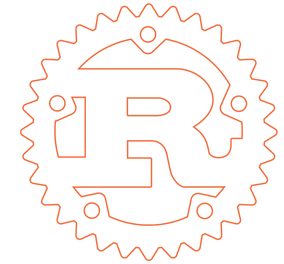
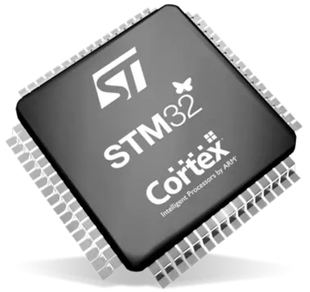


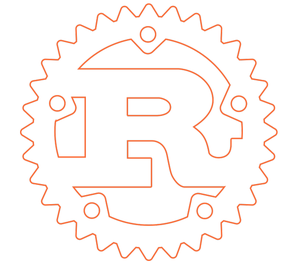
# Начинаем Embedded на Rust

## RustCon Russia, 2021

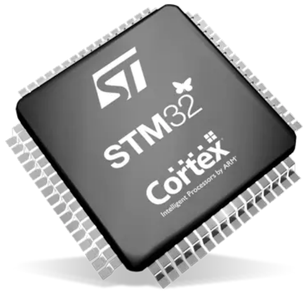
Anton Patrushev, 03/12/2021



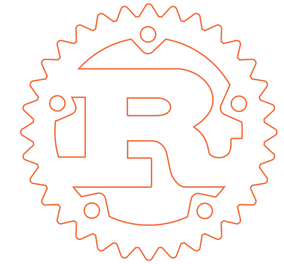
# О себе



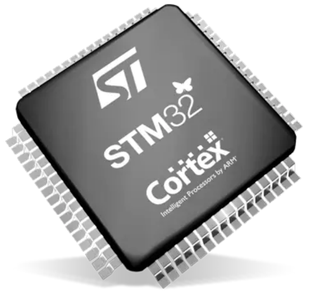
- Работал в Naumen и OOH
- CTO@Spherical
- 20 лет Python, 5+ лет C++/98
- PyCon Russia, RustCon Russia, co-founder
- Пишу на Rust с 2016 года. Неосновной язык.
  - немного production вокруг расширения Python
  - для себя - под микроконтроллеры



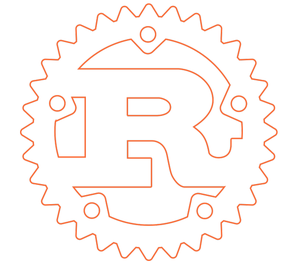
# О вас



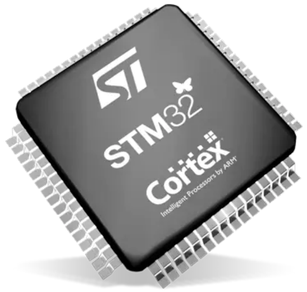
- Вы умеете пользоваться браузером и гитом!
- Вы умеете читать и писать код!
- Пишете на Rust?
- Пишете под микроконтроллеры?
- Пишете под микроконтроллеры на Rust?



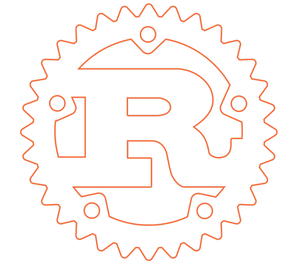
# Что здесь будет происходить



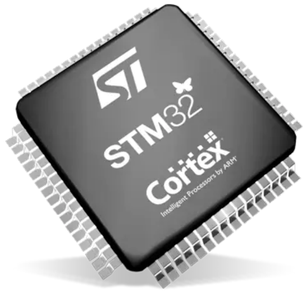
- Буду перемежать активности и повествование
- Расскажу про Rust и микроконтроллеры
- Напишем простую мигалку (очень быстро!)
- Сделаем USB устройство



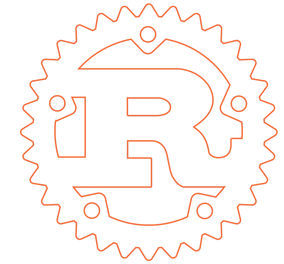
# Договоримся о правилах



- Это не доклад!
- Вопросы и замечания сразу, только поднимите руку
- Я **постараюсь** ответить (возможно кто-то вместо меня)
- Приготовьтесь работать
  - времени мало
  - материала много
  - самое интересное в конце



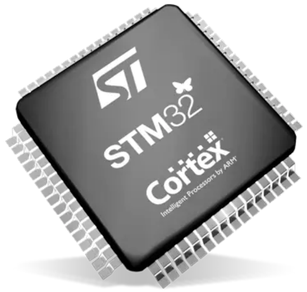
# Канальчик

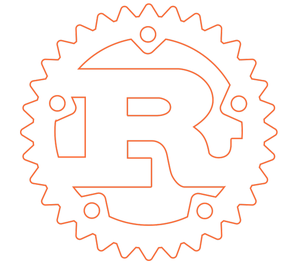


В процессе мастеркласса я буду публиковать ссылки в телеграмм чатике. Там же можно и поговорить. Во время и после.

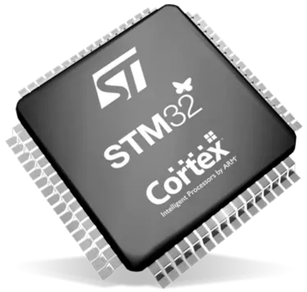
https://t.me/
rustcon_ru_21_embedd

@RustCon Russia 2021/Embedded

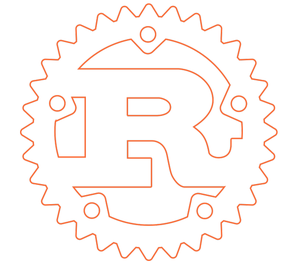




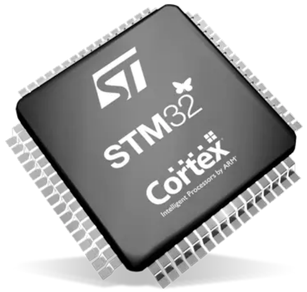
# Поехали!



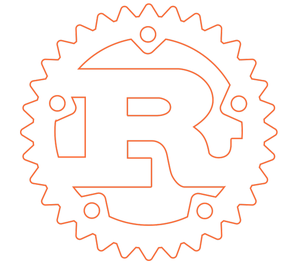
# Tooling



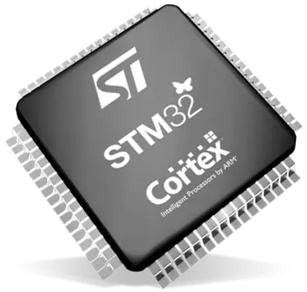
- Rust известен своими крутыми инструментами разработчика
- Самые главные
  - **rustup**, установщик
  - **cargo**, швейцарский нож
  - **crates.io**, реестр пакетов
  - **docs.rs**, документация пакетов



# Установим Rust

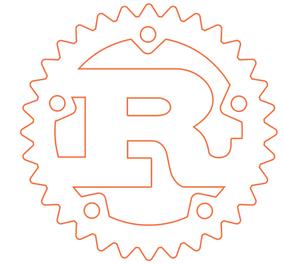


- https://rustup.rs/
  - устанавливает
    - rust
    - toolchain
  - позволяет выбирать активный toolchain
- из минусов - curlbash

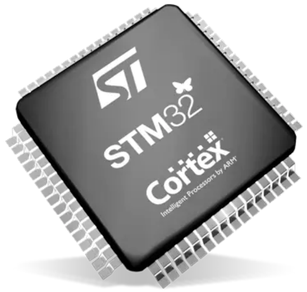


# Что мы там такое ставим

краткая справка из Википедии

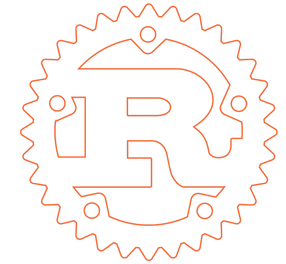


- 2006: личный проект Graydon Hoare, сотрудника Mozilla
- 2009-2010: Mozilla спонсирует проект, объявляет о нём публично
- 2010-2011: LLVM, self-hosting
- 2015: Version 1.0
- 2016-2018: Создание Rust Embedded WG
- 2020: Mozilla распускает Servo Team и большую часть Rust Team
- 2021: Rust Foundation (AWS, Huawei, Google, Microsoft, и Mozilla)

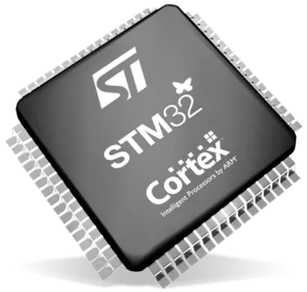


# Что мы там такое ставим

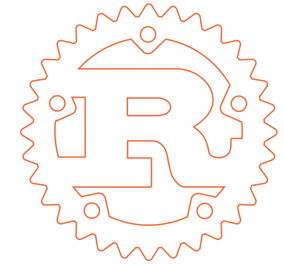
## пару слов о языке (не судите строго)



- Компилируемый
- Memory safety, RAII, optional ref-counting, stack allocation
- Ownership/Borrowing/Lifetimes
- Types
  - strong, static
  - traits (type classes), structs (user defined types)
  - inference, generics, inference
- Zero-cost abstractions

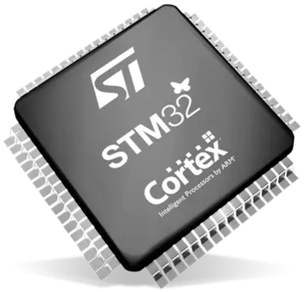


# Репа

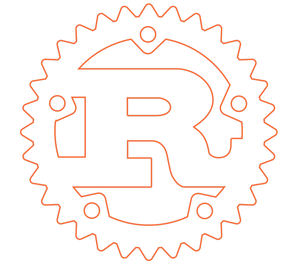


- все изменения я буду публиковать в репу на гитхабе
- она изначально пустая и будет наполняться последовательно

- https://github.com/apatrushev/rcre-2021



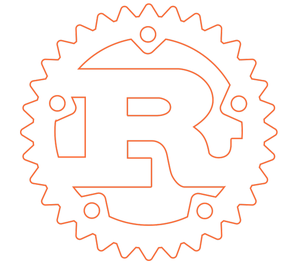
# Запустим уже что-нибудь



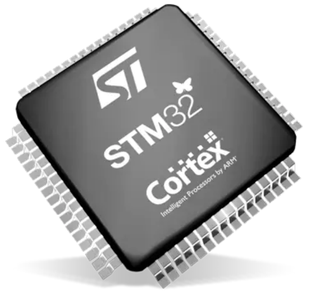
- создаём проект
  - cargo new workspace/rcre
- запускаем его
  - cargo run

- код уже в мастере!
  - ну или вот прям сейчас там окажется

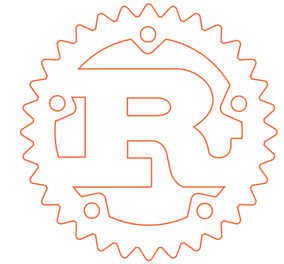
# Что там оно нагенерило



- Cargo.toml
  - это такой типа Makefile, setup.py, package.json
  - там будут всякие инструкции для cargo как обращаться с проектом
- src/main.rs
  - текст нашего хеллоу волда



# Добавим поддержку подопытного

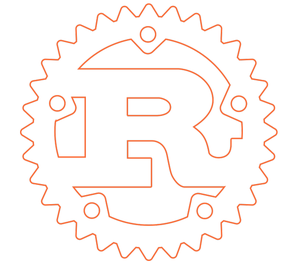


- установим поддержку целевой архитектуры для компилятора (для других досок - воспользуйтесь справочником)
- установим вспомогательные инструменты для прошивки и управления проектом

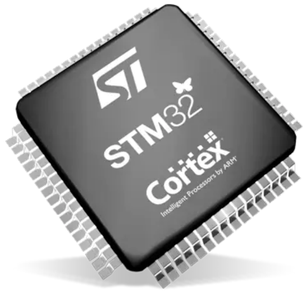
- https://github.com/apatrushev/rcre-2021/blob/main/docs/step_02.md

# Подопытный

пока оно ставится

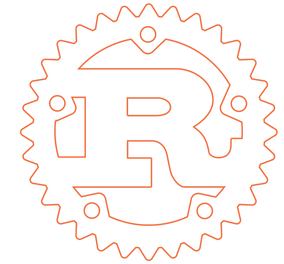


- STM32
  - семейство микроконтроллеров от L0/G0 до F7/H7
  - ARM Cortex-M, 32bit
  - отличная совместимость
  - суперская документация

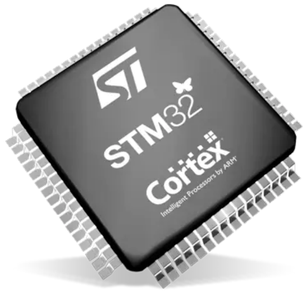


# Подопытный

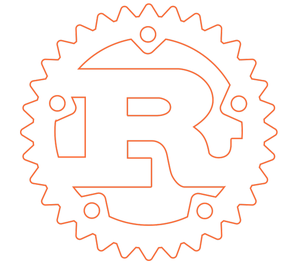
пока оно ставится



- STM32F3DISCOVERY
  - отладочная доска с STM32F303 (Cortex-M4F)
  - куча периферии прямо в камне
    - Timers, DMA
    - 80+ GPIO, ADC/DAC, CAN, I2C, UART, SPI, USB
  - немножко периферии на доске (светодиодики, гироскоп)
  - встроенный отладчик

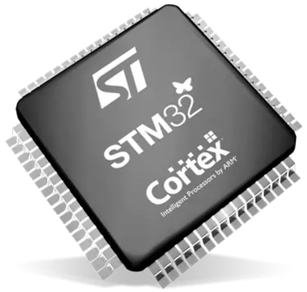


# Компилируем под target



- пробуем собрать под нашу целевую архитектуру

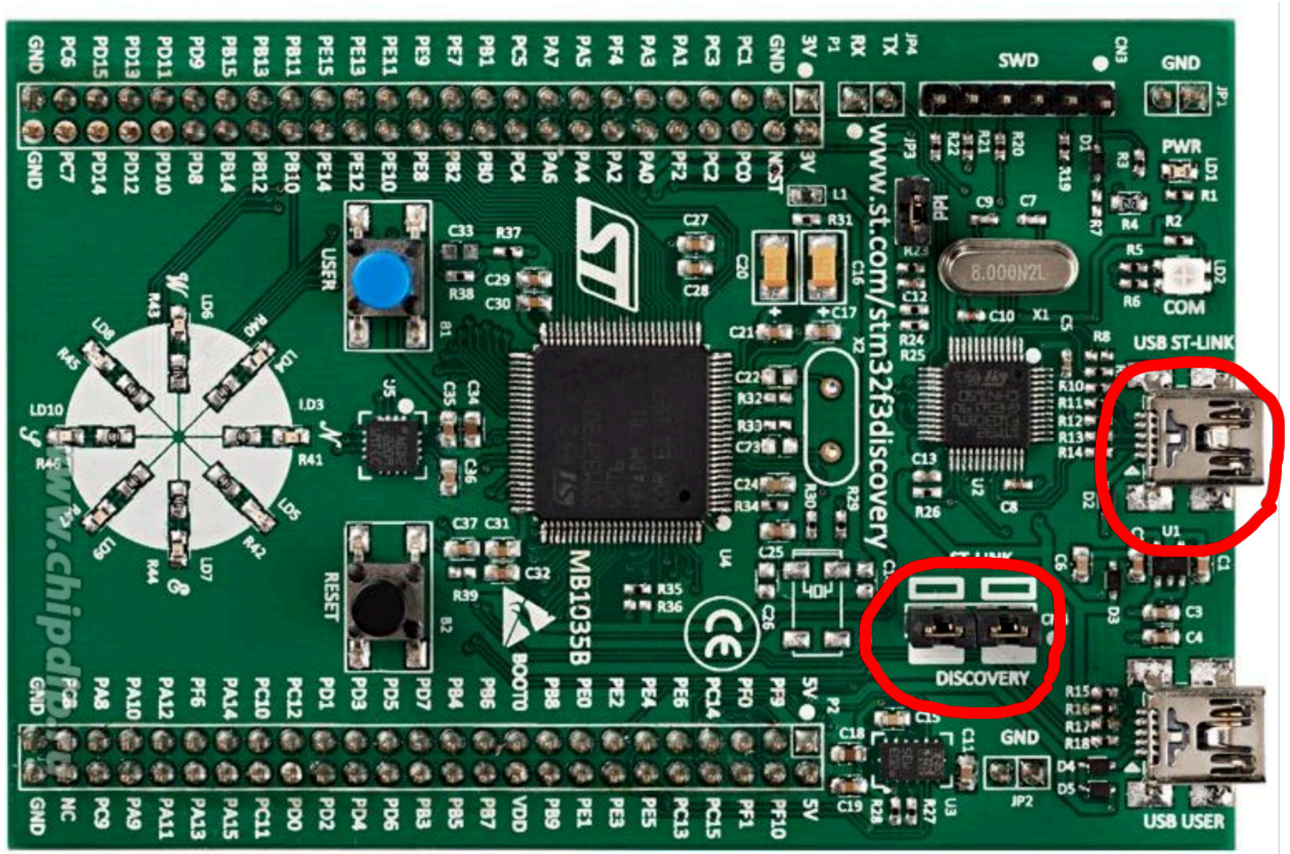
- https://github.com/apatrushev/rcre-2021/blob/main/docs/step_03.md



# Бинго!

на самом деле ещё нет

# Прошивальщик на борде

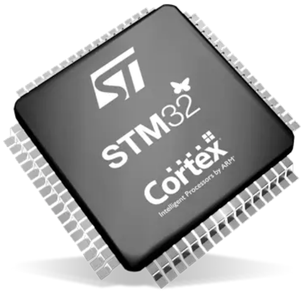


# Научимся шить

## и вязать, но в другой раз

- скомпилируем уже под конкретную железку какой-то код
- и запишем наконец-то немного своих байтиков в неё
- правда наш код будет всё ещё делать ни-че-го

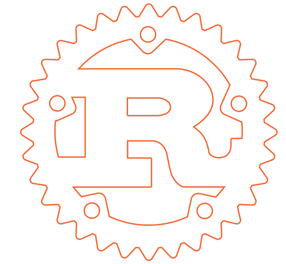
- https://github.com/apatrushev/rcre-2021/blob/main/docs/step_04.md



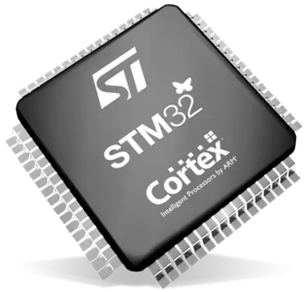
# Моргалка?

сорри, но нет, я знаю способ поморгать в 4 команды,
но решил рассказать о нём в самом самом конце

# Экосистема rust-embedded



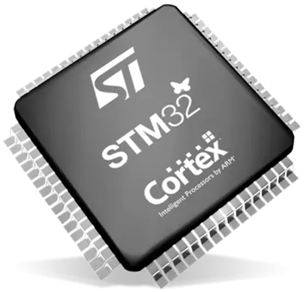
- architecture support crates (условное название)
  - cortex-m/cortex-m-rt (cortex-a, cortex-r)
  - risc-v[-rt]
  - xtensa-lx[-rt] (ESPшки)
  - avr-device (ардуинки)
- peripheral access crates (aka pac)
  - stm32f3 в нашем случае
  - по сути - для каждого процессора
- hardware abstraction layer (aka hal)
  - stm32f3xx-hal
- board support crates (aka bsp)
  - stm32f3-discovery



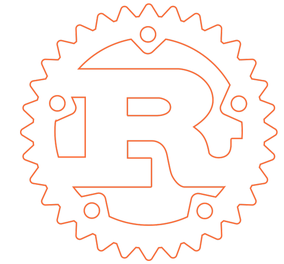
# Экосистема rust-embedded

## чтоб два раза не вставать

- embedded-hal
  - linux-embedded-hal (удобная штука для разработки и отладки драйверов)
- driver crates
  - lsm303dlhc (как пример, который стоит на нашей доске)
- rtic
  - к этому зверю мы ещё вернёмся позже более подробно
- tooling
  - cargo-embed|cargo-flash
  - probe-rs/vscode
  - probe-run

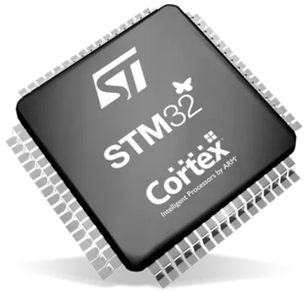


# Давай уже моргнём, сколько можно!



- моргнём, непременно моргнём! весь мир в труху! но потом! (c)
- шутка, и правда пора
- и с QR кодами пора заканчивать

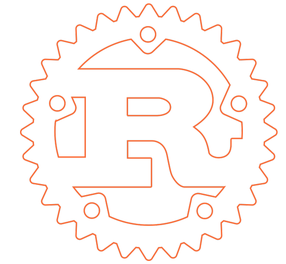
- https://github.com/apatrushev/rcre-2021/blob/main/docs/step_05.md



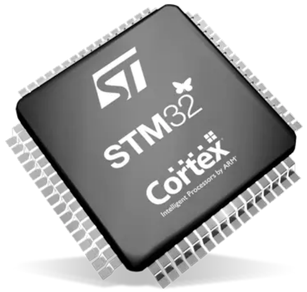
# Наконец-то!

не прошло и часа (подсчёты на момент подготовки)

# Что можно отметить

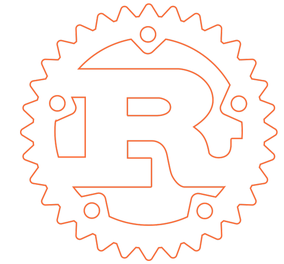


- led::Leds/compass::Compass и подобные абстракции
  - позволяют описать железо на конкретной плате
  - избавляют погромиста от ошибок с распиновкой
- абстракции hal карты (stm32f3xx_hal)
  - описывают железо конкретного процессора
  - помогают не ошибиться с ним
  - помогают оперировать приятными буковками, вместо циферок



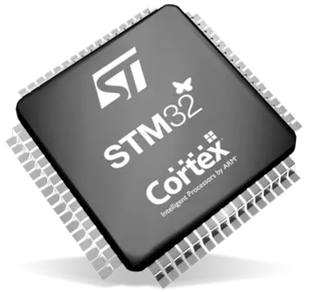
# Давайте всё-таки в 4 команды

## не дотерпел до конца



- curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
- rustup target add thumbv7em-none-eabihf
- cargo install cargo-flash
- git clone https://github.com/rubberduck203/stm32f3-discovery.git
- cd stm32f3-discovery
- cargo flash --example blinky --chip STM32F303VCTx

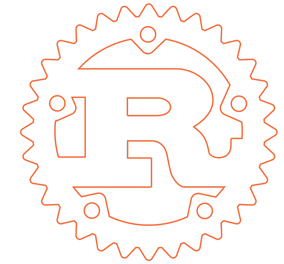
- это важный способ изучать всякие экзамплы
- не надо переписывать код - сейчас он будет в гитхабе (угадайте файл?)



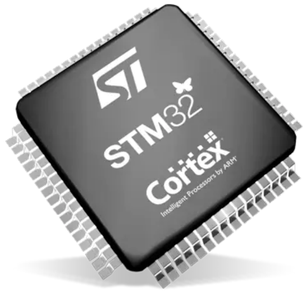
# Прерывания

куда ж без них-то?

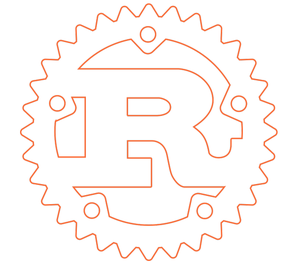
# Минусы? Полно!



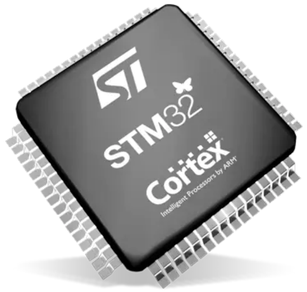
- плохо контролируемый глобальный стейт
- потенциальные data races
- инверсия приоритетов!



# А выход? RTIC!



- RTFM (http://www.rtfm-lang.org/)
- RTFMv2 (https://blog.japaric.io/rtfm-v2/)
- RTIC (https://rtic.rs/0.5/book/en/)
  - tasks
  - приоритеты
  - безопасный шаринг без data races
  - гарантия отсутствия deadlock во время компиляции (!!!)
  - минимальный оверхед, нет зависимости на аллокатор
  - полная поддержка Cortex-M



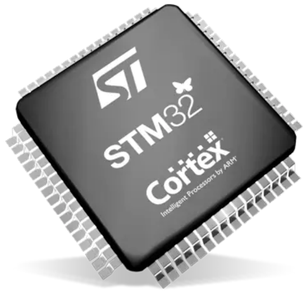
# RTIC

# Добавим немного дебага

опять слайд с куаркодом, не смог себя побороть

- probe-run =>
- сложность с режимом WFI
- RTT
  - многоканальная штука
  - cargo-embed

- https://ferrous-systems.com/blog/probe-run/



# USB

прибарахлились тулзами (RTIC, RTT) - пора

# The End.
# Вопросы?

http://patrushev.me