Creating a Bluetooth low energy product with BLE112

Tuomas Ilola

Topics

- Sensors and interfaces
- Implementing the application
- Exposing the data in a service

Sensors

- **Common hardware interfaces**
  - Analog IO
  - Digital IO
  - I2C
  - SPI
  - UART
  - USB

- **For simplicity and energy budget the sensors should have**
  - Similar operating voltage than BLE112 (2.0-3.6V)
  - As low peak and average currents as possible

Hardware design

- **Bluegiga provides reference design**
  - Evaluation kit schematics and layout
- **Design guidelines for optimal RF performance**

- **Pin assignment can be done using hardware.xml**
  - Pull up/down for Ios
  - UART/SPI and USB configuration
  - TX power level
  - ...

```
<hardware>
    <sleeposc enable="true" ppm="30" />
    <usb enable="false" endpoint="none" />
    <txpower power="15" bias="5" />
    <usart channel="1" alternate="1" baud="57600"  endpoint="api" />
    <wakeup_pin enable="true" port="0" pin="0" />
    <port index="0" tristatemask="0" pull="down" />
</hardware>
```

- **Bluegiga provides design validation**
  - Check the layout for possible RF problems

Application

- **BLE112 has space for customer applications**
  - No need for host controller
- **Two ways to create applications**
  - BGScript – supported with current version
  - C-code – support in roadmap
- **BGScript application can**
  - Control the Bluetooth stack
    - Make device visible
    - Connect to devices
  - Transfer data over Bluetooth
    - Write data to GATT database
    - Read and write data over the BT link
  - Control hardware interfaces
    - Read/Set the state of Ios
    - Send/receive data over communication interfaces (UART/SPI/USB)
- **Stack will handle power optimization**
  - Always in the lowest possible sleep mode

Application

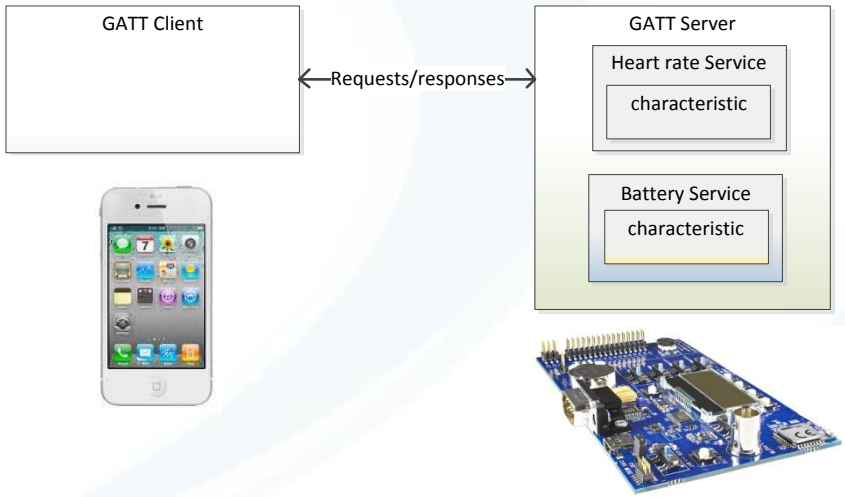
- **BGScript applications are simple and fast to develop**
  - Low line count
    - 29 lines of code to make ”FindMe” tag with display
    - ~15 lines without the display
  - Isolated from the Bluetooth stack
    - No risk at affecting Bluetooth interoperability
- **Bluegiga provides example implementatios for various use cases**
  - Health thermometer
  - Heart rate
  - SPI examples
    - Display
    - Accelometer
  - Numerous example code snipplets for different use cases

Heart rate demo

```
dim tmp(10)

event system_boot(major,minor,patch,build,ll_version,protocol,hw)
    call gap_set_mode(gap_general_discoverable,gap_undirected_connectable)
    call sm_set_bondable_mode(1)
    call hardware_set_soft_timer(32000,0,0)
end

event hardware_soft_timer(handle)
    #first read battery level
    call hardware_adc_read(15,3,0)
end

event hardware_adc_result(input,value)
    #battery level reading received, store to gatt and read potentiometer level
    if input = 15 then
        call attributes_write(xgatt_battery,0,2,value)
        #read potentiometer, decimation 128, use avdd5 as reference
        call hardware_adc_read(6,1,2)
    end if
    if input = 6 then
        #potentiometer value is measured

        #build simple characteristic value response
        tmp(0:1)=2
        #calculate some valid hr value 20-224
        tmp(1:1)=value/160+20

        call attributes_write(xgatt_hr,0,2,tmp(0:2))
    end if
end

event connection_disconnected(handle,result)
    call gap_set_mode(gap_general_discoverable,gap_undirected_connectable)
end
```

Profiles

- When the data is received from the sensors it need to be transferred over the air
- Data transactions in BLE happen using GATT database
- GATT database can be defined using XML language
  - Simple to read
  - Simple to make
  - Simple to re-use
- Bluegiga offers example GATT databases for different profiles



Profile XML part 1 – Heart rate

```
<configuration>

    <service uuid="1800">
      <description>Generic Access Profile</description>
      <characteristic uuid="2a00">
        <properties read="true" const="true" />
        <value>Bluegiga Heart Rate Demo</value>
      </characteristic>
      <characteristic uuid="2a01">
        <properties read="true" const="true" />
        <value type="hex">4142</value>
      </characteristic>
    </service>

    <service type="primary" uuid="180A" id="manufacturer">
        <characteristic uuid="2A25">
            <properties read="true" const="true" />
            <value type="hex">000780c0ffeef00d</value>
        </characteristic>
        <characteristic uuid="2A24">
            <properties read="true" const="true" />
            <value>modelnumber</value>
        </characteristic>
        <characteristic uuid="2A29">
            <properties read="true" const="true" />
            <value>Bluegiga</value>
        </characteristic>
    </service>
```

Profile XML part 2 – Heart rate

```
    <service uuid="180d">
      <description>HR demo</description>
      <include id="manufacturer" />
      <characteristic uuid="2a37" id="xgatt_hr">
        <properties notify="true" />
        <value length="2" />
      </characteristic>
    </service>

    <service uuid="e001">
      <description>Battery status</description>
      <include id="manufacturer" />
      <characteristic uuid="e101" id="xgatt_battery">
        <properties read="true" />
        <value length="2" />
      </characteristic>
    </service>
</configuration>
```

Compile

- **Using Bluegiga profile development toolkit you can compile**
  - Hardware definitions – hardware.xml
  - BGScript Application – script.bgs
  - GATT database – gatt.xml
  - Bluegiga single mode bluetooth stack

  into one hex file that you can upload to the BLE112

- **Modules can be ordered with your firmware preinstalled**

Summary

- **Define hardware interfaces**
  - hardware.xml
- **Create your application**
  - BGScript (C-support coming later)
- **Form your GATT database**
  - gatt.xml

- **Creating your own Bluetooth low energy prototype can be achieved in minutes, rather than days or months**
  - Time to market
  - Development cost
  - Certified


bluegiga
Thank you!
www.bluegiga.com