|  |  |
| --- | --- |
|  |  |

WIRELESS COMMUNICATION INTERFACE SPECIFICATIONS:

1. 1. Range: at least 30m urban (non-line of sight).

(\*typical distance between a house and a detached garage in Regina, Saskatchewan)



1. Minimum Throughput 2.5Kbps (Data) (assuming 144Hz sample rate for Current Data)

RANGE TESTING WITH XCTU:

DESCRIPTION:

Digi XCTU is a configuration and testing software that will be used to test the network performance under a variety of cases and scenarios.

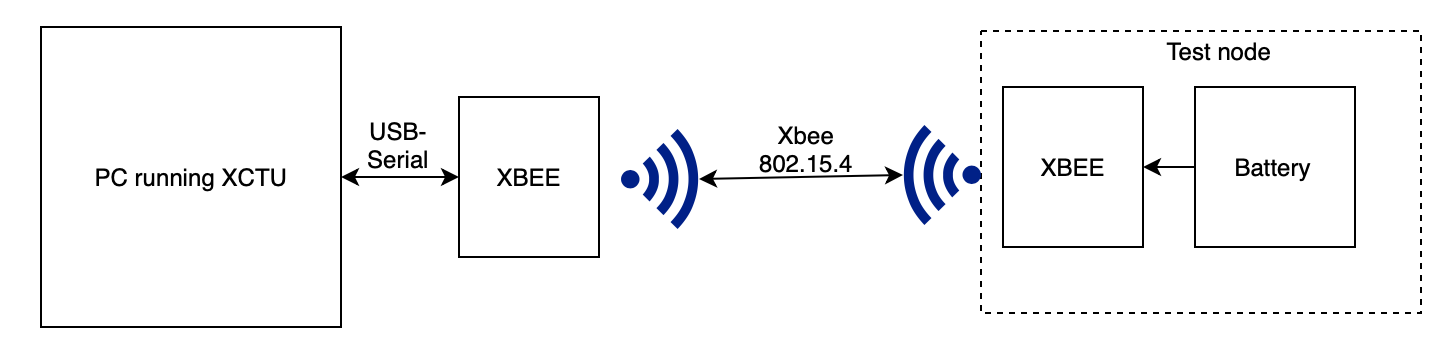
TEST REQUIREMENTS:

* PC with Xbee X-CTU
* 2x Xbee RF Nodes
* 1x USB to Serial Adaptor

INSTALL DIGI XCTU:

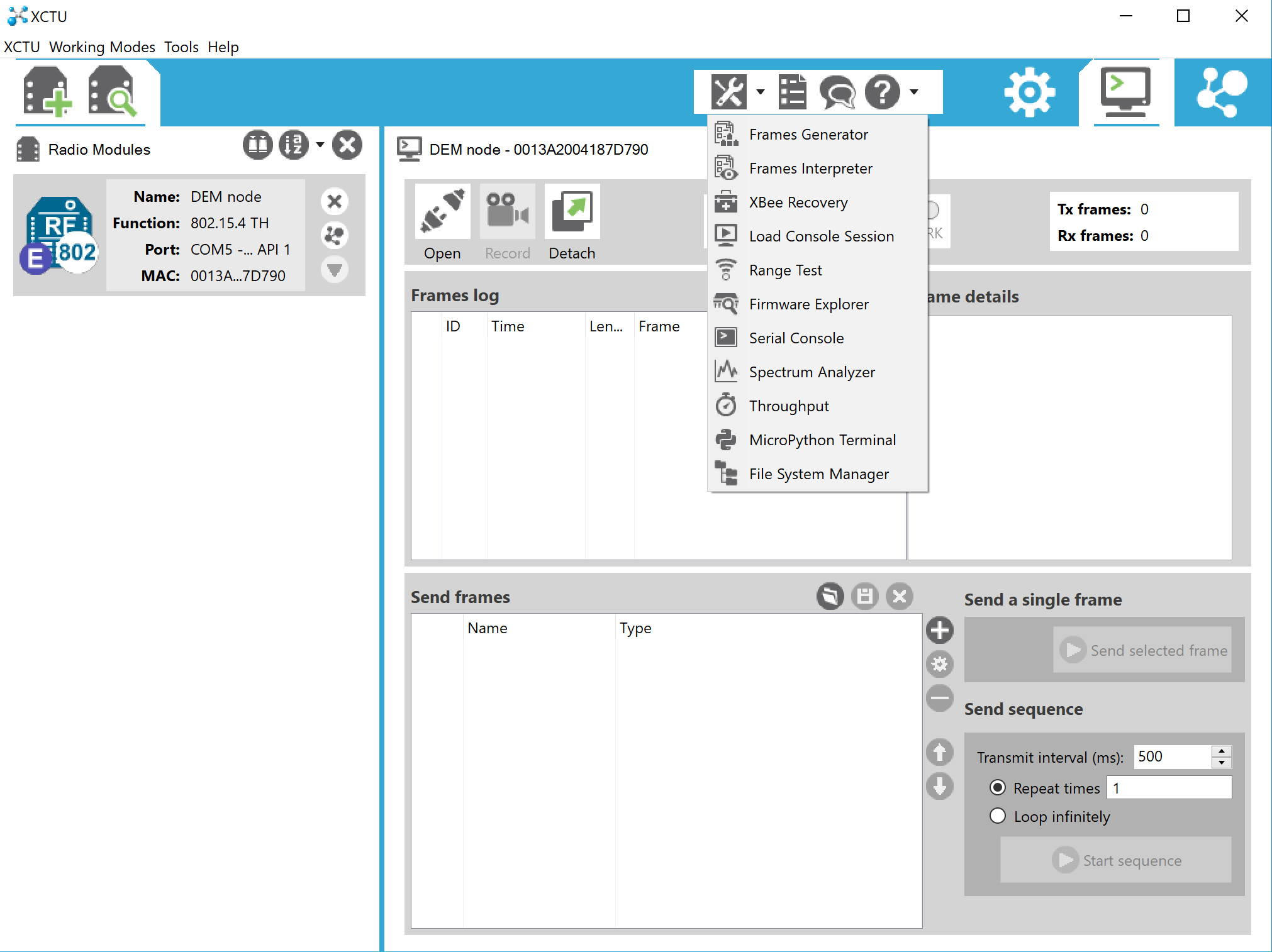
Digi XCTU can be downloaded and installed from <https://www.digi.com/products/iot-platform/xctu>

TEST SETUP:



PROCEDURE:

* Set up the Remote node as shown in the figure above.
* Connect the other Xbee to a PC running XCTU with a serial to USB connector. Ensure all the pins are properly connected (TX->RX, 3.3V->Vcc, Gnd->Gnd)
* Launch Digi XCTU and click add radio, select the serial port and connect to the attached radio.
* Select Range Test from the Tools drop down menu



* Select the local node and the remote node



- Change the Range Test Type to Loopback and set the number of packets to 100 packets.

- Set the Rx and Tx timeouts to 1000ms as shown in the figure above.

- Move the remote node to the required test distance and click on start test.

- Record results in the table below:

TEST CASES:

Test Case 1:

|  |  |
| --- | --- |
| Location: | **University of Regina, Education Building, 4th Floor** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Trial | Distance | Remote Power | Packets Sent | Packets Lost  (Packet loss rate) | Tx Errors |
| 1 | 22.86m (non-line of sight | -83dBm | 100 | 28 | 44 |
| 2 | 62m (line of sight) | -73dBm | 100 | 6 | 30 |
| 3 | - | - | - | - | - |

Comments:

The remote node was powered using an STM32F100RB microcontroller connected to a laptop. The loopback connection was done in code on the STM board.

Test Case 2:

|  |  |
| --- | --- |
| Location: | **Residential, 2-bedroom basement apartment in Harbor Landing** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Trial | Distance | Remote Power | Packets Sent | Packets Lost  (Packet loss rate) | Tx Errors |
| 1 | 12.50m | -74dBm | 100 | 0 | 2 |
| 2 | 12.50m | -82dBm | 100 | 0 | 2 |
| 3 | 20.21m | -86dBm | 100 | 0 | 1 |
| 4 | 20.21m | -84dBm | 113 | 6 | 6 |
| 5 | 33.50m | -76dBm | 122 | 5 | 34 |

Comments:

These tests were done with the central node in a basement apartment of the house, so the test node was always one floor above the main receiving node and non-line of sight in all scenarios. Distance measurements were estimated using Google Maps and are shown in the figures below.

Test Scenarios:

|  |  |
| --- | --- |
| Trial 1 & 2 | Indoor Layout:    Distance Estimation: 12.5m |
| Trial 3 & 4 | Distance Estimation: |
| Trial 5 | Distance Estimation: 33.50m |

Result Summary:

No connection loss was observed in any of the test situation, although distances and environmental factors affected communications we can compensate for these effects in system operation.

**\*\*XCTU screen captures for these tests are included in Appendix A below.**

Team member 1 Team member 2 Team member 3 Supervisor

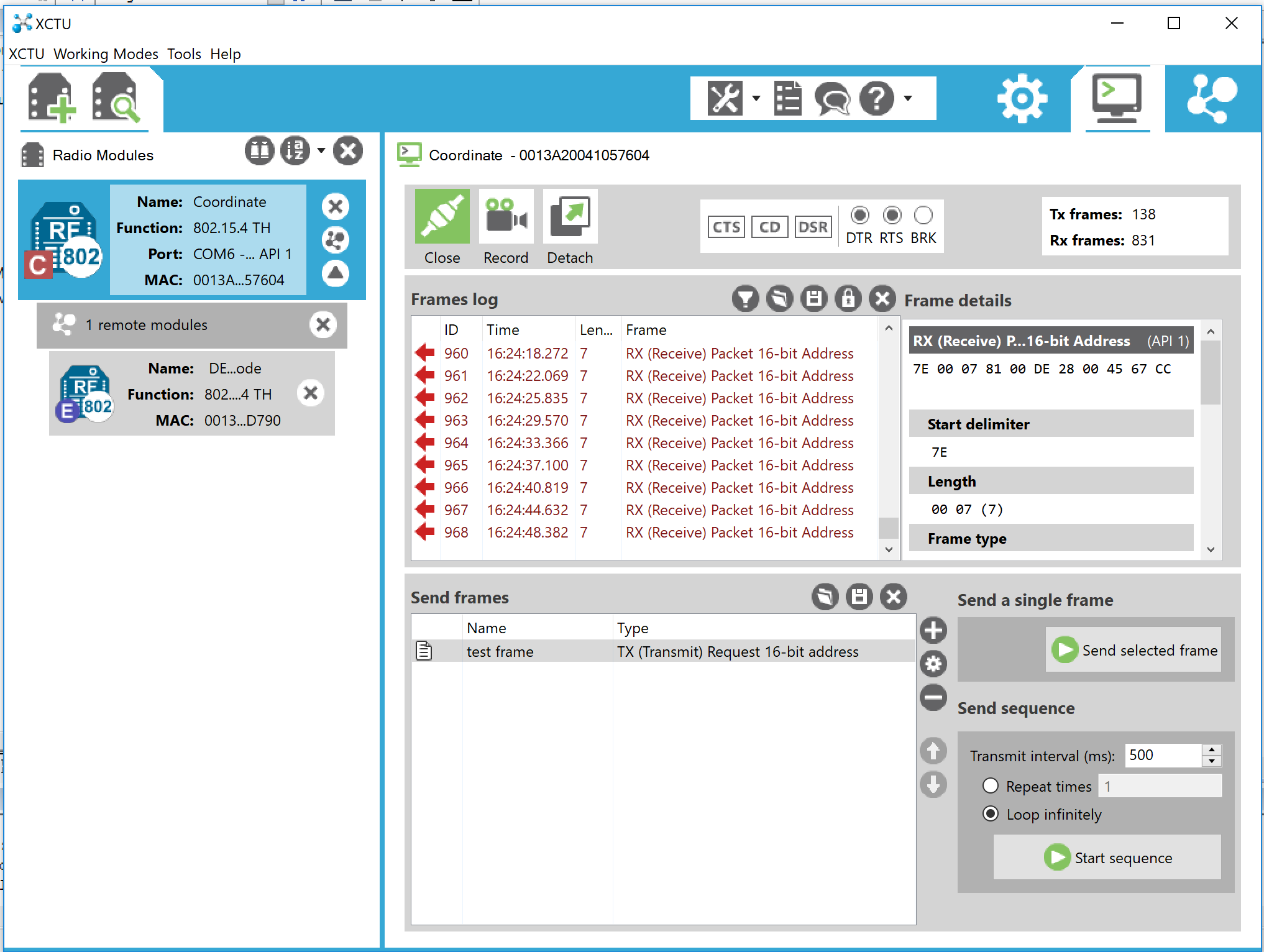
DEM COMMUNICATION INTERFACE TESTING:

TEST REQUIREMENTS:

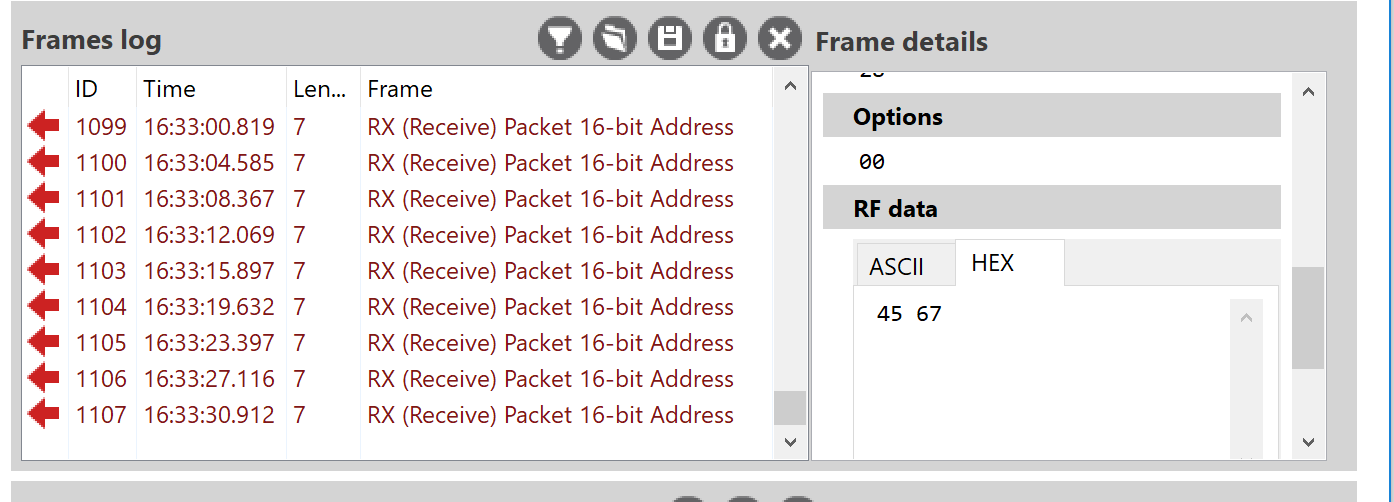
* PC running Digi XCTU and Keil uvision
* 1x TTYL serial to USB connector
* 2 Digi Xbee Radios
* STM32F100RB Microcontroller

PROCEDURE:

* Both Xbee Radios have been configured as required. Connect the Coordinator Xbee to the PC using the provided TTYL serial to USB connector.
* Launch Digi XCTU and click on the add radio icon
* Select the appropriate serial port for the radio
* Once the radio is connected, Switch to console working mode and open a serial connection with the radio
* Connect the end node radio to the STM microcontroller; Xbee RX to PA2, TX to PA3, Xbee Vcc to 3.3V and Ground to any ground pin on the microcontroller
* Launch the Xbee test keil uvision project and build
* Flash the program to the microcontroller and press the reset button on the microcontroller
* You should begin to register received packets on Digi XCTU



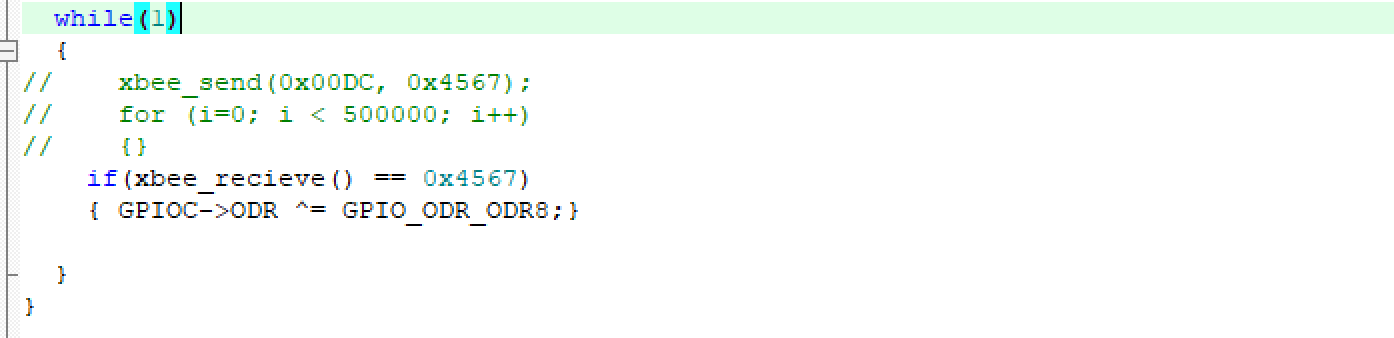
* Verify that the data received is the same as sent from the STM microcontroller. This is done by inspecting the received frame details on XCTU as shown below



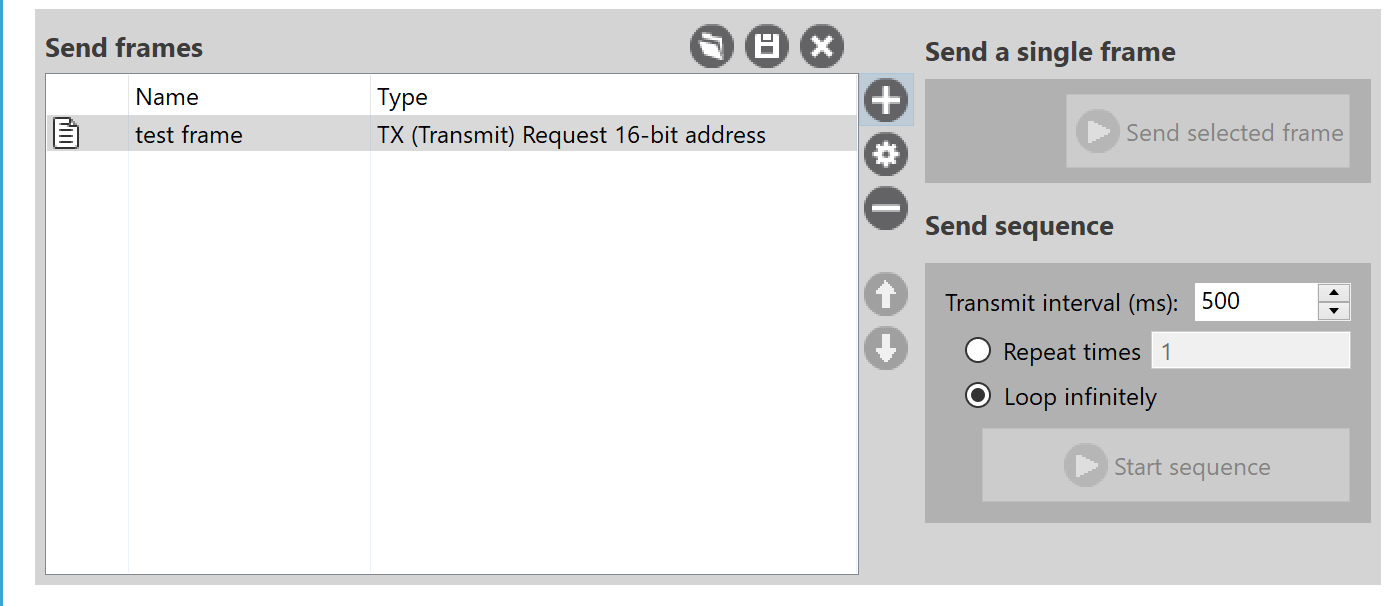
* Record the received data, should be 0x4567 (Arbitrarily Chosen). You can modify the program to test with different 2-byte data.

|  |  |
| --- | --- |
| Expected outcome: | Received Data = 0x4567 |
| Result: | Received Data = 0x4567 |

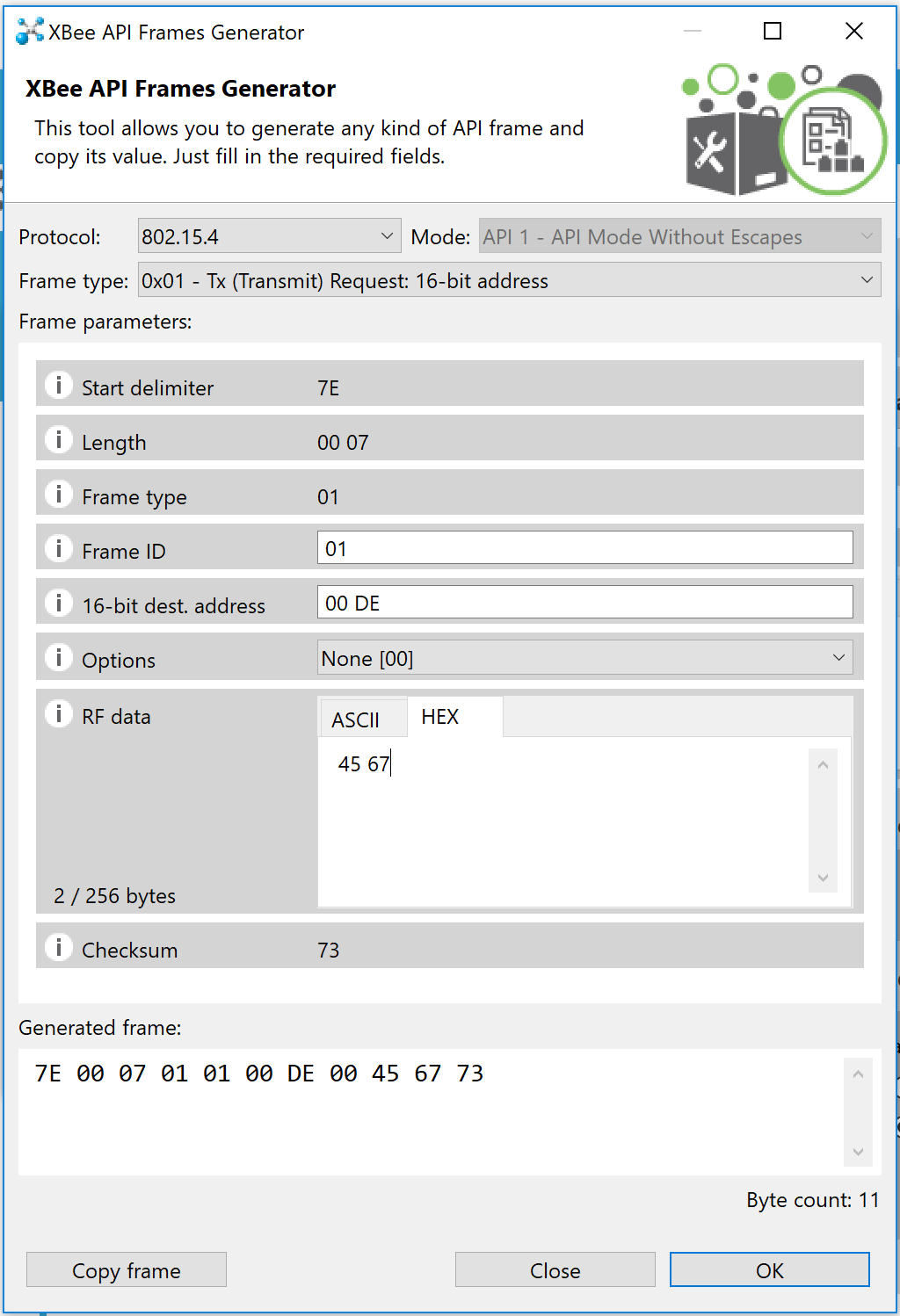
* Uncomment the code segment for receiving data and comment on the sending code segment as shown below. This code segment toggles the LED on PC8 of the microcontroller if the received data is equal to 0x4567



* Run the program.
* On Digi XCTU, click on the plus (+) sign shown in the figure below to make a new frame.



* Select a frame name and click “Create frame using frame generator tool”
* Set the frame type to 0x01-Tx (Transmit) Request: 16-bit address
* Set the destination address 0x00DE (this is the address of the DEM Radio)
* Set the data to “45 67” Hex.



* Click OK and click add frame.
* Click Start sequence to start transmitting the frame. Check that the blue LED on the microcontroller toggles.

|  |  |
| --- | --- |
| Expected outcome: | Blue LED on the STM microcontroller toggles |
| Result: | Test passed. Blue LED toggles. |

Team member 1 Team member 2 Team member 3 Supervisor

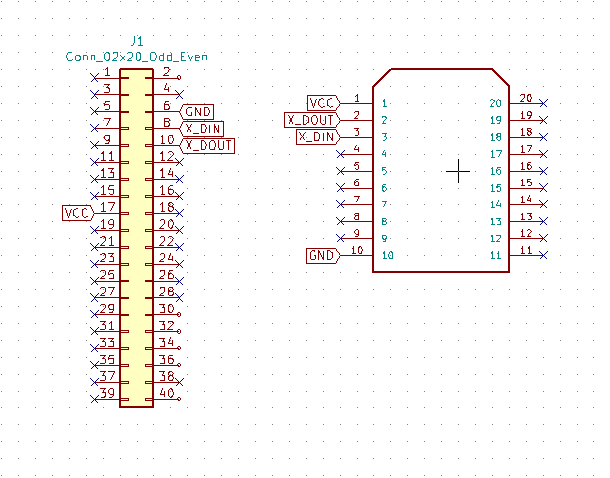
END-TO-END TESTING:

TEST REQUIREMENTS:

* PC running keil uvision.
* 1x ttyl serial to usb connector
* 2x Digi Xbee Radios
* 1x Raspberry Pi
* 1x STM32F100RB

PROCEDURE:

* Both Xbee Radios have been configured as required. Connect the Coordinator Xbee to the raspberry pi as shown in the figure below.



* Connect the end node xbee radio to the STM32F100RB microcontroller; Xbee RX to PA2, TX to PA3, Xbee Vcc to 3.3V and Ground to any ground pin on the microcontroller
* Open the DEM comms folder and launch the project on keil uvision.
* Run the program and flash the target. The program toggles the blue LED on the STM microcontroller if the received data is equal to 0x5890 (Arbitrarily chosen).
* Press the reset button on the STM microcontroller.
* Turn the raspberry pi. And open a terminal window.
* Navigate to the xbeetest directory using ***cd Desktop/xbeetest***command.
* Run main.py using typing ***python main.py***

|  |  |
| --- | --- |
| Expected outcome: | Blue LED on the STM microcontroller toggles |
| Result: | Test passed. Blue LED toggles. |

Team member 1 Team member 2 Team member 3 Supervisor

INTEGRATION TESTING:

DESCRIPTION:

The aim of this test is to test the remote appliance switching and reading of current sensor simultaneously. A test program will run on the DEM and continuously send current readings to the CCU and also listen for switching commands. Another program will run on the CCU and will be responsible for displaying the received current readings and periodically send switching signals to the DEM. Both systems will have counters for indicating the number of transmitted and received packets to determine the throughput of the communications network in a variety of situations and a range of distances. For accuracy, the data received can be compared to measurements from an ammeter.

TEST REQUIREMENTS:

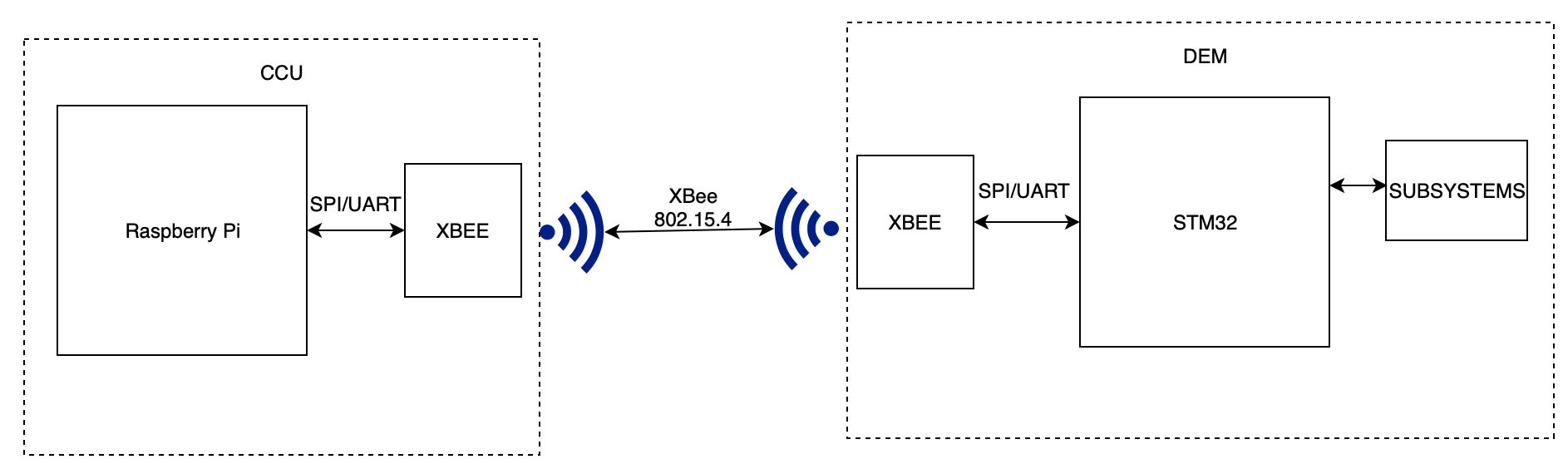
Hardware:

* CCU
* DEM prototype with subsystems (CT and switching Circuit)

Software:

* Commstest.py
* Commstestnode.c

TEST SETUP:

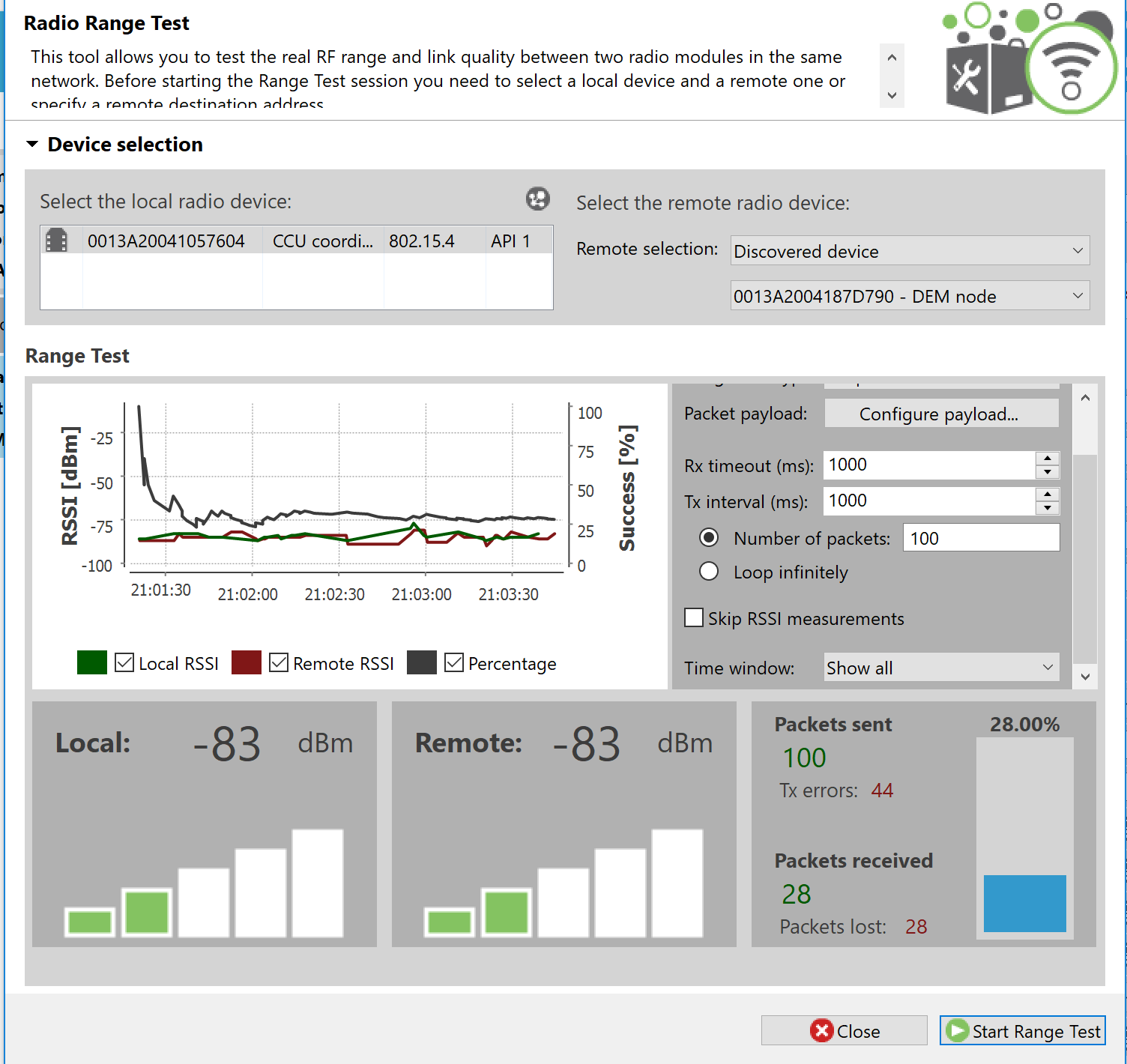


Team member 1 Team member 2 Team member 3 Supervisor

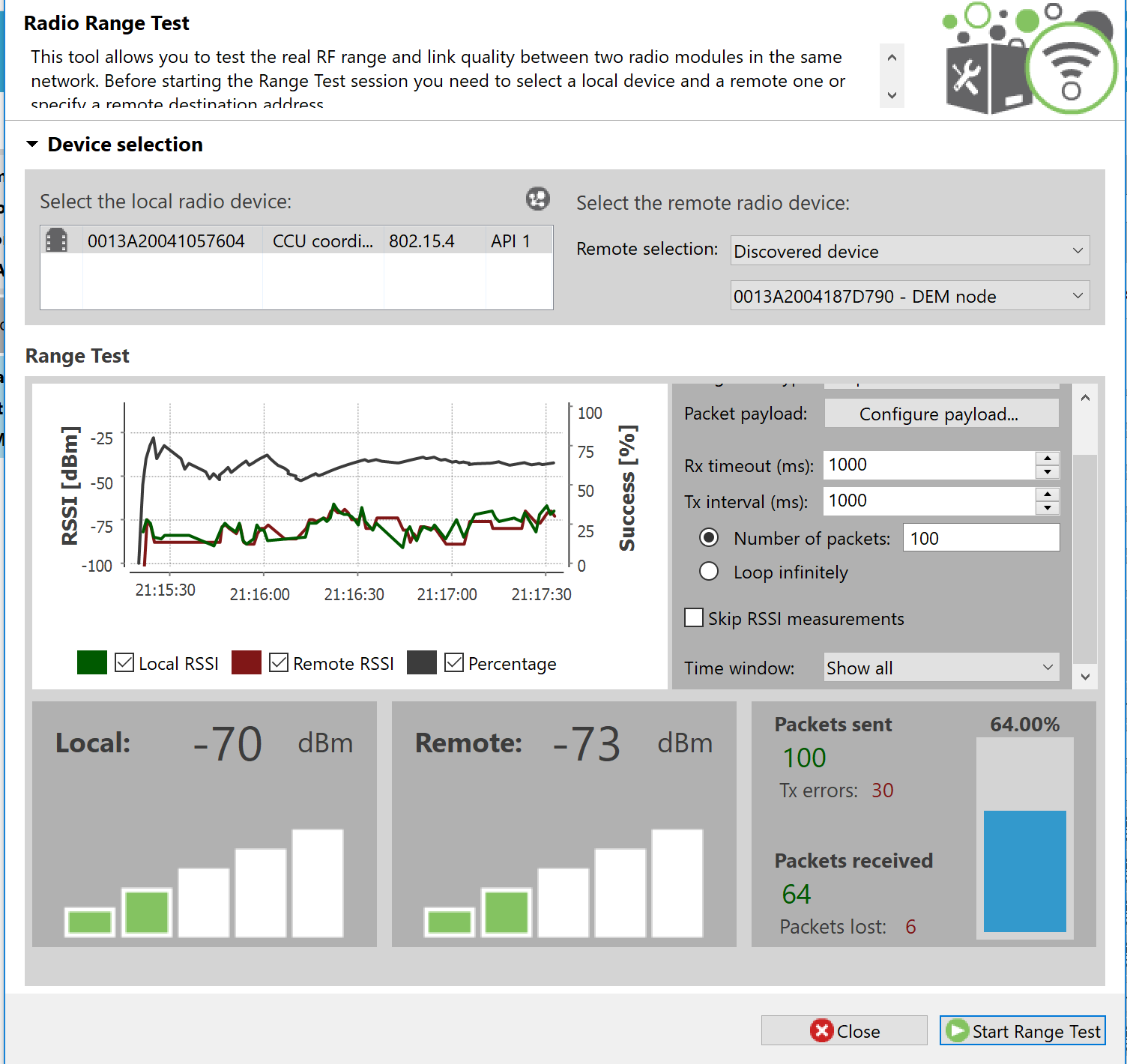
**APPENDIX A**

**Range Testing Data: (University of Regina Education building 4th floor)**

1. 22.86m (non-line of sight:

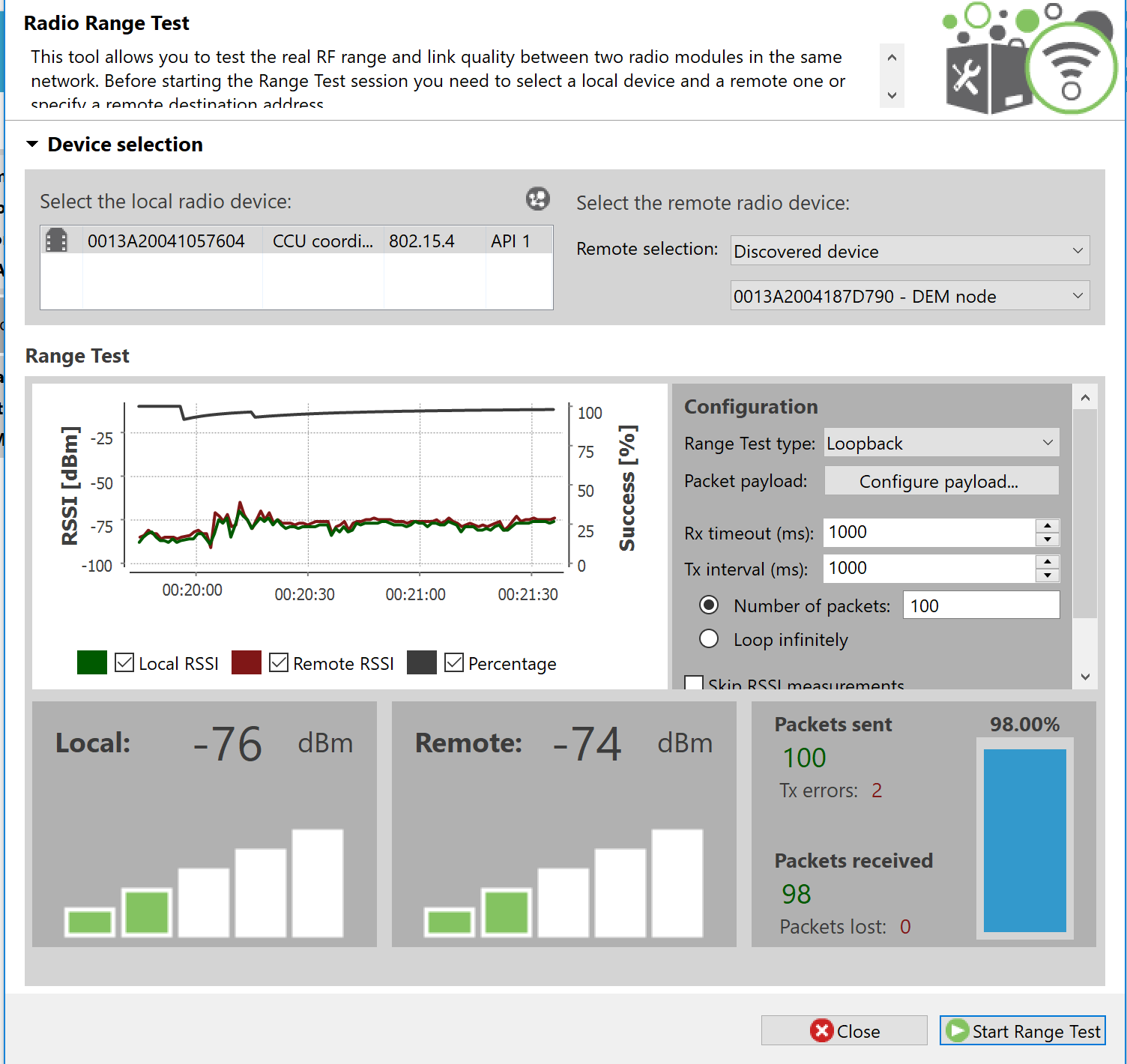
****

1. 62m (line of sight):

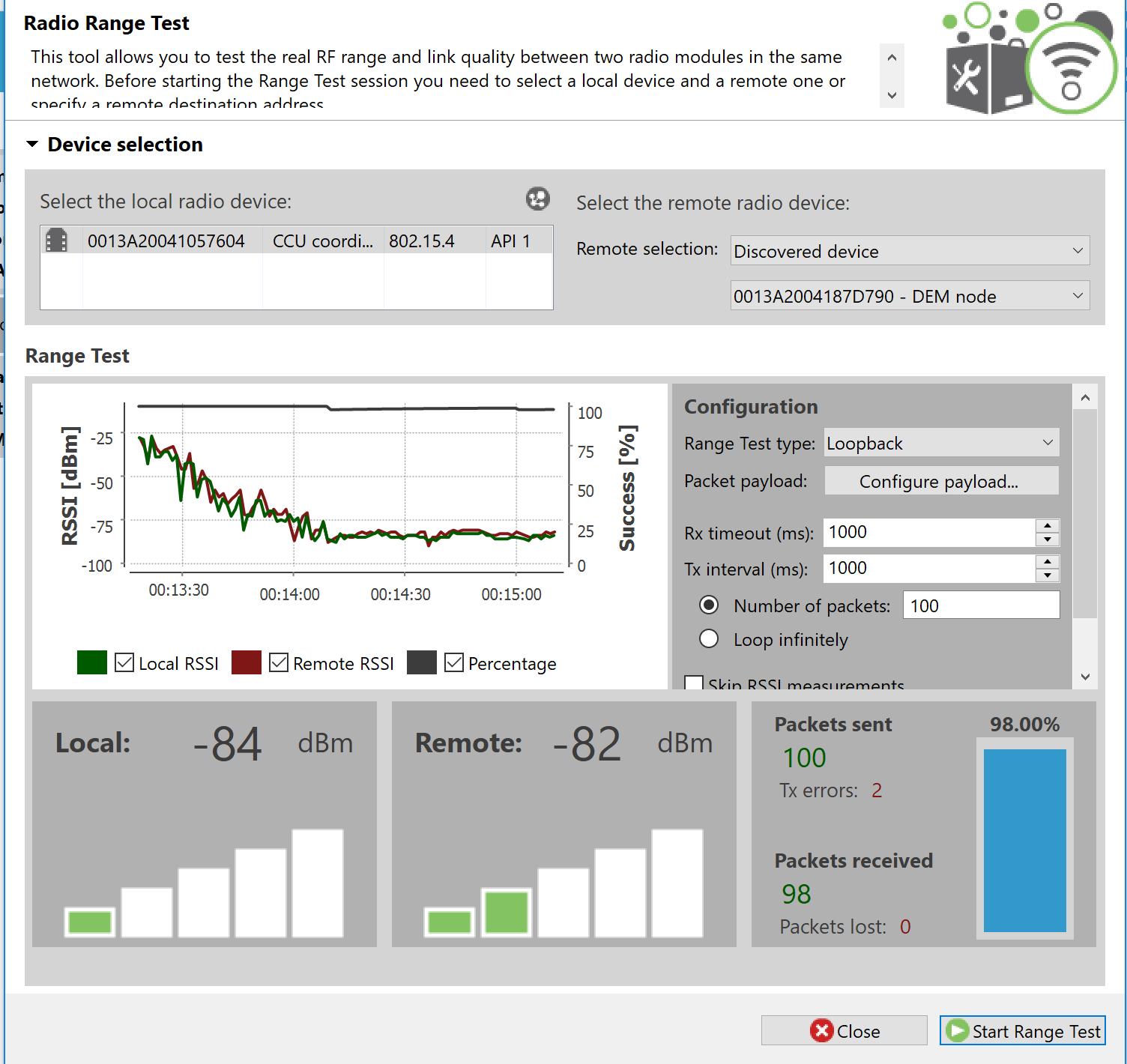


**Residential, 2-bedroom basement apartment in Harbor Landing:**

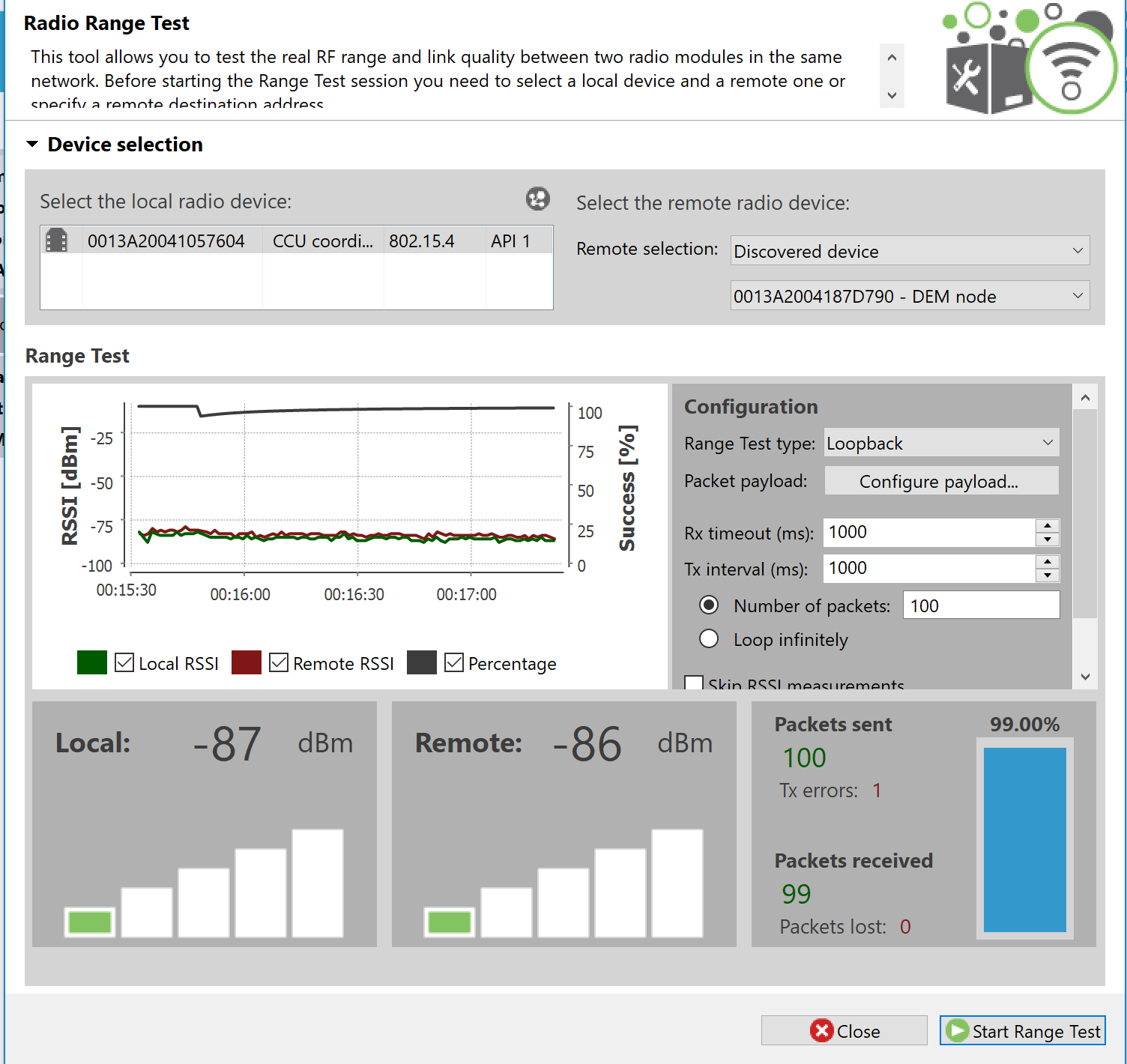
1. 12.50m:

****

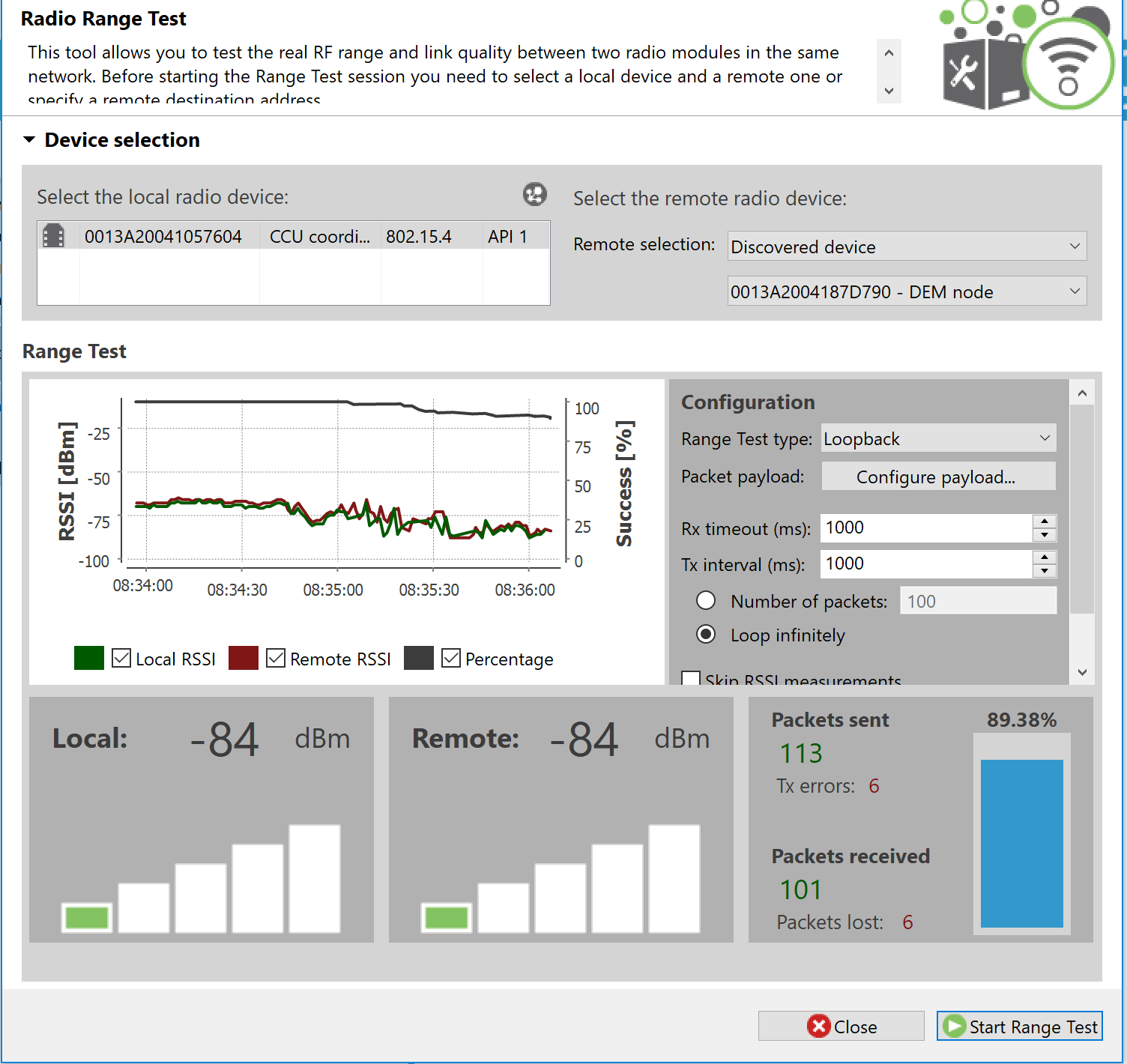
1. 12.50m

****

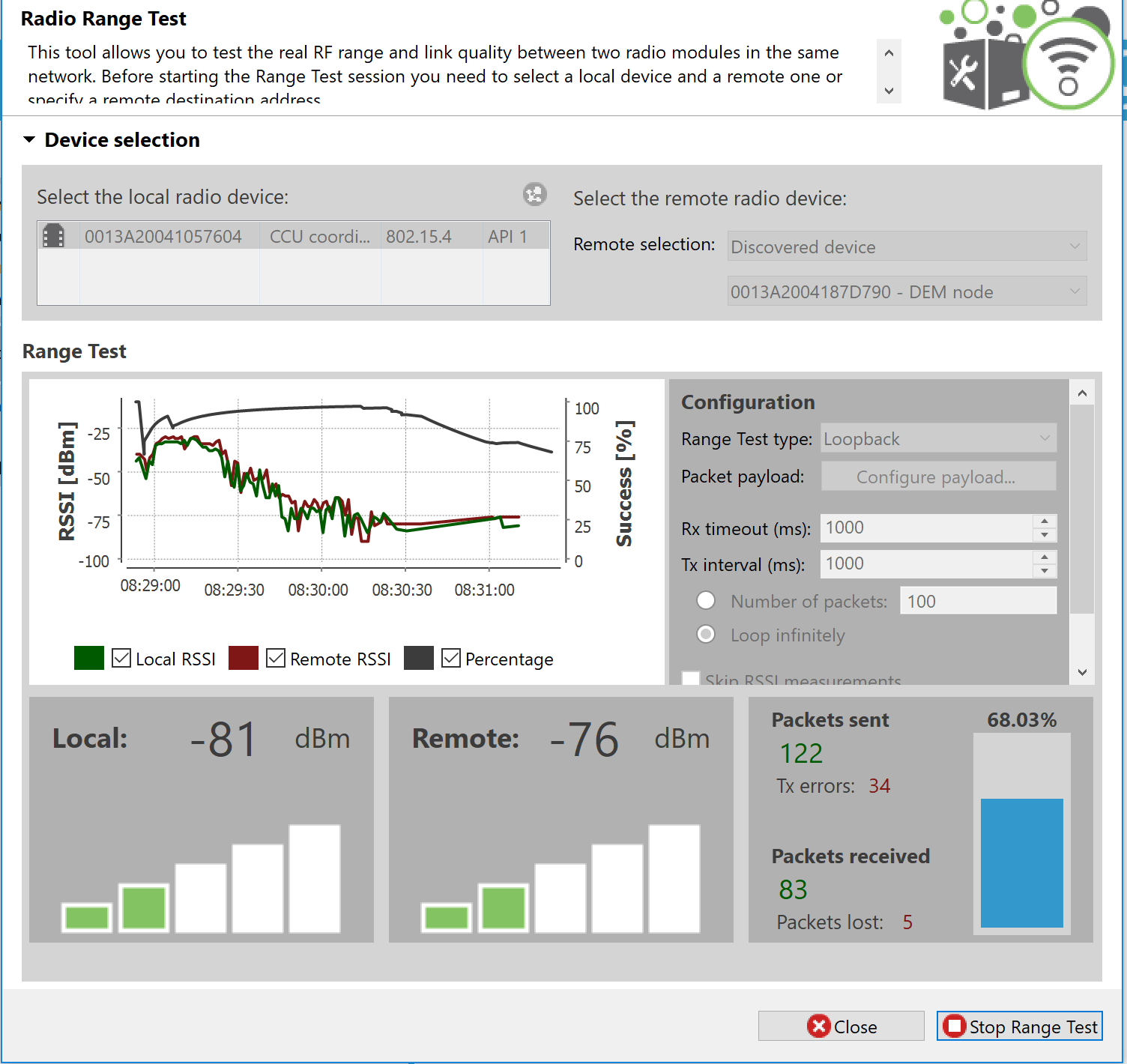
1. 20.21m:

****

1. 20.21m

****

1. 33.50m:

****