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

Kaloyan Krastev*

October 17, 2023

*Triple Helix Consulting[10]

table of contents

1 introduction

These instructions[6] follow the configuration and build of a Linux-based operating system for *Raspberry* π - 4 - *model B*[8] with *Yocto*[1]. Find project overview in [7].

The *operating system* (OS) build is done in four steps and instructions are organized in four corresponding sections as follows.

- section 2 get *metadata*
- section 3 configure OS build
- section 4 build OS *image*
- section 5 copy *image* to *SD* card

Section 6 is dedicated to post-install issues like the configuration of the WiFi interface from the command line.

2 download

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* defines the following layer types.

- base layer
  base metadata for the build
- machine aka *board support package* (BSP) layer
  hardware (HW) support
- distribution layer
  policy configuration
- *software* (SW) layer
  additional SW
- miscellaneous layer
  for layers that do not fall in upper categories

A complete list of *github* SW repositories used in this project includes *Yocto*, the BSP, a SW layer with custom recipes, the configuration and the source code of the application and the dependencies. Note that for a relatively simple application I must fetch six SW repositories. Follow links for details.

- *Yocto* reference distribution yoctoproject.org/poky.git
- BSP layer for *Raspberry* π boards agherzan/meta-raspberrypi.git

- *Yocto* configuration TripleHelixConsulting/yocto_x86_BasicConfig.git
- SW layer kaloyanski/meta-thc.git
- Immediate mode *graphical user interface* (GUI) kaloyanski/imgui_aar
- OpenGL library glfw/glfw.git

```
mkdir <your layer directory>

git clone -b kirkstone \
git@github.com:yoctoproject/poky.git \
<your layer directory>

git clone -b kirkstone \
git@github.com:openembedded/meta-openembedded.git \
<your layer directory>


mkdir <your layer directory>/rpi
git clone -b kirkstone \
git@github.com:agherzan/meta-raspberrypi \
<your layer directory>/rpi

mkdir <your layer directory>/thc
git clone git@github.com:kaloyanski/meta-thc.git \
<your layer directory>/thc

mkdir -p <your build directory>/conf
git clone \
git@github.com:TripleHelixConsulting/yocto_x86_BasicCo
```

<your build directory>/conf

3 configuration

Dear ImGui[2] is a bloat-free GUI library for C++. It outputs optimized vertex buffers that you can render anytime in your 3D-pipeline-enabled application. It is fast, portable, renderer agnostic, and self-contained (no external dependencies). *Dear ImGui* is designed to enable fast iterations and to empower programmers to create content creation tools and visualization/debug tools (as opposed to UI for the average end-user). It favors simplicity and productivity toward this goal and lacks certain features commonly found in more high-level libraries. *Dear ImGui* is particularly suited to integration in game engines (for tooling), real-time 3D applications, full-screen applications, embedded applications, or any applications on console platforms where operating system features are non-standard.

Dear ImGui depends on *GLFW*[3], an open-source, multi-platform library for *OpenGL*, *OpenGL ES* and *Vulkan* development on the desktop. It provides a simple API for creating windows, contexts and surfaces, receiving input and events. *GLFW* is written in *C* and supports *Windows*, *macOS*, *X*11 and *Wayland*.

Dear ImGui is licensed under the *MIT* License. *GLFW* is licensed under the *zlib/libpng* license.

3.1 layers

Here is a list of *Yocto* layers. The project reference distribution is *poky*.

- *meta*
  User-space data

- *meta – poky*
  Yocto reference distribution

- *meta – raspberrypi*
  This[4] is the general HW specific BSP overlay for the *RaspberryPi* device. The core BSP part of *meta – raspberrypi* works with different *OpenEmbedded/ Yocto* distributions and layer stacks. In short, the recipes to build the kernel and kernel modules are in this layer. For details see the package *linux-raspberrypi*. In addition, here is the HW specific firmware. By chance, the build configuration corresponds the specific HW, in this case *Raspberry π - 4 - model B*.

- *meta – thc*
  I have introduced a new *Yocto* SW layer to control the build of *Dear ImGui* and *GLFW*. As long as the source codes have a standard build configuration, the *bitbake* recipes are straightforward. Both instructions inherit *cmake*.

4 build

4.1 configuration

Yocto provides a list of image types. For obvious reasons, I have chosen $core-image-x11$[1] - a very basic X11 image with a terminal. In the main build configuration, apart from *Dear ImGui* and *GLFW*, I have added the following packages;

- $os-release$
  OS identification
- *Dropbear*
  Compact *secure shell* (SSH) server[5]
- *dhcpcd*
  dynamic host configuration protocol (DHCP) client[9]
- *thcp*
  OS post-configuration scripts

5 install

The total size of the operating system is between from 250 up to $384MB$ or $79MB$ *tar.bz* archive, including kernel *ARM*, 64 *bit* boot executable *image* of $23MB$, a *Raspberry* π - 4 - *model B* configuration of Linux 5.15. The total size of kernel modules is $21MB$. Happily this kernel release has a $long-term\ support$ (LTS). The list of packages included in the OS *image* in Table 1 gives a good idea of the contents.

Yocto provides multiple package and *image* formats. Further, different ways exist to install *images* on *SD* card. The result is an

package	description
packagegroup-core-boot	boot
packagegroup-base-extended	base
run-postinsts	post
opkg	package manager
psplash-raspberrypi	*Raspberry* π - 4 - *model B* splash
packagegroup-core-x11-base	the *X* server
os-release	OS identifier
dropbear	SSH server
dhcpcd	DHCP client
thcp	SW layer
glfw	*OpenGL*
imgui	*Dear ImGui*

Table 1: A list of packages in *core-image-x*11-*raspberrypi*4-64

OS with two partitions only - */root* and */boot*. There are not *swap* and *home* partitions. I recommend the classic command-line tool *dd* to copy data. It works fine with different *image* formats like $rpi-sdimg$, *hddimg* and *wic*. The last format is recommended. Find the card device name, usually */dev/sda*, unmount it with *umount* if it is mounted, and do copy data with the simple command

$$dd\ if{=}whatever.wic\ of{=}/dev/sda$$

Run this command with *root* privileges and be careful to not specify the device name of your hard drive. This will take a while. When it is over, put the card in you *Raspberry* π - 4 - *model B* and turn it on. That's it.

6 run

Connected embedded systems can communicate to one another and to cloud-based *platform-as-a-service* (PaaS) solutions. In addition, a remote control may be required. An SSH server is a standard solution for both problems.

Wireless connection is established via classic command-line tools like *ip*, *iw*, *dhcpcd*, and *wpa_supplicant*. Custom shell scripts are installed in */usr/bin*, as well as a running GUI example to demonstrate the usage of the *Dear ImGui* library. Once an *internet protocol* (IP) address is assigned, the SSH server by *Dropbear* allows for a secured remote login, remote control and file transfer.

7 outlook

This reports the progress in the development of a custom Linux-based OS for *Raspberry* π - 4 - *model B*[8]. The kernel version of this embedded OS is Linux release 5.15. An example GUI application using the *Dear ImGui* library is built as a part of the OS *image*. In addition, an SSH server provides remote connection, data transfer and device control. As the OS is now functional, performance and real-time tests are ongoing.

acronyms

BSP *board support package*

SSH *secure shell*

GUI *graphical user interface*

SW *software*

HW *hardware*

OS *operating system*

DHCP *dynamic host configuration protocol*

IP *internet protocol*

PaaS *platform-as-a-service*

LTS *long – term support*

bibliography

[1] Yocto Project community. *Yocto Project*. 2023. URL: https://www.yoctoproject.org (visited on 2023).

[2] Omar Cornut. *Dear ImGui*. 2023. URL: https://github.com/ocornut/imgui (visited on 2023).

[3] Marcus Geelnard and Camilla Löwy. *OpenGL*. 2023. URL: https://www.glfw.org (visited on 2023).

[4] Andrei Gherzan. *meta-raspberrypi*. 2023. URL: https://github.com/agherzan/meta-raspberrypi (visited on 2023).

[5] Matt Johnston. *Dropbear SSH*. 2023. URL: https://matt.ucc.asn.au/dropbear/dropbear.html (visited on 2023).

[6] Kaloyan Krastev. *Building Embedded Operating System with IMGUI Demo for Raspberry π - 4 - model B with* Yocto. 2023. URL: https://kaloyanski.github.io/meta-thc/thchowto.html (visited on 2023).

[7] Kaloyan Krastev. *Embedded Operating System for Raspberry π - 4 - model B with* Yocto. 2023. URL: https://kaloyanski.github.io/meta-thc/thcport.html (visited on 2023).

[8] Raspberry π Ltd. *Raspberi π*. 2023. URL: https://www.raspberrypi.com (visited on 2023).

[9] Roy Marples. *dhcpcd*. 2023. URL: https://roy.marples.name/projects/dhcpcd (visited on 2023).

[10] Atanas Rusev. *Triple Helix Consulting*. 2023. URL: https://triplehelix-consulting.com (visited on 2023).