BLE Shield for Arduino

Dr. Michael Kroll

June 26, 2012

Agenda

- Bluetooth Low Energy in a "very small" nutshell
- BLE Shield for Arduino
  - Intention for the Project
  - Description of the Hardware
  - Description of the Firmware (BGScript)
- Demo

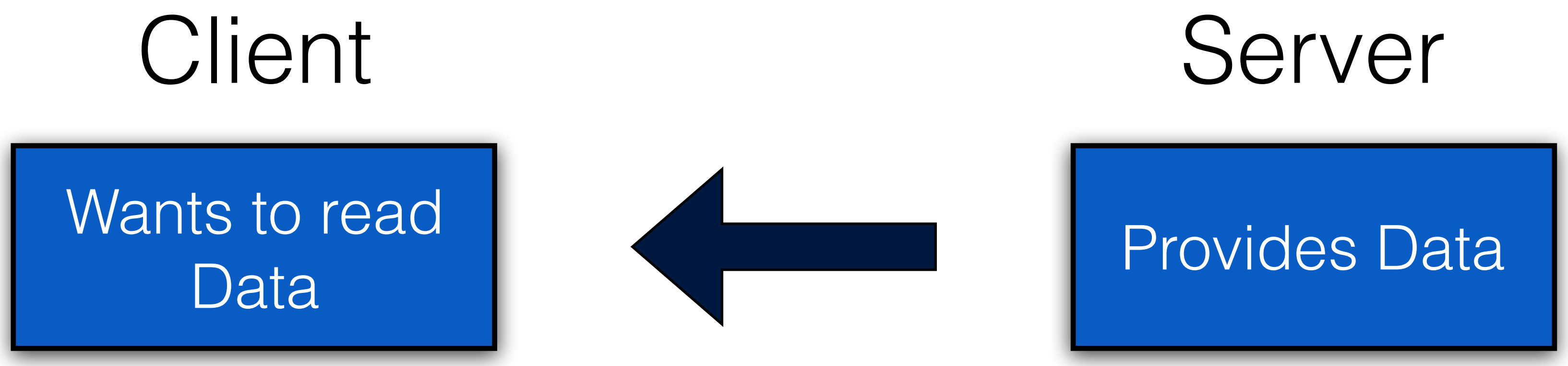
Bluetooth Low Energy

- Less time on the air
- Less energy when on the air resulting in small amouts of data which can be transferred
- Completely new architecture
- Not compatible with classic Bluetooth

BLE Terms: "Modes"

- Dual Mode
  - Supporting Bluetooth Classic and Bluetooth Low Energy such as Notebooks and Smartphones
- Single Mode
  - Bluetooth LE enabled peripherals e.g. Polar H7 Heart Rate Sensor or the BLE112 Evaluation Board

BLE Terms: "Client & Server"

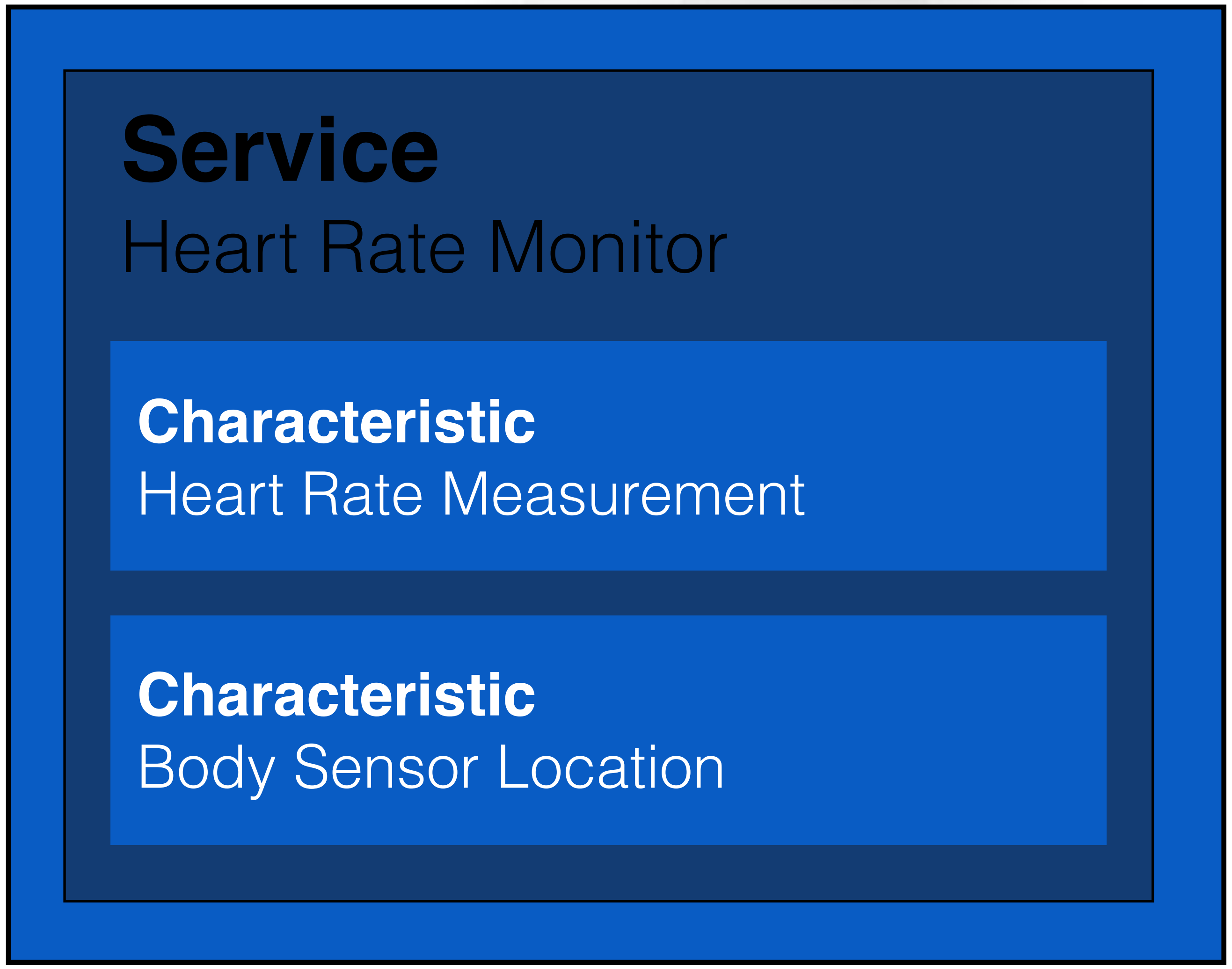


Desktop Computer,
Notebook or Smartphone

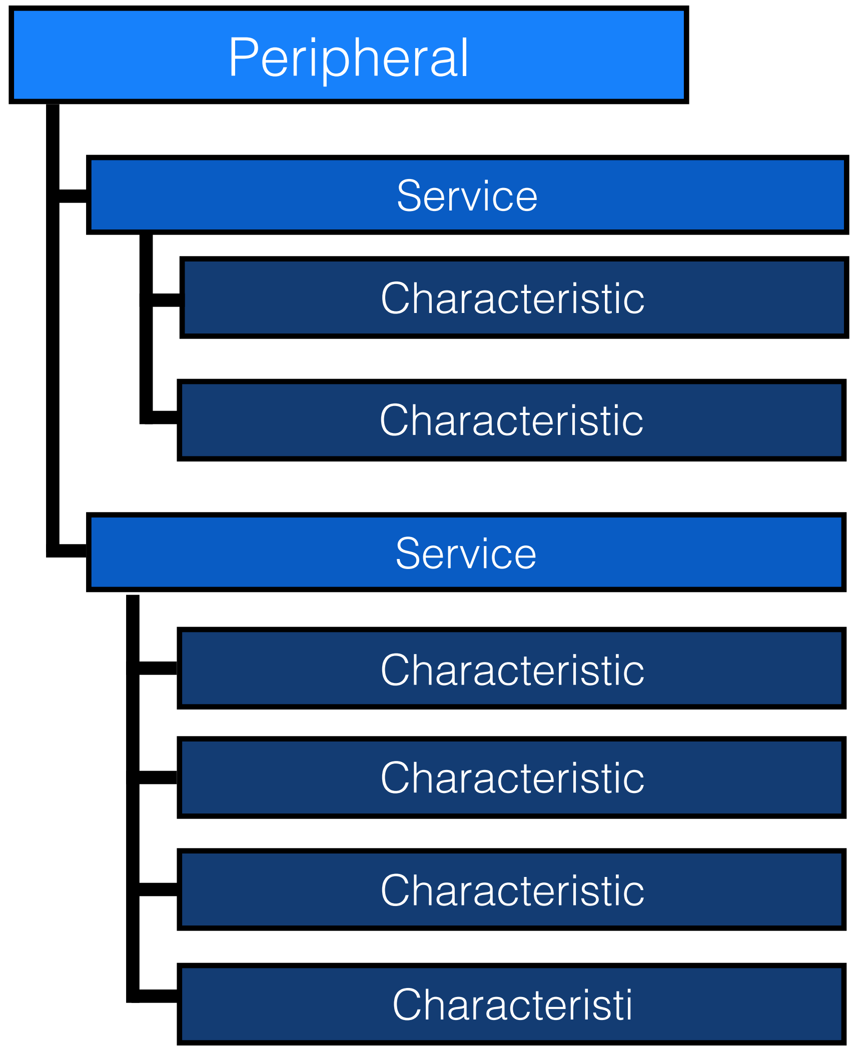
Heart Rate Sensor

Services and Characteristics

Server



Tree



More on BLE

- http://www.bluetooth.org
- iOS related WWDC 2012 Sessions
  - 703 - Core Bluetooth 101
  - 705 - Advanced Core Bluetooth

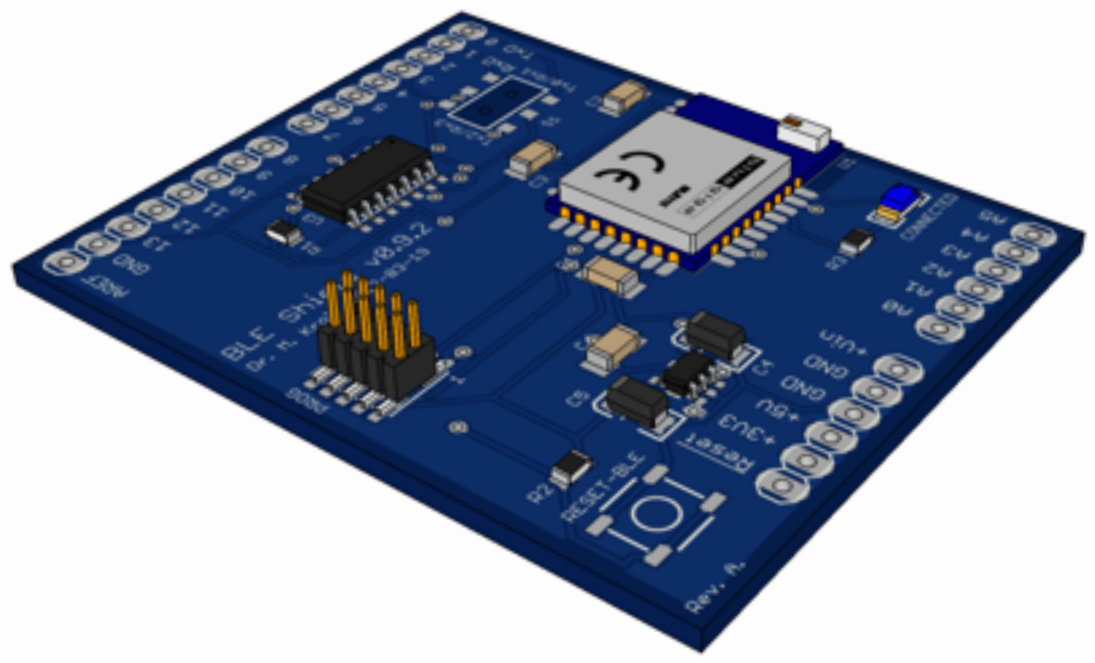
Intention for the Project

With iOS5.0 Apple introduced the CoreBluetooth Framework initially supported on the iPhone4S

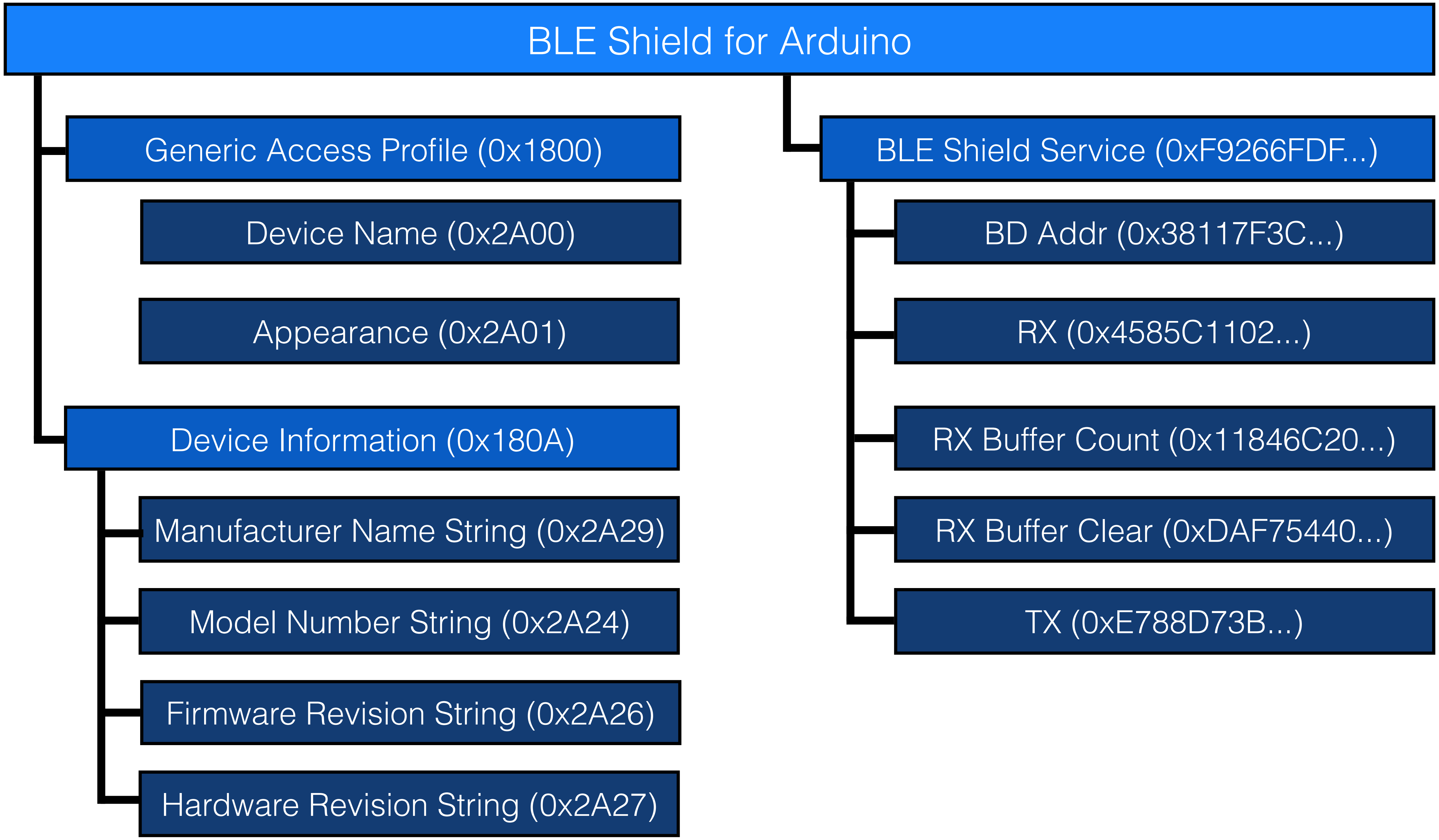
- BLE peripherals are NOT part of MFi
- enabling the iPhone4S to cummunicate with BLE enabled Peripherals
- Why not create a BLE Shield for Arduino?

BLE Shield for Arduino

- Bluegiga BLE112 module
- Reset switch
- Rx/Tx can be set to Pin 0/1 or 2/3
- Compatible with 3.3V and 5V Arduino boards
- Programming header to flash the BLE112 firmware



Services and Characteristics


BLE Shield for Arduino
Generic Access Profile (0x1800)
Device Name (0x2A00)
Appearance (0x2A01)
Device Information (0x180A)
Manufacturer Name String (0x2A29)
Model Number String (0x2A24)
Firmware Revision String (0x2A26)
Hardware Revision String (0x2A27)
BLE Shield Service (0xF9266FDF...)
BD Addr (0x38117F3C...)
RX (0x4585C1102...)
RX Buffer Count (0x11846C20...)
RX Buffer Clear (0xDAF75440...)
TX (0xE788D73B...)

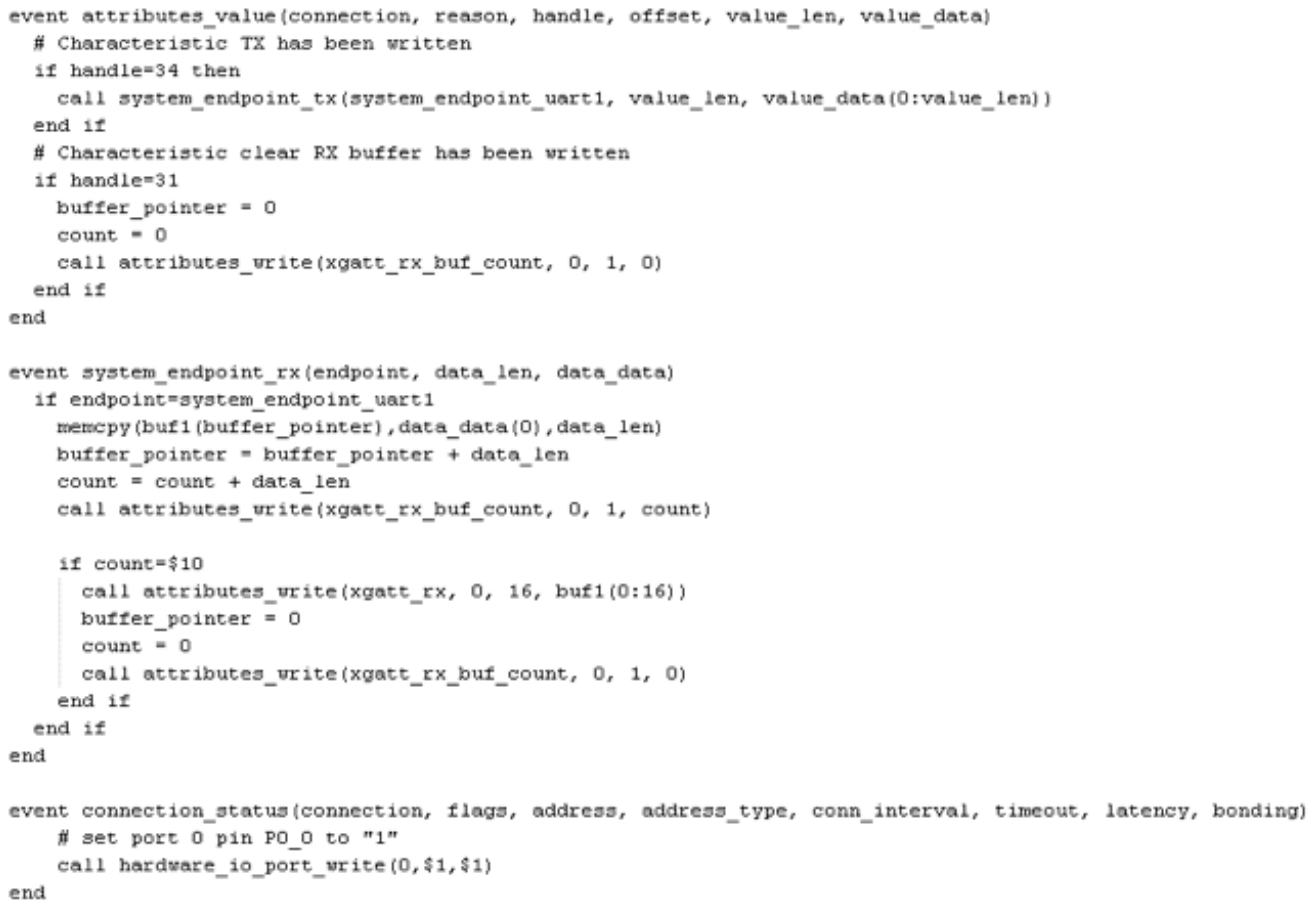
BLE112 Firmware

- Programmed in BGScript provided by BlueGiga in 60 lines of code + XML Config and GATT
- Software Buffer of 16 bytes that is notified to the client if filled.
- Buffer Count to read how many bytes has been received
- Buffer clear to reset the internal buffer count.

Firmware Part 1

```
dim buf1(16)
dim count
dim buffer_pointer

event system_boot(major, minor, patch, build, ll_version, protocol, hw)
    # set port 1 to output
    call hardware_io_port_config_direction(0, $f)
    # set port 1 pin P1_0 to "0" which is used for the blue Connection LED
    call hardware_io_port_write(0, $1, $0)
    #set to advertising mode
    call gap_set_mode(gap_general_discoverable,gap_undirected_connectable)
    #set the buffer pointer to 0
    buffer_pointer=0
    #set the received counter to 0
    count=0
    #set bondable mode
    call sm_set_bondable_mode(1)
end

event connection_disconnected(connection ,reason)
    #connection disconnected, continue advertising
    call gap_set_mode(gap_general_discoverable,gap_undirected_connectable)
    # set port 0 pin P0_0 to "0"
    call hardware_io_port_write(0, $1, $0)
end
```

Firmware Part 2



```
event attributes_value(connection, reason, handle, offset, value_len, value_data)
  # Characteristic TX has been written
  if handle=34 then
    call system_endpoint_tx(system_endpoint_uart1, value_len, value_data(0:value_len))
  end if
  # Characteristic clear RX buffer has been written
  if handle=31
    buffer_pointer = 0
    count = 0
    call attributes_write(xgatt_rx_buf_count, 0, 1, 0)
  end if
end

event system_endpoint_rx(endpoint, data_len, data_data)
  if endpoint=system_endpoint_uart1
    memcpy(buf1(buffer_pointer),data_data(0),data_len)
    buffer_pointer = buffer_pointer + data_len
    count = count + data_len
    call attributes_write(xgatt_rx_buf_count, 0, 1, count)

    if count=$10
      call attributes_write(xgatt_rx, 0, 16, buf1(0:16))
      buffer_pointer = 0
      count = 0
      call attributes_write(xgatt_rx_buf_count, 0, 1, 0)
    end if
  end if
end

event connection_status(connection, flags, address, address_type, conn_interval, timeout, latency, bonding)
    # set port 0 pin P0_0 to "1"
    call hardware_io_port_write(0,$1,$1)
end
```

Preparing the Demo

- In order to run the demo an Arduino needs to be installed with a simple testing sketch
- To see the services and characteristics which has been added to the shield an iPhona App needs to be installed

Arduino Sketch

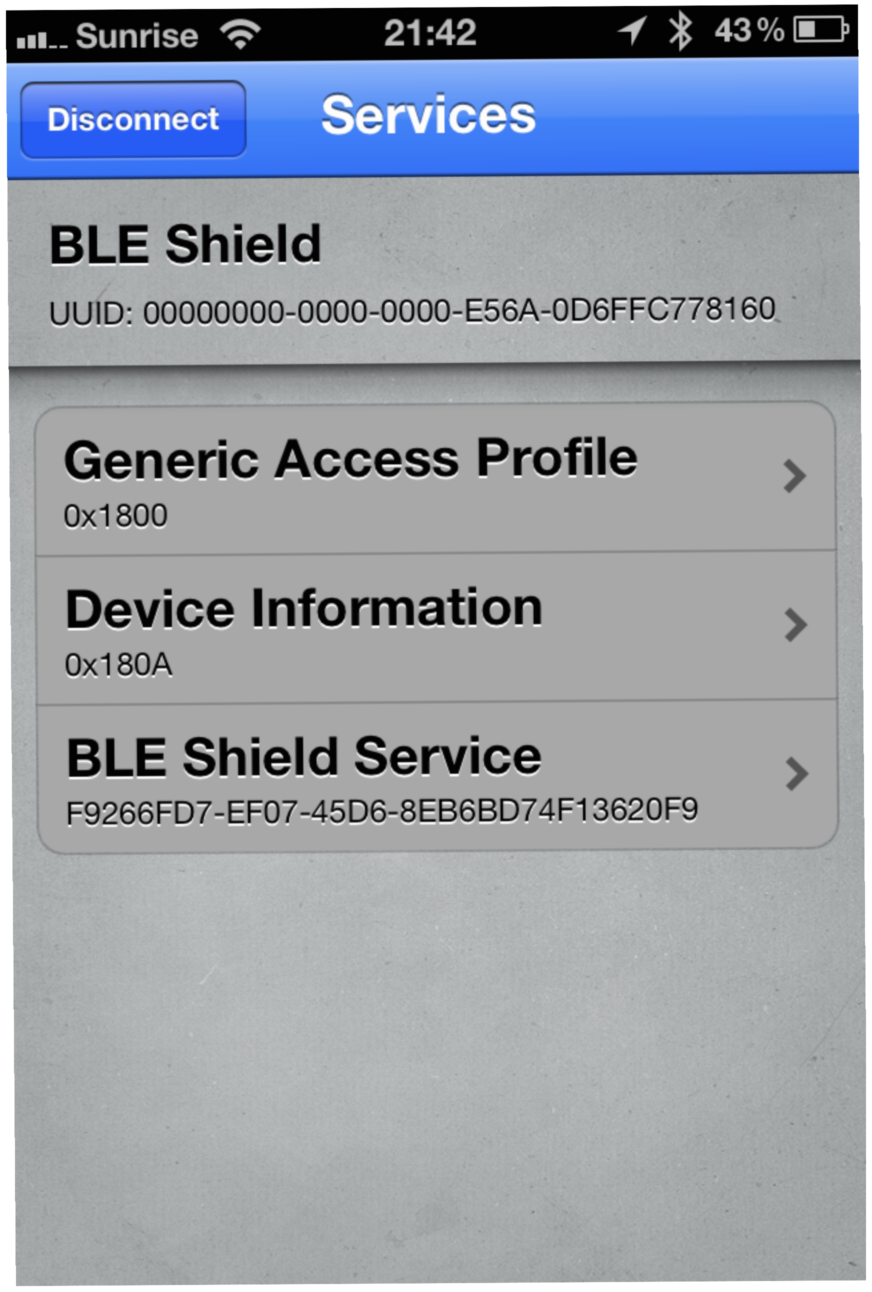
Send 4 bytes random data per second

```
/*
 * BLE Shield Test Sketch for Arduino 1.0
 * by Dr. Michael Kroll 2012
 */
#include <SoftwareSerial.h>

SoftwareSerial bleShield(2, 3);

long previousMillis = 0;
long interval = 1000;

void setup()
{
  // set the data rate for the SoftwareSerial port
  bleShield.begin(19200);
  Serial.begin(19200);
  randomSeed(analogRead(0));
}
```

```
void loop() // run over and over
{

  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis > interval) {
    previousMillis = currentMillis;

    int randNumber1 = random(255);
    int randNumber2 = random(255);
    int randNumber3 = random(255);
    int randNumber4 = random(255);

    bleShield.write(randNumber1);
    bleShield.write(randNumber2);
    bleShield.write(randNumber3);
    bleShield.write(randNumber4);
  }

  if (bleShield.available()) {
    Serial.write(bleShield.read());
  }
}
```

iPhone app BLExplr

- Generic App to discover, connect and read/write Services and Characteristics
- Available in the App Store
- Is used in the Demo to show the BLE Shield in action



BLE Shield
UUID: 00000000-0000-0000-E56A-0D6FFC778160

Generic Access Profile
0x1800

Device Information
0x180A

BLE Shield Service
F9266FD7-EF07-45D6-8EB6BD74F13620F9

Contact

Dr. Michael Kroll

dr.michael.kroll@gmail.com

http://www.mkroll.mobi

Beta Testers in Zurich wanted!

There will be a pre production run of 10 shields im order to get feedback before building hundreds.

Demo