UG497: BT122 User's Guide

This document provides information about BT122 SDK's structure and installation procedure, usage of developer SW 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 examples guides.

Table of Contents

1 Version History

Table 1.1. Version history with comments

Version	Comment
1.0	Initial version

2 SDK

The SDK (Software Development Kit) is a set of tools for users 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 official Silicon Labs web page and run the executable. Follow the on-screen instructions.

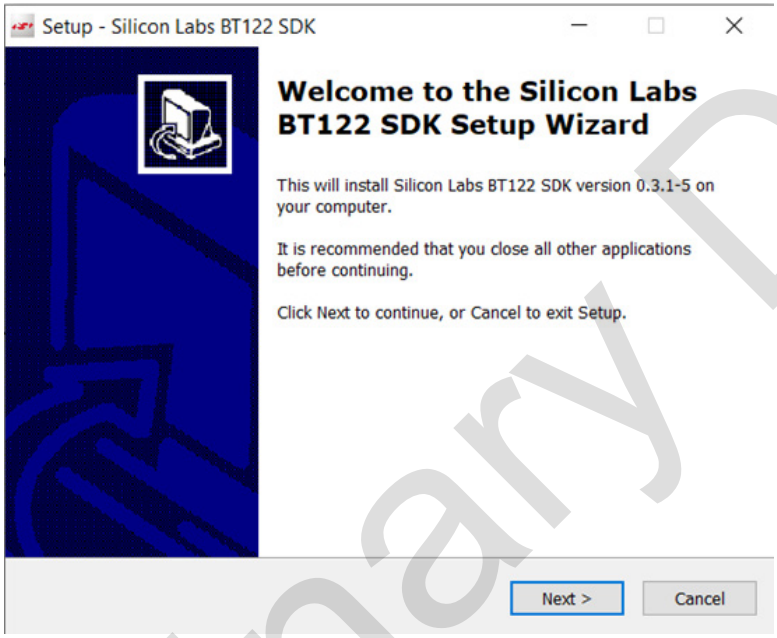

Figure 2.1 SDK Installer

2.2 Folder structure

Once the SDK installation is finished you will see the following folders in the installation directory:

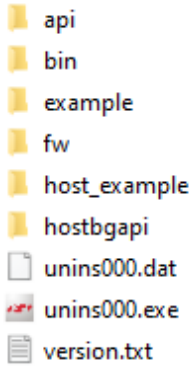

Figure 2.2 SDK Folders

The folders contain the following:

- **API** – includes a raw XML description of the Bluetooth stack's API. It could be used, for example, to automatically generate parsers for the API.
- **BIN** –Includes SDK executable sources. Programs such as BGTool , BIMBER or BGBuild can be found there.
- **EXAMPLE** – includes example BT122 projects.
- **FW** – contains all binaries required by the firmware. Also, FWs for on-board J-Link/VCOM interface can be found there.
- **HOST_EXAMPLE** – contains example code for 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 target for users, who want to use both Bluetooth BR/EDR and LE functionality and use all the 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 it can be easily ported to various processor architectures.
- **BGScript** interpreter and scripting language allow applications to be developed into the Bluetooth Dual Mode modules built-in MCU. It allows simple end user applications or enhanced functionality to be developed directly into the Bluetooth Dual Mode module which means no external host MCU is necessarily needed. BGScript applications can be executed at the same time as the BGAPI is used, allowing the possibility to implement some functionality on the Bluetooth module and some on the host.
- **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 into 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 controller 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 tools created because of end of iWRAP modules firmware support. Migrator allows to convert dump of Persistent Storage from iWRAP modules into BT122 project files (included BGScript).

3 Preparing the Board

3.1 Board Features

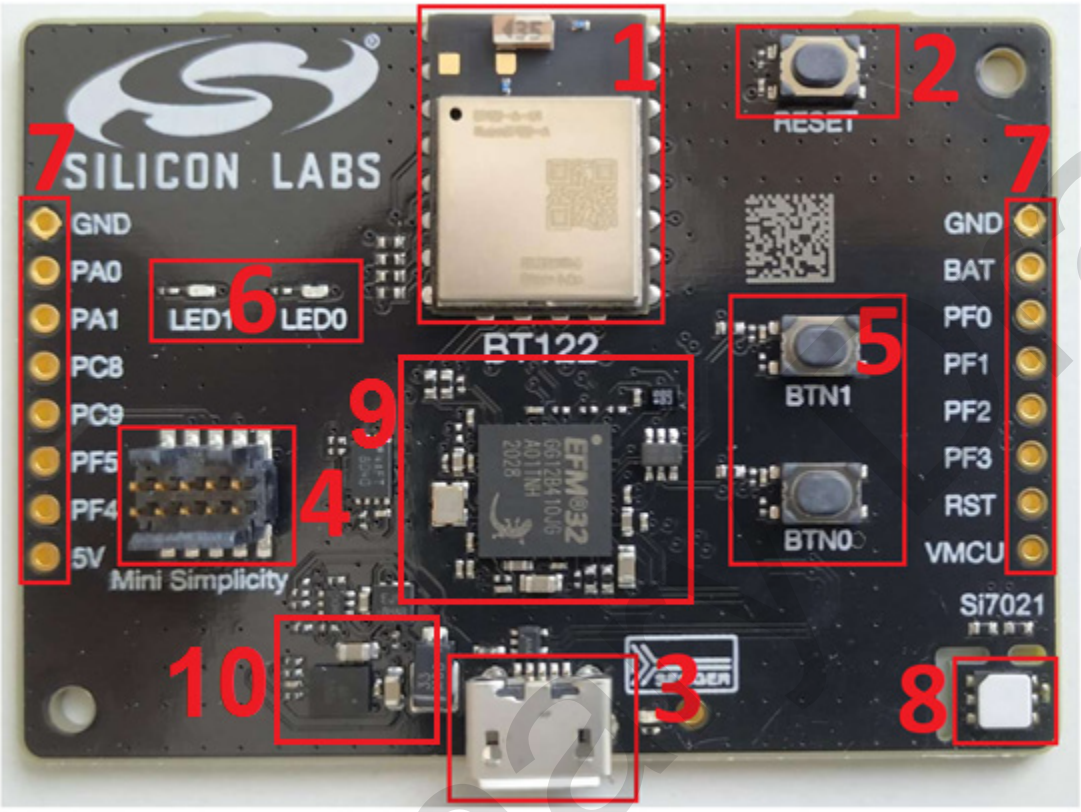

Figure 3.1 BRD4315A Development Board with BT122 module

BRD4315A Development Board includes:

1. BT122 module.
2. Reset button.
3. Micro USB connector for communication (J-Link and VCOM).
4. Mini Simplicity connector.
5. Buttons – BTN0 (PF3) and BTN1 (PF2). If a button is pressed, pin state will be low; when it is released, pin state will be high.
6. LEDs – LED0 (PF3) and LED1 (PF2). When pin state is low, LED turns on; when pin state is high, LED turns off. LEDs are connected to the same pins as buttons are.
7. Breakout pads with all signals available for the user (more details in Table 2).
8. Si7021 – temperature and humidity sensor connected to the module via the I2C bus (PA0 – SCL, PA1 – SDA).
9. J-Link and VCOM circuit.
10. Power supply circuit.

3.2 Available Pins

Table 3.1 Some of the internal Development Board signals are available for user 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
I²C	SCL	SDA								
ADC Input	ADC_CH0	ADC_CH1						ADC_CH2		
Programing Interface					SWCLK/ TCK	SWDIO/ TMS	SWO/ TDO	TDI		

3.3 Programming via J-Link

Once the USB connection is established between the PC and 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 of 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 3.2.

Development board for BT122 – BRD4315A – got the J-Link (SWD) interface available and already connected as the on-board programmer for BT122.

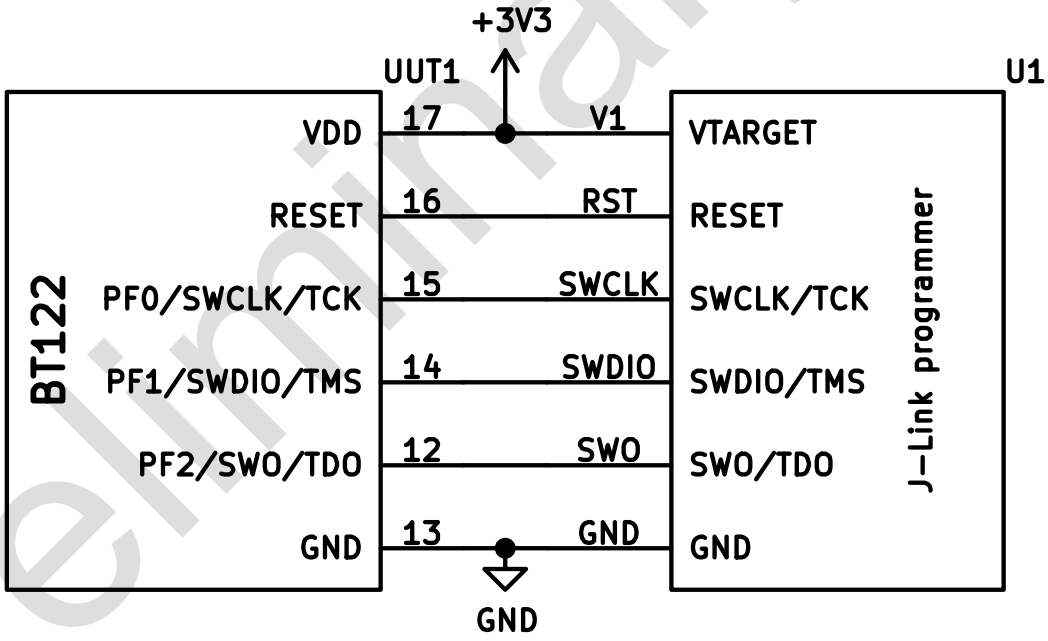

Figure 3.2 BT122 Firmware Update via J-Link Debugger and SWD Interface

3.4 Programming via Mini Simplicity Connector

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

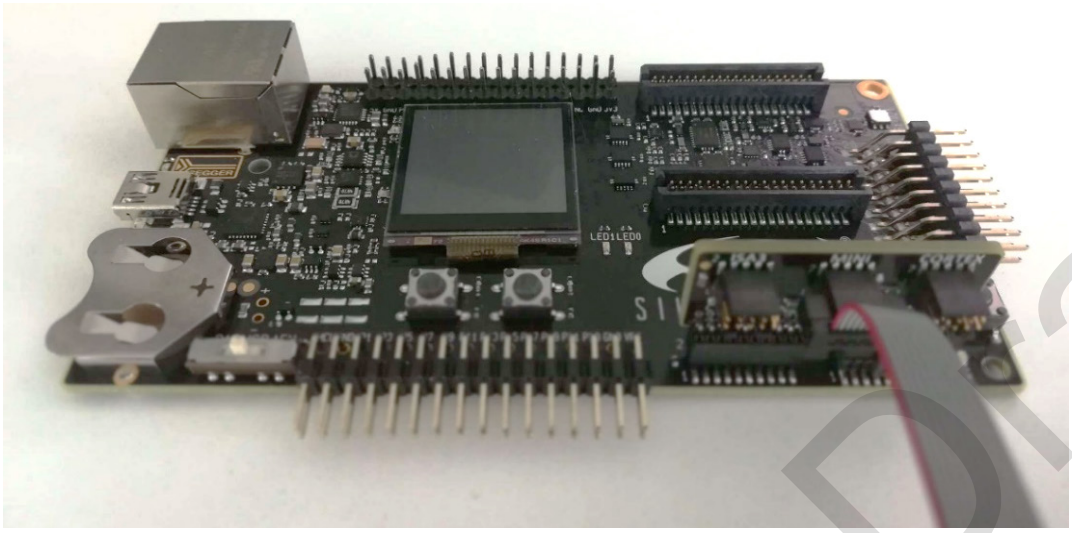

Figure 3.3 WSTK board with STK/WSTK Debug Adapter Connected

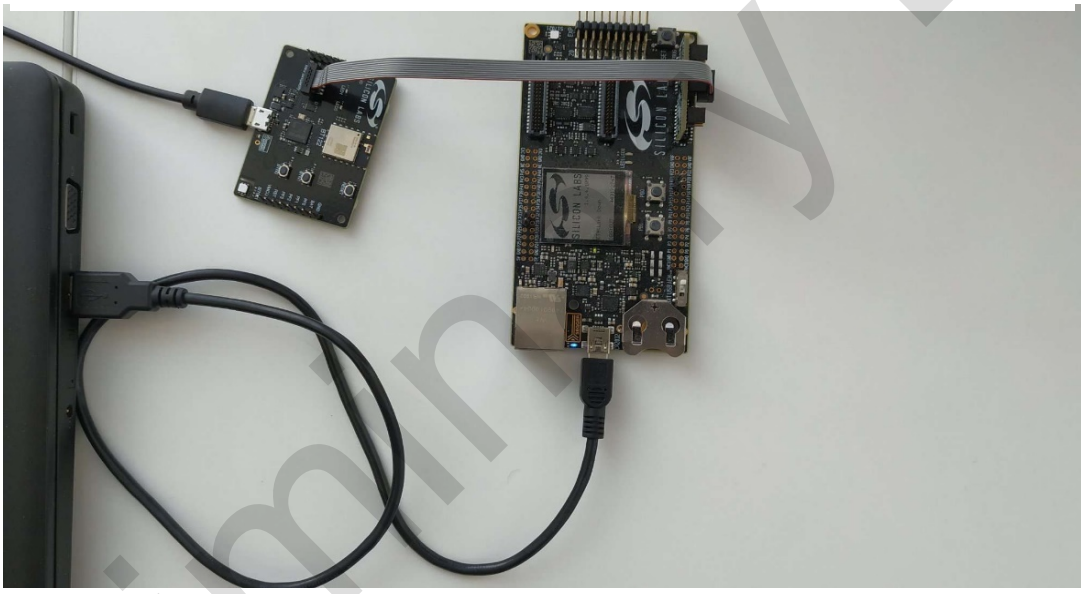

Figure 3.4 Finished Setup

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 3.4.

3.5 Programming via UART DFU

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, the new program can be uploaded 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.

Comparing to the J-Link method, user needs to make sure 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 `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.

Development board for BT122 – BRD4315A – got the UART/USB bridge with interface called VCOM – available on-board and already connected to BT122 module.

On-board UART/USB bridge FW can be updated to support baud rate setting 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>"
- Or by 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"

For special applications that require the highest performance of UART/USB connection, user 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, please 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.

4 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 the J-Link. 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 into the SDK package to be used by more advanced users outside the BGTool according to ones needs.

For Alpha delivery, all BT122 Development boards have the following VCOM settings: baud rate 115200 bps, handshake: auto (no flow control). Changes can be made according to the application needs.

Memory of BT122 modules must be considered as erased, so they must be flashed with proper application (bootloader and firmware) using J-Link (SWD) interface, before starting to use them for the first time. DFU cannot be used here because there is no DFU bootloader present in the module.

If you have prepared/selected the application to be uploaded to the BT122 memory and you use built-in VCOM solution, you need to configure the on-board UART/USB bridge settings according to your application settings – that will be most likely required, since all example apps use RTS/CTS based flow control (handshake).

4.1 VCOM configuration

To make sure that you will be able to connect to the board in the Interactive view and use all flashing options in Upload tool view it is needed to configure VCOM options. This can be done under the Development board VCOM tab (Figure 4.1).

Note: In case of external UART bridge, this subchapter can be skipped.

Development board's VCOM bridge does not use flow control by default, so to correctly connect to BGTool, first it is needed to check what settings the desired application has and set the same in mentioned tab. After correctly selecting Serial Number, baud rate and RTS/CTS option, press the Set button and if any system prompt will arise (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 baud rate is permanently stored in memory (until it is changed again).

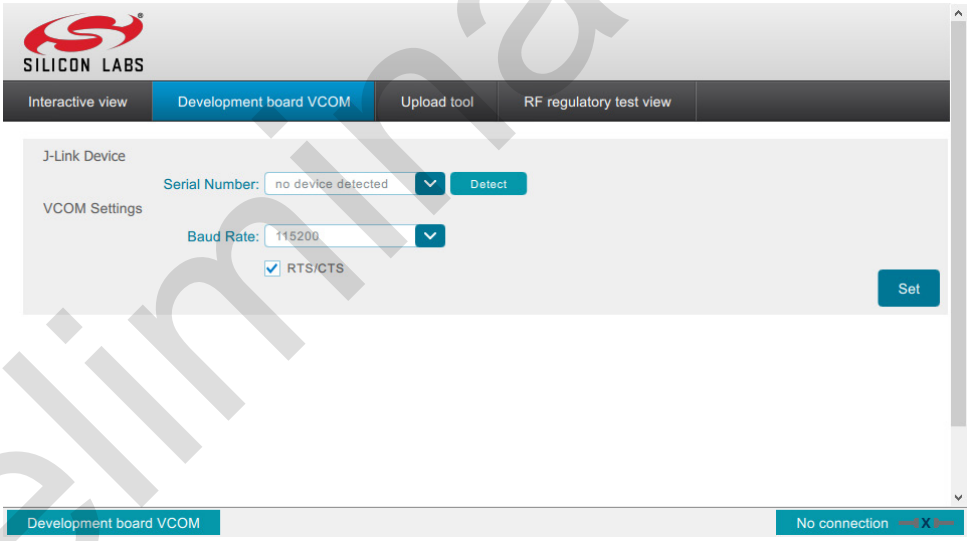

Figure 4.1 BGTool in Development board VCOM mode

The firmware does not support configuration over USB-CDC (although this may change in the future), nonetheless it is recommended to set the baud rate in the PC-side driver.

Additionally, for changing the VCOM bridge baud rate through CLI as an alternative to 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`"

Note: Once set, the baud rate is permanently stored in memory (until it is changed again).

4.2 Programming the Device

At the top of the BGTool's window there is a tab named **Upload tool** (Figure 4.2).

First, select a **Project File**. It is a *.xml file which 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. More information about it can be found in BT122 Project Configuration Guide. Browse for the desired project configuration file and select it.

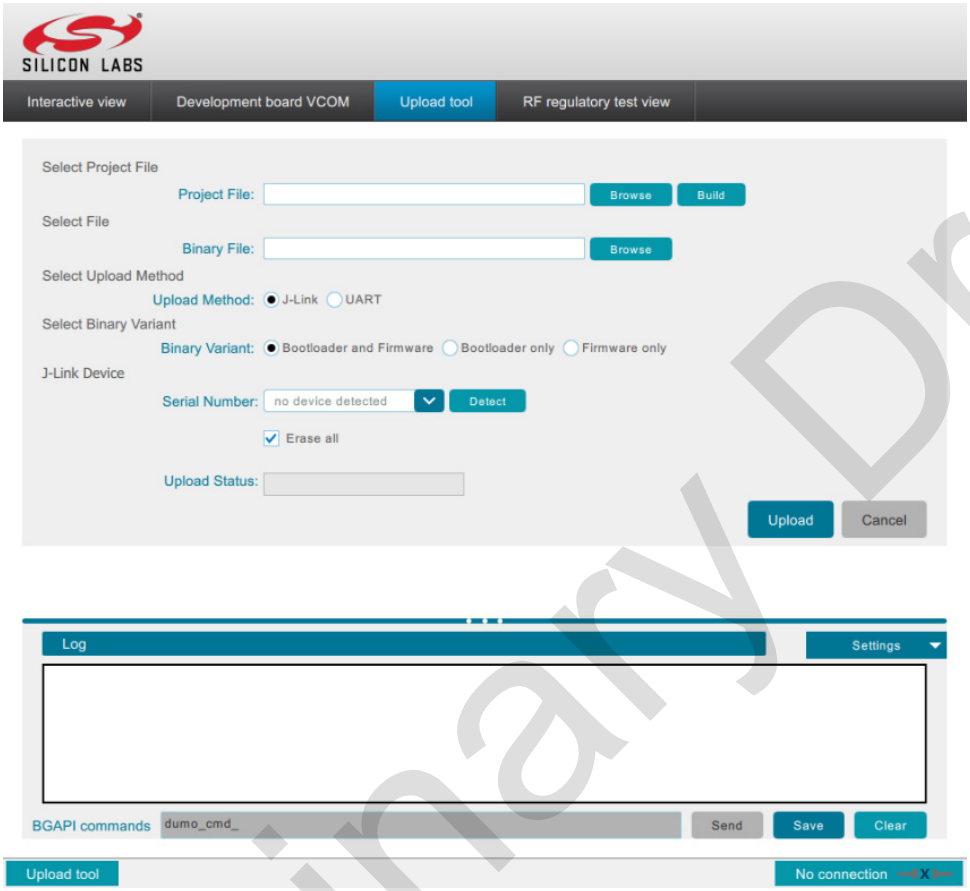

Figure 4.2 Upload Tool

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

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

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

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.

You do not have to build the project every time. If a binary file is already generated, you can browse for it and select it manually.

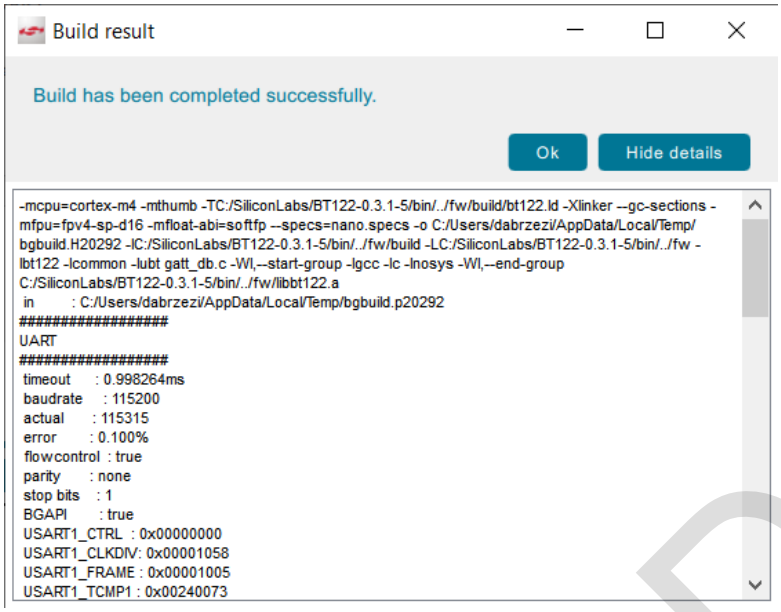

Figure 4.3 Project Building

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

4.2.1 J-Link

Selected binary variant will be uploaded through J-Link – this method is faster 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).

4.2.2 UART

Used when the device is in DFU (Device Firmware Update) 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.

Note: If a new module is use, first upload an example (e.g. *bt122* from SDK package). This should be done via the J-Link interface as there is no DFU bootloader in the module yet.

For DFU update, user needs to make sure that the proper bootloader is flashed into 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. Flow control must be used as well.

After selecting the interface option, you need to specify the variant of the built binary you want to upload to the module. It is a very important selection, since this option selects not only the correct file, but also the address range that this file will be written into. So, you need to select the variant that corresponds to the binary file content that you have selected. Please double check the selected file and options to avoid restoring the device. It is worth mentioning that it is good to keep the names of the files generated by BGBuild, because thanks to 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 be able to connect to the board in the Interactive view, and use all flashing options in 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 done also for VCOM settings in the BGTool to keep the ability to connect to the module.

4.3 Opening a Connection

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

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

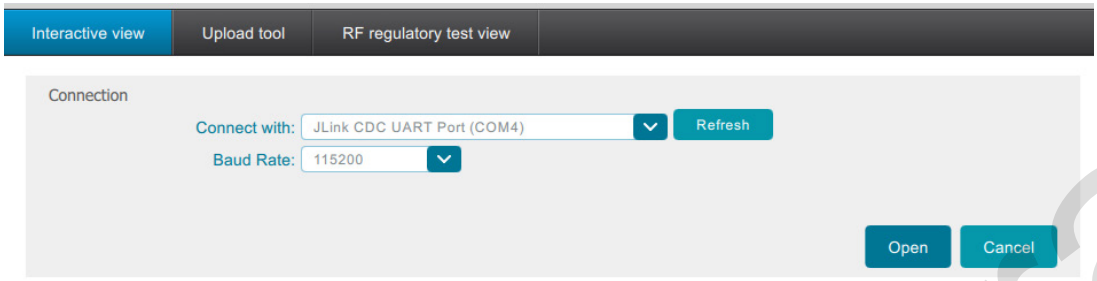

Figure 4.4 Opening a Connection

After that, a console will appear through which commands can be sent 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 then the device should respond with a **`dumo_rsp_system_get_bt_address`** response containing the device's address. If this happens, this means that everything works.

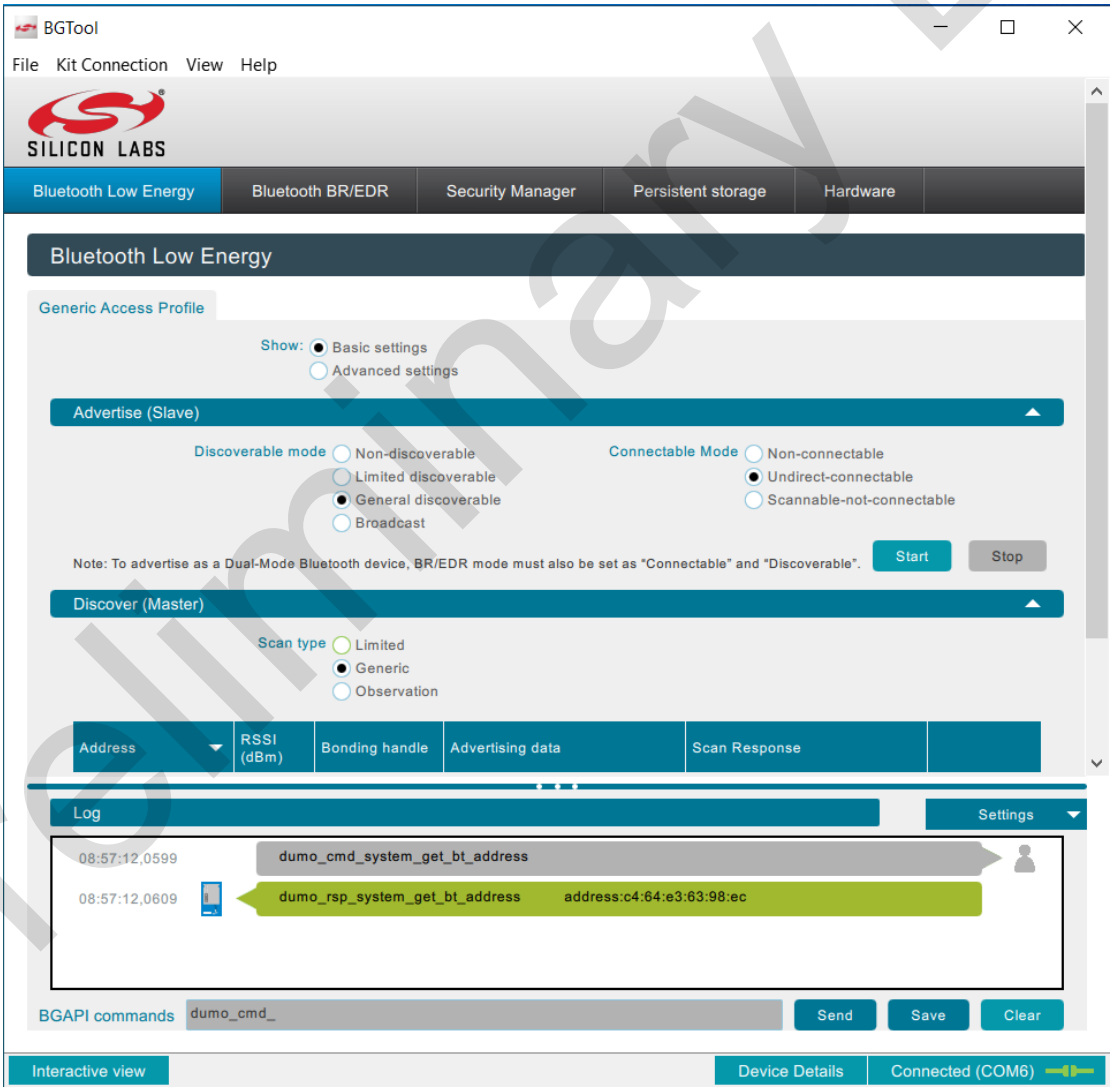

Figure 4.5 BT122 Responding with its address

4.4 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 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.

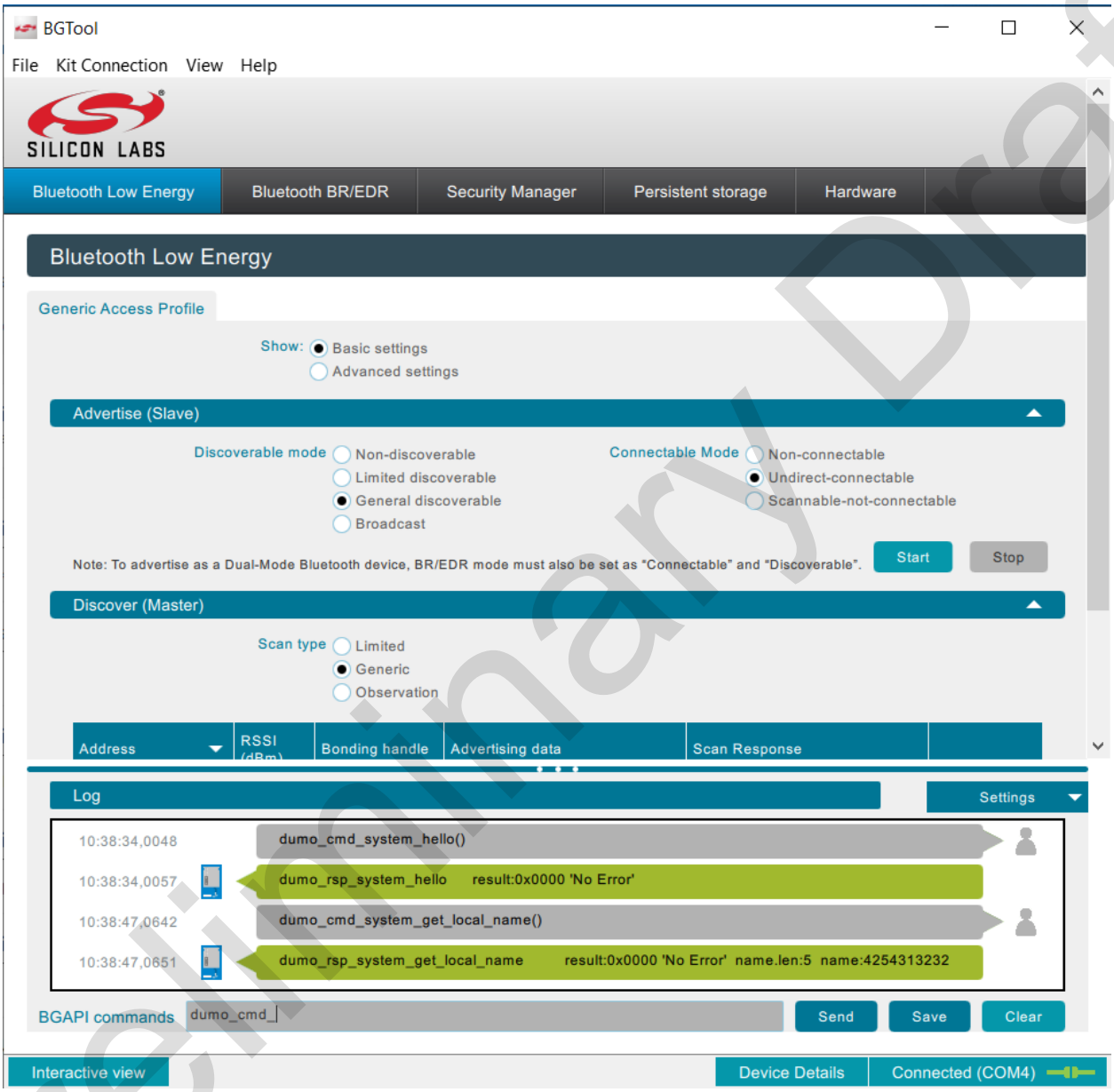

Figure 4.6 Sending Commands Manually

4.5 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 give you access to 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, etc.) by pressing the **Start** buttons under the appropriate tabs.

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

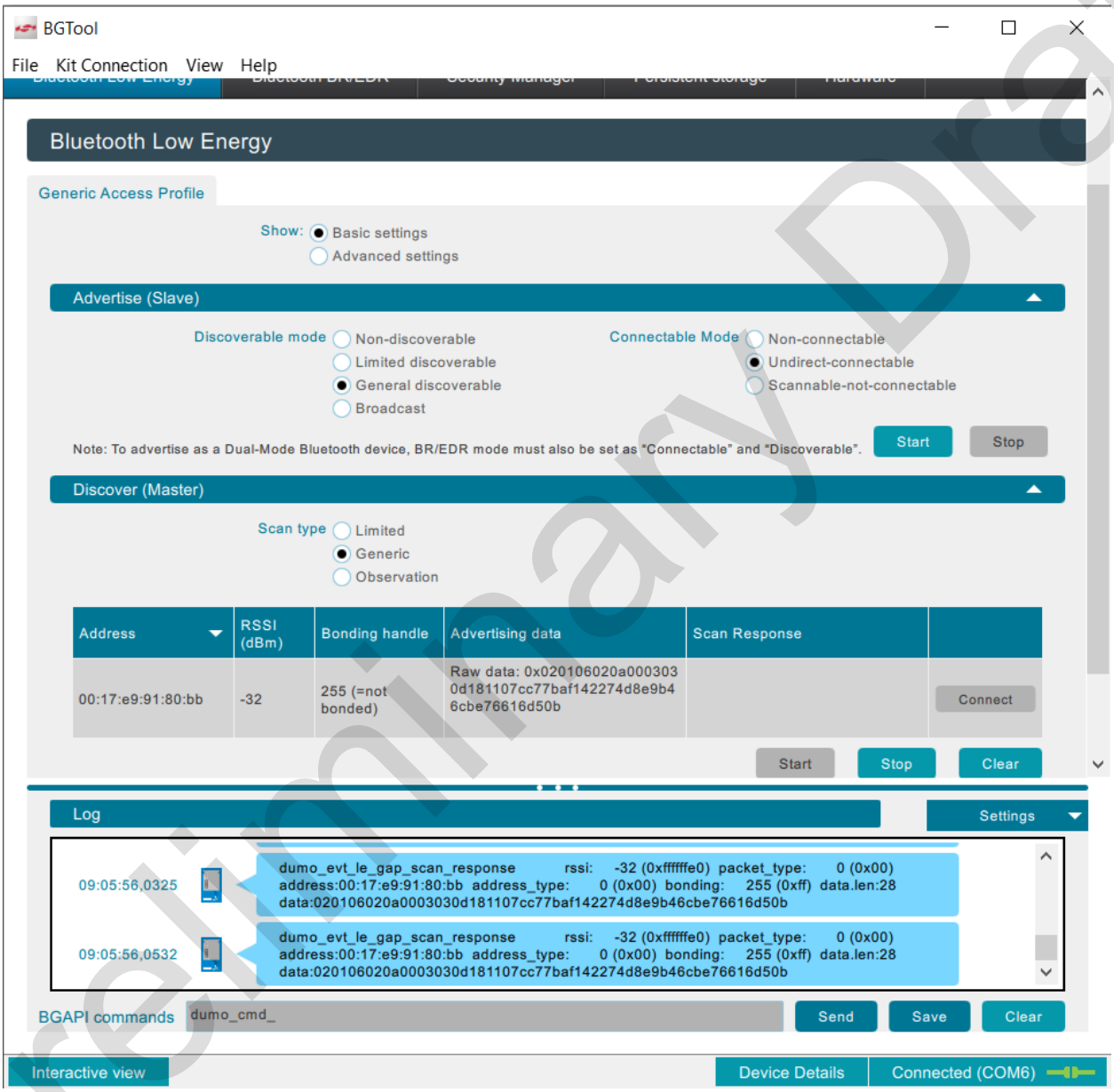

Figure 4.7 Example – Scanning for BLE Advertising Packets

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.

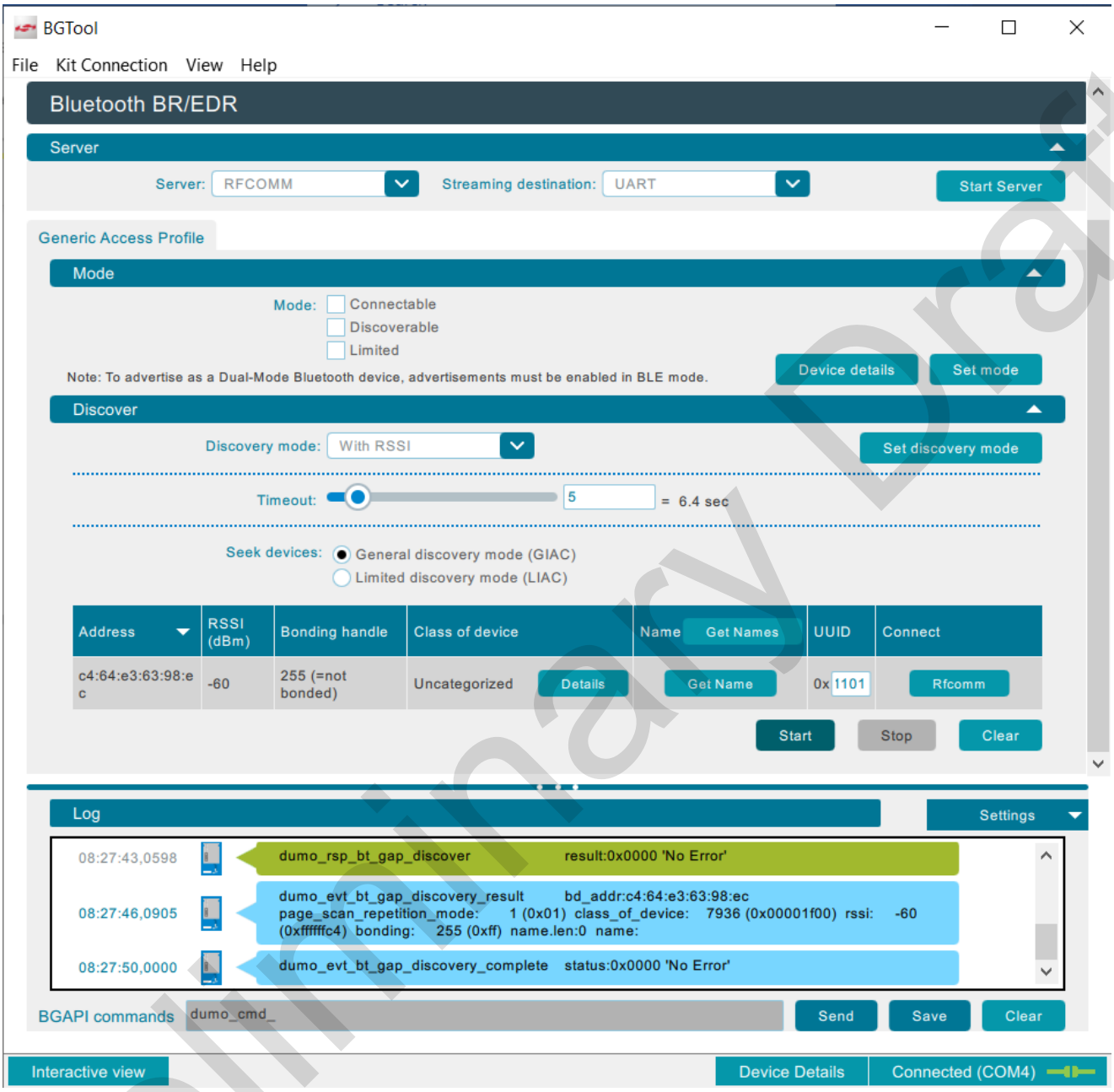

Figure 4.8 Example - Bluetooth BR/EDR Discovery

The **Security Manager** tab allows you to set and check Security details (MITM, LE Security Manager options, Bondable mode) and MAC addresses of bonded devices.

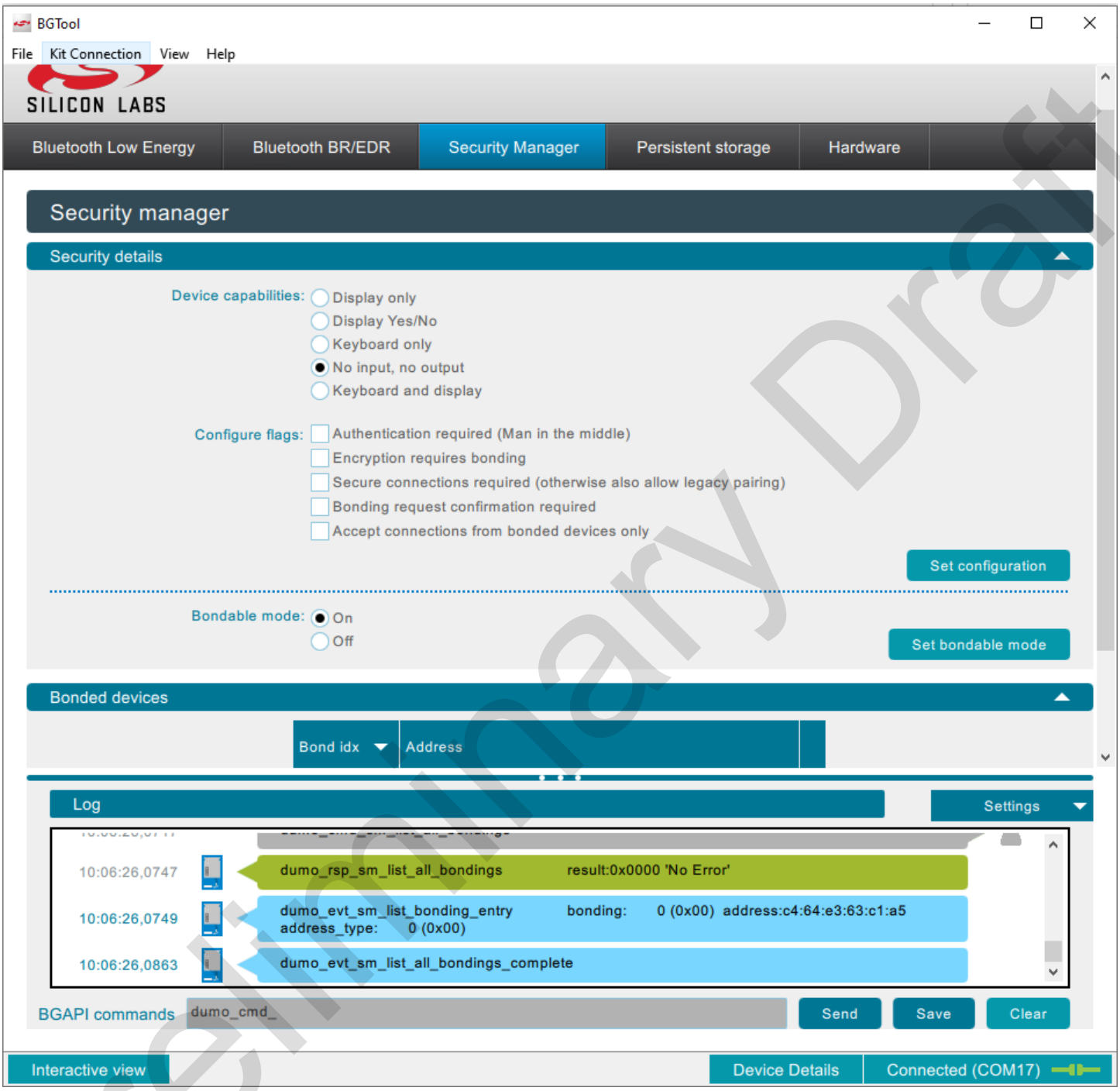

Figure 4.9 Security Manager

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

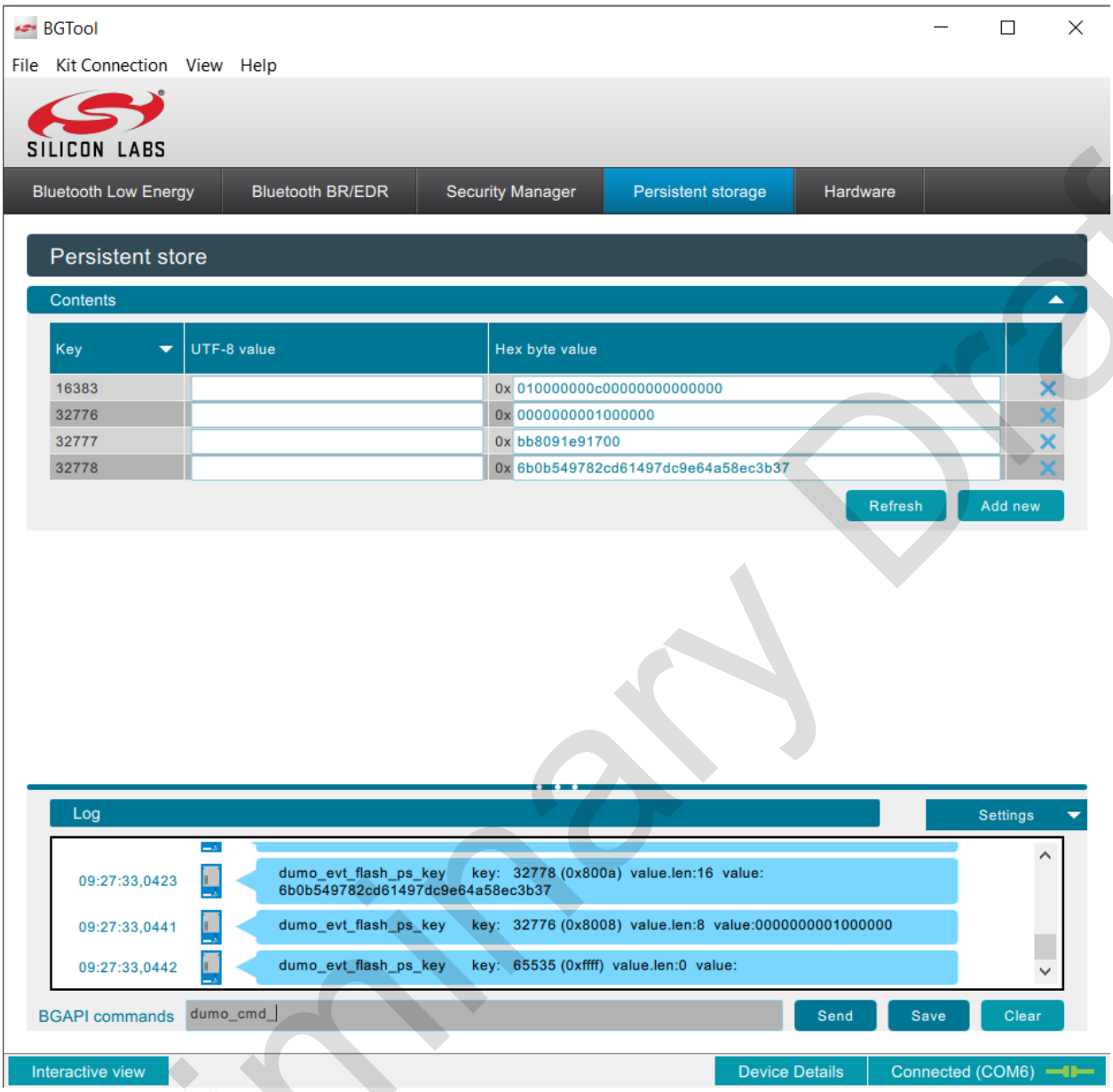

Figure 4.10 Persistent Storage Tab

4.6 RF Regulatory Test

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

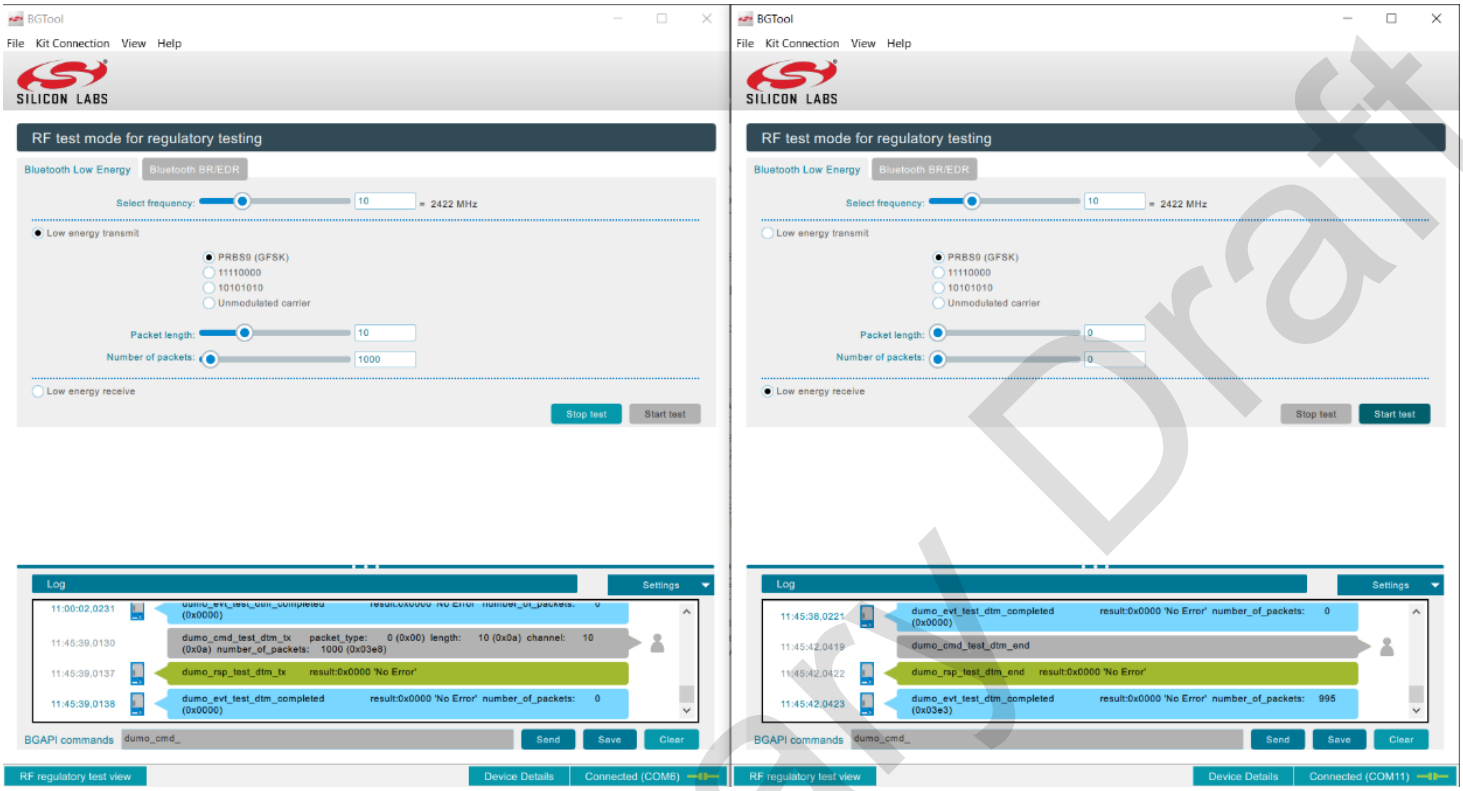

Figure 4.11 DTM test

In the **Bluetooth Low Energy** tab, BLE connection can be tested by connecting two BT122s and opening two BGTools. 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** it should be visible how many packets were received during the test.

In the **Bluetooth BR/EDR** tab one can test the transmission and reception of packets. The easiest method to test whether one device is receiving packets is to measure its current consumption (if the module is receiving, 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. The size of the packets 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 be equal when using **Single Frequency**. Data **Whitening** can be enabled or disabled. On the transmitter side, one can **Enable Rx for transmit test**, which will make the device receive packets while transmitting. The other device will only receive packets.

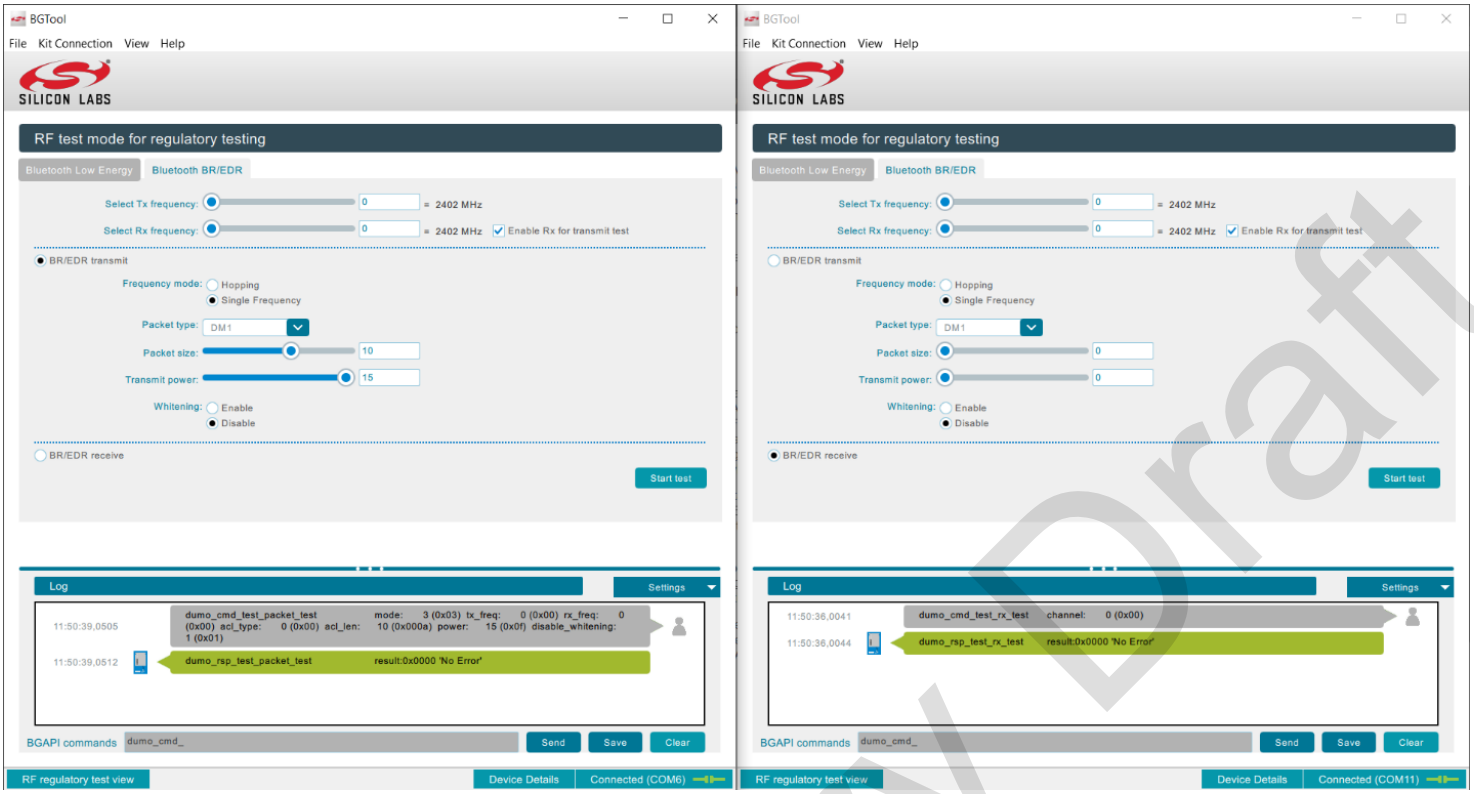

Figure 4.12 Bluetooth BR/EDR Packet Test

5 Quick Example Guide

To upload an example into the module, follow the instructions in section 5.2, where it is shown how to browse for desired project.xml file. For instance, to upload the bridge example, select example/bridge/project.xml file, build, and upload it afterward.

5.1 BRIDGE

This example "converts" transparently Bluetooth Classic SPP (Serial Port Profile) to Bluetooth Low Energy Serial. When a device sends arbitrary 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 with each other.

Go to Bluetooth & other devices settings on your PC and turn the Bluetooth on. Next, click Add Bluetooth or other device. An 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.

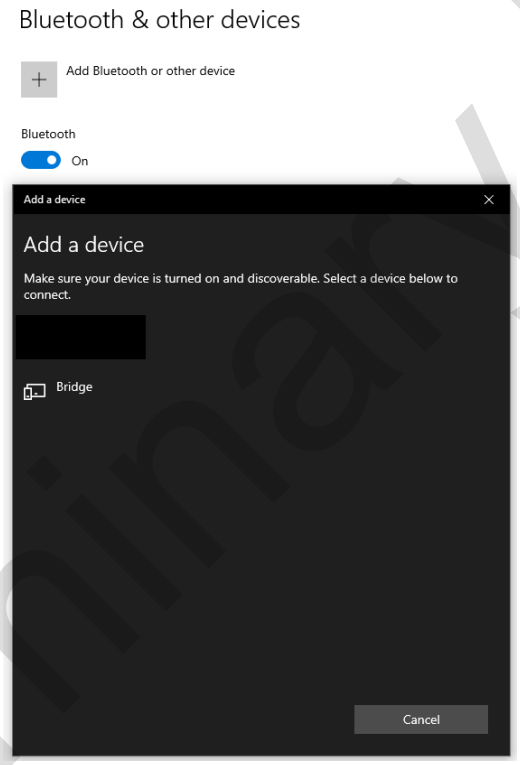

Figure 5.1 Adding a Bluetooth Device

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

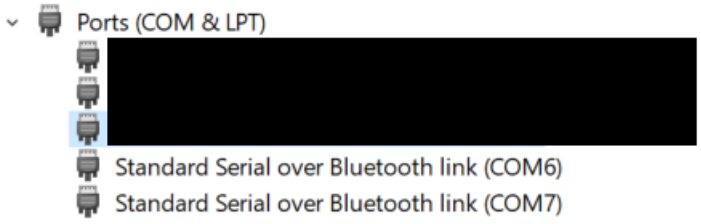

Figure 5.2 Bluetooth Serial Ports

Next, open three instances of a serial port terminal application (such as PuTTY, RealTerm, etc.) and connect to the two BT122 already programmed BT121 modules with 115200 baud rate. The third terminal should be connected to the first listed Bluetooth link in order (for example, COM6 in Figure 5.3). Restart both the BRIDGE and LE_CABLE_REPLACEMENT_CLIENT.

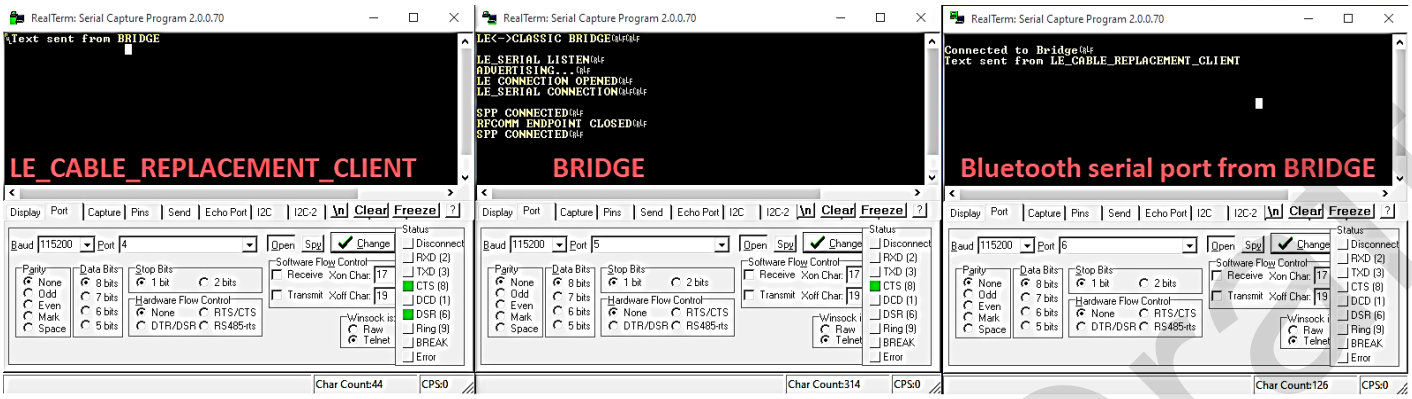

Figure 5.3 Sending Text with LE Serial and SPP

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

5.2 BT122

This is an empty example without any BGScript code, and it configures the BGAPI to be available over the UART interface. This is typically used for controlling the module using the BGTool. The hardware.xml file can be modified to test different peripherals and features, such as sleep modes, GPIO, I2C, or ADC. One can consider this example as the starting point of developing your own application, as it is an unmodified and default project. Guidelines on how to include the necessary resources in the project and how to configure the hardware interface settings for the modules are described in 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, and PF3 pin (BTN0) is configured as 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 5.4 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 BTN1 is pressed, data will be equal to 0x0000.

Using the dumo_cmd_hardware_write_gpio(2, 0x0004, 0x0004) command, LED0 can be turned off. With the **`dumo_cmd_hardware_write_gpio(2, 0x0004, 0x0000)`** command it can be turned on again.

5.3 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 – 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 the DFU mode.

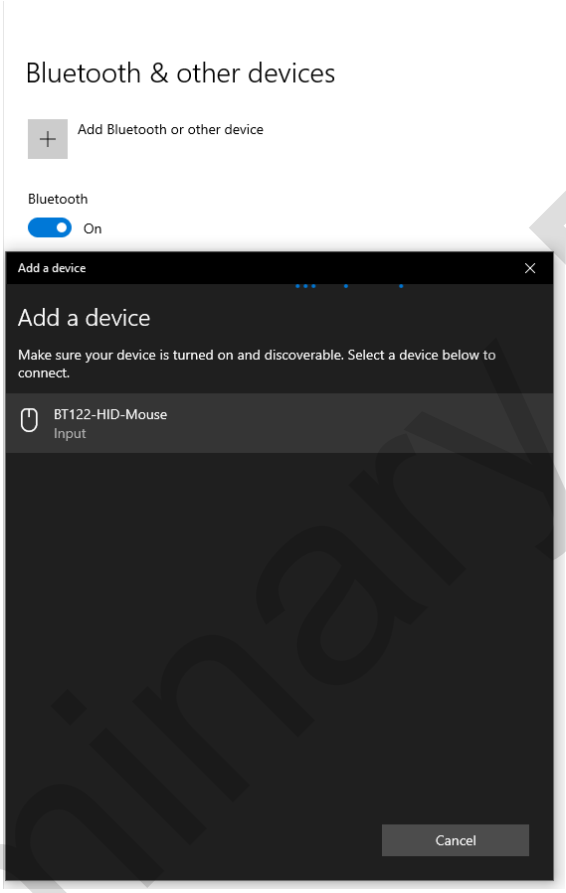

Figure 5.5 Adding HID Device

5.4 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, and HID reports can be sent via the BGTool and appropriate commands. This example is very similar to the BT122 example, but with a pre-configured HID.

5.5 LE_CABLE_REPLACEMENT_CLIENT/SERVER

These examples should be used 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 with each other. Open two serial port terminals (such as PuTTY, RealTerm, etc). Connect with both devices with 115200 baud rate. Now, when type any text in one terminal, the same text will appear in the other terminal.

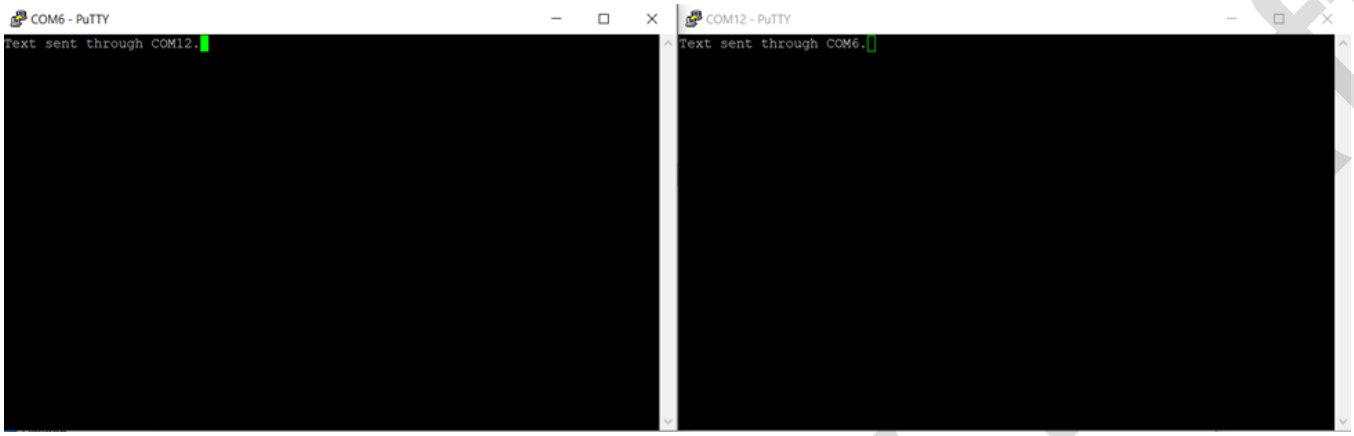

Figure 5.6 Terminals Connected to Appropriate COM Ports

5.6 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 some application which allows searching BLE devices (for example in PC can be used Bluetooth LE Explorer) and try to search device (should be visible as BT122 Si7021 Sensor) like in Figure 5.7.

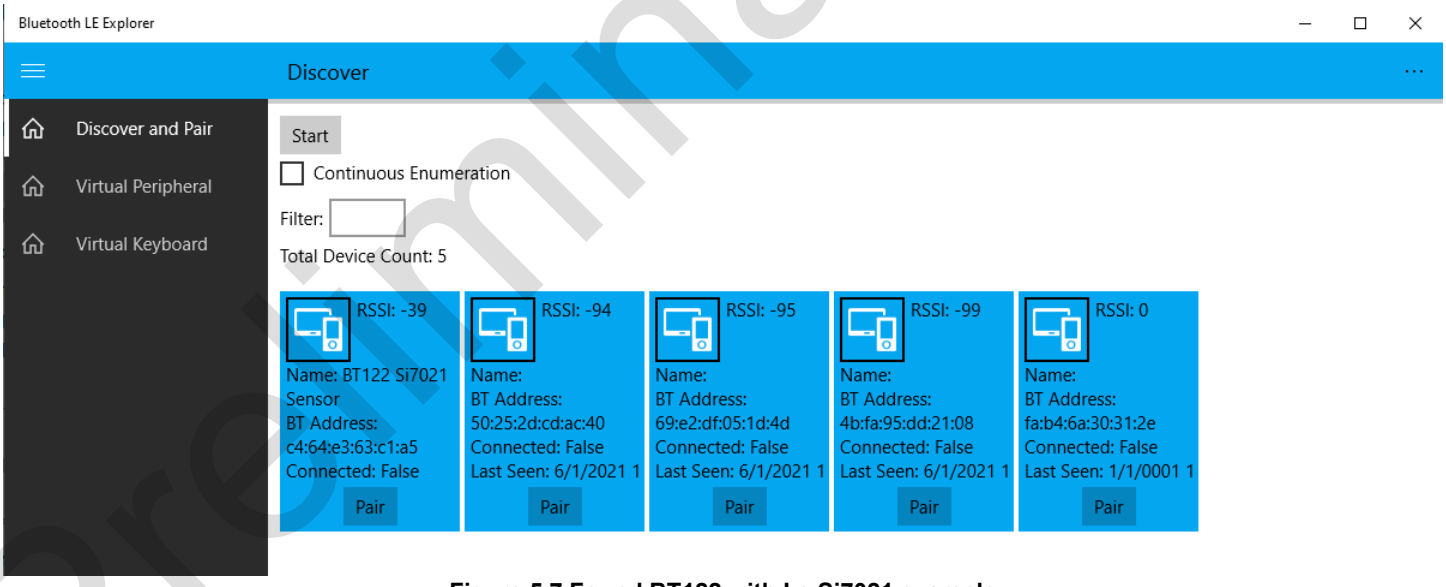

Figure 5.7 Found BT122 with Le Si7021 example

Next try to establish connection with modules. If devices are correctly connected all implemented services and characteristics should be visible for user like in Figure 5.8.

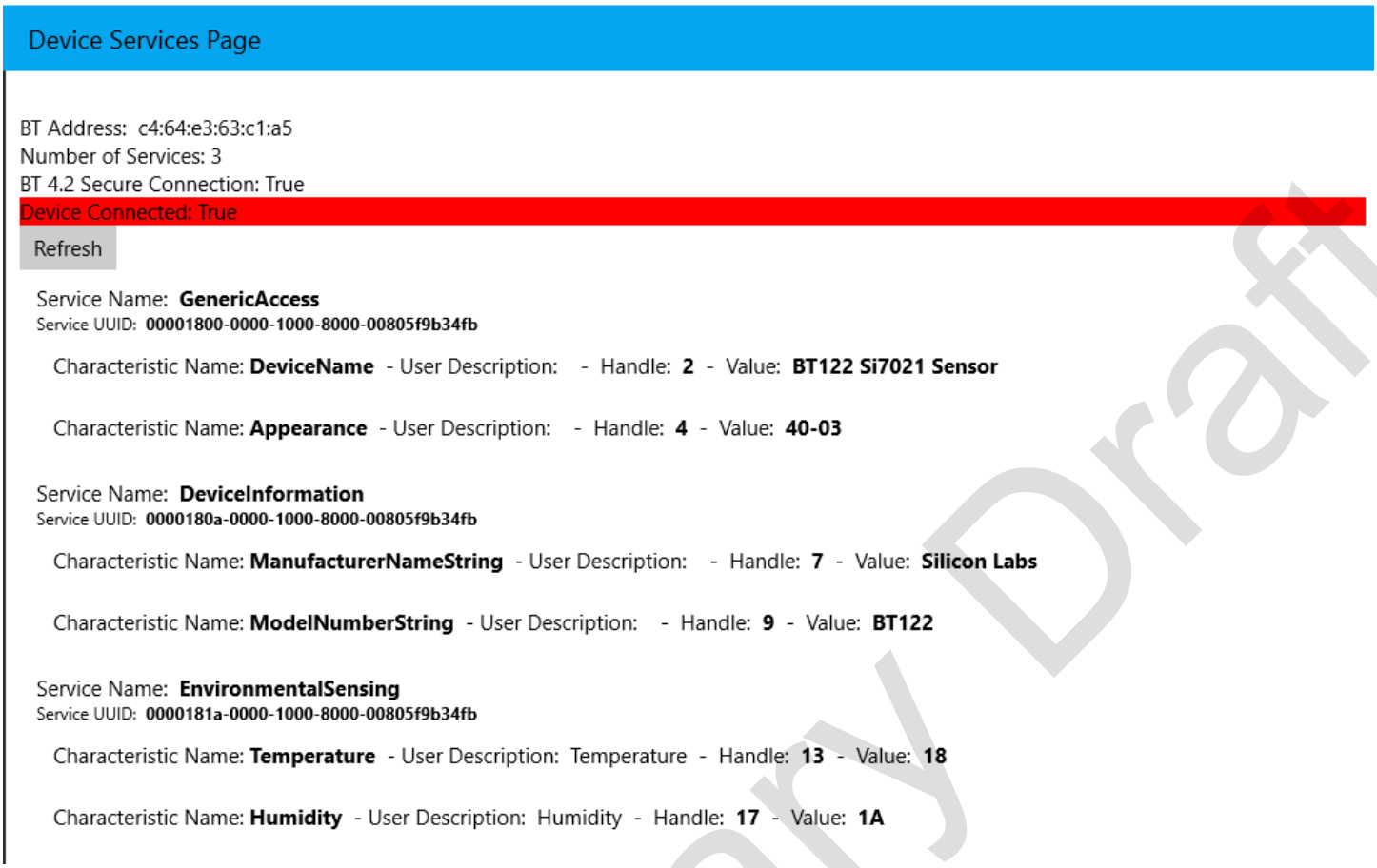

Figure 5.8 Services and characteristics of example visible in remote device

Next characteristics from EnviromentalSensing service can be notified to instantly read data if its value will be changed. Also, value can be read manually by read options. The temperature is measured in ˚C and the relative humidity in %. More details regarding the Si7021 sensor can be found in the product datasheet.

5.7 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 (Master) tab. When “BT122 Thermometer” is found, Stop the discovery procedure, and Connect to the device.

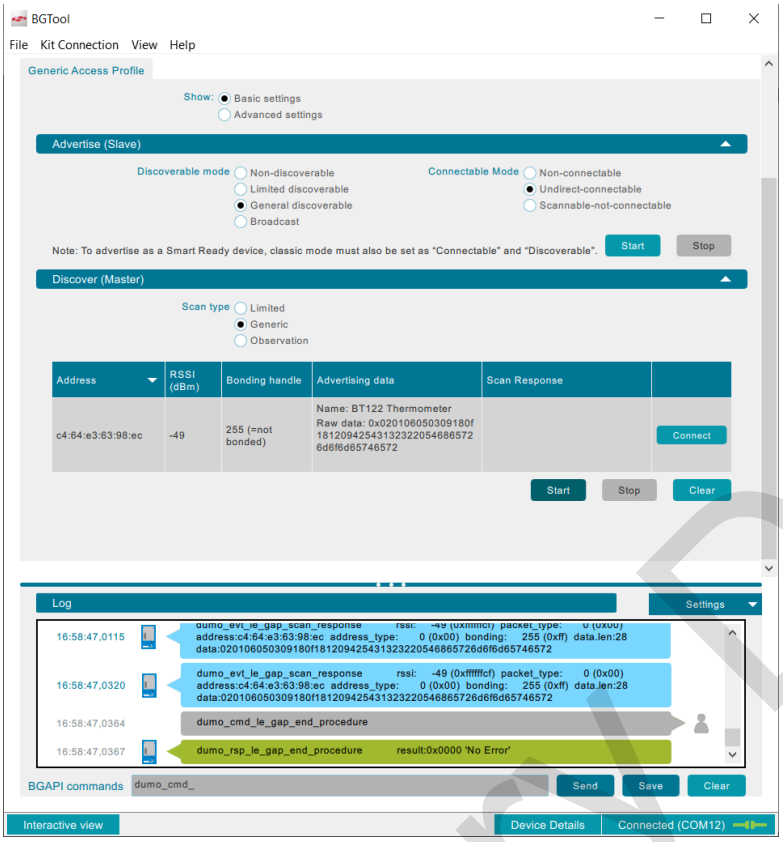

Figure 5.9 BT122 Thermometer is Discovered

Once 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.

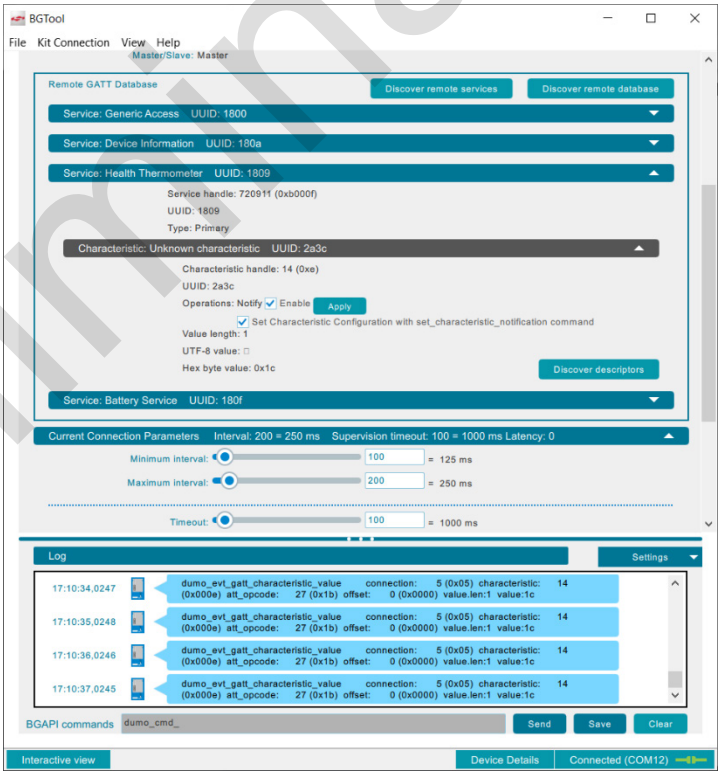

Figure 5.10 Health Thermometer Characteristics Updating with Notifications

5.8 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 power over the device, but the BGScript will still respond to events which are handled in the BGScript code.

The device can operate as a master (the device will search for a RFCOMM server with a given address) or as a slave (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 master and holds the MAC address of the device which has RFCOMM server configured. You can find more details in the *uart_modes.bgs* file.

If the device is configured to be a **master** (script_version = 1), set the MAC address of a slave 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) and upload this 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 and 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**. Configuration should look like this:

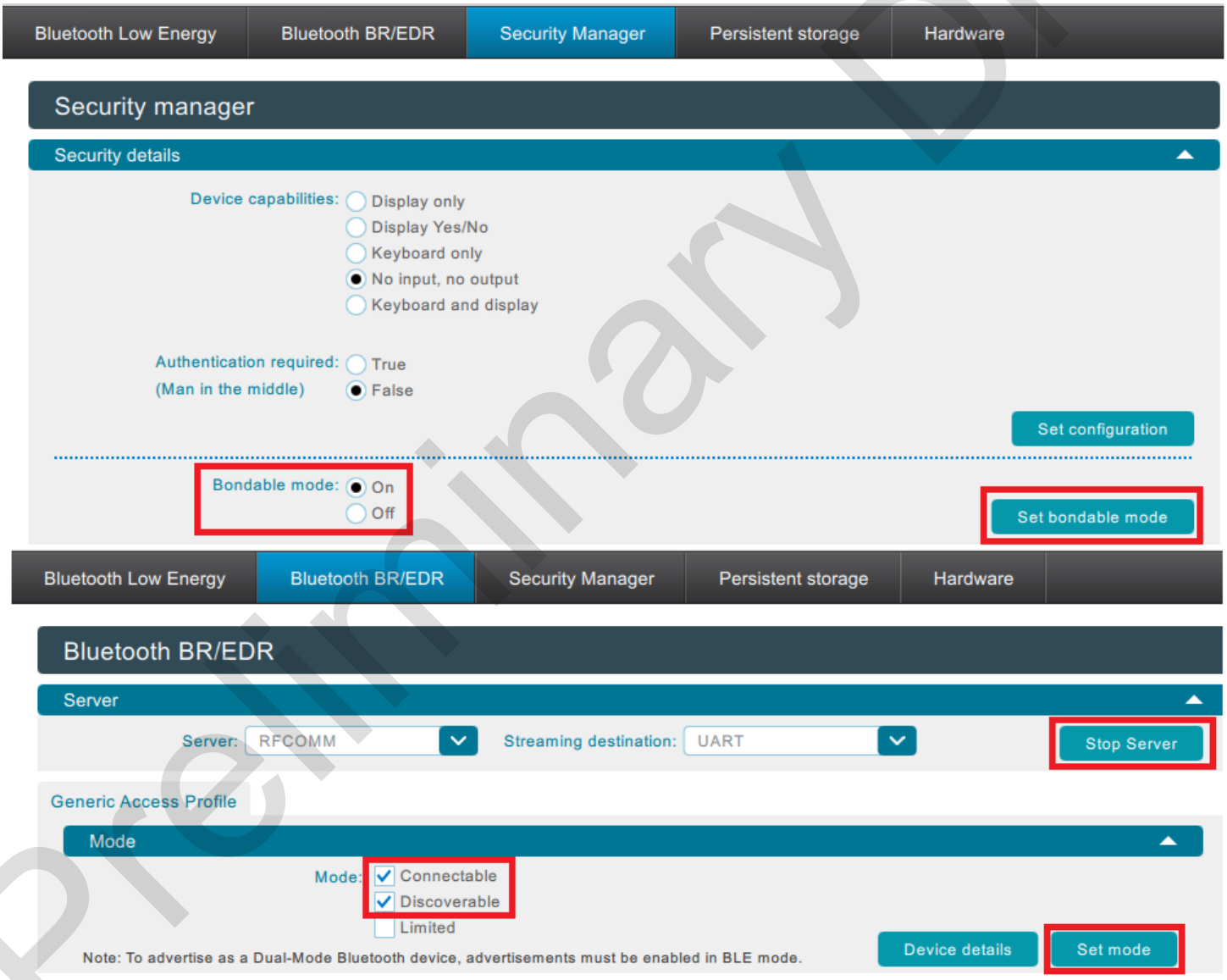

Figure 5.11 BT122 as a Slave Configuration

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

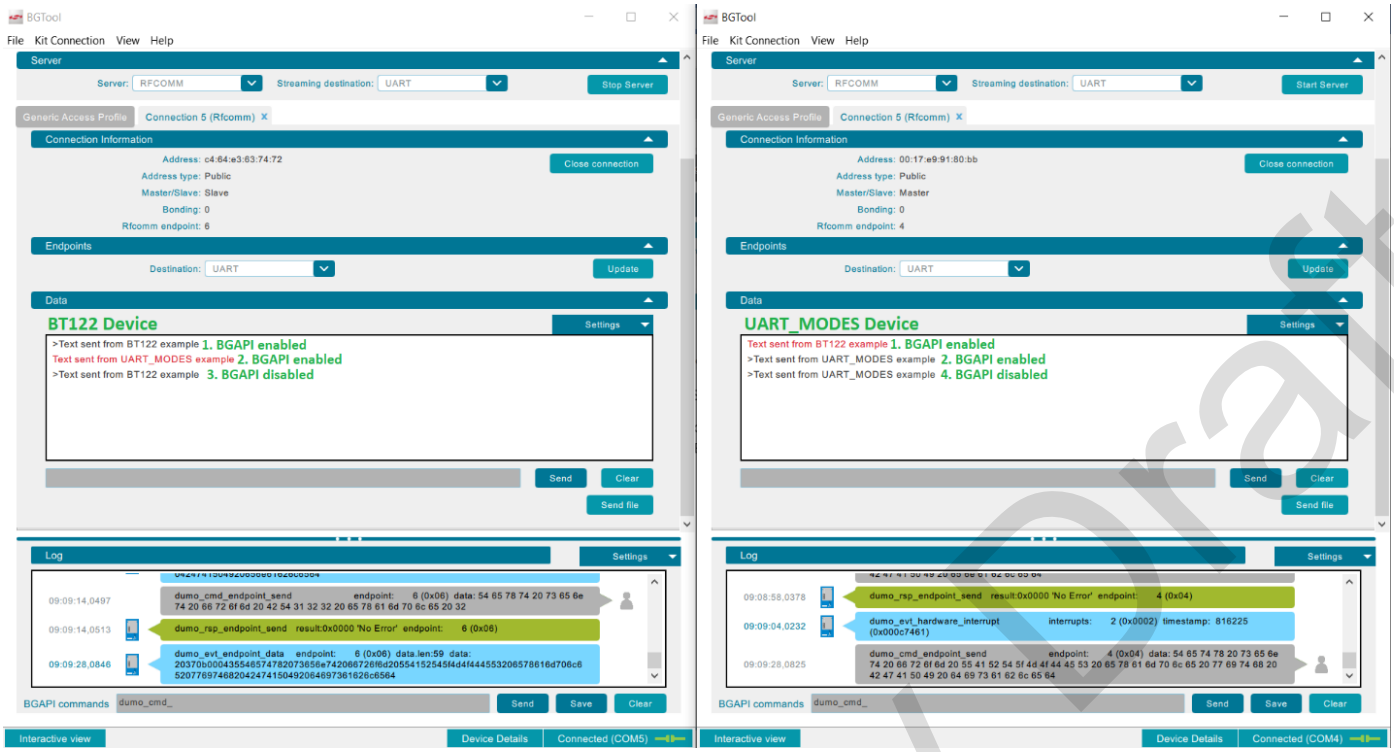

Figure 5.12 BT122 Slave (Left) and UART_Modes Master (Right)

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 **slave** (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 is like the one in Figure 5.12.

Figure 5.13 Connecting to the UART_MODES Slave

Smart. Connected. Energy-Friendly.

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 Laboratories:

BT122-DK4315A