Streaming

В прошлый раз разобрались как хранить, а главное передавать в web объекты. Через документные базы с Http-Rest интерфейсом CouchDB или ArangoDB. Но они (базы) плохо подходят для часто-обновляемых данных. А у нас котировки, индикаторы, события по несколько штук в секунду. Если на каждый чих обновлять документ в базе, то ничто не выдержит.

В огромном современном мире, за пределами муравейника MetaTrader, подобные данные называются ТЕЛЕМЕТРИЕЙ. И для их передачи, хранения, визуализации и обработки есть громадное количество средств.

1. Очереди сообщений

Посмотрите вокруг: дома, дороги, всякое оборудование. И везде есть датчики. Которые в реальном времени делают какие-то измерения и как-то это передают. Для этого используют "очереди сообщений". Приборы посылают (публикуют) данные, получатели (подписчики) их получают. Промежуточные сервера обеспечивают хранение/целостность/доставку.

Опять-же таких протоколов чуть более чем один, мы будем рассматривать самый простой и самый распространённый. Кто занимался самоделками а-ля "Умный дом" или "домашняя метеостанция", вот точно его знают.

Встречаем с фанфарами: MQTT (Message Queue Telemetry Transport) http://mqtt.org

Устроен дивно просто: публикатор публикует произвольные данные в топик (тему), подписчики на неё подписывается и получает данные. Темы организуются в виде иерархий. Опционально последнее переданное может сохранятся на сервере.

2. Устанавливаем брокер MQTT

Забыл сказать - все сервера MQTT называются сладким для трейдеров словом "брокер" :-) И между брокерами настраиваются бриджи - кто кому и какие пересылает данные.

Выбираем. На странице https://mqtt.org/software/ в разделе Servers/Brokers их очень много. Протокол широко известен и применяется в индустрии.

Следуя принципу "чем проще тем лучше", возьмём Mosquitto. https://www.mosquitto.org/ . Во первых он очень маленький, реальный москит. Системные требования стремятся к 0 - он работает на устройствах с мизером памяти. И при этом полностью реализует Mqtt-over-Websocket - мы можем получать данные в реальном времени прямо на http+js страницу.

Классика: загрузили, установили, работает..

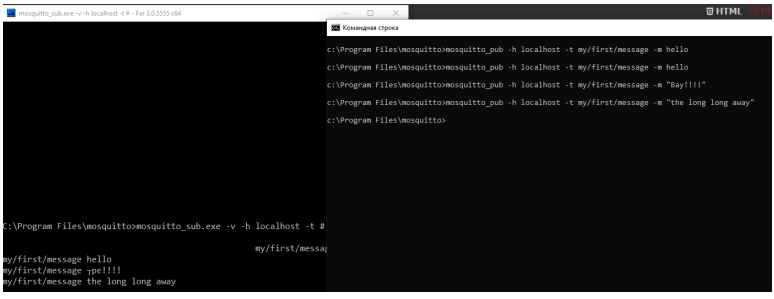
2.1. Проверяем, заодно и знакомимся

Открываем ДВА терминала. В обоих терминалах перехоим в каталог установки Mosquitto (по умолчанию он C:/Program Files/mosquitto) `cd "c:/Program Files/mosquitto"`

первый терминал будет играть роль подписчика, набираем команду : `mosquitto_sub -v -h localhost -t #` . И пока ничего не наблюдаем. Никто ведь ничего пока не шлёт

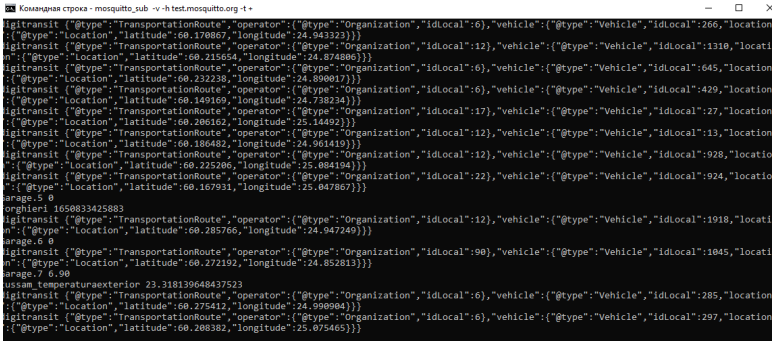
второй терминал будет играть роль источника/публикатора, набираем `mosquitto_pub -h localhost -t my/first/message -t hello`

Шайтан ! работает



Чтобы вас убедить что оно всё работает очень быстро и с очень большими объёмами, подключимся к тестовому серверу mosquitto.

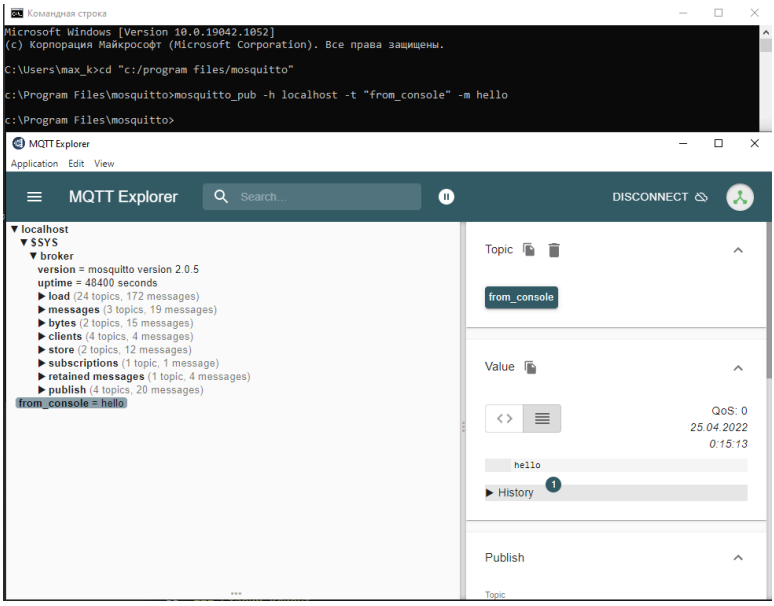
В любом из открытых пока ещё терминалов наберём команду: `mosquitto_sub -v -h test.mosquitto.org -t #`



этот вал данных - это просто кто-то тестирует свой софт. Сервер test.mosquitto.org доступен для тестовых целей.

2.2. Ставим клиент

Для удобства поставим GUI клиент. Которых опять-же много, но на мой взгляд наиболее удобный это MQTT Explorer (http://mqtt-explorer.com/) . Можете кстати сравнить по весу с брокером mosquitto :-) Вот они современные технологии !



вот так вот он выглядит, я ему даже чего-то написал из консоли

2.3. Читаем доки

Перед дальнейшими шагами, крайне рекомендую переключиться на чтение оф. документации

- Рекомендованные ссылки с mqtt.org https://mqtt.org/getting-started/
- Команды и настройки брокера mosquitto https://www.mosquitto.org/documentation/

3. Цепляем к странице

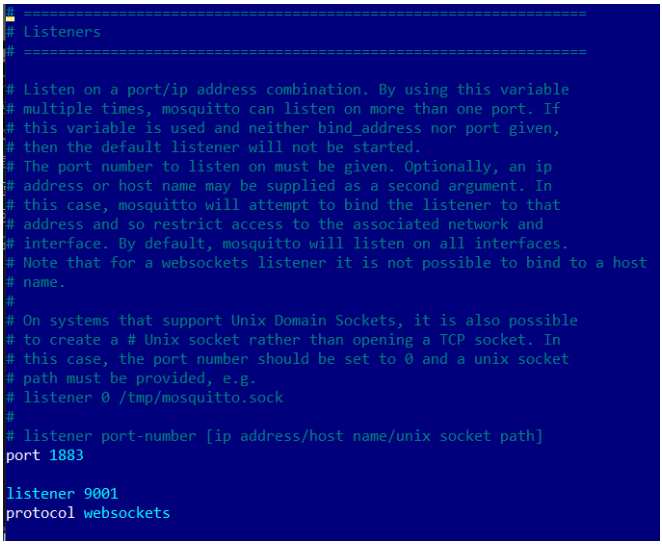
Сразу после установки наш mqtt брокер работает сам по себе, для своих клиентов и только по стандартному mqtt протоколу. Чтобы включить интерфейс WebSocket его надо чуть-чуть поднастроить

3.1. Настройка WebSocket в Mosquitto

Конфигурация у нас лежит в c:/Program Files/mosquitto/mosquitto.conf ; Открываем в редакторе и на основе прочитанной ранее документации вбиваем две строчки.

```
listener 9001
protocol websockets
```

Весть конфиг получится :



все прочие строчки в конфиге закоментированы. Рестартуем сервис mosquitto и переходим к следующему шагу

3.2. Тестовая html страница

Из громадного обилия инструкций mqtt+js выбираем https://www.wut.de/e-577ww-07-apus-000.php , там всё очень детально, красиво и по строчкам рассмотрено.

Иллюстративно, совсем маленькая страница: ничего не рисует, а просто всё выводит в журнал

```
<!doctype html>
<html lang="de">
 <head>
  <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.min.js"></script>
<!--- copy-paste кода уважаемых немцев с https://www.wut.de/e-577ww-07-apus-000.php -->
<!--- ОСТАВЛЕН ТОЛЬКО ВЫВОД В КОНСОЛЬ --->
<script>
var hostname = "localhost";         // сервер
var port = 9001;                    // порт WebSocket
var clientId = "webio4mqttexample"; // произвольный ClientId
clientId += new Date().getUTCMilliseconds();;   // должен быть уник.
 поэтому добавлено время
```

```
var username = "";                  // можно и login+pass
var password = "";
var subscription = "#";             // подписываемся на все топики

// создаём клиент, задаём колбеки
mqttClient = new Paho.MQTT.Client(hostname, port, clientId);
mqttClient.onMessageArrived = MessageArrived;
mqttClient.onConnectionLost = ConnectionLost;
Connect();

/*Initiates a connection to the MQTT broker*/
function Connect(){
 mqttClient.connect({
 onSuccess: Connected,
 onFailure: ConnectionFailed,
 keepAliveInterval: 10,
 userName: username,
 useSSL: false, // SSL возможен, но мы ещё не настраивали
 password: password});
}

/*Callback for successful MQTT connection */
function Connected() {
 console.log("Connected");
 mqttClient.subscribe(subscription);
}

/*Callback for failed connection*/
function ConnectionFailed(res) {
 console.log("Connect failed:" + res.errorMessage);
}

/*Callback for lost connection*/
function ConnectionLost(res) {
 if (res.errorCode !== 0) {
  console.log("Connection lost:" + res.errorMessage);
  Connect();
 }
}

/*Callback for incoming message processing */
function MessageArrived(message) {
 console.log(message.destinationName +" : " + message.payloadString);
}
</script>
<!--- --->
```

```
</head>

 <body>
        <h1>Press F12 for see JS debug console</h1>
 </body>
</html>
```

открываем в броузере localhost/mqtt_client.html , а через MQTT Explorer чего-нить публикуем. Всё опубликованное отлично принимается на страницу и выводится в журнал.

Треть работы сделана - инфрастуктура передачи real-time данных развёрнута и проверена.

4. MetaTrader и MQTT

Для меня лично самое простое решение - использовать проект ATcl, там на борту уже есть и mqtt клиент и сервер. Почти любые скрипты tcl могут работать внутри mql.

Другое правильное и быстрое решение - написать на C/C++ DLL-ку, практически обёртку и её спокойно использовать. У mqtt очень простые клиентские библиотеки, всё пишется легко.

Но мы не ищем лёгких путей, будем считать что робот не имеет прав использовать DLL. К тому-же решения C/C++ запрещены на mql5.com

Будем выкручиваться через локальный веб сервер. Всё одно он у нас есть, обеспечивает обращения MetaTrader к нестандартным портам

4.1. Запускаем CGI на локальном сервере

В старые-старые времена CGI (Common Gateway Interface) https://ru.wikipedia.org/wiki/CGI эксплуатировался вовсю. Сейчас для генерации страниц на стороне сервера используются php, node.js и python.

Логика будет следующей:

- MetaTrader через WebRequest методом POST обращается например на `localhost/cgi-bin/mqtt?{topic}` и отдаёт {данные}

- веб-сервер запускает промежуточную программу, которая в свою очередь запускает `mosquitto_pub -h localhost -t {topic} -m {данные}`

Почему не писать сразу на php ? во первых в демонстрационных целях, **"как запустить программу из MetaTrader, если DLL запрещены"**; во вторых вдруг вы предпочитаете node.js а я тут с php :-) И наконец - из CGI всегда можно сделать FastCGI и для случая когда "надо запускать внешнюю программу" будет работать быстрее.

4.2. Разрешаем CGI на локальном сервере

у меня сервер apache, соответсвенно открываю его конфиг и правлю. В httpd.conf нахожу секцию относящуюся к CGI и разрешаю, плюс указываю какие файлы можно запускать

```
#
# "${SRVROOT}/cgi-bin" should be changed to whatever your ScriptAliased
# CGI directory exists, if you have that configured.
#
<Directory "${SRVROOT}/cgi-bin">
    AllowOverride None
    Options +ExecCGI
    Require all granted
    AddHandler cgi-script .bat .tcl .cgi
</Directory>
```

у вас будет нечто подобное

теперь пишем CGI, можно на любом знаком языке, но иллюстративно пусть будет bat

publish.bat.

```
@echo off
chcp 65001 > NUL
if "%REQUEST_METHOD%" == "GET" (
 echo Content-Type: text/plain
 echo.
 set
 goto ok
)
```

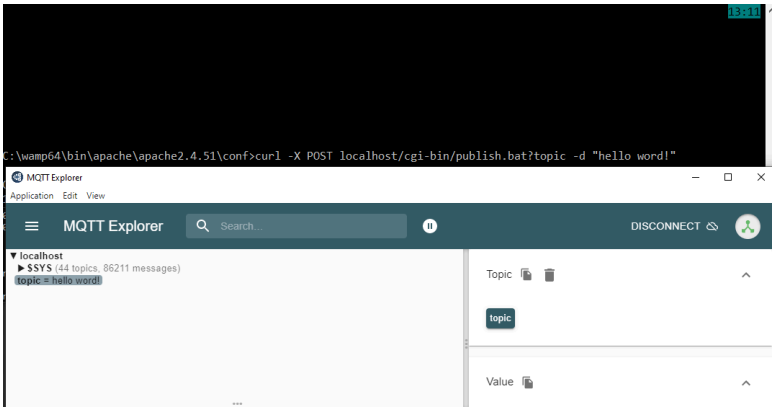
```
set /p MSG=
if "%REQUEST_METHOD%" == "POST" (
 echo Content-Type: text/plain
 echo.
 "c:\program files\mosquitto\mosquitto_pub" -h localhost -r -t
 "%QUERY_STRING%" -m "%MSG%"
 echo ok
 goto ok
)
if "%REQUEST_METHOD%" == "PUT" (
 echo Content-Type: text/plain
 echo.
 "c:\\program files\\mosquitto\\mosquitto_pub" -h localhost -t
 "%QUERY_STRING%" -m "%MSG%"
 echo ok
 goto ok
)
:ok
```

много чего нехватает, куча проверок должна быть. И минус - bat воспримет только первую строку из переданных ему данных. Как заставить работать опцию mosuitto_pub -s в этой ситуации, честно не разобрался. Без возможности -s, получены текущие ограничения:

- данные должны помещаться в одну строку
- данные не должны быть больше 1024 символа
- не должны содержать "

Главное что оно работает:

```
curl -X PUT localhost/cgi-bin/publish.bat?topic -d "hello word !"
```



мы обратились к локальному серверу http, и опубликовали сообщение. Которое получат все подписчики.

4.3. Переходим к MQL

раз уже научились публиковать, решаем что будем публиковать.

чтобы было совсем просто и демонстративно :

- mybot/candles/snapshot будет содержать последние 100 свечей и публиковаться при открытии бара
- mybot/candles/update будет содержать обновление последней свечи
- mybot/trend/snapshot какая-то линия
- mybot/trend/update

4.4. пишем код

mqtt_stream.mql.

```
#property copyright "Maxim A.Kuznetsov"
#property link      "https://www.luxtrade.tk"
#property version   "1.00"

input int MA_PERIOD=20;
input ENUM_MA_METHOD MA_MODE=MODE_LWMA;
input ENUM_APPLIED_PRICE MA_APPLIED=PRICE_MEDIAN;

datetime time0=0;          // время открытия бара текущего ТФ
int trend=INVALID_HANDLE;  // хендл индикатора тренда
```

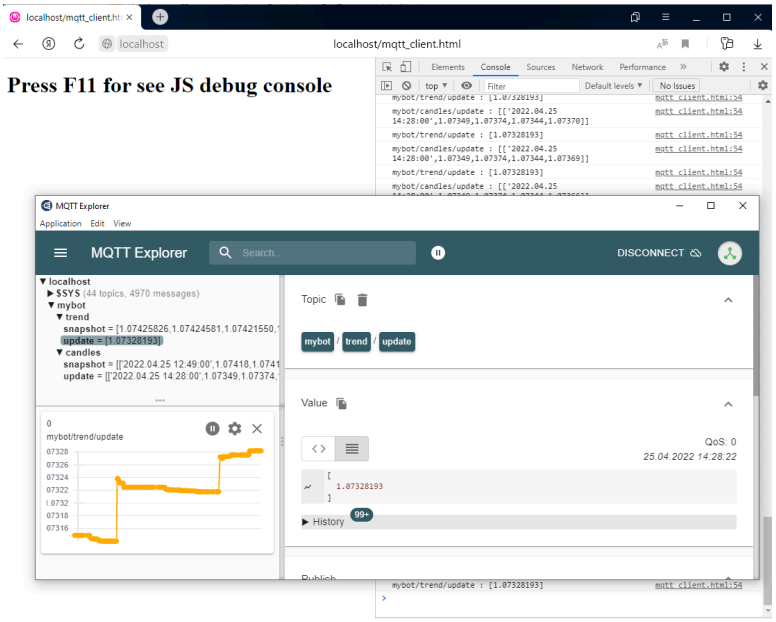
```
int OnInit()
{
   time0=0;
   trend=iMA(_Symbol,PERIOD_CURRENT,MA_PERIOD,0,MA_MODE,MA_APPLIED);
   if (trend==INVALID_HANDLE) {
      return INIT_FAILED;
   }
   EventSetTimer(60);
   return(INIT_SUCCEEDED);
}

void OnDeinit(const int reason)
{
   EventKillTimer();
}
void OnTick()
{
   datetime tmp=iTime(_Symbol,_Period,0);
   if (tmp!=0 && tmp!=time0) {
      time0=tmp;
      OnBar();
   }
   MqlRates curr[];
   if (CopyRates(_Symbol,_Period,0,1,curr)==1) {
      Publish("mybot/candles/update",curr);
   }
   double data[];
   datetime time[];
   if (CopyTime(_Symbol,_Period,0,1,time)==1 &&
 CopyBuffer(trend,0,0,1,data)==1) {
      Publish("mybot/trend/update",time,data);
   }
}
void OnBar()
{
   int size=20;
   MqlRates curr[];
   datetime time[];
   if (CopyRates(_Symbol,_Period,0,size,curr)==size) {
      Publish("mybot/candles/snapshot",curr,true);
   }
   double data[];
   if (CopyTime(_Symbol,_Period,0,size,time)==size &&
 CopyBuffer(trend,0,0,size,data)==size) {
      Publish("mybot/trend/snapshot",time,data,true);
```

```
   }
}
string TimeToISO(datetime time)
{
   return IntegerToString(1000*(long)time);
   string iso=StringFormat("%04d-%02d-%02dT%02d:%02d:
%02d",dt.year,dt.mon+1,dt.day+1,dt.hour,dt.min,dt.sec);
   return iso;
}
void Publish(string topic,MqlRates &rates[],bool retain=false)
{
   string text="";
   int total=ArraySize(rates);
   text+="[";
   for(int i=0;i<total;i++) {
      if (i) text+=",";
      text+="[";
      text+=TimeToISO(rates[i].time);
      text+=",";
      text+=DoubleToString(rates[i].open,_Digits);
      text+=",";
      text+=DoubleToString(rates[i].high,_Digits);
      text+=",";
      text+=DoubleToString(rates[i].low,_Digits);
      text+=",";
      text+=DoubleToString(rates[i].close,_Digits);
      text+="]";
   }
   text+="]";
   Publish(topic,text,retain);
}

void Publish(string topic,datetime &time[],double &data[],bool
 retain=false) {
   string text="";
   int total=ArraySize(data);
   if (total>ArraySize(time)) total=ArraySize(time);

   text+="[";
   for(int i=0;i<total;i++) {
      if (i) text+=",";
      text+="[";
      text+=TimeToISO(time[i]);
      text+=",";
      text+=DoubleToString(data[i]);
      text+="]";
```

```
   }
   text+="]";
   Publish(topic,text,retain);
}
void Publish(string topic,string text,bool retain=false) {
   string headers="Content-Type: text/json";

   char data[];
   StringToCharArray(text,data,0,WHOLE_ARRAY,CP_UTF8);
   int len=ArraySize(data);
   if (len<2) return;
   ArrayResize(data,len-1);

   char results[];
   string result_headers;
   int code=WebRequest(retain?"POST":"PUT","http://localhost/cgi-bin/
publish.bat?"+topic,headers,500,data,results,result_headers);
   if (code!=200) {
      PrintFormat("Result code=%d %s",code,result_headers);
   }

}
```

Даже не знаю что про код прокоментировать, кроме того что он максимально простой.

но результативный:



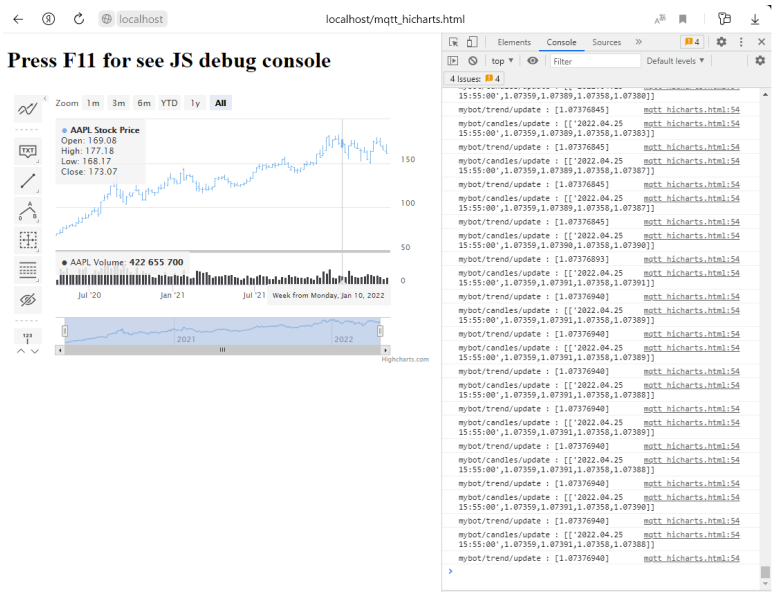
мы легко, этим элементарным кодом отдаём данные в реальном времени. На веб-страницу в том числе.

5. Возвращаемся к веб-страничке

Терминал данные передаёт, "наш брокер" их транслирует, на страничку данные поступают. Осталось только сделать выёбистую графику.

Будем рисовать нормальный чарт. Из достойных альтернатив - TradingView и HiCharts. По функционалу они почти равновесны. Выбираю исключительно по степени документированности, поэтому HiCharts (https://www.highcharts.com)

Методом copy-paste график из официальной демки https://www.highcharts.com/demo/stock/stock-tools-gui переносится на нашу страничку. Запускаем, смотрим, любуемся:



работает навороченный чарт, постоянно приходят свежие данные. Но пока-что чарт и данные не соедененны.

Что будем менять (и про что читать маны):

- в чарте уберём переключение таймфреймов - у нас их нет, робот на конкретном выбранном ТФ
- переименуем (или по образу-подобию) переделаем серию AAPL - у нас будет MyBot
- уберём серию "Volume" - мы её не транслируем
- добавим серию "Trend" - вот её-то мы транслируем
- добавим гордый Watermark - фоновую крупную подпись, чтобы визуально отличаться от обычных чартов

Эти изменения фактически декларативные - просто меняем настройки чарта глядя в документацию и примеры.

И к этому надо будет написать пару-трйку функций :

- при получении данных в топик mybot/candles/snapshot заменяем всю таймсерию.

- при получении данных в топик mybot/candles/update обновляем последнюю свечу
- аналогично с mybot/trend/snapshot mybot/trend/update

Как водится, изучаем доументацию. Самый важный для нас момент "работа с живыми данными" https://www.highcharts.com/docs/working-with-data/live-data ; Традиционно есть два способа - polling то есть периодический опрос и внешнее/отдельное соединение, как у нас.

Для работы с внешним соединением, то есть из отдельных кол-беков, необходимо сделать объект chart глобальным для страницы. Чтобы к нему можно было обращаться из любого места js. Тогда всё просто - получаем данные от Mqtt, возможно немного преобразуем и обращаемся к chart.series[x] через интерфейс (класс) https://api.highcharts.com/class-reference/Highcharts.Series#setData

Не лишний раз замечу - у HighCharts великолепная документация и примеры. Эталон для подражания.

итого, получается .mqqt_hicharts2.html

```
<!doctype html>
<html lang="ru">
 <head>
  <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/
libs/paho-mqtt/1.0.1/mqttws31.min.js"></script>

<!-- JSON5 CDN -->
<!-- это чтобы разбирать/парсить 'не-вполне' корректные json -->
<script src="https://unpkg.com/json5@^2.0.0/dist/index.min.js"></script>

<!--- copy-paste кода уважаемых немцев с https://www.wut.de/e-577ww-07-
apus-000.php -->
<!--- ОСТАВЛЕН ТОЛЬКО ВЫВОД В КОНСОЛЬ --->
<script>
var hostname = "localhost";         // сервер
var port = 9001;                    // порт WebSocket
var clientId = "webio4mqttexample"; // произвольный ClientId
clientId += new Date().getUTCMilliseconds();;   // должен быть уник.
 поэтому добавлено время
var username = "";                  // можно и login+pass
var password = "";
```

```
var subscription = "#";             // подписываемся на все топики

// создаём клиент, задаём колбеки
mqttClient = new Paho.MQTT.Client(hostname, port, clientId);
mqttClient.onMessageArrived = MessageArrived;
mqttClient.onConnectionLost = ConnectionLost;

/*Initiates a connection to the MQTT broker*/
function Connect(){
 mqttClient.connect({
 onSuccess: Connected,
 onFailure: ConnectionFailed,
 keepAliveInterval: 10,
 userName: username,
 useSSL: false, // SSL возможен, но мы ещё не настраивали
 password: password});
}

/*Callback for successful MQTT connection */
function Connected() {
 //console.log("Connected");
 mqttClient.subscribe(subscription);
}

/*Callback for failed connection*/
function ConnectionFailed(res) {
 //console.log("Connect failed:" + res.errorMessage);
}

/*Callback for lost connection*/
function ConnectionLost(res) {
 if (res.errorCode !== 0) {
  //console.log("Connection lost:" + res.errorMessage);
  Connect();
 }
}

/*Callback for incoming message processing */
function MessageArrived(message) {
 console.log(message.destinationName +" : " + message.payloadString);
 var topic=message.destinationName;
 var data=JSON5.parse(message.payloadString);
 console.log("json : " + JSON5.stringify(data));
 if ( topic == "mybot/candles/snapshot" )  OnCandlesSnapshot(data);
 else if (topic == "mybot/candles/update") OnCandlesUpdate(data);
 else if (topic == "mybot/trend/snapshot") OnTrendSnapshot(data);
```

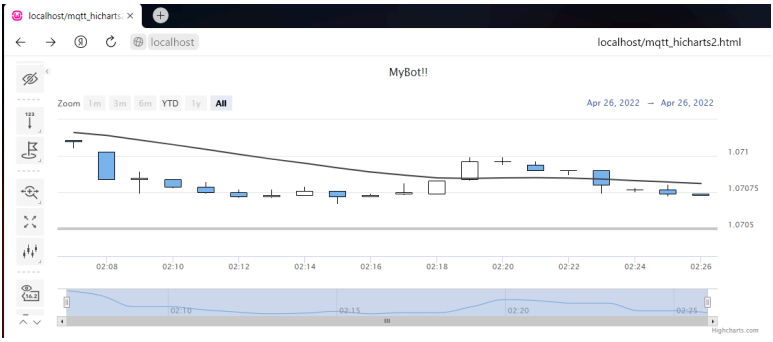
```
 else if (topic == "mybot/trend/update")   OnTrendUpdate(data);
}
</script>
<!-- StockCharts CDN -->
<link rel="stylesheet" type="text/css" href="https://
code.highcharts.com/css/stocktools/gui.css">
<link rel="stylesheet" type="text/css" href="https://
code.highcharts.com/css/annotations/popup.css">

<script src="https://code.highcharts.com/stock/highstock.js"></script>
<script src="https://code.highcharts.com/stock/modules/data.js"></
script>

<script src="https://code.highcharts.com/stock/indicators/indicators-
all.js"></script>
<script src="https://code.highcharts.com/stock/modules/drag-panes.js"></
script>

<script src="https://code.highcharts.com/modules/annotations-
advanced.js"></script>
<script src="https://code.highcharts.com/modules/price-indicator.js"></
script>
<script src="https://code.highcharts.com/modules/full-screen.js"></
script>

<script src="https://code.highcharts.com/modules/stock-tools.js"></
script>

<script src="https://code.highcharts.com/stock/modules/heikinashi.js"></
script>
<script src="https://code.highcharts.com/stock/modules/
hollowcandlestick.js"></script>

<!-- /StockCharts -->
<!-- StockCharts JS -->
<script>
let chart; // Глобальный объект чарта

window.addEventListener('load', function () {
    chart = new   Highcharts.stockChart('container', {
        title: {
            text: 'MyBot!!'
        },
        yAxis: [{
        labels: {
            align: 'left'
```

```
        },
        height: '80%',
        resize: {
            enabled: true
        }
        }, {
        labels: {
            align: 'left'
        },
        top: '80%',
        height: '20%',
        offset: 0
        }],
        tooltip: {
        shape: 'square',
        headerShape: 'callout',
        borderWidth: 0,
        shadow: false,
        positioner: function (width, height, point) {
            var chart = this.chart,
            position;

            if (point.isHeader) {
            position = {
                x: Math.max(
                // Left side limit
                chart.plotLeft,
                Math.min(
                    point.plotX + chart.plotLeft - width / 2,
                    // Right side limit
                    chart.chartWidth - width - chart.marginRight
                )
                ),
                y: point.plotY
            };
            } else {
            position = {
                x: point.series.chart.plotLeft,
                y: point.series.yAxis.top - chart.plotTop
            };
            }

            return position;
        }
        },
        series: [{
```

```
        type: 'ohlc',
        id: 'mybot-ohlc',
        name: 'MyBot',
        data: []
        }, {
        type: 'line',
        id: 'mybot-trend',
        name: 'MyBot Trend',
        data: []
        }],
        responsive: {
        rules: [{
            condition: {
            maxWidth: 800
            },
            chartOptions: {
            rangeSelector: {
                inputEnabled: false
            }
            }
        }]
        }
    }); // new HighCharts

    Connect();  // connect to MQTT

}); // window.addEventListener
</script>
<script>
// связываем события от MQTT и обновление данных чарта
function OnCandlesSnapshot ( data ) {
    chart.series[0].setData(data);
}
function OnCandlesUpdate(data) {
    var count=chart.series[0].data.length;
    chart.series[0].removePoint(count-1,false);
    chart.series[0].addPoint(data[0]);
}
function OnTrendSnapshot ( data ) {
    chart.series[1].setData(data);
}
function OnTrendUpdate(data) {
    var count=chart.series[0].data.length;
    chart.series[1].removePoint(count-1,false);
    chart.series[1].addPoint(data[0]);
}
```

```
</script>
<!-- -->
</head>
 <body>
        <div id="container" class="chart"></div>
 </body>
</html>
```

200 строк html+js, 150 строк mql , и мега-bat в 30. и выглядит это всё вот так:



Цель достигнута - MT транслирует данные в реальном времени на веб страницу.