embedded sdr

working with sdks
embedded sdr
who here has worked with sdks?
if you develop for embedded devices, you should
why would you care?
comfort
work in your environment
speed
workstation
vs. target
clean (almost) no leftovers on target
exceptions ...
if your ghetto hack consists of one .c file and compiles in under a second ...
… or you are getting ready to release your product …
the last one was kinda important ... !!!

DO NOT SHIP STUFF YOU COMPILED WITH AN SDK really DON'T
contents of an sdk: compilers, headers, libs, native tools & env scripts
setting up your environment

$ cd $SDK_PATH
$ source environment-setup-{archspecific}

in our case:
$ source environment-setup-armv7ahf-neon-oe-linux gnueabi
easy to check using $CC env variable

$ $CC --version
arm-oe-linux-gnueabi-gcc (GCC) 4.9.2
[...]

$ cat hello_fosdem.c

#include <stdio.h>
int main(int argc, char *argv[])
{
    printf("hello_fosdem!\n");
    return 0;
}

$ make hello_fosdem
wasn’t all that bad, right?
cross compiling with autotools

$ ./configure \\
--host arm-oe-linux \\
--prefix=/usr

$ make
ok ... 
admittedly 
that was a 
best case 
scenario ...
cross compiling with cmake

$ cmake -
  DCMAKE_TOOLCHAIN_FILE=<toolchain file*> \
  -
  DCMAKE_INSTALL_PREFIX=/usr/<yoursource>

$ make

*for uhd/gnuradio: cmake/Toolchains/oe-sdk_cross. cmake
alright ... we kind know how to build stuff... how do we run it?
well, you could scp it to the target ... or ...
sshfs
for development

$ mkdir mnt
$ sshfs -o allow_root user@host: //<fullpath> mnt

$ export
LD_LIBRARY_PATH=mnt/<app dependent>

$ export
PATH=mnt/<app_dependent>:PATH
suggestion: for simplicity drop it in a script
pain point
dependencies
prominent example:
uhd vs. gnuradio vs. gr-ettus
one (easy) ‘solution’:
install into sdk ...
… however …
this taints your sdk …
... and makes kitties sad ...
hack:
use staging install dir, point dependencies there

$make install DESTDIR=<yourstaging>
install to staging dir with cmake

$ make
$ make DESTDIR=~/my_staging install

again, sshfs & env mods necessary
downside: tedious, cmake sometimes needs to be clubbed to happiness
yocto’s extensible sdks do somewhat address this … see paul eggleton’s talk at elc2015
but: developer needs to learn about yocto tools ...
... but doing this he saves time when he later packages stuff for release ...
... because if he doesn't it's just kittehz will be even sadder ...
$ diff

who will now try sdks?
now go and hack
some
#cyberspectrum...

... or ask questions