Developing Long Range Bluetooth® Smart Devices
Topics

- Bluegiga BLE121LR Bluetooth Smart Long Range Module
- Bluegiga Bluetooth Smart Software
- Hardware Design with BLE121LR
- Developing a long range iBeacon with Bluegiga Bluetooth Smart Software
- iOS and Android Device Considerations
- More Information
- Questions and Answers
• **Bluetooth v.4.0**, single mode compliant
  – Supports master and slave modes
  – Up to 8 connections

• **Integrated Bluetooth Smart stack**
  – GAP, GATT, L2CAP and SMP
  – **Bluetooth** Smart profiles

• **Radio Performance**
  – Transmit power: +8 dBm
  – Receiver sensitivity: -98 dBm

• **Low Current Consumption**
  – Transmit: 36 mA
  – Transmit: 25 mA (with DC/DC)
  – Sleep mode 3: 0.5 uA

• **Flexible Peripheral Interfaces**
  – UART, SPI and I2C serial interfaces
  – PWM, GPIO
  – 12-bit ADC

• **Host Interfaces**
  – UART

• **Host Interfaces**
  – 14.7 x 13.0 x 1.8 mm

• **Programmable 8051 processor for stand-alone operation**

• **Bluetooth**, CE, FCC, IC, South-Korea and Japan qualified
• **World Leading Radio Performance**
  – +8dBm TX power and -98 dBm sensitivity
  – 5-10 x range compared to conventional *Bluetooth* Smart solutions

• **Application Hosting Capability**
  – Application code can be executed on the BLE121LR
  – No need for a separate micro controller
  – Programmable with Bluegiga BGScript™ or C

• **Flash Based**
  – On-the-Field firmware updates over UART or OTA
  – Application data can be stored on the flash

• **Bluetooth, CE, FCC, IC, Japan and Korea Qualifications**
  – Minimal qualification costs
  – Proven interoperability
**Range**

- **World Leading Radio Performance**
  - 5-10 x range compared to conventional *Bluetooth* Smart solutions
**Bluetooth Smart Software**

- **Bluetooth v.4.0, Single Mode Compliant**
  - Supports master and slave modes
  - Up to 8 simultaneous connections

- **Implements all Bluetooth Smart Functionality**
  - GAP, L2CAP, ATT, GATT
  - Security manager: bonding, encryption
  - Bluetooth Smart profiles

- **Simple API for External Host Processors**
  - BGAPI™: A simple protocol over UART or USB interfaces
  - BGLib™: A C library for host processors implementing BGAPI

- **Supports Integrated Applications**
  - BGScript™: A simple scripting language for writing applications
  - Native C application development with IAR Embedded Workbench
  - No separate host needed

- **DFU and OTA Firmware Upgrade Support**

- **Bluetooth Smart Profile Toolkit™**
  - XML based development tool for Bluetooth Smart profiles
  - Fast and simple profile development

- **Small Memory Requirements**
  - ~4-6 kB RAM
  - ~60-90 kB flash (depending of used features/profiles)

- **Bluetooth Qualified**
**Bluetooth Smart Software**

- **BGAPI™ protocol**: A simple binary command, response and event protocol between the host and the stack
  - Used when a separate host (MCU) is used to control BLE121LR over UART
  - Very small memory requirements size requirement and low implementation overhead
Bluetooth Smart Software

- **BGLib™ library**: A portable ANSI C library, which implements the BGAPI protocol
  - Easy to port to various architectures such as: ARM Cortex, PIC16/32 etc.
  - Ported to multiple programming languages: ANSI C, Java, Python and C#
  - Uses function-call back architecture

```c
/* Function */
void ble_cmd_gap_connect_direct(
    bd_addr address,
    uint8 addr_type,
    uint16 conn_interval_min,
    uint16 conn_interval_max,
    uint16 timeout
);

/* Callback */
void ble_rsp_gap_connect_direct(
    uint16 result,
    uint8 conn
);
```
Bluetooth Smart Software

- **BGScript™ scripting language**: A very simple BASIC-like application scripting language
  - Used when applications are implemented on the BLE121LR’s 8051 controller
  - Enables very fast application development and allows programs to be executed directly on the BLE121LR without the need of an external MCU

```plaintext
# System boot event listener: Executed when BLE112 is started
event system_boot(major, minor, patch, build, ll_version, protocol_version, hw )

    # Configure ADV interval to 1000ms and start advertisements an all channels
    call gap_set_adv_parameters(1600, 1600, 7)

    # Start generic advertisement and enable connections
    call gap_set_mode(2,2)

    # Start a continuous software timer, which generates interrupts every 1000ms
    call hardware_set_soft_timer(32768, 1, 0)

end
```
Bluetooth Smart Software

- **Bluetooth Smart Profile Toolkit™**: A tool for creating Bluetooth Smart profiles
  - Bluetooth Smart profiles are very simple
  - Can be described with a single file of XML
  - Profile toolkit is a Simple XML description template for Bluetooth Smart Profiles

- Several example profiles and services available
  - Heart Rate transmitter
  - Proximity reporter
  - Blood glucose sensor
  - iBeacon
  - etc.

```xml
<service uuid="1800">
  <description>Generic Access Profile</description>
  <characteristic uuid="2a00">
    <properties read="true" const="true" />
    <value>BG Demo</value>
  </characteristic>
  <characteristic uuid="2a01">
    <properties read="true" const="true" />
    <value type="hex">4142</value>
  </characteristic>
</service>
```
RF Signal Propagation

- RF signal will attenuate as the distance increases
- The distance of the (transmitter or receiver) from the ground significantly affects the range as can be seen from the chart below
BLE121LR-A Layout Guide

- For optimal performance of the antenna place the module at the edge of the PCB.
- Do not place any metal (traces, components, battery etc.) within the clearance area of the antenna.
- Connect all the GND pins directly to a solid GND plane.
- Place the GND vias as close to the GND pins as possible.
- Do not place plastic or any other dielectric material in touch with the antenna.
Ground vs. Antenna Performance

![Graph showing the relationship between the amount of GND on the sides of the module (mm) and the range (% of maximum). The graph illustrates an upward trend as the amount of GND increases.]
What are Beacons?

iBeacon is Apple’s implementation of Bluetooth Smart technology to create a way of providing location-based information and services to iPhones and other iOS devices.

The beacons are small Bluetooth Smart transmitters. Apps installed the phone can listen out for the signal transmitted by the beacons and respond accordingly when in range.

Beacons can be used to provide users with location for example is a shopping mall or present notifications for example about items in sale.

**Region monitoring**

- Enables iOS device to detect if it enters or exits a specific region. iOS will notify the app if such an event occurs even if the app is in the background or locked.

- A notification can be presented as well even if the app is closed

**Ranging**

- When the App is active ranging can be used to detect and show all discovered iBeacons

- A distance estimation can also be made
iBeacons are so simple devices that they are trivial to implement with BGScript scripting language and therefore it is used in this example.

This example implements an iBeacon with the following functionality:

- iBeacon functionality as defined by the Apple’s specification
- OTA firmware update

Project steps:

1. **Project configuration** – defines the project resources
2. **Hardware and application configuration** – defines the hardware and application properties
3. **GATT database** – defines the services and data exposed by the iBeacon
4. **BGScript code** – implements the application functionality
Project settings

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<project>
  <gatt in="gatt-BLE121LR-M256K.xml" />
  <hardware in="hardware.xml" />
  <config in="config256.xml" />
  <script in="dkble-ble121lr-m256k.bgs" />
  <device type="ble121lr-m256k" />
  <image out="out-ble121lr-m256k.hex" />
  <ota out="BLE121LR-256.ota" />
  <boot fw="bootota" />
</project>
```

- **GATT**: The GATT database file
- **Hardware**: The hardware configuration file
- **Config**: The Application configuration file
- **Script**: The BGScript code
- **Device**: Bluegiga Bluetooth Module Type
- **Image**: Firmware output file
- **Ota**: OTA firmware output image
- **Boot**: Firmware update (DFU) interface
Hardware settings

```xml
<?xml version="1.0" encoding="UTF-8" ?>

<hardware>
    <sleeposc enable="true" ppm="30" />
    <script enable="true" />
    <txpower power="15" bias="5" />
    <pmux regulator_pin="7" />
    <sleep enable="true" />
    <otaboot source="internal" />
</hardware>
```

- Sleeposc: Sleep clock is enabled
- Script: BGScripting is enabled
- TX power: TX power set to maximum value
- pmux: External DC/DC converter enabled
- sleep: Power saving modes are enabled
- otaboot: Internal flash used for OTA firmware update
- Boot: Firmware update (DFU) interface
The GATT database

The GAP service:
- UUID: 0x1800
- Device name characteristic
  - UUID: 0x2a00
  - Read property
- Device Type characteristic
  - UUID: 0x2a01
  - Read property

The OTA service:
- UUID: 128-bit UUID
- OTA control characteristic
  - UUID: 128-bit UUID
  - Write property
  - 1 byte
- OTA data characteristic
  - UUID: 128-bit UUID
  - Write property
  - 20 bytes
The BGScript Code

System Boot event is executed on start-up

- Sets advertisement parameters
- Initializes iBeacon advertisement data
- Starts advertisement

```plaintext
# System boot event listener
event system_boot(major, minor, patch, build, ll_version, protocol_version, hw)

# Set advertisement interval to 125ms.
# Use all three advertisement channels
call gap_set_adv_parameters(200, 200, 7)

# Initialize iBeacon ADV data
# Flags = LE General Discovery, single mode device (02 01 06)
advdata(0:1) = $02
advdata(1:1) = $01
advdata(2:1) = $06
# Manufacturer data
advdata(3:1) = $1a
advdata(4:1) = $ff
# Preamble
advdata(5:1) = $4c
advdata(6:1) = $00
advdata(7:1) = $02
advdata(8:1) = $15
```
OTA update -

- **OTA control** or **OTA Data** attributes are written by the remote device
- **OTA control** characteristic is used to control the OTA process
- **OTA data** characteristic carries the firmware update
  - When data is received, it’s stored to the internal flash and DFU pointer increased

```c
# Handles OTA Control Point Attribute (commands) and OTA Data Attribute (firmware update) writes # and performs the necessary actions
event attributes_value(connection, reason, handle, offset, value_len, value_data)

# Save connection handle, is always 0 if only slave
curr_connection = connection

if handle = ota_control then
  # Attribute is user attribute, reason is always write_request_user
  if value_len > 1 || offset > 0 then
    # Not a valid command -> report application error code : 0x80
    call attributes_user_write_response(connection, $80)
  else
    command = value_data(0:1)
    if command > 4 then
      # Unknown command -> report application error code : 0x80
      call attributes_user_write_response(curr_connection, $80)
    else
      if command = 3 then
        # Command 3 received -> Boot to DFU mode
        call system_reset(1)
      else
        # Other commands are not used, but still accepted in order
        # to be compatible with the external flash OTA
        # implementation
        call attributes_user_write_response(curr_connection, $0)
      end if
    end if
  end if
end if

# Check if OTA data attribute is written which carries the firmware update # and store the data to the internal flash
if handle = ota_data then
  call flash_write_data(dfu_pointer, value_len, value_data(0:value_len))
  dfu_pointer = dfu_pointer + value_len
end if
```
Testing the Application

The application can be tested for example on the DKBLE Bluetooth Smart Development kit.

Apple App Store has multiple applications for testing iBeacon functionality and Beacon Region and Beacon Ranging functionality:
- BLExplr
- Locate Beacon
Testing the Application

OTA Update can for example be tested with the BLEGUI Windows application included in the Bluegiga Bluetooth Smart Software.

The development kits include a BLED112 USB dongle that can be used as the hardware.

BLEexplr application also includes OTA update functionality.
iOS Device Considerations

- iOS can operate as central and peripheral

- Advertise the service UUIDs in advertisement packet
  - The iOS App can filter devices based on UUIDs

- Minimum connection interval ~20ms
  - When App is put to background the connection interval might be increased

- iOS devices cache services
  - Implement the generic GATT service and iOS will refresh services on every connection
iOS Device Considerations

• Need xCode developer license and OSX developer tools
  – Available at the Apple’s developer site

• MFI
  – You do not need to be part of MFI in order to develop Bluetooth Smart Apps for iOS

• Bluegiga example iOS App available
  – Download from www.bluegiga.com
  – Available in source code
Android Device Considerations

- *Bluetooth* Smart APIs available since 4.3
  - In API level 18

- Currently supported devices: Nexus 4, 6, 7 and 9. Multiple devices from other vendors as well

- Android only supports central mode (master)

- Background applications supported

- Android supports secure connections and insecure connections

- Note: Bluetooth Smart implementation is not very robust in 4.3 – multiple improvements in 4.4 and Android 5.
Android Device Considerations

• You need Android Development Kit (ADK)
  – Available on Android developer web site
  – Free-of-Charge

• You need the latest API level 18 access

• Bluegiga example Android App available
  – Download from www.bluegiga.com
  – Available in source code and as APK
More Information

• Bluegiga BLE Software and SDK, Example Applications, Documents and Smart Phone examples
  – Documentation and Downloads

• Bluetooth Smart SDK v.1.2 Introduction
  – Presentation

• Over-the-Air Firmware Update
  – Application Note

• iBeacon example and discussion
  – Example: Bluegiga Forums
  – Discussion: Bluegiga Forums
More Information

• Bluegiga
  – www.bluegiga.com
  – www.bluegiga.com/support

• Bluetooth SIG
  – www.bluetooth.org
  – www.bluetooth.com
  – http://developer.bluetooth.org

• iOS Development
  – iOS Dev Center

• Android Development
  – Android Developers
More Information

- Bluegiga BLE Software and SDK, Example Applications, Documents and Smart Phone examples
  - [Documentation and Downloads](#)

- Over-the-Air Firmware Update
  - [Application Note](#)

- iBeacons example and discussion
  - Example: [Bluegiga Forums](#)
  - Discussion: [Bluegiga Forums](#)
More Information

• Bluegiga
  – www.bluegiga.com
  – www.bluegiga.com/support

• Bluetooth SIG
  – www.bluetooth.org
  – www.bluetooth.com
  – http://developer.bluetooth.org

• iOS Development
  – iOS Dev Center

• Android Development
  – Android Developers
Thank You