Raspberry Pi 4 Environment Configuration

Jaidon Lybbert, University of Washington

January 24, 2024

Abstract

This report documents the setup and configuration of a portable development environment for a headless Raspberry Pi 4 on a cellular hotspot. The environment consists of a Windows 10 laptop, iPhone 13 Pro, and Raspberry Pi 4B. The Pi is pre-configured with the SSID and password to connect to the iPhone hotspot, and accessed from the laptop through SSH. The Pi is set up with the Neovim editor for on-device development, with language support for Python, Rust, C, C++, and Zig. For debugger support, the Windows laptop is set up with Visual Studio and VSCode with extensions for the same languages.

1 Introduction

This document is organized as follows. In Section 2 I describe my experience with embedded systems and software, motivations and intent, and take a look at the physical interface to the Raspberry Pi 4. I devise a plan for setting up my preffered environment in Section 3, describe what the actual environment ends up looking like in Section 4, and discuss the failures and hurdles I faced in Section 5. I conclude in 6 by discussing the improvements I would like to make in the future.

2 Background

2.1 Programming Experience

I graduated with my BS in Electrical Engineering at Eastern Washington University in 2022, with a concentration in Embedded Systems. As part of

my coursework, I programmed an FPGA in VHDL to implement a processor supporting a subset of the MIPS instruction set. I did several small projects using the TI ARM® Cortex®-M4F Based TM4C123G LaunchPad™ MCU evaluation board [5], and was introduced to real-time operating systems using the Arduino Mega which I used to make a kitchen timer[7] with polling and pre-emptive scheduling.

While at EWU, I was an officer in the IEEE Student Chapter, and made a controller for a skittle-sorting machine, again using the TM4C123G [9].

For two years, I worked at an aerospace startup developing real-time control software for a pressurized gas system using an Allen-Bradley PLC.

I spent the summer of 2022 doing an internship at SpaceX, where I developed tools in Python to automate workflows and manage parts data.

My graduate studies have been in high-performance parallel computing on the NVIDIA Cuda platform, with some graphics programming in OpenGL and Vulkan, and some deep neural networks. My most recent work has been developing a parallel QR decomposition algorithm with Amazon Lab126 for in-home robotics applications[6].

2.2 Experience with the Raspberry Pi

I have done some projects with the Raspberry Pi 3B, and Raspberry Pi-ZeroW. For a time, I had the Raspberry Pi 3B set up as an ad-blocking DNS server using the Pi-hole project [1], and I used the ZeroW for a weather station project as an undergrad in IEEE [4].

2.3 Motivation

My experience with bare-metal programming and parallel programming on GPUs has given me a great appreciation for on-device debuggers, and memory management. My recent interests have been in programming languages themselves, and the tradeoffs they offer. At the same time, I have been becoming more mobile, and work in a variety of places from my laptop, leaving the desktop, two monitors, and mouse at home. The single low-res screen and trackpad have led me down the path of keyboard shortcuts, macros, and Neovim to ergonomically switch between windows and edit code without using the trackpad.

These motivations form the basis of the requirements for my development environment (plus version control as a given).

- Portability
- Version control
- Support for several languages
- On-device debugging
- Compatibility with Neovim

3 Methods

3.1 Installation

The Raspberry Pi 4B is a relatively simple device to setup following the documentation [3]. I used a 16GB SD card and flashed the 64-Bit Raspberry Pi OS "Bookworm" to it on my Windows 10 laptop with the provided Raspberry Pi Imager tool. I took a screenshot of the hostname and username configuration, so I wouldn't forget it.

3.2 Portability

In advanced settings in the Imager tool, the pre-configured WiFi network was automatically filled, along with the public SSH key from my laptop. I changed the SSID and password for the WiFi network to my iPhone hotspot so I have the option to physically bring my Pi with me and connect with SSH.

The Pi connected to the hotspot on boot without issue, and I was able to SSH into it by the hostname. I immediately scanned for my home WiFi network, and added it with higher priority using the Network Manager CLI. Switching networks caused the SSH session to hang, but switching networks on my laptop and restarting the SSH connection went without issue.

I updated the Pi with APT through my home network.

3.3 Version Control

Once my packages were up to date, I generated an SSH key pair with `ssh-keygen` and copied the public key to my personal Github account. I created a new repository [8] and cloned it to the Raspberry Pi.

3.4 Language Support

At this point I checked the remaining space on my SD card with the `df -H` command. I would keep an eye on it as I began installing packages.

The full set of languages I want to support are:

- Python
- Rust
- C
- Zig
- C++

I found GCC and Python came pre-installed with the OS. I installed Rust with by fetching and executing the installer with the one-liner

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.
   rustup.rs | sh
```

as suggested by the docs [2]. I also installed CMAKE to support building C and C++ applications.

A development environment is set up on the device using Neovim with language support for Python, Rust, C, C++, and Zig. By default, Neovim supports syntax highlighting for these languages, for code completion and static analysis, language-specific plugins are added. A VSCode-based development environment is set up on the Windows 10 machine for instances when a debugger is necessary, with extensions for Python, Rust, and Zig. Visual Studio is used for C and C++, since those projects are built with MSBuild through CMAKE, which outputs Visual Studio projects. Git is installed on both machines for version control.

4 Results

I initially tried installing the Raspberry Pi OS using the same method I had used for the Raspberry Pi 3B several years ago. It used to be the case that you could pre-configure the WiFi connection by flashing the image to the SD card, then copying a `wpa_supplicant.conf` file to the boot partition before installing it to the device. This is no longer supported by the "Bookworm" Raspbery Pi OS and onwards [3].

5 Discussion

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

6 Conclusion

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

References

[1] Pi-hole: A black hole for internet advertisements. Accessed: 2024-01-22.

[2] The Rust Project Developers. Installation. Accessed: 2024-01-23.

[3] Raspberry Pi Documentation. Configuration, 2021. Accessed on 2024-01-22.

[4] EWU-IEEE-Spokane. Weather-station, 2024. Accessed: 2024-01-22.

[5] Texas Instruments. Ek-tm4c123gxl evaluation board. Accessed: 2024-01-22.

[6] Jaidon Lybbert. Mixed-precision block qr. GitHub repository, 2020.

[7] Jaidon Lybbert. Kitchen timer. `https://github.com/jaidonlybbert/EENG462/blob/main/Labs/Final/jjlybbertFinal/jjlybbertFinal.ino`, 2023. Accessed: 2024-01-22.

[8] Jaidon Lybbert. Classwork for embedded and real-time systems, university of washington, winter 2024. `https://github.com/jaidonlybbert/EEP522A-W24`, 2024. Accessed: 2024-01-23.

[9] Jaidon Lybbert, Cody Birkland, sirchicdbruk, and msheldon1986edu. Ewu-ieee-spokane/sorter-project, 2020. Accessed: 2024-01-22.

7 Acknowledgments

Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

A Appendix A

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

B Appendix B

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien, venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

C Figures and Tables

A	B	C
1	2	3
4	5	6

Table 1: Example of a table caption

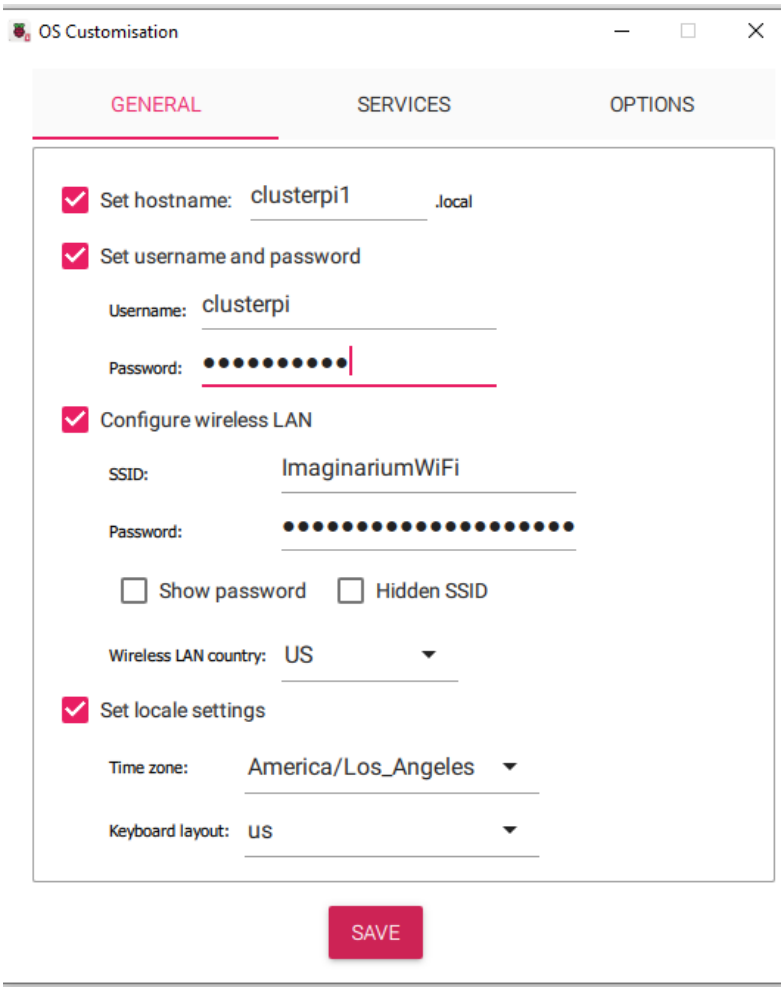

Figure 1: The Raspberry Pi Imager showing hostname and username configuration.

```
clusterpi@clusterpi1:~ $ nmcli --fields autoconnect-priority,name connection
AUTOCONNECT-PRIORITY  NAME
10                    6027 WiFi
0                     lo
0                     preconfigured
-999                  Wired connection 1
```

Figure 2: WiFi network prioritization in Network Manager. The mobile hotspot was labeled "preconfigured" by default, the home network is named by the SSID "6027 WiFi".

```
clusterpi@clusterpi1:~/.ssh $ df -H
Filesystem      Size  Used Avail Use% Mounted on
udev            1.8G     0  1.8G   0% /dev
tmpfs           398M  1.3M  397M   1% /run
/dev/mmcblk0p2   16G  4.7G  9.6G  33% /
tmpfs           2.0G  144k  2.0G   1% /dev/shm
tmpfs           5.3M   17k  5.3M   1% /run/lock
/dev/mmcblk0p1  535M   76M  459M  15% /boot/firmware
tmpfs           398M   37k  398M   1% /run/user/1000
```

Figure 3: Free space on the SD card after basic installation. The filesystem has 9GB available.

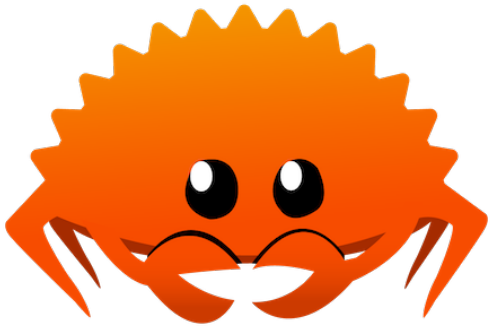

Figure 4: Example of a figure caption