State of Tock

TockWorld 7 | June 2024

A Brief History of Tock

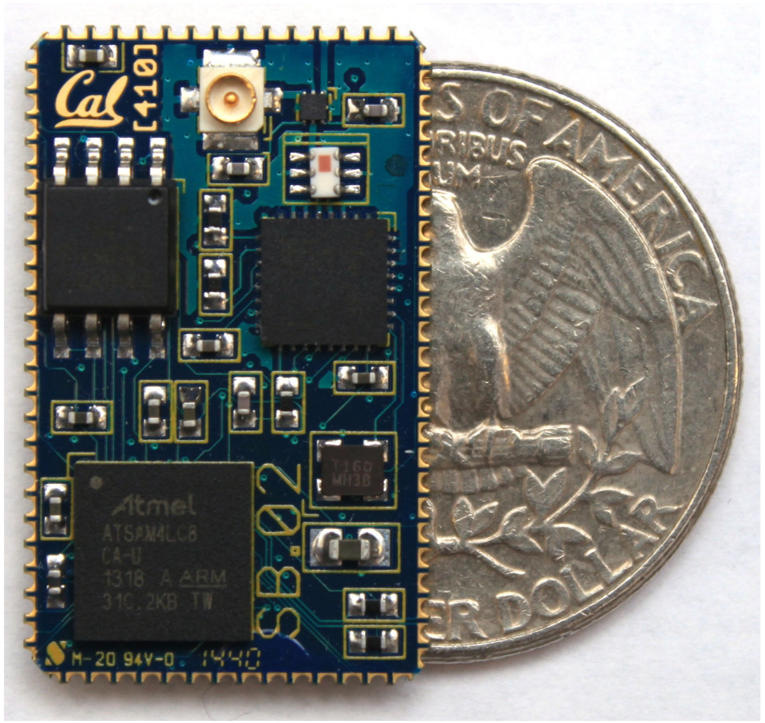
It all started July 2014

From: Philip Levis
Subject: [helena-project] SenSys poster/demo
To: helena-project@lists.stanford.edu
Date: Wed, 09 Jul 2014 13:15:12 -0700

> *Each of us has made some interesting progress on a next generation embedded platform. Michael @Berkeley has storm (an M4 + RF233), Tom and Amit @Stanford have a TinyOS nRF8001 stack and some interesting measurements for BLE, Pat and Brad @Michigan have a few Cortex M platforms, with the CC2420.*
>
> *Operating system: what should an operating system for such a device look like? Can we achieve something like the efficiency and dependability of TinyOS without being so difficult to extend and program?*

In the beginning there was `storm.rs`

Storm is a 16mm x 26mm (1/2" by 1") solder on module that combines a 32 bit Cortex M4 microcontroller with an 802.15.4 radio, and 64 Mbit of flash memory. This ultra low power module exports 80 pins via castellated edges, and is designed to be soldered into a larger PCB that contains additional sensors or actuators.



Reboot with First Tock commit

```
commit a14379b850bf47e89cd2945226cbf9bcbab5f43f
Author: Amit Aryeh Levy <amit@amitlevy.com>
Date: Tue May 19 15:29:44 2015

    Initial commit

    Barebones build system and boot to Rust on Storm
```

Since then…

- 2016: Dynamic userland code loading
- 2017: Tock training at RustConf, first deployment (Signpost)
- 2018: 1.0 release
- 2019: RISC-V support
- 2020: Pluggable scheduler
- 2021: 2.0 release, revised system call interface
- 2022: Subscribe & allow handled by kernel & read-only shared buffers
- 2022: Signed applications
- 2024: Kernel compiles on stable Rust

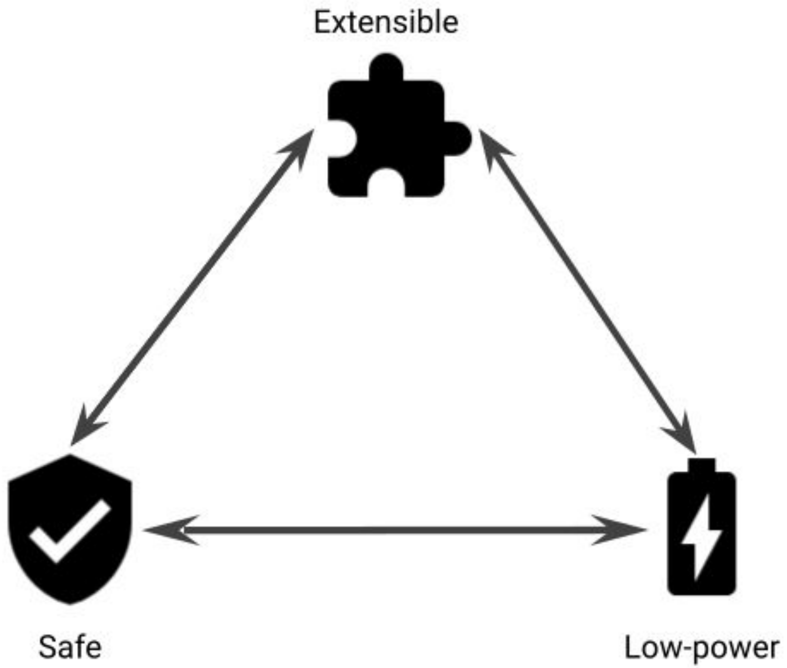
Project Mission

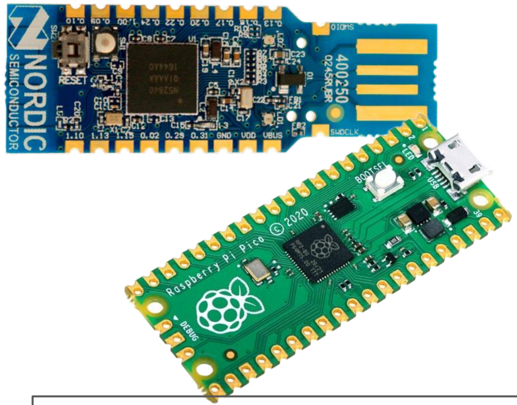
We believe in

Building embedded systems that

- are safe and secure
- people can program
- are resource efficient

A modest desire, for modest devices, with big implications.



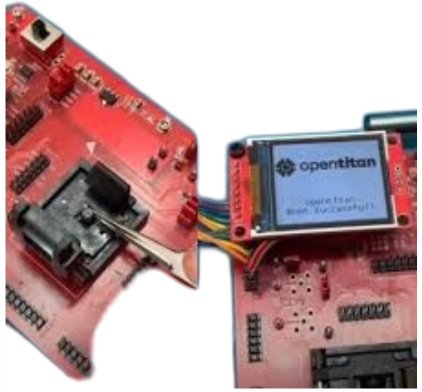


Development

HVAC Maintenance Services
Fire Detection & Alarm
Security & Access Control
Digital Video
Lighting Control & Retrofit
Smoke Control
Indoor Air Quality Services
Intrusion Detection
Energy Supply & Load Management
Water Management
Environmental Control
Mechanical Maintenance & Retrofit
Asset Locator
Enterprise Systems Integration
On-Site Technical Service
Energy Information Management

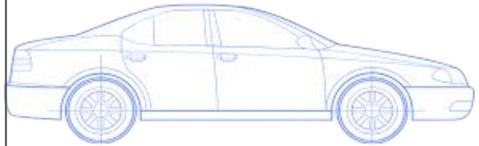
Buildings

Hardware Root-of-Trust



Payment/ID

Wearables & Medical Devices



Automotive

Tock project accomplishes this through

- Extensible & pluggable core
- Careful and pragmatic use of formal tools
    - Type-safety (Rust)
    - Hardware support for strong isolation
- Co-development with
    - Hardware
    - Language
    - Applications
- Open source collaboration
    - Practitioners
    - Researchers
    - Educators

A Year of Contributions

- Tock
  - 1524 commits, 421 merged PRs
  - 48 contributors, 26 new
- libtock-c
  - 636 commits
  - 18 contributors, 8 new
- libtock-rs
  - 161 commits
  - 13 contributors, 6 new
- tockloader
  - 80 commits
  - 10 contributors, 4 new
- book
  - 166 commits
  - 8 contributors, 2 new

Tock Today

- Deployed on millions of devices
  - Laptops, servers
  - Predominantly for hardware-root-of-trust
- Predominant development on ARM
- Predominant deployments on RISC–V
  - Notable exceptions
    - OpenSK
    - Automotive
    - Sensor networks
    - x86

Project Focus for 2024 and Beyond

Testing, Reliability, Security

- Hardware-based continuous integration
- Test coverage of kernel *functionality*
- Soundness
- Low-level assembly correctness

Next generation hardware settings

- Integrated settings
  - Non-Execute-in-Place Code Storage
  - Code-size more significant priority
  - Relocation easier! More efficient!
  - Big challenge: reference hardware implementations
- Hardware memory tagging
  - Becoming real!
    - See CHERI talk tomorrow
  - Could have significant influence on efficiency and granularity of isolation
- Multi-Chip Platforms
  - Already technically the case for, e.g., TI MCUs
  - Integrated settings interact with application core via shared memory
  - Not just SMP

Safe Integration with non-(safe)Rust Code

- Inevitable in practice
  - Cryptographic libraries
  - Vendor provided blobs
- A big problem for type-safety as isolation
- Current story
  - YOLO?
- Looking forward
  - Verified safety
  - Encapsulated Functions
  - User-level services

Still Networking

- Low-power wireless with Thread, LoRa, BLE
    - OpenThread as user-level service
    - Non-kernel networking seems inevitable
    - Need first-class support for sharing with applications
- Automotive networking
    - CAN
    - Ethernet
- USB! USB! USB!

Looking Forward to more Community

- Better supporting downstream users
    - Open communication
    - Tools & documentation
    - Responsive and open upstream
- Better supporting current and potential contributors
    - Tools & documentation
    - Remote on-hardware development
    - Trainings and teaching materials
- Better engagement with related open source
    - Rust, LLVM, RISC-V, OpenTitan, Caliptra
    - Rust embedded ecosystem