AN1446: SiWT917 RCP Wi-Fi Throughput

This document details the measurement of SiWx917 RCP throughput performance, including the observed throughput in various protocols such as TCP and UDP, for both uplink and downlink throughput of SiWx917 across different operating modes.

KEY POINTS

- Setup Requirements and Diagram
- Supported Protocols
- Expected results

Table of Contents

1. Introduction

In the Wireless communication, the term **Throughput** is defined as the number of data units transferred within a specified amount of time over a communication channel and reflects how the network is performing. In general, throughput is measured in bit/s or bps that is, the number of bits transferred in one second.

Many applications need to transfer a burst of data quickly over their Wi-Fi link before returning to sleep or wait state. The user measures the throughput that can be sustained by their device. This document provides information about the measurement of SiWT917 throughput performance, and the throughput observed in different protocols like TCP and UDP, uplink and downlink.

For more information, refer to the SiWT917 RCP Developers Guide and Getting Started Guide. implemented for SiWT917 RCP family of modules, which uses netlink sockets.

2. Prerequisites

Readers of this document are expected to be familiar with the Standard WLAN AP configuration, iperf tools, IP addressing, and DHCP.

2.1 Hardware

Following are the details for hardware requirements.

Table 2.1. Hardware Requirements

S.N.	Hardware Components	Quantity	Description
1.	SiWT917 RCP Wi-Fi 6 Single Band + BLE 5.4 Wireless Radio. **Radio boards:** BRD4346A. **Adapter board:** BRD8045B.	1	• **SiWx917_RB4346A -** SiWx917 Wi-Fi 6 and Bluetooth LE IC Co-Processor Radio board • **BRD8045B-** Adapter board to mount on Raspberry Pi Expansion Kit (RPI Connector).
2.	PC/Laptop/Embedded Platform with Linux OS	2	1. Raspberry Pi 4 with SiWT917 RPi image. 2. Dell Latitude 3520 with Ubuntu 20.04.
3.	Standard WLAN Access Point	1	To connect with Raspberry Pi 4.
4.	Monitor, mouse, and keyboard	1	To connect Raspberry Pi 4 with the monitor.
5.	Ethernet/HDMI cables	1	ASUS TUF Gaming AX5400 Dual-band Wi-Fi 6.

Note: For more information, refer to Getting Started Guide.

2.2 Software

Following are the details for software requirements.

Table 2.2. Software Requirements

S.N.	Software Components	Description
1.	SiWT917 RCP Driver	si91x-rcp-driver
2.	Kernel Version from 3.18 to 6.1	For example, In this test case, the system's kernel version is 6.1
3.	wpa supplicant	For example, wpa_supplicant 2.10.
4.	Measurement Tool - iperf	Refer to the section Configure iperf and test.

3. Functional Description SiWT917 on Raspberry Pi4

Throughput refers to how much data can be transferred in a certain amount of time. It is used to measure the performance of wireless networks.

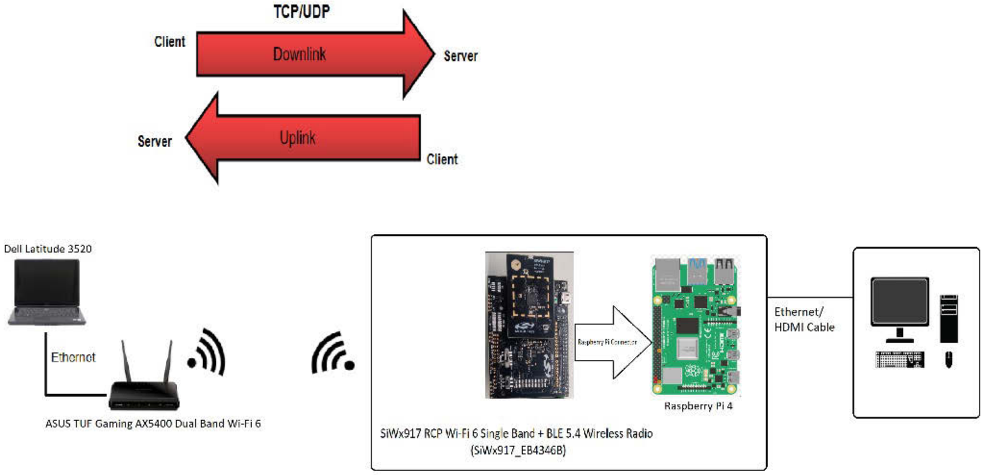


Figure 3.1. Throughput Setup Diagram

3.1 Advantages

By measuring throughput, users can determine network performance, which will help in designing their end-products.

3.2 Use Cases

Throughput test helps to calculate the application performance like audio/video data using the SiWT917 RCP module.

4. Usage Guidelines

4.1 Configuration Parameters for Driver Package

1. Download the si91x-rcp-driver
2. Unzip the driver using the following command.

```
# unzip SiWT917.x.x.x.x.zip
```

3. Enter the super user mode by giving the following command and providing the correct username and password.

```
# sudo su
```

The following section provides the steps to configure Wi-Fi station mode using startup script or manual commands. Users can choose any method.

4.1.1 Using Startup Scripts

Users can use the script at path `<system_path>/SiWT917.x.x.x.x/release/` to run Wi-Fi station mode.

```
# ./start_SiWT917.sh STA
```

Note: `<system_path>` is the location where the user has downloaded/placed the SiWT917 RCP driver in the system.

4.1.2 Manual Steps

4.1.2.1 Compiling the Driver

Change the working directory to the driver package directory and follow the compilation steps below.

1. Go to the driver package and copy all the files present in the `<system_path>/SiWT917.x.x.x.x/Firmware` folder to `/lib/firmware` by following the commands below.

```
# cd /SiWT917.x.x.x.x/Firmware/
# cp Firmware/* /lib/firmware
```

2. Configure the build flags in the driver source by navigating to the driver.

```
# cd <system_path>/SiWT917.x.x.x.x/
```

3. Build the driver using the make command.

```
# make
```

For compiling from kernel source or for other embedded platforms like the i.MX6 platform, the user can refer to the SiWT917 RCP Getting Started Guide under section **Compilation steps**.

After compilation is completed, the driver generates the following modules in the “release” folder according to the configuration.

- rsi_91x.ko
- rsi_sdio.ko

These are outlined in the following section.

4.1.2.2 Driver Installation

To install the driver, use the following commands:

1. Before installing the driver, install the dependencies using the commands below:

```
# modprobe mac80211
# modprobe bluetooth
# modprobe rfcomm
```

2. Insert rsi_91x.ko with the required module params (configuration) as shown below:

```
# insmod rsi_91x.ko dev_oper_mode=<mode> rsi_zone_enabled=<val> . .
         Example: insmod rsi_91x.ko dev_oper_mode=1 rsi_zone_enabled=0x1
```

Select dev_oper_mode as 1. For all other supported modes, refer to SiWT917 RCP Developer's Guide.

In the above example, the module param **rsi_zone_enabled** is used to program the verbosity of the debug logs. More information can be found under Debug Prints.

Now install `rsi_sdio.ko` by entering the below command :

```
# insmod rsi_sdio.ko sdio_clock = 50 Mhz
```

After a successful installation, a new wireless interface will be created as per the dev_oper_mode selection and it can be seen using the **ifconfig** command.

```
# ifconfig -a
```

Expect an output like the sample shown below with all other available interfaces included.

```
wlan0 flags = 4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
               inet6 fe80::8da:1aff:fe1e:d1c8 prefixlen 64 scopeid 0x20<link>
               ether 94:b2:16:98:ac:dc txqueuelen 1000 (Ethernet)
               RX packets: 3 bytes 372 (372.0 B)
               RX errors 0 dropped 0 overruns 0 frame 0
               TX packets: 6 bytes 696 (696.0 B)
               TX errors 0 dropped 0 overruns 0 collisions:0
```

Note: WLAN interface (wlan0) name may vary across the systems.

4.1.2.3 Installation of the Wi-Fi Client Mode

This section provides the steps to configure the Wi-Fi client mode using wpa_supplicant.

1. Before running wpa_supplicant, stop the existing network manager and unblock WLAN from rfkill. The commands below are used to stop the network-manager on different Linux distributions.

   For ubuntu, we need to use the following command:

   ```
   # service network-manager stop
   ```

   For fedora, we need to use the following command:

   ```
   # service NetworkManager stop
   ```

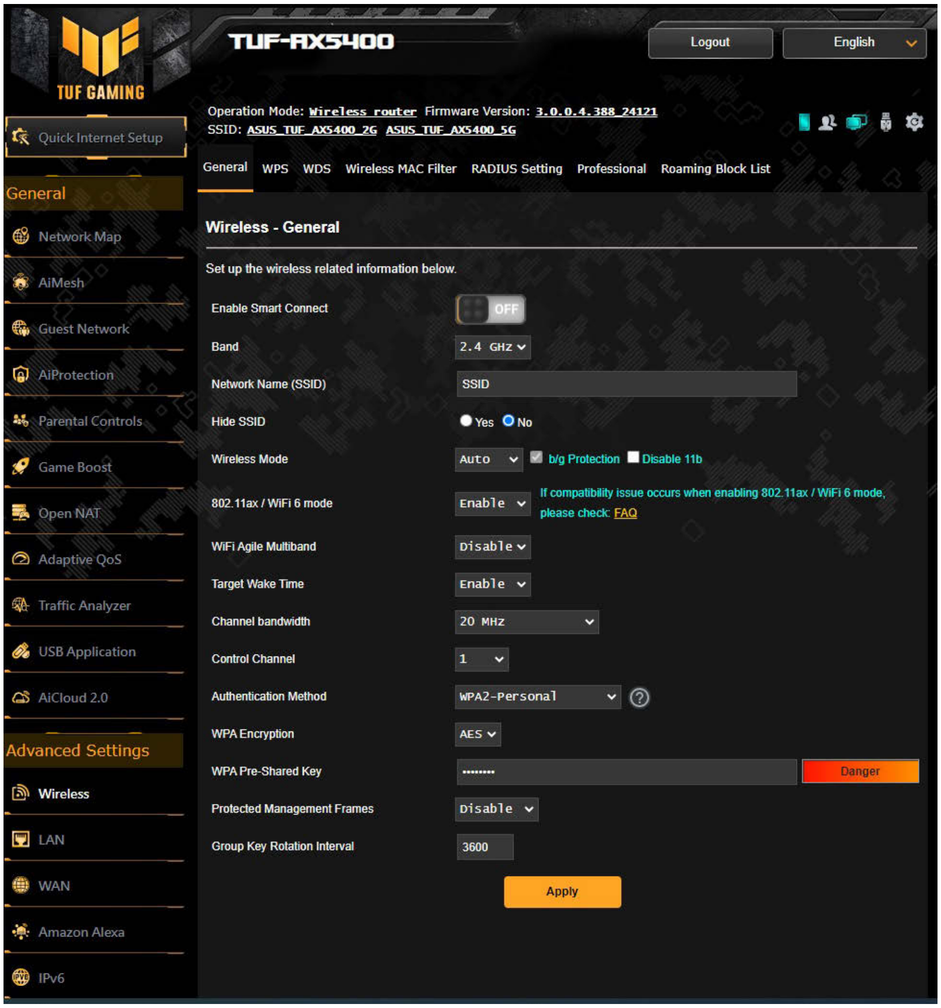
   To stop rfkill blocking WLAN, we need to use the following command:

   ```
   # rfkill unblock wlan (or) #rfkill unblock all
   ```

2. Bring up the Standard WLAN access point in the desired channel and security. For our setup, we have configured our **ASUS TUF Gaming AX5400 Dual-band Wi-Fi 6** with the following configuration as shown in the figure below.

   A few key parameters need to be enabled.

   - 11ax feature should be enabled to get high throughput.



Operation Mode: Wireless router Firmware Version: 3.0.0.4.388_24121
SSID: ASUS_TUF_AX5400_2G ASUS_TUF_AX5400_5G

Wireless - General

Set up the wireless related information below.

Enable Smart Connect: OFF
Band: 2.4 GHz
Network Name (SSID): SSID
Hide SSID: ○ Yes ● No
Wireless Mode: Auto ☑ b/g Protection □ Disable 11b
802.11ax / WiFi 6 mode: Enable — If compatibility issue occurs when enabling 802.11ax / WiFi 6 mode, please check: FAQ
WiFi Agile Multiband: Disable
Target Wake Time: Enable
Channel bandwidth: 20 MHz
Control Channel: 1
Authentication Method: WPA2-Personal
WPA Encryption: AES
WPA Pre-Shared Key: •••••••• Danger
Protected Management Frames: Disable
Group Key Rotation Interval: 3600

Figure 4.1.

3. Edit the network block present in the `sta_settings.conf` file in the **release** folder with the credentials of the **ASUS TUF Gaming AX5400 Dual-band Wi-Fi 6** access point. For our setup, we have updated in the following block.

```
ctrl_interface=/var/run/wpa_supplicant
         update_config = 1
         #Enable this network block for CCMP/TKIP mode
         network = {
                    ssid = "SSID"
                    pairwise = CCMP TKIP
                    group = CCMP TKIP
                    key_mgmt = WPA-PSK
                    psk = "12345678"
                    proto=WPA2 WPA
                    }
```

   For more details on how to update the network block for other security modes in the `sta_settings.conf` file, users must follow the SiWT917 RCP Developer's Guide.

4. Start the supplicant using the following command:

   ```
   # wpa_supplicant -i wlan0 -D nl80211 -c sta_settings.conf -dddt > supp.log &
   ```

   - **–i** option specifies the Wi-Fi interface name.
   - **<interface name> -** This name as listed in iw dev output.
   - **-D** specifies the driver interface to be used. In open-source driver, it is nl80211.
   - **-c** specifies the supplicant configuration file.
   - **-d** specifies the log level of supplicant. You can append more d's to increase the verbose.

5. To check whether the connection is successful or not, use the following command:

   ```
   # iwconfig wlan0
   ```

   For example, if connection is successful, we will see the output below:

   ```
   wlan0 IEEE 802.11bgn ESSID:"SSID" Nickname:""
                  Mode:Managed Frequency:2.412 GHz Access Point: B0:A7:B9:C4:52:CA
                  Bit Rate:39 Mb/s Tx-Power = 16 dBm
                  Retry short limit:7 RTS thr:2353 B Fragment thr:2352 B
                  Encryption key:off
                  Power Management:off
                  Link Quality = 80/80 Signal level = -28 dBm Noise level:0 dBm
                  Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
                  Tx excessive retries:0 Invalid misc:0 Missed beacon:0
   ```

   If the connection is successful, then the connected Access point SSID along with the MAC address is displayed as shown above. If it is not connected to an Access point, a message "**Not Associated**" is displayed as shown below.

   ```
   wlan0 IEEE 802.11 ESSID:off/any
                  Mode:Managed Access Point: Not-Associated Tx-Power=0 dBm
                  Retry short limit:7 RTS thr:off Fragment thr:off
                  Encryption key:off
                  Power Management:off
   ```

6. The IP address for the SiWT917-STA can be set in two ways: either get the IP address dynamically from AP or set a static IP address. To obtain a dynamic IP address from AP, use the following commands:

   ```
   # dhclient wlan0 -r
   # dhclient wlan0 -v
   ```

   To set the static IP address to SiWT917-STA, use the following command:

   ```
   # ifconfig wlan0 192.168.0.14
   ```

7. To check whether IP address is assigned or not, use the following command:

   ```
   # ifconfig wlan0
   ```

   Output:

   ```
   wlan0: flags = 4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
                   inet 192.168.0.14 netmask 255.255.255.0 broadcast 192.168.1.255
                   inet6 fe80::224:d7ff:fe56:54dc prefixlen 64 scopeid 0x20<link>
                   ether 94:b2:16:98:ac:dc txqueuelen 1000 (Ethernet)
                   RX packets 31160 bytes 31082515 (29.6 MiB)
                   RX errors 0 dropped 0 overruns 0 frame 0
                   TX packets 23356 bytes 3367496 (3.2 MiB)
                   TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
   ```

Now, users can perform data transfer tests like ping, iperf, and so on.

5. Configure iperf and Test

Iperf was developed as a modern tool for measuring maximum TCP and UDP bandwidth. Iperf allows the tuning of various parameters and UDP characteristics. Iperf reports bandwidth, delay jitter, and datagram loss. Iperf can run as a client or a server according to the arguments passed to the iperf command.

Install iperf using the following commands:

```
# sudo apt-get install iperf (for Ubuntu)
# sudo yum install iperf (for Fedora)
```

Or download and install the iperf application (for Windows) from https://iperf.fr

To run an iperf test, the iperf application should be installed on the PC/laptop and the Raspberry Pi 4 integrated with SiWT917 as follows:

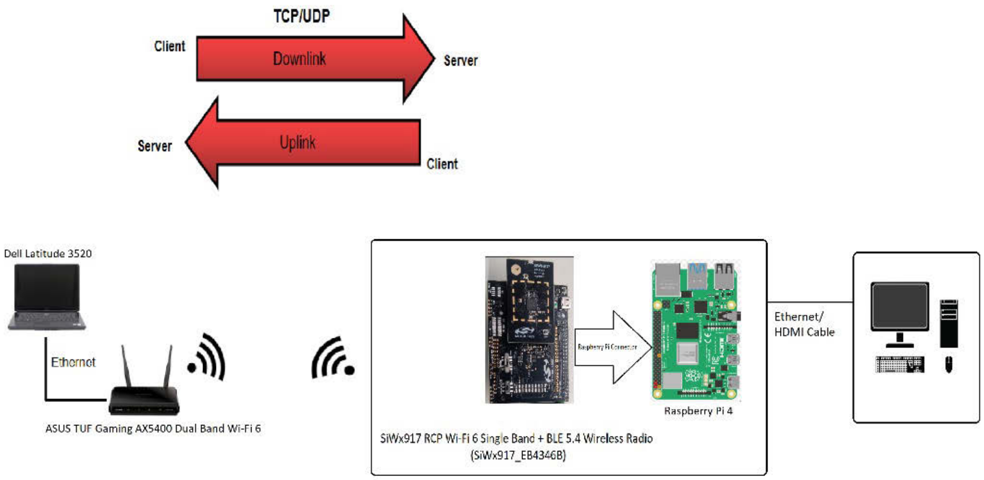


Figure 5.1. Throughput Setup Diagram

The above diagram shows SiWT917 RCP configured as SiWT917 STA and connected to “ASUS TUF Gaming AX5400 Dual-band Wi-Fi 6” over Wi-Fi. We will measure the uplink and downlink bandwidth using both TCP and UDP.

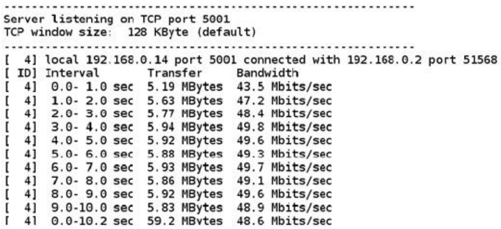
5.1 TCP Rx

Note:

- IP of SiWT917-STA is 192.168.0.14
- IP of PC/laptop connected to access point is 192.168.0.2

Run the following commands to run TCP data transfer:

- For Raspberry Pi 4 integrated with SiWT917 RCP: iperf -s -i 1
- For Dell Laptop Latitude 3520: iperf -c 192.168.0.14 -i 1 -t 10

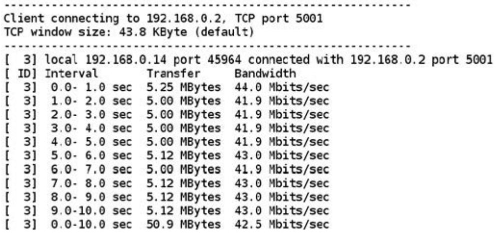


```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 192.168.0.14 port 5001 connected with 192.168.0.2 port 51568
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 1.0 sec  5.19 MBytes  43.5 Mbits/sec
[  4]  1.0- 2.0 sec  5.63 MBytes  47.2 Mbits/sec
[  4]  2.0- 3.0 sec  5.77 MBytes  48.4 Mbits/sec
[  4]  3.0- 4.0 sec  5.94 MBytes  49.8 Mbits/sec
[  4]  4.0- 5.0 sec  5.92 MBytes  49.6 Mbits/sec
[  4]  5.0- 6.0 sec  5.88 MBytes  49.3 Mbits/sec
[  4]  6.0- 7.0 sec  5.93 MBytes  49.7 Mbits/sec
[  4]  7.0- 8.0 sec  5.86 MBytes  49.1 Mbits/sec
[  4]  8.0- 9.0 sec  5.92 MBytes  49.6 Mbits/sec
[  4]  9.0-10.0 sec  5.83 MBytes  48.9 Mbits/sec
[  4]  0.0-10.2 sec  59.2 MBytes  48.6 Mbits/sec
```

5.2 TCP TX

Run the following commands to run TCP data transfer:

- For Dell Laptop Latitude 3520: **iperf -s -i 1**
- For Raspberry Pi 4 integrated with SiWT917 RCP: **iperf -c 192.168.0.2 -i 1 -t 10**

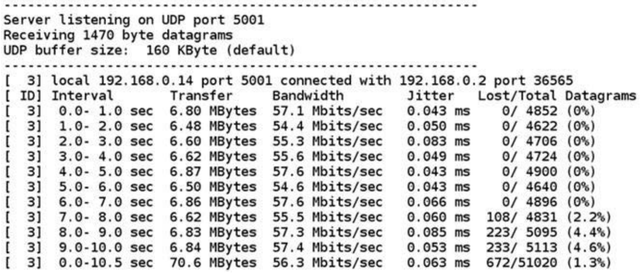


```
------------------------------------------------------------
Client connecting to 192.168.0.2, TCP port 5001
TCP window size: 43.8 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.14 port 45964 connected with 192.168.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  5.25 MBytes  44.0 Mbits/sec
[  3]  1.0- 2.0 sec  5.00 MBytes  41.9 Mbits/sec
[  3]  2.0- 3.0 sec  5.00 MBytes  41.9 Mbits/sec
[  3]  3.0- 4.0 sec  5.00 MBytes  41.9 Mbits/sec
[  3]  4.0- 5.0 sec  5.00 MBytes  41.9 Mbits/sec
[  3]  5.0- 6.0 sec  5.12 MBytes  43.0 Mbits/sec
[  3]  6.0- 7.0 sec  5.00 MBytes  41.9 Mbits/sec
[  3]  7.0- 8.0 sec  5.12 MBytes  43.0 Mbits/sec
[  3]  8.0- 9.0 sec  5.12 MBytes  43.0 Mbits/sec
[  3]  9.0-10.0 sec  5.12 MBytes  43.0 Mbits/sec
[  3]  0.0-10.0 sec  50.9 MBytes  42.5 Mbits/sec
```

5.3 UDP Rx

Run the following commands to run UDP data transfer:

- For Raspberry Pi 4 integrated with SiWT917 RCP: **iperf -s -u -i 1**
- For Dell Laptop Latitude 3520: **iperf -c 192.168.0.14 -i 1 -t 10 -u -b -50M**

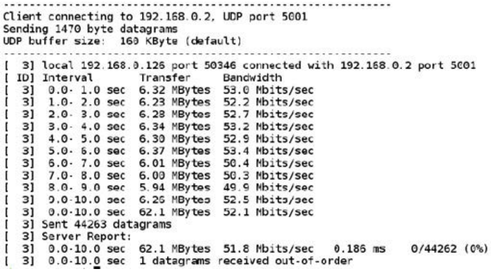


```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.14 port 5001 connected with 192.168.0.2 port 36565
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 1.0 sec  6.80 MBytes  57.1 Mbits/sec   0.043 ms    0/ 4852 (0%)
[  3]  1.0- 2.0 sec  6.48 MBytes  54.4 Mbits/sec   0.050 ms    0/ 4622 (0%)
[  3]  2.0- 3.0 sec  6.60 MBytes  55.3 Mbits/sec   0.083 ms    0/ 4706 (0%)
[  3]  3.0- 4.0 sec  6.62 MBytes  55.6 Mbits/sec   0.049 ms    0/ 4724 (0%)
[  3]  4.0- 5.0 sec  6.87 MBytes  57.6 Mbits/sec   0.043 ms    0/ 4900 (0%)
[  3]  5.0- 6.0 sec  6.50 MBytes  54.6 Mbits/sec   0.043 ms    0/ 4640 (0%)
[  3]  6.0- 7.0 sec  6.86 MBytes  57.6 Mbits/sec   0.066 ms    0/ 4896 (0%)
[  3]  7.0- 8.0 sec  6.62 MBytes  55.5 Mbits/sec   0.060 ms  108/ 4831 (2.2%)
[  3]  8.0- 9.0 sec  6.83 MBytes  57.3 Mbits/sec   0.085 ms  223/ 5095 (4.4%)
[  3]  9.0-10.0 sec  6.84 MBytes  57.4 Mbits/sec   0.053 ms  233/ 5113 (4.6%)
[  3]  0.0-10.5 sec  70.6 MBytes  56.3 Mbits/sec   0.063 ms  672/51020 (1.3%)
```

5.4 UDP Tx

Run the following commands to run UDP data transfer:

- For Raspberry Pi 4 integrated with SiWT917 RCP: **iperf -c 192.168.0.2 -i 1 -t 10 -u -b 50M**
- For Dell Laptop Latitude 3520: **iperf -s -u -i 1**



```
------------------------------------------------------------
Client connecting to 192.168.0.2, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.126 port 50346 connected with 192.168.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  6.32 MBytes  53.0 Mbits/sec
[  3]  1.0- 2.0 sec  6.23 MBytes  52.2 Mbits/sec
[  3]  2.0- 3.0 sec  6.28 MBytes  52.7 Mbits/sec
[  3]  3.0- 4.0 sec  6.34 MBytes  53.2 Mbits/sec
[  3]  4.0- 5.0 sec  6.30 MBytes  52.9 Mbits/sec
[  3]  5.0- 6.0 sec  6.37 MBytes  53.4 Mbits/sec
[  3]  6.0- 7.0 sec  6.01 MBytes  50.4 Mbits/sec
[  3]  7.0- 8.0 sec  6.00 MBytes  50.3 Mbits/sec
[  3]  8.0- 9.0 sec  5.94 MBytes  49.9 Mbits/sec
[  3]  9.0-10.0 sec  6.26 MBytes  52.5 Mbits/sec
[  3]  0.0-10.0 sec  62.1 MBytes  52.1 Mbits/sec
[  3] Sent 44263 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  62.1 MBytes  51.8 Mbits/sec   0.186 ms    0/44262 (0%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
```

6. Expected Results

Application data throughput up to 50 Mbps (Hosted Mode) in 802.11ax.

Table 6.1. Expected Throughput Results

S.N.	Operating Mode	Actual Operation Mode (Dependent on User Test Case)	Band (GHz)	Channel Width	Protocol			
					TCP		UDP	
					Uplink	Downlink	Uplink	Downlink
1.	Wi-Fi_STA Only	STA mode	2.4	20	46.2	52.6	61.1	66.2
2.	Wi-Fi_AP Only	AP mode	2.4	20	38	41.9	45.6	52.8
3.	STA+AP	STA mode	2.4	20	34.3	47.5	52.7	57.7
4.		AP mode	2.4	20	25	22.7	33.4	49.8
5.	Wi-Fi_STA + BT LE	STA mode + BLE (Advertising mode)	2.4	20	51	53.4	64	37.9

Note:

1. The above mentioned throughput numbers are verified using SDIO on Raspberry Pi 4 in shielded chamber.
2. Wi-Fi throughput varies with the environment of the test setup: range, obstacles, type of obstacles, interference, and performance of the target platform.
3. For operating _mode, refer the following table for operating mode configuration.

Table 6.2. Operating Mode Configuration

S.N.	Protocols Support	Operating Mode
1	Wi-Fi_alone_STA	1
2	Wi-Fi_alone_AP	1
3	BLE	8
4	Wi-Fi_STA + BLE	13

7. Summary/ Conclusion

By following the above procedures, SiWT917 RCP-STA can connect to a standard access point and check the Wi-Fi throughput.

8. Appendix A: Terminology

- **NL 80211 -** the new 802.11 netlink interface public header
- **TCP -** Transmission Control Protocol
- **UDP -** User Datagram Protocol
- **DHCP -** Dynamic Host Configuration Protocol
- **Uplink -** Data transfer from station (SiWT917 RCP) to access point
- **Downlink-** Data transfer from access point to station (SiWT917 RCP)
- **SiWT917-STA -** Station interface that is created for SiWT917 RCP after loading the driver
- **SiWT917-AP -** Access Point interface that is created for SiWT917 RCP after loading the driver

9. Appendix B: Refrences and Related Documentation

Refer to SiWT917 RCP Developers Guide and Getting Started Guide.

10. Appendix C: Troubleshooting

1. Ensure that the IP address is assigned for both PC/laptop (AP) and PC/laptop (SiWT917 RCP-STA) before running an iperf test.
2. Ensure that the STA is connected to the AP using the command **iwconfig**.
3. While running iperf, the server should start first, and then the client.
4. For SDIO detection:

```
# cat /sys/bus/sdio/devices/mmcXXXXX/vendor
```

11. Revision History

Revision 1.1

January, 2025

- Removed BRD4357A radio board reference from Table 2.1 Hardware Requirements on page 4.

Revision 1.0

January, 2025

- Initial release.

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications.

Trademark Information

Silicon Laboratories Inc.®, Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Redpine Signals®, WiSeConnect , n-Link, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, Gecko OS, Gecko OS Studio, Precision32®, Simplicity Studio®, Telegesis, the Telegesis Logo®, USBXpress® , Zentri, the Zentri logo and Zentri DMS, Z-Wave®, and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com