

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САХАЛИНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

**для студентов по выполнению
практических работ
по дисциплине**

ПМ.02 Сопровождение и схемотехническое обслуживание интеллектуальных интегрированных систем

**укрупненная группа:
09.00.00 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА
специальность: 09.02.08 Интеллектуальные
интегрированные системы**

базовый уровень подготовки

**Квалификация: Техник по интеллектуальным интегрированным системам
Форма обучения: очная**

Южно-Сахалинск
2025

УТВЕРЖДАЮ
Зам. директора по НМР
 Е.Н. Ермолаева
«24» марта 2025 г.

Разработчик: Перов И.А., преподаватель информационных дисциплин

Одобрено на заседании ПЦК
информационных дисциплин
Протокол № 8 от «21» марта 2025г.
Председатель ПЦК



О.Б. Куприяшина

СОДЕРЖАНИЕ

1. ОБЩИЕ ПОЛОЖЕНИЯ	2
2. ПРАКТИЧЕСКИЕ РАБОТЫ (РАЗДЕЛ 1. ИНТЕРФЕЙСЫ МИКРОКОНТРОЛЛЕРОВ)	3
Практическая работа №1-2. Передача команд с ПК на микроконтроллер (UART) – 4 часа	3
Практическая работа №3-4. Приём данных с МК на ПК (UART) – 4 часа	3
Практическая работа №5-6. Передача данных UART через Bluetooth (HC-05) – 4 часа	4
Практическая работа №7-8. Управление микроконтроллером со смартфона через Bluetooth – 4 часа	5
Практическая работа №9-10. Взаимодействие с SPI модулем (например, SD-карта или дисплей Nokia 5110) – 4 часа	6
Практическая работа №11-12. Взаимодействие с I2C модулем (например, RTC DS3231 или датчик BME280) – 4 часа	6
Практическая работа №13-14. Взаимодействие с 1-Wire температурным датчиком DS18B20 – 4 часа	7
Практическая работа №15-16. Организация управления по Ethernet (модуль W5500) – 4 часа	8
Практическая работа №17-18. Организация управления по Wi-Fi (ESP8266 как модуль или плата) – 4 часа	9
3. ПРАКТИЧЕСКИЕ РАБОТЫ (РАЗДЕЛ 2. ТЕХНИЧЕСКОЕ СОПРОВОЖДЕНИЕ ИНТЕГРИРОВАННЫХ СИСТЕМ)	10
Практическая работа №1-2. Сборка специализированной компьютерной системы – 4 часа	10
Практическая работа №3-4. Работа с виртуальной машиной. Установка ОС – 4 часа	10
Практическая работа №5. Работа с командной строкой ОС – 2 часа	11
Практическая работа №6-7. Поиск и устранение неполадок оборудования в Astra Linux – 4 часа	11
Практическая работа №8. Создание раздела в Astra Linux – 2 часа	12
Практическая работа №9-10. Создание прямых и перекрестных кабелей UTP – 4 часа	13
Практическая работа №11-12. Проектирование локальной сети – 4 часа	13
Практическая работа №13-14. Организация беспроводной сети – 4 часа	14
4. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К УЧЕБНОЙ ПРАКТИКЕ (36 часов)	15
5. КРИТЕРИИ ОЦЕНКИ ПРАКТИЧЕСКИХ РАБОТ	16
6. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА	17

1. ОБЩИЕ ПОЛОЖЕНИЯ

Цели практических работ:

- Овладеть методами настройки и отладки интерфейсов микроконтроллеров (UART, SPI, I2C, 1-Wire, Ethernet, Wi-Fi).
- Научиться выполнять мониторинг, диагностику и восстановление работоспособности интеллектуальных интегрированных систем.
- Приобрести навыки работы с операционной системой Astra Linux, настройки сетей, диагностики неисправностей.

Оборудование и ПО (общее):

- Для интерфейсов МК: плата Arduino Uno / Nano / STM32, компьютер с Arduino IDE, модули: Bluetooth HC-05/HC-06, SPI (например, SD-карта или дисплей), I2C (RTC DS3231 или датчик BME280), 1-Wire (DS18B20), Ethernet (W5500 или ENC28J60), Wi-Fi (ESP8266).
- Для ПК и сетей: компьютеры с ОС Astra Linux (или VirtualBox), кабели UTP, обжимной инструмент, сетевое тестирование, маршрутизаторы/коммутаторы (или симуляторы Cisco Packet Tracer).

Техника безопасности:

- При работе с электрическими цепями не превышать напряжение 5–12 В.
- Перед сборкой схем отключать питание.
- При обжиге кабеля UTP соблюдать осторожность с инструментом.

2. ПРАКТИЧЕСКИЕ РАБОТЫ (РАЗДЕЛ 1. ИНТЕРФЕЙСЫ МИКРОКОНТРОЛЛЕРОВ)

Практическая работа №1-2. Передача команд с ПК на микроконтроллер (UART) – 4 часа

Цель: научиться отправлять символьные команды из Serial Monitor на плату Arduino и выполнять соответствующие действия (управление светодиодом, чтение данных).

Оборудование: Arduino Uno, светодиод, резистор 220 Ом, компьютер, USB-кабель.

Задание:

1. Подключить светодиод к pin 13 (или внешний через резистор к pin 9).
2. Написать программу, которая читает команды из последовательного порта (Serial.readString() или посимвольно).
3. Реализовать команды:
 - "ON" – включить светодиод;
 - "OFF" – выключить светодиод;
 - "BLINK n" – мигнуть n раз (n – число от 1 до 10);
 - "STATUS" – отправить обратно состояние светодиода.
4. Все команды должны работать без перезагрузки, ввод в верхнем регистре.
5. Добавить эхо: МК повторяет полученную команду.

Порядок выполнения:

1. В Arduino IDE настроить скорость 9600 бод.
2. Использовать Serial.available() для проверки наличия данных.
3. Строку читать через Serial.readStringUntil('\n') или parseInt() для числа.
4. Сравнивать команды и выполнять digitalWrite().
5. Протестировать все команды из монитора порта.

Форма отчёта:

Листинг программы, схема подключения, скриншот монитора порта с вводом команд и ответами.

Контрольные вопросы:

1. Почему скорость передачи должна совпадать на обоих концах?
2. Как отправить число, а не строку?
3. Что такое «дребезг» в контексте последовательной передачи?

Критерии: правильное распознавание команд, работа мигания, эхо.

Практическая работа №3-4. Приём данных с МК на ПК (UART) – 4 часа

Цель: организовать передачу данных от микроконтроллера на компьютер (например, показания аналогового датчика).

Оборудование: Arduino, потенциометр (10 кОм), провода.

Задание:

1. Подключить потенциометр к аналоговому входу A0.
2. Написать программу, которая каждые 200 мс считывает значение АЦП (0–1023) и отправляет в Serial в виде: "ADC = 512, Voltage = 2.50 V".
3. Реализовать команду "START" – начать отправку, "STOP" – прекратить.
4. По команде "ONCE" отправить одно измерение.
5. Выводить данные в монитор порта и одновременно сохранять в файл (копированием).

Порядок выполнения:

- Использовать `analogRead()`.
- Пересчёт напряжения: $\text{voltage} = \text{value} * 5.0 / 1023$.
- Для управления потоком – флаг `bool sendData`.
- В `loop()` проверять наличие команд и флаг.

Форма отчёта:

Скриншоты вывода, листинг, график зависимости (построить в Excel на основе скопированных данных).

Контрольные вопросы:

1. Какова разрядность АЦП Arduino Uno?
2. Как изменить частоту опроса?
3. Почему при быстрой отправке данные могут «накладываться»?

Практическая работа №5-6. Передача данных UART через Bluetooth (HC-05) – 4 часа

Цель: настроить модуль Bluetooth HC-05 в режиме прозрачного канала и передавать данные на смартфон/ПК.

Оборудование: Arduino, Bluetooth HC-05, светодиод, смартфон с Bluetooth Terminal (например, приложение «Bluetooth Terminal HC-05»).

Задание:

1. Подключить HC-05 к Arduino (TX→RX, RX→TX через делитель напряжения для 5V→3.3V, питание 5V, GND).
2. Установить режим «АТ-команды», настроить имя, пароль (1234), скорость 9600.

3. Написать программу, которая принимает символ '1' – включить светодиод, '0' – выключить, и отправляет подтверждение "LED ON"/"LED OFF".
4. Со смартфона подключиться к модулю и протестировать.

Порядок выполнения:

- Для AT-команд: использовать программный Serial на пинах 10,11 (SoftwareSerial).
- После настройки переключить в обычный режим.
- В коде проверять `Bluetooth.available()` и выполнять действия.

Форма отчёта:

Схема подключения (особенно делитель напряжения на RX), листинг, скриншот смартфона с командами.

Контрольные вопросы:

1. Зачем нужен делитель напряжения на линии TX Arduino → RX HC-05?
2. Какие режимы работы у HC-05 (Master/Slave)?
3. Как изменить имя модуля?

Практическая работа №7-8. Управление микроконтроллером со смартфона через Bluetooth – 4 часа

Цель: разработать систему управления несколькими устройствами через Bluetooth с обратной связью.

Оборудование: Arduino, HC-05, 2 светодиода (красный, зелёный), сервопривод (SG90) – опционально.

Задание:

1. Подключить два светодиода к выводам 9 и 10.
2. Реализовать протокол команд:
 - `R1` – включить красный, `R0` – выключить.
 - `G1` – включить зелёный, `G0` – выключить.
 - `S<угол>` – повернуть сервопривод на угол (например, S90).
 - `STATUS` – отправить состояние всех устройств.
3. Программа должна работать в неблокирующем режиме (без `delay` в обработке команд).

Порядок выполнения:

- Парсить строки: если первый символ 'R', второй '0' или '1' и т.д.
- Для серво – выделить число после 'S'.
- Отправлять ответы: `"RED ON"`, `"SERVO 90"`.

Форма отчёта:

Листинг, видео демонстрации (или скриншоты терминала), схема.

Контрольные вопросы:

1. Как обеспечить одновременное управление без конфликта команд?
2. Что такое «конечный автомат» для парсинга команд?
3. Как добавить проверку целостности данных (CRC)?

Практическая работа №9-10. Взаимодействие с SPI модулем (например, SD-карта или дисплей Nokia 5110) – 4 часа

Цель: освоить протокол SPI на примере работы с модулем SD-карты.

Оборудование: Arduino, модуль SD-карты (SPI), карта microSD (2-8 ГБ, отформатирована в FAT16/FAT32).

Задание:

1. Подключить модуль SD: MOSI→11, MISO→12, SCK→13, CS→4 (или другой).
2. Установить библиотеку `SD.h`.
3. Написать программу, которая:
 - инициализирует карту;
 - создаёт файл `data.txt` и записывает в него строку "Hello, SPI!" + время (millis);
 - затем считывает файл и выводит в Serial.
4. Реализовать запись показаний потенциометра каждые 5 секунд в файл `log.csv` (значение, время).
5. Вывести список всех файлов на карте.

Порядок выполнения:

- Подключить CS к pin 4, использовать `SD.begin(4)`.
- Для записи: `File dataFile = SD.open("data.txt", FILE_WRITE);`.
- Для чтения: `while (dataFile.available()) Serial.write(dataFile.read());`.

Форма отчёта:

Схема, листинг, вывод Serial с содержимым файла.

Контрольные вопросы:

1. Какие линии используются в SPI (названия и назначение)?
2. Чем отличается SPI от I2C по топологии?
3. Что такое «режимы SPI» (CPOL, CPHA)?

Практическая работа №11-12. Взаимодействие с I2C модулем (например, RTC DS3231 или датчик BME280) – 4 часа

Цель: освоить протокол I2C на примере чтения реального времени из модуля часов DS3231.

Оборудование: Arduino, модуль DS3231 (I2C), провода.

Задание:

1. Подключить DS3231: SDA→A4, SCL→A5 (для Arduino Uno), питание 5V.
2. Установить библиотеку `RTClib.h`.
3. Написать программу, которая:
 - инициализирует RTC;
 - если часы не работают, устанавливает время из компиляции (**DATE, TIME**);
 - каждую секунду выводит в Serial дату и время.
4. Добавить функцию будильника: при совпадении с заданным временем

зажигать светодиод на 3 секунды.

5. Вывести также температуру со встроенного датчика (если есть в модуле).

Порядок выполнения:

- `Wire.begin()` автоматически в библиотеке.
- `rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));`
- Для будильника – проверить `if (now.hour()==alarmHour &&`

`now.minute()==alarmMinute).`

Форма отчёта:

Схема, листинг, вывод Serial с временем.

Контрольные вопросы:

1. Как адресуются устройства в I2C?
2. Сколько устройств можно подключить к одной шине?
3. Что такое «подтягивающие резисторы» и зачем они нужны?

Практическая работа №13-14. Взаимодействие с 1-Wire температурным датчиком DS18B20 – 4 часа

Цель: изучить однопроводной интерфейс 1-Wire на примере датчика температуры.

Оборудование: Arduino, датчик DS18B20 (в герметичном корпусе или TO-92), резистор 4.7 кОм, провода.

Задание:

1. Подключить DS18B20: VDD→5V, GND→GND, DQ→pin 2 (с подтяжкой 4.7 кОм к 5V).
2. Установить библиотеки `OneWire.h` и `DallasTemperature.h`.
3. Написать программу, которая каждые 2 секунды считывает температуру и выводит в Serial с точностью 0.1°C.

4. Подключить два датчика на одну шину (адреса разные). Выводить температуру каждого.
5. Реализовать оповещение (светодиод) при превышении порога 30°C.

Порядок выполнения:

- `OneWire oneWire(2); DallasTemperature sensors(&oneWire);`
- `sensors.requestTemperatures(); float t = sensors.getTempCByIndex(0);`
- Для нескольких датчиков – цикл по `getDeviceCount()`.

Форма отчёта:

Схема, листинг, вывод температур, адреса датчиков в шестнадцатеричном виде.

Контрольные вопросы:

1. Как датчики идентифицируются на шине 1-Wire?
2. Какая максимальная длина линии связи?
3. Почему нужен резистор подтяжки?

Практическая работа №15-16. Организация управления по Ethernet (модуль W5500) – 4 часа

Цель: создать простой веб-сервер на Arduino с Ethernet-модулем для управления светодиодом через браузер.

Оборудование: Arduino, модуль Ethernet W5500 (или ENC28J60), светодиод.

Задание:

1. Подключить Ethernet модуль (используя библиотеку Ethernet.h).
2. Назначить статический IP (например, 192.168.1.100).
3. Создать веб-страницу с кнопками «ON» и «OFF».
4. При нажатии кнопки отправлять HTTP GET-запрос (например, `/?led=on`),

МК изменяет состояние светодиода и возвращает обновлённую страницу.

5. Добавить отображение текущего состояния светодиода на странице.

Порядок выполнения:

- Инициализация: `Ethernet.begin(mac, ip); server.begin();`
- В `loop()`: `EthernetClient client = server.available();` разбор строки GET.
- Сформировать HTML-ответ с заголовками.

Форма отчёта:

Схема, листинг, скриншот веб-страницы в браузере.

Контрольные вопросы:

1. Какой протокол используется для передачи веб-страниц?
2. Чем отличается TCP от UDP?
3. Как сделать динамическое обновление без перезагрузки страницы (AJAX)?

Практическая работа №17-18. Организация управления по Wi-Fi (ESP8266 как модуль или плата) – 4 часа

Цель: настроить подключение к Wi-Fi сети и управление через веб-интерфейс с помощью модуля ESP-01 или платы NodeMCU.

Оборудование: ESP8266 (NodeMCU или ESP-01 с переходником), светодиод.

Задание:

1. Настроить ESP8266 в режиме станции (STA) для подключения к домашней сети Wi-Fi.
2. Создать веб-сервер на порту 80.
3. Реализовать страницу с двумя кнопками и отображением статуса светодиода.
4. Добавить возможность ввода SSID и пароля через веб-интерфейс (AP mode для начальной настройки – опционально).
5. Управление светодиодом на GPIO2 (встроенный на NodeMCU) или внешнем.

Порядок выполнения:

- Использовать библиотеку `ESP8266WiFi.h` и `ESP8266WebServer.h`.
- В `setup()`: `WiFi.begin(ssid, password)`; сервер обрабатывает `on("/", handleRoot)`; `on("/on", handleOn)`;
- В обработчиках менять `digitalWrite`.

Форма отчёта:

Схема, листинг, скриншот веб-страницы с IP-адресом.

Контрольные вопросы:

1. Какие режимы работы Wi-Fi поддерживает ESP8266?
2. Как защитить веб-интерфейс паролем?
3. Что такое MQTT и как его использовать вместо HTTP?

3. ПРАКТИЧЕСКИЕ РАБОТЫ (РАЗДЕЛ 2. ТЕХНИЧЕСКОЕ СОПРОВОЖДЕНИЕ ИНТЕГРИРОВАННЫХ СИСТЕМ)

Практическая работа №1-2. Сборка специализированной компьютерной системы – 4 часа

Цель: научиться подбирать компоненты, собирать системный блок и устанавливать базовое ПО.

Оборудование: набор комплектующих (материнская плата, процессор, ОЗУ, SSD, блок питания, корпус), отвёртки, термопаста.

Задание:

1. Выбрать компоненты для компьютера по заданию (например, для офисного ПК или для системы видеонаблюдения).
2. Собрать системный блок: установить CPU, кулер, модули ОЗУ, накопитель, подключить питание, переднюю панель.
3. Подключить монитор, клавиатуру, мышь.
4. Включить и проверить POST (вход в BIOS).
5. Установить операционную систему (Astra Linux или Windows) с загрузочной флешки.

Форма отчёта:

Фото собранной системы, список компонентов с обоснованием выбора, отчёт о POST и установке ОС.

Контрольные вопросы:

1. Как совместимость сокета процессора и материнской платы?
2. Зачем нужна термопаста?
3. Какие кабели питания используются для современных SSD?

Практическая работа №3-4. Работа с виртуальной машиной. Установка ОС – 4 часа

Цель: освоить создание виртуальных машин в VirtualBox и установку Astra Linux.

Оборудование: ПК с VirtualBox, образ Astra Linux (Common Edition).

Задание:

1. Создать виртуальную машину: 2 ГБ ОЗУ, 20 ГБ VDI-диск, сеть NAT.
2. Загрузить ISO-образ, выполнить установку Astra Linux (разметка диска автоматически, создать пользователя).
3. После установки установить гостевые дополнения (Guest Additions) для удобства.
4. Настроить сетевое соединение (мост или NAT).

5. Установить пакет `mc` (Midnight Commander) через терминал.

Форма отчёта:

Скриншоты процесса установки, настройки сети, работающего рабочего стола.

Контрольные вопросы:

1. Чем отличается виртуализация от эмуляции?
2. Какой тип сетевого адаптера в VirtualBox обеспечивает доступ к интернету из гостевой ОС?
3. Как сделать общую папку между хостовой и гостевой ОС?

Практическая работа №5. Работа с командной строкой ОС – 2 часа

Цель: освоить основные команды Linux (Astra) для навигации, управления файлами и процессами.

Оборудование: виртуальная машина с Astra Linux (из предыдущей работы).

Задание:

Выполнить последовательность команд, зафиксировав результат (можно скриншотами или текстом вывода):

1. Вывести текущий каталог (`pwd`), список файлов (`ls -la`).
2. Создать структуру каталогов: `~/lab/ {dir1, dir2}`, в `dir1` создать файл `test.txt` с текстом «Hello».
3. Скопировать файл в `dir2`, переименовать в `copy.txt`.
4. Найти все файлы с расширением `.txt` в домашней директории (`find ~ -name "*.txt"`).
5. Вывести содержимое файла `copy.txt` (`cat`).
6. Изменить права доступа на `test.txt` (только чтение для владельца) – `chmod 400`.
7. Запустить процесс `sleep 300` в фоне, затем найти его `pid` и убить (`kill`).

Форма отчёта:

Листинг выполненных команд с краткими пояснениями и скриншотами.

Контрольные вопросы:

1. Что означают права `755`?
2. Как перенаправить вывод команды в файл (`>` и `>>`)?
3. Чем отличается `kill` от `kill -9`?

Практическая работа №6-7. Поиск и устранение неполадок оборудования в Astra Linux – 4 часа

Цель: научиться диагностировать проблемы с оборудованием (видеокарта, сеть, диски) с помощью системных утилит.

Оборудование: виртуальная машина (можно симулировать отключение устройств).

Задание:

1. Вывести список PCI-устройств (`lspci`), USB-устройств (`lsusb`).
2. Проверить загрузку процессора и памяти (`top`, `htop`, `free -h`).
3. Проверить S.M.A.R.T. диска (если есть доступ – `smartctl`).
4. Сымитировать ошибку: отключить сетевой интерфейс (`sudo ifconfig eth0 down`), затем диагностировать (`ip a`, `ping`) и восстановить (`sudo ifconfig eth0 up`).
5. Посмотреть логи системы (`dmesg | tail -20`, `journalctl -xe`).
6. Создать задачу: определить, почему не работает принтер (проверить статус службы CUPS).

Форма отчёта:

Отчёт по каждому шагу с командами и выводами. Описание методики поиска неисправности.

Контрольные вопросы:

1. Как посмотреть использование дискового пространства (`df -h`, `du -sh`)?
2. Как найти процесс, занимающий много памяти (`ps aux --sort=-%mem | head`)?
3. Где хранятся логи ядра в Linux?

Практическая работа №8. Создание раздела в Astra Linux – 2 часа

Цель: научиться создавать и форматировать разделы жёсткого диска с помощью утилит `fdisk`, `mkfs`.

Оборудование: виртуальная машина с добавленным дополнительным виртуальным диском (например, 5 ГБ).

Задание:

1. Добавить к виртуальной машине второй диск (через VirtualBox).
2. Запустить `sudo fdisk -l` – определить новое устройство (например, `/dev/sdb`).
3. Создать на нём два раздела:
 - первый размером 2 ГБ, тип Linux (83);
 - второй размером 3 ГБ, также Linux.
4. Отформатировать первый раздел в ext4 (`sudo mkfs.ext4 /dev/sdb1`), второй – в vfat (FAT32) (`sudo mkfs.vfat /dev/sdb2`).
5. Создать точки монтирования `/mnt/data1` и `/mnt/data2`, смонтировать разделы.
6. Добавить автоматическое монтирование при загрузке (записать в `/etc/fstab`).

Форма отчёта:

Скриншоты процесса, вывод `df -h`, содержимое `/etc/fstab`.

Контрольные вопросы:

1. Какие типы разделов бывают (primary, extended, logical)?
2. Что такое таблица разделов GPT и MBR?
3. Как изменить размер раздела без потери данных?

Практическая работа №9-10. Создание прямых и перекрестных кабелей UTP – 4 часа

Цель: научиться обжимать витую пару по стандартам TIA/EIA-568A и 568B для прямого и кроссового кабеля.

Оборудование: кабель UTP cat5e, коннекторы RJ-45, обжимной инструмент, тестер кабеля (опционально).

Задание:

1. Изучить цветовую схему прямого кабеля (оба конца 568B):
порядок: бело-оранжевый, оранжевый, бело-зелёный, синий, бело-синий, зелёный, бело-коричневый, коричневый.
2. Обжать один прямой кабель длиной 1-2 м.
3. Обжать перекрёстный кабель (один конец 568A, другой 568B). Схема 568A: бело-зелёный, зелёный, бело-оранжевый, синий, бело-синий, оранжевый, бело-коричневый, коричневый.
4. Проверить кабели тестером (прогнать все 8 жил).
5. Подключить два компьютера через кросс-кабель (без свитча) и проверить связь (ping).

Форма отчёта:

Фотографии процесса обжима, схема распиновки, результат пинга.

Контрольные вопросы:

1. Для чего нужен перекрёстный кабель?
2. Какой стандарт используется в современных сетях (прямой)?
3. Какая максимальная длина сегмента UTP?

Практическая работа №11-12. Проектирование локальной сети – 4 часа

Цель: научиться планировать IP-адресацию, топологию и настраивать маршрутизатор/коммутатор (можно в симуляторе Cisco Packet Tracer).

Оборудование: Packet Tracer или реальное оборудование (коммутатор, маршрутизатор, ПК).

Задание:

1. Разработать сеть для офиса из 3 отделов (по 5-10 компьютеров). Адресное пространство 192.168.1.0/24. Разделить на подсети (например, /28 для каждого отдела).

2. Нарисовать топологию в Packet Tracer: коммутаторы, маршрутизатор, ПК.
3. Назначить IP-адреса, маски, шлюзы.
4. Настроить маршрутизатор (статическая маршрутизация или RIP).
5. Проверить связность `ping` между разными подсетями.

Форма отчёта:

Схема сети, таблица адресации, конфигурационные файлы (или скриншоты настроек).

Контрольные вопросы:

1. Чем отличается коммутатор от маршрутизатора?
2. Что такое VLAN и зачем он нужен?
3. Как рассчитать количество хостов в подсети по маске?

Практическая работа №13-14. Организация беспроводной сети – 4 часа

Цель: настроить Wi-Fi точку доступа и подключить клиентов, обеспечить безопасность.

Оборудование: Wi-Fi роутер (или компьютер с Wi-Fi-адаптером в режиме AP), ноутбук/смартфон.

Задание:

1. Настроить роутер (или использовать Linux с `hostapd`):
 - SSID: `Lab_WiFi`
 - Режим: 802.11n, канал 6
 - Шифрование: WPA2-PSK, пароль `Lab2025$`
2. Включить DHCP-сервер на роутере (пул 192.168.10.10-50).
3. Подключить два клиента (ноутбук, телефон), получить IP-адреса.
4. Проверить доступ в интернет (если есть) и локальную связь.
5. Включить фильтрацию по MAC-адресу (разрешить только два известных устройства).

Форма отчёта:

Скриншоты настроек роутера, вывод `iwconfig` на клиенте, результат пинга.

Контрольные вопросы:

1. Какие стандарты Wi-Fi поддерживают скорость 300 Мбит/с?
2. Чем WPA3 отличается от WPA2?
3. Как скрыть SSID и стоит ли это делать?

4. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К УЧЕБНОЙ ПРАКТИКЕ (36 часов)

Учебная практика включает следующие виды работ (каждый – 3–4 часа). Для каждого вида даны краткие задания.

№	Вид работы	Задание / результат
1	Создание загрузочного носителя	Записать ISO образ Astra Linux на USB (через Rufus или dd). Сделать загрузочную флешку.
2	Создание образа ОС	С помощью dd или Clonezilla создать образ установленной системы на внешний диск.
3	Установка принтера в Astra Linux	Подключить принтер (виртуальный или реальный), установить драйвер через CUPS, распечатать тестовую страницу.
4	Совместное использование принтера в Astra Linux	Настроить принтер как сетевой (Samba или IPP), подключить с другого компьютера.
5	Управление системными файлами в Astra Linux	Использовать mc, nano, изменить конфигурационные файлы (/etc/hosts, /etc/network/interfaces).
6	Контроль и управление системными ресурсами Astra Linux	Мониторинг CPU, памяти, диска (top, vmstat, iostat). Создать задачу cron для резервного копирования.
7	Настройка брандмауэра в Astra Linux	С помощью ufw разрешить SSH (порт 22), запретить все остальные входящие. Проверить.
8	Создание простой сети	Соединить два ПК прямым кабелем, назначить IP (192.168.1.1/24 и 192.168.1.2/24), проверить ping.
9	Настройка динамической адресации в сети	Поднять DHCP-сервер (isc-dhcp-server) на одном ПК, клиент получает адрес автоматически.
10	Настройка статической адресации в сети	Прописать статические IP на интерфейсах, настроить маршрут по умолчанию.
11	Настройка комплексной сети	Собрать сеть из двух подсетей с маршрутизатором (Linux как роутер), настроить NAT.

Форма отчёта по практике:

Дневник практики с кратким описанием каждого выполненного вида работ, скриншотами, выводами. В конце – характеристика от руководителя.

5. КРИТЕРИИ ОЦЕНКИ ПРАКТИЧЕСКИХ РАБОТ

Оценка	Требования
5 (отлично)	Работа выполнена полностью, в срок. Схема собрана правильно, программа загружена и функционирует без ошибок. Отчёт оформлен аккуратно, содержит все разделы, скриншоты, листинги. Студент свободно отвечает на контрольные вопросы.
4 (хорошо)	Работа выполнена, но есть 1-2 недочёта (например, неверно обжат один кабель, или в коде мелкая ошибка). На вопросы отвечает с небольшими затруднениями.
3 (удовлетворительно)	Работа выполнена частично (не менее 60%), основные функции работают, но есть существенные ошибки. Отчёт неполный.
2 (неудовлетворительно)	Работа не выполнена или не работает основная функция. Отсутствует отчёт. Студент не может объяснить свои действия.

6. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Сажнев А.М. Микропроцессорные системы: цифровые устройства и микропроцессоры. – М.: Юрайт, 2021.
2. Назаров А.В. и др. Эксплуатация объектов сетевой инфраструктуры. – М.: Академия, 2021.
3. Документация Arduino (arduino.cc/reference).
4. Официальное руководство по Astra Linux (astralinux.ru/support).