Developing for *La BlueFrog* under Linux using gcc and Makefiles

A suitable gcc compiler is the *gcc-arm-none-eabi* suite. It is available on Launchpad, here: *https://launchpad.net/gcc-arm-embedded/+download*
You have the option to use Linux, Windows or OS X pre-built binaries.

To flash the STM32, one option under Linux is to use the ST-Link utility developed by 'Texane' available through GitHub : *https://github.com/texane/stlink*.
Or, under Windows, once you have an .hex or a .bin executable you can program the STM32 using ST's ST-Link utility. With it you can download your code into the STM32 program memory – and do a number of things like viewing its contents, comparing it against a bin reference file etc. It is freely available from ST under reference STSW-LINK004, here : *www.st.com/web/en/catalog/tools/PF258168*
The Makefile provided inside each *La BlueFrog* sample project invoke the arm-none-eabi toolchain (to build) and Texane's ST-Link utility under Linux (to run). The Makefile is written so that it can be used mostly as is for any project that follows the same file organization as the sample projects (refer to readme file in provided STM32-Projects-xxx package). If you're departing a little bit from this structure (say, adding some libraries), or want to change compilation options, etc., there should be just a few lines to modify in the Makefile.

If you wish to replicate this environment, here is a « journal » of the steps I followed to install under Linux (Mint) the toolchain that was used to develop the BlueFrog sample projects.

INSTALLING THE GCC COMPILER FOR ARM:

- downloaded the gcc-arm-none-eabi suite from Lanchpad, here:
  https://launchpad.net/gcc-arm-embedded/+download
  (pre-built binaries for Linux, Windows and OS X are available).

- chose to create a directory /ARM under the root on my PC and install there
  gcc-arm-none-eabi-<version number>

- therefore added /ARM to my search paths -- specifying that in ~/.bashrc in my case:
```
# For ARM toolchain
export PATH=$PATH:/ARM/gcc-arm-none-eabi-4_8-2014q2/bin/
```

- Being on 64-bit Linux, for compatibility with a 32-bit executable, installed ia32_libs :
```
> sudo apt-get install ia32-lib
```

- At this stage, was able to compile a test program.
  For example, run "make build" within one of the sample project folders provided with La BlueFrog -- e.g., .../projects/BlueFrogV1-BringUp0

INSTALLING THE STM32 PROGRAMMING SOFTWARE:

Using work done by 'Texane' available through GitHub.

- downloaded stlink_master.zip from *https://github.com/texane/stlink*
- unzipped it and installed it under /ARM too

There is an automatic build procedure using Autotools. For it to work as expected, had to do the following :
- installed autoconf and automake

```
> sudo apt-get install autoconf
> sudo apt-get install automake
```

- modified permissions of directory stlink_master and its contents (...that was rather sloppy work using 777 but you can be more subtle!)

```
> sudo chmod -R 777 stlink_master
```

- also installed the following :

```
> sudo apt-get install libc6-dev
> sudo apt-get install pkg-config
> sudo apt-get install libusb-1.0-0.dev
```

Then , could successfully conclude the build as directed on the README of Texane's GitHub page (github.com/texane/stlink) :

```
> ./autogen.sh
> ./configure
> make
```

Finally, to be able to actually program the STM32 :

- added the following search path into my ~/.bashrc :

```
# For ST-Link software
export PATH=$PATH:/ARM/stlink-master
```

and, as suggested on the GitHub page :

- installed st-util :

```
> make ./st-util
```

from directory /ARM/stlink-master
- copied file 49-stlinkv*.rules, provided in the stlink-master repository, to /etc/udev/rules.d :

```
> sudo cp 49-stlinkv2.rules /etc/udev/rules.d
```

- and ran :

```
> sudo udevadm control -reload-rules
> sudo udevadm trigger
```

At this point, was able to load through the STLink-V2 programmer a test program
For example, execute "make run" within one of the sample project folders provided with La BlueFrog -- e.g., .../projects/BlueFrogV1-BringUp0