

КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ
ИМЕНИ АЛЬ-ФАРАБИ

ЛЕКЦИЯ № 2

Особенности Raspberry Pi OS и основные команды управления файловой системой мини-компьютера Raspberry Pi в технологиях Интернета вещей

Бельгибаев Бауржан Абдрахимович
Д.т.н., доцент · Кафедра «Искусственный интеллект и big data»
«Технологии в интеллектуальных системах управления в IoT» · 4 курс, бакалавриат

1 Цель лекции

Сформировать системное представление об архитектуре Raspberry Pi OS, организации файловой системы Linux, настройке сетевых интерфейсов и безопасном использовании командной строки при разработке интеллектуальных IoT-устройств.

2 План лекции

1. Роль Raspberry Pi в архитектуре IoT и интеллектуального управления
2. Архитектура Raspberry Pi OS, ядро Linux и службы
3. Файловая система Linux: каталоги, пути, права доступа
4. Командная строка и операции с файлами и процессами
5. Сетевые команды и удалённое администрирование Raspberry Pi
6. Подключение сенсоров и программных компонентов IoT
7. Надёжность, безопасность и диагностирование
8. Практический пример развёртывания IoT-узла

1 Raspberry Pi как вычислительный узел IoT

Интернет вещей (IoT) объединяет физические объекты, сенсоры, исполнительные устройства, вычислительные узлы и коммуникационные сервисы. В интеллектуальной системе управления Raspberry Pi может выступать пограничным вычислителем (edge node): собирать данные, выполнять предобработку, запускать модели машинного обучения и передавать команды исполнительным механизмам. В отличие от микроконтроллера, одноплатный компьютер работает под полноценной многозадачной ОС и поддерживает файловые системы, сетевой стек, языки высокого уровня и службы удалённого доступа.

Типовая цепочка обработки: датчик → интерфейс GPIO/I²C/SPI/UART → программа сбора данных → фильтрация/оценивание состояния → алгоритм принятия решения → управляющее воздействие → исполнительное устройство. При подключении облака добавляются MQTT/HTTP, брокер сообщений, база временных рядов и панель мониторинга.

Raspberry Pi удобен для прототипирования, однако стандартная Linux-система не гарантирует жёсткого реального времени. Для критичных по задержке контуров управления рекомендуется выносить быстрый цикл на микроконтроллер или ПЛК, оставляя Raspberry Pi функции аналитики, связи и координации.

2 Архитектура Raspberry Pi OS

Raspberry Pi OS — дистрибутив Linux на базе Debian, адаптированный для семейства Raspberry Pi. Его компоненты включают загрузочную цепочку, ядро, драйверы устройств, системные службы, библиотеки и пользовательские приложения. Конкретные инструменты управления сетью и интерфейсами зависят от версии системы; перед выполнением инструкций необходимо проверить версию ОС и модель платы.

Команды диагностики: uname -a — информация о ядре; cat /etc/os-release — выпуск дистрибутива; hostnamectl — сведения о хосте; free -h — оперативная память; df -h — заполнение файловых систем; lsblk — блочные устройства; uptime — время работы и нагрузка.

Службы обычно контролируются systemd. Команды systemctl status <служба>, systemctl start <служба>, systemctl enable <служба> позволяют проверять, запускать и включать автозапуск. Для просмотра журналов используется journalctl -u <служба> -n 50.

4 Уровни Raspberry Pi OS

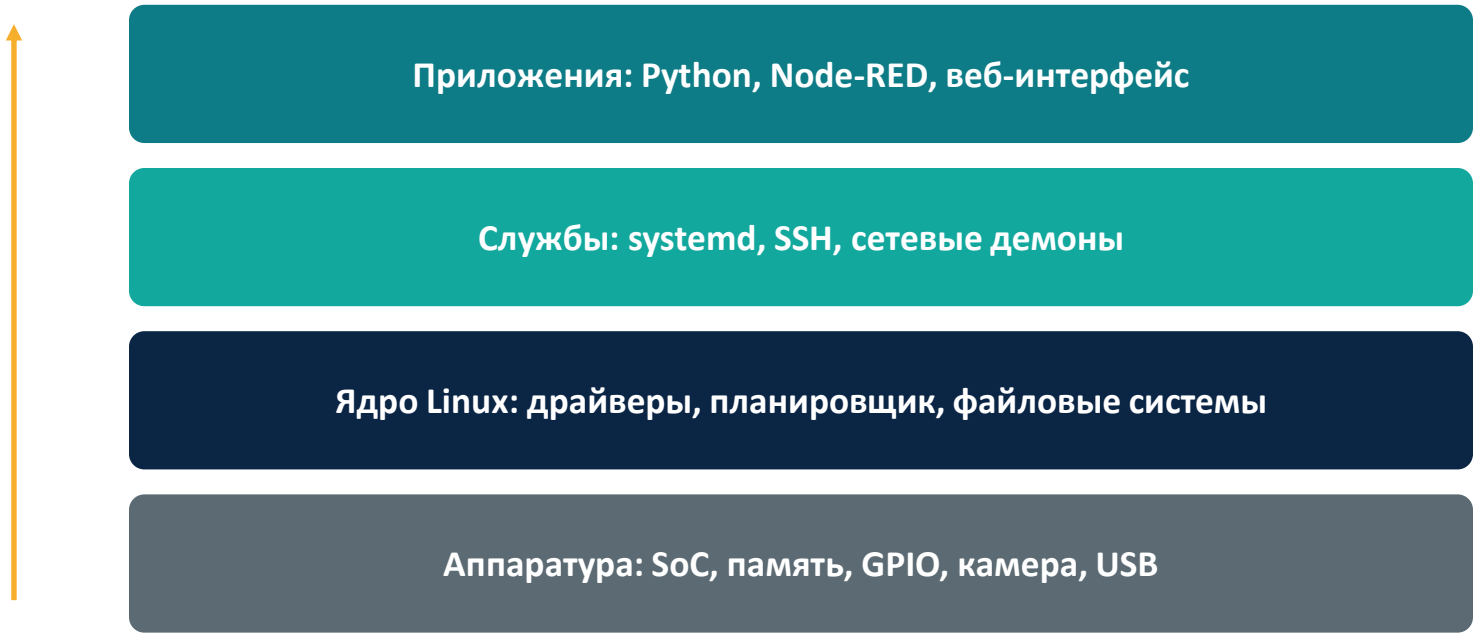

Рис. 2. Слоистая модель: прикладной код не обращается к оборудованию напрямую.

3 Структура файловой системы

В Linux единое дерево каталогов начинается с корня /. Каталог /home содержит пользовательские данные; /etc — конфигурацию; /var/log — журналы; /usr — пользовательские программы и библиотеки; /tmp — временные файлы; /dev — интерфейсы устройств; /proc и /sys — виртуальные представления состояния ядра и оборудования; /boot/firmware (в актуальных конфигурациях) — файлы загрузки.

Абсолютный путь начинается с /, например /home/pi/project/data.csv. Относительный путь вычисляется от текущего каталога. Обозначения . и .. указывают текущий и родительский каталоги; ~ — домашний каталог пользователя. Регистр символов имеет значение: Sensor.py и sensor.py — разные имена.

Права доступа

Три группы прав: владелец, группа, остальные. Буквы r, w, x означают чтение, запись и выполнение (для каталогов x — право прохода). Например, chmod 750 script.sh задаёт rwx владельцу, r-x группе и отсутствие доступа остальным. Команда chown изменяет владельца. Не следует применять chmod 777 к проектам или запускать все программы через sudo.

5 Структура каталогов Linux

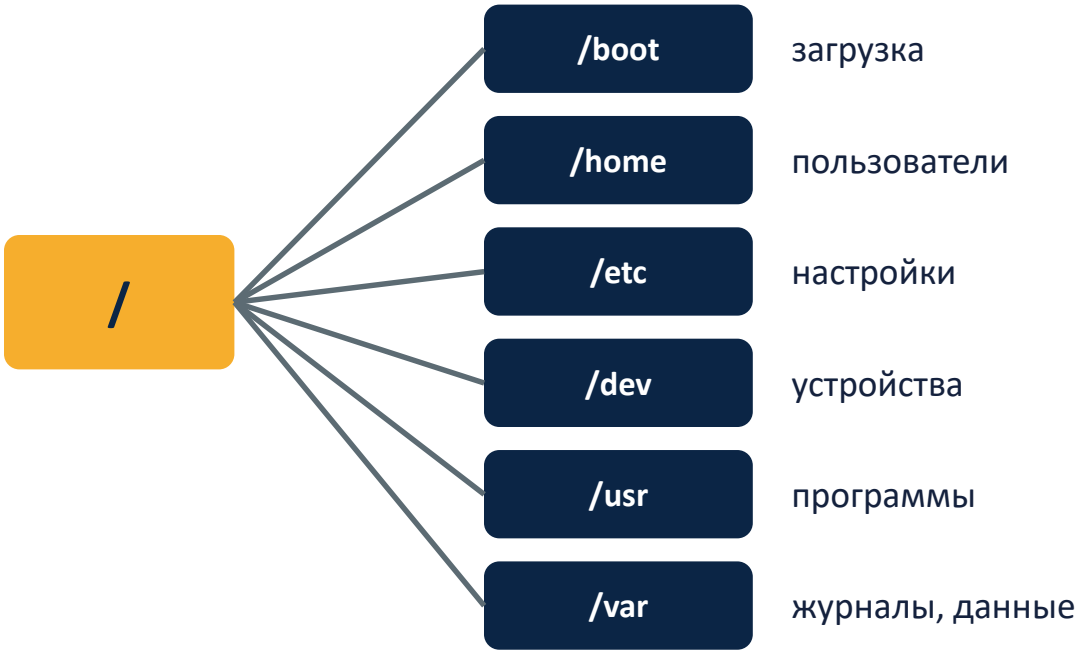

Рис. 3. Единое дерево каталогов; устройства также представлены файлами в /dev.

4 Командная строка: базовые операции

Команда	Назначение
pwd	Показать текущий каталог
ls -lah	Список файлов с размерами и скрытыми элементами
cd ~/iot_project	Перейти в каталог проекта
mkdir -p data logs	Создать подкаталоги
touch data/readings.csv	Создать пустой файл
cp source.py backup.py	Скопировать файл
mv old.py new.py	Переместить/переименовать
rm -i file.txt	Удалить файл с подтверждением
cat /etc/os-release	Показать содержимое файла

4 Командная строка: базовые операции (продолжение)

Команда	Назначение
less logs/app.log	Просмотреть длинный файл
grep -n "ERROR" logs/app.log	Найти ошибки по шаблону
find . -name "*.py"	Найти Python-файлы
ps aux	Посмотреть процессы
top	Мониторинг нагрузки
kill <PID>	Послать процессу сигнал завершения

4 Командная строка: базовые операции (продолжение)

Конвейер | передаёт стандартный вывод одной команды на вход следующей, например journalctl -u iot.service | grep ERROR. Перенаправление > перезаписывает файл, >> дописывает. Перед опасными операциями удаления или изменения прав необходимо проверять текущий каталог и целевой путь.

5 Сетевые средства Raspberry Pi OS

Сетевой узел должен иметь корректную IP-конфигурацию, DNS, маршрут по умолчанию и доступ к нужным портам. Команда ip addr показывает интерфейсы и адреса, ip route — таблицу маршрутизации, ping -c 4 1.1.1.1 — базовую достижимость IP, getent hosts example.com — разрешение имени, ss -tuln — прослушиваемые TCP/UDP-порты. Команда hostname -I может быстро показать назначенные адреса.

Для защищённого удалённого управления применяется SSH. На устройстве включают SSH через предусмотренный для установленной версии механизм настройки, затем подключаются с доверенного компьютера: ssh user@192.168.1.50. Рекомендуются ключевая аутентификация, обновления безопасности, отказ от стандартных паролей и ограничение доступа на сетевом уровне.

Для IoT-телеметрии распространён протокол MQTT с моделью публикации/подписки: датчик публикует данные в топик, брокер маршрутизирует сообщения, а клиент управления их получает. Например, topic factory/line1/temperature может содержать JSON-сообщение с временем, идентификатором датчика и температурой. В производственных системах необходимо использовать TLS, аутентификацию и контроль прав на топики.

6 Сенсоры, интерфейсы и программная среда

GPIO — цифровые линии ввода/вывода, I²C — двухпроводная шина адресуемых устройств, SPI — синхронный последовательный интерфейс, UART — асинхронный последовательный канал. Для аналоговых датчиков, не имеющих цифрового выхода, требуется внешний АЦП. Электрические уровни GPIO Raspberry Pi обычно составляют 3,3 В: прямое подключение сигналов 5 В может повредить плату.

Приложения на Python целесообразно размещать в изолированном виртуальном окружении: python3 -m venv .venv; source .venv/bin/activate; python -m pip install <package>. Зависимости фиксируют в requirements.txt. Работа с аппаратными линиями выполняется библиотеками, совместимыми с моделью платы и текущей ОС.

7 Безопасность и устойчивость IoT-узла

Типичные угрозы: слабые пароли, открытые сервисы, неподписанные обновления, перехват телеметрии, отказ питания, износ накопителя и перегрев. Меры: регулярные обновления, отдельная учётная запись службы, минимальные привилегии, TLS/VPN, резервное копирование конфигурации, журналирование, watchdog и контроль свободного места. Для управления физическим объектом предусматривают безопасное состояние при потере сети и независимую аппаратную защиту.

Наблюдаемость узла включает загрузку CPU, использование памяти, температуру, сетевые задержки, частоту потерь сообщений и временные метки. Если дискретизация производится с периодом T_s, частота измерений равна $f_s = 1/T_s$. Например, при $T_s = 0{,}2$ с частота составляет 5 Гц. Фактические интервалы следует измерять, поскольку Linux-планировщик и сеть вносят джиттер.

8 Практический пример: IoT-мониторинг температуры

Задача: создать прототип для регистрации температуры в лаборатории с отправкой показаний на сервер и аварийным предупреждением при превышении порога. Состав: Raspberry Pi, цифровой температурный датчик, питание, локальная сеть, Python-приложение и MQTT-брокер. В учебном варианте датчик может быть заменён программным генератором измерений.

Алгоритм

1. Проверить версию Raspberry Pi OS и состояние сетевого интерфейса.
2. Создать ~/iot_project, каталоги data и logs, виртуальное окружение.
3. Подключить датчик с соблюдением уровней напряжения и проверить драйвер.
4. Периодически считывать значение, проверять диапазон допустимых измерений.
5. Присваивать временную метку, сохранять запись локально, публиковать в MQTT.
6. При превышении порога формировать предупреждение; исполнительное управление выполнять только через безопасный проверенный контур.

Практический пример: IoT-мониторинг температуры (продолжение)

7. Проверить работу при отключении сети и перезапуске устройства.

Математическое описание

Пусть $y_k = T_k + v_k$ — измеренная температура, где T_k — истинное значение, v_k — шум измерения. Экспоненциальное сглаживание задаётся формулой: $\hat{T}_k = \alpha\, y_k + (1 - \alpha)\, \hat{T}_{(k-1)}$, $0 < \alpha \le 1$. При $\alpha = 0{,}2$ и предыдущей оценке 24,0 °C и новом измерении 26,0 °C получаем $\hat{T}_k = 24{,}4$ °C. Тревога активируется, если $\hat{T}_k > T_{max}$ в течение N последовательных отсчётов; это снижает вероятность ложных срабатываний.

Показатели качества: доля успешно полученных пакетов $P = N_{received} / N_{sent} \cdot 100\%$; средняя задержка $\bar{L} = (1/n)\, \Sigma(t_{receive,i} - t_{send,i})$; средняя абсолютная ошибка измерения $MAE = (1/n)\, \Sigma|\hat{T}_i - T_{ref,i}|$. При оценке точности необходим эталонный измеритель и синхронизация временных меток.

9 Вопросы для исследовательского обсуждения

На уровне докторантуры важно не только настроить систему, но и доказать её пригодность. Следует определить ограничения по времени реакции, допустимую вероятность отказа, требования к воспроизводимости, методику сравнения edge- и cloud-обработки, а также способы количественной оценки надёжности и безопасности. В исследовательском отчёте фиксируются модель платы, выпуск ОС, версия библиотек, схема соединений, параметры сети, набор тестовых сценариев и исходные данные.

12 Заключение

Raspberry Pi OS обеспечивает универсальную программную среду для создания интеллектуальных IoT-узлов: от работы с файловой системой и сетями до запуска алгоритмов компьютерного зрения. Надёжное применение в системах управления требует грамотного разделения задач между Linux-компьютером и детерминированными контроллерами, контроля прав доступа, мониторинга и экспериментальной проверки характеристик.

13 Контрольные вопросы

1. Какую роль Raspberry Pi выполняет в многоуровневой архитектуре IoT?
2. Чем Linux-компьютер отличается от микроконтроллера в контуре управления?
3. Какие компоненты входят в Raspberry Pi OS?
4. Чем абсолютный путь отличается от относительного?
5. Каково назначение каталогов /etc, /var/log, /dev и /home?
6. Как интерпретировать права доступа rwxr-x---?
7. Как безопасно изменить владельца и права файла?
8. Какие команды используются для диагностики процессов и памяти?
9. Как проверить IP-адрес, маршрут и доступность узла?
10. В чём назначение SSH и какие меры его защиты необходимы?
11. Как работает модель публикации/подписки MQTT?

13 Контрольные вопросы (продолжение)

12. Какие особенности электрического подключения GPIO необходимо учитывать?
13. Почему стандартная Raspberry Pi OS не гарантирует жёсткого реального времени?
14. Как измерить задержку и потери пакетов IoT-системы?
15. Какие эксперименты необходимы для верификации прототипа мониторинга?

Рекомендуемая литература и документация

1. Перри Ли. Архитектура интернета вещей. М.: ДМК Пресс, 2018. (Источник из силлабуса.)

2. Матюшин А. О. Программирование микроконтроллеров: стратегия и тактика. М.: ДМК, 2016. (Источник из силлабуса.)

3. Raspberry Pi Documentation. https://www.raspberrypi.com/documentation/

4. Debian Administrator’s Handbook. https://debian-handbook.info/

5. Python Documentation. https://docs.python.org/3/

6. MQTT Version 5.0. OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/