

## 5.2. Введение в IBM Cloud

Site: [Samsung Innovation Campus](#)  
Course: Internet of Things  
Book: 5.2. Введение в IBM Cloud

Printed by: Антон Файтельсон  
Date: Saturday, 21 October 2023, 7:39 PM

Table of contents

- 5.2.1. Регистрация и создание виртуального устройства
- 5.2.2. Коммуникация с устройством
- 5.2.3. Отправка данных с платы микроконтроллера

### 5.2.1. Регистрация и создание виртуального устройства

Чтобы сделать какое-то свое приложение, давайте зарегистрируемся в IBM Cloud. Это бесплатно. Если от вас будут просить данные банковской карточки, можно не соглашаться: нам достаточно будет lite-аккаунта. Это верно по состоянию на конец 2021 года; учтите, что политика IBM периодически меняется. Если данные руководства становится невозможно выполнить из-за необходимости ввести номер карточки, то можно пройти аналогичные руководства по ThingsBoard или иной бесплатной облачной платформе.

После того, как вы залогинились, сразу же переходите сюда: <https://cloud.ibm.com/catalog>

Это - каталог доступных приложений. Ищем в нем сразу Internet of Things:

The screenshot shows the IBM Cloud catalog search results for 'internet of things'. The search bar at the top contains 'internet of things'. Below the search bar, there are several service cards. The first card is 'Internet of Things Platform' by IBM, described as 'This service is the hub of all things IBM IoT, it is where you can set up and manage your connected devices so that your apps can...'. It is labeled 'Lite' and 'Free'. Other cards include 'AT&T Flow Designer', 'AT&T IoT Data Plans', 'Bosch IoT Rollouts', 'Car Diagnostic API', 'Portworx Enterprise', 'Precision Location', and 'UnificationEngine'. Each card provides a brief description and a 'Free' label.

И внутри нажмите сразу кнопку Create. Это позволит вам подключать устройства Интернета вещей - не более 500 штук, не более 200 Мб памяти на каждое, что для учебных целей более чем достаточно.

Появится такая картинка. Нажимаем кнопку Launch!

The screenshot shows a landing page titled 'Let's get started with IBM Watson IoT Platform'. It features a large blue icon of a device with a central circle and four connecting lines. To the right of the icon, the text reads: 'Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world.' Below this text are two buttons: 'Launch' (in blue) and 'Docs' (in light gray).

Появится экран вашей панели управления устройствами Интернета вещей:

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Boards', 'Devices', 'Members', 'Apps', 'Access Management', 'Usage', 'Security', and 'Settings'. The 'Devices' section is active, showing a 'Diagnose' button. Below this, there is a summary of all devices that have been added. A table with columns 'Device ID', 'Status', 'Device Type', and 'Class ID' is shown, but it is empty. A message states 'You don't have any devices.' with a 'Create a device.' button. The bottom of the dashboard has a 'Device Simulator' toggle and a filter icon.

Адрес у этой панели будет наподобие <https://pm78kw.internetofthings.ibmcloud.com/>, то есть такой немного странный URL со случайными буквами и цифрами. Фактически - это название вашей организации в терминах IBM Cloud. Поскольку вы не создавали никакой организации (такая возможность тоже есть), то у вас получилось случайное название.

Мы сразу находимся на вкладке Devices, так что давайте сразу создадим какое-нибудь устройство. Нажмите Create a device. Однако, от вас сразу же потребуют указать тип устройства и его идентификатор:

Add Device

Identity

Device Information

Security

Summary

Select a device type for the device that you are adding and give the device a unique ID.

Device Type

Select or create a device type...

Device ID

Enter Device ID

У нас пока нет никаких типов устройств, поэтому идем во вкладку Device Types. Там придумайте какой-нибудь свой тип устройства, к примеру, STM32Board. В реальной жизни это идентификатор модели устройства. В следующем окне вам предложат ввести метаданные (версия прошивки, производитель и прочее) - здесь можно ничего не заполнять.

Add Type

Identity

Device Information

Device types group devices that have similar characteristics, such as model number, firmware version, or location. Give the device type a type.

Type

Device

Or

Gateway

Name

STM32Board

The device type name is used to identify the device type uniquely and uses a restricted set of characters to make it suitable for API use.

Description

Итак, после того, как добавили тип, появляется возможность зарегистрировать новое устройство:

You added the new device type: STM32Board

Register Device

Advanced Flow

Optional

Register Devices, Define Interfaces

Now that you added a device type, you can register and connect devices for this type.

Register Devices

Получаться будет примерно так:

Add Device

Identity

Device Information

Select a device type for the device that you are adding and give the device a unique ID.

Device Type

STM32Board

Device ID

my\_test\_board

Поля с детальной информацией об устройстве пока можно опять пропустить. А вот дальше интересное: вам предложат сгенерировать аутентификационный ключ:

Add Device

Identity

Device Information

Security

There are two options for selecting a device authentication token.

Auto-generated authentication token (default)

Allow the service to generate an authentication token for you. Tokens are 18 characters and contain a mix of alphanumeric characters and symbols. The token is returned to you at the end of the device registration process.

Self-provided authentication token

Provide your own authentication token for this device. The token must be between 8 and 36 characters and contain a mix of lowercase and uppercase letters, numbers, and symbols, which can include hyphens, underscores, and periods. Do not use repeated characters, dictionary words, user names, or other predefined sequences.

Authentication Token

Enter an optional token

Make a note of the generated token. Lost authentication tokens cannot be recovered. Tokens are encrypted before being stored.

Authentication token are encrypted before we store them.

Рекомендуется ничего не сочинять и взять автоматически сгенерированный ключ. Получится примерно так:

Device Credentials	
You registered your device to the organization. Add these credentials to the device to the device is connected, you can navigate to view connection and event details.	
Organization ID	pm78kw
Device Type	STM32Board
Device ID	my_test_board
Authentication Method	use-token-auth
Authentication Token	6bKRIMjq1L(xBW)p-1

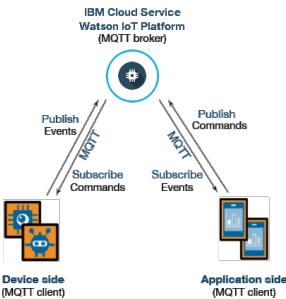
Сохраните себе этот ключ (в данном примере 6bKRIMjq1L(xBW)p-1) в надежное место. Потом, после закрытия сеанса браузера с этой страницей, вы больше никогда не увидите этого ключа.

Теперь ваше устройство видно в общем списке устройств и вы можете смотреть последние события и статус.

5.2.2. Коммуникация с устройством

Сейчас попробуем устроить сеанс коммуникации с нашим виртуальным устройством, которое мы только что создали. Общение будет происходить по MQTT.

То, как все это устроено, наглядно показано на схеме:



Параметры сервера:

- Адрес - `org_id.messaging.internetofthings.ibmcloud.com`. В моем случае это будет `utr1r3.messaging.internetofthings.ibmcloud.com`
- Порты:
  - 1883 (отключен по умолчанию)
  - 8883 (с шифрованием) - мы будем использовать именно этот
  - 443 (для вебсокетов)

Параметры для авторизации:

- Имя пользователя - `use-token-auth`
- Пароль - тот самый ключ, который вы получили выше
- Client ID - в формате `<d>:<orgId>:<deviceType>:<deviceId>`. Где начальное `d` означает "устройство", а `g` - шлюз (gateway).

Примерно так это будет выглядеть в [MQTTX](#). Используйте эту или другую программу. MQTTLens не годится - там не поддерживается TLS.:

Connections

New Collection

< Back

New

Connect

General

\* Name

IBM Watson (Device)

ⓘ

\* Client ID

d:utr1r3:STM32Board:my\_test\_device

ⓘ ⓘ

\* Host

mqtt://

utr1r3.messaging.internetofthings.ibmcloud.com

\* Port

8883

ⓘ

Username

use-token-auth

Password

.....

SSL/TLS

☒ true

☐ false

\* Certificate

☒ CA signed server

☐ Self signed

SSL Secure

☒

ⓘ

Параметр SSL/TLS отметьте true. Здесь важно, что используется протокол защиты TLS, чтобы пароль не передавался в открытом виде.

Если при подключении будет ошибка, то вы можете ее посмотреть в интерфейсе IBM Cloud, на страничке вашего устройства, в разделе Logs. К примеру, вот как будет выглядеть попытка авторизации, если вы указали неправильные параметры (к примеру, не указали соединение по TLS):

Device Drilldown - my\_test\_board

Connection Information

Recent Events

State

Device Information

Metadata

Diagnostics

Connection Logs

Device Actions

Connection Logs

View logs for the device connection to Watson IoT Platform

Message	Timestamp	⌵
Closed connection. The operation is not authorized.	12 июля 2020 г., 14:56	
Closed connection. The operation is not authorized.	12 июля 2020 г., 14:56	
Closed connection. The operation is not authorized.	12 июля 2020 г., 14:56	
Closed connection. The operation is not authorized.	12 июля 2020 г., 14:56	
Closed connection. The operation is not authorized.	12 июля 2020 г., 14:56	
Closed connection. The operation is not authorized.	12 июля 2020 г., 14:56	
Closed connection. The operation is not authorized.	12 июля 2020 г., 14:56	
Closed connection. The operation is not authorized.	12 июля 2020 г., 14:56	
Closed connection. The operation is not authorized.	12 июля 2020 г., 14:56	
Closed connection. The operation is not authorized.	12 июля 2020 г., 14:56	

Там же можно посмотреть на ошибки в названиях топиков, к примеру.

Если все данные ввели правильно, то увидите статус на вкладке "Устройства":

<input type="checkbox"/>	Device ID	Status	Device Type
>	<input type="checkbox"/> my_test_board	<span>Connected</span>	STM32Board

Форматы топиков и сообщений

Формат названия топика для сообщений от устройства будет такой:

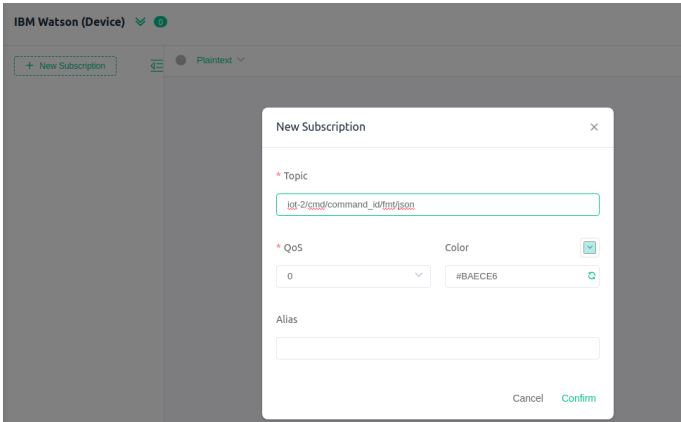
```
iot-2/evt/event_id/fmt/format_string
```

Где event\_id – ваш собственный идентификатор события (придумайте любой), а format\_string говорит о том, в каком формате следует декодировать сообщение (XML, JSON или любой другой). То есть такое сообщение будет о том, что на устройстве произошло какое-то событие, и устройство хочет об этом сигнализировать, публикуя сообщение в топик.

А формат названия топика для команд устройству - такой:

```
iot-2/cmd/command_id/fmt/format_string
```

Всё аналогично. Это для тех топиков, из которых устройство хочет получать команды, и соответственно, подписывается на них. Подпишитесь для пробы:



### Пример отправки сообщения от устройства в облако

Отправим сообщение от устройства. Имя топика будет такое:

```
iot-2/evt/my_event/fmt/json
```

А сообщение в формате JSON такое: {"key1": "5"}

Увидите, что всё сработало, и данные появились в системе.

Device ID

Status

Device Type

Class ID

Date Added

Descriptive Location

Added By

Device Class

▼

my\_test\_board

Connected

STM32Board

Device

18 мая 2020 г., 16:59

tatyana.a.volkova@gmail.com

→

...

Identity

Device Information

Recent Events

State

Logs

✕

Showing Raw Data | No Interfaces Available

Property

Value

Type

Event

Last Received

key1

5

String

my\_event

несколько секунд назад

Items per page 50

▼

1–1 of 1 item

1 of 1 page

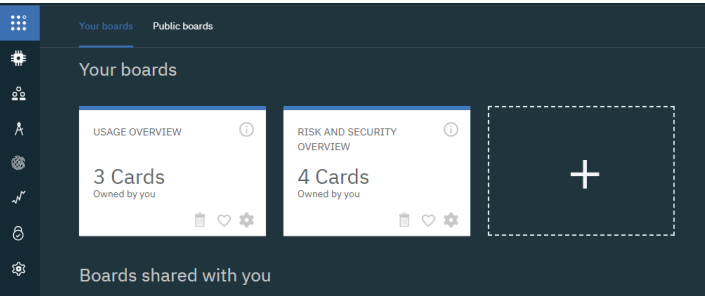
<

1

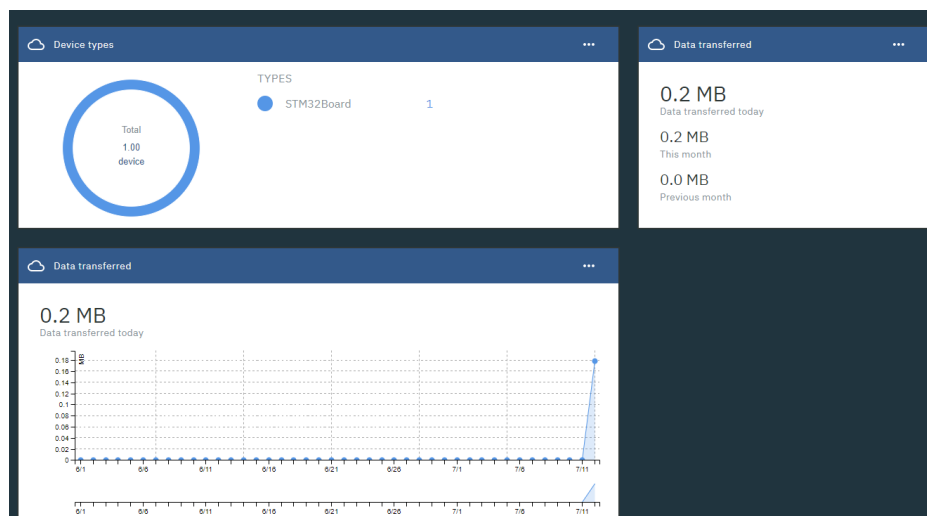
▼

>

Добавим график. Переходим на главный экран:

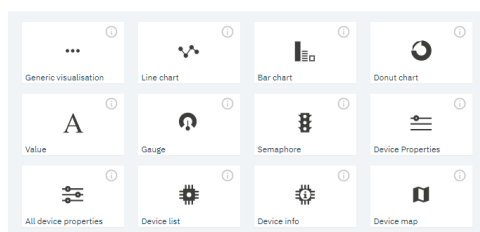


Нажав на Usage Overview, увидите статистику использования устройства и трафика:

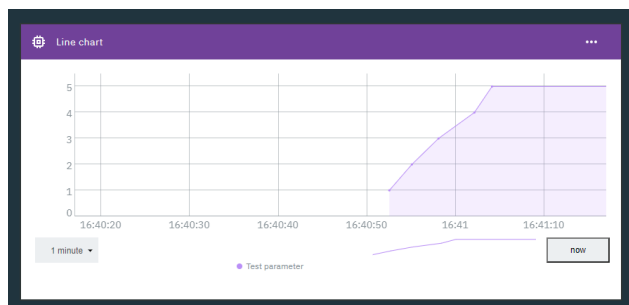


А нажав Add New Card, можно добавить график именно с данными:

#### Devices



Выглядеть он будет так - для примера, когда мы передали сообщение с параметром key1 равным от 1 до 5, как в предыдущем примере:





9 of 11

10/21/23, 16:39

```
message.qos = MQTT::QOS0;
message.retained = false;
message.dup = false;
message.payload = (void *)buf;
message.payloadlen = strlen(buf);
rc = client.publish(topic, message);
printf("Message sent\n");
printf("Demo concluded successfully \n\n");

return 0;
}
```

Результат выполнения этого кода будет следующий:

```
Starting Starting IBM MQTT demo:
Connecting to the network...
Opening connection to remote 158.175.111.155 port 1883
Connected socket
Sending message:
{"d":{"ST":"Nucleo-IoT-mbed","Temp":1,"Pressure":2,"Humidity":3}}
Message sent
Demo concluded successfully
```

my\_test\_board

Disconnected

STM32Board

Device

Identity

Device Information

Recent Events

State

Logs

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
statusEvent	{"d":{"ST":"Nucleo-IoT-mbed","Temp":1,"Pressur...	json	a few seconds ago

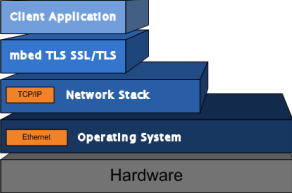
Теперь вы знаете, как формировать сообщение и отправлять его. Как вы видите, пример вообще почти ничем не отличается от примера с MQTT, единственные тонкости - указать `client_id` и топик, и правильно сформировать тело сообщения.

Для тех, у кого Mbed 5-й версии.

В этих руководствах мы рекомендуем использовать Mbed версии 6. Но может случиться так, что у вас окажется плата, которая поддерживает максимум пятую версию. Что делать в этом случае?

Всё почти так же, но не совсем. Понадобится библиотека `mbedits`, которая по умолчанию включена в Mbed шестой версии, а вот в пятой придется повозиться. Ее можно импортировать отсюда: <https://github.com/ARMmbed/mbedits>, через стандартный механизм добавления библиотек в проект Mbed.

Какое место в стеке используемых технологий занимает эта библиотека, исчерпывающе показано на этой картинке:



То есть это добавочный уровень над TCP/IP, который мы уже протестировали в предыдущих лабораторных.

```
#include "MQTTmbed.h"
#include <MQTTClientMbedOs.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("Starting IBM MQTT demo\n");

    TCPSocket socket;
    NetworkInterface *net = NetworkInterface::get_default_instance();
    if (!net) {
        printf("Error! No network interface found.\n");
        return 0;
    }
    printf("Connecting to the network...\n");

    nsapi_size_or_error_t rc = net->connect();
    if (rc != 0) {
        printf("Error! _net->connect() returned: %d\n", rc);
        return -1;
    }

    rc = socket.open(net);
    if (rc != 0) {
        printf("Error! _socket.open() returned: %d\n", rc);
        return -1;
    }

    SocketAddress address;
    net->gethostname("utrir3.messaging.internetofthings.ibmcloud.com",
        &address);
    address.set_port(1883);

    printf("Opening connection to remote %s port %d\n",
        address.get_ip_address(), address.get_port());
    rc = socket.connect(address);
    if (rc != 0) {
        printf("Error! _socket.connect() returned: %d\n", rc);
        return -1;
    }
    printf("Connected socket\n");

    MQTTClient client(&socket);
    MQTTPacket_connectData data = MQTTPacket_connectData_initializer;
    data.MQTTVersion = 3;
    data.clientID.cstring = "d:utrir3-STM32Board:my_test_board";
    data.username.cstring = "use-token-auth";
    data.password.cstring = "c">Msg@9ts0Kk70@";
    if (rc = client.connect(data) != 0)
        printf("rc from MQTT connect is %d\n", rc);

    MQTT::Message message;
    char *topic = "iot-2/evt/statusEvent/fmt/json";
    char buf[100];
    int temp = 1;
    int press = 2;
    int hum = 3;
    sprintf(buf,
        "{\"d\":{\"ST\":\"Nucleo-IoT-mbed\",\"Temp\":%d,\"Pressure\":%d,\"Humidity\":%d}}",
        temp, press, hum);
    printf("Sending message: %s\n", buf);
    message.qos = MQTT::QOS0;
    message.retained = false;
```

```
message.dup = false;
message.payload = (void *)buf;
message.payloadlen = strlen(buf);
rc = client.publish(topic, message);
printf("Message sent\n");
printf("Demo concluded successfully\n");

return 0;
}
```

[Reset user tour on this page](#)