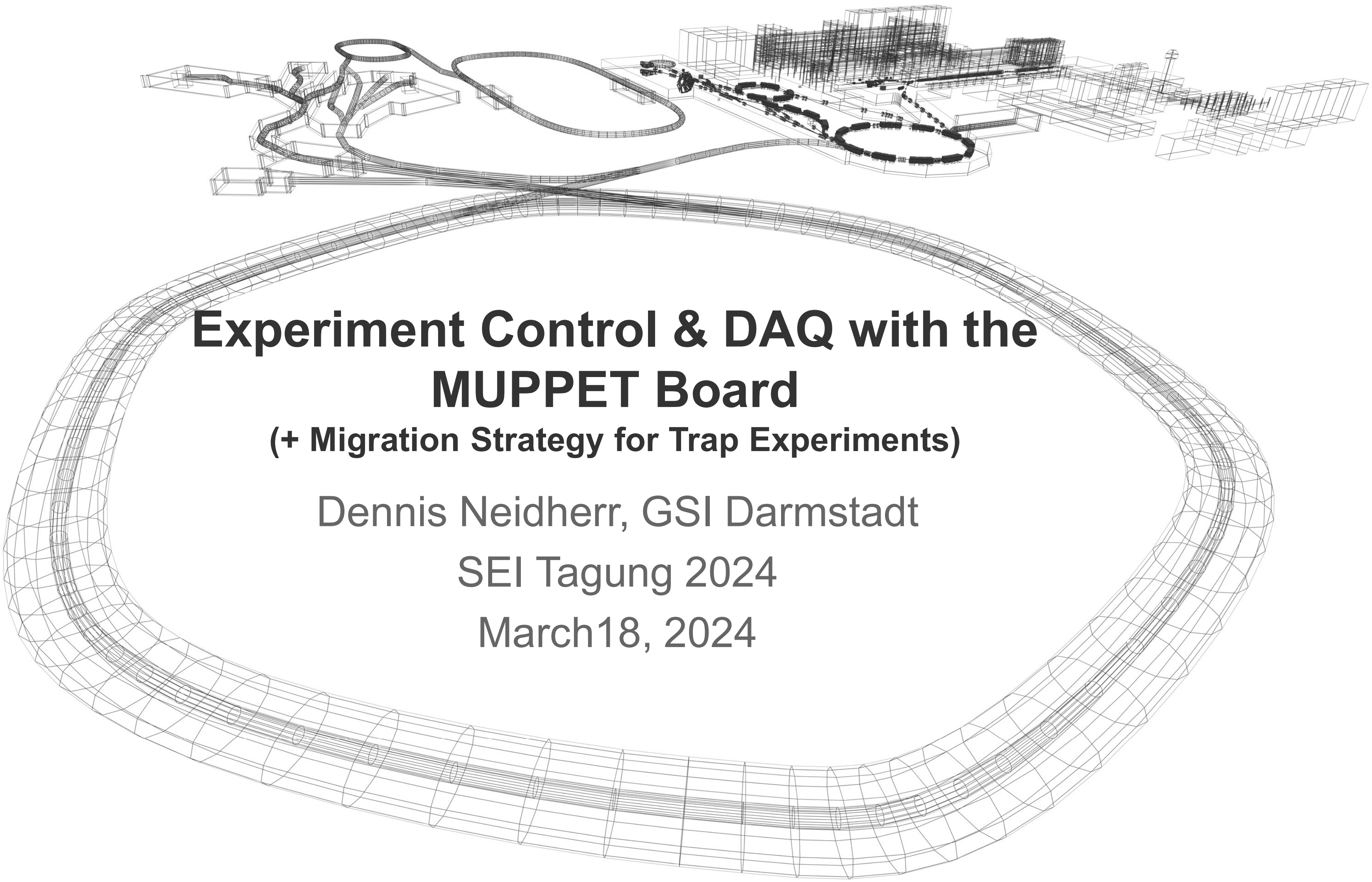


Experiment Control & DAQ with the MUPPET Board

(+ Migration Strategy for Trap Experiments)

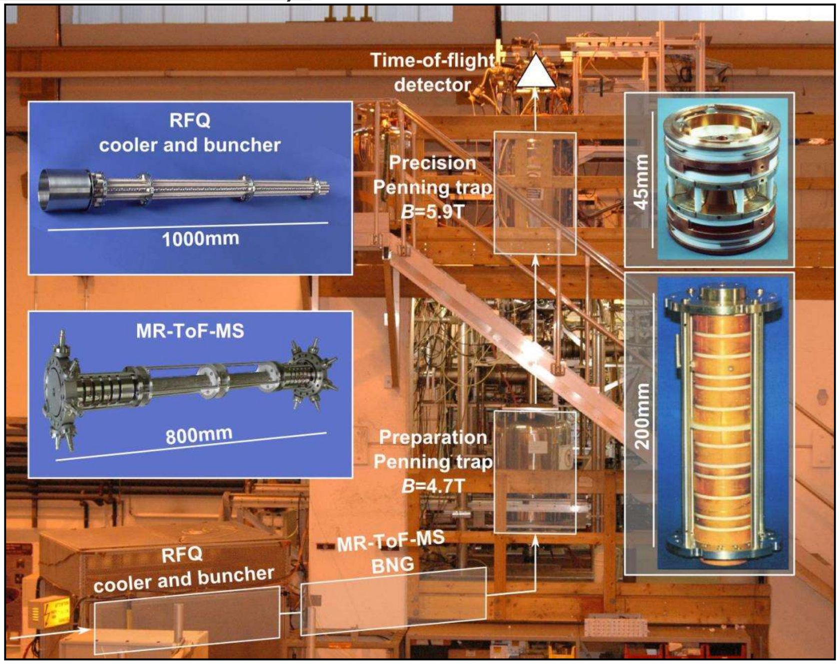
Dennis Neidherr, GSI Darmstadt

SEI Tagung 2024

March18, 2024

Motivation: Again Migration from LabVIEW to “something else”

- In the past, two LabVIEW based control system frameworks for small- and medium-size experiments (*CS-framework* and **CS++**):
  - object orientated
  - distributed
  - event-driven
- The goal was (and is) that students could run, maintain, and modify the setup by themselves (with remote assistance) ➔ LabVIEW (one tool for everything)
- Example (ISOLTRAP /CERN):



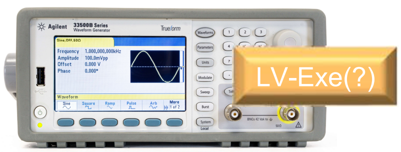
Control of mainly
- power supplies, and
- frequency generators

\+ ns timing and DAQ

Structure of “trapper” control systems

Devices

- distributed (often PC needed)
- can be quickly replaced



Pub-Sub
Ethernet

Sequencer

Pub-Sub
Ethernet

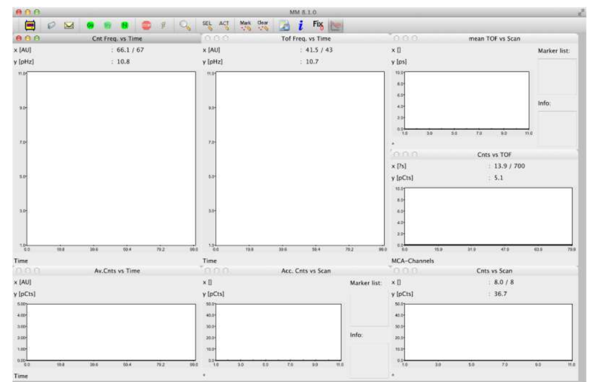
LabVIEW

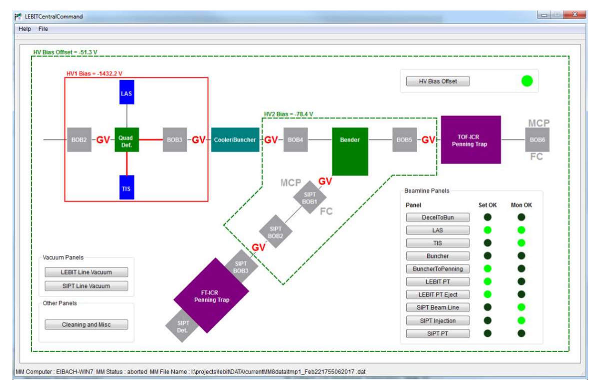
Qt

User Interfaces

- configuration of measurement
- online analysis

Pub-Sub
Ethernet

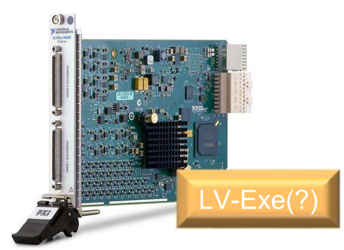




Pub-Sub
Ethernet

NI FPGA boards

2 boards: (PPG + TDC)



Structure of “trapper” control systems

Devices

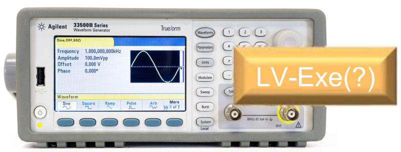
- distributed (often PC needed)
- can be quickly replaced

LabVIEW

Qt

User Interfaces

- configuration of measurement
- online analysis

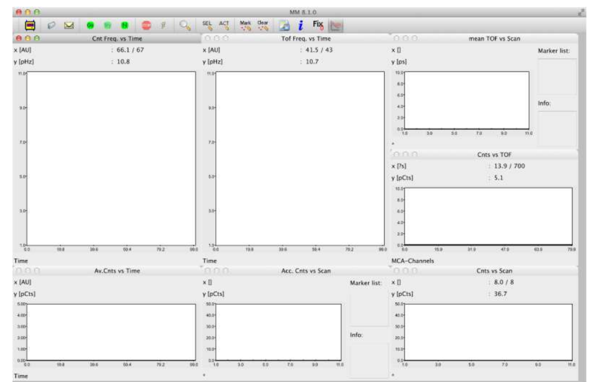


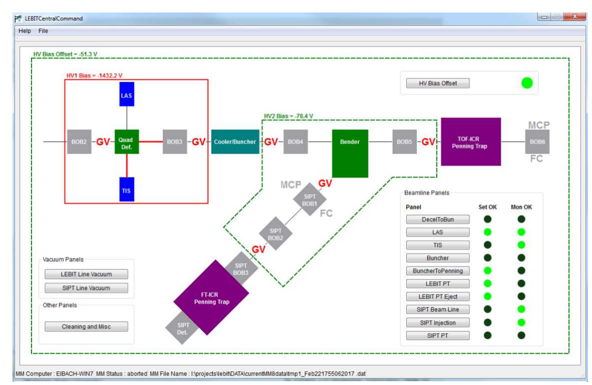
Pub-Sub
Ethernet

LabVIEW abandoned

Pub-Sub
Ethernet

Pub-Sub
Ethernet

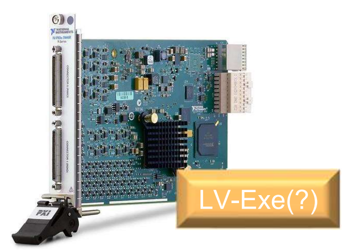




NI FPGA boards

2 boards: (PPG + TDC)

Pub-Sub
Ethernet

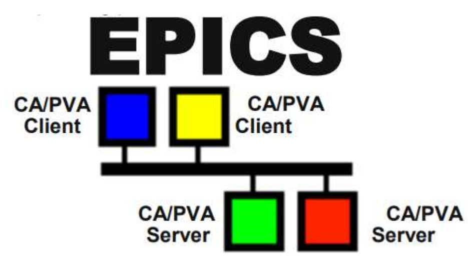


Migration strategy

1. Convert the Sequencer to Python (easy to manipulate in the future by everyone),

2. for the hardware layer and the network communication switch to the **E**xperimental **P**hysics and **I**ndustrial **C**ontrol **s**ystem (EPICS), and

    - Publisher / Subscriber architecture like in our old frameworks
    - Configuring instead of Programming
    - Very often: logic included in network variables called "records"
    - In this project: Use records only as network variables for the moment



3. search for an alternative for the two NI FPGA cards in use.

Colleagues in our department had an idea…

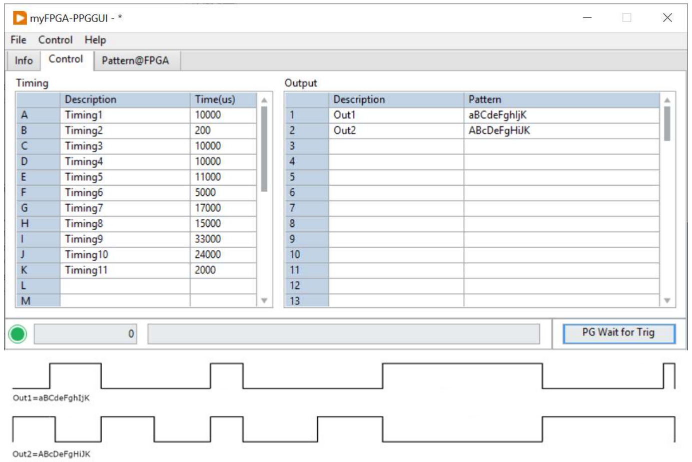
Multi-purpose FPGA - MUPPET

Some specifications:

- Xilinx ARTIX-7 FPGA
- 8 LEMO NIM or LVTTL (programable) outputs
- 8 LEMO 50 Ohm any-in inputs (NIM and TTL adjustable per input)
- with extension adapters more inputs / outputs can be realized
- input for an external clock
- Raspberry PI connector with SPI interface

Two built-in MUPPET applications

- Pulse-Pattern-Generation (PPG):
  - Some specs:
    - 833 ps granularity
    - Jitter defined by ext. clock (<< 10 ps)
    - Accuracy defined by ext. clock (Rb: ppb)
  - Different methods to configure pattern:
    - single pulses: Pulse(channel, start, length)
    - complete sequence: Sequence([Timing], [Output])
  - An external trigger can be used for synchronization
  - Publish status (running, finished, waiting for trigger, etc)



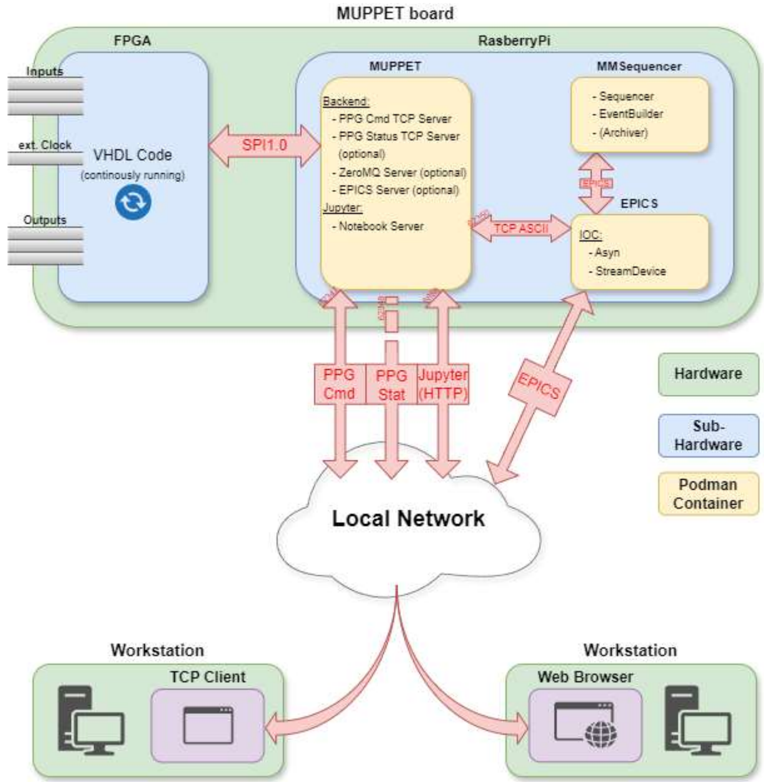
- Time measurements (TDC):
  - Measure timing difference between external trigger and (n) data input(s)
  - Create histogram(s) with acquired timing data
  - Binning with 415 ps

- It might be possible to improve here presented specs if needed

- Both applications need to be backwards compatible with current LabVIEW solutions

Software Implementation

- Connect a Raspberry PI for different user interfaces
- All software (including VHDL code): open source and can be modified by user
- 3 podman containers (later maybe only 2):
  - MUPPET for the direct communication with the FPGA via TCP/IP server(s)
  - EPICS for the creation of an IOC and the communication with the MUPPET container
  - MMSequencer for the full control of a trap experiment
- still has to be decided if containers are helpful at all



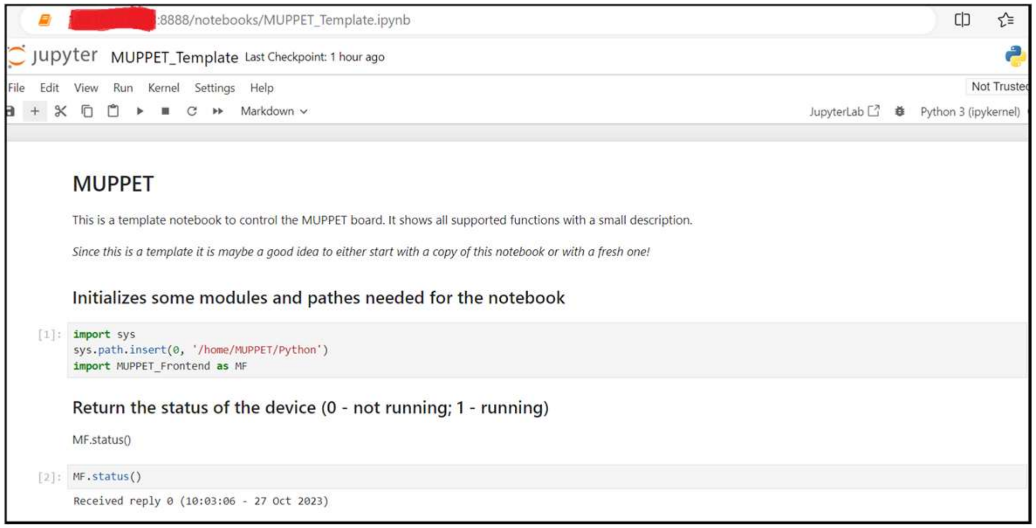
MUPPET Backend script

- Exclusive access to the FPGA
- At the moment 4 running threads (TDC not yet implemented):
    - **ppg_cmd**: Simple TCP/IP server for ASCII cmds from outside (request-reply)
    - **ppg_status**: TCP/IP server which automatically publishes status changes
    - **zmq**: A zeroMQ server used internally to pass data between threads
    - **epics**: Another TCP/IP server whose single purpose is to communicate with StreamDevice module in EPICS container

```python
def execute_command(data, zeromq_int_socket):
    """
    Send a command to the MUPPET FPGA and returns the answer. This is the only place
    in code where the FPGA access happens to be sure that these are atomic actions.
    The command and possible parameters are separated via '_' in data.
    """
    global init_error
    splidata = data.split('_', 1)
    cmd = splidata[0]
    if init_error == 'No error' or cmd == 'INIT':  # if Init_error active only a new INIT is allowed
        try:
            global cycle_period  # the cycle_period will be always the longest channel
            global status_active
            if len(splidata) > 1:
                para = splidata[1]
            if cmd == 'INIT':  # more a reset
                initMUPPET()
                if init_error == 'No error':
                    return 'INIT_OK'
                else:
                    return init_error
            elif cmd == 'STARTPPG':  # start PPG; parameter determines mode
                if int(para) == 0:
                    ms.seq_free_running(1)
                    ms.seq_ext_trig_en(0)
                elif int(para) == 1:
                    ms.seq_free_running(0)
                    ms.seq_ext_trig_en(1)
                elif int(para) == 2:
                    ms.seq_free_running(0)
                    ms.seq_ext_trig_en(0)
                    ms.seq_arm_single_shot()
                ms.sequencer_enable()  # enable MUPPET pulser
                return 'STARTPPG_OK'
            elif cmd == 'ABORTPPG':  # abort PPG
                ms.seq_free_running(0)
                ms.seq_ext_trig_en(0)
                ms.sequencer_reset()
                return 'ABORTPPG_OK'
            elif cmd == 'PULSE':  # configure one PPG pulse with times
                pulsepara = para.split('_', 2)
                ms.pulse(ch=int(pulsepara[0]), start=pulsepara[1], width=pulsepara[2])
                ms.enable_ch(int(pulsepara[0]))
                if (float(pulsepara[1]) + float(pulsepara[2])) > cycle_period:
                    cycle_period = float(pulsepara[1]) + float(pulsepara[2])
                    ms.set_cycle_period(str(cycle_period))
                    printMsg('Set new cycle period: ' + str(cycle_period), 1)
                ms.configure_ppg_channels()
                ms.seq_update_config()
                return 'PULSE_OK'
```

MUPPET control interfaces

- Idea is to offer many different interfaces for the user to choose from:
  - TCP/IP request-reply (for example for an available driver in LabVIEW)
  - EPICS (via StreamDevice)
  - ZeroMQ (the internal ZeroMQ server can be made accessible from outside relatively easy)
  - Jupyter (very useful for very simple applications since it requires only a network PC with a running browser)
  - (MQTT, etc.)



Mass measure sequencer

- For compatibility reasons logic and interfaces cannot be changed (exception step from DIM to EPICS)
- Three independent threads / scripts running:
  - **Sequencer:** controls the measurement sequence:
    i. Init = configure complete setup
    ii. Set Scan Device
    iii. Start PPG
    iv. Wait until PPG is finished
    v. Start with DAQ
    vi. Stop or continue with step ii.
  - **EventBuilder:** Receives data from the DAQ triggered by the sequencer and creates an event and publishes it (the online GUI and the Archiver will subscribe)
  - **Archiver:** subscribes to “EventBuilder:eventData” records and saves the content into a file with a specific format (should probably not run on the RasPI, instead it should run on the node with the online GUI)

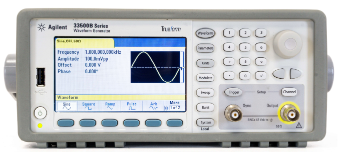
```python
def __get_value_from_device_callback(self, value):
    # print('Got data: ' + str(value, encoding='utf-8'))  # DEBUG line
    if int(str(value, encoding='utf-8').split('_')[1][:-1], 16) == self.__readbackrnd:
        # print('correct callback received: ', int(str(value, encoding='utf-8').rsplit('_')[1][:-1], 16))  # DEBUG Line
        self.__get_return_value = str(value, encoding='utf-8').split('_')[0]
        # print('Return-Value: ' + self.__get_return_value)  # DEBUG line
        self.__pvmon.close()
        self.__get_value_lock.release()

async def __get_value_from_device(self, url, timeout):
    self.__readbackrnd = random.randint(0, 100000)  # generate random number to be sent with the command as ID
    await self.__get_value_lock.acquire()
    self.__pvmon = camonitor(url, self.__get_value_from_device_callback)
    self.__trigger_url = url[:url.rfind(':') + 1] + 'Trigger' + url[url.rfind(':') + 1:]  # EPICS Naming Convention used: See EPICS db file!
    # print('Trigger-URL: ' + self.__trigger_url)  # DEBUG line
    # print('Read-URL: ' + url)  # DEBUG line
    await caput(self.__trigger_url + '.A', self.__readbackrnd)
    # print('Try to get Data')  # DEBUG line
    tstart = timer()
    while self.__get_value_lock.locked():  # Maybe there is a better solution for this?
        await asyncio.sleep(0.001)
        if (timer() - tstart) > timeout:
            raise TimeoutError()
    # print('Got Data')  # DEBUG line
    return self.__get_return_value
```

Future “trapper” control system

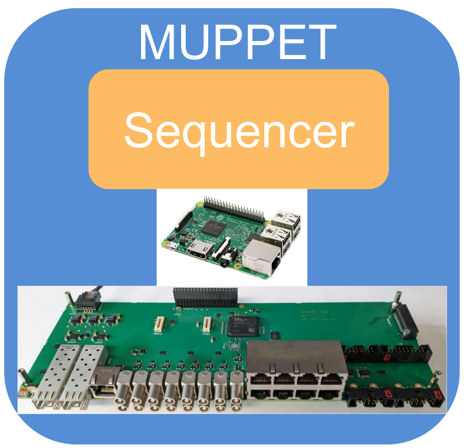
The architecture of future control systems should be much easier:

Devices

- distributed (no PCs involved!)
- Maybe additional RasPIs required
- can be quickly replaced



Pub-Sub
Ehernet

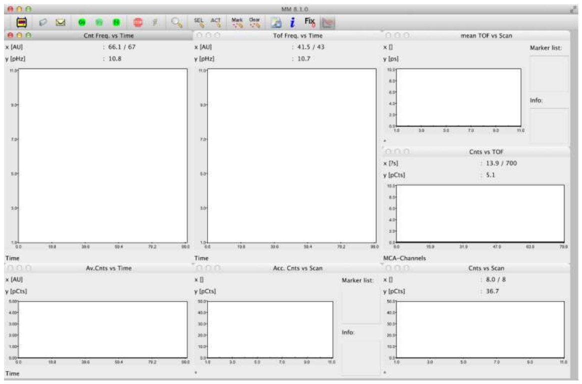


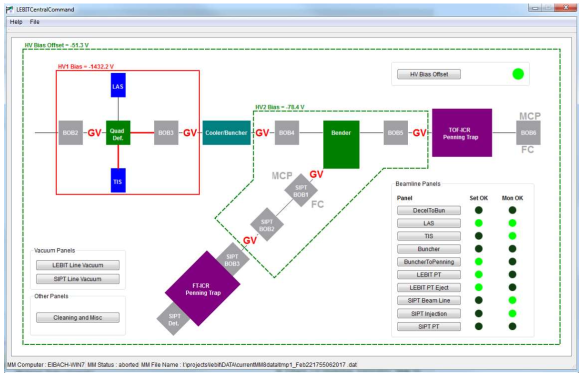
Pub-Sub
Ehernet

Pub-Sub
Ehernet

User Interfaces

- configuration of measurement
- online analysis





Summary

- Nowadays no one has to create own frameworks anymore, there are many open source solutions available for all kind of specific control systems
- For us, a combination of Python, EPICS, and a self-made FPGA-solution seems feasible
- Main goal is flexibility:
  - easy integration in already existing systems, and
  - easy control as a standalone application
  - easy way to modify all parts of the software
- Additional features still to be discussed like:
  - other interfaces like MQTT?
  - more complex pattern (loops)?
  - RasPi-free mode with direct ethernet connection to MUPPET?

Thank you for your attention!

https://git.gsi.de/EKS/Python/MUPPET