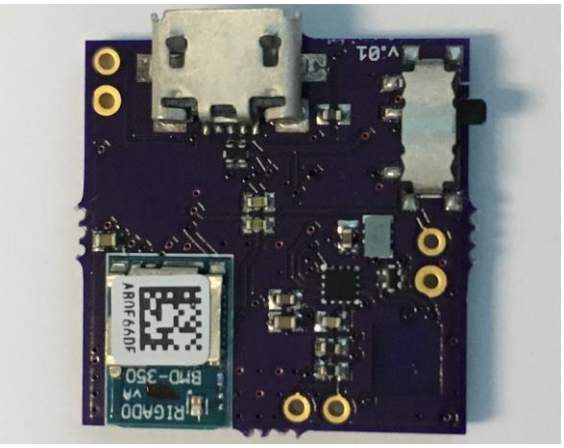
Nordic UART Service (NUS) BLE Data Link and Over-the-Air (OTA) Firmware Update Utility

By Greg Tomasch

This document is intended to provide the background information necessary incorporate Bluetooth Low Energy (BLE) connectivity to the STM32L4 family of microcontroller units (MCU's). This is done by connecting one of the STM32L4's UART ports to an nRF52-based BLE NUS peripheral running in UART pass-through mode, in this case a Rigado BMD-350. A test utility for updating the STM32L4's firmware over-the-air using the same BLE connectivity will also be described.

Teaming up the STM32L4 and the BMD-350

In this case the BMD-350 has been built into a custom board based upon the STM32L433CC MCU and is connected to the UART1 port but it could just as easily be a BMD-350 module connected to any of the Pesky Products STM32L4 development boards using any available UART.



The BMD-350 is addressed and configured using Rigado's native firmware ("BMDware") using the "AT" command interface. (https://www.rigado.com/download/bmdware-datasheet-ibeacon-uart-bridge-firmware/) The BMD-350 "#RST" pin is connected to Arduino pin "1" of the STM32L4 while the "AT_MODE" pin is connected to Arduino pin "26". However, any selection of these pins can be configured as desired in the example sketch in the "Config.h" tab. All of the necessary steps to configure the BMD-350 and put it into UART pass-through mode are executed in "BLE.cpp" and "BLE.h" modules included in the "STM32_BMD350…" Arduino example sketch. Once configured, updating a paired BLE Central Role device with results is done by simply writing data to UART1 of the STM32L4 using "`Serial1.print()`" or "`Serial1.write()`" It should be noted that hardware flow control is not implemented so

care should be taken not to push too much data to the BMD-350 at once or data loss can result. Typically, a message of 40bytes or less can be pushed to the BMD-350 with satisfactory results.

BLE OTA Firmware Updates for the IU STM32L4 Board

The infrastructure for BLE OTA firmware updates is implemented in the "BLE_OTA.cpp", "BLE_OTA.h", "Host_WirelessSerial.cpp" and "Host_WirelessSerial.h" modules included in the "STM32_BMD350…" example sketch. The "Host_WirelessSerial" modules comprise the serial protocol used to receive the OTA firmware update request, receive the OTA firmware data block messages and transmit ACK/NAK messages back to the host firmware update utility. The "BLE_OTA" modules constitute a sub-loop that manages:

- Testing of the available flash memory on the STM32L4 with respect to the desired new firmware image size
- Erasing flash memory for the new firmware image
- ACK'ing the update request from the host firmware update utility
- ACK'ing each data block message and writing the firmware image data payload to the target STM32L4's flash memory
- Resetting the STM32L4 once the new firmware image is complete

If the "BLE_OTA" sub-loop ACK's the OTA update request but does not receive the first data block message after multiple tries, the OTA firmware transaction will be NAK'd and the target STM32L4 will resume operation on the original firmware without harm. Similarly, if the data block count indicates a skipped block, the OTA firmware transaction will be NAK'd as well and normal operation will resume. In short, the only way the STM32L4 host MCU will reset and attempt to use the new firmware image is:

1. There is enough free flash memory to accept the new firmware image without encroaching on the existing firmware image
2. Each data block message passes the message preamble, valid decode command, data payload size and bitwise XOR checksum tests included in the serial protocol. If not, the data message will be discarded and the target STM32L4 will request re-send of the current data block message
3. All data block serial numbers are received without gaps and add up to the total required for the complete firmware image

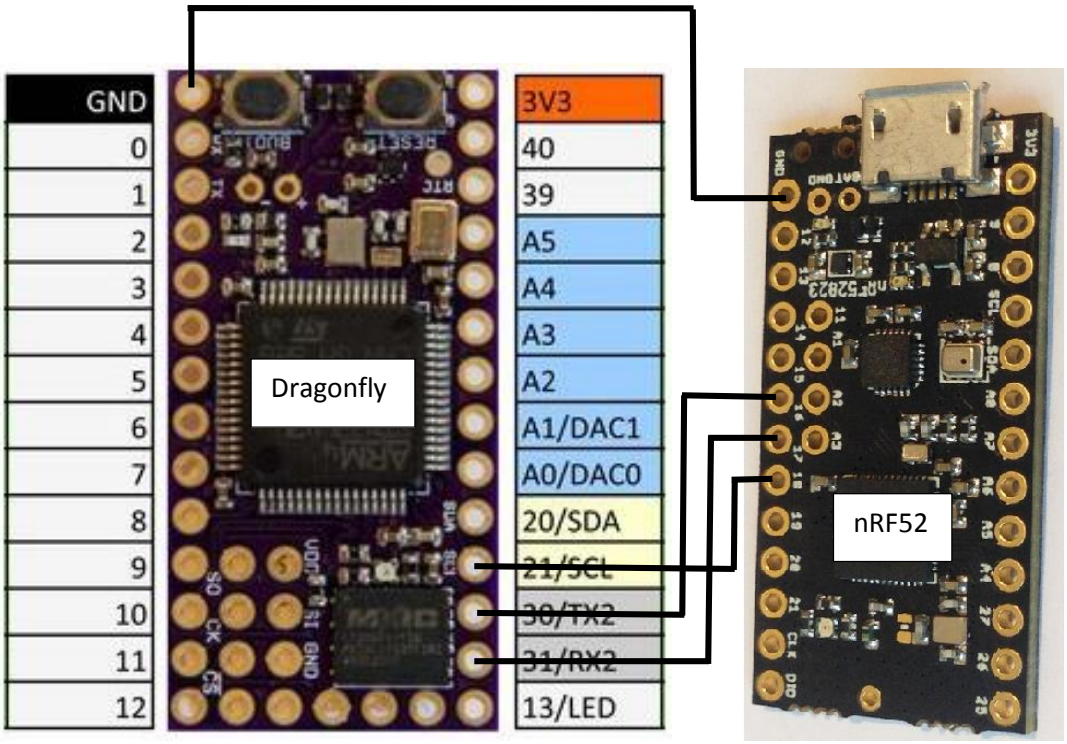
BLE Data Monitor and OTA Firmware Update Host Utility

The host utility actually performs two roles: 1) It receives output data updates from the target STM32L4 MCU over the BLE/UART bridge and passes it through to the host utility's USB serial port and 2) Acts as the Central Role BLE host and sends OTA firmware updates to the target STM32L4 MCU.

For the second role, there are several key functions that the BLE OTA firmware update host must perform:

A. It must be able to locate, open, read and parse the ".iap" firmware image file for the desired STM32L4 firmware update
B. It must serve as a BLE NUS Central Role device that pairs with the BMD-350 NUS Peripheral Role device on the IU board. It must also be able to exchange information with the BMD-350 while it is in "UART pass-through" mode
C. It must be able to encode parsed portions of the ".iap" firmware image file into message packets that conform to the serial protocol
D. It must be able to decode and process ACK/NAK messages packetized by the serial protocol sent from the target STM32L4 MCU and take appropriate action on them to manage firmware image data integrity and flow over the BLE/UART bridge

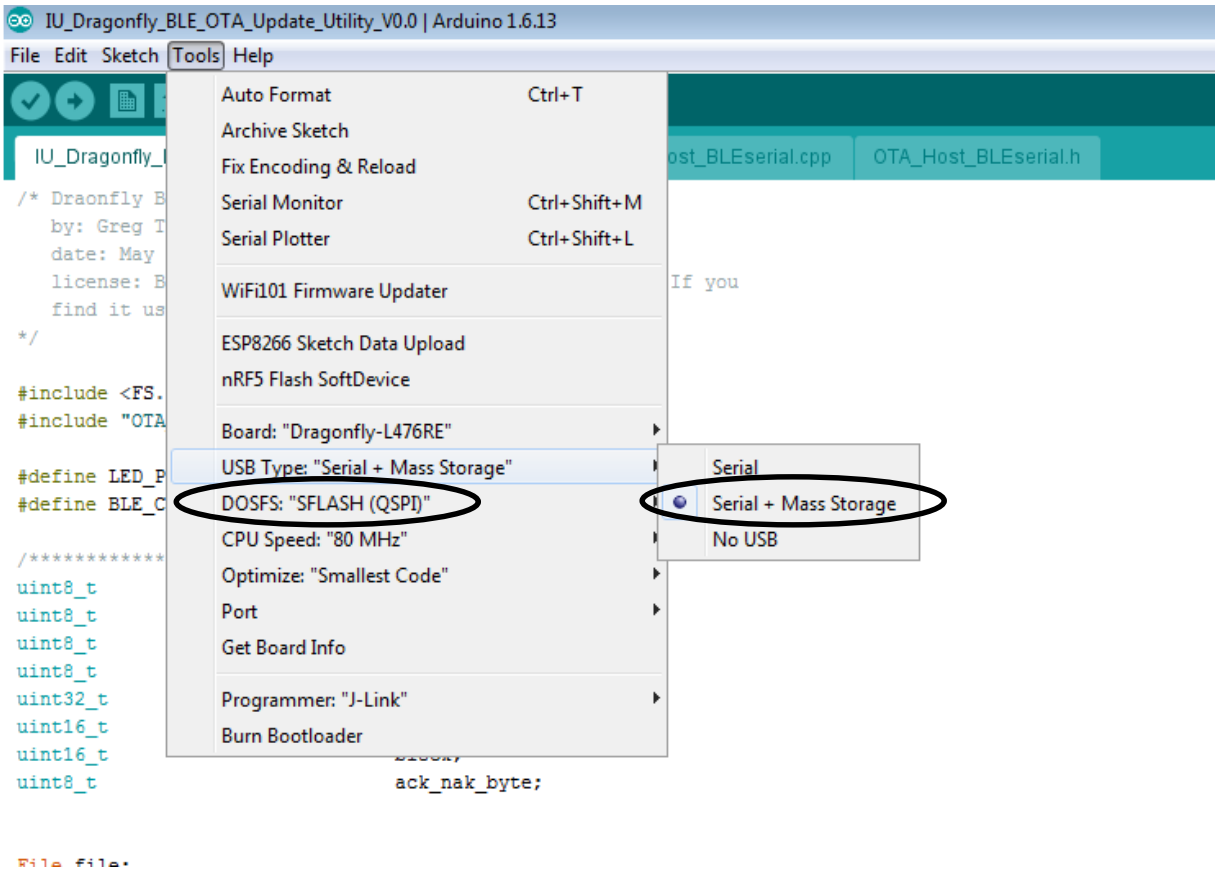
Typically, a smartphone or tablet device running a custom application would be used to fulfill this role. Another option for multiple similar peripheral devices would be to use a NUS-capable BLE hub and develop an application to parse the ".iap" firmware image file, packetize the data block messages and route them to the correct peripheral device. However, for purposes of development and testing we have utilized two MCU development boards together to demonstrate OTA firmware update to a single IU peripheral board paired to a single NUS Central Role device. The two boards are the Pesky Products STM32L4 "Dragonfly" development board (https://www.tindie.com/products/TleraCorp/dragonfly-stm32l476-development-board/?pt=ac_prod_search) and the Pesky Products nRF52 development board (https://www.tindie.com/products/onehorse/nrf52832-development-board/). The Dragonfly uses the powerful STM32L476RE version of the STM32 MCU complete with many peripheral options. This board is also equipped with a 128 Mbit QSPI flash chip that supports a drag-and-drop Windows virtual disk drive. Using this powerful package, it is possible to make new firmware files available to the MCU and read/parse them with ease. The nRF52 development board was selected because it can be programmed to act as a BLE NUS Central Role device to form the BLE/UART bridge with the BMD-350 peripheral on the STM32L4 target board.



The Development boards are connected by a total of four leads:

Dragonfly Board Node	nRF52 Board Node
GND	GND
D30 (TX2)	16 (RX)
D31 (RX2)	17 (TX)
D21	18

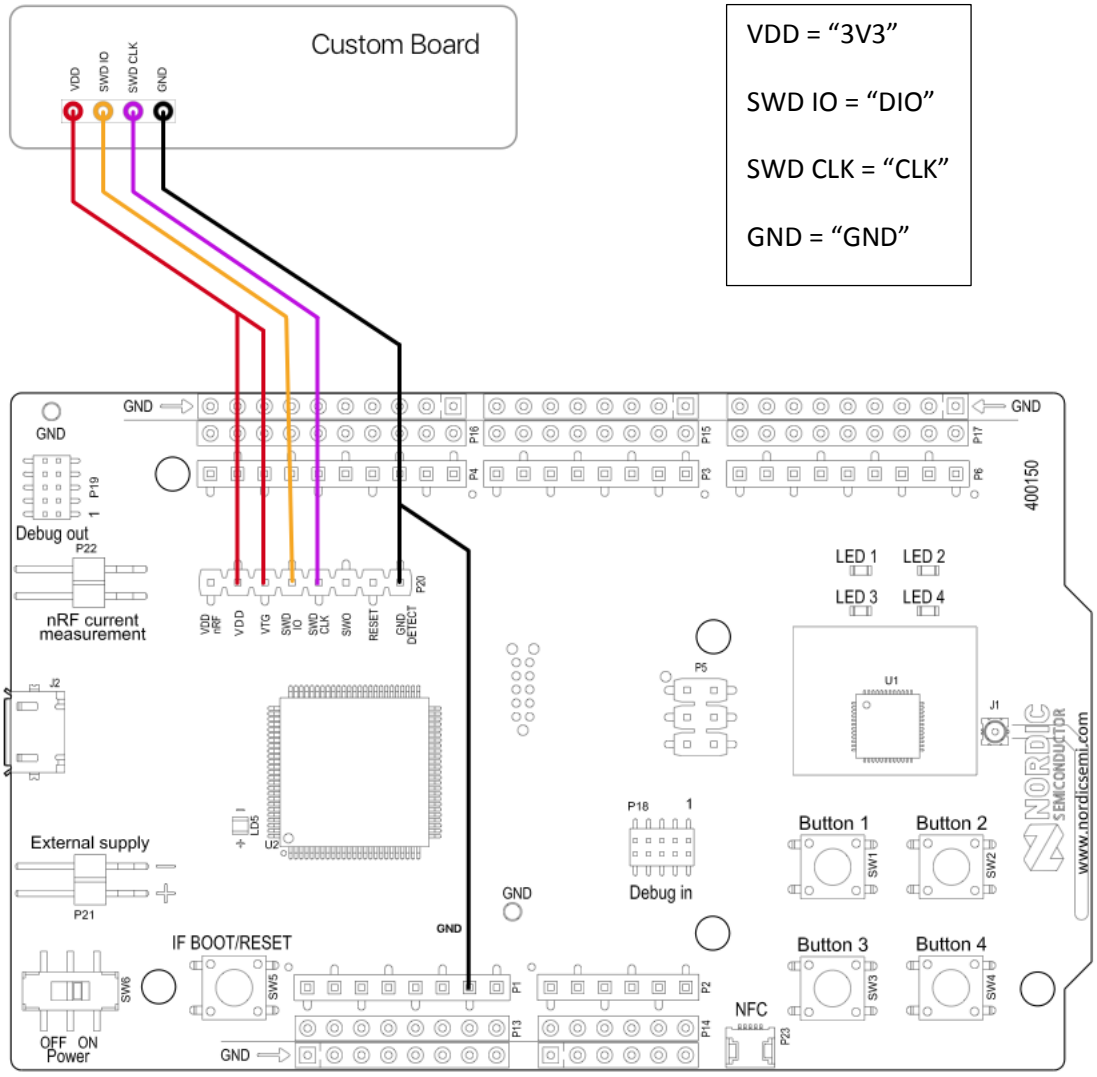
The grounds of the two boards are connected to provide the same signal reference, Dragonfly pins D30 and D31 are connected to nRF52 pins 16 and 17 in null modem configuration and Dragonfly pin D21 is connected to nRF52 pin 18. When the nRF52 development board pairs with the BMD-350, nRF52 pin 18 is pulled logic low. Dragonfly pin D21 is configured as an input so the Dragonfly can sense when the BLE/UART bridge is functional. The Dragonfly is programmed with the "STM32L4_nRF52dev_BLE_OTA_Update_Monitor_Utility_V0.0.ino" Arduino sketch using Thomas Roell's STM32L4 Arduino core and the Arduino IDE (https://github.com/GrumpyOldPizza/arduino-STM32L4). The nRF52 is programmed with the Nordic "s132_nrf5_3.0.0_softdevice.hex" and "nrf52832.hex" executables using the Nordic pca10040 DK board and the nRFgo Studio application. (https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52-DK) Programming the Dragonfly board is straightforward using the Arduino IDE. However, it should be noted that the proper compile options should be selected:



Under “USB Type:” the “Serial + Mass Storage” option should be selected. Similarly, under “DOSFS:” the “SFLASH(QSPI)” selection is correct.

Programming the nRF52 to Perform the BLE Central Role

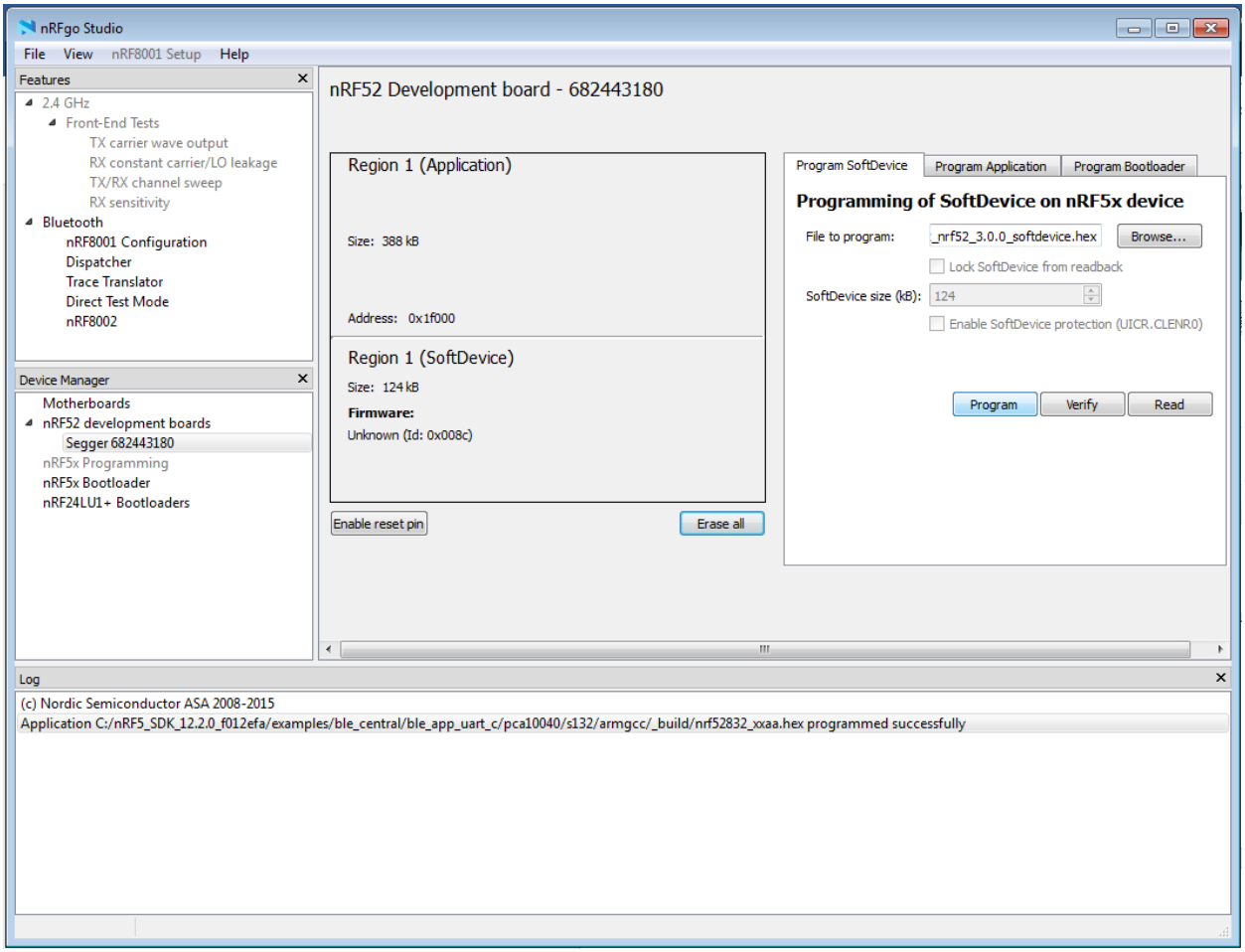
Programming of the nRF52 is a bit more complicated. The basic BLE NUS Central Role programming is an example in the nRF_SDK_12.2.0 software development kit package. This example has been slightly modified to route the UART RX and TX pins to pins 16 and 17 (respectively) on the nRF52 development board and to make pin 18 toggle low when the board pairs with the BMD-350 on the IU target board. In order to program the compiled code onto the nRF52 development board, a Nordic nRF52 pca10040 DK board is useful. Disconnect from the Dragonfly and connect the nRF52 development (“Custom”) board to the DK board:



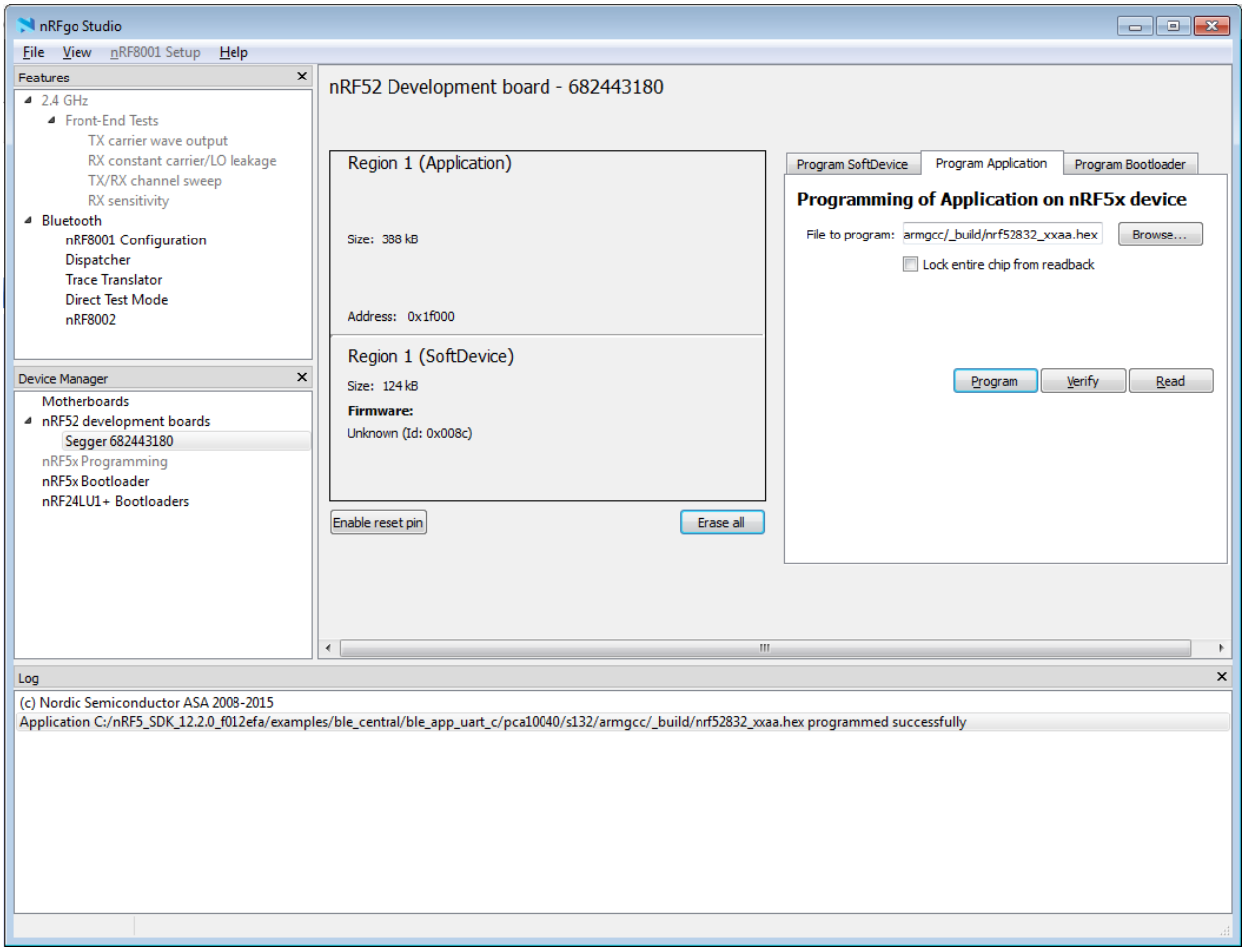
Where the pin designations in quotes refer to those stenciled on the nRF52 development board.

Power up the nRF52 development board with a micro USB cable; this board does not have a serial adapter so you will not see a serial port appear for it. Next, plug the nRF52-DK board into an active USB port and turn it on… Let the various drivers load; when finished, a virtual drive window will open for the Segger J-Link programmer.

It is assumed that the user went to the “Downloads” tab of the nRF52-DK link listed above and installed the appropriate version of nRFgo Studio. Open the application and plug in the nRF52-DK board with the nRF52 development board attached as shown above.



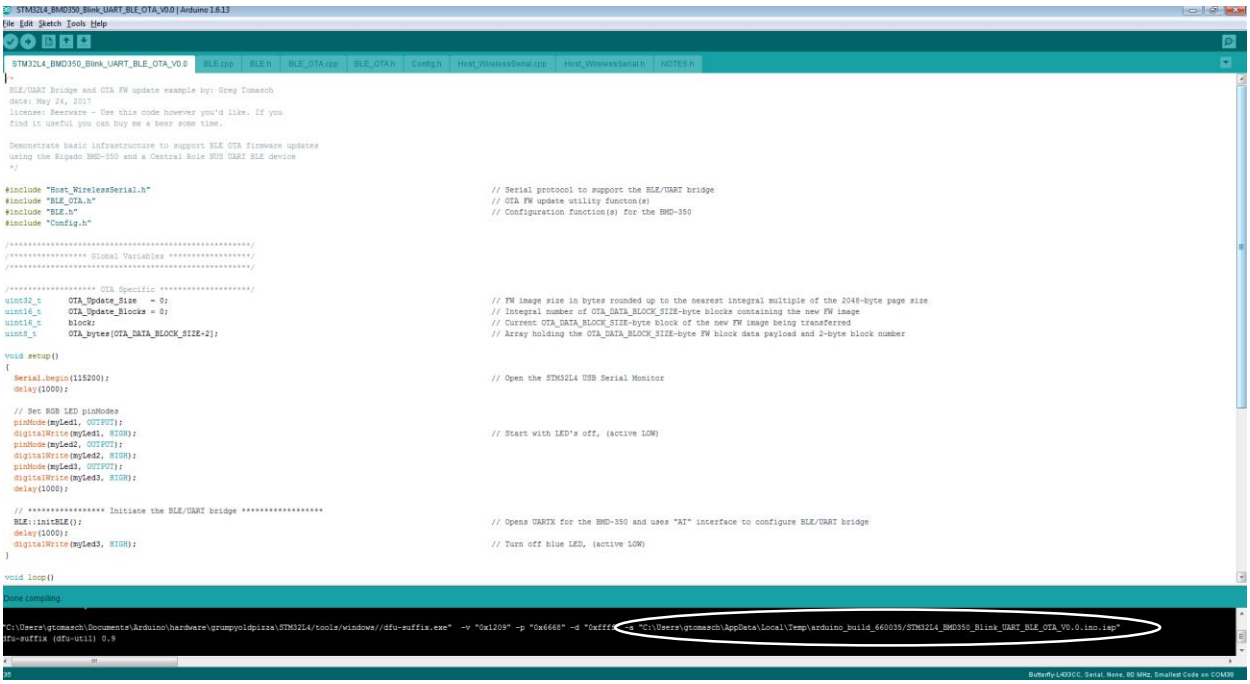
Select the “Segger XXXXXXXXX” entry under nRF52 development boards and click on the “Program SoftDevice” tab. You should see screen shown above… Click the “Browse…” button and navigate to the “s132_nrf5_3.0.0_softdevice.hex” executable file. Click the “Program” button and nRFgo Studio will flash the SoftDevice to the nRF52 development board. Next, click on the “Program Application” tab, click the “Browse…” button and navigate to the “nrf52832.hex” executable file. You should see:



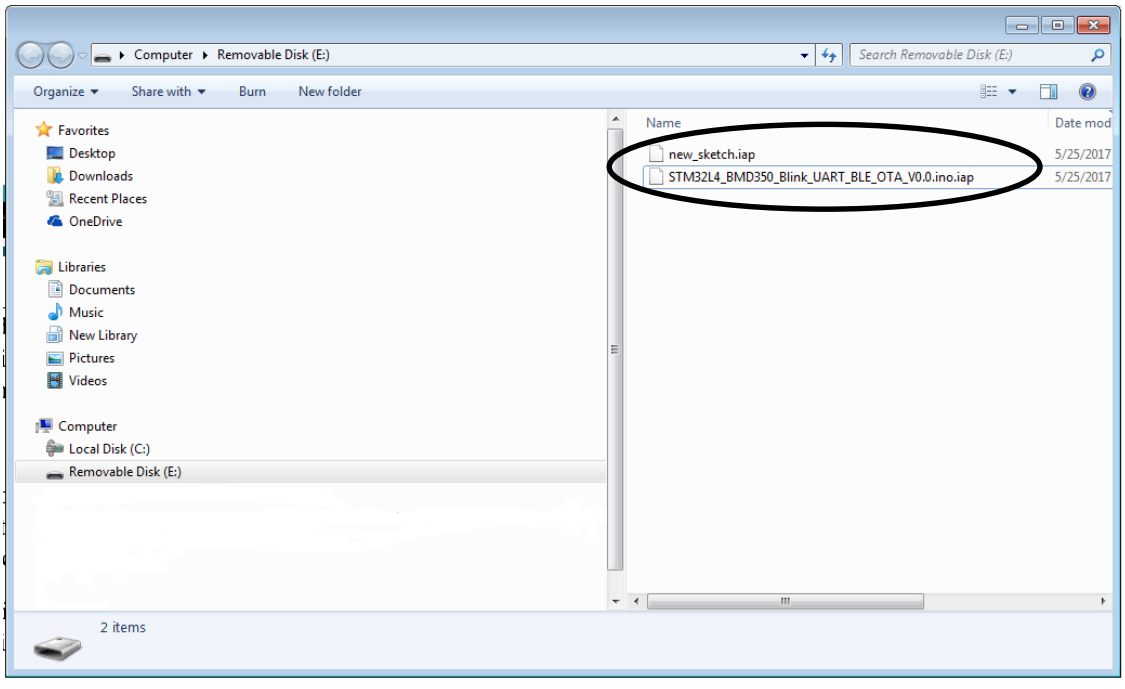
Click the “Program” button again to flash the BLE Central Role UART application to the nRF52 development board. Now the nRF52 development board is properly flashed; both boards can be powered down and the nRF52 development board may be disconnected from the nRF52-DK board. Re-connect the Dragonfly and nRF52 boards as previously described. Now the pair of boards constitutes the BLE data monitoring and OTA firmware update utility…

Using the OTA Firmware Update Utility to Perform an OTA Firmware Update

Compile and upload the “STM32L4_BMD350_Blink_UART_BLE_OTA_V0.0” sketch to the IU board using the Arduino IDE… It should send data over both the IU board’s USB serial and over the BMD-350 UART pass-through. Next, go to the Arduino IDE instance just used to compile the IU board’s firmware. Go to the message window at the bottom of the screen and scroll both up and right until you locate the path message as to where the “.iap” firmware image file is located:



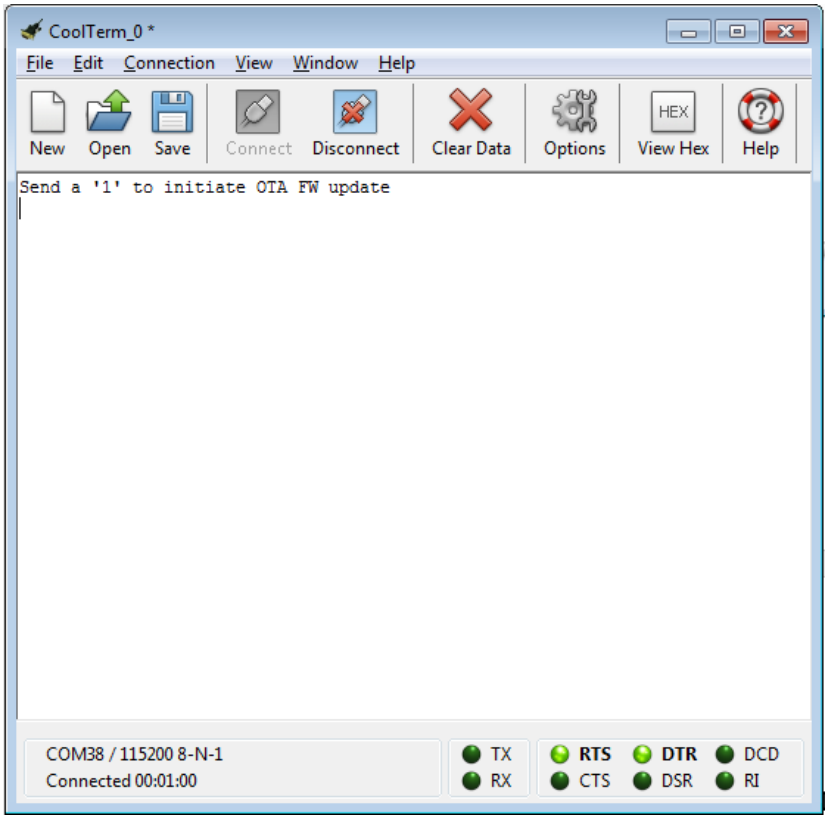
In this case, the file is “IU_Butterfly_BMX055_BMP280_UART_BLE_OTA_V0.0.ino.iap”. Typically, it will be located in a “C:\Users\.....\AppData\Local\Temp\arduino_build_xxxxxx” directory where “xxxxxx” is a numerical designation assigned by the IDE. This is the firmware image file… Right-click on the “.iap” file and copy it to the Dragonfly virtual drive:



Copy and rename the file to “new_sketch.iap”. This is the file that will be parsed and transferred to the IU target board during the OTA firmware update. This is necessary as the “DOSFS”

library used by the Dragonfly sketch for file manipulation does not support file name wildcarding.

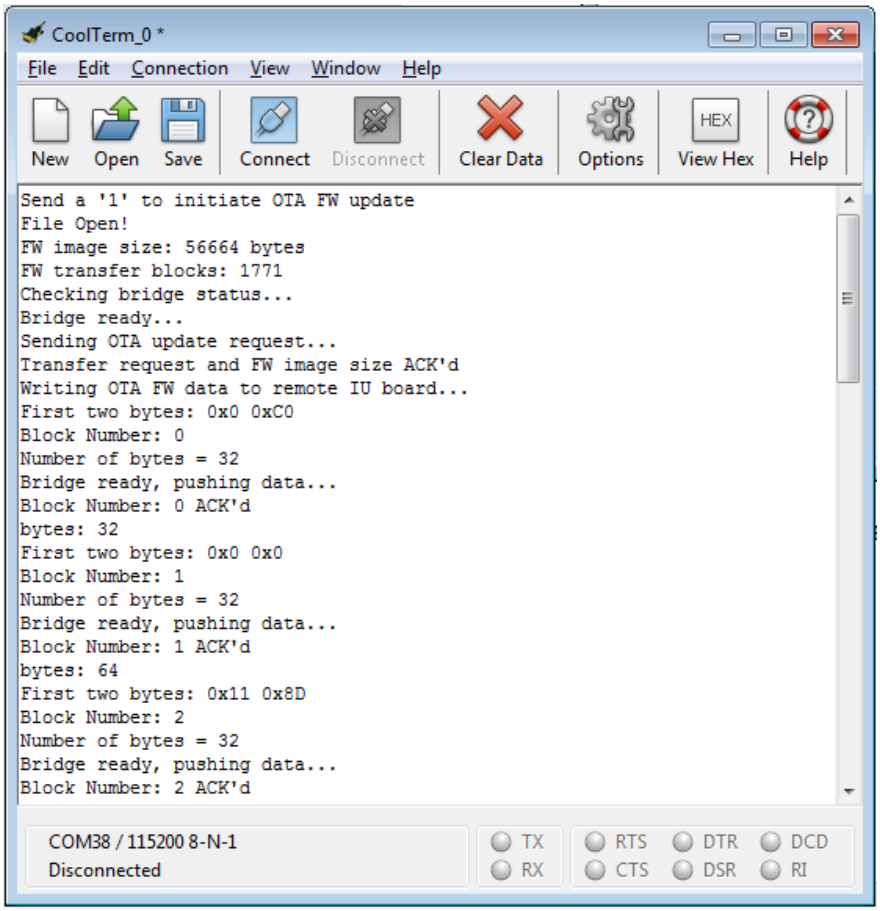
Power up all of the boards including the target IU board and make sure that there is no other Central Role device that will attempt to pair with the BMD-350. The blue LED on the nRF52 development board will be on steady when it is paired with the BMD-350 on the IU target board. Next, open a serial terminal and connect to the Dragonfly USB serial port at 115200 baud, 8-N-1 parity and no flow control. The Dragonfly OTA update utility will indicate that sending a “1” will initiate OTA firmware update:



After the initial message has been printed in the serial monitor, data updates from the STM32L4 board will also be posted in the window. Even though original message will eventually be obscured, the application will still respond to a “1” sent from the serial monitor.

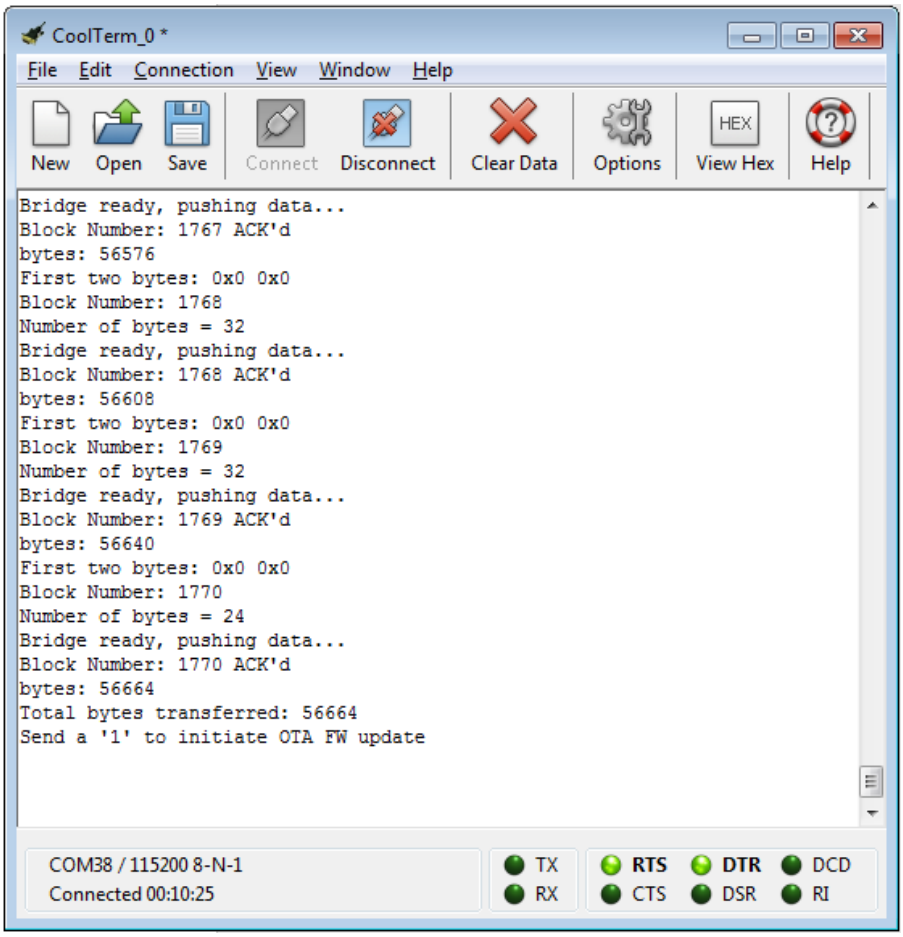
Send a “1” as requested and the firmware update will commence. The terminal messages provide a wealth of information:

- New firmware image size
- Total number of data blocks to transfer the firmware image
- BLE/UART bridge status
- Message ACK/NAK status
- First two bytes of the data block payload
- Data block payload size
- Transferred byte count



Upon successful completion, the total transferred byte count is indicated followed by a prompt to send a “1” to initiate another firmware update. Once the target STM32L4 MCU reboots, data updates will resume. At this point, another version of the target IU board firmware can be copied

to the Dragonfly virtual drive, renamed to “new_sketch.iap” and the OTA firmware update process can be repeated at any time by sending a “1” from the serial terminal window…



Once all desired firmware updates are complete, the utility will resume monitoring of the BLE UART messages coming from the target STM32L4 board.