GROLAB CLOUD UPLOAD GUIDE

SPP_GL_003

Open Grow

Preface

Thank you for purchasing GroLab system.

This manual describes how to configure and use the GroLab Cloud Upload functionalities, as well as how to troubleshoot potential problems.

If your issue isn't listed on this guide please contact our support center at support@opengrow.pt.

How to Read this Manual

The present manual is divided into three main sections:

- Cloud Upload Configuration;
- JSON Data Format;
- Troubleshoot.

Symbols in this Manual

In this manual, some important items are described with the symbols shown below. Be sure to read these items before using this equipment.

Note	Indicates information to which you should pay attention when operating the equipment.
Tip	Describes handy information that is useful to know when operating the equipment.
📖	Pages describing items related to what you are currently doing. See these pages as required.

Target Audience for this Manual

This is a manual that is aimed at general users and administrators of GroLab systems.

Screens in this Manual

The details on the screens may differ depending on the equipment version and configurations, such as the GroLab Software settings and the OS version.

Legal Notice

Documentation

Make sure that you have the current version of the documentation. Each document displays the release date and respective version on the Header of each page.

The documentation latest version is available at the Open Grow website: opengrow.pt/.

Documentation Feedback

The feedback given by our customers is always important and help us to constantly grow. In this way, please send any suggestions, error reports or omissions in this document to info@opengrow.pt. Please, include the document title, version, chapter and section titles of the text on which you are reporting.

Index

1. Cloud Upload Configuration

1.1 Introduction

This main section refers to the configuration of the GroLab Cloud Upload functionalities. These functionalities allow a periodical upload of the data logs generated by GroNode to a server. The data logs contain information about all the sensors and devices connected to the GroLab modules, as well as the state of alarms and schedules.

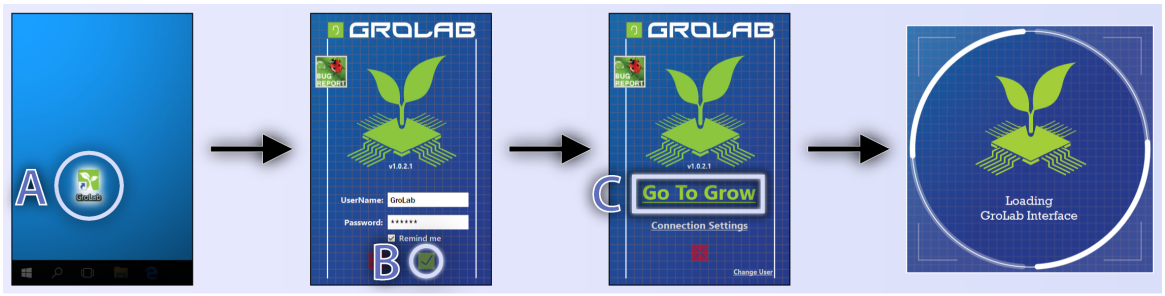
1.2 Pre-steps

Before starting the configuration some pre-steps should be done:

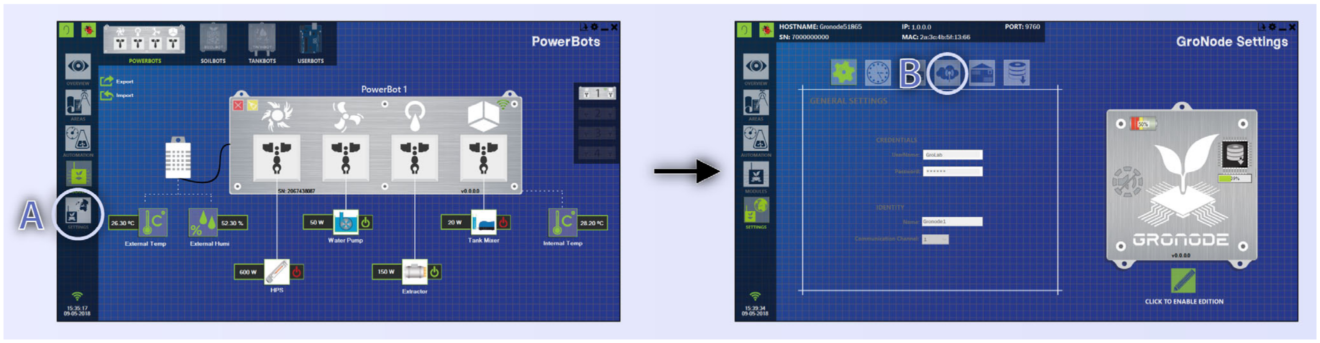
- It is **required to have access to a server** that can **handle TCP connections and to receive PUT requests** with JSON data. Make sure you know the server's IP and port.
- After the **server receives** the **JSON data**, it **should** then **be parsed and processed** based on the user requirements, for that it may be required to contract a third-party service or to develop your own software (the JSON Data Format is explained in detail in the main section 2).
- It is required that **GroNode is correctly installed and configured** (ensure you followed the official GroNode manual).
- It is required that **GroNode has an Internet connection**.
- It is required to have a **PC with the GroLab Software** installed.
- It is required to **have at least one GroLab module besides GroNode**, e.g. a PowerBot.

1.3 GroLab Software Configuration

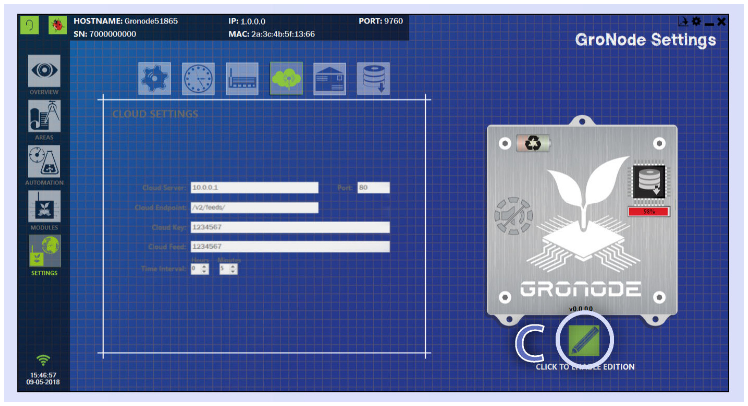
After ensuring the pre-steps are all achieved, it's time to start configuring the Cloud Upload functionalities. First, **open the GroLab Software (A)** and **connect to the GroNode (B and C)**, if you need assistance with this step please refer to the official GroNode manual or GroLab tutorials. Anyway, the basic process is the following:



Now with the GroLab Software main menu open, **navigate to the GroNode Settings section (A)** and **then to the Cloud Settings tab (B)**, as shown in the following picture.

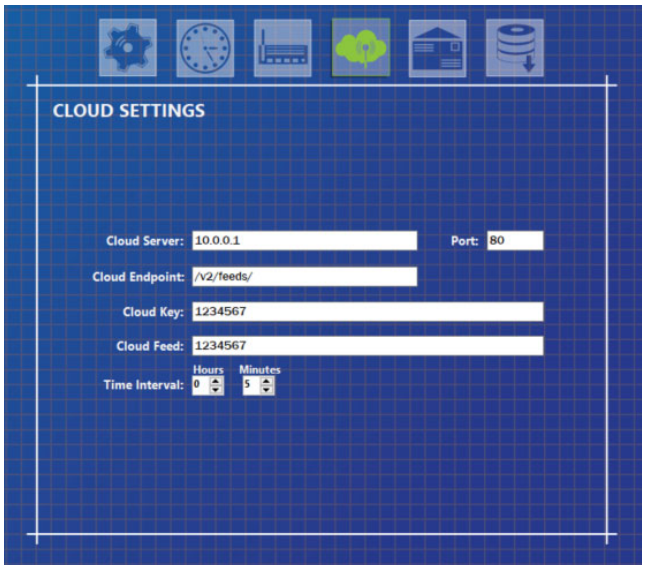


After navigating to the Cloud Settings tab, you should now **click to enable editing (C)**, as shown in the below picture.



1.3.1 Cloud Settings

Now it’s time to look at each field of Cloud Settings:



CLOUD SETTINGS

Cloud Server:	10.0.0.1	Port:	80
Cloud Endpoint:	/v2/feeds/		
Cloud Key:	1234567		
Cloud Feed:	1234567		
Time Interval:	Hours 0	Minutes 5	

- **Cloud Server:** the server’s IP;
- **Port:** the server’s port;
- **Cloud Endpoint:** the path identifier for the data (feeds path) in the remote server*****;
- **Cloud Key:** the API key/password*****;
- **Cloud Feed:** the identifier of the data (*feed id*)*****;
- **Time Interval:** the time interval between each upload.

***variable based on server functionalities.**

Since these **fields should be configured in accordance with the server** that will be used to receive the data, we can't provide the exact configuration for each scenario. However, for the majority of the cases just filling the Cloud Server and the Port fields with the remote server's IP and port, is enough to start receiving the data on the server.

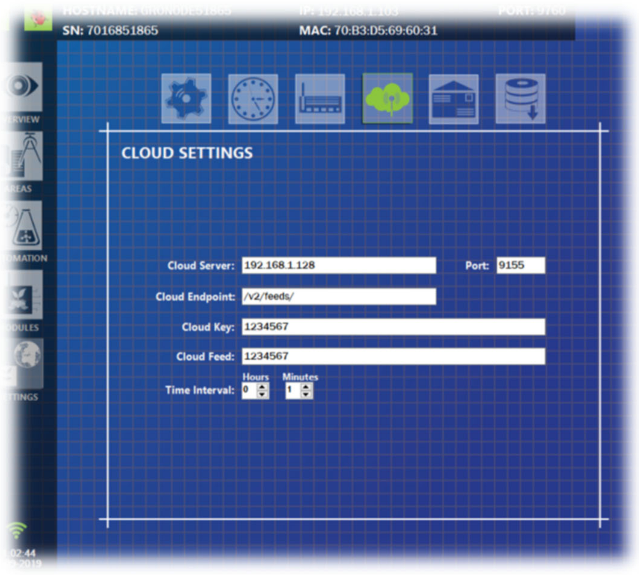
Note: Port forwarding may be required in both GroNode and server networks (normally done through the router and/or physical/digital firewalls). If you need assistance with port forwarding, please refer to your network equipment (as well as firewall, if it's present) official manuals.

2. JSON Data Format

2.1 Introduction

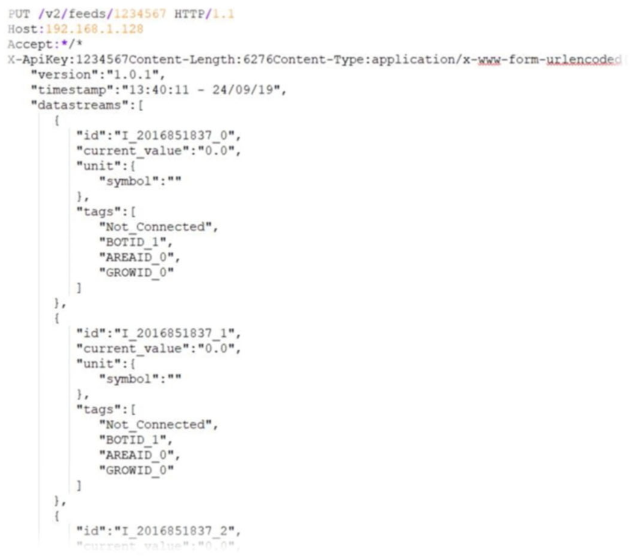
This main section describes the JSON data format used in the GroLab Cloud Upload functionalities. The information in this main section is intended to be used to create software to parse the JSON data. If you are not familiar with JSON, we recommend you to refer to the following W3Schools page: w3schools.com/whatis/whatis_json.asp.

Before explaining the JSON data format and to have a general overview, let's start by looking at an **example configuration** and an **excerpt of the PUT request** sent to the server with the JSON data.


SN: 7016851865 MAC: 70:B3:D5:69:60:31

CLOUD SETTINGS

Cloud Server: 192.168.1.128 Port: 9155
Cloud Endpoint: /v2/feeds/
Cloud Key: 1234567
Cloud Feed: 1234567
Time Interval: Hours 0 Minutes 1


```
PUT /v2/feeds/1234567 HTTP/1.1
Host:192.168.1.128
Accept:*/*
X-ApiKey:1234567Content-Length:6276Content-Type:application/x-www-form-urlencoded
   "version":"1.0.1",
   "timestamp":"13:40:11 - 24/09/19",
   "datastreams":[
      {
         "id":"I_2016851837_0",
         "current_value":"0.0",
         "unit":{
            "symbol":""
         },
         "tags":[
            "Not_Connected",
            "BOTID_1",
            "AREAID_0",
            "GROWID_0"
         ]
      },
      {
         "id":"I_2016851837_1",
         "current_value":"0.0",
         "unit":{
            "symbol":""
         },
         "tags":[
            "Not_Connected",
            "BOTID_1",
            "AREAID_0",
            "GROWID_0"
         ]
      },
      {
         "id":"I_2016851837_2",
         "current_value":"0.0",
```

2.2 JSON Header

The JSON header contains the following keys:

- **"version":** represents the current JSON data format version, the **version** taking into consideration **for this main section is the 1.0.1**;
- **"timestamp":** represents the time and date when the data is generated. It is present in the following format: **"hh:mm:ss - DD/MM/YY"**.

In the below picture there is an example extracted from a JSON data:

```
"version":"1.0.1",
"timestamp":"13:40:11 - 24/09/19",
```

2.3 JSON DataStreams

After the header follows the content of the data logs, it can be found inside the **"datastreams" key**. As explained before on this document, the data logs contain information about all the sensors and devices connected to the GroLab modules, as well as the state of alarms and schedules.

The information is sorted as:

1. **Modules;**
    a. First module's inputs;
    b. Then the module's outputs;
    c. Repeat for the next module.
2. **Schedules;**
3. **Alarms.**

To make it simple **from now one each block of data** (input, output, schedule, and alarm) will be **called piece**.

All the pieces have the following keys in common:

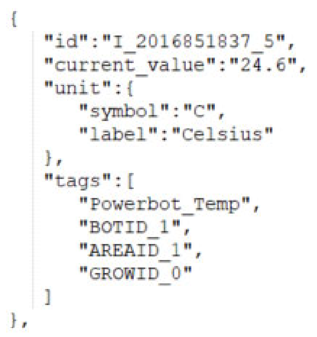
- **"id":** it's the identification of the piece, it has the following formats:
    - **"L_NNNNNNNNNN_N"** when referring to an input or output, in which the 'L' represents a letter and the 'N' a number. The letter can be I or O, meaning Input or Output, the first sequence of numbers refers to the module's serial number*, then the last number refers to the input/output *id* (from 0 to 9) ;
    - **"L_N⁺"** when referring to a schedule or alarm, in which the 'L' represents a letter and the 'N⁺' one or more numbers. The letter can be S or A, meaning Schedule or Alarm, then the number refers to the schedule/alarm *id* (from 1 to 200).

*If the serial number starts by 2 means it's a PowerBot; if it starts by 3 it's a TankBot; if it starts by 4 it's a SoilBot; if it starts by 5 it's a UserBot.

- **"current_value":** for the case, it's an input/output piece refers to the value of it, for the case it's an alarm/schedule piece refers if inactive/idle (0) or active/running (255);
- **"tags":** includes extra info, like the name, associated area and grow *ids*, we will explore this in detail further in this main section.

2.3.1 Inputs

The first pieces of the data logs you should find are relative to the inputs of the first module (unless the module doesn't have inputs). Those pieces should look similar to those shown in the below picture.



```
{
   "id":"I_2016851837_5",
   "current_value":"24.6",
   "unit":{
      "symbol":"C",
      "label":"Celsius"
   },
   "tags":[
      "Powerbot_Temp",
      "BOTID_1",
      "AREAID_1",
      "GROWID_0"
   ]
},
```

As explained before, the **key "id"** contains the **identification of the piece**, in this case, it's an **input (letter "I")**, that belongs to a PowerBot with the **serial number 2016851837** and this **input has the *id* 5**.

The next **key "current_value"** it's the **current value of the input**, in this case, it's **"24.6"**, at this point we still don't know the measurement unit so let's look for the next key "unit".

The **"unit" key** can have **one or two keys inside** of it, the **"symbol" (always has it)** and the **"label" (the ON/OFF input types doesn't have this key):**

- **"symbol":** the letter(s)/symbol that represents the value, in this example it's the 'C' from ºC;
- **"label":** the description of the **"symbol"**, in this case, Celsius.

The last **key** it's the **"tags"** and it **has several values** in it:

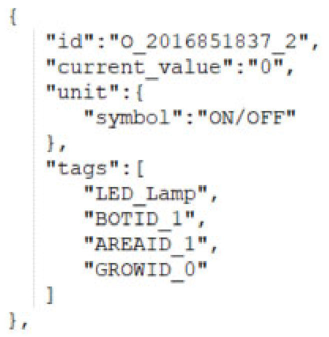
- **"Powerbot_Temp":** the first value refers to the input name;
- **"BOTID_1":** this value refers to the module *id* to which the input belongs to, in this case, the input belongs to the module with *id* 1;
- **"AREAID_1":** this value refers to the area *id* to which the input is associated, in this case, the input is associated with the area 1;
- **"GROWID_0":** this value refers to the grow *id* to which the input is associated, in this case, the input is not associated with any grow (grow *id* 0 does not exist).

  Note: If the input is shared between the two grows, the *id* that represents that scenario is 224 (GROWID_224).

In summary, this is example piece represents an input with the *id* 5, which is a temperature sensor that belongs to the PowerBot with *id* 1 and serial number 2016851837. The sensor is currently measuring 24.6ºC (Celsius) and belongs to the area with *id* 1.

2.3.2 Outputs

After the inputs, we have the outputs. Below you can find an example of an output piece.



```
{
    "id":"O_2016851837_2",
    "current_value":"0",
    "unit":{
       "symbol":"ON/OFF"
    },
    "tags":[
       "LED_Lamp",
       "BOTID_1",
       "AREAID_1",
       "GROWID_0"
    ]
},
```

As explained before, the **key "id"** contains the **identification of the piece**, in this case, it's an **output (letter "O")**, that belongs to a PowerBot with the **serial number 2016851837** and this **input has the *id* 2**.

The next **key "current_value"** it's the current value of the input, in this case, it's **zero**, which **means the output is OFF**, for the case it's **ON** the value **would be 255**.

The **"unit" key** has one key inside of it:

- **"symbol":** the letter(s)/symbol that represents the value, in this example it's the 'ON/OFF'.

The last **key** it's the **"tags"** and it **has several values** in it:

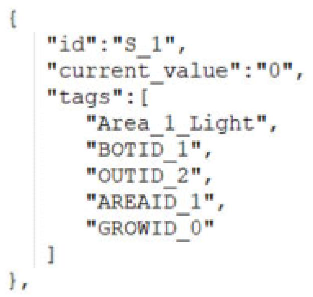
- **"LED_Lamp":** the first value refers to the output name;
- **"BOTID_1":** this value refers to the module *id* to which the output belongs to, in this case, the output belongs to the module with *id* 1;
- **"AREAID_1":** this value refers to the area *id* to which the output is associated, in this case, the output is associated with the area 1;
- **"GROWID_0":** this value refers to the grow *id* to which the output is associated, in this case, the output is not associated with any grow (grow *id* 0 does not exist).

  Note: If the output is shared between the two grows, the *id* that represents that scenario is 224 (GROWID_224).

In summary, this is example piece represents an output with the *id* 2, which is an LED lamp that belongs to the PowerBot with *id* 1 and serial number 2016851837. The lamp is currently OFF and belongs to the area with *id* 1.

2.3.3 Schedules

After all the inputs and outputs of each module, the next pieces that follow are the schedules. Below you can find an example of a schedule piece.



```
{
   "id":"S_1",
   "current_value":"0",
   "tags":[
      "Area_1_Light",
      "BOTID_1",
      "OUTID_2",
      "AREAID_1",
      "GROWID_0"
   ]
},
```

As explained before, the **key "id"** contains the **identification of the piece**, in this case, it's a **schedule (letter "S")** and **has the *id* 1**.

The next **key "current_value"** it's the **current state of the schedule**, in this case, it's **zero**, which **means the schedule is not running**, case it was **running** the value **would be 255**.

The last **key** it's the **"tags"** and it **has several values** in it:

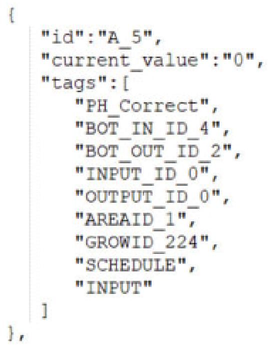
- **"Area_1_Light":** the first value refers to the schedule name;
- **"BOTID_1":** this means the schedule's output belongs to the module with *id* 1;
- **"OUTID_2":** this means the schedule's output has the *id* 2;
- **"AREAID_1":** this means the output belongs to the area with *id* 1;
- **"GROWID_0":** this means the output does not belongs to any grow (grow *id* 0 does not exist).

  Note: If the output is shared between the two grows, the *id* that represents that scenario is 224 (GROWID_224).

In summary, this is example piece represents a schedule with the *id* 1, the schedule controls the output with *id* 2 from the module with *id* 1. The schedule is currently not running and to the area with *id* 1.

2.3.4 Alarms

The last pieces are the alarms. Below you can find an example of an alarm piece.



```
{
   "id":"A_5",
   "current_value":"0",
   "tags":[
      "PH_Correct",
      "BOT_IN_ID_4",
      "BOT_OUT_ID_2",
      "INPUT_ID_0",
      "OUTPUT_ID_0",
      "AREAID_1",
      "GROWID_224",
      "SCHEDULE",
      "INPUT"
   ]
},
```

As explained before, the **key "id"** contains the **identification of the piece**, in this case, it's an **alarm (letter "A")** and **has the *id* 5**.

The next **key "current_value"** it's the **current state of the alarm**, in this case, it is **zero**, which **means the alarm is not active**, for the case it's **active** the value **would be 255**.

The last **key** it's the **"tags"** and it **has several values** in it:

- **"PH_Correct":** the first value refers to the alarm name;
- **"BOT_IN_ID_4":** this means the alarm's input condition belongs to the module with *id* 4;
- **"BOT_OUT_ID_2":** this means the alarm's output condition belongs to the module with *id* 2;
- **"INPUT_ID_0":** this means the alarm's input condition has the *id* 0;
- **"OUTPUT_ID_0":** this means the alarm's output has the *id* 0;
- **"AREAID_1":** this means the output belongs to the area with *id* 1;
- **"GROWID_224":** this means the output is shared between the two grows of the area;
- **"SCHEDULE":** this value refers to the type of alarm action, in this case, it's a timed action. This value could also be "TURN_ON" which means a simple turn on action, or "TURN OFF" which means a simple turn off action, or "MAIL" for the cases the only action the alarm do is to send an email;
- **"INPUT":** this means the alarm's input conditions is referring to an input, if it was referring an output this value would be "OUTPUT".

In summary, this is example piece represents an alarm with the *id* 5, the alarm's input condition is based on an input with *id* 0 that belong to the module with *id* 4. The alarm has an output condition composed of an output with *id* 0 that belongs to the module with *id* 2. The alarm is currently not running and it is from the timed action type.

3. Troubleshoot

3.1 Introduction

This main section will cover the most common issues regarding the Cloud Upload functionalities.

If your problem or issue is not present on the following topics please contact our support center through the email support@opengrow.pt.

3.2 Data Transmission

3.2.1 Server does not receive the data

If your server doesn't appear to be receiving any data from the GroNode, please refer to this section.

Related Problems

- The server is unreachable;
- The GroNode is not being able to establish the connection;

Problem Description

The user configured the Cloud Upload settings based on the server info (IP, port…) but the server appears to not be receiving any data from the GroNode.

Possible Problem/Corrective Action

- **GroNode is not in the same network as the server (case using an internal server).**

  If using an internal server, you should ensure the GroNode and the server is in the same network. Also, ensure that you used the server IP in the network and not the external network IP.

- **GroNode doesn't have access to the Internet (case using a remote server).**

  Ensure GroNode has access to the Internet, otherwise, it will be impossible to establish a connection with the remote server.

- **GroNode is not able to establish a connection because it's being blocked by a Firewall.**

  Ensure GroNode is not being blocked by any physical or digital firewall. The most common scenario is that the user is using the home router, in this case, ensure the router is not blocking the connection (please refer to the router's official instructions manual). Port forwarding may be also necessary.

- **GroNode is not being able to establish or send any data, I've ensured it has an Internet connection and it's not being blocked by anything.**

  In this case, try to reset the GroNode. If this solution doesn't work, please contact our support center.

- **Server is not being able to receive the data.**

  If you are using a remote server, ensure you are using the correct IP and port, it's a common mistake to use the local IP instead of the external IP, so please ensure you used the correct IP. Also, make sure the server is accessible, if you have a way to ensure the IP and port are accessible, please try it. In any case, you can test the IP/port at the following page https://www.yougetsignal.com/tools/open-ports/ (external connection only).

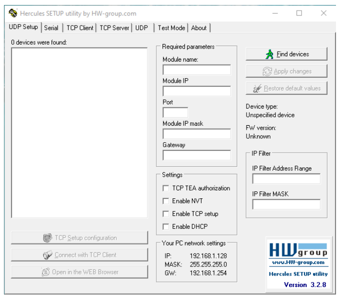
3.2.2 Simple way to try the Cloud Upload

If you are having issues to set up or making the Cloud Upload work as intended. We recommend you to start by trying a simple method we present in this section.

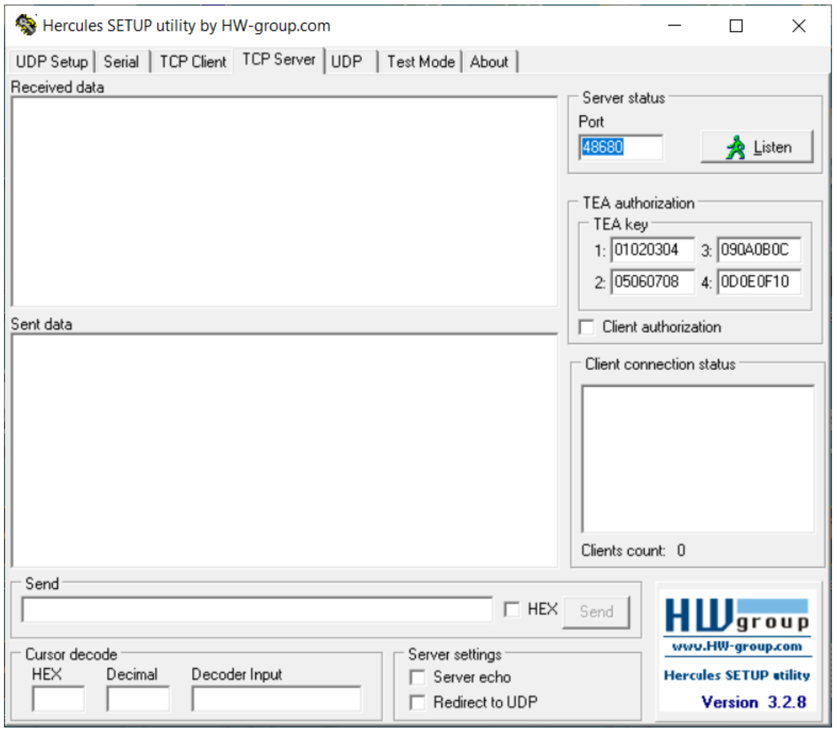
Before start explaining the steps, we will start by explaining a bit about what we will do. Basically, we will use a local PC (same network as GroNode) to emulate a TCP server able to receive the data from GroNode.

In this way, a PC is required as well as the download and the installation of a software called "Hercules SETUP utility" by HW group s.r.o., this software can be download at hw-group.com/software/hercules-setup-utility.

After downloading/installing the software, please execute it. It should display the window shown in the following picture.



The next step is to click the tab named “TCP Server” and then set the desired port (you should ensure the port is not being used by other processes) as shown in the below picture.

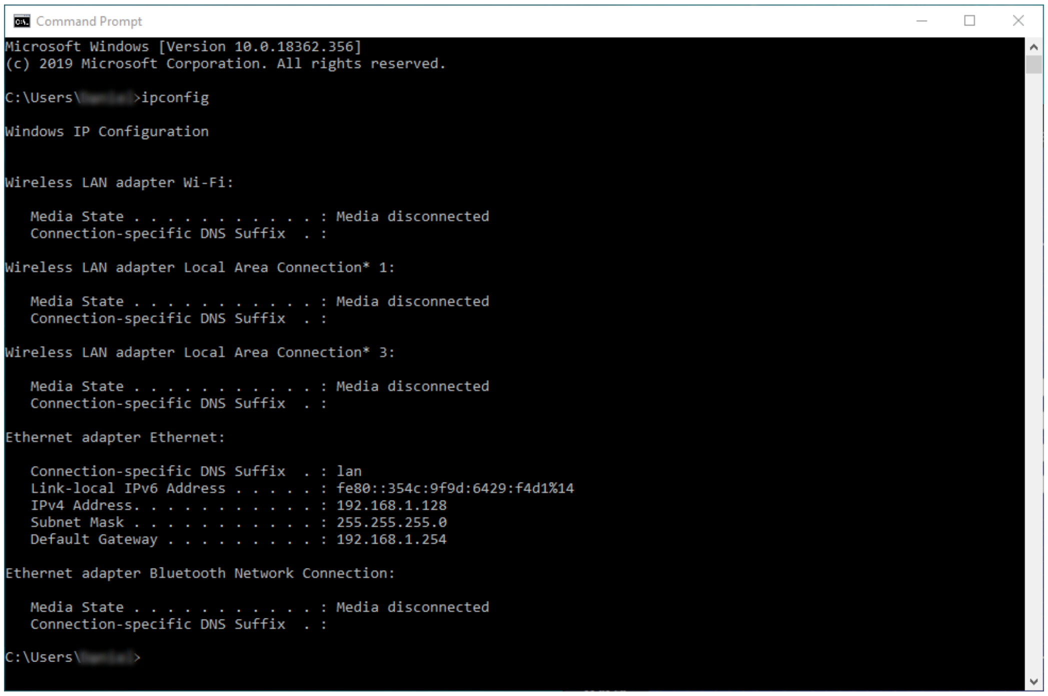


Port: 48680

TEA key
1: 01020304 3: 090A0B0C
2: 05060708 4: 0D0E0F10

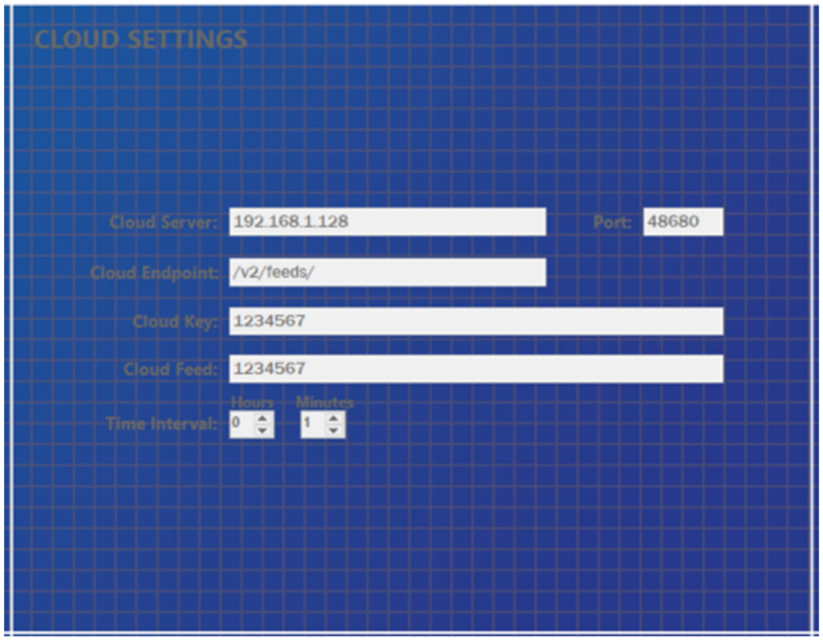
Clients count: 0

Now it’s time to click the “Listen” button and you have a TCP server running in your PC. The next step is to find the local IP of your PC. For that, open the Windows Command Prompt and run the “ipconfig” command as shown in the following picture.



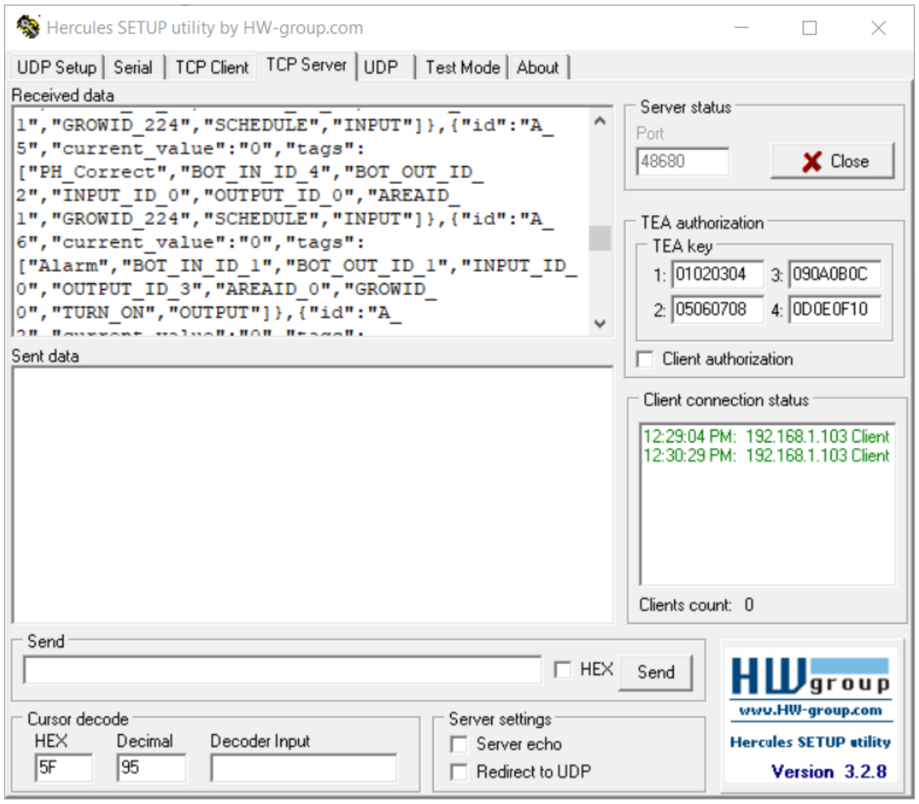
```
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>ipconfig

Windows IP Configuration


Wireless LAN adapter Wi-Fi:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : lan
   Link-local IPv6 Address . . . . . : fe80::354c:9f9d:6429:f4d1%14
   IPv4 Address. . . . . . . . . . . : 192.168.1.128
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.254

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\[redacted]>
```

Now you should see the IP of your PC in the IPv4 Address field, in the example case is 192.168.1.128. Use that IP to configure the Cloud Upload settings through the GroLab Software as shown in the below picture.


CLOUD SETTINGS

Cloud Server: 192.168.1.128
Port: 48680
Cloud Endpoint: /v2/feeds/
Cloud Key: 1234567
Cloud Feed: 1234567
Time Interval: Hours 0 Minutes 1

At the point you submit the settings, your GroNode should be ready to start sending the data to the PC. In this example, we configure a 1 minute time interval, so it will take just 1 minute to start seeing some data being printed on the console of the "Hercules SETUP utility", as shown in the following picture.



Received data

```
1","GROWID_224","SCHEDULE","INPUT"]},{"id":"A_
5","current_value":"0","tags":
["PH_Correct","BOT_IN_ID_4","BOT_OUT_ID_
2","INPUT_ID_0","OUTPUT_ID_0","AREAID_
1","GROWID_224","SCHEDULE","INPUT"]},{"id":"A_
6","current_value":"0","tags":
["Alarm","BOT_IN_ID_1","BOT_OUT_ID_1","INPUT_ID_
0","OUTPUT_ID_3","AREAID_0","GROWID_
0","TURN_ON","OUTPUT"]},{"id":"A_
```

Sent data

Server status
Port: 48680

TEA authorization
TEA key
1: 01020304 3: 090A0B0C
2: 05060708 4: 0D0E0F10
☐ Client authorization

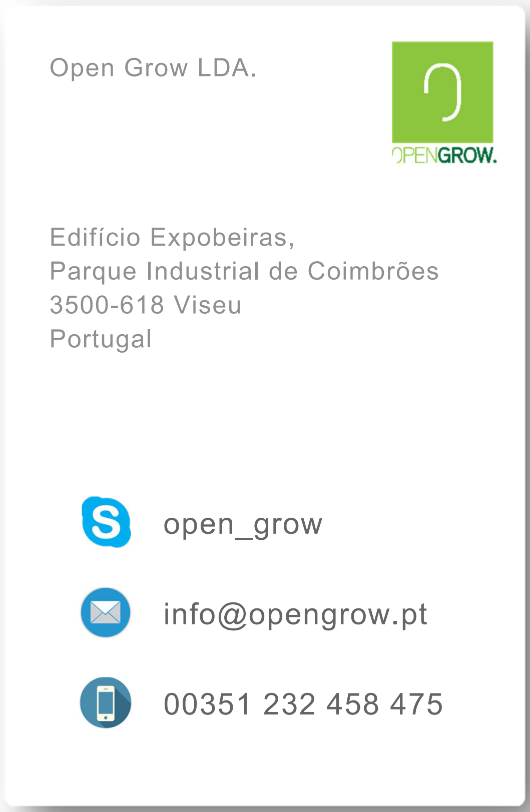
Client connection status

```
12:29:04 PM:  192.168.1.103 Client
12:30:29 PM:  192.168.1.103 Client
```

Clients count: 0

Cursor decode
HEX: 5F
Decimal: 95
Decoder Input:

Server settings
☐ Server echo
☐ Redirect to UDP



Open Grow LDA.

Edifício Expobeiras,
Parque Industrial de Coimbrões
3500-618 Viseu
Portugal

open_grow

info@opengrow.pt

00351 232 458 475