# Tock OS

“

*An embedded operating system designed for running multiple concurrent, mutually distrustful applications on low-memory and low-power microcontrollers.*

## TockOS

- A **pre-emptive** embedded OS (runs on MCUs)
  - Cortex-M
  - RISC-V
- Uses memory protection (**MPU** required)
- Has separate **kernel** and **user space**
  - most embedded OS have the one piece software philosophy
- Runs **untrusted apps** in user space
- Kernel (and drivers) written in **Rust**
- Apps written in C/C++ or Rust (any language that can be compiled)

## Functional Components

### Applications

- user space processes
- any language
- independent executable

`System calls (syscall)`

### Capsules

- hardware independent drivers
- inside the kernel
- safe rust

`Hardware Interface Layer (HIL)`

### HAL

- hardware dependent driver
- unsafe rust

# Architecture

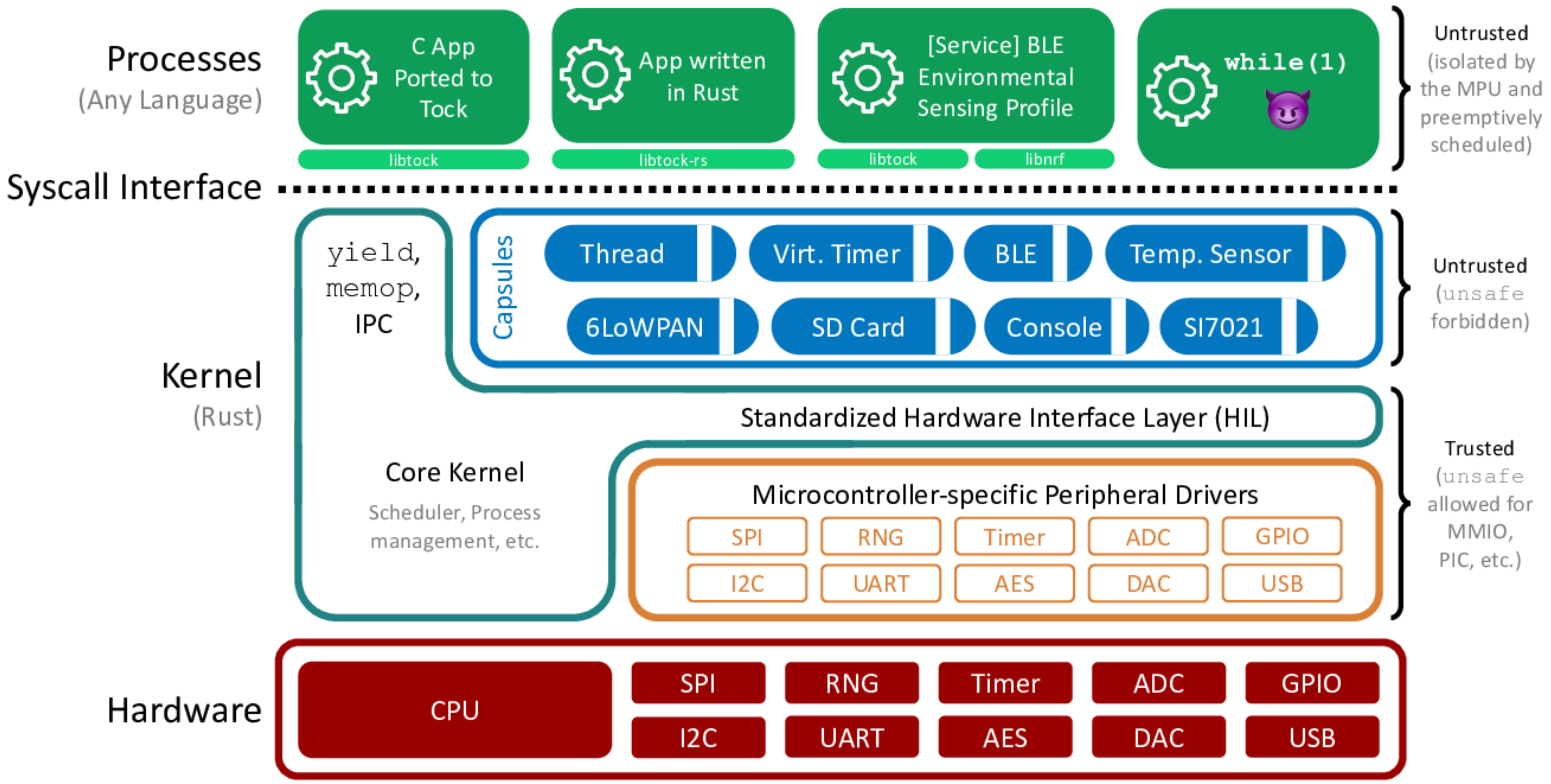


Image from https://github.com/tock/tock/blob/master/doc/Overview.md

## Kernel

- Non pre-emptive
  - capsules (drivers) run to completion
  - async API
- Does not allocate dynamic memory
  - only static buffers
  - no out of memory errors in kernel
- Grants
  - memory for capsules in user processes
  - allocated at the start of a process (if possible)

## Used by

OpenSK

*Open-source implementation for security keys written in Rust that supports both FIDO U2F and FIDO2 standards.*

Google Inc.

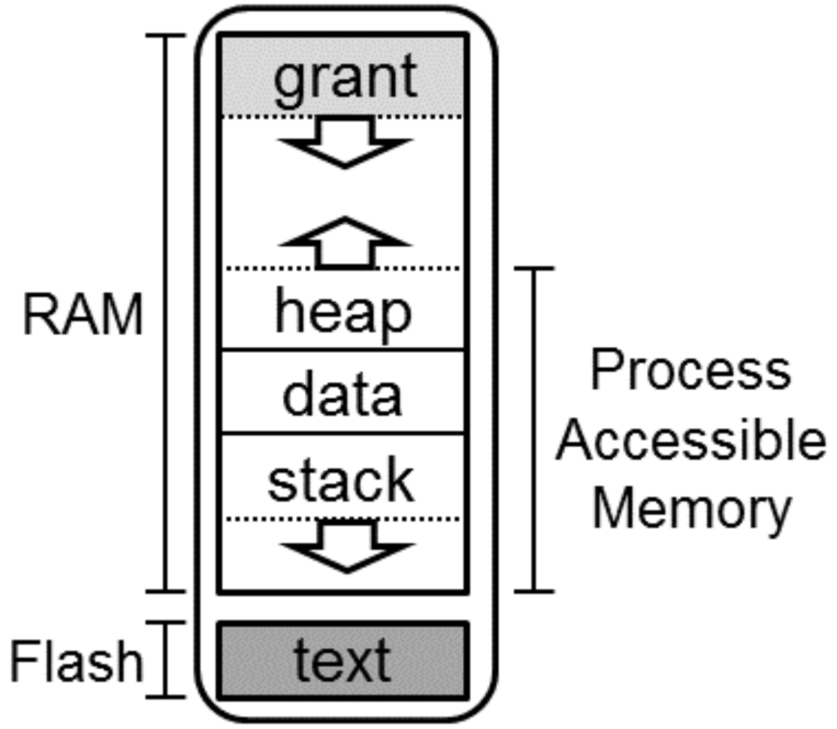
opentitan

*The first open source project building a transparent, high-quality reference design and integration guidelines for silicon root of trust (RoT) chips.*

LowRISC, ETH Zurich, G+D Mobile Security, Google, Nuvoton, Western Digital

# Application (Process)

- Standalone executable
  - *compiled without TockOS kernel*
- Memory Protection
  - *MPU Regions*
- Can (*seg*)fault
- Relocatable code
  - where the compiler allows it
- IPC
  - service discovery



# Memory

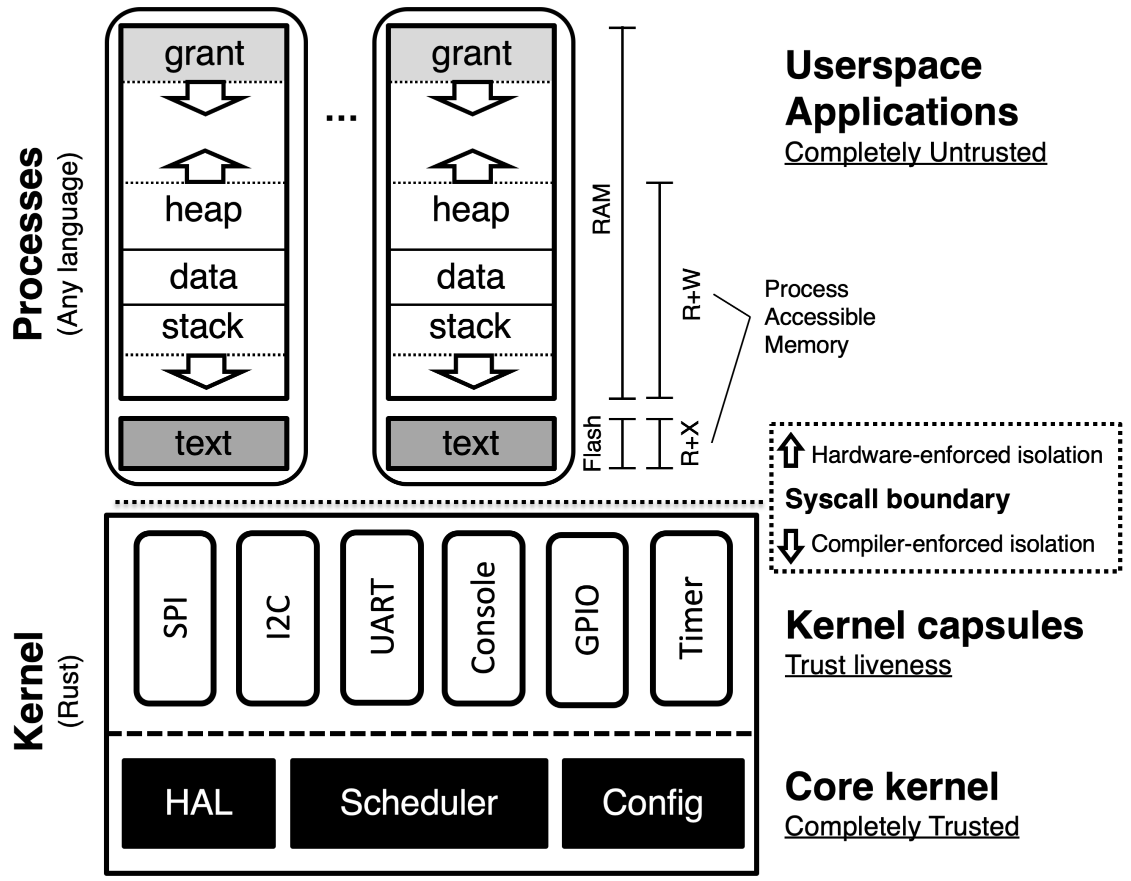
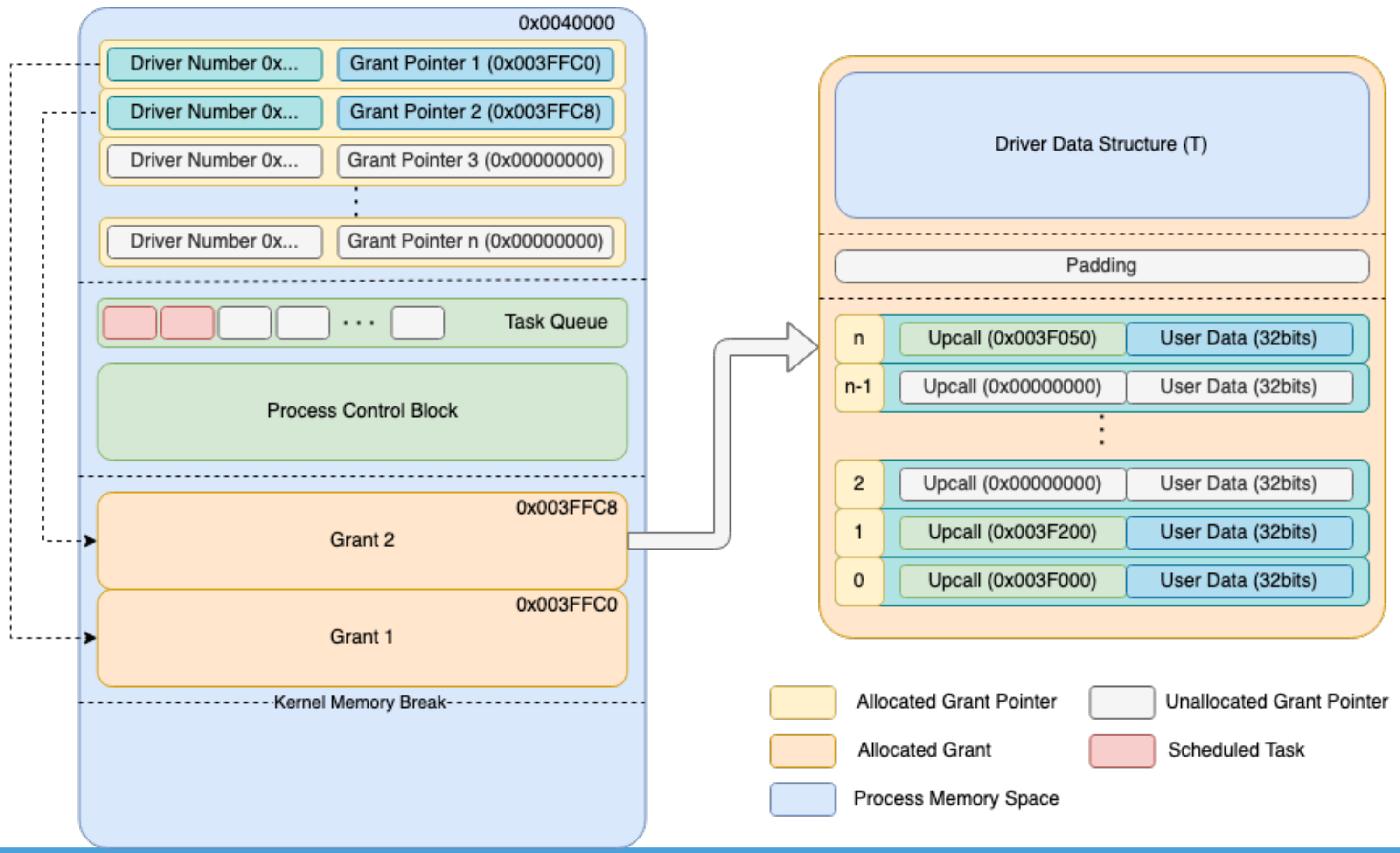


Image from https://github.com/tock/tock/blob/master/doc/Design.md

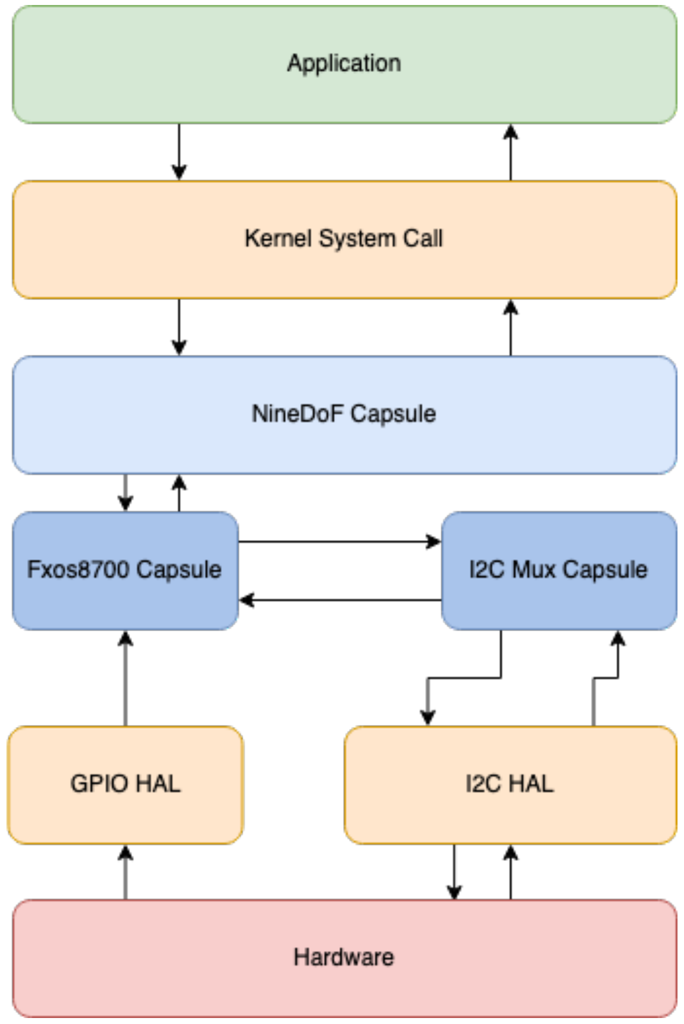
# Grant



0x0040000
Driver Number 0x...
Grant Pointer 1 (0x003FFC0)
Driver Number 0x...
Grant Pointer 2 (0x003FFC8)
Driver Number 0x...
Grant Pointer 3 (0x00000000)
Driver Number 0x...
Grant Pointer n (0x00000000)
Task Queue
Process Control Block
0x003FFC8
Grant 2
0x003FFC0
Grant 1
Kernel Memory Break
Driver Data Structure (T)
Padding
n
Upcall (0x003F050)
User Data (32bits)
n-1
Upcall (0x00000000)
User Data (32bits)
2
Upcall (0x00000000)
User Data (32bits)
1
Upcall (0x003F200)
User Data (32bits)
0
Upcall (0x003F000)
User Data (32bits)
Allocated Grant Pointer
Unallocated Grant Pointer
Allocated Grant
Scheduled Task
Process Memory Space

## Only 7 Syscalls

- 0 – Yield
- 1 – Subscribe
- 2 – Command
- 3,4 – Allow
- 5 – Memop
- 6 – Exit



## Yield

- Processes have a callback queue
  - similar with an event loop system
- Suspends the process until a callback is available
- Callbacks are called only when the process is yielded
  - when *main* returns, libtock-c runs **`while (true) yield();`**

## Subscribe

- The process registers a callback function

- Parameters
  - *capsule_number* – id of the driver
  - *subscribe_number* – a sub_command number, specific to the driver
  - *callback* – pointer to a function or NULL
  - *user_data* – any pointer

## Command

- The process sends a command to a driver
    - similar with *ioctl* from Linux

- Parameters
    - *capsule_number* – id of the driver
    - *command_number* – a command_command number, specific to the driver
    - *data1* – usize parameter
    - *data2* – usize parameter

## ReadWrite/ReadOnly Allow

- The process shares a buffer with a driver

- Parameters
  - *capsule_number* – id of the driver
  - *allow_number* – an allow_command number, specific to the driver
  - *pointer* – pointer to the buffer data
  - *size* – size of the buffer data

## Memop

- The process requests a memory action
  - similar with *brk* and *sbrk* from Linux

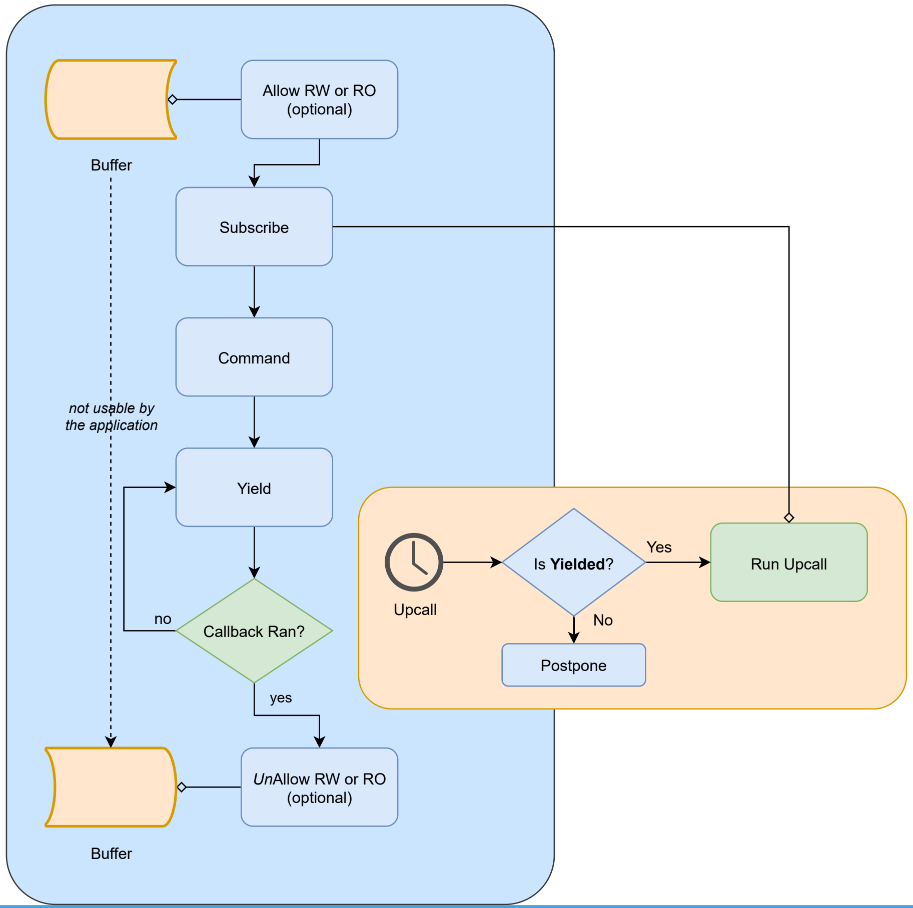
- Parameters
  - *op_type* – action id
  - *argument* – usize parameter

## Exit

- Stop or restart the process

- Parameters
  - *restart* – action id
  - *completion_code* – usize

# Full system call



## Userland libraries

**LibTock-C**

- Stable (recommended)
- newlib
  - libc
  - libm
- libc+
  - *Lua53*
  - *LittlevGL*
- RV32: issue with code relocation

**LibTock-RS**

- not stable yet
- core
  - active development
- issue with relocation
  - Compiler problem

## Tock Executable

- Tock Binary Format
    - TBF
    - Tock header with memory and loading requirements
    - Process binary
- Tock Application Bundle
    - TAB
    - several TBF files for several architectures
    - ARM M0, M3, M4, RV32-IMAC and RV32-IMC

## Tockloader

- Manage Tock OS Application
- Uses TAB files
- Written in Python
- Needs implementation for several boards
- *Small App Store*

## Future Plans

- In Progress
  - Ethernet for STM32
  - WiFi for Raspberry Pi Pico W
  - Tockloader Rust version

Tock

**www.tockos.org**

Thank you