**Документация для USB-Коммуникатора**

Quick start:

1. Клоним котлин проект отсюда <https://github.com/Kekega/usb_comm_kotlin>

git clone <https://github.com/Kekega/usb_comm_kotlin.git>

1. Ставим приложение на телефон (в Android Studio)
2. Отключаемся от Android Studio
3. Клоним питон скрипт отсюда <https://github.com/alien-agent/USB-Communicator-Script>

git clone <https://github.com/alien-agent/USB-Communicator-Script.git>

1. Подключаем телефон
2. Запускаем питон скрипт в нужном режиме (read или write)

### python [usbmachine.py](https://github.com/alien-agent/USB-Communicator-Script/blob/main/usbmachine.py) read

Если используется ОС Linux:

### sudo -E python [usbmachine.py](https://github.com/alien-agent/USB-Communicator-Script/blob/main/usbmachine.py) read

1. Готово

Подробнее

Данное приложение позволяет коммуницировать с ПК на Linux, MacOS или Windows (двустороняя коммуникация). Коммуникация с ПК происходит с помощью Android Accessory Protocol, в данном случае [Python-скрипт](https://github.com/alien-agent/USB-Communicator-Script) на ПК презентует себя как акссессуар. Поиск доступного устройства происходит полностью автоматически, никаких дополнительных действий для переключения между режимами PC выполнять не нужно.

## Подключение к ПК

* Подключить смартфон с установленным приложением к ПК с помощью кабеля. **В случае Windows необходимо установить правильные драйвера**, чтобы смартфон определялся как USB-устройство, а не MTP. На MacOS и Linux дополнительных действий не требуется.
* Запустить скрипт в соответствии с инструкцией в режиме read или write.
* Дождаться, пока скрипт переведет смартфон в Accessory Mode.
* Должен сработать USB-Filter, который автоматически предложит открыть данное приложение. Также его можно открыть вручную.
* Приложение запросит разрешение на доступ к USB-устройству. Если приложение было открыто через IntentFilter, доступ будет выдан автоматически. В случае успеха, в приложении будет отображаться заголовок Connected to Accessory.
* Готово!

Запуск приложения и подключение кабеля можно осуществлять в любом порядке, приложение самостоятельно увидит, что устройство появилось в списке доступных устройств и запросит доступ. бб

## Возможные проблемы

Python-скрипт может выдавать ошибку no backend available. Это означает что в системе не установлена libusb, либо установлены неправильные драйвера для смартфона (в случае Windows). Также эта ошибка возможна в случае, если библиотеке pyusb не удалось найти путь к libusb. В таком случае его нужно указать вручную, см. пример в начале файла для MacOS с процессором Apple Silicon.

Вылетает на шаге Triggering accessory mode - нет правильных драйверов (Windows). Телефон не должен определяться как MTP/FTP/... Берем программу [https://zadig.akeo.ie](https://zadig.akeo.ie/) и прошиваем на телефон драйвер libusb (libusb предварительно нужно установить).

### Нет прав (для ОС Linux). Запускаем так: sudo –E python [usbmachine.py](https://github.com/alien-agent/USB-Communicator-Script/blob/main/usbmachine.py) read

**Документация для приложения на android**

<https://github.com/Kekega/usb_comm_kotlin>

**MainActivity.kt**

Главная активность приложения, представляющая пользовательский интерфейс и взаимодействие с USB-устройствами.

**SerialEngine.kt**

Управляет подключением и взаимодействием с USB-серийным устройством.

### **UsbEngine.kt**

### Обеспечивает взаимодействие с USB-устройствами и их обработку.

### **AccessoryEngine.kt**

Управляет подключением и взаимодействием с USB-аксессуарами.

### **IUsbCallback**

Определяет методы для обработки событий подключения и отключения USB-устройств, а также приема данных.

**Документация для python приложения**

Python-скрипт, разработанный для взаимодействия с устройствами USB через протоколы USB-коммуникации. Скрипт поддерживает различные режимы работы, позволяя пользователям записывать данные на устройство, считывать данные с устройства или взаимодействовать с Arduino, подключенным через USB.

**Требования**

- Python 3.x

- Необходимые пакеты Python:

- `typer`

- `rich`

- `usb`

- `serial`

**Использование**

*Запуск Скрипта*

git clone <https://github.com/alien-agent/USB-Communicator-Script.git>

cd usb-communicator-script

python usbmachine.py [режим]

*Режимы*

Скрипт поддерживает следующие режимы:

- *Режим Чтения*: Считывание данных с подключенного USB-устройства.

- *Режим Записи*: Запись данных на подключенное USB-устройство.

- *Режим Записи для Arduino*: Взаимодействие с Arduino, подключенной через USB.

*Примеры*

python script\_name.py read

python script\_name.py write

python script\_name.py write-arduino

**Структура Скрипта**

**Зависимости**

- `os`: предоставляет возможность взаимодействия с операционной системой.

- `platform`: позволяет получать информацию о платформе системы.

- `time`: используется для добавления задержек в скрипт.

- `dataclasses`: используется для создания структурированных классов с автоматически добавленными специальными методами.

- `enum`: предоставляет поддержку перечислений.

- `serial`: позволяет взаимодействовать с устройствами чрез serial port.

- `typer`: библиотека для создания интерфейсов командной строки.

- `rich`: библиотека для улучшения вывода в терминале.

**Классы**

*Перечисление `AppMode`*

- Определяет различные режимы приложения: Чтение, Запись, ЗаписьArduino.

*Класс `App`*

- Атрибуты:

- `device`: Объект USB-устройства.

- `console`: Экземпляр класса Rich Console для улучшенного вывода в терминал.

- Методы:

- `select\_device()`: Выбор USB-устройства для взаимодействия.

- `prepare\_device()`: Подготовка выбранного устройства выполнением необходимых команд.

- `set\_protocol()`: Установка протокола связи для устройства.

- `send\_accessory\_parameters()`: Отправка параметров аксессуара на устройство.

- `set\_accessory\_mode()`: Запуск режима аксессуара на устройстве.

- `accept\_data()`: Считывание и вывод данных с устройства.

- `write()`: Запись введенных пользователем данных на подключенное USB-устройство.

- `write\_arduino()`: Взаимодействие с Arduino, подключенной через USB.

*Функция `main`*

- Основная точка входа в скрипт.

- Анализирует аргументы командной строки и выполняет соответствующий режим.

**Примечание и устранения неполадок**

- Скрипт использует библиотеки `usb` и `serial` для USB-коммуникации. Убедитесь, что эти библиотеки установлены перед запуском скрипта.

- При использовании программы при использовании ОС Linux может потребоваться запуск от имени администратора:

sudo python script\_name.py read

- Для взаимодействия по USB проводу при использовании ОС Windows может потребоваться установка необходимых для этого драйверов. Это можно сделать с помощью данной программы: <https://zadig.akeo.ie/>