Title: README Nexmon CSI Extractor Installation on RaspberryPi

Document Number: 190h-5c

CTI One Corporation

Table 1a. Document History

2023-08-29	Installation of Nexmon CSI Extractor on RaspberryPi /media/harry/easystore1/backup-2020-2-15/CTI/3proejcts/3-8-smart-tech/3-8-4-CTI/3-8-4-6-products/AIV200/190-robots-health/190h-nano-control-wireless-app/190h-5-positioning-wireless/190h-5c-readme-wifi-positioning$	YY KQ
2023-09-20	Add CSI Explorer Python Code, and Appendix A and B	YY

Table 1b. Testing and Release Approval Form

2023-09-21	Tested by YY, KQ, Pending for final testing at Facility C and pending for final release by HL	HL
2023-09-21	Approved for release to the general public	HL

Table 2. References

Number	Name and URL	Note
1.	Raspberry Pi OS (previously called Raspbian) https://www.raspberrypi.com/software/	
2.	Nexmon https://github.com/seemoo-lab/nexmon	

3.	Nexmon CSI https://github.com/seemoo-lab/nexmon_csi https://github.com/nexmonster/nexmon_csi/tree/pi-5.10.92	
4.	Nexmon CSI Bin https://github.com/nexmonster/nexmon_csi_bin	
5.	Nexmon CSI Python https://github.com/nexmonster/nexmon_csi/tree/feature/python/utils/python	
6.	CSI Human Activity https://ieee-dataport.org/open-access/csi-human-activity In Data and Scripts for Experiment 1 ReadMe.pdf	

Table 3. Prerequisite

Software Prerequisite No.	Description and Version	Note
1.	Ubuntu 18.04	To install Raspberry Pi OS
2.	Raspberry Pi OS Kernel 5.10.92	Version selection is from nexmon

		Github https://github.com/seemoo-lab/nexmon https://github.com/seemoo-lab/nexmon_csi https://github.com/nexmonster/nexmon_csi/tree/pi-5.10.92
Hardware Prerequisite No.	Description and Version	
1.	Raspberry Pi 4 Model B Rev. 1.1 (YY tested) Rev. 1.5 (KQ tested)	
2.	SD card	
3.	Micro HDMI cable	
4.	USB Type C cable	Power Supply
5.	Ethernet cable	
6.	WiFi Router (AP)	To provide WiFi

		Network

Nexmon CSI Extractor Installation on RaspberryPi

Section 1. Install Raspberry Pi OS

1. Install Raspberry Pi OS

Raspberry Pi OS Kernel 5.10.92 is an OS version which supports nexmon and nexmon CSI (https://github.com/nexmonster/nexmon_csi/tree/pi-5.10.92)

This fork and branch are for Raspberry Pi 3B+ and 4 variants.

Device	Raspberry Pi 3B+ and 4
Raspbian	Raspbian Buster Lite 2022-01-28
Chip	BCM43455c0 (built-in)
Nexmon_csi Commit	c03757
Nexmon Commit	6abd07
Date	March 24, 2022

Figure 1. Raspberry Pi OS

Raspberry Pi OS (Previously Raspbian) Release History
(https://en.wikipedia.org/wiki/Raspberry_Pi_OS)

Release date	Debian version	Linux Kernel	GCC	APT	X Server	Pi 1/1+	Pi 2	Pi 3	Pi Zero W	Pi 3+	Pi 4	Pi Zero 2 W
2021-10-30		5.10.63		[illegible]		✓	✓	✓	✓	✓	✓	✓
2021-12-03	11 (Bullseye)		10.2.1	2.2.4	1.20.11	✓	✓	✓	✓	✓	✓	✓
2022-01-28		5.10.92				✓	✓	✓	✓	✓	✓	✓
2022-03-08		5.10.103				✓	✓	✓	✓	✓	✓	✓
2022-04-04		5.15.30				✓	✓	✓	✓	✓	✓	✓
2022-09-06		5.15.61				✓	✓	✓	✓	✓	✓	✓
2022-09-22						✓	✓	✓	✓	✓	✓	✓
2023-02-21		5.15.84				✓	✓	✓	✓	✓	✓	✓
2023-05-03		6.1.21				✓	✓	✓	✓	✓	✓	✓

Figure 2. Raspberry Pi OS Release History

1.1 Install rpi-imager

Open a terminal and execute the command below on the host OS;

$ sudo snap install rpi-imager

1.2 Download Raspberry Pi OS image file

Download Raspberry Pi OS 32bit lite 2022-01-28 release version, kernel 5.10.92.

2022-01-28-raspios-bullseye-armhf-lite.zip

https://downloads.raspberrypi.org/raspios_lite_armhf/images/raspios_lite_armhf-2022-01-28/

1.3 Extract 2022-01-28-raspios-bullseye-armhf-lite.img from 2022-01-28-raspios-bullseye-armhf-lite.zip

1.4 Execute rpi-imager

Open a terminal and execute the command below on the host OS;

$ rpi-imager

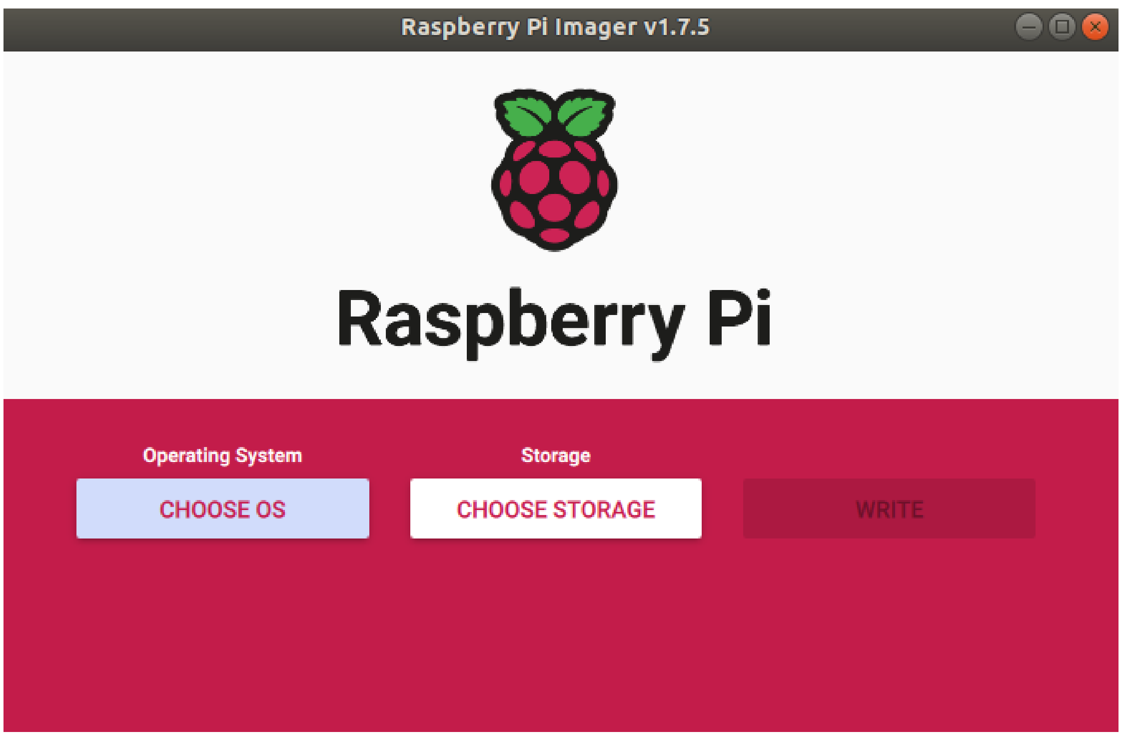

Figure 3. Raspberry Pi Imager

1.5 Click “CHOOSE OS” and select “Use custom”

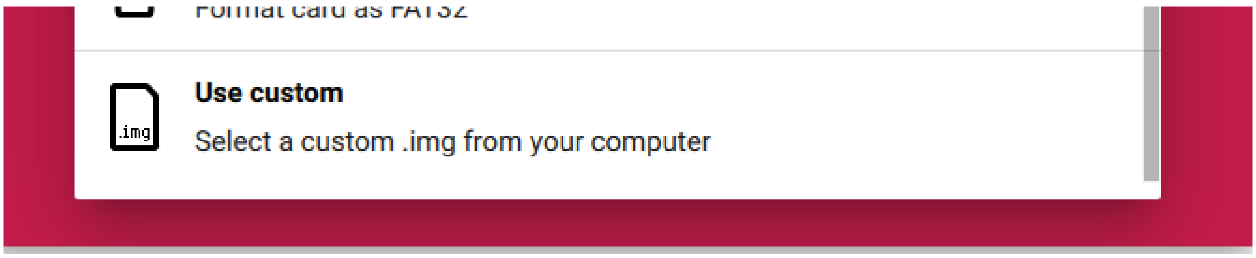

Figure 4. Raspberry Pi Imager Use custom

1.6 Choose 2022-01-28-raspios-bullseye-armhf-lite.img

1.7 Click “CHOOSE STORAGE” and select the target SD card

1.8 Click “WRITE”

2. Raspberry Pi OS Settings and SSH

https://github.com/nexmonster/nexmon_csi/tree/pi-5.10.92#readme

https://www.raspberrypi.com/documentation/computers/remote-access.html

2.1 SSH Enabling

For headless setup, SSH can be enabled by placing a file named ssh, without any extension, onto the boot partition of the SD Card. When the Raspberry Pi boots, it looks for the ssh file. If it is found, SSH is enabled and the file is deleted. The content of the file does not matter; it could contain text, or nothing at all.

2.2 Set the SD card to Raspberry Pi and power up

User: pi

Password: raspberry

Log in Raspberry Pi OS and execute the command below on Raspberry Pi to show the IP address of Raspberry Pi;

$ifconfig

2.2a (Optional) Connect to Raspberry Pi from the host OS via SSH

Open a terminal on the host OS and execute the following command;

ssh pi@10.42.0.32

Note: the example IP address should be changed

2.3 Set Country, Language, and Time zone

$ sudo raspi-config

2.3.1. Select "5 Localisation Options"

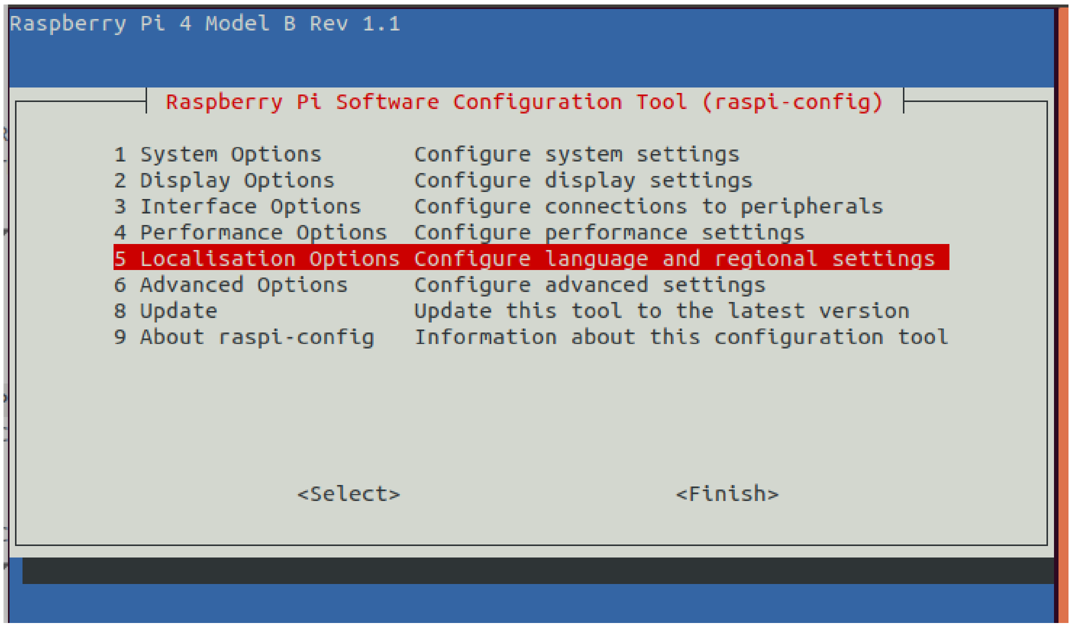

Figure 5. Localisation Options

2.3.2. Select "I1 Change Locale"

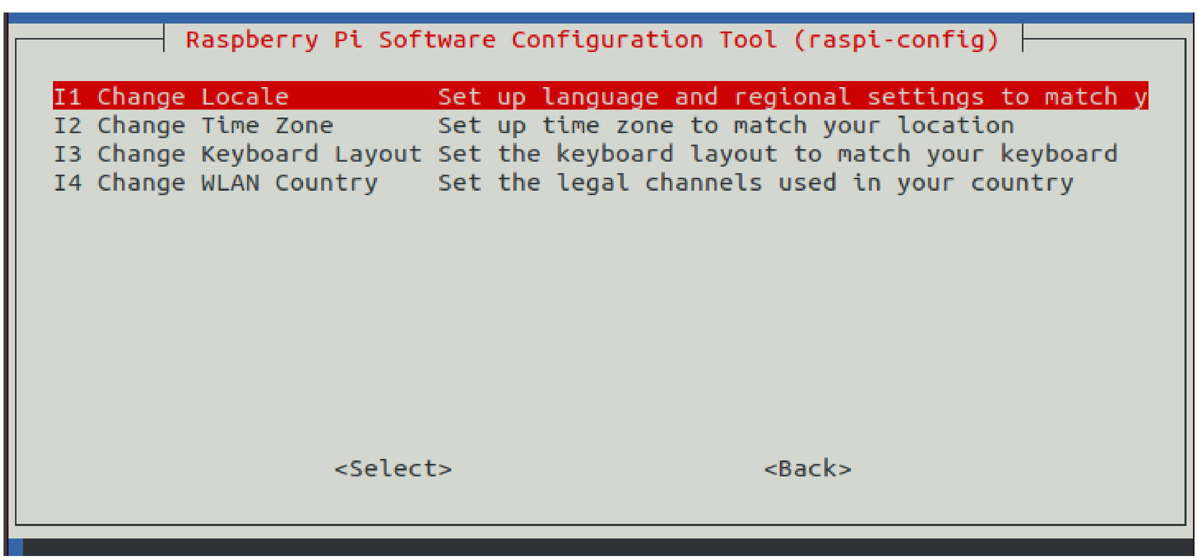

Figure 6. Change Locale

2.3.3. Find en_GB.UTF-8 and hit space key to remove "*"

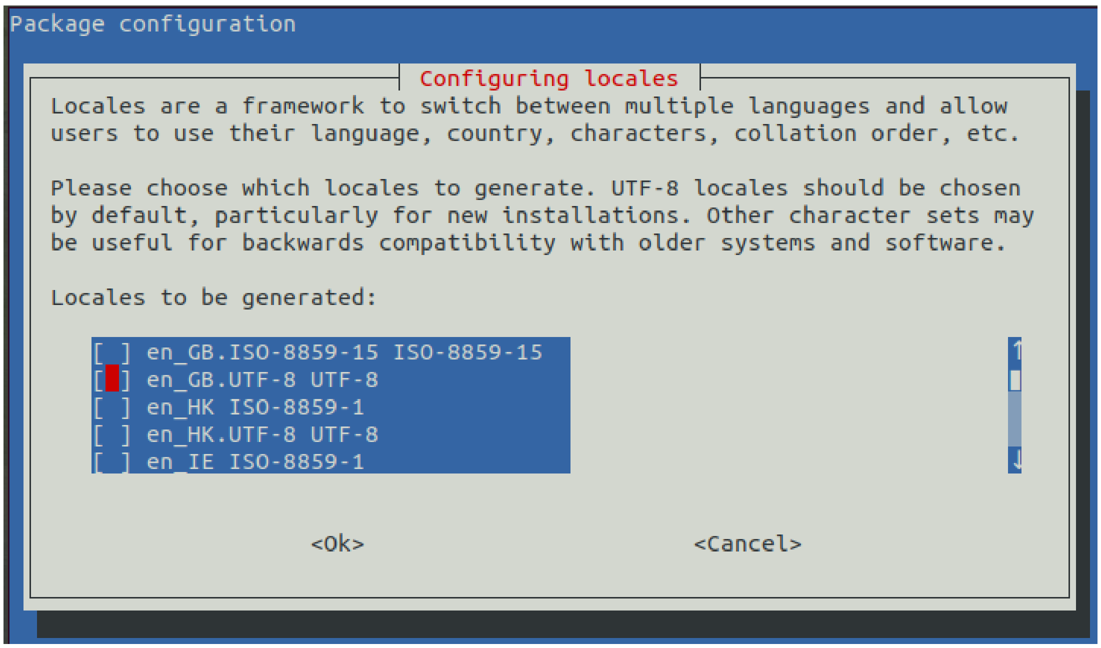

Figure 7. Disable en_GB.UTF-8

2.3.4. Find en_US.UTF-8 UTF-8, hit space key to choose, and then hit Tab key to select <OK> and hit space key

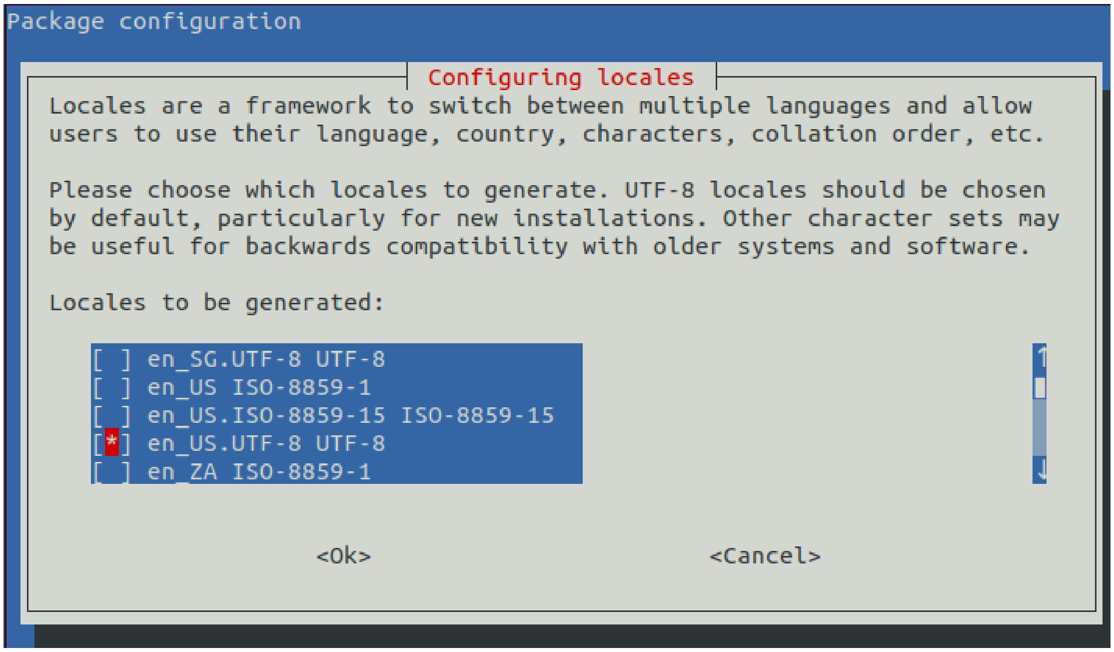

Figure 8. Enable en_US.UTF-8

2.3.5. Choose “en_US.UTF-8”, and then hit Tab key to select <OK> and hit space key

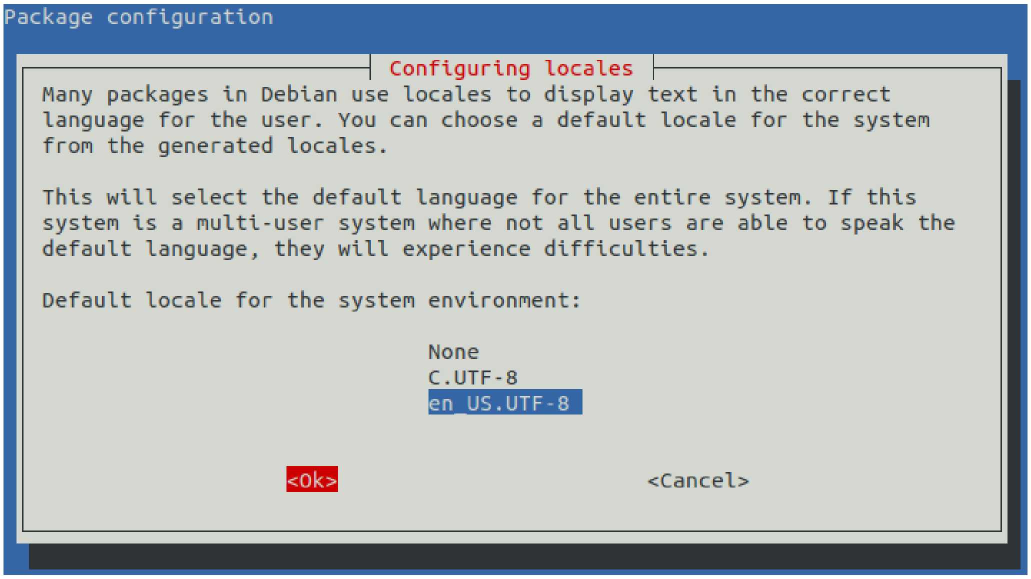

Figure 9. Confirm en_US.UTF-8

2.3.6. Select “I2 Change Time Zone”

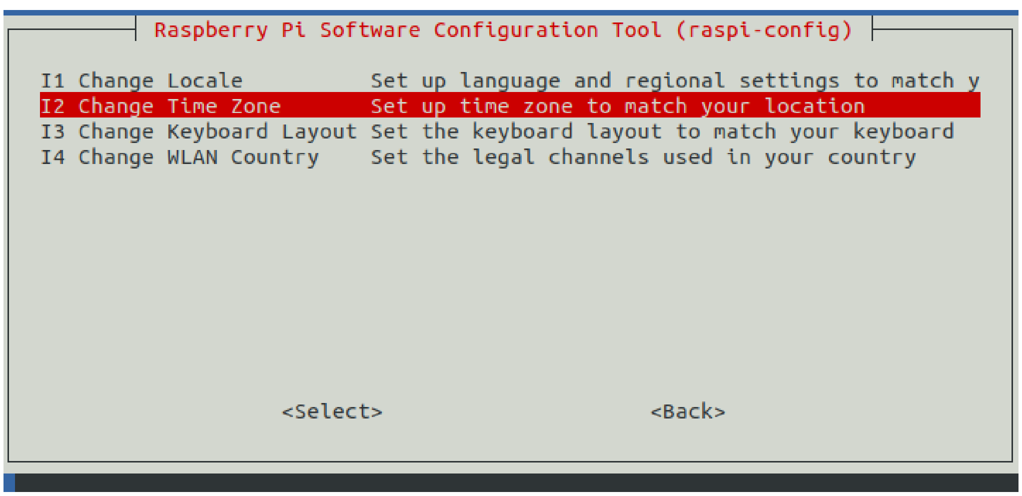

Figure 10. Change Time Zone

2.3.7. Choose “US”, and then hit Tab key to select <OK> and hit space key

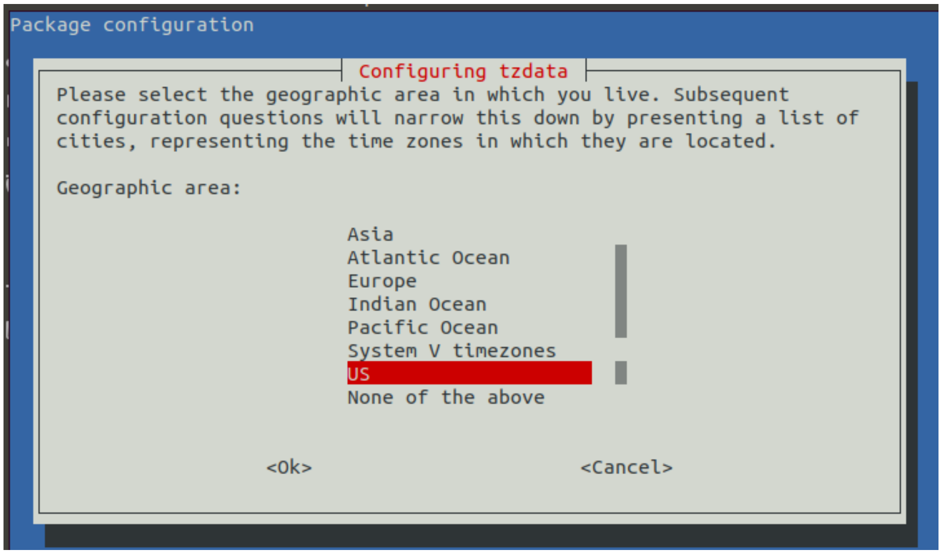

Figure 11. Change Time Zone to US

2.3.8. Choose “Pacific Ocean”, and then hit Tab key to select <OK> and hit space key

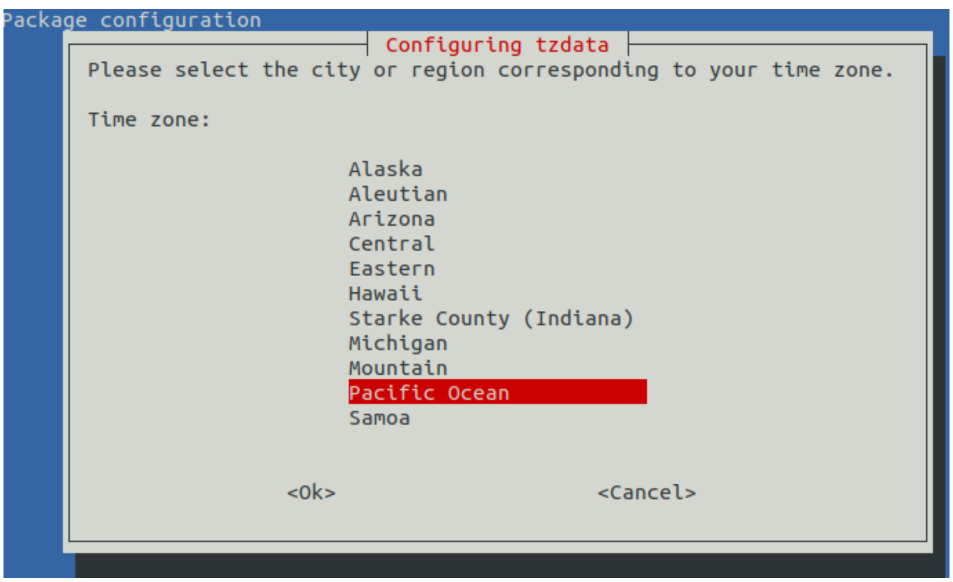

Figure 12. Set Time Zone as Pacific Ocean

2.3.9. Select “I4 Change WLAN Country”

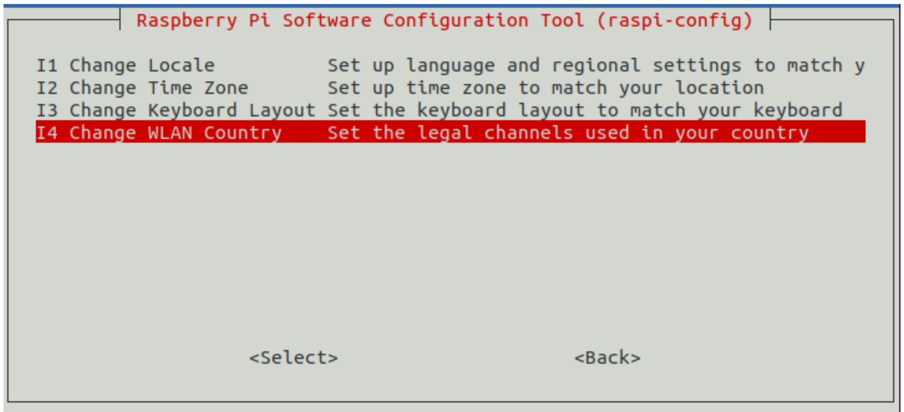

Figure 13. Change WLAN Country

2.3.10. Choose “US United States”, and then hit Tab key to select <OK> and hit space key

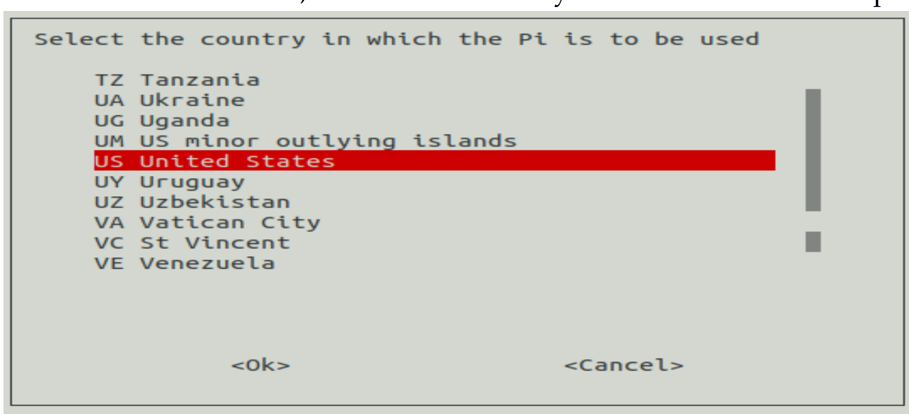

Figure 14. Choose US

3. Installing Nexmon-CSI

3.1. Check BCM43455c0 (WiFi chip) firmware file

$ sudo su

ls -al /lib/firmware/brcm/brcmfmac43455-sdio.bin

The result is like below;

lrwxrwxrwx 1 root root 31 Jan 14 2022 /lib/firmware/brcm/brcmfmac43455-sdio.bin -> ../cypress/cyfmac43455-sdio.bin

3.2. Remove soft link of BCM43455c0 (WiFi chip) firmware

rm /lib/firmware/brcm/brcmfmac43455-sdio.bin

3.3. Execute Nexmon-CSI bin installing shell

curl -fsSL https://raw.githubusercontent.com/nexmonster/nexmon_csi_bin/main/install.sh | sudo bash

The message "Done. Please reboot and see the Usage section for Nexmon_csi." will be shown when the installation finish.

3.4. Reboot the Raspberry Pi

reboot

The content of install.sh:

https://github.com/nexmonster/nexmon_csi_bin/blob/main/install.sh

==

```
#!/bin/bash
set -Eeuo pipefail
shopt -s inherit_errexit

cd "/home/pi/" # best way to ensure this is a Pi lol

mkdir -p "/home/pi/.picsi/bins/" && cd "$_"
```

```
# Download and extract binaries
if ! wget "https://github.com/nexmonster/nexmon_csi_bin/raw/main/base/$(uname -r).tar.xz"; then
    echo "Pre-compiled binaries probably don't exist for your kernel's version: $(uname -r)."
    echo "Please create a new Issue on Github and tell us what kernel you are using."
    exit
fi

tar -xvJf "$(uname -r).tar.xz" && cd "$(uname -r)"

# install nexutil
ln -s "$PWD/nexutil/nexutil" "/usr/local/bin/nexutil"

# install makecsiparams
ln -s "$PWD/makecsiparams/makecsiparams" "/usr/local/bin/mcp"
ln -s "$PWD/makecsiparams/makecsiparams" "/usr/local/bin/makecsiparams"

# install firmware and driver
cp "$PWD/patched/brcmfmac43455-sdio.bin" "/lib/firmware/brcm/brcmfmac43455-sdio.bin"
cp "$PWD/patched/brcmfmac.ko" "$(modinfo brcmfmac -n)"

depmod -a

# Unblock wifi
rfkill unblock all

# Set WiFi country and expand storage
```

```
raspi-config nonint do_wifi_country US || true
raspi-config nonint do_expand_rootfs || true

# Install tcpdump
apt update -y
apt install -y tcpdump

# disable wpa_supplicant
printf "denyinterfaces wlan0\\ninterface wlan0\\n\\tnohook wpa_supplicant\\n" >> /etc/dhcpcd.conf
killall "wpa_supplicant"
systemctl disable --now wpa_supplicant
apt remove -y wpasupplicant

echo "Done. Please reboot and see the Usage section for Nexmon_csi."
==============================================
```

4. Usage of Nexmon-CSI

(https://github.com/nexmonster/nexmon_csi/tree/pi-5.10.92#readme)

4.1. Use **`mcp`** (*makecsiparams*) to generate a base64 encoded parameter string which is used to configure the extractor. This example collects CSI on **channel 36** with **bandwidth 80 MHz** on first core of the WiFi chip, for the first spatial stream. Raspberry Pi has only one core, and a single antenna, so the -C, and -N options don't need changing.

$ mcp -C 1 -N 1 -c 36/80

result: KuABEQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==

mcp supports several other features like filtering data by Mac IDs or by FrameControl byte. Run mcp -h to see all available options.

Note: To find the channel of WiFi network, iwlist can be used after wlan0 up

$ sudo iwlist scan

4.2. Start the wireless network interface

$ sudo ifconfig wlan0 up

4.3. Set CSI extraction parameters using nexutil

$ sudo nexutil -Iwlan0 -s500 -b -l34 -vKuABEQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==

"KuABEQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==" is the result gotten from 4.1.

4.4. Create a listening interface

Add a listening interface named mon0 using the iw tool.

$ sudo iw dev wlan0 interface add mon0 type monitor

$ sudo ip link set mon0 up

4.5. Collect CSI

Collect CSI by listening on socket 5500 for UDP packets

$ sudo tcpdump -i wlan0 dst port 5500

Store 1000 CSI samples in a pcap file

$ sudo tcpdump -i wlan0 dst port 5500 -vv -w output.pcap -c 1000

5. Analyzing the CSI Data Using TCP Dump Pcap File and Wireshark

The pcap file can be opened in Wireshark**.**

5.1 Transfer the pcap file from Pi to the host.

Run the command on the host:

$ sudo scp -i ~/.ssh/id_rsa pi@10.42.0.32:/home/pi/output.pcap ~/Documents

Make sure you replace 10.42.0.32 with the actual IP address of your Raspberry Pi, and ~/Documents the actual path on the host machine where you want to save the file.

5.2 Install Wireshark on Ubuntu host

Open a terminal and execute following commands;

$ sudo add-apt-repository ppa:wireshark-dev/stable

$ sudo apt update

$ sudo apt install wireshark

5.3 Open the pacp file in Wireshark

Open Wireshark, then click on “File” and “Open” to open the output.pcap file.

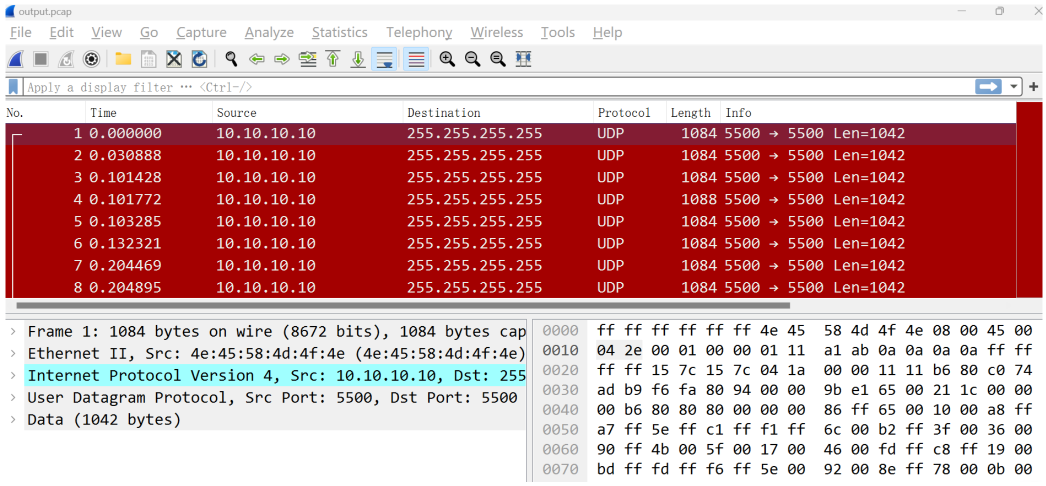

Figure 15. CSI Data Wireshark

5.4 Generate the txt file

File → Export Packet Dissections → As plain text

```
No.     Time           Source                Destination           Protocol Length Info
      1 0.000000       10.10.10.10           255.255.255.255       UDP      1084   5500 → 5500 Len=1042

Frame 1: 1084 bytes on wire (8672 bits), 1084 bytes captured (8672 bits)
Ethernet II, Src: 4e:45:58:4d:4f:4e (4e:45:58:4d:4f:4e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 10.10.10.10, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 5500, Dst Port: 5500
Data (1042 bytes)

0000  11 11 b6 80 c0 74 ad b9 f6 fa 80 94 00 00 9b e1   .....t..........
0010  65 00 21 1c 00 00 00 b6 80 80 80 00 00 00 86 ff   e.!.............
0020  65 00 10 00 a8 ff a7 ff 5e ff c1 ff f1 ff 6c 00   e.......^.....l.
0030  b2 ff 3f 00 36 00 90 ff 4b 00 5f 00 17 00 46 00   ..?.6...K._...F.
0040  fd ff c8 ff 19 00 bd ff fd ff f6 ff 5e 00 92 00   ............^...
0050  8e ff 78 00 0b 00 75 00 64 00 62 00 29 00 17 00   ..x...u.d.b.)...
0060  04 00 fd ff e7 ff e8 ff f7 ff 08 00 86 ff 41 00   ..............A.
0070  ef ff 6c 00 7f ff 1a 00 3a 00 98 00 a0 ff 20 00   ..l.....:..... .
0080  43 00 0d 00 a2 ff 39 00 ac ff 2e 00 23 00 b7 ff   C.....9.....#...
0090  da ff 39 00 ff ff f3 ff d1 ff 68 ff 3e 00 bd ff   ..9.......h.>...
00a0  fa ff df ff 27 00 32 00 2b 00 93 ff d2 ff 4b 00   ....'.2.+.....K.
00b0  5d 00 39 ff cf ff 68 00 11 00 26 00 3f 00 ab ff   ].9...h...&.?...
00c0  39 00 9c ff b1 ff ee ff db ff d2 ff 6a ff be 00   9...........j...
00d0  53 00 47 00 18 00 ee fe 2f 00 b5 ff c2 ff 78 01   S.G...../.....x.
00e0  52 01 98 fe 49 00 73 09 36 03 4f 04 60 02 3d ff   R...I.s.6.O.`.=.
00f0  8f ff 9b ff 1f 00 5d ff fd ff 66 ff c1 ff 26 00   ......]...f...&.
0100  f5 ff 58 00 27 00 31 01 e5 ff fc ff e7 ff e6 ff   ..X.'.1.........
0110  cf ff a3 00 ee ff b1 ff fd ff 35 00 58 00 9e ff   ..........5.X...
0120  f0 ff 32 ff 18 00 19 00 71 00 28 00 2e 00 ad ff   ..2.....q.(.....
0130  b3 ff 7c ff 3f 00 48 00 77 00 01 00 80 00 a2 00   ..|.?.H.w.......
0140  07 00 a6 00 1b 00 b4 ff a8 00 b2 00 44 00 67 01   ............D.g.
0150  0d ff 1e 00 a4 00 db ff a5 00 a6 00 86 ff 7e 00   ..............~.
0160  98 ff 39 00 f7 ff f8 ff bb ff ff ff c1 ff 72 ff   ..9...........r.
0170  67 00 7b ff 92 ff 9b 00 e6 ff ae ff 45 00 44 00   g.{.........E.D.
```

Figure 16. CSI Data Text

6. Python CSI Explorer

CSI Decoder written in Python

https://github.com/nexmonster/nexmon_csi/tree/feature/python/utils/python

6.1 Download Nexmon CSI branch feature/python

https://github.com/nexmonster/nexmon_csi/tree/feature/python

Note: Download as a zip file and extract contents.

6.2 Create csiexplorer_UDP.py in utils/python folder as the following

==

```
'''-----------------------------------------------------------------------------------------------------------
* Company Name : CTI One Corporation                                  *
* Program name : csiexplorer_UDP.py (Testing)                         *
* Coded By     : YY                                                   *
* Date         : 2023-09-01                                           *
* Updated By   : YY                                                   *
* Date         : 2023-09-15                                           *
* Version      : v1.1.0                                               *
* Copyright    : Copyright (c) 2023 CTI One Corporation               *
* Purpose      : WiFi Positioning, retrieve CSI data from WiFi signal using UDP communication     *
* References   : Base source code from https://github.com/nexmonster/nexmon_csi/tree/feature/python/utils/python *
*              : https://github.com/nexmonster/nexmon_csi             *
*              : https://github.com/seemoo-lab/nexmon_csi             *
*              : 190h-5-positioning-wireless-v2-hl--yy-kq-2023-8-18.odp        *
*              : nnn-nn-README-Nexmon-CSI-Extractor-Installation-RaspberryPi-v5-YY-KQ-2023-8-28.odt    *
*              :                                                      *
*              : v1.0.0 2023-09-01 YY Create from base code           *
*              : v1.1.0 2023-09-15 YY Remove the plot part which causes out of memory if this code is run for an extended period *
-----------------------------------------------------------------------------------------------------------'''
import time
import importlib
import config
```

```
import socket
import numpy as np
import datetime

IP_ADDRESS = "0.0.0.0"
PORT = 5500

decoder = importlib.import_module(f'decoders.{config.decoder}') # This is also an import


def string_is_int(s):
    '''
    Check if a string is an integer
    '''
    try:
        int(s)
        return True
    except ValueError:
        return False


def main():

    #
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  # SOCK_DGRAM for UDP communication
    server_socket.bind((IP_ADDRESS, PORT))

    while True:
        # Receive CSI Data vis UDP port 5500
        csi_data, client_addr = server_socket.recvfrom(1024)
```

```
        samples = decoder.read_csi_udp(csi_data)
        num_subcarrier = int(samples.bandwidth * 3.2)


        index = 0


        if config.print_samples:
            print(datetime.datetime.now().strftime("%Y/%m/%d %H:%M:%S"))
            samples.print(index)


        if config.plot_samples:
            csi = samples.get_csi(
                index,
                config.remove_null_subcarriers,
                config.remove_pilot_subcarriers
            )


            # Print out Subcarrier, Amplitude, Phase
            amplitudes = np.abs(csi)
            phases = np.angle(csi, deg=True)
            subcarrier_range = np.arange(-1 * num_subcarrier / 2, num_subcarrier / 2)
            # Print Subcarrier, Amplitude, and Phase values
            for idx, subcarrier in enumerate(subcarrier_range):
                print(f"Subcarrier: {subcarrier}, Amplitude: {amplitudes[idx]}, Phase: {phases[idx]}")




        else:
            print('Unknown command. Type help.')




if __name__ == "__main__":
```

main()

==

6.3 Modify decoders/interleaved.py as the following

==

```
'''-----------------------------------------------------------------------------------------------------------
* Company Name : CTI One Corporation                                   *
* Program name : interleaved.py (Testing)                              *
* Coded By     : YY                                                    *
* Date         : 2023-09-01                                            *
* Updated By   : YY                                                    *
* Date         : 2023-09-01                                            *
* Version      : v1.0.0                                                *
* Copyright    : Copyright (c) 2023 CTI One Corporation                *
* Purpose      : WiFi Positioning, retrieve CSI data from WiFi signal using UDP communication     *
* References   : Base source code from https://github.com/nexmonster/nexmon_csi/tree/feature/python/utils/python *
*              : https://github.com/nexmonster/nexmon_csi              *
*              : https://github.com/seemoo-lab/nexmon_csi              *
*              : 190h-5-positioning-wireless-v2-hl--yy-kq-2023-8-18.odp        *
*              : nnn-nn-README-Nexmon-CSI-Extractor-Installation-RaspberryPi-v6-YY-KQ-2023-9-15.odt     *
*              :                                                       *
*              : v1.0.0 2023-09-01 YY Create from base code            *
*              : v1.1.0 2023-09-15 YY Add read_csi_udp() function      *
-----------------------------------------------------------------------------------------------------------'''
'''
Interleaved
===========

Fast and efficient methods to extract
Interleaved CSI samples in PCAP files.
```

```
~230k samples per second.

Suitable for bcm43455c0 and bcm4339 chips.

Requires Numpy.

Usage
-----

import decoders.interleaved as decoder

samples = decoder.read_pcap('path_to_pcap_file')

Bandwidth is inferred from the pcap file, but
can also be explicitly set:
samples = decoder.read_pcap('path_to_pcap_file', bandwidth=40)
'''

__all__ = [
    'read_pcap'
]

import os
import numpy as np

# Indexes of Null and Pilot OFDM subcarriers
# https://www.oreilly.com/library/view/80211ac-a-survival/9781449357702/ch02.html
nulls = {
    20: [x+32 for x in [
        -32, -31, -30, -29,
```

```
        31,  30,  29,  0
    ]],

    40: [x+64 for x in [
        -64, -63, -62, -61, -60, -59, -1,
          63,  62,  61,  60,  59,  1,  0
    ]],

    80: [x+128 for x in [
        -128, -127, -126, -125, -124, -123, -1,
          127,  126,  125,  124,  123,  1,  0
    ]],

    160: [x+256 for x in [
        -256, -255, -254, -253, -252, -251, -129, -128, -127, -5, -4, -3, -2, -1,
          255,  254,  253,  252,  251,  129,  128,  127,  5,  4,  3,  3,  1,  0
    ]]
}

pilots = {
    20: [x+32 for x in [
        -21, -7,
         21,  7
    ]],

    40: [x+64 for x in [
        -53, -25, -11,
         53,  25,  11
    ]],

    80: [x+128 for x in [
```

```
        -103, -75, -39, -11,
         103,  75,  39,  11
    ]],

    160: [x+256 for x in [
        -231, -203, -167, -139, -117, -89, -53, -25,
         231,  203,  167,  139,  117,  89,  53,  25
    ]]
}

class SampleSet(object):
    '''
        A helper class to contain data read
        from pcap files.
    '''
    def __init__(self, samples, bandwidth):
        self.rssi, self.fctl, self.mac, self.seq, self.css, self.csi = samples

        self.nsamples = self.csi.shape[0]
        self.bandwidth = bandwidth
        print("self.bandwidth:", self.bandwidth)
        print("self.rssi:", self.rssi)

    def get_rssi(self, index):
        print("get_rssi(self, index): ", index)
        print("get_rssi() len(self.rss) ", len(self.rssi))
        return self.rssi[index]

    def get_fctl(self, index):
        return self.fctl[index]
```

```
def get_mac(self, index):
    return self.mac[index*6: (index+1)*6]


def get_seq(self, index):
    sc = int.from_bytes( #uint16: SC
        self.seq[index*2: (index+1)*2],
        byteorder = 'little',
        signed = False
    )
    fn = sc % 16 # Fragment Number
    sc = int((sc - fn)/16) # Sequence Number

    return (sc, fn)


def get_css(self, index):
    return self.css[index*2: (index+1)*2]


def get_csi(self, index, rm_nulls=False, rm_pilots=False):
    csi = self.csi[index].copy()
    if rm_nulls:
        csi[nulls[self.bandwidth]] = 0
    if rm_pilots:
        csi[pilots[self.bandwidth]] = 0

    return csi


def print(self, index):
    # Mac ID
    macid = self.get_mac(index).hex()
    macid = ':'.join([macid[i:i+2] for i in range(0, len(macid), 2)])
```

```
        # Sequence control
        sc, fn = self.get_seq(index)

        # Core and Spatial Stream
        css = self.get_css(index).hex()

        rssi = self.get_rssi(index)
        fctl = self.get_fctl(index)

        print(
            f'''
Sample #{index}
---------------
Source Mac ID: {macid}
Sequence: {sc}.{fn}
Core and Spatial Stream: 0x{css}
RSSI: {rssi}
FCTL: {fctl}
            '''
        )



def __find_bandwidth(incl_len):
    '''
        Determines bandwidth
        from length of packets.

        incl_len is the 4 bytes
        indicating the length of the
```

```
    packet in packet header
    https://wiki.wireshark.org/Development/LibpcapFileFormat/

    This function is immune to small
    changes in packet lengths.
'''

pkt_len = int.from_bytes(
    incl_len,
    byteorder='little',
    signed=False
)

# The number of bytes before we
# have CSI data is 60. By adding
# 128-60 to frame_len, bandwidth
# will be calculated correctly even
# if frame_len changes +/- 128
# Some packets have zero padding.
# 128 = 20 * 3.2 * 4
nbytes_before_csi = 60
pkt_len += (128 - nbytes_before_csi)

bandwidth = 20 * int(
    pkt_len // (20 * 3.2 * 4)
)

return bandwidth
```

```
def __find_nsamples_max(pcap_filesize, nsub):
    '''
        Returns an estimate for the maximum possible number
        of samples in the pcap file.

        The size of the pcap file is divided by the size of
        a packet to calculate the number of samples. However,
        some packets have a padding of a few bytes, so the value
        returned is slightly higher than the actual number of
        samples in the pcap file.
    '''

    # PCAP global header is 24 bytes
    # PCAP packet header is 12 bytes
    # Ethernet + IP + UDP headers are 46 bytes
    # Nexmon metadata is 18 bytes
    # CSI is nsub*4 bytes long
    #
    # So each packet is 12 + 46 + 18 + nsub*4 bytes long
    nsamples_max = int(
        (pcap_filesize - 24) / (
            12 + 46 + 18 + (nsub*4)
        )
    )

    return nsamples_max

def read_pcap(pcap_filepath, bandwidth=0, nsamples_max=0):
    '''
        Reads CSI samples from
```

```
    a pcap file. A SampleSet
    object is returned.

    Bandwidth and maximum samples
    are inferred from the pcap file by
    default, but you can also set them explicitly.
'''

pcap_filesize = os.stat(pcap_filepath).st_size
with open(pcap_filepath, 'rb') as pcapfile:
    fc = pcapfile.read()

if bandwidth == 0:
    bandwidth = __find_bandwidth(
        # 32-36 is where the incl_len
        # bytes for the first frame are
        # located.
        # https://wiki.wireshark.org/Development/LibpcapFileFormat/
        fc[32:36]
    )
# Number of OFDM sub-carriers
nsub = int(bandwidth * 3.2)

if nsamples_max == 0:
    nsamples_max = __find_nsamples_max(pcap_filesize, nsub)

# Preallocating memory
rssi = bytearray(nsamples_max * 1)
fctl = bytearray(nsamples_max * 1)
mac = bytearray(nsamples_max * 6)
seq = bytearray(nsamples_max * 2)
```

```
css = bytearray(nsamples_max * 2)
csi = bytearray(nsamples_max * nsub * 4)

# Pointer to current location in file.
# This is faster than using file.tell()
# =24 to skip pcap global header
ptr = 24

nsamples = 0
while ptr < pcap_filesize:
    # Read frame header
    # Skip over Eth, IP, UDP
    ptr += 8
    frame_len = int.from_bytes(
        fc[ptr: ptr+4],
        byteorder='little',
        signed=False
    )
    ptr += 50

    # 2 bytes: Magic Bytes              @ 0 - 1
    # 1 bytes: RSSI                     @ 2 - 2
    # 1 bytes: FCTL                     @ 3 - 3
    # 6 bytes: Source Mac ID            @ 4 - 10
    # 2 bytes: Sequence Number          @ 10 - 12
    # 2 bytes: Core and Spatial Stream  @ 12 - 14
    # 2 bytes: ChanSpec                 @ 14 - 16
    # 2 bytes: Chip Version             @ 16 - 18
    # nsub*4 bytes: CSI Data            @ 18 - 18 + nsub*4

    rssi[nsamples] = fc[ptr+2]
```

```
    fctl[nsamples] = fc[ptr+3]
    mac[nsamples*6: (nsamples+1)*6] = fc[ptr+4: ptr+10]
    seq[nsamples*2: (nsamples+1)*2] = fc[ptr+10: ptr+12]
    css[nsamples*2: (nsamples+1)*2] = fc[ptr+12: ptr+14]
    csi[nsamples*(nsub*4): (nsamples+1)*(nsub*4)] = fc[ptr+18: ptr+18 + nsub*4]

    ptr += (frame_len - 42)
    nsamples += 1

# Convert CSI bytes to numpy array
csi_np = np.frombuffer(
    csi,
    dtype = np.int16,
    count = nsub * 2 * nsamples
)

# Cast numpy 1-d array to matrix
csi_np = csi_np.reshape((nsamples, nsub * 2))

# Convert csi into complex numbers
csi_cmplx = np.fft.fftshift(
        csi_np[:nsamples, ::2] + 1.j * csi_np[:nsamples, 1::2], axes=(1,)
)

# Convert RSSI to Two's complement form
rssi = np.frombuffer(rssi, dtype=np.int8, count = nsamples)

return SampleSet(
    (
        rssi,
        fctl,
```

```
            mac,
            seq,
            css,
            csi_cmplx,
        ),
        bandwidth
    )


# YY 2023-09-15 Read CSI data received via UDP port 5500. CSI data starts 0x1111 the magic bytes
def read_csi_udp(csi_data, bandwidth=0, nsamples_max=0):
    '''
        Reads a CSI sample from UDP communication.
        A SampleSet object is returned.

        # Nexmon metadata is 18 bytes
        # CSI is nsub*4 bytes long
        Number of subcareer = (Received Data - Nexmon metadata)/4
        Number of subcareer = (274–18)/4=64   # Example the received data size is 274 in 802.11a
        Bandwidth = Number of subcareer / 3.2
    '''

    csi_data_size = len(csi_data)
    print("csi_data_size:", csi_data_size)

    '''
    with open("received_data", "wb") as file:
        file.write(fc)
    '''

    # YY 2023-09-07 calculate the number of subcarrierrs
```

```
# Nexmon metadata is 18 bytes
# CSI is nsub*4 bytes long
# (Received Data size - 18)/4 = the number of subcarrierrs
nsub = int((csi_data_size-18)/4)

# YY 2023-09-14 ToDo What _find_bandwidth() is
# read_pcap() uses _find_bandwidth() to calculate the bandwidth
# bandwidth is 20MHz, 40MHz, 80MHz or 160MHz depends on the type of WiFi(IEEE802.11 b/a/g/n/ac)
# IEEE802.11 a/g use 20 MHz bandwidth
bandwidth = int(nsub/3.2)
# bandwidth = 80

# YY 2023-09-07 Only 1 sample is handled in a loop
nsamples_max = 1
# if nsamples_max == 0:
#     nsamples_max = __find_nsamples_max(csi_data_size, nsub)

print("nsamples_max:", nsamples_max)

# Preallocating memory
rssi = bytearray(nsamples_max * 1)
fctl = bytearray(nsamples_max * 1)
mac = bytearray(nsamples_max * 6)
seq = bytearray(nsamples_max * 2)
css = bytearray(nsamples_max * 2)
csi = bytearray(nsamples_max * nsub * 4)

# Pointer to current location in file.
# This is faster than using file.tell()
# =24 to skip pcap global header
# ptr = 24
```

```
ptr = 0

nsamples = 0

# while ptr < csi_data_size:


# Read frame header

# Skip over Eth, IP, UDP

# ptr += 8

# frame_len = int.from_bytes(

#     csi_data[ptr: ptr + 4],

#     byteorder='little',

#     signed=False

# )

# ptr += 50


# 2 bytes: Magic Bytes              @ 0 - 1

# 1 bytes: RSSI                     @ 2 - 2

# 1 bytes: FCTL                     @ 3 - 3

# 6 bytes: Source Mac ID            @ 4 - 10

# 2 bytes: Sequence Number          @ 10 - 12

# 2 bytes: Core and Spatial Stream  @ 12 - 14

# 2 bytes: ChanSpec                 @ 14 - 16

# 2 bytes: Chip Version             @ 16 - 18

# nsub*4 bytes: CSI Data            @ 18 - 18 + nsub*4


rssi[nsamples] = csi_data[ptr + 2]

fctl[nsamples] = csi_data[ptr + 3]

mac[nsamples * 6: (nsamples + 1) * 6] = csi_data[ptr + 4: ptr + 10]

seq[nsamples * 2: (nsamples + 1) * 2] = csi_data[ptr + 10: ptr + 12]

css[nsamples * 2: (nsamples + 1) * 2] = csi_data[ptr + 12: ptr + 14]

csi[nsamples * (nsub * 4): (nsamples + 1) * (nsub * 4)] = csi_data[ptr + 18: ptr + 18 + nsub * 4]
```

```
# ptr += (frame_len - 42)

nsamples += 1


# Convert CSI bytes to numpy array

csi_np = np.frombuffer(

    csi,

    dtype=np.int16,

    count=nsub * 2 * nsamples

)


# Cast numpy 1-d array to matrix

csi_np = csi_np.reshape((nsamples, nsub * 2))


# Convert csi into complex numbers

csi_cmplx = np.fft.fftshift(

    csi_np[:nsamples, ::2] + 1.j * csi_np[:nsamples, 1::2], axes=(1,)

)


# Convert RSSI to Two's complement form

rssi = np.frombuffer(rssi, dtype=np.int8, count=nsamples)


return SampleSet(

    (

        rssi,

        fctl,

        mac,

        seq,

        css,

        csi_cmplx,

    ),

    bandwidth
```

```
  )


if __name__ == "__main__":
  samples = read_pcap('pcap_files/output-40.pcap')
```

===

6.3a Send the Python code from host OS to Pi if Nexmon CSI Github code was downloaded and modified on the host OS.

(1) Compress the nexmon_csi-feature-python folder as a zip file

(2) $sudo scp -i ~/.ssh/id_rsa ./nexmon_csi-feature-python.zip pi@10.42.0.32:/home/pi/

(3) Extract the contents of the zip fil

e on Pi OS

6.4 Log in Pi OS or access to Pi OS from the host OS via ssh

$ ssh pi@10.42.0.32

6.5. Generate Nexmon CSI string parameter

$ mcp -C 1 -N 1 -c 36/80

result: KuABEQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==

6.6. Start the wireless network interface

$ sudo ifconfig wlan0 up

6.7. Set CSI extraction parameters using nexutil

$ sudo nexutil -Iwlan0 -s500 -b -l34 -
vKuABEQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==

"KuABEQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==" is the result gotten from 6.5.

6.8. Create a listening interface

Add a listening interface named mon0 using the iw tool.

```
$ sudo iw dev wlan0 interface add mon0 type monitor
$ sudo ip link set mon0 up
```

6.9. Check if the setup is correct by using TCPDump

```
$ sudo tcpdump -i wlan0 dst port 5500
```

Push Ctrl + c to stop the dump

6.10. Execute Python code

```
$ python csiexplorer_UDP.py
```

Appendix A. CSI Nexmon Header

https://github.com/nexmonster/nexmon_csi/tree/pi-5.10.92

Bytes	Type	Name	Description
2	`uint16`	Magic Bytes	0x1111
1	`uint8`	RSSI	RSSI in Two's Complement form
1	`uint8`	FrameControl Byte	Byte that shows the WiFi Frame Type
6	`uint8[6]`	Source Mac	Source Mac ID of the WiFi Frame
2	`uint16`	Sequence Number	Sequence number of the WiFi Frame
2	`uint16`	Core and Spatial Stream	Lowest 3 bytes indicate the Core, and the next three bits indicate the Spatial Stream number.
2	`uint16`	Chanspec	Chanspec used during extraction. See `nexutil -k`.
2	`uint16`	Chip Version	Chip Version
variable	`int16[]`	CSI Data	Each CSI sample is 4 bytes with interleaved Int16 Real and Int16 Imaginary. There are `bandwidth * 3.2` OFDM subcarriers per channel, and a CSI sample for every subcarrier is present.

Note: Nexmon CSI Github page has FrameControl Byte as 2 bytes. It is wrong. FrameControl Byte is 1 byte.

Nexmon CSI Header/Meta data size is 18 bytes

Appendix B. The number of OFDM subcarriers

IEEE Standard	20 MHz	40 MHz	80MHz	160MHz
802.11a (5 GHz)	52 (Data: 48, Pilot: 4) / 64	-	-	-
802.11g (2.4 GHz)	52 (Data: 48, Pilot: 4) / 64	-	-	-
802.11n (2.4/5 GHz)	56 (Data: 52, Pilot: 4) / 64 or 52 (Data: 48, Pilot: 4) / 64	114 (Data: 108, Pilot: 6) / 128 or 104 (Data: 96, Pilot: 8) / 128	-	-
802.11ac (5 GHz)	56 (Data: 52, Pilot: 4) / 64	114 (Data: 108, Pilot: 6) / 128	242 (Data: 234, Pilot:8) / 256	484 (Data:468, Pilot: 16) / 512

This table has only WiFi in the US data.

52/64 = 52 OFDM subcarriers / 64 carrier separation (carrier separation is also called just "subcarrier")

Reference:

https://en.wikipedia.org/wiki/Wi-Fi

https://en.wikipedia.org/wiki/IEEE_802.11a-1999

https://en.wikipedia.org/wiki/IEEE_802.11g-2003

https://en.wikipedia.org/wiki/IEEE_802.11n-2009

https://rfmw.em.keysight.com/wireless/helpfiles/n7617a/mimo_ofdm_signal_structure.htm#:~:text=In%20the%2040MHz%20HT%20transmission,2%20and%202%20to%2058.

https://rfmw.em.keysight.com/wireless/helpfiles/89600b/webhelp/Subsystems/wlan-mimo/content/mimo_80211n_overview.htm

https://en.wikipedia.org/wiki/IEEE_802.11ac-2013 https://rfmw.em.keysight.com/wireless/helpfiles/n7617b/Content/Main/802.11ac%20Signal%20Structure.htm

https://www.oreilly.com/library/view/80211ac-a-survival/9781449357702/ch02.html

(END)