

Tock Embedded OS Training

RustConf 2017

Tock is a...

1. Secure
2. Embedded
3. Operating System
4. for Low Resource
5. Microcontrollers

Secure



~~Secure~~ Safe

Tock has isolation primitives that allow you to build secure systems.

Embedded

Definition A

Operating system, applications and hardware are tightly integrated.

Definition B

You'll likely be writing the kernel.

Operating System

Tock itself provides services to components in the system:

- ▶ Scheduling
- ▶ Communication
- ▶ Hardware multiplexing

Low Resource

- ▶ 10s μ A average power draw
- ▶ 10s of kB of RAM
- ▶ Moderate clock speeds

Microcontrollers

System-on-a-chip with integrated flash, SRAM, CPU and a bunch of hardware controllers.

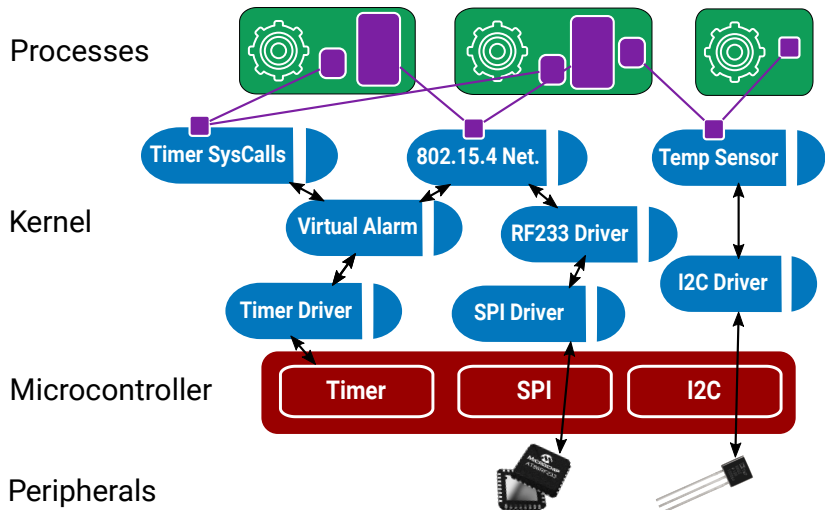
Typically:

- ▶ Communication: UART, SPI, I2C, USB, CAN...
- ▶ External I/O: GPIO, external interrupt, ADC, DAC
- ▶ Timers: RTC, countdown timers

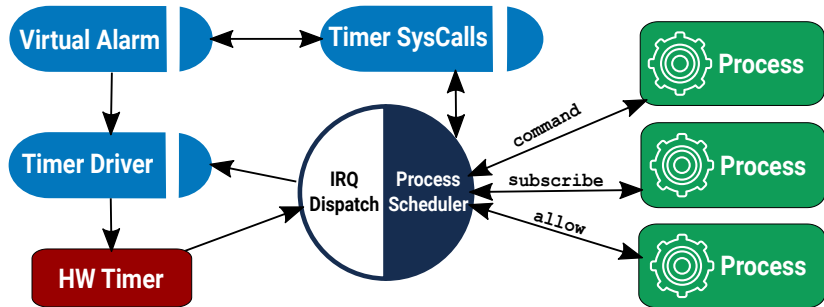
Maybe...

- ▶ Radio (Bluetooth, 15.4)
- ▶ Cryptographic accelerators
- ▶ Other specialized hardware...

Two types of components: capsules and processes



Two types of scheduling: cooperative and preemptive



Part 1: Hardware, tools and development environment

Hail

SAM4L

Accelerometer

MCU

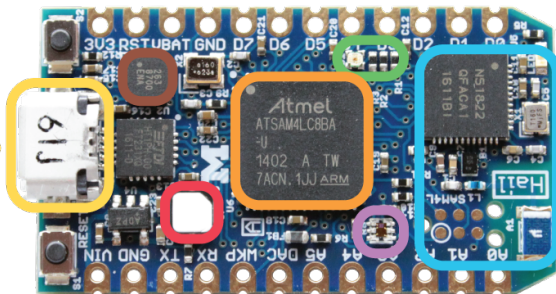
RGB LED

USB

BLE

Temperature &
Humidity

Light Sensor



Binaries on-board

- ▶ Bootloader
- ▶ Kernel
- ▶ Processes

Tools

- ▶ `make`
- ▶ Rust nightly (`asm!`, compiling core)
- ▶ `xargo` to automate compiling base libraries
- ▶ `arm-none-eabi GCC` to link binaries
- ▶ `tockloader` to interact with Hail and the bootloader

Tools: tockloader

Write a binary to a particular address in flash

```
$ tockloader flash --address 0x1000 \  
    target/thumbv7em-none-eabi/release/hail.bin
```

Program a process in Tock Binary Format¹:

```
$ tockloader install myapp.tab
```

Restart the board and connect to the debug console:

```
$ tockloader listen
```

¹TBFs are relocatable process binaries prefixed with headers like the package name. .tab is a tarball of TBFs for different architectures as well as a metadata file for tockloader.

Check your understanding

1. What kinds of binaries exist on a Tock board? Hint: There are three, and only two can be programmed using `tockloader`.
2. What are the differences between capsules and processes? What performance and memory overhead does each entail? Why would you choose to write something as a process instead of a capsule and vice versa?
3. Clearly, the kernel should never enter an infinite loop. But is it acceptable for a process to spin? What about a capsule?

Hands-on: Set-up development environment

1. Compile and flash the kernel
2. Compile and program `ble-env-sense` service
3. (Optional) Add some other apps from the repo, like `blink` and `sensors`
4. (Optional) Familiarize yourself with `tockloader` commands
 - ▶ `uninstall`
 - ▶ `list`
 - ▶ `erase-apps`

Part 2: The kernel

Hands-on: Write and add a capsule to the
kernel

Part 3: User space