



Geekbrains

Разработка системы сбора данных для “Умного дома”

Программа:
Специализация
Бармин Егор Александрович

Город
Год

Содержание

Глава 1. Теоретическая часть (~15 стр.)

1.1 VMware workstation 17.

1.2 Raspberry pi

1.3 MQTT broker

1.4 Node-red

1.5 InfluxDB

1.6 Клиент-издатель

1.7 Клиент-подписчик

Глава 2. Практическая часть(~15 стр)

2.1 Установка MQTT брокера

2.2 Установка Node-red

2.3 Установка InfluxDB

2.4 Настройка: MQTT брокера, Node-red и InfluxDB

2.5 Приложения: Издатель и подписчик

Глава 3. Тестирование(~20 стр.)

3.1 Тестирование связности MQTT брокера, Node-red, InfluxDB

3.2 Тестирование клиента-издателя

3.3 Тестирование клиента-подписчика

3.4 Общее тестирование всего проекта

Заключение (~ 4 стр.)

Список используемой литературы

Приложения

Введение

Тема проекта: Разработка системы сбора данных для “умного дома”

Какую проблему решает: Сбор обработка и визуализация данных

Задачи:

1. Изучить литературу, касающуюся темы проекта.
2. Создать среду сбора, обработки и визуализации данных.
3. Создать клиента издателя.
4. Создать клиента подписчика.
5. Проверить работу программ.

Инструменты: VMware workstation 17, VScode, Eclipse Mosquitto, Node-red, InfluxDB, Git, Geeny.

Глава 1 Теоретическая часть.

1.1 VMware workstation 17.

VMware workstation - это система виртуализации.

Она позволяет создавать на обычном компьютере виртуальную систему со собственными ресурсами, на которой можно установить любую операционную систему.

VMWare Workstation часто используется для тестирования программного обеспечения и создания виртуальной среды для работы с программами, последствия работы которых нежелательны для основной операционной системы.

1.2 Raspberry Pi

Raspberry Pi - Raspberry Pi — это миниатюрный одноплатный компьютер, который поместится на ладони взрослого человека.

Несмотря на свои скромные размеры, плата имеет высокую производительность, что позволяет ей выйти на один уровень со стационарными ПК.

Основная отличительная черта Raspberry Pi от обычных компьютеров — наличие программируемых портов ввода-вывода GPIO. С помощью них можно управлять различными устройствами и принимать телеметрию с датчиков.

1.3 MQTT broker

MQTT broker - упрощенный сетевой протокол, работающий поверх TCP/IP, ориентированный на обмен сообщениями между устройствами по принципу «издатель — подписчик».

1.4 Node-red

Node-red - предоставляет веб-редактор потоков, который можно использовать для создания функций JavaScript

1.5 InfluxDB

InfluxDB - система управления баз данных для хранения временных рядов

Основной фокус — хранение больших объёмов данных с метками времени (таких как данные мониторинга, метрики приложений и показания датчиков) и их обработка в условиях высокой нагрузки на запись.

1.6 Клиент-издатель

Консольная программа ,которая будет эмулировать работу датчика температуры.

Она будет считывать с текстового файла значения температуры и передавать в произвольный топик в mqtt брокер.

1.7 Клиент-подписчик

Консольная программа ,которая будет подписываться на топики, считывать значения температуры и будет вести запись в log файл.

Глава 2 Практическая часть

2.1 Установка MQTT брокера

Переходим на официальный сайт Mosquitto и смотрим документацию на установку mqtt брокера на Raspberry Pi

Для добавления ключа подписи, в командной строке вводим:

```
<wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key>
```

```
<sudo apt-key add mosquitto-repo.gpg.key>
```

Перейдем по пути чтобы сделать репозиторий доступным для apt:

```
<cd /etc/apt/sources.list.d/>
```

Скачиваем нужный репозиторий:

```
<sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list>
```

Обновляем информацию apt:

```
<apt-get update>
```

Устанавливаем MQTT брокер:

```
<apt-get install mosquitto>
```

Проверим MQTT:

<mosquitto>

Если все успешно получим:

```
stat@raspberrypi: ~  
File Edit Tabs Help  
stat@raspberrypi:~ $ mosquitto  
1722633099: mosquitto version 2.0.11 starting  
1722633099: Using default config.  
1722633099: Starting in local only mode. Connections will only be possible from clients running on this machine.  
1722633099: Create a configuration file which defines a listener to allow remote access.  
1722633099: For more details see https://mosquitto.org/documentation/authentication-methods/  
1722633099: Opening ipv4 listen socket on port 1883.  
1722633099: Error: Address already in use  
1722633099: Opening ipv6 listen socket on port 1883.  
1722633099: Error: Address already in use  
stat@raspberrypi:~ $
```


2.2 Установка Node-red

Переходим на официальный сайт Node-red и смотрим документацию на установку его в Raspberry Pi

В командной строке вводим для запуска скрипта установки:

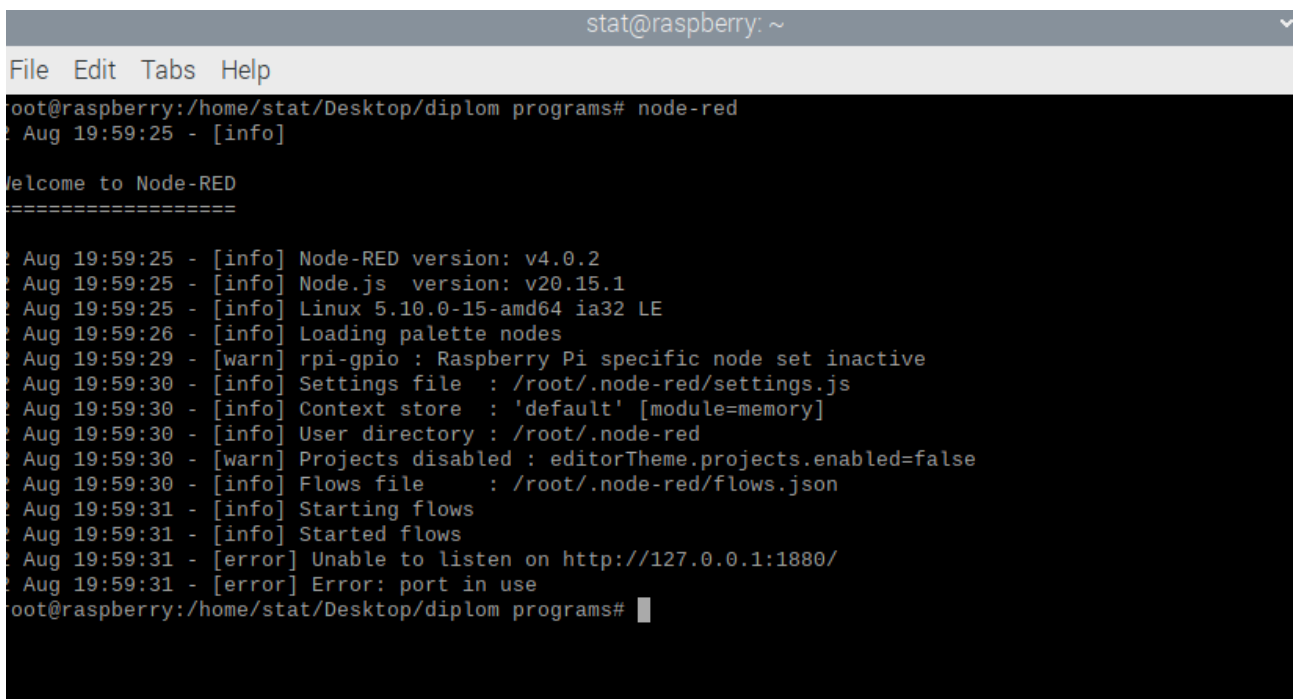
```
<bash<(curl-sL  
https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)>
```

в этом скрипте производится установка Node-red и его конфигурация.

Проверим Node-red:

```
<node-red>
```

Если все успешно получим:



```
stat@raspberrypi: ~  
File Edit Tabs Help  
root@raspberrypi:/home/stat/Desktop/diplom programs# node-red  
Aug 19:59:25 - [info]  
Welcome to Node-RED  
=====  
Aug 19:59:25 - [info] Node-RED version: v4.0.2  
Aug 19:59:25 - [info] Node.js version: v20.15.1  
Aug 19:59:25 - [info] Linux 5.10.0-15-amd64 ia32 LE  
Aug 19:59:26 - [info] Loading palette nodes  
Aug 19:59:29 - [warn] rpi-gpio : Raspberry Pi specific node set inactive  
Aug 19:59:30 - [info] Settings file : /root/.node-red/settings.js  
Aug 19:59:30 - [info] Context store : 'default' [module=memory]  
Aug 19:59:30 - [info] User directory : /root/.node-red  
Aug 19:59:30 - [warn] Projects disabled : editorTheme.projects.enabled=false  
Aug 19:59:30 - [info] Flows file : /root/.node-red/flows.json  
Aug 19:59:31 - [info] Starting flows  
Aug 19:59:31 - [info] Started flows  
Aug 19:59:31 - [error] Unable to listen on http://127.0.0.1:1880/  
Aug 19:59:31 - [error] Error: port in use  
root@raspberrypi:/home/stat/Desktop/diplom programs#
```

Включаем автозапуск:

```
<sudo systemctl enable nodered.service>
```

2.3 Установка InfluxDB

Переходим на официальный сайт InfluxDB и смотрим документацию на установку его в Raspberry Pi

Скачаем пакет:

```
<curl -LO  
https://download.influxdata.com/influxdb/releases/influxdb2_2.7.8-1_amd64.deb>
```

Установим пакет:

```
<sudo dpkg -i influxdb2_2.7.8-1_amd64.deb>
```

И запустим сервис:

```
<sudo service influxdb start>
```

2.4 Настройка: MQTT брокера, Node-red и InfluxDB

Настройка MQTT брокера:

Перейдем в директорию:

```
<cd /etc/mosquitto/conf.d>
```

Создадим конфигурационный файл:

```
<nano default.conf>
```

Заполним его:

```
<allow_anonymous false  
password_file /etc/mosquitto/passwd  
listener 1883>
```

В этом файле запрещаем анонимные подключения, без логина и пароля, указываем файл в котором будет логин и пароль для подключения к брокеру и разрешаем подключение вне loopback.

Создадим файл логина пароля по указанному пути в конфиге mosquitto:

<nano passwd>

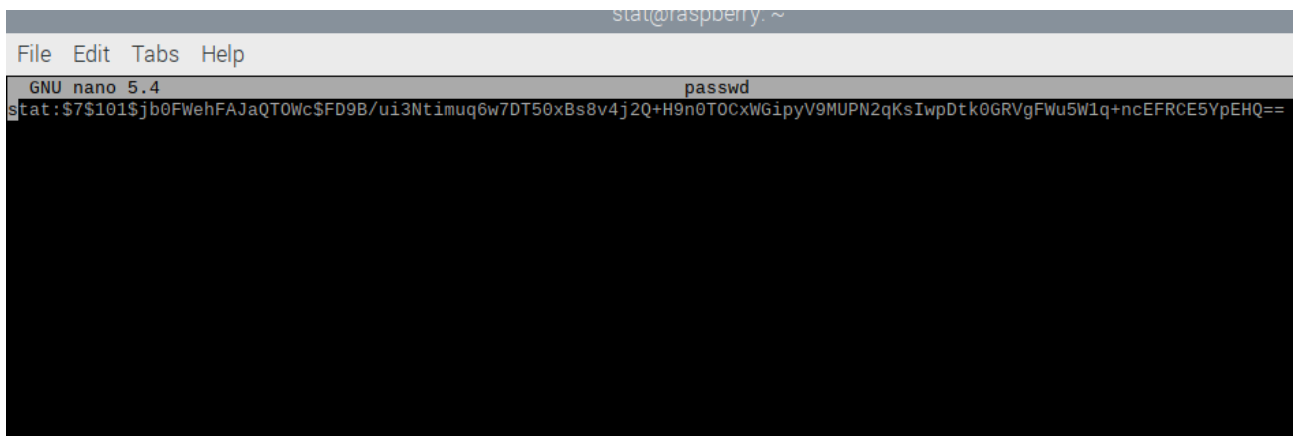
Заполним его:

<login:password>

Далее зашифруем пароль:

<mosquitto_passwd -U passwd>

И проверяем пароль:



```
stat@raspberrypi: ~  
File Edit Tabs Help  
GNU nano 5.4 passwd  
stat:$7$101$jb0FWehFAJaQT0wc$FD9B/ui3Nti3muq6w7DT50xBs8v4j2Q+H9n0T0CxWGipyV9MUPN2qKsIwpDt k0GRVgFWu5W1q+ncEFRCE5YpEHQ==
```

Настройка InfluxDB:

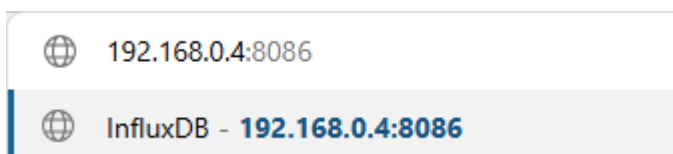
В командной строке вводим:

<ip a>

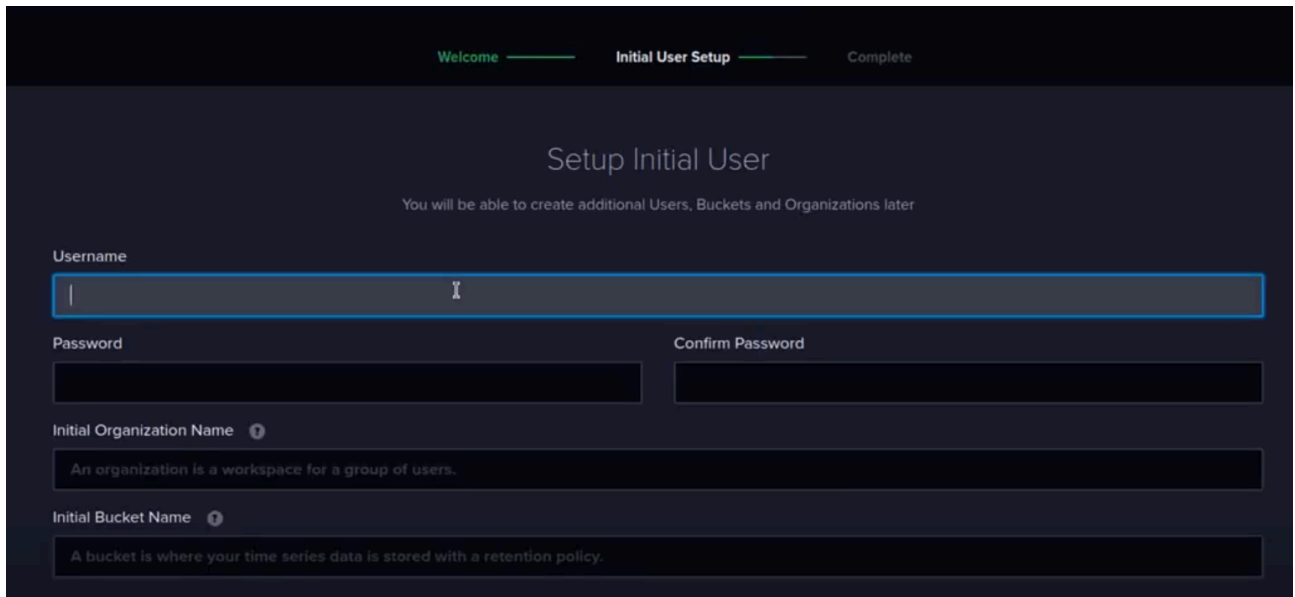
и смотрим наш ip адрес:

```
stat@raspberrypi: ~  
File Edit Tabs Help  
root@raspberrypi:/etc/mosquitto# ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000  
    link/ether 00:0c:29:9c:d7:8e brd ff:ff:ff:ff:ff:ff  
    inet 192.168.0.4/24 brd 192.168.0.255 scope global dynamic noprefixroute eth0  
        valid_lft 229953sec preferred_lft 197553sec  
    inet6 fe80::eb45:5b50:270c:4f1c/64 scope link  
        valid_lft forever preferred_lft forever  
root@raspberrypi:/etc/mosquitto#
```

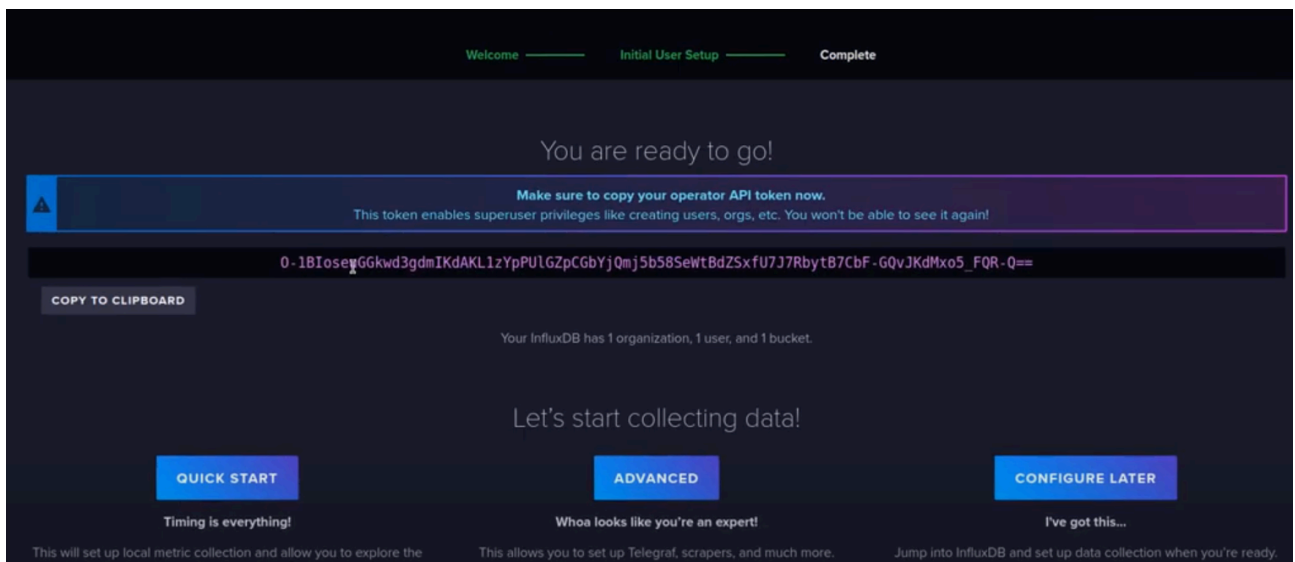
вводим адрес в строку браузера и указываем порт:



Попадаем на начальную настройку конфигурации
вводим логин, пароль, название организации и bucket:



сохраняем токен:



Настройка Node-red:

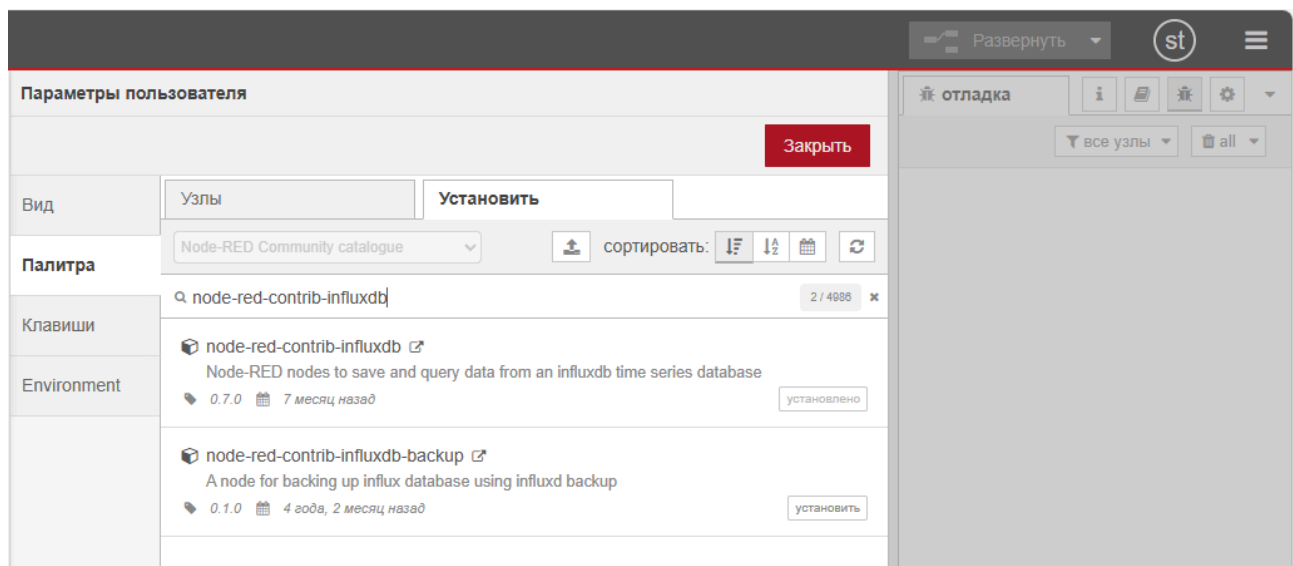
Вводим адрес в строку браузера и указываем порт:



Вводим логин пароль и попадаем в рабочую зону Node-red.

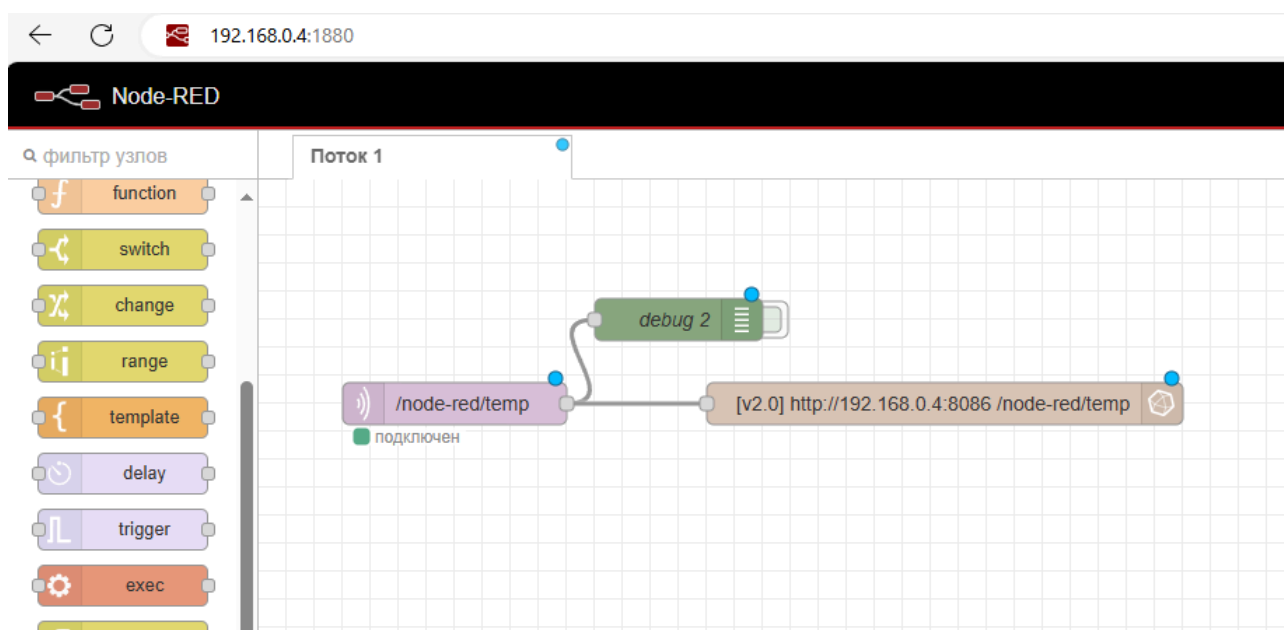
Для дальнейшей работы с InfluxDB потребуется установить плагин в Node-red, переходим в управление палитрой->установить и в поиске(screen):

<node-red-contrib-influxdb>



Теперь настроим ноды в Node-red:

Добавим ноды: mqtt in и influxdb out:



Настройка mqtt in:

в строке сервер добавляем новое подключение,

далее в строке сервер вводим ip адрес виртуальной машины и порт MQTT брокера:

Изменить узел mqtt in > Изменить узел mqtt-broker

Удалить

Отмена

Обновить

Свойства

Имя

Имя

Соединение

Безопасность

Сообщения

Сервер

192.168.0.4

Порт

1883

☒ Connect automatically

☐ TLS

Protocol

MQTT V5

ID клиента

Оставьте пустым для автоматически сгенерированного

Keep-alive время (сек)

60

Session

☒ Use clean start

Session Expiry (secs)

User Properties

none

в окне безопасность вводим логин пароль от MQTT брокера:

Изменить узел mqtt in > **Изменить узел mqtt-broker**

УдалитьОтменаОбновить

⚙ **Свойства**

⚙

🔑 **Имя**

Имя

Соединение

Безопасность

Сообщения

👤 **Имя польз.**

stat

🔒 **Пароль**

.....

В строке Тема вводим топик который хотим прослушивать с MQTT брокера.

В строке QoS выбираем 1:

Изменить узел mqtt in

Удалить

Отмена

Готово

Свойства

Сервер

192.168.0.4:1883

+

Action

Subscribe to single topic

Тема

/node-red/temp

QoS

1

Flags

☐ Do not receive messages published by this client

☒ Keep retain flag of original publish

Retained message handling

Send retained messages

Выход

автоопределение (разобрать объект JSON, строка ил

Имя

Имя

Настройка influxdb out:

в строке сервер добавляем новое подключение, далее в строке Version выбираем 2.0

в строке URL вводим адрес виртуальной машины с портом InfluxDB.

в строке Token вводим токен полученный в InfluxDB:

Изменить узел influxdb out > Изменить узел influxdb

Удалить Отмена Обновить

⚙ Свойства

🏷 Имя

Имя

📄 Version

2.0 ▼

📄 URL

http://192.168.0.4:8086

🔒 Token

.....

🕒 Connection timeout (seconds)

10

☒ Verify server certificate

Далее после добавления нового подключения,
в строке Organization вводим организацию из InfluxDB
в строке Bucket вводим bucket из InfluxDB
в строке Measurement вводим произвольное название передаваемого измерения:

Изменить узел influxdb out

Удалить

Отмена

Готово

⚙ Свойства

⚙

📄

🔗

🏷 Имя

Имя

📄 Server

[v2.0] http://192.168.0.4:8086

▼

✎

+

👤 Organization

stat_org

📄 Bucket

stat_bucket

📡 Measurement

/node-red/temp

🕒 Time Precision

Milliseconds (ms)

▼

2.5 Приложения: Издатель и подписчик

Для работы программ издателя и подписчика потребуется библиотека `paho.mqtt.c`, также для работы библиотеки надо установить библиотеку `OpenSSL`

`OpenSSL`:

```
<sudo apt-get install libssl-dev>
```

`Paho.mqtt.c`:

```
<git clone https://github.com/eclipse/paho.mqtt.c>
```

Переходим в библиотеку:

```
<make>
```

И устанавливаем библиотеку:

```
<sudo make install>
```

Издатель:

Подключаем библиотеки и прописываем define:

```
C sub.c  C pubsub_opts.h  C pub.c  X
C pub.c > sensor > hour
1  #include <stdio.h>
2  #include <stdint.h>
3  #include <time.h>
4  #include <stdlib.h>
5  #include <string.h>
6  #include "MQTTClient.h"
7
8  #define CLIENTID    "sensor"
9  #define PAYLOAD     "USEFULL_DATA"
10 #define QOS         1
11 #define TIMEOUT     10000L
12 #define DELAY       5
13
```

создаем структуру sensor и функцию addrecord с помощью которых из файла с температурой, запишутся данные в массив:

```
C sub.c  C pubsub_opts.h  C pub.c  X  Release Notes: 1.92.0  MakeFile
C pub.c > sensor > hour
11 #define TIMEOUT     10000L
12 #define DELAY       5
13
14 struct sensor
15 {
16     uint16_t  year;
17     uint8_t   month;
18     uint16_t  day;
19     uint8_t   hour;
20     uint8_t   minute;
21     int8_t    temp;
22 };
23
24 void addrecord(struct sensor* info, int number, uint16_t year, uint8_t month, uint16_t day, uint8_t hour, uint8_t minute, int8_t temp)
25 {
26     info[number].year  = year;
27     info[number].month = month;
28     info[number].day   = day;
29     info[number].hour  = hour;
30     info[number].minute = minute;
31     info[number].temp  = temp;
32 }
33
```

объявим переменные:

```
int main()
{
    int max = 20;
    int rc;
    int Y,M,D,H,MIN,T;
    int r;
    int count      = 0;
    int i          = 0;
    char str_filename[max];
    char str_address [max];
    char str_user    [max];
    char str_password[max];
    char str_topic   [max];
    const char *address;
    const char *username;
    const char *password;
    const char *topic;
```

rc = код возврата

Y,M,D,H,MIN,T = переменные времени и температуры

r = количество считанных из файла данных

count и i = переменные счетчики

5 массивов типа char = служат для записи login параметров

4 const char = служат для записи login параметров

Здесь вводим из консоли логин,пароль,адрес, топик и название файла из которого будет считана температура:

```
printf("Input login:");           //login
scanf("%s", str_user);

printf("Input password:");       //password
scanf("%s", str_password);

printf("Input host + port:");    //host
scanf("%s", str_address);

printf("input topic:");         //topic
scanf("%s", str_topic);

printf("Input filename:");      //filename
scanf("%s", str_filename);

username = str_user;
address  = str_address;
password = str_password;
topic    = str_topic;
```

Инициализируем настройки подключения и создаем клиента:

```
74 MQTTClient client;
75 MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
76 MQTTClient_message pubmsg = MQTTClient_message_initializer;
77 MQTTClient_deliveryToken token;
78
79 MQTTClient_create(&client, address, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
80 conn_opts.keepAliveInterval = 20;
81 conn_opts.cleansession      = 1;
82 conn_opts.username          = username;
83 conn_opts.password           = password;
```

Проверяем удалось ли подключиться:

```
85     if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS)
86     {
87         printf("Failed to connect, return code %d\n", rc);
88         exit(-1);
89     }
```

выделяем память структуре, открываем файл и проверяем на ошибки данных в текстовом файле:

```
91     struct sensor*info = malloc(365*24*60*sizeof(struct sensor));
92     FILE *file;
93     file = fopen(str_filename,"r");
94
95
96     for (;(r=fscanf(file,"%d;%d;%d;%d;%d;%d",&Y,&M,&D,&H,&MIN,&T))>0;count++)
97     {
98         if (r<6)
99         {
100             char s[20], c;
101             r = fscanf (file, "%[^\\n]%c",s,&c);
102             printf("Wrong format in line %s\n",s);
103         }
104         else
105         {
106             printf("%d %d %d %d %d %d\n",Y,M,D,H,MIN,T);
107             addrecord(info,count,Y,M,D,H,MIN,T);
108         }
109     }
110     fclose(file);
```

Далее в цикле считываются данные из массивов и публикуются в MQTT брокер:

```
112     while(1)
113     {
114         clock_t begin = clock();
115
116         char str[255];
117         sprintf(str,"%d",info[i++].temp);
118         printf("%s,%d\n",str,i);
119
120         if(i>=count)
121             i=0;
122
123
124         pubmsg.payload = str;
125         pubmsg.payloadlen = strlen(str);
126         pubmsg.qos = QOS;
127         pubmsg.retained = 0;
128         MQTTClient_publishMessage(client, topic, &pubmsg, &token);
129         rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
130
131         while ((double)(clock() - begin)/CLOCKS_PER_SEC<DELAY)
132             {}
133     }
134
```

Подписчик:

Подключаем библиотеки:

```
19
20 #include <MQTTClient.h>
21 #include <MQTTClientPersistence.h>
22 #include "pubsub_opts.h"
23 #include <unistd.h>
24 #include <sys/time.h>
25 #include <time.h>
26
27
28 #include <stdio.h>
29 #include <signal.h>
30 #include <string.h>
31 #include <stdlib.h>
```

Подключаем структуру из библиотеки:

```
37 struct pubsub_opts opts =
38 {
39     0, 0, 0, 0, "\n", 100, /* debug/app options */
40     NULL, NULL, 1, 0, 0, /* message options */
41     MQTTVERSION_DEFAULT, "topic", "paho-cs-sub", 0, 0, "login", "password", "address_mqtt", "port", NULL, 10, /* MQTT options */
42     NULL, NULL, 0, 0, /* will options */
43     0, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, /* TLS options */
44     0, {NULL, NULL}, /* MQTT V5 options */
45     NULL, NULL, /* HTTP and HTTPS proxies */
46 };
```

Функция myconnect инициализирует настройки подключения:

```
48 int myconnect(MQTTClient client)
49 {
50     MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
51     MQTTClient_SSLOptions ssl_opts = MQTTClient_SSLOptions_initializer;
52     MQTTClient_willOptions will_opts = MQTTClient_willOptions_initializer;
53     int rc = 0;
54
55     if (opts.verbose)
56         printf("Connecting\n");
57
58     if (opts.MQTTVersion == MQTTVERSION_5)
59     {
60         MQTTClient_connectOptions conn_opts5 = MQTTClient_connectOptions_initializer5;
61         conn_opts = conn_opts5;
62     }
63     conn_opts.keepAliveInterval = opts.keepalive;
64     conn_opts.username = opts.username;
65     conn_opts.password = opts.password;
66     conn_opts.MQTTVersion = opts.MQTTVersion;
67     conn_opts.httpProxy = opts.http_proxy;
68     conn_opts.httpsProxy = opts.https_proxy;
```

```
70     if (opts.will_topic) /* will options */
71     {
72         will_opts.message = opts.will_payload;
73         will_opts.topicName = opts.will_topic;
74         will_opts.qos = opts.will_qos;
75         will_opts.retained = opts.will_retain;
76         conn_opts.will = &will_opts;
77     }
78
79     if (opts.connection && (strcmp(opts.connection, "ssl://", 6) == 0 ||
80         strcmp(opts.connection, "wss://", 6) == 0))
81     {
82         if (opts.insecure)
83             ssl_opts.verify = 0;
84         else
85             ssl_opts.verify = 1;
86         ssl_opts.CApath = opts.cacpath;
87         ssl_opts.keyStore = opts.cert;
88         ssl_opts.trustStore = opts.cacfile;
89         ssl_opts.privateKey = opts.key;
90         ssl_opts.privateKeyPassword = opts.keypass;
91         ssl_opts.enabledCipherSuites = opts.ciphers;
92         conn_opts.ssl = &ssl_opts;
93     }
```

```

94
95     if (opts.MQTTVersion == MQTTVERSION_5)
96     {
97         MQTTProperties props = MQTTProperties_initializer;
98         MQTTProperties willProps = MQTTProperties_initializer;
99         MQTTResponse response = MQTTResponse_initializer;
100
101         conn_opts.cleanstart = 1;
102         response = MQTTClient_connect5(client, &conn_opts, &props, &willProps);
103         rc = response.reasonCode;
104         MQTTResponse_free(response);
105     }
106     else
107     {
108         conn_opts.cleansession = 1;
109         rc = MQTTClient_connect(client, &conn_opts);
110     }
111
112     if (opts.verbose && rc == MQTTCLIENT_SUCCESS)
113         fprintf(stderr, "Connected\n");
114     else if (rc != MQTTCLIENT_SUCCESS && !opts.quiet)
115         fprintf(stderr, "Connect failed return code: %s\n", MQTTClient_strerror(rc));
116
117     return rc;
118 }

```

Вводим из консоли логин, пароль, адрес, топик и название файла в который будут сохраняться пришедшие значения:

```
126 int main(int argc, char** argv)
127 {
128     int max = 15;
129     char ip [max];
130     char user [max];
131     char pass [max];
132     char topic[max];
133     char port [max];
134     char log_filename[max];
135
136     printf("Input login:"); //login
137     scanf("%s", user);
138
139     printf("Input password:"); //password
140     scanf("%s", pass);
141
142     printf("Input host:"); //host
143     scanf("%s", ip);
144
145     printf("Input port:"); //port
146     scanf("%s", port);
147
148     printf("input topic:"); //topic
149     scanf("%s", topic);
150
151     printf("Input log filename:"); //log filename
152     scanf("%s", log_filename);
153
154     opts.username = user;
155     opts.password = pass;
156     opts.host = ip;
157     opts.topic = topic;
158     opts.port = port;
```

Дебаг проверки и проверки на подключение к MQTT брокеру:

```
172     if (strchr(opts.topic, '#') || strchr(opts.topic, '+'))
173         opts.verbose = 1;
174
175     if (opts.connection)
176         url = opts.connection;
177     else
178     {
179         url = malloc(100);
180         sprintf(url, "%s:%s", opts.host, opts.port);
181     }
182     if (opts.verbose)
183         printf("URL is %s\n", url);
184
185     if (opts.tracelevel > 0)
186     {
187         MQTTClient_setTraceCallback(trace_callback);
188         MQTTClient_setTraceLevel(opts.tracelevel);
189     }
190
191     if (opts.MQTTVersion >= MQTTVERSION_5)
192         createOpts.MQTTVersion = MQTTVERSION_5;
193     rc = MQTTClient_createWithOptions(&client, url, opts.clientid, MQTTCLIENT_PERSISTENCE_NONE,
194         NULL, &createOpts);
195     if (rc != MQTTCLIENT_SUCCESS)
196     {
197         if (!opts.quiet)
198             fprintf(stderr, "Failed to create client, return code: %s\n", MQTTClient_strerror(rc));
199         exit(EXIT_FAILURE);
200     }
201
```

```
201
202
203     if (myconnect(client) != MQTTCLIENT_SUCCESS)
204         goto exit;
205
206     if (opts.MQTTVersion >= MQTTVERSION_5)
207     {
208         MQTTResponse response = MQTTClient_subscribe5(client, opts.topic, opts.qos, NULL, NULL);
209         rc = response.reasonCode;
210         MQTTResponse_free(response);
211     }
212     else
213         rc = MQTTClient_subscribe(client, opts.topic, opts.qos);
214     if (rc != MQTTCLIENT_SUCCESS && rc != opts.qos)
215     {
216         if (!opts.quiet)
217             fprintf(stderr, "Error %d subscribing to topic %s\n", rc, opts.topic);
218         goto exit;
219     }
220
```


Инициализация функций времени и объявления массивов для вывода и записи в файл значений:

```
221 while (!toStop)
222 {
223     time_t mytime = time (NULL);
224     struct tm *now = localtime(&mytime);
225
226     char* topicName = topic;
227     int topicLen;
228     MQTTClient_message* message = NULL;
229
230     rc = MQTTClient_receive(client, &topicName, &topicLen, &message, 1000);
231     if (rc == MQTTCLIENT_DISCONNECTED)
232         myconnect(client);
233     else if (message)
234     {
235         size_t delimlen = 0;
236
237         if (opts.verbose)
238             printf("%s\t", topicName);
239         if (opts.delimiter)
240             delimlen = strlen(opts.delimiter);
241         if (opts.delimiter == NULL || (message->payloadlen > delimlen &&
242             strncmp(opts.delimiter, &((char*)message->payload)[message->payloadlen - delimlen], delimlen) == 0))
243             printf("%.s", message->payloadlen, (char*)message->payload);
244         else
245             printf("%.s%s", message->payloadlen, (char*)message->payload, opts.delimiter);
246
247         char str[255]={0};
248         char time[20];
249         char date[20];
250
```

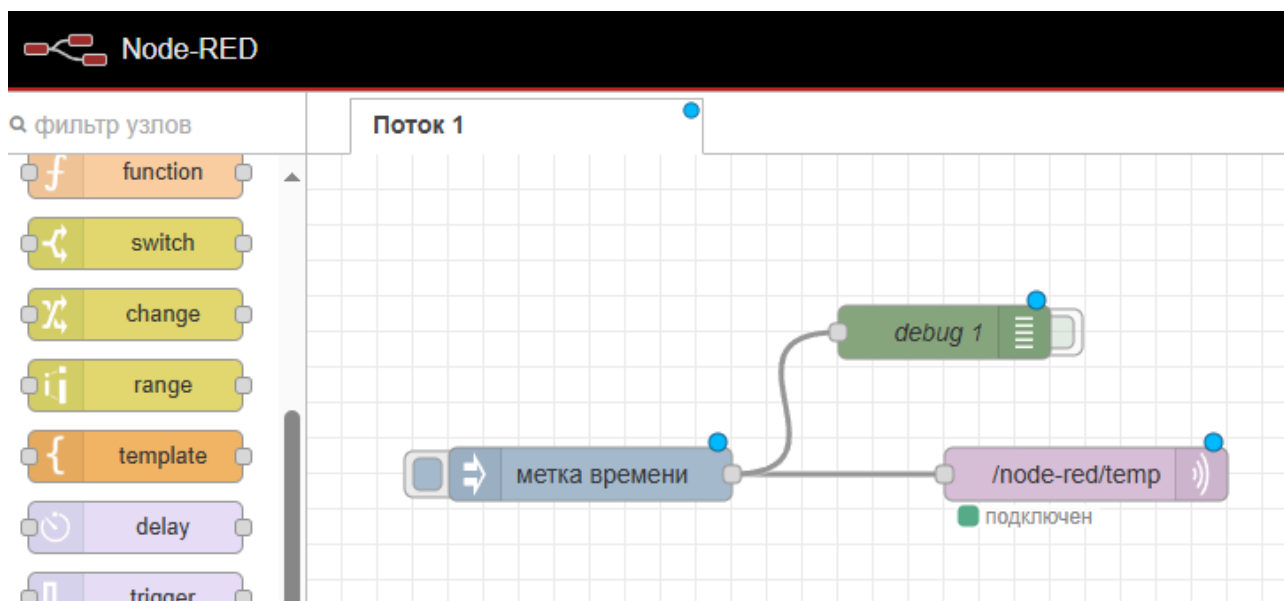
Вывод пришедших значений в консоль и запись в файл:

```
250
251         strftime(date,sizeof(str),"%D",now);
252         strftime(time,sizeof(str),"%T",now);
253
254         strcat(str,(char*)message->payload);
255         printf("%s\n",str);
256         printf("%s\n",time);
257         printf("%s\n",date);
258         fprintf(file,"Temp: %s Time: %s Date: %s\n",str,time,date);
259
260
261         fflush(stdout);
262         MQTTClient_freeMessage(&message);
263         MQTTClient_free(topicName);
264         printf("\n=> Press any key and <Enter>, # to end the connection\n");
265         char c;
266         while((c=getchar())!='\n')
267             if(c == '#')
268             {
269                 goto exit;
270             }
271     }
272 }
273
274 exit:
275     fclose(file);
276     MQTTClient_disconnect(client, 0);
277
```

Глава 3 Тестирование

3.1 Тестирование связности MQTT брокера, Node-red, InfluxDB

Для теста MQTT брокера, Node-red и InfluxDB, построим в Node-red эмулятор датчика из 2 нод это: метка времени и mqtt out:



настройка нод:

mqtt out:

в строке сервер из списка выбираем ip адрес виртуальной машины и порт брокера.

в строке прописываем топик.

в строке QoS выбираем 1:

Изменить узел mqtt out

УдалитьОтменаГотово

Свойства

Сервер

192.168.0.4:1883

Тема

/node-red/temp

QoS

1

Хранить

User Properties

▼ none

Response topic

▼ none

Content Type

▼ none

Expiry (secs)

▼ none

Имя

Имя

Совет: Оставьте тему, qos или хранение пустыми, если Вы хотите устанавливать их через свойства msg.

генерируем значения меткой времени и смотрим полученный результат в InfluxDB:

5

Data Explorer

Switch to old Data Explorer

Table

CUSTOMIZE

Local

SAVE AS

_start	_stop	_time	_value	_field	_measurement
2024-08-03 08:28:02 GMT+3	2024-08-03 09:28:02 GMT+3	2024-08-03 09:28:00 GMT+3	1 722 666 477 904,13	value	mode-redtemp
2024-08-03 08:28:02 GMT+3	2024-08-03 09:28:02 GMT+3	2024-08-03 09:28:02 GMT+3	1 722 666 488 821	value	mode-redtemp

3.2 Тестирование клиента-издателя

Собираем клиента через gcc:

```
<gcc pub.c -o pub -lpaho-mqtt3cs>
```

Запускаем издателя и вводим данные для подключения:



```
geany_run_script_UFAVR2.sh
File Edit Tabs Help
Input login:stat
Input password:ss
Input host + port:192.168.0.4:1883
input topic:MQTT
Input filename:filename.csv
```

После успешного подключения клиент должен отправлять данные температуры:

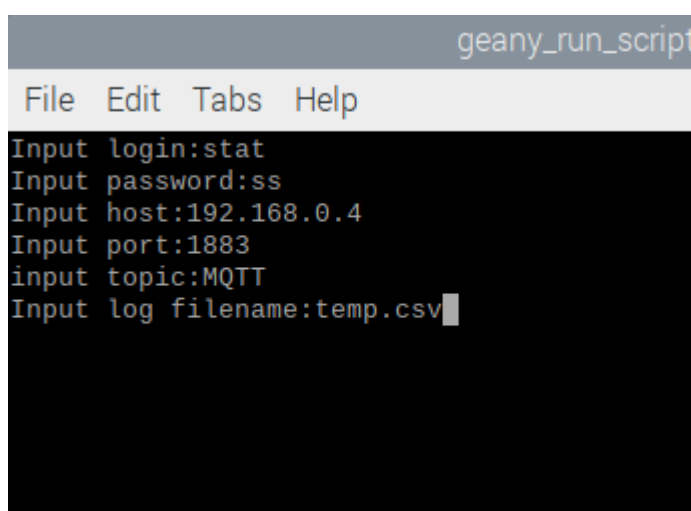
```
geany_run_3
File Edit Tabs Help
Input login:stat
Input password:ss
Input host + port:192.168.0.4:1883
input topic:MQTT
Input filename:filename.csv
2021 1 16 1 1 -47
2021 1 16 1 3 -44
2021 1 16 1 4 -43
Wrong format in line jbkfjdb;1;5;#000?
2021 2 16 1 1 -25
2021 2 17 1 1 -30
2021 3 16 1 1 -10
2021 4 16 1 1 100
2021 5 16 1 1 10
2021 6 16 1 1 25
2021 7 16 1 1 30
2021 8 16 1 1 20
2021 9 16 1 1 18
2021 10 16 1 1 2
2021 11 16 1 1 -5
2021 12 16 1 1 -20
-47,1
-44,2
-43,3
```

3.3 Тестирование клиента-подписчика

Собираем клиента через gcc:

```
<gcc sub.c -o sub -lpaho-mqtt3cs>
```

Запускаем подписчика и вводим данные для подключения:



```
geany_run_script
File Edit Tabs Help
Input login:stat
Input password:ss
Input host:192.168.0.4
Input port:1883
input topic:MQTT
Input log filename:temp.csv
```

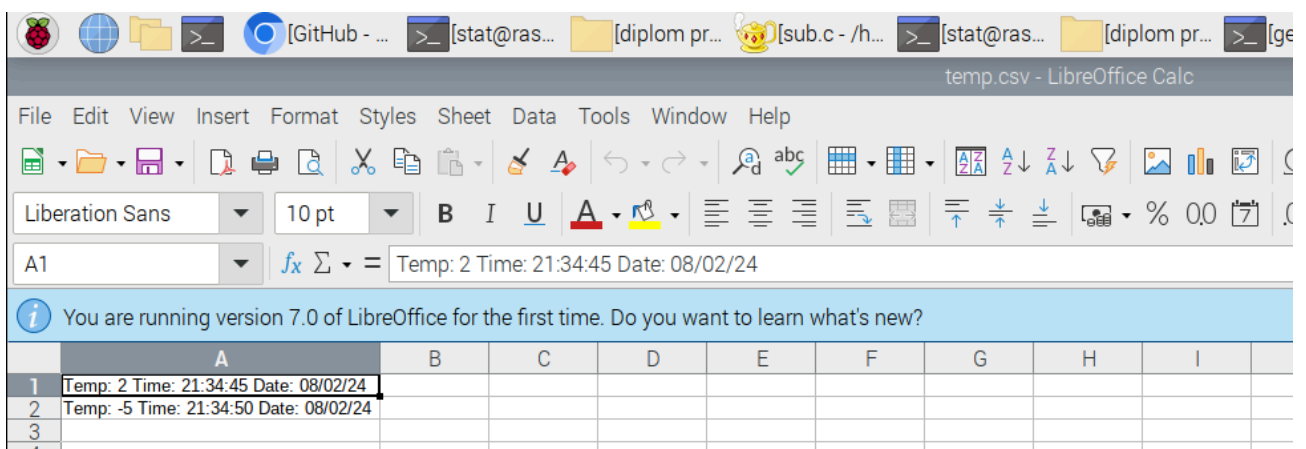
После успешного подключения клиент должен получать данные температуры:

```
geany_run_script_XH
File Edit Tabs Help
Input login:stat
Input password:ss
Input host:192.168.0.4
Input port:1883
input topic:MQTT
Input log filename:temp.csv
2
2
21:34:45
08/02/24

=> Press any key and <Enter>, # to end the connection
-5
-5
21:34:50
08/02/24

=> Press any key and <Enter>, # to end the connection
█
```

И после завершения работы программы создается log файл:



temp.csv - LibreOffice Calc										
File Edit View Insert Format Styles Sheet Data Tools Window Help										
Liberation Sans 10 pt B I U A % 0.0										
A1 fx Σ = Temp: 2 Time: 21:34:45 Date: 08/02/24										
You are running version 7.0 of LibreOffice for the first time. Do you want to learn what's new?										
	A	B	C	D	E	F	G	H	I	
1	Temp: 2 Time: 21:34:45 Date: 08/02/24									
2	Temp: -5 Time: 21:34:50 Date: 08/02/24									
3										
4										

3.4 Общее тестирование всего проекта

Клиент-издатель будет передавать данные в топик из которого они отправятся в InfluxDB и клиенту-подписчику

```
stat@raspberrypi: ~  
File Edit Tabs Help  
Input login:stat  
Input password:ss  
Input host + port:192.168.0.4:1883  
input topic:/node-red/temp  
Input filename:filename.csv  
2021 1 16 1 1 -47  
2021 1 16 1 3 -44  
2021 1 16 1 4 -43  
Wrong format in line jbkfjdb;1;5;#000?  
2021 2 16 1 1 -25  
2021 2 17 1 1 -30  
2021 3 16 1 1 -10  
2021 4 16 1 1 100  
2021 5 16 1 1 10  
2021 6 16 1 1 25  
2021 7 16 1 1 30  
2021 8 16 1 1 20  
2021 9 16 1 1 18  
2021 10 16 1 1 2  
2021 11 16 1 1 -5  
2021 12 16 1 1 -20  
-47,1  
-44,2
```

```
stat@raspberrypi: ~  
File Edit Tabs Help  
root@raspberrypi:/home/stat/Desktop/diplom programs# ./sub  
Input login:stat  
Input password:ss  
Input host:192.168.0.4  
Input port:1883  
input topic:/node-red/temp  
Input log filename:temp.csv  
-47  
-47  
21:56:53  
08/02/24  
  
=> Press any key and <Enter>, # to end the connection  
-44  
-44  
21:56:58  
08/02/24  
  
=> Press any key and <Enter>, # to end the connection  
█
```

Данные из InfluxDB:

5

↑

📈

📄

🔧

📅

🔔

⚙️

Data Explorer

Switch to old Data Explorer ☒

Table

CUSTOMIZE

Local

SAVE AS

2024-08-03 08:58:09 GMT+3	2024-08-03 09:58:09 GMT+3	2024-08-03 09:28:10 GMT+3	1 722 666 489 021	value	/node-red/temp
2024-08-03 08:58:09 GMT+3	2024-08-03 09:58:09 GMT+3	2024-08-03 09:57:00 GMT+3	-45,50	value	/node-red/temp
2024-08-03 08:58:09 GMT+3	2024-08-03 09:58:09 GMT+3	2024-08-03 09:57:10 GMT+3	-21,50	value	/node-red/temp
2024-08-03 08:58:09 GMT+3	2024-08-03 09:58:09 GMT+3	2024-08-03 09:57:20 GMT+3	-27,50	value	/node-red/temp
2024-08-03 08:58:09 GMT+3	2024-08-03 09:58:09 GMT+3	2024-08-03 09:57:30 GMT+3	45	value	/node-red/temp
2024-08-03 08:58:09 GMT+3	2024-08-03 09:58:09 GMT+3	2024-08-03 09:57:40 GMT+3	17,50	value	/node-red/temp
2024-08-03 08:58:09 GMT+3	2024-08-03 09:58:09 GMT+3	2024-08-03 09:57:50 GMT+3	25	value	/node-red/temp
2024-08-03 08:58:09 GMT+3	2024-08-03 09:58:09 GMT+3	2024-08-03 09:58:00 GMT+3	10	value	/node-red/temp
2024-08-03 08:58:09 GMT+3	2024-08-03 09:58:09 GMT+3	2024-08-03 09:58:09 GMT+3	-5	value	/node-red/temp

Заключение

В итоге получилась минималистичная система сбора телеметрии и ее хранения, которую можно уже использовать на практике.

В нее можно добавить Grafana для более наглядной визуализации данных.

Также в систему можно добавить другие возможные датчики.

Список используемой литературы

- [*Node-RED \(nodered.org\)*](https://nodered.org/)
- [*Eclipse Mosquitto*](#)
- [*InfluxDB Time Series Data Platform / InfluxData*](#)

Приложения

1. https://github.com/billybobys/fx4Jf95_Ql