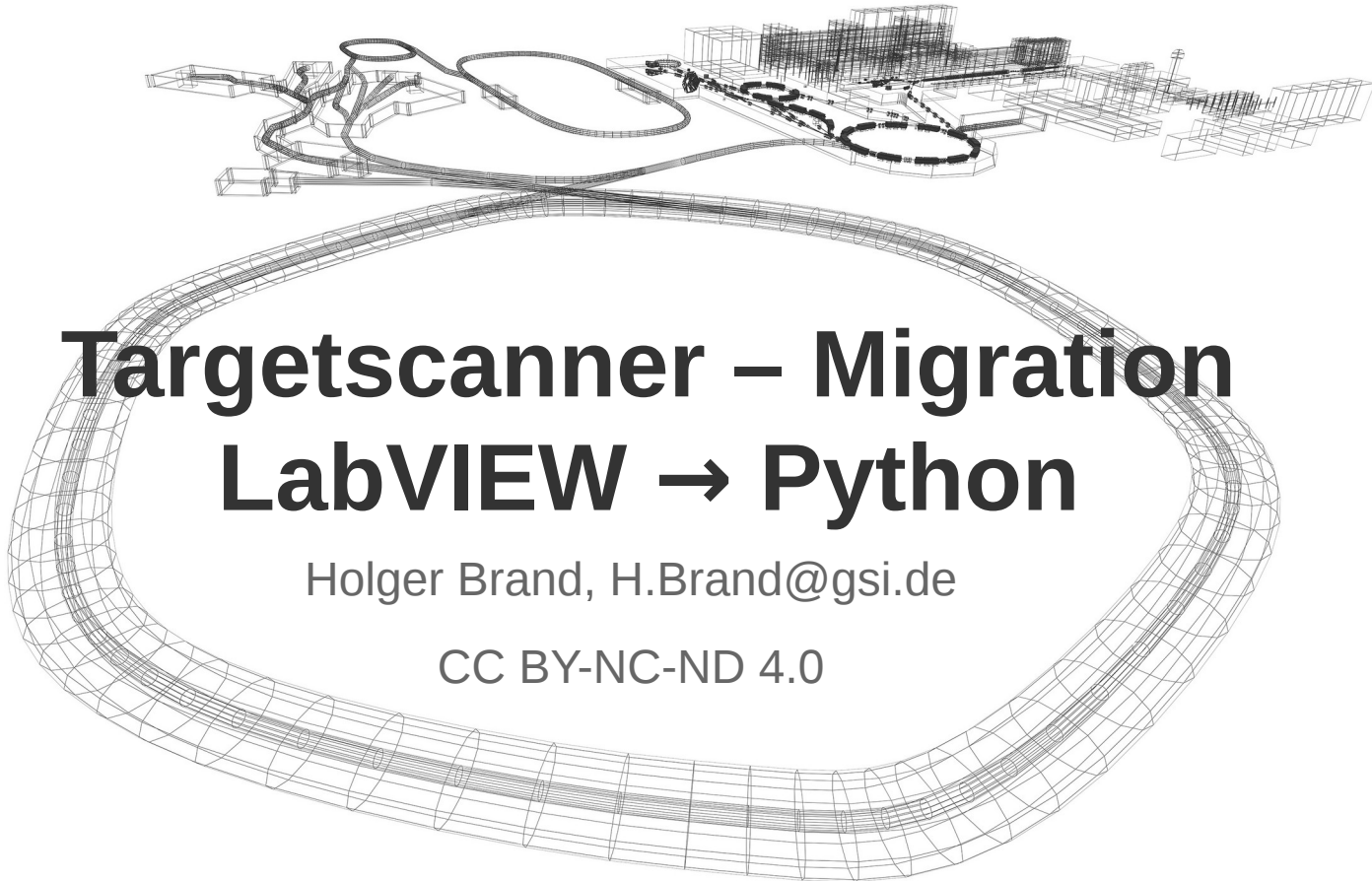


A detailed wireframe model of a particle accelerator, showing a large circular ring and various smaller components and structures. The model is rendered in a light gray, transparent style, allowing the text to be clearly visible in the center.

Targetscanner – Migration LabVIEW → Python

Holger Brand, H.Brand@gsi.de

CC BY-NC-ND 4.0

■ Targetscanner Project

- Start: 1999 with LabVIEW 5.0
- Publication: 2001 LabVIEW 6i
 - „Automated Thickness Measurement of Targets and Degraders“
Virtuelle Instrumente in der Praxis -
Begleitband zum Kongress VIP 2001
Rahman Jamal /Hans Jaschinski (Hrdg.)
Heidelberg Hüthig, 2001
Praxiswissen Elektronik-Industrie: Meßtechnik
ISBN 3-7785-2829-7
- Last update: 2021 with LabVIEW 2021 SP1
- Repository: <https://git.gsi.de/EE-LV/EE-HB/TargetScanner>
 - Tag: Release-1-1-4

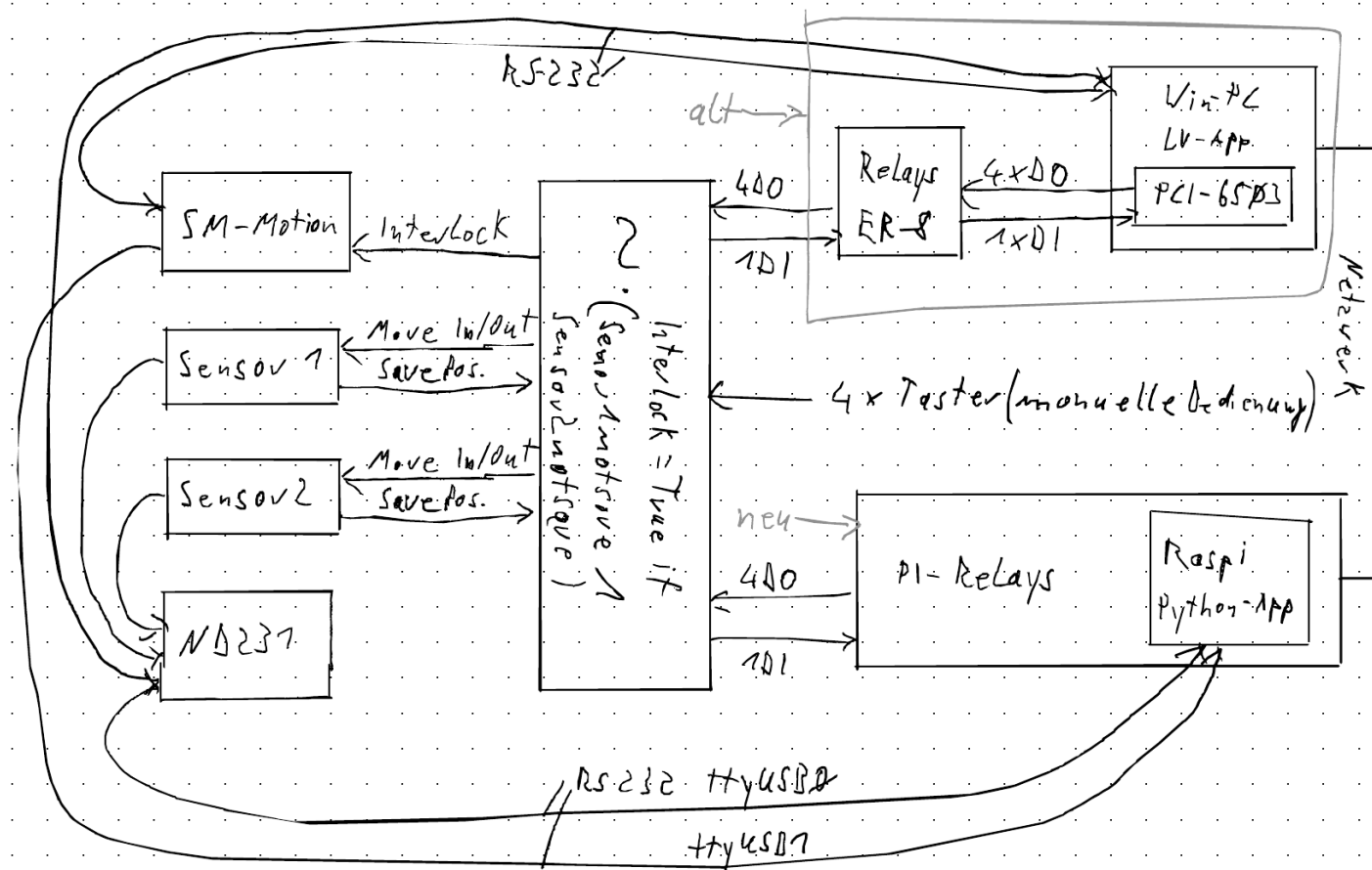
■ Migration to LabVIEW NXG 5.1

- Deprecated before migration was finished.



- Show separate VIP 2001 talk.

Targetscanner Sketch



PyAcdaq: Python based Automation Control and Data Acquisition

(inspired by CS/CS++)

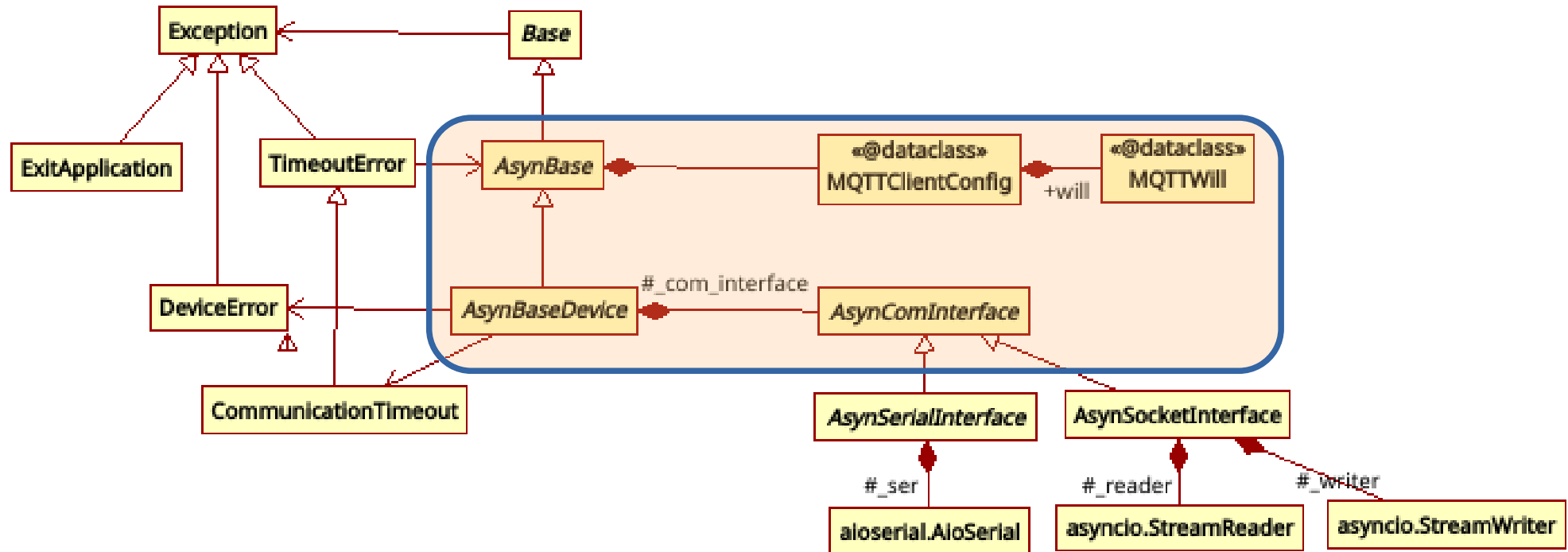


- **Actor based**, https://git.gsi.de/EKS/Python/ACDAQ/PyAcdaq/-/tree/master/src/pyacdaq/actor?ref_type=heads
 - Actor based approach demonstrated with ESR/SIS-18 Bakeout System
<https://git.gsi.de/EKS/Python/ACDAQ/VacuumHeatingSystem>
- **Asyncio based***, https://git.gsi.de/EKS/Python/ACDAQ/PyAcdaq/-/tree/develop/src/pyacdaq/asyn?ref_type=heads
 - Based on asyncio avoiding problems with multi-threading or multi-processing
 - One process – one thread - one single memory space
 - Simple synchronization methods available
 - Barriers, Conditions, Events, Locks, Queues, Semaphores
 - Separation and loose coupling of Operation-, Control- and Device-Layer by MQTT
 - Scaling by distribution → Processes / Computer Nodes
- **Extensible** with other Open Source Tools → Database, InfluxDB, Grafana as necessary
- More features could be added, if adopted in community.

* With experience during migration of some Artemis LabVIEW programs: <https://git.gsi.de/EKS/Python/Experiments/Artemis>

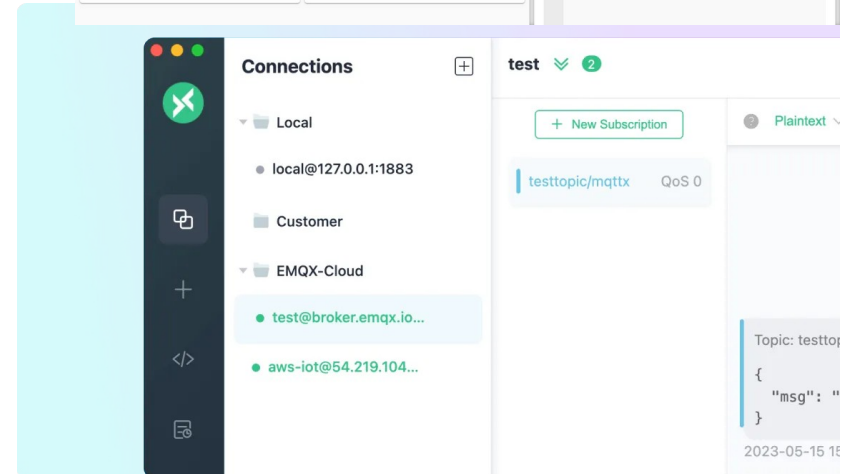
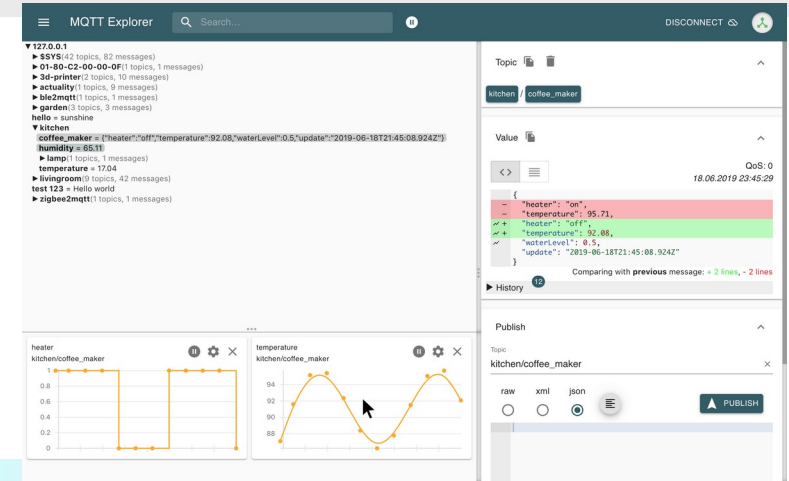
- Simplified UML Class Diagram

<https://apps.kde.org/de/umbrello/>



Python Packages / Tools

- Asyncio built into Python
<https://docs.python.org/3.12/library/asyncio.html>
- Aioserial: Serial communication
<https://pypi.org/project/aioserial/>
- MQTT v5 Network Protocol (publisher/subscriber)
<https://mosquitto.org/>
 - aiomqtt: <https://pypi.org/project/aiomqtt/> based on:
 - paho-mqtt: <https://pypi.org/project/paho-mqtt/>
 - Generic GUIs
 - MQTT-Explorer: <https://mqtt-explorer.com/>
 - MQTTeX: <https://mqttx.app/>
- Pint: <https://pypi.org/project/Pint/>
 - Physical units
 - <https://pint.readthedocs.io/en/stable/>
- Tkinter GUI Widgets
<https://docs.python.org/3.12/library/tkinter.html>



- asyncio is a library to write concurrent code using the `async/await` syntax.
- asyncio is often a perfect fit for IO-bound and high-level structured network code.
 - asyncio is used as a foundation for multiple Python asynchronous frameworks that provide high-performance network and web-servers, database connection libraries, distributed task queues, etc.
- asyncio provides a set of high-level APIs to:
 - `run Python coroutines` concurrently and have full control over their execution;
 - perform `network IO and IPC`;
 - control `subprocesses`;
 - distribute tasks via `queues`;
 - `synchronize` concurrent code;
- Additionally, there are low-level APIs for library and framework developers to:
 - create and manage `event loops`, which provide asynchronous APIs for `networking`, running `subprocesses`, handling `OS signals`, etc;
 - implement efficient protocols using `transports`;
 - `bridge` callback-based libraries and code with `async/await` syntax.

■ Eclipse Mosquitto

- Is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers.
- The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers.
- The Mosquitto project also provides a C library for implementing MQTT clients, and the very popular `mosquitto_pub` and `mosquitto_sub` command line MQTT clients.
- Mosquitto is part of the [Eclipse Foundation](#), and is an [iot.eclipse.org](#) project, with its development driven by [Cedalo](#).

■ paho-mqtt

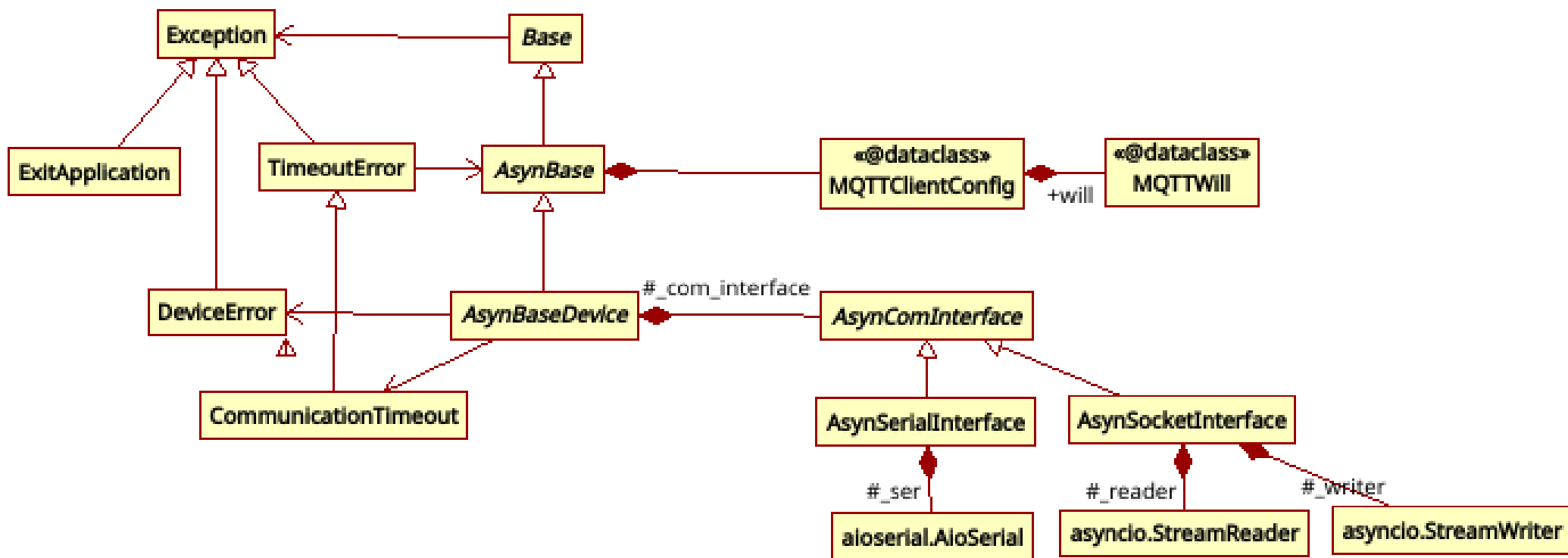
- [Eclipse Paho](#) MQTT Python client library, which implements versions 5.0, 3.1.1, and 3.1 of the MQTT protocol.

■ aiomqtt

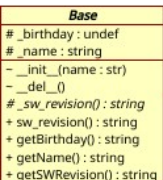
- No more callbacks! 👍
- No more return codes (welcome to the `MqttError`)
- Graceful disconnection (forget about `on_unsubscribe`, `on_disconnect`, etc.)
- Supports MQTT versions 5.0, 3.1.1 and 3.1
- Fully type-hinted

```
async with Client("test.mosquitto.org") as client:  
    await client.publish("temperature/outside", payload=28.4)
```

```
async with Client("test.mosquitto.org") as client:  
    await client.subscribe("temperature/#")  
    async for message in client.messages:  
        print(message.payload)
```



AsynBase & AsynBaseDevice



```
AsynBase

#_mqtt_client: MQTTHandler = None
#_command_queue: asyncio.Queue
#_command_task: asyncio.Task = None
#_periodic_active: bool = False
#_periodic_counter: int = -1
#_periodic_interval: pint.Quantity = -1. * ureg.s
#_periodic_wakeup: asyncio.Event = asyncio.Event()
#_periodic_task: asyncio.Task = None
#_subscriptions: set[str] = set()
#_subscribe_task: asyncio.Task = None
#_periodic_time: float = 0.
#_top_level_topic: str
#_mqtt_client_connected: bool = False
#_mqtt_config: MQTTClientConfig

-__init__(name: str, mqtt_config: MQTTClientConfig, top_level_topic: str)
-__del__()
-__mqtt_connect(): bool «async»
-__mqtt_disconnect(): bool «async»
-__command_loop() «async»
-__subscriber_loop() «async»
-__periodic_loop() «async»
-__publish_all_topics() «async»
#_command_help(): str
#_handle_command(q_item: list[Any]) «async»
#_handle_subscriptions(topic_levels: list[str], payload: str, message: aiomqtt.message.Message) «async»
#_periodic_action() «async»
#_periodic_start() «async»
#_periodic_stop() «async»
#_publish_topics() «async»
#_start() «async»
#_stop() «async»
#_subscribe_topics(): set[str] «async»
#_sw_revision(): str
+command_help(): str
+start(subscribe_task: bool = False, periodic_interval: pint.Quantity = -1. * ureg.s) «async»
+start_periodic(periodic_interval: pint.Quantity = -1. * ureg.s) «async»
+stop() «async»
+stop_periodic() «async»
+get_command_queue(): asyncio.Queue
+get_periodic_active(): bool
+get_periodic_counter(): int
+get_periodic_interval(): pint.Quantity
+get_periodic_time(): float
+set_periodic_interval(periodic_interval: pint.Quantity) «async»
```

```
AsynBaseDevice

#_resource: str
#_id_query_flag: bool
#_reset_flag: bool
#_reset_options: str
#_selftest_flag: bool
#_firmware_revision: str = NA
#_hardware_revision: str = NA
#_idn: str
#_initialized: bool = False
#_reset_result: str
#_selftest_code: int = -1
#_selftest_message: str = NA
#_serial_number: str = NA
#_com_interface: AsynComInterface = None
#_com_interface_connected: bool = False
#_com_lock: asyncio.Lock = asyncio.Lock()

-__init__(name: str, mqtt_config: MQTTClientConfig, top_level_topic: str, resource: str = None, id_query_flag: bool = True, reset_flag: bool = True, reset_options: str = None, selftest_flag: bool = True)
-__del__()
-__initialize(resource: str = None, id_query_flag: bool = True, reset_flag: bool = True, reset_options: str = None, selftest_flag: bool = True) «async»
#_id_query(): str «async»
#_reset(reset_options: str = None) «async»
#_revision_query() «async»
#_selftest() «async»
#_sr_query(): str «async»
#_close() «async»
#_command_help(): str
#_handle_command(q_item: list[Any]) «async»
#_handle_subscriptions(topic_levels: list[str], payload: str, message: aiomqtt.message.Message) «async»
#_periodic_action() «async»
#_periodic_start() «async»
#_periodic_stop() «async»
#_publish_topics() «async»
#_start(subscribe_task: bool = False, periodic_interval: pint.Quantity = -1. * ureg.s) «async»
#_stop() «async»
#_subscribe_topics(): list[str]
#_sw_revision(): str
+initialize(resource: str = None, id_query_flag: bool = True, reset_flag: bool = True, reset_options: str = None, selftest_flag: bool = True) «async»
+id_query() «async»
+reset(reset_options: str = None) «async»
+revision_query() «async»
+selftest() «async»
+close() «async»
+get_idn(): str
+get_selftest_code(): int
+get_selftest_message(): str
+get_serial_number(): str
+get_resource(): str
```

MyAsyncBase & MyAsyncBaseDevice



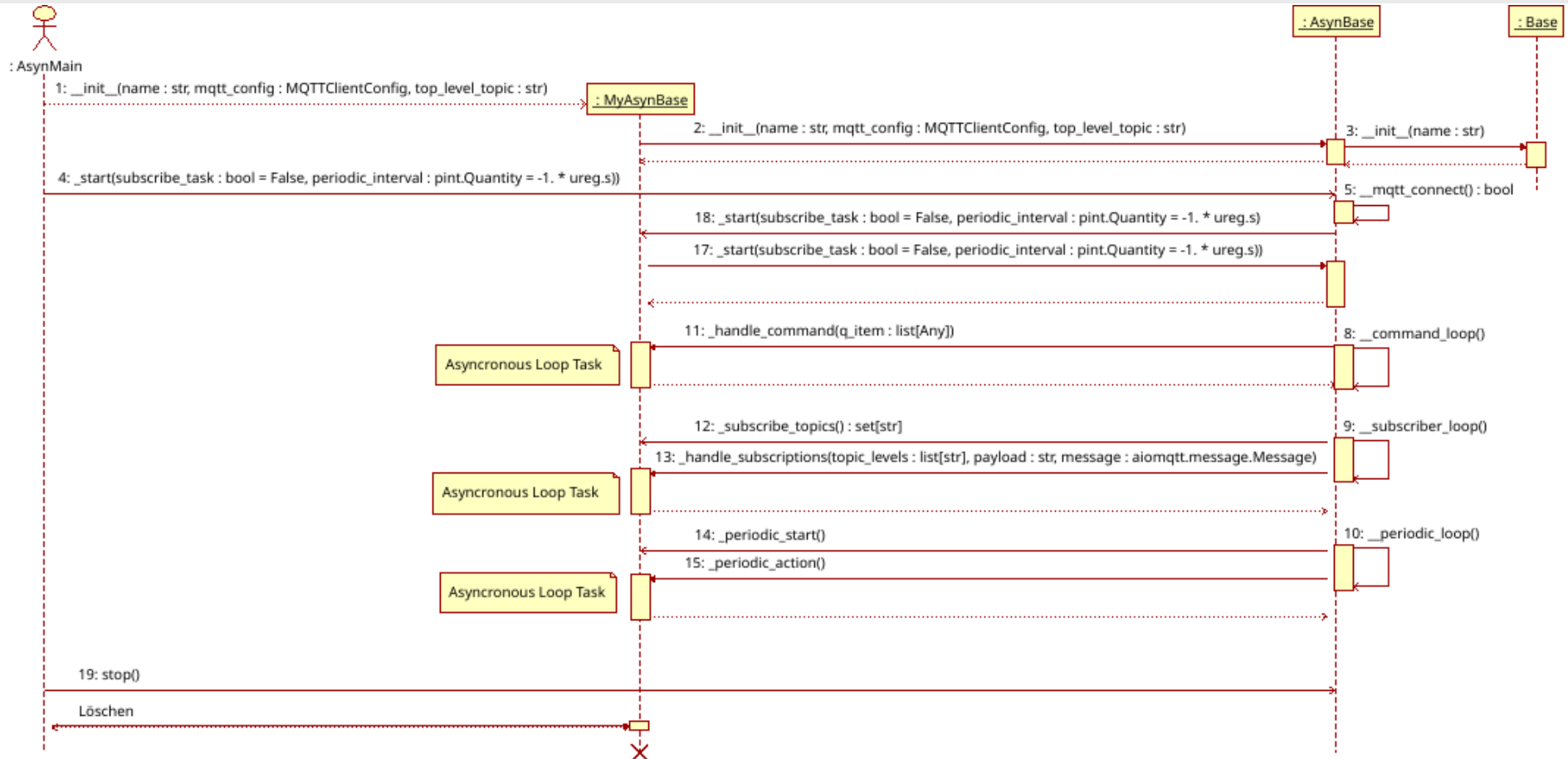
MyAsyncBase

```
#_parameter : int = -1
-__init__(name : str, mqtt_config : MQTTClientConfig, top_level_topic : str)
-__del__()
-__command_help() : str
-_handle_command(q_item : list[Any]) «async»
-_handle_subscriptions(topic_levels : list[str], payload : str, message : aiomqtt.message.Message) «async»
-_periodic_start() «async»
-_periodic_action() «async»
-_periodic_stop() «async»
-_publish_topics() «async»
-_start(subscribe_task : bool = False, periodic_interval : pint.Quantity = -1. * ureg.s) «async»
-_stop() «async»
-_subscribe_topics() : set[str] «async»
-_sw_revision() : str
+command(parameter : int = 0) : int «async»
+get_parameter() : int
+set_parameter(parameter : int = 0) : int
```

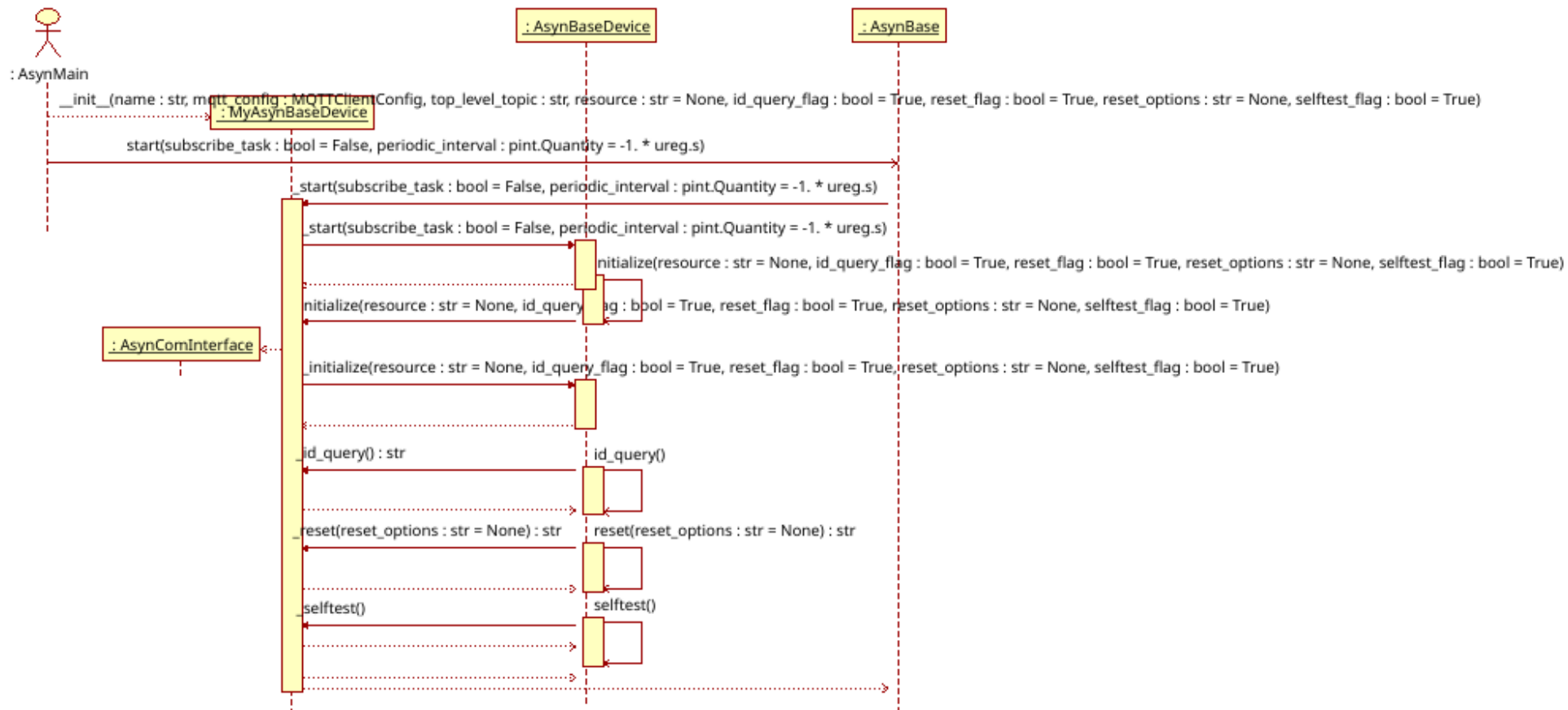
MyAsyncBaseDevice

```
#_parameter : int = -1
-__init__(name : str, mqtt_config : MQTTClientConfig, top_level_topic : str, resource : str = None, id_query_flag : bool = True, reset_flag : bool = True, reset_options : str = None, selftest_flag : bool = True)
-__del__()
-_close() «async»
-__command_help() : str
-_handle_command(q_item : list[Any]) «async»
-_handle_subscriptions(topic_levels : list[str], payload : str, message : aiomqtt.message.Message) «async»
-_id_query() : str «async»
-_initialize(resource : str = None, id_query_flag : bool = True, reset_flag : bool = True, reset_options : str = None, selftest_flag : bool = True)
-_periodic_action() «async»
-_periodic_start() «async»
-_periodic_stop() «async»
-_publish_topics() «async»
-_reset(reset_options : str = None) : str «async»
-_revision_query() «async»
-_selftest() «async»
-_sn_query() : str «async»
-_start(subscribe_task : bool = False, periodic_interval : pint.Quantity = -1. * ureg.s) «async»
-_stop() «async»
-_subscribe_topics() : set[str] «async»
-_sw_revision() : str
+command(parameter : int = 0) : int
+get_parameter() : int
+set_parameter(parameter : int = 0) : int
```

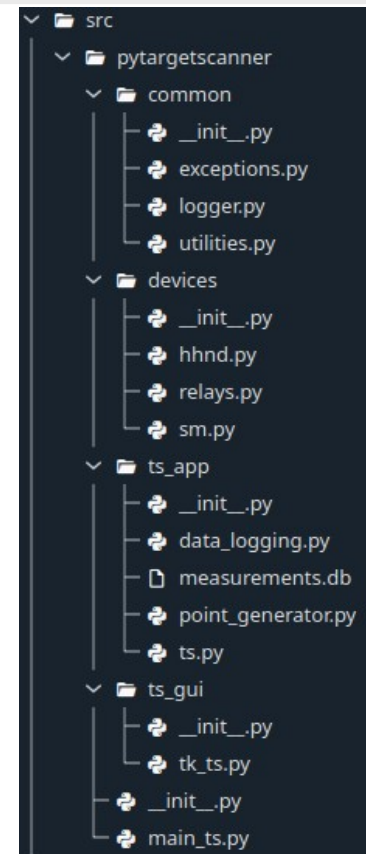
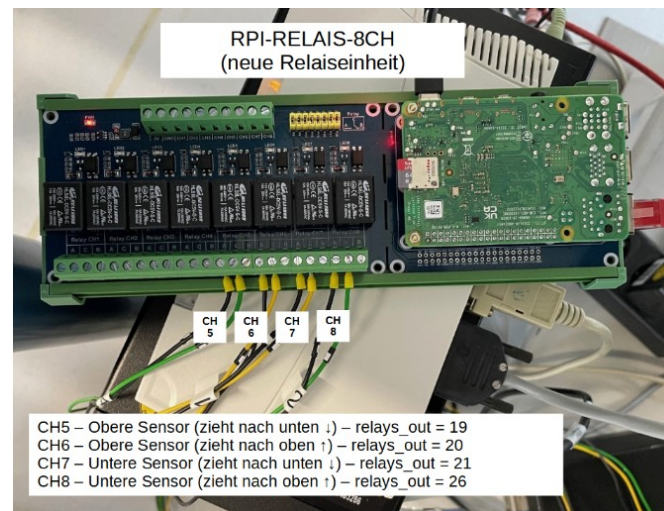
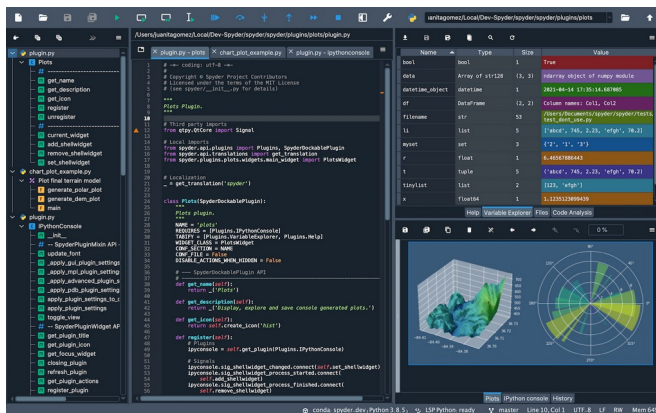
AsynBase Start Sequence



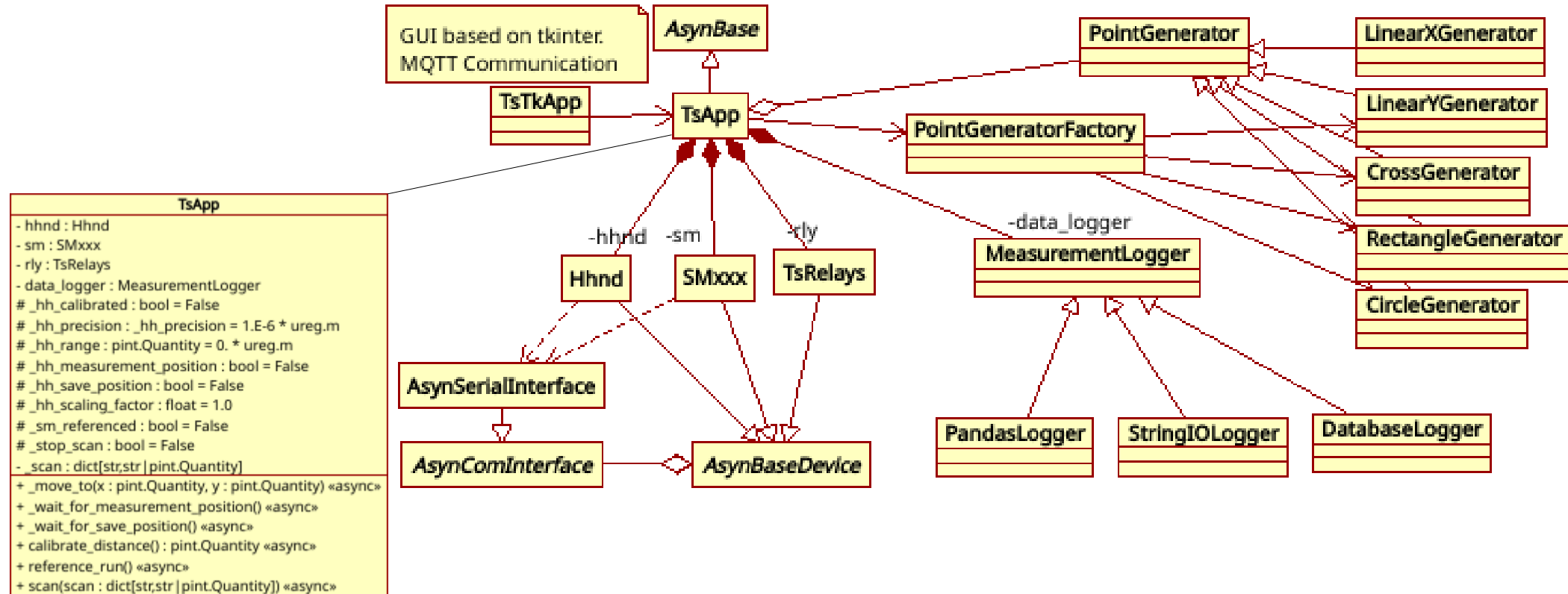
AsyncBaseDevice Start Sequence

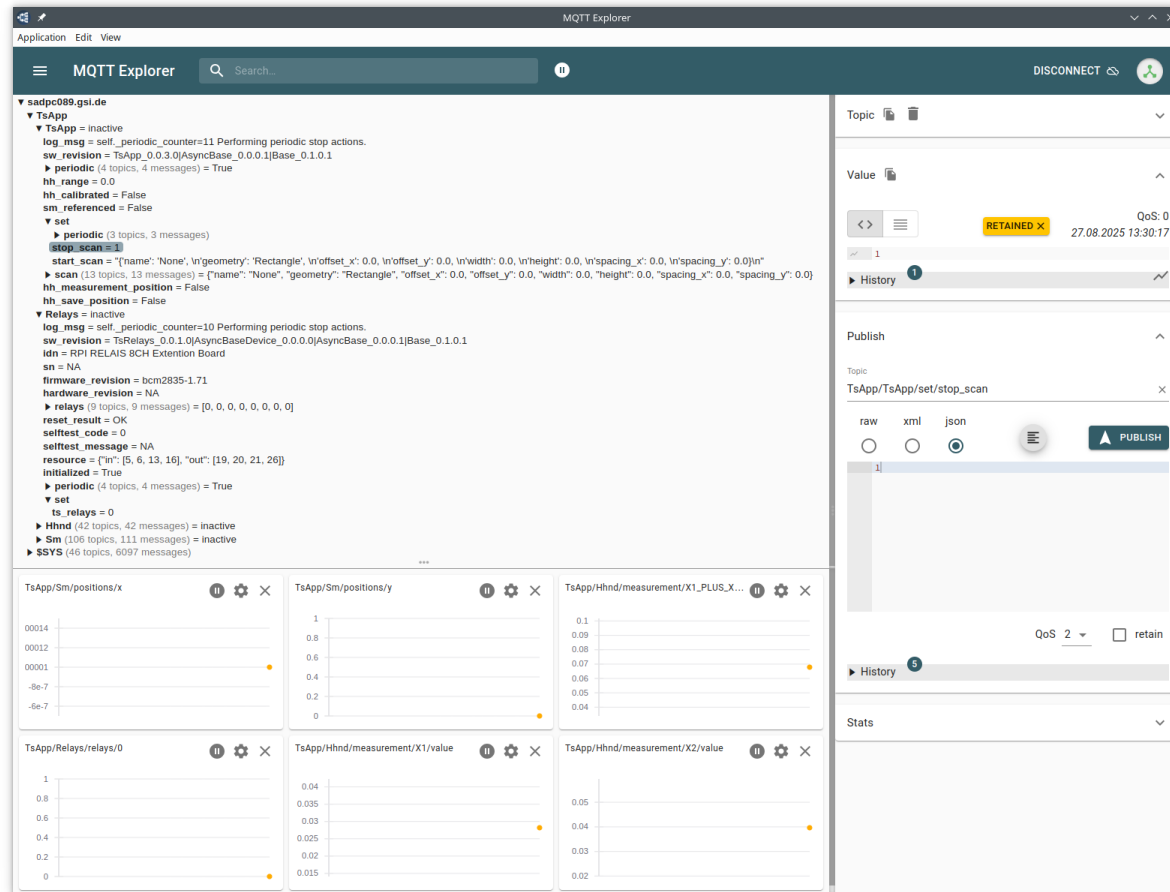


- Git: <https://git.gsi.de/EKS/Python/ACDAQ/TargetScanner>
- Raspberry Pi - Relais-Modul, 8 Channel, Hutschiene
https://www.reichelt.de/de/de/shop/produkt/raspberry_pi_-_relais-modul_8_channel_hutschiene_hls8l-dc5v-s-253984?r=1
- USB-RS232 4-Port
- Python 3.12 / Spyder
 - IDE: Spyder <https://www.spyder-ide.org/>



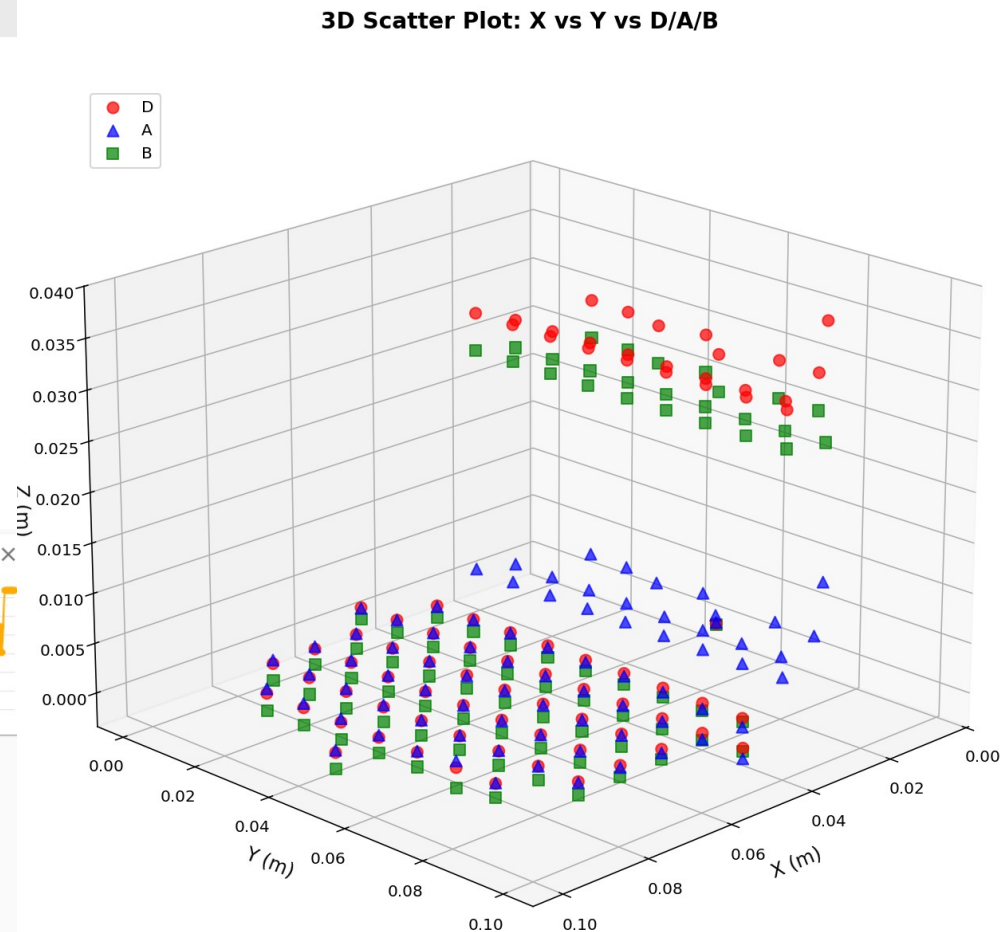
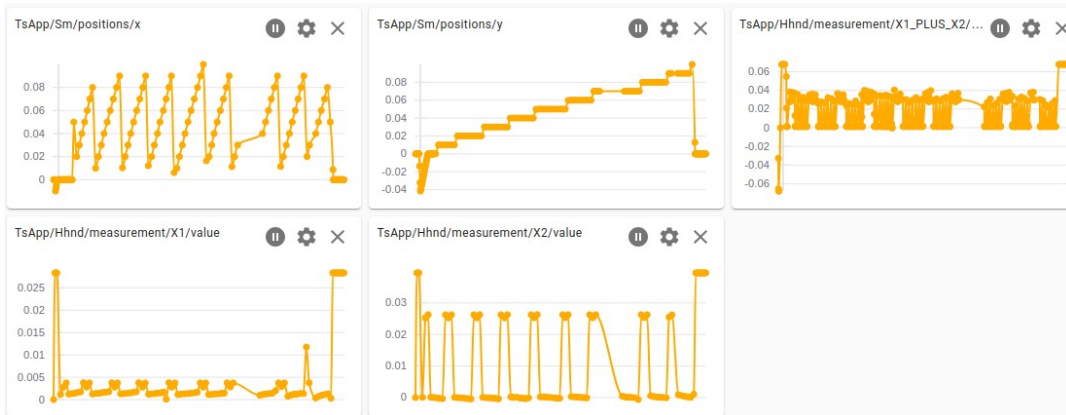
Targetscanner Class Diagram





Ein erster Scan via MQTT-Explorer!

```
{  
  "name": "Test",  
  "geometry": "Circle",  
  "offset_x": 0.0,  
  "offset_y": 0.0,  
  "width": 0.1,  
  "height": 0.1,  
  "spacing_x": 0.01,  
  "spacing_y": 0.01  
}
```



Targetscanner GUI: Connect & Publish



TsTkApp MQTT Client

File Help

Connection Publish Subscribe Messages TsApp ND231 Relays SM

Connection Settings

Broker Hostname: sadpc089 Port: 1883

Client ID: TsTkApp-client-1756201041 Protocol Version: MQTT 5.0

Authentication

Username: Password:

Will Message

☒ Use Will Message

Topic: clients/status

Message: {"status": "offline", "timestamp": "2025-05-17T10:00:00Z"}

QoS: ☐ 0 ☒ 1 ☐ 2 ☒ Retain

MQTT 5.0 Will Properties

Properties (JSON): {"message_expiry_interval": 3600, "content_type": "application/json"}

Connect Disconnect

TsTkApp MQTT Client

File Help

Connection Publish Subscribe Messages TsApp ND231 Relays SM

Publish Message

Topic: home/sensors/temperature

Message:

```
{
  "temperature": 22.5,
  "unit": "celsius",
  "timestamp": "2025-05-17T10:00:00Z"
}
```

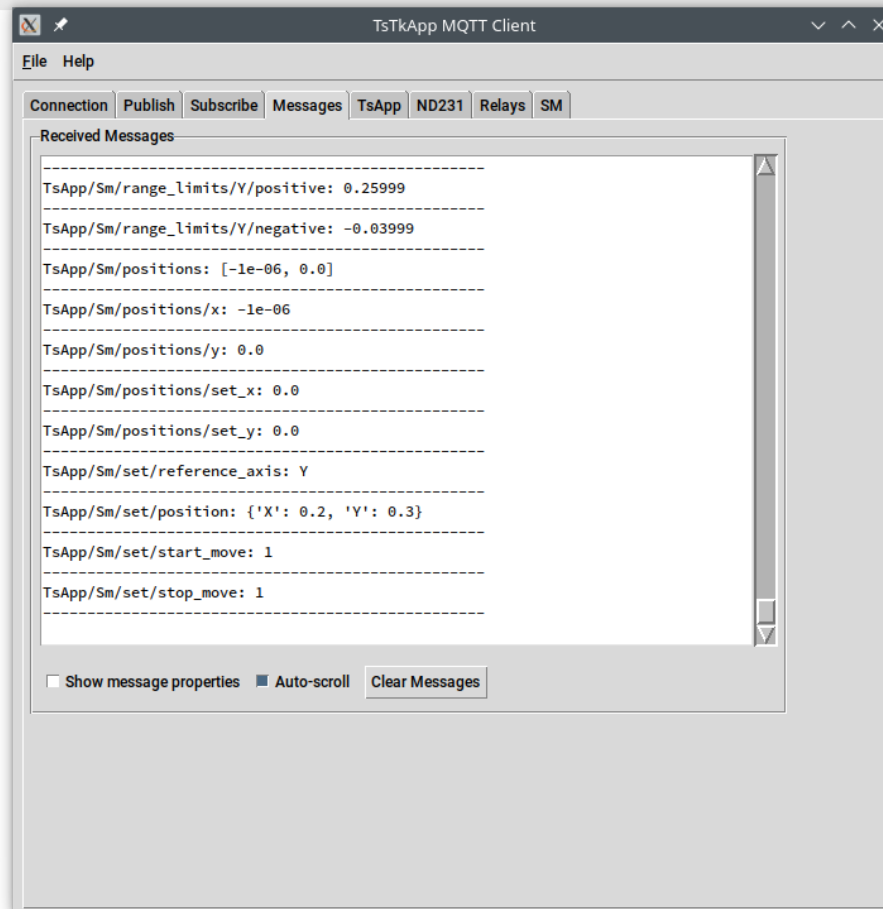
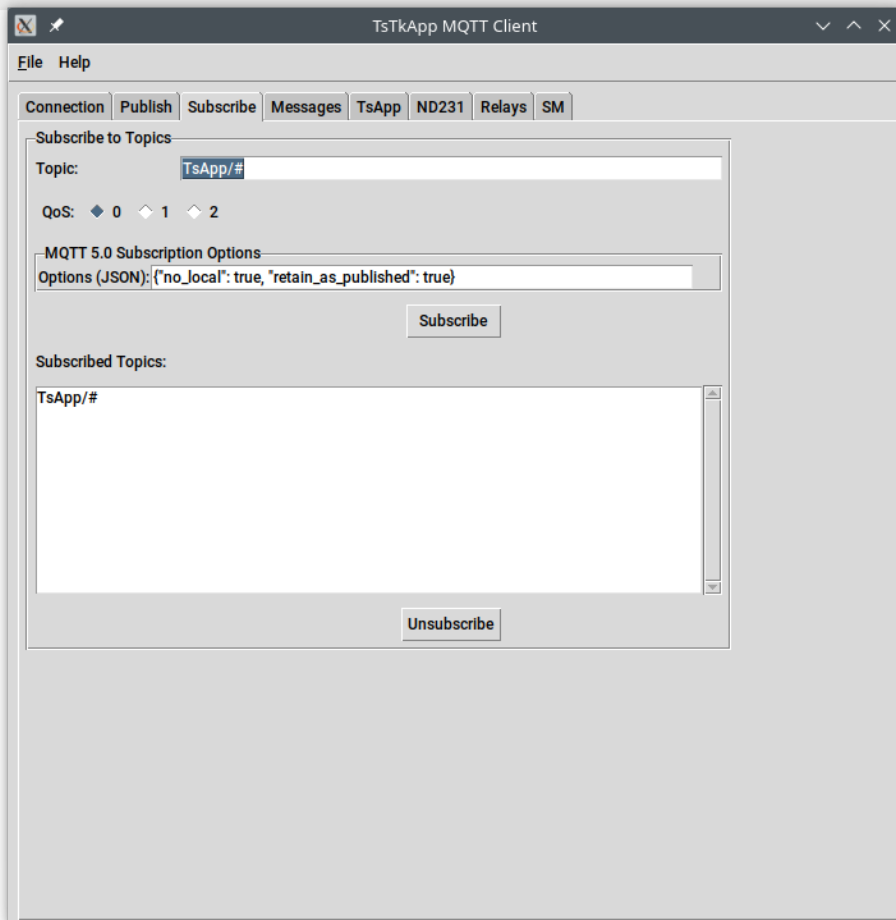
QoS: ☒ 0 ☒ 1 ☐ 2 ☐ Retain

MQTT 5.0 Properties

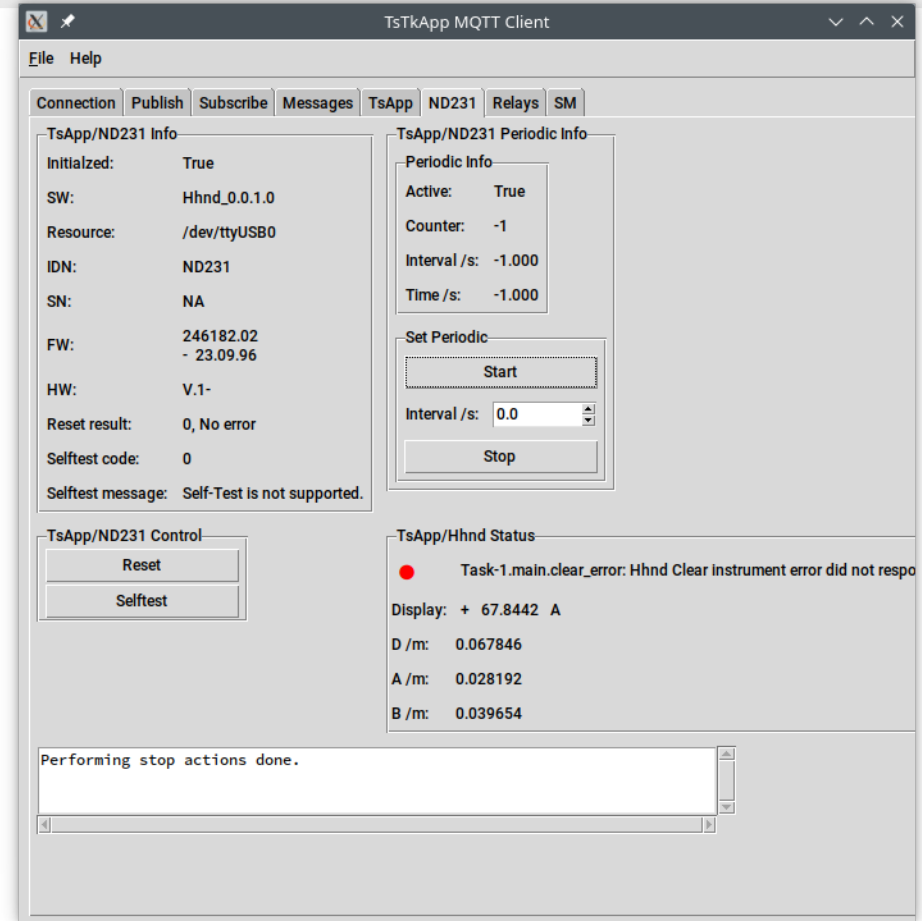
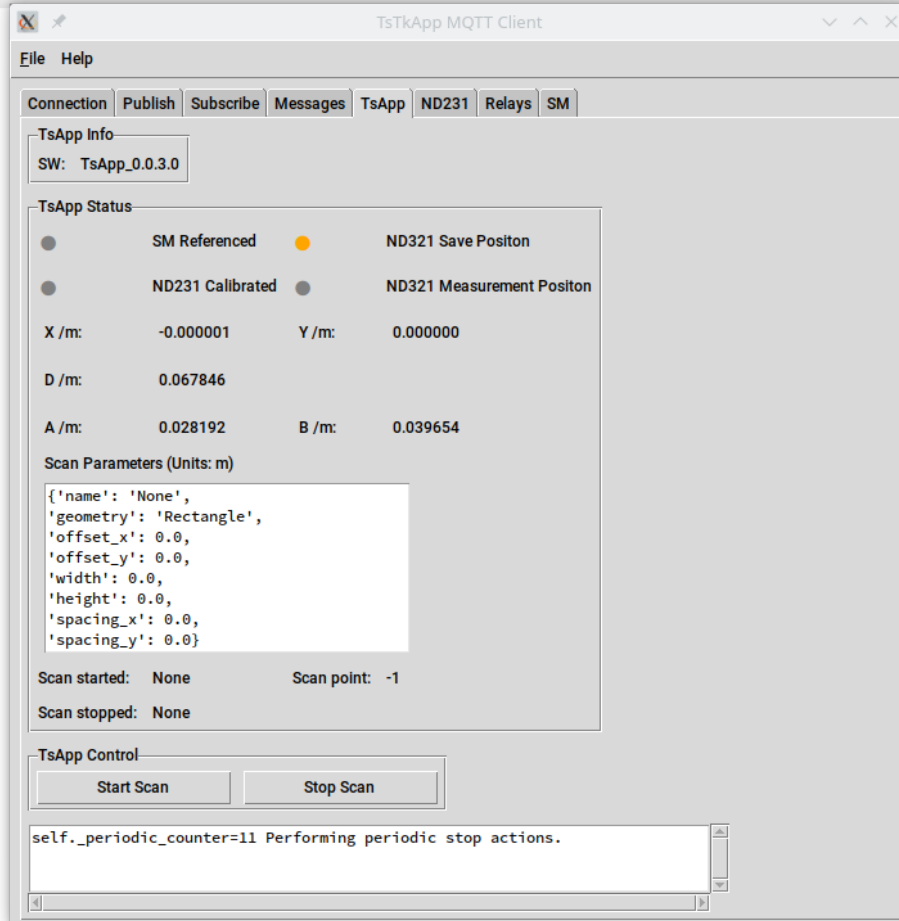
Properties (JSON): {"content_type": "application/json", "message_expiry_interval": 3600}

Publish

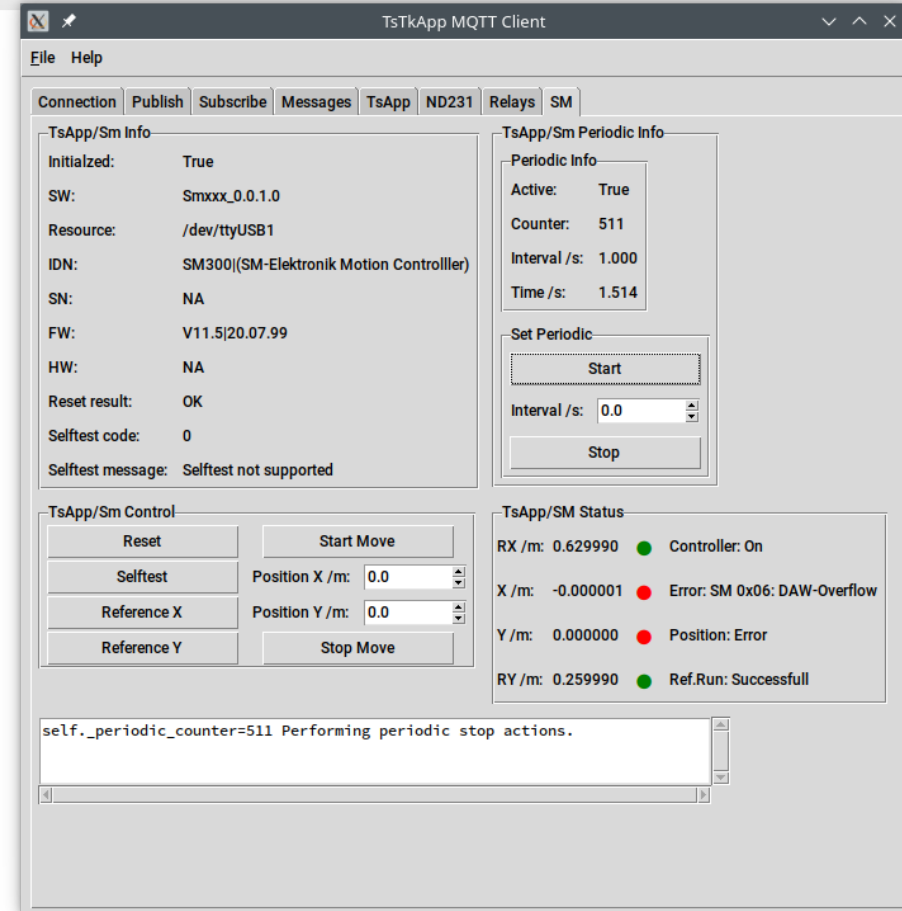
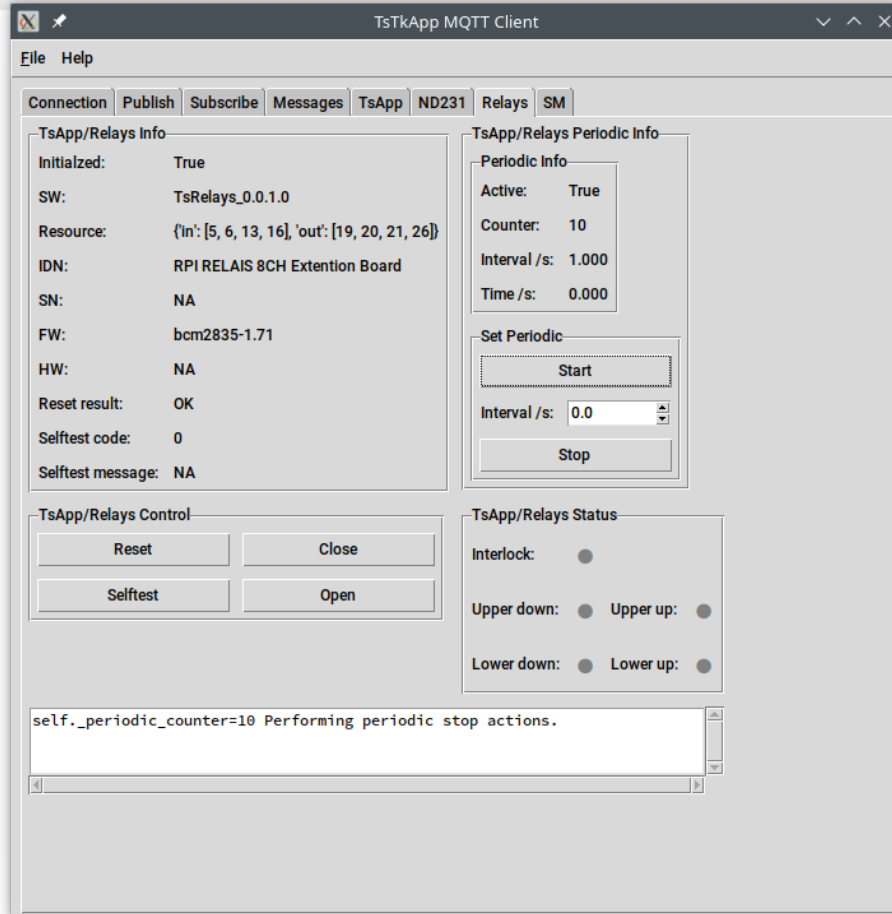
Targetscanner GUI: Subscribe & Monitor



Targetscanner GUI: TsApp & ND231

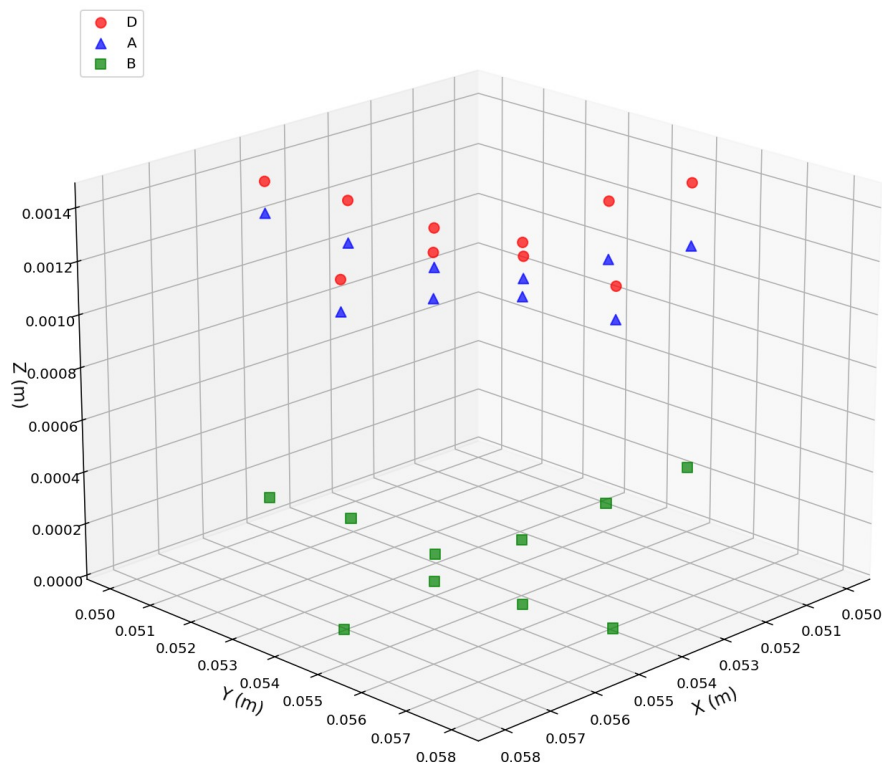


Targetscanner GUI: Relays & SM

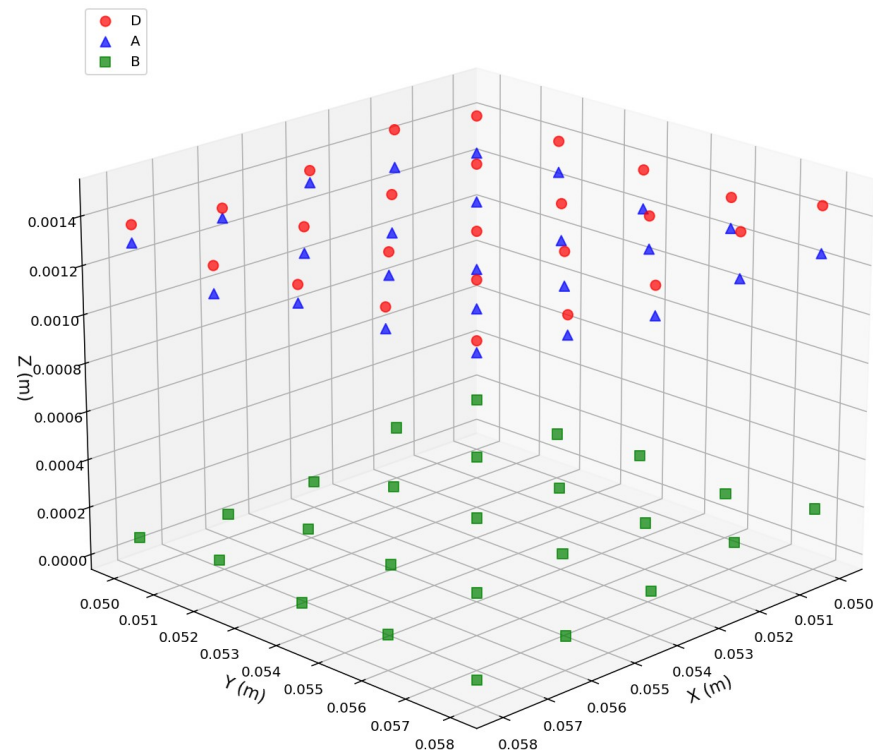


First Measurements with GUI

3D Scatter Plot: X vs Y vs D/A/B



3D Scatter Plot: X vs Y vs D/A/B



- ***Automated Thickness Measurement of Targets and Degraders***
Virtuelle Instrumente in der Praxis - Begleitband zum Kongress VIP 2001
Rahman Jamal /Hans Jaschinski (Hrdg.)
Heidelberg Hüthig, 2001
Praxiswissen Elektronik-Industrie: Meßtechnik
ISBN 3-7785-2829-7
- ***Python 3 – Das umfassende Handbuch***
Johannes Ernesti, Peter Kaiser
Rheinwerk Verlag, Bonn
5. Aktualisierte Auflage 2017, korrigierter Nachdruck 2018
ISBN 978-3-8362-5846-7
- ***Numerisches Python***
Arbeiten mit NumPy, Matplotlib und Pandas
Bern Klein
Carl Hanser Verlag München, 2019
ISBN 978-3-44645076-9
- ***Python Asyncio Mastery***
Discover Modern Asynchronous Programming In Python With Asyncio
Jason Brownlee, 2023
<https://superfastpython.com/python-asyncio-mastery/>
- Python Mattermost Community@GSI: <https://mattermost.gsi.de/python>

- Thanks for interest.
 - Special thanks to Dennis Klein for support and discussions related to Python.
- Discussion is open.
 - LabVIEW and Python Code can be shown on request.