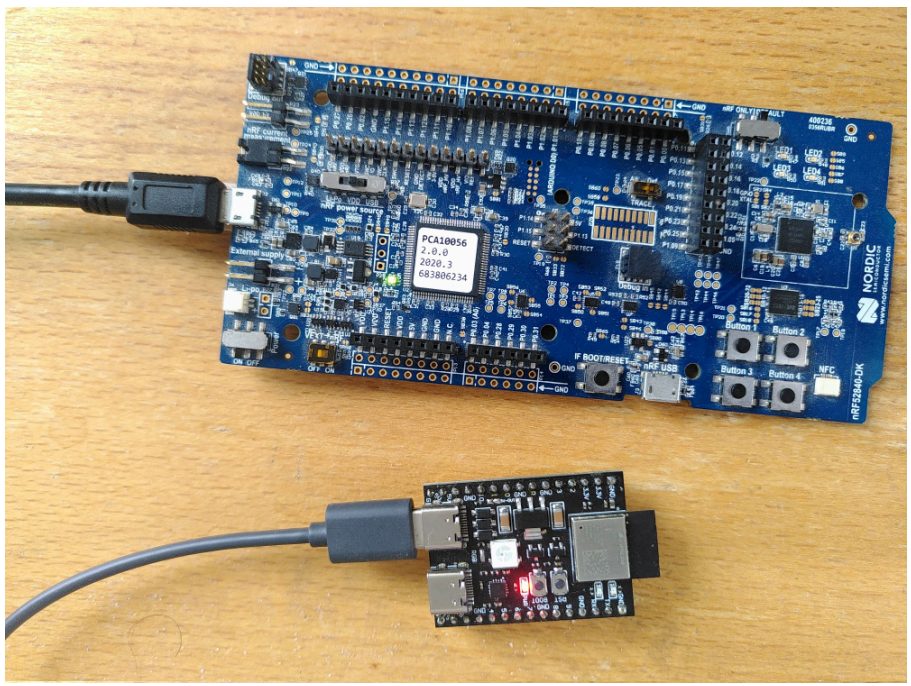
The Future Must be Updated

About me

Koen Zandberg

- Embedded software engineer
- RIOT Maintainer since 2017
- Currently maintainer of Ariel OS
  - Working on Over-the-air Updates

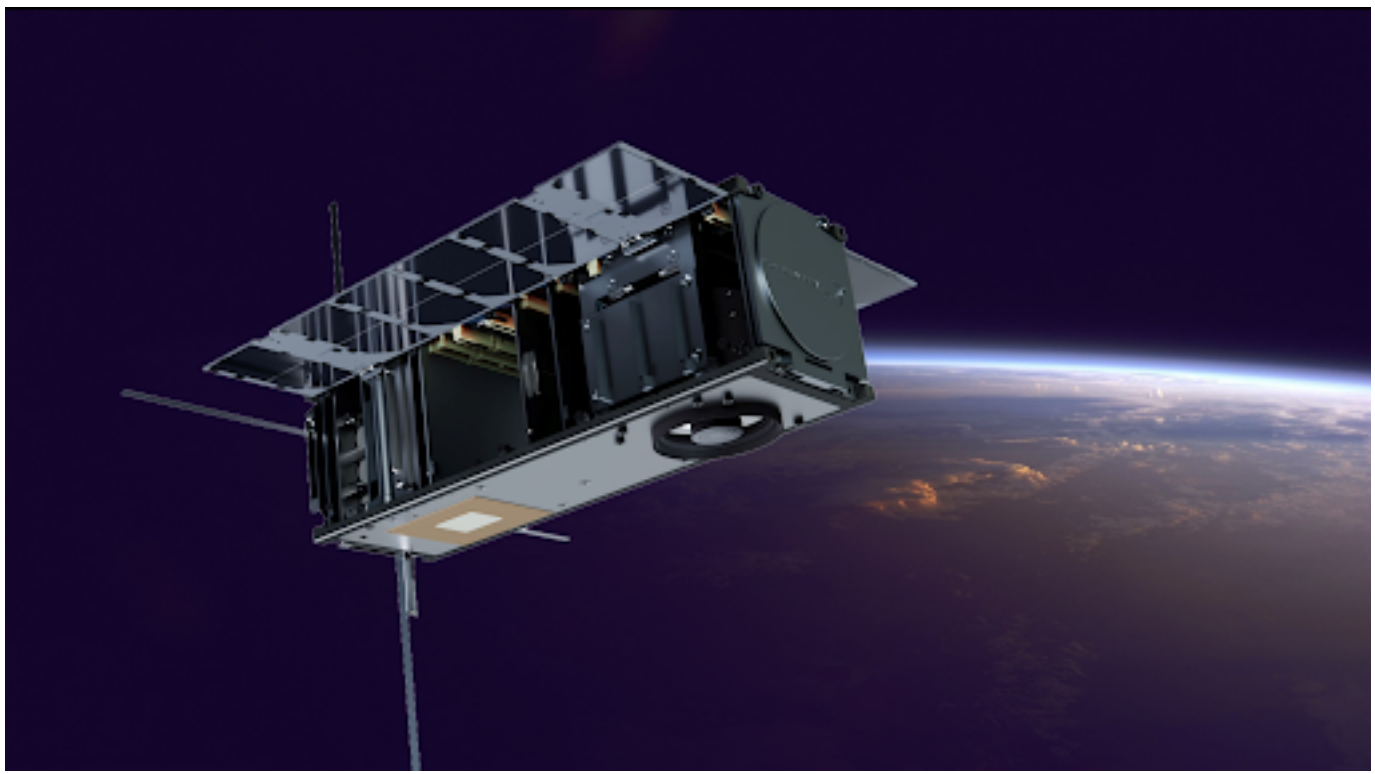
Local Development



Documented workflow:

```
laze build run -b nrf52840dk -C examples/hello-world
```

Remote Development



Why Update

- Bug fixes
- New features
- Adjusting mission
- Configuration updates

Why Update

- Bug fixes
- New features
- Adjusting mission
- Configuration updates

Cannot secure what you cannot update

Outline

Background

Installing our update

Embedded equivalent of:

```
curl http://firmware.example.com/install-latest.sh | sh
```

Installing our update

Embedded equivalent of:

~~`curl http://firmware.example.com/install-latest.sh | sh`~~

```
coap-client coap://firmware.example.com/install-latest.sh | sh
```

Installing our update

Embedded equivalent of:

~~`curl http://firmware.example.com/install-latest.sh | sh`~~

```
coap-client coap://firmware.example.com/install-latest.sh | sh
```

We can do better!

Security

- Firmware tampering
- Replay attack
- Incorrect firmware for device

Standards

Already out there:

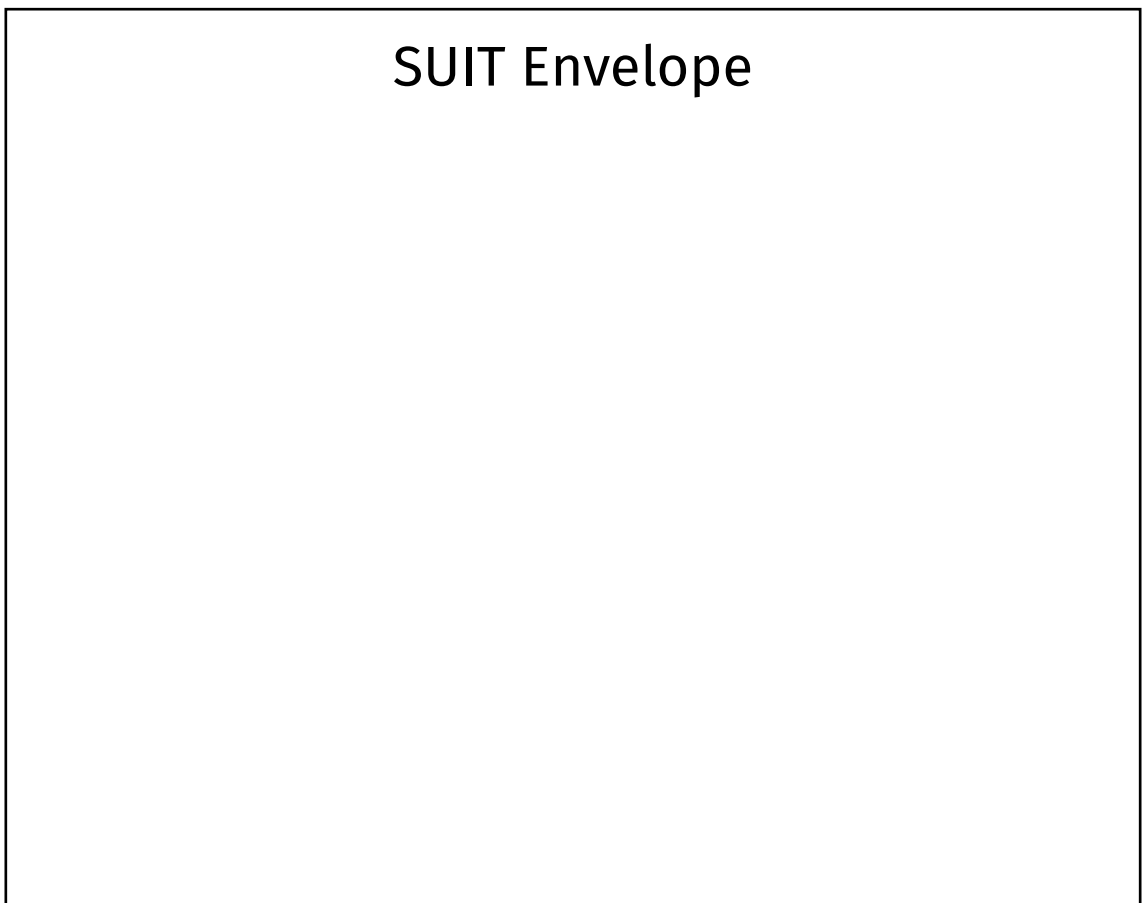
- **Application layer**: CoAP
- **Data format**: CBOR
- **Authentication**: COSE
- **Software Updates**: SUIT

The building blocks are there

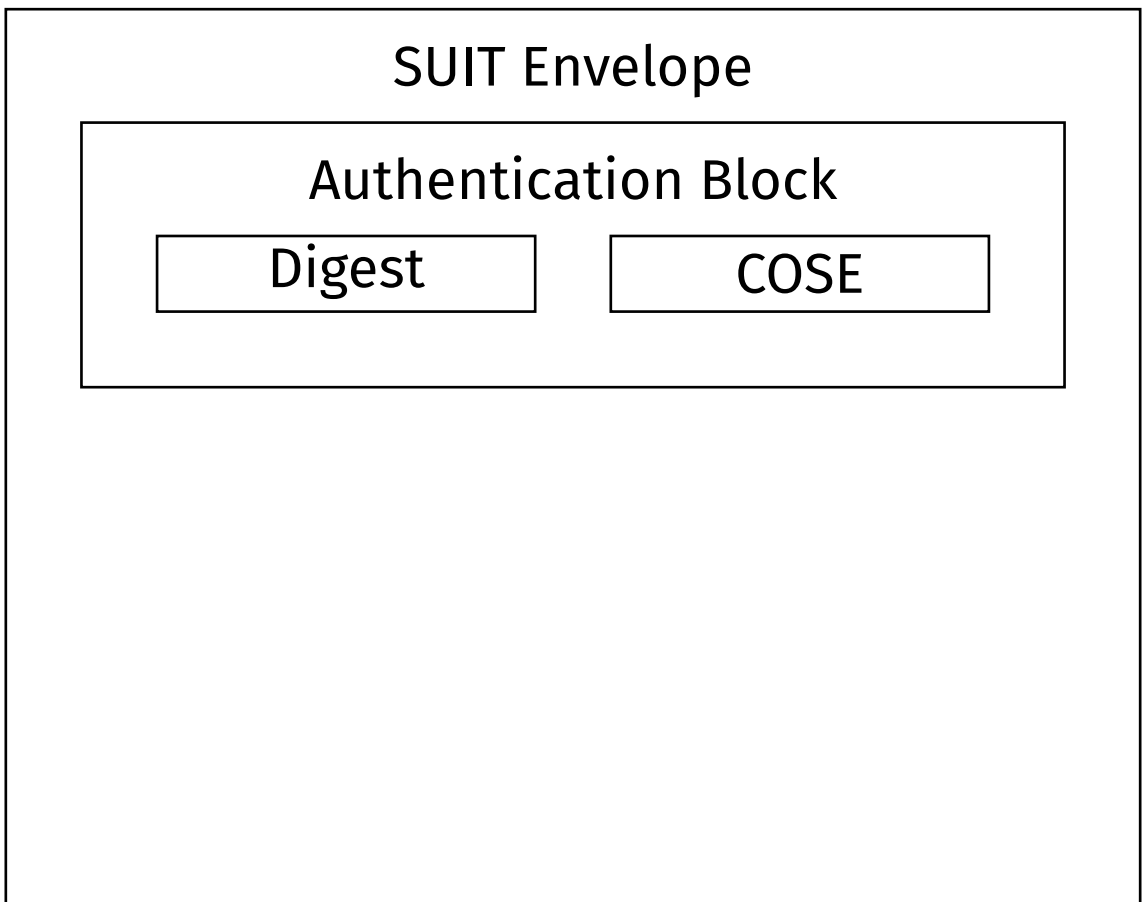
SUIT

- Manifest, describes the update process
- Contains the payload conditions and installation process.
- CBOR encoding
- COSE authentication

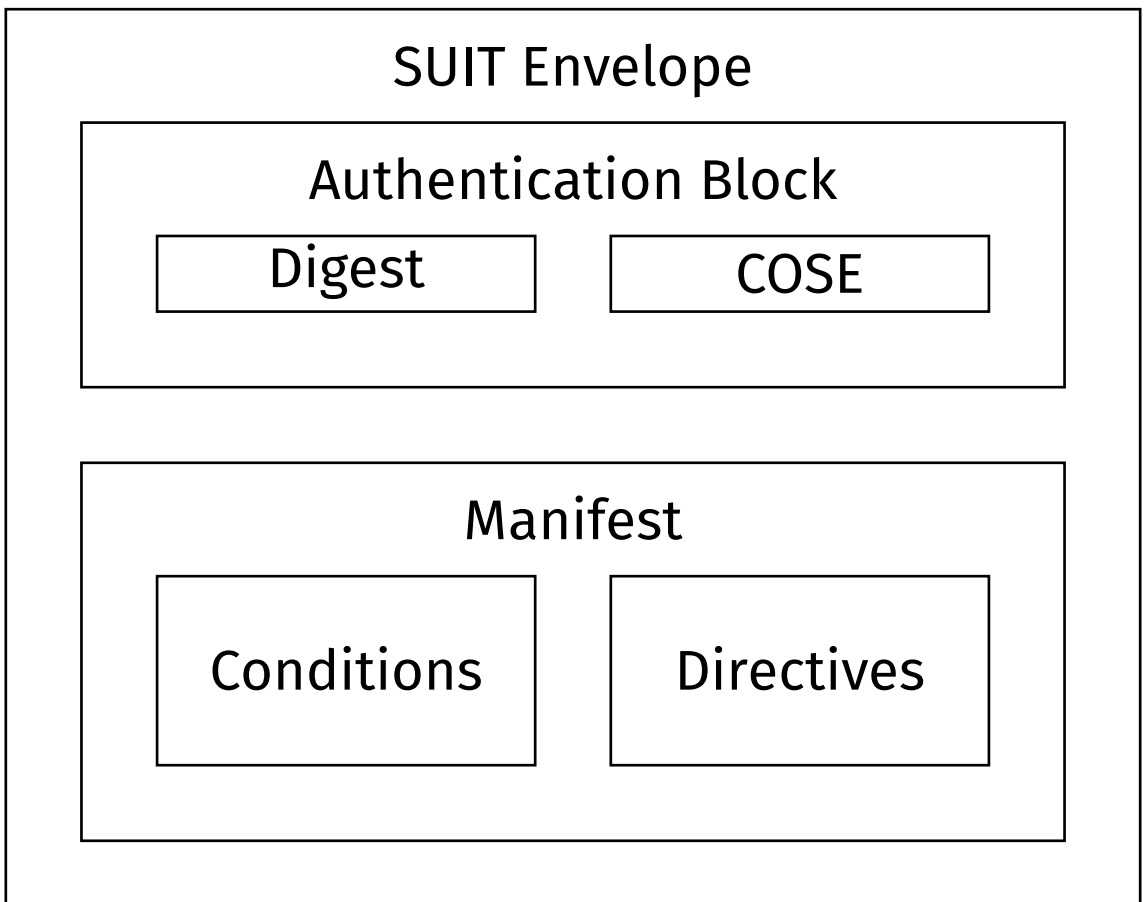
SUIT



SUIT

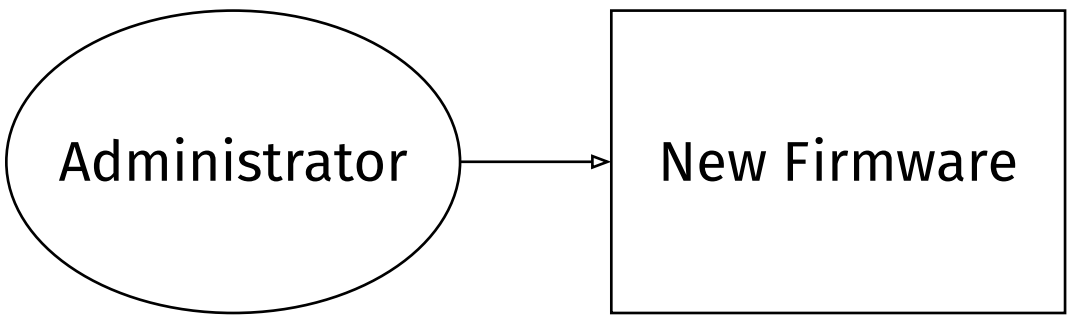


SUIT

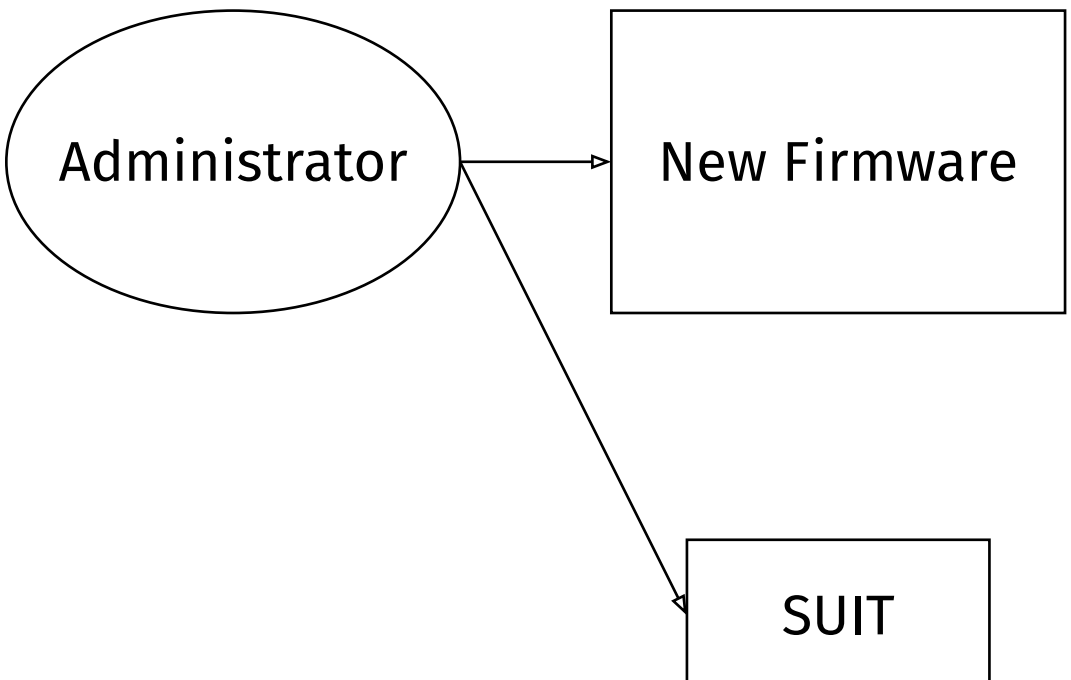

SUIT Envelope
Authentication Block
Digest
COSE
Manifest
Conditions
Directives

Workflow

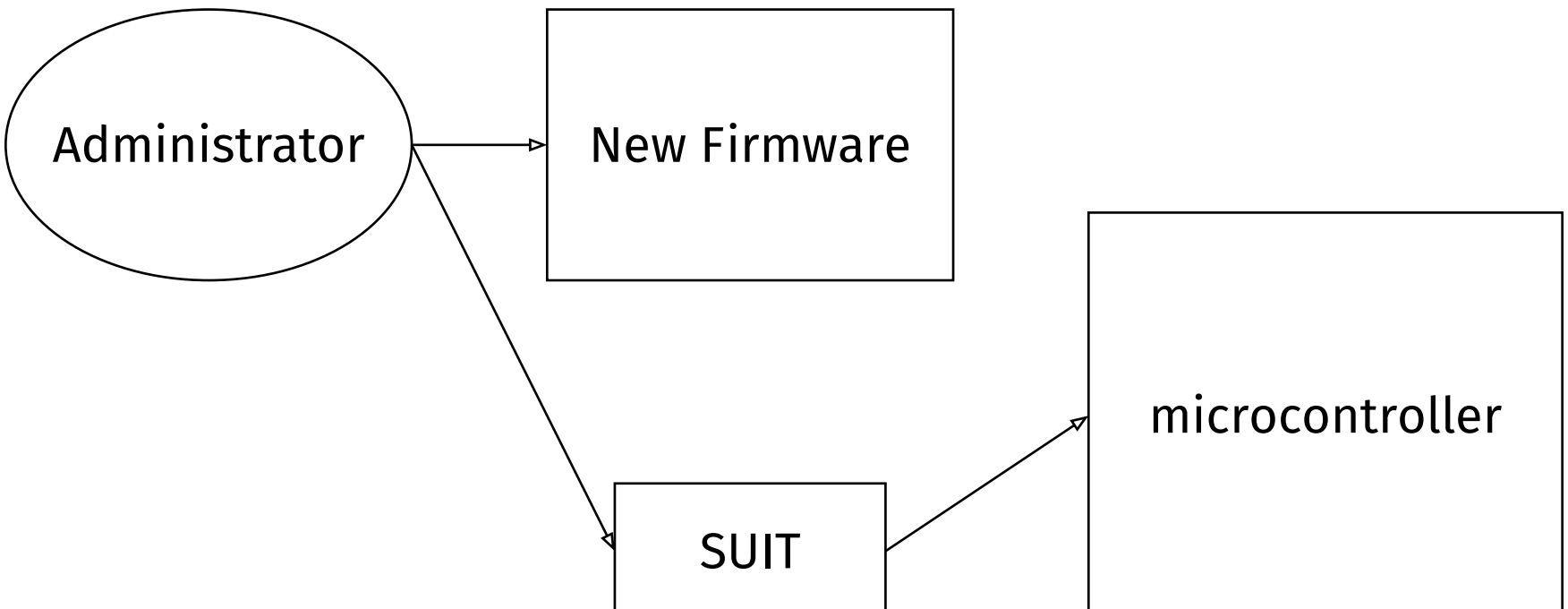
Workflow



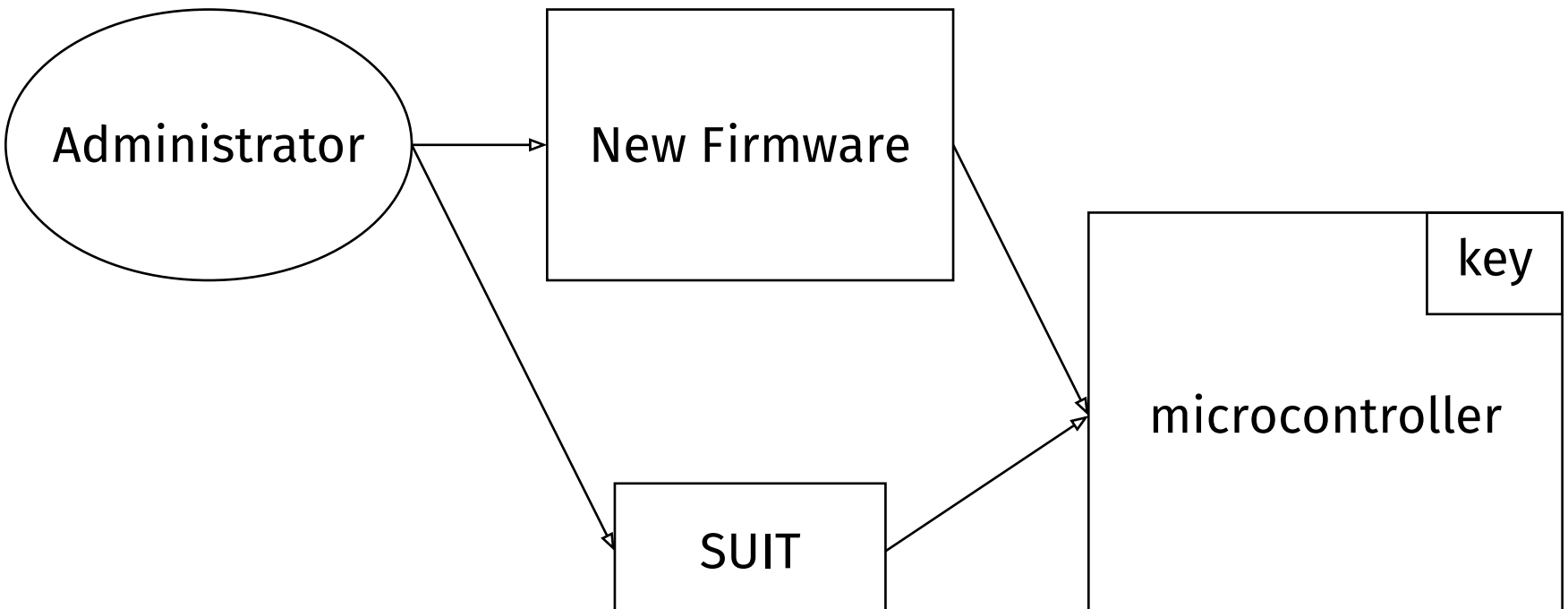
Workflow



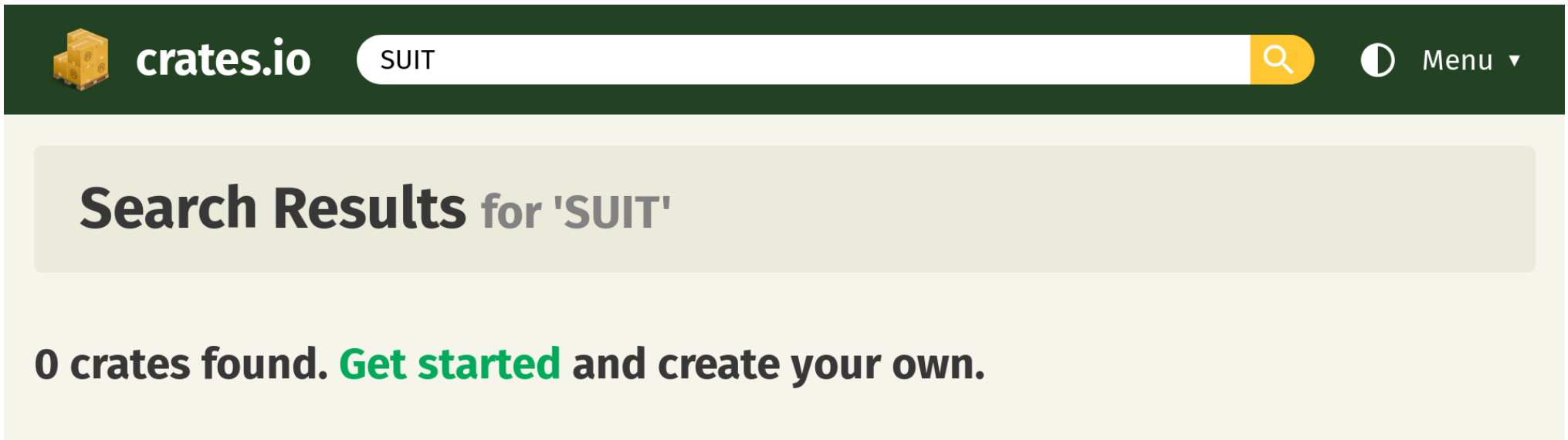
Workflow



Workflow



SUIT


crates.io
SUIT
Menu
Search Results for 'SUIT'
0 crates found. Get started and create your own.

Dress-Up

Dress-Up

- Parser-only implementation of the SUIT manifest format
- Minicbor
- no_std
- Protocol and storage agnostic
- Requires OS integration

Workflow

1. Provide conditions and directive implementations
2. Call Dress-up with received manifest
3. Parse Manifest
4. Authenticate manifest
4. Dress-up calls implementations
5. ...
6. Updated firmware

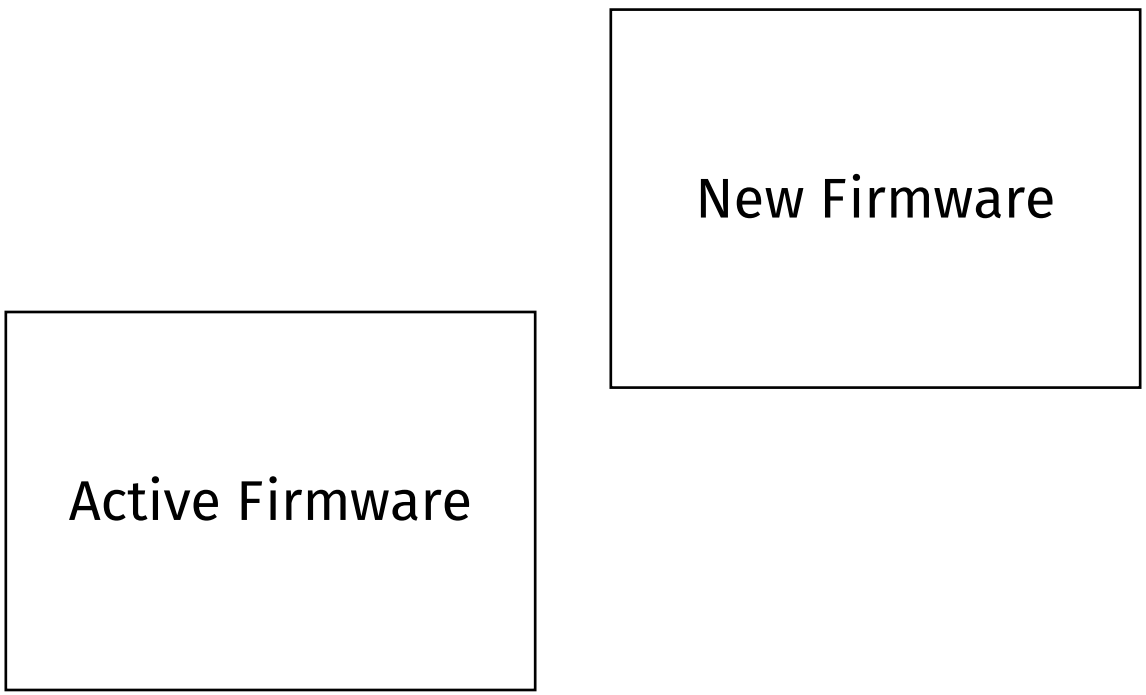
Dress-Up

```Rust
let suit = SuitManifest::from_bytes(&bytes);
let suit = suit.authenticate(|cose, payload| {
  authenticate(cose, payload)
})?;
let envelope = suit.envelope()?;
let manifest = envelope.manifest()?;
if let Err(e) = manifest.execute_payload_installation(&hooks).await {
    error!("Could not install payload: {:?}", e);
}
```

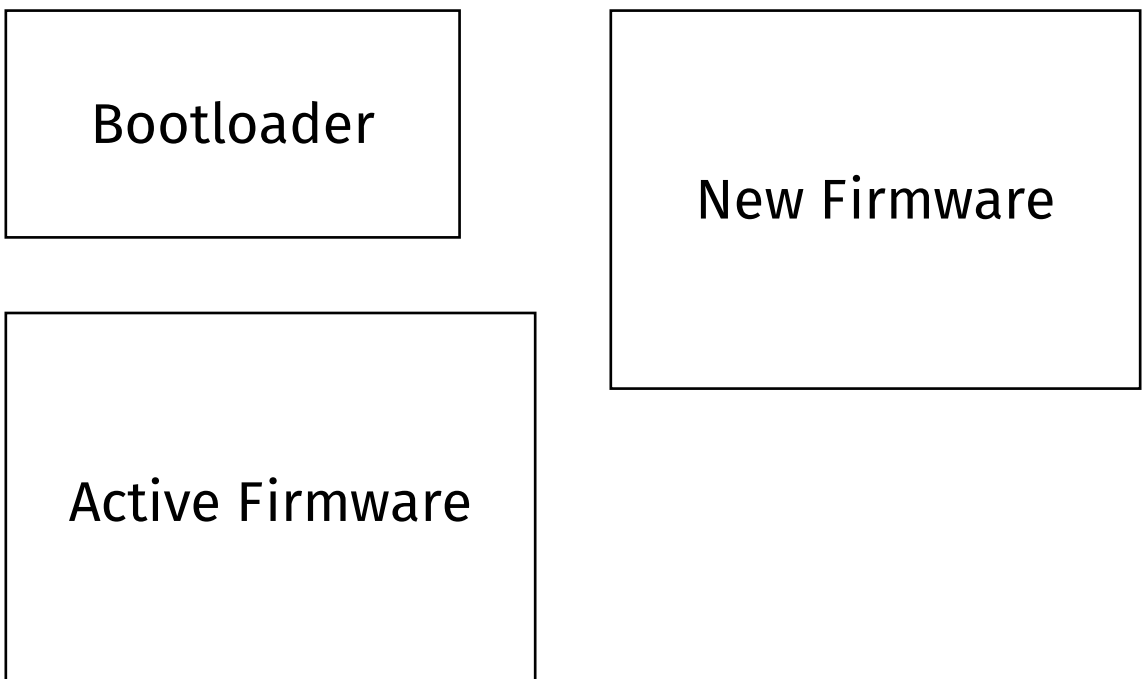
Memory Layout

Requirements

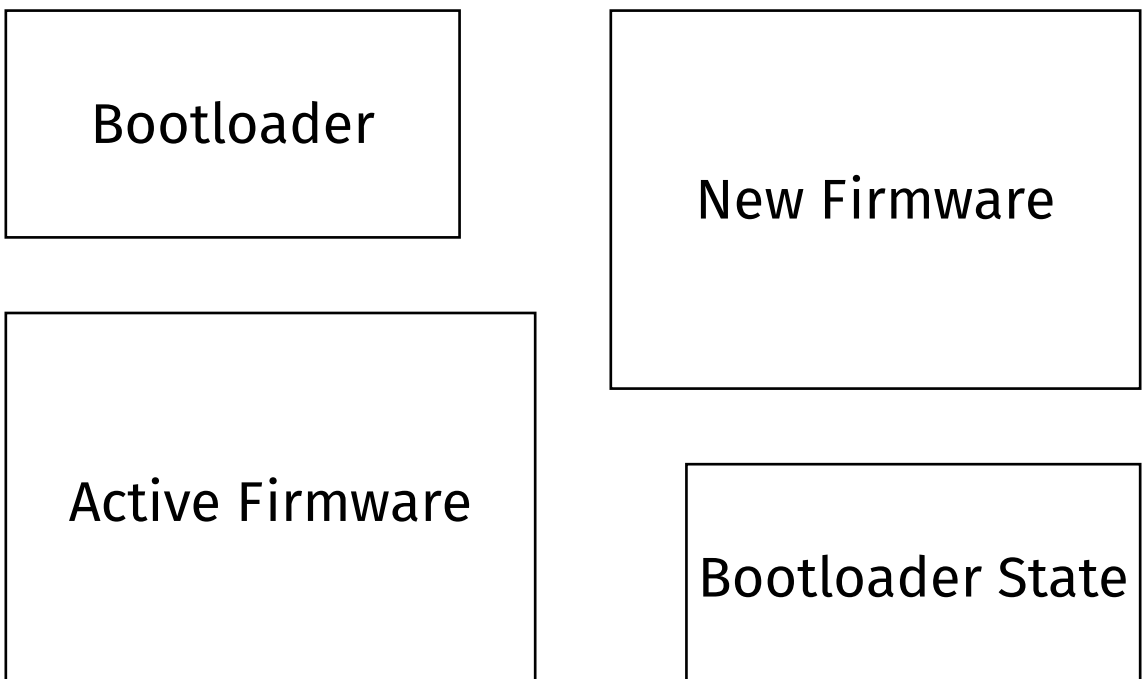
Requirements



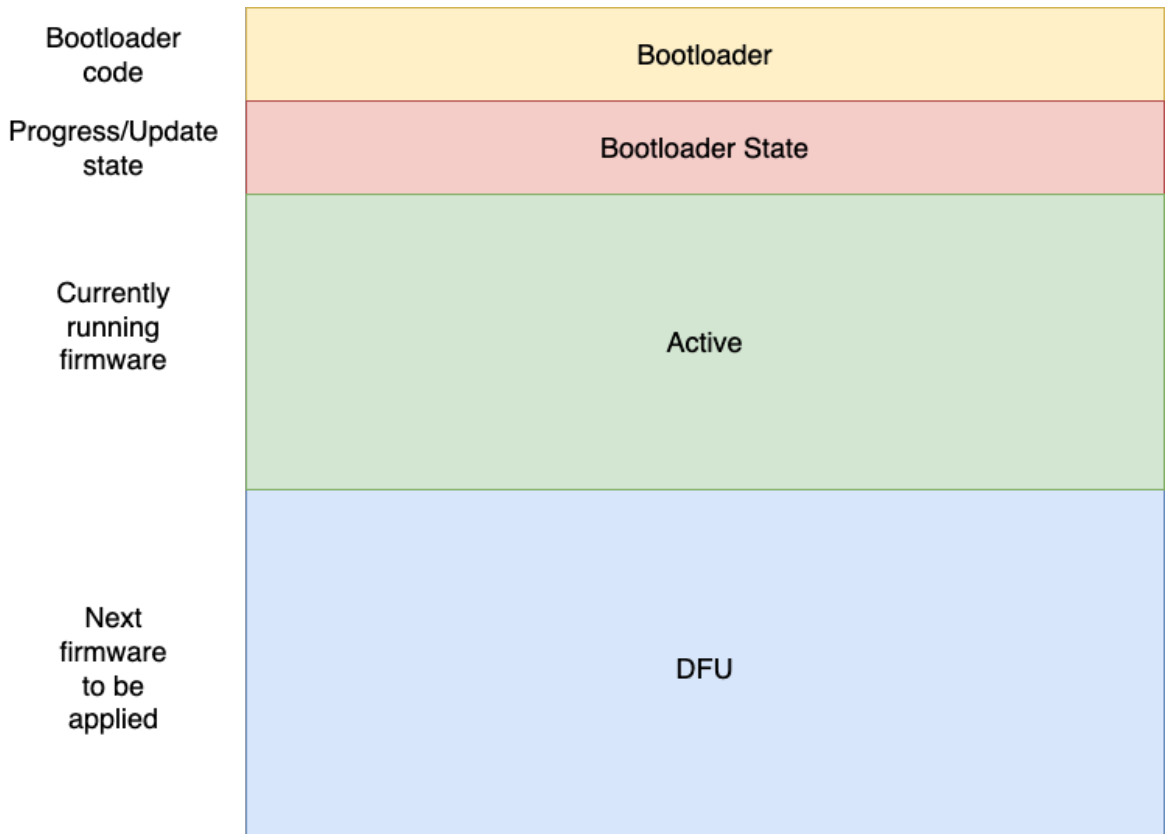
Requirements



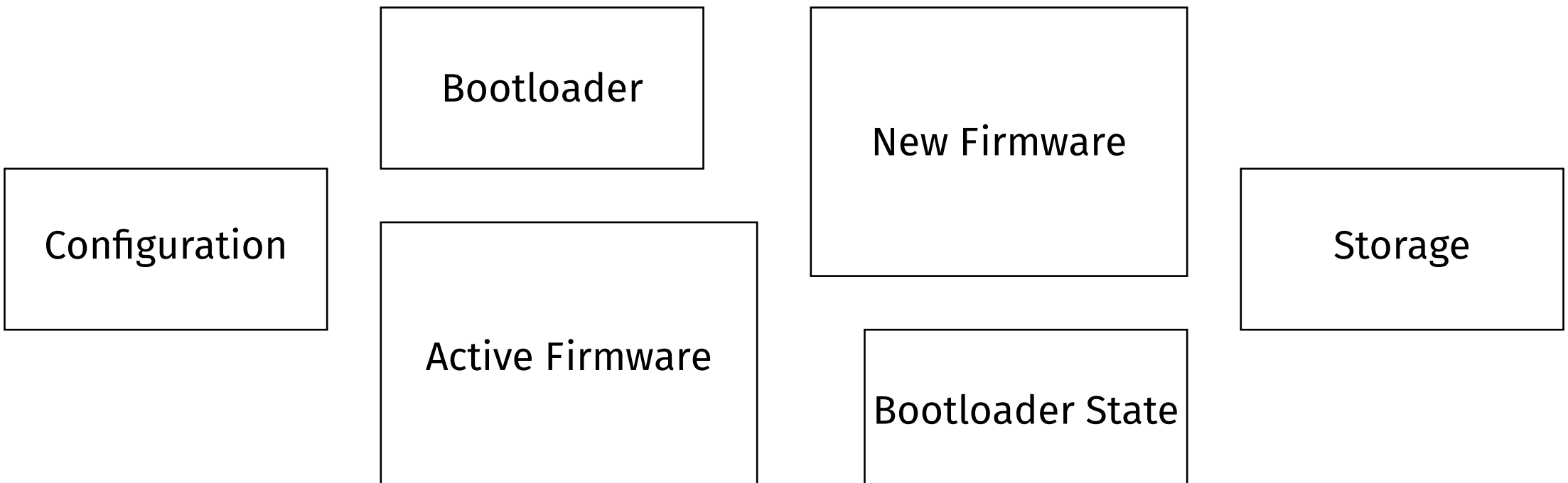
Requirements



Solution: Embassy-boot



Solution?


Bootloader
New Firmware
Configuration
Storage
Active Firmware
Bootloader State

Ariel OS: memory layout

- Chip Memory size
  - Page size
  - Total pages
- Bootloader
- Storage
- Updates

Ariel OS: memory layout

- Chip Memory size
  - Page size
  - Total pages
- Bootloader
- Storage
- Updates

Depends on the active modules and the target board

Memsolve

Requirements

Generate the linker layout at built-time

- Partitions & Requirements
  - Minimum size
  - Minimum flash pages
  - Alignment
  - Specific address
  - As large as possible
  - Relative size

Requirements

Generate the linker layout at built-time

- Partitions & Requirements
  - Minimum size
  - Minimum flash pages
  - Alignment
  - Specific address
  - As large as possible
  - Relative size
- **NP-hard** problem

Memsolve

- Describe requirements as a set of linear equations.
- Linear programming solver
- Built-time non-volatile memory partitioning

Memsolve

- Describe requirements as a set of linear equations.
- Linear programming solver
- Built-time non-volatile memory partitioning

Turns vague memory requirements into real linker memory regions

Solving the layout

- MCU: **8 pages** of flash total
- Bootloader: **2 pages**, start of flash
- Update State: **1 page**
- Firmware: As large as possible

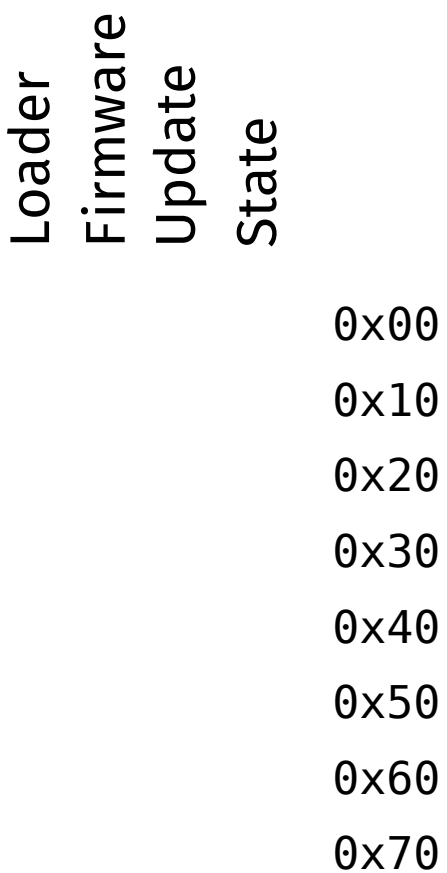
- Update: 1 page larger than the firmware

Solving the layout

- MCU: **8 pages** of flash total
- Bootloader: **2 pages**, start of flash
- Update State: **1 page**
- Firmware: As large as possible, **2 pages**
- Update: 1 page larger than the firmware, **3 pages**

Solving the layout

- MCU: **8 pages** of flash total
- Bootloader: **2 pages**, start of flash
- Update State: **1 page**
- Firmware: As large as possible, **2 pages**
- Update: 1 page larger than the firmware, **3 pages**



Solving the layout

- MCU: **8 pages** of flash total
- Bootloader: **2 pages**, start of flash
- Update State: **1 page**
- Firmware: As large as possible, **2 pages**
- Update: 1 page larger than the firmware, **3 pages**

Loader	Firmware	Update	State	
x	x	x	x	0x00
	x	x	x	0x10
	x	x	x	0x20
	x	x	x	0x30
	x	x	x	0x40
	x	x	x	0x50
	x		x	0x60
			x	0x70

Solving the layout

- Partitions must be allocated exactly once
- Addresses must have at most one partition starting
- Partitions block following addresses according to their size

Loader	Firmware	Update	State	
x	x	x	x	0x00
	x	x	x	0x10
	x	x	x	0x20
	x	x	x	0x30
	x	x	x	0x40
	x	x	x	0x50
	x		x	0x60
			x	0x70

Solving the layout

- Partitions must be allocated exactly once
- Addresses must have at most one partition starting
- Partitions block following addresses according to their size

Loader	Firmware	Update	State	
1	0	0	0	0x00
	0	0	0	0x10
	0	0	1	0x20
	0	1	0	0x30
	0	0	0	0x40
	0	0	0	0x50
	1		0	0x60
			0	0x70

Solved the layout

Generating LD `MEMORY` regions

```
1 MEMORY
2 {
3     Loader : ORIGIN = 0x0, LENGTH = 0x20
4     State : ORIGIN = 0x20, LENGTH = 0x10
5     Update : ORIGIN = 0x30, LENGTH = 0x30
6     Firmware : ORIGIN = 0x60, LENGTH = 0x20
7 }
```

Integration

How to draw an owl

1.

2.

1. Draw some circles

2. Draw the rest of the fucking owl

Future Work

Building blocks

- Dress-up
  - SUIT manifest parser
- Memsolve
  - Partitions non-volatile memory

To be done

- Manifest submission
- Non-volatile memory writing
- Payload retrieval
- Bootloader integration

Next Up:

- Integrate Memsolve:
  - Storage module
  - OTA update module

Next Up:

- Integrate Memsolve:
  - Storage module
  - OTA update module
- Integrate Dress-Up:
  - Submit SUIT manifest
  - Trigger update process
  - Provide payload retrieval functions (CoAP)
  - Provide storage partitions

Admin side:

- SUIT manifest generator
- Firmware store
- Management interface

Admin side:

- SUIT manifest generator
- Firmware store
- **Fleet** Management interface

Conclusion

Building blocks:

- Dress-Up
- Memsolve

Challenges:

- Ariel OS: Coherent integration
- Admin side: Fleet management

Thanks!

https://crates.io/crates/dress-up

https://crates.io/crates/memsolve