COMP 4900E - Real Time Operating Systems

Carleton University

Updating a Raspberry Pi Over The Air (OTA) Using QNX

April 10, 2024

Anish Phadnis: 101201491

Courtney Thirukumaran: 101191547

Emmanuel Richmond: 101108167

Huseyin Kabasakal: 101156317

Manuel Osvaldo Lebron Flores: 100958229

Professor

Jun Huang

Abstract

Our project entertains the journey of our innovative Over-the-Air (OTA) software updating solution utilizing the QNX real-time operating system (RTOS) and the Raspberry Pi platform. With today's rapidly evolving technological landscape, ensuring that devices remain up to date with the latest software versions that respective companies release is paramount for optimal performance and security for each consumer's device. Our topic is especially of value to clients in remote locations as traditional methods of software updating often require manual intervention, which can be costly and time-consuming. Thanks to the generous sponsorship of Blackberry, our group has endeavoured to leverage the many capabilities of QNX and flexibility of the Raspberry Pi in order to shift the software updating process to function wirelessly by enabling remote updates over the air.

Acknowledgement

We would like to express our gratitude to Professor Jun Huang's guidance and support provided over the course of this project. We appreciate his efforts to ensure we had access to all the tools we needed to proceed successfully in our exploration of an OTA software solution. We would also like to thank Blackberry for giving us QNX licenses and technical support throughout our project.

GitHub Repository

https://github.com/Antel0pe/Raspberry-Pi-OTA-Demo

Walkthrough Video Guide

See YouTube: https://www.youtube.com/watch?v=b2t9bmDXT34

Walkthrough Tutorial

See GitHub (Technical Tutorial Manual - Updating a Raspberry Pi Over The Air (OTA) Using QNX): https://github.com/Antel0pe/Raspberry-Pi-OTA-Demo

Research Project Slides

See GitHub (OTA Slideshow Demo.pdf): https://github.com/Antel0pe/Raspberry-Pi-OTA-Demo

Table of Contents

1. Introduction

The growth of technologically embedded systems in modern markets around the world continues to revolutionize several industries. From automotive to healthcare, tech that offers enhanced functionality, connectivity, and efficiency to standard every-day products seems to be a successful factor for most companies. However, it's important to note that as the developmental rate in expansion of connected devices continues to grow, so do the challenges associated with managing and maintaining the relevancy and security of the software that is powering them. Ensuring that devices remain updated with the latest software versions is crucial to mitigate vulnerabilities, enhance performance, all while introducing new features that will aid in the scalability of the product as a whole. Traditional methods of software updating often require manual interventions or physical access to specialized tools in order for any kind of improvement to occur. This approach not only consumes valuable time and resources for clients and the companies responsible, but also poses subsequent risks such as potential disruptions to device operation or even security breaches that may take place during the update process. While analyzing the effects of this matter, the need for an efficient and reliable Over-the-Air (OTA) software updating solution becomes increasingly evident. In response to this challenge, our project explores the development and implementation of an OTA solution tailored specifically for QNX-based embedded systems that are deployed on the Raspberry Pi platform. QNX, renowned for its reliability, scalability, and real-time performance, will serve as the perfect foundation for producing such a software solution. By providing a robust operating environment conducive to market-critical applications, Blackberry's Raspberry Pi will serve as a versatile and cost-effective computing platform that offers an ideal space for prototyping and deploying solutions for connected devices.

1.1) Overview of Project

Our team researched various approaches to perform an integration of the QNX operating system image loader and implement an Over The Air (OTA) software solution on a Raspberry Pi device. In the end, we decided to create scripts that will transfer the new image to the board and place it in the correct partition within the SD card. Our script later ensures that the board is rebooted so the update takes effect. The old QNX version we will be using is version 7.1 and

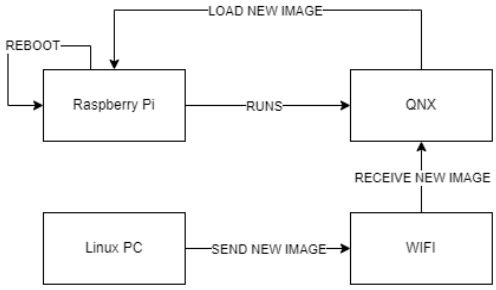
with our scripts, we updated the QNX OS to version 8.0. At a high level, this entails: creating scripts that can modify the original bin file which runs QNX 7.1. This requires the scripts located within the build file to mount the SD card and modify the files which are used to boot QNX. The new updated files of QNX 8.0 will be placed here and then the system will be rebooted. Note that U-Boot is responsible for initializing the hardware and loading the QNX kernel into memory before booting the operating system on the Raspberry Pi. U-Boot provides a flexible and customizable environment that allows users to configure various boot parameters, manage memory, and load kernel images from different storage devices. U-Boot's characteristics allow us to make changes to its system and reboot updated versions of the QNX OS. Throughout the course of our project, we mainly used QNX documentation provided by BlackBerry to deal with bugs, and to determine appropriate approaches to implement changes to the QNX OS on the Raspberry Pi.

The following lists the high-level functionality of our solution (and will be explored more in the following report):

Windows 11 PC sends image via WIFI to Raspberry Pi board running QNX.

1. Linux PC connects via Wifi.
2. Linux PC sends new image via wifi.
3. QNX receives new image via wifi.
4. QNX loads a new image onto the SD card on Raspberry Pi.
5. Raspberry Pi reboots.
6. Raspberry Pi runs the new QNX image.

Figure 1: Network Architecture of Project



You can access our project at the following GitHub repository link for more details on the build files, scripts, and our overall project design: [COMP 4900E - OTA Project]

2. Contributions

Over the course of a couple months, we met as a group each week to discuss possible approaches to our software solution, ensure roadblocks were solved, and maintain clear communication between team members.

Listed below is the overall breakdown of contributions made by various members of our group:

Anish Phadnis

- Research (high-level & low-level, Wifi connection), Development & Testing, Report Write-up (OTA Tutorial Guide), CLI Figures & Screenshots in Report, Group Discussions & Debug Sessions, Slideshow Demo

Courtney Thirukumaran

- Research (high-level), Report Write-up (Acknowledgements, Overview of Project, Contribution, OTA Tutorial Guide, Conclusion/Future Work), Group Discussions & Debug Sessions, Slideshow Demo

Emmanuel Richmond

- Research (high-level & low-level), Report Write-up (Abstract, Introduction), Group Discussions & Debug Sessions, Slideshow Demo

Huseyin Kabasakal

- Research (high-level & low-level), Development & Testing, Group Discussions & Debug Sessions, Slideshow Demo, Report Write-up (OTA Tutorial Guide)

Manuel Osvaldo Lebron Flores

- Research (high-level & low-level), Testing, Report Write-up (OTA Tutorial Guide), CLI Figures & Screenshots in Report, Group Discussions & Debug Sessions, Slideshow Demo

3. OTA Tutorial Guide

This section will focus on the steps involved in replicating our project. This will allow readers to understand the process involved in performing an OTA update on a Raspberry Pi as well as understand how our group determined our solution. Readers will also gain insights about real-time operating systems in the real world on a small-scale.

3.0) Setup-Prerequisites:

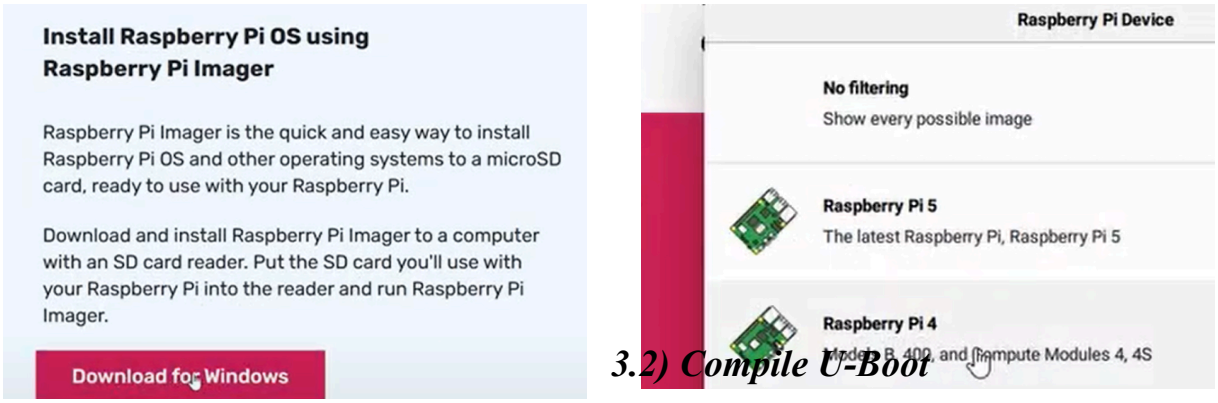
Prerequisite Tools:
PC: Windows 11
OS: QNX 7.1/8.0, Windows 11
Compiler: gcc 13.2
Hardware: Raspberry Pi 4 Model B (Processor: Cortex A72), Micro USB, USB to TTL serial cable, Power Supply, SD Card

3.1) Setting up Raspberry Pi & Tools

1. Follow the guide provided by Blackberry entitled: BSP User's Guide Raspberry Pi 4 Board
    a. NOTE: The BSP guide is only available to QNX license holders[1].

2. Additionally, complete the steps involved to prepare a bootable microSD card using the Raspberry Pi 4 Board.
    a. NOTE: There is a known problem where Windows OS cannot format SD cards greater than 32 GB so you may need an additional program such as a mini-tool partition wizard[2] to format the card.

Figure: Depicts the process of installing the Raspberry Pi imager and Raspberry Pi OS.



3.2) Compile U-Boot

Download the needed files to compile U-Boot on the Raspberry Pi 4[3] (without any updates - just a dry compile to first setup U-Boot on the Pi).

NOTE: The setup only works on Debian OS. It will create a uboot.img that should replace the temporary image created by the wizard on the SD card.

Figure: Debian VM

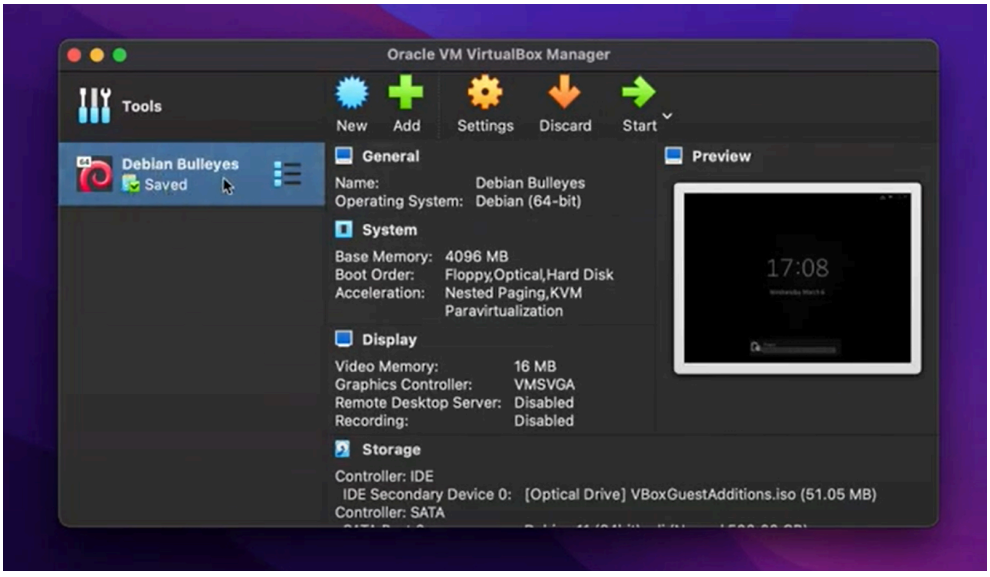


Figure: Copying Boot Files from Linked Documentation to Compile on Debian

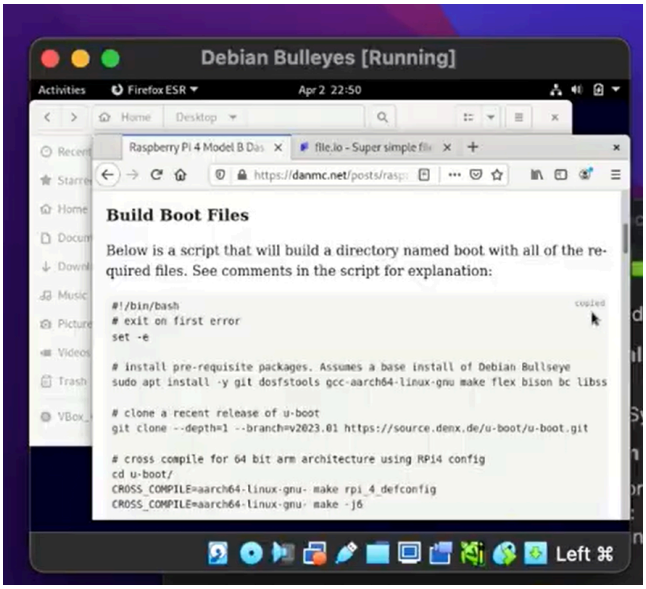


Figure: Running the Copied Bash File on VM

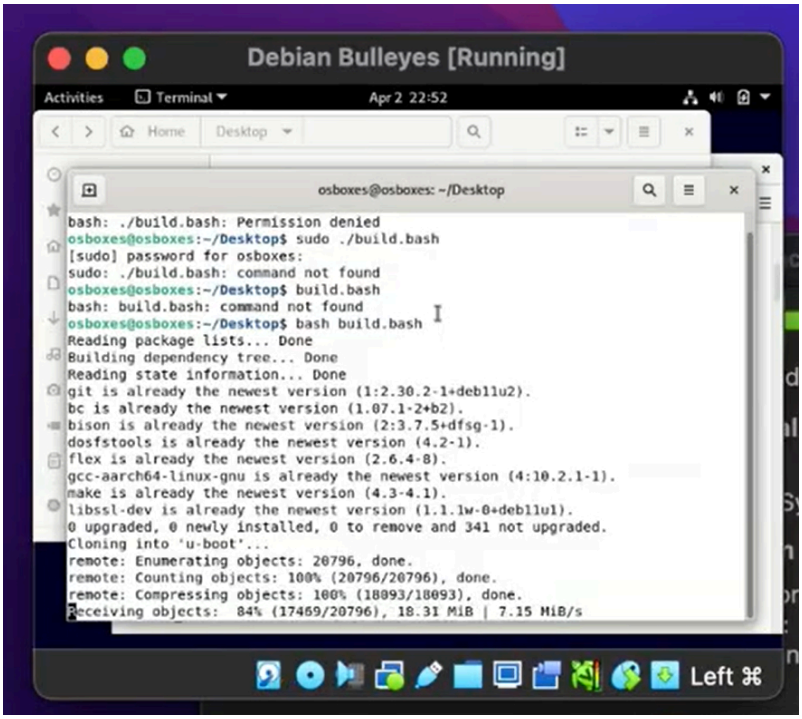
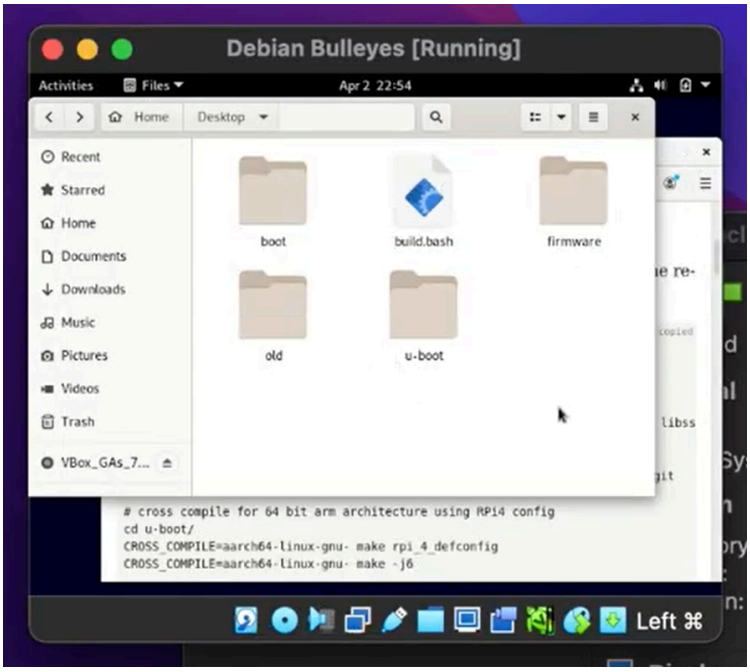


Figure: Boot, Firmware & U-Boot Folders Created



3.3) Windows 11 SSH Set up

Download the needed files to run OpenSSH Server[4] for a Windows 11 Host. This allows the Pi to remotely access files on your computer and enables wireless transfer of images.

Figure: Installing OpenSSH Server on Windows 11

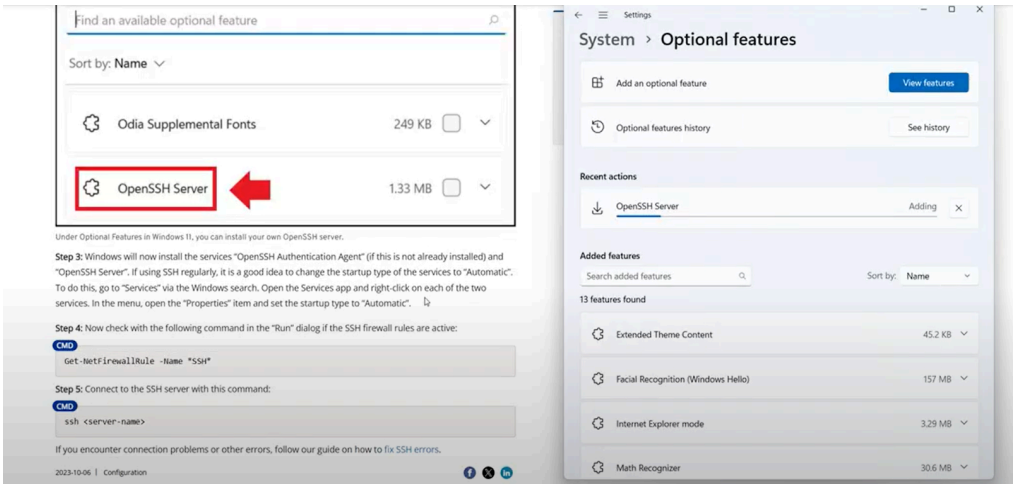
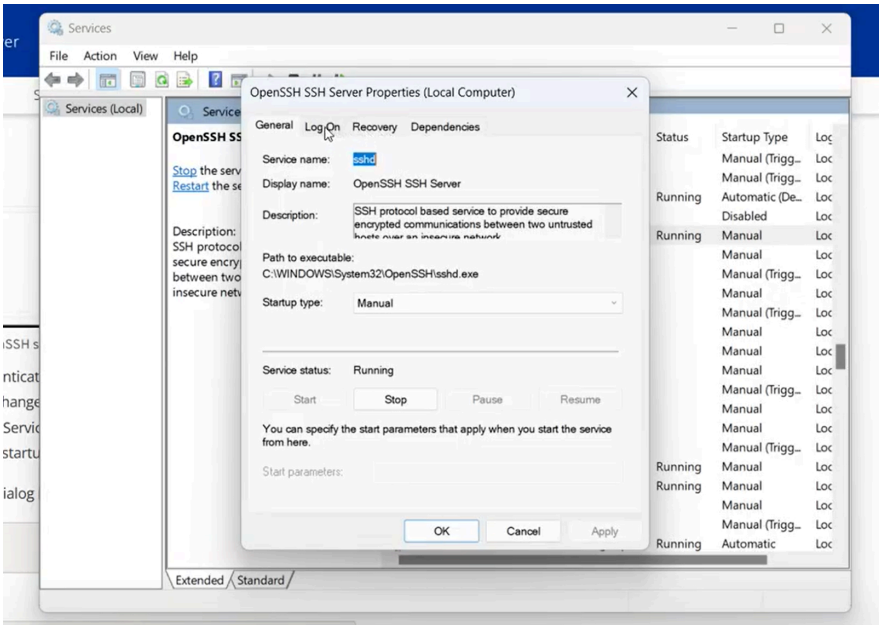


Figure: Running OpenSSH Server and Authentication Agent on Windows



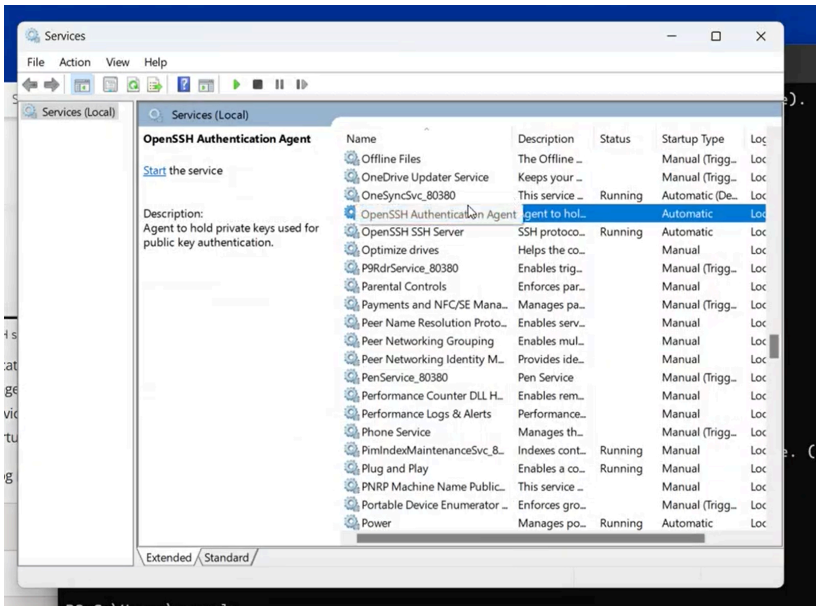


Figure: Ensuring SSH Successfully Connects

NOTE: Have a password for your PC!

3.4) Download Software Center and QNX

Download QNX Software Center to install QNX SDP 7.1. You can follow Carleton University's QNX on Raspberry Pi tutorial[5] under the "Install software on development computer" section to do so.

NOTE: Licensing will be needed. This can be requested after you have downloaded and created an account with Blackberry QNX.

Figure: Section to Install on Carleton University Website[5]

Figure: Login/Create Account with QNX Center

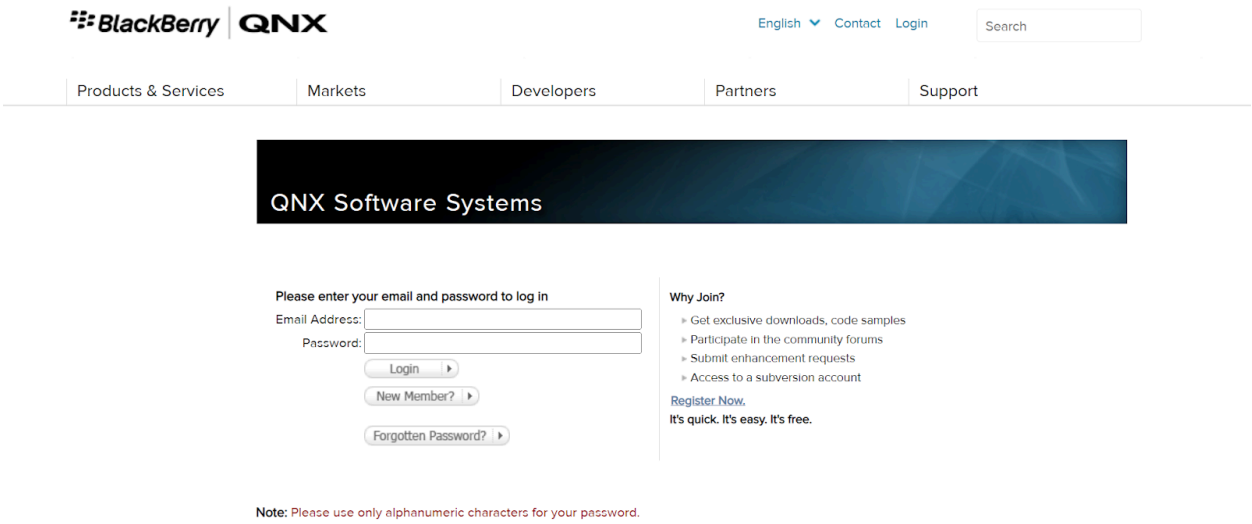


Figure: Under the Download Section on the Website Click the Following

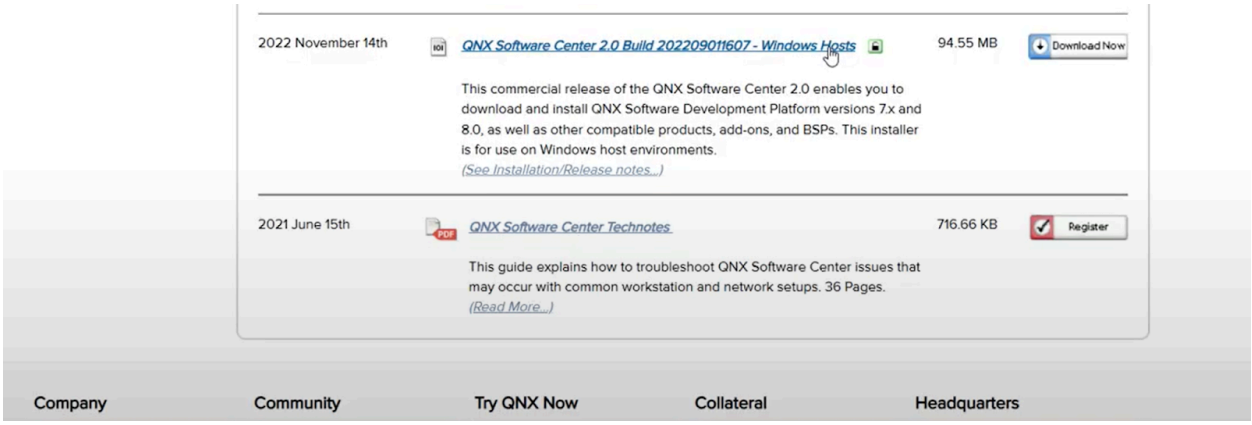
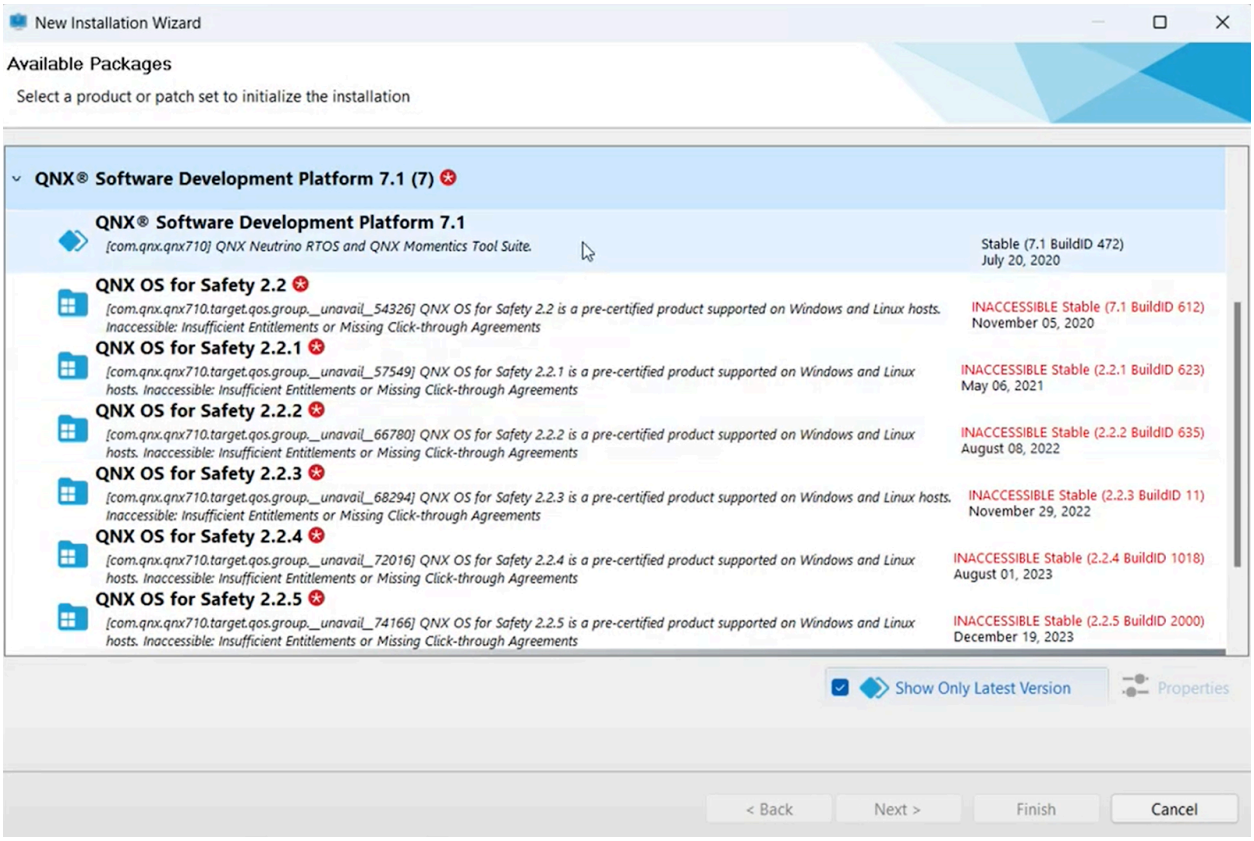


Figure: QNX Software Center Opens & Should Activate 7.1 License



3.5) Download And Set up BSP

To set up the BSP, follow the steps found under the “Download and set up the BSP” section of the QNX guide[1].

a. NOTE: This may involve editing some environment variables on your PC.

Figure: Editing the necessary environment variables.

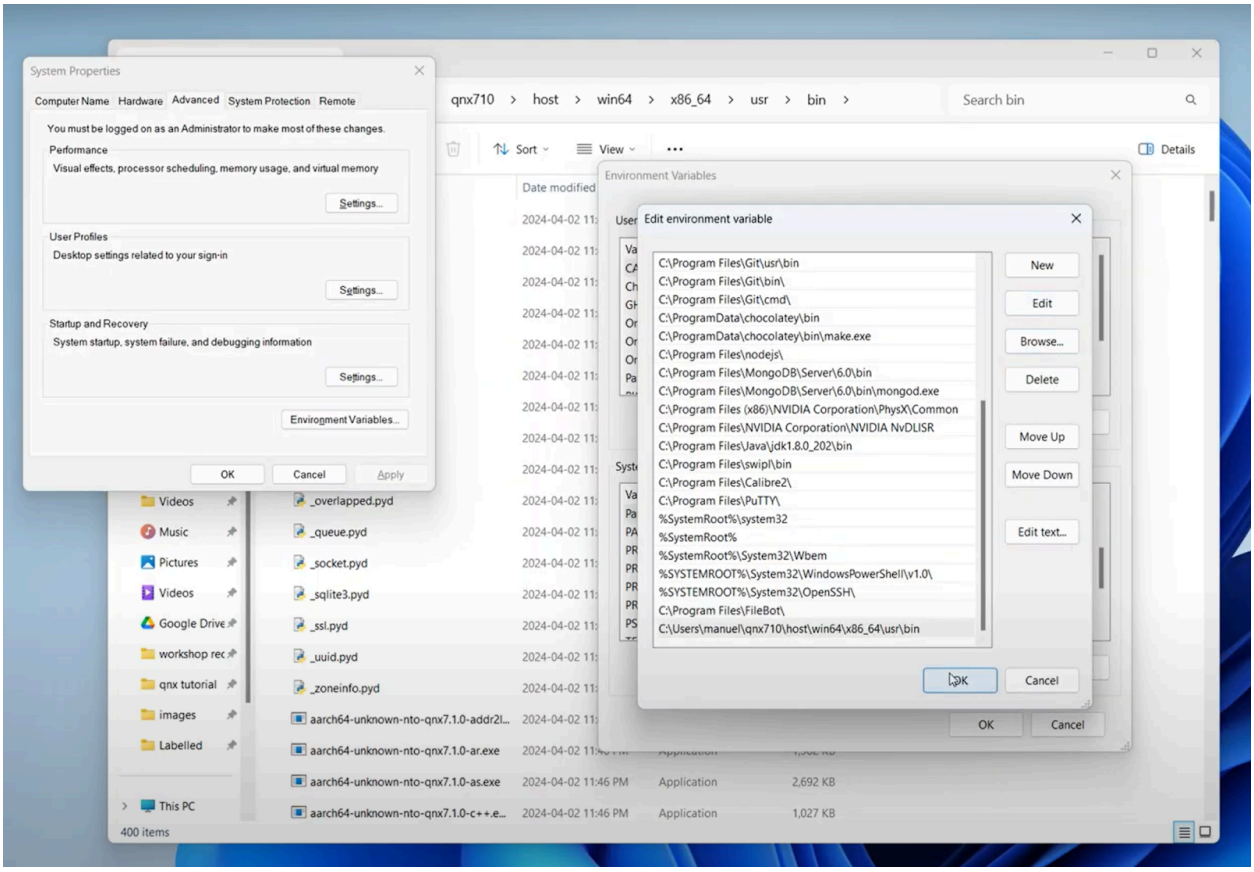


Figure: Running ‘qnxsdp-env.bat’ to define necessary QNX specific environment variables

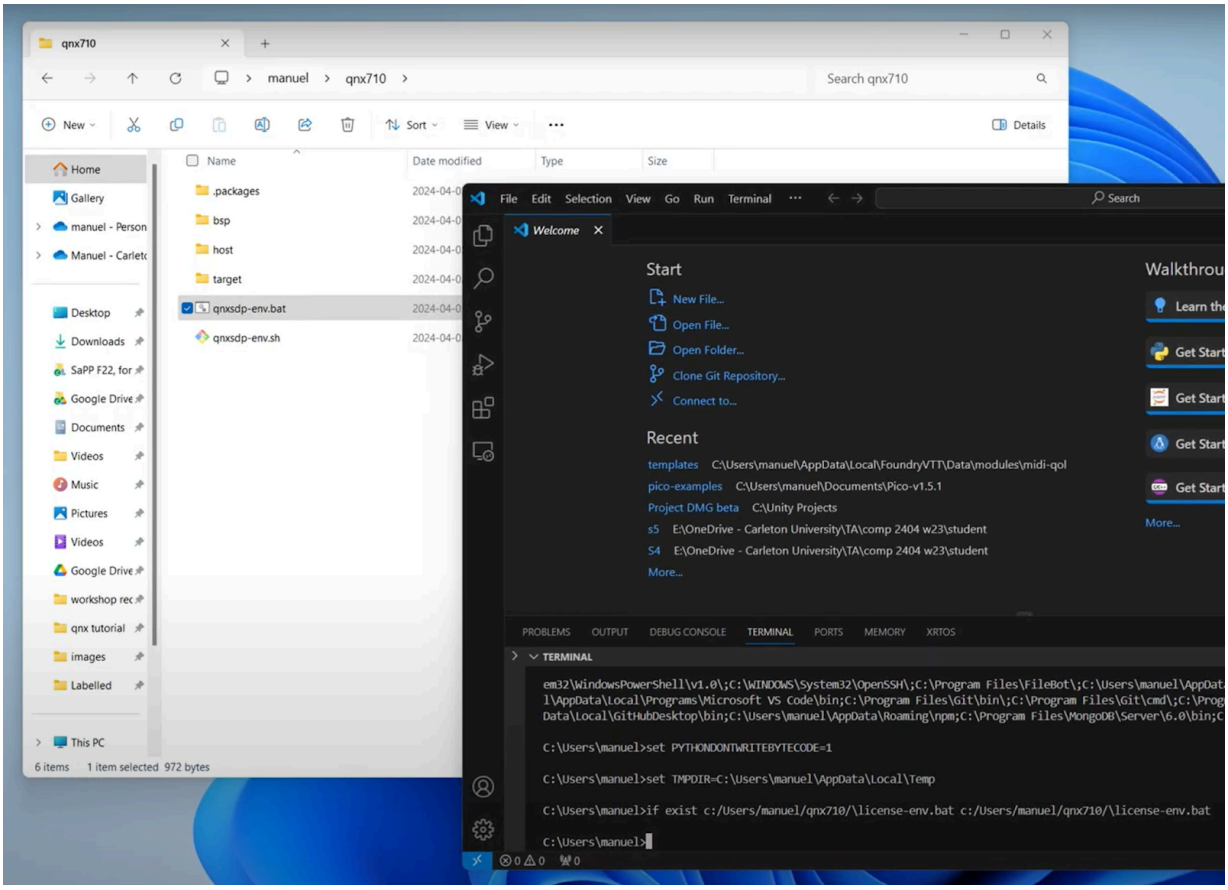
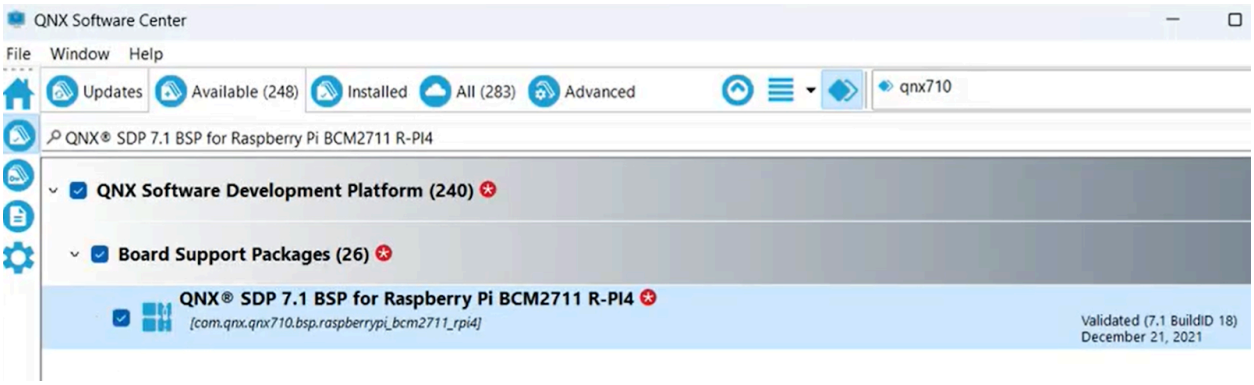


Figure: Ensure the Following QNX Module is Downloaded for BSP



3.6) WiFi Exploration & Connection

Download the needed files to allow Wifi to work on your device through the QNX Software Center. These will be various QNX modules which will add files to your QNX folder. The modules are found in the figures below. Module names are as follows[6]: QNX SDP 7.1 Wireless Driver for the Broadcom BCM4339 (wpa 2.9), QNX SDP 7.1 BSP for Raspberry Pi BCM2711 R-P14.

> NOTE: This is where we start to explore the "Over the Air" aspect of the project. Up until now you were able to perform data transfers between a PC and the Raspberry Pi using a serial cable. But now, we want to do this solely through WiFi. This begins more of an "exploration".

Figure: Ensure the Following are Downloaded

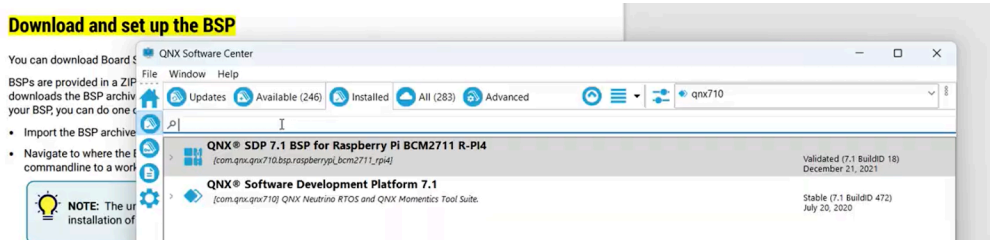
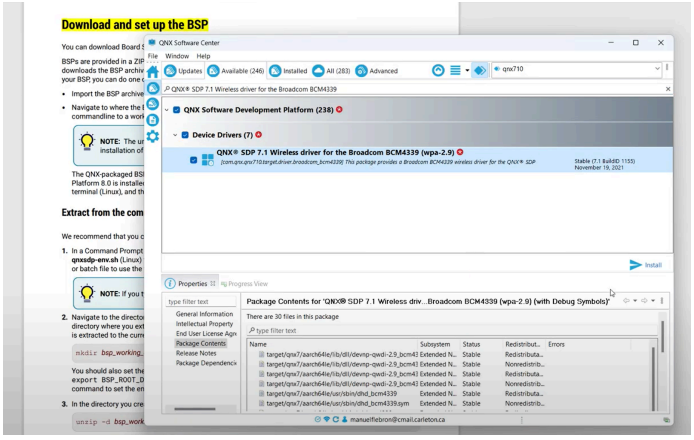


Figure: Downloading Driver for Wifi



NOTE: SSH is the means to establish a remote connection with the Raspberry Pi that is found on our WiFi network.

3.7) Build File Changes

In order to enable the features required for OTA updating, we have to make some modifications to our default build file downloaded from the QNX Software Center. In this report we will briefly highlight our major changes, but you can find the complete diff of our build file modifications here, and we have also linked a full in depth tutorial on page 2.

> NOTE: To access the .build file, go to the BSP folder you downloaded, click the "images" folder, and open rpi4.build (this is your build file).

3.7.1) WiFi

Firstly, we include tools like wpa_cli, wpa_passphrase, and wpa_supplicant in our QNX image, to negotiate WiFi protocols and connections for us.[6]

```
wpa_cli=wpa_cli-2.9
wpa_passphrase=wpa_passphrase-2.9
wpa_supplicant=wpa_supplicant-2.9
```

Then we can start our network drivers and tools on startup so our Pi is initialized for WiFi connections.

```
display_msg Starting WIFI Network driver...
     mount -T io-pkt -o
nvram=/etc/hotspot/nvram.txt,fw=/etc/hotspot/firmware.bin,clm_blob=/e
tc/hotspot/firmware.clm_blob,sdio_baseaddr=0xfe300000,sdio_irq=158,sd
io_verbose=1,drv_supp=3,key_delay=5
/lib/dll/devnp-qwdi-2.9_bcm4339-rpi4.so

     wpa_supplicant -Dqwdi -B -i bcm0 -c /etc/wpa_supplicant.conf
     dhclient -m -lf /dev/shmem/dhclient_wifi.leases -pf
/dev/shmem/dhclient_wifi.pid -nw bcm0
```

Optionally, we can define our WiFi connection details like the network name and password in our build file, so the Pi can attempt to connect to it on startup.

```
[uid=0 gid=0 perms=0777] /etc/wpa_supplicant.conf = {
ctrl_interface=/var/run/wpa_supplicant
update_config=1
network=\{
      ssid="INSERT WIFI NAME HERE"
      psk=INSERT PSK HERE
      key_mgmt=WPA-PSK
      proto=WPA2
      pairwise=CCMP
      group=CCMP
\}
}
```

Make sure to replace the ssid field with your network name and the psk with either your plaintext or hashed password. To hash your password, run the following:

```
wpa_passphrase your_network_name your_plaintext_password
```

3.7.2) SSH

Firstly, we include tools like ssh, ssh-keygen, and scp to enable wireless file transfer.[7]

```
/usr/sbin/sshd=sshd
/usr/bin/scp=scp
/usr/bin/ssh=ssh
/usr/libexec/sftp-server=${QNX_TARGET}/${PROCESSOR}/usr/libexec/sftp-
server
ssh-keygen
```

Below is a script to setup SSH configuration and tools on startup[7]

```
[perms=0755] /scripts/ssh-start.sh = {
#!/bin/sh

with_umask ()
\{
	local oldmask ret
	[ "$#" -ge 1 ] || return

	oldmask=$(umask) || return
	umask "$1" || return
	shift

	ret=0
	"$@" || ret=$?

	umask "$\{oldmask?\}"
	return "$\{ret:?\}"
\}

tools_ssh_keygen ()
\{
	local ssh_etcdir ssh_vardir type filename etclink varkey
	ssh_etcdir="/etc/ssh"
	ssh_vardir="/var/etc/ssh"

	for type in rsa
	do
	filename="ssh_host_$\{type:?\}_key"
	etclink="$\{ssh_etcdir?\}/$\{filename:?\}"
	varkey="$\{ssh_vardir?\}/$\{filename:?\}"

	if [ ! -f "$\{etclink:?\}" ]  # link currently invalid
	then
		ssh-keygen -t "$\{type:?\}" -N '' -f "$\{varkey:?\}" ||
return
	fi
	done
```

```
\}

tools_sshd ()
\{
      local x
      \{
    cp /etc/sshd_config /etc/ssh/
      with_umask 022 tools_ssh_keygen
      ssh_path=$(command -v sshd) && "$ssh_path"
      \} &
\}

# This script is intended to run once/boot only, at startup
running=/dev/shmem/ssh-start.run
[ -e $running ] && exit

tools_sshd;

touch $running
unset running

}
```

3.7.3) SD Card

Here we set up the SD card drivers and automount the contents of the card to /sdcard[8]. This is where the current image and any future images we want to boot to will be located.

```
display_msg Starting SDMMC driver for SD card (SDIO0)...
      devb-sdmmc-bcm2711 mem name=below1G sdio
addr=0xfe340000,irq=158,bs=bmstr_base=0xc0000000 disk name=sd
      devb-sdmmc-bcm2711 sdio addr=0xfe340000,irq=158 disk name=sd
      waitfor /dev/sd0t12
      mount -t dos /dev/sd0t12 /sdcard
      waitfor /sdcard
```

3.7.4) OTA Bash Script

The OTA update script below, will be located at /scripts/get_new_image.sh. It will replace the current image running on the board with the new image and then reboot the board.

```
[perms=0755] /scripts/get_new_image.sh = {
#!/bin/sh

# CMD Line Argument 1: Source host ex. root@8.8.8.8
# CMD Line Argument 2: Path to source file ex. C:/Users/me/file

rm -f /sdcard/ifs-rp4.bin
scp $1:$2 /sdcard/ifs-rpi4.bin
shutdown

}
```

3.7.5) Compile QNX Image

Compile the updated build file by performing the steps in the BSP User's Guide[1], "Build the BSP" section.

General Steps:

1. Navigate to the BSP folder.
2. Use make to compile the image.
3. The compiled image will be located in the 'images' folder

NOTE: If you have issues calling make, it is likely because the make file cannot find "qconfig.mk". You'll need to re-run "qnxsdp-env.bat" as done in part 6 of this section.

3.8) Make Raspberry Pi Bootable

Make the Pi bootable (ie. update the config file, save images to the SD card).

General Steps:

1. Delete the old temporary image created by the wizard.
2. Modify the config file that is on the SD card (this was created by the wizard) such that the config.txt file looks like the following:

```
arm_64bit=1
cmdline=startup.txt
device_tree=bcm2711-rpi-4-b.dtb
#enable_jtag_gpio=1
enable_uart=1
force_turbo=1
gpu_mem=16
max_framebuffers=2
kernel=u-boot.bin

Where kernel=YOUR_UBOOT_IMAGE_NAME.bin
```

NOTE: YOUR_UBOOT_IMAGE_NAME is your U-Boot image name.

3. Put U-Boot and the QNX image on the SD card.

3.9) Set up Serial Connection with the Raspberry Pi

Connect the Pi to a PC using a serial connection[9] (may need to install an application to interpret the serial connection like PUTTY for windows).

Figure: Serial Connection Interpreter Settings

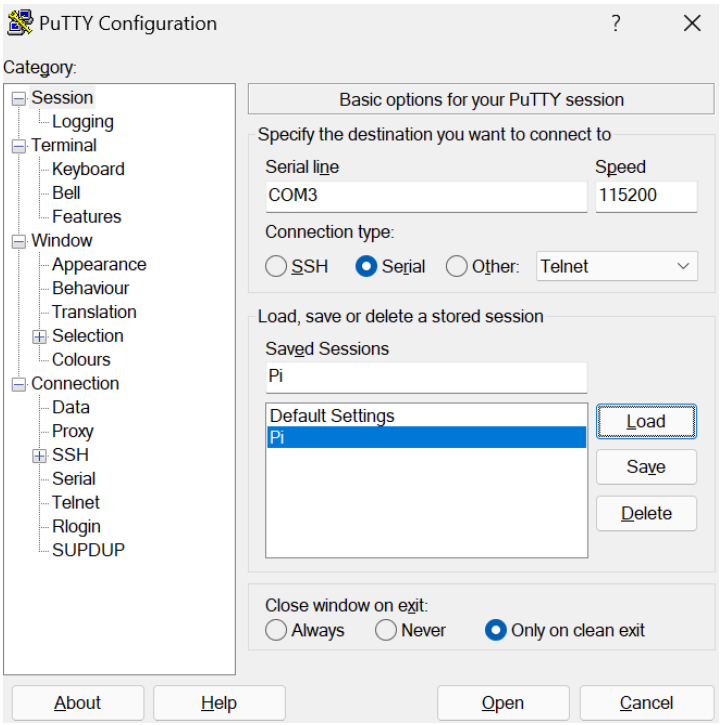
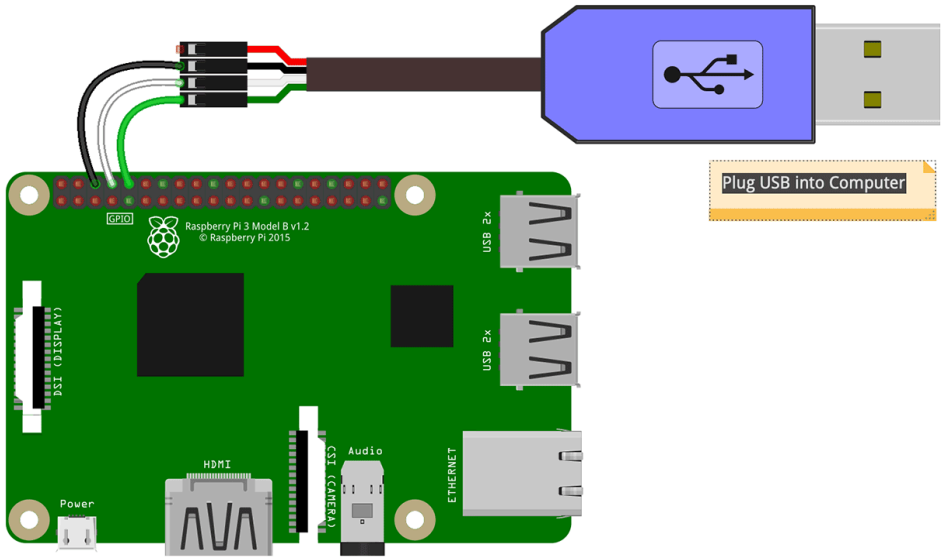


Figure: Serial Connections on the Pi[9]



3.10) Set up Auto-boot using U-Boot

Once the Raspberry Pi is bootable, interrupt the autoboot from within uboot and type the code below. This will cause U-Boot to autolaunch the QNX OTA image on subsequent boots. This example assumes the image name is ifs-rpi4.bin.

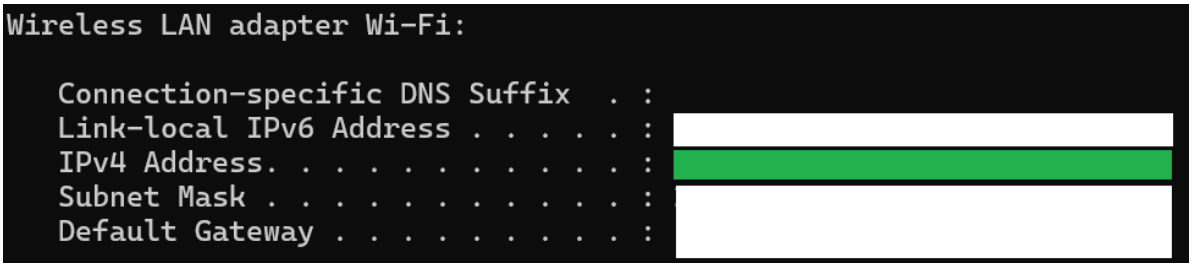
```
setenv sd_load "fatload mmc 0 0x80000 ifs-rpi4.bin\; go 0x80000\;"
setenv bootcmd "run sd_load\;"
saveenv
run bootcmd
```

3.11) How To Use The OTA Script

This final step will show you how to use the script in section 3.7.4 to perform an OTA update.

Step 1: Find the address, username, and password of the ssh server.

Figure: Run ‘ipconfig’ on the Windows 11 host - the green box shows the source address for your PC, one of the arguments to the bash script.



NOTE: If your windows user name is “manuel” then the address is:
manuel@YOURIPV4ADDRESS
Your password is your Windows 11 user password. If you do not have one defined this will not work.

Step 2: Find and copy the absolute address of the new image on your host.

Example:

E:\Desktop\updated_image_location\ifs-rpi4.bin

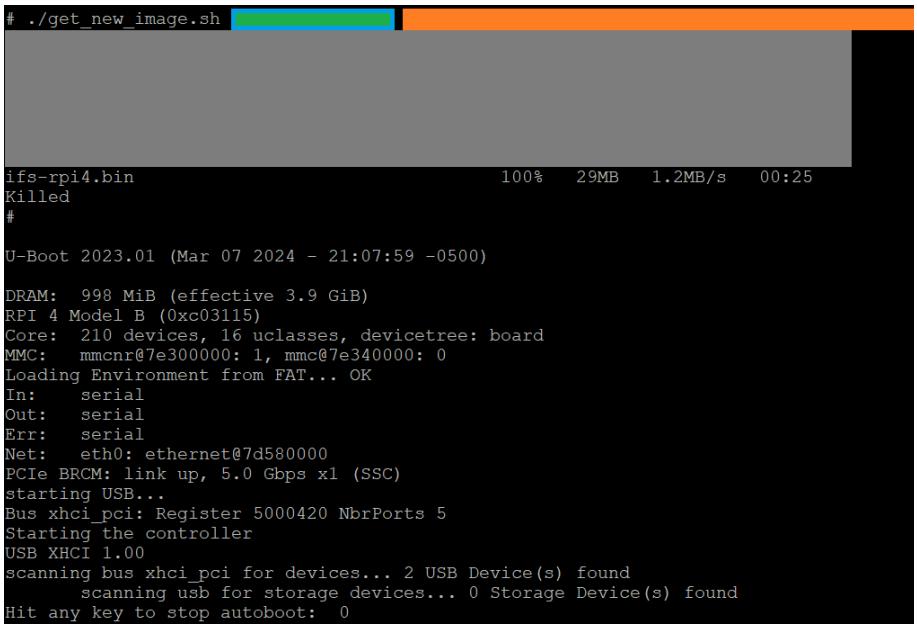
Then **replace** the \ with / as follows:

E:/Desktop/updated_image_location/ifs-rpi4.bin

Step 3: To perform an OTA update, run the bash script located at /scripts/get_new_image.sh as shown below. The first argument is the SSH address of your host and the second argument is the new image's absolute path.

```
./get_new_image.sh USERNAME@YOURIPV4ADDRESS C:/PATH/TO/NEW/IMAGE
```

Figure: Successfully running the OTA bash script (where the green is the address, orange is the path, gray is the password prompt):



As shown above, the new image is successfully downloaded onto the Pi, the Pi restarts, and then successfully autoboots to the new image.

Congratulations! You have successfully performed an OTA update on your Raspberry Pi! This process detailed all the steps we went through to come to our solution.

4. Future Work

Moving forward, there are several avenues we could explore to expand the reach of our project. One potential direction is to make our solution function seamlessly on all WiFi protocols as well as cellular networks. This expansion would require modifications to our scripts in order to be compatible with any necessary protocols as well as security measures needed to ensure the safe transmission of data.

With regards to expanding this project to support additional WiFi protocols and cellular networks, we will need to consider the specific characteristics and requirements of these networks. Here's a high level potential approach:

Additional WiFi Protocols:

- Identify the specific WiFi protocols to support (e.g., 802.11ac, 802.11ax).
- Update build files, scripts, and configurations to include support for these protocols by looking at RFC's and other documentation.
- Test the updated scripts to ensure compatibility and functionality.

Cellular Networks[10]:

- Research the cellular network standards and protocols used in the target environment (e.g., LTE, 5G).
- Obtain the needed hardware (e.g., cellular modem) and software (e.g., drivers) to enable cellular connectivity on the device.
    - A cellular modem issuing a USB is a common approach when dealing with Raspberry Pi's.
- Integrate the cellular modem with the Raspberry Pi device and configure the network settings.
- Develop scripts or applications to manage cellular connectivity, such as connecting to the network, handling data transmission, and managing network resources.

Security Considerations:

- Implement security measures for both WiFi and cellular networks, such as encryption, authentication, and secure data transmission protocols now that the scale has broadened (and generally it is good to consider security measures).

Testing and Validation:

- Conduct thorough testing of the updated system to ensure compatibility, and performance across different networks.

Adding OTA functionality to Non-QNX Based Raspberry Pi Systems

Our guide focuses on creating a QNX image that can update itself wirelessly on a Raspberry Pi 4. However, our solution can also be extended to provide OTA update functionality to any image whatsoever, by loading your target desired image alongside this OTA image. While running the target image, if we want to perform an update, we boot into the OTA image, transfer the new target image wirelessly onto our Pi, and then reboot into this new target image.

Replace the U-Boot script from section 3.7.10 with the following, making sure to replace TARGET_IMAGE_NAME and OTA_IMAGE_NAME:

Script:

```
target_image=fatload mmc 0 0x80000 TARGET_IMAGE_NAME\;
sd_load=if run target_image\;then go 0x80000\; else fatload mmc 0 0x80000
OTA_IMAGE_NAME.bin\; go 0x80000\; fi
setenv bootcmd run sd_load\;
saveenv
run bootcmd
```

This script searches for your target image and boots into it if it exists. If no target image exists, it boots into the OTA image.

When you want to perform an OTA update, delete your current target image, then restart. Since there is no existing target image, U-Boot will automatically boot into the OTA image where you can run the bash script to wirelessly transfer a new target image. You

will have to make modifications to the OTA bash script too, which you will have to do **before** you build the QNX OTA image in previous steps.

Script:

```
[perms=0755] /scripts/get_new_image.sh = {
#!/bin/sh

# CMD Line Argument 1: Source host ex. root@8.8.8.8
# CMD Line Argument 2: Path to source file ex. C:/Users/me/file

rm -f /sdcard/ifs-rpi4.bin
scp $1:$2 /sdcard/ifs-rpi4.bin
shutdown

}
```

Change the name “ifs-rpi4.bin” to the name of your desired payload. Depending on your Raspberry Pi image, you may have to add additional code in order to get all of the outdated files removed and to successfully transfer the new files.

Overall, there are various avenues we can explore using our project as a starting point.

5. Conclusion & Final Remarks

In this paper, we have detailed the development and implementation of an OTA update software solution tailored specifically for QNX-based embedded systems deployed on the Raspberry Pi platform. We described how to create scripts that will transfer a new payload to the board and place it in the correct location on the SD Card. Our script later ensures that the board is rebooted so that the update takes effect.

Overall, our project demonstrates the feasibility and benefits of using OTA update software solutions for managing and maintaining embedded systems, offering a scalable and efficient approach to ensuring software relevance in modern embedded systems.

All in all, this project can be improved to make very powerful OTA tools, including aspects such as cellular-connection based OTA or a general OTA update functionality to any Raspberry Pi system. We hope that this project has provided a good foundation for future endeavors in the world of real-time operating systems.

6. References

[1] QNX® Software Development Platform 8.0 BSP User's Guide Raspberry Pi 4 Board (2023rd ed.). (n.d.).

[2] Minitool Partition Wizard Free 12.8. Best Free Partition Manager for Windows | MiniTool Partition Wizard Free. (n.d.). https://www.partitionwizard.com/free-partition-manager.html

[3] Raspberry Pi 4 Model B Das U-Boot | danmc.net. (2023, February 4). https://danmc.net/posts/raspberry-pi-4-b-u-boot/

[4] IONOS editorial team. (2023, October 6). How to use the Windows 11 SSH client. IONOS Digital Guide. https://www.ionos.ca/digitalguide/server/configuration/windows-11-ssh/

[5] QNX 7.1 on Raspberry PI 4 - Research Computing Services. (2021, July 29). Research Computing Services. https://carleton.ca/rcs/qnx/installing-qnx-on-raspberry-pi-4/

[6] BlackBerry QNX Wifi Support. (n.d.). https://www.qnx.com/developers/articles/rel_6836_0.html#WiFiSupport

[7] BlackBerry QNX SSH Guide. (n.d.). https://www.qnx.com/developers/docs/7.1/index.html#com.qnx.doc.neutrino.utilities/topic/s/ssh.html

[8] QNX® Software Development Platform 7.1 Board Support Package for the Raspberry Pi 4 Board (Model B): Release Notes. (n.d.). https://www.qnx.com/developers/articles/rel_6836_0.html#Issues

[9] Adafruit's Raspberry Pi Lesson 5. Using a console cable. (2012, December 18). Adafruit Learning System. https://learn.adafruit.com/adafruits-raspberry-pi-lesson-5-using-a-console-cable/connect-the-lead

[10] Using 4G LTE wireless modems on a Raspberry Pi | Jeff Geerling. (2022, January 20). https://www.jeffgeerling.com/blog/2022/using-4g-lte-wireless-modems-on-raspberry-pi