

# RS9116\_RS14100 Wireless SAPI Examples

Version 1.5

October 2018

---

#### **Redpine Signals, Inc.**

2107 North First Street, Suite #540, San Jose, California 95131,  
United States of America.

Phone: +1-408-748-3385, Fax: +1-408-705-2019  
Email: [sales@redpinesignals.com](mailto:sales@redpinesignals.com)  
Website: [www.redpinesignals.com](http://www.redpinesignals.com)

---

# Disclaimer

The information in this document pertains to information related to Redpine Signals, Inc. products. This information is provided as a service to our customers, and may be used for information purposes only.

Redpine assumes no liabilities or responsibilities for errors or omissions in this document. This document may be changed at any time at Redpine's sole discretion without any prior notice to anyone. Redpine is not committed to updating this document in the future.

Copyright © 2018 Redpine Signals, Inc. All rights reserved.

---

# Table of Contents

1	Introduction to RS9116 & RS14100 Wireless SAPI Examples.....	20
2	BT Classic Examples.....	21
2.1	BT PER Example .....	21
2.1.1	Application Overview.....	21
2.1.2	Application Setup.....	21
2.1.3	Configuration and Execution of the Application .....	22
2.2	BT SPP Master Example.....	26
2.2.1	Application Overview.....	26
2.2.2	Application Setup.....	26
2.2.3	Configuration and Execution of the Application .....	27
2.3	BT SPP Slave Example .....	31
2.3.1	Application Overview.....	31
2.3.2	Application Setup.....	31
2.3.3	Configuration and Execution of the Application .....	32
2.4	SSP Test Example.....	37
2.4.1	Application Overview.....	37
2.4.2	Application Setup.....	37
2.4.3	Configuration and Execution of the Application .....	38
2.5	BT A2DP Source with AVRCP Example .....	43
2.5.1	Application Overview.....	43
2.5.2	Application Setup.....	43
2.5.3	Configuration and Execution of the Application .....	44
2.6	BT A2DP Sink Example.....	47
2.6.1	Application Overview.....	47
2.6.2	Application Setup.....	47
2.6.3	Configuration and Execution of the Application .....	48
3	BT LE Applications .....	50
3.1	Heart Rate Example .....	50
3.1.1	Application Overview.....	50
3.1.2	Application Setup.....	50
3.1.3	Configuration and Execution of the Application .....	51

---

3.2 Simple Central Example .....	57
3.2.1 Application Overview.....	57
3.2.2 Application Setup.....	57
3.2.3 Configuration and Execution of the Application .....	58
3.3 Simple Peripheral Example .....	60
3.3.1 Application Overview.....	60
3.3.2 Application Setup.....	60
3.3.3 Configuration and Execution of the Application .....	61
3.4 Simple Chat Example.....	63
3.4.1 Application Overview.....	63
3.4.2 Application Setup.....	64
3.4.3 Configuration and Execution of the Application .....	65
3.5 SMP Example.....	68
3.5.1 Application Overview.....	68
3.5.2 Application Setup.....	69
3.5.3 Configuration and Execution of the Application .....	70
3.6 Simple Peripheral Power Save Example.....	73
3.6.1 Application Overview.....	73
3.6.2 Application Setup.....	73
3.6.3 Configuration and Execution of the Application .....	74
3.7 BLE Immediate Alert Client Example .....	79
3.7.1 Application Overview.....	79
3.7.2 Application Setup.....	79
3.7.3 Details of the Application .....	79
3.7.4 Configuration and Execution of the Application .....	79
3.7.5 Configuring the Application .....	79
3.7.6 Executing the Application.....	82
3.8 iBeacon Example.....	82
3.8.1 Application Overview.....	82
3.8.2 Application Setup.....	82
3.8.3 Application Setup.....	84
3.8.4 Configuration and Execution of the Application .....	85
3.9 Privacy Example.....	87
3.9.1 Feature Overview .....	87

---

---

3.9.2 Application Overview.....	87
3.9.3 Application Setup.....	88
3.9.4 Configuration and Execution of the Application .....	88
3.10 Proximity Profile .....	94
3.10.1 Application Overview.....	94
3.10.2 Application Setup.....	95
3.10.3 Configuration and Execution of the Application .....	96
3.11 Heart Rate Profile Example .....	99
3.11.1 Objective.....	99
3.12 BLE Long Read Example .....	106
3.12.1 Application Overview.....	106
3.12.2 Configuration and Execution of the Application .....	107
3.13 BLE Long Range and 2Mbps .....	110
3.13.1 Application Overview.....	110
3.13.2 Application Setup.....	110
3.13.3 Configuration and Execution of the Application .....	111
3.14 Blood Pressure Profile Example .....	112
3.14.1 Objective.....	112
3.14.2 GATT Requirements .....	113
3.14.3 Service Definition .....	113
3.14.4 Service Characteristics .....	113
3.14.5 Process for Blood Pressure Server and Client .....	115
3.14.6 Example Setup .....	115
3.14.7 Test-Cases .....	116
3.15 BLE Data Length Extinctions Example .....	119
3.15.1 Application Overview.....	119
3.15.2 Application Setup.....	120
3.15.3 Configuration and Execution of the Application .....	120
3.16 LE Secure Connections Example .....	122
3.16.1 Application Overview.....	122
3.16.2 Application Setup.....	122
3.16.3 Configuration and Execution of the Application .....	123
3.17 BLE-L2CAP Connection Based Flow Control Example .....	125
3.17.1 Application Overview.....	125

---

---

3.17.2 Application Setup.....	125
3.17.3 Configuration and Execution of the Application .....	126
<b>3.18 Battery Service .....</b>	<b>129</b>
3.18.1 Application Overview.....	129
3.18.2 Application Setup.....	129
3.18.3 Configuration and Execution of the Application .....	131
<b>3.19 Health Thermometer .....</b>	<b>138</b>
3.19.1 Application Overview.....	138
3.19.2 Application Setup.....	138
3.19.3 Configuration and Execution of the Application .....	140
<b>3.20 BLE PER Example .....</b>	<b>144</b>
3.20.1 Application Overview.....	144
3.20.2 Application Setup.....	144
3.20.3 Configuration and Execution of the Application .....	145
<b>3.21 Blood Pressure Service (BLS) Example .....</b>	<b>148</b>
3.21.1 Definitions and Acronyms.....	148
3.21.2 Application Overview.....	149
3.21.3 Sequence of Events.....	149
3.21.4 Application Setup.....	149
3.21.5 Configuration and Execution of the Application .....	150
<b>3.22 Glucose Service (GLS) Example .....</b>	<b>158</b>
3.22.1 Definitions and Acronyms.....	158
3.22.2 Application Overview.....	158
3.22.3 Sequence of Events.....	159
3.22.4 Application Setup.....	159
3.22.5 Configuration and Execution of the Application .....	161
<b>3.23 Human Interface Device Service (HIDS) Example .....</b>	<b>166</b>
3.23.1 Definitions and Acronyms.....	166
3.23.2 Application Overview.....	166
3.23.3 Sequence of Events.....	167
3.23.4 Application Setup.....	167
3.23.5 WiSeMCU / WiSeConnect based Setup Requirements .....	167
3.23.6 Configuration and Execution of the Application .....	168
<b>3.24 BLE Whitelist Example .....</b>	<b>176</b>

---

---

3.24.1 Application Overview.....	176
3.24.2 Application Setup.....	177
3.24.3 Configuration and Execution of the Application .....	178
<b>3.25 BLE Dual Role Example.....</b>	<b>180</b>
3.25.1 Application Overview.....	180
3.25.2 Application Setup.....	180
3.25.3 Configuration and Execution of the Application .....	180
<b>4 Security APIs.....</b>	<b>184</b>
4.1 AES Application.....	184
4.1.1 Example.....	184
4.1.2 Example Setup .....	184
4.1.3 WiSeMCU / WiSeConnect based Setup Requirements.....	184
4.1.4 Configuration and Execution of the Application .....	184
4.2 ECDH Application.....	185
4.2.1 Example.....	185
4.2.2 Example Setup .....	185
4.2.3 Configuration and Execution of the Application .....	185
4.3 Exponentiation Application.....	186
4.3.1 Example.....	186
4.3.2 Example Setup .....	186
4.3.3 Configuration and Execution of the Application .....	187
4.4 HMAC SHA Application.....	187
4.4.1 Example.....	187
4.4.2 Example Setup .....	187
4.4.3 Configuration and Execution of the Application .....	188
4.5 PUF AES Example .....	188
4.5.1 Example.....	188
4.5.2 Example Setup .....	189
4.5.3 Configuration and Execution of the Application .....	189
4.6 PUF Configuration Example .....	190
4.6.1 Example.....	190
4.6.2 Example Setup .....	190
4.6.3 Configuration and Execution of the Application .....	190

---

---

4.7 PUF Demo Example .....	191
4.7.1 Example.....	191
4.7.2 Example Setup .....	191
4.7.3 Configuration and Execution of the Application .....	192
4.8 PUF Key AES Example .....	196
4.8.1 Example.....	196
4.8.2 Example Setup .....	197
4.8.3 Configuration and Execution of the Application .....	197
4.9 SHA application.....	198
4.9.1 Example.....	198
4.9.2 Example Setup .....	198
4.9.3 Configuration and Execution of the Application .....	198
5 WLAN BLE .....	200
5.1 WLAN-AP BLE bridge Example.....	200
5.1.1 Application Overview.....	200
5.1.2 Application Setup.....	200
5.1.3 Application Setup.....	200
5.1.4 Configuration and Execution of the Application .....	202
5.2 Wlan Ap Ble Bridge TCPIP Bypass .....	212
5.2.1 Application Overview.....	212
5.2.2 Application Setup.....	212
5.2.3 Application Setup.....	212
5.2.4 Configuration and Execution of the Application .....	214
5.3 WLAN-STATION BLE BRIDGE Example .....	225
5.3.1 Application Overview.....	225
5.3.2 Application Setup.....	226
5.3.3 Configuration and Execution of the Application .....	228
5.4 WLAN-STATION BLE-Dual Role Bridge Example.....	238
5.4.1 Application Overview.....	238
5.4.2 Application Setup.....	239
5.4.3 Configuration and Execution of the Application .....	240
5.5 WLAN-STATION BLE-Multiple Slaves Bridge Example .....	250
5.5.1 Application Overview.....	250

---

---

5.5.2 Application Setup.....	251
5.5.3 Configuration and Execution of the Application .....	252
<b>5.6 WLAN-STATION BLE THROUGHPUT Example .....</b>	<b>256</b>
5.6.1 Introduction .....	256
5.6.2 Configuration and Execution of the Application .....	258
<b>5.7 WLAN-STATION Standby BLE Advertising Power Save Example.....</b>	<b>264</b>
5.7.1 Application Overview.....	264
5.7.2 Application Setup.....	264
5.7.3 Configuration and Execution of the Application .....	265
<b>5.8 WLAN-STATION Standby BLE connected Power Save Example.....</b>	<b>269</b>
5.8.1 Application Overview.....	269
5.8.2 Application Setup.....	270
5.8.3 Configuration and Execution of the Application .....	271
<b>6 WLAN BT .....</b>	<b>276</b>
6.1 WLAN-AP BT Bridge Example .....	276
6.1.1 Application Overview.....	276
6.1.2 Application Setup.....	276
6.1.3 Configuration and Execution of the Application .....	278
6.2 WLAN-AP BT Bridge TCPIP Bypass Example.....	284
6.2.1 Application Overview.....	284
6.2.2 Application Setup.....	284
6.2.3 Configuration and Execution of the Application .....	286
6.3 WLAN STA bridge TCPIP bypass Example.....	295
6.3.1 Application Overview.....	295
6.3.2 Application Setup.....	295
6.3.3 Application Setup.....	295
6.3.4 Configuration and Execution of the Application' .....	297
6.4 WLAN-STATION BT Bridge Example .....	304
6.4.1 Application Overview.....	304
6.4.2 Application Setup.....	305
6.4.3 Configuration and Execution of the Application .....	306
<b>6.5 WLAN-STATION BT THROUGHPUT Example .....</b>	<b>311</b>
6.5.1 Introduction .....	311

---

---

6.5.2 Configuration and Execution of the Application .....	313
<b>6.6 WLAN-STATION Standby BT Connected Power Save Example .....</b>	<b>317</b>
6.6.1 Application Overview .....	317
6.6.2 Application Setup.....	318
6.6.3 Configuration and Execution of the Application .....	318
<b>7 WLAN BT BLE .....</b>	<b>322</b>
7.1 WLAN-AP BT and BLE Bridge TCPIP Bypass Example .....	322
7.1.1 Application Overview .....	322
7.1.2 Application Setup.....	323
7.1.3 Configuration and Execution of the Application .....	324
7.2 WLAN-AP BT or BLE Bridge TCPIP Bypass Example .....	328
7.2.1 Application Overview .....	328
7.2.2 Application Setup.....	330
7.2.3 Configuration and Execution of the Application .....	331
7.3 WLAN-STA BT and BLE Bridge TCPIP Bypass with Dual Role Example .....	335
7.3.1 Application Overview .....	335
7.3.2 Application Setup.....	336
7.3.3 Configuration and Execution of the Application .....	337
<b>8 WLAN.....</b>	<b>341</b>
8.1 WLAN Power Numbers Examples.....	341
8.1.1 Tx on periodic wakeup.....	341
8.1.2 WakeOn Wireless.....	345
8.1.3 Without Retention Deepsleep .....	348
8.1.4 WLAN 1Mbps Rx & Listen LP .....	351
8.1.5 WLAN Listen HP .....	354
8.1.6 WLAN Listen LP .....	357
8.1.7 WLAN RX 1Mbps HP .....	360
8.1.8 WLAN RX 6Mbps HP .....	362
8.1.9 WLAN RX 6Mbps LP .....	365
8.1.10 WLAN RX 72Mbps HP .....	367
8.1.11 WLAN Standby.....	370
8.1.12 WLAN TX 6Mbps 10dBm .....	374
8.1.13 WLAN TX 6Mbps 20dBm .....	378
8.2 Access Point Start Example .....	381

---

---

8.2.1 Example Overview .....	381
8.2.2 Example Setup .....	381
8.2.3 Configuration and Execution of the Application .....	382
8.3 AP UDP Echo Example .....	386
8.3.1 Example Overview .....	386
8.3.2 Example Setup .....	386
8.3.3 Configuration and Execution of the Application .....	387
8.4 Concurrent Mode Example .....	392
8.4.1 Example Overview .....	392
8.4.2 Example Setup .....	392
8.4.3 Configuration and Execution of the Application .....	393
8.5 Connection using Asynchronous APIs Example .....	398
8.5.1 Asynchronous APIs Overview .....	398
8.5.2 Example Overview .....	399
8.5.3 Example Setup .....	399
8.5.4 Configuration and Execution of the Application .....	400
8.6 DHCP Option-81 (FQDN) Example .....	403
8.6.1 DHCP Option-81 Overview.....	403
8.6.2 Example Setup .....	404
8.6.3 Configuration and Execution of the Application .....	404
8.7 DHCP User Class Example .....	406
8.7.1 Protocol Overview.....	406
8.7.2 Example Overview .....	406
8.7.3 Example Setup .....	407
8.7.4 Configuration and Execution of the Application .....	407
8.7.5 Executing the Application.....	409
8.8 Enterprise client Example.....	423
8.8.1 Example Overview .....	423
8.8.2 Example Setup .....	424
8.8.3 Configuration and Execution of the Application .....	425
8.9 Firmware Upgradation From Server Example.....	430
8.9.1 Example Overview .....	430
8.9.2 Example Setup .....	430
8.9.3 Configuration and Execution of the Application .....	431

---

---

8.9.4 Executing the Application.....	433
<b>8.10 FTP Client Example .....</b>	<b>434</b>
8.10.1 Protocol Overview.....	434
8.10.2 Example Overview .....	435
8.10.3 Example Setup .....	435
8.10.4 Configuration and Execution of the Application .....	436
8.10.5 Configuration and Execution of the Application .....	439
<b>8.11 HTTP/HTTPS Client Example .....</b>	<b>444</b>
8.11.1 Protocol Overview.....	444
8.11.2 Example Overview .....	444
8.11.3 Example Setup .....	444
8.11.4 Configuration and Execution of the Application .....	445
<b>8.12 HTTP/HTTPS Client Post Data Example .....</b>	<b>451</b>
8.12.1 Protocol Overview.....	451
8.12.2 Example Overview .....	452
8.12.3 Example Setup .....	452
8.12.4 Configuration and Execution of the Application .....	453
<b>8.13 Instant Background Scan Example.....</b>	<b>459</b>
8.13.1 Example Overview .....	459
8.13.2 Example Setup .....	459
8.13.3 Configuration and Execution of the Application .....	460
<b>8.14 MFP Client.....</b>	<b>462</b>
8.14.1 Example Overview .....	462
8.14.2 Application Overview.....	463
8.14.3 Example Setup .....	463
8.14.4 Configuration and Execution of the Application .....	464
<b>8.15 MQTT Client Example.....</b>	<b>468</b>
8.15.1 Protocol Overview.....	468
8.15.2 Example Overview .....	468
8.15.3 Example Setup .....	469
8.15.4 Configuration and Execution of the Application .....	470
8.15.5 Limitations .....	478
<b>8.16 Multicast Example.....</b>	<b>478</b>
8.16.1 Protocol Overview.....	478

---

---

8.16.2 Example Overview .....	478
8.16.3 Example Setup .....	479
8.16.4 Configuration and Execution of the Application .....	480
8.17 Over The Air Firmware Upgardation From Server example.....	484
8.17.1 Example Overview .....	484
8.17.2 Example Setup .....	484
8.17.3 Configuration and Execution of the Application .....	485
8.18 Provisioning Example .....	489
8.18.1 Example Overview .....	489
8.18.2 Example Setup .....	490
8.18.3 Configuration and Execution of the Application .....	491
8.19 Raw Data Example .....	498
8.19.1 Example Overview .....	498
8.19.2 Example Setup .....	499
8.19.3 Configuration and Execution of the Application .....	499
8.20 SMTP Client Example.....	503
8.20.1 Protocol Overview.....	503
8.20.2 Example Overview .....	503
8.20.3 Example Setup .....	503
8.20.4 Configuration and Execution of the Application .....	504
8.21 SMTP-POP3 Client Example .....	510
8.21.1 Application Overview.....	510
8.21.2 Setup required .....	511
8.21.3 Description .....	511
8.21.4 Configuration and Execution of the Application .....	511
8.22 SNTP Client Example .....	517
8.22.1 Protocol Overview.....	517
8.22.2 Example Overview .....	518
8.22.3 Example Setup .....	518
8.22.4 Configuration and Execution of the Application .....	519
8.23 Station Ping Example.....	521
8.23.1 PING Overview.....	521
8.23.2 Example Overview .....	522
8.23.3 Example Setup .....	522

---

---

8.23.4 Configuration and Execution of the Application .....	523
<b>8.24 Store Configuration Profile Example .....</b>	<b>526</b>
8.24.1 Store Configuration Overview .....	526
8.24.2 Application Overview .....	526
8.24.3 Application Setup .....	526
8.24.4 Configuration and Execution of the Application .....	527
<b>8.25 TCP Client Socket Example .....</b>	<b>531</b>
8.25.1 TCP Protocol Overview .....	531
8.25.2 Example Overview .....	531
8.25.3 Example Setup .....	532
8.25.4 Configuration and Execution of the Application .....	533
<b>8.26 TCP Client Socket over SSL Application with Multiple TLS Versions.....</b>	<b>536</b>
8.26.1 SSL Overview .....	536
8.26.2 Application Overview .....	536
8.26.3 Application Setup .....	536
8.26.4 Configuration and Execution of the Application .....	537
<b>8.27 TCP Client Socket over SSL Example .....</b>	<b>541</b>
8.27.1 SSL Overview .....	541
8.27.2 Example Overview .....	541
8.27.3 Example Setup .....	541
8.27.4 Configuration and Execution of the Application .....	542
<b>8.28 TCP Server Socket Example .....</b>	<b>546</b>
8.28.1 TCP Protocol Overview .....	546
8.28.2 Example Overview .....	546
8.28.3 Example Setup .....	547
8.28.4 Configuration and Execution of the Application .....	547
<b>8.29 Throughput Example .....</b>	<b>550</b>
8.29.1 Example Overview .....	550
8.29.2 Example Setup .....	550
8.29.3 Configuration and Execution of the Application .....	551
<b>8.30 UDP Client Socket Example .....</b>	<b>555</b>
8.30.1 UDP Protocol Overview .....	555
8.30.2 Example Overview .....	555
8.30.3 Example Setup .....	555

---

---

8.30.4 Configuration and Execution of the Application .....	556
<b>8.31 UDP Sever Socket Example .....</b>	<b>560</b>
8.31.1 UDP Protocol Overview .....	560
8.31.2 Example Overview .....	560
8.31.3 Example Setup .....	560
8.31.4 Configuration and Execution of the Application .....	561
<b>8.32 Wake-Fi Transmit .....</b>	<b>564</b>
8.32.1 Example Overview .....	564
8.32.2 Example Setup .....	564
8.32.3 Configuration and Execution of the Application .....	565
<b>8.33 Web Socket Example .....</b>	<b>568</b>
8.33.1 WebSocket Overview .....	568
8.33.2 Example Overview .....	568
8.33.3 Example Setup .....	568
8.33.4 Configuration and Execution of the Application .....	569
<b>8.34 WEP Security Example .....</b>	<b>575</b>
8.34.1 WEP protocol Overview .....	575
8.34.2 Example Overview .....	575
8.34.3 Example setup.....	575
8.34.4 Configuration and Execution of the Application .....	576
<b>8.35 Wi-Fi Direct Autonomous GO Example.....</b>	<b>579</b>
8.35.1 Example Overview .....	579
8.35.2 Setup required .....	579
8.35.3 Description .....	580
8.35.4 Configuration and Execution of the Application .....	580
<b>8.36 Wi-Fi Direct Client Example .....</b>	<b>583</b>
8.36.1 Example Overview .....	583
8.36.2 Example Setup .....	584
8.36.3 Configuration and Execution of the Application .....	584
<b>8.37 Wireless Firmware Upgrade Example .....</b>	<b>587</b>
8.37.1 Example Overview .....	587
8.37.2 Example Setup .....	587
8.37.3 Configuration and Execution of the Application .....	588
<b>8.38 WPS Access Point Example.....</b>	<b>593</b>

---

---

8.38.1 WPS Overview .....	593
8.38.2 Example Overview .....	593
8.38.3 Example Setup .....	593
8.38.4 Configuration and Execution of the Application .....	594
<b>8.39 WPS PIN Example.....</b>	<b>597</b>
8.39.1 WPS Overview .....	597
8.39.2 Example Overview .....	598
8.39.3 Example Setup .....	598
8.39.4 Configuration and Execution of the Application .....	599
<b>8.40 WPS Station Example .....</b>	<b>602</b>
8.40.1 WPS Overview .....	602
8.40.2 Example Overview .....	603
8.40.3 Example Setup .....	603
8.40.4 Configuration and Execution of the Application .....	604
<b>8.41 Customized root webpage .....</b>	<b>608</b>
8.41.1 Example Overview .....	608
8.41.2 Example Setup .....	608
8.41.3 Configuration and Execution of the Application .....	609
<b>9 ZIGBEE .....</b>	<b>613</b>
<b>9.1 Zigbee Switch .....</b>	<b>613</b>
9.1.1 Application Overview.....	613
9.1.2 Example Setup .....	613
9.1.3 Configuration and Execution of the Application .....	613
<b>10 WLAN ZIGBEE.....</b>	<b>615</b>
<b>10.1 wlan_ap_zigbee_switch .....</b>	<b>615</b>
10.1.1 Application Overview.....	615
10.1.2 Configuration and Execution of the Application .....	616
<b>10.2 wlan_zigbee_switch .....</b>	<b>619</b>
10.2.1 Application Overview.....	619
10.2.2 Configuring the WLAN task.....	620
10.2.3 Configuring the ZigBee Application .....	622
<b>11 Revision Record.....</b>	<b>623</b>

---





- BT Classic Examples
- BT LE Applications
- Security APIS
- WLAN BLE
- WLAN BT
- WLAN BT BLE
- WLAN
- ZIGBEE
- WLAN ZIGBEE

---

## 1 Introduction to RS9116 & RS14100 Wireless SAPI Examples

This example is applicable to WiSeConnect<sup>TM</sup> and WiSeMCU<sup>TM</sup> parts. For simplicity, this document refers to WiSeConnect, but all discussion applies to both WiSeConnect and WiSeMCU parts. The feature(s) used in this example may or may not be available in your part. Refer to the product datasheet to verify the features available in your part.

## 2 BT Classic Examples

### 2.1 BT PER Example

#### 2.1.1 Application Overview

##### Overview

This application demonstrates how to configure the necessary parameters to start transmit or receive BT PER packets.

##### Sequence of Events

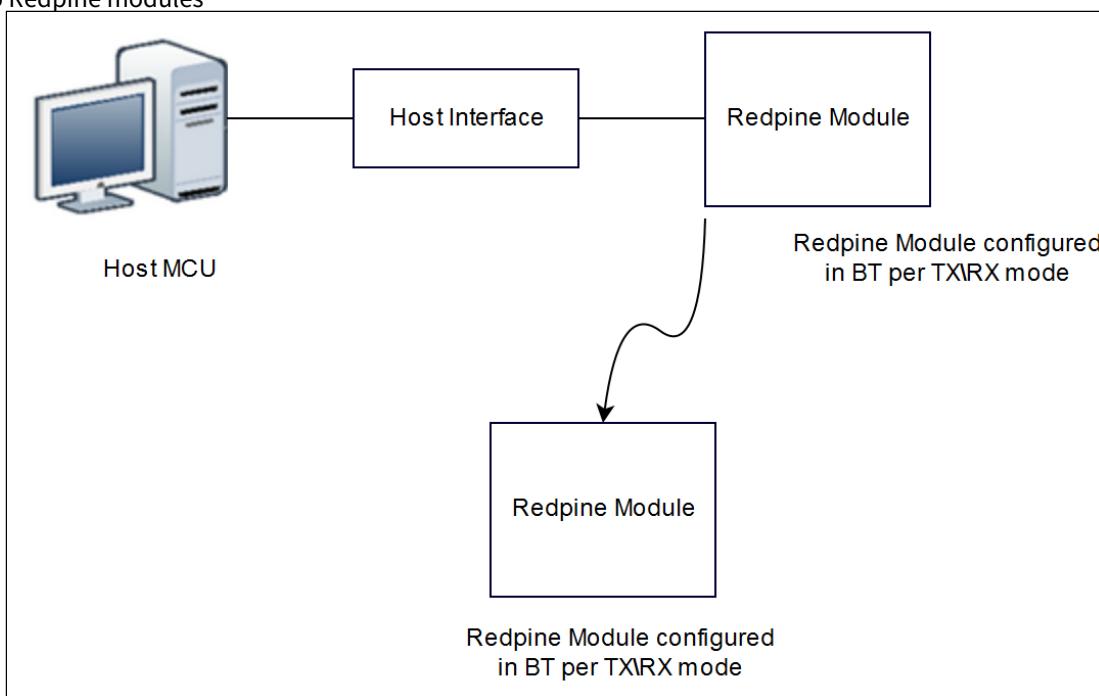
This Application explains user how to:  
Configure the BT PER TX or RX mode.

#### 2.1.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

##### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Two Redpine modules



**Figure 1: Setup Diagram**

### 2.1.3 Configuration and Execution of the Application

#### Configuring the Application

- a. Open **rsi\_bt\_per.c** file and update/modify following macros,  
**RSI\_BT\_LOCAL\_NAME** refers name of the Redpine device.

```
#define RSI_BT_LOCAL_NAME "PER"
```

**RSI\_CONFIG\_PER\_MODE** refers configuration mode BT PER TX or RX

```
#define RSI_CONFIG_PER_MODE  
RSI_BT_PER_RECEIVE_MODE
```

OR

```
#define RSI_CONFIG_PER_MODE  
RSI_BT_PER_TRANSMIT_MODE
```

**#define RSI\_CONFIG\_PER\_MODE RSI\_BT\_PER\_RECEIVE\_MODE**  
CMD\_ID refers the command id for transmit or receive

```
#define BT_TRANSMIT_CMD_ID 0x15
```

```
#define BT_RECEIVE_CMD_ID 0x16
```

**PAYOUT\_TYPE** refers type of payload to be transmitted  
'0' – Payload consists of all zeros

'1' – Payload consists of all 0xFF's  
'2' – Payload consists of all 0x55's  
'3' – Payload consists of all 0xF0's  
'4' – Payload consists of PN9 sequence.

#define SEQUENCE_0	0
#define SEQUENCE_1	1
#define SEQUENCE_2	2
#define SEQUENCE_F0	3
#define SEQUENCE_PRBS	4

#define PAYLOAD\_TYPE SEQUENCE\_F 0

**PACKET\_TYPE:** Type of the packet to be transmitted, as per the Bluetooth standard. Refer Bluetooth Core 5.0 spec.

#define PACKET\_TYPE 15

**PACKET\_LEN:** Length of the packet, in bytes, to be transmitted. Refer Bluetooth Core 5.0 spec.

#define PACKET\_LEN 339

**BT\_RX\_CHNL\_NUM-** Receive channel index, as per the Bluetooth standard. i.e., 0 to 78

**BT\_TX\_CHNL\_NUM -** Transmit channel index, as per the Bluetooth standard. i.e., 0 to 78

#define BT\_RX\_CHNL\_NUM 10

#define BT\_TX\_CHNL\_NUM 10

**SCRAMBLER\_SEED:** Initial seed to be used for whitening. It should be set to '0' in order to disable whitening.

#define SCRAMBLER\_SEED 0

**LINK\_TYPE : ACL\_LINK**

#define ACL\_LINK 1

**TX\_MODE:** Burst mode - 0   Continuous mode - 1

#define BURST\_MODE 0

#define CONTINUOUS\_MODE 1

**HOPPING TYPE :** no hopping -0   fixed hopping - 1   random hopping - 2

#define NO\_HOPPING 0

#define FIXED\_HOPPING

1

#define RANDOM\_HOPPING

2

**ANT\_SEL**: onchip antenna - 2 u.fl - 3

#define ONBOARD\_ANT\_SEL

2

#define EXT\_ANT\_SEL

3

**RF\_TYPE** : External RF - 0 Internal RF - 1

#define BT\_EXTERNAL\_RF

0

#define BT\_INTERNAL\_RF

1

**RF\_CHAIN**: WLAN\_HP\_CHAIN 0  
BT\_HP\_CHAIN 2

#define WLAN\_HP\_CHAIN\_BIT

0

#define BT\_HP\_CHAIN\_BIT

2

pll\_mode: PLL\_MODE0 - 0 PLL\_MODE1 - 1

#define PLL\_MODE\_0

0

#define PLL\_MODE\_1

1

**LOOP\_BACK\_MODE** : enable 1 or disable 0

#define LOOP\_BACK\_MODE\_DISABLE

0

#define LOOP\_BACK\_MODE\_ENABLE

1

Following are the **non-configurable** macros in the application.

**BT\_GLOBAL\_BUFF\_LEN** refers the number of bytes required by the application and the driver

#define BT\_GLOBAL\_BUFF\_LEN

10000

b. Open *rsi\_wlan\_config.h* file and update/modify following macros:

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Executing the Application

- a. Power on the WiSeConnect or WiSeMCU module and run the sapis rsi\_bt\_per application.
- b. After the program gets executed, Redpine module starts BT PER transmit or BT PER receive.
- c. For receiving purpose use other WiSeConnect or WiSeMCU module and keep it in BT PER RX mode.
- d. Check for BT PER stats whatever configured values are effecting or not.

## 2.2 BT SPP Master Example

### 2.2.1 Application Overview

#### Overview

This application demonstrates how to configure the device in Master mode and establish SPP profile connection with remote slave device and data exchange between two devices using SPP profile.  
In this Application, Redpine module configures in Master mode and initiates basic connection with remote slave device. After successful basic connection, Application waits to accept SPP profile level connection from remote device. Once SPP connection success, Application will wait for data to receive from connected remote device. If remote device sends data to redpine module, redpine module receives the data and send back the same data to remote device using SPP profile.

#### Sequence of Events

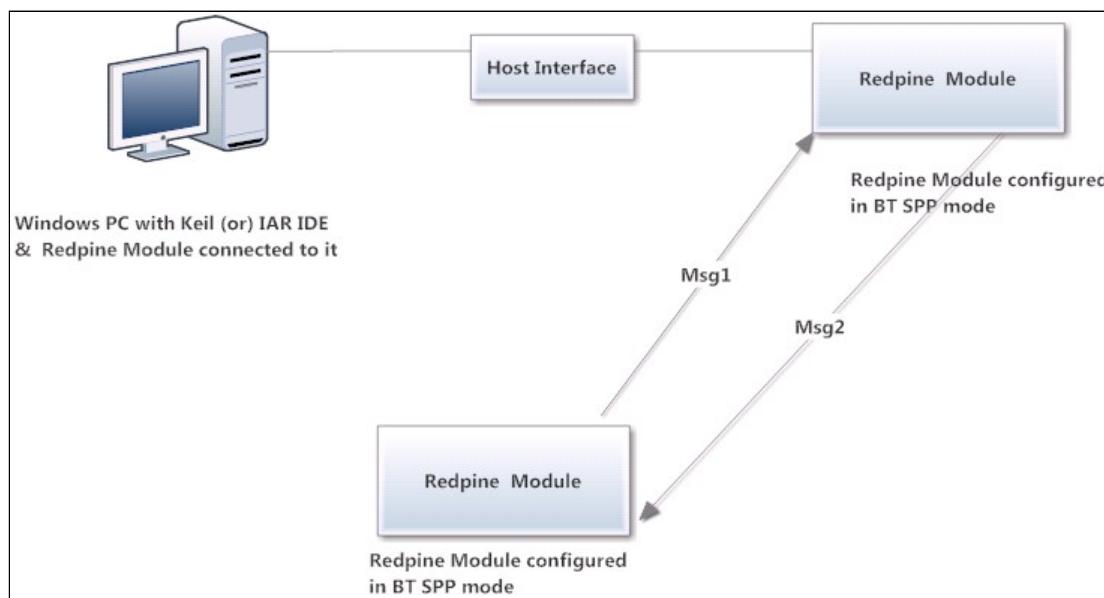
This Application explains user how to:

- Configure redpine module to act as Master
- Connect the redpine module with the Slave
- Accept SPP level connection from the Smartphone
- Loop back the received messaged

### 2.2.2 Application Setup

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE
- Redpine Module



**Figure 1: Setup Diagram**

### 2.2.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open ***rsi\_spp\_master.c*** file and update/modify following macros,  
**RSI\_BT\_LOCAL\_ANME** refers name of the WiSeConnect device.

```
#define RSI_BT_LOCAL_NAME "SPP_MASTER"
```

**PIN\_CODE** refers four bytes string required for pairing process.

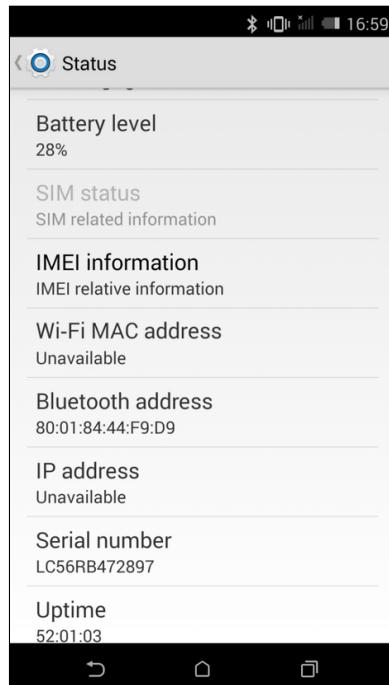
```
#define PIN_CODE "4321"
```

**REMOTE\_BD\_ADDR** refers Remote device BD address to connect.  
Provide the Smart phone BD address,

```
#define REMOTE_BD_ADDR "00:1B:DC:  
07:2C:F0"
```

**Note:**

In the smartphone, User Can check the BD address of Bluetooth device in the following location: Settings/  
Aboutphone/status/Bluetooth Address



Following are the **non-configurable** macros in the application.

**BT\_GLOBAL\_BUFF\_LEN** refers the number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN 10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Role Switch Configuration

Following 3 API's used to get the role,to set the role and to know the status of the role switch

```
//To know the role of the device
rsi_bt_get_local_device_role((int8_t *)str_conn_bd_addr,
&device_state);

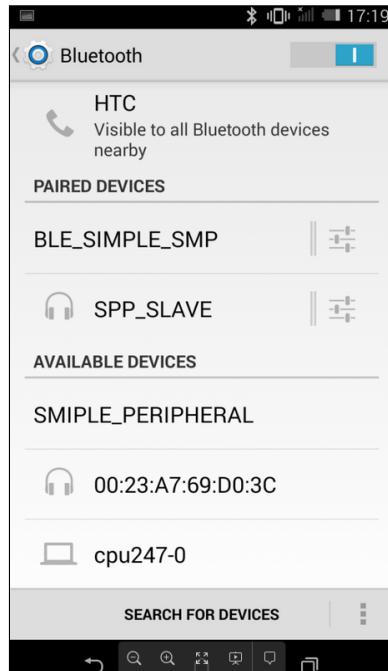
//To set the device role to either Master or Slave.(set_role = 0 -->Master, set_role = 1 -->Slave)
rsi_bt_set_local_device_role((int8_t *)str_conn_bd_addr, set_role,
&device_state);

//To know the status of the Switch Role
role_change (uint16_t resp_status, rsi_bt_event_role_change_t
*role_change_status1);
```

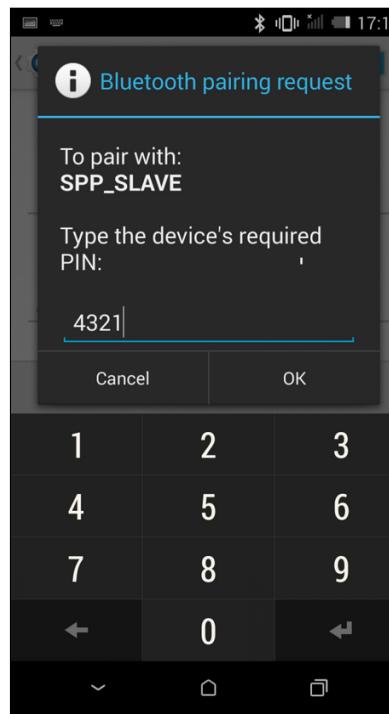
The status of the role\_change function should return success for successful role switch, otherwise fail.

### Executing the Application

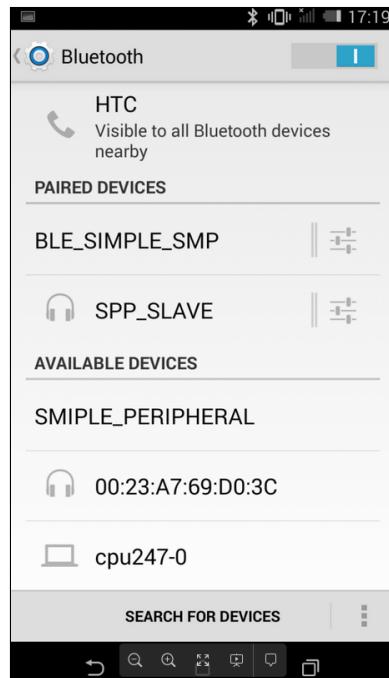
1. Power on Bluetooth in smart phone and put it in visible mode to all Bluetooth devices.



2. After the program gets executed, Redpine module initiates basic connection with the remote device (Smart phone). User has to provide **PIN\_CODE** at remote device for successful connectivity. Please find below images for connection at remote device.



3. After successful connection, In smart phone Redpine module lists under Paired devices.



4. After successful connection, Open Sena BT term Bluetooth serial app on mobile which will be in discoverable mode and do the scan from Redpine module . After successful scan, Redpine module can initiate connection to already bonded device as we have already completed basic pairing from Redpine module.

5. At remote device, Bluetooth pairing request will pop-up for SPP connection success. providing secret key (PIN\_CODE) for SPP connection success.

6. Send some data (Ex: "redpine signals") from remote device to Redpine module e and some data from Redpine device to remote device

## 2.3 BT SPP Slave Example

### 2.3.1 Application Overview

#### Overview

This application demonstrates how to configure the device in Slave mode and establish SPP profile connection with remote Master device and data exchange between two devices using SPP profile.

In this Application, Redpine module configures in Slave mode and waits to accept SPP profile level connection from remote device. After successful SPP connection, Application will wait for data to receive from connected remote device. If remote device sends data to Redpine module, Redpine module receives the data and send back the same data to remote device using SPP profile.

#### Sequence of Events

This Application explains user how to:

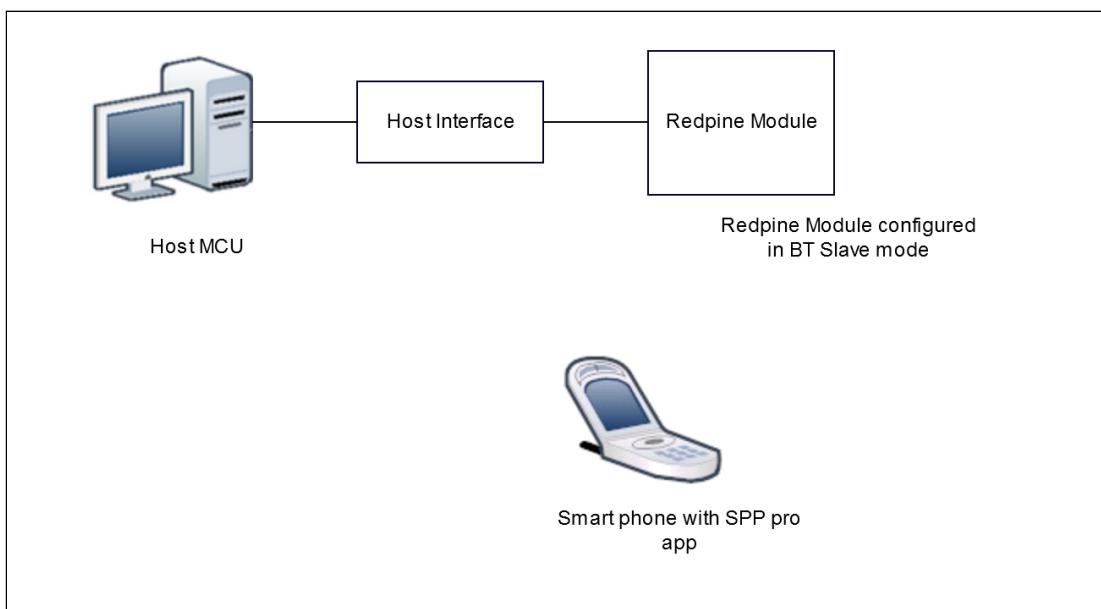
- Configure Redpine module to act as Slave
- Configure device in discoverable and connectable mode
- Accept SPP level connection from the Smartphone
- Loop back the received messages

### 2.3.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BT master device



**Figure 1: Setup Diagram**

### 2.3.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open ***rsi\_spp\_slave.c*** file and update/modify following macros,  
**RSI\_BT\_LOCAL\_ANME** refers name of the Redpine module to appear during scanning by remote devices.

```
#define RSI_BT_LOCAL_NAME  
"SPP_SLAVE"
```

**PIN\_CODE** refers four bytes string required for pairing process.

```
#define PIN_CODE "4321"
```

Following are the **non-configurable** macros in the application.

**BT\_GLOBAL\_BUFF\_LEN** refers to the number of bytes required by the application and the driver.

```
#define BT_GLOBAL_BUFF_LEN 10000
```

2. Open ***rsi\_wlan\_config.h*** file and update/modify following macros:

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Role Switch Configuration

Following 3 API's used to get the role,to set the role and to know the status of the role switch

```
//To know the role of the device
rsi_bt_get_local_device_role((int8_t *)str_conn_bd_addr,
&device_state);

//To set the device role to either Master or Slave.(set_role = 0 -->Master, set_role = 1 -->Slave)
rsi_bt_set_local_device_role((int8_t *)str_conn_bd_addr, set_role,
&device_state);

//To know the status of the Switch Role
role_change (uint16_t resp_status, rsi_bt_event_role_change_t
*role_change_status1);
```

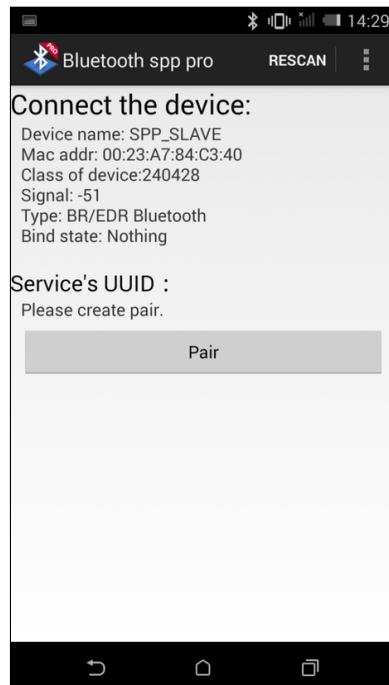
The status of the role\_change function should return success for successful role switch, otherwise fail.

## Executing the Application

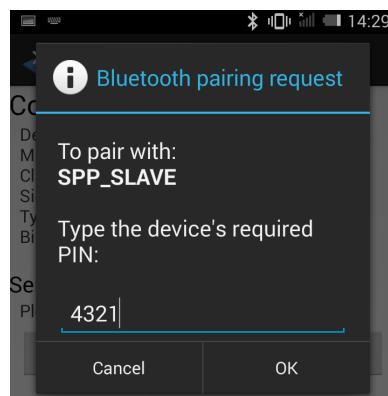
1. After the program gets executed, Redpine module initializes the SPP profile and waits for the incoming connection.
2. Open Bluetooth SPP pro app on mobile and do the scan until Redpine module (Ex: "SPP\_SLAVE") gets present in the scan list.



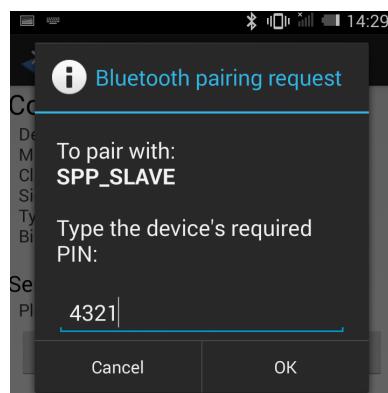
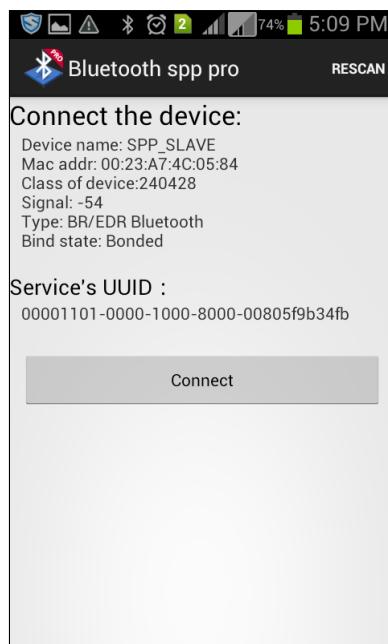
3. After the successful scan, select the device and initiate pairing to Redpine module.



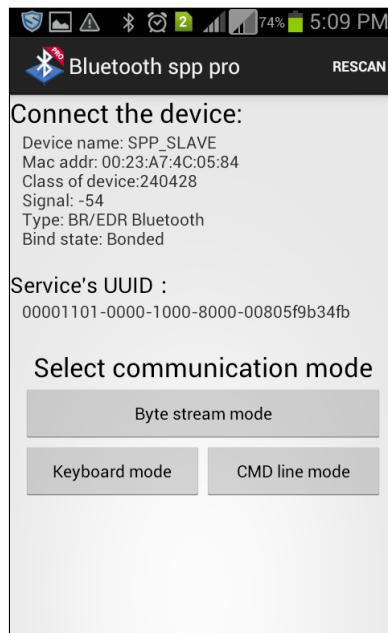
4. After initiating paring, Pairing request will pop-up at smartphone side and issue secret key which is given at Redpine module (PIN\_CODE ) side.



5. After successful pair, initiate SPP connection to Redpine module and give the secret key for receiving pairing request at remote device side.



6. After successful SPP connection, select "Byte stream mode" to send and receive the data.



7. Send some data (Ex: "redpine signals") from the remote device to Redpine device and same data will send back from Redpine device to remote device. Please refer the given image for sending and receiving data from the remote device.



---

## 2.4 SSP Test Example

### 2.4.1 Application Overview

#### Overview

This application demonstrates how to configure the device in Slave mode and establish SPP profile connection with remote Master device using secure simple paring (SSP) and data exchange between two devices using SPP profile. In this Application, Redpine module configures in Slave mode and waits to accept SPP profile level connection using secure simple pairing (SSP) from remote device. After successful SPP connection, Application will wait for data to receive from connected remote device. If remote device sends data to Redpine module, Redpine module receives the data and send back the same data to remote device using SPP profile.

#### Sequence of Events

This Application explains user how to:

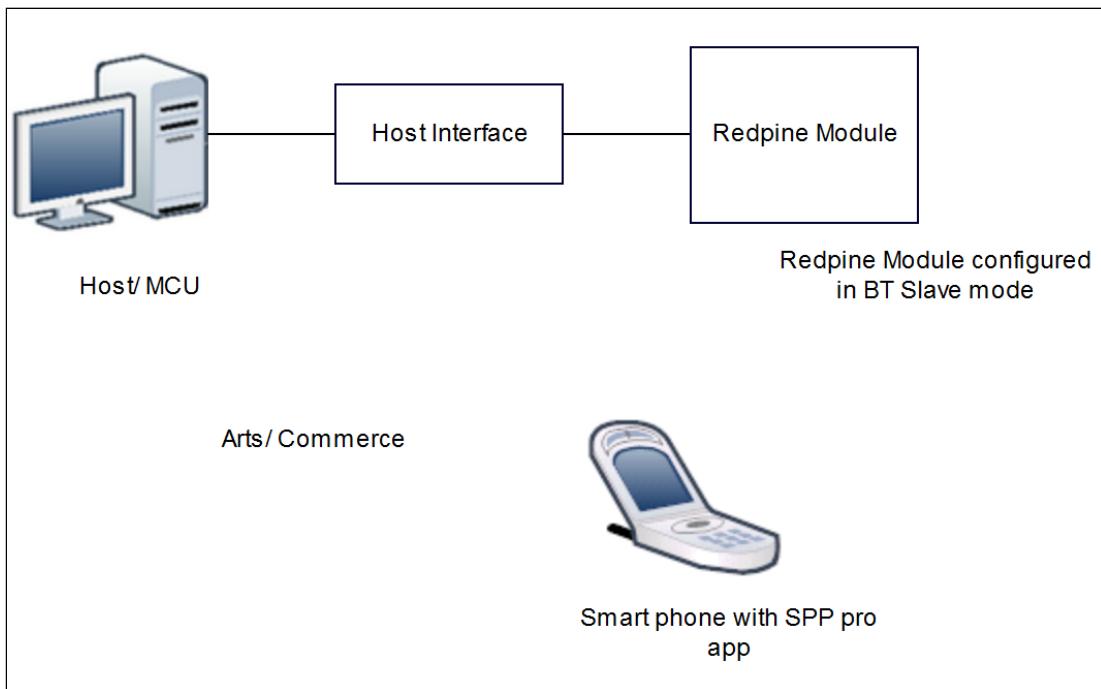
- Configure Redpine module to act as Slave
- Configure device to secure simple pairing (SSP)
- Configure device in discoverable and connectable mode
- Accept SPP level connection from the Smartphone
- Loop back the received messaged

### 2.4.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Mobile with spp application



**Figure 1: Setup Diagram**

#### 2.4.3 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_ssp\_test\_app.c** file and update/modify following macros:  
**RSI\_BT\_LOCAL\_ANME** refers name of the Redpine module to appear during scanning by remote devices.

```
#define RSI_BT_LOCAL_NAME  
"SPP_SLAVE"
```

**PIN\_CODE** refers four bytes string required for pairing process.

```
#define PIN_CODE "1234"
```

Following are the **non-configurable** macros in the application:

**BT\_GLOBAL\_BUFF\_LEN** refers to the number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN 10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

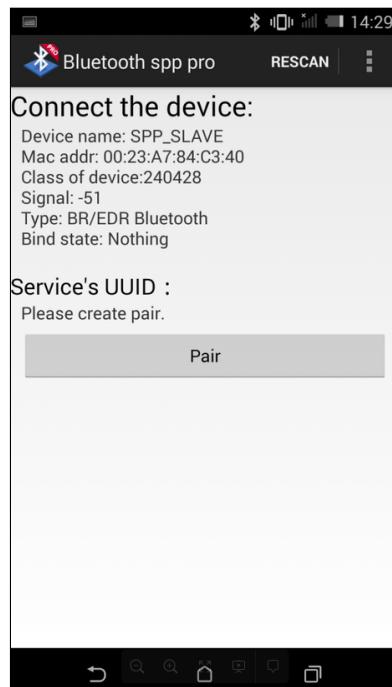
```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Executing the Application

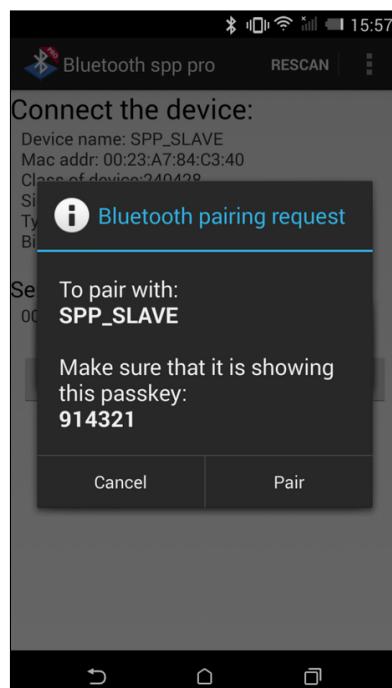
1. After the program gets executed, Redpine module initializes the SPP profile and waits for the incoming connection.
2. Open Bluetooth SPP pro app on mobile and do the scan until Redpine device (Ex: "SPP\_SLAVE") gets present in the scan list.



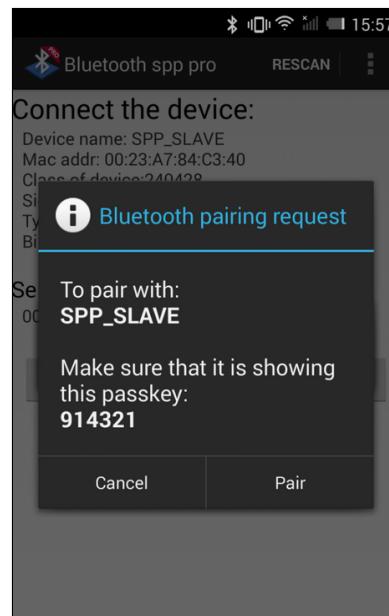
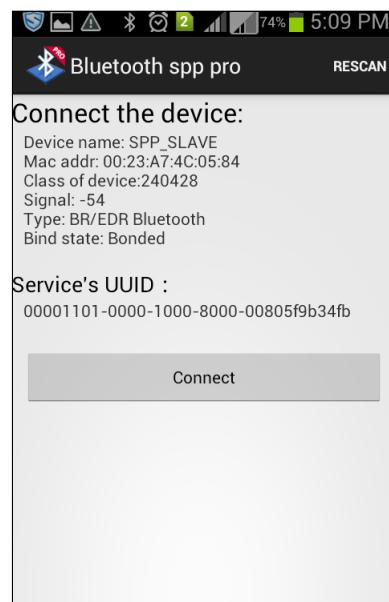
3. After successful scan, select the device and initiate pairing to Redpine device.



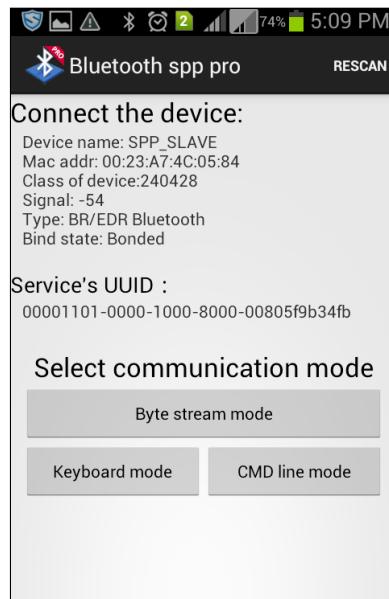
4. After initiating paring, Pairing request will pop-up at smart phone side and accept the pairing request.



5. After initiating paring, Pairing request will pop-up at smart phone side and accept the pairing request.



6. After successful SPP connection, select "Byte stream mode" to send and receive the data.



7. Send some data (Ex: "redpine signals") from remote device to Redpine device and same data will send back from Redpine device to remote device. Please find below image for sending and receiving data from remote device.



---

## 2.5 BT A2DP Source with AVRCP Example

### 2.5.1 Application Overview

#### Overview

A2DP source with AVRCP application is used to transfer the PCM data to remote device (A2DP sink device) in SBC encoded format and acts as a AVRCP target.

#### Sequence of Events

Applications explains user how to:

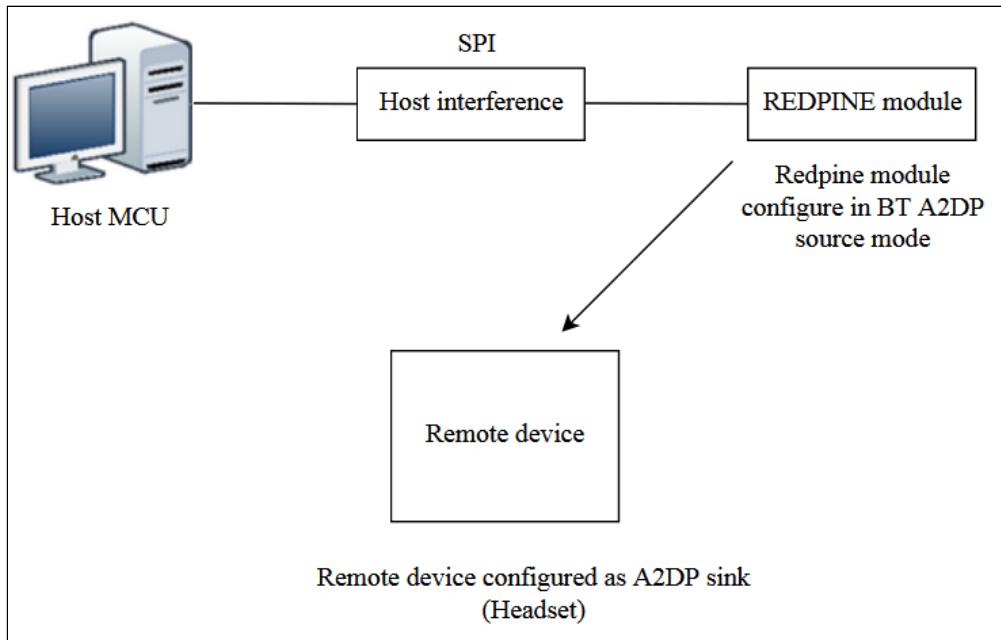
- Configure Redpine device in discoverable and connectable mode
- Configure the remote device in discoverable and connectable mode
- Initialize the physical level connection with remote device
- Initialize A2DP profile level connection with remote device
- Initialize AVRCP profile level connection with remote device
- Transfer the Audio data to the remote device.
- Pause and Play of Audio data will controlled by AVRCP profile based on the Remote device inputs.

### 2.5.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### Setup Requirements

- Redpine module
- BT Remote device(A2DP sink device)
- Host platform



**Figure 1: Setup Diagram**

### 2.5.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_a2dp\_source\_avrcp.c** file and update/modify following macros,  
**RSI\_BT\_LOCAL\_NAME** refers name of the Redpine module to appear during scanning by remote devices.

```
#define RSI_BT_LOCAL_NAME  
"A2DP_AVRCP_SOURCE"
```

**RSI\_BT\_REMOTE\_BD\_ADDR** refers BD address of the Remote device to which redpine device has to connect.

```
#define RSI_BT_REMOTE_BD_ADDR  
"00:1E:7C:25:E9:6D"
```

**PIN\_CODE** refers four bytes string required for pairing process.

```
#define PIN_CODE  
"0000"
```

**RSI\_AUDIO\_DATA\_TYPE** refers input audio file format. Select MP3\_AUDIO for .mp3 format, PCM\_AUDIO for .wav format, SBC\_AUDIO for .sbc format

```
#define PCM_AUDIO  
1  
#define SBC_AUDIO  
2  
#define MP3_AUDIO  
3  
#define RSI_AUDIO_DATA_TYPE  
MP3_AUDIO
```

**RSI\_AUDIO\_DATA\_SRC** refers audio source type. Select BIN\_FILE, if we have audio files and ARRAY if file system is not supported in host, we can pass buffered data.

```
#define BIN_FILE  
1  
#define ARRAY  
2  
#define RSI_AUDIO_DATA_SRC  
BIN_FILE
```

**BIN\_FILE\_PATH** refers input audio file path.

```
#define AUDIO_FILE_PATH  
"mp3_samples.mp3"
```

2. Following are the **non-configurable** macros in the application. **BT\_GLOBAL\_BUFF\_LEN** refers to the number of bytes required by the application and the driver.

```
#define BT_GLOBAL_BUFF_LEN  
10000
```

```
#define A2DP_BURST_MODE  
1
```

**Note:**  
**A2DP\_BURST\_MODE should be 1**

3. Open *rsi\_wlan\_config.h* file and update/modify following macros:

```
#define CONCURRENT_MODE  
RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP  
FEAT_SECURITY_OPEN  
#define RSI_TCP_IP_BYPASS  
RSI_ENABLE  
#define RSI_TCP_IP_FEATURE_BIT_MAP  
1  
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP  
EXT_FEAT_256K_MODE  
#define RSI_BAND  
RSI_BAND_2P4GHZ
```

## Executing the Application

1. Keep Remote device in discoverable and connectable mode.
2. Run the A2DP source AVRCP application
3. Now Redpine module is trying to connect with Remote device.
4. After Successful connection, Redpine device will connect with remote device in A2DP level and AVRCP level
5. After Successful A2DP and AVRCP connections, Redpine device will transfer the Audio data to remote device .
6. Redpine device will also acts as AVRCP target to support the pause and play options from the A2DP sink device(Remote device)

**Note**

- This can be run in wise-connect mode in linux pc with USB Interface.

## 2.6 BT A2DP Sink Example

### 2.6.1 Application Overview

#### Overview

A2DP sink. is used to receive the audio(SBC) data from remote A2DP source device and play through the headset.

#### Sequence of Events

Applications explains user how to:

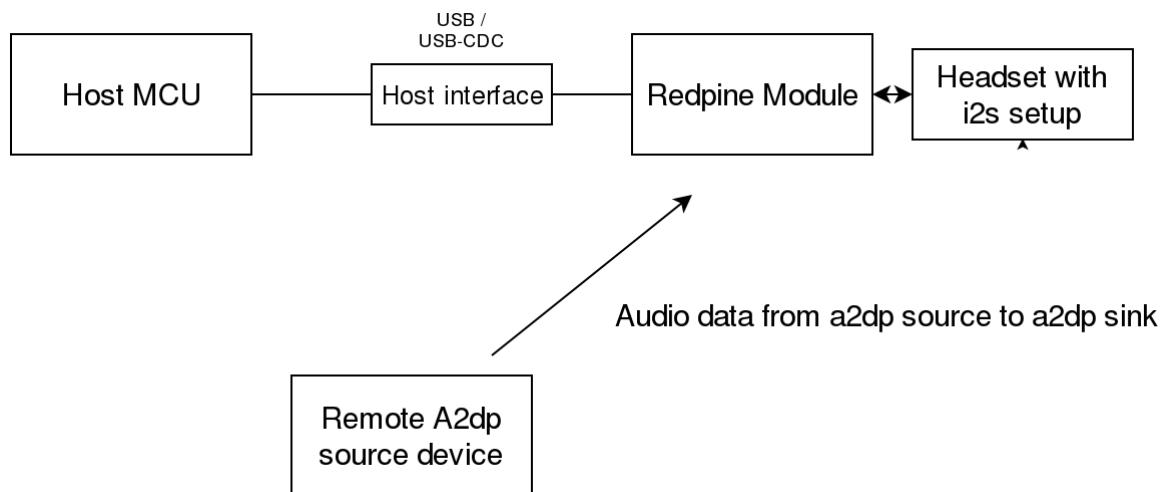
- Configure Redpine device in discoverable and connectable mode
- Initialize A2DP profile
- Initialize SBC
- Register a2dp event call backs

### 2.6.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### Setup Requirements

- Redpine module
- BT Remote device(A2DP source device)
- Host platform
- I2S setup
- Wired Headset(connected to I2S board)



**Figure 1: Setup Diagram**

### 2.6.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open ***rsi\_a2dp\_sink.c*** file and update/modify following macros,  
**RSI\_BT\_LOCAL\_NAME** refers name of the Redpine module to appear during scanning by remote devices.

```
#define RSI_BT_LOCAL_NAME  
"A2DP_sink"
```

**PIN\_CODE** refers four bytes string required for pairing process.

```
#define PIN_CODE  
"0000"
```

Following are the **non-configurable** macros in the application.

**BT\_GLOBAL\_BUFF\_LEN** refers to the number of bytes required by the application and the driver.

```
#define BT_GLOBAL_BUFF_LEN  
10000
```

2. Open ***rsi\_bt\_config.h*** file and update/modify following macros:

```
#define A2DP_PROFILE_ROLE  
A2DP_SINK_ROLE
```

Open ***rsi\_wlan\_config.h*** file and update/modify following macros:

```
#define CONCURRENT_MODE  
RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP  
FEAT_SECURITY_OPEN  
#define RSI_TCP_IP_BYPASS  
RSI_ENABLE  
#define RSI_TCP_IP_FEATURE_BIT_MAP  
1  
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP  
EXT_FEAT_256K_MODE  
#define RSI_BAND  
RSI_BAND_2P4GHZ
```

## Executing the Application

1. Configure I2S board .
2. Run the A2DP sink application
3. Now Redpine module is in discoverable and connectable mode.
4. Now Give connection from remote a2dp source device.
5. After Successful connection, Redpine device is ready to receive audio data.
6. Whenever a2dp source send audio data (playing music), Redpine module will receive audio data and route to I2S.
7. User can listen audio data(music) through headset which is connected to I2S setup.

**Note:**

I2S pins we are using for a2dp are:

- SCLK: - GPIO - 28
- LRCLK: - GPIO - 27
- DIN: - GPIO - 29
- DOUT: - GPIO - 30

**Note:**

Host interfaces supported for A2DP sink are USB , USB-CDC and UART.

## 3 BT LE Applications

### 3.1 Heart Rate Example

#### 3.1.1 Application Overview

##### Overview

This application demonstrates how to configure Heart rate as GATT server in BLE peripheral mode and explains how to do indicate operation with GATT server from connected remote device using GATT client.

In this Application, Heart rate GATT server configures with heart rate service with indicate characteristic UUID. When connected remote device writes data to writable characteristic UUID, WiseConnect device receives the data which is received on writable characteristic UUID and writes the same data to readable characteristic UUID and sends indications to the connected device (or) remote device can read the same data using read characteristic UUID if indication enabled on client side.

##### Sequence of Events

This Application explains user how to:

- Create Heart rate service
- Make the device to advertise
- Connect from remote BTLE device
- Receive the message from the connected peer/Smartphone
- Give the indications to connected device

#### 3.1.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

##### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE Smart Phone with GATT client

##### Note

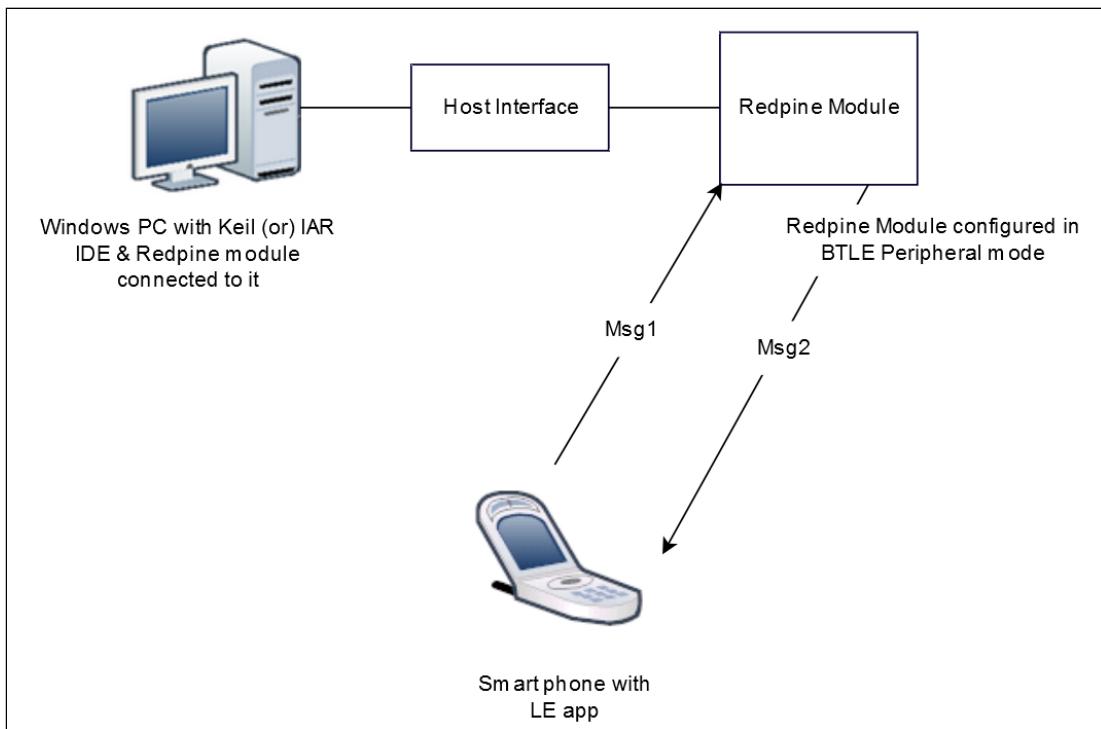
Install Light blue App for tablet for ipad mini and BLE scanner app for android smart phone.

User can download the BLE scanner App from the following link

<https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner&hl=en>

User can download the Light blue App from the following link

<https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer&hl=en>



**Figure 1: Setup Diagram**

### 3.1.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_heart\_rate.c** file and update/modify following macros,  
**RSI\_BLE\_HEART\_RATE\_UUID** refers to the attribute value of the newly created service.

```
#define RSI_BLE_HEART_RATE_SERVICE_UUID 0x180D
```

**RSI\_BLE\_HEART\_RATE\_MEASUREMENT\_UUID** refers to the attribute type of the first attribute under this service (**RSI\_BLE\_HEART\_RATE\_SERVICE\_UUID**).

```
#define RSI_BLE_HEART_RATE_MEASUREMENT_UUID 0x2A37
```

**RSI\_BLE\_SENSOR\_LOCATION\_UUID** refers to the attribute type of the second attribute under this service (**RSI\_BLE\_HEART\_RATE\_SERVICE\_UUID**).

```
#define RSI_BLE_SENSOR_LOCATION_UUID 0x2A38
```

**RSI\_BLE\_HEART\_RATE\_CONTROL\_POINT\_UUID** refers to the attribute type of the second attribute under this service (**RSI\_BLE\_HEART\_RATE\_SERVICE\_UUID**).

```
#define RSI_BLE_HEART_RATE_CONTROL_POINT_UUID 0x2A39
```

**RSI\_BLE\_MAX\_DATA\_LEN** refers to the Maximum length of the attribute data.

```
#define RSI_BLE_MAX_DATA_LEN 20
```

**BLE\_HEART\_RATE\_PROFILE** refers name of the Repine device to appear during scanning by remote devices.

```
#define RSI_BLE_HEART_RATE_PROFILE  
"BLE_HEART_RATE_PROFILE"
```

Following are the **non-configurable** macros in the application.

**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.

```
#define RSI_BLE_CHAR_SERV_UUID 0x2803
```

**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

```
#define RSI_BLE_CLIENT_CHAR_UUID 0x2902
```

**RSI\_BLE\_ATT\_PROPERTY\_READ** is used to set the read property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_READ 0x02
```

**RSI\_BLE\_ATT\_PROPERTY\_WRITE** is used to set the WRITE property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_WRITE 0x08
```

**RSI\_BLE\_ATT\_PROPERTY\_NOTIFY** is used to set the NOTIFY property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_NOTIFY 0x10
```

**RSI\_BLE\_ATT\_PROPERTY\_INDICATE** is used to set the INDICATE property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_INDICATE 0x20
```

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN 15000
```

**GATT\_ROLE** refers the role of the Redpine module to be selected.

If user configure **SERVER**, Redpine module will act as GATT SERVER, means will add heart rate profile.  
If user configure **CLIENT**, Redpine module will act as GATT CLIENT, means will connect to remote GATT server and get services.

```
#define GATT_ROLE "SERVER"
```

**RSI\_BLE\_DEV\_ADDR\_TYPE** refers address type of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR_TYPE  
LE_PUBLIC_ADDRESS
```

Valid configurations based on address type of the remote device are  
LE\_RANDOM\_ADDRESS  
LE\_PUBLIC\_ADDRESS

**Note:**

Depends on the remote device, address type will be changed.

**RSI\_BLE\_DEV\_ADDR** refers address of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR "00:1A:  
7D:DA:71:13"
```

**RSI\_REMOTE\_DEVICE\_NAME** refers the name of remote device to which Redpine device has to connect

```
#define RSI_REMOTE_DEVICE_NAME  
"BLE_PERIPHERAL"
```

**Note:**

Redpine module can connect to remote device by referring either RSI\_BLE\_DEV\_ADDR or RSI\_REMOTE\_DEVICE\_NAME of the remote device.

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

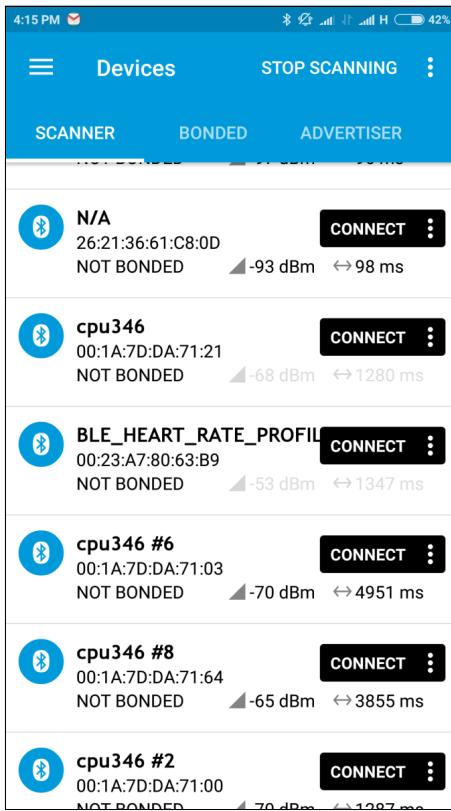
```
#define RSI_BLE_PWR_INX 8
#define RSI_BLE_PWR_SAVE_OPTIONS 0
#define RSI_DUTY_CYCLING 0
```

**Note:**

rsi\_wlan\_config.h and rsi\_ble\_config.h files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

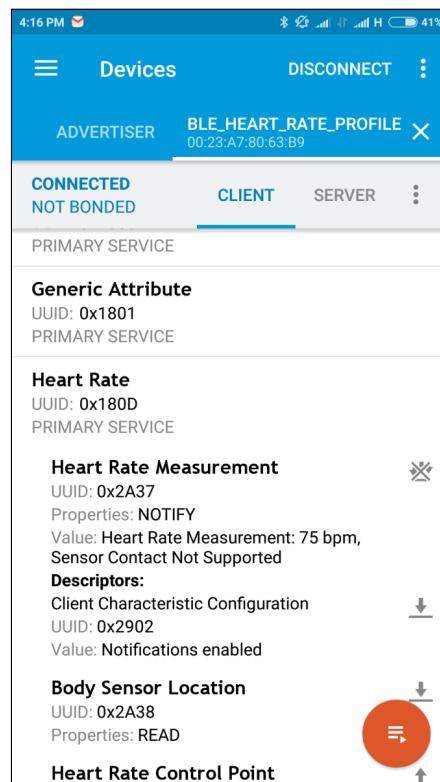
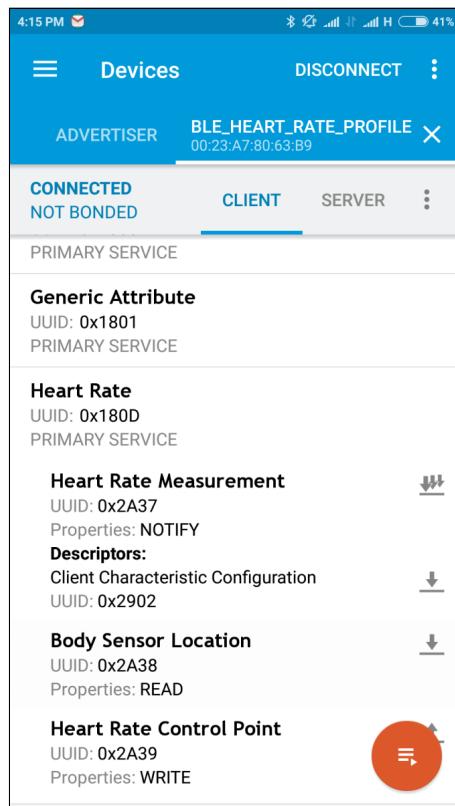
1. After the program gets executed, Redpine will be in Advertising state.
2. Open a LEApp in the Smartphone and do the scan.
3. In the App, Redpine module device will appear with the name configured in the macro **RSI\_BLE\_HEART\_RATE\_PROFILE** (Ex: "BLE\_HEART\_RATE\_PROFILE") or sometimes observed as Redpine device as internal name "SimpleBLEPeripheral".



4. Initiate connection from the App.
5. After successful connection, LE scanner displays the supported services of Redpine module.
6. Select the attribute service which is added **RSI\_BLE\_HEART\_RTAE\_PROFILE\_UUID**
7. Enable notify for the characteristic **RSI\_BLE\_HEART\_RATE\_MEASUREMENT\_UUID**

So that GATT server indicates when value updated in that particular attribute.

8. Whenever the value is updated at server it will be notified to the client which can be read at Heart\_Rate\_Measurement attribute.
9. Please refer the below images for notify operation from remote device GATT client.



## 3.2 Simple Central Example

### 3.2.1 Application Overview

#### Overview

This application demonstrates how to connect with remote BLE device in BLE central mode.

#### Sequence of Events

This Application explains user how to:

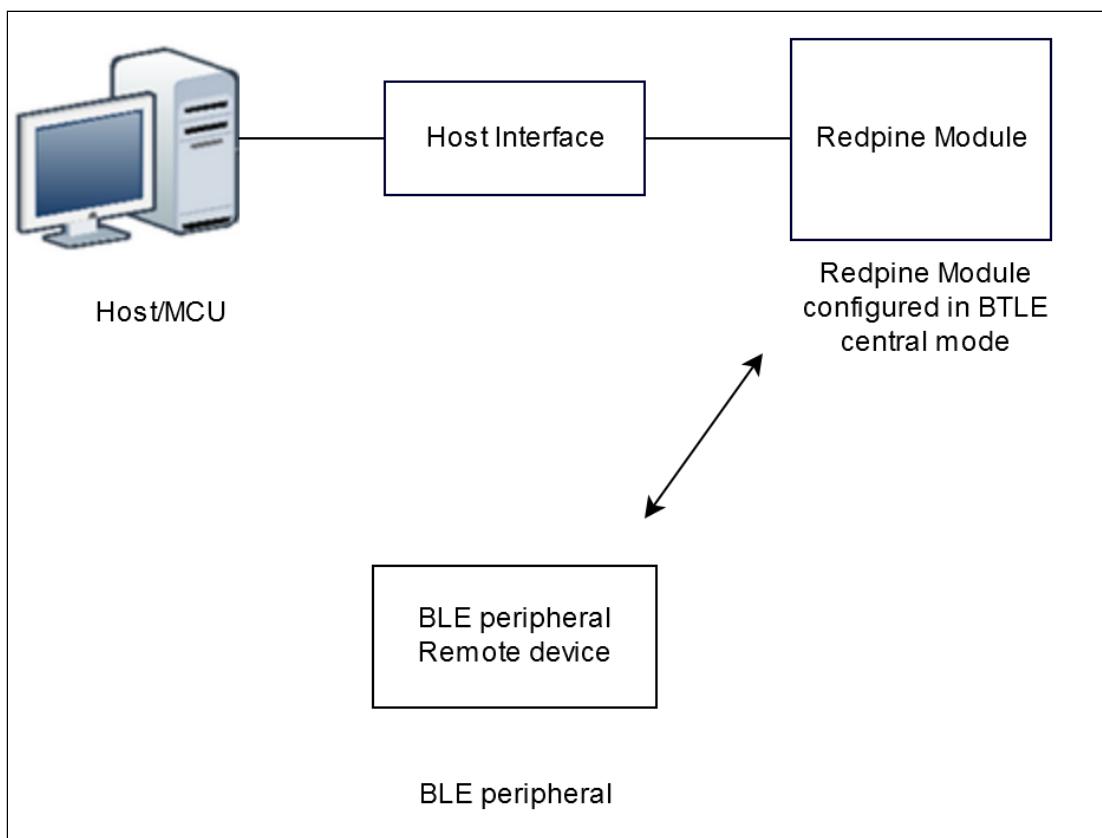
- Connect with remote BTLE peripheral device.

### 3.2.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE peripheral device



**Figure 1: Setup Diagram**

### 3.2.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_central.c** file and update/modify following macros,

**RSI\_BLE\_DEV\_ADDR\_TYPE** refers address type of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR_TYPE  
LE_PUBLIC_ADDRESS
```

Based on address type of remote device, valid configurations are  
**LE\_RANDOM\_ADDRESS**  
**LE\_PUBLIC\_ADDRESS**

**RSI\_BLE\_DEV\_ADDR** refers address of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR  
"00:1A:  
7D:DA:71:13"
```

**RSI\_REMOTE\_DEVICE\_NAME** refers the name of remote device to which Redpine device has to connect

```
#define RSI_REMOTE_DEVICE_NAME  
"BLE_PERIPHERAL"
```

**Note:**

user can configure either RSI\_BLE\_DEV\_ADDR or RSI\_REMOTE\_DEVICE\_NAME of the remote device.  
Following are the event numbers for advertising, connection and Disconnection events,

#define RSI_APP_EVENT_ADV_REPORT	0
#define RSI_APP_EVENT_CONNECTED	1
#define RSI_APP_EVENT_DISCONNECTED	2

Following are the non-configurable macros in the application.

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN 10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE  
RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP  
FEAT_SECURITY_OPEN  
#define RSI_TCP_IP_BYPASS  
RSI_DISABLE  
#define RSI_TCP_IP_FEATURE_BIT_MAP  
TCP_IP_FEAT_DHCPV4_CLIENT  
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP  
EXT_FEAT_256K_MODE  
#define RSI_BAND  
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

#define RSI_BLE_PWR_INX	8
#define RSI_BLE_PWR_SAVE_OPTIONS	0
#define RSI_DUTY_CYCLING	0

**Note**

rsi\_wlan\_config.h and rsi\_ble\_config.h files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the remote ble device in peripheral mode and put it in advertising mode.
2. After the program gets executed, Redpine device tries to connect with the remote device specified in **RSI\_BLE\_DEV\_ADDR** or **RSI\_REMOTE\_DEVICE\_NAME** macro
3. Observe that the connection is established between the desired device and Redpine device.

**Note**

Examples for ble peripherals : Blue tooth Dongle,mobile application, TA sensor tag

## 3.3 Simple Peripheral Example

### 3.3.1 Application Overview

This application demonstrates how to configure the device in simple peripheral mode and how to get connected from the remote Central device.

#### Sequence of Events

This Application explains user how to:

- Set a local name for the device
- Configure the device to advertise
- Start advertising
- Continue advertising even after disconnection with the peer

### 3.3.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable the variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and do not require host interface initialization.

**Note:**

Install Light blue App on the tablet for iPad mini and BLE scanner app for Android smartphone.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU

- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE central device

**Note**

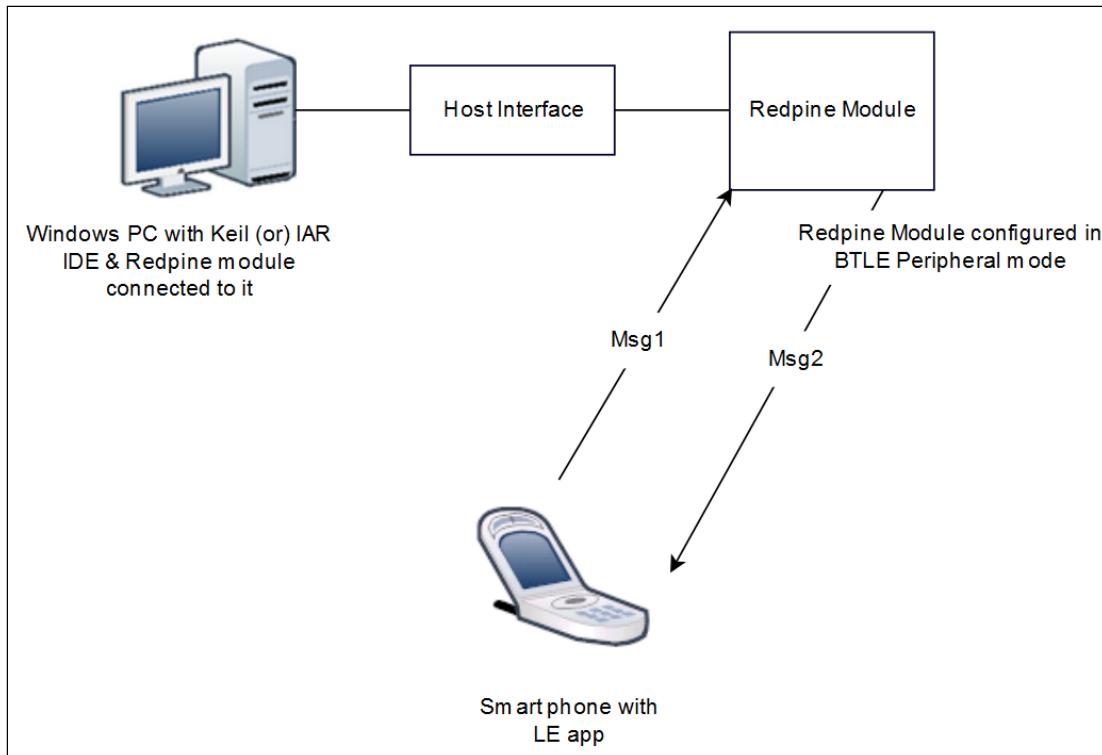
Install Light blue App on the tablet for iPad mini and BLE scanner app for Android smartphone

user can download the BLE scanner App from the following link

<https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner&hl=en>

user can download the Light blue App from the following link

<https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer&hl=en>



**Figure 1: Setup Diagram**

### 3.3.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_simple\_peripheral.c** file and update/modify following macros:  
**RSI\_BLE\_LOCAL\_NAME** refers the name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_LOCAL_NAME  
"WYZBEE_PERIPHERAL"
```

**RSI\_SEL\_ANTENNA** refers to the antenna which is to be used by Redpine module.  
If the user using internal antenna then set,

```
#define RSI_SEL_ANTENNA  
RSI_SEL_INTERNAL_ANTENNA
```

If the user using an external antenna (U.FL connector) then set, **RSI\_SEL\_EXTERNAL\_ANTENNA**.  
Following are the **non-configurable** macros in the application.  
Following are the event numbers for connection and Disconnection events.

```
#define RSI_APP_EVENT_CONNECTED  
#define RSI_APP_EVENT_DISCONNECTED
```

1

2

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN  
10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE  
RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP  
FEAT_SECURITY_OPEN  
#define RSI_TCP_IP_BYPASS  
RSI_DISABLE  
#define RSI_TCP_IP_FEATURE_BIT_MAP  
TCP_IP_FEAT_DHCPV4_CLIENT  
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP  
EXT_FEAT_256K_MODE  
#define RSI_BAND  
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

```
#define RSI_BLE_PWR_INX  
#define RSI_BLE_PWR_SAVE_OPTIONS  
#define RSI_DUTY_CYCLING
```

8

0

0

**Note:**

**rsi\_wlan\_config.h** and **rsi\_ble\_config.h** files are already set with the desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. After the program gets executed, Redpine module will be in Advertising state.
2. Open an LE App in the Smartphone and do the scan.
3. In the App, Redpine module device will appear with the name configured in the macro **RSI\_BLE\_LOCAL\_NAME** (Ex: "WYZBEE\_PERIPHERAL") or sometimes observed as the Redpine device as the internal name "SimpleBLEPeripheral".
4. Initiate connection from the mobile App.
5. Observe that the connection is established between Smartphone and Redpine module.

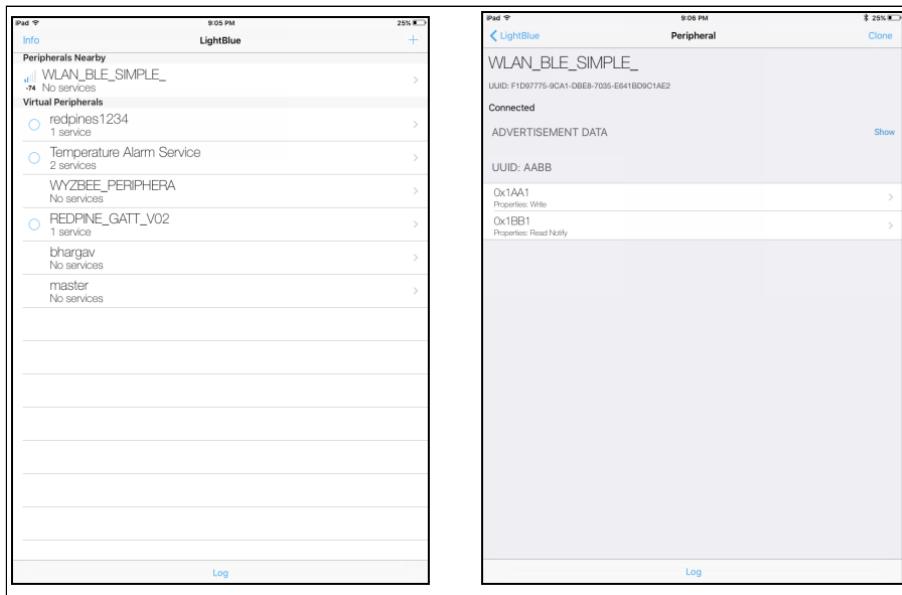


Figure 2 : Scanning for BLE devices and connecting to WLAN\_BLE\_SIMPLE device

## 3.4 Simple Chat Example

### 3.4.1 Application Overview

This application demonstrates how to configure GATT server in BLE peripheral mode and explains how to do read&write operations with GATT server from connected remote device using GATT client.

In this Application, GATT server configures with Custom service with write and readable characteristic UUIDs. When connected remote device writes data to writable characteristic UUID, Redpine device receives the data which is received on writable characteristic UUID and writes the same data to readable characteristic UUID and sends notifications to the connected device (or) remote device can read the same data using read characteristic UUID.

### Sequence of Events

This Application explains user how to:

- Create Simple chat service
- Make the device to advertise
- Connect from remote BTLE device
- Receive the message from the connected peer/Smartphone
- Loop back the received message

### 3.4.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE central device

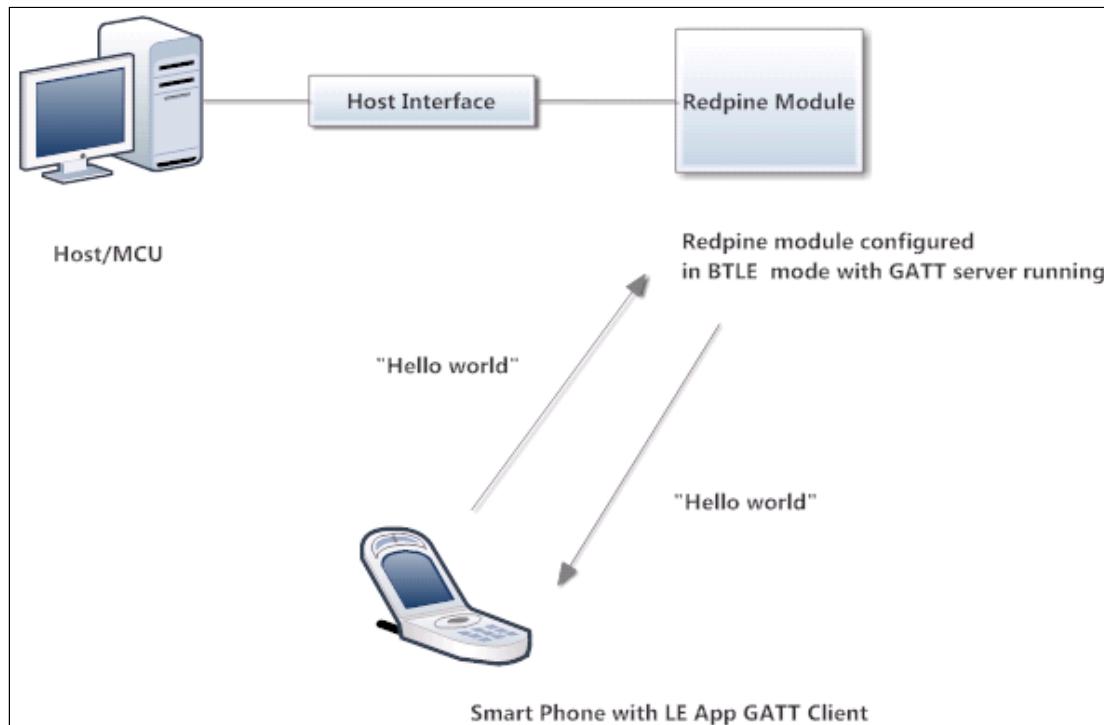
**Note**

Install Light blue App for tablet for ipad mini and BLE scanner app for android smart phone.  
user can download the BLE scanner App from the following link

<https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner&hl=en>

user can download the Light blue App from the following link

<https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer&hl=en>



**Figure 1: Setup Diagram**

### 3.4.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_simple\_chat.c** file and update/modify following macros,  
**RSI\_BLE\_NEW\_SERVICE\_UUID** refers to the attribute value of the newly created service.  
**RSI\_BLE\_ATTRIBUTE\_1\_UUID** refers to the attribute type of the first attribute under this service (**RSI\_BLE\_NEW\_SERVICE\_UUID**).

```
#define RSI_BLE_NEW_SERVICE_UUID  
0xAABB  
#define RSI_BLE_ATTRIBUTE_1_UUID  
0x1AA1
```

**RSI\_BLE\_MAX\_DATA\_LEN** refers to the Maximum length of the attribute data.

```
#define RSI_BLE_MAX_DATA_LEN  
20
```

**RSI\_BLE\_APP\_SIMPLE\_CHAT** refers name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_APP_SIMPLE_CHAT  
"BLE_SIMPLE_CHAT"
```

Following are the **non-configurable** macros in the application.

**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.  
**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

```
#define RSI_BLE_CHAR_SERV_UUID  
0x2803  
#define RSI_BLE_CLIENT_CHAR_UUID  
0x2902
```

Following are the properties

**RSI\_BLE\_ATT\_PROPERTY\_READ** is used to set the READ property to an attribute value.  
**RSI\_BLE\_ATT\_PROPERTY\_WRITE** is used to set the WRITE property to an attribute value.  
**RSI\_BLE\_ATT\_PROPERTY\_NOTIFY** is used to set the NOTIFY property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_READ  
0x02  
#define RSI_BLE_ATT_PROPERTY_WRITE  
0x08  
#define RSI_BLE_ATT_PROPERTY_NOTIFY  
0x10
```

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN  
10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE  
RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP  
FEAT_SECURITY_OPEN  
#define RSI_TCP_IP_BYPASS  
RSI_DISABLE  
#define RSI_TCP_IP_FEATURE_BIT_MAP  
TCP_IP_FEAT_DHCPV4_CLIENT  
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP  
EXT_FEAT_256K_MODE  
#define RSI_BAND  
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

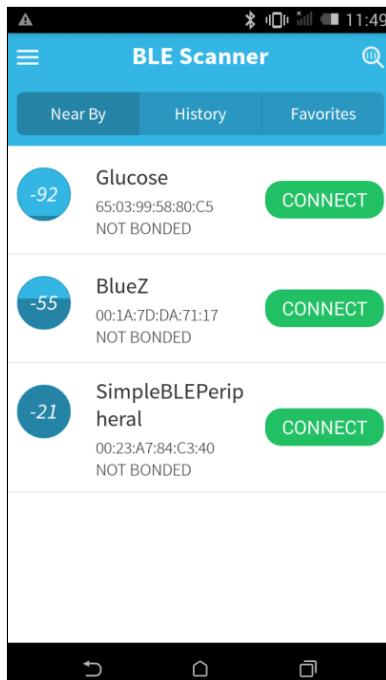
```
#define RSI_BLE_PWR_INX  
8  
#define RSI_BLE_PWR_SAVE_OPTIONS  
0  
#define RSI_DUTY_CYCLING  
0
```

**Note:**

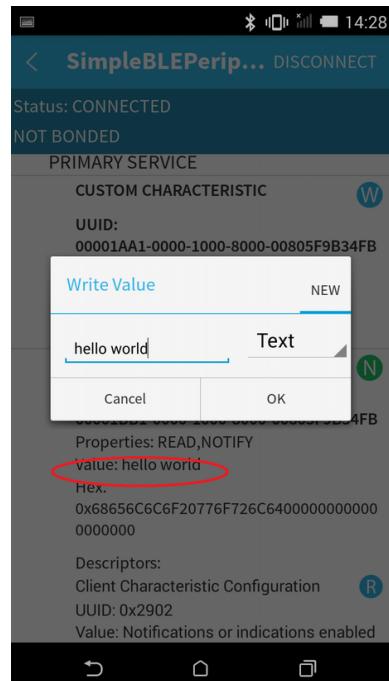
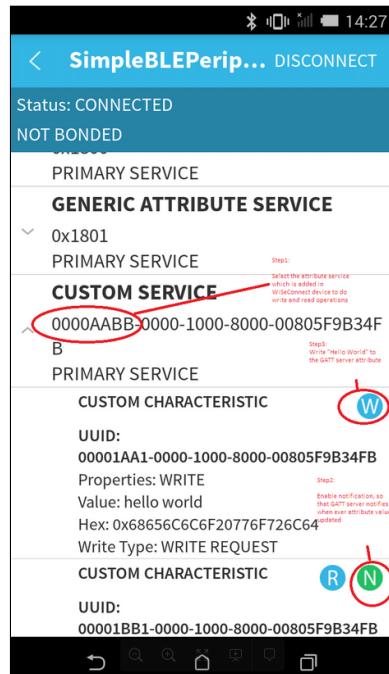
rsi\_wlan\_config.h and rsi\_ble\_config.h files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. After the program gets executed, Redpine module will be in Advertising state.
2. Open a LE App in the Smartphone and do the scan.
3. In the App, Redpine module device will appear with the name configured in the macro **RSI\_BLE\_APP\_SIMPLE\_CHAT** (Ex: "BLE\_SIMPLE\_CHAT") or sometimes observed as Redpine device as internal name "**SimpleBLEPeripheral**".



4. Initiate connection from the App.
5. After successful connection, LE scanner displays the supported services of Redpine module.
6. Select the attribute service which is added **RSI\_BLE\_NEW\_SERVICE\_UUID**  
**(Ex: 0xAABB)**
7. After selecting the service do **Write and Read** operations with GATT server.
8. Enable notifications for the read attribute **RSI\_BLE\_ATTRIBUTE\_2\_UUID**  
**(Example: 0x1BB1) So that GATT server notifies when value updated in that particular attribute.**
9. Write data (Ex: "Hello World") to attribute **RSI\_BLE\_ATTRIBUTE\_1\_UUID** (Ex: 0x1AA1). So that GATT server notifies when value updated in that particular attribute.
10. Redpine module receives the data sent by remote device and same data writes into the attribute **RSI\_BLE\_ATTRIBUTE\_2\_UUID** (Ex: 0x1BB1) and will notifies the GATT client (remote device).
11. Please refer the given below images for write and read operations from remote device GATT client.



### 3.5 SMP Example

#### 3.5.1 Application Overview

This application demonstrates how to configure the Redpine device in Central mode and connects with remote slave device and how to enable SMP(Security Manager Protocol) pairing.  
 In this application, Redpine module connects with peripheral or central device and initiates SMP pairing process in

case of master role. After successful SMP pairing, SMP encryption will be enabled in both Central and Peripheral device.

**Note :** This application is applicable for device which supports Bluetooth 4.0 and 4.1. For devices which supports Bluetooth 4.2 and above version can run LE Secure Connections application to test LE pairing.

## Sequence of Events

With slave role, This Application explains user how to:

- Configure device in peripheral/Central mode based on user configuration
- Connect with remote BTLE central/ peripheral device.
- Initiate SMP paring from central device
- Initiate SMP pair response.
- Send SMP passkey for the received SMP passkey request.
- Encryption is enabled.

### 3.5.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE Peripheral device in case of redpine module as Master(BLE central)
- BTLE Central device in case of redpine module as slave(BLE peripheral)

#### Note

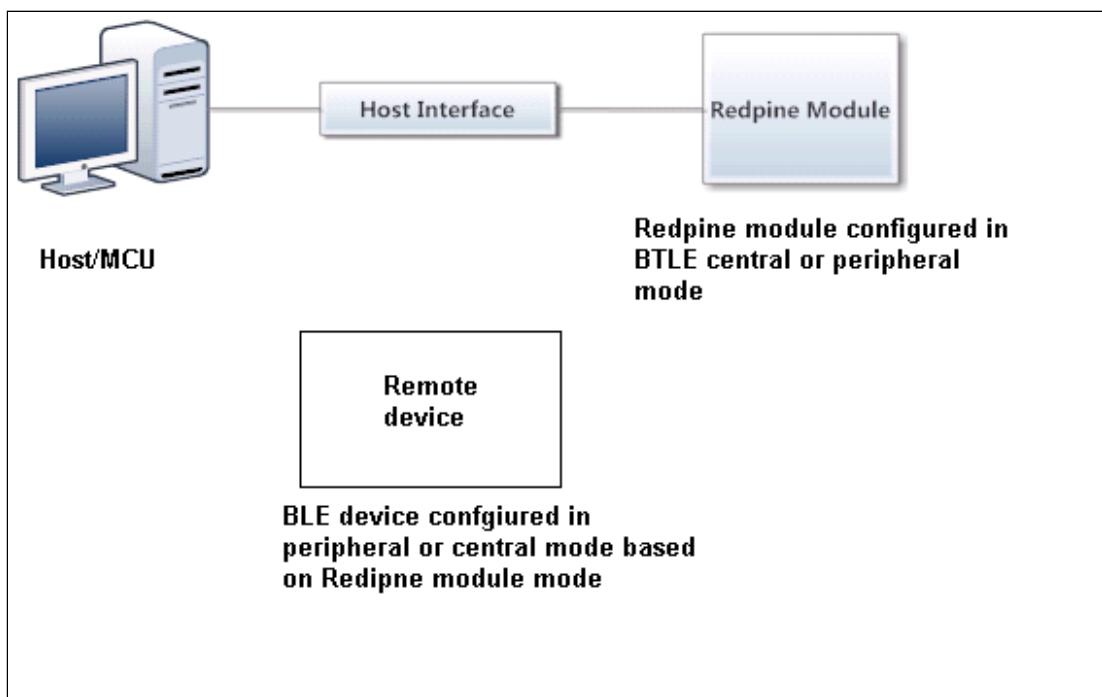
Install Light blue App for tablet for ipad mini and BLE scanner app for android smart phone.

user can download the BLE scanner App from the following link

<https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner&hl=en>

user can download the Light blue App from the following link

<https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer&hl=en>



**Figure 1: Setup Diagram**

### 3.5.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_smp.c** file and update/modify following macros,  
**ROLE** refers the role of Redpine module

#define MASTER	1
#define SLAVE	0
#define ROLE	SLAVE

**RSI\_BLE\_DEV\_ADDR\_TYPE** refers address type of the remote device to connect.

#define RSI_BLE_DEV_ADDR_TYPE	
LE_PUBLIC_ADDRESS	

**Note**

Depends on the remote device, address type will be changed.

Valid configurations are  
LE\_RANDOM\_ADDRESS  
LE\_PUBLIC\_ADDRESS

**RSI\_BLE\_DEV\_ADDR** refers address of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR "00:1A:  
7D:DA:71:13"
```

**RSI\_REMOTE\_DEVICE\_NAME** refers the name of remote device to which Redpine device has to connect

```
#define RSI_REMOTE_DEVICE_NAME  
"BLE_PERIPHERAL"
```

**Note**

Redpine module can connect to remote device by referring either RSI\_BLE\_DEV\_ADDR or RSI\_REMOTE\_DEVICE\_NAME of the remote device.

**RSI\_BLE\_SMP\_IO\_CAPABILITY** refers IO capability.

**RSI\_BLE\_SMP\_PASSKEY** refers address type of the remote device to connect.

```
#define RSI_BLE_SMP_IO_CAPABILITY 0x03  
#define RSI_BLE_SMP_PASSKEY 0
```

Following are the non-configurable macros in the application.

#define RSI_BLE_CONN_EVENT	0x01
#define RSI_BLE_DISCONNECT_EVENT	0x02
#define RSI_BLE_SMP_REQ_EVENT	0x03
#define RSI_BLE_SMP_RESP_EVENT	0x04
#define RSI_BLE_SMP_PASSKEY_EVENT	0x05
#define RSI_BLE_SMP_FAILED_EVENT	0x06
#define RSI_BLE_ENCRYPT_STARTED_EVENT	0x07

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN 10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

#define RSI_BLE_PWR_INX	8
#define RSI_BLE_PWR_SAVE_OPTIONS	0
#define RSI_DUTY_CYCLING	0

**Note**

**rsi\_wlan\_config.h** and **rsi\_ble\_config.h** files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

Following the execution process in case of Redpine device as master

1. If user select the **MASTER** role, After the program gets executed, Redpine device will be trying to connect to remote BT device.advertise the third party device.  
If user select the **SLAVE** role, After the program gets executed, Redpine device will be advertising.then connect from remote device.
2. After successful connection, flow of commands are as below:  
**Master** device will initiate SMP pairing  
**Slave** device give SMP response.  
Both devices will exchange SMP passkey as zero
3. If SMP succeed, host receives SMP encrypt enabled event. If not success, Device sends SMP failure event to host.
4. In encryption enabled event LocalEDIV, Local Rand, LocalLTK parameters will be indicated.
5. Again after disconnection, if Master want to connect, master ask for LE LTK Request event to slave by giving LocalEDIV and LocalRand, and if same, this example give LocalLTK with positive reply using Itk request reply command.

**Note**

We can also send negative reply but remote device may or may not initiate pairing again.  
Currently, in encryption enabled event EDIV, RAND, LTK are of local device so that if master initiate connection he will ask for LTK request by giving slave's (in this example) EDIV and RAND.

## 3.6 Simple Peripheral Power Save Example

### 3.6.1 Application Overview

This application demonstrates that how to configure the device in power save in Advertising mode and in connected mode in simple BLE peripheral mode.

#### Sequence of Events

This Application explains user how to:

- Set a local name for the device
- Configure the module in power save mode
- Configure the device to advertise
- Connect from the remote Master device
- Analyze power save functionality when the WiSeConnect device in Advertise mode and in the connected state using an Azilent power analyzer

### 3.6.2 Application Setup

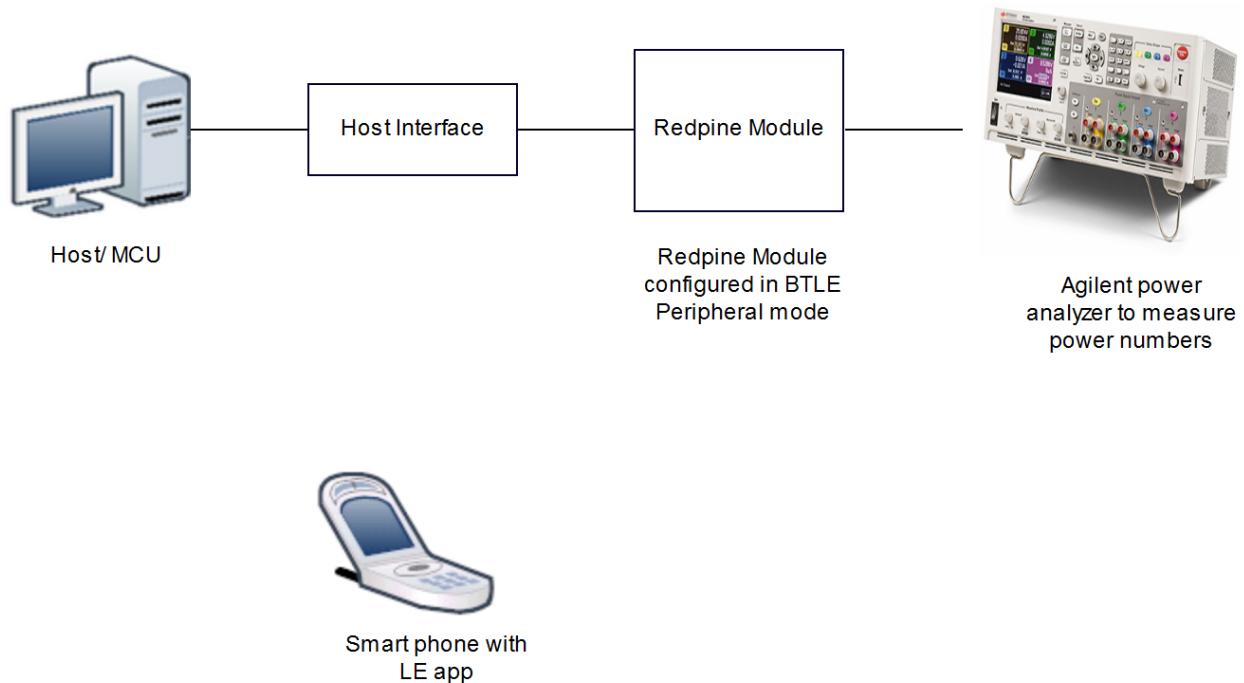
The WiSeConnect in its many variants supports SPI and UART interfaces. Depending on the interface used, the required set up is as below:

#### SPI based Setup Requirements

- Windows PC with KEIL or IAR IDE
- Redpine module
- Smartphone (Android)/tablet with LE application.  
**Example:** Install Light blue App on the tablet for iPad mini and BLE scanner app for the Android smartphone.
- Agilent power analyzer

#### UART/USB-CDC based Setup Requirements

- Windows PC with Dev-C++ IDE
- WiSeConnect device
- Smartphone (Android)/tablet with LE application.  
**Example:** Install Light blue App on the tablet for iPad mini and BLE scanner app for the Android smartphone.
- Agilent power analyzer



**Figure 1: Setup Diagram**

### 3.6.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open rsi\_ble\_peripheral.c file and update/modify following macros  
**RSI\_BLE\_LOCAL\_ANME** refers the name of the WiSeConnect device to appear during scanning by remote devices.

```
#define RSI_BLE_LOCAL_NAME  
"WLAN_BLE_SIMPLE"
```

**RSI\_SEL\_ANTENNA** refers antenna to be used by WiSeConnect module.  
If the user using internal antenna then set,

```
#define RSI_SEL_ANTENNA  
RSI_SEL_INTERNAL_ANTENNA
```

If the user using an external antenna (U.FL connector) then set,

```
#define RSI_SEL_ANTENNA  
RSI_SEL_EXTERNAL_ANTENNA
```

#### To Enable Power Save

**PSP\_MODE** refers power save profile mode. The wiseconnect device supports following power modes in BTLE,

**RSI\_ACTIVE (0):** In this mode, the module is active and power save is disabled.

**RSI\_SLEEP\_MODE\_2 (1):** This mode is applicable when the module is in Advertising state as well as in connected state. In this sleep mode, SoC will go to sleep based on GPIO handshake or Message exchange, therefore handshake is required before sending data to the module.

**RSI\_SLEEP\_MODE\_8 (8):** In this power mode, the module goes to power save when it is in the unassociated state with the remote device. In this sleep mode, SoC will go to sleep based on GPIO handshake or Message exchange, therefore handshake is required before sending the command to the module.

```
#define PSP_MODE  
RSI_SLEEP_MODE_2
```

#### Note1:

For **RSI\_SLEEP\_MODE\_2** and **RSI\_SLEEP\_MODE\_8** modes, GPIO or Message based handshake can be selected using **RSI\_HAND\_SHAKE\_TYPE** macro which is defined in **rsi\_wlan\_config.h**

#### Note2:

In this example, user can verify RSI\_SLEEP\_MODE\_2 with Message based handshake. If the user wants to verify other power modes, the user has to change the application as well as GPIO handshake signals

**PSP\_TYPE** refers power save profile type. The wiseconnect device supports following power save profile types in BTLE mode,

**RSI\_MAX\_PSP (0):** In this mode, the WiSeConnect device will be in Maximum power save mode. i.e Device will wake up for every DTIM beacon and do data Tx and Rx.

```
#define PSP_TYPE RSI_MAX_PSP
```

#### Following are the non-configurable macros in the application.

Following are the event numbers for advertising, connection and Disconnection events:

#define RSI_APP_EVENT_CONNECTED	1
#define RSI_APP_EVENT_DISCONNECTED	2

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN 10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_CUSTOM_FEATURE_BIT_MAP 0
#define RSI_BAND
RSI_BAND_2P4GHZ
```

**RSI\_HAND\_SHAKE\_TYPE** is used to select GPIO or Message based handshake in **RSI\_SLEEP\_MODE\_2** and **RSI\_SLEEP\_MODE\_8** modes.

```
#define RSI_HAND_SHAKE_TYPE MSG_BASED
```

**RSI\_SELECT\_LP\_OR\_ULP\_MODE** is used to select low power mode or ultra low power mode. Valid configurations are, **RSI\_LP\_MODE** or **RSI\_ULP\_WITH\_RAM\_RET** or **RSI\_ULP\_WITHOUT\_RAM\_RET**

**RSI\_LP\_MODE :**

In this, the module will be in Low power mode.

**RSI\_ULP\_WITH\_RAM\_RET :**

In this, the module will be in Ultra low power mode and it will remember the previous state after issuing power save mode command.

**RSI\_ULP\_WITHOUT\_RAM\_RET :**

In this, the module will be in Ultra low power mode and it will not remember the previous state after issuing power save mode command. After wakeup, the module will give CARD READY indication and the user has to issue commands from wireless initialization.

```
#define RSI_SELECT_LP_OR_ULP_MODE
RSI_ULP_WITH_RAM_RET
```

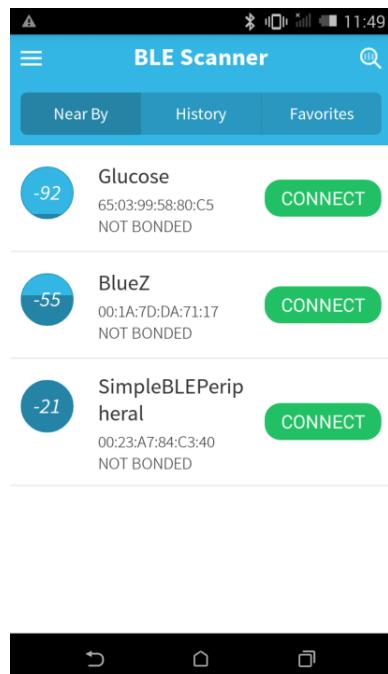
## Executing the Application

1. After the program gets executed, the WiSeConnect module would be in Advertising state with configured power save the profile.
2. The wiseconnect device will go to sleep and wakes up for every advertising interval and goes back to sleep after advertising. Please refer the given below image for power save cycle in advertising mode.



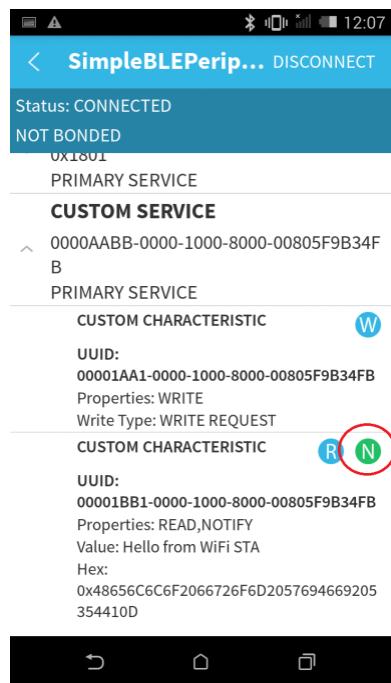
Figure 2: Power profile in advertising mode

3. Open an LE App in the Smartphone and do Scan.
4. In the App, WiSeConnect module device will appear with the name configured in the macro **RSI\_BLE\_LOCAL\_NAME** (Ex: "WLAN\_BLE\_SIMPLE") or sometimes observed as the WiSeConenct device as the internal name "SimpleBLEPeripheral".



5. Initiate connection from the mobile App.

6. After successful connection, User can see the connected state in BLE scanner app and also check the supported services by the WiSeConnect device.



7. After successful connection, Module goes to sleep and wakes up for every connection interval. Please check the below image for power save cycle after connection.



Figure 3: Power profile in the connected state

**Note**

Default configuration of connection interval of Master device (smartphone) is 18ms. So, the WiSeConnect device will wake up for every 18ms sec and goes back to sleep after advertise.

Above power save profile image capture when it is in the idle state after successful connection. So, the user may not get same profile as shown above image. It will vary based on the traffic.

## 3.7 BLE Immediate Alert Client Example

### 3.7.1 Application Overview

This application demonstrates how a GATT client device accesses a GATT server device.

Redpine module acts as a GATT client device in Central mode. Smartphone acts as a GATT server device in peripheral mode which is having immediate alert service. Set up Redpine module to act as GATT client by running the GATT client application. After connecting with the GATT Server device, actual immediate alert notification functionality will be active. In this application, when a peer device moves beyond certain range (can be configured by using the RSSI threshold) of RSSI values, it gives an alert notification to the user and is demonstrated by glowing LED with Yellow (Moving far) and Green (Moving near) colors.

### Sequence of Events

This Application explains user how to:

- Connect to BTLE device
- Bring profiles and services from the remote device
- Get rssi value every time

### 3.7.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE peripheral device

### 3.7.3 Details of the Application

The application (running in Redpine module) includes following steps.

1. Make Redpine module to act as GATT client device.
2. Connect the Redpine module with the remote device.

### 3.7.4 Configuration and Execution of the Application

### 3.7.5 Configuring the Application

1. Open **rsi\_ble\_immediate\_alert.c** file and update/modify following macros,

**RSI\_BLE\_DEV\_ADDR\_TYPE** refers address type of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR_TYPE  
LE_PUBLIC_ADDRESS
```

Valid configurations are  
LE\_RANDOM\_ADDRESS  
LE\_PUBLIC\_ADDRESS

**Note:**

Depends on the remote device, address type will be changed.

**RSI\_BLE\_DEV\_ADDR** refers address of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR  
"00:1A:7D:DA:71:13"
```

**RSI\_REMOTE\_DEVICE\_NAME** refers the name of remote device to which Redpine device has to connect

```
#define RSI_REMOTE_DEVICE_NAME  
"BLE_PERIPHERAL"
```

**Note:**

Redpine module can connect to remote device by referring either **RSI\_BLE\_DEV\_ADDR** or **RSI\_REMOTE\_DEVICE\_NAME** of the remote device.

**RSI\_BLE\_RSSI\_THRESHOLD\_VALUE**- To set the RSSI threshold for immediate alert service.

```
#define RSI_BLE_RSSI_THRESHOLD_VALUE
```

40

**RSI\_BLE\_NEW\_SERVICE\_UUID** refers to the attribute value of the newly created service.

**RSI\_BLE\_ATTRIBUTE\_1\_UUID** refers to the attribute type of the first attribute under this service (**RSI\_BLE\_NEW\_SERVICE\_UUID**).

```
#define RSI_BLE_NEW_SERVICE_UUID  
#define RSI_BLE_ATTRIBUTE_1_UUID
```

0x1802

0x2A06

**RSI\_BLE\_MAX\_DATA\_LEN** refers to the Maximum length of the attribute data.

```
#define RSI_BLE_MAX_DATA_LEN
```

20

**RSI\_BLE\_APP\_SIMPLE\_CHAT** refers name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_APP_NAME  
"IMMEDIATE_ALERT"
```

Following are the **non-configurable** macros in the application.

**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.  
**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

```
#define RSI_BLE_CHAR_SERV_UUID 0x2803  
#define RSI_BLE_CLIENT_CHAR_UUID 0x2902
```

**RSI\_BLE\_ATT\_PROPERTY\_READ** is used to set the READ property to an attribute value.  
**RSI\_BLE\_ATT\_PROPERTY\_WRITE** is used to set the WRITE property to an attribute value.  
**RSI\_BLE\_ATT\_PROPERTY\_NOTIFY** is used to set the NOTIFY property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_READ 0x02  
#define RSI_BLE_ATT_PROPERTY_WRITE 0x08  
#define RSI_BLE_ATT_PROPERTY_NOTIFY 0x10
```

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN 10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE  
RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP  
FEAT_SECURITY_OPEN  
#define RSI_TCP_IP_BYPASS  
RSI_DISABLE  
#define RSI_TCP_IP_FEATURE_BIT_MAP  
TCP_IP_FEAT_DHCPV4_CLIENT  
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP  
EXT_FEAT_256K_MODE  
#define RSI_BAND  
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

```
#define RSI_BLE_PWR_INX  
8  
#define RSI_BLE_PWR_SAVE_OPTIONS  
0  
#define RSI_DUTY_CYCLING 0
```

**Note**

**rsi\_wlan\_config.h** and **rsi\_ble\_config.h** files are already set with desired configuration in respective example folders user need not change for each example.

### 3.7.6 Executing the Application

1. After the program gets executed, Redpine module would be trying to connect to remote device
2. Advertise GATT server LE device, which has the support for immediate alert service.
3. Redpine module can connect to it.
4. After connecting, application use immediate alert service in server and depending on rssi value, a notification will be sent to user.

## 3.8 iBeacon Example

### 3.8.1 Application Overview

#### Overview

This application demonstrates how to set the iBeacon data format in advertising parameters in simple BLE peripheral mode.

#### Sequence of Events

This Application explains user how to:

- Set a local name to the device
- Set the iBeacon data to be advertised in Redpine module
- Configure the device in Advertise mode
- Scan from remote Master device

### 3.8.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Smart phone with ibeacon detector application

## iBeacon Advertise data format

iBeacon prefix 9Bytes	UUID 16bytes	Major Number 2Bytes	Minor Number 2 Bytes	Tx Power 1bytes
Adv Flags 3 Bytes	Adv Header 2 Bytes	Company ID 2 Bytes	iBeacon Type 1Byte	iBeacon Length 1Bytes

### **iBeacon Prefix:**

Vendor specific fixed value.

Default iBeacon prefix values setting by application is,

Prefix = {0x02, 0x01, 0x02, 0x1A, 0xFF, 0x4C, 0x00, 0x02, 0x15}

### **UUID:**

User generated proximity UUID.

Remote devices recognize which beacon they approach on the basis of UUID, major and minor numbers.

Default UUID, Major and Minor values setting by application is,

UUID = {0xFB, 0x0B, 0x57, 0xA2, 0x82, 0x28, 0x44, 0xCD, 0x91,

0x3A, 0x94, 0xA1, 0x22, 0xBA, 0x12, 0x06}

major\_num = {0x11, 0x22}

minor\_num = {0x33, 0x44}

**Tx power** is used to calculate distance from iBeacon.

Default Tx power value setting by application is,

Tx Power = 0x33

**Note:** If the user wants to change the prefix, UUID, Major number, Minor number and Tx Power values, Please change the following values in **rsi\_ble\_ibeacon.c** file.

### **For Prefix:**

```
<span style="color: #005032">uint8_t</span> adv[31] = {0x02, 0x01, 0x02, 0x1A, 0xFF, 0x4C, 0x00, 0x02, 0x15}; //  
prefix(9bytes)
```

### **For UUID:**

```
uint8_t uuid[16] = {0xFB, 0x0B, 0x57, 0xA2, 0x82, 0x28, 0x44, 0xCD, 0x91, 0x3A, 0x94, 0xA1, 0x22, 0xBA, 0x12,  
0x06};  
For Major Number:  
uint8_t major_num[2] = {0x11, 0x22};  
For Major Number:  
uint8_t minor_num[2] = {0x33, 0x44};  
For Tx Power:  
uint8_t tx_power = 0x33;
```

```

-- 86 * This function is used to test the BLE peripheral role and simple GAP API's.
87 */
88 int32_t rsi_ble_ibeacon(void)
89 {
90     int32_t status = 0;
91     int32_t temp_event_map = 0;
92     uint8_t remote_dev_addr[18] = {0};
93     uint8_t adv[31] = {0x02, 0x01, 0x02, 0x1A, 0xFF, 0x4C, 0x00, 0x02, 0x15}; //prefix(9
94     uint8_t uuid[16] = {0xFB, 0x0B, 0x57, 0xA2, 0x82, 0x28, 0x44, 0xCD, 0x91, 0x
95     uint8_t major_num[2] = {0x11, 0x22};
96     uint8_t minor_num[2] = {0x33, 0x44};
97     uint8_t tx_power = 0x33;
98

```

### 3.8.3 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

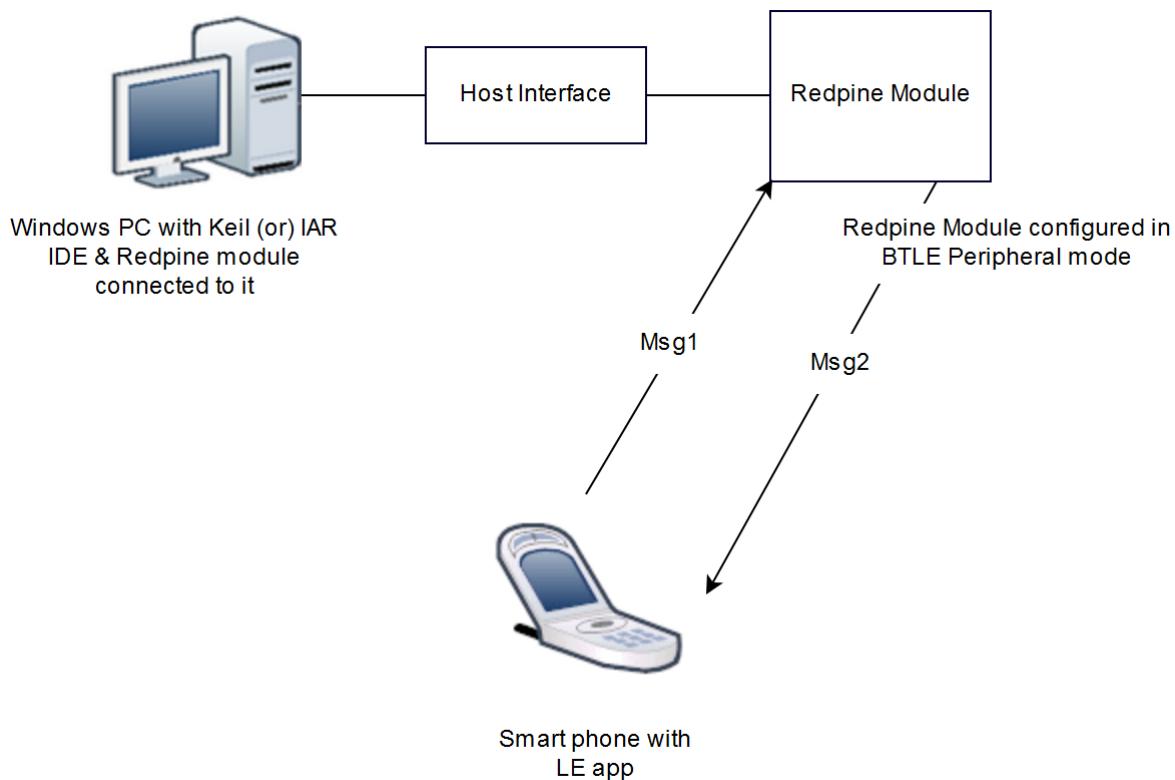
#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Smart phone with LE application

**Note:**

Install iBeaconDetector app for android smart phone.

<https://play.google.com/store/apps/details?id=youten.redo.ble.ibeacondetector>



**Figure 1: Setup diagram**

### 3.8.4 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_ibeacon.c** file and update/modify following macros:  
**RSI\_BLE\_LOCAL\_ANME** refers name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_LOCAL_NAME           "ibeacon"
```

Following are the event numbers for connection and Disconnection events,

#define RSI_APP_EVENT_CONNECTED	1
#define RSI_APP_EVENT_DISCONNECTED	2

Following are the **non-configurable** macros in the application.  
**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN
```

10000

2. Open ***rsi\_wlan\_config.h*** file and update/modify following macros,

```
#define CONCURRENT_MODE
#define RSI_FEATURE_BIT_MAP
#define RSI_TCP_IP_BYPASS
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
#define RSI_BAND
```

RSI\_DISABLE  
FEAT\_SECURITY\_OPEN  
RSI\_DISABLE  
EXT\_FEAT\_256K\_MODE  
RSI\_BAND\_2P4GHZ

3. Open ***rsi\_ble\_config.h*** file and update/modify following macros,

```
#define RSI_BLE_PWR_INX
#define RSI_BLE_PWR_SAVE_OPTIONS
#define RSI_DUTY_CYCLING
```

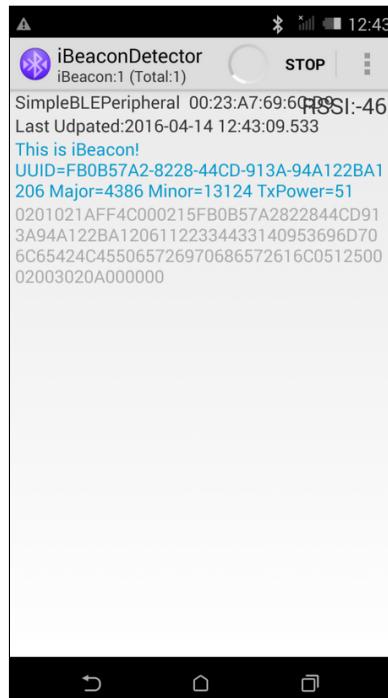
8  
0  
0

**Note:**

*rsi\_wlan\_config.h* and *rsi\_ble\_config.h* files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. After the program gets executed, Redpine module would be in Advertising state.
2. Open iBeaconDetector app in the Smartphone and do Scan.
3. In the App, Redpine module device would appear with the name configured in the macro **RSI\_BLE\_LOCAL\_NAME (Ex: "ibeacon")** or sometimes observed as "**SimpleBLEPeripheral**".
4. After successful scan, User can see the Redpine device advertised data i.e UUID, Maximum Number, Minimum Number and Tx Power in iBeaconDetector application.



### 3.9 Privacy Example

#### 3.9.1 Feature Overview

Bluetooth LE supports a feature that reduces the ability to track an LE device over a period of time by changing the Bluetooth device address on a frequent basis, called Privacy of that particular device.

The device address of the remote device referred to as the private address will be resolved by local device in order to connect to that device. The private address is generated by using Identity Resolving Key (IRK) exchange in between devices during SMP bonding procedure. Our local device will add the remote devices in one Resolving list(to maintain remote device identity addresses) along with that IRK's and enable the Resolution, sets privacy mode and connect to the remote device with remote identity address.

#### 3.9.2 Application Overview

This application demonstrates how to configure device with privacy feature by organizing resolving list and resolution process and how to connect to remote Peripheral device.

#### Sequence of Events

This Application explains user how to:

- Set a local name to the device
- Scan devices
- Connection to remote device
- SMP level connection
- Exchange of IRK's and store them
- Disconnect with remote device
- Add remote device to resolve list with identity address
- Get resolve list size
- Set resolution enable and time out
- Set privacy mode

- Connect remote device with identity address
- Start Encryption instead of SMP repairing

**Note**

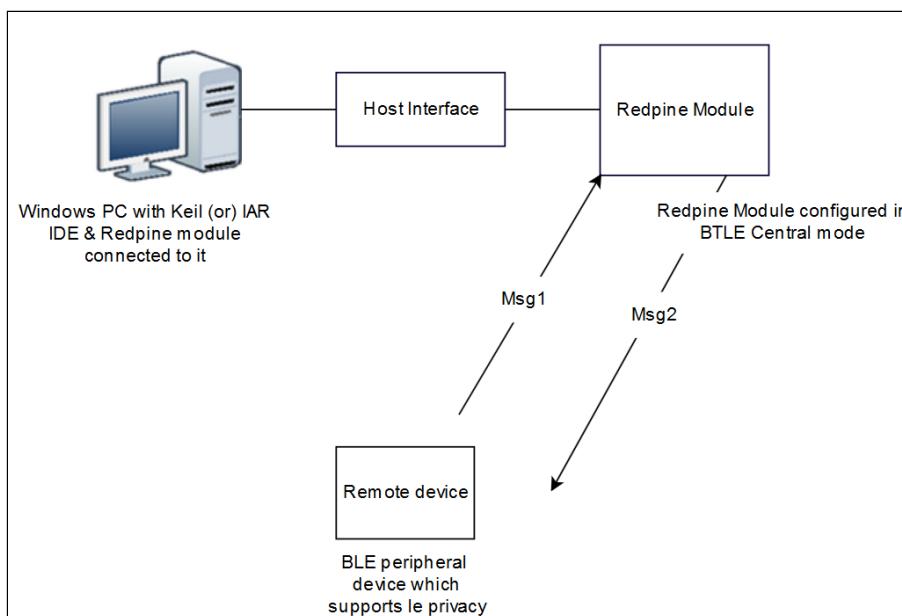
If both devices having resolution enable, enhanced connection event will come for any privacy mode. if remote device is without resolution, privacy mode should be device privacy mode.

### 3.9.3 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE peripheral device which supports privacy feature(Generally phones with the nRF Connect application)



**Figure 1: Setup Diagram**

### 3.9.4 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_le\_privacy.c** file and update/modify following macros:  
**RSI\_BLE\_LOCAL\_NAME** refers the name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_LOCAL_NAME  
"BLE_PRIVACY"
```

RSI\_REMOTE\_DEVICE\_NAME refers the name of the Remote device to which Redpine module initiate connection.

```
#define RSI_REMOTE_DEVICE_NAME  
"REMOTE_DEV"
```

RSI\_DEVICE\_ROLE refers the role of the Redpine device.

```
#define RSI_DEVICE_ROLE  
"RSI_MASTER"
```

**Note**

RSI\_DEVICE\_ROLE should be RSI\_MASTER

RSI\_DEVICE\_ROLE refers the role of the Redpine device.

```
#define RSI_DEVICE_ROLE  
"RSI_MASTER"
```

RSI\_DEVICE\_ROLE refers the role of the Redpine device.

```
#define RSI_DEVICE_ROLE  
"RSI_MASTER"
```

RSI\_BLE\_SMP\_IO\_CAPABILITY refers the IO capability of redpine device for SMP, RSI\_BLE\_SMP\_PASSKEY is smp passkey key from redpine device

#define RSI_BLE_SMP_IO_CAPABILITY	0x01
#define RSI_BLE_SMP_PASSKEY	0

Following are the **non-configurable** macros in the application.

Following are the event numbers for connection, Disconnection, and enhanced connection events.

#define RSI_APP_EVENT_ADV_REPORT	0x00
#define RSI_BLE_CONN_EVENT	0x01
#define RSI_BLE_DISCONNECT_EVENT	0x02
#define RSI_BLE_SMP_REQ_EVENT	0x03
#define RSI_BLE_SMP_RESP_EVENT	0x04
#define RSI_BLE_SMP_PASSKEY_EVENT	0x05
#define RSI_BLE_SMP_FAILED_EVENT	0x06
#define RSI_BLE_ENCRYPT_STARTED_EVENT	0x07
#define RSI_BLE_SMP_PASSKEY_DISPLAY_EVENT	0x08
#define RSI_BLE_SC_PASSKEY_EVENT	0x09
#define RSI_BLE_LTK_REQ_EVENT	0x0A
#define RSI_BLE_SECURITY_KEYS_EVENT	0x0B
#define RSI_BLE_ENHANCE_CONNECTED_EVENT	0x0C

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN 12000
```

**RSI\_BLE\_DEV\_ADDR\_TYPE** refers the address type of the remote device

```
#define RSI_BLE_REMOTE_ADDR_TYPE  
LE_PUBLIC_ADDRESS
```

**RSI\_BLE\_DEV\_ADDR\_1** refers remote device address which has to connect

```
#define RSI_BLE_REMOTE_ADDR  
"00:15:83:6A:64:17"
```

**RSI\_BLE\_SET\_RESOLVABLE\_PRIV\_ADDR\_TOUT** refers resolution timeout , that is the length of time the Controller uses a Resolvable Private Address before a new resolvable private address is generated and starts being used.

```
#define RSI_BLE_SET_RESOLVABLE_PRIV_ADDR_TOUT 120
```

**RSI\_BLE\_PROCESS\_TYPE** refers the operation to be performed on the resolving list

```
#define RSI_BLE_PROCESS_TYPE  
RSI_BLE_ADD_TO_RESOLVE_LIST
```

valid configurations for the RSI\_BLE\_PROCESS\_TYPE are  
RSI\_BLE\_ADD\_TO\_RESOLVE\_LIST  
RSI\_BLE\_REMOVE\_FROM\_RESOLVE\_LIST  
RSI\_BLE\_CLEAR\_RESOLVE\_LIST

**RSI\_BLE\_PRIVACY\_MODE** refers the privacy mode of local device

```
#define RSI_BLE_PRIVACY_MODE  
RSI_BLE_DEVICE_PRIVACY_MODE
```

**RSI\_BLE\_RESOLVING\_LIST\_SIZE** refers the resolving list size of redpine device

```
#define RSI_BLE_RESOLVING_LIST_SIZE
```

5

2. Open **rsi\_ble\_config.h** file and update/modify following macros  
**RSI\_BLE\_DEV\_ADDR\_RESOLUTION\_ENABLE** refers address resolution is enable or not. It should be 1 to enable privacy feature.

```
#define RSI_BLE_DEV_ADDR_RESOLUTION_ENABLE
```

1

**RSI\_BLE\_ADV\_DIR\_ADDR\_TYPE** refers the address type of remote device which use while advertising.

```
#define RSI_BLE_ADV_DIR_ADDR_TYPE  
LE_PUBLIC_ADDRESS
```

**RSI\_BLE\_ADV\_DIR\_ADDR** refers to which device the local device will advertise with private address, it should be one of the device in resolve list.

```
#define RSI_BLE_ADV_DIR_ADDR  
"00:15:83:6A:64:17"
```

```
#define RSI_BLE_PWR_INX  
#define RSI_BLE_PWR_SAVE_OPTIONS  
#define RSI_DUTY_CYCLING
```

30

0

0

3. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

**Note**

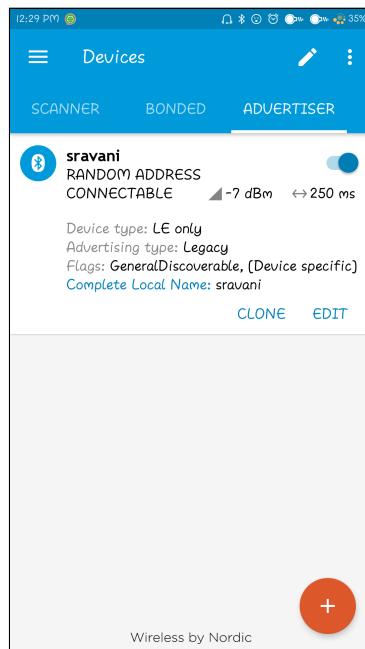
**rsi\_wlan\_config.h** and **rsi\_ble\_config.h** files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

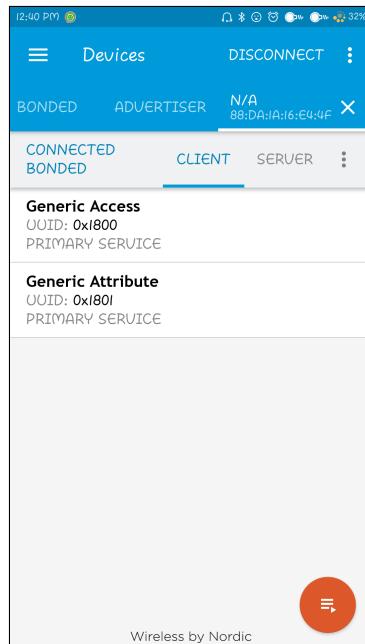
1. After the program gets executed, Redpine module will be in Scanning state.
2. Advertise remote device,
3. If Redpine module get device with name configured RSI\_BLE\_LOCAL\_NAME or bd address with address configured in RSI\_BLE\_REMOTE\_ADDR in results ,local device will try to connect with remote device.
4. After connection Redpine device which is in master mode will initiate SMP request
5. Give response from Remote device and passkey
6. After successful SMP connection security keys will exchanged between Remote device and Redpine device.
7. Redpine device will add remote device's IRK's and local IRK's in to resolve list and enable resolution
8. Give disconnect from remote device and keep in advertise mode.
9. Now Redpine module will try to connect to remote device with identity address.
10. After successful connection, Redpine module will give start encryption instead of SMP repairing.
11. Encryption will be enabled on both sides.

Please find following screen shots for reference.

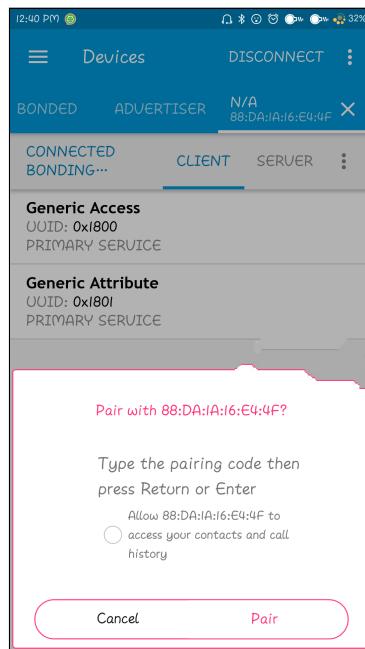
1. Advertise remote device



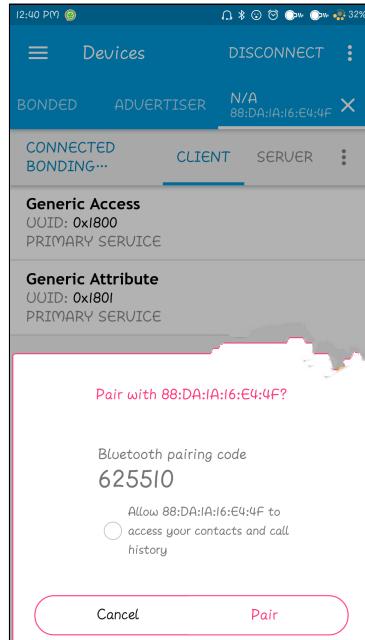
2. Connection in between redpine module and remote device



3. Pairing confirmation



#### 4. Passkey confirmation



### 3.10 Proximity Profile

#### 3.10.1 Application Overview

##### Overview

This application demonstrates how to configure Proximity as GATT server in BLE peripheral mode and explains how to do indicate operation with GATT server from connected remote device using GATT client.

In this Application, Proximity GATT server configures with Proximity service with indicate characteristic UUID. When connected remote device writes data to writable characteristic UUID, WiseConnect device receives the data which is received on writable characteristic UUID and writes the same data to readable characteristic UUID.

## Sequence of Events

This Application explains user how to:

- Create Proximity service
- Make the device to advertise
- Connect from remote BTLE device
- Receive the message from the connected peer/Smartphone
- Give the information by writing alert values to connected device write handle

### 3.10.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE Central device

#### Note

Install Light blue App for tablet for ipad mini and BLE scanner or nRF connect app for android smart phone.  
user can download the BLE scanner App from the following link

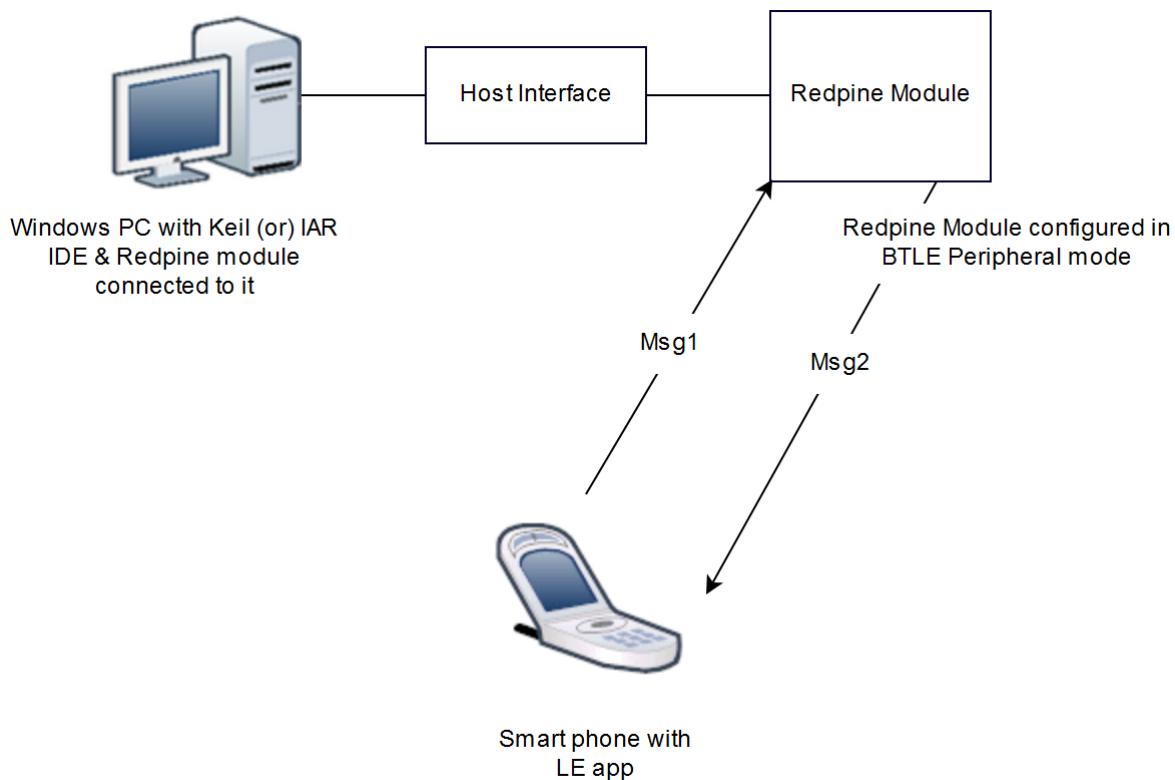
<https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner&hl=en>

user can download the Light blue App from the following link

<https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer&hl=en>

user can download the nRF connect App from the following link

<https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp>



**Figure 1: Setup Diagram**

### 3.10.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_proximity.c** file and update/modify following macros,  
**RSI\_BLE\_DEVICE\_NAME** refers the name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_APP_PROXIMITY_REPORTER  
"BLE_PROXIMITY_REPORTER"
```

Following are the event numbers for advertising, connection and Disconnection events,

#define RSI_BLE_CONN_EVENT	0x01
#define RSI_BLE_DISCONNECT_EVENT	0x02
#define RSI_LINK_LOSS_WRITE_EVENT	0x03
#define RSI_BLE_IMME_ALT_WRITE_EVENT	0x04

Following are the non-configurable macros in the application.

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN 10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

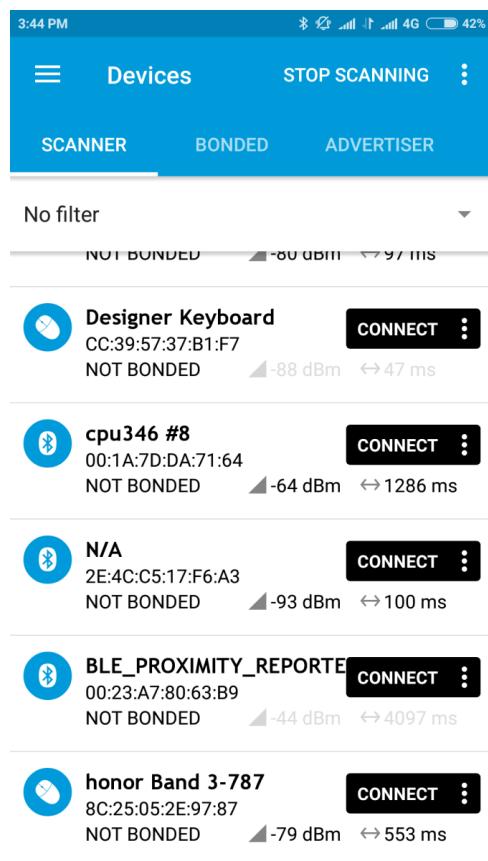
```
#define RSI_BLE_PWR_INX 8
#define RSI_BLE_PWR_SAVE_OPTIONS 0
#define RSI_DUTY_CYCLING 0
```

**Note**

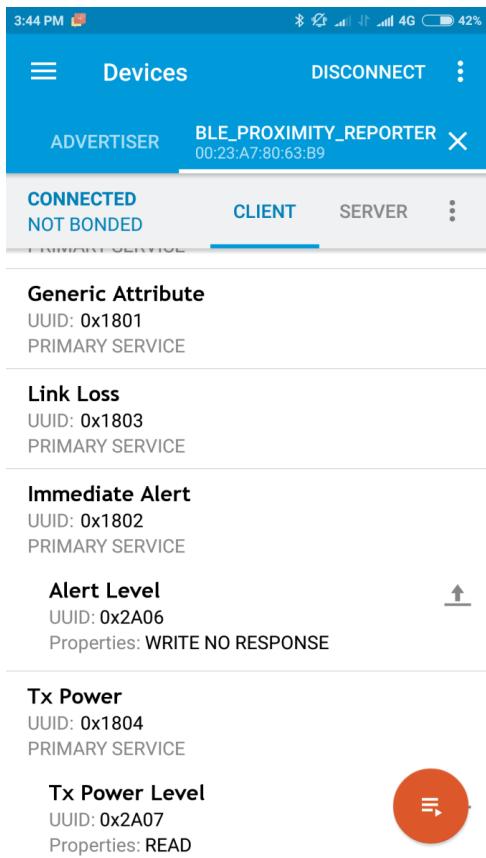
rsi\_wlan\_config.h and rsi\_ble\_config.h files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. After the program gets executed, Redpine module will be in Advertising state.
2. Open a LE SCANNER App in the Smartphone and do the scan.
3. In the App, Redpine module device will appear with the name configured in the macro **RSI\_BLE\_APP\_PROXIMITY\_REPORTER**(Ex:"BLE\_PROXIMITY\_REPORTER").
4. Please refer the given below images for write operation from remote device GATT client and check in our application at write handle.



5. Initiate connection from the App.
6. Please refer the given below images for write operation from remote device GATT client and check in our application at write handle.



### 3.11 Heart Rate Profile Example

#### 3.11.1 Objective

The Heart Rate Service exposes heart rate and other data related to a heart rate sensor intended for fitness applications.

#### GATT Requirements

- Write Characteristic Value: Mandatory if the Heart Rate Control Point characteristic is supported, otherwise excluded for this service.
- Notifications: Mandatory.
- Read Characteristic Descriptors: Mandatory.
- Write Characteristic Descriptors: Mandatory.

#### Service Declaration

The Heart Rate Service shall be instantiated as a «Primary Service». The service UUID shall be set to «Heart Rate Service». The UUID value assigned to «Heart Rate Service».

## Service Characteristics

Characteristic Name	Requirement	Mandatory Property	Optional Property	Security Permissions
Heart Rate Measurement	M	Notify		None
Heart Rate Measurement Client Characteristic Configuration descriptor	M	Read, Write		None
Body Sensor Location	O	Read		None
Heart Rate Control Point	C.1	Write		None

**Table 1 Heart Rate Service**

C.1: Mandatory if the Energy Expended feature is supported, otherwise excluded.  
 Properties not listed as Mandatory or Optional are Excluded.

- **Heart Rate Measurement:** sends the heart rate to the app.
- **Body Sensor Location:** describes where on the body to put the sensor.
- **Heart Rate Control Point:** receives a value from the user when the user wants to reset the Energy Expended measurement.

### Heart Rate Measurement field format

LSB MSB

Flags (1 byte)	Beats per minute (1 or 2 bytes)	Energy Expended (2 bytes)	RR-interval (2 - bytes)
----------------	---------------------------------	---------------------------	-------------------------

The Heart Rate Measurement characteristic is used to send a heart rate measurement ,Including Flags field (for showing the presence of optional fields and features supported), a heart rate measurement value field and , depending .Upon the contents of the Flags field, an Energy Expended field and an RR-Interval field will be selected. Size of this field is 1 byte.

**Flags Field:**  
 MSB LSB

Reserved (3 bits)	RR-interval bit (1 bit)	Energy Expended status bit (1 bit)	Sensor Contact status bit (2 bits)	Heart rate value format bit(1 bit)
-------------------	-------------------------	------------------------------------	------------------------------------	------------------------------------

The Flags field shall be included in the Heart Rate Measurement characteristic .Reserved for Future Use (RFU) bits in the Flags fields shall be set to 0.

### Heart Rate Value Format bit

The Heart Rate Value Format bit (bit 0 of the Flags field) indicates if the data format of the Heart Rate Measurement Value field is in a format of UINT8 or UINT16.When the Heart Rate Value format is sent in a UINT8 format, the Heart Rate Value .Format bit shall be set to 0. When the Heart Rate Value format is sent in a UINT16 format, the Heart Rate Value Format bit shall be set to 1.The value of the Heart Rate Value Format bit may change during a connection.

### Sensor Contact Status bits

The Sensor Contact Status bits (bits 1 and 2 of the Flags field) indicate whether or not, the Sensor Contact feature is supported and if supported whether or not skin contact is detected. The Sensor Contact Status bits (bits 1 and 2 of the Flags field) indicate whether or not, the Sensor Contact feature is supported and if supported whether or not skin contact is detected .If the Sensor Contact feature is supported by the Server, the Sensor Contact Support bit(bit 2 of the Flags field) shall be set to 1. If the Sensor Contact feature is not supported by the Server, the Sensor Contact Support bit shall be set to 0.

### Energy Expended Field

The Energy Expended field represents the accumulated energy expended in kilo Joules since the last time it was reset .If the Server supports energy expended calculations, the Energy Expended field may be included in the Heart Rate Measurement characteristic .If energy expended is used, it is typically only included in the heart Rate Measurement characteristic once every 10 measurements at a regular interval.

### RR-Interval Field

The RR-Interval field may be included in the Heart Rate Measurement characteristic if the device supports RR-Interval measurements .If RR-Interval values are present in the Heart Rate Measurement characteristic, the Server shall set bit 4 of the Flags field (RR-Interval bit) to 1 and shall include one or more RR-Interval values in the Heart Rate Measurement characteristic. Otherwise RR-Interval values shall not be included and bit 4 of the Flags field shall be set to 0.

### Body Sensor Location

The Body Sensor Location characteristic of the device is used to describe the intended location of the heart rate measurement for the device. The value of the Body Sensor Location characteristic is static while in a connection.

### Heart Rate Control Point

The Heart Rate Control Point characteristic is used to enable a Client to write control points to Server to control behavior.

### Process for Heart rate Server and Client

#### Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization. The following sections are applicable to WiSeConnect parts only.

#### Process when acted as server

- Our device will have heart rate service and characteristics and starts advertising.
- After connection with the third party all the supported characteristics will be visible to the third party device.
- Based on the flag field the respective fields get enable and accordingly heart rate measurement field is displayed.
- Once notification is enabled from the client , heart rate measurement field will be updated for every sec . Notifications can be disabled if not required.
- Client can write a value to the control point field , which will raise a gatt write event stating that a value is written to particular handle.

Handle	Attribute Type	Attribute Value
X	0x2800	0x180D
X + 1	0x2803	0x10, X + 2,0xA37
X + 2	0xA37	Heart Rate Measurement value (length is 3 to 7 bytes)
X + 3	0x2902	0x0000
X + 4	0x2803	0x02,X + 5,0xA38
X + 5	0xA38	Sensor location (length is 1 byte)
X + 6	0x2803	0x08,X + 7,0xA39
X + 7	0xA39	Heart rate Control point (length is 1 byte).

**Table 2 GATT Record for heart rate service**

Process when acted as Client

- Our device will start scanning near-by devices and gets connected to the third party device.
- After connection we will get all the supported services from the remote device.
- If the third party device supports Heart rate profile , we start reading respective characteristics that it supports.
- Characteristics with notification property are enabled and we start receiving notifications when ever any write happens to these handles.
- Based on the flag field received from the remote device heart rate measurement characteristic will be displayed.

#### Test-Cases

S.No	Test-Cases	Test Procedure	STATUS
1	Verify the IUT has an instantiation of the Heart Rate Service as a primary service.	<p>1. Discover primary services by service UUID Discover Primary Services by Service UUID, with the service UUID set to «Heart Rate Service».</p> <p>2. Verify one attribute handle range is returned, containing the starting handle and the ending handle of the service definition.</p>	PASS (SERVER – all the services are visible to the connected 3rd party CLIENT-Getting the services from remote device)

2	This test group contains test cases to verify that the characteristic property field of the characteristic declaration meets the requirements of the service. The verification is performed one property at the time.	1. Discover all characteristics of the service by executing Discover All Characteristics of a Service.  2. For a discovered characteristic, verify the characteristic properties field of the characteristic declaration meets the requirements of the service	PASS <small>(SERVER – all the characteristics are visible to the connected 3rd party)</small>  <small>CLIENT-Getting all the characteristics from remote device)</small>
3	Verify that the IUT responds appropriately when a Client writes the will induce it to send notifications Heart Rate Control Point for Reset Energy Expended.	Perform an action on the IUT that will induce it to send notifications of the Heart Rate Measurement characteristic along with Energy Expended values [Notifications – Energy Expended].  2. After receiving a characteristic with the Energy Expended field present, note the value of this field.  3. Write the Heart Rate Control Point characteristic value of 0x01 (Reset Energy Expended) by using the Characteristic Value Write test procedure in  4. After receiving another characteristic with the Energy Expended field present, confirm that the value in this field has been reset.	FAIL <small>(CLIENT – 3rd party not writing control point bit )</small> <small>(SERVER - Not implemented)</small>
4	Verify that the IUT responds appropriately when a Client writes an unsupported value into the Heart Rate Control Point.	Write to the Heart Rate Control Point characteristic a value from the Reserved for Future Use range by using the Characteristic Value Write  2. Verify the proper response of the IUT.	
5	This test group contains test cases to verify compliant operation in response to enable and disable characteristic notification. The verification is done one value at the time.  Pass as able to set notifications and disable them	Disable notification by writing value 0x0000 to the client characteristic configuration descriptor of the characteristic.  2. Enable notification by writing value 0x0001 to the client characteristic configuration descriptor of the characteristic.	PASS

6	<p>Verify the IUT can send notifications of the Heart Rate Measurement characteristic.</p> <p>Pass notifications are received</p>	<p>Perform an action on the IUT that will induce it to send notifications of the Heart Rate Measurement characteristic.</p> <p>2. A connection is established between the Lower Tester and IUT meeting the security requirements of the IUT, if not already done so prior to step 1.</p> <p>3. The Lower Tester receives an ATT_Handle_Value_Notification from the IUT containing the Heart Rate Measurement characteristic handle and value.</p> <p>4. Verify the characteristic value meets the requirements of the service.</p> <p>5. Repeat steps 3-4 for each received notification until the IUT stops sending notifications.</p> <p>6. The Lower Tester configures the Heart Rate Measurement characteristic to disable notifications.</p> <p>7. Repeat steps 1-2 with notifications disabled.</p> <p>8. Verify the Lower Tester does not receive an ATT_Handle_Value_Notification from the IUT containing the Heart Rate Measurement characteristic.</p>	PASS
7	<p>Verify the IUT can send notifications of the Heart Rate Measurement characteristic that include Energy Expended values.</p>	<p>Perform an action on the IUT that will induce it to send notifications of the Heart Rate Measurement characteristic along with Energy Expended values.</p> <p>2. A connection is established between the Lower Tester and IUT meeting the security requirements of the IUT, if not already done so prior to step 1.</p> <p>3. The Lower Tester receives an ATT_Handle_Value_Notification from the IUT containing the Heart Rate Measurement characteristic handle and value.</p> <p>4. Verify the characteristic value meets the requirements of the service.</p>	PASS (Based on server flag field)

8	Verify the IUT can send notifications of the Heart Rate Measurement characteristic that include RR-Interval values.	<p>Perform an action on the IUT that will induce it to send notifications of the Heart Rate Measurement characteristic along with RR-Interval values.</p> <p>2. A connection is established between the Lower Tester and IUT meeting the security requirements of the IUT, if not already done so prior to step 1.</p> <p>3. The Lower Tester receives an ATT_Handle_Value_Notification from the IUT containing the Heart Rate Measurement characteristic handle and value.</p> <p>4. Verify the characteristic value meets the requirements of the service.</p>	PASS (server flag field set accordingly)
9	Verify the IUT can send notifications of the Heart Rate Measurement characteristic that are 16 bits in length.	<p>Perform an action on the IUT that will induce it to send notifications of the Heart Rate Measurement characteristic along with 16-bit Heart Rate Measurement values.</p> <p>2. A connection is established between the Lower Tester and IUT meeting the security requirements of the IUT, if not already done so prior to step 1.</p> <p>3. The Lower Tester receives an ATT_Handle_Value_Notification from the IUT containing the Heart Rate Measurement characteristic handle and value.</p> <p>4. Verify the characteristic value meets the requirements of the service.</p>	PASS (server= advertise according to flag field get 16 bit notification client = scan enable notification if written 16 bit restricted to 8 bit only)

10.	<p>Verify the IUT can send notifications of the Heart Rate Measurement characteristic along with Sensor Contact feature information.</p>	<p>Perform an action on the IUT that will induce it to send notifications of the Heart Rate Measurement characteristic along with Sensor Contact information in the Flags field.</p> <p>2. A connection is established between the Lower Tester and IUT meeting the security requirements of the IUT, if not already done so prior to step 1.</p> <p>3. The Lower Tester receives an ATT_Handle_Value_Notification from the IUT containing the Heart Rate Measurement characteristic handle and value.</p> <p>4. Perform an action on the IUT that will induce a change to the Sensor Contact bit (e.g. contact with sensor, no contact with sensor).</p> <p>5. Verify the characteristic value meets the requirements of the service.</p>	<p>FAIL          (we're ignoring sensor info in flag field.)</p>
-----	--	--	--

### 3.12 BLE Long Read Example

#### 3.12.1 Application Overview

This application demonstrates how a GATT client device accesses a GATT server device for long read, means when user wants to read more than MTU(minimum of local and remote devices MTU's) size of data. Redpine module acts as a GATT client/server(based on user configuration) and explains reads/writes. Client role is initialized with Battery Service. Server role is initialized with a custom service.

#### Sequence of Events

This Application explains user how to:

- Advertising in SLAVE role
- Connects with remote device in MASTER role.
- Loop back the data came from the remote device
- Read request to the remote device

#### Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU

- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE peripheral device in case of Redpine module as master
- BTLE central device in case of Redpine module as slave

### 3.12.2 Configuration and Execution of the Application

#### Configuring the Application

1. Configure the below configurable macros in the Application file.  
**RSI\_BLE\_DEV\_ADDR\_TYPE** refers address type of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR_TYPE  
LE_PUBLIC_ADDRESS
```

Valid configurations are  
LE\_RANDOM\_ADDRESS  
LE\_PUBLIC\_ADDRESS

**Note:**

Depends on the remote device, address type will be changed.

**RSI\_BLE\_DEV\_ADDR** refers address of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR          "00:1A:  
7D:DA:71:13"
```

**RSI\_REMOTE\_DEVICE\_NAME** refers the name of remote device to which Redpine device has to connect

```
#define RSI_REMOTE_DEVICE_NAME  
"BLE_PERIPHERAL"
```

**Note:**

Redpine module can connect to remote device by referring either **RSI\_BLE\_DEV\_ADDR** or **RSI\_REMOTE\_DEVICE\_NAME** of the remote device.

**GATT\_ROLE** refers the GATT role of the Redpine device

#define SERVER	0
#define CLIENT	1
#define GATT_ROLE	CLIENT

valid configurations of **GATT\_ROLE** are:  
SERVER

CLIENT

**BT\_GLOBAL\_BUFF\_LEN** refers the Number of bytes required for the Application and the Driver.

#define BT_GLOBAL_BUFF_LEN	10000
----------------------------	-------

**RSI\_BLE\_CHAR\_SERV\_UUID** refers standard attribute type of characteristic service

**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers standard attribute type of client characteristic configuration descriptor.

#define RSI_BLE_CHAR_SERV_UUID	0x2803
#define RSI_BLE_CLIENT_CHAR_UUID	0x2902

**RSI\_BLE\_NEW\_SERVICE\_UUID** refers service uuid when module acts as server

**RSI\_BLE\_ATTRIBUTE\_1\_UUID** refers characteristic uuid when module acts as server

#define RSI_BLE_NEW_SERVICE_UUID	0xAABB
#define RSI_BLE_ATTRIBUTE_1_UUID	0x1AA1

**RSI\_BLE\_NEW\_CLIENT\_SERVICE\_UUID** refers service present in GATT server LE device.

**RSI\_BLE\_CLIENT\_ATTRIBUTE\_1\_UUID** refers characteristic present under above service in GATT server LE device.

#define RSI_BLE_NEW_CLIENT_SERVICE_UUID	0x180F
#define RSI_BLE_CLIENT_ATTRIBUTE_1_UUID	0x2A19

**RSI\_BLE\_MAX\_DATA\_LEN** refers the maximum attribute value length.

#define RSI_BLE_MAX_DATA_LEN	20
------------------------------	----

Following are event numbers for specific events

#define RSI_BLE_CONN_EVENT	1
#define RSI_BLE_DISCONNECT_EVENT	2
#define RSI_BLE_GATT_WRITE_EVENT	3
#define RSI_BLE_READ_REQ_EVENT	4
#define RSI_BLE_MTU_EVENT	5
#define RSI_BLE_GATT_PROFILE_RESP_EVENT	6
#define RSI_BLE_GATT_CHAR_SERVICES_RESP_EVENT	7

Following are the **non-configurable** macros in the application.

#define RSI_BLE_ATT_PROPERTY_READ	0x02
#define RSI_BLE_ATT_PROPERTY_WRITE	0x08
#define RSI_BLE_ATT_PROPERTY_NOTIFY	0x10

2. Open ***rsi\_wlan\_config.h*** file and update/modify following macros,

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	
FEAT_SECURITY_OPEN	
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP	
EXT_FEAT_256K_MODE	
#define RSI_BAND	
RSI_BAND_2P4GHZ	

3. Open ***rsi\_ble\_config.h*** file and update/modify following macros,

#define RSI_BLE_PWR_INX	8
#define RSI_BLE_PWR_SAVE_OPTIONS	0
#define RSI_DUTY_CYCLING	0

## Executing the Application

1. After the program gets executed,
2. In Client mode,  
Redpine module will trying to connect with remote device as specified by RSI\_BLE\_REMOTE\_BD\_ADDRESS or RSI\_REMOTE\_DEVICE\_NAME.
3. After connecting, mtu size will be updated. As per mtu(maximum transmit unit) size, read requests will be happen from Redpine device
4. In Server mode,  
Redpine module will advertise,
5. Initiate connection from master.
6. After connecting, mtu size will be updated. As per mtu size, write will be happen from Redpine device
7. In either role: If mtu size is of 100 bytes, module can read upto 98 bytes, write upto 97 bytes
8. For the data more than 20 bytes, application has to store value and send using gatt\_read\_response function whenever remote device reads some handle's data.

### Note

For read request event to be raised auth\_read flag in rsi\_ble\_add\_char\_val\_att function need to be set.  
Based on GATT\_ROLE configurable macro, this application will be act as a GATT server or GATT client device.

### 3.13 BLE Long Range and 2Mbps

#### 3.13.1 Application Overview

This application connects as a Central and can be used to update the phy rates .The PHY update Procedure is used to change the Transmit or receive PHYs, or both. The procedure can be initiated either on a request by the Host or autonomously by the Link Layer. Either the master or the slave may initiate this procedure at any time after entering the Connection State.The procedure can be initiated either on a request by the Host or autonomously by the Link Layer. Either the master or the slave may initiate this procedure at any time after entering the Connection State.

#### Sequence of Events

This Application explains how to use below commands:

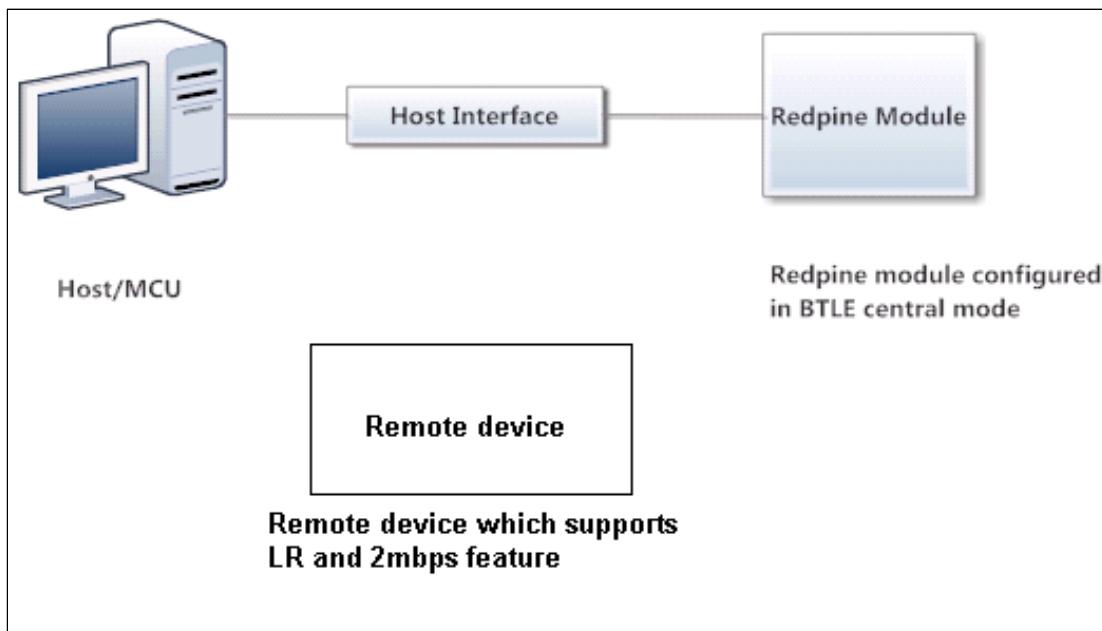
- Connect with remote BTLE peripheral device.
- Read PHY rate.
- Set PHY rate
- PHY update complete event will appear.

#### 3.13.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE peripheral device which supports BLE long range and 2Mbps feature.



**Figure 1: Setup Diagram**

### 3.13.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_datalength.c** file and update/modify following macros,  
**RSI\_BLE\_DEV\_ADDR\_TYPE** refers address type of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR_TYPE  
LE_PUBLIC_ADDRESS
```

Valid configurations are  
LE\_RANDOM\_ADDRESS  
LE\_PUBLIC\_ADDRESS

**Note:**

Depends on the remote device, address type will be changed.

**RSI\_BLE\_DEV\_ADDR** refers address of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR  
"00:1A:7D:DA:71:13"
```

**RSI\_REMOTE\_DEVICE\_NAME** refers the name of remote device to which Redpine device has to connect

```
#define RSI_REMOTE_DEVICE_NAME  
"BLE_PERIPHERAL"
```

**Note**

Redpine module can connect to remote device by referring either RSI\_BLE\_DEV\_ADDR or RSI\_REMOTE\_DEVICE\_NAME of the remote device.

Following are the event numbers for advertising, connection and Disconnection events,

#define RSI_APP_EVENT_ADV_REPORT	0
#define RSI_APP_EVENT_CONNECTED	1
#define RSI_APP_EVENT_DISCONNECTED	2
#define RSI_APP_EVENT_PHY_UPDATE_COMPLETE	3

Following are the non-configurable macros in the application.

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN
```

```
10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

```
#define RSI_BLE_PWR_INX 8
#define RSI_BLE_PWR_SAVE_OPTIONS 0
#define RSI_DUTY_CYCLING 0
```

**Note**

rsi\_wlan\_config.h and rsi\_ble\_config.h files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the remote BLE device in advertising mode
2. after the program gets executed, Redpine device tries to connect with the remote device specified in **RSI\_BLE\_DEV\_ADDR** or **RSI\_REMOTE\_DEVICE\_NAME** macro.
3. Observe that the connection is established between the desired device and Redpine device.
4. After connection, Redpine device will read PHY rate of the remote device and set PHY rate of the remote device.
5. Observe PHY update complete event after setting PHY rate.

### 3.14 Blood Pressure Profile Example

#### 3.14.1 Objective

The Blood Pressure Service exposes blood pressure and other data from a blood pressure monitor intended for health care applications.

### 3.14.2 GATT Requirements

- Notifications: Mandatory if the Intermediate Cuff Pressure characteristic is supported, otherwise excluded for this service.
- Indications : Mandatory
- Read Characteristic Descriptors : Mandatory
- Write Characteristic Descriptors : Mandatory

### 3.14.3 Service Definition

The Blood Pressure Service shall be instantiated as a «Primary Service». The service UUID shall be set to «Blood Pressure Service». The UUID value assigned to «Blood Pressure Service» UUID value 0x1810.

### 3.14.4 Service Characteristics

Characteristic Name	Requirement	Mandatory Property	Optional Property	Security Permissions
Blood Pressure Measurement	M	Indicate		None
Blood Pressure Measurement Client Characteristic Configuration descriptor	M	Read, Write		None
Intermediate Cuff Pressure	O	Notify		None
Intermediate Cuff Pressure Client Characteristic Configuration descriptor	M	Read, Write		None
Blood Pressure Feature	M	Read		

**Table 1 Blood Pressure Service Characteristics**

#### Blood Pressure Measurement

The Blood Pressure Measurement characteristic shall be used to send Blood Pressure measurements. Included in the characteristic are a Flags field (containing units of Blood Pressure and used to show presence of optional fields), the Blood Pressure Measurement Compound Value field and, depending upon the contents of the Flags field, Time Stamp (time of the measurement), Pulse Rate, User ID and Measurement Status fields.

Flags (1 byte)  [LSB]	Systolic (mm Hg) Millimeter of mercury (2 bytes)	Diastolic (mm Hg) Millimeter of mercury (2 bytes)	Mean Arterial Pressure (mm Hg) Millimeter of mercury (2 bytes)
Systolic (k Pa) Pascal (2 bytes)	Diastolic (k Pa) Pascal (2 bytes)	Mean Arterial Pressure (k Pa) Pascal (2 bytes)	Time Stamp (7 bytes)
Pulse Rate (beats per minute) (2 bytes)	User ID (1 Byte)	Measurement Status (2 bytes)	

**Table 2 Blood pressure measurement filed format**

Blood pressure unit flag (1 bit)  <ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1" ac:macro-id="623961aa-712a-4cff-89e1-d91af1e8e822"><ac:plain-text-body><![CDATA[[LSB]	Time Stamp flag (1 bit)	Pulse Rate Flag (1 bit)	User ID Flag (1 bit)	]]></ac:plain-text-body></ac:structured-macro>
Measurement Status Flag (1 bit)	Reserved for future use (3 bit)			

**Table 3 Blood pressure measurement flag field**

Body Movement Detection flag(1 bit) [LSB]	Cuff Fit Detection Flag (1 bit)	Irregular Pulse Detection Flag (1 bit)	Pulse Rate Range Detection flags (2 bits)
Measurement Position Detection Flag (1 bit)	Reserved for future use (10 bits)		

**Table 4 Blood pressure measurement status field**

### Intermediate Cuff Pressure

Used to send intermediate Cuff Pressure values to a device for display purposes while the measurement is in progress. This characteristic has the same format as the Blood Pressure Measurement characteristic. However, due to a different context, the Blood Pressure Measurement Compound Value field becomes the Intermediate Cuff Pressure Compound Value field, the Systolic sub-field becomes the Current Cuff Pressure sub-field and the Diastolic and MAP fields are unused.

Formula:

$$\text{MAP} = (\text{systolic} + 2 * (\text{Diastolic})) / 3$$

760.0 Mm Hg equals 101.325 k Pa

### Blood Pressure Feature

The Blood Pressure Feature characteristic is used to describe the supported features of the Blood Pressure Sensor.

Body Movement Detection Support bit (1 bit) [LSB]	Cuff Fit Detection Support bit (1 bit)	Irregular Pulse Detection Support bit (1 bit)	Pulse Rate Range Detection Support bit (1 bit)
Measurement position Detection Support bit (1 bit)	Multiple Bond Support bit (1 bit)	Reserved for future use (10 bits)	

**Table 5 Blood pressure feature format**

### 3.14.5 Process for Blood Pressure Server and Client

#### 3.14.6 Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization. The following sections are applicable to WiSeConnect parts only.

##### Process when acted as server

- Our device will have blood pressure service and characteristics and starts advertising.
- After connection with the third party all the supported characteristics will be visible to the third party device.
- Based on the flag field the respective fields get enable and accordingly blood pressure measurement field is displayed.
- Once notification is enabled from the client , blood press measurement field will be updated for every sec . Notifications can be disabled if not required.
- Client can write a value to the control point field , which will raise a gatt write event stating that a value is written to particular handle.

Feature	Handle	Type of attribute (UUID)	Attribute Value
Service Definition - Blood Pressure Service	X	0x2800	0x1810
Characteristic Declaration - Blood Pressure Measurement	X + 1	0x2803	0x20,X + 2,0x2A35
Characteristic Value Declaration - Blood Pressure Measurement	X + 2	0x2A35	Blood Pressure Measurement value (variable length)
Blood Pressure Measurement – Client Characteristic Configuration Descriptor	X + 3	0x2902	0x0000
Characteristic Declaration - Intermediate Cuff Pressure	X + 4	0x2803	0x10,X + 5,0x2A36
Characteristic Value Declaration - Intermediate Cuff Pressure	X + 5	0x2A36	Intermediate Cuff Pressure variable length)
Intermediate Cuff Pressure – Client Characteristic Configuration Descriptor	X + 6	0x2902	0x0000
Characteristic Declaration – Blood Pressure Feature	X + 7	0x2803	0x02,X + 8,0x2A49
Characteristic Value Declaration - Blood Pressure Feature	X + 8	0x2A49	Blood Pressure feature (2 bytes)

**Table 6 GATT Record for Blood Pressure Service**

Process when acted as Client:

- Our device will start scanning near-by devices and gets connected to the third party device.
- After connection we will get all the supported services from the remote device.
- If the third party device supports Blood pressure profile , we start reading respective characteristics that it supports.
- Characteristics with notification property are enabled and we start receiving notifications when ever any write happens to these handles.
- Based on the flag field received from the remote device blood presence measurement characteristic will be displayed.

### 3.14.7 Test-Cases

S.No	Test-Cases	Test Procedure	Status
1	<b>Service Definition:</b> Blood Pressure Service. Verify the IUT has one instantiation of the Blood Pressure Service as either a primary service or a secondary service.	<ol style="list-style-type: none"> <li>1. Send an ATT_Find_By_Type_Value_Request (0x0001, 0xFFFF), with type set to «Primary Service» and Value set to «Blood Pressure Service».</li> <li>2. If no instances of Blood Pressure Service as a primary service are found, send an ATT_Read_By_Type_Request (0x0001, 0xFFFF) to the IUT, with type set to «Include».</li> <li>3. Verify one attribute handle range is returned, containing the starting handle and the ending handle of the service definition.</li> </ol>	PASS <small>(SERVER – all the services are visible to the connected 3<sup>rd</sup> party CLIENT-Getting the services from remote device)</small>
2	<b>Characteristic Declaration:</b> Verify that the characteristic property field of the characteristic declaration meets the requirements of the service. The verification is performed one property at the time	<ol style="list-style-type: none"> <li>1. Discover all characteristics of the service.</li> <li>2. For a discovered characteristic, verify the characteristic properties field of the characteristic declaration meets the requirements of the service.</li> <li>3. Verify that mandatory characteristic blood pressure measurement and blood pressure feature are discovered.</li> </ol>	PASS <small>(SERVER – all the characteristic are visible to the connected 3<sup>rd</sup> party CLIENT-Getting the services from remote device)</small>

3	<b>Characteristic Descriptors:</b> Verify that the characteristic descriptors meet the requirements of the service. The verification is done one descriptor at the time.	<ol style="list-style-type: none"> <li>1. Discover all characteristic descriptors of the characteristic. (It returns at least one handle - UUID pair.)</li> <li>2. If the UUID in a handle - UUID pair is for a characteristic descriptor referenced in a test case, read the characteristic descriptor.</li> <li>3. Verify the value of the characteristic descriptor (mandatory Client Characteristic Configuration for blood pressure measurement) meets the requirements of the service.</li> </ol>	PASS (SERVER – all the char descriptors are visible to the connected 3 <sup>rd</sup> party CLIENT- Getting the services from remote device)
4	<b>Characteristic Read:</b> Read and verify that the characteristic values required by the service are compliant. The verification is done one value at the time	<ol style="list-style-type: none"> <li>1. If IUT permissions for the characteristic require a specific security mode or security level, establish a connection meeting those requirements.</li> <li>2. Read the characteristic value and verify the characteristic value meets the requirements of the service.</li> </ol>	PASS client can blood pressure feature characteristic

5	<p><b>Configure Indication and Notification:</b> Verify compliant operation in response to enable and disable characteristic indication or notification. The verification is done one value at the time.</p>	<ol style="list-style-type: none"> <li>1. If the IUT requires a bonding procedure then perform a bonding procedure.</li> <li>2. If IUT permissions for the characteristic descriptor require a specific security mode or security level, establish a connection meeting those requirements.</li> <li>3. Disable indication or notification by writing value 0x0000 to the client characteristic configuration descriptor of the characteristic.</li> <li>4. If the test case is for notification, enable notification by writing value 0x0001 to the client characteristic configuration descriptor of the characteristic.</li> <li>5. Otherwise, if the test case is for indication, enable indication by writing value 0x0002 to the client characteristic configuration descriptor of the characteristic.</li> <li>6. The Lower Tester reads the value of the client characteristic configuration descriptor.</li> </ol>	PASS (client need to set client configure characteristic as 0x01 for notification and 0x02 for indication)
6	<p><b>Characteristic Indication:</b> Verify compliant operation when the IUT sends indications of characteristic values. The verification is done one value at the time. Verify the IUT sends indications of characteristic values.</p>	<ol style="list-style-type: none"> <li>1. Perform an action on the IUT that will induce it to make a new value of the characteristic available.</li> <li>2. Establish the connection.</li> <li>3. The Lower Tester receives an ATT_Handle_Value_Indication from the IUT containing the characteristic handle and value.</li> <li>4. The Lower Tester sends an ATT_Handle_Value_Confirmation to the IUT.</li> <li>5. Verify the characteristic value meets the requirements of the service.</li> </ol>	PASS (client – enable the indication for blood pressure measurement characteristic using client configure characteristic server – upon reception will send indication)

7	<b>Characteristic Notification:</b> Verify compliant operation when the IUT sends notification of characteristic values. Verify the IUT sends notifications of characteristic values.	<ol style="list-style-type: none"> <li>1. The Lower Tester receives an ATT_Handle_Value_Notification from the IUT containing the Intermediate Cuff Pressure characteristic handle and value.</li> <li>2. Verify the characteristic value meets the requirements of the service.</li> </ol>	PASS (client – enable the notification for intermediate cuff pressure characteristic using client configure characteristic server – upon reception will send indication)
8	<b>Service Procedures:</b> Verify the IUT can send indications of stored Blood Pressure measurements.	<ol style="list-style-type: none"> <li>1. The Lower Tester receives an ATT_Handle_Value_Indication from the IUT containing the Blood Pressure Measurement characteristic handle and value.</li> <li>2. The Lower Tester sends an ATT_Handle_Value_Confirmation to the IUT.</li> <li>3. Repeat steps 1-2 until all stored measurements are received or the IUT terminates the connection.</li> <li>4. Verify the characteristic value in each indication contains the time stamp field.</li> <li>5. Verify the indications are received in order according to the time stamp with the oldest measurement received first.</li> </ol>	PASS

**Table 7 Blood Pressure Service Test cases**

### 3.15 BLE Data Length Extinctions Example

#### 3.15.1 Application Overview

This application acts as a Central role and can be used to set data length with connected remote device. Ble Data Packet Length Extension refers to increase in the Packet Data Unit (PDU) size from 27 to 251 bytes. This is the amount of data sent during connection events. Both the master and slave can initiate this procedure at any time after entering the Connection.

#### Sequence of Events

This Application explains sequence of below commands:

- Connect with remote BTLE peripheral device.
- Set data length.
- Data length change event will appear.

### 3.15.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE peripheral device which supports data length extension feature.

### 3.15.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_datalength.c** file and update/modify following macros,

**RSI\_BLE\_DEV\_ADDR\_TYPE** refers address type of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR_TYPE  
LE_PUBLIC_ADDRESS
```

Based on the address of the advertising device, Valid configurations are  
LE\_RANDOM\_ADDRESS  
LE\_PUBLIC\_ADDRESS

**Note**

Depends on the remote device, address type will be changed.

**RSI\_BLE\_DEV\_ADDR** refers address of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR          "00:1A:  
7D:DA:71:13"
```

**RSI\_REMOTE\_DEVICE\_NAME** refers the name of remote device to which Redpine device has to connect

```
#define RSI_REMOTE_DEVICE_NAME  
"BLE_PERIPHERAL"
```

**Note**

Redpine module can connect to remote device by referring either RSI\_BLE\_DEV\_ADDR or RSI\_REMOTE\_DEVICE\_NAME of the remote device.

Following are the event numbers for advertising, connection and Disconnection events,

#define RSI_APP_EVENT_ADV_REPORT	0
#define RSI_APP_EVENT_CONNECTED	1
#define RSI_APP_EVENT_DISCONNECTED	2
#define RSI_APP_EVENT_DATA_LENGTH_CHANGE	3

Following are the macros for setting data length(TX length and TX time)

#define TX_LEN	0x001e
#define TX_TIME	0x01f4

Following are the non-configurable macros in the application.

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

#define BT_GLOBAL_BUFF_LEN	10000
----------------------------	-------

2. Open ***rsi\_wlan\_config.h*** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

3. Open ***rsi\_ble\_config.h*** file and update/modify following macros,

#define RSI_BLE_PWR_INX	8
#define RSI_BLE_PWR_SAVE_OPTIONS	0
#define RSI_DUTY_CYCLING	0

**Note**

rsi\_wlan\_config.h and rsi\_ble\_config.h files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the remote device in peripheral mode and put it in advertising mode.
2. After the program gets executed, Redpine device tries to connect with the remote device specified in **RSI\_BLE\_DEV\_ADDR** or **RSI\_REMOTE\_DEVICE\_NAME** macro.
3. Observe that the connection is established between the desired device and Redpine device.
4. After connection Redpine device will set data length of the remote device.
5. Observe data length change event after setting data length.

## 3.16 LE Secure Connections Example

### 3.16.1 Application Overview

#### Overview

This application demonstrates how to configure the Redpine device in peripheral role and connects with remote device. By default our module has enable the SMP secure connection is enabled.

In this application, Redpine module connects with remote device and initiates SMP pairing process. After successful SMP pairing, SMP encryption will be enabled in both Central and Peripheral device.

#### Sequence of Events

This Application explains user how to:

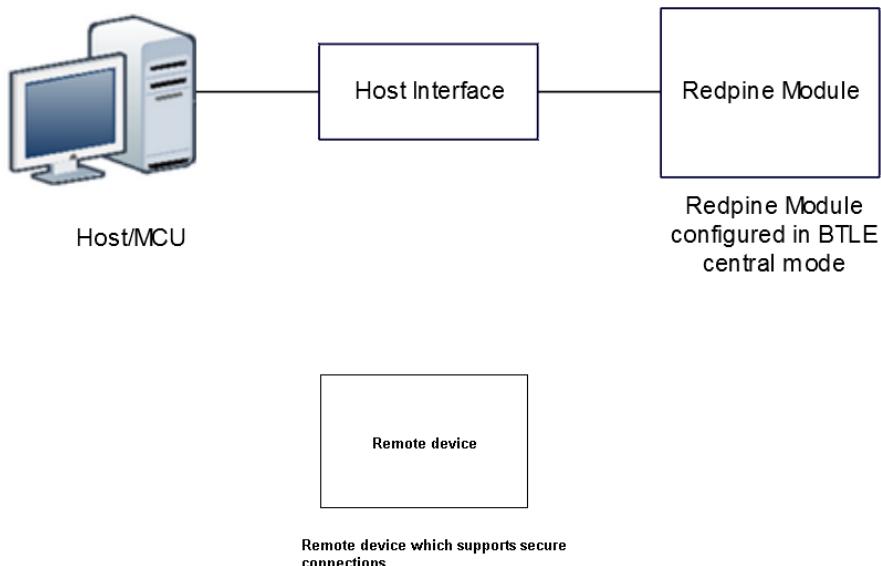
- Configure device as peripheral/Central mode
- Connect with remote device.
- Initiate SMP paring with connected remote device.
- Initiate SMP pair response for the received SMP response event.
- Received SMP passkey events on both devices
- Responding the with SMP passkey command on both sides.
- Encryption event will received on both sides.

### 3.16.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE peripheral device which supports SMP pairing(This Application uses TI sensor tag for a remote device



**Figure 1: Setup Diagram**

### 3.16.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_smp.c** file and update/modify following macros,  
**RSI\_BLE\_REMOTE\_ADDR** refers address of the remote device to connect.

```
#define RSI_BLE_REMOTE_ADDR          "00:1A:  
7D:DA:71:13"
```

**RSI\_BLE\_DEVICE\_NAME** refers the name of the WiSeConnect device to appear during scanning by remote devices.

```
#define RSI_BLE_DEVICE_NAME  
"WLAN_BLE_SIMPLE"
```

**RSI\_BLE\_SMP\_IO\_CAPABILITY** refers IO capability.

```
#define RSI_BLE_SMP_IO_CAPABILITY 0x00
```

**RSI\_BLE\_SMP\_PASSKEY** refers address type of the remote device to connect.

```
#define RSI_BLE_SMP_PASSKEY 0
```

**Following are the non-configurable macros in the application.**

#define RSI_BLE_CONN_EVENT	0x01
#define RSI_BLE_DISCONNECT_EVENT	0x02
#define RSI_BLE_SMP_REQ_EVENT	0x03
#define RSI_BLE_SMP_RESP_EVENT	0x04
#define RSI_BLE_SMP_PASSKEY_EVENT	0x05
#define RSI_BLE_SMP_FAILED_EVENT	0x06
#define RSI_BLE_ENCRYPT_STARTED_EVENT	0x07
#define RSI_BLE_SMP_PASSKEY_DISPLAY_EVENT	0x08
#define RSI_BLE_SC_PASSKEY_EVENT	0x09

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN 10000
```

2. Open ***rsi\_wlan\_config.h*** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_CUSTOM_FEATURE_BIT_MAP
#define RSI_BAND
RSI_BAND_2P4GHZ
```

0

## Executing the Application

After the program gets executed, WiSeConnect device tries to connect with the remote device specified in **RSI\_BLE\_REMOTE\_ADDR** macro.

1. Observe that the connection is established between the desired device and Redpine device.
2. After successful connection, application will initiate SMP paring and wait for SMP response event and SMP passkey request event. After receiving SMP response and SMP SC passkey events, application sends SMP response and stores passkey in numeric value and sets SMP Sc Passkey responses event. If SMP success, Device sends SMP encrypt started event to host. If not success, Device sends SMP failure event to host.

## 3.17 BLE-L2CAP Connection Based Flow Control Example

### 3.17.1 Application Overview

This application demonstrates the l2cap connection oriented channel connection.

#### Sequence of Events

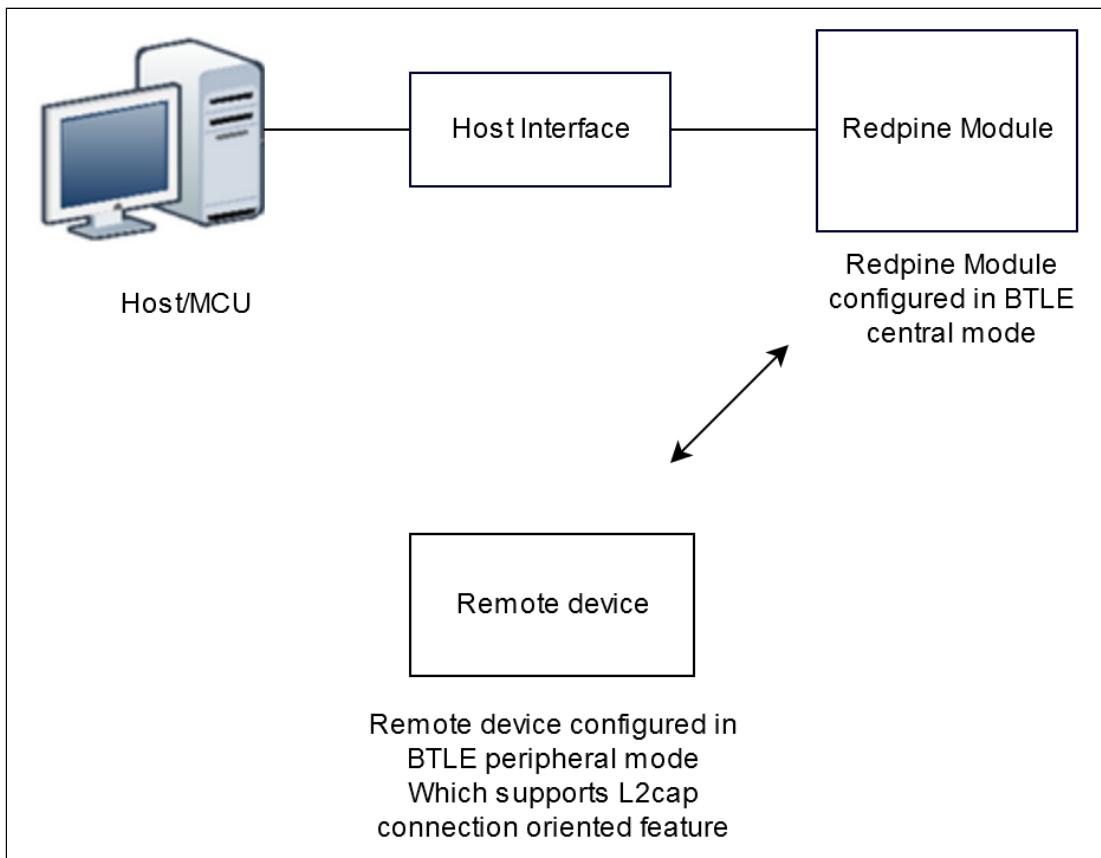
- Configure the device in Mastermode
- Scan from remote slavedevice
- Connet with the remote device.
- Configure required psm value and connect with the remote device.

#### 3.17.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE peipheral device which supports L2CAP connection based flow control feature



**Figure 1 : Setup diagram**

### 3.17.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_cbfc.c** file and update/modify following macros:

**RSI\_BLE\_LOCAL\_NAME** refers the name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_LOCAL_NAME  
"WLAN_BLE_SIMPLE_CHAT"
```

**RSI\_BLE\_DEV\_ADDR\_TYPE** refers address type of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR_TYPE  
LE_PUBLIC_ADDRESS
```

Valid configurations are  
LE\_RANDOM\_ADDRESS  
LE\_PUBLIC\_ADDRESS

**Note**

Depends on the remote device, address type will be changed.

**RSI\_BLE\_DEV\_ADDR** refers address of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR  
"00:1A:7D:DA:71:13"
```

**RSI\_REMOTE\_DEVICE\_NAME** refers the name of remote device to which Redpine device has to connect

```
#define RSI_REMOTE_DEVICE_NAME  
"BLE_PERIPHERAL"
```

**Note**

Redpine module can connect to remote device by referring either RSI\_BLE\_DEV\_ADDR or RSI\_REMOTE\_DEVICE\_NAME of the remote device.

**RSI\_SEL\_ANTENNA** refers to the antenna which is to be used by Redpine module.  
If user using internal antenna then set,

```
#define RSI_SEL_ANTENNA  
RSI_SEL_INTERNAL_ANTENNA
```

If user using external antenna (U.FL connector) then set, **RSI\_SEL\_EXTERNAL\_ANTENNA**  
Following are the **non-configurable** macros in the application.  
Following are the event numbers for events.

#define RSI_APP_EVENT_ADV_REPORT	0
#define RSI_APP_EVENT_CONNECTED	1
#define RSI_APP_EVENT_DISCONNECTED	2
#define RSI_APP_EVENT_CBFC_CONN_REQ	3
#define RSI_APP_EVENT_CBFC_CONN_CMPL	4
#define RSI_APP_EVENT_CBFC_RX_DATA	5
#define RSI_APP_EVENT_CBFC_DISCONN	6

RSI\_BLE\_PSM\_VALUE to set PSM value

```
#define RSI_BLE_PSM_VALUE
```

0x23

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN  
10000
```

2. Open **sapis/include/rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE  
RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP  
FEAT_SECURITY_OPEN  
#define RSI_TCP_IP_BYPASS  
RSI_DISABLE  
#define RSI_TCP_IP_FEATURE_BIT_MAP  
TCP_IP_FEAT_DHCPV4_CLIENT  
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP  
EXT_FEAT_256K_MODE  
#define RSI_BAND  
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

```
#define RSI_BLE_PWR_INX 8  
#define RSI_BLE_PWR_SAVE_OPTIONS 0  
#define RSI_DUTY_CYCLING 0
```

**Note**

rsi\_wlan\_config.h and rsi\_ble\_config.h files are already set with the desired configuration in respective example folders user need not change for each example

## Executing the Application

1. Configure Remote device in advertising mode.
2. Compile and launch the application.
3. After the program gets executed, Redpine module would try to connect with the device with the address specified in the macro **RSI\_BLE\_DEV\_ADDR** or **RSI\_REMOTE\_DEVICE\_NAME**
4. Observe that the connection is established between the desired device and Redpine module.
5. Now l2cap channel connection can be initiated with desired psm value.
6. After the connection got established data can be exchanged between them.

- 
7. If required l2cap disconnection command can be sent to disconnect the connection with required psm value.

## 3.18 Battery Service

### 3.18.1 Application Overview

#### Overview

This application demonstrates how to configure GATT server in BLE peripheral mode, how to configure GATT client in BLE central mode, explains how to do read & notify operations with GATT server from connected remote device using GATT client and explains how to get GATT information from remote GATT server in case of our module as client.

In this Application, Battery Service GATT server configures with Battery service with notification characteristic UUID. When connected remote device writes data to writable characteristic UUID, Redpine device receives the data which is received on writable characteristic UUID and writes the same data to readable characteristic UUID and sends notifications to the connected device (or) remote device can read the same data using read characteristic UUID if notification enabled on client side.

Battery Service GATT client will get Battery service (primary service) , Battery level service (characteristic service), and descriptors(client characteristic configuration and characteristic presentation format) information from the remote GATT server. If remote device supports notify, our module will enable notify property and will be notified by the remote GATT server when value changed.

#### Sequence of Events

#### Server

This mode explains user how to:

- Create Battery service
- Make the device to advertise
- Connect from remote BTLE device
- Receive the message from the connected peer/Smartphone
- Give the notifications to connected device

#### Client

This mode explains user how to:

- Connect to remote device based on BD address given
- Getting primary service information
- Getting characteristic services information
- Getting descriptors information
- Enable notify based on client characteristic configuration(if remote GATT server supports notify property)
- Receive notifications from remote GATT server when value changed

### 3.18.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

## WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE supported Smart phone with GATT client in case of Redpine module as GATT server
- BTLE supported Smart phone with GATT Battery server in case of Redpine as GATT client

### Note

Install Light blue App for tablet for ipad mini and BLE scanner or nRF connect app or BLE Peripheral Simulator(act like GATT server) for android smart phone.

user can download the Light blue App from the following link

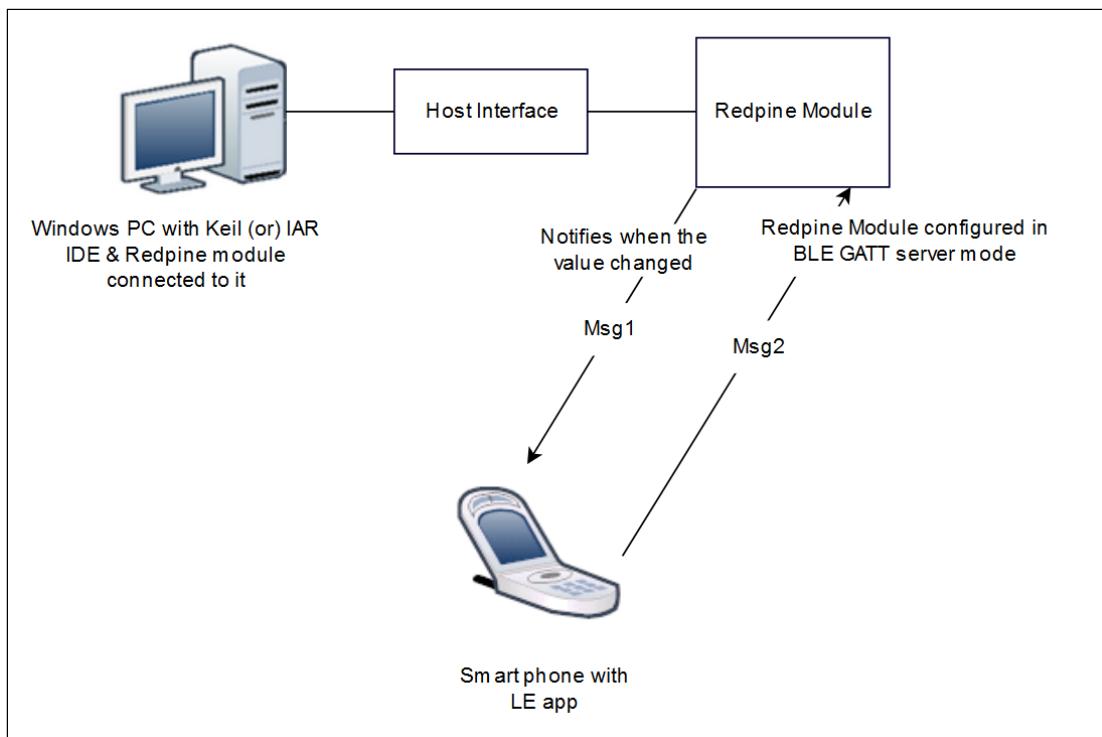
<https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer&hl=en>

user can download the nRF connect App from the following link

<https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp>

user can download the BLE Peripheral Simulator App from the following link

<https://play.google.com/store/apps/details?id=io.github.webbluetoothcg.bletestperipheral>



**Figure 1: Setup Diagram**

### 3.18.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_battery\_service.c** file and update/modify following macros,  
**RSI\_BLE\_BATTERY\_SERVICE\_UUID** refers to the attribute value of the newly created service.

```
#define RSI_BLE_BATTERY_SERVICE_UUID  
0x180F
```

**RSI\_RSI\_BLE\_BATTERY\_LEVEL\_UUID** refers to the attribute type of the first attribute under this above primary service.

```
#define RSI_RSI_BLE_BATTERY_LEVEL_UUID  
0x2A19
```

**RSI\_BLE\_MAX\_DATA\_LEN** refers to the Maximum length of the attribute data.

```
#define RSI_BLE_MAX_DATA_LEN
```

20

**BLE\_BATTERY\_SERVICE** refers name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_APP_SIMPLE_CHAT  
"BLE_BATTERY_SERVICE"
```

**GATT\_ROLE** refers the role of the Redpine module to be selected.

If user configure **SERVER**, Redpine module will act as GATT SERVER, means will add battery service profile.  
If user configure **CLIENT**, Redpine module will act as GATT CLIENT, means will connect to remote GATT server and get services.

```
#define GATT_ROLE  
"SERVER"
```

If user configure **CLIENT** role following macros should be configured.

**RSI\_BLE\_DEV\_ADDR\_TYPE** refers address type of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR_TYPE  
LE_PUBLIC_ADDRESS
```

Valid configurations are  
LE\_RANDOM\_ADDRESS  
LE\_PUBLIC\_ADDRESS

**Note**

Depends on the remote device, address type will be changed.

**RSI\_BLE\_DEV\_ADDR** refers address of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR  
"00:1A:7D:DA:71:13"
```

**RSI\_REMOTE\_DEVICE\_NAME** refers the name of remote device to which Redpine device has to connect

```
#define RSI_REMOTE_DEVICE_NAME  
"BLE_PERIPHERAL"
```

**Note**

Redpine module can connect to remote device by referring either RSI\_BLE\_DEV\_ADDR or RSI\_REMOTE\_DEVICE\_NAME of the remote device.

Following Characteristic Presentation Format fields

```
#define RSI_BLE_FORMAT  
0x04  
#define RSI_BLE_EXPONENT  
0x00  
#define RSI_BLE_UNIT  
0x27AD  
#define RSI_BLE_NAME_SPACE  
0x01  
#define RSI_BLE_DESCRIPTION  
0x00
```

Following are the non configurable macros related to client characteristic configuration.

```
#define RSI_BLE_NOTIFY  
0x01  
#define RSI_BLE_INDICATE  
0x02
```

Following are the **non-configurable** macros in the application.

**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.

```
#define RSI_BLE_CHAR_SERV_UUID  
0x2803
```

**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

```
#define RSI_BLE_CLIENT_CHAR_UUID  
0x2902
```

**RSI\_BLE\_CHAR\_PRESENTATION\_FORMAT\_UUID** refers to the attribute type of the characteristic presentation format descriptor to be added in a service.

```
#define RSI_BLE_CHAR_PRESENTATION_FORMAT_UUID  
0x2904
```

Following are the macros for the GATT properties

```
#define RSI_BLE_ATT_PROPERTY_READ  
0x02  
#define RSI_BLE_ATT_PROPERTY_WRITE  
0x08  
#define RSI_BLE_ATT_PROPERTY_NOTIFY  
0x10
```

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN  
10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

#define RSI_BLE_PWR_INX	8
#define RSI_BLE_PWR_SAVE_OPTIONS	0
#define RSI_DUTY_CYCLING	0

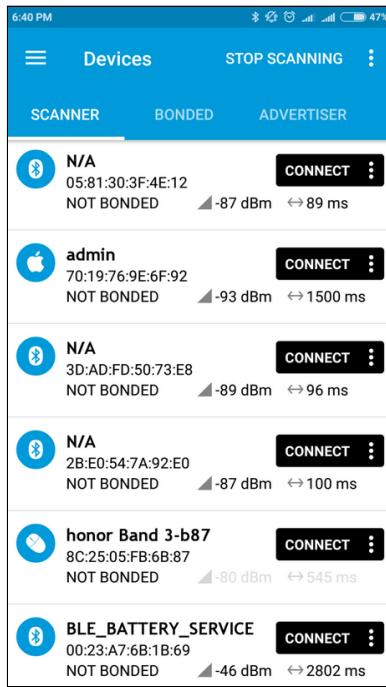
**Note:**

**rsi\_wlan\_config.h** and **rsi\_ble\_config.h** files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

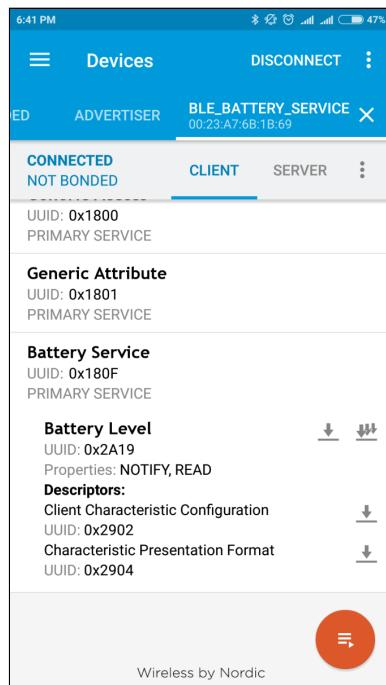
### Server role

1. After the program gets executed, Redpine module will be in Advertising state.
2. Open a nRFConnect App and do the scan.
3. In the App, Redpine module will appear with the name configured in the macro **RSI\_BLE\_APP\_SIMPLE\_CHAT** (Ex: "BLE\_BATTERY\_SERVICE") or sometimes observed as Redpine device as internal name "SimpleBLEPeripheral".

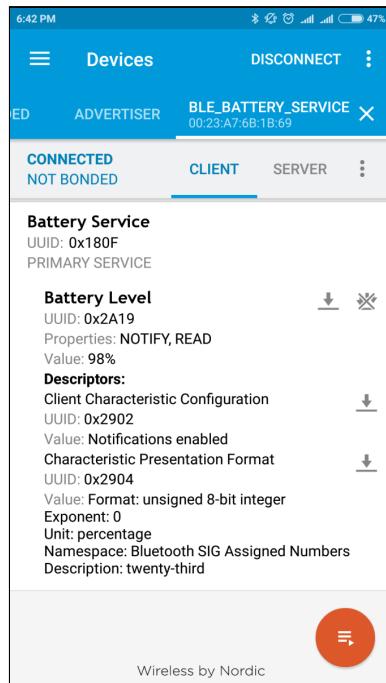


**Figure 2: Scanning for RSI\_BATTERY\_SERVICE device and connecting to it**

4. Initiate connection from the App.
5. After successful connection, nRFConnect displays the supported services of Redpine module.
6. Select the attribute service which is added **RSI\_BLE\_BATTERY\_SERVICE\_UUID**  
(Ex: 0x180F).
7. Enable Notify for the characteristic **RSI\_BLE\_BATTERY\_LEVEL\_UUID**  
(Ex: 0x2A19). So that GATT server Notifies when value updated in that particular attribute.
8. Redpine module send the Battery Service battery level data to the attribute **RSI\_BLE\_BATTERY\_LEVEL\_UUID** (Ex: 0x2A19) of the remote device and will Notifies the GATT client (remote device).
9. **RSI\_BLE\_CHAR\_PRESENTATION\_FORMAT\_UUID** will describe the value by its fields as shown in fig.
10. Please refer the given below images for Notify operation from remote device GATT client.



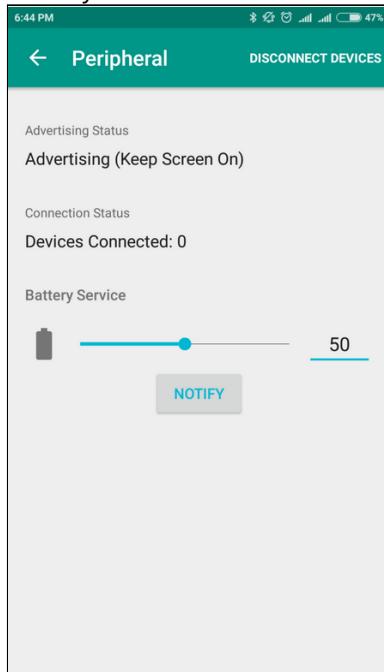
**Figure 3: Battery service and its characteristic service(Battery level)**



**Figure 4: Descriptors: Client Characteristic Configuration and Characteristic Presentation Format**

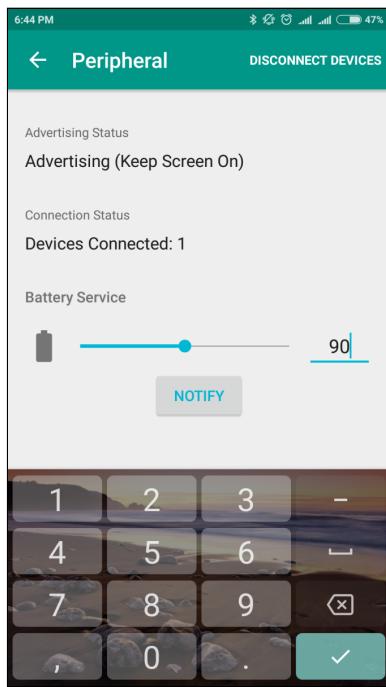
## Client role

1. Advertise a LE device which supports Battery Service.



**Figure 5: Advertising LE device which supports Battery service**

2. After the program gets executed, Redpine module will connect to that remote device based on given BD address or name
3. After successful connection Redpine module will read the services from the remote GATT server.
4. If remote device support notify property Redpine module will enable notify, and ready to receive notifications from remote device.
5. Whenever GATT server changes value and notifies that Redpine module will receive that value.



**Figure 6: Change the value and notify**

### 3.19 Health Thermometer

#### 3.19.1 Application Overview

##### Overview

This application demonstrates how to configure Health thermometer as GATT server in BLE peripheral mode and explains how to do indicate operation with GATT server from connected remote device using GATT client. In this Application, Health thermometer GATT server configures with health thermometer service with indicate characteristic UUID. When connected remote device writes data to writable characteristic UUID, Redpine device receives the data which is received on writable characteristic UUID and writes the same data to readable characteristic UUID and sends indications to the connected device (or) remote device can read the same data using read characteristic UUID if indication enabled on client side.

##### Sequence of Events

This Application explains user how to:

- Create Health thermometer service
- Make the device to advertise
- Connect from remote BTLE device
- Receive the message from the connected peer/Smartphone
- Give the indications to connected device

#### 3.19.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The

Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE supported Smart phone with GATT client

##### Note

Install Light blue App for tablet for ipad mini and BLE scanner or nRF connect app for android smart phone.  
user can download the BLE scanner App from the following link

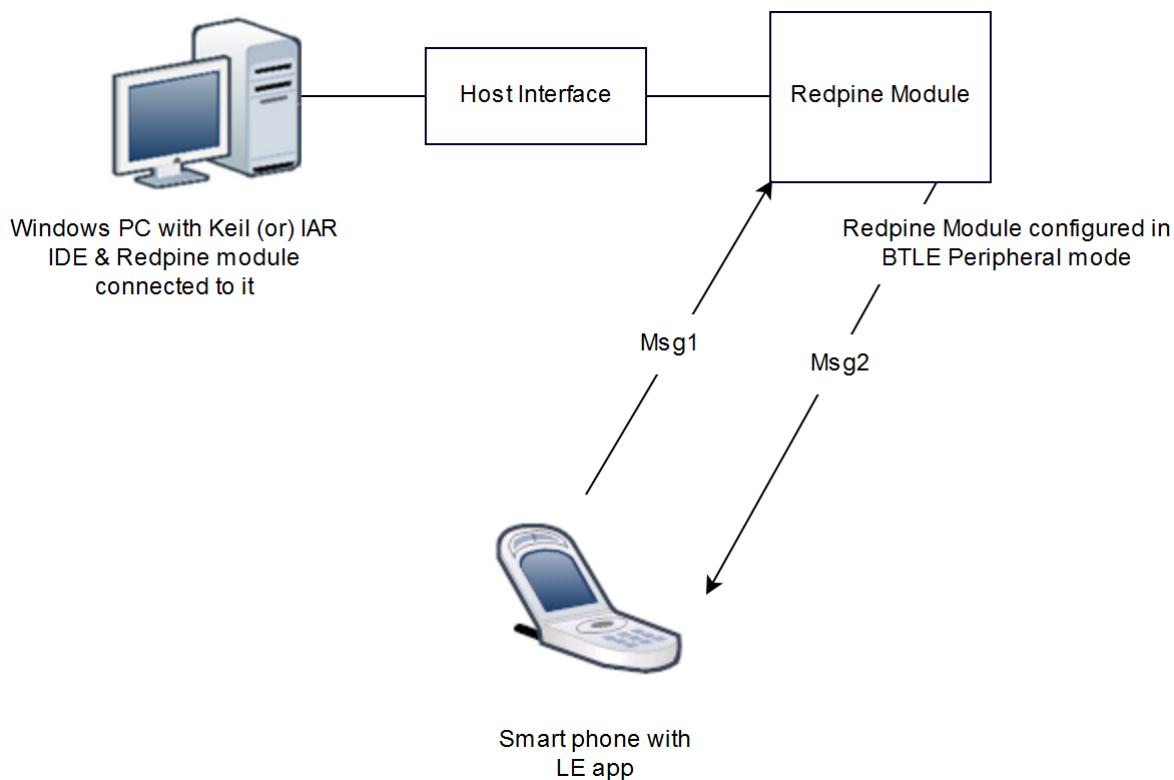
<https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner&hl=en>

user can download the Light blue App from the following link

<https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer&hl=en>

user can download the nRF connect App from the following link

<https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp>



**Figure 1 : Setup Diagram**

### 3.19.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_health\_thermometer.c** file and update/modify following macros,  
**RSI\_BLE\_HEALTH\_THERMOMETER\_UUID** refers to the attribute value of the newly created service.

```
#define RSI_BLE_HEALTH_THERMOMETER_UUID 0x1809
```

**RSI\_BLE\_TEMPERATURE\_MEASUREMENT\_UUID** refers to the attribute type of the first attribute under this service (**RSI\_BLE\_HEALTH\_THERMOMETER\_UUID**).

**RSI\_BLE\_TEMPERATURE\_TYPE\_UUID** refers to the attribute type of the second attribute under this service (**RSI\_BLE\_HEALTH\_THERMOMETER\_UUID**).

**RSI\_BLE\_INTERMEDIATE\_TEMPERATURE\_UUID** refers to the attribute type of the second attribute under this service (**RSI\_BLE\_HEALTH\_THERMOMETER\_UUID**).

```
#define RSI_BLE_TEMPERATURE_MEASUREMENT_UUID 0x2A1C
#define RSI_BLE_TEMPERATURE_TYPE_UUID 0x2A1D
#define RSI_BLE_INTERMEDIATE_TEMPERATURE_UUID 0x2A1E
```

**RSI\_BLE\_MAX\_DATA\_LEN** refers to the Maximum length of the attribute data.

```
#define RSI_BLE_MAX_DATA_LEN 20
```

**BLE\_HEALTH\_THERMOMETER** refers name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_APP_SIMPLE_CHAT
"BLE_HEALTH_THERMOMETER"
```

Following are the **non-configurable** macros in the application.

**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.

**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

```
#define RSI_BLE_CHAR_SERV_UUID 0x2803
#define RSI_BLE_CLIENT_CHAR_UUID 0x2902
```

Following are the GATT properties

#define RSI_BLE_ATT_PROPERTY_READ	0x02
#define RSI_BLE_ATT_PROPERTY_WRITE	0x08
#define RSI_BLE_ATT_PROPERTY_NOTIFY	0x10
#define RSI_BLE_ATT_PROPERTY_INDICATE	0x20

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

#define BT_GLOBAL_BUFF_LEN	10000
----------------------------	-------

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

```
#define RSI_BLE_PWR_INX          8
#define RSI_BLE_PWR_SAVE_OPTIONS   0
#define RSI_DUTY_CYCLING          0
```

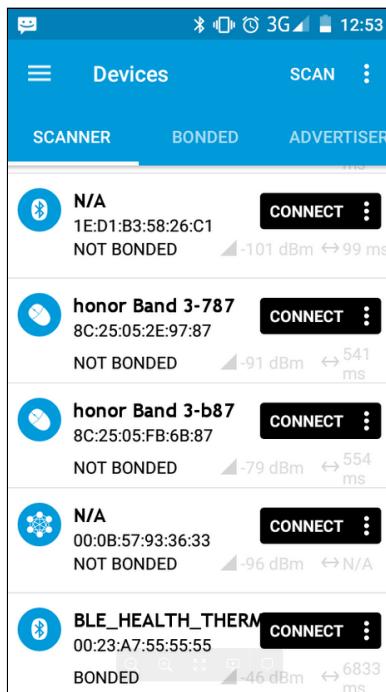
**Note**

rsi\_wlan\_config.h and rsi\_ble\_config.h files are already set with desired configuration in respective example folders user need not change for each example.

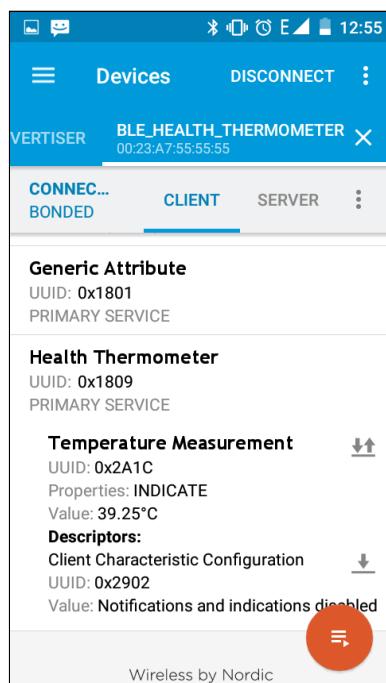
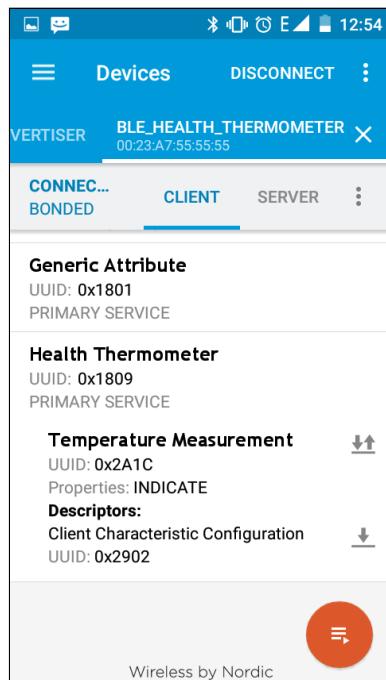
## Executing the Application

1. After the program gets executed, Redpine module will be in Advertising state.
2. Open a LE App in the Smartphone and do the scan.

- 
3. In the App, Redpine module device will appear with the name configured in the macro **RSI\_BLE\_APP\_SIMPLE\_CHAT** (Ex: "BLE\_HEALTH\_THERMOMETER") or sometimes observed as Redpine device as internal name "**SimpleBLEPeripheral**".



4. Initiate connection from the App.  
5. After successful connection, LE scanner displays the supported services of Redpine module.  
6. Select the attribute service which is added **RSI\_BLE\_HEALTH\_THERMOMETER\_UUID** (Ex: 0x1809).  
7. Enable Indicate for the characteristic **RSI\_BLE\_TEMPERATURE\_MEASUREMENT\_UUID** (Ex: 0x2A1C). So that GATT server indicates when value updated in that particular attribute.  
8. Redpine module send the health thermometer temperature measurement data to the attribute **RSI\_BLE\_TEMPERATURE\_MEASUREMENT\_UUID** (Ex: 0x2A1C) of the remote device and will indicates the GATT client (remote device).  
9. Please refer the given below images for indicate operation from remote device GATT client.



---

## 3.20 BLE PER Example

### 3.20.1 Application Overview

#### Overview

This application demonstrates how to configure the necessary parameters to start transmit or receive BLE PER packets.

#### Sequence of Events

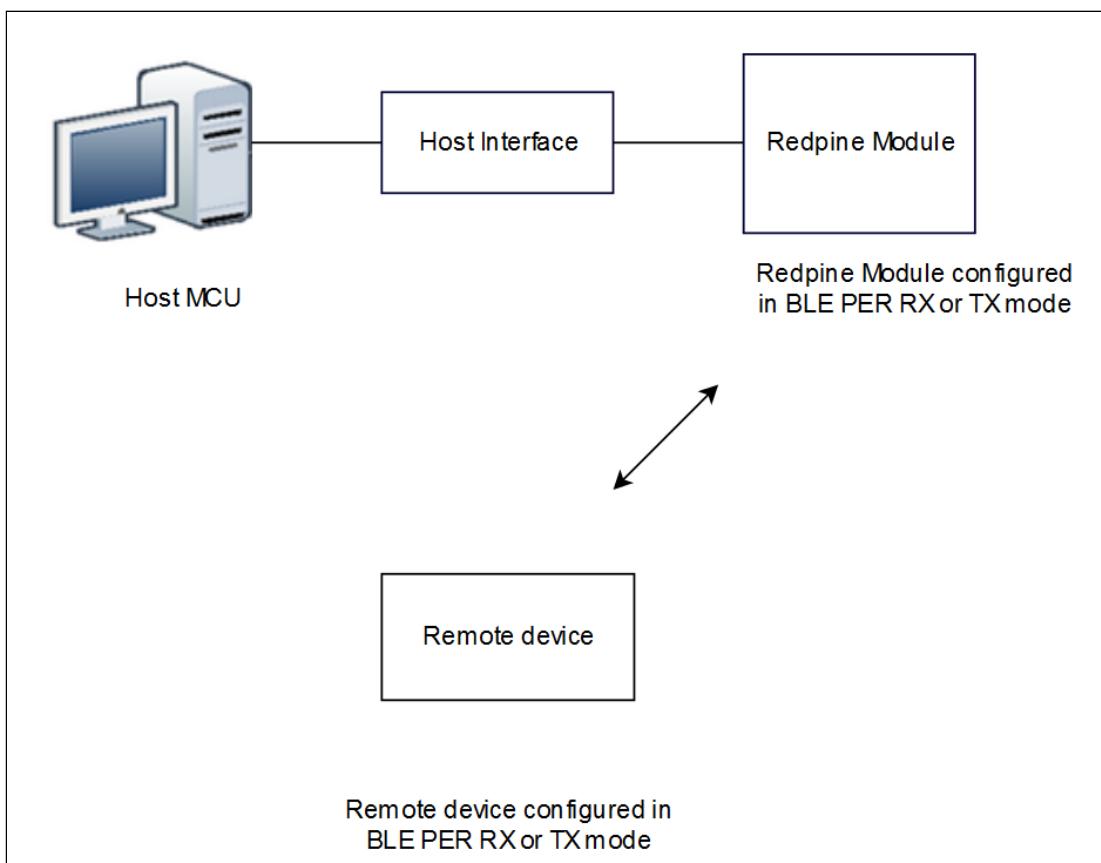
This Application explains user how to:  
Configure the BLE PER TX or RX mode.

### 3.20.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Remote device for the BLE TX or RX mode on the other side



**Figure 1: Setup Diagram**

### 3.20.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_per.c** file and update/modify following macros,

**RSI\_CONFIG\_PER\_MODE** refers configuration mode BT PER TX or RX

```
#define RSI_CONFIG_PER_MODE  
RSI_BLE_PER_TRANSMIT_MODE  
OR  
#define RSI_CONFIG_PER_MODE  
RSI_BLE_PER_RECEIVE_MODE
```

CMD\_ID refers the command id for transmit or receive

```
#define BLE_TRANSMIT_CMD_ID          0x13  
#define BLE_RECEIVE_CMD_ID           0x14
```

**PAYLOAD\_TYPE** refers type of payload to be transmitted

```
#define DATA_PRBS9                  0x00  
#define DATA_FOUR_ONES_FOUR_ZEROES   0x01  
#define DATA_ALT_ONES_AND_ZEROES     0x02  
#define DATA_PRSB15                 0x03  
#define DATA_ALL_ONES                0x04  
#define DATA_ALL_ZEROES              0x05  
#define DATA_FOUR_ZEROES_FOUR_ONES   0x06  
#define DATA_ALT_ZEROES_AND_ONES     0x07
```

**LE\_CHNL\_TYPE:** advertising channel - 0      data channel - 1

```
#define LE_ADV_CHNL_TYPE            0  
#define LE_DATA_CHNL_TYPE           1
```

**PACKET\_LEN:** Length of the packet, in bytes, to be transmitted. Packet length range 0 to 255.

```
#define BLE_TX_PKT_LEN             32
```

**BLE\_RX\_CHNL\_NUM**- Receive channel index, as per the Bluetooth standard.i.e, 0 to 39

**BLE\_TX\_CHNL\_NUM** - Transmit channel index, as per the Bluetooth standard. i.e, 0 to 39

```
#define BLE_RX_CHNL_NUM            10  
#define BLE_TX_CHNL_NUM             10
```

**BLE\_PHY\_RATE:** ,2Mbps - 2 , 125Kbps - 4, 500Kbps - 8

```
#define LE_ONE_MBPS                1  
#define LE_TWO_MBPS                2  
#define LE_125_KBPS_CODED           4  
#define LE_500_KBPS_CODED           8  
#define BLE_PHY_RATE                LE_ONE_MBPS
```

**SCRAMBLER\_SEED:** Initial seed to be used for whitening. It should be set to '0' in order to disable whitening.

```
#define SCRAMBLER_SEED 0
```

**TX\_MODE:** Burst mode - 0 Continuous mode - 1

```
#define BURST_MODE 0  
#define CONTINUOUS_MODE 1
```

**HOPPING TYPE:** no hopping -0 fixed hopping -1 random hopping -2

```
#define NO_HOPPING 0  
#define FIXED_HOPPING 1  
#define RANDOM_HOPPING 2
```

**ANT\_SEL:** onchip antenna - 2 u.f.l - 3

```
#define ONBOARD_ANT_SEL 2  
#define EXT_ANT_SEL 3
```

**RF\_TYPE:** External RF – 0 Internal RF – 1

```
#define BLE_EXTERNAL_RF 0  
#define BLE_INTERNAL_RF 1
```

**RF CHAIN:** Select the required RF chain

```
#define NO_CHAIN_SEL 0  
#define WLAN_HP_CHAIN_BIT 0  
#define WLAN_LP_CHAIN_BIT 1  
#define BT_HP_CHAIN_BIT 2  
#define BT_LP_CHAIN_BIT 3
```

pll\_mode: PLL\_MODE0 – 0 PLL\_MODE1 – 1

```
#define PLL_MODE_0 0  
#define PLL_MODE_1 1
```

**LOOP\_BACK\_MODE**: enable 1 or disable 0 #define LOOP\_BACK\_MODE\_DISABLE 0

#define LOOP_BACK_MODE_ENABLE	1
-------------------------------	---

Following are the non-configurable macros in the application.

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

#define BT_GLOBAL_BUFF_LEN	10000
----------------------------	-------

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP	EXT_FEAT_256K_MODE
#define RSI_BAND	RSI_BAND_2P4GHZ

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

#define RSI_BLE_PWR_INX	8
#define RSI_BLE_PWR_SAVE_OPTIONS	0
#define RSI_DUTY_CYCLING	0

**Note**

rsi\_wlan\_config.h and rsi\_ble\_config.h files are already set with desired configuration in respective example folders user need not change for each example

## Executing the Application

1. After the program gets executed, Redpine module starts BLE PER transmit or BLE PER receive.
2. For receiving purpose use BT dongle and keep it in BLE PER RX mode.
3. Check for BLE PER stats whatever configured values are effecting or not.

### 3.21 Blood Pressure Service (BLS) Example

#### 3.21.1 Definitions and Acronyms

- BLS - Blood Pressure Service
- GATT - Generic Attribute Profile

- UUID - Universal unique identifier
- BLE - Bluetooth low energy
- BD - Bluetooth Device

### 3.21.2 Application Overview

This application demonstrates how to configure GATT server in BLE peripheral mode, how to configure GATT client in BLE central mode, explains how to do read, notify and indicate operations with GATT server from connected remote device using GATT client and explains how to get GATT information from remote GATT server in case of our module as client.

In this Application, Blood Pressure Service GATT server configures with Blood Pressure service with notification characteristic UUID. When connected remote device writes data to writable characteristic UUID, Redpine device receives the data which is received on writable characteristic UUID and writes the same data to readable characteristic UUID and sends notifications to the connected device (or) remote device can read the same data using read characteristic UUID if notification enabled on client side.

Blood Pressure Service GATT client will get Blood Pressure service (primary service), Blood Pressure Measurement service (characteristic service), and descriptors(client characteristic configuration and characteristic presentation format) information from the remote GATT server. If remote device supports notify, our module will enable notify property and will be notified by the remote GATT server when value changed.

### 3.21.3 Sequence of Events

#### Server Role

This mode explains user how to:

- Create Blood Pressure service
- Make the device to advertise
- Connect from remote BTLE device
- Receive the message from the connected peer/Smart phone
- Give the notifications to connected device.

#### Client Role

This mode explains user how to:

- Connect to remote device based on BD address given
- Getting primary service information
- Getting characteristic services information
- Getting descriptors information
- Enable notify based on client characteristic configuration(if remote GATT server supports notify property)
- Receive notifications from remote GATT server when value changed.

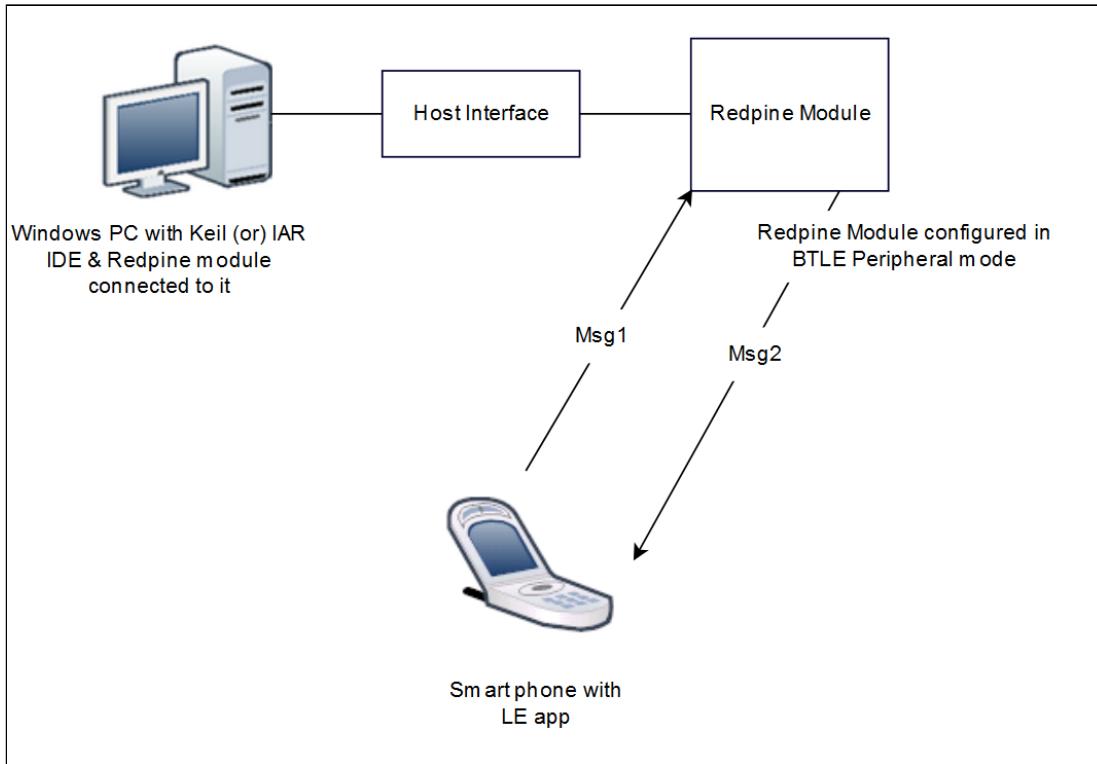
### 3.21.4 Application Setup

The Redpine module WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE supported Smart phone with GATT client in case of our module as GATT server

- BTLE supported Smart phone with GATT Blood Pressure server in case of our module as GATT client



**Figure 1: Setup Diagram**

**Note**

Install Light blue App for tablet for ipad mini and BLE scanner or nRF connect app user can download the BLE scanner App from the following link

<https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner&hl=en>

user can download the Light blue App from the following link

<https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer&hl=en>

user can download the nRF connect App from the following link

<https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp>

### 3.21.5 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_blood\_pressure.c** file and update/modify following macros.

**RSI\_BLE\_BLOOD\_PRESSURE\_SERVICE\_UUID** refers to the attribute value of the newly created service.

```
#define RSI_BLE_BLOOD_PRESSURE_SERVICE_UUID 0x1810
```

**RSI\_BLE\_BLOOD\_PRESSURE\_MEASUREMENT\_UUID** refers to the attribute type of the first attribute under this above primary service.

**RSI\_BLE\_INTERMEDIATE\_CUFF\_PRESSURE\_UUID** refers to the attribute type of the second attribute under this above primary service.

**RSI\_BLE\_BLOOD\_PRESSURE\_FEATURE\_UUID** refers to the attribute type of the third attribute under this above primary service.

```
#define RSI_BLE_BLOOD_PRESSURE_MEASUREMENT_UUID      0x2A35
#define RSI_BLE_INTERMEDIATE_CUFF_PRESSURE_UUID        0x2A36
#define RSI_BLE_BLOOD_PRESSURE_FEATURE_UUID            0x2A49
```

**RSI\_BLE\_APP\_BLOOD\_PRESSURE** refers name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_APP_BLOOD_PRESSURE                  "BLS"
```

**GATT\_ROLE** refers the role of the Redpine module to be selected.

If user configure **SERVER**, Redpine module will act as GATT SERVER, means will add blood pressure service profile.

If user configure **CLIENT**, Redpine module will act as GATT CLIENT, means will connect to remote GATT server and get services.

```
#define GATT_ROLE                                     SERVER
```

If user configure **CLIENT** role following macros should be configured.

**RSI\_BLE\_DEV\_ADDR\_TYPE** refers address type of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR_TYPE
LE_PUBLIC_ADDRESS
```

Valid configurations are  
**LE\_RANDOM\_ADDRESS**  
**LE\_PUBLIC\_ADDRESS**

**RSI\_BLE\_DEV\_ADDR** refers address of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR                         "00:1A:
7D:DA:71:13"
```

**RSI\_REMOTE\_DEVICE\_NAME** refers the name of remote device to which Redpine device has to connect

```
#define RSI_REMOTE_DEVICE_NAME  
"BLE_PERIPHERAL"
```

**Note**

User can configure either RSI\_BLE\_DEV\_ADDR or RSI\_REMOTE\_DEVICE\_NAME of the remote device.

Following are the non configurable macros related to attribute properties.

#define RSI_BLE_ATT_PROPERTY_READ	0x02
#define RSI_BLE_ATT_PROPERTY_WRITE	0x08
#define RSI_BLE_ATT_PROPERTY_WRITE_WITHOUT_RESP	0x04
#define RSI_BLE_ATT_PROPERTY_NOTIFY	0x10
#define RSI_BLE_ATT_PROPERTY_INDICATE	0x20

Following are the **non-configurable** macros in the application.

**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.  
**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

#define RSI_BLE_CHAR_SERV_UUID	0x2803
#define RSI_BLE_CLIENT_CHAR_UUID	0x2902

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

#define BT_GLOBAL_BUFF_LEN	10000
----------------------------	-------

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros, #define RSI\_BLE\_PWR\_INX 8

#define RSI_BLE_PWR_SAVE_OPTIONS	0
#define RSI_DUTY_CYCLING	0

**Note:**

**rsi\_wlan\_config.h** and **rsi\_ble\_config.h** files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

### Server role

1. After the program gets executed, Redpine module will be in Advertising state.
2. Open a nRFConnect App and do the scan. (see Figure 2)
3. In the App, Redpine module will appear with the name configured in the macro **RSI\_BLE\_APP\_BLOOD\_PRESSURE** (Ex: "BLS") or sometimes observed as Redpine device as internal name "**SimpleBLEPeripheral**".
4. Initiate connection from the App.
5. After successful connection, nRFConnect displays the supported services of Redpine module.
6. Select the attribute service which is added **RSI\_BLE\_BLOOD\_PRESSURE\_SERVICE\_UUID** (Ex: 0x1810). (see Figure 3 and 4)
7. Enable Notify for the characteristic **RSI\_BLE\_BLOOD\_PRESSURE\_MEASUREMENT\_UUID** (Ex: 0x2A35). So that GATT server Notifies when value updated in that particular attribute.(see Figure 5)
8. Redpine module send the Blood pressure measurement value to the attribute **RSI\_BLE\_BLOOD\_PRESSURE\_MEASUREMENT\_UUID** (Ex: 0x2A35) of the remote device and will indicates the GATT client (remote device).
9. **RSI\_BLE\_BLOOD\_PRESSURE\_FEATURE\_UUID** will describe the value by its fields as shown in fig.

Following are the snapshots of smart phone nRFConnect App act as a client and redpine device as a server.

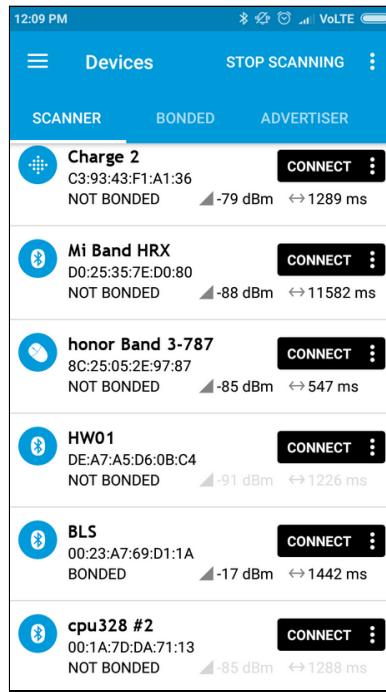


Figure 2: Scanning for BLS device & connecting to it

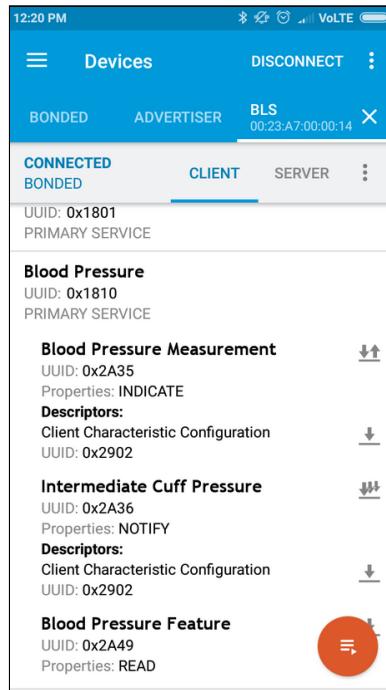
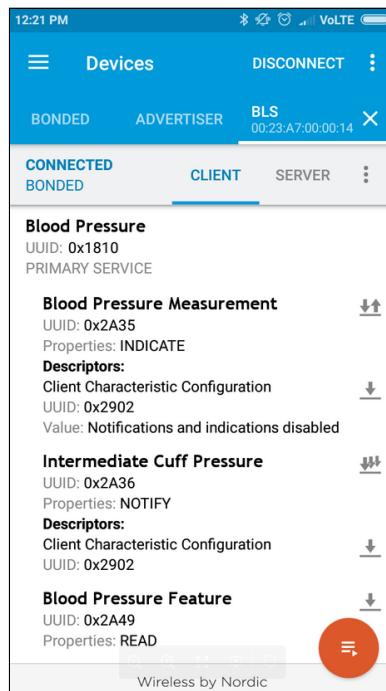
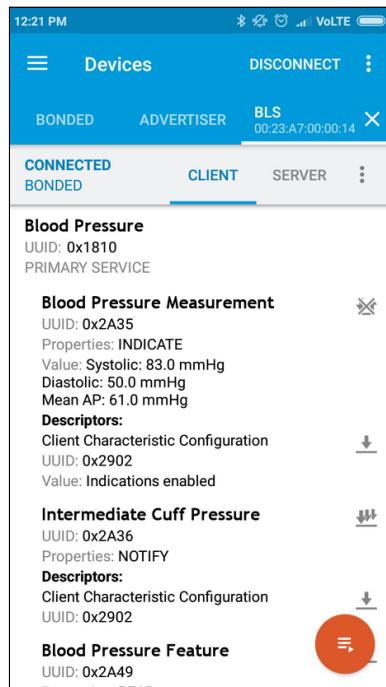


Figure 3: BLS and its characteristic discovery



**Figure 4: Client Characteristic Configuration (Indication disable)**



**Figure 5: Blood pressure measurement value (Indication enable)**

## Client role

1. Advertise a LE device which supports Blood Pressure Service.
  2. After the program gets executed, Redpine module will connect to that remote device based on given BD address.
  3. After successful connection Redpine module will read the services from the remote GATT server.
  4. If remote device support notify property Redpine module will enable notify, and ready to receive notifications from remote device.
  5. Whenever GATT server changes value and notifies that Redpine module will receive that value.
- Following are the snapshots of smart phone nRFConnect App act as a client and redpine device as a server.

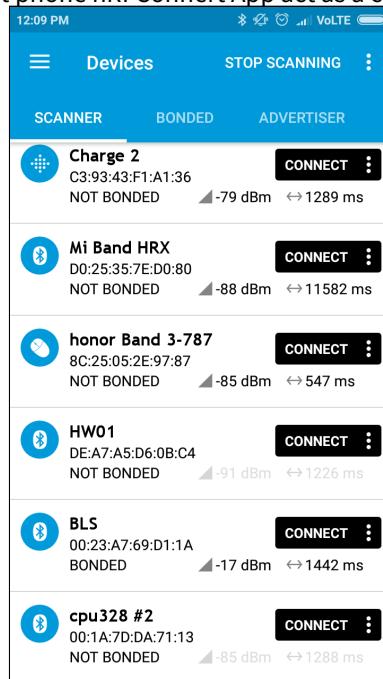


Figure 6: Scanning for BLS device & connecting to it

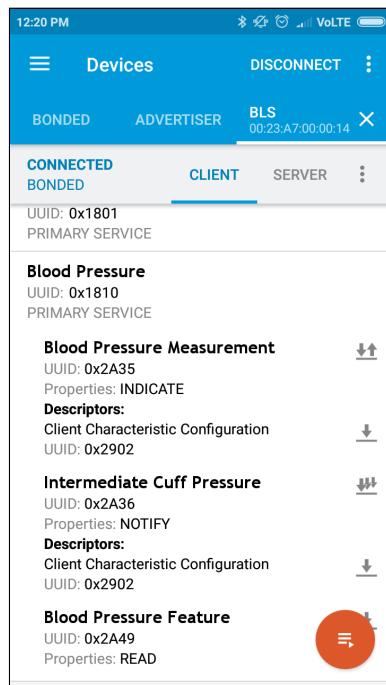


Figure 7: BLS and its characteristic discovery

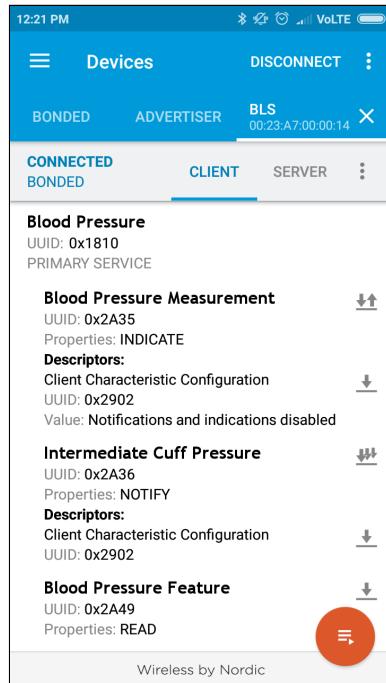
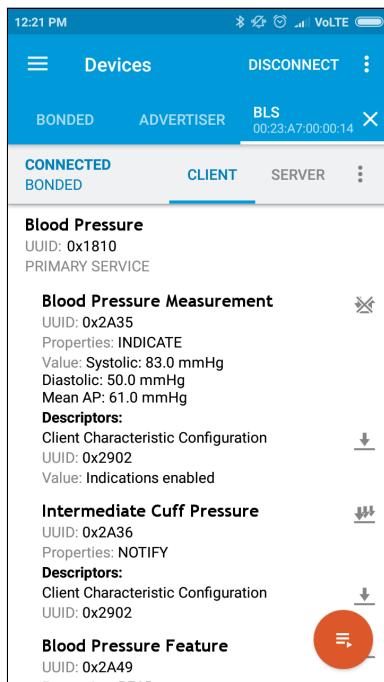


Figure 8: Client Characteristic Configuration (Indication disable)



**Figure 9: Blood pressure measurement value (Indication enable)**

## 3.22 Glucose Service (GLS) Example

### 3.22.1 Definitions and Acronyms

- GLS - Glucose Service
- GATT - Generic Attribute Profile
- UUID - Universal unique identifier
- BLE - Bluetooth low energy
- BD - Bluetooth Device

### 3.22.2 Application Overview

This application demonstrates how to configure GATT server in BLE peripheral mode, how to configure GATT client in BLE central mode, explains how to do read, notify and indicate operations with GATT server from connected remote device using GATT client and explains how to get GATT information from remote GATT server in case of our module as client.

In this Application, Glucose Service GATT server configures with Glucose service with notification characteristic UUID. When connected remote device writes data to writable characteristic UUID, Redpine device receives the data which is received on writable characteristic UUID and writes the same data to readable characteristic UUID and sends notifications to the connected device (or) remote device can read the same data using read characteristic UUID if notification enabled on client side.

Glucose Service GATT client will get Glucose service (primary service) , Glucose Measurement service (characteristic service), and descriptors(client characteristic configuration and characteristic presentation format) information from the remote GATT server. If remote device supports notify, our module will enable notify property and will be notified by the remote GATT server when value changed.

---

### 3.22.3 Sequence of Events

#### Server Role

This mode explains user how to:

- Create Glucose service
- Make the device to advertise
- Connect from remote BTLE device
- Receive the message from the connected peer/Smart phone
- Give the notifications to connected device.

#### Client Role

This mode explains user how to:

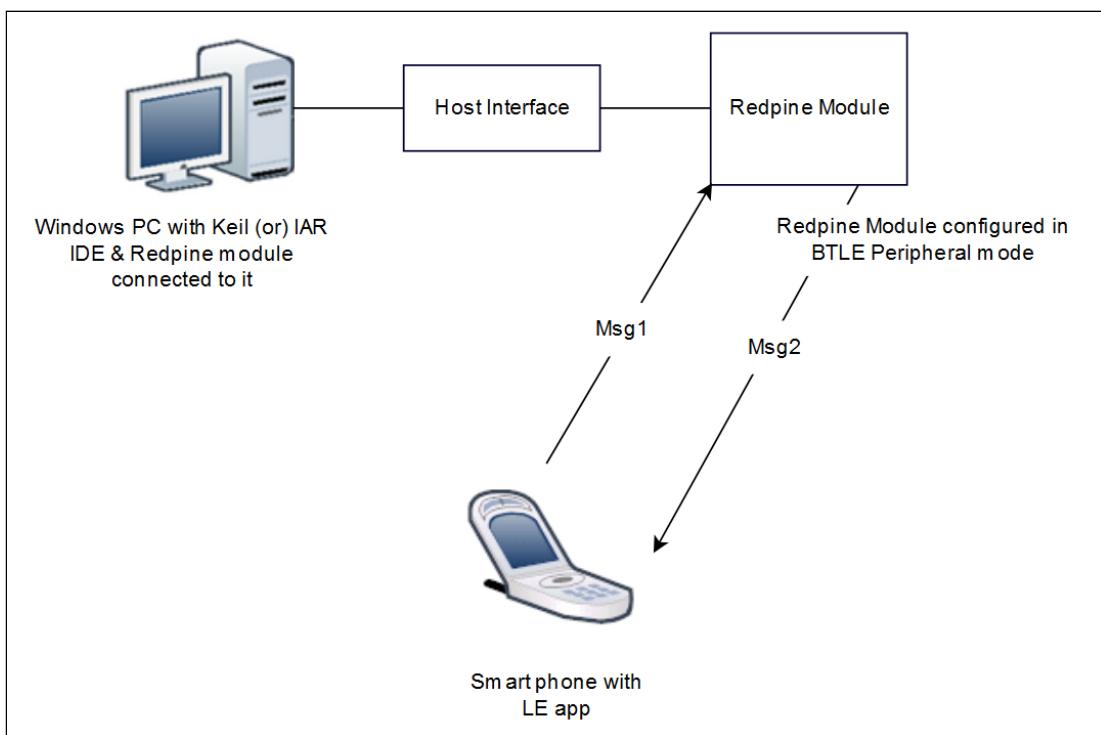
- Connect to remote device based on BD address given
- Getting primary service information
- Getting characteristic services information
- Getting descriptors information
- Enable notify based on client characteristic configuration(if remote GATT server supports notify property)
- Receive notifications from remote GATT server when value changed.

### 3.22.4 Application Setup

The Redpine module WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE supported Smart phone with GATT client in case of our module as GATT server
- BTLE supported Smart phone with GATT Glucose server in case of our module as GATT client



**Figure 1 : Setup Diagram**

**Note**

Install Light blue App for tablet for ipad mini and BLE scanner or nRF connect app user can download the BLE scanner App from the following link

<https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner&hl=en>

user can download the Light blue App from the following link

<https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer&hl=en>

user can download the nRF connect App from the following link

<https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp>

### 3.22.5 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_glucose.c** file and update/modify following macros,

**RSI\_BLE\_GLUCOSE\_SERVICE\_UUID** refers to the attribute value of the newly created service.

```
#define RSI_BLE_GLUCOSE_SERVICE_UUID 0x1808
```

**RSI\_BLE\_GLUCOSE\_MEASUREMENT\_UUID** refers to the attribute type of the first attribute under this above primary service.

**RSI\_BLE\_GLUCOSE\_MEASUREMENT\_CONTEXT\_UUID** refers to the attribute type of the second attribute under this above primary service.

**RSI\_BLE\_GLUCOSE\_FEATURE\_UUID** refers to the attribute type of the third attribute under this above primary service.

**RSI\_BLE\_RECORD\_ACCESS\_CONTROL\_POINT\_UUID** refers to the attribute type of the fourth attribute under this above primary service.

```
#define RSI_BLE_GLUCOSE_MEASUREMENT_UUID 0x2A18
#define RSI_BLE_GLUCOSE_MEASUREMENT_CONTEXT_UUID 0x2A34
#define RSI_BLE_GLUCOSE_FEATURE_UUID 0x2A51
#define RSI_BLE_RECORD_ACCESS_CONTROL_POINT_UUID 0x2A52
```

**RSI\_BLE\_APP\_GLUCOSE** refers name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_APP_GLUCOSE "GLS"
```

**GATT\_ROLE** refers the role of the Redpine module to be selected.

If user configure **SERVER**, Redpine module will act as GATT SERVER, means will add Glucose service profile.  
If user configure **CLIENT**, Redpine module will act as GATT CLIENT, means will connect to remote GATT server and get services.

```
#define GATT_ROLE SERVER
```

If user configure **CLIENT** role following macros should be configured.

**RSI\_BLE\_DEV\_ADDR\_TYPE** refers address type of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR_TYPE
LE_PUBLIC_ADDRESS
```

Valid configurations are

LE\_RANDOM\_ADDRESS  
LE\_PUBLIC\_ADDRESS

**RSI\_BLE\_DEV\_ADDR** refers address of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR "00:1A:  
7D:DA:71:13"
```

**RSI\_REMOTE\_DEVICE\_NAME** refers the name of remote device to which Redpine device has to connect

```
#define RSI_REMOTE_DEVICE_NAME  
"BLE_PERIPHERAL"
```

**Note**

User can configure either RSI\_BLE\_DEV\_ADDR or RSI\_REMOTE\_DEVICE\_NAME of the remote device.

Following are the non configurable macros related to attribute properties.

#define RSI_BLE_ATT_PROPERTY_READ	0x02
#define RSI_BLE_ATT_PROPERTY_WRITE	0x08
#define RSI_BLE_ATT_PROPERTY_WRITE_WITHOUT_RESP	0x04
#define RSI_BLE_ATT_PROPERTY_NOTIFY	0x10
#define RSI_BLE_ATT_PROPERTY_INDICATE	0x20

Following are the **non-configurable** macros in the application.

**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.  
**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

```
#define RSI_BLE_CHAR_SERV_UUID 0x2803  
#define RSI_BLE_CLIENT_CHAR_UUID 0x2902
```

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN 10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

#define RSI_BLE_PWR_INX	8
#define RSI_BLE_PWR_SAVE_OPTIONS	0
#define RSI_DUTY_CYCLING	0

**Note:**

**rsi\_wlan\_config.h** and **rsi\_ble\_config.h** files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

### Server role

1. After the program gets executed, Redpine module will be in Advertising state.
2. Open a nRFConnect App and do the scan.
3. In the App, Redpine module will appear with the name configured in the macro **RSI\_BLE\_APP\_GLUCOSE** (**Ex: "GLS"**) or sometimes observed as Redpine device as internal name "**SimpleBLEPeripheral**" (See Figure 2)
4. Initiate connection from the App.
5. After successful connection, nRFConnect displays the supported services of Redpine module.
6. Select the attribute service which is added **RSI\_BLE\_GLUCOSE\_SERVICE\_UUID** (Ex: 0x1808).
7. Enable Notify for the characteristic **RSI\_BLE\_GLUCOSE\_MEASUREMENT\_UUID** (Ex: 0x2A18). So that GATT server Notifies when value updated in that particular attribute.
8. Redpine module send the Battery Service battery level data to the attribute **RSI\_BLE\_GLUCOSE\_MEASUREMENT\_UUID** (Ex: 0x2A18) of the remote device and will Notifies the GATT client (remote device).
9. **RSI\_BLE\_GLUCOSE\_FEATURE\_UUID** will describe the value by its fields as shown in fig.
10. Figure 6 showing the use of **RSI\_BLE\_RECORD\_ACCESS\_CONTROL\_POINT\_UUID**.

Following are the snapshots of smart phone act as a client and redpine device as a server.

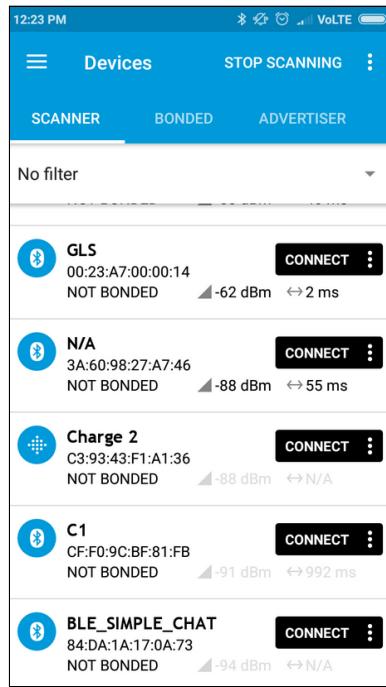


Figure 2: Scanning for GLS device & connecting to it

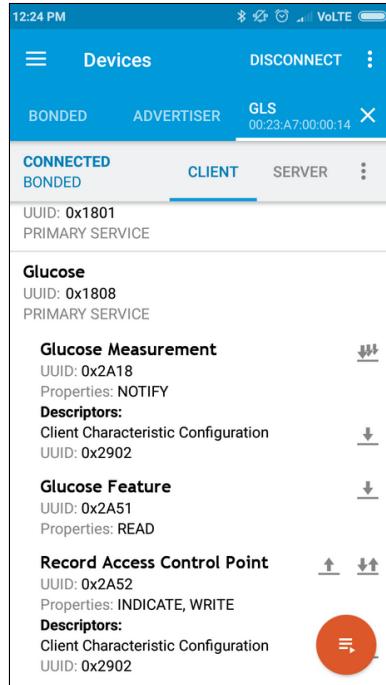
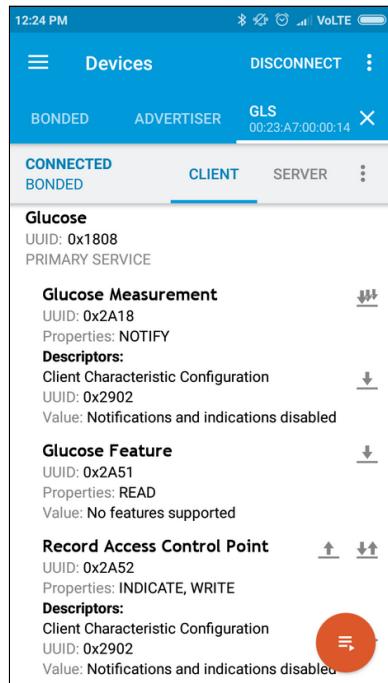
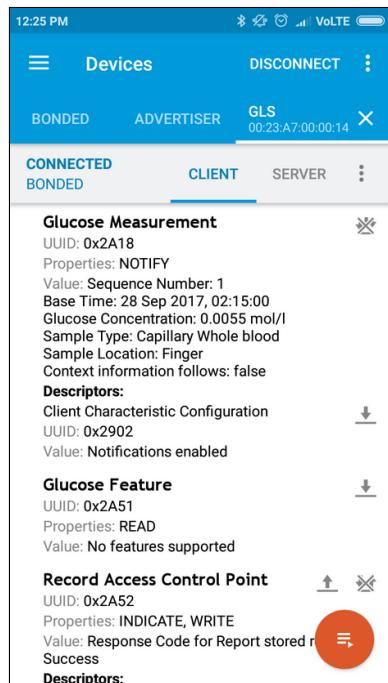


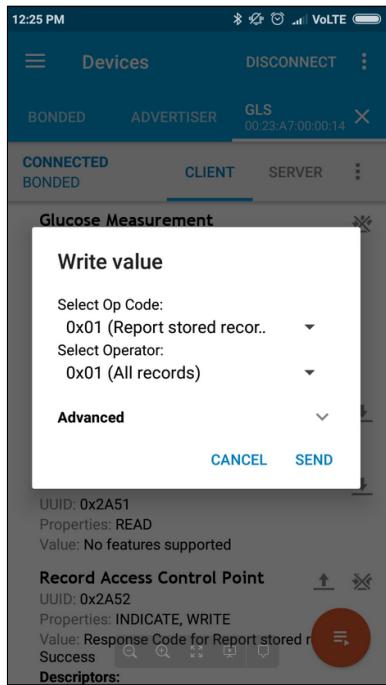
Figure 3: GLS and its characteristic discovery



**Figure 4: Client Characteristic Configuration (Indication disable)**



**Figure 5: Glucose measurement value (Indication enable)**



**Figure 6: Get records using control point characteristic**

## Client role

1. Advertise a LE device which supports Glucose Service.
2. After the program gets executed, Redpine module will connect to that remote device based on given BD address.
3. After successful connection Redpine module will read the services from the remote GATT server.
4. If remote device support notify property Redpine module will enable notify, and ready to receive notifications from remote device.
5. Whenever GATT server changes value and notifies that Redpine module will receive that value.

## 3.23 Human Interface Device Service (HIDS) Example

### 3.23.1 Definitions and Acronyms

- HIDS - Human Interface Device Service
- GATT - Generic Attribute Profile
- UUID - Universal unique identifier
- BLE - Bluetooth low energy
- BD - Bluetooth Device

### 3.23.2 Application Overview

This application demonstrates how to configure GATT server in BLE peripheral mode, how to configure GATT client in BLE central mode, explains how to do read, notify and indicate operations with GATT server from connected remote device using GATT client and explains how to get GATT information from remote GATT server in case of our module as client.

In this Application, Human Interface Device Service GATT server configures with Human Interface Device service with notification characteristic UUID. When connected remote device writes data to writable characteristic UUID,

---

Redpine device receives the data which is received on writable characteristic UUID and writes the same data to readable characteristic UUID and sends notifications to the connected device (or) remote device can read the same data using read characteristic UUID if notification enabled on client side.

Human Interface Device Service GATT client will get Human Interface Device service (primary service) , Report Map (characteristic service), and descriptors(client characteristic configuration and report reffrence) information from the remote GATT server. If remote device supports notify, our module will enable notify property and will be notified by the remote GATT server when value changed.

### 3.23.3 Sequence of Events

#### Server Role

This mode explains user how to:

- Create Human Interface Device service
- Make the device to advertise
- Connect from remote BTLE device
- Receive the message from the connected peer/Smart phone
- Give the notifications to connected device.

#### Client Role

This mode explains user how to:

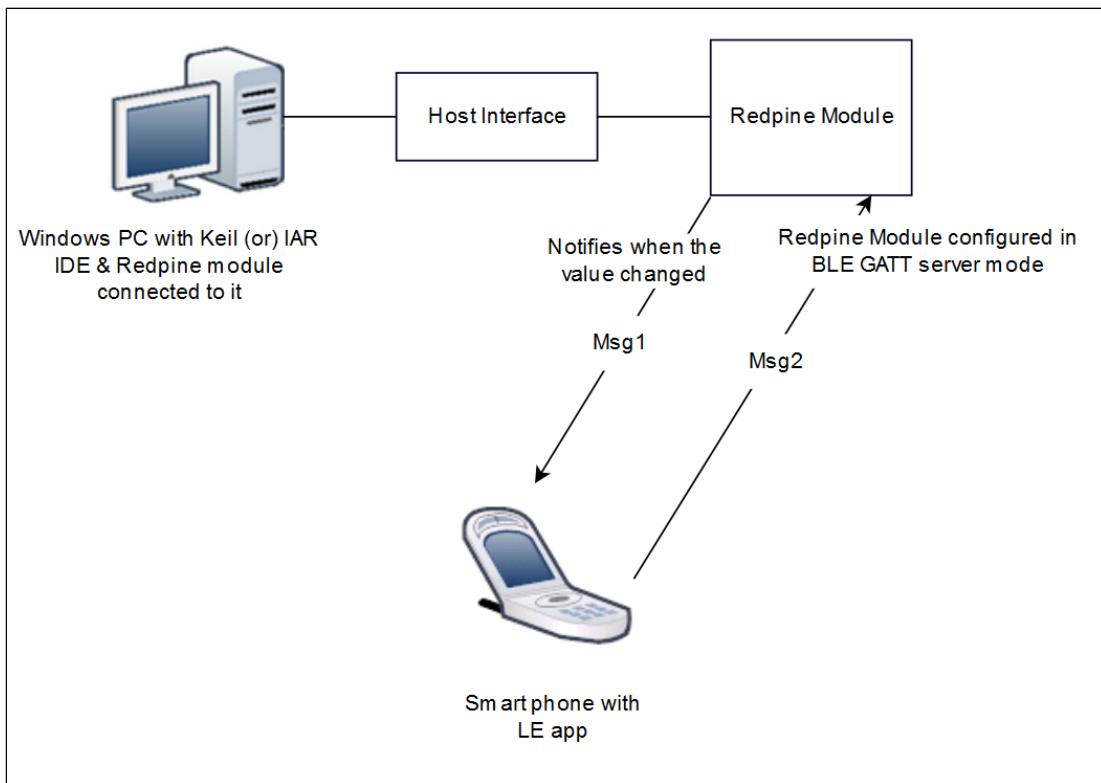
- Connect to remote device based on BD address given
- Getting primary service information
- Getting characteristic services information
- Getting descriptors information
- Enable notify based on client characteristic configuration(if remote GATT server supports notify property)
- Receive notifications from remote GATT server when value changed.

### 3.23.4 Application Setup

The Redpine module WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### 3.23.5 WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE supported Smart phone with GATT client in case of our module as GATT server
- BTLE supported Smart phone with GATT Human Interface Device server in case of our module as GATT client



**Figure 1: Setup Diagram**

**Note**

Use default Bluetooth application in smart phones which has BLE support.

### 3.23.6 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_hids.c** file and update/modify following macros.

**RSI\_BLE\_HID\_SERVICE\_UUID** refers to the attribute value of the newly created service.

```
#define RSI_BLE_HID_SERVICE_UUID 0x1812
```

**RSI\_BLE\_HID\_PROTOCOL\_MODE\_UUID** refers to the attribute type of the first attribute under this above primary service.

**RSI\_BLE\_HID\_REPORT\_UUID** refers to the attribute type of the second attribute under this above primary service.

**RSI\_BLE\_HID\_REPORT\_MAP\_UUID** refers to the attribute type of the third attribute under this above primary service.

**RSI\_BLE\_HID\_INFO\_UUID** refers to the attribute type of the fourth attribute under this above primary

service.

**RSI\_BLE\_HID\_CONTROL\_POINT\_UUID** refers to the attribute type of the fifth attribute under this above primary service.

#define RSI_BLE_HID_PROTOCOL_MODE_UUID	0x2A4E
#define RSI_BLE_HID_REPORT_UUID	0x2A4D
#define RSI_BLE_HID_REPORT_MAP_UUID	0x2A4B
#define RSI_BLE_HID_INFO_UUID	0x2A4A
#define RSI_BLE_HID_CONTROL_POINT_UUID	0x2A4C

**RSI\_BLE\_APP\_HIDS** refers name of the Redpine device to appear during scanning by remote devices.

#define RSI_BLE_APP_HIDS	"HIDS"
--------------------------	--------

**GATT\_ROLE** refers the role of the Redpine module to be selected.

If user configure **SERVER**, Redpine module will act as GATT SERVER, means will add Human Interface Device service profile.

If user configure **CLIENT**, Redpine module will act as GATT CLIENT, means will connect to remote GATT server and get services.

#define GATT_ROLE	SERVER
-------------------	--------

Valid configurations are SERVER and CLIENT.

If user configure **CLIENT** role following macros should be configured.

**RSI\_BLE\_REMOTE\_BD\_ADDRESS\_TYPE** refers address type of the remote device to connect.

#define RSI_BLE_REMOTE_BD_ADDRESS_TYPE	
RANDOM_ADDRESS	

Valid configurations are RANDOM\_ADDRESS and PUBLIC\_ADDRESS.

**RSI\_BLE\_REMOTE\_BD\_ADDRESS** refers address of the remote device to connect. Replace this with valid BD address.

#define RSI_BLE_REMOTE_BD_ADDRESS	"00:1A:
7D:DA:71:13"	

**RSI\_REMOTE\_DEVICE\_NAME** refers the name of remote device to which Redpine device has to connect

#define RSI_REMOTE_DEVICE_NAME	
"Redpine_Device"	

**Note**

User can configure either RSI\_BLE\_DEV\_ADDR or RSI\_REMOTE\_DEVICE\_NAME of the remote device.

Following are the non configurable macros related to attribute properties.

#define RSI_BLE_ATT_PROP_RD	0x02
#define RSI_BLE_ATT_PROP_WR_NO_RESP	0x04
#define RSI_BLE_ATT_PROP_WR	0x08
#define RSI_BLE_ATT_PROP_NOTIFY	0x10
#define RSI_BLE_ATT_PROP_INDICATE	0x20

Following are the **non-configurable** macros in the application.

**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.  
**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

**RSI\_BLE\_REPORT\_REFERENCE\_UUID** refers to the attribute type of the report reference descriptor to be added in a service.

#define RSI_BLE_CHAR_SERV_UUID	0x2803
#define RSI_BLE_CLIENT_CHAR_UUID	0x2902
#define RSI_BLE_REPORT_REFERENCE_UUID	0x2908

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

#define BT_GLOBAL_BUFF_LEN	10000
----------------------------	-------

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros, #define RSI\_BLE\_PWR\_INX 8

```
#define RSI_BLE_PWR_SAVE_OPTIONS 0
#define RSI_DUTY_CYCLING 0
```

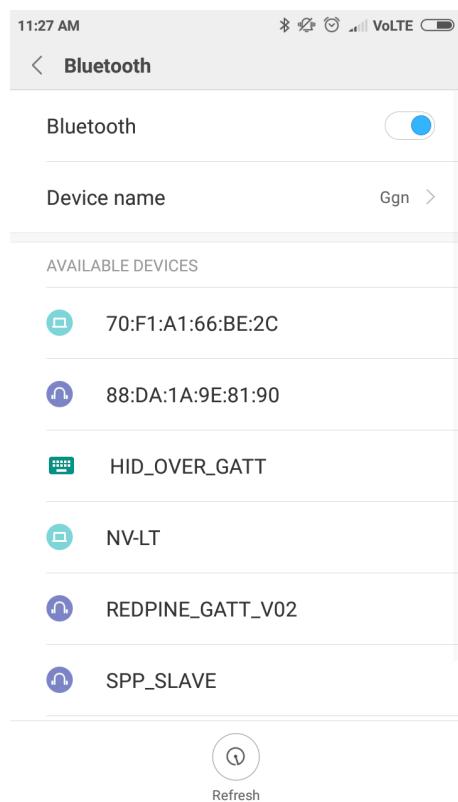
**Note**

rsi\_wlan\_config.h and rsi\_ble\_config.h files are already set with desired configuration in respective example folders user need not change for each example.

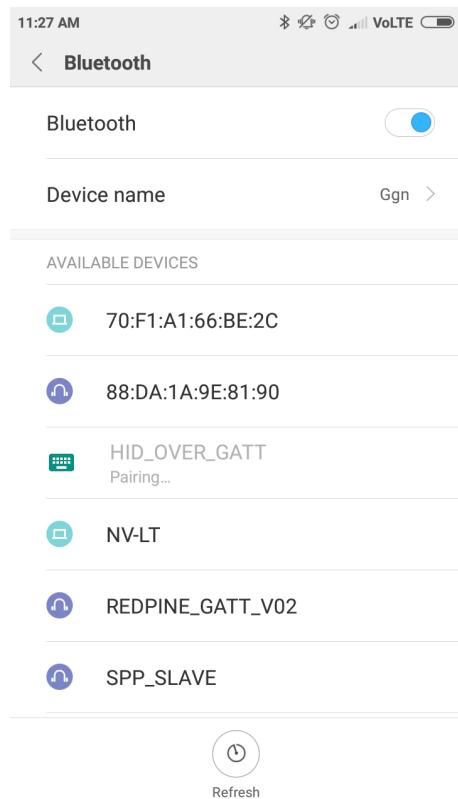
## Executing the Application

### Server role

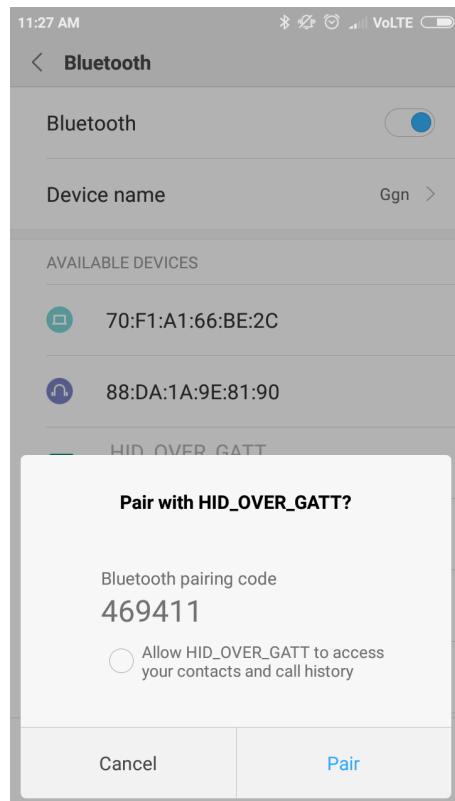
1. After the program gets executed, Redpine module will be in Advertising state.
2. Open a default Bluetooth App and do the scan.
3. In the App, Redpine module will appear with the name configured in the macro **RSI\_BLE\_APP\_HIDS (Ex: "HIDS")** or sometimes observed as Redpine device as internal name "**SimpleBLEPeripheral**".
4. Initiate connection from the App and complete the pairing process.
5. After successful connection, open note pad or any text editor in phone, you can see some text printing.
6. By default application is sending some text (i.e. "hog ") in regular interval, which will come as a notification to smart phone.
7. While connection, smart phone will do service discovery and it will find the HID service with UUID **RSI\_BLE\_HID\_SERVICE\_UUID**. After that it will read report map and enables the notification.
8. Following are the screen shots of smart phone to test HID over GATT application.



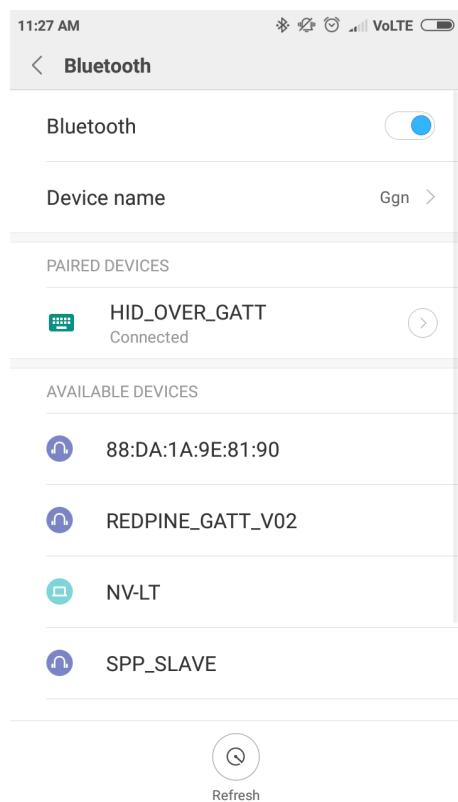
**Figure 1: Scanning for HID\_OVER\_GATT device**



**Figure 2: Connect to HID\_OVER\_GATT device**



**Figure 3: Pair with HID\_OVER\_GATT device**



**Figure 4:HID\_OVER\_GATT device Connected**



**Figure 5: Receiving data from HID\_OVER\_GATT device**

### Client role

1. Advertise a LE device which supports Human Interface Device Service.
2. After the program gets executed, Redpine module will connect to that remote device based on given BD address.
3. After successful connection Redpine module will read the services from the remote GATT server.
4. If remote device support notify property Redpine module will enable notify, and ready to receive notifications from remote device.
5. Whenever GATT server changes value and notifies that Redpine module will receive that value.

## 3.24 BLE Whitelist Example

### 3.24.1 Application Overview

#### Overview

This application is used to add a particular BD-Address to the White List. The device to connect is saved on the white list located in the LL block of the controller. This enumerates the remote devices that are allowed to communicate with the local device. The White List can restrict which device are allowed to connect to other device. If is not, is not going to connect. Once the address was saved, the connection with that device is going to be an auto connection establishment procedure. This means that the Controller autonomously establishes a connection with the device address that matches the address stored in the While List.

## Sequence of Events

This Application explains user how to:

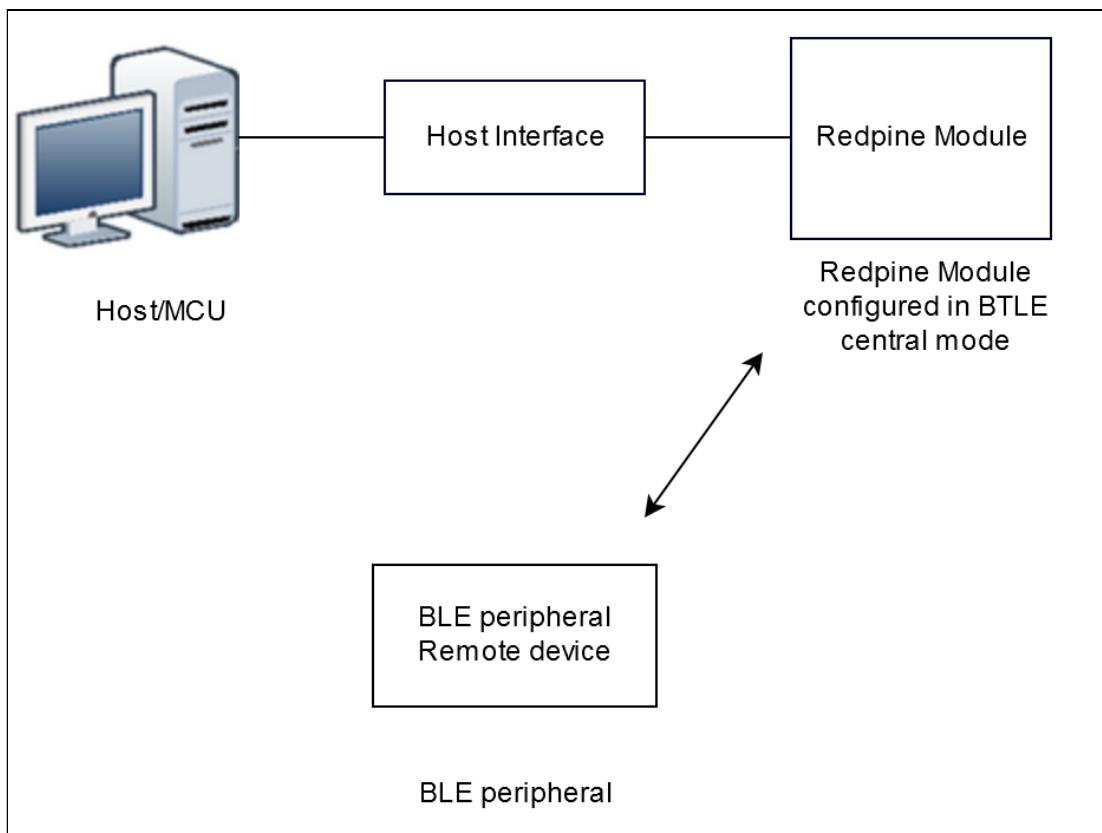
- Adding BD Address to the whitelist
- Scan remote devices
- Connect to the Whitelisted remote device

### 3.24.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE peripheral device



**Figure 1: Setup Diagram**

### 3.24.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ble\_whitelist.c** file and update/modify following macros,

**RSI\_BLE\_DEV\_ADDR\_TYPE** refers address type of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR_TYPE  
LE_PUBLIC_ADDRESS
```

Based on address type of remote device, valid configurations are  
**LE\_RANDOM\_ADDRESS**  
**LE\_PUBLIC\_ADDRESS**

**RSI\_BLE\_DEV\_ADDR** refers address of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR  
"00:1A:7D:DA:71:48"
```

**RSI\_BLE\_WHITELIST\_DEV\_ADDR1\_TYPE, RSI\_BLE\_WHITELIST\_DEV\_ADDR1\_TYPE** refers address of the remote devices to be whitelisted

```
#define RSI_BLE_WHITELIST_DEV_ADDR1_TYPE  
LE_PUBLIC_ADDRESS  
#define RSI_BLE_WHITELIST_DEV_ADDR2_TYPE  
LE_PUBLIC_ADDRESS
```

**RSI\_BLE\_WHITELIST\_DEV\_ADDR1, RSI\_BLE\_WHITELIST\_DEV\_ADDR2** refers address of the whitelisted remote devices to connect.

```
#define RSI_BLE_WHITELIST_DEV_ADDR1  
"00:1A:7D:DA:71:48"  
#define RSI_BLE_WHITELIST_DEV_ADDR2  
"00:23:A7:80:70:B9"
```

**RSI\_REMOTE\_DEVICE\_NAME** refers the name of remote device to which Redpine device has to connect

```
#define RSI_REMOTE_DEVICE_NAME  
"REDPINE_DEV"
```

**Note:**

user can configure either **RSI\_BLE\_DEV\_ADDR** or **RSI\_REMOTE\_DEVICE\_NAME** of the remote device.  
Following are the event numbers for advertising,connection and Disconnection events,

#define RSI_APP_EVENT_ADV_REPORT	0
#define RSI_APP_EVENT_CONNECTED	1
#define RSI_APP_EVENT_DISCONNECTED	2

Following are the non-configurable macros in the application.

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

#define BT_GLOBAL_BUFF_LEN	10000
----------------------------	-------

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

```
#define RSI_BLE_PWR_INX 8
#define RSI_BLE_PWR_SAVE_OPTIONS 0
#define RSI_DUTY_CYCLING 0
#define RSI_BLE_SCAN_FILTER_TYPE
SCAN_FILTER_TYPE_ONLY_WHITE_LIST
```

**Note**

rsi\_wlan\_config.h and rsi\_ble\_config.h files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the remote ble device in peripheral mode and put it in advertising mode.
2. After the program gets executed, it adds the configured remote device addresses to the whitelist, and Redpine device tries to connect only with the whitelisted remote device specified in **RSI\_BLE\_DEV\_ADDR** or **RSI\_REMOTE\_DEVICE\_NAME** macro &

- 
3. Observe that the connection is established between the desired device and Redpine device.

**Note**

Examples for ble peripherals : Blue tooth Dongle, mobile application, TA sensor tag

## 3.25 BLE Dual Role Example

### 3.25.1 Application Overview

#### Overview

This application demonstrates how to connect with multiple(6) slaves as redpine module in central mode and connect with multiple(2) masters as redpine module in peripheral mode.

#### Sequence of Events

This Application explains user how to:

- Connect with remote BTLE peripheral devices.
- Connect with remote BTLE central devices.

### 3.25.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- BTLE peripheral devices
- BTLE central devices.

### 3.25.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open ***rsi\_ble\_central.c*** file and update/modify following macros,  
**RSI\_BLE\_LOCAL\_NAME** refers the name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_LOCAL_NAME  
"WYZBEE_PERIPHERAL"
```

**RSI\_BLE\_DEV\_ADDR\_TYPE** refers address type of the remote device to connect.

```
#define RSI_BLE_DEV_ADDR_TYPE  
LE_PUBLIC_ADDRESS
```

Based on address type of remote device, valid configurations are  
LE\_RANDOM\_ADDRESS  
LE\_PUBLIC\_ADDRESS

**RSI\_BLE\_DEV\_ADDR** refers address of the remote device to connect.

#define RSI_BLE_DEV_1_ADDR 07:2C:F0"	"00:1B:DC:
#define RSI_BLE_DEV_2_ADDR 7D:DA:71:73"	"00:1A:
#define RSI_BLE_DEV_3_ADDR 7D:DA:71:44"	"00:1A:
#define RSI_BLE_DEV_4_ADDR 34:54:66"	"00:1A:7D:
#define RSI_BLE_DEV_5_ADDR 7D:DA:71:48"	"00:1A:
#define RSI_BLE_DEV_6_ADDR 7D:DA:72:13"	"00:1A:

Following are the event numbers for advertising, connection, Disconnection events and scan restart events.

#define RSI_APP_EVENT_ADV_REPORT	0
#define RSI_APP_EVENT_CONNECTED	1
#define RSI_APP_EVENT_DISCONNECTED	2
#define RSI_BLE_SCAN_RESTART_EVENT	3

Following are the non-configurable macros in the application.

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN 10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

3. Open **rsi\_ble\_config.h** file and update/modify following macros,

#define RSI_BLE_PWR_INX	8
#define RSI_BLE_PWR_SAVE_OPTIONS	0
#define RSI_DUTY_CYCLING	0

To configure the Nbr of master and No of slaves to be connected

#define RSI_BLE_MAX_NBR_SLAVES	3
#define RSI_BLE_MAX_NBR_MASTERS	1

**Note**

rsi\_wlan\_config.h and rsi\_ble\_config.h files are already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the remote ble device in peripheral mode and put it in advertising mode.
2. After the program gets executed, Redpine device tries to connect with the remote device address specified in the Macros
  - a. **example: RSI\_BLE\_DEV\_1\_ADDR**
3. Redpine device also in advertising mode, connect from the remote BLE Central device.
4. Observe that the connection is established between the desired device and Redpine device.

**Note**

Maximum we can connect with 2 Remote BLE Centrals.

**Note**

Examples for ble peripherals : Blue tooth Dongle, mobile application, TA sensor tag

## 4 Security APIs

### 4.1 AES Application

#### 4.1.1 Example

##### Overview

The Advanced Encryption Standard (AES), also known by its original name Rijndael is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.

For AES, NIST selected three members of the Rijndael family, each with a block size of 128 bits, but three different key lengths: 128, 192 and 256 bits.

AES has been adopted by the U.S. government and is now used worldwide. The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data.

This application demonstrates how to Encrypt and decrypt data using AES in WiSeConnect.

##### Sequence of Events

This application explains user the following:

- AES CBC mode encryption(256bit key)
- AES CBC mode decryption(256bit key)
- AES ECB mode encryption(256bit key)
- AES ECB mode decryption(256bit key)
- AES CTR mode encryption(256bit key)
- AES CTR mode decryption(256bit key)

#### 4.1.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### 4.1.3 WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module

#### 4.1.4 Configuration and Execution of the Application

##### Configuring the Application

Edit the **rsi\_aes\_app.c** file

- From given configuration, the following fields can be edited
  - “**msg**” refers to plain data which is fed to AES engine for encryption with all sizes of key.
  - “**key**” refers to key which is in turn used for encryption/decryption with AES engine.
  - “**iv**” refers to IV used in AES-CBC/AES-CTR mode.
- Depending on key size/mode of encryption/decryption provided input data is processed using AES engine.

## Executing the Application

1. Configure the **key**, **msg**, **iv** in the file **rsi\_aes\_app.c**
2. Build and launch the application.
3. After the program gets executed, Redpine device's AES Encryption / Decryption operations are performed by providing key to AES engine.

## 4.2 ECDH Application

### 4.2.1 Example

#### Overview

Elliptic-curve Diffie–Hellman (ECDH) is an anonymous key agreement protocol that allows two parties, each having an elliptic-curve public–private key pair, to establish a shared secret over an insecure channel. This shared secret may be directly used as a key, or to derive another key. The key, or the derived key, can then be used to encrypt subsequent communications using a symmetric-key cipher. It is a variant of the Diffie–Hellman protocol using elliptic-curve cryptography.

#### Sequence of Events

This Application demonstrates the various ECDH operations like

- ECDH point addition
- ECDH point subtraction
- ECDH point multiplication
- ECDH point double
- ECDH affinify

### 4.2.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module

### 4.2.3 Configuration and Execution of the Application

#### Configuring the Application

Edit the **rsi\_ecdh\_app.c** file :

- From given configuration,  
Input vectors sz, sy, sz and tx, ty, tz along with the scalar multiplier d are given to the ECDH.
- Depending on ECDH mode and ECDH operation respective outputs are computed for the given input data.

## Executing the Application

1. Connect Redpine device to the Windows PC with Keil or IAR IDE in running state.
2. Configure the **ECDH** in the file **rsi\_ecdh\_app.c**.
3. Build and launch the application.
4. After the program gets executed, Redpine Device generates the corresponding output for the given input data.

## 4.3 Exponentiation Application

### 4.3.1 Example

#### Overview

Diffie-Hellman key agreement protocol (also called exponential key agreement) was developed by Diffie and Hellman [DH76] in 1976. The protocol allows two users to exchange a secret key over an insecure medium without any prior secrets.

The protocol has two system parameters p and g. They both are public and may be used by all the users in a system. Parameter p is a prime number and parameter g (usually called a generator) is an integer less than p, with the following property: for every number n between 1 and p-1 inclusive, there is a power k of g such that  $n = g^k \pmod{p}$ .

Suppose Alice and Bob want to agree on a shared secret key using the Diffie-Hellman key agreement protocol. They proceed as follows:

- First, Alice generates a random private value "**a**" and Bob generates a random private value "**b**". Both a and b are drawn from the set of integers.
- They then derive their public values using parameters p and g and their private values. Alice's public value is  $g^a \pmod{p}$  and Bob's public value is  $g^b \pmod{p}$ . When finished, they exchange their public values.
- Finally, Alice computes  $g^{ab} = (g^b)^a \pmod{p}$ , and Bob computes  $g^{ab} = (g^a)^b \pmod{p}$ . Since  $g^{ab} = g^{ba} = k$ , Alice and Bob now have a shared secret key '**k**'.
- The desired operation to be performed for generating the key is  $X^E \pmod{P}$ . (analogous to  $g^{ab} \pmod{p}$  where E is the Exponent, X is the generator and P is the prime).

#### Sequence of Events

This application demonstrates

- Diffie-Hellman algorithm to compute exponentiation value or shared secret key for the given input data.

### 4.3.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module

### 4.3.3 Configuration and Execution of the Application

#### Configuring the Application

Edit the following fields in **rsi\_exp\_app.c** file.

- From given configuration,  
“**prime**” refers to the prime value data which is given as input to DH for computing exponentiation value.  
“**exp**” refers to the exponent value which is given as input to DH for computing exponentiation value.  
“**base**” refers to the base value which is given as input to DH for computing exponentiation value.
- Depending on given input data prime, exponent and base values corresponding DH key is generated.

#### Executing the Application

1. Connect Redpine device to the Windows PC with Keil or IAR IDE in running state.
2. Configure the **DH** in the file **rsi\_exp\_app.c**.
3. Build and launch the application.
4. After the program gets executed, Redpine device generates DH key for the given input data.

### 4.4 HMAC SHA Application

#### 4.4.1 Example

##### Overview

In cryptography, a keyed-hash message authentication code (HMAC) is a specific type of message authentication code (MAC) involving a cryptographic hash function and a secret cryptographic key. It may be used to simultaneously verify both the data integrity and the authentication of a message, as with any MAC.

The cryptographic strength of the HMAC depends upon the cryptographic strength of the underlying hash function, the size of its hash output, and on the size and quality of the key.

HMAC generation uses two passes of hash computation. The secret key is first used to derive two keys - inner and outer. The first pass of the algorithm produces an internal hash derived from the message and the inner key. The second pass produces the final HMAC code derived from the inner hash result and the outer key. Thus the algorithm provides better immunity against Length extension attacks.

HMAC does not encrypt the message. Instead, the message (encrypted or not) must be sent alongside the HMAC hash. Parties with the secret key will hash the message again themselves, and if it is authentic, the received and computed hashes will match.

The configuration application demonstrates how to compute digest using HMAC SHA. This application computes a digest of 64 bytes using HMAC SHA.

##### Sequence of Events

The application demonstrates:

- Computation of 64 bytes digest for the given input data using HMAC SHA.

#### 4.4.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

## WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module

### 4.4.3 Configuration and Execution of the Application

#### Configuring the Application

Edit the following fields in **rsi\_hmac\_sha\_app.c** file in the following path:

- From given configuration,  
“**msg**” refers to data which is given as input to HMAC SHA for computing digest.  
“**key**” refers to the key input for HMAC SHA.

```
#define HMAC_BUFF_LEN 4000
```

Length of the HMAC SHA buffer. It depends on the input message length and key length. It should be greater than the addition of message length and key length.

- Depending on HMAC SHA mode, digest of respective size is computed for the given input data.

#### Executing the Application

1. Connect Redpine device to the Windows PC with Keil or IAR IDE in running state.
2. Configure the **HMAC SHA** in the file **rsi\_hmac\_sha\_app.c**
3. Build and launch the application.
4. After the program gets executed, Redpine Device generates the digest of 64 bytes for the given input data.

## 4.5 PUF AES Example

### 4.5.1 Example

#### Overview

PUF, acronym for Physical Unclonable Functions, is a technology provides a secure method for storing a key, withstanding today's attack and even protecting against future potential attacks. Physical Unclonable Functions (PUFs) are defined as functions based on physical characteristics which are unique for each chip, difficult to predict, easy to evaluate and reliable. These functions should also be individual and practically impossible to duplicate and cannot be reverse engineered.

The configuration application demonstrates how to Enroll PUF using WiSeConnect.

#### Sequence of Events

This Application explains user how to:

- Start PUF
- AES CBC mode encryption(128bit key) operation using PUF

- AES CBC mode decryption(128bit key) operation using PUF
- AES CBC mode encryption(256bit key) operation using PUF
- AES CBC mode decryption(256bit key) operation using PUF
- AES ECB mode encryption(128bit key) operation using PUF
- AES ECB mode decryption(128bit key) operation using PUF
- AES ECB mode encryption(256bit key) operation using PUF
- AES ECB mode decryption(256bit key) operation using PUF
- AES MAC generation(128bit key)
- AES MAC generation(256bit key)

#### 4.5.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module

#### 4.5.3 Configuration and Execution of the Application

##### Configuring the Application

Edit the following fields in **rsi\_puf\_aes\_app.c** file

From the given configuration,

**RSI\_AES\_PLAIN\_TXT** refers to the plain data which is fed to AES engine for encryption with all sizes of key.  
**RSI\_AES\_KEY** refers to the key which is used for set\_key operation, which is in turn used for encryption/decryption with AES engine.

**RSI\_AES\_CBC\_IV** refers to IV used in AES-CBC mode.

#define RSI_AES_PLAIN_TXT	"data to be encrypted"
#define RSI_AES_KEY	"Key to encrypt the
data"	
#define RSI_AES_CBC_IV	"IV input"

Depending on key size/mode of encryption/decryption provided input data is used to perform operation using PUF and AES engine.

Update the WLAN configuration file:

**rsi\_wlan\_config.h**

RSI_CUSTOM_FEATURE_BIT_MAP	FEAT_CUSTOM_FEAT_EXTENSION_VALID
RSI_EXT_CUSTOM_FEATURE_BIT_MAP	EXT_FEAT_PUF

## Executing the Application

1. Connect WiSeConnect device to the Windows PC running Keil IDE.
2. Configure the macros in the files:
  - **rsi\_puf\_aes\_app.c**
  - **rsi\_wlan\_config.h**
3. Build and launch the application.
4. After the program gets executed, WiSeConnect Device's PUF will be Started, AES encryption/Decryption are performed by providing key to aes engine.

## 4.6 PUF Configuration Example

### 4.6.1 Example

#### Overview

PUF, acronym for Physical Unclonable Functions, is a technology which provides a secure method for storing a key, withstanding today's attack and even protecting against future potential attacks. Physical Unclonable Functions (PUFs) are defined as functions based on physical characteristics which are unique for each chip, difficult to predict, easy to evaluate and reliable. These functions should also be individual and practically impossible to duplicate and cannot be reverse engineered.

#### Sequence of Events

This Application explains user how to:

- Disable or enable PUF set key feature
- Disable or enable PUF get key feature

### 4.6.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module

### 4.6.3 Configuration and Execution of the Application

#### Configuring the Application

Edit the **rsi\_puf\_config\_app.c** file in the following path.

From the given configuration,

**RSI\_PUF\_SET\_KEY\_DIS** This macro if enabled, disables further set\_key operations on PUF  
**RSI\_PUF\_GET\_KEY\_DIS** This macro if enabled, disables further get\_key operations on PUF

Update the Wlan configuration file:  
**rsi\_wlan\_config.h**

RSI_CUSTOM_FEATURE_BIT_MAP	FEAT_CUSTOM_FEAT_EXTENSION_VALID
RSI_EXT_CUSTOM_FEATURE_BIT_MAP	EXT_FEAT_PUF

Executing the Application:

1. Connect WiSeConnect device to the Windows PC running Keil IDE.
2. Configure the macros in the files located at
  - **rsi\_puf\_config\_app.c**
  - **rsi\_wlan\_config.h**
3. Build and launch the application.
4. Build and launch the application.

## 4.7 PUF Demo Example

### 4.7.1 Example

#### Overview

PUF, acronym for Physical Unclonable Functions, is a technology which provides a secure method for storing a key to withstanding today's attack and also to protect against future potential attacks. Physical Unclonable Functions (PUFs) are defined as functions based on physical characteristics which are unique for each chip, difficult to predict, easy to evaluate and reliable in nature. These functions should also be individual and practically impossible to duplicate and cannot be reverse engineered.

This application demonstrates how we can use PUF feature for enhancing the security.

#### Sequence of Events

This Application explains user how to:

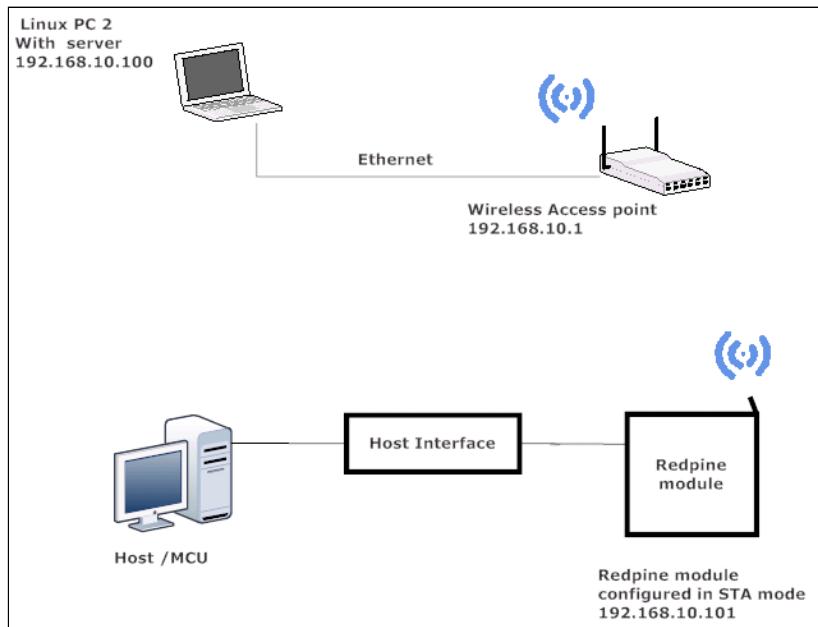
1. Connect the module to the server. After connection, the signature, intrinsic keycode and the keycode of the key is shared to the user.
2. Wait for the data from the user. After receiving the data, validate whether the user is a pre authorized one or not. If the user is an authorized one, perform the operation.
3. Based upon the message sent, (for e.g. user sent a message for locking the door as "LOCK THE DOOR") it will perform the operation.
4. If the user is an unauthorized person, do not perform the operation.

### 4.7.2 Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization. The following sections are applicable to WiSeConnect parts only.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with Keil or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine Module
- WiFi Access point
- Linux PC
- TCP server running in Linux PC



**Figure 1: Setup Diagram**

#### 4.7.3 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_puf\_demo.c** file and update / modify following macros:  
**PUF\_ENROLL** to enroll the puf.

```
#define PUF_ENROLL
```

0

**Note:** If user wants to enroll the puf then set **PUF\_ENROLL** to 1 . After enrollment is done user has to make this **PUF\_ENROLL** as 0 and run the application again. This is required only once for a module.  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode

**RSI\_WPA** - For WPA security mode

**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE    RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK             "<psk>"
```

**DEVICE\_PORT** port refers client port number

```
#define DEVICE_PORT     5001
```

**SERVER\_PORT** port refers remote server port number which is opened in PC2

```
#define SERVER_PORT      5003
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with TCP server socket.

IP address should be in long format and in little endian byte order.

**Example:** To configure "192.168.0.100" as remote IP address, update the macro **SERVER\_IP\_ADDRESS** as **0x6400A8C0**.

```
#define SERVER_IP_ADDRESS 0x6400A8C0
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN   8000
```

**To configure IP address**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE
```

1

**Note:** If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP
```

0X0A0AA8C0

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY
```

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK
```

0x00FFFFFF

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

<pre>#define CONCURRENT_MODE</pre>	RSI_DISABLE
<pre>#define RSI_FEATURE_BIT_MAP</pre>	
<pre>FEAT_SECURITY_OPEN</pre>	
<pre>#define RSI_TCP_IP_BYPASS</pre>	RSI_DISABLE
<pre>#define RSI_TCP_IP_FEATURE_BIT_MAP</pre>	
<pre>TCP_IP_FEAT_DHCPV4_CLIENT</pre>	
<pre>#define RSI_CUSTOM_FEATURE_BIT_MAP</pre>	
<pre>FEAT_CUSTOM_FEAT_EXTENSION_VALID</pre>	
<pre>#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP</pre>	EXT_FEAT_PUF
<pre>#define RSI_BAND</pre>	RSI_BAND_2P4GHZ

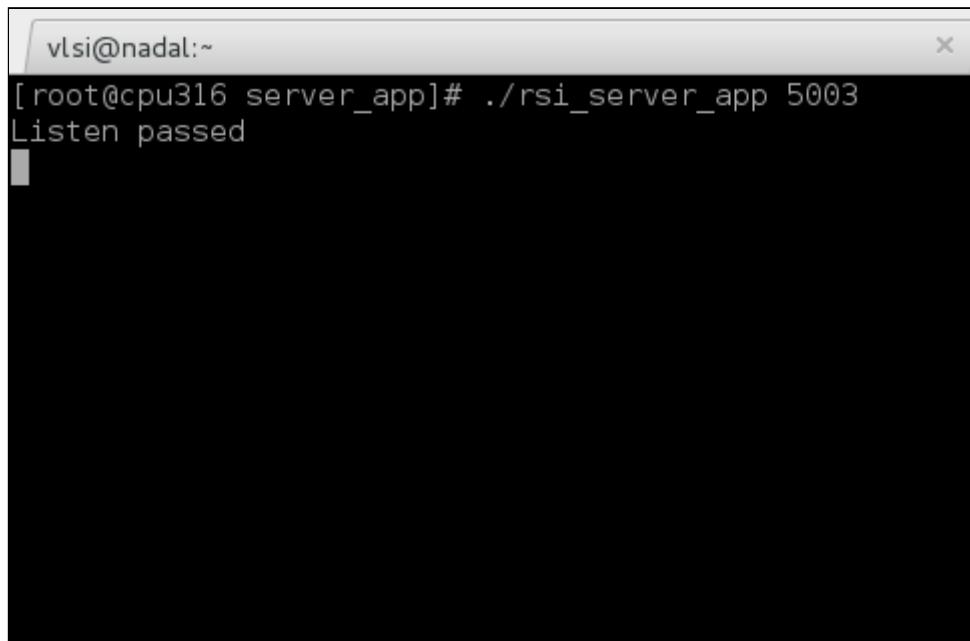
## Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect WiSeConnect device in STA mode.

Compile and run "**server\_demo**" by providing port number.

make clean; make

./rsi\_server\_app 5003



A terminal window with a black background and white text. The window title is "vlsi@nadal:~". The command entered is "[root@cpu316 server\_app]# ./rsi\_server\_app 5003". The output shows "Listen passed" followed by a blank line.

2. After the program gets executed, WiSeConnect Device would be connected to Access point having the configuration same that of in the application and get IP.

3. The Device which is configured as client will connect to remote server and sends puf keycodes along with signature data to the server.

4. Module receives the data from server and if the remote device is authorized, performs the operation(LOCK/UNLOCK) given by the server.

```
venkateshl@cpu316:/work/venkateshl/... venkateshl@cpu316:/work/venkateshl/... venkateshl@cpu316:/work/venkateshl/... [root@cpu316 server_app]# ./rsi_server_app 5003 Listen passed Connect Req from 160.167.181.191 accepted Enter the option:- 1)Lock 2)Unlock:1 Door locked Enter the option:- 1)Lock 2)Unlock:2 Door unlocked Enter the option:- 1)Lock 2)Unlock:1 Door locked Enter the option:- 1)Lock 2)Unlock:1 Door already locked please enter correct option! Enter the option:- 1)Lock 2)Unlock:2 Door unlocked Enter the option:- 1)Lock 2)Unlock:1 Door already unlocked please enter correct option! Enter the option:- 1)Lock 2)Unlock:1 Door locked Enter the option:- 1)Lock 2)Unlock:1
```

## 4.8 PUF Key AES Example

### 4.8.1 Example

## Overview

PUF, acronym for Physical Unclonable Functions, is a technology provides a secure method for storing a key, withstanding today's attack and even protecting against future potential attacks. Physical Unclonable Functions (PUFs) are defined as functions based on physical characteristics which are unique for each chip, difficult to predict, easy to evaluate and reliable. These functions should also be individual and practically impossible to duplicate and cannot be reverse engineered.

## Sequence of Events

This Application explains user how to:

- Start PUF

- Set\_key(128bit key) operation on PUF
- Get\_key(128bit key) operation on PUF
- Load\_key(128bit key) operation on PUF
- AES ECB mode encryption(128bit key) operation using PUF
- AES ECB mode decryption(128bit key) operation using PUF
- AES CBC mode encryption(128bit key) operation using PUF
- AES CBC mode decryption(128bit key) operation using PUF
- Set\_key(256bit key) operation on PUF
- Get\_key(256bit key) operation on PUF
- Load\_key(256bit key) operation on PUF
- AES ECB mode encryption(256bit key) operation using PUF
- AES ECB mode decryption(256bit key) operation using PUF
- AES CBC mode encryption(256bit key) operation using PUF
- AES CBC mode decryption(256bit key) operation using PUF

#### 4.8.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module

#### 4.8.3 Configuration and Execution of the Application

##### Configuring the Application

###### Edit the **rsi\_puf\_key\_aes\_app.c** file

1. From given configuration,  
**RSI\_AES\_PLAIN\_TXT** refers to plain data which is fed to AES engine for encryption with all sizes of key.  
**RSI\_AES\_KEY** refers to key which is used for set\_key operation, which is in turn used for encryption/decryption with AES engine.  
**RSI\_AES\_CBC\_IV** refers to IV used in AES-CBC mode.

```
#define RSI_AES_PLAIN_TXT           "data to be  
encrypted"  
#define RSI_AES_KEY                "Key to encrypt  
the data"  
#define RSI_AES_CBC_IV             "IV input"
```

- Depending on key size/mode of encryption/decryption provided input data is used to perform operation using PUF and AES engine.

###### Update the Wlan configuration file:**rsi\_wlan\_config.h**

RSI_CUSTOM_FEATURE_BIT_MAP	FEAT_CUSTOM_FEAT_EXTENSION_VALID
----------------------------	----------------------------------

RSI\_EXT\_CUSTOM\_FEATURE\_BIT\_MAP|EXT\_FEAT\_PUF

## Executing the Application

1. Connect WiSeConnect device to the Windows PC running Keil or IAR IDE.
2. Configure the macros in the files located at
  - **rsi\_puf\_key\_aes\_app.c**
  - **rsi\_wlan\_config.h**
3. Build and launch the application.
4. After the program gets executed, WiSeConnect Device's PUF will be Started, loaded with a key and AES encryption/Decryption are performed using that key from PUF.

## 4.9 SHA application

### 4.9.1 Example

In cryptography, SHA (Secure Hash Algorithm) is a cryptographic hash function designed by the United States National Security Agency and is a U.S. Federal Information Processing Standard published by the United States NIST

SHA produces a message digest based on principles similar to those used by Ronald L. Rivest of MIT in the design of the MD4 and MD5 message digest algorithms, but has a more conservative design.

SHA forms part of several widely used security applications and protocols, including TLS and SSL, PGP, SSH, S/MIME, and IPsec. Those applications can also use MD5; both MD5 and SHA are descended from MD4.

### Sequence of Events

This Application demonstrates user how to:

- Compute a digest of 64 bytes using SHA.

### 4.9.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module

### 4.9.3 Configuration and Execution of the Application

### Configuring the Application

Edit the **rsi\_sha\_app.c** file

- From given configuration,  
“SHA” refers to data which is given as input to SHA for computing digest.
- Depending on SHA mode digest of respective size is computed for the given input data.

---

## Executing the Application

1. Connect Redpine device to the Windows PC with Keil or IAR IDE in running state.
2. Configure the **SHA** in the file
  - **rsi\_sha\_app.c**
3. Build and launch the application.
4. After the program gets executed, Redpine Device generates the digest of 64 bytes for the given input data.

## 5 WLAN BLE

### 5.1 WLAN-AP BLE bridge Example

#### 5.1.1 Application Overview

##### Overview

The coex application demonstrates how information can be exchanged seamlessly using two wireless protocols (WLAN and BLE) running in the same device.

In this coex application, Redpine BTLE device connects with remote BTLE device (Smart Phone) and Redpine WiFi interface starts as an Access Point and allow stations (Windows laptop) to connect it.

The coex application has WLAN and BLE tasks and acts as an interface between remote Smartphone BTLE device and connected WiFi Station. Smartphone interacts with BLE task, while connected station interacts with WLAN task. When Smartphone connects and sends message to Redpine device, BLE task accepts and sends to WLAN task, which in turn sends to connected station. Similarly, when PC sends message to Redpine device, the message will be sent to Smartphone via BLE task.

Thus messages can be seamlessly transferred between PC and Smartphone.

##### Sequence of Events

##### WLAN Task

This Application explains user how to:

- Create device as an Access Point
- Connect stations to Redpine Access Point
- Open TCP server socket in device
- Receive TCP data sent by connected station and forward to BLE task
- Send data received by BLE task to connected station using TCP protocol

##### LE Task

This Application explains user how to:

- Create chat service
- Configure device in advertise mode
- Connect from Smart phone
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

#### 5.1.2 Application Setup

The WiSeConnect in its many variants supports SPI and UART interfaces. Depending on the interface used, the required set up is as below:

#### 5.1.3 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

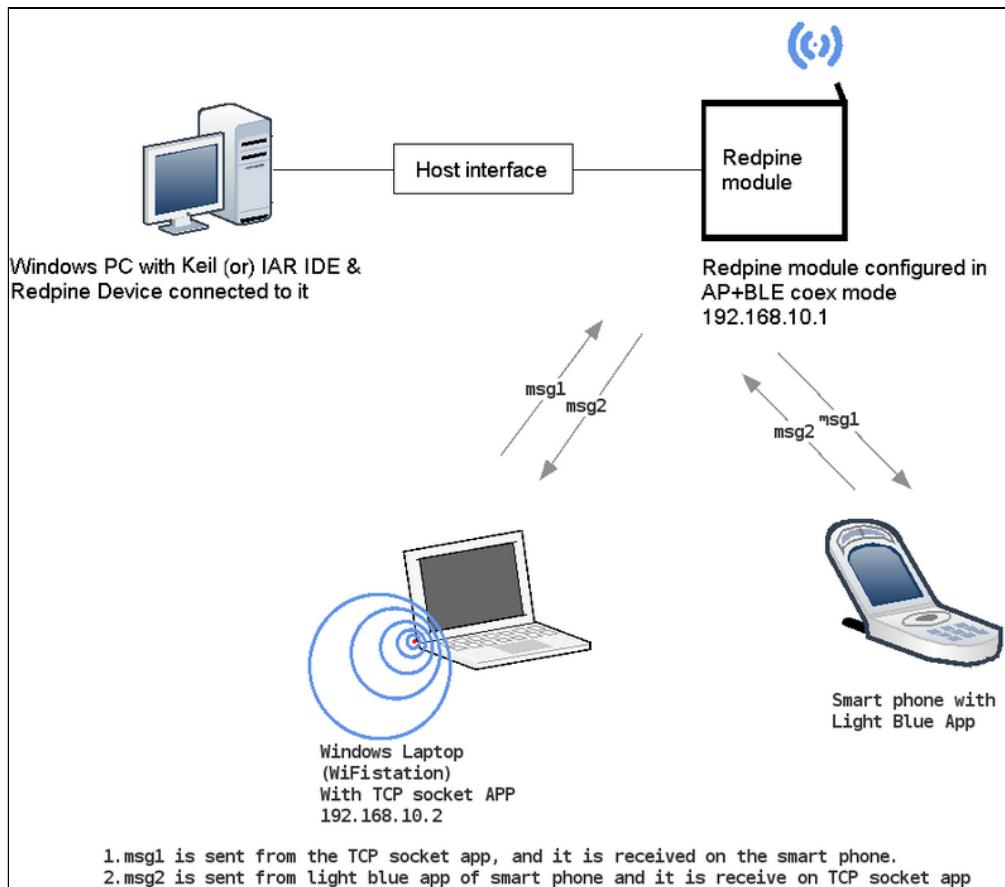
### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Windows Laptop (STA) with TCP socket application

**Note:** Download UDP socket application from below link,  
<http://sourceforge.net/projects/sockettest/files/latest/download>

- BTLE supported Smart phone with GATT client application.

**Note:**  
Install BLE scanner for GATT client application in smart phone.



**Figure 1: Set up to demonstrate WLAN AP BLE bridge Application**

### 5.1.4 Configuration and Execution of the Application

#### Configuring the Application

##### Configuring the WLAN task

1. Open **rsi\_wlan\_app.c** file and update/modify following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID "REDPINE_AP"
```

**CHANNEL\_NO** refers to the channel in which AP would be started

```
#define CHANNEL_NO 11
```

**Note:**

Valid values for **CHANNEL\_NO** are 1 to 11 in 2.4GHz and 36 to 48 & 149 to 165 in 2.4GHz. In this example default configured band is 2.4GHz. So, if user wants to use 5GHz band then user has to set **RSI\_BAND** macro to 5GHz band in **rsi\_wlan\_config.h** file.

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE RSI_WPA2
```

**ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports OPEN, TKIP, CCMP methods.

Valid configuration is:

**RSI\_CCMP** - For CCMP encryption  
**RSI\_TKIP** - For TKIP encryption  
**RSI\_NONE** - For open encryption

```
#define ENCRYPTION_TYPE RSI_CCMP
```

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons in milliseconds. Allowed values are integers from 100 to 1000 which are multiples of 100.

```
#define BEACON_INTERVAL 100
```

**DTIM\_INTERVAL** refers DTIM interval of the Access Point. Allowed values are from 1 to 255.

#define DTIM\_INTERVAL

4

**DEVICE\_PORT** port refers internal TCP server port number

#define DEVICE\_PORT

5001

**To configure IP address**

IP address to be configured to the device should be in long format and in little endian byte order.  
Example: To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

#define DEVICE\_IP

0X010AA8C0

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro GATEWAY as **0x010AA8C0**

#define GATEWAY

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK

0x00FFFFFF

**Note:**

In AP mode, configure same IP address for both DEVICE\_IP and GATEWAY macros.

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_SERVER | TCP_IP_TOTAL_SOCKETS_1 )
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENSION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Configuring the BLE Application

1. Open **rsi\_ble\_app.c** file and update/modify following macros,  
**RSI\_BLE\_NEW\_SERVICE\_UUID** refers to the attribute value of the newly created service.

```
#define RSI_BLE_NEW_SERVICE_UUID 0xAABB
```

**RSI\_BLE\_ATTRIBUTE\_1\_UUID** refers to the attribute type of the first attribute under this service  
(**RSI\_BLE\_NEW\_SERVICE\_UUID**).

```
#define RSI_BLE_ATTRIBUTE_1_UUID 0x1AA1
```

**RSI\_BLE\_ATTRIBUTE\_2\_UUID** refers to the attribute type of the second attribute under this service  
(**RSI\_BLE\_NEW\_SERVICE\_UUID**).

```
#define RSI_BLE_ATTRIBUTE_2_UUID 0x1BB1
```

**RSI\_BLE\_MAX\_DATA\_LEN** refers to the Maximum length of the attribute data.

```
#define RSI_BLE_MAX_DATA_LEN 20
```

**RSI\_BLE\_APP\_DEVICE\_NAME** refers name of the Redpine device to appear during scanning by remote devices.

```
#defineRSI_BLE_APP_DEVICE_NAME  
"WLAN_BLE_SIMPLE_CHAT"
```

Following are the **non-configurable** macros in the application.  
**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.

```
#defineRSI_BLE_CHAR_SERV_UUID 0x2803
```

**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

```
#defineRSI_BLE_CLIENT_CHAR_UUID 0x2902
```

**RSI\_BLE\_ATT\_PROPERTY\_READ** is used to set the READ property to an attribute value.

```
#defineRSI_BLE_ATT_PROPERTY_READ 0x02
```

**RSI\_BLE\_ATT\_PROPERTY\_WRITE** is used to set the WRITE property to an attribute value.

```
#defineRSI_BLE_ATT_PROPERTY_WRITE 0x08
```

**RSI\_BLE\_ATT\_PROPERTY\_NOTIFY** is used to set the NOTIFY property to an attribute value.

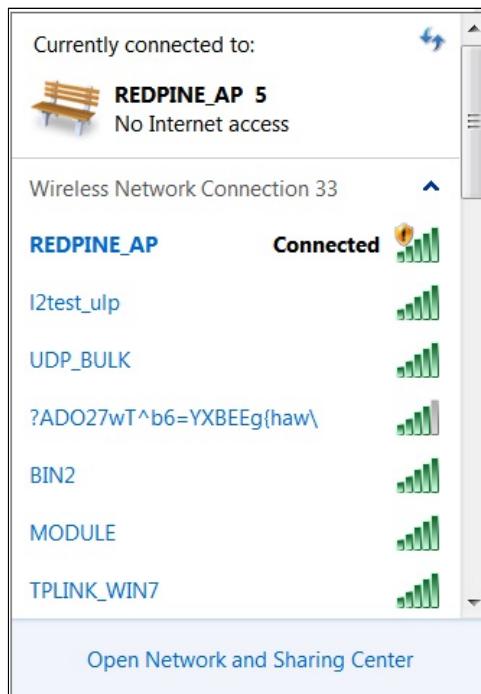
```
#defineRSI_BLE_ATT_PROPERTY_NOTIFY 0x10
```

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

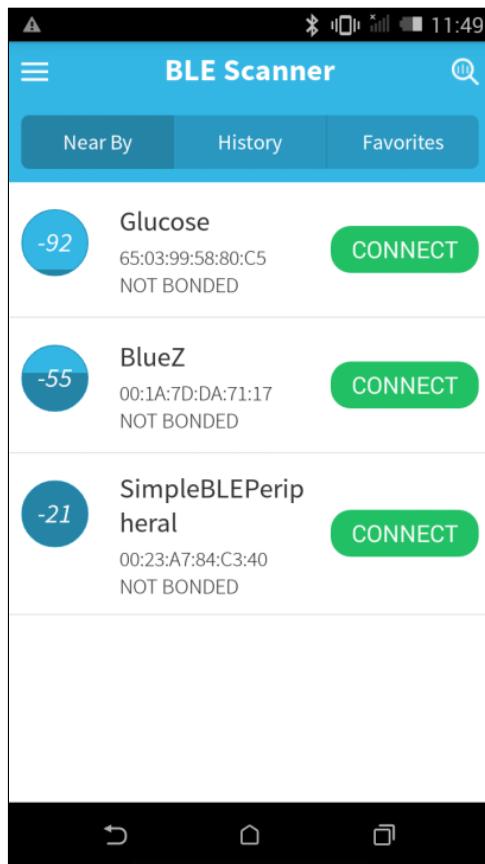
```
#defineBT_GLOBAL_BUFF_LEN 10000
```

## Executing the Application

1. After the program gets executed, Redpine BTLE is in Discoverable state and WLAN will create as an Access Point and opens TCP server socket on port number **DEVICE\_PORT**.
2. From Windows Laptop (STA), do scan and connect to Redpine Access point (Ex: "REDPINE\_AP").

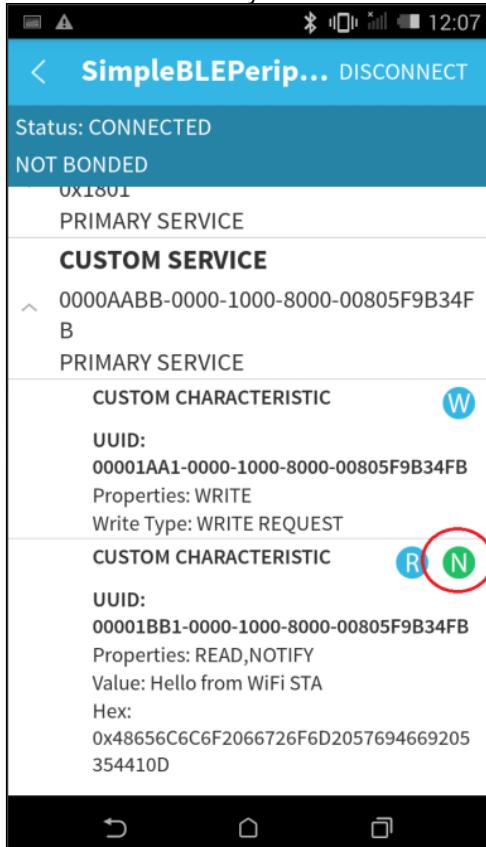


3. Open a LE App in the Smartphone and do Scan.
4. In the App, Redpine module device will appear with the name configured in the macro **RSI\_BLE\_APP\_SIMPLE\_CHAT** (Ex: "WLAN\_BLE\_SIMPLE\_CHAT") or sometimes observed as Redpine device as internal name "**SimpleBLEPeripheral**".

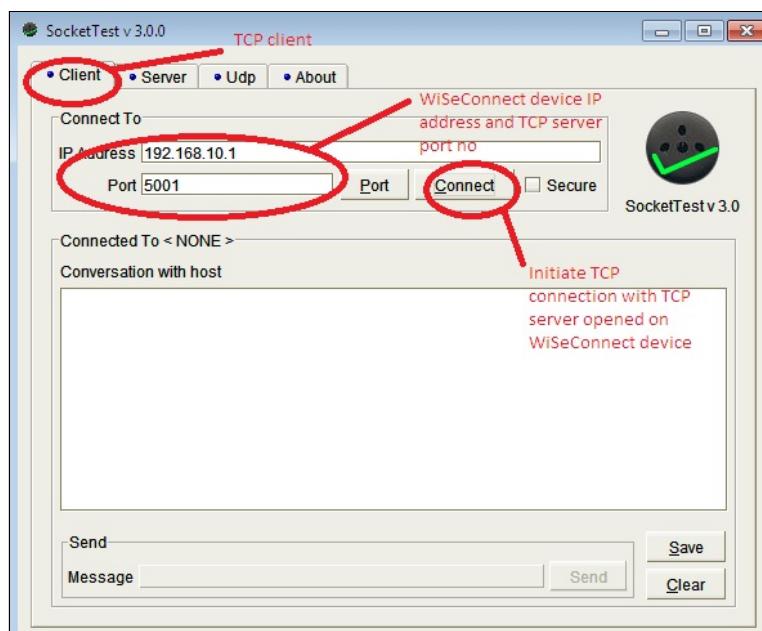


5. Initiate connection from the App.
6. After successful connection, LE scanner displays the supported services of Redpine module.  
Select the attribute service which is added **RSI\_BLE\_NEW\_SERVICE\_UUID**

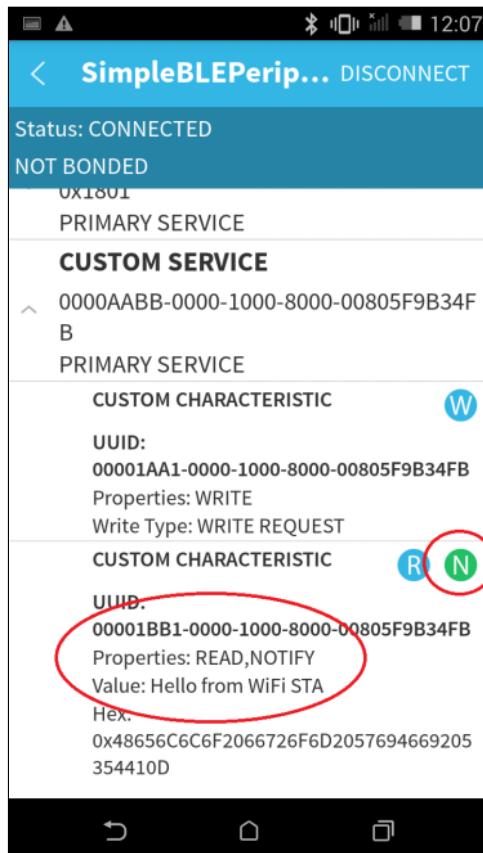
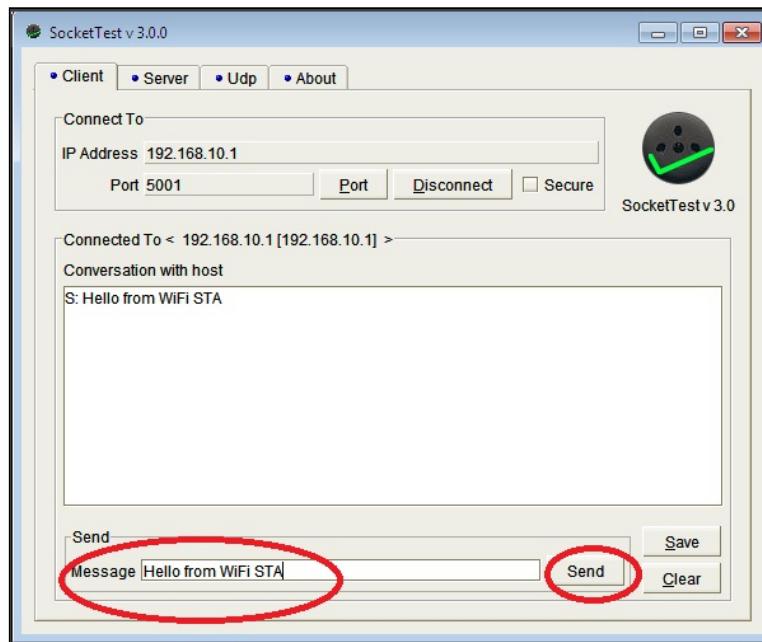
(Ex: 0xAABB) and enable Notification for attribute UUID **RSI\_BLE\_ATTRIBUTE\_2\_UUID** (Ex: 0x1BB1) to receive data sent by WiFi STA.



7. From windows laptop (STA), open TCP client application and initiate TCP connection with TCP server opened on port number **DEVICE\_PORT** in Redpine module.



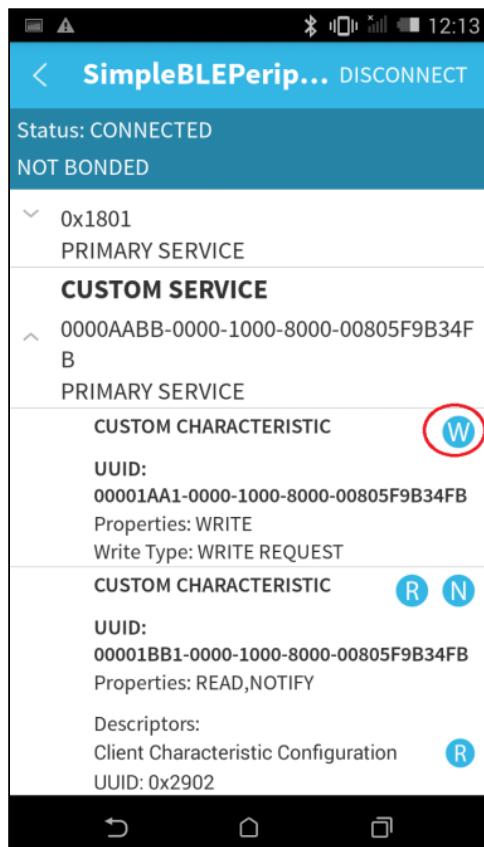
8. After successful TCP connection, send a message (Ex: "Hello from WiFi STA") from TCP client (from laptop) to Redpine device. Redpine device forwards the received message from Windows laptop to remote BTLE device which is connected to Redpine BTLE device over BTLE protocol. User can observe the message notification on attribute UUID **RSI\_BLE\_ATTRIBUTE\_2\_UUID (Ex: 0x1BB1)** in BTLE scanner app.

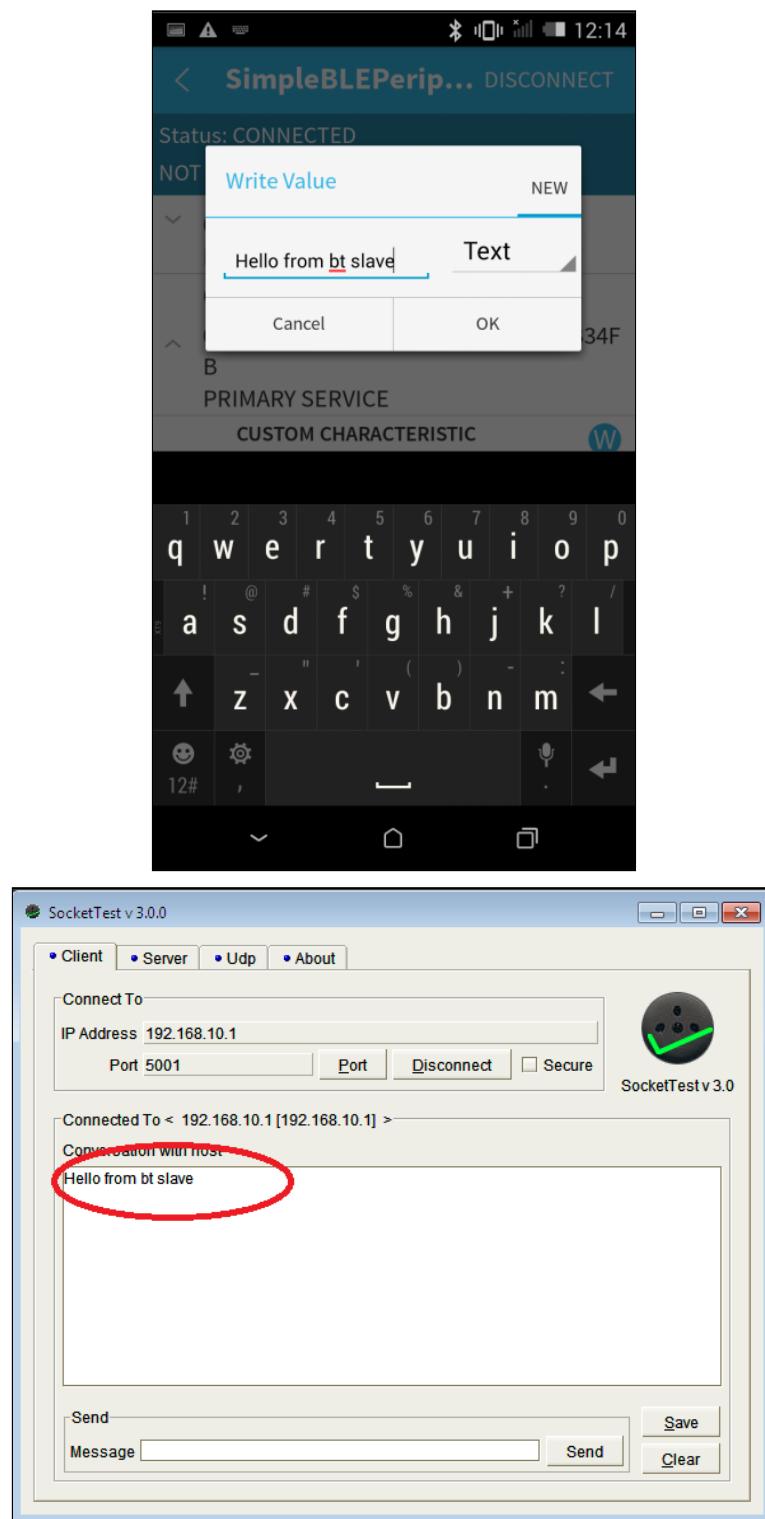


**Note:**

rsi\_wlan\_app\_send\_to\_btle() function defined in **rsi\_ble\_app.c** to send message from WLAN task to BTLE task.

9. Now send a message (Ex: "Hello from bt slave") from GATT client (from smart phone BLE scanner app) using attribute RSI\_BLE\_ATTRIBUTE\_1\_UUID (Ex: 0x1AA1) to Redpine device. Redpine device forwards the received message from BTLE remote device to WiFi STA which is connected to Redpine Access point over WiFi protocol. User can observe the message on TCP socket application.





**Note:**

`rsi_bt_app_send_to_wlan()` function defined in `rsi_wlan_ap_app.c` to send message from BTLE task to WLAN task.

## 5.2 Wlan Ap Ble Bridge TCPIP Bypass

### 5.2.1 Application Overview

#### Overview

The coex application demonstrates how information can be exchanged seamlessly using two wireless protocols (WLAN and BLE) running in the same device.

In this coex application, Redpine BTLE device connects with remote BTLE device (Smart Phone) and Redpine WiFi interface starts as an Access Point and allow stations (Windows laptop) to connect it.

The coex application has WLAN and BLE tasks and acts as an interface between remote Smartphone BTLE device and connected WiFi Station. Smartphone interacts with BLE task, while connected station interacts with WLAN task. When Smartphone connects and sends message to Redpine device, BLE task accepts and sends to WLAN task, which in turn sends to connected station. Similarly, when PC sends message to Redpine device, the message will be sent to Smartphone via BLE task.

Thus messages can be seamlessly transferred between PC and Smartphone.

#### Sequence of Events

#### WLAN Task

This Application explains user how to:

- Create device as an Access Point
- Connect stations to Redpine Access Point
- Receive UDP data sent by connected station and forward to BLE task
- Send data received by BLE task to connected station using UDP protocol

#### BLE Task

This Application explains user how to:

- Create chat service
- Configure device in advertise mode
- Connect from Smart phone
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

### 5.2.2 Application Setup

The Redpine in its many variants supports SPI and UART interfaces. Depending on the interface used, the required set up is as below:

### 5.2.3 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Windows Laptop (STA) with UDP socket application

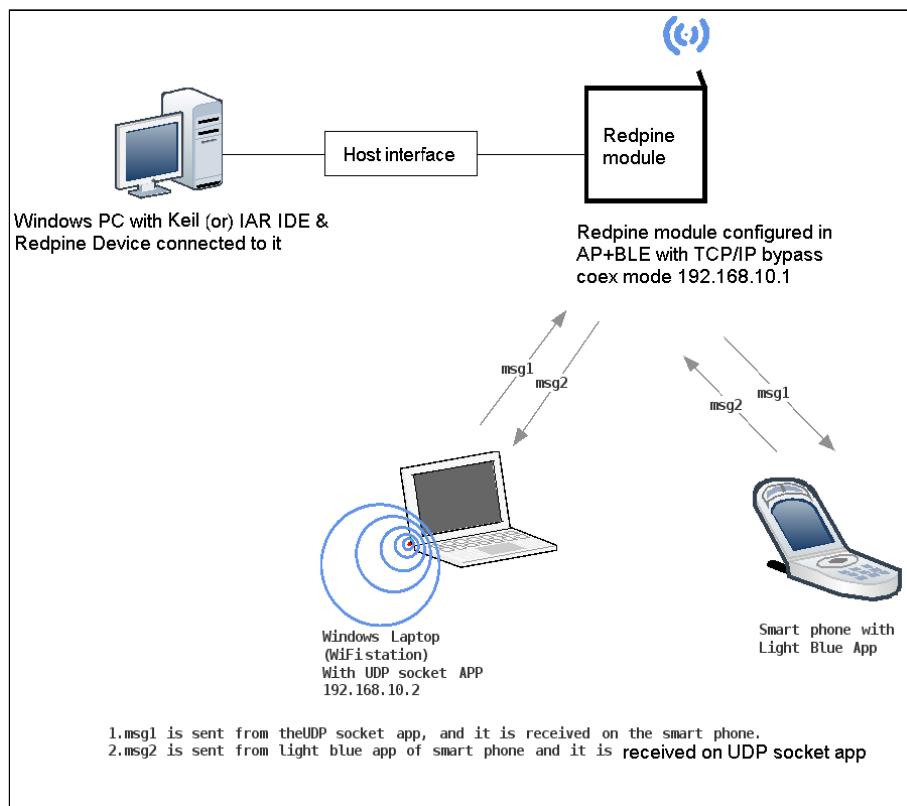
**Note:**

Download UDP socket application from below link,  
\* [http://sourceforge.net/projects/sockettest/files/latest/download\\_](http://sourceforge.net/projects/sockettest/files/latest/download_) \*

- BTLE supported Smart phone with GATT client application.

**Note:**

Install BLE scanner for GATT client application.



**Figure 1: Setup to demonstrate WLAN AP BLE bridge tcpipbypass application**

## 5.2.4 Configuration and Execution of the Application

### Configuring the Application

#### Configuring the WLAN task

1. Open **rsi\_wlan\_ap\_app.c** file and update/modify following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID  
"REDPINE_AP"
```

**CHANNEL\_NO** refers to the channel in which AP would be started

```
#define CHANNEL_NO
```

11

**Note:**

Valid values for CHANNEL\_NO are 1 to 11 in 2.4GHz and 36 to 48 & 149 to 165 in 2.4GHz. In this example default configured band is 2.4GHz. So, if user wants to use 5GHz band then user has to set RSI\_BAND macro to 5GHz band in **rsi\_wlan\_config.h** file.

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE
```

RSI\_WPA2

**ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports OPEN, TKIP, CCMP methods.

Valid configuration is:

**RSI\_CCMP** - For CCMP encryption  
**RSI\_TKIP** - For TKIP encryption  
**RSI\_NONE** - For open encryption

```
#define ENCRYPTION_TYPE
```

RSI\_CCMP

**PSK** refers to the secret key if the Access point to be configured in WPA/WPA2 security modes.

```
#define PSK  
"1234567890"
```

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons in milliseconds. Allowed values are integers from 100 to 1000 which are multiples of 100.

```
#define BEACON_INTERVAL 100
```

**DTIM\_INTERVAL** refers DTIM interval of the Access Point. Allowed values are from 1 to 255.

```
#define DTIM_INTERVAL 4
```

**DEVICE\_PORT** refers internal UDP server port number

```
#define DEVICE_PORT 5001
```

**REMOTE\_PORT** refers remote UDP server port number.

```
#define REMOTE_PORT 5001
```

**REMOTE\_IP\_ADDRESS** refers IP address of the connected STA.

IP address should be in long format and in little endian byte order.

Example: To configure "192.168.10.2" as **REMOTE\_IP\_ADDRESS** then update the macro as **0x020AA8C0**.

```
#define REMOTE_IP_ADDRESS  
0x020AA8C0
```

#### To configure IP address:

IP address to be configured to the device should be in long format and in little endian byte order.

Example: To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP 0X010AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY 0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order  
Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK          0x00FFFFFF
```

**Note:**

In AP mode, configure same IP address for both **DEVICE\_IP** and **GATEWAY** macros.

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS           RSI_ENABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_BYPASS
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENSION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Configuring the BLE Application

1. Open **rsi\_ble\_app.c** file and update/modify following macros,  
**RSI\_BLE\_NEW\_SERVICE\_UUID** refers to the attribute value of the newly created service.

```
#define RSI_BLE_NEW_SERVICE_UUID      0xAABB
```

**RSI\_BLE\_ATTRIBUTE\_1\_UUID** refers to the attribute type of the first attribute under this service  
(**RSI\_BLE\_NEW\_SERVICE\_UUID**).

```
#define RSI_BLE_ATTRIBUTE_1_UUID      0x1AA1
```

**RSI\_BLE\_ATTRIBUTE\_2\_UUID** refers to the attribute type of the second attribute under this service  
(**RSI\_BLE\_NEW\_SERVICE\_UUID**).

```
#define RSI_BLE_ATTRIBUTE_2_UUID      0x1BB1
```

**RSI\_BLE\_MAX\_DATA\_LEN** refers to the Maximum length of the attribute data.

```
#define RSI_BLE_MAX_DATA_LEN
```

20

**RSI\_BLE\_APP\_DEVICE\_NAME** refers name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_APP_DEVICE_NAME  
"WLAN_BLE_SIMPLE_CHAT"
```

Following are the **non-configurable** macros in the application.

**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.

```
#define RSI_BLE_CHAR_SERV_UUID
```

0x2803

**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

```
#define RSI_BLE_CLIENT_CHAR_UUID
```

0x2902

**RSI\_BLE\_ATT\_PROPERTY\_READ** is used to set the READ property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_READ
```

0x02

**RSI\_BLE\_ATT\_PROPERTY\_WRITE** is used to set the WRITE property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_WRITE
```

0x08

**RSI\_BLE\_ATT\_PROPERTY\_NOTIFY** is used to set the NOTIFY property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_NOTIFY
```

0x10

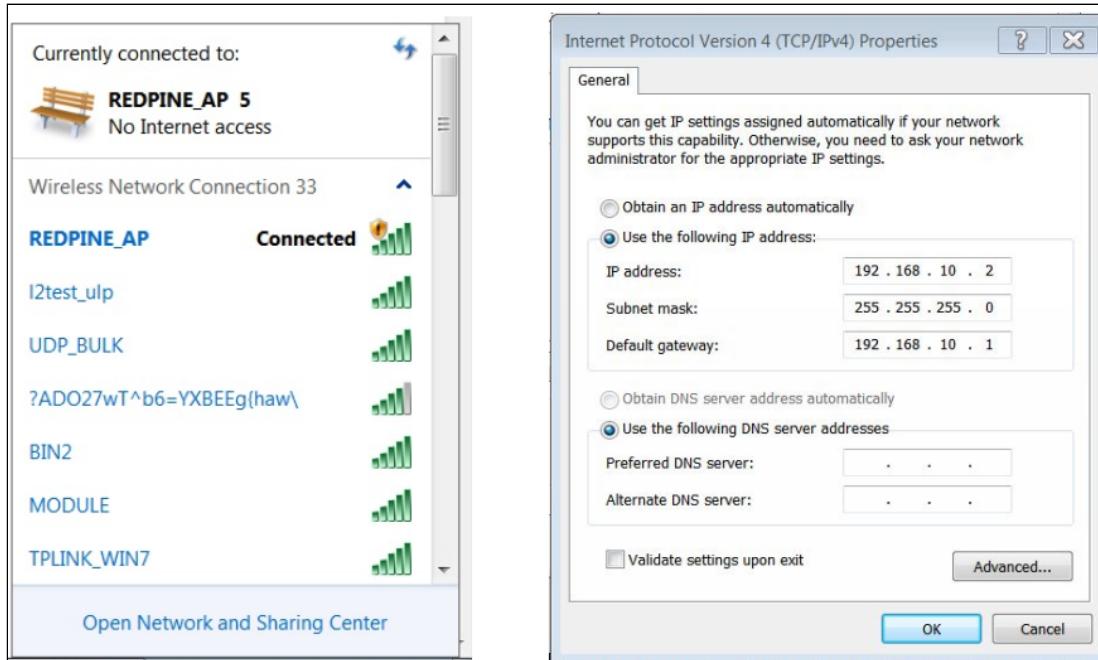
**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN
```

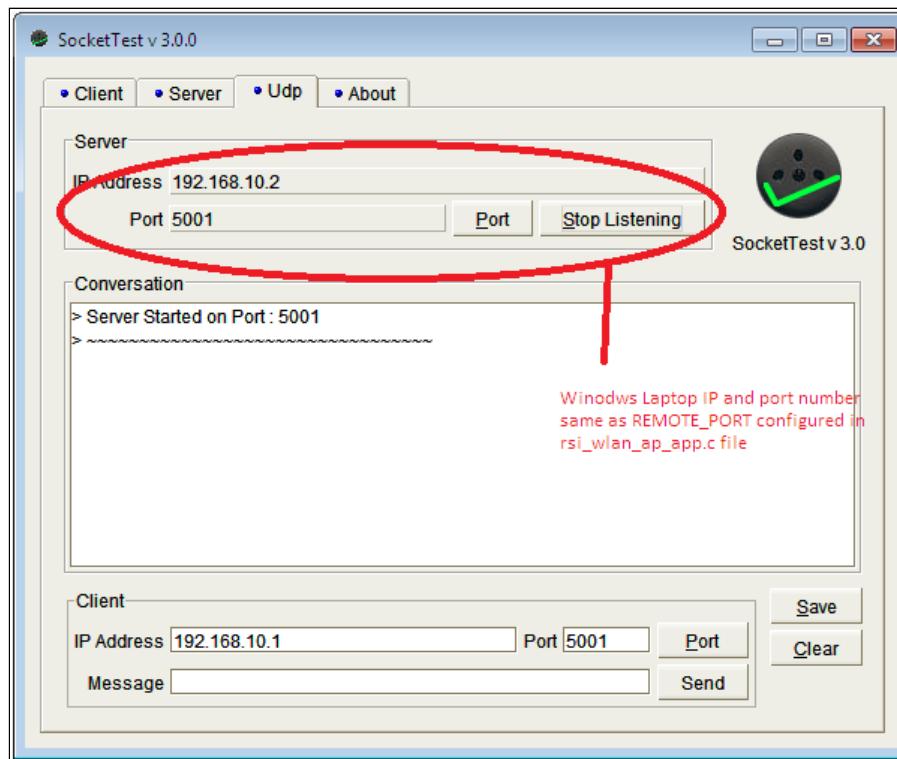
10000

## Executing the Application

1. After the program gets executed, Redpine BTLE is in Discoverable state and WLAN will create as an Access Point and opens UDP server socket on port number **DEVICE\_PORT**.
2. From Windows Laptop (STA), do scan and connect to Redpine Access point (Ex: "REDPINE\_AP") and assign static IP address to laptop which is same as **REMOTE\_IP\_ADDRESS** configured in rsi\_wlan\_ap\_app.c file.

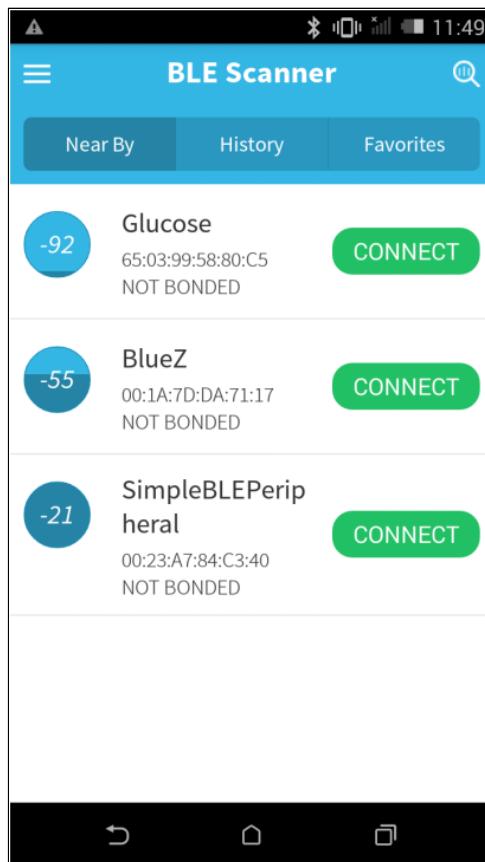


3. Install UDP socket application in Windows laptop and open UDP server socket on port number **REMOTE\_PORT** (Ex: **5001**) to receive the data sent by BTLE remote device.



4. Open a BLE scanner App in the Smartphone and do Scan.

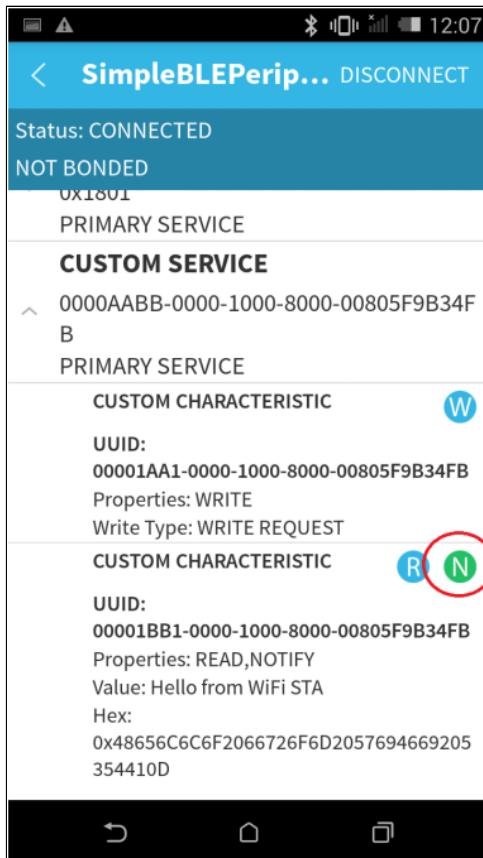
5. In the App, Redpine module device would appear with the name configured in the macro **RSI\_BLE\_APP\_SIMPLE\_CHAT** (Ex: "WLAN\_BLE\_SIMPLE\_CHAT") or sometimes observed as Redpine device as internal name "**SimpleBLEPeripheral**".



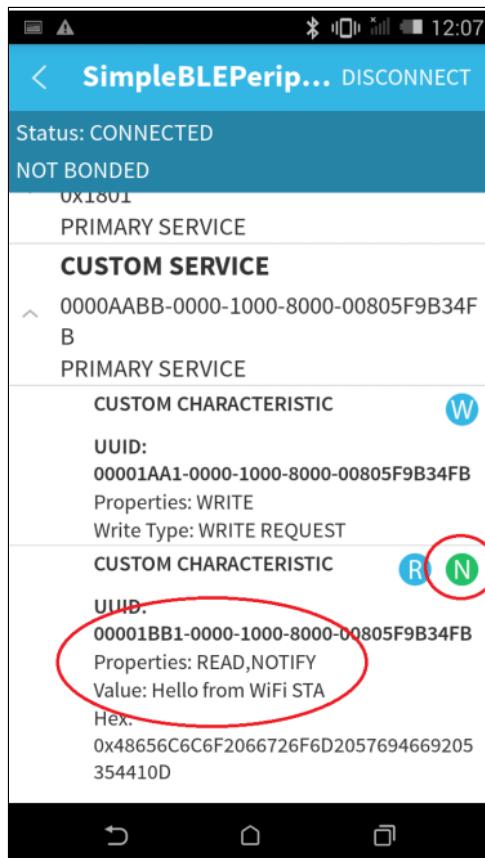
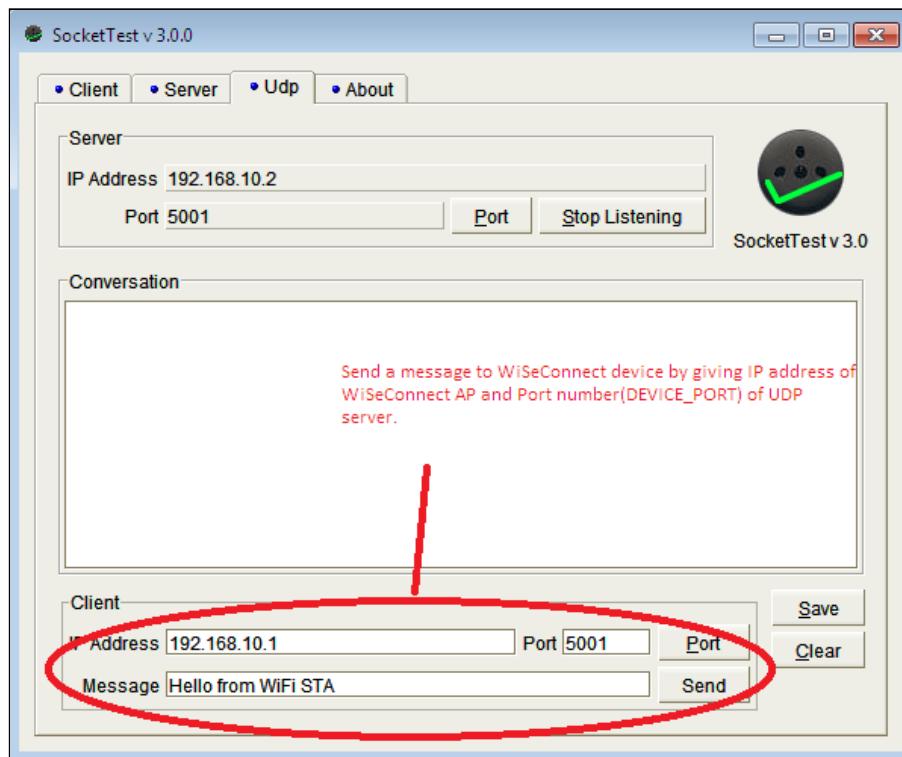
6. Initiate connection from the App.

7. After successful connection, LE scanner displays the supported services of Redpine module.

8. Select the attribute service which is added **RSI\_BLE\_NEW\_SERVICE\_UUID** (Ex: 0xAABB) and enable Notification for attribute UUID **RSI\_BLE\_ATTRIBUTE\_2\_UUID** (Ex: **0x1BB1**) to receive data sent by Wi-Fi STA.



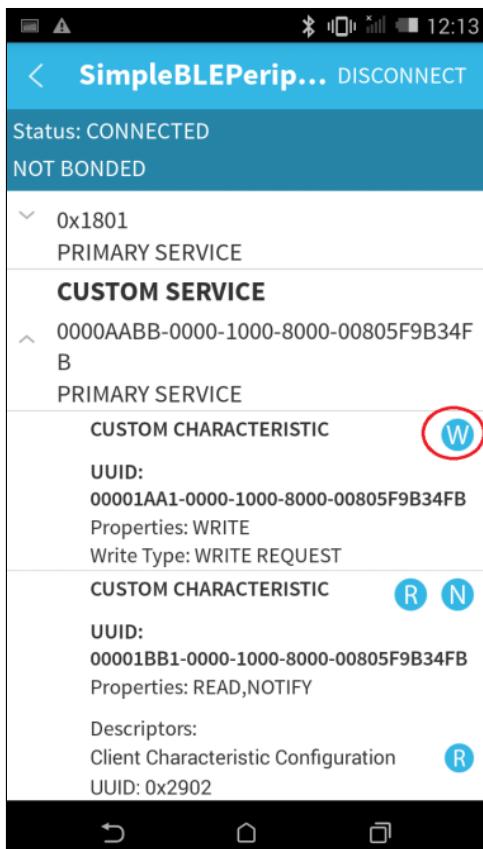
9. Successful TCP connection, send a message (Ex: "Hello from WiFi STA") from UDP client (from laptop) to Redpine device. Redpine device forwards the received message from Windows laptop to remote BTLE device which is connected to Redpine BTLE device over BTLE protocol. User can observe the message notification on attribute UUID **RSI\_BLE\_ATTRIBUTE\_2\_UUID** (Ex: **0x1BB1**) in BTLE scanner app.

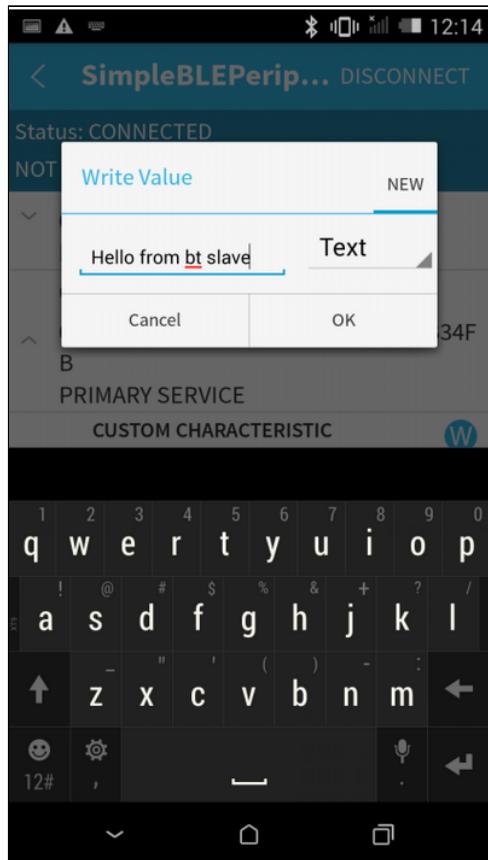


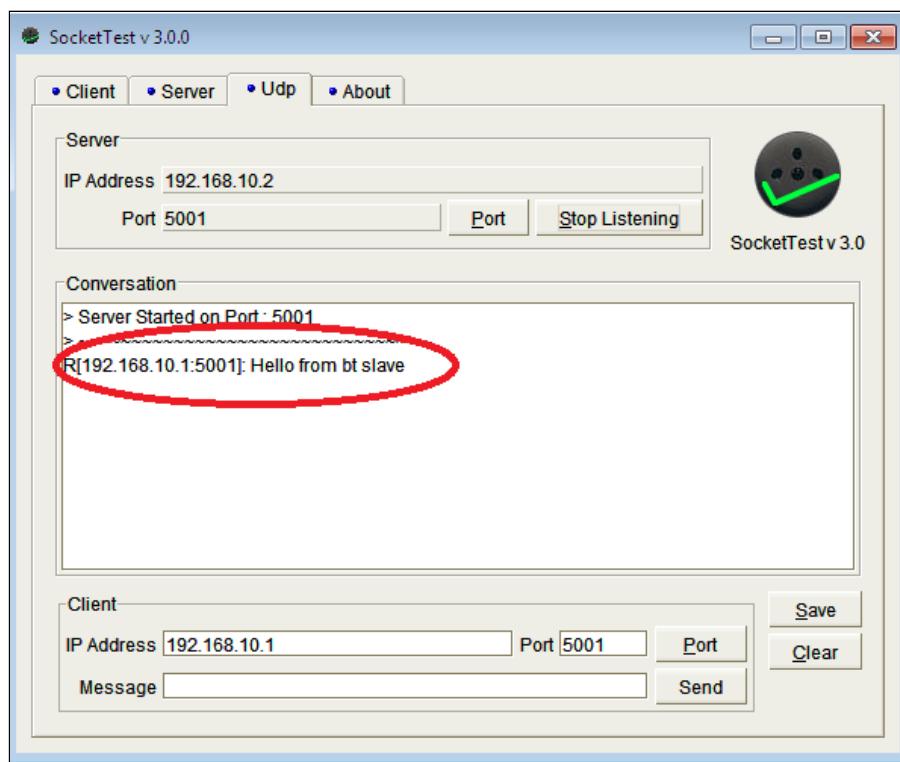
**Note:**

rsi\_wlan\_app\_send\_to\_btle() function defined in rsi\_ble\_app.c to send message from WLAN task to BTLE task.

10. Now send a message (Ex: "Hello from bt slave") from GATT client (from smart phone BLE scanner app) using attribute RSI\_BLE\_ATTRIBUTE\_1\_UUID (Ex: 0x1AA1) to Redpine device. Redpine device forwards the received message from BTLE remote device to WiFi STA which is connected to Redpine Access point over WiFi protocol. User can observe the message on UDP socket application.







**Note:**

`rsi_bt_app_send_to_wlan()` function defined in `rsi_wlan_ap_app.c` to send message from BTLE task to WLAN task.

## 5.3 WLAN-STATION BLE BRIDGE Example

### 5.3.1 Application Overview

The coex application demonstrates how information can be exchanged seamlessly using two wireless protocols (WLAN and BLE) running in the same.

In this coex application, Redpine BTLE device connects with remote BTLE device (Smart Phone) and Redpine WiFi interface connects with an Access Point in station mode and do data transfer in BTLE and WiFi interfaces.

The coex application has WLAN and BLE tasks and acts as an interface between remote Smartphone BTLE device and remote PC which is connected to Access point. Smartphone interacts with BLE task, while remote PC interacts with WLAN task. When Smartphone connects and sends message to Redpine device, BLE task accepts and sends to WLAN task, which in turn sends to remote PC which is connected to Access Point. Similarly, when remote PC sends message to Redpine device, the message will be sent to Smartphone via BLE task.

Thus messages can be seamlessly transferred between PC and Smartphone.

### Sequence of Events

#### WLAN Task

This Application explains user how to:

- Create Redpine device as Station
- Connect Redpine station to remote Access point
- Receive TCP data sent by connected station and forward to BT task

- Send data received by BT task to connected station using TCP over SSL client protocol.

## BLE Task

This Application explains user how to:

- Create chat service
- Configure device in advertise mode
- Connect from Smart phone
- Configure device in power save profile mode 2
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

### 5.3.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Access point
- Windows PC2 with SSL server application (openssl)

**Note:**

Download Open SSL for windows from below link,

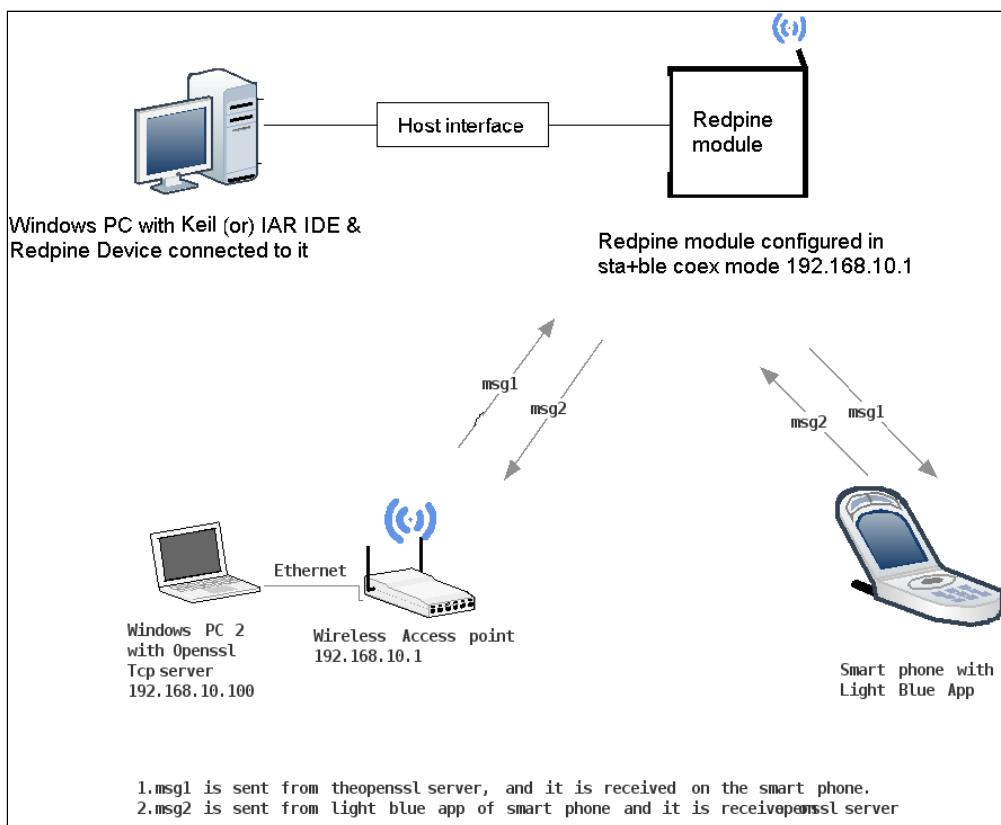
<http://ufpr.dl.sourceforge.net/project/gnuwin32/openssl/0.9.8h-1/openssl-0.9.8h-1-bin.zip>

**In WiSeMCU case, Add the certificates to the /bin folder of Open SSL from release package /Utils/scripts/cert\_generation/example/new\_certs/sslcert/ .**

- BTLE supported Smart phone with GATT client application.

**Note:**

Install BLE scanner for GATT client application.



**Figure 1: Setup to demonstrate WLAN station BLE bridge application**

### 5.3.3 Configuration and Execution of the Application

#### Configuring the Application

##### Configuring the WLAN task

1. Open **rsi\_wlan\_app.c** file and update/modify the following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap  
name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE    RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK             "<psk>"
```

**To Load certificate :**

```
#define LOAD_CERTIFICATE 1
```

If **LOAD\_CERTIFICATE** set to 1, application will load certificate which is included using **rsi\_wlan\_set\_certificate** API.

By default, application loading "cacert.pem" certificate if **LOAD\_CERTIFICATE** enable. In order to load different certificate, user has to follow the following steps:

- **rsi\_wlan\_set\_certificate** API expects the certificate in the form of linear array. So, convert the pem certificate into linear array form using python script provided in the release package "certificate\_script.py"

Ex: If the certificate is wifi-user.pem .Give the command in the following way  
python certificate\_script.py ca-cert.pem

Script will generate wifiuser.pem in which one linear array named cacert contains the certificate.

- After conversion of certificate, update *rsi\_ssl\_client.c* source file by including the certificate file and by providing the required parameters to *rsi\_wlan\_set\_certificate* API.

**Note:**

Once certificate loads into the device, it will write into the device flash. So, user need not load certificate for every boot up unless certificate change.

So define LOAD\_CERTIFICATE as 0, if certificate is already present in the Device.

**Note:**

All the certificates are given in the release package *certificates*

**DEVICE\_PORT** port refers SSL client port number

```
#define DEVICE_PORT                                <local  
port>
```

**SERVER\_PORT** port refers remote SSL server port number

```
#define SERVER_PORT                                <remote  
port>
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with SSL server socket.  
IP address should be in long format and in little endian byte order.

Example: To configure "192.168.10.100" as IP address, update the macro **SERVER\_IP\_ADDRESS** as **0x640AA8C0**.

```
#define SERVER_IP_ADDRESS                         0x640AA8C0
```

**To configure IP address:**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE                                 1
```

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP 0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order  
Example: To configure "192.168.10.1" as Gateway, update the macro GATEWAY as **0x010AA8C0**

```
#define GATEWAY 0X010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order  
Example: To configure "255.255.255.0" as network mask, update the macro NETMASK as **0x00FFFFFF**

```
#define NETMASK 0X00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_CLIENT | TCP_IP_TOTAL_SOCKETS_1 |
TCP_IP_FEAT_SSL)
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENTION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Configuring the BLE Application

1. Open **rsi\_ble\_app.c** file and update/modify following macros,  
**RSI\_BLE\_NEW\_SERVICE\_UUID** refers to the attribute value of the newly created service.

```
#define RSI_BLE_NEW_SERVICE_UUID 0xAABB
```

**RSI\_BLE\_ATTRIBUTE\_1\_UUID** refers to the attribute type of the first attribute under this service  
(**RSI\_BLE\_NEW\_SERVICE\_UUID**).

#define RSI_BLE_ATTRIBUTE_1_UUID	0x1AA1
----------------------------------	--------

**RSI\_BLE\_ATTRIBUTE\_2\_UUID** refers to the attribute type of the second attribute under this service (**RSI\_BLE\_NEW\_SERVICE\_UUID**).

#define RSI_BLE_ATTRIBUTE_2_UUID	0x1BB1
----------------------------------	--------

**RSI\_BLE\_MAX\_DATA\_LEN** refers to the Maximum length of the attribute data.

#define RSI_BLE_MAX_DATA_LEN	20
------------------------------	----

**RSI\_BLE\_APP\_DEVICE\_NAME** refers name of the Redpine device to appear during scanning by remote devices.

#define RSI_BLE_APP_DEVICE_NAME "WLAN_BLE_SIMPLE_CHAT"	
---	--

Following are the **non-configurable** macros in the application.

**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.

#define RSI_BLE_CHAR_SERV_UUID	0x2803
--------------------------------	--------

**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

#define RSI_BLE_CLIENT_CHAR_UUID	0x2902
----------------------------------	--------

**RSI\_BLE\_ATT\_PROPERTY\_READ** is used to set the READ property to an attribute value.

#define RSI_BLE_ATT_PROPERTY_READ	0x02
-----------------------------------	------

**RSI\_BLE\_ATT\_PROPERTY\_WRITE** is used to set the WRITE property to an attribute value.

#define RSI_BLE_ATT_PROPERTY_WRITE	0x08
------------------------------------	------

**RSI\_BLE\_ATT\_PROPERTY\_NOTIFY** is used to set the NOTIFY property to an attribute value.

#defineRSI\_BLE\_ATT\_PROPERTY\_NOTIFY

0x10

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver.

#defineBT\_GLOBAL\_BUFF\_LEN

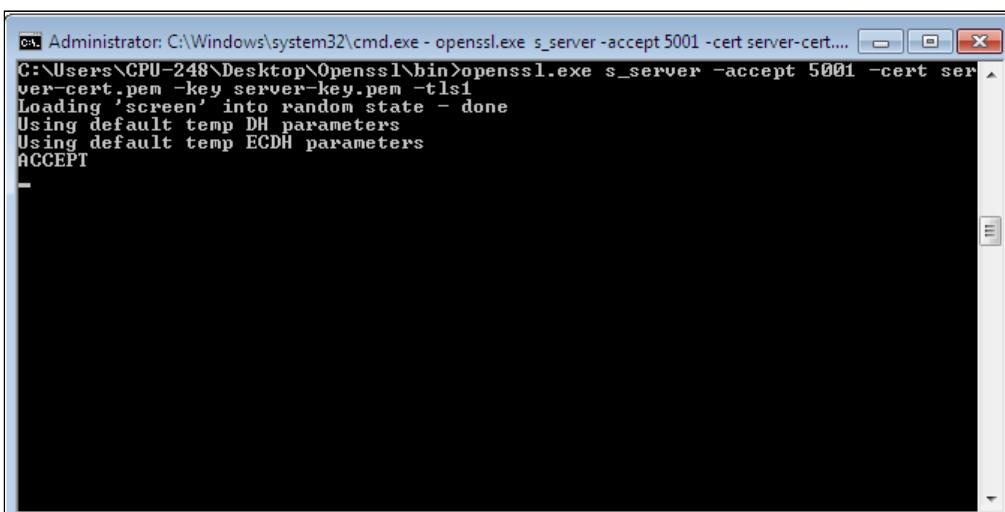
10000

## Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. In Windows PC2 which is connected to AP through LAN, Download the Openssl package from above mentioned link and run SSL server by giving following command:

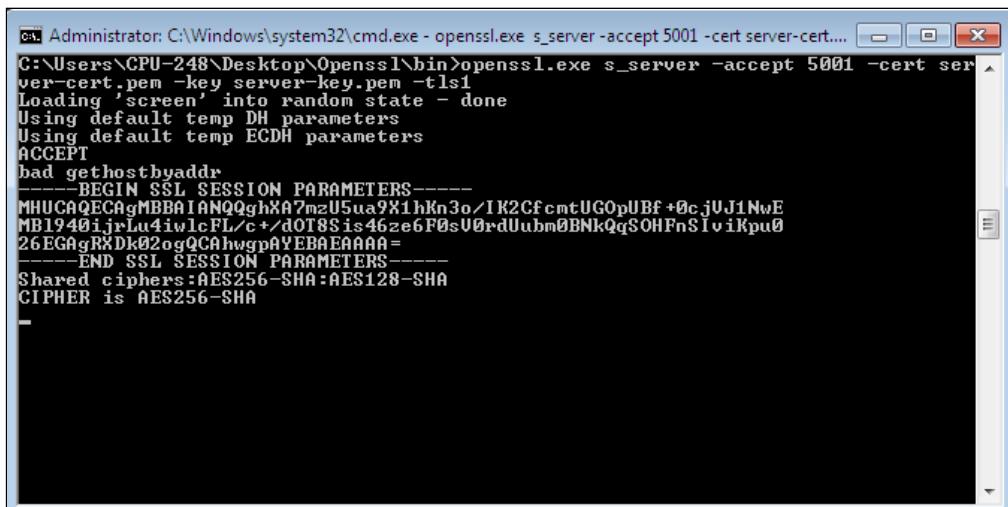
**Openssl.exe s\_server -accept<SERVER\_PORT> -cert <server\_certificate\_file\_path> -key <server\_key\_file\_path> -tls<tls\_version>**

**Example: openssl.exe s\_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1**



```
C:\Administrator: C:\Windows\system32\cmd.exe -openssl.exe s_server -accept 5001 -cert server-cert....  
C:\Users\NCPU-248\Desktop\Openssl\bin>openssl.exe s_server -accept 5001 -cert server-cert....  
Loading 'screen' into random state - done  
Using default temp DH parameters  
Using default temp ECDH parameters  
ACCEPT
```

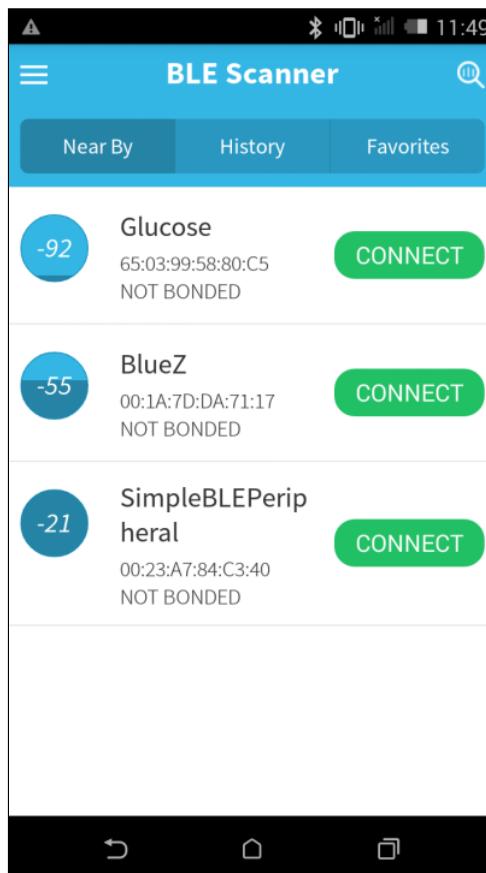
3. After the program gets executed, Redpine BLE is in Advertising state and WLAN connects to Access Point and establishes SSL connectivity with SSL server opened on Windows PC1. Please refer the given below image for connection establishment at windows PC1,



```
C:\> Administrator: C:\Windows\system32\cmd.exe - openssl.exe s_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1
C:\>Users\CPU-248\Desktop\OpenSSL\bin>openssl.exe s_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1
Loading 'screen' into random state - done
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
bad gethostbyaddr
-----BEGIN SSL SESSION PARAMETERS-----
MHUCAQECAgMBB0IAnQQghkRA7mzU5ua9X1hKn3o/IK2CfcmtUGOpUBf+0c.jUJ1NwE
MB1940ijrLu4iwLcFL/c+/dOT8Sis46ze6F0sU0rdUbm0BNkQqSOHFnSIvIKpu0
26EGAgRXDk02ogQCahwgphYEBAEAAAAA=
-----END SSL SESSION PARAMETERS-----
Shared ciphers: AES256-SHA:AES128-SHA
CIPHER is AES256-SHA
-
```

4. Open a BLE scanner App in the Smartphone and do the Scan.

5. In the App, Redpine module device will appear with the name configured in the macro **RSI\_BLE\_APP\_SIMPLE\_CHAT** (Ex: "**"WLAN\_BLE\_SIMPLE\_CHAT"**) or sometimes observed as Redpine device as internal name "**SimpleBLEPeripheral**".

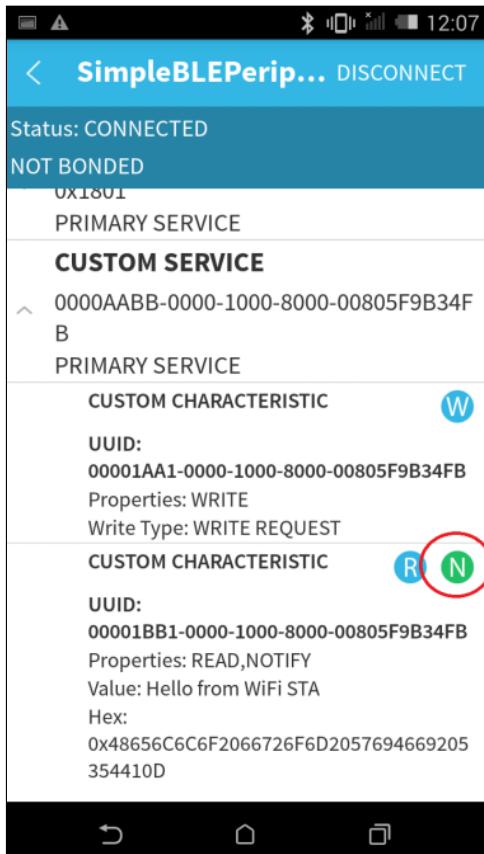


6. Initiate BLE connection from the App.

7. After successful connection, LE scanner displays the supported services of Redpine module.

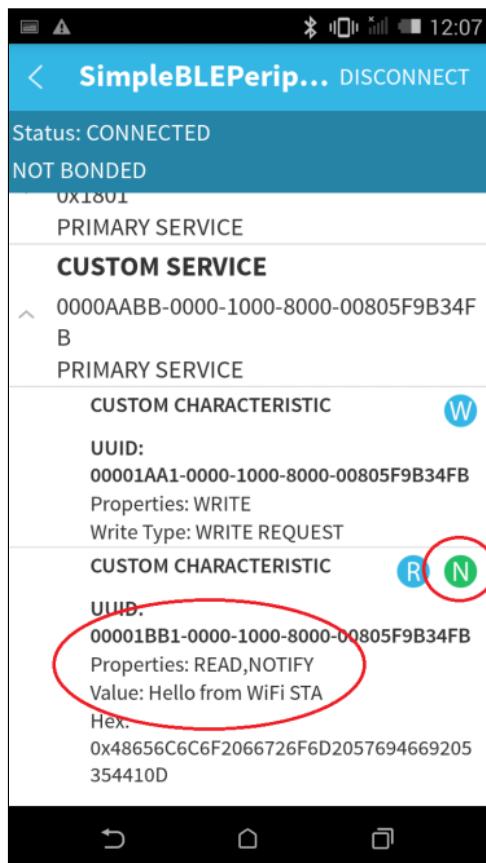
8. Select the attribute service which is added **RSI\_BLE\_NEW\_SERVICE\_UUID**

(Ex: 0xAABB) and enable Notification for attribute UUID **RSI\_BLE\_ATTRIBUTE\_2\_UUID** (Ex: 0x1BB1) to receive data sent by Wi-Fi STA.



9. Now from SSL server (windows PC1), send a message (Ex: "Hello from WiFi STA") to Redpine device. Redpine device forwards the received message from SSL server to remote BTLE device which is connected to Redpine BTLE device over BTLE protocol. User can observe the message notification on attribute UUID **RSI\_BLE\_ATTRIBUTE\_2\_UUID** (Ex: 0x1BB1) in BTLE scanner app.

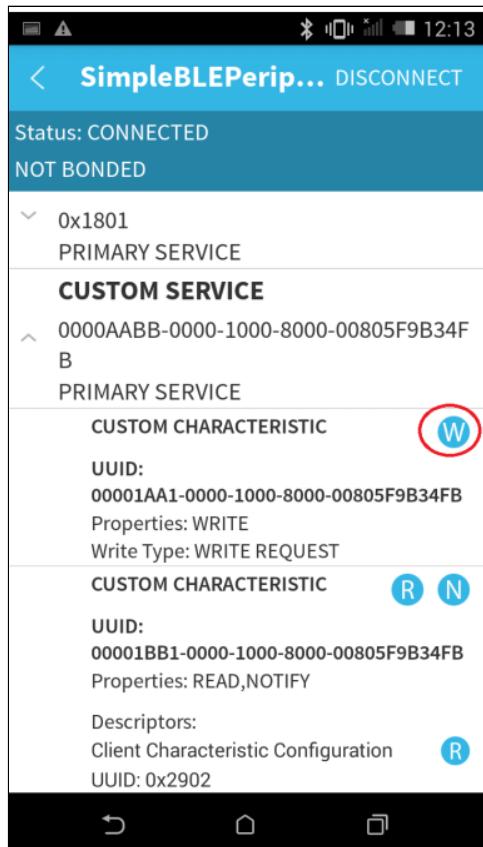
```
C:\Administrator: C:\Windows\system32\cmd.exe - openssl.exe s_server -accept 5001 -cert server-cert...  
C:\Users\CPU-248\Desktop\OpenSSL\bin>openssl.exe s_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1  
Loading 'screen' into random state - done  
Using default temp DH parameters  
Using default temp ECDH parameters  
ACCEPT  
bad gethostbyaddr  
----  
BEGIN SSL SESSION PARAMETERS----  
MHUCAQECAgMBBAIANQQghXA7mzU5ua9X1hKn3o/IK2CfcmtUGOpUBf+0cjUJ1NwE  
MB1940ijrLu4iwlcFL/c+/dOT8Sis46ze6F0sU0rdUubm0BNkQqSOHFnSIvikpu0  
26EGqgRXDk02ogQCAhwgpAYEBAEAAAA=-----  
END SSL SESSION PARAMETERS-----  
Shared cipher is AES256-SHA  
Cipher is AES256-SHA  
Hello from WiFi STA
```

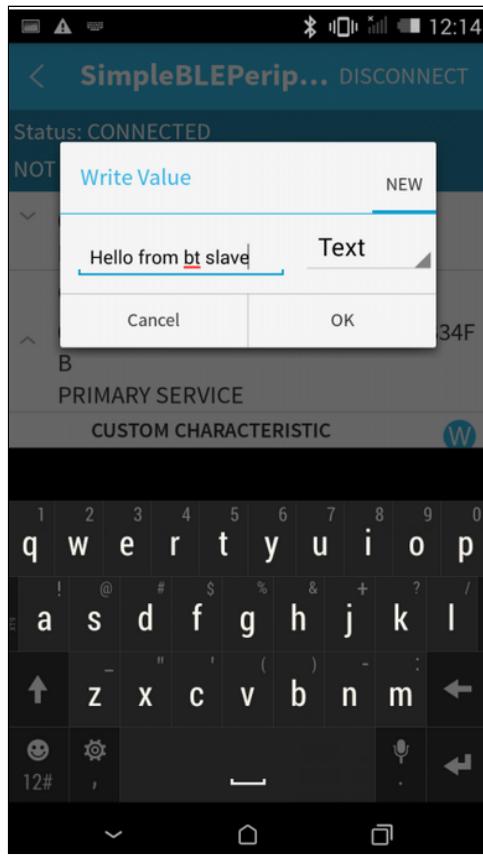


**Note:**

`rsi_wlan_app_send_to_btle()` function defined in `rsi_ble_app.c` to send message from WLAN task to BTLE task

10. Now send a message (Ex: "Hello from bt slave") from GATT client (from smart phone BLE scanner app) using attribute **RSI\_BLE\_ATTRIBUTE\_1\_UUID** (Ex: 0x1AA1) to Redpine device. Redpine device forwards the received message from BTLE remote device to SSL server over WiFi protocol. User can observe the message on UDP socket application.





```
C:\Administrator: C:\Windows\system32\cmd.exe - openssl.exe s_server -accept 5001 -cert server-cert...  
C:\Users\CPU-248\Desktop\OpenSSL\bin>openssl.exe s_server -accept 5001 -cert server-cert...  
ver-cert.pem -key server-key.pem -tls1  
Loading 'screen' into random state - done  
Using default temp DH parameters  
Using default temp ECDH parameters  
ACCEPT  
bad gethostbyaddr  
-----BEGIN SSL SESSION PARAMETERS-----  
MHUCAQECAgMBBAIANQQghXA7mzU5ua9X1hKn3o/IK2CfcmtUGOpUBf+0cjVJ1NwE  
MB1940ijrLu4iwlcFL/c+/dOT8Sis46ze6F0sU0rdUubm0BNkQqSOHFnS1v1Kpu0  
26EGgRXDk02ogQCAhwgpAYEBAEAAAA=-----END SSL SESSION PARAMETERS-----  
Shared cipher: AES256-SHA:AES128-SHA  
CIPHER is AES256-SHA  
Hello from WiFi STA  
Hello from bt slave
```

**Note:**

`rsi_bt_app_send_to_wlan()` function defined in `rsi_wlan_app.c` to send message from BTLE task to WLAN task.

## 5.4 WLAN-STATION BLE-Dual Role Bridge Example

### 5.4.1 Application Overview

The coex application demonstrates how information can be exchanged seamlessly using two wireless protocols (WLAN and BLE) running in the same device as a LE master as well as slave.

#### Description

The coex application has WLAN and BLE tasks and acts as an interface between Smartphone and sensor tag and PC. Smartphone and sensor tag interacts with BLE task, while PC Both PC and Redpine WLAN would be connected to a Wireless Access Point, thus both are connected together wirelessly interacts with WLAN task. When Smartphone and sensor tag connects and sends messages/notifications to Redpine, BLE task accepts and sends to WLAN task, which in turn sends to Access Point connected PC. Thus messages can be seamlessly transferred from Smartphone and sensor tag to Windows PC.

#### Details of the Application

Redpine WLAN acts as a Station and connects to an Access Point  
Redpine BLE acts as a Peripheral (Slave) device and Central (Master), while Smart phone acts as a Central (Master) device and Sensor tag acts as a Peripheral (Slave) device.  
Initially, proprietary Simple chat service is created at GATT Server (Redpine device) to facilitate message exchanges.

- The WLAN task (running in Redpine device) mainly includes following steps.
  1. Connects to a Access Point
  2. Exchanges data over SSL socket with the peer(Windows PC)
- The BLE task (running in Redpine device) mainly includes following steps.
  1. Creates chat service
  2. Configures the device to scanning.
  3. After connection of slave configures the device to Advertise

WLAN and BLE tasks forever run in the application to serve the asynchronous events.

#### Sequence of Events

#### WLAN Task

This Application explains user how to:

- Create Redpine device as Station
- Connect Redpine station to remote Access point
- Receive TCP data sent by connected station and forward to BT task
- Send data received by BT task to connected station using TCP over SSL client protocol.

#### BLE Task

This Application explains user how to:

- Create chat service
- Configure device in advertise mode
- Connect from Smart phone

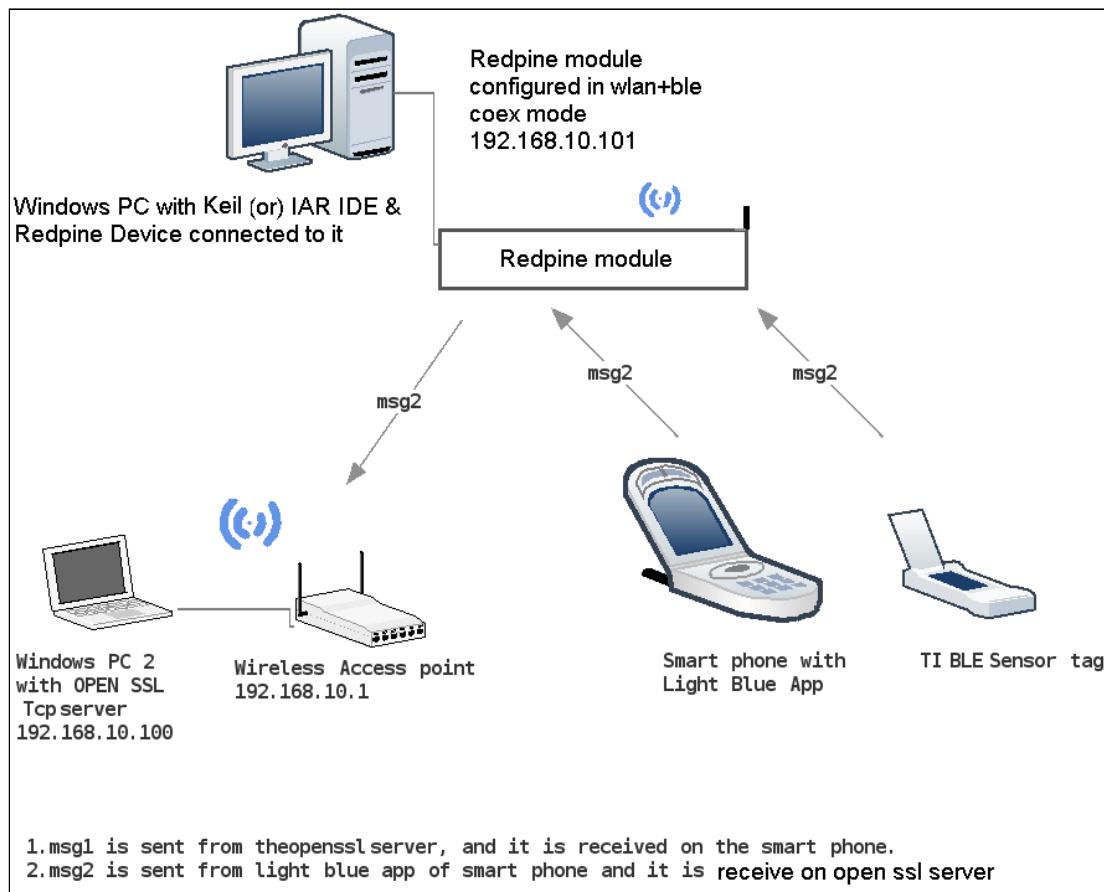
- Configure device in master mode
- Connect to remote device
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

#### 5.4.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Smart phone/tablet with LE Application (Ex: Light blue App in iPad mini)
- WLAN Access Point and a Windows PC
- TI simple sensor tag or 3<sup>rd</sup> party BLE dongle as a slave.



**Figure 1: Setup diagram**

### 5.4.3 Configuration and Execution of the Application

#### Configuring the Application

##### Configuring the WLAN task

1. Open **rsi\_wlan\_app.c** file and update/modify following macros,  
**SSID** refers to the Access point to which user wants to connect.  
**SECURITY\_TYPE** is the security type of the Access point.  
**PSK** refers to the secret key if the Access point is configured in WPA/WPA2 security modes.

```
#define SSID                                "<ap_name>"  
#define SECURITY_TYPE                         <security-type>  
#define PSK                                    ""
```

Load the SSL CA- certificate using **rsi\_wlan\_set\_certificate** API after wireless initialization.

**Note:**

**rsi\_wlan\_set\_certificate** expects the certificate in the form of linear array. Python script is provided in the release package named "**"certificate\_script.py"**" in the following path "certificates" to convert the pem certificate into linear array

Example: If the certificate is ca-cert.pem, give the command as  
python certificate\_script.py ca-cert.pem

The script will generate cacert.pem, in which one linear array named *cacert* contains the certificate  
Enable/Disable DHCP mode

- 1 – Enables DHCP mode (gets the IP from DHCP server)  
0 – Disables DHCP mode

```
#define DHCP_MODE                            1
```

If DHCP mode is disabled, then change the following macros to configure static IP address  
IP address to be configured to the device should be in long format and in little endian byte order.  
Example: To configure "192.168.10.101" as IP address, update the macro **DEVICE\_IP** as **0x650AA8C0**.

```
#define DEVICE_IP                           0x650AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order  
Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY                            0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order  
Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK	0x00FFFFFF
-----------------	------------

To establish TCP connection and transfer data to the remote socket configure the below macros. If SSL is enabled, open the socket with protocol type as 1.  
Internal socket port number.

#define DEVICE_PORT	5001
---------------------	------

Port number of the remote server

#define SERVER_PORT	5001
---------------------	------

IP address of the remote server

#define SERVER_IP_ADDRESS	0x650AA8C0
---------------------------	------------

Include rsi\_wlan\_app.c , rsi\_ble\_app.c and main.c files in the project, build and launch the application  
Open SSL server socket on remote machine

For example, to open SSL socket with port number 5001 on remote side, use the command as given below  
**openssl s\_server -accept<SERVER\_PORT> -cert <server\_certificate\_file\_path> -key <server\_key\_file\_path> -tls<tls\_version>**

**Example:** openssl s\_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1\_2

**Note:**

All the certificates are given in the release package

2. Enable/Disable power save  
1 – Enables Power save mode  
0 – Disables Power save mode

#define WLAN_POWER_SAVE	0
-------------------------	---

By default Power save is disabled.

3. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP (TCP_IP_FEAT_DHCPV4_CLIENT | TCP_IP_TOTAL_SOCKETS_1 | TCP_IP_FEAT_SSL)
#define RSI_CUSTOM_FEATURE_BIT_MAP FEAT_CUSTOM_FEAT_EXTENSION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP EXT_FEAT_256K_MODE
#define RSI_BAND RSI_BAND_2P4GHZ
```

## Configuring the BLE Application

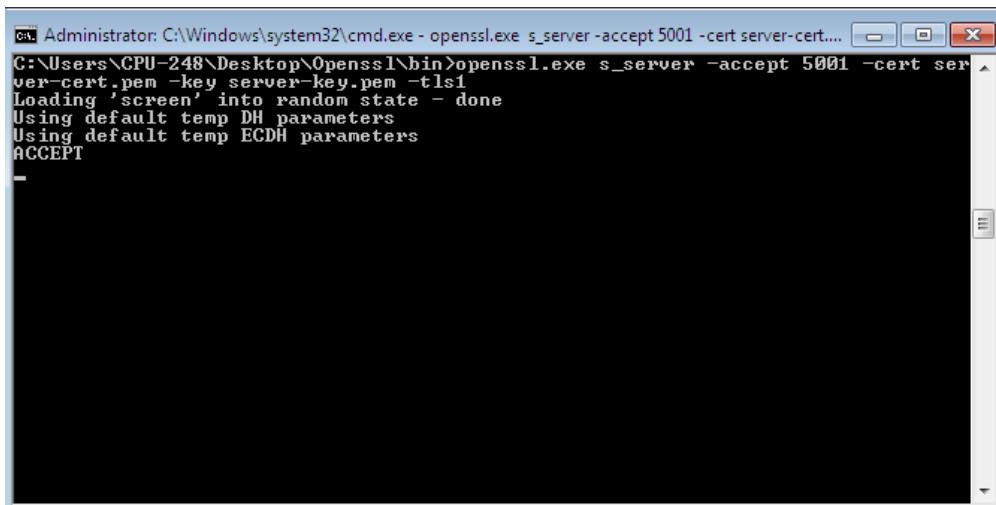
1. Open **rsi\_ble\_app.c** file and update/modify following macros,
  - RSI\_BLE\_NEW\_SERVICE\_UUID - The attribute value of the newly created service. Ex: 0xAABB
  - RSI\_BLE\_ATTRIBUTE\_1\_UUID - The attribute type of the first attribute under this Service. Ex: 0x1AA1
  - RSI\_BLE\_ATTRIBUTE\_2\_UUID - The attribute type of the second attribute under this Service. Ex: 0x1BB1
  - RSI\_BLE\_MAX\_DATA\_LEN - Maximum length of the attribute data(limited to max of 20 bytes)
  - RSI\_BLE\_APP\_DEVICE\_NAME - Name of the Redpine device to appear during Scanning by peer devices.
  - BLE\_PS\_ENABLE – To Enable/Disable power save.
  - RSI\_BLE\_DEV\_ADDR – Address of the peer device to connect.
  - BLE\_DUAL\_ROLE\_FIRST\_MASTER – To create local device as a master first.

Following are the **non-configurable** macros in the application.

- RSI\_BLE\_ATT\_PROPERTY\_READ – Used to set read property to an attribute value.
- RSI\_BLE\_ATT\_PROPERTY\_WRITE – Used to set write property to an attribute value.
- RSI\_BLE\_ATT\_PROPERTY\_NOTIFY – Used to set notify property to an attribute value.
- RSI\_BLE\_CHAR\_SERV\_UUID - The attribute type of the characteristics to be added in a service. Ex: 0x2803
- RSI\_BLE\_CLIENT\_CHAR\_UUID - The attribute type of the client characteristics descriptor to be added in a service characteristic. Ex: 0x2902
- BT\_GLOBAL\_BUFF\_LEN – Number of bytes required for the Application and the Driver.

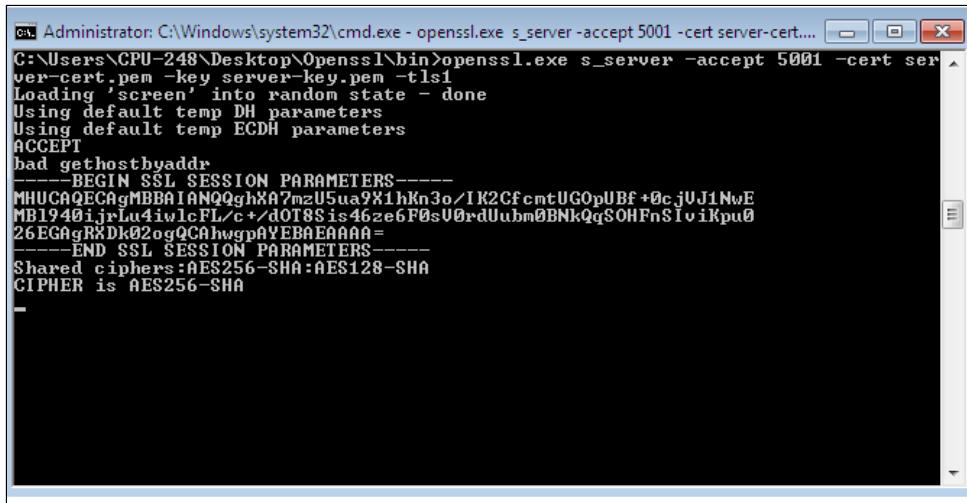
## Executing the coex Application

1. Connect Redpine device to the Windows PC running KEIL or IAR IDE
2. Build and launch the application.
3. Advertise 3<sup>rd</sup> party TI sensor tag.
4. After the program gets executed, If BLE\_DUAL\_ROLE\_FIRST\_MASTER macro as 1 then Redpine BLE is in scanning state and it creates a connection with TI sensor tag. If BLE\_DUAL\_ROLE\_FIRST\_MASTER macro as 0 then Redpine BLE is in Advertising state and WLAN has established SSL socket with peer(PC).
5. Open a LE App in the Smartphone and do scan.
6. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.  
**Openssl.exe s\_server -accept<SERVER\_PORT> -cert <server\_certificate\_file\_path> -key <server\_key\_file\_path> -tls<tls\_version>**  
**Example: openssl.exe s\_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls**



```
C:\ Administrator: C:\Windows\system32\cmd.exe - openssl.exe s_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1
C:\Users\CPU-248\Desktop\Openssl\bin>openssl.exe s_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1
Loading 'screen' into random state - done
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
-
```

WLAN connects to Access Point and establishes SSL connectivity with SSL server opened on Windows PC1. Please refer the given below image for connection establishment at windows PC1,



```
C:\ Administrator: C:\Windows\system32\cmd.exe - openssl.exe s_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1
C:\Users\CPU-248\Desktop\Openssl\bin>openssl.exe s_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1
Loading 'screen' into random state - done
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
bad gethostbyaddr
-----BEGIN SSL SESSION PARAMETERS-----
MHUCAQECAgMBBAIANQQghXA7mzU5ua9X1hKn3o/IK2CfcmtUGOpUBf+0c.jUJ1NuE
MB1940ijrLu4iwlcFL/c+dOT8Si46ze6F0sV0rdUubm0BNkQqSOHFnS1v.iKpu0
26EGAqRXDk02ogQCAhwgpAYEBAAEAAA=
-----END SSL SESSION PARAMETERS-----
Shared ciphers: AES256-SHA:AES128-SHA
CIPHER is AES256-SHA
-
```

send a message or notification from the App to Redpine BLE. Observe this message in the PC connected via SSL socket with Redpine WLAN.

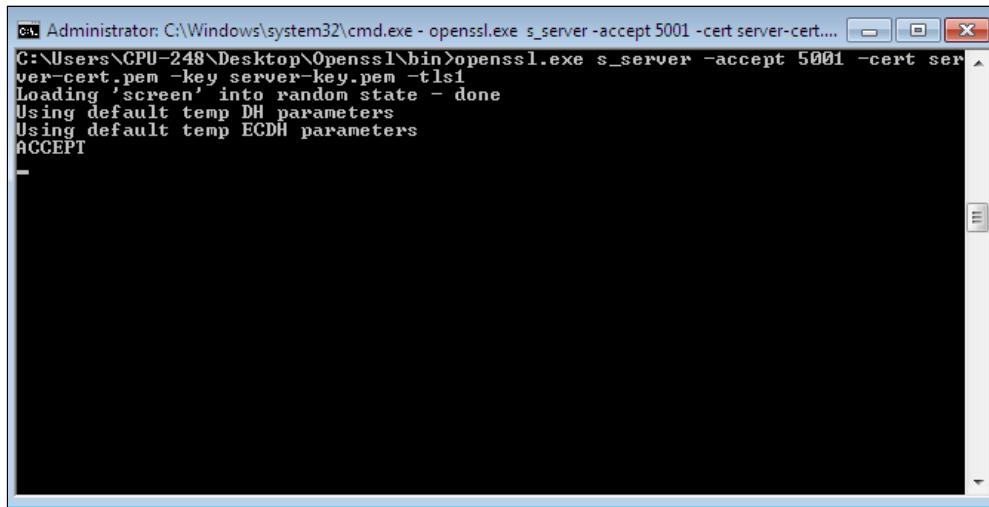
- rsi\_ble\_app\_send\_to\_wlan() function defined in rsi\_wlan\_app.c to send message from BLE task to WLAN task.

In Windows PC2 which is connected to AP through LAN, Download the Openssl package from above mentioned link and run SSL server by giving following command:

**Openssl.exe s\_server -accept<SERVER\_PORT> -cert <server\_certificate\_file\_path> -key**

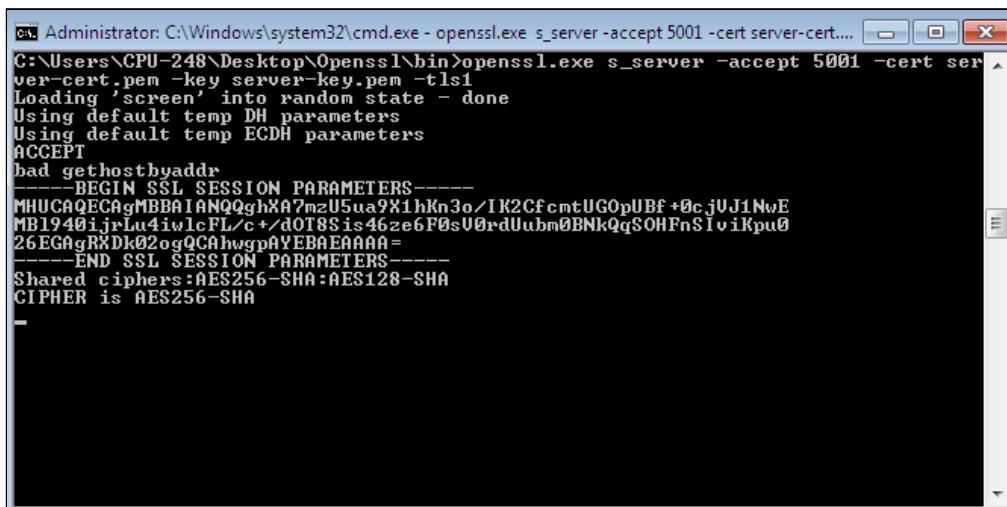
**<server\_key\_file\_path> -tls<tls\_version>**

**Example: openssl.exe s\_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1**



```
Administrator: C:\Windows\system32\cmd.exe - openssl.exe s_server -accept 5001 -cert server-cert.pem
C:\Users\CPU-248\Desktop\Openssl\bin>openssl.exe s_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1
Loading 'screen' into random state - done
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
```

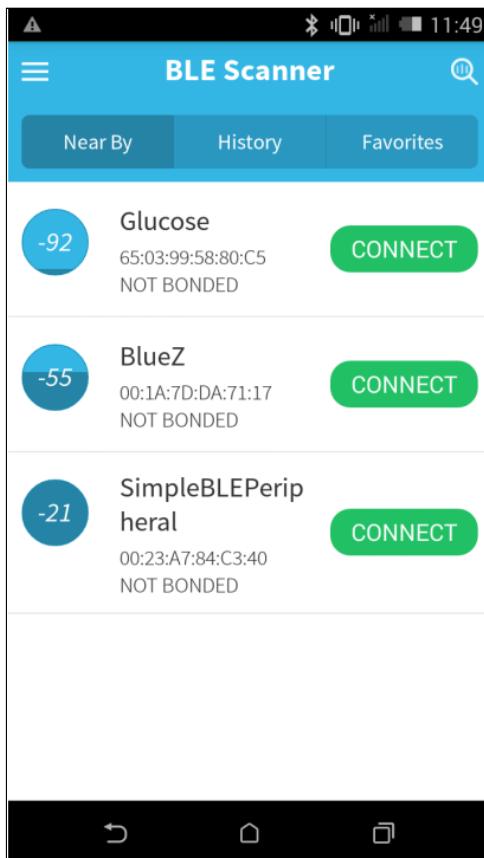
7. After the program gets executed, Redpine BLE is in Advertising state and WLAN connects to Access Point and establishes SSL connectivity with SSL server opened on Windows PC1. Please refer the given below image for connection establishment at windows PC1,



```
Administrator: C:\Windows\system32\cmd.exe - openssl.exe s_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1
Loading 'screen' into random state - done
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
bad gethostbyaddr
-----BEGIN SSL SESSION PARAMETERS-----
MHUCAQECAgMBBAIANQQghXa7mzU5ua9X1hKn3o/IK2CfcmtUGOpUBF+0cjUJ1NwE
MB1940ijrLu4iwLcFL/c+/dOT8Sis46ze6F0sU0rdUubm0BNkQqSOHFnS1vikpu0
26EGAgRXDk02ogQCAhwgpAYEBEA=====
-----END SSL SESSION PARAMETERS-----
Shared ciphers:AES256-SHA:AES128-SHA
CIPHER is AES256-SHA
```

8. Open a BLE scanner App in the Smartphone and do the Scan.

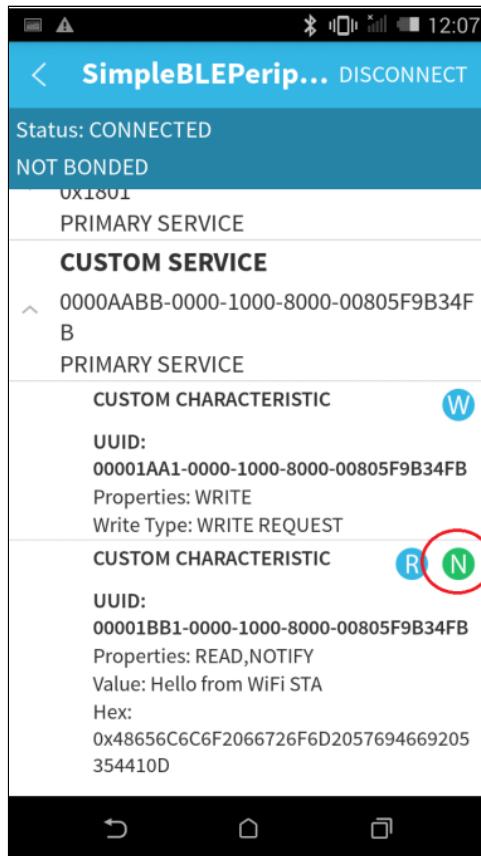
9. In the App, Redpine module device will appear with the name configured in the macro **RSI\_BLE\_APP\_SIMPLE\_CHAT** (Ex: "WLAN\_BLE\_SIMPLE\_CHAT") or sometimes observed as Redpine device as internal name "**SimpleBLEPeripheral**".



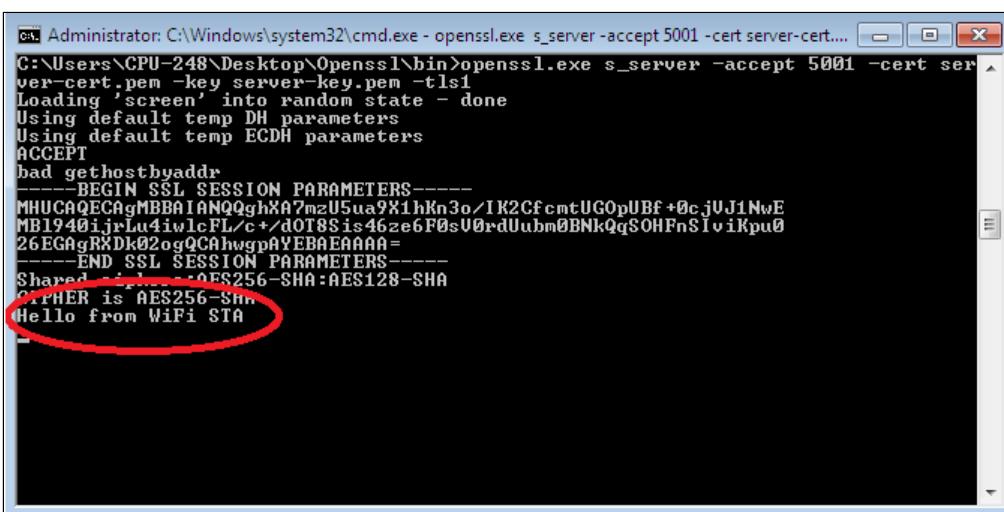
10. Initiate BLE connection from the App.

- After successful connection, LE scanner displays the supported services of Redpine module.
- Select the attribute service which is added **RSI\_BLE\_NEW\_SERVICE\_UUID**

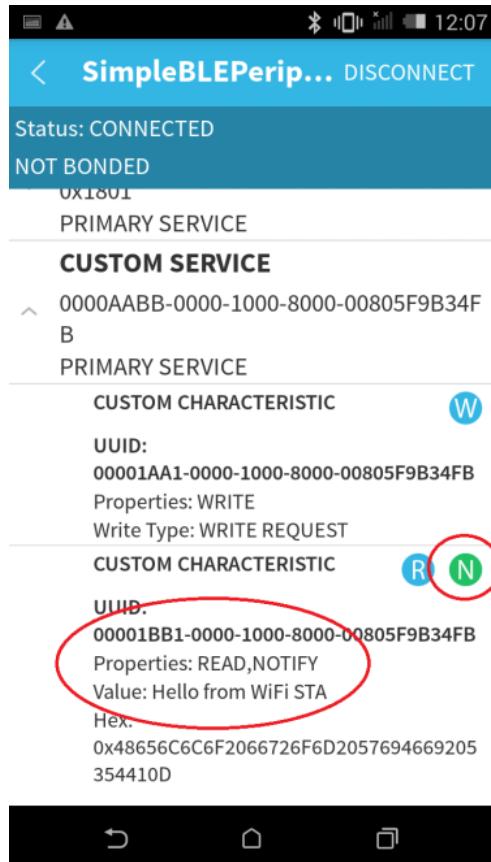
(Ex: 0xAABB) and enable Notification for attribute UUID **RSI\_BLE\_ATTRIBUTE\_2\_UUID**(Ex: **0x1BB1**) to receive data sent by WiFi STA.



11. Now from SSL server (windows PC1), send a message (Ex: "Hello from WiFi STA") to Redpine device. Redpine device forwards the received message from SSL server to remote BTLE device which is connected to Redpine BTLE device over BTLE protocol. User can observe the message notification on attribute UUID **RSI\_BLE\_ATTRIBUTE\_2\_UUID**(Ex: **0x1BB1**) in BTLE scanner app.



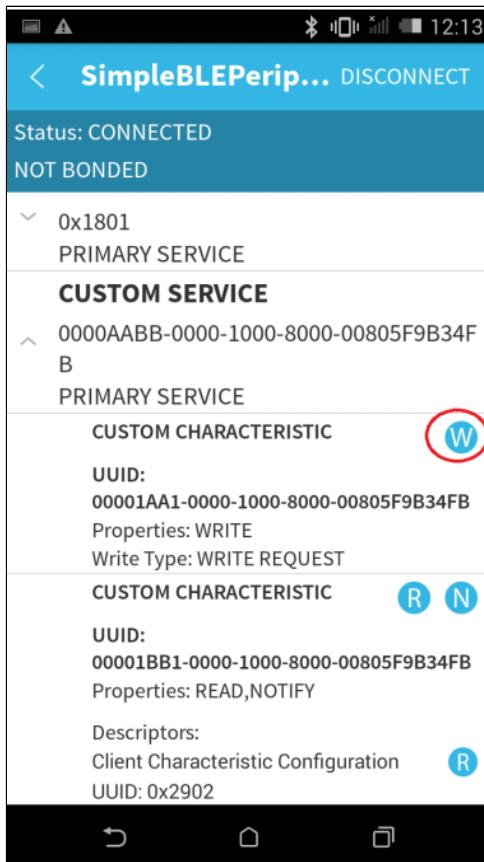
```
C:\Administrator: C:\Windows\system32\cmd.exe - openssl.exe s_server -accept 5001 -cert server-cert...
C:\Users\CPU-248\Desktop\Openssl\bin>openssl.exe s_server -accept 5001 -cert server-cert...
ver-cert.pem -key server-key.pem -tls1
Loading 'screen' into random state - done
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
bad gethostbyaddr
-----BEGIN SSL SESSION PARAMETERS-----
MHUCAQECAgMBAlIANQghXA7mzU5ua9X1hKn3o/IK2CfcmUOGOpUBF+0c_jUJ1NwE
MB1940ijrLu4iwlcFL/c+/dOT8Sis46ze6F0sU0rdUubm0BNkQqSOHPnSIViKpu0
26EGAgRxDI02ogQCAhwgpAYEBEA=====
-----END SSL SESSION PARAMETERS-----
Shared cipher is AES256-SHA:AES128-SHA
Cipher is AES256-SHA
Hello from WiFi STA
```

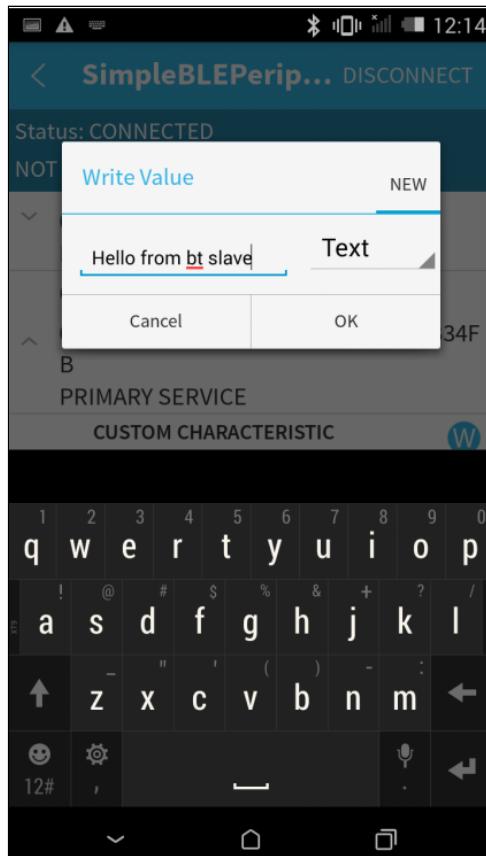


**Note:**

`rsi_wlan_app_send_to_btle()` function defined in `rsi_ble_app.c` to send message from WLAN task to BTLE task.

12. Now send a message (Ex: "Hello from bt slave") from GATT client (from smart phone BLE scanner app) using attribute **RSI\_BLE\_ATTRIBUTE\_1\_UUID** (Ex: 0x1AA1) to Redpine device. Redpine device forwards the received message from BTLE remote device to SSL server over WiFi protocol. User can observe the message on UDP socket application.





```
C:\Administrator:C:\Windows\system32\cmd.exe - openssl.exe s_server -accept 5001 -cert server-cert...  
C:\Users\CPU-248\Desktop\OpenSSL\bin>openssl.exe s_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1  
Loading 'screen' into random state - done  
Using default temp DH parameters  
Using default temp ECDH parameters  
ACCEPT  
bad gethostbyaddr  
-----  
BEGIN SSL SESSION PARAMETERS-----  
MHUCAQECgqMBBAIAQ0QghXAt?mzU5ua9X1hKn3o/IK2CfcmUGoPUBF+0c_jUJ1NwE  
MB1940ijrLu4iwlcFL/c+/dOT8Sis46ze6F0sU0rdUubm0BNkQqSOHFnS1v1Kpu0  
26EGAgRXDk02ogQCAhwgpAYEBAEAAAAA=-----  
END SSL SESSION PARAMETERS-----  
Shared ciphers: AES256-SHA:AES128-SHA  
Cipher is AES256-SHA  
Hello from WiFi STA  
Hello from bt slave
```

**Note:**

`rsi_bt_app_send_to_wlan()` function defined in `rsi_wlan_app.c` to send message from BTLE task to WLAN task.

## 5.5 WLAN-STATION BLE-Multiple Slaves Bridge Example

### 5.5.1 Application Overview

The coex application demonstrates how information can be exchanged seamlessly using two wireless protocols (WLAN and BLE) running in the same device.

#### Description

The coex application has WLAN and BLE tasks and acts as an interface between TI sensor tag and PC. TI Sensor tag interacts with BLE task, while PC Both PC and Redpine WLAN would be connected to a Wireless Access Point, thus both are connected together wirelessly interacts with WLAN task. When TI sensor tag connects and sends notifications to Redpine, BLE task receive and sends to WLAN task, which in turn sends to Access Point connected PC. Thus messages can be seamlessly receives from TI sensor tag to Windows PC.

#### Details of the Application

Redpine WLAN acts as a Station and connects to an Access Point  
Redpine BLE acts as a Central (Master) with GATT Server running in it, while TI Sensor tag acts as a Peripheral (Slave) device.  
Initially, proprietary simple chat service is created at GATT Server (Redpine device) to facilitate message/notification exchanges.

1. The WLAN task (running in Redpine device) mainly includes following steps.
  - Connects to a Access Point
  - Exchanges data over SSL socket with the peer(Windows PC)
  
1. The BLE task (running in Redpine device) mainly includes following steps.
  - Configures the device to scanning.

WLAN and BLE tasks forever run in the application to serve the asynchronous events

#### Sequence of Events

#### WLAN Task

This Application explains user how to:

- Create Redpine device as Station
- Connect Redpine station to remote Access point
- Receive TCP data sent by connected station and forward to BT task
- Send data received by BT task to connected station using TCP over SSL client

#### BLE Task

This Application explains user how to:

- Create chat service
- Configure device in Central mode
- Connect to multiple slaves one by one
- Receive data sent by Smart phone and forward to WLAN task

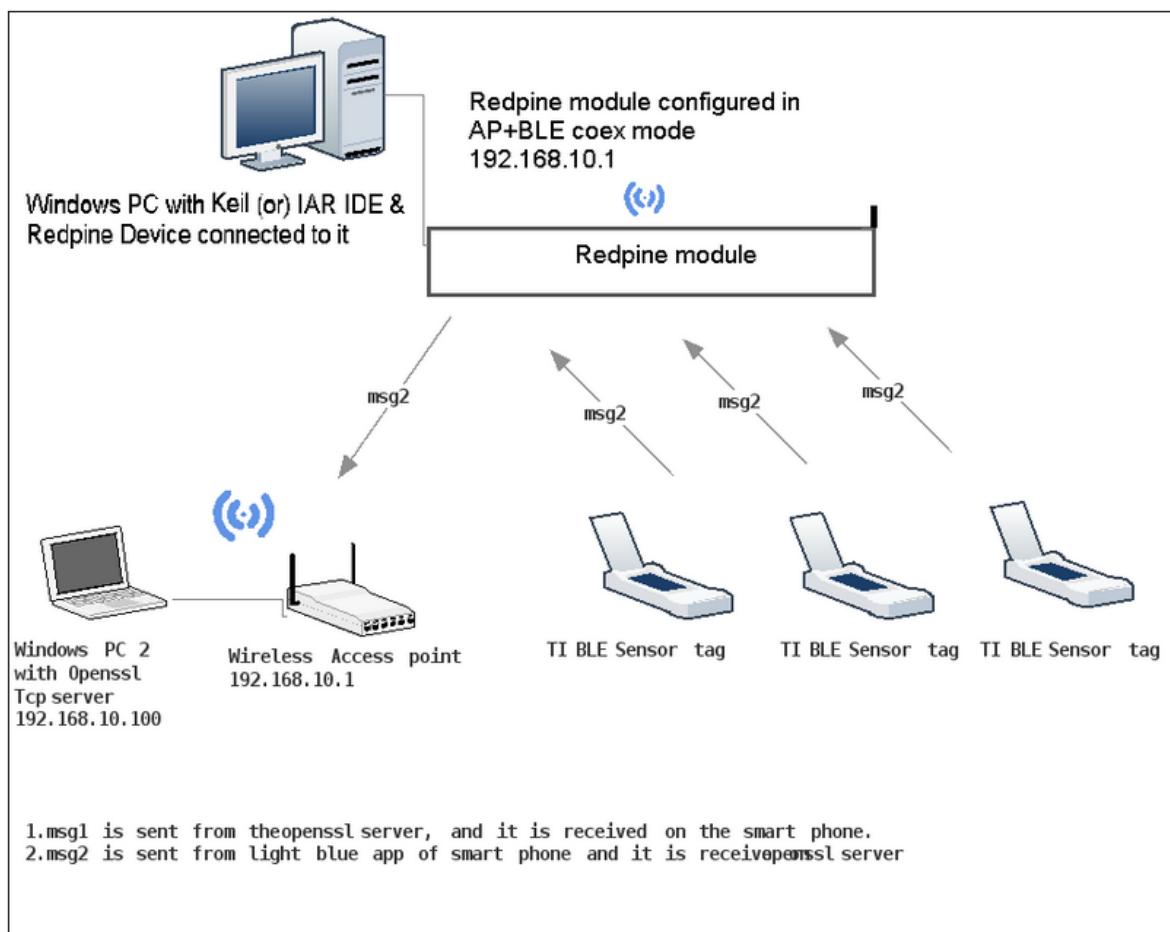
- Send data received by WLAN task and send to Smart phone

### 5.5.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- WLAN Access Point and a Windows PC with openssl support
- TI simple sensor tag or 3<sup>rd</sup> party BLE dongles



**Figure 1: Setup diagram**

### 5.5.3 Configuration and Execution of the Application

#### Configuring the Application

##### Configuring the WLAN task

1. Open **rsi\_wlan\_app.c** file and update/modify following macros,

**SSID** refers to the Access point to which user wants to connect.

**SECURITY\_TYPE** is the security type of the Access point.

**PSK** refers to the secret key if the Access point is configured in WPA/WPA2 security modes.

```
#define SSID                                "<ap_name>"  
#define SECURITY_TYPE                         <security-  
type>  
#define PSK                                    " "
```

Load the SSL CA- certificate using **rsi\_wlan\_set\_certificate** API after wireless initialization.

**Note:**

**rsi\_wlan\_set\_certificate** expects the certificate in the form of linear array. Python script is provided in the release package named "**certificate\_script.py**" in the following path "certificates" to convert the pem certificate into linear array.

Example: If the certificate is ca-cert.pem, give the command as

python certificate\_script.py ca-cert.pem

The script will generate cacert.pem, in which one linear array named *cacert* contains the certificate  
Enable/Disable DHCP mode

1 – Enables DHCP mode (gets the IP from DHCP server)

0 – Disables DHCP mode

```
#define DHCP_MODE
```

1

If DHCP mode is disabled, then change the following macros to configure static IP address

IP address to be configured to the device should be in long format and in little endian byte order.

Example: To configure "192.168.10.101" as IP address, update the macro **DEVICE\_IP** as **0x650AA8C0**.

```
#define DEVICE_IP
```

0x650AA8C0

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY
```

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order  
Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK	0x00FFFFFF
-----------------	------------

To establish TCP connection and transfer data to the remote socket configure the below macros. If SSL is enabled, open the socket with protocol type as 1.  
Internal socket port number.

#define DEVICE_PORT	5001
---------------------	------

Port number of the remote server

#define SERVER_PORT	5001
---------------------	------

IP address of the remote server

#define SERVER_IP_ADDRESS	0x650AA8C0
---------------------------	------------

Include rsi\_wlan\_app.c , rsi\_ble\_app.c and main.c files in the project, build and launch the application  
Open SSL server socket on remote machine

For example, to open SSL socket with port number 5001 on remote side, use the command as given below  
**openssl s\_server -accept<SERVER\_PORT> -cert <server\_certificate\_file\_path> -key <server\_key\_file\_path> -tls<tls\_version>**

**Example:** openssl s\_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1\_2

**Note:**

All the certificates are given in the release package

Enable/Disable power save  
1 – Enables Power save mode  
0 – Disables Power save mode

#define WLAN_POWER_SAVE	0
-------------------------	---

By default Power save is disabled.

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_CLIENT | TCP_IP_TOTAL_SOCKETS_1 |
TCP_IP_FEAT_SSL )
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENTION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Configuring the BLE Application

1. Open ***rsi\_ble\_app.c*** file and update/modify following macros,
  - RSI\_BLE\_NEW\_SERVICE\_UUID - The attribute value of the newly created service. Ex: 0xAABB
  - RSI\_BLE\_ATTRIBUTE\_1\_UUID - The attribute type of the first attribute under this Service. Ex: 0x1AA1
  - RSI\_BLE\_ATTRIBUTE\_2\_UUID - The attribute type of the second attribute under this Service. Ex: 0x1BB1
  - RSI\_BLE\_MAX\_DATA\_LEN - Maximum length of the attribute data(limited to max of 20 bytes)
  - RSI\_BLE\_APP\_DEVICE\_NAME - Name of the Redpine device to appear during Scanning by peer devices.
  - BLE\_PS\_ENABLE – To Enable/Disable power save.
  - MAX\_NUM\_OF\_SLAVES – Maximum number of slaves.
  - RSI\_BLE\_DEV\_1\_ADDR – Address of the 1<sup>st</sup> peer device to connect.
  - RSI\_BLE\_DEV\_2\_ADDR – Address of the 2<sup>nd</sup> peer device to connect.
  - RSI\_BLE\_DEV\_3\_ADDR – Address of the 3<sup>rd</sup> peer device to connect.

### **Update the BLE configuration file to enable Multiple slaves and to configure number of slaves:**

2. Open ***rsi\_ble\_config.h*** file and update/modify following macros
  - a. RSI\_BLE\_MAX\_NBR\_SLAVES – Maximum number of slaves supported by firmware which is used in Operemode feature bitmap

#### **Note:**

- The current document explains in refer to 3 slaves but the application has max of 8 slaves.
- The current application consists 3 slaves, 20 attributes, 5 services in BLE and open SSL feature in WiFi. If incase to increase slaves, services or attributes please refer WiSeConnect\_TCPIP\_Feature\_Selection\_v1.x.x document for memory limitations.

- b. RSI\_BLE\_MAX\_NBR\_ATT\_REC – Maximum number of attribute records.
- c. RSI\_BLE\_MAX\_NBR\_ATT\_SERV – Maximum number of services.

Following are the **non-configurable** macros in the application.

- RSI\_BLE\_ATT\_PROPERTY\_READ – Used to set read property to an attribute value.
- RSI\_BLE\_ATT\_PROPERTY\_WRITE - Used to set write property to an attribute value.

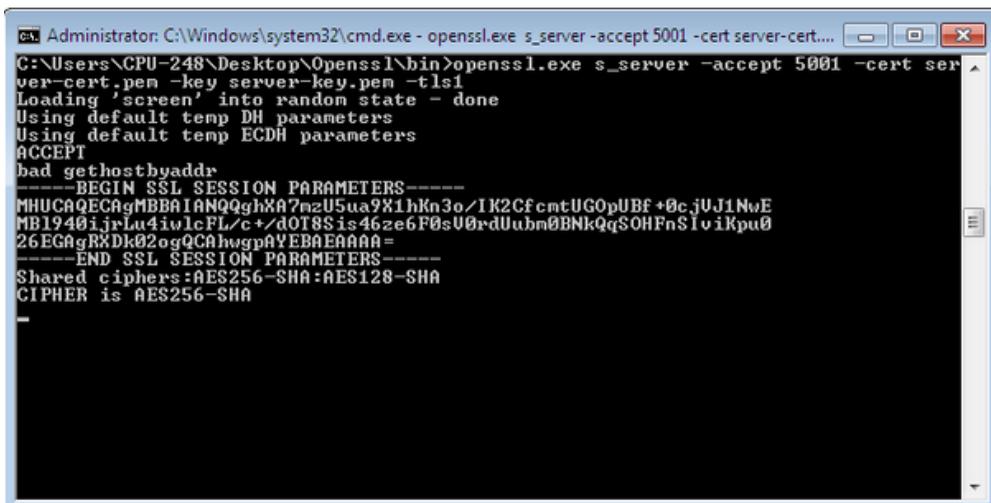
- RSI\_BLE\_ATT\_PROPERTY\_NOTIFY - Used to set notify property to an attribute value.
- RSI\_BLE\_CHAR\_SERV\_UUID - The attribute type of the characteristics to be added in a service. Ex: 0x2803
- RSI\_BLE\_CLIENT\_CHAR\_UUID - The attribute type of the client characteristics descriptor to be added in a service characteristic. Ex: 0x2902
- BT\_GLOBAL\_BUFF\_LEN – Number of bytes required for the Application and the Driver.

## Executing the Application

1. Connect Redpine device to the Windows PC running KEIL or IAR IDE
2. Build and launch the application.
3. Advertise 3<sup>rd</sup> party TI sensor tags.
4. After the program gets executed, then Redpine BLE is in scanning state and it creates a connection with TI sensor tag. Based on MAX\_NUM\_OF\_SLAVES Redpine device will goes to scanning state.
5. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
6. In Windows PC2 which is connected to AP through LAN, Download the Openssl package from above mentioned link and run SSL server by giving following command:
7. **Openssl.exe s\_server -accept<SERVER\_PORT> -cert <server\_certificate\_file\_path> -key <server\_key\_file\_path> -tls<tls\_version>**

**Example: openssl.exe s\_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1**

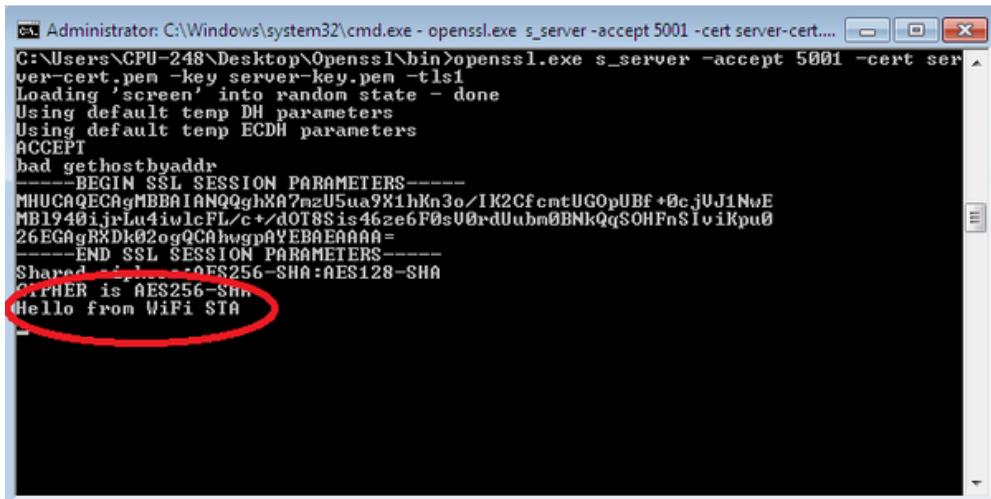
8. After the program gets executed, Redpine BLE is in Scanning state and WLAN connects to Access Point and establishes SSL connectivity with SSL server opened on Windows PC1. Please refer the given below image for connection establishment at windows PC1,



```
Administrator: C:\Windows\system32\cmd.exe - openssl.exe s_server -accept 5001 -cert server-cert.....
C:\Users\CPU-248\Desktop\Openssl\bin>openssl.exe s_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1
Loading 'screen' into random state - done
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
bad gethostbyaddr
-----BEGIN SSL SESSION PARAMETERS-----
MIUCAQECAgMBBAIAHQQghXA7mzU5ua9X1hKn3o/IK2CfcntUGOpUBf+0c.jUJ1NwE
MB19401jrLu4iuIcFL/c+dOT8Si46ze6F0sU0rdUubn0BNkQqSOHFnS1vIKpu0
26EGAgRXDk02agQCAhugnAYEBAEAAA=
-----END SSL SESSION PARAMETERS-----
Shared ciphers: AES256-SHA : AES128-SHA
CIPHER is AES256-SHA
-
```

9. After BLE connection is established, Sending start notification command to TI sensor tag. And start sending a notification from the TI sensor tag to Redpine BLE. Observe this notification in the PC connected via SSL socket with Redpine WLAN.

10. Now from SSL server (windows PC1), send a message (Ex: "Hello from WiFi STA") to Redpine device. Redpine device forwards the received message from SSL server to remote BTLE device which is connected to Redpine BTLE device over BTLE protocol. User can observe the message notification on attribute UUID **RSI\_BLE\_ATTRIBUTE\_2\_UUID(Ex: 0x1BB1)** in BTLE scanner app.



```
Administrator: C:\Windows\system32\cmd.exe - openssl.exe s_server -accept 5001 -cert server-cert...
C:\Users\CPU-248\Desktop\Openssl\bin>openssl.exe s_server -accept 5001 -cert server-cert...
ver-cert.pem -key server-key.pem -tls1
Loading 'screen' into random state - done
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
bad gethostbyaddr
-----BEGIN SSL SESSION PARAMETERS-----
MHUCAQECAgMBBAIANQqghXA7mzU5ua9X1hKn3o/IK2CfemtUGOpUBF+0cjUJ1NwE
MB1940ijrLu4iwlcFL/c+/dOT8Si46ze6F0sU0rdUuhm0BNkQqSOHFnS1viKpu0
26EGAgRXDk02ogQCAhwgnAYEBEAQAA=
-----END SSL SESSION PARAMETERS-----
Shared cipher is AES256-SHA:AES128-SHA
CurRTER is AES256-SHA
Hello from WiFi STA
```

rsi\_ble\_app\_send\_to\_wlan() function defined in rsi\_wlan\_app.c to send message from BLE task to WLAN task.  
With the help of wlan task, message is transferred to PC.

## 5.6 WLAN-STATION BLE THROUGHPUT Example

### 5.6.1 Introduction

This example is applicable to WiSeConnect™ and WiSeMCU™ parts. For simplicity, this document refers to WiSeConnect, but all discussion applies to both WiSeConnect and WiSeMCU parts. The feature(s) used in this example may or may not be available in your part. Refer to the product datasheet to verify the features available in your part.

### Application Overview

#### Overview

The coex application demonstrates throughput measurement of wifi while BLE is in connection.

#### Sequence of Events

#### WLAN Task

This application can be used to configure Redpine module in UDP client / server or TCP client / server. To measure throughput, following configuration can be applied.

- To measure UDP Tx throughput, module should configured as UDP client.
- To measure UDP Rx throughput, module should configured as UDP server.
- To measure TCP Tx throughput, module should configured as TCP client.
- To measure TCP Rx throughput, module should configured as TCP server.

## BLE Task

This Application explains user how to:

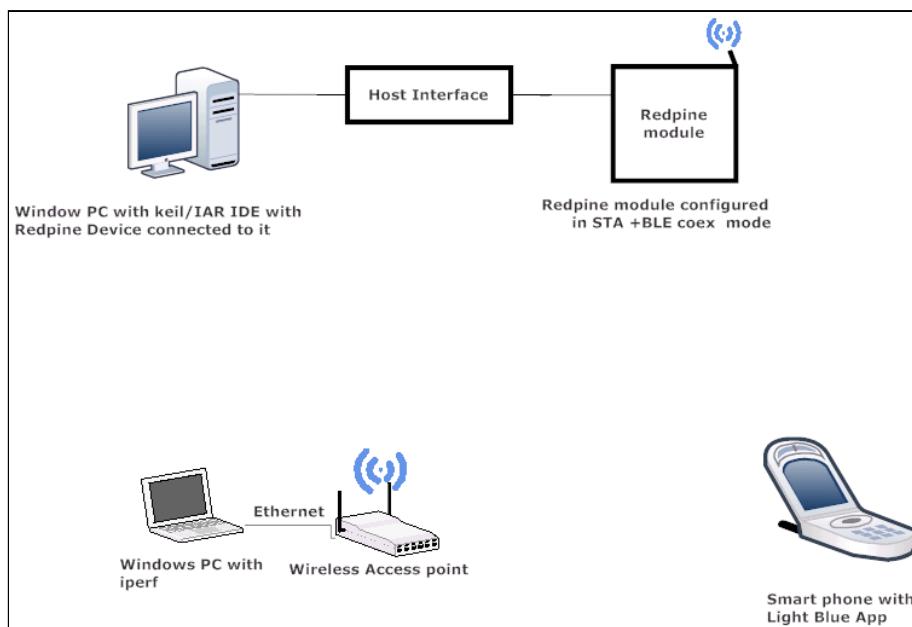
- Create chat service
- Configure device in advertise mode
- Connect from Smart phone
- Receive data sent by Smart phone

## Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

## WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Smart phone/tablet with BLE Application (Ex: GATT client application)
- WLAN Access Point and a Windows PC with iperf application



**Figure 1: Setup to demonstrate WLAN Station BT bridge application**

## Description

The coex application has WLAN and BLE tasks and acts as an interface between Smartphone and PC. Smartphone interacts with BLE task, while Both PC and Redpine WLAN would be connected to a Wireless Access Point, thus both

are connected together wirelessly interacts with WLAN task. When Smartphone connects and sends message to Redpine device, BT task accepts .Similarly, data transfer will happen for Station between AP.

### Details of the Application

Redpine WLAN acts as a Station and connects to an Access Point  
Redpine BLE acts as a Slave device.

- The WLAN task (running in Redpine device) mainly includes following steps.
  1. Connects to a Access Point
  2. Exchanges data over socket with the peer(Windows PC)
- The BLE task (running in Redpine device) mainly includes following steps.
  1. Creates chat service
  2. Configures the device in advertise mode and connectable mode.

WLAN and BLE tasks forever run in the application to serve the asynchronous events

### 5.6.2 Configuration and Execution of the Application

#### Configuring the Application

##### Configuring the WLAN task

##### Configuring the Application

1. Open **rsi\_wlan\_app.c** file and update / modify the following macros,  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which AP would be started

```
#define CHANNEL_NO
```

11

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities.  
Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE
```

RSI\_OPEN

**PSK** refers to the secret key if the Access point is to be configured in WPA/WPA2 security modes.

```
#define PSK
```

"<psk>"

Enable / Disable DHCP mode  
1-Enables DHCP mode (gets the IP from DHCP server)  
0-Disables DHCP mode

```
#define DHCP_MODE
```

<dhcp mode>

**To configure static IP address**

IP address to be configured to the device should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP
```

0X010AA8C0

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY
```

0X010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK
```

0X00FFFFFF

2. To establish UDP/TCP connection and transfer/receive data to the remote socket configure the below macros  
Internal device port number

```
#define PORT_NUM
```

<Local\_port>

Port number of the remote server

```
#define SERVER_PORT  
<Remote_port_num>
```

IP address of the remote server

```
#define SERVER_IP_ADDRESS
```

0X640AA8C0

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 8000
```

Application can use receive buffer size of 1400

```
#define BUFF_SIZE 1400
```

Application can select throughput type as UDP Tx, UDP Rx, TCP Tx or TCP Rx. Following is macro need to use.

```
#define THROUGHPUT_TYPE UDP_TX
```

Following is macro used for throughput type selection

#define UDP_TX	0
#define UDP_RX	1
#define UDP_TX	2
#define UDP_TX	3

**Note:**

In AP mode, configure same IP address for both DEVICE\_IP and GATEWAY macros.

3. Open ***rsi\_wlan\_config.h*** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_CLIENT | FEAT_AGGRAGATION )
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENTION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Configuring the BLE Application

1. Open **rsi\_ble\_app.c** file and update/modify following macros,  
**RSI\_BLE\_NEW\_SERVICE\_UUID** refers to the attribute value of the newly created service.

```
#define RSI_BLE_NEW_SERVICE_UUID 0xAABB
```

**RSI\_BLE\_ATTRIBUTE\_1\_UUID** refers to the attribute type of the first attribute under this service (**RSI\_BLE\_NEW\_SERVICE\_UUID**).

```
#define RSI_BLE_ATTRIBUTE_1_UUID 0x1AA1
```

**RSI\_BLE\_ATTRIBUTE\_2\_UUID** refers to the attribute type of the second attribute under this service (**RSI\_BLE\_NEW\_SERVICE\_UUID**).

```
#define RSI_BLE_ATTRIBUTE_2_UUID 0x1BB1
```

**RSI\_BLE\_MAX\_DATA\_LEN** refers to the Maximum length of the attribute data.

```
#define RSI_BLE_MAX_DATA_LEN 20
```

**RSI\_BLE\_APP\_DEVICE\_NAME** refers name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_APP_DEVICE_NAME  
"WLAN_BLE_SIMPLE_CHAT"
```

Following are the **non-configurable** macros in the application.

**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.

```
#define RSI_BLE_CHAR_SERV_UUID 0x2803
```

**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

```
#define RSI_BLE_CLIENT_CHAR_UUID 0x2902
```

**RSI\_BLE\_ATT\_PROPERTY\_READ** is used to set the READ property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_READ 0x02
```

**RSI\_BLE\_ATT\_PROPERTY\_WRITE** is used to set the WRITE property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_WRITE 0x08
```

**RSI\_BLE\_ATT\_PROPERTY\_NOTIFY** is used to set the NOTIFY property to an attribute value.

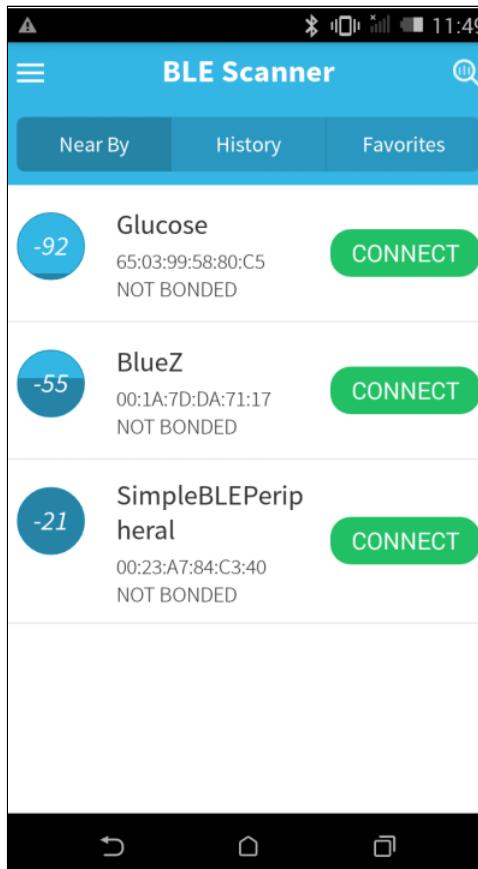
```
#define RSI_BLE_ATT_PROPERTY_NOTIFY 0x10
```

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver.

```
#define BT_GLOBAL_BUFF_LEN 10000
```

## Executing the coex Application

1. Connect WiSeConnect device to the Windows PC running KEIL or IAR IDE
2. Build and launch the application.
3. After the program gets executed, Redpine BLE is in Advertising state and WLAN has established has the configuration given
4. Open a BLE scanner App in the Smartphone and do the Scan.



5. In the App, Redpine module device will appear with the name configured in the macro **RSI\_BLE\_APP\_SIMPLE\_CHAT** (Ex: "WLAN\_BLE\_SIMPLE\_CHAT") or sometimes observed as Redpine device as internal name "**SimpleBLEPeripheral**".
6. Initiate BLE connection from the App.
7. After BT connection is established, send a message from the App to Redpine BLE
8. To measure throughput, following configuration can be applied.
  - a. To measure UDP Tx throughput, module should be configured as UDP client. Open UDP server at remote port

```
iperf.exe -s -u -p <SERVER_PORT> -i 1
```

- b. To measure UDP Rx throughput, module should be configured as UDP server. Open UDP client at remote port

```
iperf.exe -c <Module_IP> -u -p <Module_Port> -i 1 -b <Bandwidth>
```

- c. To measure TCP Tx throughput, module should be configured as TCP client. Open TCP server at remote port.

```
iperf.exe -s -p <SERVER_PORT> -i 1
```

- d. To measure TCP Rx throughput, module should be configured as TCP server. Open TCP client at remote port

```
iperf.exe -c <Module_IP> -p <module_PORT> -i 1
```

9. To measure throughput, following configuration can be applied.
10. Build and launch the application.
11. After the program gets executed, the device would be connected to Access point having the configuration same as that of in the application and get.
12. The Device which is configured as UDP / TCP server / client will connect to iperf server / client and sends / receives data continuously. It will print the throughput per second.

## 5.7 WLAN-STATION Standby BLE Advertising Power Save Example

### 5.7.1 Application Overview

The coex application demonstrates a procedure to measure power numbers in WiseMCU coex mode with wlan standby and ble advertising powersave.

In this coex application, Redpine BTLE device starts advertising and issue connected power save command to NWP. In parallel Redpine WiFi interface connects with an Access Point in station mode and issue connected power save command to NWP.

After proper coex power save mode selection then power save command go to NWP module, with successful power save enable triggering M4 to go sleep.

### Sequence of Events

#### WLAN Task

This Application explains user how to:

- Create Redpine device as Station
- Connect Redpine station to remote Access point
- Getting IP using DHCP/Static.
- Issuing powersave command
- After confirming power save enabled then triggering M4 to go sleep.

#### BLE Task

This Application explains user how to:

- Configure device in advertise mode
- Configure device in power save profile mode 2 .

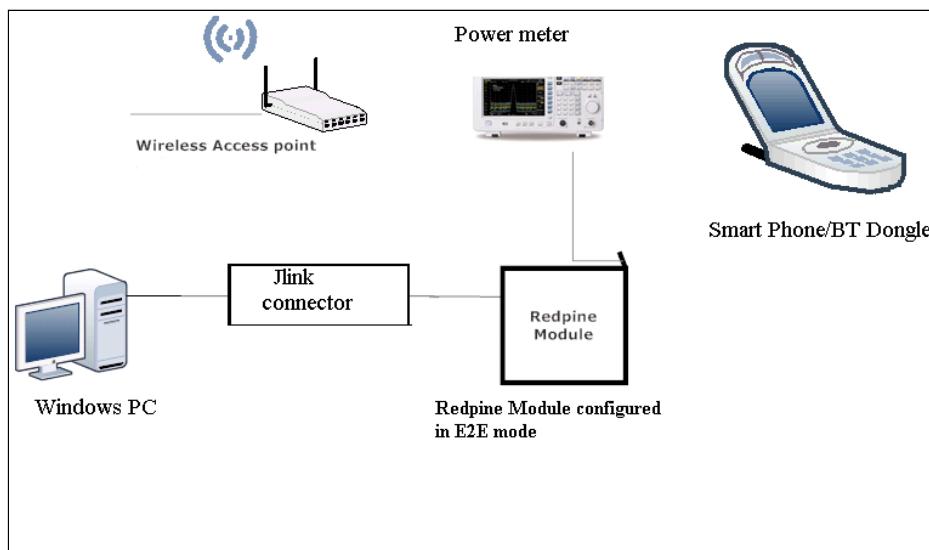
### 5.7.2 Application Setup

The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU based Setup Requirements

- Windows PC with KEIL or IAR IDE
- Redpine module
- Access point
- Jlink segger(M4 debugger)

- Smart phone/Dongle
- Power meter/ DMM



**Figure 1: Setup to demonstrate WLAN standby BLE advertising power save application**

### 5.7.3 Configuration and Execution of the Application

#### Configuring the Application

##### Configuring the WLAN task

1. Open **rsi\_wlan\_app.c** file and update/modify the following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID "ap  
name">"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels

```
#define CHANNEL_NO 0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE  
RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK  
<psk>"
```

**To configure IP address:**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE
```

1

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set DHCP\_MODE macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP  
0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY  
0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK  
0X00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT )
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENSION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Configuring the BLE Application

1. **RSI\_BLE\_NEW\_SERVICE\_UUID** refers to the attribute value of the newly created service.

```
#define RSI_BLE_NEW_SERVICE_UUID
0xAABB
```

**RSI\_BLE\_ATTRIBUTE\_1\_UUID** refers to the attribute type of the first attribute under this service  
**(RSI\_BLE\_NEW\_SERVICE\_UUID)**

```
#define RSI_BLE_ATTRIBUTE_1_UUID
0x1AA1
```

**RSI\_BLE\_ATTRIBUTE\_2\_UUID** refers to the attribute type of the second attribute under this service  
**(RSI\_BLE\_NEW\_SERVICE\_UUID)**

```
#define RSI_BLE_ATTRIBUTE_2_UUID
0x1BB1
```

**RSI\_BLE\_MAX\_DATA\_LEN** refers to the Maximum length of the attribute data.

```
#define RSI_BLE_MAX_DATA_LEN
```

20

**RSI\_BLE\_APP\_DEVICE\_NAME** refers name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_APP_DEVICE_NAME  
"WLAN_BLE_SIMPLE_CHAT"
```

Following are the **non-configurable** macros in the application.

**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.

```
#define RSI_BLE_CHAR_SERV_UUID  
0x2803
```

**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

```
#define RSI_BLE_CLIENT_CHAR_UUID  
0x2902
```

**RSI\_BLE\_ATT\_PROPERTY\_READ** is used to set the READ property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_READ 0x02
```

**RSI\_BLE\_ATT\_PROPERTY\_WRITE** is used to set the WRITE property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_WRITE 0x08
```

**RSI\_BLE\_ATT\_PROPERTY\_NOTIFY** is used to set the NOTIFY property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_NOTIFY 0x10
```

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver.

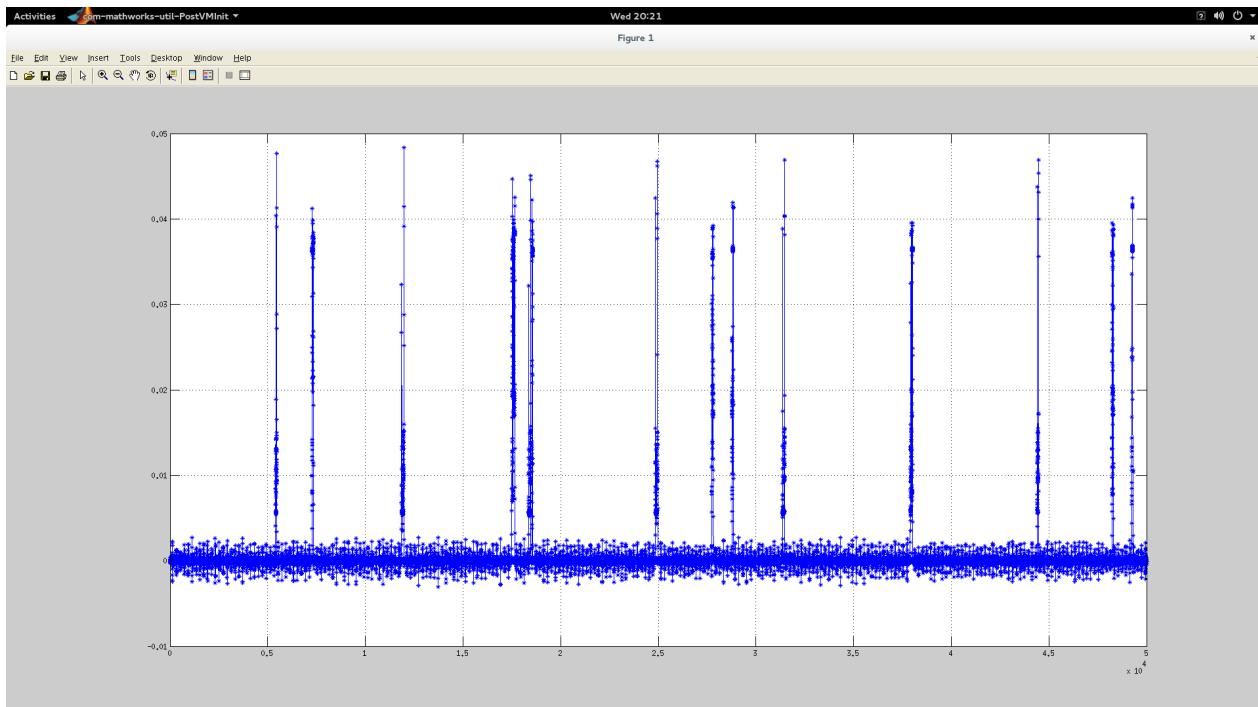
```
#define BT_GLOBAL_BUFF_LEN  
10000
```

2. Open **rsi\_ble\_config.h** file and update/modify following macros,

```
#define RSI_BLE_ADV_INT_MIN 0x400
#define RSI_BLE_ADV_INT_MAX 0x400
```

## Executing the Application

1. Make sure configuration settings as mentioned above before to compile&run the Application.
2. Configure the access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
3. After successful connection with access point ,module gets IP either using DHCP or Static based on configuration settings.
4. Trigger NWP to sleep with power mode 2 with ram retention.
5. In parallel BLE advertising continuously happen and whenever issue a connect event then BLE connection will happen then trigger NWP to sleep with power mode 2 with ram retention.
6. After proper coex power save mode selection then power save command go to the module.
7. M4 sleep trigger will happen after successful power save enable of NWP.
8. Note down power measurement as shown below.



## 5.8 WLAN-STATION Standby BLE connected Power Save Example

### 5.8.1 Application Overview

The coex application demonstrates a procedure to measure power numbers in WiseMCU coex mode with wlan standby and ble connected powersave.

In this coex application, Redpine BTLE device connects with remote BTLE device (ex:Smart phone/Dongle) and issue connected power save command to NWP .In parallel Redpine WiFi interface connects with an Access Point in station mode and issue connected power save command to NWP.

After proper coex power save mode selection then power save command go to NWP module, with successful power save enable triggering M4 to go sleep.

## Sequence of Events

### WLAN Task

This Application explains user how to:

- Create Redpine device as Station
- Connect Redpine station to remote Access point
- Getting IP using DHCP/Static.
- Issuing powersave command
- After confirming power save enabled then triggering M4 to go sleep.

### BLE Task

This Application explains user how to:

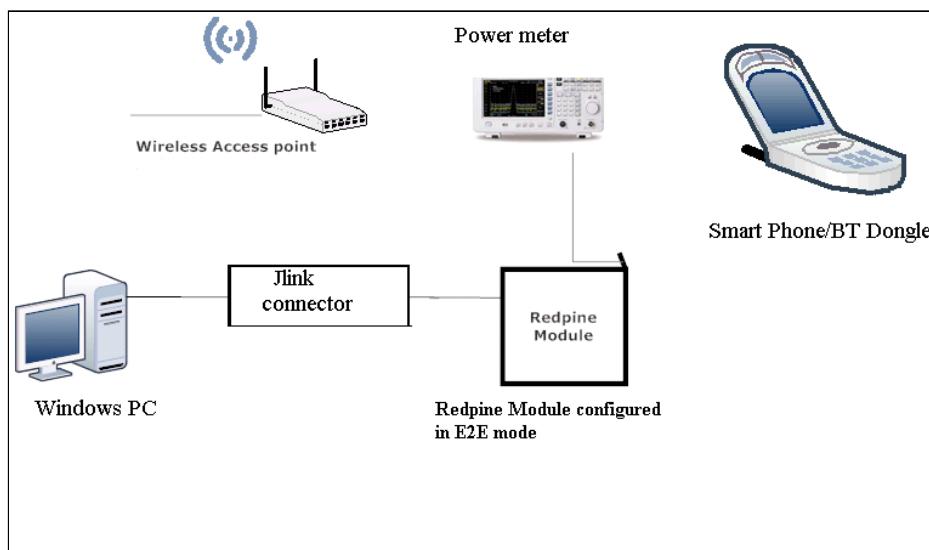
- Configure device in advertise mode
- Connect from Smart phone/Dongle
- Configure device in power save profile 2 .

### 5.8.2 Application Setup

The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU based Setup Requirements

- Windows PC with KEIL or IAR IDE
- Redpine module
- Access point
- Jlink(M4 debugger)
- Smart phone/Dongle
- Power meter/ DMM



**Figure 1: Setup to demonstrate WLAN standby BLE connected power save application**

### 5.8.3 Configuration and Execution of the Application

#### Configuring the Application

##### Configuring the WLAN task

1. Open **rsi\_wlan\_app.c** file and update/modify the following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap  
name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE    RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK             "<psk>"
```

**To configure IP address:**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE       1
```

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte

order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP          0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY          0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK          0x00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE           RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS          RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP (TCP_IP_FEAT_DHCPV4_CLIENT )
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENSION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Configuring the BLE Application

1. Open **rsi\_ble\_app.c** file and update/modify following macros,  
**RSI\_BLE\_NEW\_SERVICE\_UUID** refers to the attribute value of the newly created service.

```
#define RSI_BLE_NEW_SERVICE_UUID          0xAABB
```

**RSI\_BLE\_ATTRIBUTE\_1\_UUID** refers to the attribute type of the first attribute under this service  
(**RSI\_BLE\_NEW\_SERVICE\_UUID**).

```
#define RSI_BLE_ATTRIBUTE_1_UUID 0x1AA1
```

**RSI\_BLE\_ATTRIBUTE\_2\_UUID** refers to the attribute type of the second attribute under this service (**RSI\_BLE\_NEW\_SERVICE\_UUID**).

```
#define RSI_BLE_ATTRIBUTE_2_UUID 0x1BB1
```

**RSI\_BLE\_MAX\_DATA\_LEN** refers to the Maximum length of the attribute data.

```
#define RSI_BLE_MAX_DATA_LEN 20
```

**RSI\_BLE\_APP\_DEVICE\_NAME** refers name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BLE_APP_DEVICE_NAME  
"WLAN_BLE_SIMPLE_CHAT"
```

Following are the **non-configurable** macros in the application.

**RSI\_BLE\_CHAR\_SERV\_UUID** refers to the attribute type of the characteristics to be added in a service.

```
#define RSI_BLE_CHAR_SERV_UUID 0x2803
```

**RSI\_BLE\_CLIENT\_CHAR\_UUID** refers to the attribute type of the client characteristics descriptor to be added in a service.

```
#define RSI_BLE_CLIENT_CHAR_UUID 0x2902
```

**RSI\_BLE\_ATT\_PROPERTY\_READ** is used to set the READ property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_READ 0x02
```

**RSI\_BLE\_ATT\_PROPERTY\_WRITE** is used to set the WRITE property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_WRITE 0x08
```

**RSI\_BLE\_ATT\_PROPERTY\_NOTIFY** is used to set the NOTIFY property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_NOTIFY
```

0x10

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver.

```
#define BT_GLOBAL_BUFF_LEN
```

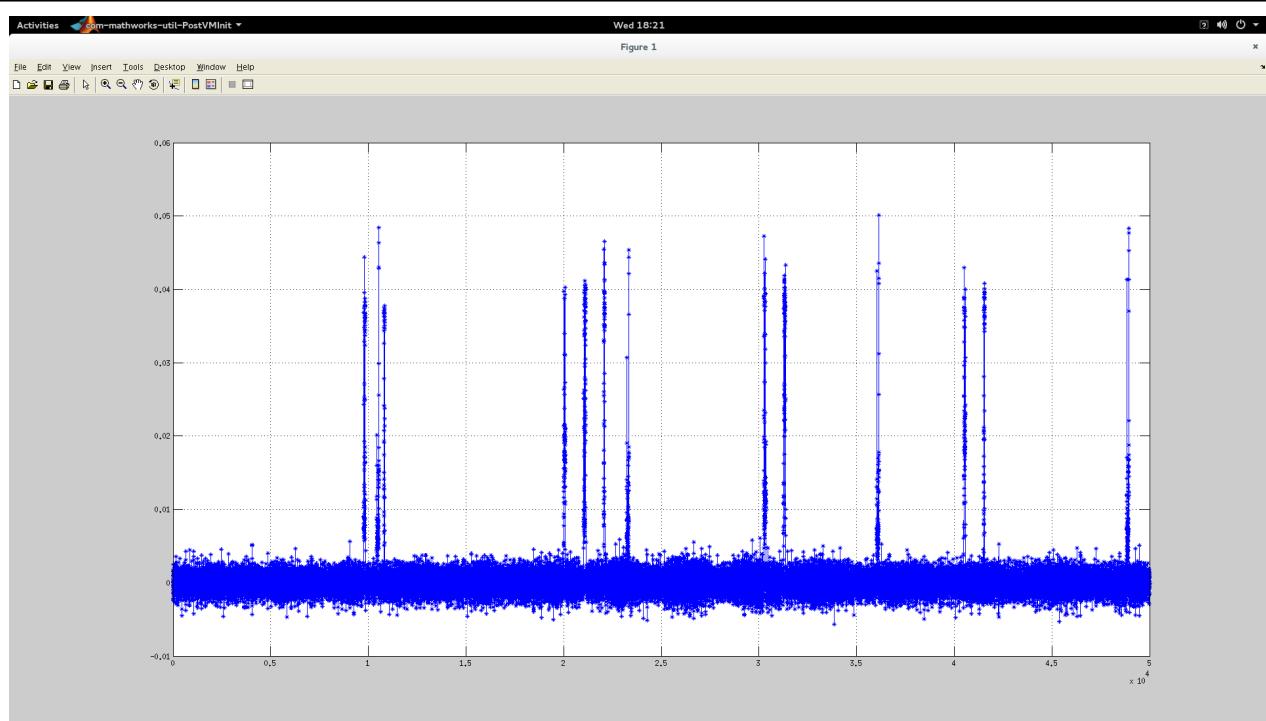
10000

2. Open **rsi\_ble\_config.h** file and update/modify following macros,

```
#define RSI_BLE_PWR_INX  
8  
#define RSI_BLE_PWR_SAVE_OPTIONS  
0  
#define RSI_DUTY_CYCLING  
0
```

## Executing the Application

1. Make sure configuration settings as mentioned above before to compile&run the Application.
2. Configure the access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
3. After successful connection with access point ,module gets IP either using DHCP or Static based on configuration settings.
4. Trigger NWP to sleep with power mode 2 with ram retention.
5. In parallel BLE advertising continuously happen and whenever issue a connect event then BLE connection will happen then trigger NWP to sleep with power mode 2 with ram retention.
6. After proper coex power save mode selection then power save command go to the module.
7. M4 sleep trigger will happen after successful power save enable of NWP.
8. Note down power measurement as shown below.



---

## 6 WLAN BT

### 6.1 WLAN-AP BT Bridge Example

#### 6.1.1 Application Overview

The coex application demonstrates how information can be exchanged seamlessly using two wireless protocols (WLAN and BT) running in the same device.

#### Sequence of Events

#### WLAN Task

This Application explains user how to:

- Create Redpine device as an Access Point
- Connect stations to Redpine Access Point
- Receive TCP data sent by connected station and forward to BT task
- Send data received by BT task to connected station using TCP Protocol.

#### BT Task

This Application explains user how to:

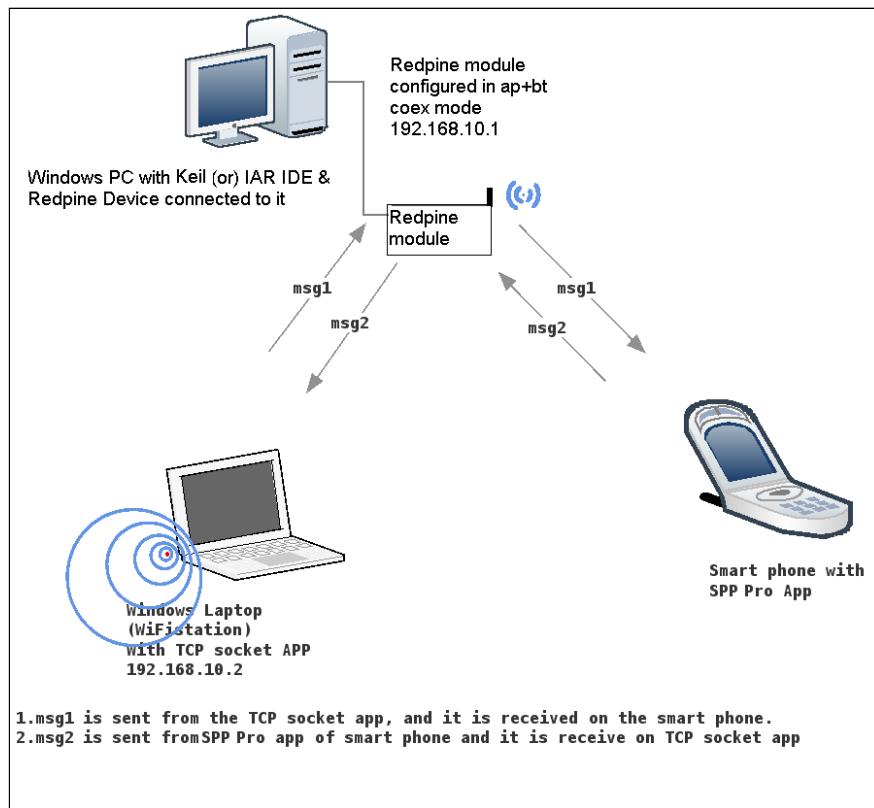
- Configure Redpine device to SPP profile mode
- Configure device in discoverable and connectable mode
- Establish SPP profile level connection with remote smart phone
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

#### 6.1.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Wi-Fi client device (PC) with TCP application



**Figure 1: Setup to demonstrate WLAN AP BT bridge Application**

## Description

The coex application has WLAN and BT tasks and acts as an interface between Smartphone and PC. Smartphone interacts with BT task, while PC Redpine module create an Access Point, remote PC (with WLAN station interface) connect to Redpine Access Point interacts with WLAN task.

When Smartphone connects and sends message to Redpine device, BT task accepts and sends to WLAN task, which in turn sends to PC. Similarly, when PC sends message to Redpine device, the message will be sent to Smartphone via BT task. Thus, messages can be seamlessly transferred between PC and Smartphone.

## Details of the Application

Redpine WLAN acts as Access Point and allow stations to connect Redpine BT acts as a slave device with SPP profile running in it, while Smart phone acts as a Master device with SPP profile running in it. Initially, proprietary Simple chat service is created with SPP profile (Redpine device) to facilitate message exchanges.

- The WLAN task (running in Redpine device) mainly includes following steps.
  1. Start up as Access point and allow stations to connect
  2. Establish TCP connection with Remote and performs data exchanges over TCP socket with the peer(PC)
- The BT task (running in Redpine device) mainly includes following steps.
  1. Creates chat service
  2. Configures the device in Discoverable mode and Connectable mode.

WLAN and BT tasks forever run in the application to serve the asynchronous events.

### 6.1.3 Configuration and Execution of the Application

#### Configuring the Application

##### Configuring the WLAN task

1. Open **rsi\_wlan\_ap\_app.c** file and update/modify following macros to start an Access-point.  
**SSID** refers to the name of the Access point .

```
#define SSID                                "<ap  
name>"
```

**CHANNEL\_NO** refers to the channel in which AP would be started

```
#define CHANNEL_NO  
<channel_no>
```

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities

```
#define SECURITY_TYPE  
<security_type>
```

**ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports Open, TKIP, CCMP methods

```
#define ENCRYPTION_TYPE  
<encryption_type>
```

**PSK** refers to the secret key if the Access point to be configured in WPA/WPA2 security modes.

```
#define PSK                                  "<psk>"
```

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons

```
#define BEACON_INTERVAL  
<beacon_interval>
```

To configure DTIM interval of the Access Point

```
#define DTIM_INTERVAL  
<dtim_interval>
```

To configure static IP address

IP address to be configured to the device should be in long format and in little endian byte order.  
Example: To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP  
0X010AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY  
0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK  
0x00FFFFFF
```

To establish TCP connection and transfer data to the remote socket, configure the below macros. Internal socket port number.

```
#define DEVICE_PORT 5001
```

Port number of the remote server

```
#define SERVER_PORT 5001
```

IP address of the remote server

```
#define SERVER_IP_ADDRESS  
0x020AA8C0
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 10000
```

2. Open TCP server socket on remote peer (PC).  
User can use Test TCP server application from the below link to create TCP Server on remote peer <http://sourceforge.net/projects/sockettest/files/latest/download>  
For example, user can open TCP server socket with port number 5001 on remote peer using the Test TCP server Application.
3. Include rsi\_wlan\_ap\_app.c, rsi\_bt\_app.c and main.c files in the project, build and launch the application.
4. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_SERVER | TCP_IP_TOTAL_SOCKETS_1)
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENTION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Configuring the BT Application

1. Edit the **rsi\_bt\_app.c** file and update/modify following macros,
2. Configure the below macros in the Application file.
  - RSI\_BT\_LOCAL\_NAME - Name of the Wyzbee (Master) device
  - PIN\_CODE - Four byte string required for pairing process.

Following are the **non-configurable** Macros in the Application file.

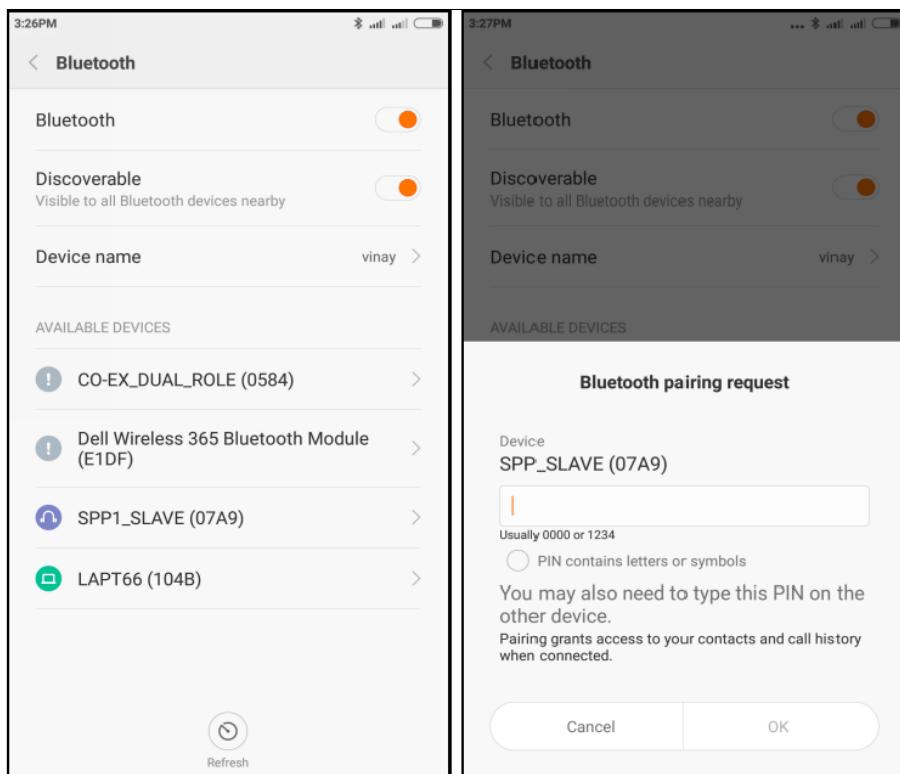
1. BT\_GLOBAL\_BUFF\_LEN – Number of bytes required for the Application and the Driver.
2. RSI\_APP\_EVENT\_CONNECTED - Event number to be set on connection establishment.
3. RSI\_APP\_EVENT\_DISCONNECTED - Event number to be set on disconnection.
4. RSI\_APP\_EVENT\_PINCODE\_REQ - Event number to be set on Pincode request for pairing.
5. RSI\_APP\_EVENT\_LINKKEY\_SAVE - Event number to be set on link key save.
6. RSI\_APP\_EVENT\_AUTH\_COMPLT - Event number to be set on authentication complete.
7. RSI\_APP\_EVENT\_LINKKEY\_REQ - Event number to be set on link key request for connection.
8. RSI\_APP\_EVENT\_SPP\_CONN - Event number to be set on SPP connection.
9. RSI\_APP\_EVENT\_SPP\_DISCONN - Event number to be set on SPP disconnection.

10. RSI\_APP\_EVENT\_SPP\_RX - Event number to be set on SPP data received from Master.

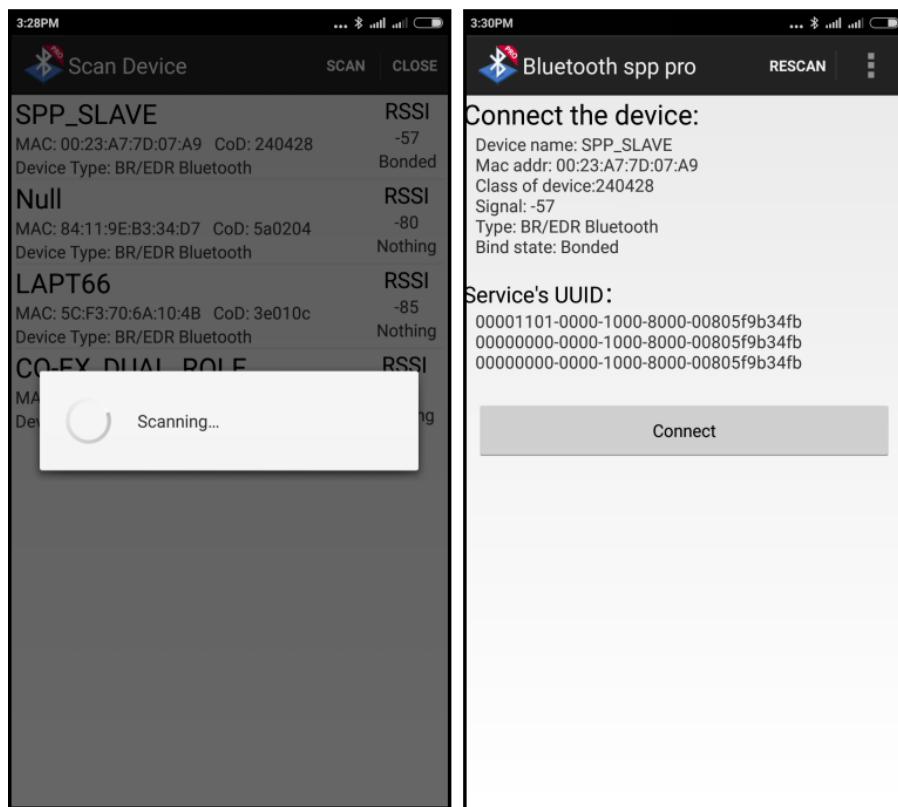
### Executing the coex Application

Connect Redpine device to the Windows PC running KEIL or IAR IDE.

1. Build and launch the application.
2. After the program gets executed, Redpine BT is in Discoverable state and WLAN will create an Access Point and wait for remote station to connect.
3. Remote PC can connect to device Access point and device will establish TCP connection with peer (PC).
4. Now initiate connection from the SPP App running in the Smartphone.

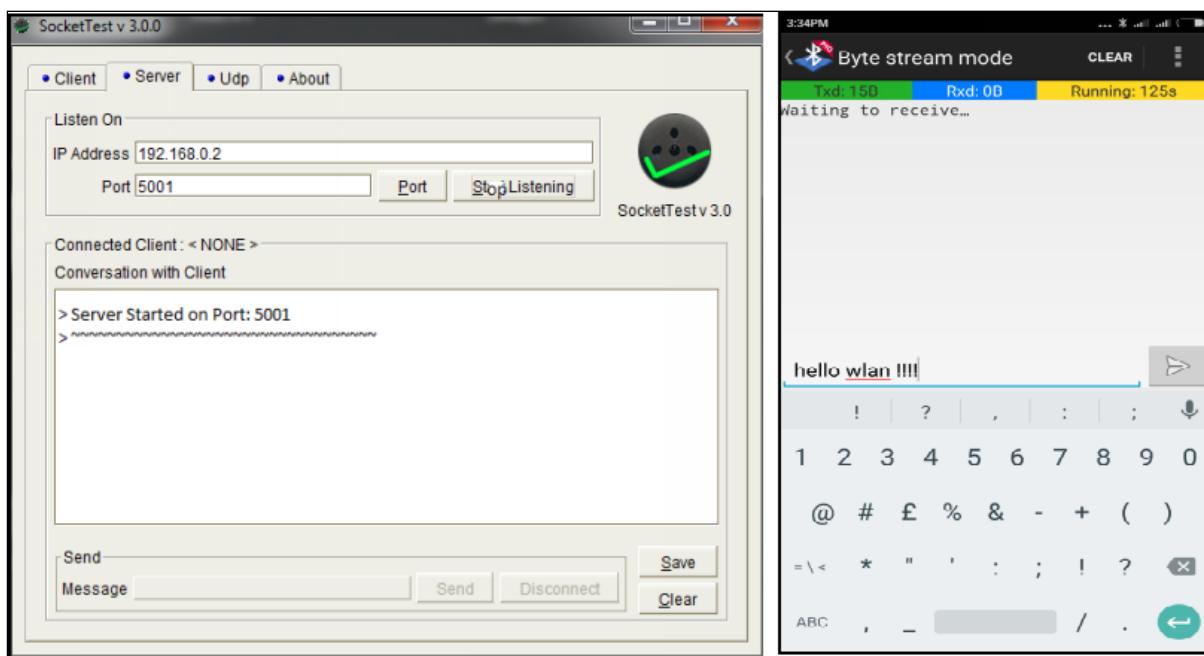


**Figure 2: Turn ON Bluetooth ,Scan for BT devices and Pair with the Rdpine BT device**



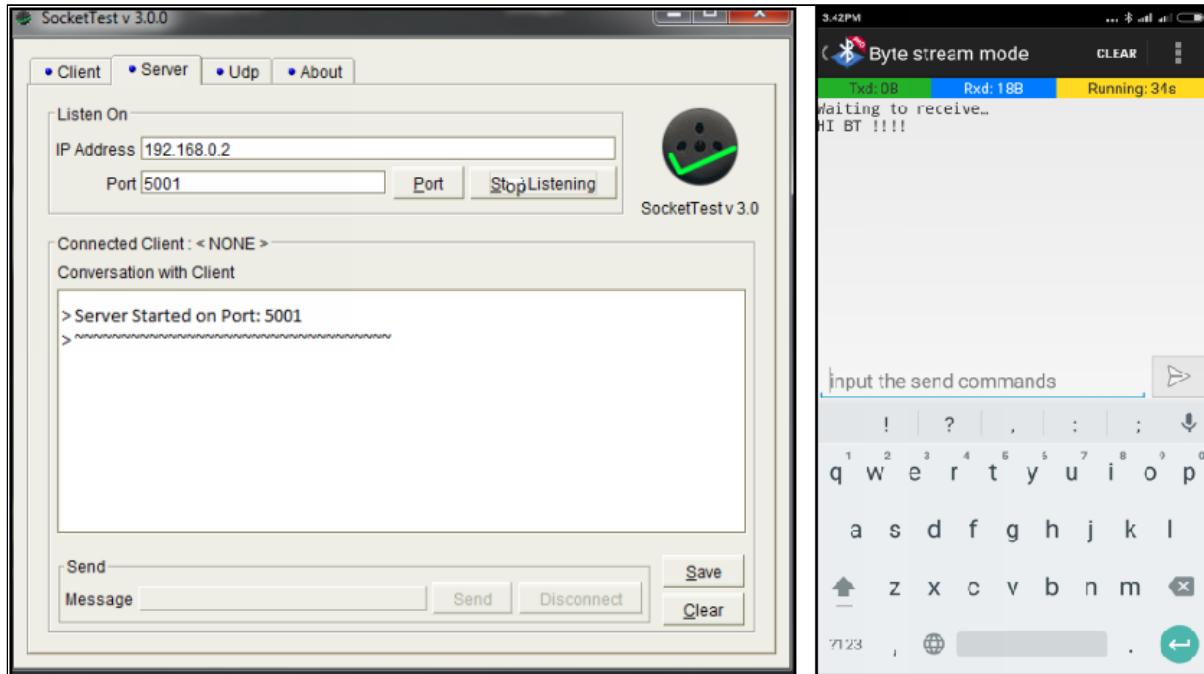
**Figure 3: Scan using SPP pro App and get connected to the Peripheral**

5. In the App, Redpine BT would appear with the name configured in the macro **RSI\_BT\_LOCAL\_NAME**.
6. After BT SPP connection is established, send a message from the App to Redpine BT. Observe this message on the PC (Socket Test GUI) connected via TCP socket with Redpine WLAN.



**Figure 4: Sending Data from BT SPP pro APP and received on Wifi Station socket App**

7. Now, send a message from PC (Socket Test GUI) to Redpine WLAN via TCP socket and observe the same in the Smartphone.



**Figure 5: Sending Data from Wifi Client Socket App and received on BT SPP Pro App**

8. `rsi_bt_app_send_to_wlan()` function defined in `rsi_wlan_ap_app.c` to send message from BT task to WLAN task.

9. With the help of wlan task, message is transferred to PC.

---

10. Message from PC to WLAN task via socket and rsi\_wlan\_app\_send\_to\_bt() function defined in rsi\_bt\_app.c called asynchronously to send message from WLAN task to BT task. From BT task message transferred to smart phone.

## 6.2 WLAN-AP BT Bridge TCPIP Bypass Example

### 6.2.1 Application Overview

#### Overview

The coex application demonstrates how information can be exchanged seamlessly using two wireless protocols (WLAN and BT) running in the same device.

In this coex application, Redpine BT device connects with remote BTLE device (Smart Phone) and Redpine WiFi interface starts as an Access Point and allow stations (Windows laptop) to connect it.

The coex application has WLAN and BT tasks and acts as an interface between remote Smartphone BTLE device and connected WiFi Station. Smartphone interacts with BT task, while connected station interacts with WLAN task. When Smartphone connects and sends message to Redpine device, BT task accepts and sends to WLAN task, which in turn sends to connected station. Similarly, when PC sends message to Redpine device, the message will be sent to Smartphone via BT task.

Thus messages can be seamlessly transferred between PC and Smartphone.

#### Sequence of Events

#### WLAN Task

This Application explains user how to:

- Create Redpine device as an Access Point
- Connect stations to Redpine Access Point
- Receive UDP data sent by connected station and forward to BT task
- Send data received by BT task to connected station using UDP protocol

#### BT Task

This Application explains user how to:

- Configure Redpine device to SPP profile mode
- Configure device in discoverable and connectable mode
- Establish SPP profile level connection with remote smart phone
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

### 6.2.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU

- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Windows Laptop (STA) with UDP socket application

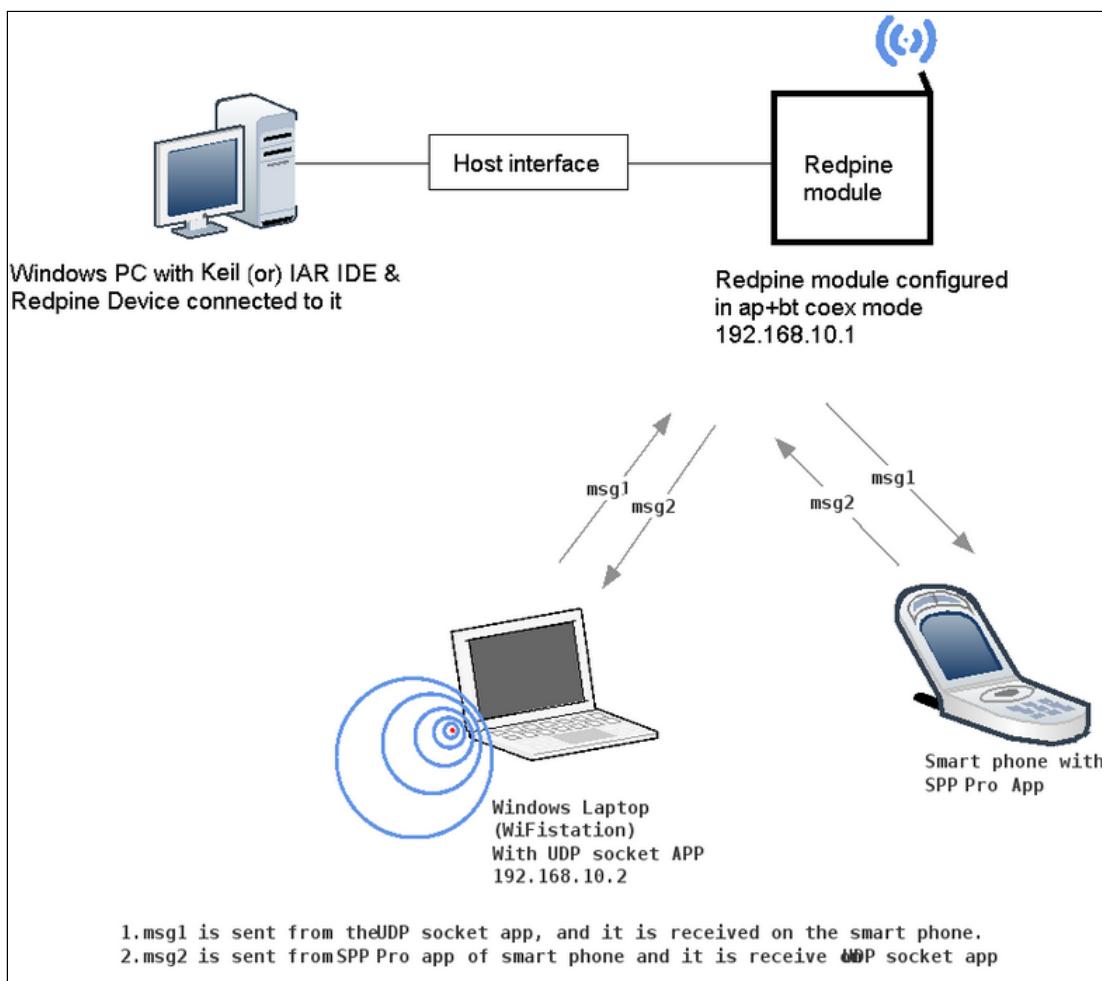
**Note:**

Download UDP socket application from below link,  
<http://sourceforge.net/projects/sockettest/files/latest/download>

- BT supported Smart phone with SPP application.

**Note:**

Install SPP Pro Application for BT SPP application.



**Figure 1: Setup to demonstrate WLAN AP BT bridge tcp ip bypass Application**

### 6.2.3 Configuration and Execution of the Application

#### Configuring the Application

##### Configuring the WLAN task

1. Open **rsi\_wlan\_ap\_app.c** file and update/modify following macros,

**SSID** refers to the name of the Access point.

```
#define SSID  
"REDPINE_AP"
```

**CHANNEL\_NO** refers to the channel in which AP would be started

```
#define CHANNEL_NO
```

11

**Note:**

Valid values for CHANNEL\_NO in 2.4GHz are 1 to 11 and 5GHZ are 36 to 48 and 149 to 165. In this example default configured band is 2.4GHz. So, if user wants to use 5GHz band then user has to set RSI\_BAND macro to 5GHz band in **rsi\_wlan\_config.h file**.

**SESECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities.  
Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE
```

RSI\_WPA2

**ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports OPEN, TKIP, CCMP methods.

Valid configuration is:  
**RSI\_CCMP** - For CCMP encryption  
**RSI\_TKIP** - For TKIP encryption  
**RSI\_NONE** - For open encryption

```
#define ENCRYPTION_TYPE
```

RSI\_CCMP

**PSK** refers to the secret key if the Access point to be configured in WPA/WPA2 security modes.

```
#define PSK "1234567890
```

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons in milliseconds. Allowed values are integers from 100 to 1000 which are multiples of 100.

```
#define BEACON_INTERVAL 100
```

**DTIM\_INTERVAL** refers DTIM interval of the Access Point. Allowed values are from 1 to 255.

```
#define DTIM_INTERVAL 4
```

**DEVICE\_PORT** port refers internal UDP server port number

```
#define DEVICE_PORT 5001
```

**REMOTE\_PORT** port refers remote UDP server port number

```
#define REMOTE_PORT 5001
```

**REMOTE\_IP\_ADDRESS** refers remote peer IP address to send data received by BT device.  
IP address of Wilress STA which is connected to WiSeConnect Access Point.

IP address should be in long format and in little endian byte order.

Example: To configure "192.168.10.2" as IP address, update the macro **REMOTE\_IP\_ADDRESS** as **0x020AA8C0**.

```
#define REMOTE_IP_ADDRESS 0x020AA8C0
```

**To configure IP address:**

IP address to be configured to the device should be in long format and in little endian byte order.

Example: To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP 0X010AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY 0X010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order  
Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK 0x00FFFFFF
```

**Note:**

**In AP mode, configure same IP address for both DEVICE\_IP and GATEWAY macros**

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_ENABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_BYPASS
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENTION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Configuring the BT task

1. Open **rsi\_bt\_app.c** file and update/modify following macros:  
**RSI\_BT\_LOCAL\_NAME** refers name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BT_LOCAL_NAME "SPP_SLAVE"
```

**PIN\_CODE** refers four bytes string required for pairing process.

```
#define PIN_CODE "4321"
```

Following are the **non-configurable** macros in the application.

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

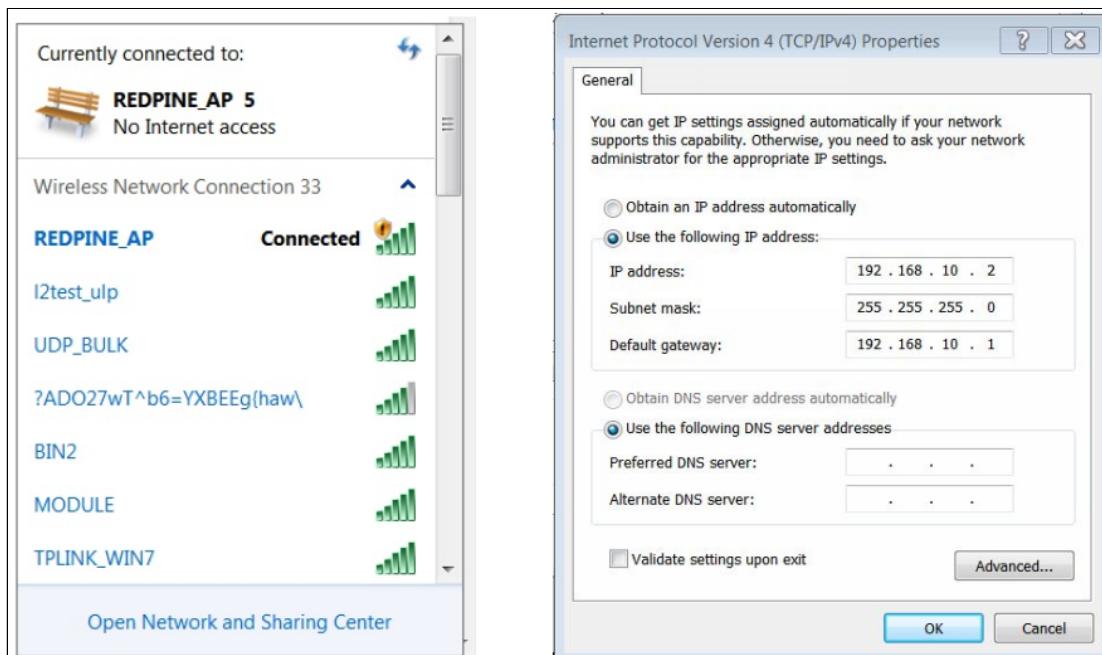
```
#define BT_GLOBAL_BUFF_LEN 10000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

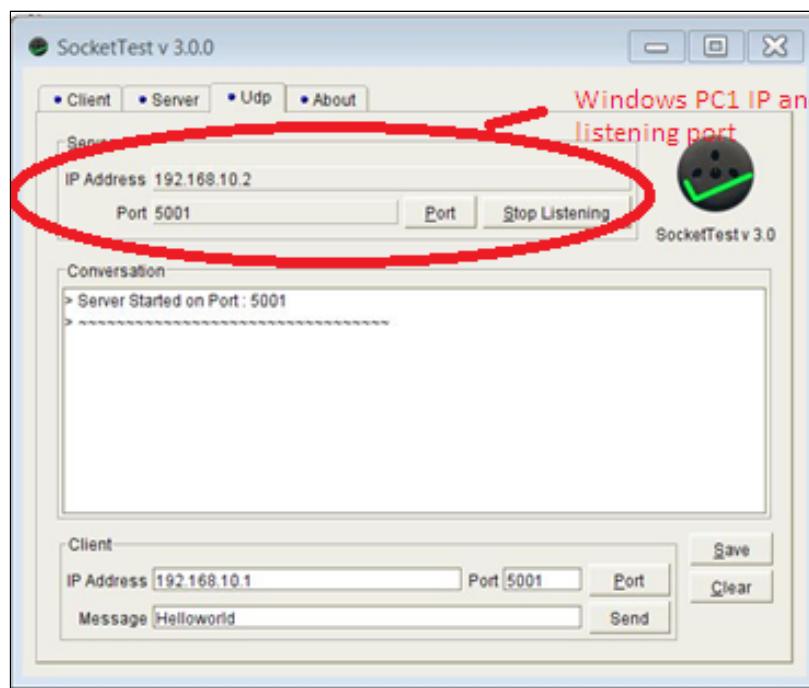
#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	
FEAT_SECURITY_OPEN	
#define RSI_TCP_IP_BYPASS	RSI_ENABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_BYPASS	
#define RSI_CUSTOM_FEATURE_BIT_MAP	0
#define RSI_BAND	RSI_BAND_2P4GHZ

## Executing the Application

1. After the program gets executed, Redpine BT is in Discoverable state and WLAN will create as an Access Point and opens UDP socket on port number **DEVICE\_PORT**.
2. From Windows Laptop (STA), do scan and connect to Redpine Access point (Ex: "REDPINE\_AP") and assign static IP address to Laptop which is same as **REMOTE\_IP\_ADDRESS** configured in rsi\_wlan\_ap\_app.c file.



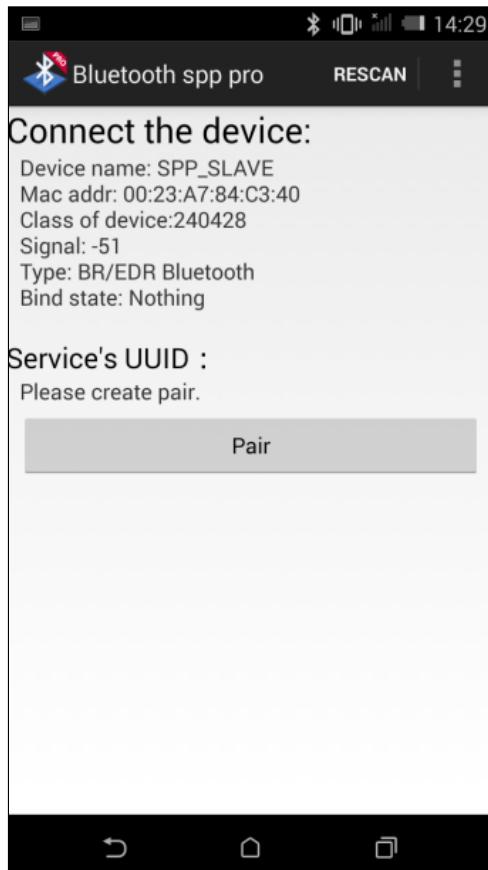
3. Install UDP socket application in Windows laptop and open UDP server socket on port number **REMOTE\_PORT** (Ex: 5001) to receive the data sent by BT remote device.



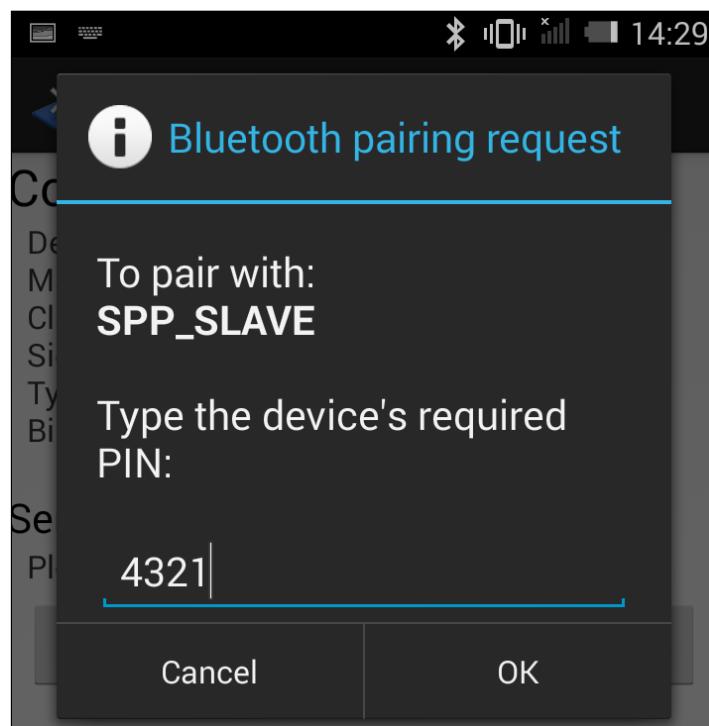
4. Open Bluetooth SPP pro app on mobile and do the scan until Redpine device (Ex: "SPP\_SLAVE") gets present in scan list.



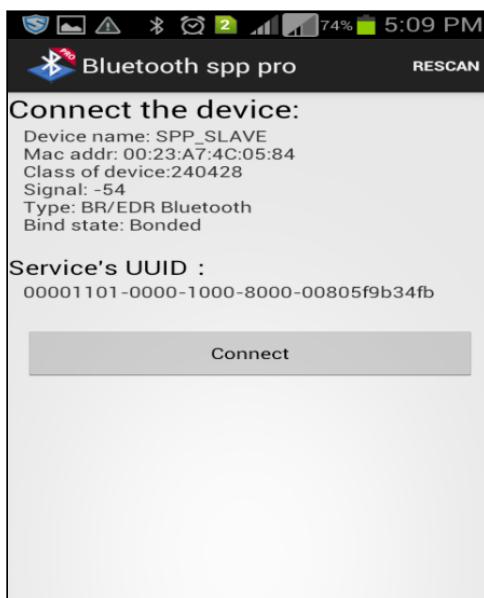
5. After successful scan, select the device and initiate pairing to Redpine device.

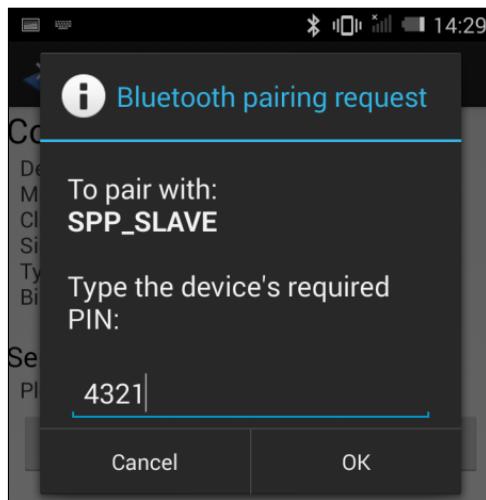


6. After initiating paring, Pairing request will pop-up at smart phone side and issue secret key which is given at Redpine device (PIN\_CODE ) side.

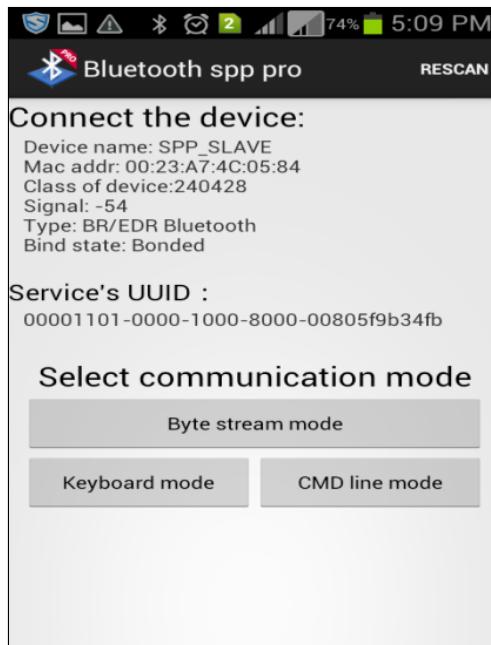


7. After successful pair, initiate SPP connection to Redpine module and give secret key for the received pairing request at remote device side.

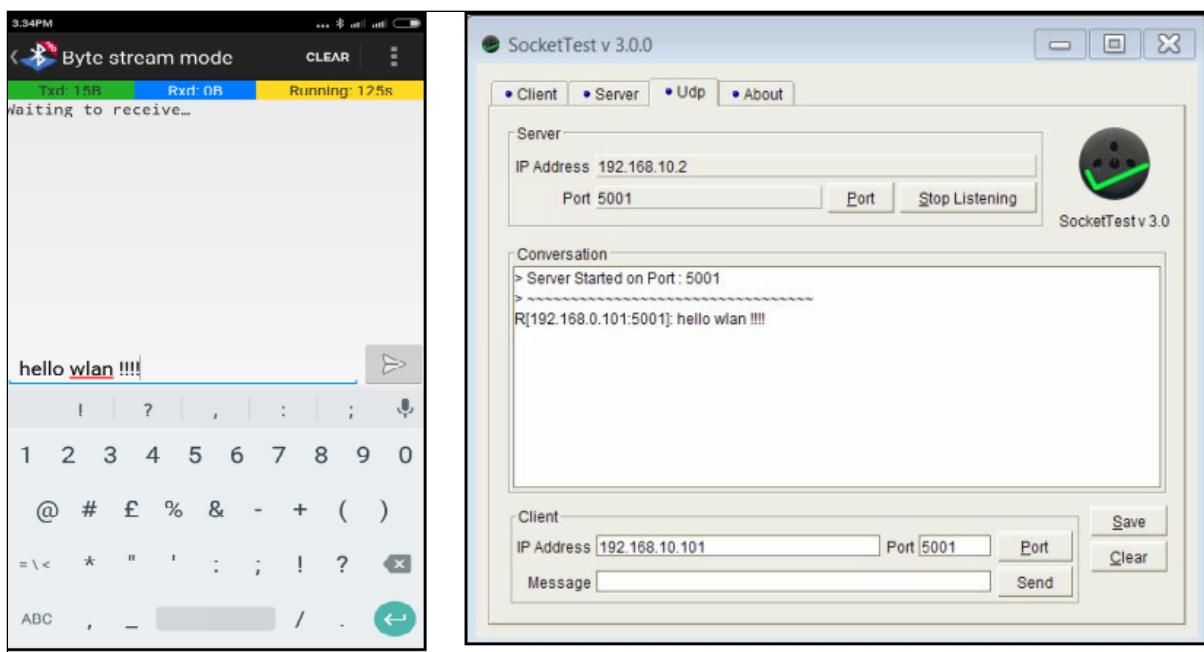




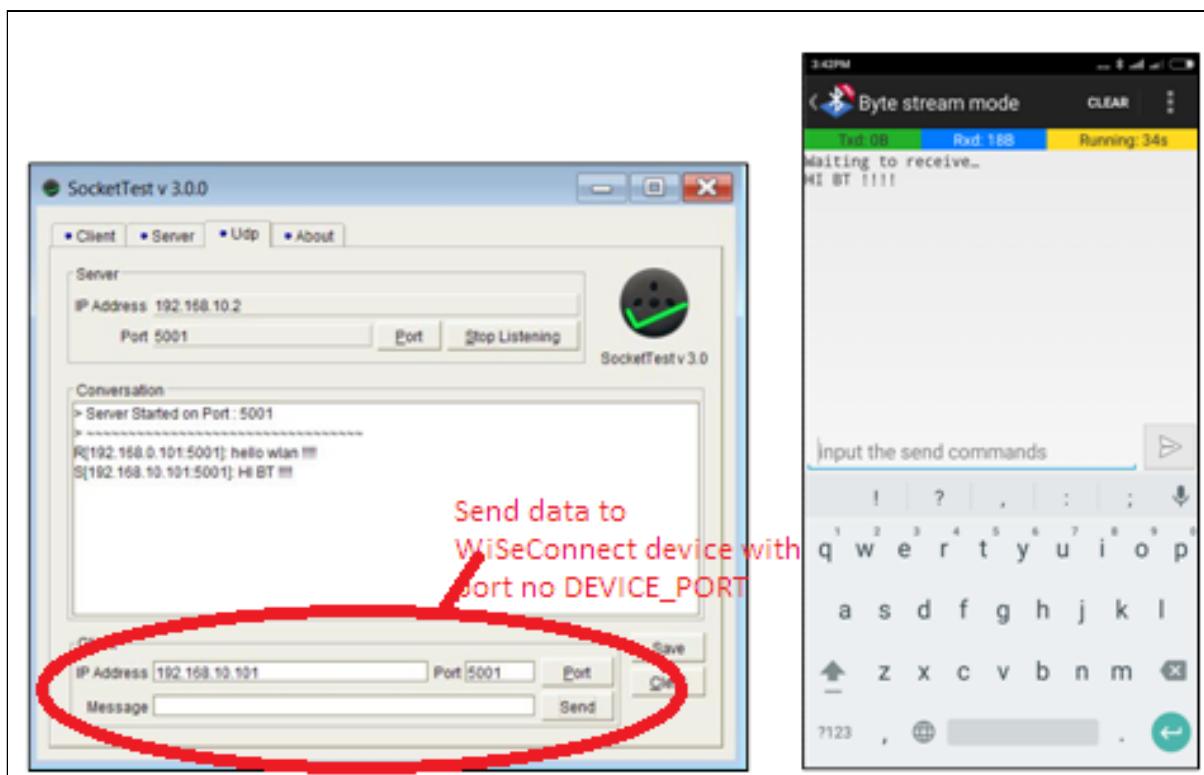
8. After successful SPP connection, select "Byte stream mode" to send and receive the data.



9. Send a message (Ex: "Hello wlan") from SPP pro APP to Redpine device. Redpine device forwards the received message from BT remote device to Windows laptop which is connected as a STA over WiFi protocol using UDP socket. User can observe the message on UDP socket opened on Windows laptop.



10. Now, send a message (Ex: "Hi BT") from Windows laptop to Redpine device using UDP socket with destination IP **DEVICE\_IP** and port number **DEVICE\_PORT**. Redpine device forwards the received message from Windows laptop to remote BT device which is connected to Redpine BT device over BT protocol. User can observe the message on SPP pro app.



**Note:**

`rsi_wlan_app_send_to_bt()` function defined in `rsi_bt_app.c` is to send message from WLAN task to BT task

## 6.3 WLAN STA bridge TCPIP bypass Example

### 6.3.1 Application Overview

#### Overview

The coex application demonstrates how information can be exchanged seamlessly using two wireless protocols (WLAN and BT) running in the same device.

In this coex application, Redpine BT device connects with remote BT device (Smart Phone) and Redpine WiFi interface connects with an Access Point in station mode and do data transfer in BT and WiFi interfaces.

The coex application has WLAN and BT tasks and acts as an interface between remote Smartphone BT device and remote PC which is connected to Access point. Smartphone interacts with BT task, while remote PC interacts with WLAN task. When Smartphone connects and sends message to Redpine device, BT task accepts and sends to WLAN task, which in turn sends to remote PC which is connected to Access Point. Similarly, when remote PC sends message to Redpine device, the message will be sent to Smartphone via BT task.

Thus messages can be seamlessly transferred between PC and Smartphone.

#### Sequence of Events

#### WLAN Task

This Application explains user how to:

- Configure Redpine device in station mode
- Connect Redpine device to Access point
- Receive UDP data sent by connected station and forward to BT task
- Send data received by BT task to connected station using UDP protocol

#### BT Task

This Application explains user how to:

- Configure Redpine device to SPP profile mode
- Configure Redpine device in discoverable and connectable mode
- Establish SPP profile level connection with remote smart phone
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

### 6.3.2 Application Setup

The WiSeConnect in its many variants supports SPI and UART interfaces. Depending on the interface used, the required set up is as below:

### 6.3.3 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Access point
- Windows PC2 with UDP socket application

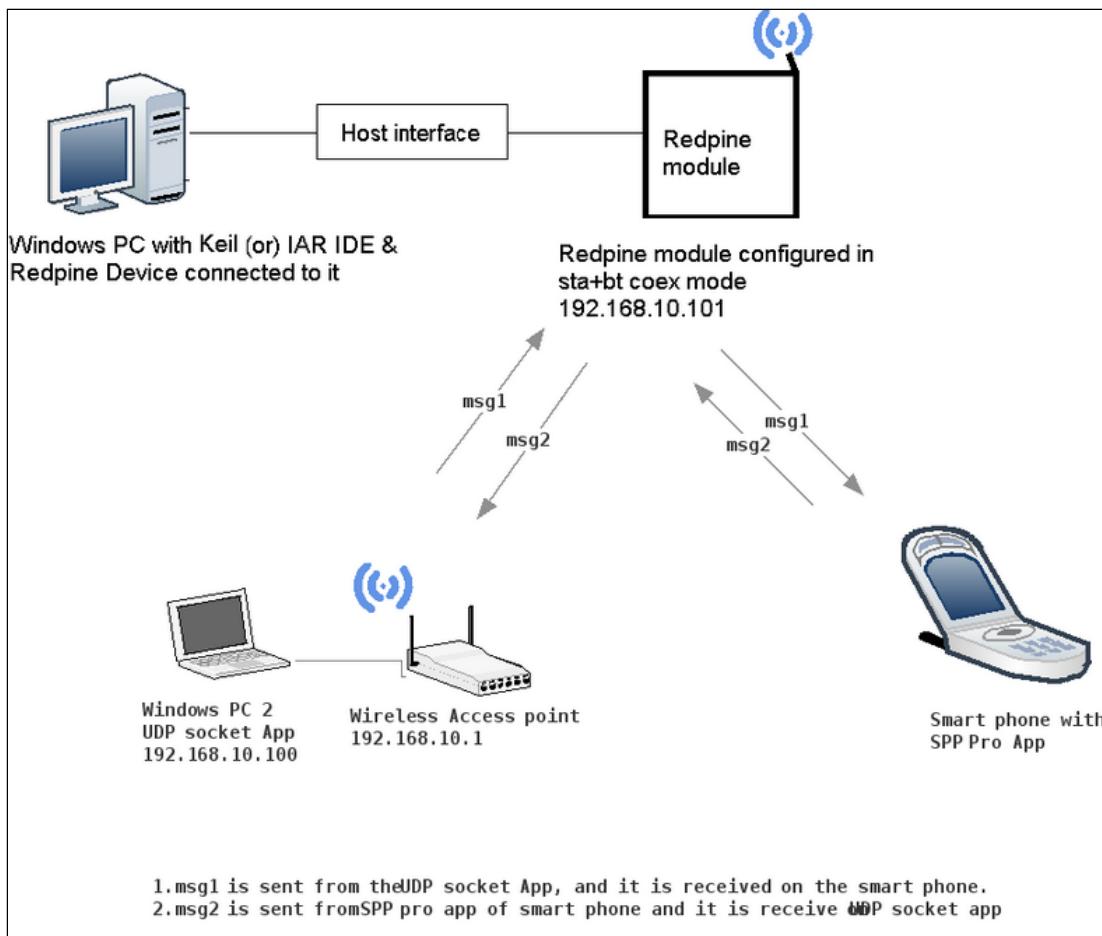
**Note:**

Download UDP socket application from below link,  
<http://sourceforge.net/projects/sockettest/files/latest/download>

- BT supported Smart phone with SPP application.

**Note:**

Install SPP Pro Application for BT SPP application.



**Figure 1: Setup to demonstrate WLAN Station BT bridge tcp ip bypass application**

### 6.3.4 Configuration and Execution of the Application'

#### Configuring the Application

##### Configuring the WLAN task

1. Open **rsi\_wlan\_app.c** file and update/modify following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE      RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK             "<psk>"
```

**DEVICE\_PORT** port refers internal UDP server port number

```
#define DEVICE_PORT      5001
```

**REMOTE\_PORT** port refers remote UDP server port number

```
#define REMOTE_PORT      5001
```

**REMOTE\_IP\_ADDRESS** refers remote peer IP address to send data received by BT device.

IP address of Windows PC2 which is connected to Access Point through LAN.

IP address should be in long format and in little endian byte order.

Example: To configure "192.168.10.2" as IP address, update the macro **REMOTE\_IP\_ADDRESS** as **0x020AA8C0**.

```
#define REMOTE_IP_ADDRESS
```

0x020AA8C0

**To configure IP address:**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE
```

0

**Note:**

Configure DHCP\_MODE macro to 0 and assign IP address through static as co-ex application running in TCP/IP bypass mode.

IP address which is to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP
```

0X0A0AA8C0

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY
```

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK
```

0x00FFFFFF

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_ENABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_BYPASS
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENSION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

### Configuring the BT Task

1. Open **rsi\_bt\_app.c** file and update/modify following macros,  
**RSI\_BT\_LOCAL\_NAME** refers name of the Redpine device to appear during scanning by remote devices.

```
#define RSI_BT_LOCAL_NAME "SPP_SLAVE"
```

**PIN\_CODE** refers four bytes string required for pairing process.

```
#define PIN_CODE "4321"
```

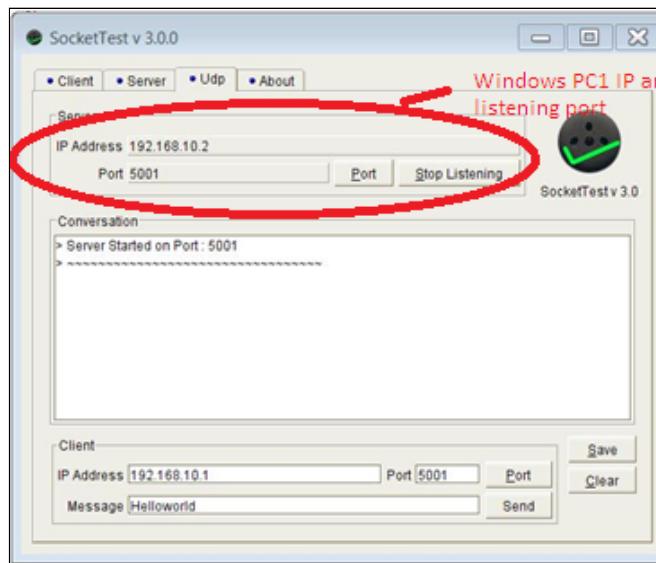
Following are the **non-configurable** macros in the application.

**BT\_GLOBAL\_BUFF\_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN 1000
```

### Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. Install UDP socket application in Windows PC1 and open UDP server socket on port number **REMOTE\_PORT** (**Ex: 5001**) to receive the data sent by BT remote device.

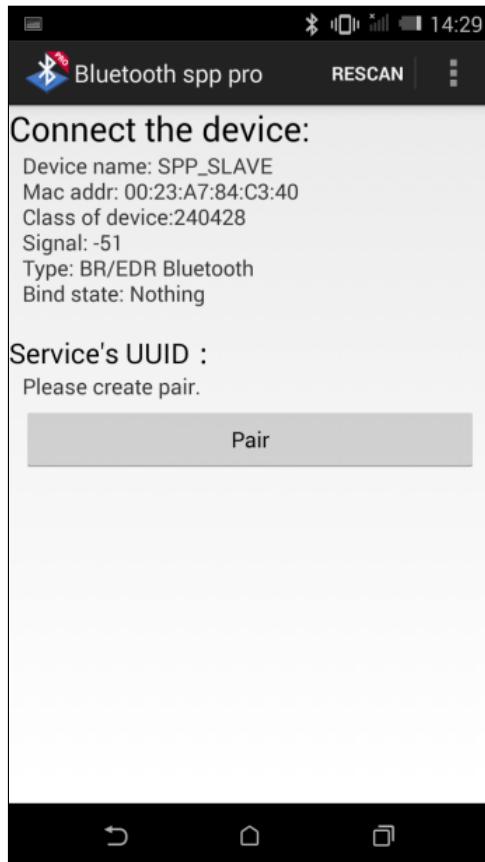


3. After the program gets executed, Redpine BT is in Discoverable state and WLAN will establish connection with wireless access point. After successful connection with wireless access point Redpine device opens UDP socket on port number **DEVICE\_PORT**.

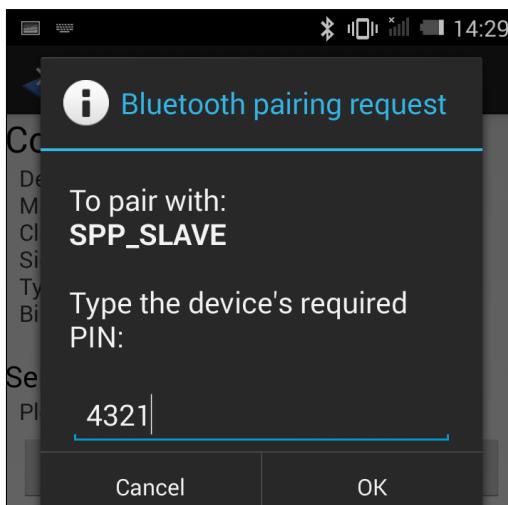
4. Open Bluetooth SPP pro app on mobile and do scan until Redpine device (Ex: "SPP\_SLAVE") present in scan list.



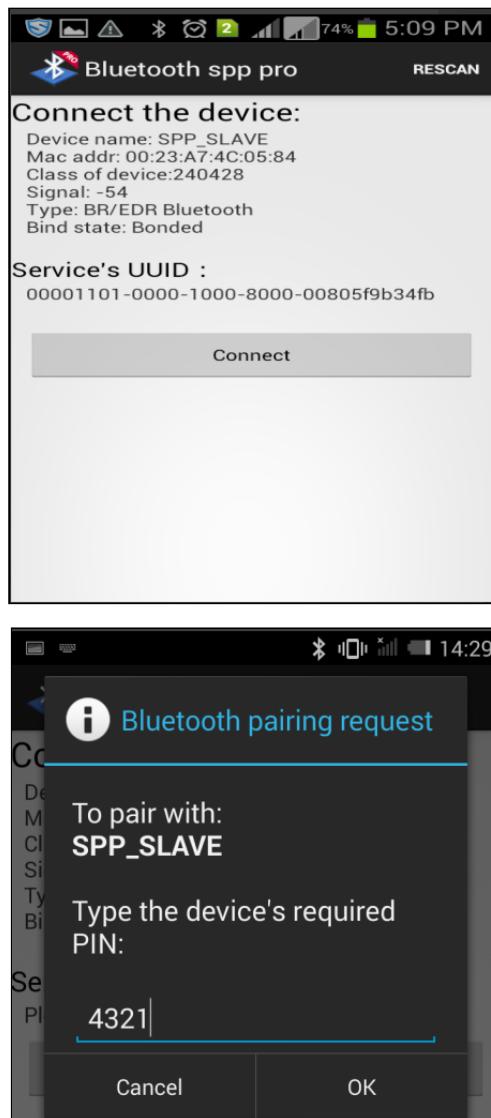
5. After successful scan, select the device and initiate pairing to Redpine device.



6. After initiating paring, Pairing request will pop-up at smart phone side and issue secret key which is given at Redpine device (PIN\_CODE ) side.



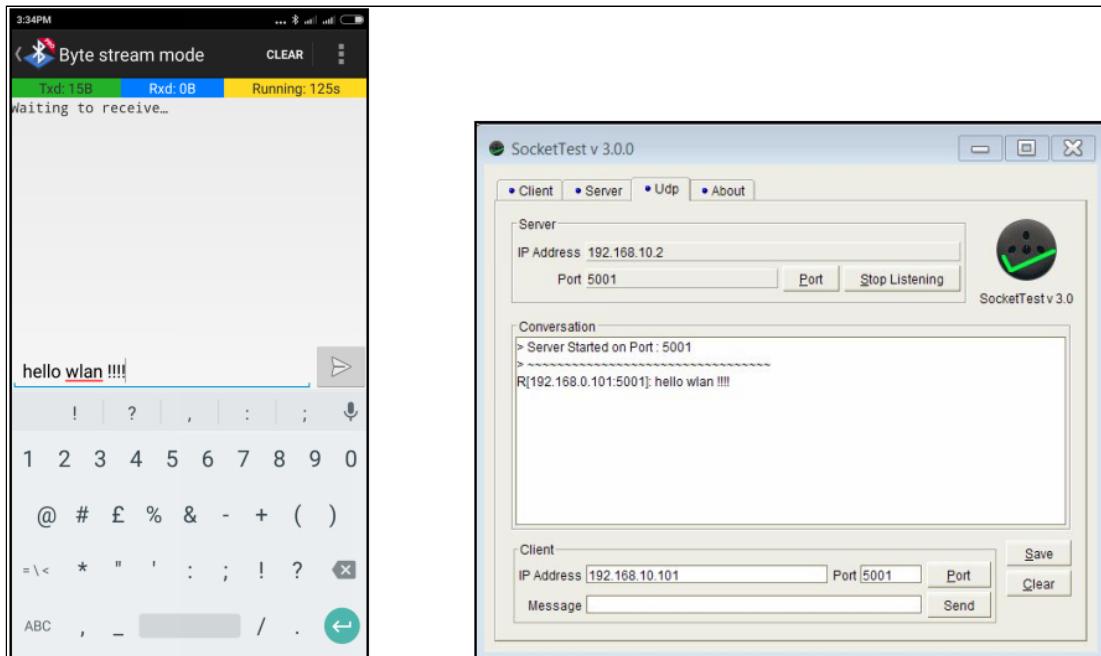
7. After successful pair, initiate SPP connection to Redpine module and give secret key for the received pairing request at remote device side.



8. After successful SPP connection, select "Byte stream mode" to send and receive the data.

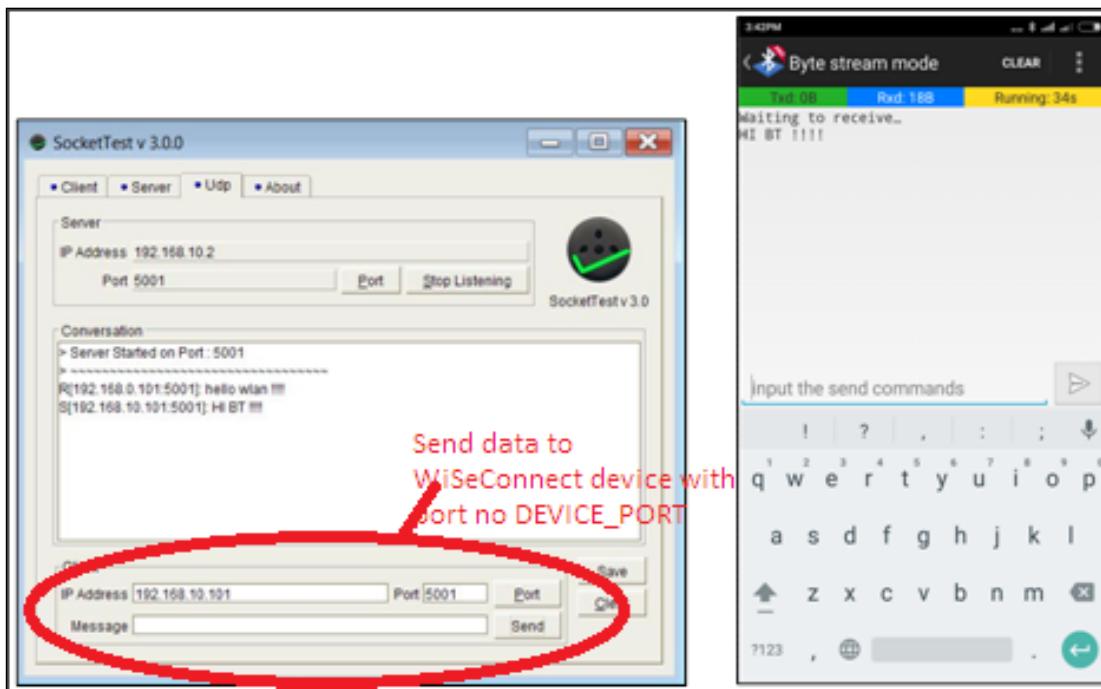


9. Send a message (Ex: "Hello wlan") from SPP pro APP to Redpine device. Redpine device forwards the received message from BT remote device to Windows PC1 which is connected to Access Point over WiFi protocol using UDP socket. User can observe the message on UDP socket opened on Windows PC1.

**Note:**

`rsi_bt_app_send_to_wlan()` function defined in `rsi_wlan_app.c` to send message from BT task to WLAN task

10. Now, send a message (Ex: "Hi BT") from Windows PC1 to Redpine device using UDP socket with destination IP **DEVICE\_IP** and port number **DEVICE\_PORT**. Redpine device forwards the received message from Windows PC1 to remote BT device which is connected to Redpine BT device over BT protocol. User can observe the message on SPP pro app.



**Note:**

rsi\_wlan\_app\_send\_to\_bt() function defined in **rsi\_bt\_app.c** to send message from WLAN task to BT task

## 6.4 WLAN-STATION BT Bridge Example

### 6.4.1 Application Overview

#### Overview

The coex application demonstrates how information can be exchanged seamlessly using two wireless protocols (WLAN and BT) running in the same device.

#### Sequence of Events

##### WLAN Task

This Application explains user how to:

- Create Redpine device as Station
- Connect Redpine station to remote Access point
- Receive UDP data sent by connected station and forward to BT task
- Send data received by BT task to connected station using UDP protocol

##### BT Task

This Application explains user how to:

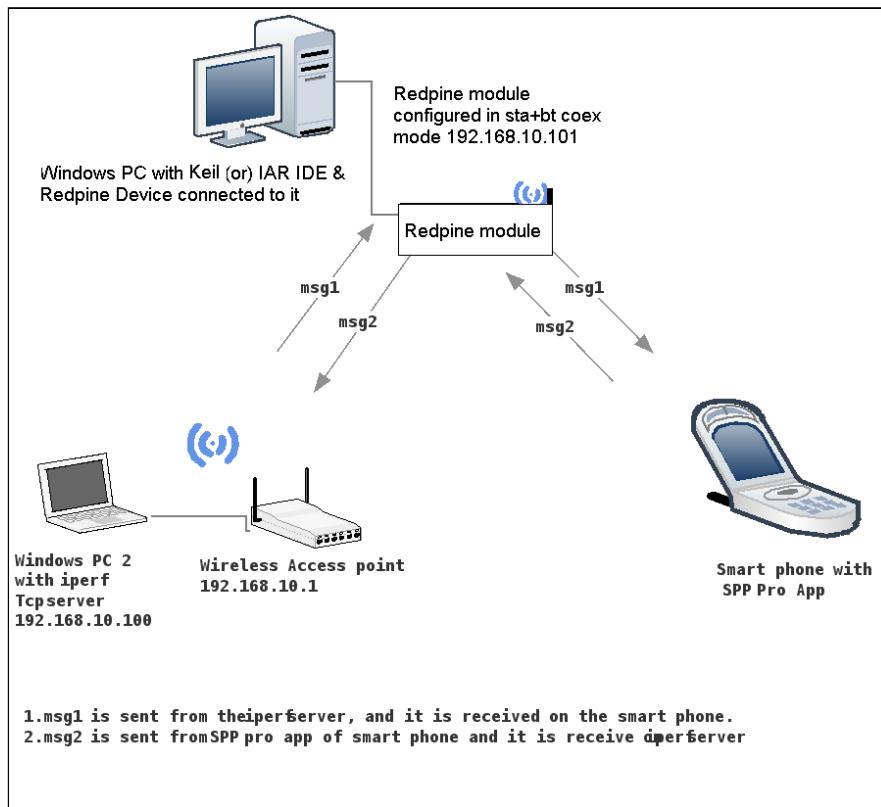
- Configure Redpine device to SPP profile mode
- Configure device in discoverable and connectable mode
- Establish SPP profile level connection with remote smart phone
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

#### 6.4.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Smart phone/tablet with BT Application (Ex: Bluetooth SPP Pro from Google play store)
- WLAN Access Point and a Windows PC with iperf application



**Figure 1: Setup to demonstrate WLAN Station BT bridge application**

## Description

The coex application has WLAN and BT tasks and acts as an interface between Smartphone and PC. Smartphone interacts with BT task, while PC Both PC and Redpine WLAN would be connected to a Wireless Access Point, thus both are connected together wirelessly interacts with WLAN task. When Smartphone connects and sends message to Redpine device, BT task accepts and sends to WLAN task, which in turn sends to Access Point connected PC. Similarly, when PC sends message to Redpine device, the message will be sent to Smartphone via BT task. Thus messages can be seamlessly transferred between Windows PC and Smartphone.

## Details of the Application

Redpine WLAN acts as a Station and connects to an Access Point  
Redpine BT acts as a Slave device with SPP profile running in it, while Smart phone acts as Master device with SPP profile running in it.  
Initially, proprietary Simple chat service is created with SPP profile (Redpine device) to facilitate message exchanges.

- The WLAN task (running in Redpine device) mainly includes following steps.
  1. Connects to a Access Point
  2. Exchanges data over TCP Server socket with the peer(Windows PC)
- The BT task (running in Redpine device) mainly includes following steps.
  1. Creates chat service
  2. Configures the device in Discoverable mode and connectable mode.

WLAN and BT tasks forever run in the application to serve the asynchronous events

### 6.4.3 Configuration and Execution of the Application

#### Configuring the Application

##### Configuring the WLAN task

1. Open **rsi\_wlan\_ap\_app.c** file and update/modify following macros in order to establish connection with the Access-point.  
**SSID** refers to the Access point to which user wants to connect.  
**SECURITY\_TYPE** is the security type of the Access point.  
**PSK** refers to the secret key if the Access point is configured in WPA/WPA2 security modes.

```
#define SSID
"<ap_name>"  

#define SECURITY_TYPE
<security-type>  

#define PSK
" "
```

2. Enable/Disable DHCP mode  
1 – Enables DHCP mode (gets the IP from DHCP server)  
0 – Disables DHCP mode

```
#define DHCP_MODE
```

1

3. If DHCP mode is disabled, then change the following macros to configure static IP address  
IP address to be configured to the device should be in long format and in little endian byte order.  
Example: To configure "192.168.10.101" as IP address, update the macro **DEVICE\_IP** as **0x650AA8C0**.

```
#define DEVICE_IP  
0x650AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY  
0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK  
0x00FFFFFF
```

To establish TCP connection and transfer data to the remote socket configure the below macros. Internal socket port number.

```
#define DEVICE_PORT
```

5001

Port number of the remote server

```
#define SERVER_PORT
```

5001

IP address of the remote server

```
#define SERVER_IP_ADDRESS  
0x650AA8C0
```

Number of packets to send

#define NUMBER\_OF\_PACKETS

1000

Application memory length which is required by the driver

#define GLOBAL\_BUFF\_LEN

8000

Include **rsi\_wlan\_app.c**, **rsi\_bt\_app.c** and **main.c** files in the project, build and launch the application Open server socket on remote machine,For example, to open TCP server socket with port number 5001 on remote side, use the command as given below

4. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_CLIENT | TCP_IP_TOTAL_SOCKETS_1)
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENSION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Configuring the BT task

1. Open **rsi\_bt\_app.c** file and update/modify following macros :

- RSI\_BT\_LOCAL\_NAME - Name of the Redpine device
- PIN\_CODE - Four byte string required for pairing process.

Following are the **non-configurable** Macros in the Application file.

1. BT\_GLOBAL\_BUFF\_LEN – Number of bytes required for the Application and the Driver.
2. RSI\_APP\_EVENT\_CONNECTED - Event number to be set on connection establishment.
3. RSI\_APP\_EVENT\_DISCONNECTED - Event number to be set on disconnection.
4. RSI\_APP\_EVENT\_PINCODE\_REQ - Event number to be set on Pincode request for pairing.
5. RSI\_APP\_EVENT\_LINKKEY\_SAVE - Event number to be set on link key save.
6. RSI\_APP\_EVENT\_AUTH\_COMPLT - Event number to be set on authentication complete.
7. RSI\_APP\_EVENT\_LINKKEY\_REQ - Event number to be set on link key request for connection.
8. RSI\_APP\_EVENT\_SPP\_CONN - Event number to be set on SPP connection.
9. RSI\_APP\_EVENT\_SPP\_DISCONN - Event number to be set on SPP disconnection.
10. RSI\_APP\_EVENT\_SPP\_RX - Event number to be set on SPP data received from Master.

## Executing the coex Application

1. Connect WiSeConnect device to the Windows PC running KEIL or IAR IDE
2. Build and launch the application.
3. After the program gets executed, Redpine BT is in Discoverable state and WLAN has established TCP server socket with peer (PC).
4. Open a BT SPP Pro App in the Smartphone and do scan.

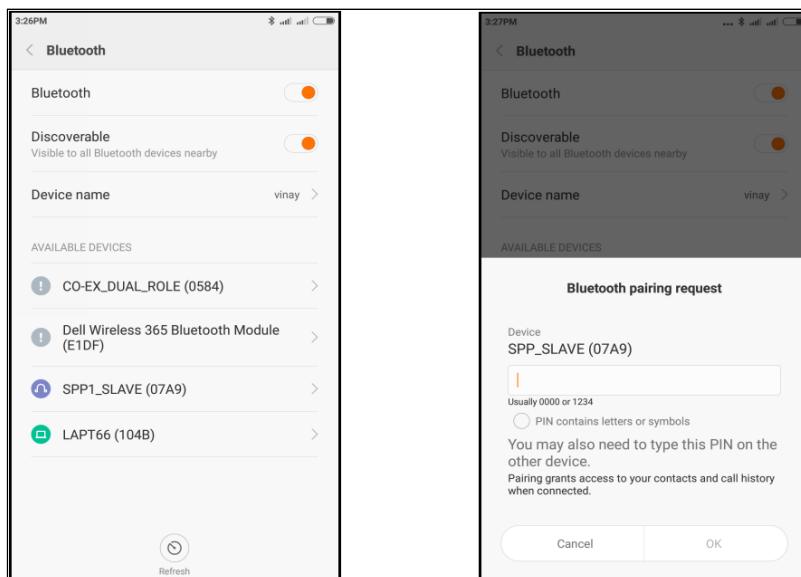


Figure 2: Turn ON Bluetooth ,Scan for BT devices and Pair with the Redpine BT device

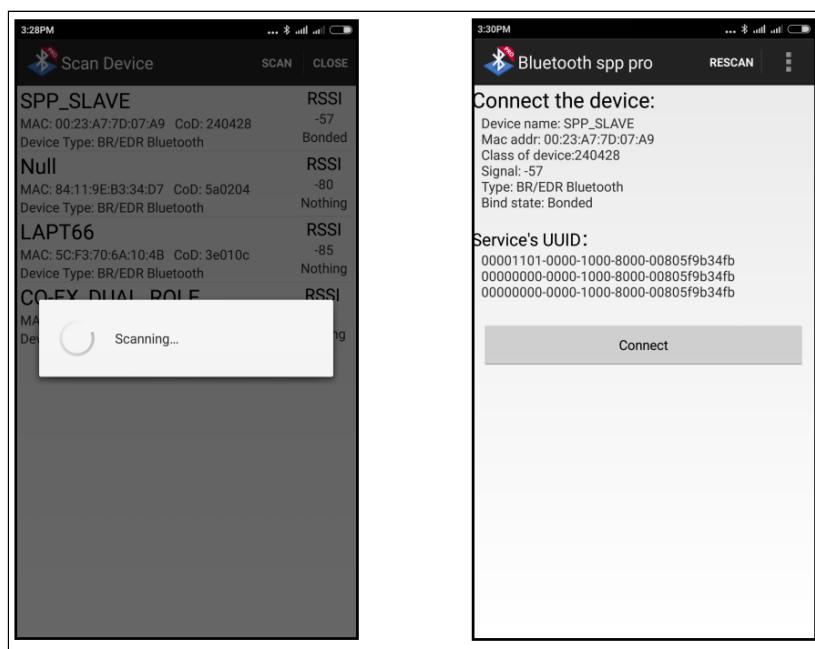
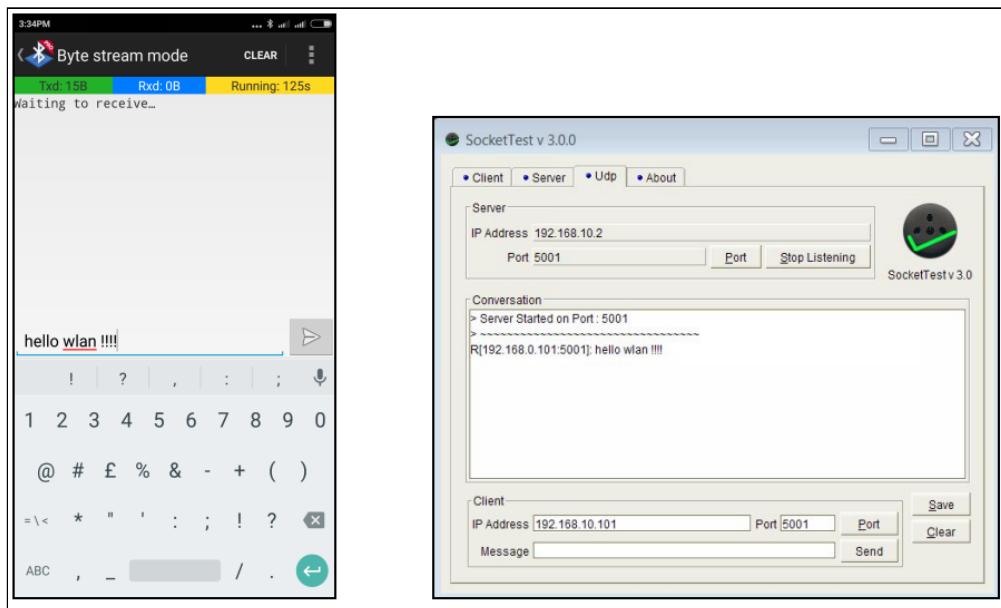
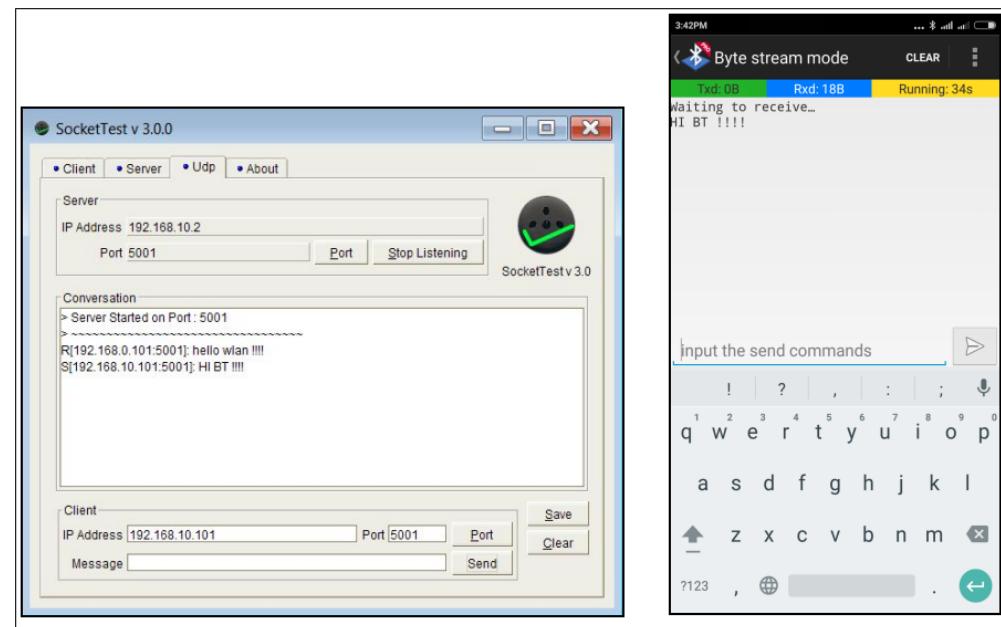


Figure 3: Scan using SPP pro App and get connected to the Peripheral

5. In the App, Redpine BT would appear with the name configured in the macro **RSI\_BT\_LOCAL\_NAME**.
6. Now initiate connection from the SPP App running in the Smartphone.
7. After BT connection is established, send a message from the App to Redpine BT. Observe this message in the PC connected via TCP server socket with WiSeConnect WLAN.



**Figure 4: Sending Data from BT SPP pro APP and received on Wifi AP socket App**



**Figure 5: Sending Data from Wifi AP Socket App and received on BT SPP Por App**

8. Now, send a message from PC to Redpine WLAN via TCP server socket and observe the same in the Smartphone

9. rsi\_bt\_app\_send\_to\_wlan() function defined in rsi\_wlan\_app.c to send message from BT task to WLAN task.

10. With the help of wlan task, message is transferred to PC.

11. Message from PC to WLAN application via socket and rsi\_wlan\_app\_send\_to\_bt() function defined in rsi\_bt\_app.c called asynchronously to send message from WLAN task to BT task. From BT task message transferred to client with the event.

## 6.5 WLAN-STATION BT THROUGHPUT Example

### 6.5.1 Introduction

This example is applicable to WiSeConnect<sup>TM</sup> and WiSeMCU<sup>TM</sup> parts. For simplicity, this document refers to WiSeConnect, but all discussion applies to both WiSeConnect and WiSeMCU parts. The feature(s) used in this example may or may not be available in your part. Refer to the product datasheet to verify the features available in your part.

### Application Overview

#### Overview

The coex application demonstrates throughput measurement of wifi while BT is in connection .

#### Sequence of Events

#### WLAN Task

This application can be used to configure Redpine module in UDP client / server or TCP client / server. To measure throughput, following configuration can be applied.

- To measure UDP Tx throughput, module should configured as UDP client.
- To measure UDP Rx throughput, module should configured as UDP server.
- To measure TCP Tx throughput, module should configured as TCP client.
- To measure TCP Rx throughput, module should configured as TCP server.

#### BT Task

This Application explains user how to:

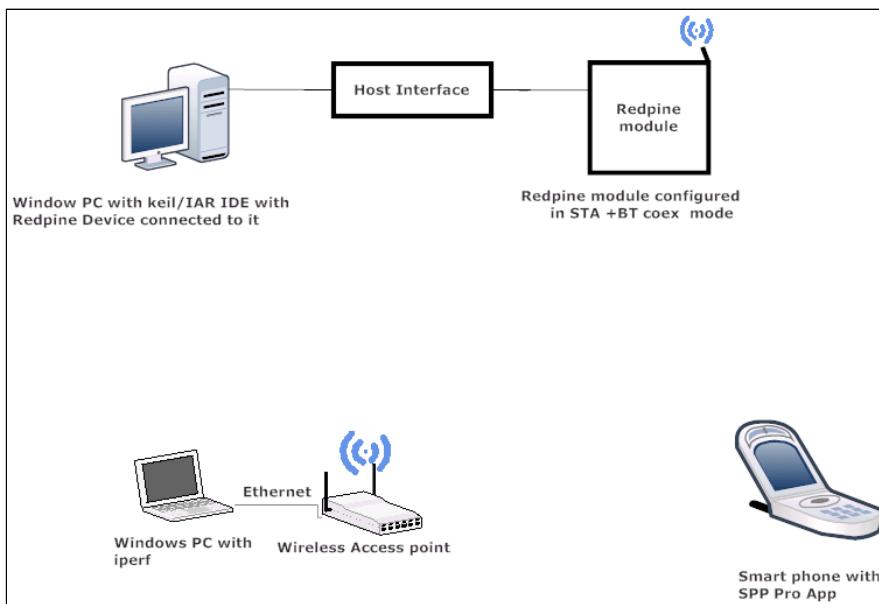
- Configure Redpine device to SPP profile mode
- Configure device in discoverable and connectable mode
- Establish SPP profile level connection with remote smart phone
- Receive data sent by Smart phone.

#### Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Smart phone/tablet with BT Application (Ex: Bluetooth SPP Pro from Google play store)
- WLAN Access Point and a Windows PC with iperf application



**Figure 1: Setup to demonstrate WLAN Station BT bridge application**

### Description

The coex application has WLAN and BT tasks and acts as an interface between Smartphone and PC. Smartphone interacts with BT task, while Both PC and Redpine WLAN would be connected to a Wireless Access Point, thus both are connected together wirelessly interacts with WLAN task. When Smartphone connects and sends message to Redpine device, BT task accepts .

Similarly, data transfer will happen for Station and AP.

### Details of the Application

Redpine WLAN acts as a Station and connects to an Access Point

Redpine BT acts as a Slave device with SPP profile running in it, while Smart phone acts as Master device with SPP profile running in it.

Initially, proprietary Simple chat service is created with SPP profile (Redpine device) to facilitate message exchanges.

- The WLAN task (running in Redpine device) mainly includes following steps.
  1. Connects to a Access Point
  2. Exchanges data over socket with the peer(Windows PC)
- The BT task (running in Redpine device) mainly includes following steps.
  1. Creates chat service

2. Configures the device in Discoverable mode and connectable mode.

WLAN and BT tasks forever run in the application to serve the asynchronous events

### 6.5.2 Configuration and Execution of the Application

#### Configuring the Application

##### Configuring the WLAN task

1. Open **rsi\_wlan\_app.c** file and update / modify the following macros,  
**SSID** refers to the name of the Access point.

#define SSID	"<ap name>"
--------------	-------------

**CHANNEL\_NO** refers to the channel in which AP would be started

#define CHANNEL_NO	11
--------------------	----

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities.  
Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

#define SECURITY_TYPE	RSI_OPEN
-----------------------	----------

**PSK** refers to the secret key if the Access point is to be configured in WPA/WPA2 security modes.

#define PSK	"<psk>"
-------------	---------

Enable / Disable DHCP mode  
1-Enables DHCP mode (gets the IP from DHCP server)  
0-Disables DHCP mode

#define DHCP_MODE	<dhcp mode>
-------------------	-------------

##### To configure static IP address

IP address to be configured to the device should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

#define DEVICE\_IP

0X010AA8C0

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro GATEWAY as **0x010AA8C0**

#define GATEWAY

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro NETMASK as **0x00FFFFFF**

#define NETMASK

0x00FFFFFF

2. To establish UDP/TCP connection and transfer/receive data to the remote socket configure the below macros  
Internal device port number

#define PORT\_NUM

<Local\_port>

Port number of the remote server

#define SERVER\_PORT

<Remote\_port\_num>

IP address of the remote server

#define SERVER\_IP\_ADDRESS

0x640AA8C0

Application memory length which is required by the driver

#define GLOBAL\_BUFF\_LEN

8000

Application can use receive buffer size of 1400

#define BUFF\_SIZE

1400

Application can select throughput type as UDP Tx, UDP Rx, TCP Tx or TCP Rx. Following is macro need to use.

```
#define THROUGHPUT_TYPE
```

UDP\_TX

Following is macro used for throughput type selection

#define UDP_TX	0
#define UDP_RX	1
#define UDP_TX	2
#define UDP_TX	3

**Note:**

In AP mode, configure same IP address for both DEVICE\_IP and GATEWAY macros.

3. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_CLIENT | FEAT_AGGRAGATION )
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENTION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Configuring the BT task

1. Open **rsi\_bt\_app.c** file and update/modify following macros:
  - RSI\_BT\_LOCAL\_NAME - Name of the Redpine device
  - PIN\_CODE - Four byte string required for pairing process.

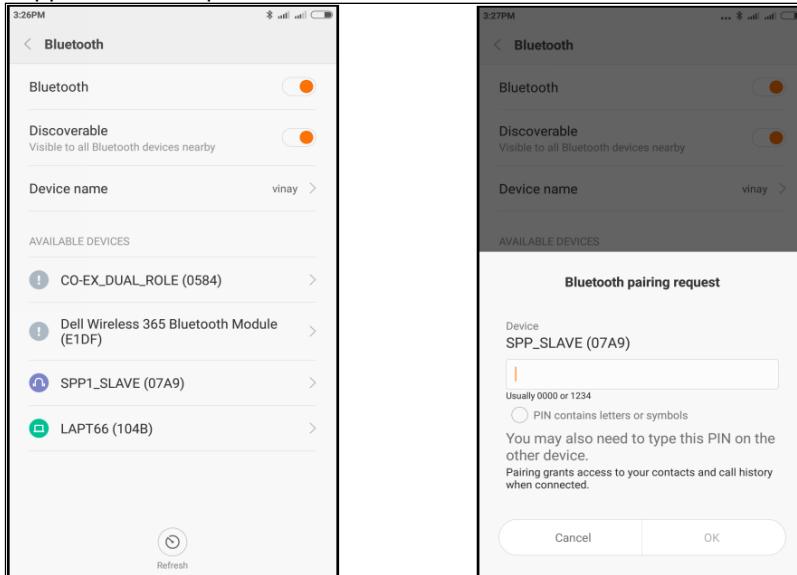
Following are the **non-configurable** Macros in the Application file.

1. BT\_GLOBAL\_BUFF\_LEN – Number of bytes required for the Application and the Driver.
2. RSI\_APP\_EVENT\_CONNECTED - Event number to be set on connection establishment.
3. RSI\_APP\_EVENT\_DISCONNECTED - Event number to be set on disconnection.
4. RSI\_APP\_EVENT\_PINCODE\_REQ - Event number to be set on Pincode request for pairing.
5. RSI\_APP\_EVENT\_LINKKEY\_SAVE - Event number to be set on link key save.
6. RSI\_APP\_EVENT\_AUTH\_COMPLT - Event number to be set on authentication complete.
7. RSI\_APP\_EVENT\_LINKKEY\_REQ - Event number to be set on link key request for connection.
8. RSI\_APP\_EVENT\_SPP\_CONN - Event number to be set on SPP connection.

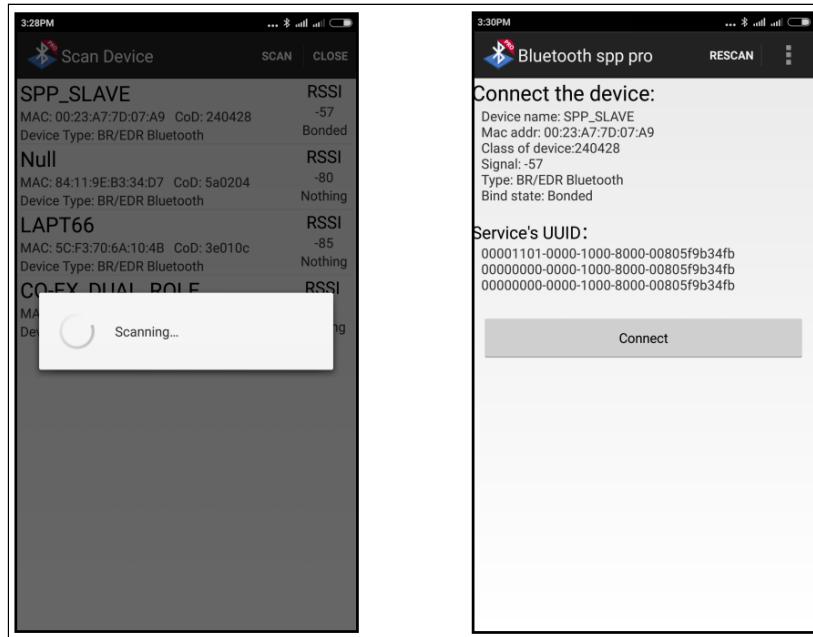
9. RSI\_APP\_EVENT\_SPP\_DISCONNECT - Event number to be set on SPP disconnection.
10. RSI\_APP\_EVENT\_SPP\_RX - Event number to be set on SPP data received from Master.

### Executing the coex Application

1. Connect WiSeConnect device to the Windows PC running KEIL or IAR IDE
2. Build and launch the application.
3. After the program gets executed, Redpine BT is in Discoverable state and WLAN has established TCP server socket with peer (PC).
4. Open a BT SPP Pro App in the Smartphone and do scan.



**Figure 2: Turn ON Bluetooth ,Scan for BT devices and Pair with the Redpine BT device**



**Figure 3: Scan using SPP pro App and get connected to the Peripheral**

5. In the App, Redpine BT would appear with the name configured in the macro **RSI\_BT\_LOCAL\_NAME**.

6. Now initiate connection from the SPP App running in the Smartphone.
7. After BT connection is established, send a message from the App to Redpine BT.
8. To measure throughput, following configuration can be applied.
  - a. To measure UDP Tx throughput, module should be configured as UDP client. Open UDP server at remote port

```
iperf.exe -s -u -p <SERVER_PORT> -i 1
```

- b. To measure UDP Rx throughput, module should be configured as UDP server. Open UDP client at remote port

```
iperf.exe -c <Module_IP> -u -p <Module_Port> -i 1 -b  
<Bandwidth>
```

- c. To measure TCP Tx throughput, module should be configured as TCP client. Open TCP server at remote port.

```
iperf.exe -s -p <SERVER_PORT> -i 1
```

- d. To measure TCP Rx throughput, module should be configured as TCP server. Open TCP client at remote port.

```
iperf.exe -c <Module_IP> -p <module_PORT> -i 1
```

9. To measure throughput, following configuration can be applied.
10. Build and launch the application.
11. After the program gets executed, the device would be connected to Access point having the configuration same as that of in the application and get.
12. The Device which is configured as UDP / TCP server / client will connect to iperf server / client and sends / receives data continuously. It will print the throughput per second.

## 6.6 WLAN-STATION Standby BT Connected Power Save Example

### 6.6.1 Application Overview

The coex application demonstrates a procedure to measure power numbers in WiseMCU coex mode with wlan standby and bt connected powersave.

In this coex application, Redpine BT device connects with remote BT device (ex:Smart phone with spp pro application) and issue connected power save command to NWP. In parallel Redpine WiFi interface connects with an Access Point in station mode and issue connected power save command to NWP.

After proper coex power save mode selection then power save command go to NWP module, with successful power save enable triggering M4 to go sleep.

### Sequence of Events

#### WLAN Task

This Application explains user how to:

- Create Redpine device as Station
- Connect Redpine station to remote Access point
- Getting IP using DHCP/Static.
- Issuing powersave command
- After confirming power save enabled then triggering M4 to go sleep.

#### BT Task

This Application explains user how to:

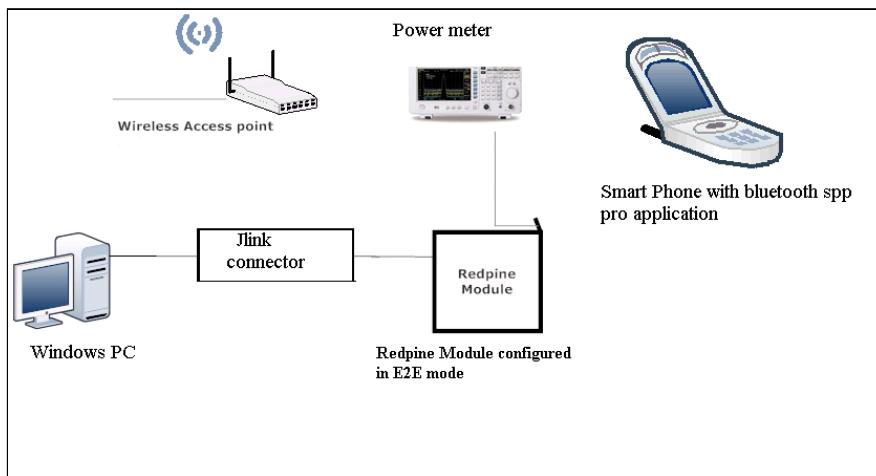
- Connect from Smart phone using Bluetooth spp pro application.
- Configure device in power save profile mode 2 .

#### 6.6.2 Application Setup

The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU based Setup Requirements

- Windows PC with KEIL or IAR IDE
- Redpine module
- Access point
- Jlink(M4 debugger)
- Smart phone/Dongle
- Power meter/ DMM



**Figure 1: Setup to demonstrate WLAN standby BT connected power save application**

#### 6.6.3 Configuration and Execution of the Application

##### Configuring the Application

##### Configuring the WLAN task

1. Open **rsi\_wlan\_app.c** file and update/modify the following macros:

**SSID** refers to the name of the Access point.

```
#define SSID           "<ap  
name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE    RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK             "<psk>"
```

**To configure IP address:**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE        1
```

**Note:**

If user wants to configure STA IP address through DHCP then set DHCP\_MODE to 1 and skip configuring the following DEVICE\_IP, GATEWAY and NETMASK macros.

(Or)

If user wants to configure STA IP address through STATIC then set DHCP\_MODE macro to "0" and configure following DEVICE\_IP, GATEWAY and NETMASK macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP        0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

#define GATEWAY	0x010AA8C0
#define NETMASK	0x00FFFFFF
#define CONCURRENT_MODE #define RSI_FEATURE_BIT_MAP FEAT_SECURITY_OPEN #define RSI_TCP_IP_BYPASS #define RSI_TCP_IP_FEATURE_BIT_MAP (TCP_IP_FEAT_DHCPV4_CLIENT ) #define RSI_CUSTOM_FEATURE_BIT_MAP FEAT_CUSTOM_FEAT_EXTENSION_VALID #define RSI_EXT_CUSTOM_FEATURE_BIT_MAP EXT_FEAT_256K_MODE #define RSI_BAND RSI_BAND_2P4GHZ	RSI_DISABLE RSI_DISABLE

## Configuring the BT task

Open **rsi\_bt\_app.c** file and update/modify following macros.

1. RSI\_BT\_LOCAL\_NAME - Name of the Wyzbee (Master) device
2. PIN\_CODE - Four byte string required for pairing process.
3. PSP\_TYPE - Power save profile type.

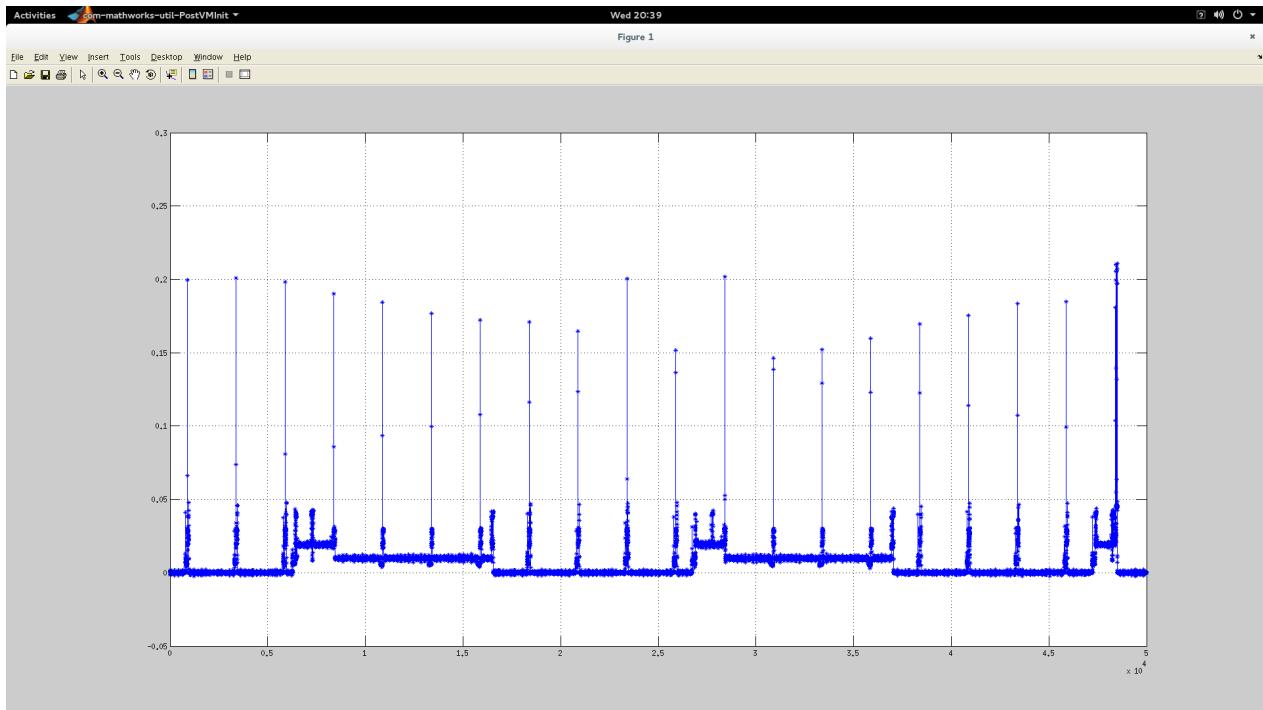
**Note:**

1. PSP\_TYPE is only valid RSI\_SLEEP\_MODE\_2.
2. RSI\_MAXRSI\_MAX\_PSP is only valid in case of BT.\_PSP is only valid in case of BT.
4. SNIFF\_MAX\_INTERVAL - Sniff Maximum interval value
5. SNIFF\_MIN\_INTERVAL – Sniff Minimum interval value
6. SNIFF\_ATTEMPT - Sniff Attempt Value
7. SNIFF\_TIME\_OUT – Sniff Timeout Value

## Executing the Application

1. Make sure configuration settings as mentioned above before to compile&run the Application.
2. Configure the access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
3. After successful connection with access point ,module gets IP either using DHCP or Static based on configuration settings.
4. Trigger NWP to sleep with power mode 2 with ram retention.
5. In parallel BT poll for events and whenever connect event trigger then connection will happen immediately issuing NWP to sleep with power mode 2 with ram retention.
6. After proper coex power save mode selection then power save command go to the module.
7. M4 sleep trigger will happen after successful power save enable of NWP.

8. Note down power measurement as shown below.



## 7 WLAN BT BLE

### 7.1 WLAN-AP BT and BLE Bridge TCPIP Bypass Example

#### 7.1.1 Application Overview

##### Overview

The coex application demonstrates how information can be exchanged seamlessly using wireless protocols (WLAN, BT and BLE) running in the same device.

##### Description

The coex application has WLAN, BLE and BT tasks and acts as an interface between Smartphones and PC. Smartphone1 interacts with BT task, Smartphone2 interacts with BLE task , while PC Redpine module create an Access Point, remote PC (with WLAN station interface) connect to Redpine Access Point interacts with WLAN task When Smartphone2 connects and sends message to Redpine device, BLE task accepts and sends to WLAN task, which in turn sends to PC on UDP port configured (In this example port 5001). When Smartphone1 connects and sends message to Redpine device, BT task accepts and sends to WLAN task, which in turn sends to PC on UDP port configured (In this example port 5002). Similarly, when PC sends message to Redpine device on UDP port (In this example 5001), the message will be sent to Smartphone2 via BLE task. And, when PC sends message to Redpine device on UDP port (In this example 5002), the message will be sent to Smartphone1 via BT task. Thus messages can be seamlessly transferred between PC and Smartphones.

##### Details of the Application

Redpine WLAN acts as Access Point and allow stations to connect.  
RedpineBT acts as a slave device with SPP profile running in it, while Smart phone acts as a Master device with SPP profile running in it.  
Redpine BLE acts as a Peripheral (Slave) device with GATT Server running in it, while Smart phone acts as a Central (Master) device with GATT Client running in it.  
Initially, proprietary Simple chat service is created with SPP profile (Redpine device) to facilitate message exchanges.

- The WLAN task (running in Redpine device) mainly includes following steps.
  1. Start up as Access point and allow stations to connect
  2. Open two UDP sockets and receive/send data
    - The BT task (running in Redpine device) mainly includes following steps.
      1. Creates chat service
      2. Configures the device in Discoverable mode and Connectable mode.
    - The BLE task (running in Redpine device) mainly includes following steps.
      1. Creates chat service
      2. Configures the device to Advertise

WLAN, BLE and BT tasks forever run in the application to serve the asynchronous events.

## Sequence of Events

### WLAN Task

This Application explains user how to:

- Create device as an Access Point
- Connect stations to Redpine Access Point
- Open TCP server socket in device
- Receive TCP data sent by connected station and forward to BLE task
- Send data received by BLE task to connected station using TCP protocol

### BT Task

This Application explains user how to:

- Configure Redpine device to SPP profile mode
- Configure device in discoverable and connectable mode
- Establish SPP profile level connection with remote smart phone
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

### LE Task

This Application explains user how to:

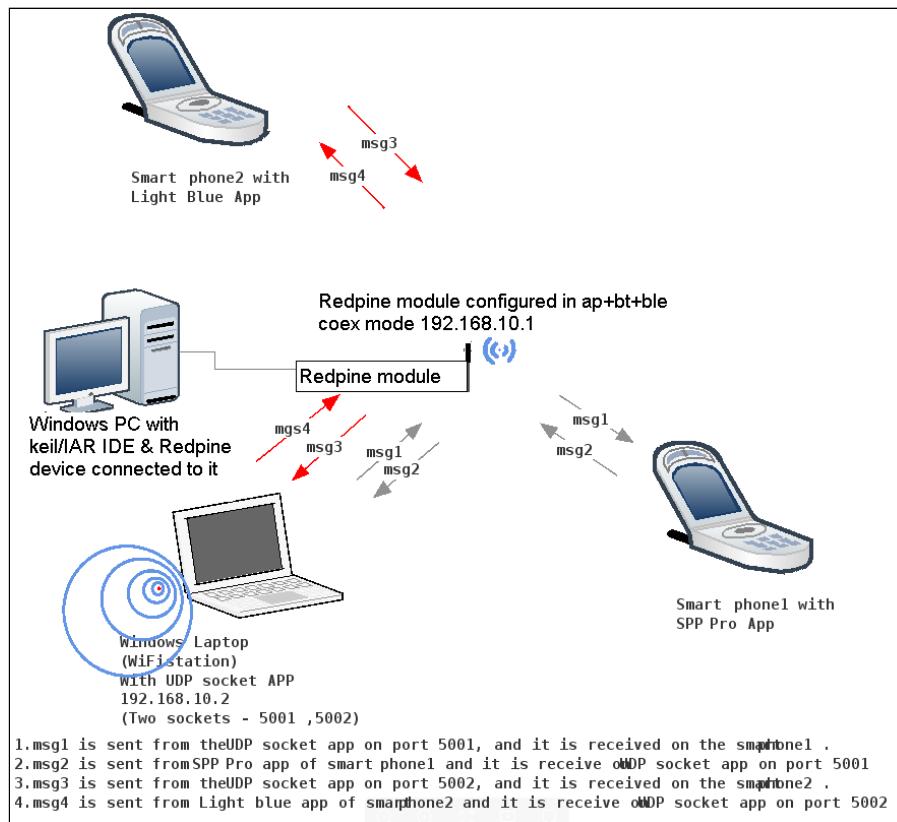
- Create chat service
- Configure device in advertise mode
- Connect from Smart phone
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

#### 7.1.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Smart phone/tablet with BT Application (Ex: Bluetooth SPP Pro)
- Smart phone/tablet with BLE Application (Ex: Light Blue APP)
- WiFi client device (PC) with UDP client application.



**Figure 1: Setup diagram**

### 7.1.3 Configuration and Execution of the Application

#### Configuring the Application

#### Configuring the WLAN task

1. Open **rsi\_wlan\_ap\_app.c** file and update/modify following macros to start an Access-point. **SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which AP would be started

```
#define CHANNEL_NO      <channel_no>
```

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities

```
#define SECURITY_TYPE <security_type>
```

**ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports Open, TKIP, CCMP methods

```
#define ENCRYPTION_TYPE <encryption_type>
```

**PSK** refers to the secret key if the Access point to be configured in WPA/WPA2 security modes.

```
#define PSK "<psk>"
```

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons

```
#define BEACON_INTERVAL <beacon_interval>
```

To configure DTIM interval of the Access Point

```
#define DTIM_INTERVAL <dtim_interval>
```

To configure static IP address

IP address to be configured to the device should be in long format and in little endian byte order.  
Example: To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP 0X010AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY 0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK 0x00FFFFFF
```

To establish UDP connection and transfer data to the remote socket, configure the below macros. Internal socket port number.

#define DEVICE_PORT_1	5001
-----------------------	------

Port number of the remote server

#define SERVER_PORT_1	5001
-----------------------	------

IP address of the remote server

#define DEVICE_PORT_2	5002
-----------------------	------

Port number of the remote server

#define SERVER_PORT_2	5002
-----------------------	------

IP address of the remote server

#define SERVER_IP_ADDRESS	0x020AA8C0
---------------------------	------------

Application memory length which is required by the driver

#define GLOBAL_BUFF_LEN	10000
-------------------------	-------

Open two UDP sockets on remote peer (PC).

User can use Test UDP application from below link to create UDP socket on remote peer <http://sourceforge.net/projects/sockettest/files/latest/download>

Include rsi\_wlan\_ap\_app.c, rsi\_ble\_app.c, rsi\_bt\_app.c and main.c files in the project, build and launch the application.

2. Open ***rsi\_wlan\_config.h*** file and update/modify following macros,

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_ENABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	TCP_IP_FEAT_BYPASS
#define RSI_CUSTOM_FEATURE_BIT_MAP	
FEAT_CUSTOM_FEAT_EXTENTION_VALID	
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP	EXT_FEAT_256K_MODE
#define RSI_BAND	RSI_BAND_2P4GHZ

## 2. OConfiguring the BT task

1. Open **rsi\_bt\_app.c** file and update/modify following macros :
  - RSI\_BT\_LOCAL\_NAME - Name of the Wyzbee (Master) device
  - PIN\_CODE - Four byte string required for pairing process.

Following are the **non-configurable** Macros in the Application file.

- BT\_GLOBAL\_BUFF\_LEN – Number of bytes required for the Application and the Driver.
- RSI\_APP\_EVENT\_CONNECTED - Event number to be set on connection establishment.
- RSI\_APP\_EVENT\_DISCONNECTED - Event number to be set on disconnection.
- RSI\_APP\_EVENT\_PINCODE\_REQ - Event number to be set on Pincode request for pairing.
- RSI\_APP\_EVENT\_LINKKEY\_SAVE - Event number to be set on link key save.
- RSI\_APP\_EVENT\_AUTH\_COMPLT - Event number to be set on authentication complete.
- RSI\_APP\_EVENT\_LINKKEY\_REQ - Event number to be set on link key request for connection.
- RSI\_APP\_EVENT\_SPP\_CONN - Event number to be set on SPP connection.
- RSI\_APP\_EVENT\_SPP\_DISCONN - Event number to be set on SPP disconnection.
- RSI\_APP\_EVENT\_SPP\_RX - Event number to be set on SPP data received from Master.

## Configuring the BLE Application

1. Open **rsi\_ble\_app.c** file and update/modify following macros,
  - RSI\_BLE\_NEW\_SERVICE\_UUID - The attribute value of the newly created service. Ex: 0xAABB
  - RSI\_BLE\_ATTRIBUTE\_1\_UUID - The attribute type of the first attribute under this Service. Ex: 0x1AA1
  - RSI\_BLE\_ATTRIBUTE\_2\_UUID - The attribute type of the second attribute under this Service. Ex: 0x1BB1
  - RSI\_BLE\_MAX\_DATA\_LEN - Maximum length of the attribute data(limited to max of 20 bytes)
  - RSI\_BLE\_SIMPLE\_CHAT\_APP\_NAME - Name of the Redpine device to appear during Scanning by peer devices.

Following are the **non-configurable** macros in the application.

- RSI\_BLE\_ATT\_PROPERTY\_READ – Used to set read property to an attribute value.
- RSI\_BLE\_ATT\_PROPERTY\_WRITE - Used to set write property to an attribute value.
- RSI\_BLE\_ATT\_PROPERTY\_NOTIFY - Used to set notify property to an attribute value.
- RSI\_BLE\_CHAR\_SERV\_UUID - The attribute type of the characteristics to be added in a service. Ex: 0x2803
- RSI\_BLE\_CLIENT\_CHAR\_UUID - The attribute type of the client characteristics descriptor to be added in a service characteristic. Ex: 0x2902
- BT\_GLOBAL\_BUFF\_LEN – Number of bytes required for the Application and the Driver.

## Executing the coex Application

1. Connect Redpine device to the Windows PC running KEIL or IAR IDE
2. Build and launch the application.
3. After the program gets executed, Redpine BT is in Discoverable state, BLE is in Advertising state and WLAN will create an Access Point and wait for remote station to connect.
4. Remote PC can connect to device Access point and device will establish UDP connection with peer (PC).
5. Now initiate connection from the SPP App running in the Smartphone1.
6. In the App, Redpine BT would appear with the name configured in the macro **RSI\_BT\_LOCAL\_NAME**.
7. After BT SPP connection is established, send a message from the App to Redpine BT. Observe this message on the PC (Socket Test GUI) connected via UDP socket (In this example port 5002) with Redpine WLAN.
8. Open a LE App in the Smartphone and do Scan.
9. In the App, Redpine BLE would appear with the name configured in the macro **RSI\_BLE\_SIMPLE\_CHAT\_APP\_NAME**.
10. Initiate BLE connection from the App.
11. Send a message from PC to Access point (Redpine device) over UDP socket (In this example port no 5001). This message is received by Access point and is sent to BLE task which in turn is sent to Smart phone2/tablet connected.
12. Now, send a message from PC (Socket Test GUI) to Redpine WLAN via UDP socket (port 5001) and observe the same in the Smartphone2.
13. `rsi_bt_app_send_to_wlan()` function defined in `rsi_wlan_ap_app.c` to send message from BT task to WLAN task.
14. With the help of wlan task, message is transferred to PC.
15. Message from PC to WLAN task via socket(5002) and `rsi_wlan_app_send_to_bt()` function defined in `rsi_bt_app.c` called asynchronously to send message from WLAN task to BT task. From BT task message transferred to smart phone1
16. Message from PC to WLAN task via socket(5001) and `rsi_wlan_app_send_to_ble()` function defined in `rsi_ble_app.c` called asynchronously to send message from WLAN task to BLE task. From BLE task message transferred to smart phone1
17. Thus messages can be seamlessly transferred between PC and Smartphones.

## 7.2 WLAN-AP BT or BLE Bridge TCPIP Bypass Example

### 7.2.1 Application Overview

#### Overview

The coex application demonstrates how information can be exchanged seamlessly using wireless protocols (WLAN, BT and BLE) running in the same device.

#### Description

The coex application has WLAN, BLE and BT tasks and acts as an interface between Smartphones and PC. Smartphone1 interacts with BT task, Smartphone2 interacts with BLE task , while PC Redpine module create an Access Point, remote PC (with WLAN station interface) connect to Redpine Access Point interacts with WLAN task When Smartphone2 connects and sends message to Redpine device, BLE task accepts and sends to WLAN task, which in turn sends to PC on UDP port configured (In this example port 5001). When Smartphone1 connects and sends message to Redpine device, BT task accepts and sends to WLAN task, which in turn sends to PC on UDP port configured (In this example port 5002). Similarly, when PC sends message to Redpine device on UDP port (In this example 5001), the message will be sent to Smartphone2 via BLE task.

And, when PC sends message to Redpine device on UDP port(In this example 5002), the message will be sent to Smartphone1 via BT task.  
Thus messages can be seamlessly transferred between PC and Smartphones.

## Details of the Application

Redpine WLAN acts as Access Point and allow stations to connect.  
RedpineBT acts as a slave device with SPP profile running in it, while Smart phone acts as a Master device with SPP profile running in it.  
Redpine BLE acts as a Peripheral (Slave) device with GATT Server running in it, while Smart phone acts as a Central (Master) device with GATT Client running in it.  
Initially, proprietary Simple chat service is created with SPP profile (Redpine device) to facilitate message exchanges.

- The WLAN task (running in Redpine device) mainly includes following steps.
  1. Start up as Access point and allow stations to connect
  2. Open two UDP sockets and receive/send data
- The BT task (running in Redpine device) mainly includes following steps.
  1. Creates chat service
  2. Configures the device in Discoverable mode and Connectable mode.
- The BLE task (running in Redpine device) mainly includes following steps.
  1. Creates chat service
  2. Configures the device to Advertise

WLAN, BLE and BT tasks forever run in the application to serve the asynchronous events.

## Sequence of Events

### WLAN Task

This Application explains user how to:

- Create device as an Access Point
- Connect stations to Redpine Access Point
- Open TCP server socket in device
- Receive TCP data sent by connected station and forward to BLE task
- Send data received by BLE task to connected station using TCP protocol

### BT Task

This Application explains user how to:

- Configure Redpine device to SPP profile mode
- Configure device in discoverable and connectable mode
- Establish SPP profile level connection with remote smart phone
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

### LE Task

This Application explains user how to:

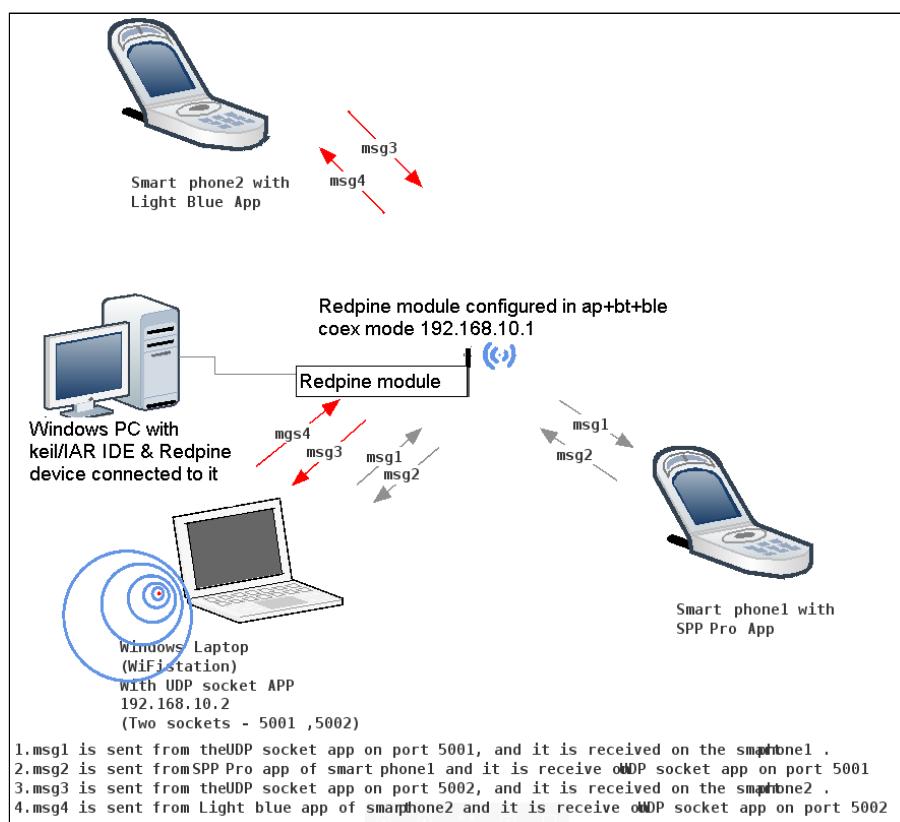
- Create chat service
- Configure device in advertise mode
- Connect from Smart phone
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

### 7.2.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Smart phone/tablet with BT Application (Ex: Bluetooth SPP Pro)
- Smart phone/tablet with BLE Application (Ex: Light Blue APP)
- WiFi client device (PC) with UDP client application.



**Figure 1: Setup diagram**

### 7.2.3 Configuration and Execution of the Application

#### Configuring the Application

##### Configuring the WLAN task

1. Open **rsi\_wlan\_ap\_app.c** file and update/modify following macros to start an Access-point.  
**SSID** refers to the name of the Access point .

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO**refers to the channel in which AP would be started

```
#define CHANNEL_NO      <channel_no>
```

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities

```
#define SECURITY_TYPE    <security_type>
```

**ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports Open, TKIP, CCMP methods

```
#define ENCRYPTION_TYPE   <encryption_type>
```

**PSK** refers to the secret key if the Access point to be configured in WPA/WPA2 security modes.

```
#define PSK             "<psk>"
```

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons

```
#define BEACON_INTERVAL    <beacon_interval>
```

To configure DTIM interval of the Access Point

```
#define DTIM_INTERVAL
```

```
<dtim_interval>
```

To configure static IP address

IP address to be configured to the device should be in long format and in little endian byte order.

Example: To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP
```

```
0X010AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY
```

```
0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK
```

```
0x00FFFFFF
```

To establish UDP connection and transfer data to the remote socket, configure the below macros. Internal socket port number.

```
#define DEVICE_PORT_1
```

```
5001
```

Port number of the remote server

```
#define SERVER_PORT_1
```

```
5001
```

IP address of the remote server

```
#define DEVICE_PORT_2
```

```
5002
```

Port number of the remote server

```
#define SERVER_PORT_2
```

```
5002
```

IP address of the remote server

#define SERVER\_IP\_ADDRESS

0x020AA8C0

Application memory length which is required by the driver

#define GLOBAL\_BUFF\_LEN

10000

Open two UDP sockets on remote peer (PC).

User can use Test UDP application from below link to create UDP socket on remote peer <http://sourceforge.net/projects/sockettest/files/latest/download>

Include rsi\_wlan\_ap\_app.c, rsi\_ble\_app.c ,rsi\_bt\_app.c and main.c files in the project, build and launch the application.

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_ENABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	TCP_IP_FEAT_BYPASS
#define RSI_CUSTOM_FEATURE_BIT_MAP	
FEAT_CUSTOM_FEAT_EXTENSION_VALID	
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP	EXT_FEAT_256K_MODE
#define RSI_BAND	RSI_BAND_2P4GHZ

2. OConfiguring the BT task

1. Open **rsi\_bt\_app.c** file and update/modify following macros :

- RSI\_BT\_LOCAL\_NAME - Name of the Wyzbee (Master) device
- PIN\_CODE - Four byte string required for pairing process.

Following are the **non-configurable** Macros in the Application file.

- BT\_GLOBAL\_BUFF\_LEN – Number of bytes required for the Application and the Driver.
- RSI\_APP\_EVENT\_CONNECTED - Event number to be set on connection establishment.
- RSI\_APP\_EVENT\_DISCONNECTED - Event number to be set on disconnection.
- RSI\_APP\_EVENT\_PINCODE\_REQ - Event number to be set on Pincode request for pairing.
- RSI\_APP\_EVENT\_LINKKEY\_SAVE - Event number to be set on link key save.
- RSI\_APP\_EVENT\_AUTH\_COMPLT - Event number to be set on authentication complete.
- RSI\_APP\_EVENT\_LINKKEY\_REQ - Event number to be set on link key request for connection.
- RSI\_APP\_EVENT\_SPP\_CONN - Event number to be set on SPP connection.
- RSI\_APP\_EVENT\_SPP\_DISCONN - Event number to be set on SPP disconnection.
- RSI\_APP\_EVENT\_SPP\_RX - Event number to be set on SPP data received from Master.

Configuring the BLE Application

1. Open **rsi\_ble\_app.c** file and update/modify following macros,

- RSI\_BLE\_NEW\_SERVICE\_UUID - The attribute value of the newly created service. Ex: 0xAABB

- RSI\_BLE\_ATTRIBUTE\_1\_UUID - The attribute type of the first attribute under this Service. Ex: 0x1AA1
- RSI\_BLE\_ATTRIBUTE\_2\_UUID - The attribute type of the second attribute under this Service. Ex: 0x1BB1
- RSI\_BLE\_MAX\_DATA\_LEN - Maximum length of the attribute data(limited to max of 20 bytes)
- RSI\_BLE\_SIMPLE\_CHAT\_APP\_NAME - Name of the Redpine device to appear during Scanning by peer devices.

Following are the **non-configurable** macros in the application.

- RSI\_BLE\_ATT\_PROPERTY\_READ – Used to set read property to an attribute value.
- RSI\_BLE\_ATT\_PROPERTY\_WRITE – Used to set write property to an attribute value.
- RSI\_BLE\_ATT\_PROPERTY\_NOTIFY – Used to set notify property to an attribute value.
- RSI\_BLE\_CHAR\_SERV\_UUID - The attribute type of the characteristics to be added in a service. Ex: 0x2803
- RSI\_BLE\_CLIENT\_CHAR\_UUID - The attribute type of the client characteristics descriptor to be added in a service characteristic. Ex: 0x2902
- BT\_GLOBAL\_BUFF\_LEN – Number of bytes required for the Application and the Driver.

## Executing the coex Application

1. Connect Redpine device to the Windows PC running KEIL or IAR IDE
2. Build and launch the application.
3. After the program gets executed, Redpine BT is in Discoverable state, BLE is in Advertising state and WLAN will create an Access Point and wait for remote station to connect.
4. Remote PC can connect to device Access point and device will establish UDP connection with peer (PC).
5. Now initiate connection from the SPP App running in the Smartphone1.
6. In the App, Redpine BT would appear with the name configured in the macro **RSI\_BT\_LOCAL\_NAME**.
7. After BT SPP connection is established, send a message from the App to Redpine BT. Observe this message on the PC (Socket Test GUI) connected via UDP socket (In this example port 5002) with Redpine WLAN.
8. Open a LE App in the Smartphone and do Scan.
9. In the App, Redpine BLE would appear with the name configured in the macro  
**RSI\_BLE\_SIMPLE\_CHAT\_APP\_NAME**.
10. Initiate BLE connection from the App.
11. Send a message from PC to Access point (Redpine device) over UDP socket (In this example port no 5001). This message is received by Access point and is sent to BLE task which in turn is sent to Smart phone2/tablet connected.
12. Now, send a message from PC (Socket Test GUI) to Redpine WLAN via UDP socket (port 5001) and observe the same in the Smartphone2.
13. rsi\_bt\_app\_send\_to\_wlan() function defined in rsi\_wlan\_ap\_app.c to send message from BT task to WLAN task.
14. With the help of wlan task, message is transferred to PC.
15. Message from PC to WLAN task via socket(5002) and rsi\_wlan\_app\_send\_to\_bt() function defined in rsi\_bt\_app.c called asynchronously to send message from WLAN task to BT task. From BT task message transferred to smart phone1
16. Message from PC to WLAN task via socket(5001) and rsi\_wlan\_app\_send\_to\_ble() function defined in rsi\_ble\_app.c called asynchronously to send message from WLAN task to BLE task. From BLE task message transferred to smart phone1
17. Thus messages can be seamlessly transferred between PC and Smartphones.

## 7.3 WLAN-STA BT and BLE Bridge TCPIP Bypass with Dual Role Example

### 7.3.1 Application Overview

#### Overview

The coex application demonstrates how information can be exchanged seamlessly using wireless protocols (WLAN, BT and BLE) running in the same device.

#### Description

The coex application has WLAN, BLE and BT tasks and acts as an interface between Smartphones and PC.

In this coex application, Redpine BT device connects with remote BT device (Smart Phone), Smartphone and sensor tag interacts with BLE task, and Redpine WiFi interface connects with an Access Point in station mode and do data transfer in BT and WiFi interfaces. The coex application has WLAN, BT and BLE tasks and acts as an interface between remote Smartphone BT device, sensor tag and remote PC which is connected to Access point. Smartphone interacts with BT task, Smartphone and sensor tag interacts with BLE task, while remote PC interacts with WLAN task. Thus messages can be seamlessly transferred between PC and Smartphone.

#### Details of the Application

Redpine WLAN acts as a Station and connects to an Access Point

Redpine BT acts as a slave device with SPP profile running in it, while Smart phone acts as a Master device with SPP profile running in it.

Redpine BLE acts as a Peripheral (Slave) device with GATT Server running in it, while Smart phone acts as a Central (Master) device with GATT Client running in it.

Initially, proprietary Simple chat service is created with SPP profile (Redpine device) to facilitate message exchanges.

- The WLAN task (running in Redpine device) mainly includes following steps.
  - a. Connects to a Access Point
  - b. Exchanges data over SSL socket with the peer(Windows PC)
- The BT task (running in Redpine device) mainly includes following steps.
  - a. Creates chat service
  - b. Configures the device in Discoverable mode and Connectable mode.
- The BLE task (running in Redpine device) mainly includes following steps.
  - a. Creates chat service
  - b. Configures the device to Advertise

WLAN, BLE and BT tasks forever run in the application to serve the asynchronous events.

#### Sequence of Events

#### WLAN Task

This Application explains user how to:

- Create Redpine device as Station
- Connect Redpine station to remote Access point
- Receive TCP data sent by connected station and forward to BT task
- Send data received by BT task to connected station using TCP protocol.

## BT Task

This Application explains user how to:

- Configure Redpine device to SPP profile mode
- Configure device in discoverable and connectable mode
- Establish SPP profile level connection with remote smart phone
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

## BLE Task

This Application explains user how to:

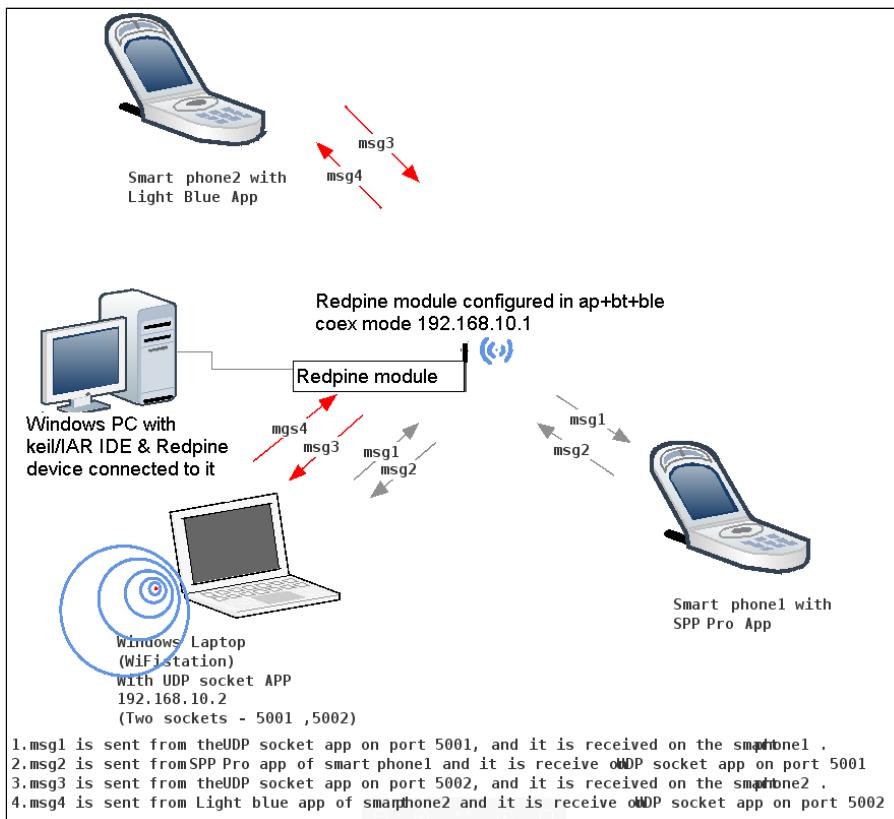
- Create chat service
- Configure device in advertise mode
- Connect from Smart phone
- Configure device in master mode
- Connect to remote device
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

### 7.3.2 Application Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Smart phone/tablet with BT Application (Ex: Bluetooth SPP Pro)
- Smart phone/tablet with BLE Application (Ex: Light Blue APP)
- WiFi client device (PC) with UDP client application.



**Figure 1: Setup diagram**

### 7.3.3 Configuration and Execution of the Application

#### Configuring the Application

#### Configuring the WLAN task

1. Open **rsi\_wlan\_app.c** file and update/modify following macros,  
**SSID** refers to the Access point to which user wants to connect.  
**SECURITY\_TYPE** is the security type of the Access point.  
**PSK** refers to the secret key if the Access point is configured in WPA/WPA2 security modes.

```
#define SSID                                "<ap_name>"  

#define SECURITY_TYPE                         <security-type>  

#define PSK                                     "
```

Load the SSL CA- certificate using **rsi\_wlan\_set\_certificate API** after wireless initialization.

**Note:**

rsi\_wlan\_set\_certificate expects the certificate in the form of linear array. Python script is provided in the release package named "**certificate\_script.py**" in the following path "certificates" to convert the pem certificate into linear array

Example: If the certificate is ca-cert.pem, give the command as  
python certificate\_script.py ca-cert.pem

The script will generate cacert.pem, in which one linear array named *cacert* contains the certificate  
Enable/Disable DHCP mode

- 1 – Enables DHCP mode (gets the IP from DHCP server)
- 0 – Disables DHCP mode

#define DHCP\_MODE

1

If DHCP mode is disabled, then change the following macros to configure static IP address

IP address to be configured to the device should be in long format and in little endian byte order.

Example: To configure "192.168.10.101" as IP address, update the macro **DEVICE\_IP** as **0x650AA8C0**.

#define DEVICE\_IP

0x650AA8C0

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

#define GATEWAY

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK

0x00FFFFFF

To establish TCP connection and transfer data to the remote socket configure the below macros. If SSL is enabled, open the socket with protocol type as 1.

Internal socket port number.

#define DEVICE\_PORT

5001

Port number of the remote server

#define SERVER\_PORT

5001

IP address of the remote server

#define SERVER\_IP\_ADDRESS

0x650AA8C0

Include rsi\_wlan\_app.c , rsi\_ble\_app.c and main.c files in the project, build and launch the application  
Open SSL server socket on remote machine

For example, to open SSL socket with port number 5001 on remote side, use the command as given below  
**openssl s\_server -accept<SERVER\_PORT> -cert <server\_certificate\_file\_path> -key <server\_key\_file\_path> -tls<tls\_version>**

**Example:** openssl s\_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1\_2

**Note:**

All the certificates are given in the release package

2. Enable/Disable power save  
1 – Enables Power save mode  
0 – Disables Power save mode

```
#define WLAN_POWER_SAVE 0
```

3. Open **rsi\_wlan\_config.h** file and update/modify following macros,

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP (TCP_IP_FEAT_DHCPV4_CLIENT   TCP_IP_TOTAL_SOCKETS_1   TCP_IP_FEAT_SSL)	
#define RSI_CUSTOM_FEATURE_BIT_MAP FEAT_CUSTOM_FEAT_EXTENTION_VALID	
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP	EXT_FEAT_256K_MODE
#define RSI_BAND	RSI_BAND_2P4GHZ

## Configuring the BT task

1. Open **rsi\_bt\_app.c** file and update/modify following macros:
  - RSI\_BT\_LOCAL\_NAME - Name of the Wyzbee (Master) device
  - PIN\_CODE - Four byte string required for pairing process.
2. Following are the **non-configurable** Macros in the Application file.
  - BT\_GLOBAL\_BUFF\_LEN – Number of bytes required for the Application and the Driver.
  - RSI\_APP\_EVENT\_CONNECTED - Event number to be set on connection establishment.
  - RSI\_APP\_EVENT\_DISCONNECTED - Event number to be set on disconnection.
  - RSI\_APP\_EVENT\_PINCODE\_REQ - Event number to be set on Pincode request for pairing.
  - RSI\_APP\_EVENT\_LINKKEY\_SAVE - Event number to be set on link key save.
  - RSI\_APP\_EVENT\_AUTH\_COMPLT - Event number to be set on authentication complete.
  - RSI\_APP\_EVENT\_LINKKEY\_REQ - Event number to be set on link key request for connection.
  - RSI\_APP\_EVENT\_SPP\_CONN - Event number to be set on SPP connection.
  - RSI\_APP\_EVENT\_SPP\_DISCONN - Event number to be set on SPP disconnection.
  - RSI\_APP\_EVENT\_SPP\_RX - Event number to be set on SPP data received from Master.

## Configuring the BLE Application

1. Open **rsi\_ble\_app.c** file and update/modify following macros,
  - RSI\_BLE\_NEW\_SERVICE\_UUID - The attribute value of the newly created service. Ex: 0xAABB
  - RSI\_BLE\_ATTRIBUTE\_1\_UUID - The attribute type of the first attribute under this Service. Ex: 0x1AA1
  - RSI\_BLE\_ATTRIBUTE\_2\_UUID - The attribute type of the second attribute under this Service. Ex: 0x1BB1
  - RSI\_BLE\_MAX\_DATA\_LEN - Maximum length of the attribute data(limited to max of 20 bytes)
  - RSI\_BLE\_APP\_DEVICE\_NAME - Name of the Redpine device to appear during Scanning by peer devices.
  - BLE\_PS\_ENABLE – To Enable/Disable power save.
  - RSI\_BLE\_DEV\_ADDR – Address of the peer device to connect.
  - BLE\_DUAL\_ROLE\_FIRST\_MASTER – To create local device as a master first.
2. Following are the **non-configurable** macros in the application.
  - RSI\_BLE\_ATT\_PROPERTY\_READ – Used to set read property to an attribute value.
  - RSI\_BLE\_ATT\_PROPERTY\_WRITE – Used to set write property to an attribute value.
  - RSI\_BLE\_ATT\_PROPERTY\_NOTIFY – Used to set notify property to an attribute value.
  - RSI\_BLE\_CHAR\_SERV\_UUID - The attribute type of the characteristics to be added in a service. Ex: 0x2803
  - RSI\_BLE\_CLIENT\_CHAR\_UUID - The attribute type of the client characteristics descriptor to be added in a service characteristic. Ex: 0x2902
  - BT\_GLOBAL\_BUFF\_LEN – Number of bytes required for the Application and the Driver.

## Executing the coex Application

1. Connect Redpine device to the Windows PC running KEIL or IAR IDE
2. Build and launch the application.
3. Advertise 3<sup>rd</sup> party TI sensor tag.
4. After the program gets executed, Redpine BT is in Discoverable state, If BLE\_DUAL\_ROLE\_FIRST\_MASTER macro as 1 then Redpine BLE is in scanning state and it creates a connection with TI sensor tag. If BLE\_DUAL\_ROLE\_FIRST\_MASTER macro as 0 then Redpine BLE is in Advertising state.
5. Remote PC can connect to device Access point and device will establish UDP connection with peer (PC).
6. Now initiate connection from the SPP App running in the Smartphone1.
7. In the App, Redpine BT would appear with the name configured in the macro **RSI\_BT\_LOCAL\_NAME**.
8. After BT SPP connection is established, send a message from the App to Redpine BT. Observe this message on the PC (Socket Test GUI) connected via UDP socket (In this example port 5002) with Redpine WLAN.
9. Open a LE App in the Smartphone and do Scan.
10. In the App, Redpine BLE would appear with the name configured in the macro **RSI\_BLE\_SIMPLE\_CHAT\_APP\_NAME**.
11. Initiate BLE connection from the App.
12. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
13. WLAN connects to Access Point and establishes SSL connectivity with SSL server opened on Windows PC1. Please refer the given below image for connection establishment at windows PC1, send a message or notification from the App to Redpine BLE. Observe this message in the PC connected via SSL socket with Redpine WLAN.
  - rsi\_ble\_app\_send\_to\_wlan() function defined in rsi\_wlan\_app.c to send message from BLE task to WLAN task.

---

## 8 WLAN

### 8.1 WLAN Power Numbers Examples

#### 8.1.1 Tx on periodic wakeup

##### Example

##### Overview

Tx on periodic wakeup test application demonstrates power consumption of Redpine device in Connected sleep mode having UDP server socket in open state.

M4 wakes up with alarm configured time and sends packet to NWP, NWP wakes up with packet from M4 then process packet after that both will go to sleep.

##### Sequence of Events

This Application explains user how to:

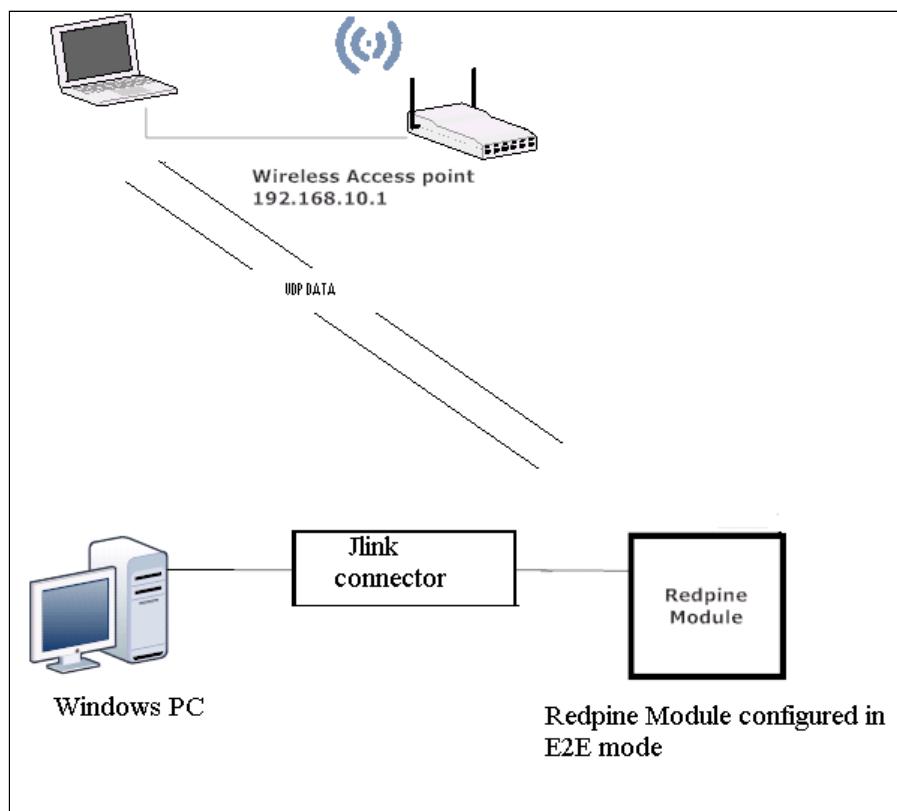
- Connect to an Access point using wireless connectivity.
- Open UDP client socket ,Trigger M4 & NWP to sleep.
- M4 wakes up with alarm configured time and then send packet to NWP.
- NWP wakes up and process packet.
- Both NWP and M4 goes to sleep.
- Above three steps were repeat continuously.

##### Example Setup

The WiSeMCU device offer integrated wireless connectivity and does not require host interface initialization.

##### WiSeMCU Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module
- Jlink connector
- Access point
- UDP server application



**Figure 1: Setup Diagram**

## Configuration and Execution of the Application

### Configuring the Application

1. Open `rsi_tx_on_periodic_wakeup.c` file and update / modify following macros:

```
#define SSID                                "ap_name"  
  
#define SCAN_CHANNEL                         <channel_no>  
  
//! Security type  
  
#define SECURITY_TYPE                      RSI_OPEN  
  
//! Password  
  
#define PSK                                 NULL  
  
//! DHCP mode 1- Enable 0- Disable  
  
#define DHCP_MODE                           1  
  
//! If DHCP mode is disabled given IP statically  
  
#if !(DHCP_MODE)  
  
    //! IP address of the module  
  
    //! E.g: 0x650AA8C0 == 192.168.10.101  
  
    #define DEVICE_IP                          0x650AA8C0  
  
    //! IP address of Gateway  
  
    //! E.g: 0x010AA8C0 == 192.168.10.1  
  
    #define GATEWAY                           0x010AA8C0  
  
    //! IP address of netmask  
  
    //! E.g: 0x00FFFFFF == 255.255.255.0  
  
    #define NETMASK                           0x00FFFFFF  
  
#endif  
  
//! Server port number  
  
#define SERVER_PORT                         5001
```

```
//! Device port number  
  
#define DEVICE_PORT 5001  
  
//! Server IP address. Should be in reverse long format  
  
//! E.g: 0x640AA8C0 == 192.168.10.100  
  
#define SERVER_IP_ADDRESS 0x640AA8C0
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	
FEAT_CUSTOM_FEAT_EXTENSION_VALID	
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP	
EXT_FEAT_ENABLE_LOW_POWER_MODE	
#define RSI_BAND	RSI_BAND_2P4GHZ
#define RSI_HAND_SHAKE_TYPE	M4_BASED
#define RSI_SELECT_LP_OR_ULP_MODE	RSI_ULP_WITH_RAM_RET
#define PLL_MODE	0
#define RF_TYPE	1
#define WIRELESS_MODE	0
#define ENABLE_PPP	0
#define AFE_TYPE	1
#define FEATURE_ENABLES	48

## Executing the Application

1. Make sure Configuration settings as mentioned above before do compile&run the Application.
2. After successful connection with access point ,module gets IP either using DHCP or Static based on configuration settings.
3. Open UDP server socket on the remote side.

- 
4. Our module creates UDP client socket.
  5. Trigger NWP to sleep with power mode 2 with ram retention followed by trigger M4 to sleep with ram retention. M4 wakes up with alarm configured time and NWP wakes up with packet from M4.
  6. Periodically M4 wakes up then send data to NWP.
  7. NWP wakes up and process the packet and goes back to sleep.
  8. The above two steps repeats continuously till data transfer is in progress.

Note: Debug prints support is added so to enable prints use DEBUG\_UART define. Connect usb cable to UART port on EVK baseboard and open cutecom with a baudrate of 115200.

### 8.1.2 WakeOn Wireless

#### Example

#### Overview

Wake On Wireless test application demonstrates the connected sleep power save functionality of Redpine device having TCP server socket in open state.

M4 will be in retention mode sleep and wakes up whenever there is a packet from NWP (Wireless Processor).

#### Sequence of Events

This Application explains user how to:

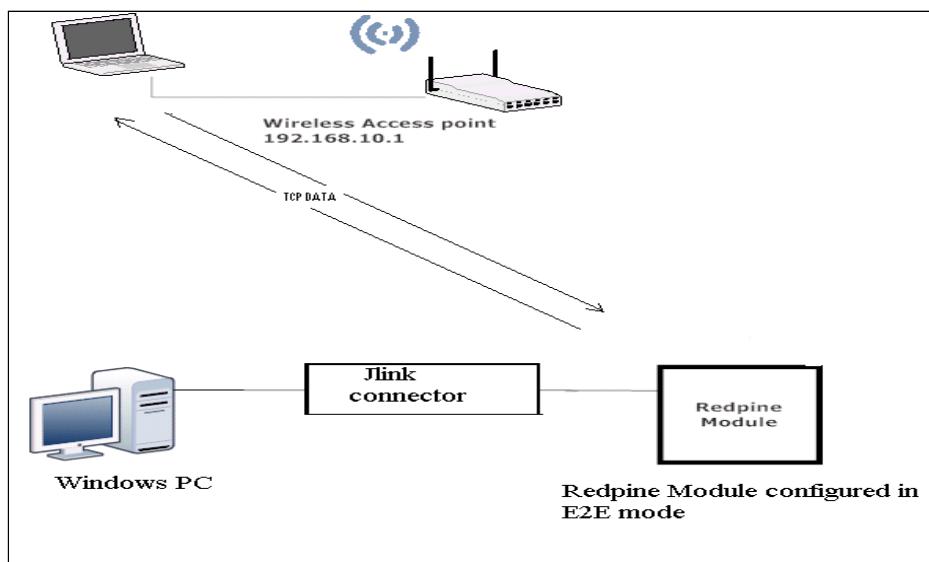
1. Connect to an access point using wireless connectivity.
2. Open TCP server socket and waits for the TCP client connection.
3. After the TCP connection from remote peer, Trigger NWP and M4 sleep.
4. NWP is in Connected Sleep mode & wakes up on DTIM interval.
5. M4 is in ram retention based sleep and waits for wireless based wakeup from NWP.
6. If any packet is pending from NWP, NWP wakes up M4, posts the packet to M4 and goes back to sleep.
7. M4 processes the packet and goes back to sleep.
8. The above four steps are repeated continuously.

#### Example Setup

The WiSeMCU device offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module
- M4 Debugger
- Access point
- TCP client application



**Figure 1: Setup Diagram**

## Configuration and Execution of the Application

### Configuring the Application

1. Open rsi\_wake\_on\_wireless.c file and update / modify following macros:

```
#define SSID                                "ap_name"  
  
#define SCAN_CHANNEL                         <channel_no>  
  
//! Security type  
  
#define SECURITY_TYPE                      RSI_OPEN  
  
//! Password  
  
#define PSK                                 NULL  
  
//! DHCP mode 1- Enable 0- Disable  
  
#define DHCP_MODE                           1  
  
//! If DHCP mode is disabled given IP statically  
  
#if !                                         DHCP_MODE)
```

```
//! IP address of the module

//! E.g: 0x650AA8C0 == 192.168.10.101

#define DEVICE_IP          0x650AA8C0

//! IP address of Gateway

//! E.g: 0x010AA8C0 == 192.168.10.1

#define GATEWAY           0x010AA8C0

//! IP address of netmask

//! E.g: 0x00FFFFFF == 255.255.255.0

#define NETMASK            0x00FFFFFF

#endif

//! Server port number

#define SERVER_PORT         5001

//! Device port number

#define DEVICE_PORT          5001

//! Server IP address. Should be in reverse long format

//! E.g: 0x640AA8C0 == 192.168.10.100

#define SERVER_IP_ADDRESS    0x640AA8C0
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	
FEAT_SECURITY_OPEN	
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	
FEAT_CUSTOM_FEAT_EXTENSION_VALID	
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP	
EXT_FEAT_ENABLE_LOW_POWER_MODE	
#define RSI_BAND	RSI_BAND_2P4GHZ
#define RSI_HAND_SHAKE_TYPE	M4_BASED
#define RSI_SELECT_LP_OR_ULP_MODE	
RSI_ULP_WITH_RAM_RET	
#define PLL_MODE	0
#define RF_TYPE	1
#define WIRELESS_MODE	0
#define ENABLE_PPP	0
#define AFE_TYPE	1
#define FEATURE_ENABLES	48

## Executing the Application

1. Make sure Configuration settings as mentioned above before to compile&run the Application.
2. After successful connection with access point ,module gets IP either using DHCP or Static based on configuration settings.
3. Then open TCP server socket and waits until a client is connected.
4. Trigger NWP to sleep with power mode 2 with ram retention followed by trigger M4 to sleep with ram retention. NWP wakes up with wireless packet and M4 wakes up with NWP packet.
5. Send data from client side then NWP wakes up and post packet to M4 after that NWP goes back to sleep.
6. M4 wakes up and process the packet then goes back to sleep.
7. The above two steps are repeated continuously.

Note: Debug prints support is added so to enable prints use DEBUG\_UART define. Connect usb cable to UART port on EVK baseboard and open cutecom with a baudrate of 115200.

### 8.1.3 Without Retention Deepsleep

## Example

### Overview

Without retention deepsleep test application demonstrates the disconnected sleep power save functionality of Redpine device.

M4 & NWP both are Without ram retain so after sleep wakeup M4 should start starting of application and has to load firmware for NWP ,these steps should happen repeatedly after sleep wakeup states.

### Sequence of Events

This Application explains user how to:

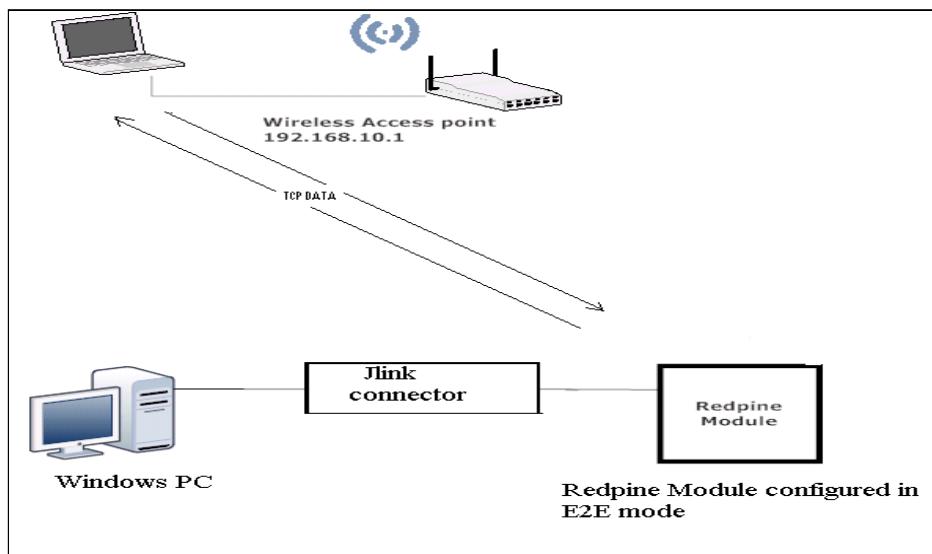
1. Scan available access points.
2. Irrespective of Scan fail/success issue power save command for NWP then for M4.
3. Both NWP & M4 are configured without ram retain.
4. After wakeup from sleep M4 will execute from start of application and load firmware for NWP.

### Example Setup

The WiSeMCU device offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module
- M4 Debugger
- Access point



### Figure 1: Setup Diagram

## Configuration and Execution of the Application

### Configuring the Application

1. Open rsi\_without\_retention\_deepsleep.c file and update / modify following macros:

```
#define SSID                                "ap_name"  
  
#define SCAN_CHANNEL                         <channel_no>  
  
//! Security type  
  
#define SECURITY_TYPE                      RSI_OPEN  
  
//! Password  
  
#define PSK                                  NULL  
  
//! DHCP mode 1- Enable 0- Disable  
  
#define DHCP_MODE                           1  
  
//! If DHCP mode is disabled given IP statically  
  
#if (!DHCP_MODE)  
  
//! IP address of the module  
  
//! E.g: 0x650AA8C0 == 192.168.10.101  
  
#define DEVICE_IP                            0x650AA8C0  
  
//! IP address of Gateway  
  
//! E.g: 0x010AA8C0 == 192.168.10.1  
  
#define GATEWAY                             0x010AA8C0  
  
//! IP address of netmask  
  
//! E.g: 0x00FFFFFF == 255.255.255.0  
  
#define NETMASK                            0x00FFFFFF  
  
#endif
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_CUSTOM_FEATURE_BIT_MAP	FEAT_CUSTOM_FEAT_EXTENTION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP	EXT_FEAT_ENABLE_LOW_POWER_MODE
#define RSI_BAND	RSI_BAND_2P4GHZ
#define RSI_HAND_SHAKE_TYPE	M4_BASED
#define RSI_SELECT_LP_OR_ULP_MODE	RSI_ULP_WITHOUT_RAM_RET
#define PLL_MODE	0
#define RF_TYPE	1
#define WIRELESS_MODE	0
#define ENABLE_PPP	0
#define AFE_TYPE	1
#define FEATURE_ENABLES	48

### Executing the Application

1. Make sure configuration settings as mentioned above before to compile&run the Application.
2. Initiate wireless init as well as band & init .
3. After success response of above ,trigger NWP to sleep with power mode 8 without ram retention followed by trigger M4 to sleep without ram retention.
4. The above two steps are repeated continuously after sleep wakeup of NWP & M4.

Note: Debug prints support is added so to enable prints use DEBUG\_UART define. Connect usb cable to UART port on EVK baseboard and open cutecom with a baudrate of 115200.

### 8.1.4 WLAN 1Mbps Rx & Listen LP

#### Example Overview

##### Overview

The Wlan 1Mbps Rx & Listen test application demonstrates how to measure power number of a Redpine device in Rx & Listen mode.

It helps us to know power consumption of our Redpine device in 1Mbps Rx & Listen LP mode.

#### Sequence of Events

This Application explains user how to:

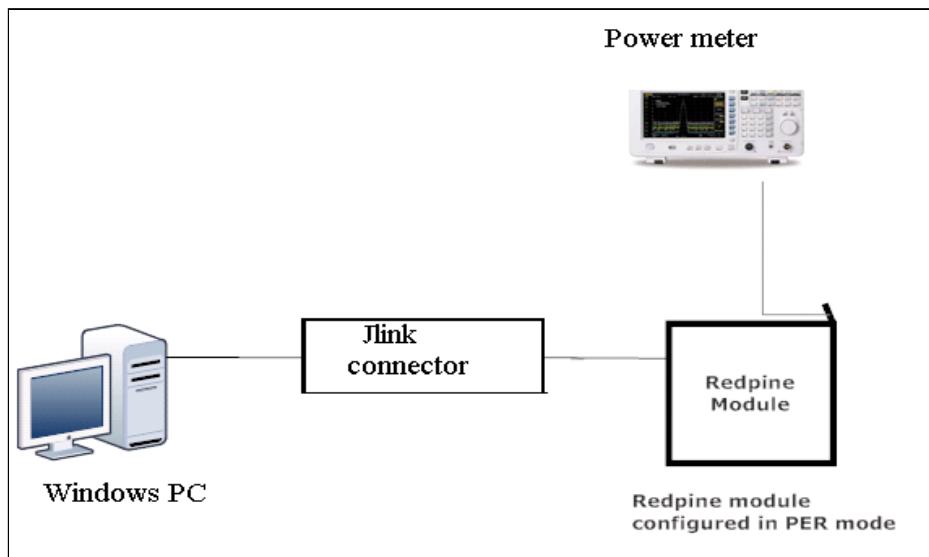
- Initialize.Radio
- Start PER stats in particular channel.
- Measure power number.

## Example Setup

The WiSeMCU device offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module
- Jlink connector
- Power Meter



**Figure 1: Setup Diagram**

## Configuration and Execution of the Application.

### Configuring the Application

1. Open **rsi\_pw\_num\_rx.c** file and update / modify following macros:

To configure the channel number in 2.4 GHz/5GHz in which PER mode has to be initialized.

```
#define RSI_TX_TEST_CHANNEL <channel_no>
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
//! To enable concurrent mode
#define CONCURRENT_MODE RSI_DISABLE

//! TCP IP BYPASS feature check
#define RSI_TCP_IP_BYPASS RSI_DISABLE

//! TCP/IP feature select bitmap for selecting TCP/IP features

#define RSI_TCP_IP_FEATURE_BIT_MAP (TCP_IP_FEAT_DHCPV4_CLIENT)

//! To set custom feature select bit map
#define RSI_CUSTOM_FEATURE_BIT_MAP 0

//! To set Extended custom feature select bit map
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP 0

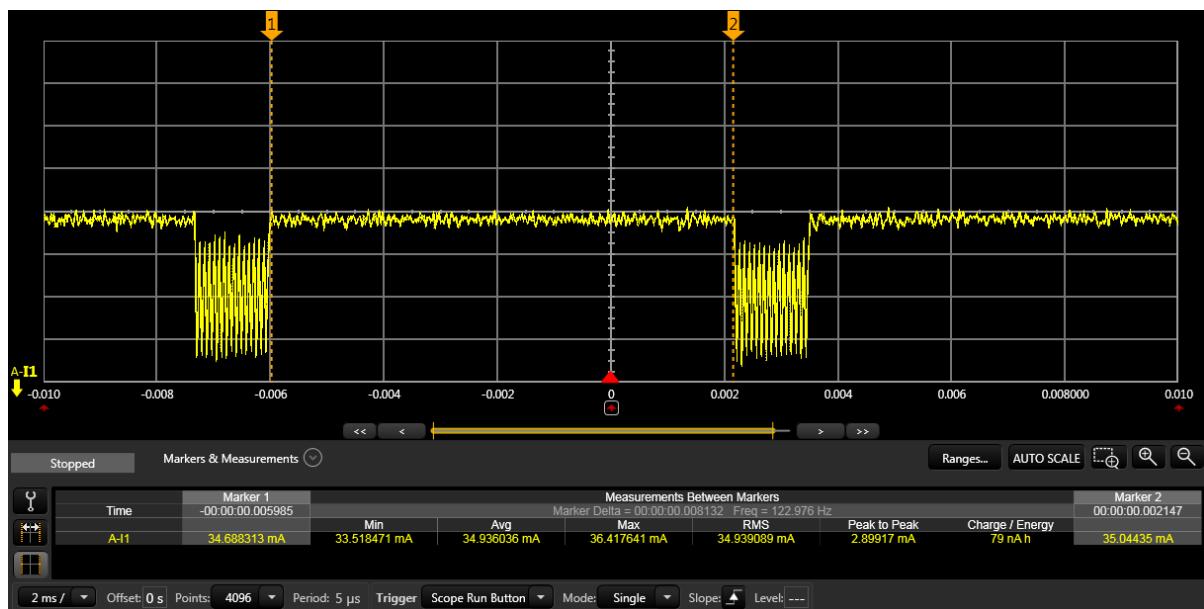
#define RSI_EXT_TCPIP_FEATURE_BITMAP 0

#define DUTY_CYCLING_ENABLE BIT(0)

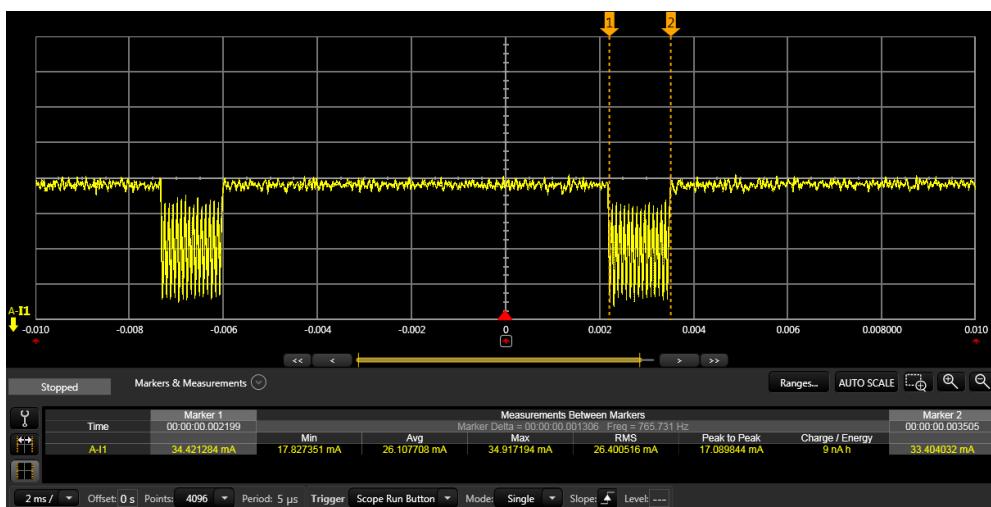
#define PLL_MODE 0
#define RF_TYPE 1 //! 0 - External RF 1- Internal RF
#define WIRELESS_MODE 12 //! 0 - HP chain 12- HP chain
#define ENABLE_PPP 0
#define AFE_TYPE 1
#define FEATURE_ENABLES DUTY_CYCLING_ENABLE
```

## Executing the Application

1. Switch on power meter and select measurement as current measure.
2. Connect the power probes of power meter to WiseMCU device.
3. Connect the Jlink connector from WiseMCU device to the Windows laptop.
4. Give power to WiseMCU device on power interface.
5. Compile Application in windows laptop. When finished, run the application.
6. PER stats starts in particular channel and make sure TX packets destined to redpine module with a rate 1Mbps.
7. Note down the power measurement as shown below.



**Fig:** wlan Rx 1Mbps with LP chain



**Fig:** Wlan Listen 1Mbps with LP chain

### 8.1.5 WLAN Listen HP

#### Example Overview

##### Overview

The Listen HP test Application demonstrates how to measure power number of a Redpine device in listen mode. It helps us to know power consumption of Redpine device in Listen HP mode.

## Sequence of Events

This Application explains user how to:

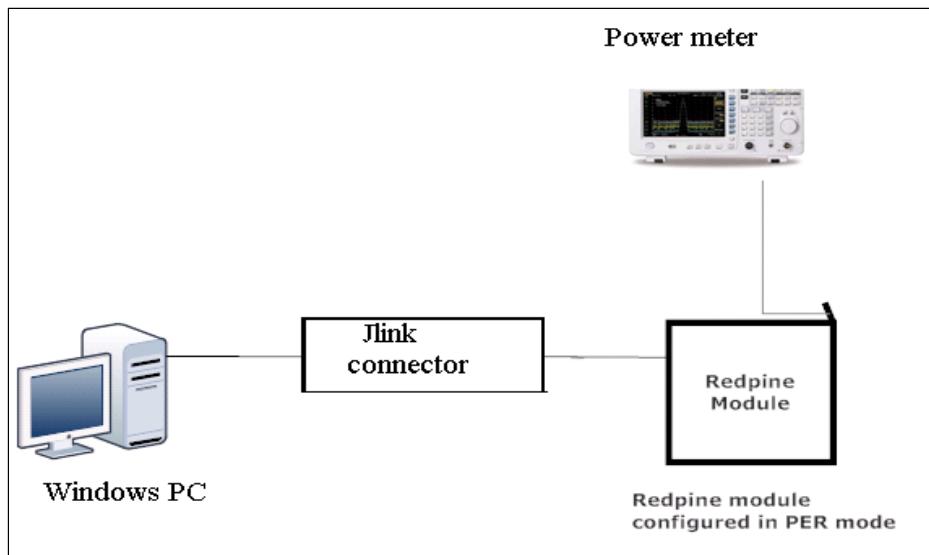
- Initialize.Radio
- Start PER stats in particular channel.
- Measure power number.

## Example Setup

The WiSeMCU device offer integrated wireless connectivity and does not require host interface initialization.

## WiSeMCU Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module
- Jlink connector
- Power Meter



**Figure 1: Setup Diagram**

## Configuration and Execution of the Application

### Configuring the Application

1. Open **rsi\_pwn\_num\_listen.c** file and update / modify following macros:  
To configure Burst mode or Continuous mode

```
#define RSI_TX_TEST_MODE
```

```
RSI_BURST_MODE
```

To configure the channel number in 2.4 GHz/5GHz in which PER mode has to be initialized.

```
#define RSI_TX_TEST_CHANNEL
```

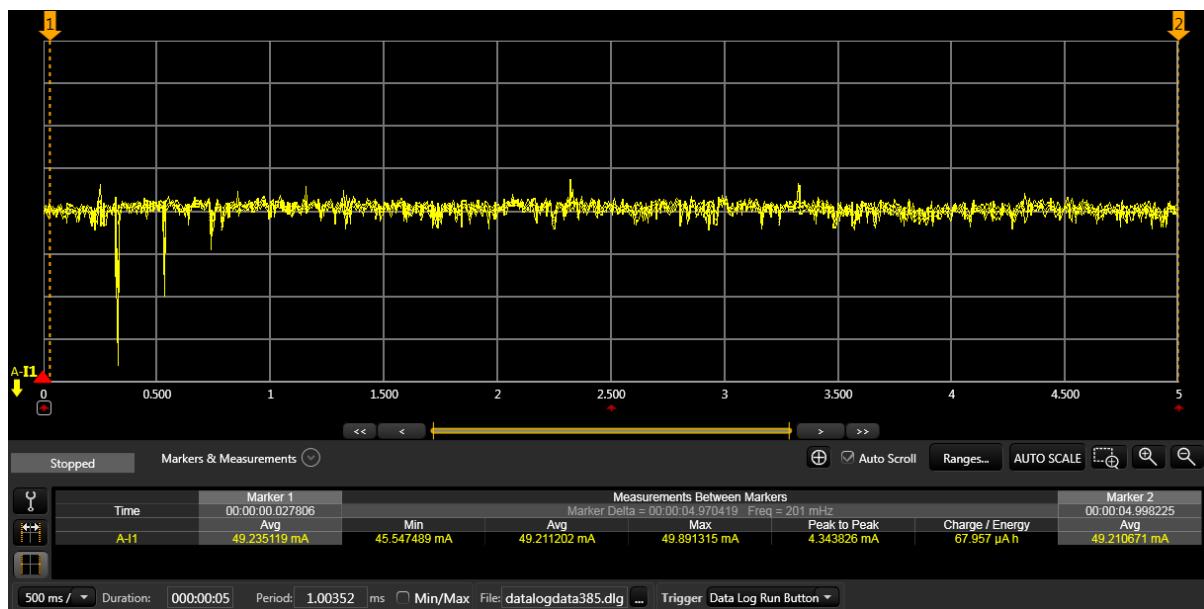
```
<channel_no>
```

2. Open rsi\_wlan\_config.h file and update/modify following macros:

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	
FEAT_SECURITY_OPEN	
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	0
#define RSI_BAND	RSI_BAND_2P4GHZ
#define PLL_MODE	0
#define RF_TYPE	1
#define WIRELESS_MODE	0
#define ENABLE_PPP	0
#define AFE_TYPE	1
#define FEATURE_ENABLES	48

## Executing the Application

1. Switch on power meter and select measurement as current measure.
2. Connect power probes of power meter to WiseMCU device.
3. Connect the Jlink connector from WiseMCU device to windows laptop.
4. Give power to WiseMCU device on power interface.
5. Compile the application in Windows laptop and run the application.
6. PER stats starts in particular channel and make sure no TX packets destined to redpine module.
7. Note down the power measurement as shown below:



### 8.1.6 WLAN Listen LP

#### Example

#### Overview

The Listen LP test application demonstrates how to measure power number of a Redpine module in a listen mode. It helps us to know power consumption of our Redpine module in Listen LP mode.

#### Sequence of Events

This Application explains user how to:

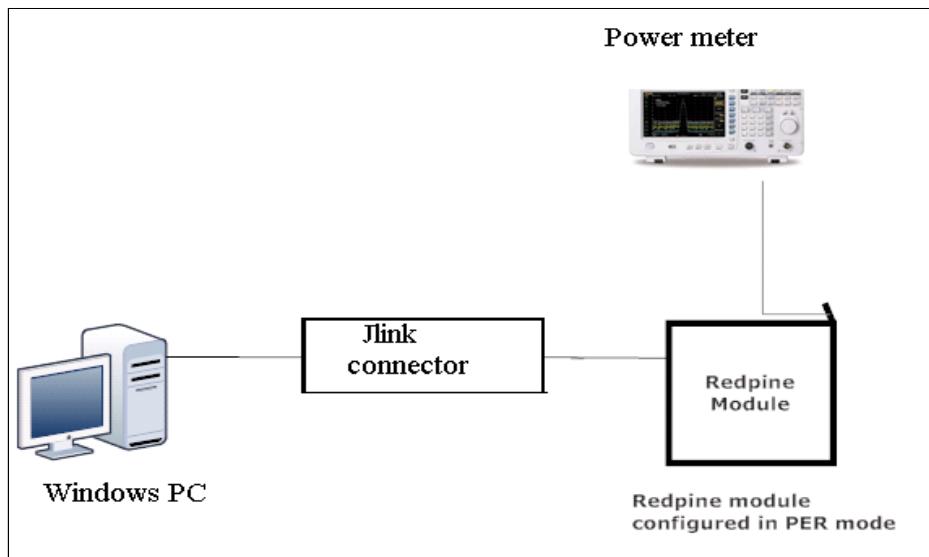
- Initialize.Radio
- Start PER stats in particular channel
- Measure power number

#### Example Setup

The WiSeMCU device offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module
- Jlink connector
- Power Meter



**Figure 1: Setup Diagram**

## Configuration and Execution of the Application

### Configuring the Application

1. Open file **rsi\_pw\_num\_listen.c** and update/modify the following macros:

To configure Burst mode or Continuous mode

```
#define RSI_TX_TEST_MODE RSI_BURST_MODE
```

To configure the channel number in 2.4 GHz/5GHz in which PER mode has to be initialized.

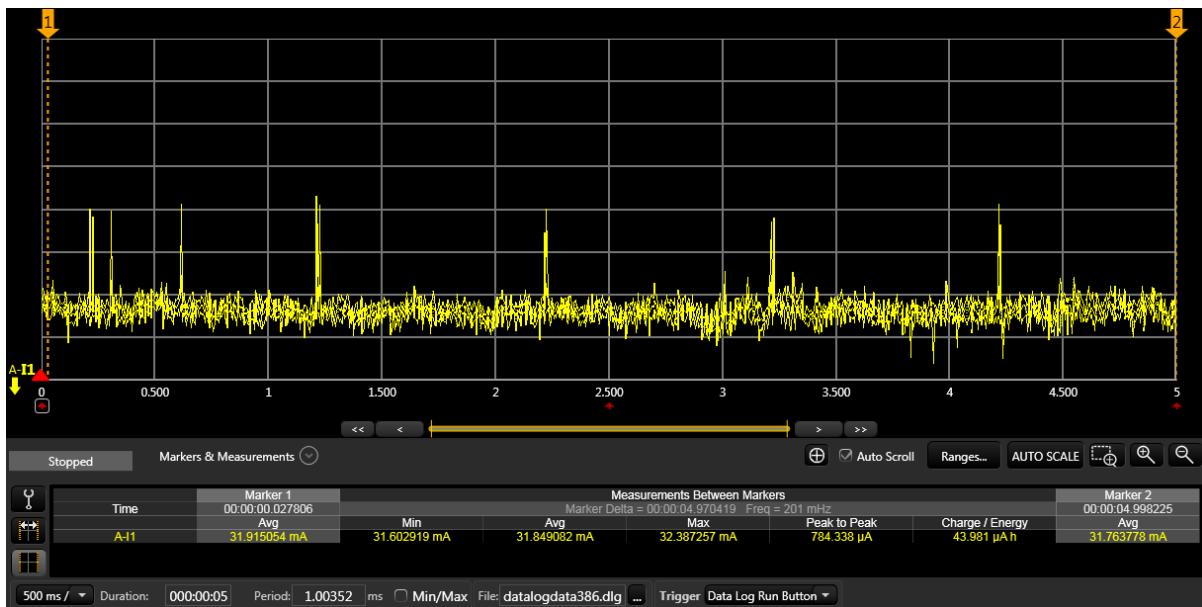
```
#define RSI_TX_TEST_CHANNEL <channel_no>
```

2. Open **rsi\_wlan\_config.h** file and update/modify the following macros:

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	0
TCP_IP_FEAT_DHCPV4_CLIENT	RSI_BAND_2P4GHZ
#define RSI_CUSTOM_FEATURE_BIT_MAP	
#define RSI_BAND	
#define PLL_MODE	0
#define RF_TYPE	1
#define WIRELESS_MODE	12
#define ENABLE_PPP	0
#define AFE_TYPE	1
#define FEATURE_ENABLES	48

## Executing the Application

1. Switch on power meter and select measurement as current measure.
2. Connect power probes of power meter to WiseMCU device.
3. Connect Jlink connector from WiseMCU device to Windows laptop.
4. Give power to WiseMCU device on power interface.
5. Compile the application in Windows laptop. When finished, run the application.
6. PER stats starts in particular channel and make sure no TX packets destined to redpine module.
7. Note down the power measurement as shown below:



### 8.1.7 WLAN RX 1Mbps HP

#### Example

#### Overview

The Rx 1Mbps HP test Application demonstrates how to measure power number of a Redpine device in Rx mode. It helps us to know power consumption of Redpine device in Rx 1Mbps HP mode.

#### Sequence of Events

This Application explains user how to:

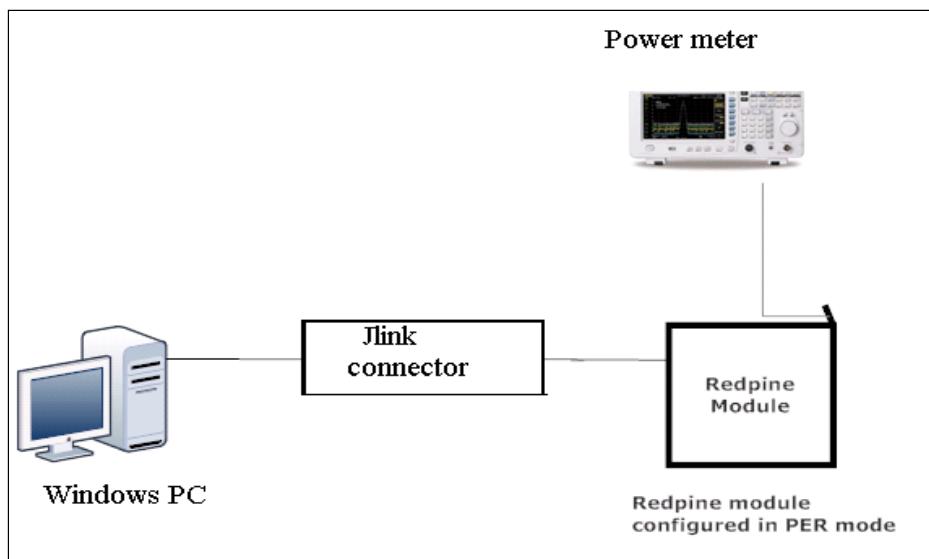
- Initialize.Radio
- Start PER stats in particular channel.
- Measure power number.

#### Example Setup

The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module
- Jlink connector
- Power Meter



**Figure 1: Setup Diagram**

## Configuration and Execution of the Application.

### Configuring the Application

1. Open **rsi\_pw\_num\_rx.c** file and update/modify the following macros:  
To configure Burst mode or Continuous mode

```
#define RSI_TX_TEST_MODE RSI_BURST_MODE
```

To configure the channel number in 2.4 GHz/5GHz in which PER mode has to be initialized.

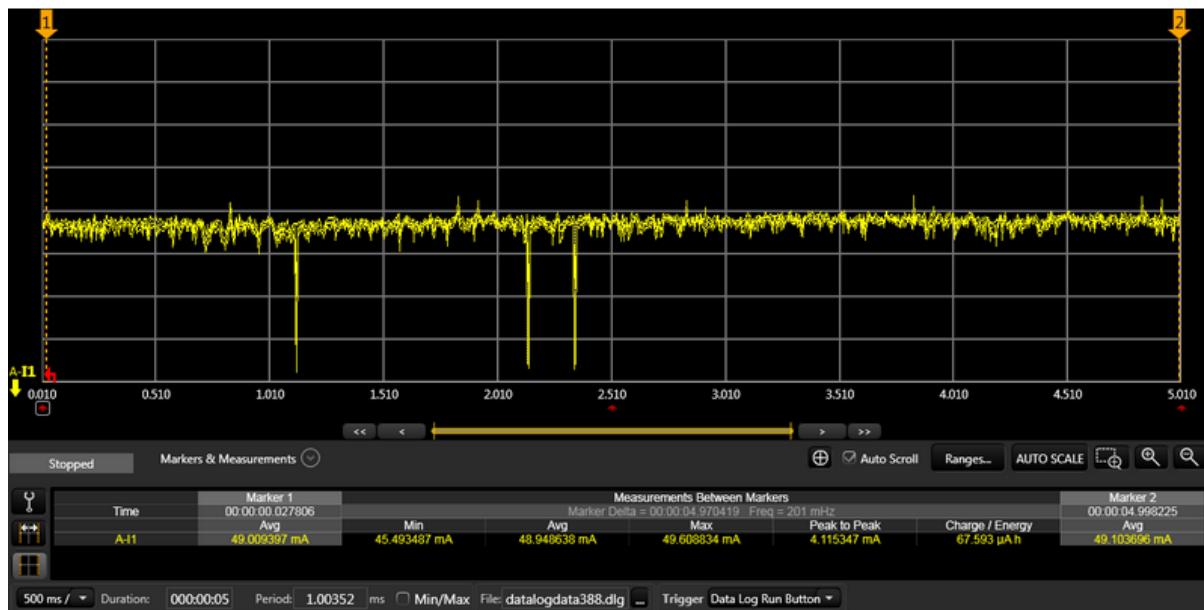
```
#define RSI_TX_TEST_CHANNEL <channel_no>
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	0
#define RSI_CUSTOM_FEATURE_BIT_MAP	RSI_BAND_2P4GHZ
#define RSI_BAND	
#define PLL_MODE	0
#define RF_TYPE	1
#define WIRELESS_MODE	0
#define ENABLE_PPP	0
#define AFE_TYPE	1
#define FEATURE_ENABLES	48

### Executing the Application

1. Switch on power meter and select measurement as current measure.
2. Connect power probes of power meter to WiseMCU device.
3. Connect the Jlink connector from WiseMCU device to the Windows laptop as well.
4. Give power to WiseMCU device on power interface.
5. Compile the application in Windows laptop. When finished, run the application.
6. PER stats starts in particular channel and make sure TX packets destined to redpine module.
7. Note down the power measurement as shown below:



### 8.1.8 WLAN RX 6Mbps HP

#### Example

#### Overview

The Rx 6Mbps HP test Application demonstrates how to measure power number of a Redpine device in Rx mode. It helps us to know power consumption of our Redpine device in Rx 6Mbps HP mode.

#### Sequence of Events

This Application explains user how to:

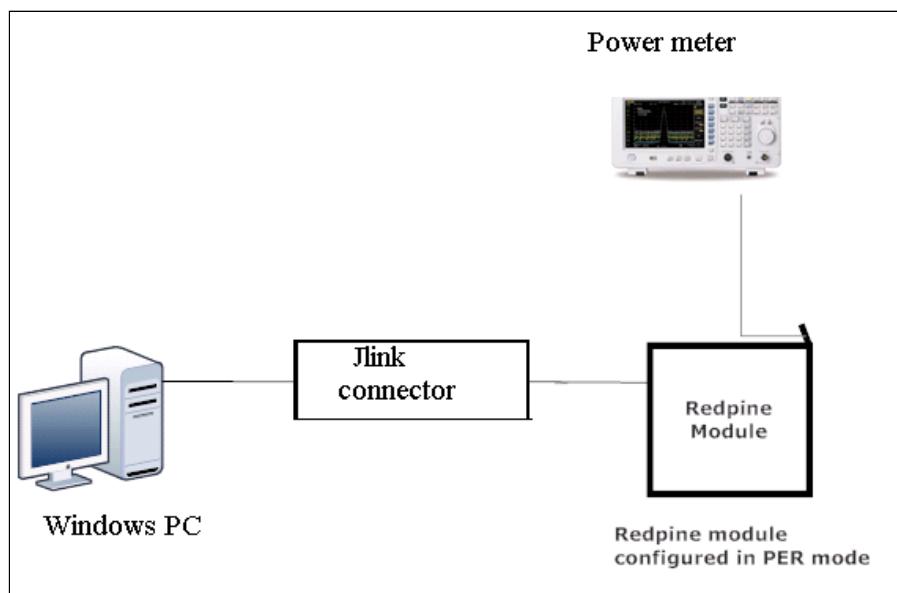
- Initialize.Radio
- Start PER stats in particular channel.
- Measure power number.

#### Example Setup

The WiSeMCU device offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module
- Jlink connector
- Power Meter



**Figure 1: Setup Diagram**

## Configuration and Execution of the Application

### Configuring the Application

1. Open **rsi\_pw\_num\_rx.c** file and update / modify following macros:  
To configure Burst mode or Continuous mode

```
#define RSI_TX_TEST_MODE  
RSI_BURST_MODE
```

To configure the channel number in 2.4 GHz/5GHz in which PER mode has to be initialized

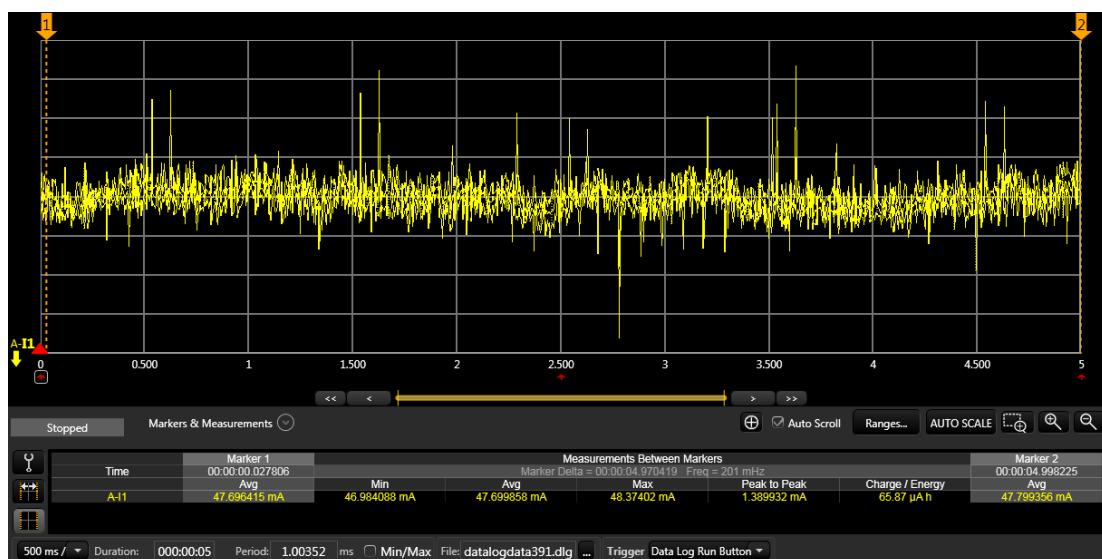
```
#define RSI_TX_TEST_CHANNEL <channel_no>
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros :

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	
FEAT_SECURITY_OPEN	
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	0
#define RSI_BAND	
RSI_BAND_2P4GHZ	
#define PLL_MODE	0
#define RF_TYPE	1
#define WIRELESS_MODE	0
#define ENABLE_PPP	0
#define AFE_TYPE	1
#define FEATURE_ENABLES	48

### Executing the Application

1. Switch on power meter then select measurement as current measure.
2. Connect power probes of power meter to WiseMCU device.
3. Connect Jlink connector from WiseMCU device to windows laptop.
4. Give power to WiseMCU device on power interface.
5. Compile Application in windows laptop then run the application.
6. PER stats starts in particular channel and make sure TX packets destined to redpine module.
7. Note down the power measurement as shown below.



### 8.1.9 WLAN RX 6Mbps LP

#### Example

#### Overview

The Rx 6Mbps LP test Application demonstrates how to measure power number of a Redpine device in Rx mode. It helps us to know power consumption of our Redpine device in Rx 6Mbps LP mode.

#### Sequence of Events

This Application explains user how to:

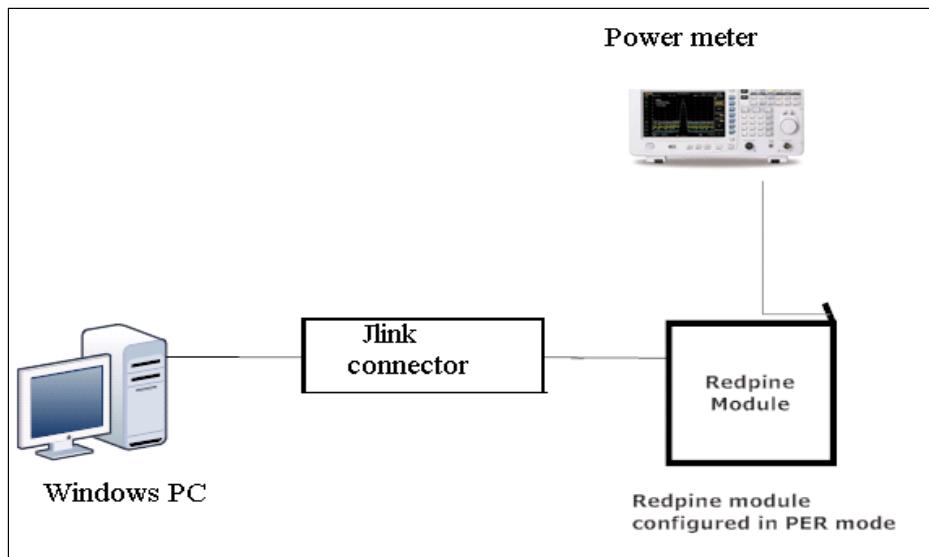
- Initialize.Radio
- Start PER stats in particular channel.
- Measure power number.

#### Example Setup

The WiSeMCU device offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module
- Jlink connector
- Power Meter



**Figure 1: Setup Diagram**

## Configuration and Execution of the Application.

### Configuring the Application

1. Open **rsi\_pw\_num\_rx.c** file and update/modify the following macros:  
To configure the channel number in 2.4 GHz/5GHz in Which PER mode has to be initialized.

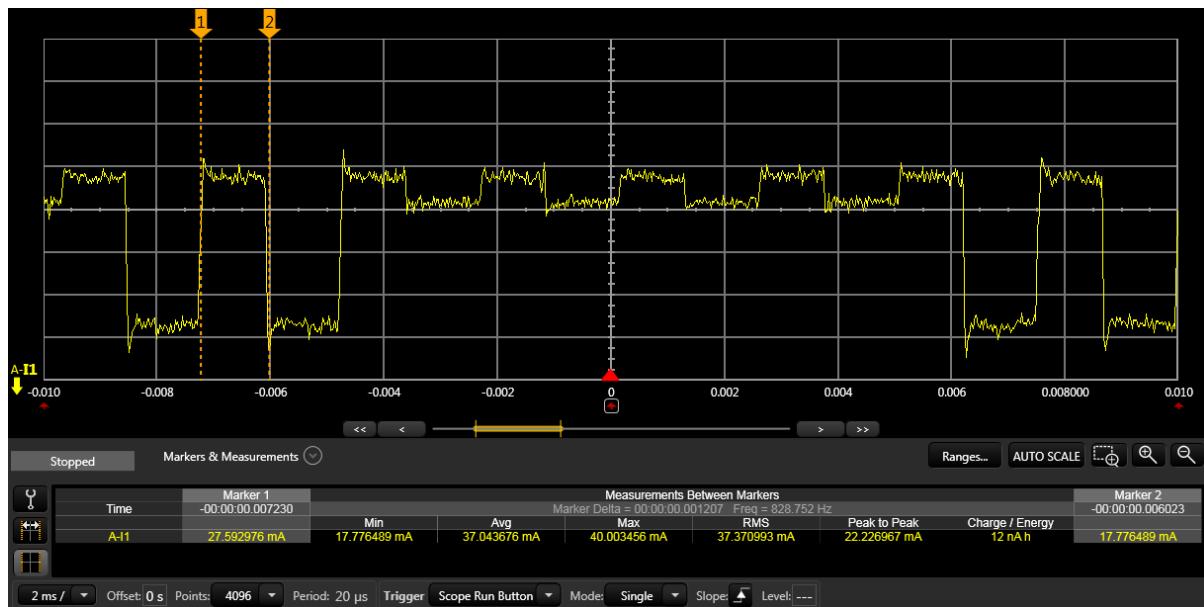
```
#define RSI_TX_TEST_CHANNEL <channel_no>
```

Open **rsi\_wlan\_config.h** file and update/modify the following macros:

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	
FEAT_SECURITY_OPEN	
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	0
#define RSI_BAND	
RSI_BAND_2P4GHZ	
#define PLL_MODE	0
#define RF_TYPE	1
#define WIRELESS_MODE	12
#define ENABLE_PPP	0
#define AFE_TYPE	1
#define FEATURE_ENABLES	0

### Executing the Application

1. Switch on power meter and select measurement as current measure.
2. Connect power probes of power meter to WiseMCU device.
3. Connect the Jlink connector from WiseMCU device to the Windows laptop.
4. Give power to WiseMCU device on power interface.
5. Compile Application in windows laptop. When finished, run the application.
6. PER stats starts in particular channel and make sure TX packets destined to redpine module with rate of 6Mbps.
7. Note down the power measurement as shown below.



**Fig:** Wlan Rx 6Mbps LP chain

### 8.1.10 WLAN RX 72Mbps HP

#### Example

#### Overview

The Rx 72Mbps HP test application demonstrates how to measure power number of a Redpine device in Rx mode. It helps us to know power consumption of our Redpine device in Rx 72Mbps HP mode.

#### Sequence of Events

This Application explains user how to:

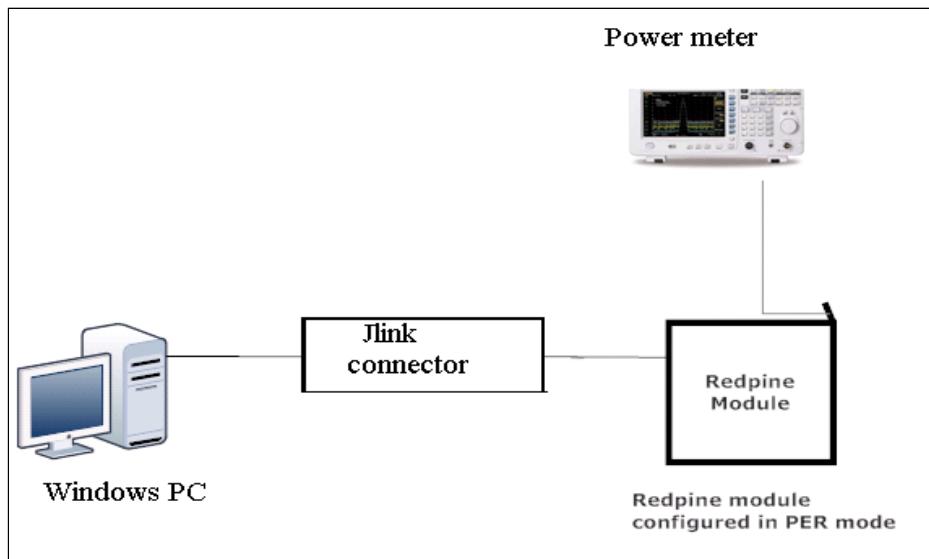
- Initialize Radio
- Start PER stats in particular channel.
- Measure power number.

#### Example Setup

The WiSeMCU device offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module
- Jlink connector
- Power Meter



**Figure 1: Setup Diagram**

Configuration and Execution of the Application.

#### Configuring the Application

1. Open **rsi\_pw\_num\_rx.c** file and update/modify the following macros:  
To configure the channel number in 2.4 GHz/5GHz in Which PER mode has to be initialized.

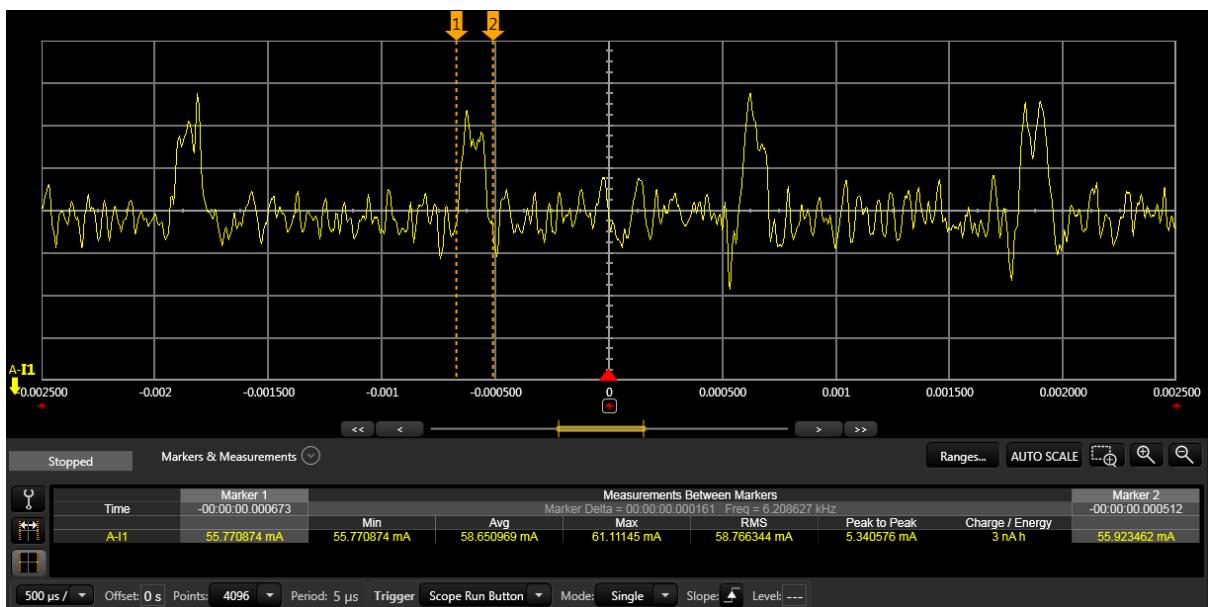
```
#define RSI_TX_TEST_CHANNEL <channel_no>
```

Open **rsi\_wlan\_config.h** file and update/modify the following macros:

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	
FEAT_SECURITY_OPEN	
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	0
#define RSI_BAND	
RSI_BAND_2P4GHZ	
#define PLL_MODE	0
#define RF_TYPE	1
#define WIRELESS_MODE	0
#define ENABLE_PPP	0
#define AFE_TYPE	1
#define FEATURE_ENABLES	0

## Executing the Application

1. Switch on power meter and select measurement as current measure.
2. Connect power probes of power meter to WiseMCU device.
3. Connect the Jlink connector from WiseMCU device to the Windows laptop.
4. Give power to WiseMCU device on power interface.
5. Compile Application in windows laptop. When finished, run the application.
6. PER stats starts in particular channel and make sure TX packets destined to redpine module with rate of 72Mbps.
7. Note down the power measurement as shown below.



**Fig:** Wlan Rx 72Mbps HP chain

### 8.1.11 WLAN Standby

#### Example Overview

##### Overview

The Standby test Application demonstrates how to measure power number of a Redpine device in Connected sleep mode.  
It helps us to know power consumption of Redpine device in Standby mode.

##### Sequence of Events

This Application explains user how to:

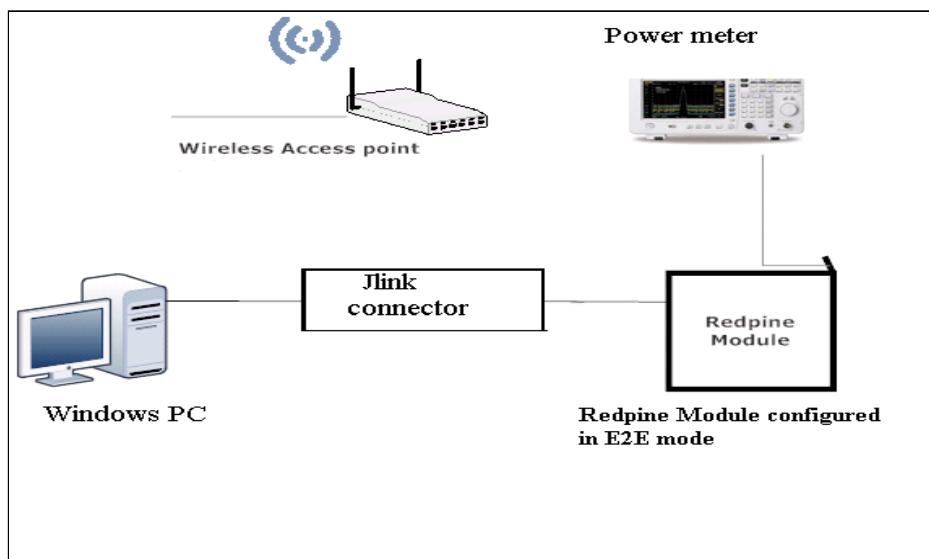
- Connect to an Access point using wireless connectivity.
- Issue power mode command
- Measure power number in connected sleep state.

##### Example Setup

The WiSeMCU device offer integrated wireless connectivity and does not require host interface initialization.

##### WiSeMCU Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module
- Jlink connector
- Access point
- Power Meter



**Figure 1: Setup Diagram**

## Configuration and Execution of the Application

### Configuring the Application

1. Open **rsi\_wlan\_standby.c** file and update/modify the following macros:

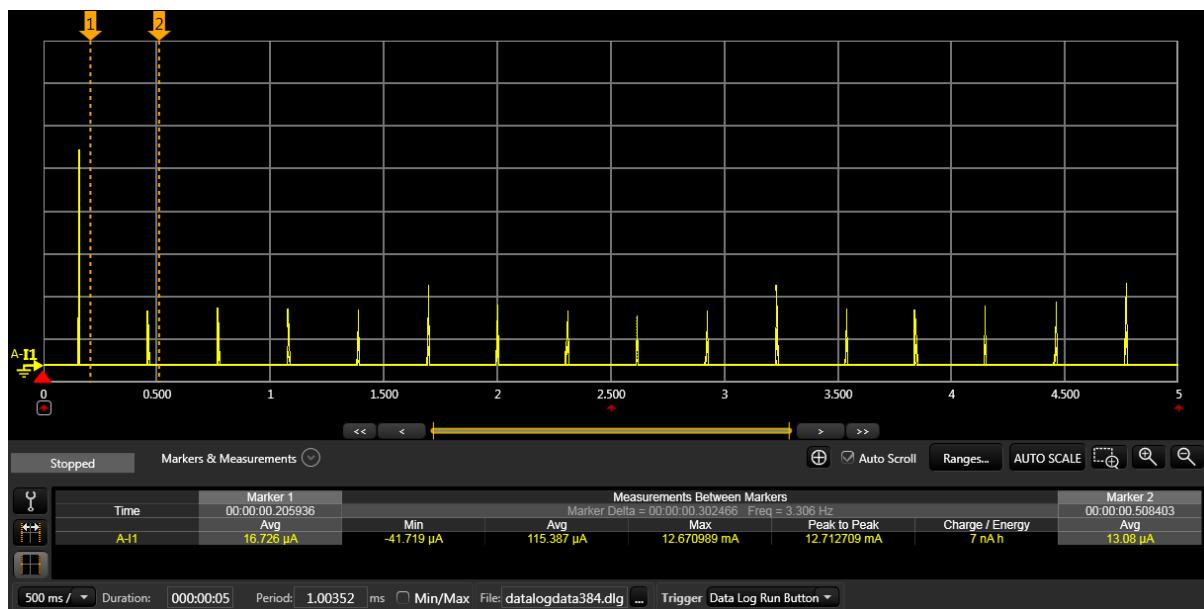
```
#define SSID                                "ap_name"
#define SCAN_CHANNEL                         <channel_no>
//! Security type
#define SECURITY_TYPE                         RSI_OPEN
//! Password
#define PSK                                     NULL
//! DHCP mode 1- Enable 0- Disable
#define DHCP_MODE                             1
//! If DHCP mode is disabled given IP statically
#if !
//! IP address of the module
//! E.g: 0x650AA8C0 == 192.168.10.101
#define DEVICE_IP                            0x650AA8C0
//! IP address of Gateway
//! E.g: 0x010AA8C0 == 192.168.10.1
#define GATEWAY                               0x010AA8C0
//! IP address of netmask
//! E.g: 0x00FFFFFF == 255.255.255.0
#define NETMASK                                0x00FFFFFF
#endif
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros :

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	
FEAT_SECURITY_OPEN	
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	
FEAT_CUSTOM_FEAT_EXTENSION_VALID	
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP	
EXT_FEAT_ENABLE_LOW_POWER_MODE	
#define RSI_BAND	
RSI_BAND_2P4GHZ	
#define RSI_HAND_SHAKE_TYPE	M4_BASED
#define RSI_SELECT_LP_OR_ULP_MODE	
RSI_ULP_WITH_RAM_RET	
#define PLL_MODE	0
#define RF_TYPE	1
#define WIRELESS_MODE	0
#define ENABLE_PPP	0
#define AFE_TYPE	1
#define FEATURE_ENABLES	48

## Executing the Application

1. Switch on power meter and select measurement as current measure.
2. Connect power probes of power meter to WiseMCU device.
3. Connect Jlink connector from WiseMCU device to windows laptop.
4. Give power to WiseMCU device on power interface.
5. Update access point settings with DTIM count as 1/3/10 and beacon interval as 100.
6. Compile Application in windows laptop then run the application.
7. After successful connection with access point then module enter into deep sleep.
8. Note down power measurement as shown below for DTIM count 1,3,10.



**Figure 2: DTIM count 3**

### 8.1.12 WLAN TX 6Mbps 10dBm

#### Example Overview

##### Overview

The TX 6Mbps 10dBm test Application demonstrates how to measure power number of a Redpine device in listen mode.  
 It helps us to know power consumption of Redpine device in TX 6Mbps 10dBm mode.

##### Sequence of Events

This Application explains user how to:

- Initialize.Radio
- Start PER with specific power,data rate,packet length and some common fields.
- Measure power number.

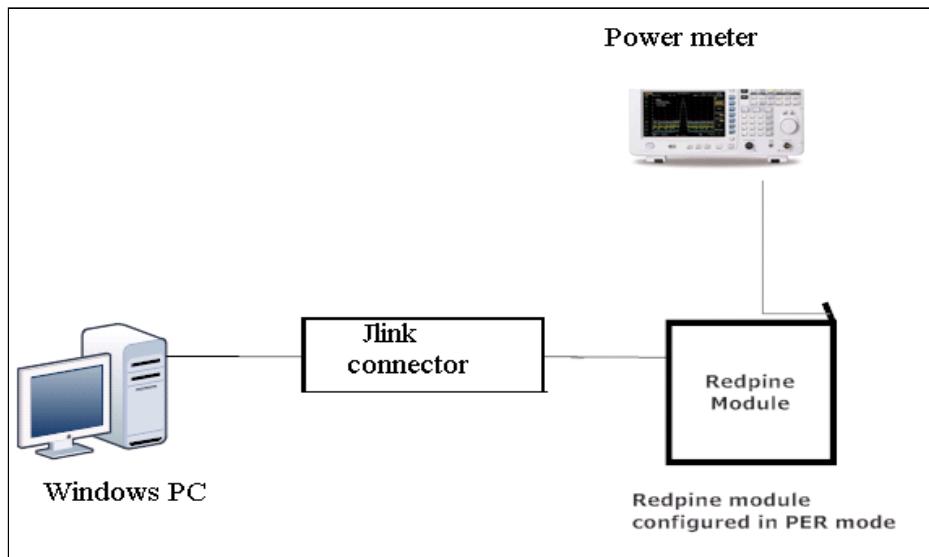
#### Example Setup

The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module

- Jlink connector
- Power Meter



**Figure 1: Setup Diagram**

## Configuration and Execution of the Application

### Configuring the Application

1. Open **rsi\_pw\_num\_tx.c** file and update / modify following macros:  
To set TX power in dbm. The valid values are from 2dbm to 18dbm for WiSeConnectTM module.

```
#define RSI_TX_TEST_POWER
```

10

To set transmit data rate

```
#define RSI_TX_TEST_RATE  
RSI_RATE_6
```

To configure length of the TX packet. Valid values are in the range of 24 to 1500 bytes in the burst mode and range of 24 to 260 bytes in the continuous mode

```
#define RSI_TX_TEST_LENGTH
```

1000

To configure Burst mode or Continuous mode

```
#define RSI_TX_TEST_MODE  
RSI_BURST_MODE
```

To configure the channel number in 2.4 GHz/5GHz in which PER mode has to be initialized.

```
#define RSI_TX_TEST_CHANNEL  
<channel_no>
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP          0
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_CUSTOM_FEATURE_BIT_MAP          0
#define RSI_BAND
RSI_BAND_2P4GHZ

#define PLL_MODE                           0

#define RF_TYPE                            1

#define WIRELESS_MODE                      0

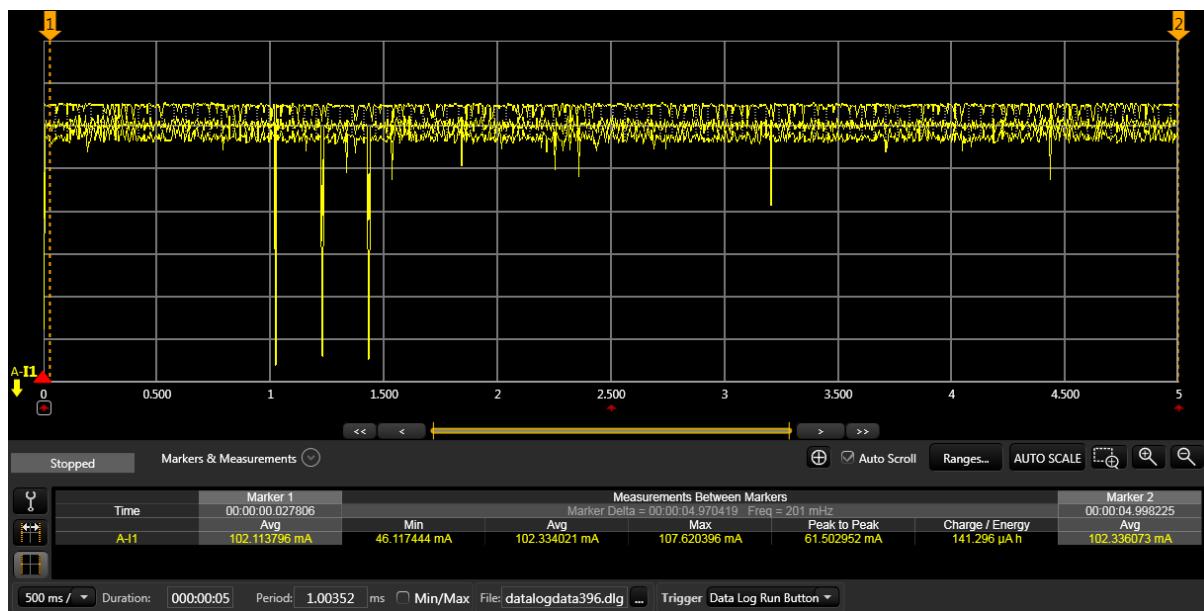
#define ENABLE_PPP                         0
#define AFE_TYPE                           1

#define FEATURE_ENABLES                     48

#define RSI_SET_REGION_SUPPORT
RSI_ENABLE
#define RSI_SET_REGION_FROM_USER_OR_BEACON 1
#define RSI_REGION_CODE                     4
#define RSI_MODULE_TYPE                     1
```

## Executing the Application

1. Switch on power meter and select measurement as current measure.
2. Connect power probes of power meter to WiseMCU device.
3. Connect Jlink connector from WiseMCU device to the Windows laptop.
4. Give power to WiseMCU device on power interface.
5. Compile Application in windows laptop then run the application.
6. Note down the power measurement as shown below:



### 8.1.13 WLAN TX 6Mbps 20dBm

#### Example Overview

##### Overview

The TX 6Mbps 20dBm test Application demonstrates how to measure power number of a Redpine device in TX mode.

It helps us to know power consumption of Redpine device in TX 6Mbps 20dBm mode.

#### Sequence of Events

This Application explains user how to:

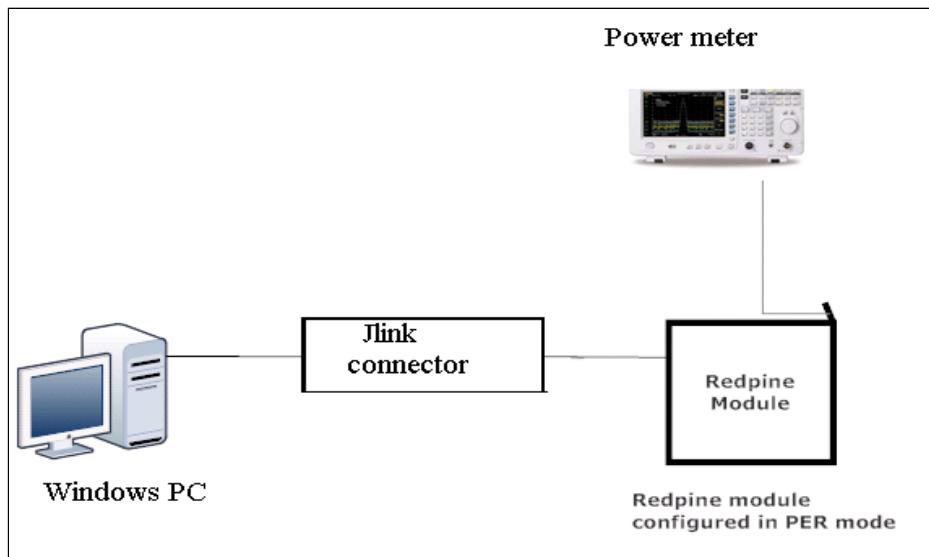
- Initialize.Radio
- Start PER with specific power,data rate,packet length and some common fields.
- Measure power number.

#### Example Setup

The WiSeMCU device offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module
- Jlink connector
- Power Meter



**Figure 1: Setup Diagram**

## Configuration and Execution of the Application

### Configuring the Application

1. Open **rsi\_pw\_num\_tx.c** file and update / modify following macros:  
To set TX power in dbm. The valid values are from 2dbm to 18dbm for WiSeConnectTM module.

```
#define RSI_TX_TEST_POWER
```

20

To set transmit data rate.

```
#define RSI_TX_TEST_RATE  
RSI_RATE_6
```

To configure length of the TX packet. Valid values are in the range of 24 to 1500 bytes in the burst mode and range of 24 to 260 bytes in the continuous mode.

```
#define RSI_TX_TEST_LENGTH
```

1000

To configure Burst mode or Continuous mode

```
#define RSI_TX_TEST_MODE  
RSI_BURST_MODE
```

To configure the channel number in 2.4 GHz/5GHz in which PER mode has to be initialized.

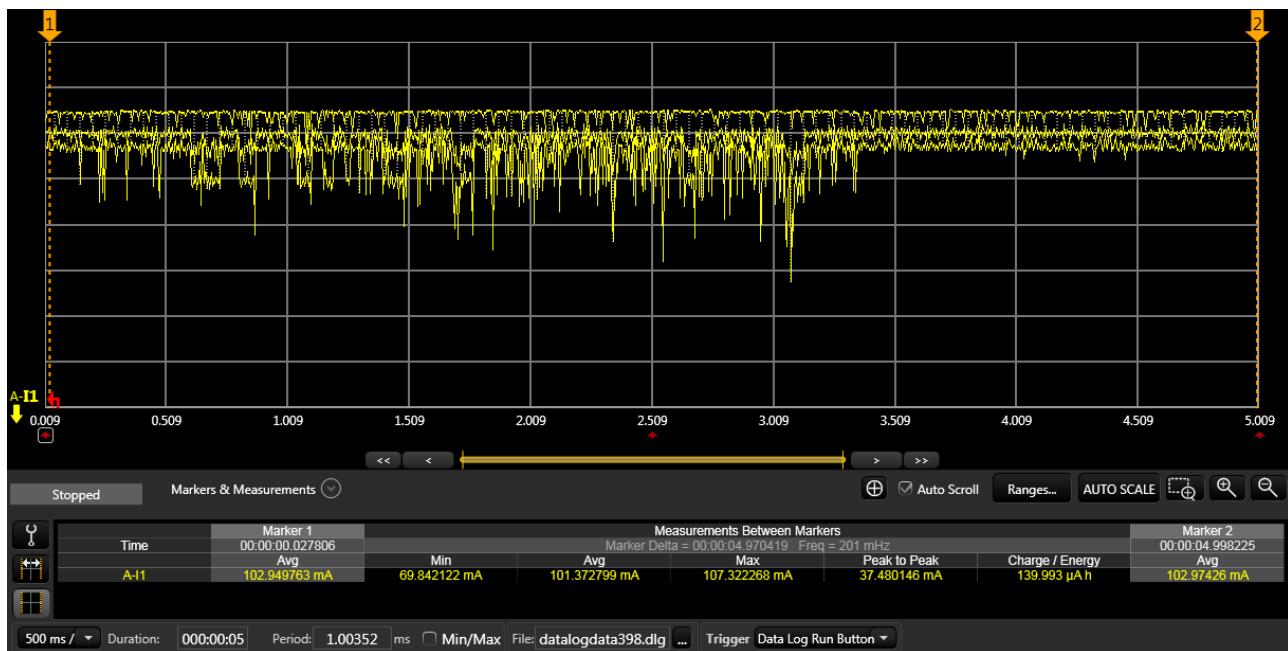
```
#define RSI_TX_TEST_CHANNEL  
<channel_no>
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define RSI_TX_TEST_CHANNEL  
<channel_no>
```

## Executing the Application

1. Switch on power meter and select measurement as current measure.
2. Connect power probes of power meter to WiseMCU device.
3. Connect Jlink connector from WiseMCU device to the Windows laptop.
4. Give power to WiseMCU device on power interface.
5. Compile the application in Windows laptop. When finished, run the application.
6. Note down the power measurement as shown below:



## 8.2 Access Point Start Example

### 8.2.1 Example Overview

#### Overview

The AP start example demonstrates how to configure the Redpine device as a soft Access point and allows stations to connect to it. The example also enables TCP data transmission from the connected Wi-Fi station to Redpine Access Point.

#### Sequence of Events

This example explains user how to:

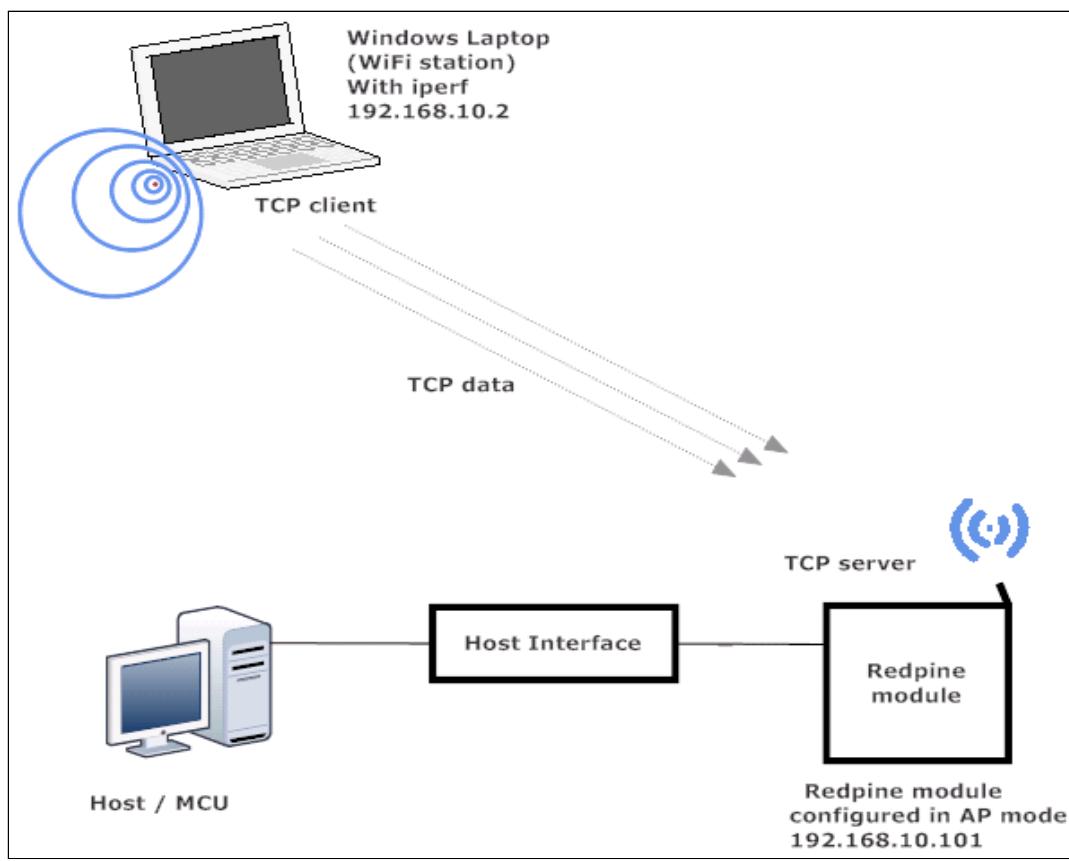
- Create device as 'Soft Access point'
- Open TCP server socket on configured port number on the device
- Connect Wi-Fi Station to device Access point
- Establish TCP connection from connected Wi-Fi Station to TCP server opened on device Access Point
- Send TCP data from Connected station to device Access point
- Read configured number of TCP data packets sent by connected WiFi station.

### 8.2.2 Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- A Mobile device as a Wi-Fi station (This example uses a windows Laptop)
- A TCP client application running on the Wi-Fi station (This example uses iperf for windows )



**Figure 1: Setup diagram**

### 8.2.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ap\_start.c** file and update / modify the following macros.  
**SSID** refers to the name of the Access point to be created.

```
#define SSID           "REDPINE_AP"
```

**CHANNEL\_NO** refers to the channel in which AP would be started

```
#define CHANNEL_NO    11
```

**Note:**

Valid values for CHANNEL\_NO in 2.4GHz band are 1 to 11 and 5GHZ band are 36 to 48 and 149 to 165. In this example default configured band is 2.4GHz.

So, if user wants to use 5GHz band then user has to set RSI\_BAND macro to 5GHz band in (Example folder \$rsi\_wlan\_config.h file).

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities.

Valid configurations are:

- RSI\_OPEN** - For OPEN security mode
- RSI\_WPA** - For WPA security mode
- RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE
```

RSI\_WPA2

**ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports OPEN, TKIP, CCMP encryption methods.

Valid configurations are:

- RSI\_CCMP** - For CCMP encryption
- RSI\_TKIP** - For TKIP encryption
- RSI\_NONE** - For open encryption

```
#define ENCRYPTION_TYPE
```

RSI\_CCMP

**PSK** refers to the secret key if the Access point to be configured in WPA/WPA2 security modes.

```
#define PSK
```

"12345678"

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons in milliseconds. Allowed values are integers from 100 to 1000 which are multiples of 100.

```
#define BEACON_INTERVAL
```

100

**DTIM\_INTERVAL** refers DTIM interval of the Access Point. Allowed values are from 1 to 255.

```
#define DTIM_INTERVAL
```

4

**DEVICE\_PORT** port refers TCP server port number

```
#define DEVICE_PORT
```

5001

**NUMBER\_OF\_PACKETS** refers how many packets to receive from remote TCP client.

```
#define NUMBER_OF_PACKETS
```

1000

**RECV\_BUFFER\_SIZE** refers receive data length

```
#define RECV_BUFFER_SIZE
```

1000

**RSI\_SET\_MAC\_ADDRESS** is enabled to set the user defined MAC address

```
#define RSI_SET_MAC_ADDRESS
```

0

**To configure IP address**

IP address to be configured to the device should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP
```

0X010AA8C0

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY
```

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK
```

0x00FFFFFF

**Note:**

In AP mode, configure same IP address for both **DEVICE\_IP** and **GATEWAY** macros

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

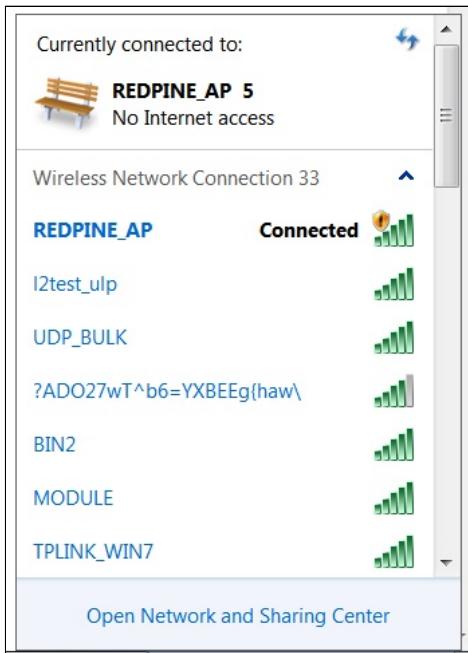
<pre>#define CONCURRENT_MODE</pre>	RSI_DISABLE
<pre>#define RSI_FEATURE_BIT_MAP</pre>	FEAT_SECURITY_PSK
<pre>#define RSI_TCP_IP_BYPASS</pre>	RSI_DISABLE
<pre>#define RSI_TCP_IP_FEATURE_BIT_MAP</pre>	
<pre>TCP_IP_FEAT_DHCPV4_SERVER</pre>	
<pre>#define RSI_CUSTOM_FEATURE_BIT_MAP</pre>	0
<pre>#define RSI_BAND</pre>	RSI_BAND_2P4GHZ

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. After the program gets executed, Device will be created as Access point with configured **SSID** (Ex: "**REDPINE\_AP**") and opens TCP server socket on **DEVICE\_PORT** and waits for TCP connection request from TCP client. Now scan and connect to Device Access Point (Ex: "REDPINE\_AP" is the AP name) from Laptop.



2. After successful connection, open iperf client from Laptop, users can download application from the link below, <https://iperf.fr/iperf-download.php#windows>

3. Connect to TCP server running on AP with port number **DEVICE\_PORT** using the following command:  
**iperf.exe -c <DEVICE\_IP> -p <DEVICE\_PORT> -i 1 -t 100**

A screenshot of an Administrator Command Prompt window on Windows 7. The command entered was "iperf\_demo.exe -c 192.168.10.1 -i 1 -t 20". The output shows the client connecting to 192.168.10.1 port 5001 and performing a 20-second test. The bandwidth results are as follows:

Interval	Transfer	Bandwidth
0.0- 1.0 sec	480 KBytes	3.93 Mbits/sec
1.0- 2.0 sec	320 KBytes	2.62 Mbits/sec
2.0- 3.0 sec	376 KBytes	3.08 Mbits/sec
3.0- 4.0 sec	472 KBytes	3.87 Mbits/sec
4.0- 5.0 sec	408 KBytes	3.34 Mbits/sec
5.0- 6.0 sec	480 KBytes	3.93 Mbits/sec
6.0- 7.0 sec	448 KBytes	3.67 Mbits/sec
7.0- 8.0 sec	392 KBytes	3.21 Mbits/sec
8.0- 9.0 sec	456 KBytes	3.74 Mbits/sec
9.0-10.0 sec	296 KBytes	2.42 Mbits/sec
10.0-11.0 sec	408 KBytes	3.34 Mbits/sec

4. The device will accept connection request and receive data on the TCP server port and Exit after receiving configured NUMBER\_OF\_PACKETS

## 8.3 AP UDP Echo Example

### 8.3.1 Example Overview

#### Overview

This Application demonstrates how to configure UDP socket for Echo service in AP TCP/IP bypass mode.

#### Sequence of Events

This Application explains user how to:

- Create Redpine device as Soft Access point in TCP/IP bypass mode
- Assign static IP to device soft Access point
- Open UDP socket for Echo service
- Connect Wi-Fi Station to device Access point
- Send UDP datagram from Connected station to device Access point
- Send UDP echo by transmitting same received data from Redpine device to connected station

### 8.3.2 Example Setup

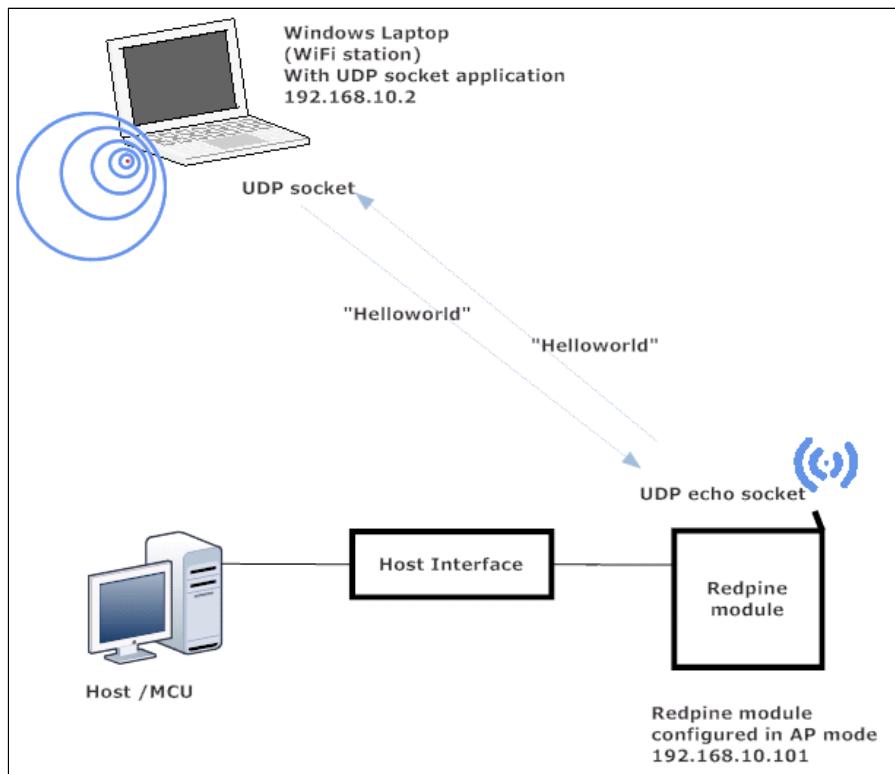
The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine Module
- A Mobile device as a Wi-Fi station (This example uses a windows Laptop)
- A UDP application running on the Wi-Fi station (This example uses SocketTest application for windows)

**Note:**

Download UDP Socket Application from below link: <http://sourceforge.net/projects/sockettest/files/latest/download>



**Figure 1: Setup diagram**

### 8.3.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_ap\_udp\_echo\_tcpipbypass.c** file and update/modify following macros  
**SSID** refers to the name of the Access point to be created.

```
#define SSID  
"REDPINE_AP"
```

**CHANNEL\_NO** refers to the channel in which AP would be started

```
#define CHANNEL_NO
```

11

**Note:**

Valid values for CHANNEL\_NO are 1 to 11 in 2.4GHz band and 36 to 48 & 149 to 165 in 5GHz band. In this example default configured band is 2.4GHz. So, if user wants to use 5GHz band then user has to set RSI\_BAND macro to 5GHz band in **rsi\_wlan\_config.h** file.

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities.

Valid configurations are:

- RSI\_OPEN** - For OPEN security mode
- RSI\_WPA** - For WPA security mode
- RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE
```

RSI\_WPA2

**ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports OPEN, TKIP, CCMP methods.

Valid configurations are:

- RSI\_CCMP** - For CCMP encryption
- RSI\_TKIP** - For TKIP encryption
- RSI\_NONE** - For open encryption

```
#define ENCRYPTION_TYPE
```

RSI\_CCMP

**PSK** refers to the secret key if the Access point to be configured in WPA/WPA2 security modes.

```
#define PSK  
"1234567890"
```

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons in milliseconds. Allowed values are integers from 100 to 1000 which are multiples of 100.

```
#define BEACON_INTERVAL
```

100

**DTIM\_INTERVAL** refers DTIM interval of the Access Point. Allowed values are from 1 to 255.

```
#define DTIM_INTERVAL
```

4

**DEVICE\_PORT** port refers internal UDP server port number

```
#define DEVICE_PORT
```

5001

**REMOTE\_PORT** port refers remote UDP server port number

#define REMOTE\_PORT

5001

**GLOBAL\_BUFF\_LEN** refers Application memory length which is required by the driver

#define GLOBAL\_BUFF\_LEN

10000

**To configure IP address**

IP address to be configured to the device should be in long format and in little endian byte order.  
Example: To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

#define DEVICE\_IP

0X010AA8C0

IP address of the gateway should also be in long format and in little endian byte order.

Example: To configure "192.168.10.1" as Gateway, update the macro GATEWAY as **0x010AA8C0**

#define GATEWAY

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order.

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK

0x00FFFFFF

**Note:**

In AP mode, configure the same IP address for both DEVICE\_IP and GATEWAY macros.

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_PSK
#define RSI_TCP_IP_BYPASS
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_BYPASS
#define RSI_CUSTOM_FEATURE_BIT_MA
#define RSI_BAND
RSI_BAND_2P4GHZ
```

RSI\_ENABLE

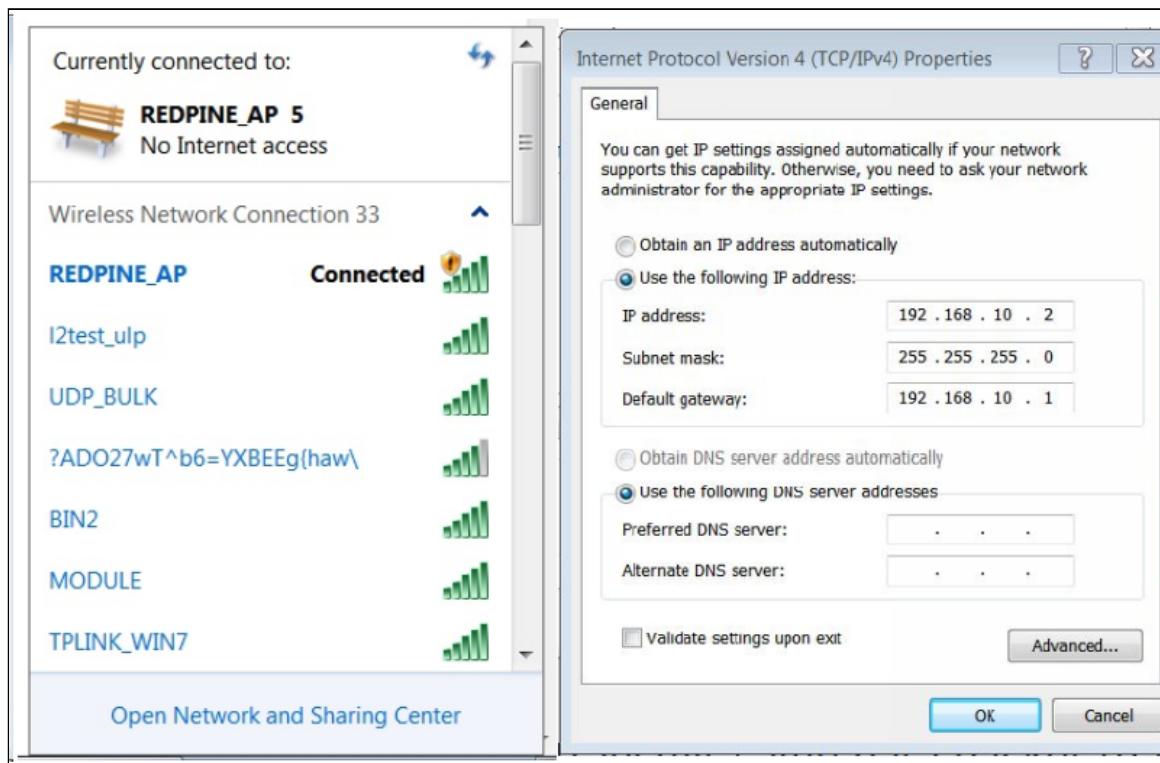
0

**Note:**

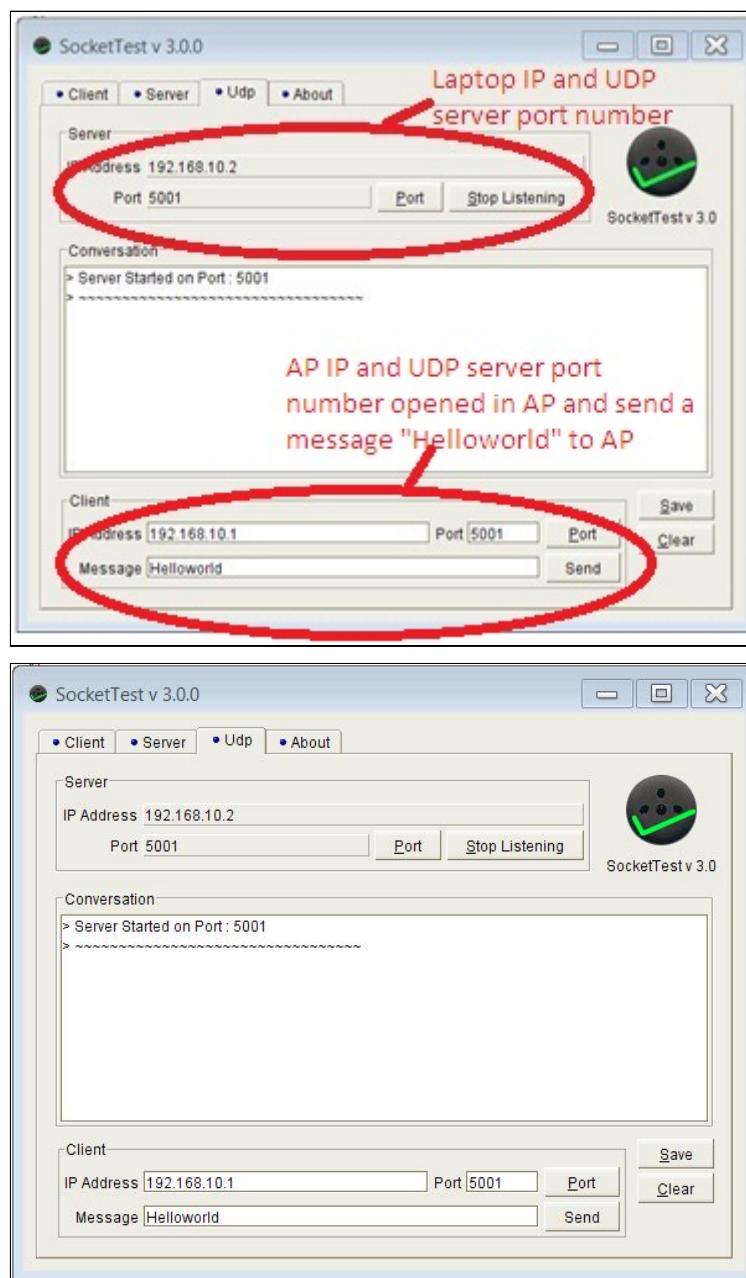
rsi\_wlan\_config.h file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

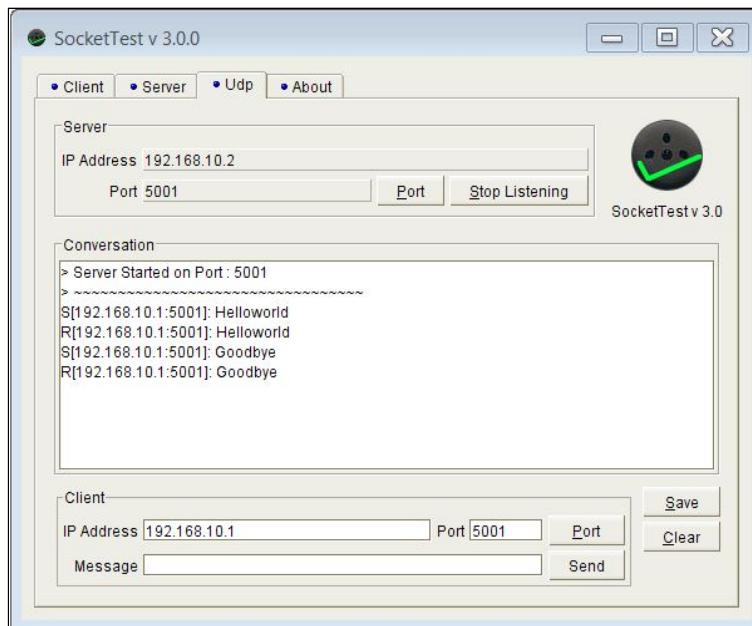
1. After the program gets executed, Redpine device will be configured as Access Point.
2. Connect a Wi-Fi station (Windows Laptop) to device AP (Ex: "REDPINE\_AP" is the AP name) and assign a static IP in the same Network of AP.



3. At remote side device (Wi-Fi Station), open SocketTest application to open UDP server socket and client socket. As per the below image, open UDP server socket on port number **REMOTE\_PORT** to receive data sent by AP and open UDP client socket with port number **DEVICE\_PORT** to send UDP data to AP.



4. Send "Helloworld" and "Goodbye" messages from UDP client to UDP server opened in AP and same messages will send back by AP to the UDP server opened on WiFi Station. The below image depicts the messages sent by Wi-Fi Station and AP.



## 8.4 Concurrent Mode Example

### 8.4.1 Example Overview

#### Overview

This application demonstrates how to configure the device in both Wi-Fi Station mode and Access point mode and how to transfer data on both modes.

In this Application, Redpine device starts as access point and connects with an access point in station mode. After successful creation of access point and successful connection with the same, the application opens TCP socket and transfers TCP data in station mode and device responds for the Ping request sent by connected station with Ping Reply in Access Point mode.

#### Sequence of Events

This Application explains user how to:

- Create Redpine device as Soft Access point
- Connect with 3<sup>rd</sup> party Access Point in Station mode
- Open TCP server socket on configured port number on the device.
- Send TCP data to remote peer in station mode

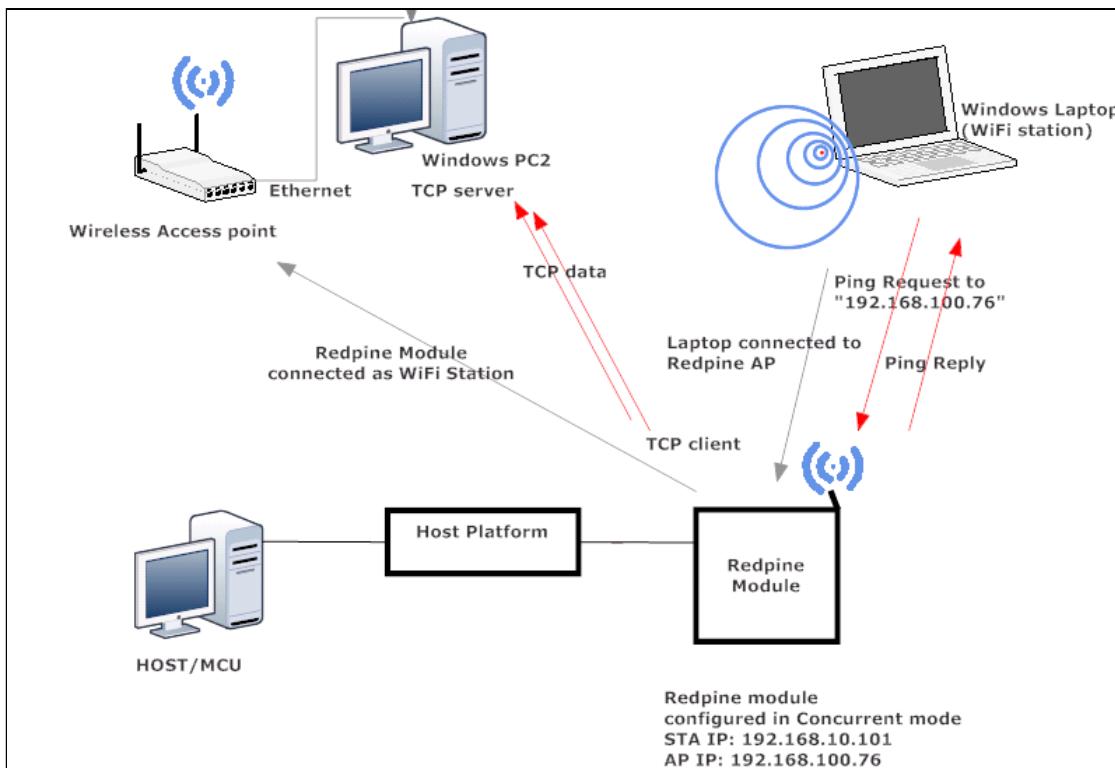
### 8.4.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with Keil or IAR IDE in case of WiSeMCU

- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine Module
- Access point
- Windows PC2
- A TCP server application running on the Wi-Fi station (This example uses iperf for windows )



**Figure 1: Setup Diagram**

#### 8.4.3 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_concurrent\_mode.c** file and update/modify following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID           "REDPINE"
```

**STA\_SECURITY\_TYPE** refers to the type of security. In concurrent mode STA supports Open, WPA, WPA2 securities.

Valid configurations are:

**RSI\_OPEN** - For OPEN security mode

**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define STA_SECURITY_TYPE RSI_OPEN
```

**STA\_PSK** refers to the STA secret key to connect with the secured Access Point.

```
#define STA_PSK NULL
```

**AP\_SSID** refers to the name of the WiSeConnect Access point would be created.

```
#define AP_SSID "REDPINE_AP"
```

**AP\_CHANNEL\_NO** refers to the channel in which AP would be started

```
#define AP_CHANNEL_NO 11
```

**Note1:**

1. Valid values for CHANNEL\_NO are 1 to 11 in 2.4GHz band and 36 to 48 & 149 to 165 in 5GHz. In this example default configured band is 2.4GHz. So, if user wants to use 5GHz band then user has to set RSI\_BAND macro to 5GHz band in *rsi\_wlan\_config.h* file.
2. In concurrent mode, STA and AP should be present in same channel. So, configure the AP\_CHANNEL\_NO to same channel in which Wireless Access point (to which WiSeConnect STA connects) exist.

**AP\_SECURITY\_TYPE** refers to the security type of the WiSeConnect Access Point. Access point supports OPEN, WPA-PSK, WPA2-PSK security modes.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define AP_SECURITY_TYPE RSI_WPA2
```

**AP\_ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports OPEN, TKIP, CCMP methods.

Valid configuration is:

**RSI\_CCMP** - For CCMP encryption  
**RSI\_TKIP** - For TKIP encryption  
**RSI\_NONE** - For open encryption

```
#define AP_ENCRYPTION_TYPE RSI_CCMP
```

**AP\_PSK** refers to the secret key if the Access point to be configured in WPA/WPA2-PSK security modes.

```
#define AP_PSK "1234567890"
```

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons in milliseconds in AP mode. Allowed values are integers from 100 to 1000 which are multiples of 100.

```
#define BEACON_INTERVAL 100
```

**DTIM\_INTERVAL** refers DTIM interval of the Access Point. Allowed values are from 1 to 255.

```
#define DTIM_INTERVAL 4
```

**DEVICE\_PORT** port refers internal TCP client port number

```
#define DEVICE_PORT 5001
```

**REMOTE\_PORT** port refers remote TCP server port number which is opened in Windows PC2.

```
#define REMOTE_PORT 5001
```

**SERVER\_IP\_ADDRESS** refers remote peer (Windows PC2) IP address to connect with TCP server socket. IP address should be in long format and in little endian byte order.  
Example: To configure "192.168.0.100" as remote IP address, update the macro **SERVER\_IP\_ADDRESS** as **0x6400A8C0**.

```
#define SERVER_IP_ADDRESS 0x6400A8C0
```

**NUMBER\_OF\_PACKETS** refers how many packets to send from TCP client to TCP server

```
#define NUMBER_OF_PACKETS 1000
```

#### To configure IP address in STA mode

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC in STA mode

```
#define DHCP_MODE 1
```

**Note:**

If the user wants to configure STA IP address through DHCP then skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.0.10" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP          0X0A00A8C0
```

IP address of the gateway should also be in long format and in little endian byte order.

Example: To configure "192.168.0.1" as Gateway, update the macro **GATEWAY** as **0x0100A8C0**

```
#define GATEWAY           0x0100A8C0
```

IP address of the network mask should also be in long format and in little endian byte order.

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK           0x00FFFFFF
```

**Note:**

1. In this application, we are not providing the facility to configure the Access Point's IP parameters. Default IP address of the Redpine Access point is "192.168.100.76"
2. In concurrent mode, IP networks of Redpine STA and Redpine Access Point should be different. So, Please configure Wireless Access Point IP network (Ex: 192.168.0.1) to other than Redpine Access point IP network.

2. Open **rsi\_wlan\_config.h** file and update/modify following macros :

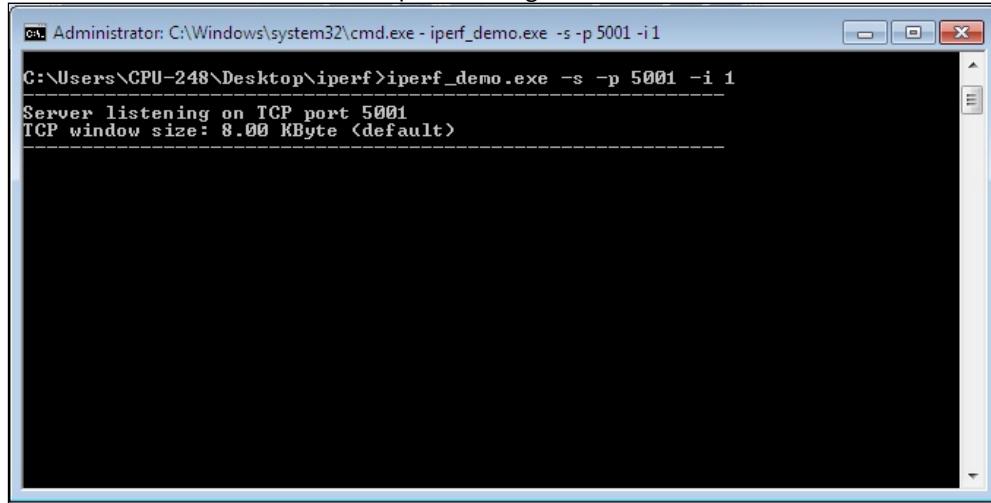
<code>#define CONCURRENT_MODE</code>	<code>RSI_ENABLE</code>
<code>#define RSI_FEATURE_BIT_MAP</code>	<code>FEAT_SECURITY_PSK</code>
<code>#define RSI_TCP_IP_BYPASS</code>	<code>RSI_DISABLE</code>
<code>#define RSI_TCP_IP_FEATURE_BIT_MAP</code>	<code>(TCP_IP_FEAT_DHCPV4_SERVER  </code>
<code>TCP_IP_FEAT_DHCPV4_CLIENT)</code>	<code>0</code>
<code>#define RSI_CUSTOM_FEATURE_BIT_MAP</code>	<code>RSI_BAND_2P4GHZ</code>
<code>#define RSI_BAND</code>	

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

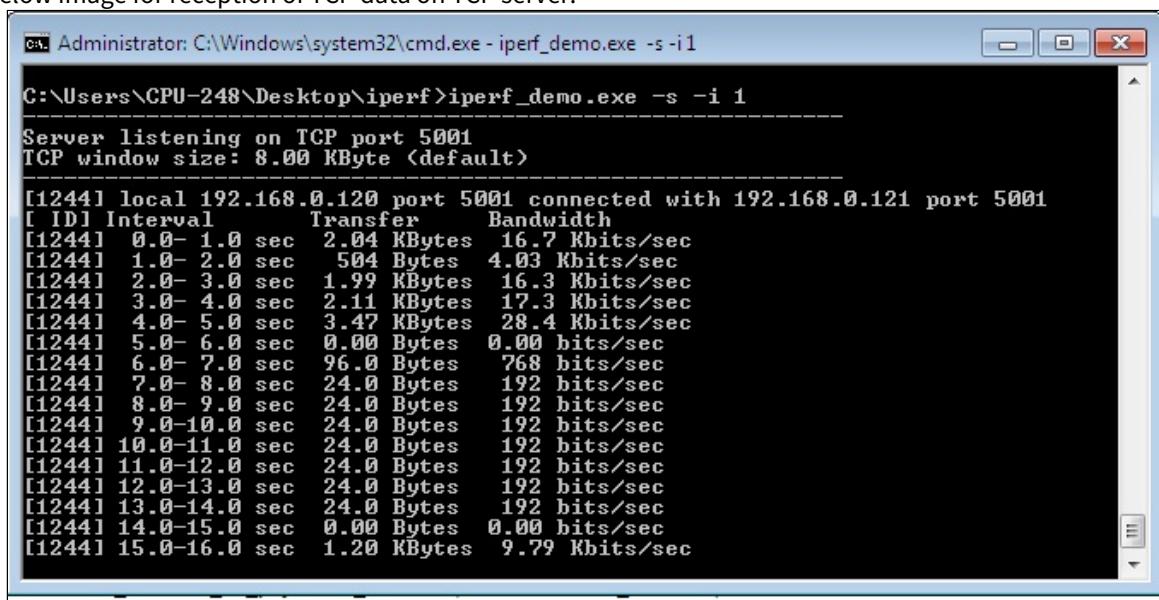
## Executing the Application

1. Configure the access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. Open TCP server application using iperf application in Windows PC2 which is connected to the Access point through LAN.



```
Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -p 5001 -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -p 5001 -i 1
Server listening on TCP port 5001
TCP window size: 8.00 KByte <default>
```

3. After the program gets executed, Redpine device will be connected to the access point and starts as an access point having the configuration same as that in the application.
4. After successful connection in STA mode, STA connects to TCP server socket opened on Windows PC2 using TCP client socket and sends configured **NUMBER\_OF\_PACKETS** to remote TCP server. Please refer the below image for reception of TCP data on TCP server.

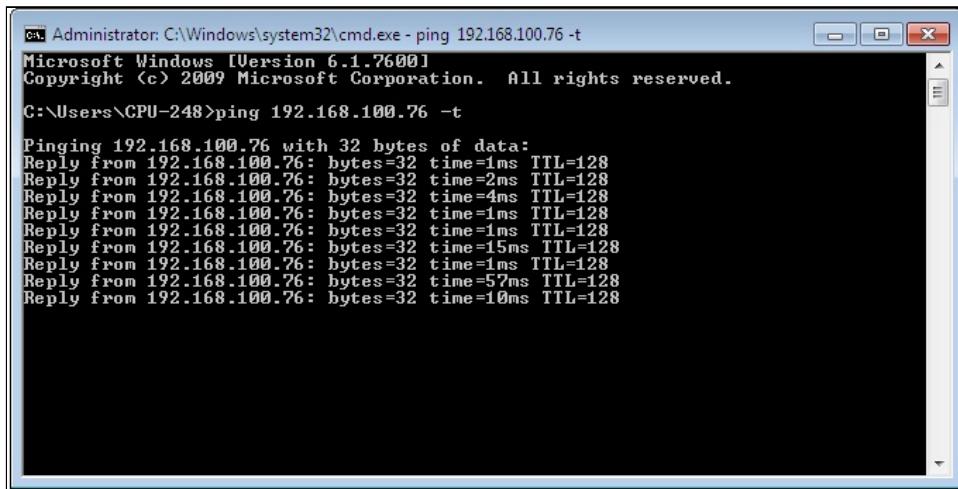


```
Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -i 1
Server listening on TCP port 5001
TCP window size: 8.00 KByte <default>
[12441] local 192.168.0.120 port 5001 connected with 192.168.0.121 port 5001
[12441] Interval Transfer Bandwidth
[12441] 0.0- 1.0 sec 2.04 KBytes 16.7 Kbits/sec
[12441] 1.0- 2.0 sec 504 Bytes 4.03 Kbits/sec
[12441] 2.0- 3.0 sec 1.99 KBytes 16.3 Kbits/sec
[12441] 3.0- 4.0 sec 2.11 KBytes 17.3 Kbits/sec
[12441] 4.0- 5.0 sec 3.47 KBytes 28.4 Kbits/sec
[12441] 5.0- 6.0 sec 0.00 Bytes 0.00 bits/sec
[12441] 6.0- 7.0 sec 96.0 Bytes 768 bits/sec
[12441] 7.0- 8.0 sec 24.0 Bytes 192 bits/sec
[12441] 8.0- 9.0 sec 24.0 Bytes 192 bits/sec
[12441] 9.0-10.0 sec 24.0 Bytes 192 bits/sec
[12441] 10.0-11.0 sec 24.0 Bytes 192 bits/sec
[12441] 11.0-12.0 sec 24.0 Bytes 192 bits/sec
[12441] 12.0-13.0 sec 24.0 Bytes 192 bits/sec
[12441] 13.0-14.0 sec 24.0 Bytes 192 bits/sec
[12441] 14.0-15.0 sec 0.00 Bytes 0.00 bits/sec
[12441] 15.0-16.0 sec 1.20 KBytes 9.79 Kbits/sec
```

5. Connect Wi-Fi STA (from laptop) to Redpine Access Point (Ex: AP name configured as "**REDPINE\_AP**").



6. After successful connection, initiate ping from Wi-Fi STA (Laptop) to device access point IP address "192.168.100.76".  
**ping 192.168.100.76 -t**
7. The device access point gives Ping reply for the received Ping Request. The below image depicts the ping success.



```
Administrator: C:\Windows\system32\cmd.exe - ping 192.168.100.76 -t
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\CPU-248>ping 192.168.100.76 -t

Pinging 192.168.100.76 with 32 bytes of data:
Reply from 192.168.100.76: bytes=32 time=1ms TTL=128
Reply from 192.168.100.76: bytes=32 time=2ms TTL=128
Reply from 192.168.100.76: bytes=32 time=4ms TTL=128
Reply from 192.168.100.76: bytes=32 time=1ms TTL=128
Reply from 192.168.100.76: bytes=32 time=1ms TTL=128
Reply from 192.168.100.76: bytes=32 time=15ms TTL=128
Reply from 192.168.100.76: bytes=32 time=1ms TTL=128
Reply from 192.168.100.76: bytes=32 time=57ms TTL=128
Reply from 192.168.100.76: bytes=32 time=10ms TTL=128
```

## 8.5 Connection using Asynchronous APIs Example

### 8.5.1 Asynchronous APIs Overview

Asynchronous APIs instruct the module and return the status. The actual response by the module is indicated to the application in the registered call backs. So the driver need not indefinitely wait for the response from the module. It can schedule its tasks.

---

## 8.5.2 Example Overview

### Overview

The application demonstrates how the Redpine device will be connected to an access point using Asynchronous APIs. After successful connection, Redpine device opens TCP socket with remote peer and sends TCP data on opened socket.

### Sequence of Events

This Application explains user how to:

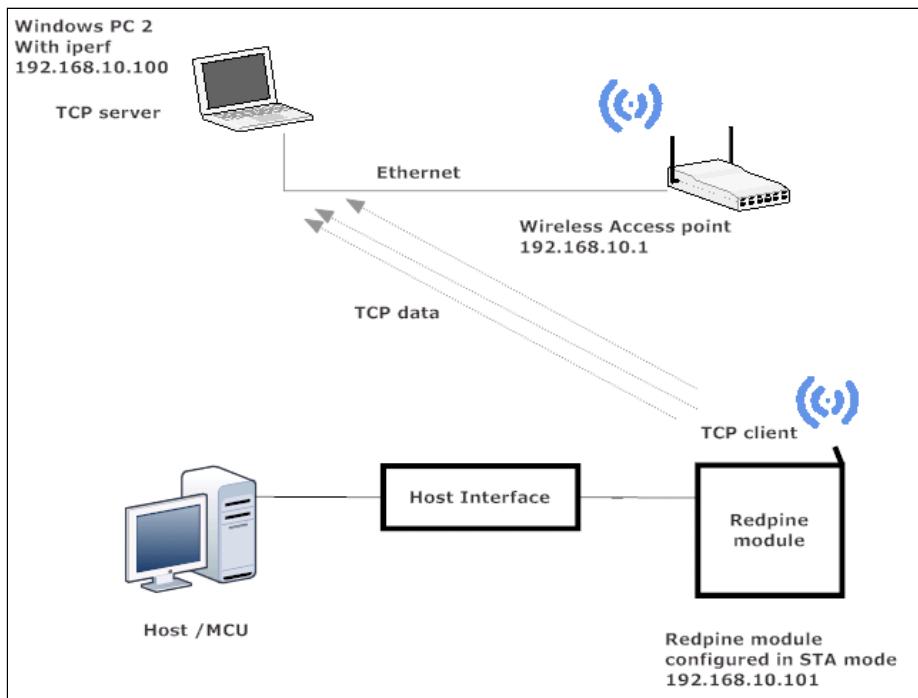
- Scan for Access Point using Asynchronous API
- Handle scan responses
- Connect to Access Point using Asynchronous API
- Handle Join response
- Open TCP client socket on configured port number on the device.
- Send TCP data from device to remote peer.

## 8.5.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with Keil or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Access Point
- Windows PC2
- A TCP Server application running on the Windows PC2 (This example uses iperf for windows )



**Figure 1: Setup Diagram**

#### 8.5.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_connection\_using\_async\_apis\_app.c** file and update/modify following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID "REDPINE_AP"
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK NULL
```

**DEVICE\_PORT** port refers module TCP client port number

```
#define DEVICE_PORT 5001
```

**SERVER\_PORT** port refers remote TCP server port number which is opened in Windows PC2.

```
#define SERVER_PORT 5001
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with TCP server socket.  
IP address should be in long format and in little endian byte order.

Example: To configure "192.168.10.100" as remote IP address, update the macro **SERVER\_IP\_ADDRESS** as **0x640AA8C0**.

```
#define SERVER_IP_ADDRESS 0x640AA8C0
```

**NUMBER\_OF\_PACKETS** refers how many packets to receive from TCP client

```
#define NUMBER_OF_PACKETS 1000
```

To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE 1
```

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP 0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

#define GATEWAY	0x010AA8C0
-----------------	------------

IP address of the network mask should also be in long format and in little endian byte order  
Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK	0x00FFFFFF
-----------------	------------

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_CUSTOM_FEATURE_BIT_MAP	0
#define RSI_BAND	RSI_BAND_2P4GHZ

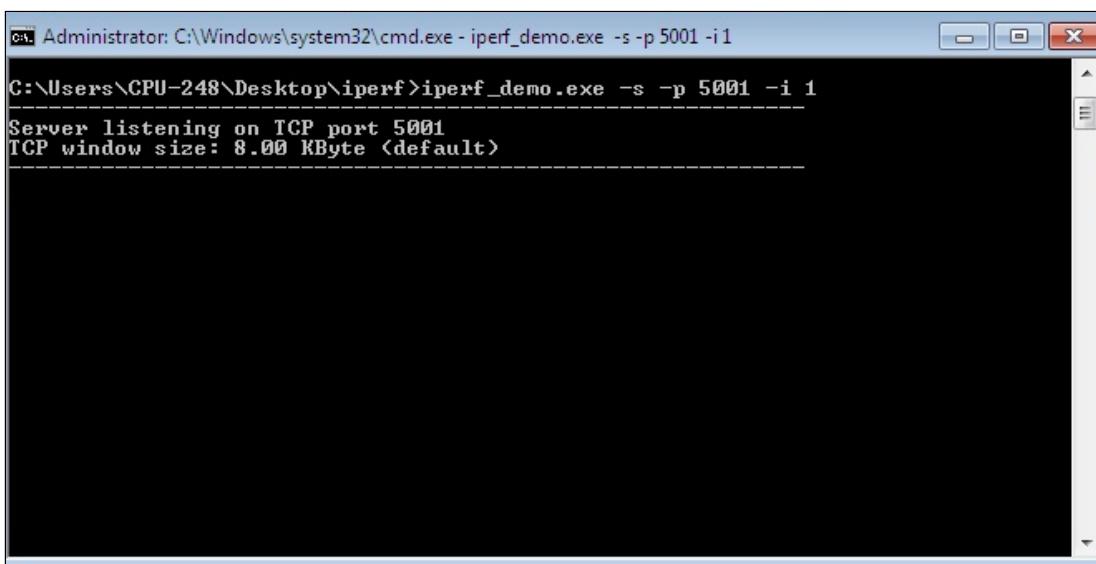
**Note:**

The **rsi\_wlan\_config.h** file is already set with the desired configuration in respective example folders user need not change for each example.

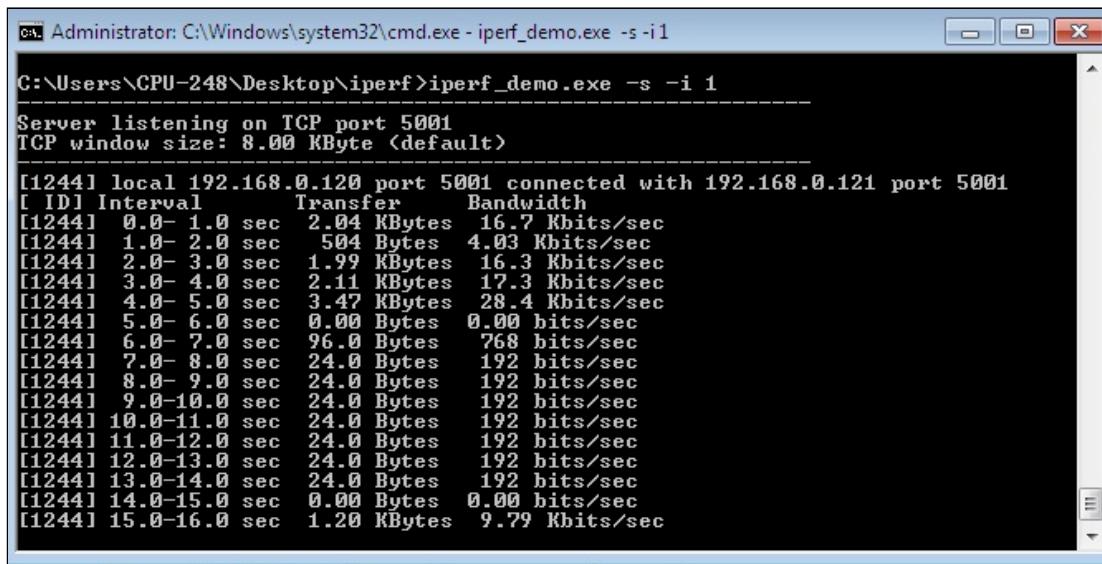
## Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. Open TCP server application using iperf application in Windows PC2 which is connected to Access point through LAN.

**iperf\_demo.exe -s -p <SERVER\_PORT> -i 1**



3. After the program gets executed, Redpine device will scan and connect to the access point and get IP.
4. After successful connection, device STA connects to TCP server socket opened on Windows PC2 using TCP client socket and sends configured **NUMBER\_OF\_PACKETS** to remote TCP server. Refer the below image for reception of TCP data on TCP server,



```
C:\ Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -i 1
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
[1244] local 192.168.0.120 port 5001 connected with 192.168.0.121 port 5001
[ ID] Interval Transfer Bandwidth
[1244] 0.0- 1.0 sec 2.04 KBytes 16.7 Kbits/sec
[1244] 1.0- 2.0 sec 504 Bytes 4.03 Kbits/sec
[1244] 2.0- 3.0 sec 1.99 KBytes 16.3 Kbits/sec
[1244] 3.0- 4.0 sec 2.11 KBytes 17.3 Kbits/sec
[1244] 4.0- 5.0 sec 3.47 KBytes 28.4 Kbits/sec
[1244] 5.0- 6.0 sec 0.00 Bytes 0.00 bits/sec
[1244] 6.0- 7.0 sec 96.0 Bytes 768 bits/sec
[1244] 7.0- 8.0 sec 24.0 Bytes 192 bits/sec
[1244] 8.0- 9.0 sec 24.0 Bytes 192 bits/sec
[1244] 9.0-10.0 sec 24.0 Bytes 192 bits/sec
[1244] 10.0-11.0 sec 24.0 Bytes 192 bits/sec
[1244] 11.0-12.0 sec 24.0 Bytes 192 bits/sec
[1244] 12.0-13.0 sec 24.0 Bytes 192 bits/sec
[1244] 13.0-14.0 sec 24.0 Bytes 192 bits/sec
[1244] 14.0-15.0 sec 0.00 Bytes 0.00 bits/sec
[1244] 15.0-16.0 sec 1.20 KBytes 9.79 Kbits/sec
```

## 8.6 DHCP Option-81 (FQDN) Example

### 8.6.1 DHCP Option-81 Overview

DHCP Option-81 is used to update the devices DNS hostname in the configured DNS server.

Option-81 can be used to exchange information about a DHCPv4 client's fully qualified domain name and about responsibility for updating the DNS RR related to the client's address assignment. DNS maintains (among other things) the information about the mapping between hosts' Fully Qualified Domain Names (FQDNs) [11] and IP addresses assigned to the hosts. The information is maintained in two types of Resource Records (RRs): A and PTR. The DNS update specification ([4]) describes a mechanism that enables DNS information to be updated over a network. The Dynamic Host Configuration Protocol for IPv4 (DHCPv4 or just DHCP in this document) [5] provides a mechanism by which a host (a DHCP client) can acquire certain configuration information, along with its address. This document specifies a DHCP option, the Client FQDN option, which can be used by DHCP clients and servers to exchange information about the client's fully qualified domain name for an address and who has the responsibility for updating the DNS with the associated A and PTR RRs.

### Overview

The application demonstrates how to configure Redpine device in client mode to send DNS update requests to the configured DNS server.

### Sequence of Events

This Application explains user how to:

- Connect to Access Point in station mode

- Send DNS update request to the configured DNS server

### 8.6.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with Keil or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine Module
- Wi-Fi Access point

### 8.6.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **dhcp\_dns\_fqdn/dhcp\_dns\_fqdn.c** file and update/modify following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is

**RSI\_OPEN** - For OPEN security mode

**RSI\_WPA** - For WPA security mode

**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE    RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK             "<psk>"
```

**To configure IP address**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

#define DHCP\_MODE

1

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

#define DEVICE\_IP

0X0A0AA8C0

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

#define GATEWAY

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK

0x00FFFFFF

Configure following macros to send DNS update to the configured DNS server IP address of the DNS server.

Example: To configure "192.168.10.1" as **RSI\_DNS\_SERVER\_IP**, update the macro **RSI\_DNS\_SERVER\_IP** as **0x0A0AA8C0**.

#define RSI\_DNS\_SERVER\_IP

0x6500A8C0

**RSI\_DNS\_TTL** refers the time to live of the hostname

#define RSI\_DNS\_TTL

53

**RSI\_ZONE\_NAME** refers zone name of the configured DNS server

#define RSI\_DNS\_ZONE\_NAME

"rps"

**RSI\_DNS\_HOST\_NAME** refers host name of the configured DNS server.

```
#define RSI_DNS_HOST_NAME "redpine"
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP TCP_IP_FEAT_DHCPV4_CLIENT|TCP_IP_FEAT_DNS_CLIENT|
TCP_IP_FEAT_ICMP)
#define RSI_CUSTOM_FEATURE_BIT_MAP 0
#define RSI_BAND RSI_BAND_2P4GHZ
```

**Note:**

`rsi_wlan_config.h` file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. After the program gets executed, Redpine module configured as client and connects to AP and gets IP.
3. After successful connection with access point, Redpine device starts sending DNS update request to the given `RSI_DNS_SERVER_IP` with configured **RSI\_DNS\_ZONE\_NAME** and **RSI\_DNS\_HOST\_NAME** to update the hostname of the device.
4. In `rsi_dhcp_dns_fqdn.c` file, **rsi\_dns\_update** API returns success status, which means that the DNS update packet is successfully sent in to the medium. When actual response comes from the DNS server, it is known from the status parameter of the callback function (**rsi\_dns\_response\_handler**) registered in the `rsi_dns_update` API.

## 8.7 DHCP User Class Example

### 8.7.1 Protocol Overview

This Option is used by a DHCP client to optionally identify the type or category of user or application it represents. A dhcp server uses the user class option to choose the address pool it allocates an address from and to select any other configuration options.

### 8.7.2 Example Overview

#### Overview

This application demonstrates how DHCP USER CLASS option can be used. In this application, the device connects to the Access Point.

## Sequence of Events

This Application explains user how to:

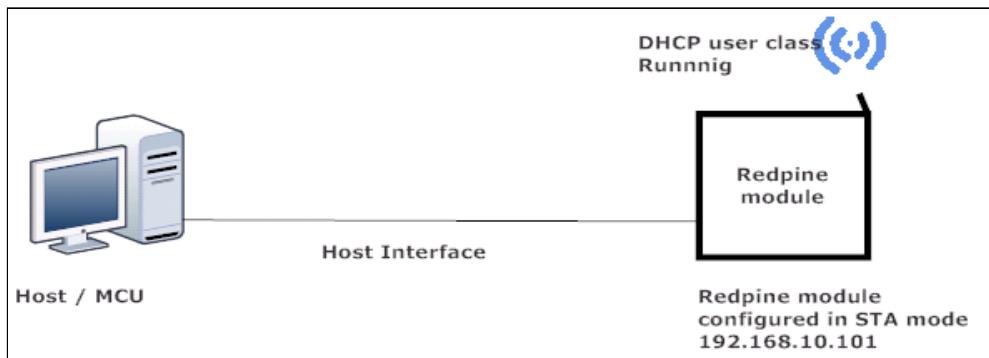
- Connect to Access Point
- Send DHCP User class option

### 8.7.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with Keil or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- LINUX PC
- Redpine Module
- Wi-Fi Access point



**Figure 1: Setup Diagram**

### 8.7.4 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_dhcp\_opt\_77.c** file and update/modify the following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels.

```
#define CHANNEL_NO          0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

- RSI\_OPEN** - For OPEN security mode
- RSI\_WPA** - For WPA security mode
- RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE        RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK                  "<psk>"
```

#### To configure DHCP user class Parameters

//MAX count is 4

```
#define RSI_DHCP_USER_CLASS_COUNT    4
```

//MAX LENGTH 33 including NULL

#define RSI_DHCP_USER_CLASS_DATA_1	"Redpine-class1"
#define RSI_DHCP_USER_CLASS_DATA_2	"Redpine-class2"
#define RSI_DHCP_USER_CLASS_DATA_3	"Redpine-class3"
#define RSI_DHCP_USER_CLASS_DATA_4	"Redpine-class4"

#### To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE             1
```

#### Note:

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

#define DEVICE\_IP

0X0A0AA8C0

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro GATEWAY as **0x010AA8C0**

#define GATEWAY

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro NETMASK as **0x00FFFFFF**

#define NETMASK

0x00FFFFFF

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
(TCP_IP_FEAT_DHCPV4_CLIENT TCP_IP_FEAT_EXTENSION_VALID)	
#define RSI_CUSTOM_FEATURE_BIT_MAP	0
#define RSI_BAND	RSI_BAND_2P4GHZ
#define RSI_EXT_TCP_IP_FEATURE_BIT_MAP	
EXT_TCP_FEAT_DHCP_OPT	77

**Note:**

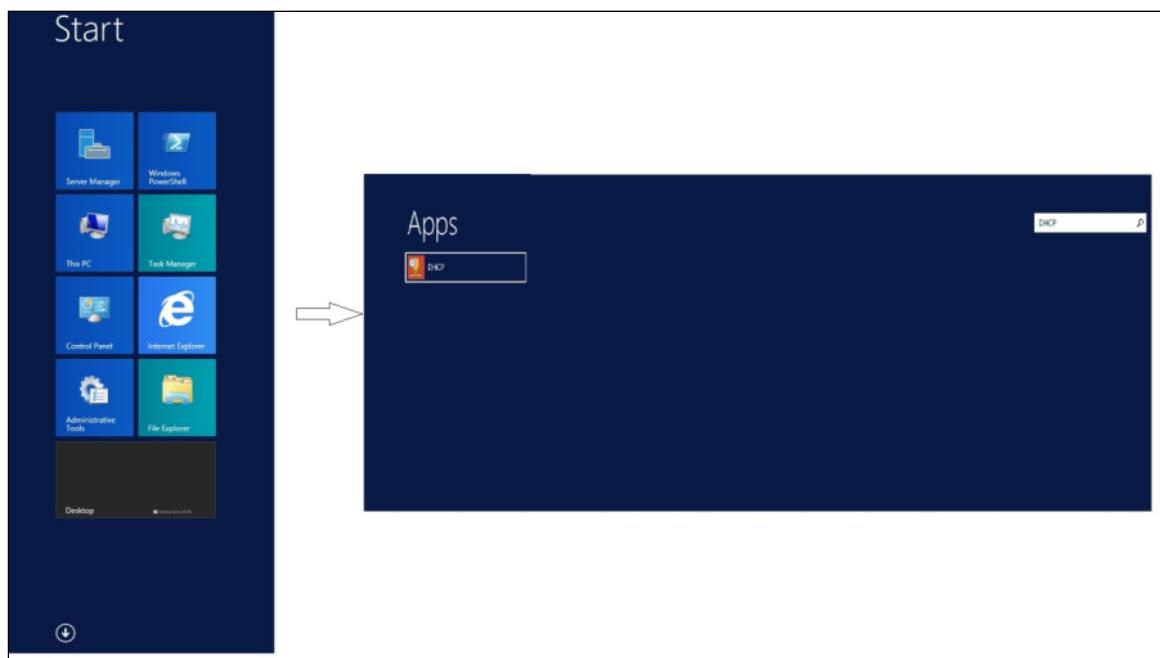
**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example

### 8.7.5 Executing the Application

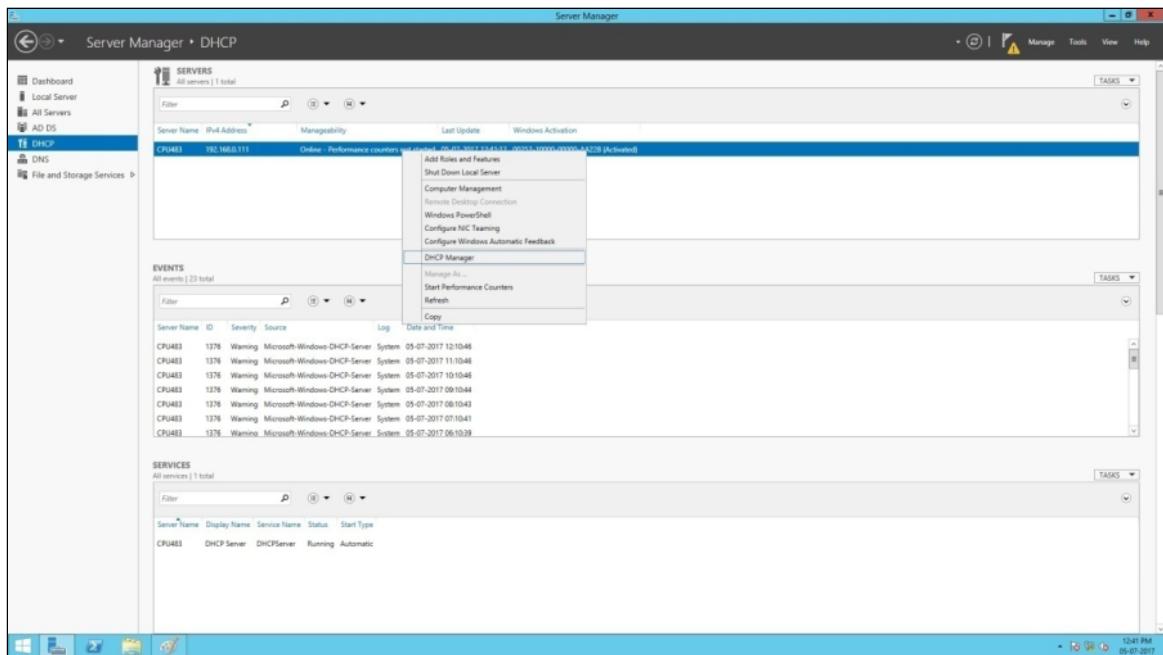
1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. After the program gets executed, the device would be connected to access point .
3. After successful connection with Access Point, Send DHCP command after getting join response.

#### WINDOWS

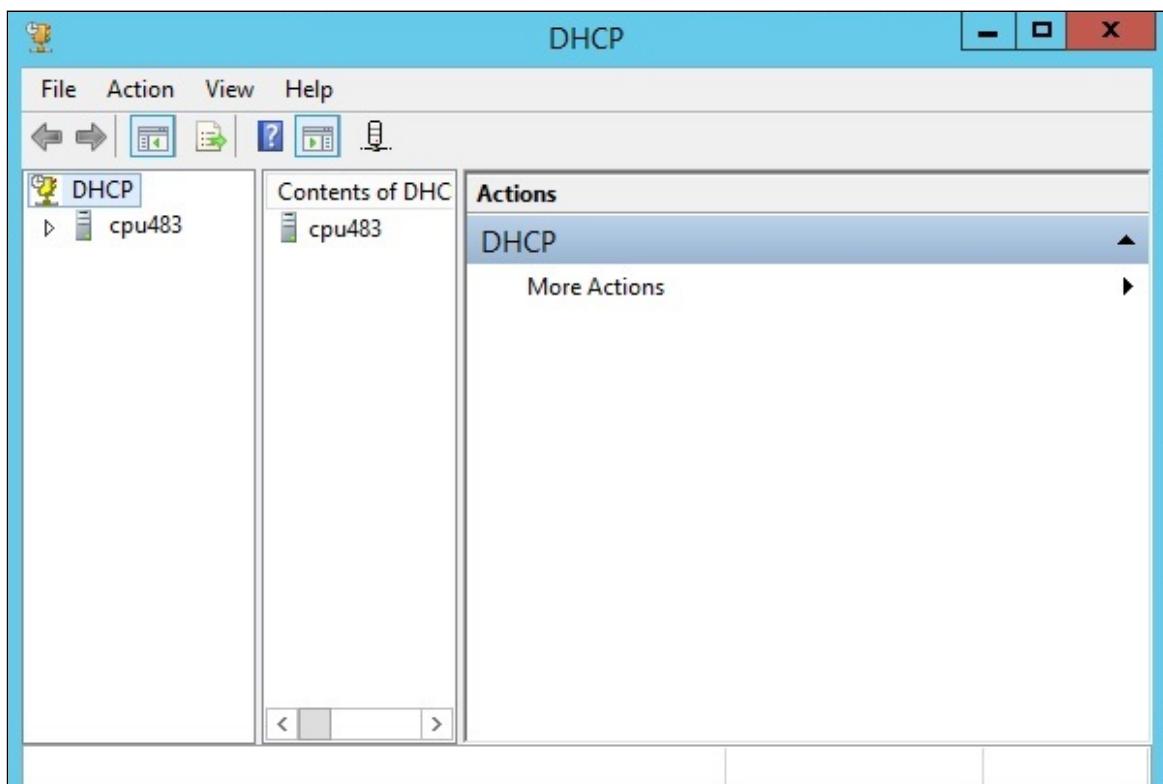
1. Bring up Windows Server 2012.
2. Click on Windows button and select "All apps", search for "DHCP". It opens Server manager.



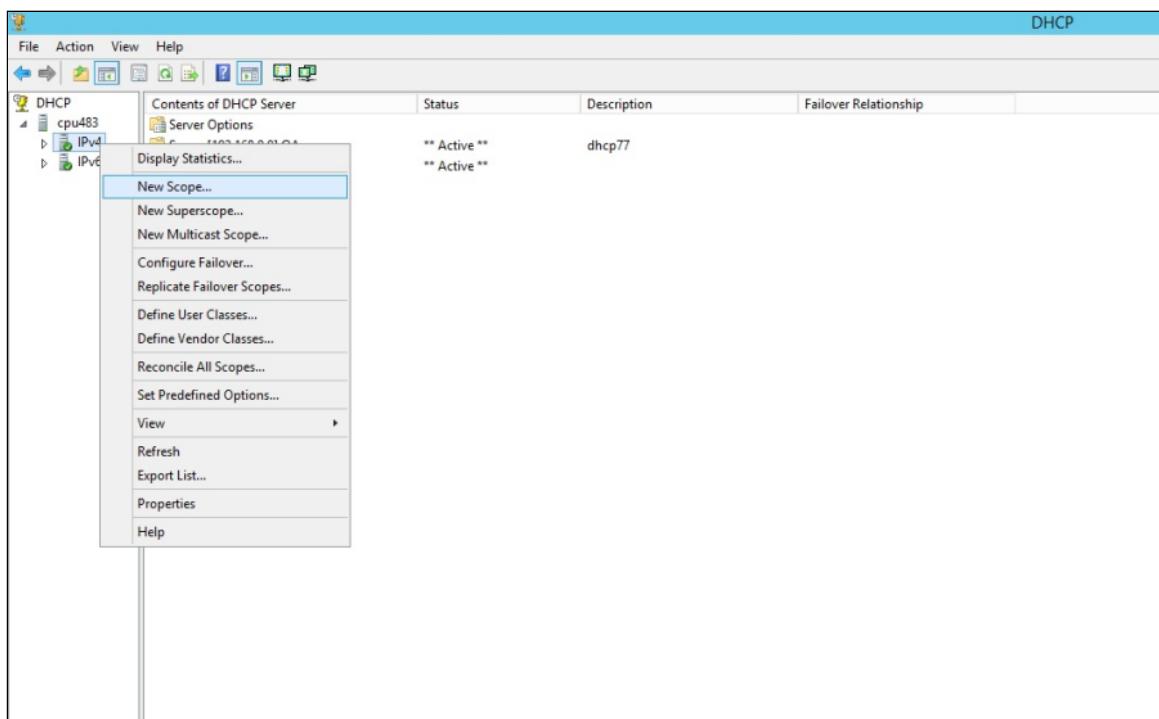
3. Click on DHCP under the Dashboard on the left side of the panel. In the right side of the panel select Server name, right click and select DHCP manager as shown in the below image.



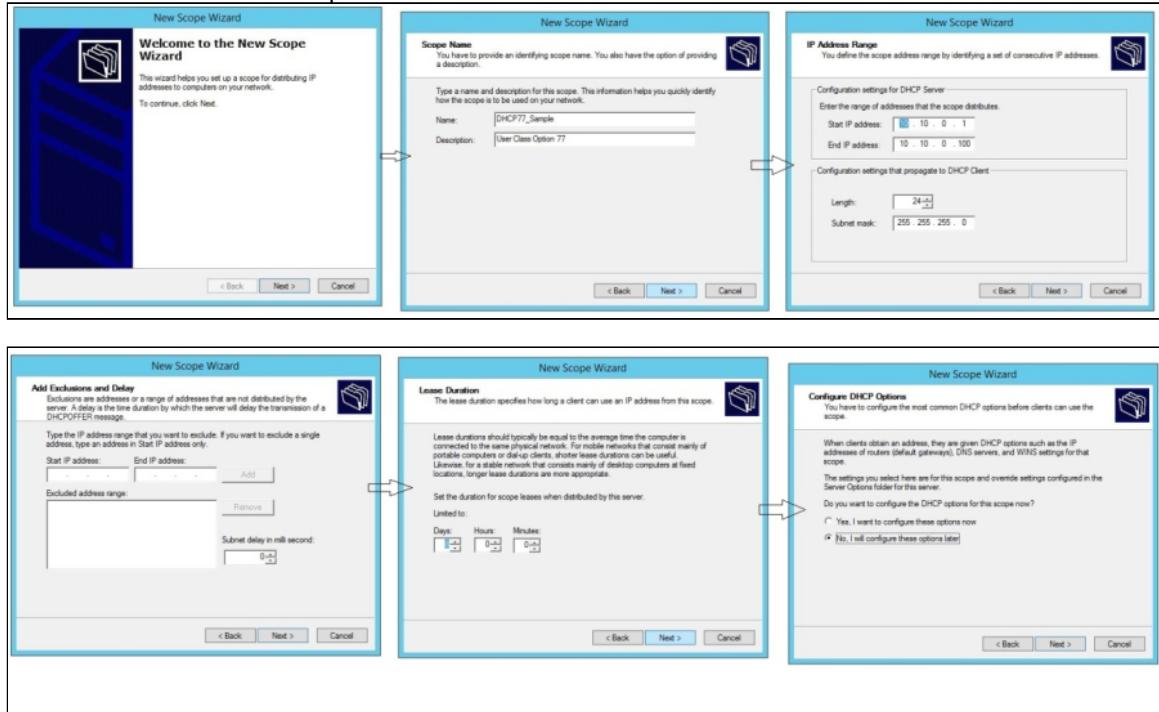
- When DHCP manager is selected it opens a DHCP window as shown below.



- In the left panel select "ipv4" under DHCP and right click to select "New Scope".



6. In the New scope wizard enter the scope name, IP address pool as required, exclusion IP address range and Lease time (usually these couple of options are not needed). Select the option "No" when dialogue box asks to set an option for settings default gateways, DNS servers and WINS settings for the current New scope created. Refer the below set of pictures.

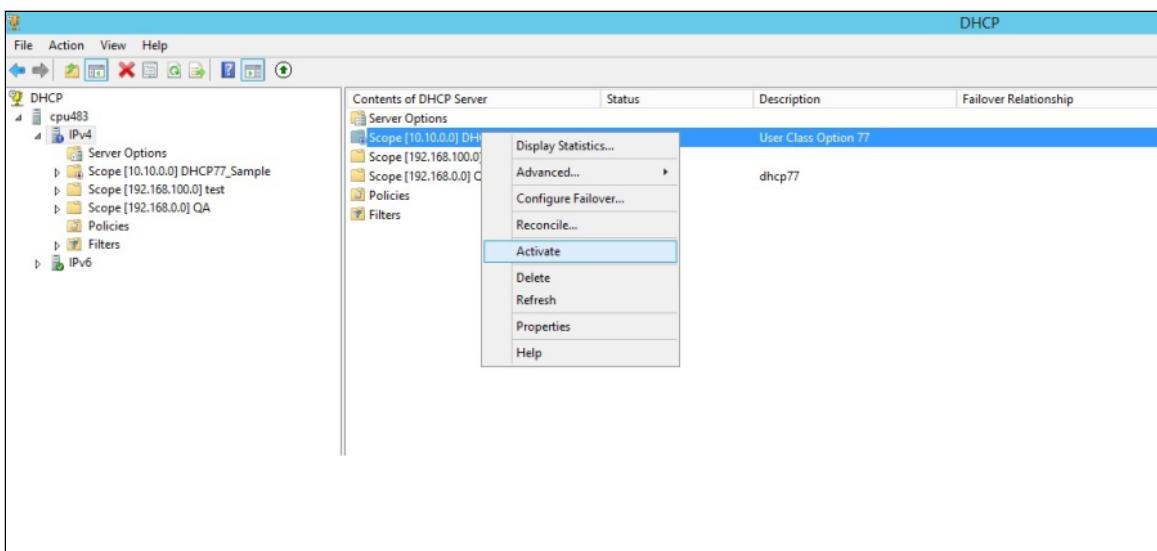




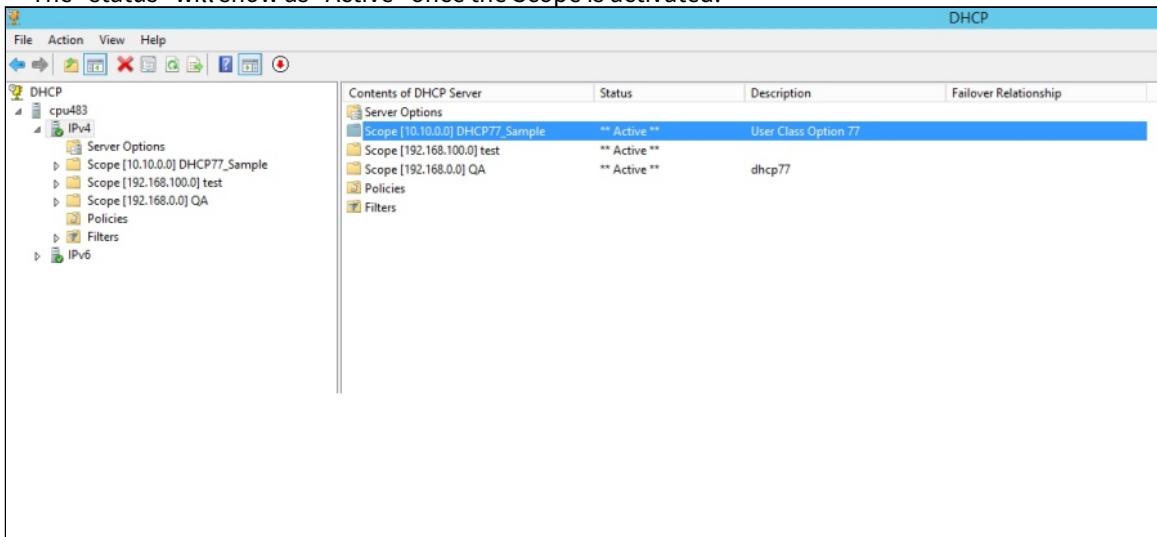
This creates the New scope (DHCP77\_Sample is shown as reference for New scope in the pics).  
7. The New scope created is inactive by default. Right click and Activate as shown below:

The screenshot shows the 'Contents of DHCP Server' table. The table has columns for 'Contents of DHCP Server', 'Status', 'Description', and 'Failover Relationship'. One row is selected, showing 'Scope [10.10.0.0] DHCP77\_Sample' with 'Inactive' status, 'User Class Option 77' description, and no failover relationship. Other rows show 'Server Options', 'Scope [192.168.0.0] QA' (Active), 'Scope [192.168.100.0] test' (Active), 'Policies', and 'Filters'.

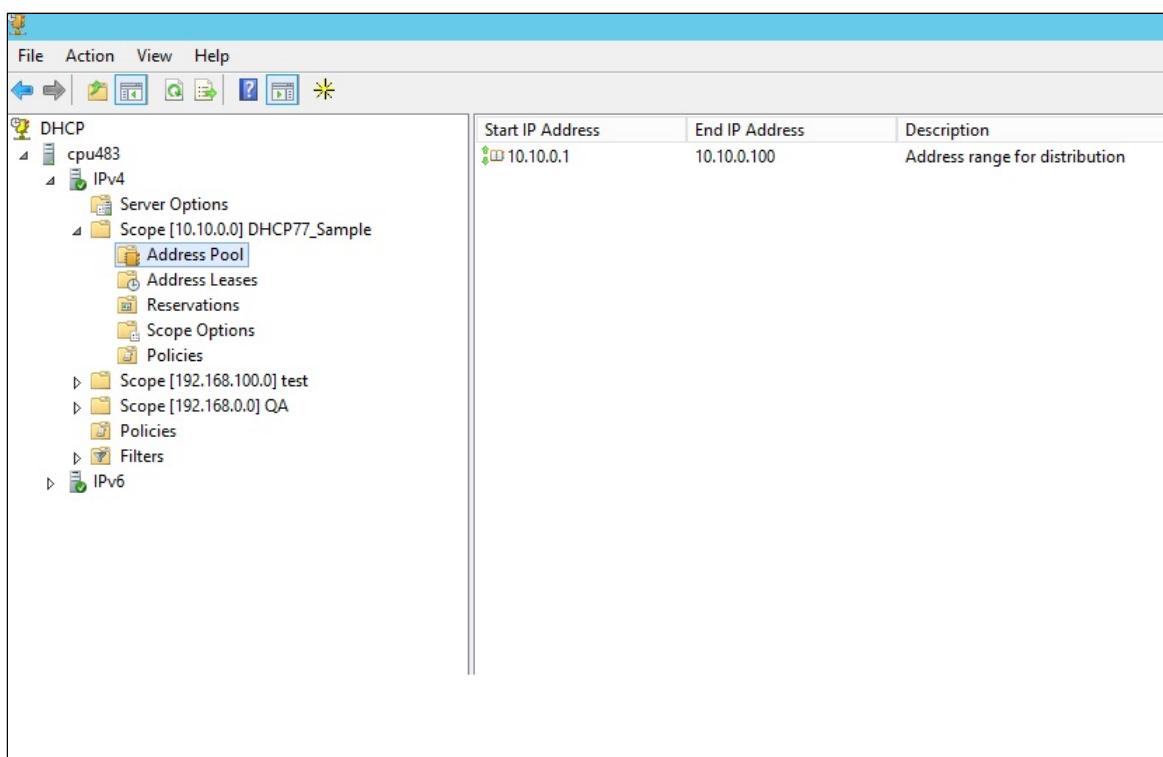
Contents of DHCP Server	Status	Description	Failover Relationship
Scope [10.10.0.0] DHCP77_Sample	Inactive	User Class Option 77	
Server Options			
Scope [192.168.0.0] QA	** Active **	dhcp77	
Scope [192.168.100.0] test	** Active **		
Policies			
Filters			



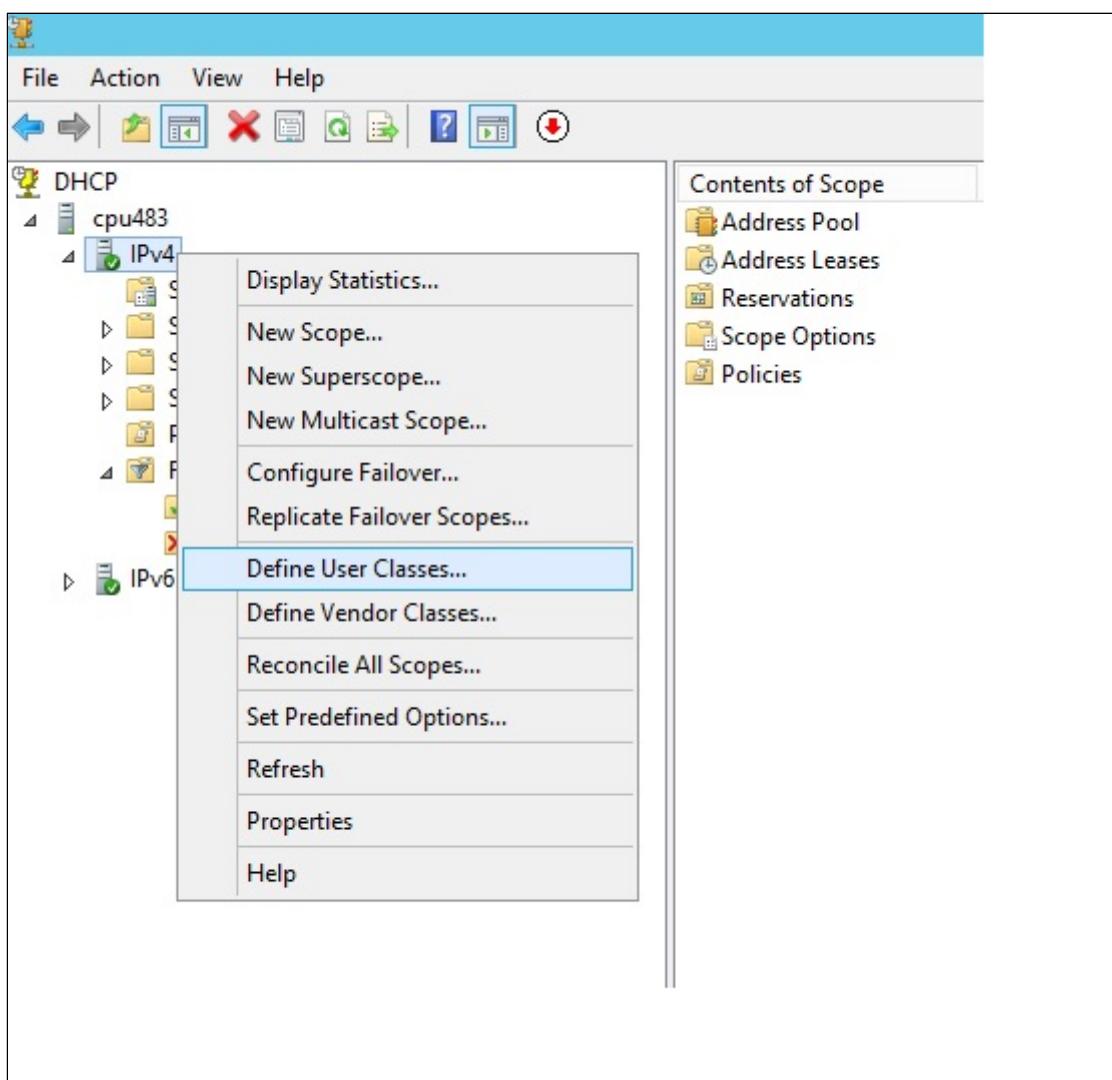
The "status" will show as "Active" once the Scope is activated.



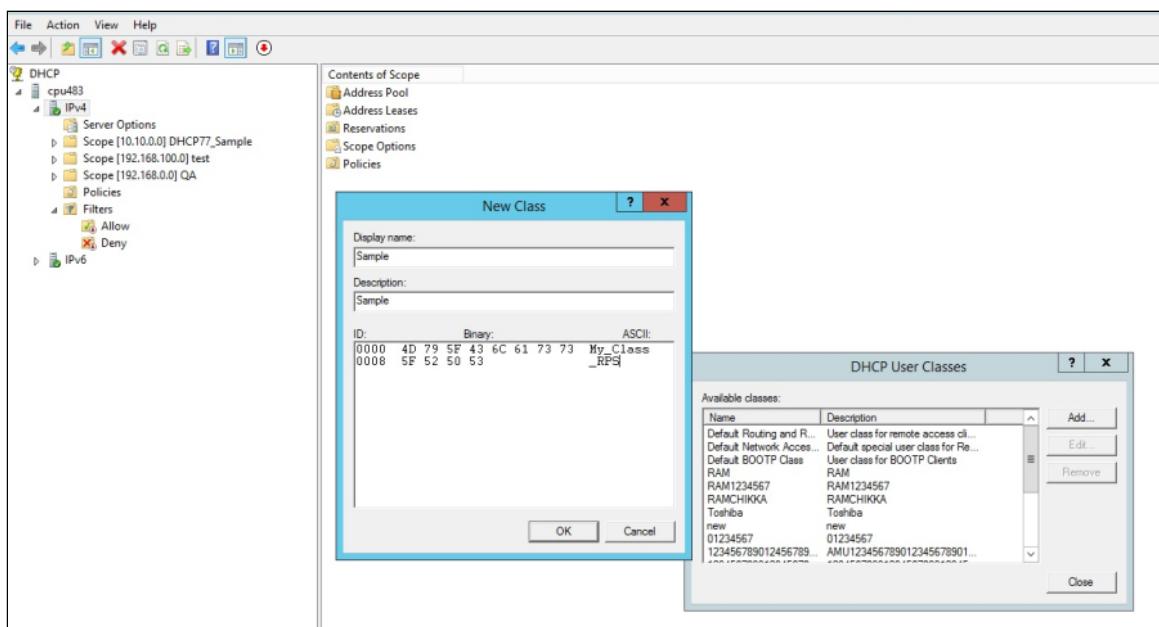
8. On the Left panel of DHCP server manager click on New scope created. It lists out Address pool, Address leases, reservations, Scope options and Policies. Click on Address Pool and observe whether the IP pool displayed in the right side of the panel is same as the IP pool assigned by the user in Step-5.



9. Click on IPv4 and right click, select "Define User classes".



This opens a dialog box with name "DHCP User Classes". Click on "Add". Enter the user name and Description as required in the new dialog box opened and click under space for ASCII. Name the user class as required (Pic below shows user class with name "My\_Class\_RPS").



10. Click on "ok".This created the DHCP class for User Option 77. The same name has to be given in the STA (client) in order to avail DHCP user class option 77.
11. Now click on the "New scope created" and right click on the "Policies", select "New Policy".
12. Name the policy as required. (Ex: Policy name in pic is "Sample\_Policy")

DHCP Policy Configuration Wizard

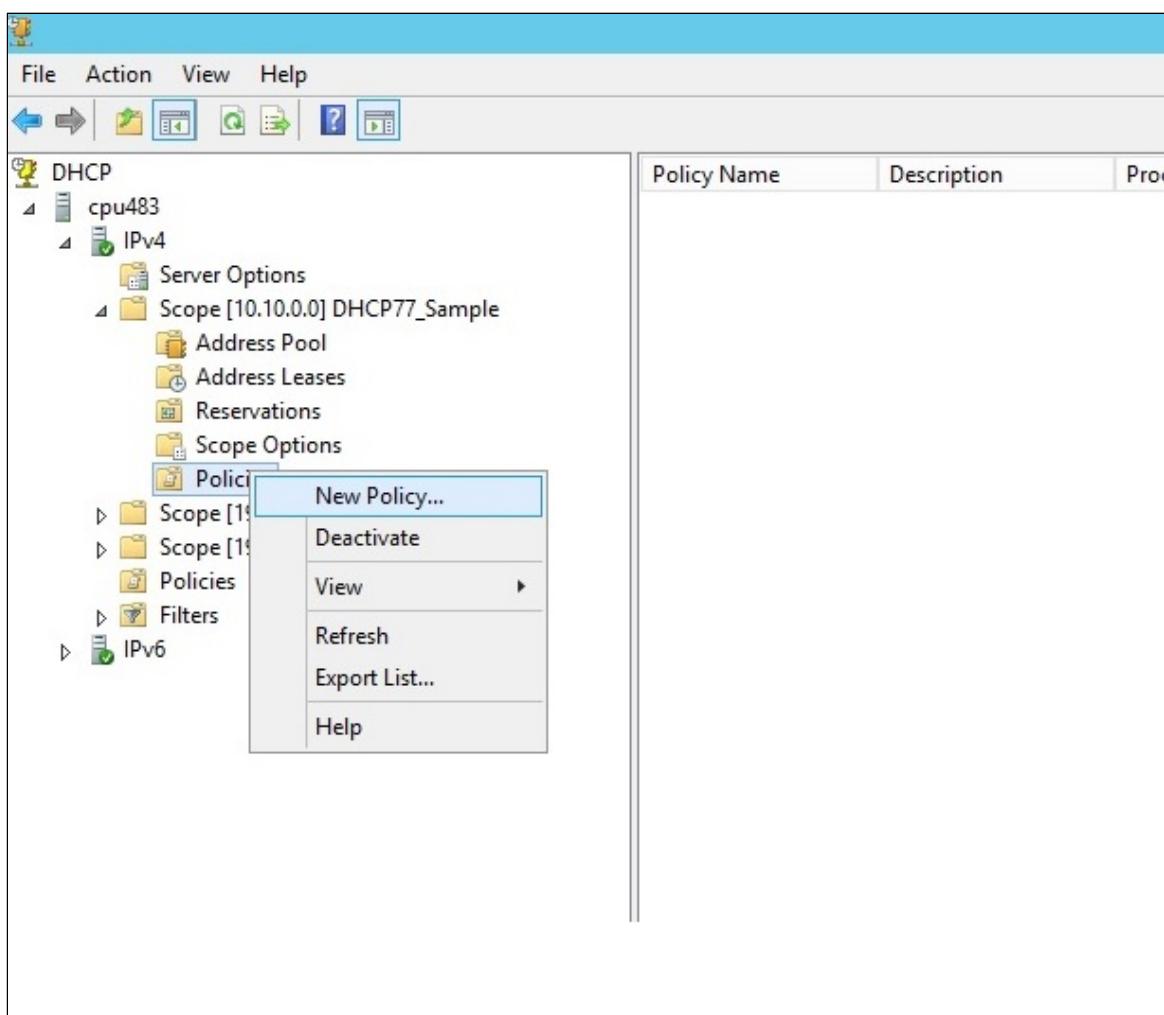
**Policy based IP Address and Option Assignment**

This feature allows you to distribute configurable settings (IP address, DHCP options) to clients based on certain conditions (e.g. vendor class, user class, MAC address, etc.).

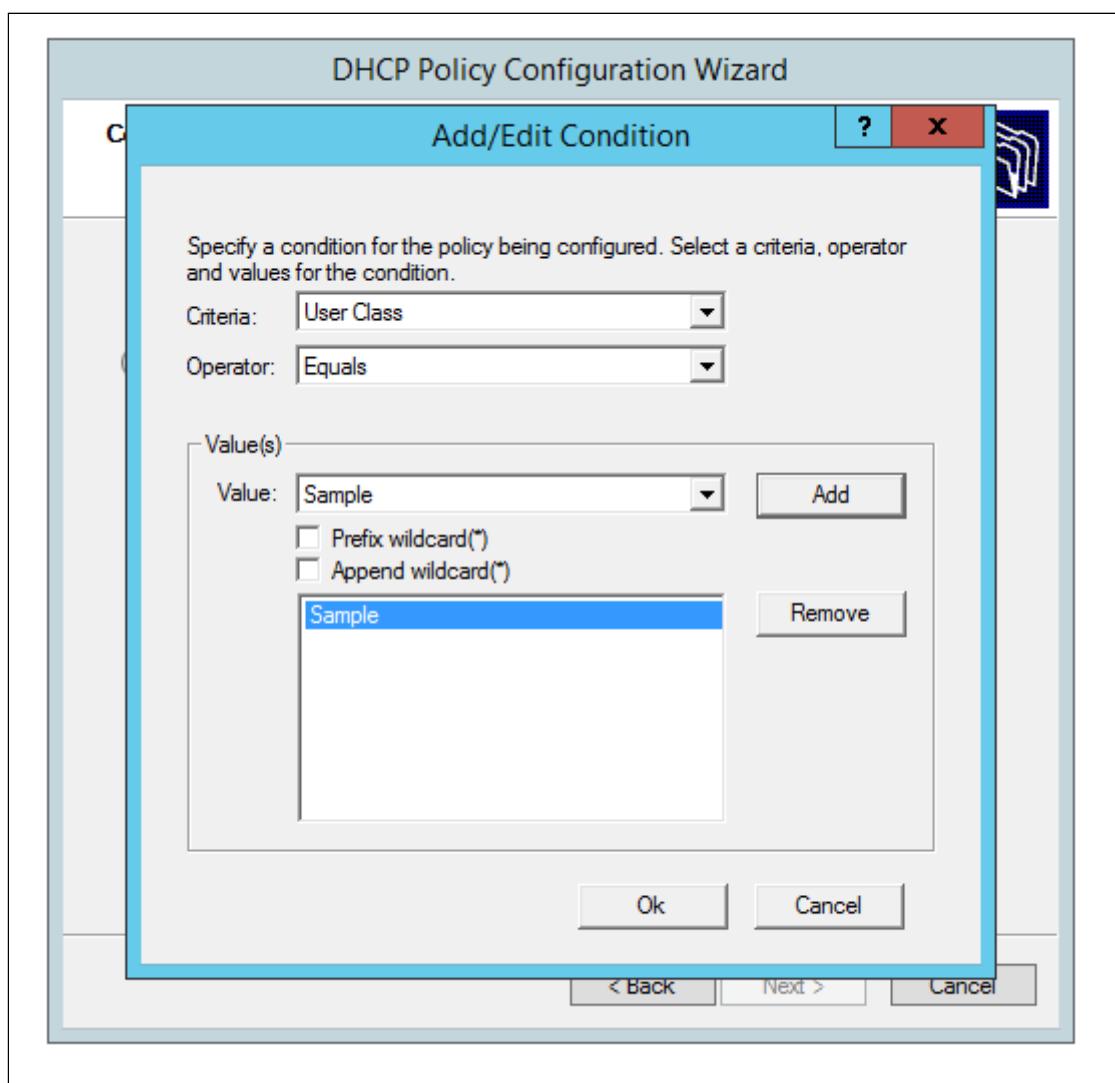
This wizard will guide you setting up a new policy. Provide a name (e.g. VoIP Phone Configuration Policy) and description (e.g. NTP Server option for VoIP Phones) for your policy.

Policy Name:

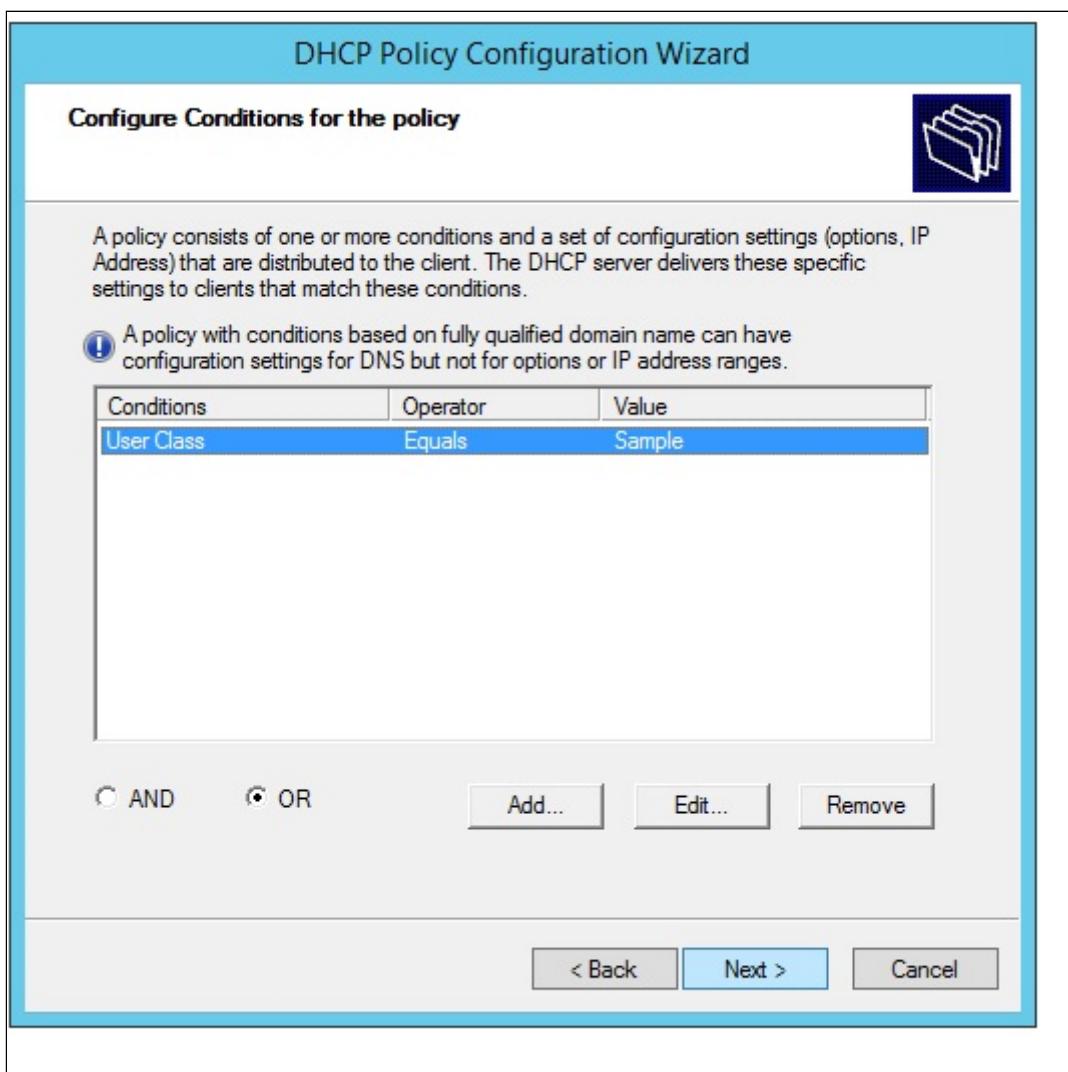
Description:



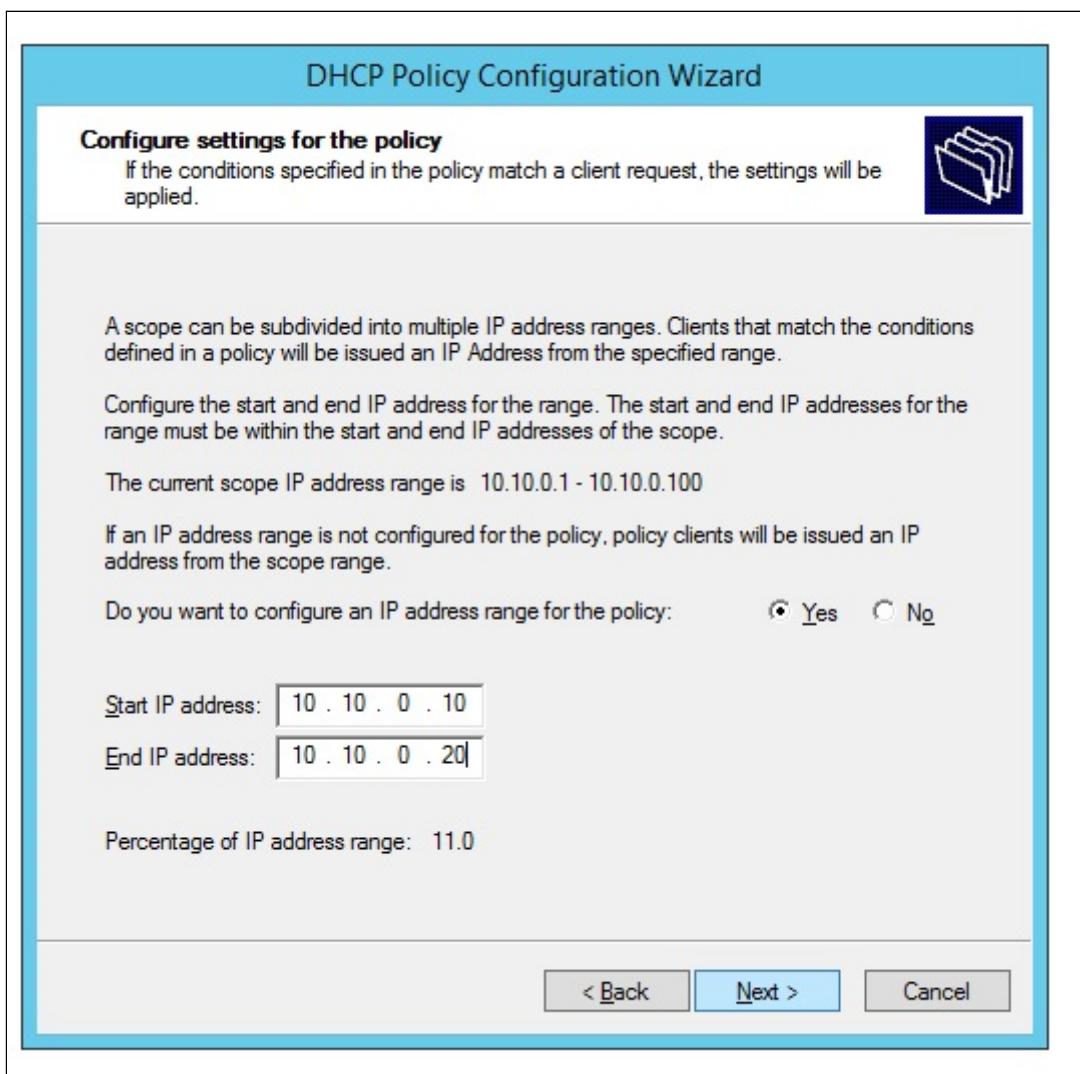
13. This opens dialog box. DHCP Policy Configuration Wizard. Click on "Add" in the dialog box opened. Select the "Criteria" as "User Class" and "Operator" as "Equals". Select value as the User-class created at step-8. (Ex: Sample is the User class created in the step-8 in the Pic)



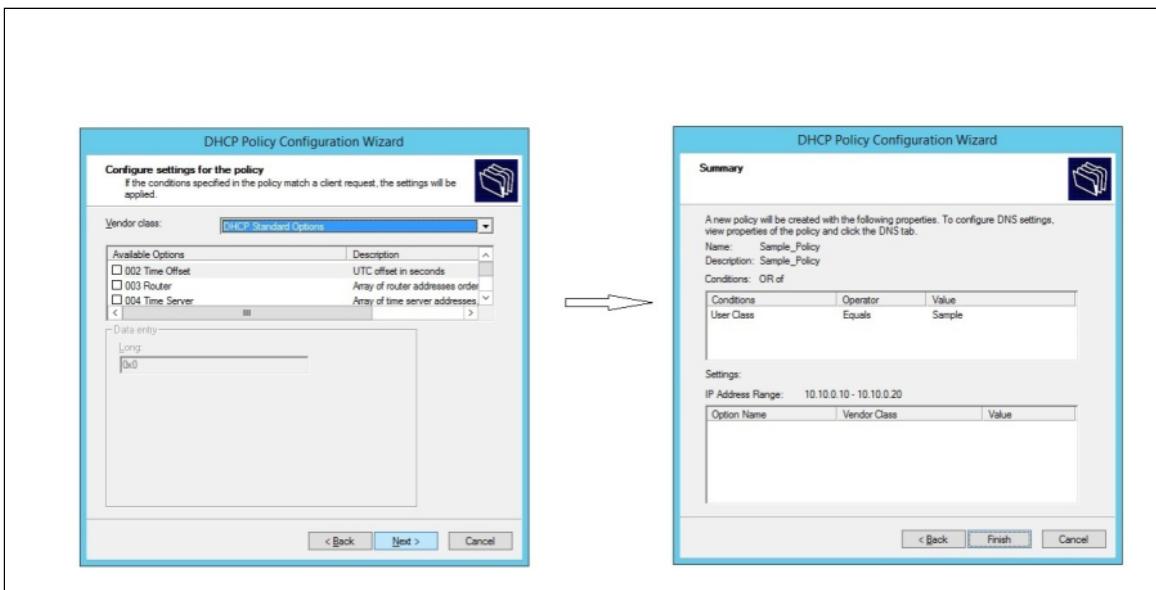
14. Click Next.



15. Select a range of IP addresses for the User class 77 and this pool must be a subset of Ip address range declared by user in Step-5.



16. Click Next to continue and finally click "Finish"



17. Finally the policies are shown in the right panel with the User class option 77 pool declared by the user.  
 (Ex: Pic shows a policy by name "Sample\_policy" with user address pool 10.10.0.10-10.10.0.20)

**Note:**

The User policy created can be verified using a Windows STA. Independent of Server (whether Linux or Windows) the usage of DHCP 77 option command is same in the Windows STA.

**Special Note:**

Windows Server 2012 supports only Single user class in the DHCP request sent from a Client. It does not support "Multiple user Class".

## 8.8 Enterprise client Example

### 8.8.1 Example Overview

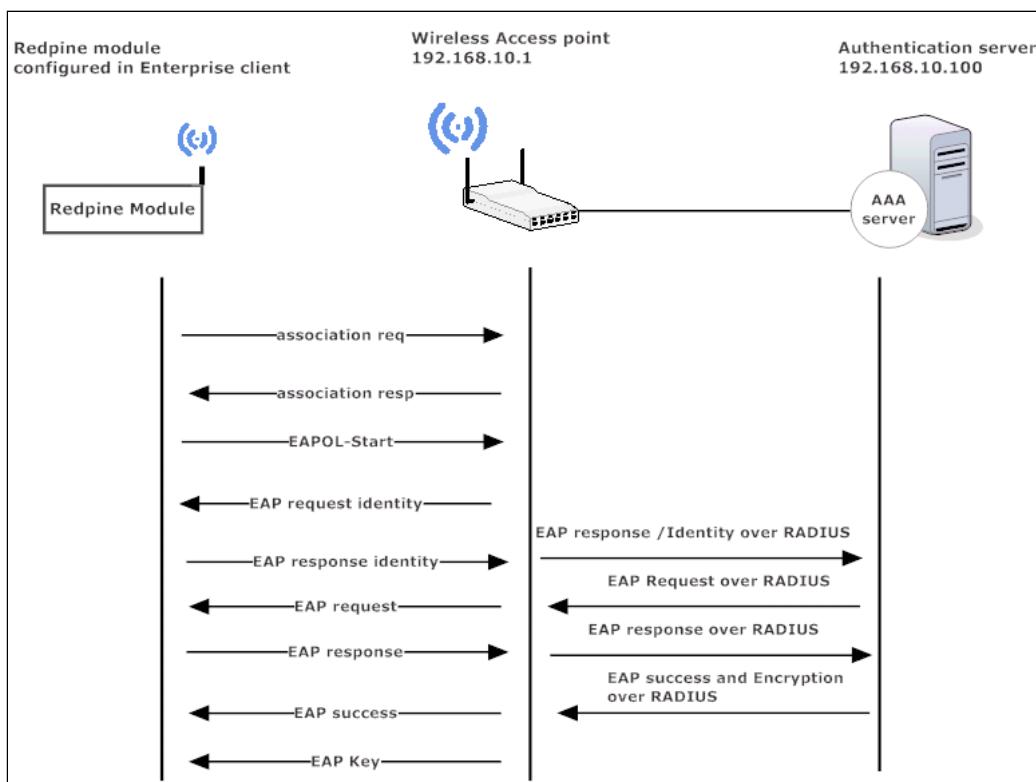
#### Overview

This Application demonstrates how to configure device in Enterprise client and connects with Enterprise secured AP and data traffic in Enterprise security mode.

In this application, the device connects to Enterprise secured AP using EAP-TLS/TTLS/PEAP/FAST method. After successful connection, Application established TCP client connection with TCP server opened on remote peer and sends TCP data on opened socket.

#### EAP overview

In wireless communications using EAP, a user requests connection to a WLAN through an AP, which then requests the identity of the user and transmits that identity to an authentication server such as RADIUS. The server asks the AP for proof of identity, which the AP gets from the user and then sends back to the server to complete the authentication.



**Figure 1: EAPOL-Keys exchange**

## Sequence of Events

This Application explains user how to:

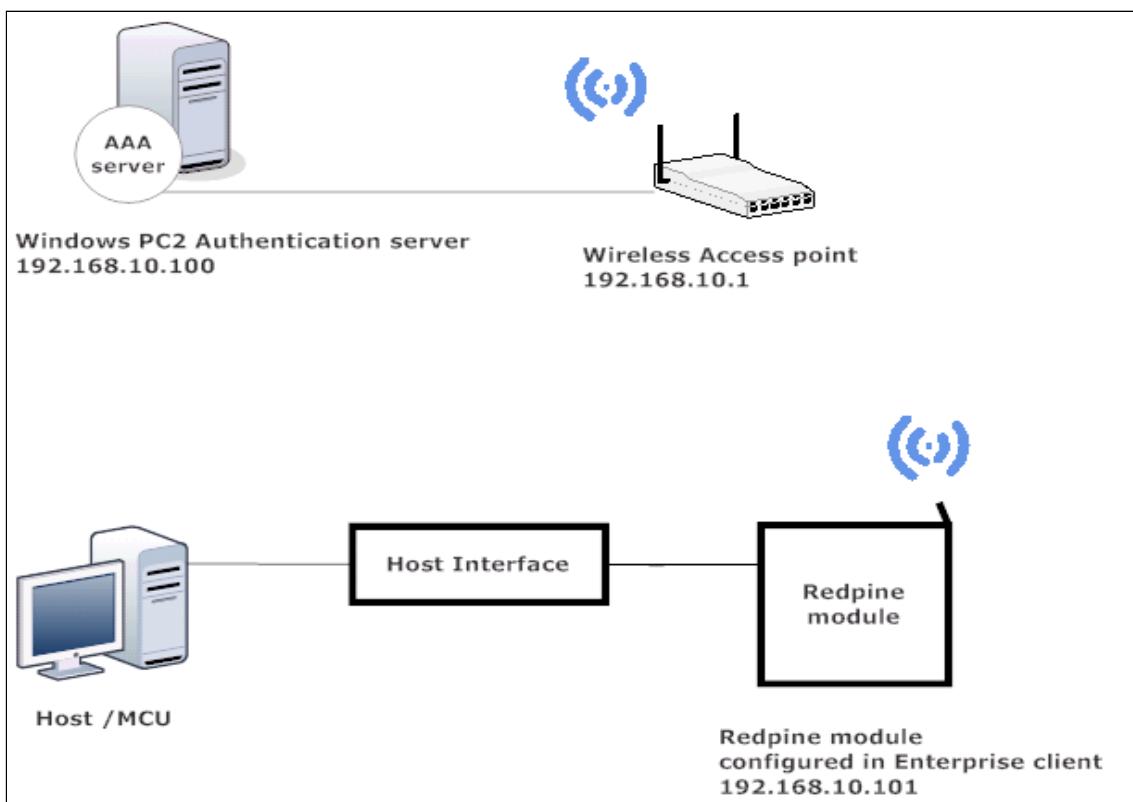
- Configure device as an Enterprise client
- Connect with Enterprise secured AP using EAP-TLS/TTLS/PEAP/FAST method
- Establish TCP connection from connected Redpine device to TCP server opened on remote peer.
- Send TCP data from the device to remote peer.

### 8.8.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with Keil or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine Module
- Windows/Linux PC2 with AAA Radius Server or Free Radius server
- TCP server application running on Windows/Linux PC2 (This example uses iperf for windows ).



**Figure 2: Setu0p Diagram**

### 8.8.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_eap\_connectivity.c** file and update/modify following macros  
**SSID** refers to the name of the Access point.

```
#define SSID  
"REDPINE_AP"
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports WPA-EAP, WPA2-EAP securities.

Valid configuration is:

**RSI\_WPA\_EAP** - For WPA-EAP security mode  
**RSI\_WPA2\_EAP** - For WPA2-EAP security mode

```
#define SECURITY_TYPE  
RSI_WPA2_EAP
```

### To Load certificate

**LOAD\_CERTIFICATE** refers whether certificate to load into module or not.

```
#define LOAD_CERTIFICATE
```

1

If **LOAD\_CERTIFICATE** set to 1, application will load certificate which is included using **rsi\_wlan\_set\_certificate** API.

By default, application is loading "wifiuser.pem" certificate when **LOAD\_CERTIFICATE** enable. In order to load different certificate, user has to do the following steps:

- **rsi\_wlan\_set\_certificate** API expects the certificate in the form of linear array. So, convert the pem certificate into linear array form using python script provided in the release package "**utilities/certificates/certificate\_script.py**"

Ex: If the certificate is wifi-user.pem .Give the command in the following way :

```
python certificate_script.py wifi-user.pem
```

Script will generate wifiuser.pem in which one linear array named wifiuser contains the certificate.

- After conversion of certificate, update **rsi\_eap\_connectivity.c** source file by including the certificate file and by providing the required parameters to **rsi\_wlan\_set\_certificate** API.
- Once certificate loads into the device, it will write into the device flash. So, user need not load certificate for every boot up unless certificate change.

So define **LOAD\_CERTIFICATE** as 0, if certificate is already present in the device.

**USER\_IDENTITY** refers to user ID which is configured in the user configuration file of the radius server. In this example, user identity is "user1".

```
#define USER_IDENTITY  
"\\"user1\\\""
```

**PASSWORD** refers to the password which is configured in the user configuration file of the Radius Server for that User Identity.

In this example, password is "test123"

```
#define PASSWORD  
"\\"test123\\\""
```

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT  
5001
```

**SERVER\_PORT** port refers remote TCP server port number which is opened in Windows PC2.

```
#define SERVER_PORT  
5001
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with TCP server socket.

IP address should be in long format and in little endian byte order.  
Example: To configure "192.168.0.100" as remote IP address, update the macro **SERVER\_IP\_ADDRESS** as **0x6400AA8C0**.

```
#define SERVER_IP_ADDRESS  
0x640AA8C0
```

**NUMBER\_OF\_PACKETS** refers how many packets to receive from TCP client

```
#define NUMBER_OF_PACKETS  
1000
```

**To configure IP address in STA mode**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC in STA mode

```
#define DHCP_MODE
```

1

**Note:**

If the user wants to configure STA IP address through DHCP then skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If the user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.0.10" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP  
0X0A00A8C0
```

IP address of the gateway should also be in long format and in little endian byte order.

Example: To configure "192.168.0.1" as Gateway, update the macro **GATEWAY** as **0x0100A8C0**

```
#define GATEWAY  
0x0100A8C0
```

IP address of the network mask should also be in long format and in little endian byte order.

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK  
0x00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE  
RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP  
FEAT_SECURITY_PSK  
#define RSI_TCP_IP_BYPASS  
RSI_DISABLE  
#define RSI_TCP_IP_FEATURE_BIT_MAP  
TCP_IP_FEAT_DHCPV4_CLIENT  
#define RSI_CUSTOM_FEATURE_BIT_MAP  
#define RSI_BAND  
RSI_BAND_2P4GHZ
```

0

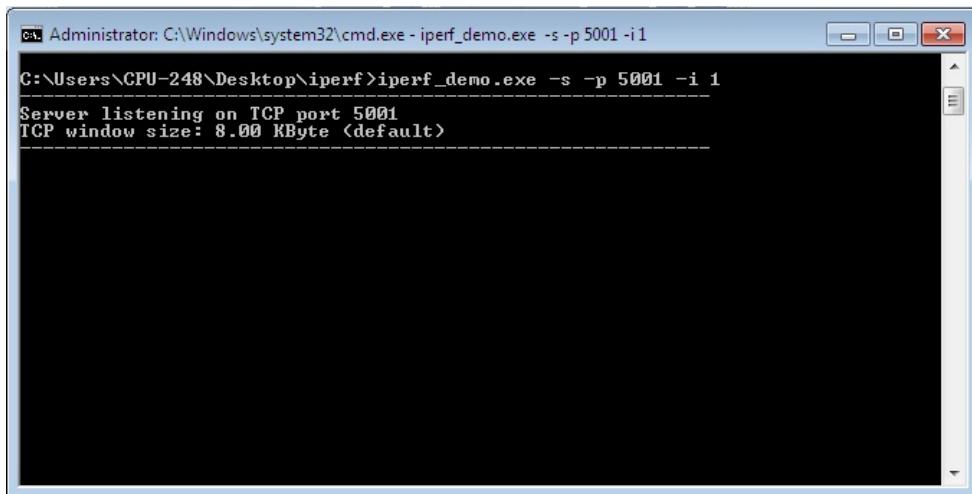
**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

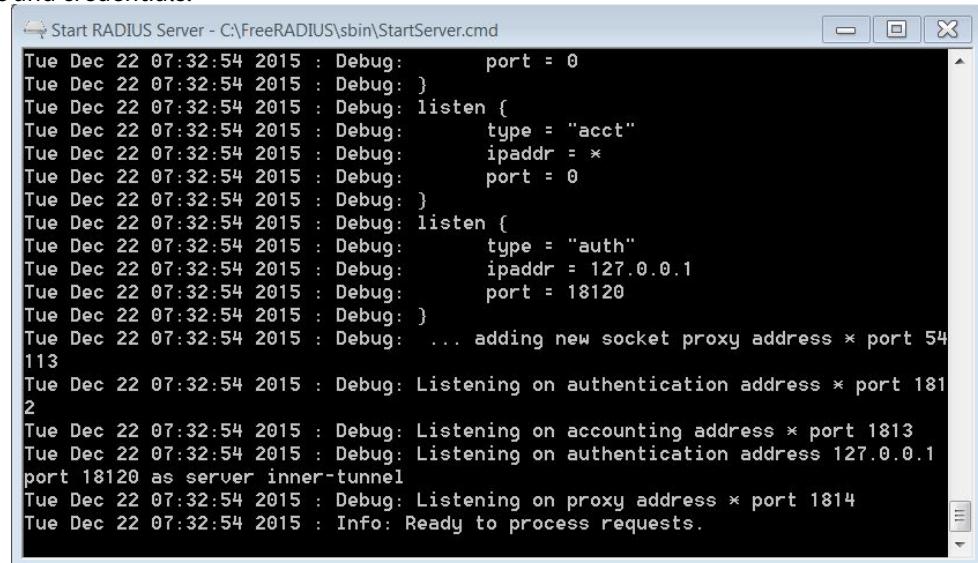
## Executing the Application

1. Configure the Access point in WPA-EAP/WPA2-EAP mode to connect the Redpine device in enterprise secured mode.
2. Open TCP server application using iperf application in Windows PC2 which is connected to the access point through LAN.

**iperf\_demo.exe -s -p <SERVER\_PORT> -i 1**



3. Run Radius server in Windows/Linux PC2 which is connected to AP through LAN by providing required certificate and credentials.

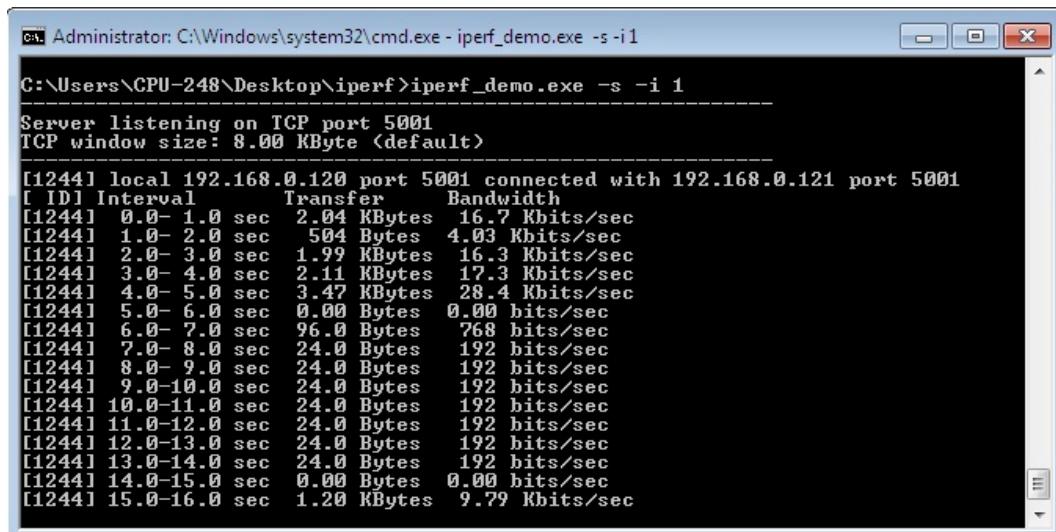


```

Tue Dec 22 07:32:54 2015 : Debug:      port = 0
Tue Dec 22 07:32:54 2015 : Debug: }
Tue Dec 22 07:32:54 2015 : Debug: listen {
Tue Dec 22 07:32:54 2015 : Debug:      type = "acct"
Tue Dec 22 07:32:54 2015 : Debug:      ipaddr = *
Tue Dec 22 07:32:54 2015 : Debug:      port = 0
Tue Dec 22 07:32:54 2015 : Debug: }
Tue Dec 22 07:32:54 2015 : Debug: listen {
Tue Dec 22 07:32:54 2015 : Debug:      type = "auth"
Tue Dec 22 07:32:54 2015 : Debug:      ipaddr = 127.0.0.1
Tue Dec 22 07:32:54 2015 : Debug:      port = 18120
Tue Dec 22 07:32:54 2015 : Debug: }
Tue Dec 22 07:32:54 2015 : Debug: ... adding new socket proxy address * port 54113
Tue Dec 22 07:32:54 2015 : Debug: Listening on authentication address * port 1812
Tue Dec 22 07:32:54 2015 : Debug: Listening on accounting address * port 1813
Tue Dec 22 07:32:54 2015 : Debug: Listening on authentication address 127.0.0.1 port 18120 as server inner-tunnel
Tue Dec 22 07:32:54 2015 : Debug: Listening on proxy address * port 1814
Tue Dec 22 07:32:54 2015 : Info: Ready to process requests.

```

4. After the program gets executed, Redpine device would be connected to access point which is in enterprise security having the configuration same as that of in the application and get IP.  
 5. After successful connection, device STA connects to TCP server socket opened on Windows/Linux PC2 using TCP client socket and sends configured **NUMBER\_OF\_PACKETS** to remote TCP server. Refer the below image for reception of TCP data on TCP server.



```

C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -i 1
-----
[1244] local 192.168.0.120 port 5001 connected with 192.168.0.121 port 5001
[ ID] Interval Transfer Bandwidth
[1244] 0.0- 1.0 sec 2.04 KBytes 16.7 Kbits/sec
[1244] 1.0- 2.0 sec 504 Bytes 4.03 Kbits/sec
[1244] 2.0- 3.0 sec 1.99 KBytes 16.3 Kbits/sec
[1244] 3.0- 4.0 sec 2.11 KBytes 17.3 Kbits/sec
[1244] 4.0- 5.0 sec 3.47 KBytes 28.4 Kbits/sec
[1244] 5.0- 6.0 sec 0.00 Bytes 0.00 bits/sec
[1244] 6.0- 7.0 sec 96.0 Bytes 768 bits/sec
[1244] 7.0- 8.0 sec 24.0 Bytes 192 bits/sec
[1244] 8.0- 9.0 sec 24.0 Bytes 192 bits/sec
[1244] 9.0-10.0 sec 24.0 Bytes 192 bits/sec
[1244] 10.0-11.0 sec 24.0 Bytes 192 bits/sec
[1244] 11.0-12.0 sec 24.0 Bytes 192 bits/sec
[1244] 12.0-13.0 sec 24.0 Bytes 192 bits/sec
[1244] 13.0-14.0 sec 24.0 Bytes 192 bits/sec
[1244] 14.0-15.0 sec 0.00 Bytes 0.00 bits/sec
[1244] 15.0-16.0 sec 1.20 KBytes 9.79 Kbits/sec

```

---

## 8.9 Firmware Upgradation From Server Example

### 8.9.1 Example Overview

#### Overview

This application demonstrates how to upgrade new firmware to Redpine device using remote TCP server. In this application, the device connects to access point and establishes TCP client connection with TCP server opened on remote peer. After successful TCP connection, application sends the firmware file request to remote TCP server and server responds with Firmware file and waits for the next firmware file request. Once firmware file receives from the TCP server, application loads the firmware file into device using firmware upgrade API and gets next firmware file from TCP server. After successful firmware upgrade, firmware upgrade API returns 0x03 response.

#### Sequence of Events

This Application explains user how to:

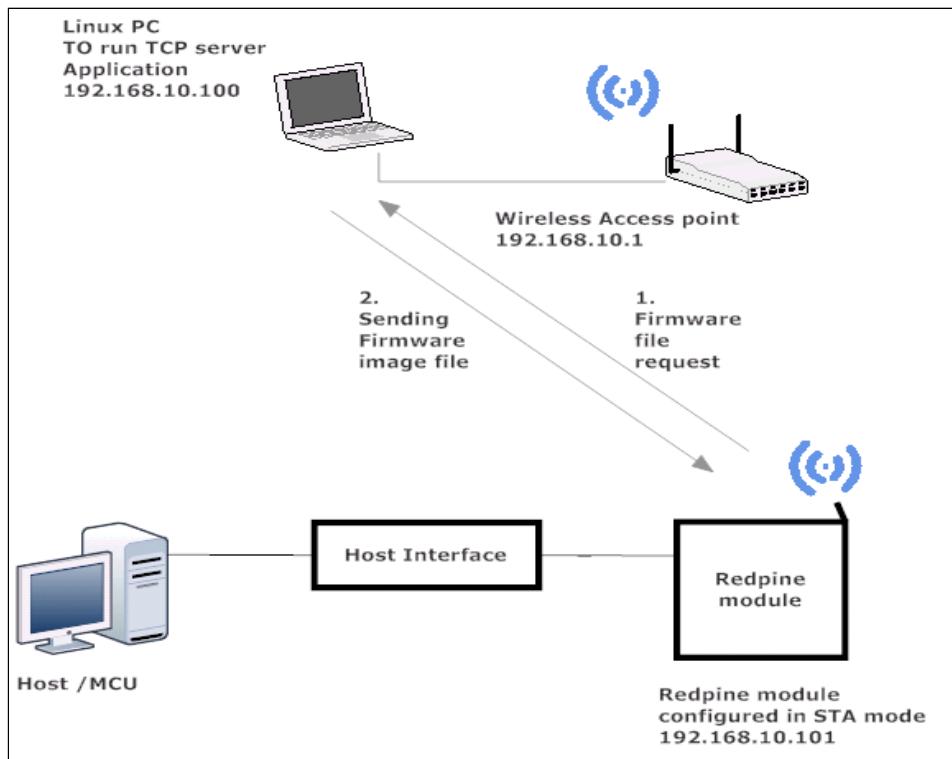
- Configure as station mode
- Open TCP server socket at Access Point
- Connect to Access Point and open TCP client socket
- Request Firmware file from remote server
- Send firmware file from remote server
- Upgrade the received Firmware into the device.

### 8.9.2 Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with Keil or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine Module
- Wireless Access point
- Linux PC with TCP server application (TCP server application providing as part of release package)



**Figure 1: Setup Diagram**

### 8.9.3 Configuration and Execution of the Application

**Note:**

Do not add `firmware_upgrade_tcp_server.c` file which is present in `Firmware_Upgrade` folder MCU project.

#### Configuring the Application

1. Open `rsi_firmware_upgradation_app.c` file and update/modify following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE      RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK "psk"</pre>
```

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT 5001</pre>
```

**SERVER\_PORT** port refers remote TCP server port number which is opened in Linux PC.

```
#define SERVER_PORT 5001</pre>
```

**SERVER\_IP\_ADDRESS** refers remote peer IP (Linux PC) address to connect with TCP server socket.  
IP address should be in long format and in little endian byte order.

Example: To configure "192.168.0.100" as remote IP address, update the macro **SERVER\_IP\_ADDRESS** as **0x6400A8C0**.

```
#define SERVER_IP_ADDRESS 0x6400A8C0</pre>
```

**RECV\_BUFFER\_SIZE** refers Memory for receive data

```
#define RECV_BUFFER_SIZE 1027</pre>
```

#### To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE 1</pre>
```

#### Note:

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

The IP address needs to be configuring to the device should be in long format and in little endian byte order.  
Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP 0X0A0AA8C0</pre>
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

#define GATEWAY	0x010AA8C0
-----------------	------------

IP address of the network mask should also be in long format and in little endian byte order.  
Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK	0x00FFFFFF
-----------------	------------

2. Open **rsi\_wlan\_config.h** file and update/modify following macros :

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	
FEAT_CUSTOM_FEAT_EXTENSION_VALID	
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP	EXT_FEAT_256K_MODE
#define RSI_BAND	RSI_BAND_2P4GHZ

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

#### 8.9.4 Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. Copy TCP server application present in release package "**firmware\_upgradation/firmware\_upgrade\_tcp\_server.c**" into Linux PC which is connected to access point through LAN.  
Compile and run by providing port number and Firmware file path
3. compile by giving **gcc firmware\_upgrade\_tcp\_server.c** command
4. Run the application **./a.out 5001 RS9116.NBZ.WC.GEN.OSI.x.x.x.rps**
5. After the program gets executed, device connects to AP and open TCP client socket.
6. After TCP connection established with remote server, application sends firmware file request to the server.
7. Server receive request and sends firmware file in chunks.
8. After receiving chunk from remote server, application again sends firmware request to server. Server will wait for the firmware request from the device before sending next chunk.
9. Packet is sent to the device in chunks as shown in the given below figure. After successful upgradation in TCP server terminal shows "reach end of file".

```

File Edit View Terminal Tabs Help
root@localhost:/tftpboot/nfs/lib/modules/2.6.... x root@localhost:/work/siva/RS9113.NBZ.WC.GE... x
size of data1==1024
send returns 1027
Pkt sent no:1529
waiting for recv
recv length == 0x3
size of data1==1024
send returns 1027
Pkt sent no:1530
waiting for recv
recv length == 0x3
size of data1==1024
send returns 1027
Pkt sent no:1531
waiting for recv
recv length == 0x3
size of data1==1024
send returns 1027
Pkt sent no:1532
waiting for recv
recv length == 0x3
size of data1==1024
send returns 1027
Pkt sent no:1533
waiting for recv
recv length == 0x3
size of data1==1024
send returns 1027
Pkt sent no:1534
waiting for recv
recv length == 0x3
size of data1==1024
send returns 1027
Pkt sent no:1535
waiting for recv
recv length == 0x3
reach end of file

```

10. In Application ***rsi\_firmware\_upgradation\_app.c,rsi\_fwup\_load*** API returns 0x03 response after successful firmware up gradation and closes TCP client socket.

**Note:**

After Firmware up-gradation, Device needs to be reboot to get effective of new firmware file. After reboot, Device will take few minutes to give CARD READY indication after first reboot.  
Wait for few minutes after power up.

## 8.10 FTP Client Example

### 8.10.1 Protocol Overview

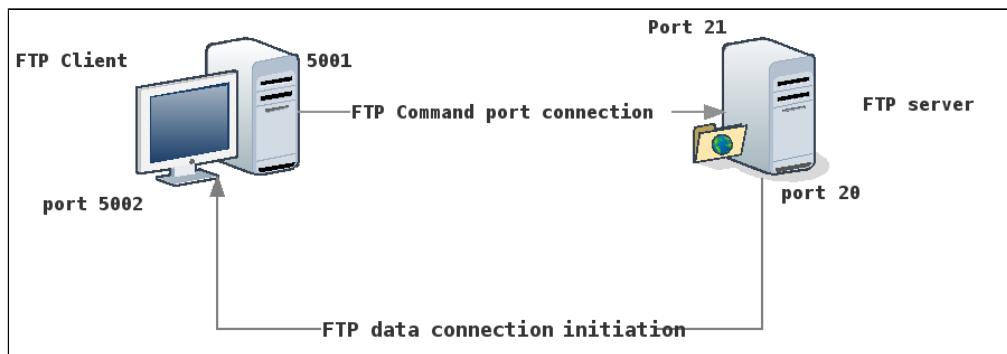
File Transfer Protocol (FTP) is a protocol through which internet users can upload files from their computers to a website or download files from a website to their PCs.

FTP is a client-server protocol that relies on two TCP communications channels between client and server and a command channel for controlling the conversation (command port) and a data channel for transmitting file content(data port). The standard port number used by FTP servers is 21 and is used only for sending commands. Clients initiate conversations with servers by requesting to download a file. Using FTP, a client can upload, download, delete, rename, move and copy files on a server. A user typically needs to log on to the FTP server, although some servers make some or all of their content available without login, which is also known as anonymous FTP.

FTP sessions work in passive or active modes. In active mode, after a client initiates a session via a command channel request, the server initiates a data connection back to the client and begins transferring data. In passive mode, the server instead uses the command channel to send the client the information it needs to open a data channel.

**Note:**

Redpine devices support only Active mode of FTP sessions.



**Figure 1: Simple FTP connection between FTP Server and FTP client**

#### 8.10.2 Example Overview

##### Overview

This application demonstrates how to connect to FTP server opened on remote peer using FTP client and how to read file from FTP server and how to write file on to the FTP server.

In this application, the Redpine device connects to Access Point and establishes FTP client connection with FTP server opened on remote peer. After successful connection, the application reads the data from "**read.txt**" file present in FTP server and writes back same data read from the file "**read.txt**" by replacing first few bytes with the string "**REDPINE FTP CLIENT DEMO**" to the FTP server by creating file "**write.txt**".

##### Sequence of Events

This Application explains user how to:

- Connect to Access Point
- Establish FTP connection with FTP server opened on remote peer
- Read data from "read.txt" file present in FTP server
- Write the data to "write.txt" file which is read from the "read.txt" file by replacing first few bytes with the string "**REDPINE FTP CLIENT DEMO**"

#### 8.10.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

##### WiSeMCU / WiSeConnect based Setup Requirements

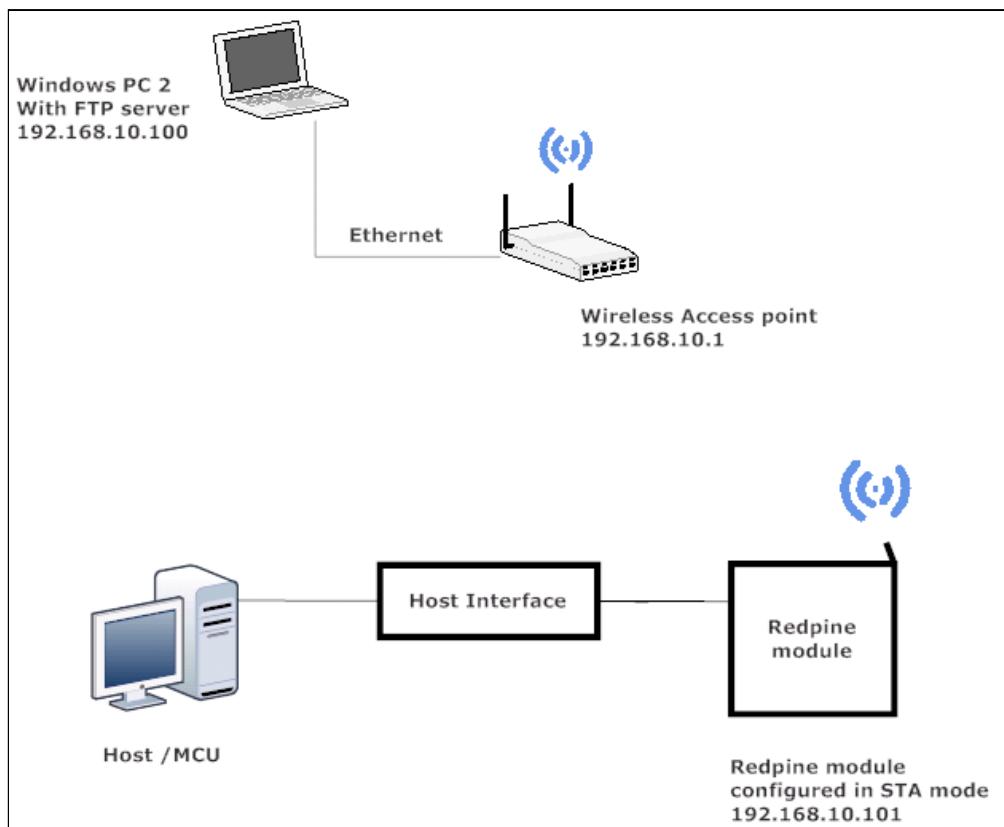
- Windows PC1 with Keil or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine Module
- Windows PC2 with FTP server installed in it

**Note1:**

FTP Server demo application can be downloaded from the given below link:  
<https://filezilla-project.org/download.php?type=server>

**Note2:**

FTP client functionality verified with FileZilla server version 0.9.51. So, recommended version of FileZilla software is "FileZilla\_Server-0\_9\_51.exe"

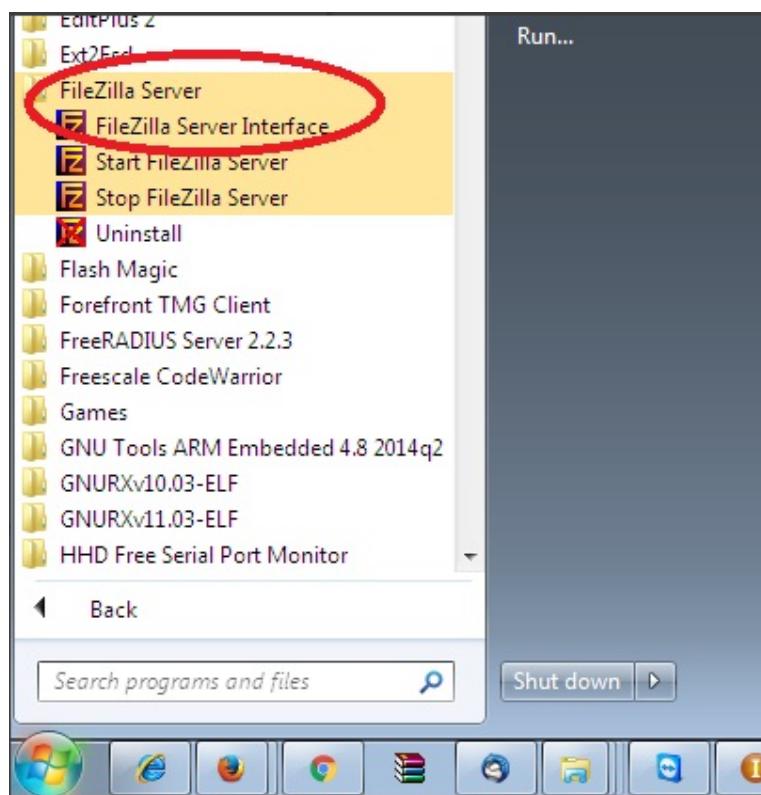


**Figure 2: FTP Client demo set up**

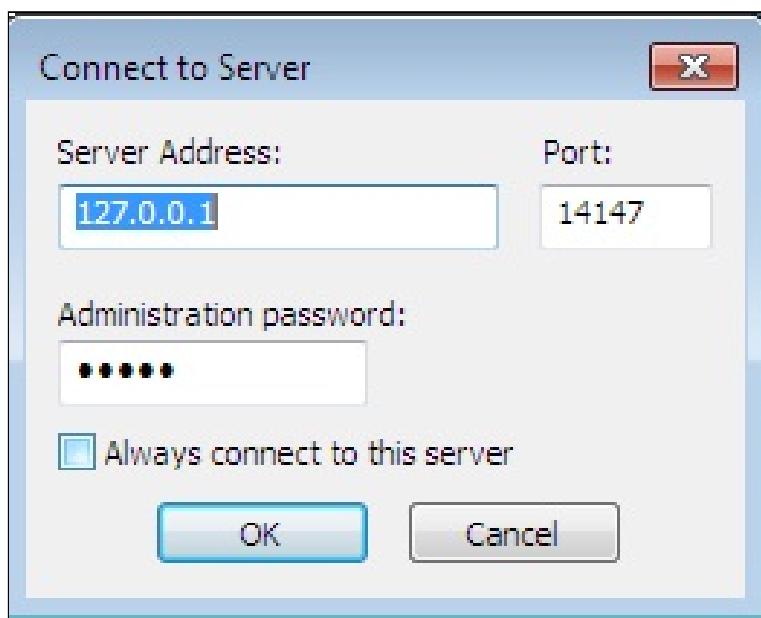
#### 8.10.4 Configuration and Execution of the Application

##### Installation of FTP server

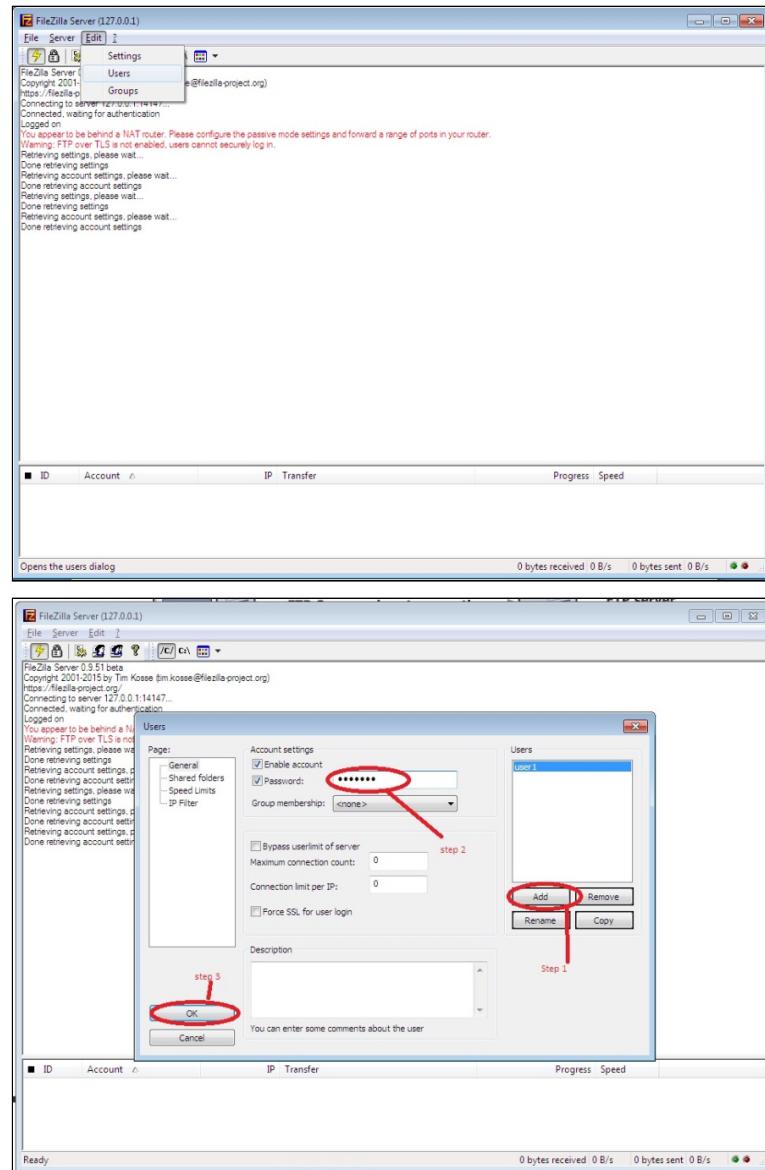
1. Download the FileZilla FTP server from below link - <https://filezilla-project.org/download.php?type=server>
2. Install downloaded FileZilla FTP server in Windows PC2 which is connected to AP through LAN.
3. Configure and run FTP server. Please refer the below images for configuring and running FileZilla FTP server.
4. After installation, open FileZilla Server interface.

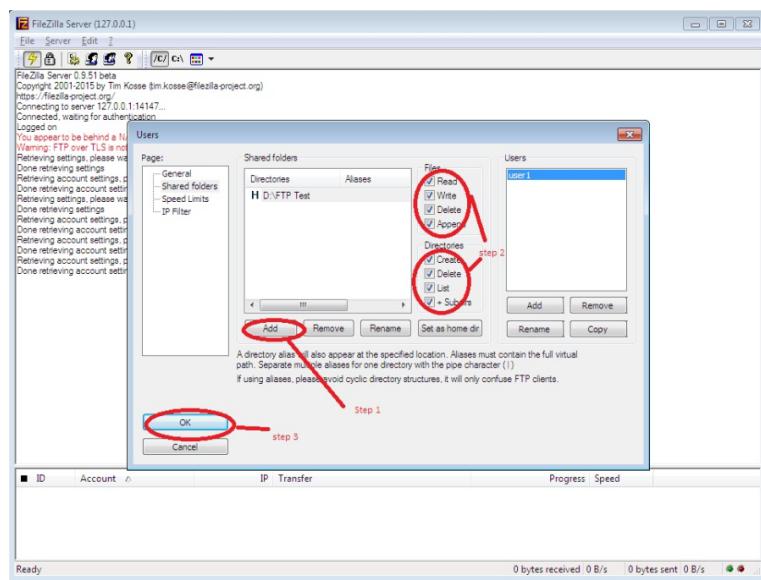


5. After opening FileZilla server interface, connect to server admin interface.

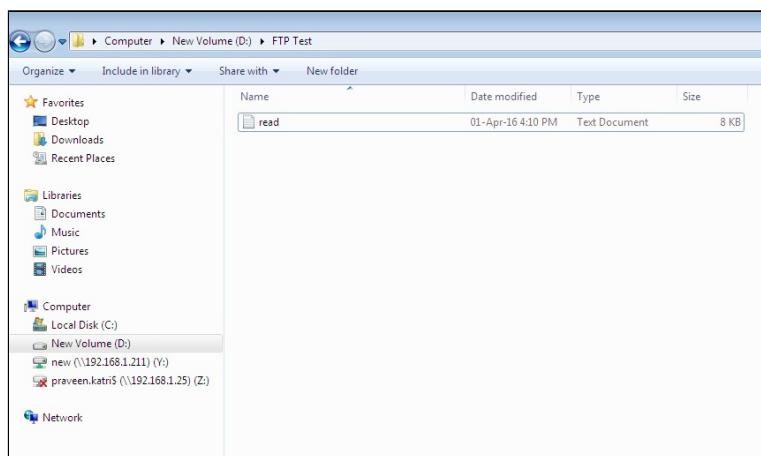


6. After connection with server, configure the user settings.





7. Place "read.txt" file in FTP directory (Ex: "D:\FTP Test\read.txt").



#### 8.10.5 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_ftp\_client.c** file and update/modify the following macros,  
**SSID** refers to the name of the Access point.

<code>#define SSID</code>	"<ap name>"
---------------------------	-------------

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode

**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

#define SECURITY\_TYPE

RSI\_OPEN

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

#define PSK

"<psk>"

**FTP\_SERVER\_PORT** port refers remote FTP server port number.  
By default FileZilla Server runs on port number 21.

#define FTP\_SERVER\_PORT

21

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with TCP server socket.  
IP address should be in long format and in little endian byte order.  
Example: To configure "192.168.0.100" as remote IP address, update the macro **SERVER\_IP\_ADDRESS** as **0x6400A8C0**.

#define SERVER\_IP\_ADDRESS

0x6400A8C0

FTP Server login username

#define FTP\_SERVER\_LOGIN\_USERNAME

"username"

FTP Server login password

#define FTP\_SERVER\_LOGIN\_PASSWORD

"password"

File to read which is on FTP server

Create a file on FTP server with the file name along with the path with data given below

#define FTP\_FILE\_TO\_READ  
(Or)  
#define FTP\_FILE\_TO\_READ  
Test\read.txt"

"read.txt"

"D:\FTP

**FILE\_CONTENT\_LENGTH** refers content length of the read file from FTP server (Ex: configure  
**FILE\_CONTENT\_LENGTH** >= Sizeof ("read.txt"))

```
#define FILE_CONTENT_LENGTH
```

```
10000
```

Filenameto createonFTP server and write the same content which is read from "read.txt"

```
#define FTP_FILE_TO_WRITE "write.txt"  
(Or)  
#define FTP_FILE_TO_WRITE "D:\FTP  
Test\write.txt"
```

To rename a file on FTP server

```
#define FTP_FILE_TO_RENAME "example.txt"
```

To set the directory on FTP server

```
#define FTP_DIRECTORY_SET "/work/FTP_EXAMPLE/  
FTP"
```

Tocreatedirectory on FTP server

```
#define FTP_DIRECTORY_CREATE "FTP"
```

To list the directories on FTP server

```
#define FTP_DIRECTORY_LIST "/work/FTP_EXAMPLE"
```

#### To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE 1
```

#### Note:

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode, it should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP          0X0A0AA8C0
```

IP address of the gateway should also be in longformatand in little endianbyteorder

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY           0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK            0x00FFFFFF
```

The following parameters need to be configureifOS is used.

WLAN task priority is given and this should be of low priority

```
#define RSI_WLAN_TASK_PRIORITY      1
```

Driver task priority is given and this should be of highest priority

```
#define RSI_DRIVER_TASK_PRIORITY    1
```

WLAN Task stack size is configured by this macro

```
#define RSI_WLAN_TASK_STACK_SIZE    500
```

Driver Task stack size is configured by this macro

```
#define RSI_DRIVER_TASK_STACK_SIZE   500
```

2. open **rsi\_wlan\_config.h** file and update/modify following macros :

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT| TCP_IP_FEAT_FTP_CLIENT)
#define RSI_CUSTOM_FEATURE_BIT_MAP 0
#define RSI_BAND RSI_BAND_2P4GHZ
```

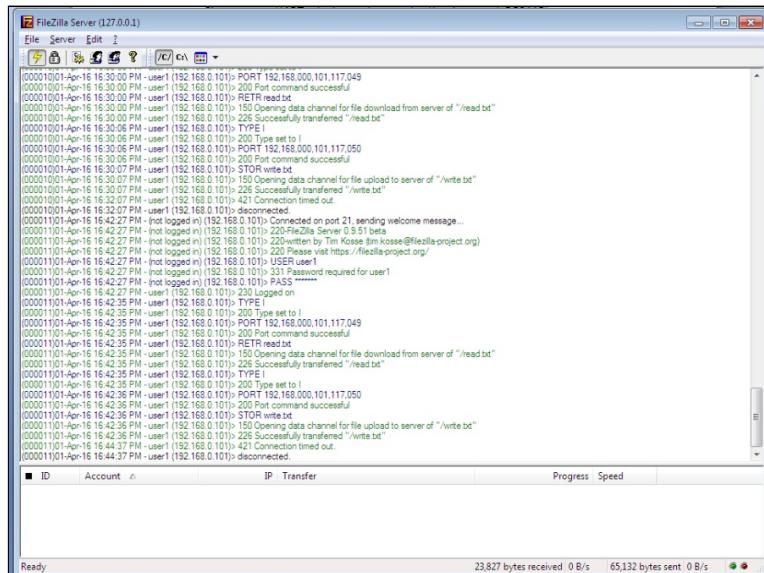
**Note:**

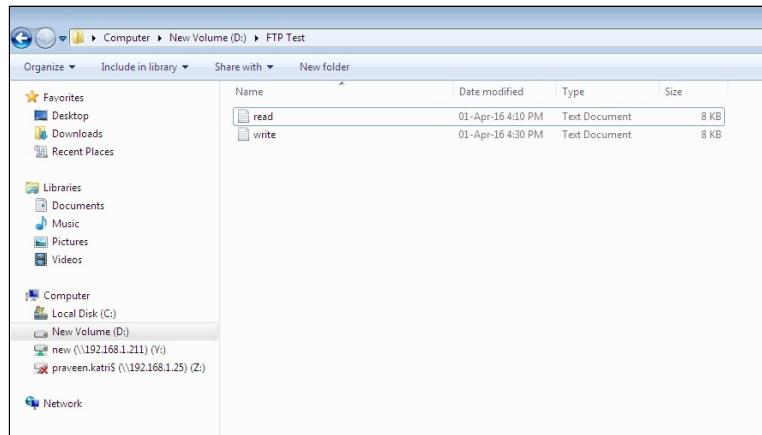
**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect the Redpine device in STA mode.
  2. Run FTP server on Windows PC2 and place "read.txt" file in the FTP directory.
  3. After the program gets executed, the Redpine device will be connected to same access point having the configuration same as that of in the application and get IP.
  4. After successful connection with the Access Point, the device connects to FTP server and reads the file content of given file ("read.txt") and creates a file name "**write.txt**" in FTP directory and writes the same content which is read from "read.txt" by replacing first few bytes with "**REDPINE FTP CLIENT DEMO**". After successful file write WiSeConnect device disconnects from FTP server.

Please refer the below images for message exchanges with FTP server and for read and write files,





## 8.11 HTTP/HTTPS Client Example

### 8.11.1 Protocol Overview

HTTP client is a client side HTTP transport library. HTTP client's purpose is to transmit and receive HTTP messages. HTTP client will not attempt to process content, execute javascript embedded in HTML pages, try to guess content type, or other functionality unrelated to the HTTP transport.

### 8.11.2 Example Overview

#### Overview

This application demonstrates how to create Redpine device as HTTP/HTTPPs client and do HTTP PUT, GET and POST operations with the HTTP/HTTPPs server opened on remote peer.

In this application, the device configures as Wi-Fi station and connects to Access point and do HTTP/HTTPPs PUT, GET and post operation with HTTP/HTTPPs server opened on remote peer.

#### Sequence of Events

This Application explains user how to:

- Load appropriate CA certificate to the Device to interact with HTTPS Server.
- Connect to Access Point
- Run HTTP/HTTPS Server Remote side.
- Request for HTTP/HTTPPs PUT, GET and POST.

### 8.11.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

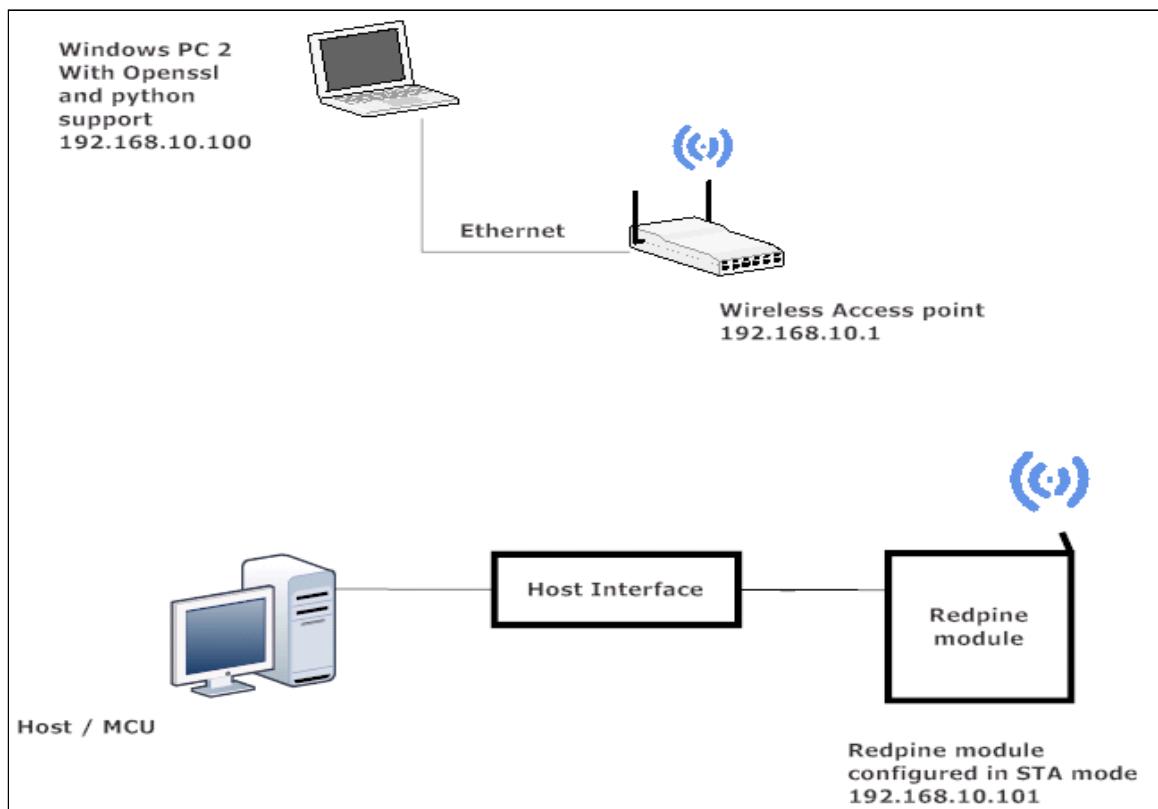
#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with Keil or IAR IDE in case of WiSeMCU

- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine Module
- Wi-Fi Access point
- Windows PC2 with openssl support and python installed

**Note:**

Installed python should support the following modules:  
Thread, HTTPServer, BaseHTTPRequestHandler, cgi, curdir, sep, sys



**Figure 1: Setup Diagram**

#### 8.11.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_http\_client\_app.c** file and update/modify following macros,  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels

```
#define CHANNEL_NO 0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK "<psk>"
```

### To Load certificate

```
#define LOAD_CERTIFICATE 1
```

If **LOAD\_CERTIFICATE** set to 1, application will load certificate which is included using `rsi_wlan_set_certificate` API.

By default, application loading "cacert.pem" certificate **LOAD\_CERTIFICATE** enable. In order to load different certificate, user has to follow the following steps:

- `rsi_wlan_set_certificate` API expects the certificate in the form of linear array. So, convert the pem certificate into linear array form using python script provided in the release package "**utilities/certificates/certificate\_script.py**"

Ex: If the certificate is wifi-user.pem .Give the command in the following way  
`python certificate_script.py ca-cert.pem`

Script will generate wifiuser.pem in which one linear array named cacert contains the certificate.

- After conversion of certificate, update `rsi_ssl_client.c` source file by including the certificate file and by providing the required parameters to `rsi_wlan_set_certificate` API.
- Once certificate loads into the device, it will write into the device flash. So, user need not load certificate for every boot up unless certificate change.

So define `LOAD_CERTIFICATE` as 0, if certificate is already present In the Device.

#### Note:

All the certificates are given in the release package

Path: `utilities/certificates`

### To configure IP address

`DHCP_MODE` refers whether IP address configured through DHCP or STATIC.

#define DHCP\_MODE

1

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

#define DEVICE\_IP

0X0A0AA8C0

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

#define GATEWAY

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK

0x00FFFFFF

To establish connection and request for HTTP PUT or HTTP GET or HTTP POST to the HTTP/HTTPS Server  
configure the below macros.

**FLAGS** refers to open normal HTTP client socket or HTTP client socket over SSL with IPv4 or IPv6  
Default configuration of application is normal HTTP client socket with IPv4.

#define

FLAGS0

(Or)

If user wants to open HTTP client socket over SSL with IPv4 then set **FLAGS** to 2 (**HTTPS\_SUPPORT**).

#define FLAGS HTTP

S\_SUPPORT

(Or)

If user wants to use HTTP client post large data then set **FLAGS** to 32 (**HTTP\_POST\_DATA**).

```
#define FLAGS HTTP_POST_DATA
```

(Or)

If user wants to open HTTP client with version 1.1 then set FLAGS to 64 (**HTTP\_V\_1\_1**).

```
#define FLAGS HTTP_V_1_1
```

(Or)

If user wants to open normal HTTP client socket with IPv6 then set FLAGS macro to 1 (**HTTPV6**).

```
#define FLAGS HTTPV6
```

(Or)

If user wants to open HTTP client socket over SSL with IPv6 then set FLAGS macro to 3 (**HHTTPV6 | HTTPS\_SUPPORT**)

```
#define FLAGS (HTTPV6 | HTTPS_SUPPORT)
```

HTTP\_PORT refers Port number of the remote HTTP/HTTPS server which is opened in Windows PC2.

```
#define HTTP_PORT 80
```

HTTP\_SERVER\_IP\_ADDRESS refers IP address of the HTTP/HTTPS server

**Note:**

HTTP\_SERVER\_IP\_ADDRESS should be as the below mentioned format as it is a string.

```
#define HTTP_SERVER_IP_ADDRESS "192.168.10.1"
```

HTTP\_URL refers HTTP resource name

```
#define HTTP_URL "/index.html"  
#define HTTP_HOSTNAME "192.168.10.1"
```

HTTP\_HOSTNAME refers host name

```
#define HTTP_HOSTNAME "192.168.10.1"
```

HTTP extended header

```
#define HTTP_EXTENDED_HEADER NULL
```

HTTP/HTTPS user name

```
#define USERNAME "admin"
```

Password for server

```
#define PASSWORD "admin"
```

HTTP/HTTPS post data

```
#define HTTP_DATA  
"employee_name=MR.REDDY&employee_id=RSXYZ123&designation=Engineer  
&company=REDPINE&location=Hyderabad"
```

Max HTTP PUT buffer length

```
#define MAX_HTTP_CLIENT_PUT_BUFFER_LENGTH 900
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 10000
```

Application buffer length

```
#define APP_BUFF_LEN 2000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP (TCP_IP_FEAT_DHCPV4_CLIENT |
TCP_IP_FEAT_HTTP_CLIENT)
```

If user wants to connect with HTTPS server set RSI\_TCP\_IP\_FEATURE\_BIT\_MAP as follows,

```
#define RSI_TCP_IP_FEATURE_BIT_MAP (TCP_IP_FEAT_DHCPV4_CLIENT |
TCP_IP_FEAT_SSL | TCP_IP_FEAT_HTTP_CLIENT)
#define RSI_CUSTOM_FEATURE_BIT_MAP 0
#define RSI_BAND RSI_BAND_2P4GHZ
```

**Note:**

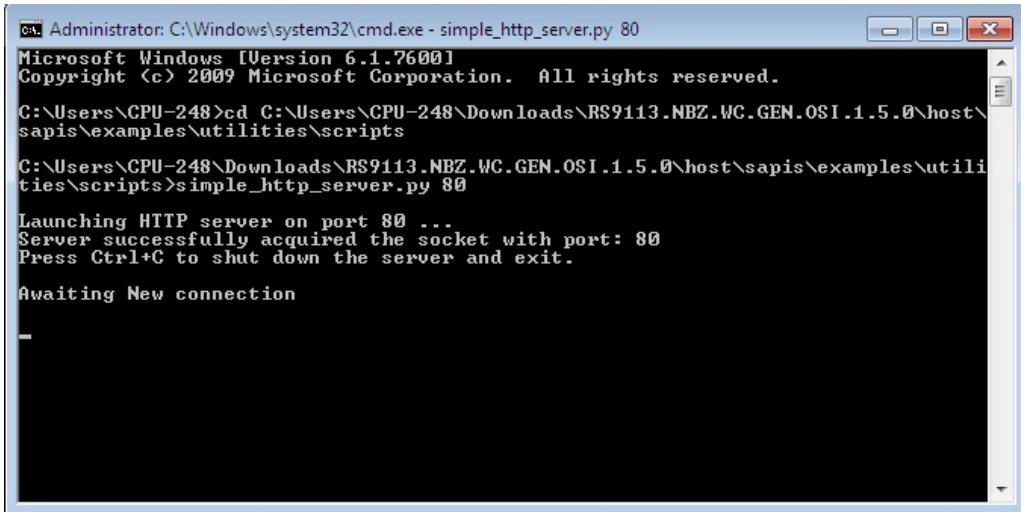
**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. In Windows PC2, install python and run HTTP server or HTTPS server,

In release package python scripts are provided to open HTTP server and HTTPS server in the following path:  
**utilities/scripts**

Run **simple\_http\_server.py** by giving port number 80 as argument to open HTTP server.



```
Administrator: C:\Windows\system32\cmd.exe - simple_http_server.py 80
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\CPU-248>cd C:\Users\CPU-248\Downloads\RS9113.NBZ.WC.GEN.OSI.1.5.0\host\sapis\examples\utilities\scripts

C:\Users\CPU-248\Downloads\RS9113.NBZ.WC.GEN.OSI.1.5.0\host\sapis\examples\utilities\scripts>simple_http_server.py 80

Launching HTTP server on port 80 ...
Server successfully acquired the socket with port: 80
Press Ctrl+C to shut down the server and exit.

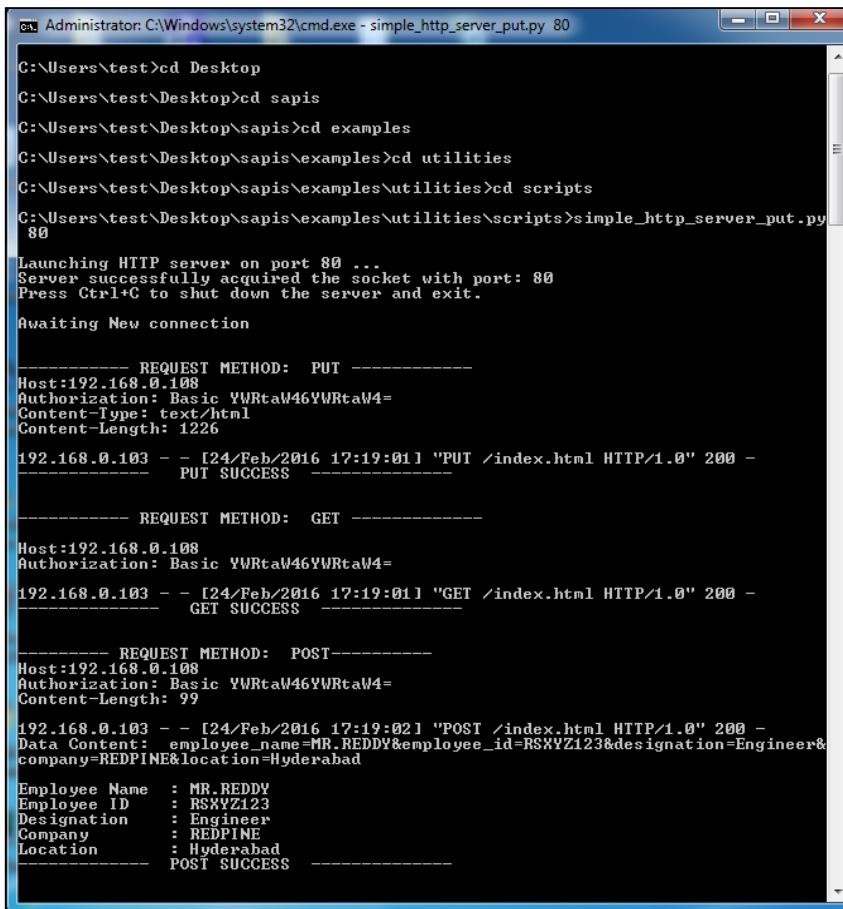
Awaiting New connection
```

**Note:**

Release package includes only HTTP server script. If user wants to test HTTPS client, then user has to run HTTPS server which supports HTTPS PUT, GET and POST.

2. After the program gets executed, the device connects to AP and get IP.
3. After successful connection with Access Point, the redpine device request for HTTP PUT to PUT/Create the file on to the server, which is given in index.txt file and wait until put file complete.
4. Remote web server accepts a PUT request and writes the received data to a file. User can find the created new file "index.html" on Windows PC2 in the following path, **utilities/scripts**
5. After successful creation of file using HTTP PUT, Redpine device request for the file "index.html" using HTTP GET method and wait until complete response receive from Server.
6. After receiving complete response for the given HTTP GET, the device post the given data in HTTP\_DATA macro to HTTP server using HTTP POST meth

User can see the log messages at HTTP server. Please find the below image for success responses for HTTP PUT, HTTP GET and HTTP POST.



```

Administrator: C:\Windows\system32\cmd.exe - simple_http_server_put.py 80
C:\Users\test>cd Desktop
C:\Users\test\Desktop>cd sapis
C:\Users\test\Desktop\sapis>cd examples
C:\Users\test\Desktop\sapis\examples>cd utilities
C:\Users\test\Desktop\sapis\examples\utilities>cd scripts
C:\Users\test\Desktop\sapis\examples\utilities\scripts>simple_http_server_put.py
80
Launching HTTP server on port 80 ...
Server successfully acquired the socket with port: 80
Press Ctrl+C to shut down the server and exit.

Awaiting New connection

----- REQUEST METHOD: PUT -----
Host:192.168.0.108
Authorization: Basic YWRtaW46YWRtaW4=
Content-Type: text/html
Content-Length: 1226
192.168.0.103 - - [24/Feb/2016 17:19:01] "PUT /index.html HTTP/1.0" 200 -
PUT SUCCESS

----- REQUEST METHOD: GET -----
Host:192.168.0.108
Authorization: Basic YWRtaW46YWRtaW4=
192.168.0.103 - - [24/Feb/2016 17:19:01] "GET /index.html HTTP/1.0" 200 -
GET SUCCESS

----- REQUEST METHOD: POST -----
Host:192.168.0.108
Authorization: Basic YWRtaW46YWRtaW4=
Content-Length: 99
192.168.0.103 - - [24/Feb/2016 17:19:02] "POST /index.html HTTP/1.0" 200 -
Data Content: employee_name=MR.REDDY&employee_id=RSXYZ123&designation=Engineer&
company=REDPINE&location=Hyderabad
Employee Name : MR.REDDY
Employee ID : RSXYZ123
Designation : Engineer
Company : REDPINE
Location : Hyderabad
POST SUCCESS
  
```

## 8.12 HTTP/HTTPS Client Post Data Example

### 8.12.1 Protocol Overview

HTTP client is a client side HTTP transport library. HTTP client's purpose is to transmit and receive HTTP messages. HTTP client will not attempt to process content, execute javascript embedded in HTML pages, try to guess content type, or other functionality unrelated to the HTTP transport.

## 8.12.2 Example Overview

### Overview

This application demonstrates how to create Redpine device as HTTP/HTTPPs client and do GET and POST operations with the HTTP/HTTPPs server opened on remote peer.  
In this application, the device configures as Wi-Fi station and connects to Access point and do HTTP/HTTPPs Post and Get operation with HTTP/HTTPPs server opened on remote peer.

### Sequence of Events

This Application explains user how to:

- Load appropriate CA certificate to the Device to interact with HTTPS Server.
- Connect to Access Point
- Run HTTP/HTTPS Server Remote side.
- Request for HTTP/HTTPPs POST and GET.

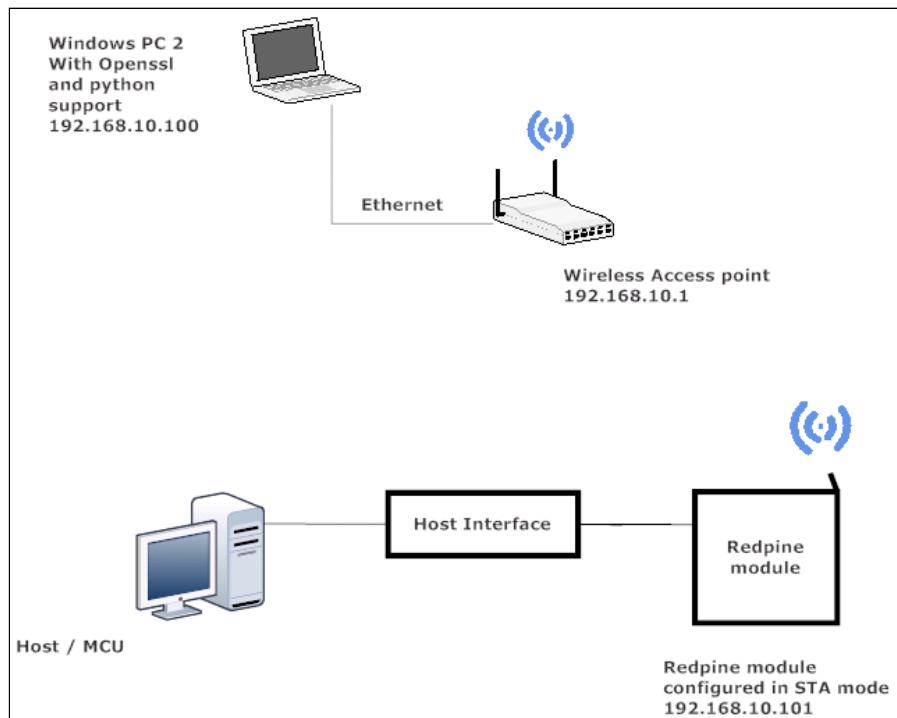
## 8.12.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization. The following sections are applicable to WiSeConnect parts only.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with Keil or IAR IDE
- Redpine Module
- WiFi Access point
- Windows PC2 with openssl support and python installed

**Note:** Installed python should support the following modules:  
Thread, HTTPServer, BaseHTTPRequestHandler, cgi, curdir, sep, sys



**Figure 1: Setup Diagrams**

#### 8.12.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_http\_client\_app.c** file and update/modify following macros,  
**SSID** refers to the name of the Access point.

```
#define SSID                                "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels

```
#define CHANNEL_NO                            0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is

**RSI\_OPEN** - For OPEN security mode

**RSI\_WPA** - For WPA security mode

**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE                         RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK           "<psk>"
```

To abort the application after the mentioned packet count

```
#define RSI_HTTP_MAX_PKT_COUNT_ABORT      10
```

To enable RSI\_ENABLE\_HTTP\_ABORT functionality, set the define ABORT to 1.

```
#define RSI_ENABLE_HTTP_ABORT              0
```

#### To Load certificate

```
#define LOAD_CERTIFICATE                1
```

If **LOAD\_CERTIFICATE** set to 1, application will load certificate which is included using `rsi_wlan_set_certificate` API.

By default, application loading "cacert.pem" certificate **LOAD\_CERTIFICATE** enable. In order to load different certificate, user has to follow the following steps:

- `rsi_wlan_set_certificate` API expects the certificate in the form of linear array. So, convert the pem certificate into linear array form using python script provided in the release package "**sapis/examples/utilities/certificates/certificate\_script.py**"

Ex: If the certificate is wifi-user.pem .Give the command in the following way  
`python certificate_script.py ca-cert.pem`

Script will generate wifiuser.pem in which one linear array named cacert contains the certificate.

- After conversion of certificate, update `rsi_ssl_client.c` source file by including the certificate file and by providing the required parameters to `rsi_wlan_set_certificate` API.
- Once certificate loads into the device, it will write into the device flash. So, user need not load certificate for every boot up unless certificate change.

So define **LOAD\_CERTIFICATE** as 0, if certificate is already present In the Device.

#### Note:

All the certificates are given in the release package Path: `sapis/examples/utilities/certificates`

#### To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE                  1
```

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set DHCP\_MODE macro to "0" and configure following DEVICE\_IP, GATEWAY and NETMASK macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

#define DEVICE_IP	0X0A0AA8C0
-------------------	------------

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

#define GATEWAY	0X010AA8C0
-----------------	------------

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK	0X00FFFFFF
-----------------	------------

To establish connection and request for HTTP PUT or HTTP GET or HTTP POST to the HTTP/HTTPS Server  
configure the below macros.  
**DEVICE\_PORT** refers internal socket port number.

#define DEVICE_PORT	5001
---------------------	------

**FLAGS** refers to open normal HTTP client socket or HTTP client socket over SSL with IPv4 or IPv6  
Default configuration of application is normal HTTP client socket with IPv4.

**#define FLAGSO**

(Or)

If user wants to open HTTP client socket over SSL with IPv4 then set FLAGS to 2 (**HTTPS\_SUPPORT**).

**#define FLAGSHHTPS\_SUPPORT**

(Or)

If user wants to open normal HTTP client socket with IPv6 then set FLAGS macro to 1 (**HTTPV6**).

#define FLAGSHHTPV	6
--------------------	---

(Or)

If user wants to open HTTP client socket over SSL with IPv6 then set FLAGS macro to 3 (**HHTPV6** | **HTTPS\_SUPPORT**)

```
#define FLAGS (HTTPV6 | HTTPS_SUPPORT)
```

If user wants to use HTTP client to post large HTTP data set FLAGS macro to 32

```
#define FLAGS (HTTP_DAT_DATA)
```

If user wants to use HTTP client version 1.1 set FLAGS macro to 64

```
#define FLAGS (HTTP_V_1_1)
```

**HTTP\_PORT** refers Port number of the remote HTTP/HTTPS server which is opened in Windows PC2.

```
#define HTTP_PORT 80
```

**HTTP\_SERVER\_IP\_ADDRESS** refers IP address of the HTTP/HTTPS server

**Note:**

HTTP\_SERVER\_IP\_ADDRESS should be as the below mentioned format as it is a string.

```
#define HTTP_SERVER_IP_ADDRESS "192.168.10.1"
```

**HTTP\_URL** refers HTTP resource name

```
#define HTTP_URL "/index.html"  
#define HTTP_HOSTNAME "192.168.10.1"
```

**HTTP\_HOSTNAME** refers host name

```
#define HTTP_HOSTNAME "192.168.10.1"
```

HTTP extended header

```
#define HTTP_EXTENDED_HEADER NULL
```

HTTP/HTTPS user name

```
#define USERNAME "admin"
```

Password for server

```
#define PASSWORD "admin"
```

Max HTTP POST DATA buffer length

```
#define MAX_HTTP_CLIENT_POST_DATA_BUFFER_LENGTH 900
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 8000
```

Application buffer length

```
#define APP_BUFF_LEN 2000
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP FEAT_SECURITY_OPEN  
#define RSI_TCP_IP_BYPASS RSI_DISABLE
```

If user wants to connect with HTTP server set **RSI\_TCP\_IP\_FEATURE\_BIT\_MAP** as follows,

```
#define RSI_TCP_IP_FEATURE_BIT_MAP (TCP_IP_FEAT_DHCPV4_CLIENT  
| TCP_IP_FEAT_HTTP_CLIENT)
```

If user wants to connect with HTTPS server set **RSI\_TCP\_IP\_FEATURE\_BIT\_MAP** as follows,

```
#define RSI_TCP_IP_FEATURE_BIT_MAP (TCP_IP_FEAT_DHCPV4_CLIENT |  
TCP_IP_FEAT_SSL | TCP_IP_FEAT_HTTP_CLIENT)  
#define RSI_CUSTOM_FEATURE_BIT_MAP 0  
#define RSI_BAND RSI_BAND_2P4GHZ
```

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

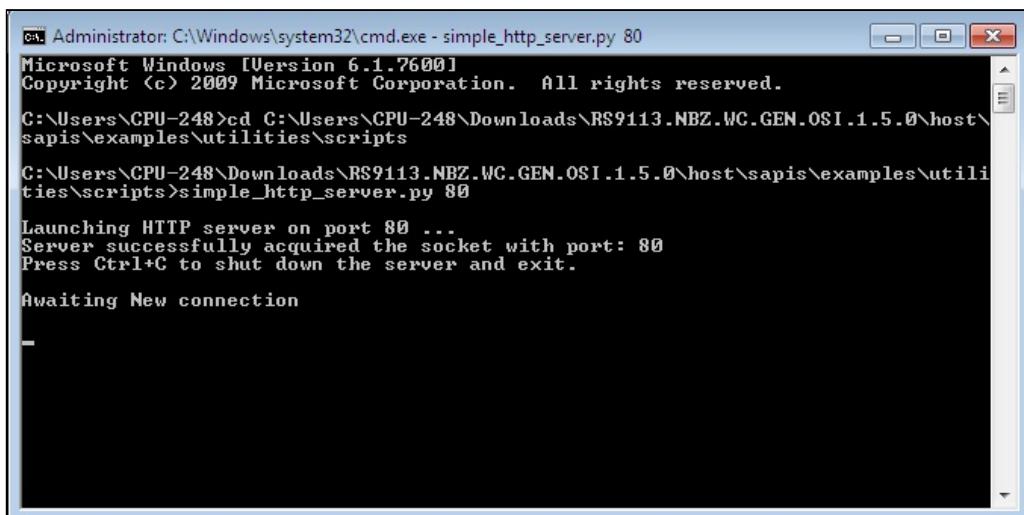
## Executing the Application

1. In Windows PC2, install python and run HTTP server or HTTPS server.

In release package python scripts are provided to open HTTP server and HTTPS server in the following path:

**sapis/examples/utilities/scripts**

Run **simple\_http\_server\_post\_data.py** by giving port number 80 as argument to open HTTP server.



```
Administrator: C:\Windows\system32\cmd.exe - simple_http_server.py 80
Microsoft Windows [Version 6.1.7600]
Copyright <c> 2009 Microsoft Corporation. All rights reserved.

C:\Users\CPU-248>cd C:\Users\CPU-248\Downloads\RS9113.NBZ.WC.GEN.OSI.1.5.0\host\sapis\examples\utilities\scripts

C:\Users\CPU-248\Downloads\RS9113.NBZ.WC.GEN.OSI.1.5.0\host\sapis\examples\utili
ties\scripts>simple_http_server.py 80

Launching HTTP server on port 80 ...
Server successfully acquired the socket with port: 80
Press Ctrl+C to shut down the server and exit.

Awaiting New connection
```

**Note:**

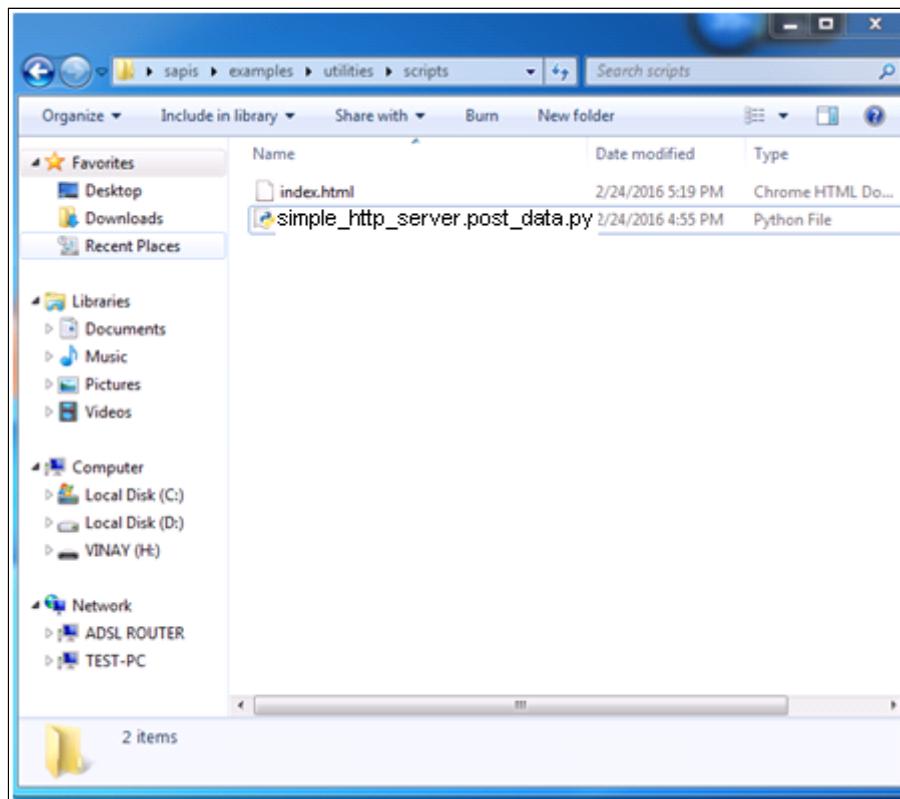
Release package includes only HTTP server script. If user wants to test HTTPS client, then user has to run HTTPS server which supports HTTPS PUT, GET and POST.

2. After the program gets executed, the redpine device connects to AP and get IP.

3. After successful connection with Access Point, the redpine device request for HTTP POST to send the user given file to the server, which is given in index.txt file and wait until post file complete.

Remote web server accepts a POST request and give response. User can find the created new file "index.html" on Windows PC2 in the following path:

**sapis/examples/utilities/scripts**



4. After successful sending offileusing HTTP POST, the device request for the file "index.html" using HTTP GET method and wait until complete responsereceivefrom Server.

## 8.13 Instant Background Scan Example

### 8.13.1 Example Overview

#### Overview

This application demonstrates how to enable Back ground scan and get results of available access points after successful connection with the Access Point in station mode.

#### Sequence of Events

This Application explains user how to:

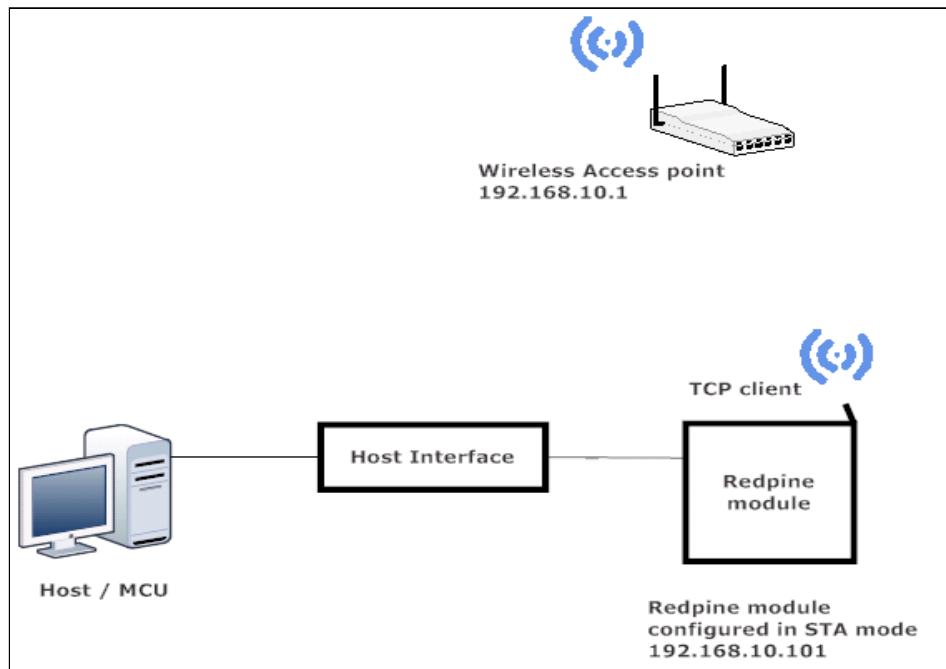
- Connect the Device to an Access point and get IP address through DHCP
- Initiate Instant Back ground scan.

### 8.13.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with Keil or IAR IDE in case of WiSeMCU
- Redpine Module
- Wi-Fi Access point



**Figure 1: Setup Diagram**

#### 8.13.3 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_instant\_bgscan.c** file and update/modify following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels.

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode

**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

#define SECURITY\_TYPE

RSI\_OPEN

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

#define PSK

"<psk>"

**To configure IP address**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

#define DHCP\_MODE

1

**Note:** If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

#define DEVICE\_IP

0X0A0AA8C0

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

#define GATEWAY

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK

0x00FFFFFF

Application memory length which is required by the driver

#define GLOBAL\_BUFF\_LEN

8000

**APP\_BUFF\_LEN** refers buffer length to read back ground scan results.

#define APP\_BUFF\_LEN

200

2. Open **rsi\_wlan\_config.h** file and update/modify following macros :

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	
FEAT_SECURITY_OPEN	
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	0
#define RSI_BAND	RSI_BAND_2P4GHZ
#define RSI_BG_SCAN_SUPPORT	RSI_ENABLE
#define RSI_BG_SCAN_ENABLE	RSI_ENABLE
#define RSI_INSTANT_BG	RSI_ENABLE
#define RSI_MULTI_PROBE	RSI_ENABLE

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. After program gets executed, the device would scan and connect to the access point and get IP.
2. After successful connection, the application initiates Instant Back ground scan. The Redpine device scans for Access Points and gives scanned Access Points information in "**rsi\_wlan\_bgscan\_profile**" API response. User can parse the response buffer "**bgscan\_results**" for Access Points details.

## 8.14 MFP Client

### 8.14.1 Example Overview

The 802.11w protocol applies only to a set of robust management frames that are protected by the Protected Management Frames ( PMF) service. These include Disassociation, De-authentication, and Robust Action frames.

When 802.11w is implemented in the wireless medium, the following occur:

- Client protection is added by the AP adding cryptographic protection to de-authentication and dissociation frames preventing them from being spoofed in a DOS attack.
- Infrastructure protection is added by adding a Security Association (SA) tear down protection mechanism consisting of an Association Comeback Time.

## 8.14.2 Application Overview

### Overview

The 802.11W client application demonstrates how to connect with MFP enable AP and how to open and use a standard TCP client socket and sends data to TCP server socket.

### Sequence of Events

This Application explains user how to:

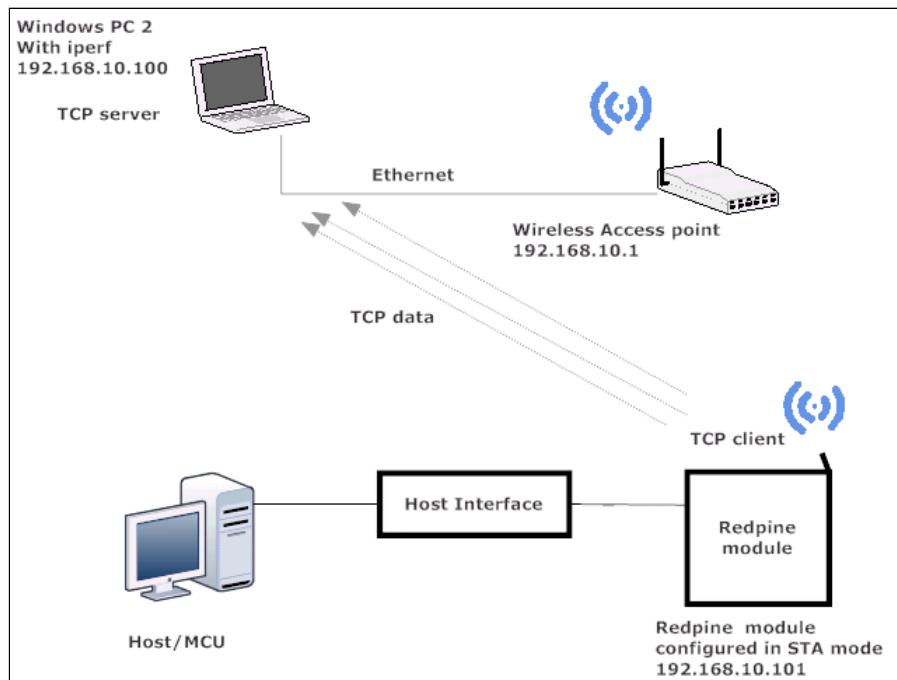
- Connect the Device to an Access point which support MFP and get IP address through DHCP
- Open TCP Server socket at Access point using iperf application.
- Open TCP client socket in device and establish TCP connection with TCP server opened in remote peer.
- Send data from Redpine device to remote peer using opened TCP socket.

### 8.14.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine Device
- Wi-Fi Access point
- Windows PC2
- TCP server application running in Windows PC2 (This application uses iperf application to open TCP server socket)



**Figure 1: Setup Diagram**

#### 8.14.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_mfp\_client.c** file and update/modify following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels.

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

**Note:**

Security must be WPA2 only

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK "<psk>"
```

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT 5001
```

**SERVER\_PORT** port refers remote TCP server port number which is opened in windows PC2.

```
#define SERVER_PORT 5001
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with TCP server socket.  
IP address should be in long format and in little endian byte order.

Example: To configure "192.168.10.100" as IP address, update the macro **DEVICE\_IP** as **0x640AA8C0**.

```
#define SERVER_IP_ADDRESS 0x640AA8C0
```

**NUMBER\_OF\_PACKETS** refers how many packets to send from device to TCP server

```
#define NUMBER_OF_PACKETS 1000
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 8000
```

**To configure IP address**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE 1
```

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set DHCP\_MODE macro to "0" and configure following DEVICE\_IP, GATEWAY and NETMASK macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

#define DEVICE_IP	0X0A0AA8C0
-------------------	------------

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

#define GATEWAY	0X010AA8C0
-----------------	------------

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK	0X00FFFFFF
-----------------	------------

2. Open **rsi\_wlan\_config.h** file and update/modify following macros :

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	
FEAT_CUSTOM_FEAT_EXTENTION_VALID	
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP	EXT_FEAT_IEEE_80211W

To enable **MFP feature**, set the following define

#define RSI_JOIN_FEATURE_BIT_MAP	MFP_CAPABLE_REQUIRED
----------------------------------	----------------------

**MFP-DISABLE** - Disable

**MFP\_CAPABLE\_ONLY**- Enable MFP Capable only

**MFP\_CAPABLE\_REQUIRED**- Enable MFP Capable and Required

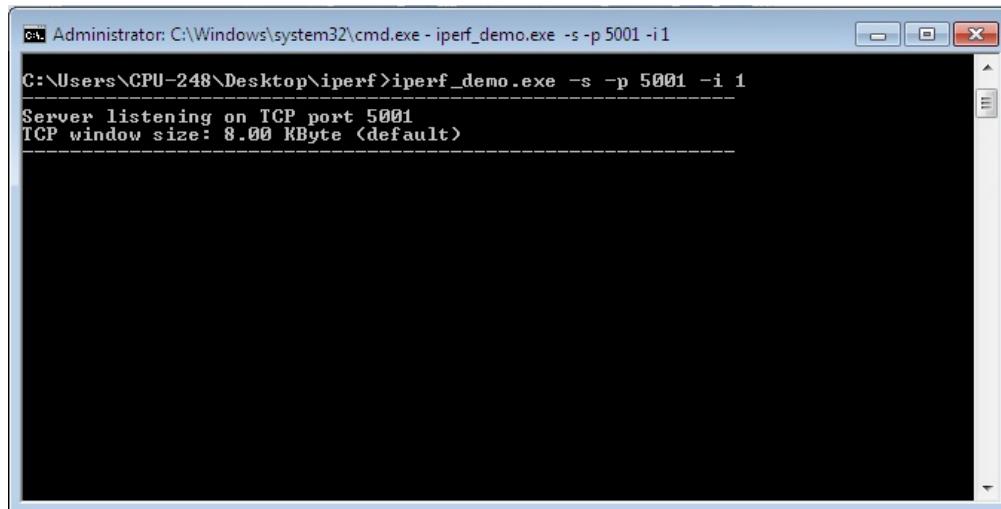
**Note:**

rsi\_wlan\_config.h file is already set with desired configuration in respective example folders user need not change for each example.

**Executing the Application**

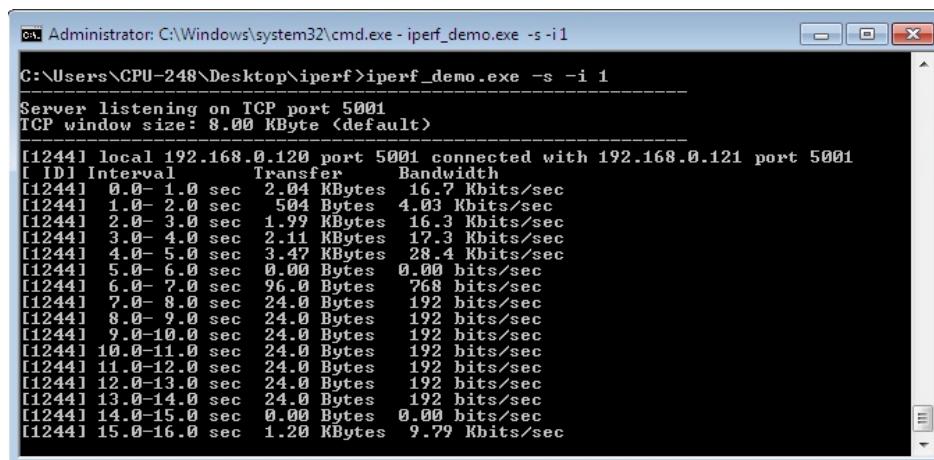
1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect the Redpine device in STA mode.
2. Open TCP server application using iperfapplication in Windows PC2 which is connected to Accesspoint through LAN.

**iperf\_demo.exe -s -p <SERVER\_PORT> -i 1**



```
Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -p 5001 -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -p 5001 -i 1
-----
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
```

3. After the program gets executed, the Redpine device would scan and connect to the access point and get IP.
4. After successful connection, Redpine device STA connects to TCP server socket opened on Windows PC2 using TCP client socket and sends configured **NUMBER\_OF\_PACKETS** to remote TCP server. The below image depicts the reception of TCP data on TCP server,



```
Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -i 1
-----
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)

[12441] local 192.168.0.120 port 5001 connected with 192.168.0.121 port 5001
[ ID] Interval Transfer Bandwidth
[12441] 0.0- 1.0 sec 2.04 KBytes 16.7 Kbits/sec
[12441] 1.0- 2.0 sec 504 Bytes 4.03 Kbits/sec
[12441] 2.0- 3.0 sec 1.99 KBytes 16.3 Kbits/sec
[12441] 3.0- 4.0 sec 2.11 KBytes 17.3 Kbits/sec
[12441] 4.0- 5.0 sec 3.47 KBytes 28.4 Kbits/sec
[12441] 5.0- 6.0 sec 0.00 Bytes 0.00 bits/sec
[12441] 6.0- 7.0 sec 96.0 Bytes 768 bits/sec
[12441] 7.0- 8.0 sec 24.0 Bytes 192 bits/sec
[12441] 8.0- 9.0 sec 24.0 Bytes 192 bits/sec
[12441] 9.0-10.0 sec 24.0 Bytes 192 bits/sec
[12441] 10.0-11.0 sec 24.0 Bytes 192 bits/sec
[12441] 11.0-12.0 sec 24.0 Bytes 192 bits/sec
[12441] 12.0-13.0 sec 24.0 Bytes 192 bits/sec
[12441] 13.0-14.0 sec 24.0 Bytes 192 bits/sec
[12441] 14.0-15.0 sec 0.00 Bytes 0.00 bits/sec
[12441] 15.0-16.0 sec 1.20 KBytes 9.79 Kbits/sec
```

## 8.15 MQTT Client Example

### 8.15.1 Protocol Overview

MQTT is a publish-subscribe based "light weight" messaging protocol for using on top of the TCP/IP protocol. The MQTT connection itself is always between one client and the broker, no client is connected to another client directly.

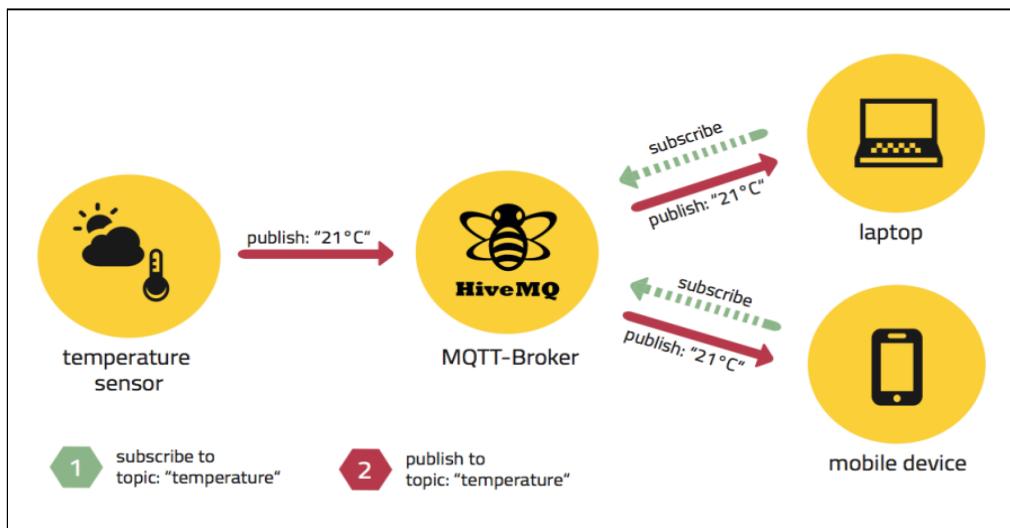
#### MQTT client

A MQTT client is any device from a micro controller to a full fledged server, that has a MQTT library running and is connecting to an MQTT broker over any kind of network. MQTT Clients can share the information on a particular topic using MQTT protocol. MQTT clients connect to the MQTT broker using TCP connection and can subscribe and publish on any desired topic. The other clients which are subscribed for that topic will receive the published messages.

#### MQTT Broker

The publish-subscribe messaging pattern requires a message broker. The broker is primarily responsible for receiving all messages, filtering them, deciding like who is interested in it and then sending the message to all subscribed clients.

It also holds the session of all persisted clients including subscriptions and missed messages. Another responsibility of the broker is the authentication and authorization of clients. A simple demonstration of subscribing and publishing of temperature is shown below:



**Figure 1: Demonstration of MQTT protocol**

### 8.15.2 Example Overview

#### Overview

This application demonstrates how to configure Redpine device as MQTT client and how to establish connection with MQTT broker and how to subscribe, publish and receive the MQTT messages from MQTT broker.

In this application, Redpine device configured as WiFi station and connects to Access Point. After successful WiFi connection, application connects to MQTT broker and subscribes to the topic "**REDPINE**" and publishes a message "**"THIS IS MQTT CLIENT DEMO FROM REDPINE"**" on that subscribed topic. And application waits to receive the data published on subscribed topic by other clients.

## Sequence of Events

This Application explains user how to:

- Connect to Access Point
- Establish MQTT client connection with MQTT broker
- Subscribe the topic "**REDPINE**"
- Publish message "**THIS IS MQTT CLIENT DEMO FROM REDPINE**" on the subscribed topic "**REDPINE**"
- Receive data published by other clients on the same subscribed topic "**REDPINE**".

### 8.15.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with Keil or IAR IDE in case of WiSeMCU
- Redpine Module
- Windows PC2 with with MQTT broker installed in it
- Windows PC3 with with MQTT client utility installed in it

#### Note:

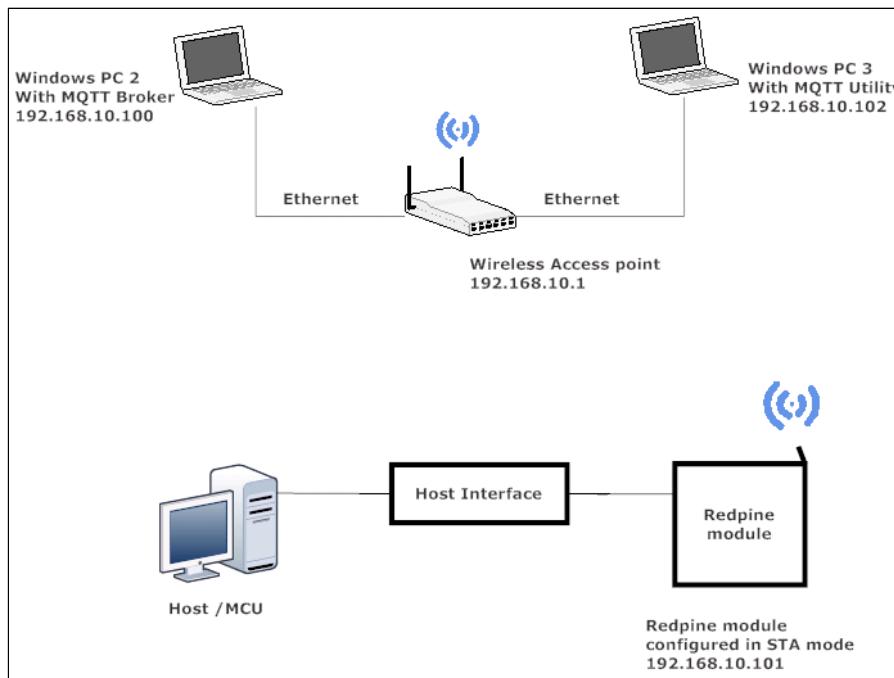
MQTT broker for different OS platforms can be downloaded from the below link

<http://mosquitto.org/download/>

**Ex:** Install "mosquitto-1.4.8-install-win32.exe"

MQTT Utility which has to be installed in Windows PC 3 can be downloaded from the below given link

<https://www.eclipse.org/downloads/download.php?file=/paho/1.0/org.eclipse.paho.mqtt.utility-1.0.0.jar>



**Figure 2: MQTT Client demo set up**

#### 8.15.4 Configuration and Execution of the Application

##### Configuring the Application

1. If user want to use asynchronous MQTT, Open **rsi\_mqtt\_client.c** file and update/modify following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE      RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK             "<psk>"
```

**CLIENT\_PORT** port refers device MQTT client port number

#define CLIENT_PORT	5001
---------------------	------

**SERVER\_PORT** port refers remote MQTT broker/server port number

#define SERVER_PORT	1883
---------------------	------

**SERVER\_IP\_ADDRESS** refers remote peer IP address (Windows PC2) to connect with MQTT broker/server socket.

IP address should be in long format and in little endian byte order.

Example: To configure "192.168.0.100" as remote IP address, update the macro **SERVER\_IP\_ADDRESS** as **0x6400A8C0**.

#define SERVER_IP_ADDRESS	0x6400A8C0
---------------------------	------------

MQTT client keep alive period

#define RSI_KEEP_ALIVE_PERIOD	100
-------------------------------	-----

Memory to initialize MQTT client Info structure

#define MQTT_CLIENT_INIT_BUFF_LEN	3500
-----------------------------------	------

Global buffer or memory which is used for MQTT client initialization. This buffer is used for the MQTT client information storage.

**uint8\_t mqqt\_client\_buffer[MQTT\_CLIENT\_INIT\_BUFF\_LEN];**

**QoS** indicates the level of assurance for delivery of an Application Message.

QoS levels are:

0 - At most once delivery

1 - At least once delivery

2 - Exactly once delivery

#define QOS	0
-------------	---

**RSI\_MQTT\_TOPIC** refers to which topic WiSeConnect MQTT client is supposed to subscribe.

#define RSI_MQTT_TOPIC	"REDPINE"
------------------------	-----------

MQTT Message to publish on the topic subscribed

**uint8\_t publish\_message[] = "THIS IS MQTT CLIENT DEMO  
FROM REDPINE"**

MQTT Client ID with which MQTT client connects to MQTT broker/server

**uint8\_t clientID[] = "MQTTCLIENT"**

User name for login credentials

**int8\_t username[] = "username"**

Password for login credentials

**int8\_t password[] = "password"**

**NUMEBR\_OF\_PACKETS** refers how many packets to receive from TCP client

#define NUMBER\_OF\_PACKETS

<no of packets>

Application memory length which is required by the driver

#define GLOBAL\_BUFF\_LEN

8000

#### To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

#define DHCP\_MODE

1

#### Note:

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through **STATIC** then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

#define DEVICE\_IP

0X0A0AA8C0

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

#define GATEWAY

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order  
Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK          0x00FFFFFF
```

The following parameters are configured if OS is used.  
WLAN task priority is given and this should be of low priority

```
#define RSI_WLAN_TASK_PRIORITY      1
```

Driver task priority is given and this should be of highest priority

```
#define RSI_DRIVER_TASK_PRIORITY    1
```

WLAN Task stack size is configured by this macro

```
#define RSI_WLAN_TASK_STACK_SIZE    500
```

Driver Task stack size is configured by this macro

```
#define RSI_DRIVER_TASK_STACK_SIZE  500
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	
FEAT_SECURITY_OPEN	
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	0
#define RSI_BAND	RSI_BAND_2P4GHZ

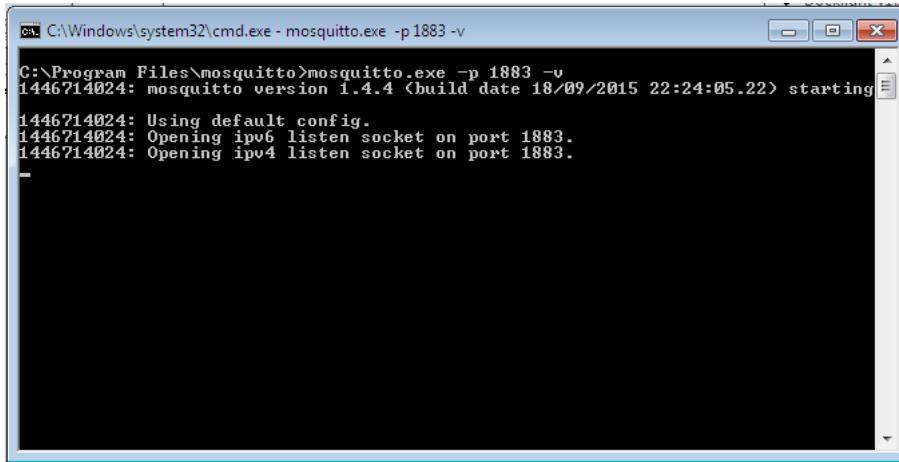
**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

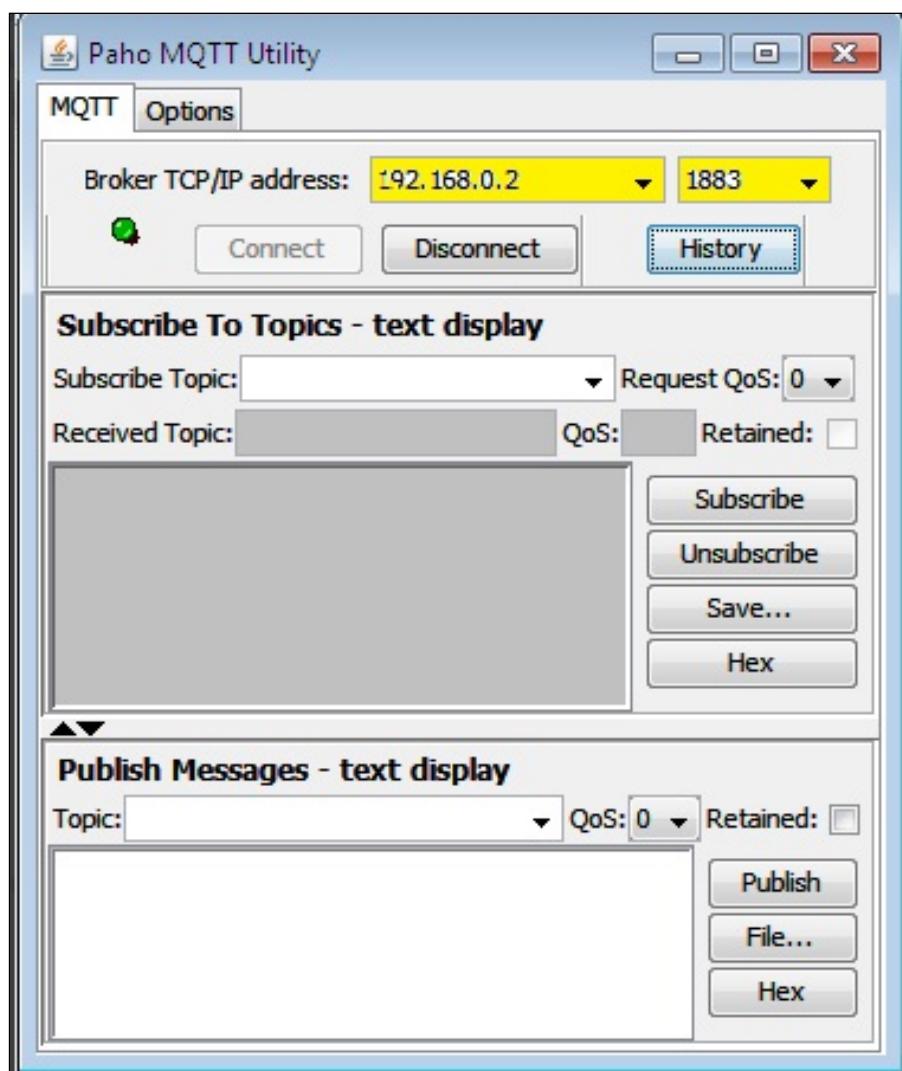
1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. Install MQTT broker in Windows PC2 which is connected to Access Point through LAN.
3. Run MQTT broker in Windows PC2 using following command. Open Command prompt and go to MQTT installed folder (Ex: C:\Program Files\mosquitto) and run the following command:

**mosquito.exe -p 1883 -v**

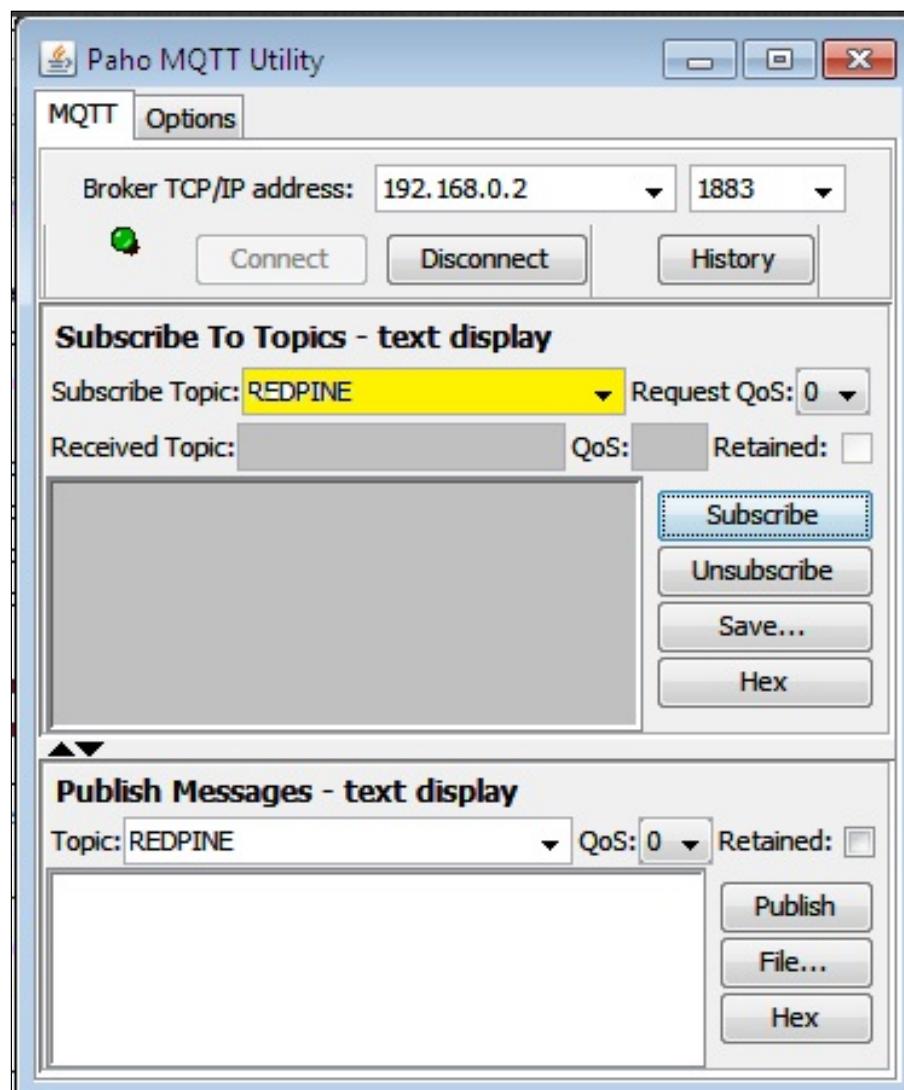


A screenshot of a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe - mosquito.exe -p 1883 -v". The window shows the following output:  
C:\Program Files\mosquitto>mosquitto.exe -p 1883 -v  
1446714024: mosquitto version 1.4.4 (build date 18/09/2015 22:24:05.22) starting  
1446714024: Using default config.  
1446714024: Opening ipv6 listen socket on port 1883.  
1446714024: Opening ipv4 listen socket on port 1883.  
-

4. Open MQTT client utility in Windows PC3 and connect to MQTT broker by giving Windows PC2 IP address and MQTT broker port number in Broker TCP/IP address field.



5. After successful connection, subscribe to the topic from MQTT client utility.



6. After the program gets executed, the Redpine device will be connected to the same access point having the configuration same as that of in the application and get IP.
7. Once the device gets connected to the MQTT broker, it will subscribe to the topic **RSI\_MQTT\_TOPIC (Ex: "REDPINE")**. The user can see the client connected and subscribe information in the MQTT broker.

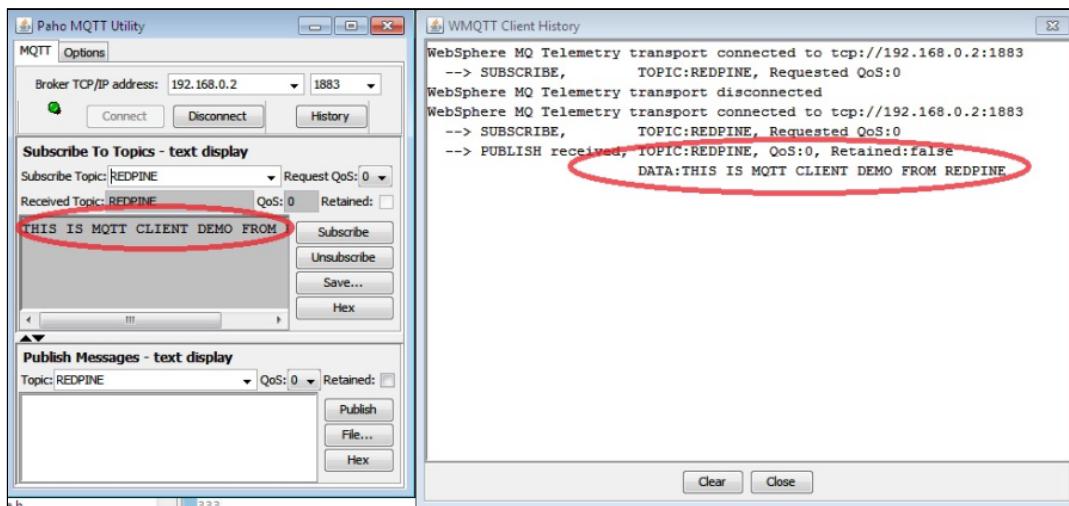
```
C:\ Administrator: C:\Windows\system32\cmd.exe - mosquitto.exe -p 1883 -v
C:\Program Files\mosquitto>mosquitto.exe -p 1883 -v
1459923726: mosquitto version 1.4.8 (build date 14/02/2016 15:33:31.09) starting
1459923726: Using default config.
1459923726: Opening ipv6 listen socket on port 1883.
1459923726: Opening ipv4 listen socket on port 1883.
1459923736: mosquitto version 1.4.8 terminating

C:\Program Files\mosquitto>
C:\Program Files\mosquitto>mosquitto.exe -p 1883 -v
1459923747: mosquitto version 1.4.8 (build date 14/02/2016 15:33:31.09) starting
1459923747: Using default config.
1459923747: Opening ipv6 listen socket on port 1883.
1459923747: Opening ipv4 listen socket on port 1883.
1459923760: New connection from 192.168.0.3 on port 1883.
1459923760: New client connected from 192.168.0.3 as MQTTCLIENT <c1, k100>.
1459923760: Sending CONNACK to MQTTCLIENT <, 0>
1459923774: Received SUBSCRIBE from MQTTCLIENT
1459923774:    REDPINE <QoS 0>
1459923774: MQTTCLIENT 0 REDPINE
1459923774: Sending SUBACK to MQTTCLIENT
```

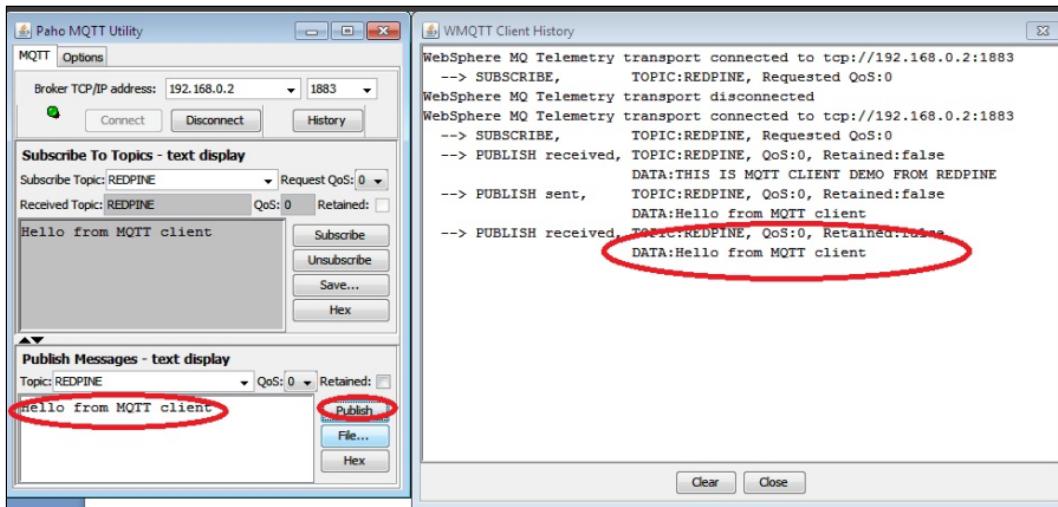
8. After successful subscription to the topic **RSI\_MQTT\_TOPIC** (Ex: "REDPINE"), the device publishes a message which is given in **publish\_message** array (Ex: "THIS IS MQTT CLIENT DEMO FROM REDPINE") on the subscribed topic.

9. MQTT client utility which is running on Windows PC3 will receive the message published by the Redpine device as it subscribes to the same topic.

Please refer to the below image for MQTT client utility and message history.



10. Now publish a message using MQTT Utility on the same topic. Now this message is the message received by the Redpine device.

**Note:**

Multiple MQTT client instances can be created

### 8.15.5 Limitations

MQTT client application keeps on polling for the data to receive on the subscribed topic irrespective of receive timeout mentioned in the rsi\_mqtt\_poll\_for\_recv\_data API.

## 8.16 Multicast Example

### 8.16.1 Protocol Overview

In Networking, Multicast IP Routing protocols are used to distribute data (for example, audio/video streaming broadcasts) to multiple recipients. Using multicast, a source can send a single copy of data to a single multicast address, which is then distributed to an entire group of recipients.

A multicast group identifies a set of recipients that are interested in a particular data stream, and is represented by an IP address from a well-defined range. Data sent to this IP address is forwarded to all members of the multicast group.

Routers between the source and recipients duplicate data packets and forward multiple copies wherever the path to recipients diverges. Group membership information is used to calculate the best routers at which to duplicate the packets in the data stream to optimize the use of the network.

### 8.16.2 Example Overview

#### Overview

This application demonstrates how to add Redpine device to a multicast group and how to send and receive multicast data on a UDP socket.

In this application, the Redpine device connects to Wi-Fi access point and opens UDP socket and joins to a Multicast group ID. After successful join, application sends data to multicast group ID and receives data from Multicast group ID using opened UDP socket.

## Sequence of Events

This Application explains user how to:

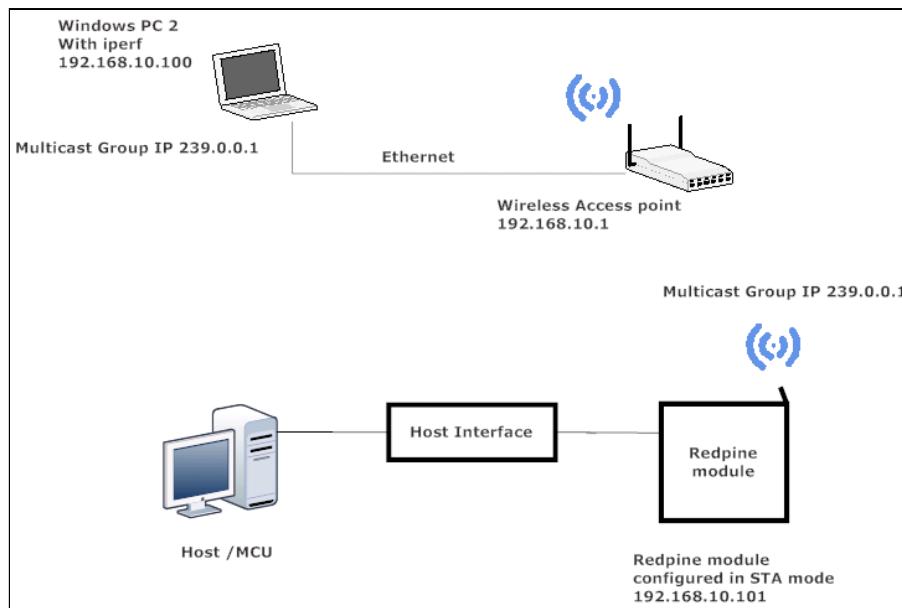
- Configure Redpine device as a Wi-Fi station
- Connect to Wi-Fi Access Point
- Open UDP socket
- Join multicast group ID
- Send UDP data to multicast group ID
- Receive UDP data coming from Multicast group

### 8.16.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect using either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with Keil or IAR IDE in case of WiSeMCU
- Redpine Module
- WLAN Access point
- Windows PC2 with iperf to send and receive Multicast data



**Figure 1: Setup Diagram**

#### 8.16.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_multicast\_app.c** file and update/modify following macros  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel to be scanned, If it is 0, device will scan all the channels

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE    RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK             "<psk>"
```

**DEVICE\_PORT** port refers internal UDP port number

```
#define DEVICE_PORT     5001
```

**SERVER\_PORT** port refers remote UDP server port number

```
#define SERVER_PORT     5002
```

**MULTICAST\_GROUP\_ADDRESS** refers the device to which multicast group address has to join.

**MULTICAST\_GROUP\_ADDRESS** address should be configured in long format and in little endian byte order.

**Example:** To configure "239.0.0.1" as multicast group IP address, update the macro

**MULTICAST\_GROUP\_ADDRESS** as **0x010000EF**.

```
#define MULTICAST_GROUP_ADDRESS 0x010000EF
```

**NUMBER\_OF\_PACKETS** refers how many packets to send/receive to/from multicast group before leaving multicast group.

```
#define NUMBER_OF_PACKETS <no of packets>
```

**RECV\_BUFFER\_SIZE** is expected size of data in each packet. If packet is half the size of receive buffer, then Device will read for the data again.

```
#define RECV_BUFFER_SIZE <receive buf size>
```

#### To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE 1
```

#### Note:

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in longformat and in little endian byte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP 0X0A0AA8C0
```

IP address of the gateway should also be in longformat and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY 0x010AA8C0
```

IP address of the network mask should also be in longformat and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK 0X00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_CUSTOM_FEATURE_BIT_MAP 0
#define RSI_BAND RSI_BAND_2P4GHZ
```

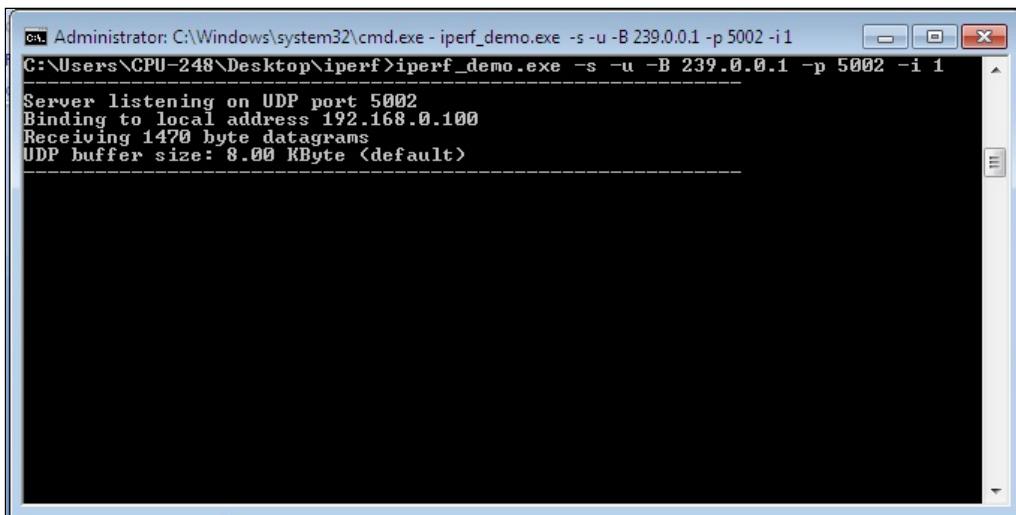
**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

### Executing the Application

1. Configure the access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. Open multicast UDP server socket on port number **SERVER\_PORT(Ex:5002)** by binding to **MULTICAST\_GROUP\_ADDRESS(Ex:239.0.0.1)** using iperf application in Windows PC2 which is connected to access point through LAN.

**iperf\_demo.exe -s -u -B 239.0.0.1 -p 5002 -i 1**



3. After the program gets executed, the Redpine device will be connected to the access point and get IP.
4. After successful connection with accesspoint, the device will join tothe multicast group and sends configured numberof UDP packets to multicast group address on port number **SERVER\_PORT**. After the device starts sending multicast data, user can see UDP receiving data on opened UDP socket on port number **SERVER\_PORT**.

```
Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -u -B 239.0.0.1 -p 5002 -i1
[1240] Sent 309 datagrams
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -u -B 239.0.0.1 -p 5002 -i 1
Server listening on UDP port 5002
Binding to local address 192.168.0.100
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
-----
[1224] local 192.168.0.100 port 5002 connected with 192.168.0.101 port 5001
[ ID] Interval Transfer Bandwidth Jitter Lost/Total Datagram
[1224] 0.0- 1.0 sec 1.15 KBytes 9.41 Kbits/sec 21.160 ms -1614698882/12146
444 <-1.3e+002%>
[1224] 1.0- 2.0 sec 1.99 KBytes 16.3 Kbits/sec 11.905 ms -85/ 0 <-1.%>
[1224] 2.0- 3.0 sec 85 datagrams received out-of-order
[1224] 2.0- 3.0 sec 1.01 KBytes 8.26 Kbits/sec 18.892 ms -43/ 0 <-1.%>
[1224] 2.0- 3.0 sec 43 datagrams received out-of-order
[1224] 3.0- 4.0 sec 1.57 KBytes 12.9 Kbits/sec 22.440 ms -67/ 0 <-1.%>
[1224] 3.0- 4.0 sec 67 datagrams received out-of-order
[1224] 4.0- 5.0 sec 1.90 KBytes 15.6 Kbits/sec 11.004 ms -81/ 0 <-1.%>
[1224] 4.0- 5.0 sec 81 datagrams received out-of-order
[1224] 5.0- 6.0 sec 1.83 KBytes 15.0 Kbits/sec 14.264 ms -78/ 0 <-1.%>
[1224] 5.0- 6.0 sec 78 datagrams received out-of-order
[1224] 6.0- 7.0 sec 1.27 KBytes 10.4 Kbits/sec 14.391 ms -54/ 0 <-1.%>
[1224] 6.0- 7.0 sec 54 datagrams received out-of-order
[1224] 7.0- 8.0 sec 1.24 KBytes 10.2 Kbits/sec 13.357 ms -53/ 0 <-1.%>
[1224] 7.0- 8.0 sec 53 datagrams received out-of-order
[1224] 8.0- 9.0 sec 1.66 KBytes 13.6 Kbits/sec 9.550 ms -71/ 0 <-1.%>
[1224] 8.0- 9.0 sec 71 datagrams received out-of-order
[1224] 9.0-10.0 sec 816 Bytes 6.53 Kbits/sec 18.346 ms -34/ 0 <-1.%>
[1224] 9.0-10.0 sec 34 datagrams received out-of-order
[1224] 10.0-11.0 sec 1.08 KBytes 8.83 Kbits/sec 20.584 ms -46/ 0 <-1.%>
[1224] 10.0-11.0 sec 46 datagrams received out-of-order
```

5. After sending configured NUMBER of UDP packets from device, remote Windows PC2 stops receiving data on SERVER\_PORT and the device waits for receiving multicast data on UDP port number **DEVICE\_PORT**.
6. From Windows PC2, after UDP data reception stops, open UDP client socket and send UDP data to multicast IP address with port number **DEVICE\_PORT** by giving following command in **iperf**, **iperf\_demo.exe -c 239.0.0.1 -p <DEVICE\_PORT> -u -i 1 -t 100 -T32**.

```
Administrator: Command Prompt - iperf_demo.exe -c 239.0.0.1 -p 5001 -u -i 1 -t 20
C:\Users\test>cd Desktop
C:\Users\test\Desktop>cd iperf
C:\Users\test\Desktop\iperf>iperf_demo.exe -c 239.0.0.1 -p 5001 -u -i 1 -t 20
-----
Client connecting to 239.0.0.1, UDP port 5001
Sending 1470 byte datagrams
Setting multicast TTL to 1
UDP buffer size: 8.00 KByte (default)
-----
[132] local 192.168.10.5 port 57212 connected with 239.0.0.1 port 5001
[ ID] Interval Transfer Bandwidth
[132] 0.0- 1.0 sec 129 KBytes 1.06 Mbits/sec
[132] 1.0- 2.0 sec 128 KBytes 1.05 Mbits/sec
[132] 2.0- 3.0 sec 128 KBytes 1.05 Mbits/sec
[132] 3.0- 4.0 sec 128 KBytes 1.05 Mbits/sec
[132] 4.0- 5.0 sec 128 KBytes 1.05 Mbits/sec
[132] 5.0- 6.0 sec 128 KBytes 1.05 Mbits/sec
[132] 6.0- 7.0 sec 128 KBytes 1.05 Mbits/sec
[132] 7.0- 8.0 sec 128 KBytes 1.05 Mbits/sec
[132] 8.0- 9.0 sec 128 KBytes 1.05 Mbits/sec
[132] 9.0-10.0 sec 129 KBytes 1.06 Mbits/sec
[132] 10.0-11.0 sec 128 KBytes 1.05 Mbits/sec
```

7. The device will read configured number of packets which are coming from joined multicast group address and leave from that joined multicast group.

## 8.17 Over The Air Firmware Upgradation From Server example

### 8.17.1 Example Overview

#### Overview

This application demonstrates how to upgrade new firmware to Redpine device using remote TCP server.

In this application Redpine device connects to Access Point and using OTAF command establishes TCP client connection with TCP server opened on remote peer. After successful TCP connection, module sends the firmware file request to remote TCP server and server responds with Firmware file and waits for the next firmware file request. Once firmware file receives from the TCP server, Module loads the firmware file into on to the modules flash. After successful firmware upgrade, OTAF api returns success response.

#### Sequence of Events

This Application explains user how to:

- Configure as station mode
- Open TCP server socket at Access Point
- Connect to Access Point
- Call OTA firmware upgrade api to request Firmware file from remote server
- Send firmware file from remote server.

### 8.17.2 Example Setup

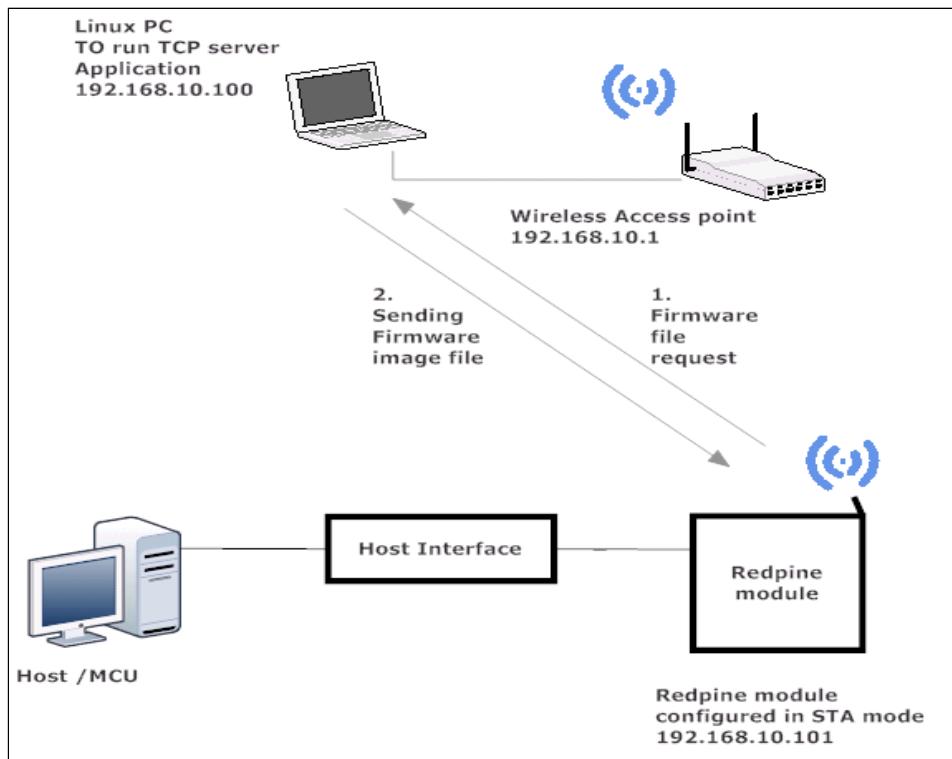
The Redpine device parts require that the host processor should be connected to the device either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Wireless Access point
- Redpine module
- Linux PC with TCP server application (TCP server application providing as part of release package)

**Note:**

TCP server application providing in release package in the following path:  
***sapis/examples/wlan/otaf/firmware\_upgrade\_ota\_server.c***



**Figure 1: Setup Diagram**

#### 8.17.3 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_ota\_firmware\_upgradation\_app.c** file and update/modify following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID                                "<ap name>"
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE                         RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK           "<psk>"
```

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT  
5001
```

**SERVER\_PORT** port refers remote TCP server port number which is opened in Linux PC.

```
#define SERVER_PORT      5001
```

**SERVER\_IP\_ADDRESS** refers remote peer IP (Linux PC) address to connect with TCP server socket.  
IP address should be in long format and in little endian byte order.

**Example:** To configure "192.168.0.100" as remote IP address, update the macro **SERVER\_IP\_ADDRESS** as **0x6400A8C0**.

```
#define SERVER_IP_ADDRESS    0x6400A8C0
```

**RECV\_BUFFER\_SIZE** refers Memory for receive data

```
#define RECV_BUFFER_SIZE     1027
```

#### To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE          1
```

#### Note:

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

The IP address needs to be configuring to the device should be in long format and in little endian byte order.  
**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP          0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro GATEWAY as **0x010AA8C0**

```
#define GATEWAY 0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order.

**Example:** To configure "255.255.255.0" as network mask, update the macro NETMASK as **0x00FFFFFF**

```
#define NETMASK 0x00FFFFFF
```

**OTAF\_SERVER\_PORT** refers remote TCP server port number which is opened in Linux PC.

```
#define OTAF_SERVER_PORT 5001
```

**OTAF\_RX\_TIMEOUT** refers remote TCP RX packet receive timeout .

```
#define OTAF_RX_TIMEOUT 200
```

**OTAF\_TCP\_RETRY\_COUNT** refers to TCP maximum retransmissions count.

```
#define OTAF_TCP_RETRY_COUNT 20
```

**OTAF\_RETRY\_COUNT** refers to OTAF upgradation retry count.

```
#define OTAF_RETRY_COUNT 10
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_PSK
#define RSI_TCP_IP_BYPASS DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_OTA | TCP_IP_FEAT_DHCPV4_CLIENT)
```

```
#define RSI_CUSTOM_FEATURE_BIT_MAP          0
#define RSI_BAND
RSI_BAND_2P4GHZ
```

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. Compile and run by providing port number and Firmware file path from the server application from the path: "**/otaf/firmware\_upgarde\_ota\_server.c**".  
**gcc firmware\_upgarde\_ota\_server.c**  
**./a.out 5001 RS9116.NBZ.WC.GEN.OSI.x.x.x.rps**
3. After the program gets executed, device connects to AP and open TCP client socket.
4. After TCP connection established with remote server, application sends firmware file request to the server.
5. Server receive request and sends Firmware file in chunks.
6. After receiving chunk from remote server, application again sends firmware request to server. Server will wait for the firmware request from WiSeConnect device before sending next chunk.
7. Packet is sent to the device in chunks as shown in the given below figure. After successful upgradtion in TCP server terminal shows "reach end of file".



The terminal window shows a log of network activity. It includes several 'send' and 'recv' operations, each with a packet number (no), size (length), and a message indicating whether it's a send or recv operation. The log ends with a message 'reach end of file'.

```
File Edit View Terminal Tabs Help
root@localhost:/ftpboot/nfs/lib/modules/2.6.... x root@localhost:/work/siva/RS9113.NBZ.WC.GE... x
size of data==1024
send returns 1027
Pkt sent no:1529
waiting for recv
recv length == 0x3
size of data==1024
send returns 1027
Pkt sent no:1530
waiting for recv
recv length == 0x3
size of data==1024
send returns 1027
Pkt sent no:1531
waiting for recv
recv length == 0x3
size of data==1024
send returns 1027
Pkt sent no:1532
waiting for recv
recv length == 0x3
size of data==1024
send returns 1027
Pkt sent no:1533
waiting for recv
recv length == 0x3
size of data==1024
send returns 1027
Pkt sent no:1534
waiting for recv
recv length == 0x3
size of data==1024
send returns 1027
Pkt sent no:1535
waiting for recv
recv length == 0x3
reach end of file
[]
```

**Note:**

After Firmware upgradation, Device needs to be reboot to get effective of new firmware file. After reboot, device will take few minutes to give CARD READY indication after first reboot.

## 8.18 Provisioning Example

### 8.18.1 Example Overview

#### Overview

The provisioning application demonstrates how to provide provisioning to connect desired Access Point using the http server, mdns functionality of Redpine device.

In this application, Redpine device starts as an Access point. After successful creation of AP, application initializes and registers the MDNS service. And application waits for the input to connect with the desired Access Point in station mode. Now, User can open provisioning page by connecting to HTTP server running in device and do scan and select the required Access point to connect and submit the configuration. Once configuration submits, Application receives the configuration set by user and restarts the device in station mode and connects to the desired Access Point.

## Sequence of Events

This Application explains user how to:

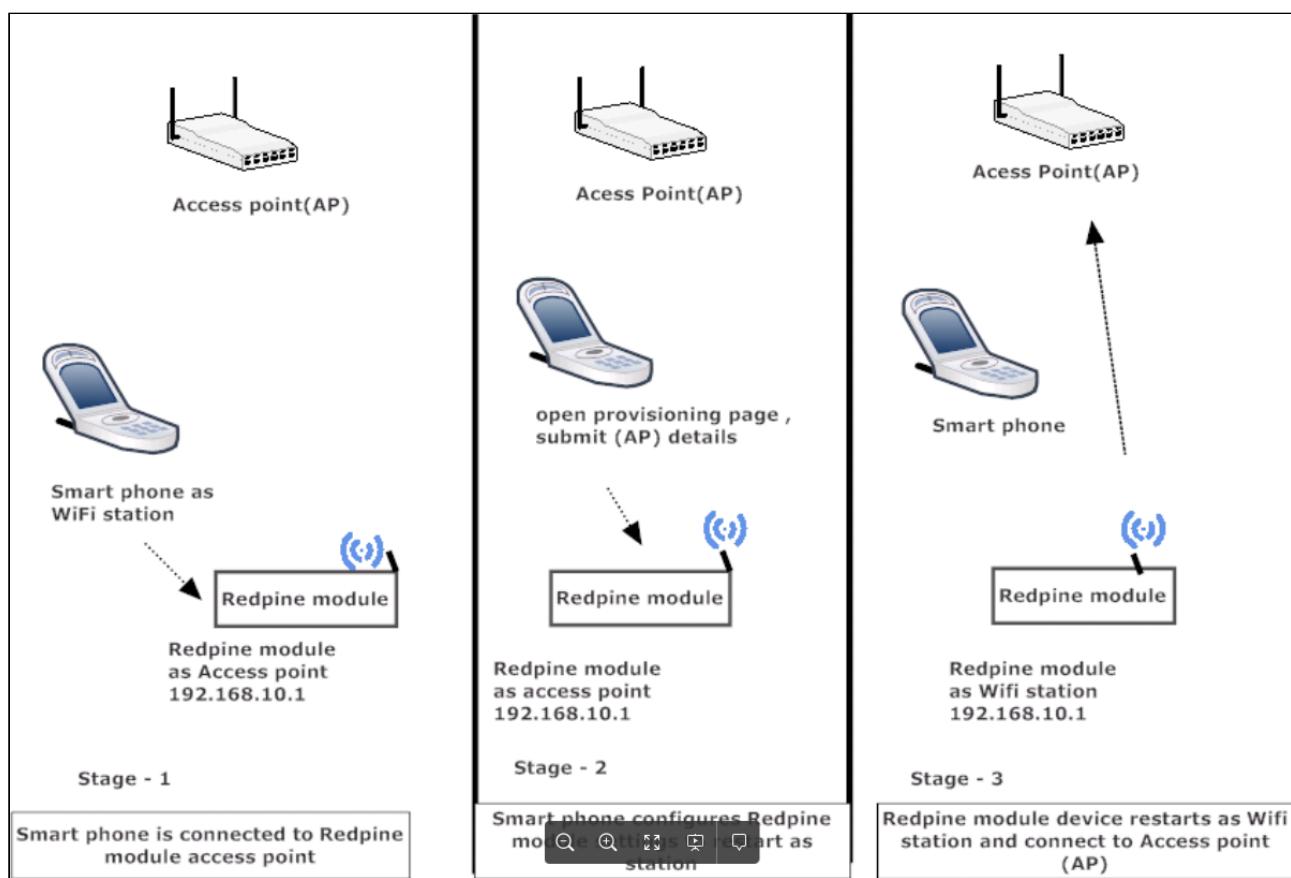
- Start Redpine device as Access Point after power up.
- Connect a station to the device and get IP address through DHCP.
- Open provisioning page of the Device from the browser of the connected station (STA).
- Click on scan button, this updates scan results in scan results table.
- Select the required Access point (AP) settings in the scan results table and submit the provisioning page.
- Redpine Device would restart as Wi-Fi station and join to the configured Access point (AP).

### 8.18.2 Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Wireless Access Point
- Smart Phone
- Redpine module



**Figure 1 Setup diagram**

#### 8.18.3 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_provisioning\_app.c\_file** and update/modify the following macros.  
 SSID refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

CHANNEL\_NO refers to the channel in which AP would be started

```
#define CHANNEL_NO
```

11

**Note:**

Valid values for **CHANNEL\_NO** are 1 to 11 in 2.4GHz band and 36 to 48 & 149 to 165 in 5GHz band. In this example default configured band is 2.4GHz. So, if user wants to use 5GHz band then the user has to set **RSI\_BAND** macro to 5GHz band in **rsi\_wlan\_config.h** file.

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities.

Valid configuration is:

- RSI\_OPEN** - For OPEN security mode
- RSI\_WPA** - For WPA security mode
- RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE
```

```
RSI_WPA2
```

**ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports OPEN, TKIP, CCMP methods.

Valid configuration is:

- RSI\_CCMP** - For CCMP encryption
- RSI\_TKIP** - For TKIP encryption
- RSI\_NONE** - For open encryption

```
#define ENCRYPTION_TYPE
```

```
<encryption_type>
```

**PSK** refers to the secret key if the Access point to be configured in WPA/WPA2 security modes.

```
#define SK
```

```
"1234567890"
```

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons in milliseconds. Allowed values are integers from 100 to 1000 which are multiples of 100.

```
#define BEACON_INTERVAL
```

```
100
```

**DTIM\_INTERVAL**refers DTIM interval of the Access Point. Allowed values are from 1 to 255.

```
#define DTIM_INTERVAL
```

```
4
```

**To configure MDNSD parameters with desired values**

To configure the MDNS IP version

Configure **MDNSD\_IP\_VERSION** to 4 for IPv4 or 6 for IPv6

```
#define MDNSD_IP_VERSION
```

```
4
```

To configure the MDNS time to live in seconds

```
#define MDNSD_INIT_TTL 300
```

To configure the time of the added service to live in seconds

```
#define MDNSD_SERVICE_TTL 300
```

MDNS service port number to be used

```
#define MDNSD_SERVICE_PORT 80
```

User can add multiple services, but this example is designed to add only one service (HTTP). So more services is set to zero.

To add multiple services user should call rsi\_mdnsd\_register\_service with multiple times, each time setting service\_more parameter to 1, and setting service\_more parameter to 0 for the last service.

```
#define MDNSD_  
#define MDNSD_SERVICE_MORE 0
```

Configure MDNS host name

```
#define MDNSD_HOST_NAME "wiseconnect.local."
```

Configure MDNS service pointer name

```
#define MDNSD_POINTER_NAME  
"_http._tcp.local."
```

Configure MDNS service name

```
#define MDNSD_SERVICE_NAME  
"wiseconnect._http._tcp.local"
```

Configure MDNS service text filed

```
#define MDNSD_SERVICE_TEXT "<text field>"
```

**FILE\_NAME** refers File name of the webpage and Json object.

```
#define FILE_NAME "provisioning.html"
```

**To configure IP address**

IP address to be configured in the device should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP 0X010AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY 0X010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK 0X00FFFFFF
```

**Note:**

In AP mode, configure same IP address for both **DEVICE\_IP** and **GATEWAY** macros

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE DISABLE
#define RSI_FEATURE_BIT_MAP FEAT_SECURITY_PSK
#define RSI_TCP_IP_BYPASS DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_SERVER|TCP_IP_FEAT_DHCPV4_CLIENT|
TCP_IP_FEAT_MDNSD|TCP_IP_FEAT_HTTP_SERVER)
```

```
#define RSI_CUSTOM_FEATURE_BIT_MAP 0
#define RSI_BAND RSI_BAND_2P4GHZ
```

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. After the program gets executed, Redpine Device will be started as Access point having the configuration same as that of in the application.
2. Now connect aSmart phone(STA) to Device and getIPaddress.
3. After successful connection open the provisioning page from STA browser by giving the following URL:  
**URL:** wiseconnect.local/provisioning.html

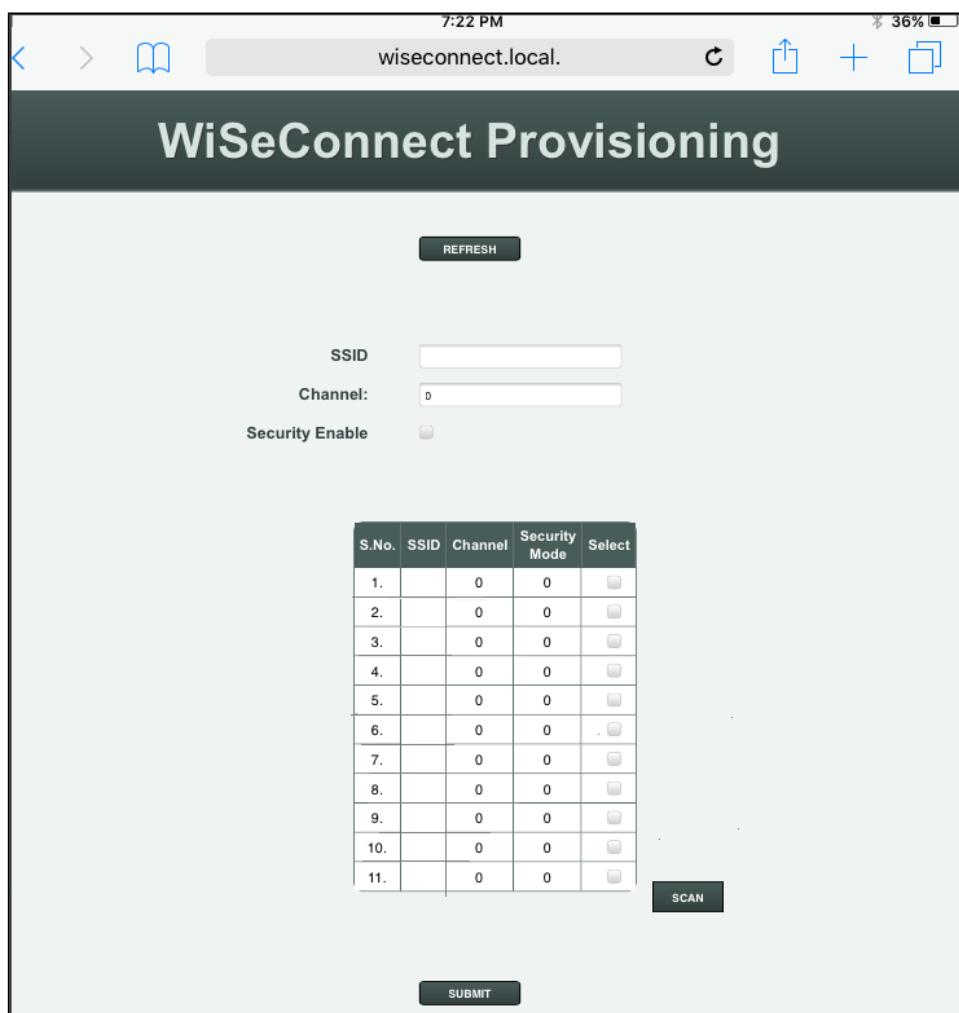
### Note:

Redpine device responds to the mdns requests destined to 5353 port only. So ensure that the WiFi station dorequest for the URL to the Device on 5353 port.

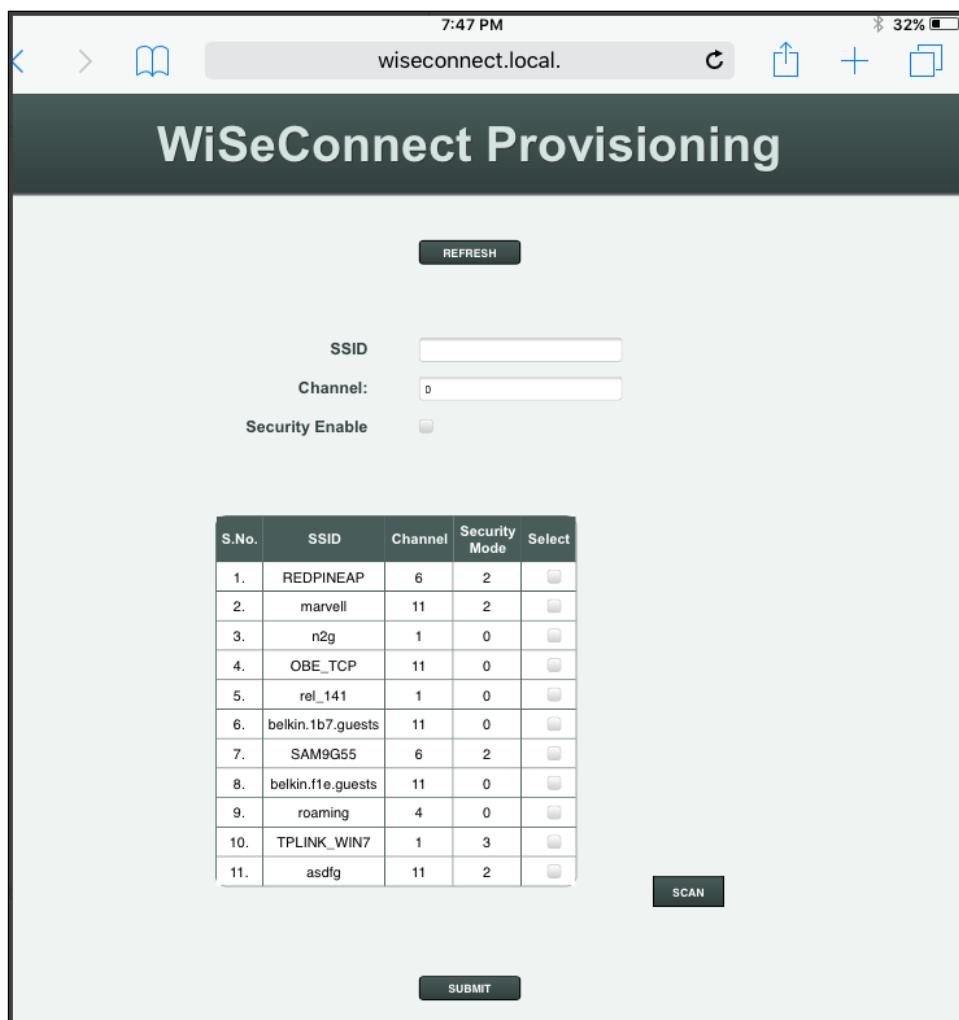
If webpage not opened with the above URL, Use following URL to open provisioning page.

**URL: DEVICE\_IP/provisioning.html** (i.e 192.168.10.1/provisioning.html)

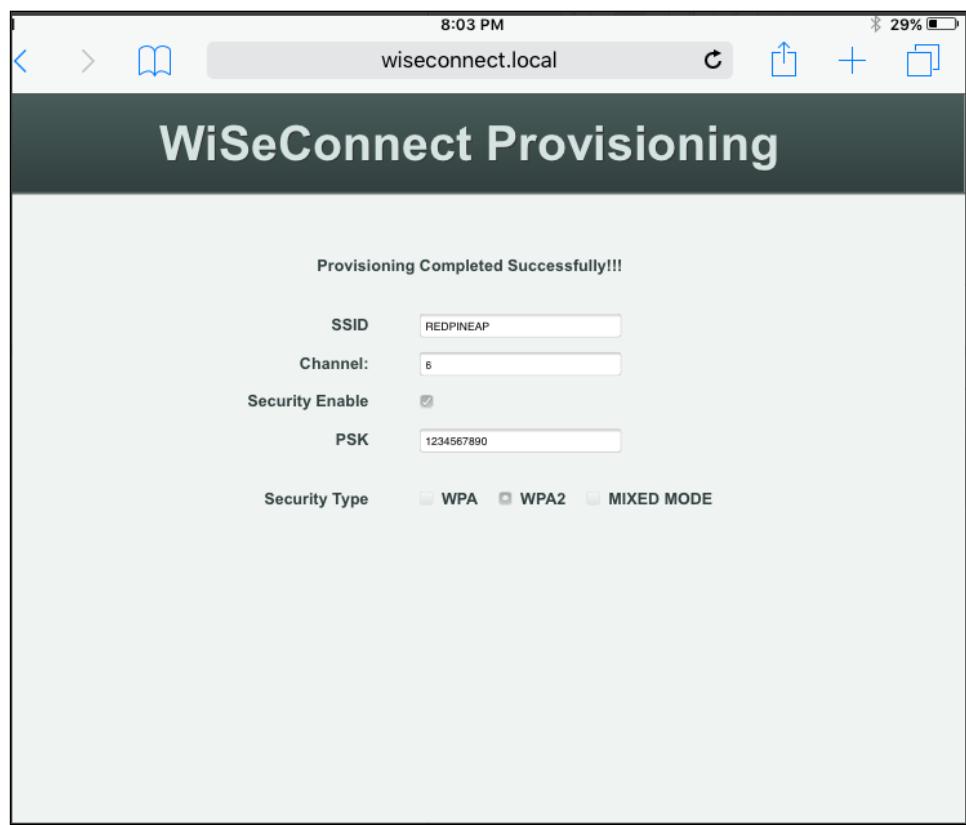
4. Provisioning page is displayed in the browser. Please refer the given below image for provisioning page,



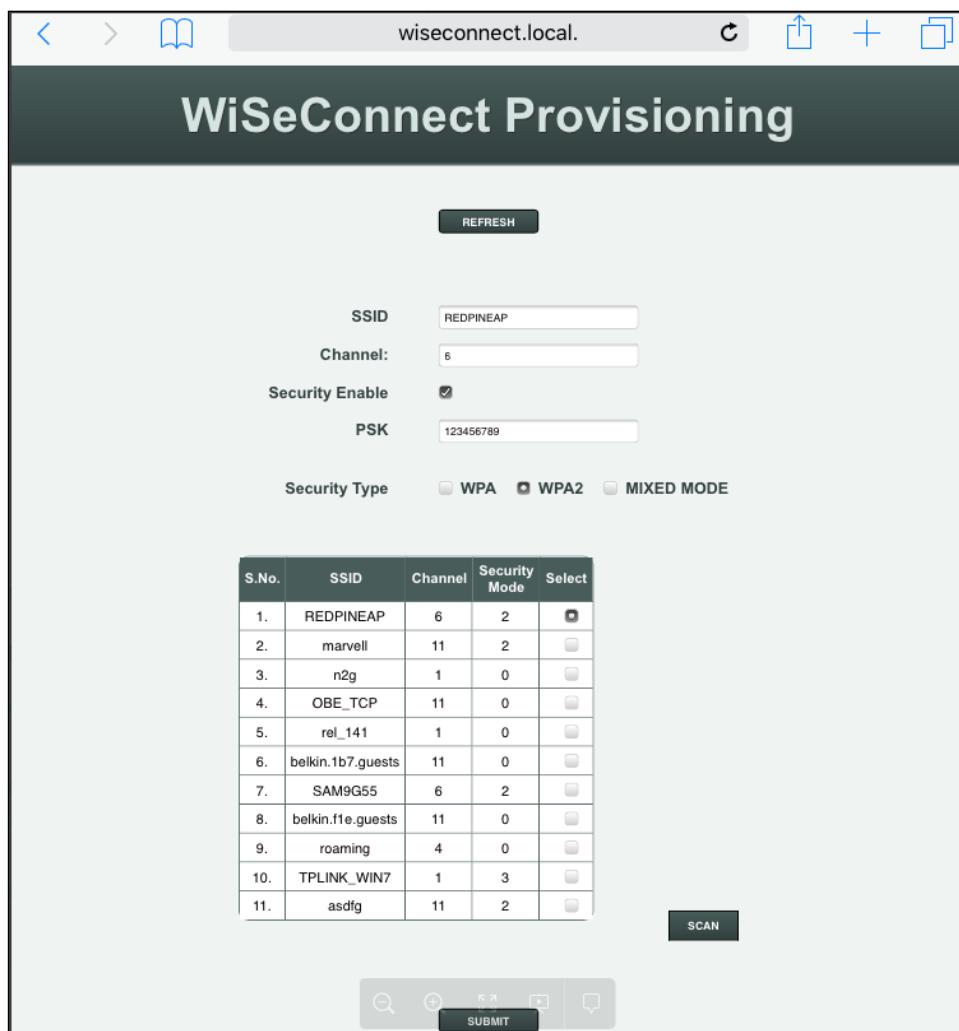
5. Click on scanbutton, scan list table will be updated after a **10 seconds** timeout.



6. Select the required Access point (AP) to which user wants to connect and provide the required PSK in case of AP secured and submit the configuration.



7. WiSeConnect device will restart as WiFi Client and automatically will connect to the selected access Point.



## 8.19 Raw Data Example

### 8.19.1 Example Overview

#### Overview

The raw data application demonstrates how the Redpine device receives the raw data packets (packets of other IP network) and sends them to host, and also how it receives raw data packets from host and sends on air. In this Application, Redpine device will be created as Access point, allow Wi-Fi stations to connect to it. It processes the ARP request packet (raw data) and sends ARP response (raw data). It also process ping request (raw data) of other IP network, and sends ping response (raw data) to it.

#### Sequence of Events

This Application explains user how to:

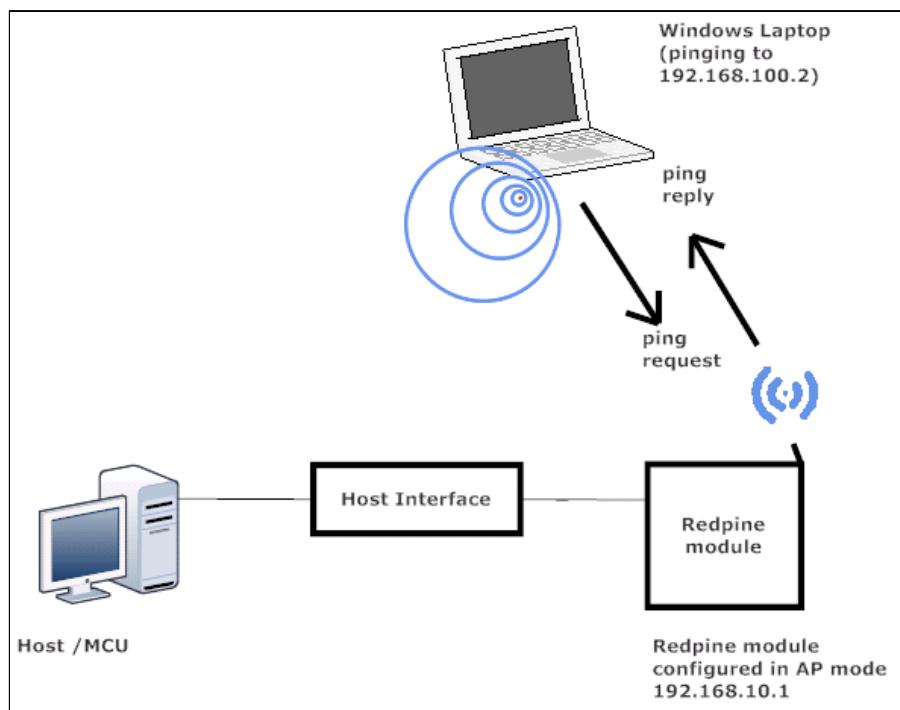
- Redpine Device starts as an access point
- Allow stations to connect
- Reply for ping request and ARP request of other networks also

### 8.19.2 Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- Windows Laptop for Wi-Fi Station



**Figure 1: Setup Diagram**

### 8.19.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_raw\_data\_app.c** file and update/modify following macros,  
**SSID** refers to the name of the Access point to be created.

```
#define SSID                                "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which AP would be started

#define CHANNEL\_NO

11

**Note:**

Valid values for **CHANNEL\_NO** are 1 to 11 in 2.4GHz and 36 to 48 & 149 to 165 in 2.4GHz. In this example default configured band is 2.4GHz. So, if user wants to use 5GHz band then user has to set **RSI\_BAND** macro to 5GHz band in **rsi\_wlan\_config.h** file.

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities.  
Valid configuration is:

- RSI\_OPEN** - For OPEN security mode
- RSI\_WPA** - For WPA security mode
- RSI\_WPA2** - For WPA2 security mode

#define SECURITY\_TYPE

RSI\_OPEN

**ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports OPEN, TKIP, CCMP methods.

Valid configuration is:

- RSI\_CCMP** - For CCMP encryption
- RSI\_TKIP** - For TKIP encryption
- RSI\_NONE** - For open encryption

#define ENCRYPTION\_TYPE

RSI\_NONE

**PSK** refers to the secret key if the Access point is to be configured in WPA/WPA2 security modes.

#define PSK

"<psk>"

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons in milliseconds. Allowed values are integers from 100 to 1000 which are multiples of 100.

#define BEACON\_INTERVAL

100

**DTIM\_INTERVAL** refers DTIM interval of the Access Point. Allowed values are from 1 to 255.

#define DTIM\_INTERVAL

4

**To configure IP address**

IP address to be configured to the device should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP 0X010AA8C0
```

IP address of the gateway should also be in longformat and in little endianbyte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro GATEWAY as **0x010AA8C0**

```
#define GATEWAY  
0X010AA8C0
```

IP address of the network mask should also be in longformat and in little endianbyte order

**Example:** To configure "255.255.255.0" as network mask, update the macro NETMASK as **0x00FFFFFF**

```
#define NETMASK  
0X00FFFFFF
```

**Note:**

In AP mode, configure same IP address for both DEVICE\_IP and GATEWAY macro

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP  
FEAT_SECURITY_OPEN  
#define RSI_TCP_IP_BYPASS RSI_DISABLE  
#define RSI_TCP_IP_FEATURE_BIT_MAP  
(TCP_IP_FEAT_DHCPV4_CLIENT | TCP_IP_FEAT_RAW_DATA)  
#define RSI_CUSTOM_FEATURE_BIT_MAP 0  
#define RSI_BAND  
RSI_BAND_2P4GHZ
```

**Note:**

rsi\_wlan\_config.h file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

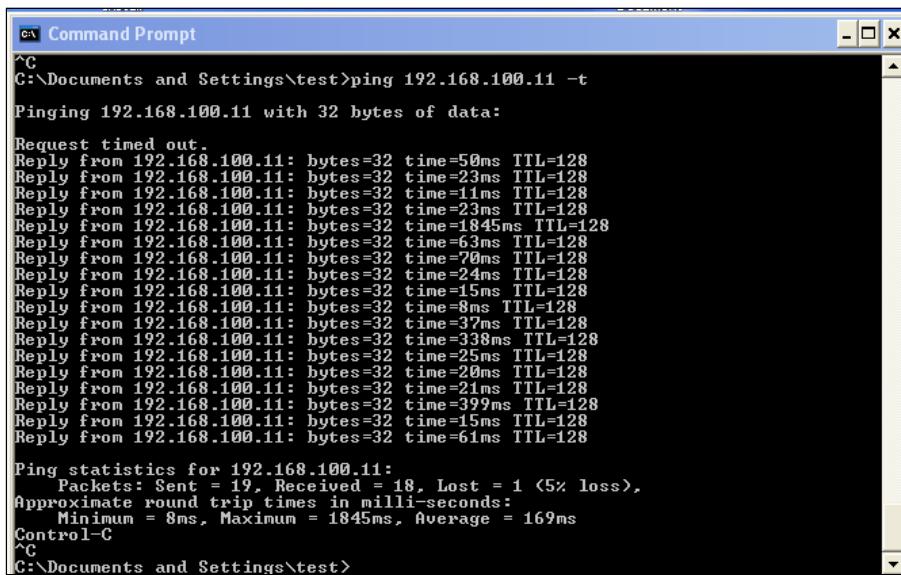
1. After the program gets executed, Redpine Device will be created as Access point and starts beaconing.
2. Now connect Wi-Fi STA (Laptop) to Redpine AP (Ex: AP SSID is "REDPINE\_AP"). After successful connection, Wi-Fi STA gets IP in the configured IP network (Ex: 192.168.10.4)



3. Initiate ping to an IP of other network (Ex: 192.168.100.11) from Wi-Fi STA (laptop).

**Ping 192.168.100.11 -t**

4. Module will reply with ARP response, if connected stations try to ping other IP (which is not in a connected network) and also responds with ping reply for the prior resolved ARP.



```

^C
C:\Documents and Settings\test>ping 192.168.100.11 -t
Pinging 192.168.100.11 with 32 bytes of data:

Request timed out.
Reply from 192.168.100.11: bytes=32 time=50ms TTL=128
Reply from 192.168.100.11: bytes=32 time=23ms TTL=128
Reply from 192.168.100.11: bytes=32 time=11ms TTL=128
Reply from 192.168.100.11: bytes=32 time=23ms TTL=128
Reply from 192.168.100.11: bytes=32 time=1845ms TTL=128
Reply from 192.168.100.11: bytes=32 time=63ms TTL=128
Reply from 192.168.100.11: bytes=32 time=70ms TTL=128
Reply from 192.168.100.11: bytes=32 time=24ms TTL=128
Reply from 192.168.100.11: bytes=32 time=15ms TTL=128
Reply from 192.168.100.11: bytes=32 time=8ms TTL=128
Reply from 192.168.100.11: bytes=32 time=37ms TTL=128
Reply from 192.168.100.11: bytes=32 time=338ms TTL=128
Reply from 192.168.100.11: bytes=32 time=25ms TTL=128
Reply from 192.168.100.11: bytes=32 time=20ms TTL=128
Reply from 192.168.100.11: bytes=32 time=21ms TTL=128
Reply from 192.168.100.11: bytes=32 time=399ms TTL=128
Reply from 192.168.100.11: bytes=32 time=15ms TTL=128
Reply from 192.168.100.11: bytes=32 time=61ms TTL=128

Ping statistics for 192.168.100.11:
    Packets: Sent = 19, Received = 18, Lost = 1 <5% loss>,
Approximate round trip times in milli-seconds:
    Minimum = 8ms, Maximum = 1845ms, Average = 169ms
Control-C
^C
C:\Documents and Settings\test>

```

## 8.20 SMTP Client Example

### 8.20.1 Protocol Overview

SMTP is a Simple Mail Transfer Protocol, which is used to send mails from mail client to mail server. SMTP is a connection-oriented, text-based protocol in which a mail sender communicates with a mail receiver by issuing command strings and supplying necessary data over a reliable ordered data stream channel, typically a Transmission Control Protocol (TCP) connection.

An SMTP session consists of commands originated by an SMTP client (the initiating agent, sender, or transmitter) and corresponding responses from the SMTP server (the listening agent, or receiver) so that the session is opened, and session parameters are exchanged.

### 8.20.2 Example Overview

#### Overview

This application demonstrates how Redpine device sends a mail to SMTP server. In this application, Redpine device connects to Access Point in station mode and connects to SMTP server using SMTP client. After successful connection with SMTP server, application sends a mail to SMTP server.

#### Sequence of Events

This Application explains user how to:

- Connect to Access Point
- Connect with SMTP server opened in remote peer
- Send a mail to remote SMTP server

### 8.20.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

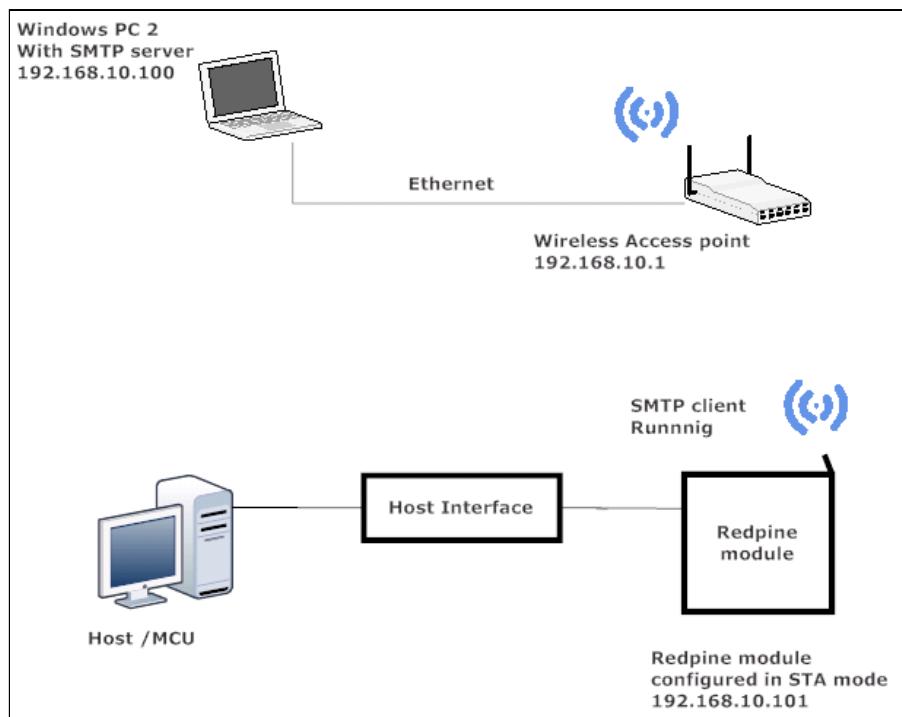
### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- WiFi Access point
- Windows PC2
- Redpine module
- A TCP server application running on the Windows PC2 (This example uses iperf for windows )

**Note:**

Download simple SMTP server from the below link:

<http://nilhcem.github.com/FakeSMTP/downloads/fakeSMTP-latest.zip>



**Figure 1: Setup Diagram**

#### 8.20.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open ***rsi\_smtp\_client\_app.c*** file and update / modify the following macros.  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels.

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode

**RSI\_WPA** - For WPA security mode

**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE    RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK             "<psk>"
```

#### To configure SMTP client Parameters

To select IPv6, FLAGS should be set to 1, by default it supports IPv4

```
#define FLAGS          0
```

To configure the SMTP server port, default port is 25. If server can listen on other non standard ports, change the **SMTP\_PORT**

```
#define SMTP_PORT       25
```

To configure the username for authentication

```
#define USERNAME        "username"
```

To configure the password for authentication

```
#define PASSWORD        "password"
```

To configure the sender's address

```
#define FROM_ADDRESS "abc@mail.local"
```

IP address of the SMTP server should be in longformat and in little endianbyte order.

**Example:** To configure "192.168.0.2" as IP address, update the macro **SERVER\_IP** as **0x010AA8C0**.

```
#define SERVER_I 0x0200A8C0
```

To configure the authentication type

Supported types are,

RSI\_SMTP\_CLIENT\_AUTH\_LOGIN (1) – for login type

RSI\_SMTP\_CLIENT\_AUTH\_PLAIN (3) – For plain authentication

```
#define AUTH_TYPE  
RSI_SMTP_CLIENT_AUTH_LOGIN
```

To configure the priority

Supported priorities are,

RSI\_SMTP\_MAIL\_PRIORITY\_LOW (1) - For Low priority

RSI\_SMTP\_MAIL\_PRIORITY\_NORMAL (2) - For Normal Priority

RSI\_SMTP\_MAIL\_PRIORITY\_HIGH (4) - For High Priority

```
#define PRIORITY  
RSI_SMTP_MAIL_PRIORITY_NORMAL
```

To configure domain name of the client.

```
#define CLIENT_DOMAIN  
"mymail.mail.com"
```

To configure mail recipient address.

```
#define MAIL_RECIPIENT_ADDRESS  
"test@mail.local"
```

To configure subject of the mail. This should be a string

```
#define MAIL SUBJECT "TEST"
```

To configure mail body.

```
#define MAIL_BODY "TEST BODY"
```

**Note:**

Mail body to the mail send API (rsi\_smtp\_client\_mail\_send\_async) need not be a string. A buffer with definite length can be pointed to the API.

**To configure IP address**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE
```

1

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in longformat and in little endianbyte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP
```

0X0A0AA8C0

IP address of the gateway should also be in longformat and in little endianbyte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY
```

0X010AA8C0

IP address of the network mask should also be in longformat and in little endianbyte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK
```

0X00FFFFFF

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:  
`#define CONCURRENT_MODE RSI_DISABLE`

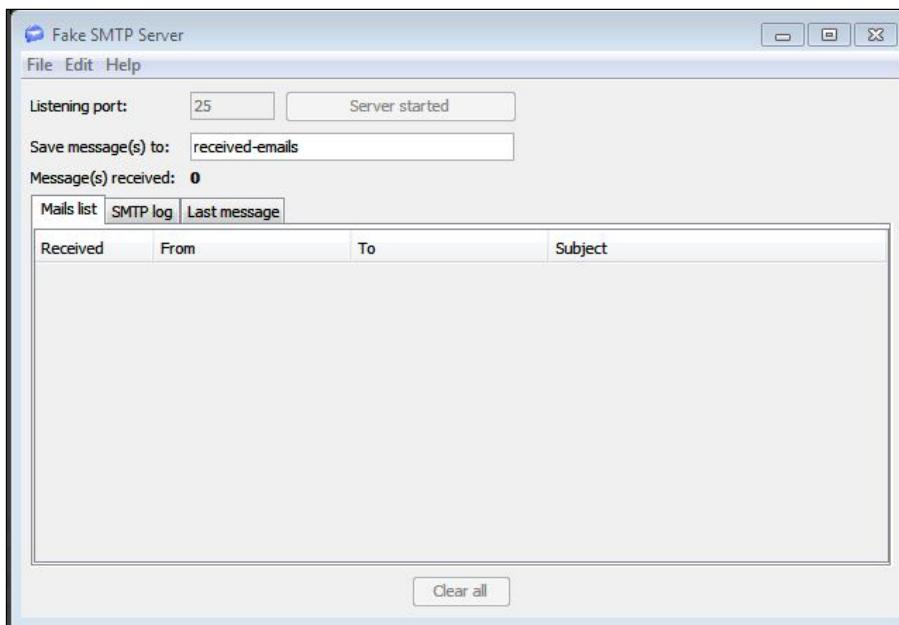
```
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_CLIENT | TCP_IP_FEAT_SMTP_CLIENT)
#define RSI_CUSTOM_FEATURE_BIT_MAP 0
#define RSI_BAND RSI_BAND_2P4GHZ
```

**Note:**

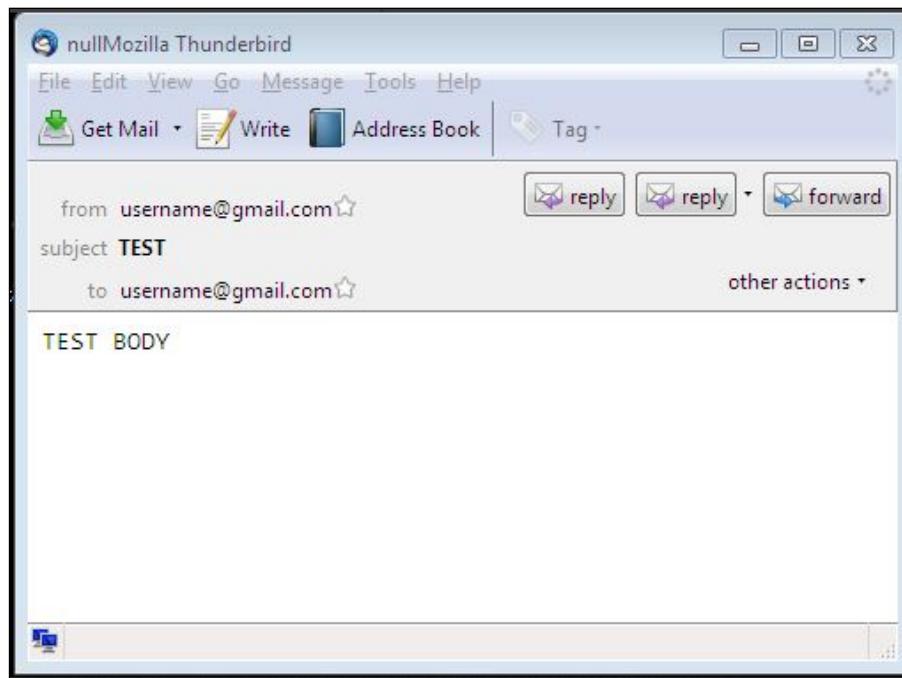
rsi\_wlan\_config.h file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

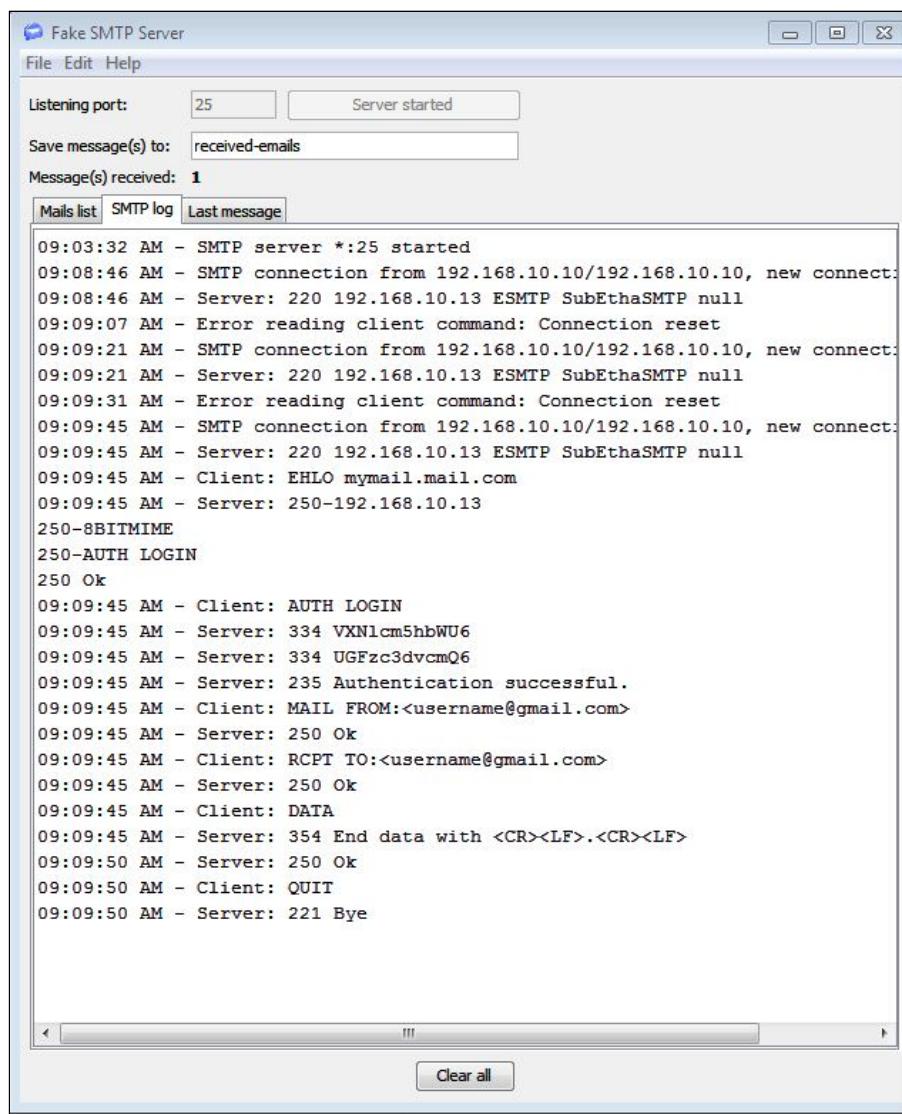
1. Configure the Access point in OPEN / WPA-PSK / WPA2-PSK mode in order to connect the Redpine device in STA mode.
2. Download SMTP server and run SMTP server in Windows PC2.



3. After the program gets executed, the Redpine Device would be connected to access point and get IP.
4. After successful connection with access Point, Device starts authenticating with the SMTP server running on Windows PC2.
5. After successful authentication, it sends a mail to the server.
6. Please refer the given below images for SMTP server receives mail sent by Redpine device.
7. Double click the mail and check mail subject and mail body sent by WiSeConnect Device.



8. Check the log messages at SMTP server.



## 8.21 SMTP-POP3 Client Example

### 8.21.1 Application Overview

SMTP is a Simple Mail Transfer Protocol, which is used to send mails from mail client to mail server. SMTP is a connection-oriented, text-based protocol in which a mail sender communicates with a mail receiver by issuing command strings and supplying necessary data over a reliable ordered data stream channel, typically a Transmission Control Protocol (TCP) connection.

An SMTP session consists of commands originated by an SMTP client (the initiating agent, sender, or transmitter) and corresponding responses from the SMTP server (the listening agent, or receiver) so that the session is opened, and session parameters are exchanged.

POP3 is a Post Office Protocol, which is designed for downloading mails from the mail server. POP3 is a connection-oriented, text-based protocol in which a POP3 client communicates with the POP3 server by issuing command strings and supplying necessary data over a reliable ordered data stream channel. POP3 supports simple download-and-delete requirements for access to remote mailboxes.

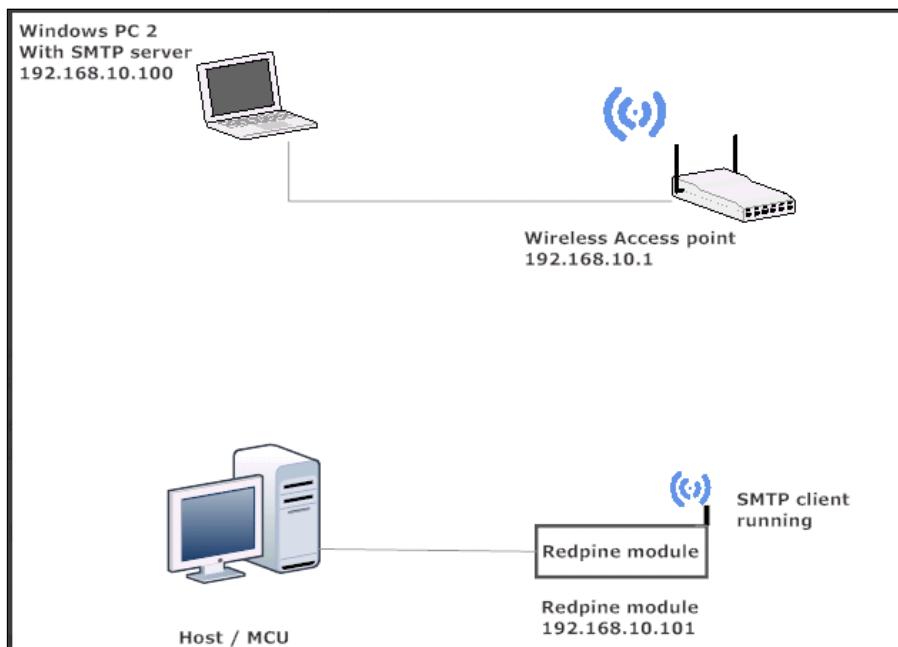
An POP3 session consists of commands originated by an POP3 client and corresponding responses from the POP3 server so that the session is opened, and session parameters are exchanged.

The SMTP-POP3 client application demonstrates how WiSeConnect device can send a mail to an SMTP server and how it can download a mail from POP3 server.

- Configure an SMTP server in a PC connected to Access point
- Connect Redpine device to Access point wirelessly
- Create SMTP client and send mail to SMTP server

#### 8.21.2 Setup required

1. Windows PC with Keil or IAR IDE
2. Redpine Module
3. Wlan Access point
4. Windows PC with a simple SMTP server installed.



**Figure 1: Setup Diagram**

#### 8.21.3 Description

This application is used to configure Redpine device to connect to a WLAN Access point and send a mail to the SMTP server running on the Windows PC to which the Access point is connected.  
User can find some simple SMTP servers from internet.

Please find one at the following location:

<http://nilhcem.github.com/FakeSMTP/downloads/fakeSMTP-latest.zip>

#### 8.21.4 Configuration and Execution of the Application

##### Configuring the Application

1. Edit the **rsi\_smtp\_pop3\_client\_app.c** file in the following path:  
**sapis/examples/wlan/smtp\_pop3\_client**  
From the given configuration,

```
#define SSID           "<ap_name>"  
#define CHANNEL_NO     0  
#define SECURITY_TYPE  <security-  
type>  
#define PSK            "<psk>"
```

**SSID** refers to the name of the Access point to connect.

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0,device will scan all channels  
**SECURITY\_TYPE** refers to the type of security

**PSK** refers to the secret key if the Access point was configured in WPA/WPA2 security modes  
To configure SMTP client Parameters

2. To select IPv6, FLAGS should be set to 1,by default it supports IPv4

```
#define FLAGS          0
```

To configure the SMTP server port,default port is 25. If server can listen on other non standard ports,  
change the SMTP\_PORT

```
#define SMTP_PORT      25
```

To configure the SMTP client username for authentication

```
#define SMTP_USERNAME  "<username>"
```

To configure the password for authentication

```
#define PASSWORD       "<password>"
```

To configure the sender's address

```
#define FROM_ADDRESS   "<from  
address>"
```

IP address of the SMTP server should be in long format and in little endian byte order.  
To configure the POP3 server port,default port is 110. If server can listen on other non standard ports,  
change the POP3\_PORT

```
#define POP3_PORT      110
```

To configure the POP3 client username for authentication

```
#define POP3_USERNAME           "<username>"
```

To configure the password for authentication

```
#define POP3_PASSWORD           "<password>"
```

Example: To configure "192.168.10.1" as IP address, update the macro **SERVER\_IP** as **0x010AA8C0**.

```
#define SERVER_IP               0x9200A8C0
```

To configure the authentication type, supported types are defined in Application include file

```
#define AUTH_TYPE                <auth_type>
```

To configure the authentication type, supported priorities are defined in Application include file

```
#define PRIORITY                 <priority>
```

To configure domain name of the client.

```
#define CLIENT_DOMAIN            "<domain name>"
```

To configure mail recipient address.

```
#define MAIL_RECIPIENT_ADDRESS    "<recipient  
address>"
```

To configure subject of the mail.This should be a string

```
#define MAIL SUBJECT              "<subject>"
```

To configure mailbody .

```
#define MAIL_BODY           "<message  
body>"
```

**Note :**

Mail body to the mail send API (rsi\_smtp\_client\_mail\_send\_async) need not be a string. A buffer with definite length can be pointed to the API

Enable/Disable DHCP mode

- 1 – Enables DHCP mode (gets the IP from DHCP server)
- 0 – Disables DHCP mode

```
#define DHCP_MODE           <dhcp mode>
```

3. To configure static IP address

IP address to be configured to the device should be in long format and in little endian byte order.

Example: To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP           0X010AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY             0X010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK              0X00FFFFFF
```

**Update Wlan configuration file:  
[sapis/include/rsi\\_wlan\\_config.h](#)**

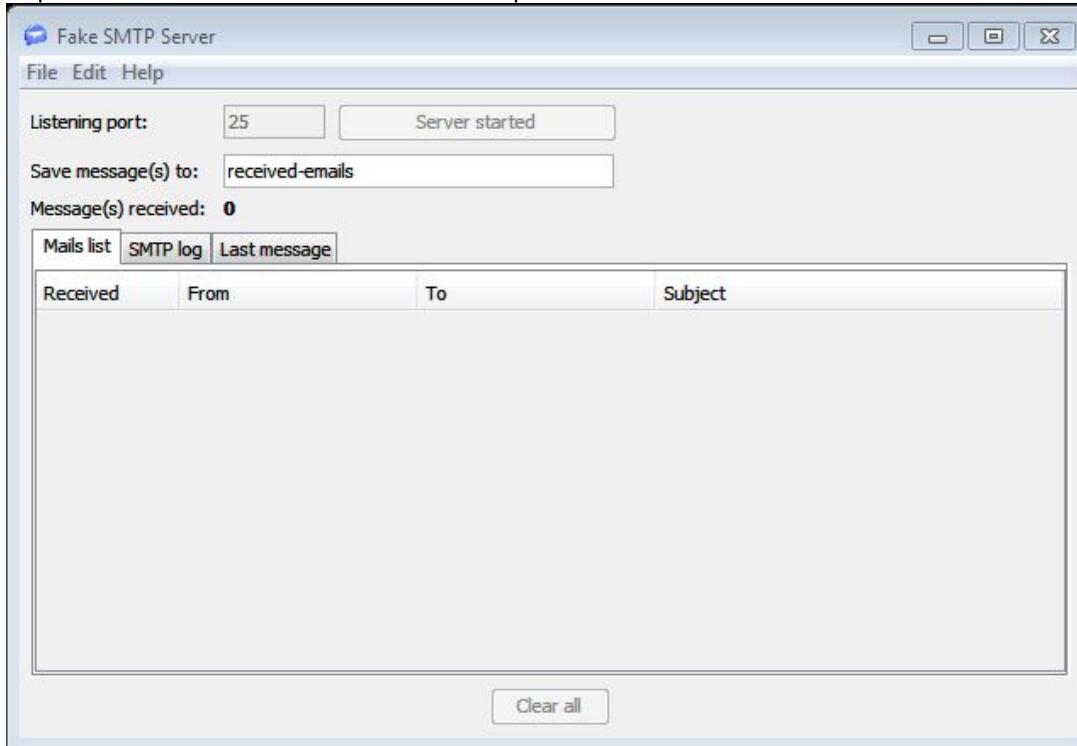
CONCURRENT_MODE	DISABLE
RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
RSI_TCP_IP_BYPASS	DISABLE
RSI_TCP_IP FEATURE_BIT_MAP	(TCP_IP_FEAT_DHCPV4_CLIENT TCP_IP_FEAT_SMTP_CLIENT TCP_IP_FEAT_POP3_CLIENT )
RSI_CUSTOM_FEATURE_BIT_MAP0	
RSI_BAND	RSI_BAND_2P4GHZ

**Note:**

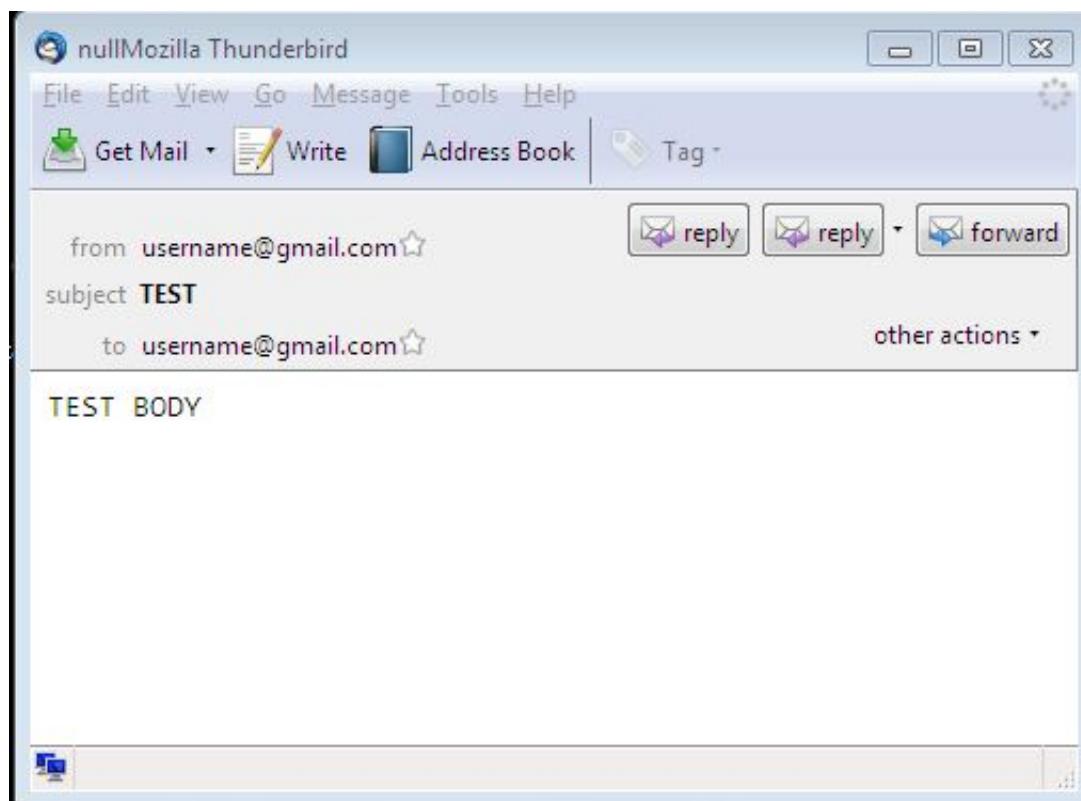
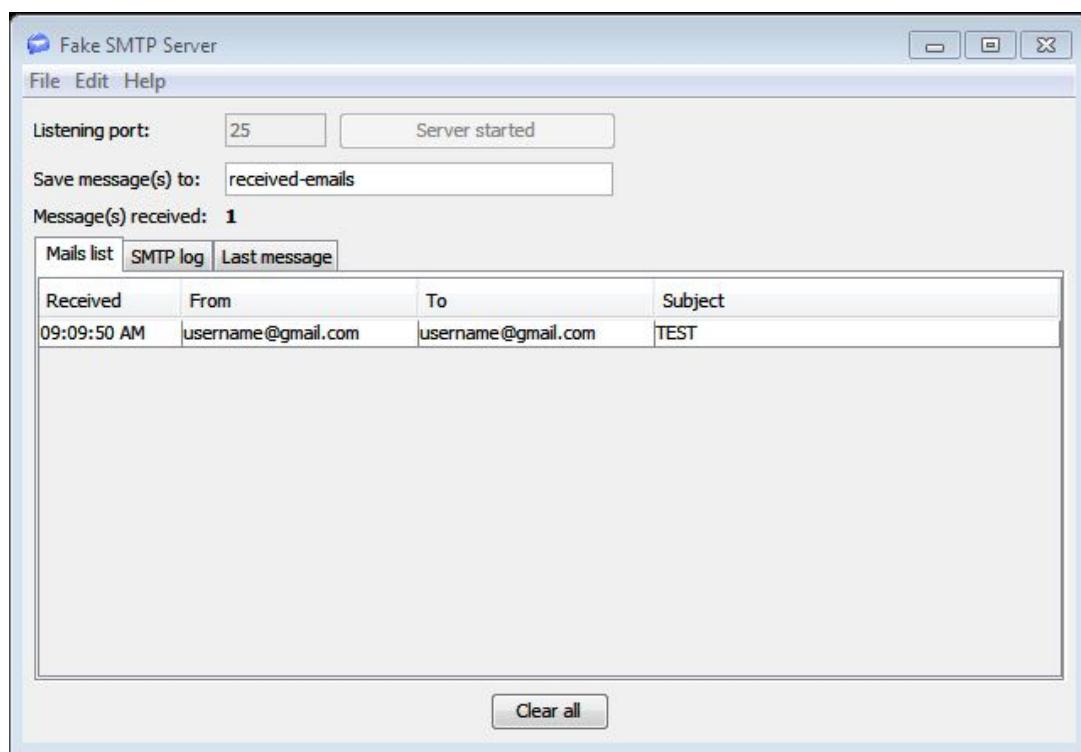
rsi\_wlan\_config.h file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Connect Redpine device to the Windows PC running Keil or IAR IDE.
2. Configure the macros in the files located at  
**rsi\_wlan\_config.h**
3. Run Simple SMTP server on the PC to which access point is connected.



4. Build and launch the application.
5. After the program gets executed, Redpine device would be connected to access point and get IP.
6. Device as an smtp client start authenticating with the SMTP server.
7. After successful authentication, it sends a mail to the server.
8. Device as an pop3client start authenticating with the POP3 server.  
After successful authentication, it receives a mail from the server.



Fake SMTP Server

File Edit Help

Listening port: 25 Server started

Save message(s) to: received-emails

Message(s) received: 1

Mails list SMTP log Last message

```

09:03:32 AM - SMTP server *:25 started
09:08:46 AM - SMTP connection from 192.168.10.10/192.168.10.10, new connect:
09:08:46 AM - Server: 220 192.168.10.13 ESMTP SubEthSMTP null
09:09:07 AM - Error reading client command: Connection reset
09:09:21 AM - SMTP connection from 192.168.10.10/192.168.10.10, new connect:
09:09:21 AM - Server: 220 192.168.10.13 ESMTP SubEthSMTP null
09:09:31 AM - Error reading client command: Connection reset
09:09:45 AM - SMTP connection from 192.168.10.10/192.168.10.10, new connect:
09:09:45 AM - Server: 220 192.168.10.13 ESMTP SubEthSMTP null
09:09:45 AM - Client: EHLO mymail.mail.com
09:09:45 AM - Server: 250-192.168.10.13
250-8BITMIME
250-AUTH LOGIN
250 Ok
09:09:45 AM - Client: AUTH LOGIN
09:09:45 AM - Server: 334 VXNlcm5hbWU6
09:09:45 AM - Server: 334 UGFzc3dvcmQ6
09:09:45 AM - Server: 235 Authentication successful.
09:09:45 AM - Client: MAIL FROM:<username@gmail.com>
09:09:45 AM - Server: 250 Ok
09:09:45 AM - Client: RCPT TO:<username@gmail.com>
09:09:45 AM - Server: 250 Ok
09:09:45 AM - Client: DATA
09:09:45 AM - Server: 354 End data with <CR><LF>.<CR><LF>
09:09:50 AM - Server: 250 Ok
09:09:50 AM - Client: QUIT
09:09:50 AM - Server: 221 Bye

```

Clear all

## 8.22 SNTP Client Example

### 8.22.1 Protocol Overview

Simple Network Time Protocol (SNTP) is a simplified version of Network Time Protocol (NTP) that is used to synchronize computer clocks on a network. This simplified version of NTP is generally used when full implementation of NTP is not needed.

SNTP is a simplified access strategy for servers and clients using NTP. SNTP synchronizes a computer's system time with a server that has already been synchronized by a source such as a radio, satellite receiver or modem.

SNTP supports unicast, multicast and anycast operating modes. In unicast mode, the client sends a request to a dedicated server by referencing its unicast address. Once a reply is received from the server, the client determines the time, roundtrip delay and local clock offset in reference to the server. In multicast mode, the server sends an

---

unsolicited message to a dedicated IPv4 or IPv6 local broadcast address. Generally, a multicast client does not send any requests to the service because of the service disruption caused by unknown and untrusted multicast servers. The disruption can be avoided through an access control mechanism that allows a client to select a designated server he or she knows and trusts.

### 8.22.2 Example Overview

#### Overview

This application demonstrates how Redpine device gets info from SNTP server. In this application, Redpine device connects to Access Point in client mode and connects to SNTP server. After successful connection with SNTP server, application gets time and date info from SNTP server.

#### Sequence of Events

This Application explains user how to:

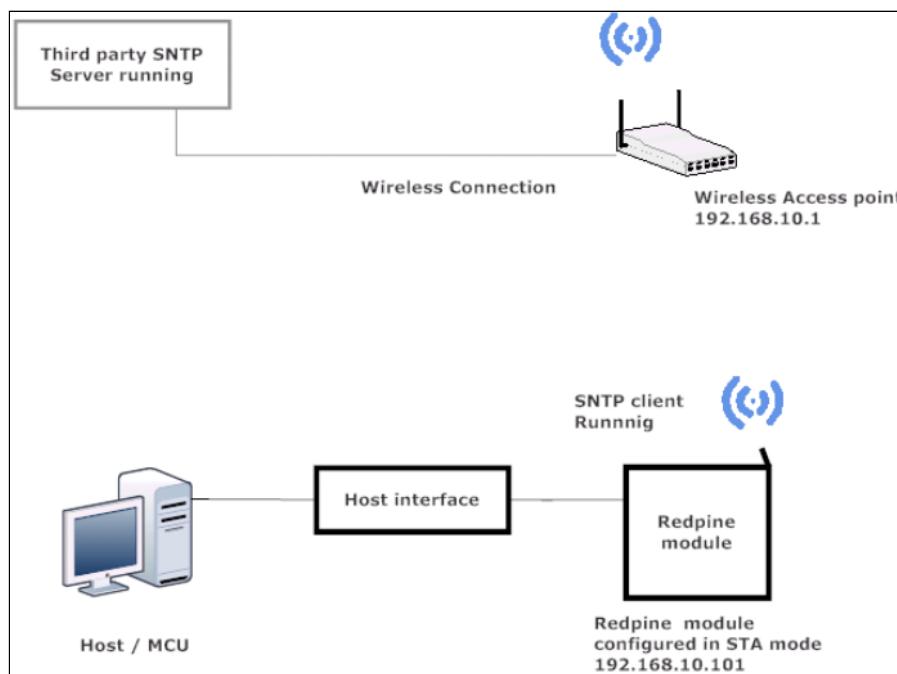
- Redpine device in a client mode
- Connect with SNTP server
- Get time and date info from the SNTP server

### 8.22.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU part offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- WiFi Access point with internet
- Redpine module



**Figure 1: Setup diagram**

#### 8.22.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_wps\_station.c** file and update/modify following macros:

**SSID** refers to the name of the Access point. In WPS method it should NULL.

```
#define SSID NULL
```

**SECURITY\_TYPE** refers to the type of security. In WPS method WiSeConnect module supports WPS push button method and WPS PIN method.

In this examples, WiSeConnect module connecting to AP using WPS push button method. So, set **SECURITY\_TYPE** macro to **RSI\_WPS\_PUSH\_BUTTON**.

```
#define SECURITY_TYPE  
RSI_WPS_PUSH_BUTTON
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes. In WPS method PSK is not needed.

```
#define PSK NULL
```

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT <local port>
```

**SERVER\_PORT** port refers remote TCP server port number which is opened in Windows PC2.

```
#define SERVER_PORT <remote port>
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with TCP server socket.

IP address should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.100" as IP address, update the macro **DEVICE\_IP** as **0x640AA8C0**.

```
#define SERVER_IP_ADDRESS 0x640AA8C0
```

**NUMBER\_OF\_PACKETS** refers how many packets to receive from TCP client

```
#define NUMBER_OF_PACKETS <no of  
packets>
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 8000
```

To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE 1
```

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP 0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order.

**Example:** To configure "192.168.10.1" as Gateway, update the macro GATEWAY as **0x010AA8C0**

```
#define GATEWAY 0x010AA8C0
```

IP address of the network mask should also be in longformatand in little endianbyteorder

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK 0x00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_CLIENT | TCP_IP_FEAT_SNTP_CLIENT)
#define RSI_CUSTOM_FEATURE_BIT_MAP 0
#define RSI_BAND
RSI_BAND_2P4GHZ
```

**Note:**

rsi\_wlan\_config.h file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the Access point in OPEN / WPA-PSK / WPA2-PSK mode in order to connect Redpine device in STA mode.
2. Connect to SNTP server and request server for information.  
eg: SNTP server ip address 128.138.141.172
3. After the program gets executed, Redpine Device would be connected to Access point and gets IP.
4. After successful connection with access Point, Device starts connection with the SNTP server.
5. After successful connection, module will send request to the server for time response.

## 8.23 Station Ping Example

### 8.23.1 PING Overview

Ping is used diagnostically to ensure that a host computer the user is trying to reach is actually operating. Ping works by sending an Internet Control Message Protocol (ICMP) Echo Request to a specified interface on the network and waiting for a reply. Ping can be used for troubleshooting to test connectivity and determine response time.

---

### 8.23.2 Example Overview

#### Overview

The application demonstrates how to configure Redpine device in client mode to send ping request to target IP address.

#### Sequence of Events

This Application explains user how to:

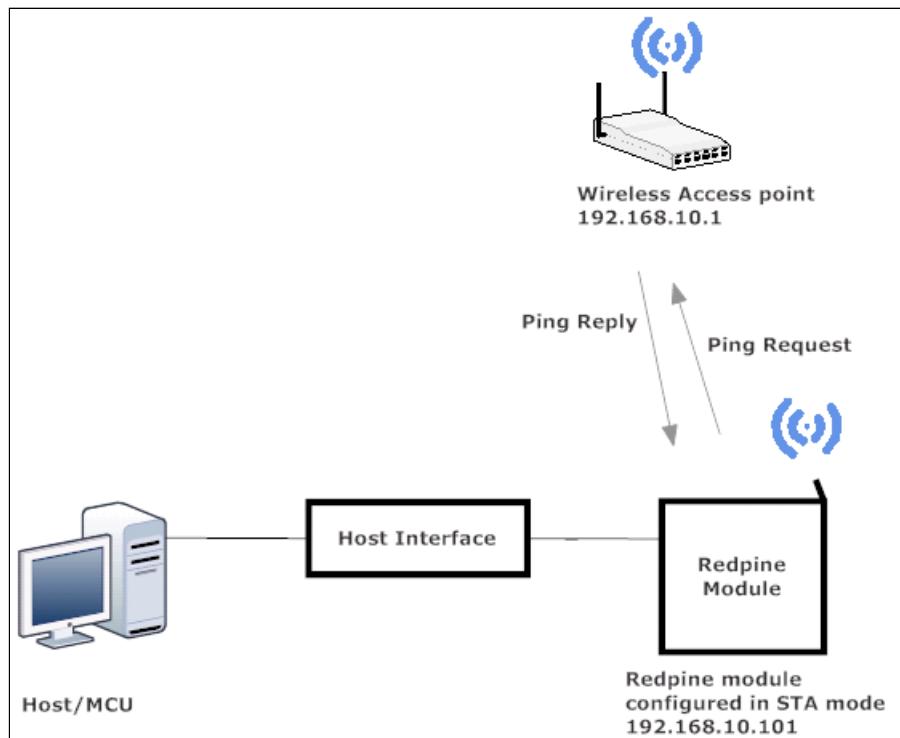
- Connect to Access Point in station mode
- Send Ping requests to configured target IP address

### 8.23.3 Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Wireless Access Point
- Redpine module
- A TCP server application running on the Windows PC2 (This example uses iperf for windows )



**Figure 1: Setup Diagram**

#### 8.23.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_station\_ping.c** file and update/modify following macros,  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels.

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE
```

RSI\_OPEN

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK "<psk>"
```

**To configure IP address**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE
```

1

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP
```

0X0A0AA8C0

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY
```

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK
```

0x00FFFFFF

Configure following macro stopping initiate ping with the remote peer  
IP address of the remote peer (AP IP address).

**Example:** To configure "192.168.10.1" as **REMOTE\_IP**, update the macro **REMOTE\_IP** as **0x010AA8C0**.

```
#define REMOTE_IP
```

0x010AA8C0

**PING\_SIZE** refers the size of ping packet.

```
#define PING_SIZE 100
```

**NUMBER\_OF\_PACKETS** refers how many number of pings to send from device.

```
#define NUMBER_OF_PACKETS 1000
```

**Application memory length which is required by the driver**

```
#define GLOBAL_BUFF_LEN 8000
```

2. User can connect to access point through PMK  
To Enable keep 1 else 0

```
#define CONNECT_WITH_PMK 0
```

**Note:**

If CONNECT\_WITH\_PMK is enable ,SECURITY\_TYPE is set to RSI\_WPA2\_PMK

3. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_CLIENT| TCP_IP_FEAT_ICMP)
#define RSI_CUSTOM_FEATURE_BIT_MAP 0
#define RSI_BAND RSI_BAND_2P4GHZ
/* ping response timeout in seconds */
#define RSI_PING_REQ_TIMEOUT 1
```

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. After the program gets executed, Redpine module configured as client and connects to AP and gets IP.

- 
3. After successful connection with the Access Point, the device starts sending ping requests to the given **REMOTE\_IP** with configured **PING\_SIZE** to check availability of target Device.
  4. Device sends the number of ping packets configured in **NUMBER\_OF\_PACKETS**.
  5. In `rsi_station_ping.c` file, **rsi\_wlan\_ping\_async** API returns success status, which means that the ping request packet is successfully sent in to the medium. When actual ping response comes from the remote node, it is known from the status parameter of the callback function (**rsi\_ping\_response\_handler**) registered in the Ping API.

## 8.24 Store Configuration Profile Example

### 8.24.1 Store Configuration Overview

This feature is used to save the parameters into non-volatile memory which are used either to join to an Access point (auto-join mode) or to create an Access point (auto-create mode).

#### In client mode

The module can connect to a pre-configured access point after it boots up. This feature facilitates fast connection to a known network.

#### In Access Point mode

The module can be configured to come up as an Access Point every time it boots-up.

### 8.24.2 Application Overview

#### Overview

This application demonstrates how to store the given configuration in non-volatile memory and how to enable the stored configuration after bootup.

In this application, default configuration storing into the non-volatile memory is AP mode profile. After saving the configuration, application enables the configuration to get effective from next bootup/reset and then resets the device. After successful reset, Redpine device starts as an Access point with the saved configuration. Now user can scan the access point from station and connect to it and verify the connectivity by initiating Ping from connected station.

#### Sequence of Events

This Application explains user how to:

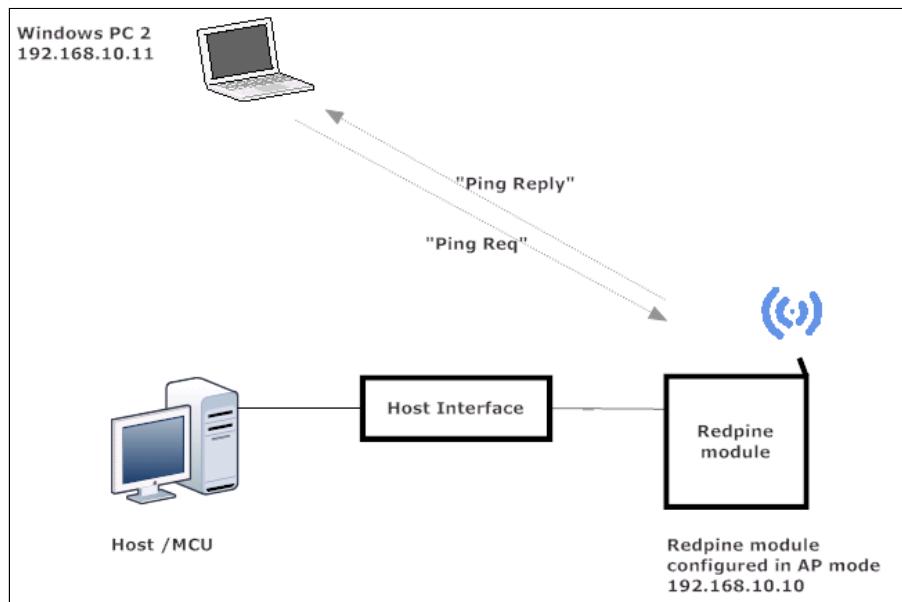
- Initialize the Redpine device.
- Fill the configuration profile based on the selected mode (Ex: AP mode configuration)
- Add profile into non-volatile memory
- Get saved profile.
- Enable auto configuration
- Reset the device
- Initialize the device

### 8.24.3 Application Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- WiFi Access point
- Redpine module
- A TCP server application running on the Windows PC2 (This example uses iperf for windows )



**Figure 1: Setup Diagram**

#### 8.24.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_store\_config\_profile\_app.c** file and update/modify following macros:  
**PROFILE\_TYPE** refers to the type of the profile storing into the non-volatile memory.  
Valid configurations are,  
**RSI\_WLAN\_PROFILE\_CLIENT - 0**  
**RSI\_WLAN\_PROFILE\_P2P - 1**  
**RSI\_WLAN\_PROFILE\_EAP - 2**  
**RSI\_WLAN\_PROFILE\_AP - 6**

```
#define PROFILE_TYPE  
RSI_WLAN_PROFILE_AP
```

**AUTO\_CONFIG\_ENABLE** refers to enable or disable auto configuration.

Valid configurations are,  
0 – to disable Auto configuration  
2 – to enable Auto configuration

#define AUTO\_CONFIG\_ENABLE

2

Application memory length which is required by the driver  
**#define GLOBAL\_BUFF\_LEN8000**

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_CUSTOM_FEATURE_BIT_MAP
#define RSI_BAND
RSI_BAND_2P4GHZ
```

0

3. Please find the **rsi\_wlan\_config.h** file for default configuration stored client/AP/EAP client/P2P mode profiles.

Following is the configuration parameters for AP mode profile.

///! Transmission data rate.Physical rate at which data has to be transmitted.

```
#define RSI_CONFIG_AP_DATA_RATE
RSI_DATA_RATE_AUTO
```

///! To set wlanfeatureselectbit map

```
#define RSI_CONFIG_AP_WLAN_FEAT_BIT_MAP
(FEAT_SECURITY_PSK)
```

///! TCP/IP feature select bitmap for selecting TCP/IP features **#define**  
RSI\_CONFIG\_AP\_TCP\_IP\_FEAT\_BIT\_MAP TCP\_IP\_FEAT\_DHCPV4\_SERVER  
///! To set custom feature selectbit map

```
#define RSI_CONFIG_AP_CUSTOM_FEAT_BIT_MAP
```

0

///! RSI\_BAND\_2P4GHZ(2.4GHz) or RSI\_BAND\_5GHZ(5GHz) or RSI\_DUAL\_BAND

```
#define RSI_CONFIG_AP_BAND  
RSI_BAND_2P4GHZ
```

//! Tx power level

```
#define RSI_CONFIG_AP_TX_POWER RSI_POWER_LEVEL_HIGH //! AP SSID  
#define RSI_CONFIG_AP_SSID  
"REDPINE_AP"
```

//! RSI\_BAND\_2P4GHZ(2.4GHz) or RSI\_BAND\_5GHZ(5GHz) or RSI\_DUAL\_BAND

```
#define RSI_CONFIG_AP_BAND  
RSI_BAND_2P4GHZ
```

//! To configure AP channel number

```
#define RSI_CONFIG_AP_CHANNEL
```

6

//! To configure security type

```
#define RSI_CONFIG_AP_SECURITY_TYPE
```

RSI\_WPA

//! To configure encryption type

```
#define RSI_CONFIG_AP_ENCRYPTION_TYPE
```

1

//! To configure PSK

```
#define RSI_CONFIG_AP_PSK
```

"1234567890"

//! To configure beacon interval

```
#define RSI_CONFIG_AP_BEACON_INTERVAL
```

100

//! To configure DTIM period

#define RSI\_CONFIG\_AP\_DTIM

2

/// This parameter is used to configure keepalive type

#define RSI\_CONFIG\_AP\_KEEP\_ALIVE\_TYPE 0 //!  
RSI\_NULL\_BASED\_KEEP\_ALIVE  
#define RSI\_CONFIG\_AP\_KEEP\_ALIVE\_COUNTER 0 //! 100

/// This parameter is used to configure keep alive period

#define RSI\_CONFIG\_AP\_KEEP\_ALIVE\_PERIOD 100

/// This parameter is used to configure maximum stations supported

#define RSI\_CONFIG\_AP\_MAX\_STATIONS\_COUNT 4

/// TCP\_STACK\_USED BIT(0) - IPv4, BIT(1) -IPv6, (BIT(0) | BIT(1)) -Both IPv4 and IPv6

#define RSI\_CONFIG\_AP\_TCP\_STACK\_USED BIT(0)

/// IP address of the module  
/// E.g: 0xA0AA8C0 == 192.168.10.10

#define RSI\_CONFIG\_AP\_IP\_ADDRESS 0xA0AA8C0

/// IP address of netmask  
/// E.g: 0x00FFFFFF == 255.255.255.0

#define RSI\_CONFIG\_AP\_SN\_MASK\_ADDRESS 0x00FFFFFF

/// IP address of Gateway  
/// E.g: 0xA0AA8C0 == 192.168.10.10

#define RSI\_CONFIG\_AP\_GATEWAY\_ADDRESS 0xA0AA8C0

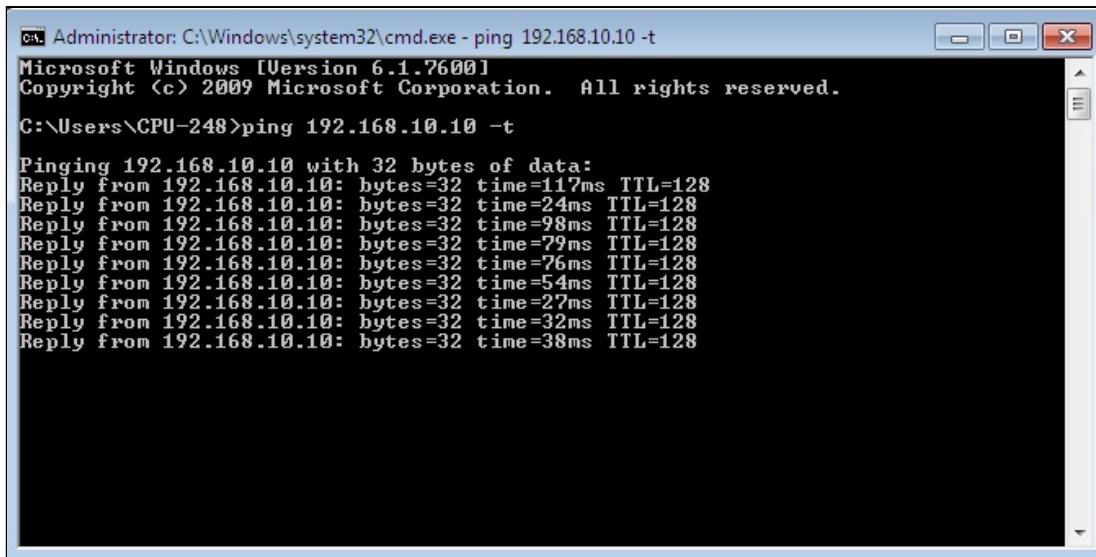
4. If user wants to store different parameters in AP/client/eap client/p2p mode profile, please update the above configuration with required parameters.

**Note:**

rsi\_wlan\_config.h file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. After program gets executed, application fills the configuration structure with the given configuration and adds the configuration to non-volatile memory and enables the auto configuration.
2. After enabling auto configuration, application resets the Redpine device. After successful reset, Redpine device bootup into Access Point mode with the stored configuration (Default saved SSID is "REDPINE\_AP").
3. Now scan from station and connect to AP and get IP through DHCP.
4. After successful connection, Initiate Ping to Access point IP address (default IP address stored is "192.168.10.10" from connected station).
5. Redpine device will respond with ping response for the received Ping Request.



```
Administrator: C:\Windows\system32\cmd.exe - ping 192.168.10.10 -t
Microsoft Windows [Version 6.1.7600]
Copyright <c> 2009 Microsoft Corporation. All rights reserved.

C:\Users\CPU-248>ping 192.168.10.10 -t

Pinging 192.168.10.10 with 32 bytes of data:
Reply from 192.168.10.10: bytes=32 time=117ms TTL=128
Reply from 192.168.10.10: bytes=32 time=24ms TTL=128
Reply from 192.168.10.10: bytes=32 time=98ms TTL=128
Reply from 192.168.10.10: bytes=32 time=79ms TTL=128
Reply from 192.168.10.10: bytes=32 time=76ms TTL=128
Reply from 192.168.10.10: bytes=32 time=54ms TTL=128
Reply from 192.168.10.10: bytes=32 time=27ms TTL=128
Reply from 192.168.10.10: bytes=32 time=32ms TTL=128
Reply from 192.168.10.10: bytes=32 time=38ms TTL=128
```

## 8.25 TCP Client Socket Example

### 8.25.1 TCP Protocol Overview

TCP (Transmission control protocol) is a connection-oriented protocol for transferring data reliably in either direction between a pair of users. When TCP client sends data to the server, it requires an acknowledgement in return. If an acknowledgement is not received, TCP automatically re-transmit the data and waits for a longer period of time till timeout. After time out socket would be closed. To open a connection, a message is sent with SYN(synchronize) flag. To close a connection, a message is sent with FIN(finish) flag. Urgent messages may also be sent by selecting the PSH(push) flag as a protocol parameter.

### 8.25.2 Example Overview

## Overview

The TCP client application demonstrates how to open and use a standard TCP client socket and sends data to TCP server socket.

## Sequence of Events

This Application explains user how to:

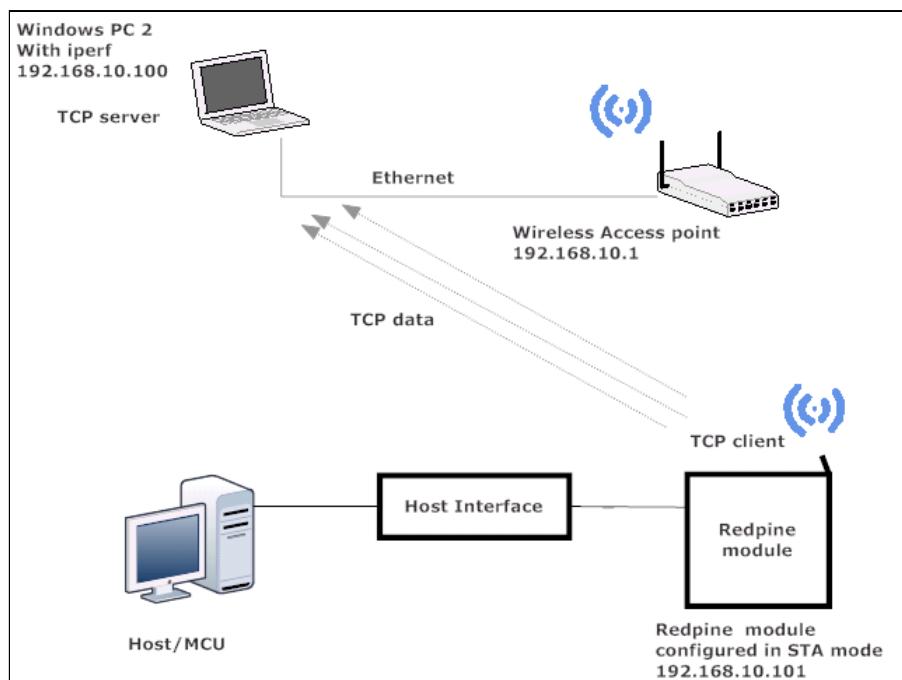
- Connect the Device to an Access point and get IP address through DHCP
- Open TCP Server socket at Access point using iperf application.
- Open TCP client socket in device and establish TCP connection with TCP server opened in remote peer.
- Send data from Redpine device to remote peer using opened TCP socket.

### 8.25.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- WPS supported Access Point
- Windows PC2
- Redpine module
- A TCP server application running on the Windows PC2 (This example uses iperf for windows )



**Figure 1: Setup Diagram**

#### 8.25.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open rsi\_tcp\_client.c file and update / modify the following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID " <ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels.

```
#define CHANNEL_NO 0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK "<psk>"
```

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT 5001
```

**SERVER\_PORT** port refers remote TCP server port number which is opened in windows PC2.

```
#define SERVER_PORT 5001
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with TCP server socket.  
IP address should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.100" as IP address, update the macro **DEVICE\_IP** as **0x640AA8C0**.

```
#define SERVER_IP_ADDRESS 0x640AA8C0
```

**NUMEBR\_OF\_PACKETS** refers how many packets to send from device to TCP server

```
#define NUMBER_OF_PACKETS 1000
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 8000
```

#### To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE 1
```

#### Note:

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP 0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY 0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK 0x00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

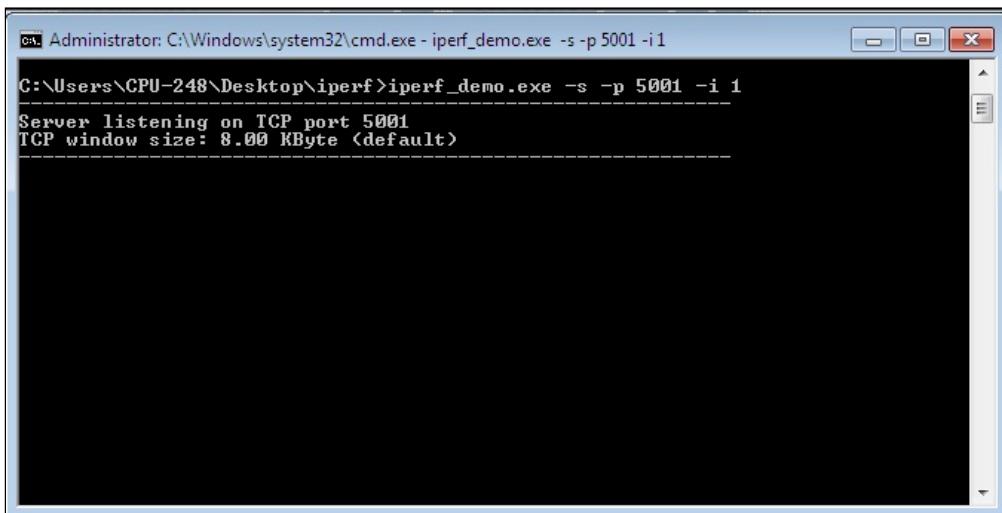
#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	0
#define RSI_CUSTOM_FEATURE_BIT_MAP	
#define RSI_BAND	RSI_BAND_2P4GHZ

**Note:**

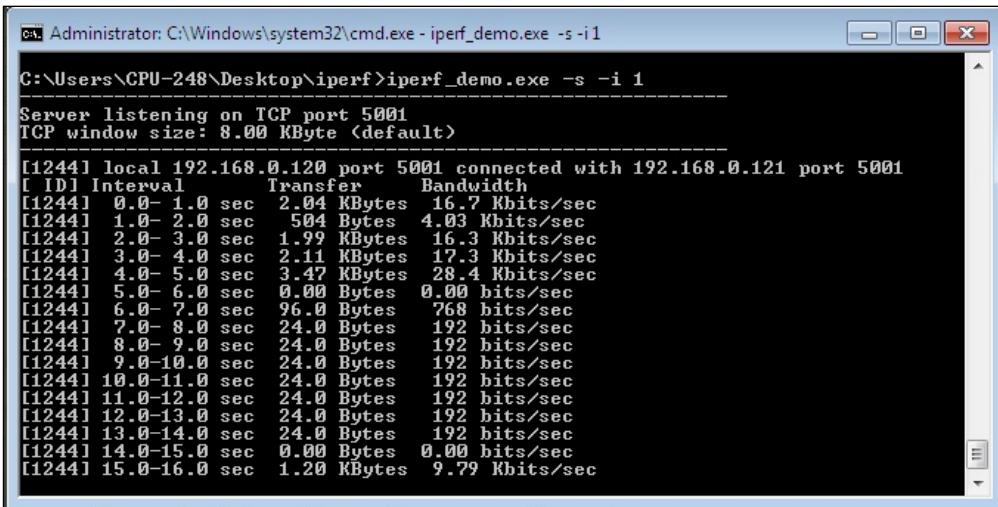
rsi\_wlan\_config.h file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. Open TCP server application using iperf application in Windows PC2 which is connected to Access point through LAN.
3. Users can download application from the link below,  
<https://iperf.fr/iperf-download.php#windows>  
**iperf\_demo.exe -s -p <SERVER\_PORT> -i 1**



4. After program gets executed, Redpine Device would scan and connect to Access point and get IP.
5. After successful connection, device STA connects to TCP server socket opened on Windows PC2 using TCP client socket and sends configured **NUMBER\_OF\_PACKETS** to remote TCP server. Please find below image for reception of TCP data on TCP server.



```
C:\ Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -i 1
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
[12441] local 192.168.0.120 port 5001 connected with 192.168.0.121 port 5001
[ ID] Interval Transfer Bandwidth
[12441] 0.0- 1.0 sec 2.04 KBytes 16.7 Kbytes/sec
[12441] 1.0- 2.0 sec 504 Bytes 4.03 Kbits/sec
[12441] 2.0- 3.0 sec 1.99 KBytes 16.3 Kbytes/sec
[12441] 3.0- 4.0 sec 2.11 KBytes 17.3 Kbits/sec
[12441] 4.0- 5.0 sec 3.47 KBytes 28.4 Kbits/sec
[12441] 5.0- 6.0 sec 0.00 Bytes 0.00 bits/sec
[12441] 6.0- 7.0 sec 96.0 Bytes 768 bits/sec
[12441] 7.0- 8.0 sec 24.0 Bytes 192 bits/sec
[12441] 8.0- 9.0 sec 24.0 Bytes 192 bits/sec
[12441] 9.0-10.0 sec 24.0 Bytes 192 bits/sec
[12441] 10.0-11.0 sec 24.0 Bytes 192 bits/sec
[12441] 11.0-12.0 sec 24.0 Bytes 192 bits/sec
[12441] 12.0-13.0 sec 24.0 Bytes 192 bits/sec
[12441] 13.0-14.0 sec 24.0 Bytes 192 bits/sec
[12441] 14.0-15.0 sec 0.00 Bytes 0.00 bits/sec
[12441] 15.0-16.0 sec 1.20 KBytes 9.79 Kbits/sec
```

## 8.26 TCP Client Socket over SSL Application with Multiple TLS Versions.

### 8.26.1 SSL Overview

SSL stands for Secure Sockets Layer. SSL is the standard security technology for establishing an Encrypted link between a web server and a browser. This link ensures that all data passed between the web servers and the browsers remain Private & Integral. Data encryption, Server authentication, Message integrity, Optional client authentication for a TCP/IP connection are the main objectives of SSL protocol. In this user can open multiple sockets with Different TLS versions.

### 8.26.2 Application Overview

#### Overview

This application demonstrates how to open and use a standard TCP client socket with secure connection using SSL and sends data on socket.

#### Sequence of Events

This Application explains user how to:

- Connect the Device to an Access point and get IP address through DHCP
- Open TCP Server socket over SSL at Access point using openssl.
- Establish TCP connection over SSL with TCP server opened on remote peer.
- Send TCP data from Redpine device to remote device

### 8.26.3 Application Setup

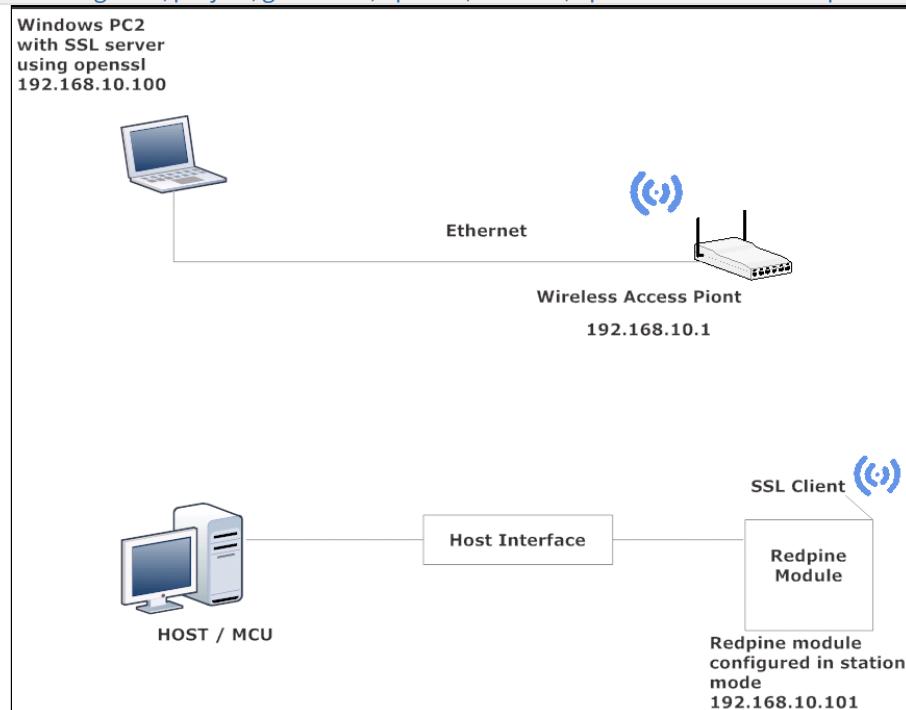
The Redpine device in its many variants supports SPI and UART interfaces. Depending on the interface used, the required set up is as below:

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Windows PC1 with CooCox IDE Spansion (MB9BF568NBGL) micro controller
- Windows PC2

- Redpine module
- TCP server over SSL running in Windows PC2 (This application uses OpenSSL to create TCP server over SSL)

**Note:** Please download openssl for windows from below link,  
[\\*http://ufpr.dl.sourceforge.net/project/gnuwin32/openssl/0.9.8h-1/openssl-0.9.8h-1-bin.zip\\*](http://ufpr.dl.sourceforge.net/project/gnuwin32/openssl/0.9.8h-1/openssl-0.9.8h-1-bin.zip)



**Figure 1: Setup Diagram**

#### 8.26.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_ssl\_client.c** file and update/modify following macros:

**SSID** refers to the name of the Access point.

```
#define SSID "ap_name"
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK NULL
```

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT1 <local  
port>
```

**SERVER\_PORT** port refers remote TCP server port number which is opened in Windows PC2.

```
#define SERVER_PORT1 <remote  
port>
```

**DEVICE\_PORT2** refers another TCP client port number

```
#define DEVICE_PORT2 <local  
port>
```

**SERVER\_PORT2** refers another remote TCP server port number which is opened in Windows PC2.

```
#define SERVER_PORT2 <remote  
port>
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with TCP server socket.  
IP address should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.100" as IP address, update the macro **DEVICE\_IP** as **0x640AA8C0**.

```
#define SERVER_IP_ADDRESS 0x640AA8C0
```

**NUMBER\_OF\_PACKETS** refers how many packets to receive from TCP client

```
#define NUMBER_OF_PACKETS <no of  
packets>
```

Application memory length which is required by the driver

#define GLOBAL_BUFF_LEN	8000
-------------------------	------

**LOAD\_CERTIFICATE** refers to load certificates into flash  
0-Already certificates are there in flash so no need to laod.  
1-Certicates will load into flash.

#define LOAD_CERTIFICATE	0
--------------------------	---

Note:If certificates are not there in flash then ssl handshake will failure.

To configure IPaddress

**DHCP\_MODE**refers whether IP address configured through DHCP or STATIC

#define DHCP_MODE	1
-------------------	---

**Note:**

If userwantsto configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**,**GATEWAY**and**NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY**and**NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

#define DEVICE_IP	0X0A0AA8C0
-------------------	------------

IP address of the gateway should also be in long format and in little endian byte order.

**Example:** To configure "192.168.10.1" as Gateway, update the macro GATEWAY as **0x010AA8C0**

#define GATEWAY	0x010AA8C0
-----------------	------------

#define NETMASK	0x00FFFFFF
-----------------	------------

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK	0x00FFFFFF
-----------------	------------

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_CLIENT | TCP_IP_FEAT_SSL)
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENSION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_SSL VERSIONS SUPPORT

#define RSI_BAND
RSI_BAND_2P4GHZ
```

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

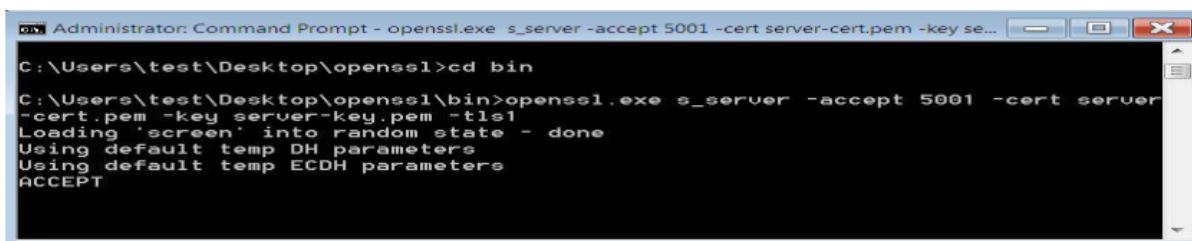
1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. In Windows PC2 which is connected to AP through LAN, Download the Openssl package from the above mentioned link and run Two SSL servers by giving the following command,  
`openssl.exe s_server -accept<SERVER_PORT> -cert <server_certificate_file_path> -key <server_key_file_path> -tls<tls_version>`

**Example: openssl.exe s\_server -accept 5001 -cert server-cert.pem -key server-key.pem**

**Example: openssl.exe s\_server -accept 5002 -cert server-cert.pem -key server-key.pem**

**Note:**

All the certificates are given in the release package. Path: sapis/examples/utilities/certificate



```
Administrator: Command Prompt - openssl.exe s_server -accept 5001 -cert server-cert.pem -key se...
C:\Users\test\Desktop\openssl>cd bin
C:\Users\test\Desktop\openssl\bin>openssl.exe s_server -accept 5001 -cert server
-cert.pem -key server-key.pem -tls1
Loading 'screen' into random state - done
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
```

3. After the program gets executed, Redpine Device would be connected to Accesspoint having the configuration same that of in the application and get IP.
4. The Device which is configured as SSL client will connect to remote SSL server and sends number of packets configured in **NUMBER\_OF\_PACKETS**.

**Note:**

Please download openssl for windows from below link,

\*<http://ufpr.dl.sourceforge.net/project/gnuwin32/openssl/0.9.8h-1/openssl-0.9.8h-1-bin.zip>\*

## 8.27 TCP Client Socket over SSL Example

### 8.27.1 SSL Overview

SSL stands for Secure Sockets Layer. SSL is the standard security technology for establishing an Encrypted link between a web server and a browser. This link ensures that all data passed between the web servers and the browsers remain Private & Integral.

Data encryption, Server authentication, Message integrity, Optional client authentication for a TCP / IP connection are the main objectives of SSL protocol.

### 8.27.2 Example Overview

#### Overview

This application demonstrates how to open and use a standard TCP client socket with secure connection using SSL and sends data on socket.

#### Sequence of Events

This Application explains user how to:

- Connect the device to an Access point and get IP address through DHCP
- Open TCP Server socket over SSL at Access point using OpenSSL
- Establish TCP connection over SSL with TCP server opened on remote peer
- Send TCP data from WiSeConnect device to remote device

### 8.27.3 Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

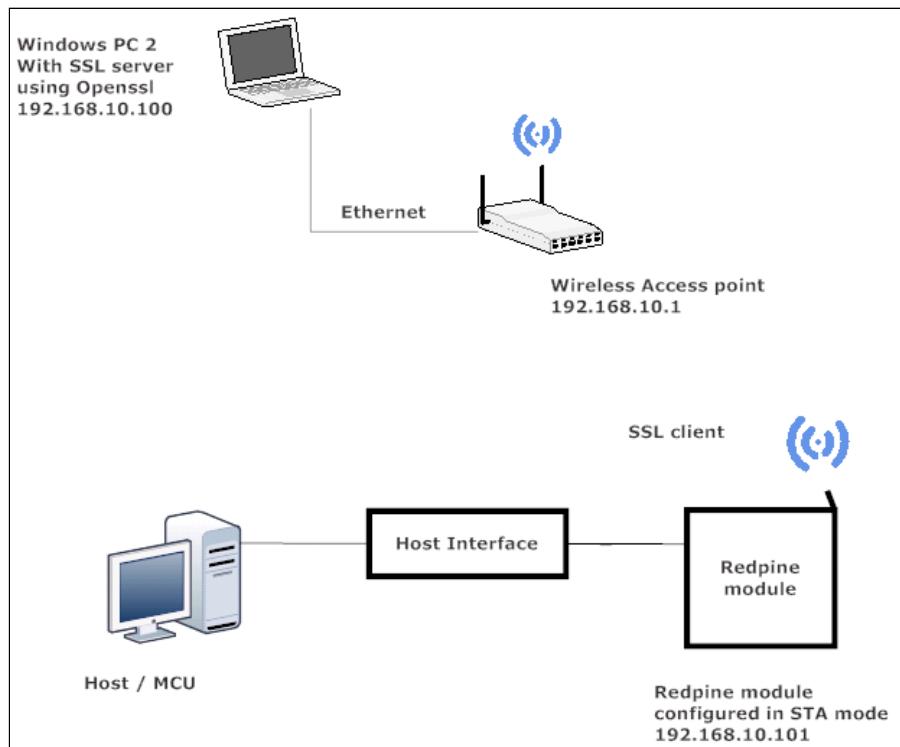
#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC2 with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine Module
- WiFi Access point
- Windows PC2
- TCP server over SSL running in Windows PC2 (This application uses OpenSSL to create TCP server over SSL)

**Note:**

Please download OpenSSL for windows from the below link:

<http://ufpr.dl.sourceforge.net/project/gnuwin32/openssl/0.9.8h-1/openssl-0.9.8h-1-bin.zip>



**Figure 1: Setup Diagram**

#### 8.27.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_ssl\_client.c** file and update / modify the following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE
```

```
RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK
```

```
"<psk>"
```

#### To Load certificate

```
#define LOAD_CERTIFICATE
```

```
1
```

If **LOAD\_CERTIFICATE** set to 1, application will load certificate which is included using `rsi_wlan_set_certificate` API.

By default, application loading "**cacert.pem**" certificate if **LOAD\_CERTIFICATE** enable. In order to load different certificate, user has to follow the following steps:

- `rsi_wlan_set_certificate` API expects the certificate in the form of lineararray. So, convert the pemcertificate into linear array form using python script provided in the release package "**sapis/examples/utilities/certificates/certificate\_script.py**".

#### Example:

If the certificate is wifi-user.pem, enter the command in the following way:

```
python certificate_script.py ca-cert.pem
```

The script will generate wifiuser.pem in which one linear array named **cacert** contains the certificate.

- After the conversion of certificate, update `rsi_ssl_client.c` source file by including the certificate file and also by providing the required parameters to `rsi_wlan_set_certificate` API.

Once the certificate loads into the device, it will write into the device flash. So, user need not load certificate for every boot up unless certificate change. So, define **LOAD\_CERTIFICATE** as 0, if certificate is already present in the device.

#### Note:

All the certificates are given in the release package.

Path: `sapis/examples/utilities/certificates`

Enable **SSL** macro to open SSL socket over TCP.

```
#define SSL
```

```
1
```

**DEVICE\_PORT** port refers SSL client port number

```
#define DEVICE_PORT
```

```
5001
```

**SERVER\_PORT** port refers remote SSL server port number which is opened in Windows PC2

```
#define SERVER_PORT
```

```
5001
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with SSL server socket.  
IP address should be in long format and in little endian byte order.

**Example:** To configure "192.168.0.100" as remote IP address, update the macro **SERVER\_IP\_ADDRESS** as **0x6400A8C0**.

```
#define SERVER_IP_ADDRESS 0x6400A8C0
```

**NUMBER\_OF\_PACKETS** refers how many packets to send from TCP client

```
#define NUMBER_OF_PACKETS <no of packets>
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 8000
```

**To configure IP address**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC.

```
#define DHCP_MODE 1
```

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP 0X0A0AA8C0
```

IP address of the gateway should also be in longformat and in little endianbyte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY 0X010AA8C0
```

IP address of the network mask should also be in longformat and in little endianbyte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK          0x00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE      RSI_DISABLE
#define RSI_FEATURE_BIT_MAP    FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS      RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_CLIENT|TCP_IP_FEAT_SSL)
#define RSI_CUSTOM_FEATURE_BIT_MAP   0
#define RSI_BAND                RSI_BAND_2P4GHZ
```

**Note:**

rsi\_wlan\_config.h file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. In Windows PC2 which is connected to AP through LAN, Download the Openssl package from above mentioned link and run SSL server by giving following command,

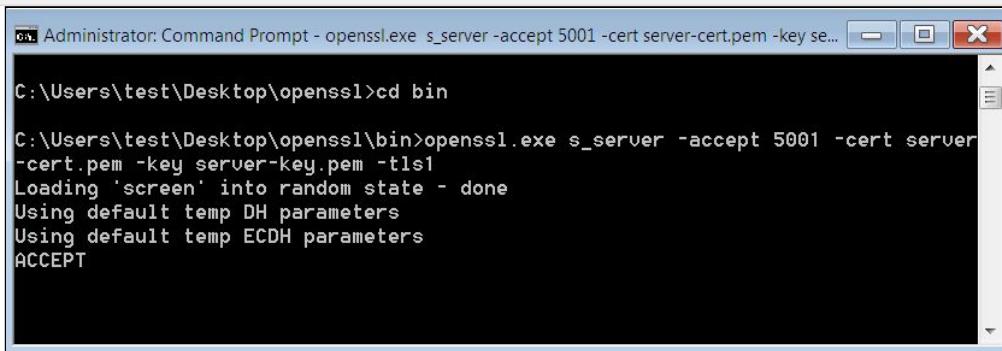
Open ssl.exe s\_server -accept<SERVER\_PORT> -cert <server\_certificate\_file\_path> -key <server\_key\_file\_path> -tls<tls\_version>

**Example:** open ssl.exe s\_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1.

**Note:**

All the certificates are given in the release package.

Path: sapis/examples/utilities/certificates



```
Administrator: Command Prompt - openssl.exe s_server -accept 5001 -cert server-cert.pem -key se...
C:\Users\test\Desktop\openssl>cd bin

C:\Users\test\Desktop\openssl\bin>openssl.exe s_server -accept 5001 -cert server
-cert.pem -key server-key.pem -tls1
Loading 'screen' into random state - done
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
```

3. After the program gets executed, the Redpine device would be connected to the access point having the configuration same as that of in the application and get IP.

4. The device which is configured as SSL client will connect to remote SSL server and sends number of packets configured in **NUMBER\_OF\_PACKETS**.

## 8.28 TCP Server Socket Example

### 8.28.1 TCP Protocol Overview

TCP(Transmission control protocol) is a connection-oriented protocol for transferring data reliably in either direction between a pair of users.

TCP server waits for the connections from TCP clients and accepts Incoming TCP connections.

## 8.28.2 Example Overview

## Overview

The TCP server application demonstrates how to open and use a standard TCP server socket and receives data from TCP client socket.

## Sequence of Events

This Application explains user how to:

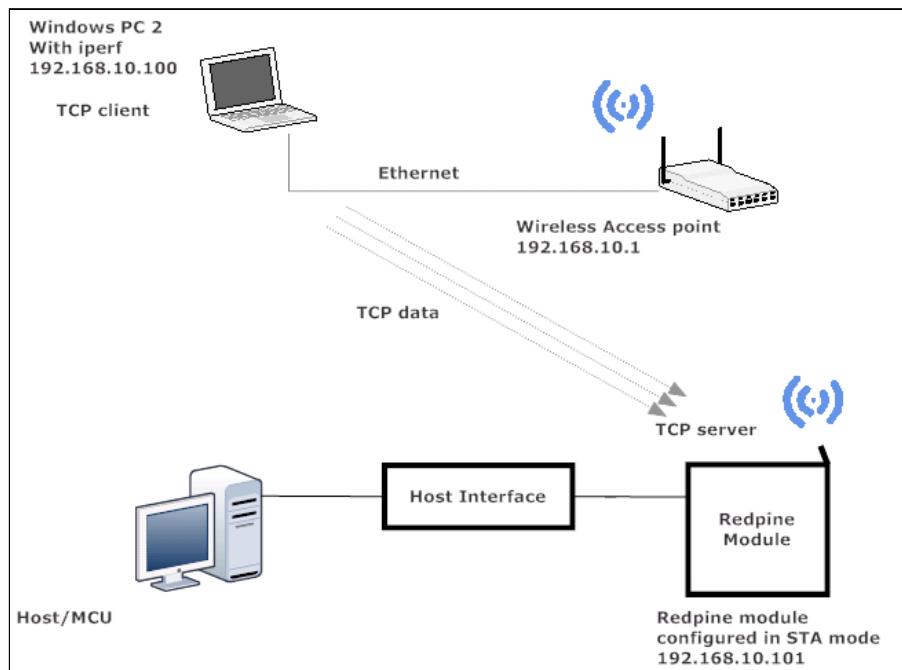
- Connect the Device to an Access point and get IP address through DHCP
  - Open TCP server socket in device
  - Connect to TCP server socket opened in device from remote peer using TCP client socket.
  - Receive data from TCP client socket.

### 8.28.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC2 with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine Module
- WiFi Access point
- Windows PC2
- TCP client application running in Windows PC2 (This application uses iperf application to open TCP client socket)



**Figure 1: Setup Diagram**

### 8.28.4 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_tcp\_server.c** file and update/modify following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration are :

**RSI\_OPEN** - For OPEN security mode

**RSI\_WPA** - For WPA security mode

**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE    RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK             "<psk>"
```

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT     5001
```

**Receive data length**

```
#define RECV_BUFFER_SIZE <recv_buf_size>
```

**NUMBER\_OF\_PACKETS** refers how many packets to receive from TCP client

```
#define NUMBER_OF_PACKETS <no of packets>
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 8000
```

**To configure IP address**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE 0
```

**Note:** Configure STATIC IP to WiSeConnect device. So that user knows the IP address of WiSeConnect device to establish TCP connection from remote peer. In case of DHCP, User has to know the assigned IP by parsing IPCONF response.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

**Example:** To configure "192.168.0.101" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP 0X6500A8C0
```

IP address of the gateway should also be in longformat and in little endianbyte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY 0X0100A8C0
```

IP address of the network mask should also be in longformat and in little endianbyte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK 0x00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	0
#define RSI_BAND	RSI_BAND_2P4GHZ

**Note:**

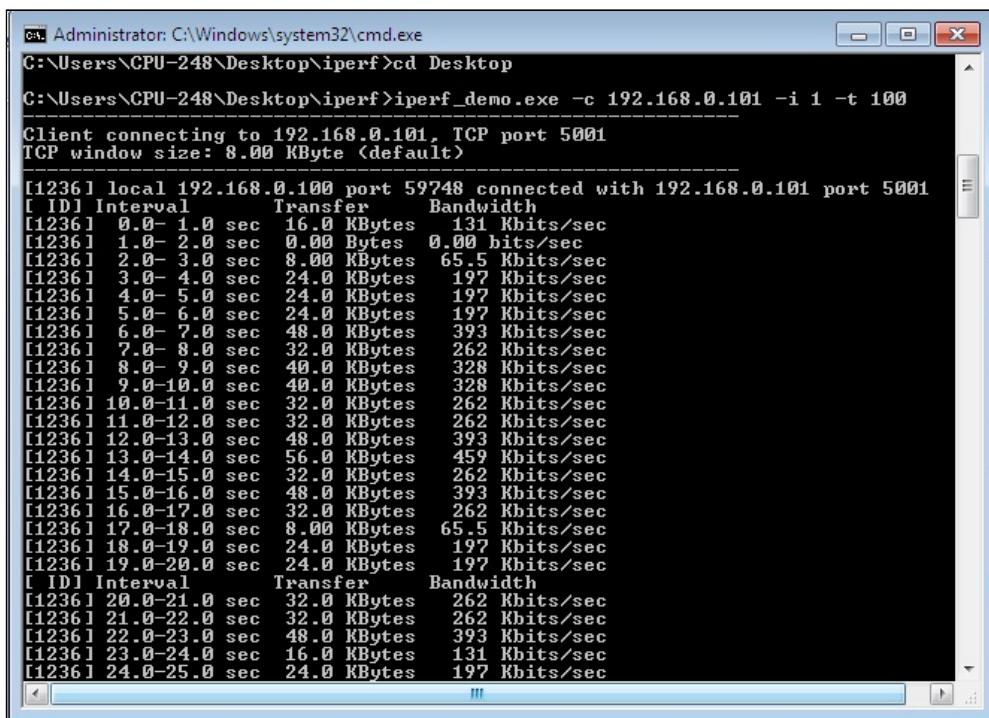
**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. After the program gets executed, Redpine device will be connected to the access point having the configuration same as that of in the application and get IP.

2. Open TCP client using iperf from Windows PC2 and connect to TCP server opened on the device on port number **DEVICE\_PORT** using the following command:

```
iperf_demo.exe -c <DEVICE_IP> -p <DEVICE_PORT> -i 1 -t 1000
```



```
Administrator: C:\Windows\system32\cmd.exe
C:\Users\CPU-248\Desktop\iperf>cd Desktop
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -c 192.168.0.101 -i 1 -t 100
Client connecting to 192.168.0.101, TCP port 5001
TCP window size: 8.00 KByte (default)

[1236] local 192.168.0.100 port 59748 connected with 192.168.0.101 port 5001
[ ID] Interval Transfer Bandwidth
[1236] 0.0- 1.0 sec 16.0 KBytes 131 Kbits/sec
[1236] 1.0- 2.0 sec 0.00 Bytes 0.00 bits/sec
[1236] 2.0- 3.0 sec 8.00 KBytes 65.5 Kbits/sec
[1236] 3.0- 4.0 sec 24.0 KBytes 197 Kbits/sec
[1236] 4.0- 5.0 sec 24.0 KBytes 197 Kbits/sec
[1236] 5.0- 6.0 sec 24.0 KBytes 197 Kbits/sec
[1236] 6.0- 7.0 sec 48.0 KBytes 393 Kbits/sec
[1236] 7.0- 8.0 sec 32.0 KBytes 262 Kbits/sec
[1236] 8.0- 9.0 sec 40.0 KBytes 328 Kbits/sec
[1236] 9.0-10.0 sec 40.0 KBytes 328 Kbits/sec
[1236] 10.0-11.0 sec 32.0 KBytes 262 Kbits/sec
[1236] 11.0-12.0 sec 32.0 KBytes 262 Kbits/sec
[1236] 12.0-13.0 sec 48.0 KBytes 393 Kbits/sec
[1236] 13.0-14.0 sec 56.0 KBytes 459 Kbits/sec
[1236] 14.0-15.0 sec 32.0 KBytes 262 Kbits/sec
[1236] 15.0-16.0 sec 48.0 KBytes 393 Kbits/sec
[1236] 16.0-17.0 sec 32.0 KBytes 262 Kbits/sec
[1236] 17.0-18.0 sec 8.00 KBytes 65.5 Kbits/sec
[1236] 18.0-19.0 sec 24.0 KBytes 197 Kbits/sec
[1236] 19.0-20.0 sec 24.0 KBytes 197 Kbits/sec
[ ID] Interval Transfer Bandwidth
[1236] 20.0-21.0 sec 32.0 KBytes 262 Kbits/sec
[1236] 21.0-22.0 sec 32.0 KBytes 262 Kbits/sec
[1236] 22.0-23.0 sec 48.0 KBytes 393 Kbits/sec
[1236] 23.0-24.0 sec 16.0 KBytes 131 Kbits/sec
[1236] 24.0-25.0 sec 24.0 KBytes 197 Kbits/sec
```

3. The Redpine device will receive the number of packets configured in **NUMBER\_OF\_PACKETS** from iperf TCP client and closes the socket.

## 8.29 Throughput Example

### 8.29.1 Example Overview

#### Overview

Throughput is the rate of production or the rate at which something can be processed. When used in the context of communication networks, such as Ethernet or packet radio, throughput or network throughput is the rate of successful message delivery over a communication channel. This application will demonstrate the throughput measurement.

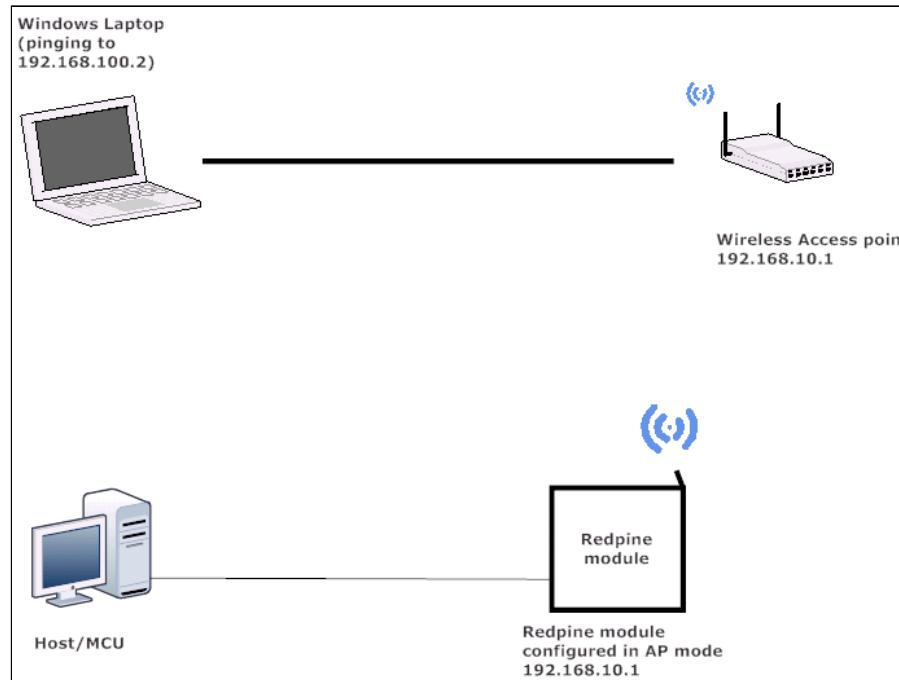
#### 8.29.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART / USB-CDC / SPI / USB) in case of WiSeConnect

- Redpine Module
- Windows Laptop with application program like iperf
- Access Point



**Figure 1: Setup Diagram**

**Description:**

This application can be used to configure Redpine module in UDP client / server or TCP client / server. To measure throughput, following configuration can be applied.

1. To measure UDP Tx throughput, module should configured as UDP client.
2. To measure UDP Rx throughput, module should configured as UDP server.
3. To measure TCP Tx throughput, module should configured as TCP client.
4. To measure TCP Rx throughput, module should configured as TCP server.

#### 8.29.3 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_throughput\_app.c** file and update / modify the following macros,  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which AP would be started

#define CHANNEL\_NO

11

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities.  
Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

#define SECURITY\_TYPE

RSI\_OPEN

**PSK** refers to the secret key if the Access point is to be configured in WPA/WPA2 security modes.

#define PSK

"<psk>"

Enable / Disable DHCP mode  
1-Enables DHCP mode (gets the IP from DHCP server)  
0-Disables DHCP mode

#define DHCP\_MODE

<dhcp mode>

#### To configure static IP address

IP address to be configured to the device should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

#define DEVICE\_IP

0X010AA8C0

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

#define GATEWAY

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK

0x00FFFFFF

2. To establish UDP/TCP connection and transfer/receive data to the remote socket configure the below macros  
Internal device port number

#define PORT\_NUM

<Local\_port>

Port number of the remote server

#define SERVER\_PORT  
<Remote\_port\_num>

IP address of the remote server

#define SERVER\_IP\_ADDRESS

0x640AA8C0

Application memory length which is required by the driver

#define GLOBAL\_BUFF\_LEN

8000

Application can use receive buffer size of 1400

#define BUFF\_SIZE

1400

Application can select throughput type as UDP Tx, UDP Rx, TCP Tx or TCP Rx. Following is macro need to use.

#define THROUGHPUT\_TYPE

UDP\_TX

Following is macro used for throughput type selection

#define UDP\_TX  
#define UDP\_RX  
#define UDP\_TX  
#define UDP\_RX

0  
1  
2  
3

**Note:**

In AP mode, configure same IP address for both DEVICE\_IP and GATEWAY macros.

3. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE DISABLE
#define RSI_FEATURE_BIT_MAP
(FEAT_SECURITY_OPEN | FEAT_AGGREGATION )
#define RSI_TCP_IP_BYPASS DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_CLIENT)
#define RSI_CUSTOM_FEATURE_BIT_MAP 0
#define RSI_BAND
RSI_BAND_2P4GHZ
```

**Note:**

rsi\_wlan\_config.h file is already set with desired configuration in respective example folders user need not change for each example.

### Executing the Application

1. Connect Redpine device to the Windows PC running Cocoox IDE.
2. Configure the macros in the files **rsi\_throughput\_app.c rsi\_wlan\_config.h**
3. To measure throughput, following configuration can be applied.
  - a) To measure UDP Tx throughput, module should be configured as UDP client. Open UDP server at remote port

```
iperf.exe -s -u -p <SERVER_PORT> -i 1
```

- b) To measure UDP Rx throughput, module should be configured as UDP server. Open UDP client at remote port

```
iperf.exe -c <Module_IP> -u -p <Module_Port> -i 1 -b <Bandwidth>
```

- c) To measure TCP Tx throughput, module should be configured as TCP client. Open TCP server at remote port.

```
iperf.exe -s -p <SERVER_PORT> -i 1
```

- d) To measure TCP Rx throughput, module should be configured as TCP server. Open TCP client at remote port.

```
iperf.exe -c <Module_IP> -p <module_PORT> -i 1
```

4. To measure throughput, following configuration can be applied.
5. Build and launch the application.
6. After the program gets executed, the device would be connected to Access point having the configuration same

---

as that of in the application and get .

7. The Device which is configured as UDP / TCP server / client will connect to iperf server / client and sends / receives data continuously. It will print the throughput per second.

## 8.30 UDP Client Socket Example

### 8.30.1 UDP Protocol Overview

UDP(USER Datagram protocol) is a connection less and non-stream oriented protocol for transferring data in either direction between a pair of users. In UDP there is no guarantee that the messages or packets sent would reach at all. No handshake in UDP Protocol because it is connection less protocol.

### 8.30.2 Example Overview

#### Overview

The UDP client application demonstrates how to open and use a standard UDP client socket and sends data to UDP server socket. Once it is configured as UDP client, it can establish and maintain a network conversation by means of application program for exchanging of data.

#### Sequence of Events

This Application explains user how to:

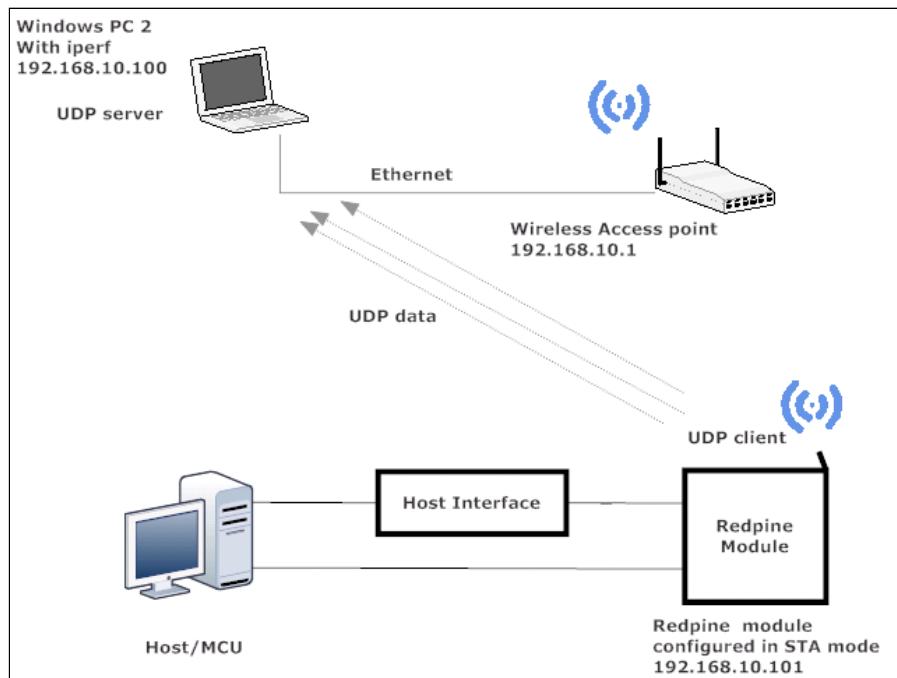
- Connect the Device to an Access point and get IP address through DHCP
- Open UDP Server socket at Access point using iperf application.
- Open UDP client socket in device
- Send data from Redpine device to remote peer using opened UDP socket.

### 8.30.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with Keil or IAR IDE in case of WiSeMCU in case of WiSeMCU
- Windows PC with Dev-C++ IDE / SPI/ USB) in case of WiSeConnect
- Redpine Module
- Wi-Fi Access point
- Windows PC2
- UDP server application running in Windows PC2 (This application uses iperf application to open UDP server socket)



**Figure 1: Setup Diagram**

#### 8.30.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_udp\_client.c** file and update/modify following macros,  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0, device will scan all channels

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK and WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE    RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK           "<psk>"
```

**DEVICE\_PORT** port refers UDP client port number

```
#define DEVICE_PORT      5001
```

**SERVER\_PORT** port refers remote UDP server which is opened in windows PC2.

```
#define SERVER_PORT      5001
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with TCP server socket.  
IP address should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.100" as IP address, update the macro **DEVICE\_IP** as **0x640AA8C0**.

```
#define SERVER_IP_ADDRESS    0x640AA8C0
```

**NUMBER\_OF\_PACKETS** refer how many packets to send from device to UDP server.

```
#define NUMBER_OF_PACKETS   1000
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN     8000
```

**To configure IP address:**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE          1
```

**Note:**

If the user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If the user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address which is to be configured to the device in STA mode should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

#define DEVICE\_IP

0X0A0AA8C0

IP address of the gateway should also be in long format and in little endian byte order.

**Example:** To configure "192.168.10.1" as Gateway, update the macro GATEWAY as **0x010AA8C0**

#define GATEWAY

0x010AA8C0

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro NETMASK as **0x00FFFFFF**

#define NETMASK

0x00FFFFFF

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	0

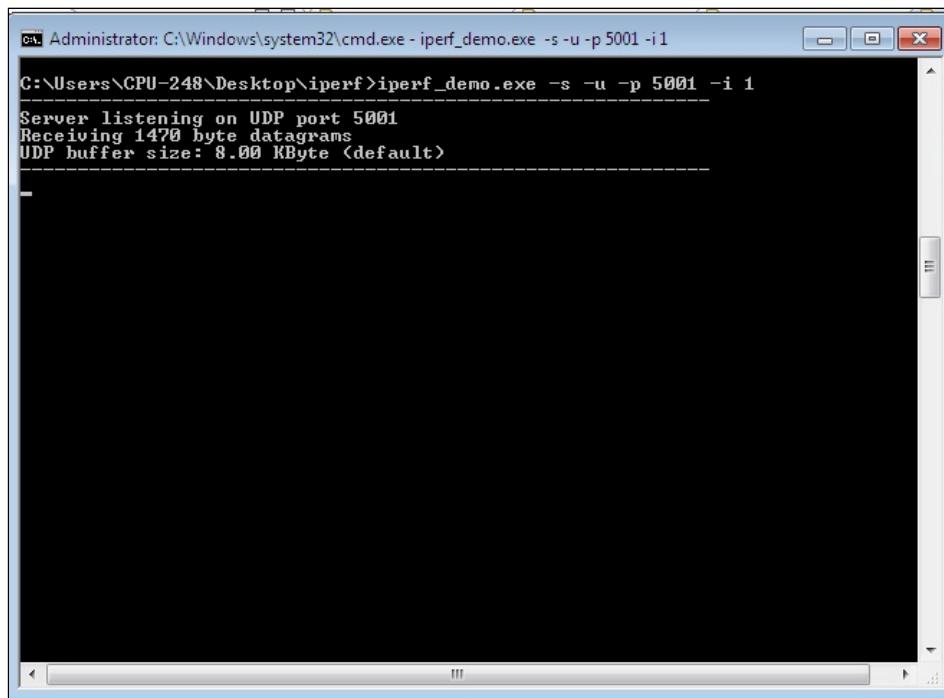
**Note:**

rsi\_wlan\_config.h file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the Access point in OPEN / WPA-PSK/WPA2-PSK mode in order to connect Redpine device in STA mode.
2. Open UDP server application using iperf application in Windows PC2 which is connected to Access point through LAN.

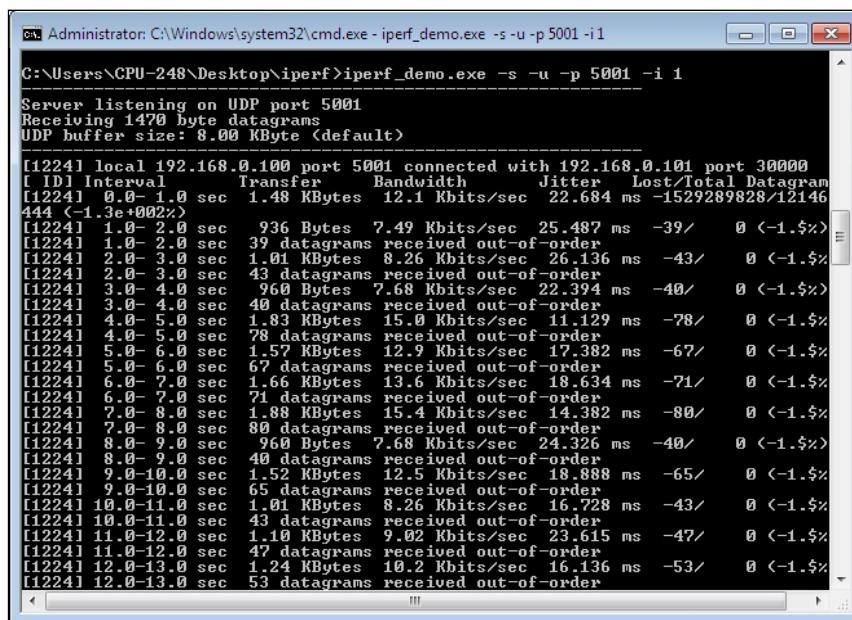
**iperf\_demo.exe -s -u -p <SERVER\_PORT> -i 1**



```
C:\> Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -u -p 5001 -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -u -p 5001 -i 1
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
```

3. After program gets executed, The Redpine device would scan and connect to Access point and get IP.

4. After successful connection, the device STA connects to UDP server socket opened on Windows PC2 using UDP client socket and sends configured **NUMBER\_OF\_PACKETS** to remote UDP server. Please refer the below image for reception of UDP data on UDP server.



```
C:\> Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -u -p 5001 -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -u -p 5001 -i 1
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
[1224] local 192.168.0.100 port 5001 connected with 192.168.0.101 port 30000
[1224] 0.0-1.0 sec 1.48 KBytes 12.1 Kbits/sec 22.684 ms -1529289828/12146
444 <-1.3e+002>
[1224] 1.0- 2.0 sec 936 Bytes 7.49 Kbytes/sec 25.487 ms -39/ 0 <-1.%>
[1224] 1.0- 2.0 sec 39 datagrams received out-of-order
[1224] 2.0- 3.0 sec 1.01 KBytes 8.26 Kbits/sec 26.136 ms -43/ 0 <-1.%>
[1224] 2.0- 3.0 sec 43 datagrams received out-of-order
[1224] 3.0- 4.0 sec 960 Bytes 7.68 Kbits/sec 22.394 ms -40/ 0 <-1.%>
[1224] 4.0- 5.0 sec 1.83 KBytes 15.0 Kbits/sec 11.129 ms -78/ 0 <-1.%>
[1224] 4.0- 5.0 sec 78 datagrams received out-of-order
[1224] 5.0- 6.0 sec 1.57 KBytes 12.9 Kbits/sec 17.382 ms -67/ 0 <-1.%>
[1224] 5.0- 6.0 sec 67 datagrams received out-of-order
[1224] 6.0- 7.0 sec 1.66 KBytes 13.6 Kbits/sec 18.634 ms -71/ 0 <-1.%>
[1224] 6.0- 7.0 sec 71 datagrams received out-of-order
[1224] 7.0- 8.0 sec 1.88 KBytes 15.4 Kbits/sec 14.382 ms -80/ 0 <-1.%>
[1224] 7.0- 8.0 sec 80 datagrams received out-of-order
[1224] 8.0- 9.0 sec 960 Bytes 7.68 Kbits/sec 24.326 ms -40/ 0 <-1.%>
[1224] 8.0- 9.0 sec 40 datagrams received out-of-order
[1224] 9.0-10.0 sec 1.52 KBytes 12.5 Kbits/sec 18.888 ms -65/ 0 <-1.%>
[1224] 9.0-10.0 sec 65 datagrams received out-of-order
[1224] 10.0-11.0 sec 1.01 KBytes 8.26 Kbits/sec 16.728 ms -43/ 0 <-1.%>
[1224] 10.0-11.0 sec 43 datagrams received out-of-order
[1224] 11.0-12.0 sec 1.10 KBytes 9.02 Kbits/sec 23.615 ms -47/ 0 <-1.%>
[1224] 11.0-12.0 sec 47 datagrams received out-of-order
[1224] 12.0-13.0 sec 1.24 KBytes 10.2 Kbits/sec 16.136 ms -53/ 0 <-1.%>
[1224] 12.0-13.0 sec 53 datagrams received out-of-order
```

## 8.31 UDP Sever Socket Example

### 8.31.1 UDP Protocol Overview

UDP (USER Datagram protocol) is a connection less and non-stream oriented protocol for transferring data in either direction between a pair of users.

### 8.31.2 Example Overview

#### Overview

The UDP server application demonstrates how to open and use a standard UDP server socket and receives data on socket sent by remote peer. Once it is configured as UDP server, it can establish and maintain a network conversation by means of application program for exchanging of data.

#### Sequence of Events

This Application explains user how to:

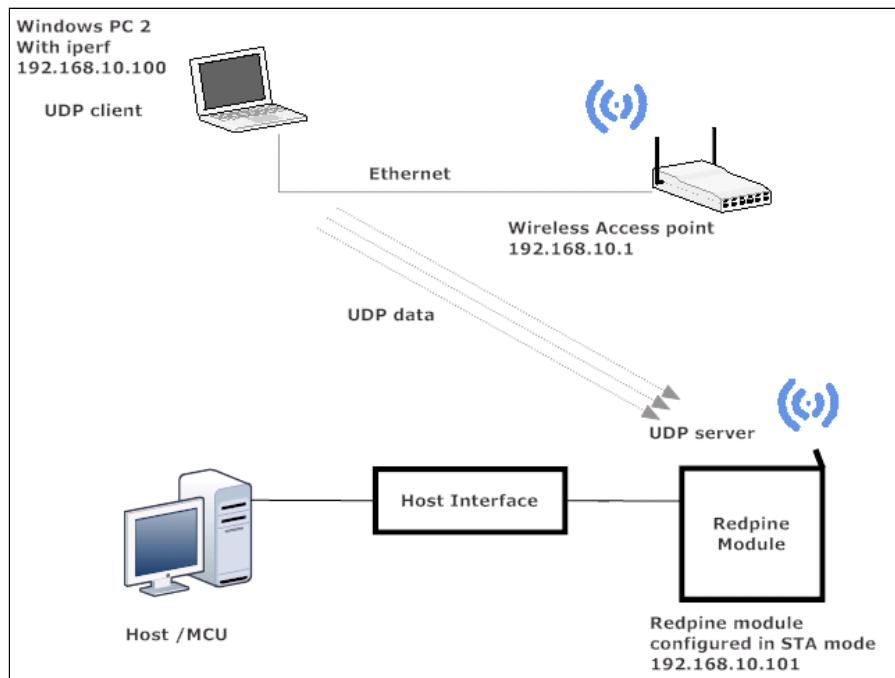
- Connect the Device to an Access point and get IP address through DHCP
- Open UDP server socket in device
- Send UDP data from remote peer to Redpine device on opened port using UDP client socket
- Receive data from UDP client socket.

### 8.31.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with Keil or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module
- WiFi Access point
- Windows PC2
- UDP client application running in Windows PC2 (This application uses iperf application to open UDP client socket)



**Figure 1: Setup diagram**

#### 8.31.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_udp\_server.c** file and update/modify following macros,  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0,device will scan all channels

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE    RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK           "<psk>"
```

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT      5001
```

**Receive data length**

```
#define RECV_BUFFER_SIZE    <recv_buf_size>
```

**NUMBER\_OF\_PACKETS** refers how many packets to receive from UDP client

```
#define NUMBER_OF_PACKETS    <no of packets>
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN       8000
```

**To configure IP address:**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE          0
```

**Note:**

Configure STATIC IP to WiSeConnect device. So that user knows the IP address of WiSeConnect device to establish UDP connection from remote peer. In case of DHCP, User has to know the assigned IP by parsing IPCONF response.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

**Example:** To configure "192.168.0.101" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP          0X6500A8C0
```

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY            0x0100A8C0
```

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK          0x00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

#define CONCURRENT_MODE	RSI_DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS	RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP	
TCP_IP_FEAT_DHCPV4_CLIENT	
#define RSI_CUSTOM_FEATURE_BIT_MAP	0
#define RSI_BAND	RSI_BAND_2P4GHZ

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. After the program gets executed, the Redpine device will be connected to Access point having the configuration as same as that of in the application and get IP.
2. Open UDP client using iperf from windows PC2  
Users can download application from the link below:  
<https://iperf.fr/iperf-download.php#windows>
3. Connect to UDP server opened on Redpine device on port number **DEVICE\_PORT** using the following command:  
**Iperf\_demo.exe -c <DEVICE\_IP> -u -p <DEVICE\_PORT> -i 1 -t 100**
4. Redpine Device will receive the number of packets configured in **NUMBER\_OF\_PACKETS** from iperf UDP client.

```
Administrator: C:\Windows\system32\cmd.exe
C:\Users\CPU-248\Desktop\iperf>
C:\Users\CPU-248\Desktop\iperf_demo.exe -c 192.168.0.101 -u -p 5001 -i 1
t 100
Client connecting to 192.168.0.101, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)

[1240] local 192.168.0.100 port 63266 connected with 192.168.0.101 port 5001
[ ID] Interval Transfer Bandwidth
[1240] 0.0- 1.0 sec 129 KBytes 1.06 Mbits/sec
[1240] 1.0- 2.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 2.0- 3.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 3.0- 4.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 4.0- 5.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 5.0- 6.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 6.0- 7.0 sec 129 KBytes 1.06 Mbits/sec
[1240] 7.0- 8.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 8.0- 9.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 9.0-10.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 10.0-11.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 11.0-12.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 12.0-13.0 sec 129 KBytes 1.06 Mbits/sec
[1240] 13.0-14.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 14.0-15.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 15.0-16.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 16.0-17.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 17.0-18.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 18.0-19.0 sec 129 KBytes 1.06 Mbits/sec
[1240] 19.0-20.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 20.0-21.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 21.0-22.0 sec 128 KBytes 1.05 Mbits/sec
[1240] 22.0-23.0 sec 128 KBytes 1.05 Mbits/sec
```

## 8.32 Wake-Fi Transmit

### 8.32.1 Example Overview

#### Overview

This application triggers the wake up pattern to awake the Redpine modules that are configured wake up on RF interrupt

#### Sequence of Events

This example explains user how to:

- Customize transmission pattern and trigger the packet on air

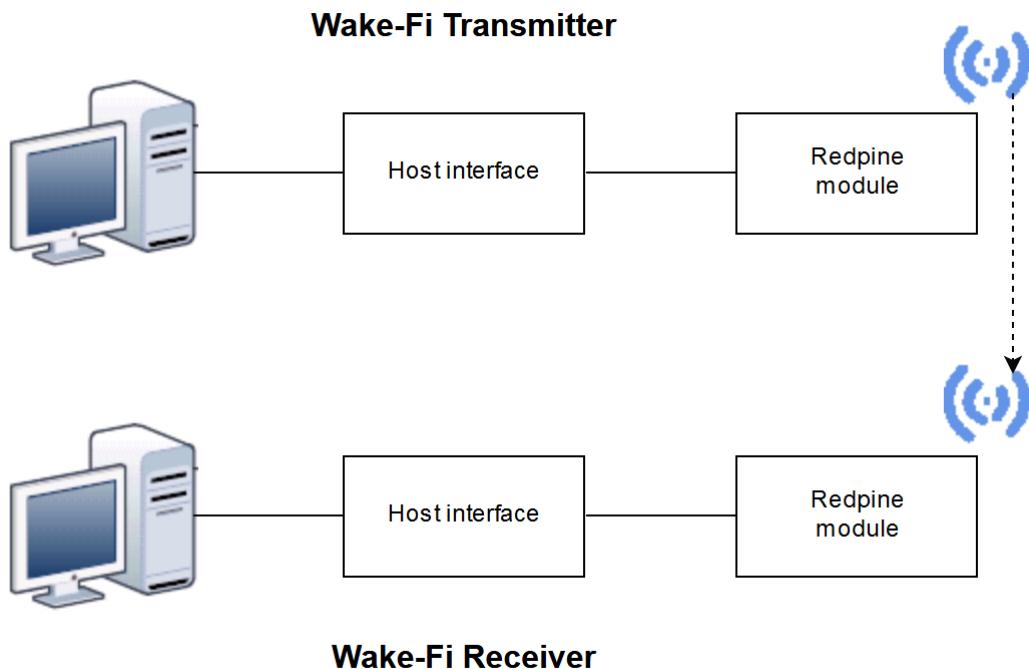
### 8.32.2 Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect

- Redpine module



**Figure 1: Setup diagram**

### 8.32.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_wakefi.h** file and update modify the following macros,  
**RSI\_WUTX\_SEL** refers to the type of radio to be used while transmission of pattern on air.  
Valid configurations are:  
**RSI\_ZB\_TX** - To select Zigbee radio  
**RSI\_WLAN\_TX** - To select WLAN radio  
**RSI\_BT\_TX** - To select BT radio  
**RSI\_BLE\_TX** - To select BLE radio

```
#define RSI_WUTX_SEL  
RSI_WLAN_TX
```

**RSI\_CHANNEL\_NO** refers to the channel to be used for the transmission of pattern

```
#define RSI_CHANNEL_NO
```

11

**Note:**

Valid values for CHANNEL\_NO in 2.4GHz band are 1 to 11 and 5GHZ band are 36 to 48 and 149 to 165. In this example default configured band is 2.4GHz.

So, if user wants to use 5GHz band then user has to set RSI\_BAND macro to 5GHz band in (Example folder \$) rsi\_wlan\_config.h file.

**WUTX\_MODE-** To select transmission mode

Valid configurations are:

**UNICAST**- To enable unicast transmission to a specific node

**BROADCAST**- To enable broadcast of wake-fi pattern

```
#define RSI_WUTX_MODE  
RSI_UNICAST
```

**RSI\_WUTX\_L1\_ENABLE-** To enable/disable L1 pattern

```
#define RSI_WUTX_L1_ENABLE  
RSI_ENABLE
```

**RSI\_WUTX\_TRAIL\_ENABLE -**

```
#define RSI_WUTX_TRAIL_ENABLE  
RSI_DISABLE
```

**RSI\_WUTX\_L1\_LENGTH** -Following are the valid configurations:

**RSI\_L1\_LEN\_2** - 2 bit pattern

**RSI\_L1\_LEN\_4** - 4 bit pattern

**RSI\_L1\_LEN\_8**- 8 bit pattern

**RSI\_L1\_LEN\_16**- 16 bit pattern

```
#define RSI_WUTX_L1_LENGTH  
RSI_L1_LEN_2
```

**RSI\_WUTX\_L2\_LENGTH** -Following are the valid configurations:

**RSI\_L2\_LEN\_1**- 1 bit pattern

**RSI\_L2\_LEN\_2**- 2 bit pattern

**RSI\_L2\_LEN\_4**- 4 bit pattern

**RSI\_L2\_LEN\_8**- 8 bit pattern

**RSI\_L2\_LEN\_16**- 16 bit pattern

**RSI\_L2\_LEN\_32**- 32 bit pattern

**RSI\_L2\_LEN\_64**- 64 bit pattern

```
#define RSI_WUTX_L2_LENGTH  
RSI_L2_LEN_64
```

**RSI\_WUTX\_TRIAL\_LENGTH**- Following are the valid configurations:

**RSI\_TRIAL\_LEN\_64** - 64 bit pattern  
**RSI\_TRIAL\_LEN\_128** - 128 bit pattern  
**RSI\_TRIAL\_LEN\_192** - 192 bit pattern  
**RSI\_TRIAL\_LEN\_256** - 256 bit pattern

2. Open **rsi\_wakefi\_tx.c** file and update/modify following macros,

```
#define RSI_WUTX_TRIAL_LENGTH  
RSI_TRIAL_LEN_64
```

Edit pattern byte stream for L1,L2,TRAIL DATA based on requirement

NOTE: If your pattern is less than MAX\_LENGTH then append with 0(zero)s

ex: **RSI\_WUTX\_L1\_LENGTH** is 2bit then **uint8\_t L1[L1\_PATTERN\_MAX\_LENGTH]={0x02,0x00};**

```
//! Array to hold L1 pattern values  
uint8_t L1[L1_PATTERN_MAX_LENGTH]={0};  
  
//! Array to hold L2 pattern values  
uint8_t  
L2[L2_PATTERN_MAX_LENGTH]={0xE3,0x71,0x38,0x30,0xE6,0xE6,0xC1,0xE  
3};  
  
//! Array to hold trail data values  
uint8_t trail_data[TRAIL_DATA_LENGTH]={0};
```

3. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
//! To set custom feature select bit map  
#define RSI_CUSTOM_FEATURE_BIT_MAP BIT(31) //!Enable Extended  
Custom Feature bitmap  
  
//! To set Extended custom feature select bit map  
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP 0x100 //! Enable WakeFi  
Wutx Feature enable bit(8)
```

## Executing the Application

1. Connect Redpine device to the PC open IDE.
2. Configure the macros in the files located at,  
**rsi\_wakefi\_tx.c**  
**rsi\_wakefi.h**  
**rsi\_wlan\_config.h**
3. Build and launch the application.
4. Refer to Wake-Fi RX application document to configure Wake-Fi Rx module to continue this test

- 
5. After the program gets executed, Redpine module starts transmission of the pattern on air. A RED LED (TRI LED) blink indication is provided on board on every successful transmission.

## 8.33 Web Socket Example

### 8.33.1 WebSocket Overview

WebSocket is designed to be implemented in [web browsers](#) and [web servers](#), but it can be used by any client or server application. The WebSocket Protocol is an independent TCP-based protocol. Its only relationship to [HTTP](#) is that its [handshake](#) is interpreted by HTTP servers as an [Upgrade request](#). WebSocket enables streams of messages on top of TCP.

### 8.33.2 Example Overview

#### Overview

This application demonstrates how to configure device in client mode to open Web socket to transmit data over Websocket to Web Server.

#### Sequence of Events

This Application explains user how to:

- Connect the Device to an Access point and get IP address through DHCP
- Connect to Web server opened on remote peer using web socket client
- Send data to web socket server

### 8.33.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

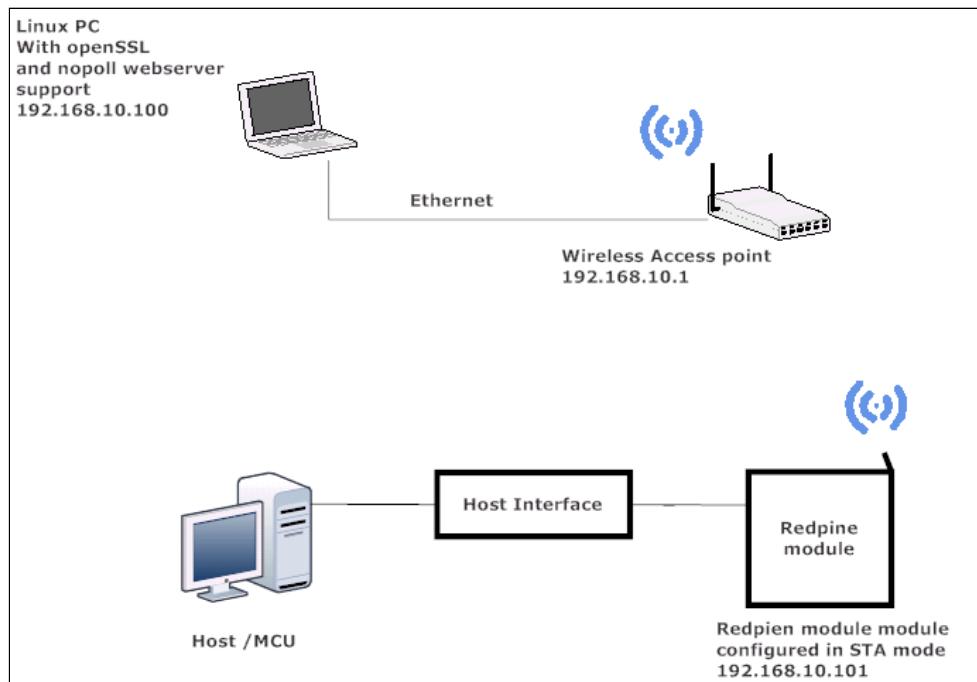
- Windows PC with Keil or IAR IDE in case of WiSeMCU in case of WiSeMCU
- Windows PC with Dev-C++ IDE / SPI/ USB) in case of WiSeConnect
- Redpine module
- Wi-Fi Access point
- Linux/Window PC with Web socket server and open SSL support ( This application using no-poll server for web server)

**Note:** Download **No-Poll server** from the below link:

<http://www.aspl.es/nopoll/downloads/nopoll-0.3.2.b232.tar.gz>

**Note:**

Installation of open SSL is needed.



**Figure 1: Setup Diagram**

#### 8.33.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_websocket\_client\_app.c** file and update/modify following macros:  
**SSID** refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to the channel in which device should scan. If it is 0,device will scan all channels.

```
#define CHANNEL_NO      0
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode  
**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE    RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK           "<psk>"
```

#### To Load certificate

```
#define LOAD_CERTIFICATE    1
```

If **LOAD\_CERTIFICATE** set to 1, application will load certificate which is included using `rsi_wlan_set_certificate` API.

By default, application loading "**cacert.pem**" certificate if **LOAD\_CERTIFICATE** is enable. In order to load different certificate, user has to follow the following steps :

- `rsi_wlan_set_certificate` API expects the certificate in the form offline array. So, convert the pem certificate into linear array form using python script provided in the release package "**sapis/examples/utilities/certificates/certificate\_script.py**"

**Example:** If the certificate is WiFi-user.pem .Give the command in the following way `python certificate_script.py ca-cert.pem`

Script will generate WiFi user.pem in which one linear array name dca cert contains the certificate.

- After conversion of certificate, update **rsi\_ssl\_client.c** source file by including the certificate file and by providing the required parameters to `rsi_wlan_set_certificate` API.

#### Note:

Once certificate loads into the device, it will write into the device flash. So, user need not load certificate for every boot up unless certificate change.

So define **LOAD\_CERTIFICATE** as 0, if certificate is already present in the device.

#### Note:

All the certificates are given in the release package.

**Path:** *sapis/examples/utilities/certificates*

#### To Open Websocket client:

**FLAGS** refer to open normal web socket or web socket over SSL with IPv4 or IPv6

If User wants to open normal web socket client set **FLAGS** to 0 or user wants to open Web socket over SSL then set **FLAGS** to 2 (WEB\_SOCKET\_SSL). Default configuration is Normal Web socket client

```
#define FLAGS          0
```

Port number of there mote web socket server. Default configuration of server port number is Normal Web socket server.

```
#define SERVER_PORT    1234
```

#### Note:

If user wants to open Web socket over SSL then update **SERVER\_PORT** macro with 1235 or 1236 as in no poll SSL web socket server is running on port numbers 1235 and 1236

IP address of the remote web socket server

IP address should be configured in long format and in little endian byte order.

**Example:** To configure "192.168.10.101" as IP address, update the macro **SERVER\_IP** as **0x650AA8C0**.

```
#define SERVER_IP          0x650AA8C0
```

Web socket resource name, maximum 50 characters

```
#define WEB_SOCKET_RESOURCE_NAME      "<resource_name>"
```

Websockethostname, maximum 50 characters

```
#define WEB_SOCKET_HOST_NAME        "<host_name>"
```

Message to send remote server

```
#define MESSAGE                  "<message>"
```

**FIN\_BIT** is used to indicate whether it is the last packet or not.

In this example, **FIN\_BIT** is setting in last packet of configured number of packets (**NUMBER\_OF\_PACKETS**). After receiving packet with FIN BIT set, web socket server sends back the received data to WiSeConnect device.

```
#define FIN_BIT                128
```

Number of packets to send

```
#define NUMBER_OF_PACKETS       1000
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN        8000
```

#### To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE1
```

**Note:** If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in longformatand in little endianbyteorder.  
**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP          0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY          0X010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK          0X00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE           RSI_DISABLE
#define RSI_FEATURE_BIT_MAP        FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS          RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP (TCP_IP_FEAT_DHCPV4_CLIENT | TCP_IP_FEAT_SSL)
#define RSI_CUSTOM_FEATURE_BIT_MAP 0
#define RSI_BAND                   RSI_BAND_2P4GHZ
```

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect Redpine device in STA mode.
2. Install no-poll server in Linux/Windows PC and run Websocket server. Please follow the below steps to run no-poll server in LINUX PC,
  - Download No-Poll server from below link,  
<http://www.aspl.es/nopoll/downloads/nopoll-0.3.2.b232.tar.gz>
  - Unzip the package and do ". /configure", "make" and "make install".

- After successful installation, go to test folder to run sample websocket server.
- If user wants to test websocket over SSL, Copy server-cert.pem and server-key certificates from release package (sapis/examples/utilities/certificates) to test folder.
- Open nopoll-regression-listener.c file and change certificates to server-cert.pem and server-key.pem in nopoll\_listener\_set\_certificate(listener2, "server-cert.pem", "server-key.pem", NULL) and nopoll\_ctx\_set\_certificate(ctx, NULL, "server-cert.pem", "server-key.pem", NULL) functions.

```

/* call to create a listener */
listener = nopoll_listener_new (ctx, "0.0.0.0", "1234");
if (!nopoll_conn_is_ok (listener)) {
    printf ("ERROR: Expected to find proper listener connection status, but found..\n");
    return -1;
}
printf ("noPoll listener started at: %s:%s (refs: %d)..\\n", nopoll_conn_host (listener), nopoll_conn_port (listener), nopoll_conn_ref_count
);

/* now start a TLS version */
printf ("Test: starting listener with TLS (TLSv1) at :1235\\n");
listener2 = nopoll_listener_tls_new (ctx, "0.0.0.0", "1235");
if (!nopoll_conn_is_ok (listener2)) {
    printf ("ERROR: Expected to find proper listener TLS connection status, but found..\n");
    return -1;
} /* end if */

/* configure certificates to be used by this listener */
if (!nopoll_listener_set_certificate (listener2, "test-certificate.crt", "test-private.key", NULL)) {
    printf ("ERROR: unable to configure certificate for TLS socket..\n");
    return -1;
}

/* register certificates at context level */
if (!nopoll_ctx_set_certificate (ctx, NULL, "test-certificate.crt", "test-private.key", NULL)) {
    printf ("ERROR: unable to setup certificates at context level..\n");
    return -1;
}

```

- Compile the source code by giving "make" command in test folder.
- Run server by giving "./nopoll-regression-listener" command.
- After running the nopoll server, it will open Four sockets on port numbers 1234, 1235, 1236 and 1237.  
 1234 – For Normal Websocket server  
 1235 – For TLSv1 websocket server  
 1236 – For SSLv23 websocket server  
 12367– For SSLv3 websocket server

test@cpu380:/home/test/praveen/r

File Edit View Search Terminal Tabs Help

test@cpu380:/home/test/divya\_jan7/RS... x test@cpu380:/home/test/RS9113.NBZ... x

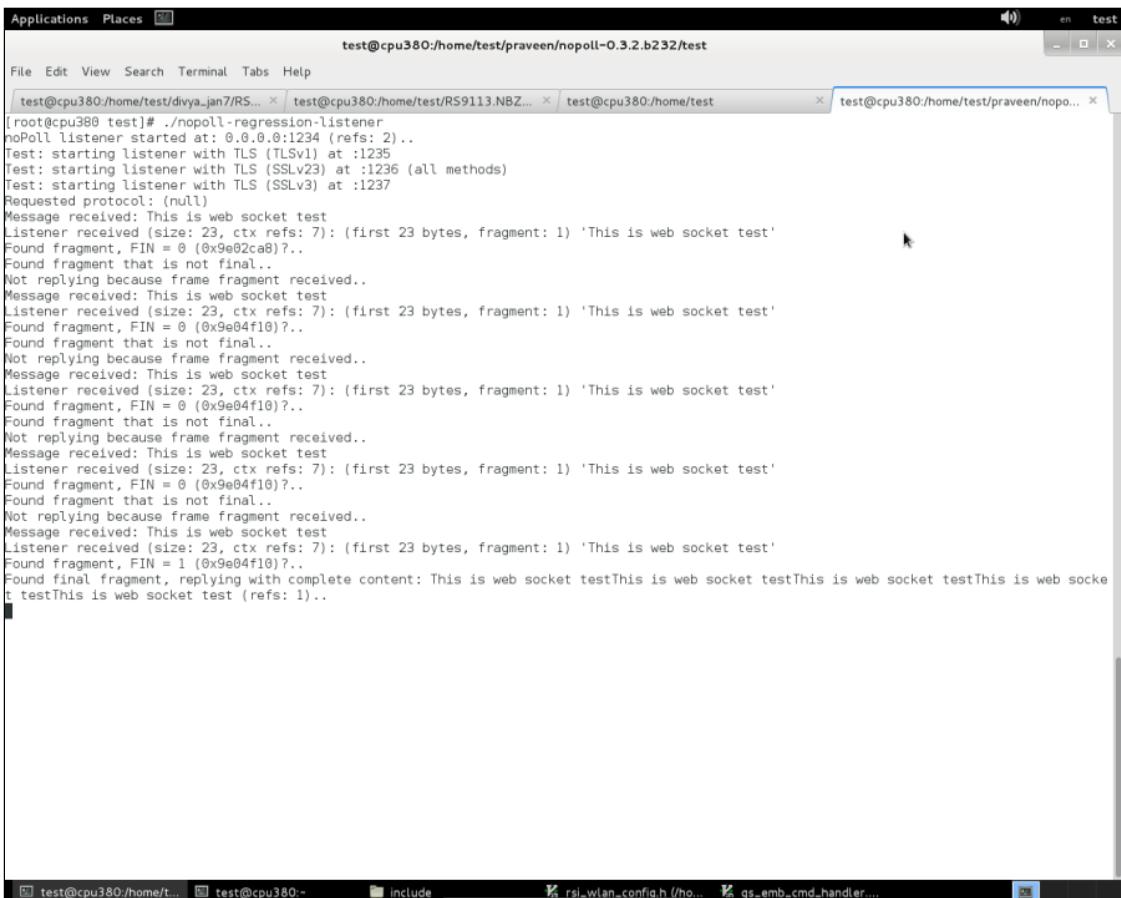
```
[root@cpu380 test]# ./nopoll-regression-listener
noPoll listener started at 0.0.0.0:1234 (refs: 2) .. Normal Websocket server
Test: starting listener with TLS (TLSv1) at :1235
Test: starting listener with TLS (SSLv23) at :1236 (all methods)
Test: starting listener with TLS (SSLv3) at :1237
```

websocket server with  
SSL

3. After the program gets executed, the device would be connected to access point having the configuration same as that of in the application and get IP.

4. The Device which is configured as Websocket client will connect to remote SSL server and sends number of packets configured in **NUMBER\_OF\_PACKETS**.

Please refer the below image for Data Rx on Websocket server.



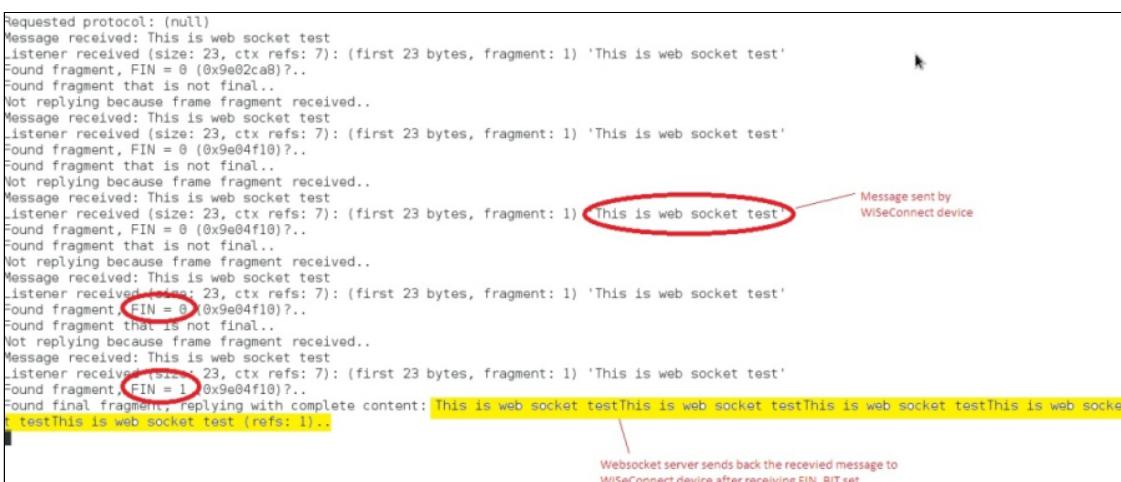
```

test@cpu380:/home/test/praveen/nopoll-0.3.2.b232/test
[root@cpu380 test]# ./nopoll-regression-listener
noPoll listener started at: 0.0.0.0:1234 (refs: 2)..
Test: starting listener with TLS (TLSv1) at :1235
Test: starting listener with TLS (SSLv23) at :1236 (all methods)
Test: starting listener with TLS (SSLv3) at :1237
Requested protocol: (null)
Message received: This is web socket test
Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 0 (0x9e02ca8)?..
Found fragment that is not final..
Not replying because frame fragment received..
Message received: This is web socket test
Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 0 (0x9e04f10)?..
Found fragment that is not final..
Not replying because frame fragment received..
Message received: This is web socket test
Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 0 (0x9e04f10)?..
Found fragment that is not final..
Not replying because frame fragment received..
Message received: This is web socket test
Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 0 (0x9e04f10)?..
Found fragment that is not final..
Not replying because frame fragment received..
Message received: This is web socket test
Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 1 (0x9e04f10)?..
Found final fragment, replying with complete content: This is web socket testThis is web socket testThis is web socket testThis is web socket test (refs: 1)..

```

5. From application, set FIN\_BIT in last data packet. After receiving data packet with FIN\_BIT set, WebSocket server sends back the data received until FIN\_BIT set to Redpine device.

Please find the below image for WebSocket server sending back data to the Redpine device after receiving packet with FIN\_BIT set,



```

Requested protocol: (null)
Message received: This is web socket test
Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 0 (0x9e02ca8)?..
Found fragment that is not final..
Not replying because frame fragment received..
Message received: This is web socket test
Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 0 (0x9e04f10)?..
Found fragment that is not final..
Not replying because frame fragment received..
Message received: This is web socket test
Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 0 (0x9e04f10)?..
Found fragment that is not final..
Not replying because frame fragment received..
Message received: This is web socket test
Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 0 (0x9e04f10)?..
Found fragment that is not final..
Not replying because frame fragment received..
Message received: This is web socket test
Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 1 (0x9e04f10)?..
Found final fragment, replying with complete content: This is web socket testThis is web socket testThis is web socket testThis is web socket test (refs: 1)..

```

6. The device will receive the message sent by WebSocket server and initiate connection close to websocket server. Refer the below image for connection close at websocket server.

```

Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 0 (0x9e04f10)?..
Found fragment that is not final..
Not replying because frame fragment received..
Message received: This is web socket test
Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 0 (0x9e04f10)?..
Found fragment that is not final..
Not replying because frame fragment received..
Message received: This is web socket test
Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 0 (0x9e04f10)?..
Found fragment that is not final..
Not replying because frame fragment received..
Message received: This is web socket test
Listener received (size: 23, ctx refs: 7): (first 23 bytes, fragment: 1) 'This is web socket test'
Found fragment, FIN = 1 (0x9e04f10)?..
Found final fragment, replying with complete content: This is web socket testThis is web socket test
t testThis is web socket test (refs: 1)..
Reg test: called connection close (TLS: 1)..

```

## 8.34 WEP Security Example

### 8.34.1 WEP protocol Overview

Wired Equivalent Privacy (WEP) is a security protocol for wireless networks that encrypts transmitted data . The disadvantage is that without any security, your data can be intercepted without difficulty. WEP has three settings: Off (no security), 64-bit (weak security), 128-bit (a bit better security). WEP is not difficult to crack, and using it reduces performance slightly. However, WEP was an early attempt to secure wireless networks, and better security is now available such as DES, VPN, and WPA.

### 8.34.2 Example Overview

#### Overview

The WEP security application demonstrates how to connect to a WEP secured Access point and open a standard TCP client socket and sends data on server socket with the Redpine device.

#### Sequence of events

- Configure an Access point in WEP secured mode
- Connect the Redpine device to the WEP secured AP and get IP through DHCP
- Server socket at Access point has to be opened by means of application program like iperf
- TCP client socket would be connected to the server socket that is opened
- Once the connection is established, Data is sent to the TCP server

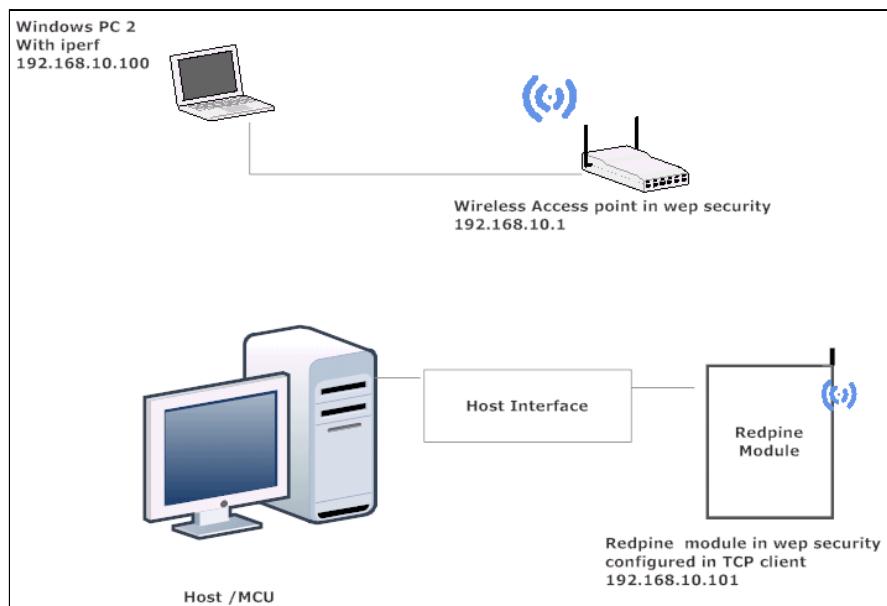
### 8.34.3 Example setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with Keil or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect

- WiFi Access point
- Access Point with WEP security
- Application program like iperf
- Windows PC2
- TCP server application running in Windows PC2 (This application uses iperf application to open TCP server socket)



**Figure1: setup diagram**

#### 8.34.4 Configuration and Execution of the Application

Configuring the Application:

1. Open **rsi\_tcp\_client.c** files and update/modify following macros:

**SSID** refers to the name of the Access point to be created.

```
#define SSID           "<ap name>"
```

**CHANNEL\_NO** refers to particular channel used to scan by the device. If channel is 0 then it will scan all channels.

```
#define CHANNEL_NO      <channel_num>
```

**SECURITY\_TYPE** refers to type of security WEP (RSI\_WEP) **WEP\_INDEX** refers to the one of the four keys to be used for connection **WEP\_KEY0**, **WEP\_KEY1**, **WEP\_KEY2**, **WEP\_KEY3** are the keys , they can be 10 bytes or 26 bytes.

```
#define SECURITY_TYPE RSI_WEP
#define WEP_INDEX "<index>"
#define WEPKEY0 "<wep key 0>"
#define WEPKEY1 "<wep key 1>"
#define WEPKEY2 "<wep key 2>"
#define WEPKEY3 "<wep key 3>"
```

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT 5001
```

**SERVER\_PORT** port refers remote TCP server port number which is opened in windows PC2.

```
#define SERVER_PORT 5001
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with TCP server socket.  
IP address should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.100" as IP address, update the macro **DEVICE\_IP** as **0x640AA8C0**.

```
#define SERVER_IP_ADDRESS 0x640AA8C0
```

**NUMBER\_OF\_PACKETS** refers how many packets to send from device to TCP server

```
#define NUMBER_OF_PACKETS 1000
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 8000
```

**To configure IP address**

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE 1
```

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP          0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY          0x010AA8C0
```

IP address of the network mask should also be in longformat and in little endianbyte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK          0x00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE           RSI_DISABLE
#define RSI_FEATURE_BIT_MAP        FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS          RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_CUSTOM_FEATURE_BIT_MAP 0
#define RSI_BAND                   RSI_BAND_2P4GHZ
```

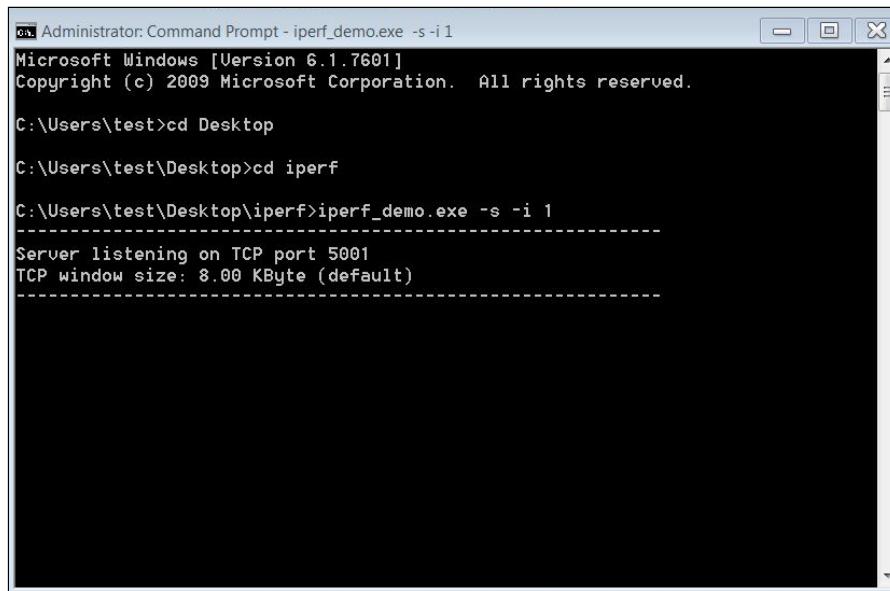
**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example

## Executing the Application

1. Configure the Access point in security WEP (RSI\_WEP)**WEP\_INDEX** refers to the one of the four keys to be used for connection **WEP\_KEY0**, **WEP\_KEY1**, **WEP\_KEY2**, **WEP\_KEY3** are thekeys , they can be 10 bytes or 26 bytes.

2. Open an iperf, Users can download application from the link below,  
<https://iperf.fr/iperf-download.php#windows> TCP server listening on port SERVER\_PORT on remote machine in the following format.  
**iperf.exe -s -p <SERVER\_PORT> -i 1**



```
Administrator: Command Prompt - iperf_demo.exe -s -i 1
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\test>cd Desktop
C:\Users\test\Desktop>cd iperf
C:\Users\test\Desktop\iperf>iperf_demo.exe -s -i 1
-----
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
-----
```

3. After program gets executed, Redpine Device would scan and connect to Access point and get IP. The Device which is configured as TCP client will connect to iperf server and sends number of packets configured in NUMBER\_OF\_PACKETS.

## 8.35 Wi-Fi Direct Autonomous GO Example

### 8.35.1 Example Overview

Wi-Fi Direct is a [Wi-Fi](#) standard enabling devices to easily connect with each other without requiring a [wireless access point](#). It is usable for everything from internet browsing to file transfer and to communicate with more than one device simultaneously at typical Wi-Fi speeds. One advantage of Wi-Fi Direct is the ability to connect devices even if they are from different manufacturers.

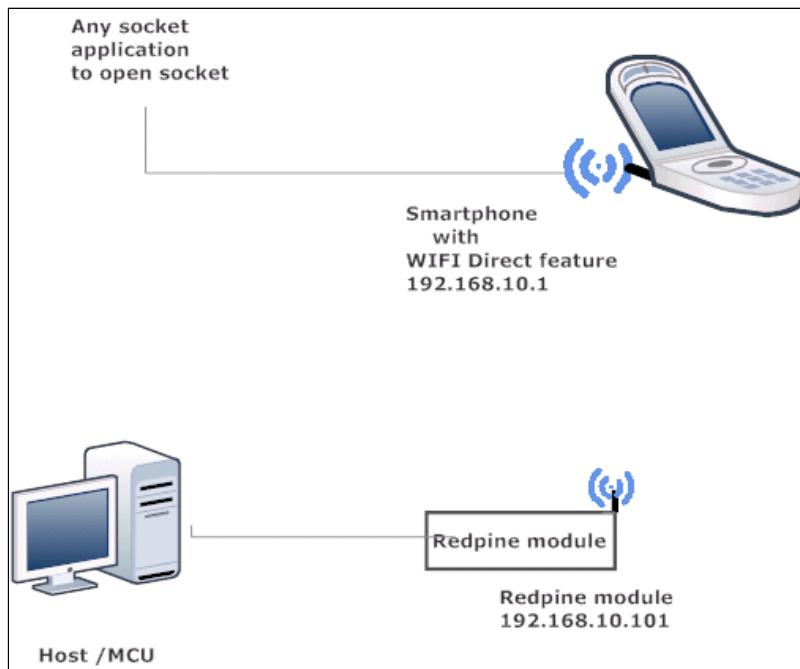
Wi-Fi Direct may not only replace the need for [routers](#), but may also replace the need of Bluetooth for applications that do not rely on [low energy](#).

The application demonstrates how to configure device in WIFI Direct in autonomous GO mode and how to open socket to transmit or receive data. The procedure is outlined below:

- Configure device in WIFI Direct auto GO mode
- Connect to peer device
- Open TCP socket

### 8.35.2 Setup required

1. Windows PC with Keil or IAR IDE in case of WiSeMCU
2. Redpine Module
3. Remote device or peer device(Smart Phone) configured in WIFI Direct mode and this device is used to join the created group.



### 8.35.3 Description

Redpine device act as WiFi Direct Autonomous GO mode. Device will first create Autonomous GO without any device discovery and negotiation. Once WiFi Direct Auto GO is created it will wait for remote device to join the group. When remote device sends connection request to the device, it will simply accept the connection. Auto GO **CONNECTION\_TYPE** is set to **ACCEPT\_CONNECTION\_REQ** device wait for remote device join request after that device will send join request.

After joining remote device may act as client, as it is simply joining the auto GO.

Need to enable the DHCP server feature for the WiFi Direct Auto GO. Device open TCP socket to data transmit and receive. At the remote side device also need to open TCP socket.

### 8.35.4 Configuration and Execution of the Application

#### Configuring the application

1. Edit the **rsi\_wfd\_client.c** file in the following path to establish connection with the remote WiFi Direct device.  
**sapis/examples/wlan/wifi\_direct/**

From given configuration,

**CONNECTION\_TYPE** is set to send join request or wait for join request from remote device.

```
#define CONNECTION_TYPE  
ACCEPT_CONNECTION_REQ
```

**REMOTE\_DEVICE** is WiFi-Direct device which is going to join the autonomous group.

```
#define REMOTE_DEVICE  
"<device_name>"
```

**RSI\_DEVICE\_NAME** refers to device name for module. Maximum length of this field is 32 characters.

```
#define RSI_DEVICE_NAME  
"<ap_name>"
```

**POST\_FIX\_SSID** refers to post fix SSID for device

```
#define POST_FIX_SSID  
"<post_ssid>"
```

**GO\_INTENT** this determines whether the device is intended to form a GO(Group Owner) or work as a WIFI-Direct peer node. If GO Intent value 16 then module will form a autonomous group without any device discovery

```
#define GO_INTENT 16
```

**CHANNEL** refers to channel in which device would operate

```
#define CHANNEL 0
```

**PSK** refers to password, this PSK is used if the module becomes a GO owner

```
#define PSK "<psk>"
```

2. To open TCP server socket and receive TCP data configure the below macros.

Internal socket port number.

```
#define DEVICE_PORT 5001
```

Number of packets to send

```
#define NUMBER_OF_PACKETS 1000
```

Application memory length which is required by the driver

#define GLOBAL_BUFF_LEN	8000
-------------------------	------

Edit the Wlan configuration file: **sapis/include/rsi\_wlan\_config.h**

<b>CONCURRENT_MODE</b>	<b>DISABLE</b>
<b>RSI_FEATURE_BIT_MAP</b>	<b>FEAT_SECURITY_OPEN</b>
<b>RSI_TCP_IP_BYPASS</b>	<b>DISABLE</b>
<b>RSI_TCP_IP_FEATURE_BIT_MAP</b>	<b>(TCP_IP_FEAT_DHCPV4_SERVER)</b>
<b>RSI_CUSTOM_FEATURE_BIT_MAP0</b>	
<b>RSI_BAND</b>	<b>RSI_BAND_2P4GHZ</b>

**Note:**

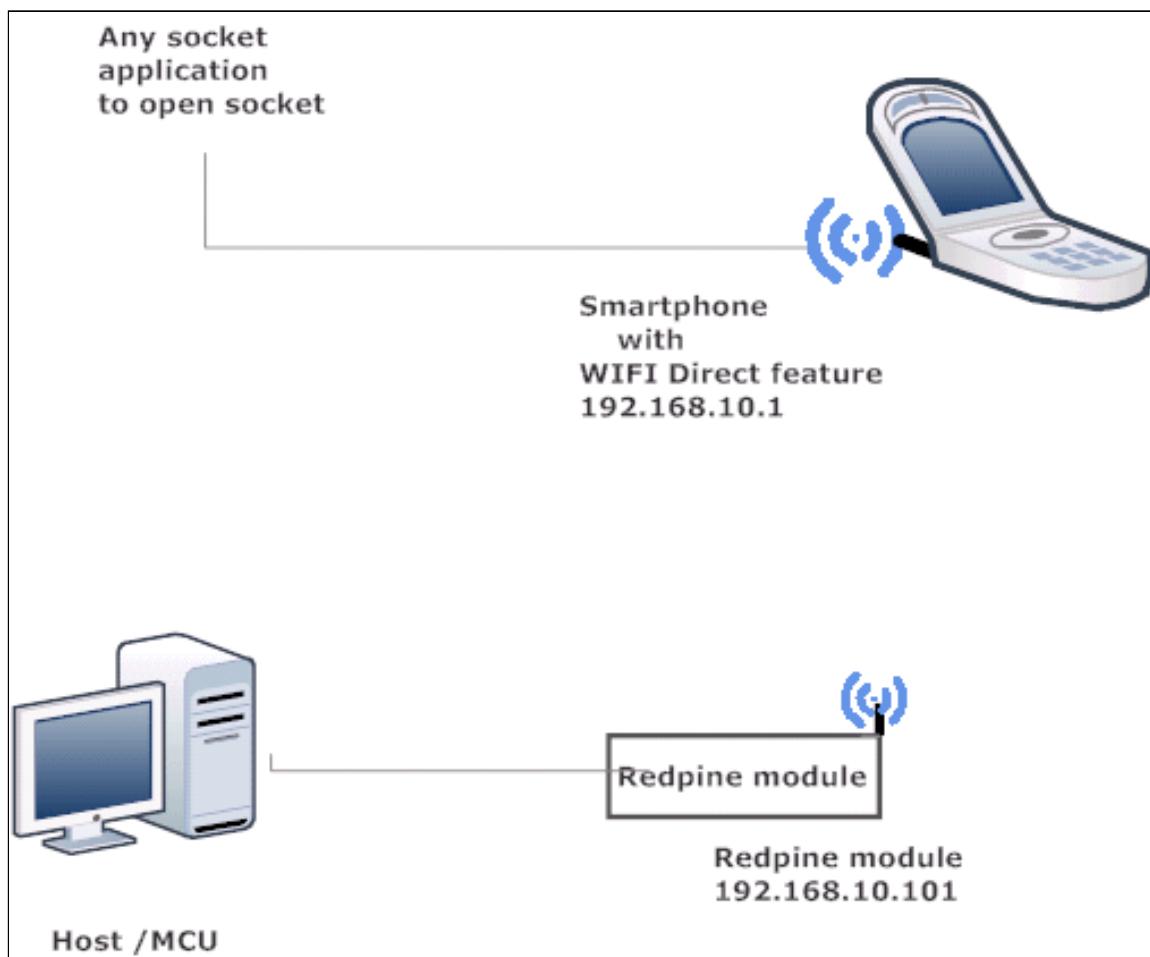
Supported security modes list are mentioned in **rsi\_wlan\_apis.h** which is at follow path: /sapis/include/rsi\_wlan\_apis.h

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Connect Redpine device to the PC open IDE.
2. Configure the macros in the files located at  
**rsi\_wfd\_auto\_go.c**  
**rsi\_wlan\_config.h**
3. Build and launch the application.
4. Configure remote device in WIFI-Direct to connect to module.
5. After the program gets executed, WiSeConnect module creates a WFD autonomous group and wait for remote device to join the group.



### Wi-Fi Direct Device Connected

1. Remote device will send invitation request to join the group.
2. Once module receives the connection request, it calls `rsi_wlan_wfd_connect` sAPI with its GO name as ssid to start WiFi Direct session i.e. credential sharing with remote peer and then remote peer will connect by using the shared credentials.
3. Device will wait for client to join the group. Once remote device join the group, TCP server socket opened in module and waits for TCP connection from remote peer.
4. Open the TCP client socket and send data from remote peer using any TCP application like iperf.
5. Redpine device receives TCP data configured number of packets sent by remote peer and exits.

## 8.36 Wi-Fi Direct Client Example

### 8.36.1 Example Overview

#### Overview

Wi-Fi Direct is a [Wi-Fi](#) standard enabling devices to easily connect with each other without requiring a [wireless access point](#). It is usable for everything from internet browsing to file transfer and to communicate with more than one device simultaneously at typical Wi-Fi speeds. One advantage of Wi-Fi Direct is the ability to connect devices

even if they are from different manufacturers. Wi-Fi Direct may not only replace the need for [routers](#), but may also replace the need of Bluetooth for applications that do not rely on [low energy](#).

### Sequence of Events

The application demonstrates how to configure device in WIFI Direct mode and how to open socket to transmit or receive data. Following are steps:

- Configure in WIFI Direct mode
- Connect to peer device
- Open TCP socket

#### 8.36.2 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Redpine module
- Remote device or peer device(Smart Phone) configured in WIFI Direct mode to which device will connect.

#### Description:

Redpine device act as Wi-Fi Direct client mode.

Device wait for remote device to join or send join request based on **CONNECTION\_TYPE** macro set.

If **CONNECTION\_TYPE** is set to **ACCEPT\_CONNECTION\_REQ** device wait for remote device join request after that device will send join request.

If **CONNECTION\_TYPE** is set to **SEND\_CONNECTION\_REQ** device first send join request to remote device to join. After joining device may act as GO(Group Owner) or client, based on GO Intent value, negotiation will happen between device and remote side device.

Get the IP address for device. Device open TCP socket to data transmit and receive. At the remote side device also need to open TCP socket.

#### 8.36.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open **rsi\_wifi\_direct.c** file and update / modify the following macros.

**CONNECTION\_TYPE** is set to send join request or wait for join request from remote device.

```
#define CONNECTION_TYPE  
SEND_CONNECTION_REQ
```

**REMOTE\_DEVICE** is **WIFI-Direct device to which wants to connect**.

```
#define REMOTE_DEVICE  
"<device_name>"
```

**RSI\_DEVICE\_NAME** refers to device name for module. Maximum length of this field is 32 characters.

```
#define RSI_DEVICE_NAME  
"<ap_name>"
```

**POST\_FIX\_SSID** refers to post fix SSID for device

```
#define POST_FIX_SSID  
"<post_ssid>"
```

**GO\_INTENT** this determines whether the device is intended to form a **GO(Group Owner)** or work as a **WIFI-Direct peer node**. **GO Intent value is between 0 to 16**

```
#define GO_INTENT
```

0

**CHANNEL** refers to channel in which device would operate

```
#define CHANNEL
```

0

**PSK** referstopassword, this PSK is used if the module becomes a GO owner

```
#define PSK  
"<psk>"
```

## 2. Enable/Disable DHCP mode

1 – Enables DHCP mode (gets the IP from DHCP server)  
0 – Disables DHCP mode

```
#define DHCP_MODE
```

1

If DHCP mode is disabled, then change the following macros to configure static IP address

IP address to be configured to the device should be in long format and in little endian byte order.  
Example: To configure "192.168.10.101" as IP address, update the macro **DEVICE\_IP** as **0x650AA8C0**.

```
#define DEVICE_IP  
0x650AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order  
Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY  
0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order  
Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK  
0x00FFFFFF
```

To open TCP server socket and receive TCP dataconfigure the below macros.

Internal socket port number.

```
#define DEVICE_PORT  
5001
```

Number of packets to send

```
#define NUMBER_OF_PACKETS  
1000
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN  
8000
```

**Note:**

Supported security modes list are mentioned in rsi\_wlan\_apis.h which is at follow path: /sapis/include/rsi\_wlan\_apis.h

<b>CONCURRENT_MODE</b>	<b>DISABLE</b>
<b>RSI_FEATURE_BIT_MAP</b>	<b>FEAT_SECURITY_OPEN</b>
<b>RSI_TCP_IP_BYPASS</b>	<b>DISABLE</b>
<b>RSI_TCP_IP_FEATURE_BIT_MAP</b>	(TCP_IP_FEAT_DHCPV4_CLIENT TCP_IP_FEAT_DHCPV4_SERVER )

<b>RSI_CUSTOM_FEATURE_BIT_MAP 0</b>	
<b>RSI_BAND</b>	<b>RSI_BAND_2P4GHZ</b>

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Connect Redpine device to the PC open IDE.
2. Configure the macros in the files located at  
**rsi\_wfd\_client.c**  
**rsi\_wlan\_config.h**
3. Build and launch the application.
4. Configure remote device in WIFI-Direct to connect to module
5. After the program gets executed, Redpine module configured as Wi-Fi-Direct client and send join request or wait for join request.  
WiFi Direct Device Connection Request Device Connected
6. After sending connect request GO negotiation start.
7. After negotiation, IP assign to module and TCP server socket opened in module and waits for TCP connection from remote peer.
8. Open the TCP client socket and send data from remote peer using any TCP application like iperf.
9. Redpine device receives TCP data configured number of packets sent by remote peer and exits.

## 8.37 Wireless Firmware Upgrade Example

### 8.37.1 Example Overview

#### Overview

The Wireless Firmware upgrade application demonstrates how WiSeConnect device would be created as Access point and allow stations to connect to update the firmware through webpage.

#### Sequence of Events

This Application explains user how to:

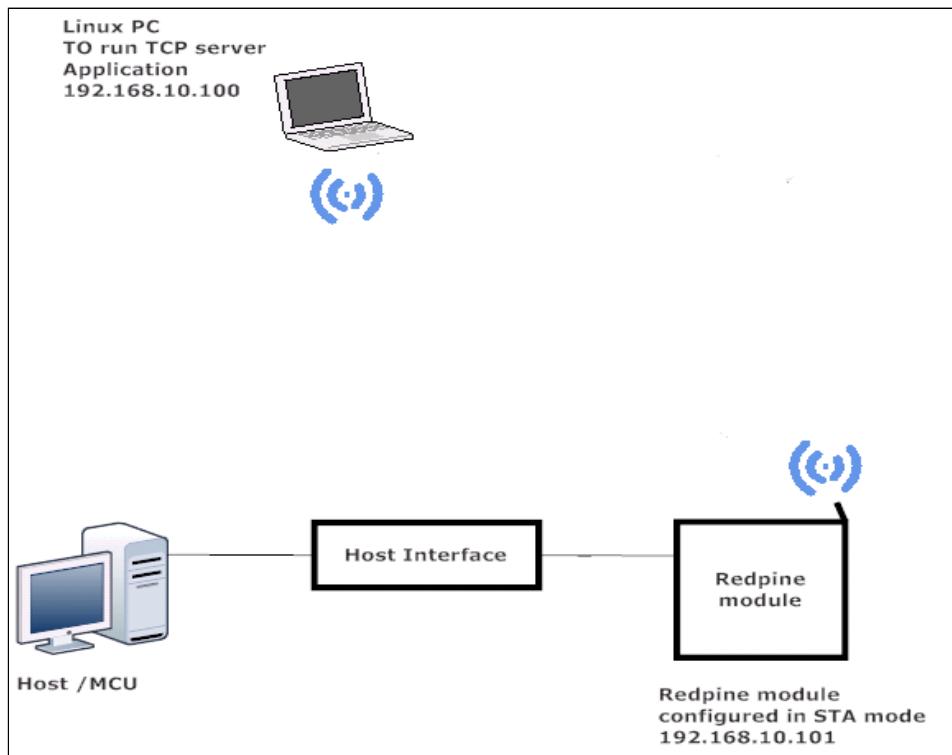
- Redpine Device starts as an Access point
- Allows stations to connect to update firmware through webpage.
- A Laptop having WiFi card can be used as Wireless station.
- Redpine device creates a network and acts as router between the connected stations
- Upgrade the received Firmware into the device.

### 8.37.2 Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with Keil or IAR IDE in case of WiSeMCU
- Redpine Module
- Wireless Access point
- Linux PC with TCP server application (TCP server application providing as part of release package)



**Figure 1: Setup Diagram**

### 8.37.3 Configuration and Execution of the Application

#### Configuring the Application

1. Open rsi\_wfup\_app.c file and update/modify following macros:From given configuration,

**SSID** refers to the name of the Access point .

```
#define SSID           "<ap  
name>"
```

**CHANNEL\_NO** refers to the channel in which AP would be started

```
#define CHANNEL_NO  
<channel_no>
```

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities

```
#define SECURITY_TYPE  
<security_type>
```

**ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports Open,TKIP, CCMP methods

```
#define ENCRYPTION_TYPE  
<encryption_type>
```

**PSK** refers to the secret key if the Access point to be configured in WPA/WPA2 security modes.

```
#define PSK  
"<psk>"
```

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons

```
#define BEACON_INTERVAL  
<beacon_interval>
```

To configure DTIM interval of the Access Point

```
#define DTIM_INTERVAL  
<dtim_interval>
```

FLAGS :

```
#define FLAGS  
WEB_PAGE_ASSOCIATED_TO_JSON
```

2. To configure IPaddress

IP address to be configured to the device should be in long format and in little endian byte order.  
**Example:** To configure “192.168.10.1” as IP address, update the macro DEVICE\_IP as 0x010AA8C0.

```
#define DEVICE_IP  
0X010AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure “192.168.10.1” as Gateway, update the macro GATEWAY as 0x010AA8C0

```
#define GATEWAY  
0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure “255.255.255.0” as network mask, update the macro NETMASK as 0x00FFFFFF

```
#define NETMASK  
0x00FFFFFF
```

3. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE  
RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP  
FEAT_SECURITY_OPEN  
#define RSI_TCP_IP_BYPASS  
RSI_DISABLE  
#define RSI_TCP_IP_FEATURE_BIT_MAP  
(TCP_IP_FEAT_DHCPV4_SERVER | TCP_IP_FEAT_HTTP_SERVER)  
#define RSI_CUSTOM_FEATURE_BIT_MAP  
FEAT_CUSTOM_FEAT_EXTENSION_VALID  
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP  
EXT_FEAT_256K_MODE  
#define RSI_BAND  
RSI_BAND_2P4GHZ
```

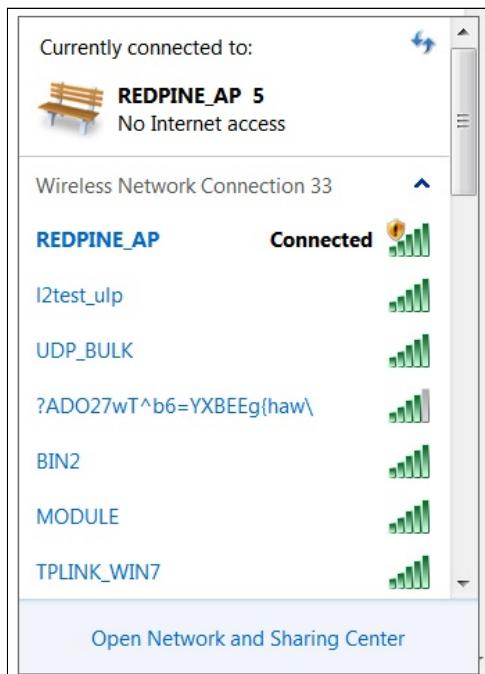
**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Connect WiSeConnect device to the Windows PC running Cocoox IDE.
2. Configure the macros in the files **rsi\_wfup\_app.c** & **rsi\_wlan\_config.h**
3. Build and launch the application.
4. After the program gets executed, Redpine Device would be created as Access point and starts Beaconsing.

5. TCP server socket is created and wait for accepting any TCP connection request on that port.
6. Now connect Laptop as WiFi station and open Webpage by typing the IP address of the module.



The screenshot shows the WiSeConnect configuration interface. The top navigation bar includes the Redpine Signals logo and the text "WiSeConnect<sup>®</sup>". The main menu has tabs for "CONFIGURATION" and "ADMINISTRATION".

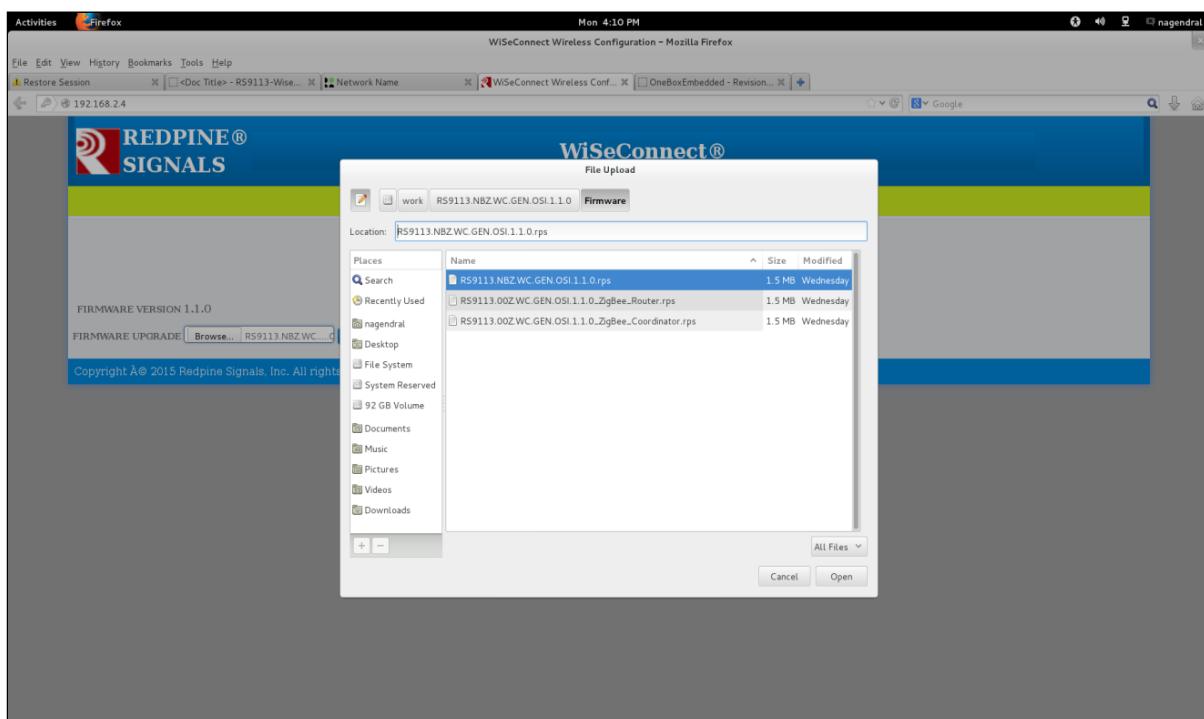
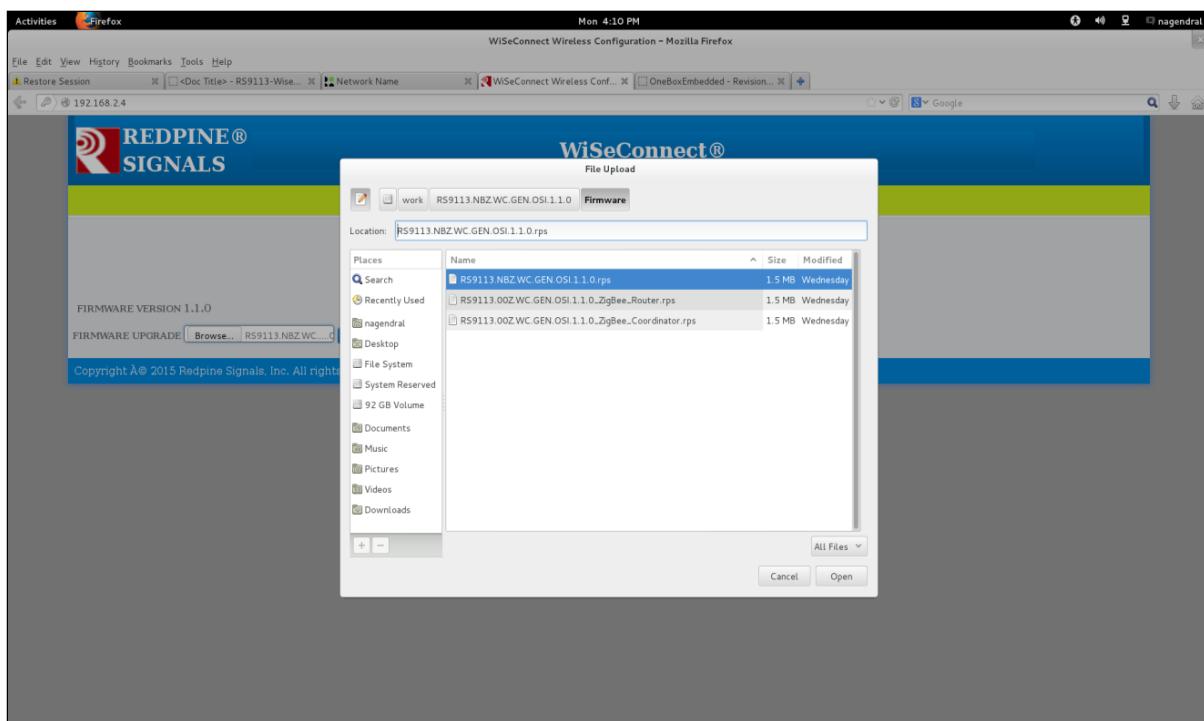
The left panel contains the "Basic Settings" section:

- Mode: Client (selected), AP
- BAND (GHZ): 2.4 (selected), 5.0, Dual
- SSID: mdnsd\_start
- TX POWER: High Power
- CHANNEL: 0
- SECURITY ENABLE:
- 802.11N:
- AGGREGATION:
- CUSTOM FEATURE: SELECT BITMAP: 0

The right panel contains the "IP Settings" and "Network Stack Settings" sections:

- IP VERSION: IPv4
- DHCPV4 CLIENT:
- Network Stack Settings:
  - DYNAMIC WEBPAGES (JSON):
  - HTTP CLIENT:
  - DNS CLIENT:
  - SNMP AGENT:
  - SSL:
  - PING:
  - HTTPS:
  - FTP CLIENT:

At the bottom, there are "Submit" and "Clear" buttons, and a copyright notice: "Copyright © 2015 Redpine Signals, Inc. All rights reserved."



**Figure 1:** Firmware Upgrade successfully

## 8.38 WPS Access Point Example

### 8.38.1 WPS Overview

WPS (WiFi Protected setup) is a network standard to create a secure wireless home network using PUSH button and PIN button method.

In Push button method, in which the user has to push a button, either an actual or virtual one, on both the access point and the new wireless client device. On most devices, this discovery mode turns itself off as soon as a connection is established or after a delay (typically 2 minutes or less).

In PIN method, in which the user has to enter secret PIN on both the access point and the new wireless client device. On most devices, this discovery mode turns itself off as soon as a connection is established or after a delay (typically 2 minutes or less).

### 8.38.2 Example Overview

#### Overview

Redpine device supports both Push button method and WPS pin method.

The WPS Access point application demonstrates how to configure the Redpine device as an access point and allow client device to connect to it using WPS PUSH method. The application also enables TCP data transmission from connected Wi-Fi Station to device Access Point.

#### Sequence of Events

This Application explains user how to:

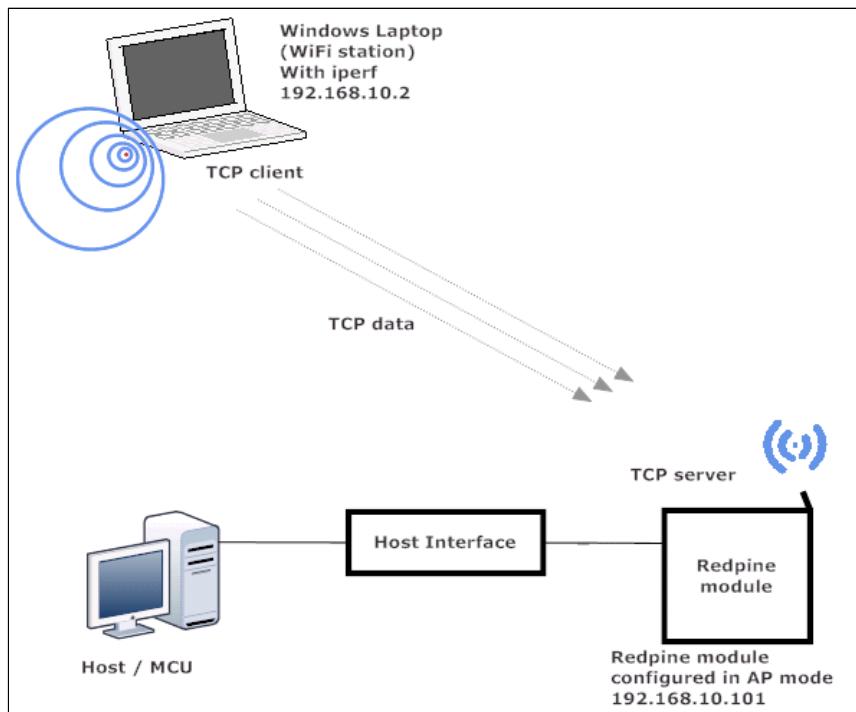
- Create WiSeConnect device as an Access point
- Enable WPS PUSH button method in Access Point
- Connect to client device to the access point using WPS PUSH method
- Open TCP client socket after successful connection
- Establish TCP connection with the TCP server opened in remote peer
- Send TCP data from remote peer to WiSeConnect device

### 8.38.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module as Access Point.
- WPS supported Mobile device as Wi-Fi station.
- A TCP server application running on the Windows PC2 (This example uses iperf for windows )



**Figure 1: Setup Diagram**

#### 8.38.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_wps\_ap.c** file and update / modify the following macros:  
**SSID** refers to the name of the Access point to be created. In WPS method it should NULL.

```
#define SSID  
"REDPINE_AP"
```

**SECURITY\_TYPE** refers to the type of security. In WPS method WiSeConnect module supports WPS push button method and WPS PIN method.

In this examples, WiSeConnect module connects to AP using WPS push button method. So, set **SECURITY\_TYPE** macro to **RSI\_WPS\_PUSH\_BUTTON**.

```
#define SECURITY_TYPE  
RSI_WPA2
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes. In WPS method PSK is not needed.

```
#define PSK NULL
```

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT <local  
port>
```

**SERVER\_PORT** port refers remote TCP server port number which is opened in Windows PC2.

```
#define SERVER_PORT <remote  
port>
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with TCP server socket.  
IP address should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.100" as IP address, update the macro **DEVICE\_IP** as **0x640AA8C0**.

```
#define SERVER_IP_ADDRESS  
0x640AA8C0
```

**NUMBER\_OF\_PACKETS** refers how many packets to receive from TCP client

```
#define NUMBER_OF_PACKETS <no of  
packets>
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 8000
```

To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE 1
```

**Note:** If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP  
0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order.

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY  
0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK  
0x00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update /modify the following macros :

```
#define CONCURRENT_MODE  
RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP  
FEAT_SECURITY_OPEN  
#define RSI_TCP_IP_BYPASS  
RSI_DISABLE  
#define RSI_TCP_IP_FEATURE_BIT_MAP  
TCP_IP_FEAT_DHCPV4_SERVER  
#define RSI_CUSTOM_FEATURE_BIT_MAP 0  
#define RSI_BAND  
RSI_BAND_2P4GHZ
```

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

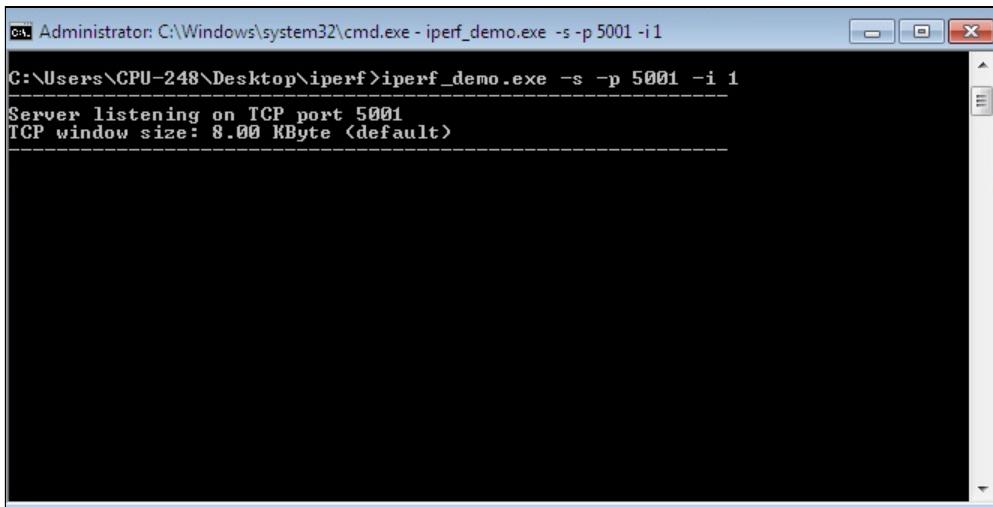
## Executing the Application

1. Configure the Access point in WPA2-PSK mode in order to connect Redpine device in STA mode.
2. Open TCP server application using iperf application in Windows PC1 which is connected to Access point through LAN.

3. Users can download application from the link below:

<https://iperf.fr/iperf-download.php#windows>

**iperf\_demo.exe -s -p <SERVER\_PORT> -i 1**



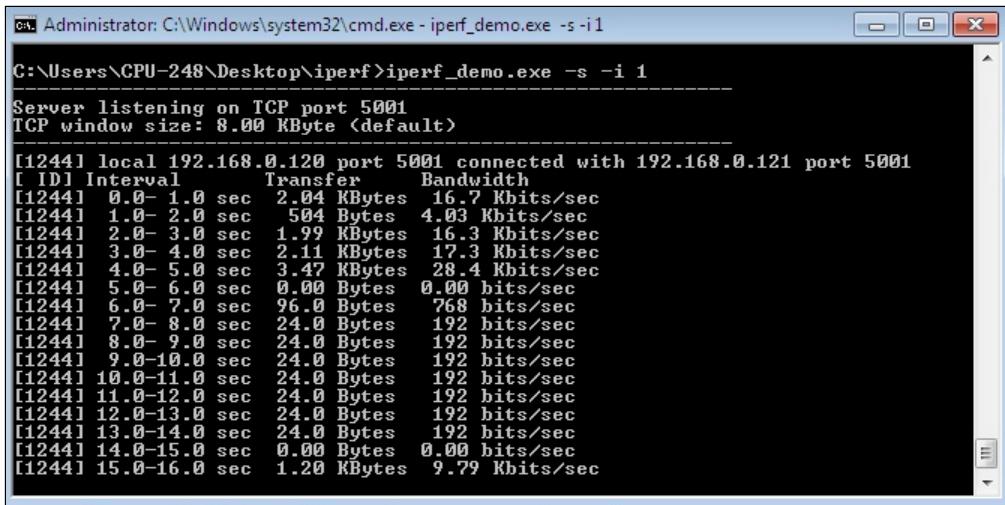
```
C:\ Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -p 5001 -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -p 5001 -i 1
Server listening on TCP port 5001
TCP window size: 8.00 KByte <default>
```

4. Press the WPS Push button on Access point which lasts for 2 minutes.

5. After program gets executed, the Redpine device would scan and connect to the access point using WPS push button method and get IP.

6. After successful connection, the device STA connects to TCP server socket opened on Windows PC1 using TCP client socket and sends configured **NUMBER\_OF\_PACKETS** to remote TCP server.

Please refer the below image for reception of TCP data on TCP server.



```
C:\ Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -i 1
Server listening on TCP port 5001
TCP window size: 8.00 KByte <default>
[12441] local 192.168.0.120 port 5001 connected with 192.168.0.121 port 5001
[ ID] Interval Transfer Bandwidth
[12441] 0.0- 1.0 sec 2.04 KBytes 16.7 Kbits/sec
[12441] 1.0- 2.0 sec 504 Bytes 4.03 Kbits/sec
[12441] 2.0- 3.0 sec 1.99 KBytes 16.3 Kbits/sec
[12441] 3.0- 4.0 sec 2.11 KBytes 17.3 Kbits/sec
[12441] 4.0- 5.0 sec 3.47 KBytes 28.4 Kbits/sec
[12441] 5.0- 6.0 sec 0.00 Bytes 0.00 bits/sec
[12441] 6.0- 7.0 sec 96.0 Bytes 768 bits/sec
[12441] 7.0- 8.0 sec 24.0 Bytes 192 bits/sec
[12441] 8.0- 9.0 sec 24.0 Bytes 192 bits/sec
[12441] 9.0-10.0 sec 24.0 Bytes 192 bits/sec
[12441] 10.0-11.0 sec 24.0 Bytes 192 bits/sec
[12441] 11.0-12.0 sec 24.0 Bytes 192 bits/sec
[12441] 12.0-13.0 sec 24.0 Bytes 192 bits/sec
[12441] 13.0-14.0 sec 24.0 Bytes 192 bits/sec
[12441] 14.0-15.0 sec 0.00 Bytes 0.00 bits/sec
[12441] 15.0-16.0 sec 1.20 KBytes 9.79 Kbits/sec
```

## 8.39 WPS PIN Example

### 8.39.1 WPS Overview

WPS (Wi-Fi Protected setup) is a network standard to create a secure wireless home network using PUSH button and PIN button method.

In Push button method, in which the user has to push a button, either an actual or virtual one, on both the access point and the new wireless client device. On most devices, this discovery mode turns itself off as soon as a connection is established or after a delay (typically 2 minutes or less).

---

In PIN method, in which the user has to enter secret PIN on both the access point and the new wireless client device. On most devices, this discovery mode turns itself off as soon as a connection is established or after a delay (typically 2 minutes or less).

### 8.39.2 Example Overview

#### Overview

Redpine device supports both Push button method and wps pin method to connect to an Access point. The WPS station application demonstrates how Redpine device would be connected to secured (WPA2) Access point using WPS PIN method.

In this application, after successful connection with the access point using WPS PIN method, Redpine device opens TCP socket and sends TCP data to configured remote peer.

#### Sequence of Events

This Application explains user how to:

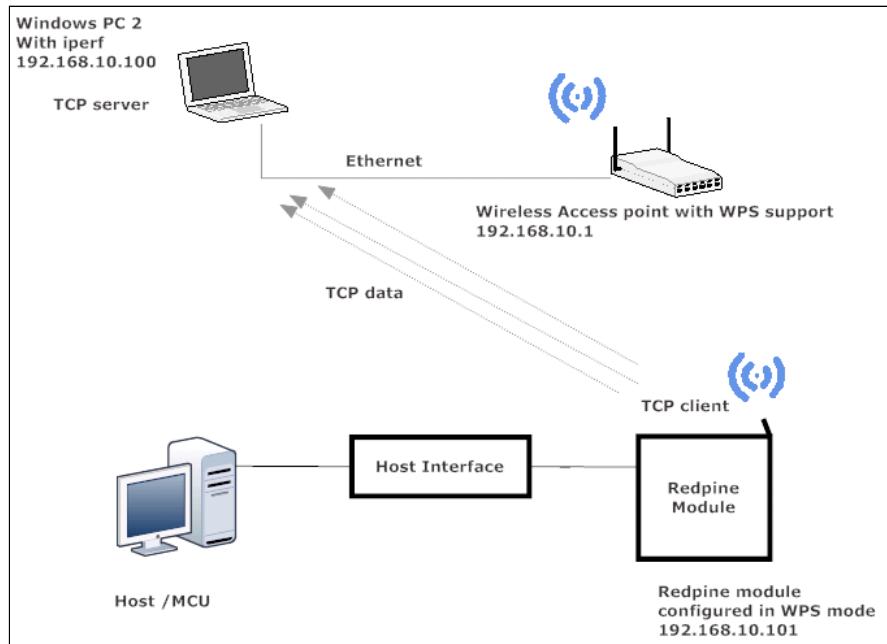
- Configure device as WPS enabled station
- Enable WPS PIN method in Access Point
- Connect to WPS enabled Access Point using PIN method
- Open TCP client socket after successful connection
- Establish TCP connection with the TCP server opened in remote peer
- Send TCP data from WiSeConnect device to remote peer

### 8.39.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC2 with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- WPS supported Access Point
- Windows PC2
- Redpine module
- A TCP server application running on the Windows PC2 (This example uses iperf for windows )



**Figure 1: Setup Diagram**

#### 8.39.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_wps\_station.c** file and update/modify following macros:

**SSID** refers to the name of the Access point. In WPS method it should NULL.

```
#define SSID NULL
```

**SECURITY\_TYPE** refers to the type of security. In WPS method Redpine module supports WPS push button method and WPS PIN method.

In this example, Redpine module connecting to AP using WPS PIN method. To use the access point generated pin, set **SECURITY\_TYPE** macro to **RSI\_WPS\_PIN**.

or if the pin has to be generated by Redpine module and used set **SECURITY\_TYPE** macro to **RSI\_USE\_GENERATED\_WPSPIN**.

```
#define SECURITY_TYPE
RSI_WPS_PIN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes. In WPS PIN method PSK is the pin if user wants to use Access point generated pin

Or

This can be NULL if Redpine Module generated pin is used

```
#define PSK  
"40517175"
```

If the user wants to enter the pin generated by the client device ,then the pin should be entered in the WPS PIN settings in the access point configuration. The pin is generated in the join response. After entering the pin ,the device will be added successfully.

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT  
port> <local
```

**SERVER\_PORT** portrefers remote TCP server port number which is opened in Windows PC2.

```
#define SERVER_PORT  
port> <remote
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with TCP server socket.  
IP address should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.100" as IP address, update the macro **DEVICE\_IP** as **0x640AA8C0**.

```
#define SERVER_IP_ADDRESS  
0x640AA8C0
```

**NUMEBR\_OF\_PACKETS** refers how many packets to receive from TCP client

```
#define NUMBER_OF_PACKETS  
packets> <no of
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN & 8000
```

To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE & 1
```

**(OR)**

If user wants to configure STA IP address through STATIC, then set **DHCP\_MODE** macro to "**0**" and configure the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP  
0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order.

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY  
0X010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order.

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**.

```
#define NETMASK  
0X00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE  
RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP  
FEAT_SECURITY_OPEN  
#define RSI_TCP_IP_BYPASS  
RSI_DISABLE  
#define RSI_TCP_IP_FEATURE_BIT_MAP  
TCP_IP_FEAT_DHCPV4_CLIENT  
#define RSI_CUSTOM_FEATURE_BIT_MAP 0  
#define RSI_BAND  
RSI_BAND_2P4GHZ
```

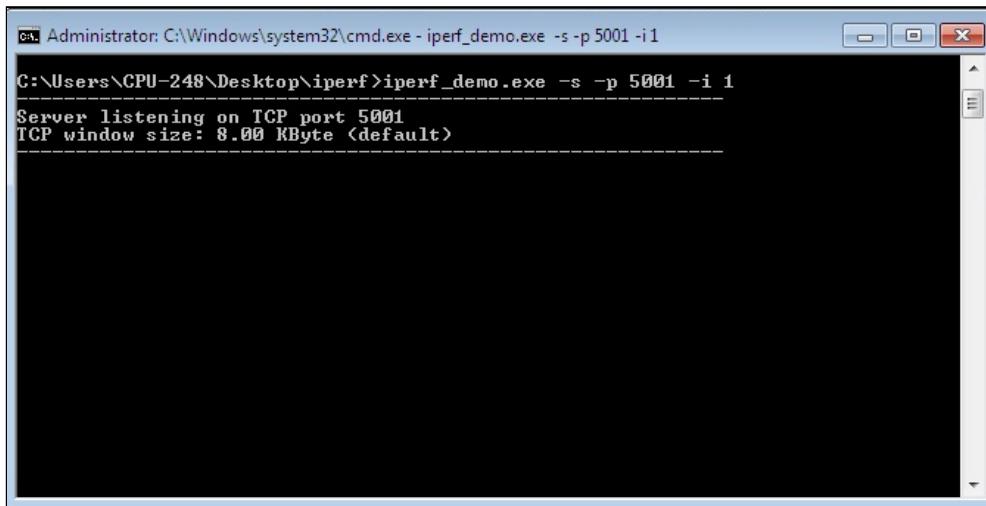
**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

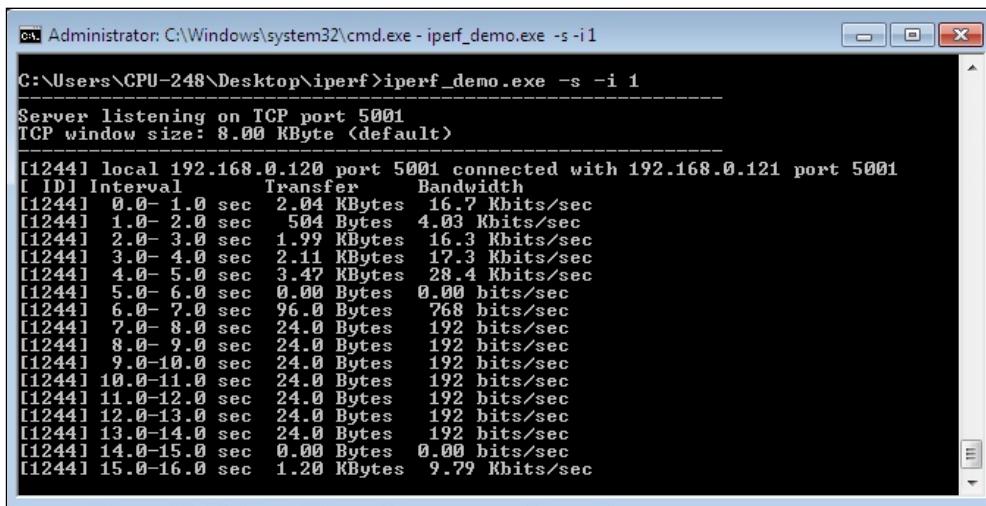
1. Configure the Access point in WPS mode to connect Redpine module in STA mode.
2. Open TCP server application using iperf application in Windows PC1 which is connected to the access point through LAN.
3. Users can download application from the link below,  
<https://iperf.fr/iperf-download.php#windows>

**iperf\_demo.exe -s -p <SERVER\_PORT> -i 1**



```
Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -p 5001 -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -p 5001 -i 1
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
```

4. Enter the WPS pin generated in access point for successful connection and start data transfer.



```
Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -i 1
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
[12441] local 192.168.0.120 port 5001 connected with 192.168.0.121 port 5001
[12441]   Interval      Transfer     Bandwidth
[12441]  0.0- 1.0 sec  2.04 KBytes  16.7 Kbits/sec
[12441]  1.0- 2.0 sec  504 Bytes   4.03 Kbits/sec
[12441]  2.0- 3.0 sec  1.99 KBytes  16.3 Kbits/sec
[12441]  3.0- 4.0 sec  2.11 KBytes  17.3 Kbits/sec
[12441]  4.0- 5.0 sec  3.47 KBytes  28.4 Kbits/sec
[12441]  5.0- 6.0 sec  0.00 Bytes   0.00 bits/sec
[12441]  6.0- 7.0 sec  96.0 Bytes   768 bits/sec
[12441]  7.0- 8.0 sec  24.0 Bytes   192 bits/sec
[12441]  8.0- 9.0 sec  24.0 Bytes   192 bits/sec
[12441]  9.0-10.0 sec 24.0 Bytes   192 bits/sec
[12441] 10.0-11.0 sec 24.0 Bytes   192 bits/sec
[12441] 11.0-12.0 sec 24.0 Bytes   192 bits/sec
[12441] 12.0-13.0 sec 24.0 Bytes   192 bits/sec
[12441] 13.0-14.0 sec 24.0 Bytes   192 bits/sec
[12441] 14.0-15.0 sec 0.00 Bytes   0.00 bits/sec
[12441] 15.0-16.0 sec 1.20 KBytes  9.79 Kbits/sec
```

## 8.40 WPS Station Example

### 8.40.1 WPS Overview

WPS (Wi-Fi Protected setup) is a network standard to create a secure wireless home network using PUSH button and PIN button method.

In Push button method, in which the user has to push a button, either an actual or virtual one, on both the access point and the new wireless client device. On most devices, this discovery mode turns itself off as soon as a connection is established or after a delay (typically 2 minutes or less).

---

In PIN method, in which the user has to enter secret PIN on both the access point and the new wireless client device. On most devices, this discovery mode turns itself off as soon as a connection is established or after a delay (typically 2 minutes or less).

#### 8.40.2 Example Overview

##### Overview

Redpine Module supports both Push button method and WPS pin method to connect to an Access point. The WPS station application demonstrates how Redpine module would be connected to secured (WPA2) Access point using WPS PUSH method.

In this application, after successful connection with the access point using WPS PUSH button method, Redpine device opens TCP socket and sends TCP data to the configured remote peer.

##### Sequence of Events

This Application explains user how to:

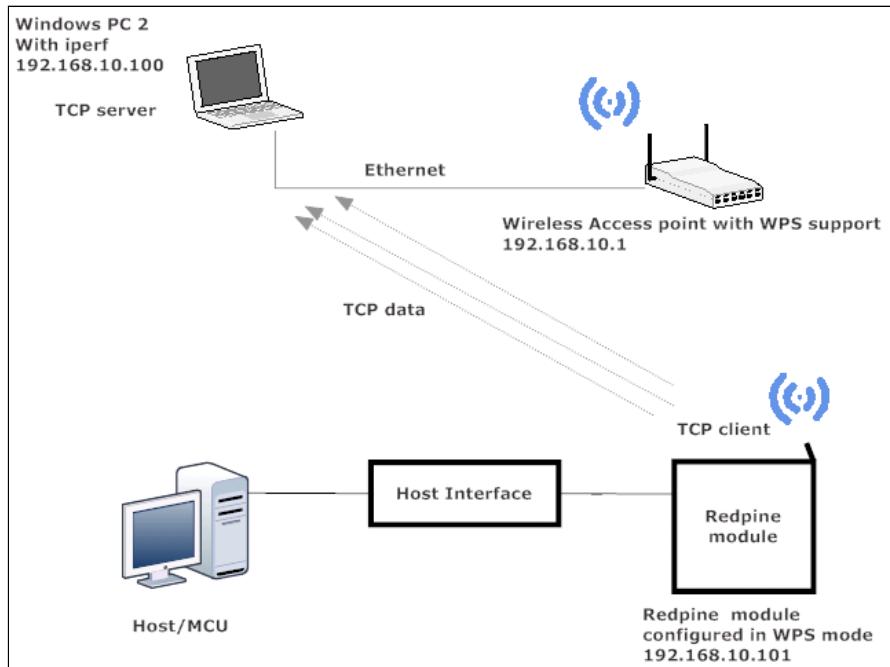
- Configure device as WPS enable station
- Enable WPS PUSH button method in Access Point
- Connect to WPS enabled Access Point using PUSH button method
- Open TCP client socket after successful connection
- Establish TCP connection with the TCP server opened in remote peer
- Send TCP data from WiSeConnect device to remote peer

#### 8.40.3 Example Setup

The WiSeConnect parts require that the host processor should be connected to the WiSeConnect either using SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

##### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- WPS supported Access Point
- Windows PC2
- Redpine module
- A TCP server application running on the Windows PC2 (This example uses iperf for windows )



**Figure 1: Setup Diagram**

#### 8.40.4 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_wps\_station.c** file and update/modify following macros:

**SSID** refers to the name of the Access point. In WPS method it should NULL.

```
#define SSID NULL
```

**SECURITY\_TYPE** refers to the type of security. In WPS method WiSeConnect module supports WPS push button method and WPS PIN method.

In this examples, WiSeConnect module connecting to AP using WPS push button method. So, set **SECURITY\_TYPE** macro to **RSI\_WPS\_PUSH\_BUTTON**.

```
#define SECURITY_TYPE
RSI_WPS_PUSH_BUTTON
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes. In WPS method PSK is not needed.

```
#define PSK NULL
```

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT          <local  
port>
```

**SERVER\_PORT** port refers remote TCP server port number which is opened in Windows PC2.

```
#define SERVER_PORT          <remote  
port>
```

**SERVER\_IP\_ADDRESS** refers remote peer IP address to connect with TCP server socket.  
IP address should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.100" as IP address, update the macro **DEVICE\_IP** as **0x640AA8C0**.

```
#define SERVER_IP_ADDRESS  
0x640AA8C0
```

**NUMBER\_OF\_PACKETS** refers how many packets to receive from TCP client

```
#define NUMBER_OF_PACKETS      <no of  
packets>
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN        8000
```

To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE               1
```

**Note:**

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure STA IP address through STATIC then set **DHCP\_MODE** macro to "0" and configure following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

IP address to be configured to the device in STA mode should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP  
0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order.

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY  
0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK  
0x00FFFFFF
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE  
RSI_DISABLE  
#define RSI_FEATURE_BIT_MAP  
FEAT_SECURITY_OPEN  
#define RSI_TCP_IP_BYPASS  
RSI_DISABLE  
#define RSI_TCP_IP_FEATURE_BIT_MAP  
TCP_IP_FEAT_DHCPV4_CLIENT  
#define RSI_CUSTOM_FEATURE_BIT_MAP 0  
#define RSI_BAND  
RSI_BAND_2P4GHZ
```

**Note:**

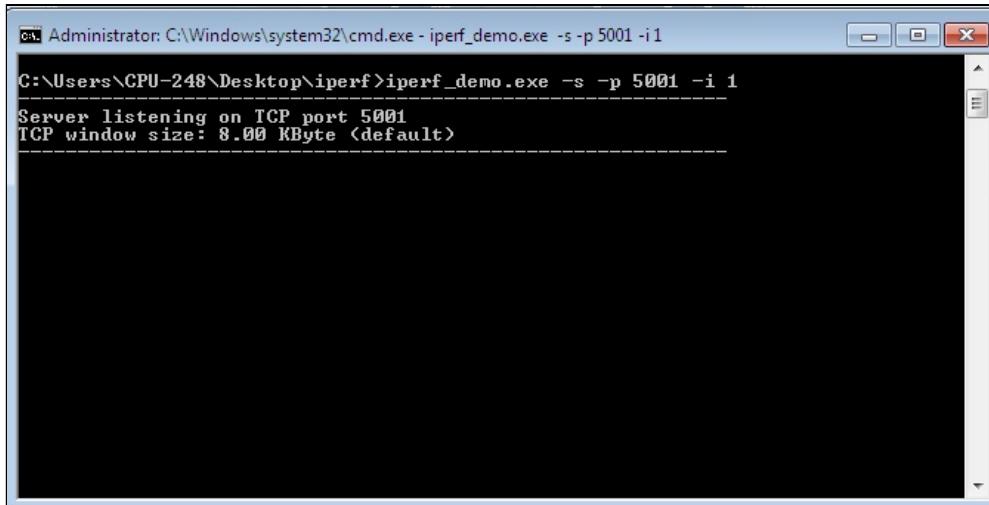
**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. Configure the Access point in WPA2-PSK mode in order to connect Redpine device in STA mode.
2. Open TCP server application using iperf application in Windows PC1 which is connected to Access point through LAN.

3. Users can download application from the link below,  
<https://iperf.fr/iperf-download.php#windows>

**iperf\_demo.exe -s -p <SERVER\_PORT> -i 1**



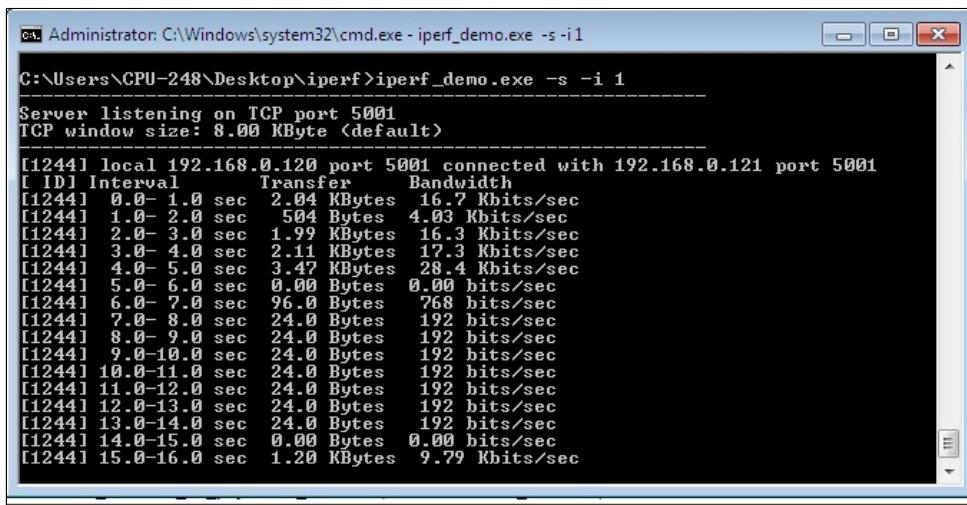
```
C:\ Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -p 5001 -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -p 5001 -i 1
Server listening on TCP port 5001
TCP window size: 8.00 KByte <default>
```

4. Press the WPS Push button on Access point and it lasts for 2 minutes.

5. After program gets executed, Redpine Device would scan and connect to access point using WPS push button method and get IP.

6. After successful connection, Redpine device STA connects to TCP server socket opened on Windows PC1 using TCP client socket and sends configured **NUMBER\_OF\_PACKETS** to remote TCP server.

Please refer the below image for reception of TCP data on TCP server,



```
C:\ Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -i 1
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -i 1
Server listening on TCP port 5001
TCP window size: 8.00 KByte <default>
[12441] local 192.168.0.120 port 5001 connected with 192.168.0.121 port 5001
[ ID] Interval Transfer Bandwidth
[12441] 0.0- 1.0 sec 2.04 KBytes 16.7 Kbits/sec
[12441] 1.0- 2.0 sec 504 Bytes 4.03 Kbits/sec
[12441] 2.0- 3.0 sec 1.99 KBytes 16.3 Kbits/sec
[12441] 3.0- 4.0 sec 2.11 KBytes 17.3 Kbits/sec
[12441] 4.0- 5.0 sec 3.47 KBytes 28.4 Kbits/sec
[12441] 5.0- 6.0 sec 0.00 Bytes 0.00 bits/sec
[12441] 6.0- 7.0 sec 96.0 Bytes 768 bits/sec
[12441] 7.0- 8.0 sec 24.0 Bytes 192 bits/sec
[12441] 8.0- 9.0 sec 24.0 Bytes 192 bits/sec
[12441] 9.0-10.0 sec 24.0 Bytes 192 bits/sec
[12441] 10.0-11.0 sec 24.0 Bytes 192 bits/sec
[12441] 11.0-12.0 sec 24.0 Bytes 192 bits/sec
[12441] 12.0-13.0 sec 24.0 Bytes 192 bits/sec
[12441] 13.0-14.0 sec 24.0 Bytes 192 bits/sec
[12441] 14.0-15.0 sec 0.00 Bytes 0.00 bits/sec
[12441] 15.0-16.0 sec 1.20 KBytes 9.79 Kbits/sec
```

---

## 8.41 Customized root webpage

### 8.41.1 Example Overview

#### Overview

The Custom root webpage application demonstrate how to customize the root webpage. In this application, Redpine device starts as an Access point. After successful creation of AP, User can open the root webpage by connecting to HTTP server running in device.

#### Sequence of Events

This Application explains user how to:

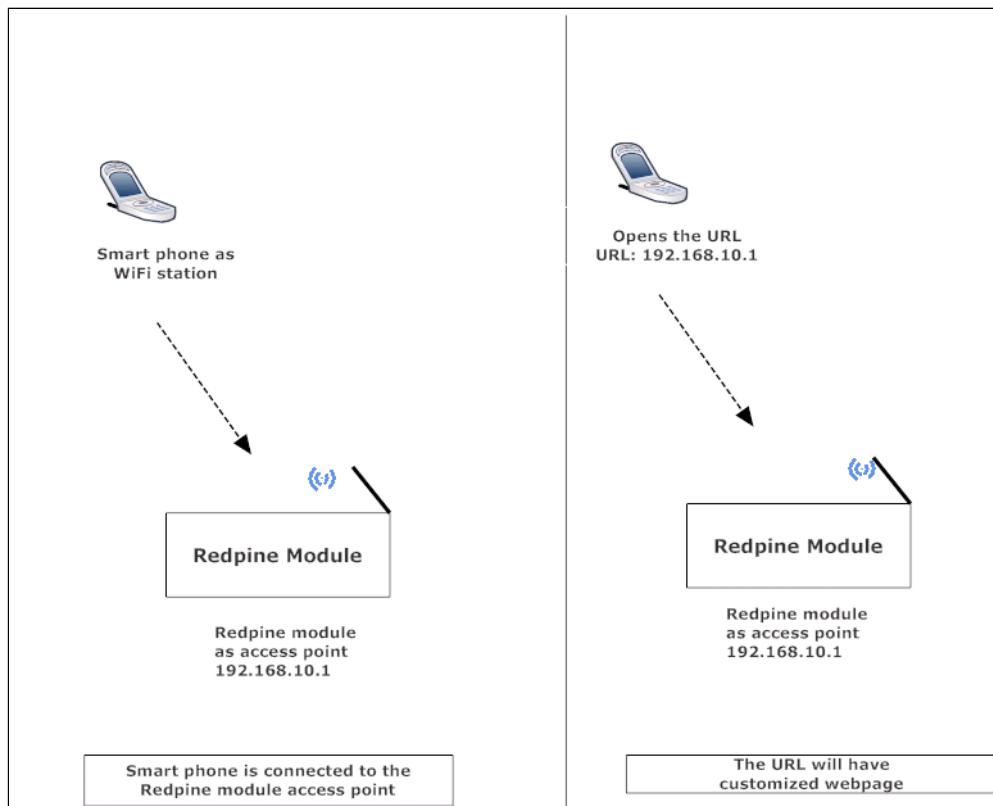
- Start Redpine device as Access Point and load a custom webpage to the root webpage.
- Connect a station to the device and get IP address through DHCP.
- Open root webpage of the Device from the browser of the connected station (STA).
- The root webpage now has the desired page instead of the default page.

### 8.41.2 Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

#### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC1 with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Wireless Access Point
- Smart Phone
- Redpine module



#### 8.41.3 Configuration and Execution of the Application

##### Configuring the Application

1. Open **rsi\_customized\_root\_webpage.c\_file** and update/modify the following macros.  
SSID refers to the name of the Access point.

```
#define SSID           "<ap name>"
```

CHANNEL\_NO refers to the channel in which AP would be started

#define CHANNEL_NO	11
--------------------	----

**Note:**

lid values for **CHANNEL\_NO** are 1 to 11 in 2.4GHz band and 36 to 48 & 149 to 165 in 5GHz band. In this example default configured band is 2.4GHz. So, if user wants to use 5GHz band then the user has to set **RSI\_BAND** macro to 5GHz band in **rsi\_wlan\_config.h** file.

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities.  
Valid configuration is:  
**RSI\_OPEN** - For OPEN security mode

**RSI\_WPA** - For WPA security mode  
**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE RSI_WPA2
```

**ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports OPEN, TKIP, CCMP methods.

Valid configuration is:  
**RSI\_CCMP** - For CCMP encryption  
**RSI\_TKIP** - For TKIP encryption  
**RSI\_NONE** - For open encryption

```
#define ENCRYPTION_TYPE <encryption_type>
```

**PSK** refers to the secret key if the Access point to be configured in WPA/WPA2 security modes.

```
#define SK "1234567890"
```

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons in milliseconds. Allowed values are integers from 100 to 1000 which are multiples of 100.

```
#define BEACON_INTERVAL 100
```

**DTIM\_INTERVAL** refers DTIM interval of the Access Point. Allowed values are from 1 to 255.

```
#define DTIM_INTERVAL 4
```

#### To configure IP address

IP address to be configured in the device should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP 0X010AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY 0X010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

#define NETMASK	0X00FFFFFF
-----------------	------------

**Note:**

In AP mode, configure same IP address for both **DEVICE\_IP** and **GATEWAY** macros

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

#define CONCURRENT_MODE	DISABLE
#define RSI_FEATURE_BIT_MAP	FEAT_SECURITY_PSK
#define RSI_TCP_IP_BYPASS	DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP (TCP_IP_FEAT_DHCPV4_SERVER TCP_IP_FEAT_HTTP_SERVER   TCP_IP_FEAT_EXTENSION_VALID)	
#define RSI_EXT_TCPIP_FEATURE_BITMAP EXT_TCP_IP_HTTP_SERVER_BYPASS	

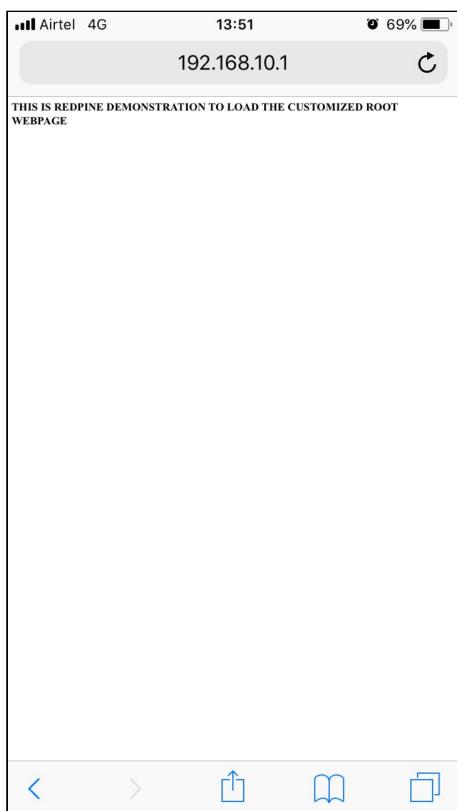
#define RSI_CUSTOM_FEATURE_BIT_MAP	0
#define RSI_BAND	RSI_BAND_2P4GHZ

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders user need not change for each example.

## Executing the Application

1. After the program gets executed, Redpine Device will be started as Access point having the configuration same as that of in the application.
2. Now connect aSmart phone(STA) to Device and get IP address.
3. After successful connection open the provisioning page from STA browser by giving the following URL:  
**URL: DEVICE\_IP**  
**When the webpage is loaded, user can see the desired webpage instead of the default webpage.**



## 9 ZIGBEE

### 9.1 Zigbee Switch

#### 9.1.1 Application Overview

##### Overview

This is a sample application demonstrating a zigbee end device as switch ,which sends toggle command to zigbee Coordinator as Light.

##### Sequence of Events

This Application explains user how to:

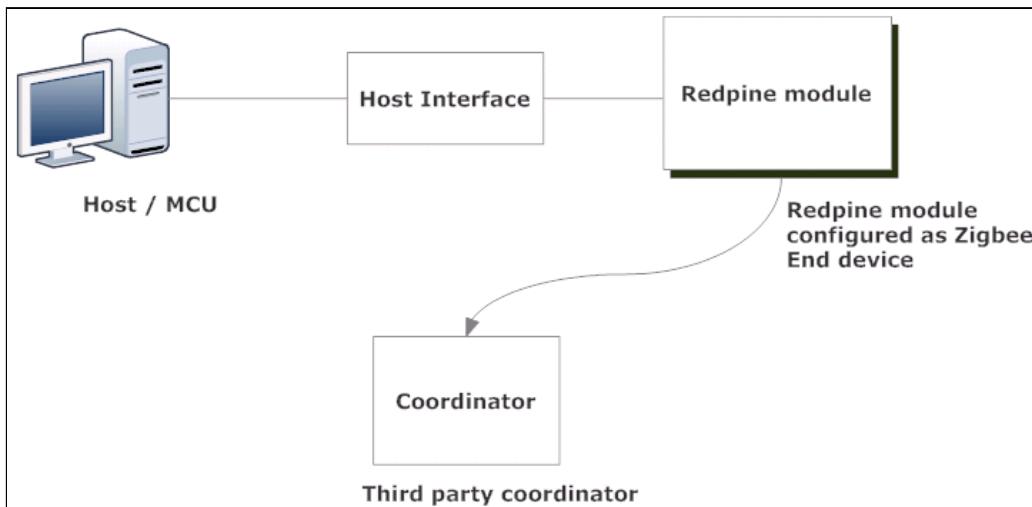
Configure the Zigbee end device.

#### 9.1.2 Example Setup

The WiSeConnect parts require that the host processor is connected to the WiSeConnect using either SPI, UART or USB host interface. The host processor firmware needs to properly initialize the selected host interface. The Redpine Wireless SAPI framework provides necessary HAL APIs to enable variety of host processors. The WiSeMCU parts offer integrated wireless connectivity and does not require host interface initialization.

##### WiSeMCU / WiSeConnect based Setup Requirements

- Windows PC with KEIL or IAR IDE in case of WiSeMCU
- Windows / Linux PC with Host interface(UART/ USB-CDC/ SPI/ USB) in case of WiSeConnect
- Redpine module



#### 9.1.3 Configuration and Execution of the Application

##### Configuring the Application

Open ***rsi\_zb\_config.h*** file and update/modify following macro

**g\_CHANNEL\_MASK\_c** refers to the channel in which the coordinator is operating.

This macro defines a 32-bit mask starting from the 11th bit from the LSB to the 26th bit, consisting of 16 channels represented by each bit.

For example : channel mask for 26th channel

```
#define g_CHANNEL_MASK_c           g_MASK_FOR_26_CHANNEL_C
```

### Executing the Application

1. Power on the WiSeConnect or WiSeMCU module and run the sapis rsi\_wc\_app application.
2. After the program gets executed, Redpine module starts zigbee end device.
3. The end-device connects to the coordinator and sends toggle command in a loop.

## 10 WLAN ZIGBEE

### 10.1 wlan\_ap\_zigbee\_switch

#### 10.1.1 Application Overview

The coex application demonstrates how information can be exchanged seamlessly using two wireless protocols (WLAN and ZB) running in the same device.

##### **Setup required:**

1. Linux PC.
2. ZigBee End Device
3. ZigBee Coordinator.
4. WLAN Access Point.
5. WLAN Station Card.

##### **Description:**

The coex application has WLAN and ZigBee tasks.

ZigBee end device connects to the ZB coordinator.

The socket opened at AP sends message to client which is in listen mode, Wlan task accepts and sends to ZigBee task, which in turn sends command to ZigBee Light.

On reception of data confirmation, the ZigBee task sends “RECVD CNFRM” string to the AP via client task.

Thus messages can be seamlessly transferred between Linux PC and Coordinator.

#### Description

The coex application has WLAN and ZB tasks and acts as an interface between ZB coordinator and PC. ZB coordinator interacts with ZB task, while PC Redpine module create an Access Point, remote PC (with WLAN station interface) connect to Redpine Access Point interacts with WLAN task

When ZB coordinator connects and sends message to Redpine device, ZB task accepts and sends to WLAN task, which in turn sends to PC. Similarly, when PC sends message to Redpine device, the message will be sent to ZB coordinator via ZB task. Thus, messages can be seamlessly transferred between PC and ZB coordinator.

#### Details of the Application

Redpine WLAN acts as Access Point and allow stations to connect and Redpine ZB acts as a end device and tries to connect with ZB coordinator devices.

- The WLAN task (running in Redpine device) mainly includes following steps.
  1. Start up as Access point and allow stations to connect
  2. Establish TCP connection with Remote and performs data exchanges over TCP socket with the peer(PC)
- The ZB task (running in Redpine device) mainly includes following steps.
  1. Start up as ZB end device and connects to ZB coordinator device.
  2. Establish connection with remote device and performs responds for the inputs given by coordinator

WLAN and ZB tasks forever run in the application to serve the asynchronous events.

## 10.1.2 Configuration and Execution of the Application

### Configuring the Application

#### Configuring the WLAN task

1. Open **rsi\_wlan\_ap\_app.c** file and update/modify following macros to start an Access-point.  
**SSID** refers to the name of the Access point .

```
#define SSID                                "<ap  
name>"
```

**CHANNEL\_NO** refers to the channel in which AP would be started

```
#define CHANNEL_NO  
<channel_no>
```

**SECURITY\_TYPE** refers to the type of security .Access point supports Open, WPA, WPA2 securities

```
#define SECURITY_TYPE  
<security_type
```

**ENCRYPTION\_TYPE** refers to the type of Encryption method .Access point supports Open, TKIP, CCMP methods

```
#define ENCRYPTION_TYPE  
<encryption_type
```

**PSK** refers to the secret key if the Access point to be configured in WPA/WPA2 security modes.

```
#define PSK                                  "<psk>"
```

**BEACON\_INTERVAL** refers to the time delay between two consecutive beacons

```
#define GATEWAY  
0x010AA8C0
```

To configure DTIM interval of the Access Point

```
#define DTIM_INTERVAL  
<dtim_interval>
```

To configure static IP address

IP address to be configured to the device should be in long format and in little endian byte order.  
Example: To configure "192.168.10.1" as IP address, update the macro **DEVICE\_IP** as **0x010AA8C0**.

```
#define DEVICE_IP  
0X010AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY  
0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK  
0x00FFFFFF
```

To establish TCP connection and transfer data to the remote socket, configure the below macros. Internal socket port number.

```
#define DEVICE_PORT 5001
```

Port number of the remote server

```
#define SERVER_PORT 5001
```

IP address of the remote server

```
#define SERVER_IP_ADDRESS  
0x020AA8C0
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 10000
```

2. Open TCP server socket on remote peer (PC).  
User can use Test TCP server application from the below link to create TCP Server on remote peer <http://sourceforge.net/projects/sockettest/files/latest/download>  
For example, user can open TCP server socket with port number 5001 on remote peer using the Test TCP server Application.
3. Include rsi\_wlan\_ap\_app.c, rsi\_zb\_app.c and main.c files in the project, build and launch the application.
4. Open **rsi\_wlan\_config.h** file and update/modify following macros,

```
#define CONCURRENT_MODE
RSI_DISABLE
#define RSI_FEATURE_BIT_MAP
FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS
RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP
(TCP_IP_FEAT_DHCPV4_SERVER | TCP_IP_TOTAL_SOCKETS_1)
#define RSI_CUSTOM_FEATURE_BIT_MAP
FEAT_CUSTOM_FEAT_EXTENTION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP
EXT_FEAT_256K_MODE
#define RSI_BAND
RSI_BAND_2P4GHZ
```

## Configuring the ZB Application

1. Configure the below macros in the Application file.  
Open **rsi\_zb\_app.c** file and update/modify following macro  
**g\_CHANNEL\_MASK\_c** refers to the channel in which the coordinator is operating.

This macro defines a 32-bit mask starting from the 11th bit from the LSB to the 26th bit, consisting of 16 channels represented by each bit.

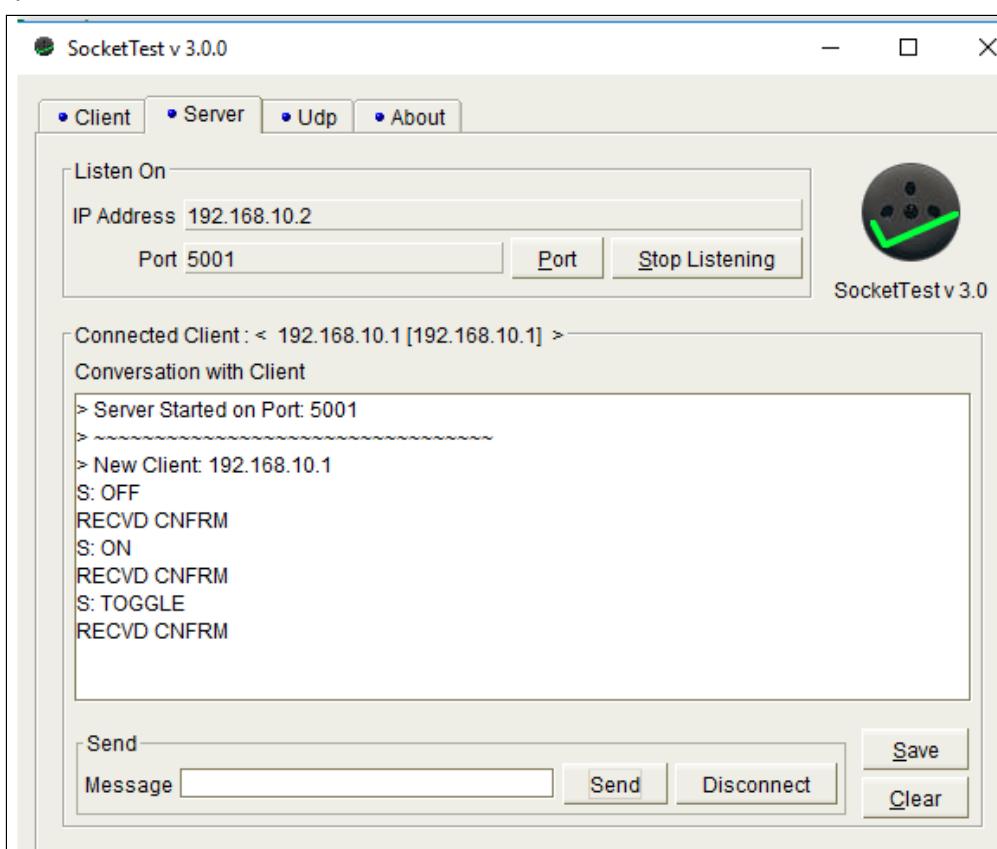
For example : channel mask for 26th channel

```
#define g_CHANNEL_MASK_c      g_MASK_FOR_26_CHANNEL_c
```

## Executing the coex Application

Connect Redpine device to the Windows PC running KEIL or IAR IDE.

1. Build and launch the application.
2. After the program gets executed, Redpine ZB is in scan state and WLAN will create an Access Point and wait for remote station to connect.
3. Remote PC can connect to device Access point and device will establish TCP connection with peer (PC).
4. Now initiate connection from the ZB coordinator.
5. After ZB connection is established, send a message from the App to Redpine ZB. Observe this message on the PC (Socket Test GUI) connected via TCP socket with Redpine WLAN.
6. Now, send a message from PC (Socket Test GUI) to Redpine WLAN via TCP socket and observe the same in the ZB coordinator.
7. `rsi_zb_app_send_to_wlan()` function defined in `rsi_wlan_ap_app.c` to send message from ZB task to WLAN task.
8. With the help of wlan task, message is transferred to PC.
9. Message from PC to WLAN task via socket and `rsi_wlan_app_send_to_zb()` function defined in `rsi_zb_app.c` called asynchronously to send message from WLAN task to ZB task. From ZB task message transferred to smart phone.



## 10.2 wlan\_zigbee\_switch

### 10.2.1 Application Overview

The coex application demonstrates how information can be exchanged seamlessly using two wireless protocols (WLAN and ZigBee) running in the same device.

#### Setup required

1. Linux PC.

- 
2. ZigBee Light Coordinator.
  3. WLAN Access Point and a Linux PC with open ssl support .

## Description

The coex application has WLAN and ZigBee tasks. ZigBee end device connects to the coordinator. The Client connects to the AP. The socket opened at AP sends message to client which is in listen mode, WLAN task accepts and sends to ZigBee task, which in turn sends command to ZigBee Light. On reception of data confirmation, the ZigBee task sends “RECVD CNFRM” string to the AP via client task. Thus messages can be seamlessly transferred between Linux PC and Coordinator.

## Details of the Application

WiSeConnect/ WiSeConnect Plus WLAN acts as a Station and connects to an Access Point WiSeConnect/ WiSeConnect Plus ZigBee acts as a HA switch device.

Initially, ZigBee end-device is connects to the coordinator and then command exchanges occur as follows.

- The WLAN task (running in WiSeConnect/ WiSeConnect Plus device) mainly includes following steps.
  - a. Connects to a Access Point
  - b. Exchanges data over SSL socket with the peer(Linux PC) .
- The ZigBee task (running in WiSeConnect/ WiSeConnect Plus device) mainly includes following steps.
  - a. Receives the command from wlan task.
  - b. Sends the command to the HA light.
  - c. On receiving confirmation sends back confirm string to the AP.
- WLAN and ZigBee tasks forever run in the application to serve the asynchronous events.

### 10.2.2 Configuring the WLAN task

Edit the **rsi\_wlan\_app.c** file in the following path to establish connection with the Access-point.

**host/sapis/examples/wlan\_zigbee/wlan\_ap\_zigbee\_switch**

1. From given configuration,  
**SSID** refers to the Access point to which user wants to connect.  
**SECURITY\_TYPE** is the security type of the Access point.  
**PSK** refers to the secret key if the Access point is configured in WPA/WPA2 security modes.

```
#define SSID "<ap_name>"  
#define SECURITY_TYPE <security-type>  
#define PSK ""
```

2. Load the SSL CA- certificate using **rsi\_wlan\_set\_certificate** API after wireless initialization.

**Note:**

**rsi\_wlan\_set\_certificate** expects the certificate in the form of linear array. Python script is provided in the release package named “certificate\_script.py” in the following path “Host MCU/examples/wlan/certificates” to convert the pem certificate into linear array

Example: If the certificate is ca-cert.pem, give the command as  
python certificate\_script.py ca-cert.pem

The script will generate cacert.pem, in which one linear array named *cacert* contains the certificate.

3. Enable/Disable DHCP mode  
1 – Enables DHCP mode (gets the IP from DHCP server)

0 – Disables DHCP mode

```
#define DHCP_MODE 1
```

4. If DHCP mode is disabled, then change the following macros to configure static IP address IP address to be configured to the device should be in long format and in little endian byte order. Example:  
To configure “192.168.10.101” as IP address, update the macro **DEVICE\_IP** as **0x650AA8C0**.

```
#define DEVICE_IP 0x650AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order Example:  
To configure “192.168.10.1” as Gateway, update the macro **GATEWAY** as **0x010AA8C0**.

```
#define GATEWAY 0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order Example: To configure “255.255.255.0” as network mask, update the macro **NETMASK** as **0x00FFFFFF**.

```
#define GATEWAY 0x010AA8C0
```

5. To establish TCP connection and transfer data to the remote socket configure the below macros. If SSL is enabled, open the socket with protocol type as 1.  
Internal socket port number.

```
#define DEVICE_PORT 5001
```

Port number of the remote server

```
#define SERVER_PORT 5001
```

IP address of the remote server

```
#define SERVER_IP_ADDRESS 0x650AA8C0
```

To enable SSL on socket creation (on device)

```
#define SSL 1
```

Number of packets to send

```
#define NUMBER_OF_PACKETS 1000
```

Application memory length which is required by the driver

```
#define GLOBAL_BUFF_LEN 8000
```

6. Include rsi\_wlan\_app.c , rsi\_zigbee\_app.c and main.c files in the project, build and launch the application

7. Open SSL server socket on remote machine

For example, to open SSL socket with port number 5001 on remote side, use the command as given below.  
**openssl s\_server -accept<SERVER\_PORT> -cert  
<server\_certificate\_file\_path> -key  
<server\_key\_file\_path> -tls<tls\_version>**

**Example: openssl s\_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1\_2**

**Note:**

All the certificates are given in the release package path: Host\_MCU/examples/wlan/certificates

Then type commands

TOGGLE - To send TOGGLE command to the HA Light.

ON - To send On command to the HA Light.

OFF - To send OFF command to the HA Light.

On successfully reception of commands the HA Light replies back with “RECVD CNFRM” string.

### 10.2.3 Configuring the ZigBee Application

1. Edit the **rsi\_zb\_app.c** file in the following path of the Application **host/sapis/examples/wlan\_zigbee/wlan\_ap\_zigbee\_switch**
2. Edit **rsi\_zb\_app.c** and update the following parameters of the device to establish connection with the Coordinator.  
From given configuration, **g\_CHANNEL\_MASK\_c** refers to the channel in which the coordinator is operating. This macro defines a 32-bit mask starting from the 11th bit from the LSB to the 26th bit, consisting of 16 channels represented by each bit.  
For example : channel mask for 26th channel

```
#define g_MASK_FOR_26_CHANNEL_c 0x04000000
#define g_CHANNEL_MASK_c g_MASK_FOR_26_CHANNEL_c
```

Run the application, the end-device connects to the coordinator and sends command to the coordinator as received from the wlan remote client.

## 11 Revision Record

S.No.	Version number	Date	Change
1	v1.0	October 2017	Initial version
2	v1.1	June 2018	Added New API's
3	v1.2	July 2018	Added rsi_wlan_pmk_generate API
4	v1.3	July 2018	Added rsi_setsockopt API
5	v1.4	August 2018	Added rsi_accept_async API
6	v1.5	October 2018	Added PMK feature in <b>station ping</b> Application <a href="http://192.168.1.215:8090/display/RPD/Station+Ping+Example">http://192.168.1.215:8090/display/RPD/Station+Ping+Example</a> Added a2dp burst mode command and more data req event A2DP_BURST_MODE should be 1 Added BT A2DP Sink Example