ICOtronic Package Documentation

MyTooliT

Contents

1 Introduction

ICOtronic is a

- Python library (based on `python-can`) and
- a collection of tools and scripts

for the ICOtronic system. Currently the main purpose of the software is:

- **Data collection**
  - directly via the API or
  - the script `icon`
- **Testing** the functionality of
  - a Stationary Transceiver Unit (STU) or
  - a sensor node, such as a
    - Sensory Holder Assembly (SHA)/Sensory Tool Holder (STH) or
    - Sensory Milling Head (SMH)

The software reads data from the Stationary Transceiver Unit (STU) via CAN using the MyTooliT protocol. The STU itself reads from and writes data to the sensor nodes via Bluetooth.

1.1 Requirements

1.1.1 Hardware

In order to use the ICOtronic system you need at least:

- a PCAN adapter:

  including:

  - power injector, and
  - power supply unit (for the power injector):

    Note: Other CAN adapters supported by python-can should work as well. However, you need to update the configuration for the CAN connection accordingly.
- a Stationary Transceiver Unit:
- a sensor node, such as a Sensory Tool Holder:

Figure 1: PCAN Adapter

Figure 2: Power Injector

1.1.1.1 Setup

1. Connect the power injector
    1. to the PCAN adapter, and
    2. the power supply unit.
2. Connect the USB connector of the PCAN adapter to your computer.
3. Make sure that your sensor node (SHA/STH/SMH) is connected to a power source. For an STH this usually means that you should check that the battery is (fully) charged.

1.1.2 Software

1.1.2.1 Python The ICOtronic package supports `3.10` and later. We recommend you use the 64-bit version of Python.

You can download Python here. When you install the software, please do not forget to enable the checkbox **"Add Python to PATH"** in the setup window of the installer.

1.1.2.2 PCAN Driver To communicate with the STU you need a driver that works with the PCAN adapter. The text below describes how to install/enable this driver on

- Linux,
- macOS, and
- Windows.

1.1.2.2.1 Linux You need to make sure that your CAN adapter is available via the SocketCAN interface.

The following steps describe one possible option to configure the CAN interface on (Fedora, Ubuntu) Linux **manually**.

1. Connect the CAN adapter to the computer that runs Linux (or alternatively a Linux VM)
2. Check the list of available interfaces:

    ```
    networkctl list
    ```

    The command output should list the CAN interface with the name `can0`
3. Configure the CAN interface with the following command:

    ```
    sudo ip link set can0 type can bitrate 1000000
    ```

4. Bring up the CAN interface

    ```
    sudo ip link set can0 up
    ```

You can also configure the CAN interface **automatically**. For that purpose please store the following text:

```
[Match]
Name=can*

[CAN]
BitRate=1000000
```

in a file called `/etc/systemd/network/can.network`. Afterwards enable `networkd` and reload the configuration with the commands:

```
sudo systemctl enable systemd-networkd
sudo systemctl restart systemd-networkd
# Note: The command `networkctl reload` only works in systemd 244 or newer
sudo networkctl reload || sudo systemctl reload systemd-networkd
```

You can check the status of the CAN connection with the command:

```
networkctl list
```

If everything works as expected, then the output of the command should look similar to the text below:

```
IDX LINK   TYPE     OPERATIONAL SETUP
  …
  7 can0   can      carrier     configured
```

Sources:

- SocketCAN device on Ubuntu Core
- Question: How can I automatically bring up CAN interface using netplan?
- networkd › systemd › Wiki › ubuntuusers

1.1.2.2.2 macOS On macOS you can use the PCBUSB library to add support for the PCAN adapter. For more information on how to install this library, please take a look at the issue "How to install the PCBUSB-Library on Mac".

1.1.2.2.3 Windows You can find the download link for the PCAN Windows driver here. Please make sure that you include the "PCAN-Basic API" when you install the software.

1.1.2.3 Simplicity Commander (Optional) For the production tests that require a firmware flash you need to **either** install

- Simplicity Studio or
- Simplicity Commander.

If you choose the first option, then please make sure to install the Simplicity Commander tool inside Simplicity Studio.

1.1.2.3.1 Linux Please add the path to `commander` to the list `commands` → `path` → `linux` in the configuration.

1.1.2.3.2 macOS If you install Simplicity Studio or Simplicity Commander in the standard install path (`/Applications`) you do not need to change the config. If you put the application in a different directory, then please add the path to `commander` to the list `commands` → `path` → `mac` in the configuration.

1.1.2.3.3 Windows

- If you installed Simplicity Studio (including Simplicity Studio) to the standard location, then you do not need to change the configuration.
- If you download Simplicity Commander directly, then the tests assume that you unzipped the files into the directory `C:\SiliconLabs\Simplicity Commander` or `C:\Program Files\Simplicity Commander`.
- If you did not use any of the standard install path, then please add the path to `commander.exe` to the list `commands` → `path` → `windows` in the configuration.

  Notes:

  - You also need to install **J-Link** for Simplicity Commander to work with a USB programmer.
  - If your device does not show up in Simplicity Commander and only appears as **USB BULK device in the Windows device manager**, then please follow the steps described in the Segger Knowledge Base.

1.1.2.3.4 Additional Notes

- If you **do not want to change the config**, and Simplicity Commander (`commander`) is not part of the standard search locations for your operating system, then please make sure that `commander` is accessible via the `PATH` environment variable.
- Please note, that you **do not need to install Simplicity Commander** if you just want to **collect data** with the ICOtronic library.

1.2 Install

Please use the following command:

```
pip install icotronic
```

to install the latest official version of the ICOtronic library from PyPi. Afterwards you can use the various scripts included in the package.

1.2.1 Install the Package Using Windows Terminal

1. Install (Windows) Terminal if you have not done so already; On Windows 11 this application should be installed by default.
2. Open Terminal
3. Copy and paste the following text into the Terminal

   ```
   pip install icotronic
   ```

4. Press Return
5. Wait until the install process finished successfully

1.2.2 Install the Development Version of ICOtronic

Note: Please only use the command below, **if you know what you are doing**!

```
pip install 'git+https://github.com/MyTooliT/ICOtronic'
```

1.2.3 Troubleshooting

1.2.3.1 Import Errors If one of the scripts fails with an error message that looks similar to the following text on Windows:

```
Traceback (most recent call last):
…
    from numexpr.interpreter import MAX_THREADS, use_vml, __BLOCK_SIZE1__
ImportError: DLL load failed while importing interpreter: The specified module could not be found.

DLL load failed while importing interpreter: The specified module could not be found.
```

then you probably need to install the "Microsoft Visual C++ Redistributable package". You can download the latest version

- for the `x64` architecture, i.e. for AMD and Intel CPUs, here and
- for the `ARM64` architecture here.

1.2.3.2 Insufficient Rights **If you do not have sufficient rights** to install the package you can also try to install the package in the user package folder:

```
pip install --user icotronic
```

1.2.3.3 Unable to Locate HDF5 The installation of the ICOtronic package might fail with an error message that looks like this:

```
… implicit declaration of function 'H5close'
```

If you uses Homebrew on an Apple Silicon based Mac you can use the following commands to fix this problem:

```
pip uninstall -y tables
brew install hdf5 c-blosc2 lzo bzip2
export BLOSC_DIR=/opt/homebrew/opt/c-blosc
export BZIP2_DIR=/opt/homebrew/opt/bzip2
export LZO_DIR=/opt/homebrew/opt/lzo
export HDF5_DIR=/opt/homebrew/opt/hdf5
pip install --no-cache-dir tables
```

1.2.3.4 HDF5 Library Not Loaded Some of the ICOtronic scripts might fail with an error message that looks like this on macOS:

```
    Library not loaded: /opt/homebrew/opt/hdf5/lib/libhdf5.….dylib
```

In that case you might have installed an outdated cached version of PyTables. You should be able to fix this issue using the same steps as described above.

1.2.3.5 Unable to open OpenBLAS library

If one of the ICOtronic scripts fails with the error message:

> ImportError: libopenblas.so.0: cannot open shared object file: No such file or directory

on Raspbian (or some other GNU/Linux version based on Debian) then you probably need to install the OpenBLAS library:

```
sudo apt-get install libopenblas-dev
```

1.2.3.6 Unknown Command

If `pip install` prints **warnings about the path** that look like this:

> The script … is installed in `'…\Scripts'` which is not on PATH.

then please add the text between the single quotes (without the quotes) to your PATH environment variable. Here `…\Scripts` is just a placeholder. Please use the value that `pip install` prints on your machine. If

- you used the installer from the Python website (and checked "Add Python to PATH") or
- you used winget

to install Python, then the warning above should not appear. On the other hand, the **Python version from the Microsoft Store might not add the `Scripts` directory** to your path.

2 Data Collection

The ICOtronic package provides two different options to collect sensor data with the ICOtronic system:

1. Use the API to write your own Python scripts: To learn more about this option, please take a look at the API documentation.
2. Use the `icon` script to collect data: We will describe how to measure some basic data with this script in the text below

   Note: Since ICOn currently only provides very basic functionality for data collection you might be happier using one of the tools below. Both are currently based on an older (deprecated) version of this Python package:

   - ICOdaq: A closed source Electron application for Windows
   - ICOc: A text based UI for Windows

2.1 Sensor Node Identifiers

To connect to a sensor node (SHA, STH, SMH) you need to know an identifier for the node, which can be one of the following:

- Bluetooth advertisement **name**: up to 8 characters, no special characters allowed
- Bluetooth **MAC address**: 8 Bytes, usually written in hexadecimal format
- **Node number**: starting with 0, up to the number of available sensor nodes minus one (e.g. for 8 sensor nodes, the maximum sensor node number will be 7)

If you do not know any of the above identifiers for your sensor node you can use the command

```
icon list
```

to show the available nodes:

```
Name: Test-STH, Number: 0, MAC Address: 08-6B-D7-01-DE-81, RSSI: -51
```

In the example above, we see that one sensor node is available with the following identifiers:

- node number: `0`
- name: `Test-STH`
- MAC address: `08-6B-D7-01-DE-81`

> **Note:** The last value "-51" of the example output is the current received signal strength indication (RSSI).

2.2 Collecting Data

After you determined one of the identifiers (name, MAC address, node number) of your sensor node you can use the command:

```
icon measure
```

to collect measurement data. For example, to collect data from an STH with the name "Test-STH" for 10 seconds you can use the following command:

```
icon measure --name Test-STH --time 10
```

After the measurement took place the command will print some information about the collected data, including the location of the HDF5 measurement file:

```
Sample Rate: 9523.81 Hz
Data Loss: 0.0 %
Filepath: Measurement_2025-05-14_15-11-25.hdf5
```

By default measurement files will be stored in the current working directory with

- a name starting with the text `Measurement`
- followed by a date/time-stamp,
- and the extension `.hdf5`.

2.3 Measurement Data

> **Note:** ICOn **assumes** that the sensor node always measures **acceleration data** in multiples of the gravity of earth, commonly referred as g or g_0. While this is true for most of the sensor hardware (such as STHs), some sensor nodes measure other values, e.g. force or temperature. Even in this case the measurement software will (incorrectly) convert the data into multiples of g. We are **working on adding support for configuring the sensor type** in the firmware and the ICOtronic package to **fix this issue**.

To take a look at the measurement data you can use the tool HDFView.

> **Note:** Unfortunately you need to create a free account to download the program. If you do not want to register, then you can try if one of the accounts listed at BugMeNot works. Another option is to download the application from here. Just click on the folder for the latest version of the application (`hdfview-…`) and afterwards on the folder `bin` to see a list of compressed binaries (`.zip` & `.tar.gz`) for the different supported operating systems.

The screenshot below shows a measurement file produced by the ICOtronic library:

As you can see the table with the name `acceleration` stores the acceleration data. The screenshot above displays the metadata of the table. The most important meta attributes here are probably:

- `Start_Time`, which contains the start time of the measurement run in ISO format, and
- `Sensor_Range`, which specifies the range of the used acceleration sensor in multiples of earth's gravitation (g 9.81 m/s²).

After you double click on the acceleration table on the left, HDFView will show you the actual acceleration data:

As you can infer from the `x` column above the table shows the acceleration measurement data (in multiples of g) for a single axis. The table below describes the meaning of the columns:

Column	Description	Unit
counter	A cyclic counter value (0–255) sent with the acceleration data to recognize lost packets	–
timestamp	The timestamp for the measured value in microseconds since the measurement start	s
x	Acceleration in the x direction as multiples of earth's gravitation	g (9.81 m/s²)

Depending on your sensor and your settings the table might also contain columns for the `y` and/or `z` axis.

If you want you can also use HDFView to print a simple graph for your acceleration data. To do that:

1. Select the values for the the ordinate (e.g. click on the x column to select all acceleration data for the x axis)
2. Click on the graph icon in the top left corner
3. Choose the data for the abscissa (e.g. the timestamp column)
4. Click on the "OK" button

The screenshot below shows an example of such a graph:

For a more advanced analysis of the data files you can use our collection of measurement utilities ICOlyzer.

2.3.1 Adding Custom Metadata

Sometimes you also want to add additional data about a measurement. To do that you can also use HDFView. Since the tool opens files in read-only mode by default you need to change the default file access mode to "Read/Write" first:

1. Open HDFView

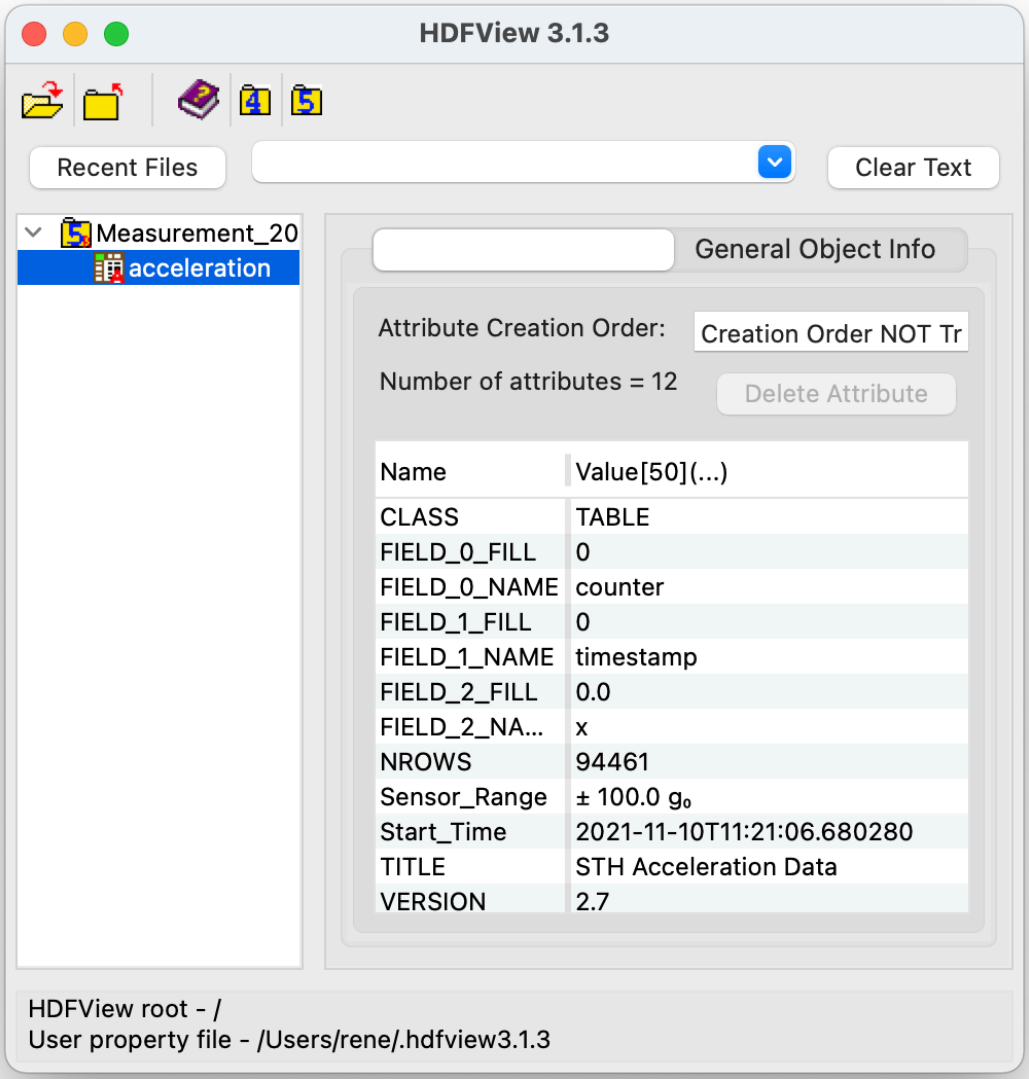

Figure 3: Main Window of HDFView

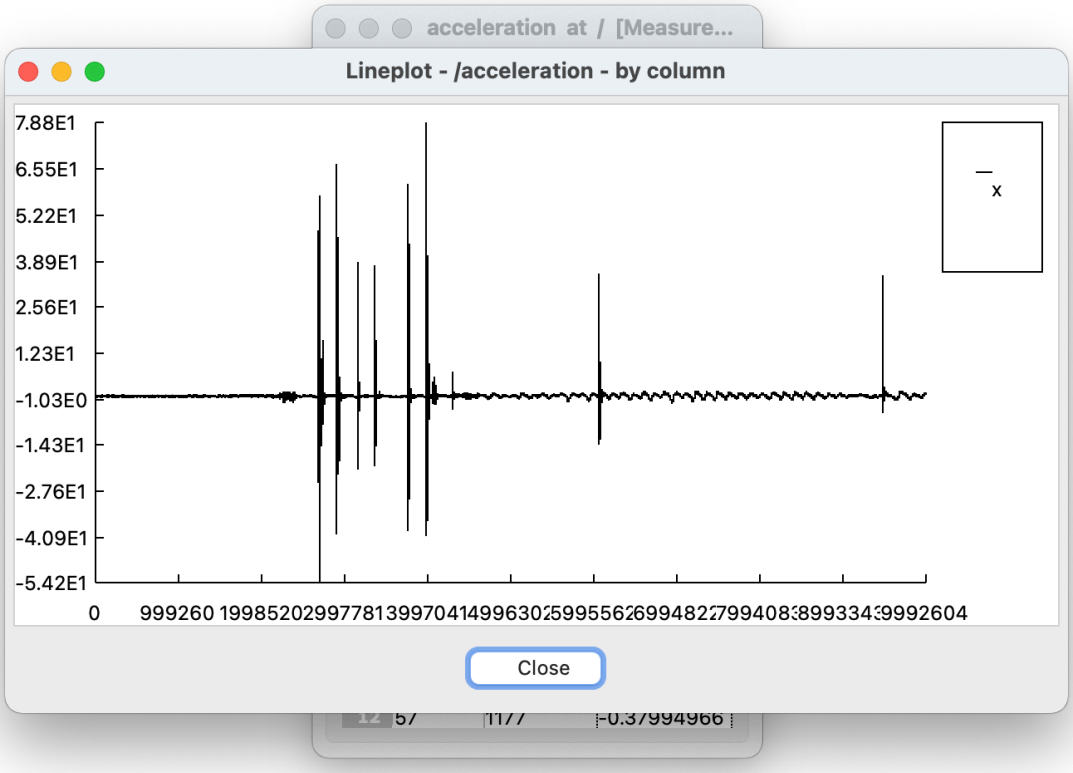

Figure 4: Acceleration Graph in HDFView

2. Click on “Tools” → “User Options”
3. Select “General Settings”
4. Under the text “Default File Access Mode” choose “Read/Write”
5. Close HDFView

Now you should be able to add and modify attributes. For example, to add a revolutions per minute (RPM) value of `15000` you can use the following steps:

1. Open the measurement file in HDFView
2. Click on the table “acceleration” in the left part of the window
3. In the tab “Object Attribute Info” on the right, click on the button “Add attribute”
4. Check that “Object List” contains the value “/acceleration”
5. Enter the text “RPM” in the field “Name”
6. In the field “Value” enter the text “15000”
7. The “Datatype Class” should be set to “INTEGER”
8. For the size (in bits) choose a bit length that is large enough to store the value. In our example everything equal to or larger than 16 bits should work.
9. Optionally you can also check “Unsigned”, if you are sure that you only want to store positive values
10. Click the button “OK”

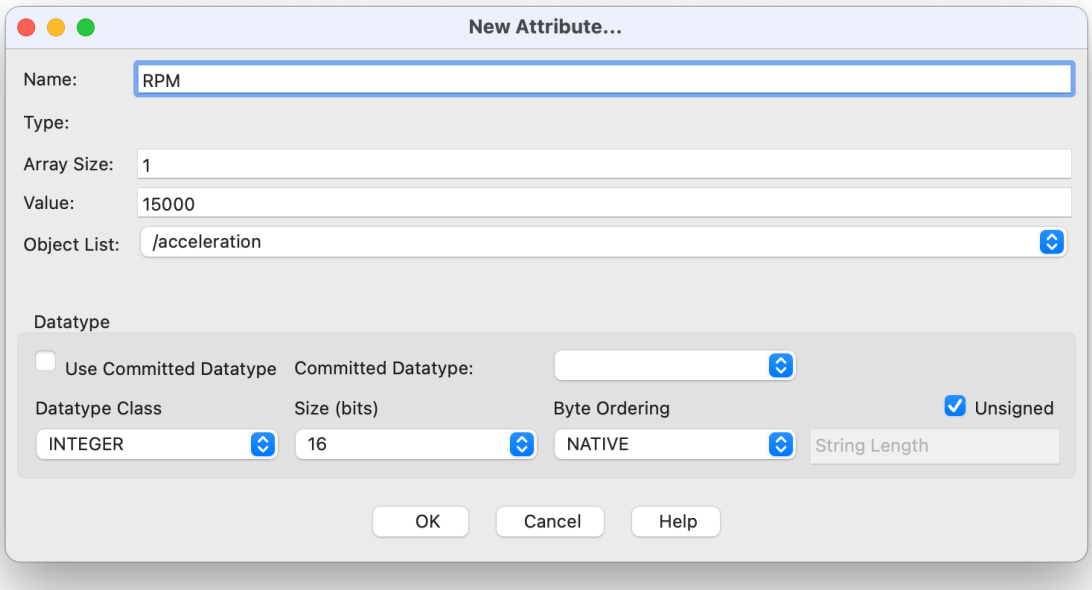

Figure 5: HDFView: RPM Attribute

Sometimes you also want to add some general purpose data. For that you can use the “STRING” datatype class. For example, to store the text “hello world” in an attribute called “Comment” you can do the following

1. Repeat steps 1. – 4. from above
2. Choose “STRING” as “Datatype Class”
3. Under “Array Size” choose a length that is large enough to store the text such as “1000” (every size larger than or equal to 11 characters should work)

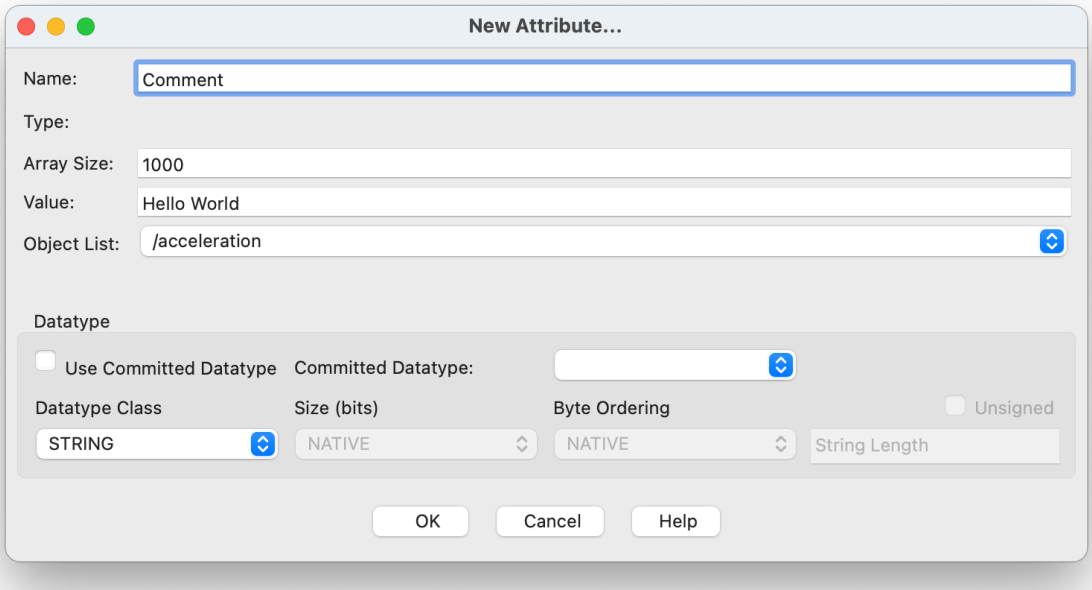

Figure 6: HDFView: Comment Attribute

4. Click the button “OK”

If you want you can also add multiline text. Since you can not add newlines using in HDFView directly, we recommend you open your favorite text editor to write the text and then copy and paste the text into the value field. HDFView will only show the last line of the pasted text. However, after you copy and paste the text into another program you will see that HDFView stored the text including the newlines.

3 Tutorials

3.1 Changing Configuration Values

> **Note:** If you only use the `icon` command line tool, then you most probably do not need to change the configuration at all.

All configuration options are currently stored in YAML files (handled by the configuration library Dynaconf). The default values are stored inside the package itself. If you want to overwrite or extend these values you should create a user configuration file. To do that you can use the command:

```
icon config
```

which will open the the user configuration in your default text editor. You can then edit this file and save your changes to update the configuration. For a list of available options, please take a look at the default configuration. Please make sure to not make any mistakes when you edit this file. Otherwise (parts of) the ICOtronic commands will not work, printing an error message about the (first) incorrect configuration value.

3.1.1 Adding the Path to Simplicity Commander on Linux

1. Open the user configuration file in your default text editor using the command line tool `icon`:

```
icon config
```

2. Add the path to Simplicity commander (e.g. `/opt/Simplicity Commander/commander/`) to the list `commands` → `path` → `linux`:

```
commands:
  path:
    linux:
      - /opt/Simplicity Commander/commander/
```

   Note: Keys (such as `commands`, `path` and `linux`) in the example above are case-insensitive in Dynaconf, e.g. it does not matter if you use `commands` or `COMMANDS` in the example above.

3. Store the modified configuration file

3.1.2 Setting up the Test Environment

1. Open the user configuration in your default text editor:

```
icon config
```

   If you never edited the configuration before, then the displayed text should be the same as in the file linked here.

2. Change the production date to the one of your PCB in ISO date format:

```
# Use the production date "February the 1st of the year 3456"
production date: &production_date 3456-02-01
```

3. Change the user name to the name of the person that runs the test:

```
# Use "Jane Doe" as name for the test operator
user name: &username Jane Doe
```

4. Change the holder type (only relevant for the test report):

```
# Specify the holder type (the holder that contains the PCB)
# as "D 10x130 HSK-A63"
holder type: &holder_type D 10x130 HSK-A63
```

5. Change the holder name (Bluetooth advertisement name) to the one of your sensor node. If your are not sure about the name you can use the `icon` command line tool to determine the name. The STH and SMH tests use this value to connect to the node.

```
# Connect to the sensor node with the name "untested"
holder name: &holder_name untested
```

6. Update the serial number of the sensor node. The STH and SMH tests change the sensor node Bluetooth advertisement name to this value, after the EEPROM (part of the) test was executed, **if the state value is set to `Epoxied`**.

```yaml
# Use the value "tested" as sensor node name,
# after the EEPROM test (succeeded)
holder serial number: &holder_serial tested
```

7. Change the state value to

   - `Bare PCB`, if the **sensor node test** (SMH/STH test) should flash **the sensor node** or to
   - `Epoxied` if the test should not flash the sensor node.

```yaml
# Do not flash the chip in the SMH/STH test
state: &state Epoxied
```

3.2 ICOn CLI Tool

3.2.1 Help

To show the available subcommands and options of `icon` you can use the option `-h` or `--help`:

```
icon -h
```

To show the available options for a certain subcommand add the subcommand before the option. For example, to show the help text for the subcommand `measure` you can use the following command:

```
icon measure -h
```

3.2.2 Listing Available Sensor Nodes

To print a list of all available sensor nodes, including their identifiers (name, MAC address, node number), please use the subcommand `list`:

```
icon list
```

3.2.3 Collecting Measurement Data

To collect and store measurement data from an STH you can uses the subcommand `measure`:

```
icon measure
```

By default the command will collect streaming data for 10 seconds for the first measurement channel and store the data in a file starting with the name `Measurement` and the extension `.hdf5` in the current working directory.

3.2.3.1 Specifying the Sensor Hardware

The `measure` subcommand requires that you specify one of the identifiers of a sensor node.

To connect to a sensor node by **name** use the option `-n` or `--name`. For example, the command below collects data from the sensor node with the name `Test-STH`:

```
icon measure -n 'Test-STH'
```

You can also use the MAC address to connect to a certain sensor node with the option `-m` or `--mac-address`:

```
icon measure -m '08-6B-D7-01-DE-81'
```

To connect using the node number use the option `-d` or `--node-number`:

```
icon measure -d 0
```

3.2.3.2 Changing the Run Time To change the run time of the measurement you can use the option `-t`, which takes the runtime in seconds as argument. The command

```
icon measure -t 300
```

for example, changes the runtime to 300 seconds (5 minutes).

3.2.3.3 Channel Selection To enable the measurement for the first ("x") channel and second ("y") measurement channel for

- an "older" STH (Firmware `2.x`, `BGM113` chip) or
- a "newer" STH (Firmware `3.x`, `BGM121` chip)

you can use the following command:

```
icon measure -1 1 -2 2 -3 0
```

Here:

- `0` indicates that you want to disable the specified measurement channel, while
- using the same number for the measurement channel (option) and the sensor/hardware channel (argument for the option) specifies that you want to use the specified channel.

Since the default value

- for the option `-1` is already `1`, and
- for the option `-3` is already `0`

you can also leave out these options to arrive at the shorter command:

```
icon measure -2 2
```

> **Note:** Due to a problem in the current firmware the amount of **paket loss is much higher**, if you
>
> - use the standard ADC configuration values, and
> - enable data transmission for **exactly 2 (channels)**.
>
> We strongly recommend you **use either one or three channels**.

For newer STH versions (Firmware `3.x`, `BGM121` chip) or SMHs (Sensory Milling Heads) you can also change the hardware/sensor channel for the first, second and third measurement channel. For example, to select

- hardware channel 8 for the first measurement channel
- hardware channel 1 for the second measurement channel, and
- hardware channel 3 for the third measurement channel

you can use the following command:

```
icon measure -1 8 -2 1 -3 3
```

If you just want to enable/set a measurement channel and use the hardware channel with the same number you can also just leave the argument for the specific measurement channel empty. For example, to use

- hardware channel 1 for measurement channel 1,
- hardware channel 2 for measurement channel 2, and
- hardware channel 3 for measurement channel 3

you can use the following command:

```
icon measure -1 -2 -3
```

or even shorter, since the default value for measurement channel 1 is hardware channel 1:

```
icon measure -2 -3
```

3.2.3.4 Changing the Reference Voltage

For certain sensor nodes you have to change the reference voltage to retrieve a proper measurement value. For example, STHs that use a ± 40 g acceleration sensor (ADXL356) require a reference voltage of 1.8 V instead of the usual supply voltage (`VDD`) of 3.3 V. To select the correct reference voltage for these nodes at startup use the option `-v 1.8`:

```
icon measure -v 1.8
```

3.2.3.5 Changing the Sampling Rate

You can change the sampling rate by modifying the parameters of the ADC (analog digital converter). There are 3 parameters which influence the sampling rate.

- **Prescaler**: Prescaler used by the ADC to get the sample points (`-s`, `--prescaler`)
- **Acquisition Time**: Time the ADC holds a value to get a sampling point (`-a`, `--acquisition`)
- **Oversampling Rate**: Oversampling rate of the ADC (`-o`, `--oversampling`)

The formula which can be used to calculate the sampling rate can be found in the general ICOtronic system documentation under the section “Sampling Rate”. Please be aware that the actual sample rate might be slightly lower, even if there is no data loss.

For example, to use a sampling rate of about 2381 Hz you can use the following command:

```
icon measure --prescaler 2 --acquisition 8 --oversampling 256
```

3.2.4 Renaming a Sensor Node

To change the name of a sensor you can use the subcommand `rename`. For example, to change the name of the sensor node with the Bluetooth MAC address `08-6B-D7-01-DE-81` to `Test-STH` use the following command:

```
icon rename -m 08-6B-D7-01-DE-81 Test-STH
```

For more information about the command you can use the option `-h`/`--help`:

```
icon rename -h
```

3.2.5 Opening the User Configuration

To open the user configuration file, you can use the subcommand `config`:

```
icon config
```

If the file does not exist yet, then it will be created and filled with the content of the default user configuration. For more information on how to change the configuration, please take a look at the section "Changing Configuration Values".

3.2.6 STU Commands

To list all available STU subcommands, please use the option `-h` (or `--help`):

```
icon stu -h
```

3.2.6.1 Enable STU OTA Mode To enable the Bluetooth advertising of the STU and hence the "over the air" firmware update, please run the following command:

```
icon stu ota
```

3.2.6.2 Retrieve the Bluetooth STU MAC Address To retrieve the STU Bluetooth address you can use the following command:

```
icon stu mac
```

3.2.6.3 Reset STU To reset the STU please use the following command:

```
icon stu reset
```

3.2.6.4 Determining Data Loss Depending on

- the hardware of the computer and
- the used sampling frequency

the ICOtronic library might not be able to keep up with the rate of measurement data that is collected by the STU and stored in the buffer of the CAN adapter. The result in this case will be a certain rate of data loss, since the CAN adapter will get rid of old data if it is not collected fast enough.

To minimize the chance of this kind of data loss you can use the command

```
icon dataloss
```

to determine the CPU usage and data loss for the current computer at certain sample rates.

3.3 Production Tests

This tutorial lists the usual steps to test a sensory holder assembly or a sensory tool holder.

3.3.1 General

To run the production tests for one of the ICOtronic nodes, please execute one of the following commands:

Command	Node
`test-stu`	Stationary Transceiver Unit (STU)
`test-sth`	Sensory Holder Assembly (SHA), Sensory Tool Holder (STH)
`test-smh`	Sensory Milling Head (SMH)

For a list of available command line options, please use the option `-h` after one of the commands e.g.:

```
test-sth -h
```

3.3.1.1 Specific Tests

To only run a single test you need the specify its name. For example, to run the test `test__firmware_flash` of the STU you can use the following command:

```
test-stu TestSTU.test__firmware_flash
```

You can also run specific tests using pattern matching. To do that use the command line option `-k`. For example, to run the firmware flash and the connection test of the STH test you can use the command:

```
test-sth -k flash -k connection
```

which executes all tests that contain the text `flash` or `connection`.

3.3.2 STH

The text below gives you a more detailed step-by-step guide on how to run the tests of the STH.

1. **Note:** You can **skip this step, if you do not want to run the flash test**. To skip the flash test, please set `sth` → `status` in the configuration to `Epoxied`.

   Please create a directory called `Firmware` in the current user's `Documents` directory (`~/Documents`).

   Note: To open the user's home directory on Windows you can use the following command in (Windows) Terminal:

```
ii ~/Documents
```

Then put the current version of the STH firmware into this directory. Afterwards the directory and file structure should look like this:

```
~
  Documents
          Firmware
                  manufacturingImageSthv2.1.10.hex
```

As alternative to the steps above you can also change the variable `sth` → `firmware` → `location` → `flash` in the configuration to point to the firmware that should be used for the flash test.

2. Make sure that the configuration values are set correctly. You probably need to change at least the Bluetooth advertisement name (`sth` → `name`) to the name of the STH you want to test.

3. Please open your favorite Terminal application and execute the STH test using the command `test-sth`. For more information about this command, please take a look at the section "General" above.

   Please note, that the test will rename the tested STH

   - to a **Base64 encoded version of the Bluetooth MAC address**, if `sth` → `status` is set to `Bare PCB`, or
   - to the **serial number** (`sth` → `serial number`), if you set the status to `Epoxied`.

3.3.3 SMH

The preparation steps for the SMH test are very similar to the ones of the STH test.

1. Please make sure that the config value that stores the SMH firmware filepath (`smh` → `firmware` → `location` → `flash`) points to the correct firmware. If you have not downloaded a firmware image for the SMH you can do so here.

2. Check that the configuration values such as the SMH name (`smh` → `name`) are set correctly.

3. Please execute the test using the following command:

   ```
   test-smh
   ```

3.3.4 STU

The following description shows you how to run the STU tests.

1. **Note:** You can **skip this step, if you do not want to run the flash test**.

   Please take a look at step 1 of the description for the STH test and replace every occurrence of STH (or `sth`) with STU (or `stu`).

   > **Note:** The STU test always uploads the flash file to the board, i.e. the setting `stu` → `status` is **not** read/used by the STU tests.

   In the end of this step the directory structure should look like this:

```
~
  Documents
        Firmware
              manufacturingImageStuv2.1.10.hex
```

You can find the current version of the STU firmware here.

2. Please take a look at the section "General" to find out how to execute the production tests for the STU. If you want to run the connection and EEPROM test (aka **all tests except the flash test**, i.e. the EEPROM and connection test), then please execute the following command:

```
test-stu -k eeprom -k connection
```

3.3.5 Firmware Versions

The (non-exhaustive) table below shows the compatible firmware for a certain node. The production tests assume that you use **firmware that includes the bootloader**.

Node	Hardware Version	Microcontroller	Firmware
STH	1.3	BGM113	• Version 2.1.10
STH	2.2	BGM123	• Aladdin
SMH	2.1	BGM121	• Version 3.0.0 • Version E3016 Beta

4 Code Examples

For information on how to write your own code including sample code please take a look at the API documentation.

5 Virtualization

You can also use the ICOtronic package with various virtualization software. For that to work you have to make sure that (at least) the PEAK CAN adapter is attached to the virtual guest operating system. For some virtualization software you might have to install additional software for that to work. For example, VirtualBox requires that you install the VirtualBox Extension Pack before you can use USB 2 and USB 3 devices.

> **Note:** Please be advised that the **VirtualBox Extension Pack is paid software** even though you can download and use it without any license key. **Oracle might come after you, if you do not pay for the license**, even if you use the Extension Pack in an educational setting.

The table below shows some of the virtualization software we tried and that worked (when we tested it).

Virtualization Software	Host OS	Host Architecture	Guest OS	Guest Architecture	Notes
Parallels Desktop	macOS	x64	Ubuntu 20.04	x64	
Parallels Desktop	macOS	x64	Windows 10	x64	
Parallels Desktop	macOS	ARM64	Fedora 36	ARM64	
Parallels Desktop	macOS	ARM64	Windows 11	ARM64, x64	JLink (and hence Simplicity Commander) only works with programming adapters that support WinUSB
VirtualBox	macOS	x64	Windows 10	x64	
VirtualBox	Windows	x64	Fedora 36	x64	
WSL 2	Windows	x64	Ubuntu 22.04	x64	

5.1 Windows Subsystem for Linux 2

Using ICOtronic in the WSL 2 currently requires using a custom Linux kernel. We **would not recommend** using ICOtronic with this type of virtualization software, since the setup requires quite some amount of work and time. Nevertheless the steps below should show you how you can use the PEAK CAN adapter and hence the ICOtronic package with WSL 2.

1. Install WSL 2 (Windows Shell):

   ```
   wsl --install
   ```

2. Install Ubuntu 22.04 VM (Windows Shell):

   1. Install Ubuntu 22.04 from the Microsoft Store
   2. Open the Ubuntu 22.04 application
      1. Choose a user name
      2. Choose a password
   3. Execute the following commands in a Powershell session

      ```
      wsl --setdefault Ubuntu-22.04
      wsl --set-version Ubuntu-22.04 2
      ```

      The second command might fail, if `Ubuntu-22.04` already uses WSL 2. In this case please just ignore the error message.

3. Create Custom Kernel

   Windows Shell:

   Note: Please replace `<user>` with your (Linux) username (e.g. `rene`)

   ```
   cd ~/Documents
   mkdir WSL
   cd WSL
   wsl --export Ubuntu-22.04 CANbuntu.tar
   wsl --import CANbuntu CANbuntu CANbuntu.tar
   wsl --distribution CANbuntu --user <user>
   ```

Linux Shell:

```
sudo apt update
sudo apt upgrade -y
sudo apt install -y bc build-essential flex bison libssl-dev libelf-dev \
                    libncurses-dev autoconf libudev-dev libtool dwarves
cd ~
git clone https://github.com/microsoft/WSL2-Linux-Kernel.git
cd WSL2-Linux-Kernel
uname -r # 5.15.74.2-microsoft-standard-WSL2 → branch …5.15.y
git checkout linux-msft-wsl-5.15.y
cat /proc/config.gz | gunzip > .config
make menuconfig
```

Make sure the following features are enabled:

- Device Drivers → USB Support
- Device Drivers → USB Support → USB announce new devices
- Device Drivers → USB Support → USB Modem (CDC ACM) support
- Device Drivers → USB Support → USB/IP
- Device Drivers → USB Support → USB/IP → VHCI HCD
- Device Drivers → USB Support → USB Serial Converter Support
- Device Drivers → USB Support → USB Serial Converter Support → USB FTDI Single port Serial Driver

Enable the following features:

- Networking support → CAN bus subsystem support
- Networking support → CAN bus subsystem support → Raw CAN Protocol
- Networking support → CAN bus subsystem support → CAN device drivers → Virtual Local CAN Interface
- Networking support → CAN bus subsystem support → CAN device drivers → Serial / USB serial CAN Adaptors (slcan)
- Networking support → CAN bus subsystem support → CAN device drivers → CAN USB Interfaces → PEAK PCAN-USB/USB Pro interfaces for CAN 2.0b/CAN-FD

Save the modified kernel configuration.

Linux Shell:

```
touch .scmversion
make
sudo make modules_install
sudo make install
```

4. Install `usbipd-win` (Linux Shell):

```
cd tools/usb/usbip
./autogen.sh
./configure
sudo make install
sudo cp libsrc/.libs/libusbip.so.0 /lib/libusbip.so.0
sudo apt-get install -y hwdata
```

5. Copy image (Linux Shell):

Note: Please replace `<user>` with your (Windows) username (e.g. `rene`)

```
cd ~/WSL2-Linux-Kernel
cp arch/x86/boot/bzImage /mnt/c/Users/<user>/Documents/WSL/canbuntu-bzImage
```

6. Create `.wslconfig` in (root of) Windows user directory and store the following text:

```
[wsl2]
kernel=c:\\users\\<user>\\Documents\\WSL\\canbuntu-bzImage
```

Note: Please replace `<user>` with your (Windows) username (e.g. `rene`)

7. Set default distribution (Windows Shell)

```
wsl --setdefault CANbuntu
```

8. Shutdown and restart WSL (Windows Shell):

```
wsl --shutdown
wsl -d CANbuntu --cd "~"
```

9. Change default user of WSL distro (Linux Shell)

   ```
   sudo nano /etc/wsl.conf
   ```

   Insert the following text:

   ```
   [user]
   default=<user>
   ```

   Note: Please replace `<user>` with your (Windows) username (e.g. `rene`)

   Save the file and exit `nano`:

   1. Ctrl + O
   2.
   3. Ctrl + X)

10. Restart WSL: See step 8

11. Install `usbipd` (Windows Shell):

```
winget install usbipd
```

12. Attach CAN-Adapter to Linux VM (Windows Shell)

```
usbipd wsl list
# …
# 5-3    0c72:0012  PCAN-USB FD                       Not attached
# …
usbipd wsl attach -d CANbuntu --busid 5-3
usbipd wsl list
# …
# 5-3    0c72:0012  PCAN-USB FD                       Attached - CANbuntu
# …
```

13. Check for PEAK CAN adapter in Linux (Optional, Linux Shell):

```
dmesg | grep peak_usb
# …
# peak_usb 1-1:1.0: PEAK-System PCAN-USB FD v1 fw v3.2.0 (1 channels)
# …

lsusb
# …
# Bus 001 Device 002: ID 0c72:0012 PEAK System PCAN-USB FD
# …
```

14. Add virtual link for CAN device (Linux Shell)

```
sudo ip link set can0 type can bitrate 1000000
sudo ip link set can0 up
```

    Note: If the commands above fail with the error message:

    RTNETLINK answers: Connection timed out

    then please disconnect and connect the USB CAN adapter. After that attach it to the Linux VM again (see step 12).

15. Install `pip` (Linux Shell):

```
sudo apt install -y python3-pip
```

16. Install ICOtronic (Linux Shell)

```
cd ~
mkdir Documents
cd Documents
git clone https://github.com/MyTooliT/ICOtronic.git
cd ICOtronic
python3 -m pip install --prefix=$(python3 -m site --user-base) -e .
```

17. Run a script to test that everything works as expected (Linux Shell)

```
icon list
```

    If the command above fails with the message

    ```
    Command 'icon' not found…
    ```

    then you might have to logout and login into the WSL session again before you execute `icon list` again.

    Notes:

    - You only need to repeat steps
    - 12: attach the CAN adapter to the VM in Windows and
    - 14: create the link for the CAN device in Linux

    after you set up everything properly once.

    - Unfortunately configuring the CAN interface automatically does not seem to work (reliably) on WSL yet

5.1.1 Installing/Using Simplicity Commander

1. Download and unpack Simplicity Commander (Linux Shell)

```
sudo apt install -y unzip
mkdir -p ~/Downloads
cd ~/Downloads
wget https://www.silabs.com/documents/public/software/SimplicityCommander-Linux.zip
unzip SimplicityCommander-Linux.zip
cd SimplicityCommander-Linux
tar xf Commander-cli_linux_x86_64*.tar.bz
mkdir -p ~/Applications
mv commander-cli ~/Applications/commander
ln -s ~/Applications/commander/commander-cli ~/Applications/commander/commander
# Fix J-Link connection
# https://wiki.segger.com/J-Link_cannot_connect_to_the_CPU
sudo cp ~/Applications/commander/99-jlink.rules /etc/udev/rules.d/
```

2. Add `commander` binary to path (Linux Shell)

    1. Open `~/.profile` in `nano`:

    ```
    nano ~/.profile
    ```

    2. Add the following text at the bottom:

    ```
    if [ -d "$HOME/Applications/commander" ] ; then
        PATH="$HOME/Applications/commander:$PATH"
    fi
    ```

    3. Save your changes

3. Restart WSL

4. Connect programming adapter to Linux (Windows Shell)

```
usbipd wsl list
# …
# 5-4    1366:0105  JLink CDC UART Port (COM3), J-Link driver Not attached
# …
usbipd wsl attach --busid 5-4
```

5. Check if `commander` JLink connection works without using `sudo` (Linux Shell)

```
commander adapter dbgmode OUT
# Setting debug mode to OUT...
# DONE
```

    Note:
    If the command above fails with the error message:

    `error while loading shared libraries: libGL.so.1: cannot open shared object file…`

    then you need to install `libgl1`:

```
sudo apt install -y libgl1
```

5.1.2 Run Tests in WSL

Note: The tests use the command-line `xdg-open`. For the tests also work on Linux, then you need to install the Python package `xdg-open-wsl`:

```
pip3 install --user git+https://github.com/cpbotha/xdg-open-wsl.git
```

1. Start WSL (Windows Shell)

```
wsl -d CANbuntu --cd "~/Documents/ICOtronic"
```

2. Connect programming/CAN adapter to Linux (Windows Shell):

```
$jlink_id = usbipd wsl list | Select-String JLink | %{$_ -replace '(\d-\d).*','$1'}
$can_id = usbipd wsl list | Select-String PCAN-USB | %{$_ -replace '(\d-\d).*','$1'}
usbipd wsl attach -d CANbuntu --busid $jlink_id
usbipd wsl attach -d CANbuntu --busid $can_id
```

3. Configure CAN interface (Linux Shell):

```
sudo ip link set can0 type can bitrate 1000000
sudo ip link set can0 up
```

4. Run tests (Linux Shell):

```
make run
```

6 Containerization

6.1 Docker on Linux

The text below shows how you can use (code of) the ICOtronic package in a Docker container on a **Linux host**. The description on how to move the interface of the Docker container is an adaption of an article/video from the "Chemnitzer Linux-Tage".

6.1.1 Creating a Docker Image

To create a Docker image that contains ICOtronic just install the package with `pip` inside your `Dockerfile`. We recommend that you use a virtual environment to install the package. For an example, please take a look at the `Dockerfile` in the folder Docker.

6.1.2 Building the Docker Image

If you do not want to create a `Dockerfile` yourself, you can build an image based on our Docker example file:

```
docker build -t mytoolit/icotronic -f Docker/Dockerfile .
```

6.1.3 Using ICOtronic in the Docker Container

1. Run the container **(Terminal 1)**

    1. Open a new terminal window
    2. Open a shell in the Docker container

    ```
    docker run --rm -it --name icotronic mytoolit/icotronic
    ```

2. Move the CAN interface into the network space of the Docker container **(Terminal 2)**

    ```
    export DOCKERPID="$(docker inspect -f '{{ .State.Pid }}' icotronic)"
    sudo ip link set can0 netns "$DOCKERPID"
    sudo nsenter -t "$DOCKERPID" -n ip link set can0 type can bitrate 1000000
    sudo nsenter -t "$DOCKERPID" -n ip link set can0 up
    ```

    Note: Alternatively you can also add the option `--network host` to the Docker command from step 1. Please check out the Docker documentation to learn more about the consequences of using this option.

3. Run a test command in Docker container **(Terminal 1)** e.g.:

    ```
    icon list
    ```

7 Scripts

After you installed the ICOtronic package various helper scripts are available:

- `convert-base64-mac`: Utility to convert a Base64 encoded 8 character text into a Bluetooth MAC address
- `convert-mac-base64`: Convert Bluetooth MAC address into a (Base64) encoded 8 character string
- `check-eeprom`: Write a byte value into the cells of an EEPROM page an check how many of the values are read incorrectly after an reset
- `icon`: Controller software for the ICOtronic system
- `test-smh`: Test code for the SMH
- `test-sth`: Test code for the STH/SHA
- `test-stu`: Test code for the STU

7.1 EEPROM Check

The script `check-eeprom` connects to an STH using its MAC address. Afterwards it writes a given byte value (default: `10`) into all the cells of an EEPROM page. It then resets the STH, connects again and shows the amount of incorrect EEPROM bytes. It repeats the last steps 5 times before it prints all values in the EEPROM page.

The command below shows how to execute the EEPROM check for the STH with MAC address `08:6b:d7:01:de:81`:

```
check-eeprom 08:6b:d7:01:de:81
```

You can specify the value that should be written into the EEPROM cells using the option `--value`:

```
check-eeprom 08:6b:d7:01:de:81 --value 42
```

7.2 ICOn

The command ICOn provides various subcommands to work with the ICOtronic system. For more information, please take a look at the tutorial section about the tool.

7.3 MAC Address Conversion

The utility `convert-mac-base64` returns the Base64 encoded version of a MAC address. We use the encoded addresses as unique Bluetooth advertisement name for the STH (or SHA). Unfortunately we can not use the MAC address directly because the maximum length of the name is limited to 8 characters. To decode the Base64 name back into a Bluetooth address you can use the script `convert-base64-mac`.

7.3.1 Examples

```
# Convert a MAC address into an 8 character name
convert-mac-base64 08:6b:d7:01:de:81
#> CGuXAd6B

# Convert the Base64 encoded name back into a MAC address
convert-base64-mac CGvXAd6B
#> 08:6b:d7:01:de:81
```

7.4 Test-STH

The command `test-sth` executes the tests for the STH (`sth.py`). All command line arguments of the wrapper will be forwarded to `sth.py`.

7.5 Test-STU

The command `test-stu` is a wrapper that executes the tests for the STU (`stu.py`). All command line arguments of the wrapper will be forwarded to `stu.py`.

7.6 Test-SMH

The command `test-smh` is a wrapper that executes the tests for the SMH (`smh.py`). All command line arguments of the wrapper will be forwarded to `smh.py`.

8 Development

8.1 Install

You can use the instructions below, if you want to work on the code of the ICOtronic package, i.e. add additional features or fix bugs.

1. Clone the repository to a directory of your choice. You can either use the command line tool `git`:

   ```
   git clone https://github.com/MyTooliT/ICOtronic.git
   ```

   or one of the many available graphical user interfaces for Git to do that.

2. Install ICOtronic with Poetry

   1. Change your working directory to the (root) directory of the cloned repository
   2. Install ICOtronic:

      ```
      poetry lock && poetry install --all-extras
      ```

      Notes:

      - The command above will install the package in a virtual environment.
      - You need to prefix commands, such as `icon` with the command `poetry run` (e.g. `poetry run icon`) to execute it in this virtual environment.
      - Using `poetry run` will only work in the root folder of the repository (that contains `pyproject.toml`).

3. Install other required tools (for tests)

   - `hdf5`: For the command line tool `h5dump` (Linux/macOS). You can install hdf5 via Homebrew:

     ```
     brew install hdf5
     ```

8.2 Style

Please use the guidelines from PEP 8. For code formatting we currently use Black, which should format code according to PEP 8 by default.

To format the whole code base you can use the following command in the root of the repository:

```
poetry black .
```

For development we recommend that you use a tool or plugin that reformats your code with Black every time you save. This way we can ensure that we use a consistent style for the whole code base.

8.3 Tests

The following text describes some of the measures we should take to keep the software stable.

Please only push your changes to the `main` branch, if you think there are no new bugs or regressions. The `main` branch **should always contain a working version of the software**. Please **always run**

- the **automatic test** (`make run`) for **every supported OS** (Linux, macOS, Windows) and
- the **manual tests** on Windows

before you push to the `main` branch.

8.3.1 Code Checks

8.3.1.1 Flake8 We check the code with Flake8. Please use the following command in the root of the repository to make sure you did not add any code that introduces warnings:

```
poetry run flake8
```

8.3.1.2 mypy To check the type hint in the code base we use the static code checker mypy. Please use the following command in the root of the repository to check the code base for type problems:

```
poetry run mypy icotronic
```

8.3.1.3 Pylint We currently use Pylint to check the code:

```
poetry run pylint .
```

8.3.2 Automatic Tests

8.3.2.0.1 Usage Please run the following command in the root of the repository:

```
poetry run pytest -v
```

and make sure that it reports no test failures.

8.3.3 Manual Tests

8.3.3.1 STH Test

1. Call the command `poetry run test-sth` for a working STH
2. Wait for the command execution
3. Check that the command shows no error messages
4. Open the PDF report (`STH Test.pdf`) in the repository root and make sure that it includes the correct test data

8.3.3.2 STU Test

1. Call the command `poetry run test-stu` (or `poetry run test-stu -k eeprom -k connection` when you want to skip the flash test) for a working STU
2. Wait for the command execution
3. Check that the command shows no error messages
4. Open the PDF report (`STU Test.pdf`) in the repository root and make sure that it includes the correct test data

8.3.3.2.1 Extended Tests The text below specifies extended manual test that should be executed before we release a new version of the ICOtronic package. Please note that the tests assume that you more or less use the default configuration values.

Check the Performance of the Library

1. Open your favorite terminal application and change your working directory to the root of the repository
2. Remove HDF5 files from the repository:

   ```
   rm *.hdf5
   ```

   > **Note:** You can ignore errors about "no matches for wildcard" on Linux and macOS. This message just tells you that there is no file with the extension `hdf5` in the current directory.

3. Check that no HDF5 files exist in the repository. The following command should not produce any output:

   ```
   ls *.hdf5
   ```

4. Give your test STH the name "Test-STH"
5. Run the following command

   ```
   poetry run icon measure -t 300 -n Test-STH
   ```

   - The command should not print any **no error messages**.
   - The **data loss must be below 1 %**.

6. Check that the repo now contains a HDF5 (`*.hdf5`) file

   ```
   ls *.hdf5
   ```

7. Open the file in HDFView
8. Check that the timestamp of the last value in the `acceleration` table has **approximately the value** `30 000 000` (all values above `29 900 000` should be fine).

8.3.4 Combined Checks & Tests

While you need to execute some test for the ICOtronic package manually, other tests and checks can be automated.

> **Note:** For the text below we assume that you installed `make` on your machine.

To run all checks, the STH test and the STU test use the following `make` command:

```
make run
```

Afterwards make sure there were no (unexpected) errors in the output of the STH and STU test.

8.4 Release

1. Check that the **CI jobs** for the `main` branch finished successfully
2. Check that the most recent "Read the Docs" build of the documentation ran successfully
3. Check that the **checks and tests** run without any problems on **Linux**, **macOS** and **Windows**
   1. Set the value of `sth` → `status` in the configuration to `Epoxied`
   2. Execute the command:

```
make run
```

in the root of the repository

4. Check that the **firmware flash** works in Windows

   - Execute `poetry run test-sth -v`

     1. once with `sth` → `status` set to `Epoxied`, and
     2. once set to `Bare PCB`

     in the configuration. To make sure, that the STU flash test also works, please use both STU test commands described in the section "STU Test".

     If you follow the steps above you make sure that the **flash tests work** for both STU and STH, and there are **no unintentional consequences of (not) flashing the chip** before you run the other parts of the test suite.

5. Execute the **extended manual tests** in Windows and check that everything works as expected.

6. Update the release notes:

   1. Open the release notes for the latest version
   2. Replace links with a permanent version:

      For example instead of

      - `../../something.txt` use
      - `https://github.com/MyTooliT/ICOtronic/blob/REVISION/something.txt`,

      where `REVISION` is the latest version of the main branch (e.g. `8568893f` for version `1.0.5`)
   3. Commit your changes

7. Change the version number and commit your changes (please replace `<VERSION>` with the version number e.g. `1.0.5`):

```
poetry version <VERSION>
export icotronic_version="$(poetry version -s)"
git commit -a -m "Release: Release version $icotronic_version"
git tag "$icotronic_version"
git push && git push --tags
```

   Note: GitHub Actions will publish a package based on the tagged commit and upload it to PyPi.

8. Create a new release here

   1. Insert the version number (e.g. `1.0.5`) into the tag field
   2. For the release title use "Version ", where `<VERSION>` specifies the version number (e.g. "Version 1.0.5")
   3. Paste the release notes for the lastest release into the main text field
   4. Click on "Publish Release"

   Note: Alternatively you can also use the `gh` command:

```
gh release create
```

   to create the release notes.

9. Close the milestone for the latest release number