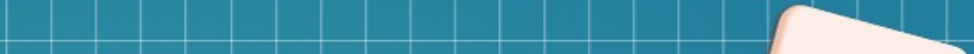
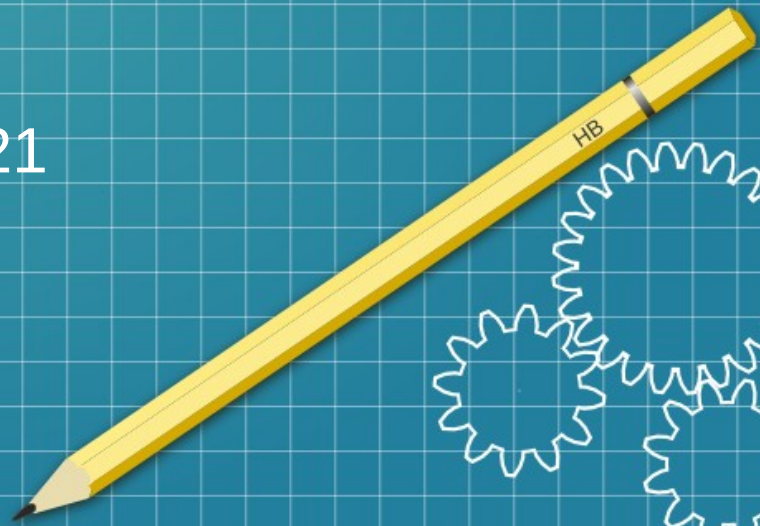
Bluetooth & BLE

Rob Probin, 6 April 2021





What is BLE?

- BLE = Bluetooth Low Energy = Bluetooth LE
  - formerly marketed as Bluetooth Smart
- Different than classic Bluetooth
- **No** RFCOMM/SPP (serial port profile)
  - You probably know SPP from the HC05 module
  - ***BLE has no standard serial (stream) interface***
- Everything is based on attributes / characteristics
  - Basically an **'object-based'** approach
- That doesn't stop many people (hobbyists and professionals) making it work as a serial port!

Conceptually: Low Power, Low Cost, '***Sensor***' type protocol.

Why BLE has become popular?

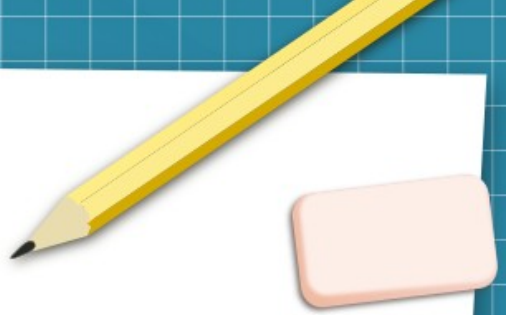
- Basically Apple & iOS and **Classic** Bluetooth
  - Apple, for its iOS based products, iPhone® and iPad®, only support some specific Bluetooth profiles/protocols
  - For classic Bluetooth they require authorisation
  - In order to communicate with an iOS device via Bluetooth, must obtain full approval and certification from Apple before they are manufactured and made available to the public.
- However, BLE is basically open to do what you want with … on ***all*** platforms.

BLE Operating System Support

OS*	Support
IOS 5 and later	Excellent on more modern versions
Mac OS 10.10*	Very good
Linux (incl. Raspberry Pi)	Seems very good. (BlueZ 5.0)
Android 4.3 and later*	Very good
Win 10	Good – using this, seems to work fine.
Previous Windows	Windows 8 has ‘native support’ – but I wouldn’t like to support any of them.
Windows Phone / BlackBerry	Apparently has ‘native support’ – but again, who knows what that means.

WARNING: Different hardware has an effect on BLE support, as does the OS version. List of iPhone model’s BLE support is well documented. Other platforms...

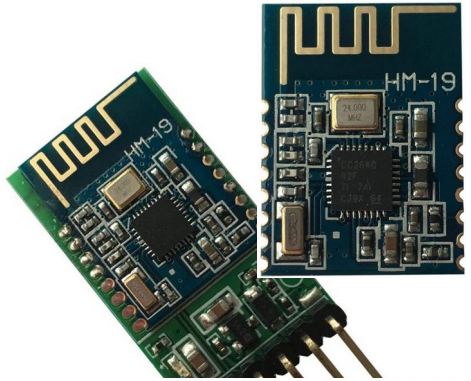
BLE Concepts

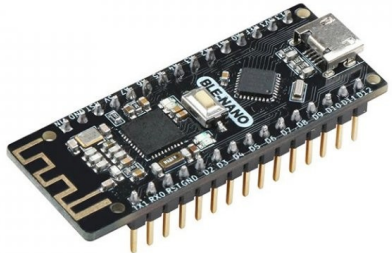


- Central (e.g. Phone, Computer)
- Peripheral (e.g. Device, like UKMarsbot)
- All GAP/GATT build on attributes
- Advertising (can use for broadcasting data!) [GAP]
- Connected (read/write attributes) [GATT]
- Characteristics/Attributes
  - Description, properties, hierarchy of services
- UUID 16-bit (standard) and 128-bit (open/custom)
  - Summary: Just make your own 128-bit numbers
  - https://www.bluetooth.com/specifications/assigned-numbers/
- There are a few profiles - basically specify services & attributes provided.
  - Most profiles are boring https://en.wikipedia.org/wiki/Bluetooth_Low_Energy

Serial over BLE

- Although not standardised, you obviously can transmit serial over BLE
- e.g.
  - Nordic Serial “Nordic UART Service” (NUS)
    - Semi-standard for Nordic devices
    - e.g. Fanstel
  - Other:
    - HM19 (Ti chip)
    - Keywish Nano BLE (Ti Chip)

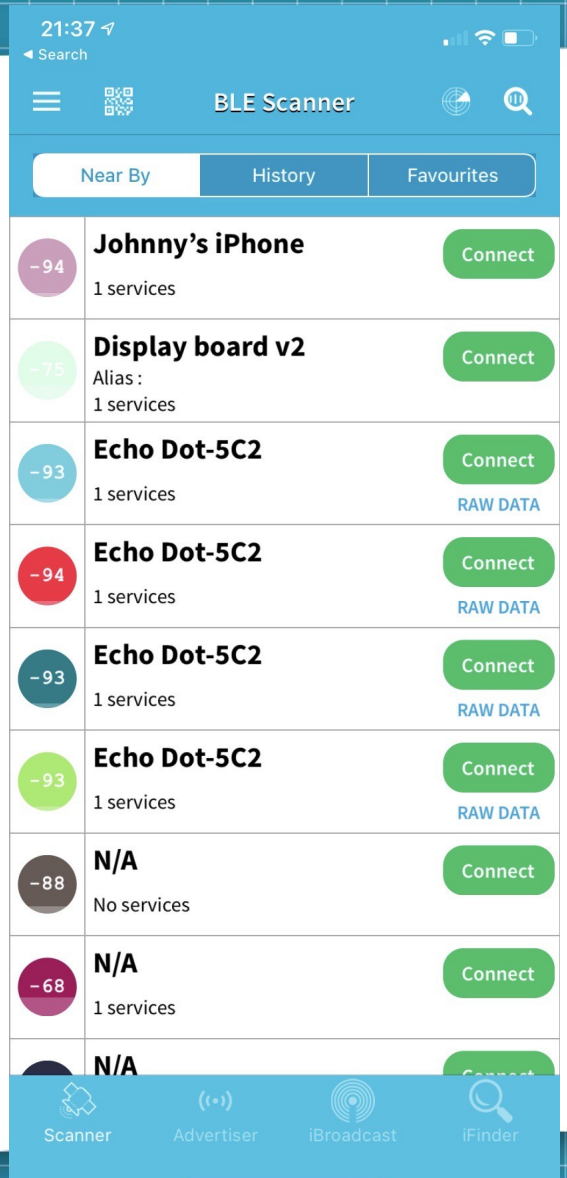


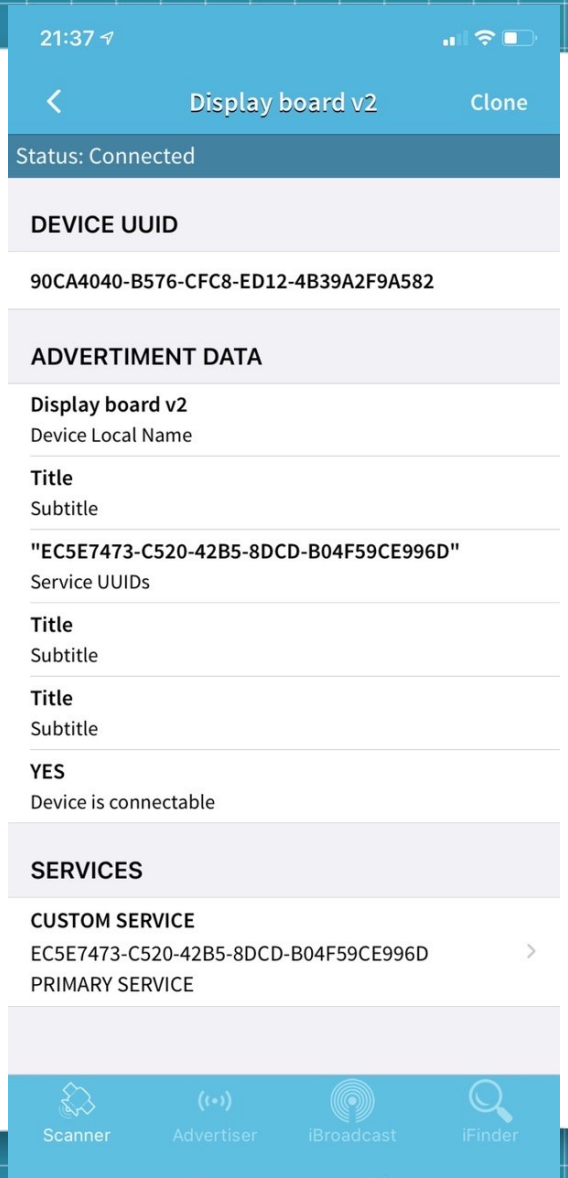


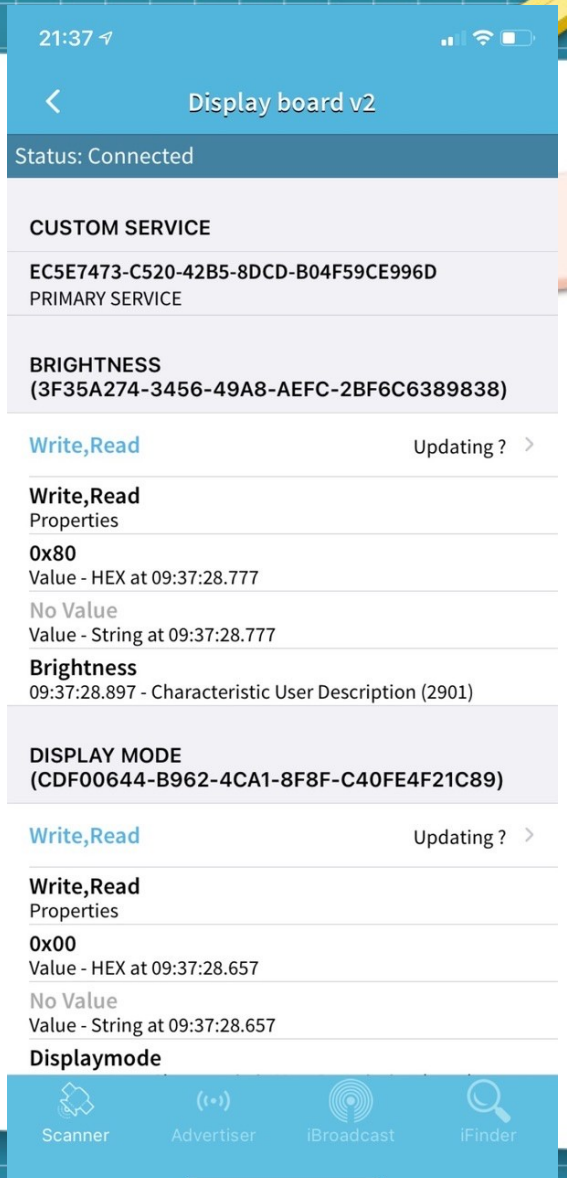
- Why bad idea? Because splits data model & data owner from stack – more transmissions, more power, more latency. (But sometimes it’s the right solution...)
- <Add characteristics details here>

IOS/Android Generic BLE apps

- Interesting, if you are using characteristics/attributes, these are very good.
  - BLE Scanner
  - LightBlue by Punch Through
  - TI ‘Starter’
  - Nordic nRF Connect


21:37
Search
BLE Scanner
Near By
History
Favourites
-94
Johnny's iPhone
1 services
Connect
-75
Display board v2
Alias :
1 services
Connect
-93
Echo Dot-5C2
1 services
Connect
RAW DATA
-94
Echo Dot-5C2
1 services
Connect
RAW DATA
-93
Echo Dot-5C2
1 services
Connect
RAW DATA
-93
Echo Dot-5C2
1 services
Connect
RAW DATA
-88
N/A
No services
Connect
-68
N/A
1 services
Connect
N/A
Scanner
Advertiser
iBroadcast
iFinder


21:37
Display board v2
Clone
Status: Connected
DEVICE UUID
90CA4040-B576-CFC8-ED12-4B39A2F9A582
ADVERTIMENT DATA
Display board v2
Device Local Name
Title
Subtitle
"EC5E7473-C520-42B5-8DCD-B04F59CE996D"
Service UUIDs
Title
Subtitle
Title
Subtitle
YES
Device is connectable
SERVICES
CUSTOM SERVICE
EC5E7473-C520-42B5-8DCD-B04F59CE996D
PRIMARY SERVICE
Scanner
Advertiser
iBroadcast
iFinder


21:37
Display board v2
Status: Connected
CUSTOM SERVICE
EC5E7473-C520-42B5-8DCD-B04F59CE996D
PRIMARY SERVICE
BRIGHTNESS
(3F35A274-3456-49A8-AEFC-2BF6C6389838)
Write,Read
Updating ?
Write,Read
Properties
0x80
Value - HEX at 09:37:28.777
No Value
Value - String at 09:37:28.777
Brightness
09:37:28.897 - Characteristic User Description (2901)
DISPLAY MODE
(CDF00644-B962-4CA1-8F8F-C40FE4F21C89)
Write,Read
Updating ?
Write,Read
Properties
0x00
Value - HEX at 09:37:28.657
No Value
Value - String at 09:37:28.657
Displaymode
Scanner
Advertiser
iBroadcast
iFinder

Example code on Arduino Nano 33 BLE

- LED Display 2
  - Another xmas project :-)
- <View code on GitHub>
  - https://github.com/robzed/LED_Display2
- Applicable to Robots!

Changes

- Bluetooth 4.2
  - DLE – data link extension ….. basically data payloads bigger than 20 bytes.
  - Biggest payload (via MTU) has to be negotiated.
    - Not just stack, but per device / central limitations..
  - Can fragment at L2CAP layer, but there are problems with dropped fragments so lots of implementations handle it above characteristics.
- Bluetooth 5:
  - Support for 2Mbit PHY
  - Support for longer distance ‘Coded’ (=slower by combining symbols)
  -
- NOTE: Transfer speeds is a mixture of MTU, speed, and inter-frame time.

Libs/Tools for Programming BLE

- Desktop Cross-platform
  - Python https://github.com/adafruit/Adafruit_CircuitPython_BLE
    - Pretty basic, but seems to work on Windows.
  - Lots of single platform tools, but worry about their support.
- Mobile cross-platform
  - <fill this in>
- Tools:
  - Windows: Terrible, write your own
  - Mobile (see previous slides) - ***excellent***
  - Mac – BlueSee, Light Blue – ***good***
  - Linux – don’t know

References:

https://en.wikipedia.org/wiki/List_of_Bluetooth_protocols#Radio_frequency_communication_(RFCOMM)

https://learn.adafruit.com/circuitpython-ble-libraries-on-any-computer?view=all

https://circuitpython.readthedocs.io/projects/ble/en/latest/examples.html

https://github.com/adafruit/Adafruit_CircuitPython_BLE