Лабораторная работа 3.1 «Схемы распределительных систем с IoT. Виды и типы.»

(Методические указания по проекту STUDY_FOG)

ВВЕДЕНИЕ

Соблюдение стандартов, таких как ГОСТ, обеспечивает единообразие, понятность и универсальность создаваемых схем. Это позволяет разным специалистам «говорить на одном языке» при проектировании, сборке и обслуживании сложных систем. В данной работе мы будем ориентироваться на стандарты ГОСТ 2.701 и 2.702 для классификации и правил выполнения схем.

Цель лабораторной работы:

Изучить принципы построения схем распределительных систем IoT в соответствии со стандартами ГОСТ, освоить инструменты Google Colab, PlantUML Editor и draw.io для визуализации архитектуры и приобрести навык создания структурных и функциональных диаграмм (Deployment и State) для IoT-системы.

Документы и стандарты

ГОСТ 2.701 – «Единая система конструкторской документации (ЕСКД). Схемы. Виды и типы. Общие требования к выполнению». Назначение и классификация схем.

ГОСТ 2.702 – «Единая система конструкторской документации. Правила выполнения электрических схем». Графика, обозначения и связи.

ГОСТ 34.201 – «Информационные технологии. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем». Состав и обозначение документов для автоматизированных систем.

ГОСТ Р 57193 – «Системная и программная инженерия. Процессы жизненного цикла систем». Полный жизненный цикл системы, включая

замысел, разработку, производство, эксплуатацию и снятие с эксплуатации систем, приобретение и постава систем.

ISO/IEC/IEEE 15288

Типы схем по ГОСТ 2.701

Структурная – основные функциональные части (Edge/Fog/Cloud) и связи между ними.

Функциональная – процессы/потоки данных и управления (события, задержки и т.д.).

Принципиальная – уровень элементов для электроники и логики (датчики, интерфейсы).

Соединений/подключения/расположения – топология портов, шин, кабелей, размещения узлов и т.д.

Диаграммы для распределённых систем IoT (UML/SysML ↔ ГОСТ)

• *Deployment* – узлы и сети: Edge-устройства, Fog-шлюз (Home_Gateway), Cloud-сервисы → структурная схема.

• *Component* – сервисы и драйверы (MQTT-брокер, Ingestor, Threshold/ML) → структурная/функциональная.

• *Sequence/Activity* – обмены и потоки (publish/subscribe, алерты) → функциональная.

• *State* – режимы устройств/узлов (online/offline, in_work) → функциональная (состояний).

• *Package/Context* – границы подсистем, ответственность и уровни.

Для описания распределенных систем IoT наиболее наглядными и практичными являются **структурные** и **функциональные** схемы. В отличие от детализированных принципиальных схем, они позволяют абстрагироваться от аппаратной реализации и сосредоточиться на ключевых компонентах, связях и потоках данных, что делает их понятными для более широкой аудитории, включая не-инженеров. В контексте программной инженерии этим целям лучше всего соответствуют диаграммы UML/SysML. В данной работе мы сосредоточимся на:

Deployment – диаграмме (UML) как аналоге структурной схемы (ГОСТ): она показывает «из чего состоит система» — устройства, узлы и физические соединения.

State – диаграмме (UML) как аналоге функциональной схемы состояний (ГОСТ): она показывает «как ведет себя компонент» — его реакцию на события и смену состояний.

Необходимые навыки:

Базовые навыки работы с компьютером, умение пользоваться веб-браузером. Опыт работы с графическими редакторами или программированием приветствуется, но не является обязательным.

1. Компоненты системы

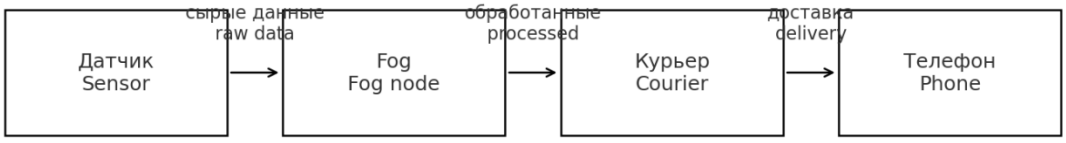
Система состоит из пяти компонентов, распределенных по уровням вычислений:

Уровни архитектуры:

- **Краевой уровень (Edge): Робот-датчик** — сбор сырых данных
- **Туманный уровень (Fog): Fog-обработчик** — локальная обработка
- **Облачный уровень (Cloud): Ноутбук-сервер** — аналитика и хранение

Связующие компоненты:

- **Робот-курьер** — транспортировка данных
- **Смартфон** — буферизация и передача"



Путь задержки / Latency path:
Датчик→Fog→Курьер→Телефон
Sensor→Fog→Courier→Phone

Рисунок 1 – схема работы компонентов системы

Следующая схема представляет собой логическую модель компонентов распределенной системы "Умный дом". Она показывает основные функциональные блоки и направление потока данных между ними. Каждый компонент выполняет строго определенную роль в цепочке обработки информации: от сбора данных (робот-датчик) до их конечного анализа (ноутбук-сервер).

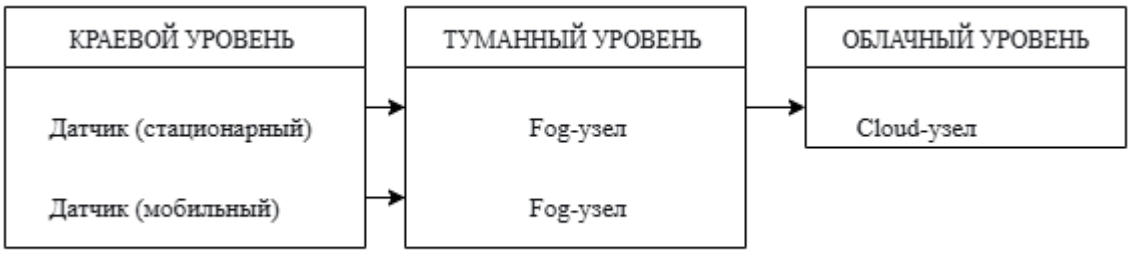


Рисунок 2 - Схема архитектуры системы

Схема на рисунке 3 является **технической реализацией** логической модели, показанной на Рисунке 2. Она детализирует как именно компоненты взаимодействуют на практике, показывает сетевые интерфейсы, протоколы передачи данных и физическую структуру системы. Здесь наглядно отображено разделение на сетевые сегменты (например, локальную сеть датчиков и облачный сервер).

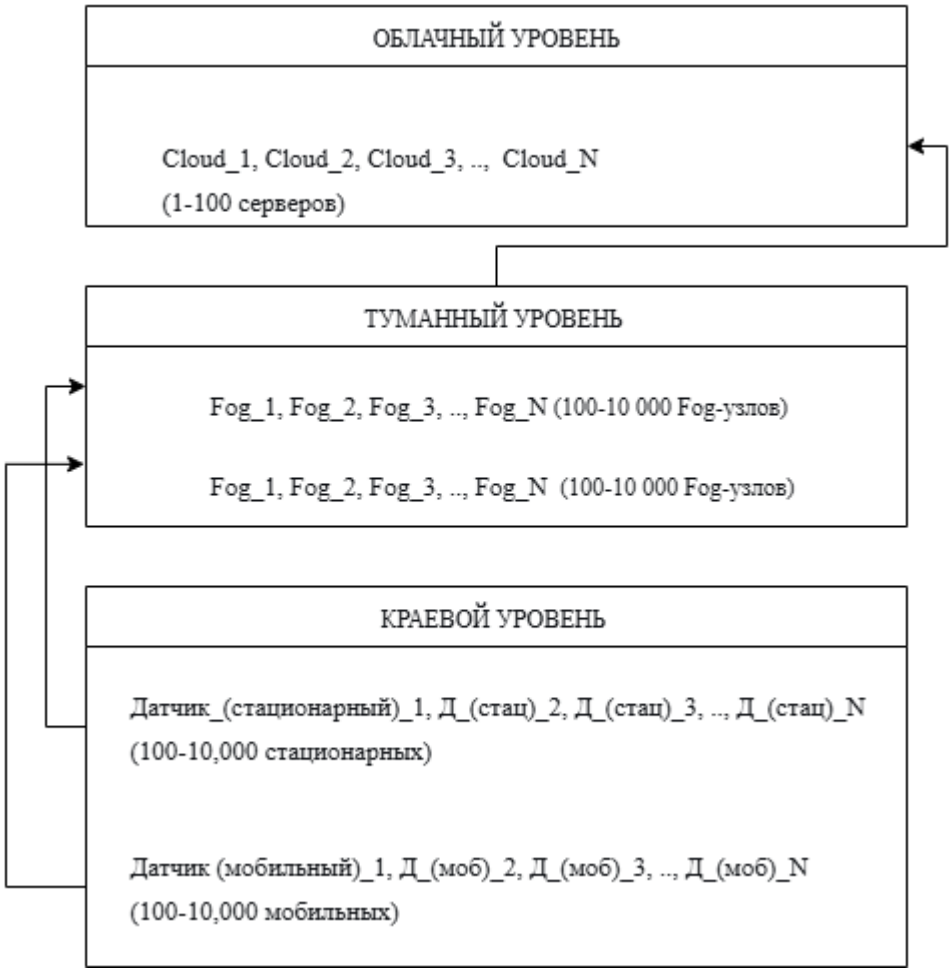


Рисунок 3 – Схема эталонной архитектуры системы

Взаимосвязь и взаимодействие между схемами

Обе схемы описывают одну и ту же систему, но на разных уровнях абстракции:

- **Рисунок 2** отвечает на вопрос **"ЧТО делает система?"** — показывает основные роли и поток данных.
- **Рисунок 3** отвечает на вопрос **"КАК система это делает?"** — показывает техническую реализацию и сетевые взаимодействия.

Краткое описание взаимодействия:

1. **Робот-датчик** собирает данные и передает их **Fog-обработчику** по локальному сетевому протоколу (например, Wi-Fi или Bluetooth)

2. **Fog-обработчик** выполняет первичную обработку и передает результат **роботу-курьеру** через локальную сеть

3. **Робот-курьер** доставляет данные **смартфону** через интернет (мобильная сеть 4G/5G)

4. **Смартфон** буферизует данные и передает их **ноутбуку-серверу** через облачное API или прямое сетевое соединение

2. GOOGLE COLAB

Google Colab – бесплатная облачная среда разработки, которая позволяет писать и выполнять код на Python прямо в браузере.

Diagrams – библиотека для Python, которая позволяет программно создавать диаграммы архитектуры облачных систем.

2.1 Пример работы в Google Colab

1. Откройте новый блокнот на https://colab.research.google.com.

2. В новой ячейке установите и импортируйте библиотеку для построения графиков:

>>> !pip install diagrams

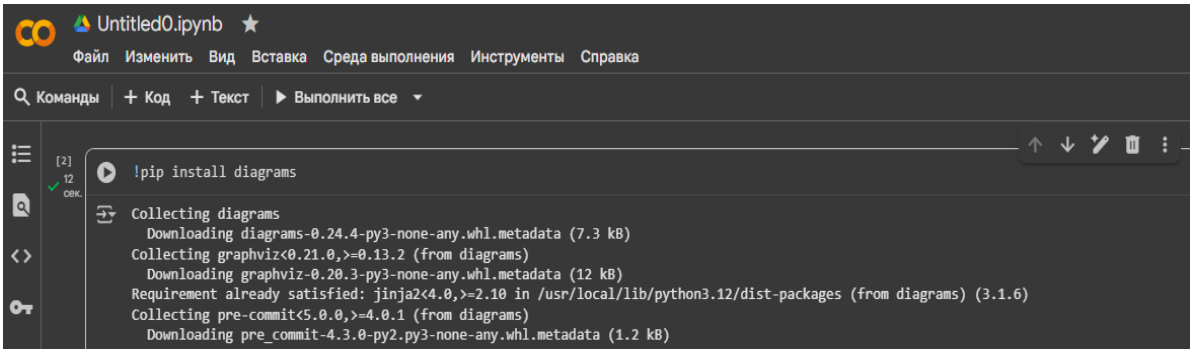


Рисунок 4 – Реализация команды для установки и импорта библиотеки

3. Создайте новую ячейку и скопируйте в нее следующий код для визуализации архитектуры:

```
from diagrams import Diagram
from diagrams.aws.compute import EC2
```

```
from diagrams.aws.network import InternetGateway
from diagrams.onprem.client import User  # Замена для mobile.Client
from diagrams.onprem.compute import Server
from diagrams.aws.iot import IotSensor  # Более подходящий компонент для датчика

# Укажите путь для сохранения картинки. В Colab это будет текущая директория.
with Diagram("Архитектура IoT системы 'Умный дом'", show=False, filename="iot_architecture"):
    # Определяем ноды (компоненты) системы с более подходящими иконками
    sensor = IotSensor("Робот-датчик")
    fog = EC2("Fog-обработчик")
    courier = InternetGateway("Робот-курьер")
    smartphone = User("Смартфон\n(Буфер)")
    server = Server("Ноутбук-сервер")

    # Строим конвейер обработки данных
    sensor >> fog >> courier >> smartphone >> server

print("Схема сохранена как 'iot_architecture.png'")
print("Файл будет доступен в левой панели управления файлами Colab")
```

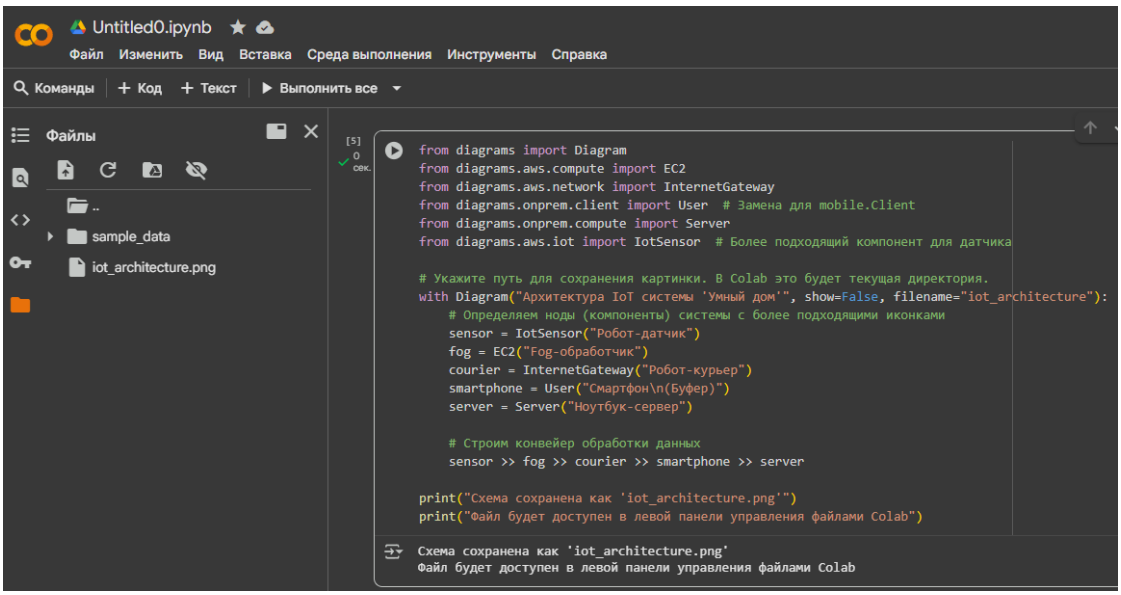


Рисунок 5 – Реализация кода

Запустите ячейку. После выполнения код создаст файл iot_architecture.png в левой панели управления файлами. Откройте его, чтобы увидеть схему вашей системы.

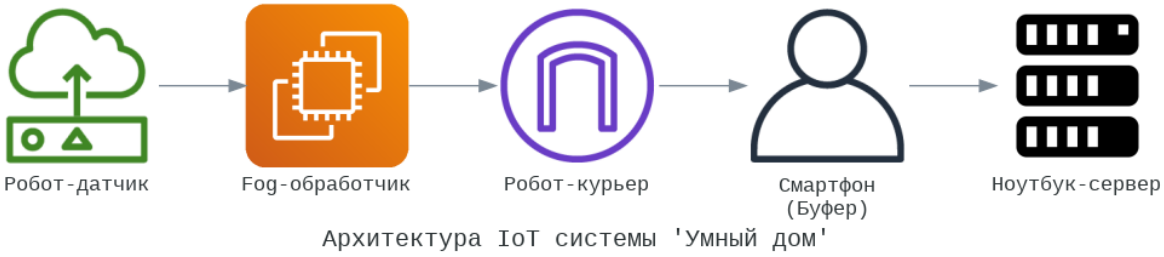


Рисунок 6 – Полученная диаграмма

2.2 Задание для построения схемы через Google Colab

Используя пример из раздела 2.1, постройте **Deployment-диаграмму** для системы "Умный дом", отображающую физические узлы и связи между ними.

Измените код следующим образом:

- Добавьте второго "Робота-датчика"

- Добавьте второго "Fog-обработчика"
- Измените направление данных: первый датчик → первый Fog, второй датчик → второй Fog
- Оба Fog-обработчика должны передавать данные на "Ноутбук-сервер"
- Измените иконки для объектов

Примечание: Для просмотра всех доступных иконок и их названий в библиотеке Diagrams посетите официальную документацию: https://diagrams.mingrammer.com/docs/nodes/aws

На этом сайте вы найдете полный список категорий (AWS, Azure, GCP, On-premise и др.) с примерами кода для каждого элемента.

3. PLANTUML EDITOR

PlantUML Editor — это инструмент для создания, редактирования и визуализации диаграмм с помощью простого текстового языка.

3.1 Пример работы в PlantUML Editor

1. Откройте онлайн-редактор PlantUML: https://www.plantuml.com/plantuml/

2. Скопируйте и вставьте следующий код для создания Deployment-диаграммы в панель редактора.

Листинг:

```
@startuml
skinparam shadowing false
skinparam defaultFontName Arial
skinparam nodesep 10
skinparam ranksep 20

title Deployment-диаграмма IoT системы "Умный дом"

node "Краевой уровень (Edge)" as edge_layer {
```

[Робот-датчик 1]
[Робот-датчик 2]
[Робот-датчик N]
}

node "Туманный уровень (Fog)" as fog_layer {
(Fog-обработчик 1)
(Fog-обработчик 2)
}

cloud "Облачный уровень (Cloud)" as cloud_layer {
(Ноутбук-сервер)
}

[Робот-датчик 1] --> (Fog-обработчик 1) : Wi-Fi
[Робот-датчик 2] --> (Fog-обработчик 1) : BLE
[Робот-датчик N] --> (Fog-обработчик 2) : LoRaWAN
(Fog-обработчик 1) --> (Ноутбук-сервер) : HTTPS
(Fog-обработчик 2) --> (Ноутбук-сервер) : MQTT
@enduml

```
@startuml
skinparam shadowing false
skinparam defaultFontName Arial
skinparam nodesep 10
skinparam ranksep 20

title Deployment-диаграмма IoT системы "Умный дом"

node "Краевой уровень (Edge)" as edge_layer {
  [Робот-датчик 1]
  [Робот-датчик 2]
  [Робот-датчик N]
}

node "Туманный уровень (Fog)" as fog_layer {
  (Fog-обработчик 1)
  (Fog-обработчик 2)
}

cloud "Облачный уровень (Cloud)" as cloud_layer {
  (Ноутбук-сервер)
}

[Робот-датчик 1] --> (Fog-обработчик 1) : Wi-Fi
[Робот-датчик 2] --> (Fog-обработчик 1) : BLE
[Робот-датчик N] --> (Fog-обработчик 2) : LoRaWAN
(Fog-обработчик 1) --> (Ноутбук-сервер) : HTTPS
(Fog-обработчик 2) --> (Ноутбук-сервер) : MQTT
@enduml
```

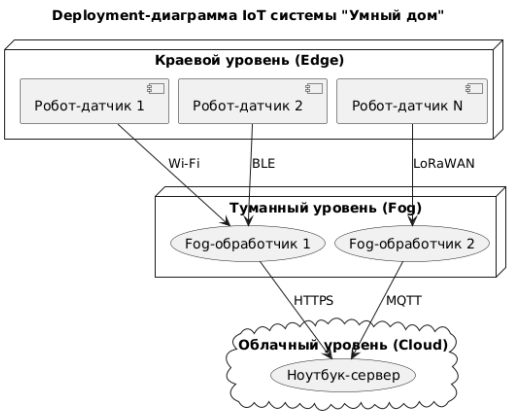


Рисунок 7 – Структурная схема, созданная с помощью PlantUML

3. Диаграмма должна появиться сама, вы увидите структурную схему, отображающую узлы системы и связи между ними.

4. Теперь создадим State-диаграмму. Очистите редактор и вставьте следующий код:

```
@startuml
' Раздел 3.1: Пример State-диаграммы в PlantUML
skinparam defaultFontName Arial

title State-диаграмма Fog-обработчика

[*] --> Ожидание

Ожидание --> Сбор_данных : получен запрос
Сбор_данных --> Обработка : данные считаны
Обработка --> Передача : результат готов
Передача --> Ожидание : отправлено подтверждение

Ожидание --> Перегрузка : очередь переполнена
Перегрузка --> [*] : аварийное отключение
@enduml
```

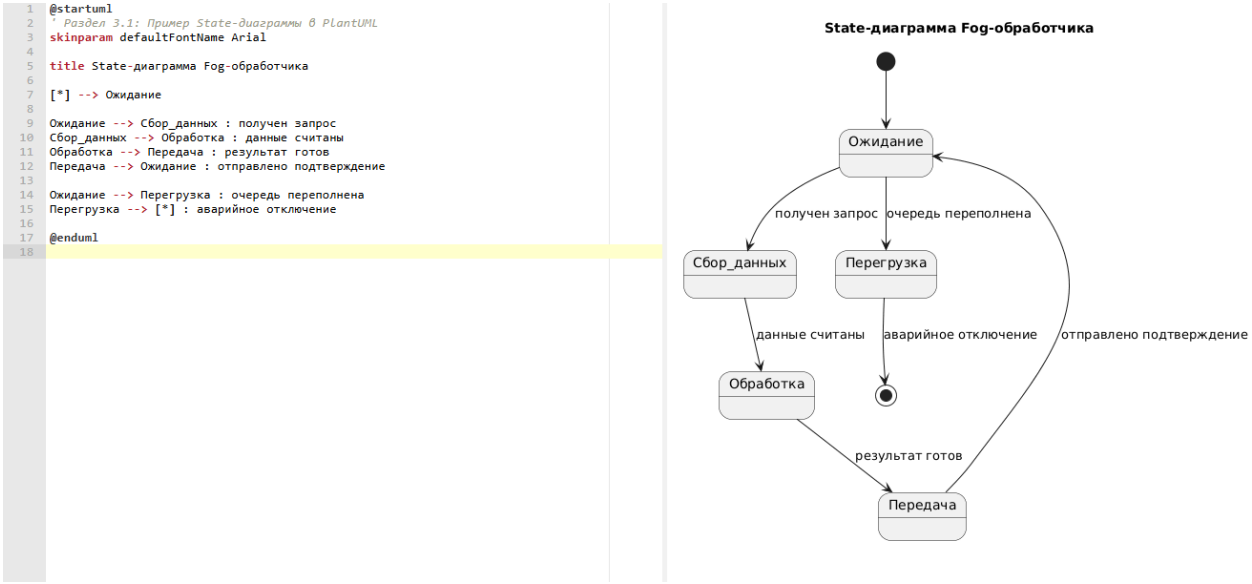


Рисунок 8 – State-диаграмма

3.2 Задание для построения схемы через PlantUML Editor

Используя примеры выше, создайте в PlantUML Editor:

1. **Deployment-диаграмму**, отображающую не менее 3 краевых устройств, 2 Fog-узлов и 1 облачного сервера. Измените названия компонентов и типы соединений на свои (например, ZigBee, Ethernet).

2. **State-диаграмму** для "Робота-курьера". Добавьте состояния, например: Зарядка, Ожидание_задания, Навигация_к_смартфону, Передача_данных. Сохраните обе диаграммы как PNG-файлы для отчета.

4. DRAW.IO

Draw.io (с 2020 года официально называется **Diagrams.net**) — бесплатный онлайн-сервис для создания диаграмм, блок-схем и визуальных схем любой сложности.

4.1 Пример работы в Draw.io

1. Перейдите на сайт https://app.diagrams.net/.

2. Создайте новую диаграмму ("File" -> "New").

3. В левой панели выберите раздел "Network". Перетащите на холм иконки, соответствующие компонентам IoT-системы:

- **Дачик (Sensor)** для робота-датчика.
- **Компьютер (Computer)** или **Сервер (Server)** для Fog-обработчика и ноутбука-сервера.
- **Мобильное устройство (Mobile)** для смартфона.
- Из раздела "General" добавьте стрелки и блоки для соединений и подписей.

4. Соедините компоненты линиями, имитируя поток данных от датчика к серверу. Подпишите каждый компонент и линию связи (протокол).

4.2 Задание для построения схемы через Draw.io

Повторите в Draw.io логическую модель системы (аналог Рисунка 2 из данного документа), используя базовые фигуры (прямоугольники, стрелки) из раздела "General". Схема должна наглядно показывать пять основных компонентов системы и направление потока данных между ними. Сохраните диаграмму в формате PNG.

5. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Используя один из изученных инструментов (Google Colab, PlantUML Editor или Draw.io), постройте две диаграммы для распределенной IoT-системы, соответствующей вашему варианту из таблицы ниже:

Deployment-диаграмму (структурная схема), отображающую физические узлы системы (краевые устройства, Fog-узлы, облачные серверы) и связи между ними.

State-диаграмму (функциональная схема состояний), описывающую изменение состояния одного из Fog-узлов (например: Режим ожидания -> Получение задачи -> Обработка -> Передача результата -> Перегрузка/Ожидание).

За основу возьмите трехуровневую архитектуру «Край-Туман-Облако». Количество устройств на каждом уровне и общую характеристику системы смотрите в вашем варианте, который показан в Таблице 1.

Таблица 1 – Варианты индивидуальных заданий

Вариант	**Краевые устройства (Edge)**	**Fog-узлы**	**Облачные серверы (Cloud)**	**Характеристика системы**
1	5	2	1	Много Edge на Fog
2	3	3	1	Сбалансированное
3	4	2	2	Равномерное распределение
4	6	2	1	Высокая нагрузка на Fog
5	3	4	1	Избыток Fog-узлов
6	5	3	2	Оптимальное соотношение
7	7	2	1	Потенциальное узкое место
8	4	1	2	Минимум Fog-ресурсов
9	6	3	1	Умеренная нагрузка
10	5	2	3	Мощная облачная инфраструктура

6. СОСТАВЛЕНИЕ ОТЧЁТА

Заполните отчёт по выполнению лабораторной работы. Сделайте выводы.