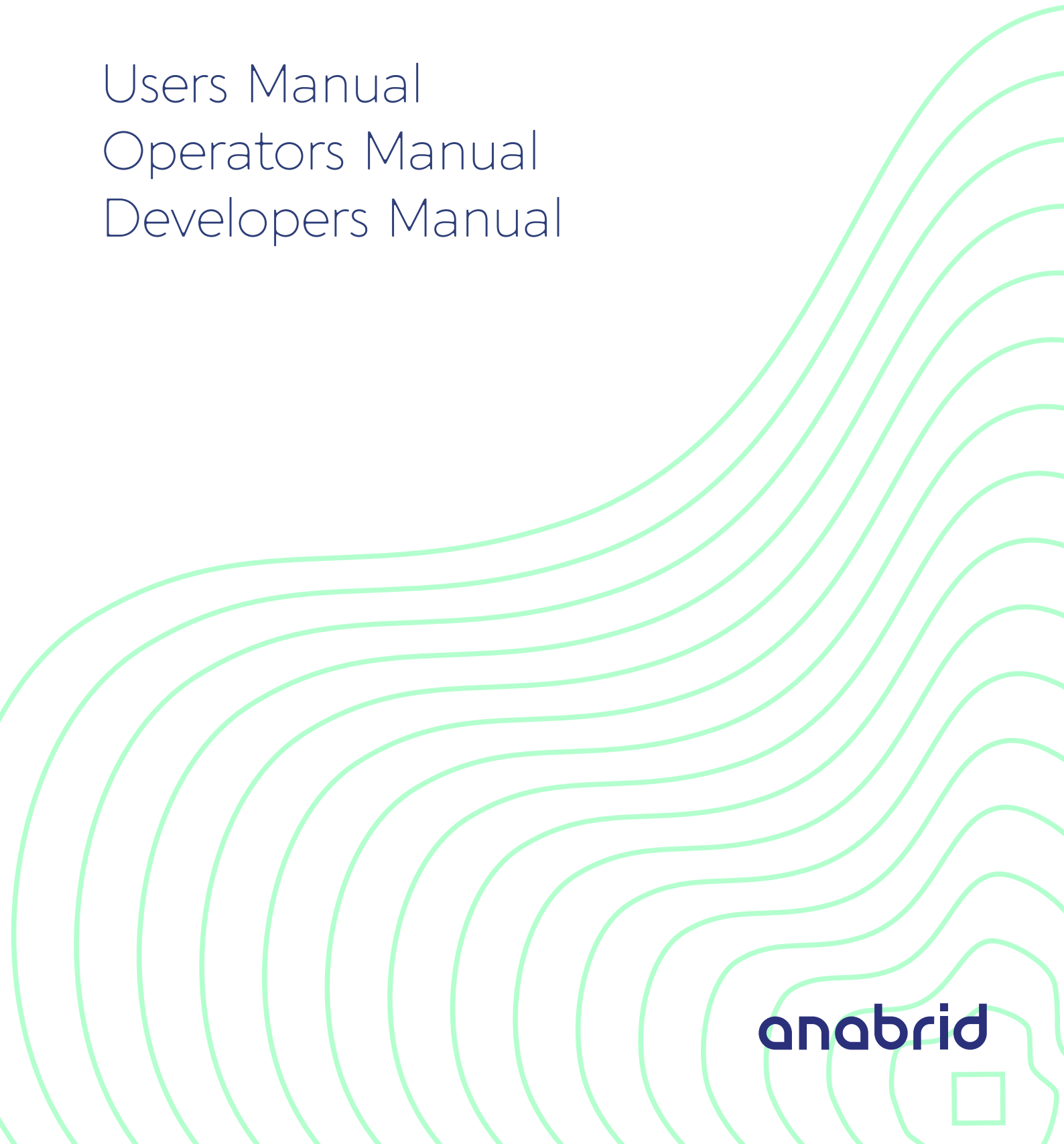
redac handbook

Users Manual
Operators Manual
Developers Manual



anabrid

USERS MANUAL

This is the official manual for the *Reconfigurable Discrete Analog Computer*, in short REDAC (https://redac.anabrid.com), made by anabrid GmbH (https://anabrid.com).

You can find a printable PDF version of this manual at https://redac.anabrid.com/manual.pdf and a browser friendly online version at https://anabrid.dev/docs/redac-manual/dirhtml/

Version of this document
v1.0-NN-ge07d716 (for details see section Section 18.1.1)

Build time
Apr 25, 2025

Authors
anabrid GmbH, Am Stadtpark 3, 12167 Berlin, Germany

How to read the manual: As a regular user, you only need the *User manual*. If you are a system administrator/operator, you should furthermore read the *Operators manual*. For software and hardware developers, the *Developers manual* is of most interest. All manual parts should be read in order, i.e. always consider the user manual first and the developer manual last.

Tip

You can read the latest version of this document online at
https://redac.anabrid.com/manual.pdf

Part I

Users manual

CHAPTER

ONE

PREFACE

Analog computers differ substantially from current digital computers in that they do not work by executing an algorithm in a step-by-step fashion. Instead they consist of a number of *computing elements*, each capable of performing a certain mathematical operation such as summation, multiplication or time-integration. A *program* for an analog computer describes how these computing elements are to be connected in order to create a *model* (an *analogue*) of the problem to be solved.

A (very) simple problem like computing $a(b+c)$ could be implemented on a classic digital computer as shown in this exemplary assembler listing:

```
LOAD  A, R0
LOAD  B, R1
LOAD  C, R2
ADD   R1, R2, R1
MULT  R0, R1, R0
STORE R0, ...
```

This straightforward algorithm requires six individual steps to compute the desired result. Contrast this with the analog computer program shown in figure Fig. 1.1. This setup requires just two computing elements, one summer and one multiplier. The summer is fed with the values b and c while the multiplier is connected to the output of this summer and to the value a.

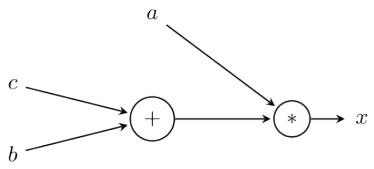


Fig. 1.1: Analog computer setup for solving $x = a(b+c)$

Typically, values are represented by voltages or currents in an analog computer so that only a single connection is necessary between computing elements.

The advantages of this analog computing approach are manyfold. Most notable are the extremely high degree of parallelism (there are no central memory, no data dependencies, no synchronisation points, etc., so that the computing elements work in full parallelism), the resulting high speed of computation and the inherent very high energy efficiency of analog computing.

While a digital computer can basically solve every problem given enough time and memory, an analog computer setup needs as many computing elements as there are operations in the governing equations of the problem to be solved. A little bit more mathematically, one can say that the size of a digital computer is constant while its time to solution typically grow much faster than just linearly

with the size of the problem (making many problems basically intractable on classic digital computers). The size of the analog computer on the other hand grows linearly with the problem size but, and this cannot be overestimated, the time to solution remains constant.

Classic analog computers were impressive systems, typically featuring a large patchpanel with thousands of jacks, all connected to a vast number of computing elements, such as integrators, summers, coefficient potentiometers, multipliers, etc. Programming these machines was as much of an art as a science and was quite time consuming due to the hundreds or even thousands of connections that had to be made manually. ([ulmannAP2] describes the history of analog computing in great detail.)

Nowadays, the patchpanel is a museum piece and the actual connection of the computing elements is done electronically, under the control of an attached digital computer; this greatly simplifies the programming. Using appropriate libraries as shown below, the analog computer can be used as a mathematical machine without having to understand the underlying electronic implementation in detail.

Since the basics of analog computer programming are outside the scope of this user guide, please refer to [*ulmannAP2*] for detailed information on the subject or consult the section Section 5.

CHAPTER

TWO

GETTING STARTED

REDAC is a fully software-reconfigurable analog-digital hybrid computer made by Anabrid (https://anabrid.com). The computer is intended to be used for datacenters, research and industry. It is specifically designed for solving differential equations, simulating complex models, and forecasting dynamic systems with exceptional speed and throughput—all while operating with minimal or no cooling requirements.

This system excels in a wide range of demanding applications, including realistic simulations of turbulent fluid dynamics, structural and flight dynamics, control systems, route optimization, big data analytics, real-time sensor processing, and much more.

REDAC is built on a groundbreaking analog dataflow architecture that delivers unparalleled performance, achieving at least 1,000 times the speed of traditional computing systems while consuming 10,000 times less power. It comes fully integrated with a comprehensive suite of software tools from Anabrid, including a powerful compiler, bindings for popular programming languages and libraries, and robust cloud system administration features.

REDAC is one of the first commercially available modern analog-digital hybrid computers and allows to solve up to about one thousand coupled differential equations (*DEQ*s) of 1^{st} order or five hundred DEQs 2^{nd} order, etc. Thanks to analog multipliers, non-linear equations can also be easily implemented.

The REDAC system contains a complex hierarchy of analog elements and digital processors (see *Architecture Reference*). Despite other modes of operation exist, this makes it very convenient to use the system as a remote user via the internet.

2.1 Requirements

REDAC is a modern analog-digital hybrid computer primarily meant to be used in a datacenter or "cloud" context. That means you will access it similar to cloud computing services from Google or Amazon, straight from your computer over the internet. This reduces the demand on your local computer to a bare minimum. In fact, this guide promotes our python reference client implementation which has little dependencies and will run on virtually any modern operating system.

Therefore, as a regular user of REDAC, this is the requirement list:

- A notebook or desktop computer with internet access or at least some sort of direct/company network access to REDAC. Most likely your operator provides you access to REDAC over conventional IPv4 TCP/IP networking.
- The operating systems Apple Mac OS X (macOS 10.9 Mavericks or later), Microsoft Windows (Windows 7 or later) or GNU/Linux are supported. Administrator rights are typically not required.

- At least Python 3.10 is required for the reference client software.
- As access is limited and precious, you first need to get a user account. Depending on the structure of your organization, you will be able to apply/register on a website or you have to manually contact the operators to have them creating a user account for you.

Please note that the REDAC's software may be operated in different states (see reconfigurable) which virtualizes the device such that multiple users may operate on _partitions_ of the device independently and in parallel. The mode is set by the operator and may be queried using the provided CLI tool (see below).

2.2 Installing the Python client

The REDAC programming interface can be used from a variety of programming languages. However, we first and foremost support the *python scientific programming ecosystem*, consisting of a standard set of tools such as Scipy (https://scipy.org/) and its components such as Numpy (https://numpy.org/).

In order to proceed with REDAC, you need the python client available at your computer. If you prefer, you can also choose some hosted Jupyter Notebook environment such as *Google Colab* or *Binder* (or the hosted one by Anabrid, see *List of provided services*). The python client very little dependencies. Virtually any computer running a modern python version should do it. You can run the client even on a Raspberry Pi if you prefer.

The software package anabrid-redac-client (https://pypi.org/project/anabrid-redac-client/) is released on public python package index (in short *pip*). In order to install it, you need the `python-pip` or `pip` command available on your computer. Installation is then as easy as typing

```
pip install anabrid-redac-client
```

from a regular python installation, i.e. a system shell with a working `python` and `pip` executable on the `PATH`. It is suggested to run this within a Virtual Environment (https://virtualenv.pypa.io/en/latest/user_guide.html). It is generally not suggested to install this system-wide with `sudo`.

A typical terminal session during installation looks like this:

```
you@yournotebook $ python -m venv venv
you@yournotebook $ source venv/bin/activate
(venv) you@yournotebook $ pip install anabrid-redac-client
Collecting anabrid-redac-client
  Downloading anabrid_redac_client-0.1.0-py3-none-any.whl.metadata (892 bytes)
[...]
Installing collected packages: urllib3, typing-extensions, termcolor, six,␣
↪shellingham, setuptools, python-dotenv, pygments, propcache, multidict, mdurl,␣
↪lucipy, idna, frozenlist, click, charset-normalizer, certifi, attrs, annotated-
↪types, aiohappyeyeballs, yaspin, yarl, requests, python-dateutil, pydantic-core,
↪ markdown-it-py, aiosignal, rich, pydantic, aiohttp, typer, anabrid-redac-core,␣
↪anabrid-redac-client
Successfully installed aiohappyeyeballs-2.4.6 aiohttp-3.11.12 aiosignal-1.3.2␣
↪anabrid-redac-client-0.1.0 anabrid-redac-core-0.1.0 annotated-types-0.7.0 attrs-
↪25.1.0 certifi-2025.1.31 charset-normalizer-3.4.1 click-8.1.8 frozenlist-1.5.0␣
↪idna-3.10 lucipy-1.6.0 markdown-it-py-3.0.0 mdurl-0.1.2 multidict-6.1.0␣
```

(continues on next page)

(continued from previous page)

```
↪propcache-0.2.1 pydantic-2.10.6 pydantic-core-2.27.2 pygments-2.19.1 python-
↪dateutil-2.9.0.post0 python-dotenv-1.0.1 requests-2.32.3 rich-13.9.4 setuptools-
↪75.8.0 shellingham-1.5.4 six-1.17.0 termcolor-2.3.0 typer-0.15.1 typing-
↪extensions-4.12.2 urllib3-2.3.0 yarl-1.18.3 yaspin-3.1.0

[notice] A new release of pip is available: 24.2 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
```

After successful installation, you should have the exectuable `redacli` available. Call `redacli --help` to test it:

```
(venv) you@yournotebook $ redacli --help

Usage: redacli [OPTIONS] COMMAND [ARGS]...

+---------------------------- Options -----------------------------+
| --install-completion   Install completion for the current shell |
| --show-completion      Show completion for the current shell,   |
|                        to copy it or customize the installation |
| --help                 Show this message and exit               |
+-----------------------------------------------------------------+

+---------------------------- Commands ----------------------------+
| run    submit    logs    status    results    partitions    health |
+-----------------------------------------------------------------+
```

If you system cannot find the executable, you most likely did not work within a virtual environment and do not have the python default binary path as part of your `PATH` variable. On UNIX-like systems, this goes typially into `~/.local/bin`, for instance you can type `export PATH="$PATH:$HOME/.local/bin"` to enable this location. You can also check the output of `python -m site --user-site` to find out where your python installation currently installs packages to.

2.3 Logging in

This first access happens in your webbrowser. Visit the website https://redac.anabrid.com and hit the *log in* button in the upper right. You should be then provided a page where you can register a user account or login with your useraccount. In order to connect to REDAC from the client, you currently have to store your *username* and *password* as environment variables. You can do this, for instance, by typing

```
you@yournotebook $ export REDAC_USERNAME="your-name@example.com"
you@yournotebook $ export REDAC_PASSWORD="yourHopefullyComplicatedPasswordGoesHere
↪"
```

When connecting, you also might need a hostname. This is, in the moment, just `https://redac.anabrid.com` and this option is already configured as the default and can be omitted.

> **Note**
>
> For further details about the registration procedure for the REDAC computer operated for DLR by

anabrid, contact your local system administrator or see *Anabrid Operations*. For technical information about the authentification procedure, see also *Authentification* in the operators manual.

2.4 Using the CLI

The REDAC may be operated in a *partitioned* setup where multiple parts of the system are isolated against each other through the software stack. For each of these partitions, there is a seperate *job queue* which contains jobs for this partition. Jobs in these queues may be submitted by diffeent users and each job is identified through an unique ID. Currently, there are two options to access the system:

1. Through the Python class `REDACClient` as demonstrated in *Example Applications*.
2. Through the CLI executable `redacli`.

This section covers the latter. Please first login to the system according to the instructions above. Given a netlist with filename `config.json`, one run of the netlist with default parameters on partition `PART` is submitted by

```
you@yournotebook $  redacli --host https://redac.anabrid.com --partition PART␣
↪config.json
```

Note that `https://redac.anabrid.com` is the current default and thus may be omitted. The client then continues to update the status until the job has been completed (with either the results or an error):

```
>>> redacli run test/data/config.json
◯ Job submitted with ID: 56bcc72f-aaa8-4c04-96f6-4eb6425372f6
◯ Job 56bcc72f-aaa8-4c04-96f6-4eb6425372f6 completed with status: COMPLETED
```

In this example, we submitted a job that is saved under its ID `56bcc72f-aaa8-4c04-96f6-4eb6425372f6`. With this ID, we can subsequently retrieve the results:

```
>>> redacli results 56bcc72f-aaa8-4c04-96f6-4eb6425372f6
{
  "/0/0": [(omitted)],
  "/1/0": [(omitted)]
}
```

Note that each item of the results contains data for one `adc_channel` from the netlist, by the scheme `/carrier/adc_channel`, where both carrier and `adc_channels` are indexed starting with `0`. Furthermore, we may download the job's log using the `logs` command:

```
>>> redacli logs 56bcc72f-aaa8-4c04-96f6-4eb6425372f6
[/forwarder/0/56bcc72f-aaa8-4c04-96f6-4eb6425372f6] 2025-02-16 20:51:39: Starting␣
↪task processing...
[/forwarder/0/56bcc72f-aaa8-4c04-96f6-4eb6425372f6] 2025-02-16 20:51:40: New run␣
↪state: JobSequenceEnum.RECEIVED_SET_CIRCUIT
[/forwarder/0/56bcc72f-aaa8-4c04-96f6-4eb6425372f6] 2025-02-16 20:51:40: New run␣
↪state: JobSequenceEnum.RECEIVED_STATUS_TAKE_OFF
[/forwarder/0/56bcc72f-aaa8-4c04-96f6-4eb6425372f6] 2025-02-16 20:51:40: New run␣
```

(continues on next page)

(continued from previous page)

```
↪state: JobSequenceEnum.RECEIVED_STATUS_DONE
[/forwarder/0/56bcc72f-aaa8-4c04-96f6-4eb6425372f6] 2025-02-16 20:51:40: Finished␣
↪processing (reason: TASK_FINISH)
```

These logs are fairly specific to the structure of the software stck, but may serve as the first indicator when dealing with issues in the software stack.

Last but not least, `redacli` enables users to receive information about the system, namely the *partitioning scheme*:

```
>>> redacli partitions
# Partition 0
        - Job queue length: 1
        - Carriers:
          - 00-00-00-00-00-00
# Partition 1
        - Job queue length: 2
        - Carriers:
          - 00-00-00-00-00-01
```

showing the carriers (mREDACs) per partition and the number of jobs waiting for processing in each queue as well as the temperatures in the system:

```
>>> redacli health
```

which prints a list of individual components in the system as well as their temperatures.

CHAPTER

THREE

ARCHITECTURE INTRODUCTION

Classic analog computers typically featured a large central patch panel consisting of thousands of sockets by means of which computing elements were connected with each other using patch cables. This was very cumbersome, took a long time to manually program, was error prone, and did not allow for rapid program changes. Fortunately with the REDAC system these patch panels are finally relegated to museums where they belong.

This section provides a brief overview about the architecture of the Reconfigurable Discrete Analog Computer, in short REDAC. For a more extensive documentation, see *Architecture Reference* in the *developers manual*.

3.1 REDAC internals

REDAC is a complex system consisting of a plethora of inner, "nested" subsystems. At the lowest levels, these systems are either digital or analog, rendering the overall system as a hybrid computer. The vast majority of electronics in REDAC is analog compute circuitery and analog interconnection networks, the digital parts are in fact only a tiny fraction of administrative control structures, realized in heterogenous computer architectures (both embedded, digital switching network and server grade processors).

REDAC follows the black box approach of operational amplifier based computing. This means REDAC is a giant replacement for an arithmetic-logical unit (ALU) of a traditional digital processor. The analog computing elements of REDAC can carry out basic mathematical operations such as addition and multiplication, but they can carry out also advanced operations such as integration in time. They do so at high accuracy and completely continous in time and with continous values. This is an important property of the REDAC compute elements but also its interconnection network, in short *CTCV* (continuous time, continuous variable). For the analog network, this means that the connections between the analog computing elements within a REDAC system are not based on switched capacitors but instead on static (reconfigurable) switching matrices. Although a switched capacitor network could have simplified the actual hardware implementation, it would immediately invalidate the low energy footprint of the analogy compute paradigm.

3.2 The system matrix

When using an analog computer such as REDAC, one maps a mathematical set of equations onto an electronical circuit, also refered to as compute graph. When drawing the adjacency matrix of this graph, one can think of REDAC as a large interconnection matrix of size N^2 with N compute elements at the edges, as illustrated in figure Fig. 3.1.

This illustration should be understood as following: At the left side, there are computing elements which feed their output to a column of a matrix (this is called a *fan-out*). Each matrix element is real-

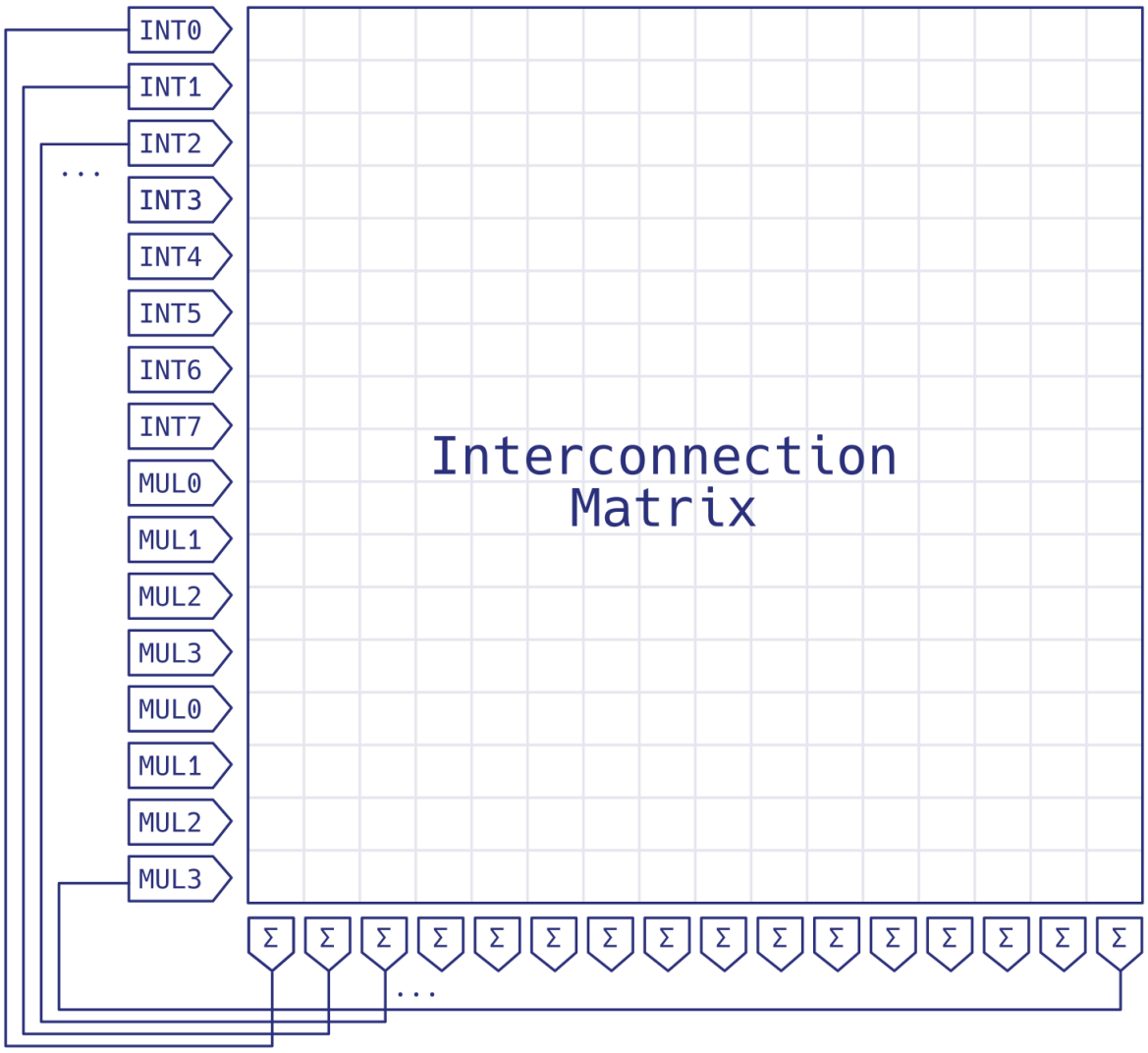


Fig. 3.1: Interconnection matrix with implicit summing capabilities (Note: in fact this figure shows a *Single cluster*).

valued and constant during analog computation time (this is conceptually called a *potentiometer*). One can imagine that each cell in the row subsequently holds the incoming compute result multiplied with the matrix value at that place. The figure shows a special kind of matrix which has the property of building the column sum, i.e. there is a *fan-in* happening which reduces the values by building a sum. There is then a 1:1 correspondence of columns and rows, i.e. in the simplest case row i feeds back to column i (monadic compute element) whereas the summing reduction is a bit simplistic for dyadic compute elements such as a multiplier which has (in theory at least, in REDAC exactly) two inputs which are supposed to be multiplied together.

This way, arbitrary connections between compute elements are possible (also refered to as *all-to-all* connectivity). In practice, however, REDAC allows only for *sparse* matrices, i.e. does not allow arbitrary connections between computing elements but only a tiny subset, with a sparsity beyond 90%. It is the job of the software to determine possible connections in order to make your mathematical problem fit on the computer.

3.3 Available compuing elements

In its final expansion stage, REDAC will have at the ballpark order of 10^4 computing elements of each kind, i.e. time integrators or multipliers. As already indicated above, addition is done “for free” by the interconnection scheme and thus available in much wider terms. In the same spirit, REDAC has at the order of at least 10^5 digital potentiometers (“weighted edges” in the interconnection graph) and many more non-weighted connections.

The computer will have about 10^{3-4} analog-to-digital converters that can be routed to different computing elements, making flexible read-outs and debuggings of complex calculations possible.

The computer can be partitioned (time-sharing) by means which follow its hierarchical structure. Explaining this structure is beyond the scope of this short introduction. Instead, see section Section 14 to read more about the computer architecture.

CHAPTER

FOUR

EXAMPLE APPLICATIONS

In the current software revision of REDAC, users are supposed to be given circuit configuration files (a concept also known as netlist (https://en.wikipedia.org/wiki/Netlist)) which are refered to as `config.json` in the following. These circuit configuration files encode the actual mathematical problems. Their generation is done by software by tooling which will provided at a later stage.

The usage of REDAC is currently therefore basically the submission of a `config.json` file, next to some basic information about the simulation parameters (such as simulation runtimes or for data querying of measurement variables).

Given the Python client (section *Getting started*), you can invoke examples either from your operating system command line or from python code, using the given client as a library.

For the following example, please download the `exemplaric config.json file` first.

In the print version of this document, a compact version of the config file is printed for illustrative purpose. Please do not try to type in this file at your computer but instead head to the online version of this document and download the ASCII file for straight usage.

```
1 {"00-00-00-00-00-00":{"/0":{"/C":{"elements":[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
  ↪0.0,0.0,0.0,0.0,0.0,-1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
  ↪0,0.0,0.0,0.0]},"/I":{"outputs":[[15],[],[],[],[],[],[],[],[],[],[],[],[],[],[],
  ↪[]],"upscaling":[false,false,false,false,false,false,false,false,false,false,
  ↪false,false,false,false,false,false,false,false,false,false,false,false,false,
  ↪false,false,false,false,false,false,false,false,false]},"/M0":{"elements":[{"ic
  ↪":-0.82,"k":10000},{"ic":0.0,"k":10000},{"ic":0.0,"k":10000},{"ic":0.0,"k
  ↪":10000},{"ic":0.0,"k":10000},{"ic":0.0,"k":10000},{"ic":0.0,"k":10000},{"ic":0.
  ↪0,"k":10000}]},"/M1":{},"/U":{"outputs":[null,null,null,null,null,null,null,
  ↪null,null,null,null,null,null,null,null,0,null,null,null,null,null,null,null,null,
  ↪null,null,null,null,null,null,null,null,null]}},"/T":{"muxes":[null,null,null,
  ↪null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,
  ↪null,null,null,null,null,1,null,null,null,null,0,null,null,null,null,null,null,
  ↪null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,
  ↪null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,
  ↪null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,
  ↪null,null,null,null,null,null,null,null,null,null,null,null,null]},"adc_channels
  ↪":[0]},"00-00-00-00-00-02":{"/0":{"/C":{"elements":[0.0,0.0,0.0,0.0,0.0,0.0,0.0,
  ↪0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
  ↪0.0,0.0,0.0,0.0,0.0]},"/I":{"outputs":[[10],[],[],[],[],[],[],[],[],[],[],[],[],
  ↪[],[],[]],"upscaling":[false,false,false,false,false,false,false,false,false,
  ↪false,false,false,false,false,false,false,false,false,false,false,false,false,
  ↪false,false,false,false,false,false,false,false,false,false]},"/M0":{"elements
```

(continues on next page)

(continued from previous page)

```
↪":[{"ic":0.0,"k":10000},{"ic":0.0,"k":10000},{"ic":0.0,"k":10000},{"ic":0.0,"k
↪":10000},{"ic":0.0,"k":10000},{"ic":0.0,"k":10000},{"ic":0.0,"k":10000},{"ic":0.
↪0,"k":10000}]},"/M1":{},"/U":{"outputs":[null,null,null,null,null,null,null,
↪null,null,null,null,0,null,null,null,null,null,null,null,null,null,null,null,
↪null,null,null,null,null,null,null,null,null]}},"/T":{"muxes":[null,null,null,
↪null,null,null,null,null,null,0,null,null,1,null,null,null,null,null,null,null,
↪null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,
↪null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,
↪null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,
↪null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,
↪null,null,null,null,null,null,null,null,null,null,null]},"adc_channels
↪":[0]}}
```

We primarily support the Python programming language as the reference implementation. The way how to upload a configuration file is like

```
you@localhost> redacli run --host https://redac.anabrid.com [--partition␣
↪partition_id] config.json
[CSV output of measurement variables]
```

or, straight from the python programming language (REPL):

```
you@localhost> python
>>> import json
>>> from anabrid.redaccess.api.client.redac_client import REDACClient
>>> redac = REDACClient("https://redac.anabrid.com")
>>> redac.login(username, password) # to be set by the user
>>> with open("config.json", "r") as f:
>>>     config = json.load(f)
>>> job_id, results = redac.solve(config, 0)
>>> logs = redac.log(job_id)
```

Afterwards, the variables *results* resp. *logs* will hold the job results in a JSON format and the job's log as a string array.

Note that other than these high-level functions, *REDACClient* contains several functions that allow to submit one or multiple job asynchronously, to multiple partitions, then wait for them to finish and collect their results all at once:

```
import asyncio
import json
from anabrid.redaccess.api.client.redac_client import REDACClient

async def submit_batch():
  redac = REDACClient("https://redac.anabrid.com")
  redac.login(username, password)
  with open("config.json", "r") as f:
    config = json.load(f)

  # asynchronously submit three jobs
```

(continues on next page)

(continued from previous page)

```
  jobs = [None, None, None]
  for part_id in range(3):
    job = {
      "config": config,
      "partition_id": part_id,
      "label": "Job"
    }
    jobs[part_id] = asyncio.create_task(redac.submit_job(job))

  # wait until the three jobs were successfully submitted (and returned their ID)
  for part_id in range(3):
    while not jobs[part_id].done():
      asyncio.sleep(0.5)

  # not wait until the jobs are done (in the meantime, you can do some other work)
  for part_id in range(3):
    job_id = jobs[part_id].result()
    is_done = False

    while not is_done:
      is_done = (await redac.get_job_status(job_id).status == "COMPLETED")
      asyncio.sleep(0.5)

  # finally, download and print results
  for part_id in range(3):
    job_id = jobs[part_id].result()
    results = await redac.get_job_results(job_id)
    print(json.dumps(results))

if __name__ == "__main__":
  asyncio.run(submit_batch())
```

Save this script as `batch.py`, the execute it with `python batch.py`. If you are not familiar with the `async` syntax used here, please refer to the official Python developer's documentation.

CHAPTER

FIVE

FURTHER READING

You reached the end of the user-relevant manual. There is more to read, thought: If you are interested to take a deep dive into REDAC programming, you can read the *Developer manual*. If you want to become an administrator of the system, consider reading the *Operators manual* next.

In order to learn more about analog computing, we recommend the books and publications authored by anabrid and the scientific community. In particular, you may be interested to follow these weblinks for further recommendations:

- Videos and podcasts on analog computing (https://anabrid.com/videos-podcasts/)
- Research articles and primers on analog computing (https://anabrid.com/articles-books/)

CHAPTER

SIX

ANABRID OPERATIONS

redac
Anabrid Systems Operations

This section is dedicated to the *REDAC1 Systems Operations Team* at *anabrid GmbH*. The term *REDAC1* refers to the REDAC system with serial number 1, which is build for the customer DLR-QCI (https://qci.dlr.de/). During 2025, anabrid will operate the system on a transitional basis for and in the interests of the customer. In particular, software solutions that are *not* part of REDAC are provided and operated within this transitional time. These are listed in this section and their use is documented. There is no documentation regarding administration, as administration is carried out by the Anabrid Systems Operations Team.

Within this transitional time, the system operated by anabrid is globally reachable via the internet at https://redac.anabrid.com

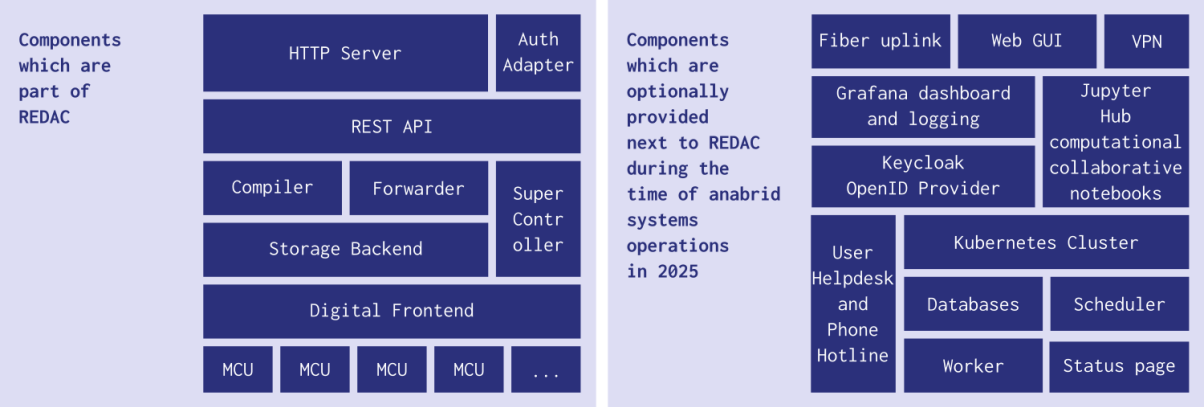


Fig. 6.1: Overview about the REDAC-internal services vs. the services operated by anabrid.

6.1 List of provided services

The following services are not part of REDAC but part of the operations teams, next to the actual REDAC hardware and software maintenance. Similar systems are supposed to be hosted by a REDAC customer. See also figure *Overview about the REDAC-internal services vs. the services operated by anabrid.* for an overview.

Systems Access to `redac.anabrid.com`

General system access is provided over anabrid paid 1gbit/sec symmetric internet fiber up-

link which has a guaranteed uptime of certainly less then 99% (i.e. 14min downtime per day (https://uptime.is/99)).

Systems website landing page "User Interface" https://redac.anabrid.com/ui/
This is a little static website (Sveltekit (https://svelte.dev/docs/kit/introduction)) which is managed, amongst other static assets, via SFTP. The website itself, in particular, is a *goodie* and not part of the official *software infrastructure*

Status Page https://status.anabrid.net/status/redac1
Anabrid operates a number of status services which allows to have a quick overview about the availability and reliability of the services. The reachability of the REDAC1 System is monitored on this website.

Grafana dashboard https://stats.redac.anabrid.com/
For demonstrating/displaying temperature readout. Runs as a docker container on the SuperController.

JupyterHub instance https://jupyter.redac.anabrid.com/
This is a Kubernetes cluster operated by the OPs team which has direct access to REDAC internal network. However, it is physically hosted and served not on SuperController but dedicated bare metal digital hardware.

Authentification https://auth.redac.anabrid.com/
Anabrid transitionally operates the user authentification system for REDAC1. The underlying software system *Keycloak* is an official suggestion to be used also by the customer within the REDAC software infrastructure, cf. *Authentification*.

User Helpdesk https://helpdesk.anabrid.com/
Anabrid runs a ticketing system in order to provide help for REDAC1 users. This is a company wide ticketing system where anabrid employees help on improving the customers jorney. Users can open a ticket straight on the helpdesk or write a mail to helpdesk@anabrid.com. Please insert the term `REDAC1` in the subject message of your tickets. Clearly, this ticketing system is not part of REDAC infrastructure.

Anabrid User Hotline `0049 30 629 3047 20`
Anabrid runs a telephone line which allows people to ask for help within regular office hours. There is no guarantee that support can be provided via telephone. However, the onsite team in Ulm is always happy to receive inqueries at a direct visit.

6.2 User account creation

Self-Registration is enabled for all user accounts ending on `@dlr.de`. Otherwise, user accounts have to be manually activated by the operations staff. Primary contact happens via the anabrid helpdesk.

> **Note**
>
> REDAC has a standalone user account system (see *Authentification*) which is by no means related to any institutional user managament systems by DLR or anabrid.

6.3 Profile user data policy

Users who want to use the system up have to agree to the european General Data Protection Regulation (GDPR). The following minimum amount of data is collected:

Technical Data

- User IP address
- technical browser/client identification information (for instance software versions)
- Queries sent to the system and replies got

Retention period: Logging and Performance evaluation, data is intentionally preserved only for a fixed period of time and never leaves the system (i.e. the *Super Controller*).

Organizational Data

- Personal identification features (real name, business address, institutional affiliation, telephone number, e-mail address)
- System Username
- System Password (only stored in encrypted form according to common cryptographic industry standards)
- A list of authorised security tokens such as a SSH public key or a JWT access token.
- Given consent obtained for personal data processing in accordance with the GDPR (General Data Protection Regulation, legislation in Germany)

Retention period: Data are preserved for the full period of operations, even after the user intentionally deleted his account. This is neccessary for project reporting and evaluation.

Note

When using any services provided by anabrid, the general privacy policy by anabrid applies. It can be found at https://anabrid.com/privacy-policy. Furthermore, the general terms and conditions by anabrid apply. They can be found at https://anabrid.com/agb

Part II

Operators manual

CHAPTER

SEVEN

INTRODUCTION

Welcome to the Administrators and Operators handbook of REDAC, the Reconfigurable Discrete Analog Computer. Please ensure you have read the *User manual* before approaching this part of the system documentation.

Generally, REDAC is a professional high precision computer system intended for industrial settings. The system is of low environmental demand but requires nevertheless some considerations for the *Physical Setup*. Furthermore, the system generally requires *low to no regular maintenance*. Depending on your overall computer network, a substantial initial setup phase might be neccessary. The relevant steps are described, amongst others, in the section *Internal network*.

7.1 Intended Audience

As a systems administrator or operator, you have a decent understanding in common IT networking and Linux server administration. Before reading this, you should know at least how to configure at small and home office (SOHO) level networking equipment.

Warning

You are supposed to read this manual before doing any kind of physical work close to or with REDAC. It contains essential guides how to deal with this kind of novel computing equipment.

7.2 Confidential Credentials given at system handover

At system purchase, you are prodived an access credential card. This card contains sensitive information which you must not share with regular users nor system developers. It contains strong passwords individually assigned to your system. As the system administrator, you are supposed to take over these access credentials and manage them. Consider using a password manager to track them. Furthermore, you are strongly encouraged to change the default passwords. This card should look roughly like the following table.

Table 7.1: REDAC Credentials Overview

Subsystem	Scope	User	Password
Mikrotik	Web GUI	*admin*	*provided*
Mikrotik	Wifi	SSID *provided*	*provided*
Super Controller	SSH/X11 login	*provided*	*provided*
Super Controller	IPMI	*povided*	*provided*
Keycloak	OpenID managament	*provided*	*provided*

7.3 Public Information on your REDAC installation

Next to the confidential system access information, you will get a card that contains the following information which help to identify the device(s) you have at hand. They are not secret and can be shared with users and developers, if neccessary:

- *MAC addresses* of the *Mikrotik* uplink ethernet port and Wifi access point
- *Serial number(s)* of the overall redac device and/or its individual parts

These public data should also be attached on a label on the device itself.

CHAPTER

EIGHT

PHYSICAL SETUP

This section provides an overview about physical handling of the system. These information are relevant for on-site operators in particular at preparing the installation site, mounting, dismounting and moving the computer. For an overview, consider the following short specs:

Dimensions and Weight (Upper limits for one rack)
2 x 3 x 2 m, weight: 800 kg

Typical Dimension of one Rack
HBT 1800 x 800 x 800 mm (37 HE)

Form factors for outer structures
DS8837 (https://www.netzwerkschrank24.de/37-he-19-zoll-netzwerkschrank-mit-glastur-800-x-800-x-1800-mm-bxtxh.html) server rack (metal and glass) and EuropacPRO Schroff (https://may-static.de/44966.pdf?1583829815925) chassis (aluminium)

Max Operating Pressure
1030 hPa

Max Environmental Operating Temperature
60° Celsius

8.1 Spatial requirements

When setting up REDAC for the first time or evaluating potential locations for its installation, it is essential to carefully consider the following aspects to ensure optimal functionality and ease of use.

Suitable Rooms
REDAC is highly versatile in terms of placement and does not impose strict requirements on the type of room. It can be installed in a standard office space with a carpeted floor, or even in a small, poorly ventilated storage room. The system's design ensures reliable operation in such varied environments. However, for long-term usability and maintenance comfort, a clean and relatively dust-free area is recommended to minimize cleaning needs and protect components from excessive wear due to environmental factors.

Temperature and Ventilation Requirements
REDAC does not depend on specialized temperature control or ventilation systems. All analog components within REDAC are passively cooled, ensuring silent and reliable operation without the need for fans or external cooling mechanisms. The active cooling requirements are limited to the digital components, such as the routing switch and server, which use standard built-in cooling systems. These components are designed to operate effectively in typical office or lab conditions, with ambient temperatures ranging from 10°C to 60°C.

Physical Accessibility
To facilitate setup and maintenance, it is recommended to leave approximately 50 cm of clearance in each direction around the REDAC rack. This space allows technicians to access all necessary components without difficulty during servicing. While this is not strictly required for daily operations, it can significantly simplify routine maintenance and troubleshooting procedures.

Power
REDAC requires a single-phase 120V/240V AC mains connection secured with an ordinary 16A fuse. These conditions are available in virtually any data center, working office or laboratory. For details, see *Powering the system*.

Network
REDAC requires a standard Ethernet connection with 10/100/1000 BASE-T and an RJ45 interface. This type of network connectivity is widely available and cost-effective. A single network cable is sufficient for typical operations, and a redundant uplink is not necessary unless advanced failover or high-availability configurations are desired. For most scenarios, basic network access is adequate to meet operational requirements.

8.2 Safety Instructions for Physical Handling of the Device

When physically handling the REDAC system, it is crucial to follow these safety instructions to ensure both personal safety and the integrity of the equipment:

Always Wear an ESD Wrist Strap When Opening the Rack from Behind
The REDAC rack frame is fully grounded and equipped with several pre-installed ESD wrist straps connected directly to the frame. Always ensure you are wearing one of these straps before coming into contact with any electrical components, especially when accessing the device from the rear side. This precaution is essential for protecting the sensitive electronics within REDAC from electrostatic discharge, which can cause irreversible damage to critical components. Make sure the wrist strap is properly secured and functioning before beginning any work.

Do Not Modify the Hardware Setup
Under no circumstances should the hardware configuration of REDAC be altered without proper authorization. Opening or unmounting the iREDAC chassis, or making unauthorized modifications, will void the warranty and liability coverage of the entire system. Hardware changes are strictly reserved for trained and certified personnel with specific expertise in handling the system. It is important to note that most system operators are not certified for such tasks. Unauthorized alterations can compromise the functionality and safety of the device.

8.3 On-Device Indentifier Tags

The REDAC rack has the following on-device type plates for device identification:

- Each iREDAC has a unique serial number (a short integer as well as a UUID which is displayed as QR code). The iREDAC labels are attached on the backplane.
- The REDAC rack has a bigger type plate at the back side. There is also a label which shows the EAN number of the device as a 1D barcode, suitable for scanning while inventarization.

REDAC has a EAN-13 (European Article Number) which is a GTIN (Global Trade Item Number) which is a UPC (Universal Product Code). This number is `1-700001-948842`. This number refers to a iREDAC, which is at a rack level scale the building block of REDAC. The iREDACs also have a CE certification which is indicated by the appropriate CE symbol.

For legal reasons, the type plates must not be removed or artificially covered.

8.4 Moving

When relocating or moving the REDAC rack, it is important to carefully consider the following points to ensure both the safety of the equipment and the personnel involved.

Move Avoiding Vibrations
: Although the REDAC rack is equipped with wheels, their use is not recommended for covering longer distances. Wheels are suitable for minor adjustments in position but may expose the rack to excessive vibrations during extended movement. Instead, it is advisable to use equipment specifically designed for smooth and vibration-reduced transport, such as a pallet truck or a similar lifting device. This approach minimizes the risk of damage to the internal components, ensuring the longevity and reliability of the system.

At Least Two People
: A minimum of two people is required at all times when moving the REDAC rack. This precaution helps to maintain control and balance of the large and heavy structure, significantly reducing the risk of the rack tipping over. Coordination between the individuals involved is essential to ensure safe handling, particularly when navigating tight spaces or uneven surfaces.

Make Use of the Installation Supports
: Before moving the rack, ensure that the installation supports are securely unmounted to allow free movement. Once the relocation is complete, re-mount the installation supports to stabilize the rack and relieve stress on the wheels. These supports are critical for fixing the rack's position and maintaining its stability during operation. Always keep the appropriate wrench readily available to lock or unlock the installation supports as needed.

Lock the Door Before Moving
: To prevent the rack door from opening unintentionally during movement, ensure that the door is securely locked before starting the relocation process. This simple step protects the contents of the rack and avoids potential damage or injury caused by the door swinging open while in transit.

8.5 Device Disposal

Proper disposal and recycling of the REDAC system are essential to ensure compliance with environmental regulations and to minimize the impact of electronic waste. For this purpose, please contact the manufacturers for guidance on the appropriate steps to take when disposing of the system.

Electronic waste disposal is a free service offered by Anabrid GmbH, the manufacturer of REDAC. They provide expert assistance to ensure that all components of the system are handled responsibly, including recycling of electronic parts and disposal of hazardous materials in accordance with applicable laws and environmental standards.

When preparing the system for disposal, take the following steps:

1. *Contact Anabrid GmbH*: Reach out to our customer service or designated support channels to inform them about the disposal and to arrange for the necessary logistics. They may provide you with shipping instructions or coordinate a collection service.
2. *Decommission the System Safely*: Disconnect the system from power and network connections. Anabrid GmbH will take care of packing and relocating for transporation. Be sure to include all components, such as power supplies and networking hardware, if requested.
4. *Follow Local Regulations*: If you are unable to use Anabrid GmbH's disposal service, ensure that the device is recycled through a certified electronic waste handler in compliance with local laws and regulations.

By utilizing Anabrid GmbH's free disposal service, you contribute to a sustainable lifecycle for electronic devices, ensuring that valuable materials are recovered and hazardous waste is properly managed. For further details, consult the documentation provided by Anabrid GmbH or visit their our website.

CHAPTER

NINE

POWERING THE SYSTEM

As noted earlier, the REDAC system imposes minimal requirements on its power supply. Any single-phase safety plug operating on 120V/240V AC mains with a maximum current of 16A is sufficient for proper operation.

The REDAC-specific (mostly analog) components are powered with high quality 24V DC power supplies with a maximum rating of 300W each.

Important

A correctly grounded mains connection is crucial for safe and reliable operation. Most data centers, working offices, or professional laboratories already provide suitable power infrastructure. Ensure that the grounding is verified to avoid potential safety risks and to guarantee system stability.

While the use of a data center power delivery unit (PDU) or an uninterruptible power supply (UPS) is not mandatory, these devices can offer significant benefits. A UPS, particularly one with a battery-backed design, provides additional protection against power surges, outages, and voltage fluctuations. Moreover, a UPS can stabilize the input voltage, which may enhance the performance of the internal power supply. This stability can, in turn, improve the accuracy and reliability of REDAC's computing operations, making it a worthwhile consideration for critical applications.

9.1 System Startup

REDAC features a 19-inch switchable power socket located at the rear of the rack (see Fig. 9.1). This design allows the entire system to be powered on with a single flip of the master switch, provided the mains connection has been properly plugged in and secured.

Once power is supplied to the system, the following sequence typically occurs automatically (ordered by their time to readiness):

1. The Ethernet networking equipment (router or switch) powers on immediately, establishing the network connections required for system communication.
3. The analog power supply activates, which subsequently powers all connected analog modules and prepares them for operation. This also means that all Microcontrollers power up and wait for an IP address with DHCP calls until the routing switch assigns them an IP address.
2. The server, also known as the supercontroller, powers on, waits for an IP address and starts up all relevant services in correct order. They then begin to initialize their control and managament processes.

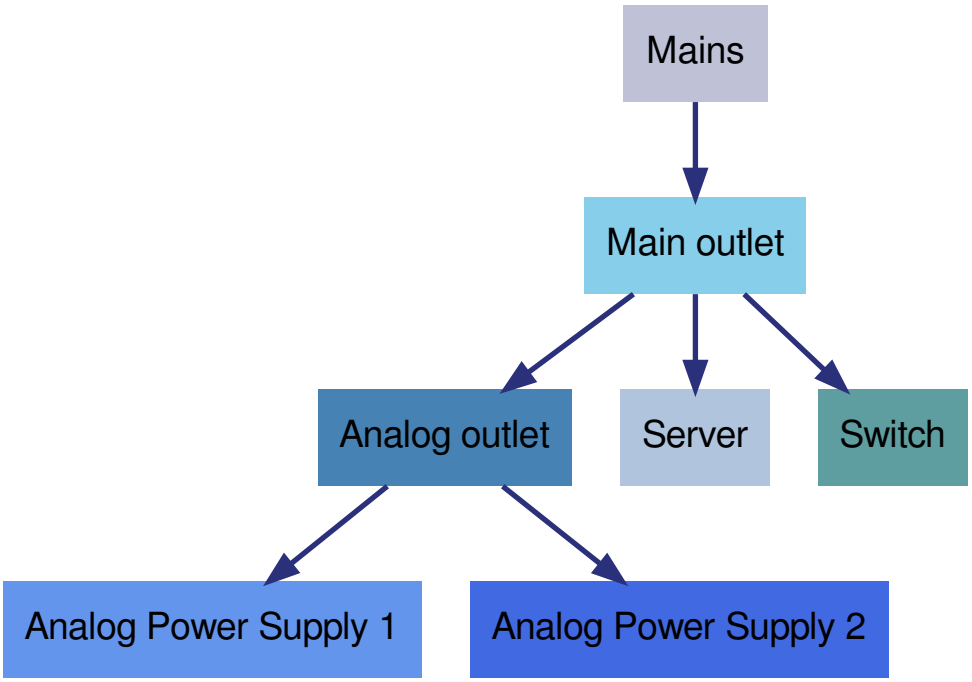


Fig. 9.1: 240V power distribution in REDAC

Although each of these components can be powered on or off individually if necessary, this is rarely required under normal operating conditions. The automated startup process ensures that the system reaches an operational state efficiently and with minimal user intervention. However, users should verify that the system initializes correctly and observe for any unexpected behavior during startup, especially after relocation or maintenance.

Note

The REDAC system operated by anabrid (see *Anabrid Operations*) can be monitored at https://status.anabrid.net/status/redac1 in order to see when all systems are up and running after powerup.

9.2 System Shutdown

When shutting down the REDAC system, follow these guidelines to ensure a safe and proper shutdown process:

1. The Ethernet networking equipment (router or switch) can be powered off without any adverse effects on the system. Since the networking components do not store critical data, there is no risk of data loss when they are turned off.
2. The server (supercontroller) requires a manual shutdown sequence to prevent data loss. This is typically done by executing the command `sudo shutdown -h now` at the Linux command line. This ensures that all running processes are terminated gracefully, and any data in memory is safely written to storage before the system powers down.
3. The analog components of the system can be powered off without any significant effects. These parts do not rely on active processes or stored data and can be safely disconnected from power as needed.

Note

While a sudden power loss is generally not critical for the REDAC system, it is worth noting that the server is equipped with a journaling file system. This feature ensures data integrity and facilitates recovery mechanisms even in the event of an abrupt shutdown. In such cases, the only potential loss would involve volatile database entries, which are typically of minor importance in practical applications. Despite this resilience, a controlled shutdown process is always recommended to avoid unnecessary recovery procedures and ensure smooth system operation.

CHAPTER

TEN

INTERNAL NETWORK

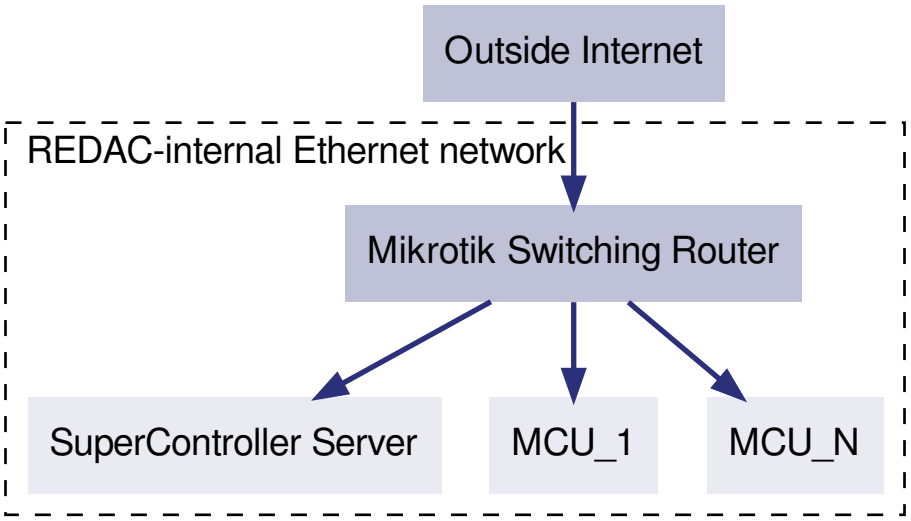


Fig. 10.1: Graph of the logical (internal) Ethernet of REDAC. See main text for explanation.

REDAC features an internal IPv4 network somewhat similar to a typical home subscriber network (Fig. 10.1). This network has the following physical components:

Mikrotik Switching Router
: Off-the-shelf rack mount router including a 24port front facing ethernet switch. The first port of the switch is dedicated to the external uplink while all other ports are bridged to an internal network. The device is introduced in section *The Mikrotik Router*.

Super Controller (x86 Server)
: An off-the-shelf low power low noise low depth server with IPMI out of band remote management facilities. The device can also be used as a terminal server interactively. For further details see section *The Super Controller Server*.

Hybrid Controllers (Teensy Microcontrollers)
: The network contains dozens of microcontroller units (MCU), each of them connected with 100Base-TX to the switch and centrally managed by the super controller. These devices provide/require no direct network access by system operators.

This section will present the particular components in detail and explain relevant configuration steps.

Note

At the time of reading this, you should have the *Confidential Credentials given at system handover* next to the *Public Information on your REDAC installation* ready at your hand. Note that the authentification systems of REDAC internal network structure are scoped and not unified/synchronized. That means that each device has a local user and password configuration and no single login exists. Do not confuse the scope of the credentials. credentials listed with the REDAC *public* application level authentification, which are managed by OpenID and discussed in the section *Authentification*.

10.1 Initial network setup/access

By default, the router behaves as a IPv4 DHCP client at the external interface. If you have a DHCP server running in the outer network at your control, this default configuration might be suitable for you. In this case, you can access the super controller server by means of the TCP ports exposed (by NAT in the router). You can also take the server as an intermediate to configure the router. Note that the router configuration interface (web, ssh) is only accessible from the internal network.

Tip

By default, the REDAC Router behaves similiar to a IPv4 home subscriber setup (think of a Fritzbox in Germany): There is a restrictive firewall allowing no ingoing traffic, doing masquerading and exposing thus only a single IP address at the outer network. System operators are encouraged to enter the internal network with their equipment, in particular for setup and maintenance.

If you cannot make use of REDAC acting as a DHCP client but need to have it a given and fixed address, you need to *get access to the internal network*, which requires physical access. There are two principal ways how to do that:

- Connect *your* notebook/computer to the internal network or
- Connect a monitor and keyboard/mouse to the super controller and make use of it as a standalone computer.

Note

The neccessary ports (USB and VGA) for connecting a terminal to the server are located at the front of the rack. If there are no free USB slots, you can plug out some USB devices while not using the Analog part of REDAC. Make sure to connect all USB devices back after having finished the administrative work, or use a USB hub if neccessary.

For connecting your own computer to the internal REDAC network, there exist two options:

- Either you connect the Wifi hotspot opened by the router. Keep the SSID and WPS2 password credentials at hand in order to connect.
- Or you connect physically with a network cable to one of the free ports on the switch.

In both ways, your computer should be assigned an address by the REDAC internal DHCP server. If you cannot run a DHCP client on your computer, you may choose a fixed address freely in the range `192.168.104.5 - 192.168.104.9` with subnet mask `255.255.255.0` and gateway `192.168.104.1`. The gateway acts also as DNS server.

If you, instead, choose to make use of the super controller as an interactive computer, you must have the username and password credentials at hand to login at the desktop greeter. After successful login, the linux computer will launch a desktop environment which allows you to start the usual applications such as a web browser or a terminal. If you do not have a mouse available, hit `CTRL + ALT + F2` or `F3` in order to switch the virtual desktop to a TTY instance which allows you to use the linux shell without a graphical user interface.

Warning

Keep in mind that the direct access to the internal REDAC network is **only** intended for system administrators and not regular users. The reason is that ordinary users should not communicate directly with the microcontrollers. Otherwise, the Super Controller is bypassed and is prevented from doing his work correctly, messing up proper system managament.

The same is true for the Super Controller graphical interface! If you want to provide your users a console like desktop access in the same room, for instance, please setup your own computers which are put *outside* of REDAC.

10.2 The Mikrotik Router

The Mikrotik router runs the proprietary operating system *RouterOS* by Mikrotik. It is rich in features and the supplier provides a detailed documentation available at https://help.mikrotik.com/docs/. If you are not familiar with this technology, please read the section *Getting started* in the linked Mikrotik documentation.

The REDAC ships with the mikrotik model type `CRS326-24G-2S+RM`. It features a managed 24 port gigabit switch next to the routing capabilities.

10.2.1 Relevant settings

The router comes with REDAC-specific default settings (given below) which only use a tiny fraction of all functions. Virtually any professional grade router available on the market, such as Cisco, Linksys, Juniper and other can do the same. Note that the MCUs only have a fast ethernet uplink (100mbit) and thus modern multi-gigabit powerhorses are not neccessary. The Mikrotik is therefore a sufficient and energy saving choice.

The following router configuration options must not be changed by the systems operator:

Internal ethernet configuration
: All ports are bridged except port 1, which belongs to the external port. The wifi (activated by default) is also bridged to the internal network.

Internal network DHCP server and IP configuration
: There is a single subnet `192.168.104.0/24` with DHCP enabled. The router has the internal IP `192.168.104.1/32`. There is no IPv6 active in the REDAC internal network.

TCP port forwarding (NAT) for the super controller
: The super controller has a fixed IP address by the mikrotik DHCP server. TCP ports `22, 80, 443` are `dst-nat` to the super controller.

In contrast, the system operator is invited to change not only, but in particular the following default options:

- Changing the behaviour at the external interface, called `ether1`, in particular changing the IPv4 configuration or enabling IPv6.
- Changing or disabling the Wifi AP
- Changing (sharpening or softing) the default firewall
- Changing the access credentials
- Making use of advanced functions such as VPN services.

10.2.2 How to access the REDAC Ethernet/IP Routing Switch

When being connected to REDACs internal network, access the mikrotik web browser based graphical user interface by pointing your browser to `http://192.168.104.1`. Your browser should show you a login mask where you have to fill out the given credentials. At Fig. 10.2 you can see a typical screenshot of the web browser based graphical user interface of the router after having logged in successfully .The page shown allows to turn of the DHCP client behaviour of the REDAC router.

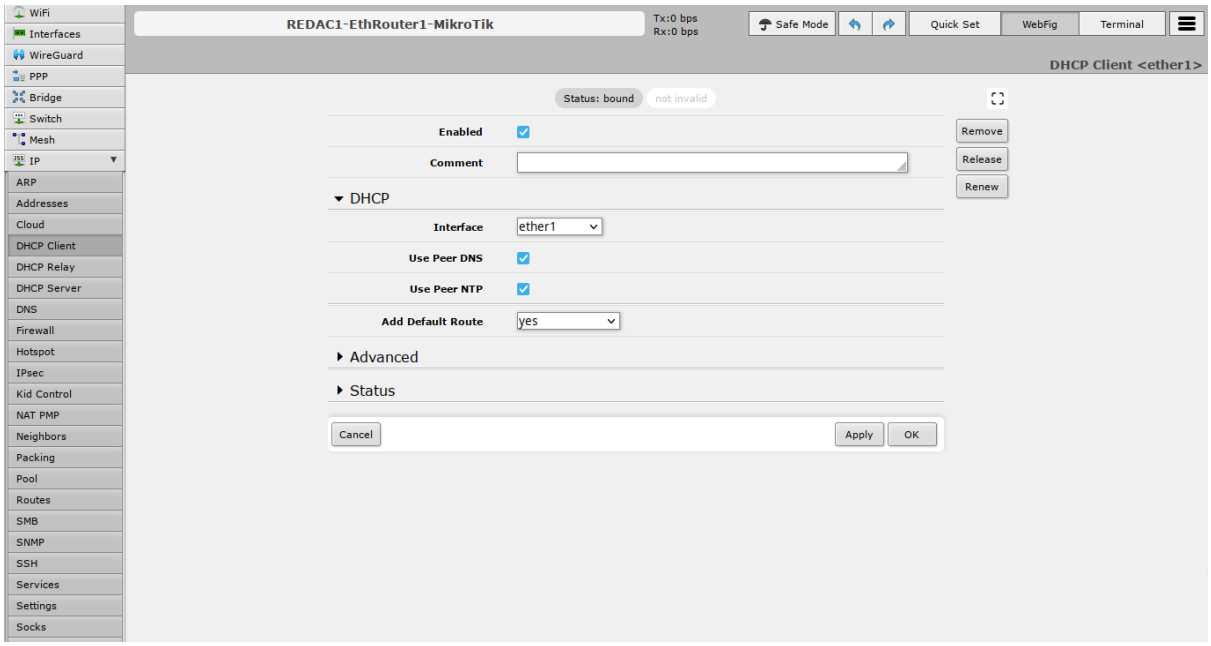


Fig. 10.2: Mikrotik RouterOS Web Browser GUI screenshot

Alternatively, the router provides an SSH console which allows you to display and edit the router settings without a graphical user interface. This comes in handy in particular when you access the system with the linux console (VGA/keyboard directly connected to super controller) without having a USB mouse at hand.

```
some@computer $ ssh admin@192.168.104.1
...
[admin@REDAC1-EthRouter1-MikroTik] > /ip/address/print
Flags: D - DYNAMIC
Columns: ADDRESS, NETWORK, INTERFACE
#   ADDRESS            NETWORK         INTERFACE
```

(continues on next page)

(continued from previous page)

```
0   192.168.104.1/24    192.168.104.0  bridge
1   10.110.0.26/32      10.110.0.0     wg-tinybridge
2 D 192.168.100.168/24  192.168.100.0  ether1
```

10.2.3 Mikrotik default configuration

The device default configuration is shown in the following listing. It can be obtained by typing `/export compact` into the console. Save this listing to a text file and you can restore the device configuration at any time:

```
# 2025-01-08 20:00:42 by RouterOS 7.13.5
# software id = 1SRY-CS15
#
# model = CRS326-24G-2S+
# serial number = HGC09YWZXTY
/interface bridge
add admin-mac=D4:01:C3:80:2D:EB auto-mac=no comment=defconf name=bridge
/interface list
add name=WAN
add name=LAN
/ip hotspot profile
set [ find default=yes ] html-directory=hotspot
/ip pool
add name=dhcp ranges=192.168.104.10-192.168.104.250
/ip dhcp-server
add address-pool=dhcp interface=bridge name=dhcp1
/port
set 0 name=serial0
/interface bridge port
add bridge=bridge comment=defconf disabled=yes interface=ether1
add bridge=bridge comment=defconf interface=ether2
add bridge=bridge comment=defconf interface=ether3
add bridge=bridge comment=defconf interface=ether4
add bridge=bridge comment=defconf interface=ether5
add bridge=bridge comment=defconf interface=ether6
add bridge=bridge comment=defconf interface=ether7
add bridge=bridge comment=defconf interface=ether8
add bridge=bridge comment=defconf interface=ether9
add bridge=bridge comment=defconf interface=ether10
add bridge=bridge comment=defconf interface=ether11
add bridge=bridge comment=defconf interface=ether12
add bridge=bridge comment=defconf interface=ether13
add bridge=bridge comment=defconf interface=ether14
add bridge=bridge comment=defconf interface=ether15
add bridge=bridge comment=defconf interface=ether16
add bridge=bridge comment=defconf interface=ether17
add bridge=bridge comment=defconf interface=ether18
add bridge=bridge comment=defconf interface=ether19
add bridge=bridge comment=defconf interface=ether20
```

(continues on next page)

(continued from previous page)

```
add bridge=bridge comment=defconf interface=ether21
add bridge=bridge comment=defconf interface=ether22
add bridge=bridge comment=defconf interface=ether23
add bridge=bridge comment=defconf interface=ether24
add bridge=bridge comment=defconf interface=sfp-sfpplus1
add bridge=bridge comment=defconf interface=sfp-sfpplus2
/interface list member
add interface=ether1 list=WAN
add interface=bridge list=LAN
/ip address
add address=192.168.104.1/24 interface=bridge network=192.168.104.0
/ip dhcp-client
add interface=ether1
/ip dhcp-server network
add address=192.168.104.0/24 dns-server=192.168.104.1,8.8.8.8 gateway=192.168.104.
↪1 netmask=24
/ip firewall nat
add action=masquerade chain=srcnat out-interface-list=WAN
/system clock
set time-zone-name=Europe/Berlin
/system identity
set name=REDAC1-EthRouter1-MikroTik
/system note
set show-at-login=no
/system routerboard settings
set boot-os=router-os enter-setup-on=delete-key
```

Note that this listing does not include any default passwords, i.e. it will not change the device users or wifi access.

10.3 The Super Controller Server

The Super Controller is the name both for a certain kind of software service and the physical server built into REDAC. This section will only cover the physical server and its operating system. For applications running on this server, see for instance *Services*, *Authentification* and in general the overview of *REDAC-specific software* in the developers manual.

10.3.1 Server Hardware

REDAC ships with a 1U low noise Mini-ITX server based on the SuperMicro Xeon Broadwell SoC server-grade mainboard `X10SDV-4C` and equipped with 32GB RAM and 1 TB SSD. The low depth enclosure has a single power supply and front facing I/O, making it easy to connect a KVM (monitor/keyboard) or USB stick. Most imporant is the IPMI feature which ensures out of band / lights of managament of the server.

This server was chosen due to its compact form factor and not for high availability. In principle virtually any other rack mountable server is suitable to fulfill the job, in particular any product made by large server companies such as Dell, HP or Supermicro.

10.3.2 Server Operating System

The Super Controller server is shipping with the *Ubuntu Server* GNU/Linux distribution and is primarily supposed to be managed via SSH. Access is possible straight from outside the REDAC network as the SSH port is exposed by the router. It is up to the system administrator to also expose the IPMI or to lock down the ports if no remote managament is needed or realized in other fashions.

In general, administrators are expected to be able to manage a Linux server and it is outside the scope of this manual to discuss how to obtain graphical access from remote or how to upgrade the operating system. The same applies with basic monitoring and health checks at an operating system level. The system comes in standard configuration for *Ubuntu Server*, which means for instance that security updates for system packages are installed automatically if internet access is available.

10.3.3 Login and usage of the server

There are a number of differnet ways to access the server: Either you connect physically (see also *Initial network setup/access*) and use the standard VGA terminal (`getty` login) or graphical login (X11 login greeter). Or you connect via network, primarily using SSH. We have the lightweight Xfce (https://xfce.org/) desktop environment preinstalled which includes common tools to get the everyday job done in the graphical user interface.

Via SSH, you have a number of options: Either you use plain SSH, which is just fine for any advanced Linux user and in particular operator. Or you combine SSH with graphical forwarding (i.e. `ssh -x`). The most advanced way is to use the free *terminal server/remote desktop software* X2go (https://wiki.x2go.org/) which is preinstalled. Again, Xfce should be chosen as the installed GUI. A typical configuration screen is shown in figure Fig. 10.3, whereas figure Fig. 10.4 is shown a typical running session.

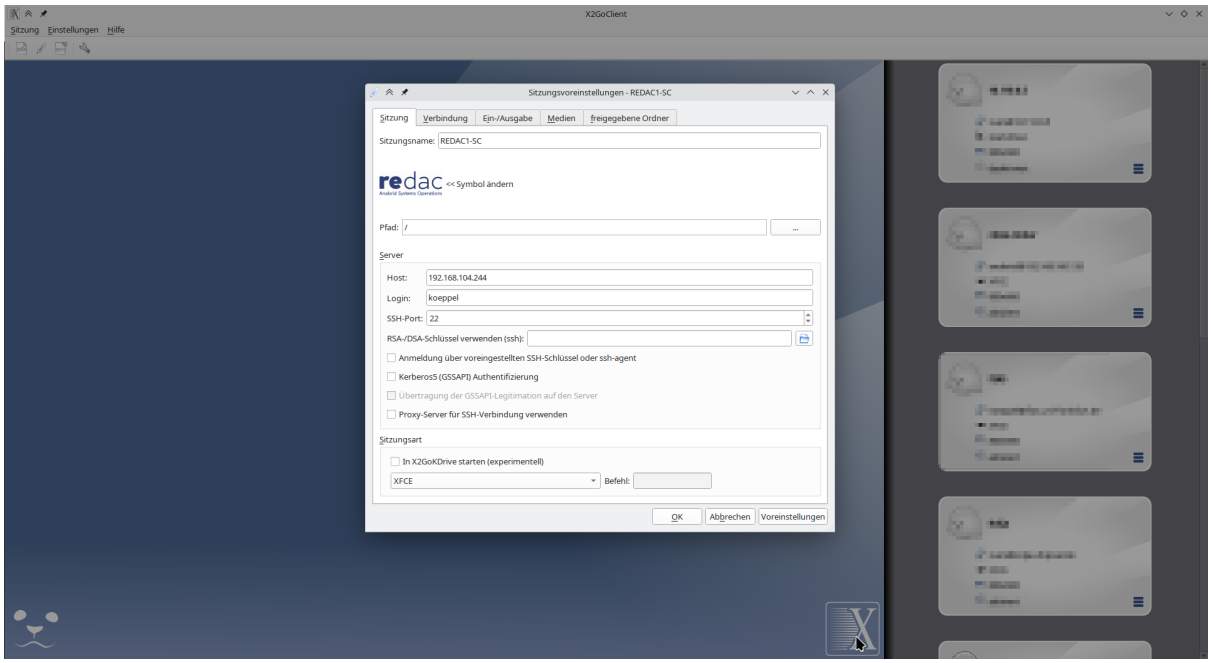


Fig. 10.3: X2Go session initialization screenshot, showing the configuration options.

In order to have a better experience at the (remote) desktop, we recommend to disable the *compositor* (3D features such as windows dropping shadows). We recommend to use `xfce4-terminal` as the terminal application and `firefox-esr` as the web browser.

A powerful feature of X2Go is that it allows you to interrupt and resume sessions the same way

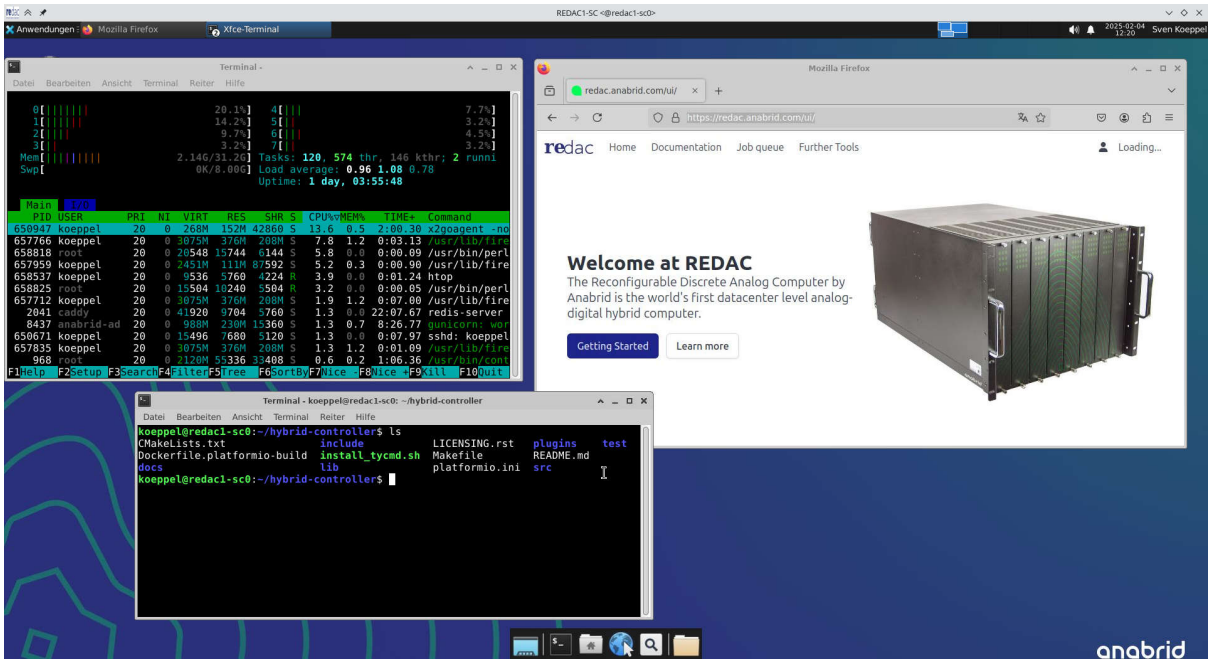


Fig. 10.4: Graphical XFCE session running on the REDAC login server, showing a web browser and a terminal.

as terminal multiplexers such as GNU screen (https://www.gnu.org/software/screen/) and tmux (https://github.com/tmux/tmux/wiki) allow for the terminal. However, the session is by default *not* shared with the direct terminal access session.

10.4 How user access works from outside

This section explains how general network access works, exemplaric for the *Anabrid Operations*:

First, the domain `redac.anabrid.com` resolves to the fiber internet uplink of the anabrid Ulm labs. The router has a Destination-NAT firewall rule which forwards incoming traffic on TCP ports such as 80 (HTTP) or 443 (HTTPS) to the REDAC SuperController within the REDAC internal network. This is possible because on-site, the REDAC internal network is accessible from outside. If this was not the case, the REDAC itself (i.e. the *The Mikrotik Router*) should do a similar DNAT port forarding, given that it has a single external IP address by default.

Warning

DNAT port forwardings and reverse proxying to IP addresses can be fragile if the IP addresses of the relevant devices or services are not static. These situations *do* happen in practice and result in errors which can be hard to trace in particular if administrators are not familiar with the setup. For instance, as a design choice in this network, the IP addresses are managed by permanent, yet principally dynamic DHCP service entries. In particular, devices with multiple network interfaces or changing MAC addresses are subject to different IP addresses if network cables are plugged into other interfaces or MAC addresses are changed manually. On an application level, TCP ports can easily become blocked by stalled server processes where some “clever” server code decides to listen at some other address, rendering the whole system unreachable. In order to be able to debug such problems quickly when they arise, administrators are urged to learn the route of user requests from outside to the inner parts of REDAC and back.

On the SuperController server, there is the caddy webserver (https://caddyserver.com/) running. It provides easy access to Let's Encrypt (https://letsencrypt.org/) certificates and reverse proxying to relevant services. The caddy webserver is steered by the configuration file `/etc/caddy/Caddyfile` (see https://caddyserver.com/docs/caddyfile for syntax explanation) which has the following exemplaric content:

```
# Caddyfile for the SuperController Webserver

(error_handler) {
        handle_errors {
                respond "{http.error.status_code} - {http.error.message} -- hint, check system availability at https://status.anabrid.net/status/redac1"
        }
}

redac.anabrid.com {
        root * /home/caddy-redac.anabrid.com
        file_server

        redir / /ui

        redir /manual.pdf https://anabrid.dev/docs/redac-manual/redacmanual.pdf

        redir /auth  https://auth.redac.anabrid.com/

        redir /api /api/
        reverse_proxy /api/* 127.0.0.1:8081  #  redaccess code = main REST interface

        import error_handler
}

auth.redac.anabrid.com  {
        reverse_proxy 127.0.0.1:8080  #  keycloak = main Authentification
        import error_handler
}

stats.redac.anabrid.com  {
        reverse_proxy 127.0.0.1:3000  #  grafana
        import error_handler
}

jupyter.redac.anabrid.com {
        reverse_proxy 192.168.102.251:8080  #  *EXTERNAL* jupyter notebook server
        import error_handler
}

# ...
```

This example shows how the Caddy server distributes incoming requests based on their domain name or path. Given the *current* permeability between the Ulm lab network and the REDAC inter-

nal network, it is easy to reverse proxy also network services which are not physically hosted on the same server.

CHAPTER

ELEVEN

SERVICES

This section lists the names and explains the maintenance of the REDAC-specific software on *The Super Controller Server*. This also includes the supervision of log files and their general availability.

First of all, all REDAC software comes dockerized (https://www.docker.com/) and/or is maintained as systemd services (https://systemd.io/). One of the most important properties is that all relevant services come up *automatically* on system boot. This is ensured by having

- Python daemons being directly started by systemd units (within their relevant virtualenv (https://virtualenv.pypa.io/en/latest/user_guide.html)).
- Single docker daemons being directly started by systemd units.
- Docker-Compose (https://docs.docker.com/compose/) services being automatically started by docker.

If you don't know how to maintain a sytemd service, please advise any contemporary linux administrators manual. The same applies with docker and in particular `docker-compose`.

11.1 Relevant systemd units

The relevant systemd units are named:

- `redaccess-forward.service` (*Redaccess (Middleware)*)
- `redaccess-api.service` (*Redaccess (Middleware)*)
- `supercontroller-proxy.service` (*Pybrid Proxy*)
- `docker-keycloak.service` (*Authentification*)

You can edit the service description files in `/etc/systemd/system/<nameOfService>`.

For each service, you can its status with `sudo systemctl status <nameOfService>`. You can view the relevant logfiles with `sudo journalctl --unit=<nameOfUnit>` (with `<nameOfService> = <nameOfUnit>.service`). Helpful options for `journalctl` are the *follow flag* `-f` to get interactive, continous output as well as the date filtering such as `--since today` to see the relevant logs only.

This is the output of a typical status report:

```
you@redac1-sc0 $ sudo systemctl status supercontroller-proxy.service
o supercontroller-proxy.service - REDAC SuperControl Proxy
    Loaded: loaded (/etc/systemd/system/supercontroller-proxy.service; static)
    Active: active (running) since Tue 2025-02-11 07:34:48 UTC; 5min ago
  Main PID: 3718259 (python)
```

(continues on next page)

(continued from previous page)

```
     Tasks: 2 (limit: 38307)
    Memory: 26.5M (peak: 27.0M)
       CPU: 995ms
    CGroup: /system.slice/supercontroller-proxy.service
            +- 3718259 /home/anabrid-admin/.cache/pypoetry/virtualenvs/pybrid-
↪computing-XZRhPXGJ-py3.12/bin/python -m pybrid.cli.base --log-level=DEBUG redac␣
↪-h 192.168.104.0/24 proxy --ma>
...
Feb 11 07:34:57 redac1-sc0 python[3718259]: 57.677 | WARNING | proxy | Target for␣
↪MAC mapping from 00-00-00-00-00-00 to 04-E9-E5-17-E5-4F does not exist.
Feb 11 07:34:57 redac1-sc0 python[3718259]: Starting proxy on 0.0.0.0:5732...␣
↪Press Ctrl+C to exit.
```

Furthermore, the following services are part of ubuntu software packages, they are not custom REDAC software but relevant for correct operation:

- caddy with its configuration file at `/etc/caddy/Caddyfile`. This is the HTTPS webserver, REST reverse proxy and main entrypoint. After changing the configuration file, a `service caddy reload` is sufficient.
- `ssh` is the OpenSSH server which is crucial for managing the system from remote.
- `lightdm` is the graphical display manager (greeter), see also *Login and usage of the server*. When you have trouble with the graphical terminal, try to restart this service.

11.2 Docker service overview

The following docker services are used/installed:

- `redac-keycloak` is a single keycloak (https://hub.docker.com/r/keycloak/keycloak) image from dockerhub (https://hub.docker.com), managed by a systemd unit (see above) with relevant options in the service description file.
- Grafana as a `docker-compose` setup

You can get a quick overview about running services with these commands:

```
you@redac1-sc0 $ sudo docker ps
CONTAINER ID   IMAGE                                    COMMAND                  ␣
↪CREATED          STATUS                  PORTS                                  ␣
↪                                                 NAMES
bb9982649e60   quay.io/keycloak/keycloak:26.0.7         "/opt/keycloak/bin/k. . . "  ␣
↪47 minutes ago   Up 47 minutes           8443/tcp, 0.0.0.0:8080->8080/tcp,␣
↪:::8080->8080/tcp, 9000/tcp                          redac-keycloak
b9ce90eaa284   redis2influx                             "poetry run python -. . . "  ␣
↪21 hours ago     Up 14 hours                                                    ␣
↪                                                 redis2influx
f0b7f3d79034   grafana/grafana-oss                      "/run.sh"                21␣
↪hours ago     Up 14 hours             0.0.0.0:3000->3000/tcp, :::3000->3000/tcp␣
↪                                            grafana
515dd4d984cc   influxdb:2                               "/entrypoint.sh infl. . . "  ␣
↪21 hours ago     Up 14 hours             0.0.0.0:8086->8086/tcp, :::8086->8086/
```

(continues on next page)

(continued from previous page)

```
↪tcp                                                    influxdb
c1ef8172ab3b   redis/redis-stack                          "/entrypoint.sh"         21␣
↪hours ago     Up 14 hours             0.0.0.0:6379->6379/tcp, :::6379->6379/tcp,
↪ 0.0.0.0:8001->8001/tcp, :::8001->8001/tcp   redis
20dacd1c8e2a   ghcr.io/goauthentik/server:2024.12.2   "dumb-init -- ak wor. . . "  ␣
↪4 weeks ago      Up 14 hours (healthy)                                          ␣
↪                                                    authentik_worker_1
4376dc22d03d   ghcr.io/goauthentik/server:2024.12.2   "dumb-init -- ak ser. . . "  ␣
↪4 weeks ago      Up 14 hours (healthy)   0.0.0.0:9000->9000/tcp, :::9000->9000/
↪tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp   authentik_server_1
fddcad498baf   postgres:16-alpine                      "docker-entrypoint.s. . . "  ␣
↪4 weeks ago      Up 14 hours (healthy)   5432/tcp                               ␣
↪                                                    authentik_postgresql_1
716681800dab   redis:alpine                            "docker-entrypoint.s. . . "  ␣
↪4 weeks ago      Up 14 hours (healthy)   6379/tcp                               ␣
↪                                                    authentik_redis_1
```

For instance, the Grafana installation is managed with

```
you@redac1-sc0:/path/to/grafana # docker-compose ps
    Name                Command            State                                ␣
↪      Ports
-------------------------------------------------------------------------------
↪-------------------------------------------------
grafana        /run.sh                     Up      0.0.0.0:3000->3000/tcp,:::3000->
↪3000/tcp
influxdb       /entrypoint.sh influxd      Up      0.0.0.0:8086->8086/tcp,:::8086->
↪8086/tcp
redis          /entrypoint.sh              Up      0.0.0.0:6379->6379/tcp,:::6379->
↪6379/tcp, 0.0.0.0:8001->8001/tcp,:::8001->8001/tcp
redis2influx   poetry run python -m src    Up
```

Further details will follow as soon as the software is more mature. There will be a focus on installation-specific details.

11.3 How to (re-)install the relevant software on the server

In most cases, reinstallation will be as easy as an `docker-compose pull`. Note that the REDAC software for the super controller is not open sourced and thus updates will be provided by anabrid if there is an appropriate contract discussing this in detail.

In the moment, please refer to the section about *Software* in the developer's manual for further detail.

CHAPTER

TWELVE

AUTHENTIFICATION

For user authentification, the REDAC Frontend Access Code (*redaccess*) uses the OpenID (https://openid.net/) standard. For convenience, the identiy and access managament server Keycloak (https://www.keycloak.org/) is running on the *Super Controller*. Keycloak provides user federation, strong authentication, user management, fine-grained authorization, and more. This section describes the capabilities of this software and how to manage access to REDAC. Administrators are strongly encouraged to read the relevant Keycloak guides (https://www.keycloak.org/guides) and documentation. This section will only cover the REDAC-specific configuration settings and intentionally does not provide a general introduction into the Keycloak software.

Note

The Keycloak OpenID service is supposed to secure access to all **user facing services**. This intentionally does not cover any services of the *Internal network*. All operator and administrator services have their local on-device user account managament. This is also a resilience measure in order to enable maintainability of the overall system even if REDAC-specific services are down.

12.1 Scope and capablities of Keycloak on REDAC

By default the Keycloak installation has two realms:

1. The *master* (or *default*) realm which is only used for managing Keycloak itself. Login is at https://auth.redac.anabrid.com/ which shows a big warning that this is not the correct login for ordinary users.
2. The *redac1-realm* for actual user authentification, currently managed by *Anabrid for QCI*. The most important URL for managing users and their groups in this realm is https://auth.redac.anabrid.com/admin/master/console/#/redac1-realm/users. For programming against this realm, the information at the ressource https://auth.redac.anabrid.com/realms/redac1-realm/.well-known/openid-configuration may be interesting to get the URLs of OpenID endpoints.

The Keycloak itself is currently configured to be able to send out E-Mails via an anabrid mailserver with the sending address `redac1-keycloak@anabrid.dev`. This is a *no-reply* sender only mail account for the keycloak daemon.

Keycloak is versatile in client policies and modifying the user registration.

12.2 REDAC Keycloak clients

In Keycloak language, *clients* refer to programs which want to authenticate against REDAC. Currently, the following *OpenID connect* clients with *authentification and authorization* capabilities are registered:

- `redaccess`, served at https://redac.anabrid.com/api
- `jupyterhub-wup`, served at https://jupyter.redac.anabrid.com/
- Keycloak-internals such as the account console at https://auth.redac.anabrid.com/realms/redac1-realm/account/

The following *OpenID connect* clients with *public access only* capabilities are registered:

- `redac-gui`, served at https://redac.anabrid.com/ui

Part III

Developers manual

CHAPTER

THIRTEEN

INTRODUCTION

Welcome to the Software and Hardware Developers handbook of REDAC, the Reconfigurable Discrete Analog Computer. Please ensure you have read the *User manual* before approaching this part of the system documentation.

REDAC is primarily supposed to be used by engineers and scientists. Depending on your domain of expertise, it may be sufficient for you to adopt your problems to the programming interfaces provided by REDAC. This *developer manual* is primarily meant for people who want or need to know more about the internals of REDAC analog and digital parts.

Please note that the level of access to the inner workings of the system depends on your particular legal situation: Despite some parts of REDAC are open sourced (or supposed to be open sourced soon), most of the system is neither open source nor open hardware. You, your institute or your company may need to purchase a license, sign a non disclosure agreement (NDA) or take another other legal action in order to gain access to the information of interest. Please contact your systems operator or contact person for further information.

CHAPTER

FOURTEEN

ARCHITECTURE REFERENCE

This section provides a deep dive into REDAC architecture, suitable for developers and experienced users who want to dig into the connectivity provided by the device. You should read the *introductory section on REDAC architecture* first.

14.1 REDAC Hierarchy

REDAC stands for *Reconfigurable Discrete Analog Computer*. This means it is constructed using *discrete* components, which are individual integrated circuits (ICs). These ICs house, for instance, multiple operational amplifiers (Opamps), serving as the fundamental building blocks for the electronic analog computing paradigm employed by REDAC.

The system is designed with a highly modular, hierarchical, and repetitive architecture. At each hierarchical level, components from the lower levels are arranged in a consistent, repetitive structure while remaining individually replaceable. This design achieves a balance between high regularity for efficient organization and substantial flexibility for customization. This modularity not only simplifies maintenance and scalability but also enables tailored configurations to suit specific computational needs. The following figure Fig. 14.1 provides an overview picture emphasizing on the hierarchical and modular nature of REDAC.

The figure introduces a lot of concepts which are presented in detail in the following sections. Despite REDAC is an analog-digital hybrid computer, the architectural description will primarily explain the *analog* connectivity, apart from the description of digital circuits.

14.1.1 Single cluster

A *cluster* is the atom of the REDAC computer. The circuits at this level are alsor refered to as *macro cell* in the literature. A cluster provides all-to-all connectivity between up to 16 computing elements. Coupling is implemented by switching matrices under control of the attached digital computer, basically resembling a *crossbar switch*.

Schematically, Figure Fig. 14.2 provides a way to visualize the interconnection matrix with real numbers representing coupling strengths. The cluster system matrix is the adjacency matrix for the interconnection graph between the computing elements. The entries are referred to as *weights*. As a special property, the system matrix does *implicit summing* which can be thought of as matrix multiplication linear algebra operations: $C_i^{\text{in}} = \sum_{j=0}^{16} M_{ij} C_j^{\text{out}}, \quad i,j \in [0,16]$ where C_i^{in} is the input to the i-th computing element, C_i^{out} is its output, and the M_{ij} are the weights. The previous equation describes *monadic* computing elements with one input and one output. The integrator is the only monadic computing element in a cluster system. It computes the time integral over its time-varying input signal. REDAC also provides another kind of computing elements, which are *dyadic*, i.e. feature two inputs and one output: Multipliers. The multipliers implemented allow for full four-quadrant-

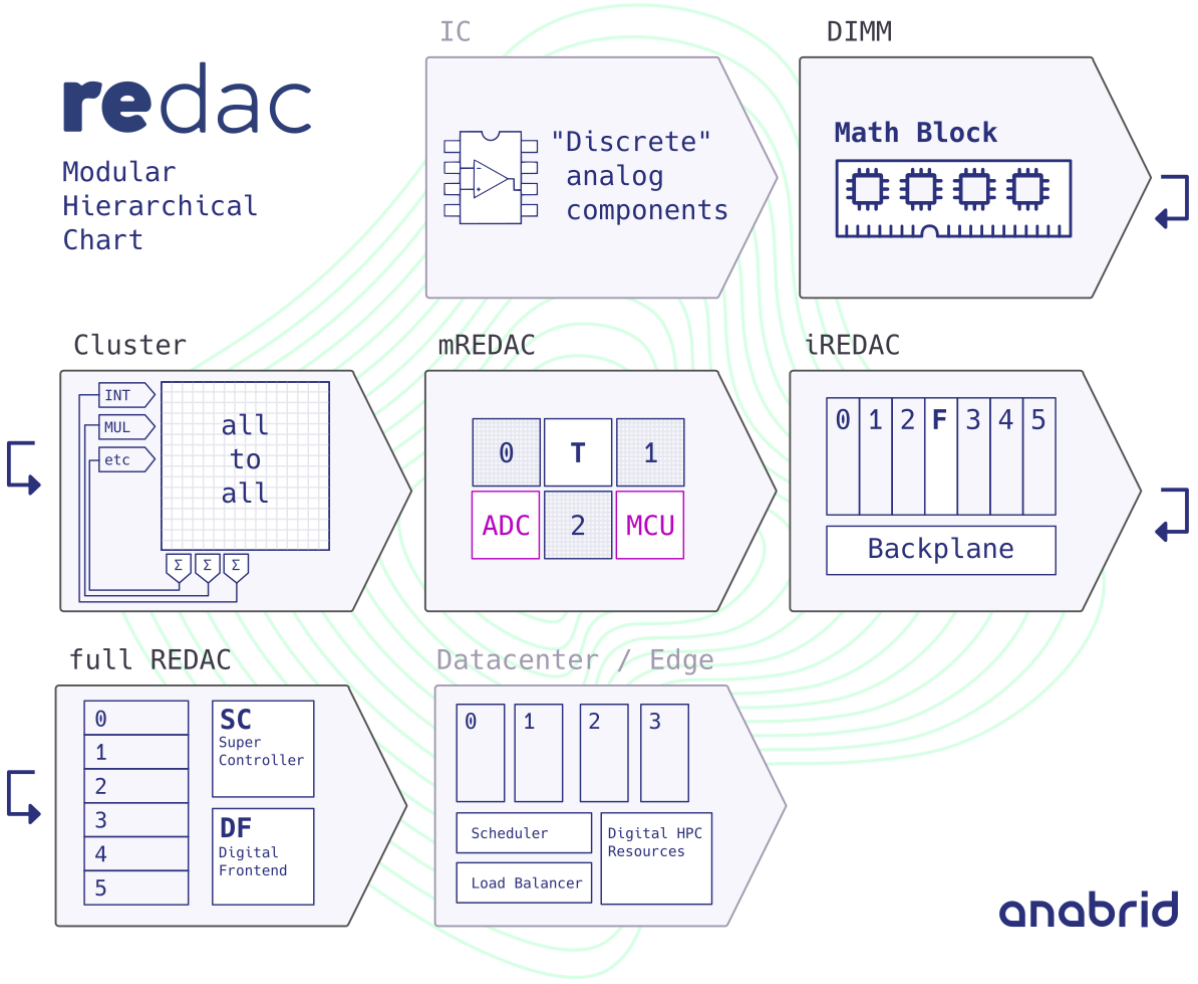


Fig. 14.1: Abstract hierarchy diagram of REDAC.

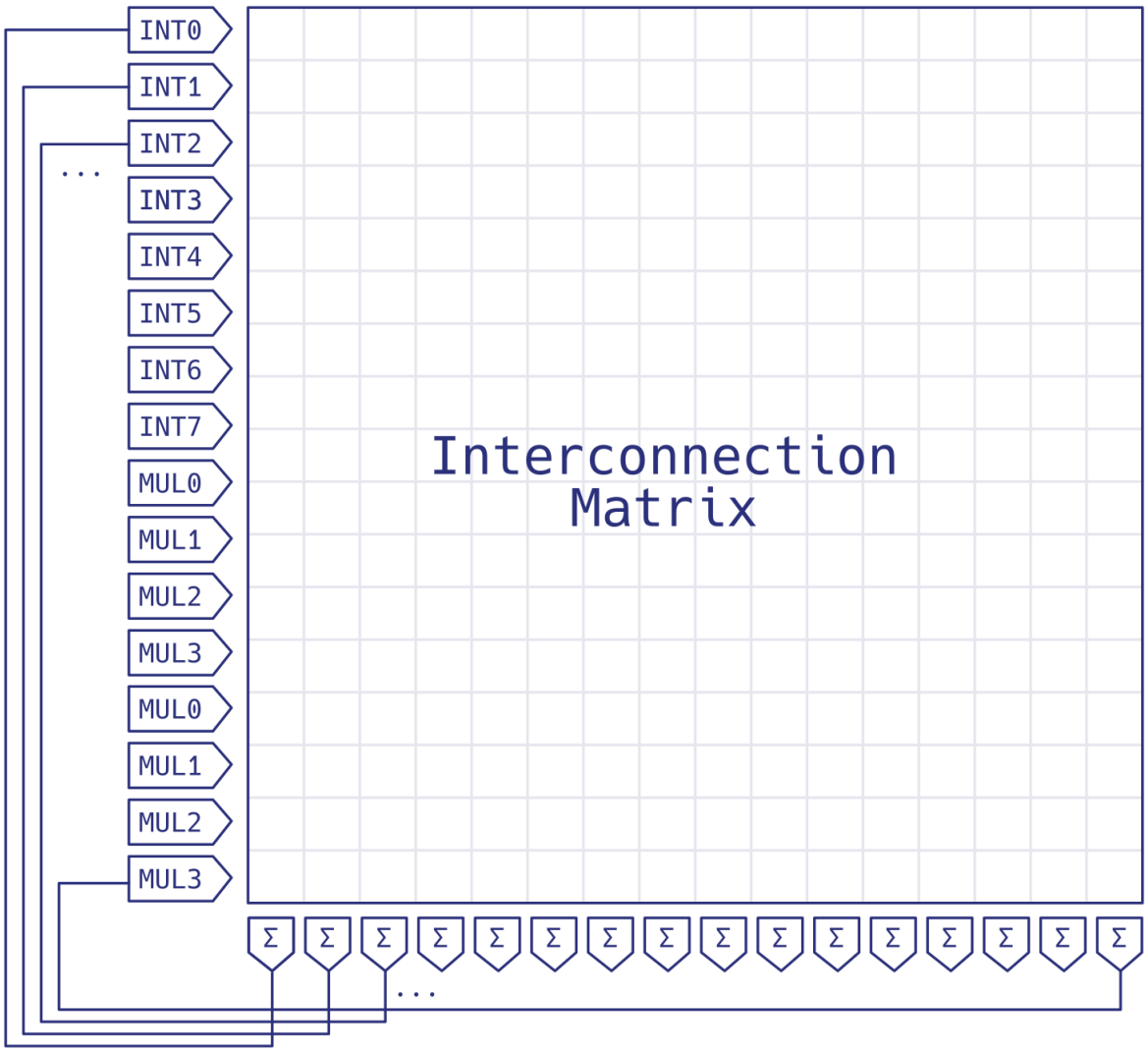


Fig. 14.2: The single cluster system reduced to its system matrix. The entries shown are for a Lorenz system.

multiplication, i.e. each of the input values can take on values within the whole machine unit interval $[-1, 1]$.

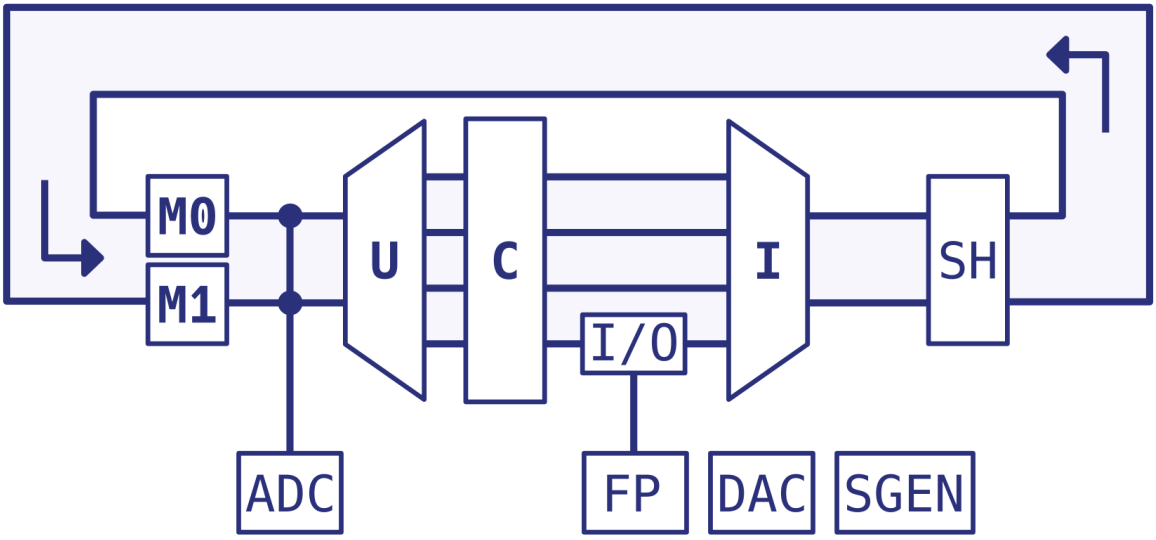


Fig. 14.3: The single cluster as a closed-loop feedback circuit (*turbine diagram*).

Figure Fig. 14.3 shows a more detailed block diagram of the REDAC. It shows the closed loop *analog compute path*. The labeled shapes in the diagram each represent a particular *blocks* (in bold letters, referred to as "entities" in the REDAC terminology) and auxiliary *elements* (in normal letters). The figure does not show any digital control/configuration signal paths. Most notably, the **U**-, **C**- and **I**-blocks together form the **UCI** matrix, which corresponds to the simplified *interconnection matrix* shown before in Figure Fig. 14.2. In contrast, Figure Fig. 14.3 emphasizes the internal *fanout* and *fanin* properties of the U- and I-blocks, resulting in a turbine-like appearance of this circuit representation.

The UCI matrix has 16 inputs, 16 outputs and 32 internal signal paths, each of which contains a coefficient element. These 32 paths are also called *lanes* and correspond to up to 32 nonzero entries in the system matrix, yielding a maximum matrix density of $32/16^2 = 12.5$. This corresponds to a minimum sparsity of 87.5%.

14.1.2 mREDAC level

mREDAC stands for *minimal* or *motherboard* REDAC and is the smallest autonomous part of the system. It contains one microcontroller (MCU) which has an Ethernet uplink and steers all digital signals on the motherboard. The board hosts three clusters which share a single OP/IC line. The three clusters are interconnected by means of one T block. This minimal self-standing analog hybrid computer is completed with analog aquisition circuitery (eight ADC channels) as well as digital/analog external I/O. This way, an mREDAC is basically the smallest standalone unit of REDAC.

14.1.3 iREDAC level

iREDAC stands for *intermediate* or *interconnecting* REDAC. At this level, seven mREDAC units are connected to a single backpanel. Interconnection is enabled by means of several T blocks. Furthermore, several digital busses on the back panel enable the orchestration and in particular synchronization of the mREDACs. The iREDAC comes in a modular 19" rack-mounted blade-style appearance where mREDACs can be conveniently accessed from the front for maintenance purpose.

14.1.4 REDAC level

The whole REDAC system is composed by order of magnitude 10 iREDAC units. The large number of microcontrollers are steered by a single *supercontroller* (in short *SC*). This term refers both to the hardware (a single server grade computer) and a particular networking software which serves as single end point for all analog parts of REDAC. From the hardware perspective, the REDAC further contains Ethernet/IP administration (in the simplest case one switch and one router). From the software perspective, REDAC contains a managament interface ontop of the SC which does job managament, user authorization, encryption, provides API endpoints and custom software support such as compute intensive circuit transpilation. For further information on the software architecture, please refer to the *Developer Manual*.

14.2 Block reference

This section provides a reference of all function blocks within REDAC. Typically, a *block* is realized as a DIMM form factor PCB. Furthermore, typically every block is also an *entity* in the concept of the REDAC hybrid controller (see firmware documentation for details).

14.2.1 Cluster blocks

All blocks within a single cluster are part of the compute path and shown in figure Fig. 14.3.

`M0` and `M1` blocks
: M-blocks are *math blocks*. Each can have up to 8 analog input and 8 analog output signals. Math blocks contain various elements and allow digital configuration of these elements. In contrast to classic analog computers, there are no summers available as explicit computing elements, as summation is done implicitly in the I-block. This enhances the overall flexibility considerably. For details see *Math blocks*.

`U` block
: The output signals of these M-blocks are connected to the U-block which contains a 16×32 crossbar switch. Since the output signals of the M-blocks are voltages, this crossbar switch can distribute one signal to several outputs at once. It therefore serves as a 16:32 fan-out, allowing to distribute the input signals arbitrarily on 32 internal lanes within the UCI interconnection matrix.

`C` block
: These 32 output signals (still represented by voltages) are then fed into the *coefficient block*. This contains 32 digitally controlled coefficient potentiometers with 11 bits resolution and an additional sign bit. Thus, the C-Block implements one coefficient per lane by means of a multiplying DAC (a certain type of digital analog converter). These coefficients allow scaling of values with factors in the interval $[-1, 1]$. The output of these coefficients are now currents instead of voltages and feed the I-block.

`I` block
: Following the C-block is the I-block. It, too, is a crossbar switch, this time with 32 inputs and 16 outputs. Working with currents, it is now possible to implicitly sum several inputs, thus eliminating the need for explicit summers as computing elements. The 16 output lines of the I-block are connected to the inputs of the computing elements contained on the M-blocks. The I-block also features a programmable gain of either factor 1 or 10, allowing for a broader dynamical range of the machine.

`SH` block
: This block contains 16 *sample and hold* elements to correct offset errors of the computing ele-

ments. It is transparent for the analog signal path and only serves for signal conditioning. It is part of the sophisticated error correction techniques of the LUCIDAC.

14.2.2 mREDAC and iREDAC level blocks

`CTRL` block
: This block contains the *hybrid controller* based on the MCU. It is responsible for setting up the computing elements, coefficient potentiometers, crossbar switches. It also takes care of the communication with the digital upstream computer (the *client* in a networking setting or *host* in a USB setting). The control block physically holds the MCU as well as eight analog to digital converters.

`T` block
: This block is used for interconnecting clusters. The `T` stands for *tee piece* but also *topology* or *transparent*. One jokes that it also stands for *Tricky* or *T-Rex*, given its highly regular yet unintiuitive way of interconnecting a large number of analog signals. The `T` block is not part of a single cluster. Each mREDAC has one `T` block and each iREDAC backplane has three `T` blocks.

  Technically, a T-block provides 96 voltage coupled analog I/O that go to 24 analog switches. Each switch is a 1-to-3 fanout or 3-to-1 fanin, depending on the information flow in the usage mode.

14.2.3 Math blocks

Despite the implicit summing in the networking topology, all linear and nonlinear analog computation is performed on the Math blocks (M-blocks). REDAC math blocks are modular and can be exchanged, despite this requires opening the device.

A single cluster has slots for two math blocks. As of now there are two types of M-blocks available:

Integrator Math block `MInt`
: contains eight integrators (each with one input with implicit summing, one output). Integrators have an internal analog state (the current integration value), a digital state (IC / OP / HALT state machine), and a hybrid state (initial conditions and time scaling factor k_0).

Multiplier Math block `MMul`
: contains four multipliers (each with two implicit summing inputs, one output). The four remaining outputs of this block are used as identity elements. They can be used for more flexibility in the connection topology.

In normal conditions, one cluster contains eight integrators and four multipliers.

14.3 Digital Network

The REDAC digital network is actually made out of several networks:

- The multi-purpose digital bus within a single mREDAC. It has roughly 11 address lines and roughly 5 data lines which are primarily used for SPI communication. This bus is extended to the iREDAC backplanes.
- *The REDAC-internal network* used for communication between MCUs and SuperController (never used to communicate directly between MCUs).
- Global (serial) mode lines steered by a single MCU which is currently USB-connected to the SuperController.

CHAPTER

FIFTEEN

SOFTWARE

The REDAC's software stack drives a distributed and heterogenous system of computers. Figure Fig. 15.1 shows a high level overview of this system.

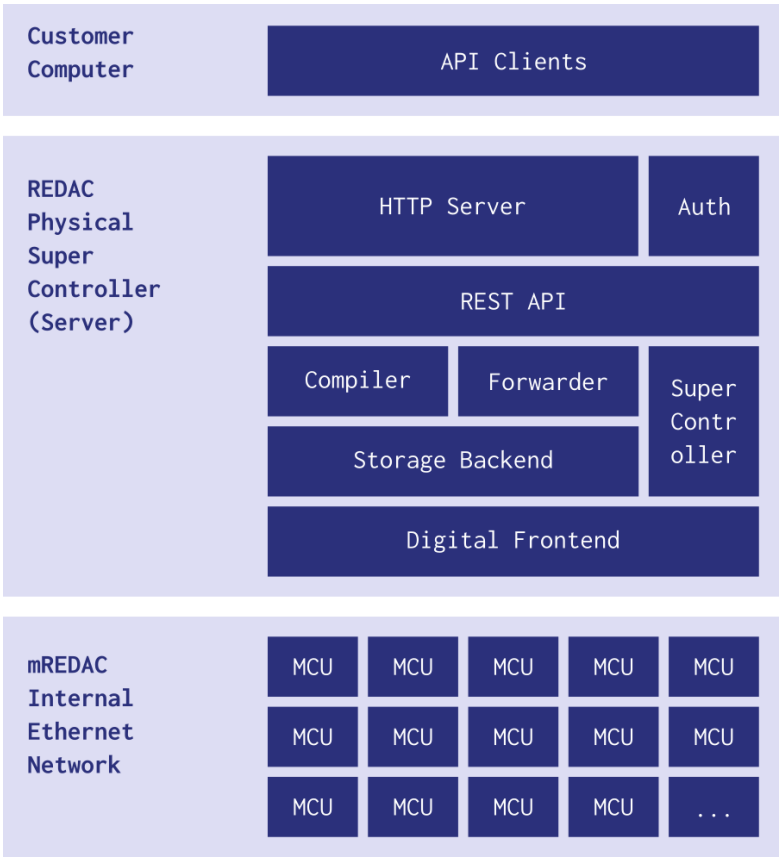


Fig. 15.1: Abstract software diagram of REDAC.

The purpose of this section is to give a quick overview about the particular parts and to refer to their respective documentation. The following sections will go top down.

15.1 Client Computers

On the customers or client computer, a small code is running, called the *REST API client*. REDAC has an *OpenAPI* defined frontend which makes it easy to explore the interaction methods programmatically and to generate client codes.

A reference implementation of this client is currently available in Python on the official python package index at anabrid-redac-client (https://pypi.org/project/anabrid-redac-client/). Getting this code

started is part of the *Getting started* section in the users guide. This code is supposed to be open sourced at Github. Further clients shall be generated in more programming languages in order to make it easy to operate with REDAC from various settings.

15.2 Software on the SuperController

The *SuperController* is mastering the internal network (see also section *Internal network* in the operators guide). The name *SuperController* refers both to the physical server as well as to a major software component which manages the embedded hybrid controllers. See also section *Services* in the operators manual for a description how to manage this software from a devops perspective.

15.2.1 Redaccess (Middleware)

At the interface between the SC and the REDAC public access is a lot of middleware codes which are not meant to be released as open source. Most notable is the *Redaccess* code. It orchestrates a dockerized infrastructure containing parts such as the Redis (https://redis.io/) and PostgreSQL (https://www.postgresql.org/) databases or the RabbitMQ (https://www.rabbitmq.com/) broker. Furthermore, part of the software stack is an OpenID provider (Keycloak (https://www.keycloak.org/)), all running on standard server hardware and software, see the operator's manual on *Internal network* for details.

REDAC is equipped with a number of supplementary software for monitoring or enhancing user experience, such as a little landing website or a JupyterHub (https://jupyter.org/hub) instance. However, none of these services are part of the core functionality and not neccessarily run on hardware directly related to REDAC.

15.2.2 Pybrid Proxy

The REDAC-internal ethernet/IP network with its hunderds of microcontrollers is centrally managed by the *Pybrid Proxy* software. The tasks of this software are the following:

- Application level proxy which provides a single entrypoint at TCP port 5732 speaking the firmware JSONL-protocol, effectively resembling a "big virtual hybrid controller". This means it has to distribute incoming queries to the relevant microcontrollers and collect their replies again.
- Doing all the bookkeeping and resolving logical device addresses (encoding their positiong in the connectivity graph) to physical devices (encoding their actual addressing in the digital network, i.e. by MAC and IP address).
- Zookeping: Monitoring and scanning available mREDACs or their microcontrollers, respectively. Ensuring the relevant firmware is available and updating it when needed. Steering the power and turning off parts of the system which are not neededd. Monitoring their health for instance by polling the temperature sensors available in the system.
- Organization of *run groups* by means of steering a system wide digital bus which is currently USB-connected to the server.

Pybrid is written in asynchronous python and documented at https://anabrid.dev/docs/pybrid/. The code forms the *digital frontend* to all analog circuitery. In regular use, this code should be the only software which communicates with the embedded controllers directly.

15.3 Firmware on Microcontrollers

REDAC has a hierarchical design where on the lowest level (*mREDAC level*) there are microcontrollers (MCU) with similiar firmware images. The following diagram Fig. 15.2 provides an overview of the firmware architecture, with a focus on the communication.

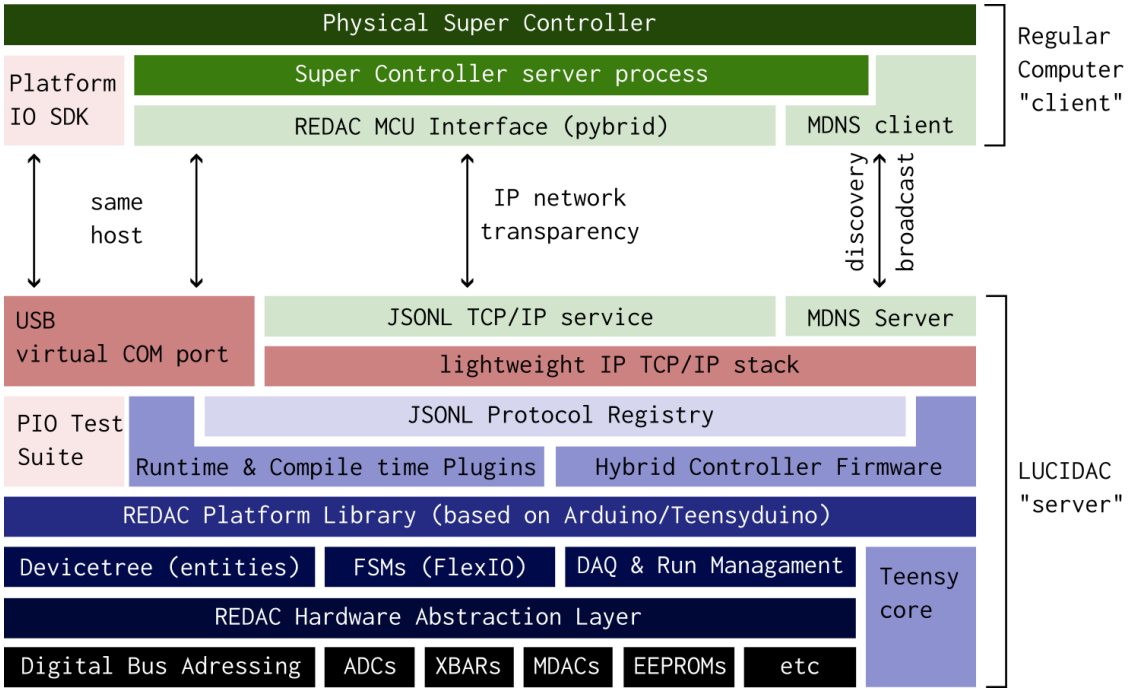


Fig. 15.2: REDAC embedded hybrid controller firmware architecture

It is important to note that end users **never** directly communicate with the embedded controllers. Even the direct contact to the SuperController is prohibited by the intermediate software.

Despite the MCUs are actually interchangable, REDAC is equipped with 600Mhz 32bit ARM Cortex-M7 single core Teensy 4.1 development boards (https://www.pjrc.com/teensy/). The firmware is developed as a PlatformIO (https://platformio.org/) project and is huge: It contains more then 20,000 lines of code in dozens of folders. The firmware has been open sourced in the past at https://github.com/anabrid/lucidac-firmware/ and is intensively documented at https://anabrid.dev/docs/lucidac-firmware/sphinx/dirhtml/.

CHAPTER

SIXTEEN

USAGE MODES

The REDAC system is still in development, and thus are the usage modes. Despite we have a single user mode in mind which solely goes via a single entrypoint, which does scheduling, load balancing and the like, in special situations a number of different usage modes are supported in the moment. A graphical overview in a block schema diagram is provided in Fig. 16.1 and subsequentialy discussed.

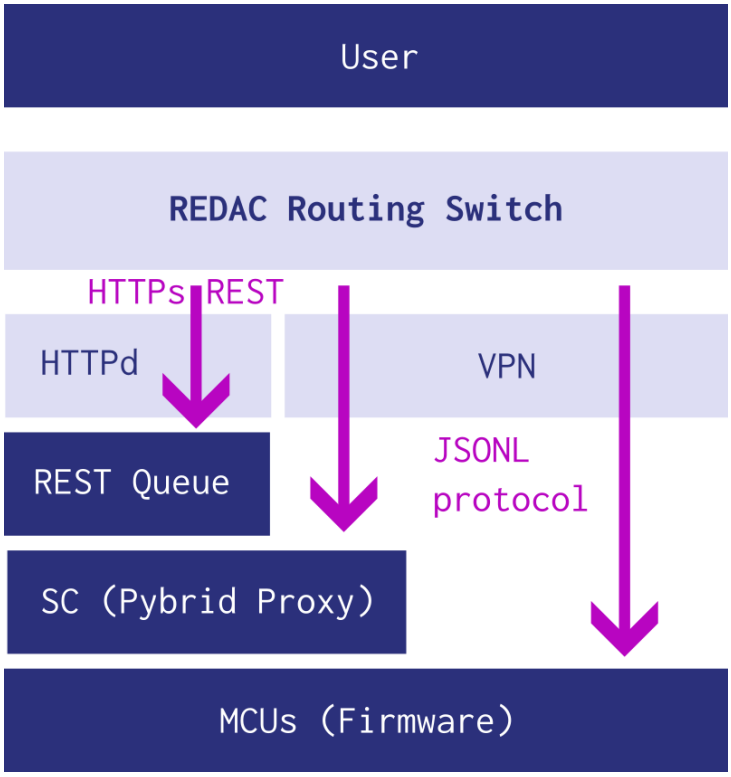


Fig. 16.1: The different inofficial user access variants for REDAC

16.1 Queue access

The primary access is indirected by the *Redaccess (Middleware)* via HTTP and REST. This way, ordinary encrypted HTTPS can be used and the regular *Authentification* methods can be employed. Therefore, this access method is the only one which is advertised in the REDAC user manual.

16.2 Immediate SuperController access

The immediate mode access to the *Pybrid Proxy* is technically only possible when being within the *Internal network*. This is for instance possible via a VPN solution or with appropriate firewall rules from outside. It is important to note that in this mode, no user authentification is applied, computing time is not accounted and the TCP connection to the SuperController is not encrypted. Furthermore, the protocol is no more HTTP but the specific JSONL protocol which is documented in the pybrid and firmware documentations.

This immediate mode can be interesting for debugging, for experienced users, in particular single system users. Users are suggested to use the `pybrid` python package to connect to the pybrid proxy.

16.3 Immediate Microcontroller access

In this variant, even the SuperController is short-circuited and users directly access the mREDAC-embedded microcontrollers. This is possible if users are part of the *Internal network* (see previous section). However, as a drawback to the immediate SuperController access, in this mode users can not make use of the global synchronization line. That means that synchronized computations accross more then a single mREDAC are not possible.

This kind of access is strongly discouraged whenever any part of the higher level software abstraction is active, i.e. even if the pybrid proxy is active. If users want to make use of this access mode, they should use the `pybrid` or `lucipy` sofware packages in order to address the mREDAC boards.

16.4 JupyterHub/Browser access

JupyterHub is a hosted computation software provided by *Anabrid Operations*. It is basically a hybrid mode which allows all kind of uses. It combines a number of advantages:

- User does not need to install any software/development environment locally. He finds the open source scientific programming environment *jupyter* readily available in browser, comparable to Matlab.
- Regular queue access is possible
- However, also direct access is possible if neccessary since the instance has access to the REDAC internal network.

CHAPTER

SEVENTEEN

HARDWARE

REDAC consists of a large number of custom printed circuit boards (PCB) designs made by anabrid. The KiCad EDA (https://www.kicad.org/) software was used to produce the schematics and layouts. The PCBs host carefully selected third party analog, digital or mixed signal chips. Next to PCBs, REDAC consists of metal support structures in standard industry format (DIN subrack kits, IEC 60529) made of aluminium or steel as well as industry-standard cable/connectors of various types.

The whole schematics of the REDAC computer require roughly 200 to 300 pages when being printed out. The circuits are versioned, this versioning is carefully tracked in the firmware development so there are lowlevel “drivers” for every piece of hardware developed (despite, of course, old hardware revisions are typically deprecated).

Some REDAC circuits have international patents granted or pending.

Readers interested in the REDAC hardware should read the high-level description of *Architecture Reference* first. It is important to note that, despite being modular, the whole REDAC system should be seen as a unity. This means that when dealing with a complex system such as REDAC from a hardware perspective, one must assume *leaky abstraction* or a *violation of seperation of concerns*. For instance, despite an *iREDAC* should be an independent unit standing for itself, in REDAC several iREDACs are steered by a single digital SuperController and thus there is certain interdependence. The same might be true due to shared power supplies, for instance.

For schematics access, please contact the person who is/was responsible for purchasing your REDAC setup, your administrator/operator or the manufacturer, anabrid (https://anabrid.com/contact).

CHAPTER

EIGHTEEN

APPENDIX

18.1 On the REDAC handbook

Most of the developers manual is about *external* documentation and material, in particular in the *Software* section but also the *Hardware* section. In contrast, this section is all about the REDAC manual itself, i.e. this documentation.

Technically, this is a documentation made with Sphinx (https://www.sphinx-doc.org/). This allows a single reStructuredText (https://docutils.sourceforge.io/rst.html) base to be generated to interactive HTML websites as well as printable Latex PDFs.

18.1.1 About the versions

This documentation is versioned because it is improved over the time. You are currently viewing the document release v1.0-NN-ge07d716 from Apr 25, 2025.

This version string is compatible to git describe (https://git-scm.com/docs/git-describe#_examples), i.e. it reads something like `v1.2.3-NN-g1234567`. Here, `v1.2.3` refers to the actual version whereas the suffix `-NN-g1234567` is only there if this version is actually *unnamed* and follows after N commits after the versioned commit. The actual git shorthash is then suffixed after the `-g`.

18.2 List of Abbreviations

REDAC
: REConfigurable Analog Computer

Model-1
: Analog Paradigm Model-1 Computer

INT
: Integrator

INV
: Inverter

MULT
: Multiplier

DIMM
: Dual Inline Memory Module (Form factor for PCBs)

DDR
: Double Data Rate

DDA
: Digital Differential Analyzer

ACN
: Analog Compute Node

CN
: Compute Node

XBAR
: Crosspoint-switch / Crossbar-switch

AD
: Analog Digital

DA
: Digital Analog

AC
: Alternating Current

DC
: Direct Current

DC/DC
: DC-to-DC Converter (Power Supply)

DAC
: Digital to Analog Converter

MDAC
: Multiplying Digital to Analog Converter

ADC
: Analog to Digital Converter

DPT
: Digital Potentiometer

LSB
: Least Significant Bit

INL
: Integral Nonlinearity

DNL
: Differential Nonlinearity

VDD
: Drain Power Voltage

CMOS
: Complementary Metal-Oxide-Semiconductor

NMOS
: N-type Metal-Oxide-Semiconductor

PMOS
: P-type Metal-Oxide-Semiconductor

PCB
: Printed Circuit Board (Electronic Circuit Board)

IC
: Integrated Circuit or Initial Condition (depending on context)

OP
: Operating Mode

HALT
: Halting Mode

AMP
: (Operational) Amplifier

OPA
: Operational Amplifier

OPV
: Operational Amplifier

SMD
: Surface Mount Device

SOIC
: Small Outline Integrated Circuit

TSSOP
: Thin Shrink Small Outline Package

VSSOP
: Very Thin Shrink Small Outline Package

GBP
: Great Britain Pound

PT1
: First Order System with One Pole

GND
: Ground

RRO
: Rail to Rail Output

RRI
: Rail to Rail Input

CM
: Common Mode

SR
: Slewrate

GBW
: Gain Bandwidth (Product)

MUX
: Multiplexer

SPST
Single Pole Single Throw

SPDT
Single Pole Dual Throw

SPI
Serial Peripheral Interface

M-Block
Math Block

U-Block
Voltage Block

C-Block
Coefficient Block

I-Block
Current Block

T-Block
Topology Block

FDT
Feedthrough

XTALK
Crosstalk

PPV
Perturbation Projection Vector

NP
Nondeterministic Polynomial (Complexity Context)

P
Polynomial (Complexity Context)

MPDE
Multi-Time Partial Differential Equation

PDE
Partial Differential Equation

pDGL
Partial Differential Equation (German: Partielle Differentialgleichung)

DMA
Direct Memory Access

HC
Hybrid Controller

CU
Control Unit

FlexIO
Flexible Input and Output Peripheral (NXP)

GPIO
General Purpose Input and Output Pins/Ports

CLK
Clock Signal in Serial Interfaces (e.g., SPI)

MISO
Master In, Slave Out (SPI Data Line)

MOSI
Master Out, Slave In (SPI Data Line)

CS
Chip Select (SPI Signal Line)

CNVST
Conversion Start Signal of an ADC

PWM
Pulse Width Modulation

CMP
Comparator

18.3 Glossary

Backplane
A circuit board for housing and connecting carrier modules. The circuit board is the defining element of the iREDAC.

Carrier Module
A circuit board with DIMM slots and a VG strip: accommodates functional blocks and the CTRL block. This board is the defining element of the mREDAC and LUCIDAC.

Cluster
2x M-, 1x U-, 1x C-, 1x I-Block.

Carrier Module Prototype
1x Cluster and 1 CTRL-Block on a carrier module (=LUCIDAC, but not explicitly mentioned).

REDAC Carrier Module
3x Clusters and 1 CTRL-Block on a carrier module.

Functional Block
General term for M-, U-, C-, I-Block.

CTRL-Block
Control block: Interface between analog switching technology and digital computer.

Cluster
2x M-, 1x U-, 1x C-, 1x I-Block.

M-/U-/C-/I-Block
Mathematical or computing element, voltage coupling, coefficient, current coupling block.

Configuration, Circuit, Program
A data record that fully describes the analog interconnection of the REDAC. In the case of mREDAC, this primarily includes the UCI configuration, MIntConfig, and UBlockAltSignals. The

UCI configuration can be represented in different matrix-like conventions or as a list of routes. See also specializations below, such as Irreducible Configuration or Cluster Configuration.

Interconnection Configuration
(As an alternative or German equivalent to "Configuration, Circuit, Program," or at least a preferred term in the report) A specific interconnection between virtual or real computing elements. In current usage, it is unclear whether the configurations (e.g., initial values) of the elements are included, but they actually should be.

Cluster Configuration
A restriction of the configuration, e.g., to a single cluster.

Setting, Adjustments (but not: Configuration)
(Typically user-configurable runtime) settings in the Teensy that affect only the functionality of the Teensy itself and not the mREDAC configuration, i.e., the analog circuit. An example is the IP address of the Teensy.

Note: A setting is something the administrator sets. A config is something the user sets.

Lane
One of 32 lines through the UCI block: Leads from a U-block output through a DPT into an I-block input. As a data type, a lane is an integer in the range [0,31]. Also called a UCI lane.

The term "Lane Index" should be avoided; a lane always refers to an index.

Crosslane, short: Clane
One of 16 lines leading from the MBlock into the UBlock or from the IBlock back into the MBlock.

The term "clane index" should be avoided; a clane always refers to an index.

(Physical) Route
4-tuple: (uin, uout, cval, iout), where uin and iout are crosslanes and uout is a lane. cval is a coefficient value (float in the range [-20,+20]). There are a maximum of 32 routes in the mREDAC.

The iout value in a route is optional, as it can also lead to a virtual output (see below).

Compute Element, Element Description
A type description of a computing element, i.e., it describes the name and inputs/outputs but not the internal structure or behavior.

Element Name
By convention, the following computing element names are used in mREDAC: Mul, Int, Daq, Extout, Extin, Const, Pot.

Physical Compute Element
A computing element in the M-Block or C-Block, currently Int, Mult, and Coeff. The term refers to an abstract element, i.e., its type (structure/behavior) and not where it is located in the computer.

Virtual Compute Element
A computing element that cannot be physically localized cleanly, currently including External Inputs, External Outputs, DAQ Outputs, Constant Generators, and (arguably) Summators.

Stateful Compute Element
In general, compute elements are stateless. In mREDAC, only integrators and potentiometers have an internal state, and their types contain information such as IC, k0, or coeff-value.

Source, Output Compute Element
A monadic compute element with no output, i.e., a compute element with an empty input-

port list and an output-port list containing only one element, conventionally called “source.” In mREDAC, these include External Input elements or constants.

All source elements in mREDAC are necessarily virtual computing elements.

Sink, Input Compute Element
The opposite of a source. In mREDAC, these are, for example, DAQ Output elements.

All sink elements in mREDAC are necessarily virtual computing elements.

Compute Element Port
Each computing element has a list of inputs and outputs. Entries in this list are called “ports.” By convention, monadic compute elements have only an “in” and an “out” port. Dyadic compute elements conventionally have “a” and “b” ports as outputs.

Assigned (Physical/Virtual) Compute Element
A computing element localized in the system: Typically, computing elements can be described by an index (in mREDAC) or by a hierarchical device-tree-like structure (in REDAC).

Virtual/Physical Assigned Compute Element Port
A combination of the above concepts, meaning a localized port of a specific computing element in the system. This concept also applies to virtual computing elements, which are simply numbered.

Virtual Route, also: Logical Connection
An unrouted route that connects two virtual (typically assigned) compute element ports.

Logical Route
A virtual route after Pick & Place, meaning it has, in the case of mREDAC, one or more candidate lanes assigned.

Auxiliary Block Configuration
Something like U-Block-Alt-Signals or MInt-Block configurations.

Irreducible Configuration
Description of the UCI matrix with as few degrees of freedom/variants as possible. This is either a description as a route list or as three U/C/I lists, all of which are input-centric.

Input-Centric Configuration, Output-Centric Configuration
Describes different ways to represent (linearize) UCI matrices. In the U-Block, the adjacency list of the U-Matrix is simultaneously an input-centric configuration and thus irreducible, as it is a simple list. In the I-Block, it is an output-centric configuration and thus a list of lists that can be represented as an input-centric one.

MBlock Setup
Describes which type of MBlocks sit in which MBlock slots. In the simplest case, this is a mapping of slot indices/coordinates to MBlock types.

Routing Error
Description of a result of a compilation step.

Physical Routing
Result of the P&R compilation step. The result includes not only physical routes but also a list of potential routing errors and the U-Alt-Block allocation.

SendEnvelope and RecvEnvelope
Generic envelope types for the JSON protocol. They always have an id (string, should be a UUID), a message type (string, from a list of supported messages), and a message type. A JSON schema should exist that covers all possible envelopes. The RecvEnvelope is similar.

Serialized Program, Exported Configuration
: Dump of the data structure of a configuration in an appropriate data format such as JSON/YAML/etc. This should also include version information for reloading.

Structure Description (!= Hardware Description)
: The description of the fundamental mathematical structure of an analog computer. In REDAC, for example, this includes the fact that signals pass through a coefficient and are then summed, meaning only fully multiplied terms can be computed, such as *a*x + a*y* but not *a*(x+y)*. The structure description of an analog computer is static and does not change.

  Required for the synthesis step of the compiler.

Behavioral Description (!= Hardware Description)
: The description of the behavior of the analog computing elements and thus the analog computer. This corresponds to the mathematical modeling of a computing element. For example, an integrator is described by the temporal integral over its input signal. The behavioral description of a computing element is static and does not change.

Hardware Description (= Resource Description, = Hardware Configuration)
: The list of all available computing elements (potentially implicitly defined by MBlock types) and the available interconnections (potentially implicitly defined by types and positions on backplanes, etc.).

LIST OF FIGURES

LIST OF TABLES

EU-Konformitätserklärung

gemäß der Richtlinie 2011/65/EU (RoHS) vom 8.07.2011

CE

Nr: *AN-4-170000-194884*

Hersteller/Bevollmächtigter 1):

anabrid GmbH
Am Stadtpark 3
D-12167 Berlin
E-Mail.: office@anabrid.com

Die alleinige Verantwortung für die Ausstellung dieser Konformitätserklärung trägt der Hersteller (bzw. Installationsbetrieb): anabrid GmbH

Gegenstand der Erklärung: *REDAC - digital programmierbarer Analogcomputer*

Der oben beschriebene Gegenstand der Erklärung erfüllt die Vorschriften der Richtlinie 2011/65/EU des Europäischen Parlaments und des Rates vom 8. Juni 2011 zur Beschränkung der Verwendung bestimmter gefährlicher Stoffe in Elektro- und Elektronikgeräten.

Angewandte harmonisierte Normen insbesondere:

- EN 55032 (Funkstörungen)
- EN 61000 (Netzstörungen)

Angewandte sonstige technische Normen und Spezifikationen:

- RoHS-Richtlinie 2011/65/EU
- EMV-Richtlinie 2014/30/EU

Unterzeichnet für und im Namen von: anabrid GmbH

Ort/Datum der Ausstellung: Berlin / 27.01.2025

Angabe zur Person des Unterzeichners:

Lars Heimann, Geschäftsführer anabrid GmbH
(Name, Position)

Unterschrift:……

anabrid
anabrid GmbH
Am Stadtpark 3
12167 Berlin
www.anabrid.com

1) „Bevollmächtigter“ ist jede in der Union ansässige natürliche oder juristische Person, die von einem Hersteller schriftlich beauftragt wurde, in seinem Namen bestimmte Aufgaben wahrzunehmen;

BIBLIOGRAPHY

[ulmannAP2] Bernd Ulmann: Analog and Hybrid Computer Programming, De Gruyter, ISBN 9783110662207, https://doi.org/10.1515/9783110662207 or amazong book listing (https://www.amazon.de/dp/B0893VHTHL?ref_=cm_sw_r_kb_dp_zy7uFbNZ5HRN9&tag=httpwwwvaxn 21&linkCode=kpe)

INDEX