



life.augmented

Distribution package hands on

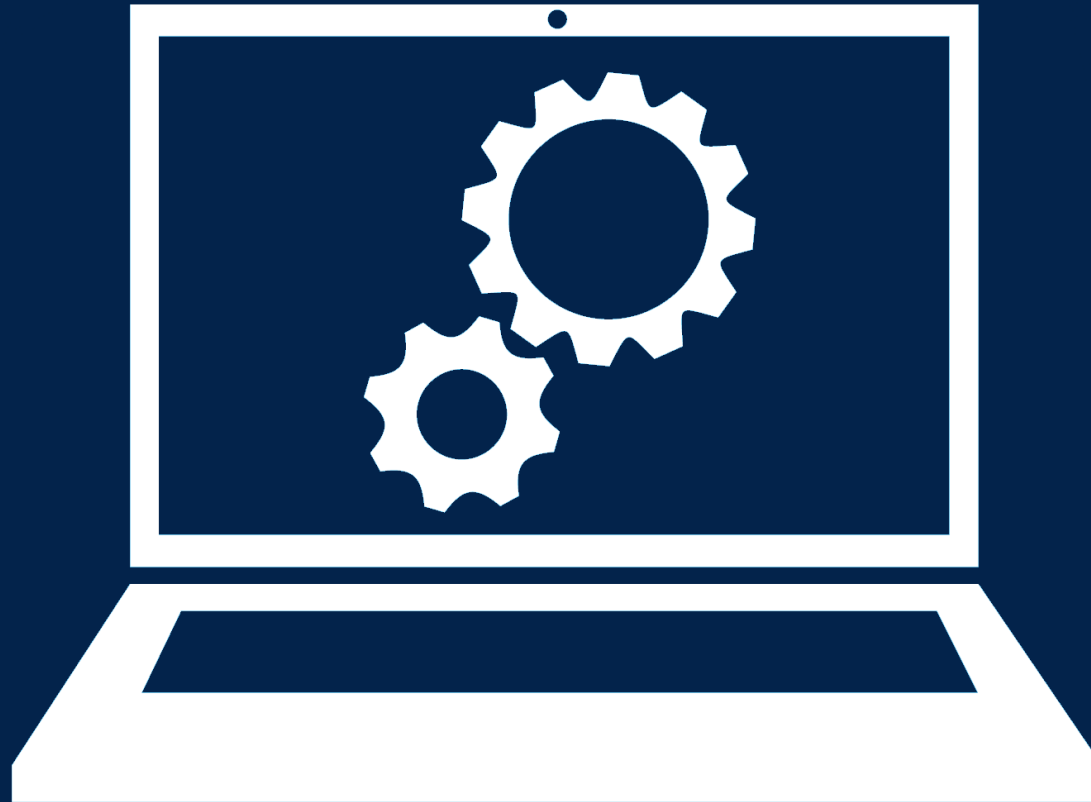
Cindy Ge

March 2021

Agenda

- 1 Prerequisites
- 2 Bitbake Workflow
- 3 SDK Generation
- 4 Integrate User Application

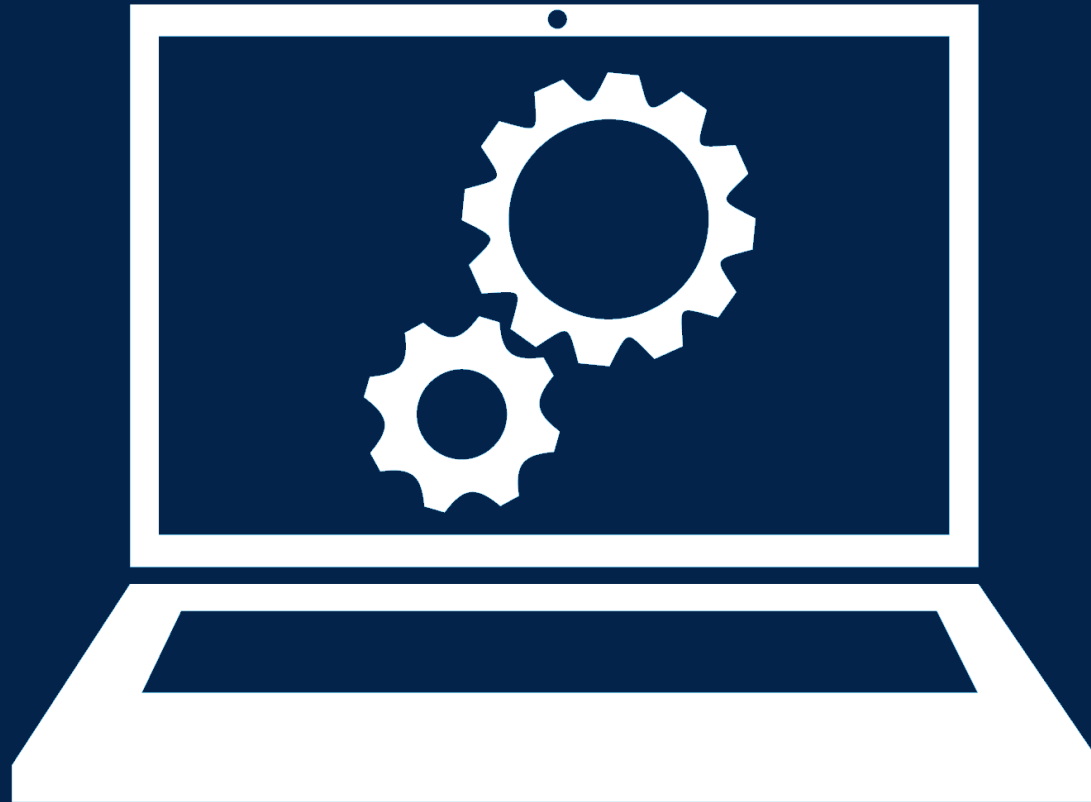
Prerequisites



Prerequisites

- DK1/2 board boot successfully with same release of STM32MPU Embedded Software distribution.
- Development PC (Virtual Machine)
- OpenSTLinux distribution package has been installed

Bitbake Workflow



In the Distribution Package classroom, bitbake workflow has been introduced, for deeper understanding, this class will take the linux-stm32mp recipe compilation as an example to show the detail step by step.

Before starting, let's have a look at:

- Linux kernel working directory:

```
/work/distribution/build-openstlinuxweston-stm32mp1/tmp-glibc/work/stm32mp1-ostl-linux-gnueabi/linux-stm32mp
```

- Linux kernel source code directory:

```
/work/distribution/build-openstlinuxweston-stm32mp1/tmp-glibc/work-shared/stm32mp1/kernel-source/
```

- Cleaning the kernel:

```
PC $> bitbake -c clean linux-stm32mp
```

- List all the task defined in linux-stm32mp recipe:

```
PC $> bitbake linux-stm32mp -c listtasks
```

Source Fetching

The first stages of building a recipe are to fetch and unpack the source code:

```
PC $> bitbake linux-stm32mp -c fetch
```

```
PC $> bitbake linux-stm32mp -c unpack
```

Then check the source and work directories:

```
PC $> ls -l tmp-glibc/work/stm32mp1-ostl-linux-gnueabi/linux-stm32mp/5.4.56-r0/  
total 32  
drwxr-xr-x 3 xxx xxx 4096 Mar 11 22:40 5.4  
drwxr-xr-x 2 xxx xxx 4096 Mar 11 22:40 build  
drwxr-xr-x 3 xxx xxx 4096 Mar 11 22:40 fragments  
lrwxrwxrwx 1 xxx xxx 174 Mar 11 22:40 linux-5.4.56 -> /local/home/ xxx/openstlinux-5.4-dunfell-mp1-20-11-  
12/build-openstlinuxweston-stm32mp1/tmp-glibc/work-shared/stm32mp1/kernel-source  
drwxrwxr-x 2 xxx xxx 4096 Mar 11 22:40 recipe-sysroot  
drwxrwxr-x 7 xxx xxx 4096 Mar 11 22:40 recipe-sysroot-native  
drwxr-xr-x 2 xxx xxx 4096 Mar 11 22:40 source-date-epoch  
drwxrwxr-x 2 xxx xxx 4096 Mar 11 22:40 temp
```

Once source code is fetched and unpacked, BitBake locates patch files and applies them to the source files:

```
PC $> ls tmp-glibc/work-shared/stm32mp1/kernel-source/drivers/usb/typec/  
altmodes bus.h Kconfig mux tcpm ucsi  
bus.c class.c Makefile mux.c tps6598x.c
```

```
PC $> bitbake linux-stm32mp -c patch
```

Then check if the do_patch worked:

```
PC $> ls tmp-glibc/work-shared/stm32mp1/kernel-source/drivers/usb/typec/  
altmodes bus.h Kconfig mux tcpm typec_stusb.c  
bus.c class.c Makefile mux.c tps6598x.c ucsi
```

Configuration, Compilation, and Staging

After source code is patched, BitBake executes tasks that configure and compile the source code. Once compilation occurs, the files are copied to a holding area (staged) in preparation for packaging:

```
PC $> bitbake linux-stm32mp -c configure
```

Then check the build directory:

```
PC $> ls -al tmp-glibc/work/stm32mp1-ostl-linux-gnueabi/linux-stm32mp/5.4.56-r0/build/
```

Try to run:

```
PC $> bitbake linux-stm32mp -c menuconfig
```

to see what happened, this task is special for kernel.

Configuration, Compilation, and Staging

Compile the recipe

```
PC $> bitbake linux-stm32mp -c compile
```

Then check the build directory:

```
PC $> ls -al tmp-glibc/work/stm32mp1-ostl-linux-gnueabi/linux-stm32mp/5.4.56-r0/build/
```

The kernel images and dtbs were created.

Then run:

```
PC $> bitbake linux-stm32mp -c install
```

From the log, we can see the task `do_compile_kernelmodules` will be done first.

After `do_install`, the image folder will be created with the images and binary:

```
PC $> ls -al tmp-glibc/work/stm32mp1-ostl-linux-gnueabi/linux-stm32mp/5.4.56-r0/image
```

Package Splitting

After source code is configured, compiled, and staged, the build system analyzes the results and splits the output into packages:

```
PC $> bitbake linux-stm32mp -c package
```

The `do_package` task create two directories in the work directory: `package/` and `packages-split/`, check the detail:

```
PC $> ls -al tmp-glibc/work/stm32mp1-ostl-linux-gnueabi/linux-stm32mp/5.4.56-r0/package
```

```
PC $> ls -al tmp-glibc/work/stm32mp1-ostl-linux-gnueabi/linux-stm32mp/5.4.56-r0/packages-split
```

packages-split/kernel-imagebootfs will be used by bootfs generation.

Image Generation

Once packages are split and stored in the Package Feeds area, the build system uses BitBake to generate the root filesystem image , for linux-stm32mp recipe, the bootfs image should be compiled firstly:

```
PC $> bitbake -C compile st-image-bootfs
```

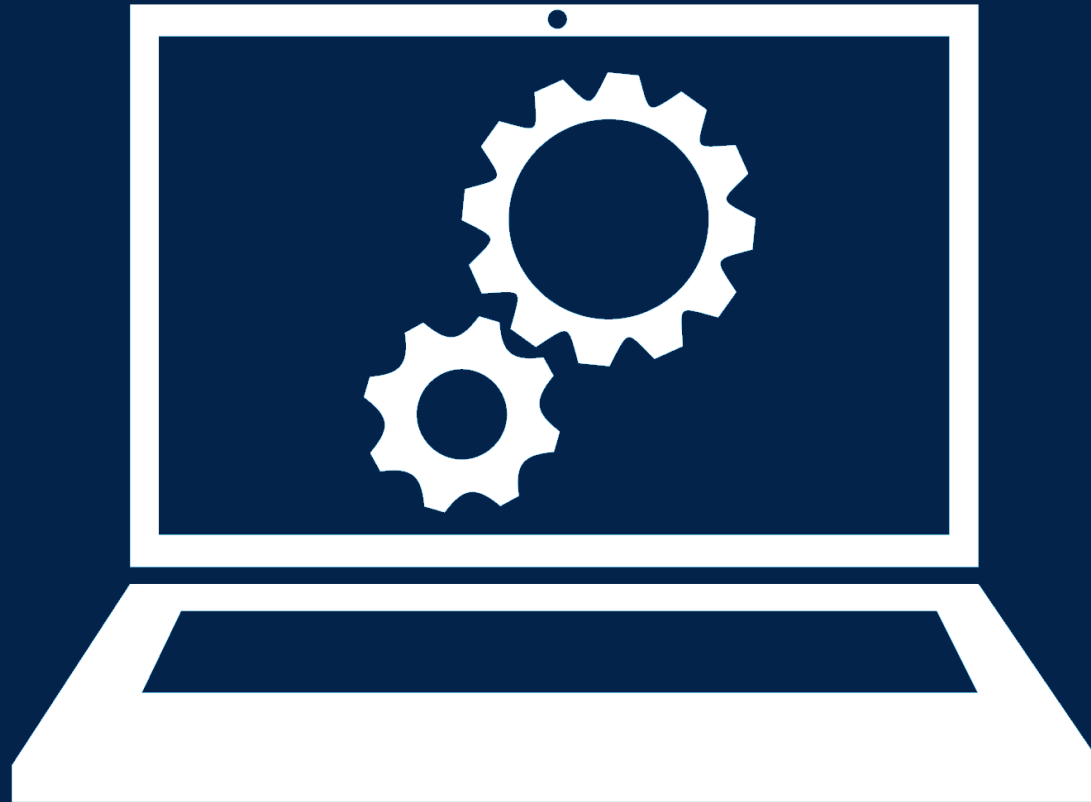
Check the bootfs image has been updated:

```
PC $> ls -l tmp-glibc/deploy/images/stm32mp1/st-image-bootfs-openstlinux-weston-stm32mp1*
```

Generate the root filesystem image:

```
PC $> bitbake st-image-weston
```

SDK Generation



Prerequisites

- The Distribution Package relative to your STM32 microprocessor Series is installed.

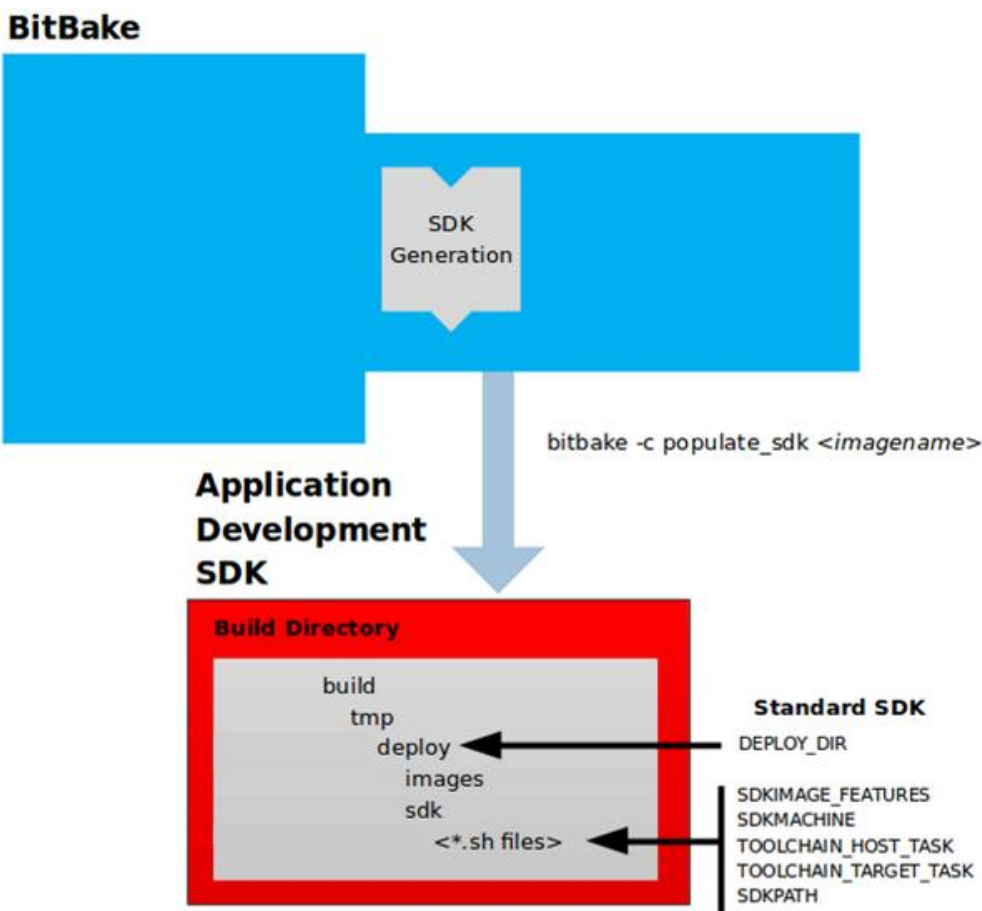
On the installation:

- some pieces of software might have been modified or integrated
- the build environment script has been executed
- the selected image has been rebuilt

SDK Generation

The *do_populate_sdk* task helps to create the standard SDK and handles two parts: a target part and a host part.

The target part is built for the target hardware and includes libraries and headers. The host part is the part of the SDK that runs on the host machine.



SDK Generation

- Check that the build environment script has been executed, and that the current directory is the build directory of the OpenSTLinux distribution (for example, *openstlinux-20-06-24/build-openstlinuxweston-stm32mp1*)

```
PC $> DISTRO=openstlinux-weston MACHINE=stm32mp1 source layers/meta-st/scripts/envsetup.sh
```

- Generate the SDK installation files (including the installation script) for a standard SDK with the following command :

```
PC: $> bitbake -c populate_sdk <image>
```

Example:

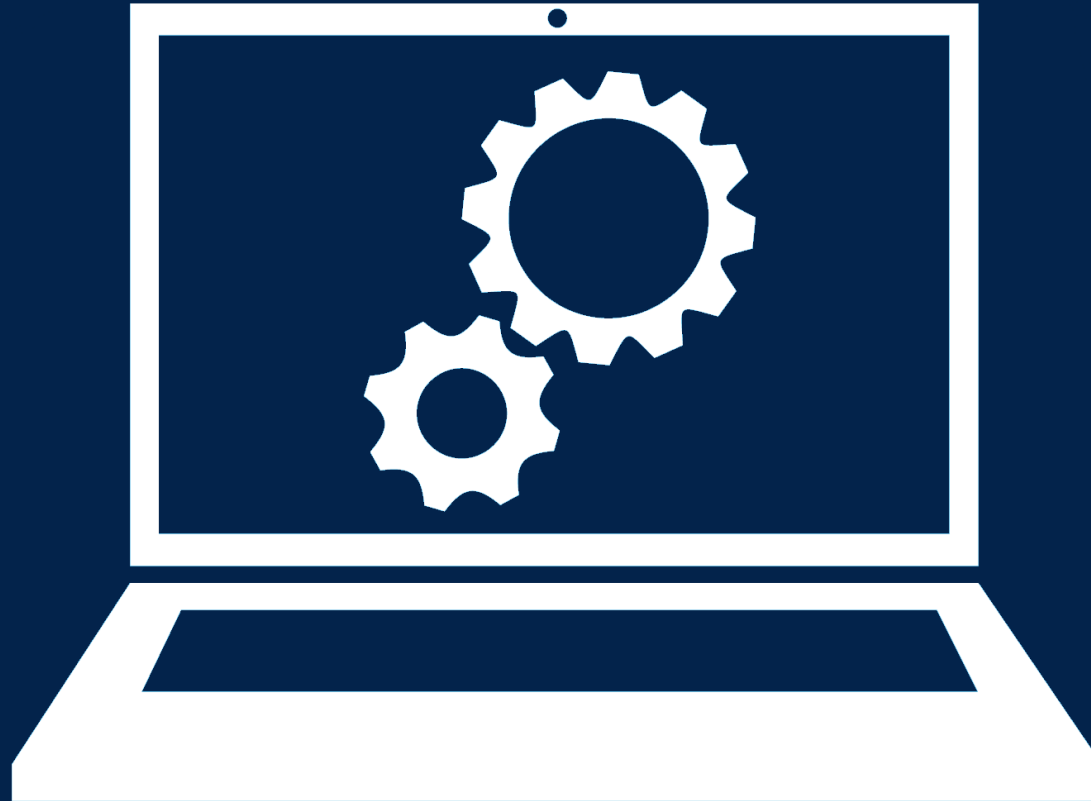
```
PC $> bitbake -c populate_sdk st-image-weston
```

SDK Generation

- The SDK installation files (*<image>-<distro>-<machine>-<host machine>-toolchain-<Yocto release>-snapshot.**) are written to the *deploy/sdk* directory inside the build directory.
 - *<host machine>*: Host machine on which SDK is generated, x86_64 (64-bit host machine; this is the only supported value).
 - *<Yocto release>*: Release number of the Yocto Project; For example, 3.1 (aka dunfell).

```
PC $> ls tmp-glibc/deploy/sdk/  
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot.host.manifest  
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot.license  
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot-license_content.html  
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot.sh  
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot.target.manifest  
st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot.testdata.json
```

Integrate User Application



- Story

As a developer, you are asked to integrate a new application or a new library on the Yocto Project, the source code application maybe located in various areas.

- How to do?

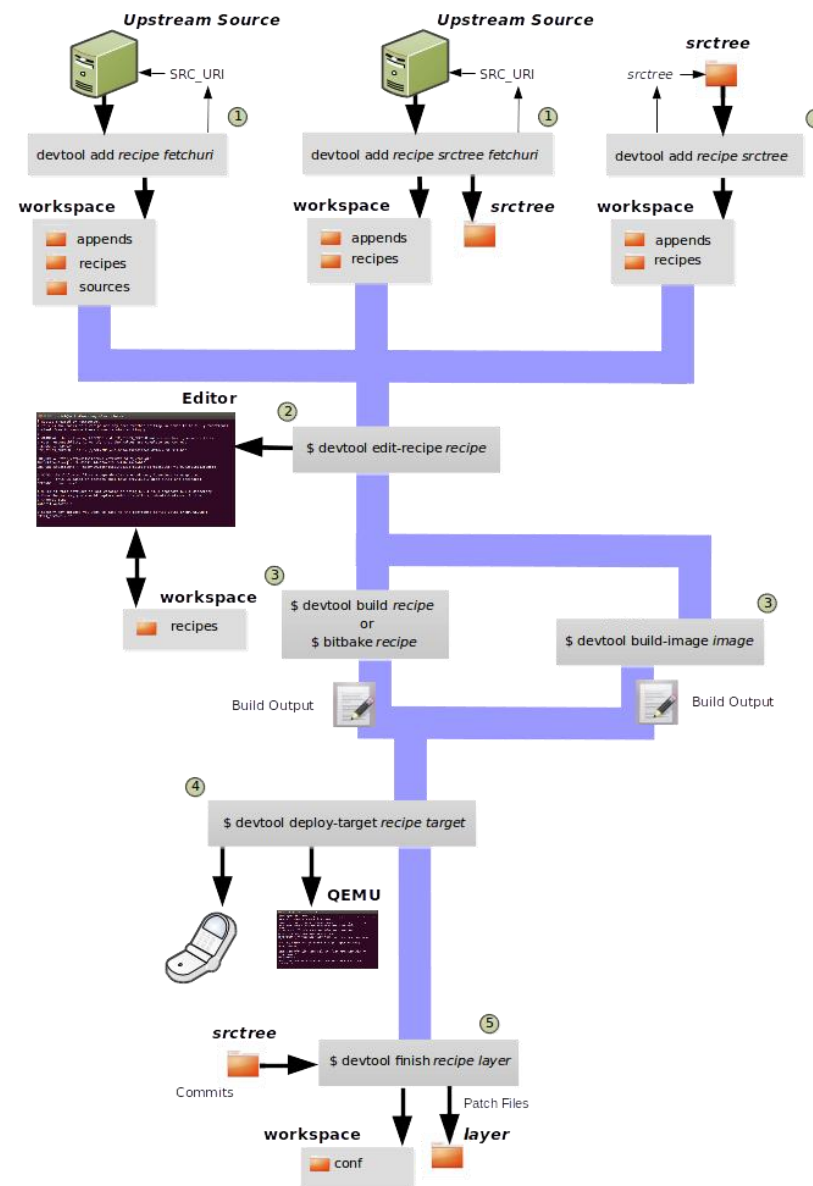
All you need is access to the code, a recipe, and a controlled area in which to do your work.

- Create a new layer
- Generating the New Recipe
- Edit the Recipe
- Build the Recipe
- Deploy to the target device build output
- Populate the layer with the new recipe

Analysis

Note:

During this training, the application source code is similar with the right scenario, the source code have been downloaded from remote side, the local path: `/work/mysources/phytool`



Creating a new layer

- Why create a layer?

When you want to integrate or modify a distribution, the OpenEmbedded principle is to append modification/addons in a dedicated place outside of the already available distribution layer.

- Check the layers included in the distribution:

PC: \$> cd /work/distribution

PC: \$> source layers/meta-st/scripts/envsetup.sh

PC: \$> bitbake-layers show-layers

```
$ bitbake-layers show-layers
layer                                path                                priority
=====
meta-oe                             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-oe 6
meta-gnome                           /local/openstlinux-18-01-23/layers/meta-openembedded/meta-gnome 7
meta-xfce                             /local/openstlinux-18-01-23/layers/meta-openembedded/meta-xfce 7
meta-initramfs                       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-initramfs 8
meta-multimedia                      /local/openstlinux-18-01-23/layers/meta-openembedded/meta-multimedia 6
meta-networking                      /local/openstlinux-18-01-23/layers/meta-openembedded/meta-networking 5
meta-webserver                       /local/openstlinux-18-01-23/layers/meta-openembedded/meta-webserver 6
meta-fileystems                      /local/openstlinux-18-01-23/layers/meta-openembedded/meta-fileystems 6
meta-perl                            /local/openstlinux-18-01-23/layers/meta-openembedded/meta-perl 6
meta-python                          /local/openstlinux-18-01-23/layers/meta-openembedded/meta-python 7
meta-st-stm32mp                      /local/openstlinux-18-01-23/layers/meta-st/meta-st-stm32mp 6
meta-qt5                             /local/openstlinux-18-01-23/layers/meta-qt5 7
meta-st-openstlinux                  /local/openstlinux-18-01-23/layers/meta-st/meta-st-openstlinux 5
```

Creating a new layer

- Create the layer *meta-my-custo-layer* on *meta-st* directory with priority set to 7:

```
PC: $> source layers/meta-st/scripts/envsetup.sh
```

```
PC: $> bitbake-layers create-layer --priority 7 ../layers/meta-st/meta-my-custo-layer
```

```
PC: $> cd ../layers/meta-st/
```

```
PC: $> tree meta-my-custo-layer/
```

- Add the new layer to your configuration

```
PC: $> source layers/meta-st/scripts/envsetup.sh
```

```
PC: $> bitbake-layers show-layers
```

```
PC: $> bitbake-layers add-layer ../layers/meta-st/meta-my-custo-layer/
```

```
PC: $> bitbake-layers show-layers
```

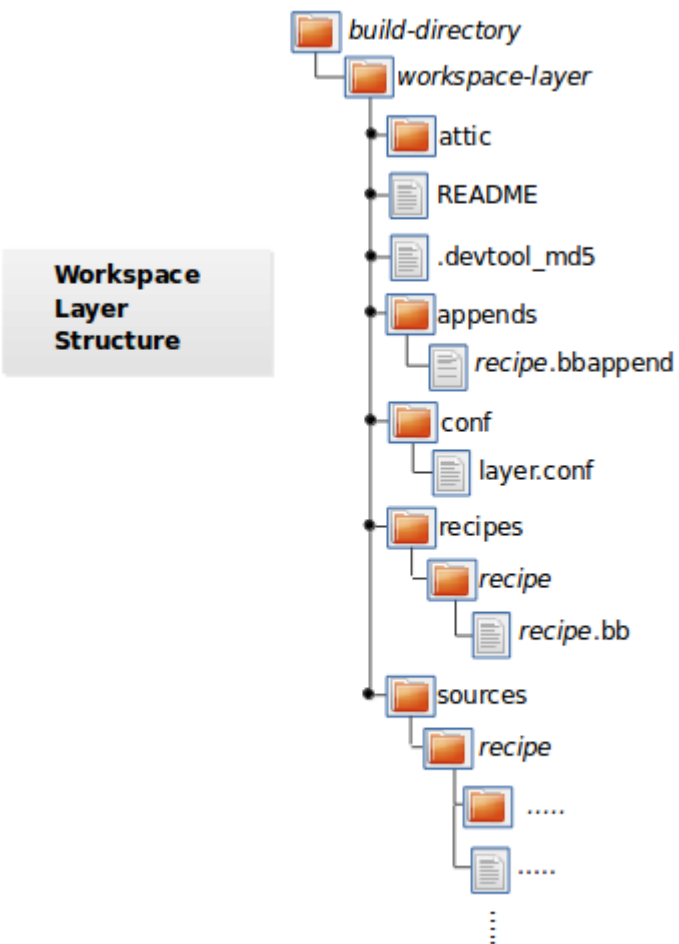
Generating the New Recipe

- Devtool can help you to generate recipe in your OpenEmbedded build setup and more specifically in a layer you manage.
- devtool uses a "Workspace" layer in which to accomplish builds. This layer is not specific to any single devtool command but is rather a common working area used across the tool.
- Way of working

```
PC: $> devtool add phytool /work/mysources/phytool
```

devtool create on workspace a recipe for phytool, have a look at directory:

```
PC: $> cd workspace
```



Edit, build recipe and deploy

- Edit the recipe with devtool

```
PC: $> devtool edit-recipe phytool
```

- Build the recipe

```
PC: $> devtool build phytool
```

- Deploy to the target

```
PC: $> devtool deploy-target phytool root@192.168.7.1
```

- Check on board, the phytool was installed at /usr/bin

```
Board: $> phytool read eth0/0:3/1
```

Populate the layer with the new recipe

- Populate the layer with the new recipe

```
PC: $> mkdir ../layers/meta-st/meta-my-custo-layer/recipes-custom/
```

```
PC: $> mkdir ../layers/meta-st/meta-my-custo-layer/recipes-custom/phytool
```

```
PC: $> cp workspace/recipes/phytool/phytol_git.bb ../layers/meta-st/meta-my-custo-layer/recipes-custom/phytool
```

- Add the new recipe to your image in *meta-st-openstlinux* layer

```
diff --git a/recipes-st/images/st-image-weston.bb b/recipes-st/images/st-image-weston.bb
```

```
index 46998f4..ea2ff31 100644
```

```
--- a/recipes-st/images/st-image-weston.bb
```

```
+++ b/recipes-st/images/st-image-weston.bb
```

```
@@ -39,6 +39,7 @@ CORE_IMAGE_EXTRA_INSTALL += " \
```

```
    ${@bb.utils.contains('COMBINED_FEATURES', 'tpm2', 'packagegroup-security-tpm2', '', d)} \
```

```
    \ packagegroup-st-demo \
```

```
+ phytool \ "
```

Populate the layer with the new recipe

- Rebuild the image to install the phytool

PC: \$> bitbake st-image-weston

- Check the result

PC: \$> cd build-openstlinuxweston-stm32mp1/tmp-glibc/deploy/images/stm32mp1

PC: \$> mkdir /work/test

PC: \$> sudo mount -t ext4 st-image-weston-openstlinux-Weston-stm32mp1.ext4 /work/test

PC: \$> cd /work/test

PC: \$> find -name phytool

Questions

- Please create a new layer named meta-my-test, then add it to layer configuration(10 pts)
- Modify the recipe Linux-stm32mp to:
 - Fetch the kernel source code from below link (20 pts):
[git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git](https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git)
Commit ID: d26b3110041a9fddc6c6e36398f53f7eab8cff82
 - Integrate the fdcan.patch (generated during developer package hands on) into this recipe(20 pts)

Thank you