Разработка all-in-one тестовой станции для производства

(в условиях COVID-19)

SBER DEVICES

Кто мы и что мы делаем?

Александр Фомин

— 7 лет в QA
— Занимался автоматизацией тестирования систем для автоуправления тракторами в сельском хозяйстве (precision agriculture)
— Участвовал в R&D проекте — автономный трактор
— Руководит командой автоматизации тестирования HW в SberDevices
— В свободное от работы время увлекается разработкой под android

Кто мы и что мы делаем?

Александр Гришин

— Более 10 лет в QA
— Работал над 3D сканерами и навигаторами дополненной реальности
— Обеспечивал качество умных устройств от Яндекса: Станция, Модуль, Мини и другие
— Руководит отделом аппаратного тестирования в SberDevices
— PO и PM по всем тестовым инструментам в SberDevices

О чём расскажем?

— Как сделать хорошо умные устройства в Китае
— Что для этого нужно
— Какие решения мы использовали
— С какими трудностями столкнулись
— Как их победили
— Что из этого вышло


BACK
TO
THE
PAST

— **Зачем нужна all-in-one станция?**
— Формируем чек-лист проверок
— Заказываем чембер в Китае
— Вопрос-ответ перед разработкой софта
— Пишем первый код
— Знакомство с чембером в Москве
— Пытаемся подружить код и чембер
— Чембера на фабрике в Китае

С чего начинался SberDevices?

Конечно же с идеи сделать мир лучше! Но как?
Сделать телевизоры наших пользователей умнее
при помощи СберБокса!

Дальше больше?

Захотелось по-настоящему удивить всех, а также показать на что способны наши технологии…
Так родился SberPortal!

Что внутри Портала?

Процессор с NPU (Neural Processing Unit)
Микрофонная матрица
Wi-Fi и Bluetooth
Мощный звук
Дисплей
Камера
Android

…И немножечко магии

Зачем нужна all-in-one станция?

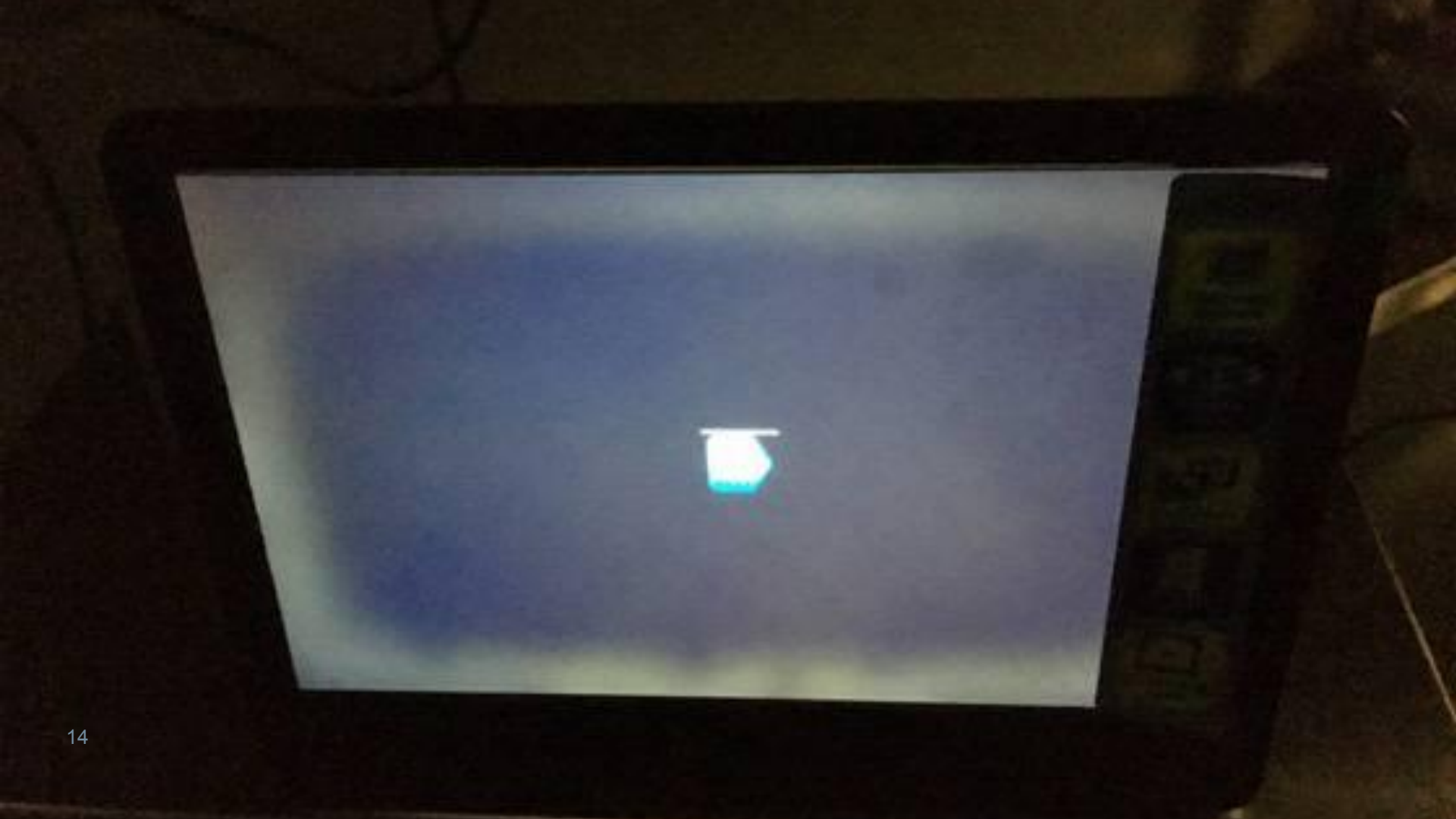
Сделано в Китае

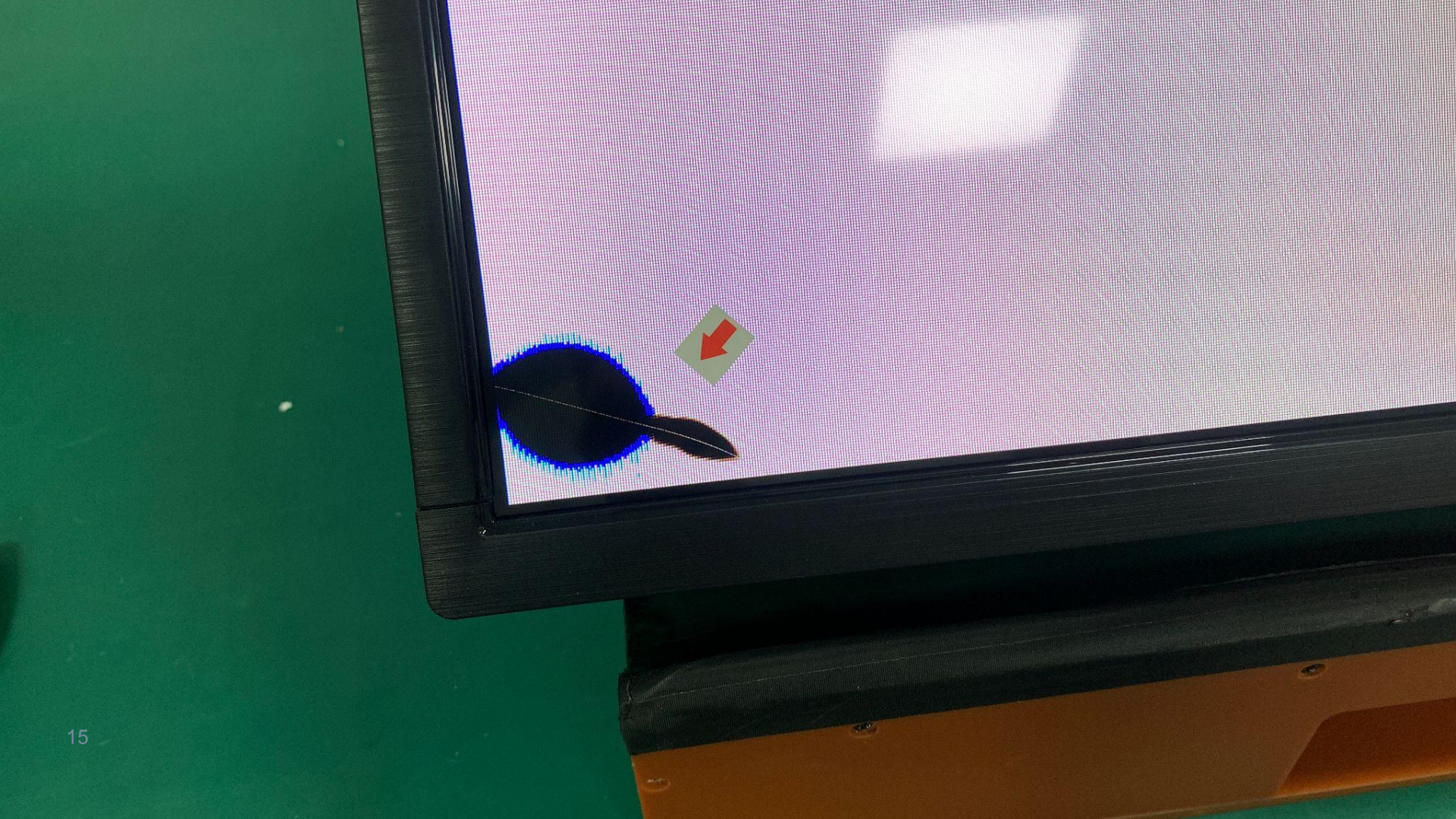
— У фабрики и SberDevices разные KPI: фабрике нужно много и быстро, SberDevices — качественно и в срок

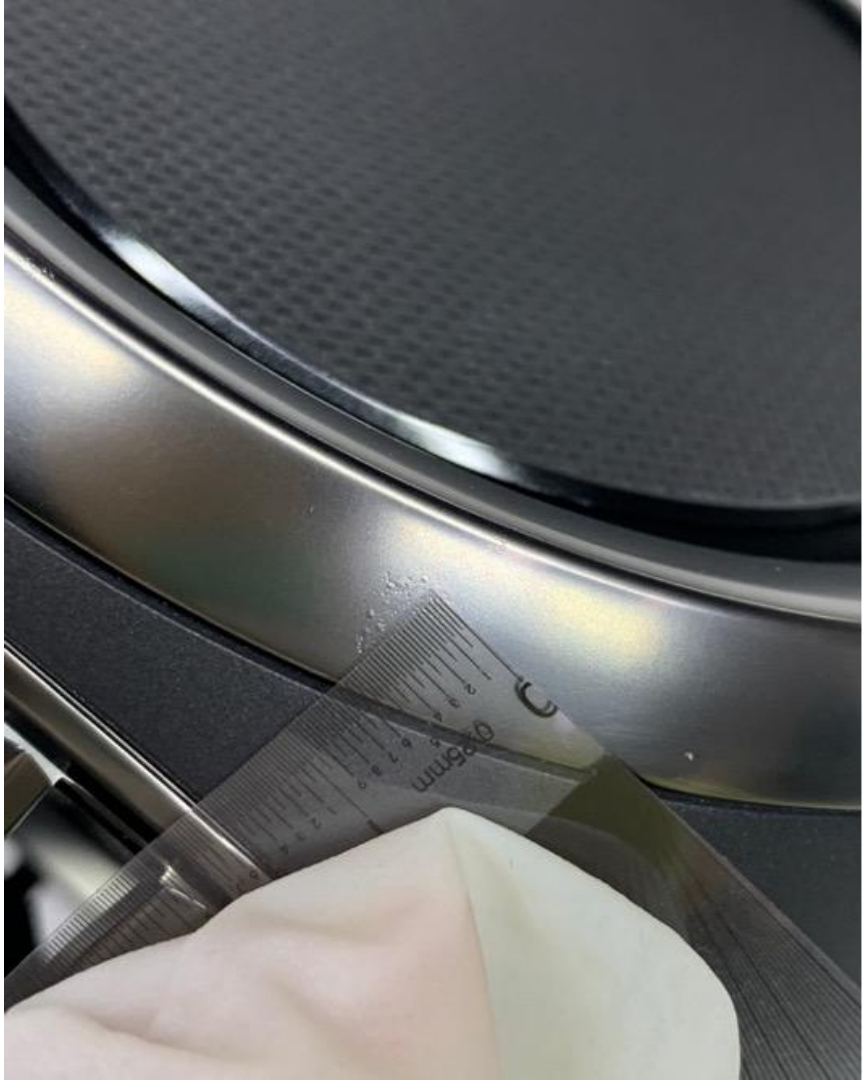
— Контроль качества — не всегда про качество: фабричные методики тестирования основываются лишь на проверке физических параметров, не поднимаясь на уровень софта

— Рядовым сотрудникам фабрики нет дела до качества продукции — главное увидеть заветный PASS

Как оно? На фабрике?







Как обеспечить качество?

— Осознать объем тестируемого функционала и объем тестов

— Приступить к написанию тест плана и тест кейсов

— Задуматься о том: сможет ли фабрика выдержать наши критерии качества?

Как сделать хорошо?

— Изучили опыт западных компаний

— Поняли, что обеспечение качества на фабрике — это многогранный процесс

— Отделили мух от котлет и приступили к делу

Решение есть!

Поняли, что нам необходимо создать систему, которая будет сама проверять устройства и отвечать на вопрос о качестве устройства

Приступили к разработке
All-in-One тестовой станции или же Чембера

Но что такое Чембер?

Решение есть!

Чембер — звуко- и радио-непроницаемая камера, внутри которой расположено контрольно-измерительное оборудование

Размещается на фабрике, управляется удаленно

Цель чембера

— Проверять в одинаковых условиях
— Проверять то, что не может человек
— Работать 24/7
— Беспристрастно выносить результат тестирования
— Выявлять аппаратный брак на ранних стадиях

SBER DEVICES

— Зачем нужна all-in-one станция?

— **Формируем чек-лист проверок**

— Заказываем чембер в Китае

— Вопрос-ответ перед разработкой софта

— Пишем первый код

— Знакомство с чембером в Москве

— Пытаемся подружить код и чембер

— Чембера на фабрике в Китае

Что будем проверять?

Для обеспечения выпуска качественной продукции нам необходимо проверять на каждом устройстве:

— Wi-fi и Bluetooth
— Наличие термоинтерфейса
— Камеру
— Дисплей
— Кнопки
— Микрофоны
— Динамик

Поговорим про звук?

Почему звук важен?

— Основной интерфейс взаимодействия с устройством — это микрофоны

— SberPortal обладает качественным динамиком

HW компоненты, взаимодействующие со звуком

— Микрофоны (микрофонная матрица)

— Динамики

Динамик

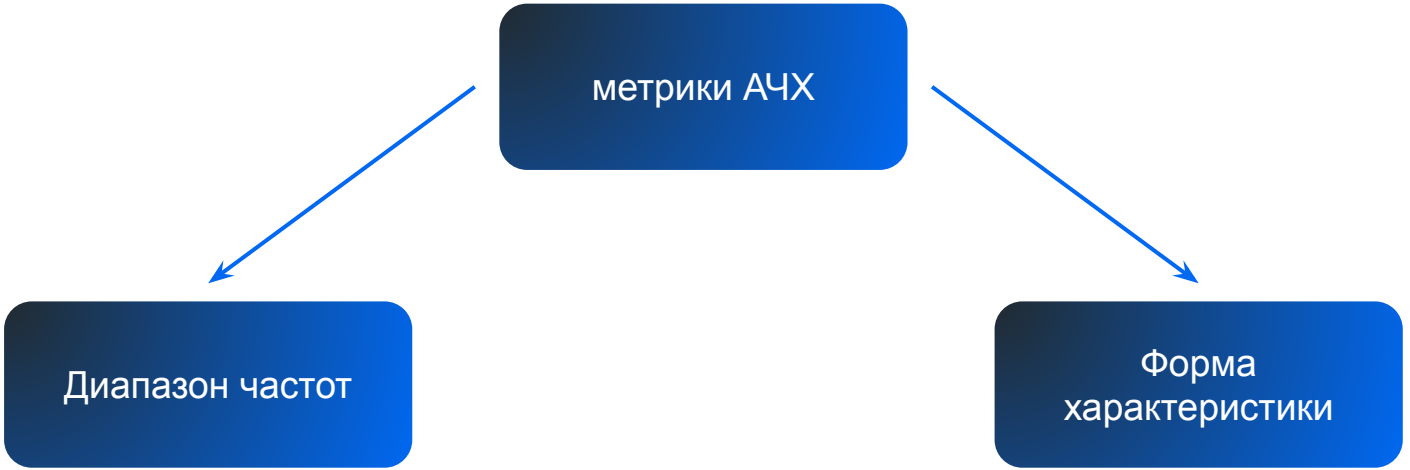
Характеристики звука, которые мы снимаем:

— АЧХ — Frequency response, FR, амплитудно-частотная характеристика

— КНИ — Total harmonic distortion, THD, коэффициент нелинейных искажений

АЧХ

Метрики АЧХ



Диапазон частот АЧХ

Полоса частот, в которой АЧХ аудиосистемы определен

Примеры диапазонов частот из жизни:

- Сабвуфер: 20 – 65 Гц
- Человеческое ухо: 20 – 20 000 Гц
- Телефонная линия: 300 – 3 400 Гц
- Современная музыка: 40 – 12 000 Гц

Диапазон частот АЧХ

Пример

Лунная соната

40 – 12 000 Гц

+

20 – 65 Гц

Сабвуфер

=

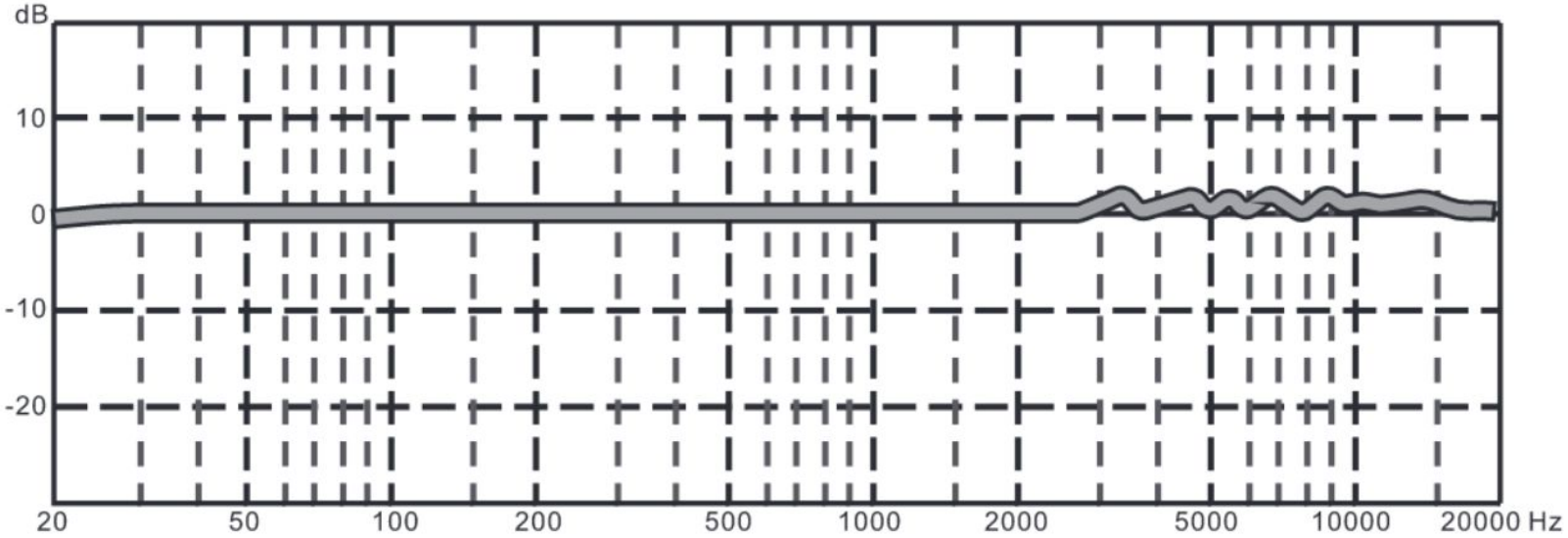
Форма АЧХ

— Под формой АЧХ понимают в прямом смысле геометрическую форму характеристики

— Для каждой задачи нужен АЧХ “своей” формы

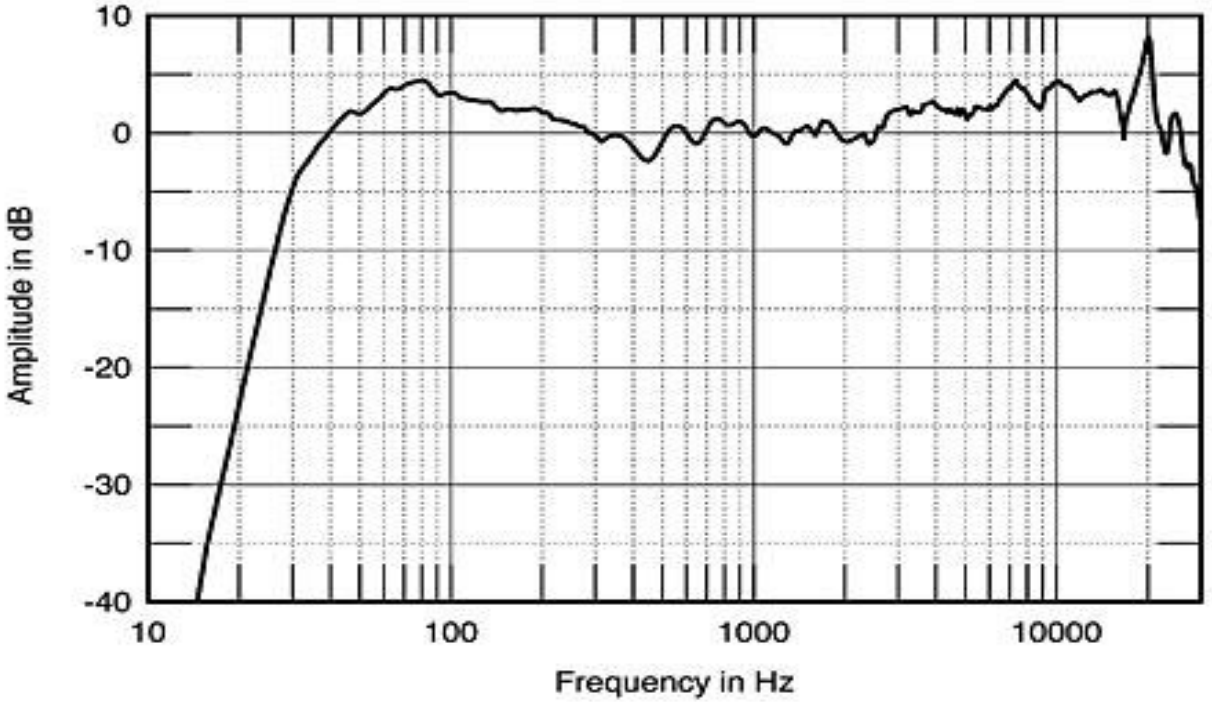
Форма АЧХ

Измерительный микрофон

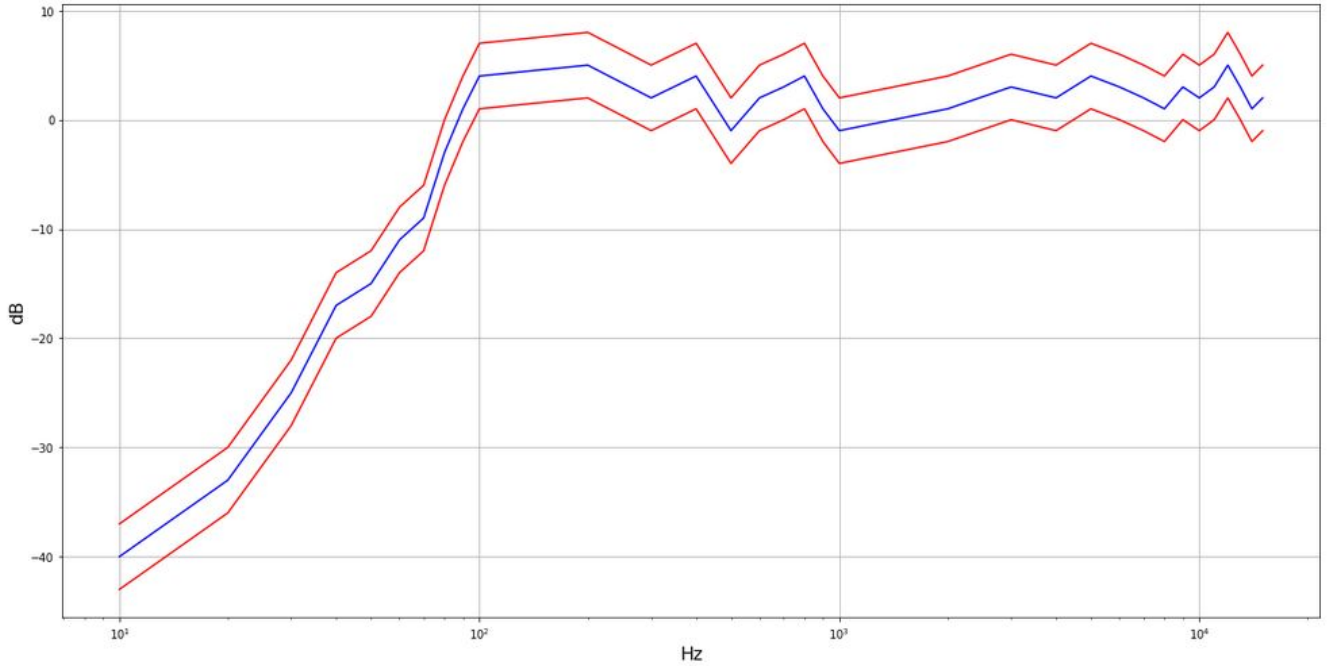


Форма АЧХ

Колонка



Критерий корректности АЧХ



КНИ — коэффициент нелинейных искажений

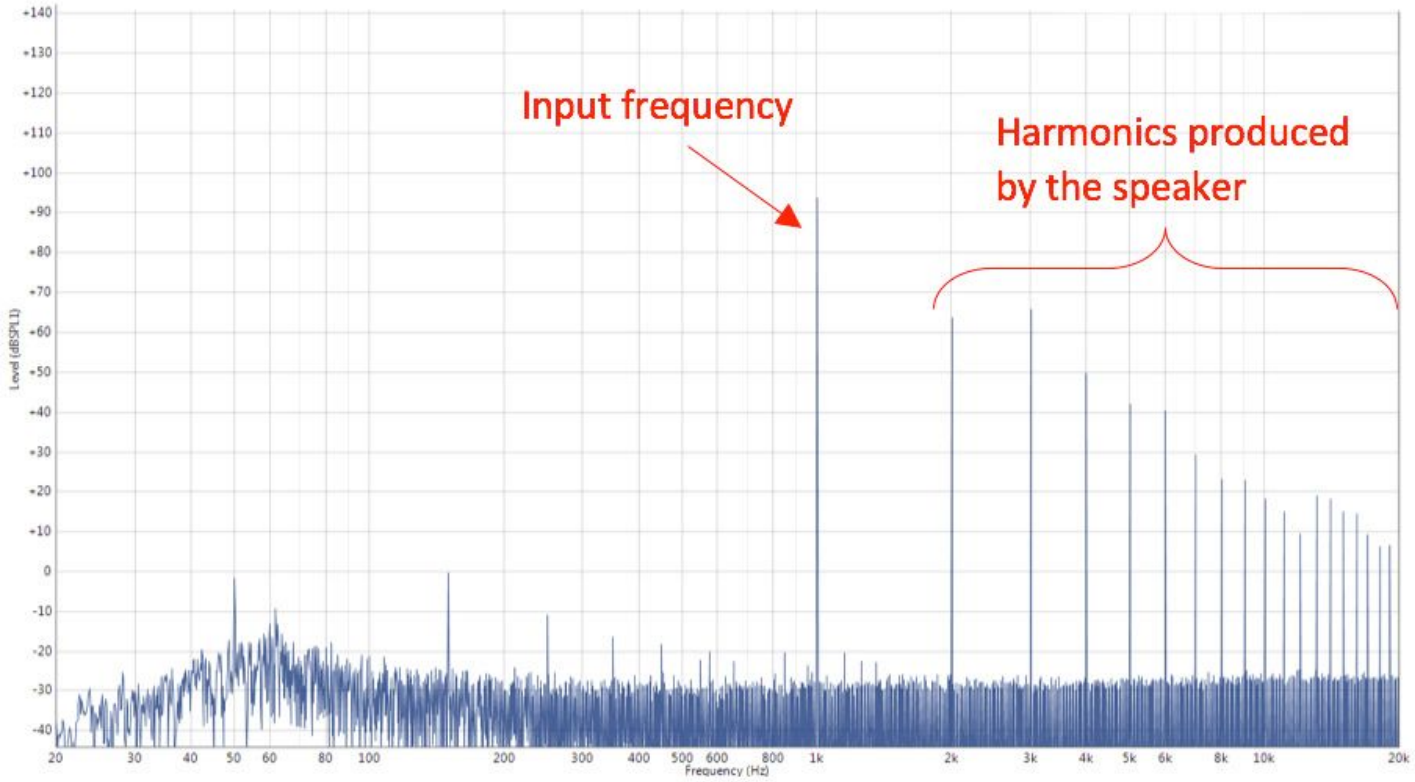
— Нелинейные искажения создают в выходном сигнале гармоники, которых не было во входном сигнале (с кратными частотами)

— Нелинейные искажения = потери энергии входного сигнала

— В аудиотракте возникают нелинейные искажения. Но, в общем, нелинейные искажения — это не только про аудио

— КНИ — количественная оценка нелинейных искажений. Может измеряться в %, либо в дБ

Пример КНИ

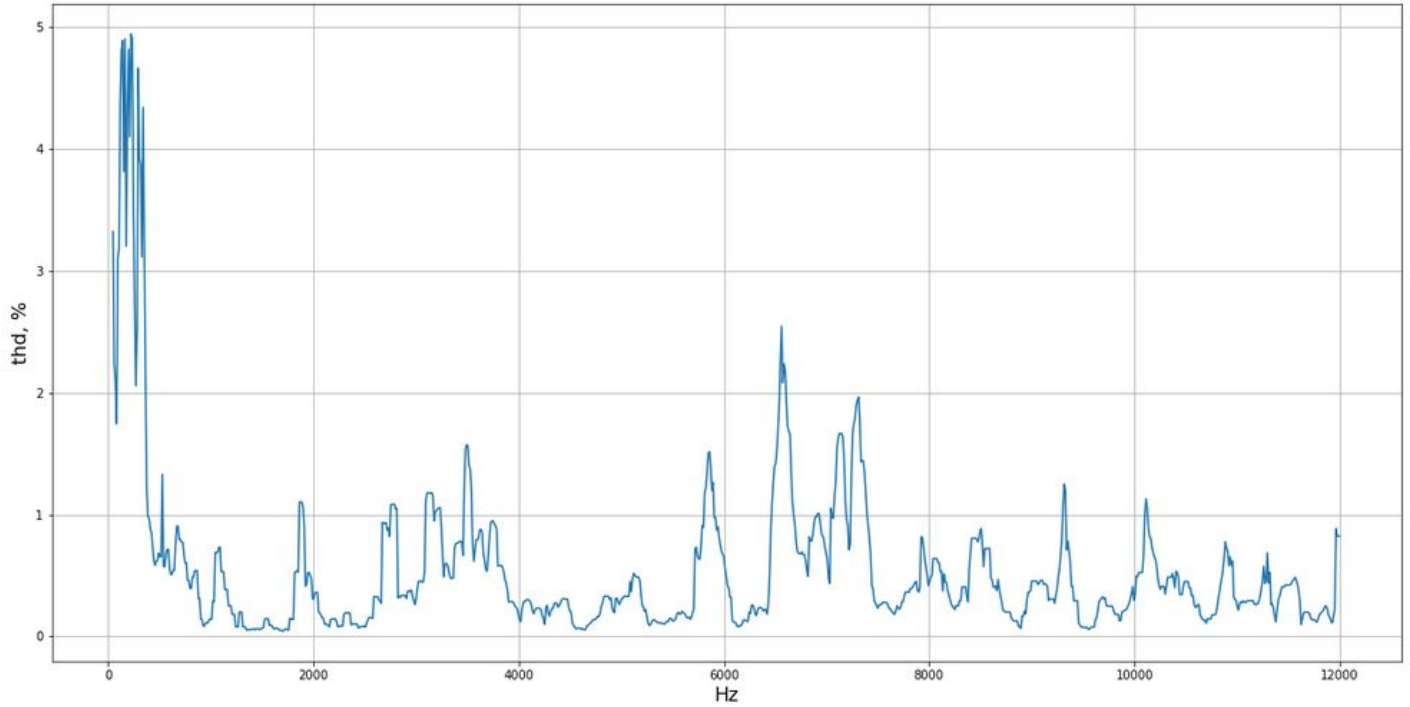


КНИ простыми словами

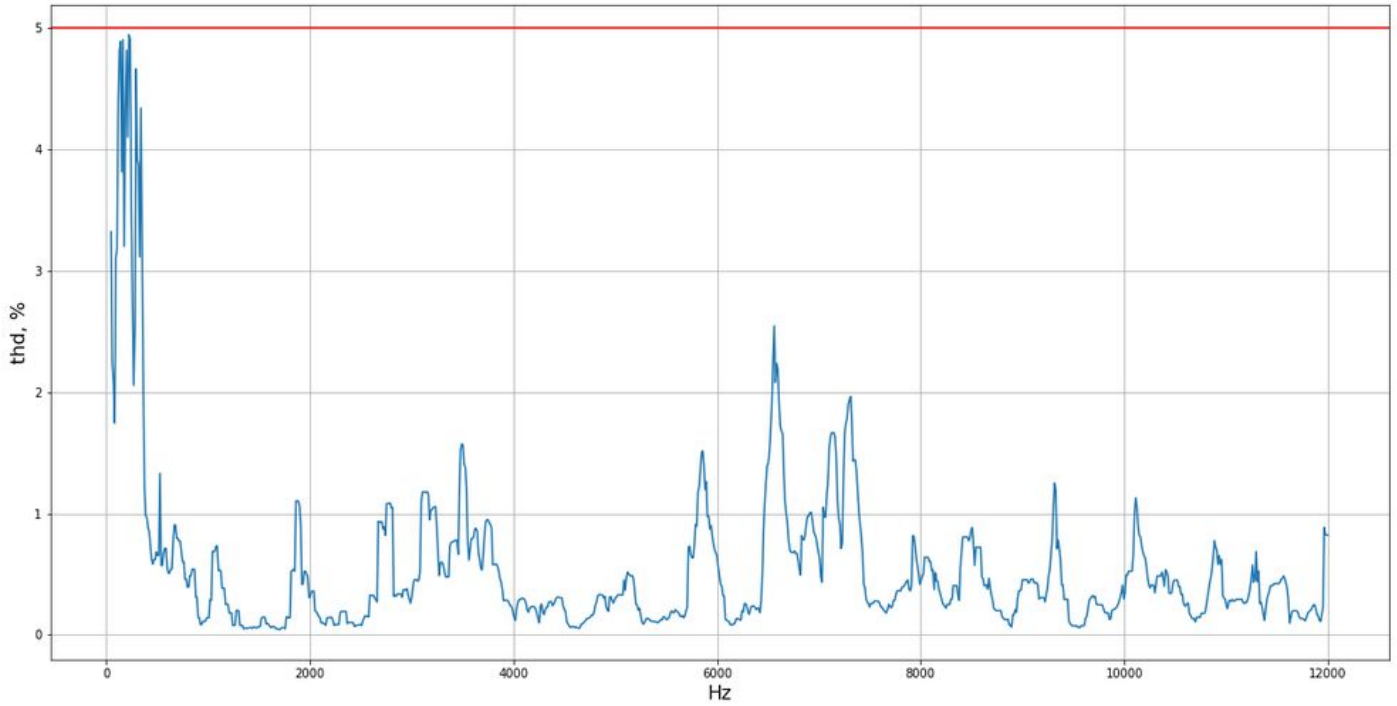
Чем КНИ выше, тем хуже

Например КНИ 15 % на 1кГц значит, что 15 % от энергии сигнала на 1кГц будет “потеряно” в виде звука на кратных гармониках

Пример значений КНИ для всего спектра частот



Критерий корректности КНИ



Микрофонная матрица

— Позволяет записывать звук более высокого качества

— Возможность определять направление на источник звука и не только

— Микрофоны в матрице должны быть “одинаковыми” по характеристикам

Микрофоны

На примере SberBox Top

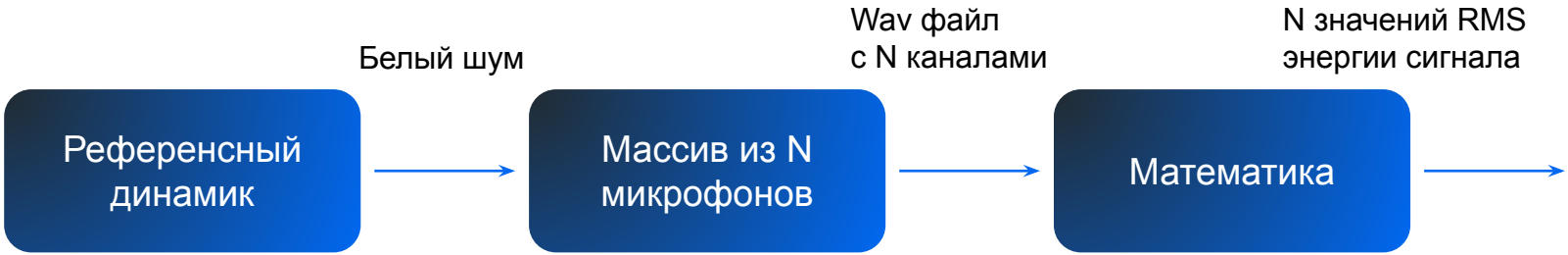
Микрофонная матрица

Микрофоны. Какие проблемы чаще встречаются?

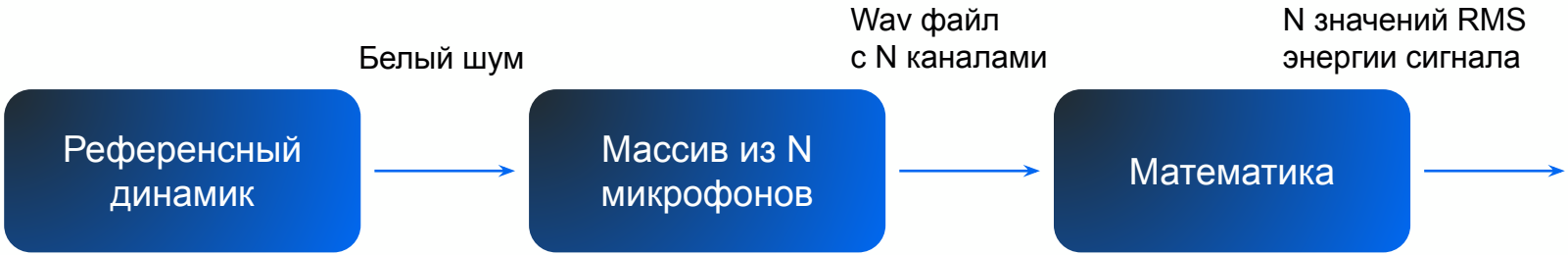
— Микрофон записывает более “слабый” звук, чем должен

— Наличие посторонних артефактов в записи

Микрофоны. Как мы оцениваем качество?



Микрофоны. Как мы оцениваем качество?



— **RMS — Root Mean Square** или среднее квадратическое Интегральная характеристика для одного из N микрофонов. Грубо говоря, это средняя энергия, записанная микрофоном по всем частотам спектра

— Проверка занимает ~ 20 секунд

Микрофоны. RMS простыми словами

Ситуации:

— Если микрофон с дефектом пишет звук слабее, чем должен, то RMS такого микрофона будет ниже, чем у “хорошего”

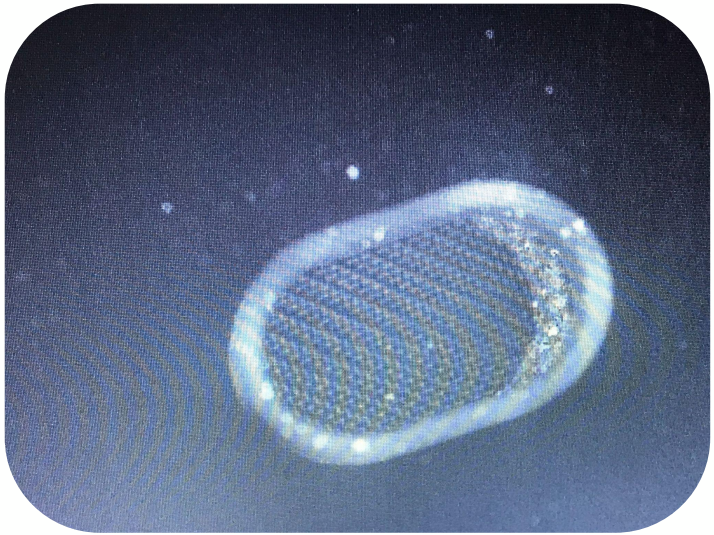
— Если в записи присутствуют артефакты, то артефакты = лишняя энергия. RMS от этого растет.

Изучив datasheet микрофонов, а также имея статистику измерений с устройств, можно оценить **допустимое “окно”**, в которое должен попадать RMS.

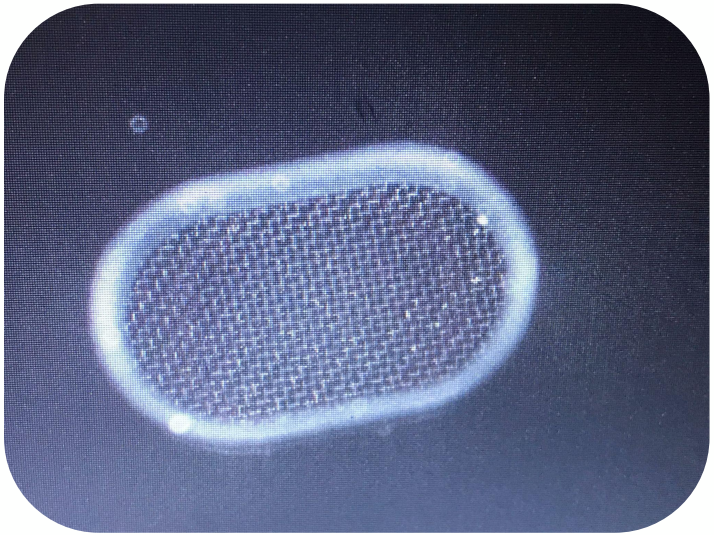
Автотест может с помощью простого сравнения фактического значения RMS с допустимым окном вынести решение, что с микрофоном что-то “не так”.

Микрофоны. Некачественный монтаж.

Плохо



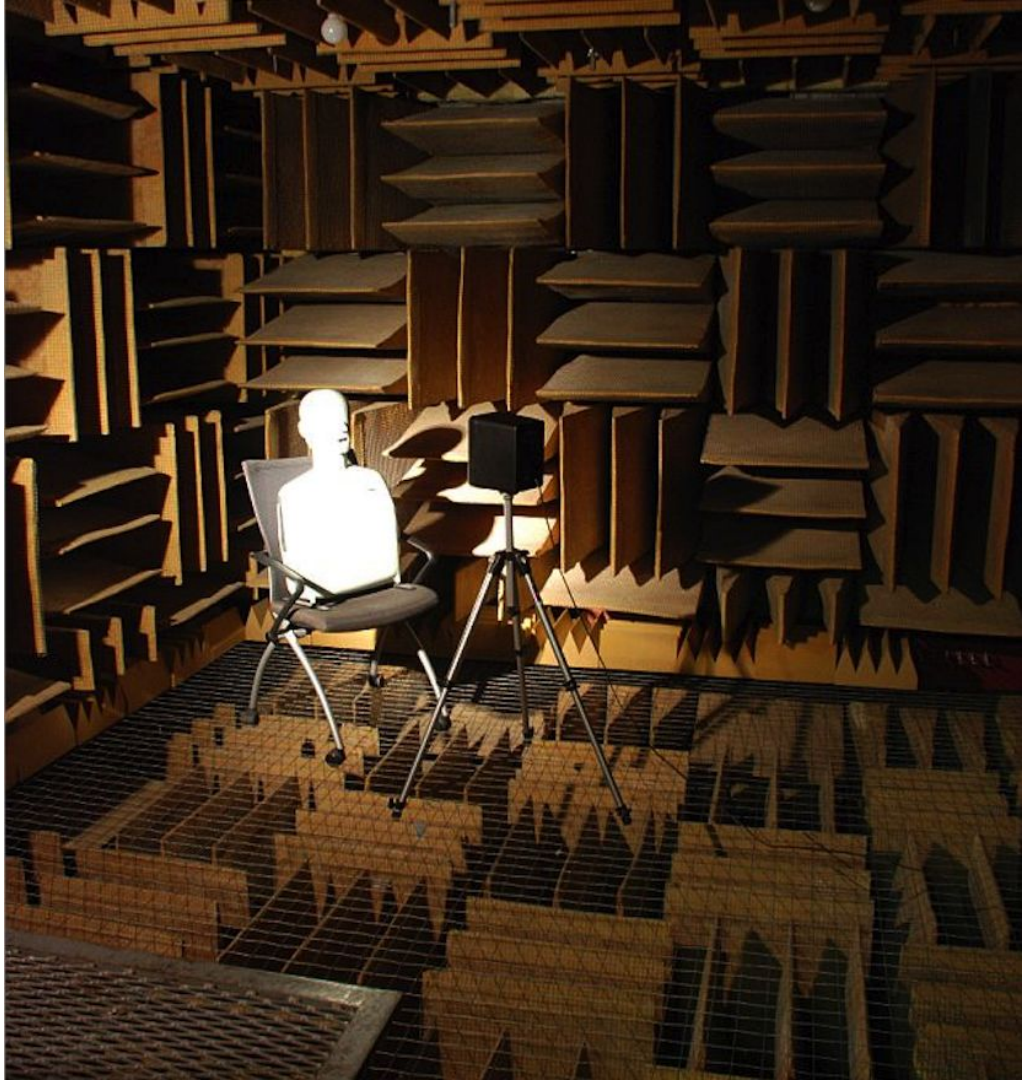
Хорошо



Для проведения акустических тестов чембер должен быть акустической камерой

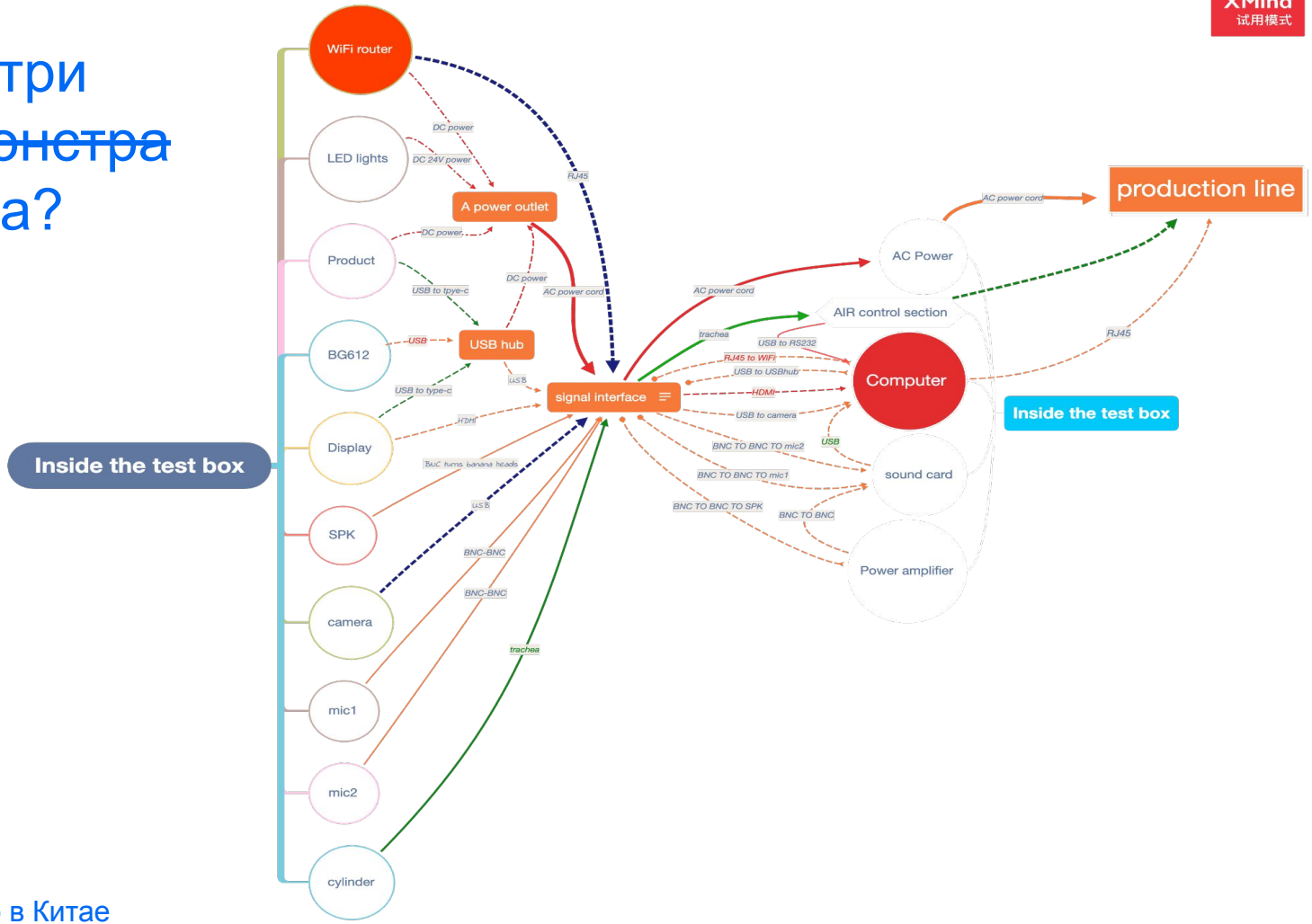
— Обеспечивает звукоизоляцию от внешней среды

— Не допускает переотражения звуковых волн внутри



— Зачем нужна all-in-one станция?
— Формируем чек-лист проверок
— **Заказываем чембер в Китае**
— Вопрос-ответ перед разработкой софта
— Пишем первый код
— Знакомство с чембером в Москве
— Пытаемся подружить код и чембер
— Чембера на фабрике в Китае

Что внутри этого ~~монстра~~ Чембера?



Заказываем чембер в Китае

С чего начинается Чембер?

— Создание прототипа на столе

— Выбор подрядчика

— Выбор оборудования

— Разработка первой 3D модели

— Производство опытного образца + верификация в Москве

С Чембером понятно, а что было дальше?

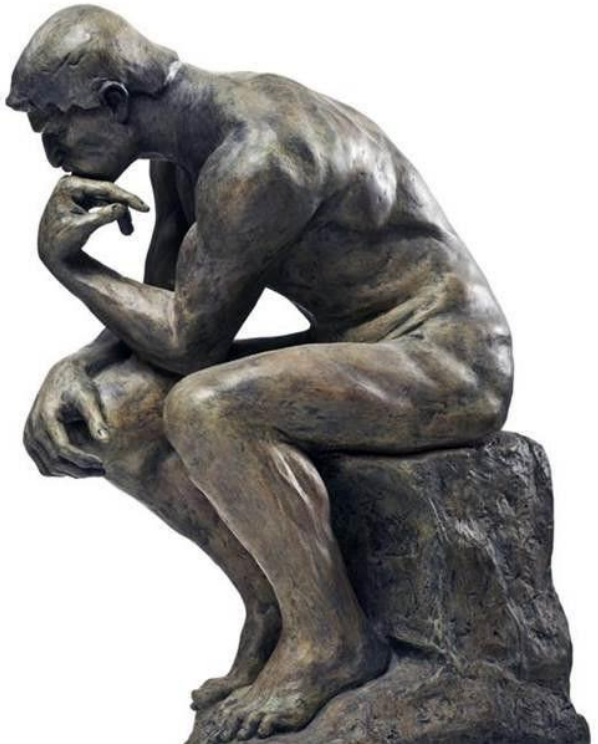
— Определились с оборудованием и моделью чембера

— Нашли фабрику и начали производства пилотного образца

— **Встал вопрос: а как его оживить?** Как заставить приносить пользу?

— Зачем нужна all-in-one станция?
— Формируем чек-лист проверок
— Заказываем чембер в Китае
— **Вопрос-ответ перед разработкой софта**
— Пишем первый код
— Знакомство с чембером в Москве
— Пытаемся подружить код и чембер
— Чембера на фабрике в Китае

Вопрос-ответ перед разработкой софта



Интерфейс взаимодействия с устройством?

Должен быть:

— Простой

— Гибкий

— Желательно готовый

ADB — Android debug bridge

Позволяет:

— выполнять shell команды

— загружать / скачивать файлы

— устанавливать / запускать приложения

— выполнять тач команды

— тонну всего другого...

Язык разработки?

Python

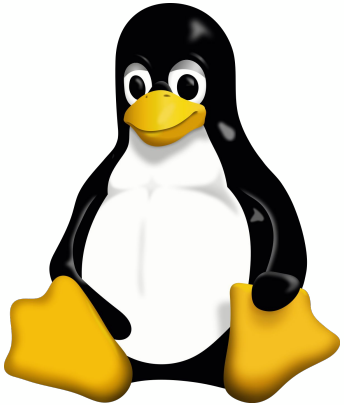
Почему?

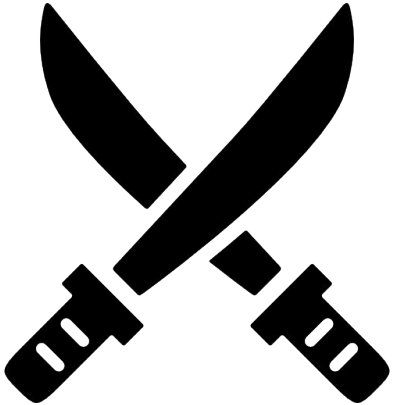
— Высокая скорость разработки

— Разнообразие готовых библиотек

— В тестовой станции нет жестких ограничений по памяти / времени исполнения кода

Операционная система?





Ubuntu

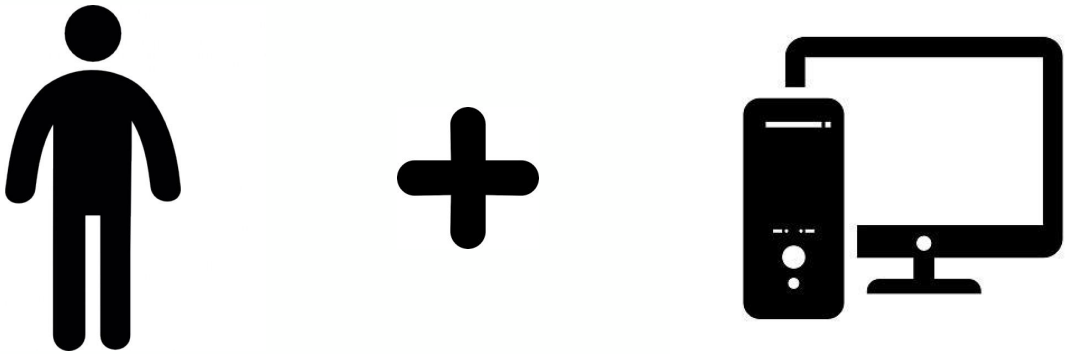
Почему?

— Наличие опыта разработки под Linux и Ubuntu

— Отсутствие опыта разработки под Windows

— Гибкость Linux

Как взаимодействовать с оператором?



Одностраничный landing page в браузере + режим kiosk

Landing page позволяет реализовать простой промышленный интерфейс для оператора

Kiosk:

— Оператор не может использовать никакие другие приложения помимо целевого

— Становится сложнее залезть в наши исходники и отчеты

Landing page

Как раскатывать софт на фабрике?

Рассматривали варианты:

— Докер

— Полноценный Ubuntu образ + флешка

— Накатывать wheels

Ubuntu image + USB флешка

Почему?

— Планируем редкий деплой (раз в 1-2 месяца)

— Возможность переустановить всё под ноль

— Легкость в установке с помощью инженера поддержки на фабрике

Что есть результат работы чембера?

+

=

?

Отчет, сэр... 🧐

Отчет — это текстовое описание результатов + дополнительная метаинформация в удобном формате

Также помимо текста предполагаем хранить сырые данные - звуковые файлы, изображения (метадату)

А как анализировать отчеты? Если их… тысячи?

Elastic + Kibana

— Kibana и Elastic работают в связке из коробки, ничего не надо кодить

— Kibana позволяет строить красивые понятные графики

— В Kibana можно сделать красивые дашборды

Звучит как план...

Технические решения

— Интерфейс взаимодействия с устройством — **adb**

— Язык разработки — **Python**

— ОС — **Ubuntu**

— Интерфейс взаимодействия с оператором — **landing page + kiosk**

— Результат работы чембера — **отчет**

— Визуализация, анализ — **kibana + elastic**

— Зачем нужна all-in-one станция?
— Формируем чек-лист проверок
— Заказываем чембер в Китае
— Вопрос-ответ перед разработкой софта
— **Пишем первый код**
— Знакомство с чембером в Москве
— Пытаемся подружить код и чембер
— Чембера на фабрике в Китае

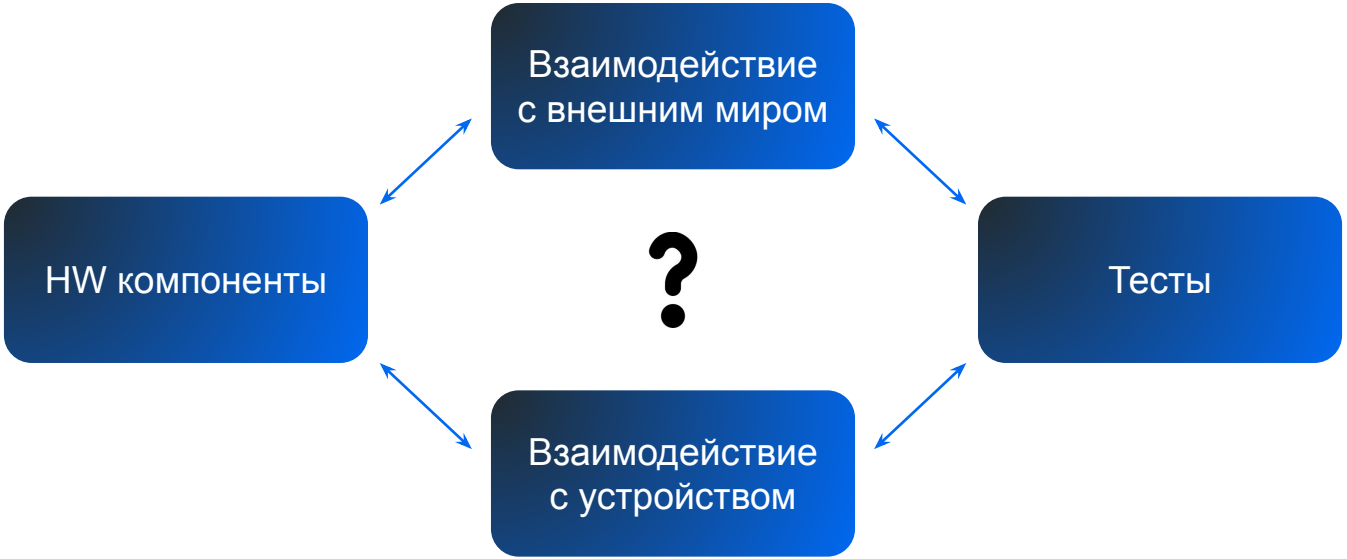
```
1
2
3 if __name__ == "__main__":
4     assert True
5
6
```

Количество кода обычно растет как снежным ком

Бездумно разрабатываемый код быстро становится бардаком

Раскладываем по полочкам

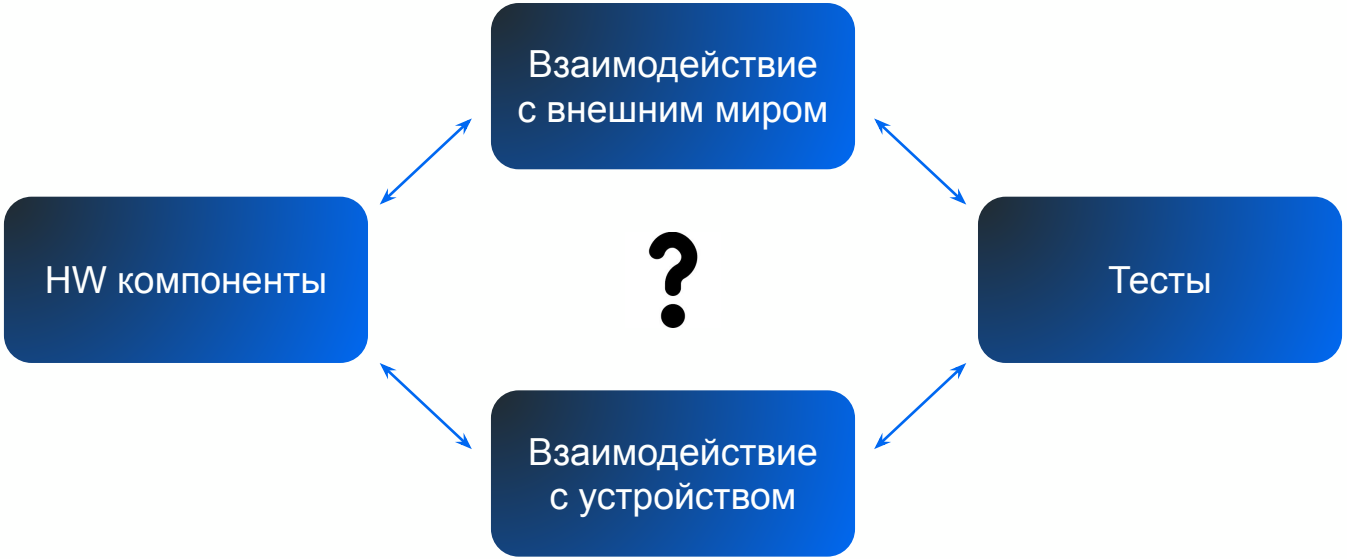
Разделим задачи нашего кода на 4 блока



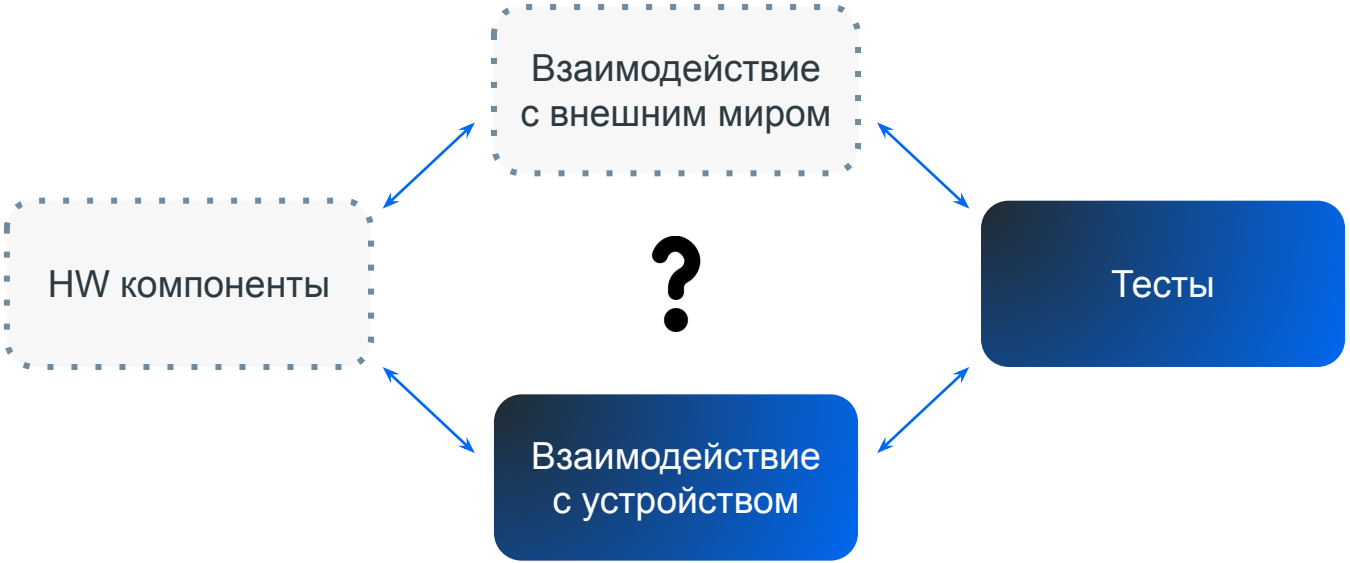
Сейчас сложно ответить на все вопросы

— Мало знаний о продукте (наш продукт — чембер)
— Опыта работы с hardware компонентами еще нет
— Нет полного понимания workflow фабрики
— Этот список можно продолжать до бесконечности...

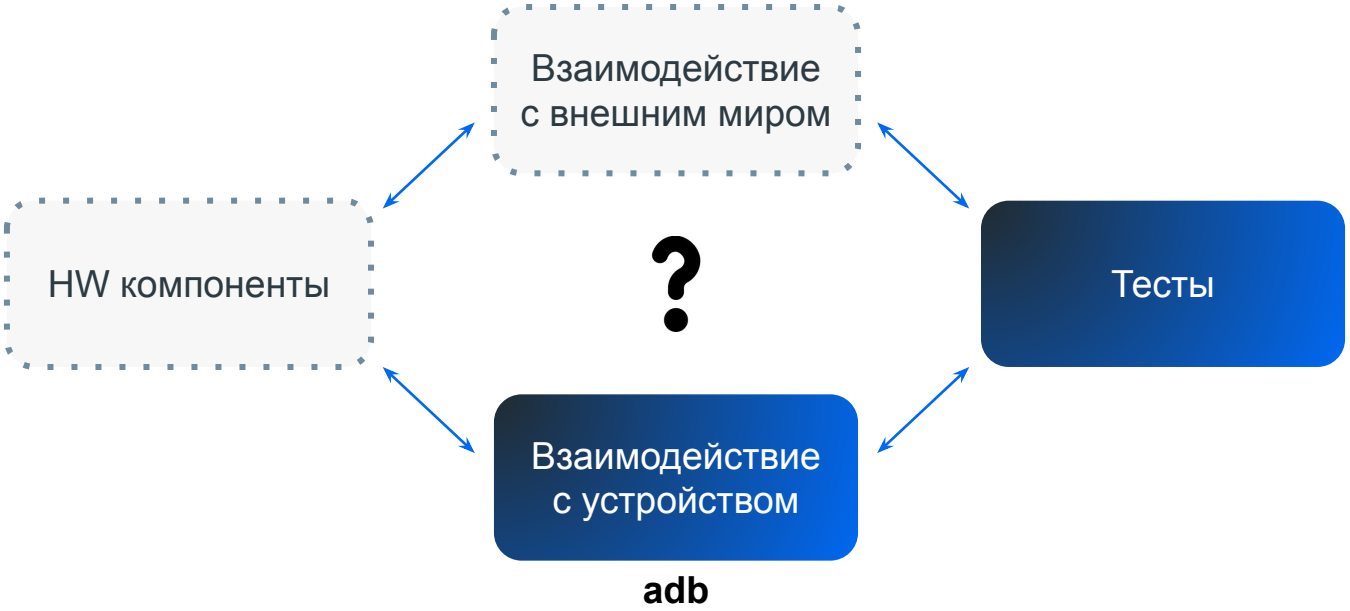
С чего-то надо начинать. Например, с приоритетов



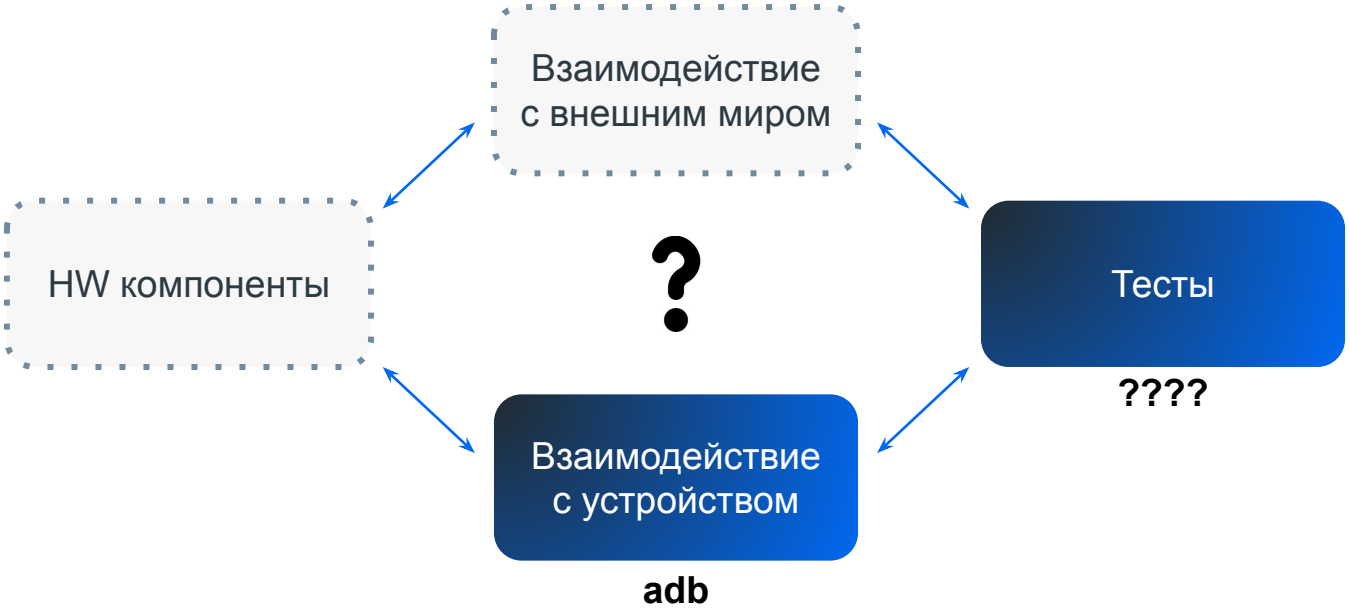
С чего-то надо начинать. Например, с приоритетов



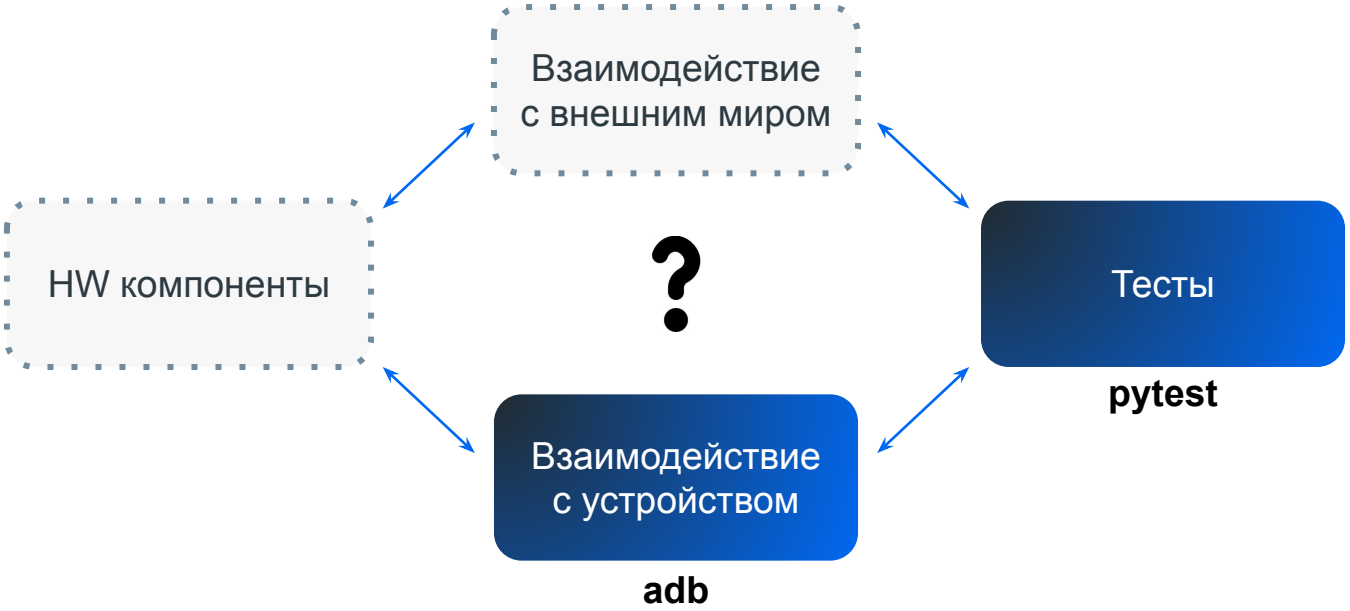
С чего-то надо начинать. Например, с приоритетов



С чего-то надо начинать. Например, с приоритетов



С чего-то надо начинать. Например, с приоритетов



Pytest. Возможности

— Позволяет легко “находить” тесты в коде с помощью декораторов pytest.mark(“...”)

— Разбирается с падением кода в тестах

— Из коробки умеет распараллеливать выполнение

— Формирует текстовый отчет

— И не только…

Pytest: helps you write better programs

Наполняем наш код тестами. Начинаем с малого

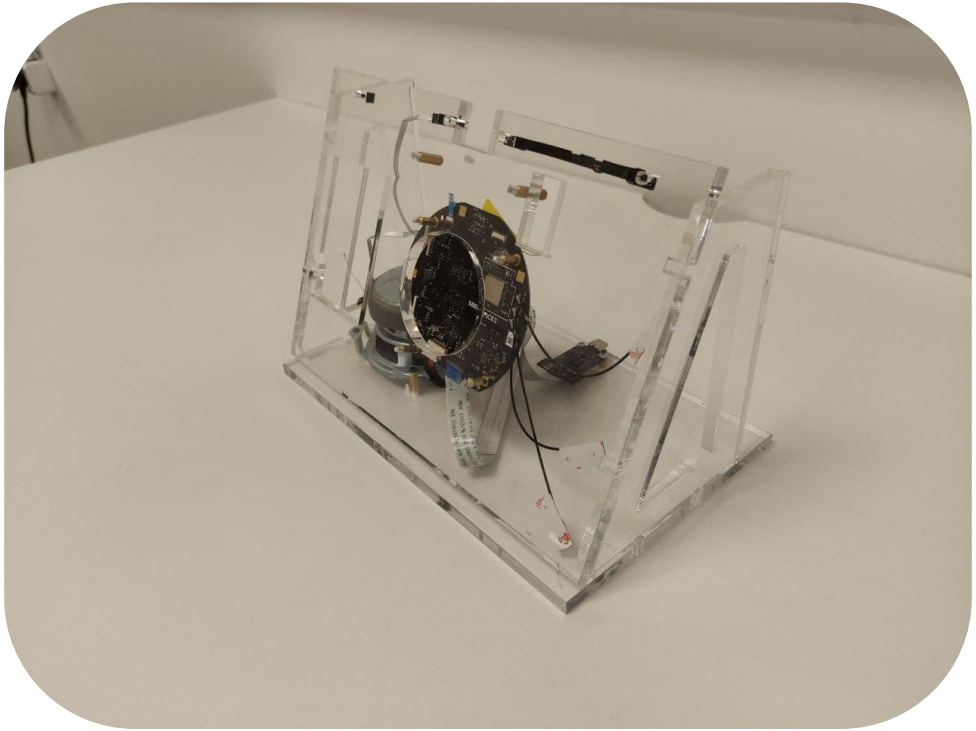
Реализуем простые проверки. Учимся:

— Проверяем версию софта

— Извлекаем системную информацию (мак адрес, id и т.д.)

Тем временем появился первый прототип DUT (Device Under Test)

На самом деле он был такой 🙈



Учимся записывать / проигрывать звук через adb

Осваиваем работу с камерой

Научились

— Проигрывать, записывать звук

— Фотографировать камерой

— Выводить изображение на экран

— Работать с Wi-Fi

— И все это с помощью adb и готовых инструментов — Open Camera, tinycap, tinyplay и т.д.

Но есть проблема...

Когда ты 100 раз за день послушал белый шум через динамик

Очень неудобно, поэтому добавляем несколько сущностей

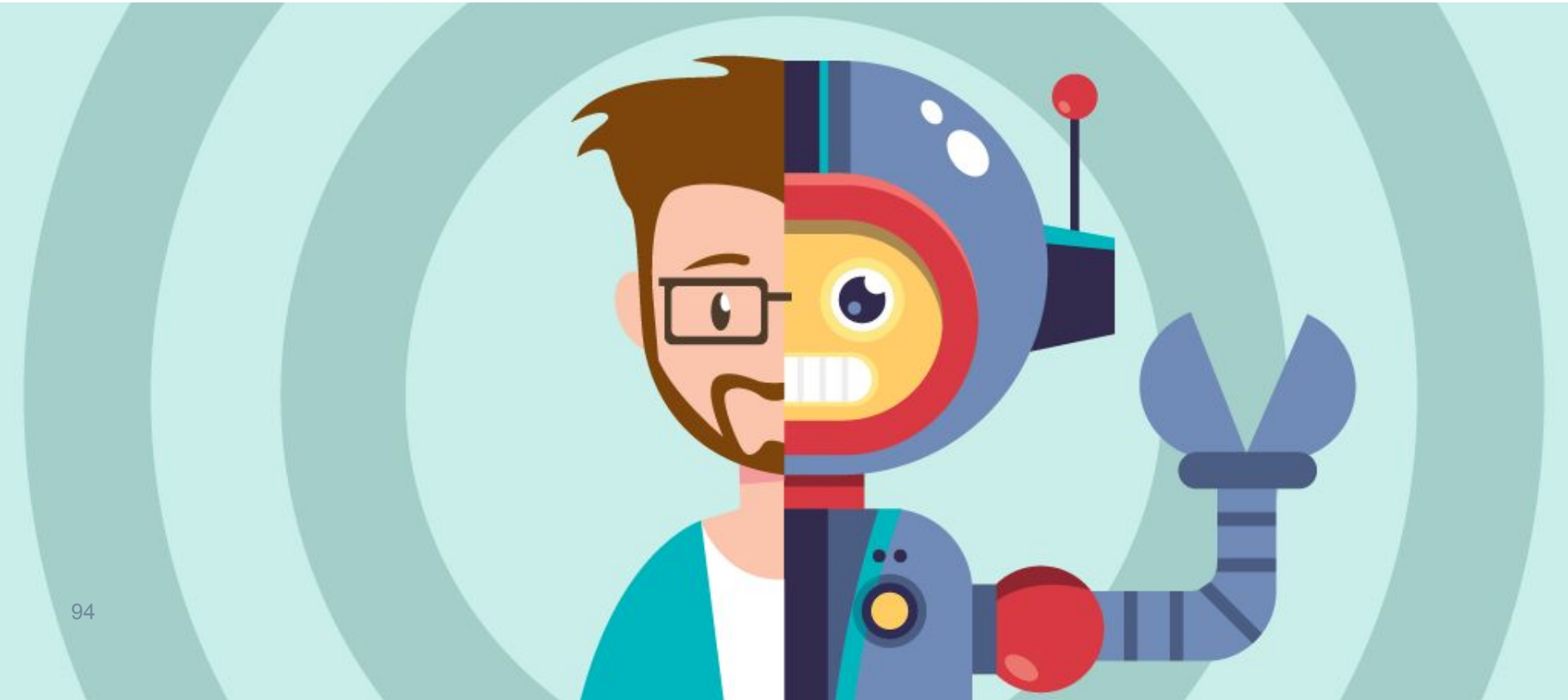
— **Конфиги** — описание используемых hardware компонентов

— **Тест сьюты** — набор запускаемых тестов в виде json

```
{
  "tests": {
    "test1": true,
    "test2": false,
    "test3": true,
    "test4": false
  }
}
```

```
@mark.test1
def test_do_something():
    pass


@mark.test1
def test_do_something_again():
    pass


@mark.test2
def test_do_another_thing():
    pass
```

Автоматизировать или не автоматизировать?



Разделяем тесты

Ручные

Автоматизированные

Примеры тестов

— **Ручные** — проверка кнопок, проверка корректности тача экрана

— **Автоматизированные** — подключение к точке доступа Wi-Fi, проверка камеры, оценка качества динамика, микрофонов и т.д.

Автоматизированные проверки выполняются в несколько процессов для ускорения, ручные — всегда последовательно.

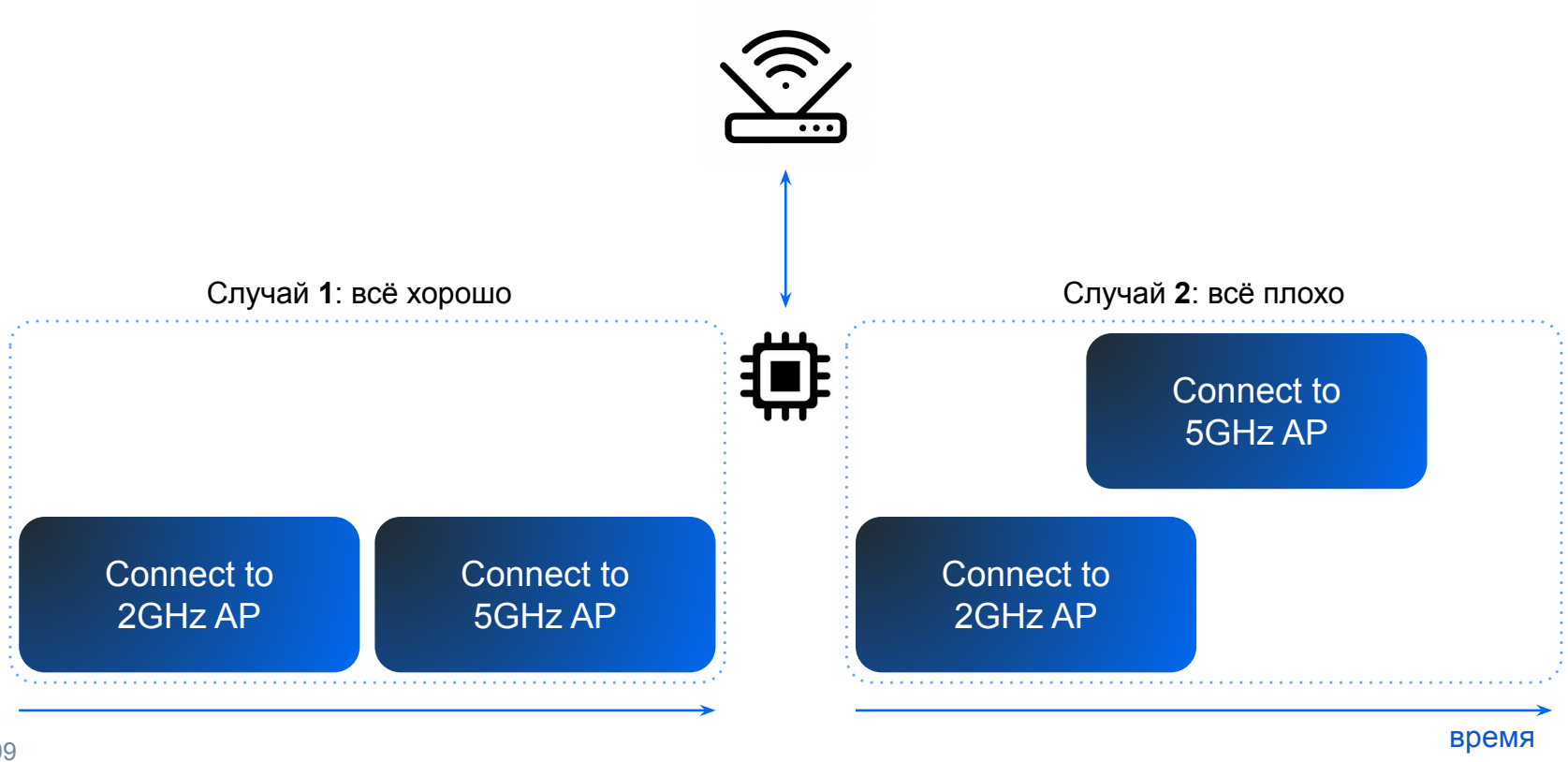
Flaky тесты?

Например, такое встречалось с тестами Wi-Fi

```
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0 -- /home/afomin/projects/chamber/venv/bin/python
cachedir: .pytest_cache
rootdir: /home/afomin/projects/chamber/star_chamber, configfile: pytest.ini
collected 4 items

star_chamber/star_chamber/test_wifi.py::test_wifi_rssi[2] PASSED          [ 25%]
star_chamber/star_chamber/test_wifi.py::test_wifi_rssi[5] PASSED          [ 50%]
star_chamber/star_chamber/test_wifi.py::test_wifi_connect[2] FAILED       [ 75%]
star_chamber/star_chamber/test_wifi.py::test_wifi_connect[5] FAILED       [100%]
```

```
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0 -- /home/afomin/projects/chamber/venv/bin/python
cachedir: .pytest_cache
rootdir: /home/afomin/projects/chamber/star_chamber, configfile: pytest.ini
collected 4 items

star_chamber/star_chamber/test_wifi.py::test_wifi_rssi[2] PASSED          [ 25%]
star_chamber/star_chamber/test_wifi.py::test_wifi_rssi[5] PASSED          [ 50%]
star_chamber/star_chamber/test_wifi.py::test_wifi_connect[2] PASSED       [ 75%]
star_chamber/star_chamber/test_wifi.py::test_wifi_connect[5] PASSED       [100%]
```

Почему так?



Наше разделение несовершенно… Чиним

Ручные

Автоматизированные

Параллельные

Последовательные

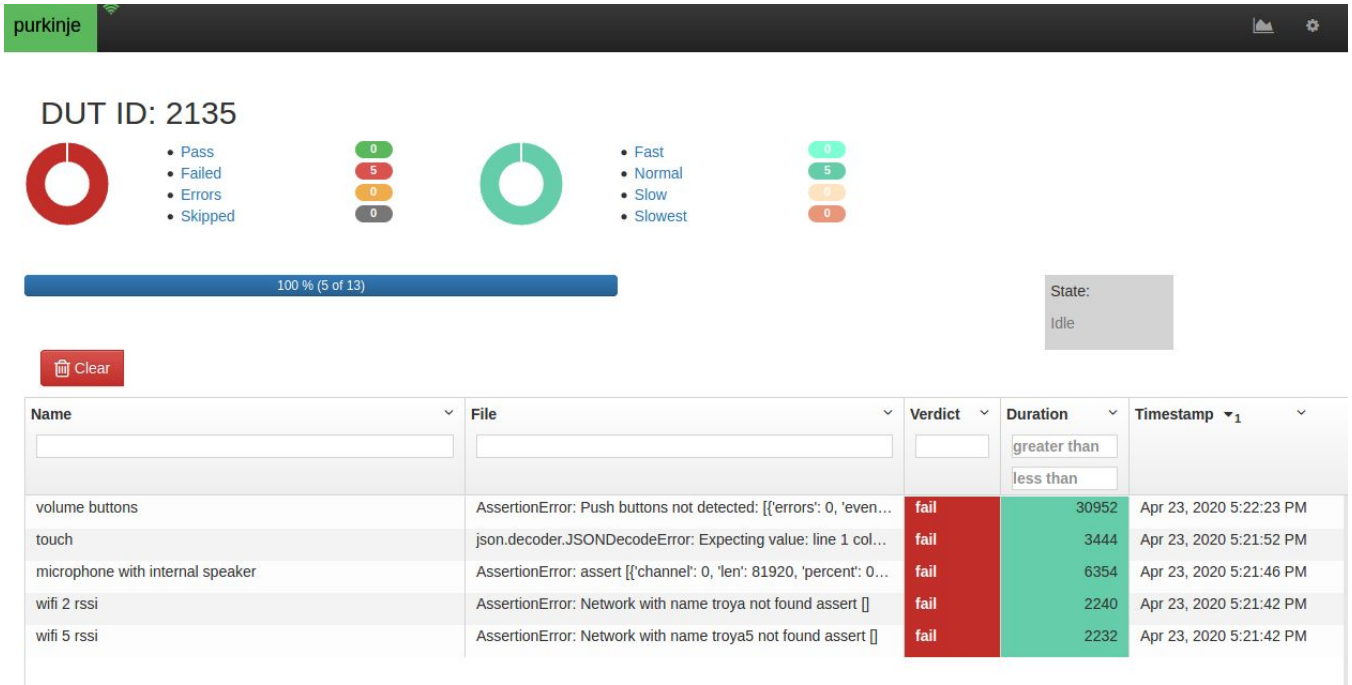
Визуализация

— Основной приоритет сейчас — расширение тестового покрытия
— Однако визуализации тоже нужно уделить время
— Вкладывать много ресурсов в визуализацию нет возможности
— Ищем готовое решение

Визуализация сейчас

```
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0 -- /home/afomin/projects/chamber/venv/bin/python
cachedir: .pytest_cache
rootdir: /home/afomin/projects/chamber/star_chamber, configfile: pytest.ini
collected 15 items

star_chamber/star_chamber/test_display.py::test_display_pixels[red] PASSED      [  6%]
star_chamber/star_chamber/test_display.py::test_display_pixels[green] PASSED    [ 13%]
star_chamber/star_chamber/test_display.py::test_display_pixels[blue] PASSED     [ 20%]
star_chamber/star_chamber/test_sound.py::test_speaker_THD PASSED                [ 26%]
star_chamber/star_chamber/test_sound.py::test_speaker_FR PASSED                 [ 33%]
star_chamber/star_chamber/test_sound.py::test_mic_gain[1] PASSED                [ 40%]
star_chamber/star_chamber/test_sound.py::test_mic_gain[2] PASSED                [ 46%]
star_chamber/star_chamber/test_sound.py::test_mic_gain[3] PASSED                [ 53%]
star_chamber/star_chamber/test_sound.py::test_mic_gain[4] PASSED                [ 60%]
star_chamber/star_chamber/test_sound.py::test_mic_gain[5] PASSED                [ 66%]
star_chamber/star_chamber/test_sound.py::test_mic_gain[6] PASSED                [ 73%]
star_chamber/star_chamber/test_wifi.py::test_wifi_rssi[2] PASSED                [ 80%]
star_chamber/star_chamber/test_wifi.py::test_wifi_rssi[5] PASSED                [ 86%]
star_chamber/star_chamber/test_wifi.py::test_wifi_connect[2] PASSED             [ 93%]
star_chamber/star_chamber/test_wifi.py::test_wifi_connect[5] PASSED             [100%]

============================== 15 passed in 200.22s (0:03:20) ==============================
```

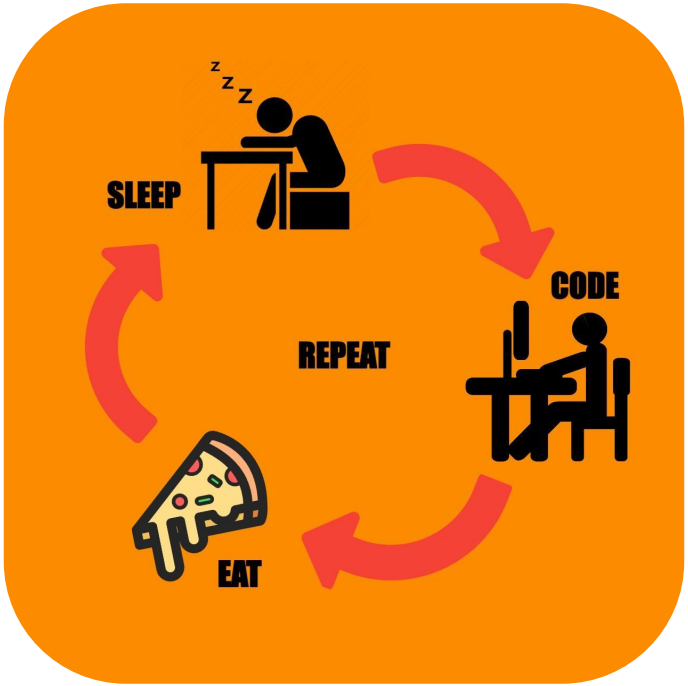
Purkinje



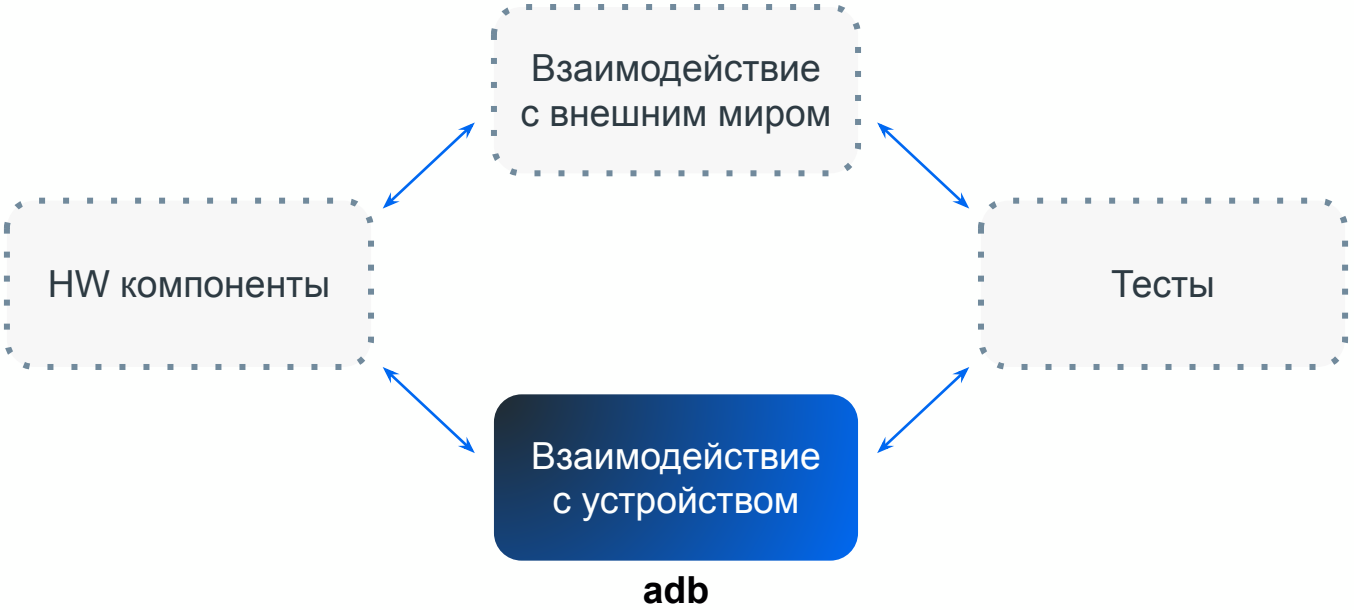
Purkinje

— Готовое решение
— Бесшовно стыкуется с pytest
— Отображает progress bar, результаты тестов, ошибки
— Закрывает наши текущие потребности

Продолжаем итеративно наращивать тестовое покрытие



Все наши тесты по-прежнему используют adb



adb — так ли он хорош?

— Мы вынуждены ждать загрузки девайса в Android Загрузка в Android — это дополнительное время

— Некоторые вещи неудобно делать через adb и это становится квестом — запустить приложение, нажать на экран в такую-то точку, скачать картинку

— adb — очень крутой инструмент, над разработкой которой работают лучшие инженеры

— Как и любой инструмент, у него есть плюсы и минусы

Чего нам не хватает?

— Не хочется ждать загрузки девайса в Android

— Единообразный интерфейс взаимодействия с устройством

— Желательно, чтобы он ещё и сохранялся, когда будут следующие девайсы

SDTS — Sber Devices Test Suite

Факты:

— Образ Linux, в котором есть только то, что нам нужно

— Чтобы загрузить SDTS образ в девайс, не нужно ждать загрузки андроида (fastboot boot sdts.image)

— Все взаимодействие с устройством происходит через http-ручки

— Процесс разработки полностью контролируется нашей R&D hardware командой

SDTS — выигрываем время

Также SDTS действительно позволяет экономить немного времени на загрузке:

— adb в среднем грузится **за 22 сек**

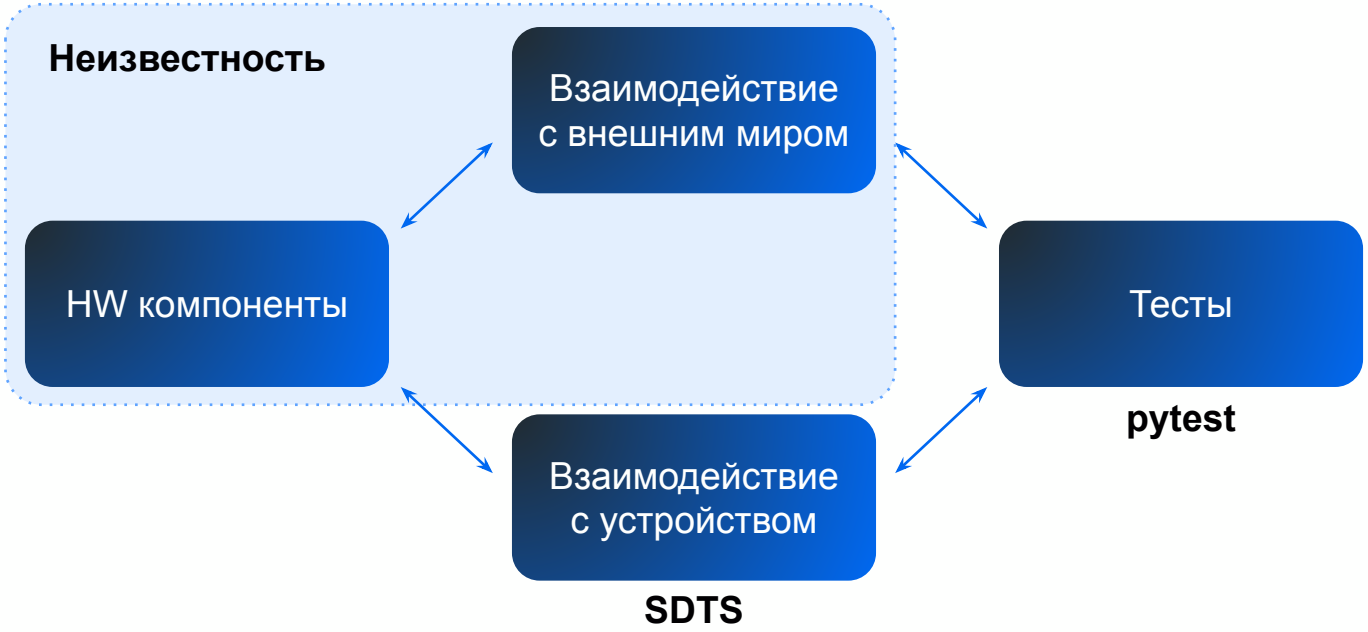
— sdts — **за 14 сек**

Кажется, что это немного, но попробуем оценить на примере из 10 000 устройств:

(22 – 14) x 10 000 / 3 600 = 22 часа чистого времени

В деньгах это немаленькие цифры

Картина мира



Технические решения

— Для контроля тестов мы решили использовать Pytest

— Разделили проверки на 3 категории — manual, automated background, foreground

— Ввели json описание текущей конфигурации оборудования — config, описание тестов — test suite

— Purkinje — веб-интерфейс из коробки

— К нам приехал первый прототип тестируемого устройства

— Перешли от adb к SDTS

— Зачем нужна all-in-one станция?
— Формируем чек-лист проверок
— Заказываем чембер в Китае
— Вопрос-ответ перед разработкой софта
— Пишем первый код
— **Знакомство с чембером в Москве**
— Пытаемся подружить код и чембер
— Чембера на фабрике в Китае

Знакомство с чембером в Москве

```
1 <Hello chamber/>
```

Софт готов, но где же Чембер?

— 2 м^3 объема

— 800 кг веса

— Хрупкость внутренней конструкцией

¯_(ツ)_/¯

Знакомство с чембером в Москве

Знакомство с чембером в Москве

Первое знакомство

— Обнаружили много проблем в качестве исполнения и компонентов

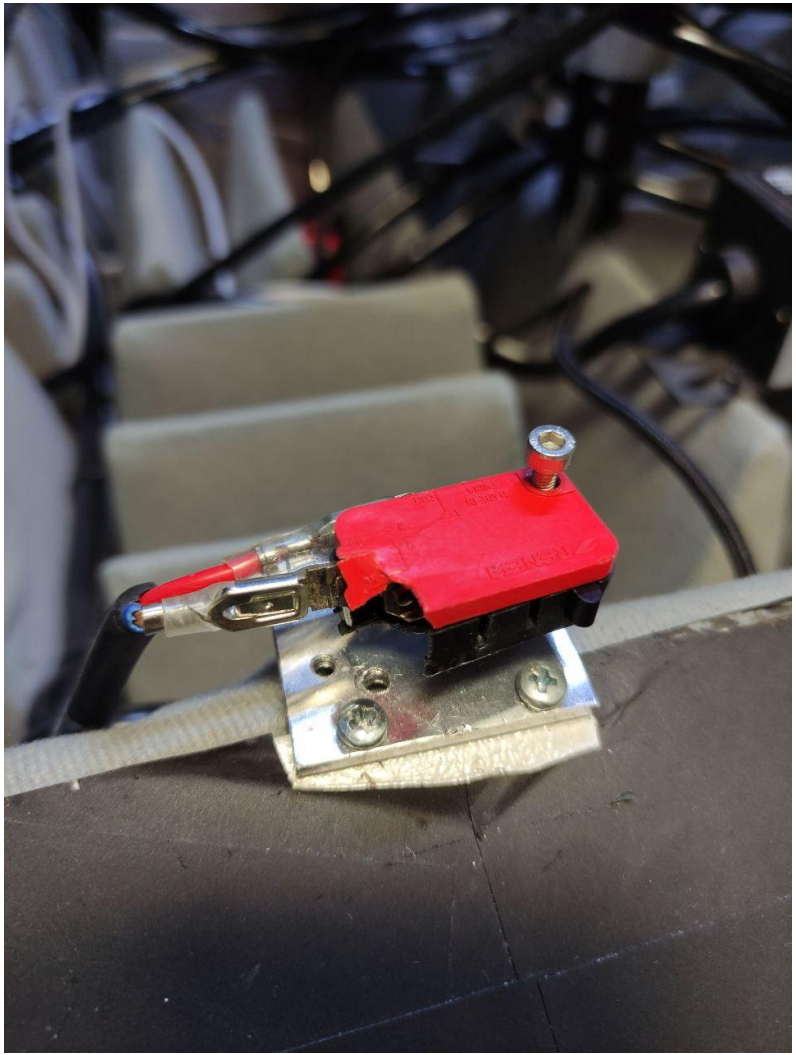
— **СЮРПРИЗ**, но без компрессора, основные зажимы Чембера не работали

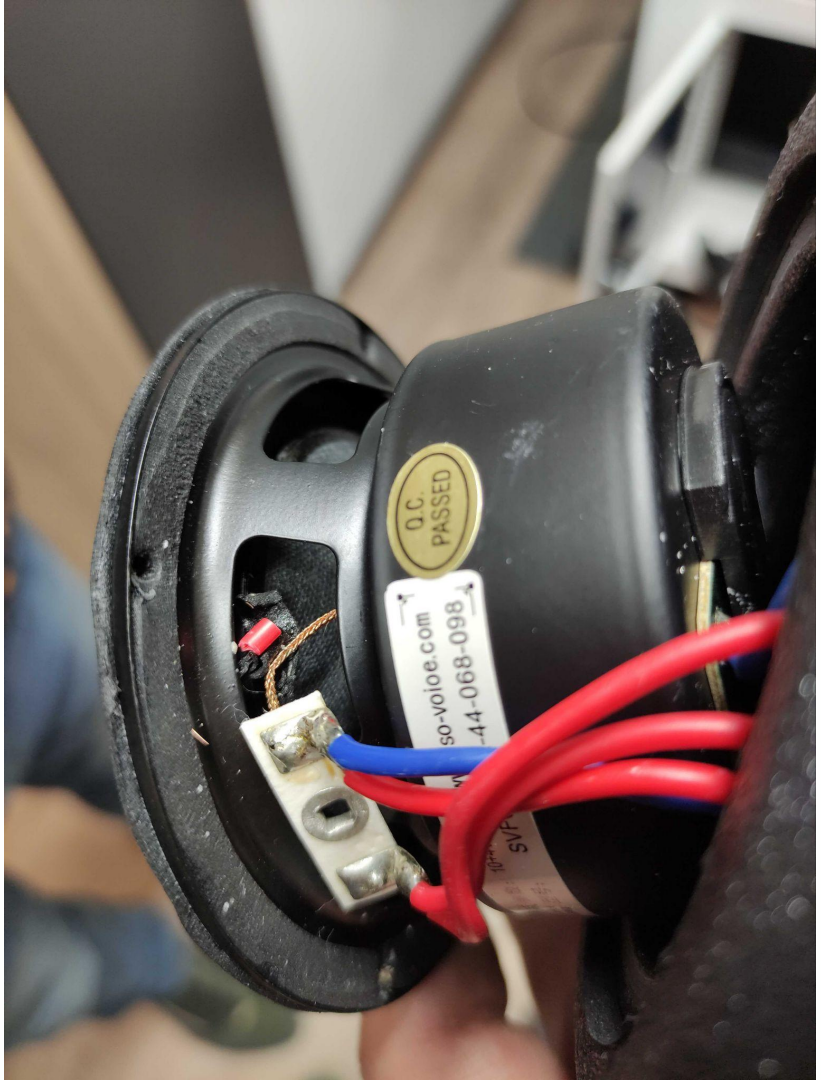
— Все провода были неправильно подключены и абсолютно никак не маркированы…

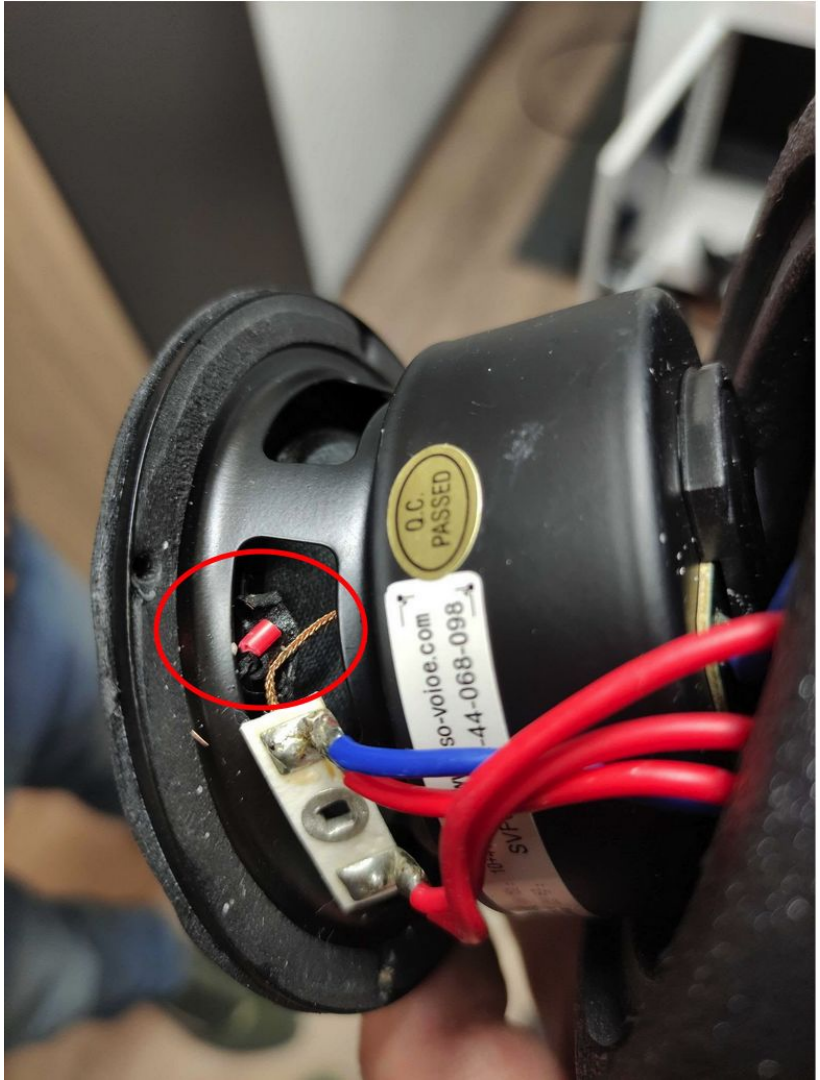
— Посредственное качество комплектующих

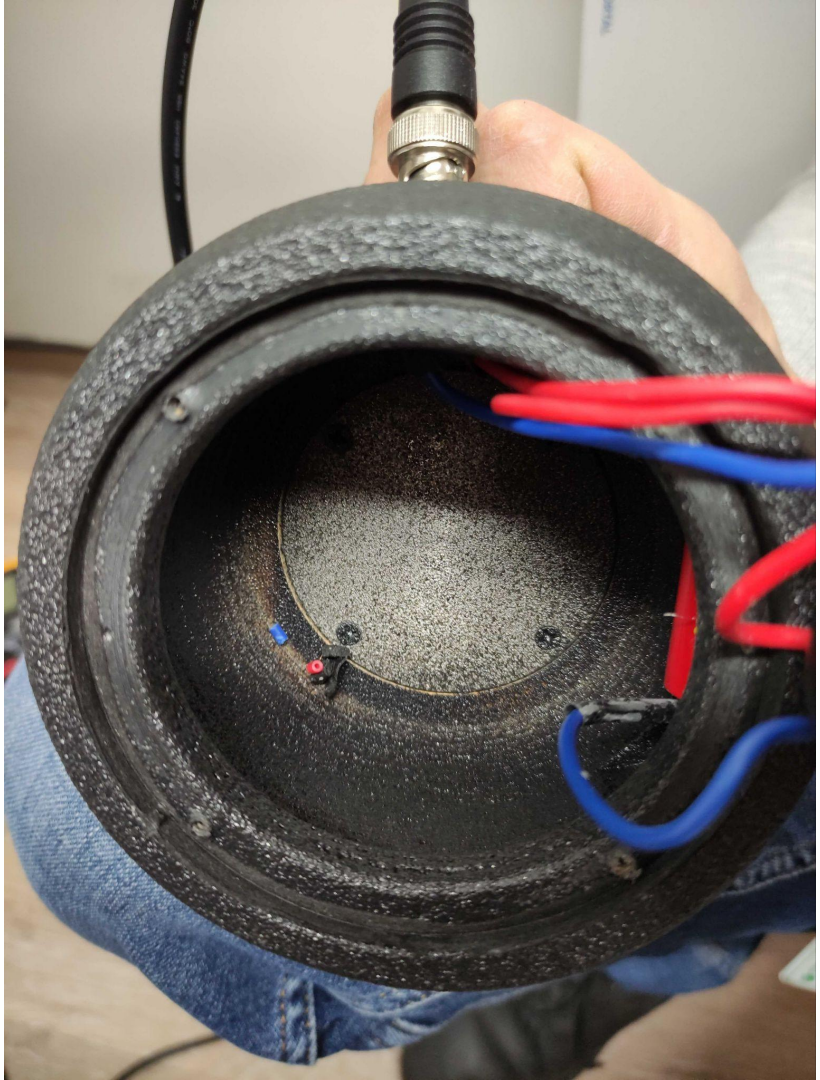
Знакомство с чембером в Москве

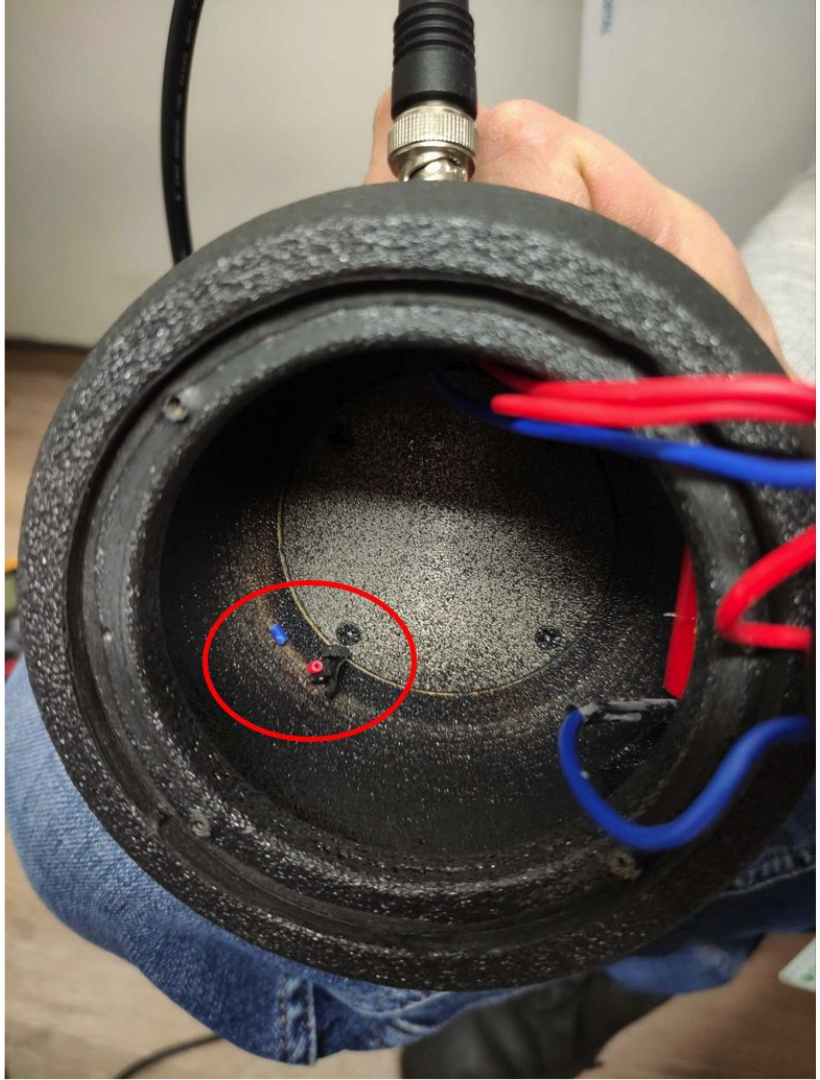
Знакомство с чембером в Москве









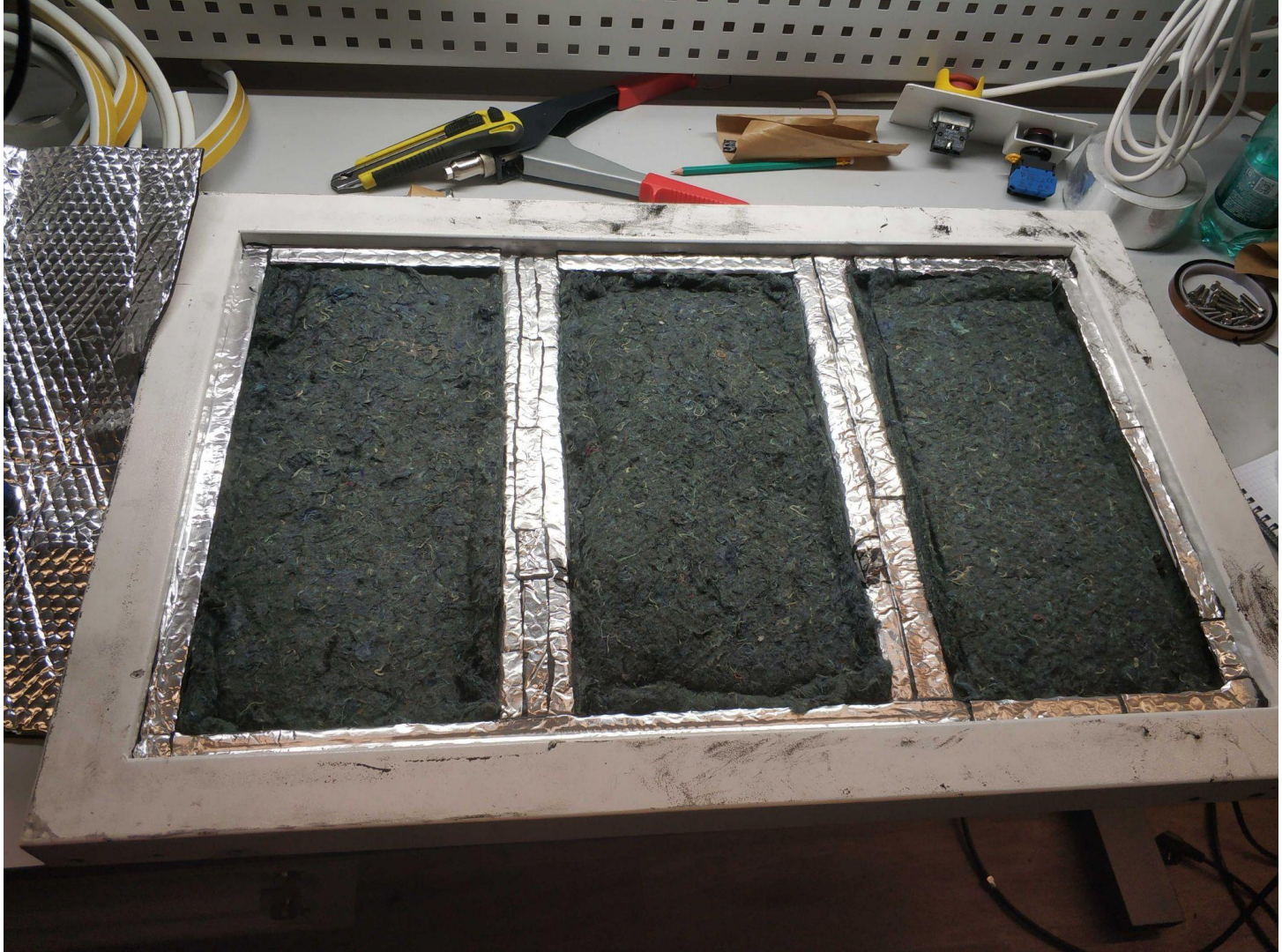


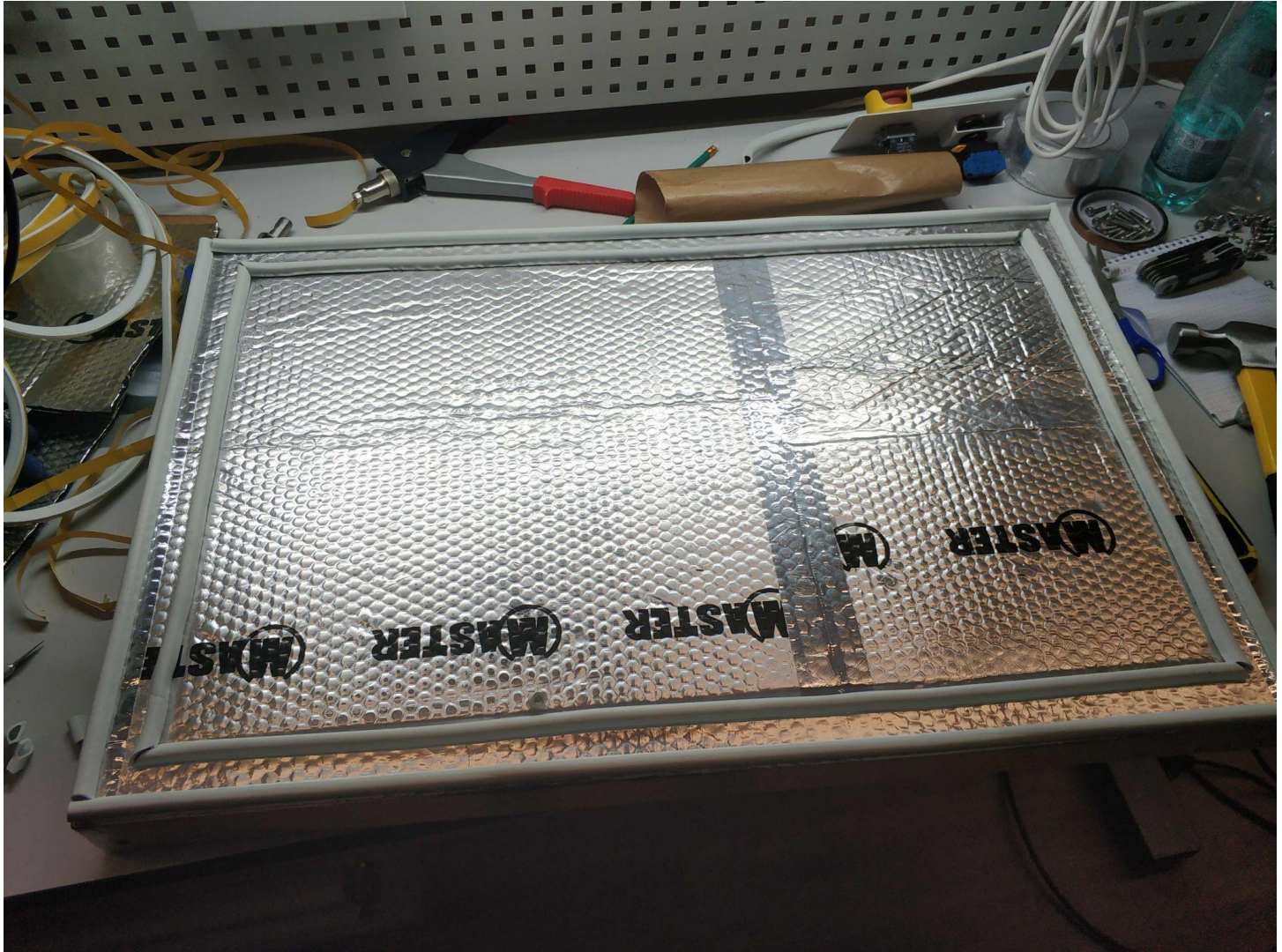
Вы можете это исправить?...

WE CAN DO IT!

💪

Знакомство с чембером в Москве





Выводы?

— Никому нельзя доверять, тем более при удаленной работе

— Фиксируйте все договоренности письменно

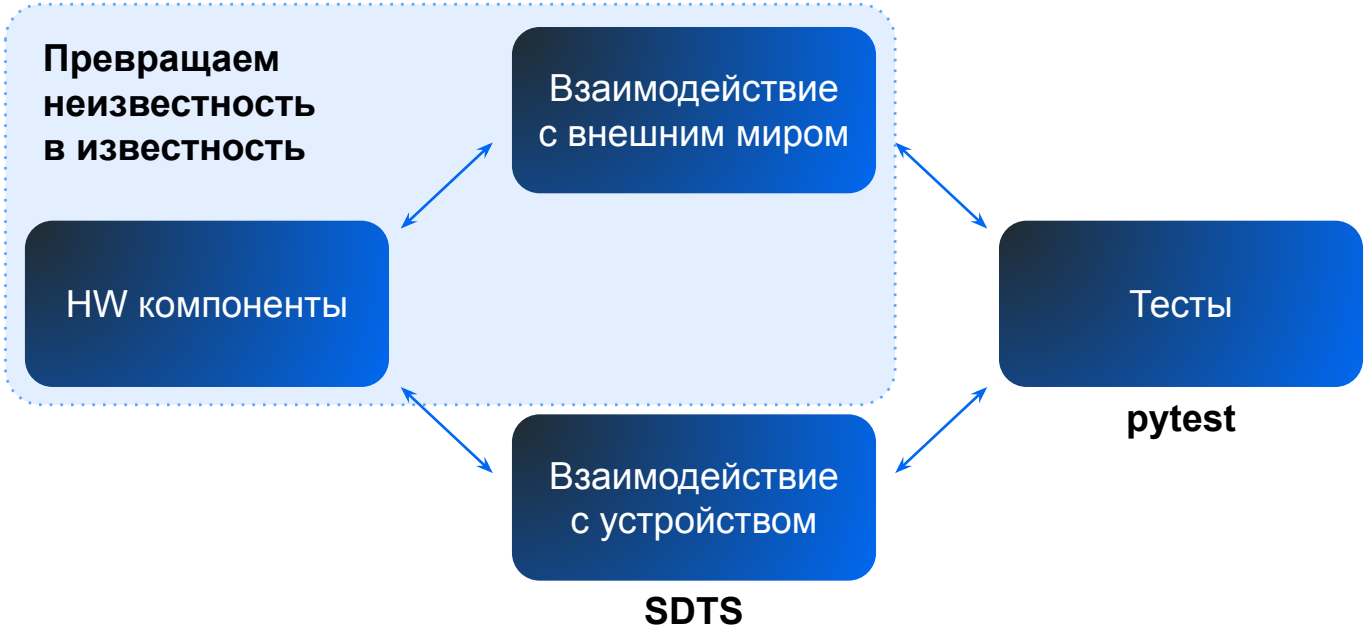
— Любую проблему можно исправить (если очень захотеть)

— Зачем нужна all-in-one станция?
— Формируем чек-лист проверок
— Заказываем чембер в Китае
— Вопрос-ответ перед разработкой софта
— Пишем первый код
— Знакомство с чембером в Москве
— **Пытаемся подружить код и чембер**
— Чембера на фабрике в Китае

Пытаемся подружить код и чембер

🤝

Пишем обвязку для HW компонентов и учимся работать с контроллером чембера

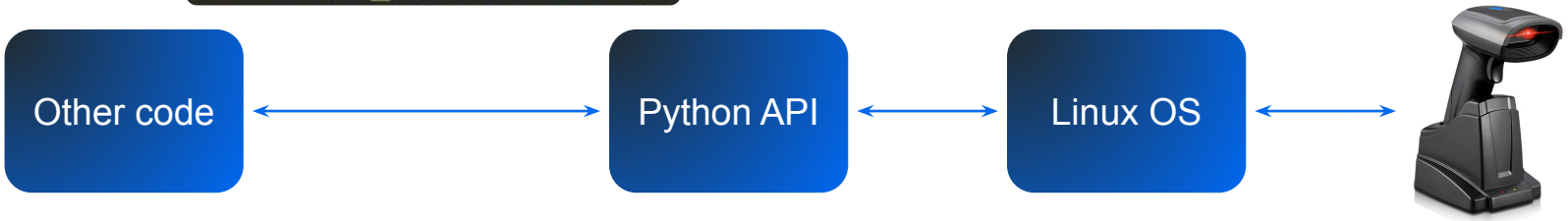


Пытаемся подружить код и чембер

Обвязка для HW компонентов

на примере сканера штрихкода

```
def check_barcode(self):
```

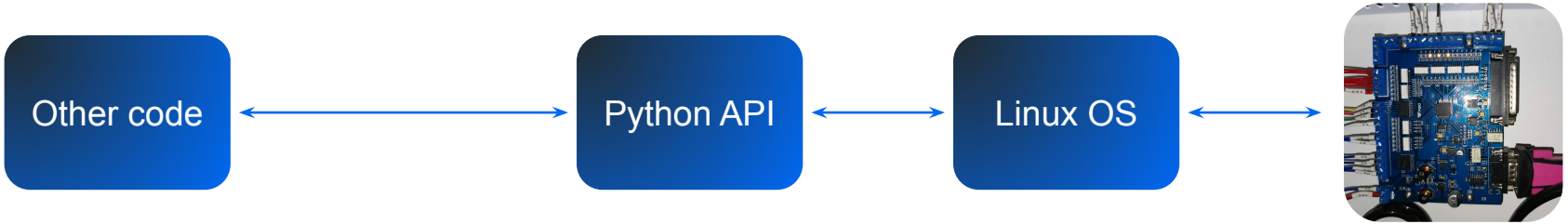


Взаимодействие с внешним миром

Простейший интерфейс:

— Запрос текущего состояния

— Перевести чембер в состояние X



Оно работает!

Но этого недостаточно 😔

Стресс-тест — гарантия качества

Требования к чемберу:

— Работоспособность 24/7

— Неубиваемость вне зависимости от действий оператора

SBER DEVICES

Итоги стресс-теста

Пытаемся подружить код и чембер

Итоги стресс-теста

— Наш софт работает только когда всё идет “по плану”

— В случае отклонения от плана всё безвозвратно ломается

— Отдавать софт в текущем виде на фабрику нельзя

Почему так происходит? 🤔

Это наш код

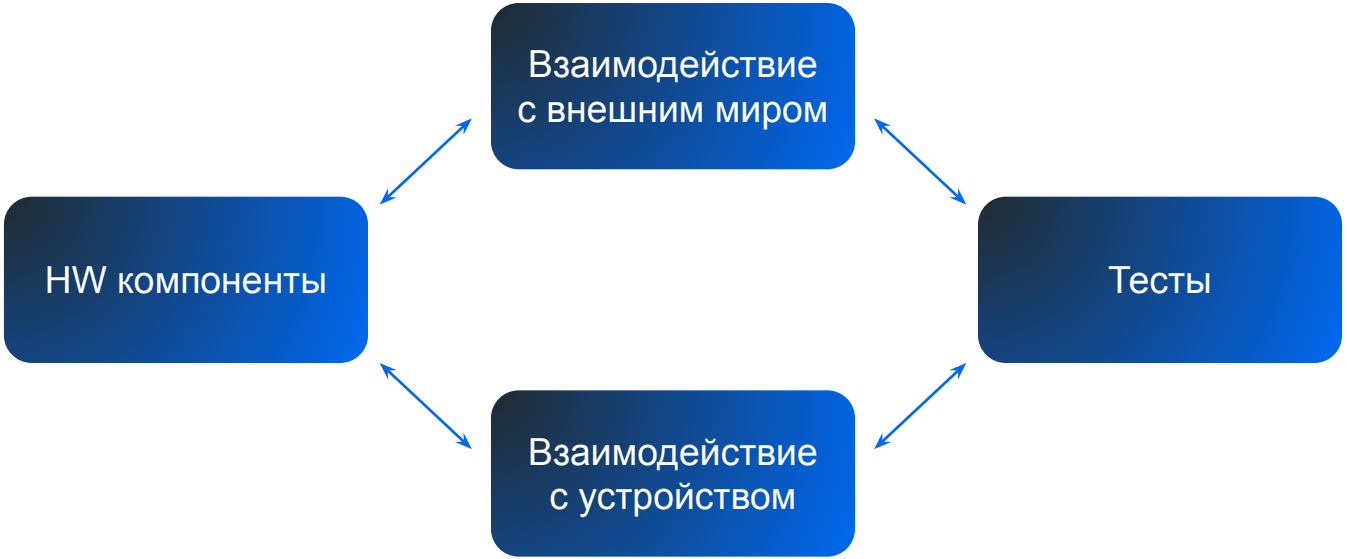
На слайде ничего не видно.
Но его и не нужно видеть.

Он — ужасен.

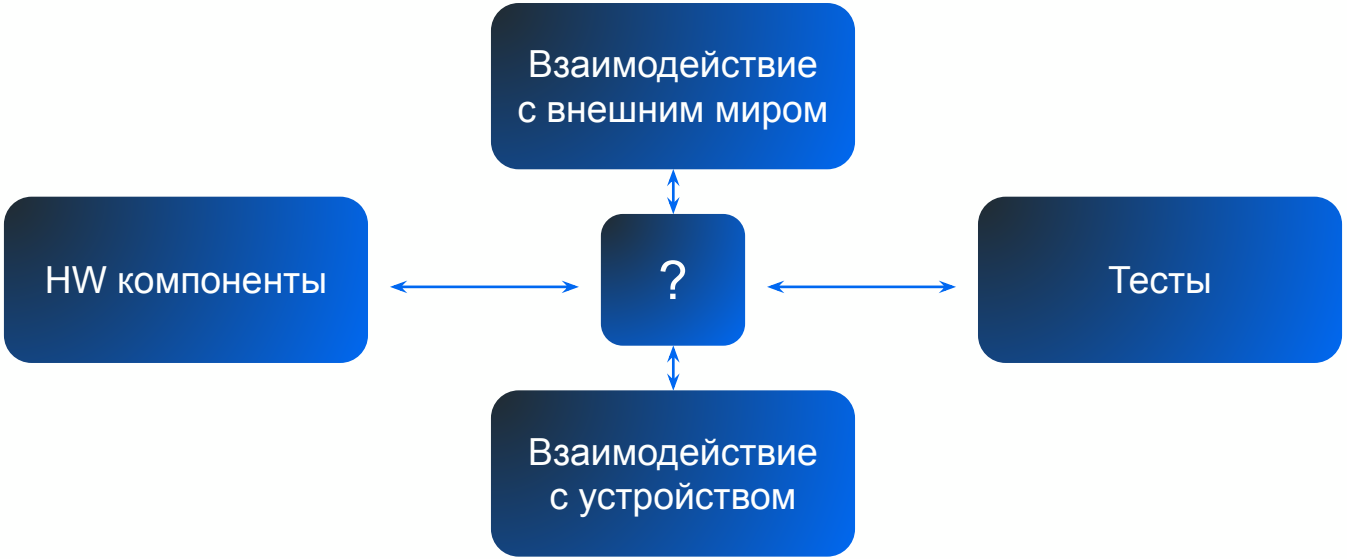
Пытаемся подружить код и чембер

```
    while True:
        try:

            barcode = self.get_barcode()
            start_test_timestamp = datetime.datetime.now().timestamp()
            folder_name = "{}_{}".format(start_test_timestamp, barcode)
            test_report_path = "/tmp/reports/{}".format(folder_name)
            full_report_path = "{}/full_report_{}.json".format(test_report_path, barcode)
            logger.info('create test report folder')
            Path(test_report_path).mkdir(parents=True, exist_ok=True)
            logger.info('Activate webgui test session')
            self.start_test_session(raw_test_information, barcode)

            if self.config['logging']['save_chamber_log_in_report']:
                test_run_handler = logging.FileHandler(test_report_path + "/chamber.log")
                test_run_handler.setLevel(self.config['logging']['level'])
                if self.config['logging']['ignore_list']:
                    test_run_handler.addFilter(LogFilter(self.config['logging']['ignore_list']))
                root_logger.addHandler(test_run_handler)

            for component in self.components:
                self.components[component].init_testrun()

            self.run_pre_testrun_checks()
            self.check_report_size_check()

            for component in self.components:
                self.components[component].wait_ready()

            manager = multiprocessing.Manager()
            reports_proxy = manager.list()

            tests_args = [barcode, test_report_path, reports_proxy]
            if 'chamber_controller' in self.components:
                tests_args.append(self.components["chamber_controller"].chamber_controller.port)


            process = multiprocessing.Process(target=self.run_tests, args=tuple(tests_args))
            process.start()

            while process.is_alive():
                try:
                    for i in self.testrun_checks:
                        func = i[0]
                        message = i[1]

                        if not func():
                            process.terminate()
                            logger.error(message)
                            raise Exception(message)
                    time.sleep(1)
                except Exception as e:
                    process.terminate()
                    raise e

            reports = []
            reports.extend(reports_proxy)
            manager.shutdown()

            report = self.merge_reports(reports)
            if 'power_supply' in self.components:
                report['power_consumption'] = self.components['power_supply'].get_report()
                webgui_client = WebSocketClient()
                webgui_client.put_test("Power Consumption", Status.Passed, "", 0)
            with open(full_report_path, mode="w") as file:
                file.write(json.dumps(report, indent=2))
                logger.debug('full log created')

            if self.config['ftp']['enable']:
                shutil.move(test_report_path, Path(self.config['ftp']['report_folder']).joinpath(folder_name))
                logger.debug('report uploaded')
            logger.info('testrun time: {}'.format(datetime.datetime.now().timestamp() - start_test_timestamp))

            self.save_sdts_logs(test_report_path)

        except StarChamberLoaderException as e:
            logger.error(e)
            continue
        except Exception as e:
            logger.critical(e)
        finally:
            if 'chamber_controller' in self.components:
                try:
                    self.components['chamber_controller'].chamber_controller.door_open()
                except Exception as e:
                    logger.error(e)
            if self.config['logging']['save_chamber_log_in_report']:
                for i in range(len(root_logger.handlers)):
                    try:
                        if root_logger.handlers[i] is test_run_handler:
                            root_logger.handlers.pop(i)
                            test_run_handler.close()
                            break
                    except UnboundLocalError:
                        pass
```

Более предметно



Как должно быть?



Что за квадратик с вопросом?

Он должен:

— Коммутировать взаимодействие с другими блоками

— Быть достаточно простым и надежным

Стейт машина

— Описываем набор возможных состояний чембера

— А также переходы между состояниями

Стейт машина

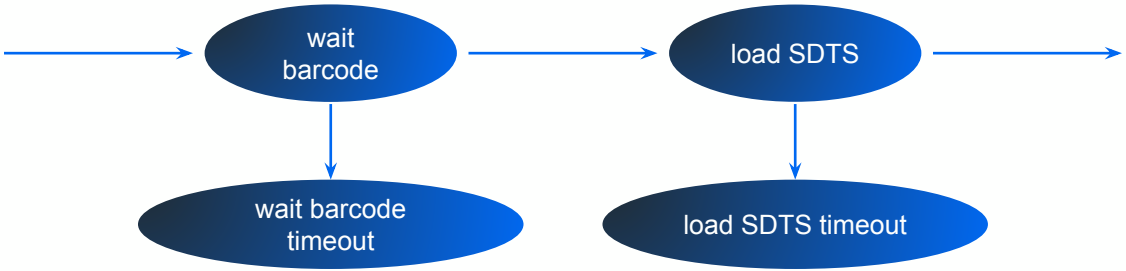
— Описываем набор возможных состояний чембера

— А также переходы между состояниями

Стейт машина

— Описываем набор возможных состояний чембера

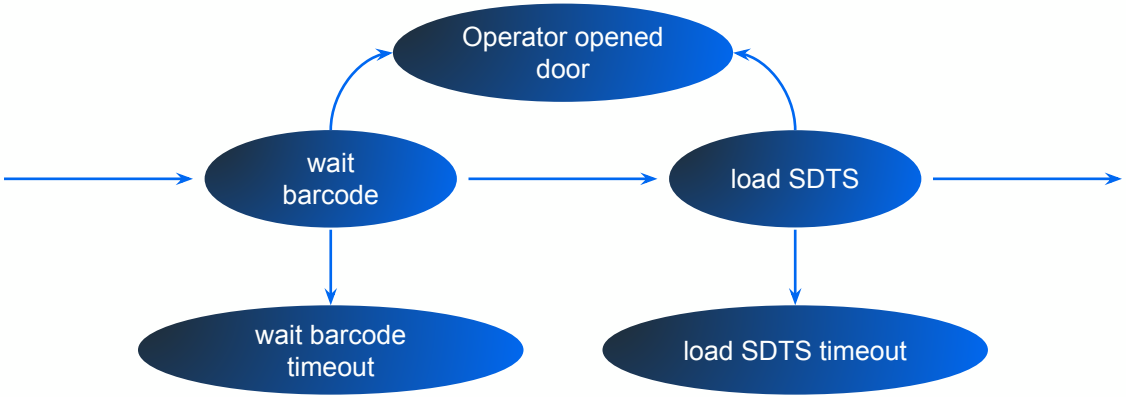
— А также переходы между состояниями



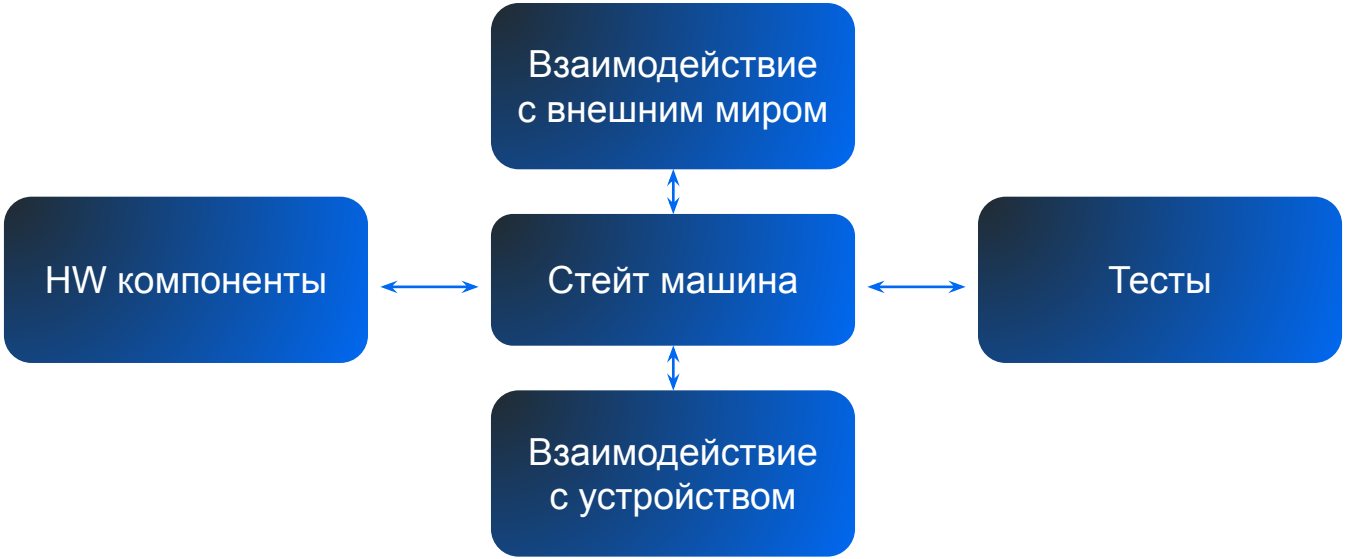
Стейт машина

— Описываем набор возможных состояний чембера

— А также переходы между состояниями



Планируем сделать так



Прошло 2 недели ⏳

Стейт машина готова

Количество стейтов — 60

Пытаемся подружить код и чембер

Оно снова работает!

— Теперь точно работает

— Стресс-тест проходится на ура!

— Любая нештатная ситуация переводит стейт-машину в соответствующий state

Вот только откуда оператор чембера узнает текущий стейт?

Нам уже не хватает возможностей Purkinje

— Мы хотим показывать оператору текущий статус

— А еще мы хотим показывать оператору историю статусов

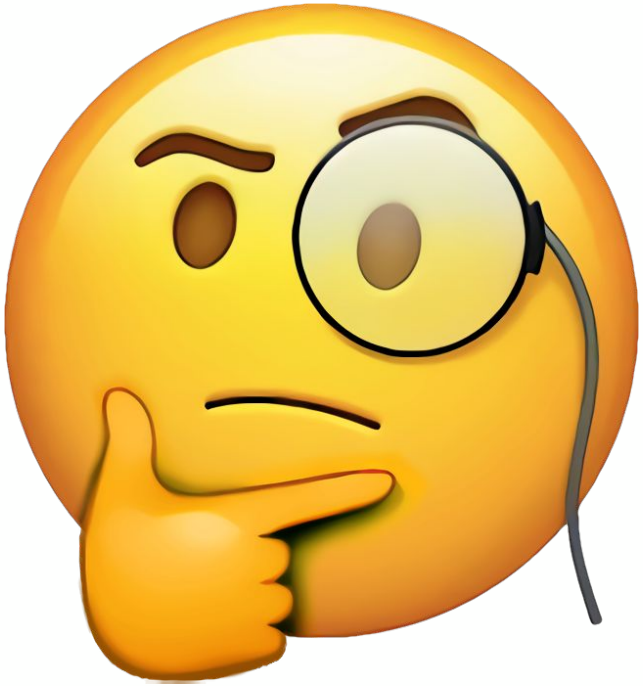
— А еще ошибки

— А еще warnings

— А еще мы хотим добавить версию софта

— А в будущем хотим управлять конфигурацией нашего кода из веб-интерфейса

Что будем делать?



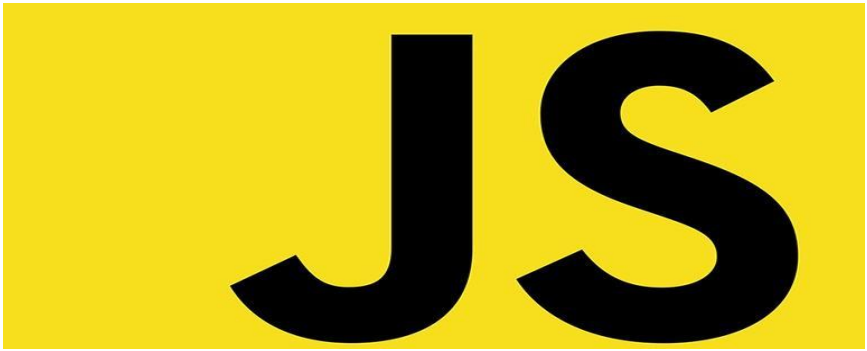
Javascript для начинающих

— Делаем простой интерфейс на чистом Javascript

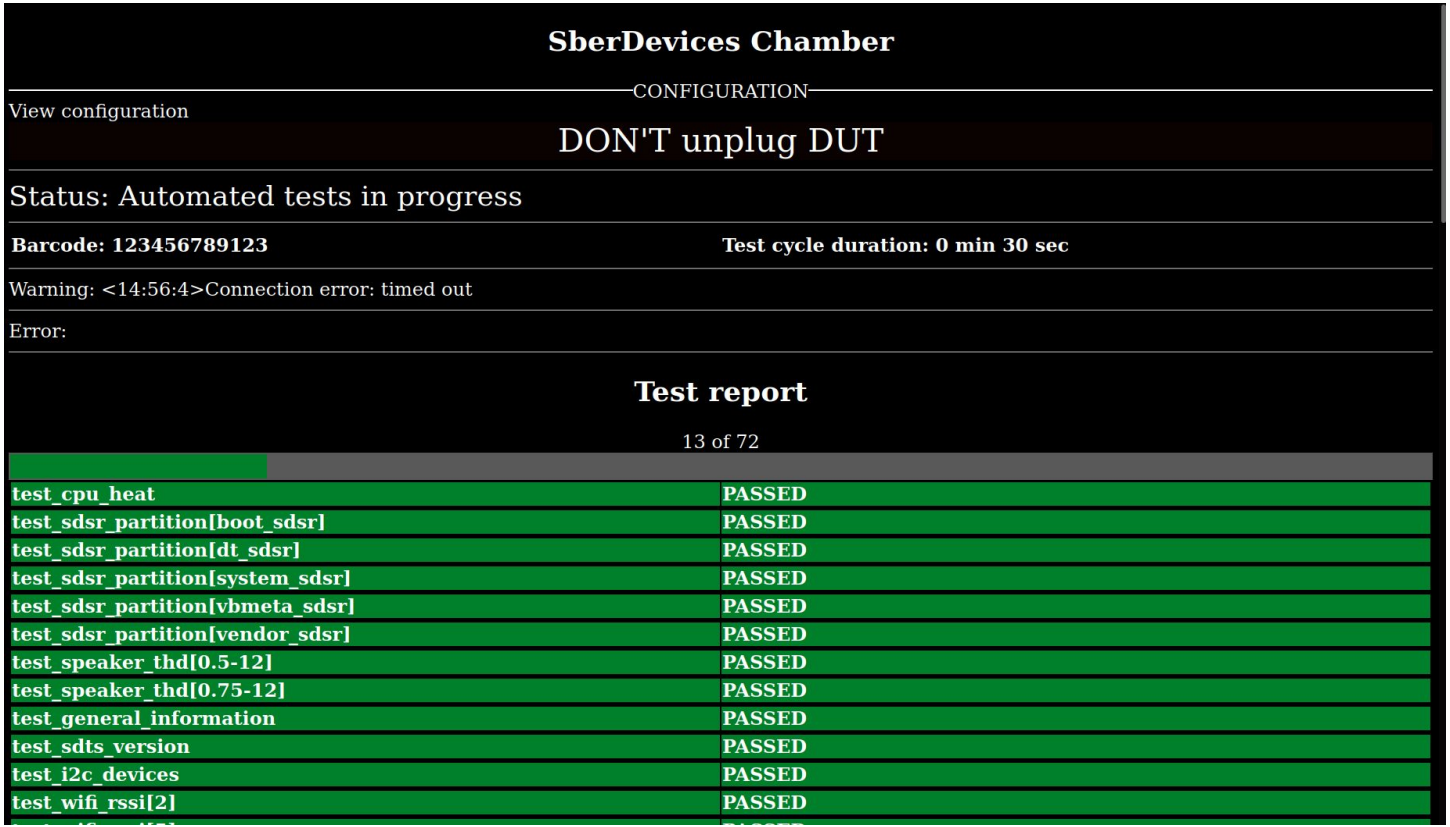
— Перед этим не забываем его освоить

— Начинаем с программы минимум — воспроизводим функционал Purkinje

— Появится ещё время — реализовываем хотелки



Web-интерфейс готов



SberDevices Chamber

CONFIGURATION

View configuration

DON'T unplug DUT

Status: Automated tests in progress

Barcode: 123456789123 **Test cycle duration: 0 min 30 sec**

Warning: <14:56:4>Connection error: timed out

Error:

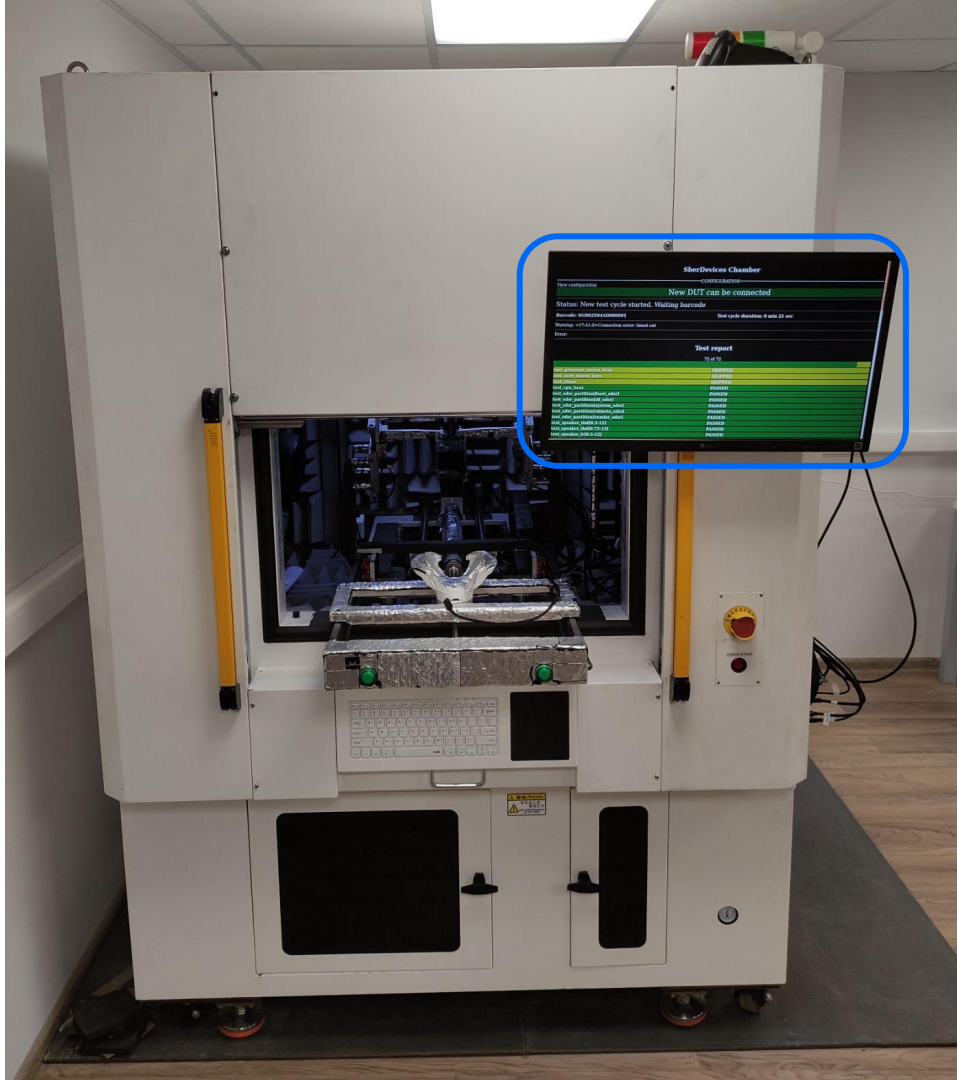
Test report

13 of 72

test_cpu_heat	PASSED
test_sdsr_partition[boot_sdsr]	PASSED
test_sdsr_partition[dt_sdsr]	PASSED
test_sdsr_partition[system_sdsr]	PASSED
test_sdsr_partition[vbmeta_sdsr]	PASSED
test_sdsr_partition[vendor_sdsr]	PASSED
test_speaker_thd[0.5-12]	PASSED
test_speaker_thd[0.75-12]	PASSED
test_general_information	PASSED
test_sdts_version	PASSED
test_i2c_devices	PASSED
test_wifi_rssi[2]	PASSED

Пытаемся подружить код и чембер

Глазами оператора



Пытаемся подружить код и чембер

Код и чембер подружили 😎

Время поднимать инфраструктуру

Задачи:

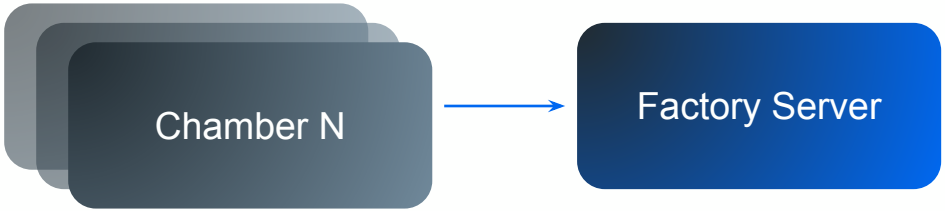
— Обеспечить передачу отчетов с конфиденциальной информацией с чемберов на фабрике в облако

— Визуализация и удобный анализ данных

Время поднимать инфраструктуру

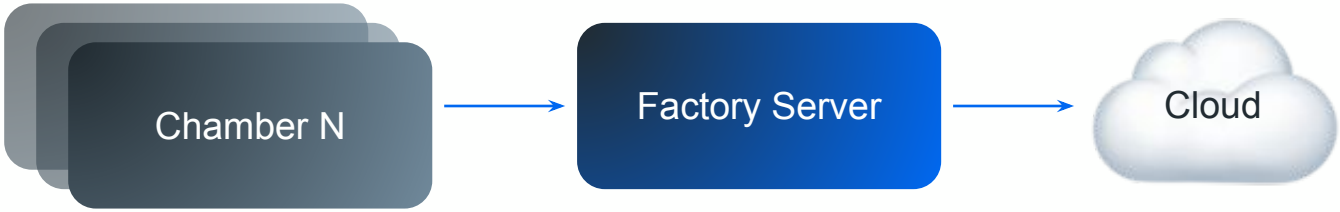
Пытаемся подружить код и чембер

Время поднимать инфраструктуру

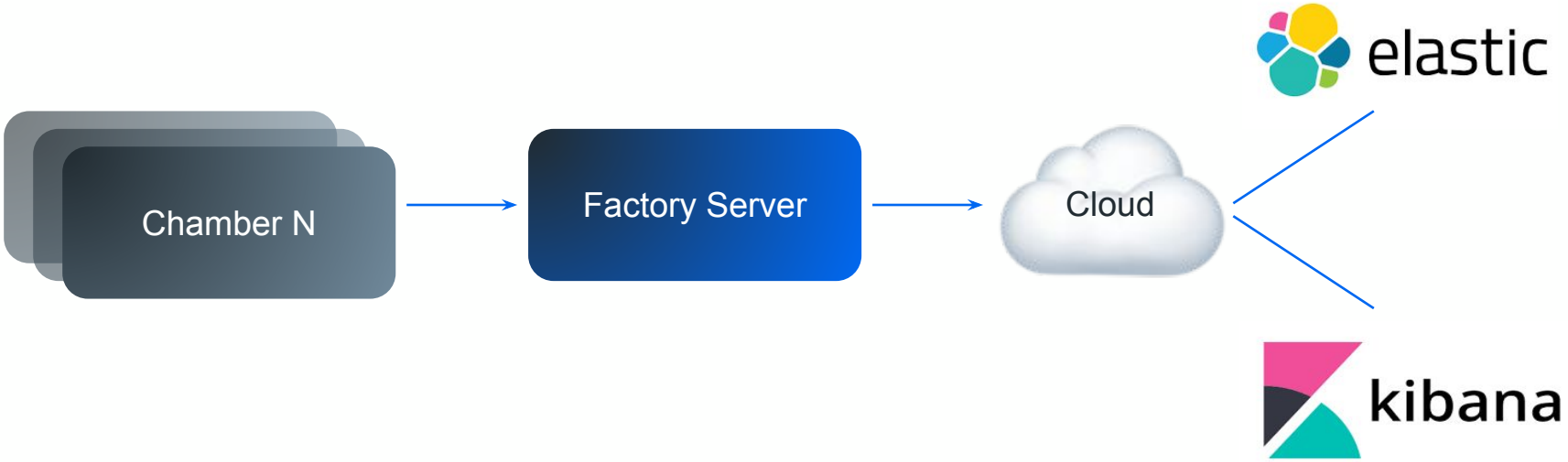


Пытаемся подружить код и чембер

Время поднимать инфраструктуру

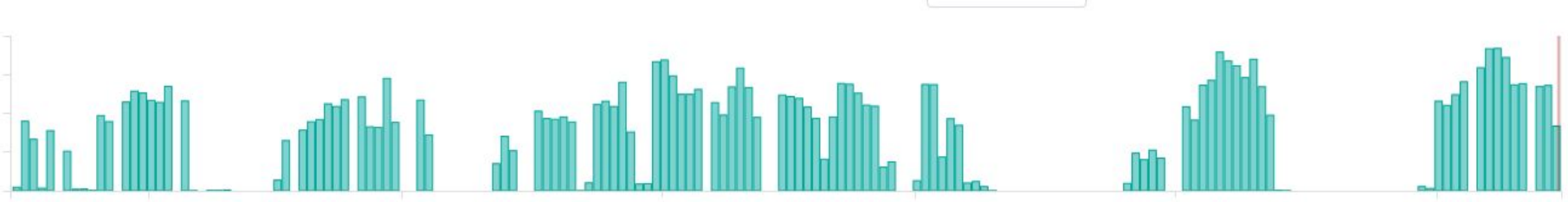


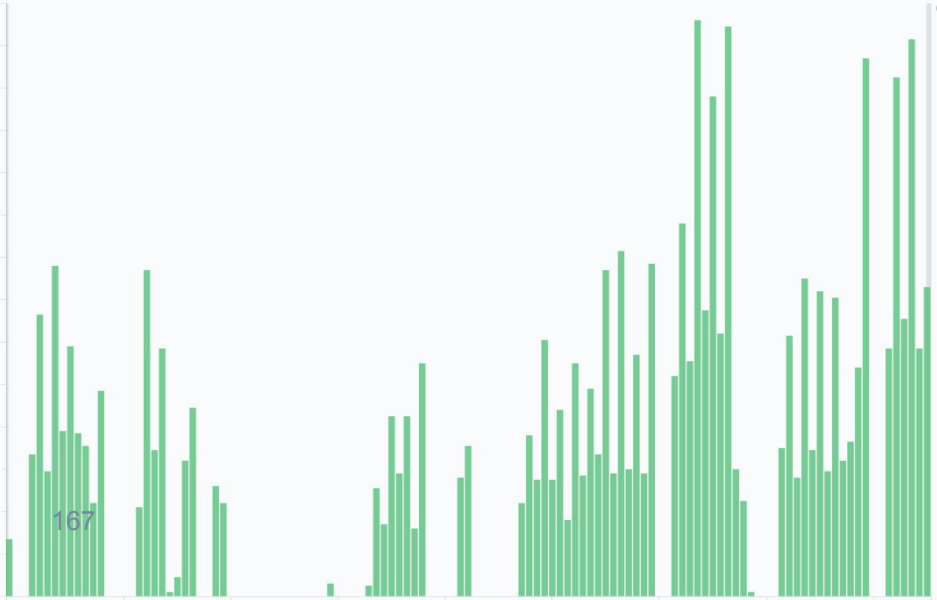
Время поднимать инфраструктуру

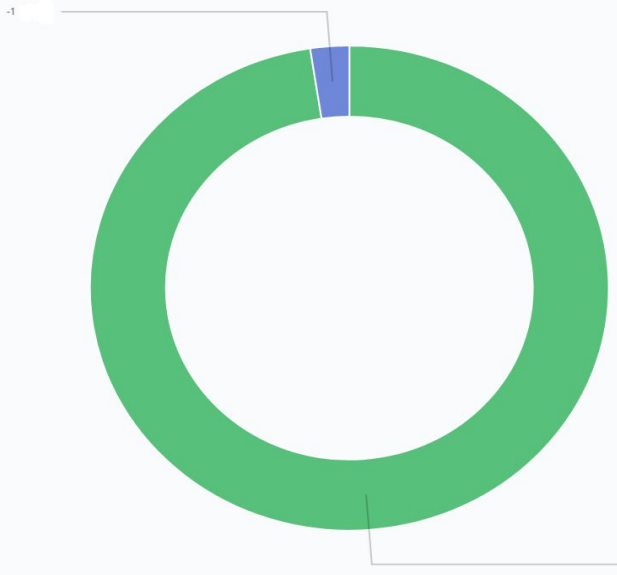


Пытаемся подружить код и чембер

Kibana







Инфраструктура готова!

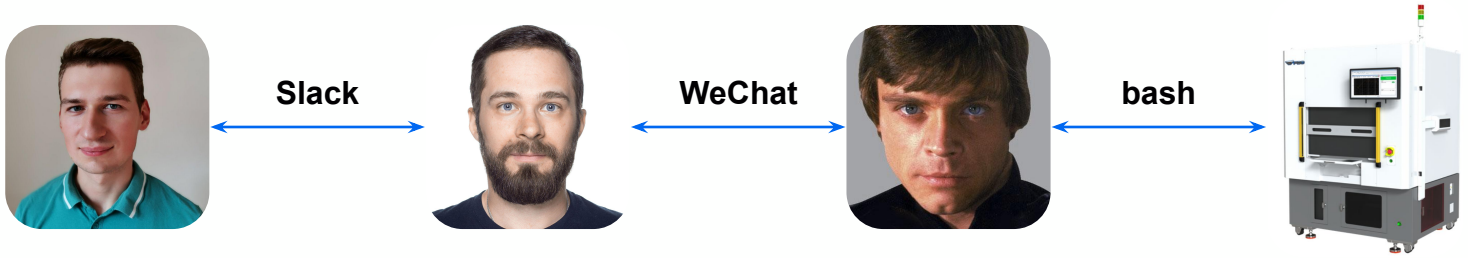
Технические решения:

— Добавили стейт машину. Теперь мы проходим стресс-тесты
— Новый web-интерфейс. Отказались от purkinje, написали альтернативу на js
— Настроили инфраструктуру для анализа статистики

— Зачем нужна all-in-one станция?
— Формируем чек-лист проверок
— Заказываем чембер в Китае
— Вопрос-ответ перед разработкой софта
— Пишем первый код
— Знакомство с чембером в Москве
— Пытаемся подружить код и чембер
— **Чембера на фабрике в Китае**

Чембера на фабрике в Китае

Pipeline запуска



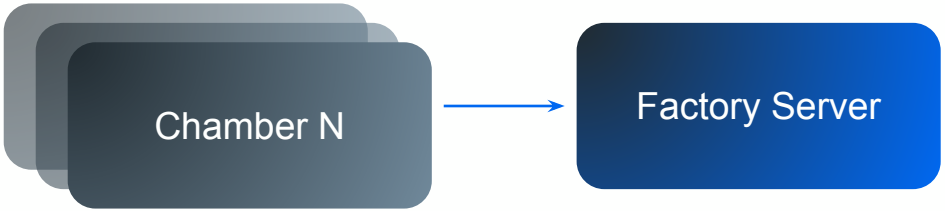
Чембера на фабрике в Китае

Новая угроза — интеграция с фабрикой

На современной китайской фабрике все процессы отражены в специальном софте, который называется SFIS или **Shop Floor Information System**

Чембера на фабрике в Китае

Как мы себе представляли инфраструктуру



Чембера на фабрике в Китае

Как должно быть на самом деле

Чембера на фабрике в Китае

Как обычно происходит интеграция

— С тобой могут поделиться документацией

— А могут даже кодом

Правда, документация обычно такая

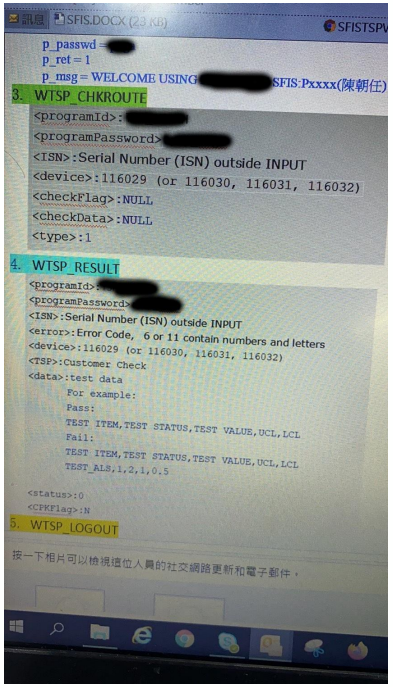
You may find the detailed in the attached document.

Connect → WTSP_LOGIN → WTSP_CHKROUTE → WTSP_RESULT → WTSP_LOGOUT

1. GetDatabaseInformation
2. WTSP_LOGIN

<programId>:[redacted]
<programPassword>:[redacted]
<op>:Operator ID,Outside input,"s13006601"
<password>:null
<device>:116029 (or 116030, 116031, 116032)
<TSP>:Customer Check
<status>:1 expression Login

Login ok:
p_name = [redacted]
p_passwd = [redacted]
p_ret = 1



Чембера на фабрике в Китае

Но нам повезло, с нами даже поделились кодом

— Все выглядело интуитивно просто – 4 метода

— Бери и пользуйся

```
def SFIS_LOGOUT_DB(self):
    return True
```

Интегрируемся

— Написали воркера

— Попробовали запушить свой результат, но что-то пошло не так...

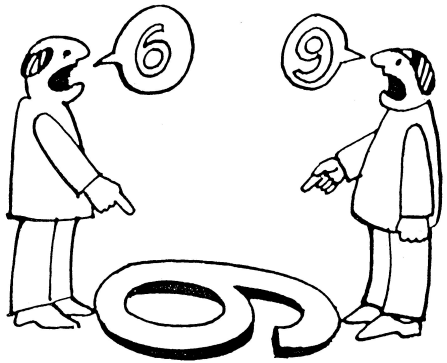
Трудности перевода

— Решили созвониться с фабрикой, проговорить голосом

— В первый раз в моей жизни встреча длилась 4 часа

— Рисовали много схем, чтобы понять какие методы в какой момент надо вызывать

— В итоге разобрались. Всё заработало.



Чембера на фабрике в Китае

Чембера на фабрике в Китае

Они же все должны были быть одинаковыми…

— В чембере используется 3 акустических устройства: два динамика и один микрофон

— На звук влияет вообще всё – кабели, качество исполнения колонок, коннекторы, звукоизоляция внутри акустической камеры

— Первый прогон показал, что чембера отличались друг от друга

Чембера на фабрике в Китае

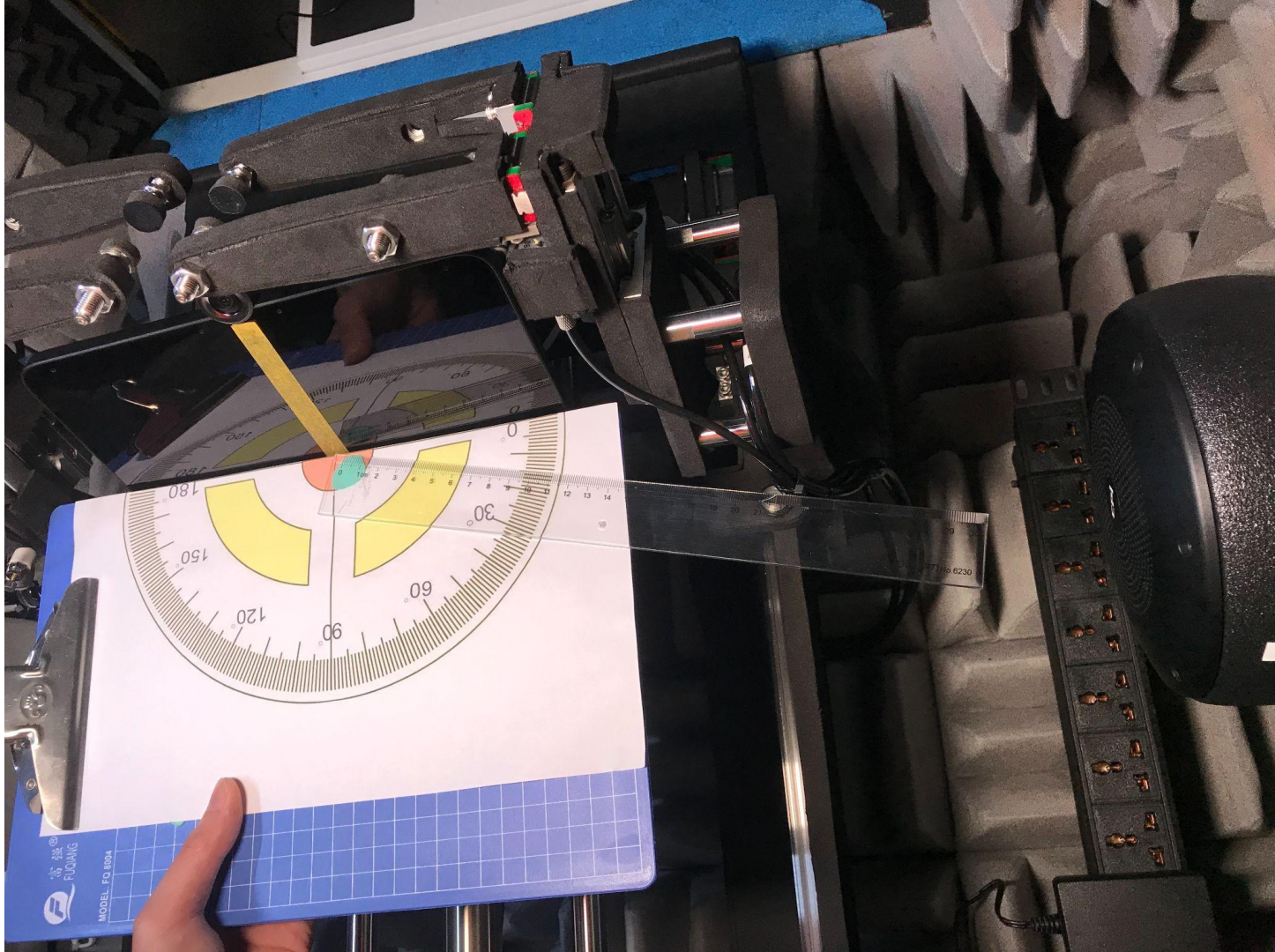
Калибруем чембера

— Изучаем результаты измерений, полученных с одних и тех же юнитов

— Поиск и устранение первопричин в аппаратуре

— Тюнинг порогов и бенчмарков

Чембера на фабрике в Китае



PVT партия — первое испытание чемберов

Чембера на фабрике в Китае

MP — Mass Production

Итоги

Перед нами стояла задача сделать all-in-one тестовую станцию для контроля качества на производстве

Мы прошли этот путь с нуля и справились с этой задачей

Работа с Китаем

— При заказе оборудования из Китая всегда всё нужно перепроверять

  На примере чембера: каждый болт, провод, компонент

— Даже каждый выход усилителя лучше проверить

  Китайцы могут прислать вам усилитель, в котором два из трёх выходов работают, а третий — нет. Вам с этим оборудованием ещё жить в продакшене N лет

Кодинг

— Работая с железом, нужно писать код, устойчивый к изменениям вокруг

— Код с фабрики может быть с сюрпризами. Заложите на них дополнительное время

— Будьте готовы к плохой документации или ее полному отсутствию

— После старта производства тестовое покрытие станции может быть расширено в зависимости от возникающих потребностей контроля качества

Контроль качества

— Никогда не забывайте, что цель фабрики — произвести и продать как можно больше

— Тестовые станции фабрики (на наш взгляд) калибруются не исходя из каких-то физических предпосылок, а из статистики

— Выхлоп производства должен стремиться к 100 %. Остальное — это издержки производства

И всё это, чтобы SberPortal был качественным...

Q&A + контакты

Александр Фомин

Александр Гришин