NuttX RTOS

ARM Development

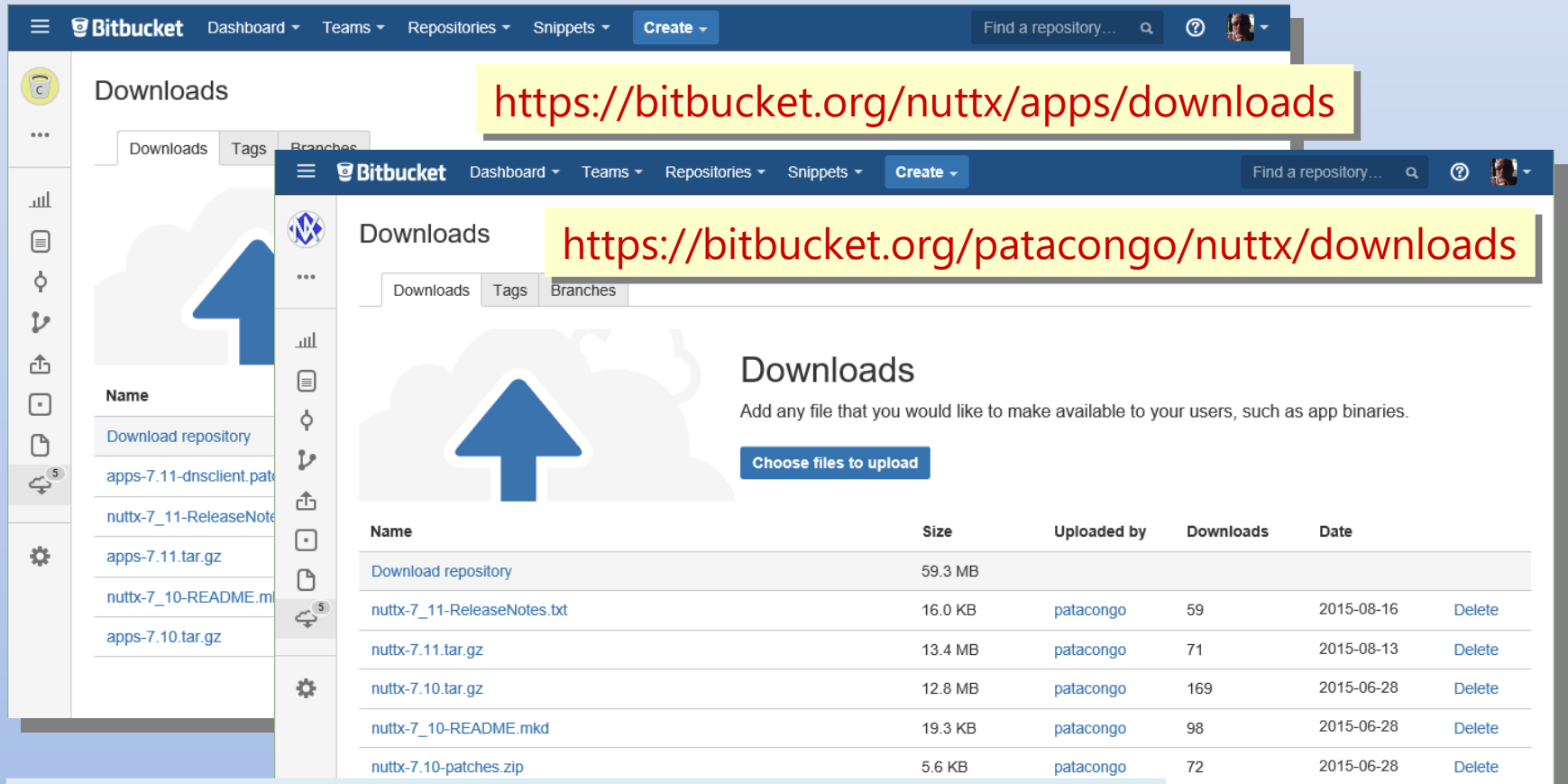
Gregory Nutt

Overview

- Getting the Source Code
- Configuring NuttX
- The NuttX Simulator
- The NuttShell
- ARM Build Tools
- ARM Cross Development
- Build Modes
- JTAG
- Debugging the ARM Target

Getting the Source Code

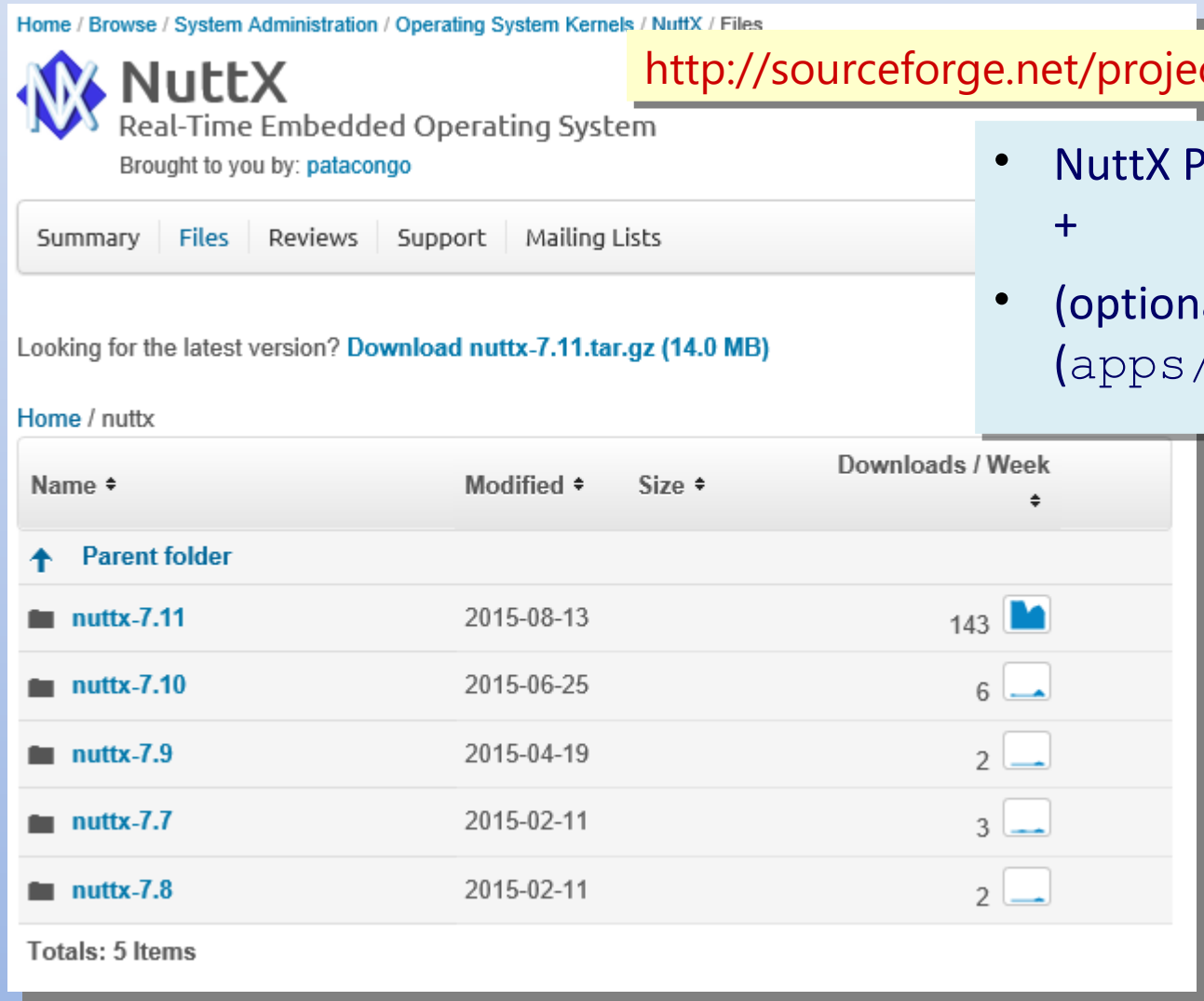
Frozen Versions



- NuttX Package (Core OS) +
- (optional) Application (`apps/`) package

Getting the Source Code

Frozen Versions (Cont'd)

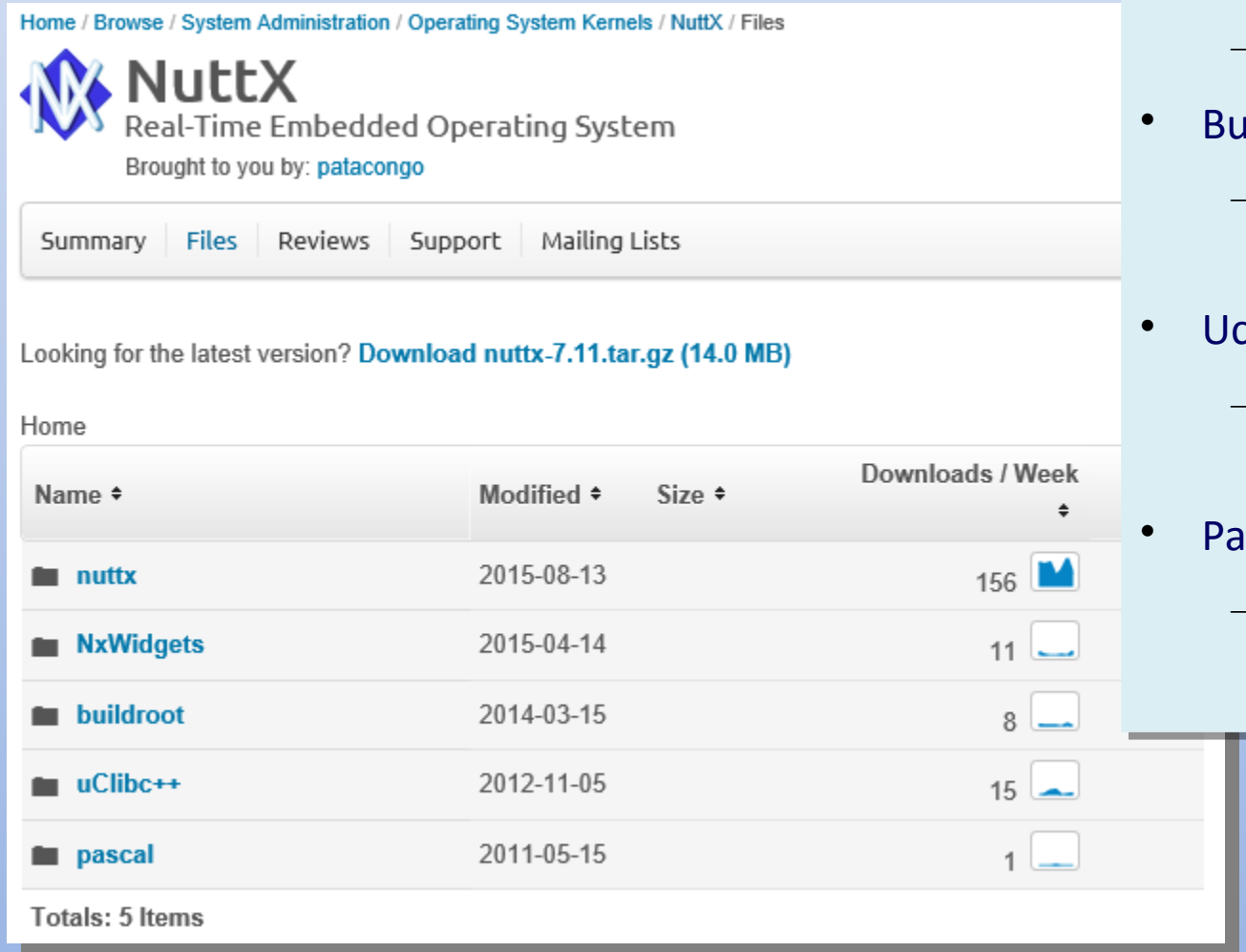


http://sourceforge.net/projects/nuttx/files/nuttx

- NuttX Package (Core OS)
  +
- (optional) Application (`apps/`) package

Getting the Source Code
Other Frozen Packages

http://sourceforge.net/projects/nuttx/files

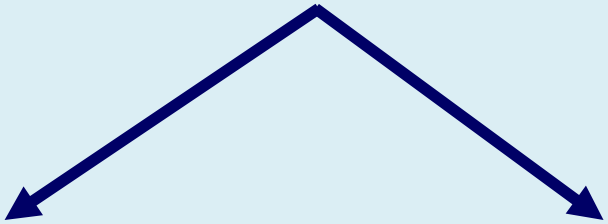


- NxWidgets:
  - Graphics Package
- Buildroot:
  - NuttX customized toolchain
- Uclibc++:
  - Standard C++ Libarary
- Pascal:
  - Pascal p-code compiler

Getting the Source Code
Creating a Build Directory

- Download `nuttx-`*x.y*`.tar.gz` and `apps-`*x.y*`.tar.gz`
- Unpack

```
tar zxf nuttx-x.y.tar.gz
tar zxf apps-x.y.tar.gz
```

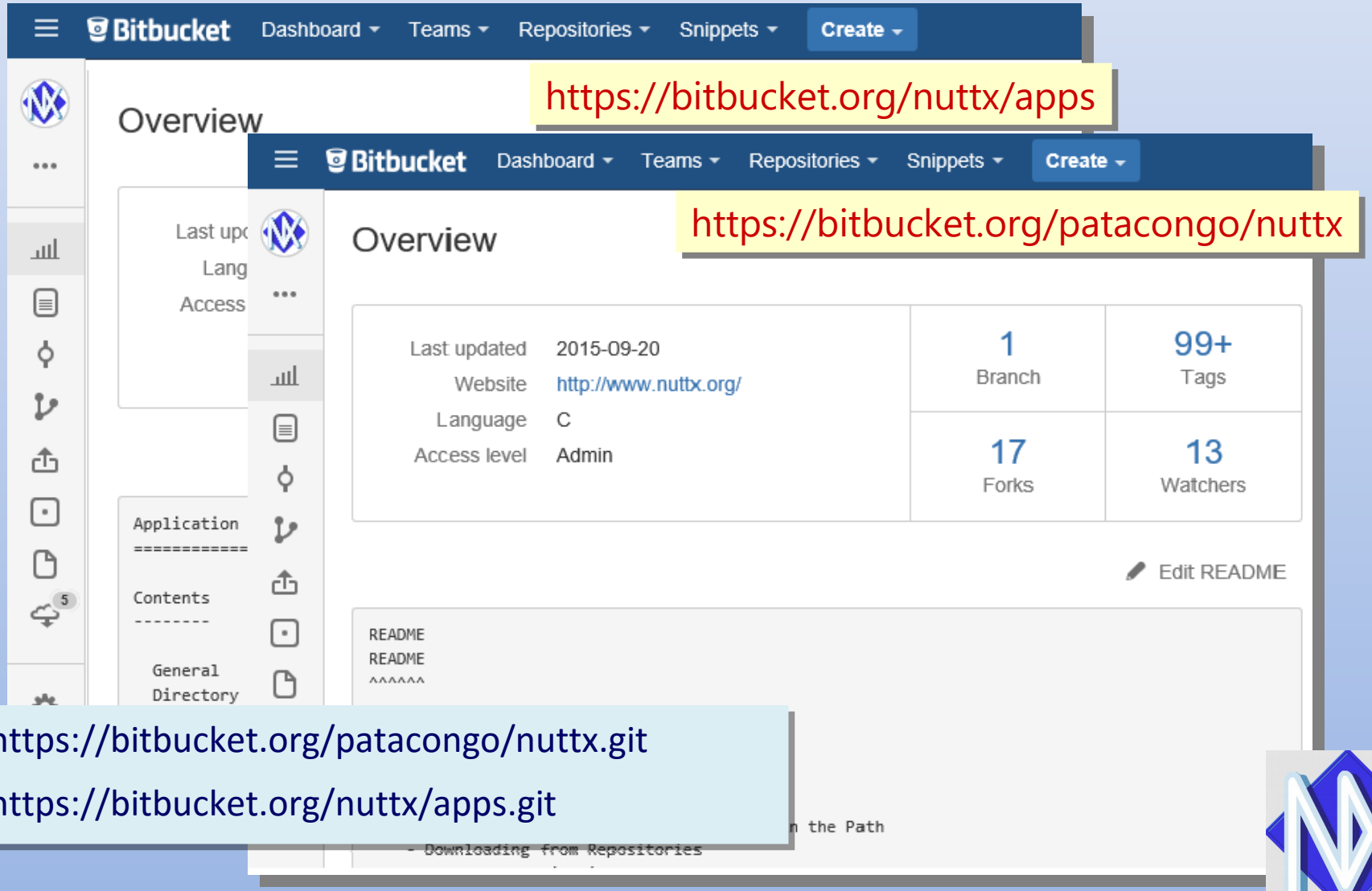


`nuttx-`*x.y* `apps-`*x.y*

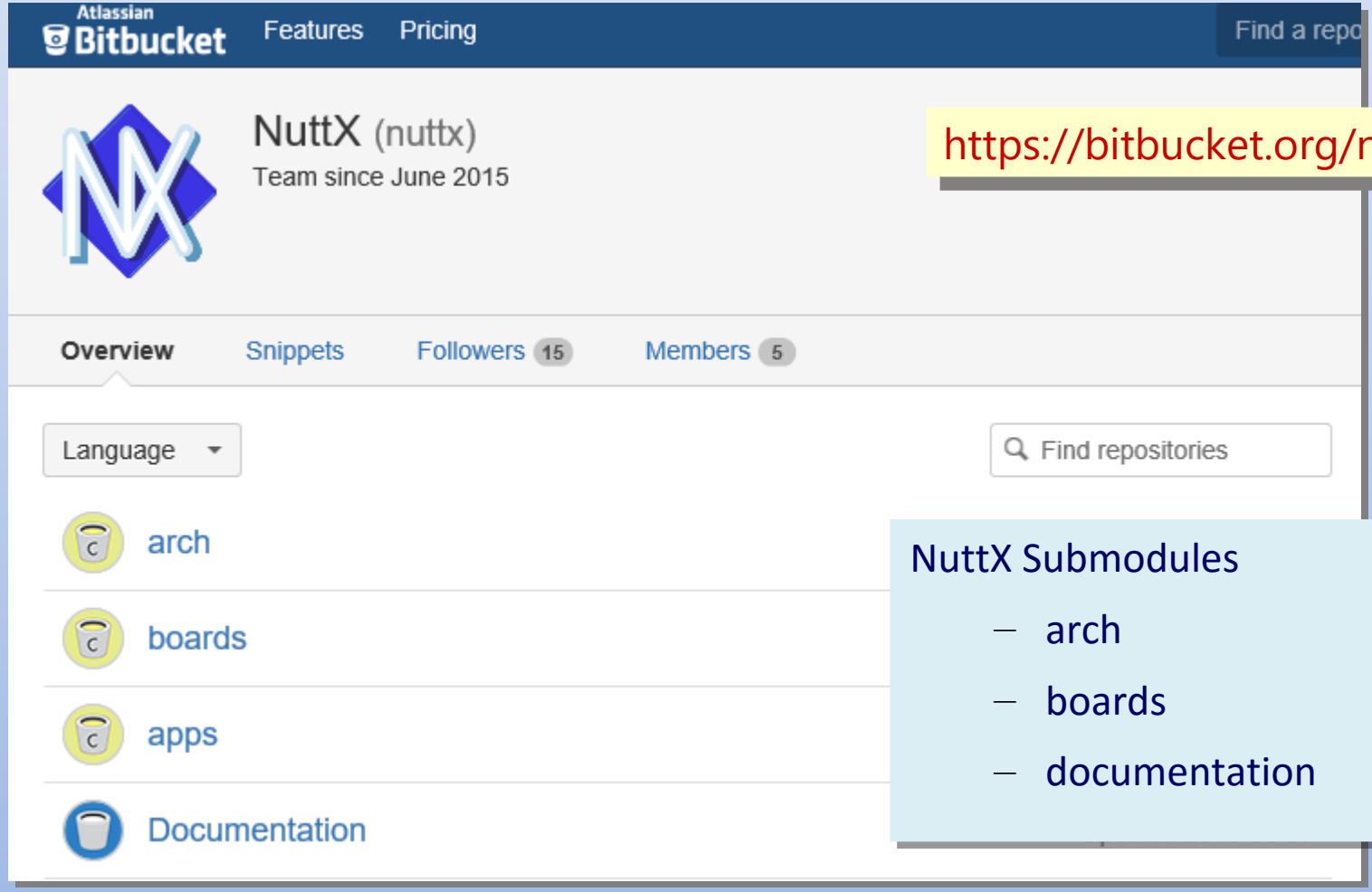
Where *x.y* is a frozen version number like 7.11

Getting the Source Code

GIT Repositories



Getting the Source Code
Other Repositories



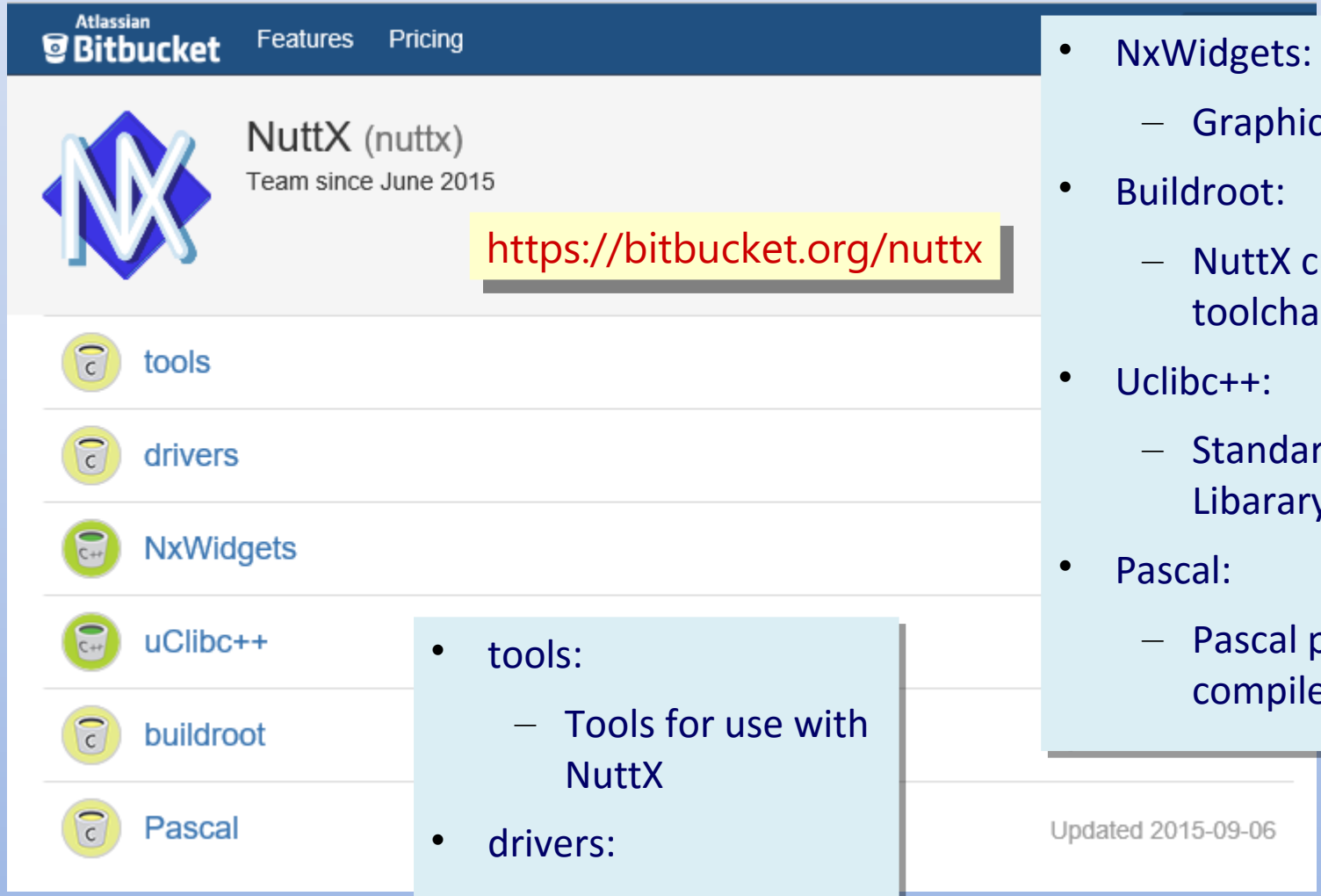
https://bitbucket.org/nuttx

NuttX Submodules

- arch
- boards
- documentation

Getting the Source Code

Other Repositories (Cont'd)



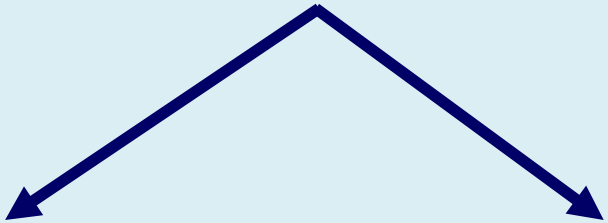
- tools:
  - Tools for use with NuttX
- drivers:
  - GPL drivers

- NxWidgets:
  - Graphics Package
- Buildroot:
  - NuttX customized toolchain
- Uclibc++:
  - Standard C++ Libarary
- Pascal:
  - Pascal p-code compiler

Getting the Source Code
Cloning a Build Directory

Clone Repositories

```
git clone https://bitbucket.org/patacongo/nuttx.git nuttx
git clone https://bitbucket.org/nuttx/apps.git apps
```



Initialize GIT Sub-modules

```
cd nuttx
git submodule init
git submodule update
```

More Info:

http://www.nuttx.org/doku.php?id=downloads

Configuring NuttX

Building kconfig-frontends

```
./configure --enable-mconf --disable-nconf --disable-gconf
--disable-qconf

LD_RUN_PATH=/usr/local/lib

make

make install
```

May need root privileges to install

Graphical configuration Tools

Qt:

```
./configure … --enable-qconf
```

GTK:

```
./configure ... --enable-gconf ...
```

Configuring NuttX

Board Support Logic in sub-directories of `configs/`

Form: `configs/`*<board-name>*

Example: `configs/stm32f4discovery`

```
cd tools/
./configure.sh <code-name>/<configuration>
```

Example: `./configure.sh stm32f4discovery/nsh`

Equivalent Manual Configuration:

```
cp stm32f4discovery/nsh/defconfig .config
cp stm32f4discovery/nsh/Make.defs Make.defs
cp stm32f4discovery/nsh/setenv.sh setenv.sh
```

Additional Pre-Build Steps

Modify configuration for build environment:

```
make menuconfig
```

- Build Host: Linux, Windows, OSX, etc.
- Windows Build Framework: Cygwin, MSYS, Native
- Toolchain

Make sure configuration is up to date:

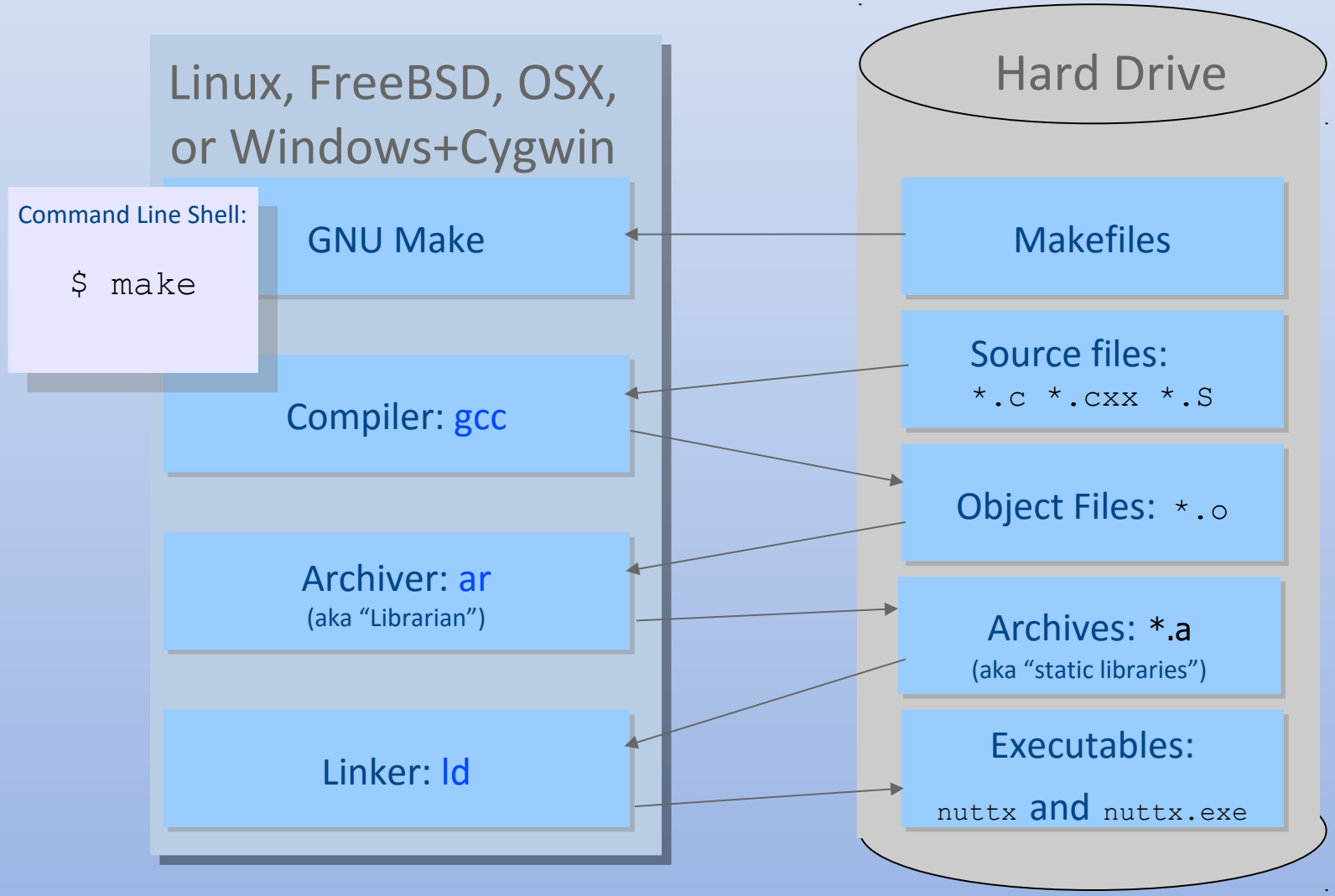
```
make oldconfig
```

Establish PATH to build tool binaries:

```
. ./setenv.sh
```

Optional
Needs to be customized

Building Executables



Building the NuttX Simulator

Building the NuttX Simulator

```
$ cd nuttx/tools
$ ./configure.sh sim/nsh
$ cd ..
$ make menuconfig
$ . ./setenv.sh
$ make
```

Example

Linux->Windows

Cygwin

32/64 bit

Microsoft ABI

Refresh Configuration
Configuration build environment

Optional
Need path to build tool binaries

Running the NuttX Simulator

```
$ ./nuttx.exe
```

The NuttShell (NSH)

A “thin” program to interface with the OS

Many commands similar to the bash shell

```
nsh> help
```

Debugging the Simulator

1 Reconfigure

- Enable debug symbols in the configuration
- Disable optimization (optional)

2 Rebuild

3 `gdb nuttx.exe`

Using ddd graphical front end:

- `export DISPLAY=:0`
- `ddd nuttx.exe &`

ARM Build Tools

Linux / OSX

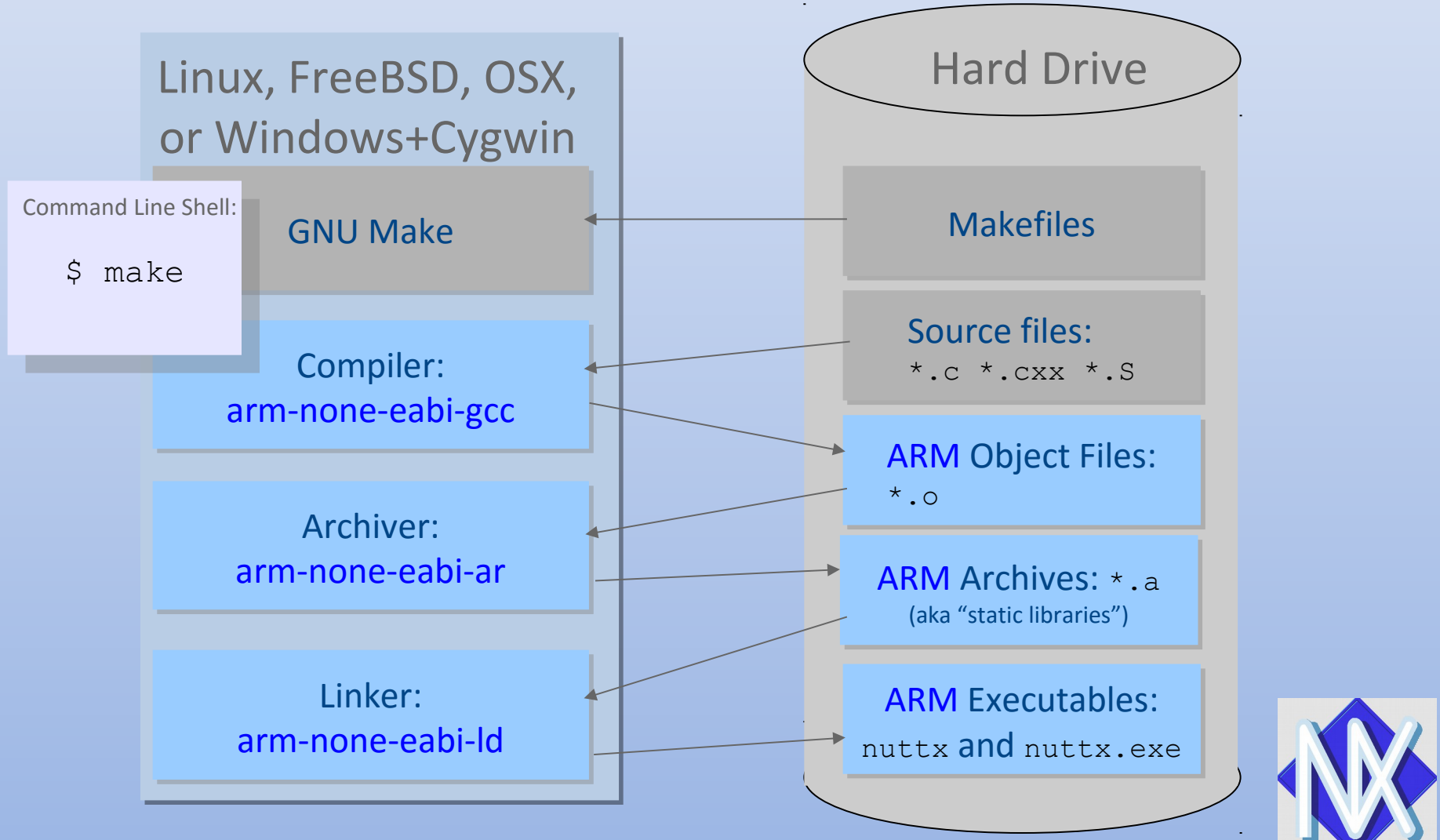
- *GCC Toolchain*: arm-none-eabi-gcc
  - https://launchpad.net/gcc-arm-embedded
- *Configuration Tool*: kconfig-frontends
  Same configuration tool used with Linux kernel
  - http://ymorin.is-a-geek.org/projects/kconfig-frontends
  - https://bitbucket.org/nuttx/tools
  - https://bitbucket.org/nuttx/buildroot
- *Other Tools*: genromfs
  - http://romfs.sourceforge.net/
  - https://bitbucket.org/nuttx/tools

ARM Build Tools

Windows

- Cygwin
  - http://cygwin.com/
  - Creates POSIX development environment on Windows
  - Use same Linux tools (built for Cygwin)
  - Can also integrate Native Windows Tools
- MSYS is another option
- Native Windows Build is also possible

ARM Cross Development

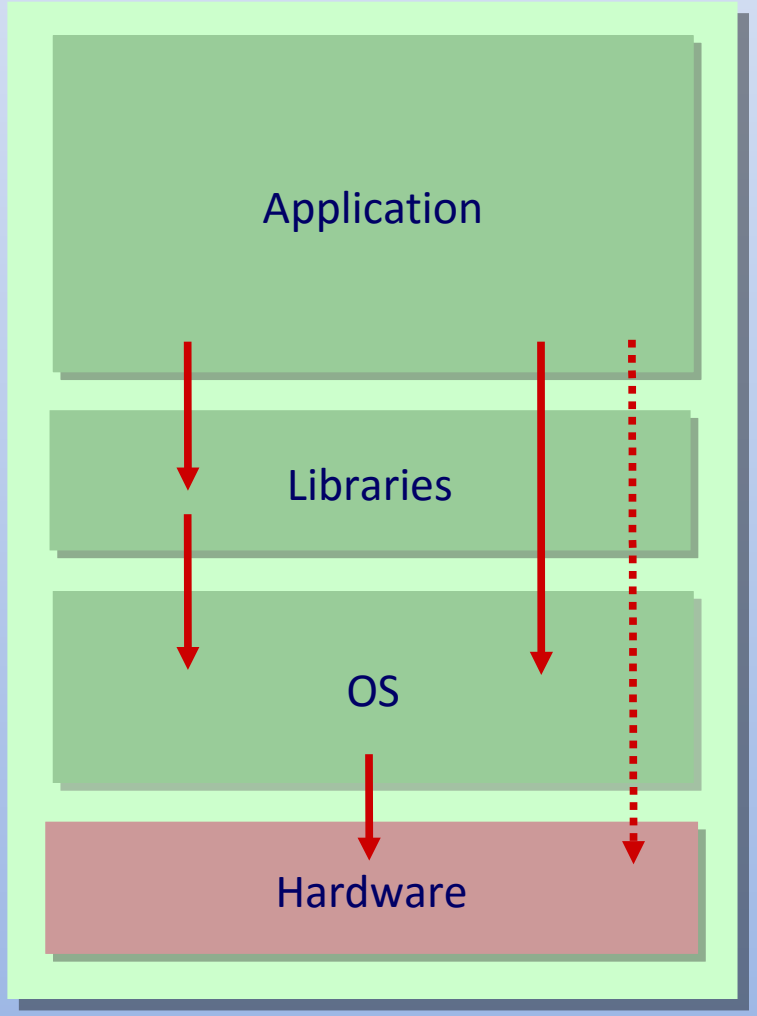


make export

Will extract all start-up files, libraries, and header files into a package that can be then be used in another build environment. Example, with an IDE

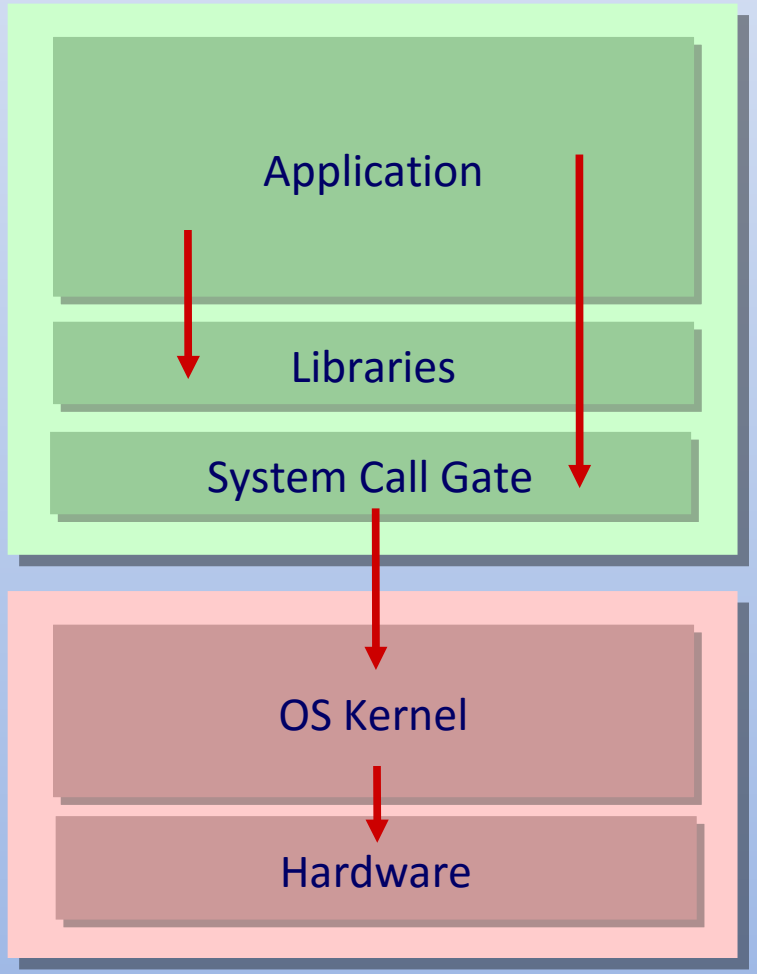
```
make export
```

Linked Binary
(*Flat* Build Mode)



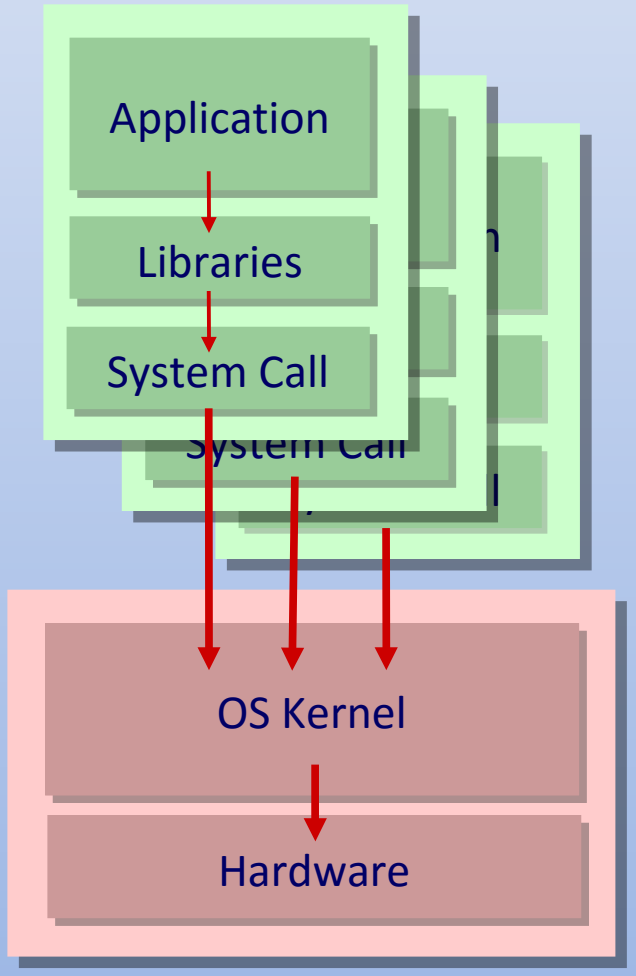
- *Flat* address space: All memory accessible by the application.
- Nothing is protected.
- Application can directly access OS/hardware resources
- No special hardware required
- `CONFIG_BUILD_FLAT=y`

Linked Binaries
(*Protected* Build Mode)



- OS resources protected from application.
- Application accesses OS resources indirectly through a *call gate*
- Require Memory Protection Unit (MPU)
- Example, Cortex-M
- `CONFIG_BUILD_PROTECTED=y`

Linked Binaries
(*Kernel* Build Mode)



- *Processes*: Each application protected in a separate address space.
- Require Memory Management Unit (MMU)
- Example, Cortex-A
- `CONFIG_BUILD_KERNEL=y`

JTAG/SWD

IC Debug Ports

Debuggers communicate through JTAG or SWD

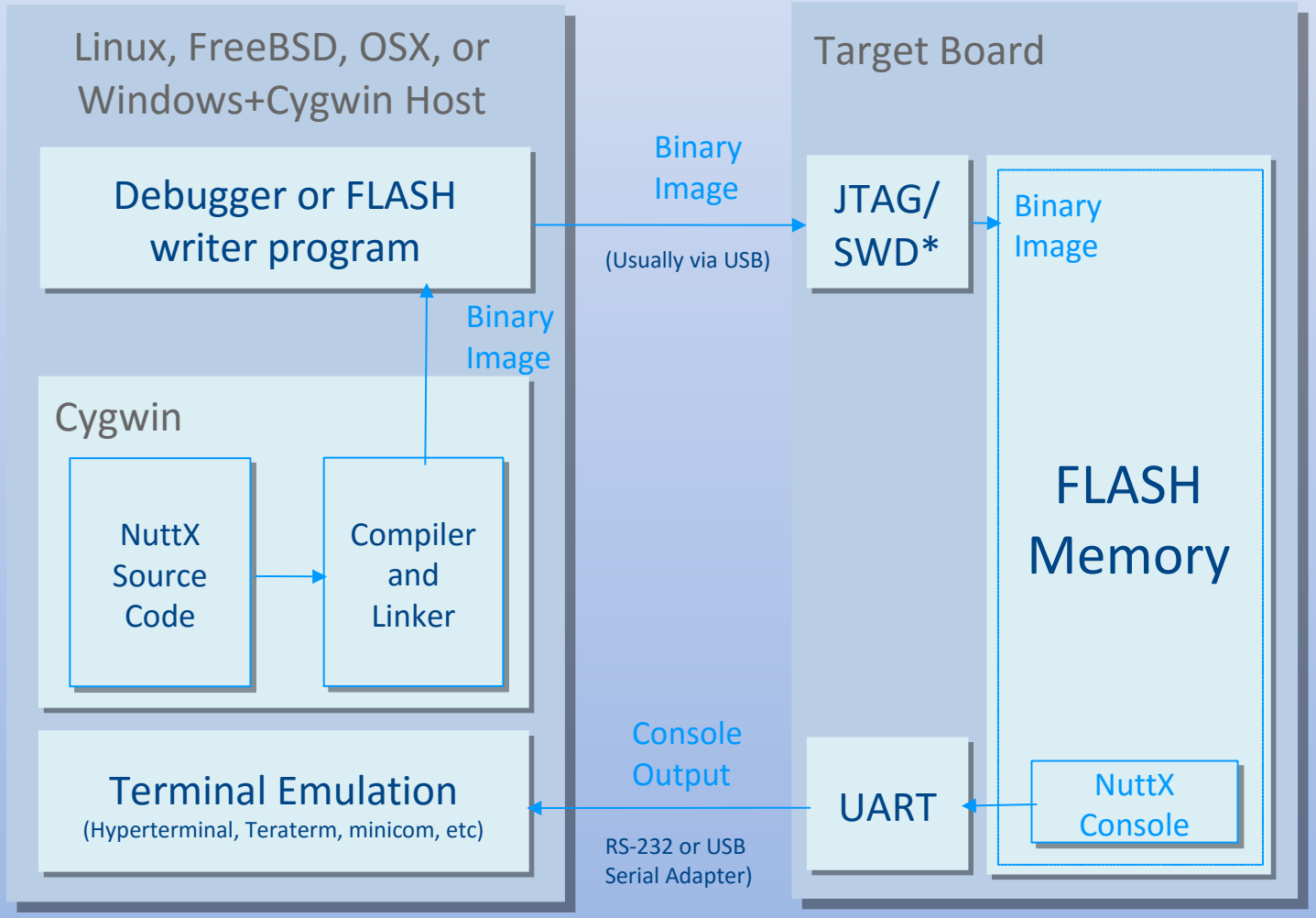
Load code into FLASH

Debug code (breakpoints, single stepping, etc.)

Joint Test Action Group (JTAG) – IEEE 1149.1

Serial Wire Debug (SWD) – ARM

Hardware Connections



*Not usually on the board.

Debugging the ARM Target

1 Reconfigure

- Enable debug symbols in the configuration
- Disable optimization (optional)

2 Rebuild

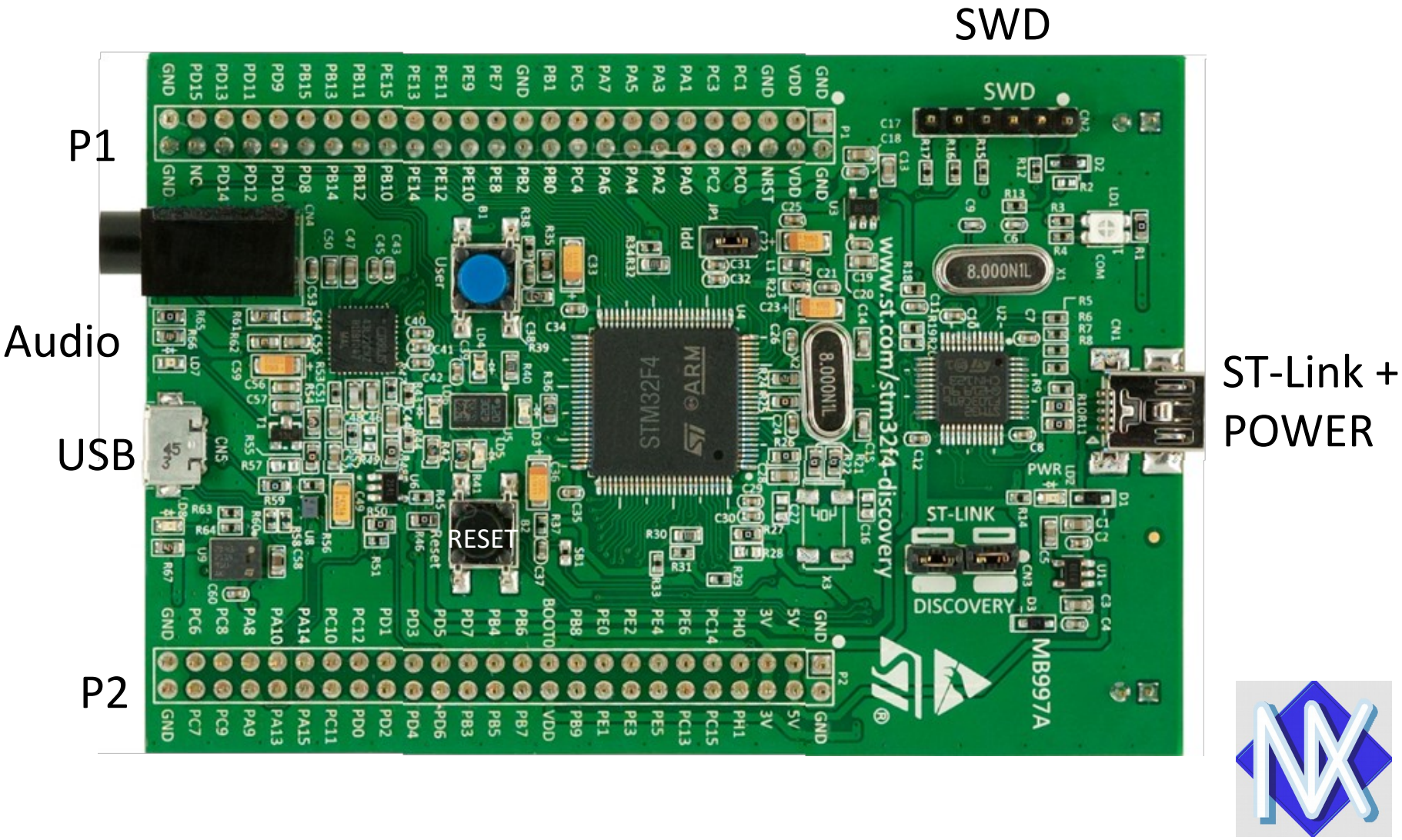
3 Start the GDB server (OpenOCD, Segger J-Link, others)

4 `arm-none-eabi-gdb nuttx.exe`

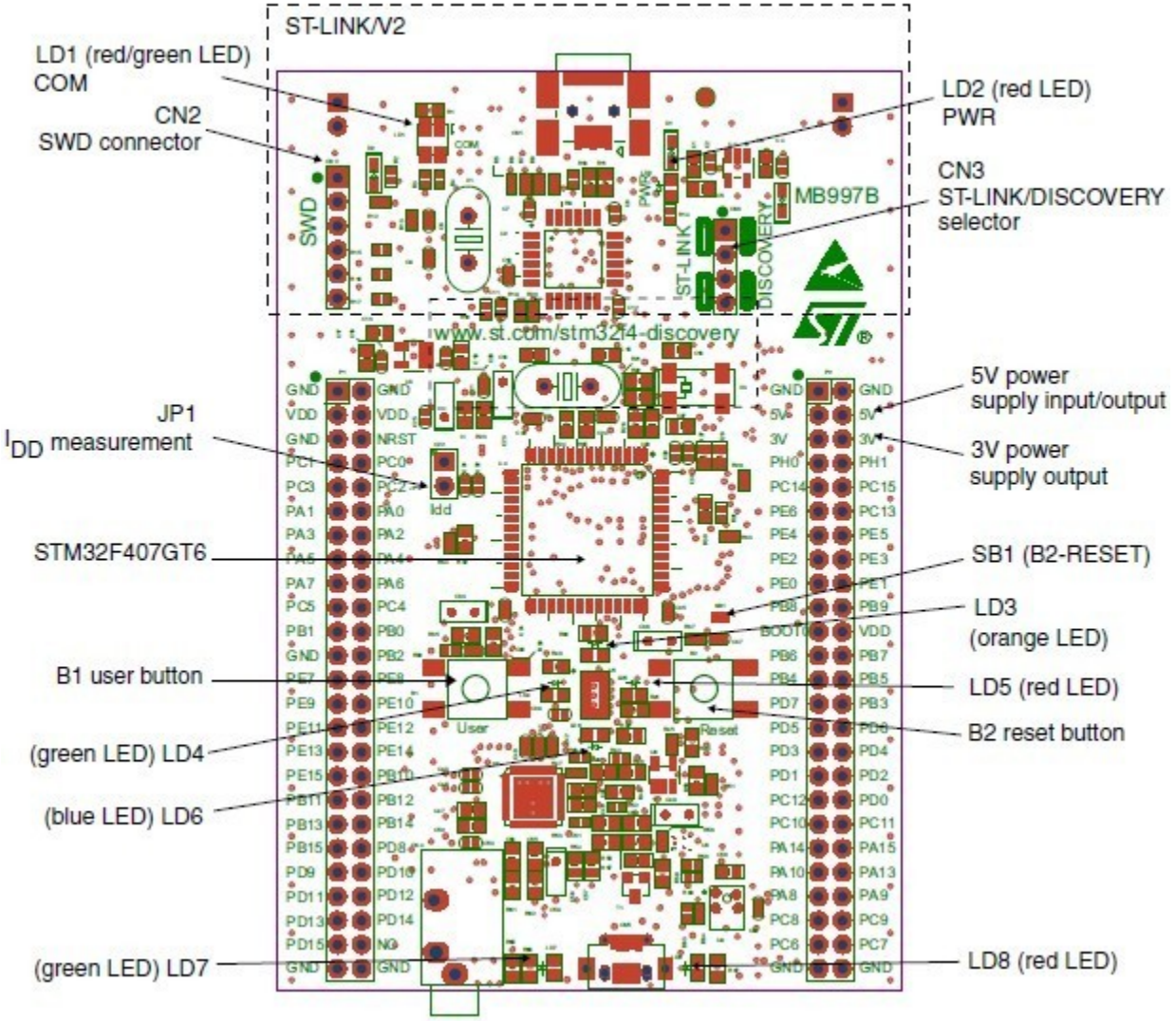
Using ddd graphical front end:

- `export DISPLAY=:0`
- `ddd --debugger arm-none-eabi-gdb nuttx &`

The STM32F4Discovery



The STMicro STM32F4-Discovery



Atmel SAMV71 Xplained Ultra