Светослав Карасев

DriverKit

Гоняем данные по проводу без мам пап и MFi

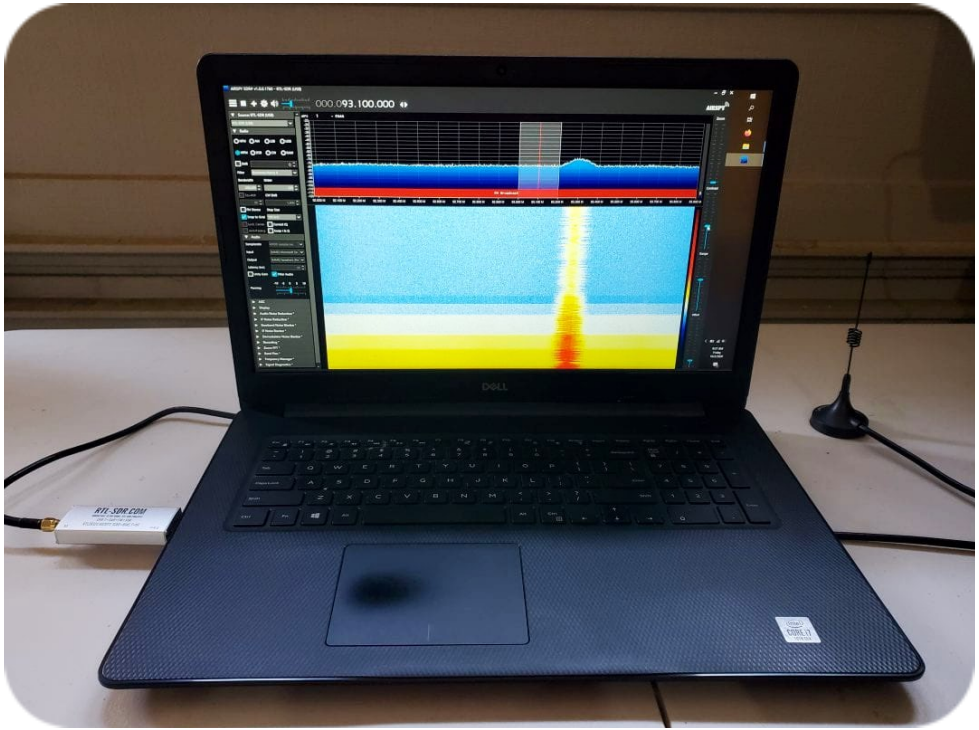
Светослав Карасев

iOS Tech Lead

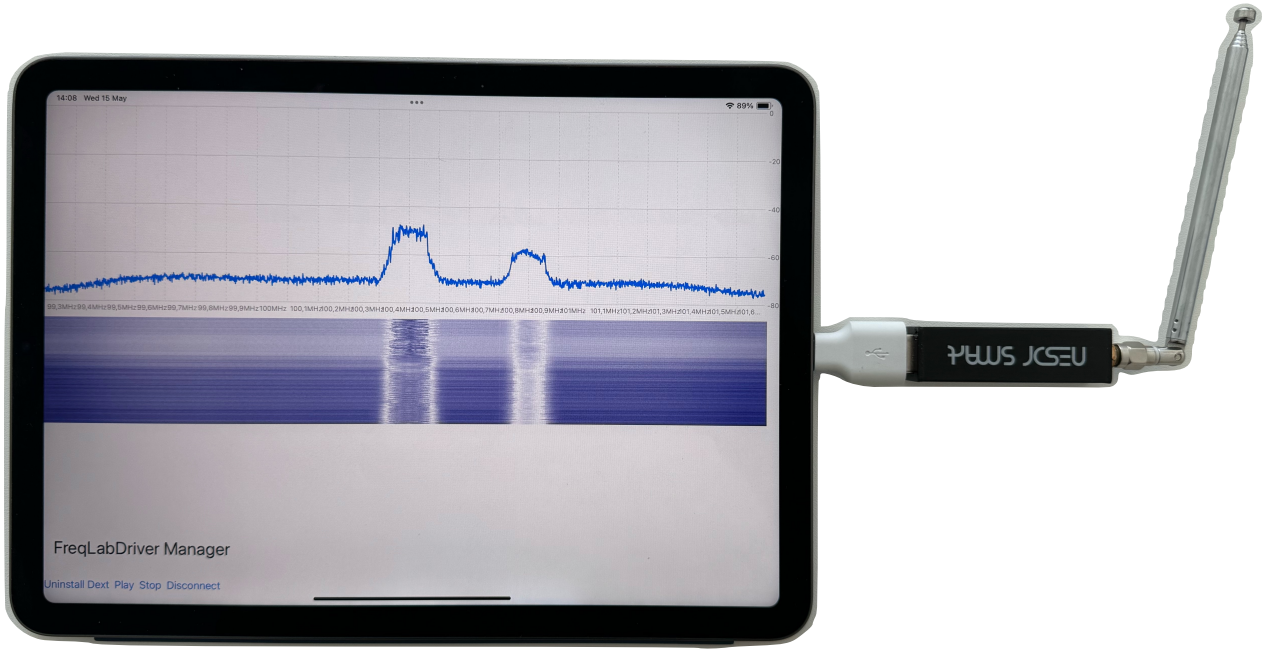
- В нативной iOS разработке с 2015
- Иногда выступаю
- Немного реверс-инженер
- Немного радиолюбитель

Software-defined Radio (SDR)

- Радиостанции
- AIS (корабли)
- ADS-B (самолеты)
- Переговоры самолетов, кораблей, радиолюбителей, служб, etc
- Wi-Fi, Bluetooth
- Снимки погоды со спутников




Wednesday, 15 May
14:07


14:08 Wed 15 May
89%
FreqLabDriver Manager
Uninstall Dext Play Stop Disconnect

Зачем?

Работа с периферийным устройством

Например:

- Обмен большим потоком данных с минимальной задержкой
- Обработка радиосигналов (Software Defined Radio)
- Прошить flipper
- Внешние камеры (инфракрасные)
- Профессиональное аудио оборудование
- Etc.

Что нас ждет?

1

Рассмотрим какие есть варианты для коммуникации между периферийным девайсом и Apple устройствами

2

Сравним

Wi-Fi Bluetooth USB

3

Узнаем, как миллиарды разных устройств передают информацию всего по двум проводам и понимают друг друга

4

Создадим свое примитивное периферийное устройство

5

- Посмотрим, как писать драйвера
- Вспомним C++
- Убедимся, что Swift прекрасен

6

Разберемся с отладкой

Практический пример

Практический пример

Варианты

01.

Wi-Fi

- Можно создать сеть и быстро обмениваться данными
- Довольно простой способ
- Устройство без сим-карты не будет иметь выход в интернет во время действий в этой сети
- Ваше устройство должно иметь Wi-Fi модуль

02.

Bluetooth

- Простота* и доступность**
- Скорость до 2 Мбит/с
- Нужен bluetooth модуль в вашем устройстве

03.

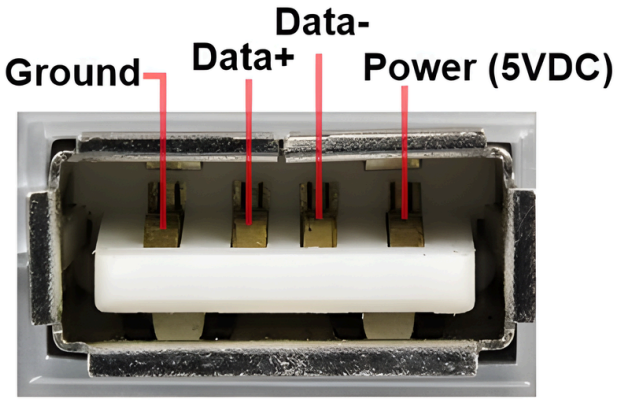
USB

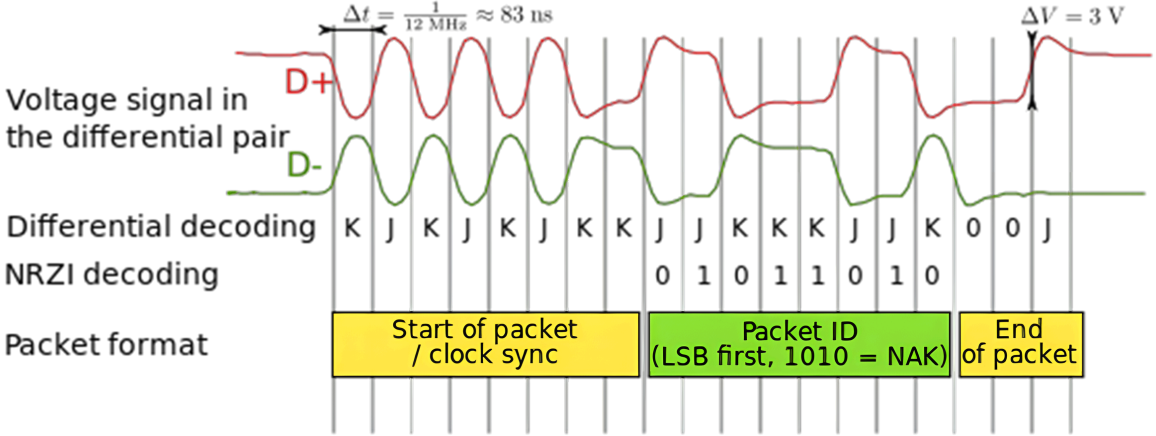
- До 480 Мбит/с по USB 2.0
- Невозможно нормально использовать на iPhone без MFi (made for iPhone): программы для создания лицензированных аксессуаров для устройств Apple
- Изменения появились в iOS 16

Часть 1

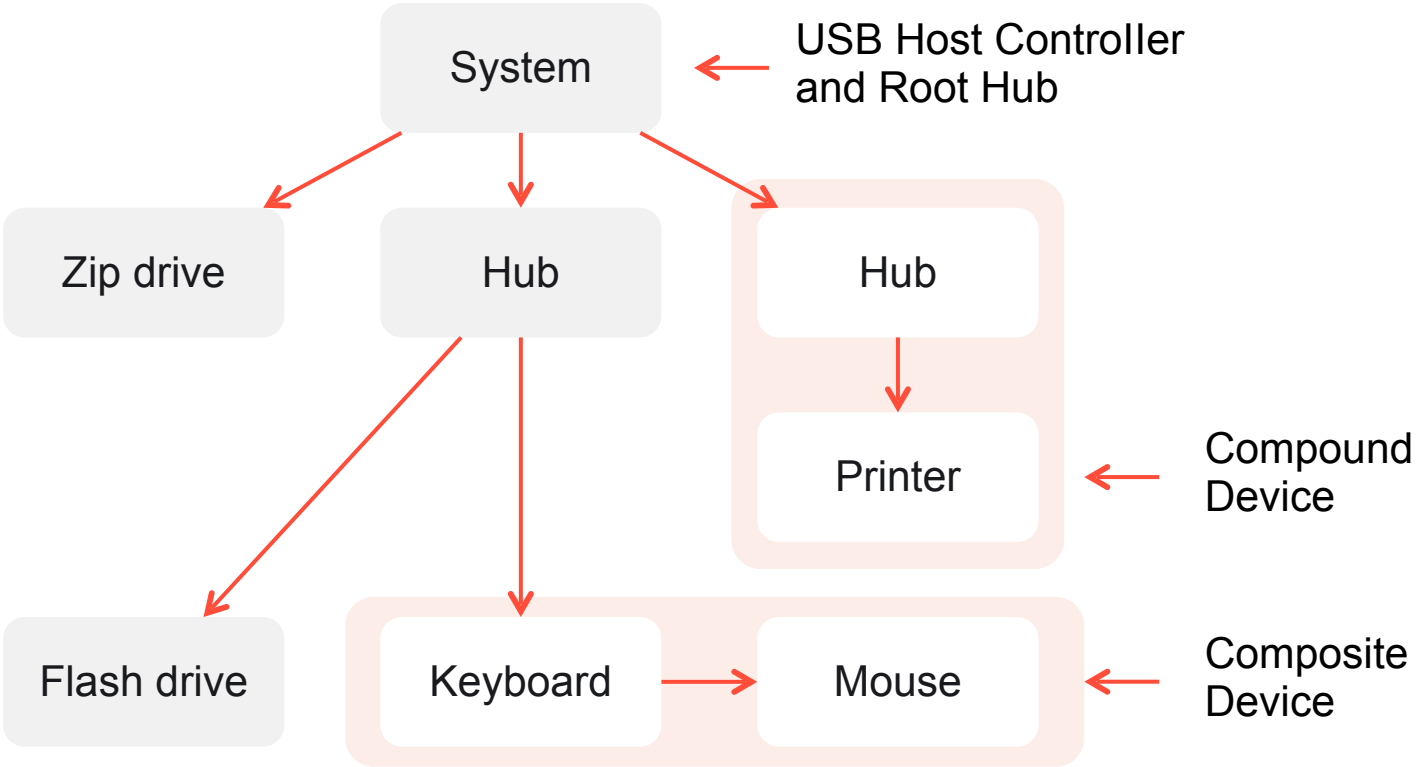
USB

Universal Serial Bus 2.0





Устройства на шине USB



Некоторые классы устройств

Класс устройств массового хранения
Mass Storage Class, MSC

Класс «человеческих» устройств ввода
Human Interface Device Class, HID

Класс принтеров
Printer Class

Класс изображений
Image Class

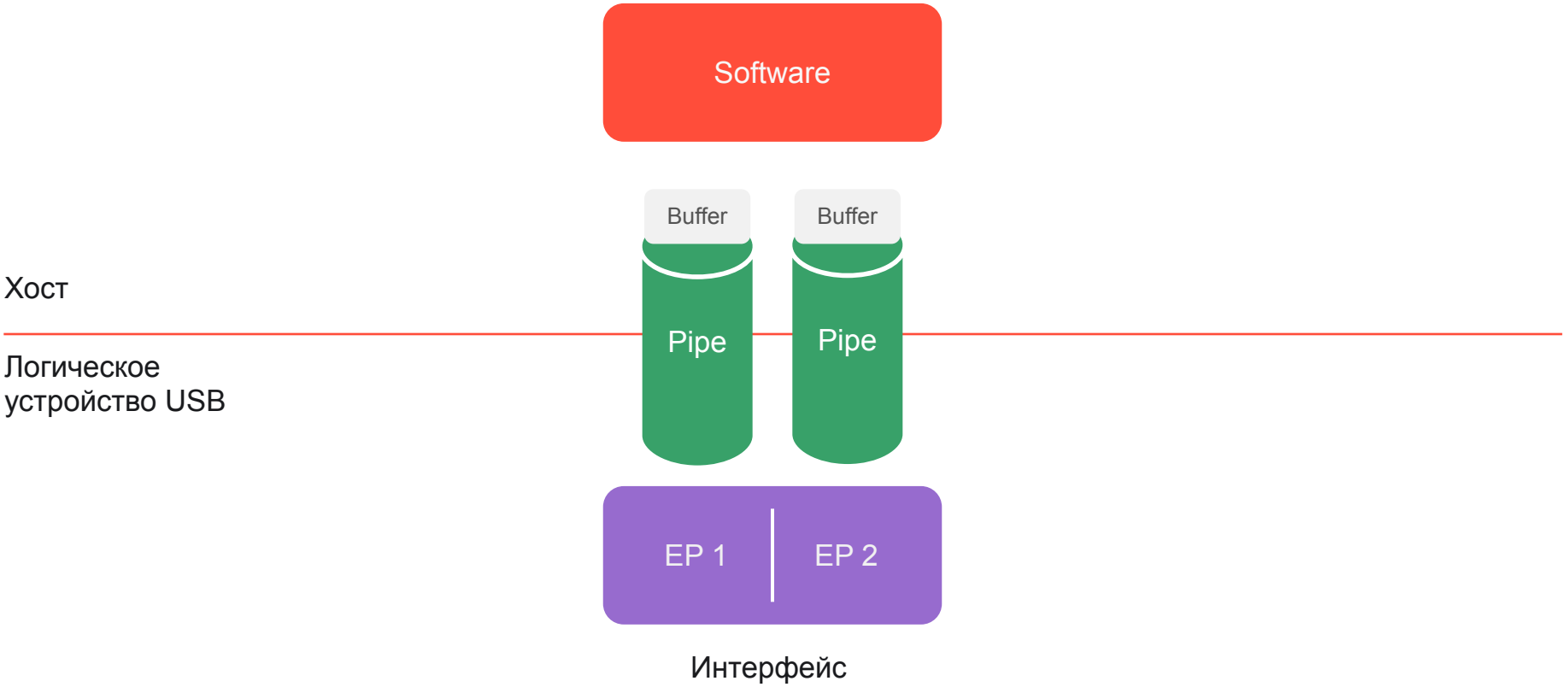
Класс звукового устройства
Audio Device Class

Класс видеоустройств
Video Class

Класс устройств связи
Communication Device Class

Класс разнообразных устройств
Miscellaneous Class

Класс Vendor Specific
Класс специфичных
для производителя устройств


Software
Buffer
Buffer
Хост
Pipe
Pipe
Логическое
устройство USB
EP 1
EP 2
Интерфейс

Типы передачи данных

01.

Изохронные передачи

Isochronous Transfers

02.

Передачи с прерыванием

Interrupt Transfers

03.

Массовые передачи

Bulk Transfers

04.

Управляющие передачи

Control Transfers

Что почитать

https://radiohlam.ru/usb1/

Мини итог

USB 2.0

01.

Очень просто

- Всего 4 провода
- 2 провода для передачи данных

02.

Много классов устройств

- Хранение информации
- Устройства ввода
- Связь
- Сложные устройства

03.

Знакомый обмен информацией

- Интерфейсы
- Эндпоинты
- Пайпы

Часть 2

Делаем железку

Делаем внешний сканер

1

Нужен devboard
с Wi-Fi на борту

2

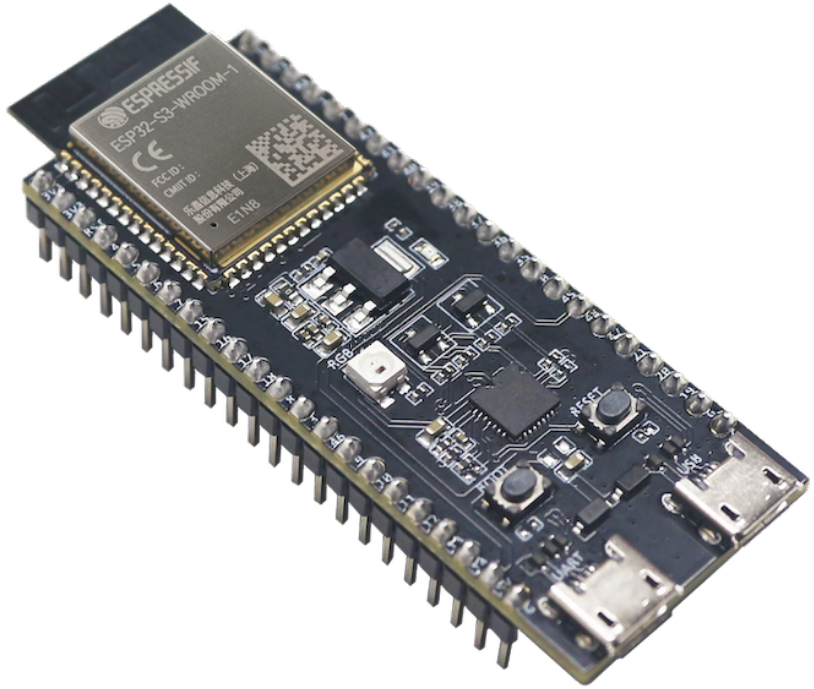
Нужна возможность
реализовывать USB
устройства

3

Желательно
сразу 2 USB порта

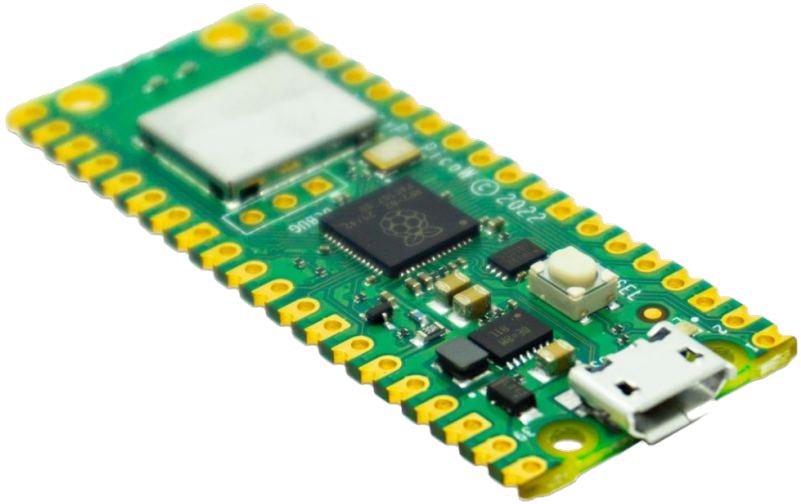
ESP32-S3

- Двухядерный, 32-битный, 240 МГц процессор
- Wi-Fi и Bluetooth
- 2 USB
- 600₽ на AliExpress



RaspberyPi Pico W

- Двухъядерный процессор Arm Cortex M0+, гибкая тактовая частота до 133 МГц
- Wi-Fi, Bluetooth
- USB 1.1 с поддержкой устройств и хостов
- 6€ в Европе, от 800₽ на AliExpress



Настраиваем устройство

Воспользуемся библиотекой TinyUSB

```
tusb_desc_device_t const desc_device =
    {
        .bcdUSB = USB_BCD, // USB 2.0
        .bDeviceClass = TUSB_CLASS_MISC,
        .idVendor = USB_VID,
        .idProduct = USB_PID,
        .bcdDevice = 0x0100,
       ...
    };

uint8_t const desc_fs_configuration[] =
    {
        TUD_CONFIG_DESCRIPTOR(1, ITF_NUM_TOTAL, 0, CONFIG_TOTAL_LEN, 0x00, 100),
        TUD_CDC_DESCRIPTOR(ITF_NUM_CDC, 4, EPNUM_CDC_NOTIF, 8, EPNUM_CDC_OUT, EPNUM_CDC_IN, 64),
        TUD_VENDOR_DESCRIPTOR(ITF_NUM_VENDOR, 5, EPNUM_VENDOR_OUT, 0x80 | EPNUM_VENDOR_IN, 64),
    };
```

Все работает в цикле

```
static void usb_task(void *pvParam)
{
    (void)pvParam;

    do
    {
        // TinyUSB device task
        tud_task();
        cdc_task();
        vendor_task();
    } while (true);
}
```

Сканируем сети

```
void cdc_task(void)
{
...
    uint32_t count = tud_cdc_read(buf, sizeof(buf));
    (void)count;

    if (buf[0] == '1') {
            uint16_t size = 20;
            wifi_network wifiList[20] = { 0 };
            int number = scan_wifi(wifiList, size);
            for (int i = 0; i < number; i++)
            {
                ESP_LOGI("RECEIVED", "%s %d", wifiList[i].ssid, wifiList[i].rssi);
                char* ssid = (char*)wifiList[i].ssid;
                tud_cdc_write(ssid, strlen(ssid));
                tud_cdc_write("\r\n", 2);
            }
            tud_cdc_write_flush();
    } else {
...
```

Тестируем без драйверов


^ Control
+
A


^ Control
+
\

Что почитать

https://github.com/svedm/esp32-usb-wifiscanner/tree/master

Мини итог

Создание устройства

01.

Подойдет любой недорогой МК с поддержкой USB

- ESP 32-S3
- RPi Pico W
- Можно взять и готовое устройство для своих целей

02.

Код писать не сложно, есть embeded-swift

В прошивке МК задаются основные параметры USB-устройства

Часть 3

Немного теории Darwin

Подключение устройства

Подключение устройства

01 Определение устройства

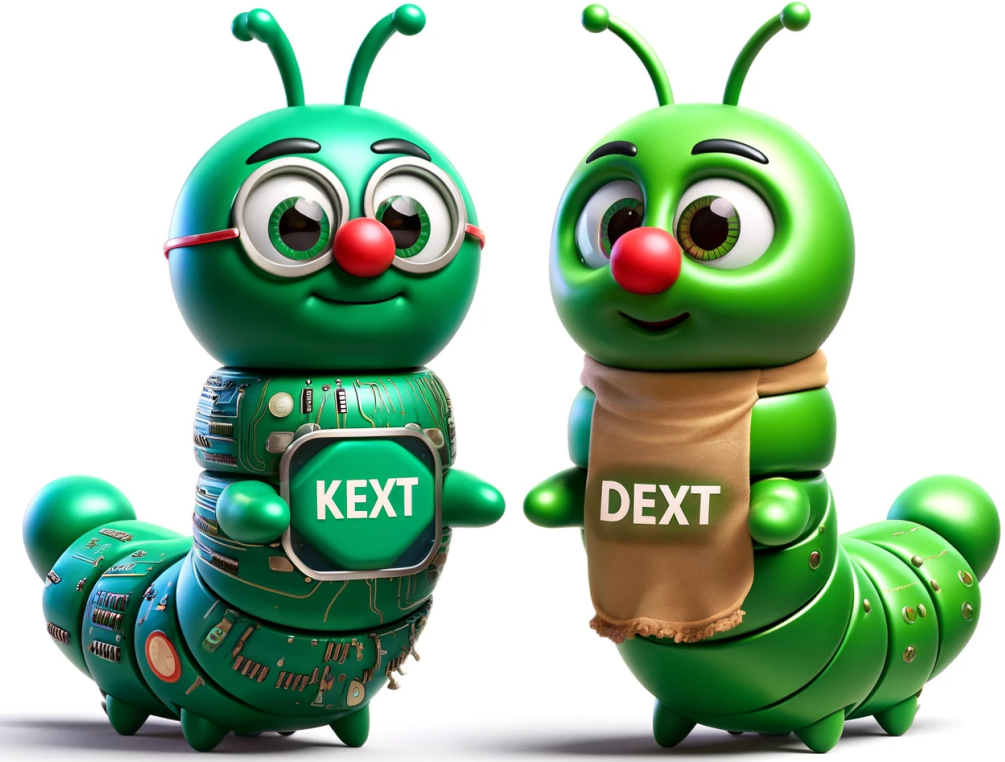
02 Запрос идентификационной информации

03 Поиск подходящего драйвера

04 Загрузка и активация драйвера

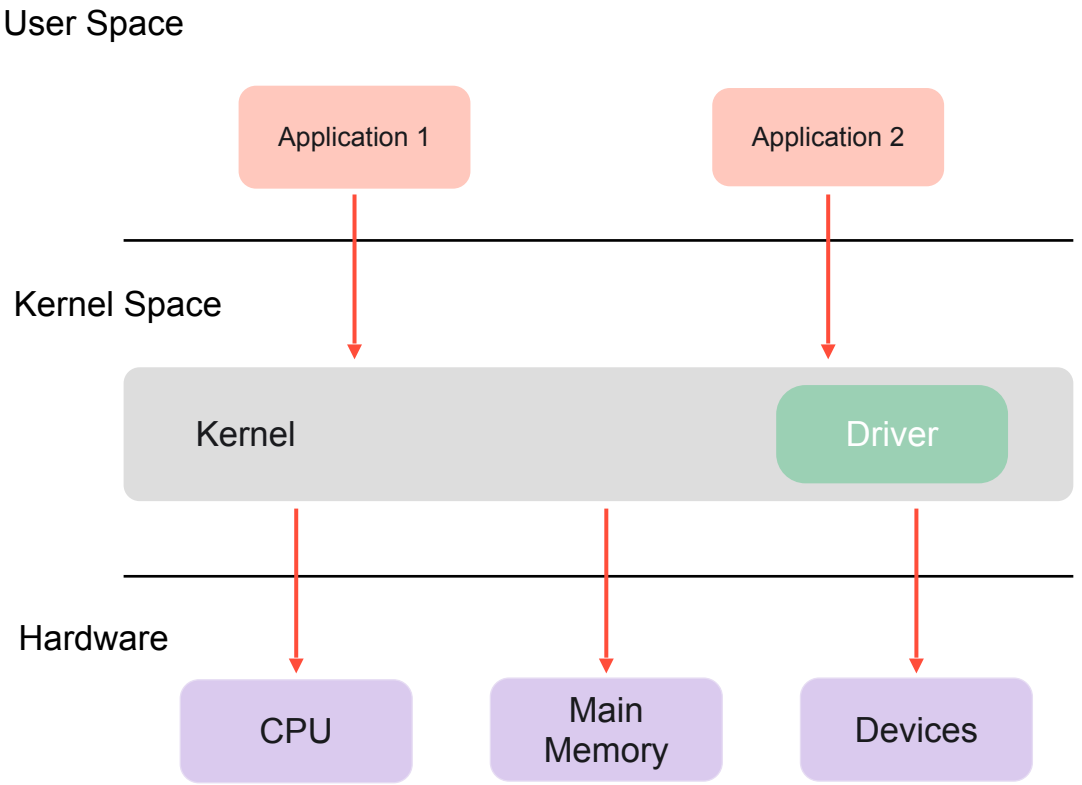
05 Мониторинг и управление

Драйвера в macOS



Kernel Extension

- Исполняется в пространстве ядра
- Если падает, то вместе с ядром
- Для отладки нужен второй ПК



IOKit

1. Позволяет писать драйвера и работать с ними в «уникальном» объектно-ориентированом стиле
2. Написан на Embedded C++
3. Реализует драйверную модель macOS

От IOKit не уйти

OS classes (general)	OSObject	База
IOKit classes (general)	IORegistry Entry	Запись в реестре
	IOService	Сервис, реализует жизненный цикл
Family classes (general)	IOKit family superclasses	Классы-помощники

DriverKit Extension

- macOS 10.15+ (Catalina)
- iOS 16.0+
- iPadOS 16.0+

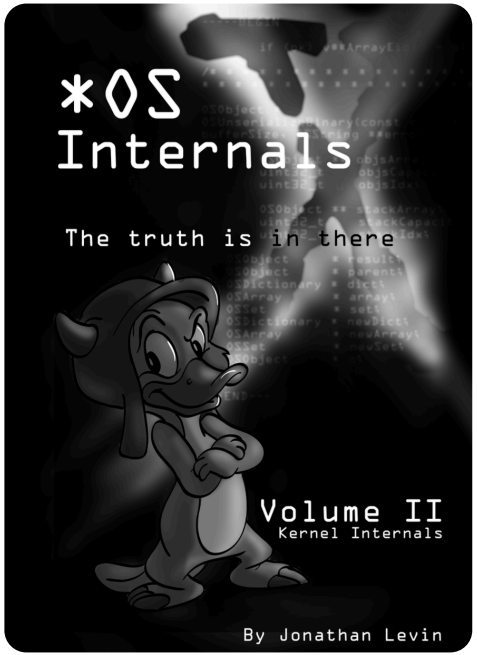
- Работает в пользовательском пространстве
- Падения не приводят к фатальным последствиям*
- Можно дебажить через LLDB без дополнительного ПК
- Такие классные урезанные плюсы
- Работает на iPad с M1 и выше
- Рекомендован к использованию

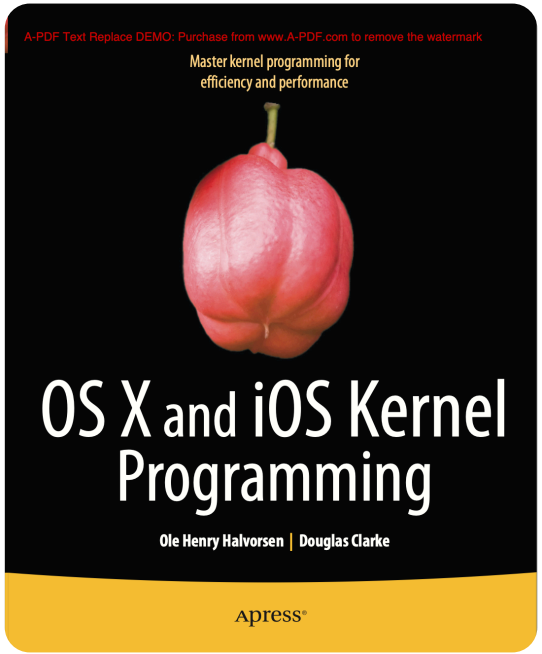
DriverKit

- USBDriverKit (iPadOS, macOS)
- PCIDriverKit (iPadOS, macOS)
- AudioDriverKit (iPadOS, macOS)
- SerialDriverKit (macOS)
- NetworkingDriverKit (macOS)
- HIDDriverKit (macOS)

Очередная полезная литература

IOKit Fundamentals:
Apple archive





Мини итог

Darwin

1

При подключении устройства подбирается драйвер.
ОС следит за его состоянием

2

Есть 2 типа драйверов

3

KEXT работает
в пространстве ядра

4

DEXT в работает
пользовательском пространстве

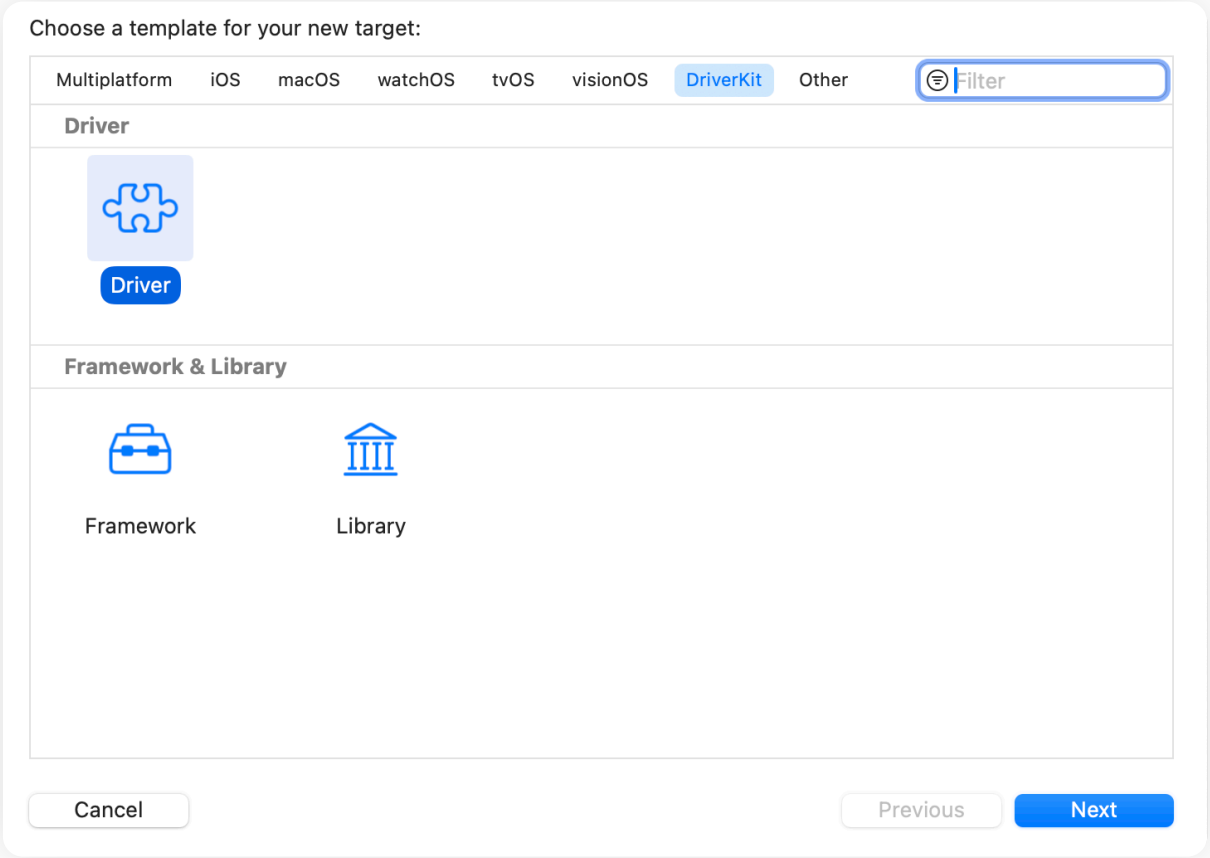
5

IOKit + DriverKit

Часть 4.1

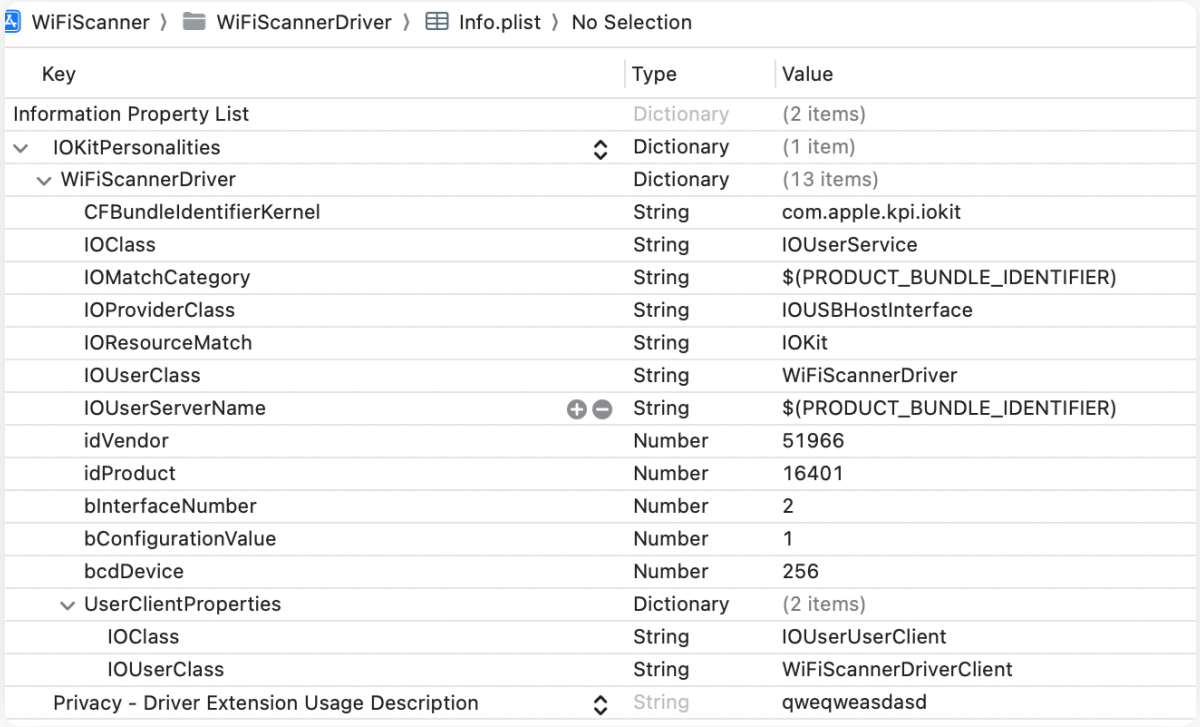
От слов к делу

Хватит теории — пишем драйвер



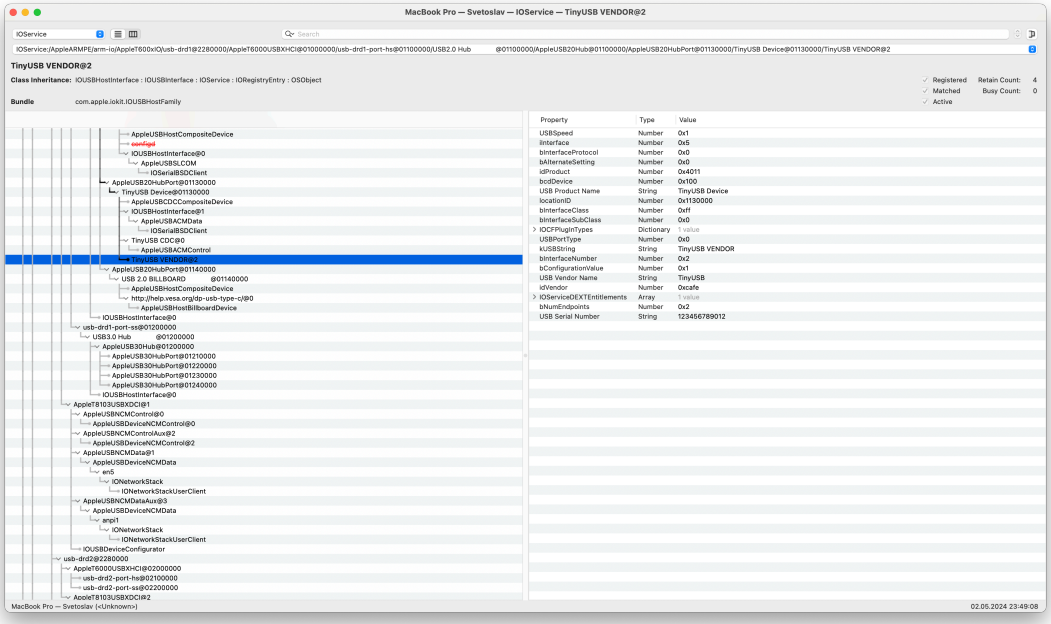
Мэтчинг драйвера и устройства

- Сопоставление класса
- Пассивный мэтчинг по параметрам
- Активный мэтчинг



IORegistry

ioreg -irbln "TinyUSB Device"

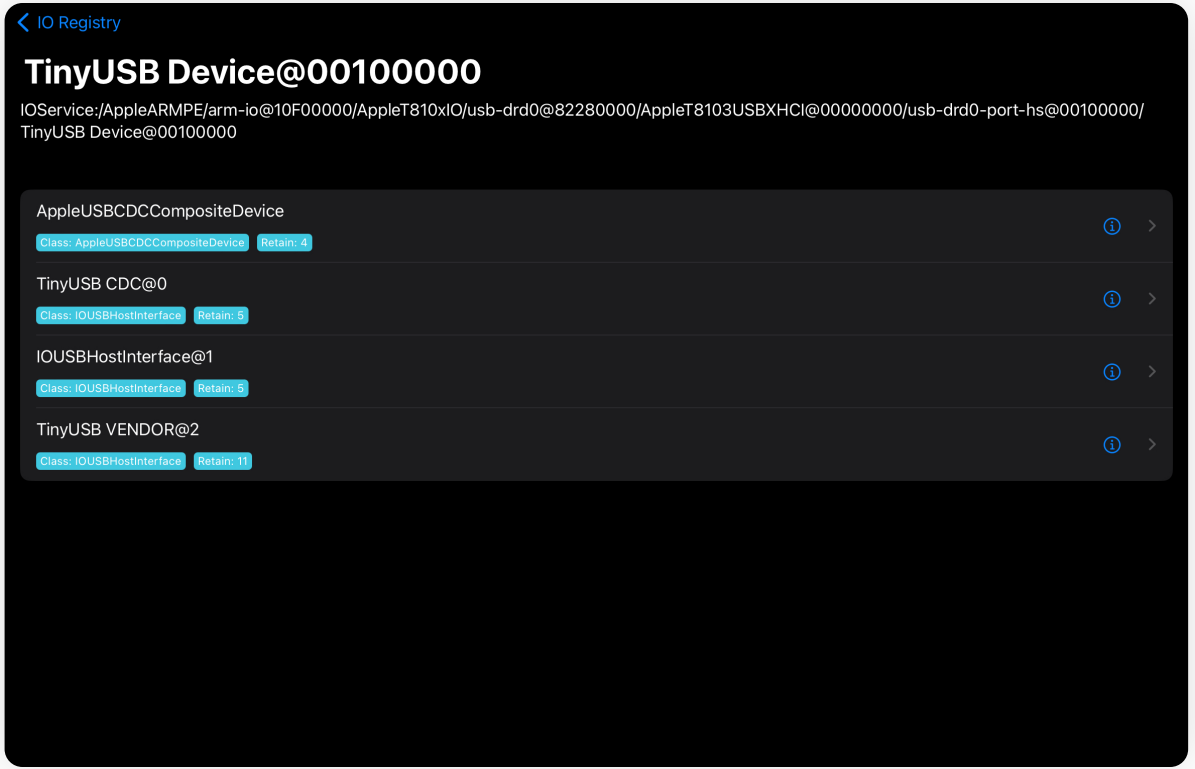



```
+-o TinyUSB VENDOR@2  <class IORegistryEntry:IOService:IOUSBInterface:IOUSBHostInterface, id 0x1000021d4, registered, matched$
    {
      "USBSpeed" = 1
      "iInterface" = 5
      "bInterfaceProtocol" = 0
      "bAlternateSetting" = 0
      "idProduct" = 16401
      "bcdDevice" = 256
      "USB Product Name" = "TinyUSB Device"
      "locationID" = 18022400
      "bInterfaceClass" = 255
      "bInterfaceSubClass" = 0
      "IOCFPlugInTypes" = {"2d9786c6-9ef3-11d4-ad51-000a27052861"="IOUSBHostFamily.kext/Contents/PlugIns/IOUSBLib.bundle"}
      "USBPortType" = 0
      "kUSBString" = "TinyUSB VENDOR"
      "bInterfaceNumber" = 2
      "bConfigurationValue" = 1
      "USB Vendor Name" = "TinyUSB"
      "idVendor" = 51966
      "IOServiceDEXTEntitlements" = (("com.apple.developer.driverkit.transport.usb"))
      "bNumEndpoints" = 2
      "USB Serial Number" = "123456789012"
    }
```

Как посмотреть на iPad?

IO Explorer

https://github.com/svedm/IOExplorer



Пассивный мэтчинг

Table 1-2 Keys for finding a USB device

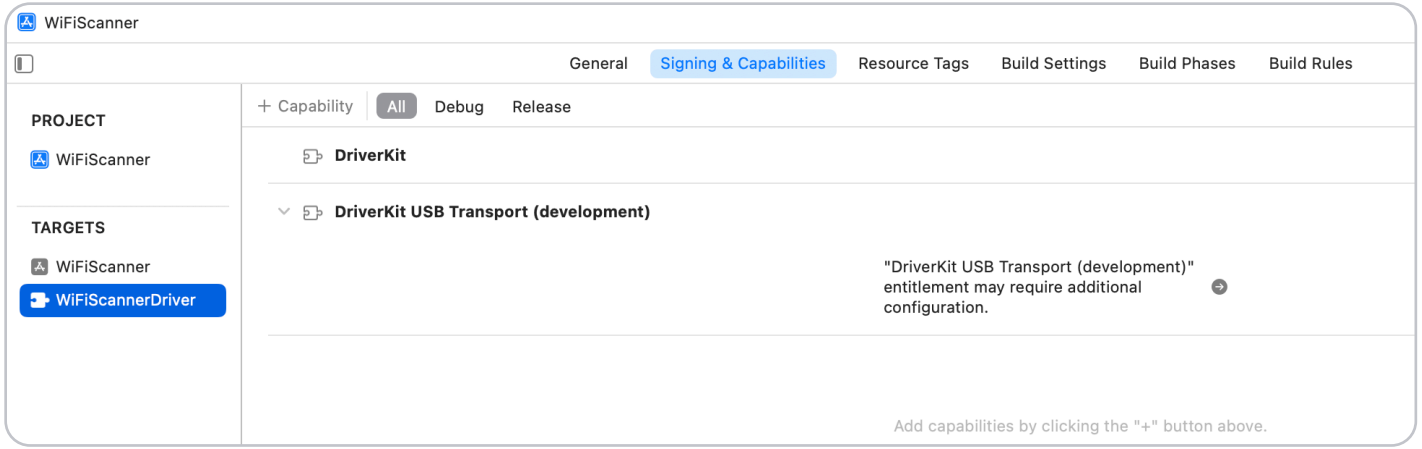
Key	Notes
`idVendor` + `idProduct` + `bcdDevice`	`bcdDevice` contains the release number of the device
`idVendor` + `idProduct`	
`idVendor` + `bDeviceSubClass` + `bDeviceProtocol`	Use this key only if the device's `bDeviceClass` is $FF
`idVendor` + `bDeviceSubClass`	Use this key only if the device's `bDeviceClass` is $FF
`bDeviceClass` + `bDeviceSubClass` + `bDeviceProtocol`	Use this key only if the device's `bDeviceClass` is *not* $FF
`bDeviceClass` + `bDeviceSubClass`	Use this key only if the device's `bDeviceClass` is *not* $FF

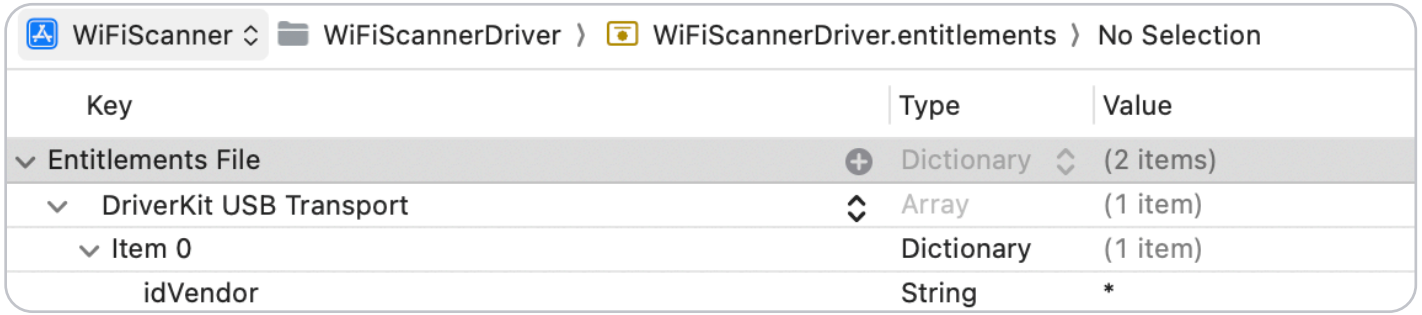
Table 1-3 lists the keys you can use to find interfaces (not devices). Each key element is a piece of information contained in an interface descriptor for a USB device.

Table 1-3 Keys for finding a USB interface

Key	Notes
`idVendor` + `idProduct` + `bcdDevice` + `bConfigurationValue` + `bInterfaceNumber`	
`idVendor` + `idProduct` + `bConfigurationValue` + `bInterfaceNumber`	
`idVendor` + `bInterfaceSubClass` + `bInterfaceProtocol`	Use this key only if `bInterfaceClass` is $FF
`idVendor` + `bInterfaceSubClass`	Use this key only if `bInterfaceSubClass` is $FF
`bInterfaceClass` + `bInterfaceSubClass` + `bInterfaceProtocol`	Use this key only if `bInterfaceSubClass` is *not* $FF
`bInterfaceClass` + `bInterfaceSubClass`	Use this key only if `bInterfaceSubClass` is *not* $FF

Не забудьте про capatibilities

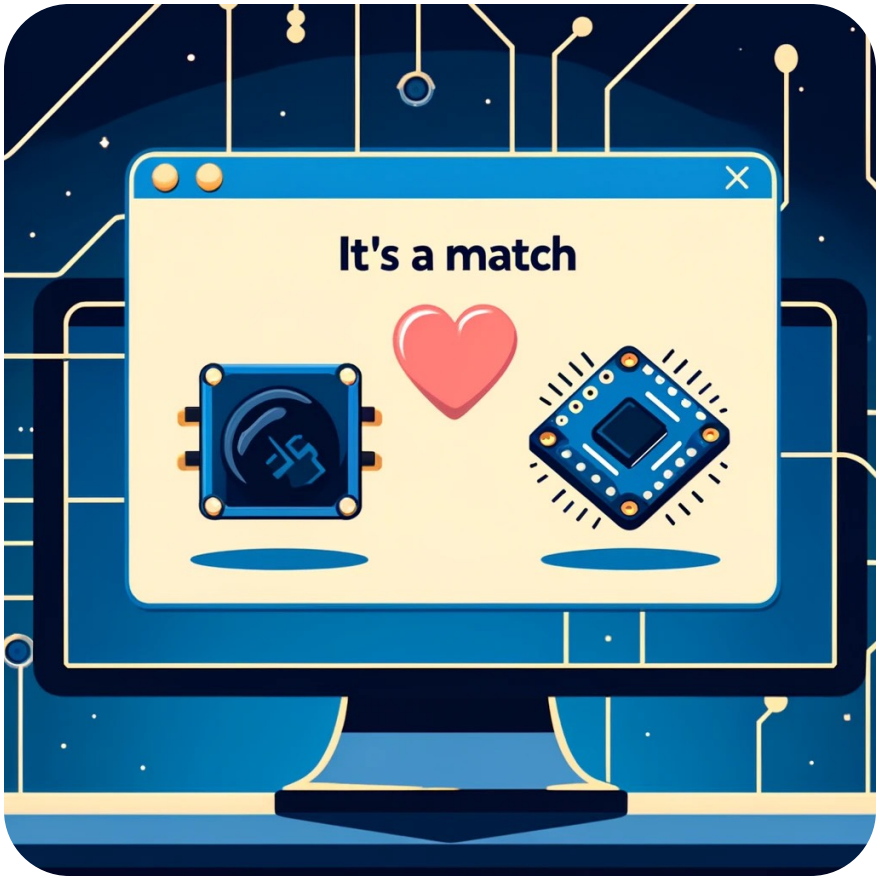




Запуск на macOS

- Нужно отключить SIP
- systemextensionsctl developer on

```
1  private let dextIdentifier = "net.svedm.WiFiScanner.WiFiScannerDriver"
2
3  let request = OSSystemExtensionRequest
4      .activationRequest(forExtensionWithIdentifier: dextIdentifier, queue: .main)
5  request.delegate = self
6
7  OSSystemExtensionManager.shared.submitRequest(request)
```


It's a match

```
|
+-o TinyUSB VENDOR@2  <class IORegistryEntry:IOService:IOUSBInterface:IOUSBHostInterface, id 0x10000ed76, registered, matched, active, busy 0 (49 ms), retain 7>
  | {
  |   "USBSpeed" = 1
  |   "iInterface" = 5
  |   "bInterfaceProtocol" = 0
  |   "bAlternateSetting" = 0
  |   "idProduct" = 16401
  |   "bcdDevice" = 256
  |   "USB Product Name" = "TinyUSB Device"
  |   "locationID" = 18022400
  |   "bInterfaceClass" = 255
  |   "bInterfaceSubClass" = 0
  |   "IOCFPlugInTypes" = {"2d9786c6-9ef3-11d4-ad51-000a27052861"="IOUSBHostFamily.kext/Contents/PlugIns/IOUSBLib.bundle"}
  |   "USBPortType" = 0
  |   "kUSBString" = "TinyUSB VENDOR"
  |   "bInterfaceNumber" = 2
  |   "bConfigurationValue" = 1
  |   "USB Vendor Name" = "TinyUSB"
  |   "idVendor" = 51966
  |   "IOServiceDEXTEntitlements" = (("com.apple.developer.driverkit.transport.usb"))
  |   "bNumEndpoints" = 2
  |   "IODEXTMatchCount" = 1
  |   "USB Serial Number" = "123456789012"
  | }
  |
  +-o WiFiScannerDriver  <class IORegistryEntry:IOService:IOUserService, id 0x10000edf0, registered, matched, active, busy 0 (0 ms), retain 9>
      {
        "IOClass" = "IOUserService"
        "kOSBundleDextUniqueIdentifier" = <05b62aabee2cfe022ac152ec8d129a34a42619c7bbf0f46196b66892c92c70c7>
        "IOUserClass" = "WiFiScannerDriver"
        "IOUserServerName" = "net.svedm.WiFiScanner.WiFiScannerDriver"
        "IOResourceMatch" = "IOKit"
        "IOPersonalityPublisher" = "net.svedm.WiFiScanner.WiFiScannerDriver"
        "idProduct" = 16401
        "bcdDevice" = 256
        "IOProviderClass" = "IOUSBHostInterface"
        "IOPowerManagement" = {"CapabilityFlags"=2,"MaxPowerState"=2,"CurrentPowerState"=2}
        "IOProbeScore" = 100000
        "IOUserServerCDHash" = "94a1f68c486417d39d849be1b1c11037ec92c91f"
        "IOMatchedPersonality" = {"IOClass"="IOUserService","CFBundleIdentifier"="net.svedm.WiFiScanner.WiFiScannerDriver","IOProviderClass"="IOUSBHostInterface","IOUserServerCDHash"="94a1f68c486417d39d849be1b1c11037ec92c91f","bcdDevice"=256,$
        "IOUserClasses" = ("WiFiScannerDriver","IOService","OSObject")
        "CFBundleIdentifierKernel" = "com.apple.kpi.iokit"
        "bConfigurationValue" = 1
        "IOMatchCategory" = "net.svedm.WiFiScanner.WiFiScannerDriver"
        "CFBundleIdentifier" = "net.svedm.WiFiScanner.WiFiScannerDriver"
        "bInterfaceNumber" = 2
        "idVendor" = 51966
        "UserClientProperties" = {"IOClass"="IOUserUserClient","IOUserClass"="WiFiScannerDriverClient"}
      }
```

Если мэтча не случилось?

- **Правильно ли составлен plist, совпадает ли в нем IOProviderClass с тем что вы запланировали**

  Проверить можно собрав CFDictionary для IOServiceGetMatchingServices

- **Совпадает ли провижн и entitlements**

- **Логи: https://developer.apple.com/forums/thread/705868**

- **log show --predicate 'sender == "sysextd" or sender CONTAINS "com.example“’ --info --debug --last 1h**

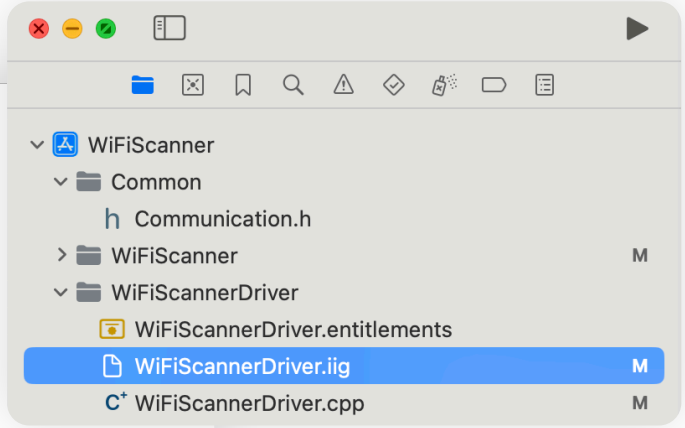
Пишем логику

```
#include <DriverKit/IOService.iig>

class WiFiScannerDriver: public IOService
{
public:
    virtual bool init(void) override;
    virtual kern_return_t Start(IOService * provider) override;
    virtual kern_return_t Stop(IOService* provider) override;
    virtual void free(void) override;

    virtual kern_return_t NewUserClient(uint32_t type, IOUserClient** userClient) override;

protected:
    kern_return_t openDevice() LOCALONLY;
    kern_return_t closeDevice() LOCALONLY;
    kern_return_t writeToPipe(OSData *data) LOCALONLY;
    kern_return_t readFromPipe(IOMemoryDescriptor* buffer, uint32_t* bytesTransferred) LOCALONLY;
    kern_return_t readFromPipeAsync(IOMemoryDescriptor *buffer, OSAction *completion) LOCALONLY;
    uint16_t getMaxPacketSize() LOCALONLY;
};
```



::Start

```
1  ivars->interface = OSDynamicCast(IOUSBHostInterface, provider);
2
3  ret = RegisterService();
```

Ищем пайпы

```
kern_return_t WiFiScannerDriver::openDevice()
{
    const IOUSBConfigurationDescriptor* cfg_descriptor = ivars->interface->CopyConfigurationDescriptor();
    const IOUSBInterfaceDescriptor* int_descriptor = ivars->interface->GetInterfaceDescriptor(cfg_descriptor);

    ret = ivars->interface->Open(this, 0, NULL);
    const IOUSBEndpointDescriptor* endpointDescriptor = NULL;

    while((endpointDescriptor = IOUSBGetNextEndpointDescriptor(
                                                        cfg_descriptor,
                                                        int_descriptor,
                                                        (IOUSBDescriptorHeader*)endpointDescriptor
                                                        )) != NULL)
    {
        if (endpointDescriptor->bEndpointAddress == 0x83) {
            IOLog("inPipe found");
            ret = ivars->interface->CopyPipe(endpointDescriptor->bEndpointAddress, &ivars->inPipe);
            ivars->inMaxPacketSize = endpointDescriptor->wMaxPacketSize;
        }
        if (endpointDescriptor->bEndpointAddress == 0x3) {
            IOLog("outPipe found");
            ret = ivars->interface->CopyPipe(endpointDescriptor->bEndpointAddress, &ivars->outPipe);
        }
    }
```

Читаем и пишем

```
// Sync read/write

ret = ivars->inPipe->IO(buffer, (uint32_t)bufferLength, bytesTransferred, 1000 * 3);

// Async read/write

// In .iig
virtual void ReadComplete(OSAction *action,
    IOReturn status,
    uint32_t actualByteCount,
    uint64_t completionTimestamp
) TYPE(IOUSBHostPipe::CompleteAsyncIO);

// In ::Start
OSAction* readCompletionCallback;
ret = CreateActionReadComplete(0, &ivars->readCompletionCallback);

// r/w
ret = ivars->inPipe->AsyncIO(buffer, (uint32_t)bufferLength, completion, 1000 * 3);
```

Какими данными будем обмениваться?

1. Отправляем байт команды
2. Получаем сишную структуру
3. Но можно делать как хотите: json, protobuf, etc

```
1  typedef struct {
2      uint8_t ssid[33];
3      int8_t rssi;
4  } wifi_network;
```

Как достучаться до драйвера из приложения?

```
#include <DriverKit/IOUserClient.iig>

class WiFiScannerDriverClient: public IOUserClient {
public:
...
    virtual kern_return_t ExternalMethod(
                                         uint64_t selector,
                                         IOUserClientMethodArguments* arguments,
                                         const IOUserClientMethodDispatch* dispatch,
                                         OSObject* target,
                                         void* reference
                                         ) override;

    virtual kern_return_t CopyClientMemoryForType(
                                                  uint64_t type,
                                                  uint64_t *options,
                                                  IOMemoryDescriptor **memory
                                                  ) override;
```

External method args

```
1  struct IOUserClientMethodArguments {
2      uint64_t                        version;
3      uint64_t                        selector;
4      OSAction                      * completion;
5      const uint64_t                * scalarInput;
6      uint32_t                        scalarInputCount;
7      OSData                        * structureInput;
8      IOMemoryDescriptor            * structureInputDescriptor;
9      uint64_t                      * scalarOutput;
10     uint32_t                        scalarOutputCount;
11     OSData                        * structureOutput;
12     IOMemoryDescriptor            * structureOutputDescriptor;
13     uint64_t                        structureOutputMaximumSize;
14     uint64_t                        __reserved[30];
15 };
```

В коде приложения

```
    private lazy var asyncCallback: IOAsyncCallback = { (
        _ refcon: UnsafeMutableRawPointer?,
        _ result: IOReturn,
        _ args: UnsafeMutablePointer<UnsafeMutableRawPointer?>?,
        _ numArgs: UInt32
    ) -> Void in
        Task.detached(priority: .userInitiated) {
            DriverService.newDataNotifications.forEach { $0() }
        }
    }

    private func setupAsyncCommunication() {
        let globalRunLoop = CFRunLoopGetCurrent()
        CFRunLoopAddSource(globalRunLoop, runLoopSource?.takeUnretainedValue(), .defaultMode)

        withUnsafePointer(to: asyncCallback) { asyncPtr in
            asyncPtr.withMemoryRebound(
                to: io_user_reference_t.self,
                capacity: MemoryLayout<io_user_reference_t>.size
            ) { castedPtr in
                withUnsafeMutableBytes(of: &asyncRef) { ptr in
                    let ptr = ptr.baseAddress?.assumingMemoryBound(to: io_user_reference_t.self)
                    ptr?[kIOAsyncCalloutFuncIndex] = castedPtr.pointee
                }
            }
        }
    }

    private func registerAsyncCallback() {
        var inputStruct = CommunicationData()
        var outputStruct = CommunicationData()
        let inputSize = MemoryLayout<CommunicationData>.size
        var outputSize = MemoryLayout<CommunicationData>.size


        withUnsafeMutablePointer(to: &asyncRef) { ptr in
            ptr.withMemoryRebound(to: UInt64.self, capacity: MemoryLayout<UInt64>.size) { asyncPtr in
                let result = IOConnectCallAsyncStructMethod(
                    connection,
                    MessageType_RegisterAsyncCallback,
                    machNotificationPort,
                    asyncPtr,
                    UInt32(kIOAsyncCalloutCount),
                    &inputStruct,
                    inputSize,
                    &outputStruct,
                    &outputSize
                )
```

Чуть проще

1. IOServiceGetMatchingService
2. IOServiceOpen
3. IOConnectCallAsyncStructMethod
   (IOConnectCallScalarMethod, etc)
4. IOConnectMapMemory64

Мини итог

Создали драйвер

01.

Научились мэтчить

- Plist
- Ioreg
- Docs

02.

Драйвер ↔️ устройство

- Нашли эндпоинты
- Пайпы
- IO/AsyncIO

03.

Приложение ↔️ Драйвер

- Колбэки
- Общий буфер

Часть 4.2

Нюансы

macOS

01 com.apple.developer.driverkit.allow-any-userclient-access

02 com.apple.developer.driverkit.userclient-access

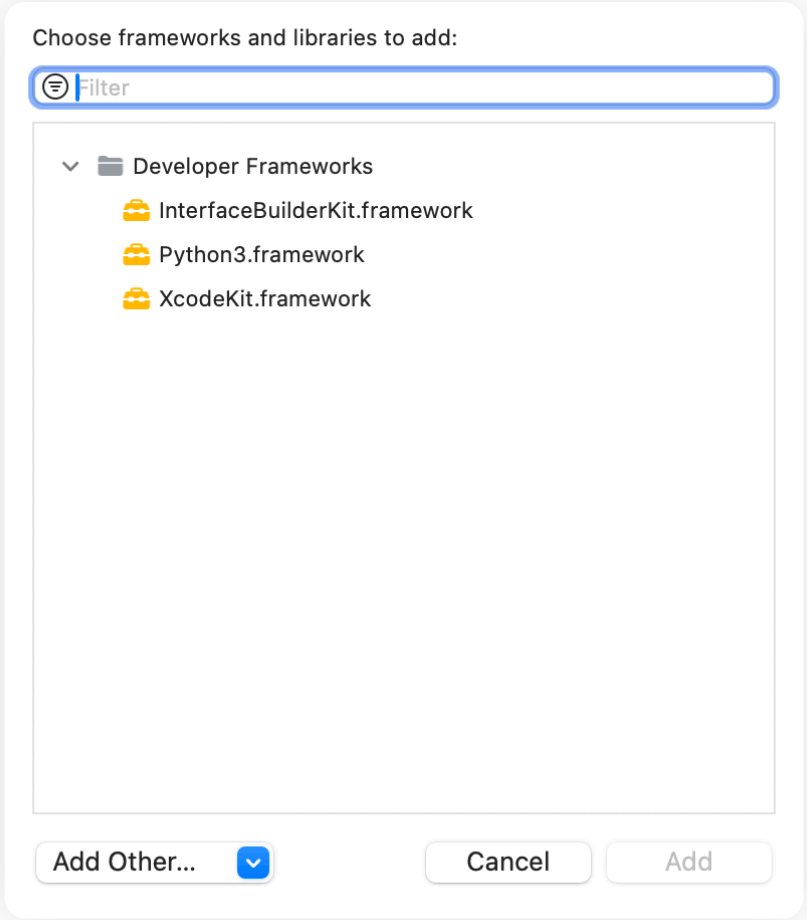
03 Sign to run locally

04 AD_HOC_CODE_SIGNING_ALLOWED = YES

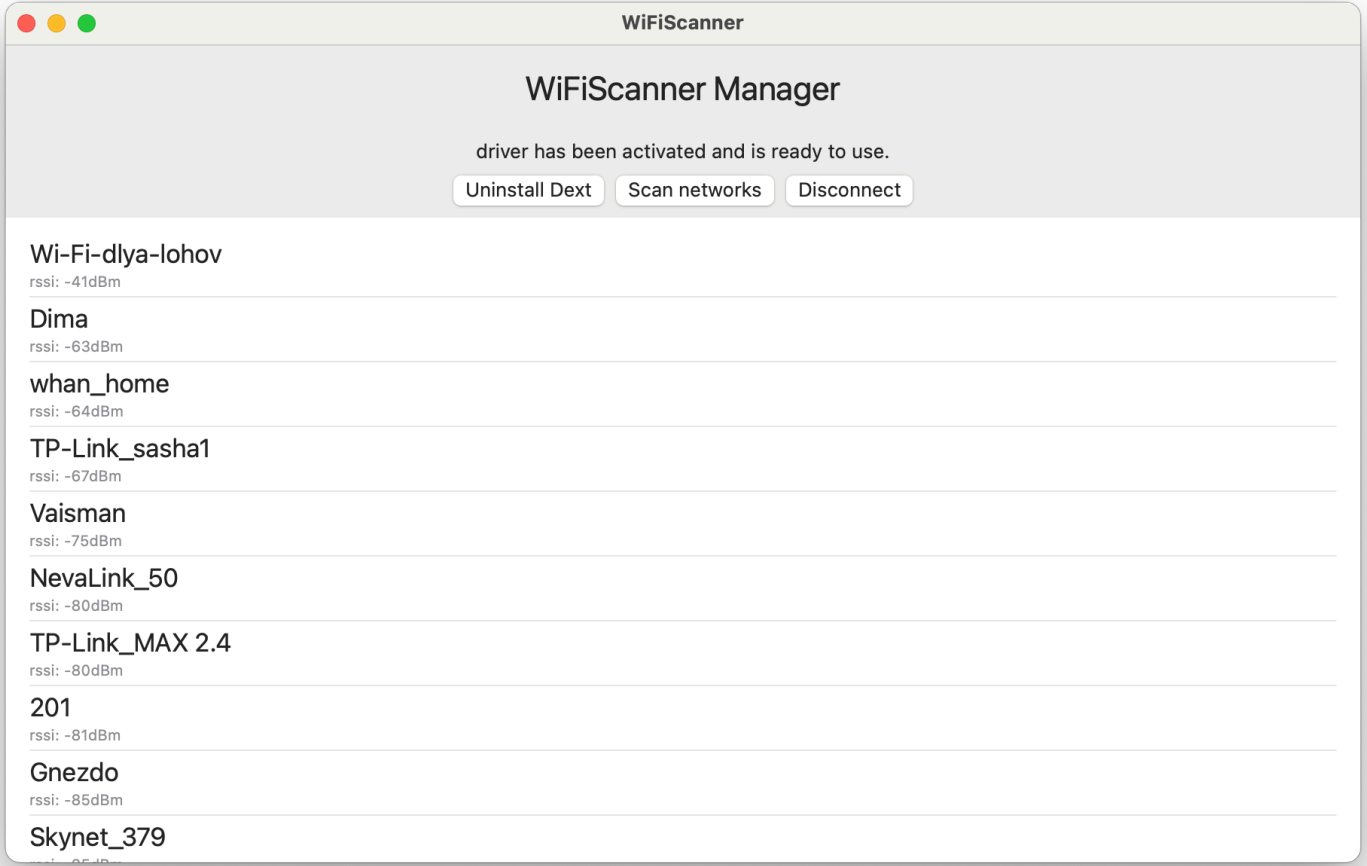
Xcode 15

Нужно линковать USBDriverKit

/Applications/Xcode.app/Contents/Developer/Platforms/DriverKit.platform/Developer/SDKs/DriverKit.sdk/System/DriverKit/System/Library/Frameworks



Успех неизбежен

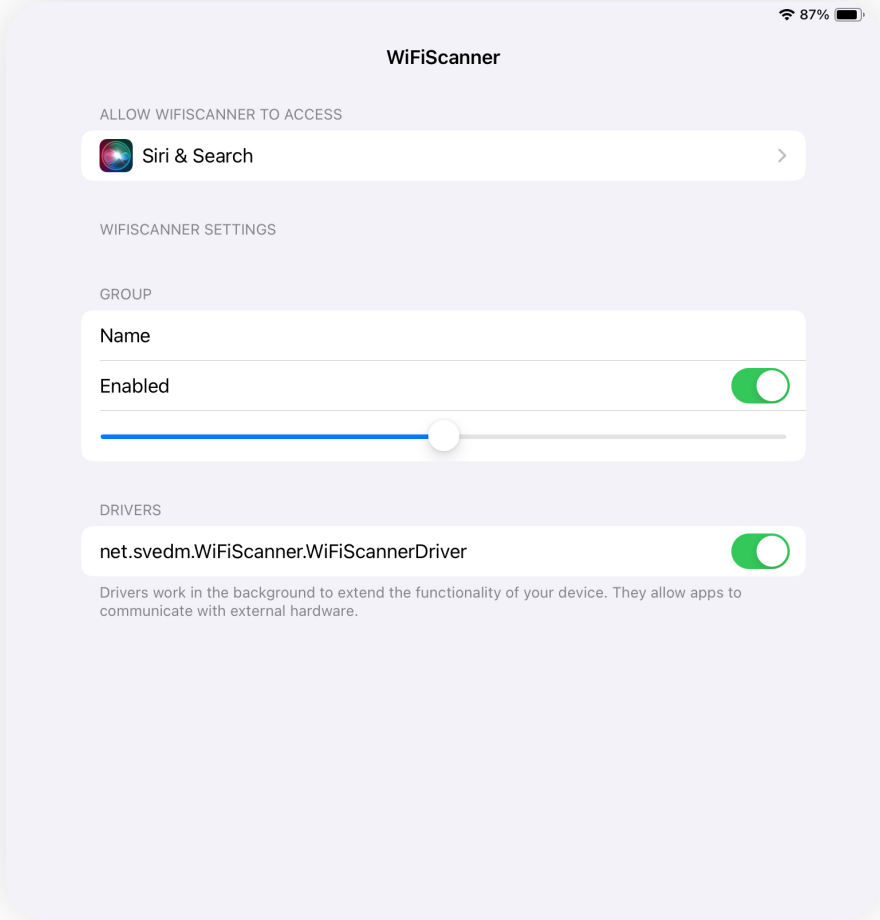


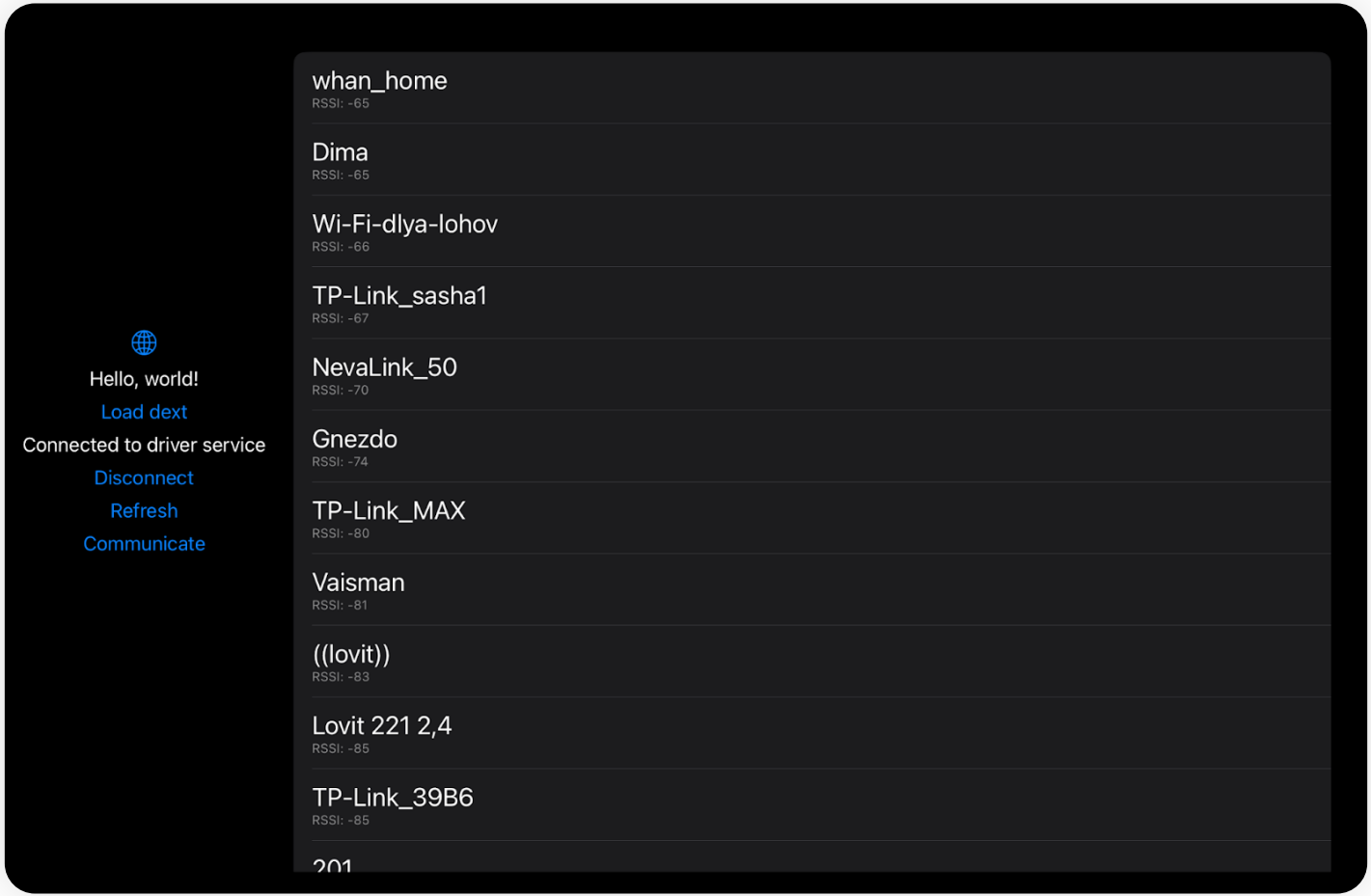
Часть 4.3

Запускаем на iPad

Запускаем на iPad

1. Убираем com.apple.developer.driverkit.allow-any-userclient-access
2. Убираем “Sign to Run Localy”
3. Нужен bridging header с IOKit
4. Добавить Settings.bundle, если в настройках приложения нет драйвера




Hello, world!
Load dext
Connected to driver service
Disconnect
Refresh
Communicate
whan_home
RSSI: -65
Dima
RSSI: -65
Wi-Fi-dlya-lohov
RSSI: -66
TP-Link_sasha1
RSSI: -67
NevaLink_50
RSSI: -70
Gnezdo
RSSI: -74
TP-Link_MAX
RSSI: -80
Vaisman
RSSI: -81
((lovit))
RSSI: -83
Lovit 221 2,4
RSSI: -85
TP-Link_39B6
RSSI: -85
201

Часть 5

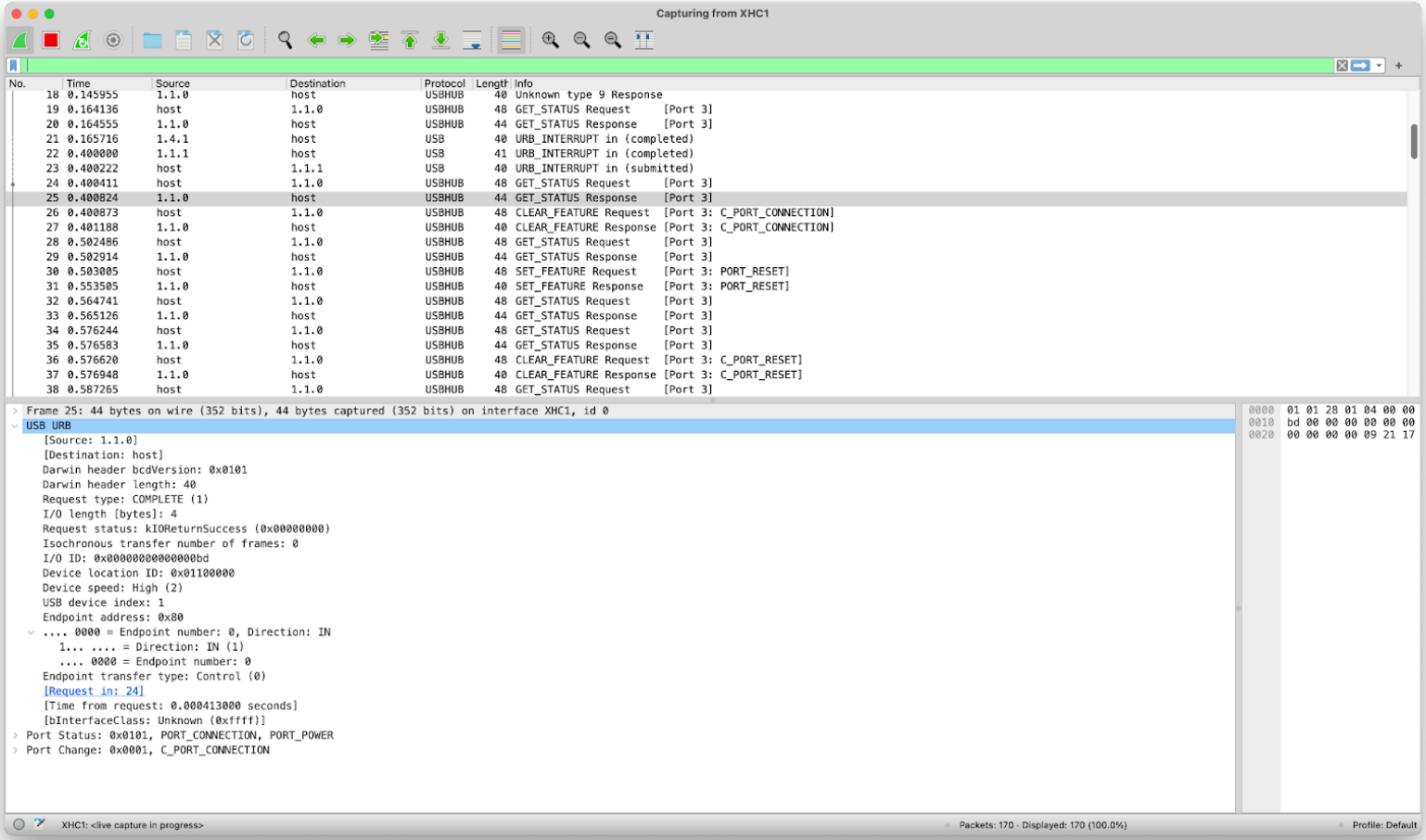
Отладка

System Extension

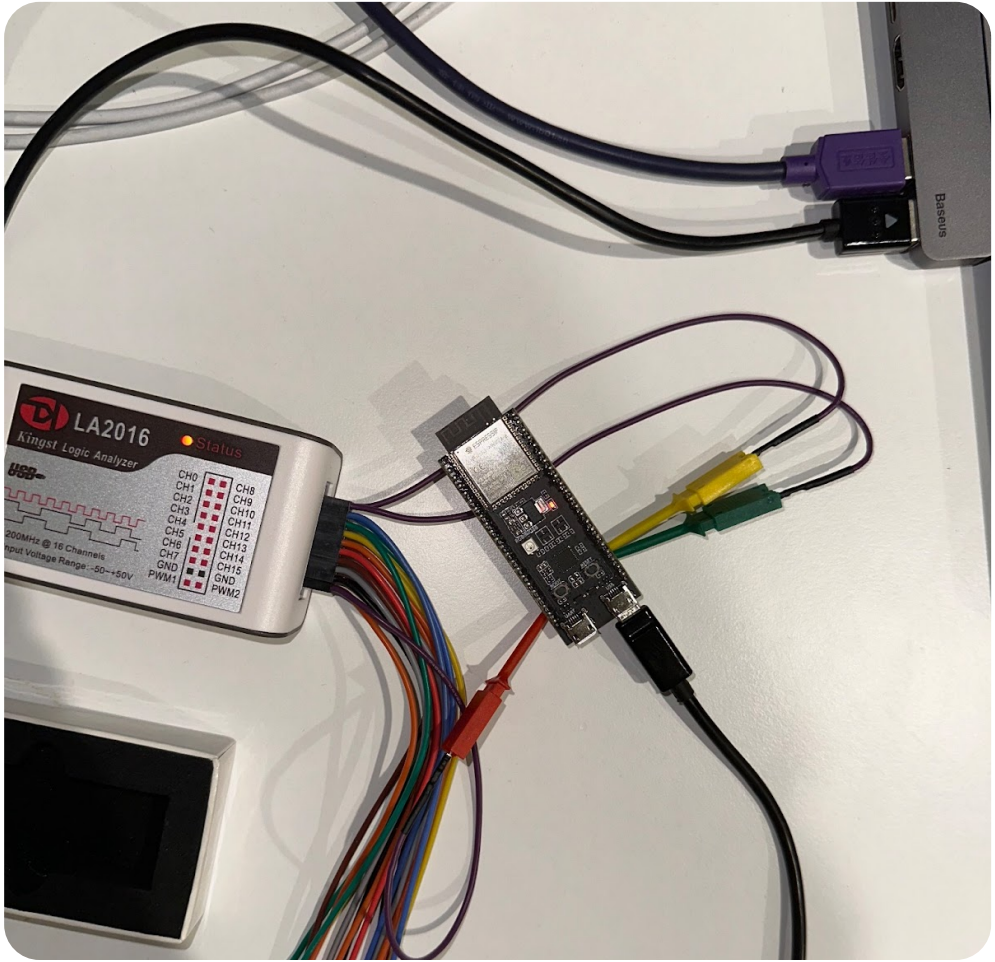
1. ps ax | grep -i “%DRIVER_NAME%”
2. → attach to PID
3. log show --predicate 'sender == "sysextd" or sender CONTAINS "com.example"' --info --debug --last 1h
4. ioreg

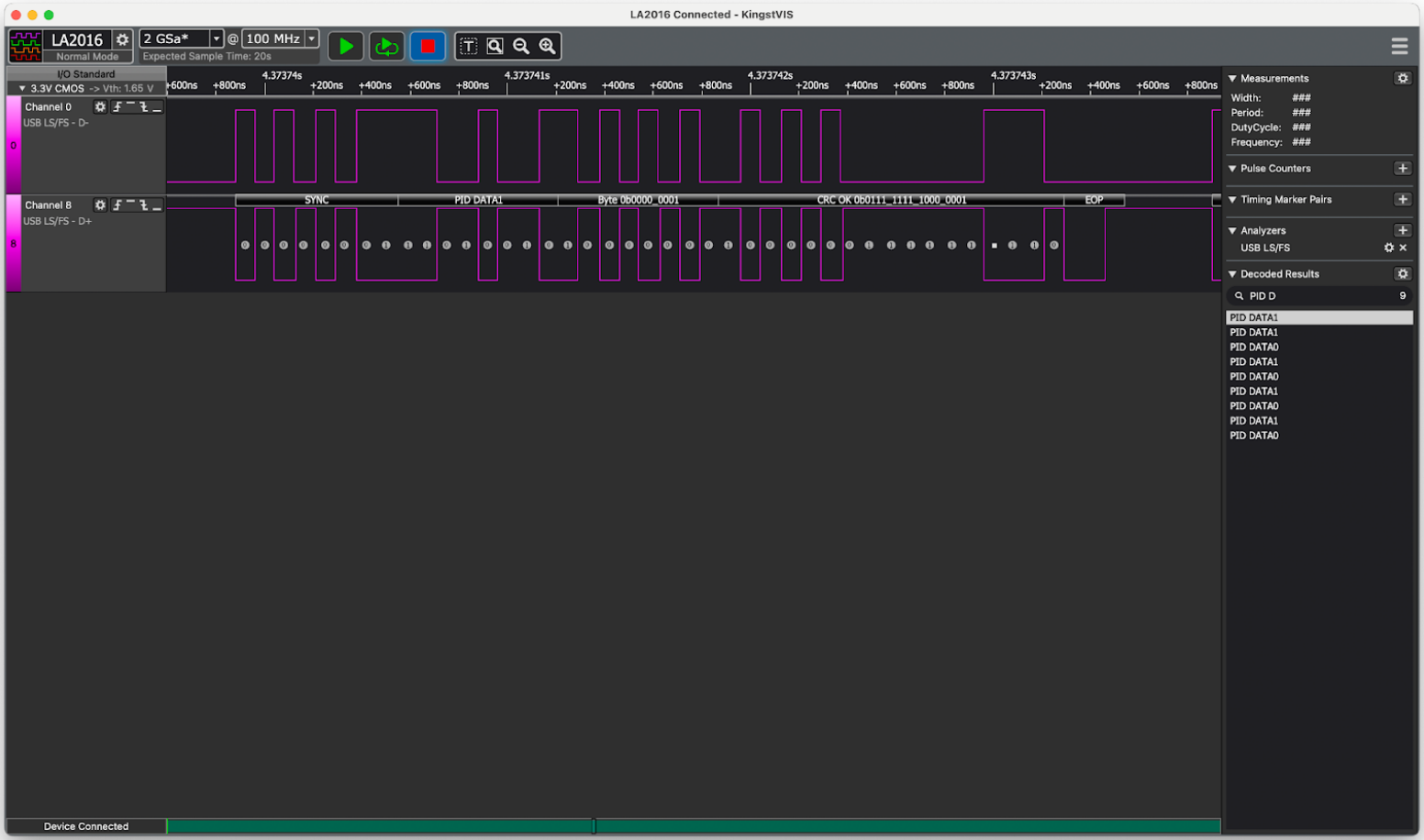
Wireshark

1. ioreg -w0 -rc AppleUSBHostController
2. +-o AppleT6000USBXHCI@01000000 <class AppleT6000USBXHCI, id 0x1000005f3, registered, matched, active, busy 0 (534 ms), retain 96>
3. sudo ifconfig XHC1 up


Capturing from XHC1
No. Time Source Destination Protocol Length Info
18 0.145955 1.1.0 host USBHUB 40 Unknown type 9 Response
19 0.164136 host 1.1.0 USBHUB 48 GET_STATUS Request [Port 3]
20 0.164555 1.1.0 host USBHUB 44 GET_STATUS Response [Port 3]
21 0.165716 1.4.1 host USB 40 URB_INTERRUPT in (completed)
22 0.400000 1.1.1 host USB 41 URB_INTERRUPT in (completed)
23 0.400222 host 1.1.1 USB 40 URB_INTERRUPT in (submitted)
24 0.400411 host 1.1.0 USBHUB 48 GET_STATUS Request [Port 3]
25 0.400824 1.1.0 host USBHUB 44 GET_STATUS Response [Port 3]
26 0.400873 host 1.1.0 USBHUB 48 CLEAR_FEATURE Request [Port 3: C_PORT_CONNECTION]
27 0.401188 1.1.0 host USBHUB 40 CLEAR_FEATURE Response [Port 3: C_PORT_CONNECTION]
28 0.502486 host 1.1.0 USBHUB 48 GET_STATUS Request [Port 3]
29 0.502914 1.1.0 host USBHUB 44 GET_STATUS Response [Port 3]
30 0.503005 host 1.1.0 USBHUB 48 SET_FEATURE Request [Port 3: PORT_RESET]
31 0.553505 1.1.0 host USBHUB 40 SET_FEATURE Response [Port 3: PORT_RESET]
32 0.564741 host 1.1.0 USBHUB 48 GET_STATUS Request [Port 3]
33 0.565126 1.1.0 host USBHUB 44 GET_STATUS Response [Port 3]
34 0.576244 host 1.1.0 USBHUB 48 GET_STATUS Request [Port 3]
35 0.576583 1.1.0 host USBHUB 44 GET_STATUS Response [Port 3]
36 0.576620 host 1.1.0 USBHUB 48 CLEAR_FEATURE Request [Port 3: C_PORT_RESET]
37 0.576948 1.1.0 host USBHUB 40 CLEAR_FEATURE Response [Port 3: C_PORT_RESET]
38 0.587265 host 1.1.0 USBHUB 48 GET_STATUS Request [Port 3]
Frame 25: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface XHC1, id 0
USB URB
[Source: 1.1.0]
[Destination: host]
Darwin header bcdVersion: 0x0101
Darwin header length: 40
Request type: COMPLETE (1)
I/O length [bytes]: 4
Request status: kIOReturnSuccess (0x00000000)
Isochronous transfer number of frames: 0
I/O ID: 0x00000000000000bd
Device location ID: 0x01100000
Device speed: High (2)
USB device index: 1
Endpoint address: 0x80
.... 0000 = Endpoint number: 0, Direction: IN
1... .... = Direction: IN (1)
.... 0000 = Endpoint number: 0
Endpoint transfer type: Control (0)
[Request in: 24]
[Time from request: 0.000413000 seconds]
[bInterfaceClass: Unknown (0xffff)]
Port Status: 0x0101, PORT_CONNECTION, PORT_POWER
Port Change: 0x0001, C_PORT_CONNECTION
0000 01 01 28 01 04 00 00
0010 bd 00 00 00 00 00 00
0020 00 00 00 00 09 21 17
XHC1: <live capture in progress>
Packets: 170 · Displayed: 170 (100.0%)
Profile: Default

Logic analyzer




LA2016 Connected - KingstVIS
LA2016
Normal Mode
2 GSa*
@ 100 MHz
Expected Sample Time: 20s
I/O Standard
3.3V CMOS -> Vth: 1.65 V
Channel 0
USB LS/FS - D-
Channel 8
USB LS/FS - D+
SYNC
PID DATA1
Byte 0b0000_0001
CRC OK 0b0111_1111_1000_0001
EOP
Measurements
Width: ###
Period: ###
DutyCycle: ###
Frequency: ###
Pulse Counters
Timing Marker Pairs
Analyzers
USB LS/FS
Decoded Results
PID D
9
PID DATA1
PID DATA1
PID DATA0
PID DATA1
PID DATA0
PID DATA1
PID DATA0
PID DATA1
PID DATA0
Device Connected

Часть 6 (эпическая)

А как же iPhone?

Jailbreak

```
<dict>
    <key>com.apple.security.exception.iokit-user-client-class</key>
    <array>
        <string>AppleUSBHostDeviceUserClient</string>
        <string>AppleUSBHostInterfaceUserClient</string>
    </array>
    <key>com.apple.system.diagnostics.iokit-properties</key>
    <true/>
</dict>
```

Итоги

1

Узнали, как работает USB

- Какие есть классы устройств, устройства могут быть составными
- Как устроена передача данных: interfaces, endpoints, pipes

2

Разобрались, как создать себе устройство для тестирования

- Можно использовать девборды ESP32 и Raspberry PI Pico W
- Писать придется на C

Итоги

1. Узнали, какие бывают драйверы
2. Разобрались, как происходит сопоставление устройства и драйвера
3. Получили общее понимание что делать, чтобы создать свой драйвер
4. Разобрались, как запускать это все на iPadOS
5. Запомнили, что можно найти эту презу: в ней есть много лайфхаков, если что-то не работает
6. Разобрались с дебагом

Tg: @svedm

Спасибо!