Writing a BLE application is a snap with Apache Mynewt*
(* incubating at ASF)

Aditi Hilbert

ApacheloT @ ApacheCon, 2017
An Open Source OS for MCUs

- Tested, open source networking stacks
- Pre-emptive power optimized RTOS
- Secure Bootloader and Image Upgrade
- Encrypted flash filesystem
- Build & Package Management
- Management Interfaces

<table>
<thead>
<tr>
<th>Security</th>
<th>Networking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stats &amp; Logging</td>
<td>Console &amp; Shell</td>
</tr>
<tr>
<td>Drivers</td>
<td>Config &amp; Upgrade</td>
</tr>
<tr>
<td>Power</td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>HAL</td>
</tr>
<tr>
<td>Secure Bootloader &amp; FFS</td>
<td></td>
</tr>
</tbody>
</table>
Why An Open-Source Bluetooth Stack?

**Debug Your Bluetooth**
- Proprietary stacks often come without source code access
- Mysterious CPU disappearances go unexplained

**More Flexible Architecture**
- Ability to operate in both host-only and controller-only modes
- Removes lock-in to a single vendor due to stack
- Allows you to make feature/function trade-offs

**Better Performance**
- Ability to tune memory and code footprint for application requirements
- Unified buffering scheme eliminates copies
Newt - Build and Package Management Tool

Manage Large Code Bases
- Break your system into smaller, testable components
- Understand your code-base with built-in tools

Control Debug and Production
- Define targets that control build options
- Manage debugging for individual boards
- Generate manufacturing images and upgradable software images

Enable Collaboration
- Version and release components independently
- Develop cross-RTOS reusable tools (bluetooth stack, USB stack, Sensor APIs)
Projects are collections of Packages

Packages are individual components that define software: think libraries

Projects can be released and versioned, in which case they are known as Repositories

Repositories can be relied on by multiple projects
Newt Basics - Build

- Targets define build
- Build is a combination of app and bsp
- **Target**, app and bsp are packages, and packages define dependencies and build settings
- Packages can take different decisions based upon system configuration
Newt Basics – System Configuration

• Packages can create system configuration settings in syscfg.yml (syscfg.defs)

• System configuration settings can be overridden by specifying syscfg.vals

• Packages can change build settings and dependencies based upon syscfg definitions

• Source code files can conditionally compile functionality using #if MYNEWT_VAL(SETTING)

• The newt tool automatically detects conflicts, and displays settings
Newt Basics – Debug and Load

- BSPs define debug scripts: debug, download
- Script name is `<bsp>_debug`, `<bsp>_download`
- These scripts allow newt debug, and newt load commands to operate
- Newt debug and load automatically handle starting Jlink, openocd and providing GDB remotes
Pushing Boundaries – App Success Stories

• Beacons

• Connections galore!

• Multiple advertisers in a single device on BLE 4.2

• Connecting to multiple centrals

• Guaranteed connection events from multiple peripherals with guaranteed throughput

Demo Time!
Nimble Configuration

Sample system configuration (syscfg.yml)

```
BLE_ROLE_CENTRAL:
  description: 'Enables Central role'
  value: 1

BLE_ROLE_PERIPHERAL:
  description: 'Enables Peripheral role'
  value: 1

BLE_ROLE_BROADCASTER:
  description: 'Enables Broadcaster role'
  value: 1

BLE_ROLE_OBSERVER:
  description: 'Enables Observer role'
  value: 1

BLE_MAX_CONNECTIONS:
  description: 'Maximum # of concurrent connections
  value: 16
```

Configuration items – simple to complex

- Enable advertising of Eddystone beacons
- Enable support for multi-advertisers
  ...
- The number of slots that will be allocated to each connection
- Maximum size of the PDU's in a connection

Nimble Supports All Roles (Broadcaster, Observer, Central, Peripheral)
- Multiple roles supported simultaneously (can be both Central and Peripheral)
- Supports up to 32 concurrent connections
Managing BLE devices and app

- **Newtmgr**
  - Application protocol to define connection profiles to remotely query, configure, manage devices
  - In golang
  - In JS with Node for newtmgr

- **OIC 1.1**
  - Constrained web server on device
    - Protocols and services to host resources and attributes
  - Automatic discovery and functional interactions
  - CoAP messaging, CBOR encoding
  - Mynewt optimizations and additional transport
    - BLE, serial
Newtmgr - Image Upgrade over BLE

- https://nnewtmgr.surge.sh
  - Contributor: Jacob Rosenthal

- Overview
  - Slot 0 is primary image, device always boots up from here
  - Slot 1 holds upgrade image
  - Test – success sets it to be swapped into slot 0
  - Reset – bootloader swaps images using scratch, brings up device from new image in slot 0
  - Confirm – makes new image in slot 0 permanent

Demo Time!
OIC 1.1 Basics

- RESTful architecture with client and server
- Resources identified by URI
- Built-in discovery and subscription mechanisms
- Resources consist of key/value pairs
- CRUDN: Create/Retrieve/Update/Delete/Notify
- COAP, HTTP & others

```
/light/1
{
  rt: “oic.r.light”
  if: [ “oic.if.baseline”, “oic.if.rw” ]
  state: “on”
}
```
Sensor Framework in Mynewt

- Resource discovery – unicast or multicast
- Real-time processing
- Time-series support
- Simple APIs
- Wide array of sensor types

Enables collaboration with several Apache data collection and processing projects!
Sensor Drivers and API

• Data structure for sensor data
  – E.g. X, Y, Z data for accelerometer data

• Sensor manager
  – initialize multiple sensors, listener registration, read data, etc.

• Device driver
  – Configure, calibrate, last read time for specific sensors

• Sensor configuration using sysefconf.yml
  – Enable/disable logging, stats, shell console, I2C interface etc. for specific app

• Initialize sensor in the board bsp

Demo Time!
## Code Optimized Size for Your Stack

<table>
<thead>
<tr>
<th>Feature</th>
<th>Code</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure Bootloader</td>
<td>12KB</td>
<td></td>
</tr>
<tr>
<td>Core RTOS</td>
<td>7KB</td>
<td>1KB</td>
</tr>
<tr>
<td>Bluetooth Controller - Base</td>
<td>20KB</td>
<td>10KB (1 conn)</td>
</tr>
<tr>
<td>Bluetooth Host - Base</td>
<td>28KB</td>
<td></td>
</tr>
<tr>
<td>Bluetooth Controller - Complete</td>
<td>26KB</td>
<td>18KB (1 conn)</td>
</tr>
<tr>
<td>Bluetooth Host - Complete</td>
<td>68KB</td>
<td></td>
</tr>
<tr>
<td>Bluetooth Stack <em>per-connection</em></td>
<td></td>
<td>652 bytes</td>
</tr>
<tr>
<td><em>OIC - Optimized for Mynewt</em></td>
<td>15KB</td>
<td>1KB</td>
</tr>
<tr>
<td><em>OIC – Prior to Optimization</em></td>
<td>23KB</td>
<td>12KB</td>
</tr>
<tr>
<td>Secure Software Upgrade</td>
<td>5KB</td>
<td>400 bytes</td>
</tr>
</tbody>
</table>
References

Host

- http://runtime.io
- https://github.com/apache/incubator-mynewt-core
- https://mynewt.apache.org
- Demo: https://www.youtube.com/watch?v=LCmwTWLcVBs
- OIC 1.1 Specs: https://openconnectivity.org/resources/specifications