

**РТУ МИРЭА (ТУ)**

Центр проектирования интегральных схем, устройств наноэлектроники и  
микросистем

**Методическое пособие по лабораторным  
работам**

«Багет-ПЛК1-01»  
(ОС Linux, Debian 10)



Москва

2023

# Оглавление

Введение	2
1 Установка ОС на целевую платформу	9

# Введение

Методическое пособие описывает основные шаги по разработке программного обеспечения для встраиваемых систем. В качестве целевой платформы используется плата «Багет ПЛК1-01» с установленным микроконтроллером Комдив-МК К5500ВК018, производства ФГУ ФНЦ НИИСИ РАН. В состав К5500ВК018 входит микропроцессорное ядро с архитектурой MIPS64, включая сопроцессор вещественной арифметики (IEEE754), кэш-память первого уровня (16+16 кбайт), кэш-память второго уровня (128 кбайт), буфер трансляции виртуальных адресов на 64 адреса, контроллер динамической памяти DDR3L, восемь таймеров-счетчиков, сторожевой таймер, часы реального времени, два контроллера Ethernet 10/100, контроллеры последовательных интерфейсов RS-232C (4 шт.), I2C (2 шт.), SPI (3 шт.), QSPI, CAN 2.0 (2 шт.), USB 2.0. Серийный выпуск начат в 2021 году.

ПЛК1-01 выполнен в виде печатной платы без корпуса, на которой расположены соединители для подключения первичного электропитания и внешних устройств, органы управления (пользовательские кнопки SW1, SW2), светодиодные индикаторы (VD1 – VD5), переключики для управления режимами работы ПЛК1-01 (SA4, SA5, SA6, SA7, SA9).

Назначение	Вывод процессора
Кнопка SW2	GPIO D 6
Зелёный светодиод VD2	GPIO D 4
Жёлтый светодиод VD5	GPIO D 5

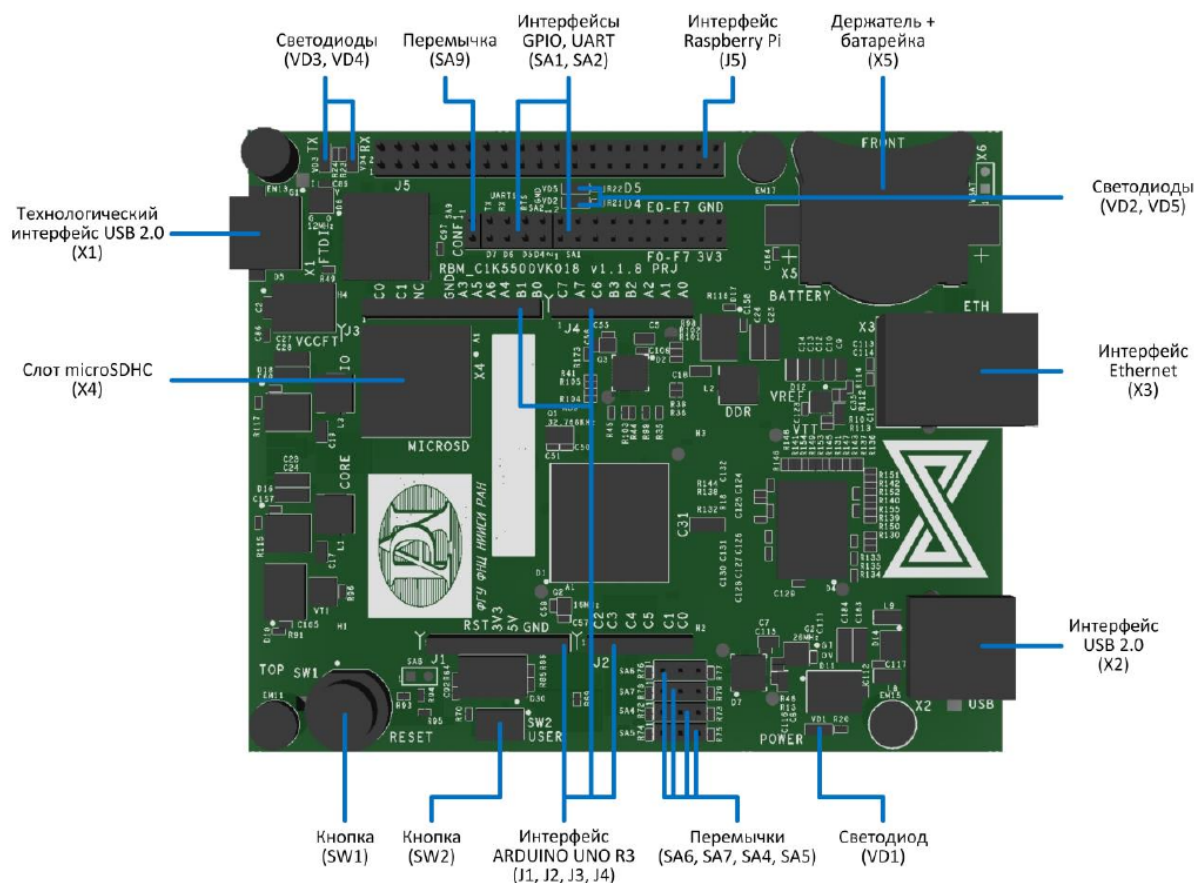


Рисунок 1. Расположение портов ввода-вывода

Для организации питания часов реального времени (RTC) микроконтроллера K5500BK018 при отсутствии основного питания предусмотрена установка автономного элемента питания – батарейки типа CR2032 +3 В. Расположение соединителей, органов управления и индикаторов ПЛК1-01 приведено на рисунке 1, назначение выводов на некоторых из разъёмов приведены на рисунке 2.

## Переходная плата

Для удобства подключения к интерфейсам целевой платы, была разработана переходная плата, с выведенными сигнальными линиями, сгруппированными по интерфейсам (рисунок 3).

UART2 можно использовать как для работы с интерфейсом UART2,

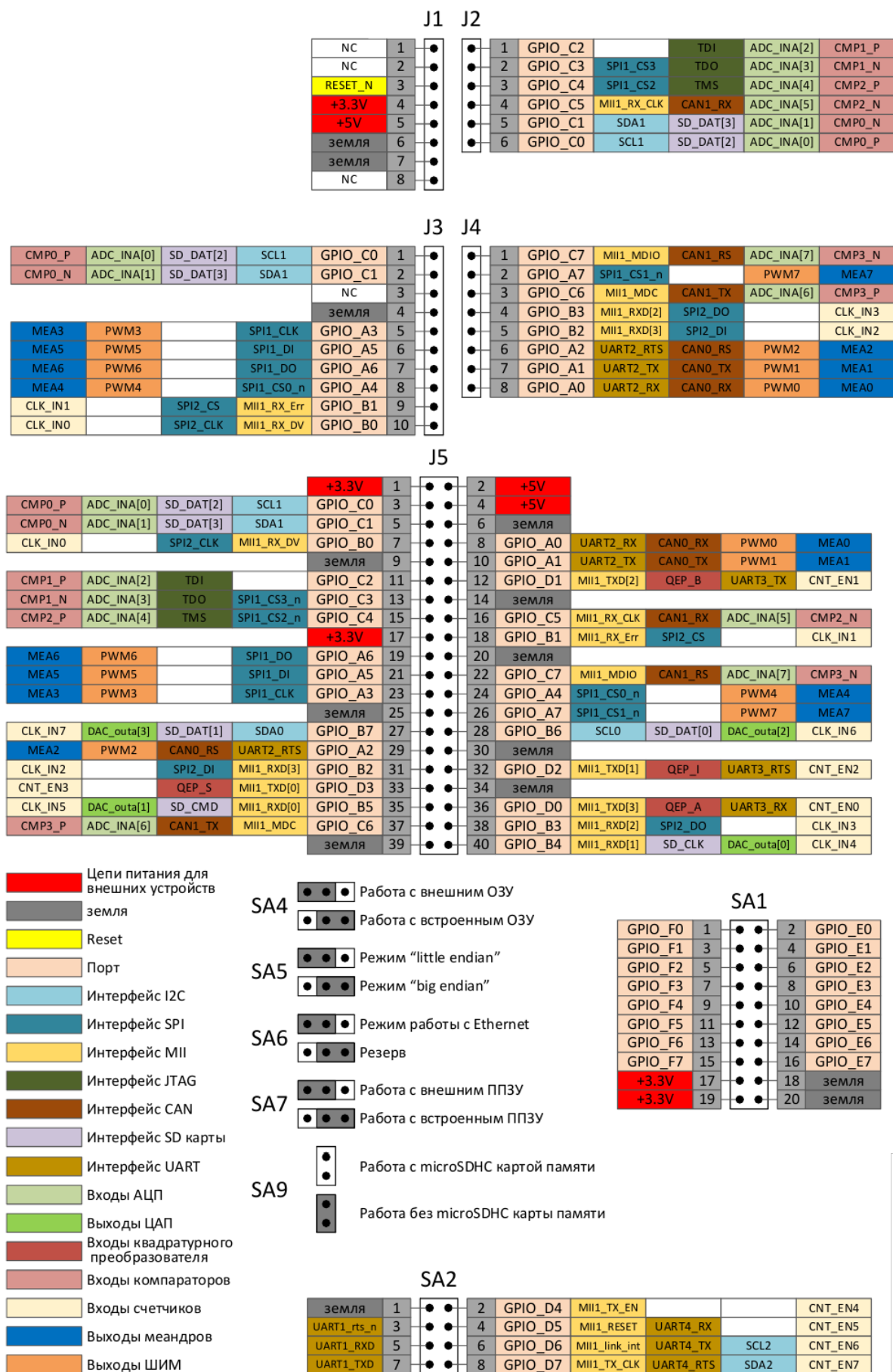


Рисунок 2. Разводка выводов процессора по разъёмам ввода-вывода

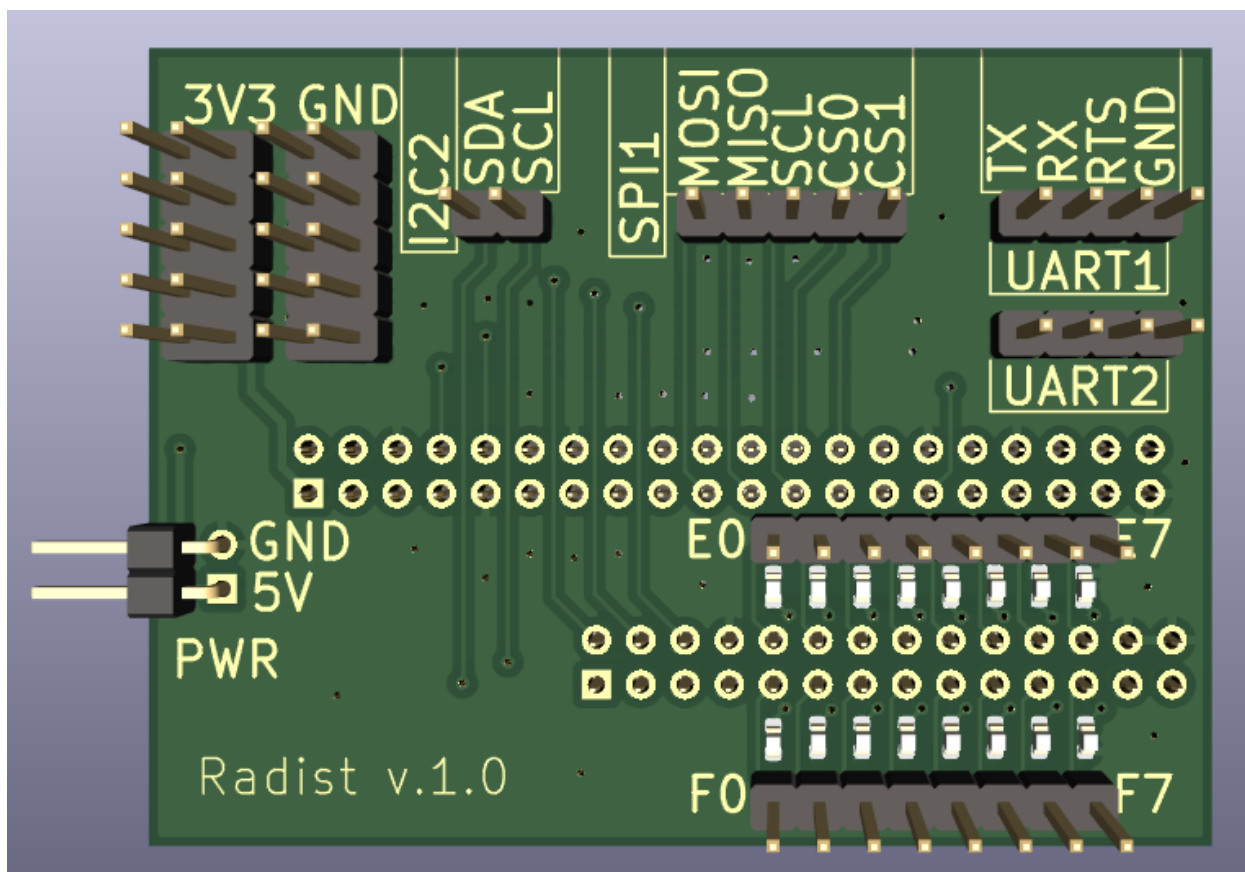


Рисунок 3. Переходная плата

так и для работы с интерфейсом CAN0. Выводы GPIO банков E и F дублируются на светодиоды для отладочных целей. При необходимости, можно подключать к выводам кнопки, или использовать их для формирования управляющих сигналов для модулей. Также обратите внимание, что одна из сигнальных линий I2C интерфейса используется кнопкой SW2 на плате, по этой причине не стоит использовать кнопку при активации этого интерфейса.

## Как запускается Linux?

Для запуска ОС Linux в общем случае должны пройти следующие этапы:

1. ROM BOOT — запуск встроенного загрузчика, определяющего ис-

точник дальнейшей загрузки. Этот код нельзя исправить его работа и формат образа (заголовки, значение бит и прочее) зависит от производителя.

2. First Stage Bootloader (FSBL) — загрузчик первой ступени, задача этого загрузчика, начальная инициализация процессора, и загрузка пользовательского кода в DDR память. Данный загрузчик опциональный, и у некоторых производителей может отсутствовать. Работает из встроенной RAM памяти процессора.
3. Second Stage Bootloader (SSBL) — это загрузчик второго уровня, задача которого заключается в подготовке системы к запуску ядра Linux, с пользовательскими параметрами (точка монтирования rootfs, скорость работы терминала для логов и прочее). Наиболее распространённый вариант загрузчика проекта Das U-Boot. В целевой плате используется другой загрузчик, набирающий популярность — Bearbox.
4. Kernel — на этом этапе за работу процессора отвечает ядро, происходит финальная конфигурация, активация дополнительных ядер (если они есть и активирована функция многоядерной поддержки), монтирование rootfs, запуск сервисов и прочее.

## Описание окружения

Для работы с платой используется виртуальная машина на базе Ubuntu 20.04. На виртуальной машине установлены пакеты для компиляции кода под Linux для MIPS64 архитектуры `mips64el-linux-gnu-gcc` (C компилятор) и `mips64el-linux-gnu-g++` (C++ компилятор), компилятор дерева устройств `device-tree-compiler`, архив с заголовочными файлами ядра Linux установленного на плате, утилита `debootstrap`, IDE VSCode и прочие полезные инструменты.

Имя пользователя **student** пароль **usrstudent**. Пользователь поддерживает работу через `sudo` (получение прав суперпользователя он же администратор).

В конце некоторых работ есть задания для самостоятельного выполнения. При этом есть простые задачи, предполагающие незначительное изменение кода, а также есть помеченные \*. Последние выполняются по желанию, при этом нужно быть готовым, что придётся активно пользоваться поиском, и расходовать клетки серого вещества.

**ВНИМАНИЕ!** Запомните, что далее по тексту, если просят выполнить команду, в начале которой написано **\$** , значит команду нужно выполнить на целевой плате, а если с **#** - в консоли виртуальной машины. При этом **\$** или **#** писать не нужно!

Если команда заканчивается символом **\** и на следующей строке есть запись, то это запись относится к одной команде, Вы можете записать как видите, так как символ **\** в консоли Linux означает многострочную команду или, опустив символ **\**, продолжить вводить текст приведённый на следующей строке методички.

### Примеры:

Авторы хотят, что бы Вы в консоли виртуальной машины (ВМ) записали `echo Hello` и нажали кнопку Enter:

---

```
# echo Hello
```

---

Авторы хотят, что бы Вы в консоли целевой платы записали `echo FOOBAR` и нажали кнопку Enter:

---

```
$ echo FOOBAR
```

---

Авторы хотят, что бы Вы в консоли виртуальной машины (ВМ) написали очень длинную команду (да-да, вечер не обещает быть томным) и



нажали кнопку Enter:

---

```
# echo \  
FOOBAR
```

---

# Лабораторная работа № 1

## Установка ОС на целевую платформу

**Цель:** Установить дистрибутив Debian 10 на целевую платформу.

**Описание:** Нужно понимать, что ОС Linux делится на две части. Ядро (Kernel) и файловую систему (rootfs). Ядро определяет какие возможности по взаимодействию с железом доступны пользователю и приложениям. Rootfs определяет какие инструменты есть в системе, порядок инициализации, запуска сервисов и прочие прикладные штуки. Дистрибутив это в большей степени наполнение rootfs. Не каждый дистрибутив имеет поддержку иной от x86 архитектур.

Для установки debian воспользуемся инструментом под названием debootstrap, который отвечает за создание rootfs. Подробнее про этот инструмент, и возможности по его управлению, Вы можете найти на wiki проекта Debian.

### 1.1 Подготовка rootfs

**1.1.1** Запустите виртуальную машину. Логин и пароль для входа: student / usrstudent.

### 1.1.2 Запустите консоль нажав комбинацию клавиш на клавиатуре **Ctrl+Alt+T**

**Внимание!** В консоли есть возможность автоматического продолжения ввода. Для этого необходимо ввести первые символы команды, или пути, и нажать клавишу TAB. Ввод продолжится до тех пор, пока не появится неопределённостью (к примеру у Вас есть два файла `foo_bar1` и `foo_bar2`, при нажатии TAB будет вставлен текст до цифры). Двойное нажатие TAB приведёт к выводу всех возможных вариантов продолжения, если таковые есть.

### 1.1.3 Создайте и перейдите в рабочий каталог в котором будет создан образ `rootfs`.

---

```
# mkdir -p $BAGET/lab_01
# cd $BAGET/lab_01
```

---

### 1.1.4 Запустите утилиту `debootstrap` (при необходимости введите пароль `usrstudent`):

---

```
# sudo debootstrap --include=aptitude,nano,wget \
--foreign \
--arch=mips64el buster rootfs
```

---

*--include=A,B,C..* - добавить в сборку указанные пакеты

*--foreign* — только сгенерировать наполнение, применяется когда архитектура на которой запускается утилита отлична от архитектуры назначения.

*--arch=mips64el* — указываем целевую архитектуру

*buster* — версия сборки

*rootfs* — путь к папке назначения, где будут размещены файлы

**1.1.5** Для продолжения установки, нам понадобится утилита `qemu` позволяющая эмулировать различные архитектуры. Скопируем исполняемый файл `qemu`:

---

```
# sudo cp /usr/bin/qemu-mips64el-static ./rootfs/usr/bin
```

---

и перейдём в созданную `rootfs` (привет дедушка контейнеров, `chroot`)

---

```
# sudo chroot ./rootfs
```

---

## 1.2 Настройка rootfs

### 1.2.1 Завершим работу debootstrap

---

```
# export LANG=en_US.UTF-8
# /debootstrap/debootstrap --second-stage
```

---

**1.2.2** Добавим источники для установки ПО, для этого выполним следующие команды

---

```
# echo deb http://ftp.debian.org/debian buster \
main contrib non-free >> /etc/apt/sources.list

# echo deb-src http://ftp.debian.org/debian buster \
main contrib non-free >> /etc/apt/sources.list

# echo deb http://ftp.debian.org/debian buster-updates \
main contrib non-free >> /etc/apt/sources.list

# echo deb-src http://ftp.debian.org/debian buster-updates \
main contrib non-free >> /etc/apt/sources.list
```

---

### 1.2.3 Обновим список, и установим ряд приложений

---

```
# apt-get update
# apt-get install -y dialog sudo less i2c-tools evtest mc \
```

```
openssh-server resolvconf hwinfo net-tools
```

---

### 1.2.4 Зададим пароль для root пользователя

---

```
# passwd root
```

---

После чего введите root (внимание, курсор двигаться не будет, это политика безопасности Linux, при вводе пароля курсор не перемещается, что бы нельзя было установить количество символов). И нажмите Enter

Затем Вас попросят повторить пароль, снова введите root и нажмите Enter

### 1.2.5 Настроим работу сетевого интерфейса со статическим адресом, для этого создадим файл в паке /etc/network/interfaces.d/

---

```
# nano /etc/network/interfaces.d/eth0
```

---

и впишите туда следующие строки:

```
auto eth0
    iface eth0 inet static
    address 192.168.100.200
    netmask 255.255.255.0
    network 192.168.100.0
```

Первая строка определяет когда настраивается интерфейс. Auto означает, что при загрузке ОС. Для USB адаптеров, лучше выбирать allow-hotplug, что бы сократить время загрузки ОС.

Вторая строка определяет, что мы будем использовать статический ip адрес, далее идёт конфигурация адресов.

Выходим из редактора nano, нажав комбинацию клавиш Ctrl+X. Вам будет задан вопрос о сохранении изменения в файле, жмём Y и клавишу Enter.

## 1.2.6 Настроим имя компьютера

```
# echo baget > /etc/hostname
```

## 1.2.7 Завершим настройку rootfs

```
# exit  
# sudo rm ./rootfs/usr/bin/qemu-mips64el-static
```

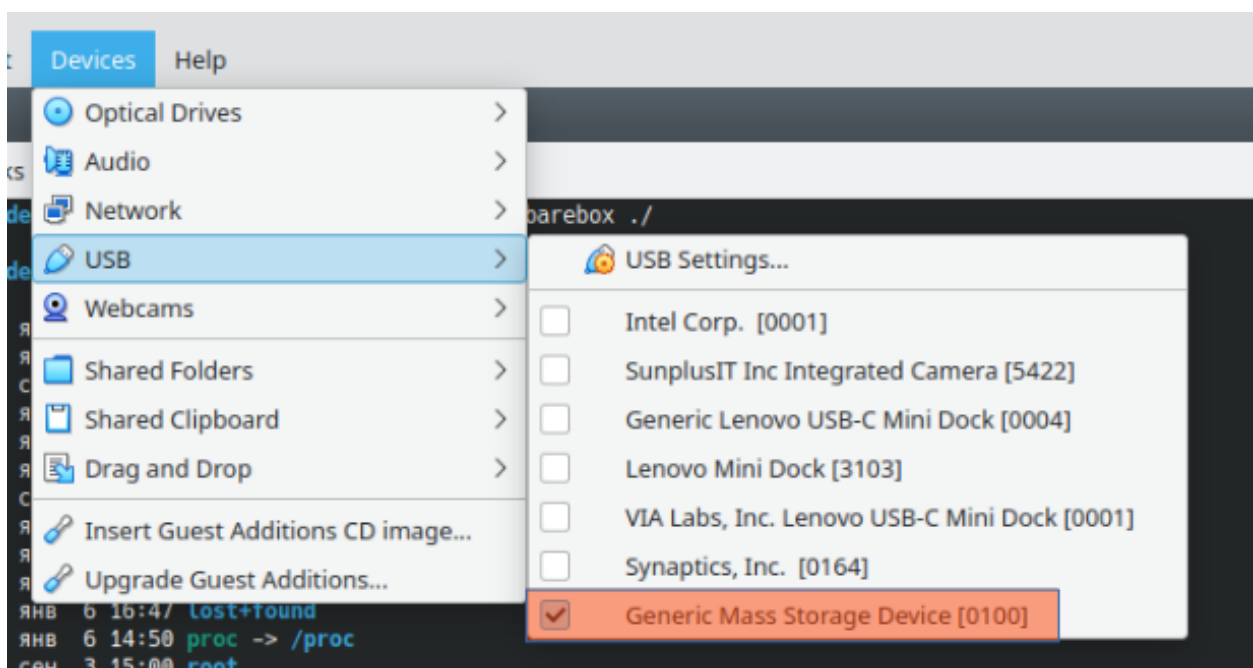
## 1.2.8 Скопируйте папку barebox из каталога support в корень созданной rootfs

```
# sudo cp -r $BAGET/support/barebox/ ./rootfs/
```

В данной папке лежит скомпилированное ядро Linux (vmlinux...), скомпилированное дерево устройств (k5500vk018\_rbm.dtb) и скрипт выполняемый загрузчиком (barebox.sh).

# 1.3 Проверка

## 1.3.1 Вставьте SD карту в ПК, и подключите её к виртуальной машине



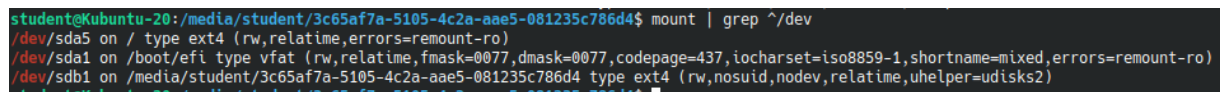
**1.3.2** Необходимо узнать точку монтирования SD карты. Она может отличаться в зависимости от ридера, что Вы используете. Для упрощения ввода, выполните команду в консоле:

---

```
# mount | grep ^/dev
```

---

для получения списка примонтированных устройств, при этом командой `grep` происходит фильтрация лишнего вывода, и отображается только список блочных устройств. Вывод будет примерно как на рисунке



```
student@kubuntu-20:/media/student/3c65af7a-5105-4c2a-aae5-081235c786d4$ mount | grep ^/dev
/dev/sda5 on / type ext4 (rw,relatime,errors=remount-ro)
/dev/sda1 on /boot/efi type vfat (rw,relatime,fmask=0077,dmask=0077,codepage=437,iocharset=iso8859-1,shortname=mixed,errors=remount-ro)
/dev/sdb1 on /media/student/3c65af7a-5105-4c2a-aae5-081235c786d4 type ext4 (rw,nosuid,nodev,relatime,uhelper=udisks2)
```

Первые две строчки, это разделы виртуального жёсткого диска, далее точка монтирования SD карты (`/media/student/...`)

Для упрощения ввода, создадим временную системную переменную, которой назначим путь, для примера(!) на рисунке выше:

---

```
# export BAGET_SD=\
/media/student/3c65af7a-5105-4c2a-aae5-081235c786d4
```

---

Нужно помнить, что эта переменная существует только в том терминале, где был вызван `export`, в остальных окнах терминала она не существует, если Вы закроете текущее окно, или Вам понадобится эта переменная в другом окне, необходимо будет повторить процедуру присвоения.

**1.3.3** Удалите все файлы с sd карты

---

```
# sudo rm -rf $BAGET_SD/*
```

---

скопируйте созданную `rootfs` на sd карту

---

```
# sudo cp -a $BAGET/lab_01/rootfs/* $BAGET_SD/
```

---

и отмонтируйте SD карту

---

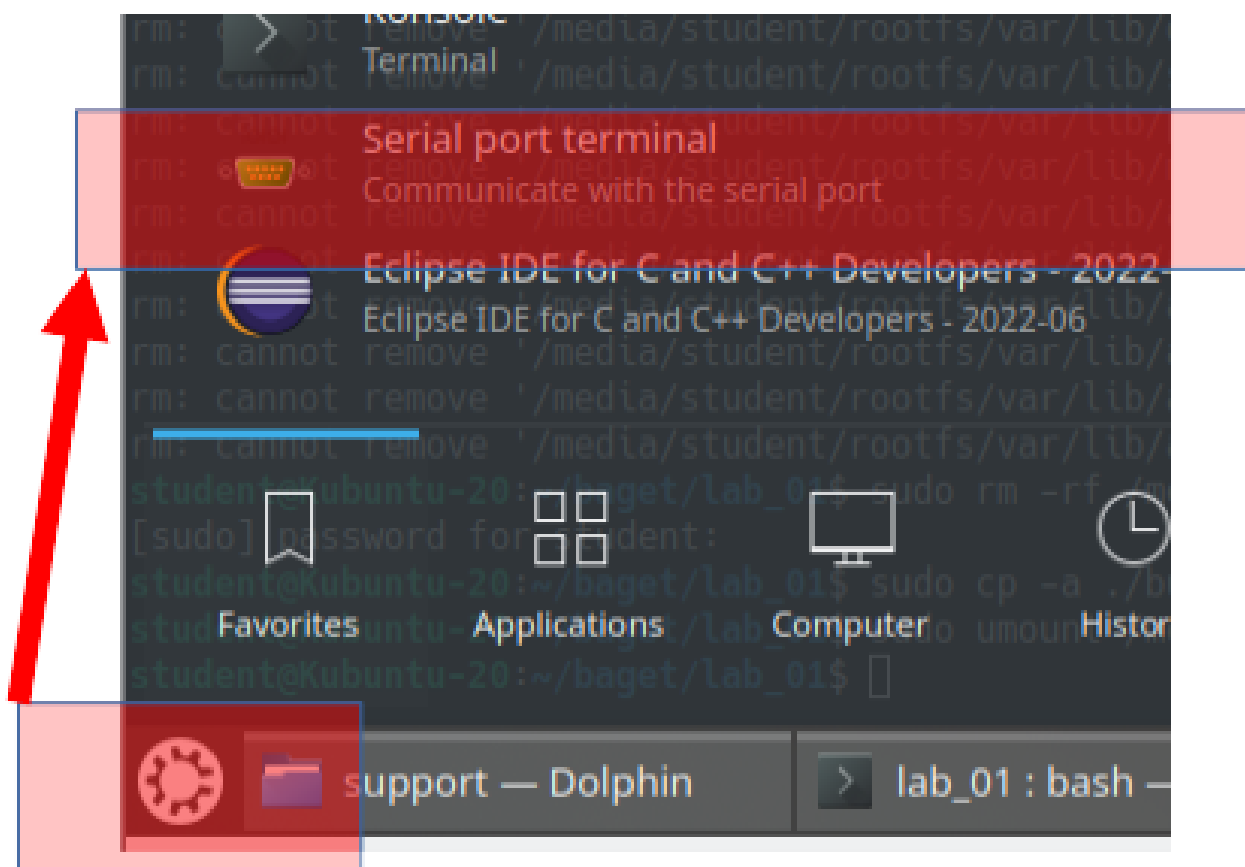
```
# sudo umount $BAGET_SD
```

---

(это важно, так как в ОС Linux запись данных на носители происходит в фоне, и если вы отключите накопитель «неправильно» то нет гарантии, что данные успели записаться на носитель!). Дождитесь когда команда выполниться, после чего достаньте SD карту и вставьте её в плату.

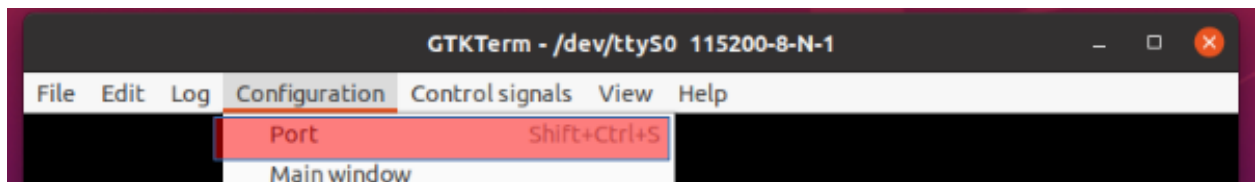
**1.3.4** Подключите USB шнуром плату к ПК. Проверьте, и при необходимости подключите USB устройство FTDI RBM\_C1K5500VK018 к виртуальной машине (меню Device→USB).

**1.3.5** Запустите программу gtkterm или нажмите Ctrl + Shift + T для открытия новой вкладки в окне терминала и выполните команду minicom (это консольная утилита, в виртуальной машине эта утилита уже настроена на нужный порт, с нужными параметрами)



**1.3.6** Выберите порт /dev/ttyUSB1 через меню Configuration->Port.





Нажмите кнопку ОК.

если в окне не появился текст, и на нажатие клавиши Enter так же нет реакции, нажмите на кнопку Reset на плате. Изучите вывод, там будет написано, что пошло не так.

Один из вариантов, когда при записи файлов на SD карту что-то поби-лось. В этом случае, вы увидите много записей как в примере ниже:

```
srisa-sdhci 1b50b000.sdhci@1b50b000.of: registered as mci0
srisa-sdhci 1b50b000.sdhci@1b50b000.of: SDHCI timeout while waiting for
done
srisa-sdhci 1b50b000.sdhci@1b50b000.of: error on command 8
srisa-sdhci 1b50b000.sdhci@1b50b000.of: state = 0400 03ff, interrupt =
8001 0001
srisa-sdhci 1b50b000.sdhci@1b50b000.of: r0 00000000 r1 00000000 r2
00000000 r3 00000000
```

Нужно повторить все шаги этого раздела. **При вынимании SD кар-ты убедитесь, что плата обесточена!**

### 1.3.7 Введите в качестве логина и пароля root

Поздравляем, Вы запустили Debian дистрибутив на плате.

### 1.3.8 Выключите плату, для чего в начале введите команду

---

```
$ poweroff
```

---

дождитесь, как появиться надпись

```
reboot: System halt
```

после чего отключите USB кабель от ПК или платы.