Wasefire

A framework for secure firmware development

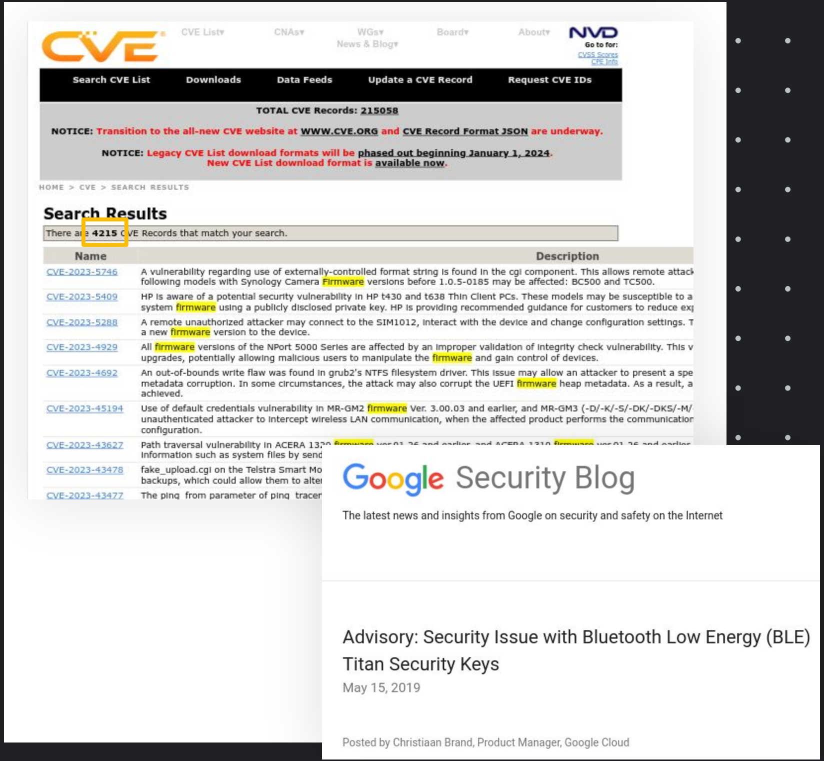
Julien Cretin
Google

Jean-Michel Picod
Google

with the help of other Googlers and external collaborators

Security and Privacy Research

Firmware security issues are still too frequent



CVE List▾ CNAs▾ WGs▾ News & Blog▾ Board▾ About▾

NVD Go to for: CVSS Scores CPE Info

Search CVE List | Downloads | Data Feeds | Update a CVE Record | Request CVE IDs

TOTAL CVE Records: 215058

NOTICE: Transition to the all-new CVE website at WWW.CVE.ORG and CVE Record Format JSON are underway.

NOTICE: Legacy CVE List download formats will be phased out beginning January 1, 2024.
New CVE List download format is available now.

HOME > CVE > SEARCH RESULTS

Search Results

There are 4215 CVE Records that match your search.

Name	Description
CVE-2023-5746	A vulnerability regarding use of externally-controlled format string is found in the cgi component. This allows remote attack following models with Synology Camera Firmware versions before 1.0.5-0185 may be affected: BC500 and TC500.
CVE-2023-5409	HP is aware of a potential security vulnerability in HP t430 and t638 Thin Client PCs. These models may be susceptible to a system firmware using a publicly disclosed private key. HP is providing recommended guidance for customers to reduce ex
CVE-2023-5288	A remote unauthorized attacker may connect to the SIM1012, interact with the device and change configuration settings. T a new firmware version to the device.
CVE-2023-4929	All firmware versions of the NPort 5000 Series are affected by an improper validation of integrity check vulnerability. This v upgrades, potentially allowing malicious users to manipulate the firmware and gain control of devices.
CVE-2023-4692	An out-of-bounds write flaw was found in grub2's NTFS filesystem driver. This issue may allow an attacker to present a spe metadata corruption. In some circumstances, the attack may also corrupt the UEFI firmware heap metadata. As a result, a achieved.
CVE-2023-45194	Use of default credentials vulnerability in MR-GM2 firmware Ver. 3.00.03 and earlier, and MR-GM3 (-D/-K/-S/-DK/-DKS/-M/ unauthenticated attacker to intercept wireless LAN communication, when the affected product performs the communication configuration.
CVE-2023-43627	Path traversal vulnerability in ACERA 1320 firmware ver.01.26 and earlier, and ACERA 1310 firmware ver.01.26 and earlier information such as system files by send
CVE-2023-43478	fake_upload.cgi on the Telstra Smart Mo backups, which could allow them to alter
CVE-2023-43477	The ping from parameter of ping tracer

Google Security Blog

The latest news and insights from Google on security and safety on the Internet

Advisory: Security Issue with Bluetooth Low Energy (BLE) Titan Security Keys

May 15, 2019

Posted by Christiaan Brand, Product Manager, Google Cloud

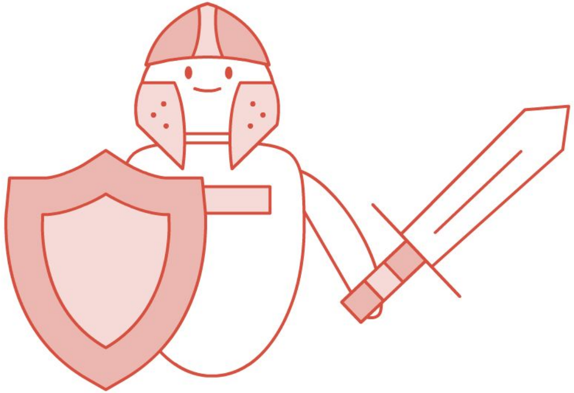
Common security issues

Memory corruption

Default credentials

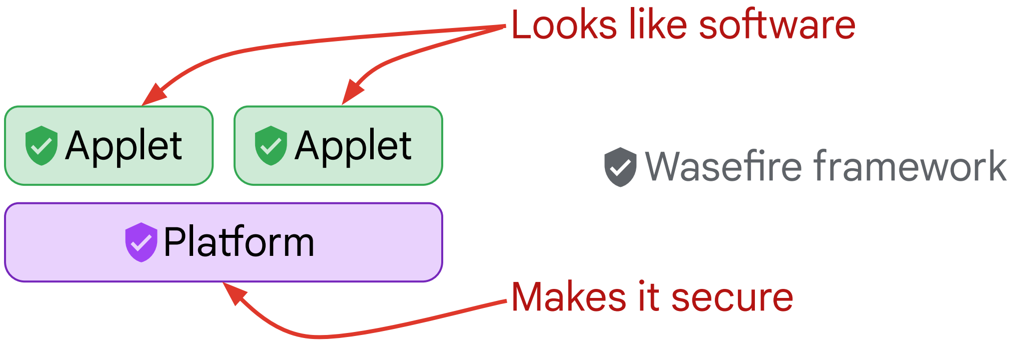
Side channel attacks

Manual or no updates



We need a secure platform that takes care of the security on behalf of the developer

Introducing Wasefire



Looks like software

Applet

Applet

Wasefire framework

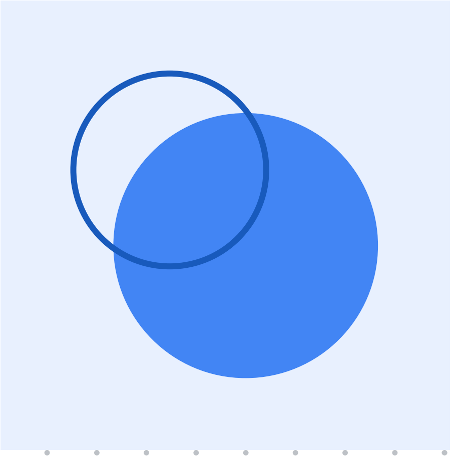
Platform

Makes it secure

Agenda

- Goals and properties
- Architecture and design
- Guided tutorial
- Current state and future work
- Guided exercises

Goals and properties



Secure by default

Hardened platform

Built with side-channel attacks and fault injections in mind

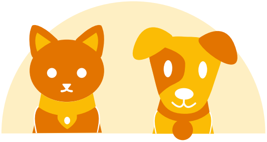
Sandboxed applets

Applets need specific API permissions to interact with the platform (and hardware)

All upgradable

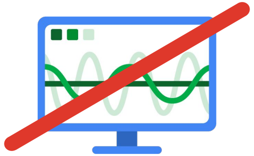
Platform and applets are upgradable by design

For all software engineers



Language agnostic

Developers can use the language, IDE, and workflow they are most comfortable with



Hardware independent

Development can be done on a desktop machine without special hardware

Open source

Hardware vendors can provide proprietary platforms

Why not Global Platform?

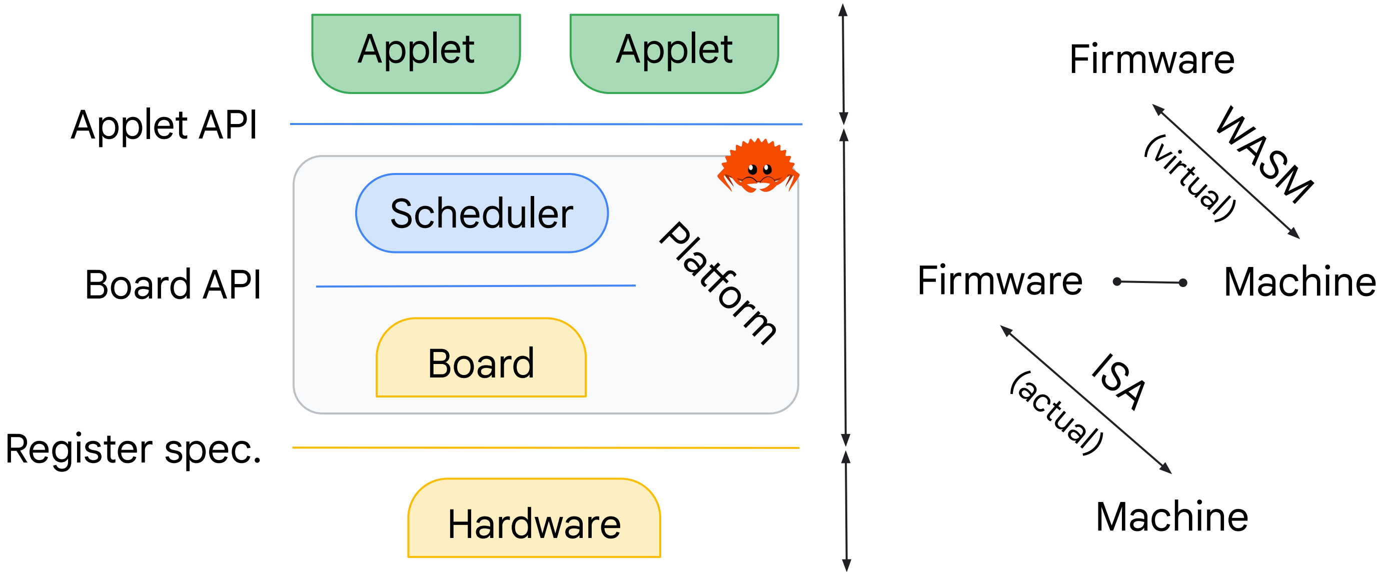
More than NFC and smartcard

More languages to choose from

Execution and validation specified

Architecture and design

WebAssembly virtual machine



Applet and board APIs

Communication

GPIO, LED, buttons, UART, SPI, USB, BLE, ...

Cryptography

TRNG, DRBG, AES, SHA, HMAC, ECC, PQC, ...

Foundational toolkit

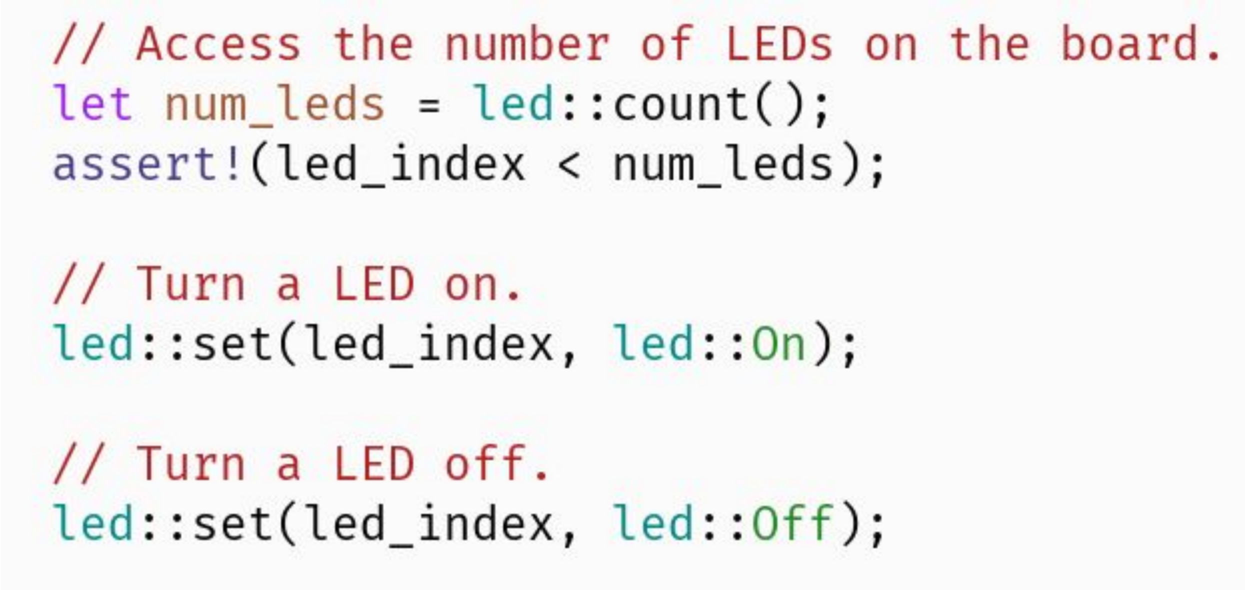
Debug output, perf. measurements, timers, storage, ...

Guided tutorial

https://google.github.io/wasefire/quick/index.html

https://docs.rs/wasefire

How to use LEDs

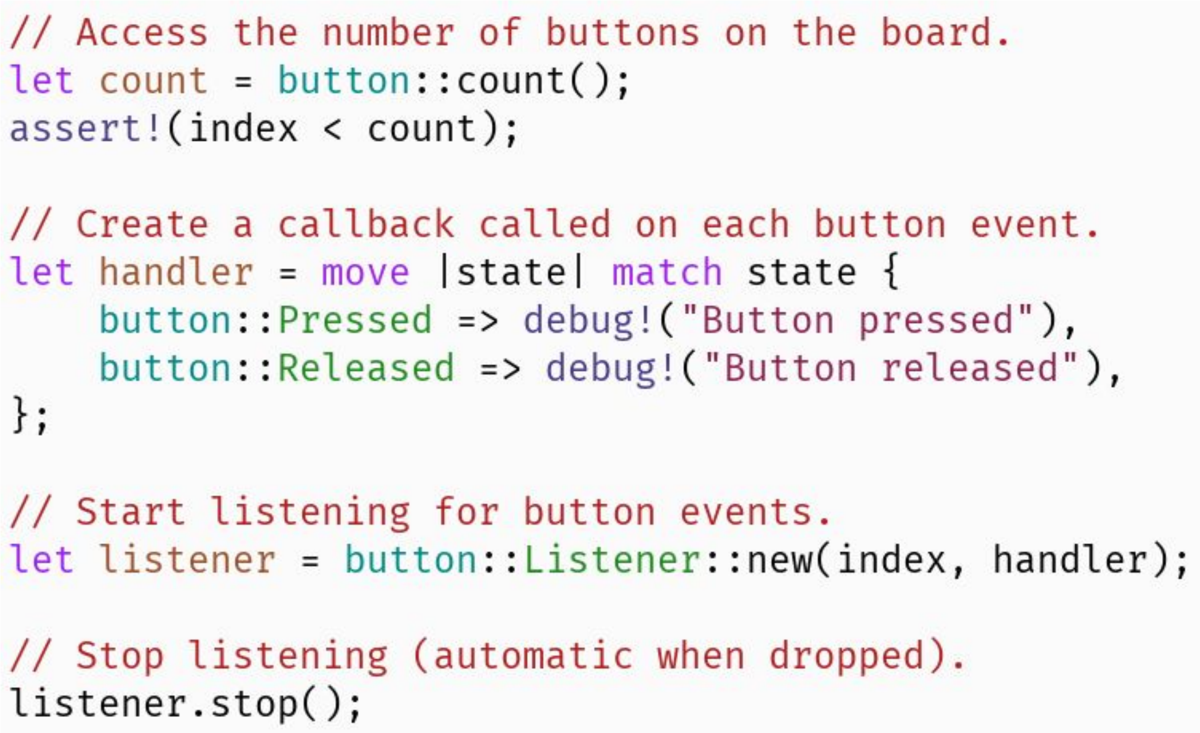


```
// Access the number of LEDs on the board.
let num_leds = led::count();
assert!(led_index < num_leds);

// Turn a LED on.
led::set(led_index, led::On);

// Turn a LED off.
led::set(led_index, led::Off);
```

How to use buttons



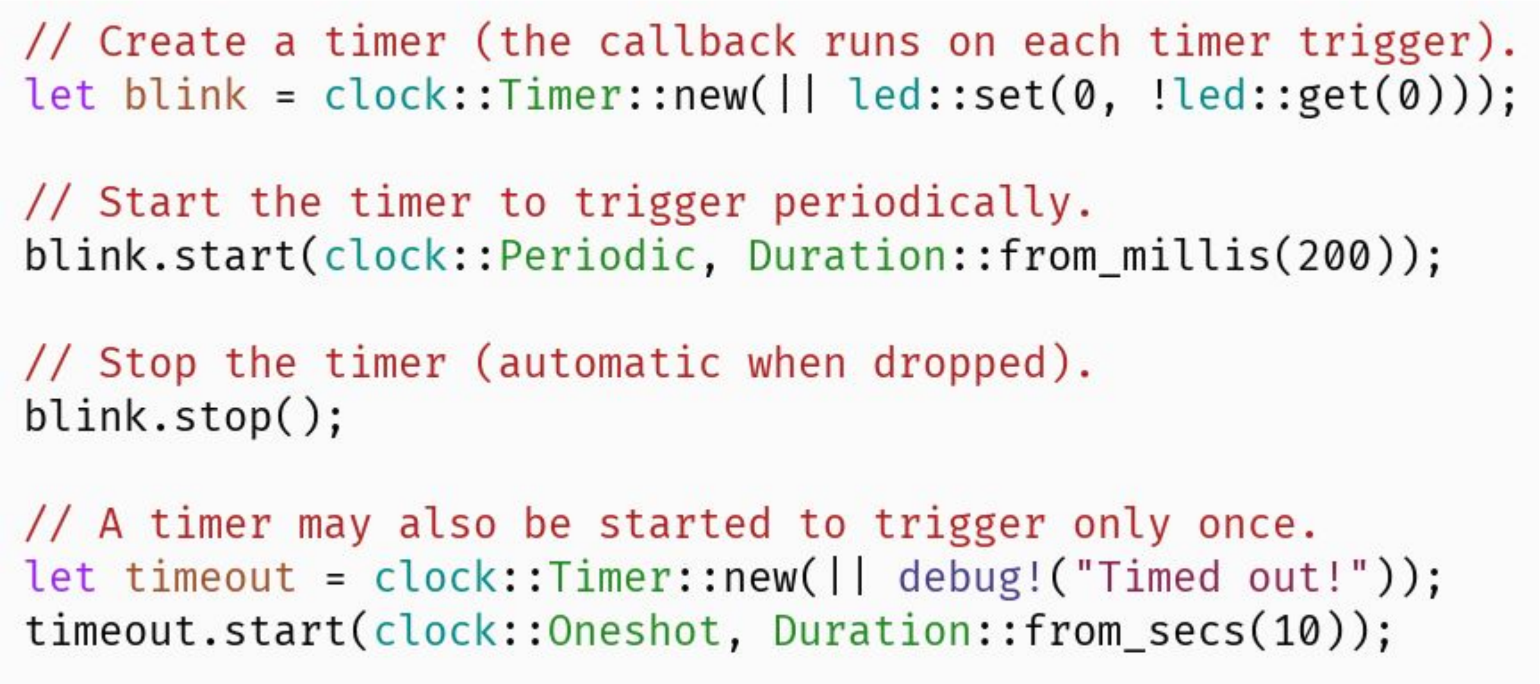
```
// Access the number of buttons on the board.
let count = button::count();
assert!(index < count);

// Create a callback called on each button event.
let handler = move |state| match state {
    button::Pressed => debug!("Button pressed"),
    button::Released => debug!("Button released"),
};

// Start listening for button events.
let listener = button::Listener::new(index, handler);

// Stop listening (automatic when dropped).
listener.stop();
```

How to use timers



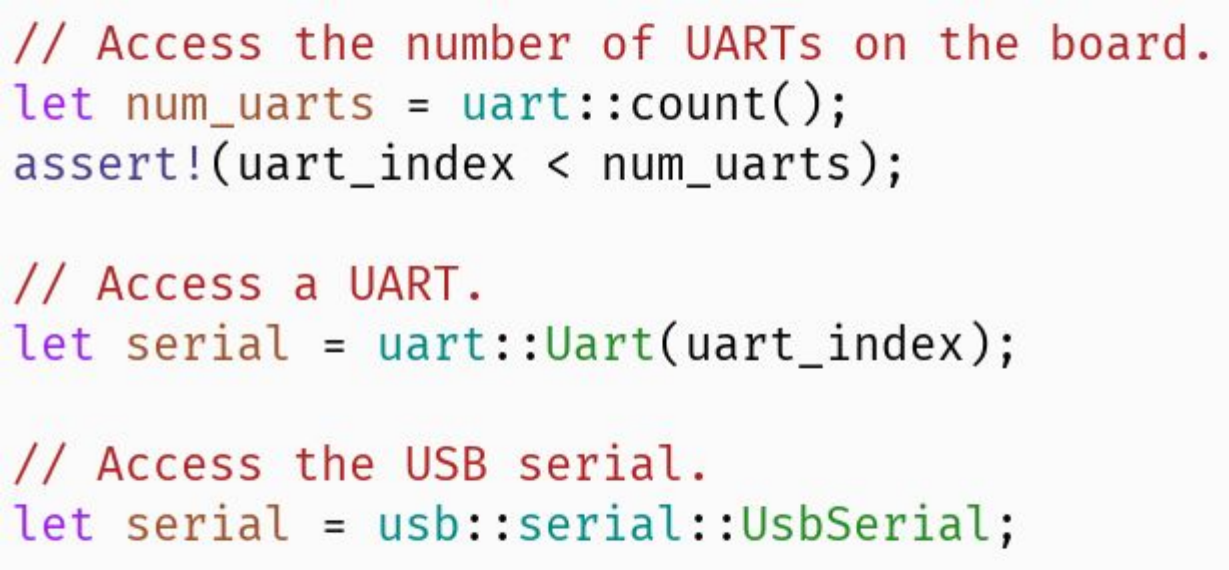
```
// Create a timer (the callback runs on each timer trigger).
let blink = clock::Timer::new(|| led::set(0, !led::get(0)));

// Start the timer to trigger periodically.
blink.start(clock::Periodic, Duration::from_millis(200));

// Stop the timer (automatic when dropped).
blink.stop();

// A timer may also be started to trigger only once.
let timeout = clock::Timer::new(|| debug!("Timed out!"));
timeout.start(clock::Oneshot, Duration::from_secs(10));
```

How to access a serial

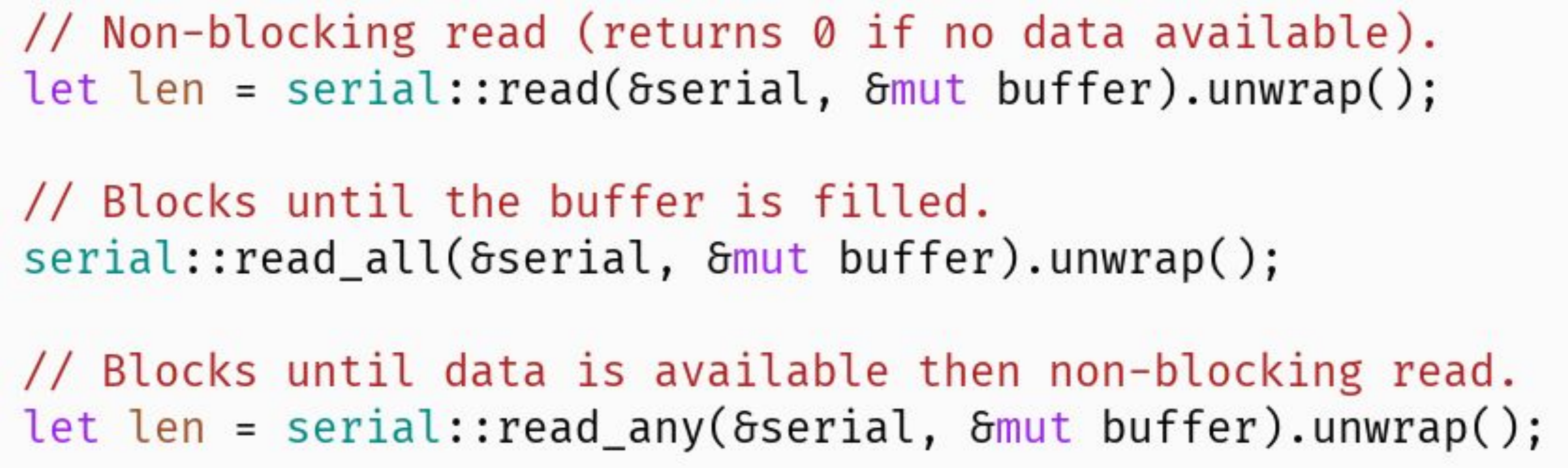


```
// Access the number of UARTs on the board.
let num_uarts = uart::count();
assert!(uart_index < num_uarts);

// Access a UART.
let serial = uart::Uart(uart_index);

// Access the USB serial.
let serial = usb::serial::UsbSerial;
```

How to read from a serial

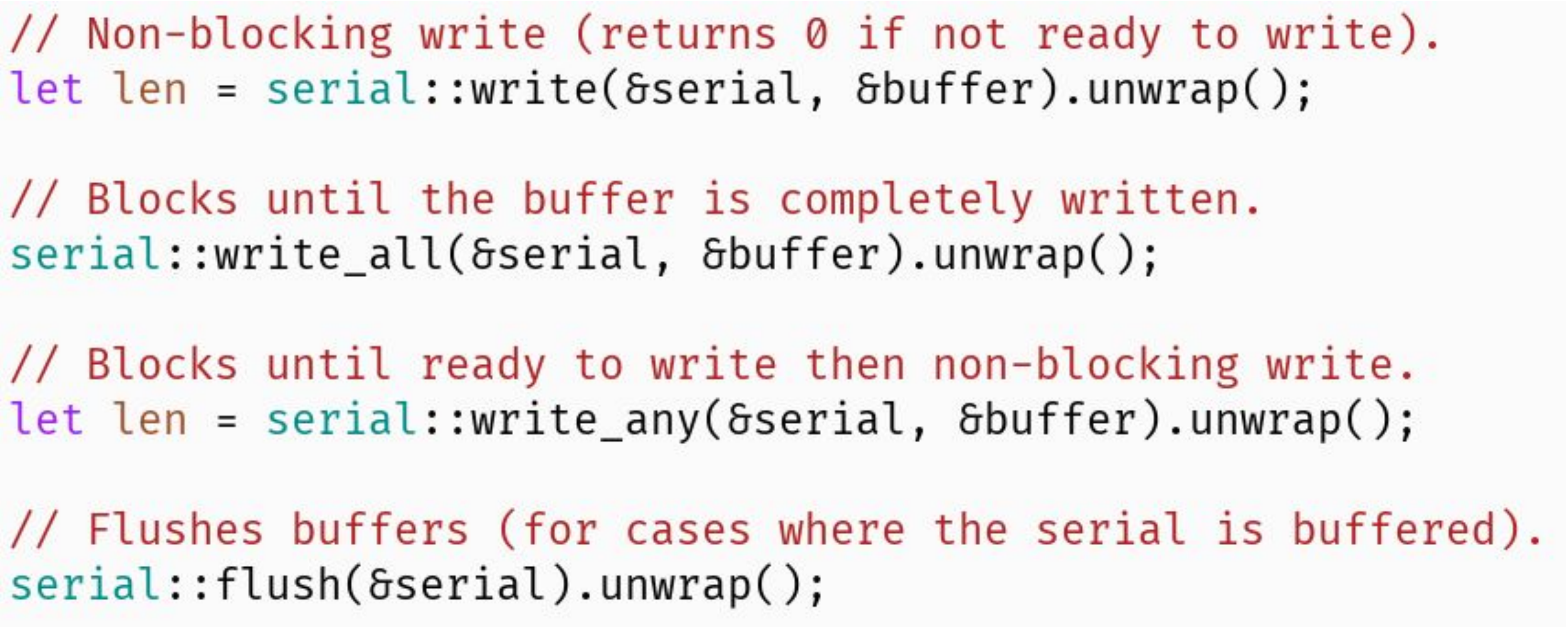


```
// Non-blocking read (returns 0 if no data available).
let len = serial::read(&serial, &mut buffer).unwrap();

// Blocks until the buffer is filled.
serial::read_all(&serial, &mut buffer).unwrap();

// Blocks until data is available then non-blocking read.
let len = serial::read_any(&serial, &mut buffer).unwrap();
```

How to write to a serial



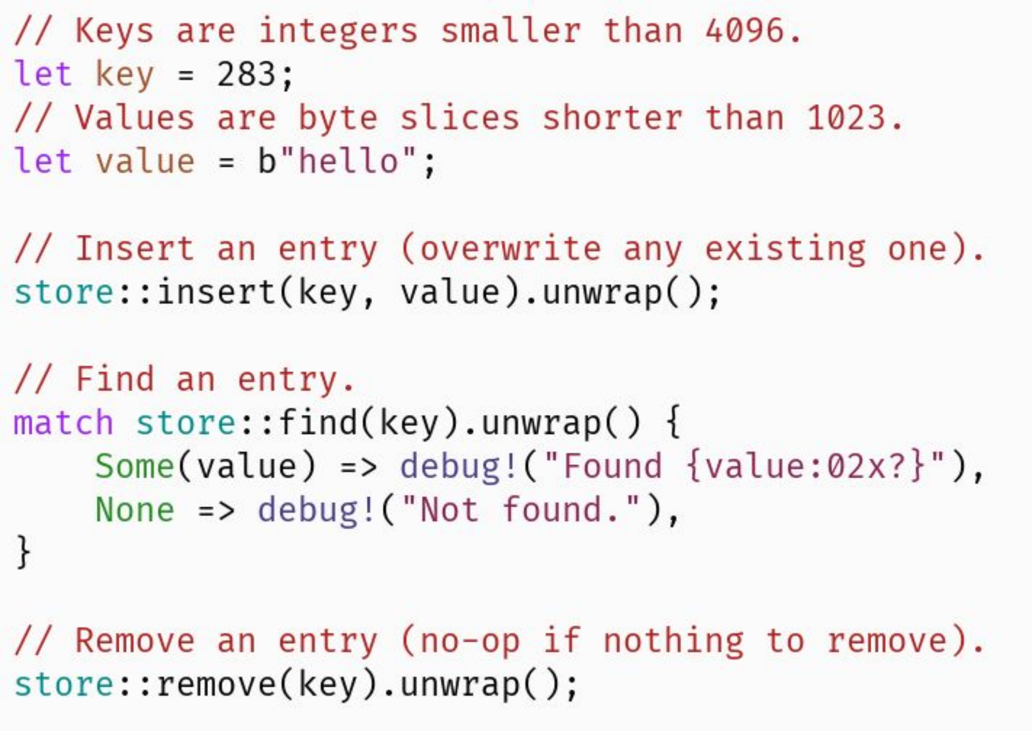
```
// Non-blocking write (returns 0 if not ready to write).
let len = serial::write(&serial, &buffer).unwrap();

// Blocks until the buffer is completely written.
serial::write_all(&serial, &buffer).unwrap();

// Blocks until ready to write then non-blocking write.
let len = serial::write_any(&serial, &buffer).unwrap();

// Flushes buffers (for cases where the serial is buffered).
serial::flush(&serial).unwrap();
```

How to use the persistent store



```
// Keys are integers smaller than 4096.
let key = 283;
// Values are byte slices shorter than 1023.
let value = b"hello";

// Insert an entry (overwrite any existing one).
store::insert(key, value).unwrap();

// Find an entry.
match store::find(key).unwrap() {
    Some(value) => debug!("Found {value:02x?}"),
    None => debug!("Not found."),
}

// Remove an entry (no-op if nothing to remove).
store::remove(key).unwrap();
```

Current state and future work

Where we are

Boards	Nordic nRF52840, RISC-V board to come soon Host with web UI
Applet Languages	Rust and AssemblyScript (low-level API only)
Scheduler	Currently limited to running 1 applet on 1 core Multi-applets and multi-core support to be added
Platform	Upgradability and applet management to be added

Already being used!

Used internally

One project so far and OpenSK will be ported to Wasefire

2 research projects with one university

Improving hardware performance and security

Ongoing research project with 5 universities

Result will be a hardware device powered by Wasefire

Takeaways

- Wasefire is still in its early stages
- Contributions and collaborations are welcome
- Ongoing research projects with academics

Guided exercises

https://google.github.io/wasefire/exercises/index.html

(You'll need a Github account or a Linux laptop)

Thank you!

For more information:

https://google.github.io/wasefire/

Google

Security and Privacy Research