ЛЕКЦИЯ № 8 · Проект MATLAB IoT

# Функционал мини-ПК Raspberry Pi как управляющего IoT-устройства. Взаимодействие ThingSpeak с MATLAB на базе Raspberry Pi




**Бельгибаев Бауржан Абдрахимович**
Д.т.н., доцент
Кафедра «Искусственный интеллект и big data»
ОП 6B07113 «Интеллектуальные системы управления»

# Цель лекции

Изучить применение Raspberry Pi как вычислительного узла IoT-системы, организацию локального управления и сбора данных, архитектуру ThingSpeak, способы передачи измерений в облачный канал и методы MATLAB-аналитики. Научиться проектировать безопасный учебный прототип с корректной обработкой временных меток, ограничений сервиса и ошибок связи.

# План лекции

1. Роль Raspberry Pi в системах IoT и отличие от микроконтроллера
2. Аппаратные интерфейсы, исполнительные устройства и безопасность
3. Архитектура локального управления и облачной аналитики
4. Обзор ThingSpeak: каналы, поля, записи и ключи доступа
5. Подготовка Raspberry Pi и формирование телеметрии
6. Передача данных в ThingSpeak через HTTP API
7. Чтение данных ThingSpeak средствами MATLAB
8. MATLAB-аналитика: фильтрация, визуализация и пороги
9. Временная синхронизация, ограничения частоты и буферизация
10. Обратная связь, команды и границы облачного управления
11. Информационная безопасность и диагностика отказов
12. Сквозной проект мониторинга микроклимата

1

# Raspberry Pi в системах IoT: вычислительный узел, а не обычный микроконтроллер

Raspberry Pi представляет собой одноплатный компьютер, обычно работающий под управлением операционной системы Linux. В отличие от типичного микроконтроллера, он поддерживает полноценную файловую систему, многозадачность, сетевые службы и программные среды высокого уровня. Поэтому корректнее называть Raspberry Pi мини-ПК или одноплатным компьютером, даже если в учебной теме он рассматривается как управляющий IoT-узел.

Микроконтроллеры, такие как устройства семейства Arduino, часто применяются для простых задач ввода-вывода и локального управления с относительно предсказуемым временем исполнения. Raspberry Pi удобен для интеграции нескольких протоколов, локального хранения данных, обработки изображений и сетевого обмена, но обычная операционная система не гарантирует жёсткого реального времени.

В типовой IoT-архитектуре Raspberry Pi выполняет сбор показаний датчиков, предварительную проверку и фильтрацию, запись в журнал и отправку в облако. При необходимости он также управляет исполнительными устройствами через драйверы. Облачный сервис хранит историю, отображает графики и запускает аналитические вычисления.

# 1 Raspberry Pi в системах IoT: вычислительный узел, а не обычный микроконтроллер (продолжение)

**Разделение задач важно для надёжности:** действия, критичные для безопасности и времени реакции, следует реализовывать локально, независимо от доступа к интернету. Облачные расчёты полезны для наблюдения, диагностики и оптимизации, но не должны быть единственной защитой физического процесса.

**Пример:** система мониторинга микроклимата считывает температуру и влажность раз в 20 секунд, формирует локальное предупреждение при превышении порога и передаёт историю в ThingSpeak. При пропадании сети Raspberry Pi продолжает сохранять измерения в локальном журнале.

# 2 Аппаратные интерфейсы и безопасное подключение

Для подключения цифровых датчиков Raspberry Pi может использовать GPIO, I²C, SPI и UART. GPIO обеспечивает дискретный ввод-вывод; I²C позволяет подключать устройства по адресуемой шине; SPI предназначен для синхронного обмена; UART — для последовательной передачи байтов.

Логические уровни GPIO Raspberry Pi обычно составляют 3,3 В. Подача 5 В непосредственно на вход может повредить плату. Реле, двигатели, вентиляторы и другие силовые нагрузки подключают через подходящие драйверы, ключи, опторазвязку или специализированные модули с соблюдением требований электробезопасности.

Большинство Raspberry Pi не имеют встроенного универсального аналогового входа. Для аналоговых датчиков требуется внешний АЦП или отдельный контроллер, который оцифровывает сигнал. Цифровые датчики могут передавать значения непосредственно по I²C, SPI или UART.

Перед лабораторной работой необходимо изучить документацию конкретной платы и датчика, проверить питание, назначение контактов, допустимый ток и требования к общей земле. Нельзя считать все датчики и модули взаимозаменяемыми только потому, что они используют одинаковый разъём.

# 2 Аппаратные интерфейсы и безопасное подключение (продолжение)

Для учебного прототипа рекомендуется начинать с датчика температуры и влажности, а не с исполнительного устройства. После проверки достоверности показаний можно добавить локальную индикацию на светодиоде через резистор или безопасный интерфейсный модуль.

# 3 Архитектура локального управления и облачной аналитики

Взаимодействие Raspberry Pi, ThingSpeak и MATLAB удобно представить как последовательность уровней: физический объект → датчик → Raspberry Pi → сеть → канал ThingSpeak → MATLAB-аналитика → визуализация и отчёт. Каждый уровень имеет собственные ограничения и источники ошибок.

На физическом уровне датчик преобразует температуру или влажность в измерительный сигнал. На уровне Raspberry Pi программа добавляет временную метку, проверяет диапазон, сохраняет запись и при необходимости выполняет локальную логику. На сетевом уровне сообщение передаётся по HTTPS.

ThingSpeak обеспечивает облачное хранение и визуализацию временных рядов. MATLAB используется для анализа записей, построения графиков, вычисления статистик и разработки диагностических алгоритмов. Доступность конкретных возможностей зависит от аккаунта, лицензии и текущих условий сервиса.

Для систем мониторинга типична асинхронная архитектура: измерения поступают периодически, а анализ выполняется по мере необходимости. Для управляющих воздействий важно учитывать задержки, недоступность сервиса и возможное повторное получение сообщений.

3

# Архитектура локального управления и облачной аналитики (продолжение)

Если интернет недоступен, локальный сбор и локальная защита должны продолжать работать. После восстановления соединения сохранённые данные можно передать с исходными временными метками, если выбранный способ загрузки и права доступа поддерживают такой сценарий.

# 4 ThingSpeak: каналы, поля и ключи доступа

**ThingSpeak —** облачная платформа для сбора, хранения, визуализации и анализа IoT-данных, интегрированная с возможностями MATLAB. Основная логическая единица хранения — канал (channel). Канал содержит временные записи, а измеряемые параметры обычно размещаются в отдельных полях (fields).

Например, в учебном канале Field 1 можно назначить температуре, Field 2 — относительной влажности. Названия полей следует задать в настройках канала, а единицы измерения указать в описании. Это помогает избежать путаницы при дальнейшей обработке.

Для доступа к каналу применяются идентификатор канала и ключи API с различными правами. Write API Key предназначен для записи, Read API Key — для чтения закрытого канала. Секретные ключи не следует размещать в публичных отчётах, презентациях или открытых репозиториях.

Каждая запись может содержать временную метку и значения нескольких полей. При обработке следует учитывать, что отсутствующие значения могут быть представлены как пропуски. Необходимо отдельно проверять корректность и полноту рядов.

Частота обновления и другие ограничения зависят от текущих условий аккаунта и лицензии. Перед экспериментом следует проверить действующие ограничения ThingSpeak, а не предполагать, что произвольная частота передачи допустима.

# 5 Подготовка Raspberry Pi и формирование телеметрии

Перед передачей данных Raspberry Pi необходимо подключить к сети, настроить системное время, установить программную среду и проверить считывание датчика. Рекомендуется начать с тестовых чисел и убедиться, что облачный канал принимает сообщения, прежде чем подключать реальный сенсор.

Программа сбора должна разделять измерение, проверку и отправку. Функция измерения возвращает значение и признак успешности; функция проверки определяет, находится ли оно в физически допустимом диапазоне; функция отправки выполняет сетевой запрос и анализирует ответ.

Показания желательно снабжать временными метками в UTC и локальным порядковым номером. Это помогает выявлять пропуски и дубликаты. Если сеть недоступна, данные сохраняются в файл или локальную базу до восстановления соединения.

В учебном примере можно использовать искусственные значения температуры 25,4 °C и влажности 61 %. Их необходимо явно обозначать как моделируемые данные. Для реального проекта вместо этих констант следует использовать вызов проверенного драйвера датчика.

Программа не должна бесконечно повторять запросы при ошибке. Требуется ограничить число попыток, предусмотреть паузы и вести журнал отказов. Это защищает устройство и сервис от чрезмерной нагрузки.

# 6 Передача данных в ThingSpeak через HTTP

ThingSpeak предоставляет веб-интерфейсы для загрузки измерений. В простом сценарии Raspberry Pi формирует HTTPS-запрос к API обновления канала, передавая ключ записи и значения полей. Точный формат запроса и доступные параметры необходимо сверять с актуальной документацией сервиса.

Для учебного канала с температурой в Field 1 и влажностью в Field 2 формируется сообщение с параметрами field1 и field2. После отправки программа проверяет HTTP-статус и содержимое ответа, поскольку успешное установление сетевого соединения ещё не доказывает, что запись принята.

В Python можно использовать библиотеку requests для отправки POST-запроса. Ключ API следует получать из защищённой конфигурации или переменной окружения, а не записывать непосредственно в учебный файл. Пример ниже показывает структуру запроса, но требует собственного канала и ключа.

При ошибке сети или отклонении запроса данные следует сохранить локально и повторить отправку позднее. Нужно учитывать ограничения частоты обновления и не выполнять повторные запросы с чрезмерно коротким интервалом.

6

# Передача данных в ThingSpeak через HTTP (продолжение)

Для диагностики полезно логировать время измерения, номер записи, код ответа, длительность запроса и причину ошибки. В отчёте нельзя публиковать секретный ключ, даже если канал предназначен только для лабораторной работы.

```python
# Python на Raspberry Pi: учебный пример отправки
import os
import requests
api_key = os.environ["THINGSPEAK_WRITE_KEY"]
url = "https://api.thingspeak.com/update"
payload = {
    "api_key": api_key,
    "field1": 25.4,   # учебная температура, °C
    "field2": 61.0    # учебная влажность, %
}
try:
    response = requests.post(url, data=payload, timeout=10)
    response.raise_for_status()
```

6

# Передача данных в ThingSpeak через HTTP (продолжение)

```
    entry_id = response.text.strip()
    if entry_id == "0":
        raise RuntimeError("ThingSpeak не принял запись")
    print("Идентификатор записи:", entry_id)
except (requests.RequestException, RuntimeError) as exc:
    print("Ошибка передачи:", exc)
    # В реальном проекте: запись в локальный буфер
```

# 7 Чтение данных ThingSpeak в MATLAB

MATLAB позволяет получать данные канала ThingSpeak с помощью специализированных функций, если они доступны в используемой среде. Типовой подход заключается в указании идентификатора канала и номера поля; для закрытого канала дополнительно требуется ключ чтения.

Функция thingSpeakRead может возвращать значения измерений и временные метки. Перед анализом следует проверить размерности, наличие данных, формат времени и конечность значений. При отсутствии записей программа должна корректно завершать обработку, а не выдавать неверный средний показатель.

Для построения графика используется plot(time,values). Подписи осей должны содержать единицы, а заголовок — указывать источник и характер данных. Если временные метки представлены как datetime, MATLAB поддерживает их непосредственное отображение по оси времени.

При необходимости можно получать несколько полей и рассчитывать взаимосвязи температуры и влажности. Однако корреляция двух рядов не доказывает причинную зависимость, особенно при наличии суточных трендов или общего влияющего фактора.

# 7 Чтение данных ThingSpeak в MATLAB (продолжение)

Пример ниже требует реального идентификатора канала и, для закрытого канала, ключа чтения. Он не создаёт измерения автоматически и не подтверждает наличие данных в каком-либо канале.

```
% MATLAB: пример чтения Field 1 из ThingSpeak
channelID = 123456;  % заменить реальным ID
readKey = 'YOUR_READ_API_KEY'; % хранить безопасно
[T,time] = thingSpeakRead(channelID, ...
    'Fields',1,'NumPoints',100, ...
    'ReadKey',readKey);
valid = isfinite(T);
if any(valid)
    figure;
    plot(time(valid),T(valid),'-o');
    grid on;
    xlabel('Время'); ylabel('Температура, °C');
    title('Измерения из ThingSpeak');
```

# 7 Чтение данных ThingSpeak в MATLAB (продолжение)

```
else
    disp('Нет корректных данных для анализа');
end
```

8

# MATLAB-аналитика: фильтрация и диагностические показатели

После получения временного ряда в MATLAB вычисляют среднее значение, минимум, максимум, разброс и число превышений порога. Для массива T среднее определяется как mean(T), а стандартное отклонение — std(T). Перед вычислением необходимо исключить или отдельно обработать недостоверные значения.

Для подавления случайного шума можно использовать скользящее среднее movmean(T,w), где w — размер окна в отсчётах. Увеличение окна уменьшает колебания, но может сгладить кратковременные события. Поэтому диагностический алгоритм должен учитывать компромисс между шумоподавлением и задержкой реакции.

Скорость изменения температуры оценивается по формуле ΔT/Δt. При неравномерных временных метках следует использовать фактические интервалы, а не условный постоянный шаг. Если время представлено как datetime, разность можно перевести в секунды.

Пороговый контроль T>=Tmax возвращает логический массив. Количество превышений определяется sum(T>=Tmax), а их моменты — логической индексацией. Для предотвращения частых переключений около порога может применяться гистерезис.

8

# MATLAB-аналитика: фильтрация и диагностические показатели (продолжение)

Статистический анализ следует дополнять физической интерпретацией. Например, рост температуры может быть нормальным этапом нагрева, а не неисправностью. Правила предупреждения должны соответствовать конкретному технологическому процессу.

```
% MATLAB: аналитика учебного ряда
T = [24.8 25.1 25.5 NaN 26.3 27.0 28.2 28.8];
valid = isfinite(T);
Tv = T(valid);
Tfiltered = movmean(Tv,3);
threshold = 28;
fprintf('Среднее: %.2f °C\n',mean(Tv));
fprintf('Максимум: %.2f °C\n',max(Tv));
fprintf('Превышений: %d\n',sum(Tv>=threshold));
plot(Tv,'o-'); hold on;
plot(Tfiltered,'-','LineWidth',1.5);
yline(threshold,'--','Порог');
grid on; legend('Данные','Скользящее среднее');
```

# 9 Время, ограничения частоты и буферизация

Период измерения Δt и период отправки в облако не обязаны совпадать. Например, датчик может опрашиваться каждую секунду, а данные отправляться пакетами реже. Такая архитектура снижает число сетевых запросов, но требует локального хранения промежуточных отсчётов.

При работе с ThingSpeak необходимо учитывать ограничения на частоту обновления канала, размер запросов и допустимый объём данных. Эти параметры зависят от текущих условий сервиса. Планирование эксперимента следует начинать с проверки документации и параметров учётной записи.

Если измерения приходят чаще, чем их можно передать, Raspberry Pi должен использовать очередь или буфер. При переполнении буфера требуется заранее определённая политика: остановить сбор, удалить старые записи либо сохранить данные на диск. Для научного эксперимента предпочтительно явно регистрировать потери.

Для корректного анализа важна исходная временная метка измерения, а не только время получения записи облаком. Сетевые задержки могут нарушать равномерность наблюдаемого ряда. При загрузке исторических данных необходимо убедиться, что выбранный API поддерживает сохранение исходных меток.

# 9 Время, ограничения частоты и буферизация (продолжение)

Для синхронизации времени полезно использовать системные механизмы NTP. При сбоях синхронизации программа должна отмечать недостоверные временные метки и не рассчитывать скорость изменения параметра по ошибочным интервалам.

# 10 Обратная связь и границы облачного управления

IoT-система может не только публиковать измерения, но и получать команды от внешнего приложения. Однако команда, поступившая через интернет, может быть задержана, потеряна или повторена. Поэтому проектирование обратной связи требует отдельного анализа рисков.

Для учебной демонстрации допустимо рассмотреть облачную команду как запрос на изменение режима индикации или на запись в журнал. Для управления нагревателями, двигателями и другими физическими устройствами следует применять локальные ограничения и защитные блокировки.

**Типовая архитектура разделяет два контура:** локальный быстрый контур управления и облачный медленный контур наблюдения или задания некритичных параметров. Локальный контур должен сохранять работоспособность при отсутствии сети.

Перед исполнением внешней команды необходимо проверить источник, допустимость значения, актуальность временной метки и текущее состояние оборудования. Желательно использовать подтверждения и защиту от повторного выполнения.

В инженерном отчёте следует чётко различать «мониторинг», «удалённую настройку» и «автоматическое управление». Наличие графика ThingSpeak и MATLAB-скрипта само по себе не означает, что реализована замкнутая система управления.

# 11 Информационная безопасность и отказоустойчивость

Ключи API предоставляют доступ к данным и функциям сервиса. Их следует хранить вне исходного кода, разграничивать права чтения и записи и обновлять при подозрении на раскрытие. Для передачи данных предпочтительно использовать HTTPS.

Raspberry Pi является сетевым компьютером и требует обычных мер защиты: обновлений, ограниченного доступа, надёжной аутентификации, отключения ненужных служб и контроля журналов. Открытые сетевые порты увеличивают поверхность атаки.

При отказе облака или интернета программа должна продолжать локальный сбор и при необходимости локальное предупреждение. Необходимо вести журнал ошибок, ограничивать число повторов и избегать блокирования основного цикла измерения на длительном сетевом запросе.

Достоверность данных проверяется на нескольких уровнях: электрический сигнал, программный драйвер, локальный журнал, сетевой запрос и облачная запись. Ошибка на любом уровне может исказить аналитический вывод.

Перед демонстрацией следует проверить работу при отключении Wi-Fi, пустом ответе API, неверном ключе, NaN от датчика и переполнении локального буфера. Эти испытания важнее, чем демонстрация только штатного режима.

# 12 Сквозной проект: мониторинг микроклимата

**Постановка:** создать учебную систему мониторинга температуры и влажности на базе Raspberry Pi, ThingSpeak и MATLAB. На первом этапе используются искусственно сформированные значения. На втором этапе они заменяются показаниями реального датчика, а результаты передаются в созданный студентом канал.

**Структура канала:** Field 1 — температура, °C; Field 2 — относительная влажность, %. Raspberry Pi измеряет и проверяет значения, сохраняет их локально, а затем отправляет по HTTPS с соблюдением ограничений частоты обновления. MATLAB читает историю и строит графики.

**Алгоритм проверки:** температура и влажность должны быть конечными; влажность — находиться в пределах 0–100 %. Для температуры физически допустимый диапазон определяется характеристиками конкретного датчика. Некорректные значения не передаются как обычные измерения.

Для демонстрации аналитики зададим учебные пороги: температура не ниже 30 °C или влажность не ниже 75 %. При превышении формируется предупреждение. Эти значения выбраны только для учебного примера и не являются нормативными.

12

# Сквозной проект: мониторинг микроклимата (продолжение)

Результат лабораторного проекта должен включать структурную схему, описание датчика и интерфейса, скрипт сбора данных, конфигурацию канала без секретных ключей, графики MATLAB, статистические показатели и описание поведения при ошибках связи.

Критериями качества служат корректность единиц измерения, отсутствие потерь без регистрации, воспроизводимость анализа, понятная визуализация, проверка ошибочных данных и соблюдение электрической и информационной безопасности.

```
% MATLAB: автономный учебный пример анализа
clear; clc; close all;
time_s = (0:20:140).';
temp_C = [24.5;25.1;26.0;27.4;29.2;30.1;30.6;29.8];
humidity_pct = [58;60;62;66;70;73;76;74];
valid = isfinite(temp_C) & isfinite(humidity_pct) ...
    & humidity_pct>=0 & humidity_pct<=100;
warning = valid & (temp_C>=30 | humidity_pct>=75);
data = table(time_s,temp_C,humidity_pct,valid,warning);
```

12

# Сквозной проект: мониторинг микроклимата (продолжение)

```
disp(data);
fprintf('Предупреждений: %d\n',sum(warning));
figure;
plot(time_s,temp_C,'-o','LineWidth',1.5); hold on;
yline(30,'--','Учебный порог 30 °C');
grid on; xlabel('Время, с'); ylabel('Температура, °C');
title('Учебный мониторинг микроклимата');
```

# 3 Заключение

Рассмотрена роль Raspberry Pi как одноплатного вычислительного узла IoT, изучены аппаратные интерфейсы, структура каналов ThingSpeak, принципы передачи данных через HTTP и обработки записей в MATLAB. Отдельное внимание уделено частоте обновления, временным меткам, локальной буферизации, защите ключей API и разделению облачной аналитики и локального безопасного управления. Сквозной проект показывает полный путь от измерения до визуализации и диагностического вывода.

# Контрольные вопросы (1/2)

1. Чем Raspberry Pi отличается от типичного микроконтроллера?

2. Какие функции Raspberry Pi выполняет в IoT-архитектуре?

3. Почему облачный сервис не должен быть единственным контуром аварийной защиты?

4. Какие интерфейсы Raspberry Pi используются для подключения датчиков?

5. Почему необходимо соблюдать уровень 3,3 В на GPIO?

6. Как подключают аналоговые датчики к Raspberry Pi?

7. Какие уровни содержит архитектура «датчик — Raspberry Pi — ThingSpeak — MATLAB»?

8. Что такое канал ThingSpeak?

9. Как используются поля Field 1 и Field 2?

10. Для чего нужны ключи Read API Key и Write API Key?

11. Почему нельзя публиковать ключи доступа?

12. Как формируется HTTP-запрос на отправку измерения?

13. Почему важно проверять ответ API после отправки?

# Контрольные вопросы (2/2)

14. Как MATLAB получает данные ThingSpeak?

15. Какие проверки выполняют перед обработкой временного ряда?

16. Как вычислить среднее, максимум и число превышений?

17. Каковы ограничения скользящего среднего?

18. Чем отличается период измерения от периода публикации?

19. Почему необходимо учитывать ограничения частоты обновления ThingSpeak?

20. Как организовать локальный буфер при отсутствии сети?

21. Почему важны временные метки UTC?

22. Какие риски возникают при облачном управлении исполнительными устройствами?

23. Как обеспечить устойчивость к сетевым ошибкам?

24. Какие испытания необходимы перед демонстрацией системы?

25. Что должен содержать отчёт по учебному IoT-проекту?

# Самостоятельные задания

1. Разработайте структурную схему подключения датчика температуры и влажности к Raspberry Pi с передачей в ThingSpeak.

2. Сформируйте тестовую таблицу из 30 измерений с шагом 20 секунд и реализуйте проверку достоверности.

3. Создайте в MATLAB скрипт визуализации температуры и влажности с обозначением учебных порогов.

4. Опишите алгоритм локального буферирования при отключении сети на 5 минут.

5. Сравните возможности локального мониторинга Raspberry Pi и облачной аналитики ThingSpeak.

6. Подготовьте перечень проверок электрической безопасности, защиты API-ключей и обработки сетевых ошибок.

# Рекомендуемые источники

- Силлабус дисциплины «Проект MATLAB IoT», тема лекции №8.
- MathWorks. ThingSpeak Documentation — https://www.mathworks.com/help/thingspeak/
- ThingSpeak API Documentation — https://www.mathworks.com/help/thingspeak/rest-api.html
- MathWorks. MATLAB Documentation — https://www.mathworks.com/help/matlab/
- Raspberry Pi Documentation — https://www.raspberrypi.com/documentation/