МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий
Кафедра Математического обеспечения и стандартизации
информационных технологий

Отчет по практическим работам 9-12

по дисциплине

«Технологические основы Интернета вещей»

Москва 2021

Выполнили: студенты группы ИВБО-02-19

К. Ю. Денисов

И. А. Кремнев

А. М. Сосунов

Д. Н. Федосеев

Принял: ассистент

Ю. А. Воронцов

СОДЕРЖАНИЕ

0.1 Практическая работа №9. Знакомство с облачными платформами IoT

0.1.1 Регистрация на платформе ThingsBoard

ThingsBoard имеет тестовый сервер в сборке Community Edition для проверки доступных функций платформы и тестирования своих приложений. Для регистрации на платформе необходимо перейти по данной ссылке.

Зарегистрируемся на платформе ThingsBoard для выполнения данных практических работ.

0.1.2 Создание виртуальных устройств в облаке

Создадим в облаке следующие виртуальные устройства для получения данных:

а) Датчик качества воздуха;

б) Датчик освещенности;

в) Датчик напряжения.

Создадим для каждого устройства свой профиль (виртуальное устройство), соответствующий передаваемым на устройство данным. В качестве протокола для профилей устройств используем **MQTT** (см. Рисунок 0.1,0.2).

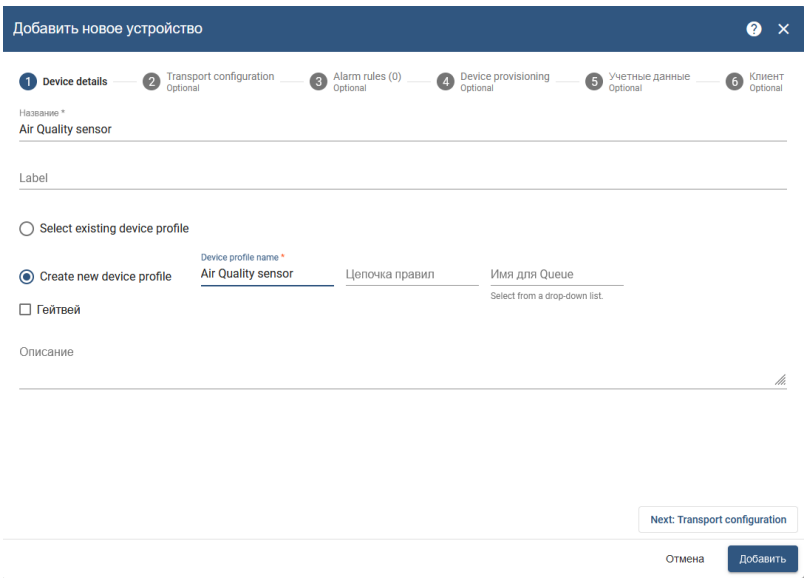


Рисунок 0.1 — Создание устройства на платформе ThingsBoard

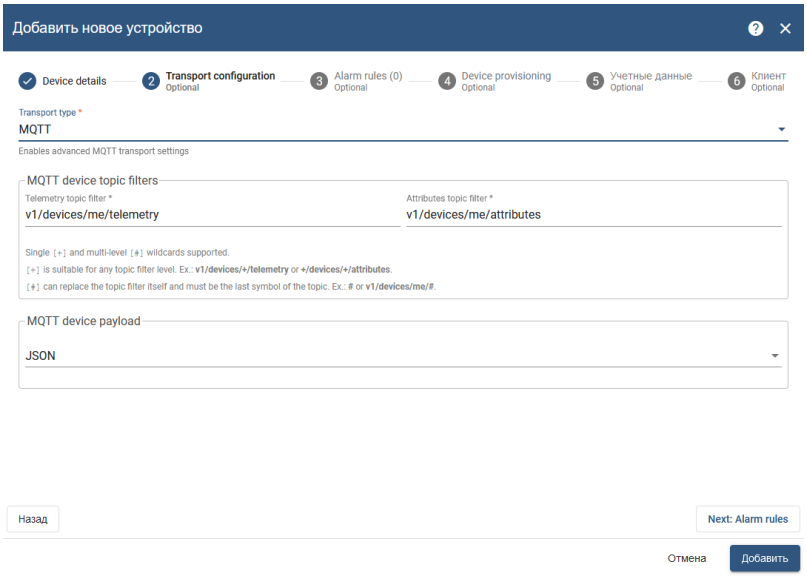


Рисунок 0.2 — Настройка устройства на платформе ThingsBoard

0.1.3 Отправка данных в облако

Выполним передачу тестовых данных в каждое из созданных устройств, список которых приведен на Рисунке 0.3.

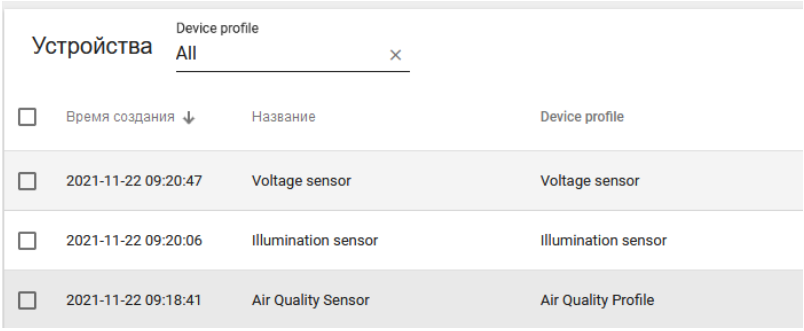


Рисунок 0.3 — Список зарегистрированных устройств

Приведем команду, с помощью которой осуществляется процесс ответа на сообщение с телеметрией в топик устройства с параметром "motion"(см. Рисунок (см. Рисунок 0.4).

Рисунок 0.4 — Отправка данных на устройства

После отправки, тестовые данные отображаются в веб-интерфейсе платформы ThingsBoard в виде представленном на Рисунках 0.5–0.7.

Данные соответствуют типу устройства и передаются при помощи утилиты `mosquito_pub`.

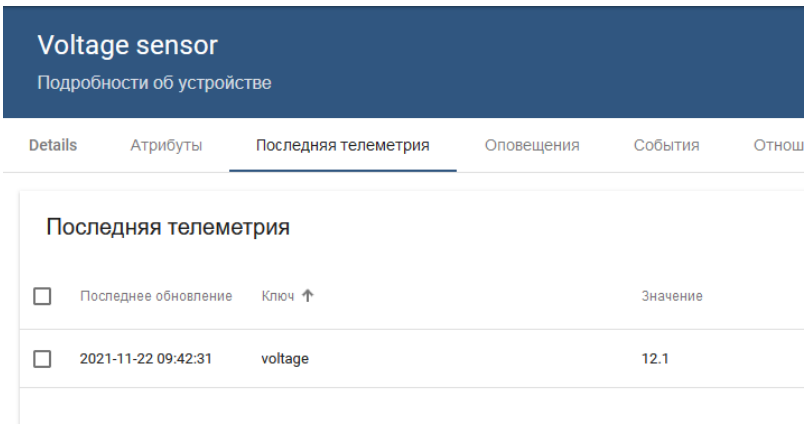


Рисунок 0.5 — Данные с датчика напряжения

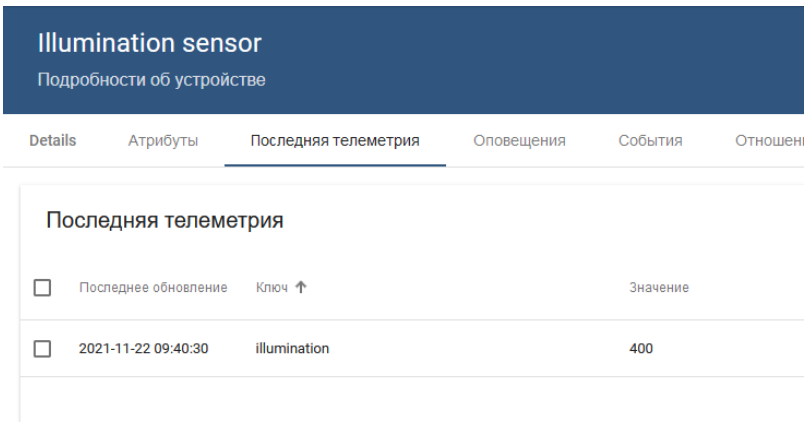


Рисунок 0.6 — Данные с датчика освещения

0.2 Дополнительное задание практической работы №9

0.2.1 Выбор облачного решения

В качестве облачного решения была выбрана платформа ThingsBoard. Данный выбор был сделан по следующим причинам:

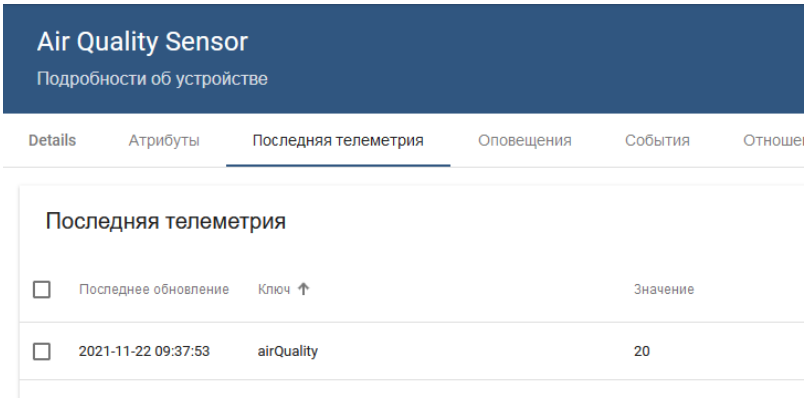


Рисунок 0.7 — Данные с датчика качества воздуха

— Понятный пользовательский интерфейс;

— Популярность платформы;

— Возможность установить платформу локально или использовать готовую облачную среду.

0.2.2 Реализация отправки данных

Реализуем отправку данных с программного эмулятора реального физического устройства в облачную платформу ThingsBoard. Приведем листинг скрипта на языке программирования Python:

```
1 import paho.mqtt.client as paho
2 import sys
3 import json
4 import schedule
5 import datetime
6 import time
7 import random
8
```

```
# Connection parameters to the MQTT broker
broker="demo.thingsboard.io"
port=1883

USERNAME = "i1Ohu4cIi2Q7OqUAHf21" # Login to connect to the broker


def job():
    now = datetime.datetime.now()
    motion = random.randint(20, 800)
    noise = random.randint(20, 800)
    doorState = random.randint(0, 1)
    print("[" + now.strftime("%H:%M %d.%m.%Y") + "] data: " +
       "motion: " + str(motion) + ", noise: " + str(noise) + ",
       door: " + ("Closed" if doorState == 0 else "Open"))

    data = {
        "timestamp" : now.isoformat(),
        "motion" : motion,
        "noise": noise,
        "door_open" : doorState
    }

    pahoClient.publish("v1/devices/me/telemetry", json.dumps(data))


def main():
    # Creating and configuring an instance of the Client class to
       connect to the MQTT broker
    global pahoClient
    pahoClient  = paho.Client("control1")
    pahoClient.username_pw_set(USERNAME)
    pahoClient.connect(broker,port)

    schedule.every(2).seconds.do(job)

    while 1:
```

```
43          schedule.run_pending()
44          time.sleep(1)
45
46
47  if __name__ == "__main__":
48      main()
```

В результате запуска данного скрипта происходит соединение с MQTT-брокером, создание экземпляра класса Client для MQTT-брокера, с последнующей передачей данных в облако (см. Рисунки 0.8, 0.9).

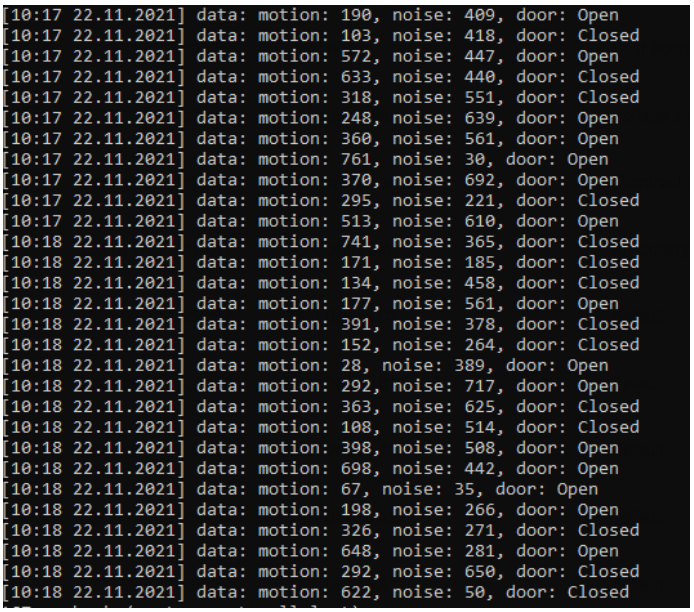
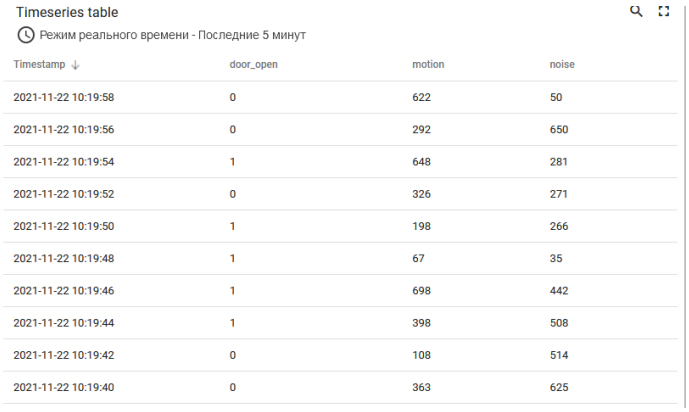


Рисунок 0.8 — Оправка телеметрических данных с устройств



Timeseries table

Режим реального времени - Последние 5 минут

Timestamp ↓	door_open	motion	noise
2021-11-22 10:19:58	0	622	50
2021-11-22 10:19:56	0	292	650
2021-11-22 10:19:54	1	648	281
2021-11-22 10:19:52	0	326	271
2021-11-22 10:19:50	1	198	266
2021-11-22 10:19:48	1	67	35
2021-11-22 10:19:46	1	698	442
2021-11-22 10:19:44	1	398	508
2021-11-22 10:19:42	0	108	514
2021-11-22 10:19:40	0	363	625

Рисунок 0.9 — Прием телеметрических данных облачной платформой

0.3 Практическая работа №10. Управление устройствами при помощи платформ Интернета вещей

Реализуем следующие сценарии из практической работы №3 при помощи цепочек правил ThingsBoard.

а) Включение и выключение вентилятора по датчику движения;

б) Включение и выключения индикации зеленым и красным светом комбинированного датчика по кнопкам.

0.3.1 Реализация сценария управления вентилятором

На платформе ThingsBoard создадим виртуальное устройство, которое будет прообразом реального вентилятора (см. Рисунок 0.10).

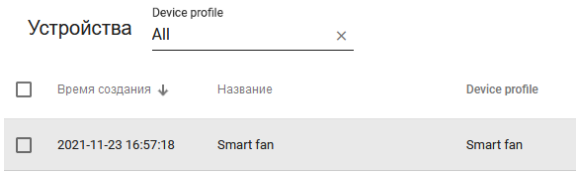


Рисунок 0.10 — Виртуальный вентилятор

Создадим цепочку правил для контроля за состоянием вентилятора. Когда значения, передаваемые датчиком движения превышают 700 условных единиц, вентилятор должен включаться. При уменьшении значения ниже 700, вентилятор должен выключаться. Приве-

денная на Рисунке 0.11 цепочка правил описывает данный сценарий управления устройством.

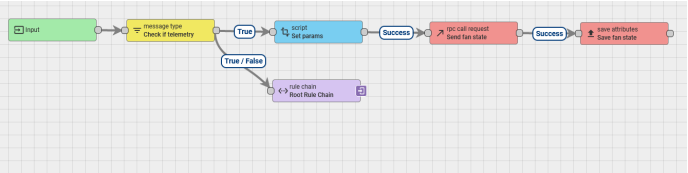


Рисунок 0.11 — Цепочка правил для управления вентилятором

Узел формирования параметров вентилятора

Узел трансформации данных при помощи скрипта позволяет переформировать объект, содержащий в себе данные приходящего сообщения: его основную полезную нагрузку, метаданные, а также тип сообщения.

Поведение узла описывается при помощи Java Script. Изначально в приходящей телеметрии предполагается наличие параметра `motion`. На основании этого параметра вычисляет новое состояние увлажнителя и формируется новый объект сообщения с этим состоянием.

Данный объект содержит в себе несколько свойств: *method* — это наименование метода, при помощи которого можно будет идентифицировать необходимое действие на устройстве, а также свойство *params*, содержащее как раз состояние устройства, в которое его необходимо привести.

В дальнейшем этот объект будет отправлен на конечное устройство для смены его состояния. Изменим тип события на событие загрузки атрибутов устройства — POST_ATTRUBUTE_REQUEST. Узел возвращает объект, содержащий в себе основное сообщение, метаданные, а также тип сообщения. Полный код приведен ниже.

```
function getNewFanState(motion){
    return motion > 700;
}

let newMsg = {};
let newMsgType = '';

newMsg = {
    "method" : "setFanState",
    "params":{
        "state": getNewFanState(msg.motion)
    }
};

newMsgType = "POST_ATTRIBUTES_REQUEST";


return {msg: newMsg, metadata: metadata, msgType: newMsgType};
```

После реализации узла формирования параметров можно приступать к тестированию созданной цепочки.

Подпишемся на топик запросов (v1/devices/me/rpc/request/+) созданного для данного виртуального вентилятора. Воспользуемся для этого следующую командой:

```
mosquitto_sub -v -h "demo.thingsboard.io" -t
   "v1/devices/me/rpc/request/+" -u "nS83gBMMgHxwPq4yXVW9"
```

После чего опубликуем сообщение с телеметрией в топик устройства с параметром `motion`:

```
mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t
   "v1/devices/me/telemetry" -u "nS83gBMMgHxwPq4yXVW9" -m
   "{"motion": 800}"
```

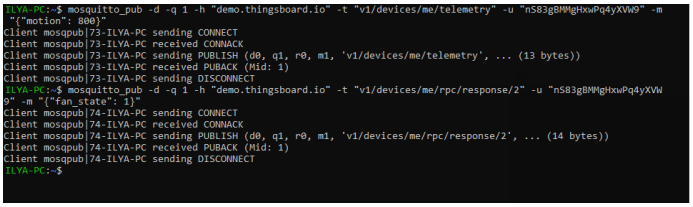


Рисунок 0.12 — Публикация сообщения с телеметрией вентилятора

Чтобы отправить ответ на опубликованный запрос на смену состояния, воспользуемся также утилитой `mosquito_pub`.

Для отправки ответа на созданный запрос необходимо послать сообщение в топик *v1/devices/me/rpc/response/2*. Воспользуемся для этого следующей командой:

```
mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t
   "v1/devices/me/rpc/response/2" -u "nS83gBMMgHxwPq4yXVW9" -m
   "{"fan_state": 1}"
```

После этого можно проверить в облаке поступившую телеметрию и аргументы устройства, посланные в ответ на запрос. Результаты представлены на Рисунках 0.13 и 0.14.

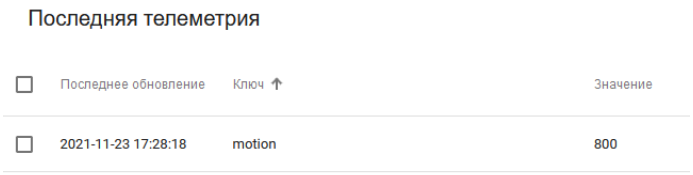


Рисунок 0.13 — Телеметрия облачного устройства «вентилятор»

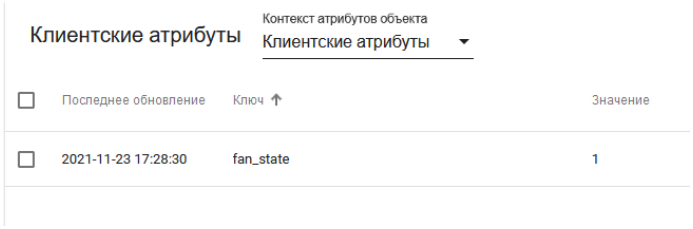


Рисунок 0.14 — Атрибуты облачного устройства «вентилятор»

0.3.2 Реализация сценария управления лампами

На платформе ThingsBoard создадим виртуальное устройство, которое будет прообразом реальных блока светодиодных ламп (см. Рисунок 0.15).

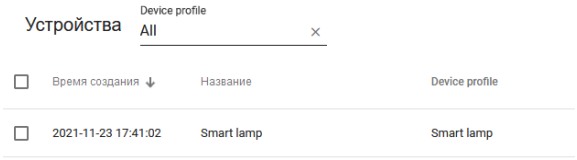


Рисунок 0.15 — Виртуальный блок ламп

Создадим цепочку правил для контроля за состоянием блока ламп. Световые индикаторы должны сигнализировать о включении и выключении комбинированного датчика. При включении датчика следует включить зеленую лампу, а при выключении — красную.

Приведенная на Рисунке 0.16 цепочка правил описывает данный сценарий управления устройством.

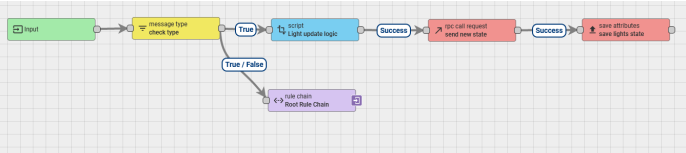


Рисунок 0.16 — Цепочка правил для управления блоком ламп

Узел формирования параметров вентилятора

Полный код, описывающий поведение узла формирования параметров блока умных ламп приведен ниже.

```
function GetRedLightState(msg){
    return msg.redButton == 1 && msg.greenButton == 0;
}

function GetGreenLightState(msg){
    return msg.redButton == 0 && msg.greenButton == 1;
}

let newMsg = {};
let newMsgType = '';

newMsg = {
    "method" : "setLightsState",
    "params":{
        "redLight": GetRedLightState(msg),
        "greenLight": GetGreenLightState(msg)
    }
};

newMsgType = "POST_ATTRIBUTES_REQUEST";

return {msg: msg, metadata: metadata, msgType: msgType};
```

После реализации узла формирования параметров можно приступать к тестированию созданной цепочки.

Подпишемся на топик запросов (v1/devices/me/rpc/request/+) созданного для данного виртуального контроллера. Воспользуемся для этого следующую командой:

```
mosquitto_sub -v -h "demo.thingsboard.io" -t
    "v1/devices/me/rpc/request/+" -u "RxcWZHP60DTvKzcgWB0T"
```

После чего опубликуем сообщение с телеметрией в топик устройства с параметрами `redButton` и `greenButton`:

```
mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t
    "v1/devices/me/telemetry" -u "RxcWZHP60DTvKzcgWB0T" -m
    "{"redButton": 1, "greenButton": 0}"
```

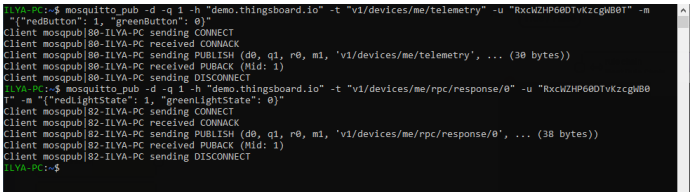


Рисунок 0.17 — Публикация сообщения с телеметрией блока ламп

Чтобы отправить ответ на опубликованный запрос на смену состояния, воспользуемся также утилитой `mosquito_pub`.

Для отправки ответа на созданный запрос необходимо послать сообщение в топик *v1/devices/me/rpc/response/2*. Воспользуемся для этого следующей командой:

```
mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t
    "v1/devices/me/rpc/response/2" -u "RxcWZHP60DTvKzcgWB0T" -m
    "{"redLightState": 1, "greenLightState": 0}"
```

После этого можно проверить в облаке поступившую телеметрию и аргументы устройства, посланные в ответ на запрос. Результаты представлены на Рисунках 0.18 и 0.19.

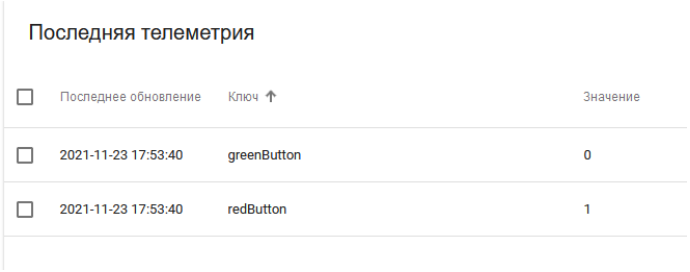


Рисунок 0.18 — Телеметрия облачного устройства «блок светодиодных ламп»

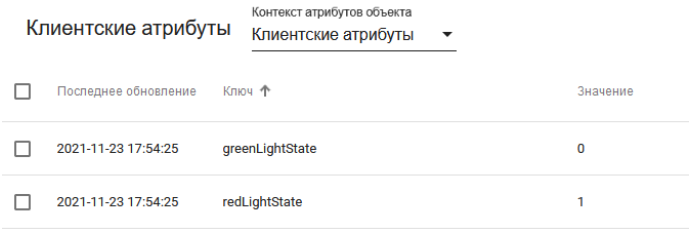


Рисунок 0.19 — Атрибуты облачного устройства «блок светодиодных ламп»

0.4 Дополнительное задание практической работы № 10

Описание взаимодействия пользователя с интерфейсом

Опишите взаимодействие пользователя с интерфейсом вашего решения при помощи графической нотации, к примеру, use-case диаграмм.

0.4.1 Требования к интерфейсу пользователя

Для разрабатываемого нами программного решения был выбран интерфейс telegram – бота. Взаимодействие с данным интерфейсом состоит в отправке текстовых сообщений в мессенджере и основано на принципе «запрос-ответ» — данные предоставляются пользователю по требованию.

Для удобного взаимодействия с информационной системой данные должны предоставляться пользователю в графическом, табличном и текстовом виде. Это упросит взаимодействие с программным инструментом.

Следует реализовать возможность выбора отслеживаемого на данный момент датчика, регистрации нового датчика (прибора), получения сведений о состоянии измерительного прибора, получении текущей телеметрии и статистических данных.

Сведения о передаваемой датчиками телеметрии следует представлять в виде таблиц, графиков или структурированного текста. Также нужно предоставить пользователю возможность получать сведения о данных за определенный временной промежуток, проводить их агрегирование.

0.4.2 Макеты интерфейса приложения

После проведения анализа существующих решений было сформировано представление о том, как следует выстроить взаимодействие с пользователем.

Пользователю предлагается набор команд, которые понимает telegram–бот, их список приведен над полем ввода. На каждую команду бот отвечает установленным образом, предоставляя пользователю информацию о состоянии элементов системы.

Решения использующие аналогичный интерфейс взаимодействия приведены на Рисунках 0.20–0.22.

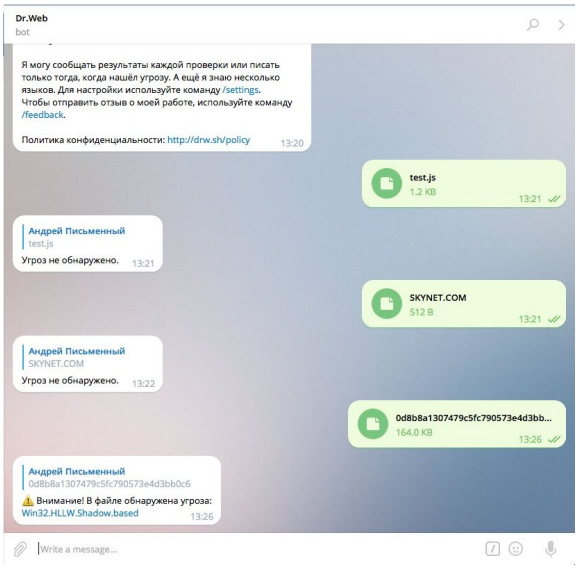


Рисунок 0.20 – Макет 1

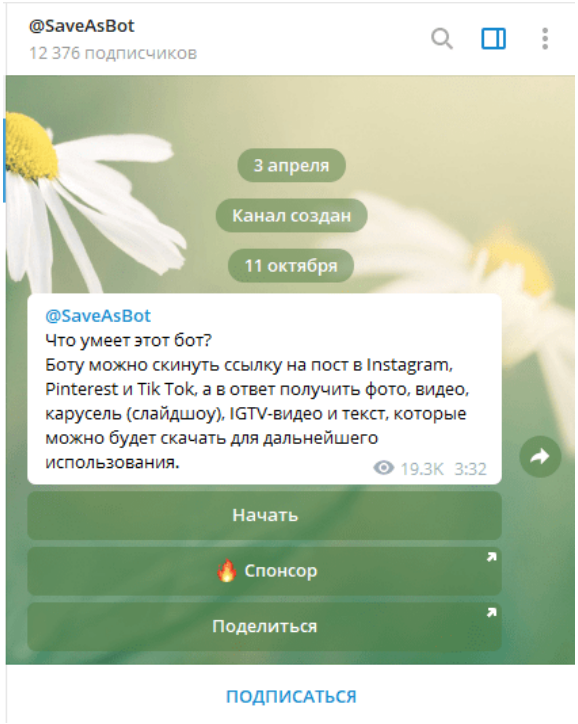


Рисунок 0.21 – Макет 2

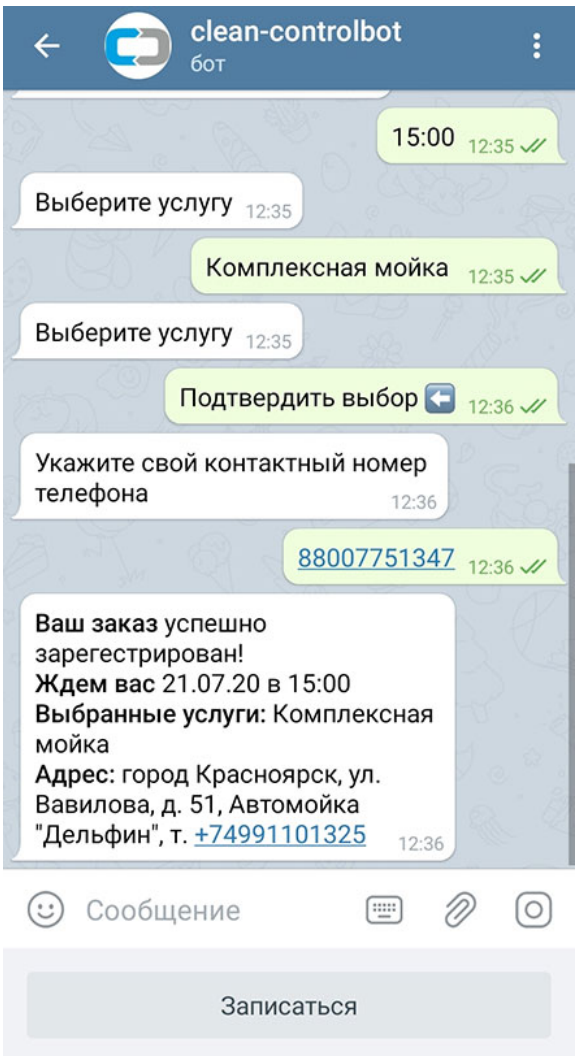


Рисунок 0.22 – Макет 3

0.5 Практическая работа № 11. Реакции платформ Интернета вещей на приходящие данные

0.6 Добавление обработчиков тревожных сигналов

Добавим в цепочки правил управления вентилятором тревогу при выходе приходящего параметра за допустимые границы.

Добавим в цепочки правил управления блоком светодиодов тревогу при отсутствии ожидаемого параметра в приходящем сообщении.

Также для обеих цепочек добавим тревогу при поступлении неверного ответа от физического устройства.