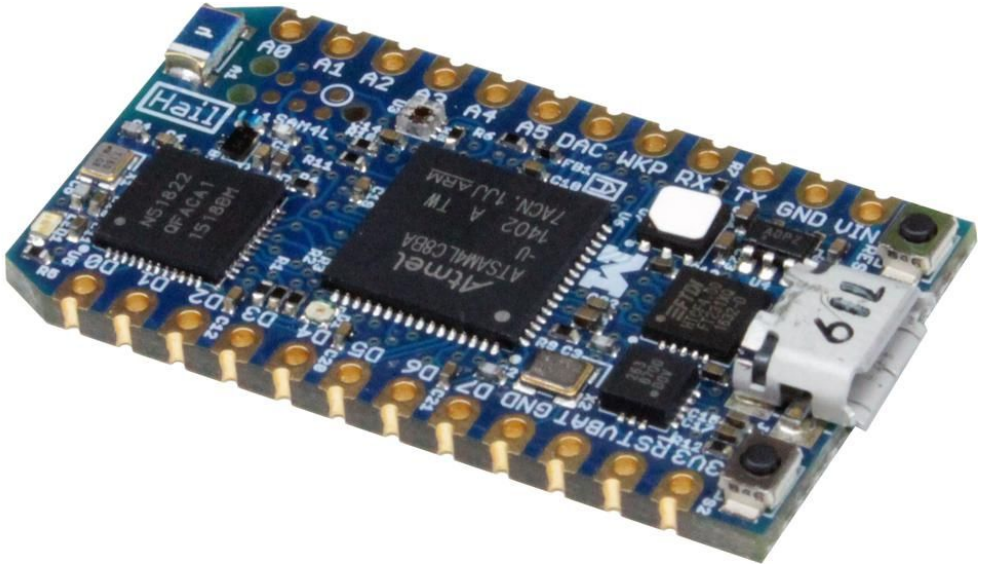
Hail / TockOS

von Daniel & Fabrice

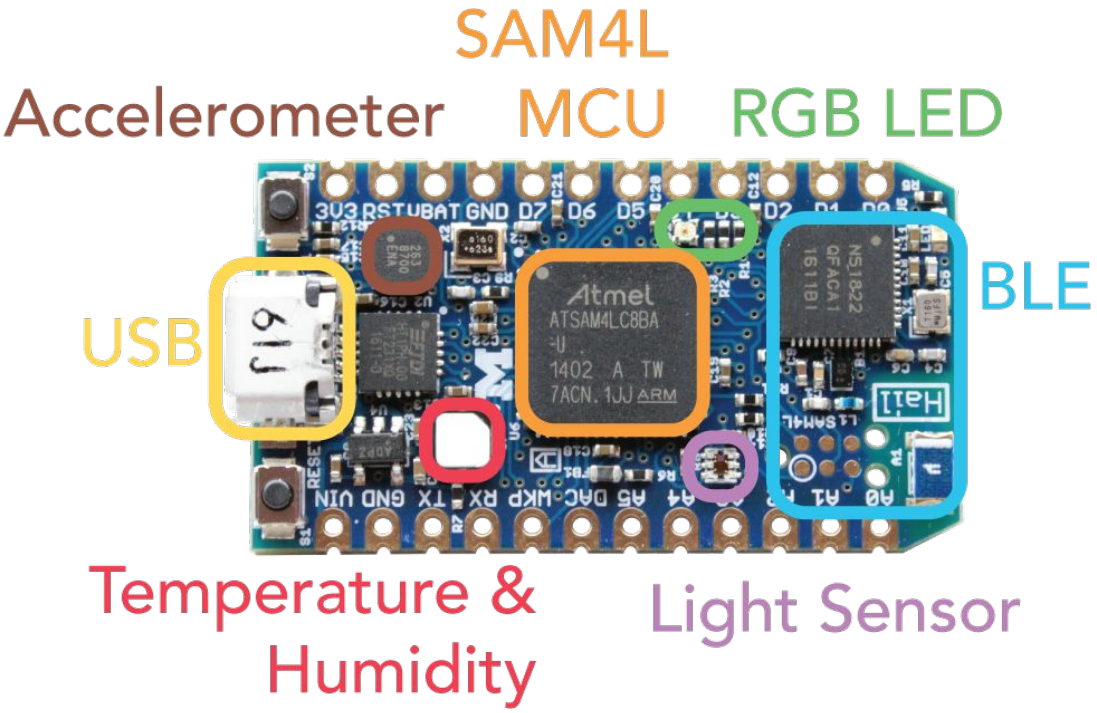
Einleitung

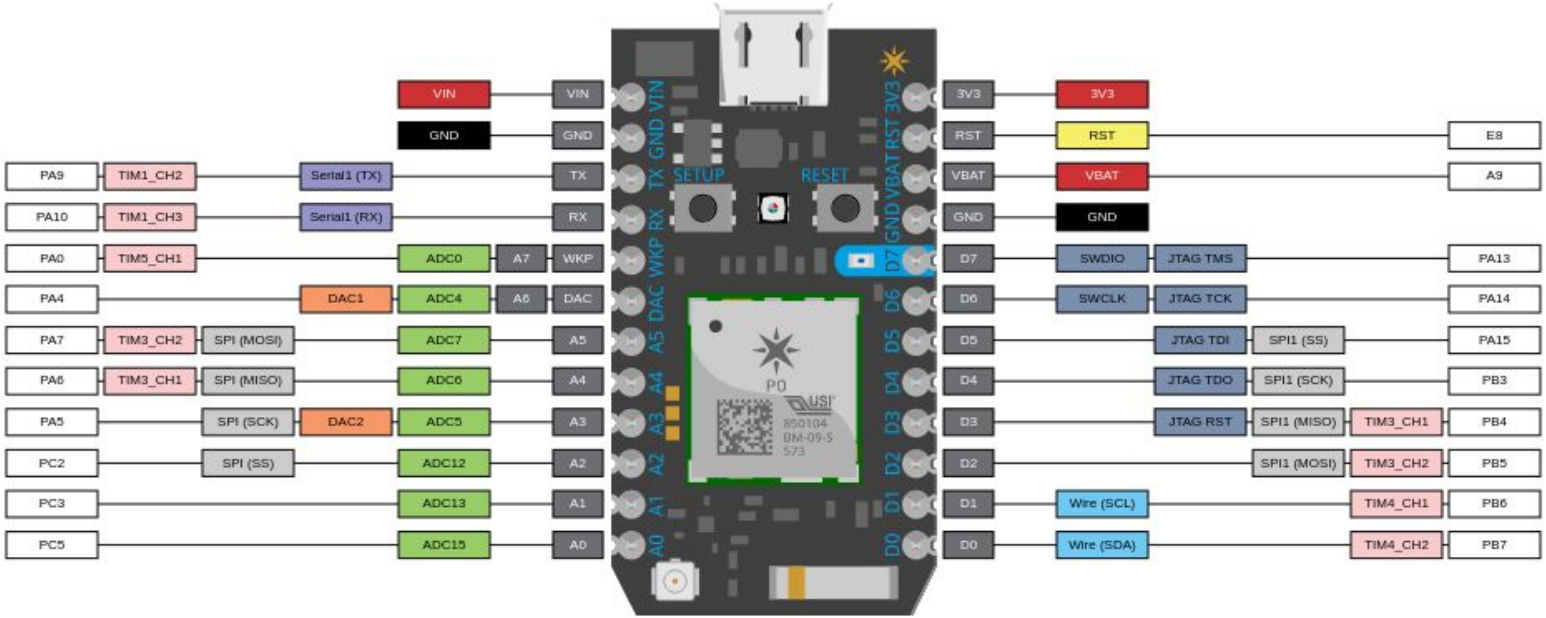

Hail
A0
A1
A2
A3
A4
A5
DAC
WKP
RX
TX
GND
VIN
D0
D1
D2
D3
D4
D5
D6
D7
GND
3V3
RST
VBAT

TO-DO Liste:

- Benötigte Software installieren
- Apps kompilieren
- Apps auf Board laden und testen
- Eigene Apps schreiben
- Kernel/Bibliotheken erweitern

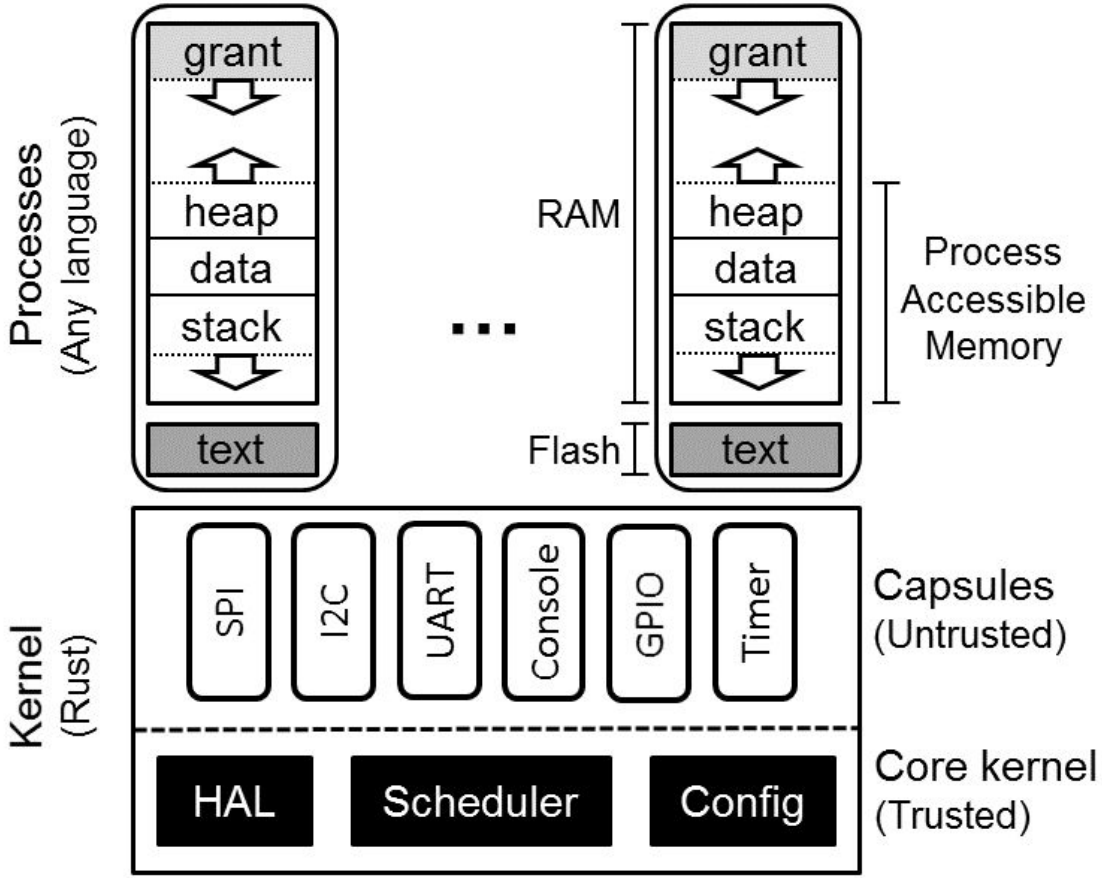
Hail Board


SAM4L
Accelerometer
MCU
RGB LED
BLE
USB
Temperature &
Humidity
Light Sensor
Atmel
ATSAM4LC8BA
-U
Hail


VIN
GND
PA9
TIM1_CH2
Serial1 (TX)
PA10
TIM1_CH3
Serial1 (RX)
PA0
TIM5_CH1
ADC0
A7
WKP
PA4
DAC1
ADC4
A6
DAC
PA7
TIM3_CH2
SPI (MOSI)
ADC7
A5
PA6
TIM3_CH1
SPI (MISO)
ADC6
A4
PA5
SPI (SCK)
DAC2
ADC5
A3
PC2
SPI (SS)
ADC12
A2
PC3
ADC13
A1
PC5
ADC15
A0
SETUP
RESET
PO
3V3
RST
VBAT
GND
D7
D6
D5
D4
D3
D2
D1
D0
SWDIO
JTAG TMS
SWCLK
JTAG TCK
JTAG TDI
SPI1 (SS)
JTAG TDO
SPI1 (SCK)
JTAG RST
SPI1 (MISO)
SPI1 (MOSI)
Wire (SCL)
Wire (SDA)
TIM3_CH1
TIM3_CH2
TIM4_CH1
TIM4_CH2
E8
A9
PA13
PA14
PA15
PB3
PB4
PB5
PB6
PB7

TockOS

- Betriebssystem für Embedded-Bereich
- Vollständig in Rust geschrieben
- Managen von Trade-off zwischen Isolation und Ressourcen
- Anwendungen wird grundsätzlich nicht vertraut


Processes
(Any language)
grant
heap
data
stack
text
...
RAM
Flash
grant
heap
data
stack
text
Process
Accessible
Memory
Kernel
(Rust)
SPI
I2C
UART
Console
GPIO
Timer
Capsules
(Untrusted)
HAL
Scheduler
Config
Core kernel
(Trusted)

Capsules

- Rust Struct + dessen Funktionen
- Kooperatives Scheduling, privilegierter Modus, Single-threaded, isoliert von Kernel + Userland, Kernel-Stack

Processes

- Preemptives Scheduling, unprivilegiert, Hardwareseitige Isolation während Runtime (MPU), eigene Memory Region

Grants

- Memory-Regionen, die geschützt sind vor Zugriffen von Prozessen

Category	Capsule	Process
Protection	Language	Hardware
Memory Overhead	None	Separate stack
Protection Granularity	Fine	Coarse
Concurrency	Cooperative	Preemptive
Update at Runtime	No	Yes

Memory Protection:

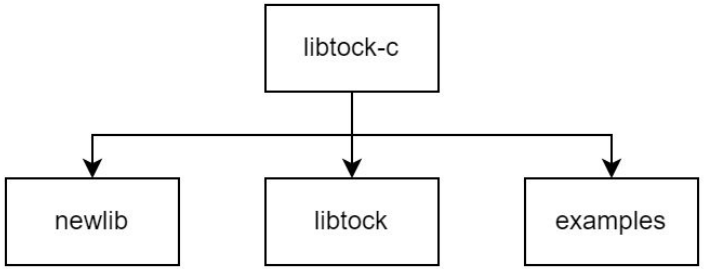
- Compile-Time Memory Safety durch Rust
- MPU (Memory Protection Unit), limitiert Zugriff auf Adressen
- Grants - Gesicherter Bereich im Prozess-Speicher

Memory Management:

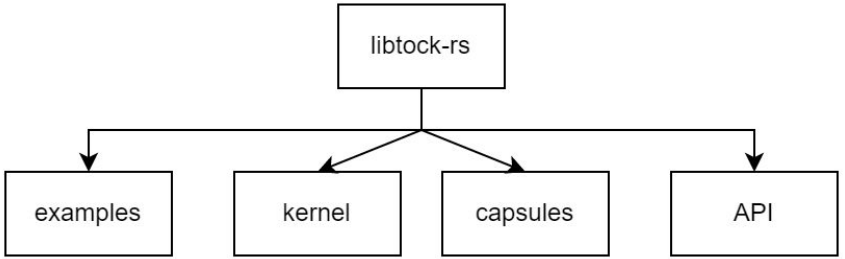
- Keine dynamischen Allokationen durch Capsules
- Daher Allokationen über die Grant-Regionen

Tock Userland

Libtock-C



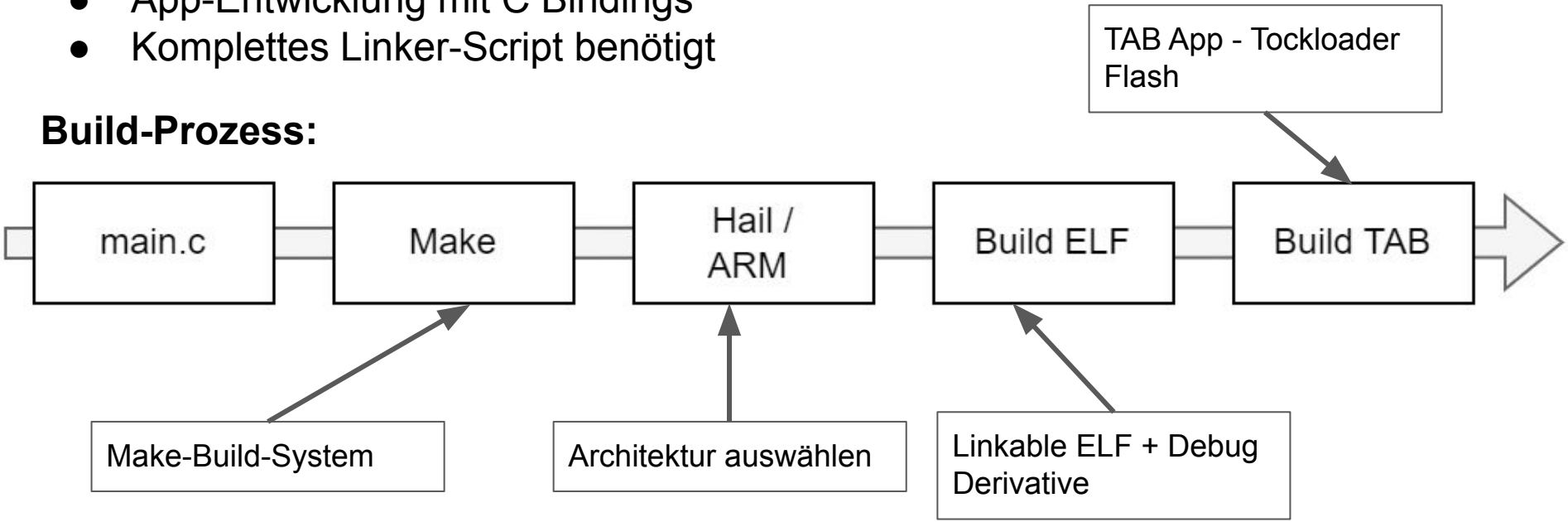
Libtock-RS



Libtock-C:

- App-Entwicklung mit C Bindings
- Komplettes Linker-Script benötigt

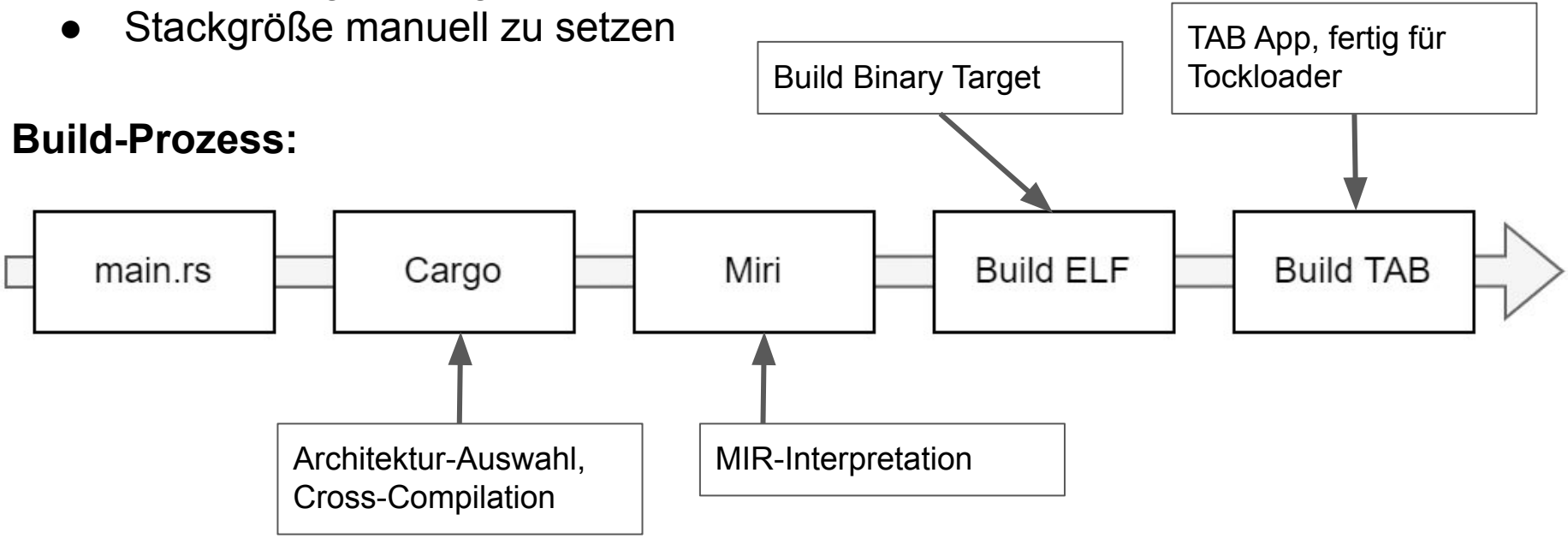
Build-Prozess:



Libtock-RS:

- WIP - wenig Bindings für Userland
- Stackgröße manuell zu setzen

Build-Prozess:



Tockloader

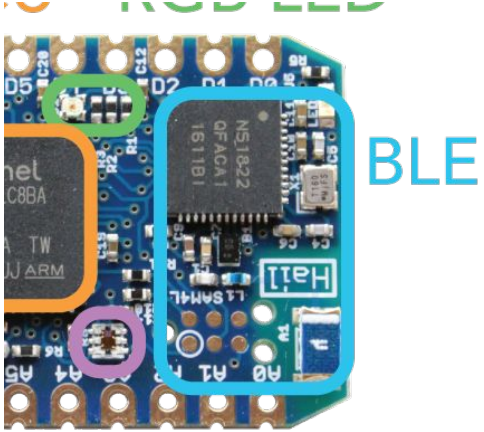
- Python-Tool zum installieren von Tock-Apps auf Hardware

Wichtigste Befehle
`$ tockloader install`
`$ tockloader uninstall <name>`
`$ tockloader list`
`$ tockloader listen`
`$ tockloader update <name>`
`$ tockloader erase-apps`

BLE Radio

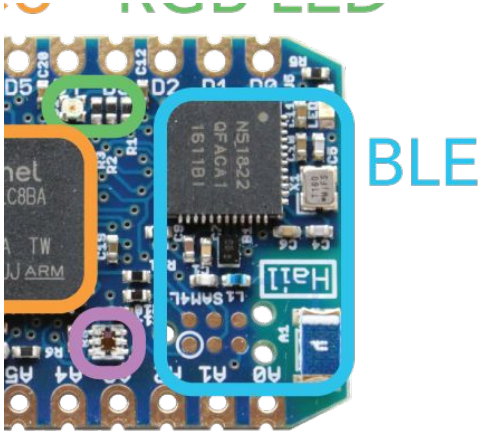
```c
int ble_start_advertising(int pdu_type, uint8_t* advd, int len, uint16_t interval) {
  allow_ro_return_t err = allow_readonly(BLE_DRIVER_NUMBER, BLE_CFG_ADV_BUF_ALLOWRO, advd, len);
  if (!err.success) return tock_status_to_returncode(err.status);

  syscall_return_t res = command(BLE_DRIVER_NUMBER, BLE_ADV_START_CMD, pdu_type, interval);
  return tock_command_return_novalue_to_returncode(res);
}
```

```
pub fn start_advertising(pdu_type: PDUType, advertising_data: &[u8], interval: u16) -> Result<u32, ErrorCode> {
    share::scope::<
        AllowRo<_, DRIVER_NUM, { allow_ro::ADV_DATA }>,
        _,
        _,
    >(|handle| {
        S::allow_ro::<C, DRIVER_NUM, { allow_ro::ADV_DATA }>(handle, advertising_data)?;
        S::command(DRIVER_NUM, BLE_ADV_START, pdu_type as u32, interval as u32).to_result()
    })
}
```



BLE Radio

- In C: libnrfserialization
- simple_ble setzt darauf auf und bietet High-Level API
- libnrfserialization ist ein Tock-spezifisches Projekt
- Rust-Bindings nicht vorhanden



Demo

Fazit

- Entwicklung von Anwendungen aufgrund mangelnden Supports und Dokumentation gestaltet sich schwierig
- Viele Features fehlen in der Rust-Bibliothek
- App-Entwicklung in C dennoch einigermaßen gut möglich

Bildquellen

1. https://www.tockos.org/blog/2017/introducing-hail/, 12.10.2022
2. https://docs.particle.io/reference/datasheets/wi-fi/photon-datasheet/, 13.10.2022
3. https://www.tockos.org/documentation/design/, 13.10.2022