

# Status of the Optoboard microservice

19<sup>th</sup> March 2024

Daniele Dal Santo

on behalf of the Bern ATLAS Team

*u<sup>b</sup>*

---

<sup>b</sup>  
**UNIVERSITÄT  
BERN**

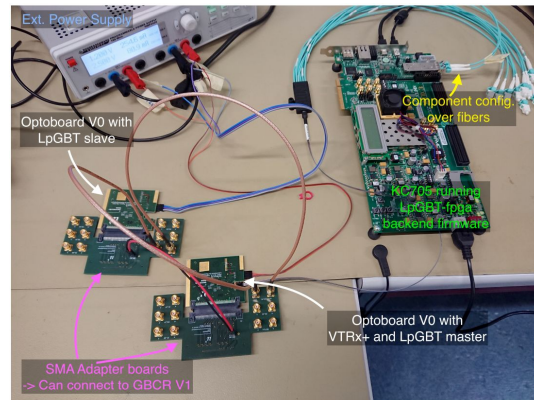
**AEC**  
ALBERT EINSTEIN CENTER  
FOR FUNDAMENTAL PHYSICS



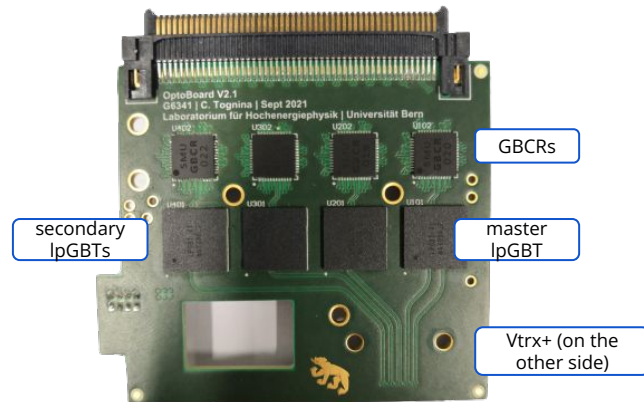
# Recap of the Previous Update

1. Original software developed for configuring single IpGBT and VTRx+ on a Optoboard V0
  - a. no FELIX cards available at the time, KC705 FPGA used with custom firmware
  - b. supported communication tool: *flpgbtconf*
2. Optoboard V1: include slave ASICs → use IpGBT master I<sup>2</sup>C controller
  - a. expanded original scripts
3. Optoboard > V2.1: increased number of users  
optoboard\_felix restructured with a more modular structure
  - a. introduction of *InitOpto* CLI entrypoint
  - b. added useful ASIC specific methods (see [here](#))
  - c. added support for IC communication over *felixcore* via *ic-over-netio*

[Latest update \(01.02.23\)](#) on the Optoboard software by Roman Müller



setup from 2019



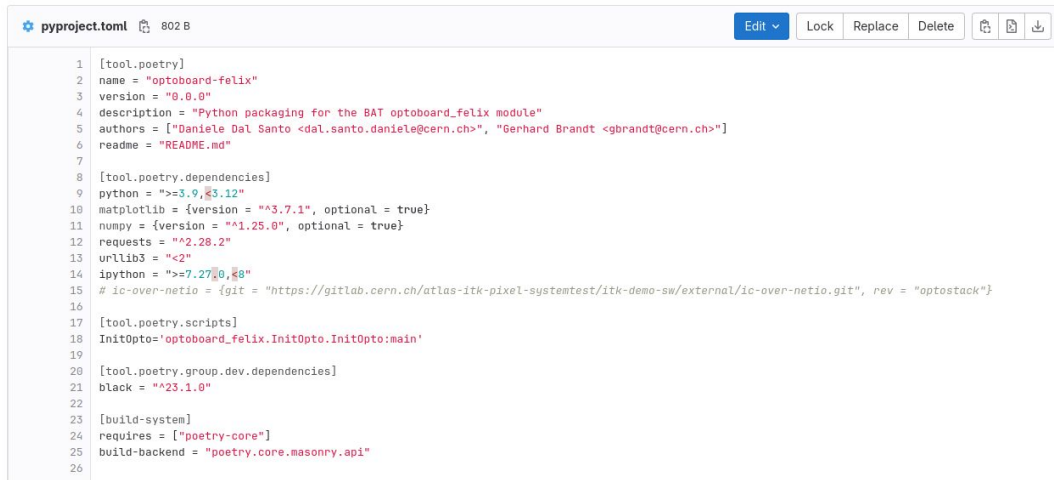
optoboard felix reorganized as pip installable package

On repository tags:

- publish package to [Package Registry](#)
- automatic [software documentation](#) update

*pip install optoboard-felix --index-url https://gitlab.cern.ch/api/v4/projects/113584/packages/pypi/simple*

Latest tag: 1.0.34

A screenshot of a code editor showing a pyproject.toml file. The file is 802 B in size and has a blue 'pyproject.toml' icon in the top left. The editor has a light blue background and a dark blue sidebar on the left. The code is in a light blue font. The code defines a package named 'optoboard-felix' with version '0.0.0'. It includes a description, authors, and a README. It also lists dependencies for python, matplotlib, numpy, requests, urllib3, and ipython. There are also scripts and dev dependencies defined. The code is as follows:

```
1 [tool.poetry]
2 name = "optoboard-felix"
3 version = "0.0.0"
4 description = "Python packaging for the BAT optoboard_felix module"
5 authors = ["Daniele Dal Santo <dal.santo.daniele@cern.ch>", "Gerhard Brandt <gbrandt@cern.ch>"]
6 readme = "README.md"
7
8 [tool.poetry.dependencies]
9 python = ">=3.9, <3.12"
10 matplotlib = {version = "3.7.1", optional = true}
11 numpy = {version = "1.25.0", optional = true}
12 requests = "2.28.2"
13 urllib3 = "2"
14 ipython = "7.27.0, <8"
15 # ic-over-netio = {git = "https://gitlab.cern.ch/atlas-itk-pixel-systemtest/itk-demo-sw/external/ic-over-netio.git", rev = "optostack"}
16
17 [tool.poetry.scripts]
18 InitOpto = 'optoboard_felix.InitOpto.InitOpto:main'
19
20 [tool.poetry.group.dev.dependencies]
21 black = "23.1.0"
22
23 [build-system]
24 requires = ["poetry-core"]
25 build-backend = "poetry.core.masonry.api"
26
```

*InitOpto* shim to run [InitOpto.py](#) and initialize optoboard instances

```
InitOpto [-h]
  [-P/--optoListPath OPTOLISTPATH]
  [-p/--config_path CONFIG_PATH]
  [-s/--optoboard_serial OPTOBOARD_SERIAL]
  [-v/--vtrx_v {1.2,1.3}]
  [-θ/--flx_G {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23}]
  [-d/--flx_d FLX_D]
  [-T/--commToolName {"flpgbtconf","flpgbtconfAPI","ic-over-netio","itk-ic-over-netio-next","lpgbt-com"}]
  [--woflxcore {0,1}]
  [-c/--configure {0,1}]
  [-D/--debug {0,1}]
  [-t/--test_mode {0,1}]
  [--configInDB {0,1}]
  [--CONFIGDB_ADDRESS CONFIGDB_ADDRESS]
  [-i]
```

1 OB instance,  
'opto'

```
InitOpto --optoboard_serial 4400072 --vtrx_v 1.3 --flx_G 0 --flx_d 0 -i --configure 1
>>> opto.lpgbt1.check_PUSM_status()
>>> opto.lpgbt1.swap_rx_polarity(0)
>>> opto.lpgbt1.swap_tx_polarity(0)
```

OB instances,  
'optoList'

```
InitOpto --optoListPath optoConfigList_example.json -i
```

optoConfigList\_example.json 459 B

```
1 {
2   "OB0": {
3     "optoboard_serial": "30000000",
4     "vtrx_v": "1.3",
5     "flx_G": 0,
6     "flx_d": 0,
7     "configure": 0,
8     "test_mode": 1,
9     "config_path": "",
10    "debug": 0,
11    "configInDB": 0,
12    "commToolName": ""
13  },
14  "OB1": {
15    "optoboard_serial": "33000000",
16    "vtrx_v": "1.3",
17    "flx_G": 0,
18    "flx_d": 0,
19    "configure": 0,
20    "test_mode": 1,
21    "config_path": "",
22    "debug": 0,
23    "configInDB": 0,
24    "commToolName": ""
25  }
26 }
27
```

*ConfigGUI.json* has  
this structure

# optoboard-felix: InitOpto configuration

*InitOpto* requires:

- configuration of IC communication tool in [CommConfig.py](#)

```
TX_PORT = int(os.environ["TX_PORT"]) if "TX_PORT" in os.environ else 12340
RX_PORT = int(os.environ["RX_PORT"]) if "RX_PORT" in os.environ else 12350
TX_TAG = int(os.environ["TX_TAG"]) if "TX_TAG" in os.environ else 17
RX_TAG = int(os.environ["RX_TAG"]) if "RX_TAG" in os.environ else 29
INTERFACE = str(os.environ["INTERFACE"]) if "INTERFACE" in os.environ else "127.0.0.1"
FELIX_BUS = str(os.environ["FELIX_BUS"]) if "FELIX_BUS" in os.environ else "/bus"
```

```
CommConfig = { "IChandler_config": { "ip": INTERFACE, "tx_port": TX_PORT, "rx_port": RX_PORT, "tx_tag": TX_TAG, "rx_tag": RX_TAG },
               "lpgbt_to_felixStar_config": { "interface": INTERFACE, "bus_dir": FELIX_BUS, "tx_tag": TX_TAG, "rx_tag": RX_TAG }, }
```

both felixcore and  
felixstar supported

- knowledge of composition of each Optoboard, given its serial number
  - [components.json](#) created from [productionDB](#)

```
"4200042": {
  "optoboard_v": 4,
  "lpgbt_v": 1,
  "lpgbt_master_addr": 116,
  "lpgbt1": 1,
  "lpgbt2": 0,
  "lpgbt3": 1,
  "lpgbt4": 0,
  "gbcv_v": 2,
  "gbc1": 1,
  "gbc2": 0,
  "gbc3": 1,
  "gbc4": 0,
  "efused": 0,
  "I2C_master": 0
},
```

- IpGBT: rx/tx polarities & (phase)
- GBCR: equalization strength

- soft error counter and scan
- parallel softerror counter
- (rx,tx) scan
- swap rx and tx polarity
- set gbcr equalization
- phase training
- phase BERT scan (soon available)
- ...

needs to be added to  
custom config files

6

Refine IC communication library for *felixStar*:

- move communication to the slaves ASICs to C++
- reorganize the read/write commands to target fields instead of registers

# itk-demo-optoboard-api: image

Latest tag: *demi-2*

itk-demo-optoboard-api

- flask and gunicorn (API server)
- celery (distributed task queue)

opto-base-image: *almalinux:9-base*

- *felix-05-00-03-rm5-stand-alone*
- optoboard-felix
- ic-over-netio for *felixcore*
- lpGBT-com-next for *felixStar*

At container startup:

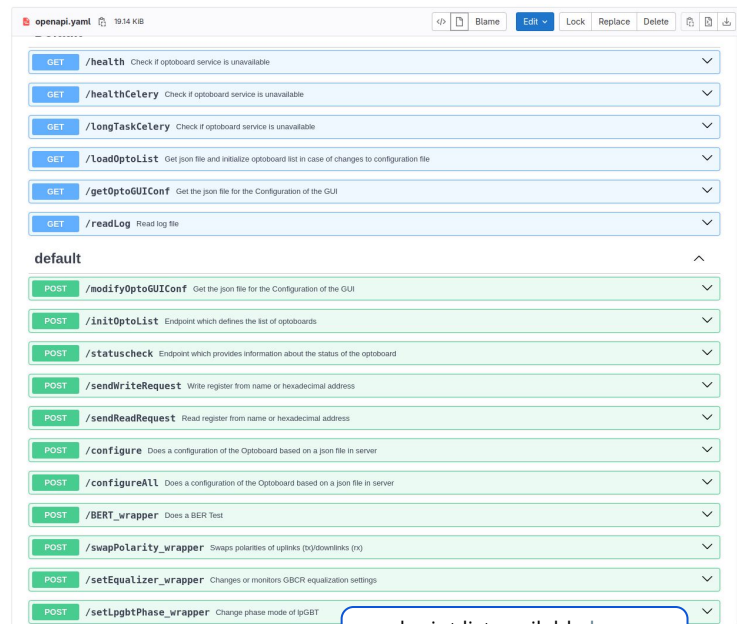
- clone user and group ID
- possible to start API server or celery WORKER based on env variable



# optoboard-felix: usage

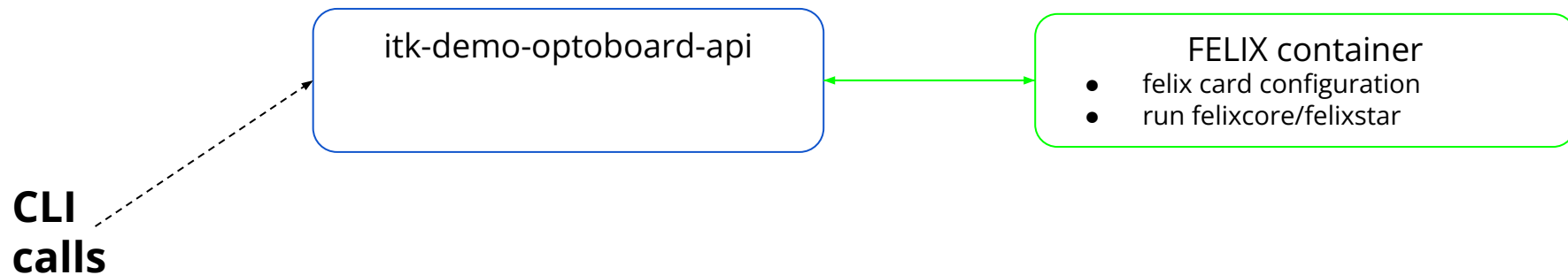
Usage of *optoboard-felix* (running Felix container is assumed):

- **call *InitOpto*** in shell attached to itk-demo-optoboard-api container
- **call API endpoints** (requires API, WORKER & rabbitmq)
  - OB instances initialized based on *ConfigGUI.json*, mounted on worker
  - celery for task queuing and asynchronous execution → currently 1 queue per OB in an Optobox
    - individual queues controlled via flower
    - parallel OB configuration possible!
- **UI ...**

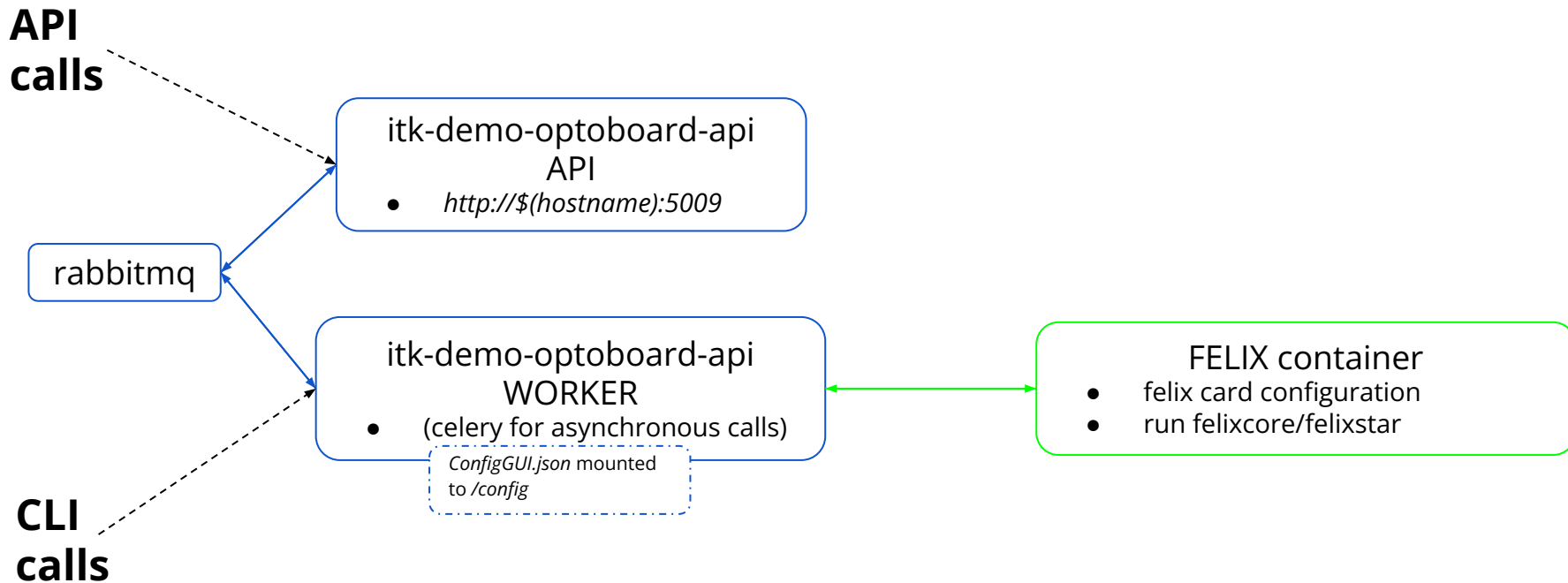


endpoint list available [here](http://localhost:5009/ui) or  
on <http://localhost:5009/ui>

# opto-stack: structure and entrypoints



# opto-stack: structure and entrypoints



Custom tests can be easily designed by calling DCS, Felix, Optoboard, etc. APIs

```
def configureOB(OB: int):
    headers = {
        'accept': 'application/json',
        'Content-Type': 'application/json',
    }

    json_data = {
        'activeGbcrr': 'None',
        'activeLpgbt': 'None',
        'optoboardPosition': 'OB'+str(OB),
    }

    response = requests.post(OPTO_API+'/'+'optoboard/api/configure', headers=headers, json=json_data)
    if response.status_code != 200:
        print("configureOB"+str(OB)+ " failed")
    response_text = response.text.strip('\n ')
    if response_text == "Optoboard configuration failed":
        print("Optoboard "+str(OB)+ " configuration failed")
```

```
def getLinkAlignment(link = None):
    headers = {
        'accept': 'application/json',
    }

    response = requests.get(FELIX_API+'/'+'getlink', headers=headers)
    if response.status_code != 200:
        print("getLinkAlignment failed")
        return

    linkalignment = response.json()
    if link:
        return linkalignment[str(link)]
    else:
        return linkalignment
```

[L. Flores' script to test multiple OBs using API calls](#)

- Image created in 2 steps:
  - *node:16.3.0* image to build JSX project
  - *nginx:stable-alpine* to serve the web content
- Web UI available on *http://\$(hostname --fqdn):5089*
- One panel per OB, based on *ConfigGUI.json* mounted to */config* in the worker

## Latest tag: *demi-2*

Optoboard GUI

Configure all Optoboards Edit GUI Configuration SRI Health Celery health

### OBO

Select device:   Read back:

Select a register:   Read back:

Select a register:  Value to be written:

BERT limit:

Uplink:  Meas Time:

Channel:

Med Frequency:  High Frequency:

Link (Rx/Tx):

Link number:

Select mode:

Group:  Phase:

Manual selection: ☐

### OBI

Select device:   Read back:

Select a register:   Read back:

Select a register:  Value to be written:

BERT limit:

Uplink:  Meas Time:

Channel:

Med Frequency:  High Frequency:

Link (Rx/Tx):

Link number:

Select mode:

Manual selection: ☐

### Edit GUI Configuration

Select default configuration: Choose Configuration\*

Basic

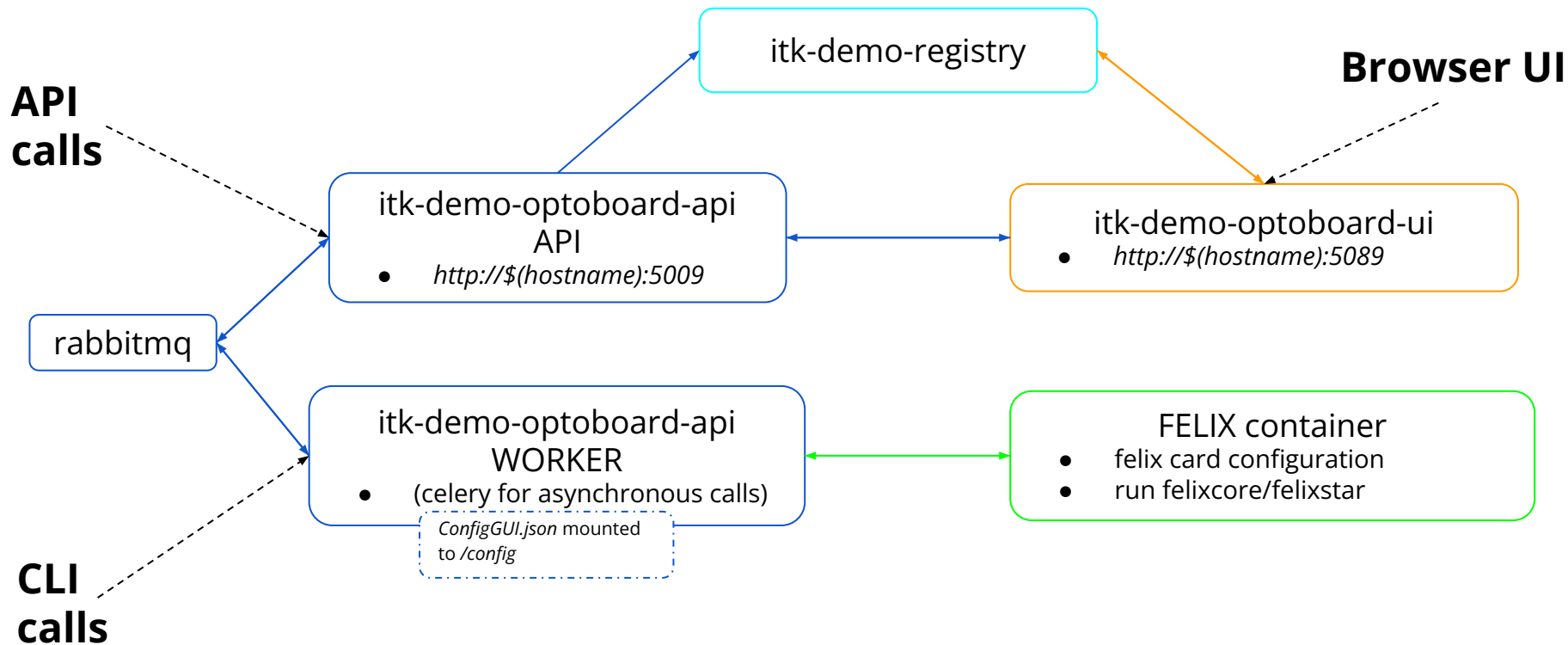
```
1 {  
2   "OBO": {  
3     "ConfigurationDB": "http://localhost:5000",  
4     "comToName": "ic-over-netio",  
5     "configInDB": false,  
6     "config_path": "",  
7     "configure": 0,  
8     "debug": false,  
9     "rtx_g": "g",  
10    "rtx_g": "g",  
11    "optoboard_serial": "44000073",  
12    "test_mode": false,  
13    "vtrx_v": "1.3",  
14  },  
15 }
```

- also possible to directly edit .json  
- will be moved to configDB

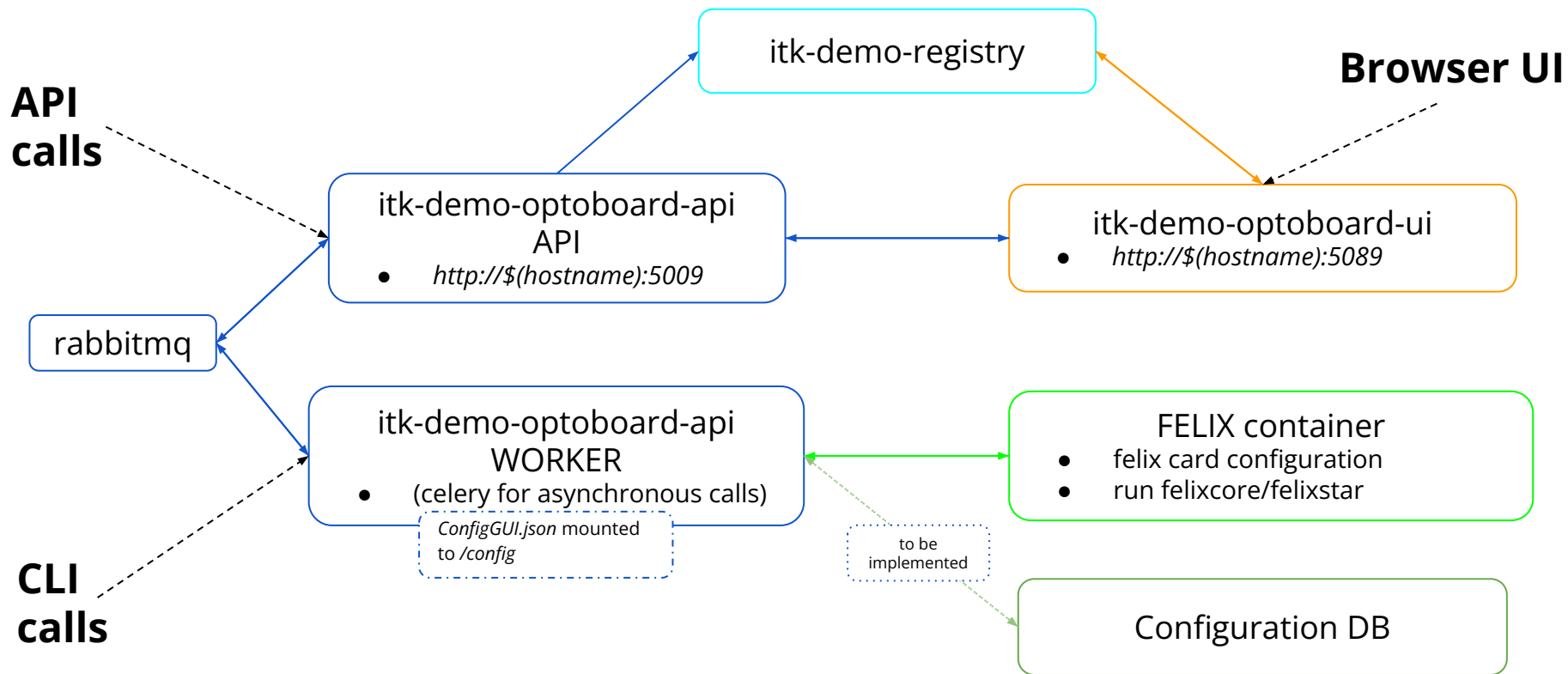
```
2023-11-29 16:17:11,083 - INFO - CommWrapper.py - Initialized IHandler with IP: 127.0.0.1, TX port: 12340, RX port: 12350, TX tag: 17, RX tag: 29  
2023-11-29 16:17:11,084 - WARNING - CommWrapper.py - IHandler.connect() failed! Check if the method is available and if the IHandler configuration is correct!  
2023-11-29 16:17:11,084 - INFO - CommWrapper.py - CommWrapper object initialised!  
2023-11-29 16:17:11,085 - INFO - Optoboard.py - Logger level at INFO  
2023-11-29 16:17:11,085 - INFO - Optoboard.py - component: OrderedDict([('optoboard_v', 3), ('lpgbt_v', 1), ('lpgbt_master_addr', 116), ('lpgbt1', 1), ('lpgbt2', 0), ('lpgbt3', 1), ('lpgbt4', 0), ('gbc_v', 2), ('gbc1', 1), ('gbc2', 0), ('gbc3', 1), ('gbc4', 0), ('efused', 0), ('I2C_master', 0)])  
2023-11-29 16:17:11,086 - INFO - Lpgbt.py - lpgbt1 object initialised!  
2023-11-29 16:17:11,087 - INFO - Lpgbt.py - lpgbt3 object initialised!  
2023-11-29 16:17:11,088 - INFO - Gbcr.py - gbcr1 object initialised!  
2023-11-29 16:17:11,088 - INFO - Gbcr.py - gbcr3 object initialised!  
2023-11-29 16:17:11,089 - INFO - Vtrx.py - vtrx object initialised!  
2023-11-29 16:17:11,089 - INFO - Optoboard.py - Optoboard object initialised!  
2023-11-29 16:17:11,089 - INFO - celeryTasks.py - Optoboard in position OBO has been added!
```

[Jump to the bottom](#)

# opto-stack: structure and entryptoints



# opto-stack: structure and entryptoints





# Deployment, outlook and useful links

2 ways (follow the instructions in the READMEs):

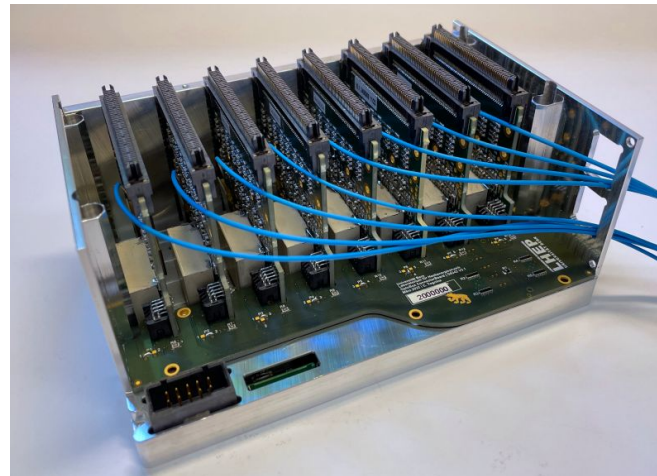
- via the stacks, using Dockge, example [here](#)
- via the itk-demo-optoboard [example folder](#)

ITk-demo-optoboard outlook:

- restructure of celery backend
- improve error handling
- ...

Useful links:

- [Bern-Optoboard](#) mattermost channel
- [Optoboard software documentation](#)
- Report problems and requests
  - [optoboard felix issues](#)
  - [itk-demo-optoboard issues](#)



# Status of the Optoboard microservice

19<sup>th</sup> March 2024

Daniele Dal Santo

on behalf of the Bern ATLAS Team

*u<sup>b</sup>*

---

<sup>b</sup>  
**UNIVERSITÄT  
BERN**

**AEC**  
ALBERT EINSTEIN CENTER  
FOR FUNDAMENTAL PHYSICS

