /dev/zero | head -c 32768 > f4.bin
	cat conf/festo.txt /dev/zero | head -c 32768 >> f4.bin
	cat obj_app/stmbl.bin >> f4.bin
	cat obj_f3_boot/f3_boot.bin /dev/zero | head -c 16384 > f3.bin
	cat obj_hvf3/hvf3.bin >> f3.bin
```