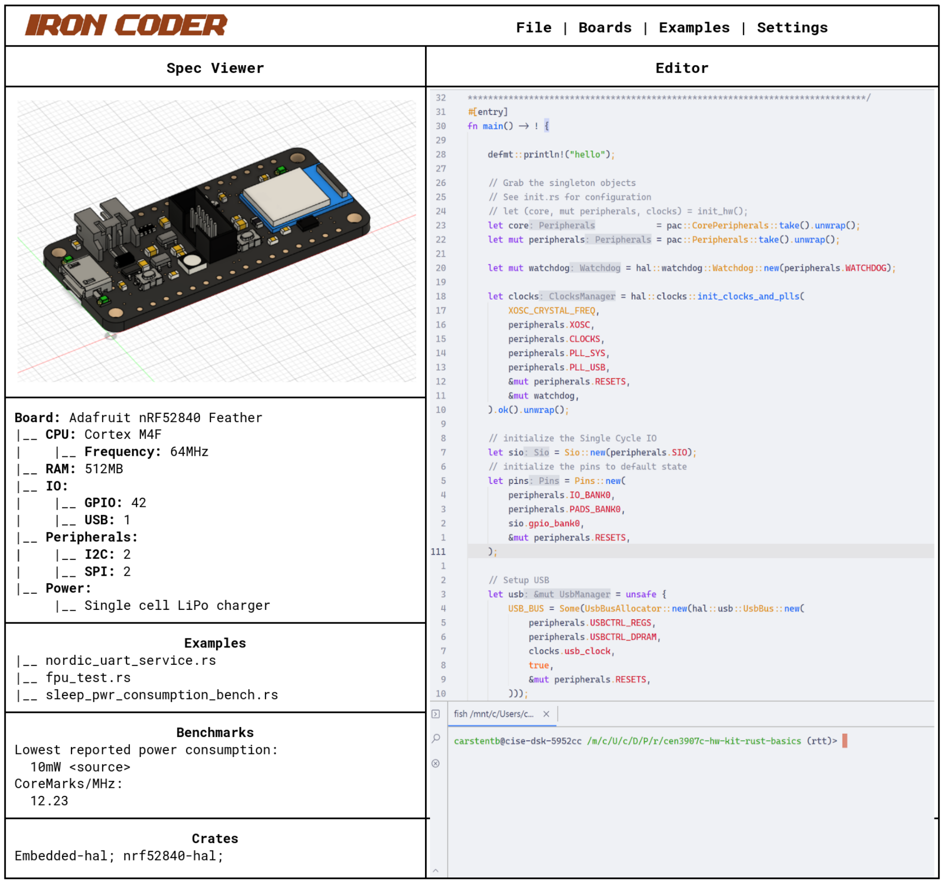
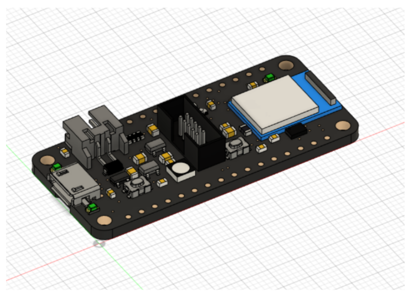
IRON CODER

An EEL5905 project proposal by
Carsten Thue-Bludworth
Advisor: Jeremiah Blanchard

Iron Coder is a gamified IDE for embedded Rust development. The primary goal of the project is to produce an accessible programming environment for engineers to explore the relationships between hardware and software. Additionally, the project aims to provide a community-driven centralized hub for many of the recent innovations in embedded Rust. This includes highlighting and ranking some of the "de facto" standard crates in embedded Rust, providing project templates for various types of development, and accumulating a wealth of example code to work from.

The Arduino programming environment took the maker-space by storm in the early 2000s, providing an accessible hardware and software development platform for embedded devices in the "Arduino" programming language – a wrapper around C++. Iron Coder seeks a parallel goal, but with a focus on Rust. The Rust programming language is rapidly maturing and offers many modern enhancements over older languages like C/C++. While many resources currently exist for learning Rust in the embedded and hosted domains, there is no equivalent to Arduino, which has enabled beginners and experts alike to rapidly and intuitively target specific hardware in an integrated environment.





```
Board: Adafruit nRF52840 Feather
|__ CPU: Cortex M4F
|     |__ Frequency: 64MHz
|__ RAM: 512MB
|__ IO:
|     |__ GPIO: 42
|     |__ USB: 1
|__ Peripherals:
|     |__ I2C: 2
|     |__ SPI: 2
|__ Power:
      |__ Single cell LiPo charger
```

Examples

```
|__ nordic_uart_service.rs
|__ fpu_test.rs
|__ sleep_pwr_consumption_bench.rs
```

Benchmarks

```
Lowest reported power consumption:
  10mW <source>
CoreMarks/MHz:
  12.23
```

Crates

Embedded-hal; nrf52840-hal;

```
*****************************************************************************/
#[entry]
fn main() -> ! {

    defmt::println!("hello");

    // Grab the singleton objects
    // See init.rs for configuration
    // let (core, mut peripherals, clocks) = init_hw();
    let core: Peripherals            = pac::CorePeripherals::take().unwrap();
    let mut peripherals: Peripherals = pac::Peripherals::take().unwrap();

    let mut watchdog: Watchdog = hal::watchdog::Watchdog::new(peripherals.WATCHDOG);

    let clocks: ClocksManager = hal::clocks::init_clocks_and_plls(
        XOSC_CRYSTAL_FREQ,
        peripherals.XOSC,
        peripherals.CLOCKS,
        peripherals.PLL_SYS,
        peripherals.PLL_USB,
        &mut peripherals.RESETS,
        &mut watchdog,
    ).ok().unwrap();

    // initialize the Single Cycle IO
    let sio: Sio = Sio::new(peripherals.SIO);
    // initialize the pins to default state
    let pins: Pins = Pins::new(
        peripherals.IO_BANK0,
        peripherals.PADS_BANK0,
        sio.gpio_bank0,
        &mut peripherals.RESETS,
    );

    // Setup USB
    let usb: &mut UsbManager = unsafe {
        USB_BUS = Some(UsbBusAllocator::new(hal::usb::UsbBus::new(
            peripherals.USBCTRL_REGS,
            peripherals.USBCTRL_DPRAM,
            clocks.usb_clock,
            true,
            &mut peripherals.RESETS,
        )));
```

```
fish /mnt/c/Users/c...
carstentb@cise-dsk-5952cc /m/c/U/c/D/P/r/cen3907c-hw-kit-rust-basics (rtt)>
```

Impact

Iron Coder has the potential to be a powerful tool for teaching and learning embedded development. The platform aims to entice newcomers to programming, with an interactive and entertaining UI, while also serving the needs of more advanced development through its robust, performant, and extensible architecture. Introducing more engineers to Rust will help it grow, mature, and find its way into production. The project will continue development through open-source publishing. The Rust community has many exciting areas of active development, highlighted by the "are we … yet" communities.

Iron Coder has the potential to provide a hub for code and community around some of these areas in the context of embedded development.

Statement of Work

The initial major features of ***Iron Coder*** will include:

- ★ Built-in text editor and terminal for writing, building, and debugging Rust firmware.
- ★ A hardware database that links target hardware to de-facto standard Rust crates (e.g. *embedded-hal*, *cortex-m-rt*, *rp2040-hal*, etc) and example code. The IDE should easily allow for the addition of new target hardware platforms by the community.
    - ➢ Initial development will limit the scope to a small selection of main and peripheral boards in the Adafruit Feather Ecosystem, a modular, diverse set of development boards.
- ★ A spec viewer that shows properties of the current target hardware: A 3D model of the hardware setup, a description of its specifications (processor type, memory layout, on-chip peripherals, etc), a list of resources for working with it (datasheets, related crates, example code, etc), and benchmarks run that measure various performance characteristics of the hardware.

Prior Art & Inspiration

Iron Coder will have a balance of modern and retro technical aesthetics, acknowledging the history and evolution of computer technology. Elements of the interface will be gamified to provide the coder with an interactive and exciting experience. In the future, the program will include "achievements" and "missions" related to educational coding objectives as well as contributions of new hardware, examples, or benchmarks to the database. Existing platforms such as Github incorporate this gamified social aspect. Monospaced fonts and pixel art will be used where appropriate to remind the coder of their connection to history and a bit-based world, whereas quality rendering of 3D models, charts (related to benchmarking), and editor functionality (code folding, syntax highlighting, etc) will provide a modern and robust coding experience.

Another primary inspiration for the project is CircuitPython, an embedded development ecosystem that provides an easy-to-use Python interpreter targeting a specific set of development boards. Oriented for beginning coders, the ecosystem prioritizes ease-of-use, accessible documentation, and a smooth development workflow, but it does not emphasize performance analysis, nor does it have an integrated editor (instead encouraging use of the Mu Editor).

The backend of Iron Coder will focus on cross-platform stability and performance on modest hardware. As a Rust-oriented IDE, ***Iron Coder*** will be built with Rust, utilizing growing tools in the ecosystem such as the Bevy game engine and the Lapce code editor.

Learning Objectives

★ Develop a detailed understanding of the Rust programming language in the context of bare-metal target hardware (for code written *using* Iron Coder) and general-purpose Rust (for the Iron Coder project *itself*).

★ Build software engineering skills through the construction of an open-source, cross-platform, performant, and professional application available for public use.

★ Critically analyze the multi-faceted performance metrics of embedded systems; initial code examples for Iron Coder will target a small set of boards with varying specifications, and the performance measurements of these systems will be evaluated.

★ Gain an understanding of contemporary research regarding the relationship between programming environments, gamification, and learning, and incorporate current trends into the design of Iron Coder.

Timeline and Deliverables

05/15/2023 - 05/21/2023 (1 week)	Develop project structure, including library/framework decisions and major aesthetic and technical design decisions; setup project repository and non-code document templates. **(D) -> Design Plan and initial repo**
05/22/2023 - 06/11/2023 (3 weeks)	Begin implementation towards prototype build, focusing on major features including code editor, spec viewer, and hardware/code database. **(D) -> Prototype build** (sets foundation for all major features, gets test suite up and running)
06/12/2023 - 06/25/2023 (2 weeks)	Work towards the first vertical slice, focusing on editor functionality (tabs, split screen views, syntax highlighting/folding, opening/saving code **(D) -> Editor vertical slice**
06/26/2023 - 07/16/2023 (3 weeks)	Work towards the second vertical slice, focusing on functionality of the hardware database and spec viewer. **(D) -> Spec Viewer vertical slice**
07/17/2023 - 08/11/2023 (3.5 weeks)	Integration of prior work, bug fixes, art enhancements, and writing of final report. **(D) -> Beta version of Iron Coder, buildable from source for Windows, Mac, and Linux; x86 and aarch64 architectures.** **(D) -> Final report in IEEE format outlining the system architecture, current state, future enhancements, and impact on Rust's embedded development ecosystem.**