

EasyHomeCloud Manual

EasyHomeCloud_v0.5.5

Оглавление

1.	Общая информация.....	1
2.	Настройка и запуск EasyHomeCloud.....	2
2.1	Windows.....	2
2.2	Linux	2
2.3	Docker	2
3.	Установка и настройка вспомогательных инструментов.....	3
3.1	Postman	3
3.2	MQTT Broker	4
	Mosquitto (Windows)	4
	EMQX (Windows 10+)	4
3.3	NodeRED	6
4.	Монитор EasyHomeCloud	12
5.	Настройка EasyHomeCloud с помощью settings.ini	14

1. Общая информация

Система EasyHomeCloud передаёт информацию о регистрах контроллеров клиентам и от клиентов – контроллерам, создавая связь многие ко многим. EasyHomeCloud создает возможность подключения клиентов к контроллерам, у которых динамически изменяемый адрес, поскольку контроллеры и клиенты должны подключаться по известному адресу сервера EasyHomeCloud.

Изначально клиент реализован на стеке C++/Qt устанавливаемым приложением EasyHome на Windows, Linux, Android, Mac OS. Общение напрямую реализовано по протоколу Modbus TCP. Поэтому, EasyHomeCloud, в первую очередь, реализует трансляцию таких запросов. Это позволяет выделить статический IP-адрес только для сервера с EasyHomeCloud, к которому подключаются контроллеры и клиенты, имеющие любой IP-адрес (контроллеры, к которым подключаются клиенты напрямую без EasyHomeCloud, должны иметь статический IP-адрес).

Следующий шаг – это адаптация EasyHomeCloud под работу с web-клиентами. Для этого было создано настоящее HTTP API для клиента. Система EasyHomeCloud имеет возможность управления контроллерами посредством HTTP 1.1 POST и GET запросов с телом в формате JSON.

Поскольку в интерфейсах клиентов существуют элементы, которые необходимо каждую секунду обновлять, возникает потребность введения протокола MQTT, который снимет с клиентов и сервера нагрузку, возникающую при ежесекундном запросе от клиента по каждому необходимому регистру (особенно в случае, если несколько клиентов спрашивают одни и те же регистры или их диапазоны). Вместо этого, клиент выполняет подписку на регистры, которые затем

читает из MQTT брокера. ПО EasyHomeCloud по мере изменения данных регистров публикует их в MQTT брокер. Поскольку существует проблема перегрузки сервера EasyHomeCloud запросами на уже устаревшие и неактуальные регистры, существует механизм очистки таких регистров по истечении времени подписки или отключении всех клиентов от ПЛК.

Сама подписка реализуется клиентом с помощью HTTP запроса с соответствующим параметром.

2. Настройка и запуск EasyHomeCloud

2.1 Windows

- Открыть папку **EasyHomeCloud-win**. Запустить **EasyHomeCloud.exe**.

2.2 Linux

- Открыть папку **EasyHomeCloud-linux**. Выполнить

```
sudo chmod +x EasyHomeCloud
./EasyHomeCloud
```

2.3 Docker

- Выполнить загрузку архива **eh_docker_image.tar** в систему Docker и запустить его образ

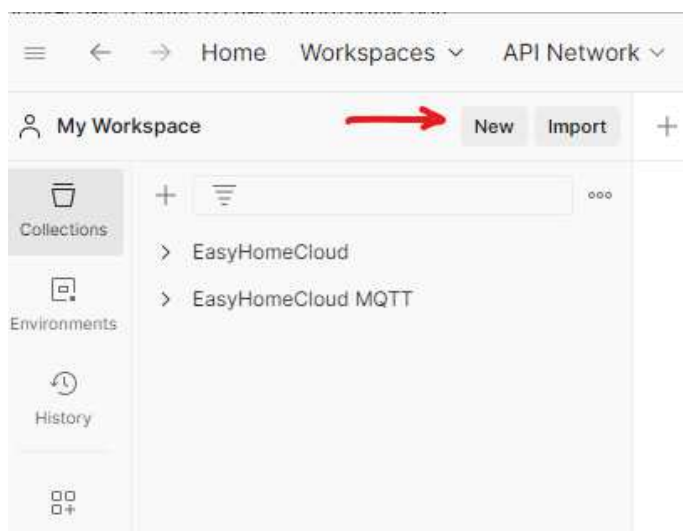
```
docker load --input eh_docker_image.tar

sudo docker run -p 1500:1500 -p 59501:59501 -p 59503:59503 -p 59534:59534 -p
59543:59543 easyhomecloud
```

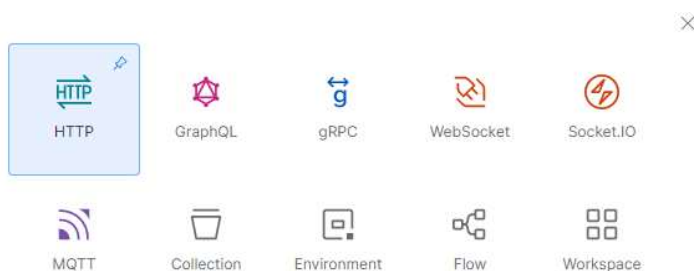
3. Установка и настройка вспомогательных инструментов

3.1 Postman

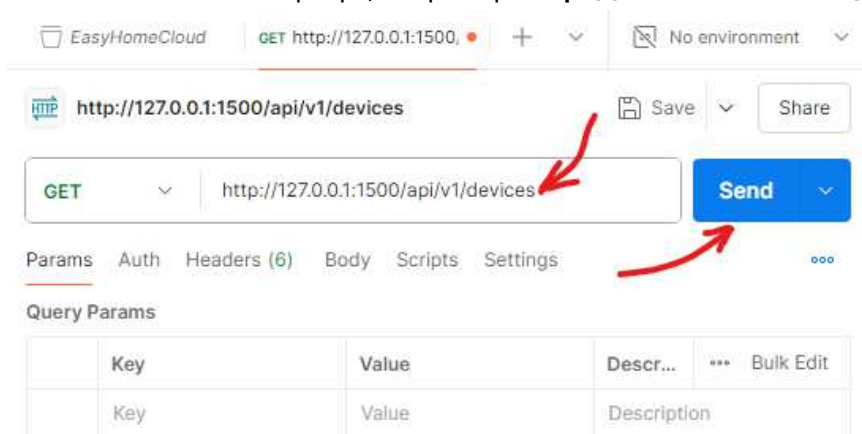
1. [Скачать с официального сайта Postman](#) и установить
2. Создать новый запрос



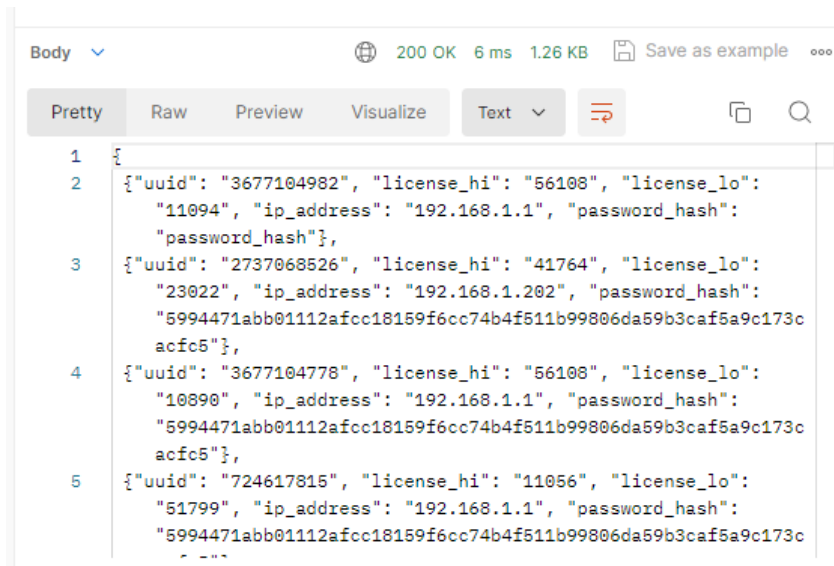
3. Выбрать HTTP запросы



4. Заполнить поле сервера, например **http://127.0.0.1:1500/api/v1/devices** и нажать **Send**



Это позволит отправить GET запрос и получить ответ от **запущенного сервера EasyHomeCloud** в виде списка известных ему устройств в поле вывода:



Пример HTTP-запроса, сформированного и отправленного через программу Postman:

POST /api/v1/device_request HTTP/1.1

Host: 127.0.0.1:1500

Content-Type: application/json

Content-Length: 185

```

{
  "password": "5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf5a9c173cacfc5",
  "request_type": "get",
  "licenseID_hi": 17111,
  "licenseID_lo": 20836,
  "word": {
    "range_begin": 310,
    "range_end": 314,
    "list": [315, 316]
  }
}

```

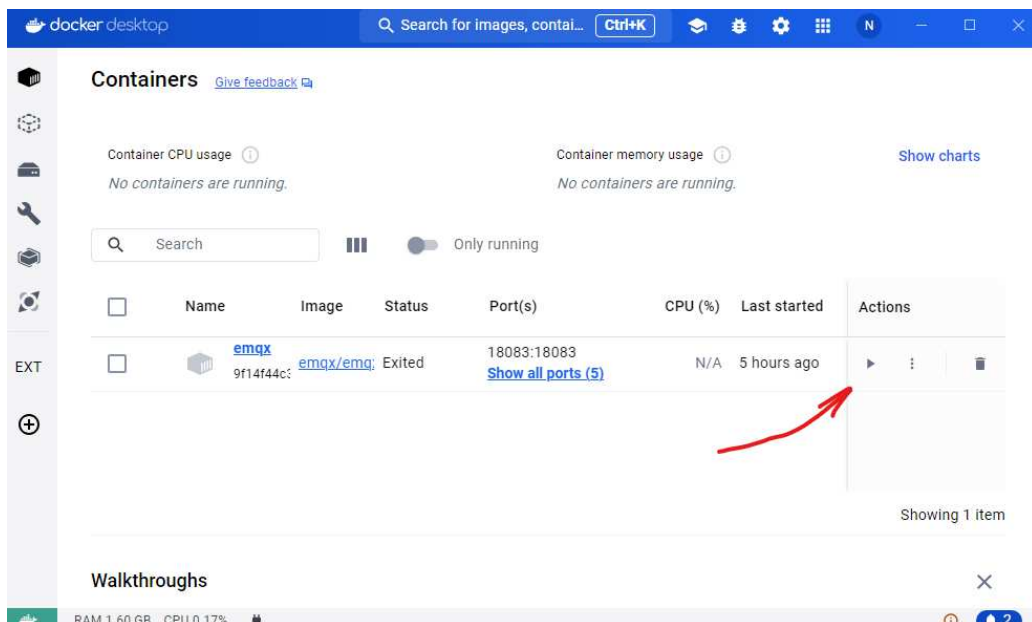
3.2 MQTT Broker

Mosquitto (Windows)

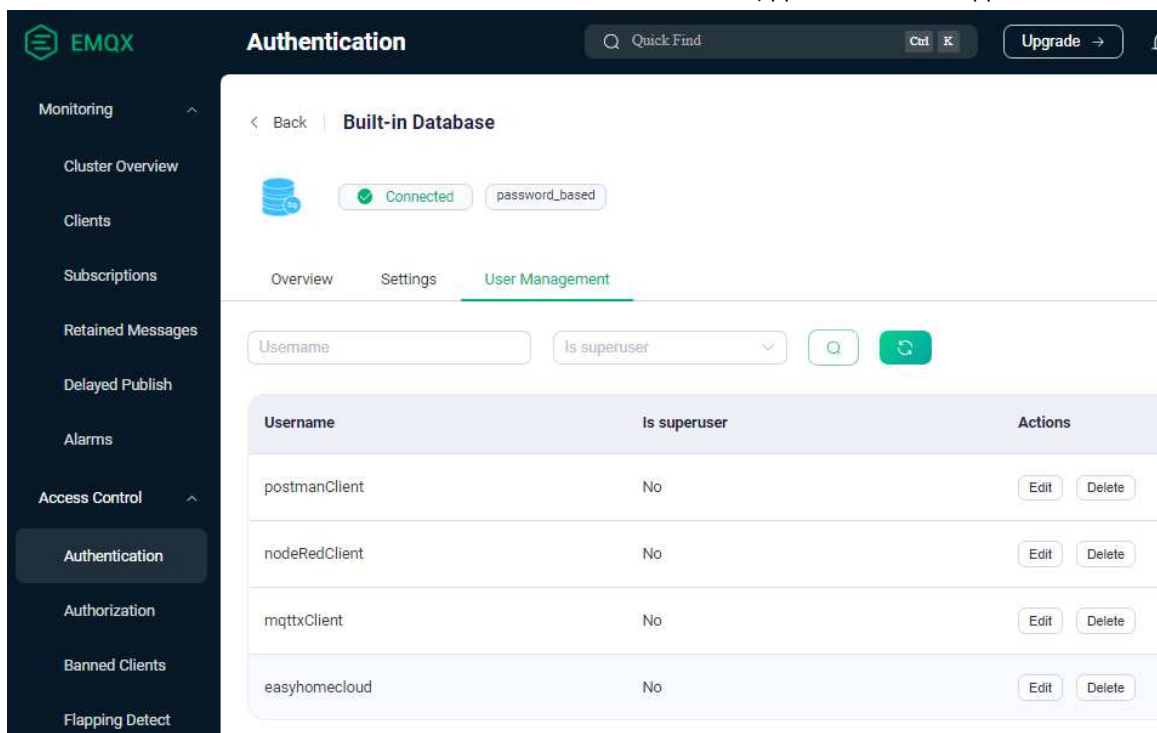
1. Установить [брокер Mosquitto](#)
2. Запустить брокер (для Windows – обычно C:\Program Files\mosquitto\mosquitto.exe)

EMQX (Windows 10+)

1. Установить [Docker Desktop](#) для развертывания MQTT брокера
2. Установить Docker контейнер [MQTT брокер EMQX](#)
3. Запустить брокер



4. Подключиться через браузер по адресу <http://127.0.0.1:18083/>
5. Выполнить регистрацию/авторизацию
6. Добавить клиентов (в том числе для тестов через Postman)
 - a. Для этого перейти во вкладку **Access Control** -> **Authentication**
 - b. В правом верхнем углу нажать **Create**. Выбрать **Mechanism – Password-Based, Backend – Built-in Database, Configuration** оставить заполненной по-умолчанию. Нажать **Create**
 - c. В поле **Action** нажать на **Users** и заполнить поля, добавив необходимых клиентов



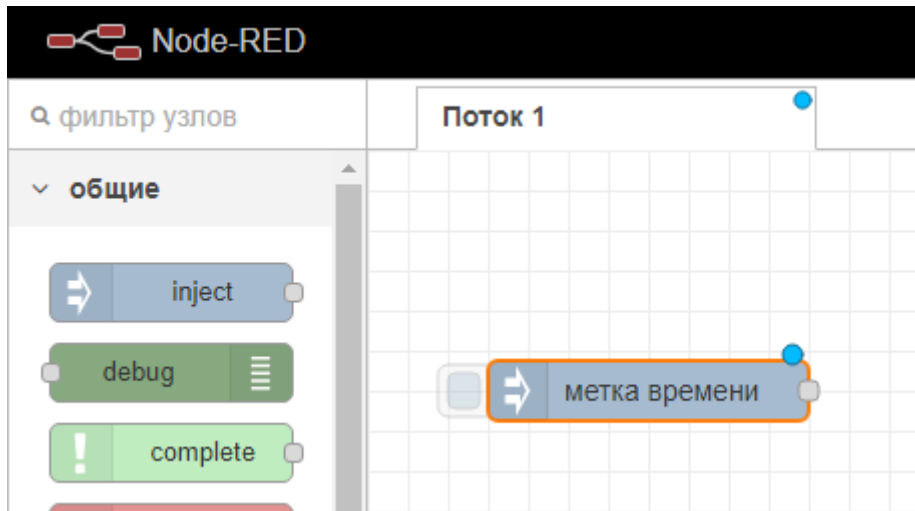
Например,

Для клиента EasyHomeCloud имя пользователя и пароль выбраны – easyhomecloud (указывается в settings.ini),

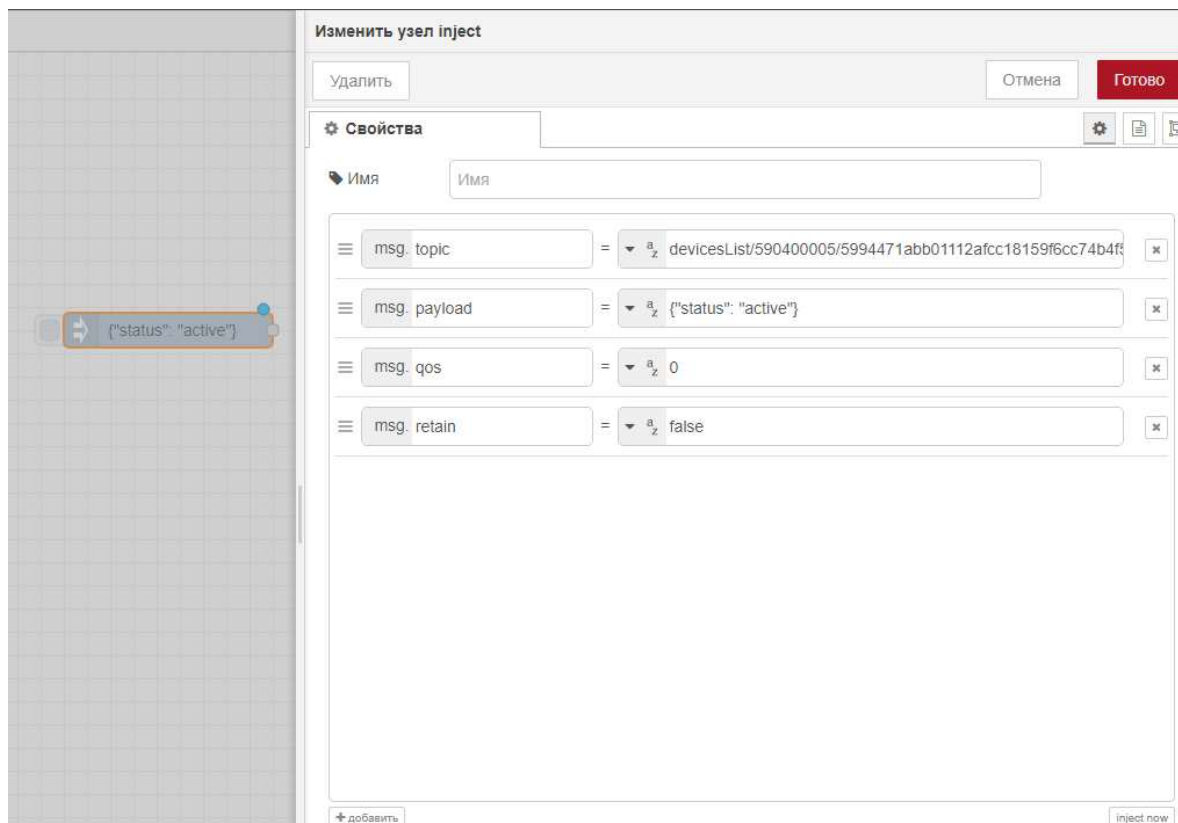
Для клиента Node-red – nodeRedClient – указывается для блоков **mqtt** в Node-red во вкладке **Безопасность**.

3.3 NodeRED

1. Выполните [инструкции с официального сайта Node-red](#) для установки
2. Запустите Node-red с помощью команды **node-red** в командной строке
3. Войдите в среду разработки через браузер <http://127.0.0.1:1880/>
4. Построим конструкцию, которая будет сообщать облаку о том, что хотя бы один клиент читает регистры устройства
- 4.1. Из категории **общие** добавьте на поле блок **inject**, который будет генерировать объект с сообщением по нажатию на кнопку:

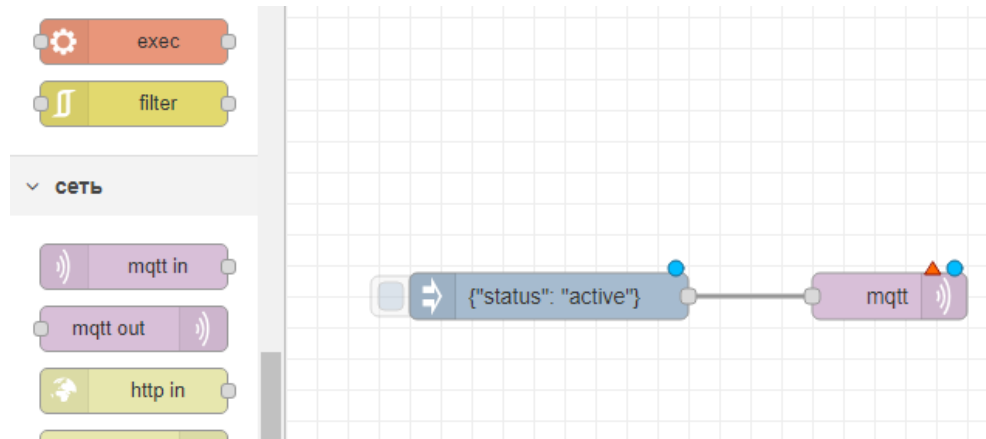


- 4.2. Двойным нажатием ЛКМ откройте настройку блока и добавьте поля:
 - a. `topic = devicesList/<ПЛК_ID>/<passwordHash>/statusOfClient`
 - b. `payload = {"status": "active"}`
 - c. `qos = 0`
 - d. `retain = false`



- 4.3. Нажать **Готово**

4.4. Добавить блок **mqtt out** из вкладки **сеть** и соединить его с блоком **inject**



4.5. В настройках блока добавить MQTT сервер, нажатием на редактирование

The screenshot shows the 'Изменить узел mqtt out' (Edit mqtt out node) dialog box. At the top, there are buttons for 'Удалить' (Delete), 'Отмена' (Cancel), and 'Готово' (Done). Below these is the 'Свойства' (Properties) tab. The 'Сервер' (Server) field is set to 'localhost:1883'. A red arrow points to a small edit icon (pencil) next to this field. Other fields include 'Тема' (Topic) set to 'Тема', 'QoS' set to a dropdown, 'Хранить' (Store) with a checkbox, and 'Имя' (Name) set to 'Имя'. A yellow tip box at the bottom states: 'Совет: Оставьте тему, qos или хранение пустыми, если Вы хотите устанавливать их через свойства msg.'

4.6. В настройках сервера установить адрес сервера и порт, например **localhost** и **1883**

The screenshot shows the 'Изменить узел mqtt-broker' (Edit mqtt-broker node) dialog box. It has tabs for 'Свойства' (Properties), 'Соединение' (Connection), 'Безопасность' (Security), and 'Сообщения' (Messages). The 'Соединение' tab is active. The 'Сервер' (Server) field is set to 'localhost' and the 'Порт' (Port) field is set to '1883'. There are checkboxes for 'Connect automatically' (checked) and 'TLS' (unchecked). The 'Protocol' dropdown is set to 'MQTT V3.1.1'. The 'ID клиента' (Client ID) field is empty with a hint: 'Оставьте пустым для автоматически сгенерированного'. The 'Keep-alive время (сек)' (Keep-alive time (sec)) field is set to '60'. The 'i Session' checkbox 'Использовать чистую сессию' (Use clean session) is checked. Buttons for 'Удалить' (Delete), 'Отмена' (Cancel), and 'Обновить' (Update) are at the top.

4.7. Во вкладке **Безопасность** укажем имя пользователя и пароль, как зарезервировали в настройках брокера (nodeRedClient)

Изменить узел mqtt out > Изменить узел mqtt-broker

Удалить Отмена Обновить

Свойства

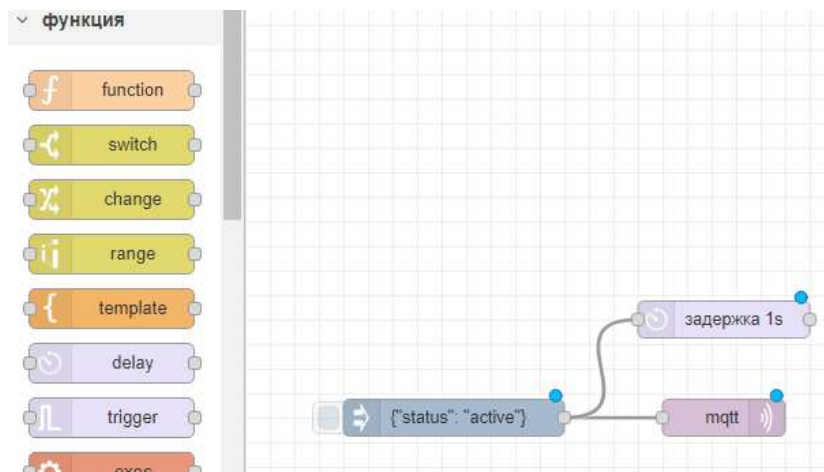
Имя

Соединение Безопасность Сообщения

Имя польз.

Пароль

5. Создадим модуль для отправки HTTP запроса для подписки на регистры
 - 5.1. Добавим **delay** из категории **функции** и соединим его с блоком **inject**, параллельно блоку **mqtt out**. Установим задержку в размере 1 секунды



- 5.2. Последовательно подключим блок **function** из категории **функции**. Зададим ему следующую функцию:

```

const lic_hi = 9008;
const lic_lo = 51717;

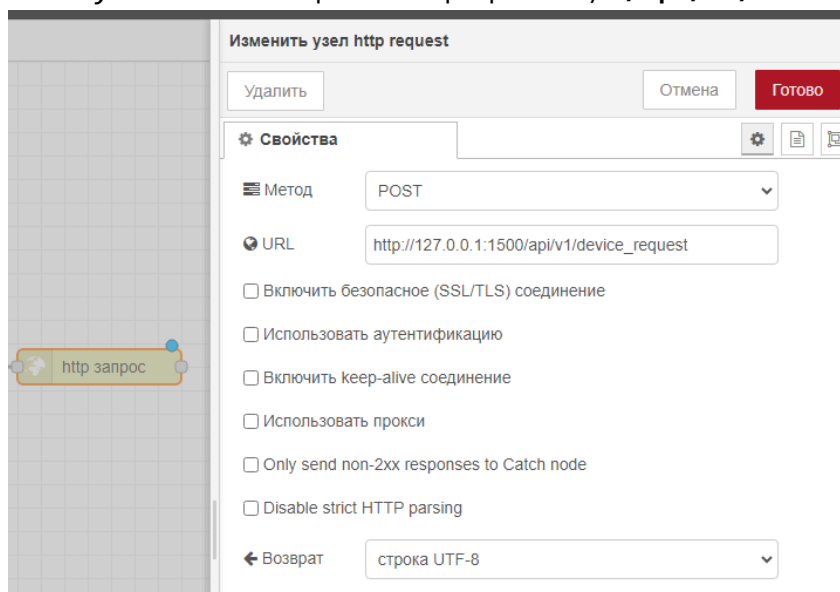
let jsonObj = {
  "password":
"5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf5a9c173cacfc5",
  "request_type": "subscribe",

  "licenseID_hi": lic_hi,
  "licenseID_lo": lic_lo,

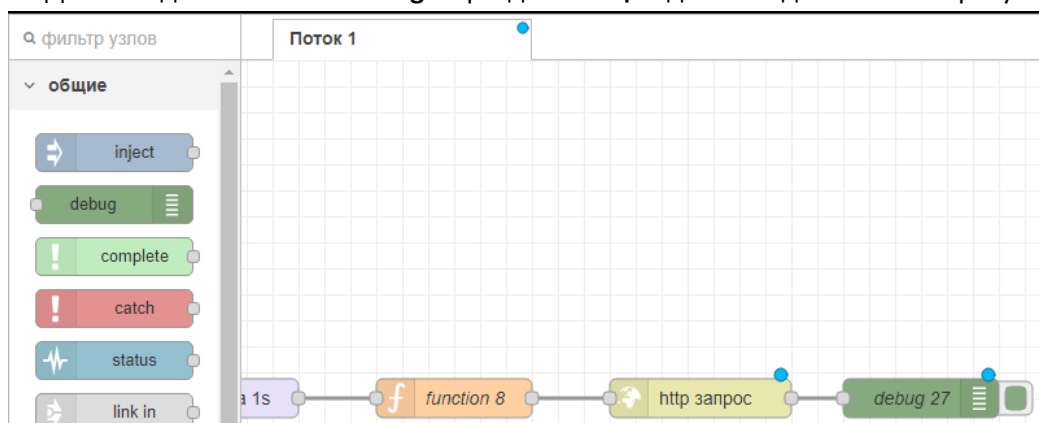
  "word": {
    "range_begin": 400,
    "range_end": 800,
    "list": [333, 334, 335, 336, 340, 344, 348, 349, 351, 353, 358, 359,
367, 376, 378, 380, 381]
  }
}
msg.payload = jsonObj;
return msg;

```

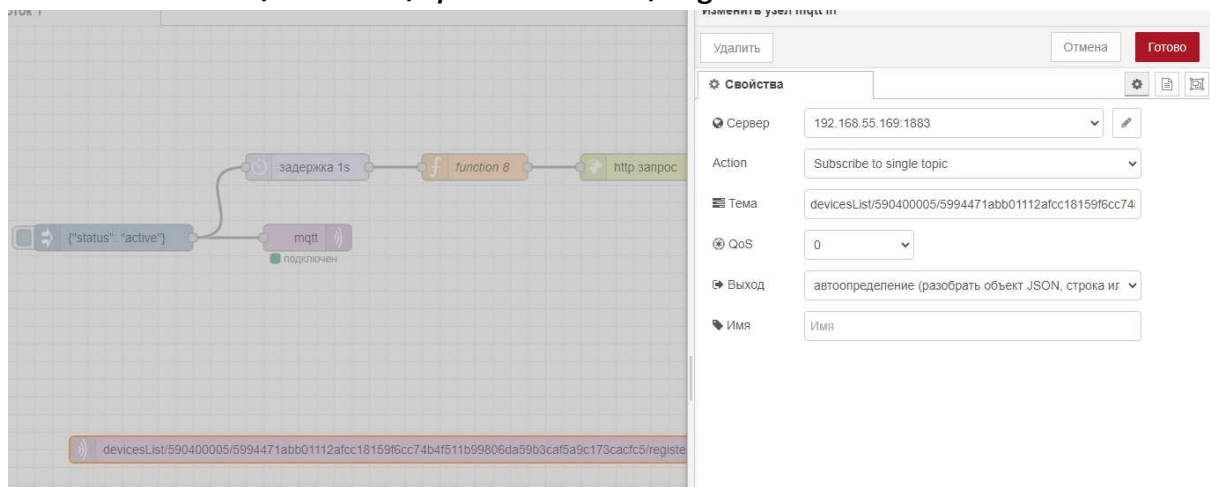
5.3. Добавим последовательно блок **http request** из раздела **сеть**. Зададим URL (адрес EasyHomeCloud и порт HTTP сервера 1500) + **/api/v1/device_request**



5.4. Далее подключим блок **debug** из раздела **общие** для вывода ответов в правую часть интерфейса

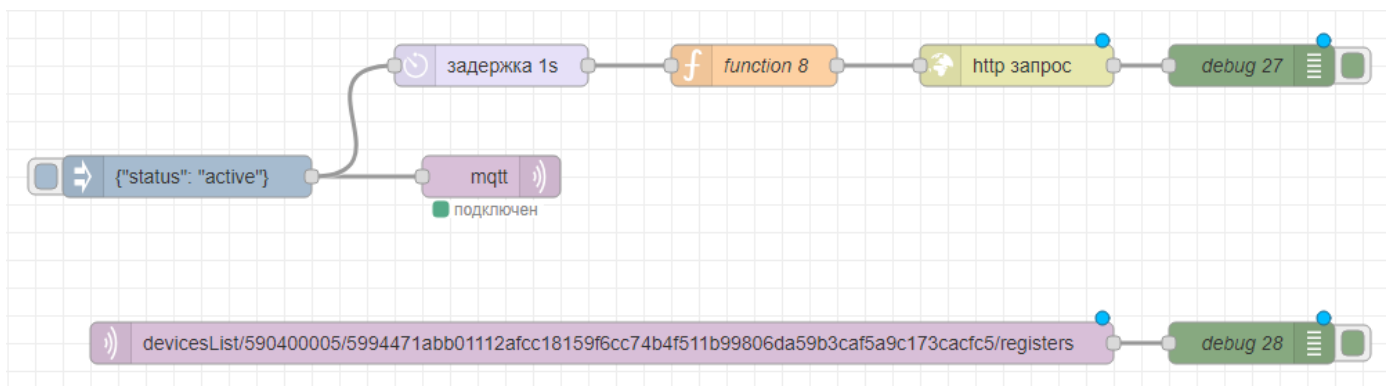


6. Создадим модуль для приёма mqtt сообщений от брокера по подписке.
- 6.1. Добавим в пустое пространство этого же потока (вкладки) блок **mqtt in** из раздела **сеть**.
- Выставим настройки следующим образом:
- Выберем тот же сервер, что и ранее
 - Подпишемся на один топик – топик для регистров конкретного устройства:
devicesList/<ПЛК ID>/<passwordHash>/registers

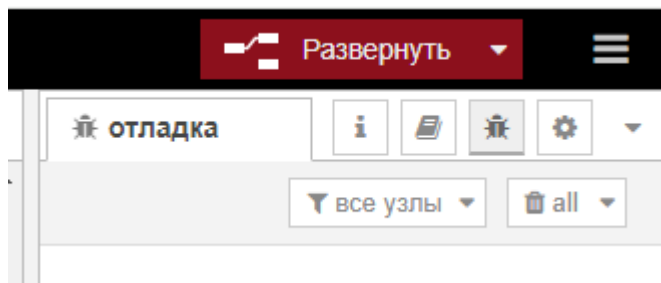


- Добавим блок **debug** для вывода приходящих сообщений от брокера

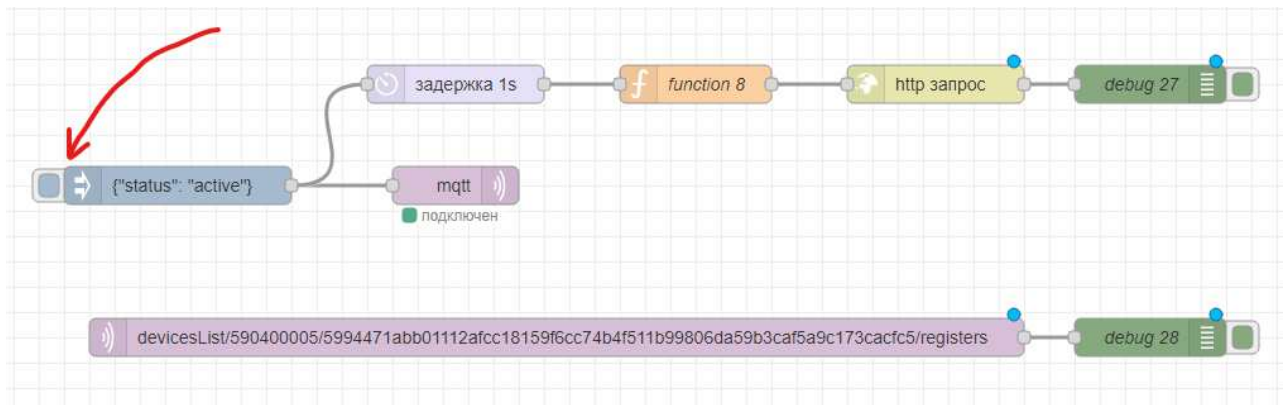
В результате получим программу:



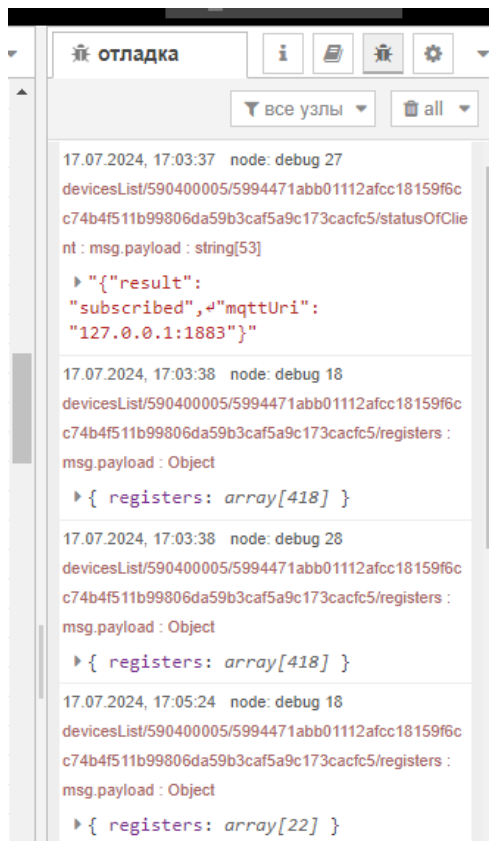
7. Развернём поток с помощью соответствующей кнопки в правом верхнем углу:



8. Выполним эмуляцию клиента (отправим http-запрос с подпиской на mqtt-топик регистров устройства) с помощью нажатия на кнопку блока **inject**:



В результате, в правом окне **отладка** будут выводиться все пришедшие сообщения:



Приложение отображает необходимую информацию в режиме реального времени:

```

C:\C++_QT_Projects\cloud\Ea X + v
-----
12:53:25 - Incoming connection for DEVICES, IP: 192.168.1.1
12:53:25 - Device ID: [29487, 14348], 192.168.1.1 connected.
-----
12:56:22 - Incoming connection for DEVICES, IP: 192.168.1.202
12:56:23 - Device ID: [31541, 15476], 192.168.1.202 connected.
-----
12:56:27 - Incoming connection for DEVICES, IP: 192.168.1.1
12:56:27 - Device ID: [9008, 51717], 192.168.1.1 connected.
-----
13:00:10 - Incoming connection for CLIENTS, IP: 192.168.1.1
13:00:10 - Client 192.168.1.1 authenticated to the device ID: [29487, 14348]
13:00:10 - Device ID: [9008, 51717], 192.168.1.1 disconnected.

N0 - ID: [29487, 14348], 192.168.1.1 - EHs:1 - MQTT_Regs:0 - MB(reqs;regs)/s:0.90;25.40
  1st con:27/08/24 12:53:25 - last:27/08/24 12:53:25 - lost:-
N1 - ID: [31541, 15476], 192.168.1.202 - EHs:0 - MQTT_Regs:0 - MB(reqs;regs)/s:0.20;4.40
  1st con:27/08/24 12:56:23 - last:27/08/24 12:56:23 - lost:-
! N2 ID: [9008, 51717], 192.168.1.1
  1st con:27/08/24 12:56:27 - last:27/08/24 12:56:27 - lost:27/08/24 13:00:10

-----
| devices count| clients count| last client connect| last client lost|
|              |              | 27/08/24 13:00:10| -|
-----
| first device connect| last device connect| last device lost|
| 27/08/24 12:53:25 N0| 27/08/24 12:56:27 N2| 27/08/24 13:00:10 N2|
-----
| last MQTT connect| last MQTT lost| MQTT Status| HTTP reqs/s|
| 27/08/24 12:49:36| -| connected| 0.00|
-----
~v0.5.5_w

```

Оно выводит в консоль следующие блоки информации:

- Блок логов о подключениях:

```
=====
12:53:25 - Incoming connection for DEVICES, IP: 192.168.1.1
12:53:25 - Device ID: [29487, 14348], 192.168.1.1 connected.
=====
12:56:22 - Incoming connection for DEVICES, IP: 192.168.1.202
12:56:23 - Device ID: [31541, 15476], 192.168.1.202 connected.
=====
12:56:27 - Incoming connection for DEVICES, IP: 192.168.1.1
12:56:27 - Device ID: [9008, 51717], 192.168.1.1 connected.
=====
13:00:10 - Incoming connection for CLIENTS, IP: 192.168.1.1
13:00:10 - Client 192.168.1.1 authenticated to the device ID: [29487, 14348]
13:00:10 - Device ID: [9008, 51717], 192.168.1.1 disconnected.
```

- Список подключенных контроллеров:

```
N0 - ID: [29487, 14348], 192.168.1.1 - EHs:1 - MQTT_Regs:0 - MB(reqs;regs)/s:0.80;22.40
  1st con:27/08/24 12:53:25 - last:27/08/24 12:53:25 - lost:-
N1 - ID: [31541, 15476], 192.168.1.202 - EHs:0 - MQTT_Regs:0 - MB(reqs;regs)/s:0.20;4.40
  1st con:27/08/24 12:56:23 - last:27/08/24 12:56:23 - lost:-
! N2 ID: [9008, 51717], 192.168.1.1
  1st con:27/08/24 12:56:27 - last:27/08/24 12:56:27 - lost:27/08/24 13:00:10
```

Один контроллер представлен в виде двух строк. Восклицательный знак ! свидетельствует о том, что контроллер отключен. В первой строке отображается информация в следующем порядке:

1. Порядковый номер контроллера в данном списке
2. License ID контроллера (сначала часть HIGH, затем LOW)
3. IP-адрес, с которого совершено подключение непосредственно к приложению EasyHomeCloud
4. EHS – количество подключенных приложений EasyHome (клиенты, подключенные к этому ПЛК)

5. MQTT_Regs – Количество регистров, на которые подписаны web-клиенты по протоколу MQTT, которые необходимо периодически опрашивать у ПЛК
6. MB(reqs;regs)/s – количество запросов;регистров в секунду в среднем за последние 10 секунд, которые были запрошены приложением EasyHomeCloud у ПЛК

Пункты 4-6 не отображаются, если ПЛК отключен от EasyHomeCloud (очевидно, что они = 0 в таком случае).

Вторая строка сообщает о времени и дате первого и последнего подключений, а также о времени последнего отключения (утрате соединения).

- Таблица общей информации:

devices count	clients count	last client connect	last client lost
2	1	27/08/24 13:00:10	-
first device connect	last device connect	last device lost	
27/08/24 12:53:25 N0	27/08/24 12:56:27 N2	27/08/24 13:00:10 N2	
last MQTT connect	last MQTT lost	MQTT Status	HTTP reqs/s
27/08/24 12:49:36	-	connected	0.00

-v0.5.5_w

В **первой** строке отображается информация в следующем порядке:

1. devices count – количество подключенных устройств
2. clients count – количество подключенных клиентов EasyHome
3. last client connect – время самого последнего подключения из всех клиентов EasyHome
4. last client lost – время самого последнего отключения из всех клиентов EasyHome

Во **второй** строке отображается информация в следующем порядке:

5. first device connect – время самого первого подключения из всех ПЛК с указанием номера этого устройства в списке подключенных контроллеров Nx
6. last device connect – время самого последнего подключения из всех ПЛК с указанием номера этого устройства в списке подключенных контроллеров Nx
7. last device lost – время самого последнего отключения из всех ПЛК с указанием номера этого устройства в списке подключенных контроллеров Nx

В **третьей** строке отображается информация в следующем порядке:

8. last MQTT connect – время последнего подключения к MQTT-брокеру
9. last MQTT lost – время последнего отключения от MQTT-брокера
10. MQTT Status – сообщение о том, подключен или отключен MQTT-брокер
11. HTTP reqs/s – количество HTTP-запросов от web-клиентов к приложению EasyHomeCloud в секунду (усредненное за последние 10 секунд)

5. Настройка EasyHomeCloud с помощью settings.ini

Файл settings.ini находится рядом с исполняемым файлом EasyHomeCloud. На данном этапе образ docker необходимо пересобирать при необходимости запустить docker-контейнер с измененными настройками

[Logger]

//Заголовок раздела настроек логгера

MAX_LOG_SIZE_MB=10

//максимальный размер файла логов. В случае превышения старый сохраняется и создается новый файл, в который продолжается запись

[General]

//Заголовок раздела общих настроек

DEVICE_PORT=59503

//Порт для подключения устройств по TCP без SSL

CLIENT_PORT=59501

//Порт для подключения клиентов EasyHome по TCP без SSL

DEVICE_SSL_PORT=59543

//Порт для подключения устройств по TCP с SSL

CLIENT_SSL_PORT=59534

//Порт для подключения клиентов EasyHome по TCP с SSL

AUTH_TIMEOUT_SEC=5

//Время, через которое разрывается соединение между клиентом и сервером, если один из них отключился

GUI_UPDATE_PERIOD=6

//Период отрисовки информации в консоли (gui)

CLONNED_ID_IP_TIMEOUT=300

//Период, после которого устройство с тем же ID, но другим IP может подключиться после отключения с предыдущего IP

[EasyhomeProtocol]

//Заголовок раздела настроек работы с ПЛК

KEEP_ALIVE_SEC=120

//Время жизни соединения (с клиентом или устройством)

CASH_LIFE_TIME_MS=1000

//Время жизни кэша (данных о регистрах), мс

PERIODICAL_REQUEST_INTERVAL_MS=5100

//Частота периодического запроса, чтобы ПЛК не отключил сервер

TIMER_CASH_WAITING=200

//Время ожидания кэша, мс. Для исключения повторного запроса с такими же регистрами на ПЛК.

MQTT_BLOCK_MAX_SIZE=700

//Максимальное количество регистров в одном запросе на ПЛК

MQTT_BLOCK_INTERVAL_MS=200

// Время ожидания между отправками запросов на ПЛК, мс.

[MQTT]

//Заголовок раздела настроек MQTT-клиента

//Если заголовок присутствует, то модуль запускается с указанными в блоке настройками, либо с настройками по-умолчанию. Если заголовок отсутствует – модуль не запускается.

MQTT_BROKER_URI=127.0.0.1:1883

//Адрес брокера

MQTT_NAME=easyhomecloud

//Имя текущего клиента брокера (задается в настройках брокера)

MQTT_PASS=easyhomecloud

//Пароль текущего клиента брокера (задается в настройках брокера)

DATA_UPD_PERIOD=850

//Период опроса всех регистров всех контроллеров для выявления новых данных, мс

MESS_READ_PERIOD=1000

//Период чтения MQTT топика с информацией о подписанных на устройства клиентах, мс.

CLIENTS_LIFETIME=15000000

// Время жизни подписанных на устройства клиентов, мс. При отправке сообщения об активности от любого клиента в топик устройства (devicesList/<ПЛК ID>/<passwordHash>/statusOfClient) этот таймер запускается для данного устройства. По истечении таймера список регистров для мониторинга у этого устройства очищается

REGS_LIFETIME=15000000

// Время жизни конкретного регистра для каждого устройства, мс. По истечении времени регистр удаляется из списка для мониторинга. Обновляется таймер при подписке любого клиента на конкретный регистр (в том числе, в составе диапазона регистров).

[HTTP]

//Заголовок раздела настроек HTTP(S)-сервера

HTTP_REQS_PER_SEC=100.0

//Количество запросов в секунду

HTTP_REQS_MAX=2.0

//Количество запросов для отправки один за другим сразу

[Database]

//Заголовок раздела настроек базы данных. Их лучше оставить неизменно такими, как ниже:

TYPE=SQLite

NAME=database/cloud_db.db

SCHEME_NAME=

HOSTNAME=

PORT=0

LOGIN=

PASSWORD=

POOL_SIZE=4