tourmaline
core

Как мы Grab'нули проблем при embedded разработке

16 апреля 2024

Кузьмичева Ольга

Разработчик встраиваемых систем

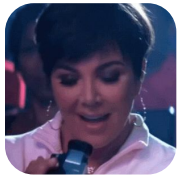
Кирпичников Антон

Разработчик встраиваемых систем

Содержание

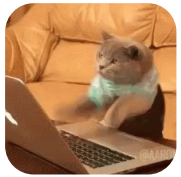
- Стандарт обеспечения подлинности данных
- Как мы писали драйвер крипточипа для видеокамеры
- На какие грабли мы наступили
- Наш подход к юнит-тестированию кода устройств

О проекте



Устройство

Видеокамера, фиксирующая подлинность произведенного контента

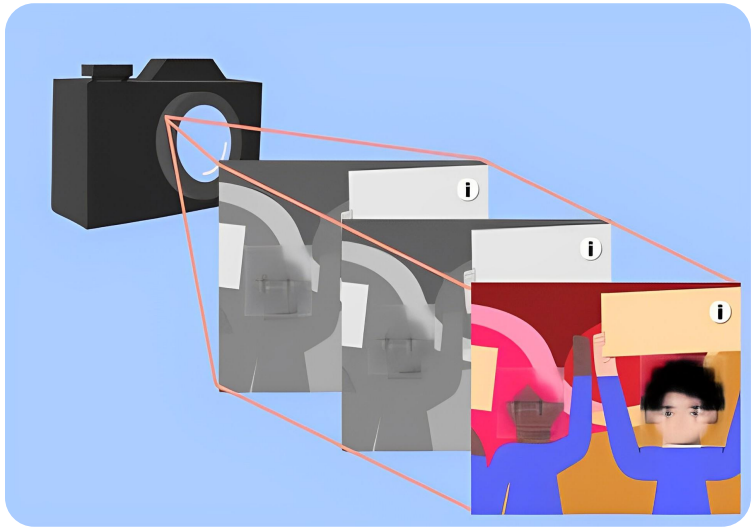


Мы занимались

Написанием драйвера для чипа безопасности, обеспечивающего подлинность контента

Подлинность контента

Важно доказать, что события, которые происходили у меня на глазах, были именно такими, какими я их видел.



C2PA

Стандарт, использующий метаданные и криптографические методы, для предоставления проверяемой информации о происхождении, истории и модификациях части контента.

https://c2pa.org

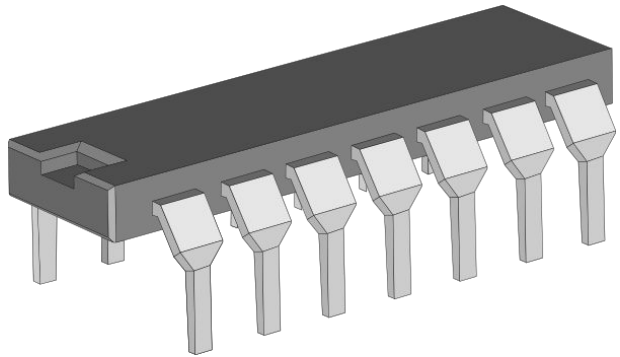
C2PA Membership

Security controller

Цель: Подпись контента

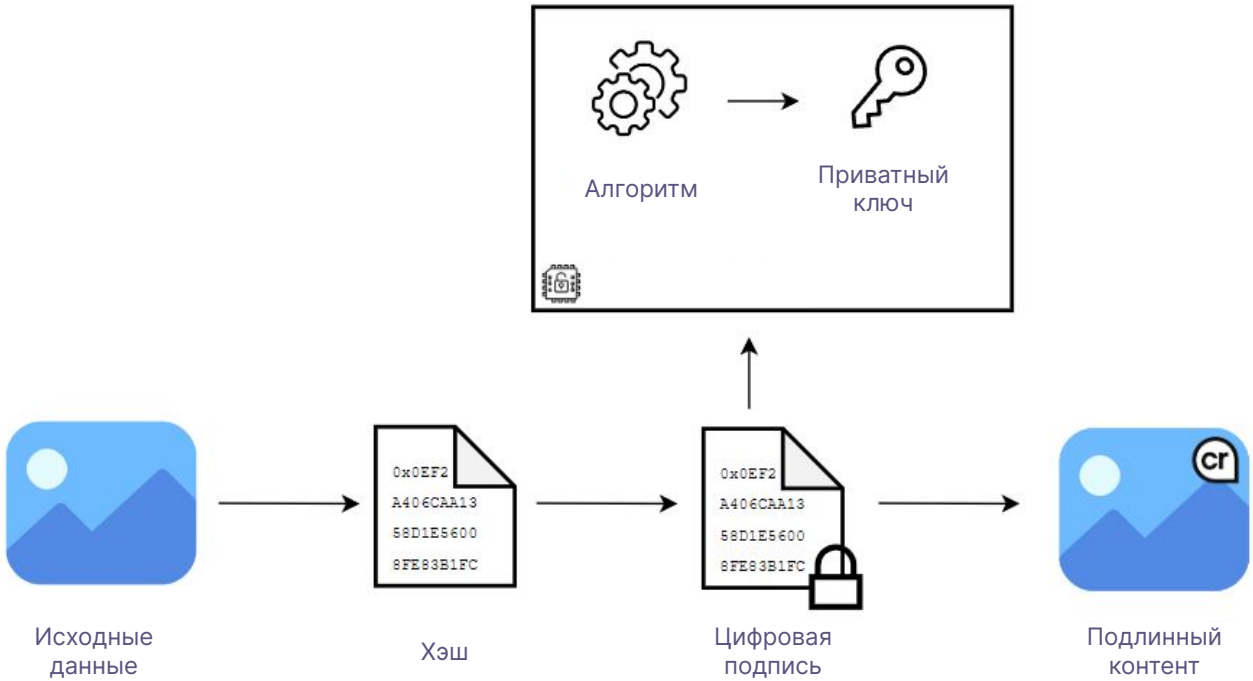
Описание

- Надежное хранение приватного ключа
- Поддержка различных криптоалгоритмов (ECC, RSA, AES, TLS)



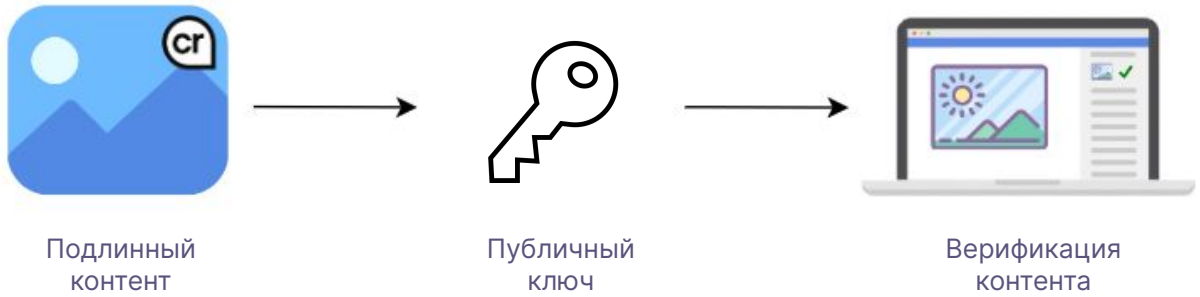
Как это работает

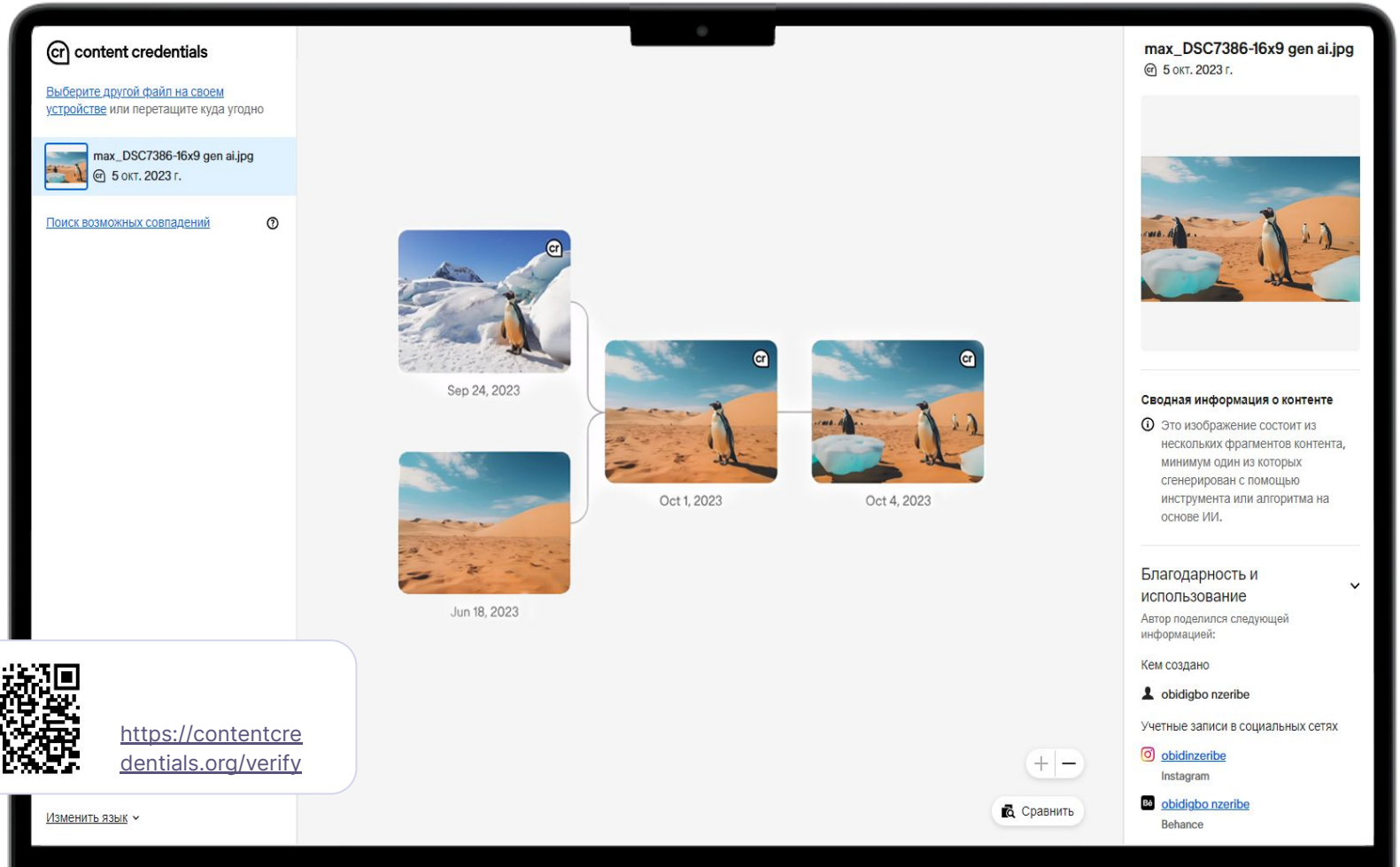
Создание подлинного контента



Как это работает

Проверка подлинного контента





https://contentcredentials.org/verify

Как была построена работа

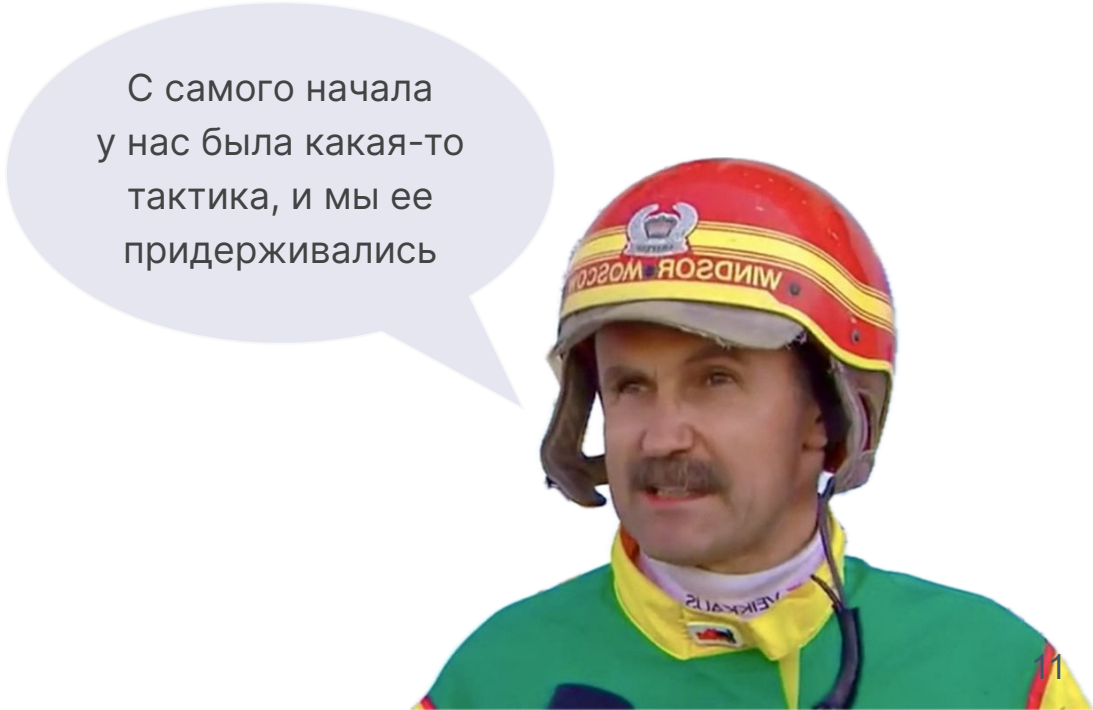
1) Изучение документации

2) Разработка драйвера

3) Внедрение в устройство

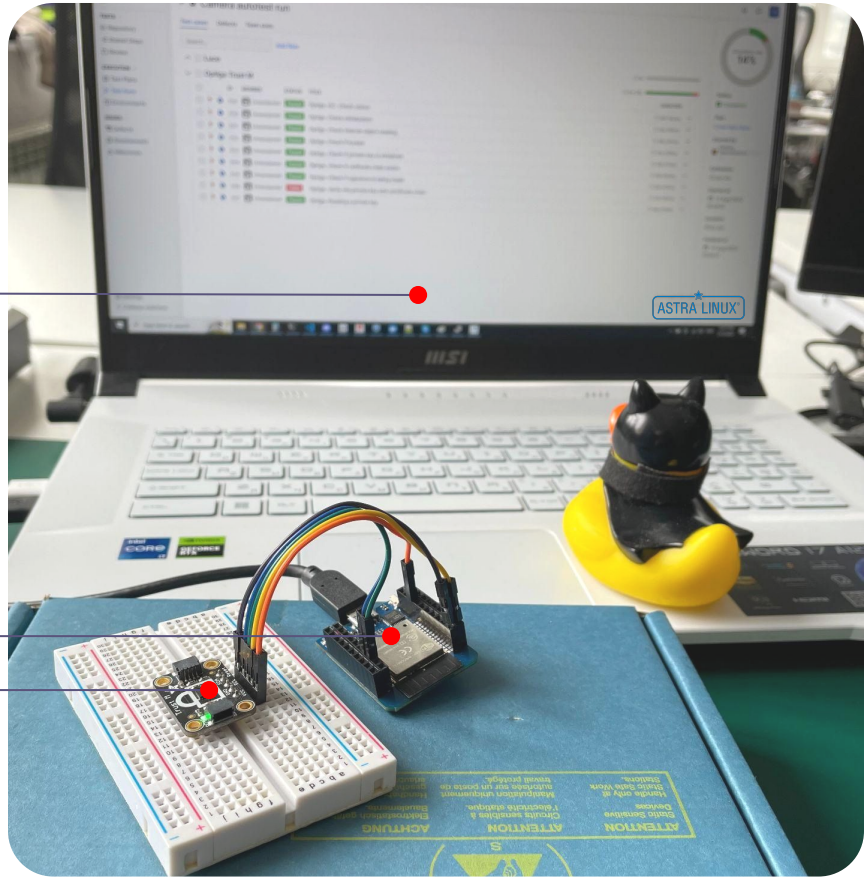
4) Полноценное тестирование

5) Исправление багов



Тестовый стенд

- ПК
- Контроллер ESP32
- Security chip

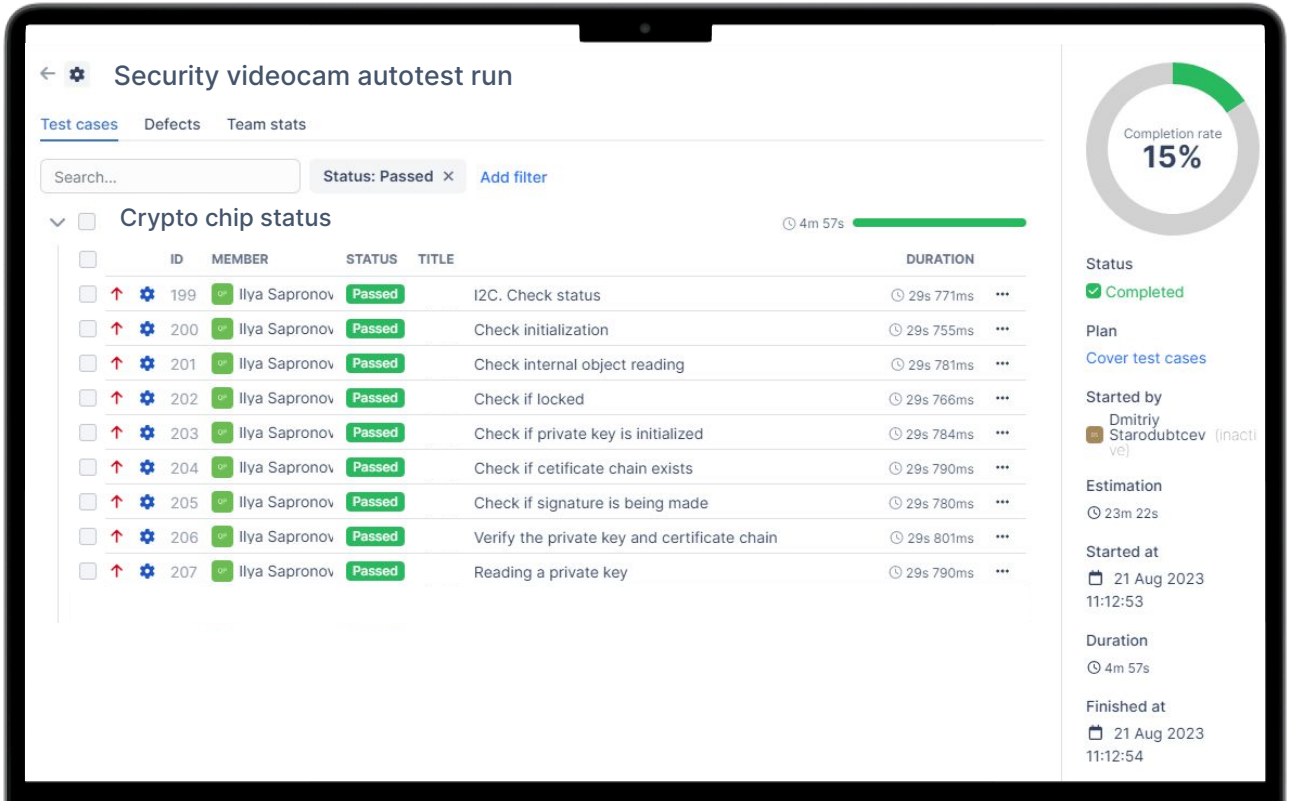


Наши приоритеты тестирования

- Интеграционное тестирование
- Тестирование безопасности
- Нагрузочное тестирование
- Приемочное тестирование

Наши приоритеты тестирования

Интеграционное тестирование



Наши приоритеты тестирования

Нагрузочное тестирование

```
while(1)
{
	crypto_chip.send_message("sign_data:data1");
	script.msleep(1);
	crypto_chip.send_message("sign_data:data2");
	script.msleep(1);
	crypto_chip.send_message("sign_data:data3");
	script.msleep(1);
	crypto_chip.send_message("sign_data:data4");
	script.msleep(1);
}
```

Наши приоритеты тестирования

Тестирование безопасности

300

Security. Brute Force

General | Properties | Runs | History | Defects

Pre-conditions

Chip is initialized

Steps

1 Enter command to toolbox command line

Input data

send_message(“brute_force;100000”)

Expected result

Logs with 100000 messages “Wrong password”

301

Security. Change Password

General | Properties | Runs | History | Defects

Pre-conditions

Chip is initialized

Steps

1 Enter command to toolbox command line

Input data

send_message(“write_password_object; olololo”)

Expected result

Message “Changing this object is not allowed”

302

Security. Sniff password

General | Properties | Runs | History | Defects

Pre-conditions

Chip is initialized

Steps

1 Solder to the I2C bus

2 Start signing

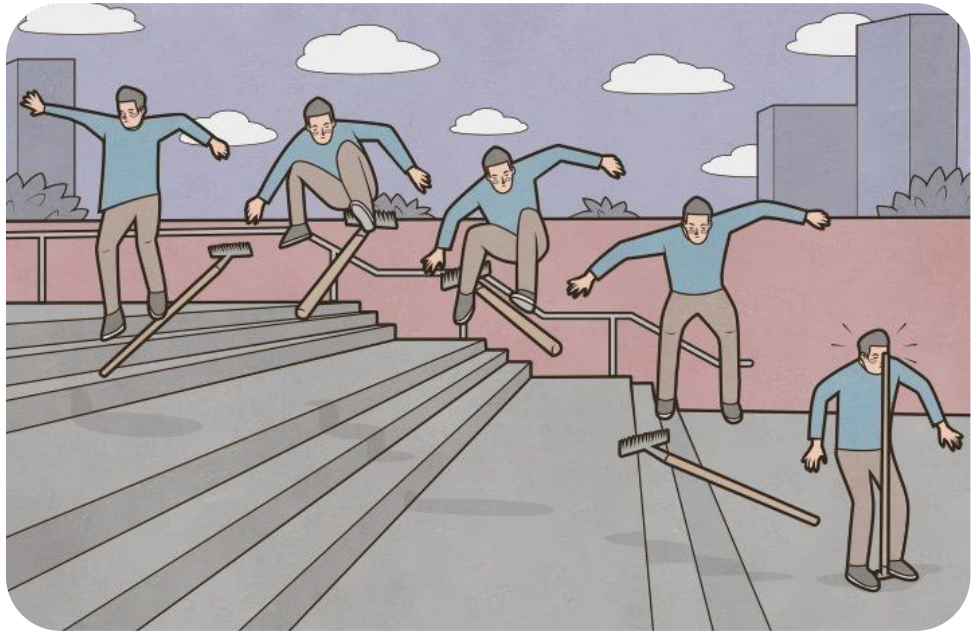
3 Process

4 Search for password in raw data

Expected result

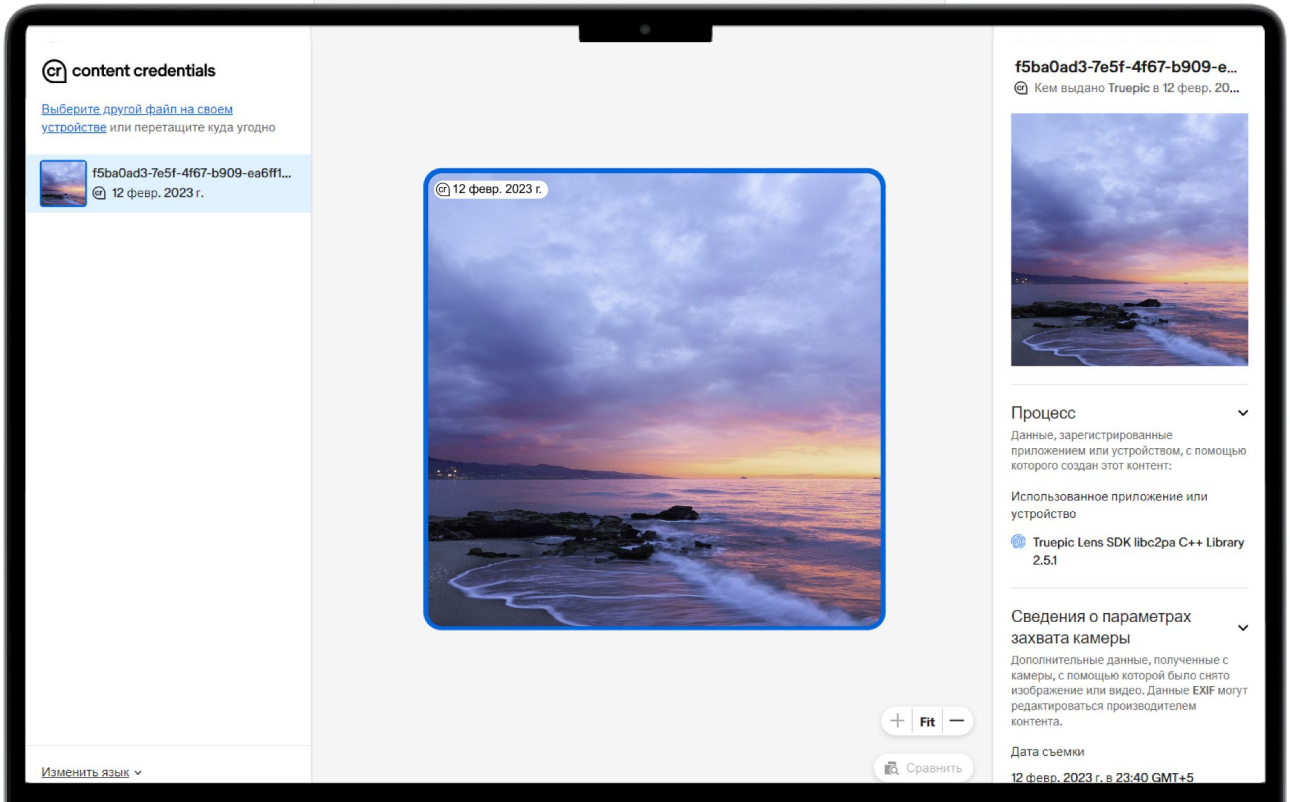
Password was not found in raw data

Коротко о нашей разработке



Наши приоритеты тестирования

Приемочное тестирование



Наши приоритеты тестирования

Автоматизация приемочных тестов

Проверка метаданных
на устройстве с помощью
сторонних библиотек

mbedTLS openSSL

Генерация контента
и его валидация
вне устройства

cypress

Грабля #1

Особенности нашего проекта

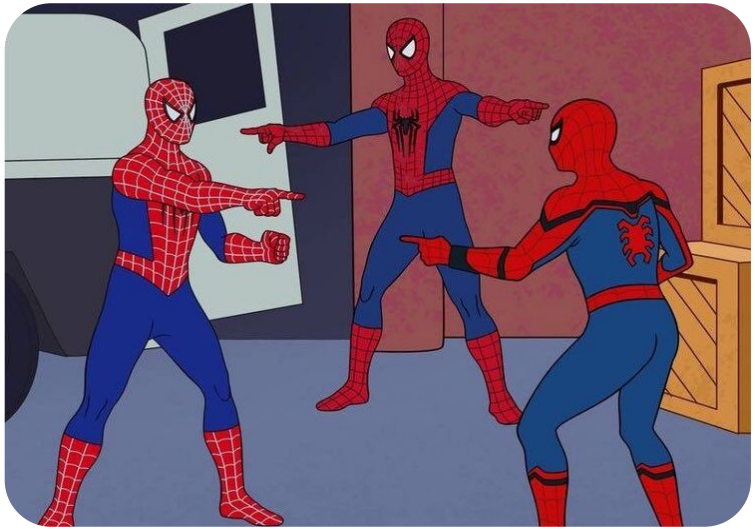
Чистый C

Криптография

Много Legacy

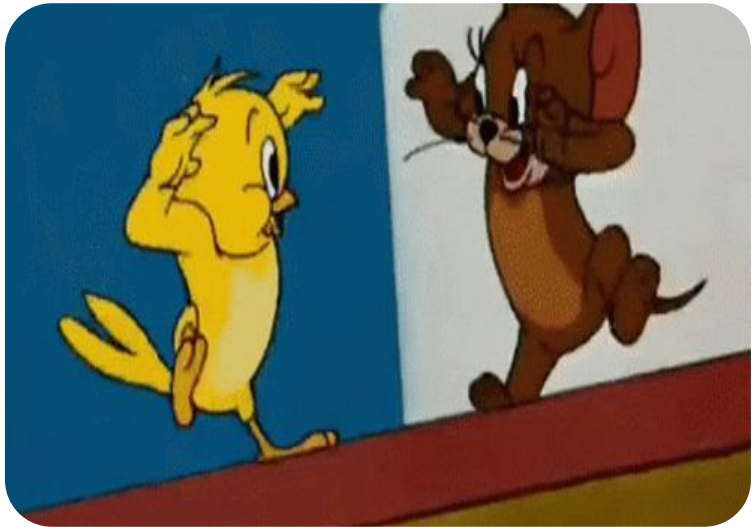
Real Time Operating System

Грабля #1



А кто будет тестировать?

- Отсутствие embedded специализации у тестировщиков
- Разработчики вынуждены сами заниматься тестированием



Тестировщики Разработчики

Вынужденное тесное сотрудничество

- Повышение квалификации тестировщиков
- Разработка проходит с мыслями о тестировании

Грабля #2

Почему так медленно?

То, что хорошо работает при низкой нагрузке, может сломаться при частых запросах.


Подозрительно частые запросы. Буду отвечать медленнее.
Я не хакер, клянусь..

Своевременное нагрузочное тестирование

Поможет выявить неочевидные детали работы устройства!

Грабля #3

Дыра в безопасности

Кажется мы предусмотрели всё...
...или нет?

Обращение к «белым хакерам»

- Свежий взгляд на безопасность проекта
- Команда с опытом взлома устройств

Грабля #3: итог теста безопасности

Все хорошо!

> Пароль не удалось обнаружить
> в eMMC памяти

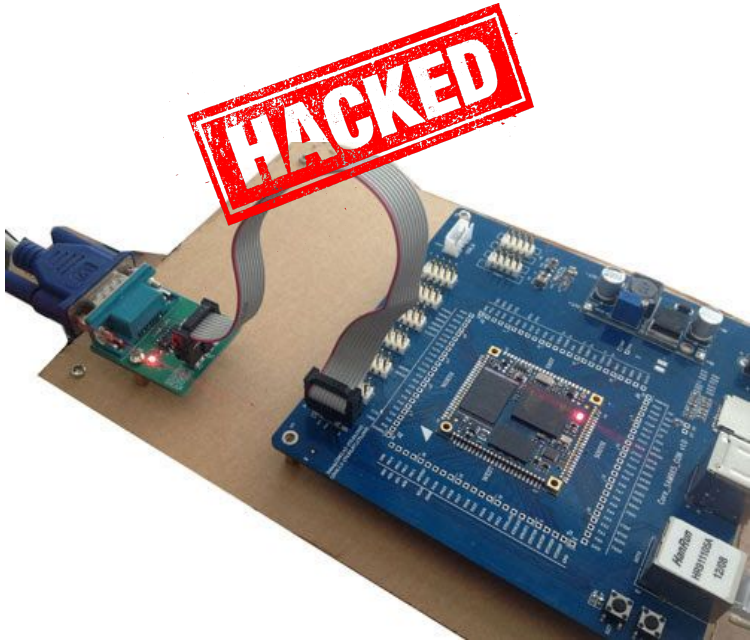
Грабля #3: итог теста безопасности

Или нет..

Открытый дебаг интерфейс

Незашифрованная память

Отладочная информация



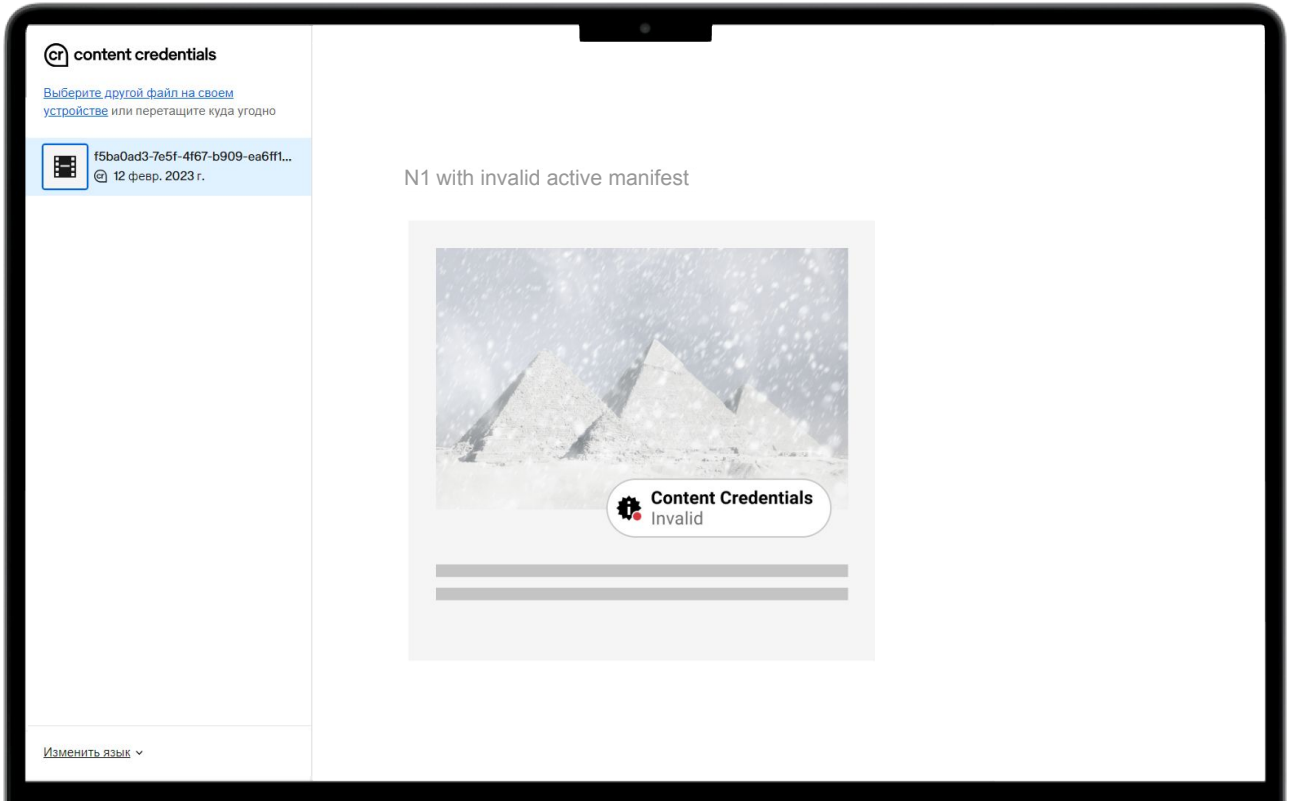
Грабля #4

Принес домой – а оно не работает

- Крипточип был не инициализирован
- Отсутствует сертификат
- Неверный сертификат

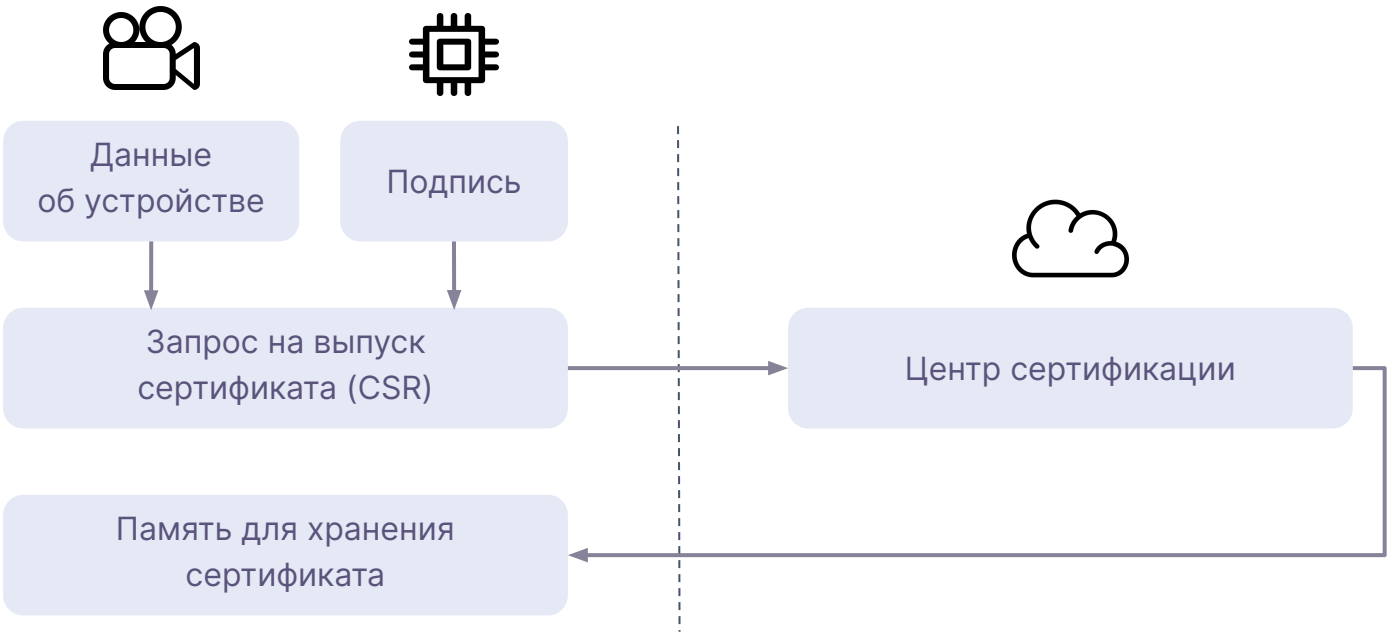
Грабля #4: как это выглядит

Сломанная верификация



Грабля #4: почему так может быть

Не всем сертификатам можно доверять



Грабля #5

Как мы чуть не потеряли много 💸💸💸

Некоторые изменения конфигурации могут оказаться необратимыми.

Тестирование на производстве

- Скрипт проверки валидной работы крипточипа
- Приемочные тесты

Грабля #6

Third-party библиотеки

У любой Third-party библиотеки
могут проявиться особенности работы
на target платформе.

Грабля #6: как не наступить

Тестирование third-party библиотек

Отдельное тестирование используемого функционала third-party библиотек.

Грабля #7

Отсутствие юнит тестов

When you finally find the person
who wrote all the bad code

Грабля #7: проблема

Почему писать юнит тесты сложно

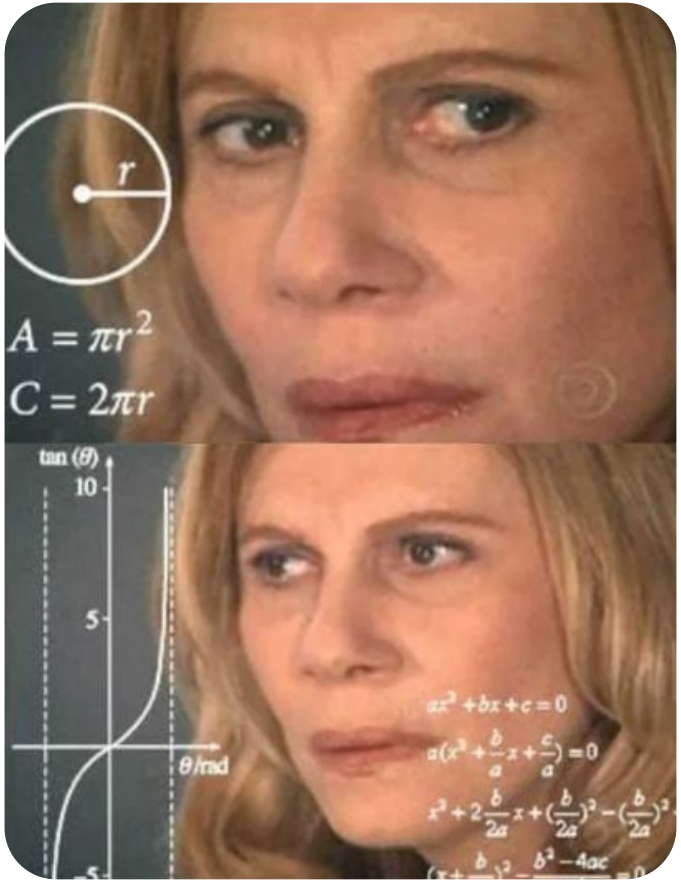
1) Вычисления в реальном времени

2) Зависимость от аппаратного обеспечения

Грабля #7: как не наступить

Покрыть тестами весь код

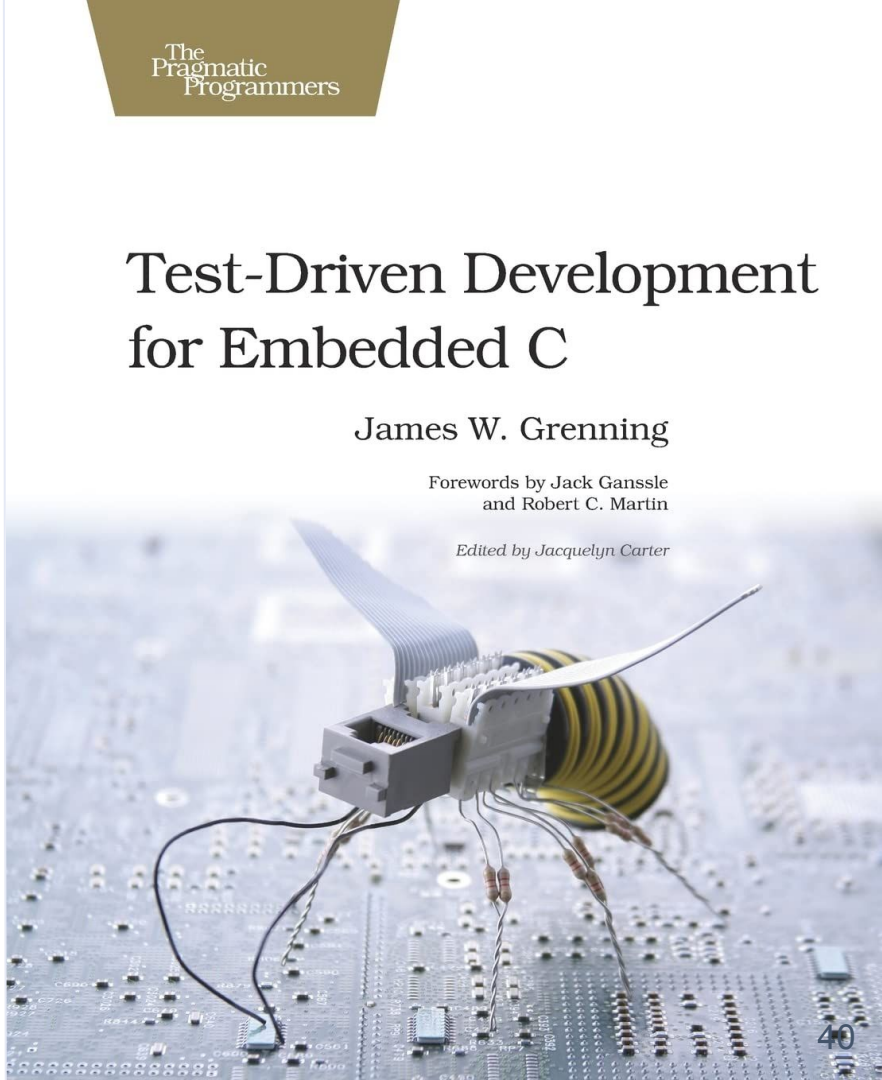
А есть ли способ покрыть юнит тестами как software, так и hardware код?



Два решения

- Тестирование в целевой среде
- Двойное тестирование

https://www.amazon.com/Driven-Development-Embedded-Pragmatic-Programmers/dp/193435662X

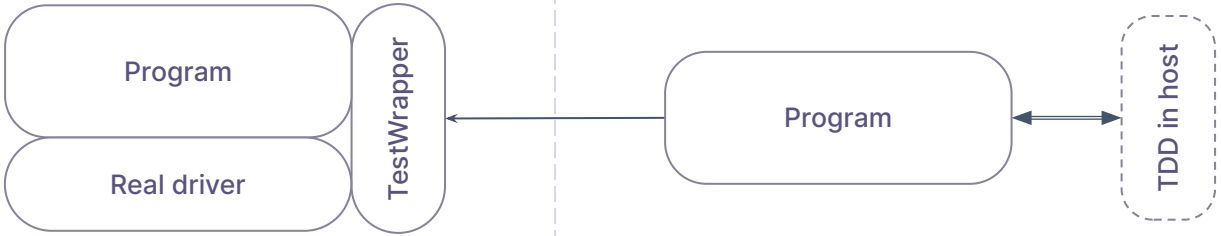


Первое решение

Тестирование в целевой среде

target

host



Первое решение

Тестирование в целевой среде

Преимущества:

- максимально возможное покрытие кода тестами
- тесты и код устройства запускаются в одной среде

Недостатки:

- медленный запуск тестов
- сложность установки

Второе решение

Двойное тестирование (Dual Targeting)

target

host

Второе решение

Тестирование в хост-среде (Dual Targeting)

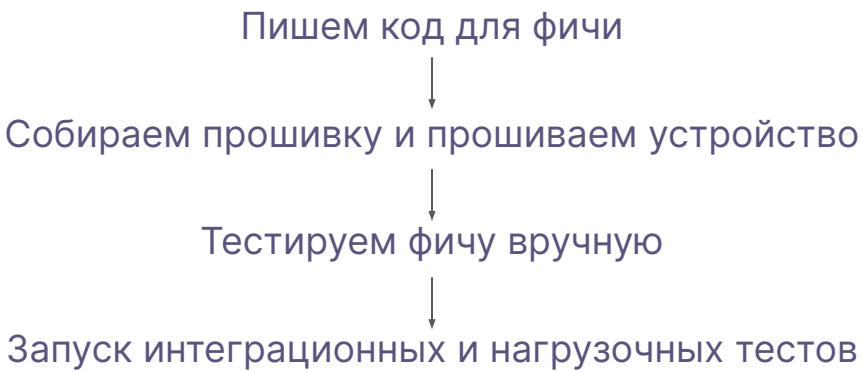
Преимущества:

- тесты запускаются быстро
- код можно писать без доступа к устройству
- разделение чистого кода и кода с зависимостями устройства

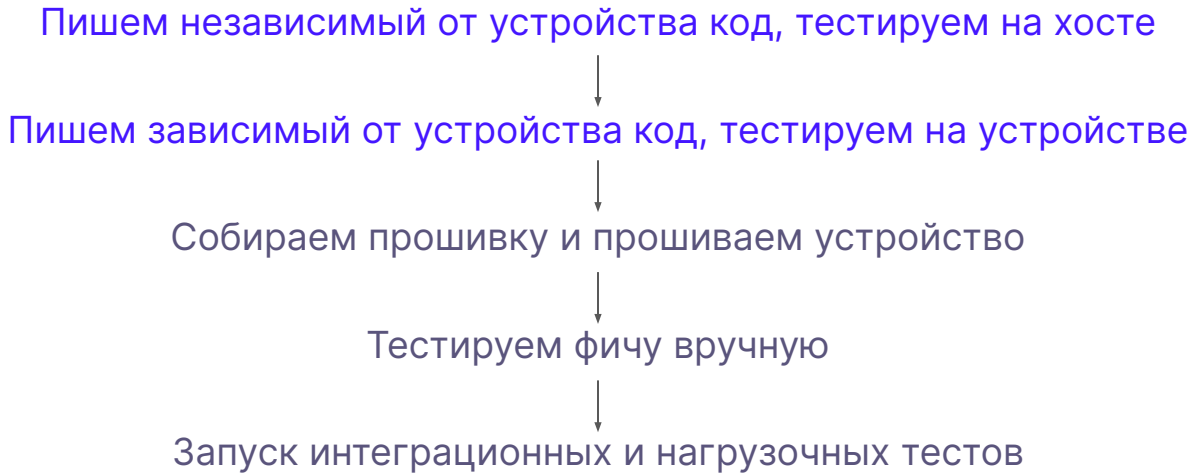
Недостатки:

- тесты и код устройства запускаются в разных средах

Как мы разрабатывали



Что изменилось с Dual Targeting



Независимый от устройства код

- Алгоритмы
- Вычисления, обработка данных
- Внешние библиотеки

Пример использования Dual Targeting

Настраиваем среду

1) Конфигурация тестовой среды
    - IDE - Visual Studio
    - Тестовый фреймворк для software - GoogleTest
    - Тестовый фреймворк для hardware - самописный
2) Конфигурируем проект
    - для x86 платформы
    - для целевой платформы

Пример использования Dual Targeting. Software код.

🔴 Пишем тест, который провалится

```
TEST(CertificateChain, HaveAtLeastOneCert)
{
	char *certificate_chain = "MIICzTCCAbUCAQAwgYcxCzAJBgNVBAYTAkdC\GP";
	uint8_t ret_code;

	ret_code = checkCertificateChain(certificate_chain);

	EXPECT_EQ(0, ret_code);
}
```

🟢 Делаем тест зеленым

```
uint8_t checkCertificateChain(char *_certificate_chain)
{
     char *c_begin_certificate = "-----BEGIN CERTIFICATE-----";

     uint16_t certificate_chain_length = strlen(_certificate_chain);
     uint16_t begin_found = 0;

     if ((begin_found = getSubstringCount(
        certificate_chain,
        c_begin_certificate,
        certificate_chain_length)) == 0 )
     {
        return -1;
     }

      // do the same for END CERTIFICATE footer
     char *c_end_certificate = "-----END CERTIFICATE-----";
}
```

🔵 Рефакторинг

- Делаем код более читаемым
- Оптимизируем работу

Зависимый от устройства код

- Файловая система
- Драйвера
- Обработка прерываний

Пример использования Dual Targeting. Hardware код.

🔴 Пишем тест, который провалится

```
TEST(CertificateChain, ReadCertificateChain)
{
	char *file_name = "test/dummy.txt";
	certificate_chain = calloc(BUFFER_SIZE, sizeof(uint8_t ));
	uint8_t ret_code;
	uint8_t system_partition = SYSTEM_DATA;

	ret_code = readFileFromStorage(SYSTEM_DATA, file_name, certificate_chain);

	EXPECT_EQ(0, ret_code);
}
```

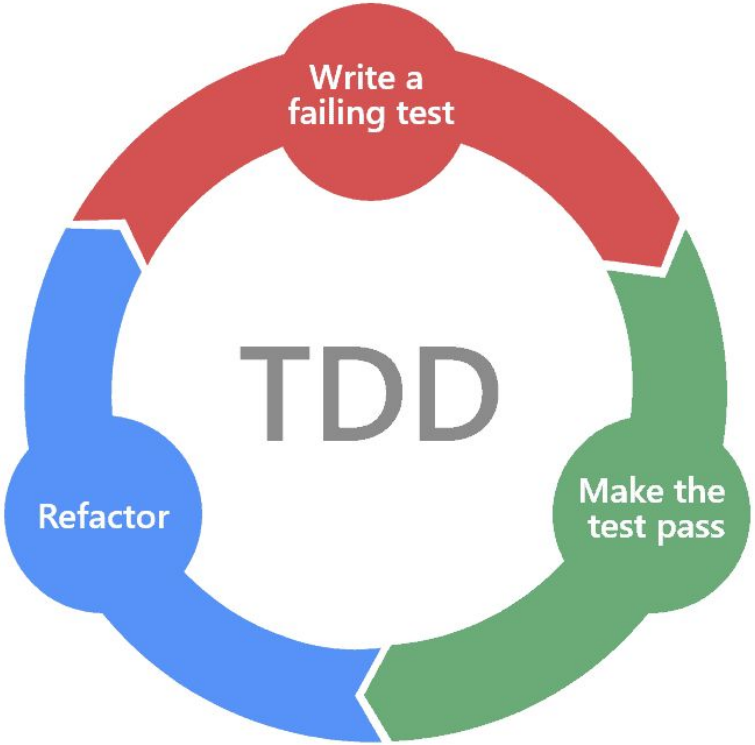
🟢 Делаем тест зеленым

```
uint8_t readFileFromStorage(uint8_t  partition, char* file_full_name, char*_buffer);
{
	if (partition == null || file_full_name == null || buffer == null)
	{
		return ERROR_INVALID_PARAMETER;
	}
	if(((uint32_t)buffer % BUFFER_ADDRESS_ALIGNMENT) != 0)
	{
		return ERROR_INVALID_ALIGNMENT;
	}

	mutex.lock();
	// trigger some HW low level functions
	mutex.unlock();

	return OK;
}
```

🔵 Рефакторинг

- Делаем код более читаемым
- Оптимизируем работу

Да, это ТДД



Стоило ли оно того?

Плюсы

- Архитектура кода
- Легче поддерживать код
- Код служит документацией

Минусы

- Не является панацеей
- Высокий порог вхождения

Возвращаясь к граблям

- Тщательно выбирайте приоритеты тестирования
- Объединяйте тестирование и разработку
- Не для всех проблем есть решение, но всегда найдется компромисс

Даже если все работает, всегда есть, к чему стремиться :)

Разработчики встраиваемых систем

Кузьмичева Ольга & Кирпичников Антон

okuzmicheva@tourmalinecore.com

akirpichnikov@tourmalinecore.com

Web site
tourmalinecore.com

Vk
tourmalinecore