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

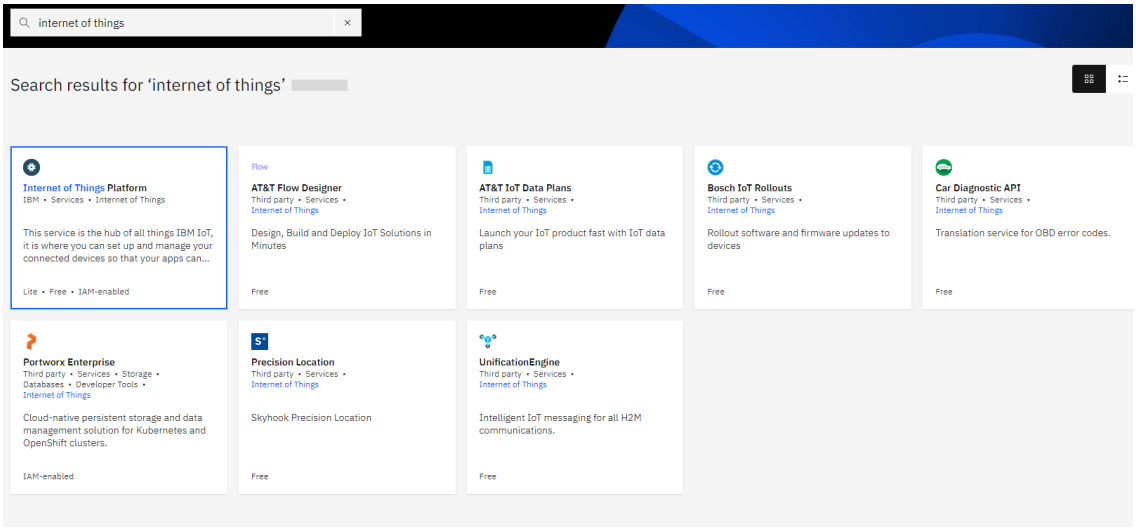
Table of contents

5.2.1. Регистрация и создание виртуального устройства

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.

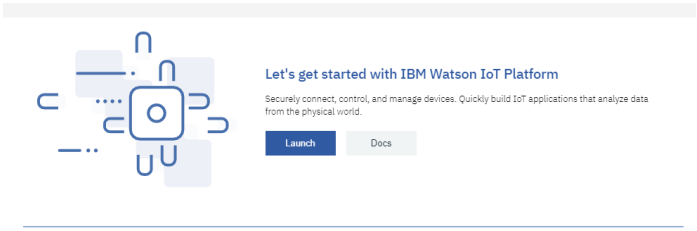
После того, как вы залогинились, сразу же переходите сюда: https://cloud.ibm.com/catalog

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

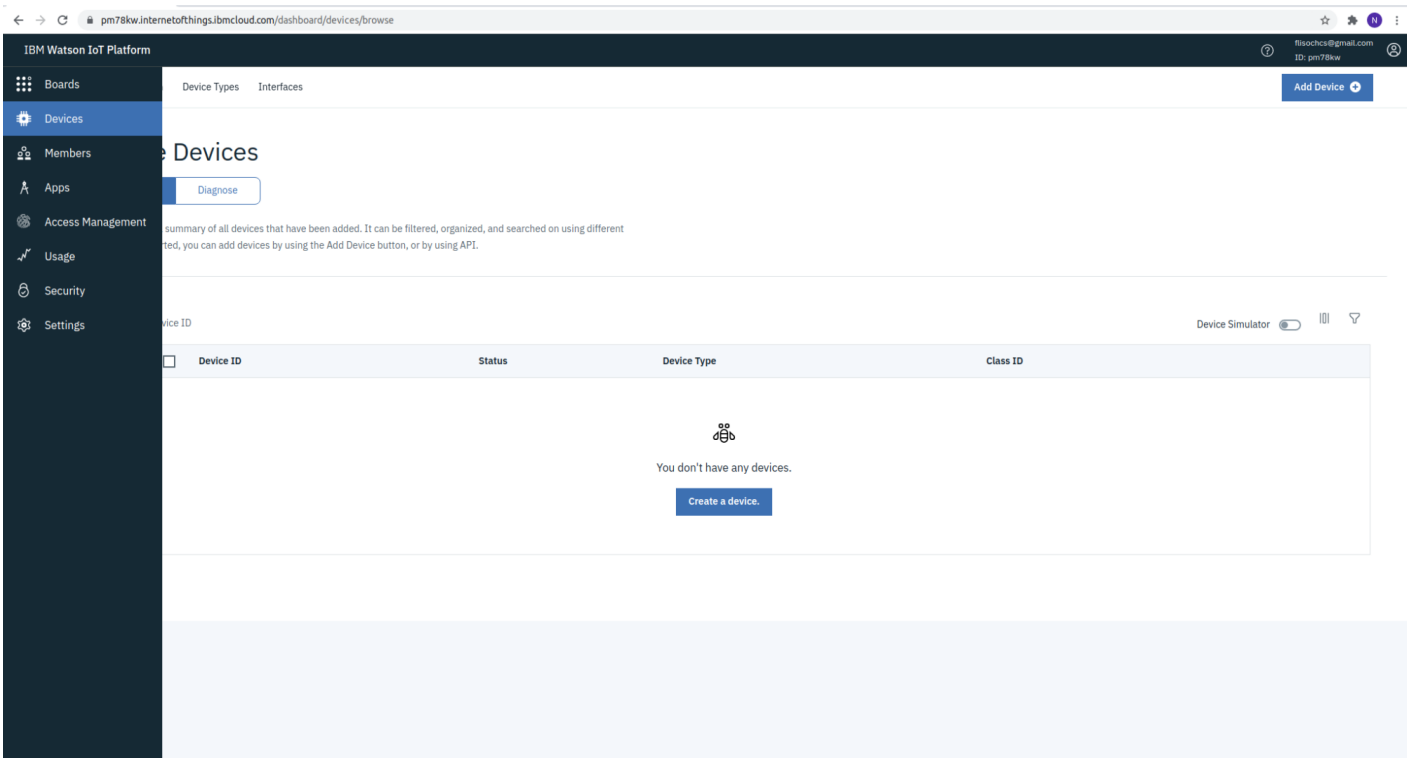


И внутри нажмите сразу кнопку Create. Это позволит вам подключать устройства Интернета вещей - не более 500 штук, не более 200 Мб памяти на каждое, что для учебных целей более чем достаточно.

Появится такая картинка. Нажимаем кнопку Launch!

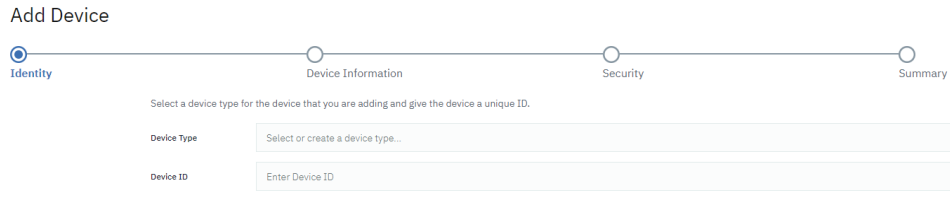


Появится экран вашей панели управления устройствами Интернета вещей:

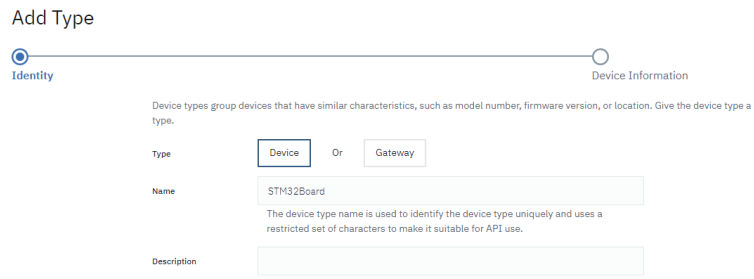


Адрес у этой панели будет наподобие https://pm78kw.internetofthings.ibmcloud.com/, то есть такой немного странный URL со случайными буквами и цифрами. Фактически - это название вашей организации в терминах IBM Cloud. Поскольку вы не создавали никакой организации (такая возможность тоже есть), то у вас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

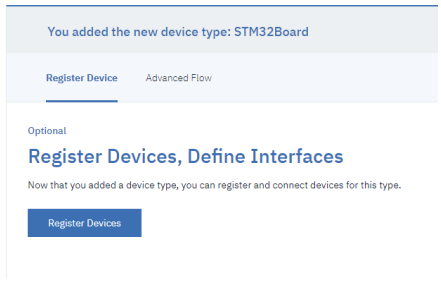
Мы сразу находимся на вкладке Devices, так что давайте сразу создадим какое-нибудь устройство. Нажмите Create a device. Однако, от вас сразу же потребуют указать тип устройства и его идентификатор:



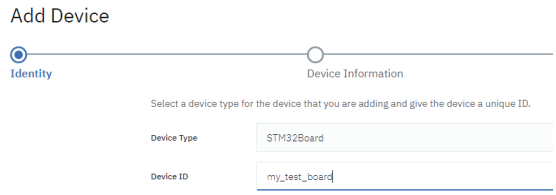
У нас пока нет никаких типов устройств, поэтому идем во вкладку Device Types. Там придумайте какой-нибудь свой тип устройства, к примеру, STM32Board. В реальной жизни это идентификатор модели устройства. В следующем окне вам предложат ввести метаданные (версия прошивки, производитель и прочее) - здесь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



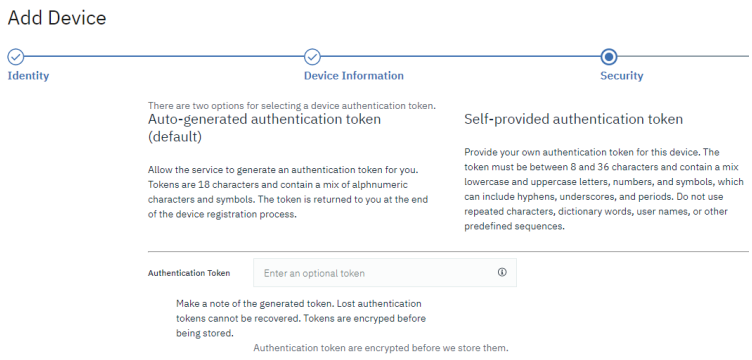
Итак, после того, как добавили тип, появляется возможность зарегистрировать новое устройство:



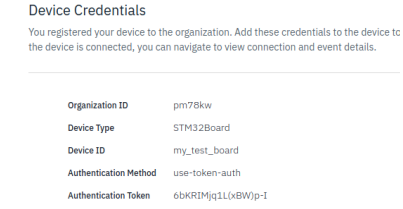
Получаться будет примерно так:



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



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



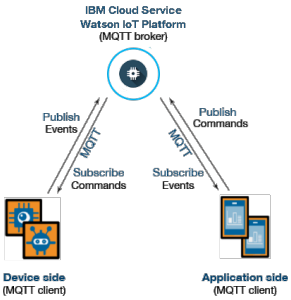
Сохраните себе этот ключ (в данном примере 6bKRIMjq1L(xBW)p-I) в надежное место. Потом, после закрытия сеанса браузера с этой страницей, вы больше никогда не увидите этого ключа.

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:



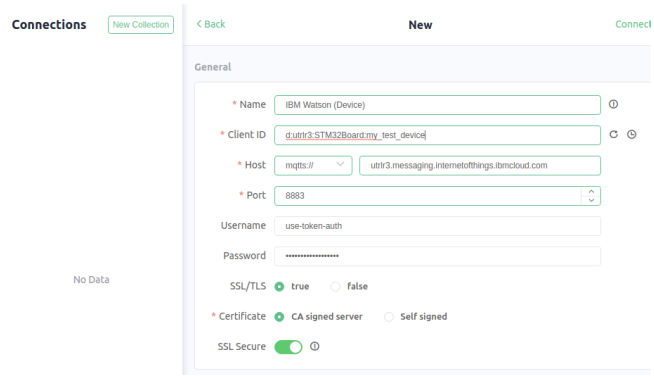
Параметры сервера:

- Адрес - org_id.messaging.internetofthings.ibmcloud.com. В моем случае это будет utrlr3.messaging.internetofthings.ibmcloud.com
- Порты:
  - 1883 (отключен по умолчанию)
  - 8883 (с шифрованием) - мы будем использовать именно этот
  - 443 (для вебсокетов)

Параметры для авторизации:

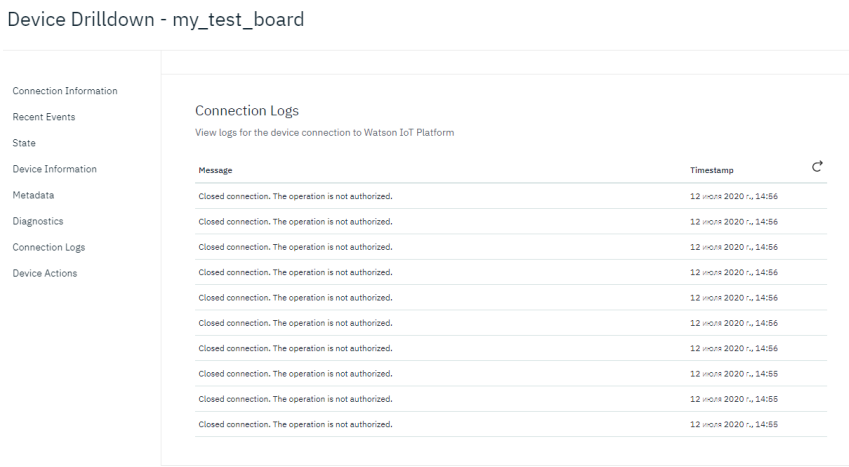
- Имя пользователя - use-token-auth
- Пароль - тот самый ключ, который вы получили выше
- Client ID - в формате d:<orgId>:<deviceType>:<deviceId>. Где начальное d означает "устройство", а g - шлюз (gateway).

Примерно так это будет выглядеть в MQTT X. Используйте эту или другую программу. MQTTLens не годится - там не поддерживается TLS. :



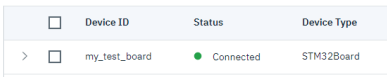
Параметр SSL/TLS отметьте true. Здесь важно, что используется протокол защиты TLS, чтобы пароль не передавался в открытом виде.

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):



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

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



Форматы топиков и сообщений

Формат названия топика для сообщений от устройства будет такой:

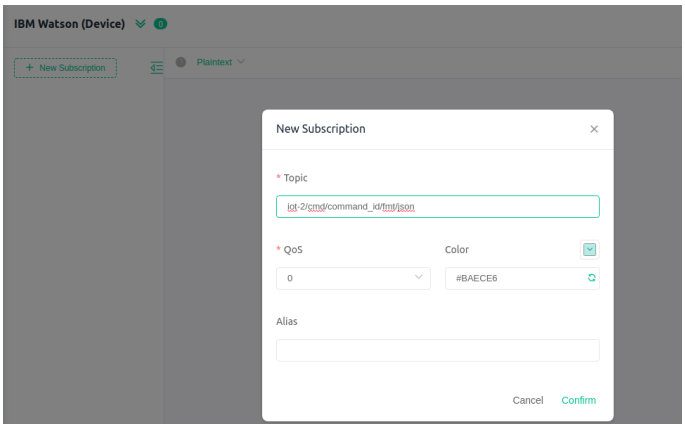
```
iot-2/evt/event_id/fmt/format_string
```

Где event_id - ваш собственный идентификатор события (придумайте любой), а format_string говорит о том, в каком формате следует раскодировать 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:



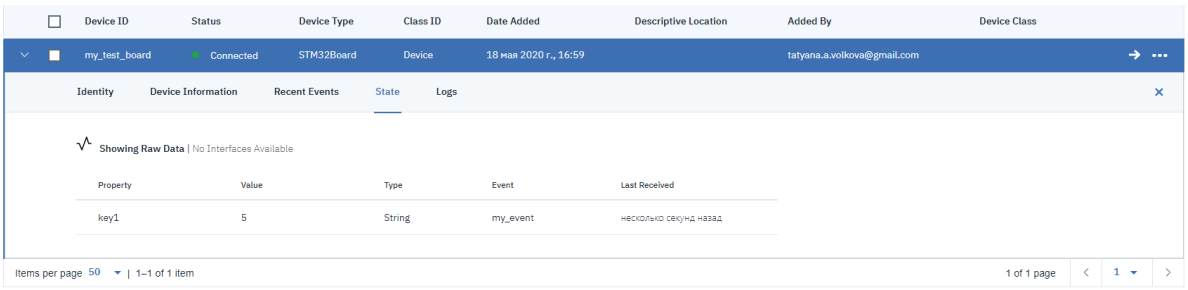
Пример отправки сообщения от устройства в облако

Отправим сообщение от устройства. Имя топика будет такое:

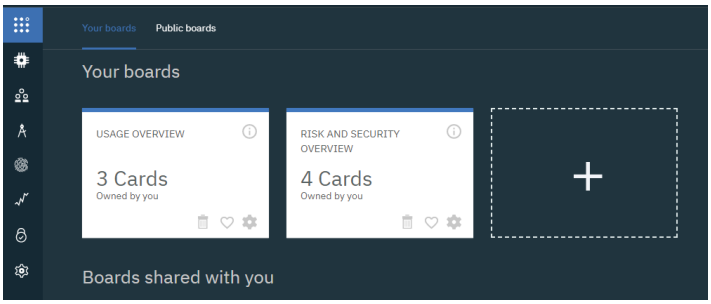
```
iot-2/evt/my_event/fmt/json
```

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

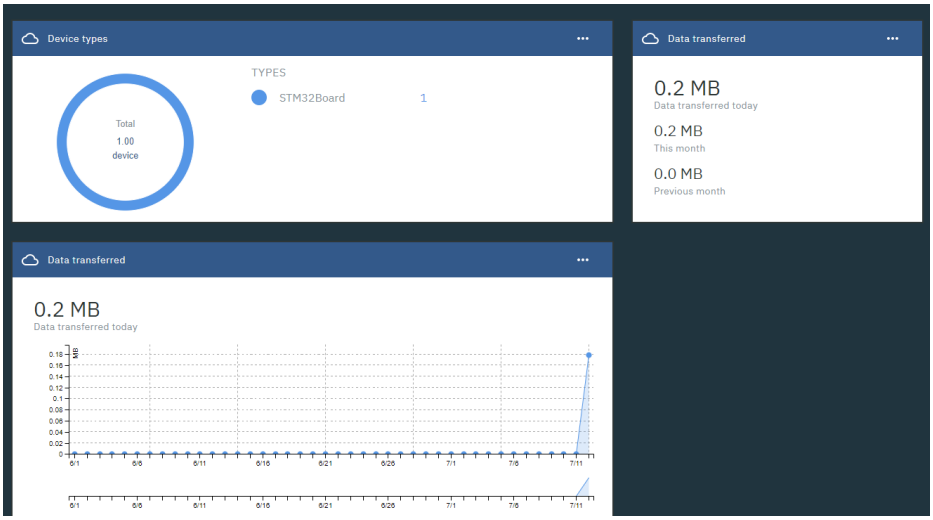
Увидите, что всё сработало, и данные появились в системе.



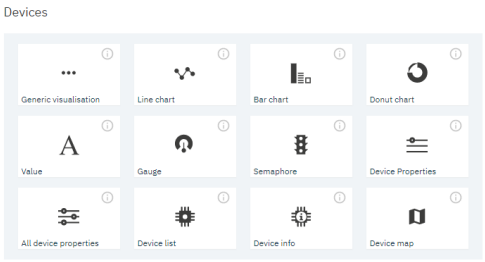
Добавим график. Переходим на главный экран:



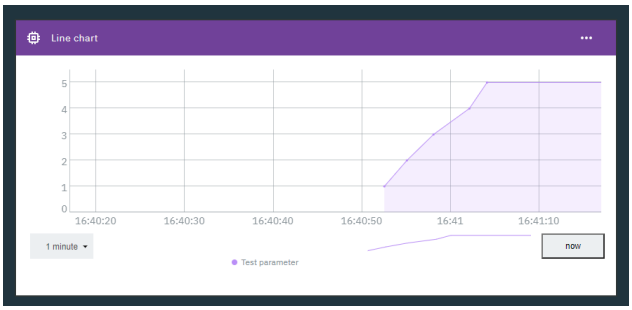
Нажав на Usage Overview, увидите статистику использования устройства и трафика:



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



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



5.2.3. Отправка данных с платы микроконтроллера

Теперь, конечно, сразу интересно, а можно ли отправлять какие-то свои данные туда, от своего устройства, а не симулированного. Ответ: да. Причем опять нам не нужно нигде регистрироваться и создавать аккаунт, всё взаимодействие по-прежнему будет осуществляться через уникальный deviceID.

IBM Cloud поддерживает коммуникацию через MQTT (предпочтительно) или через HTTP. Давайте попробуем сделать через MQTT.

Нам понадобится:

- Плата микроконтроллера (показывать будем на примере STM32 Nucleo, желательно, чтобы плата поддерживала Mbed версии 6, если это не так - идите в конец этого руководства, там есть ремарка для более ранней версии)
- Соединение платы с Интернетом (показывать будем на примере WiFi)

Из программных требований, нужно, чтобы для вашего устройства были библиотеки для подключения к Интернету и для работы с MQTT. То есть уже должны быть выполнены соответствующие лабораторные работы.

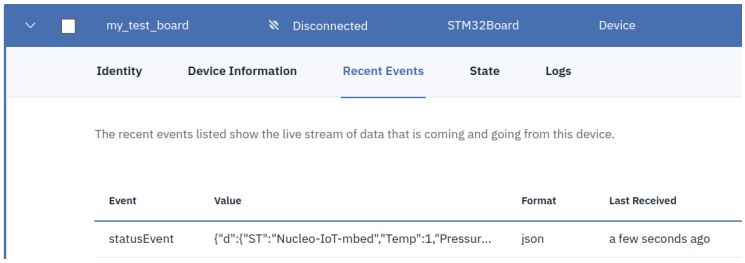
Вот демо-код для подключения к Платформе интернета вещей IBM. Здесь отправляется сообщение с четырьмя параметрами - один с названием платы, и три имитируют показания различных сенсоров. Для целей демонстрации здесь стоят просто константы, но ничего не мешает вам подставить здесь значения реальных сенсоров.

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

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 inteface found.\n");
   return 0;
 }
 printf("Connecting to the network...\r\n");

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

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

 SocketAddress address;
 net->gethostbyname("utrlr3.messaging.internetofthings.ibmcloud.com",
                    &address);
 address.set_port(1883);

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

 MQTTClient client(&socket);
 MQTTPacket_connectData data = MQTTPacket_connectData_initializer;
 data.MQTTVersion = 3;
 data.clientID.cstring = "d:utrlr3:STM32Board:my_test_board";
 data.username.cstring = "use-token-auth";
 data.password.cstring = "c)-rMerJ@9Ix0Kk?0@";
 if ((rc = client.connect(data)) != 0)
   printf("rc from MQTT connect is %d\r\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: \n%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 \r\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
```



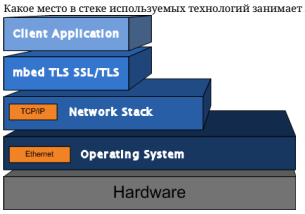
Теперь вы знаете, как формировать сообщение и отправлять его. Как вы видите, пример вообще почти ничем не отличается от примера с MQTT, единственные тонкости - указать **client_id** и топик, и правильно сформировать тело сообщения.

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 mbedtls, которая по умолчанию включена в Mbed шестой версии, а вот в пятой придется повозиться. Ее можно импортировать отсюда: https://github.com/ARMmbed/mbedtls, через стандартный механизм добавления библиотеки 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 <cstdio>

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 inteface found.\n");
  return 0;
 }
 printf("Connecting to the network...\r\n");

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

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

 SocketAddress address;
 net->gethostbyname("utrlr3.messaging.internetofthings.ibmcloud.com",
                    &address);
 address.set_port(1883);

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

 MQTTClient client(&socket);
 MQTTPacket_connectData data = MQTTPacket_connectData_initializer;
 data.MQTTVersion = 3;
 data.clientID.cstring = "d:utrlr3:STM32Board:my_test_board";
 data.username.cstring = "use-token-auth";
 data.password.cstring = "c)-rMerJ@9Ix0Kk?0@";
 if ((rc = client.connect(data)) != 0)
  printf("rc from MQTT connect is %d\r\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: \n%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 \r\n");

    return 0;
}
```

Reset user tour on this page