Micropytonize Limifrog

Tobias Badertscher

May 11, 2016

The following document describes how to put Micropython on Limifrog. I assume that you have a Linux operation system (eg. Debian) and the related toolchain installed (`make` and `gcc-arm-none-eabi`). For other OSs please adapt the description to your environment.

The bash shell in/output is written in `verbatim`, code you should enter in the bash shell is prepended by > and this character must not be entered.

1 Download the source code

Donwload the micropython source code by opening a bash console in the target folder and enter:

```
> git clone https://github.com/micropython/micropython.git
```

This will result in a subfolder `micropython`.

2 Building the binaries

In the bash shell enter:

```
> cd micropython/stmhal/
```

and build the LIMIFROG software with:

```
> make BOARD=LIMIFROG
Use make V=1 or set BUILD_VERBOSE in your environment to increase build verbosity.
mkdir -p build-LIMIFROG/genhdr
```

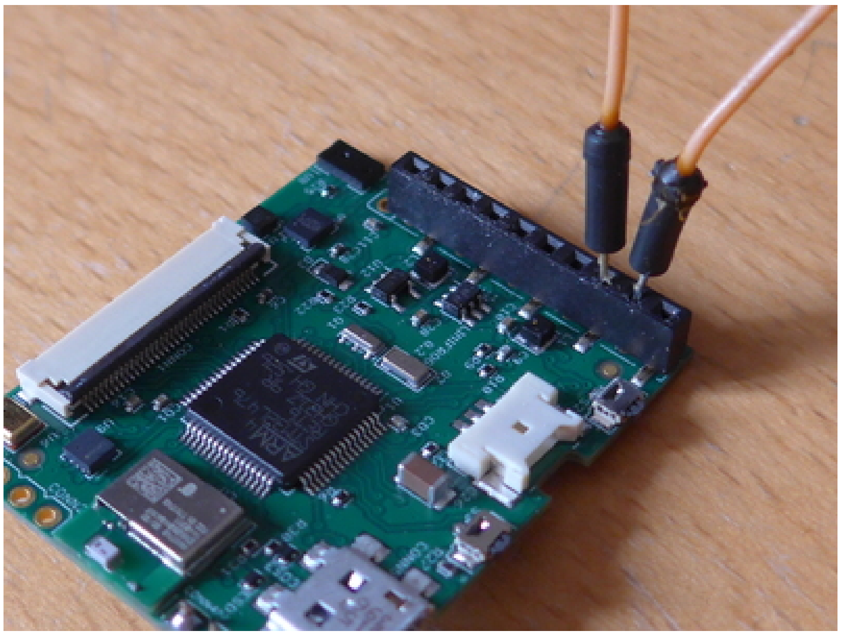


Figure 1: Place a shortcut between Pin 9 and pin 10 of the Limifrog Extension port.

```
....
LINK build-LIMIFROG/firmware.elf
   text    data     bss     dec     hex filename
 286880     104   27708  314692   4cd44 build-LIMIFROG/firmware.elf
Create build-LIMIFROG/firmware.dfu
Create build-LIMIFROG/firmware.hex
```

At the end of the build there will be a subfolder `build-LIMIFROG` and there you will find the binary `firmware.dfu` we will deploy in the next step to Limifrog.

3 Deploy the binary with dfu

To put Limifrog in the update mode (dfu-mode) connect pin 9 and pin 10 of limifrogs extension port (Fig. 1).

You will now see a connected dfu board (Bus/device will maybe be different on your PC).

```
> lsusb
Bus 010 Device 018: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
```

Now deploy the code by calling the `deploy` make target:

```
> make BOARD=LIMIFROG deploy
Use make V=1 or set BUILD_VERBOSE in your environment to increase build verbosity.
Writing build-LIMIFROG/firmware.dfu to the board
File: build-LIMIFROG/firmware.dfu
    DfuSe v1, image size: 287285, targets: 1
    Target 0, alt setting: 0, name: "ST...", size: 287000, elements: 2
      0, address: 0x08000000, size: 392
      1, address: 0x08020000, size: 286592
    usb: 0483:df11, device: 0x0000, dfu: 0x011a, UFD, 16, 0xea3bb0db
Writing memory...
0x08000000     392 [=========================] 100%
0x08020000  286592 [=========================] 100%
Exiting DFU...
```

which will take a few seconds.
If the make commands complains with following message:

```
> make BOARD=LIMIFROG deploy
Writing build-LIMIFROG/firmware.dfu to the board
Traceback (most recent call last):
  File "../tools/pydfu.py", line 20, in <module>
    import usb.core
ImportError: No module named usb.core
make: *** [deploy] Error 1
```

maybe the `python3-usb` package is missing on your system. Otherwise open the makefile with your favourite text editor, search for the line

```
USE_PYDFU ?= 1
```

and set the value to 0. In this case you should have the `dfu-util` package installed on your system. If this still does not work try to execute the command as `sudo`:

```
> sudo make BOARD=LIMIFROG deploy
```

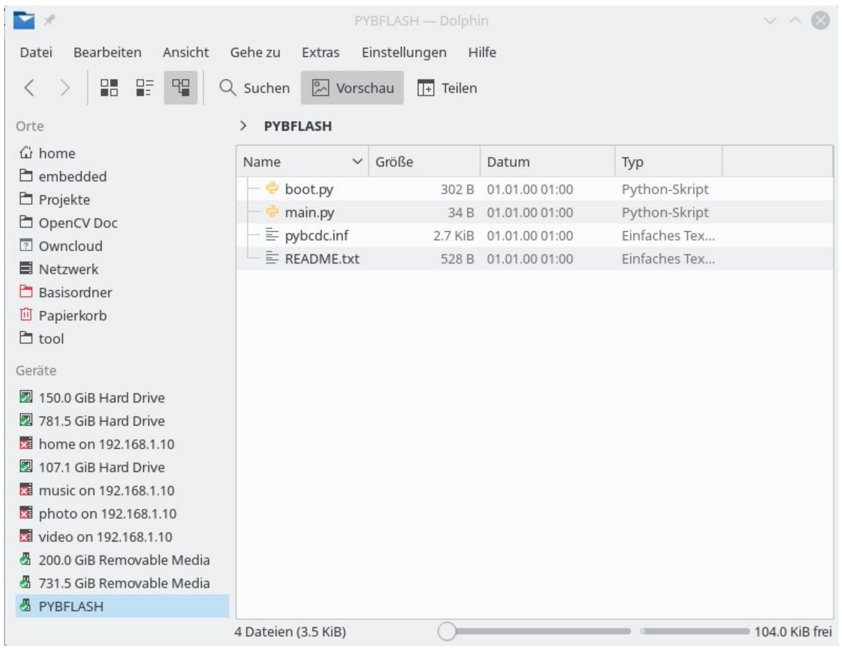
4 Deploy the binary with STLINK

To deploy micropython to Limifrog with ST-LINK connect your emulator to Limifrog as described in the Limifrog documentation. You will see a connected ST-LINK device on the USB bus (Bus/device will maybe be different on your PC):

```
> lsusb
Bus 010 Device 021: ID 0483:3748 STMicroelectronics ST-LINK/V2
```

Now deploy the code by calling the `deploy-stlink` make target:

```
> make BOARD=LIMIFROG deploy-stlink
Use make V=1 or set BUILD_VERBOSE in your environment to increase build verbosity.
Writing build-LIMIFROG/firmware0.bin to the board via ST-LINK
...
Flash page at addr: 0x08000000 erased
```



PYBFLASH

Name	Größe	Datum	Typ
boot.py	302 B	01.01.00 01:00	Python-Skript
main.py	34 B	01.01.00 01:00	Python-Skript
pybcdc.inf	2.7 KiB	01.01.00 01:00	Einfaches Tex...
README.txt	528 B	01.01.00 01:00	Einfaches Tex...

Figure 2: Contend of Limifrog USB-memory device.

```
....
Writing build-LIMIFROG/firmware1.bin to the board via ST-LINK
....
2016-05-07T19:38:46 INFO src/stlink-common.c: Successfully loaded flash loader in sram
size: 32768
...
size: 24448
2016-05-07T19:38:52 INFO src/stlink-common.c: Starting verification of write complete
2016-05-07T19:38:55 INFO src/stlink-common.c: Flash written and verified! jolly good!
```

5 Using micropython

Unplug Limifrog and then plug the micro-usb plug again into your Limifrog. Your PC will detect a new USB memory device which you can open (Fig. 2).
There is example python-code to run on Limifrog on github:

```
> git clone https://github.com/tobbad/micropython_lib.git
```

This repo is highly experimental but can be used as first step to write your own code. For the Repo structure please consult the README.md file. In the board specific subfolder (eg. `board/LIMIFROG`) there is a script (`transfer.sh`) to transfer python code for the related board to the micropthon-USB-memory-stick. You can adapt the

script to you system by changing the TARGET variable which should contain the location of the mounted Limifrog-USB-memory-stick in your file system hierarchy. If the Limifrog-USB-drive is mounted this `transfer.sh`-script copies the sensor modules, general libraries and board support code to Limifrog.
Further there will be a USB-CDC device connected to you PC:

```
> ll /dev/ttyA*
crw-rw-rw- 1 root dialout 166, 0 Mai  7 20:03 /dev/ttyACM0
```

You can connect from the terminal (Read eval print loop (REPL) Mode) to that device (or in an application of your choice) and pushing the `Enter` key:

```
> screen /dev/ttyACM0 115200
Traceback (most recent call last):
  File "<stdin>", line 1
SyntaxError: invalid syntax
>>>
```

It may take a few seconds after the Limifrog reset till you can connect. There is some help information which you can get with the help() command:

```
>>> help()
...
...
>>>
```

Most important is the `CTRL-D` (soft reset) and the `CTRL-E` (paste mode) command. If you enter `CTRL-D` micropython will reload the code from the memory device, do a soft-reset and run the `main.py` script:

```
PYB: sync filesystems
PYB: soft reboot
Limifrog board:
vddh: OFF
buck: ON
ldo : OFF
lcd : OFF
high_current: ON
Magnetic field = (-1.267e-05, -5.323e-05, 6.004e-05) T
Pressure 9.524e+02 mBar, Height 519.5 m, Temperature = 27.5 C
Acceleration  (1.638e+02, -8.394e+01, -9.529e+02)
Rotation      (1.400e+03, -3.220e+03, -9.800e+02)
Temperature   20.81 C
Distance sensor:
  idModuleRevMajor: 1
  idModelRevMinor: 3
  idTime: 2829
  idModuleRevMinor: 2
  idModelRevMajor: 1
  idDate: 2306
  idModel: 180
Distance      16.0 mm
MicroPython v1.8-14-g57c81b3-dirty on 2016-05-08; LIMIFROG with STM32L476
Type "help()" for more information.
```

If you have a look at the `main.py` script, you will see that the sensors are just once read out and the measurement is returned on the command shell. After reading all sensors the script terminates.

If you'd like to have a repeated measurement of the distance try to add following code at the end of the `main.py` script or enter following code on the REPL by copy-paste (Using `CTRL-E`, copy/paste code to the terminal and then enter `CTRL-D`):

```
while True:
    print("Distance %.1f %s" % (dist.value(), dist.unit()))
    pyb.delay(100) # Wait 100 ms
```

If you set the `SEL` variable in the top of `transfer.sh` to `"2"` an example which uses the display is copied to Limifrog. This scripts reads out in regular interval the acceleration sensor and uses this measurement to control a dot on the screen in a while-forever-loop. You can terminate this loop by pressing `CTRL-C`.