Изображение выглядит как эмблема, символ, герб, нашивка

Автоматически созданное описание

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА** - **Российский технологический университет"**  **РТУ МИРЭА** |

**Институт** Информационных Технологий

**Кафедра** Математического обеспечения и стандартизации информационных технологий (МОСИТ)

**Отчёт по практическим работам №5-8**

**по дисциплине**

**«Технологические основы Интернета вещей»**

Выполнили студенты группы: ИКБО-04-22 Кликушин В.И.

Егоров Л.А.

Корольков А.Д.

Приняла ассистент \_\_Куликова И.В.\_\_ *(Ф.И.О. преподавателя)*

|  |  |  |
| --- | --- | --- |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2024

# Содержание

[1 Практическая работа №5 по теме «Измерительные и исполнительные устройства в Интернете вещей» 4](#_Toc181444827)

[1.1 Введение 4](#_Toc181444828)

[1.2 Датчик температуры 1-wire DS18B20 4](#_Toc181444829)

[1.3 Датчик уровня шума в составе WB-MS v-3 4](#_Toc181444830)

[1.4 Модуль реле трёхканальный WB-MR3 5](#_Toc181444831)

[1.5 Описание протоколов работы с устройствами 6](#_Toc181444832)

[1.5.1 Modbus RTU 6](#_Toc181444833)

[1.5.2 Wire 7](#_Toc181444834)

[1.5.3 I²C (Inter-Integrated Circuit) 8](#_Toc181444835)

[1.5.4 CAN (Controller Area Network) 8](#_Toc181444836)

[1.6 Вывод 9](#_Toc181444837)

[2 Практическая работа № 6 по теме «Основы работы с протоколом MQTT. Брокераж сообщений» 10](#_Toc181444838)

[2.1 Введение 10](#_Toc181444839)

[2.2 Ход выполнения работы 10](#_Toc181444840)

[2.3 Вывод 12](#_Toc181444841)

[3 Практическая работа №7 по теме «Форматы представления данных» 13](#_Toc181444842)

[3.1 Введение 13](#_Toc181444843)

[3.2 Ход выполнения работы 13](#_Toc181444844)

[3.3 Вывод 18](#_Toc181444845)

[4 Практическая работа №8 по теме «Визуализация данных в Интернете вещей» 19](#_Toc181444846)

[4.1 Введение 19](#_Toc181444847)

[4.2 Ход выполнения работы 19](#_Toc181444848)

[4.3 Вывод 23](#_Toc181444849)

[Заключение 24](#_Toc181444850)

# 1 Практическая работа №5 по теме «Измерительные и исполнительные устройства в Интернете вещей»

## 1.1 Введение

Цель работы: познакомиться с устройствами в составе стенда и их характеристиками.

## 1.2 Датчик температуры 1-wire DS18B20

**Название датчика/устройства**: Датчик температуры 1-wire DS18B20.

**Тип измерения**: Аналоговый**.**

**Измеряемые параметры и диапазон измерения**: Температура, диапазон измерения -55–125 °C.

**Точность**: ±0.5°C**.**

**Напряжение питания**: 3–5.5 В.

**Уникальный идентификатор датчика в веб-интерфейсе**: wb-m1w2-14/External/Sensor2**.**

**Использующийся протокол передачи данных**: 1-wire.

**Интерфейс управления (шина)**: 1-wire**.**

**Описание входов и выходов, схема подключения**: на вход датчика подаётся окружающий воздух, на выходе – сигнал, содержащий информацию о температуре воздуха, схема подключения: три входа – для данных, земли и питания.

## 1.3 Датчик уровня шума в составе WB-MS v-3

**Название датчика/устройства**: Датчик уровня шума в составе WB-MS v-3.

**Тип измерения**: Аналоговый**.**

**Измеряемые параметры и диапазон измерения**: Уровень шума, диапазон измерения 38–105 дБа.

**Точность**: ±1 дБа**.**

**Напряжение питания**: 9–28 В.

**Уникальный идентификатор датчика в веб-интерфейсе**: wb-msw-v3-21/Sound/Level**.**

**Использующийся протокол передачи данных**: Modbus RTU.

**Интерфейс управления (шина)**: RS-485**.**

**Описание входов и выходов, схема подключения**: на вход датчика подаётся звук, на выходе – сила звука в дБа, датчик является частью комплексного датчика и дополнительных действий для подключения не требует.

## 1.4 Модуль реле трёхканальный WB-MR3

**Название датчика/устройства**: Модуль реле с RS-485.

**Тип измерения**: Аналоговый**.**

**Измеряемые параметры и диапазон измерения**: реле, переменный ток 2–16 А.

**Точность**: ±1 А**.**

**Напряжение питания**: 9–28 В.

**Уникальный идентификатор датчика в веб-интерфейсе**: WB-MR3-56**.**

**Использующийся протокол передачи данных**: Modbus RTU.

**Интерфейс управления (шина)**: RS-485**.**

**Описание входов и выходов, схема подключения**: датчик имеет четыре входа, три из которых управляют своим выходами, а четвёртый полностью отключает все каналы реле, три аналоговых выхода, на выходе управляющие сигналы. Входы подключаются к управляющим элементам, выходы – к нагрузкам.

## 1.5 Описание протоколов работы с устройствами

### 1.5.1 Modbus RTU

Принцип работы:

Modbus RTU (Remote Terminal Unit) — это протокол передачи данных, основанный на запросах и ответах между устройствами. Главный контроллер (мастер) запрашивает данные от одного или нескольких подчиненных устройств (ведомых), передавая команды и получая данные. Обмен данными осуществляется через последовательный интерфейс (чаще всего RS-485 или RS-232). Формат данных – последовательный, с использованием четности и контрольной суммы CRC для обеспечения точности передачи.

Преимущества:

* Простота реализации и универсальность;
* Широко распространен в промышленной автоматизации;
* Поддерживает длинные линии связи (до 1200 метров) и большое количество устройств (до 247);
* Высокая надежность и возможность использования в шумных средах (особенно с RS-485).

Недостатки:

* Ограниченная скорость передачи данных (обычно до 115200 бит/с);
* Протокол не поддерживает динамическое подключение устройств;
* Нет встроенной системы приоритетов — возможны задержки в больших сетях.

Сфера применения:

* Промышленная автоматизация (контроллеры, датчики, устройства управления);
* Умные дома;

Энергетические системы (мониторинг и управление).

### 1.5.2 Wire

Принцип работы:

1-Wire — это протокол для передачи данных по одной линии, где одна и та же линия используется для питания, синхронизации и передачи данных. В устройстве могут быть ведущий и множество ведомых, которые идентифицируются по уникальным 64-битным идентификаторам. Передача данных осуществляется с помощью серийных сигналов, а энергоснабжение ведомых устройств — через ту же линию.

Преимущества:

* Простота проводки — всего одна линия для питания и данных;
* Экономичность — низкая стоимость как реализации, так и самого оборудования;
* Возможность подключения большого количества устройств на одну шину.

Недостатки:

* Низкая скорость передачи данных (до 16.3 кбит/с);
* Ограниченная длина линии (до 100 метров);
* Подверженность шумам, что может влиять на стабильность передачи данных на больших расстояниях.

Сфера применения:

* Сенсоры температуры (например, DS18B20);
* Устройства учета (счетчики);
* Применения в бытовой технике, умных домах.

### 1.5.3 I²C (Inter-Integrated Circuit)

Принцип работы:

I²C — это двунаправленный двухпроводный протокол (SDA для данных и SCL для тактового сигнала) для связи между интегральными схемами. Ведущий (мастер) управляет тактовым сигналом и инициирует передачу данных. Ведомые устройства (слейвы) принимают или передают данные по команде мастера. Каждый ведомый имеет уникальный адрес, который используется для выборки устройства.

Преимущества:

* Простота подключения множества устройств через два провода;
* Встроенные механизмы обнаружения ошибок и подтверждения передачи;
* Возможность динамического подключения устройств.

Недостатки:

* Ограниченная длина шины (до 1 метра) без специальных буферов;
* Относительно низкая скорость передачи данных (обычно до 400 кбит/с, максимум 3.4 Мбит/с в Fast Mode+);
* Возможны конфликты на шине при наличии большого количества устройств.

Сфера применения:

* Встроенные системы (микроконтроллеры, микросхемы);
* Сенсоры (температуры, давления, влажности);
* Электроника и периферия (дисплеи, память, RTC).

### 1.5.4 CAN (Controller Area Network)

Принцип работы:

CAN — это протокол передачи данных для связи между контроллерами в реальном времени, использующий архитектуру «множественный доступ с приоритетом по контролю столкновений» (CSMA/CA). Все устройства на шине могут передавать данные, но приоритеты сообщений определяют, какие из них будут обработаны первыми в случае коллизии. CAN поддерживает как точку-точку, так и многоточечные соединения.

Преимущества:

* Высокая надежность, благодаря механизму приоритета сообщений и обработке ошибок;
* Поддержка работы в реальном времени;
* Высокая скорость передачи данных (до 1 Мбит/с на стандартной линии длиной до 40 метров);
* Возможность использования в сложных и шумных условиях.

Недостатки:

* Сложность реализации по сравнению с простыми протоколами (например, I²C или 1-Wire);
* Ограниченная длина линии связи при высоких скоростях (на скоростях выше 500 кбит/с длина линии ограничена 100 метрами).

Сфера применения:

* Автомобильная промышленность (встроенные системы автомобиля);
* Промышленная автоматизация;
* Робототехника и системы управления реального времени.

## 1.6 Вывод

В ходе описания датчиков и устройств было проведено ознакомление с их характеристиками и принципами работы. Изучена документация устройств и их технические характеристики, а также протоколы передачи данных.

# 2 Практическая работа № 6 по теме «Основы работы с протоколом MQTT. Брокераж сообщений»

## 2.1 Введение

Цель работы: познакомиться с основами работы с протоколом MQTT и опробовать отправку сообщений на устройство посредством брокера сообщений.

## 2.2 Ход выполнения работы

Ход работы:

1. Подключиться к консоли WirenBoard посредством SSH.
2. При помощи утилиты mosquito\_sub подписаться на сообщения датчиков, представленных в Таблице 2.2.1:

Таблица 2.2.1 – Задание варианта

|  |  |
| --- | --- |
| № варианта | Датчики |
| 4 | 1. Датчик движения WB-MSW v3  2. Датчик температуры WB-MS v2 |

1. Включить или изменить поведение устройств посредством отправки сообщений в соответствующий топик.

Выполнение работы:

**Задание 1:**

Подключение по SSH осуществлялось с помощью команды, представленной в Листинге 2.1. Результат подключения представлен на Рисунке 2.1.

Листинг 2.1 – Подключение по SSH

ssh user@192.168.1.15

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 2.1 – Результат подключения по SSH

**Задание 2:**

Подписка на сообщение датчика движения представлена командой в Листинге 2.2. Данные, отправленные с датчика, представлены на Рисунке 2.2.

Листинг 2.2 – Подписка на сообщения датчика движения

mosquitto\_sub -t '/devices/wb-msw-v3\_21/controls/Current Motion' -v -p 1883

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2.2 – Данные с датчика движения

Подписка на сообщение датчика температуры представлена командой в Листинге 2.3. Данные, отправленные с датчика, представлены на Рисунке 2.3.

Листинг 2.3 – Подписка на сообщения датчика температуры

mosquitto\_sub -t '/devices/wb-ms\_11/controls/Temperature' -v -p 1883

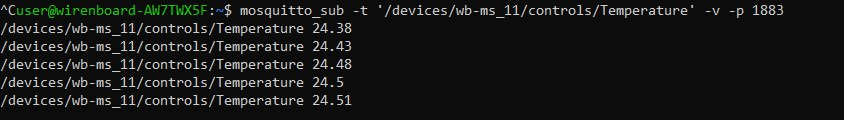


Рисунок 2.3 – Данные с датчика температуры

**Задание 3:**

Включение подсветки в кнопке 29 осуществлено командой в Листинге 2.4.

Листинг 2.4 – Команда включения подсветки

mosquitto\_pub -t '/devices/button\_light/controls/button3/on' -m "1" -p 1883

Включение имитации подачи воды осуществлено командой в Листинге 2.5:

Листинг 2.4 – Команда включения имитации подачи воды

mosquitto\_pub -t '/devices/water\_control/controls/valve' -m "1" -p 1883

## 2.3 Вывод

В ходе выполнения работы изучен протокол брокера сообщений MQTT, а также осуществлены подписка на сообщения от датчиков и отправка сигналов на датчики. При подписке на сообщения от датчиков в консоль отправляются данные с них после изменения значений. Также с помощью брокера сделана отправка сигналов на датчики с целью включения подсветки в кнопке и включения имитации подачи воды.

# 3 Практическая работа №7 по теме «Форматы представления данных»

## 3.1 Введение

Цель работы: изучить подключение к стенду посредством брокера сообщений MQTT, используя язык программирования MQTT и библиотеку paho-mqtt, выполнить сбор данных с датчиков и представить их в двух форматах: JSON и XML.

## 3.2 Ход выполнения работы

Ход работы:

* + - 1. На языке Python реализуйте программу (скрипт), которая бы каждые 5 секунд упаковывала последние полученные данные в файлы формата JSON и XML. В одной записи должно быть 6 полей: 4 показаний датчиков, время формирования файла, номер чемодана (последние две цифры IP-адреса). Датчики выбраны по персональному варианту №4: Датчик движения устройства WB-MSW v.3 (5), Датчик шума устройства WB-MSW v.3 (5), Датчик освещенности устройства WB-MS v.2 (12), Датчик температуры устройства WB-MS v.2 (12).
      2. На любом языке программирования реализуйте программу-парсер, которая бы выводила в консоль данные, полученные из сгенерированных в п.1 файлов.

Выполнение работы:

Структура данных, которая сохраняется в файлах, представлена в Листинге 3.1.

Листинг 3.1 – Структура данных с датчиков

motion: int

Окончание Листинга 3.1

sound\_level: float

illuminance: float

temperature: float

file\_time: datetime

case\_id: int

Для сохранения данных в формате XML и JSON написана программа на языке Python, код которой представлен в Листинге 3.2.

Листинг 3.2 – Код программы для пункта 1

import paho.mqtt.client as mqtt

import time

import json

import xml.etree.ElementTree as ET

import os

import csv

from datetime import datetime, timedelta

ip = '192.168.1.15'

port = 1883

topics = [

    '/devices/wb-msw-v3\_21/controls/Current Motion',

    '/devices/wb-msw-v3\_21/controls/Sound Level',

    '/devices/wb-msw-v3\_21/controls/Illuminance',

    '/devices/wb-msw-v3\_21/controls/Temperature',

]

current\_values = {}

result\_json = []

def connect\_mqtt():

    def on\_connect(client, userdata, flags, rc):

        if rc == 0:

            print("Успешное подключение к брокеру")

        else:

            print("Не удалось подключиться, код %d\n", rc)

    client = mqtt.Client()

    client.on\_connect = on\_connect

    client.connect(ip, port)

    return client

def subscribe(client, topic):

    def on\_message(client, userdata, msg):

        print(f"Received {msg.payload.decode()} from {msg.topic} topic")

        current\_values.update({msg.topic: msg.payload.decode()})

    client.subscribe(topic)

    client.on\_message = on\_message

def append\_json(motion, sound\_level, illuminance, temperature):

    global result\_json

    result\_json.append({

        'motion': motion,

        'sound\_level': sound\_level,

        'illuminance': illuminance,

Окончание Листинга 3.2

        'temperature': temperature,

        'file\_time': time.ctime(),

        'case\_id': ip.split('.')[-1]

    })

def write\_json():

    with open('data.json', 'w', encoding='utf-8') as file:

        json.dump(result\_json, file, indent=4)

def write\_xml():

    root = ET.Element('data')

    for item in result\_json:

        data\_item = ET.SubElement(root, 'item')

        for key in item:

            value = ET.SubElement(data\_item, key)

            value.text = item[key]

    with open('data.xml', 'w', encoding='utf-8') as file:

        file.write(ET.tostring(root, encoding='unicode'))

def main():

    client = connect\_mqtt()

    for topic in topics:

        subscribe(client, topic)

    client.loop\_start()

    end\_time = datetime.now() + timedelta(minutes=1)

    time.sleep(1)

    while datetime.now() < end\_time:

        temperature = current\_values.get(topics[0])

        humidity = current\_values.get(topics[1])

        air\_quality = current\_values.get(topics[2])

        sound\_level = current\_values.get(topics[3])

        append\_json(temperature, humidity, air\_quality, sound\_level)

        time.sleep(5)

    write\_json()

    write\_xml()

    client.loop\_stop()

    print("Данные сохранены в XML и JSON файл в течение 1 минуты.")

if \_\_name\_\_ == '\_\_main\_\_':

    main()

Результат выполнения программы представлен на Рисунке 3.1. Содержимое файлов в формате JSON и XML представлено на Рисунках 3.2 – 3.3.

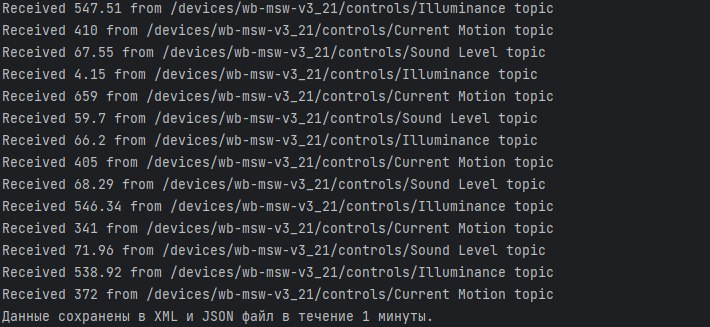


Рисунок 3.1 – Результат выполнения программы

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 3.2 – Данные в формате JSON

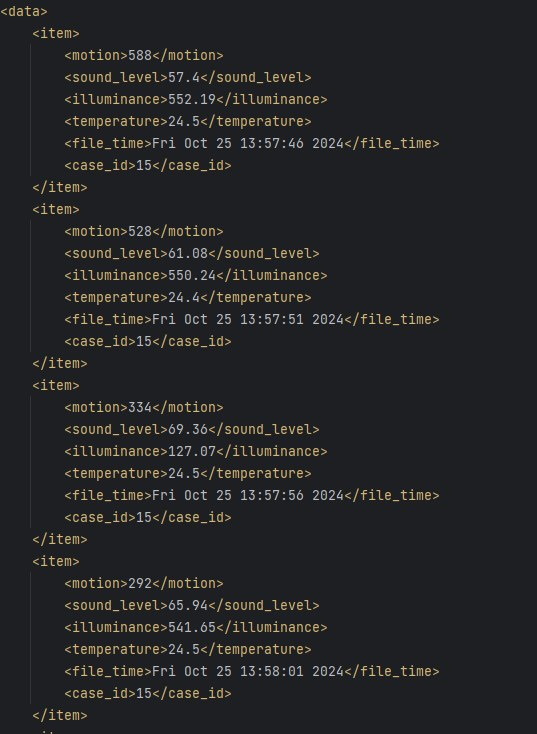


Рисунок 3.3 – Данные в формате XML

Для парсера данных в формате XML написана программа на языке Python, код которой представлен в Листинге 3.3. Результат выполнения программы представлен на Рисунке 3.4.

Листинг 3.3 – Парсер XML данных

import xml.etree.ElementTree as ET

xml\_data = ET.parse('data.xml')

result\_json = []

for item in xml\_data.findall('item'):

    item\_values = {'motion': None, 'sound\_level': None, 'illuminance': None, 'temperature': None, 'file\_time': None, 'case\_id': None}

    for key in item\_values:

Окончание Листинга 3.3

        item\_values[key] = item.find(key).text

    result\_json.append(item\_values)

print(result\_json)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.4 – Результат выполнения парсера

Для парсера данных в формате JSON написана программа на языке Python, код которой представлен в Листинге 3.4. Результат выполнения программы представлен на Рисунке 3.5.

Листинг 3.4 – Парсер JSON данных

import json

with open('data.json', 'r') as file:

    result = json.load(file)

print(result)

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 3.5 – Результат выполнения парсера

## 3.3 Вывод

В ходе выполнения работы выполнено подключение к стенду посредством библиотеки paho-mqtt на языке Python и собраны данные в форматах JSON и XML. Данные в формате JSON легче собирать и в дальнейшем парсить, а XML требует построения иерархического дерева для хранения данных, которое сложнее составлять и парсить.

# 4 Практическая работа №8 по теме «Визуализация данных в Интернете вещей»

## 4.1 Введение

Цель работы: изучить подключение к стенду посредством брокера сообщений MQTT, используя язык программирования MQTT и библиотеку paho-mqtt, выполнить сбор данных с датчиков в формате CSV и представить их в виде графиков.

## 4.2 Ход выполнения работы

Ход работы:

1. Подпишитесь на несколько MQTT топиков стенда согласно вариантам c компьютера в аудитории или личного устройства и собирайте данные с датчиков в течение 10 или более минут. Получаемые данные должны сохраняться в локальную базу данных или CSV-файл (на выбор). Датчики выбраны по персональному варианту №4: Датчик движения устройства WB-MSW v.3 (5), датчик температуры устройства WB-MS v.2 (12), напряжение на любом устройстве стенда.
2. Напишите скрипты, позволяющие визуализировать полученные статистические данные в виде столбчатой диаграммы, линейного графика и круговой диагрммы.

Выполнение работы:

Для сбора данных с датчиков в течение 10 минут написана программа на языке Python, код которой представлен в Листинге 4.1.

Листинг 4.1 – Код программы для сбора данных

import paho.mqtt.client as mqtt

import time

from datetime import datetime, timedelta

import pandas as pd

Продолжение Листинга 4.1

ip = '192.168.1.15'

port = 1883

topics = [

    '/devices/wb-msw-v3\_21/controls/Current Motion',

    '/devices/wb-msw-v3\_21/controls/Temperature',

    '/devices/wb-map12e\_23/controls/Ch 1 P L2'

]

current\_values = {}

result\_json = []

def connect\_mqtt():

    def on\_connect(client, userdata, flags, rc):

        if rc == 0:

            print("Успешное подключение к брокеру")

        else:

            print("Не удалось подключиться, код %d\n", rc)

    client = mqtt.Client()

    client.on\_connect = on\_connect

    client.connect(ip, port)

    return client

def subscribe(client, topic):

    def on\_message(client, userdata, msg):

        print(f"Received {msg.payload.decode()} from {msg.topic} topic")

        current\_values.update({msg.topic: msg.payload.decode()})

    client.subscribe(topic)

    client.on\_message = on\_message

def append\_json(motion, temperature, power):

    global result\_json

    result\_json.append({

        'motion': motion,

        'temperature': temperature,

        'power': power,

        'file\_time': time.ctime(),

        'case\_id': ip.split('.')[-1]

    })

def write\_csv():

    df = pd.DataFrame(result\_json)

    df.to\_csv('data.csv', index=False)

def main():

    client = connect\_mqtt()

    for topic in topics:

        subscribe(client, topic)

    client.loop\_start()

    end\_time = datetime.now() + timedelta(minutes=10)

    time.sleep(1)

    while datetime.now() < end\_time:

        motion = current\_values.get(topics[0])

        temperature = current\_values.get(topics[1])

Окончание Листинга 4.1

        power = current\_values.get(topics[2])

        append\_json(motion, temperature, power)

        time.sleep(5)

    write\_csv()

    client.loop\_stop()

    print("Данные сохранены в csv в течение 10 минут.")

if \_\_name\_\_ == '\_\_main\_\_':

    main()

Далее написана программа, реализующая показ трёх графиков, используя библиотеку matplotlib для языка Python, код которой представлен в Листинге 4.2. Графики представлены на Рисунках 4.1 – 4.3.

Листинг 4.2 – Код показа графиков данных

import matplotlib.pyplot as plt

import pandas as pd

def draw\_bar\_plot(data: pd.DataFrame):

    df = data['motion']

    df\_max = df.max()

    df\_min = df.min()

    first = int((df\_max - df\_min) / 3 + df\_min)

    second = int((df\_max - df\_min) / 3 + first)

    first\_df = df.loc[df.between(df\_min, first)]

    second\_df = df.loc[df.between(first, second)]

    third\_df = df.loc[df.between(second, df\_max)]

    df\_len = [len(first\_df), len(second\_df), len(third\_df)]

    plt.bar([f'[{df\_min};{first}]', f'[{first};{second}]', f'[{second};{df\_max}]'], df\_len)

    plt.xlabel('Интервалы показаний датчика движения')

    plt.ylabel('Частоты показаний')

    plt.show()

def draw\_plot(data: pd.DataFrame):

    df = data['temperature']

    plt.plot(df)

    plt.ylabel('Показания температуры')

    plt.show()

def draw\_pie\_plot(data: pd.DataFrame):

    df = data['power']

    df\_max = df.max()

    df\_min = df.min()

    first = round((df\_max - df\_min) / 3 + df\_min, 2)

    second = round((df\_max - df\_min) / 3 + first, 2)

    first\_df = (df.loc[df.between(df\_min, first)])

    second\_df = df.loc[df.between(first, second)]

    third\_df = df.loc[df.between(second, df\_max)]

    df\_len = [len(first\_df), len(second\_df), len(third\_df)]

    plt.pie(df\_len, labels=[f'[{df\_min};{first}]', f'[{first};{second}]', f'[{second};{df\_max}]'])

    plt.title('Показания датчика напряжения, В')

Окончание Листинга 4.2

    plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

    df = pd.read\_csv('data.csv')

    draw\_bar\_plot(df)

    draw\_plot(df)

    draw\_pie\_plot(df)

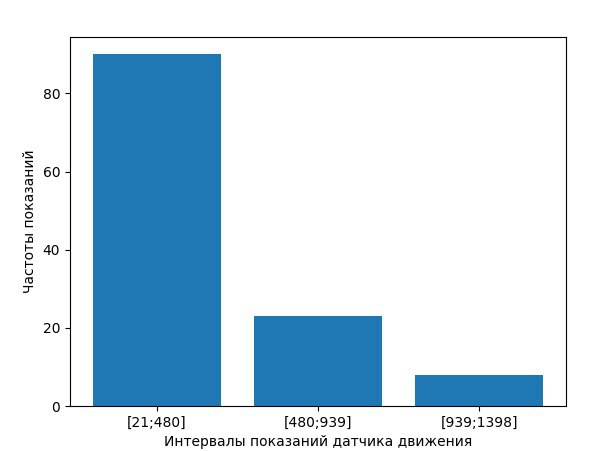


Рисунок 4.1 – Столбчатая диаграмма частотности показаний датчика движения по интервалам

Изображение выглядит как текст, линия, снимок экрана, диаграмма

Автоматически созданное описание

Рисунок 4.2 – Линейный график показаний температуры с течением времени

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

Рисунок 4.3 – Круговая диаграмма показаний датчика напряжения по интервалам

## 4.3 Вывод

В ходе выполнения работы освоены навыки работы с библиотеками языка Python: paho-mqtt для подключения к стенду и подписки на сообщения, и matplotlib для визуализации полученных данных. Столбчатая диаграмма показала, что в течение 10 минут показания датчика движения были достаточно низкими, что говорит о том, что около стенда мало двигались. Линейный график показаний температуры показал, что температура за время измерений незначительно увеличилась. Круговая диаграмма показания напряжения показала, что напряжение преимущественно находилось в одном и том же диапозоне, и никогда резко не изменялось.

# Заключение

В ходе выполнения практических работ 5-8 мы глубже погрузились в изучение архитектуры и взаимодействия устройств Интернета вещей. В рамках работы мы познакомились с измерительными и исполнительными устройствами, а также познакомились с различным протоколам связи (Modbus RTU, 1-Wire, I²C, CAN).

Часть работы включала подключение к контроллеру через SSH для настройки и управления устройствами, подписку на MQTT-топики для мониторинга данных с датчиков в реальном времени.   
Также, мы написали скрипты, которые каждые 5 секунд собирали данные с датчиков и сохраняли их в форматах JSON, XML и CSV. Мы также реализовали парсеры для этих данных и использовали их для визуализации собранной информации.