UcalManager Server Docs

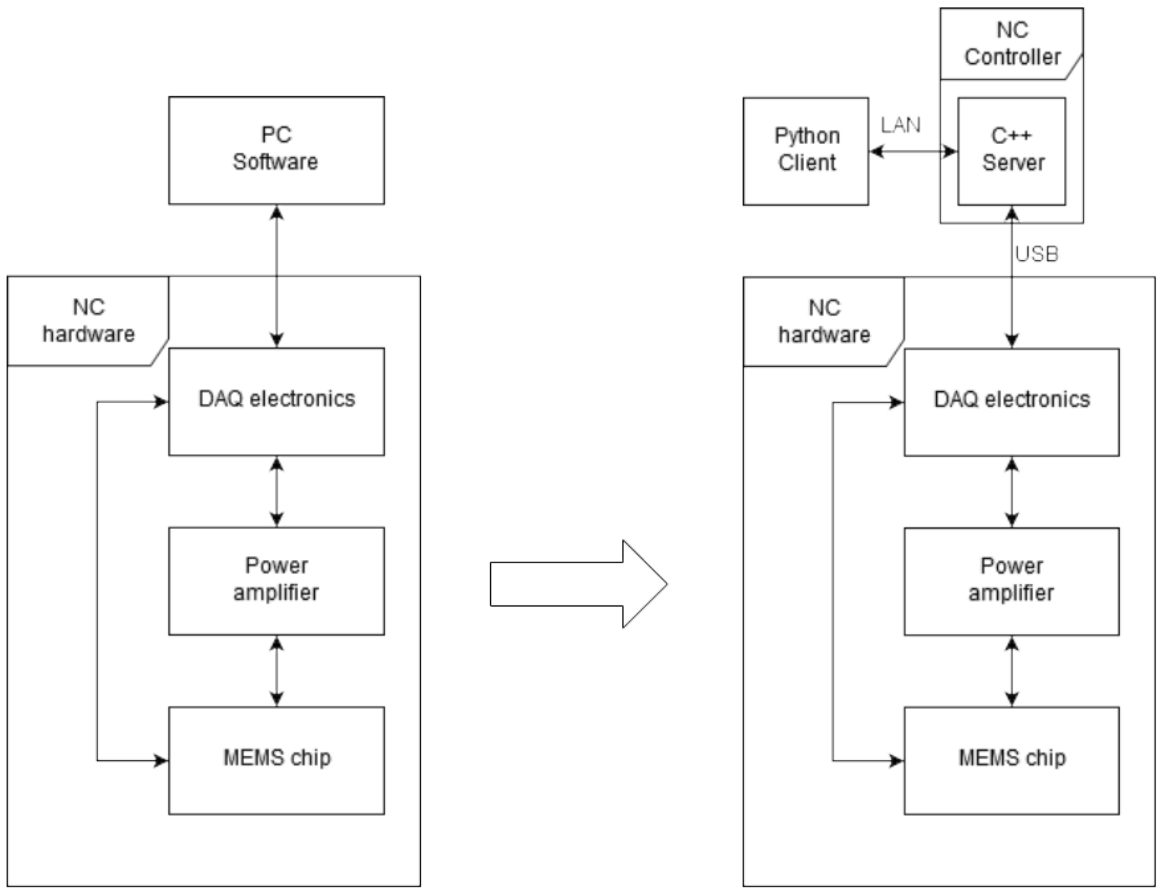
Problem

There are several problems in current design of nanocalorimeter (NC) setup:

• It can't use new chips for heating
• It can't carry out long experiments
• It doesn't provide API for third party
• It is Windows-only because of electronics software dependencies
• It's parts are hard to change without replacing others
• It's GUI is not very user-friendly

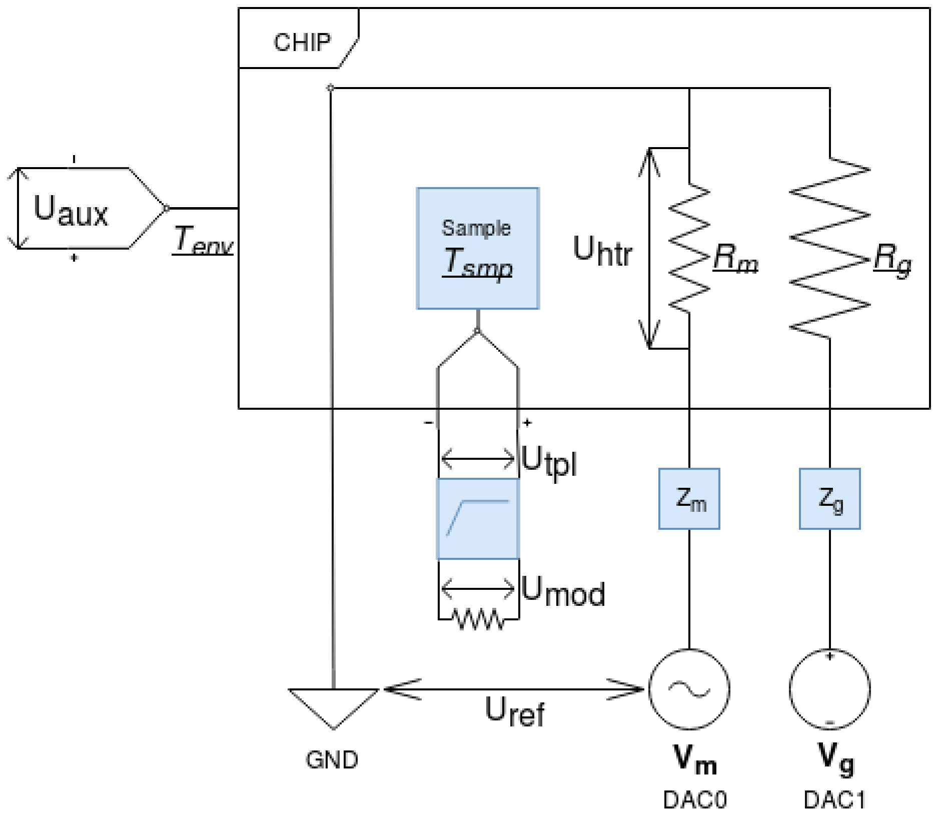
Solution

The solution for these problems is a change of setup structure: to add controller with **Server** software for hardware management and **Client** software as an interface to the controller. Client can be used in a third-party software, such as GUI, for better user experience.



Current state/ project plan comparison

Hardware scheme



There are two main channels that allow to control heating of the chip: **Vm** (voltage applied to “modulation heater” circuit) and **Vg** (voltage applied to “guard heater” circuit). More control channels could be used, e.g. for signaling about heating start.

There are five measurable signals:

- **Uref** - identical to Vm
- **Utpl** - thermopile voltage raw signal
- **Umod** - thermopile voltage signal after high pass filter
- **Uhtr** - voltage measured on modulation heater directly
- **Uaux** - external thermocouple voltage signal

Physical quantities can be calculated based on measured signals .

Concepts

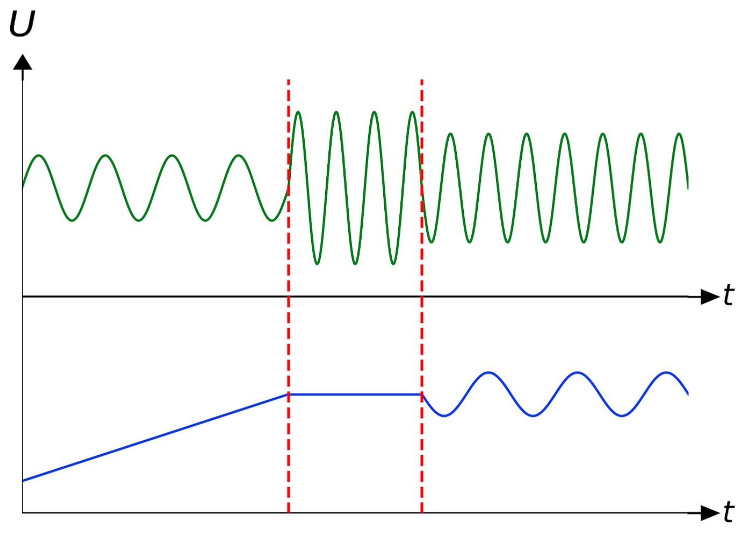
Controlling software is divided on two parts: Server (C++) and Client (Python). Server provides API, which can be used easily with Client. Most of the data management and algorithms (e.g. calibrations) should be implemented in Client to make it easier to extend the functionality of the setup. This means that Server will control hardware using only voltage-level values, not physical quantities (temperature, heat e.t.c.).

This document specifies only Server-related principal functionality, while convenient interface implementation is up to the Client part.

General approach for working with NC can be described by two simple steps:

1. User defines the **Program** (planned actions) that will be applied to the sample
2. User starts program execution, program ends by timer or user action. Measured data is available in real time

Program consists of **Blocks -** composition of voltage profiles at one or several channels. Block either has finite duration or awaits for termination by user command. While Block may control more than one channel, usually only one channel is used.



Example of Program, that consists of three different blocks.

Use Cases

Here are descriptions of the key scenarios, which are expected to become valid at some moment, **with priority decreasing from the first one to the last one**.

Finite-time experiment

User wants to perform an experiment, where the sample is heated/cooled sequentially with specific power/temperature curves and specific time duration.

1. User connects to the Server with Client
2. User specifies heating program as a sequence of Blocks to approximate desired heating curve
3. User commands to start program execution
4. Server starts program execution, measured data is available for User in real time
5. Program **ends on timer**, data is available for User till new program configuration

Infinite-time experiment

User wants to perform an experiment, where Block without time limits is used at the end of the program. It means that the program won't be terminated without Users command. Infinite-time blocks can't use more than one channel. Voltage profile must be periodic (e.g. constant or sine).

1. User connects to the Server with Client
2. User adds initials part of the heating program as a sequence of finite-time Blocks
3. User **adds special Block without time limit** on execution as the last block in Program
4. User commands to start program execution
5. Server starts program execution, measured data is available for User in real time
6. Program **ends on user command,** data is available till new program configuration

Experiment with emergency stop

User is performing one of the experiments listed below. During the experiment User decides to stop the experiment for any reason, but still wants to get measured data.

1. User connects to the Server with Client
2. User specifies heating program as a sequence of Blocks
3. User commands to start program execution
4. Server starts program execution, measurement data is available for User in real time
5. User decides that experiment should be stopped and **commands to cancel** it
6. Server stops program turning off all the heaters; data is available till next program configuration

Experiment with feedback correction

User wants to perform an experiment, where a specific Block is executed with feedback correction based on some variable, e.g. sine signal based on measured amplitude.
Block can be either finite or infinite, however feedback would be useless when duration is too short.

1. User connects to the Server with Client
2. User specifies initial part of the heating program as a sequence of Blocks
3. User **adds feedback loop** option for chosen Blocks and specifies target variable
4. User commands to start program execution
5. Server starts program execution, measurement data is available for User in real time
6. Program ends, data is available till new program configuration

Experiment with start-signal emission

User wants to signal for integration with third-party equipment at the beginning of the Block execution.

The feature is available for all the experiments listed below.

1. User connects to the Server with Client
2. User specifies heating program as a sequence of Blocks
3. User **specifies the Block with signal emission feature**
4. User commands to start program execution
5. Server starts program execution, measured data is available for User in real time
6. When mentioned Block starts execution, **signal is emitted**
7. Program ends, data is available till new program configuration

Experiment with start-signal reception

User wants to specify a heating program and start it on receive of external signal-trigger from third-party equipment.

1. User connects to the Server with Client
2. User specifies heating program as a sequence of Blocks to approximate desired heating curve
3. User **specifies trigger-waiting feature** for the first Block
4. User commands to start program execution, **Server awaits for trigger-signal**
5. **Trigger-signal comes, Server starts program execution**, measured data is available for User in real time
6. Program ends, data is available till new program configuration

Experiment with multi-channel block

User wants to perform an experiment, which uses both heaters simultaneously. This scenario may have various restrictions due to DaqBoard limitations.

1. User connects to the Server with Client
2. User specifies heating program. One of blocks is a **special multi-channel block, that allows to configure several channels**, e.g. constant and sine voltage profiles on two different channels
3. User commands to start program execution
4. Server starts program execution, measured data is available for User in real time
5. Program ends, data is available till new program configuration