Building Embedded Operating System with IMGUI Demo for *Raspberry π - 4 - model B* with *Yocto*

Kaloyan Krastev*

Friday 3rd November, 2023 10:49

subversion revision -2

*Triple Helix Consulting

table of contents

1 introduction

These instructions[**kalohowto2023**] follow the configuration and build of a Linux-based operating system for *Raspberry* π - 4 - *model B*[**raspberry**] with *Yocto*[**yocto**]. Find project overview in [**kalo2023**].

The **OS!** (**OS!**) build is done in several steps organized in corresponding sections as follows. Read in Section **??** how to fetch *metadata*. Section **??** shows how to configure the **OS!** build. In Section **??** learn how to build the **OS!** *image* and see how to copy *image* to *SD* card in Section **??**. Section **??** is dedicated to post-install issues like the configuration of the WiFi interface from the command line.

2 metadata

Metadata is a set of instructions to build targets. It is organized in *recipe* files with the *.bb* suffix. Further there are *class* files with the suffix *.bbclass* with information shared between *recipes*. Finally, there are configuration files with the extension *.conf*. These define configuration variables to control the build process. *Metadata* is organized in *layers*. Layers logically separate information of a project. *OpenEmbedded*[**oe**] defines the following layer types.

- base layers contain base *metadata* for the build
- machine aka **BSP!** (**BSP!**) layers include **HW!** (**HW!**) support
- distribution layers hold the policy configuration
- **SW!** (**SW!**) layers are used for additional **SW!**
- miscellaneous layers do not fall in upper categories

The complete list of *github* **SW!** *metadata* repositories used in this project includes *Yocto* layers, the *Raspberry* π - 4 - *model B* **BSP!** layer, a **SW!** layer with custom recipes, and the build configuration itself. Please refer [**kalo2023**] for details.

In short, users fetch *metadata* in contrast to the *real data* fetched by *bitbake* during **OS!** build. See Section **??** for details. It is an user decision where to put fetched *metadata*. However, it is nice to have all layer sub-directories in one location. In these instructions this location is referred as <layer_directory>. The second directory to create is the <build_directory>. This is where the build and build configuration live. I suggest that this one is not inside the <layer_directory> to not mix *data* and *metadata*.

2.1 requirements

It is very likely that you will need to install *Yocto* requirements[**yoctoqb**] to be able to run *bitbake*. There you find a list of packages to install. *Yocto* sanity checked distributions are *poky*-3.3, *poky*-3.4, *Ubuntu*-18.04, *Ubuntu*-20.04, *Ubuntu*-22.04, $Fedora$-37, $Debian-11$, $OpenSUSEleap$-15.3, $AlmaLinux$-8.8. I use *bitbake* on a rolling release $Manjaro$ Linux. It should not be complicated to satisfy *Yocto* on machines with GNU/Linux operating system. Maybe binaries are not the same on different **HW!** architectures, but the **OS!** is a simplified open-source **OS!** with a Linux kernel with the proper **HW!** configuration.

Install the following packages;

- *git*
- *tar*
- *python*
- *gcc*
- *GNU make*

Find more details in *Yocto* documentation at [**yoctoqb**]. You may need to install in addition $diffstat$, $unzip$, $texinfo$, $chrpath$, $wget$, $xterm$, sdl, $rpcsvc-proto$, $socat$, $cpio$, $lz4$ and $inetutils$ packages. As a double check, make sure to have the following command-line tools on your host machine: $chrpath$, $diffstat$, $lz4c$, $rpcgen$. Then have a look at your storage device. Fetched *metadata* requires 412 MB of free space. The build may need up to 30 GB or 50 GB if intermediate files are kept. Read for the *bitbake* class rm_work in Section **??**.