Simple Bike Computer[1]

doing something useful with BLTE as a geek

git clone https://github.com/deadfalkon/simple-bike-computer-presentation.git

pull requests are welcome!

---

[1] Presentation made with DecksetApp

about me

@volkersfreunde

github.com/falkorichter

mobile software Ingenieur

work@sensorberg

about.me/falkorichter

Content

- What is BTLE
- Simple Bike Computer
  - Hardware/setup - bike
  - Hardware - phone
- Swift iOS/MacOSX
- optional: Android

BTLE

Bluetooth Low Energy®, a.k.a. Bluetooth Smart®
* simple low energy data transfer
* send simple bits of data fast
* don´t sent alot of data
* easy binding

BTLE

To help consumers identify compatibility and ensure connectivity with products and applications incorporating Bluetooth® Core Specification version 4.0 (or higher), the Bluetooth SIG has developed the Bluetooth Smart and Bluetooth Smart Ready trademarks.[5]

[5] bluetooth.org/en-us/bluetooth-brand/how-to-use-smart-marks

BTLE basics

Services

Services are collections of characteristics and relationships to other services that encapsulate the behavior of part of a device.[2]

[2] developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx

BTLE basics

Characteristics

Characteristics are defined attribute types that contain a single logical value.[3]

[3] [developer.bluetooth.org/gatt/characteristics/Pages/CharacteristicsHome.aspx)[https://developer.bluetooth.org/gatt/characteristics/Pages/CharacteristicsHome.aspx]

BTLE basics

Descriptors

"Descriptors are defined attributes that describe a characteristic value."[4]

[4] developer.bluetooth.org/gatt/descriptors/Pages/DescriptorsHomePage.aspx

Why a bike computer?

I ride my bike
It´s not too expensive
I want to know how far I go each week/month/year
health foo #ftw

Ingredients

Speed and Cadence Sensor

Ingredients

Ingedients

Amazon: "Geschwindigkeit und Trittfrequenz"

Any Mac / iPhone > 4S

Remove the label o the device with nail polish remover

Speed and Cadence

- official Profile all
- Cycling Speed and Cadence
  - org.bluetooth.service.cycling*speed*and_cadence mandatory
  - org.bluetooth.service.device_information optional

It´s all nicely documented.

Swift

everything is a pretty easy

stack (almost) identical on iOS & MacOSX (I had a compilation error on the same code)

characteristic.value() vs characteristic.value

Hello world of BTLE: heartRate

org.bluetooth.service.heart_rate

1 Service

1 Value (the heart rate)

-> simulate beeing a heart rate sensor

-> simulate connecting to a heart rate sensor

github.com/falkorichter/swift-simple-bike-computer

Swift become a peripheral:[6][7]

```
func startBroadcasting(){
    heartRateService.characteristics = [hearRateChracteristic]
    infoService.characteristics = [infoNameCharacteristics]

    peripheralManager.addService(infoService)
    peripheralManager.addService(heartRateService)
    var advertisementData = [
        CBAdvertisementDataServiceUUIDsKey:[infoService.UUID, heartRateService.UUID],
        CBAdvertisementDataLocalNameKey : "mac of falko"
    ]
    peripheralManager.startAdvertising(advertisementData)
}
[...]
```

[6] done: github.com/deadfalkon/swift-simple-bike-computer/blob/master/Shared/HeartBeatPeripheral.swift

[7] needed: github.com/deadfalkon/swift-simple-bike-computer/blob/master/Shared/SpeedAndCadencePeripheral.swift

Swift get notified :1 [8]

```
let CSC_SERVICE = CBUUID(string: "1816")
let CSC_MEASUREMENT  = CBUUID(string: "2A5B")

func centralManagerDidUpdateState(central: CBCentralManager!){
    switch (central.state){
    case .PoweredOn:
        central.scanForPeripheralsWithServices([CSC_SERVICE], options: nil)
    default:
        println("not powered on")
    }
}
```

[8] https://github.com/deadfalkon/swift-simple-bike-computer/blob/master/Shared/CadenceConnector.swift

Swift get notified :2

```
func centralManager(central: CBCentralManager!, didConnectPeripheral peripheral: CBPeripheral!){
    peripheral.discoverServices([CSC_SERVICE])
}
```

Swift get notified :3

```
func peripheral(peripheral: CBPeripheral!, didDiscoverServices error: NSError!){
    if(error != nil) {
        for service in peripheral.services {
            if service.UUID == CSC_SERVICE {
                peripheral.discoverCharacteristics([CSC_MEASUREMENT], forService: service as CBService)
            }
        }
    }
}
```

Swift get notified :4

```swift
func peripheral(peripheral: CBPeripheral!, didDiscoverCharacteristicsForService service: CBService!, error: NSError!) {
    if(error != nil){
        if service.UUID == CSC_SERVICE {
            for characteristic in service.characteristics{
                if (characteristic as CBCharacteristic).UUID == CSC_MEASUREMENT {
                    peripheral.setNotifyValue(true, forCharacteristic: characteristic as CBCharacteristic);
                }
            }
        }
    }
}
```

Swift get notified :5

```
func peripheral(peripheral: CBPeripheral!, didUpdateValueForCharacteristic characteristic: CBCharacteristic!, error: NSError!) {
    //do your thing
}
```