Trends der drahtlosen Kommunikation
Kapitel 4B: Bluetooth Low Energy

Prof. Dr. Wolfgang Mühlbauer

wolfgang.muehlbauer@th-rosenheim.de

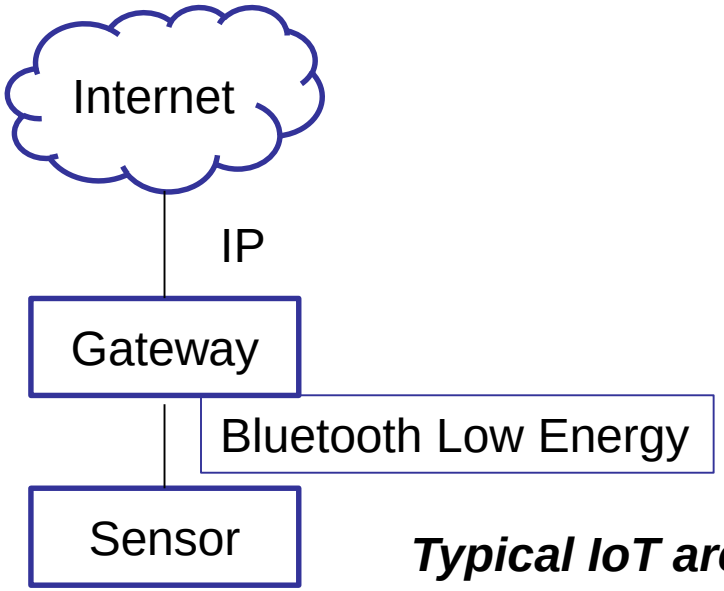
Fakultät für Informatik

Sommersemester 2019

Some slides are based on [1]

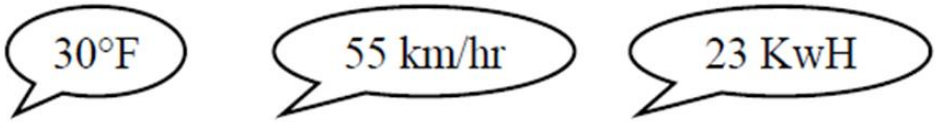
Motivation

- **Internet of Things (IoT)**
  - Small wireless sensors
    - Temperature, humidity
  - Wearable sensors
    - Heart rate, blood pressure
  - Actuators
    - Remote-controlled switches and lights, etc.



Typical IoT architecture

- ***Core requirements***
  - Very limited battery capacity (coin cell!)
  - Infrequent transmission of information is sufficient.
  - Small data volumes.
- Sensors and actuators generally do not implement heavy TCP/IP



Bluetooth Classic vs. Bluetooth Low Energy (BLE)

Problem with Bluetooth Classic: Power consumption was still too high!!

Bluetooth Classic

- Version 1.0 up to now
- Based on connections
- Continuous data stream (e.g., voice, file transfer)
- Supported by smartphones and notebooks

Bluetooth Low Energy ("Wibree")

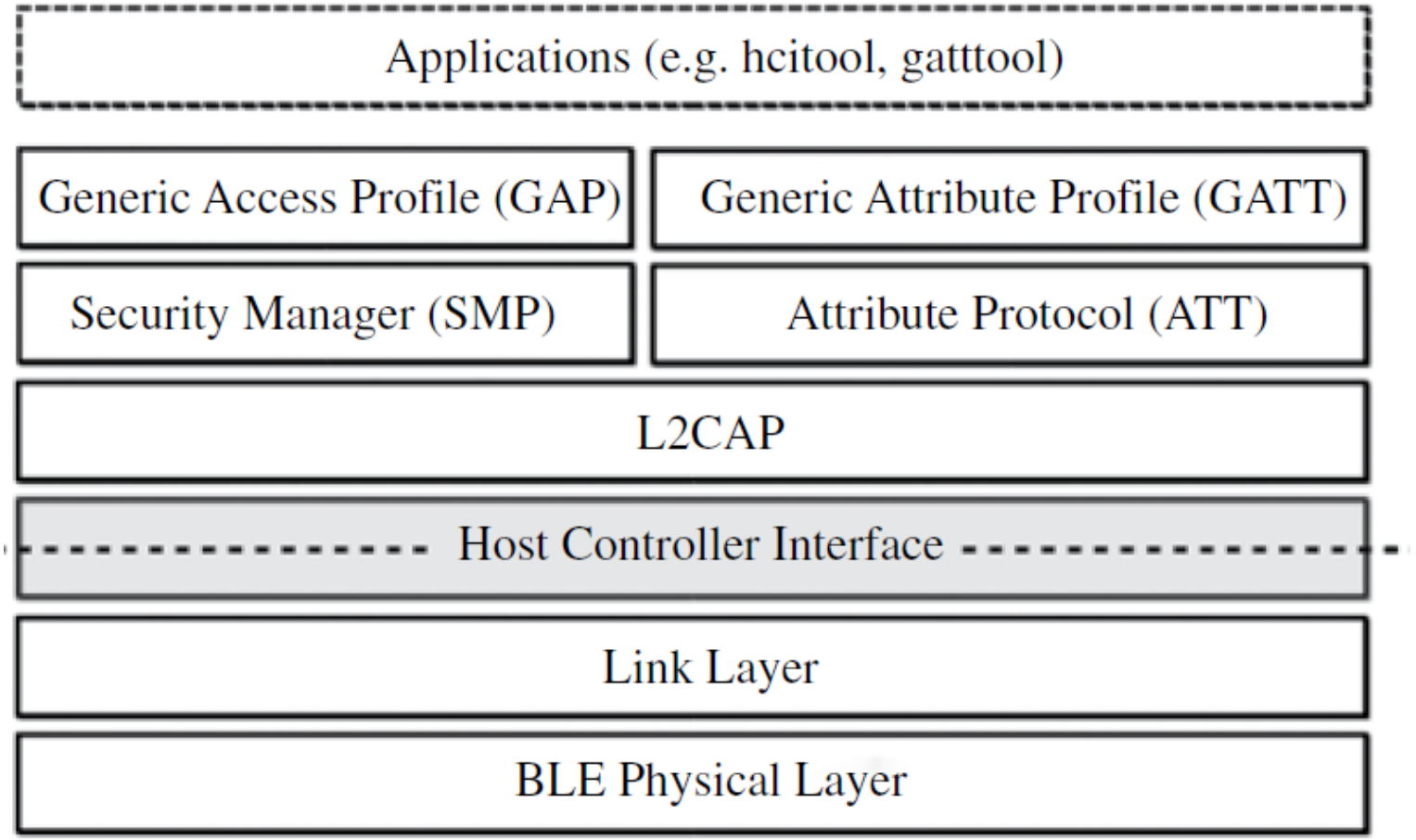
- Since Version 4.0
- No real connections, communications as short as possible.
- Reading/writing values of a remote device, distributing values sporadically
- Supported by smartphones, notebooks AND sensors

Bluetooth® SMART READY

Bluetooth SMART

BLE Protocol Stack

Source: [4]

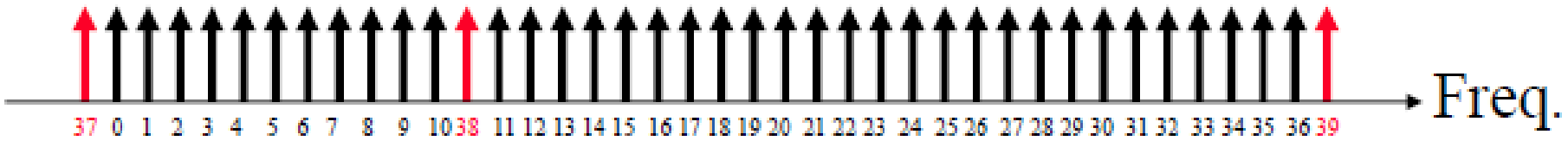


- New physical layer and link layer

BLE Physical Layer

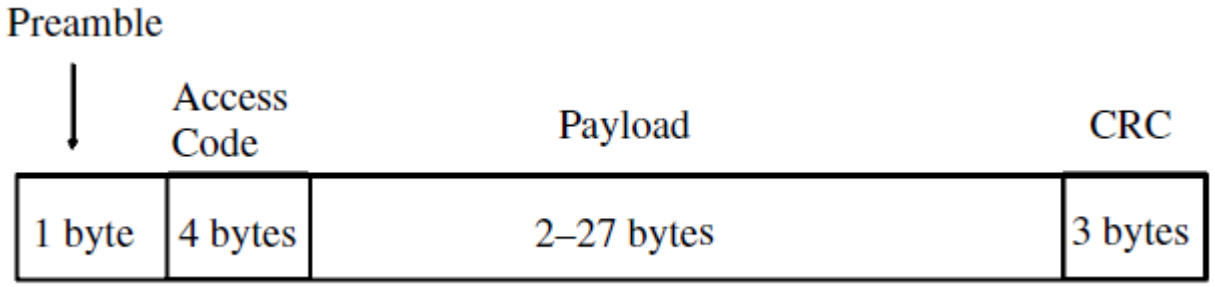
- ***Shared with*** Bluetooth Classic
  - Frequency band: 2,4 GHz, range up to 150 m
  - Adaptive frequency hopping

- ***Differences*** to Bluetooth Classic
  - 40 instead of 80 channels, each with 2 MHz instead of 1 MHz
  - 3 out of 40 channels reserved for advertisements, connection establishment.
  - Fixed data rate of 1 Mbps per channel
  - Slower hopping rate: Channel only changed at the beginning of a connection event (typically every 7,5 ms and 4 s).
  - Even during a connection, no continuous data transmission.



Simple design of BLE Link Layer

- Tasks
  - Framing, encryption
- Single frame format, no ACL or SCO
  - 4 byte access code is randomly generated for a connection
- ***Roles of a device***
  - ***General advertising****:* Broadcast presence via advertisements. Remote devices can connect to it to obtain further data, see below.
  - ***Non-connectable beacon****:* Broadcast data directly, no connection possible
  - ***Scanners****:* Device that listens for advertisement packets.

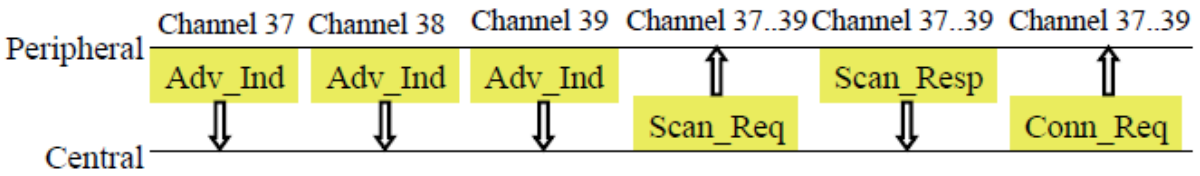


BLE 4.0/4.1 link layer packet

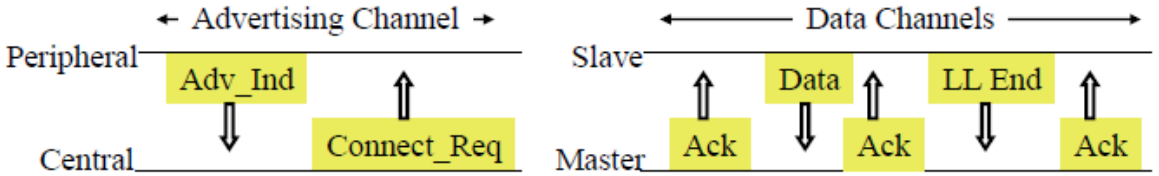
Link Layer: General Advertising Approach

- Advertisements are always sent on the reserved 3 channels
- **Scanning**
  - ***Active***: Scanning device can request additional info without a connection (Scan_Req)
  - ***Passive***: Scanning device only uses info from advertisements.
- ***Terms***
  - ***Central*** establishes a connection, ***slave*** accepts a connection.

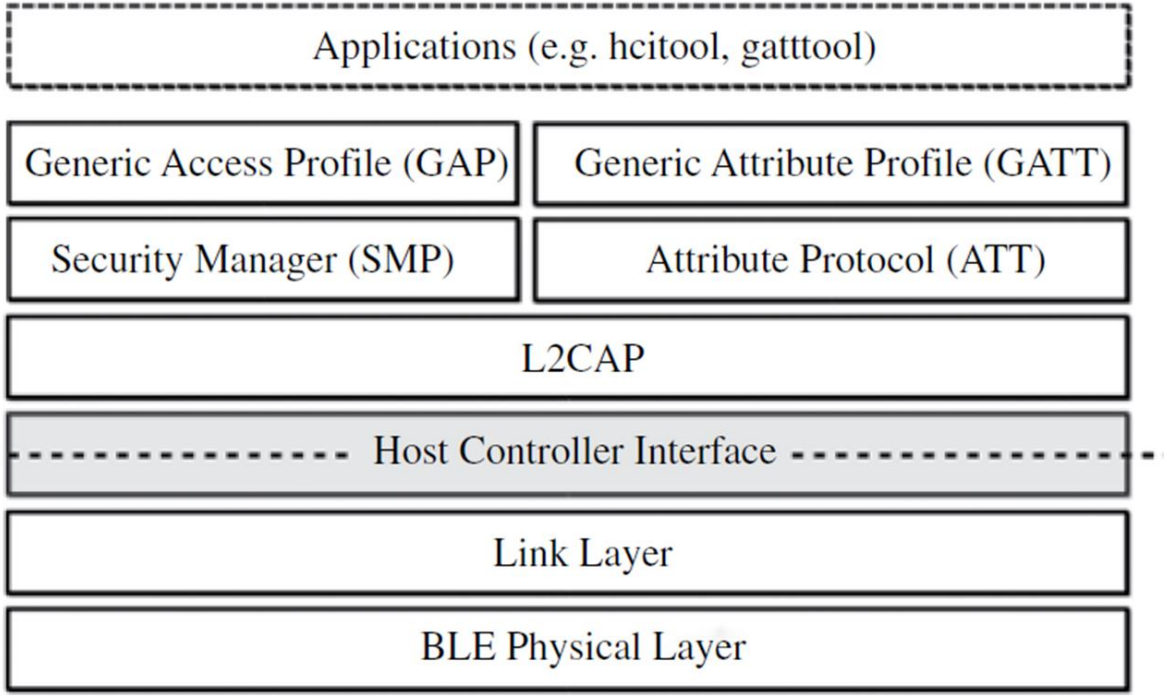
Active scanning with later connection establishment



Connection setup and data exchange



Bluetooth Low Energy: Profiles

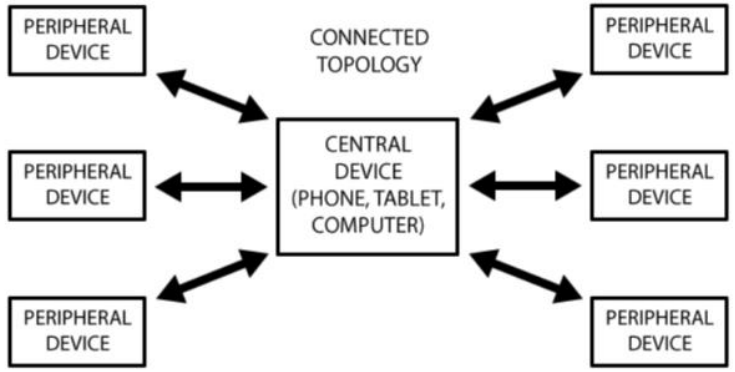


Source: [4]

- ***L2CAP*** is similar as TCP
  - Several applications can communicate over same link-layer connection.
- ***Security Manager Protocol (SMP)*** and ***Generic Access Protocol (GAP)***
  - Exchange of management info: Configure link, connection parameters, etc.
- ***Attribute Protocol (ATT)*** and ***Generic Attribute Protocol (GATT)***
  - Exchange of user data: reading and writing values to and from remote variables

BLE GAP and Connection Establishment

- ***Generic Access Profile (GAP):*** Defines procedures to
  - broadcast data
  - discover devices
  - establish connections
  - authentication / secure connections



- ***GAP roles***
  - Not connected / broadcaster
  - Not connected / observer
  - Connected / initiator of connection == **Central** (e.g., smartphone)
  - Connected / target of connection == **Peripheral** (e.g., sensor)

- ***Connections***
  - Connect Request messages defines parameters for connection events, e.g., time interval.
  - Only at periodic time intervals ("connection events") bidirectional communication between central and peripheral: Data and ACK
  - Channel change (frequency hopping) with each connection event.

Generic Attribute Profile (GATT)

- GATT is the only used profile over BLE.
- ***Specifies how to exchange user data*** over a BLE connection
  - Every transferred item of data is formatted, packed and sent using GATT.
- GATT relies on the ***Attribute Protocol (ATT)***
  - ATT allows a server to expose a set of attributes to a client.
  - Client can discover, read and write these attributes.
- ***Roles***
  - ***Gatt Client*:**
    - sends requests
    - receives responses (and server-initiated updates).
    - generally has to inquire about the presence and nature of attributes.
  - ***Gatt Server*:**
    - Answers requests.
    - Sends server-initiated updates.
- ***GAP vs. GATT***
  - GAP defines low-level interactions with devices, e.g., connection management.
  - GAP roles are independent of GATT roles.

Universally Unique Identifiers (UUID)

- 128-bit number that is globally unique.
  - Used by Bluetooth and other protocols to identify attributes.
- Specified in ITU-T Rec. X.667
- To avoid overhead, shorter UUIDs (16 and 32 bit) exist
  - Vendors need to register to obtain such short UUIDs.
  - 128-bit UUID can be reconstructed from shorter UUID types, see below.
  - Examples: https://www.bluetooth.com/specifications/gatt/services/
- Unregistered vendor-specific UUIDs
  - Full 128-bit UUID must be used at all times

UUID structure

```
Short version(16 bit) :   0000xxxx
Short version(32 bit) :   xxxxxxxx
Short version(128 bit):   xxxxxxxx-0000-1000-8000-00805F9B34FB
```

Attributes

- Smallest data entity defined by GATT.
  - Attributes generally stored on GATT server.
  - Yet: local copies frequently kept on GATT client, e.g., Android.

- Attributes are composed of the following fields.
  - ***Type***
    - Indicated by UUID, see https://www.bluetooth.com/specifications/gatt/services/
    - Attributes can be discovered based on type.
    - > 1 attribute with same UUID can exist.
  - ***Handle***
    - 16-bit address for each attribute on a particular GATT server.
    - Never changes during a connection, (rarely changes between connections).
    - Client can discover attributes within a handle range, e.g., 0xFF00 – 0xFF10
  - ***Permissions***
    - Access permissions (read/write), encryption, authorization
  - ***Value***
    - Actual content
    - Can store additional information about other attributes (*metadata*) or actual “useful data”.

Attributes in ATT

- Completely flat.
- Set of attributes contained in a GATT server is conceptually like a table.

Handle	Type	Permissions	Value	Value length
0x0201	$UUID_1$ (16-bit)	Read only, no security	0x180A	2
0x0202	$UUID_2$ (16-bit)	Read only, no security	0x2A29	2
0x0215	$UUID_3$ (16-bit)	Read/write, authorization required	"a readable UTF-8 string"	23
0x030C	$UUID_4$ (128-bit)	Write only, no security	{0xFF, 0xFF, 0x00, 0x00}	4
0x030D	$UUID_5$ (128-bit)	Read/write, authenticated encryption required	36.43	8
0x031A	$UUID_1$ (16-bit)	Read only, no security	0x1801	2

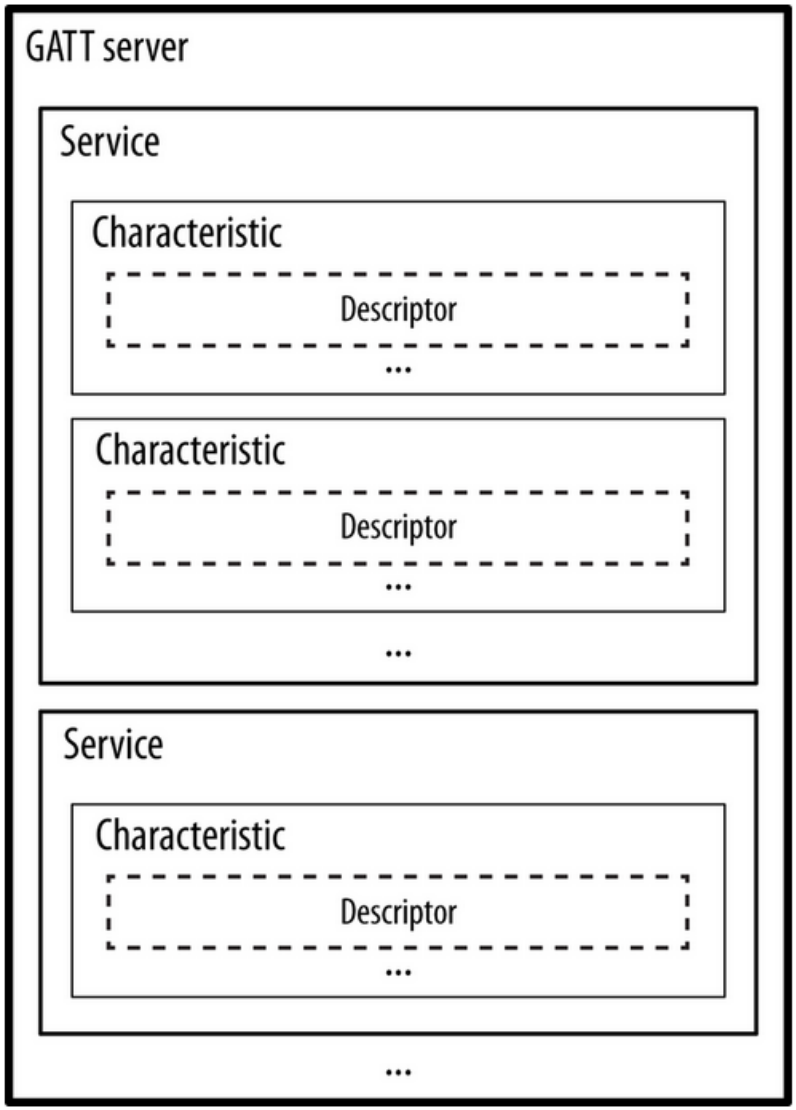
Ficticious GATT table
Source: [7]

Attribute and Data Hierarchy

- GATT organizes attributes in a hierarchy (unlike ATT)

- ***Service***
  - Groups conceptually related attributes (“characteristics”).
  - E.g.: weather station has temperature, humidity service, etc.
  - **Service declaration**: 1st attribute (lowest handle) in group
    - Type: UUID is always 0x2800.
    - Value: describes the service, see https://www.bluetooth.com/specifications/gatt/services/
    - Each service contains multiple characteristics, see below.

- ***Characteristics***
  - “Container” for user data, Includes at least 2 attributes
  - ***Characteristic Declaration Attribute***: Mandatory 1st attribute
    - Type: UUID is always 0x2803.
    - Value: properties, handle and UUID of this particular Characteristic Value (== next attribute)
  - ***Characteristic Value Attribute***
    - Contains actual data value
    - Generally next handle after Characteristic Declaration.



Source: [7]

Example: Sensor Tag CC2650

- Device that provides weather services (humidity, temperature, etc.)
- 0x23: Service Declaration
  - UUID always 0x2800
- 0x23: Characteristic Declaration of first characteristic within service
  - UUID always 0x2803
  - Provides hints on permissions, handle of data (2nd byte) and contains UUID (big endian!)
- 0x25: Characteristic Value of first characteristic within service

Handle UUID Value First characteristic

0x21	33	0x2803	GATT Characteristic Declaration	02:22:00:50:2A	R	PnP ID
0x22	34	0x2A50	PnP ID	01:0D:00:00:00:10:01	R	
0x23	35	0x2800	GATT Primary Service Declaration	F000AA00-0451-4000-B000-000000000000	R	IR Temperature Service
0x24	36	0x2803	GATT Characteristic Declaration	12:25:00:00:00:00:00:00:00:00:B0:00:40:51:04:01:AA:00:F0	R	IR Temperature Data
0x25	37	0xAA01	IR Temperature Data	00:00:00:00	RN	ObjectLSB:ObjectMSB:AmbientLSB:AmbientMSB

Notifications

- Most GATT servers can send ***server-initiated updates.***
  - Inform GATT client asynchronously when a Characteristic Value changes.
- GATT servers implement a switch to enable/disable server-initiated updates.
  - Client can decide if it wants to be informed about certain changes.
- **Client Characteristic Configuration**
  - Special attribute type that acts as a switch to enable/disable notifications.
  - 2-bit field that can be set and cleared by client.
  - UUID: 0x2902.
- Details, see attribute table of CC2650.

0x26	38	0x2902	Client Characteristic Configuration	01:00	RW	Write "01:00" to enable notifications,

BLE Applications

- Proximity
  - in car, in room 303, in the supermarket
- Locator
  - Keys, watches, animals
- Health devices
  - Heart rate monitor, physical activities monitors, thermometer
- Sensors
  - Temperature, battery status, tire pressure
- Remote control
  - Open/close locks, turn on lights

BLE Beacons

- **Approach**
  - BLE devices that only broadcast information, no BLE connections.
  - Info included in broadcasts:
    - UUID: Unique for each beacon device!
    - Transmit power level of beacon, i.e. their presence
  - Smartphones detect signals from multiple beacons, leveraging this info.

- **Possible Applications**
  - Indoor positioning
  - Link to a website if certain UUIDs are received.
  - Mobile marketing: After position has been identified, send commercial to user.
  - Geo-fencing, e.g. detect if child leaves a designated area

BLE Beacons

- **Standards**
  - Apple: **iBeacon**
  - Google: **Eddystone** / Google Beacon Platform
  - https://www.blueupbeacons.com/docs/WorkshopBeacons.pdf (sehr gutter Link)

- **Approach**
  - Deploy beacons, i.e. BLE “peripherals” that permanently announce their presence
  - Beacon uniquely identified by UUID
  - Central devices (e.g., cell phone) detect beacon and trigger an action

- **Standards**
  - Apple: iBeacon
  - Google: Eddystone

Quellenverzeichnis

[1] A. Tanenbaum, D. Wetherall. *Computer Networks*, Fifth Edition, Pearson, 2014
dt. Ausgabe: "*Computernetzwerke"*

[2] http://dilbert.com/strip/2012-05-21, abgerufen am 08.11.2016

[3] Handyfunktionen: Anzahl der Personen in Deutschland, die die Bluetooth-/Infrarotschnittstelle ihres Handys oder Smartphones nutzen, von 2013 bis 2015 (in Millionen), IFD Allensbach, ACTA, 2015

[4] M. Sauter, *From Gsm to Lte-advanced Pro and 5g: An Introduction to Mobile Networks and Mobile Broadband,* Chapter 7, 3. Auflage, John Wiley & Sons Inc., 2017 (eBook)

[5] https://commons.wikimedia.org/wiki/File:Coin-cells.jpg, abgerufen am 08.11.2016

[6] https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt, abgerufen am 08.11.2016

[7] https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html, abgerufen am 06.05.2019