UG497: BT122 User's Guide

This document describes BT122 SDK structure and installation procedure, using developer software tools (BGTool and BGBuild), and programming the development board. It also contains a short development board introduction with examples included in the SDK package.

KEY FEATURES

- SDK introduction
- BGTool User Guide
- Attached SDK example guides

Table of Contents

1 Version History

Table 1.1. Version History with Comments

Version	Comment
1.0	Initial version
1.1	Updated UART/SPI interface information Updated BGTool figures
1.2	Modified as per the changes in the new design: • USB Micro B connector added

2 SDK

The Software Development Kit (SDK) is a set of tools used to develop applications for a specific hardware. Using the delivered files and instructions, developers can configure devices according to their specific requirements. It provides different ways to control the modules, for example with BGAPI commands (custom communications protocol, which can be used to externally control modules through UART) or BGScript (an event-based language, in which a set of specified BGAPI commands is being executed upon an event occurrence).

2.1 Installation

Download the installation package (SiliconLabs_BT122-X.X.X-X.exe) from the official Silicon Labs Web page and run the executable. Follow the on-screen instructions.

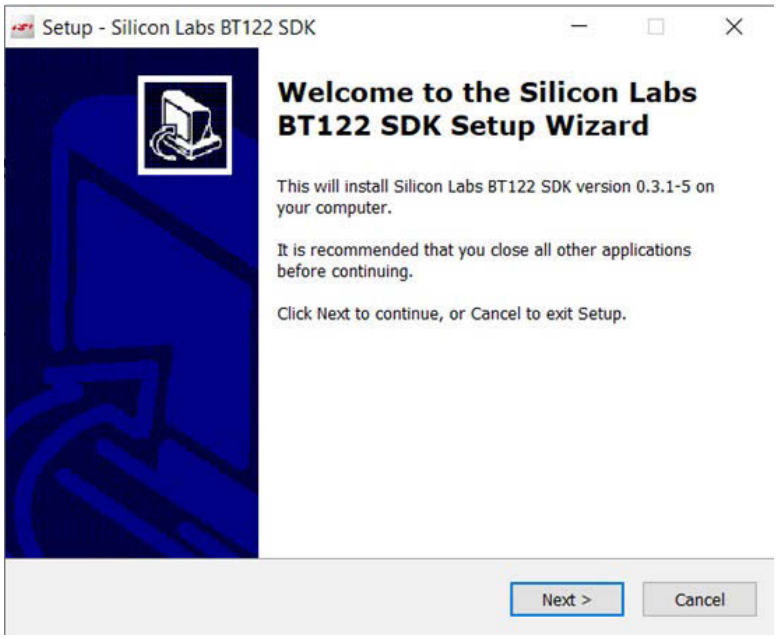

Figure 2.1 SDK Installer

2.2 Folder structure

After the SDK installation is complete, you will see the following folders in the installation directory:

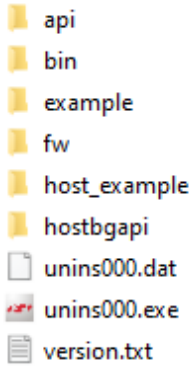

Figure 2.2 SDK Folders

The folders contain the following:

- **API** – Includes a raw XML description of the Bluetooth stack's API. It can be used, for example, to automatically generate parsers for the API.
- **BIN** – Includes SDK executable sources and programs, such as BGTool, BIMBER, or BGBuild.
- **EXAMPLE** – Includes BT122 example projects.
- **FW** – Contains all binaries required by the firmware and FWs for on-board J-Link/VCOM interface.
- **HOST_EXAMPLE** – Contains example code for the host, which uses BGAPI serial protocol with BGLIB library.
- **HOSTBGAPI** – Includes BGAPI commands documentation and BGLIB header files, which need to be included in the projects implementing BGAPI library.

The Silicon Labs Bluetooth Dual Mode SDK includes the following APIs, components, and tools:

- **The Bluetooth Dual Mode stack**.
- **BGAPI** is a binary serial protocol API to the Silicon Labs Bluetooth Dual Mode stack over UART interface. BGAPI is targeted for users, who want to use both Bluetooth BR/EDR and LE functionality and use all features in the Bluetooth Dual Mode stack form an external host such as a low power MCU.
- **BGLIB** is a host library for external MCUs and implements a reference parser for the BGAPI serial protocol. BGLIB is delivered in C source code as part of the Bluetooth Dual Mode SDK and can be easily ported to various processor architectures.
- **BGScript** interpreter and scripting language allow applications to be developed for the Bluetooth Dual Mode module's built-in MCU. It allows simple end user applications or enhanced functionality to be programmed directly in the Bluetooth Dual Mode module, which means no external host MCU is needed. In addition, the BGAPI protocol can be used together with the implemented BGScript applications, allowing external host to take over parts of the execution logic.
- **Profile Toolkit** is a simple XML-based description language, which can be used to easily and quickly develop GATT-based service and characteristic databases for the Bluetooth Dual Mode module.
- **BGBuild compiler** is a free-of-charge compiler that compiles the Bluetooth Dual Mode Stack, the BGScript application and the Bluetooth GATT services to the firmware binary that can be installed to the Bluetooth Dual Mode modules.
- **BGTool** is a graphical user interface application and a developer tool, which allows the Bluetooth Dual Mode module to be controlled over the host interface using the BGAPI serial protocol. BGTool is a useful tool for testing the Bluetooth Dual Mode module and evaluating its functionality and APIs.
- **DFU Tools** are also included as part of the SDK allowing the firmware to be updated over the UART interface.
- **BIMBER** is a simple migrator tool created because of the end of iWRAP modules firmware support. Migrator allows to convert dump of Persistent Storage from iWRAP modules into BT122 project files (including BGScript).

3 Kit Hardware

3.1 Board Features

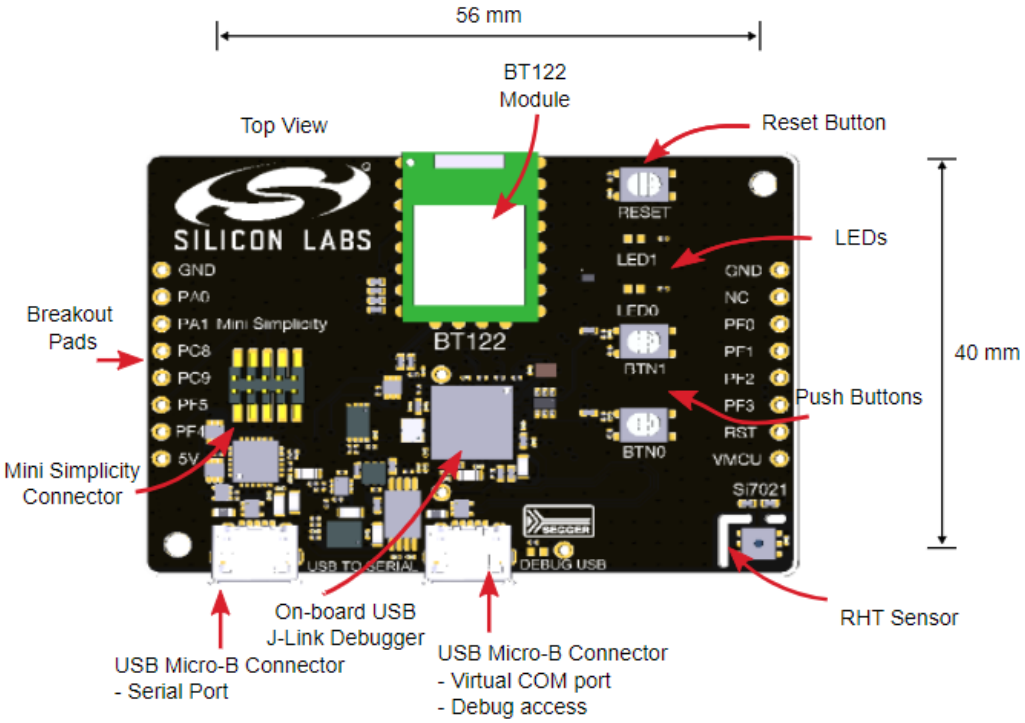

Figure 3.1 BRD4315B Development Board with BT122 Module

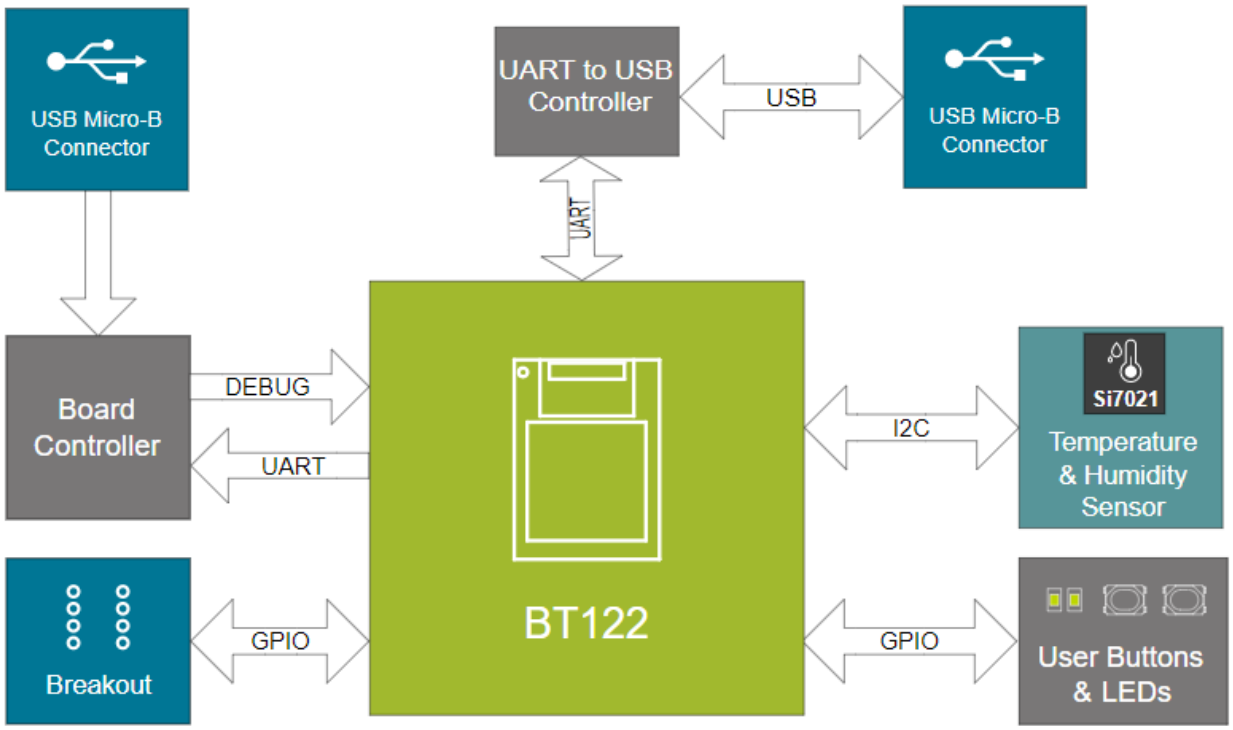

Figure 3.2 BRD4315B Block Diagram

3.2 On-Board J-Link Debugger

The kit contains a microcontroller separate from the BT122 that provides an on-board J-Link debugger through the USB micro-B port. This microcontroller is referred to as the "On-board Debugger" and is not programmable by the user. When the USB cable is removed, the on-board debugger goes into a low-power shutoff mode (EM4S), consuming typically around 20 nA.

The on-board debugger also provides a Virtual COM port (VCOM), which provides a way to transfer application data between the host PC and the target processor over UART.

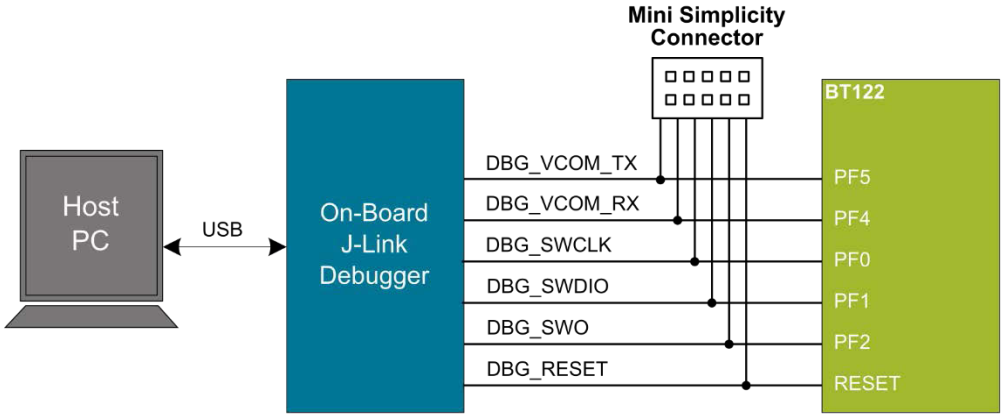

Figure 3.1 On-Board Debugger Connections

3.3 Power Supply

The kit can either be powered by the on-board LDO when the USB cable is plugged in or through the Mini Simplicity connector, which also enables using the Advanced Energy Monitor (AEM) available on select hosts. Power through USB can be either from Debug USB port or USB to Serial Port. The power multiplexer on-board switches automatically between the power from either USB based on the power level.

Note: The kit should not receive power through both the USB cable and the Mini Simplicity connector simultaneously. To power the kit through Mini Simplicity connector, remove the kit USB cable. If you want to power the kit over USB and use the on-board debugger, the Mini Simplicity connector should not be connected to an external debug/power source.

Note: When both the power inputs to the Power Mux are equal, input from Debug USB takes priority.

3.3.1 Energy Consumption Measurement

The total energy consumed by the BT122 and its peripherals (including the LEDs, buttons and the Si7021 relative humidity and temperature sensor) can be measured through the Mini Simplicity connector using a Silicon Labs kit with Advanced Energy Monitor (AEM), such as the Wireless Starter Kit (WSTK, recommended), or by connecting an external power supply to the VMCU pin on the kit breakout pads. The USB cable should be disconnected for energy measurement purposes, which will put the on-board debugger in a powered-down state where it does not contribute to the energy consumption.

3.4 Connectors

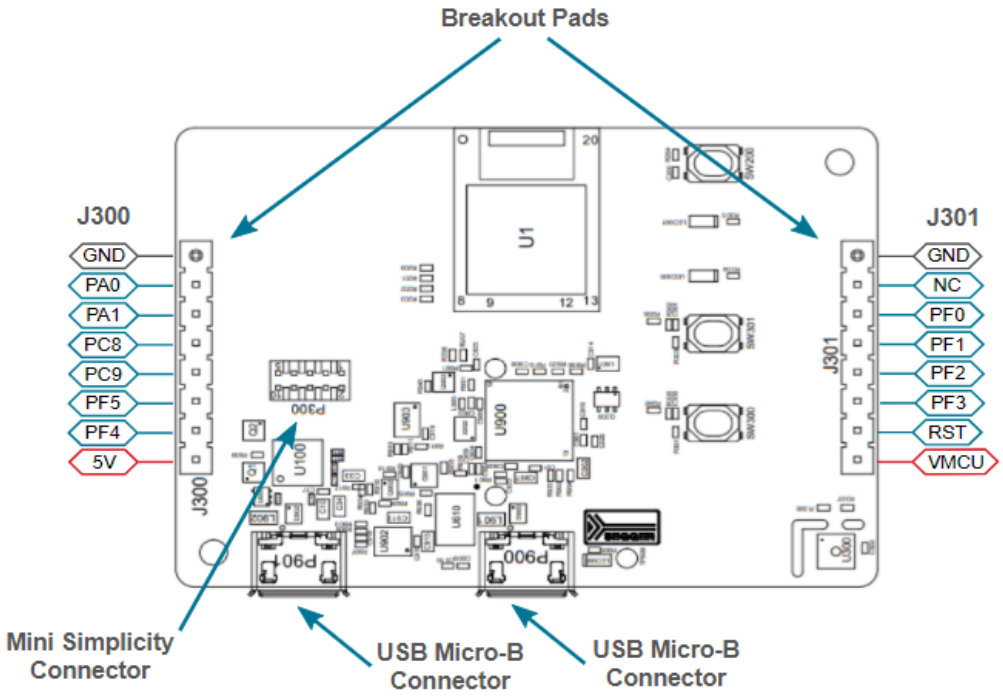

Figure 3.2 BRD4315B Connectors

3.4.1 Breakout Pads

Table 3.1 I internal Development Board Signals on Breakout Pads

Peripheral Function	GPIO Name									
	PA0	PA1	PC8	PC9	PF0	PF1	PF2	PF3	PF4	PF5
Pin Number	4	5	6	7	15	14	12	11	10	9
5V Tolerant	N	N	Y	Y	Y	Y	Y	Y	Y	Y
UART			CTS	RTS					RX	TX
SPI			CLK	CS					MISO	MOSI
I²C	SCL	SDA								
ADC Input	ADC_CH0	ADC_CH1						ADC_CH2		
Programing Interface					SWCLK/ TCK	SWDIO/ TMS	SWO/ TDO	TDI		

3.4.2 Mini Simplicity Connector

The Mini Simplicity connector is a standardized interface used by Silicon Labs debug tools, exposing the SWD and Virtual COM (VCOM) port on a 2x5 pin 1.27 mm pitch connector. See section 4.2 for details about connecting the kit to a Silicon Labs Wireless Starter Kit (WSTK) through the Mini Simplicity connector.

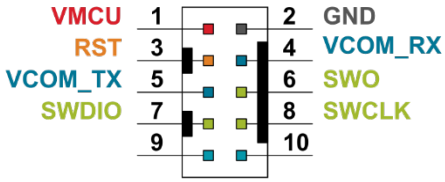

Figure 3.3 Mini Simplicity Pin Connections

Table 3.2 Mini Simplicity Pin Description

Pin Number	Connection	Function	Description
1	VMCU	VAEM	Target voltage on the debugged application. Supplied and monitored by the AEM when powered by a WSTK with its power selection switch in the "AEM" position.
2	GND	GND	Ground.
3	BT122 Reset pin	DBG_RST	Reset.
4	PF4	VCOM_RX	Virtual COM RX.
5	PF5	VCOM_TX	Virtual COM TX.
6	PF2	DBG_SWO	Serial Wire Output.
7	PF1	DBG_SWDIO	Serial Wire Data.
8	PF0	DBG_SWCLK	Serial Wire Clock.
9	NC		No Connection
10	NC		No Connection

3.4.3 USB Micro-B Connector

There are 2 USB Micro-B connectors on the board:

- USB to Serial
- Debug USB

Board can be powered directly from either of these two USB ports. The Debug USB port can be used for uploading code, debugging, and as Virtual COM port. USB to Serial port acts as COM port and communicates with the BT122 chip.

3.5 Peripherals

3.5.1 User Push Buttons and LEDs

The kit has two user push buttons, both located on the right side of the board. The kit also has two yellow LEDs, connected to the same GPIO pins as the push buttons via a driver transistor. The push buttons/LEDs are connected to pins PF3 and PF2 respectively. Both the push buttons and the LEDs are active low, and the push buttons are de-bounced by an RC filter with a time constant of 1 ms.

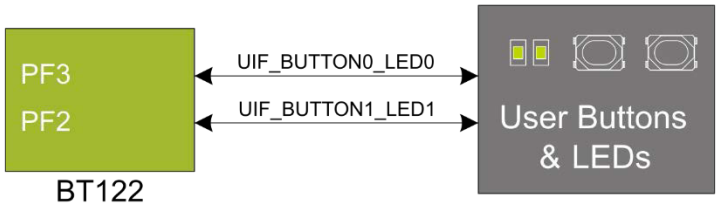

Figure 3.4 User Push Buttons and LEDs Pin Connections

3.5.2 Si7021 Relative Humidity and Temperature Sensor

The Si7021 I2C relative humidity and temperature sensor is a monolithic CMOS IC integrating humidity and temperature sensor elements, an analog-to-digital converter, signal processing, calibration data, and an I^2C Interface. The patented use of industry standard, low-K polymeric dielectrics for sensing humidity enables the construction of low-power, monolithic CMOS Sensor ICs with low drift and hysteresis, and excellent long term stability.

The Si7021 offers an accurate, low-power, factory-calibrated digital solution ideal for measuring humidity, dew-point, and temperature, in applications ranging from HVAC/R and asset tracking to industrial and consumer platforms.

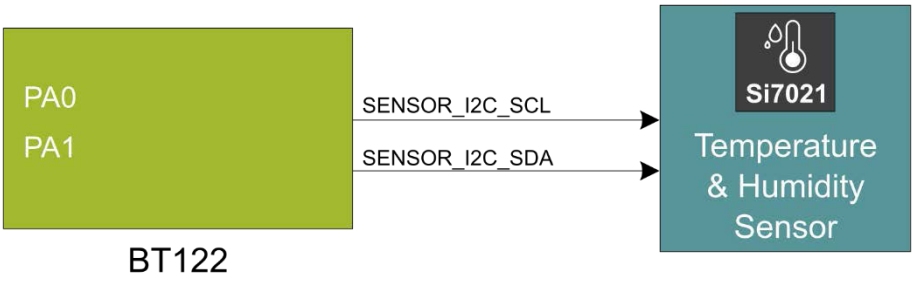

Figure 3.5 Si7021 Relative Humidity and Temperature Sensor Pin Connections

Note: Due to self-heating from the on-board LDO, temperature measurements are slightly off when running off USB power. More accurate temperature measurements are achieved when powering the board through the Mini Simplicity connector.

4 Preparing the Board

4.1 Programming via J-Link

After the USB connection is established between the PC and the Development Board, the computer should detect the new port as a **J-Link CDC UART Port**. The BGTool can then be used to upload new firmware, as explained in chapter 5.

The main method for uploading the software to the BT122 module is through the J-Link (SWD) interface. This process is effective and most comprehensive. It allows complete module erase and upload of the bootloader and firmware to the flash memory. It also allows one to repair a crashed bootloader (for example, when user would flash, by mistake, firmware-only variant of binary file to the bootloader memory range of MCU flash). Proper connection of the J-Link (SWD) is shown in Figure 4.1.

Development board for BT122 – BRD4315B – has the J-Link (SWD) interface available and already connected as the on-board programmer for BT122.

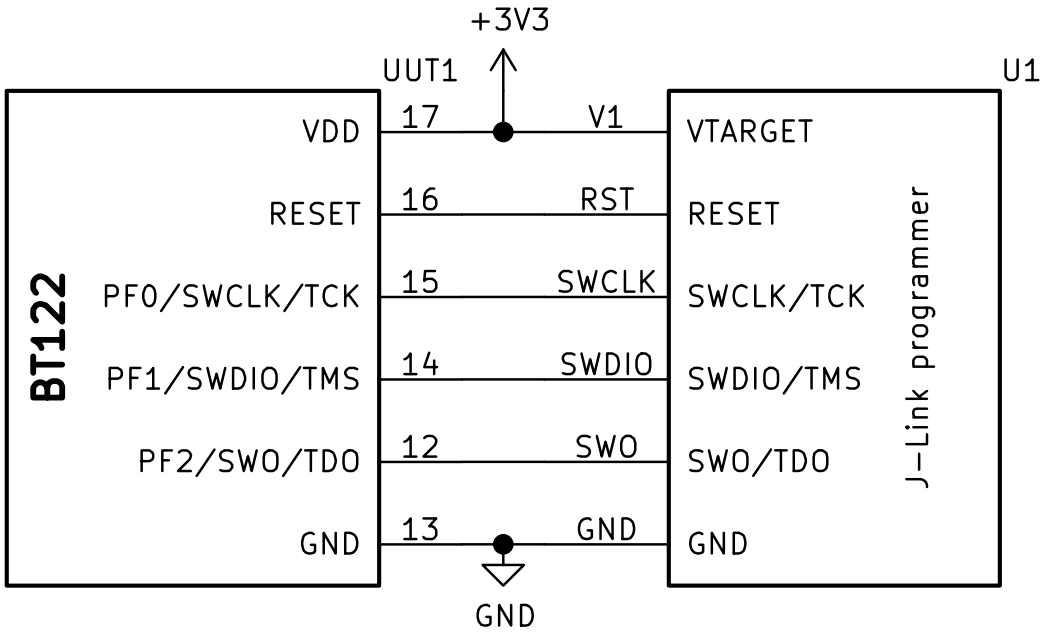

Figure 4.1 BT122 Firmware Update via J-Link Debugger and SWD Interface

4.2 Programming via Mini Simplicity Connector

Additionally, the mini-simplicity connector is also available on the Development board.

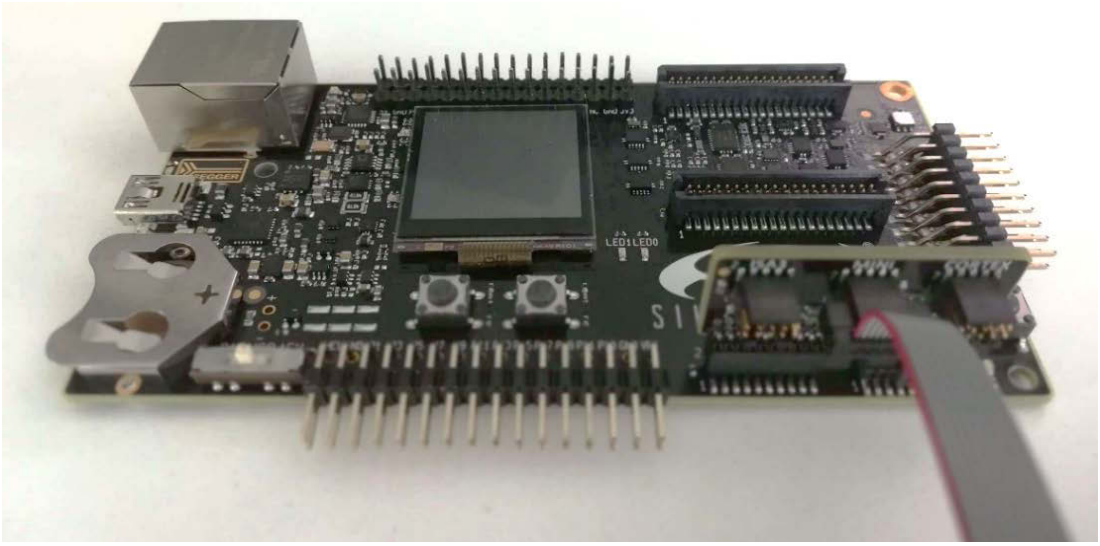

Figure 4.2 WSTK Board with STK/WSTK Debug Adapter Connected

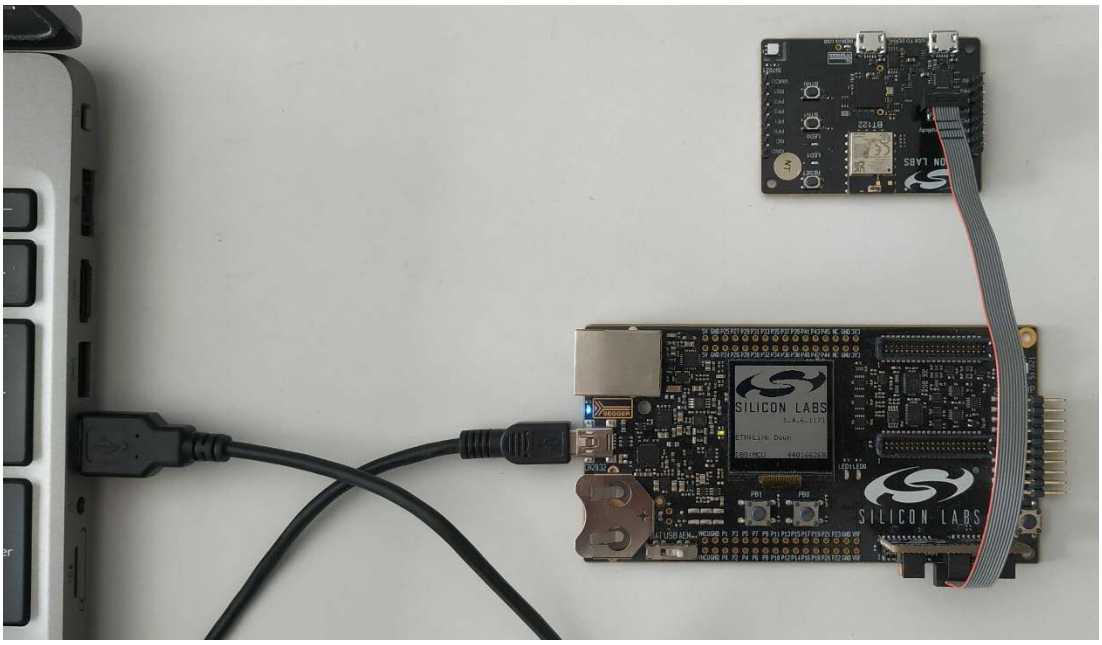

Figure 4.3 Finished Setup

First, the Silicon Labs WSTK board should be prepared to be used as a standalone J-Link debugger. The power switch should be set to USB. Plug the STK/WSTK Debug Adapter (SLSDA001A) into the debug connectors (two connectors near the RESET button). Connect the BT122 Development Board and STK/WSTK with J-Link tape using Mini Simplicity Connector (on BT122 side) and the Debug Adapter (on WSTK side). Both boards should be connected by USB to the power supply. The whole setup should look like the image in Figure 4.3.

4.3 Programming via UART DFU

The Device Firmware Update (DFU) programming mode is meant to be used when J-Link programming is not available. When the BT122 module/device is booted up in DFU mode, upload the new program through UART. In such cases, it is a Bootloader's responsibility to handle writing the application into the flash memory. Once the upload is completed, the bootloader will start program execution.

In contrast to the J-Link method, the user needs to ensure that the proper bootloader is flashed into the BT122 module before starting the update procedure.

To upload through the UART DFU, while the device is configured to use BGScript and is not responding to the BGAPI commands, it is necessary to implement a specific sequence in the application, which would allow the module to be booted into **DFU mode** (for example, in some GPIO interrupt handler).

To enter **DFU mode** in BGScript, use the `dfu_reset(1)` command.

Remember that during the DFU procedure, UART baud rates of the device and the PC must match. Flow control must be used as well.

The Development board for BT122, BRD4315B, has the UART/USB bridge with interface called VCOM, which is available on-board and already connected to the BT122 module.

On-board UART/USB bridge firmware can be updated to support baud rate settings up to 3 Mbps. This can be done using Commander program (available under the SDK package directory …\BT122-x.x.x\bin\commander) in two different ways:

- Using command line and typing "commander adapter fwupgrade -s <kitserial> <path_to_emz_file>"
- Using GUI (run commader.exe)
  - Go to the Kit-tab in Commander.
  - Select browse in the "Installation package"-pane.
  - Navigate and select the emz-file (.emz file with improved FW is available under SDK package directory …\BT122-x.x.x\fw\).
  - Select "Install package"

If you're using a previous version of BT122 – BRD4315A for special applications that require the highest performance of UART/USB connection, you can choose to connect the external UART/USB bridge to use with BRD4315A instead of the built-in VCOM interface. For example, CP2102 UART/USB bridge. To do so, follow the steps below:

1. Desolder 0 ohm resistors on RX, TX, CTS, and RTS lines that connect BT122 module with on-board debugger - R202, R203, R207, R208.
2. Connect your external bridge lines to corresponding lines available on the left breakout pad.
3. Connect the power supply to your external bridge.

In the case of BRD4315B, the CP2102 UART/USB bridge is available on the board and connected to the second USB Micro-B connector.

5 BGTool

The BGTool lets users communicate with the module using BGAPI commands or a more user-friendly graphical interface and to build projects using the BGBuild compiler, and then upload them to the module via J-Link. The program can be found in the SDK directory after installation (latest SDK package can be found on the module's Website). To open the BGTool, go to the bin folder and run bgtool.exe. BGTool uses the Simplicity Commander application and BGAPI-based DFU application to execute the update process. Both are included separately in the SDK package for more advanced users, who may not need to use the BGTool.

115200 baud rate and RTS/CTS flow control enabled are the default production settings for VCOM configuration. If an application that has been prepared/selected to be uploaded uses non default UART/USB bridge settings and built-in VCOM, configure the on-board UART/USB bridge settings according to your application settings.

5.1 VCOM configuration

115200 baud rate and RTS/CTS flow control enabled are the default VCOM configuration production settings. To make sure that connection to the board in the interactive view and usage of all flashing options are possible when VCOM settings are other than factory, configure VCOM options under the "Development board VCOM" tab (Figure 5.1).

Note: If you are using an external bridge, omit the part about VCOM configuration.

After correctly selecting the Serial Number, baud rate, and RTS/CTS option, press the Set button. If any system prompt arises, such as telnet connection request, accept them. Telnet connection is done locally, is safe, and is a part of the process.

Note: Once set, the VCOM settings are permanently stored in memory until they are modified again.

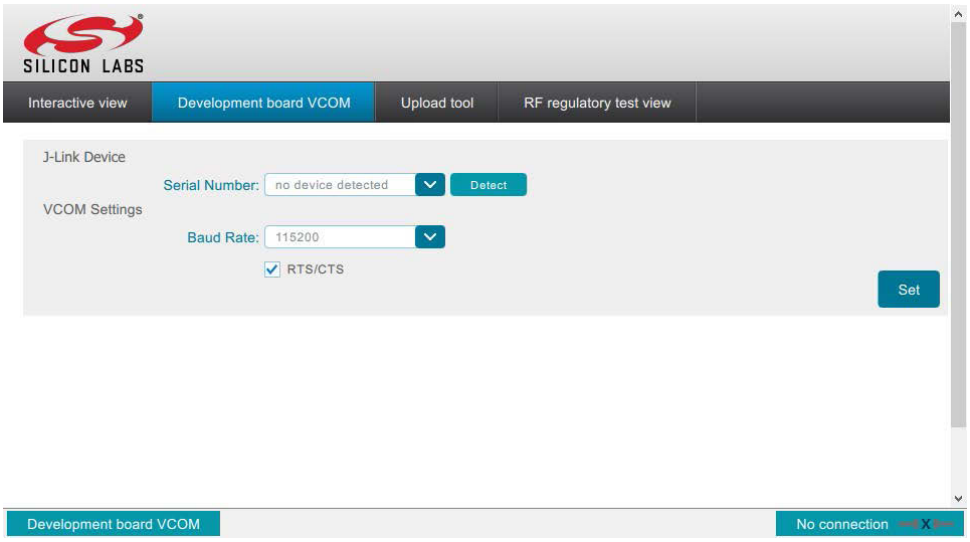

Figure 5.1 BGTool in Development Board VCOM Mode

The firmware does not support configuration over USB-CDC. Nonetheless, set the baud rate in the PC-side driver.

Additionally, you can change the VCOM bridge baud rate through CLI as an alternative to the previously mentioned BGTool GUI method:

1. Silink can be found under the SDK package directory …\BT122-x.x.x\bin\silink.
2. Start Silink from the CLI with the following command: "`silink -sn <kitserial> -automap 49000`".
3. Choose telnet localhost 49002.
4. Configure baudrate with "`serial vcom config speed <baudrate> handshake rtscts`".

5.2 Programming the Device

The **Upload tool** tab is at the top of the BGTool window (Figure 5.2).

First, select a **Project File**, which is an *.xml file that contains the project configuration of features, such as hardware peripherals configuration, BGScript file path (if used), or GATT services and characteristics. Everything is already configured in the **examples**' directory. For more information, see the BT122 Project Configuration Guide. Browse for the desired project configuration file and select it.

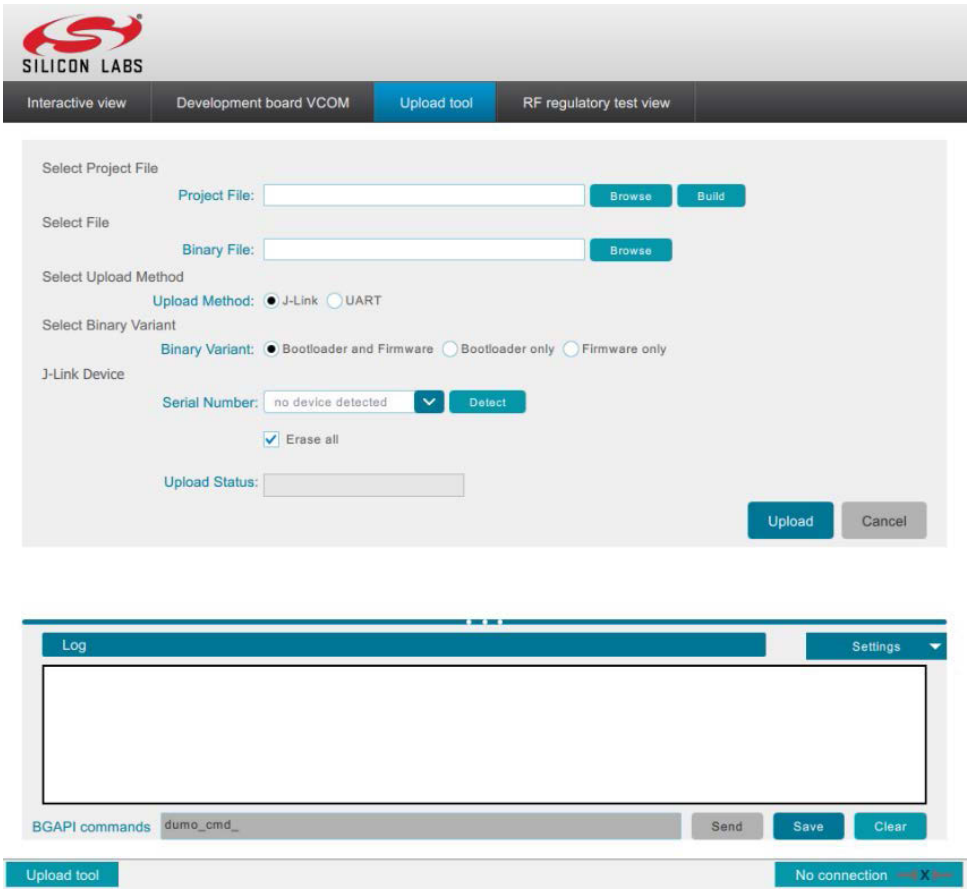

Figure 5.2 Upload Tool

To build the project, call the BGBuild tool by pressing the Build button. BGBuild checks all configuration files, compiles them, and generates firmware binaries.

After step three, .bin files should be created and placed in the same folder as the project.xml file:

- Firmware only variant consists only of the firmware part of the build and is suitable if you want to keep bootloader intact and update only the firmware part of the application.
- Bootloader only variant consists only of the bootloader part of the build.
- Combined firmware and bootloader variant is a classic approach, suitable for updating the whole module (for example after a full erasure).

After the compilation, a Binary File with firmware and bootloader combined will appear automatically in the Binary File field. Keep in mind that BGTool only supports .bin files.

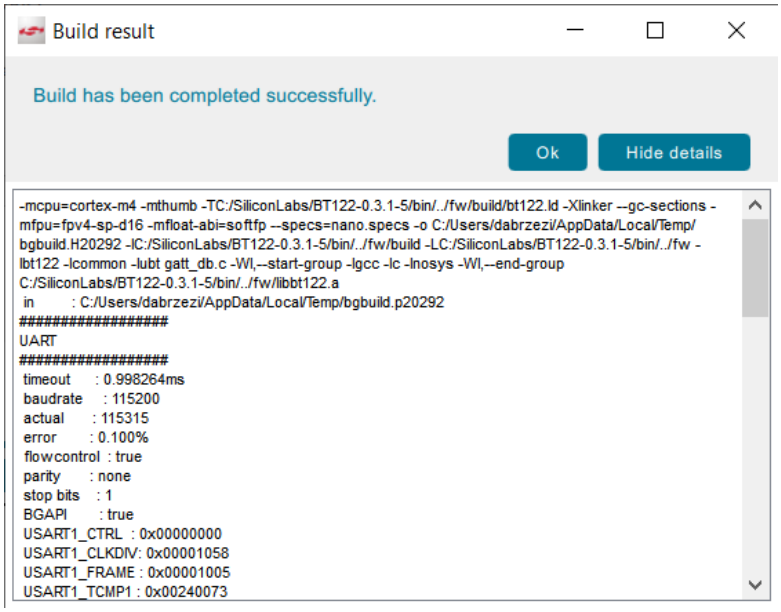

Figure 5.3 Project Building

Next, it is necessary to choose a way of Upload Method – meaning the update interface:

5.2.1 J-Link

Selected binary variant will be uploaded through J-Link, which is a faster method and can, by selecting "Erase All" option , perform a "clean" firmware installation. Make sure that you select the correct J-Link device (its serial number in case there is more than one device connected).

5.2.2 UART

UART is used when the device is in Device Firmware Update (DFU) mode. Selected binary variant will be updated through UART, using selected Port, Baud rate, and flow control. Flow control is selected with the **RTS/CTS** checkbox. Normally, the device must be set in DFU mode by BGAPI command or in the BGScript, but the BGTool will automatically send the **`dumo_cmd_dfu_reset(1)`** command, putting the device in DFU mode.

For DFU update, make sure that the proper bootloader is flashed in the BT122 module before starting the update procedure. Also, when using BGScript, it must have an option to boot into the DFU mode. Remember that during the DFU procedure, VCOM baud rate, UART baud rates of the device and the PC must match. You also must use flow control.

After selecting the interface option, specify the variant of the built binary that will be uploaded to the module. This is a very important selection because this option selects not only the correct file, but also the address range that this file will be written into. It is important to choose a variant that corresponds to the binary file content that you have selected. It is worth mentioning that it is good to keep the names of the files generated by BGBuild, because based on them, BGTool will suggest the correct options (Binary Variant and filename/path) automatically.

At the end of the process, select the Serial Number/Port and appropriate settings, such as erase all (clear whole FLASH) for J-Link and RTS/CTS, baud rate for UART. If all necessary fields are filled, click Upload. After the process is over, the "Done!" message should appear in the Upload Status field. If this does not appear, analyze the logs and check the steps again.

To connect to the board in the Interactive view and use all flashing options in the Upload tool view, make sure that all UART/USB bridge options are the same for VCOM, uploaded application, and in the Interactive view connection options or DFU upload options.

Note: Each UART/USB connection settings change done with API commands must be also done for VCOM settings in the BGTool to be able to connect to the module.

5.3 Opening a Connection

After the BT122 firmware is uploaded, BGTool can be used to communicate and test the module.

Select **Interactive View**. Connect with the **J-Link CDC UART Port** using the **Baud Rate** configured in the project.xml file. Press **Open** to initiate a connection with the module.

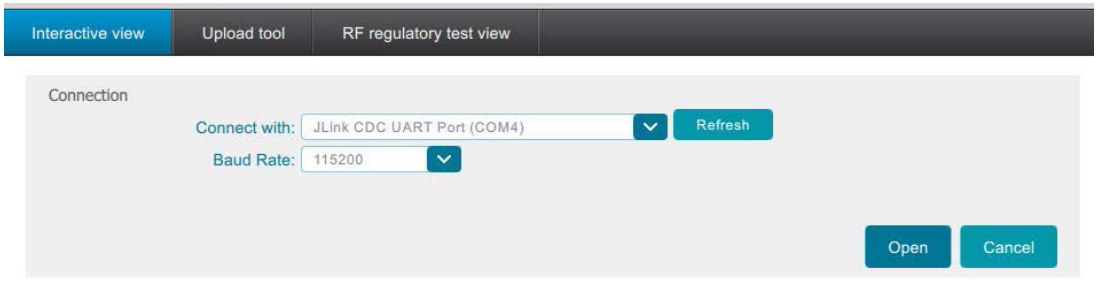

Figure 5.4 Opening a Connection

After that, a console will appear for sending commands and responses (green highlighted)/events (blue highlighted) will be received in the log at the bottom of the window. The BGTool will automatically send a **`dumo_cmd_system_get_bt_address`** command and the device should respond with a **`dumo_rsp_system_get_bt_address`** response containing the device's address. If this happens, it means that everything works correctly.

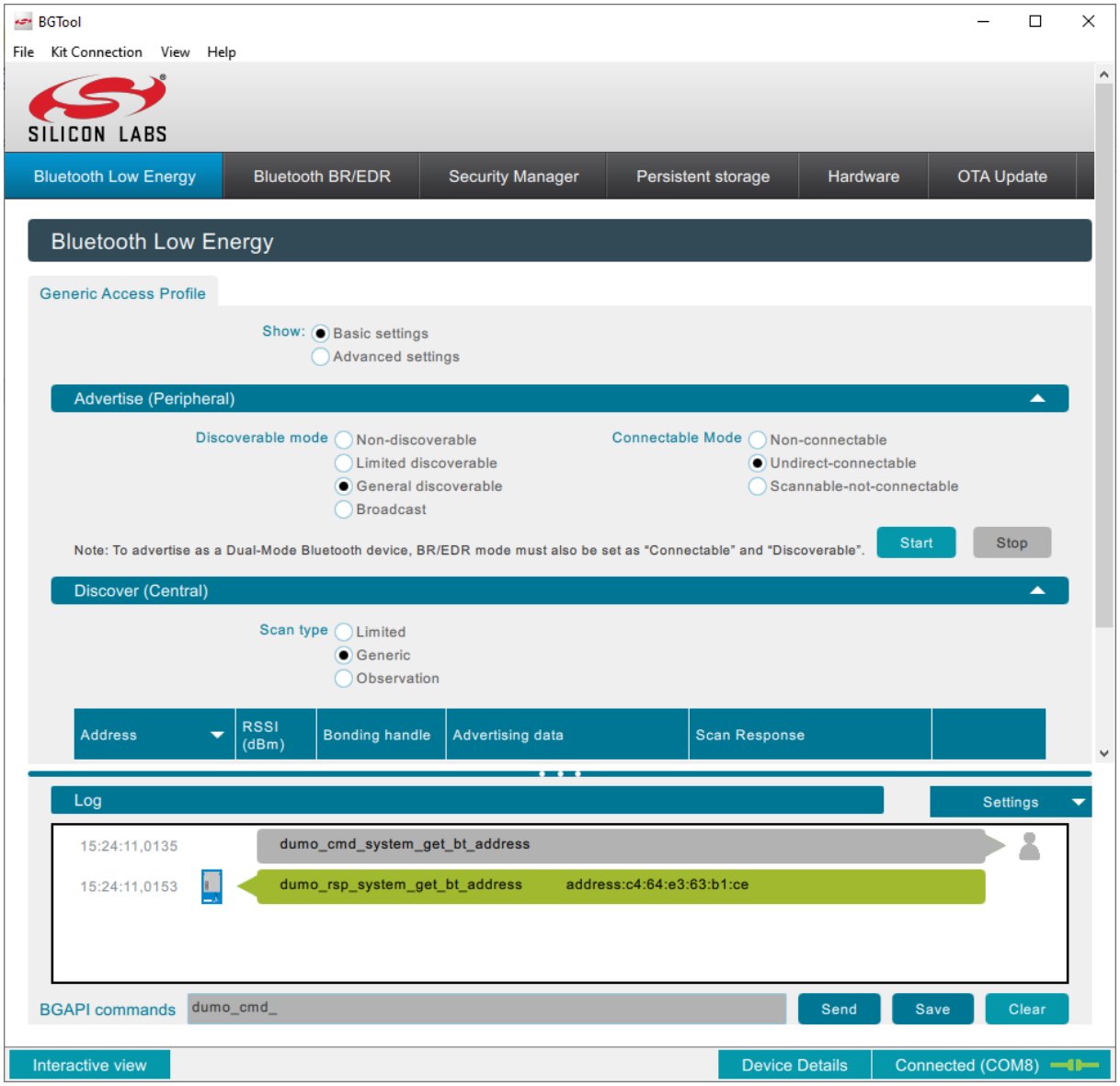

Figure 5.5 BT122 Responding with Its Address

5.4 Sending Commands Manually in Interactive View

The commands should be entered in the BGAPI commands text box on the bottom of the window, which the program will send to be executed by the device. They can be also be selected using arrows on keyboard and pressing Tab. If the command expects arguments, they must be added in parentheses in the right order. For instance, if **`dumo_cmd_system_hello()`** command is sent, the device should answer with an implemented **`dumo_rsp_system_hello`** response. If **`dumo_cmd_system_get_local_name()`** is sent, the device will answer with **`dumo_rsp_system_get_local_name`** response, which also includes local name of the device in an array of ASCII characters in hexadecimal format.

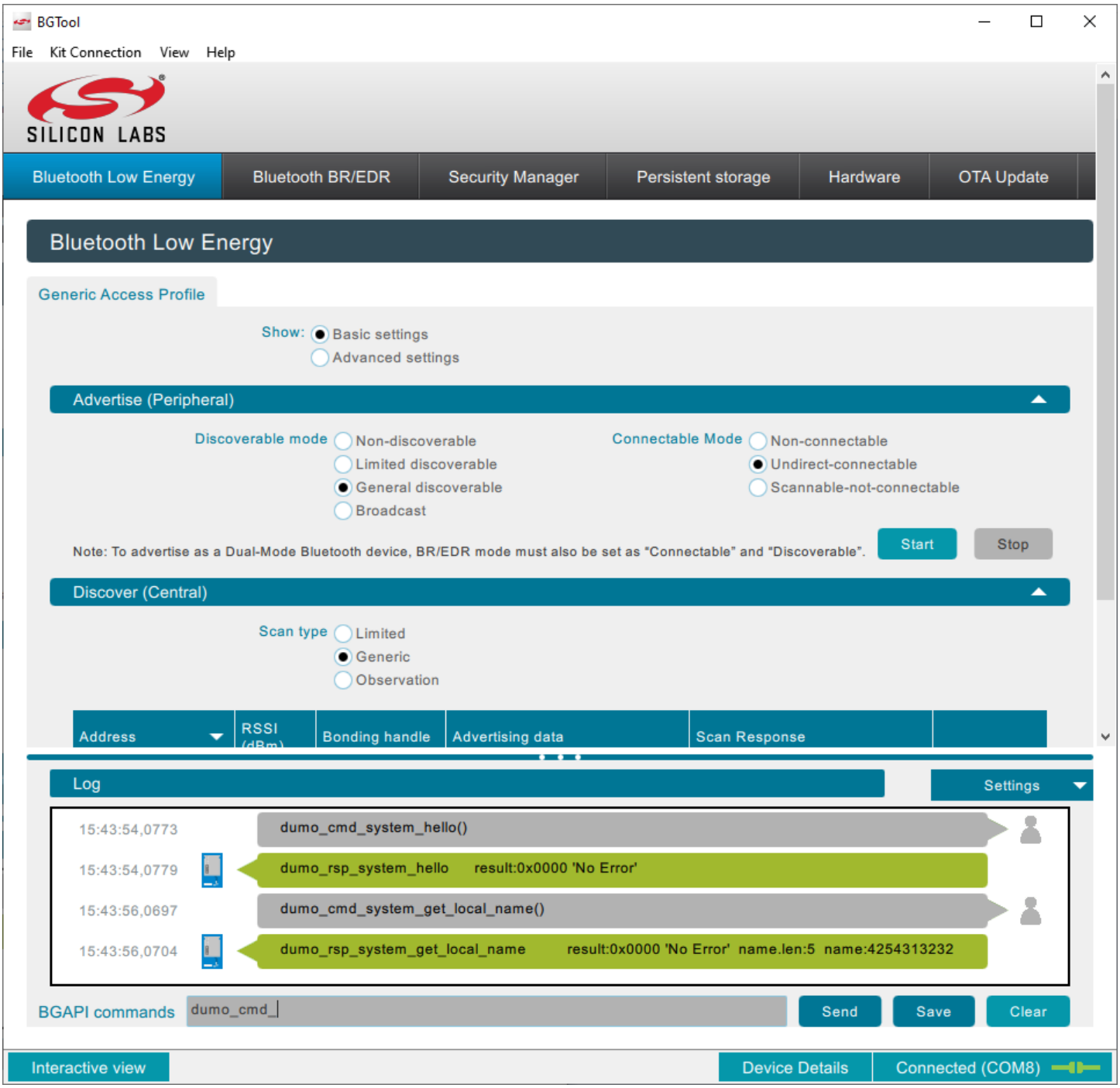

Figure 5.6 Sending Commands Manually

5.5 Interactive View Options

Interactive view lets users configure the BT122 Bluetooth settings in a more user-friendly manner. The BGTool will then send appropriate commands to the device.

The **Bluetooth Low Energy** tab will let you access the BLE settings. For example, advertising packets can be sent to nearby devices or devices can be discovered (the tab will show discovered device's address, RSSI, and so on.) by pressing the **Start** buttons under the appropriate tabs.

Note: While switching the UART baud rates via BGAPI commands, modify the related parameters on the development board VCOM bridge.

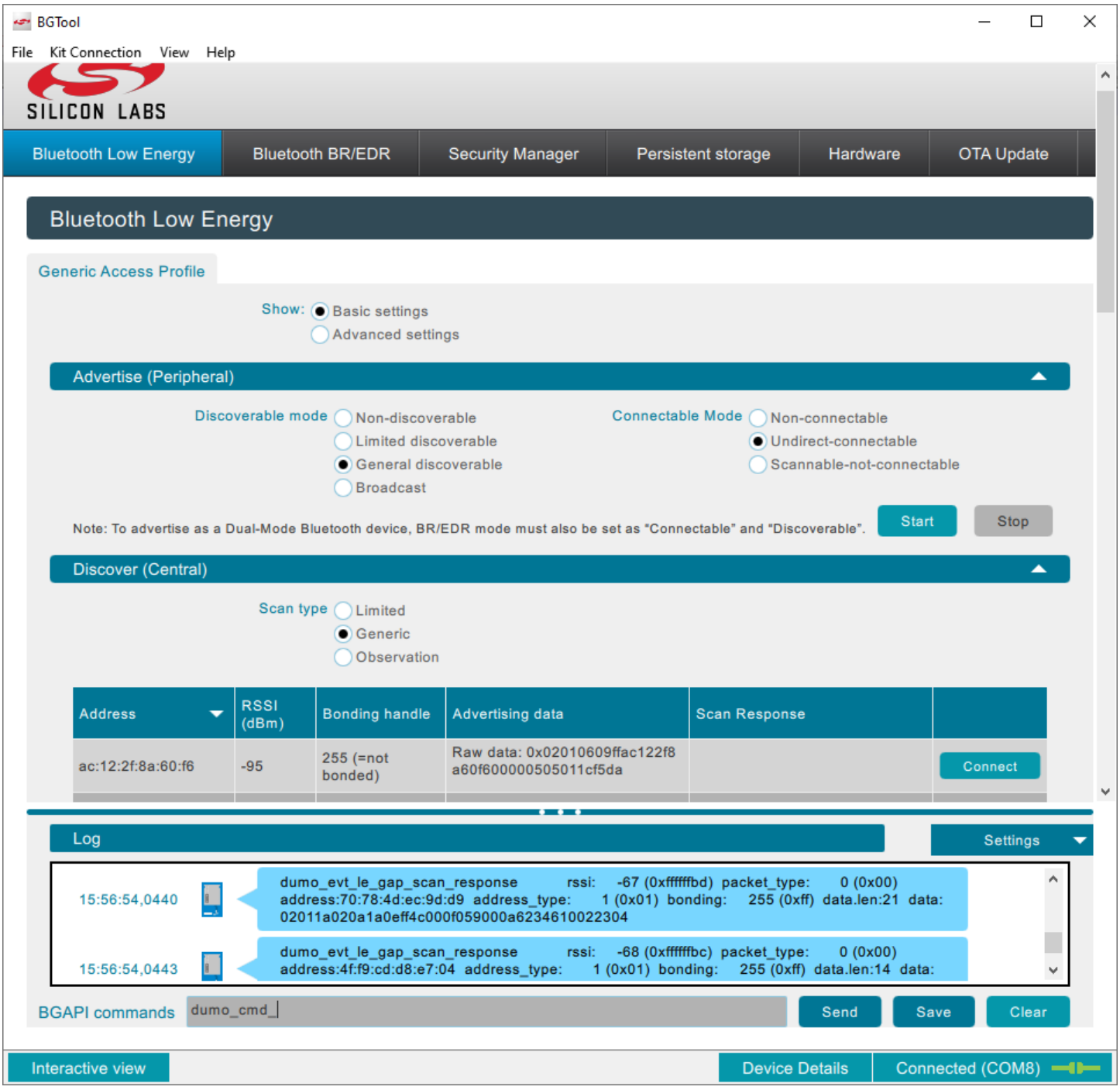

Figure 5.7 Example – Scanning for BLE Advertising Packets

The **Bluetooth BR/EDR** tab will give access to Bluetooth Classic settings. As with BLE, you can start a server, set GAP mode, discovery mode, and discover devices. For example, choosing option with RSSI and name discovery mode and then pressing start will show devices with their address, RSSI, and their name.

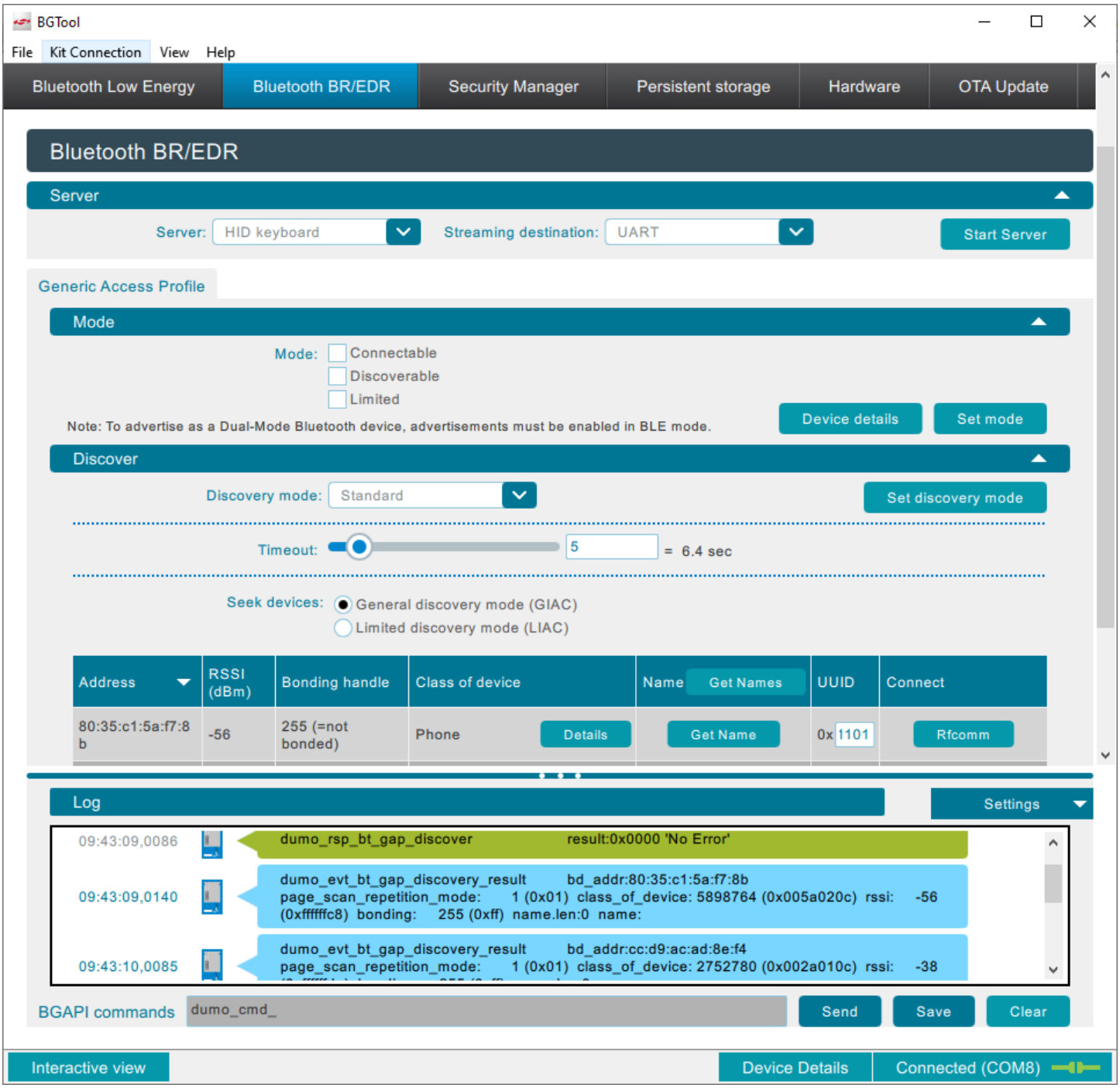

Figure 5.8 Example - Bluetooth BR/EDR Discovery

The **Security Manager** tab allows setting and checking Security details (MITM, LE Security Manager options, Bondable mode), and MAC addresses of bonded devices.

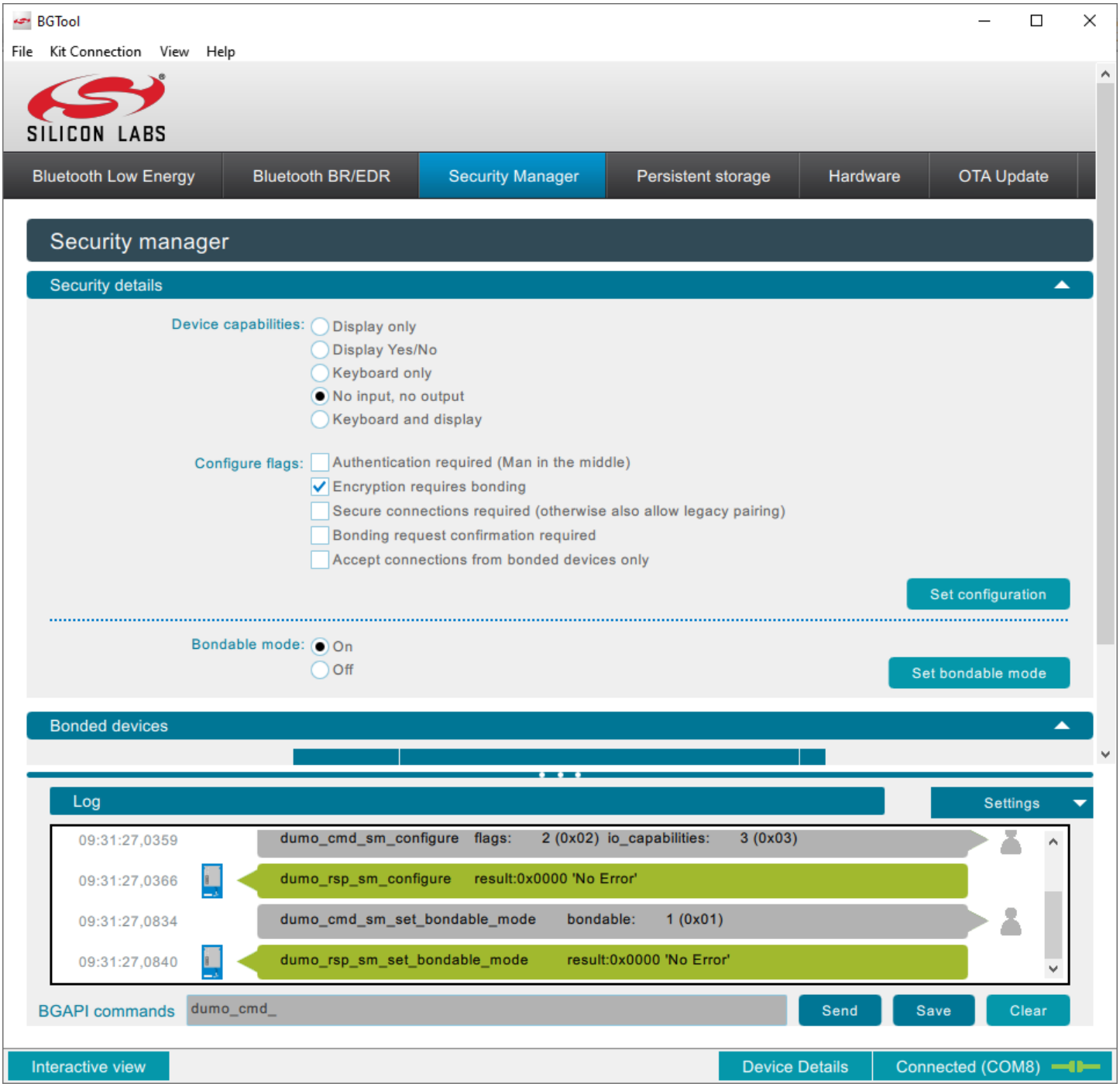

Figure 5.9 Security Manager

Persistent storage tab displays currently saved PS keys in the device.

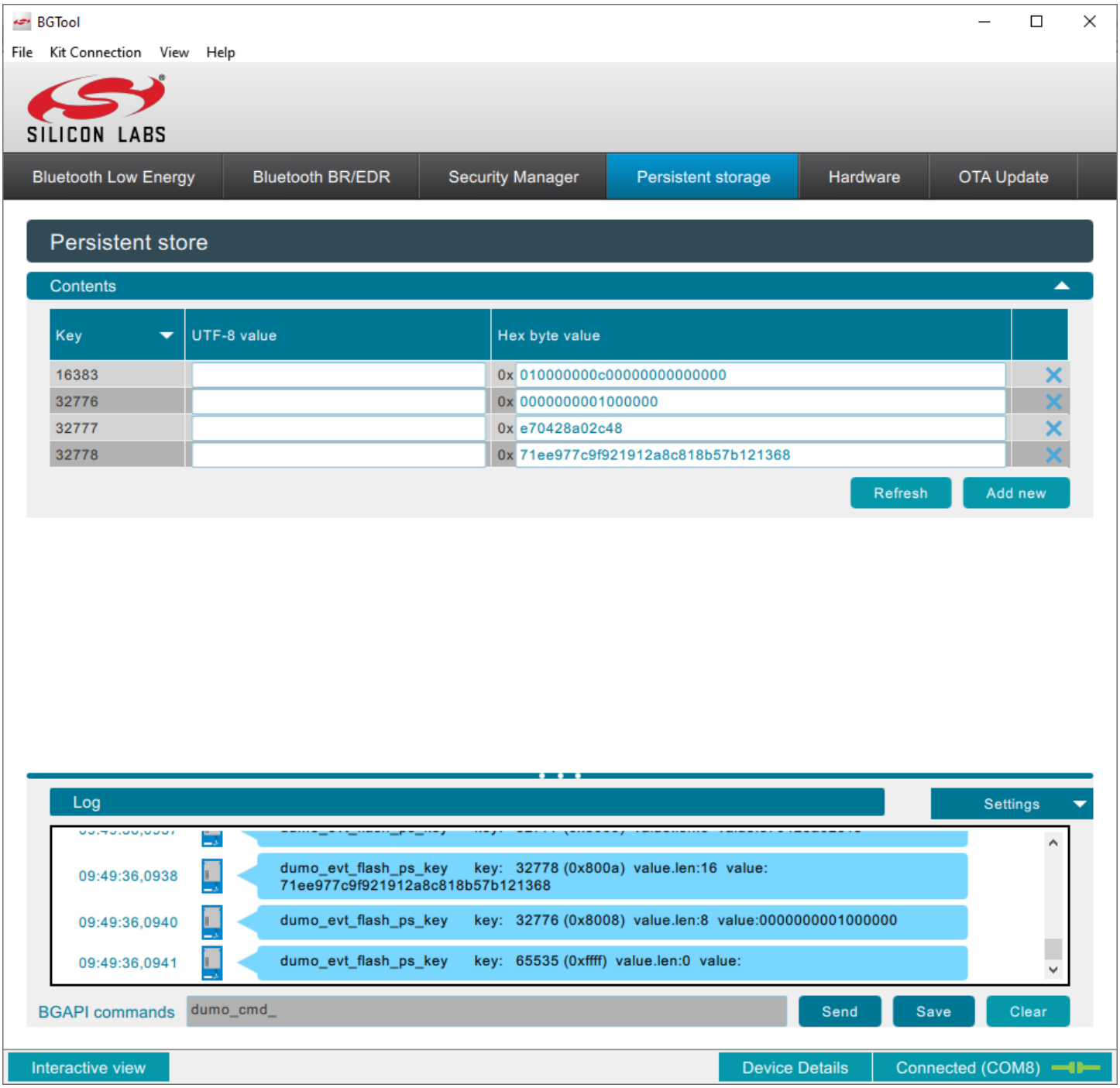

Figure 5.10 Persistent Storage Tab

5.6 RF Regulatory Test

Open the **RF regulatory test** window by pressing **View** at the top of the window and then selecting **RF regulatory test**. The loaded view allows testing BLE and BR/EDR RF capabilities of the BT122 module. **Keep in mind that sending BGAPI directly will give you more control.**

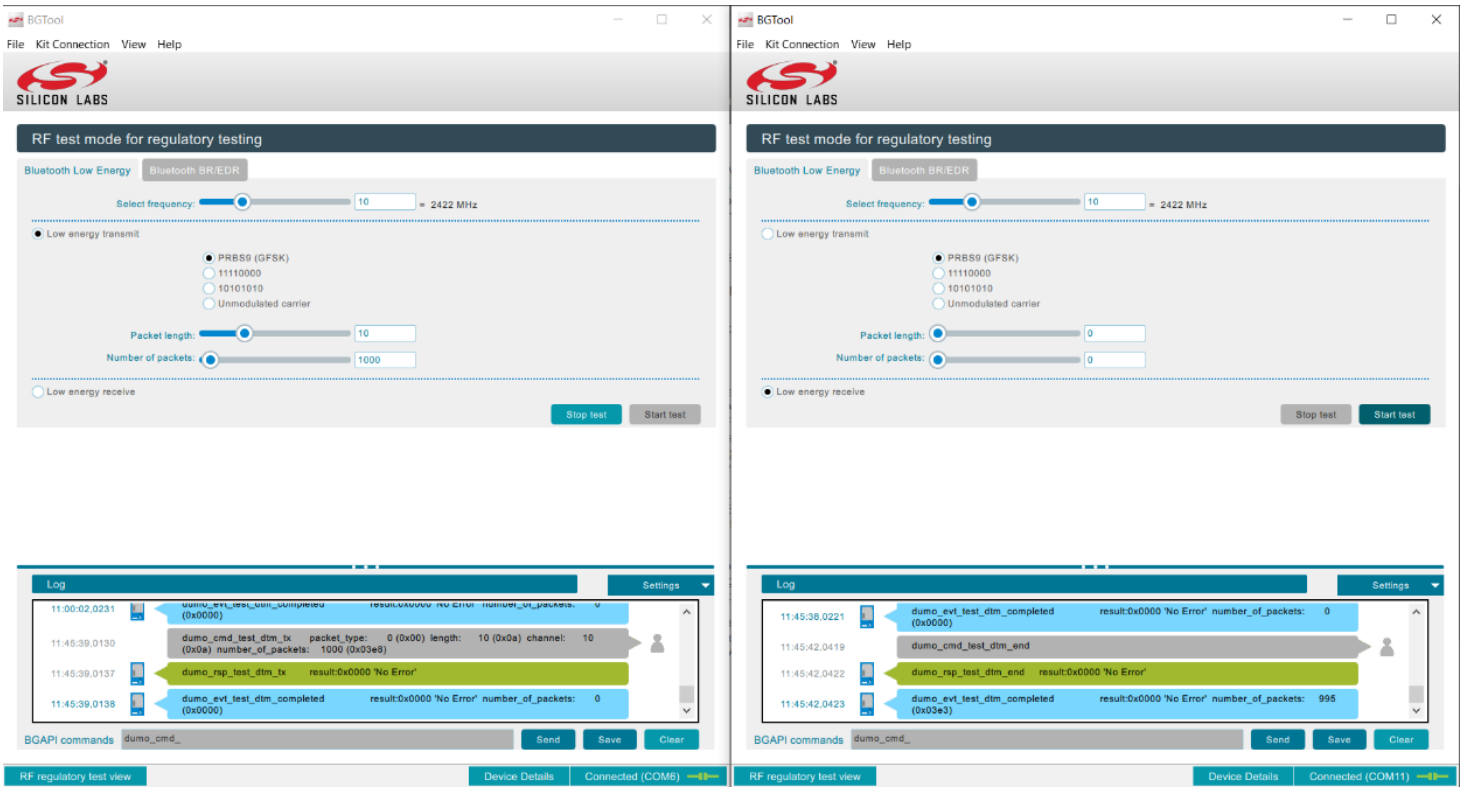

Figure 5.11 DTM Test

To test BLE connection open two BGTool windows and connect two BT122s in the **Bluetooth Low Energy** tab. One module will **transmit Low Energy packets** and the other will **receive Low energy packets**. First, **Start test** on the receiving device and then on the transmitting one. After that, press **Stop test** on the receiving device. In the **Log,** you should see how many packets were received during the test.

In the **Bluetooth BR/EDR** tab, you can test packet transmission and reception. The easiest method to test whether one device is receiving packets is to measure its current consumption. If the module is receiving packets, current consumption increases.

Connect two BGTools to two BT122 devices. One device will transmit packets chosen in **Packet type** field, using either **Hopping** or **Single Frequency** Mode. Packet size is set by the **Packet size** slider. Transmitter power is set by the **Transmit power** slider. Make sure to **Select Tx/Rx Frequencies** on both devices to equal when using **Single Frequency**. Data **Whitening** can be enabled or disabled. On the transmitter, you can **Enable RX for transmit test**, which will make the device receive packets while transmitting. The other device will only receive packets.

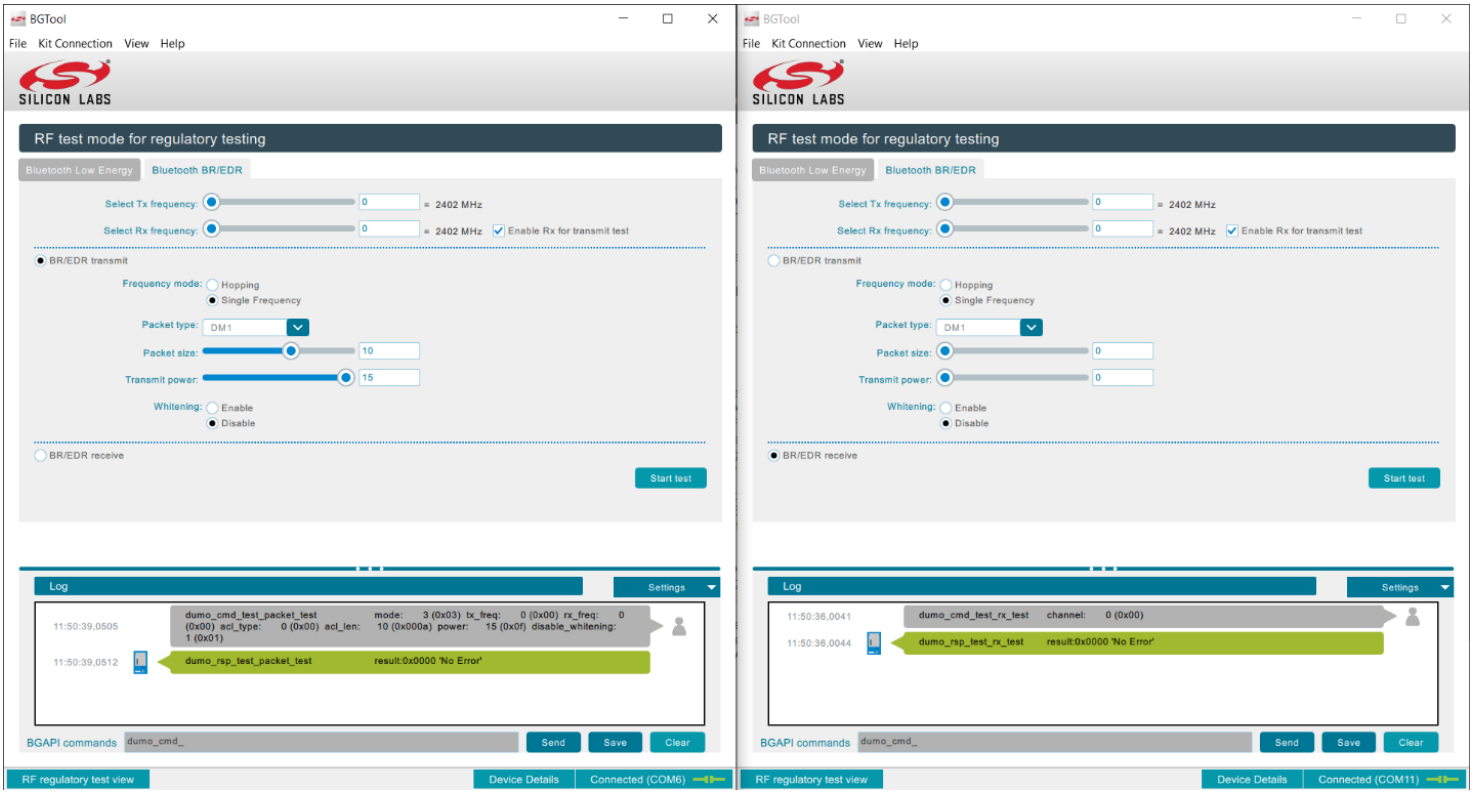

Figure 5.12 Bluetooth BR/EDR Packet Test

6 Quick Example Guide

To upload an example into the module, follow the instructions in section 5.2, where you can see how to browse for a desired project.xml file. For instance, to upload the bridge example, select example/bridge/project.xml file, build, and upload it afterwards.

6.1 BRIDGE

This example "converts" transparently Bluetooth Classic SPP (Serial Port Profile) to Bluetooth Low Energy Serial. When a device sends any data through SPP, the module resends it through BLE Serial to the other device back and forth. A BRIDGE project is implemented to work with the LE_CABLE_REPLACEMENT_CLIENT example.

Build and upload a LE_CABLE_REPLACEMENT_CLIENT example using BGTool to one device, and BRIDGE to the second device. BGScript code is designed in such a way that the Client and the Bridge will connect to each other.

Go to Bluetooth & other devices settings on your PC and turn the Bluetooth on. Next, click Add Bluetooth or other device. Add a device window will appear. Select the Bluetooth option and wait for Bridge BT devices to be discovered. Click it and wait until the device is ready.

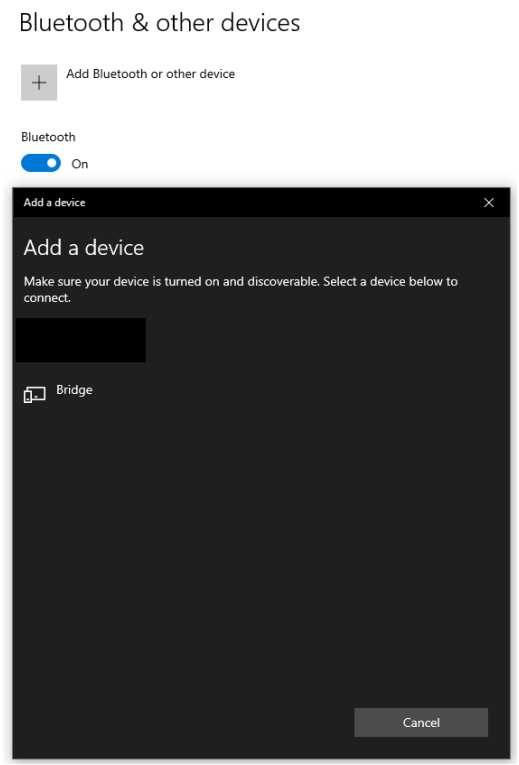

Figure 6.1 Adding a Bluetooth Device

Additional two serial COM ports should now appear in the Device Manager, as follows:

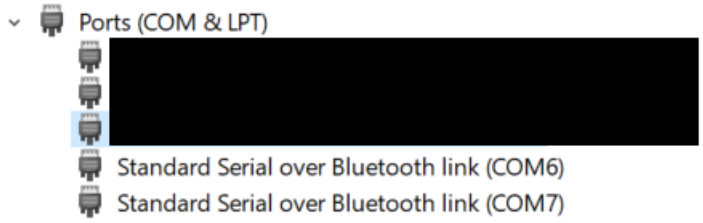

Figure 6.2 Bluetooth Serial Ports

Next, open three instances of a serial port terminal application (such as PuTTY, RealTerm, and so on) and connect to the two BT122 already programmed modules with 115200 baud rate. The third terminal should be connected to the first listed Bluetooth link in order (for example, COM6 in Figure 6.3). Restart both the BRIDGE and LE_CABLE_REPLACEMENT_CLIENT.

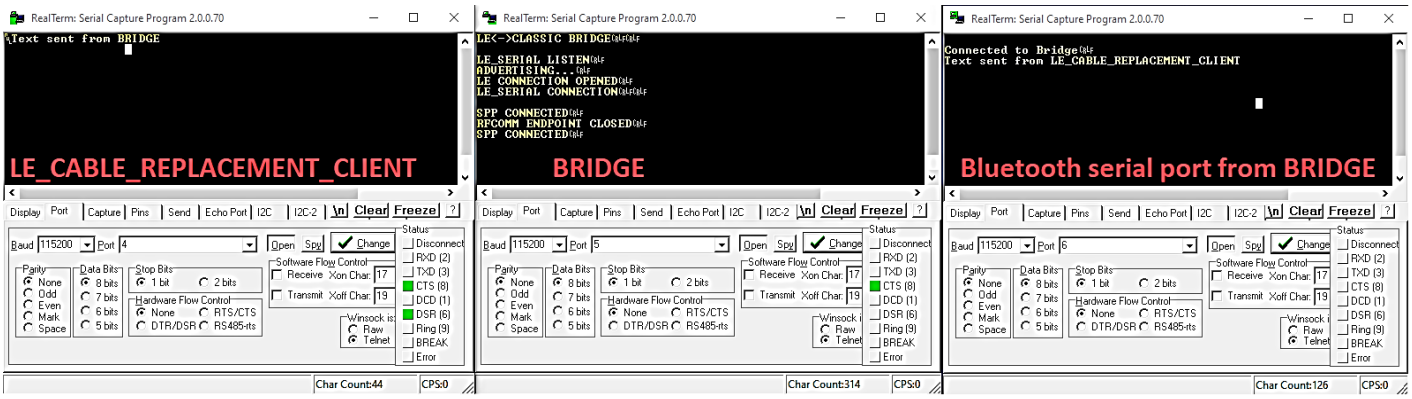

Figure 6.3 Sending Text with LE Serial and SPP

When you type data in the LE_CABLE_REPLACEMENT_CLIENT terminal, the same text should appear in the Bluetooth serial port terminal.

6.2 BT122

This is an empty example without any BGScript code which configures the BGAPI to be available over the UART interface. This is typically used for controlling the module using the BGTool. Modify the hardware.xml file to test different peripherals and features, such as sleep modes, GPIO, I2C, or ADC. You can consider this example as a starting point for developing your own application because it is an unmodified default project. For guidelines about including the necessary resources in the project and configuring the hardware interface settings for the modules, see the **Project Configuration Guide**.

An example hardware configuration, where the sleep functionality of the module is disabled and UART with BGAPI protocol is enabled along with appropriate UART parameters is shown below. PF2 pin (LED1) is configured as GPIO output pin. PF3 pin (BTN0) is configured as a GPIO input with pullup.

```
<hardware>

    <!-- Sleep modes disabled -->
    <sleep enabled="false"/>
    <!-- UART @115200bps, RTS/CTS and BGAPI serial protocol enabled -->
    <uart baud="115200" flowcontrol="true" bgapi="true" />
    <!-- Set PF2 (LED1) as output -->
    <port index="2" output="0x0004" />
    <!-- Set PF3 (BTN0) as input with pullup -->
    <port index="2" input="0x0008" pullup="0x0008" />

</hardware>
```

Figure 6.4 BT122 Project Hardware Configuration .xml File

Open the Upload Tool tab in BGTool, and select the BT122 example project file, build it, and upload the generated binary file into the module. Next, connect with the device in the Interactive View.

Now, use the **`dumo_cmd_hardware_read_gpio(2, 0x0008)`** command to read the state of the BTN1. In the response, received data will show which pins are in the high state. If BTN1 is released, data will be equal to 0x0008. If the BTN1 is pressed, data will be equal to 0x0000.

Using the dumo_cmd_hardware_write_gpio(2, 0x0004, 0x0004) command, you can turn LED0 off. With the **`dumo_cmd_hardware_write_gpio(2, 0x0004, 0x0000)`** command, you can turn it back on.

6.3 CLASSIC_HID_MOUSE_DEMO

In this example, the device is configured as a Human Interface Device (HID) mouse device. BGScript code will take control over the device and BGAPI commands sent to the module will not be processed.

Upload example to the device. Go to **Bluetooth & other devices** on your computer and enable Bluetooth. Click Add Bluetooth or another device. An **Add a device** window will appear. Select the Bluetooth option and wait for the **BT122-HID-Mouse** to be discovered. Click it and wait until the device is ready.

Now, when BTN0 (PF3) is pressed while a HID connection exists, a debug message is sent to UART and a HID report is sent indicating a left mouse button click. If BTN1 (PF2) is pressed, a debug message is sent and mouse pointer will move down 5 pixels. If HID connection does not exist, the device will enter DFU mode.

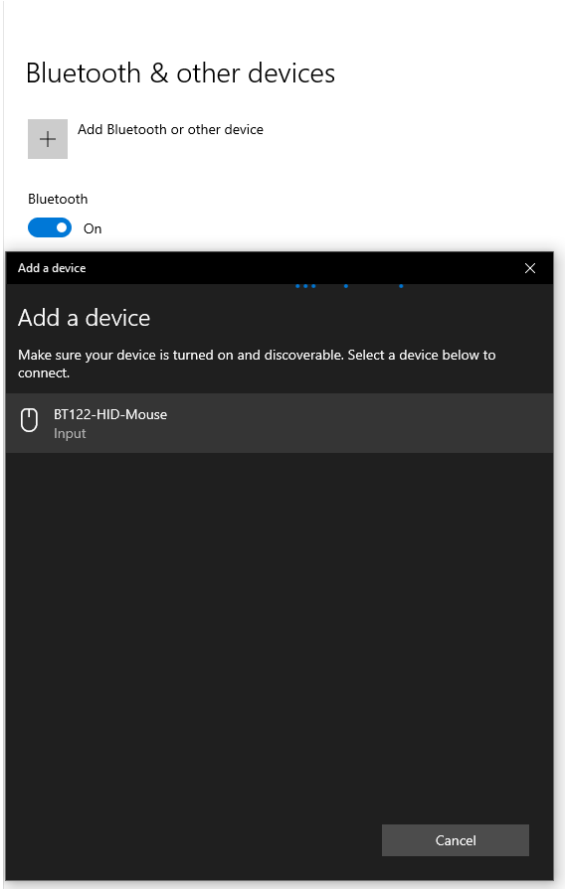

Figure 6.5 Adding HID Device

6.4 BT122_HID_ BGAPI

In this example, the device is configured as a HID, a mouse or keyboard, just like in the previous example. No BGScript is uploaded. HID reports can be sent via the BGTool and appropriate commands. This example is very similar to the BT122 example, but includes a pre-configured HID.

6.5 LE_CABLE_REPLACEMENT_CLIENT/SERVER

Use these examples together, where one device is a Client and the other a Server. The devices will stream serial data through the first device's UART, LE Serial service, and the second device's UART.

Using the BGTool, build and upload the LE_CABLE_REPLACEMENT_CLIENT example to one device, and the LE_CABLE_REPLACEMENT_SERVER example to the second device. BGScript code is implemented in such a way that the Client and the Server will connect to each other. Open two serial port terminals (such as PuTTY, RealTerm, and so on). Connect with both devices with 115200 baud rate. Now, when typing any text in one terminal, the same text will appear in the other terminal.

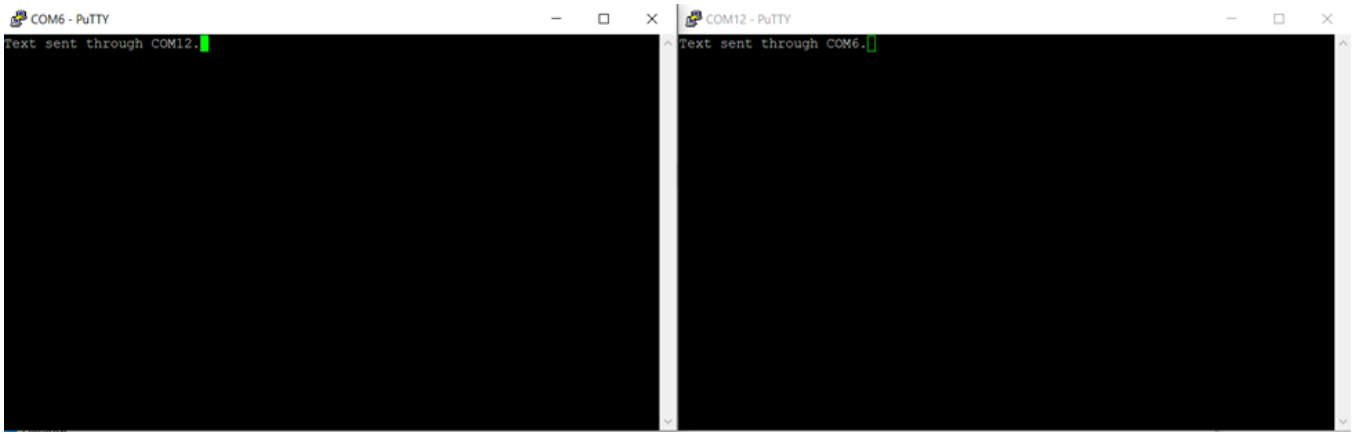

Figure 6.6 Terminals Connected to Appropriate COM Ports

6.6 LE_SI7021

In this example, the module reads temperature and humidity from an on-board Si7021 sensor. Data is sent through BLE by notifications to connected devices.

Upload the LE-SI7021 example to one device and open an application, which allows searching BLE devices, for example a Bluetooth LE Explorer, and try to search for a device. The device should be visible as a BT122 Si7021 Sensor, as shown in the figure below.

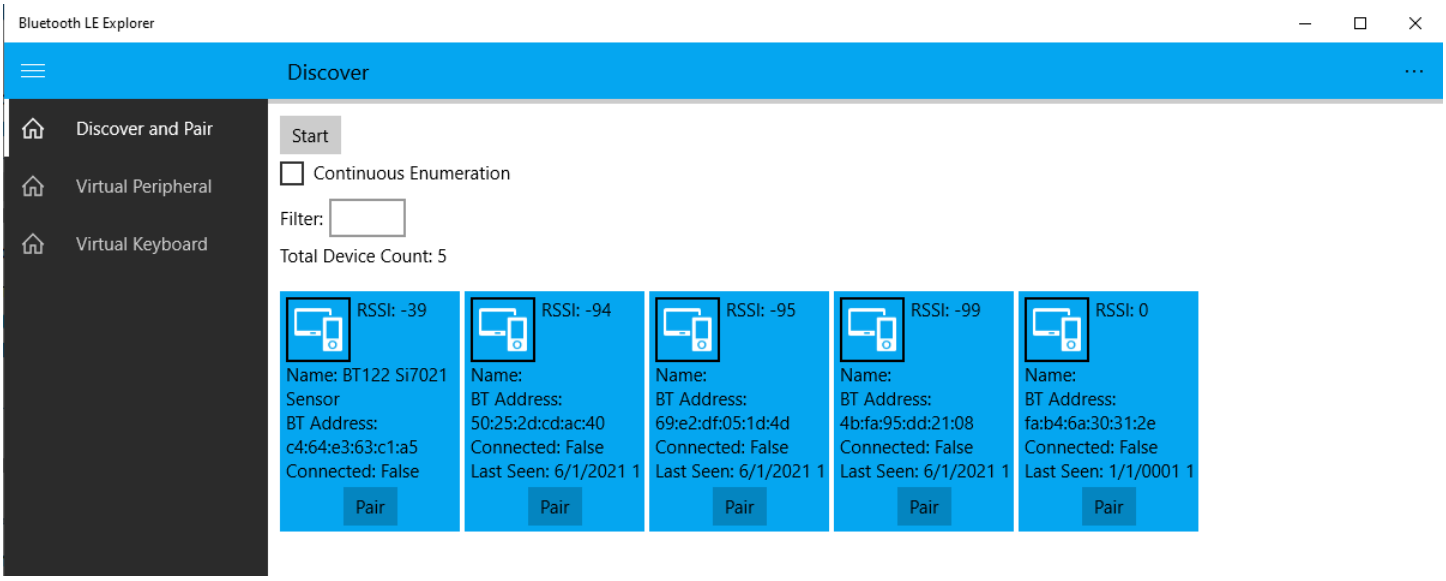

Figure 6.7 Found BT122 with LE Si7021 Example

Next, try to establish a connection with modules. If devices are correctly connected, all implemented services and characteristics should be visible, as shown in the figure below.

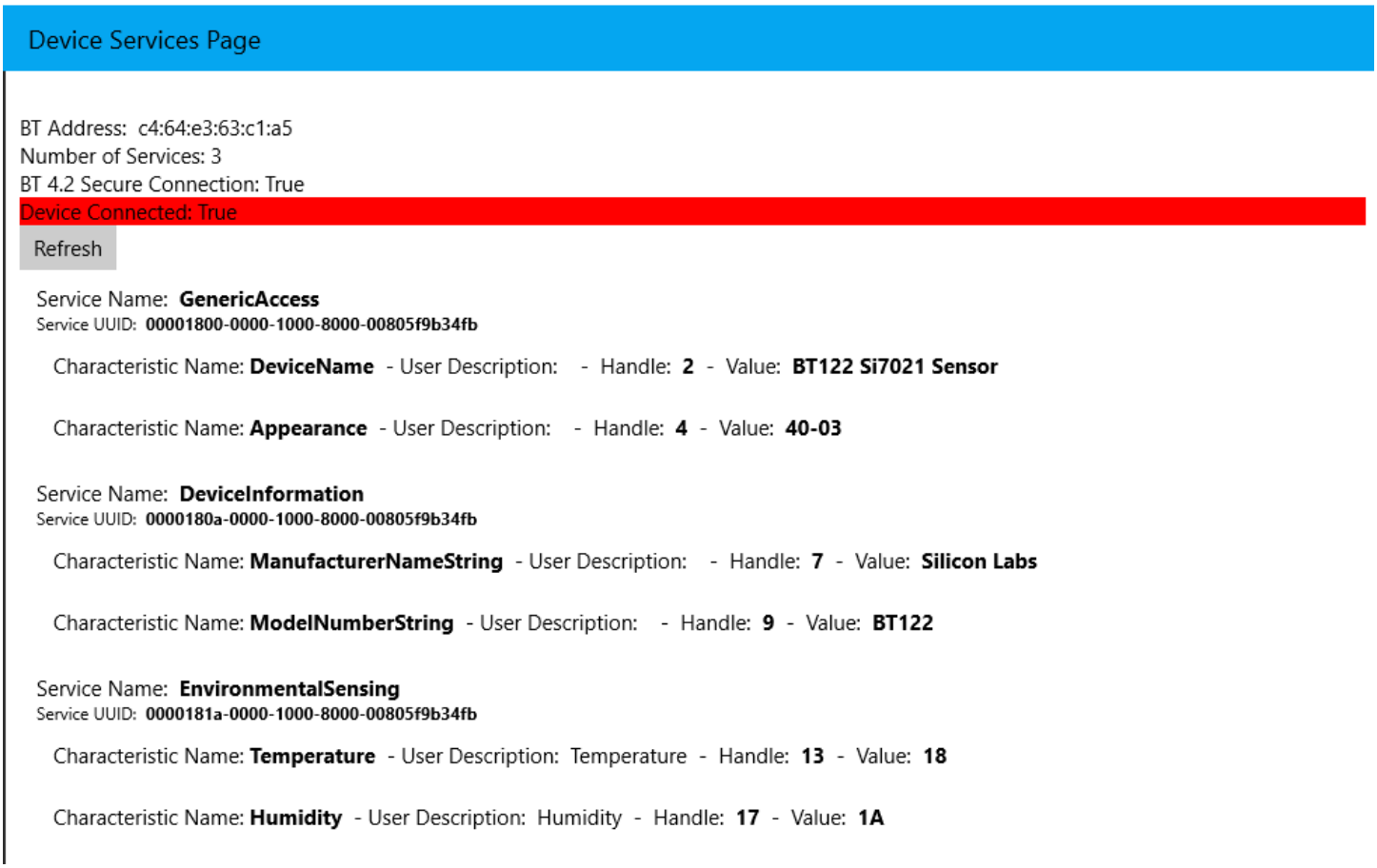

Figure 6.8 Services and Characteristics of Example Visible in Remote Device

Next, characteristics from EnviromentalSensing service can be notified to instantly read data if its value is changed. Also, value can be read manually by read options. The temperature is measured in ˚C and the relative humidity in %. For more details about the Si7021 sensor, see the product data sheet.

6.7 LE_TEMPERATURE_SENSOR

In this example, ADC will measure the supply voltage or temperature of the module MCU. These results will be stored in GATT characteristic, which can be read by the other device.

Upload the LE_TEMPERATURE_SENSOR example to one device and the BT122 example to the other device. With the BGTool, connect to the device with the BT122 example. In the Bluetooth Low Energy tab, press Start in the Discover (Central) tab. When “BT122 Thermometer” is found, stop the discovery procedure, and connect to the device.

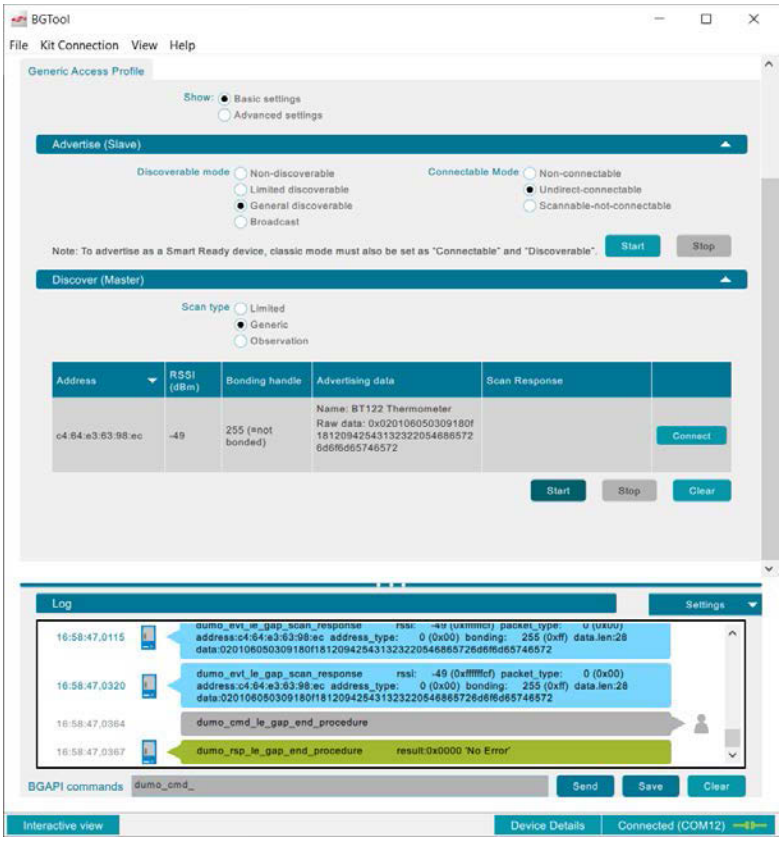

Figure 6.9 BT122 Thermometer is Discovered

After the connection is established, press the **Discover remote services** button to display the Thermometer's services. Open the **Health Thermometer** service and **Discover Characteristics**. Enable the **Notify** operation and press **Apply**. Now, every second the Thermometer device will send the measured temperature in degrees Celsius (hexadecimal format), which you can find in the **Hex byte value** field.

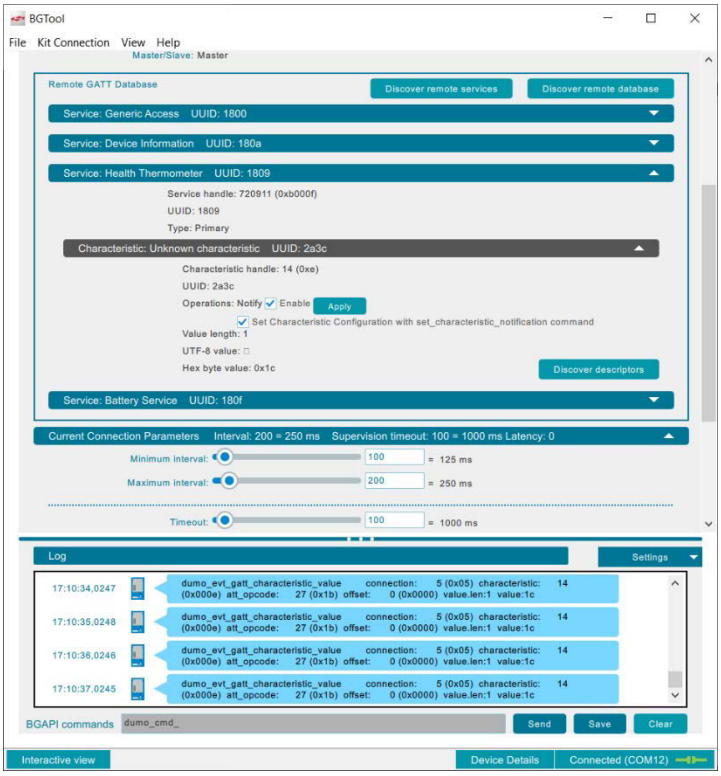

Figure 6.10 Health Thermometer Characteristics Updating with Notifications

6.8 UART_MODES

In this example, the module will ignore or process incoming BGAPI commands, depending on the pushbutton state. As to the first option, the BGScript has full control over the device, and so the BGTool will not, for example, be capable of exchanging data via the Bluetooth link. When the BGAPI commands can be received, the user holds some control over the device, but the BGScript will still respond to events which are handled in the BGScript code.

The device can operate as a central (the device will search for a RFCOMM server with a given address) or as a peripheral (the device will start a RFCOMM server, to which other devices can connect to), depending on the value of "script_version" variable. The "slave_address" buffer in the BGScript code is used only when the device operates as a Central and holds the MAC address of the device, which has RFCOMM server configured. For more details, see the *uart_modes.bgs* file.

If the device is configured to be a **central** (script_version = 1), set the MAC address of a peripheral device in the *uart_modes.bgs* file (slave_address buffer. To find the MAC address of a device, use the `dumo_cmd_system_get_bt_address()` command **(**see chapter 5.3), upload the modified script/example/project to the first module, and upload the BT122 demo to the other one. On the device programmed with the BT122 example, go to the **Security Manager** tab, change **Bondable mode** to **On**, and press **Set bondable mode**. Next, go to the **Bluetooth BR/EDR** tab and change Generic Access Profile **Mode** to **Connectable** and **Discoverable**, then, press **Set mode**. You can now **Start Server**. The configuration should look like this:

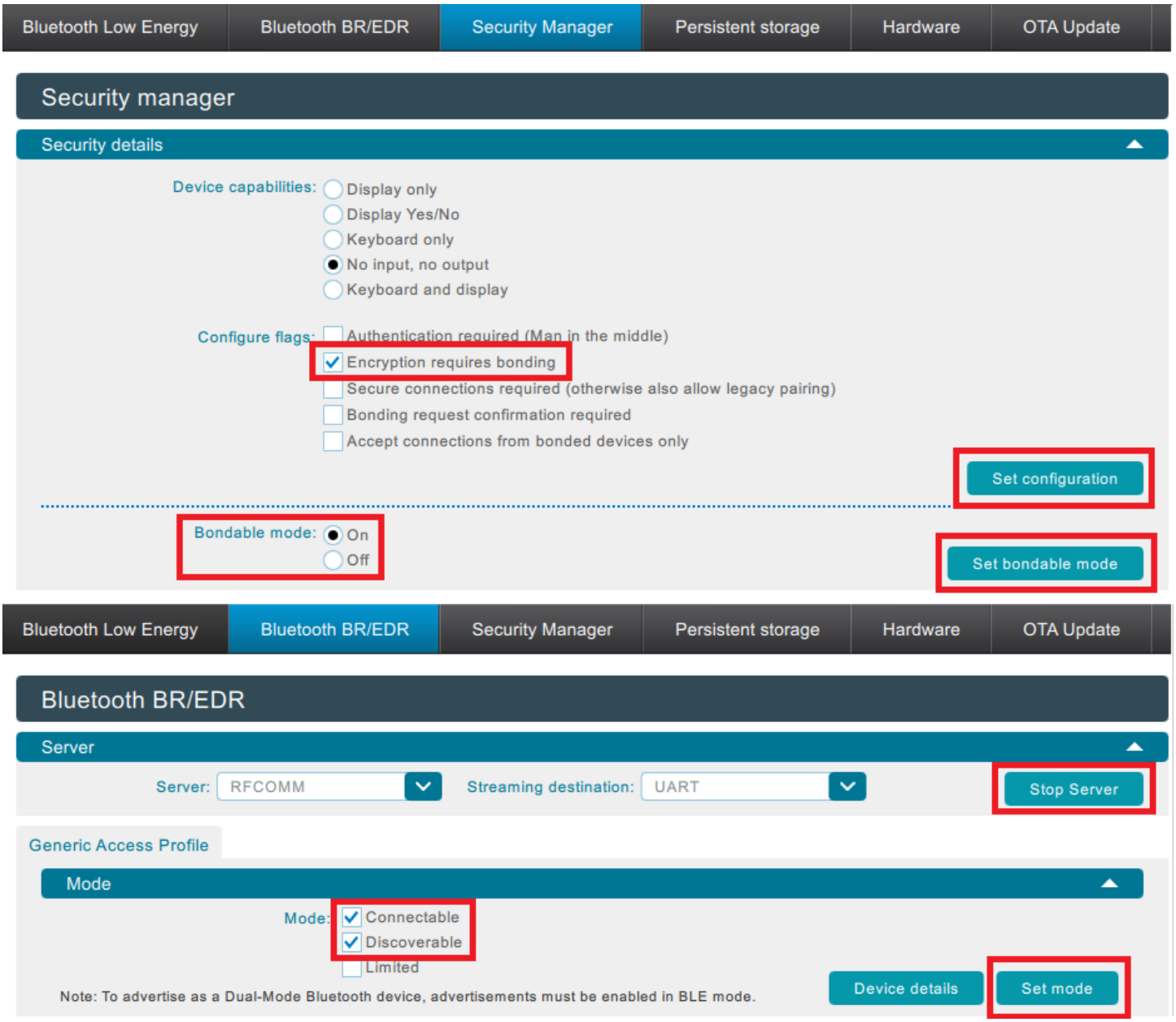

Figure 6.11 BT122 as Peripheral Configuration

The device with the UART_MODES example configured as a central should now be able to connect to the BT122 peripheral, as a **Connection #** tab will appear. If it does not connect automatically, reset the UART_MODES device manually using the RESET button.

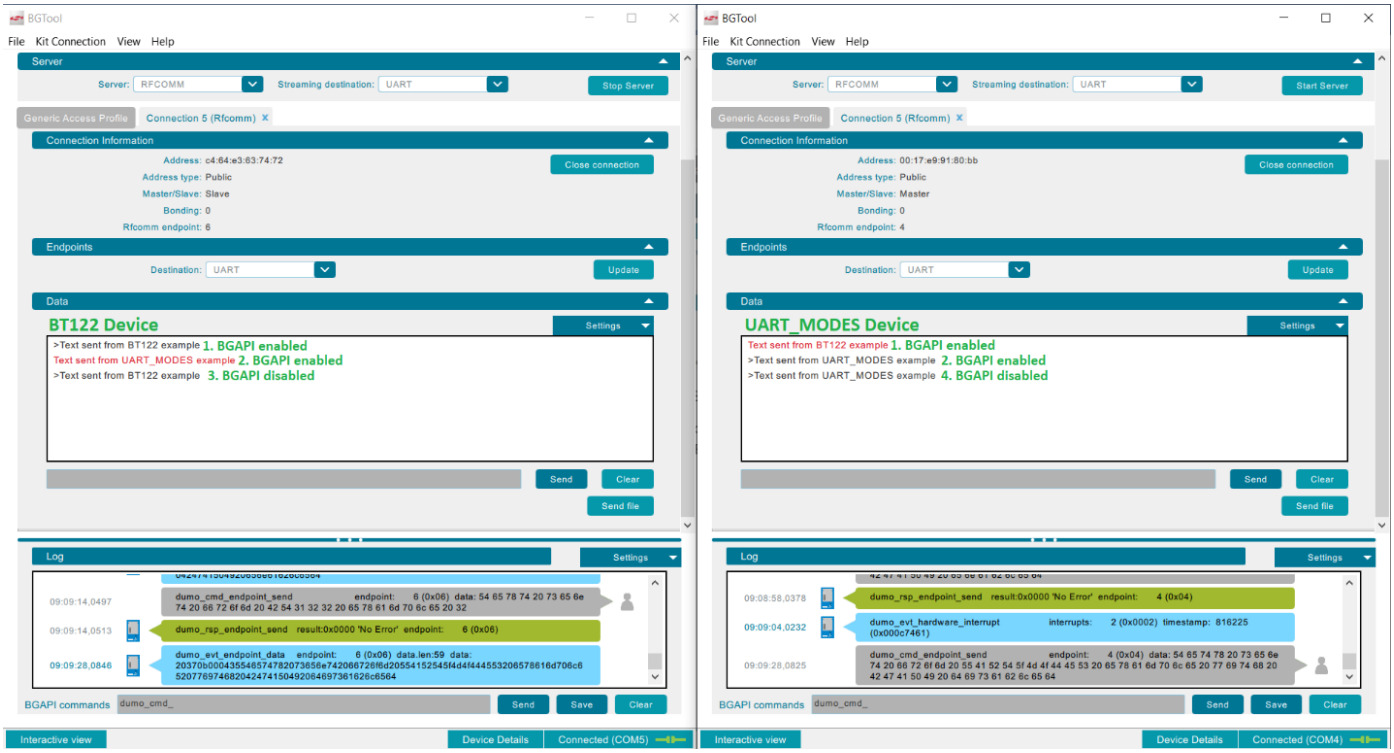

Figure 6.12 BT122 Peripheral (Left) and UART_Modes Central (Right)

By default, BGAPI is enabled on the UART_MODES device. The first text is sent from BT122 device and it is received by UART_MODES BGTool. The second text is sent from the UART_MODES device and it is received by BT122 BGTool.

When BTN0 is pressed, the UART_MODES device will disable BGAPI mode and will not be able to receive BGAPI commands or send received data back to the BGTool. The third text is sent from the BT122 device and it is received by UART_MODES device, but the BGAPI event is not sent to UART_MODES BGTool. The fourth text is sent through the UART_MODES BGTool to the device, but the BGAPI commands are disabled and the UART_MODES device ignores it.

When BTN1 is pressed, the device will enable BGAPI back again. After reset, BGAPI is enabled by default.

If UART_MODES is configured to be a **peripheral** (script_version = 0) device, the BT122 example should have only the **Security Manager** configuration set.

To connect to the UART_MODES device, set bonding configuration, as described above, and in the **Bluetooth BR/EDR** tab in the **Discover** section press **Start**. The UART_MODES device address should appear. Press the **RFCOMM** button to connect. The exchange of data looks like Figure 6.12.

Figure 6.13 Connecting to the UART_MODES Peripheral

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

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

www.silabs.com

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Silicon Labs:

BT122-DK4315B