

COOPER is a flexible sensor platform for the Internet-of-Things (IoT) applications. It has been designed to meet the highest quality standards for environmental monitoring, low-power operation from batteries and wireless communication. Customized sensor and firmware configuration is available on request including silicone strap color customization, logo printing (using colored UV printing technology) and high-speed CNC milling in the top cover. The device is powered from 3x AA Alkaline 1.5V cells and it can provide up to 3 years of service time from the battery installation. Service time is given by the sensor measurement frequency, chosen communication technology, and communication interval.

Integrated Sensors

- ☐ Acceleration
- ☐ Acoustic noise
- ☐ Altitude (sea level)
- ☐ Atmospheric pressure
- ☐ Ambient temperature
- ☐ Battery voltage
- ☐ CO₂ concentration
- ☐ Light intensity
- ☐ Motion detection (PIR)
- ☐ Relative air humidity
- ☐ VOC concentration



COOPER Dongle as a gateway device.



Basic Features

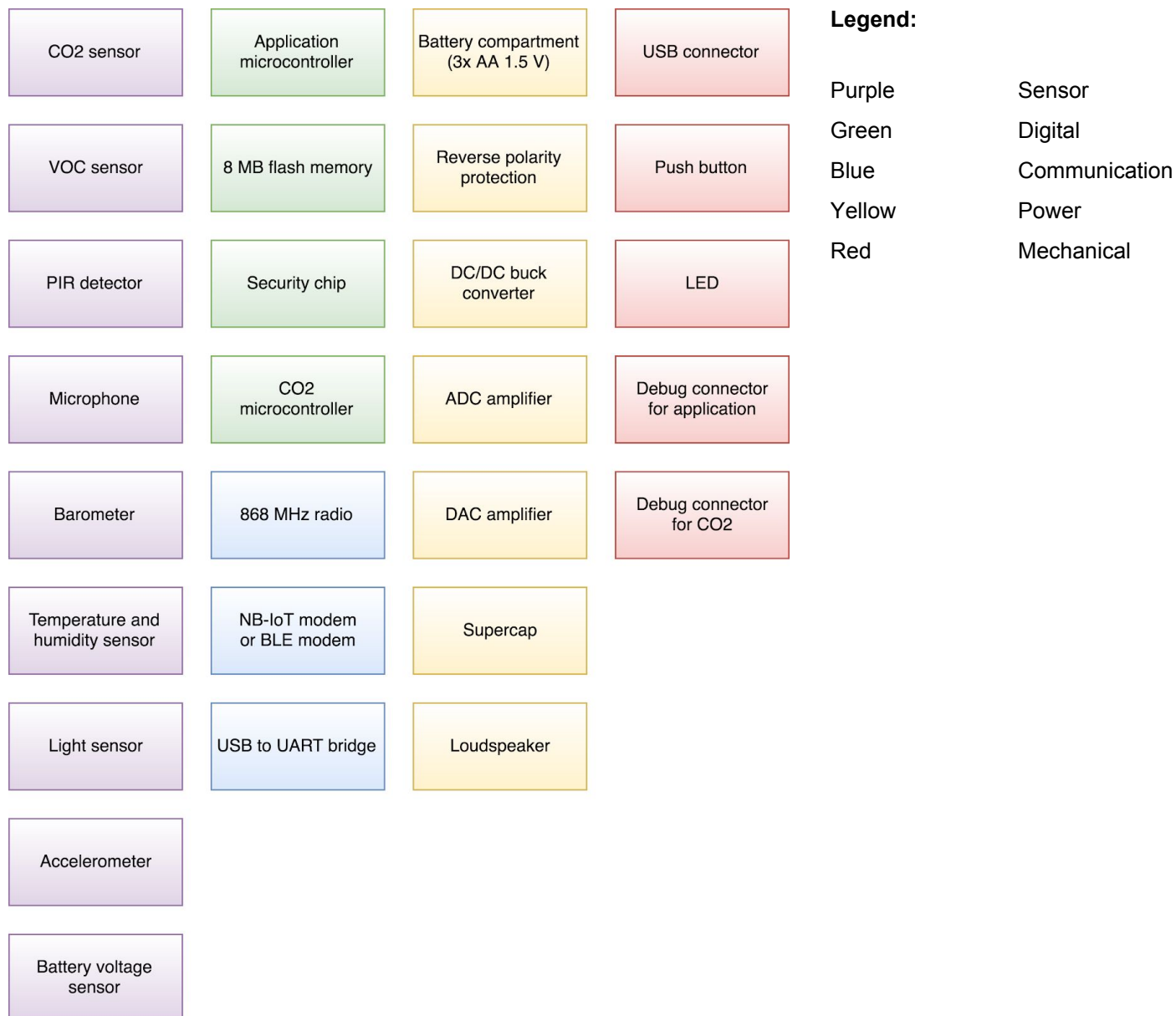
- ☐ Battery operation (3x AA 1.5V)
- ☐ 868 MHz ISM low-power radio (encrypted communication - AES-128)
- ☐ NB-IoT communication (optional)
- ☐ LoRa communication (optional)
- ☐ Sigfox communication (optional)
- ☐ Bluetooth Low Energy (optional)
- ☐ Push button with LED backlight (optional)
- ☐ Loudspeaker with speech output (optional)
- ☐ USB interface (with power support)
- ☐ Firmware updates over USB
- ☐ Dimensions (mm) 145(l) x 88(w) x 33(h)

Applications

- ☐ Environmental monitoring
- ☐ HVAC systems
- ☐ Facility management
- ☐ Smart home
- ☐ Schools
- ☐ Education
- ☐ Development kit for IoT

Block Diagram

The following diagram illustrates basic sensor device blocks:



Basic Parameters

Parameter	Min.	Typ.	Max.	Unit
Operating supply voltage	3.1		5.5	V
Idle current consumption		40		μA
Operating temperature range ⁽¹⁾	0		+50	° C
Operating temperature range ⁽²⁾	-20		+70	° C
Storage temperature	-20		+70	° C
Operating relative air humidity	0		95	%
Storage relative air humidity	0		95	%

Operational Description

The sensor device operates from 3x AA 1.5 V Alkaline cells for 3 years and communicates to the gateway device using 868 MHz ISM frequency. The battery voltage is always monitored and reported to the gateway device. The sensor device features a USB connector for host communication (using AT commands) and firmware update. Optionally, the sensor device can be also powered from this USB connector and batteries do not have to be inserted (if they are inserted at the same time with the USB cable, they are not discharged and the USB power is used).

When the device is powered, the red LED will turn on for 1 second. The device starts normal operation right away and scans sensors as described in the following table:

Sensor	Unit	Min	Max	Filter type	Interval
Acceleration	g	-2	2	Average (8 samples)	5 seconds
Acoustic noise	dBA	0	150	RMS (4096 samples)	120 seconds
Altitude	m	-698	11,775	Average (8 samples)	60 seconds
Ambient temperature	° C	-40	125	Average (8 samples)	30 seconds
Atmospheric pressure	Pa	20	110,000	Average (8 samples)	60 seconds
Battery voltage	V	0	5.5	Average (8 samples)	60 seconds
CO2 concentration	ppm	0	10,000	None (internal IIR)	120 seconds
Illuminance	lux	0	83,000	Average (8 samples)	30 seconds
Motion detection ¹	-	0	65,535	None	Always
Orientation ²	-	1	6	Average (8 samples)	5 seconds
Relative air humidity	%	0	100	Average (8 samples)	30 seconds
VOC concentration	ppb	0	60,000	Average (8 samples)	30 seconds

Note 1: Motion count is a counter that only increments and normally overflows. The number of events in a given time window is detected on the receiver's side.

Note 2: The number corresponds to a throwing dice.

The measured data are transmitted to the gateway device every 5 minutes. The transmission can be immediately started by a button press (this is indicated by 1 second LED pulse). Each unit has a unique 48-bit serial number, which is used for communication. It is printed on the back side of the device and can be read using a barcode scanner.

In order to receive data on the gateway device, the sensor has to be paired in the gateway. This can be achieved either using AT commands, or Cooper Control Tool (both described below).

CO₂ Sensor Calibration

You can calibrate the sensor in the outdoors so it will set up its own lowest CO₂ level to 400 ppm.

Procedure to start the calibration:

1. Press and hold the button for 6 seconds (button is located next to the USB connector).
2. Release the button.
3. The LED will start to blink every 5 seconds.
4. Place the Cooper outdoor to the fresh air.
5. The complete calibration takes 72 minutes.
6. Calibration is done when the LED stops blinking.

AT Commands

Both sensor device and gateway device can be controlled using AT commands over the USB interface. The USB interface provides virtual serial port (USB CDC) interface which is always available when the USB cable is plugged in.

These are the basic AT command configuration and usage rules:

- The device is further in AT commands context referred to as MT (Mobile Terminal).
- The host is further in AT commands context referred to as TE (Terminal Equipment).
- Serial port parameters are 115200 Bd, 8 data bits, no parity, 1 stop bit.
- From TE it is possible to use any new line delimiter combination combination of <CR> (0x0d) or <LF> (0x0a).
- The new line is always delimited from MT using <CR><LF>.
- MT supports the possibility to clear input buffer using <ESC> key (0x1b).
- MT supports the correction of the last character using <BS> key (0x08).
- Every AT command starts with the "AT" string prefix.
- Some commands are part of existing standards and inspired by existing equipment.
- Vendor-specific AT commands always start with "AT\$".
- Every command is terminated with "OK" in case of success, or with "ERROR" in case of failure.

- All commands are case-sensitive and have to follow patterns exactly described below.
- URC messages (Unsolicited Result Codes) is a kind of message, which can appear from MT at any time (these are asynchronous events).
- Messages that appear on the serial port and start with the "#" character, represent logging messages, which can (like URC messages) appear at any time.

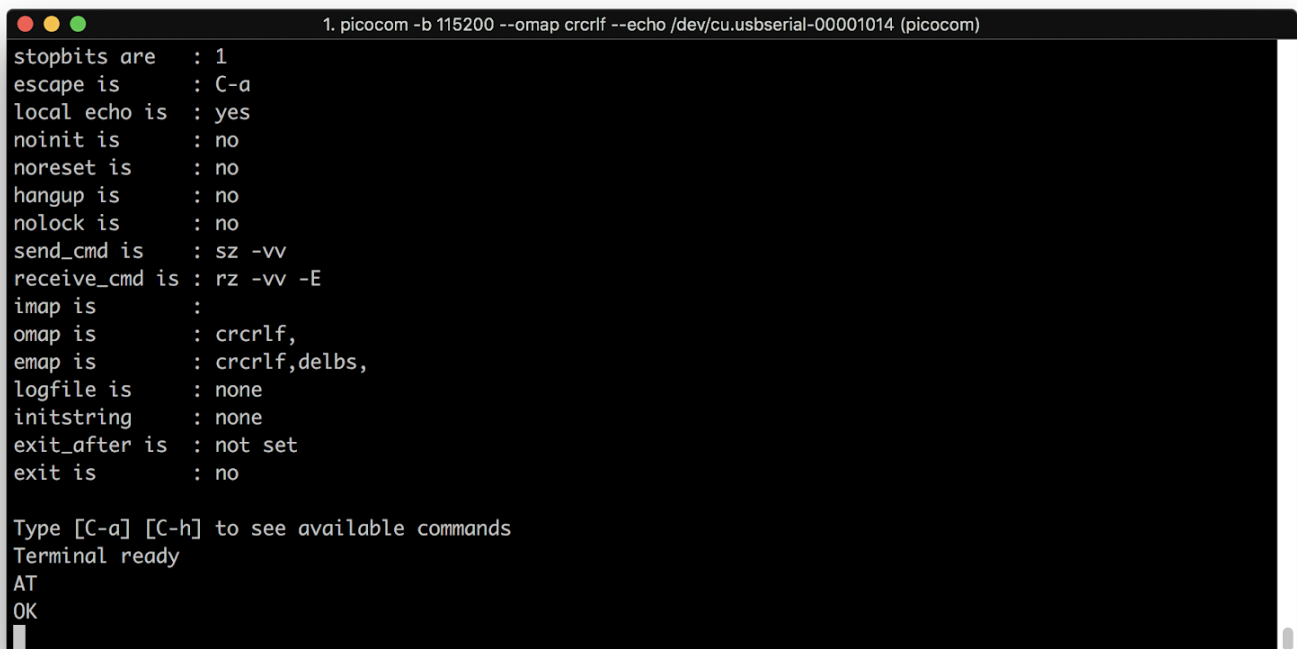
Interfacing AT Commands

There are many serial port terminals that can be used to interface AT commands. For example, on macOS and Linux, you can use **picocom**. On macOS this can be easily installed using Homebrew:

```
brew install picocom
```

Then you can start the program as:

```
picocom -b 115200 --omap crclrf --echo /dev/cu.usbserial-00001014
```



```
1. picocom -b 115200 --omap crclrf --echo /dev/cu.usbserial-00001014 (picocom)
stopbits are : 1
escape is : C-a
local echo is : yes
noinit is : no
noreset is : no
hangup is : no
nolock is : no
send_cmd is : sz -vv
receive_cmd is : rz -vv -E
imap is :
omap is : crclrf,
emap is : crclrf,delbs,
logfile is : none
initstring : none
exit_after is : not set
exit is : no

Type [C-a] [C-h] to see available commands
Terminal ready
AT
OK
```

Common AT Commands

The following AT commands are supported on both sensor and gateway device.

AT - Communication test

This command only serves the purpose to test communication with MT.

Format: AT

Example:

Command: AT

Response: OK

AT&F - Restore configuration to factory defaults

This command puts the device configuration to factory default settings. It does not store the settings to non-volatile memory (see AT&W below).

Format: AT&F

Example:

Command: AT&F

Response: OK

AT&W - Store configuration to non-volatile memory

This command saves the current configuration settings to a non-volatile (EEPROM) memory.

Format: AT&W

Example:

Command: AT&W

Response: OK

ATI - Request product information

This command reads the compact product information of MT.

Format: ATI

Example:

Command: ATI

Response: COOPER R1.1 0.1.0

 OK

AT+CGMI - Request manufacturer identification

This command reads the manufacturer identification of MT.

Format: AT+CGMI

Example:

Command: AT+CGMI

Response: HARDWARIO s.r.o.

 OK

AT+CGMM - Request model identification

This command reads the model identification of MT.

Format: AT+CGMM

Example:

Command: AT+CGMM

Response: COOPER R1.1

 OK

AT+CGMR - Request revision identification

This command reads the serial number of MT.

Format: AT+CGMR

Example:

Command: AT+CGMR

Response: 0.1.0

OK

AT+CGSN - Read serial number identification

This command reads the serial number (id) of MT.

Format: AT+CGSN

Example:

Command: AT+CGSN

Response: 0123456789012345

OK

AT+CLAC - List available AT commands

This command lists all available AT commands.

Format: AT+CLAC

Example:

Command: AT+CLAC

Response: AT

AT&F

...

OK

AT\$CHANNEL - Set/read radio channel

This command allows to set or read radio channel. The supported range of channels is 0..19. The command affects configuration settings, which have to be permanently stored using the AT&W command.

Format: AT\$CHANNEL?

 AT\$CHANNEL=<channel>

Example:

Command: AT\$CHANNEL?

Response: \$CHANNEL: 0

 OK

Command: AT\$CHANNEL=1

Response: OK

AT\$KEY - Set encryption key (AES-128)

This command allows setting key for encrypted radio communication. It expects the key in the hexadecimal format (32 characters). The command affects configuration settings, which have to be permanently stored using the AT&W command.

Format: AT\$KEY=<key>

Example:

Command: AT\$KEY=f2e891014be3e94151c66249203e2246

Response: OK

AT\$STATUS - Retrieve device status

This command retrieves the current device status.

Format: AT\$STATUS

Example:

Command: AT\$STATUS

Response: \$STATUS: "Acceleration",0.11,0.00,0.96

\$STATUS: "Altitude",321.8

\$STATUS: "CO2 Concentration"

\$STATUS: "Humidity",48.4

\$STATUS: "Illuminance",28

\$STATUS: "Orientation",1

\$STATUS: "Press Count",0

\$STATUS: "Pressure",97521

\$STATUS: "Sound Level",0

\$STATUS: "Temperature",24.64

\$STATUS: "VOC Concentration"

\$STATUS: "Voltage",0.01

OK

AT\$SEND - Send data

This command send data immediately.

Format: AT\$SEND

Example:

Command: AT\$SEND

Response: OK

AT\$PULSE - Pulse LED

This command pulses LED on MT for 3 seconds.

Format: AT\$PULSE

Example:

Command: AT\$PULSE

Response: OK

AT\$BEEP - Beep speaker

This command run beep speaker on MT for 3 seconds.

Format: AT\$BEEP

Example:

Command: AT\$BEEP

Response: OK

AT\$HELP - List help

This command lists AT command help.

Format: AT\$HELP

Example:

Command: AT\$HELP

Response: AT&F Restore configuration to factory defaults
AT&W Store configuration to non-volatile memory
...
AT\$HELP Print this help
OK

AT\$LOCK - Lock configuration

This command allows setting password and lock configuration to the read-only mode. It expects a parameter of up to 12 ASCII characters - all printable characters inside the quotation marks are allowed. The command affects configuration settings, which have to be permanently stored using the AT&W command.

Format: AT\$LOCK="<password>"

Example:

Command: AT\$LOCK="jzvuK5oTwBs6"

Response: OK

AT\$UNLOCK - Unlock configuration

This command allows removing password and unlock configuration. It expects a parameter of up to 12 characters - all printable characters inside the quotation marks are allowed. The command affects configuration settings, which have to be permanently stored using the AT&W command.

Format: AT\$UNLOCK="<password>"

Example:

Command: AT\$UNLOCK="jzvuK5oTwBs6"

Response: OK

AT\$CONFIG - Configuration

This command allows configuration some parameters. The command affects configuration settings, which have to be permanently stored using the AT&W command.

Parameters:

- "Report Interval", uint: seconds, default: 300, min: 30, max: 65535

Format: AT\$CONFIG

AT\$CONFIG="<name>",<value>

Example read:

Command: AT\$CONFIG

Response: \$CONFIG: "Report Interval",300

OK

Example write:

Command: AT\$CONFIG="Report Interval",600

Response: OK

Gateway AT Commands

The following AT commands are supported on the gateway device.

AT\$LIST - List nodes

This command prints whitelist of all the nodes.

Format: AT\$LIST

Example:

Command: AT\$LIST

Response: 0123456789012345,"Room 1"
5432109876543210,""
OK

AT\$ATTACH - Attach new node

This command adds a new node to the whitelist. The command affects configuration settings, which have to be permanently stored using the AT&W command.

Format: AT\$ATTACH=<id>,<key>,"<alias>"

AT\$ATTACH=<id>,<key>

Example:

Command: AT\$ATTACH=0123456789012345,f2e891014be3e94151c66249203e2246,"Room 1"

Response: OK

AT\$DETACH - Detach existing node

This command removes the existing node from the whitelist. The command affects configuration settings, which have to be permanently stored using the AT&W command.

Format: AT\$DETACH=<id>

Example:

Command: AT\$DETACH=0123456789012345

Response: OK

AT\$PURGE - Remove all paired nodes

This command deletes all paired nodes. The command affects configuration settings, which have to be permanently stored using the AT&W command.

Format: AT\$PURGE

Example:

Command: AT\$PURGE

Response: OK

Gateway URC Messages

\$BOOT - Device restart

This URC informs TE about the MT restart.

Format: \$BOOT

\$PRESS - Push button press

This URC informs about the push button press event. It provides a number of press events.

Format: \$PRESS: <count>

\$RECV - Message from node

This URC provides extracted message information from the node.

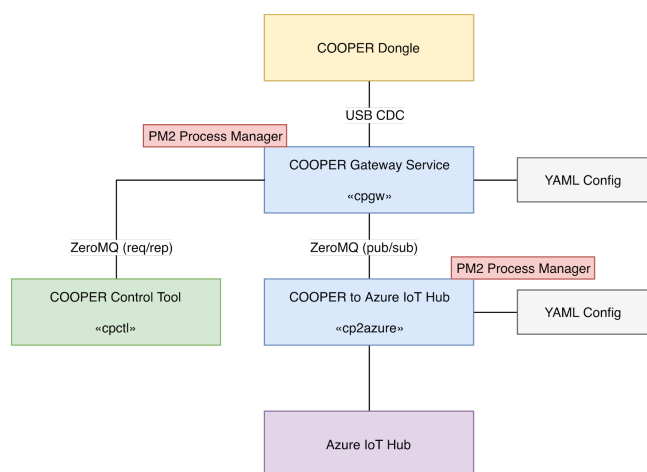
Format: \$RECV: <rss>,<id>,<sequence>,<altitude>, *(there is no new line in communication)*
 <co2_conc>,<humidity>,<illuminance>, *(there is no new line in communication)*
 <motion_count>,<orientation>,<press_count>, *(there is no new line in communication)*
 <pressure>,<sound_level>,<temperature>, *(there is no new line in communication)*
 <voc_conc>,<voltage>

Note: If some parameter is missing, it means it is invalid (sensor is not yet ready, or sensor is broken).

Hub Software

In the context of the COOPER ecosystem, the word "hub" means the IoT edge gateway environment - e.g. Raspberry Pi. Typically it always-on Linux machine.

Several tools are available for fast integration. All the tools are implemented and compatible with Python 3.5+ and available through **PyPI** (Python Package Index). The architecture of the tools is best described using the following block diagram:



Physical **COOPER Dongle** (gateway device) is connected to the USB port of the host machine. **COOPER Gateway Service** (cpgw) connects to the virtual serial port of the **COOPER Dongle** and provides 2 optional **ZeroMQ** sockets:

- Socket **PUB/SUB** for publishing of the data from devices.
- Socker **REQ/REP** for accepting commands from other clients.

The advantage of **ZeroMQ** is, that the whole system is distributable and can run across multiple machines as well as locally. Moreover, it addresses many reliability or difficult to solve issues related to the raw TCP/Unix sockets (reconnects, timeouts, fragmentation, etc.). **ZeroMQ** offers binding for pretty much every popular programming language.

All the tools in the ecosystem support configuration using a **YAML file**. YAML is a simple structure, compatible with JSON (actually, it is a superset of JSON), but better for readability.

```
device: /dev/ttyUSB0
zmq:
  publisher:
    host: 0.0.0.0
    port: 5680
  dispatcher:
    host: 0.0.0.0
    port: 5681
```

This is the example YAML configuration file for the **COOPER Gateway Service** (cpgw).

Note: The indentation must be followed carefully (2 spaces per indentation level).

Source repositories are available at HARDWARIO GitHub organization: <https://github.com/hardwario>

All tools can be installed from PyPI on Windows, macOS and Linux using the following command:

```
sudo pip3 install --upgrade cpgw cpctl cp2azure
```

It is recommended to run services which are supposed to run in the background using the [PM2 Process Manager](#).

All tools use the common logging mechanism (standard **logging** library available in Python) and can be configured using the YAML configuration files.

COOPER Control Tool

This is a CLI (command line interface) application, which allows to:

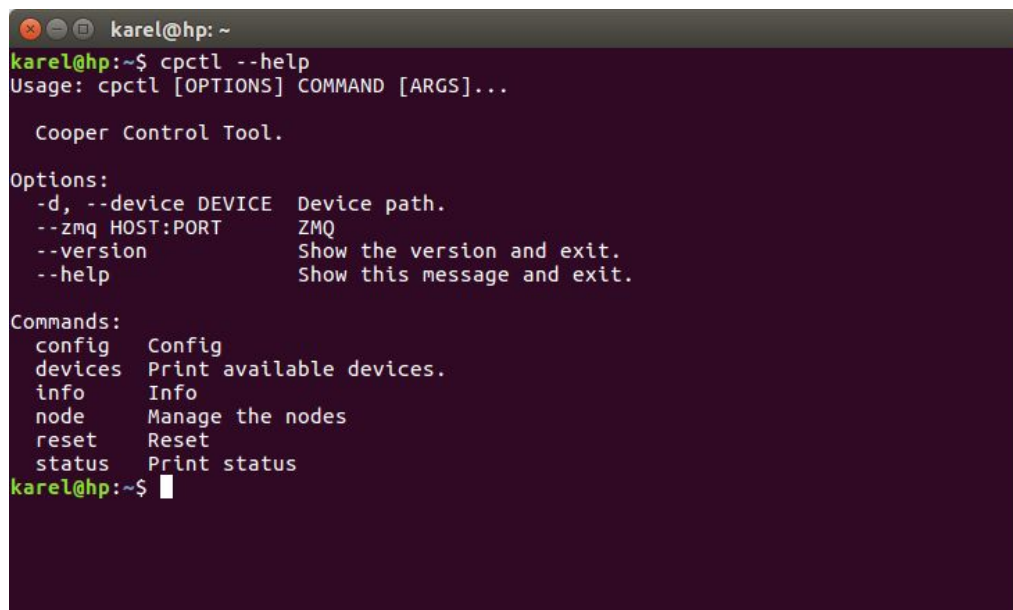
1. Send commands to **COOPER Gateway Service** using ZeroMQ.
2. Send commands to **COOPER Dongle** via serial port.
3. Send commands to **COOPER Sensor** via serial port.

You can list connected devices (via serial port) using this command:

```
cpctl devices
```

For more information, please use the built-in help:

```
cpctl --help
```



```
karel@hp: ~  
karel@hp:~$ cpctl --help  
Usage: cpctl [OPTIONS] COMMAND [ARGS]...  
  
Cooper Control Tool.  
  
Options:  
  -d, --device DEVICE  Device path.  
  --zmq HOST:PORT       ZMQ  
  --version             Show the version and exit.  
  --help               Show this message and exit.  
  
Commands:  
  config  Config  
  devices Print available devices.  
  info    Info  
  node    Manage the nodes  
  reset   Reset  
  status  Print status  
karel@hp:~$
```

Set random AES-128 key to node and add this node to dongle use zmq:

```
cpctl config key --generate --add-node-to-dongle-zmq 127.0.0.1:5681
```

COOPER to Azure IoT Hub Bridge

This is a service application which works as a bridge between the ZeroMQ pub/sub socket and the **Azure IoT Hub**. It requires a **Device Connection String** in the configuration file and, of course, ZeroMQ socket configuration.

On reception of the JSON message on the pub/sub socket, it assembles message for the **Azure IoT Hub** and sends it.

Installation Cheatsheet

```
sudo apt-get install libboost-python-dev
```

```
sudo pip3 install --upgrade cpgw
```

```
sudo pip3 install --upgrade cpctl
```

```
sudo pip3 install --upgrade cp2azure
```

```
sudo mkdir -p /etc/cooper/
```

```
sudo nano /etc/cooper/cpgw.yml
```

```
device: /dev/ttyUSB0
```

```
zmq:
```

```
  publisher:
```

```
    host: 127.0.0.1
```

```
    port: 5680
```

```
  dispatcher:
```

```
    host: 127.0.0.1
```

```
    port: 5681
```

```
sudo nano /etc/cooper/cp2azure.yml
```

```
zmq:
```

```
  host: 127.0.0.1
```

```
  port: 5680
```

```
  timeout: 5000
```

```
azure:
```

```
  connection_string: <INSERT YOUR CONNECTION STRING HERE>
```

```
pm2 start /usr/bin/python3 --name "cpgw" -- /usr/local/bin/cpgw -c /etc/cooper/cpgw.yml
```

```
pm2 start /usr/bin/python3 --name "cp2azure" -- `which cp2azure` -c /etc/cooper/cp2azure.yml
```

```
pm2 save
```

Troubleshooting

If the devices do not report data, please follow this procedure:

1. Check if the **cpgw** process is running.
2. Check if the proxy service running (e.g. **cp2azure**).
3. Check if the hub has a working internet connectivity.
4. Check if the device is operating:
 - a. Press the button on it (can be located next to the USB connector).
 - b. Check if red LED lights up after the button press.
 - c. If you cannot see the LED response, try to replace the batteries.
5. Check if data from devices are being received (use **cp2stdout** tool).
6. Check the logs of the corresponding services, e.g.:

```
journalctl -u cpgw.service -f  
journalctl -u cp2azure.service -f
```

If none of the above helps, please contact our technical support at support@hardwario.com.

Contact Information

The product is designed, manufactured and sold by HARDWARIO.

Website: www.hardwario.com

E-mail: support@hardwario.com

Company address:

HARDWARIO s.r.o.
U Jezu 525/4
460 01 Liberec
Czechia

CID: 04998511

VAT: CZ04998511

The company is registered in the Commercial Register, maintained by the Regional Court in Ústí nad Labem, Section C, File 37399.



Revision History

Rev 1.0	August 30, 2018	<ul style="list-style-type: none">• Initial revision
Rev 1.1	September 18, 2018	<ul style="list-style-type: none">• Updated AT commands
Rev 1.2	October 3, 2018	<ul style="list-style-type: none">• Added Gateway Software description
Rev 1.3	November 20, 2018	<ul style="list-style-type: none">• Updated AT commands• Updated Software description
Rev 1.4	November 23, 2018	<ul style="list-style-type: none">• Added Troubleshooting chapter
Rev 1.5	April 24, 2019	<ul style="list-style-type: none">• Add CO₂ calibration description