An Incubator Project in the Apache Software Foundation

http://mynewt.apache.org/

runtime.io

13 July 2016

Apache Mynewt

Open Source OS for Constrained IoT

- MCU / Hardware independent
- ARM Cortex-M*, AVR, MIPS, more...

- RISC-V

http://mynewt.apache.org/

```c
#include <os/os.h>

void
task1_handler(void *arg)
{
    struct os_task *t;

    /* Set the led pin for the E407 devboard */
    g_led_pin = LED_BLINK_PIN;
    gpio_init_out(g_led_pin, 1);

    while (1) {
        t = os_sched_get_current_task();
        assert(t->t_func == task1_handler);

        ++g_task1_loops;

        /* Wait one second */
        os_time_delay(1000);

        /* Toggle the LED */
        gpio_toggle(g_led_pin);
    }
}
```

Apache mynewt

Problem and Context

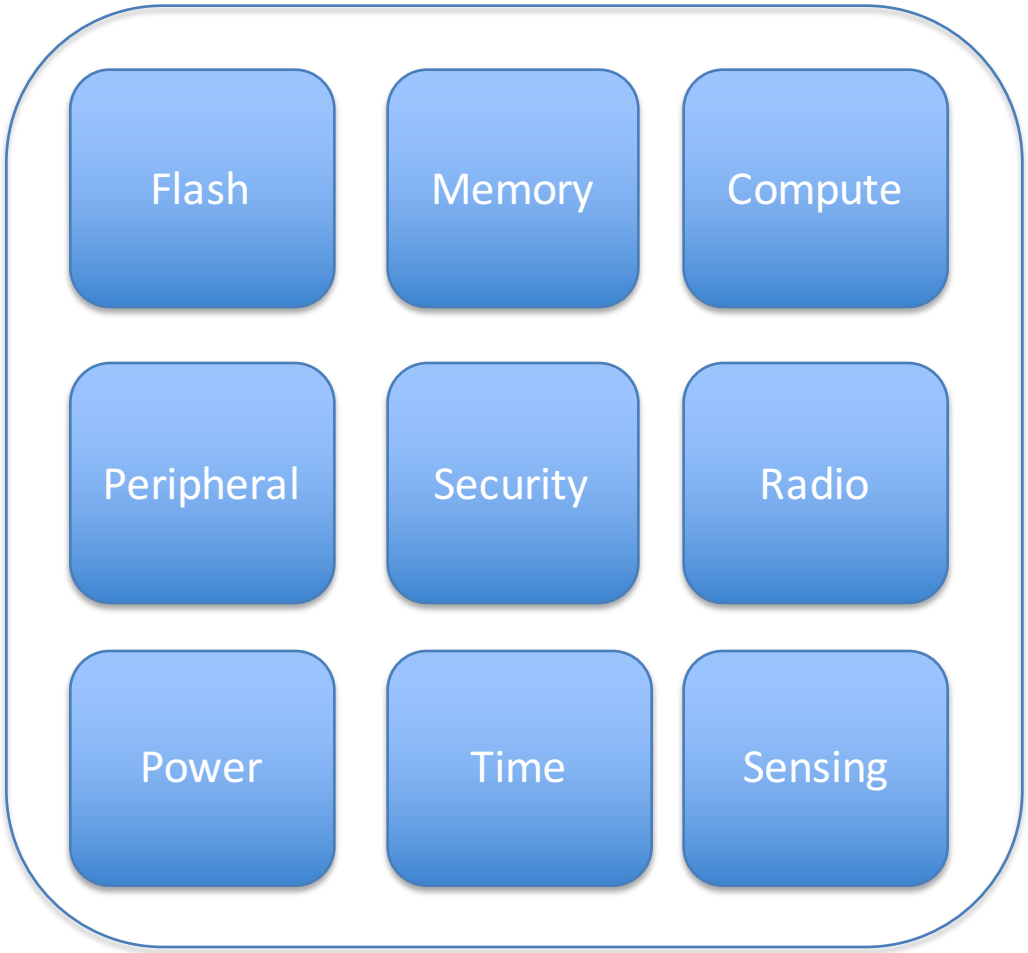
First release of a successful IoT product…

…now make that repeatable, please.

Apache mynewt

IoT is being driven by the System on Chip (SoC)

Apache Mynewt addresses everything needed for SoC development



Characteristics

- CPU: 48MHz-300MHz, Cortex-M
- Radios: BLE, Wi-Fi, 802.15.4(g)
- Flash/RAM: 512KB/64KB (today), 1MB/256KB (this year)
- Size: 3-12mm

Benefits

- Inexpensive
- Low Power
- Easy to manufacture

Apache Mynewt is “Linux” for devices that cannot run Linux

Runtime with Mynewt Modernizes Development for SoCs

RISC-V v1.0
13 July 2016

Without Runtime

- Choose Compiler
- Buy RTOS
- Build Bootloader
- Write Flash Drivers
- Develop Logging
- Support Chip
- Buy Net Stack
- MFRG Support
- Write Gateway
- Develop SW Update

With Runtime

- Use Apache Mynewt
- Connect to Runtime Management Service
- Provision AWS environment
- Develop Application

Development: Faster, Consistent, Repeatable. Escape HW lock-in.

Another View: Initial Bluetooth Low Energy System

Cloud

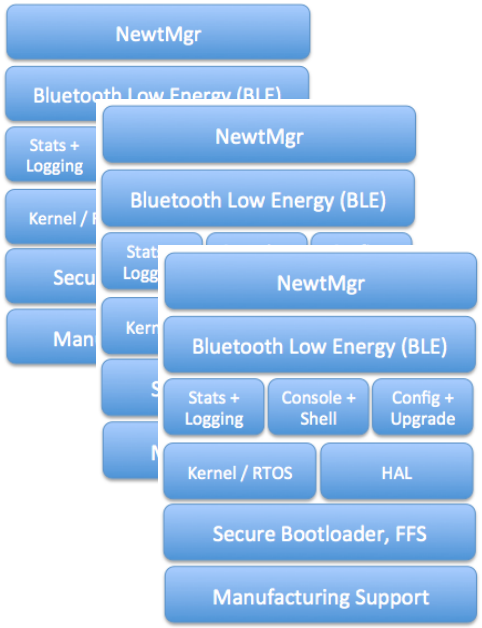
- Device Provisioning
- Software Upgrade
- Failure Analysis

Gateway

- NewtMgr

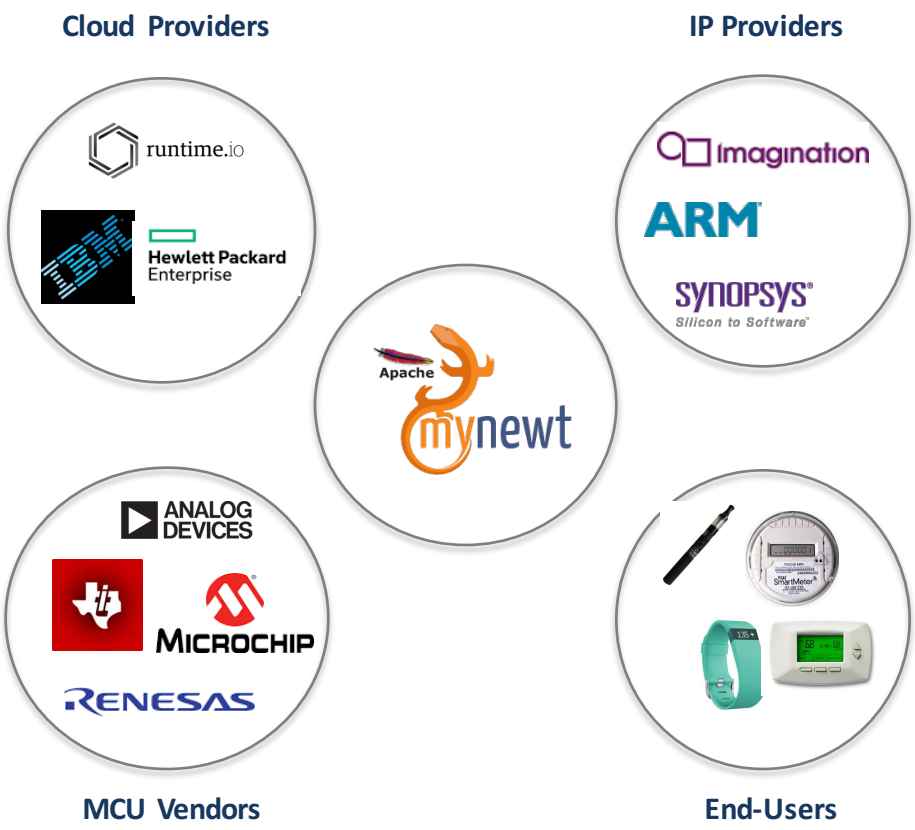
- Discovery
- Registration
- Provisioning
- Stats & Logging
- Configuration Management
- Core Management
- Image Upgrade

OS



Only end-to-end platform built atop FOSS; usage-based cloud

A Community Effort



Why the ASF?

- Liberal Apache 2.0 license
- Individuals, not Corporations
- Meritocracy
- Strong licensing and IP policies
- Long history of working with large organizations: IBM, Oracle, Pivotal/EMC
- Many years experience managing large, complex projects: Apache Web Server, Hadoop, Cassandra, Kafka, Subversion, etc.

Community-driven Open Source: Best Way to Maintain Healthy User Ecosystem

Apache mynewt

Apache Mynewt Users

(today)

- Bluetooth connected products
  - Medical: everything
  - Consumer/Enterprise: locks, lights
  - Industrial
- Makers
  - Home
  - Hardware Labs
  - Clothing

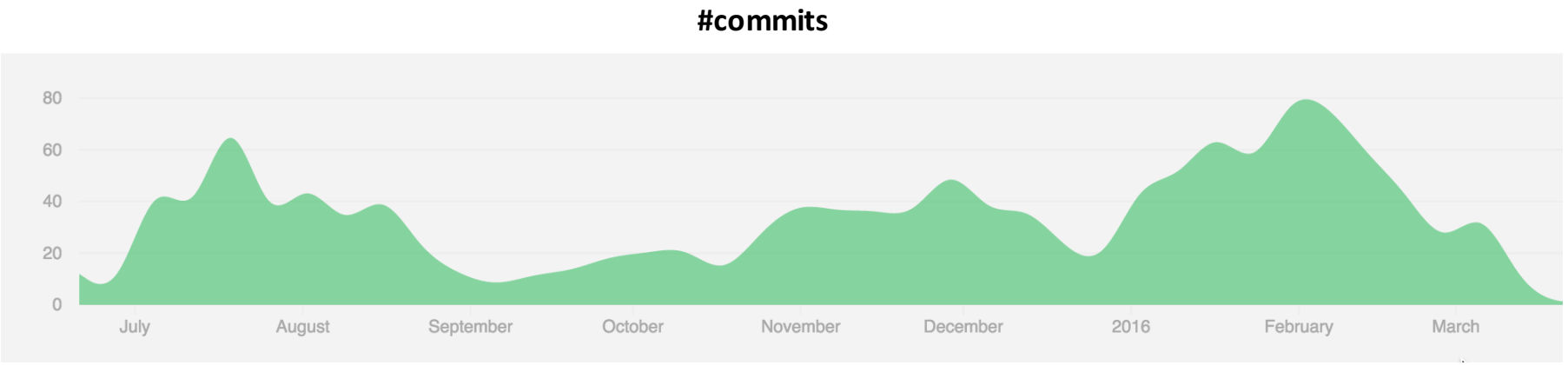
(tomorrow)

- Bluetooth Low Energy 4.2 → Bluetooth 5
- Industrial Wireless Sensor Networks
- Wi-Fi Products
- Who Knows?

- Power Optimization
- Mesh networking
- Security
- Sensor Algorithms
- Control Systems

Scale Makes Problems Interesting

Project Statistics



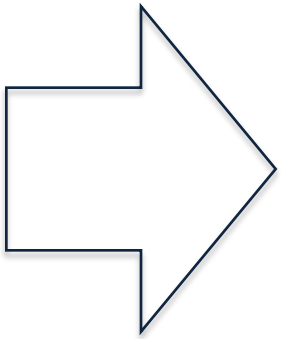
- 280,000 lines of code (March 2016)
- Initial support for Simulator, Nordic NRF51/52, STM32F3/4 and Arduino
  - Announced: Arduino Primo, Arduino Otto
- Active contributors, We Want You!
  - ✓ http://mynewt.apache.org/

Apache mynewt

Build and Package Management

Goals

- Maintain and re-use packages across multiple products
- Manage debug and production build setups
- Make it easy to find, install 3rd party libraries
- Efficiency: use only what you need



Description

- Everything is a package. Each package describes its dependencies to the rest of the world
- A collection of packages is called an application
- There are a few special packages: BSP and Project. Project contains main() and BSP defines linker script and hardware layout
- Targets are used to combine projects and BSP
- Packages can be distributed, upgraded and installed remotely

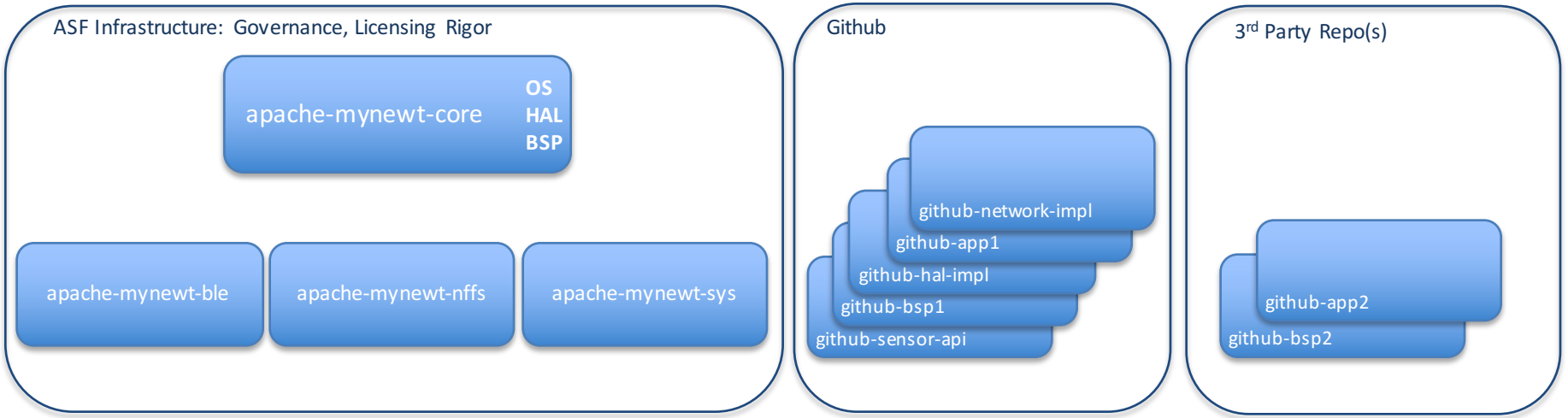
Compose across multiple repos

Apache mynewt

Project Structure (coming soon)

"core" broken into appropriate sub-projects

RISC-V v1.0
13 July 2016



- ASF governance structure (PMCs) corresponds with sub-package structure
- ASF repositories clean, Apache 2.0 license
- Users can assemble projects sourced from multiple repos

Composability across multiple repos provides flexibility

Apache mynewt

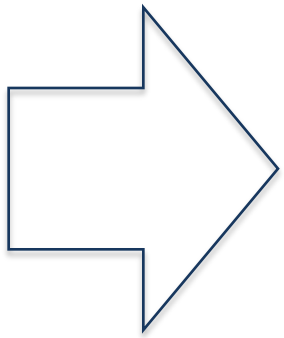
BSP and HAL

Goals

Provide quality drivers for major MCU platforms

Design for cross-platform: well-defined APIs for HAL, BSP and drivers

Make it easy to add board specific definitions



Description

- BSP definition is provided in <app>/hw/bsp
- HAL definitions are in hw/hal and contain uniform, cross-platform APIs
- MCU definitions are in hw/mcu and provide implementations for various MCUs
  - Hierarchy allows code-reuse within MCU families
- BSPs for common dev kits are available as packages (e.g., Nordic nRF51/2 DK)
- BSPs depend on MCUs
- BSP + MCU provides implementation for HAL APIs

Chip Vendors: We Want You!

Apache mynewt

Kernel

- Pre-emptive, multi-tasking RTOS
  - Strict, priority-based scheduling
  - Up to 253 different priority levels
- Tickless kernel
- Power management
- Resource utilization tracking
- Built-in tasks:
  - Idle
  - Sanity

```c
#include <os/os.h>

#include <assert.h>

/* Task 1 */
#define TASK1_PRIO (1)
#define TASK1_STACK_SIZE    OS_STACK_ALIGN(1024)
struct os_task task1;
os_stack_t stack1[TASK1_STACK_SIZE];
static volatile int g_task1_loops;

void
task1_handler(void *arg)
{
    while (1) {
        ++g_task1_loops;

        /* Wait one second */
        os_time_delay(1000);
    }
}

int
main(int argc, char **argv)
{
    int rc;

    os_init();

    os_task_init(&task1, "task1", task1_handler, NULL,
            TASK1_PRIO, OS_WAIT_FOREVER, stack1, TASK1_STACK_SIZE);

    os_start();

    assert(0);

    return (0);
}
```

Apache mynewt

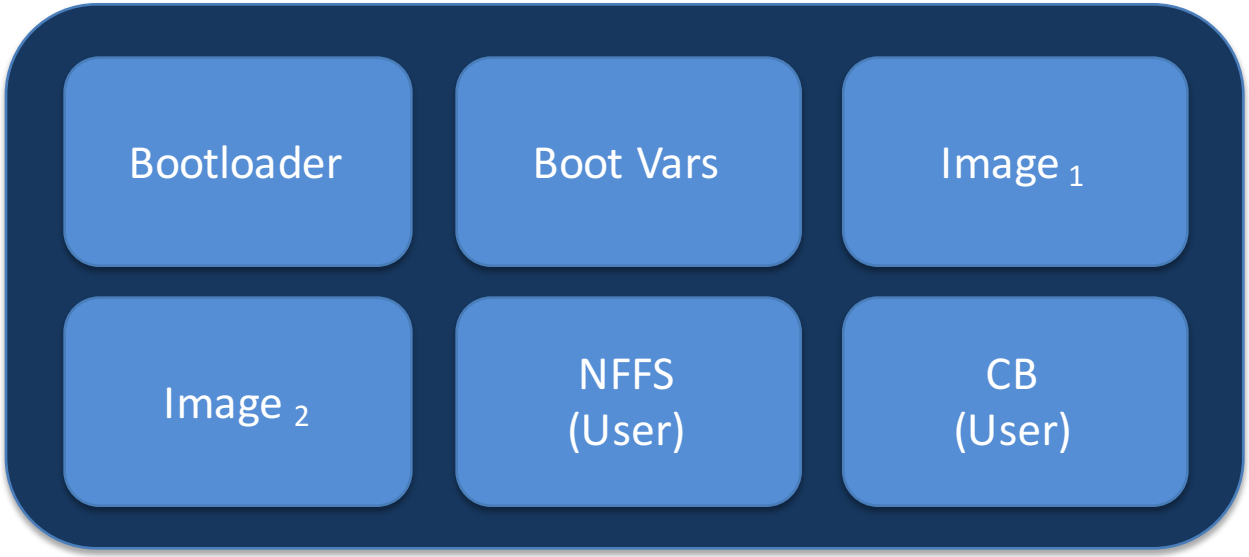
Energy Efficient Event Model

- Event Queues provide a mechanism for “mostly sleeping” asynchronous tasks
- Wake-up on:
  - Message from another task
  - Timer
  - I/O state change
  - Incoming packet
  - Watchdog
- Perform operations:
  - Send an alert
  - Respond to a request
  - Schedule a wakeup
- Go back to sleep

```
struct os_eventq task1_evq;
struct os_eventq task2_evq;

#define OS_EVENT_T_PING (OS_EVENT_PERUSER)
#define OS_EVENT_T_PONG (OS_EVENT_PERUSER + 1)

void
task1_handler(void *arg)
{
    struct os_event *ev;
    struct os_event ping_ev;

    ping_ev.ev_type = OS_EVENT_T_PING;
    ping_ev.ev_arg = NULL;

    os_eventq_put(&task2_evq, &ping_ev);

    while (1) {
        ev = os_eventq_get(&task1_evq);
        assert(ev->ev_type == OS_EVENT_T_PONG);

        os_eventq_put(&task2_evq, &ping_ev);

        ++g_task1_loops;

        /* Wait one second */
        os_time_delay(1000);
    }
}

void
task2_handler(void *arg)
{
    struct os_event *ev;
    struct os_event pong_ev;

    pong_ev.ev_type = OS_EVENT_T_PONG;
    pong_ev.ev_arg = NULL;

    while (1) {
        ev = os_eventq_get(&task2_evq);
        assert(ev->ev_type == OS_EVENT_T_PING);

        os_eventq_put(&task1_evq, &pong_ev);

        ++g_task2_loops;
    }
}
```

Apache mynewt

Energy Efficient Event Model



- Bootloader can be located in ROM or Flash
  - Options for internal and external flashes
- Performs integrity check and swaps images
- Images contain SHA-256 hash and RSA signature
- NFFS optional
  - Provides a log-structured flash filesystem designed for small flashes
- CB (Circular Buffer) optional
  - Provides implementation of flash circular buffer

System Security

What We're Protecting	How We Protect It
Bootloader	• First stage bootloader can operate from ROM and verify signature of second stage bootloader
Images	• All images have SHA-256 of image contents • Images support signing with ECC or RSA 2048 bit signatures • Second stage bootloader can verify image signature
Network Interfaces	• Full support for BLE 4.2 security at 1.0 release, including link-layer and app-layer

Simulator and Test Framework

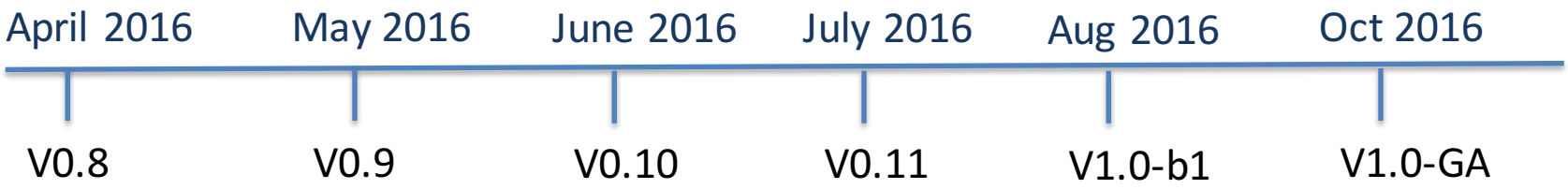
- In addition to various MCU ports: OS, HAL, FS and the majority of packages can run on Mac, Linux
- Develop your code on the host and then port to the real hardware
- Unit test framework is incorporated to all of the packages: ability to run unit tests on simulated environment and real hardware
- OS and libraries are fully regression tested to ensure API compatibility between releases

```
/* Test some error cases */
TEST_ASSERT(os_mutex_init(NULL)      == OS_INVALID_PARM);
TEST_ASSERT(os_mutex_delete(NULL)    == OS_INVALID_PARM);
TEST_ASSERT(os_mutex_release(NULL)   == OS_INVALID_PARM);
TEST_ASSERT(os_mutex_pend(NULL, 0)   == OS_INVALID_PARM);

/* Get the mutex */
err = os_mutex_pend(mu, 0);
TEST_ASSERT(err == 0,
            "Did not get free mutex immediately (err=%d)", err);

/* Check mutex internals */
TEST_ASSERT(mu->mu_owner == t && mu->mu_level == 1 &&
            mu->mu_prio == t->t_prio && SLIST_EMPTY(&mu->mu_head),
            "Mutex internals not correct after getting mutex\n"
            "Mutex: owner=%p prio=%u level=%u head=%p\n"
            "Task: task=%p prio=%u",
            mu->mu_owner, mu->mu_prio, mu->mu_level,
            SLIST_FIRST(&mu->mu_head),
            t, t->t_prio);

/* Get the mutex again; should be level 2 */
err = os_mutex_pend(mu, 0);
TEST_ASSERT(err == 0, "Did not get my mutex immediately (err=%d)", err);
```

Apache mynewt

Apache Mynewt Roadmap

RISC-V v1.0
13 July 2016



Highlights

v0.8: First release, BLE 4.2, FFS, Kernel, Console, Shell, Secure Boot
v0.9: Expanded HW support and HAL
v0.10: Wi-Fi & IP support
v0.11: Full Bluetooth Support / Qualification
v1.0 (GA) API compatibility, Full Regression Testing

THANK YOU

http://mynewt.apache.org/

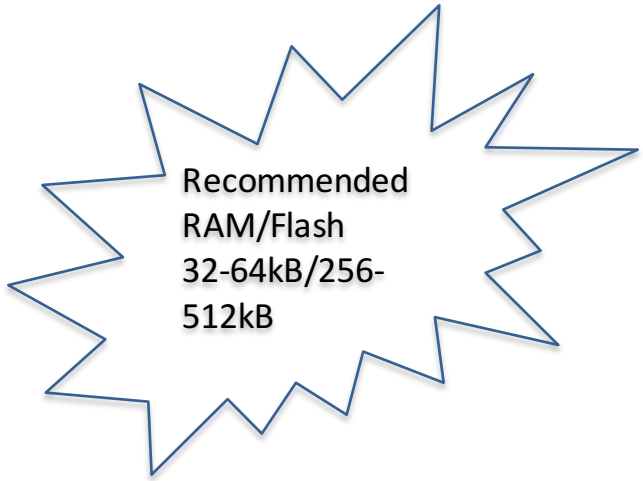
Mailing List: dev@mynewt.incubator.apache.org

IRC: #mynewt on freenode

Image Size

Mynewt Components/Composition	Min RAM	Min ROM/On-chip Flash
Core OS kernel	<1kB	<6kB
Blinky application (incl. GPIO, HAL, console, shell)	17kB	23kB
Bootloader (incl. Newtron Flash File System)	4.5kB	25kB
NimBLE stack (incl. both peripheral and central roles, legacy pairing)	4.5kB	69kB
BLE example application “bleprph” (incl. OS, radio, NimBLE)	15kB	99kB

```
$ newt target show primoblinky
targets/primoblinky
    app=@apache-mynewt-core/apps/blinky
    bsp=@apache-mynewt-core/hw/bsp/arduino_primo_nrf52
    build_profile=optimized
$ newt size primoblinky
  FLASH      RAM
     58      242 *fill*
    600      500 arduino_primo_nrf52.a
   3148       32 baselibc.a
    461    13848 blinky.a
     64        0 cmsis-core.a
   1236       37 config.a
    124        0 crt0.o
      8        0 crti.o
     16        0 crtn.o
    937      196 full.a
     20        8 hal.a
   2759        0 json.a
     96     1068 libg.a
   1508        0 libgcc.a
    154      240 newtmgr.a
   1395       40 nrf52xxx.a
   5029      813 os.a
   2025      310 shell.a
   1155       66 stats.a
   2251        0 util.a

objsize
   text    data     bss     dec     hex filename
  23012    1356   15644   40012    9c4c ~/dev/arduinotests/bin/primoblinky/apps/blinky/blinky.elf
$
```



RAM Requirements for BLE Applications

RISC-V v1.0
13 July 2016

BLE Component	BLE Configuration Element	Default (kB)	Element Size (kB)	Default Size (kB)	Size for x # of connections
Host (.bss+ .data)		3828			
Host (runtime RAM reqs)	HCI buffer	3	64	204	Independent
	Max # of concurrent connections	1	80	80	x*80
	Max # of services	5	8	40	Independent
	Max # of config descriptors (peripheral)	1	4	4	(x+1)*3
	Max # of concurrent GATT procedures	2	40	80	Independent
	Max. # of total ATT attributes	36	32	1152	Independent
	max_prep_entries (for partial writes)	6	12	72	Independent
	Max # of L2CAP channels (3 per connection)	3	28	84	x*3
	Max # concurrent L2CAP signalling procedures	2	20	40	Independent
	Max # concurrent Security Manager procedures	1	360	360	Independent
Controller (BSS + data)	Max # of concurrent connections	1	416	416	x*416
	# of duplicate scan advertisers	8	8	64	Independent
	# of scan response advertisers	8	8	64	Independent
	Whitelist size	8	8	64	Independent
	Resolvable private address list	4	40	160	Independent
	RNG buffer size	32	1	32	Independent

Runtime RAM requirement for NimBLE with defaults (including security): 6744 kB

Power Management