SmartBond™ WiRa™ Wireless Ranging SDK

This document provides basic information to help developers get familiar with the DA1469x Wireless Ranging application and modify or create a new Wireless Ranging application based on it.

Contents

Figures

1. Terms and Definitions

ADC	Analog-to-Digital Converter
AGC	Automatic Gain Control
API	Application Programming Interface
CLI	Command-Line Interface
CMAC	Configurable Medium Access Controller
DK	Development Kit
DTE	Dialog Tone Exchange
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GPIO	General Purpose Input Output
IFFT	Inverse Fast Fourier Transform
IQ	In-Phase and Quadrature
ISM	Industrial, Scientific, and Medical (radio band)
LCD	Liquid-Crystal Display
LE	Low Energy
Msps	Mega samples per second
MTU	Maximum Transmission Unit
NVM	Non-Volatile Memory
PDU	Protocol Data Unit
RSSI	Receive Signal Strength Indication
RF	Radio Frequency
SDK	Software Development Kit
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
UUID	Universally Unique Identifier

2. References

[1] DA1469x, Datasheet, Renesas Electronics.

[2] UM-B-057, SmartSnippets™ Studio User Guide, User Manual, Renesas Electronics.

[3] UM-B-092, DA1469x Software Platform Reference, User Manual, Renesas Electronics.

[4] UM-B-093, DA1469x PRO Development Kit, User Manual, Renesas Electronics.

[5] UM-B-103, DA14695 USB Kit, User Manual, Renesas Electronics.

[6] UM-B-162, DA14695 SmartBond Module Development Kit, Renesas Electronics.

Note 1 References are for the latest published version, unless otherwise indicated.

3. Introduction

The estimation of range or distancing between two (or more) objects using radio waves is not new. In fact, the use of RF signal strength measurement, RSSI (Receive Signal Strength Indication) has been in use for decades. While in a noise-free, "free space" environment with clear line-of-sight this technique can yield quite accurate results only consuming nominal power; in more typical environments, such as indoors and especially when a product is being worn or moved around, multipath (reflected version of the "direct path" signal, bouncing off walls/floors/and so on) as well as attenuation of the direct path signal (blocked by hands, arms, bodies, furniture, and so on) presents a real challenge to RSSI, which intrinsically relies on the RF signal's amplitude to estimate distance.

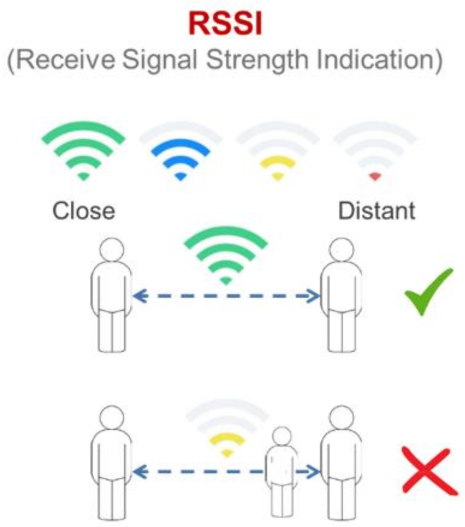


Figure 1. Use of RSSI in a noise-free environment

An alternative approach to the use of RF signal amplitude (RSSI) estimating is developed and made available by Renesas. This technique is trademarked WiRa™ (Wireless Ranging) and instead of relying on amplitude measurements, it utilizes phase-based ranging. In this alternative approach, the phase difference between adjacent ISM band channels is measured and because the two channels are at different (known) frequencies, this phase difference changes over distance in direct proportion to the distance traveled.

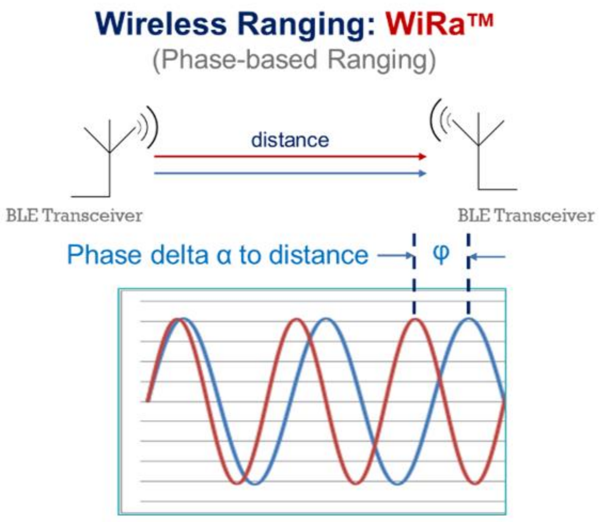


Figure 2. Wireless Ranging (WiRa) technique

While WiRa ranging can be far more tolerant to the direct path's signal attenuation, on its own this technique is still vulnerable to multipath signal noise. Renesas has therefore developed advanced algorithms to help address this issue.

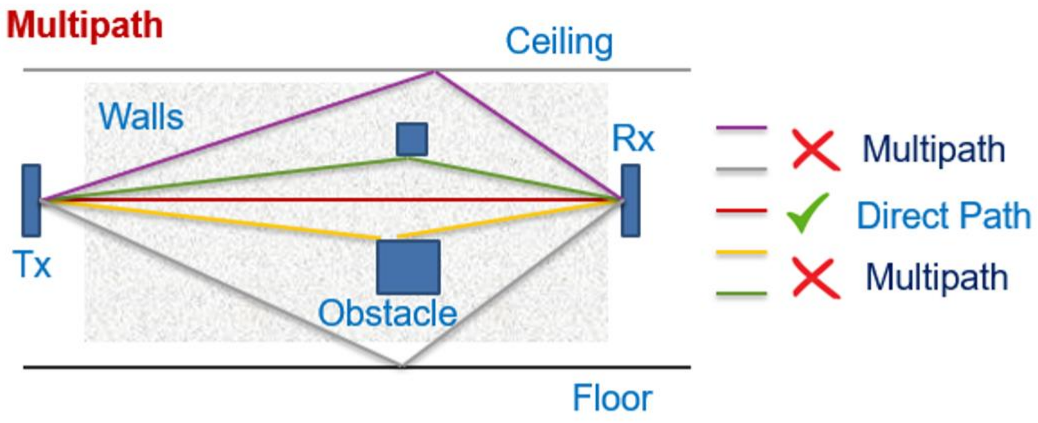


Figure 3. Multipath in a typical indoor environment

WiRa operates in the unlicensed 2.4 GHz ISM (Industrial, Scientific, Medical) band and therefore must tolerate noise sources from other ISM band constituents (for example, Bluetooth® LE, Wi-Fi, and so on) and the WiRa SDK includes an interference filtering solution that can tolerate strong noise presence in multiple channels.

Renesas delivers the WiRa™ SDK today, without licensing fees or royalties and you are invited to test, do field trials, and deploy what we developed. WiRa™ SDK runs on the DA1469x family of Bluetooth® LE SoCs.

4. WiRa Measurement Technology

4.1 Dialog Tone Exchange

The distance between two devices is calculated by measuring the phase difference of continuous wave signals at various frequencies that travel from one device to the other. Knowing this phase difference one can calculate the distance between the two devices.

To achieve this, both devices exchange continuous wave signals (tones) on a pre-defined set of frequencies at pre-defined points in time. This procedure is called Dialog Tone Exchange (DTE). The device that initiates this procedure is called the Initiator and the other device is called the Responder.

The Initiator transmits a tone at a specific frequency. The Responder receives this tone and captures the In-phase and Quadrature (IQ) data of the tone. Then roles are exchanged. The Responder transmits a tone and the Initiator receives the tone and captures the IQ data. The above procedure is called an atom and is repeated for all frequencies in the pre-defined set of frequencies.

The following parameters of DTE can be configured for each device:

- The role, either Initiator or Responder.
- The set of frequencies used in the tone exchange. The set is defined by setting the starting frequency, the frequency step, and the number of frequencies to use. The maximum number is 40 frequencies.
- The duration of each tone. The default value is 56 µs.
- The Radio transmit power during DTE. The default value is the maximum power (+6 dBm).

The devices must be synchronized to accurately calculate the phase difference of arrival (PDoA) between sequential tones of different frequencies.

The signal magnitude A and phase angle Ø of each tone exchanged are calculated as:

$$A\hat{}2 = I\hat{}2 + Q\hat{}2 \qquad (1)$$

$$\emptyset = arc\ tan\ (Q/I) \qquad (2)$$

Both devices calculate the phase and amplitude data of the received tones. The Responder then sends the results to the Initiator. The Initiator uses all results to calculate the distance.

A Bluetooth® LE connection is used for both the DTE synchronization and the exchange of the phase and amplitude results using a custom Bluetooth LE DTE Data service.

The tone exchange synchronization is based on the TX and RX timestamps related to the last packet transmission of the Bluetooth LE connection event. The tone exchange starts at a pre-defined time offset after the last Bluetooth LE packet transmission.

This exchange takes place in specific connection events. To achieve event synchronization, the Bluetooth LE connection event counter is used. In a Bluetooth LE connection, the role of the Initiator device is assigned to the central node and the Responder is assigned to the peripheral. The Initiator transmits to the Responder a "start request" control packet that contains an event counter value for a connection event instance in the future. At the end of the event that matches that counter, the tone exchange takes place if the Responder has enough time to receive this information and prepare for the tone exchange before that instance passes.

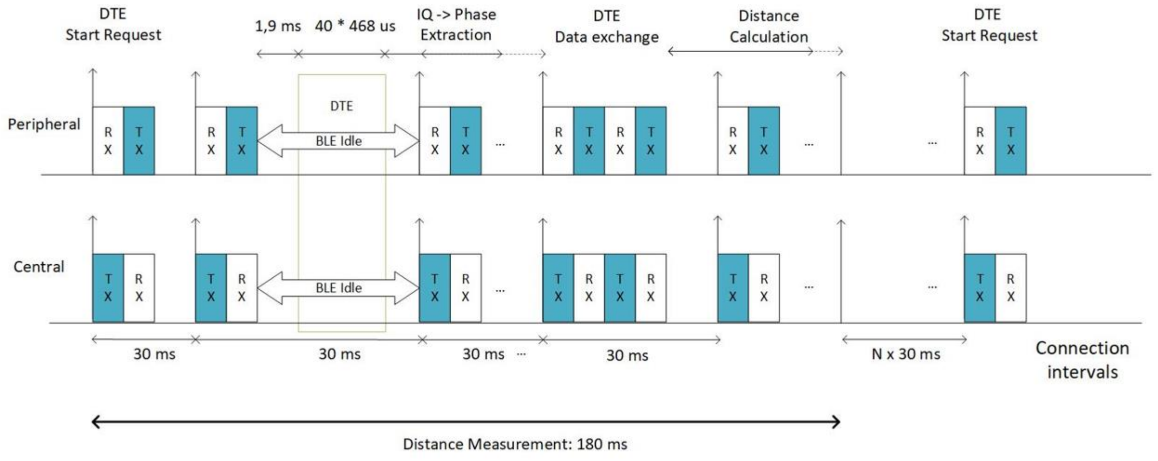


Figure 4. Distance measurement timeline

4.2 Data flow

The WiRa devices work in pairs. Each pair consists of one Initiator and one Responder device. When a connection is established between the Initiator and the Responder, the distance calculation procedure starts. The following sequence is executed to calculate the distance between the two devices:

1. The Initiator sends a Link Layer DTE START measurement request with a connection event counter that refers to a future instance.
2. When that instance comes, the two devices synchronize and carry out the tone exchange.
3. Both devices perform the phase calculations.
4. The Responder device sends the intermediate results to the Initiator.
5. The Initiator device combines the local and remote results received by the Responder, calculates the distance, and optionally sends the distance measurement to the Responder.

Figure 5 shows the above steps. The last step is optional and represented with dotted arrows.

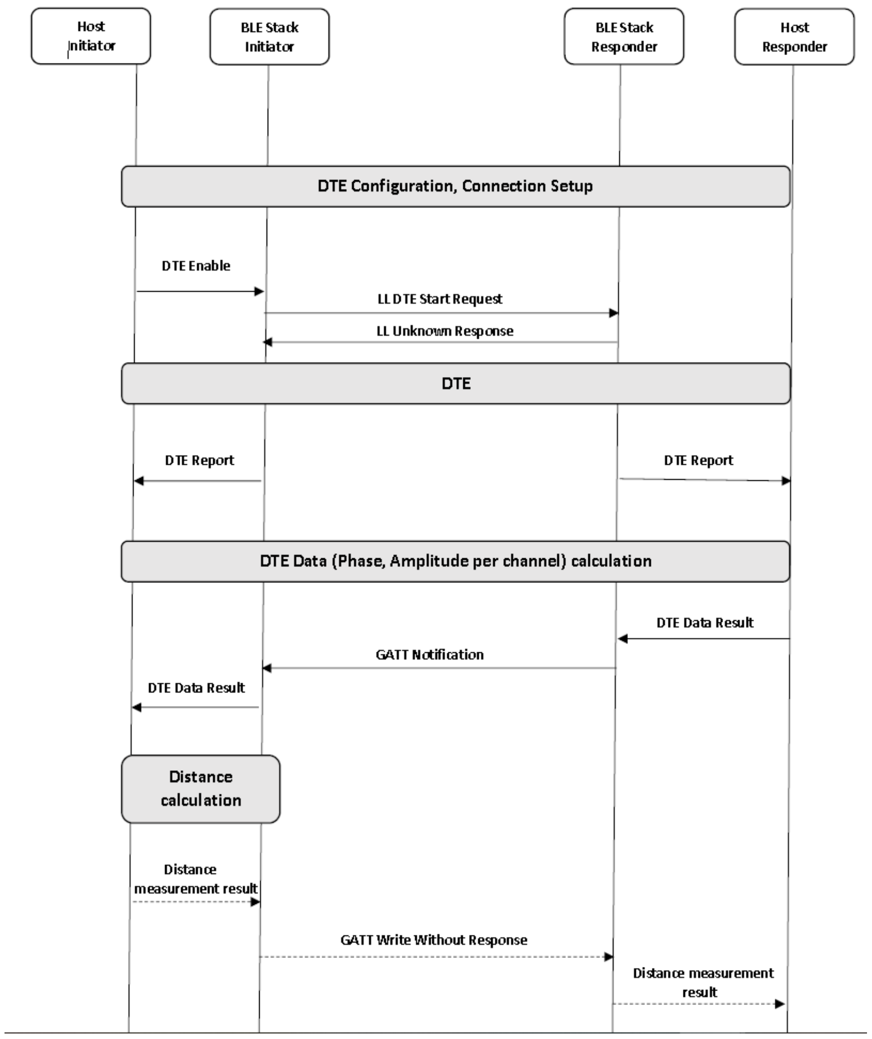


Figure 5. Application data flow

4.3 Implementation

4.3.1 IQ data acquisition

The radio is equipped with a hardware monitoring block (RFMON) that gets the data provided by the RF Unit, packs the data in 32-bit words, and stores them in the system memory. These data contain the 9-bit RF-ADC samples (IQ data) that are sampled at a frequency of 8 Msps.

The DTE module is responsible for the collection of IQ data for different frequencies. For each frequency, IQ data are collected through the RF monitor block to the IQ data buffer. At the end of the DTE, the data buffer contains the IQ data slices for the tones received in all frequencies.

4.3.2 Distance calculation algorithms

When the acquisition is finished, distance results are extracted from the sampled measurement data. All operations specific to distance estimation are implemented inside directory `sdk/interfaces/ble/wira/src/calc/`.

There are three algorithms available for distance calculation:

- Phase-based distance calculation (Section 4.3.2.1)
- Matrix pencil-based distance calculation (Section 4.3.2.2)
- IFFT-based distance calculation (Section 4.3.2.2).

The following steps are common to all algorithms:

1. Input data pre-conditioning:
    a. Extract IQ data from acquired test bus data samples (sampling rate of 8 Msps).
    b. Sample down by a factor of two to a sampling rate of 4 Msps. The number of samples to be processed is drastically reduced but the data samples still yield similar results.
2. Extract one representative sample per atom/measurement frequency (output data of this step is called "device-specific frequency profile" here):
    a. Compensate DC offset per atom (can be deactivated by defining `CWD_COMP_DC_OFFSET` to "false" in `sdk/ble/wira/src/calc/cw_distance_internal.h`).
    b. Go from the IQ domain into a polar domain (extract phases and magnitudes).
    c. Mix phases down by the intermediate frequency `f_if`.
    d. Apply a linear fit on the phase data of each atom to get one frequency offset and one phase value per atom.
    e. Average the magnitude values to get one magnitude per atom.
3. Average the frequency offset for all atoms.
4. Send the frequency profile of the Responder to the Initiator.

4.3.2.1 Phase-based distance calculation

If neither `WIRA_USE_MATRIX_PENCIL_DIST_CALC` nor `WIRA_USE_IFFT_DIST_CALC` are defined in the demo application's `wira_config.h`, then the phase-based distance calculation algorithm is used. The distance is determined as follows:

1. Retrieve the sum phase data of both devices per atom to get a Channel Frequency Profile.
2. Estimate the slope of the Channel Frequency Profile (phases).
    a. Build phase differences between subsequent phase values.
    b. Average the phase differences.
3. Map the estimated phase difference to a distance and take that as the estimated result.

The underlying relation between distance D, phase difference $\Delta\varphi$, frequency difference Δf, and speed of light c is:

$$D = \frac{c}{4\pi} \times \frac{-1 \times \sum_{N-1} \Delta\varphi_n}{(N-1) \times \Delta f} \tag{3}$$

where:

- D is the distance in m
- c is the speed of light in m/s
- $\Delta\varphi$ is the phase difference in radians
- Δf is the frequency difference in Hz
- N is the number of atoms.

4.3.2.2 Matrix pencil or IFFT-based distance calculation

The following steps are common regardless of the algorithm used for final distance calculation:

1. Combine device-specific frequency profiles into measurement-specific time offset function (phase_minus).
    a. Subtract the Responder phases from the Initiator phases.
2. Calculate offsets:
    a. Differentiate `phase_minus`.
    b. Determine the quadratic component (`c_quad`) of `phase_minus`.
    c. Correct the quadratic component of `phase_minus`.
    d. Determine the linear component (`c_lin`) of `phase_minus`.

3. Remove offsets:
    a. Create correction function offsets.
    b. Correct the Initiator and the Responder phases and calculate the corresponding channel transfer functions `phase_init_corr` and `phase_refl_corr`.
4. Fuse the Initiator and the Responder channel transfer functions:
    a. Combine `phase_init_corr` and `phase_refl_corr into com_IQ_fused`.

Matrix pencil-based distance calculation algorithm is used if `WIRA_USE_MATRIX_PENCIL_DIST_CALC` is defined and `WIRA_USE_IFFT_DIST_CALC` is undefined. Then the distance is determined by feeding the fused channel frequency response to Matrix Pencil processing block:

1. Estimate the path delay for each atom.
2. Find the shortest path delay and map its value to a distance.

IFFT-based distance calculation algorithm is used if `WIRA_USE_IFFT_DIST_CALC` is defined and `WIRA_USE_MATRIX_PENCIL_DIST_CALC` is undefined. Then the distance is determined as follows:

1. Go to the time domain through iFFT to get fused one-way channel impulse response.
2. Detect the most reasonable peak of the fused one-way channel impulse response:
    a. Detect as many maxima of the Channel Impulse Response as frequencies used during the measurement (default: 40).
    b. Remove all peaks below a configurable threshold to filter the peak list.
    c. Calculate the corresponding distance to each peak above the threshold.
    d. Find the peak with the shortest distance and take that as the estimated result.

4.3.3 Distance filtering

Two filters can be optionally applied to the raw distance measurements:

- **Sliding average filter**: the sample window size is controlled by the compile time switch `DIST_MA_WINDOW`.
- **Kalman filter**: an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, is applied to produce estimates of unknown variables that tend to be more accurate than those based on a single measurement, by estimating a joint probability distribution over the variables for each timeframe. There is a tradeoff between responsiveness and smoothing of the output of the Kalman filter, that can be controlled by variance process noise (q) and variance measurement noise (S) parameters. Visualization of the impact of S and q parameters are shown in Figure 6 and Figure 7.

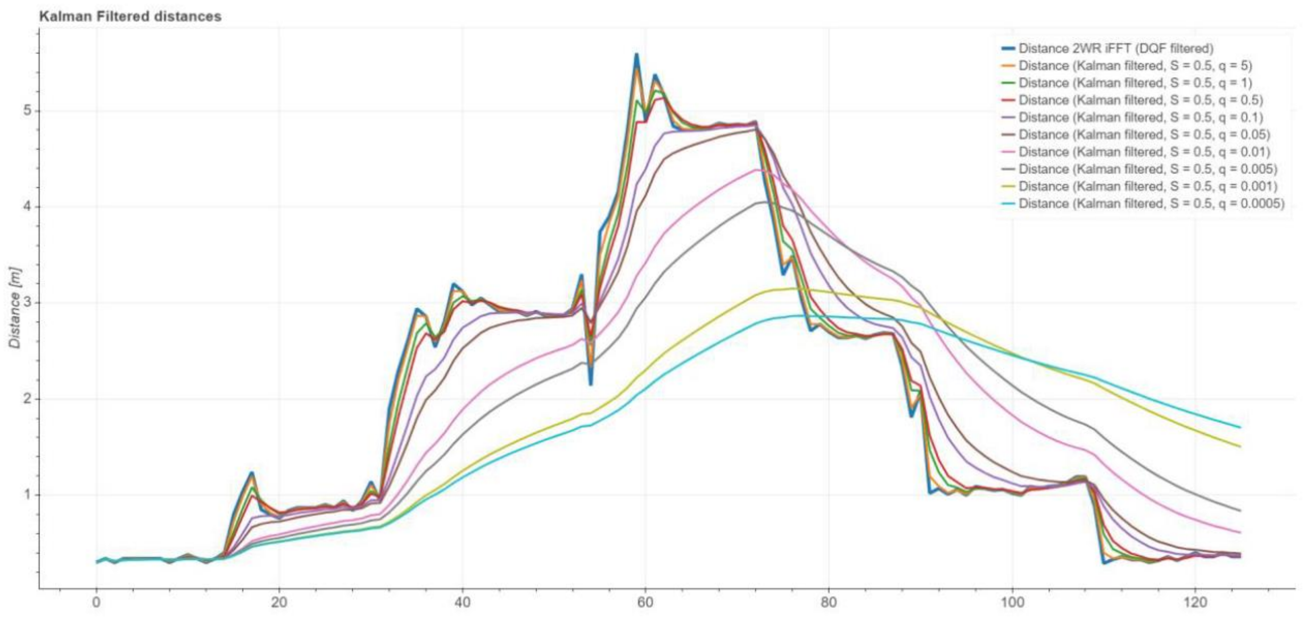


Figure 6. Influence of process noise variance in Kalman's filter responsiveness

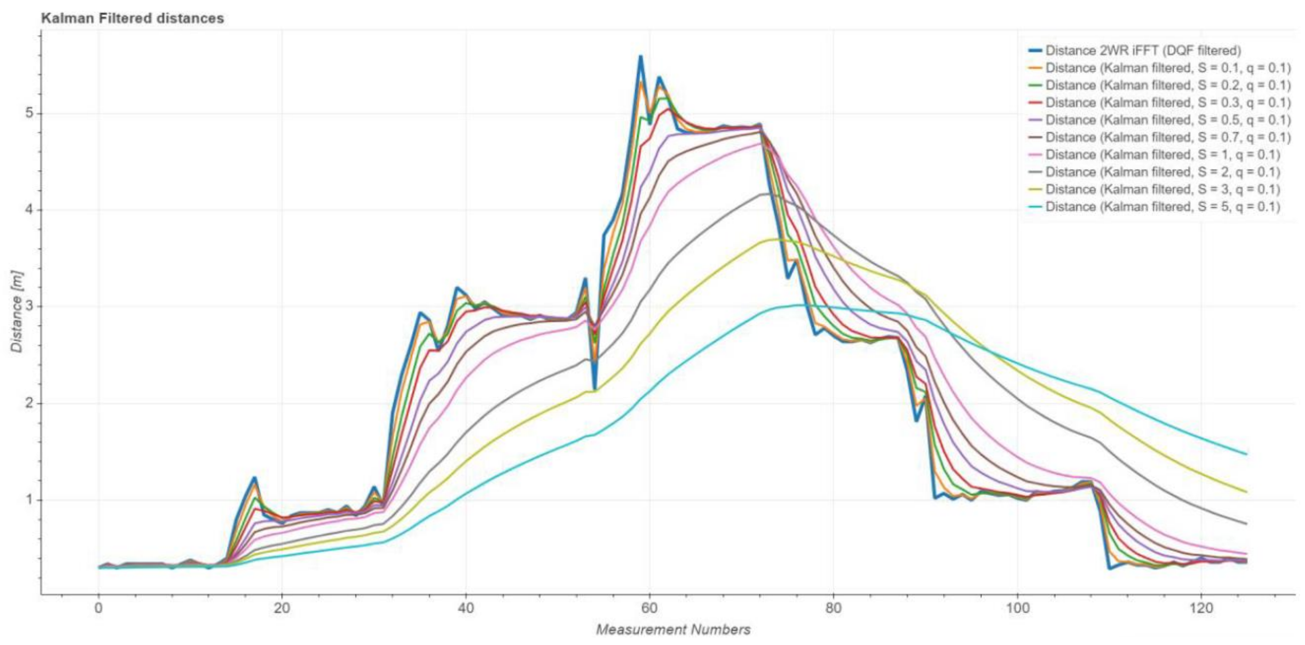


Figure 7. Influence of measurement noise variance in Kalman's filter responsiveness

The default value for variance process noise (q) is 0.1 and for variance measurement noise it is 0.5.

4.4 Software Architecture

Figure 8 shows different software blocks of the WiRa application. Each rectangle indicates a module. The dashed blocks represent optional features that can be enabled for specific demo applications at compile time.

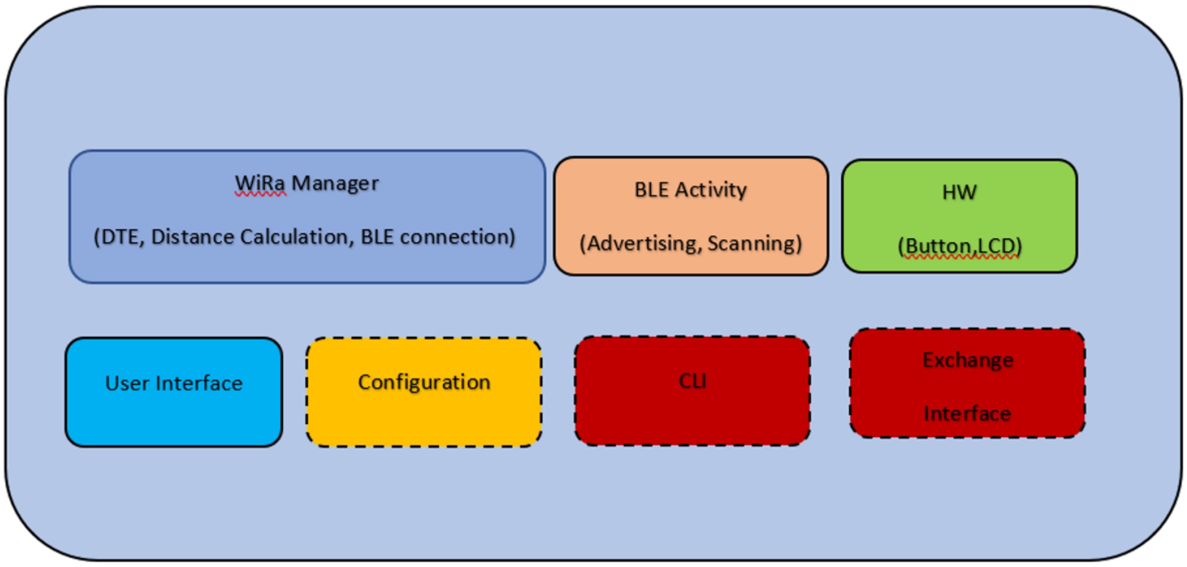


Figure 8. WiRa application software modules

WiRa manager is responsible for the DTE configuration, the initialization of the Initiator and the Responder, the connection between them, and for starting a WiRa measurement. A detailed description can be found in Section 4.5.

The Bluetooth® LE activity module set a Responder to the ADVERTISING state to be detected from an Initiator which is set to the SCANNING state.

The hardware module consists of an LCD and buttons to make WiRa applications user-friendly.

The high-level interface function `wira_lcd_init()` initializes the display and the function `wira_lcd_draw_measurement_string()` draws a string with the RSSI and the distance measurement value in a display.

The K1 button press is detected by checking the level of the GPIO which is connected to the button.

For the available WiRa applications, a button press either sets the device in Configuration mode or prints WiRa statistics depending on the duration of the button press.

The user interface module contains different font implementations and basic graphic functions for text drawing in an LCD. If `CONFIG_USE_WIRA_KIT` is defined, the LCD configuration is also enabled.

Using the Configuration module, you can change some non-volatile parameters, such as the role, the set ID, the offset, parameters for the Kalman filter, and so on. The configuration is done through the command line by keeping the K1 button pressed during reset (Section 5.1.6). For the `wira_basic_demo` and `wira_tracking_demo`, the configuration is done through an Android application (Appendix E).

The CLI is a special mode only for the Initiator device to scan for Responders and connect to them manually. It is supported only in `wira_eval_demo` and a detailed description is in Section 5.1.7.

The Exchange interface is an external interface module that supports communication with an external host through RAM using the Segger J-Link interface. Only the `wira_eval_demo` supports this interface which is described in detail in Section 5.1.8.

4.5 WiRa manager

4.5.1 Introduction

WiRa manager is a software module that provides a simple Application Programming Interface (API) that can be used in an application to establish a connection between an Initiator and a Responder and perform distance measurements.

A configuration file named `wira_config.h` can be used to set various built-time configuration parameters. If such a file is not included in the application code, then the default configuration file located in folder `\sdk\ble\wira\include` is used.

4.5.2 WiRa Bluetooth® LE manager API

4.5.2.1 Initialization and configuration functions

4.5.2.1.1 void wira_mgr_init(wira_cbs_t *cbs, gap_dte_params_t *dte_params)

This function is used to initialize the WiRa manager. Callback functions can be configured using parameter `cbs` which points to a structure of the following type:

```
typedef struct {
        wira_status_cb_t     status_cb;
        wira_iq_data_cb_t    iq_data_cb;
        wira_phase_data_cb_t phase_data_cb;
} wira_cbs_t;
```

The function assigned to `status_cb` is called from the WiRa manager to notify the application that the status of the manager changed and to provide the WiRa measurement results.

The function assigned to `iq_data_cb` is called when new IQ data is captured.

The function assigned to `phase_data_cb` is called when new phase and magnitude data is calculated.

DTE configuration is defined using the parameter `dte_params` which points to a structure of the following type:

```
typedef struct {
        bool is_initiator;
        uint8_t nb_atoms;
        uint16_t meas_length_us;
        uint16_t f_start_mhz;
        uint8_t f_step_mhz;
        uint8_t agc_freeze_lvl;
        bool agc_freeze_auto;
        uint8_t tx_power;
} gap_dte_params_t;
```

Default values as defined in `wira_config.h` are used if `dte_params` is NULL.

The configuration parameters are the following:

- `is_initiator`: set to true to configure the device as Initiator, false to configure it as Responder.
- `f_start_mhz`: the frequency of the first atom. The default value is 2402 MHz.
- `f_step_mhz`: the frequency difference between two consecutive atoms. The default value is 2 MHz.
- `nb_atoms`: the total number of atoms. The default value is 40.
- `meas_length_us`: the length of the data section to be copied from each atom. The default value is 56 µs.
- `agc_freeze_lvl`: set Radio AGC gain to the specified level. If `agc_freeze_auto` is set to true, this value is ignored.
- `agc_freeze_auto`: set to true to enable automatic gain control.
- `tx_power`: set the Radio transmit power. The default value is +6 dBm.

4.5.2.1.2 void wira_mgr_periph_init(wira_validate_connection_func_t func)

This function is used to initialize the WiRa Responder (in a peripheral role). Parameter `func` can be set to point to a function that is used for validating an incoming connection. If this function returns `WIRA_VALID_OK` the WiRa connection is established. If it returns `WIRA_VALID_LOW_BAT`, then a low battery indication is sent to the Initiator to terminate the connection. If it returns `WIRA_VALID_NO_WIRA_PEER`, then WiRa is not enabled.

4.5.2.1.3 void wira_mgr_periph_data_service_init(void)

This function is used to initialize the custom WiRa Data Service.

4.5.2.1.4 void wira_mgr_periph_enable(bool enable)

This function enables or disables the WiRa Responder. If WiRa Responder is disabled, it is not responding to `LL_DTE_START` requests.

4.5.2.1.5 void wira_mgr_periph_set_id(wira_id_t id)

This function sets the ID of the Responder.

4.5.2.1.6 void wira_mgr_central_init(void)

This function is used to initialize the WiRa Initiator (in the central role).

4.5.2.1.7 void wira_mgr_central_set_id(wira_id_t id)

This function sets the ID of the Initiator.

4.5.2.2 Event handlers

4.5.2.2.1 bool wira_mgr_ble_handler(ble_evt_hdr_t *hdr)

This function must be called from the Bluetooth® LE task of the application to handle Bluetooth® LE events needed for the operation of the WiRa manager.

4.5.2.2.2 void wira_mgr_notif_handler(uint32_t notif)

This function must be called from the Bluetooth® task of the application to handle notification events needed for the operation of the WiRa manager.

4.5.2.3 Connection and measurement functions

4.5.2.3.1 void wira_mgr_central_connect(bd_address_t *addr, const gap_ext_conn_params_t *conn_params, bool disconnect_after_measurement)

This function can be used to establish a connection with a Responder having the BD address assigned to the `addr` parameter. Custom connection parameters can be set using the `conn_params` parameter. If set to NULL, the default connection parameters defined in `wira_config.h` are used. If `disconnect_after_measurement` is set to true, the connection is dropped immediately after the WiRa measurement. When a connection is established, the status callback defined in `wira_mgr_init()` is called to provide the connection context to the application.

4.5.2.3.2 void wira_mgr_central_start_measurement(conn_context_t *ctx)

This function can be used to start a WiRa measurement after a connection is established with the Responder. Parameter `ctx` must point to the connection context provided when the status callback was called. If set to NULL, WiRa measurement is performed with the last Responder with which a connection was established.

4.5.2.3.3 void wira_mgr_central_connect_and_start_measurement(bd_address_t *addr, bool disconnect_after_measurement)

This function is a combination of `wira_mgr_central_connect()` and `wira_mgr_central_start_measurement()`. If a connection is already established, then WiRa measurement starts immediately.

4.5.2.4 Errors and Statistics

The following error codes (as defined in enumeration `dte_error_t`) are reported and logged:

- `DTE_NO_ERROR`: the measurement sequence finished without any error.
- `DTE_ERROR_INSTANT_PASSED`: failed to start measurement because the requested connection event counter passed.
- `DTE_ERROR_SYNC_FAILED`: failed to synchronize with the partner device.
- `DTE_ERROR_RESULTS_TO`: failed to receive either the report with the acquisition data from the RF Unit or the intermediate results from the remote device.
- `DTE_ERROR_DISCONNECTED`: failed because the peer is disconnected.

Every device maintains a table with measurement statistics for each DTE capable connection which can be printed to the debug console by calling the function `wira_mgr_print_statistics_report()`.

5. Demo Applications

5.1 WiRa Evaluation Demo application

5.1.1 Introduction

This application shows how Bluetooth® LE and Dialog Tone Exchange (DTE) are used to get distance measurements between two devices, one Initiator and one Responder. This demo application is the default demo in the development kits. It supports a configuration mode to change parameters easily through the terminal. It provides also a CLI special mode only for the Initiator, where you can perform a set of commands on demand.

5.1.2 Description

A Bluetooth® LE connection is used for both the DTE synchronization and the exchange of the intermediate DTE Data. There are two configuration parameters related to the Bluetooth® LE connection: the role and the set number.
A device is configured to have the role of an Initiator (Section 5.1.2.1) or the role of a Responder (Section 5.1.2.2).
The set number is used to distinguish two different WiRa sets that are simultaneously operated in each other's vicinity (within the Bluetooth® LE reception range).
Both devices, Initiator and Responder go through a sequence of states before the measurement cycle can start. Initially, the Responder starts advertising with a custom DTE Data service UUID and the set number in the advertising data. The Initiator starts scanning, looking for the DTE Data service UUID and a matching set number in the advertising reports it receives. If both the UUID and the set number match, the two devices get connected.

5.1.2.1 Initiator Role

The sequence of states for the Initiator is the following:

- **SCANNING state**: the device starts scanning, looking for the WiRa Data service's UUID and a matching set number in the received advertising reports. If both the UUID and the set number match, the device attempts to connect with the remote device.
- **CONNECTED state**: the connection attempt is successful
- **MAXIMUM TRASMISSION UNIT (MTU) EXCHANGE State**: the Initiator requests the maximum allowed MTU to increase the results throughput. Data packet length extension is by default enabled on the DA1469x devices, so the PDU Payload Length in the Maximum Transmit Data Channel is 251 octets.
- **DISCOVERY state**: the device discovers Bluetooth LE services and related attributes.

After the discovery is completed, the device starts the measurement cycle.
In CLI mode, the SCANNING is bypassed, and the device enters a CONNECTING State in which an attempt is made to connect to a remote device with a specific Bluetooth® address that is a parameter of the connect command.

5.1.2.2 Responder role

The sequence of states for the Responder is the following:

1. **ADVERTISING state**: the device starts advertising with the WiRa Data service's UUID and the device set number in the advertising data. Then, the Responder waits for connection requests from the Initiator device.
2. **CONNECTED state**: after the two devices connect, the Responder replies to the discovery and MTU requests and waits for a START request to start the measurement sequence.

5.1.3 Configuration options

The functionality and the performance of the demo can be customized by modifying various build-time or run-time configuration parameters.

5.1.3.1 Build-Time Configuration Parameters

The build-time configuration parameters are defined in configuration files located at `projects/dk_apps/wira/wira_eval_demo/config/` folder. There are configuration parameters related to WiRa, parameters related to the specific demo, and global parameters.

Configuration parameters related to WiRa are defined in `wira_config.h`. These parameters define the behavior of the WiRa manager.

- `wira_config.h`:
  - WIRA_RESPONDER_ROLE and WIRA_INITIATOR_ROLE are both defined, and the development kit starts as an Initiator, as it is defined by the DEFAULT_DTE_ROLE in `wira_eval_demo_config.h`.
  - WIRA_MAX_DISTANCE_CM: WiRa calculated distance greater than the max distance is rejected.
  - WIRA_MIN_RSSI_THRESHOLD: if the RSSI is lower than this minimum threshold, the measurement is rejected.
  - WIRA_CW_OFFSET_CM: a constant offset that is deducted from the WiRa measurements.
  - WIRA_USE_IFFT_DIST_CALC and WIRA_USE_MATRIX_PENCIL_DIST_CALC: these are the two methods for distance calculation. Only one can be defined. Matrix pencil distance calculation is used by default. If none of them is defined, then phase-based distance calculation is used to calculate the distance (Section 4.3.2).
  - WIRA_MULTILINK_SUPPORT: if it is defined, WiRa manager can handle multiple connections.
  - WIRA_SEND_RESULT_TO_RESPONDER: if it is defined, the Initiator sends the result to the Responder by writing the value to the WiRa data R result characteristic.
  - WIRA_MEASUREMENT_TO_MS: time within which a WiRa measurement should be performed. If it is not, then the measurement is considered invalid.
  - WIRA_ENABLE_DTE_STATISTICS: if it is defined, DTE statistics are enabled.
  - WIRA_DEBUG_MEASUREMENTS: if it is defined, WiRa log messages are enabled to get raw measurements and additional details about WiRa.

  Configuration parameters related to the demo application are defined in `wira_eval_demo_config.h`.
- `wira_eval_demo_config.h`:
  - EXCHANGE_MODE: if it is set to 1, the application exchanges data with an external Python application (Section 5.1.8). It is set to 0 by default.
  - DEFAULT_DTE_ROLE: configures the default role of a device. For the `wira_eval_demo` application, the role is set to Initiator by default.
  - The connection parameters are defined by the symbols:
    - BLE_CONN_INTERVAL_MIN
    - BLE_CONN_INTERVAL_MAX
    - BLE_CONN_SLAVE_LATENCY
    - BLE_CONN_SUPERVISION_TO
  - The scan parameters for the Initiator are defined by the symbols:
    - BLE_SCAN_INTERVAL
    - BLE_SCAN_WINDOW
  - The advertisement parameters for the Responder are defined by the symbols:
    - BLE_ADV_INTERVAL_MIN
    - BLE_ADV_INTERVAL_MAX
  - PRINT_DTE_RSSI: if it is set to 1, the RSSI is printed on the console.
  - LCD_DTE_RSSI: if it is set to 1, the RSSI is printed on the LCD.
  - PRINT_WIRA_COMPACT: if it is set to 1, a compact print of the raw result is performed.
  - DIST_MA_WINDOW: defines a 2^N sized sliding window filter to apply on raw values for average distance reporting.
  - HOST_PM_MODE_ACTIVE: if it is set to 1, Active power management mode is configured instead of the extended sleep.

  Global project configuration parameters are defined in `custom_config_qspi.h`.
- `custom_config_qspi.h`:

- Enable UART logging: if the symbol CONFIG_RETARGET is defined, then the application log messages are sent to the UART interface. The UART configuration settings are the following:
  - Baud rate: 115200
  - Data bits: 8
  - Parity: None
  - Stop bits: 1.

5.1.4 Running the Demo after unboxing the Development kits

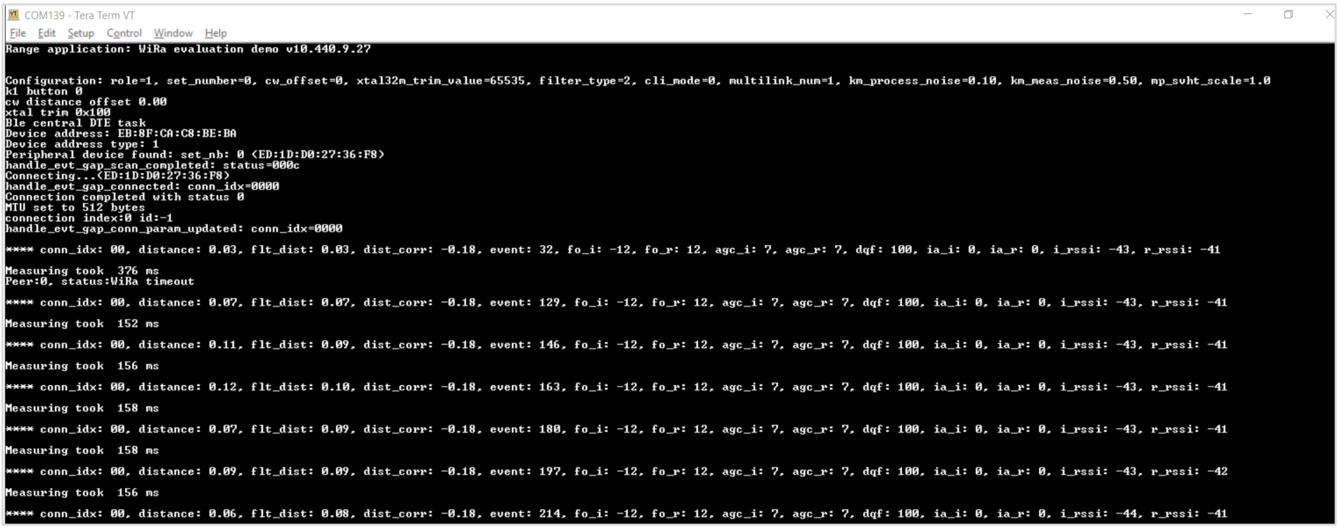
After plugging each Smartbond Wireless Ranging USB Development Kit into a USB port, the WiRa Evaluation Demo application runs. There are one Initiator and one Responder in the box. During the startup, you can see on the display "Initiator connecting" or "Responder connecting", which is an indication that the devices are not connected yet, but they scan/advertise respectively.

After a connection is established between the two devices, you can read the distance (in the unit of meters) between the two devices from both displays.

Open for each device a UART terminal with the following configuration to get the logs:

- Baudrate: 115200
- Bits: 8
- Parity: None
- Stop bits: 1.

On the Initiator's console, you can see something like in Figure 9.



```
Range application: WiRa evaluation demo v10.440.9.27

Configuration: role=1, set_number=0, cw_offset=0, xtal32m_trim_value=65535, filter_type=2, cli_mode=0, multilink_num=1, km_process_noise=0.10, km_meas_noise=0.50, mp_svht_scale=1.0
k1 button 0
cw distance offset 0.00
xtal trim 0x100
Ble central DTE task
Device address: EB:8F:CA:C8:BE:BA
Device address type: 1
Peripheral device found: set_nb: 0 (ED:1D:D0:27:36:F8)
handle_evt_gap_scan_completed: status=000c
Connecting...(ED:1D:D0:27:36:F8)
handle_evt_gap_connected: conn_idx=0000
Connection completed with status 0
MTU set to 512 bytes
connection index:0 id:-1
handle_evt_gap_conn_param_updated: conn_idx=0000

**** conn_idx: 00, distance: 0.03, flt_dist: 0.03, dist_corr: -0.18, event: 32, fo_i: -12, fo_r: 12, agc_i: 7, agc_r: 7, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -43, r_rssi: -41

Measuring took  376 ms
Peer:0, status:WiRa timeout

**** conn_idx: 00, distance: 0.07, flt_dist: 0.07, dist_corr: -0.18, event: 129, fo_i: -12, fo_r: 12, agc_i: 7, agc_r: 7, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -43, r_rssi: -41

Measuring took  152 ms

**** conn_idx: 00, distance: 0.11, flt_dist: 0.09, dist_corr: -0.18, event: 146, fo_i: -12, fo_r: 12, agc_i: 7, agc_r: 7, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -43, r_rssi: -41

Measuring took  156 ms

**** conn_idx: 00, distance: 0.12, flt_dist: 0.10, dist_corr: -0.18, event: 163, fo_i: -12, fo_r: 12, agc_i: 7, agc_r: 7, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -43, r_rssi: -41

Measuring took  158 ms

**** conn_idx: 00, distance: 0.07, flt_dist: 0.09, dist_corr: -0.18, event: 180, fo_i: -12, fo_r: 12, agc_i: 7, agc_r: 7, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -43, r_rssi: -41

Measuring took  158 ms

**** conn_idx: 00, distance: 0.09, flt_dist: 0.09, dist_corr: -0.18, event: 197, fo_i: -12, fo_r: 12, agc_i: 7, agc_r: 7, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -43, r_rssi: -42

Measuring took  156 ms

**** conn_idx: 00, distance: 0.06, flt_dist: 0.08, dist_corr: -0.18, event: 214, fo_i: -12, fo_r: 12, agc_i: 7, agc_r: 7, dqf: 100, ia_i: 0, ia_r: 0, i_rssi: -44, r_rssi: -41
```

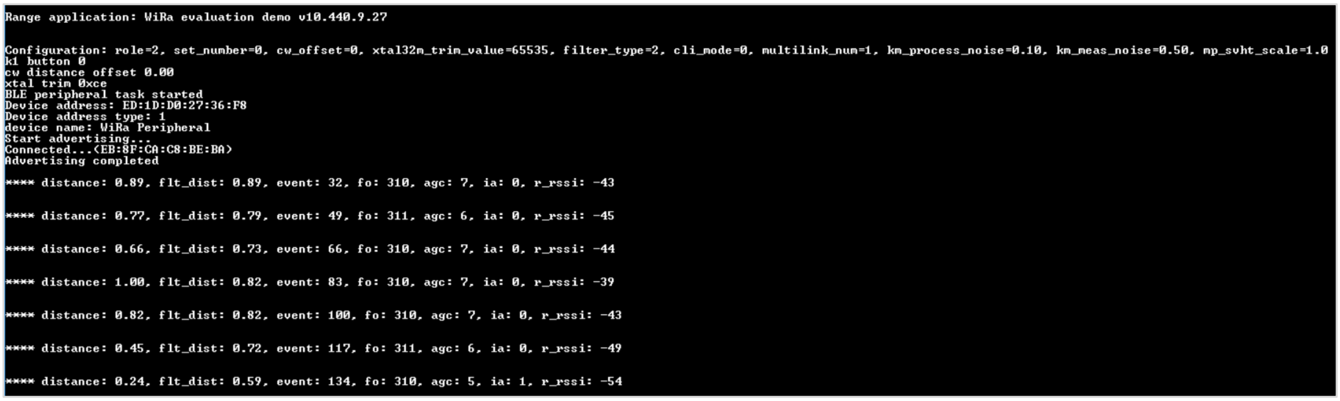
Figure 9. Logging information of Initiator

At the beginning of the logging board's configuration is printed. Afterward, when a peripheral is scanned the connection procedure started and, in the log, the Responder's BD address, the MTU, and the connection index are printed. The output log contains the following information:

- **conn_idx**: is the Bluetooth LE connection that provides the anchor times for the ranging information related radio measurements.
- **distance**: is the actual distance measured in meters after the **cw_offset** is applied.
- **flt_dist**: if distance filtering is enabled, this is the output of a filter applied to the valid distance measurements (with dqf:100). Three filter options are available:
  - no-filtering
  - moving average configurable sample window
  - Kalman filter
- **dist_corr**: is a correction factor automatically applied to the calculated distance to compensate for the influence of the offset between the 32-MHz clock crystals (up to +/-50 ppm) on the Initiator and Responder. Therefore, crystals and crystal trimming fulfilling regular Bluetooth specification suffice for WiRa use as well.
- **event**: is the connection event at which a radio measurement took place.

- **fo_i**: is the frequency offset in kHz for the Initiator.
- **fo_r**: is the frequency offset in kHz for the Responder.
- **argc_i**: is the AGC gain value of the Initiator used during the data acquisition.
- **argc_r**: is the AGC gain value of the Responder used during the data acquisition.
- **dqf**: is a distance quality factor based on the frequency offset values of both sides. It should be 100 for a good measurement.
- **ia_i** and **ia_r**: are the numbers of Tone Exchange steps with invalid IQ data as identified during the IQ data acquisition phase. Currently, if there are any such steps, the whole measurement is dropped (dqf = 0).
- **i_rssi** and **r_rssi**: are the RSSI values of the last Bluetooth LE reception of the Initiator and the Responder respectively.
- **measuring time:** it is the time from sending the start command until the measurement is stopped.

Figure 10 shows the Responder's logging information. As you can see less information is printed than in the Initiator's log.



```
Range application: WiRa evaluation demo v10.440.9.27

Configuration: role=2, set_number=0, cw_offset=0, xtal32m_trim_value=65535, filter_type=2, cli_mode=0, multilink_num=1, km_process_noise=0.10, km_meas_noise=0.50, mp_svht_scale=1.0
k1 button 0
cw distance offset 0.00
xtal trim 0xce
BLE peripheral task started
Device address: ED:1D:D0:27:36:F8
Device address type: 1
device name: WiRa Peripheral
Start advertising...
Connected...(EB:8F:CA:C8:BE:BA)
Advertising completed

**** distance: 0.89, flt_dist: 0.89, event: 32, fo: 310, agc: 7, ia: 0, r_rssi: -43

**** distance: 0.77, flt_dist: 0.79, event: 49, fo: 311, agc: 6, ia: 0, r_rssi: -45

**** distance: 0.66, flt_dist: 0.73, event: 66, fo: 310, agc: 7, ia: 0, r_rssi: -44

**** distance: 1.00, flt_dist: 0.82, event: 83, fo: 310, agc: 7, ia: 0, r_rssi: -39

**** distance: 0.82, flt_dist: 0.82, event: 100, fo: 310, agc: 7, ia: 0, r_rssi: -43

**** distance: 0.45, flt_dist: 0.72, event: 117, fo: 311, agc: 6, ia: 0, r_rssi: -49

**** distance: 0.24, flt_dist: 0.59, event: 134, fo: 310, agc: 5, ia: 1, r_rssi: -54
```

Figure 10. Logging information of Responder

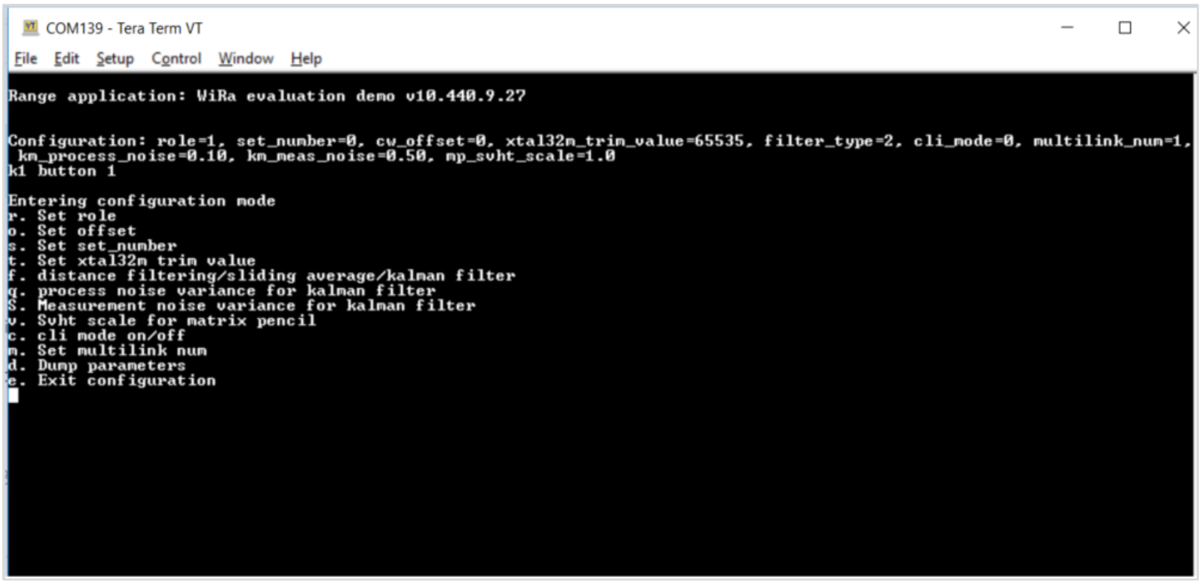
5.1.5 Downloading and running the Demo

The following steps are required to run the `wira_eval_demo` in case another firmware is already programmed to the board:

1. Configure the device as described in Appendix A.
2. Power up the first device.
3. Download the firmware to the device:
    a. If the device is already programmed with firmware that supports SUOTA, then the firmware can be updated over the air as described in Appendix C. The SUOTA images for the supported boards (DA1469x PRO DK, DA14695 Module DK, USB Kit, and WiRa Kit) are provided in folder `/binaries/wira/wira_eval_demo/suota_images/`.
    b. If the device does not support SUOTA, then:
        i. Use the SmartSnippets Toolbox to erase the data of the whole Flash area, as described in Appendix B in step 8. This is required to delete the old partition table.
        ii. Use the SmartSnippets Toolbox to program the device as described in Appendix B. The binary files for the supported boards (DA1469x PRO DK, DA14695 Module DK, USB Kit, and WiRa Kit) are provided in folder `/binaries/wira/wira_eval_demo/binaries/`.

5.1.6 Configuration mode

This demo supports a configuration mode and you can change some non-volatile parameters, such as the role, the set ID, the offset, enable the CLI mode, parameters for the Kalman filter, and so on. The device enters configuration mode by holding the K1 button pressed during RESET. A RESET is invoked by pressing the K2 button. The configuration menu is as in Figure 11.



```
Range application: WiRa evaluation demo v10.440.9.27

Configuration: role=1, set_number=0, cw_offset=0, xtal32m_trim_value=65535, filter_type=2, cli_mode=0, multilink_num=1,
 km_process_noise=0.10, km_meas_noise=0.50, mp_svht_scale=1.0
k1 button 1

Entering configuration mode
r. Set role
o. Set offset
s. Set set_number
t. Set xtal32m trim value
f. distance filtering/sliding average/kalman filter
q. process noise variance for kalman filter
S. Measurement noise variance for kalman filter
v. Svht scale for matrix pencil
c. cli mode on/off
m. Set multilink num
d. Dump parameters
e. Exit configuration
```

Figure 11. Configuration mode options

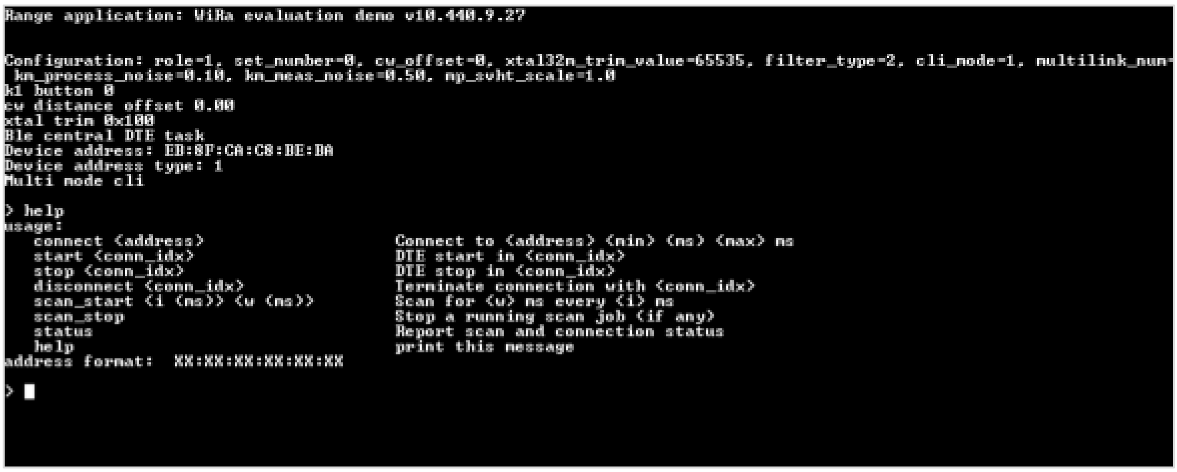
A detailed description of the available options is below:

- **r**: configure the role which is related to the acquisition TX/RX sequence. Two roles are defined, the Initiator and the Responder.
- **o**: set the offset in centimeters. This offset adjusts the distance measurement so that the correct distance result is reported. Initially, calibration of the distance measurements at fixed distances is needed to calculate the initial offset value.
- **s**: set the set ID. It is a number less or equal to 254. It is used at normal operation to uniquely identify an Initiator/Responder set. The Initiator device uses this number to identify the Responder to which it connects. It is not meaningful if CLI mode is enabled.
- **t**: set the XTAL frequency trimming register. This value may change if auto trimming is enabled.
- **f**: if it is set to zero, no filtering is applied and all measurements are reported raw to the output. If it is set to a value other than zero, then you can select between two possible filters that can be applied to the distance measurements:
  - **sliding average filter (1)**: a 2^N sliding window filter to apply to the raw values. The sample window size is controlled by the compile time switch DIST_MA_WINDOW.
  - **Kalman filter (2)**: is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe.

  q and **S**: there is a tradeoff between responsiveness and smoothing of Kalman's filter output, which can be controlled by the tuning parameters of `km_variance_process_noise_q` and `km_variance_measurement_noise_S`.
- **v**: set the SVHT scale for matrix pencil.
- **c**: if `cli_mode` is set, the Initiator operates in Manual mode leveraging a `cli` for assisted user setup. The `cli_mode` is described in detail in Section 5.1.7. This setting does not apply to Responder and is therefore ignored.
- **m**: set the max number of connections. The Initiator continues to scan for more connections after a connection is established until `multilink_num` is reached. Currently the DTE range measurement is triggered for every connection in a Round-Robin fashion. The value of `multilink_num` is ignored if the device is:
  - a Responder or
  - an Initiator that operates in CLI mode.
- **d**: print the parameters.
- **e**: the device exits from Configuration mode, it resets with the new parameters applied.

5.1.7 CLI mode

The CLI is a special mode for the Initiator, which may be enabled in the Configuration mode, as described in Section 5.1.6, where the setting for CLI mode is set to on. When operating in CLI mode, the Initiator waits for you to issue any of the available CLI commands (Figure 12). For more detailed description and functionality, see Section 5.1.7.1.



```
Range application: WiRa evaluation demo v10.440.9.27

Configuration: role=1, set_number=0, cw_offset=0, xtal32m_trim_value=65535, filter_type=2, cli_mode=1, multilink_num=
 km_process_noise=0.10, km_meas_noise=0.50, mp_svht_scale=1.0
k1 button 0
cw distance offset 0.00
xtal trim 0x100
Ble central DTE task
Device address: ED:8F:CA:C8:DE:DA
Device address type: 1
Multi mode cli

> help
usage:
   connect <address>                    Connect to <address> <min> <ms> <max> ms
   start <conn_idx>                     DTE start in <conn_idx>
   stop <conn_idx>                      DTE stop in <conn_idx>
   disconnect <conn_idx>                Terminate connection with <conn_idx>
   scan_start <i (ms)> <w (ms)>         Scan for <w> ms every <i> ms
   scan_stop                            Stop a running scan job (if any)
   status                               Report scan and connection status
   help                                 print this message
address format:  XX:XX:XX:XX:XX:XX

> █
```

Figure 12. CLI mode for Initiator

The CLI functionality is implemented in file `projects/dk_apps/wira/wira_eval_demo/src/cli.c`. The function `cli_init()` initializes the CLI task. This task receives input strings and notifies the waiting Bluetooth® LE central task that a string is read from the CLI. Function `cli_get_clistr()` returns the current input string that is read from the CLI. Function `readline()` reads a line from the CLI and `parseline()` parses the line to get a command.

5.1.7.1 CLI commands

The following CLI commands are available:

- **connect <address> <min> <max>**: this command takes one mandatory and two optional input arguments. The first argument must be the address of the remote device to try to connect to and must be in the form <xx:xx:xx:xx:xx:xx>. Additionally, up to two input arguments may be used to specify the minimum and maximum values for the connection interval. If any of the optional arguments are not given, the default value of 30 ms is used.

NOTE
Before attempting to make a connection, stop any user-scheduled scan . For more details, see command **scan_stop**.

- **start <conn_idx>**: with this command, the Initiator is instructed to trigger the DTE range measurement for the specified connection index.
- **stop <conn_idx>**: the device stops triggering any further DTE range measurements for this connection index.
- **disconnect <conn_idx>**: if a valid connection index is provided as an input argument, the Initiator tries to terminate the corresponding connection.
- **scan_start <interval> <window>**: the Initiator tries to schedule a scan job of <window> milliseconds every <interval> milliseconds. The scan interval needs to be greater than the scan window otherwise the command fails. The status command, described below, can be used to check whether you have a scan job already scheduled in the background or not. Currently, the command provides no console output and its usage is intended for testing purposes only.
- **scan_stop**: the device tries to remove any scan job you scheduled.
- **status**: Helper command to display information about the available connections. Before execution, entries of the format <conn_idx> <address> <DTE status> are printed on the console output. DTE status indicates whether the range measurement for this connection is unsupported, disabled, or enabled. If DTE ranging is supported, then the start and stop commands described earlier can be used to enable or disable it. Besides the list, the command output also provides information regarding the number of connections, active scan jobs, and ongoing advertising.

5.1.8 Exchange interface

The Exchange interface is an external interface module that supports communication with an external host through RAM using the SEGGER J-Link. Only the **wira_eval_demo** supports this interface. To enable the external interface functionality, set the define `EXCHANGE_MODE` to 1 in file `projects/dk_apps/wira/wira_eval_demo/config/wira_eval_demo_config.h`.

This mode supports Python interaction through SEGGER J-Link using PyMon. PyMon is a python package that enables the control of a Bluetooth® SoC through a SEGGER J-Link debugger probe.

The header file `projects/dk_apps/wira/wira_eval_demo/include/exchange_mem.h` describes a sample set of parameters for data exchange between the application and Python. There are three different types of parameters:

- Bidirectional flags used for a handshake between the DTE application and Python scripts
- Configuration parameters set from Python
- Data parameters sent to Python.

These parameters are stored in a special `.exchange_section` so that they can be located at a fixed predefined address that Python scripts can access. For this reason, the original `sections.ld.h` SDK file is modified and the `.exchange_section` is added.

There are four main functions for the exchange mode support:

- `exchange_mem_init()` for handshake parameters initialization from the application side
- `exchange_mem_params_exchange()` for setting the `is_initiator`, `f_start_mhz`, `f_step_mhz`, `nb_atoms` DTE configuration parameters from Python
- `exchange_mem_iq_data_set()` and `exchange_mem_phase_data_set()` for sending IQ data and phase/magnitude information to Python.

The exchange functions implement a blocking handshake where they set a ready flag, and then wait for Python code to set parameters/copy information and then clear the flag.

There is an initial handshake between the application and python in the function `exchange_mem_params_exchange()`.

Then at every measurement, there are two handshakes, one for IQ data `exchange_mem_iq_data_set()` and one for phase and magnitude data exchange `exchange_mem_phase_data_set()`.

A python script **dte_iq_data_analyzer.py** to test the IQ data and phase exchange and the PyMon module is available in `projects/host_apps/python_iqdata_tools`.

The script can be tested with the Spyder python environment that is included in Anaconda, the Python Data Science Platform.

To show the plots in Spyder in their own interactive window, go to **Tools** > **Preferences** > **Ipython console** > **Graphics tab** > **Graphics backend** > **Backend** and set it to **Automatic**.

5.2 WiRa basic application Demo

5.2.1 Introduction

This application demonstrates the use of WiRa technology to estimate the distance between two devices, one acting as an Initiator and one as a Responder.

Each device is assigned a set ID. The Initiator connects to the Responder with the same set ID to measure the distance.

Bursts of multiple WiRa events are executed. The data of all WiRa events are evaluated, filtered, and processed to calculate the distance estimation. The procedure is repeated with a predefined period.

The device enters Hibernation mode when the device remains inactive for a while.

The set ID, number of WiRa events per measurement, measurement period, and inactivity period can be configured using the WiRa Android application (See Appendix E).

Device logs including WiRa event data and distance estimation are sent to the UART or USB CDC interface. The information is also transmitted in beacon advertising packets by the Initiator that is monitored using the WiRa Android application (See Appendix E).

The application can run in DA1469x PRO DK, DA14695 Module DK, USB kit, or WiRa Kit boards. Pre-built binaries and SUOTA images are included in `/wira/wira_basic_app_demo/binaries` and `binaries/wira/wira_basic_app_demo/suota_images` folders respectively.

5.2.2 Configuration options

The functionality and the performance of the demo can be customized by modifying various build-time or run-time configuration parameters.

5.2.2.1 Build-time configuration parameters

The build-time configuration parameters are defined in configuration files located in the `projects/dk_apps/wira/wira_basic_app_demo/config` folder.

5.2.2.1.1 WiRa configuration parameters defined in `wira_config.h`

The configuration parameters in this file define the behavior of the WiRa manager:

- **WiRa device role**: either Initiator or Responder. The Initiator role is enabled when the symbol WIRA_INITIATOR_ROLE is defined. The Responder role is enabled when the symbol WIRA_RESPONDER_ROLE is defined. At least one role must be defined. Both roles can be defined. The default role is defined by the symbol DEFAULT_WIRA_ROLE in file `custom_config_qspi.h` and can be changed at run time using the WiRa configuration BLE service.
- **WiRa connection parameters**: the connection parameters are defined using symbols WIRA_CONN_INTERVAL_MIN, WIRA_CONN_INTERVAL_MAX, WIRA_CONN_SLAVE_LATENCY, and WIRA_CONN_SUPERVISION_TO. A short connection interval must be defined to minimize the duration of the WiRa measurement event.
- **Keep-alive connection parameters**: if the time between WiRa measurements is long, then the system can be configured to change the connection parameters between measurements to optimize the power consumption by defining the symbol WIRA_USE_KEEPALIVE_CON_PARAMS. Keep-alive connection parameters are applied after the WiRa measurement event. WiRa connection parameters are applied before performing the next WiRa measurement. The keep-alive connections parameters are defined using symbols WIRA_CONN_KEEPALIVE_INTERVAL_MIN, WIRA_CONN_KEEPALIVE_INTERVAL_MAX, WIRA_CONN_KEEPALIVE_SLAVE_LATENCY, and WIRA_CONN_KEEPALIVE_SUPERVISION_TO.
- **Service discovery bypass**: if WIRA_USE_FIXED_CHAR_HANDLES, then the Initiator bypasses the service discovery when a connection to a Responder is established. The handles of the characteristics of the WiRa service are considered fixed and known. This reduces the time for the first WiRa measurement after the connection establishment. The handles are defined using symbols WIRA_I_RESULT_VAL_HANDLER, WIRA_R_RESULT_VAL_HANDLER, and WIRA_R_RESULT_CCC_HANDLER.
- **WiRa manager logging**: logging can be enabled by defining macro LOG_WIRA.

5.2.2.1.2 Demo configuration parameters defined in `demo_config.h`

The configuration parameters in this file define the behavior of the demo application:

- **Drop the connection after WiRa measurement**: if the symbol WIRA_DISCONNECT_AFTER_MEASUREMENT is defined, the Initiator drops the connection after the WiRa measurement. The Initiator reconnects to the device after the WiRa measurement period. If WIRA_DISCONNECT_AFTER_MEASUREMENT is not defined, the Initiator remains connected to the Responder.
- **Inactivity timeout**: if the symbol ENABLE_INACTIVITY_TIMEOUT is defined, the device is powered down after an inactivity timeout is defined by the symbol DEFAULT_WIRA_INACTIVITY_TIMEOUT in file `custom_config_qspi.h`. It can be configured using the WiRa configuration service.
- **WiRa result beaconing**: if INITIATOR_BEACON_WIRA_RESULTS is defined, then the Initiator beacons the WiRa measurement results using non-connectable advertising packets. The data in the advertising packets are described in the file `wira_beacon_readme.md` located in the `/dk_apps/wira/common/utils` project folder. The advertising interval is defined using symbols BLE_BEACON_INTERVAL_MIN_MS and BLE_BEACON_INTERVAL_MAX_MS in milliseconds.
- **Initiator scanning parameters**: the scanning parameters for the Initiator (central) role are defined using symbols BLE_SCAN_INTERVAL_MS and BLE_SCAN_WINDOW_MS in milliseconds.
- **Responder advertising parameters**: The advertising parameters for the Responder (peripheral) role are defined using symbols BLE_ADV_INTERVAL_MIN_MS and BLE_ADV_INTERVAL_MAX_MS in milliseconds.

5.2.2.1.3 Global project configuration parameters defined in file `custom_config_qspi.h`

- **Enable UART logging**: if the symbol CONFIG_RETARGET is defined, then application log messages are sent to the UART interface. The UART configuration settings are the following:
  - Baud rate: 115200
  - Data bits: 8
  - Parity: None
  - Stop bits: 1
- **Enable USB CDC logging**: if the symbol CONFIG_RETARGET_USB_CDC is defined, then application log messages are sent to the USB CDC interface.
- **Enable colorful logging**: if the symbol USE_COLORS_IN_LOGS is defined, colors are used to improve the readability of the log messages. This symbol must not be defined if the terminal application is not able to display colors.

5.2.2.2 Run-time configuration parameters

If the WiRa configuration service is enabled in file `custom_config_qspi.h`, then the WiRa Android application can be used to change the following parameters at run-time as described in Appendix E. The description of these parameters is also provided in this Appendix. The default values are defined in file custom_config_qspi.h.

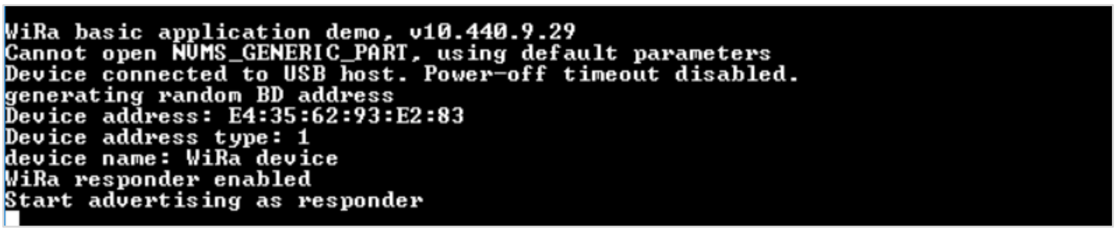
- **WiRa role**: either Initiator or Responder. The default role is defined by the symbol DEFAULT_WIRA_ROLE. The device is configured as Responder by default. This parameter has no effect if only one role has been enabled in `wira_config.h` when building the project.
- **Inactivity timeout**: the default value is 10 minutes and it is defined by the symbol DEFAULT_WIRA_INACTIVITY_TIMEOUT.
- **WiRa set number**: the default value is zero and it is defined by the symbol DEFAULT_WIRA_SET_NUM.
- **Number of WiRa events per WiRa measurement**: the default value is four and it is defined by the symbol DEFAULT_WIRA_NUM_OF_EVT.
- **WiRa measurement period**: the default value is 10 and it is defined by the symbol DEFAULT_WIRA_PERIOD_SEC.
- **SVHT scale**: the default value is 60% and it is defined by the symbol DEFAULT_WIRA_MATPEN_SVHT_SCALE.
- **Distance offset**: the default value is zero and it is defined by the symbol DEFAULT_WIRA_DISTANCE_OFFSET_CM. The distance offset is subtracted from all WiRa distance measurements.
- **Filter type**: the default value is none and it is defined by the symbol DEFAULT_WIRA_FILTER_TYPE. Sliding average or Kalman filter can be selected.

- **Kalman filter process noise**: the default value is 10% and it is defined by the symbol DEFAULT_WIRA_KALMAN_PROCESS_NOISE.
- **Kalman filter measurement noise**: the default value is 50% and it is defined by the symbol DEFAULT_WIRA_KALMAN_MEASUREMENT_NOISE.
- **RSSI threshold**: the default value is -85 dBm and it is defined by the symbol DEFAULT_WIRA_RSSI_LOW_THRESHOLD. WiRa measurements, having RSSI lower than this value, are ignored.
- **Maximum WiRa distance threshold**: the default value is 100 dm (10 m) and it is defined by the symbol DEFAULT_WIRA_DISTANCE_HIGH_THRESHOLD. WiRa measurements with a distance longer than this value are ignored.

5.2.3 Run the Demo

1. Configure the device as described in Appendix A.
2. Power up the first device.
3. Download the firmware to the device:
   - If the device is already programmed with firmware that supports SUOTA, then the firmware can be updated over the air as described in Appendix C. The SUOTA images for the supported boards (DA1469x PRO DK, DA14695 Module DK, USB Kit, and WiRa Kit) are provided in the folder `/binaries/wira/wira_basic_app_demo/suota_images`.
   - If the device does not support SUOTA, then:
     - Use the SmartSnippets Toolbox to erase the data of the whole Flash area, as described in Appendix B. This is required to delete the old partition table. A new partition table with SUOTA support is created when the demo firmware is executed for the first time.
     - Use the SmartSnippets Toolbox to program the device as described in Appendix B. The binary files for the supported boards (DA1469x PRO DK, DA14695 Module DK, USB Kit, and WiRa Kit) are provided in folder `/binaries/wira/wira_basic_app_demo/binaries`.
4. (Optional) Connect the device to the USB port of a computer and use a terminal program to get the debugging logs from the device. Configure the terminal with the following settings:
   - Baud rate: 115200
   - Data bits: 8
   - Parity: None
   - Stop bits: 1.

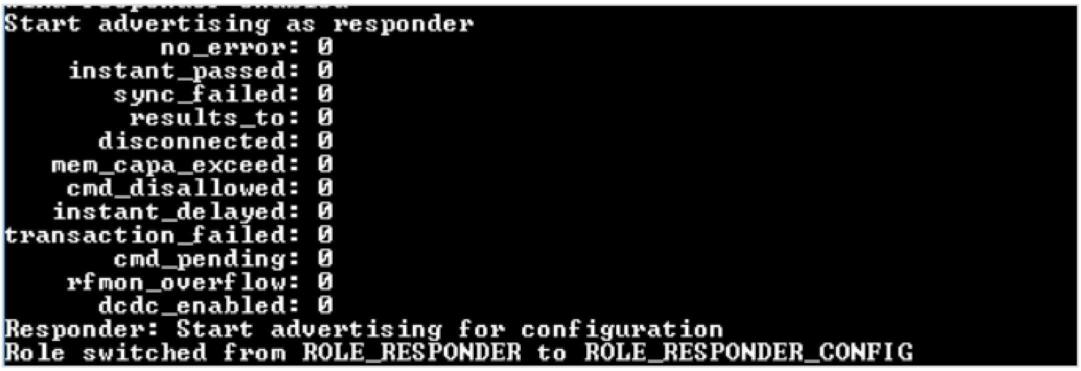
   If the device is upgraded successfully with wira_basic_app_demo, at the first reset, the output log in the terminal is as in Figure 13.



```
WiRa basic application demo, v10.440.9.29
Cannot open NVMS_GENERIC_PART, using default parameters
Device connected to USB host. Power-off timeout disabled.
generating random BD address
Device address: E4:35:62:93:E2:83
Device address type: 1
device name: WiRa device
WiRa responder enabled
Start advertising as responder
```

Figure 13. Output log at the first reset

5. Repeat the above procedure for the second device.
6. Configure the first device:
   a. To enter configuration mode, long press the button (K1) of the first device.

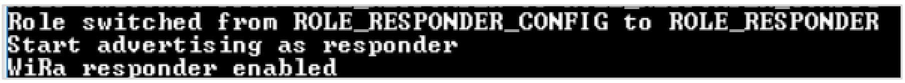


```
Start advertising as responder
        no_error: 0
    instant_passed: 0
       sync_failed: 0
        results_to: 0
      disconnected: 0
   mem_capa_exceed: 0
    cmd_disallowed: 0
   instant_delayed: 0
transaction_failed: 0
       cmd_pending: 0
     rfmon_overflow: 0
      dcdc_enabled: 0
Responder: Start advertising for configuration
Role switched from ROLE_RESPONDER to ROLE_RESPONDER_CONFIG
```

Figure 14. Configuration mode

b. The device remains in configuration for 10 seconds to allow the Android application to connect and configure the device as described in Appendix E. WiRa set number, inactivity timeout, a number of WiRa events, and WiRa measurement period can be configured.

   If you do not connect to the device using the application within 10 seconds, then the device resumes normal operation and the following message is logged to the console:

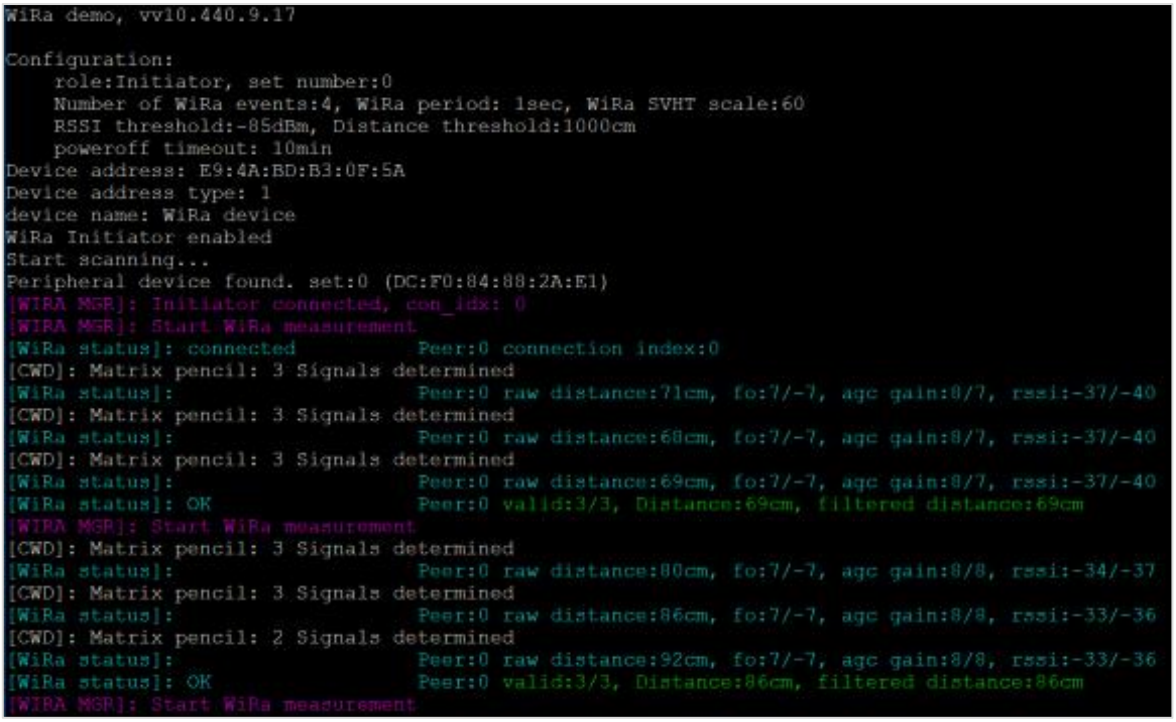


```
Role switched from ROLE_RESPONDER_CONFIG to ROLE_RESPONDER
Start advertising as responder
WiRa responder enabled
```

Figure 15. Output log when switching from Configuration to Normal mode

c. Disconnect the Android application from the device.

7. Repeat the previous step to configure the second device as Initiator. The set ID of the Initiator must match the set ID of the Responder.
8. When both devices are disconnected from the Android application, the Initiator connects to the Responder with the same set ID and distance measurement is started.
9. Results are reported to the debugging console:



```
WiRa demo, vv10.440.9.17

Configuration:
    role:Initiator, set number:0
    Number of WiRa events:4, WiRa period: 1sec, WiRa SVHT scale:60
    RSSI threshold:-85dBm, Distance threshold:1000cm
    poweroff timeout: 10min
Device address: E9:4A:BD:B3:0F:5A
Device address type: 1
device name: WiRa device
WiRa Initiator enabled
Start scanning...
Peripheral device found. set:0 (DC:F0:84:88:2A:E1)
[WIRA MGR]: Initiator connected, con_idx: 0
[WIRA MGR]: Start WiRa measurement
[WiRa status]: connected          Peer:0 connection index:0
[CWD]: Matrix pencil: 3 Signals determined
[WiRa status]:                    Peer:0 raw distance:71cm, fo:7/-7, agc gain:8/7, rssi:-37/-40
[CWD]: Matrix pencil: 3 Signals determined
[WiRa status]:                    Peer:0 raw distance:68cm, fo:7/-7, agc gain:8/7, rssi:-37/-40
[CWD]: Matrix pencil: 3 Signals determined
[WiRa status]:                    Peer:0 raw distance:69cm, fo:7/-7, agc gain:8/7, rssi:-37/-40
[WiRa status]: OK                 Peer:0 valid:3/3, Distance:69cm, filtered distance:69cm
[WIRA MGR]: Start WiRa measurement
[CWD]: Matrix pencil: 3 Signals determined
[WiRa status]:                    Peer:0 raw distance:80cm, fo:7/-7, agc gain:8/8, rssi:-34/-37
[CWD]: Matrix pencil: 3 Signals determined
[WiRa status]:                    Peer:0 raw distance:86cm, fo:7/-7, agc gain:8/8, rssi:-33/-36
[CWD]: Matrix pencil: 2 Signals determined
[WiRa status]:                    Peer:0 raw distance:92cm, fo:7/-7, agc gain:8/8, rssi:-33/-36
[WiRa status]: OK                 Peer:0 valid:3/3, Distance:86cm, filtered distance:86cm
[WIRA MGR]: Start WiRa measurement
```

Figure 16. Output log of wira_basic_demo

- WiRa manager logs are printed in purple color.
- WiRa distance results are logged in blue color.

- Distance estimations using the individual WiRa distance results are logged in green color.

Results are also beaconed by the Initiator and can be monitored using the Android application as described in Appendix E.

5.3 WiRa tracking Demo

5.3.1 Introduction

The goal of this demo is to combine RSSI and WiRa measurement data to decide if a device is close enough to another device to take a decision (for example, turn on/off a LED, open/close a door). The RSSI is used to get a rough estimation of the distance between the devices and WiRa measurements are performed to improve the distance estimation.

There are two possible implementations:

- Initiators are placed at a fixed point (for example, at a door) and the Responders simulate the users. Initiators are scanning continuously for users and start WiRa when it is required.
- Responders are placed at a fixed point (for example, at a door) and the Initiators simulate the users. Initiators are scanning continuously for doors and start WiRa when required.

The `wira_tracking_demo` is compatible with DA1469x PRO DK, DA14695 Module DK, USB Kit, and WiRa Kit boards. The hardware boards are described in Appendix A.

5.3.2 Configuration options

Under the `projects/dk_apps/wira/wira_tracking_demo/config` folder there are three configuration files: `custom_config_qspi.h`, `demo_config.h` and `wira_config.h`. If you want to change one of the defines or parameters in these files, rebuild the application and upgrade the development kit with the new image. To build an application, see Appendix D. To upgrade a device, see Appendix B (using SmartSnippets Toolbox), Appendix C (upgrade over SUOTA).

- **custom_config_qspi.h**:
  - Enable UART logging: If the symbol `CONFIG_RETARGET` is defined, then application log messages are sent to the UART interface. The UART configuration settings are the following:
    - Baud rate: 115200
    - Data bits: 8
    - Parity: None
    - Stop bits: 1.
  - Enable USB CDC logging: If the symbol `CONFIG_RETARGET_USB_CDC` is defined, then application log messages are sent to the USB CDC interface.
  - WIRA_USE_CONFIGURATION_SERVICE: if it is defined, WiRa configuration service is enabled. Then the WiRa Android application can be used to change the following parameters at run-time as described in Appendix E. There is also a description of these parameters in Appendix E. The default values for these parameters are defined also in `custom_config_qspi.h` file.
  - DEFAULT_WIRA_ROLE: the WiRa role can be either Initiator or Responder. The default value is set to WIRA_ROLE_RESPONDER.
  - DEFAULT_WIRA_INACTIVITY_TIMEOUT: after this time in minutes the device goes to sleep. The default value is set to 0, which means that the device remains forever active.
  - DEFAULT_WIRA_SET_NUM: an Initiator and a Responder with the same number can connect to each other. The default value is set to 0.
  - DEFAULT_WIRA_NUM_OF_EVT: number of WiRa events executed for each WiRa measurement. The default value is set to 4.
  - DEFAULT_WIRA_PERIOD_SEC: in this period in seconds WiRa performs DEFAULT_WIRA_NUM_OF_EVT events.
  - DEFAULT_WIRA_MATPEN_SVHT_SCALE: the default value is set to 60.
  - DEFAULT_WIRA_DISTANCE_OFFSET_CM: an offset that is deducted from the WiRa measurement. The default value is set to 0.
  - DEFAULT_WIRA_RSSI_LOW_THRESHOLD: the default value is set to -85 dBm.
  - DEFAULT_WIRA_RSSI_HIGH_THRESHOLD: the default value is set to -38 dBm.

   - DEFAULT_WIRA_DISTANCE_LOW_THRESHOLD: the default value is set to 10 dm.
   - DEFAULT_WIRA_DISTANCE_HIGH_THRESHOLD: the default value is set to 100 dm.
- **demo_config.h**: the definition and configuration parameters in this file are demo specific.
   - ENABLE_INACTIVITY_TIMEOUT: if it is defined, the device goes to sleep after a time that is defined by DEFAULT_WIRA_INACTIVITY_TIMEOUT. This timeout can be changed by the Android WiRa application at run time. The default value is set to 0 to disable inactivity timeout.
   - WIRA_ROLE_SWITCH_TIMEOUT_SEC: after a long press of button K1 the Initiator or the Responder enters the Configuration mode to change their parameters. You can connect to the development kit through the Android WiRa application within WIRA_ROLE_SWITCH_TIMEOUT_SEC seconds.
   - INITIATOR_BEACON_WIRA_RESULTS: if it is defined, the WiRa measurement results are included in Initiator's beacon. The advertising parameters used in the Initiator for beaconing the WiRa measurements are:
      - BLE_BEACON_INTERVAL_MIN_MS
      - BLE_BEACON_INTERVAL_MAX_MS
   - INITIATOR_BEACON_WIRA_DETAILS: if it is defined, raw measurements and RSSI are included in Initiator's beacon.
   - The scan parameters for the Initiator role are defined by the following symbols:
      - BLE_SCAN_INTERVAL_MS, set to 220 ms
      - BLE_SCAN_WINDOW_MS, set to 220 ms

     Both parameters are set to a low value to optimize the time that an Initiator needs to find and process a Responder.
   - The advertising parameters for the Responder are defined by the following symbols:
      - BLE_ADV_INTERVAL_MIN_MS, set to 50 ms
      - BLE_ADV_INTERVAL_MAX_MS, set to 70 ms

     The advertisement interval is set to a maximum of 70 ms to be detected by the Initiator which scans on each channel for 220 ms.
   - INFO_DEBUG_MESSAGES: if it is defined, info log messages are enabled for more logs and debugging reasons.
   - Enable colorful logging: if the symbol `USE_COLORS_IN_LOGS` is defined, colors are used to improve the readability of the log messages. If the terminal application cannot display colors, this symbol must not be defined.
- **wira_config.h**: the configuration parameters in this file define the behavior of the WiRa manager. The available parametersused by `wira_tracking_demo` are the following:
   - WIRA_RESPONDER_ROLE and WIRA_INITIATOR_ROLE: if both are defined, the development kit starts as a Responder, as it is defined by the DEFAULT_WIRA_ROLE as mentioned before. If only WIRA_RESPONDER_ROLE is defined, the device starts as a Responder and if only WIRA_INITIATOR_ROLE is defined, it starts as an Initiator.
   - WIRA_EVT_NUM: the number of WiRa events that are needed to calculate one distance result.
   - WIRA_EVT_NUM_MAX: the maximum number of WiRa events to calculate one distance result.
   - WIRA_MAX_DISTANCE_CM: if WiRa calculated distance is greater than the maximum distance, the measurement is rejected.
   - WIRA_MIN_RSSI_THRESHOLD: if the RSSI is lower than this minimum threshold, the measurement is rejected.
   - WIRA_OUTLIER_FILTER_RATIO: ratio to filter-out outliers in %. A measured distance is considered an outlier if the distance is greater than the average by this ratio.
   - WIRA_CW_OFFSET_CM: a constant offset that is deducted from the WiRa measurement.
   - WIRA_USE_CLOSEST_DISTANCE_EST: if it is defined and all measurements are considered outliers, the smallest distance from the average is reported as the WiRa measurement. If it is not defined, the average is used. In both cases, the measurement is considered invalid.
   - WIRA_USE_IFFT_DIST_CALC and WIRA_USE_MATRIX_PENCIL_DIST_CALC: these are the two methods for distance calculation. Only one can be defined. Matrix pencil distance calculation is used by default.

- WIRA_USE_KEEPALIVE_CON_PARAMS: if it is defined, the connection parameters between WiRa measurements can be changed to reduce power consumption. The keepalive parameters are applied between WiRa measurement events. The available keepalive connection parameters are:
  - WIRA_CONN_KEEPALIVE_INTERVAL_MIN
  - WIRA_CONN_KEEPALIVE_INTERVAL_MAX
  - WIRA_CONN_KEEPALIVE_SLAVE_LATENCY
  - WIRA_CONN_KEEPALIVE_SUPERVISION_TO
- The connection parameters for establishing a connection to a Responder are:
  - WIRA_CONN_INTERVAL_MIN
  - WIRA_CONN_INTERVAL_MAX
  - WIRA_CONN_SLAVE_LATENCY
  - WIRA_CONN_SUPERVISION_TO
- The scan parameters for establishing a connection are:
  - WIRA_SCAN_INTERVAL_FOR_CONNECTION
  - WIRA_SCAN_WINDOW_FOR_CONNECTION
- WIRA_SEND_RESULT_TO_RESPONDER: if it is defined, the Initiator sends the result to the Responder by writing the value to the WiRa data R result characteristic.
- WIRA_USED_FIXED_CHAR_HANDLES: if it is defined, the Initiator skips the service discovery and uses the fixed WiRa handlers for exchanging data with the Responder device.

  The handles of the characteristics of the WiRa service are considered fixed and known. This reduces the time for the first WiRa measurement after the connection establishment. The handles are:
  - WIRA_I_RESULT_VAL_HANDLER
  - WIRA_R_RESULT_VAL_HANDLER
  - WIRA_R_RESULT_CCC_HANDLER
- WIRA_CONNECTION_TO_SEC: Time within a connection between an Initiator and a Responder should be established. If this time passes, the Initiator aborts the connection.
- WIRA_DISCONNECTION_TO_SEC: time within a WiRa measurement should be performed. If it is not performed, the Initiator drops the connection.
- WIRA_MEASUREMENT_TO_MS: time within a WiRa measurement should be performed. If it is not performed, the measurement is considered invalid.
- WIRA_ENABLE_DTE_STATISTICS: if it is defined, DTE statistics are enabled.
- WIRA_DEBUG_MEASUREMENTS: if it is defined, WiRa log messages are enabled to get raw measurements and more details about WiRa.

5.3.3 Run the Demo

For the WiRa tracking demo, at least two devices are needed, one configured as Initiator and one as a Responder. Initially, the Responder starts advertising with a custom DTE Data service UUID and the set number in the advertising data. The Initiator starts scanning, looking for the DTE Data service UUID and a matching set number in the advertising reports it receives. If both the UUID and the set number match, the two devices get connected.

When testing with more than two devices, you can see how a Responder interacts with more than one Initiator. A prioritization mechanism is applied to decide when and with whom a WiRa measurement should be performed.

To run the demo, follow the steps:

1. Power up the device.
2. Download the firmware to the device:
   a. If the device is already programmed with firmware that supports SUOTA, then the firmware can be updated over the air as described in Appendix C. The SUOTA images for the supported boards (DA1469x PRO DK, DA14695 Module DK, USB Kit, and WiRa Kit) are provided in folder `/binaries/wira/wira_tracking_demo/suota_images`.
   b. If the device does not support SUOTA, then:

i. Use the SmartSnippets Toolbox to erase the data of the whole Flash area, as described in Appendix B. This is required to delete the old partition table. A new partition table with SUOTA support is created when the demo firmware is executed for the first time.

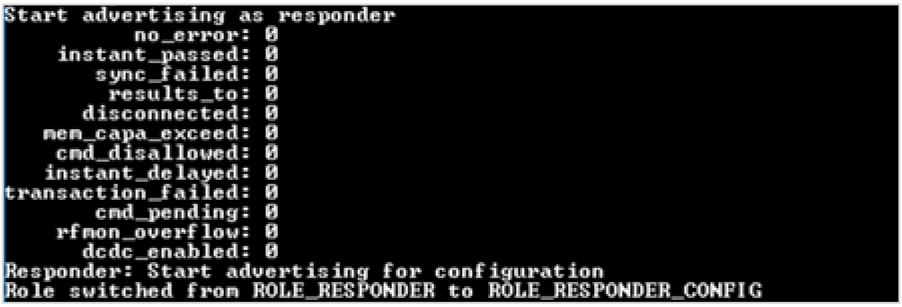
ii. Use the SmartSnippets Toolbox to program the device as described in Appendix B. The binary files for the supported boards (DA1469x PRO DK, DA14695 Module DK, USB Kit, and WiRa Kit) are provided in the folder `/binaries/wira/wira_tracking_demo/binaries`.

3. (Optional) Connect the device to the USB port of a computer and use a terminal program to get the debugging logs from the device. Configure the terminal with the following settings:

- Baud rate: 115200
- Data bits: 8
- Parity: None
- Stop bits: 1

After resetting the development kit, it starts by default as a Responder. To change the device's role to Initiator, use the Android WiRa application, following the steps below:

a. Long press the button (K1) of the first device to enter configuration mode.

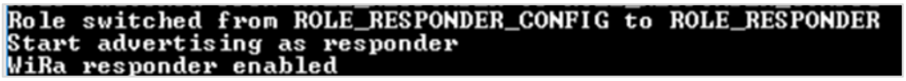


```
Start advertising as responder
          no_error: 0
    instant_passed: 0
       sync_failed: 0
        results_to: 0
      disconnected: 0
   mem_capa_exceed: 0
    cmd_disallowed: 0
   instant_delayed: 0
transaction_failed: 0
       cmd_pending: 0
    rfmon_overflow: 0
      dcdc_enabled: 0
Responder: Start advertising for configuration
Role switched from ROLE_RESPONDER to ROLE_RESPONDER_CONFIG
```

Figure 17. Configuration mode

b. The device remains in configuration mode for 10 seconds to allow the Android application to connect and configure the device as described in Appendix E. WiRa set number, inactivity timeout, a number of WiRa events, and WiRa measurement period can be configured.

If you do not connect to the device using the application within 10 seconds, then the device resumes normal operation and the following message is logged to the console:



```
Role switched from ROLE_RESPONDER_CONFIG to ROLE_RESPONDER
Start advertising as responder
WiRa responder enabled
```

Figure 18. Output log when switching from Configuration to Normal mode

c. Disconnect the Android application from the device.

Repeat the steps for all the devices. At least one Responder should exist in the demo setup.

1. Place all the Initiators at 2 meters distance from each other. This scenario simulates the implementation in which the Initiators are placed at fixed points and the Responders simulate the users, who move around.
2. Move the Responder from one Initiator to the other and when USB Kit, or DA1469x PRO DK or DA14695 Module DK are used, check the logs and LED's state on the Initiator's side. In case WiRa KITs are used, there is no LED indication because this LED is used by the LCD interface. In the case of WiRa Kit, check the LCD where the distance is displayed and when the devices are close enough the color of the first line on the display changes from white to red.
3. For each Initiator, open a UART terminal with the following configuration to get the logs:

- Baudrate: 115200
- Bits: 8
- Parity: None
- Stop bits: 1

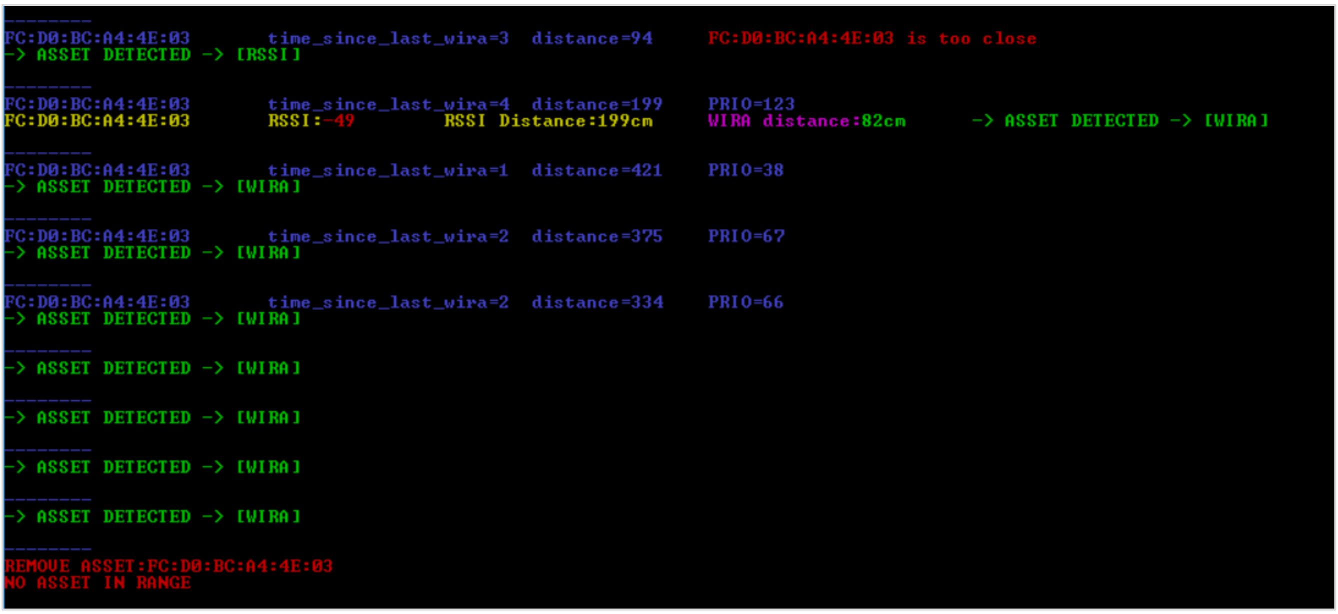
In this demo the Responders start advertising and the Initiators start scanning for Responders. Responders with RSSI lower than the DEFAULT_WIRA_RSSI_LOW_THRESHOLD (-85 dBm) are filtered out and not processed further because they are too far away. For the other Responders, the distance is calculated by the RSSI and a priority calculation is performed to decide about the WiRa measurement Responders are prioritized according to their RSSI and WiRa distance. The following rules are applied for the priority calculation:

- WiRa is not needed if the RSSI value is higher than the DEFAULT_WIRA_RSSI_THRESHOLD (-38 dBm). In this case, the Responder is within 1 m radius. (DEFAULT_WIRA_DISTANCE_LOW_THRESHOLD).
- Responders without a WiRa measurement have priority over those with a WiRa measurement. At least one WiRa measurement is needed for each Responder.
- Priority depends on the time passed since the last WiRa measurement.
- Responders with lower RSSI have higher priority.
- WiRa is performed when the calculated priority is above a threshold. In our demo, the priority threshold is set to 90.

After each WiRa measurement, a decision is taken:

- Either the LED on the development kit (DA1469x Pro DK, DA14695 Module DK, USB Kit) is turned on at the Initiator's side to indicate that a Responder is within 1 m radius or it is turned off/remains off if the Responder is farther than 1 m from the Initiator.
- For WiRa Kit, the color on the LCD changes from white to red on the Initiator's display, when a Responder is within 1 m radius.

The logs in the Initiator's terminal are shown in Figure 19.



```
---------
FC:D0:BC:A4:4E:03       time_since_last_wira=3  distance=94     FC:D0:BC:A4:4E:03 is too close
-> ASSET DETECTED -> [RSSI]
---------
FC:D0:BC:A4:4E:03       time_since_last_wira=4  distance=199    PRIO=123
FC:D0:BC:A4:4E:03       RSSI:-49        RSSI Distance:199cm     WIRA distance:82cm      -> ASSET DETECTED -> [WIRA]
---------
FC:D0:BC:A4:4E:03       time_since_last_wira=1  distance=421    PRIO=38
-> ASSET DETECTED -> [WIRA]
---------
FC:D0:BC:A4:4E:03       time_since_last_wira=2  distance=375    PRIO=67
-> ASSET DETECTED -> [WIRA]
---------
FC:D0:BC:A4:4E:03       time_since_last_wira=2  distance=334    PRIO=66
-> ASSET DETECTED -> [WIRA]
---------
-> ASSET DETECTED -> [WIRA]
---------
-> ASSET DETECTED -> [WIRA]
---------
-> ASSET DETECTED -> [WIRA]
---------
-> ASSET DETECTED -> [WIRA]
---------
REMOVE ASSET:FC:D0:BC:A4:4E:03
NO ASSET IN RANGE
```

Figure 19. Logs in the Initiator's terminal

After each scan completion, the blue line is printed, reporting the Responder's BD address, the time in seconds since the last WiRa measurement for the specific Responder, the distance calculated by the RSSI reported in scan results, and the calculated priority. The priority calculation results combine the previous two values.

At the first line, the Responder is very close to the Initiator, which means the RSSI is higher than the predefined threshold (-38 dBm). The decision is that an asset is detected by the RSSI and no priority calculation is needed.

At the next scan, the priority is 123, greater than the predefined threshold (90), so a WiRa measurement is performed. After WiRa completion, in the next line (yellow and purple) the results are reported.

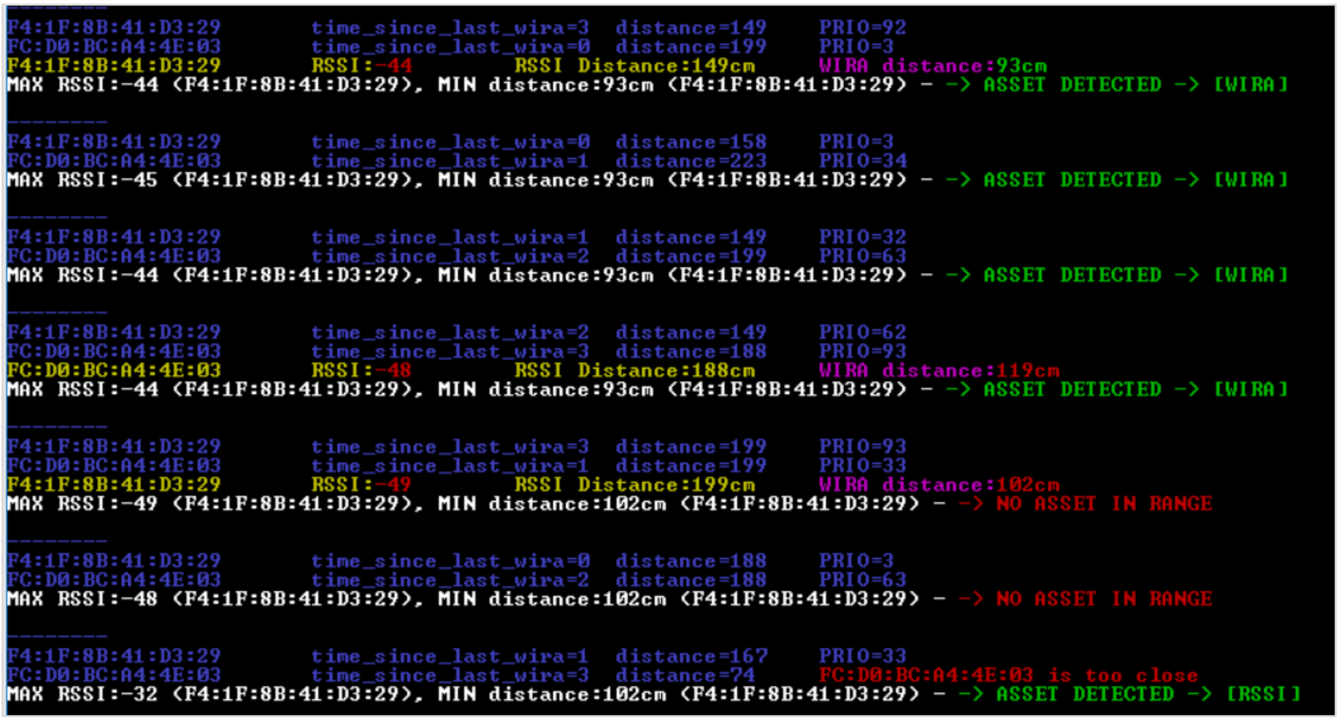
The results consist of the Responder's BD address, the RSSI, the calculated distance using the RSSI, the calculated WiRa distance, and at the end the result.

In the WiRa results line, the RSSI is lower than the predefined threshold (-38 dBm), therefore the value (-49 dBm) is printed in red color. However, the calculated WiRa distance is lower than the predefined threshold (1 m), therefore the value (82 cm) is printed in green color and the decision is taken due to WiRa measurement (distance). The asset is detected in range and consequently, the LED on the development kit (DA1469x PRO DK, DA14695 Module DK, USB Kit) is turned on or the color on the display changes to red (WiRa Kit).

At the next three scan completions, the priority is lower than 90, so no WiRa is performed. Afterward, the Responder is switched off. It is not present anymore in the scan results; therefore, no blue lines are printed.

After an absence in five consecutive scan results, the Responder is removed from the asset list and the LED on the development kit is turned off (DA1469x PRO DK, DA14695 Module DK, USB Kit) or the color on the LCD is changed to white (WiRa Kit). “No asset in range” printout appears.

If there are two or more Responders in the test setup, the output of the log changes as shown in Figure 20.



```
--------
F4:1F:8B:41:D3:29        time_since_last_wira=3  distance=149     PRIO=92
FC:D0:BC:A4:4E:03        time_since_last_wira=0  distance=199     PRIO=3
F4:1F:8B:41:D3:29        RSSI:-44        RSSI Distance:149cm      WIRA distance:93cm
MAX RSSI:-44 (F4:1F:8B:41:D3:29), MIN distance:93cm (F4:1F:8B:41:D3:29) - -> ASSET DETECTED -> [WIRA]

--------
F4:1F:8B:41:D3:29        time_since_last_wira=0  distance=158     PRIO=3
FC:D0:BC:A4:4E:03        time_since_last_wira=1  distance=223     PRIO=34
MAX RSSI:-45 (F4:1F:8B:41:D3:29), MIN distance:93cm (F4:1F:8B:41:D3:29) - -> ASSET DETECTED -> [WIRA]

--------
F4:1F:8B:41:D3:29        time_since_last_wira=1  distance=149     PRIO=32
FC:D0:BC:A4:4E:03        time_since_last_wira=2  distance=199     PRIO=63
MAX RSSI:-44 (F4:1F:8B:41:D3:29), MIN distance:93cm (F4:1F:8B:41:D3:29) - -> ASSET DETECTED -> [WIRA]

--------
F4:1F:8B:41:D3:29        time_since_last_wira=2  distance=149     PRIO=62
FC:D0:BC:A4:4E:03        time_since_last_wira=3  distance=188     PRIO=93
FC:D0:BC:A4:4E:03        RSSI:-48        RSSI Distance:188cm      WIRA distance:119cm
MAX RSSI:-44 (F4:1F:8B:41:D3:29), MIN distance:93cm (F4:1F:8B:41:D3:29) - -> ASSET DETECTED -> [WIRA]

--------
F4:1F:8B:41:D3:29        time_since_last_wira=3  distance=199     PRIO=93
FC:D0:BC:A4:4E:03        time_since_last_wira=1  distance=199     PRIO=33
F4:1F:8B:41:D3:29        RSSI:-49        RSSI Distance:199cm      WIRA distance:102cm
MAX RSSI:-49 (F4:1F:8B:41:D3:29), MIN distance:102cm (F4:1F:8B:41:D3:29) - -> NO ASSET IN RANGE

--------
F4:1F:8B:41:D3:29        time_since_last_wira=0  distance=188     PRIO=3
FC:D0:BC:A4:4E:03        time_since_last_wira=2  distance=188     PRIO=63
MAX RSSI:-48 (F4:1F:8B:41:D3:29), MIN distance:102cm (F4:1F:8B:41:D3:29) - -> NO ASSET IN RANGE

--------
F4:1F:8B:41:D3:29        time_since_last_wira=1  distance=167     PRIO=33
FC:D0:BC:A4:4E:03        time_since_last_wira=3  distance=74      FC:D0:BC:A4:4E:03 is too close
MAX RSSI:-32 (F4:1F:8B:41:D3:29), MIN distance:102cm (F4:1F:8B:41:D3:29) - -> ASSET DETECTED -> [RSSI]
```

Figure 20. Output log with two responders

In the above log (Figure 20), there are two Responders in range. After scan completion, there is one blue line for each Responder and at the next line (white color) the maximum RSSI and the last minimum calculated WiRa distance of the two Responders are printed, to show which Responder triggered the decision.

For example, at the first run, the priority of F4:1F:8B:41:D3:29 is 92, higher than 90 and a WiRa measurement is performed with this Responder. The calculated WiRa distance of F4:1F:8B:41:D3:29 is 93 cm, lower than the predefined threshold (1 m), so the asset is detected by WiRa. The LED of the development kit (DA1469x PRO DK, DA14695 Module DK, USB Kit) is turned on or the color on the LCD changes to red (WiRa Kit).

The "MIN distance" (white line) remains 93 cm for the next three scans. At the third scan the priority of FC:D0:BC:A4:4E:03 is 93, so a WiRa is performed with this Responder, but the calculated distance is more than 1 meter so the LED remains on due to the distance of the other Responder.

At the next scan, a new WiRa measurement is performed with F4:1F:8B:41:D3:29 due to priority, but both Responders are not in the predefined range, so the result is “no asset in range” and the development kit’s LED is turned off (DA1469x PRO DK, DA14695 Module DK, USB Kit), display color is changed to white (WiRa Kit).

At the last scan, the RSSI of FC:D0:BC:A4:4E:03 Responder is -32 dBm, higher than the predefined threshold (-38 dBm), so the asset is detected by its RSSI, no WiRa is needed. LED is turned on (DA1469x PRO DK, DA14695 Module DK, USB Kit), display color changes to red (WiRa Kit).

5.3.4 Detailed software description

5.3.4.1 Basic structures

When the Initiator receives the BLE_EVT_GAP_ADV_REPORT or the BLE_EVT_GAP_EXT_ADV_REPORT message, the code collects the data from the advertisement report and saves them in **scan_results_t** structure.

```
typedef struct scan_results_s
{
       struct scan_results_s *next;
       scanned_dev_entry dev
} scan_results_t;
```

It is a linked list with the structure **scanned_dev_entry** as the member that contains each entry of the advertisement report.

```
typedef struct

{
 bd_address_t    addr;
 int16_t              rssi;
 uint8_t              count;
} scanned_dev_entry;
```

where:

- **addr** is the BD address of the device.
- **rssi** is the reported RSSI for each device in the advertisement report.
- **count** is a variable that indicates how many times a device appears in one advertisement report.

When the Initiator receives the BLE_EVT_GAP_SCAN_COMPLETED or BLE_EVT_GAP_EXT_SCAN_COMPLETED message, the process of the scanned devices begins. The data is processed and finally, a list with available assets is created and used to perform WiRa or not and take decisions like turn on/off a LED, open/close a door, and so on.

There is the following structure:

```
typedef struct
{
        void*              scan_list;
        void*              unique_scan_entries_list;
        void *             assets_list;
 OS_TIMER    scan_duration_tmr;
} tracking_env_t;
```

The member **scan_duration_tmr** is a timer that stops a scan after a period of 3*scan_interval milliseconds.

All the collected data from an advertisement report is saved in the **scan_list** member. Then another list is created, member **unique_scan_entries_list**, that contains a unique entry for each device that was in the scan results. This unique entry consists of the BD address of the device and the average RSSI. The creation of a list with all the candidate assets for WiRa measurement is the next step. This list is the member **assets_list**.

All the assets are saved in the following linked list:

```
typedef struct assets_s
{
struct assets_s *next;
tracking_info_dev_t dev;
} assets_t;
```

Where each asset/device is described by the **tracking_info_dev_t** structure which has the following members:

```
typedef struct
{
    bd_address_t         addr;
int8_t               rssi;
int8_t               rssi_vals[RSSI_CIRC_BUFF_MAX];
uint16_t             distance_by_rssi;
uint8_t              updated_last_scan;
   uint32_t              t_wira_s;
   uint8_t               num_of_wira;
int16_t              distance;
uint16_t             prio;
bool                 decision;
uint8_t              head;
uint8_t              tail;
} tracking_info_dev_t;
```

where:

- **addr** is the BD address of the asset.
- **rssi** is the average RSSI of the asset in one advertisement report.
- **rssi_vals** is a circular buffer that keeps the last RSSI values of the asset. How many RSSI values are kept is defined by RSSI_CIRC_BUFF_MAX.
- **distance_by_rssi** is the calculated distance in cm using the RSSI member. It is needed for the priority calculation **updated_last_scan** to be set to 1 when an asset was present in the last scan results.
- **t_wira_s** time in seconds since the last WiRa measurement for the asset. It is needed for the priority calculation.
- **num_of_wira** is the total number of performed WiRa measurements.
- **distance** is the calculated distance in cm by WiRa measurement.
- **prio** is the calculated priority. If a WiRa measurement is performed, depends on the priority.
- **decision** is a boolean variable which is set to false after a WiRa measurement and to true to perform a WiRa measurement.

Appendix A Development Boards

A.1 DA1469x PRO Development Kit

A.1.1 Description

The DA1469x PRO Development Kit hardware is described in Ref. [4].

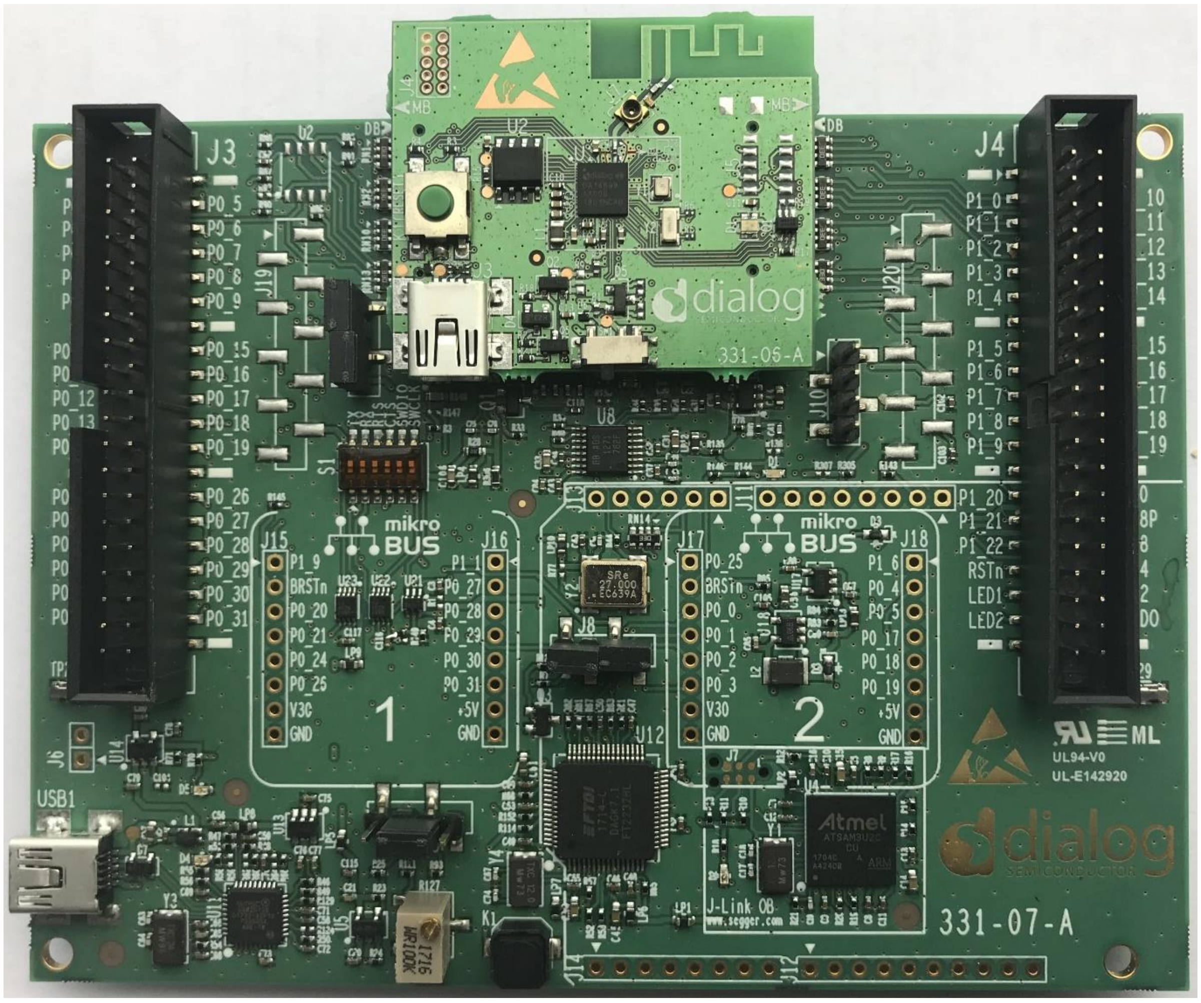


Figure 21. DA14695 PRO DK

A.1.2 Buttons and switches

The available buttons are:

- K2 (Reset button): Hardware resets the DA1469x daughter board.
- K1: If the button is pressed momentarily, it prints WiRa statistics on the terminal (if WIRA_DEBUG_MEASUREMENTS is defined in `wira_config.h`). If the button is pressed for 2 seconds, PRO-DK enters the Configuration mode either it is configured as a Responder or as an Initiator. It advertises for 10 seconds, within which you can connect to the device using the WiRa Android application to change configuration parameters.

A.1.3 Connecting hardware and powering on

NOTE
Do not change any of the jumper positions or remove/misalign the daughterboard.

To flash the application firmware image, follow the instructions in Appendix B (using SmartSnippets Toolbox), Appendix C (SUOTA) first. The available binaries that are used by Toolbox for DA1469x PRO DK are located under the `binaries/wira/<project_name>/binaries` folder and their filenames end in `Pro_DK_macronix_sst.bin`. The proper images that are used by SUOTA for DA1469x PRO DK are located under the `binaries/wira/<project_name>/suota_images` folder and their filenames end in `Pro_DK.img`. The boards are powered through USB. To power it on, use the provided USB cable to connect the DA1469x PRO DK to a laptop or desktop.

When the DA1469x PRO DK is to be connected to a power adapter or power bank, you need to solder a header on J6 and install a jumper there to bypass the enumeration with a USB host.

After power-on through USB, the DA1469x PRO board programmed with one of the three available applications enters Scanning or Advertising mode based on the configuration of the non-volatile role parameter.

A.2 DA1469x USB Kit

A.2.1 Description

The DA1469x USB Kit hardware is described in Ref. [5]. In case an LCD monitor is attached to the USB kit, the device is called DA1469x WiRa Kit. Without the LCD it is called a USB Kit. For all the demo applications there are the corresponding Build configurations for each USB Kit.

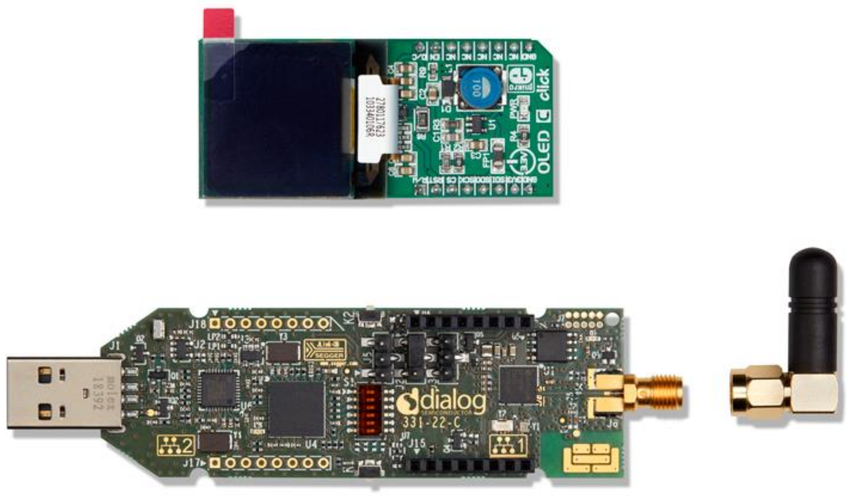


Figure 22. DA14695 Wireless Ranging Kit (WiRa Kit) hardware components with external antenna

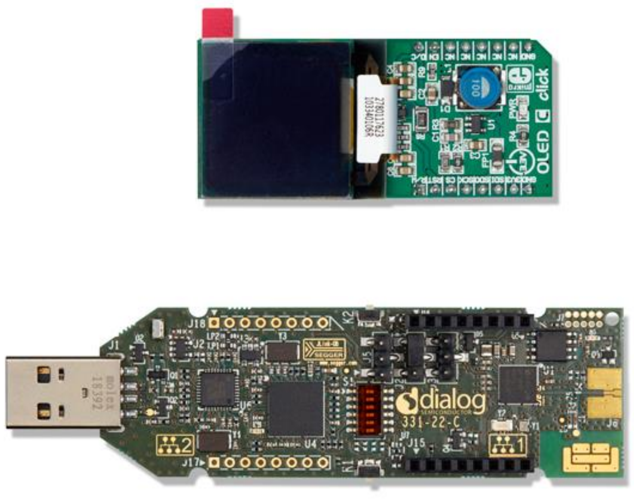


Figure 23. DA14695 Wireless Ranging Kit (WiRa Kit) hardware components with ZOR antenna

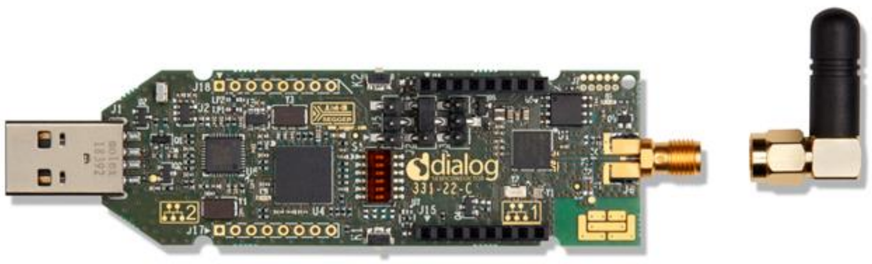


Figure 24. DA14695 USB Kit hardware components with external antenna

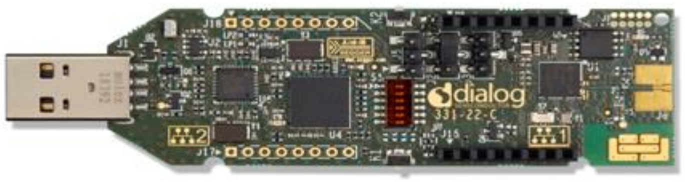


Figure 25. DA14695 USB Kit components with ZOR antenna

A.2.2 Buttons and switches

The available buttons are:

- K2 (Reset button): hardware resets the USB Kit.
- K1: if the button is pressed momentarily, it prints WiRa statistics on the terminal (if WIRA_DEBUG_MEASUREMENTS is defined in `wira_config.h`). If the button is pressed for 2 seconds, the USB Kit enters the Configuration mode either it is configured as Responder or as an Initiator. It advertises for 10 seconds, within which you can connect to the device through the Android application to change configuration parameters. In this Configuration mode, it can also be accessed through SUOTA mobile application to perform Firmware upgrade.

A.2.3 LCD support

USB Kit supports the OLED C click display connected to the mikroBUS socket.

The following hardware modification is required on the DA1469x USB Kit board:

- Add sockets to J17 and J18 for OLED C Click Display to be clicked. Hardware modifications are already applied in-house for all delivered DA1469x Wireless Ranging development kits.

A.2.4 Connecting hardware and powering on

NOTE
Do not change any of the jumper positions or remove/misalign the daughterboard.

To flash the application firmware image, follow the instructions in Appendix B (to use SmartSnippets Toolbox) or Appendix C (to use SUOTA), first. The available binaries that can be used by toolbox for DA1469x USB Kit and WiRa Kit are located under the `binaries/wira/<project_name>/binaries` folder and their filenames end in `USB_DK_macronix_sst.bin` and `WIRA_KIT_macronix_sst.bin` respectively. The proper images that can be used by SUOTA for DA1469x USB Kit and WiRa Kit are located under the `binaries/wira/<project_name>/suota_images` folder and their filenames end in `USB_DK.img` and `WIRA_KIT.img` respectively. The boards are powered through USB. To power it on, use the provided USB cable to connect the DA1469x USB Kit to a laptop or desktop.

When the DA1469x USB Kit is to be connected to a power adapter or power bank, diode D9 near K2 must be removed.

After power-on through USB, the DA1469x USB Kit programmed with one of the three available applications enters Scanning or Advertising mode based on the configuration of the non-volatile role parameter.

A.3 DA14695 Module Development Kit

A.3.1 Description

The DA14695 Module Development Kit hardware is described in Ref. [6].

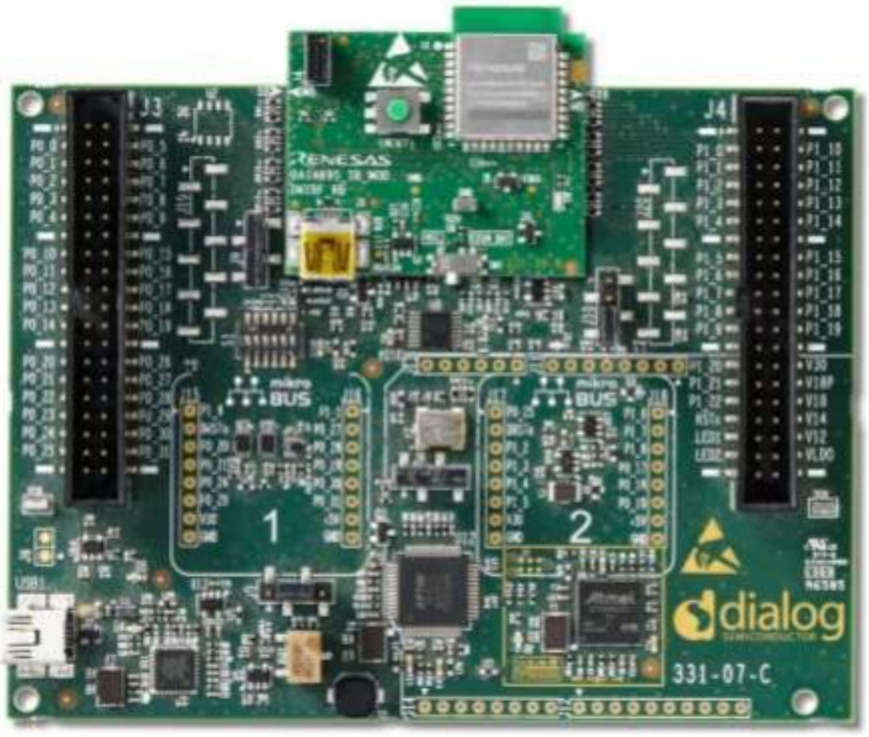


Figure 26. DA14695 Smartbond Module Development Kit

A.3.2 Buttons and switches

The available buttons are:

- Reset: hardware resets the DA14695 module daughter board.
- K1: if the button is pressed momentarily, it prints WiRa statistics on the terminal (if WIRA_DEBUG_MEASUREMENTS is defined in `wira_config.h`). If the button is pressed for 2 seconds, DA14695 Module DK enters the configuration mode either it is configured as a Responder or as an Initiator. It advertises for 10 seconds, within which you can connect to the device using the WiRa Android application to change configuration parameters.

A.3.3 Connecting hardware and powering on

NOTE
Do not change any of the jumper positions or remove/misalign the daughterboard.

To flash the application firmware image, follow the instructions in Appendix B (using SmartSnippets Toolbox), Appendix C (SUOTA) first. The available binaries that are used by Toolbox for DA14695 module DK are located under the `binaries/wira/<project_name>/binaries` folder and their filenames end in `Pro_DK_puya_sst.bin`. The proper images that are used by SUOTA for DA14695 module DK are located under the `binaries/wira/<project_name>/suota_images` folder and their filenames end in `Pro_DK.img`. The boards are powered through USB. To power it on, use the provided USB cable to connect the DA14695 module DK to a laptop or desktop.

When the DA14695 module DK is to be connected to a power adapter or power bank, you need to solder a header on J6 and install a jumper there to bypass the enumeration with a USB host.

After power-on through USB, the DA14695 module DK board programmed with one of the three available applications enters Scanning or Advertising mode based on the configuration of the non-volatile role parameter.

Appendix B Upgrade Development Kits Using SmartSnippets Toolbox

To upgrade a development kit using SmartSnippets Toolbox:

1. Download and install to your PC/laptop the latest version of SmartSnippets Toolbox (**v5.0.20** or **higher**) from: https://www.renesas.com/eu/en/software-tool/smartbond-development-tools.
2. Plug the development kit into the USB port of the PC/laptop, and then open the SmartSnippets Toolbox.
3. Select the device and the communication interface (JTAG or UART).
4. Go to **Board** > **Device**, and then select **DA1469x-00**.

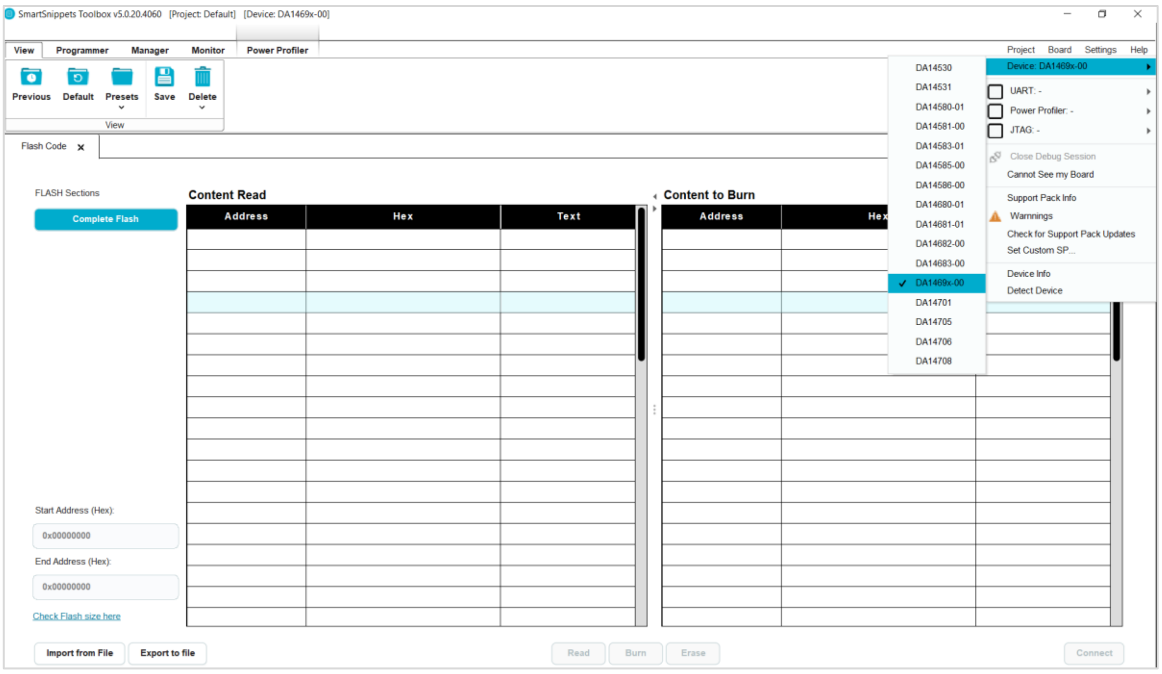


Figure 27. Select the hardware device

5. Select the **JTAG** option.

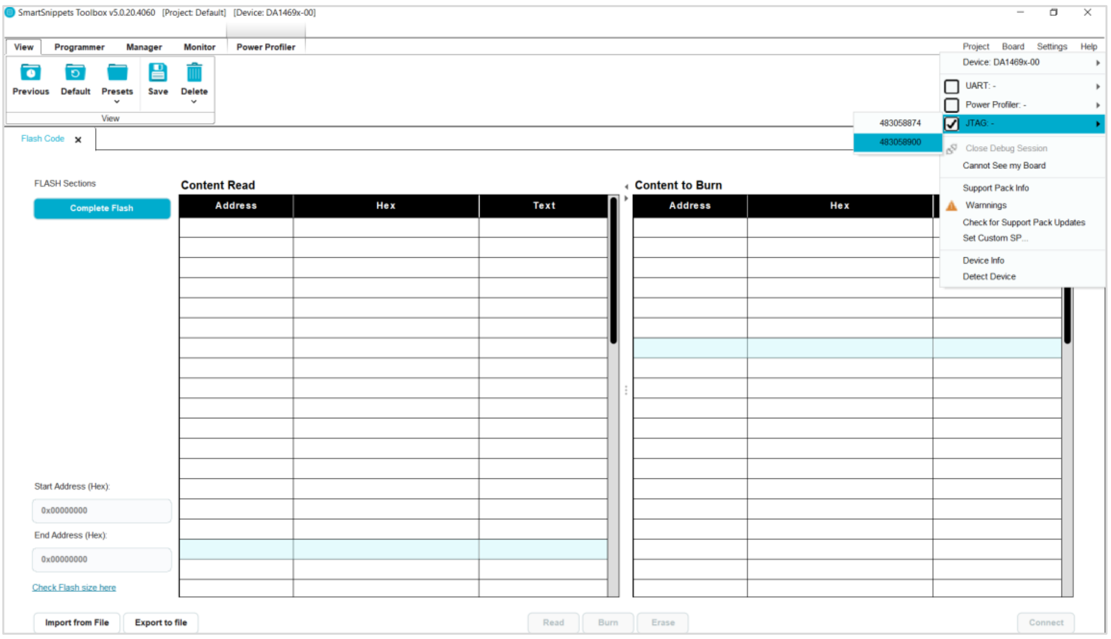


Figure 28. Select the communication way

6. To upgrade the development kit, go to **Programmer** > **Flash Code**.

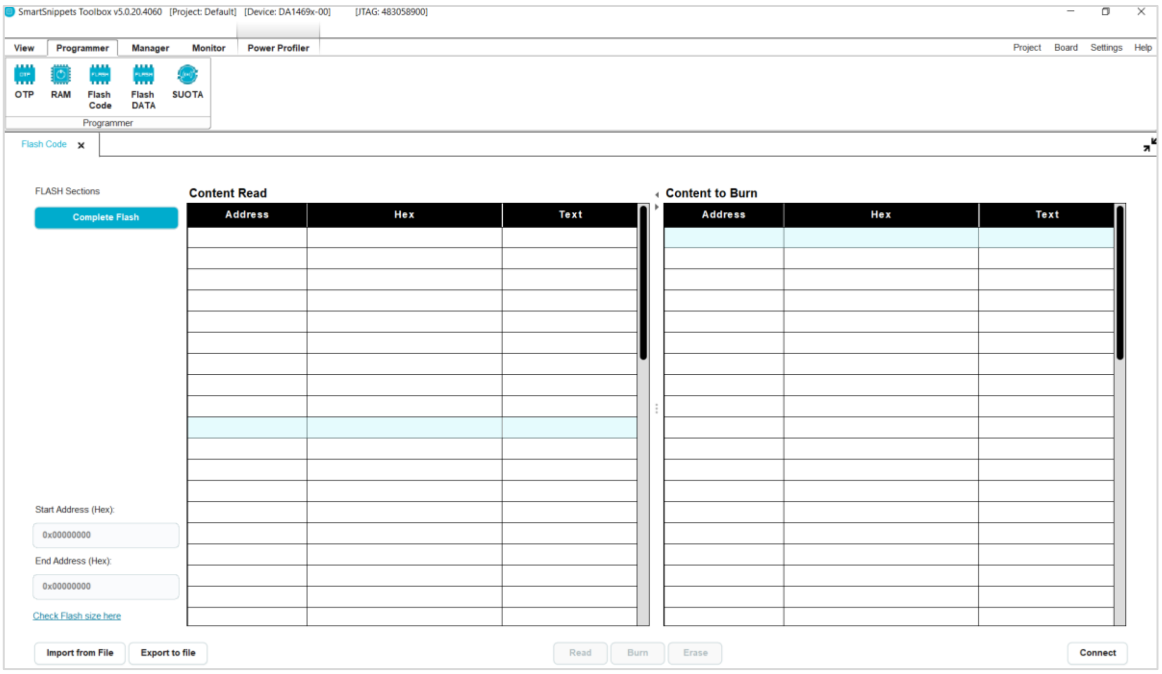
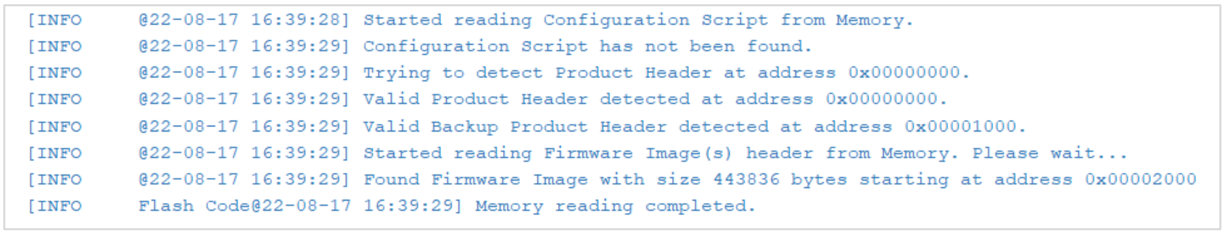


Figure 29. Flash code initial screen

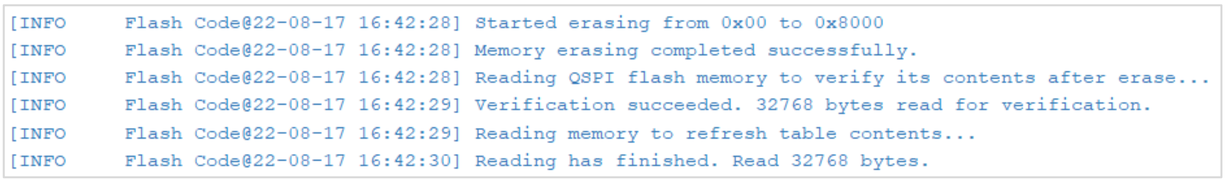
7. Click **Connect** and wait until that Toolbox is connected successfully to the device. After a successful connection, in the **Log** window, the messages as in Figure 30 appear.



```
[INFO    @22-08-17 16:39:28] Started reading Configuration Script from Memory.
[INFO    @22-08-17 16:39:29] Configuration Script has not been found.
[INFO    @22-08-17 16:39:29] Trying to detect Product Header at address 0x00000000.
[INFO    @22-08-17 16:39:29] Valid Product Header detected at address 0x00000000.
[INFO    @22-08-17 16:39:29] Valid Backup Product Header detected at address 0x00001000.
[INFO    @22-08-17 16:39:29] Started reading Firmware Image(s) header from Memory. Please wait...
[INFO    @22-08-17 16:39:29] Found Firmware Image with size 443836 bytes starting at address 0x00002000
[INFO    Flash Code@22-08-17 16:39:29] Memory reading completed.
```

Figure 30. Toolbox log after successful connection with hardware board

8. To erase the flash, click **Erase**. After completion, check the Log file to see the messages as in Figure 31.



```
[INFO    Flash Code@22-08-17 16:42:28] Started erasing from 0x00 to 0x8000
[INFO    Flash Code@22-08-17 16:42:28] Memory erasing completed successfully.
[INFO    Flash Code@22-08-17 16:42:28] Reading QSPI flash memory to verify its contents after erase...
[INFO    Flash Code@22-08-17 16:42:29] Verification succeeded. 32768 bytes read for verification.
[INFO    Flash Code@22-08-17 16:42:29] Reading memory to refresh table contents...
[INFO    Flash Code@22-08-17 16:42:30] Reading has finished. Read 32768 bytes.
```

Figure 31. Toolbox log after successful flash erase

9. If you want to erase the whole flash, to delete the partition table also, do the following:

   a. Check the flash size by clicking **Check Flash size here**.

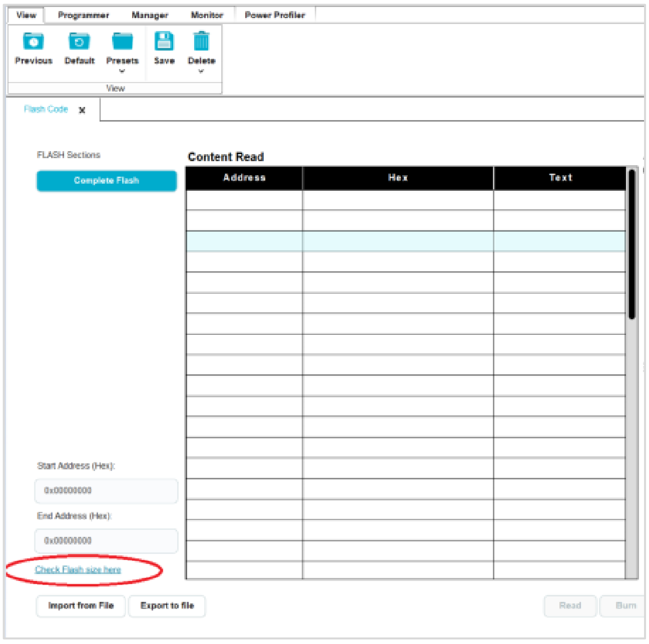


Figure 32. Check flash size option

A dialog appears with the flash size. In this case, the flash size is 4 MB.

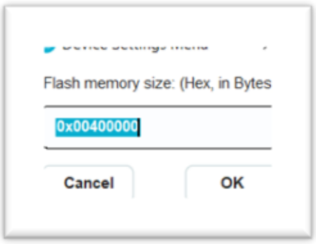
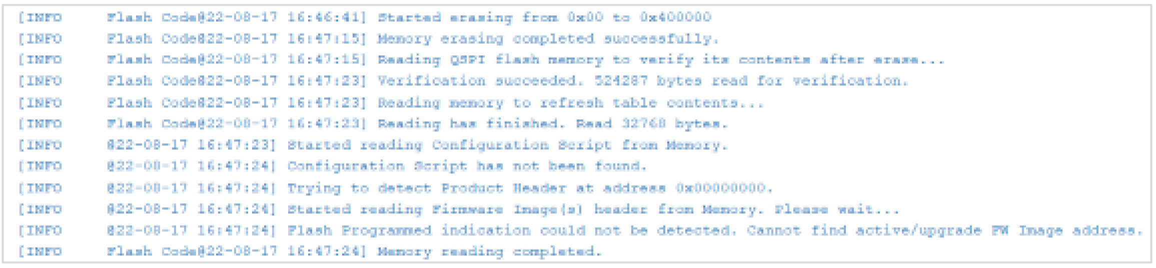


Figure 33. Dialog flash memory size

b. Click **OK** and the End Address is filled automatically with the flash size. Then click **Erase**. The erase procedure takes a few seconds.

In the **Log** window, the messages as in Figure 34 appear.

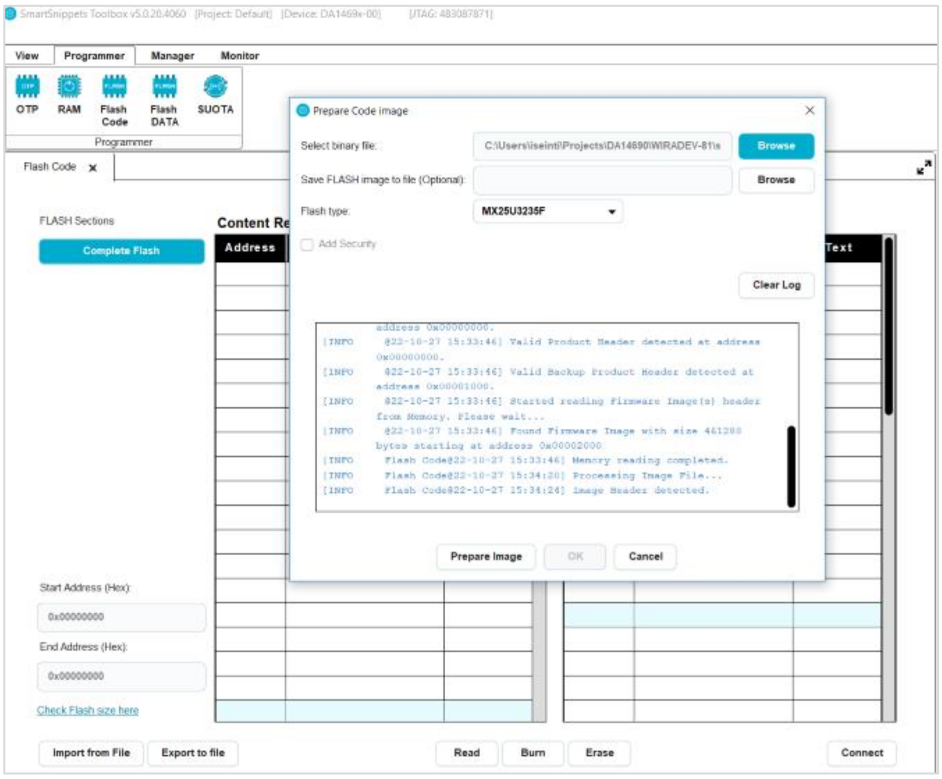


```
[INFO     Flash Code@22-08-17 16:46:41] Started erasing from 0x00 to 0x400000
[INFO     Flash Code@22-08-17 16:47:15] Memory erasing completed successfully.
[INFO     Flash Code@22-08-17 16:47:15] Reading QSPI flash memory to verify its contents after erase...
[INFO     Flash Code@22-08-17 16:47:23] Verification succeeded. 524287 bytes read for verification.
[INFO     Flash Code@22-08-17 16:47:23] Reading memory to refresh table contents...
[INFO     Flash Code@22-08-17 16:47:23] Reading has finished. Read 32768 bytes.
[INFO     @22-08-17 16:47:23] Started reading Configuration Script from Memory.
[INFO     @22-08-17 16:47:24] Configuration Script has not been found.
[INFO     @22-08-17 16:47:24] Trying to detect Product Header at address 0x00000000.
[INFO     @22-08-17 16:47:24] Started reading Firmware Image(s) header from Memory. Please wait...
[INFO     @22-08-17 16:47:24] Flash Programmed indication could not be detected. Cannot find active/upgrade FW Image address.
[INFO     Flash Code@22-08-17 16:47:24] Memory reading completed.
```

Figure 34. Toolbox log after erasing successfully whole flash memory

10. To burn the desired image to flash, click **Import from file**, and a dialog appears. Select the location where the desired file is saved by clicking **Browse**.

    All the images are located under the `binaries/wira/<demo_name>/binaries` folder.



Select binary file: C:\Users\iseint\Projects\DA14690\WIRADEV-81\s

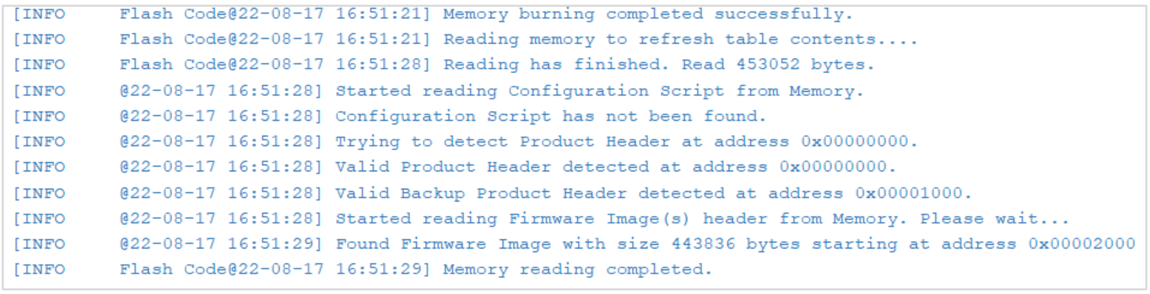
Save FLASH image to file (Optional):

Flash type: MX25U3235F

```
         address 0x00000000.
[INFO     @22-10-27 15:33:46] Valid Product Header detected at address
         0x00000000.
[INFO     @22-10-27 15:33:46] Valid Backup Product Header detected at
         address 0x00001000.
[INFO     @22-10-27 15:33:46] Started reading Firmware Image(s) header
         from Memory. Please wait...
[INFO     @22-10-27 15:33:46] Found Firmware Image with size 461288
         bytes starting at address 0x00002000
[INFO     Flash Code@22-10-27 15:33:46] Memory reading completed.
[INFO     Flash Code@22-10-27 15:34:20] Processing Image File...
[INFO     Flash Code@22-10-27 15:34:26] Image Header detected.
```

Figure 35. Dialog to choose binary to burn the flash memory

11. Click **OK**, and then click **Burn**.
12. In the **Log** window, check for the messages as in Figure 36.



```
[INFO     Flash Code@22-08-17 16:51:21] Memory burning completed successfully.
[INFO     Flash Code@22-08-17 16:51:21] Reading memory to refresh table contents....
[INFO     Flash Code@22-08-17 16:51:28] Reading has finished. Read 453052 bytes.
[INFO     @22-08-17 16:51:28] Started reading Configuration Script from Memory.
[INFO     @22-08-17 16:51:28] Configuration Script has not been found.
[INFO     @22-08-17 16:51:28] Trying to detect Product Header at address 0x00000000.
[INFO     @22-08-17 16:51:28] Valid Product Header detected at address 0x00000000.
[INFO     @22-08-17 16:51:28] Valid Backup Product Header detected at address 0x00001000.
[INFO     @22-08-17 16:51:28] Started reading Firmware Image(s) header from Memory. Please wait...
[INFO     @22-08-17 16:51:29] Found Firmware Image with size 443836 bytes starting at address 0x00002000
[INFO     Flash Code@22-08-17 16:51:29] Memory reading completed.
```

Figure 36. Toolbox log after successful flash burning

13. Press the Reset button (K2 button) on the device. The device starts with the upgraded image.

Appendix C Upgrade Development Kits over SUOTA

To upgrade a device using SUOTA, the current flashed image should support SUOTA.

⚠	**IMPORTANT**
wira_eval_demo does not support SUOTA. To change from wira_eval_demo to another demo, the procedure described in Appendix B must be followed.	

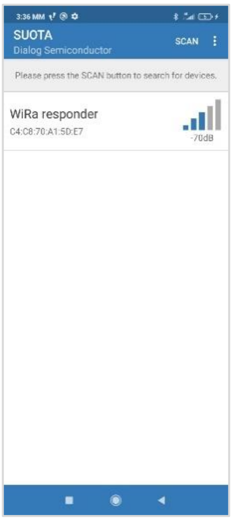
To upgrade a development kit using the SUOTA service, execute the following steps:

1. From Play Store, download the Renesas SUOTA (a.k.a. Dialog SUOTA) app and install it on an Android mobile phone.
2. On the Android phone, a folder named Suota is created. Copy the desired image in this folder, that is located under the binaries/wira/<project_name>/suota_images folder.
   The file extension of all SUOTA image files is .img.

   For example, to upgrade the development kit with the wira_tracking_demo image, copy the file named wira_tracking_demo_release_xx.img to the Suota folder of the mobile phone. Where xx is:

   - Pro_DK: if the Pro DK or DA14695 Module DK will be upgraded.
   - USB_DK: if the USB kit without the LCD will be upgraded.
   - WIRA_KIT: If the USB kit with the LCD connected will be upgraded.

   To upgrade the development kit with the wira_basic_app_demo image, copy the file named wira_basic_app_demo_release_xx.img to the Suota folder of the mobile phone.
3. Plug the development kit into the USB port of the PC.
4. Enable Bluetooth, Location services, and then open the SUOTA app on the mobile phone.

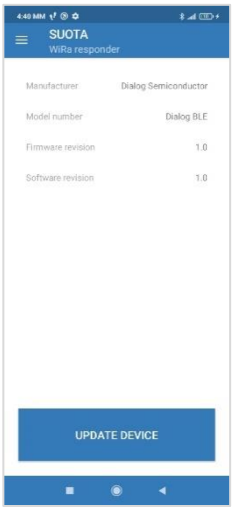


Please press the SCAN button to search for devices.

WiRa responder
C4:C8:70:A1:5D:E7
-70dB

Figure 37. Initial screen of SUOTA application

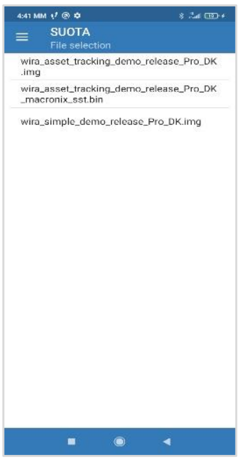
5. Connect to the device that needs to be upgraded.


SUOTA
WiRa responder

Manufacturer	Dialog Semiconductor
Model number	Dialog BLE
Firmware revision	1.0
Software revision	1.0

Figure 38. Screen with the selected device for update

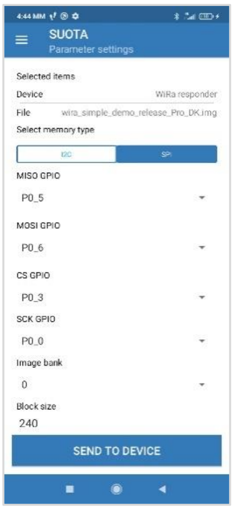
6. Tap **Update Device** and then select the desired image.


SUOTA
File selection

wira_asset_tracking_demo_release_Pro_DK.img
wira_asset_tracking_demo_release_Pro_DK_macronix_sst.bin
wira_simple_demo_release_Pro_DK.img

Figure 39. Screen to select the desired image to update the device

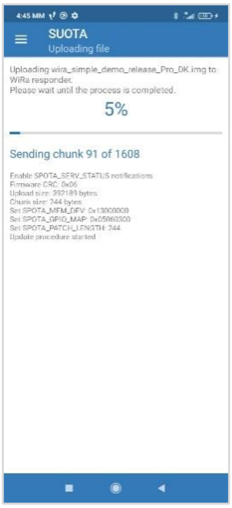
7. Tap **Send to Device**.


SUOTA
Parameter settings

Selected items
Device: WiRa responder
File: wira_simple_demo_release_Pro_DK.img
Select memory type: I2C [✓ SPI]
MISO GPIO: P0_5
MOSI GPIO: P0_6
CS GPIO: P0_3
SCK GPIO: P0_0
Image bank: 0
Block size: 240

Figure 40. Screen to configure the communication way for update

8. The upgrade procedure starts automatically. It might take some seconds.



SUOTA
Uploading file

Uploading wira_simple_demo_release_Pro_DK.img to WiRa responder.
Please wait until the process is completed.

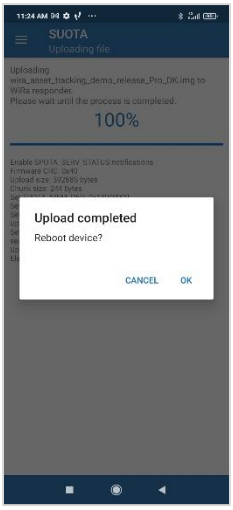
5%

Sending chunk 91 of 1608

```
Enable SPOTA_SERV_STATUS notifications
Firmware CRC: 0x06
Upload size: 392189 bytes
Chunk size: 244 bytes
Set SPOTA_MEM_DEV: 0x13000000
Set SPOTA_GPIO_MAP: 0x05060300
Set SPOTA_PATCH_LENGTH: 244
Update procedure started
```

Figure 41. SUOTA procedure started

The upgrade procedure finishes with success if the dialog as in Figure 42 appears. Tap **OK** and close the Android application. The device reboots and starts with the new image.



SUOTA
Uploading file

Uploading wira_asset_tracking_demo_release_Pro_DK.img to WiRa responder.
Please wait until the process is completed.

100%

```
Enable SPOTA_SERV_STATUS notifications
Firmware CRC: 0x40
Upload size: 392885 bytes
Chunk size: 244 bytes
```

Upload completed

Reboot device?

Figure 42. Successful SUOTA procedure

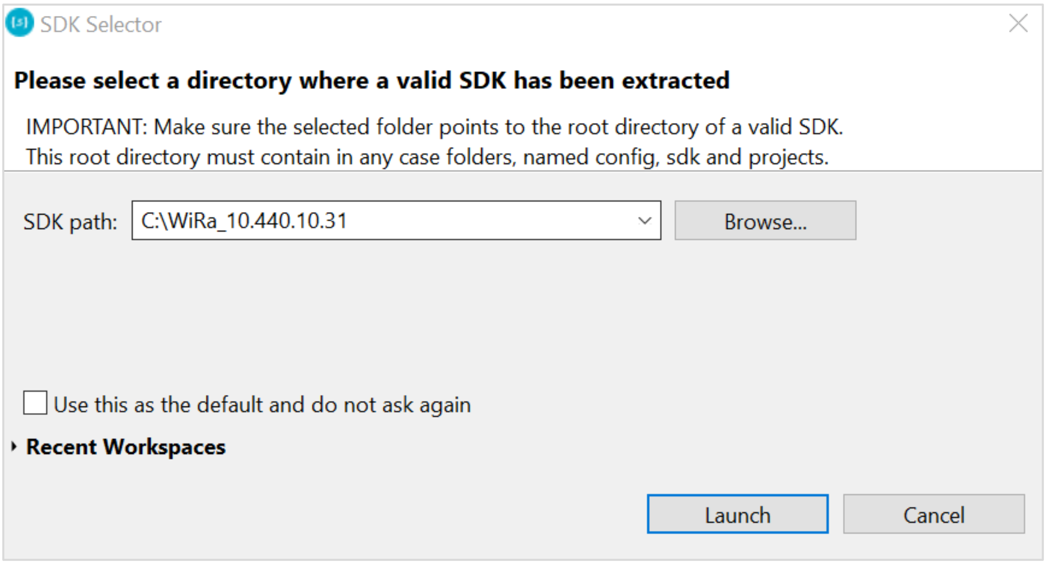
Appendix D Import, Build, and Burn the WiRa Demo Applications

There are three WiRa demo applications available:

- wira_eval_demo
- wira_basic_app_demo
- and wira_tracking_demo.

For each demo, there are prebuilt binaries and images located under `binaries/wira/<demo_app_name>/binaries` and `binaries/wira/<demo_app_name>/suota_images` folders. The binaries in this folder are updated when a project is rebuilt. To import a project, execute the steps:

1. Unzip the WiRa SDK release zip file (for example, WiRa_10.440.10.31.zip) into a known location on your drive.
2. All software development is based on SmartSnippets™ Studio. Follow the setup instructions in Section 1 of Ref. [2] for a fresh installation.
3. After SmartSnippets Studio installation, launch the program. On the first screen, enter the path of the folder where the WiRa SDK is located using the **Browse** button, and then click **Launch**.


SDK Selector

Please select a directory where a valid SDK has been extracted

IMPORTANT: Make sure the selected folder points to the root directory of a valid SDK.
This root directory must contain in any case folders, named config, sdk and projects.

SDK path: C:\WiRa_10.440.10.31

Figure 43. SDK location path selection

4. In the **Tools** section of the welcome screen, select IDE. The regular Eclipse environment view now appears.

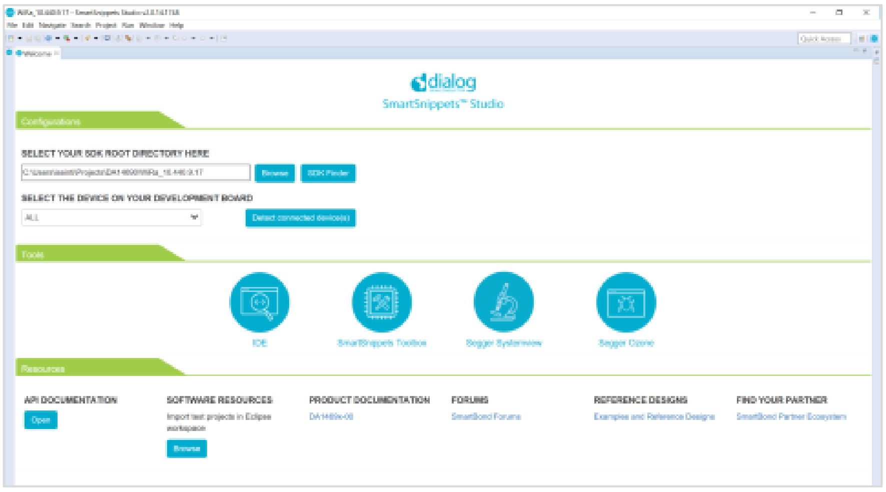


Figure 44. SmartSnippets studio welcome screen

5. To import the demo applications, select **File** > **Import**, then select **General** > **Existing Projects into Workspace**, and click **Next**.

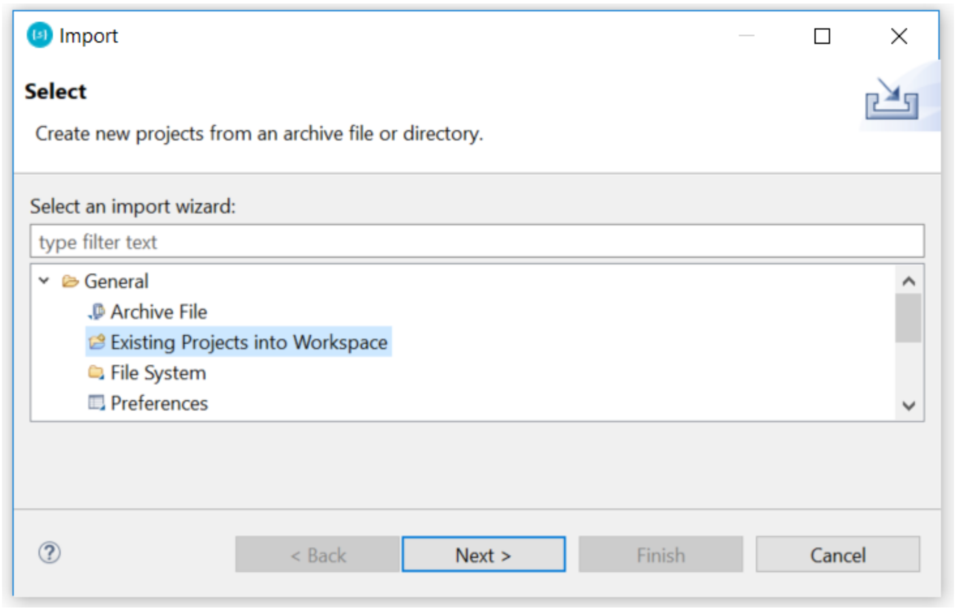
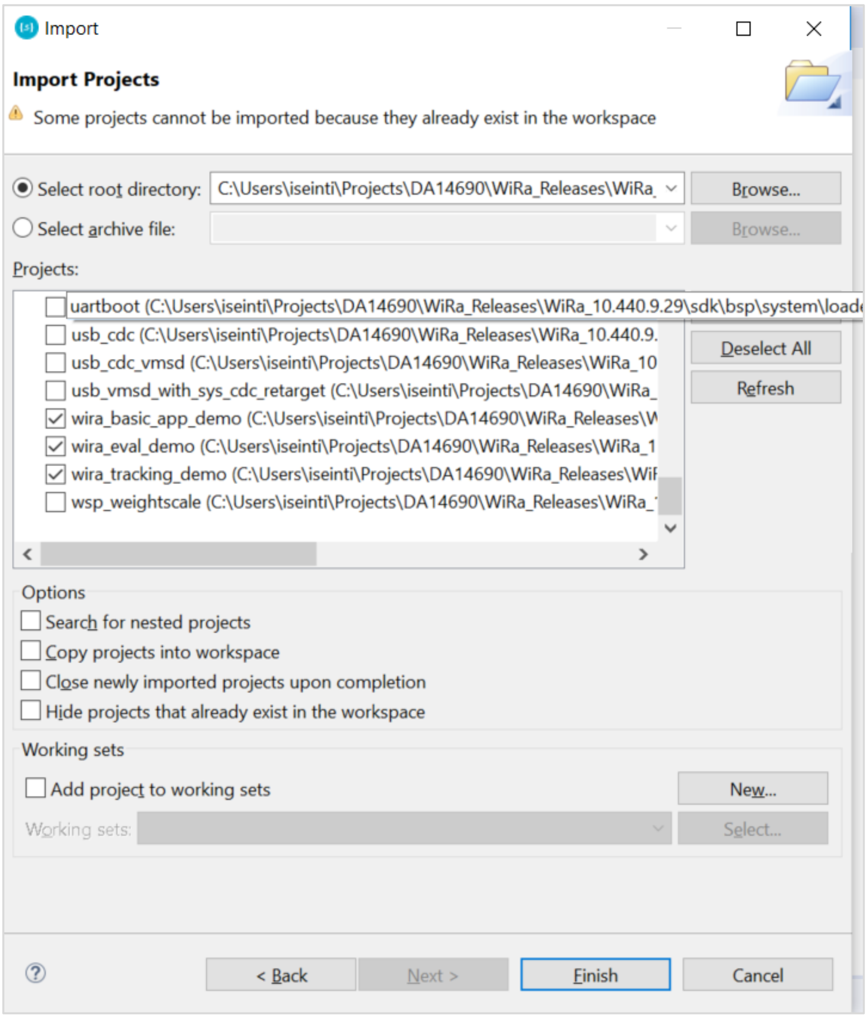


Figure 45. Import SmartSnippets studio project screen

6. Browse the path of the SDK release folder, click **Deselect all**, and from the project list, select only wira_eval_demo, wira_basic_app_demo, wira_tracking_demo projects, and python_scripts. Then click **Finish**.


Import

Import Projects

Some projects cannot be imported because they already exist in the workspace

Select root directory: C:\Users\iseinti\Projects\DA14690\WiRa_Releases\WiRa_

Select archive file:

Projects:

- [ ] uartboot (C:\Users\iseinti\Projects\DA14690\WiRa_Releases\WiRa_10.440.9.29\sdk\bsp\system\load
- [ ] usb_cdc (C:\Users\iseinti\Projects\DA14690\WiRa_Releases\WiRa_10.440.9.
- [ ] usb_cdc_vmsd (C:\Users\iseinti\Projects\DA14690\WiRa_Releases\WiRa_10
- [ ] usb_vmsd_with_sys_cdc_retarget (C:\Users\iseinti\Projects\DA14690\WiRa_
- [x] wira_basic_app_demo (C:\Users\iseinti\Projects\DA14690\WiRa_Releases\W
- [x] wira_eval_demo (C:\Users\iseinti\Projects\DA14690\WiRa_Releases\WiRa_1
- [x] wira_tracking_demo (C:\Users\iseinti\Projects\DA14690\WiRa_Releases\WiF
- [ ] wsp_weightscale (C:\Users\iseinti\Projects\DA14690\WiRa_Releases\WiRa_

Options

- [ ] Search for nested projects
- [ ] Copy projects into workspace
- [ ] Close newly imported projects upon completion
- [ ] Hide projects that already exist in the workspace

Working sets

- [ ] Add project to working sets

Working sets:

Figure 46. List with all available projects to import

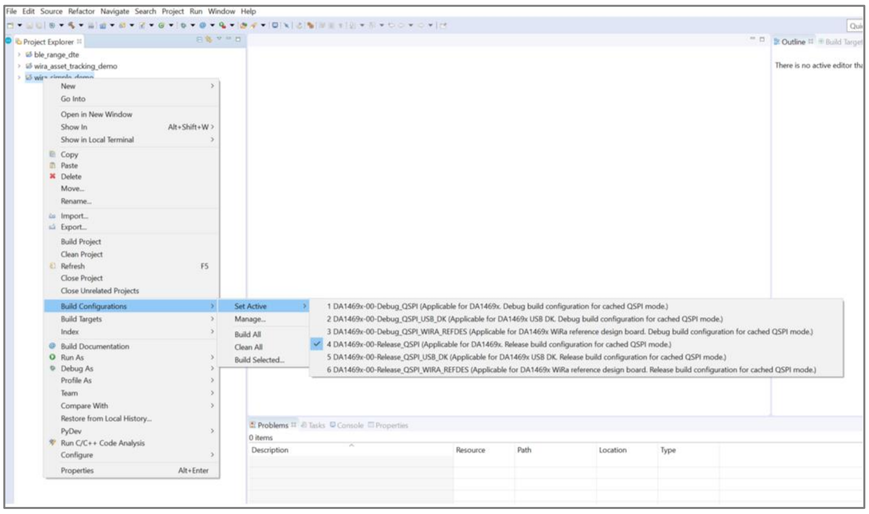
Python_scripts are needed to flash the development kits using SmartSnippets Studio. This procedure is described below.

To build one of the projects, first, the proper Build configuration should be selected. There are six build configurations available. The selection depends on the mode (Debug, Release) and the development kit used (DA1469x PRO DK, DA14695 module DK, USB Kit, WiRa Kit):

- DA1469x-00-Debug_QSPI: Debug mode for PRO DK or for DA14695 module DK.
- DA1469x-00-Debug_QSPI_USB_DK: Debug mode for USB Kit.
- DA1469x-00-Debug_QSPI_WIRA_KIT: Debug mode for USB WiRa Kit.
- DA1469x-00-Release_QSPI: Release mode for PRO DK or for DA14695 module DK.
- DA1469x-00-Release_QSPI_USB_DK: Release mode for USB Kit.
- DA1469x-00-Release_QSPI_WIRA_KIT: Release mode USB WiRa Kit.

To build a project, follow the steps:

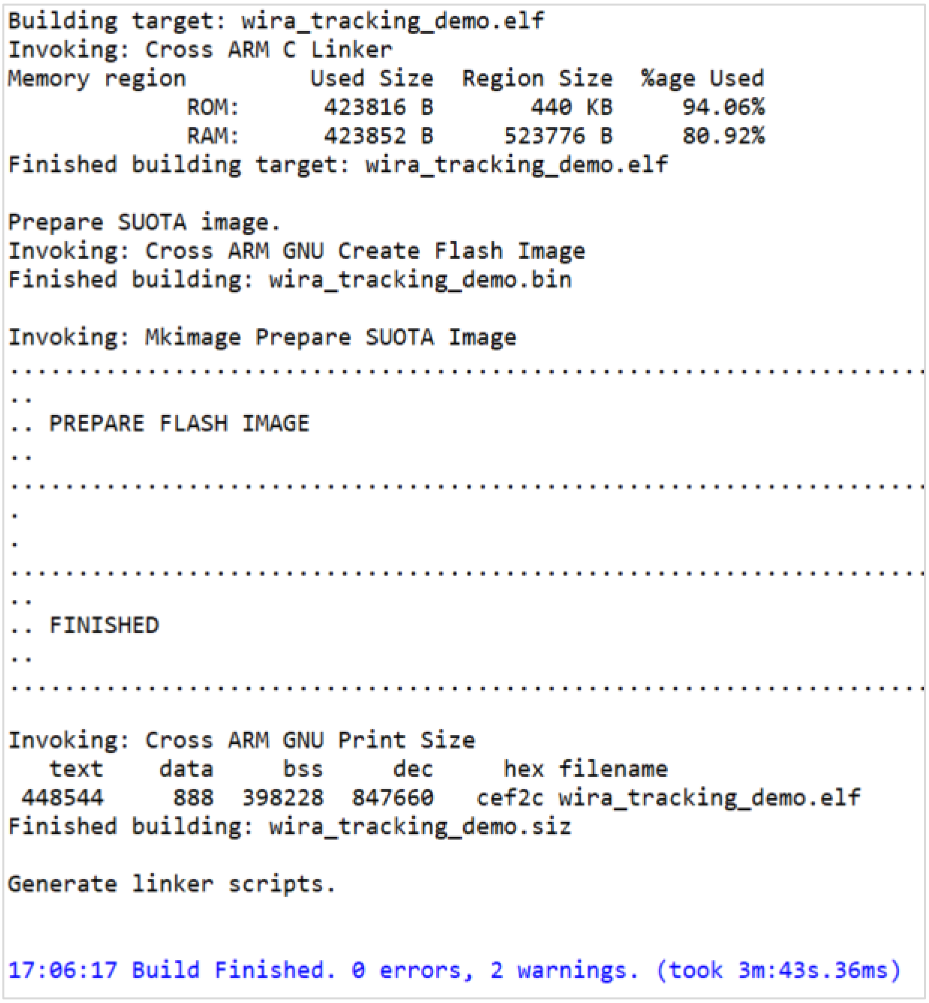
1. Right-click the project name, select **Build Configurations** > **Set Active**, and choose the proper build configuration.



1 DA1469x-00-Debug_QSPI (Applicable for DA1469x. Debug build configuration for cached QSPI mode.)
2 DA1469x-00-Debug_QSPI_USB_DK (Applicable for DA1469x USB DK. Debug build configuration for cached QSPI mode.)
3 DA1469x-00-Debug_QSPI_WIRA_REFDES (Applicable for DA1469x WiRa reference design board. Debug build configuration for cached QSPI mode.)
✓ 4 DA1469x-00-Release_QSPI (Applicable for DA1469x. Release build configuration for cached QSPI mode.)
5 DA1469x-00-Release_QSPI_USB_DK (Applicable for DA1469x USB DK. Release build configuration for cached QSPI mode.)
6 DA1469x-00-Release_QSPI_WIRA_REFDES (Applicable for DA1469x WiRa reference design board. Release build configuration for cached QSPI mode.)

Figure 47. Selection of build configuration

2. Make desired changes in the code, right-click the project name, and select Build project.
3. After a successful build in the Console Window, you can see the messages as in Figure 48.



```
Building target: wira_tracking_demo.elf
Invoking: Cross ARM C Linker
Memory region         Used Size  Region Size  %age Used
             ROM:      423816 B       440 KB     94.06%
             RAM:      423852 B     523776 B     80.92%
Finished building target: wira_tracking_demo.elf

Prepare SUOTA image.
Invoking: Cross ARM GNU Create Flash Image
Finished building: wira_tracking_demo.bin

Invoking: Mkimage Prepare SUOTA Image
...................................................................
..
.. PREPARE FLASH IMAGE
..
...................................................................
.
.
...................................................................
..
.. FINISHED
..
...................................................................

Invoking: Cross ARM GNU Print Size
   text    data     bss     dec     hex filename
 448544     888  398228  847660   cef2c wira_tracking_demo.elf
Finished building: wira_tracking_demo.siz

Generate linker scripts.


17:06:17 Build Finished. 0 errors, 2 warnings. (took 3m:43s.36ms)
```

Figure 48. Console output after successful project build

The binary and image are copied to the `binaries/wira/<demo_app_name>/binaries` and `binaries/wira/<demo_app_name>/suota_images` folders. The development kits can be upgraded with these images either over SUOTA (see Appendix C), binaries using SmartSnippets Toolbox (see Appendix B), or using python_scripts. To flash a development kit using python_scripts, perform the following steps:

1. In the Project Explorer, select the project that you would like to write to the QSPI flash.
2. Click the **External Tools** button on the toolbar.
3. You need to run the configuration script to define which flash is used only the first time. Select program_qpi_config.

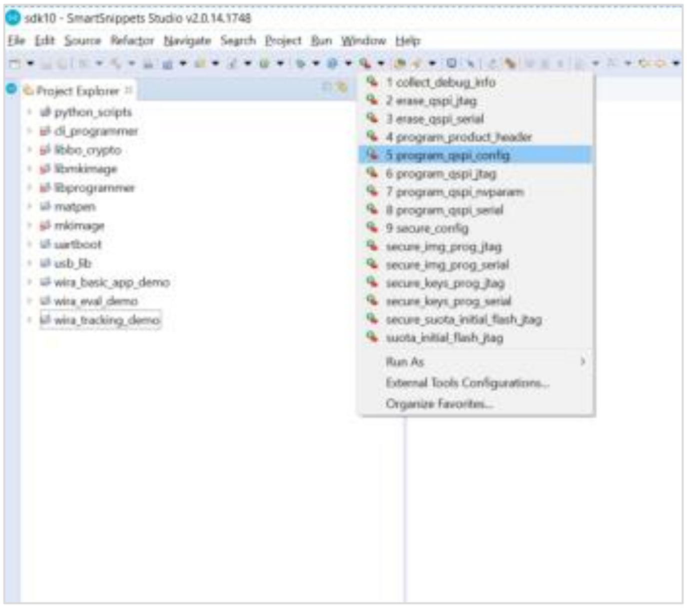
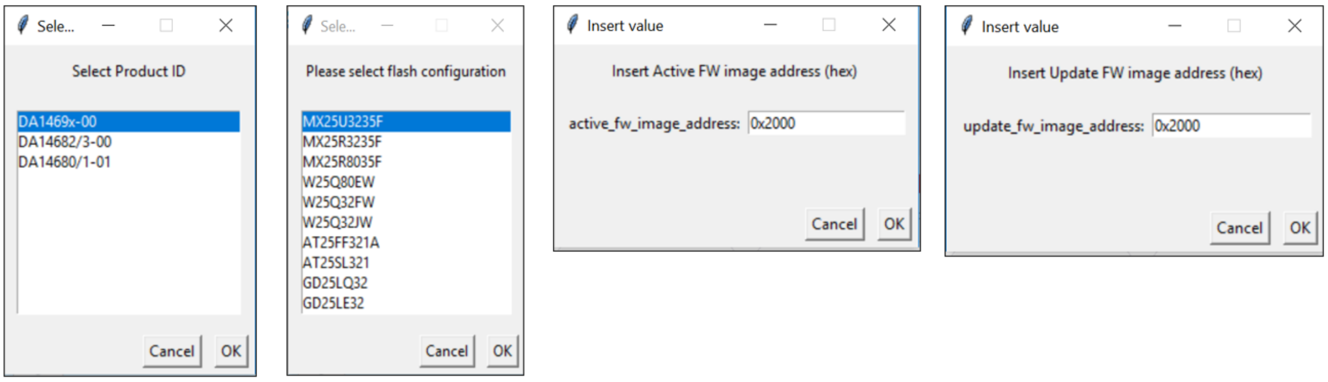


Figure 49. Configure the flash memory for the desired board

4. Select the following options for the DA1469x development kit's configuration:
   a. Product ID: DA1469x-00
   b. Flash configuration: - MX25U3235F for DA1469x Pro-DK, USB-kit, and WiRa-kit
      - P25Q32SL for DA14695 module DK
   c. Active firmware image address: 0x2000
   d. Update firmware image address: 0x2000


Select Product ID
DA1469x-00
DA14682/3-00
DA14680/1-01

Please select flash configuration
MX25U3235F
MX25R3235F
MX25R8035F
W25Q80EW
W25Q32FW
W25Q32JW
AT25FF321A
AT25SL321
GD25LQ32
GD25LE32

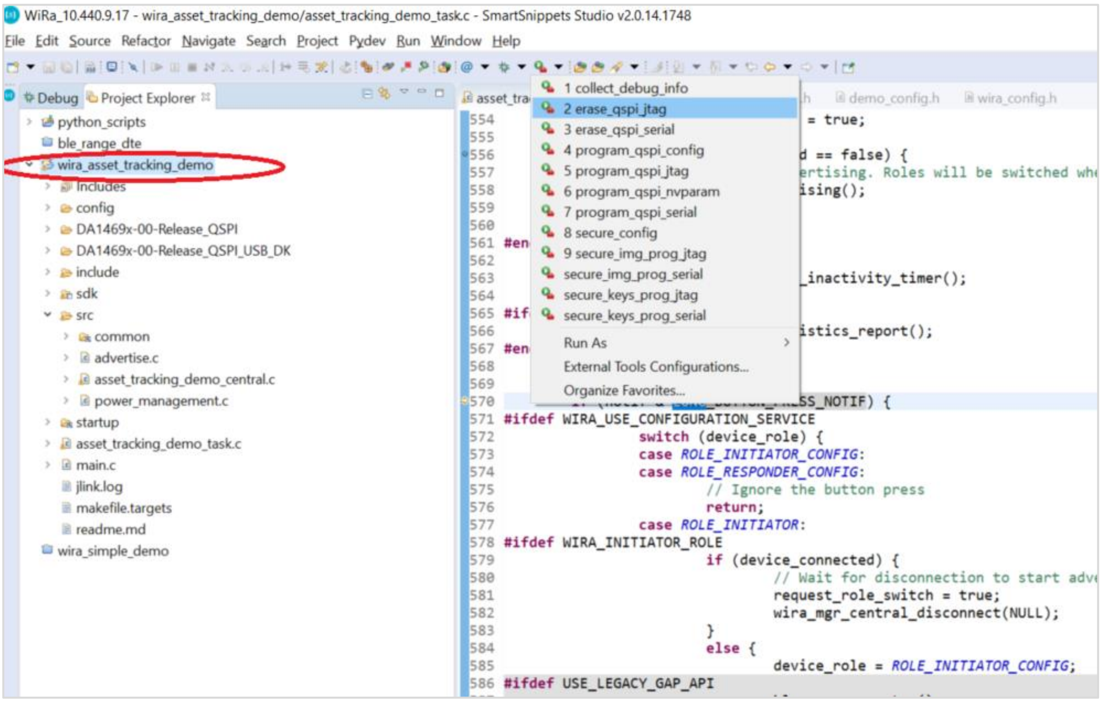
Insert Active FW image address (hex)
active_fw_image_address: 0x2000

Insert Update FW image address (hex)
update_fw_image_address: 0x2000

Figure 50. DA1469x Development Kit's configuration

The configuration is done. Now the flash can be erased if needed or burned directly skipping step 5 and following instructions in step 6.

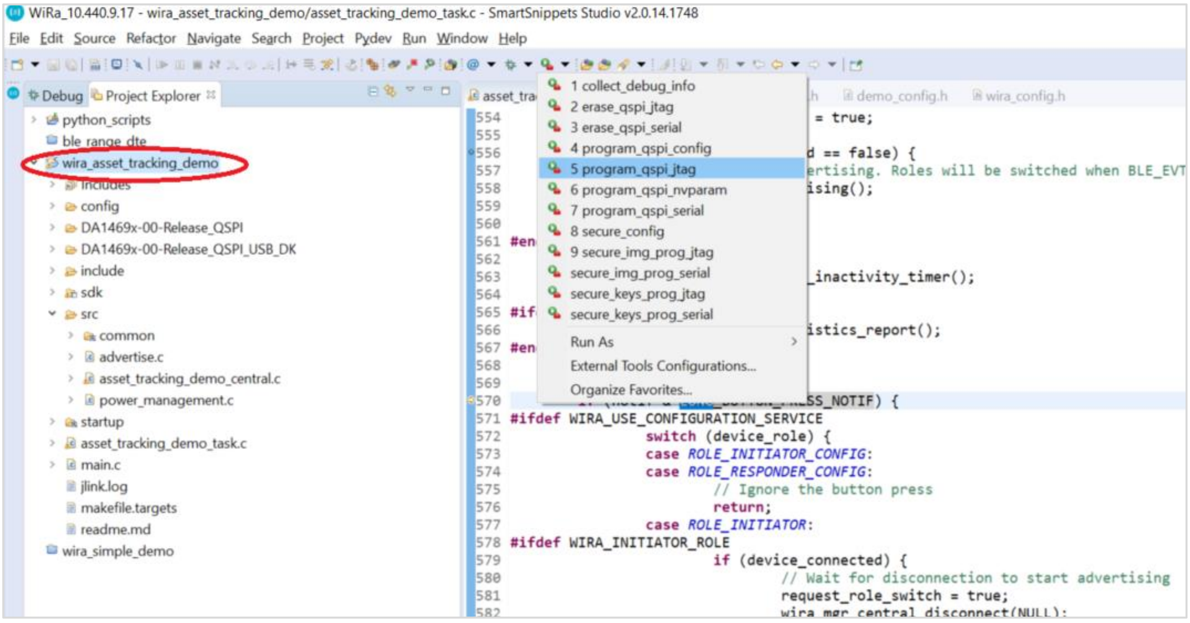
5. To erase the QSPI flash, select the erase_qspi_jtag option.



```
                                  = true;

                                 d == false) {
                                 ertising. Roles will be switched whe
                                 ising();

#en

                                 _inactivity_timer();

#if                              istics_report();
#en

                                 SS_NOTIF) {
#ifdef WIRA_USE_CONFIGURATION_SERVICE
                switch (device_role) {
                case ROLE_INITIATOR_CONFIG:
                case ROLE_RESPONDER_CONFIG:
                        // Ignore the button press
                        return;
                case ROLE_INITIATOR:
#ifdef WIRA_INITIATOR_ROLE
                        if (device_connected) {
                                // Wait for disconnection to start adv
                                request_role_switch = true;
                                wira_mgr_central_disconnect(NULL);
                        }
                        else {
                                device_role = ROLE_INITIATOR_CONFIG;
#ifdef USE_LEGACY_GAP_API
```

Figure 51. Erase flash memory through Python scripts

6. To write the binary to QSPI flash, select the program_qspi_jtag option.



```
                                  = true;

                                 d == false) {
                                 ertising. Roles will be switched when BLE_EVT
                                 ising();

#en

                                 _inactivity_timer();

#if                              istics_report();
#en

                                 SS_NOTIF) {
#ifdef WIRA_USE_CONFIGURATION_SERVICE
                switch (device_role) {
                case ROLE_INITIATOR_CONFIG:
                case ROLE_RESPONDER_CONFIG:
                        // Ignore the button press
                        return;
                case ROLE_INITIATOR:
#ifdef WIRA_INITIATOR_ROLE
                        if (device_connected) {
                                // Wait for disconnection to start advertising
                                request_role_switch = true;
                                wira_mgr_central_disconnect(NULL);
```

Figure 52. Burn flash memory through Python scripts

7. If more than one development kit are connected to the computer, the dialog as in Figure 53 appears in step 5 and step 6, asking you to select which one should be erased or flashed respectively.

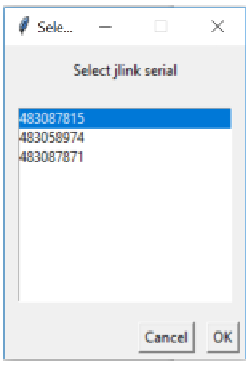
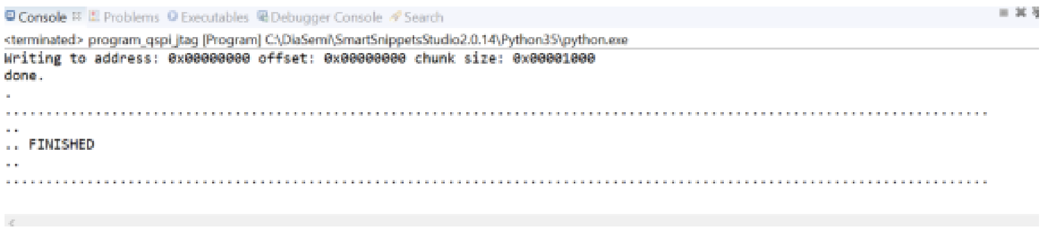


Figure 53. Dialog to select the desired board

8. After selecting the desired device, click **OK** and the procedure starts.

When the write-to-flash procedure succeeds, in the console window, you can see the output as in Figure 54.



```
<terminated> program_qspi_jtag [Program] C:\DiaSemi\SmartSnippetsStudio2.0.14\Python35\python.exe
Writing to address: 0x00000000 offset: 0x00000000 chunk size: 0x00001000
done.
.
..............................................................................................................................
..
.. FINISHED
..
..............................................................................................................................
```

Figure 54. Console output after successful flash programming

Appendix E Using the Android Application

From Play Store, download the "Renesas WiRa" Android application and install it to a mobile phone.

Using the WiRa application, you can:

- Change the configuration of a development kit (Initiator, Responder).
- Read/change parameters of an existing Initiator or Responder.
- Monitor one or more Initiators. A graph is created with the calculated distances for each Initiator.

After launching WiRa application, the "Bluetooth" and "Location" services should be enabled on Android mobile phone.

E.1 Change role from Responder to Initiator

1. To enter the Configuration mode, press the K1 button of the board for more than 1 second. The device remains in this mode for 10 seconds to allow the Android application to connect to the device.
2. To find the current available Responders, tap the **Scan** button or the **Refresh** button under the more options icon, and then select the WiRa Responder of which you want to change the configuration.

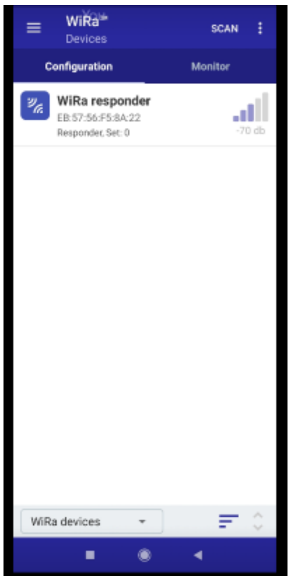


Figure 55. Initial screen of WiRa application with scan results

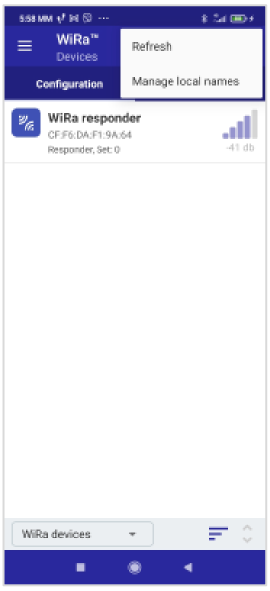
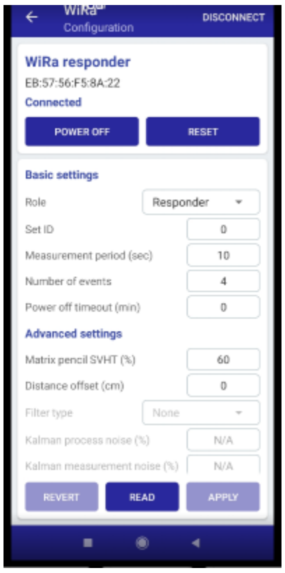


Figure 56. Menu for refreshing scan results

3. In the Role list, change role to Initiator, then press **Apply**, and then **Disconnect**. The development kit starts to scan as Initiator for Responders.



WiRa responder
EB:57:56:F5:8A:22
Connected

Basic settings

Role	Responder
Set ID	0
Measurement period (sec)	10
Number of events	4
Power off timeout (min)	0

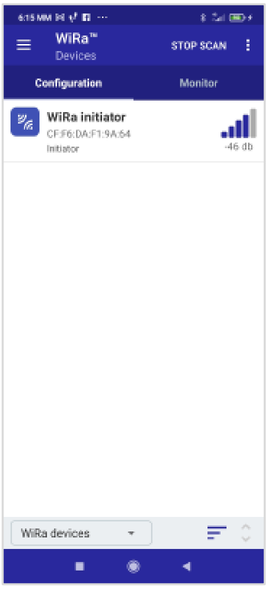
Advanced settings

Matrix pencil SVHT (%)	60
Distance offset (cm)	0
Filter type	None
Kalman process noise (%)	N/A
Kalman measurement noise (%)	N/A

Figure 57. Configuration screen to change the role to Initiator

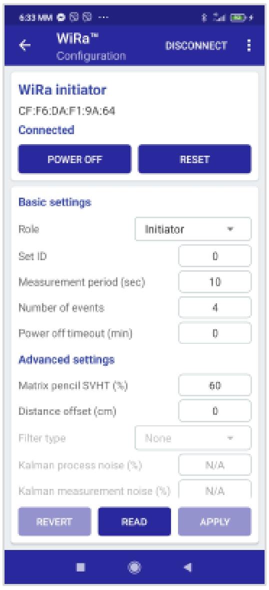
E.2 Change role from Initiator to Responder

1. To enter the configuration mode, press the K1 button of the board for more than 1 second. The device remains in this mode for 10 seconds to allow the Android application to connect to the device.
2. Tap the **Scan** button or the **Refresh** button under the more options icon to find the current available Initiators. Select the WiRa Initiator of which you want to change the configuration.
3. Change Role to Responder in the drop-down list, then press **Apply**, and then **Disconnect**. The development kit starts to advertise as a Responder.



WiRa initiator
CF:F6:DA:F1:9A:64
Initiator
-46 db

Figure 58. Initial screen of WiRa application with scan results



WiRa initiator
CF:F6:DA:F1:9A:64
Connected

Basic settings

Role	Initiator
Set ID	0
Measurement period (sec)	10
Number of events	4
Power off timeout (min)	0

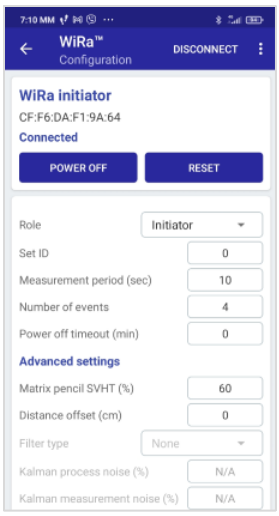
Advanced settings

Matrix pencil SVHT (%)	60
Distance offset (cm)	0
Filter type	None
Kalman process noise (%)	N/A
Kalman measurement noise (%)	N/A

Figure 59. Configuration screen to change the role to Responder

E.3 Change parameters using WiRa application

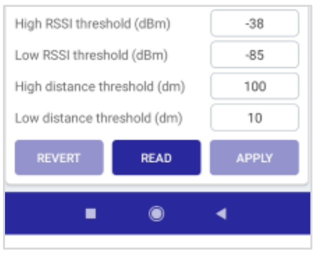
Using the WiRa application, you can change real-time parameters either for a Responder or for an Initiator:



WiRa initiator
CF:F6:DA:F1:9A:64
Connected

Role	Initiator
Set ID	0
Measurement period (sec)	10
Number of events	4
Power off timeout (min)	0
Advanced settings	
Matrix pencil SVHT (%)	60
Distance offset (cm)	0
Filter type	None
Kalman process noise (%)	N/A
Kalman measurement noise (%)	N/A

Figure 60. Configuration parameters (part 1)

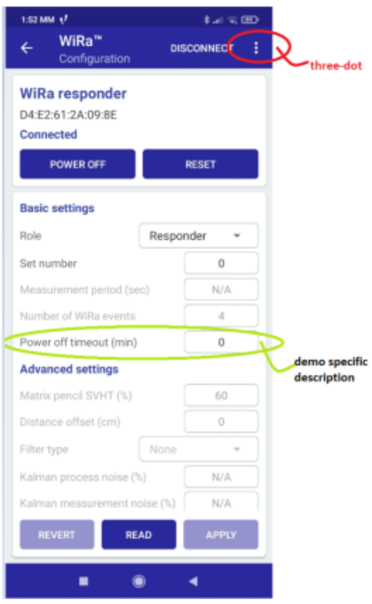


High RSSI threshold (dBm)	-38
Low RSSI threshold (dBm)	-85
High distance threshold (dm)	100
Low distance threshold (dm)	10

Figure 61. Configuration parameters (part 2)

- Set ID: Initiator and Responder with the same ID match and a connection between them will be established automatically.
- Measurement period (s): in this period, a number of WiRa events are performed.
- Number of events: number of WiRa events that are executed in one measurement period.
- Power off timeout (min): after this timeout, the device goes to sleep. If it is set to 0, the device remains forever active.
- Matrix pencil SVHT (%): Singular Value Hard Threshold scale for Matrix Pencil.
- Distance offset (cm): a constant offset that is deducted from the WiRa measurement.
- Filter type: a sliding average or Kalman filter can be applied to the WiRa measurements.
- Kalman process noise (%): the process noise parameter of the Kalman filter as described in Section 4.3.3.
- Kalman measurement noise (%): the measurement noise parameter of the Kalman filter as described in Section 4.3.3.
- RSSI and distance thresholds: two RSSI and two distance thresholds can be defined. These are application-specific thresholds and the use of which are described for each demo in Section 5.2.2.2 and Section 5.3.3. The default description of the above parameters can be customized for each demo.

Moreover, the application gives the option to display demo-specific descriptions for each parameter, if "Use description" is selected in the more options menu. Figure 62, Figure 63, and Figure 64 show how a description of a parameter changes for a specific demo, after selecting the "Use description" option.



WiRa responder
D4:E2:61:2A:09:8E
Connected
POWER OFF | RESET

Basic settings

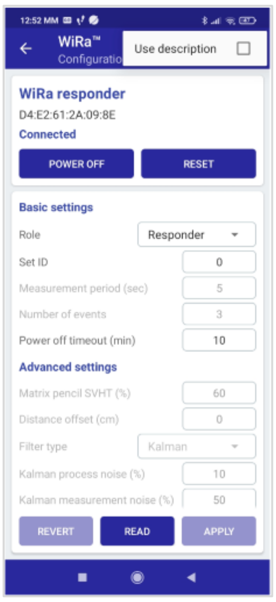
Role	Responder
Set number	0
Measurement period (sec)	N/A
Number of WiRa events	4
Power off timeout (min)	0

Advanced settings

Matrix pencil SVHT (%)	60
Distance offset (cm)	0
Filter type	None
Kalman process noise (%)	N/A
Kalman measurement noise (%)	N/A

REVERT | READ | APPLY

Figure 62. Initial configuration screen



WiRa responder
D4:E2:61:2A:09:8E
Connected
POWER OFF | RESET

Basic settings

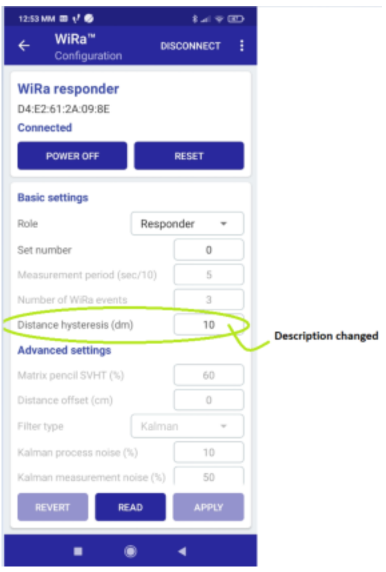
Role	Responder
Set ID	0
Measurement period (sec)	5
Number of events	3
Power off timeout (min)	10

Advanced settings

Matrix pencil SVHT (%)	60
Distance offset (cm)	0
Filter type	Kalman
Kalman process noise (%)	10
Kalman measurement noise (%)	50

REVERT | READ | APPLY

Figure 63. "Use Description" dialog



WiRa responder
D4:E2:61:2A:09:8E
Connected
POWER OFF | RESET

Basic settings

Role	Responder
Set number	0
Measurement period (sec/10)	5
Number of WiRa events	3
Distance hysteresis (dm)	10

Advanced settings

Matrix pencil SVHT (%)	60
Distance offset (cm)	0
Filter type	Kalman
Kalman process noise (%)	10
Kalman measurement noise (%)	50

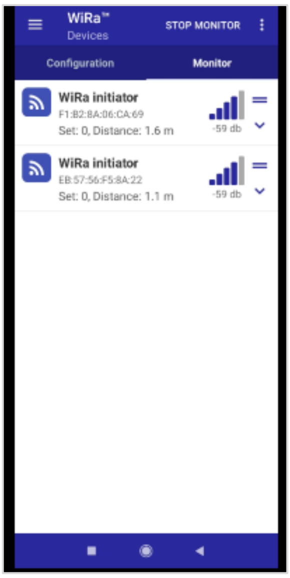
REVERT | READ | APPLY

Figure 64. Result after checking "Use Description"

E.4 Monitor one or more initiators

You can monitor the measurements reported by one or more Initiators using the WiRa Android application. To do that, follow the steps:

1. Go to the **Monitor** subtab and tap **Monitor**.

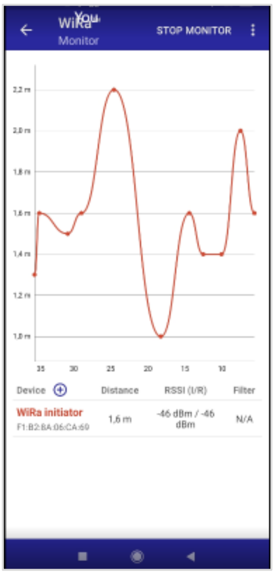


WiRa initiator
F1:B2:8A:06:CA:69
Set: 0, Distance: 1.6 m

WiRa initiator
EB:57:56:F5:8A:22
Set: 0, Distance: 1.1 m

Figure 65. Monitor tab

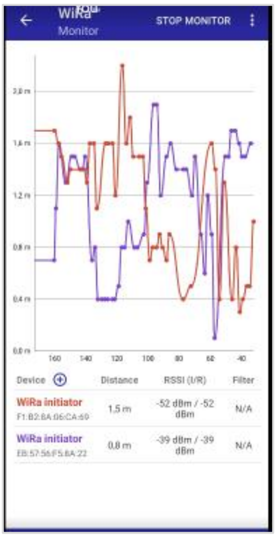
2. Select the first Initiator and on the next page, tap Device + to monitor the second Initiator.



Device	Distance	RSSI (I/R)	Filter
WiRa initiator F1:B2:BA:06:CA:69	1,6 m	-46 dBm / -46 dBm	N/A

Figure 66. Monitor the distance between an Initiator and a Responder

Each Initiator is monitored with a different color as it is shown in Figure 67.



Device	Distance	RSSI (I/R)	Filter
WiRa initiator F1:B2:BA:06:CA:69	1,5 m	-52 dBm / -52 dBm	N/A
WiRa initiator EB:57:56:F5:BA:22	0,8 m	-39 dBm / -39 dBm	N/A

Figure 67. Monitor the distance from a Responder for two Initiators

You can also monitor a rolling distance average. This feature can be enabled by taping the Filter option for the desired Initiator. A window appears, where you should define a time window in seconds. For this time window, the distance average is calculated and updated continuously and monitored on the screen with dots.

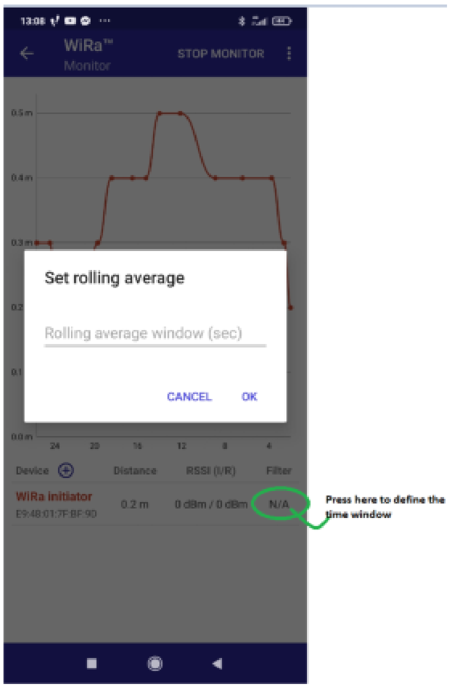


Figure 68. Dialog to set the rolling average window

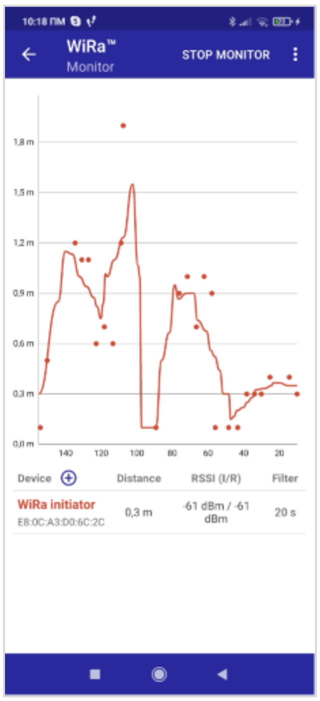


Figure 69. Monitor the rolling distance average

Revision History

Revision	Date	Description
3.6	Jan 24, 2024	Added support for DA14695 module DK
3.5	Nov 15, 2022	Removed Appendix D, Standalone Flash Programmer
3.4	Nov 2, 2022	Created Appendix E
3.3	Sept 1, 2022	Updated the demos and other Appendixes
3.2	July 29, 2022	Added WiRa manager section
3.1	June 2, 2022	Update Appendixes A, B, and C
2.2	May 31, 2022	New version for WiRa SDK 10.440.10
2.1	July 22, 2020	Rebranded to Renesas
2.0	July 2, 2020	Editorial changes
1.0	May 7, 2020	Added image of USB Kit with ZOR antenna

Status Definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

RoHS Compliance