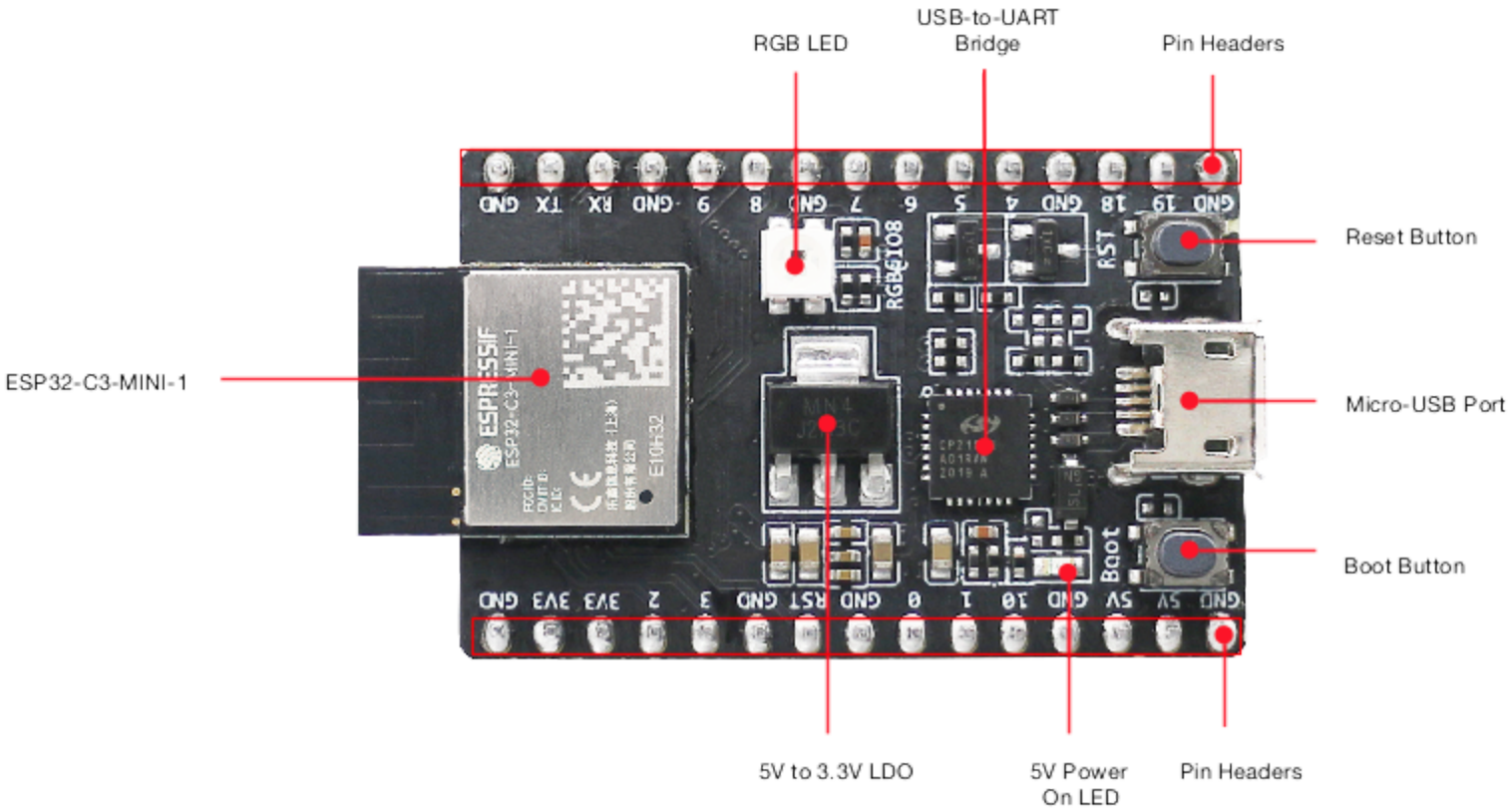
esp-rs Introduction

Scott Mabin

Goals of this session

- Understand how esp-idf is used in conjunction Rust
- Tooling available
- Flash and run Standard library example on C3

About the hardware



STD - esp-idf

- A development framework for all Espressif chips since 2016 (esp32 and newer) written in C.
- Handles atomic emulation for targets (esp32c3 included) without atomics.
- esp-idf tooling is mostly written in Python, but `idf-env`, written in Rust, is rapidly gaining features.

STD - Rust

- esp std port is `unix` like.
- Makes use of libc through the newlib component of esp-idf.
- Reference PR - rust#87666
- Supports:
  - Threads
  - Networking
  - Storage (File System)
  - Rng
  - Syncronization primitives (Mutex, RWLock)

STD - Build system

- `embuild` - manages the build and configuration of the esp-idf project.
- `-Z build-std` - Using cargo to build the standard library.
- `ldproxy` - used to to collect all the linker args from esp-idf and use them in the final link with Rust.
- **Note**: Does not play well with Windows currently, a few issues with long paths / long command line args.

Tooling - espflash

- Rewrite of esptool.py with convenient Cargo integrations.
- Communication with the ROM bootloader via serial to flash programs.
- Handles all sorts of esp-idf specifics, partition table, bootloader etc.

Tooling - probe-rs

- RISC-V & ARM only.
- USB-SERIAL-JTAG peripheral built into the esp32c3 silicon supported in probe-rs v0.14.
- Flashing works flawlessly, but probe-rs is not "esp-idf aware".
- Debugging RISC-V chips with probe-rs is mostly there, but unwinding the stack is buggy. See this tracking issue.

Example

- Lets get the built-in neopixel flashing

Example - Prequisites

- `rustup toolchain install nightly` & `rustup update nightly`.
- `cargo install ldproxy --force`
- `cargo install cargo-espflash`
- `git clone https://github.com/MabezDev/esp32c3-idf-led-example`.

Example - Build

Inside `esp32c3-idf-led-example`, run:

```
cargo +nightly espflash --monitor $DEVICE_PATH
```

Where $DEVICE_PATH is the path to the serial port for the esp32c3.

Examples:

- Linux: `/dev/ttyUSB0`
- MacOS: `/dev/cu.usbserial-A700dYoR` or `/dev/ttyUSB0`
- Windows: `COM1`

Bonus - Xtensa

- Prior to the C3 all Espressif chips are based on the Xtensa architecture.
- ISA is not public, only licensees have access.
  - We made a unofficial ISA doc available here
- LLVM backend developed by Espressif
- Rust for esp fork utilises the Xtensa LLVM backend

Links

- esp-idf
- The esp book
- esp-rs organisation
- esp-rs roadmap