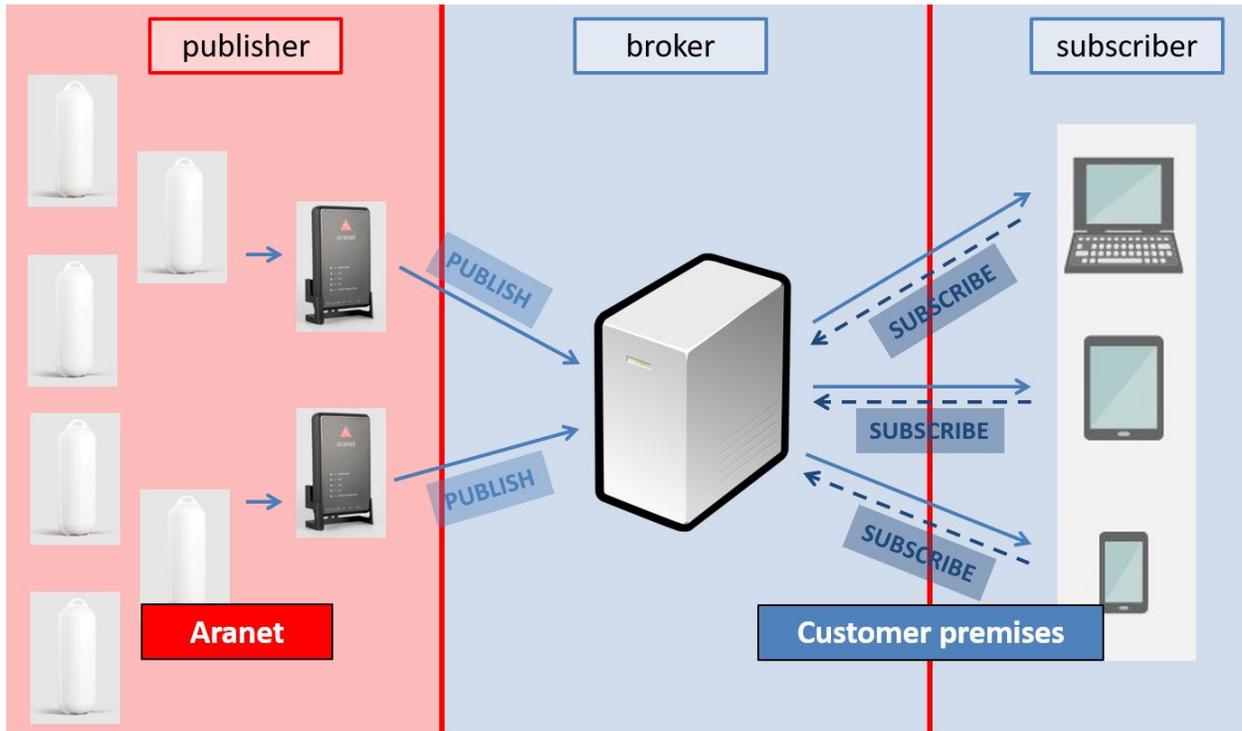


Aranet MQTT functionality and integration with Amazon AWS and Microsoft Azure

1. General MQTT network structure:



2. MQTT message format

Sensor measurement data messages from the PRO base can be published on the MQTT broker in 3 following formats (hierarchy):

1) raw

in topic structure `<root topic name>/<PRO base serial number>/sensors/<sensor ID>/measurements/<measurement type>` where

- `<root topic name>` - Aranet PRO base station MQTT message identification name which should be configured on the base MQTT page Root topic field. For more details see below [Aranet PRO base station configuration interface](#)
- `<PRO base serial number>` - serial number of PRO base station;
- `<sensor ID>` - 6 HEX digit sensor ID where the first digit is the sensor segment (for details see [Segments for sensors](#) document) and remaining 5 digits are from sensor marking from the physical label on the sensor body which can be seen also in PRO base station graphical user interface;

- d) **<measurement type>** can be one of the following:
- temperature** – data is given in degrees C (Celsius);
 - humidity** – relative humidity data is given in percentage %;
 - co2** – carbon dioxide concentration level data given in ppm(parts per million);
 - co2Abc** – shows whether CO2 manual (**0**) or automatic (**1**) calibration mode is enabled for the sensor;
 - atmosphericpressure** – atmospheric pressure data are given in Pa (Pascal);
 - voltage** – data are given in V (Volts);
 - current** – electric current data given in A (Ampere);
 - weight** – tarred weight in kg (kilogram);
 - weight raw** – untarred weight in kg (kilogram);
 - illuminance** – data from LUX sensor given in lx (lux);
 - distance** – data are given in m (meters);
 - vwc** -volumetric water content data of soil/substrate given as a fraction of one whole;
 - bec** – bulk electric conductivity data are given in S/m (Siemens per meter);
 - pec** - pore water electrical conductivity data are given in S/m (Siemens per meter);
 - dp** - dielectric permittivity data of soil or substrate given in absolute numbers;
 - ppfd** - photosynthetic photon flux density data are given in micromol/(m²s) (micromol per square meters multiplied by seconds);
 - pulses** – periodic pulses in absolute numbers;
 - pulsescumulative** - cumulative pulses in absolute numbers;
 - co** – carbon monoxide concentration level data are given in ppm (parts per million);
 - differentialpressure** – data are given in Pa (Pascal);
 - derived** – derived measurements in user-defined units;
 - rsi** – received signal strength data given in dBm;
 - battery** – battery charge level which is given as a fraction of one whole;
 - time** – measurement time in Unix epoch format:
<https://www.freeformatter.com/epoch-timestamp-to-date-converter.html>

Additionally measurement units for the sensor data according to measurement type is published in topics: `<root topic name>/<PRO base serial number>/sensors/<sensor ID> /measurements/<measurement type>/units`

```

▼ broker.hivemq.com
  ▼ Aranetest
    ▼ 394260700033
      ▼ sensors
        ▼ 100051
          productNumber = TDSPT001
          ▼ measurements
            ▼ humidity = 42.0
              units = %
            ▼ temperature = 19.950
              units = C
            ▼ rssi = -74
              units = dBm
            time = 1618671102
            ▼ battery = 0.07
              units = /

```

2) JSON

in topic structure `<root topic name>/<PRO base serial number>/sensors/<sensor ID>/json/measurements`

```
▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 2021B7
          name = 2021B7
          productNumber = TDSPT306
          ▼ json
            measurements = { "temperature": "21.800", "rssi": "-74", "time": 1618671680, "battery": "0.93" }
```

3) Azure format for sensor data publishing to Azure IoT Hub platform:

```
▼ devices/349681000816/messages/events/msgType=sensorMeasurements&uid=101306
```

```
{
  "sensors": [
    {
      "uid": "101306",
      "measurements": [
        {
          "measurement": "humidity",
          "value": "38.0",
          "units": "%"
        },
        {
          "measurement": "temperature",
          "value": "21.850",
          "units": "C"
        },
        {
          "measurement": "rssi",
          "value": "-47",
          "units": "dBm"
        },
        {
          "measurement": "time",
          "value": 1618691111
        },
        {
          "measurement": "battery",
          "value": "0.98",
          "units": "/"
        }
      ]
    }
  ]
}
```

Sensor alarm messages from PRO base is published on the MQTT broker in following hierarchy(format):
<root topic name>/<PRO base serial number>/sensors/<sensor ID>/alarms/ +

- a. **battery/activeSince** – showing time in Unix epoch format when low battery charge alarm appeared in the sensor:

```
▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 30014E
          productNumber = TDSPC004
          name = 30014E
            ▼ alarms
              ▼ battery
                activeSince = 1618650883
```

- b. **channel/activeSince** – showing time in Unix epoch format when Aranet PRO base station recorded the event when sensor started using different radio channel than configured on the base itself:

```
▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 2021B7
          name = 2021B7
          productNumber = TDSPT306
            ▼ alarms
              ▼ channel
                activeSince = 1618673067
```

- c. **packetsLost/activeSince** – showing time in Unix epoch format when Aranet PRO base station recorded that measurement data from some sensor is not received/missing:

```
▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 101306
          ▼ alarms
            ▼ packetsLost
              activeSince = 1618673049
```

d. **errorFlags/**

- a. **value** - showing number error code value when instead of measurements error message was received from the sensor;
- b. **activeSince** - showing time in Unix epoch format when instead of the measurement error message was received from the sensors:

```
▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 1022FF
          name = 1022FF
          productNumber = TDSPT409
          ▼ alarms
            ▼ errorFlags
              value = 33
              activeSince = 1618676856
```

e. **<measurement>** - shows for which measured parameter configured alarm threshold was breached;

- a. **value** – shows measurement value that generated the alarm;
- b. **diff** – shows value by what configured alarm threshold was breached. It is positive when the upper threshold was breached and negative when the lower threshold is breached;
- c. **activeSince** - shows time in Unix epoch format when alarm threshold was breached:

```
▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 1022FF
          name = 1022FF
          productNumber = TDSPT409
          ► measurements (9 topics, 45 messages)
          ▼ alarms
            ▼ temperature
              value = 24.65
              diff = 18.25
              activeSince = 1618677267
```

Aranet PRO base station publishes also:

- 1) name that is assigned to the sensor on the Aranet PRO base station in topic <root topic name>/<PRO base serial number>/sensors/<sensor ID>/name and
- 2) product number of the sensor in topic <root topic name>/<PRO base serial number>/sensors/<sensor ID>/productNumber:

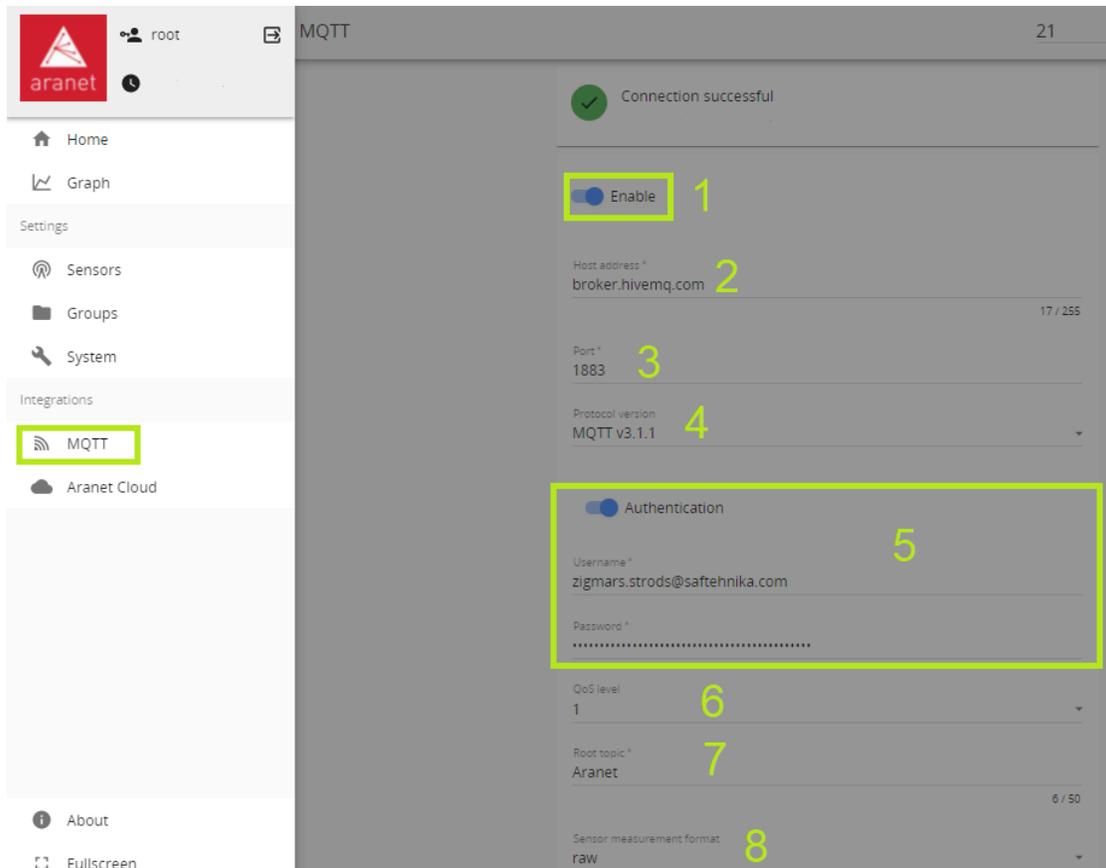
```
▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      ▼ sensors
        ▼ 2021B7
          name = Name of the sensor
          productNumber = TDSPT306
```

- 3) name of the Aranet PRO base station itself in topic <root topic name>/<PRO base serial number>/name:

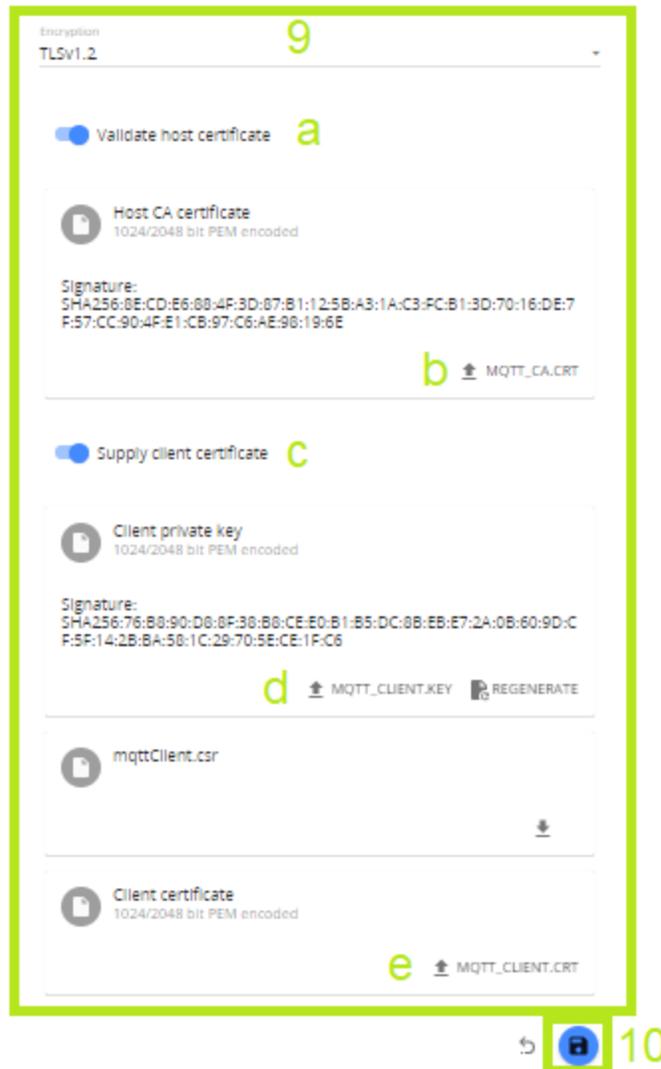
```
▼ broker.hivemq.com
  ▼ Aranet
    ▼ 349681000816
      name = Name of the Base
```

3. Aranet PRO base station configuration interface

Aranet PRO base station connection to MQTT broker is configured in the **MQTT** section of the graphical user interface. In the example below we will use configuration for connection to Hivemq public MQTT broker *broker.hivemq.com*:



- 1) **Enable** – allows enabling/disabling MQTT data transmission from Aranet PRO base station;
- 2) **Host address** – allows configuring IP address or hostname for the MQTT broker;
- 3) **Port** – allows selecting the TCP port used for the connection to the MQTT broker. The most common ports are **1883** or **8883**;
- 4) **Protocol version** – allows selecting MQTT protocol version used for connection to MQTT broker. The broker should support this version;
- 5) **Authentication** - upon necessity allows enabling additional authentication for the connection to MQTT broker and enter
 - a. **Username** and
 - b. **Password** for such connection authentication;
- 6) **QoS level** (0, 1 or 2) for MQTT message delivery on the MQTT broker can be selected as necessary <http://www.steves-internet-guide.com/understanding-mqtt-qos-levels-part-1/> and <http://www.steves-internet-guide.com/understanding-mqtt-qos-2/>;
- 7) **Root topic** – allows selecting root topic name with what MQTT messages will be published from Aranet PRO base station on MQTT broker. In our example, we will use the name **Aranet**;
- 8) **Sensor measurement format** – allows selecting format (**raw**, **JSON** or **Azure**) in which MQTT messages from Aranet PRO base station will be published on MQTT broker;

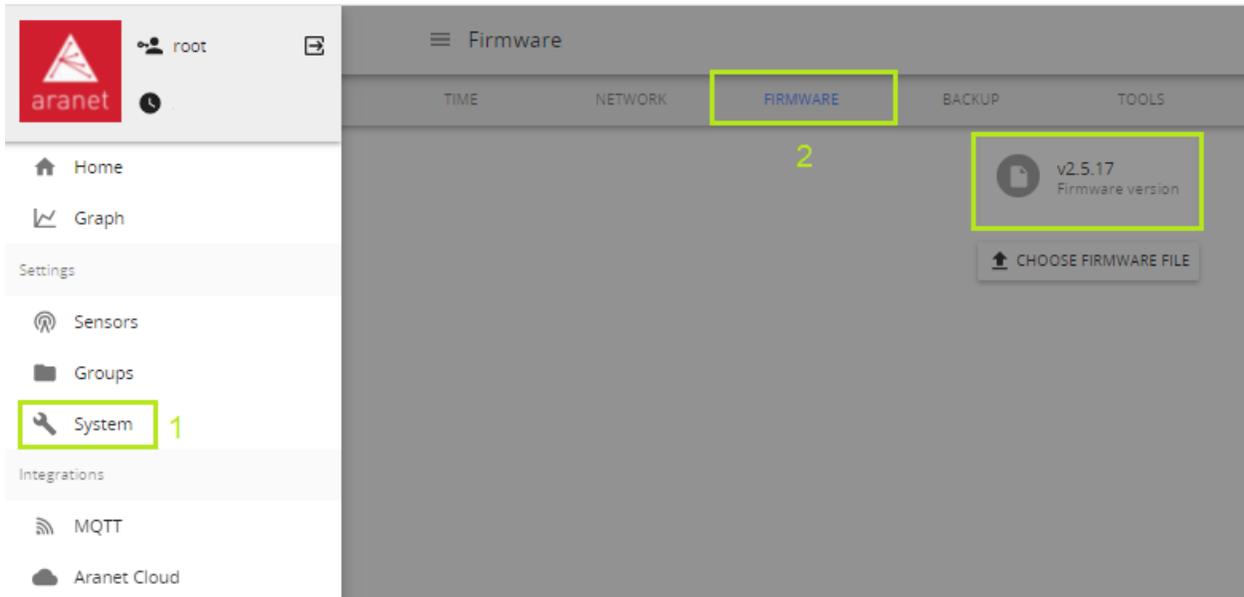


- 9) **Encryption** - upon necessity allows configuration of additional encrypted certificates (TLS version 1.1, 1.2 or 1.3) to be used for the more secure connection to the MQTT broker;
- a. **Validate host certificate** – enable to upload necessary secure connection certificates;
 - b. MQTT_CA.CRT - press to upload root CA certificate in PEM format for MQTT broker;
 - c. **Supply client certificate** - enable to upload the device public certificate and private key for secure connection to MQTT broker
 - d. MQTT_CLIENT.KEY - press to upload the Aranet PRO base station private key for secure connection to MQTT broker;
 - e. MQTT_CLIENT.CRT - press to upload the Aranet PRO base station public key for secure connection to MQTT broker
- 10) When all necessary configuration parameters are entered, they should be saved by pressing the

blue Save icon . If configured MQTT connection is successful, then **Connection successful** message will be shown on the top of the page showing also the precise time when the connection was established.

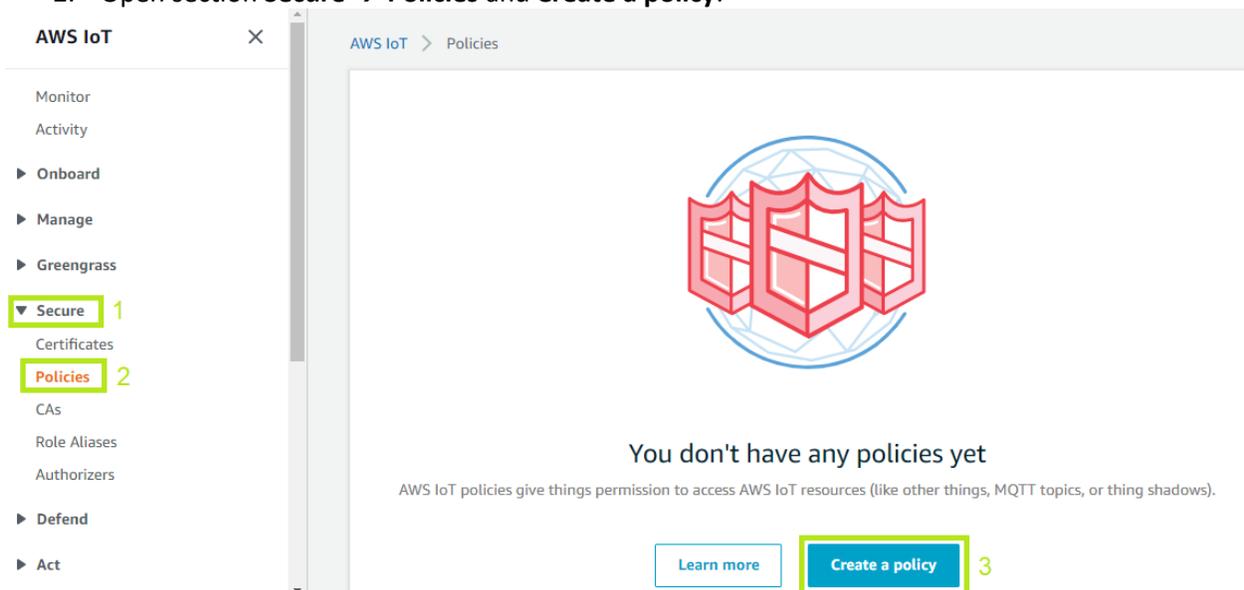
4. MQTT connection configuration with Amazon AWS platform

Aranet PRO base station allows all sensor data publishing directly to AWS IoT Core, but here base only should have firmware version at least 2.5.17. So before proceeding further, please first check the firmware version of Aranet PRO base station in the graphical user interface section **System** → **FIRMWARE** and if it is older than 2.5.17, then update to the latest version available from <https://aranet.com/downloads/> section of our webpage:



The MQTT connection configuration with the Amazon AWS platform itself can be done in the following steps:

1. Log in to Your AWS account and go to the IoT Core section of the platform;
2. Open section **Secure** → **Policies** and **Create a policy**:



3. Enter the **Name** of the policy, select **Action** as *iot.** and **Resource ARN** as ***, click to **Allow Effect** and press on **Create** button:

Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the [AWS IoT Policies documentation page](#).

Name 1

Add statements

Policy statements define the types of actions that can be performed by a resource. Advanced mode

Action 2

Resource ARN 3

Effect Allow Deny 4 Remove

Add statement

Create

4. Then go to section **Manage** → **Things** and press on **Register a thing**:

AWS IoT

- Monitor
- Activity
- ▶ Onboard
- ▼ **Manage** 1
 - Things** 2
 - Types
 - Thing groups
 - Billing groups
 - Jobs
 - Tunnels
- ▶ Greengrass
- ▼ Secure
 - Certificates
 - Policies
 - CAs
 - Role Aliases

AWS IoT > Things



You don't have any things yet
A thing is the representation of a device in the cloud.

Learn more **Register a thing** 3

5. Press on **Create a single thing**:

Creating AWS IoT things

An IoT thing is a representation and record of your physical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. [Learn more.](#)

Register a single AWS IoT thing
Create a thing in your registry

Create a single thing

Bulk register many AWS IoT things

Create things in your registry for a large number of devices already using AWS IoT, or register devices so they are ready to connect to AWS IoT.

Create many things

Cancel

Create a single thing

6. Enter the **Name** for AWS connection with Aranet PRO base station and press **Next**:

CREATE A THING STEP 1/3

Add your device to the thing registry

This step creates an entry in the thing registry and a thing shadow for your device.

Name

 1

Apply a type to this thing

Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type

No type selected Create a type

Add this thing to a group

Adding your thing to a group allows you to manage devices remotely using Jobs.

Thing Group

Groups / Create group Change

Set searchable thing attributes (optional)

Enter a value for one or more of these attributes so that you can search for your things in the registry.

Attribute key	Value	
<input type="text" value="Provide an attribute key, e.g. Manufacturer"/>	<input type="text" value="Provide an attribute value, e.g. Acme-Corporation"/>	Clear
Add another		

Show thing shadow ▾

Cancel Back Next 2

7. Press on **Create certificate**:

CREATE A THING STEP 2/3

Add a certificate for your thing

A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)

This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

Create certificate

Create with CSR

Upload your own certificate signing request (CSR) based on a private key you own.

Create with CSR

8. **Download** both **A certificate for this thing** and **A private key** on Your computer and press on **Activate**:

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	e7249df18e.cert.pem	1	Download
A public key	e7249df18e.public.key		Download
A private key	e7249df18e.private.key	2	Download

You also need to download a root CA for AWS IoT:

A root CA for AWS IoT **Download** 4

Activate 3

Cancel Done **Attach a policy**

9. After that click on **Download** link next to **A root CA for AWS IoT** and from the opened new page right click with the mouse to **Save link as...** for **Amazon Root CA 1**:

CA certificates for server authentication

Depending on which type of data endpoint you are using and which cipher suite you have negotiated, AWS IoT Core server authentication certificates are signed by one of the following root CA certificates:

VeriSign Endpoints (legacy)

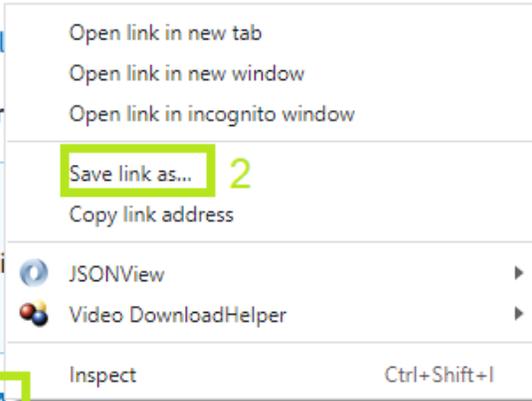
- RSA 2048 bit key: [VeriSign Class 3 Public](#)

Amazon Trust Services Endpoints (preferred)

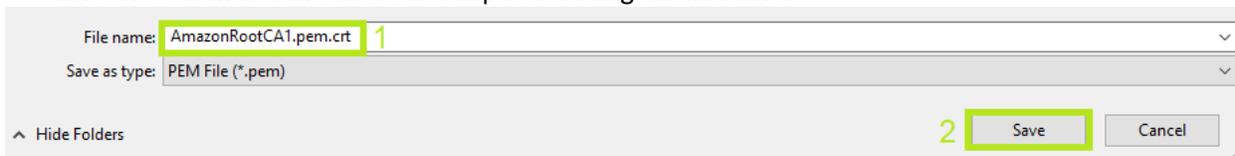
Note

You might need to right click these links and save these certificates as files.

- RSA 2048 bit key: [Amazon Root CA 1](#)
- RSA 4096 bit key: Amazon Root CA 2. Reserved for future use.



10. Save certificate file on Your computer adding extension .crt



11. Next go back to the **Certificate** page and press on **Attach a policy** button:

Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	e7249df18e.cert.pem	Download
A public key	e7249df18e.public.key	Download
A private key	e7249df18e.private.key	Download

You also need to download a root CA for AWS IoT:
A root CA for AWS IoT [Download](#)

[Activate](#)

Cancel [Done](#) [Attach a policy](#)

12. Select the previously created in step 3 and click on **Register Thing**:

CREATE A THING

Add a policy for your thing

STEP 3/3

Select a policy to attach to this certificate:

- aranet_policy_name 1 [View](#)

1 policy selected 2 [Register Thing](#)

13. Next go to section **Settings** and copy Your **AWS Endpoint** address:

AWS IoT

- Monitor
- Activity
- Onboard
- Manage
 - Things
 - Types
 - Thing groups
 - Billing groups
 - Jobs
 - Tunnels
- Greengrass
- Secure
 - Certificates
 - Policies
 - CAs
 - Role Aliases
 - Authorizers
- Defend
- Act
- Test
- Software
 - Settings** 1
 - Learn

AWS IoT > Settings

Settings

Device data endpoint Info [Refresh](#)

Your devices can use your account's device data endpoint to connect to AWS.

Each of your things has a REST API available at this endpoint. MQTT clients and AWS IoT Device SDKs also use this endpoint.

Endpoint

[a2b057iup6w5fv-abs.iot.us-east-2.amazonaws.com](#) 2

Domain configurations

You can create domain configurations to simplify tasks such as migrating devices to AWS IoT Core, migrating application infrastructure to AWS IoT Core and maintaining brand identity.

[Actions](#) [Create domain configuration](#)

Name	Domain name	Status	Service type	Date updated
No domain configurations You don't have any domain configurations.				

[Create domain configuration](#)

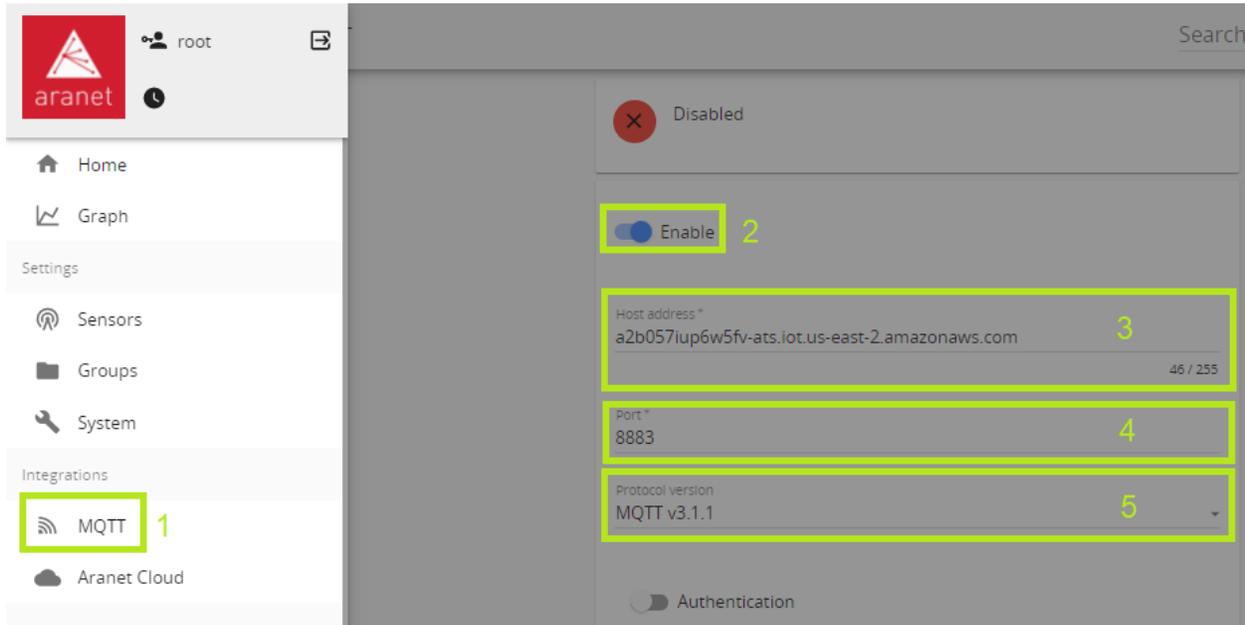
Logs

 Info [Manage logs](#)

You can manage AWS IoT logging to log helpful information to CloudWatch Logs.

As messages from your devices pass through the message broker and the rules engine, AWS IoT logs process events which can be helpful in troubleshooting.

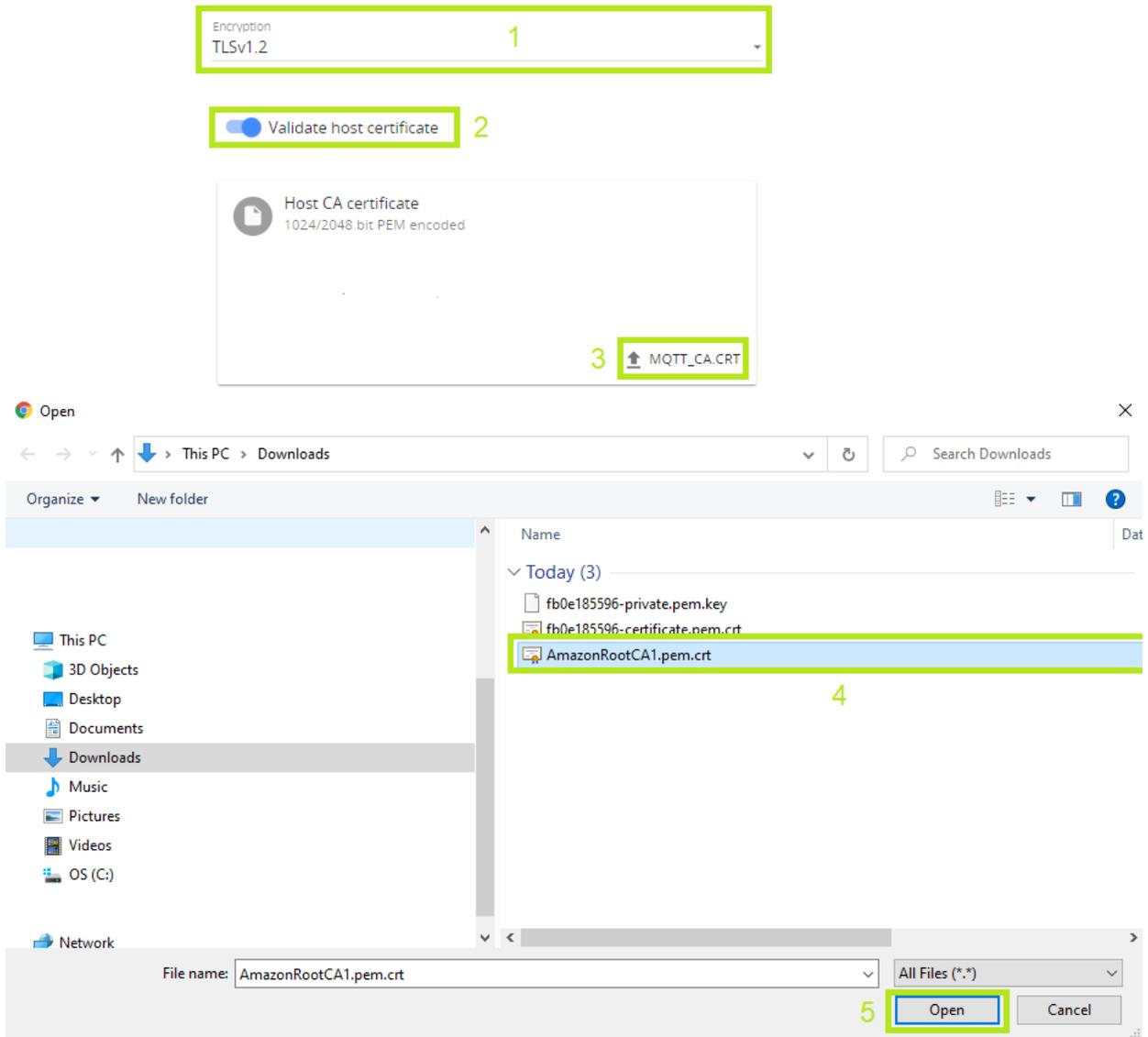
14. Now log in to Aranet PRO base station graphical user interface and go to the **MQTT** section. **Enable** MQTT connection and paste previously copied AWS Endpoint information in **Host address** field. Enter **Port = 8883** and choose **Protocol version = MQTT v3.1.1** without **Authentication**:



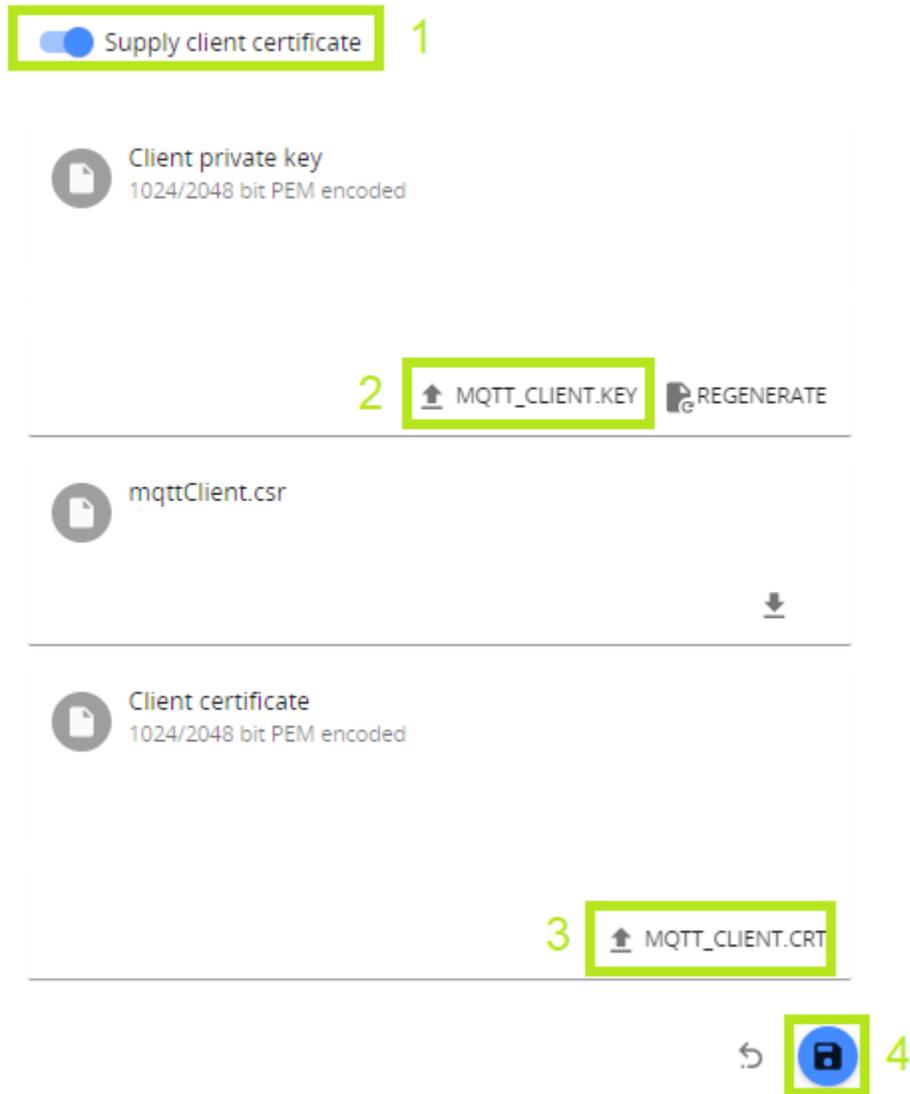
15. Next set necessary **QoS level (0 or 1, but 2 is not supported by AWS)** for MQTT message delivery in AWS system, their **Root topic** and **Sensor measurement format (raw or JSON)**

The screenshot shows the MQTT configuration page with three fields highlighted by green boxes and numbered. The first field is 'QoS level' with a dropdown menu set to '1', highlighted with a green box and a green '1'. The second field is 'Root topic' with a text input field containing 'Aranet', highlighted with a green box and a green '2'. The third field is 'Sensor measurement format' with a dropdown menu set to 'raw', highlighted with a green box and a green '3'.

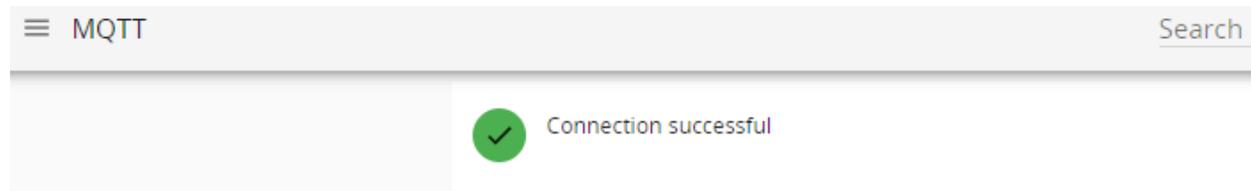
16. Then select **Encryption = TLSv1.2**, enable **Validate host certificate** and clicking on **MQTT_CA.CRT** upload **AmazonRootCA1.pem.crt** from step 9:



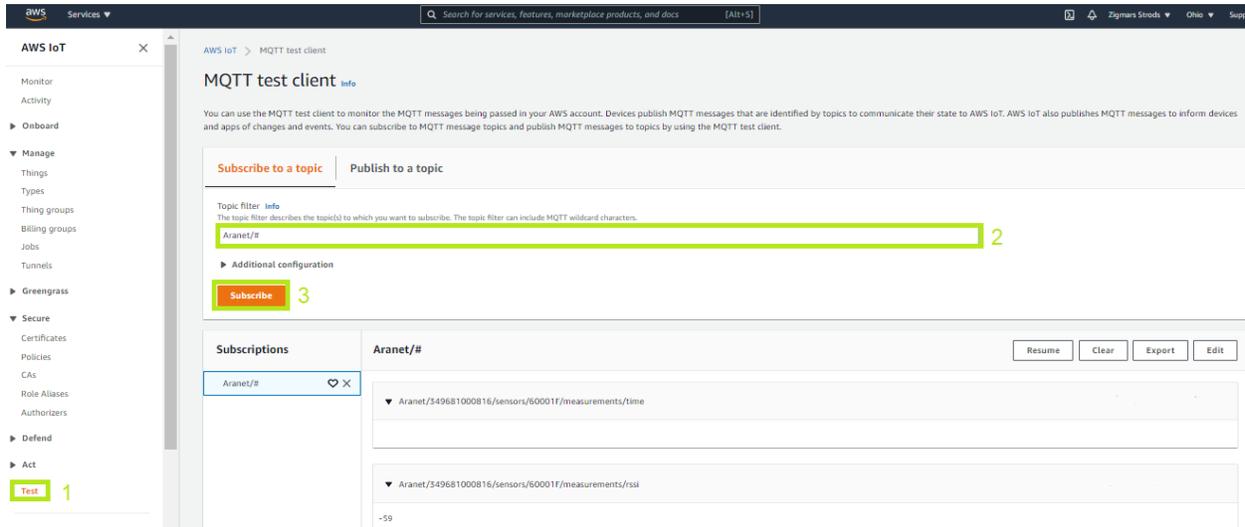
17. Finally enable **Supply client certificate**, clicking on **MQTT_CLIENT.KEY** upload *...-private.pem.key* file, but clicking on **MQTT_CLIENT.CRT** upload *-certificate.pem.crt* file and then press on **Save** icon:



And now Aranet PRO base station MQTT connection to AWS Core IoT service should be enabled and sensor data should be published on the AWS platform. Connection success should be indicated on the top of the MQTT page of the Aranet Pro base station:

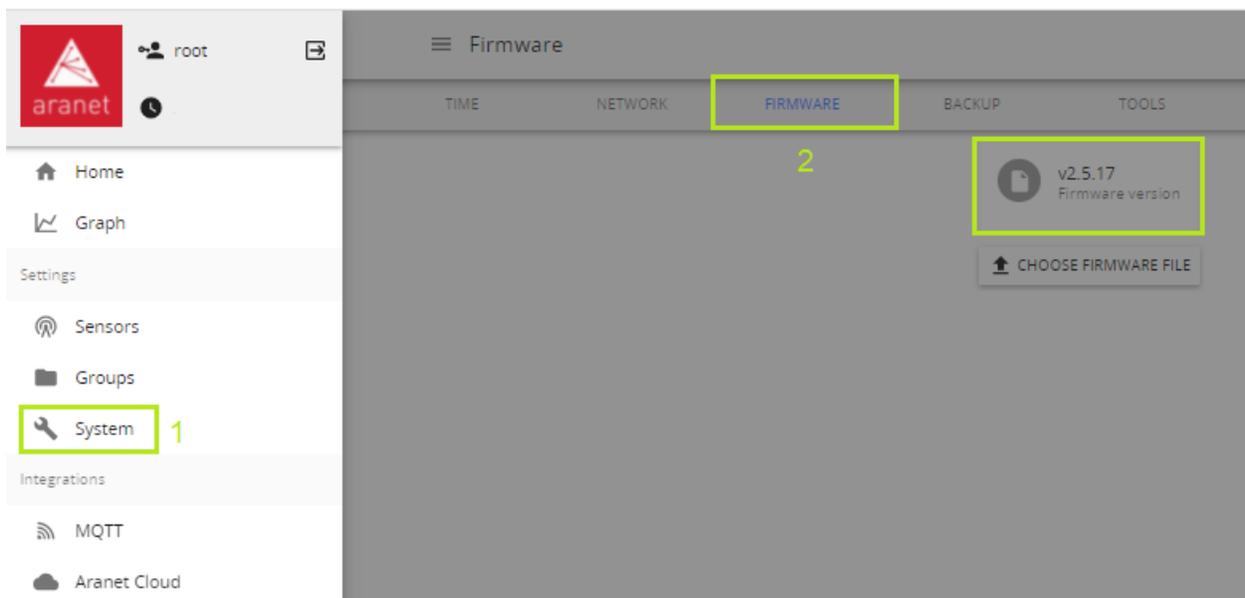


From the AWS platform data reception could be checked from section **Test** entering the Root or any other relevant topic for the previously configured Aranet PRO base station (in our example, **Aranet/#**) and pressing on **Subscribe** button:

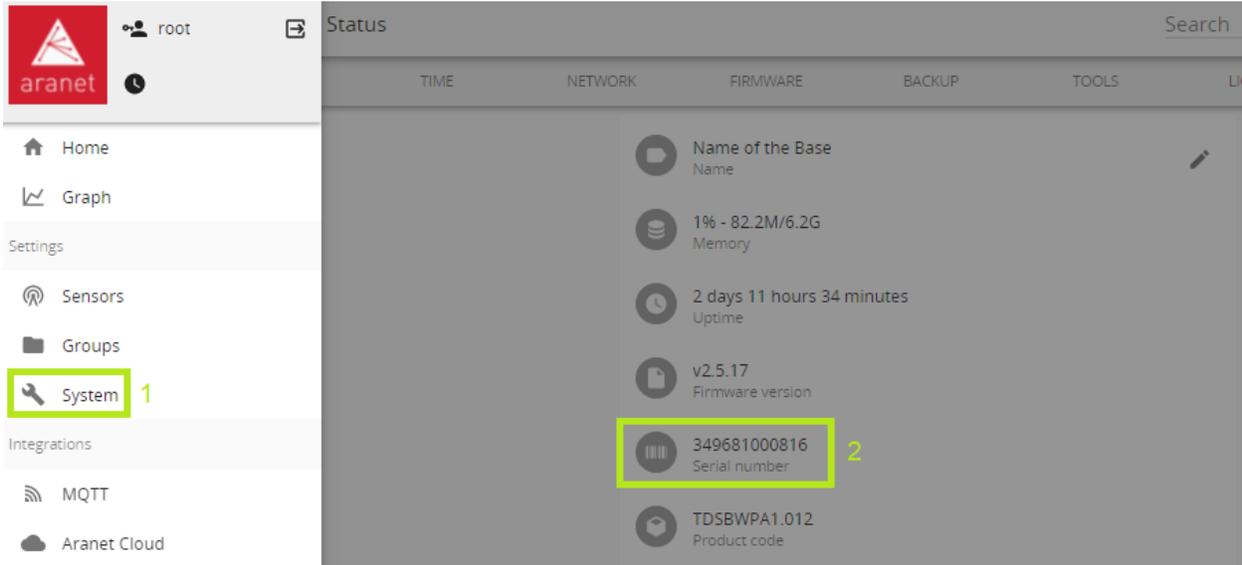


5. MQTT connection configuration with Azure IoT Hub

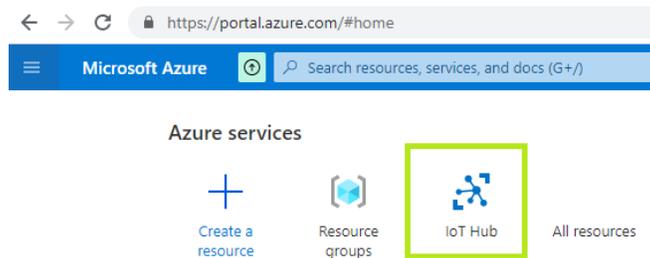
Aranet PRO base station allows also all sensor data publishing directly to Azure IoT Hub, but here the base should also have the firmware version at least 2.5.17. So before proceeding further, please first check the firmware version of the Aranet PRO base station in the graphical user interface section **System** → **FIRMWARE** and if it is older than 2.5.17, then update to the latest version available from <https://aranet.com/downloads/> section of our webpage:



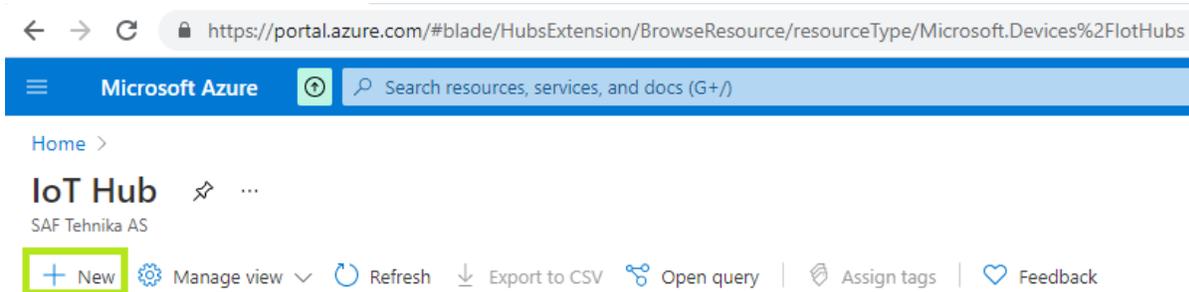
1. Next check and save for further use the **Serial number** of the Aranet PRO base station You have. It can be seen on the label on the back of the physical device or from the graphical user interface section **System** → **STATUS**:



2. Log in to Your Azure account and create a new **IoT Hub** resource:



3. Click on **New**:



- In the next window **Create new** for **Resource group**, enter the necessary **Name** and press on **OK** button:

The screenshot shows the Azure portal interface for creating a new IoT Hub. The left sidebar displays the 'IoT Hub' section for 'SAF Tehnika AS' with a 'New' button and a 'Manage view' dropdown. The main content area shows the 'IoT hub' configuration page for 'Microsoft'. The 'Project details' section includes fields for 'Subscription', 'Resource group', 'Region', and 'IoT hub name'. The 'Resource group' dropdown is open, showing '(New) Aranet' and a 'Create new' button highlighted with a green box and the number 1. A modal dialog is open, showing the 'Name' field with 'Aranet' entered, highlighted with a green box and the number 2. The 'OK' button is highlighted with a green box and the number 3.

- In the same window enter necessary **IoT hub name** and press **Review + create** button:

The screenshot shows the Azure portal interface for creating a new IoT Hub. The left sidebar displays the 'IoT Hub' section for 'SAF Tehnika AS' with a 'New' button and a 'Manage view' dropdown. The main content area shows the 'IoT hub' configuration page for 'Microsoft'. The 'Project details' section includes fields for 'Subscription', 'Resource group', 'Region', and 'IoT hub name'. The 'IoT hub name' field is filled with 'aranetpro', highlighted with a green box and the number 1. The 'Review + create' button is highlighted with a green box and the number 2.

6. Then press **Create** button again:

Microsoft Azure Search resources, services, and docs (G+)

Home > IoT Hub >

IoT Hub

SAF Tehnika AS

+ New Manage view

Filter for any field...

Name ↑↓

No IoT hub to display

Create an IoT hub to help you connect, monitor, and manage billions of your IoT assets.

[Learn more about IoT Hub](#) [Quickstart: send telemetry from device](#)

Create IoT hub

IoT hub

Microsoft

Validation passed.

Basics Networking Management Tags Review + create

Basics

Subscription	Free Trial
Resource group	Aranet
Region	East US
IoT hub name	aranetpro

Networking

Connectivity method	Public endpoint (all networks)
Private endpoint connections	None

Management

Pricing and scale tier	S1
------------------------	----

Create < Previous: Tags Next > Automation options

7. Wait for the Azure system to deploy the new resource and when it is done press on **Go to resource** button:

Microsoft Azure Search resources, services, and docs (G+)

Home >

aranetpro-4200621 | Overview

Deployment

Search (Ctrl+/) Delete Cancel Redeploy Refresh

We'd love your feedback! →

Your deployment is complete

Deployment name: aranetpro-4200621
Subscription: Free Trial
Resource group: Aranet
Correlation ID: 898475af-8c0c-43be-8e62-773985ae6b27

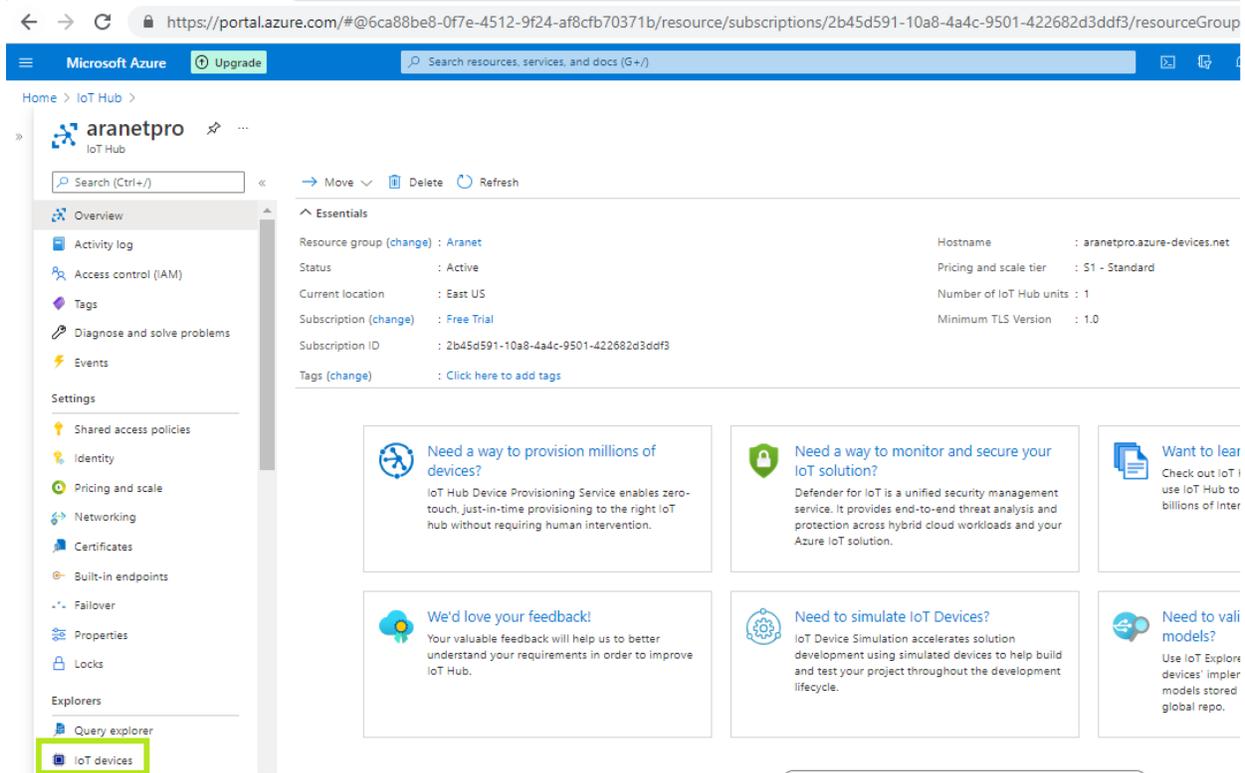
Deployment details (Download)

Next steps

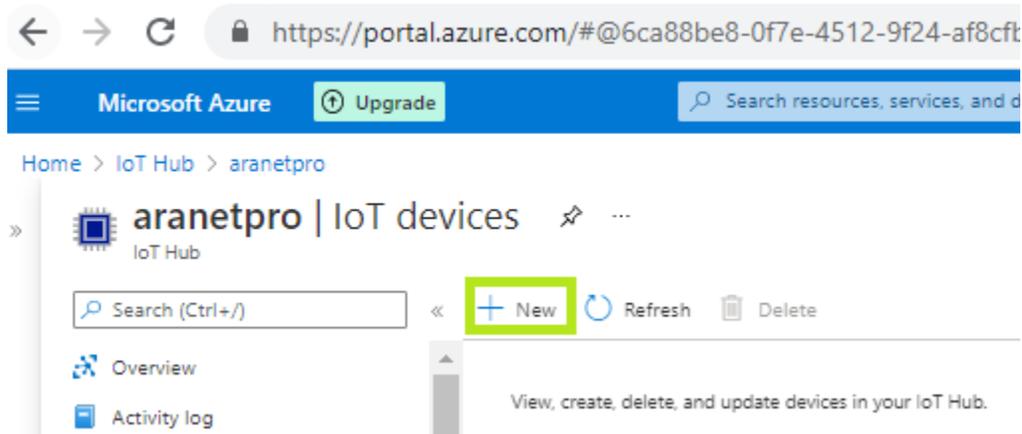
- Add and configure IoT Devices Recommended
- Configure routing rules for device messaging Recommended

Go to resource

8. Then select the **IoT devices** section from the left side menu:



9. Press on **New**:



10. Just enter or copy the serial number of the Aranet PRO base station from step 1 in the **Device ID** field without any additional symbols and characters and press the **Save** button:

Home > IoT Hub > aranetpro >

Create a device

Find Certified for Azure IoT devices in the Device Catalog

Device ID * 1

Authentication type Symmetric key X.509 Self-Signed X.509 CA Signed

Primary key *

Secondary key *

Auto-generate keys

Connect this device to an IoT hub Enable Disable

Parent device **No parent device**
[Set a parent device](#)

Save 2

11. Next go to section **Shared access policies**, click on **iothubowner** record and copy information from field **Connection string—primary key**:

Microsoft Azure Upgrade Search resources, services, and docs (G+?) zigmars.strods@safehn...

Home > IoT Hub > aranetpro

aranetpro | Shared access policies

IoT Hub uses permissions to grant access to each IoT hub endpoint. Permissions limit the access to an IoT hub based on functionality.

Policy	Permissions
iothubowner 2	registry.write, service.connect, device.connect
service	service.connect
device	device.connect
registryRead	registry.read
registryReadWrite	registry.write

Shared access keys

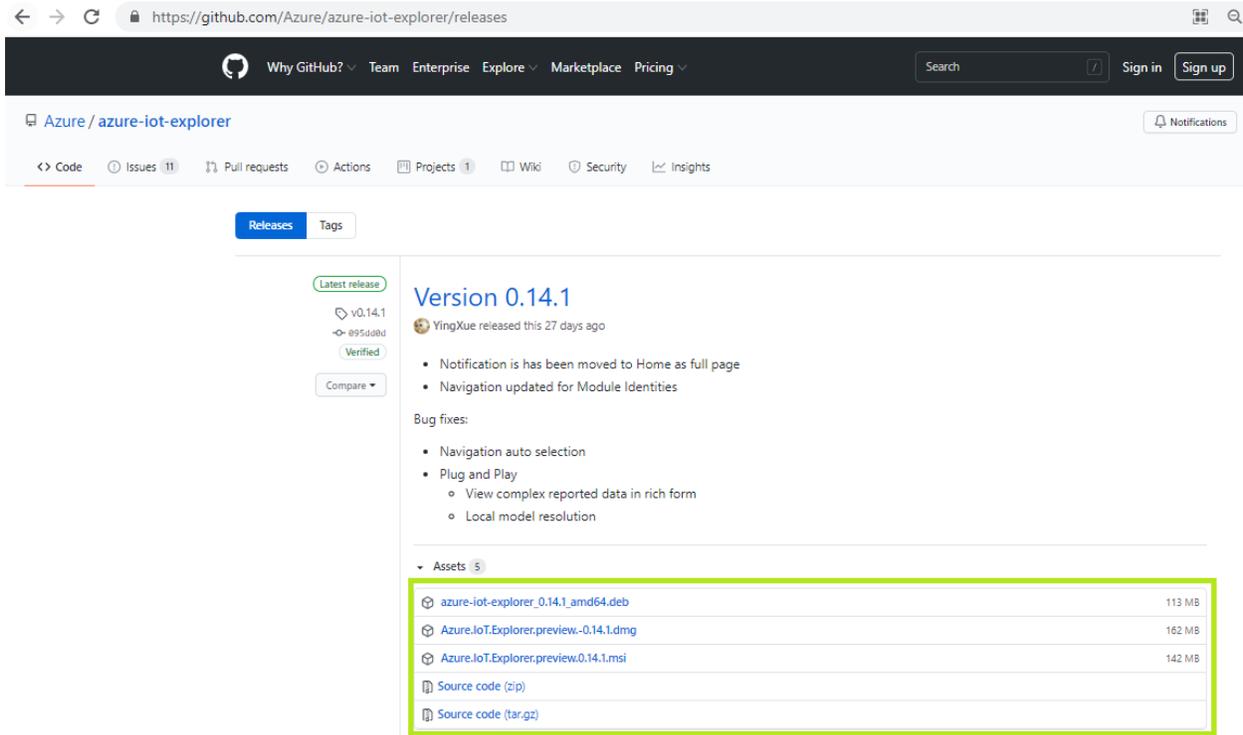
Primary key 3

Secondary key

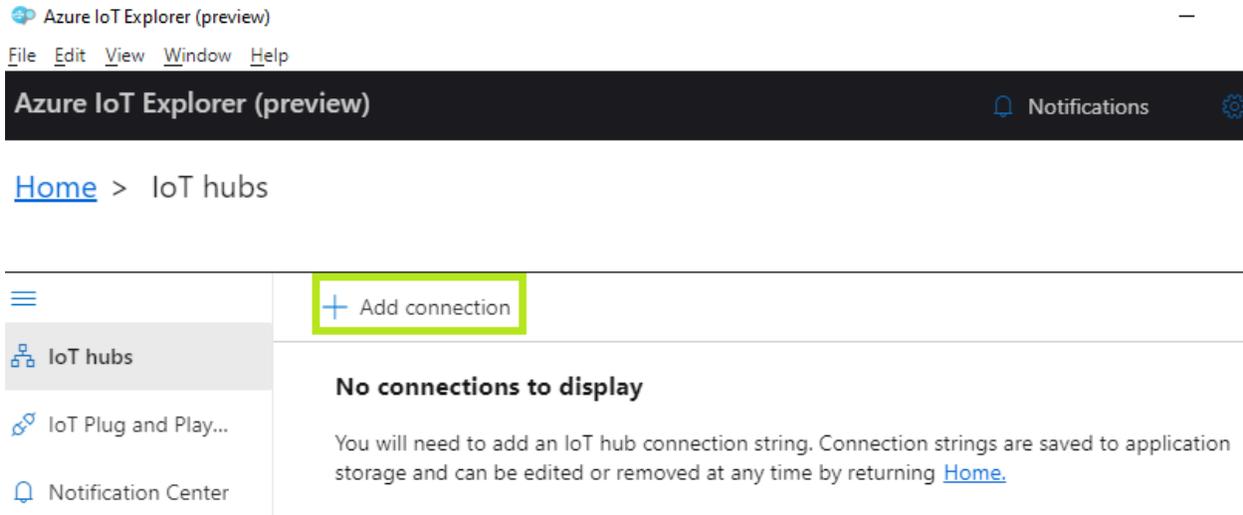
Connection string—primary key 3

Connection string—secondary key

12. Now go to <https://github.com/Azure/azure-iot-explorer/releases> and install Azure IoT explorer on Your operating system:



13. Launch Azure IoT explorer application and press on Add connection



14. In the opened window paste string from step 11 and press the **Save** button:

Azure IoT Explorer (preview)

File Edit View Window Help

Azure IoT Explorer (preview) Notifications Settings

Home > IoT hubs

+ Add connection string

No connections

You will need storage and...

Help:

Where do I get an IoT hub connection string?

Please do not save your hub connection string to any unsafe locations

Connection string *

```
HostName=aranetpro.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=ozS1cioq4
```

1

Host name

aranetpro.azure-devices.net

Shared access policy name

iothubowner

Shared access policy key

.....

2

Save Cancel

15. Now click on the previously create Aranet PRO base station IoT devices object:

Azure IoT Explorer (preview)

File Edit View Window Help

Azure IoT Explorer (preview) Notifications Settings

Home > aranetpro > Devices

+ New Refresh Delete

Query by device ID... Add query parameter

Device ID	Status	Connection st...	Authenticatio...	Last status up...	IoT Plug and ...	Edge device
349681000816	Enabled	Disconnected	Sas	--		

16. Click on **Connection string with SAS token** subsection, select **Primary key** as **Symmetric key**, enter necessary **Expiration (minutes)** time, for example, 9999999999, and press on **Generate** button and then copy generated **SAS token connection string**:

Azure IoT Explorer (preview)

File Edit View Window Help

Azure IoT Explorer (preview) Notifications Settings

Home > arantepro > Devices > 349681000816 > Device identity

Save Manage keys

Device identity

Device ID

Primary key

Secondary key

Primary connection string

Secondary connection string

Connection string with SAS token

Symmetric key *

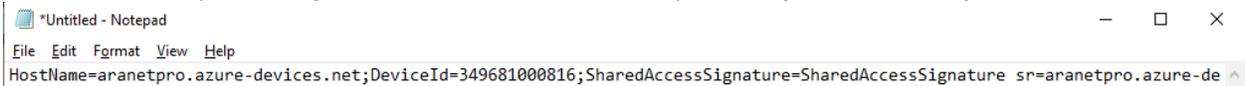
Expiration (minutes)

SAS token connection string

Generate

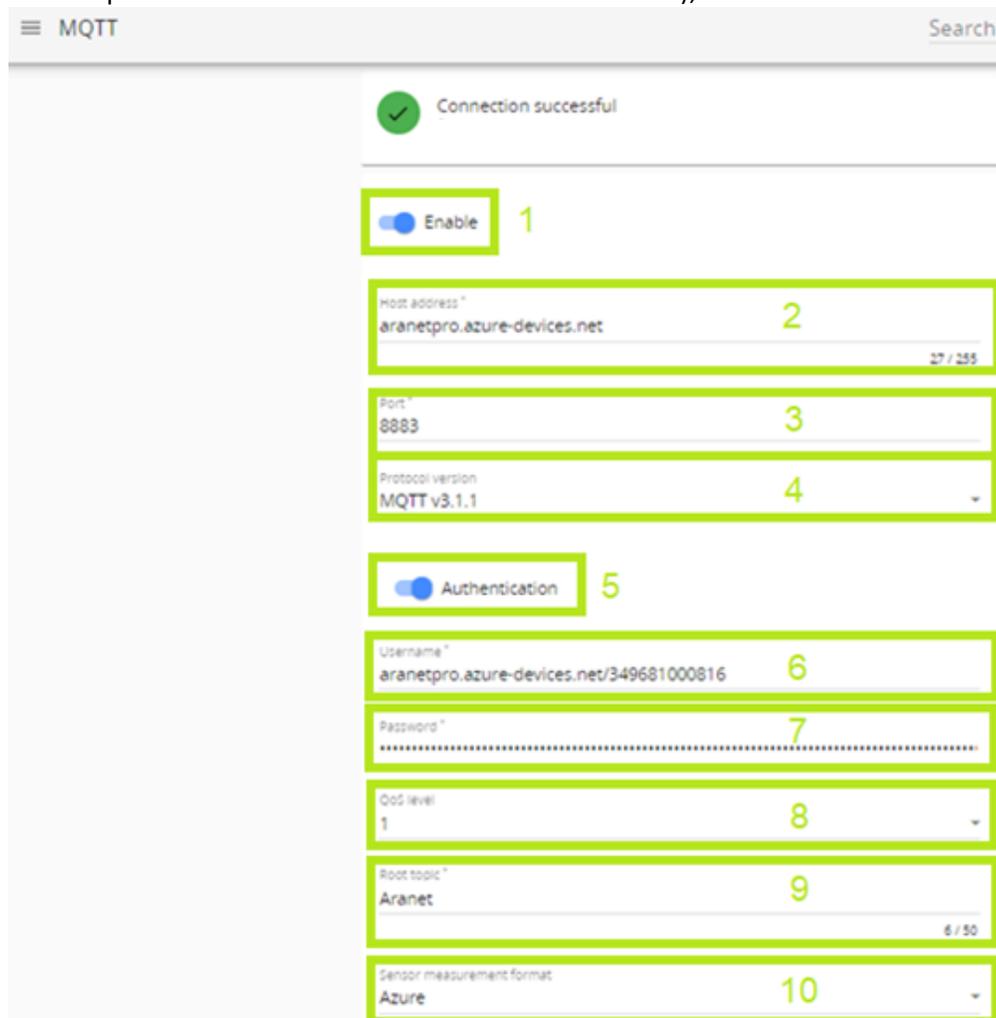
Connect this device to IoT hub
 Enable

17. Paste copied string in some text editor, for example, **Notepad** or **Microsoft Word**:



18. Now open Aranet Pro base station graphical user interface section **MQTT** and:

1. **Enable** MQTT connection;
2. paste **HostName=** value from the text editor in the **Host address** field (in our example **aranetpro.azure-devices.net**);
3. select **Port** as **8883**;
4. select **Protocol version** as **MQTT v3.1.1**;
5. **Enable Authentication**;
6. in **Username** field paste **HostName=** value again, then slash "/" and serial number of Aranet PRO base station (in our example, **aranetpro.azure-devices.net/349681000816**);
7. in **Password** field paste all the text string that comes after **SharedAccessSignature=** (in our example, **SharedAccessSignature sr=aranetpro.azure-de.... etc.**);
8. select **QoS level** as **0** or **1** (2 is not supported by Azure);
9. enter any **Root topic** value as You want;
10. select **Sensor measurement format** as **Azure** (sensor measurements will not be accepted by Azure platform if the format is selected as raw or JSON);

A screenshot of the Aranet Pro MQTT configuration interface. The interface shows a 'Connection successful' message at the top. Below it, several configuration fields are highlighted with green boxes and numbered 1 through 10, corresponding to the steps in the previous list. The fields include: 1. Enable (checkbox checked), 2. Host address (aranetpro.azure-devices.net), 3. Port (8883), 4. Protocol version (MQTT v3.1.1), 5. Authentication (checkbox checked), 6. Username (aranetpro.azure-devices.net/349681000816), 7. Password (masked with dots), 8. QoS level (1), 9. Root topic (Aranet), and 10. Sensor measurement format (Azure).

11. select **Encryption** as *TLSv1.2*;
12. and finally press on the **Save** icon:



19. If the connection to the Azure platform is successful then the corresponding success message will be shown on top of the MQTT page in Aranet PRO base station graphical user interface. Additionally user can check what data is published on the Azure IoT hub from the Azure IoT explorer application by clicking on the **Telemetry** section and then pressing on **Start** button:

