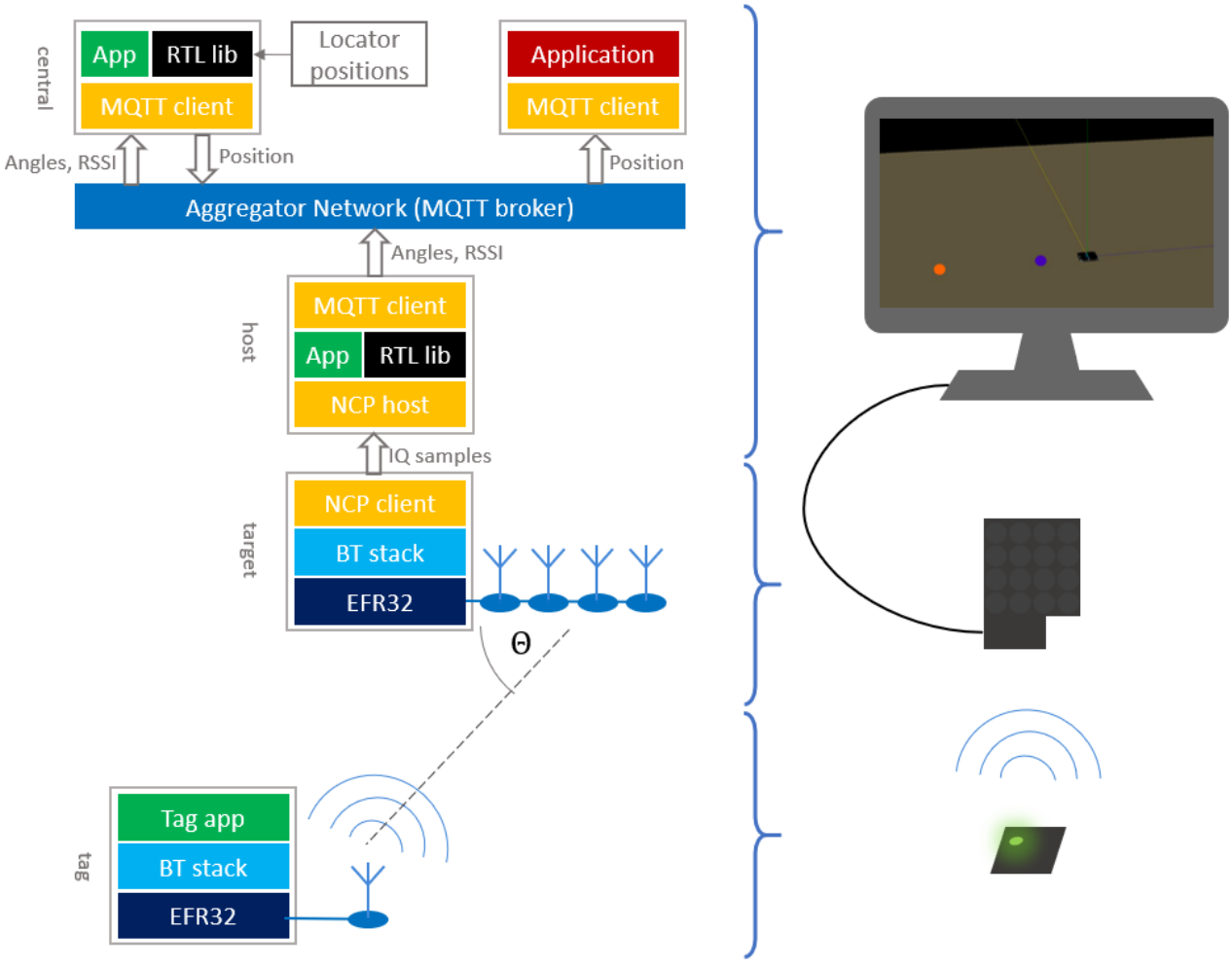
Bluetooth Direction Finding Lab Manual

This lab manual walks you through the Direction Finding sample apps of the Bluetooth SDK. It helps you set up the development environment, and put together a fully functional Real Time Locating system. Using a Thunderboard and a Direction Finding board, by the end of this lab, you should be able to see the approximate positions of the thunderboard in real time, displayed on a 3D graphical interface on your PC.

KEY FEATURES

- Setting up development environment
- AoA asset tags
- AoA single locator
- AoA Multi-locator
- Displaying the positions



1 Prerequisites

To complete this lab, you will need the followings:

- A Thunderboard BG22 board, or any Silicon Labs radio board with EFR32BG22 part or BGM220 module
- A Direction Finding (antenna array) board, BRD4185A, with a WSTK
- Simplicity Studio 5 installed, with Gecko SDK v3.1 including Bluetooth SDKv3.1
- A PC with Windows, Linux or Mac OS. This lab focuses on Windows, but other OSs can be used as well.

2 Setting up AoA locator host application development environment

Windows

2.1 Download and install MSYS2: https://www.msys2.org/

2.2 Open the Mintty bash. Make sure to start Mingw-w64 64 (mingw64.exe) when launching Mintty. MSYS2. 32-bit versions will not work.

2.3 Install GCC for MinGW-w64:

```
pacman -S make mingw-w64-x86_64-gcc
```

2.4 Download and install Mosquito MQTT broker:

https://mosquitto.org/download/

2.5 Install MQTT Explorer (or any similar MQTT client) for debugging MQTT data.

http://mqtt-explorer.com/

2.6 Install Python 3.7 (exactly this version)

https://www.python.org/downloads/release/python-370/

Linux

2.1 Install mosquitto libraries

```
sudo apt install libmosquitto-dev
```

2.2 Install MQTT Explorer (or any similar MQTT client) for debugging MQTT data.

```
snap install mqtt-explorer
```

2.3 If your python version is not v3.7, install Python 3.7

```
sudo apt install python3.7
```

Mac OS

2.1 Download and install Mosquito MQTT broker:

https://mosquitto.org/download/

2.2 Install MQTT Explorer (or any similar MQTT client) for debugging MQTT data.

http://mqtt-explorer.com/

2.3 Install Python 3.7 (exactly this version)

https://www.python.org/downloads/release/python-370/

3 Flashing the Thunderboard with Asset Tag sample app

The Thunderboard will serve as an asset tag to be tracked. For this reason it must be programmed with an asset tag sample app that is able to transmit Constant Tone Extensions (CTEs).

3.1 Open Simplicity Studio 5

3.2 Attach your Thunderboard BG22 to the PC, and select it on the Debug Adapters tab of Simplicity Studio

3.3 In the Launcher view's Overview tab, make sure that Gecko SDK v3.1 is selected as the Preferred SDK

3.4 Open the Example Projects & Demos tab

3.5 Find the Bluetooth – SoC AoA Asset Tag sample app, and click CREATE

3.6 Build the sample app (use the build icon or Project > Build Project)

3.7 Flash the sample app to the board:

a. in the project folder find the output folder of the compiler (e.g. GNU ARM v7.2.1 - Debug)

b. right click on soc_aoa_asset_tag.hex and select Flash to Device

c. click Program

If you have not had flashed a bootloader to the device earlier, then program the bootloader:

3.8 Use the same Flash Programmer window in Simplicity Studio with which you flashed the project, or open Simplicity Commander

3.9 Browse for the following bootloader sample app: C:\SiliconLabs\SimplicityStudio\v5\developer\sdks\gecko_sdk_suite\v3.1\platform\bootloader\sample-apps\bootloader-storage-internal-single-512k\efr32mg22c224f512im40-brd4182a\bootloader-storage-internal-single-512k.s37

Note: you may need to change the file format filtering from .hex to .s37 while browsing!

3.10 Click Program

4 Flashing the Antenna Array board with NCP AoA Locator sample app

The Antenna Array board (Direction Finding board) will run the Bluetooth stack on the locator side, and it will also serve as an NCP client toward the host. Since the application runs on the host, the NCP client firmware can be flashed as it is.

4.1 Open Simplicity Studio 5

4.2 Plug the Antenna Array board to WSTK mainboard, attach the WSTK to your PC, and select it on the Debug Adapters tab. It will be listed as Direction Finding board.

4.3 In the launcher view make sure that Gecko SDK v3.1 is the Preferred SDK

4.4 Open the Example Projects & Demos tab

4.5 Find the Bluetooth – NCP AoA Locator demo, and click RUN

This will flash the prebuilt sample app to the board. If you have not had flashed a bootloader to the device earlier, then program the bootloader:

4.6. Use the Flash Programmer window in Simplicity Studio or open Simplicity Commander

4.7 Browse for the following bootloader sample app: C:\SiliconLabs\SimplicityStudio\v5\developer\sdks\gecko_sdk_suite\v3.1\platform\bootloader\sample-apps\bootloader-storage-internal-single-512k\efr32mg22c224f512im40-brd4182a\bootloader-storage-internal-single-512k.s37

Note: you may need to change the file format filtering from .hex to .s37 while browsing!

4.8 Click Program

5 Building and running the AoA locator host application

The AoA locator host application is responsible for controlling the stack (by receiving events from the stack and sending commands to the stack). Its main purpose is to

- find the tag
- either connect to the tag or just receive advertisements from it
- enable CTE reception from the tag
- collect the IQ samples
- init the Real Time Locating (RTL) library
- feed the RTL library with the IQ samples
- obtain the angle estimation from the RTL library
- publish the angle estimation to the MQTT network

To build the aoa_locator host application

5.1 If you use windows, open MSYS2 MinGW 64bit as shown in section 2

5.2 Change to the $G*SDK_DIR/app/Bluetooth/example_host/aoa_locator* directory:

```
cd /c/SiliconLabs/SimplicityStudio/v5/developer/sdks/gecko_sdk_suite/v3.1/app/blue-tooth/example_host/aoa_locator
```

5.3 Build the AoA locator host application for Silabs mode by running *make*

5.4 Alternatively, build the AoA locator host application for Connection mode by running

```
make APP_MODE=conn
```

This is usually not recommended for multiple locators, but can be used if only one locator and one tag is used for a demonstration

To run the aoa_locator host application:

5.5 Open Task Manager, open the Services tab, and make sure that MQTT service (mosquitto.exe) is running on your computer. If the service is stopped, start it (right click > start). If you cannot see the mosquitto service, make sure it was installed properly.

5.6 Push the reset button on the WSTK

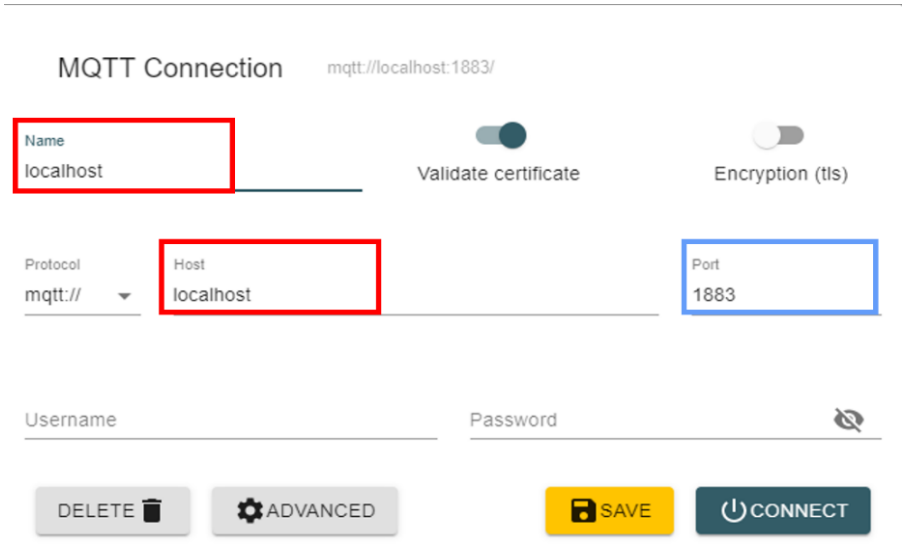
5.7 Find the COM port number assigned to your antenna array board, e.g. using a terminal program

5.8 Run the application with the port number as a parameter, e.g.:

./exe/aoa_locator.exe -u COM10

6 Checking the MQTT traffic with MQTT Explorer

6.1 Open MQTT Explorer and apply the following configurations (host: localhost, port: 1883). If you are running your MQTT broker on any other machine than your PC, you need to replace Host with the IP address of that machine.



6.2 Now you should see something similar like the following.

7 Building and running the AoA multilocator host application

The AoA multilocator host application is responsible for collecting angle data from all locators and estimating the position of the tag from these angles and from the known positions of the locators using the RTL library. In this lab we are using a single locator only, so seemingly the multilocator demo does not add any value. However the procedure of setting up a Real Time Locating System can be learned with a single locator, as well, and can be applied for multiple locators later.

In a single locator use case, the aoa_multilocator sample app will use the azimuth, elevation and the RSSI data, received from the aoa_locator, instead of the angle data provided by multiple locators. These values can be used to give a rough estimation for the x,y,z coordinates of the tag. The position won't be accurate but can be used for demonstration purposes. After the position is estimated the aoa_multilocator sample app will publish it to the MQTT network and it can be obtained by other apps.

To build the aoa_multilocator host application:

7.1 If you use windows, open MSYS2 MinGW 64bit as shown in section 2

7.2 Change to the $G*SDK_DIR/app/Bluetooth/example_host/aoa_multilocator* directory:

```
cd /c/SiliconLabs/SimplicityStudio/v5/developer/sdks/gecko_sdk_suite/v3.1/app/blue-
tooth/example_host/aoa_multilocator
```

7.3 Build the app by running *make*

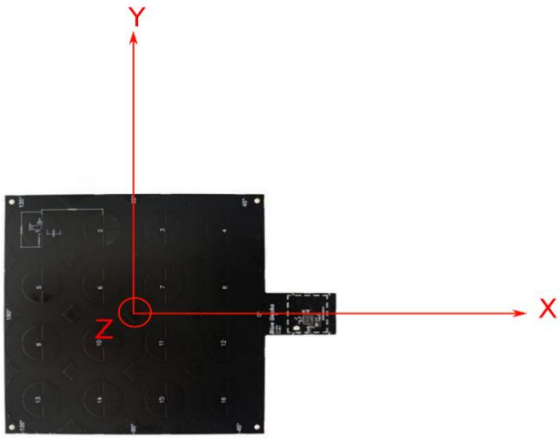
In order to estimate the position of asset tags correctly, the multi-locator host application needs to know the ID of each locator, their position relative to a local coordinate system, and their orientation with respect to the X, Y, and Z axis. This information is provided to the multi-locator host using a JSON config file during runtime.

To create the multi-locator configuration file:

7.4 Navigate to *GSDK_DIR/app/bluetooth/example_host/aoa_multilocator/config/* and open the *multilocator_config.json* file with your favorite editor

7.5 Modify the configuration parameters as below (or simply overwrite the whole content with the structure below). With these settings the coordinate system will be situated like in the figure on the right.

```
{
  "id": "multilocator_test_room",
  "locators": [
    {
      "id": "ble-pd-111111111111",
      "coordinate": {
        "x": 0.0,
        "y": 0.0,
        "z": 0.0
      },
      "orientation": {
        "x": 0.0,
        "y": 0.0,
        "z": 0.0
      }
    }
  ]
}
```



7.6 The locator_id is formed as ble-<ADDRESS_TYPE>-<BLE_ADDRESS>, where <ADDRESS_TYPE> is either sr (for static random) or pd (for public device). <BLE_ADDRESS> is the 6-byte address without any separators. You can find the locator BLE ADDRESS at the beginning of the debug print when running the single locator host application. Find your locator's device address and replace the "id" field in the configuration file with it, e.g.:

```
"id": "ble-pd-842E1431C72A"
```

7.7 Save and close the file.

Now you can run the multilocator sample app to turn your angle data into position data:

7.8 First make sure, that aoa_locator.exe is already running

7.9 Open a new instance of MSYS2 MinGW 64bit (as shown in section 2)

7.10 Change to the $G*SDK_DIR/app/Bluetooth/example_host/aoa_multilocator* directory:

```
cd /c/SiliconLabs/SimplicityStudio/v5/developer/sdks/gecko_sdk_suite/v3.1/app/blue-
tooth/example_host/aoa_multilocator
```

7.11 Run the application with the configuration file provided as a parameter:

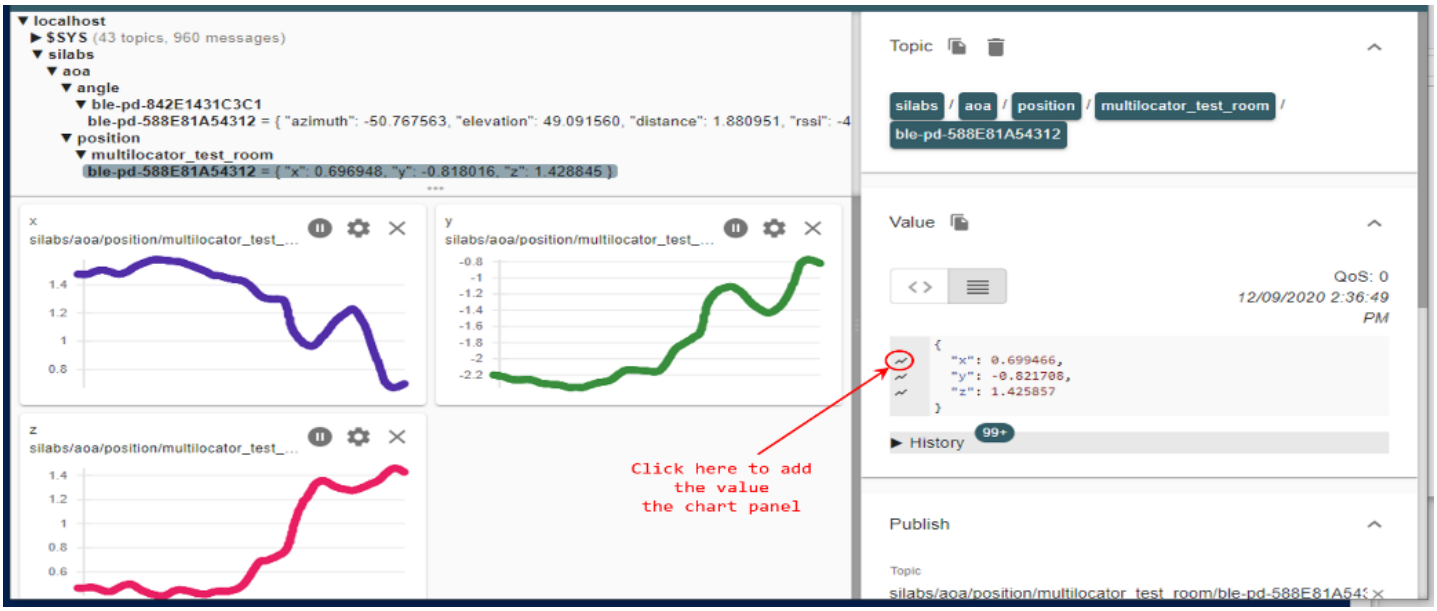
```
./exe/aoa_multilocator.exe -c ./config/multilocator_config.json
```

7.12 If you don't see any data being processed, check again if your locator ID matches your locator address.

8 Checking the MQTT traffic with MQTT Explorer – again

8.1 Open MQTT Explorer and monitor the Position MQTT messages

8.2 Add the x, y, z position values to the chart panel if you want to see the changes in graphs. This can be done by clicking on the small icons on the left of the values, as shown in the next figure.



9 Running Python Visualization Script for Silicon Labs Multi-Locator Sample Application

Looking at the bunch of position data in the MQTT explorer does not give a nice visual feedback about the position of the asset tag. Although the MQTT Explorer can show graphs how the x, y and z coordinates change, it is still not really satisfying. Displaying the tag in a 2D or 3D coordinate system seems like a good solution, but that needs a new application made for this purpose. That is why a visualization script is also provided in the Bluetooth SDK. The script is written in python. It reads the position data from the MQTT network, and displays it on a nice GUI.

To run the visualization script:

9.1 Make sure Python 3.7 is installed in your environment. The version should be exactly 3.7, as Python 2.x or 3.8 will not work

9.2 Run the following commands to make sure *pip* is up to date:

```
py -3.7 -m pip install --upgrade pip
```

```
py -3.7 -m pip install --upgrade setuptools
```

alternatively you can use

```
python3.7 -m pip install –upgrade pip
```

```
python3.7 -m pip install –upgrade setuptools
```

9.3 Install the following packages to your Python 3.7 installation

- pyqtgraph
- pyqt5==5.14.0 (this exact version for Linux)
- pyopengl

- numpy
- Pillow
- paho-mqtt

To install these packages, open a normal command prompt and type the following command for each package:

```
py -3.7 -m pip install <package name>
```

Alternatively, you can use:

```
python3.7 -m pip install <package name>
```

9.4 If installation of one of the previous packages cannot be done, the Python visualization may not work properly. Consider running in pipenv or docker in that case.

9.5 Make sure that both the aoa_locator and the aoa_multilocator apps are running

9.6 Open a normal command prompt and navigate to the folder:

```
cd c:/SiliconLabs/SimplicityStudio/v5/developer/sdks/gecko_sdk_suite/v3.1/app/blue-tooth/example_host/aoa_multilocator_gui
```

9.7 Run the visualization script with either of the following commands:

```
py -3.7 app.py
```

or `python3.7 app.py`

Note: in case you do not use the default multilocator_config.json file found under /app/buetooth/example_host/aoa_multilocator/config (e.g. because you wanted to keep the original file and made a copy), then define the location of your config file with the -c switch:

```
py -3.7 app.py -c ./../aoa_multilocator/config/my_config.json
```

Similarly, if you are not using the default MQTT host:port settings, then define them using the -m switch

9.8 Now you should be able to see the GUI start, displaying the locators and tags. Note, that because of using a single locator only, the accuracy will not be very good, and the filtering makes the movement of the tag quite slow on the display.

To move around the 3D view, use your mouse and the *shift* and *ctrl* keys.