



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»

Выполнили: студенты группы ИВБО-02-19

К. Ю. Денисов
И. А. Кремнев
А. М. Сосунов
Д. Н. Федосеев

Принял: ассистент

Ю. А. Воронцов

Москва 2021

Содержание

1	Практическая работа №9.	
	Знакомство с облачными платформами IoT	3
1.1	Регистрация на платформе ThingsBoard	3
1.2	Создание виртуальных устройств в облаке	3
1.3	Отправка данных в облако	4
2	Дополнительное задание практической работы №9	6
2.1	Выбор облачного решения	6
2.2	Реализация отправки данных	6
3	Практическая работа №10.	
	Управление устройствами при помощи	
	платформ Интернета вещей	9

1 Практическая работа №9.

Знакомство с облачными платформами IoT

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.

1.2 Создание виртуальных устройств в облаке

Создадим в облаке следующие виртуальные устройства для получения данных:

1. Датчик качества воздуха;
2. Датчик освещенности;
3. 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 1,2).

The screenshot shows the 'Добавить новое устройство' (Add New Device) form in the ThingsBoard interface. The form is divided into six steps: 1. Device details, 2. Transport configuration (Optional), 3. Alarm rules (0) (Optional), 4. Device provisioning (Optional), 5. Account details (Optional), and 6. Client (Optional). The first step, 'Device details', is active. It contains the following fields: 'Название *' (Name) with the value 'Air Quality sensor', 'Label', 'Select existing device profile' (radio button), 'Create new device profile' (radio button, selected), 'Device profile name *' with the value 'Air Quality sensor', 'Цепочка правил' (Rule chain), 'Имя для Queue' (Queue name) with the value 'Select from a drop-down list.', 'Гейтвей' (Gateway) (checkbox), and 'Описание' (Description). At the bottom right, there is a 'Next: Transport configuration' button, an 'Отмена' (Cancel) button, and a 'Добавить' (Add) button.

Рисунок 1 — Создание устройства на платформе ThingsBoard

Добавить новое устройство

1
Device details

2
Transport configuration
Optional

3
Alarm rules (0)
Optional

4
Device provisioning
Optional

5
Учетные данные
Optional

6
Клиент
Optional

Transport type *
MQTT
Enables advanced MQTT transport settings

MQTT device topic filters

Telemetry topic filter *
v1/devices/me/telemetry

Attributes topic filter *
v1/devices/me/attributes

Single [+] and multi-level [#] wildcards supported.
[+] is suitable for any topic filter level. Ex: v1/devices/+/telemetry or +/devices/+/attributes.
[#] can replace the topic filter itself and must be the last symbol of the topic. Ex.: # or v1/devices/me/#.

MQTT device payload
JSON

Назад

Next: Alarm rules

Отмена

Добавить

Рисунок 2 — Настройка устройства на платформе ThingsBoard

1.3 Отправка данных в облако

Выполним передачу тестовых данных в каждое из созданных устройств, список которых приведен на рисунке 3.

Устройства	Device profile	
	All	×
<input type="checkbox"/>	Время создания ↓	Название
<input type="checkbox"/>	2021-11-22 09:20:47	Voltage sensor
<input type="checkbox"/>	2021-11-22 09:20:06	Illumination sensor
<input type="checkbox"/>	2021-11-22 09:18:41	Air Quality Sensor
		Device profile
		Voltage sensor
		Illumination sensor
		Air Quality Profile

Рисунок 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 4).

```

ILYA-PC:~$ mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "nS83gBMMghXwPq4yXVw9" -m
{"motion": -800}
Client mosqpub|64-ILYA-PC sending CONNECT
Client mosqpub|64-ILYA-PC received CONNACK
Client mosqpub|64-ILYA-PC sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (14 bytes))
Client mosqpub|64-ILYA-PC received PUBACK (Mid: 1)
Client mosqpub|64-ILYA-PC sending DISCONNECT
ILYA-PC:~$

```

Рисунок 4 — Отправка данных на устройства

После отправки, тестовые данные отображаются в веб-интерфейсе платформы ThingsBoard в виде представленном на рисунках 5–7.

Voltage sensor			
Подробности об устройстве			
Details	Атрибуты	Последняя телеметрия	Оповещения
События			
Отношения			
Последняя телеметрия			
<input type="checkbox"/>	Последнее обновление	Ключ ↑	Значение
<input type="checkbox"/>	2021-11-22 09:42:31	voltage	12.1

Рисунок 5 — Данные с датчика напряжения

Illumination sensor			
Подробности об устройстве			
Details	Атрибуты	Последняя телеметрия	Оповещения
События			
Отношения			
Последняя телеметрия			
<input type="checkbox"/>	Последнее обновление	Ключ ↑	Значение
<input type="checkbox"/>	2021-11-22 09:40:30	illumination	400

Рисунок 6 — Данные с датчика освещения

Air Quality Sensor			
Подробности об устройстве			
Details	Атрибуты	Последняя телеметрия	Оповещения
События			
Отношения			
Последняя телеметрия			
<input type="checkbox"/>	Последнее обновление	Ключ ↑	Значение
<input type="checkbox"/>	2021-11-22 09:37:53	airQuality	20

Рисунок 7 — Данные с датчика качества воздуха

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

2 Дополнительное задание практической работы №9

2.1 Выбор облачного решения

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

- Понятный пользовательский интерфейс;
- Популярность платформы;
- Возможность установить платформу локально или использовать готовую облачную среду.

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:

/home/denilai/Documents/repos/latex/scripts/dop9-publish.py

```
import paho.mqtt.client as paho
2 import sys
import json
4 import schedule
import datetime
6 import time
import random

8
# Connection parameters to the MQTT broker
10 broker="demo.thingsboard.io"
port=1883
12
13 USERNAME = "i1Ohu4cIi2Q7OqUAHf21" # Login to connect to the broker
14
15
16 def job():
    now = datetime.datetime.now()
18    motion = random.randint(20, 800)
    noise = random.randint(20, 800)
20    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"))
```

```

22     data = {
23         "timestamp" : now.isoformat() ,
24         "motion" : motion ,
25         "noise" : noise ,
26         "door_open" : doorState
27     }
28
29     pahoClient.publish("v1/devices/me/telemetry", json.dumps(data))
30
31
32 def main():
33     # Creating and configuring an instance of the Client class to connect to the
34     # MQTT broker
35     global pahoClient
36     pahoClient = paho.Client("controll")
37     pahoClient.username_pw_set(USERNAME)
38     pahoClient.connect(broker, port)
39
40     schedule.every(2).seconds.do(job)
41
42     while 1:
43         schedule.run_pending()
44         time.sleep(1)
45
46 if __name__ == "__main__":
47     main()
48

```

В результате запуска данного скрипта происходит соединение с MQTT-брокером, создание экземпляра класса Client для MQTT-брокера, с последней передачей данных в облако (см. рисунки 8, 9).

```

[10:17 22.11.2021] data: motion: 190, noise: 409, door: Open
[10:17 22.11.2021] data: motion: 103, noise: 418, door: Closed
[10:17 22.11.2021] data: motion: 572, noise: 447, door: Open
[10:17 22.11.2021] data: motion: 633, noise: 440, door: Closed
[10:17 22.11.2021] data: motion: 318, noise: 551, door: Closed
[10:17 22.11.2021] data: motion: 248, noise: 639, door: Open
[10:17 22.11.2021] data: motion: 360, noise: 561, door: Open
[10:17 22.11.2021] data: motion: 761, noise: 30, door: Open
[10:17 22.11.2021] data: motion: 370, noise: 692, door: Open
[10:17 22.11.2021] data: motion: 295, noise: 221, door: Closed
[10:17 22.11.2021] data: motion: 513, noise: 610, door: Open
[10:18 22.11.2021] data: motion: 741, noise: 365, door: Closed
[10:18 22.11.2021] data: motion: 171, noise: 185, door: Closed
[10:18 22.11.2021] data: motion: 134, noise: 458, door: Closed
[10:18 22.11.2021] data: motion: 177, noise: 561, door: Open
[10:18 22.11.2021] data: motion: 391, noise: 378, door: Closed
[10:18 22.11.2021] data: motion: 152, noise: 264, door: Closed
[10:18 22.11.2021] data: motion: 28, noise: 389, door: Open
[10:18 22.11.2021] data: motion: 292, noise: 717, door: Open
[10:18 22.11.2021] data: motion: 363, noise: 625, door: Closed
[10:18 22.11.2021] data: motion: 108, noise: 514, door: Closed
[10:18 22.11.2021] data: motion: 398, noise: 508, door: Open
[10:18 22.11.2021] data: motion: 698, noise: 442, door: Open
[10:18 22.11.2021] data: motion: 67, noise: 35, door: Open
[10:18 22.11.2021] data: motion: 198, noise: 266, door: Open
[10:18 22.11.2021] data: motion: 326, noise: 271, door: Closed
[10:18 22.11.2021] data: motion: 648, noise: 281, door: Open
[10:18 22.11.2021] data: motion: 292, noise: 650, door: Closed
[10:18 22.11.2021] data: motion: 622, noise: 50, door: Closed

```

Рисунок 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 9 — Прием телеметрических данных облачной платформой

3 Практическая работа №10.

Управление устройствами при помощи платформ Интернета вещей

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

1. Включение и выключение вентилятора по датчику движения;
2. Включение и выключения индикации зеленым и красным светом комбинированного датчика по кнопкам.

Реализация сценария управления вентилятором

На платформе ThingsBoard создадим виртуальное устройство, которое будет прообразом реального вентилятора (см. рисунок 10).

Устройства			Device profile	
			All	×
<input type="checkbox"/>	Время создания ↓	Название	Device profile	
<input type="checkbox"/>	2021-11-23 16:57:18	Smart fan	Smart fan	

Рисунок 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. Приведенная на рисунке 11 цепочка правил описывает данный сценарий управления устройством.

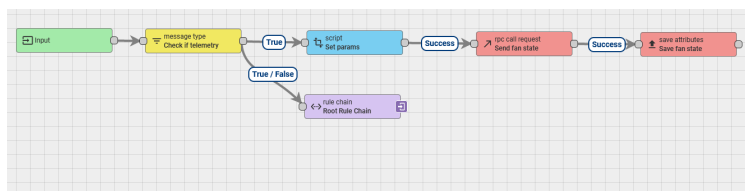


Рисунок 11 — Цепочка правил для управления вентилятором

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 входящей 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_ATTRIBUTES_REQUEST`. Узел возвращает объект, содержащий в себе основное сообщение, метаданные, а также тип сообщения. Полный код приведен ниже.

/home/denilai/Documents/repos/latex/scripts/fan_chain.js

```
function getNewFanState(motion){
2   return motion > 700;
3 }
4
5 let newMsg = {};
6 let newMsgType = '';
7
8 newMsg = {
9   "method" : "setFanState",
10  "params":{
11    "state": getNewFanState(msg.motion)
12  }
13 };
14
15 newMsgType = "POST_ATTRIBUTES_REQUEST";
16
17
18 return {msg: newMsg, metadata: metadata, msgType: newMsgType};
```

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

Подпишемся на топик запросов (`v1/devices/me/rpc/request/+`) созданного для данного виртуального вентилятора. Воспользуемся для этого следующей 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}'
```

```
ILYA-PC:~$ mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "nS83gBMMgHxwPq4yXVW9" -m '{"motion": 800}'
Client mosqpub73-ILYA-PC sending CONNECT
Client mosqpub73-ILYA-PC received CONNACK
Client mosqpub73-ILYA-PC sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (13 bytes))
Client mosqpub73-ILYA-PC received PUBACK (Mid: 1)
Client mosqpub73-ILYA-PC sending DISCONNECT
ILYA-PC:~$ mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/rpc/response/2" -u "nS83gBMMgHxwPq4yXVW9" -m '{"fan_state": 1}'
Client mosqpub74-ILYA-PC sending CONNECT
Client mosqpub74-ILYA-PC received CONNACK
Client mosqpub74-ILYA-PC sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/rpc/response/2', ... (14 bytes))
Client mosqpub74-ILYA-PC received PUBACK (Mid: 1)
Client mosqpub74-ILYA-PC sending DISCONNECT
ILYA-PC:~$
```

Рисунок 12 — Публикация сообщения с телеметрией вентилятора

Чтобы отправить ответ на опубликованный запрос на смену состояния, воспользуемся также утилитой `mosquitto_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 13 и 14.


Последняя телеметрия			
<input type="checkbox"/>	Последнее обновление	Ключ 	Значение
<input type="checkbox"/>	2021-11-23 17:28:18	motion	800

Рисунок 13 — Телеметрия облачного устройства «вентилятор»

Клиентские атрибуты			Контекст атрибутов объекта
			Клиентские атрибуты
<input type="checkbox"/>	Последнее обновление	Ключ ↑	Значение
<input type="checkbox"/>	2021-11-23 17:28:30	fan_state	1

Рисунок 14 — Атрибуты облачного устройства «вентилятор»

Реализация сценария управления лампами

На платформе ThingsBoard создадим виртуальное устройство, которое будет прообразом реальных блока светодиодных ламп (см. рисунок 15).

Устройства		Device profile	
		All	×
<input type="checkbox"/>	Время создания ↓	Название	Device profile
<input type="checkbox"/>	2021-11-23 17:41:02	Smart lamp	Smart lamp

Рисунок 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 16 цепочка правил описывает данный сценарий управления устройством.

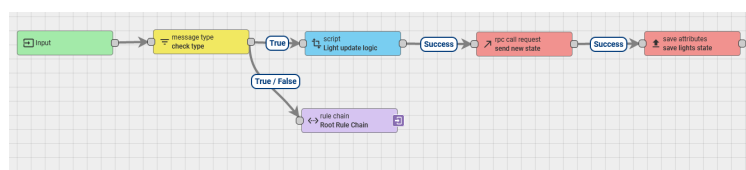


Рисунок 16 — Цепочка правил для управления блоком ламп

Узел формирования параметров вентилятора

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

/home/denilai/Documents/repos/latex/scripts/light_chain.js

```

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

```

```

function GetGreenLightState(msg){
6   return msg.redButton == 0 && msg.greenButton == 1;
}
8
let newMsg = {};
10 let newMsgType = '';

12 newMsg = {
    "method" : "setLightsState",
14    "params":{
        "redLight": GetRedLightState(msg) ,
16        "greenLight": GetGreenLightState(msg)
    }
18 };

20 newMsgType = "POST_ATTRIBUTES_REQUEST";

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

```

После реализации узла формирования параметров можно приступить 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"

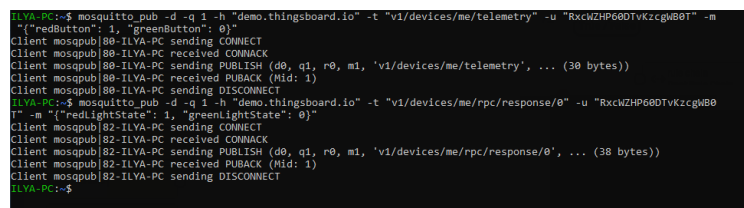
```

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}'

```



```

ILYA-PC:~$ mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -t "v1/devices/me/telemetry" -u "RxcWZHP60DTvKzcgWB0T" -m
{"redButton": 1, "greenButton": 0}
Client mosqpub[80-ILYA-PC] sending CONNECT
Client mosqsub[80-ILYA-PC] received CONNACK
Client mosqpub[80-ILYA-PC] sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (38 bytes))
Client mosqpub[80-ILYA-PC] received PUBACK (Mid: 1)
Client mosqpub[80-ILYA-PC] sending DISCONNECT
ILYA-PC:~$ mosquitto_sub -v -h "demo.thingsboard.io" -t "v1/devices/me/rpc/request/+" -u "RxcWZHP60DTvKzcgWB0T"
Client mosqsub[82-ILYA-PC] sending CONNECT
Client mosqsub[82-ILYA-PC] received CONNACK
Client mosqsub[82-ILYA-PC] sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/rpc/request/+', ... (38 bytes))
Client mosqsub[82-ILYA-PC] received PUBACK (Mid: 1)
Client mosqsub[82-ILYA-PC] sending DISCONNECT
ILYA-PC:~$

```

Рисунок 17 — Публикация сообщения с телеметрией блока ламп

Чтобы отправить ответ на опубликованный запрос на смену состояния, воспользуемся также утилитой `mosquitto_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 18 и 19.


Последняя телеметрия			
<input type="checkbox"/>	Последнее обновление	Ключ 	Значение
<input type="checkbox"/>	2021-11-23 17:53:40	greenButton	0
<input type="checkbox"/>	2021-11-23 17:53:40	redButton	1

Рисунок 18 — Телеметрия облачного устройства «блок светодиодных ламп»



Клиентские атрибуты			
		Контекст атрибутов объекта	
		Клиентские атрибуты 	
<input type="checkbox"/>	Последнее обновление	Ключ 	Значение
<input type="checkbox"/>	2021-11-23 17:54:25	greenLightState	0
<input type="checkbox"/>	2021-11-23 17:54:25	redLightState	1

Рисунок 19 — Атрибуты облачного устройства «блок светодиодных ламп»