COMP 4900E - Real Time Operating Systems

Carleton University

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

Technical Tutorial Manual

April 10, 2024

Anish Phadnis: 10201491

Courtney Thirukumaran: 101191547

Emmanuel Richmond: 101108167

Huseyin Kabasakal: 101156317

Manuel Osvaldo Lebron Flores: 100958229

Professor

Jun Huang

Table of Contents

Introduction

This tutorial is meant to be used as a step-by-step guide for anyone trying to recreate our project. We will show you how to add OTA update functionality to a Raspberry Pi using QNX. Please see our GitHub, where you will find a video version of the guide as well as a report. The video guides have also been combined and uploaded to YouTube.

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

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

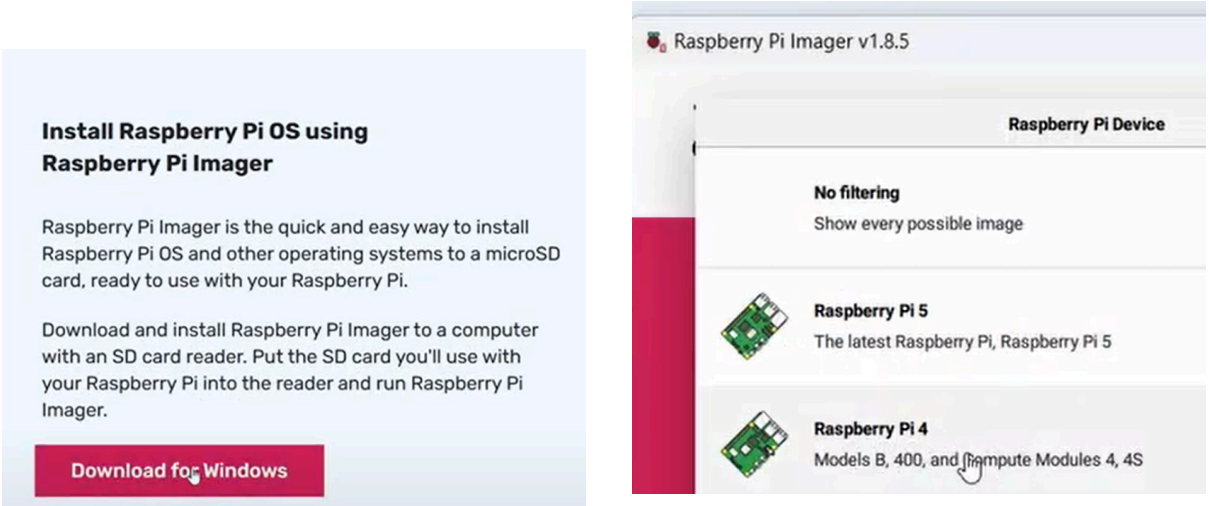
Setup-Prerequisites:

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

Step 1) Set up Raspberry Pi

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.
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 to format the card.

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



Step 2) Compile U-Boot

Download the needed files to compile U-Boot on the Raspberry Pi 4 (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

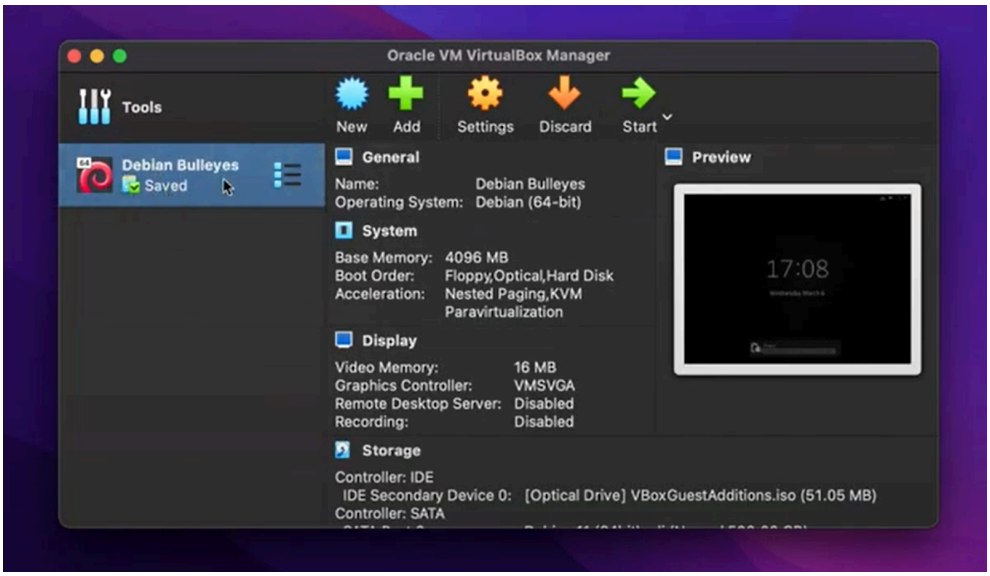


Figure: Copying Boot Files from Linked Documentation to Compile on Debian

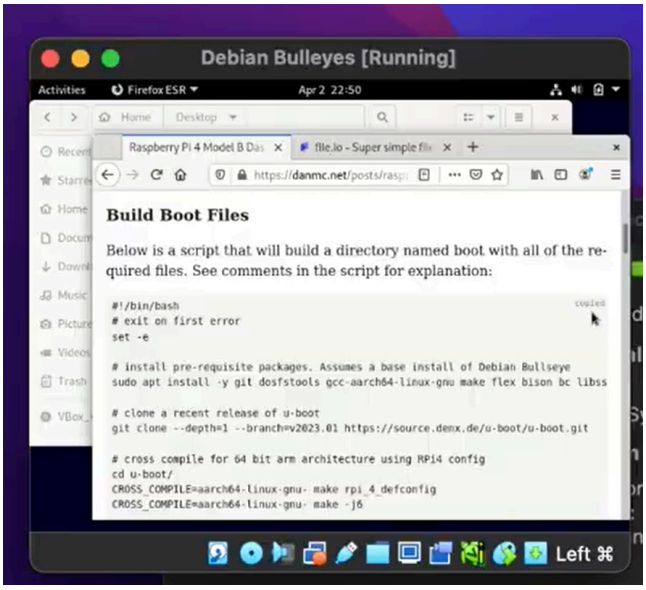


Figure: Running the Copied Bash File on VM

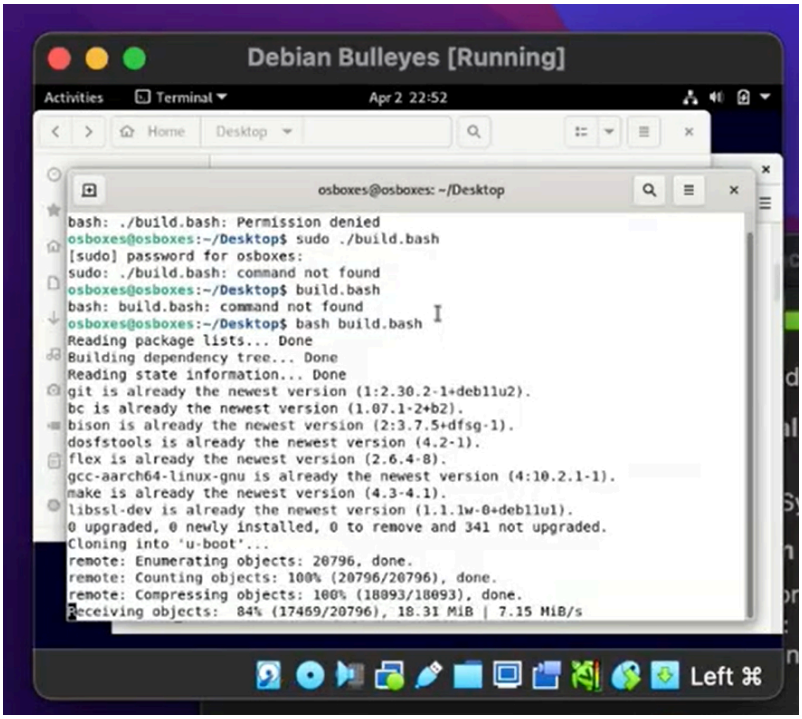
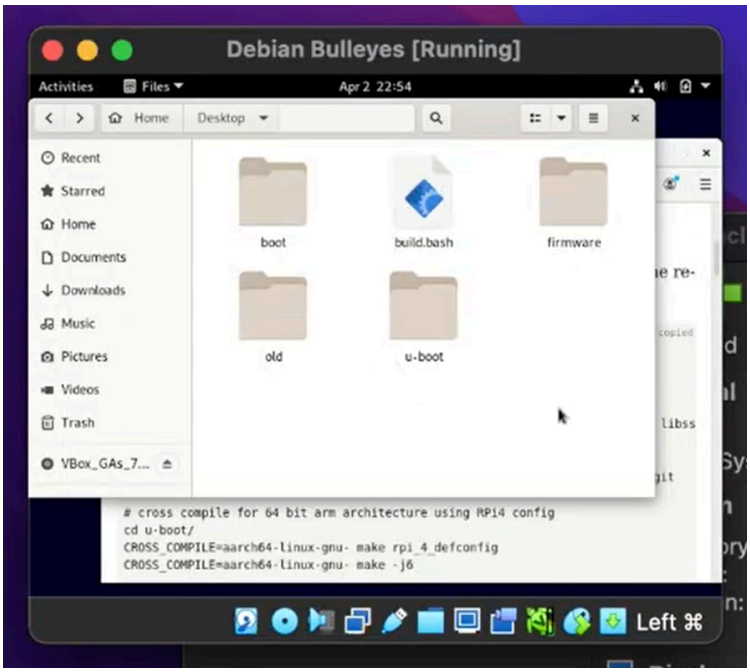


Figure: Boot, Firmware & U-Boot Folders Created



Step 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

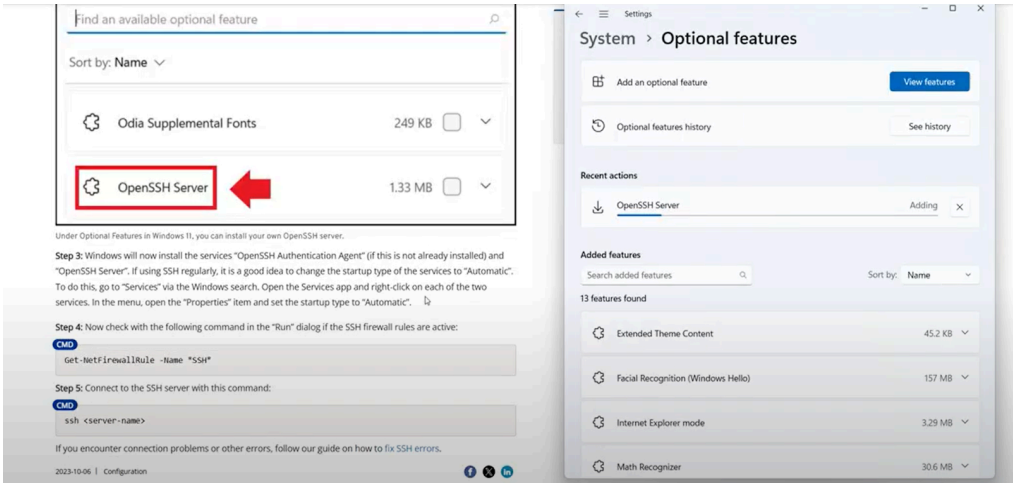
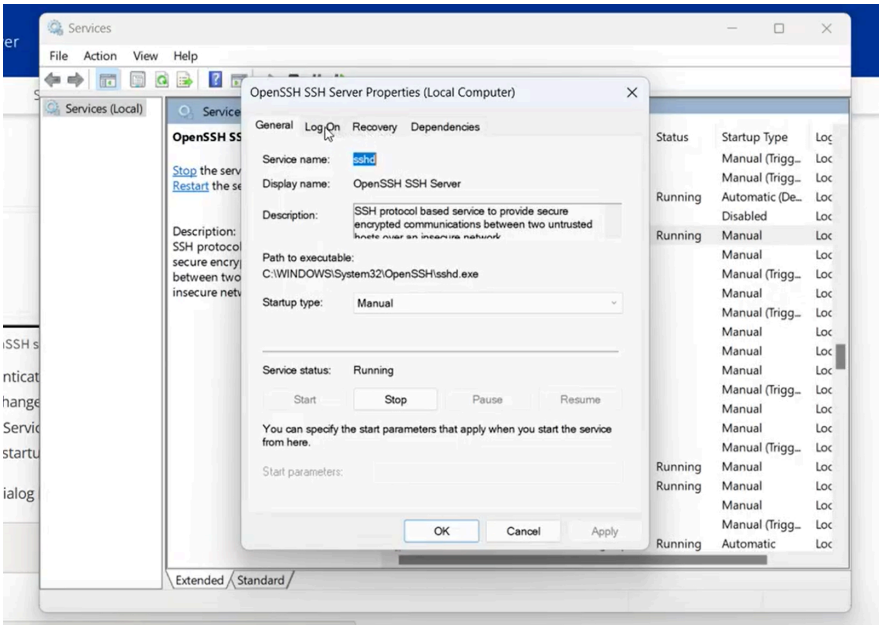


Figure: Running OpenSSH Server and Authentication Agent on Windows



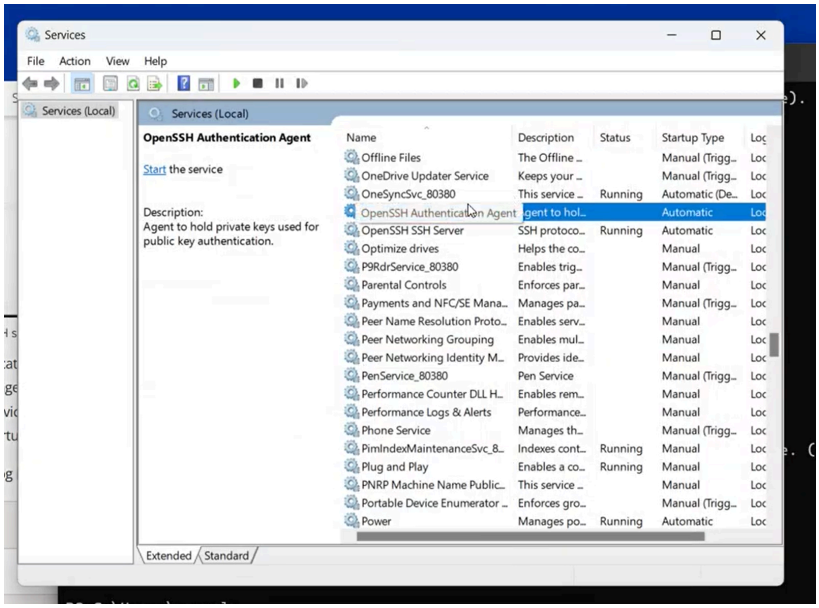


Figure: Ensuring SSH Successfully Connects

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\manuel> ipconfig | select-string  ('(\s)+IPv4.+\s(?<IP>(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}))(\s)*') -AllMatc
hes | %{ $_.Matches } | % { $_.Groups["IP"]} | %{ $_.Value }
192.168.2.16
192.168.56.1
PS C:\Users\manuel> ssh manuel@192.168.56.1
manuel@192.168.56.1's password:
```

NOTE: Have a password for your PC!

1. Download QNX Software Center to retrieve QNX SDP 7.1 (the original version of QNX that we will update to 8.0 later in this tutorial). You can follow Carleton University's QNX on Raspberry Pi tutorial under "Install software on development computer" section to do so.
    a. NOTE: Licensing will be needed. This can be requested after you have downloaded and created an account with Blackberry QNX.

Step 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 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]

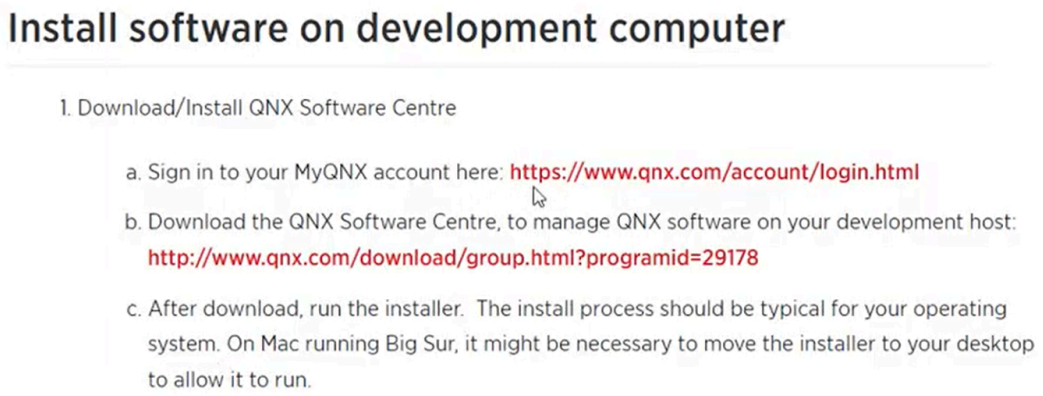


Figure: Login/Create Account with QNX Center

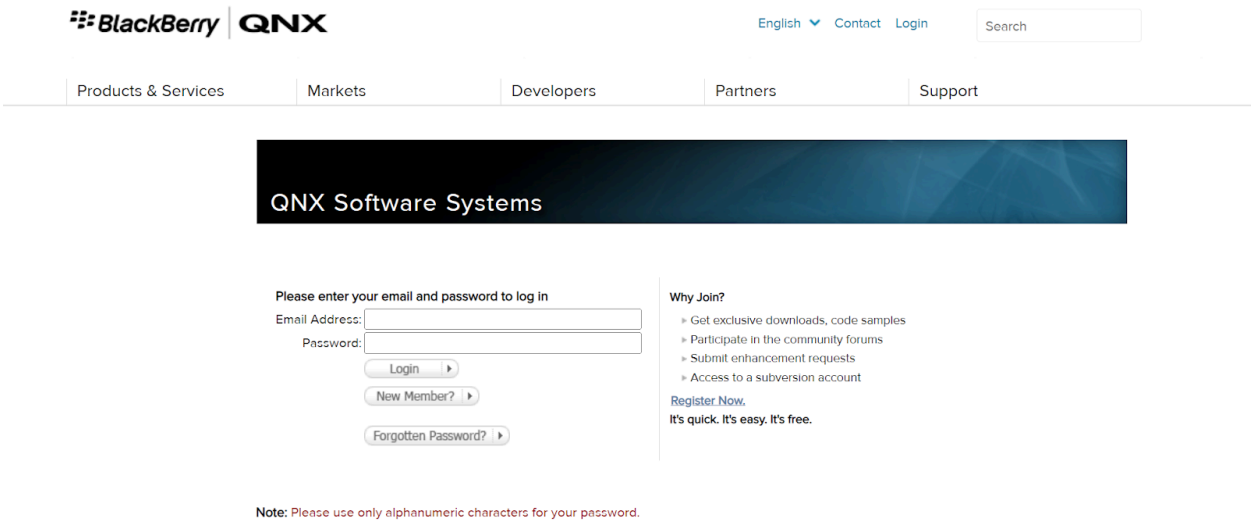


Figure: Under the Download Section on the Website Click the Following

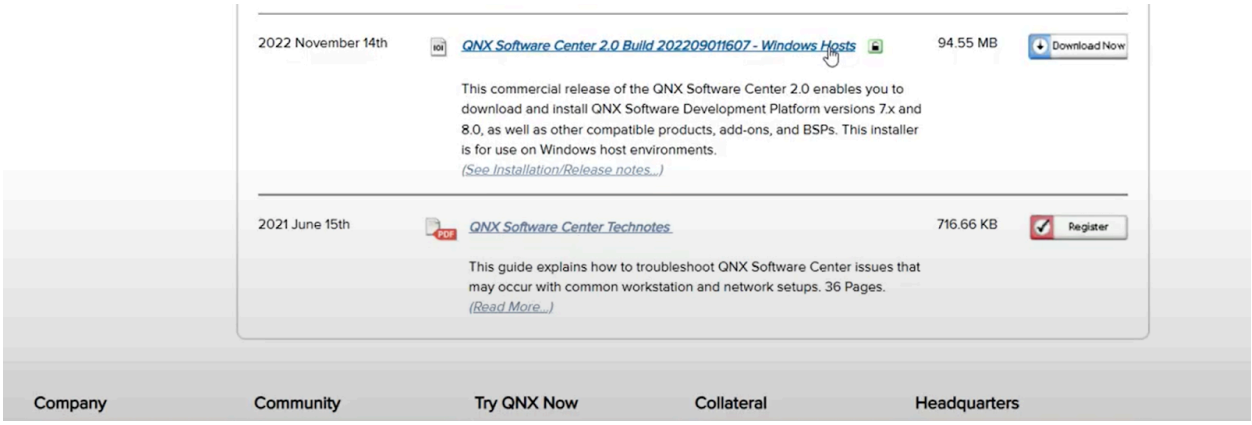
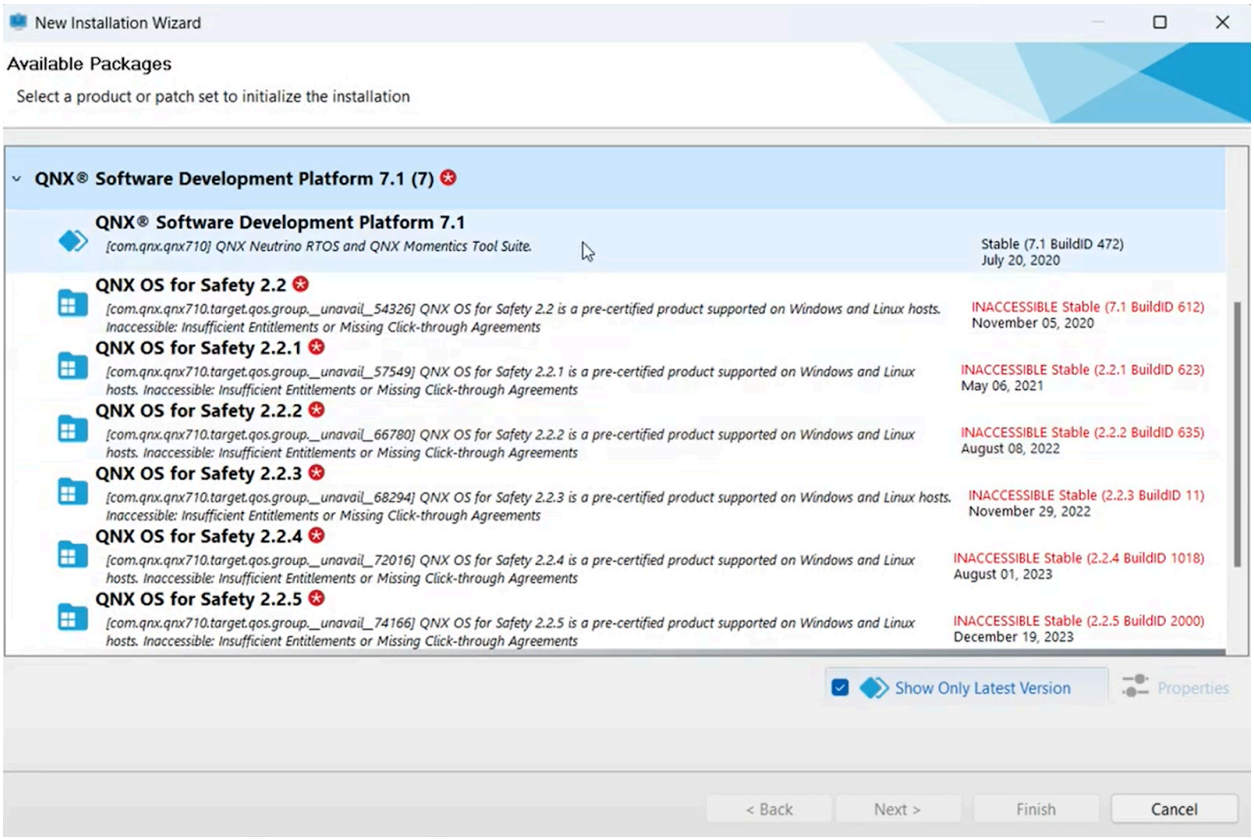


Figure: QNX Software Center Opens & Should Activate 7.1 License



Step 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.

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

Figure: Editing the necessary environment variables.

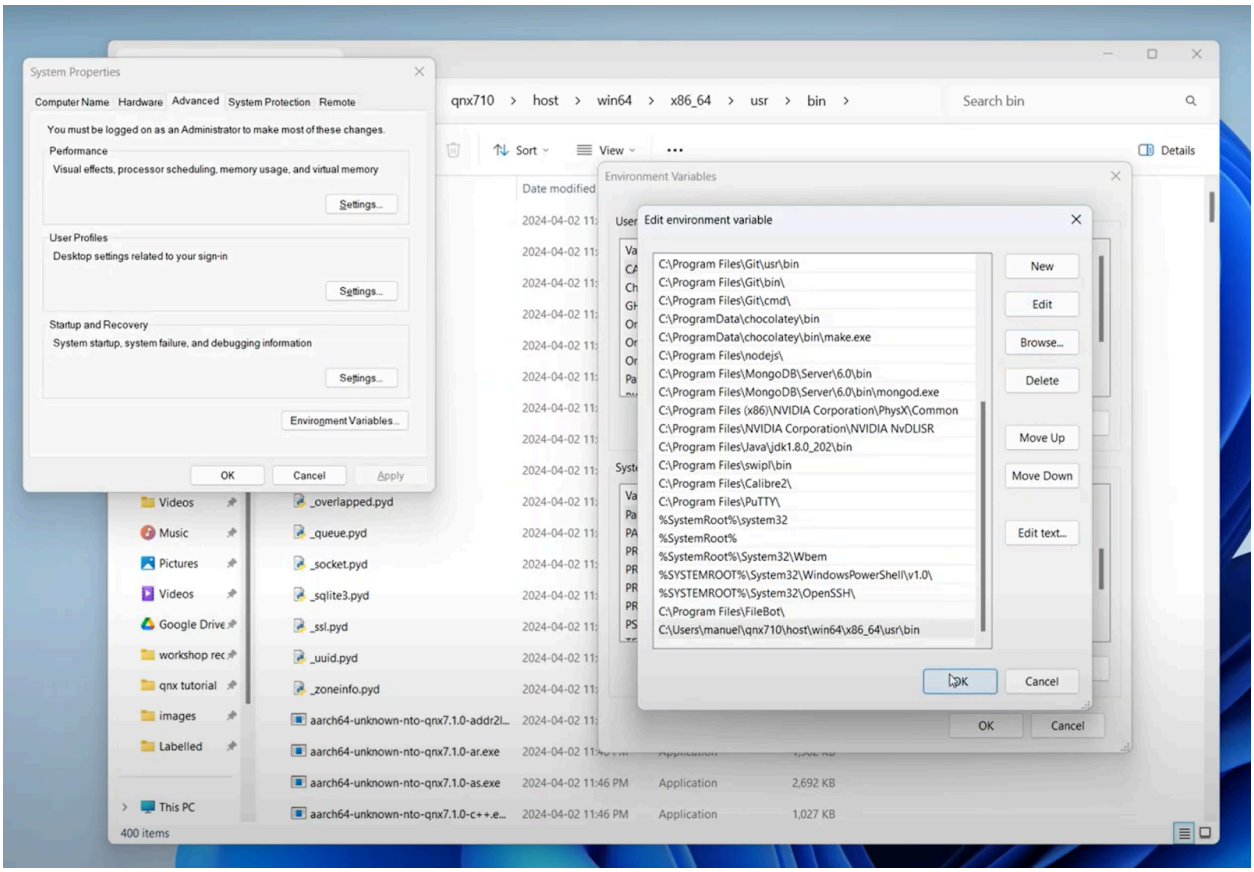


Figure: Running ‘qnxsdp-env.bat’ to define necessary QNX specific environment variables

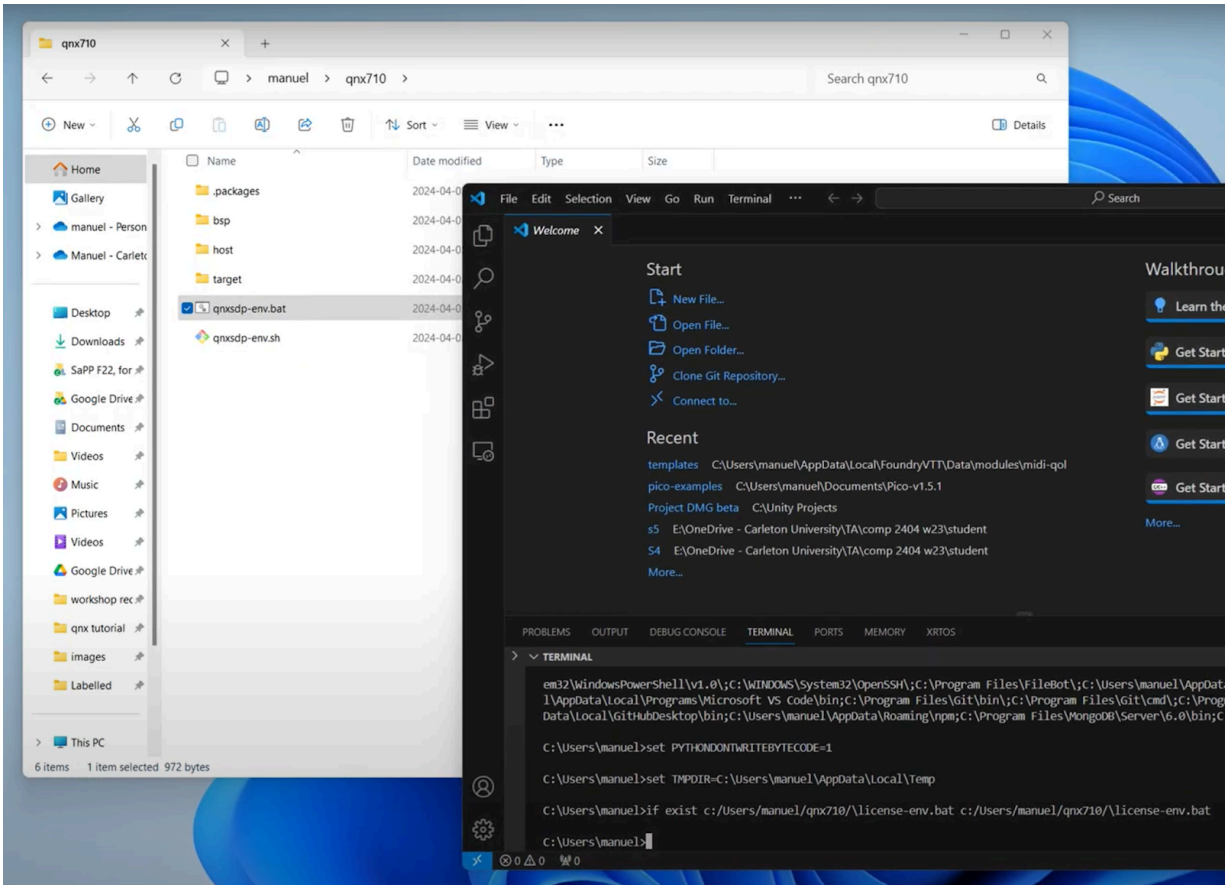
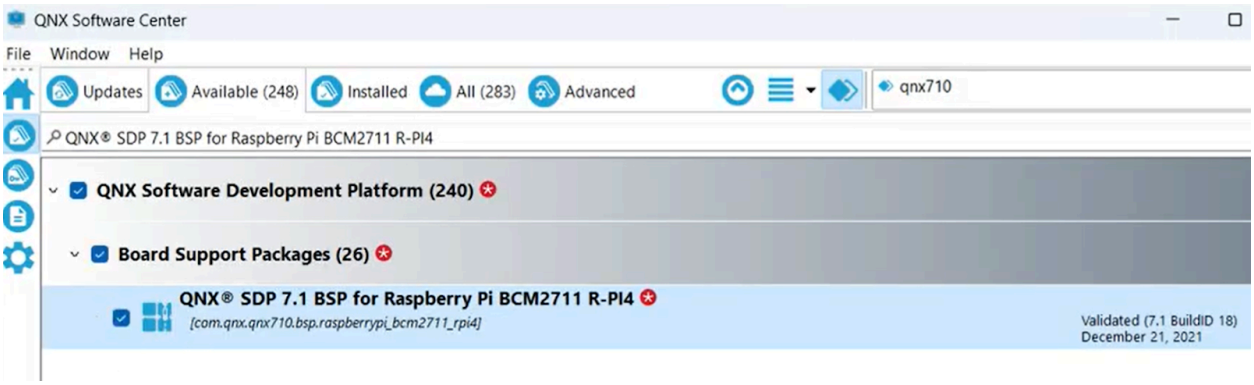


Figure: Ensure the Following QNX Module is Downloaded for BSP



Step 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: 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

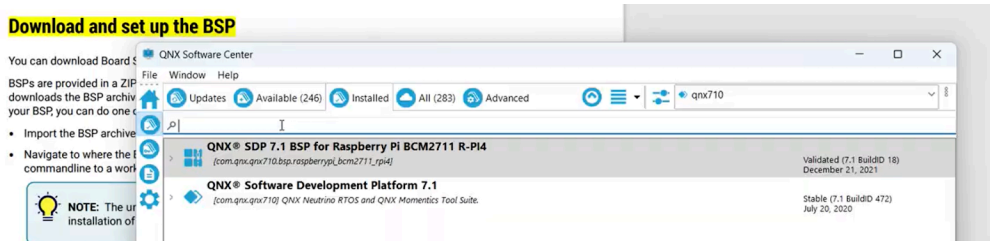
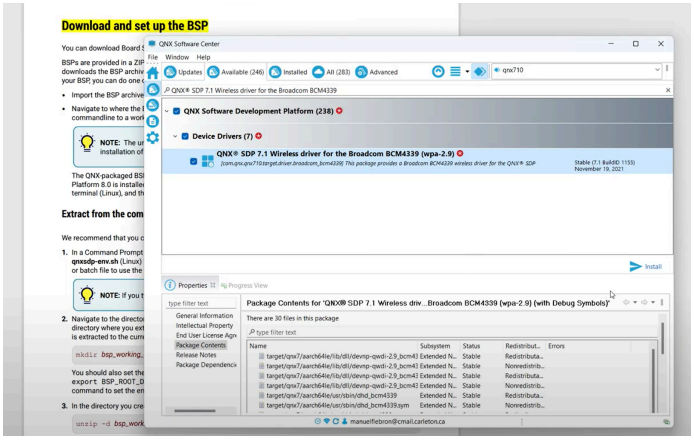


Figure: Downloading Driver for Wifi



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

Step 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. You can also find [the complete diff of our build file modifications here](#).

> 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).

Step 7.1) Set up SSH

- Step 7.1.1) Add a Working SSH Script
- Step 7.1.2) Add Auto Start SSH Script functionality
- Step 7.1.3) Add Other SSH Fixes To Build file (outside of SSH script)

Step 7.1.1) Add a Working SSH Script

- Step 7.1.1a) Add SSH Script To Build File
- Step 7.1.1b) Make fixes to SSH Script

Step 7.1.1a) Add SSH Script To Build File

Follow the instructions from QNX to add an SSH script to the build file: https://www.qnx.com/developers/docs/7.1/index.html#com.qnx.doc.neutrino.utilities/topic/s/ssh.html

```
#########################################################################
######
## sshd support
#########################################################################
######
/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
mkdir
chmod
touch
waitfor
```

```
[uid=0 gid=0 perms=0644 search=${QNX_TARGET}/etc/ssh]
/etc/ssh/ssh_known_hosts=ssh_known_hosts

[uid=0 gid=0 perms=0644] sshd_config={
Protocol 2
LoginGraceTime 600
PermitRootLogin yes             # NOT SECURE, FOR DEBUGGING
PermitEmptyPasswords yes        # NOT SECURE, FOR DEBUGGING
UsePrivilegeSeparation no       # NOT SECURE, FOR DEBUGGING
Subsystem       sftp    /usr/libexec/sftp-server
}

[perms=0640] /etc/shadow = {
root:E4m.vvfkKBbRo:1231323780:0:0
}

[perms=0644] /etc/passwd = {
root:x:0:0:Superuser:/root:/bin/sh
sshd:x:15:6:sshd:/var/chroot/sshd:/bin/false
}

/root/.profile = {
PS1='$(hostname)#'
export SYSNAME=nto
export TERM=xterm
#export PATH=/proc/boot:/sbin:/bin:/usr/bin:/opt/bin/sbin:/usr/sbin
#export
LD_LIBRARY_PATH=/proc/boot:/lib:/usr/lib:/lib/dll:/opt/lib:/lib/dll/p
ci
}

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

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

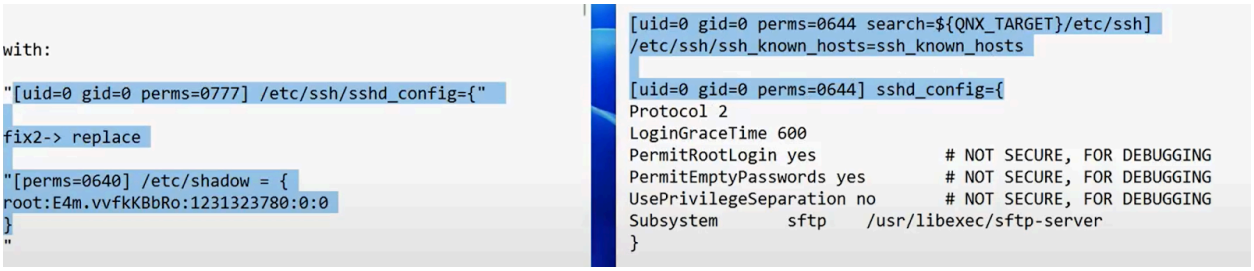
```
    oldmask=$(umask) || return
    umask "$1" || return
    shift

    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 /proc/boot/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

}

[perms=0755 uid=0 gid=0 type=dir] /var/chroot/sshd
[type=link] /etc/ssh = /dev/shmem
[type=link] /var/etc/ssh = /dev/shmem
```

Step 7.1.1b) Make fixes to SSH Script

Fix 1:

Figure: Replacing Code (left is code that needs to replace the highlighted code on the right)

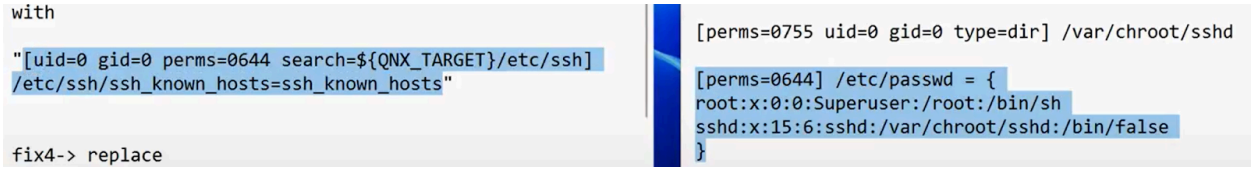


Fix 2:

Figure: Replacing Code (left is code that needs to replace the highlighted code on the right)

Fix 3:

Figure: Replacing Code (left is code that needs to replace the highlighted code on the right)

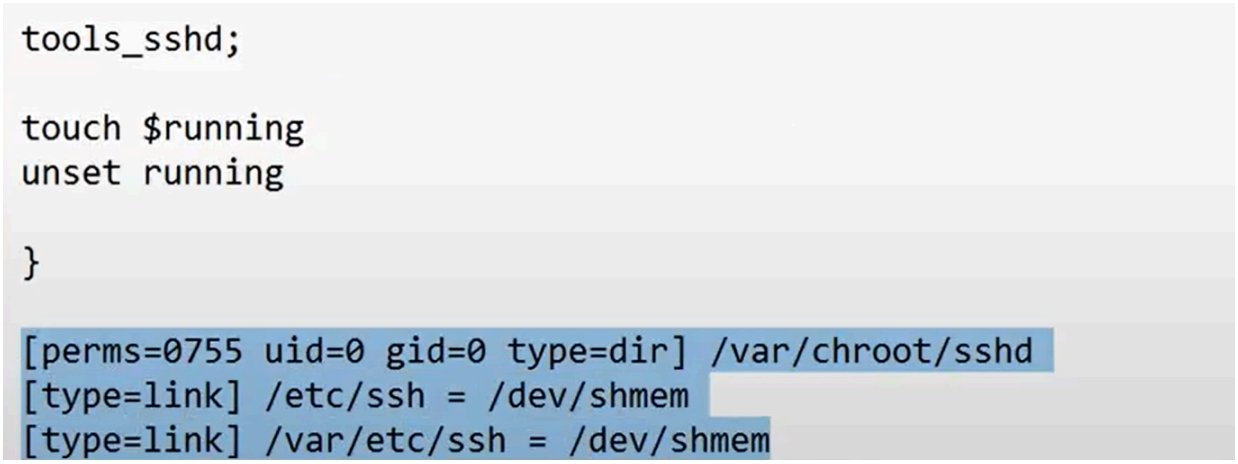


Fix 4:

Figure: Replacing Code (left is code that needs to replace the highlighted code on the right)

Fix 5:

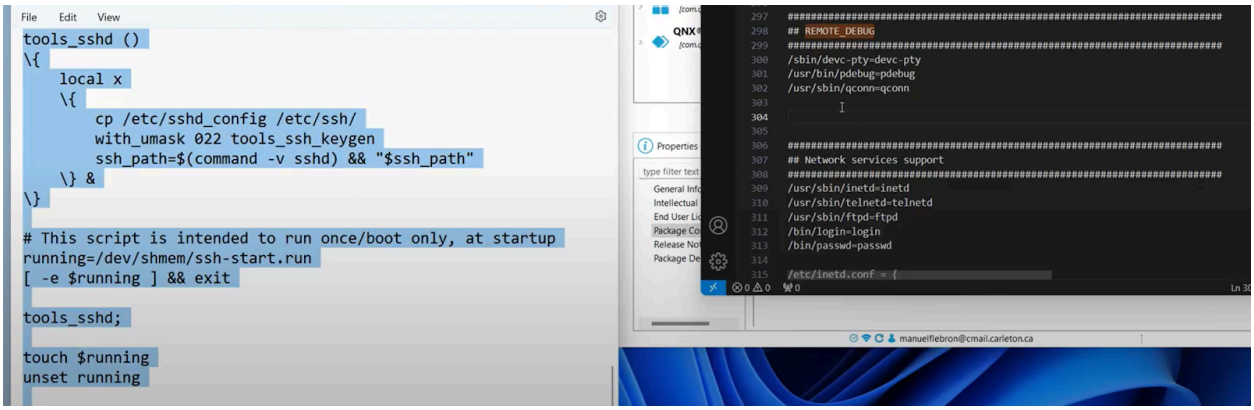
Figure: Remove the Highlighted Code



Step 7.1.2) Add Auto Start SSH Script functionality

Adding a script into the build file to auto start SSH. Add the script to the network driver section.

Figure: Copying Over the Fixed Code



Step 7.1.3) Add Other SSH Fixes To Build file (outside of SSH script)

Auto start the scp/ssh session using some custom code. This should be added to the Network driver section in the .build file. The line that needs to be added is: **/scripts/ssh-start.sh**

Figure: Visual Depiction of Added Code

```
#############################################################################
## Network driver
#############################################################################
display_msg Starting Network driver...
io-pkt-v6-hc
mount -T io-pkt /lib/dll/devnp-genet.so
if_up -p genet0
dhclient -m -lf /dev/shmem/dhclient.leases -pf /dev/shmem/dhclient.pid -nw genet0

/scripts/ssh-start.sh
```

Step 3: Adding other SSH fixes outside of the actual SSH script within the .build file. To find lines of code to replace within the .build file, probably easiest to CTRL+F.

Fix 1:

Figure: Replace Unhighlighted Code in .build file with Highlighted Code

```
"#[type=link] /var/run/wpa_supplicant=/dev/shmem"
```

```
[type=link] /var/run/wpa_supplicant=/dev/shmem"
```

Fix 2:

Figure: Replace Unhighlighted Code in .build file with Highlighted Code

```
"[type=link] /etc/ssh = /dev/shmem
[type=link] /var/etc/ssh = /dev/shmem"
```

```
"[type=link] /etc/resolv.conf=/tmp/resolv.conf"
```

Fix 3:

Figure: Add the Unhighlighted Code under the Highlighted Code found in the .build file

```
[type=link] /etc/resolv.conf=/tmp/resolv.conf
[type=link] /etc/ssh = /dev/shmem
[type=link] /var/etc/ssh = /dev/shmem
```

Fix 4:

Figure: Add the highlighted code between the unhighlighted lines found in the .build file

```
"/usr/bin/ssh=ssh"
```

```
"/usr/sbin/sshd=sshd
/usr/sbin/tcpdump=tcpdump"
```

Fix 5:

Figure: Add the highlighted code after the unhighlighted lines found in the .build file

```
"
/etc/sshd_config = {
Protocol 2
LoginGraceTime 600
PermitRootLogin yes             # NOT SECURE, FOR DEBUGGING
PermitEmptyPasswords yes        # NOT SECURE, FOR DEBUGGING
UsePrivilegeSeparation no       # NOT SECURE, FOR DEBUGGING
Subsystem       sftp    /usr/libexec/sftp-server
}
"
```

```
"/etc/inetd.conf = {
telnet  stream  tcp nowait  root    /usr/sbin/telnetd   telnetd
ftp     stream tcp nowait   root    /usr/sbin/ftpd
in.ftpd -l
}"
```

Fix 6:

Figure: Replace the unhighlighted code in the .build file with the highlighted code

```
"export
LD_LIBRARY_PATH=/proc/boot:/lib:/usr/lib:/lib/dll:/lib/dll/pci"
```

```
"export LD_LIBRARY_PAT   ot:/lib:/usr/lib:/lib/dll:/lib/dll/pci"
```

Fix 7:

Figure: Replace the unhighlighted code in the .build file with the highlighted code

```
"[uid=0 gid=0 type=dir dperms=0755] /etc"
```

```
"#[uid=0 gid=0 type=dir dperms=0755] /etc"
```

Step 7.2): Enable Wifi

Step 7.2.1) Add QNX Wifi Build Changes
Step 7.2.2) Add Fixes to QNX's Wifi Build Changes

Step 7.2.1): Add QNX Wifi Build Changes

Enable Wifi by following THE QNX Wifi Support Guide. You will need to uncomment some lines from the build file as well as add your Wifi name and password to the build file. In our example, we hashed the Wifi password prior to this tutorial but you can use a plain text version as well. We will now detail the steps we took (ie. fixes, etc).

Fix 1:

We want to add the Wifi driver to our .build file.

Figure: Replace the unhighlighted code in the .build file with the highlighted code

```
"#/lib/dll/devnp-qwdi-2.9_bcm4339-rpi4.so=devnp-qwdi-2.9_bcm4339-
rpi4.so"
```

```
"/lib/dll/devnp-qwdi-2.9_bcm4339-rpi4.so=devnp-qwdi-2.9_bcm4339-rpi4.so"
```

Fix 2:

We want to add Wifi firmware files to our .build file.

Figure: Replace the unhighlighted code in the .build file with the highlighted code

```
"#/etc/hotspot/firmware.bin=
${QNX_TARGET}/etc/firmware/firmware.bcm43455.default.bin
#/etc/hotspot/nvram.txt=
${QNX_TARGET}/etc/firmware/nvram.bcm43455.rpi4.default.txt
#/etc/hotspot/firmware.clm_blob=
${QNX_TARGET}/etc/firmware/bcm43455.default.clm_blob"
```

```
"/etc/hotspot/firmware.bin=
${QNX_TARGET}/etc/firmware/firmware.bcm43455.default.bin
/etc/hotspot/nvram.txt=
${QNX_TARGET}/etc/firmware/nvram.bcm43455.rpi4.default.txt
/etc/hotspot/firmware.clm_blob=
${QNX_TARGET}/etc/firmware/bcm43455.default.clm_blob"
```

Fix 3:

We need to add wpa_cli, wpa_passphrase and wpa_supplicant.

Figure: Replace the unhighlighted code in the .build file with the highlighted code

```
"#wpa_cli=wpa_cli-2.9
#wpa_passphrase=wpa_passphrase-2.9
#wpa_supplicant=wpa_supplicant-2.9"
```

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

Fix 4:

Start the Wifi driver which will create a Wifi network interface as “bcm0”

Figure: Replace the unhighlighted code in the .build file with the highlighted code

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

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

Fix 5:

Start wpa_supplicant which can be done in the startup script if the wpa_supplicant.conf is pre-defined, or at run-time)

Figure: Replace the unhighlighted code in the .build file with the highlighted code (forgot to get highlighted)

```
"#wpa_supplicant -Dqwdi -t -Z100 -ibcm0 -c /tmp/wpa_supplicant.conf -
C/var/run/wpa_supplicant/ -g/var/run/wpa_supplicant_global &"
```

```
wpa_supplicant -Dqwdi -B -i bcm0 -c /etc/wpa_supplicant.conf
```

Fix 6:

Add Wifi name and password to the .build. NOTE: In the updated code, replace the ssid and psk with your own Wifi name and password. In our case, our psk was hashed using wpa_passphrase but you can use a plaintext password too.

Figure: Replace the unhighlighted code in the .build file with the highlighted code

```
"

esac
}
"
```

```
"
}

[uid=0 gid=0 perms=0777] /etc/wpa_supplicant.conf = {
ctrl_interface=/var/run/wpa_supplicant
update_config=1
network=\{
        ssid="Free wifi for everyone!"
        psk=
701534535ba18e4dd55dbad511d295b887c036ff7ee9fcace545579133264894
        key_mgmt=WPA-PSK
        proto=WPA2
        pairwise=CCMP
        group=CCMP
\}
}
"
```

Step 7.2.2) Add Fixes to QNX's Wifi Build Changes

If you attempt to enable Wifi with the QNX guide from above, it will fail. Use the following custom fix we found to get it working.

Fix 1:

Replace:

"wpa_supplicant -Dqwdi -t -Z100 -ibcm0 -c /tmp/wpa_supplicant.conf -C/var/run/wpa_supplicant/ -g/var/run/wpa_supplicant_global &"

With:

"wpa_supplicant -Dqwdi -B -i bcm0 -c /etc/wpa_supplicant.conf"

Fix 2:

Replace:

"#dhclient -m -lf /dev/shmem/dhclient.leases -pf /dev/shmem/dhclient.pid -nw bcm0"

With:

"dhclient -m -lf /dev/shmem/dhclient_wifi.leases -pf /dev/shmem/dhclient_wifi.pid -nw bcm0"

Step 7.3) Add SD Card Functionality

Step 7.3.1) Add fixes to SD card driver (known issue)
Step 7.3.2) Auto Mount SD Card

Step 7.3.1) Add fixes to SD card driver (known issue)

See the QNX Guide on fixes to the SD card driver:
(https://www.qnx.com/developers/articles/rel_6836_0.html#Issues)

Specifically See:
“Workaround: Use "devb-sdmmc-bcm2711 sdio addr=0xfe340000,irq=158 disk name=sd" on the BCM 2711ZPKFSB06C0T SoC based boards.”

Fix 1:

Figure: Add the the highlighted code under the highlighted code in the .build file

```
"devb-sdmmc-bcm2711 sdio addr=0xfe340000,irq=158 disk name=sd"
```

```
"devb-sdmmc-bcm2711 mem name=below1G sdio addr=0xfe340000,irq=
158,bs=bmstr_base=0xc0000000 disk name=sd"
```

Step 7.3.2) Auto Mount SD Card

Add a few lines to the build file to get the SD card to auto mount.

Fix 1:

Figure: Last Three Lines of Code are what need to be added to .build file

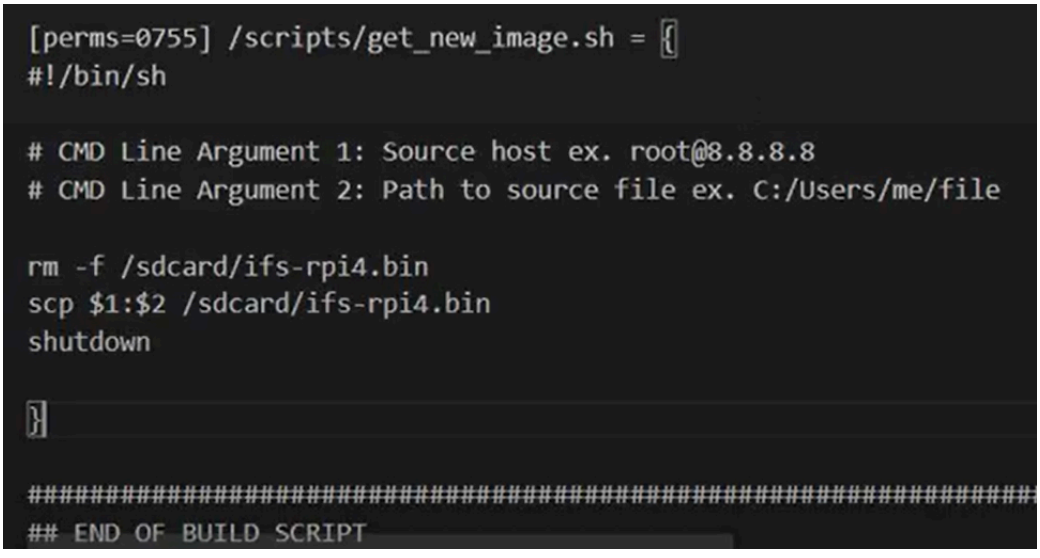
```
#########################################################################
## SD memory card driver
## Note: "bmstr_base" option is required for busmaster memory
## address translation
#########################################################################
display_msg Starting SDMMC driver for SD card (SDIO0)...
devb-sdmmc-bcm2711 mem name=below1G sdio addr=0xfe340000,irq=158,bs=bmstr_base=0xc0000000 disk name=
devb-sdmmc-bcm2711 sdio addr=0xfe340000,irq=158 disk name=sd
waitfor /dev/sd0t12
mount -t dos /dev/sd0t12 /sdcard
waitfor /sdcard
```

Step 7.4) Add an 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

}
```

Figure: Adding Build Script to End of Build File



Step 7.5) Compile QNX Image

Compile the updated build file by performing the steps in the BSP User's Guide, "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.

Step 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.

Step 9) Set up Serial Connection with the Raspberry Pi

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

Figure: Serial Connection Interpreter Settings

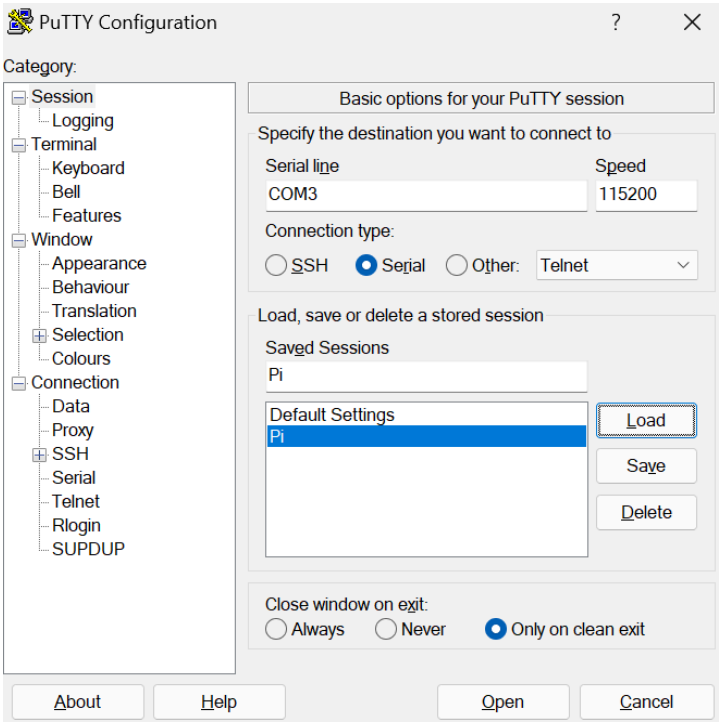
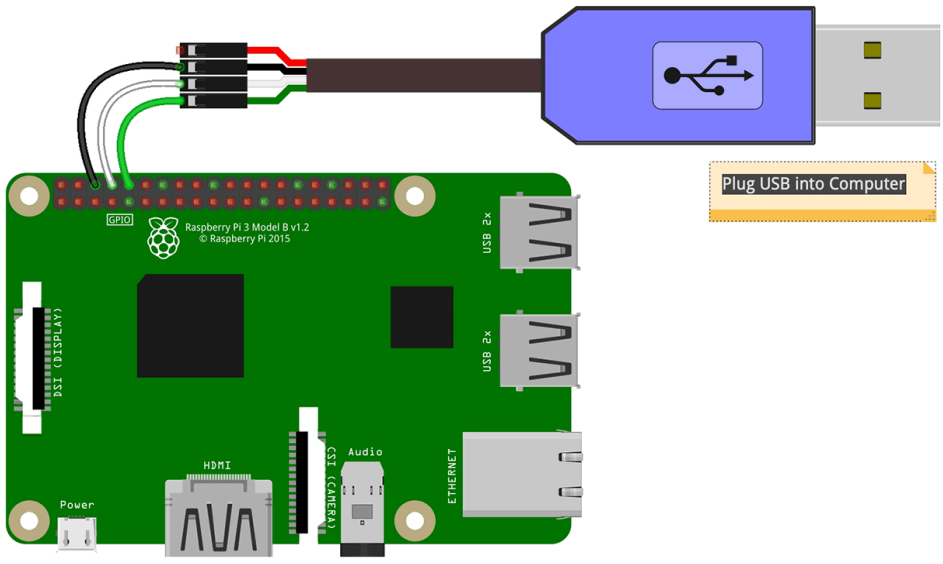


Figure: Serial Connections on the Pi



Step 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
```

IMPORTANT NOTE:

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 our target desired image alongside our 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.

```
fatload mmc 0 0x80000 ifs-rpi4.bin; go 0x80000;
```

To do so, replace ‘ifs-rpi4.bin’ with your desired image name in the scripts above. Additionally replace the U-Boot script with the following:

```
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 into U-Boot. Since there is no existing target image, U-Boot will automatically boot into the OTA image where you can run the bash script as usual. 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 lines of code to this script in order to get all of the outdated files removed and to scp in the new files.

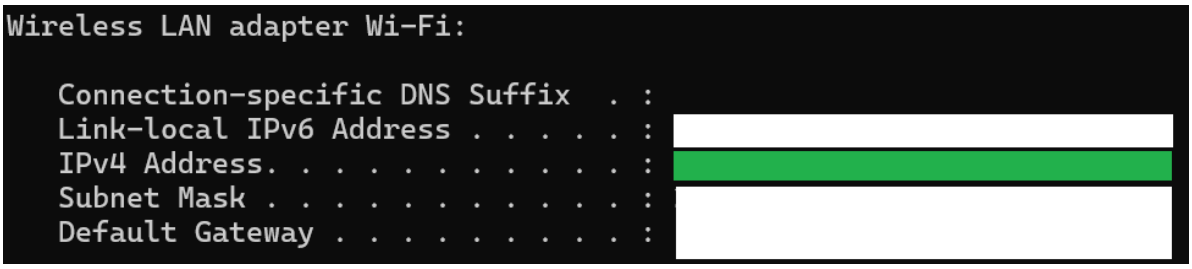
Now your pi will autoboot U-Boot, and then U-Bootwill autoboot to your desired payload. To update you will reboot, interrupt U-Boot autoboot, manually boot into the OTA QNX image, then run the OTA bash script to update the desired payload.

Step 11) How To Use The OTA Script

This final step will show you how to use the script in Step 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

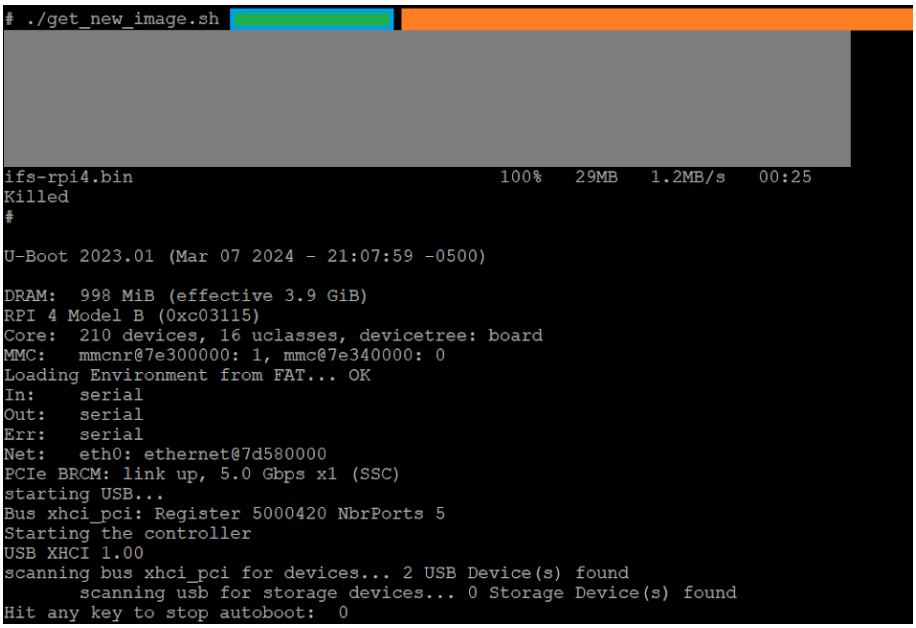
Then **replace** the \ with / as follows:

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.