ARIS EDGE

BLE 3D Demo

Application Note

Outline

Acronyms 4
1. Introduction 5
2. System Requirements 5
2.1 Hardware Requirements 5
2.2 Software Requirements 5
2.3 Demo loading 5
2.3.1 Synergy S1 programming 5
2.3.2 Silicon Labs MGM111 programming 8
3. Design overview 9
3.1 Functionalities 9
3.2 Architecture 9
3.3 Data sharing 10
3.4 BLE interface 11
3.4.1 S1 to BLE communication 11
3.4.2 BLE communication sequence 12

Revisions

REVISION	DATE	DESCRIPTION	STATUS	AUTHOR	REVISER
0.1	28/09/2017	Document created.	draft	C. Tagliaferri	
1.0	28/09/2017	Document released	release	C. Tagliaferri	A. Ricci

Disclaimer

RELOC s.r.l.
HEADQUARTERS
Via Borsari, 23/A 43126 – Parma (Italy)
Phone +39-0521-1759942
www.reloc.it
info@reloc.it

Acronyms

APP	Mobile Application
BLE	Bluetooth Low Energy
DOF	Degrees of Freedom
IMU	Inertial Measurement Unit
MCU	Micro Controller Unit
S1	Renesas Synergy S1 MCU

1. Introduction

The ARIS EDGE “BLE 3D” project demonstrates sensing and communication capabilities of the ARIS EDGE board.

BLE 3D demo code periodically reads values from the on-board BNO055 sensor, a 9-axis absolute orientation inertial unit, and makes the ARIS EDGE board orientation available on the BLE interface.

ARIS Tools mobile application is provided in order to receive the ARIS EDGE board orientation through BLE interface and display it graphically.

<The project is based on Renesas Synergy SSP 1.3.0 framework and the demo workspace is tested using Renesas e2 studio (version 5.4.0.023).

In the following, the architecture and functionalities of the ARIS EDGE BLE 3D demo are detailed.

2. System Requirements

2.1 Hardware Requirements

- ARIS EDGE board R1.1
- Mini-USB cable
- J-Link programmer used for MGM111 firmware loading
- Android smartphone with “ARIS Tools” application (*please note*: the iOS version of the “ARIS Tools” APP does not support the BLE 3D demo).

2.2 Software Requirements

The following tools have been exploited for demo application development:

- (S1 firmware) Renesas Synergy SSP 1.3.0 framework
- (S1 firmware) e2studio 5.4.0.023
- (MGM BLE firmware) BGtool 2.3.3-2346

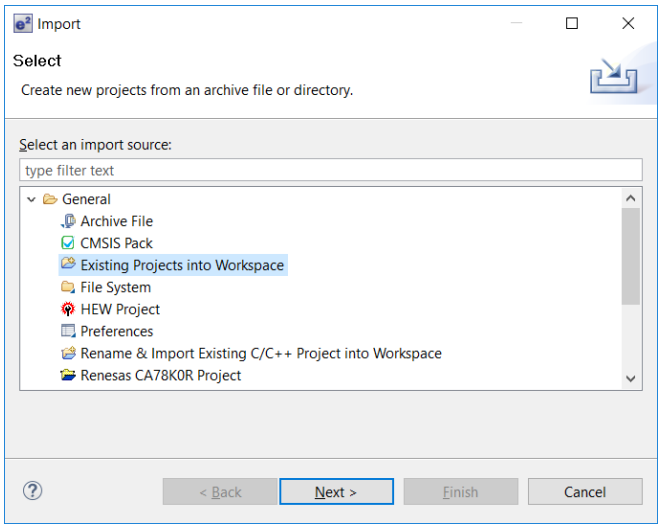
2.3 Demo loading

In order to execute the ARIS 3D demo, both Synergy S1 microcontroller and Silicon Labs MGM111 system-on-chip should be programmed.

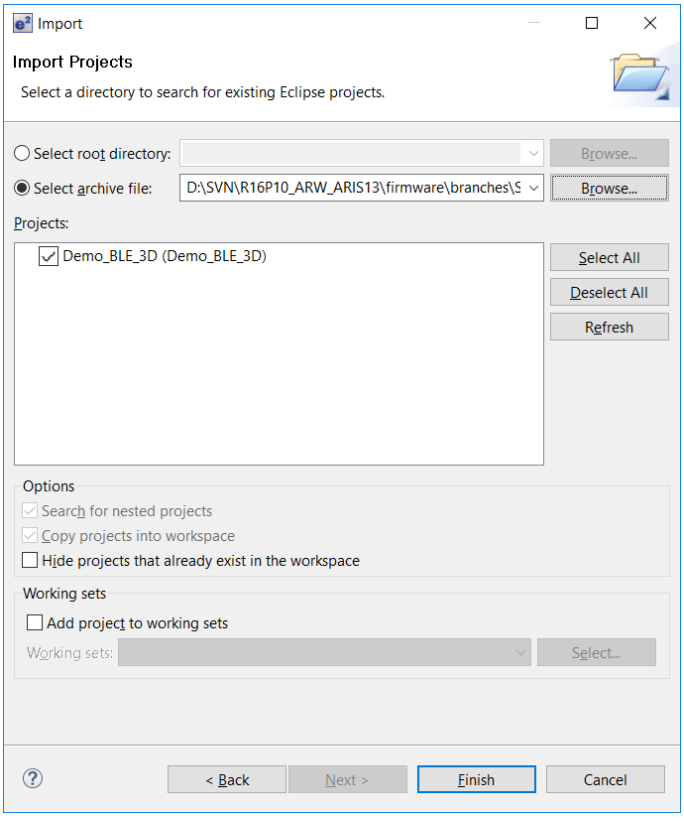
2.3.1 Synergy S1 programming

Synergy S1 programming could be performed exploiting e2studio IDE and the ARIS EDGE on-board J-Link debugger.

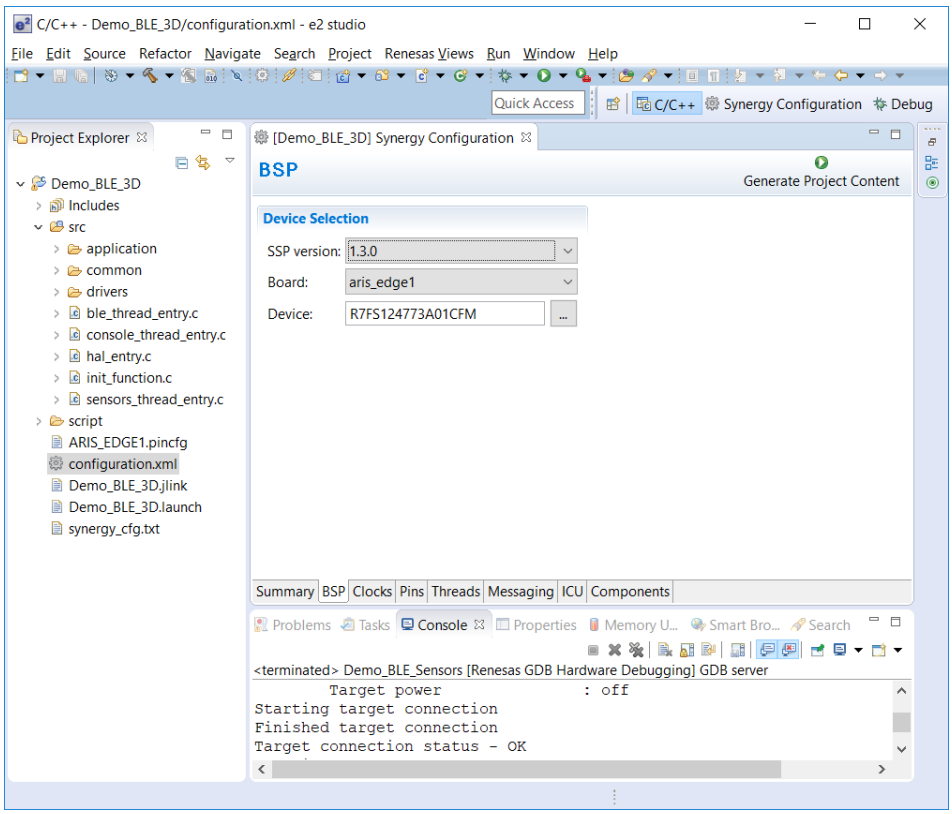
Connect the ARIS EDGE board J5 mini-USB to the host PC. Launch e2studio IDE and import the ARIS EDGE 3D demo Synergy archive project by selecting `File` → `Import` from IDE menu and `Existing Projects into Workspace` option in the dialog box; press the `Next` button.



Browse for the S1 project archive file (`ARIS_EDGE__Demo_BLE_3D_v1_3_0.zip`) and press `Finish` button to confirm the importing procedure.

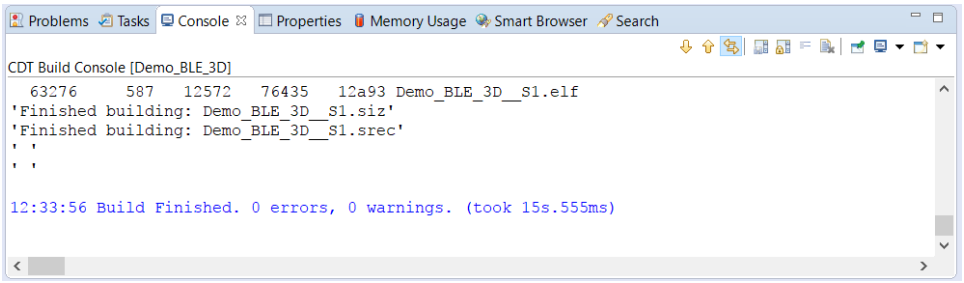


After successful project import, navigate the project tree in `Project Explorer` and double click on project configuration file, `configuration.xml`.



Generate Synergy project contents by pressing the `Generate Project Content` button.

Build the project selecting the `Project` → `Build All` menu. Build process should finish without errors nor warnings (see the `Console` tab below).



In order to download the demo project into S1 device, open the `Run` → `Debug Configurations` dialog, select the `Demo_BLE_3D` configuration and press `Debug` button to confirm.

The IDE automatically moves to the `Debug` perspective. Launch the demo code by pressing the `Resume` button twice.

2.3.2 Silicon Labs MGM111 programming

ARIS 3D demo requires that "`Demo_BLE_3D__MGM.bin`" binary file is loaded into MGM111 device.

Please refer to "ARIS EDGE User Guide" – section 3.3 for the detailed procedure.

3. Design overview

3.1 Functionalities

ARIS EDGE BLE 3D demo implements functionalities as follows:

- Periodically sampling of the on-board sensor:
  - BNO055 9-axis absolute orientation sensor
- Communication with the multiprotocol MGM111 module in order to make the ARIS EDGE board orientation available to external BLE clients.
- Trace the whole system's events using a serial port as debug output.

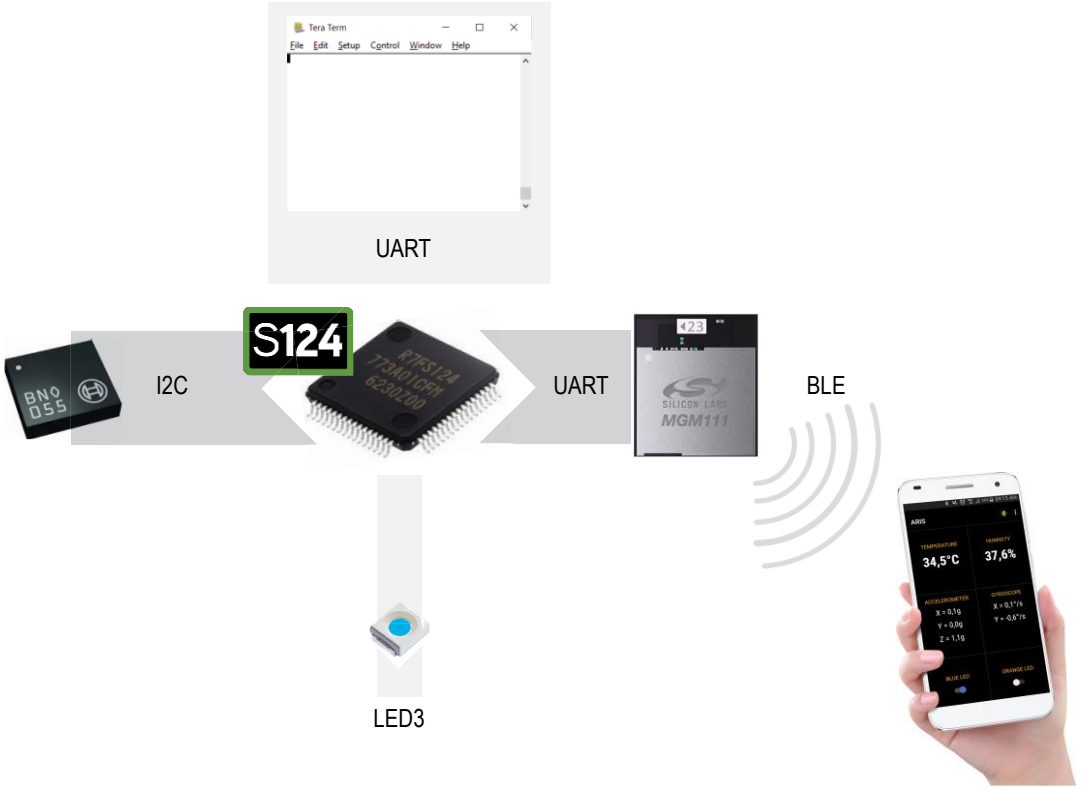


Figure 1 - Demo 3D high level diagram.

3.2 Architecture

In the 3D demo three threads are present. Each one of them handles a specific functionality:

- Sensors Thread: continuously sample the board orientation reading from the BNO055 the quaternions values.
- BLE Thread: its purpose is to control the communication with the MGM111 external module using a serial port.

- Console Thread: periodically reads the orientation and output the quaternions values on the debug serial console.

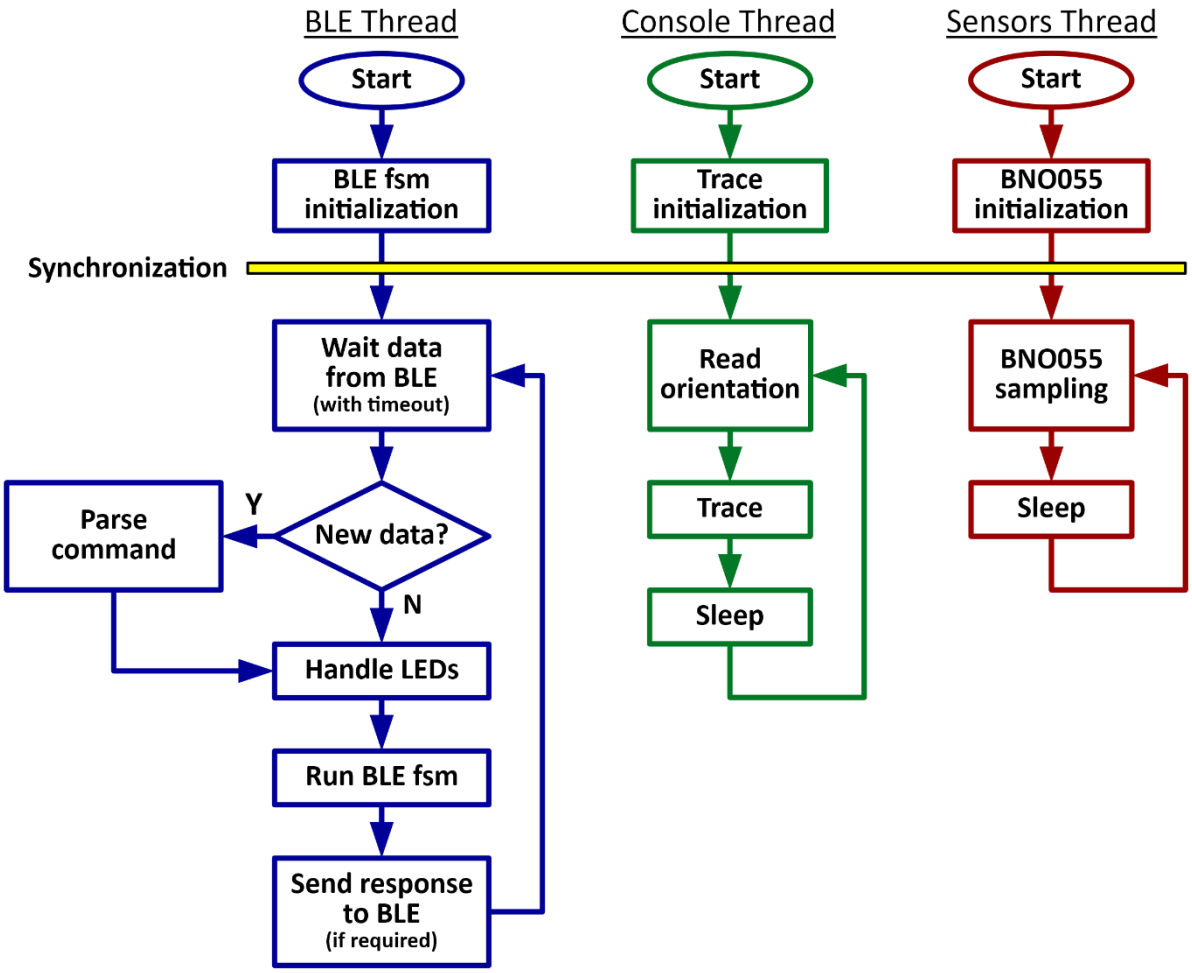


Figure 2 - Threads blocks diagram.

As it can be noticed from Figure 2, a synchronization system is present at boot time in order to wait that all the shared components are correctly initialized before their use. This system is based on a simple event-flags ThreadX primitive.

3.3 Data sharing

In a multi-thread system where data must be accessible from different threads, a mutex protection system must be used.

In this demo an advanced sharing system has been implemented, the “data queue” module. Its main features are:

- Oppositely to the standard synergy messages system, it is allowed to have different polling frequency to the data from different threads. If a thread is too slow to empty the queue, or even stops to read, when there is no more space in the queue this module simply discards the oldest data.

- No owner concept, everyone can write to (or read from) the module.
- Mutex protection for multi-thread applications.
- Possibility to read always the last data only (no queue).

Note that this module has been specifically designed for this application requirements. For example, data loss in case of a slow-reader is acceptable and writers will always be able to add data to the queue (ignoring the queue status).

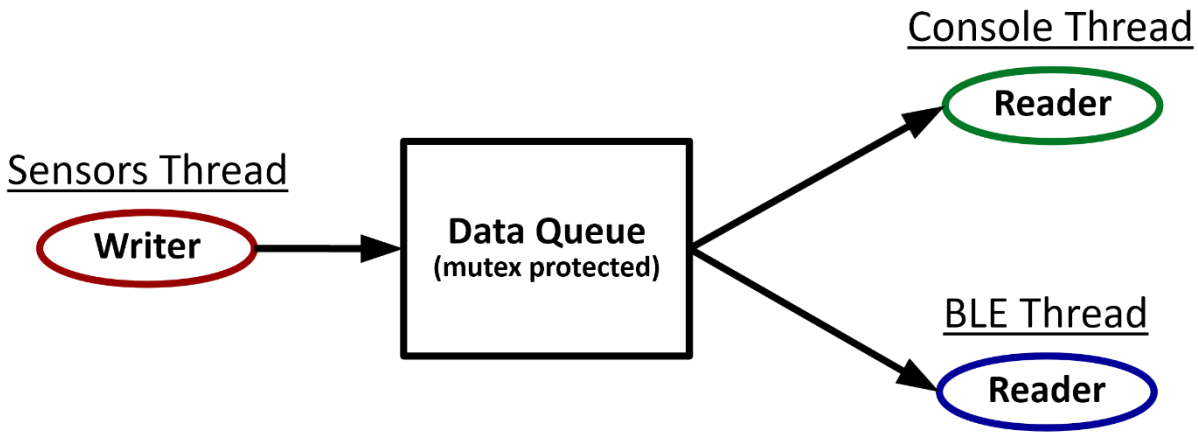


Figure 3 - Data flow.

3.4 BLE interface

The multiprotocol MGM111 has been programmed with a custom firmware that enables the BLE interface and shows these services:

Service UUID	Characteristic	Description
AA40 (custom)	AA41	Quaternion WXYZ data (array of 4 floats)

Clients can subscribe to the characteristic so that they will be automatically notified when the values change. This is the mode used from the mobile APP.

3.4.1 S1 to BLE communication

The protocol between BLE and S1 is made of a predefined fixed-length commands.

The “BLE to S1” commands are used to inform the S1 of some user requests or for acknowledge a previous S1 command. These commands have a 6-char fixed length with a “\r\n” terminator used as alignment control.

BLE to S1	
Command (6 chars)	**Description**
`"K  \r\n"`	Acknowledge to sensor’s data
`"C  \r\n"`	BLE client connected event
`"D  \r\n"`	BLE client disconnected event

The “S1 to BLE” commands are instead the commands sent to the BLE module for data values notifications.

S1 to BLE	
Command (19 chars)	**Description**
`"P  wwwwxxxxyyyyzzzz"`	New orientation value, where: ▪ `wwww` = Quaternion W ▪ `xxxx` = Quaternion X ▪ `yyyy` = Quaternion Y ▪ `zzzz` = Quaternion Z

3.4.2 BLE communication sequence

BLE module has been programmed using BGScript so its reactivity is low.

The commands sent to the module must be alternated with a sleep period and a trivial acknowledge mechanism has been also used.

At boot time no communication is initialized between S1 and BLE module.

The trigger event is a client connection, communicated from the BLE module to the S1 MCU. After this event the S1 starts to read the last sensor’ values from the “data queue” and send them to the

BLE, one value at a time. After each value, the MCU waits for an acknowledge response from the BLE and then sleep for a predefined period before sending the next value.

The loop ends when a disconnection event is received from the BLE module only.

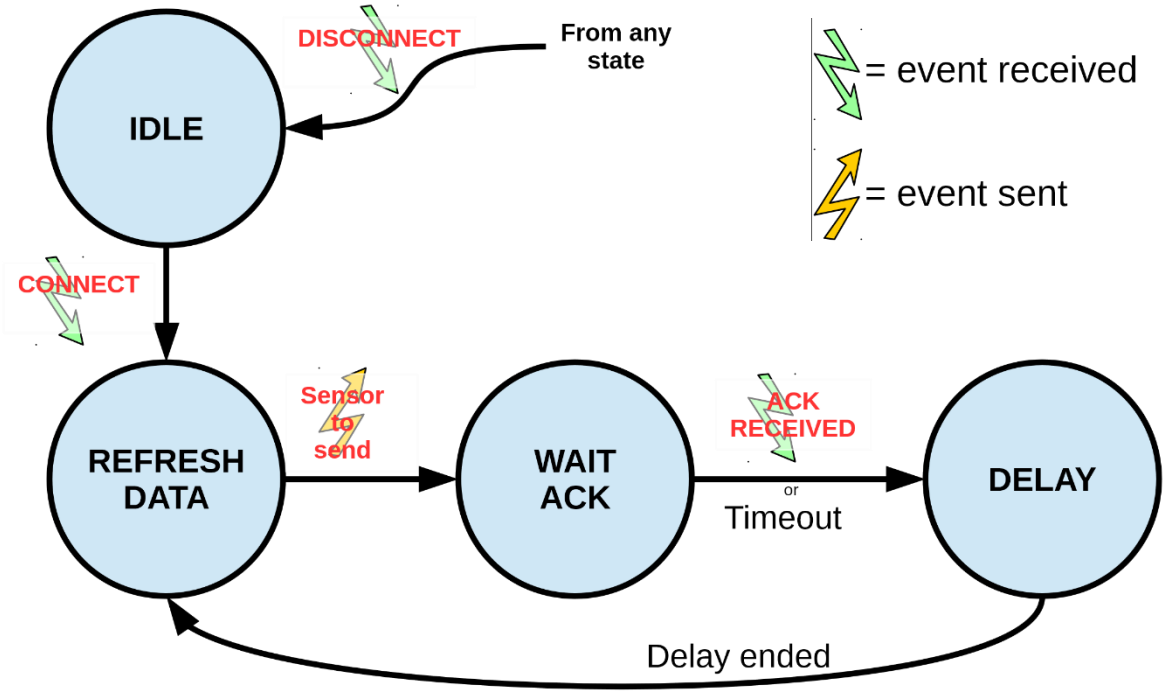


Figure 4 - BLE finite state machine.