



# PAPAGO TH 2DI DO

**Environment monitor:**  
Measures temperature, humidity and  
calculates dew point  
+ two contact inputs and a relay  
Ethernet or WiFi interface  
PoE or external power



IoT Parrot		⚙️
Temperature	80 °C 20 °C	5.2 °C
Humidity	60 % 20 %	24.8 %
Dew point		-1.3 °C
Power meter	<input type="checkbox"/>	3.58 kWh ⚙️
Beer cans	<input type="checkbox"/>	42 ⚙️
Relay	<input checked="" type="checkbox"/>	SET RESET
📶 2024/06/06 13:00:42 Administrator PAPAGO 1TH 2DI 1DO WiFi v. 1.8/4 en.papouch.com		

# PAPAGO TH 2DI DO

## Datasheet

Created: 14.11.2014

Last update: 24.10 2025 12:08

Number of pages: 48

© 2025 Papouch s.r.o.

---

## Papouch s.r.o.

Address:

**Strasnicka 3164  
102 00 Prague 10  
Czech Republic**

Phone:

**+420 267 314 267**

Web:

**en.papouch.com**

Mail:

**info@papouch.com**



## TABLE OF CONTENTS

Table of contents .....	3	SNMP objects – general .....	28
Firmware versions.....	3	Traps .....	29
Getting to know Papago.....	4	MODBUS TCP .....	29
PAPAGO TH 2DI DO .....	5	Output.....	29
Connection.....	6	Input state read.....	29
Configuration .....	8	Counters .....	30
Network .....	9	Sensor .....	31
Security.....	10	SPINEL.....	32
E-mail .....	11	Temperature reading .....	32
SNMP .....	12	Read input states .....	34
HTTP GET .....	12	Read output state .....	34
Inputs and outputs configuration section ....	18	Output settings.....	35
Sensor Section .....	19	Pulse on output.....	35
Other Settings.....	20	Reading counters.....	36
Configuration via Telnet protocol.....	21	Setting or subtracting counter value .....	37
Connection .....	21	Reading of name and version .....	38
IP address is not known.....	21	Reading of manufacturing data .....	38
IP address is known.....	22	Automatic message .....	39
Telnet main menu.....	22	Indications .....	42
Server .....	22	Reset .....	43
Factory Defaults .....	23	Technical parameters .....	44
Exit without save .....	23	Integrated temperature and humidity sensor	44
Save and exit .....	23	Standalone temperature sensor .....	45
XML .....	24	Sensor cable.....	46
Units codes.....	25	Other parameters.....	46
SNMP .....	26	Default settings of the Ethernet.....	47
Objects – variables .....	27	Available designs.....	47

## Firmware versions

### 10/2025: ETH v.2.2, WiFi v.2.6

- Added instructions for timed output switching in the Spinel protocol ([Pulse on output](#)).

### 06/2024: ETH v.1.8, WiFi v.2.3

- Added MQTT communication protocol and TCP communication timeout. New START event in http get and MQTT. New responsive web. Removed GUID parameter in get.

### Version 1.05

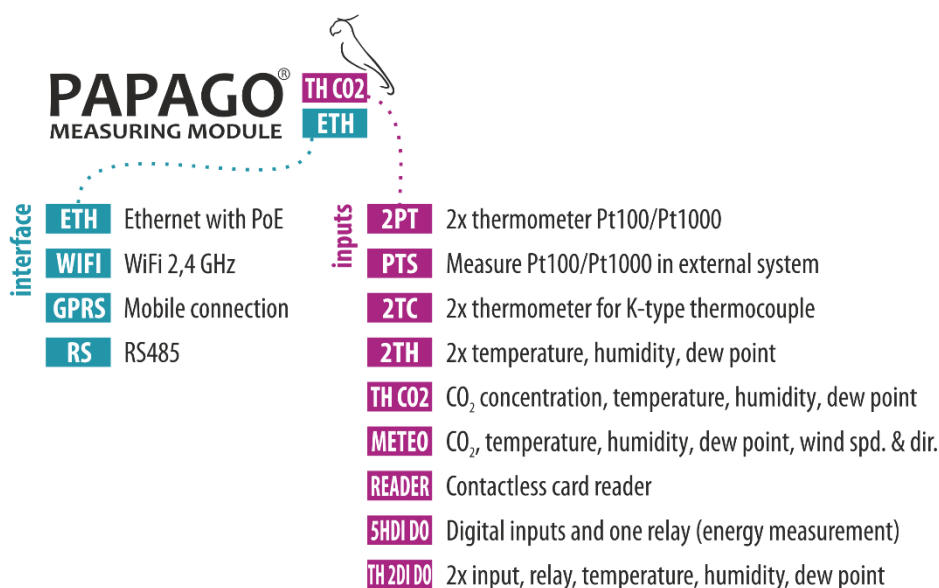
- Added support for new TH3 sensor.

### Version 1.0

- First version.

## GETTING TO KNOW PAPAGO

PAPAGO is a family of devices with uniform appearance and communication capabilities. It allows to combine communication interfaces on one side and measuring sensors (inputs) on the other side.



### Applications

- Temperature and humidity measurement in industry, buildings, server rooms and other environments.
- Measurement of temperature for heating systems.
- Monitoring temperatures in warehouses and archives.
- Monitoring the manufacturing process.
- Monitoring temperature, humidity and reached limits.
- Environmental monitoring via the Internet.
- Measurement for the HACCP system.

### Common Features

- Ethernet or WiFi interface to an internal website and many standard communication protocols.
- Ethernet versions with PoE power supply. This eliminates the need to use an external power supply, but the possibility to connect the AC adapter is available.
- Internal memory and real time backup. Measured data is stored in memory with a time stamp in case communication is lost and cannot be sent via http get request or MQTT. When the connection is restored, the data is automatically sent.
- Elegant but robust metal box that can be mounted on a DIN rail. The box bears descriptions that allow connection without having to consult the manual. Also LED indicators for all important states help commissioning.
- The possibility to display, store and analyse data in the Wix program.

### Communication Options

Depending on used interface PAPAGO has different communication options. PAPAGO can be controlled via a web interface or via software for Windows. Machine data-reading is possible using various standard methods, so PAPAGO can be easily integrated into your existing systems. You can choose the option that is appropriate for your location:

		automatic control						user control		
		MODBUS	HTTP GET	MQTT	EMAIL	SNMP	XML	SPINEL	WEB	WIX
ETH	TCP	✓	✓	✓	✓	✓	✓	✓	✓	✓
WIFI	TCP	✓	✓	✓	✓	✓	✓	✓	✓	✓

**Machine reading:** [Modbus TCP](#), [MQTT](#), [HTTP GET](#) with encryption, [e-mail](#), [SNMP](#), [XML](#), [Spinel](#)

**User control:** [Web interface](#), Wix software

## PAPAGO TH 2DI DO

PAPAGO TH 2DI DO is used as an environment monitor – it measures temperature, humidity and dew point. It also has two contact inputs to connect either contact or a pulse output and a relay.

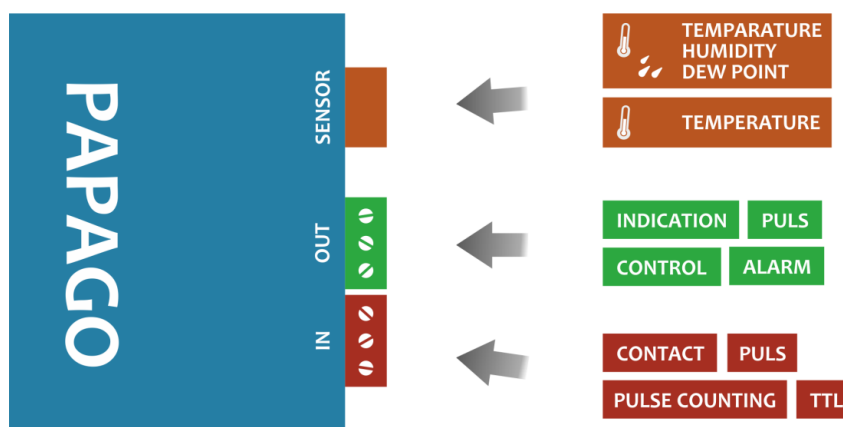


Fig. 1 – use possibilities

**Sensor input (SENSOR)** can accept one of these two types of sensors:

- ❖ Sensor [TH3](#)..... temperature: -40 to 125 °C; humidity: 0 to 100 %
- ❖ Sensor [TEMP](#)..... temperature: -55 to 125 °C

**Relay (OUT)** is meant for:

- ❖ Switching load with low voltage and power.
- ❖ Manual control (on/off or pulse), input mirroring or an alarm for measured value.

**Digital inputs (IN)** can be connected with:

- ❖ Contact.
- ❖ Pulse output from measuring gauges like electricity, water, gas, heat and other.
- ❖ Logic output of 5V TTL levels.

- Family of measuring devices with Ethernet or WiFi 2,4 GHz interface.
- User readable data via responsive web interface or Wix software.
- Machine data reading using Modbus TCP, MQTT, http get, SNMP, XML, email or Spinel (depending on the type of communication interface).
- Option to encrypt data in HTTP get with 128bit AES encryption.
- Power supply from PoE (according to IEEE 802.3af, Ethernet versions only) or from an external power supply.
- External DC power supply 11 to 58 V.
- Current consumption typically 72 mA at 24 V (depending on version).

## CONNECTION

- 1) Ethernet version: Connect the device by a normal uncrossed cable for computer networks to the switch.
- 2) Ethernet version: If the device cannot be powered by the switch via PoE according to the IEEE 802.3af standard, connect a power adapter to the coaxial connector next to the connector for the Ethernet. DC voltage in the range of 11-58 V is expected. (The positive pole is inside, the input for the power supply has reverse polarity protection.)

WiFi version: Connect a power adapter to the coaxial connector next to antenna. DC voltage in the range of 11-58 V is expected. (The positive pole is inside, the input for the power supply has reverse polarity protection.)



fig. 2 – Front view with sensor connector and input/output terminals

- 3) Sensor connector: Connect temperature, or combined temperature/humidity sensor (sold separately). Terminal contact and relay connection shown in fig. 2.
- 4) Ethernet version: Now it is necessary to set the correct IP address of the device. The default IP address is 192.168.1.254 and network mask 255.255.255.0. If your network is not compatible with this range, set the IP address of the device using [Ethernet Configurator](#).

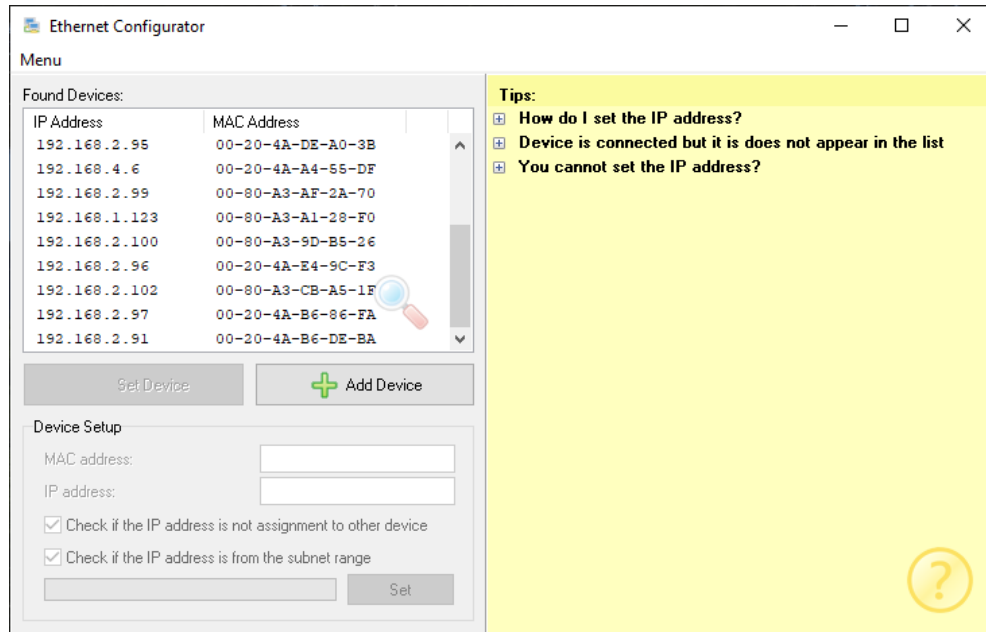


fig. 3 – Ethernet Configurator for setting the IP address

**WiFi version:** Connect your Papago to a windows PC using the supplied micro USB cable.<sup>1</sup> Run *Papago WiFi Configurator* software, you can download it on papouch.com. Set-up Papago to your WiFi network parameters so you can access it from that network.

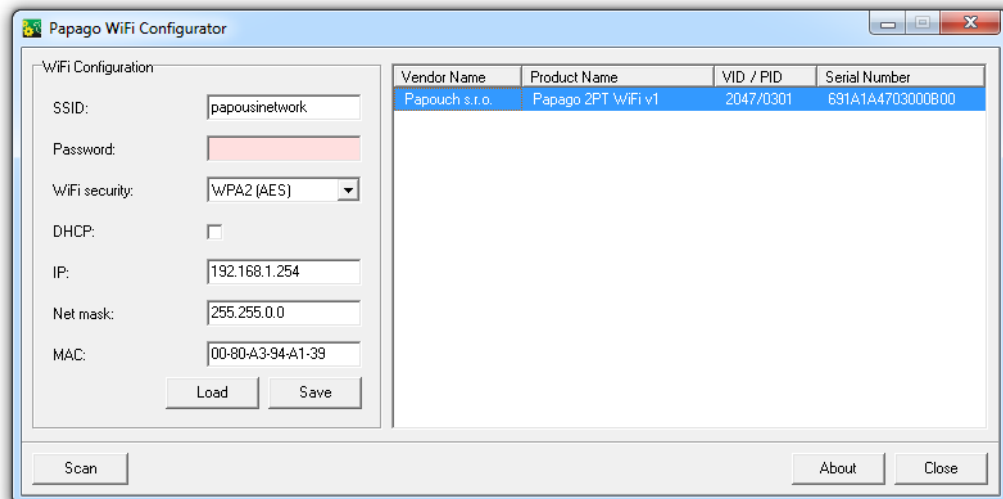


fig. 4 – WiFi configuration via USB

- 5) After setting the address, you can connect to a Web browser at the address specified as follows: <http://192.168.1.254/> (The example is given for the default IP address.)

<sup>1</sup> In Windows 7 or higher driver will be installed automatically.

**CONFIGURATION**

Configuration is done via a web interface. The basic network parameters can also be set via Telnet (see page 21). **The web interface** is accessible on the IP address of the device. (The default address is 192.168.1.254.)

After entering the IP address, the main page will appear showing the latest measured values.

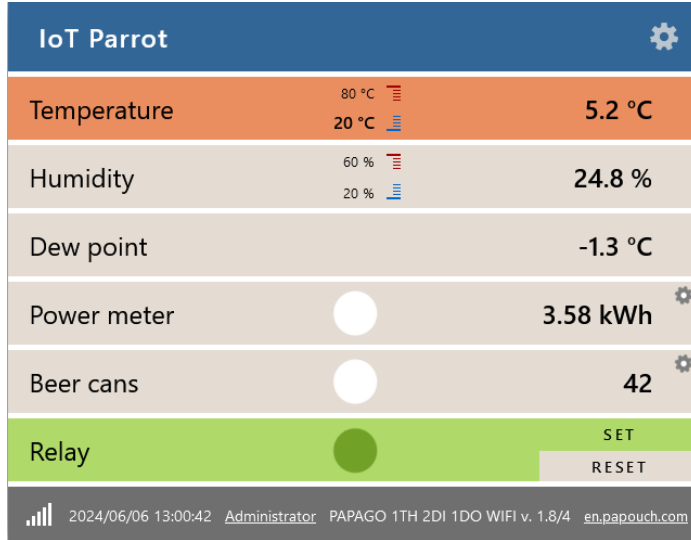


fig. 5 – Web interface with temperature-humidity sensor

**The web interface is secured** with a username and password. It is possible to choose a password separately for the user (he can only view the current values on the main page; his login name is always **user**) and separately for the administrator (he can also change settings; his login name is always **admin**).

**The configuration** is displayed by clicking on the gear symbol on the top right. The configuration is divided into sections according to the types of settings and is available in English and Czech.

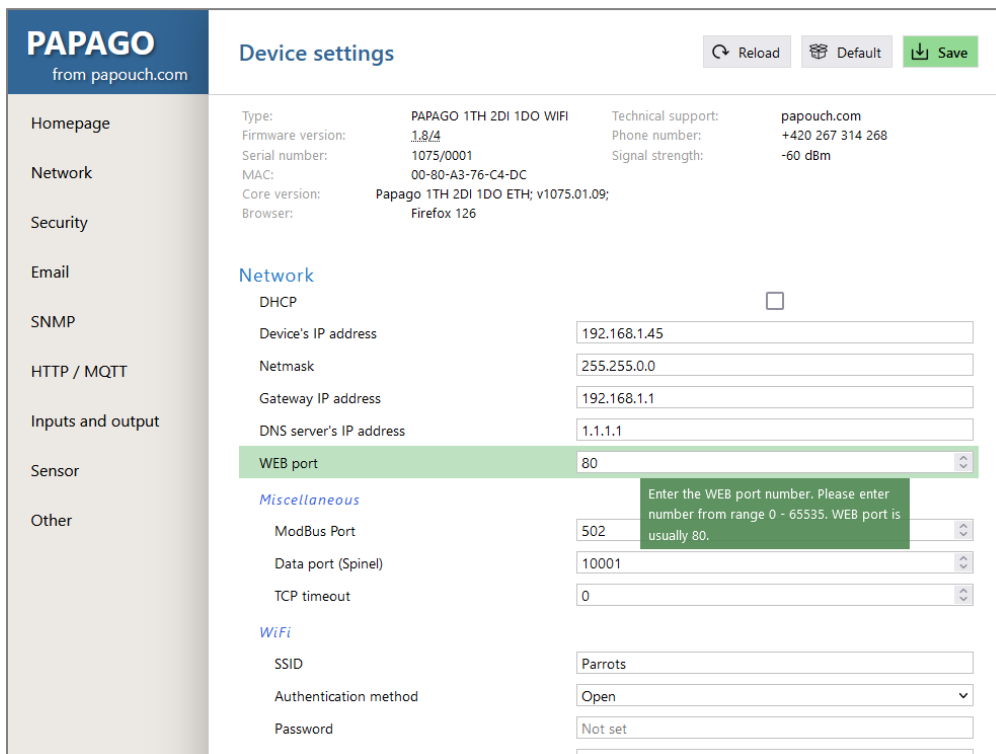


fig. 6 - Configuration of Papago

## Network

This section contains the configuration of network parameters.

### Network

DHCP	<input type="checkbox"/>
Device's IP address	<input type="text" value="192.168.1.254"/>
Netmask	<input type="text" value="255.255.255.0"/>
Gateway IP address	<input type="text" value="192.168.1.111"/>
DNS server's IP address	<input type="text" value="1.1.1.1"/>
WEB port	<input type="text" value="80"/>

### Miscellaneous

ModBus Port	<input type="text" value="502"/>
Data port (Spinel)	<input type="text" value="10001"/>
TCP timeout	<input type="text" value="60"/>

### WiFi

SSID	<input type="text" value="MyWifi"/>
Authentication method	<input type="text" value="WPA2 (AES)"/>
Password	<input type="password" value="....."/>
Re-enter key	<input type="password" value="....."/>

fig. 7 - network configuration

If the box for assigning addresses via DHCP is ticked, the fields for *Device's IP address*, *Netmask*, *Gateway IP address* and *DNS server's IP address* are reset and upon reloading the settings they are filled again with data obtained from the DHCP server.

If you have a **version with WiFi interface** in the section *Network* is also following parameters:

- As *Authentication method* is available these options: *Open*, *WEP (open)*, *WEP (shared)*, *WPA (TKIP)*, *WPA (AES)*, *WPA2 (TKIP)*, *WPA2 (AES)*, *WPA2 (Mixed)*.
- Password length is 8 to 30 characters.<sup>2</sup>

<sup>2</sup> The password can contain the following characters:

!#\$%()\*+,-./0123456789:;=?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[]^\_abcdefghijklmnopqrstuvwxyz{|}~

## Security

---

The section for setting the password of the user (can only access the main page) and the administrator (has access to both the main page and the settings).

### Security

User password	<input type="text" value="Keep original password"/>
Confirm user password	<input type="text"/>
Administrator's password	<input type="text" value="Keep original password"/>
Confirm administrator's password	<input type="text"/>
Current administrator's password	<input type="text"/>
Disable Telnet (advanced users only)	<input type="checkbox"/>
Disable fw upgrade (advanced users only!)	<input type="checkbox"/>

fig. 8 - Access security settings

After saving, the passwords are no longer displayed for security reasons. The fields for entering the password show *Not set*, if the password has not been entered, or *Keep original password*, if the password has been entered but is not to be displayed.

The user name is always 'user', the administrator name is always 'admin'. Passwords have a maximum length of 16 characters.<sup>2</sup>

If user has a password, administrator must also have a password. For security reasons, password is not displayed after saving.

Last two items - *Disable Telnet* and *Disable fw upgrade* - are only available in Ethernet version.

**Caution:** If you disable Telnet protocol and/or upgrade fw and an error occurs during a firmware upgrade, manufacturer service may be required!

## E-mail

The device can send periodic emails with the current status of the pulse counters, or if there is a change in the input or output. (The email sending feature in Papago ETH can only be used with SMTP servers that do not require SSL encrypted communication.)

### E-mail

After the event	<input checked="" type="checkbox"/>
Periodically	Once a week
Sending time	5
To	admin@example.com
From	papago@example.com
Server address	smtp.example.com
Host name	
Port	25
<b>Security &amp; Authentication</b>	
Connection security	None
Authentication method	Password, transmitted insecurely
User name	papago@example.com
User password	●●●●●●●●●●●●●●●●
Re-enter the password	●●●●●●●●●●●●●●●●
<input type="button" value="Send test mail"/>	

fig. 9 – email settings

- **After the event:** On exceeding the limits set at the sensor or on changing the input status.
- **E-mail sending:** Can be once an hour, once a day, once a week or once a month. For each of these variations, the next field *Send Time* allows you to set which minute, hour, day of the week or day of the month to send the email. The function is intended for periodic sending of the current status of energy meters, etc.
- **Connection security:** In the Ethernet interface version, only the None option is available, so it is not possible to communicate with a server requiring SSL encryption. In the WiFi interface version, the STARTTLS option is available as well.

### E-mail example:

On leaving the limits, an email is sent with the subject *Papago 1TH 2DI 1DO ETH\_info\_IoT Parrots* and the text according to this example:

```
Input 1 is 29 m3. State is: OFF
Input 2 is 0.049 kWh. State is: OFF
State Output is: OFF
Temperature is in range. Value is 26.7 °C.
Humidity is in range. Value is 61.5 %.
Dewpoint is in range. Value is 18.6 °C.
```

## SNMP

Here you can configure communication via SNMP used for data collection in large networks.

### SNMP

Enable SNMP	<input checked="" type="checkbox"/>
Enable sending traps	<input checked="" type="checkbox"/>
Traps: On change	<input checked="" type="checkbox"/>
Traps: Periodic	<input type="text" value="15"/>
SNMP manager	<input type="text" value="192.168.1.110"/>
Read community name	<input type="text" value="public"/>
Write community name	<input type="text" value="private"/>

fig. 10 - settings for communication via SNMP

The period of sending periodic traps can be 0 to 1440 minutes (1 day).

For description of SNMP objects see page 26.

## HTTP GET

In this section is set the sending of measured data to the remote server using http get requests or via MQTT. The protocol type is selected in the first item of this section - the choices are *None*, *MQTT* and *HTTP GET*.

If a message with measurement data cannot be sent, it is stored in a circular buffer with a capacity of 120 messages. The messages are then sent as soon as the connection is re-established.

### HTTP GET

Under *Mode*, select HTTP GET.

#### HTTP / MQTT

Mode	<input type="text" value="HTTP GET"/>
After the event	<input checked="" type="checkbox"/>
Periodically	<input type="text" value="1h"/>
Server address	<input type="text" value="iot.example.com"/>
Port number	<input type="text" value="80"/>
Path / Publish topic	<input type="text" value="api/papago.php"/>
Encryption Key	<input type="text" value="....."/>
Retype Key	<input type="text" value="....."/>

fig. 11 - data sending via HTTP GET

- *After the event*: When leaving the limits and when returning to the limits, the GET will be sent outside of any set send period. It should be taken into account that with rapid changes on inputs, it is not realistic for the device to send an informational HTTP GET on absolutely every change.
- *Periodically*: Period of sending the message. Time from 10 sec to 24 hours. Enter zero to disable sending. The data can be entered in whole minutes (5m, 20m, etc.), hours (1h, 12h, etc.) or seconds.

- *Encryption key*: If an encryption key is entered (16 bit), data inside the HTTP GET are encrypted using 128 bit AES encryption (Rijandel), CFB method.<sup>3</sup>
- If a sensor is set as *Unused*, its parameters are not sent in GET.

### GET format

- *Example of get after power on (or restart)*:

```
script.php?mac=0080A397EB95&type= Papago 1TH 2DI 1DO ETH
&description=START&date_time=06/22/2023 13:20:10
```

- *Example of periodic get parameters*: (Pro přehlednost jsou vynechány znaky & mezi atributy)

```
script.php?mac=0080A397EB59 type=Papago 1TH 2DI 1DO ETH
description=LOG log_index=261 date_time=02/12/2016 12:38:40
in1_name=Input 1 in1_state=0 in1_conv=1 in1_units=m3 in1_raw=1
in2_name=Input 2 in2_state=0 in2_conv=2.000 in2_units=kWh
in2_raw=2000 out1_name=Output out1_state=0 T1V1_value=29.0
T1V1_units=°C T1V1_status=2 H1V2_value=43.2 H1V2_units=%
H1V2_status=0 D1V3_value=15.2 D1V3_units=°C D1V3_status=0
```

- *Example of a get after pressing a button in the settings*:

```
script.php?mac=0080A393A273
&type=Papago 1TH 2DI 1DO ETH&description=TEST
```

- *Example of a get encrypted with the string HoryDomkyZahrada: 3*

```
script.php?encrypted_data=B6%70%0D%6A%C4%05%19%18%07%B4%D4%F1%42%A7%7
E%6D%D6%C6%61%29%F6%47%CC%69%4A%58%12%96%0F%E1%D7%67%5C%E2%50%9E%CF%B
B%64%E4%B7%CD%BD%61%A5%E8%14%1D%2B%BA%A1%C3%0E%79%39%E2%A4%BE%0A%5
2%5D%1F%AA%E8%79%C0%B6%CA%A6%1A%5E%D5%B0%B4%D2%0C%71%6B%A7%6D%CC%26%7
D%BD%F5%6E%80%F5%DA%24%AB%44%8D%DA%48%DE%1F%16%89%C3%F4%B4%5E%17%51%7
F%C9%67%B9%BE%27%0C%44%AF%2A%9E%81%F9%7A
```

The above shown encrypted part contains the following data:

```
mac=0080A397EB95&type=Papago 1TH 2DI 1DO ETH
&description=START&date_time=05/29/2024 13:42:50
```

- *Parameters in the get are as follows*:

- **description**: Event type – START, LOG, WATCH, TEST. (START and TEST events do not contain measured values.)
  - ✓ START: Device power on or restart the network interface.
  - ✓ LOG: Periodically sent message.
  - ✓ WATCH\_TEMP: The limits of one of the sensor variables have just been exceeded.

<sup>3</sup> It is 16 bytes of initialization vector followed by encrypted data as specified in the body of standard get. We have examples of get processing for Node.js and PHP in this article on the web (Czech section of website with comments in English): [papouch.com/desifrovani-aes-v-http-getu-z-papaga-p3719/](http://papouch.com/desifrovani-aes-v-http-getu-z-papaga-p3719/)

- ✓ WATCH\_IO: The input state changed now.
- ✓ TEST: Message sent by button in device configuration.
- **mac**: MAC address of the device.
- **type**: Device type name.
- **log\_index**: Periodic message number. This number is useful for detecting whether all messages have been continuously delivered to the server. It is in the range of 0 to 65535.
- **index**: The sequence number of the message with the change information on the input. This number is useful for detecting whether all change information has been continuously delivered to the server. It is in the range of 0 to 255.
- **date\_time**: Timestamp.
- **encrypted\_data**: Parameter contains data from the encrypted GET.<sup>3</sup>

#### Inputs parameters:

- **inX\_name**: User defined input name.
- **inX\_state**: Input state: Input is OFF (0) or ON (1).
- **inX\_conv**: Current counter value converted to real value based on an entered conversion equation.
- **inX\_units**: Unit.
- **inX\_raw**: Counter value as a whole number without conversion.

#### Outputs parameters:

- **outX\_name**: User defined output name.
- **outX\_state**: Output state: Input is OFF (0) or ON (1).

#### Sensor:

The following parameters can be listed more than once if the sensor measures multiple variables. The first character can be **T** (for temperature), **H** (for humidity) or **D** (for dew point).

- **T1V1<sup>4</sup>\_value**: The first temperature as a decimal number.
- **T2V1\_value**: The second temperature as a decimal number.
- **T1V1\_units**: The unit of the first measured temperature.
- **T2V1\_units**: The unit of the second measured temperature.
- **T1V1\_status**: Status of the first value: the value is OK (0), upper limit exceeded (2), lower limit exceeded (3) invalid value (4).
- **T2V1\_status**: Status of the second value: the value is OK (0), upper limit exceeded (2), lower limit exceeded (3) invalid value (4).

---

<sup>4</sup> The number after the **letter T** indicates the serial number of the connector on the device. The number after the **letter V** indicates the serial number of the parameter from the connected sensor.

Response to the HTTP GET

If you want to send a command to change output state within the http GET answer, or subtract a value from the counter state, the server should send the answer in XML format. Answer then should contain attributes *out1* and *cnt1* or *cnt2*, which can set the output and / or subtract a value from the counter state. (XML can only contain some of these attributes.) Values shall be sent in this format:

```
<root>
  <set valid="1" out1="1" cnt1="7" cnt2="5.5" />
</root>
```

If the HTTP GET is encrypted, the answer to it has to be encrypted as well and with a following format – the whole length must be no longer than 250 characters!

```
<root>
  <set encrypted_data=%5D%13F0%9A%2E%C3%D7%1D%F1%4F%A6%1E%AC%00%90%B4%86%AE%9A%
0D%7B%30%E6%D1%14%78%8B%C3%01%24%0C%D5%EC%9E%C7%10%16%48%2E%48%9C%0A%1F%97%31%60%C
7%91%79%F8%D1%D2%A5%62 />
</root>
```

Setting counters and output by an HTTP GET

Using HTTP GET, you can change counters or output states via *set.xml* script. This script accepts non-encrypted messages only. Papago understands these commands:

- **Setting counter to a value**

```
set.xml?type=n&id=3&val=156
```

Parameter *id* is counter number, counted from 1. Parameter *val* is a new counter value. Papago expects whole or decimal number depending on the number of decimals set for this counter.

- **Subtract value from counter state**

```
set.xml?type=m&id=1&val=37.2
```

Parameter *id* is counter number, counted from 1. Parameter *val* is a value you want to subtract from the counter state. Papago expects whole or decimal number depending on the number of decimals set for this counter.

- **Turn output on**

```
set.xml?type=s&id=1
```

- **Turn output off**

```
set.xml?type=r&id=1
```

Answer to the sent GET is XML in this format:

```
<root>
  <result status="1" />
</root>
```

Should the attribute *status* be 0, it means the command was not performed because it contains errors or unexpected values.

## MQTT

Under *Mode* select MQTT. Papago sends its states to the MQTT broker in the publish topic and also monitors the subscribe topic for output control and counter subtraction.

### HTTP / MQTT

Mode	MQTT
After the event	<input checked="" type="checkbox"/>
Periodically	30s
Server address	iot.example.com
Port number	1883
Path / Publish topic	api/environment
Subscribe topic	api/control
QoS	0
User name	userPassword
Password	●●●●●●●●●●●●●●●●
Retype password	●●●●●●●●●●●●●●●●
<input type="button" value="Test message"/>	

fig. 12 - MQTT settings

- *Send periodically*: Message sending period. Time 10 s to 24 h. Enter zero to disable sending. The data can be entered in whole minutes (5m, 20m, etc.), hours (1h, 12h, etc.) or seconds.
- *After the event*: When leaving the limits and when returning to the limits, a message will be sent via the MQTT protocol. It should be taken into account that with rapid changes in the inputs, it is not realistic for the device to send a message for absolutely every change.
- QoS can only select 0.
- *Security*: SSL/TLS security is not supported.
- *Password*: Maximum 15 characters.<sup>2</sup>

### Payload format for publish topic

```
{
  "loc": "NONAME",
  "dev": "Papago 1TH 2DI 1DO ETH",
  "time": "05/22/2024 13:20:53",
  "mac": "001A2B3C4D5E",
  "description": "LOG",
  "log_index": 123,
  "dins": [
    {"id": 1, "name": "Pwr meter", "bin": 0, "val": 11, "unit": "kWh", "raw": 11000},
    {"id": 2, "name": "Sauna", "bin": 1, "val": 1689, "unit": "kWh", "raw": 1689}
  ],
  "douts": [
    {"id": 1, "name": "Relay", "bin": 1}
  ],
  "sensor": [
```

```

    {"t": "temp", "v": 12.5, "u": 0, "io": 1, "e": 0}
  ]
}

```

- **dev:** Device type
- **mac:** MAC address
- **loc:** Location
- **description:** The event type is the same as http get (see page 13).
- **log\_index:** Sequence number of periodically sent message. This way you can check continuity of sent messages.
- **time:** Message sending time according to Papago's internal clock.
- **dins:** Array of objects with the status of individual digital inputs. Each object contains the following items:
  - **id:** Input number (the first one is 1).
  - **name:** User-defined input name.
  - **bin:** Current binary state of the input (0/1).
  - **val:** The current counter value in *unit*.
  - **unit:** User-defined unit of the *val* value.
  - **raw:** Raw value of the counter state on the input without conversion to unit.
- **douts:** Array of objects with the state of each digital output. Each object contains the following items:
  - Properties *id*, *name* and *bin* with similar meaning as for inputs.
- **sensor:** Array with quantities from the connected sensor. Each element of the array contains an object with these values:
  - **t:** quantity type: temp (temperature), hum (humidity), dew (dew point)
  - **v:** quantity value
  - **u:** unit code id – see Tab. 1 on page 25.
  - **io:** sensor id – always 1
  - **e:** status / error code
    - **0:** all right
    - **2:** exceeding the upper limit of the measured range (overflow)
    - **3:** the measured value is less than the lower limit of the range (underflow)
    - **4:** sensor error

#### Payload format for subscribe topic

```

{
  "mac": "001A2B3C4D5E",
  "cnt_sub": [0,0,2.5],
  "outs": "1"
}

```

}

- **mac:** Optional parameter. If specified, Papago will process the message only if the specified MAC address is the same as the Papago MAC address.
- **cnt\_sub:** Array with five values to be subtracted from the counters. Integer or decimal is expected depending on the number of decimal places set for this counter.
- **outs:** A string with one value to set the output to. It can be 0 (turn off), 1 (turn on) or x (leave unchanged).

## Inputs and outputs configuration section

Inputs and outputs operation mode configuration.

### Inputs and output configuration

Input sampling rate

#### Input 1 counter

Input name

Method of operation

After this number of recorded impulses:

...add this value to the counter:

Decimal count

Unit

#### Input 2 counter

Input name

Method of operation

After this number of recorded impulses:

...add this value to the counter:

Decimal count

Unit

#### Output

Output name

Output mode

Relay output default state

Output pulse

Pulse length

Limits setting

fig. 13 – inputs and outputs settings

**Inputs sampling rate** is common for both inputs and represents time of how long a value has to be on the input to be considered a valid one.

**Operation mode** is essentially a way of counting pulses on the input. Counter can be *off* or it can count *leading edges*, *falling edges* or *both edges*. Once the counter is on, conversion settings can be modified to convert the pulses to a real value. (For example, if the connected electricity gauge

has a resolution of 100 pulses per kWh, enter this conversion, write kWh to the *Unit* field and the main page will display the consumption in kWh

You can also set **Operation more** on the **output**. One of these can be applied:

- Manual control
- Pulse control
- Input 1 mirroring
- Input 2 mirroring
- Thermostat for temperature
- Thermostat for humidity
- Thermostat for dew point

Once *Manual* or *Pulse control* is set, default *output* relay contact state can also be set. This state will be on after a power up or a reboot of the Papago unit.

In *Pulse mode* the output *pulse length* is set to define how long the relay should be on. Then the pulse of this length can be sent from the main page or via HTTP GET.

In the *Out of limits watching* mode, the two limit fields are used. Once the watched value goes out of these limits, the *output* relay will be turned on.

## Sensor Section

Sensor and limits configuration.

### Sensor

Connected sensor	Temperature / Humidity (TH3x) <input type="button" value="Autodetect"/>
Temperature range	-40 °F to 257 °F
Temperature unit	Fahrenheit [°F]

### Limit watching

Temperature watch	<input checked="" type="checkbox"/>
Temperature limits	100 <input type="text"/> 250 <input type="text"/>
Hysteresis	0 <input type="text"/>
Humidity watch	<input checked="" type="checkbox"/>
Humidity limits	20 <input type="text"/> 60 <input type="text"/>
Hysteresis	0 <input type="text"/>
Dew point watch	<input type="checkbox"/>
Dew point limits	-40 <input type="text"/> 257 <input type="text"/>
Hysteresis	0 <input type="text"/>

fig. 14 - configuration of one of the sensors

By pressing the *Autodetect* button, all settings for Sensor A and/or B are done automatically according to the currently connected sensor(s), above all the right type of the sensor is entered in the field *Connected sensor*.

## Other Settings

---

Here you can set the user device name, language and time synchronization with the NTP server or time synchronization with the computer.

### Other settings

Name of the device

Language

### Date and time

Synchronize device's time with NTP server

NTP server's IP address

Time zone

Auto daylight saving

DST for Southern Hemisphere

Synchronize device's time with this PC's time

*fig. 15 - other settings*

## CONFIGURATION VIA TELNET PROTOCOL

### Connection

#### IP address is not known

*It is recommended that the IP address should be set using the Ethernet Configurator software (for more information see page 7).*

- 1) Open the window of the cmd command. (In the Windows OS select Start/Run, enter `cmd` in the provided line and click Enter.)
- 2) Make the following entries into the ARP table:
  - a. Type `arp -d` and confirm by Enter. This will delete the current ARP table.
  - b. Use the following command to assign 192.168.1.254 to the module MAC address:  
`arp -s [new_ip_address] [MAC_address_of_device]`  
example: `arp -s 192.168.1.254 00-20-4a-80-65-6e`
- 3) Now open Telnet. (Type in `telnet` and click Enter.<sup>5</sup>)
- 4) Enter `open [new_ip_address] 1` and confirm.
- 5) After a while, the terminal will display an error message saying that connection failed. However, this step is necessary for the module to enter the IP address into its ARP table.
- 6) Connect to the IP address of the module. (Type in `open [IP address in dotted format] 9999` and click Enter.)
- 7) So far you have only entered the configuration mode of the module. The IP address has not yet been set. It must be set in the menu Server Configuration > IP Address. If you close the configuration mode without saving the settings and IP address configuration, the whole procedure must be repeated!
- 8) If the entered IP address is valid, the device displays an introductory text ending with:  
**Press Enter for Setup Mode**  
Press Enter within 3 seconds, otherwise the configuration mode will close.
- 9) The device will display a preview of its settings.
- 10) The preview ends with a paragraph called "Change setup:" which lists the groups of parameters that can be configured. Network parameters can be changed in the "Server" section where you can set a new network address and other parameters.

---

<sup>5</sup> In Windows 10 or higher, Telnet client is not a standard part of system. Install it using following procedure:

- a) Open the "Control Panels/Programs and Features" menu.
- b) On the left, click "Enable or disable features of Windows system" (this option requires the administrator to log in).
- c) The "Features of Windows system" window displays. Here tick the "Telnet service Client" field and click Ok. The client for Telnet will be installed.

**IP address is known**

- 1) In OS Windows choose Start/Run, enter `telnet` in the provided line and press Enter.<sup>5</sup>
- 2) Connect to the IP address of the module. (Type in `open [IP address in dotted format] 9999` and press Enter.)
- 3) If the entered IP address is valid, the device displays an introductory text ending with:  
**Press Enter for Setup Mode**  
 Press Enter within 3 seconds, otherwise the configuration mode will close.
- 4) The device will display a preview of its settings.
- 5) The preview ends with a paragraph called "Change setup:" which lists the groups of parameters that can be configured. Network parameters can be changed in the "Server" section.

**Telnet main menu**

Individual items can be chosen using the numbers written next to them. Choose the required number and press Enter.

The menu structure is as follows:

```

Change Setup:
  0 Server
  ...
  7 Defaults
  8 Exit without save
  9 Save and exit           Your choice ?

```

**Server**

Basic Ethernet settings.

This section contains the following parameters:

```

IP Address : (192) . (168) . (001) . (122)
Set Gateway IP Address (N) ?
Netmask: Number of Bits for Host Part (0=default) (16)
Change telnet config password (N) ?

```

**IP Address**

*(IP address)*

IP address of the module. The digits must be entered one by one and separated by Enter.

Default value: 192.168.1.254

**Set Gateway IP Address**

*(set the IP address of the gateway)*

**Gateway IP addr**

*(IP address of the gateway)*

In "Set Gateway IP Address" enter "Y" to change the IP address. The system then prompts you to change the Gateway IP address. The digits must be entered one by one and separated by Enter.

**Netmask***(network mask)*

Here you specify the number of bits of the IP address that make up the network part.

The Netmask is set as a number of bits determining the range of available IP addresses of the local network. If, for example, value 2 is entered, the structure of the Netmask is 255.255.255.252. The entered value specifies the number of bits from the right. The maximum is 32.

Default value: 8

Example:

The mask 255.255.255.0 (binary form: 11111111 11111111 11111111 00000000) =. number 8.

The mask 255.255.255.252 (binary form: 11111111 11111111 11111111 11111100) = number 2.

**Change telnet config password***(Set the password for Telnet)***Enter new Password***(Enter the password for Telnet)*

This parameter is used to set a new password which is required prior to any configuration via Telnet or via WEB interface (admin password).

For item "Change telnet config password", enter "Y" to change the password. The system then prompts you to change the password.

**Factory Defaults**

By pressing number 7 the device restores the default settings.

The default setting means that all parameters will return to their initial factory settings. The IP address remains unchanged; the web interface port is set to 80.

**Exit without save**

To close the configuration mode without saving the changed parameters.

**Save and exit**

This option saves the changes. If any parameter has been changed, the device is restarted. The restart takes several tens of seconds.

## XML

It is possible to obtain the last measured values, limits (thresholds) and device name from the device in the form of a text file in the XML format. The file is available at [http://\[IP-address\]/fresh.xml](http://[IP-address]/fresh.xml) – i.e. for example at <http://192.168.1.254/fresh.xml> for the default settings.

```
<root>
  <sns id="1" type="1" status="3" unit="0" val="24.50" w-min="63.40" w-max="113.10"
    type2="2" status2="2" unit2="1" val2="48.60" w-min2="-39.10" w-max2="31.60"
    type3="3" status3="0" unit3="1" val3="13.0" w-min3="0.00" w-max3="30.00"/>
  <din id="1" name="Power meter" bin="0" val="23.89 kWh" raw="825025"/>
  <din id="2" name="Sauna" bin="0" val="31.03 Pa" raw="995"/>
  <dout id="1" name="Relay" bin="0" mode="4"/>
  <status location="Parrots IoT" time="06/04/2024 12:32:03" signal="4"/>
</root>
```

fig. 16 – example of XML with actual values

In XML there are tags *sns*, *din*, *dout* and *status*:

### sns

- **id**: Quantity id – first is 1.
- **type**, **type2**, **type3**: There can be a number 1 (temperature parameters), 2 (humidity parameters) or 3 (dew point). Attributes with the same index refer to the same quantity.
- **status**, **status2**, **status3**: Describes the status of the measured value. Attributes with the same index refer to the same quantity. It can take the following values:
  - 0 ..... the value is valid and represents the currently measured value
  - 2 ..... the measured value exceeded the user-set upper limit
  - 3 ..... the measured value has fallen below the user-set lower limit
  - 4 ..... measurement error or sensor error (means damaged sensor or cable)
- **unit**, **unit2**, **unit3**: The number represents the set temperature unit code according to Tab. 1.
- **val**, **val2**, **val3**: The currently measured value as a decimal number with an accuracy of one or two decimal places, depending on the selected range and sensor type. (The validity of the value is described by the *status* attribute.)
- **w-min**, **w-min2**, **w-min3**, **w-max**, **w-max2**, **w-max3**: The lower (*w-min*) and upper (*w-max*) limit of the value set by the user. Values given as decimal numbers to one tenth.

### din

- **id**: Input number. (First number is 1.)
- **name**: User defined input name.
- **bin**: Number 0 or 1 based on the input status - off (0) or on (1).
- **val**: Calculated counter value as an integer or a decimal number including units (when entered).
- **raw**: Current counter status without calculations.

**dout**

- **id**: Output number. (First number is 1.)
- **name**: User defined output name.
- **bin**: Number 0 or 1 based on the output status - off (0) or on (1).

**status**

- **location**: User-defined name of the device.
- **time**: The current system time.
- **signal**: Number 0 to 5, with the current strength of the wifi signal (only for the variant with WiFi).

**Units codes**

In [xml](#), [MQTT](#), [SNMP](#), [Modbus](#) and [Spinel](#) the constants described in the following table are used for the unit codes of the variables. For each variable, the constants are numbered separately from zero.

code	temperature, dew point	humidity
0	°C	%
1	°F	
2	K	

Tab. 1 - Units codes

## SNMP

The SNMP protocol (version 1) contains objects with individual values. For a detailed description of the objects see below. The MIB table you can import into your SNMP manager can be downloaded from [papouch.com](http://papouch.com).



fig. 17 – SNMP objects

**Tip:** If you want to traverse the entire SNMP object tree with the SNMPWALK (Linux) utility, then you need to specify after the IP address from which node to start reading. Example:

```
snmpwalk -v1 -c public 192.168.1.254 1.3.6.1.4.1.18248
```

---

**Objects – variables**

---

**Input – State**

*Name:* inState

*Object ID:* 1.3.6.1.4.1.18248.34.1.2.1.1.1.1

*Description:* Input state as a number 0 or 1.

**Input – Counter value**

*Name:* inCounter

*Object ID:* 1.3.6.1.4.1.18248.34.1.2.1.1.2.1

*Description:* Converted counter value as a whole number. (You can subtract set value from the counter state by writing the value here). Following number of decimals has to be applied to the value to get the converted counter value.

**Input – Number of decimals**

*Name:* inDecNum

*Object ID:* 1.3.6.1.4.1.18248.34.1.2.1.1.3.1

*Description:* Number of decimals as a whole number. This number of decimals has to be applied to the value to get the converted counter value.

**Input – Unit**

*Name:* inUnit

*Object ID:* 1.3.6.1.4.1.18248.34.1.2.1.1.4.1

*Description:* User defined unit that specifies the converted value.

**Output – State**

*Name:* outState

*Object ID:* 1.3.6.1.4.1.18248.34.1.3.1.1.1.1

*Description:* Output state as a number 0 (off) or 1 (on).

**Type**

*Name:* inChType

*Object ID:* 1.3.6.1.4.1.18248.34.1.4.1.1.1 to 3<sup>6</sup>

*Description:* The type of this value. It can have one of the following values:

- 0 → Not used.
- 1 → Temperature.
- 2 → Humidity.
- 3 → Dew point.

**Status**

*Name:* inChStatus

*Object ID:* 1.3.6.1.4.1.18248.34.1.4.1.1.2.1 to 3<sup>6</sup>

*Description:* The status of this value. It describes the current status of the measured value. It can have one of the following values:

- 0 → The value is valid and within the limits.
- 1 → The value has not yet been measured.
- 2 → The value is valid and exceeds the upper limit.
- 3 → The value is valid and exceeds the lower limit.
- 4 → The value is invalid – measurement error.

**Measured value**

*Name:* inChValue

*Object ID:* 1.3.6.1.4.1.18248.34.1.4.1.1.3.1 to 3<sup>6</sup>

*Description:* The measured value as an integer. To obtain the real value, divide by ten.

**Unit**

*Name:* inChUnits

*Object ID:* 1.3.6.1.4.1.18248.34.1.4.1.1.4.1 to 3<sup>6</sup>

*Description:* Unit of the value. May contain one of the following values:

- 0 → degrees Celsius.
- 1 → degrees Fahrenheit.
- 2 → degrees Kelvin.
- 3 → percentage (humidity)

**SNMP objects – general**

---

The following two objects relate to the entire device.

**Device name**

*Name:* deviceName

*Object ID:* 1.3.6.1.4.1.18248.31.1.1.1.0

*Description:* User-defined device name.

**Alarm text**

*Name:* psAlarmString

*Object ID:* 1.3.6.1.4.1.18248.31.1.1.2.0

*Description:* Text of the alarm message sent when a threshold is exceeded.

---

<sup>6</sup> The ID of the objects shows the values from sensors A and B arranged one after another. First A, then B. The values are arranged in the order of temperature, humidity, dew point, i.e. there are 2 or 6 objects.

## Traps

### Trap 1 – Value is outside the limits

The trap contains the measured value and the limit that was exceeded.

The trap is only sent when one of the limits has been exceeded. The trap can only be delivered to a properly configured IP address of a PC with the SNMP manager.

### Trap 2 – Current measured values

The trap contains all current values as well as the name of the device set by the user.

The trap is sent only if a non-zero frequency of sending has been set.

## MODBUS TCP

### Output

#### Output state reading

To access these values, use *Read Coils* function.

Address	Access	Function	Name
0	read	0x01	<b>Output 1 state</b> 0 = output is off 1 = output is on

#### Setting output state

To access these values, use *Write Single Coil* or *Write Multiple Coils* function.

Address	Access	Function	Name
0	write	0x05 0x0F	<b>Output 1 state</b> 0 = output is off 1 = output is on

#### Input state read

To access these values use *Read Discrete Inputs* function.

Address	Access	Function	Name
0 – 1	read	0x02	<b>Inputs 1 and 2 states</b> 0 = Input is off 1 = Input is on

## Counters

### Read counters states

To access these values use *Read Holding Register* function.

Address	Access	Function	Name
<b>Counter 1</b>			
0	read	0x03	<b>Function</b> Operation mode as one of these numbers: 0 = not used (turned off in configuration) 1 = counts leading edges 2 = counts falling edges 3 = counts both edges
1, 2	read	0x03	<b>Date and time</b> Date and time in format according to the NTP.
3, 4	read	0x03	<b>Counter value as a whole number</b> Following register and number of decimals has to be applied to the value to get the converted counter value.
5	read	0x03	<b>Number of decimals</b> Number of decimals as a whole number. This number of decimals has to be applied to the previous value to get the converted counter value.
6, 7	read	0x03	<b>Counter state as a decimal number</b> State of the counter as a decimal number (32 bit float - IEEE 754).
<b>Counter 2</b>			
from 100	Counter 2 values.		

### Counter state setting

To access these values use *Write Multiple Registers* function.

Address	Access	Function	Name
<b>Counter 1</b>			
3, 4	write	0x10	<b>Counter value as a whole number</b> Enter counter value as a whole number. Number of decimals will be taken from the decimal settings from WEB configuration.
6, 7	write	0x10	<b>Counter value as a decimal number</b> Enter value as a decimal number (32 bit float - IEEE 754).
8, 9	write	0x10	<b>Value subtraction – as a whole number</b> Enter counter value as a whole number. This number will be subtracted from the current counter state. <sup>7</sup> Number of decimals will be taken from the decimal settings from WEB configuration.

<sup>7</sup> If such a number is entered that the subtraction would result in negative number, function will be cancelled and exception code 4 will be sent.

Address	Access	Function	Name
10, 11	write	0x10	<b>Value subtraction – as a decimal number</b> Enter value as a decimal number (32 bit float - IEEE 754). This number will be subtracted from the current counter state. <sup>7</sup>
<b>Counter 2</b>			
From 103	Counter 2 values.		

## Sensor

To access these values use *Read Input Register* function.

Address	Access	Function	Name
<b>Sensor 1 – head</b>			
0	read	0x04	<b>Status</b> Contains the status of the sensor. Possible values: 0 = this sensor is not used (set to Not Connected in the configuration) 1 = this sensor is used for measuring
1, 2	read	0x04	<b>Date and time</b> Date and time of the device in the format of NTP.
<b>Sensor 1 – the first parameter (temperature)</b>			
10	read	0x04	<b>Status of the measured values</b> Status of the measured values. Options: 0 = the measured value is within the measuring range 2 = exceeded upper limit of the measuring range (overflow) 3 = exceeded lower limit of the measuring range (underflow) 4 = the measured value is invalid
11	read	0x04	<b>Value in the form of signed integer</b>
12	read	0x04	<b>Value in the float format</b> The upper two bytes.
13	read	0x04	<b>Value in the float format</b> The lower two bytes.
14	read	0x04	<b>Unit</b> The unit in which information is stored in the previous registries. 0 = °C, or % for humidity 1 = °F 2 = K
<b>Sensor 1 – the second parameter (humidity)</b>			
20 to 24			
<b>Sensor 1 – the third parameter (dew point)</b>			
30 to 34			

## SPINEL

The device contains the standard Spinel protocol (format 97) for communication via the TCP data channel. Application development with this protocol is easy due to [Spinel Terminal](#), [Spinel.NET SDK on Github](#) and [Spinel online parser](#).

index	time	data	
0	14:05:59.010	2A 61 00 05 31 02 F3 49 0D	*a..1.óI.
1	14:05:59.018	2A 61 00 25 31 02 00 50 61 70 61 67 6F 20 32 50 54 20 45 54 48 3B 20 76 31 30 31 30 2E 30 31 2E 30 31 3B 20 66 39 37 EB 0D	*a.%1..Papago.2PT.ETH;.v1010.01.01;.f97ë.
2	14:06:07.369	2A 61 00 06 31 02 58 01 E2 0D	*a..1.X.ã.
3	14:06:07.378	2A 61 00 1A 31 02 00 01 01 01 80 00 00 FB 41 C9 7C 81 20 20 20 20 20 32 35 2E 31 1C 0D	*a..1.....ÛAÉ ......25.1..
4	14:06:21.483	2A 61 00 05 31 02 FA 42 0D	*a..1.úB.
5	14:06:21.484	2A 61 00 07 31 02 06 03 F2 3F 0D	*a..1...ò?.
6	14:07:14.566	2A 61 00 57 31 04 0F 58 31 31 2F 32 35 2F 32 30 31 34 20 31 34 3A 30 37 3A 33 32 01 01 01 81 00 20 20 20 20 20 20 20 20 B0 43 00 BD 41 97 79 6B 20 20 20 20 20 20 31 38 2E 39 02 01 01 82 00 20 20 20 20 20 20 20 20 20 B0 43 0C 95 43 A1 0E 49 20 20 20 20 20 33 32 32 2E 31 63 0D	*a.W1..X11/25/2014.14:07:32.....°C.½A.yk.... ..18.9.....°C..Ci. I.....322.1c.
7	14:07:20.156	TCP/IP client socket - disconnecting	
8	14:07:20.166	TCP/IP client socket - disconnect	
9	14:19:35.451	device is gone - serial, parallel - COM8	

fig. 18 - communication with the device using the Spinel Terminal program

Summary of implemented instructions:

### Temperature reading

This instruction reads the current measured values. The values are converted to the currently selected temperature unit. The measured values are returned as a sign integer, as a value in the float format and as an ASCII string.

#### Request:

Instruction code: 58H

Parameters: (sensor)

sensor	Sensor No.	length: 1 byte
The number of the sensor to be read. It is possible to choose 01H (sensor a) or 02H (sensor b).		

#### Response:

Acknowledgement code: ACK 00H

Parameters: {(sensor<sub>1</sub>)(variable<sub>1</sub>)(type<sub>1</sub>)(status<sub>1</sub>)(unit<sub>1</sub>)(unita<sub>1</sub>)(value<sub>1</sub>)} {...}

sensor	Sensor No.	length: 1 byte
This bytes indicates the sensor number and applies to all subsequent bytes until the next <i>chn</i> byte. This means that the following bytes belong to the channel with that number. It is numbered from 01H.		

variable	Variable No.	length: 1 byte
The number of the variable from the given sensor. Numbered from 01H.		

type	Variable type	length: 1 byte
The type of the variable can have one of the following values:		
	00H.....	not defined
	01H.....	temperature
	02H.....	humidity
	03H.....	dew point

status	Status of the measured value	length: 1 byte
The status of the measured value for the channel with the number given in the previous <i>chn</i> .		
bit 0 (LSb)	0 = the <b>lower limit of the monitored range</b> was not exceeded	
	1 = the lower limit of the monitored range was exceeded	
bit 1	0 = the <b>upper limit of the monitored range</b> was not exceeded	
	1 = the upper limit of the monitored range was exceeded	
bit 2	0 = the <b>lower limit of the measuring range</b> was not exceeded	
	1 = the lower limit of the measuring range was exceeded	
bit 3	0 = the <b>upper limit of the measuring range</b> was not exceeded	
	1 = the upper limit of the measuring range was exceeded	
bit 7 (MSb)	0 = the measured value is invalid	
	1 = the measured value is valid	

unit	Unit	length: 1 byte
Unit code: 0 for °C, 1 for °F or 2 Kelvin.		

units	Unit in ASCII string	length: 10 bytes
Unit Code as a right-aligned ASCII string. For example °C, °F, etc.		

value	Measured value	length: 16 bytes
The measured value from the channel with the number given in the <i>chn</i> byte.		
The values are sent simultaneously in three different formats. The first is a 16bit sign value (integer in the form of MSB:LSB), followed by two values converted for the current range based on the current setup: in the 32 bit float format according to IEEE 754 <sup>8</sup> and in the ASCII format. The values are given in the aforementioned order.		
<i>Example:</i>		
The value of 9215.85 is expressed as follows:		
0AH, 58H, 46H, 0FH, FFH, 66H, 20H, 20H, 20H, 39H, 32H, 31H, 35H, 2EH, 38H, 35H		
INT part: 0AH, 58H (2648)		
IEEE 754 part: 46H, 0FH, FFH, 66H		
ASCII part: 20H, 20H, 20H, 39H, 32H, 31H, 35H, 2EH, 38H, 35H ( 9215.85)		

### Examples:

Request – read channel 1:
2AH, 61H, 00H, 06H, 31H, 02H, 58H, 01H, E2H, 0DH
Response:
2AH, 61H, 00H, 1AH, 31H, 02H, 00H, 01H, 01H, 01H, 80H, 00H, 00H, EEH, 41H, BEH, D6H, C3H, 20H, 20H, 20H, 20H, 20H, 20H, 32H, 33H, 2EH, 38H, 93H, 0DH
The value measured on channel 1 was 21,74. Channel number: 01H Variable number: 01H Variable type: 01H Value status: 80H Unit: 00H

<sup>8</sup> The description of the IEEE 754 standard is available here: [http://en.wikipedia.org/wiki/IEEE\\_754](http://en.wikipedia.org/wiki/IEEE_754)

INT part: 00H, EEH (5434)  
 IEEE 754 part: 41H, BEH, D6H, C3H  
 ASCII part: 20H, 20H, 20H, 20H, 20H, 00H, 32H, 33H, 2EH, 38H (21.74)

## Read input states

Reads current inputs states.

### Request:

*Instruction code:* 31H

### Response:

*Acknowledge code:* ACK 00H

*Parameters:* (state)

state	Inbut states in bits	length: 1 byte
Bit oriented byte containing Input states where lowest bit (LSb) represents IN1 state, second lowest bit IN2 state.		

### Examples:

Request:
2AH, 61H, 00H, 05H, FEH, 02H, 31H, 3EH, 0DH
Response:
2AH, 61H, 00H, 06H, 31H, 02H, 00H, 02H, 39H, 0DH

## Read output state

Reads current output states.

### Request:

*Instruction code:* 30H

### Response:

*Acknowledge code:* ACK 00H

*Parameters:* (state)

state	Output state	length: 1 byte
00H = output off 01H = output on		

### Examples:

Request:
2AH, 61H, 00H, 05H, FEH, 02H, 30H, 3FH, 0DH
Response:
2AH, 61H, 00H, 06H, 31H, 02H, 00H, 01H, 3AH, 0DH

## Output settings

---

Sets output relay to the requested state.

### Request:

*Instruction code:* 20H

*Parameters:* (state)

State	Output state	length: 1 byte
01H = off		
81H = on		

### Response:

*Acknowledge code:* ACK 00H

### Examples:

Request:
2AH, 61H, 00H, 06H, FEH, 02H, 20H, 81H, CDH, 0DH
Response:

## Pulse on output

---

It switches the output on or off for a specified time interval. The pulse starts immediately after receiving this command.

It is possible to restart the pulse before the previous one has ended. Repeated pulse triggering can therefore be used, for example, in situations where it is not desirable for the output to remain switched on after a connection failure. If, for example, a pulse is triggered at the output every second for a period of 5 seconds, the relay will switch off no later than 5 seconds after the connection failure.

### Request:

*Instruction code:* 23H

*Parameters:* (time)(state)

Time	Pulse length	length: 2 bytes
Time in tens of milliseconds. A number in the range 1-65535, representing a time of 10 to 655350 ms (i.e. 655.35 s, i.e. about 11 minutes).		

State	Output state	length: 1 byte
01H = off		
81H = on		

### Response:

*Acknowledge code:* ACK 00H

### Examples:

Request:
2AH, 61H, 00H, 08H, FEH, 01H, 23H, 00H, 05H, 81H, C4H, 0DH
Response:
2AH, 61H, 00H, 05H, 31H, 31H, 00H, 0DH, 0DH

## Reading counters

Instruction reads one or more counters.

### Request:

*Instruction code:* 60H

*Parameters:* (counter)

<b>counter</b>	Sensor number	length: 1 byte
Counter number to read. Enter value 00H for all counters or numbers from 01H to 05H for individual counters.		

### Response:

*Acknowledge code:* ACK 00H

*Parameters:* {[channel][value][status][int][float][str][unit][decimals][rawint][rawstr]} {...}

<b>channel</b>		id: 00H
Input number		length: 1 byte
Input number from 1 to 5.		

<b>value</b>		id: 01H
Current input state		length: 1 byte
Current input state as a value 00H (off) or 01H (on).		

<b>status</b>		id: 02H
Counter operation mode		length: 1 byte
Can contain these operation mode codes: 00H ... no actions		

<b>int</b>		id: 03H
Counter value as a whole number		length: 4 bytes
Counter number after conversion as a whole number. (Real value can be obtained by getting the number of decimals. Number of decimals is stored in decimals parameter.)		

<b>float</b>		id: 04H
Counter value as a decimal number		length: 4 bytes
Counter value as a decimal number after conversion (32 bit float - IEEE 754).		

<b>str</b>		id: 05H
Counter value as a string		length: 10 bytes
Counter value as a string. Decimal separator is a dot. String is right-aligned.		

<b>unit</b>		id: 06H
Unit		length: 10 bytes
User defined unit. String is right-aligned		

<b>decimals</b>		id: 07H
-----------------	--	---------

Number of decimals	length: 1 byte
Number of decimals to convert the real value by and view it.	
<b>rawint</b> Raw value as a whole number	id: 08H length: 4 bytes
Counter value <u>before conversion</u> as a whole number. (Real value can be obtained by getting the number of decimals. Number of decimals is stored in decimals parameter.)	
<b>rawstr</b> Raw value as a string	id: 09H length: 10 bytes
Counter value before conversion as a string. As a decimal divider a dot is used. String is aligned to the right.	

**Examples:**

Request – read channel 1:	
2AH, 61H, 00H, 06H, FEH, 01H, 60H, 01H, 0EH, 0DH	
Response:	
2AH, 61H, 00H, 3DH, 31H, 01H, 00H,	- counter number: 0
00H, 01H,	- input state: 0
01H, 00H,	- counter state
02H, 00H,	- counter value as a whole number
03H, 00H, 00H, 00H, D2H,	- counter state as a decimal
04H, 43H, 52H, 00H, 00H,	- counter state as a string
05H, 20H, 20H, 20H, 20H, 20H, 20H, 20H, 32H, 31H, 30H,	- unit as a string
06H, 20H, 20H, 20H, 20H, 20H, 20H, 20H, C2H, B0H, 43H,	- number of decimals
07H, 00H,	- raw value as a whole number
08H, 00H, 00H, 00H, D2H,	- raw value as a string
09H, 20H, 20H, 20H, 20H, 20H, 20H, 20H, 32H, 31H, 30H,	
23H, 0DH	

**Setting or subtracting counter value**

Instruction sets given counter or subtracts a value from its current value.

**Request:**

*Instruction code:* 65H

*Parameters:* {[channel][operation][status][int][float][str][unit][decimals][rawint][rawstr]} {...}

<b>channel</b> Input number	id: 00H length: 1 byte
Input number as 1 to 2.	
<b>operation</b> Operation type	id: 01H length: 1 byte
Operation to perform is subtraction (01H) or setting (02H).	
<b>int</b> Value as a whole number	id: 02H length: 4 bytes

Value to set/subtract as a whole number.

<b>str</b> Value as a string	id: 03H length: 10 bytes
---------------------------------	-----------------------------

Value to set/subtract as a string. Decimal separator is a dot. String is right-aligned.

<b>float</b> Value as a decimal number	id: 04H length: 4 bytes
---	----------------------------

Value as a decimal number (32 bit float - IEEE 754).

**Response:**

*Acknowledge code:* ACK 00H

**Reading of name and version**

Reads the name of the device, software version and the list of possible communication formats. Set by the manufacturer.

**Request:**

*Instruction code:* F3H

**Response:**

*Acknowledgement code:* ACK 00H

*Parameters:* (string)

<b>string</b>	Name and version	length: 1 byte
---------------	------------------	----------------

Papago 1TH 2DI 1DO ETH; v1075.01.03; f97

In addition to the information described above, the string can also contain other information in sections introduced by a semicolon, space and a small letter to determine which information follows.

**Examples:**

**Request:**

2AH, 61H, 00H, 05H, 31H, 02H, F3H, 49H, 0DH

**Response:**

2AH, 61H, 00H, 2DH, 31H, 02H, 00H, 50H, 61H, 70H, 61H, 67H, 6FH, 20H, 31H, 54H, 48H, 20H, 32H, 44H, 49H, 20H, 31H, 44H, 4FH, 20H, 45H, 54H, 48H, 3BH, 20H, 76H, 31H, 30H, 37H, 35H, 2H, 30H, 31H, 2EH, 30H, 33H, 3BH, 20H, 66H, 39H, 37H, 1CH, 0DH

**Reading of manufacturing data**

This instruction reads the manufacturing data of the device.

**Request:**

*Instruction code:* FAH

**Response:**

*Acknowledgement code:* ACK 00H

*Parameters:* (product\_number)(serial\_number)(other)

<b>product_number</b>	length: 2 bytes
-----------------------	-----------------

Product number. For a device number 0227.00.03/0001 this number is 227.

**serial\_number** length: 2 bytes

Serial number. For a device number 0227.00.03/0001 this number is 1.

**other** length: 4 bytes

Other manufacturing information.

### Examples:

Request:

2AH, 61H, 00H, 05H, FEH, 02H, FAH, 75H, 0DH

### Automatic message

This response is generated when the preset limits are exceeded or when the measured value exceeds the physical range of the sensor. The message may contain information about one or more channels.

*Acknowledgement code:* ACK 0FH

*Parameters:* [event][time] {[sensor][variable][type][status][unit][unitA][value]} {...}

**event** length: 1 byte

Number of the event source

This byte specifies the event source. It can be used to distinguish the automatic message sent when the limits or measuring range are exceeded from other automated messages from the device. The value of this byte is 30H.

**time** length: 19 bytes

time of the event

Time of the event as a string in the format *mm/dd/yyyy hh:mm:ss*

**sensor** length: 1 byte

sensor number

The serial number of the sensor the following bytes belong to. Numbering starts from 01H.

**variable** length: 1 byte

variable number

The serial number of a variable from one sensor, used to distinguishing between different variables obtained from one sensor, if the sensor provides more than one. Numbering starts from 01H.

**type** length: 1 byte

variable type

The type of the variable can have one of the following values:

00H ..... not defined  
 01H ..... temperature  
 02H ..... humidity  
 03H ..... dew point

**status** length: 1 byte

Status of the measured value	
bits 0 to 3 (lower nibble)	0000 = the measured value is within the measuring range
	0001 = the lower limit of the monitored range was exceeded
	0010 = the upper limit of the monitored range was exceeded
	0100 = the lower limit of the physical range of the A/D converter was exceeded
	1000 = the upper limit of the physical range of the A/D converter was exceeded
bit 7 (MSb)	0 = the measured value is invalid
	1 = the measured value is valid

unit unit ID	length: 1 byte
The numerical designation of the unit: 00H..... °C 01H..... °F 02H..... K	

unitA unit as a string	length: 10 bytes
A right-aligned string designating the selected unit. For example "°C"	

value measured value	length: 16 bytes
The values are sent simultaneously in three different formats. The first is a 16bit sign value (integer in the form of MSB:LSB), followed by two values converted for the current range based on the current setup: in the 32 bit float format according to IEEE 754 <sup>9</sup> and in the ASCII format. The values are given in the aforementioned order.	
<i>Example:</i> The value of 9215.85 is expressed as follows: 0AH, 58H, 46H, 0FH, FFH, 66H, 20H, 20H, 20H, 39H, 32H, 31H, 35H, 2EH, 38H, 35H INT part: 0AH, 58H (2648) IEEE 754 part: 46H, 0FH, FFH, 66H ASCII part: 20H, 20H, 20H, 39H, 32H, 31H, 35H, 2EH, 38H, 35H ( 9215.85)	

**Example:**

Automatic response:
2AH, 61H, 00H, 57H, 31H, 04H, 0FH, 58H, 31H, 31H, 2FH, 32H, 35H, 2FH, 32H, 30H, 31H, 34H, 20H, 31H, 34H, 3AH, 30H, 37H, 3AH, 33H, 32H, 01H, 01H, 01H, 81H, 00H, 20H, 20H, 20H, 20H, 20H, 20H, 20H, 20H, B0H, 43H, 00H, BDH, 41H, 97H, 79H, 6BH, 20H, 20H, 20H, 20H, 20H, 20H, 31H, 38H, 2EH, 39H, 02H, 01H, 01H, 82H, 00H, 20H, 20H, 20H, 20H, 20H, 20H, 20H, 20H, 20H, B0H, 43H, 0CH, 95H, 43H, A1H, 0EH, 49H, 20H, 20H, 20H, 20H, 20H, 33H, 32H, 32H, 2EH, 31H, 63H, 0DH

<sup>9</sup> The description of the IEEE 754 standard is available here: [http://en.wikipedia.org/wiki/IEEE\\_754](http://en.wikipedia.org/wiki/IEEE_754)

Automatic information about exceeding the lower limit on channel 1 and the upper limit on channel 2. The meaning of the values sent for channel 1:

Instruction No.: 58H

ASCII time: 31H, 31H, 2FH, 32H, 35H, 2FH, 32H, 30H, 31H, 34H, 20H, 31H, 34H, 3AH, 30H, 37H, 3AH, 33H, 32H

Channel No.: 01H

Variable No.: 01H

Variable type: 01H

Value status: 81H

Units numerically: 00H

Units in ASCII: 20H, 20H, 20H, 20H, 20H, 20H, 20H, 20H, B0H, 43H

Current value:

In the form of INT: 00H, BDH

In the form of float: 41H, 97H, 79H, 6BH

In the form of ASCII: 20H, 20H, 20H, 20H, 20H, 20H, 31H, 3BH, 2EH, 39H

## INDICATIONS

### Two LEDs integrated in the Ethernet connector:

Yellow – LINK: is lit when the device is connected by cable to a switch or PC.

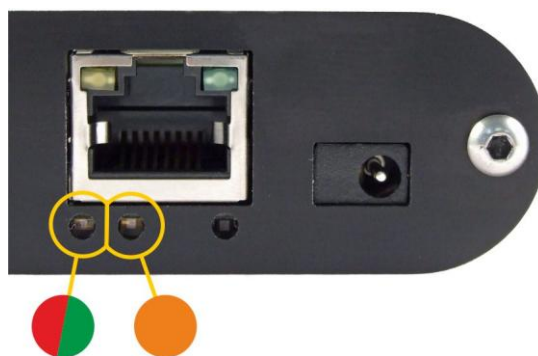
Green – ACT: indicates communication over the Ethernet.

### Two LEDs to the left under the Ethernet connector:

Yellow (right): is lit when the connection is established via Spinel or Modbus.

Red-green (left):

- the green light is lit and the red-light flashes when the device is working properly and is connected to at least one sensor
- the green and red LEDs are lit when the device works, but is not connected to any sensor
- the red LED is lit to indicate an error



### Papago with WiFi connection:

Yellow-blue (right):

- Yellow lights up if Spinel or ModBus connection is established.
- Blue lights up when the Papago is connected to a WiFi network.

Red-Green (left):

- Green lights up and red flashes if the device is OK and at least one sensor is connected.
- Green and Red light up when the device is OK but no sensor is connected.
- Red lights up in case of device fault



### Input and output status indicators:

Above each of the input terminals, and also above the input terminal, there is a red light on the side, which indicates that the contact on the input is switched, or at the output indicates that the relay contact is switched.



fig. 19 - status indicators above the digital inputs and outputs

## RESET

Use the following procedure to reset the device to "factory settings". Unlike the reset that can be done via the web or [Telnet](#), the IP address will be changed to 192.168.1.254 or assigned by the DHCP server.

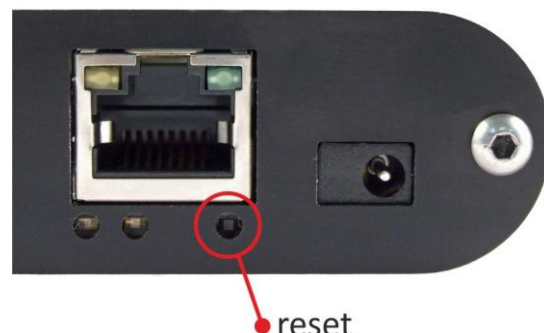
- 1) Disconnect the device from the power supply.
- 2) Press and hold the Reset button.
- 3) Continue according to how you want to assign the IP address:

- a. **IP address 192.168.1.254:**

- i. Turn on the power.
- ii. Wait for a few seconds and release the button between the 5th and 10th sec.
- iii. IP address is set, device is in "factory setting".

- b. **IP address assigned by the DHCP server** (Ethernet version only):

- i. Turn on the power.
- ii. Wait approx. 25 sec and release the button.
- iii. The device is in "factory setting". The IP address assigned by the DHCP server can be found in your DHCP server (typically in your router). The section with such assigned addresses has different names - for example, *DHCP Client List*, *DHCP Clients*, etc.



**TECHNICAL PARAMETERS**

**Integrated temperature and humidity sensor<sup>10</sup>**

Important Notice: Polymer sensor is a highly sensitive element that reacts with chemicals. Do not expose even the outer shell of the sensor to chemicals or their vapors (cleaning with alcohol, petrol etc.). Especially organic solvents and compounds can negatively affect the sensor accuracy by as tens of percent RH.

- Coverage ..... IP 54
- Dimensions ..... 40 × 16 × 10 mm
- Material ..... hardened aluminium

**Humidity sensor**

- Humidity range ..... 0 % to 100 % RH
- Recommended measurement range ..... 20 – 80 %
- Resolution ..... 1% RH
- Humidity measurement accuracy ..... see Fig. 20
- Sensor element ..... polymer sensor
- Sensor mechanical finish ..... inside hardened aluminium block

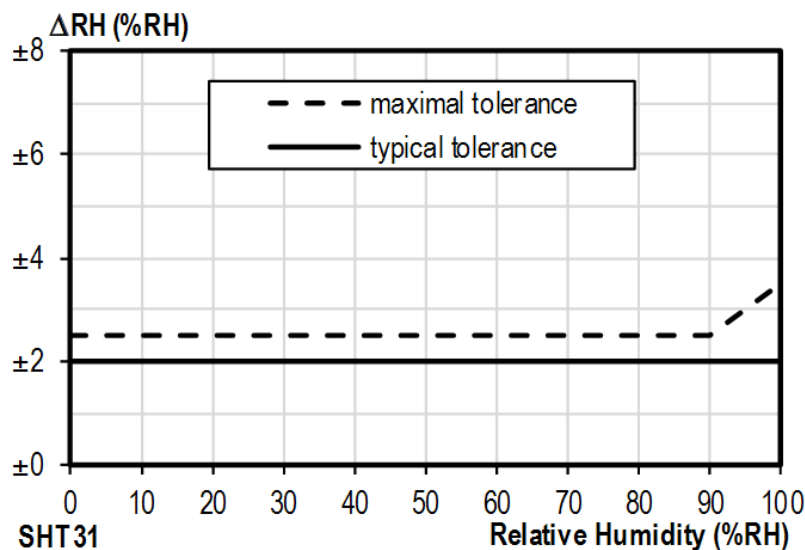


Fig. 20 – Accuracy of humidity measuring

<sup>10</sup> Sensor marked TH3 is supported in firmware including and above version 1.05. If you have an older firmware, you will have to flash the firmware to be able to read from TH3 sensor. Here are the key differences between the old version (Marked as TH2E) and TH3 version:

	TH3 (new sensor)	TH2E (old sensor)
Measurement accuracy within 0 – 10 %	±2 %	±2 to ±4 %
Measurement accuracy within 90 – 100 %	±2 %	±2 to ±4 %
Recommended measurement range	20 – 80 % RH	
Temperature measurement range	-40.0 °C to +125.0 °C	-40.0 °C to +123.8 °C
Temperature measurement accuracy	±0.3 to ±0.5 °C	±0.4 to ±2.0 °C

Operating and Maximum Range of Values

- Sensor is stable in standard range of humidity values. Long-term exposure to conditions outside these values (humidity above 80% in particular) can temporarily shift the measured-out values (by +3% for 60 hours). When the sensor is back to standard ranges, it returns to its pre-calibrated state slowly.<sup>11</sup>
- Long-term exposure to extreme conditions or to chemically aggressive vapor can speed up the aging process of the sensor significantly. It can also shift the measurements.

**Temperature sensor**

Range.....-40.0 °C to +125 °C  
 Resolution.....0.1 °C  
 Sensor element .....semiconductor  
 Sensor mechanical finish.....inside hardened aluminum block

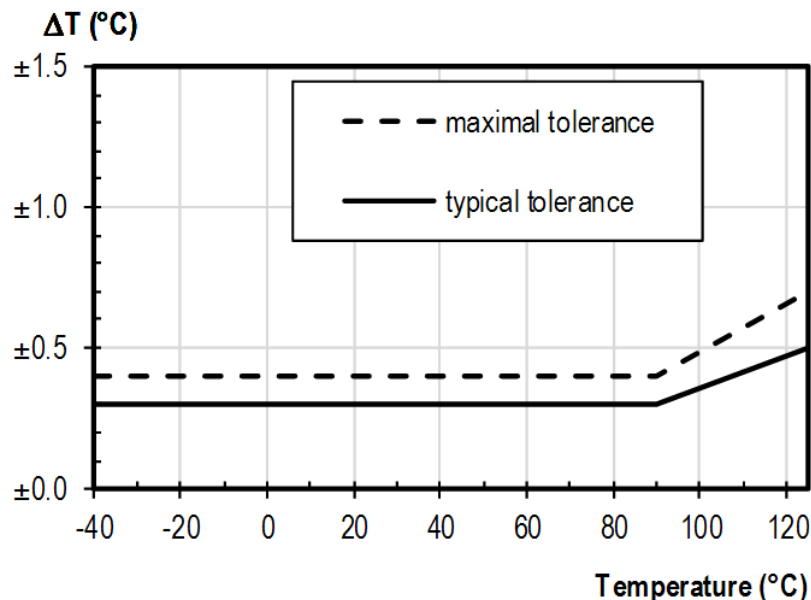


Fig. 21 – Accuracy of temperature measurement

**Standalone temperature sensor**

Sensor type .....semiconductor  
 Measuring temperature range .....-55 °C to +125 °C  
 Accuracy.....±0.5 °C in the range of -10 °C to +85 °C  
 Temperature drift .....±0.2 °C per 1000 hours at 125 °C  
 Dimensions.....normalized diameter 6 mm, length 60 mm  
 Housing material.....hardened alloy  
 Degree of protection .....IP68 (permanent immersion into 1m max.)

<sup>11</sup> You can speed up this process by doing following:

- 1) Leave the sensor in environment above 100 to 105 °C and humidity below 5 % for at least 10 hours.
- 2) Leave the sensor in environment above 20 to 30 °C and humidity approximately 75 % for around 12 hours. (Humidity 75% can be achieved with saturated solution of NaCl.)

## Sensor cable

---

Cable jacket .....	silicone rubber, blue
Wire insulation .....	FEP polymer
Length .....	standard 3 m (optionally up to 20 meters)
Measuring temperature range .....	-60 °C to +200 °C
Maximum allowable temperature .....	+220 °C
Cable diameter.....	4.3 mm (±0.1 mm)

The cable shows excellent resistance to moisture, chemicals and carbohydrates.

## Other parameters

---

### Inputs

Type .....	for contact or TTL levels
Number of inputs.....	2
Input contact on current .....	2 mA
Operating voltage.....	5 V
Maximum sampling frequency .....	1 kHz
Connector .....	slip-on terminal

### Output

Type .....	Relay contact, single pole, double throw (SPDT)
Maximum AC voltage .....	50 V
Maximum DC voltage.....	85 V
Maximum current .....	2 A
Maximum power for resistive load.....	62.5 VA / 60 W
Varistor protection .....	$U_{AC} = 60 \text{ V}$ ; $E_{MAX} = 5 \text{ J}$ ; $C = 0.64 \text{ nF}$
Connector .....	slip-on terminal

### Ethernet interface

Ethernet standard .....	TBase 10/100 Ethernet
Connector .....	RJ45
HTTP GET encryption.....	128 bit AES; Rijndael; CFB method
SNMP protocol.....	v. 1
MQTT protocol .....	v. 3.1.1 (QoS 0)

### WiFi interface

Type .....	IEEE 802.11 b/g and IEEE 802.11n (single stream), IEEE 802.11 d/h/i/j/k/w/r
Operating frequency.....	2.4 GHz
Antenna connector.....	SMA RP

**Clock circuit and internal memory**

Clock backup method (RTC) .....	capacitor (not replaceable by the user)
RTC backup time after power outage .....	5 days (if the device was previously connected to a power source for at least three hours without interruption)

**Device electronics**

PoE power supply.....	according to IEEE 802.3af
Power supply from an external source .....	11 to 58 V DC (with reverse polarity protection)
Current consumption from ext. source at 15 V ...	typically 120 mA; <i>Wi-Fi version: 31mA</i>
Current consumption from ext. source at 24 V ...	typically 72 mA; <i>Wi-Fi version: 31mA</i>
Current consumption from PoE .....	typically 32 mA
Consumption .....	typically 1.8 W
Power supply connector .....	coaxial 3.8 × 1.3 mm; + inside
Operating temperature range .....	-20 to +70 °C
Operating humidity range .....	max. 90 %, non-condensing
Dimensions (without connectors).....	88 × 70 × 25 mm
Housing material.....	anodized aluminum
Degree of protection .....	IP 30
Weight .....	typically 130 g

**Default settings of the Ethernet**

IP address .....	192.168.1.254
Netmask .....	255.255.255.0 (8 bits; mask C)
IP address of the gateway .....	0.0.0.0

**Available designs**

Mountable on 35 mm DIN rail .....	optional accessory
-----------------------------------	--------------------



fig. 22 – Papago 2TH ETH with DIN rail holder

*Do not hesitate to contact us if you have any other requirements concerning the design and functions of PAPAGO TH 2DI DO.*

# Papouch s.r.o.

Industrial data transmission, line and protocol converters, RS232, RS485, RS422, USB, Bluetooth, Ethernet, LTE, WiFi, measurement modules, smart temperature sensors, I/O modules, custom development and manufacturing.

Address:

**Strasnicka 3164  
102 00 Prague 10  
Czech Republic**

Phone:

**+420 267 314 267**

Web:

**en.papouch.com**

Mail:

**info@papouch.com**

