Expert GUI
開発状況

Rikkyo／RBRC
今井ひかる

概要

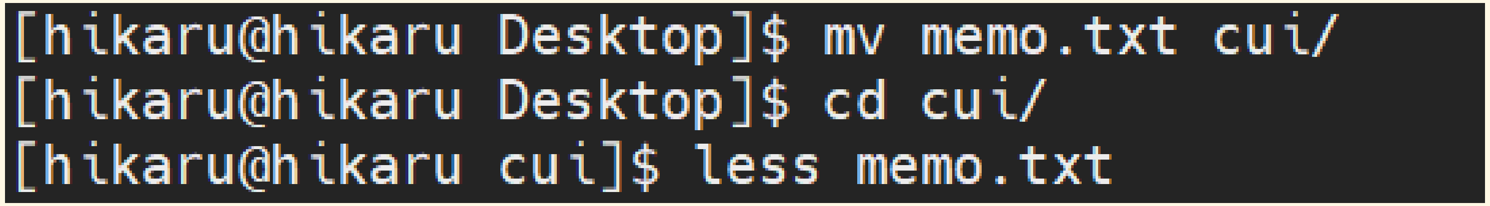
- Expert　GUIの目的について

- Expert　GUIの完成にどのようなライブラリ・技術が必要か。（DB、RPCなど）

- 現在のExpertGUIの開発状況、今後の予定について。

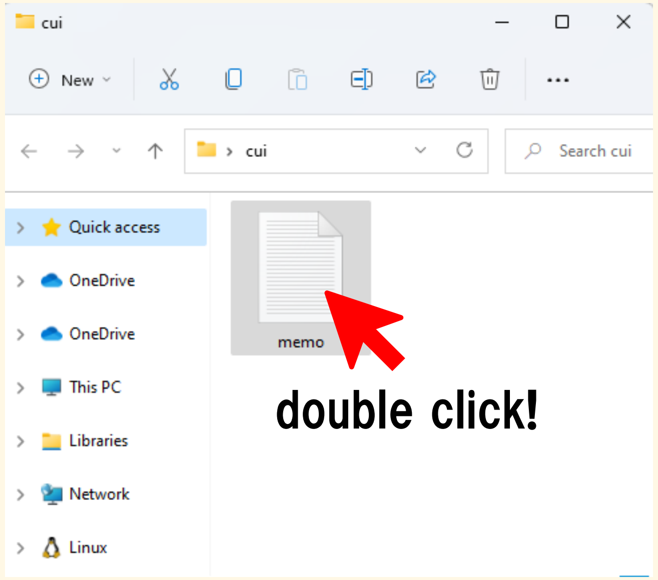
グラフィック・ユーザー・インターフェース（GUI）

ユーザー
（PCや機械に命令を送りたい人）



```
[hikaru@hikaru Desktop]$ mv memo.txt cui/
[hikaru@hikaru Desktop]$ cd cui/
[hikaru@hikaru cui]$ less memo.txt
```

コマンド・文字だけでコンピュータ・機械に、ユーザーがやりたいことを命令する

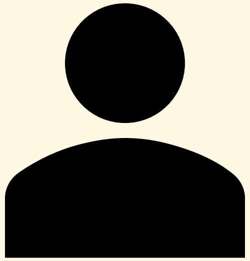


double click!

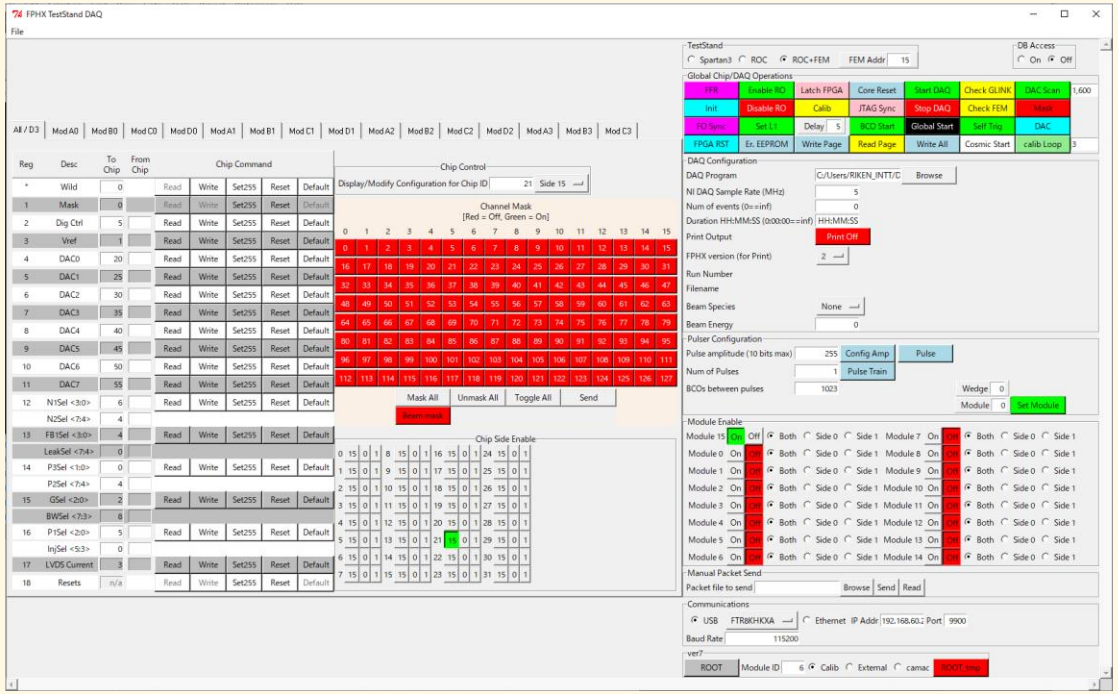
マウスやクリックを用いて、コンピュータに命令する

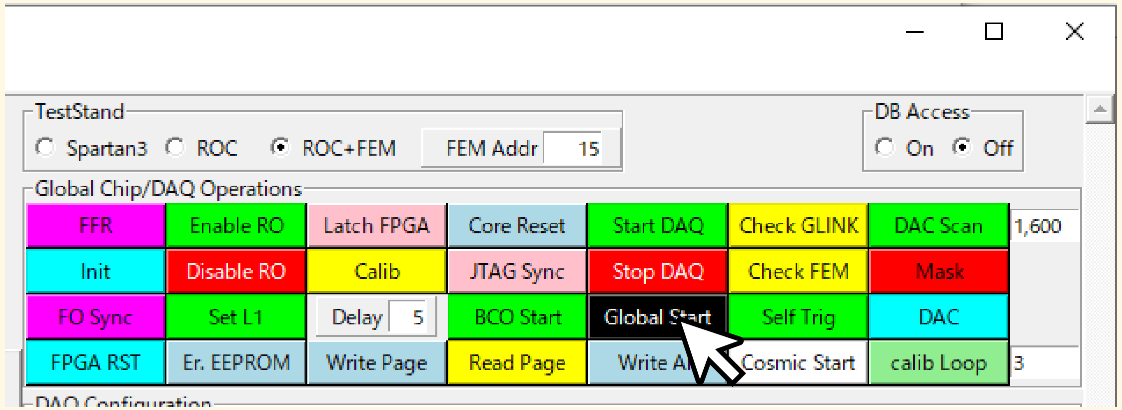
PC・機械

INTTのGUI



ユーザー





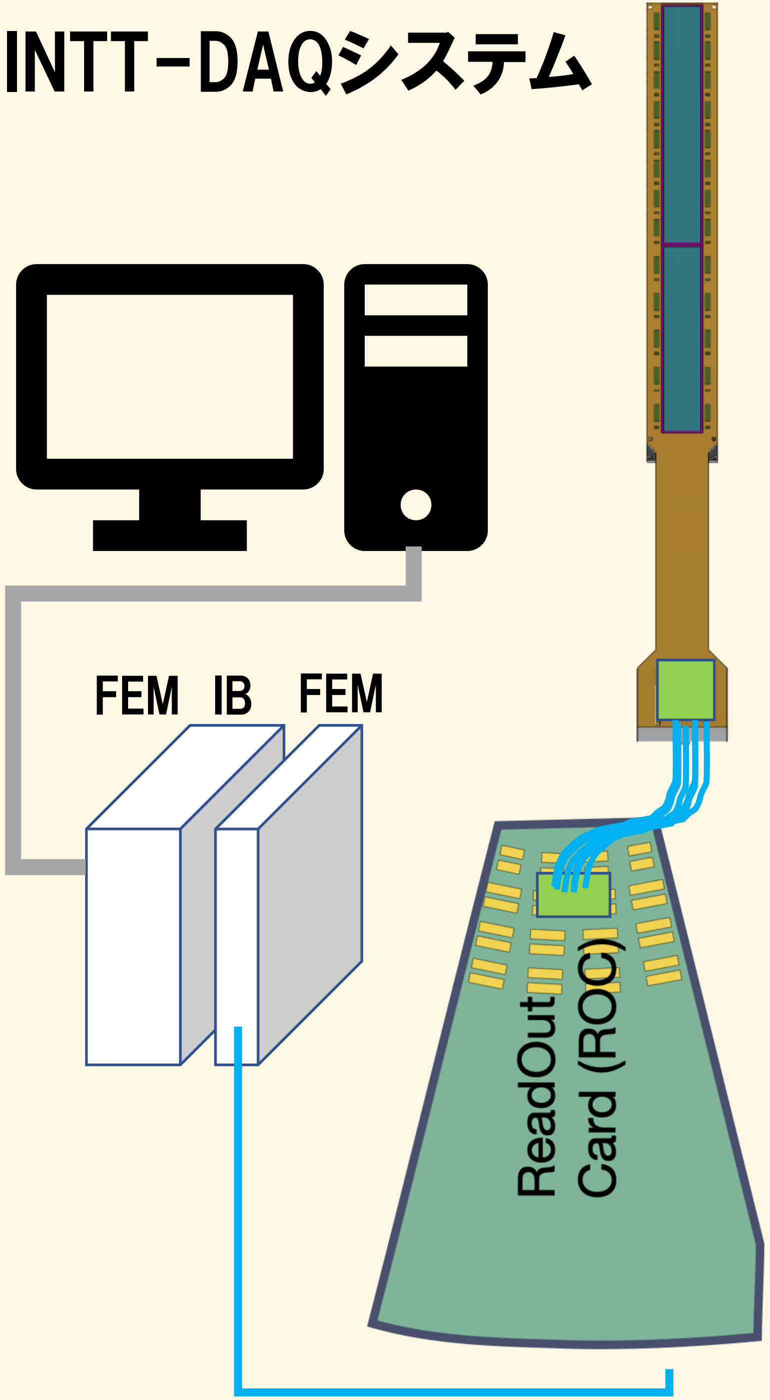
TestStand
○ Spartan3 ○ ROC ● ROC+FEM
FEM Addr 15

DB Access
○ On ● Off

Global Chip/DAQ Operations

現在のテストベンチGUI

INTT-DAQシステム



INTTのGUI

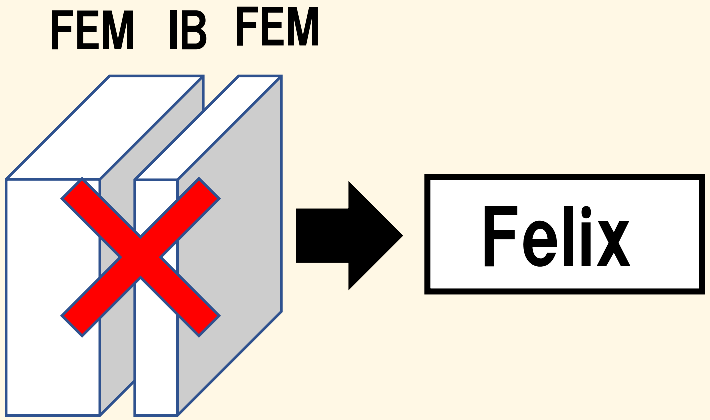
ユーザー

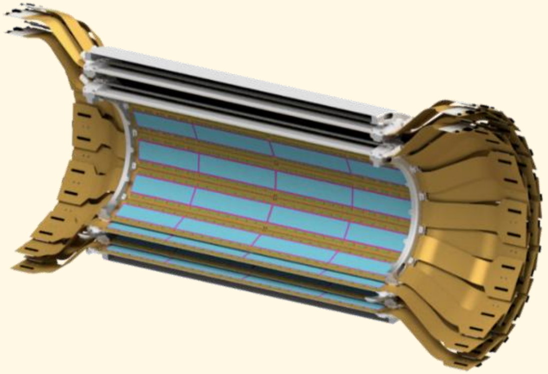
？

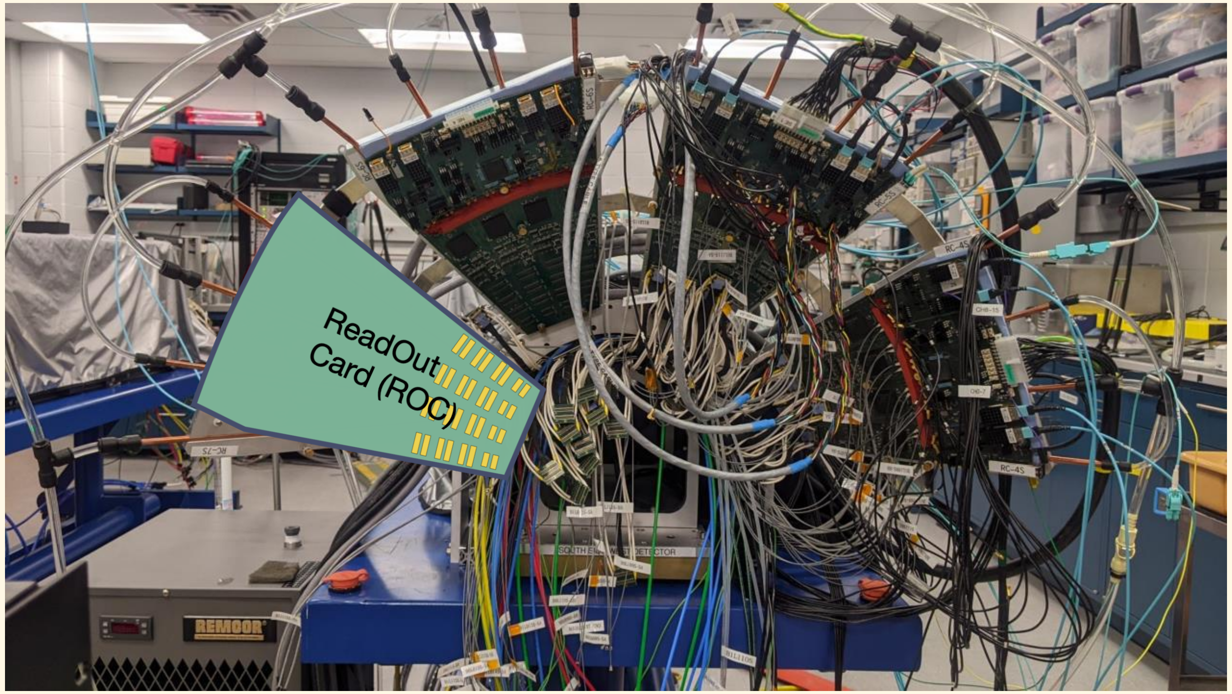
数百本のラダーと数台のROCを、Felixを通じてコントロールするGUIは、まだ無い。

→我々が開発する必要がある。

INTT-DAQシステム

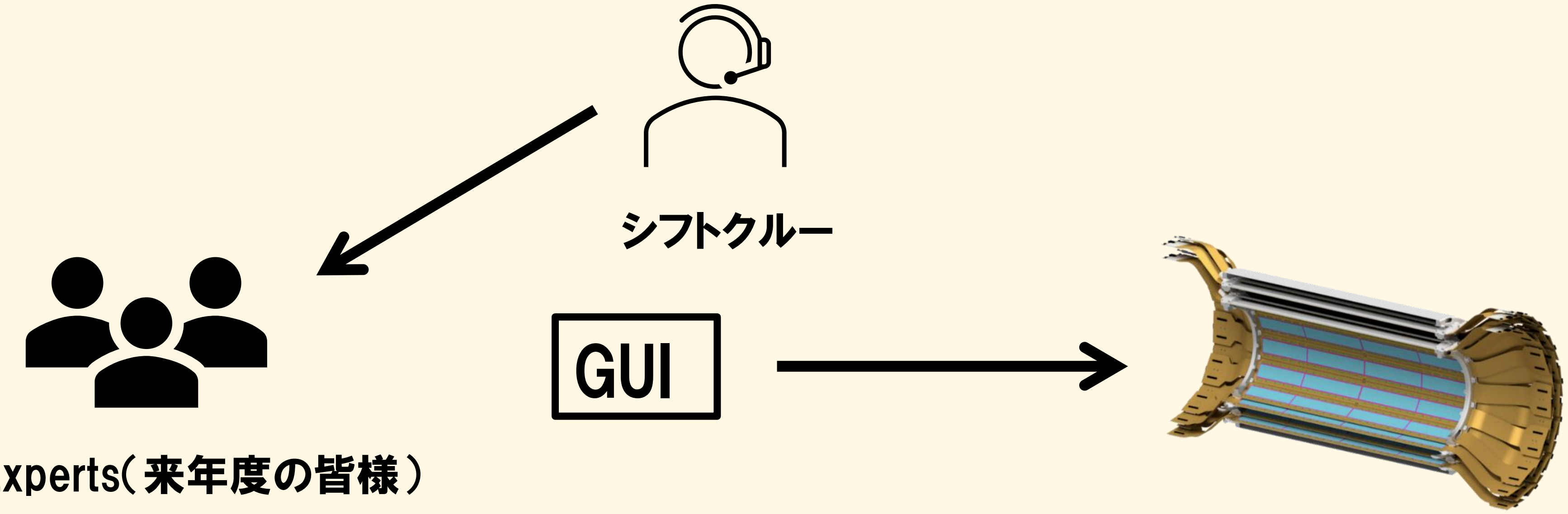






本研究の目的

- マウスのクリックなどといった**直感的な操作**でsPHENIXで使用される**数多くの**ラダー・チップ・チャンネルとROCを、**Felixを通じて**コントロールできる**GUIアプリケーション、Expert GUI**を開発する。



シフトクルー

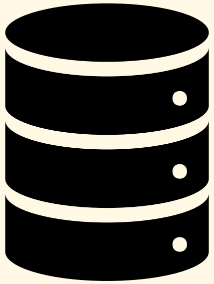
GUI

INTT Experts(来年度の皆様)

現在のラダーやROCの**状態**を**わかりやすく**GUIに表示する必要もある。

ExpertGUIに必要な技術

3. ExpertGUIがあるPCとFelixボードは
物理的に離れているため、**遠隔で**
操作する方法を確立する必要がある。



DAC値などのパラメタ更新

現在のパラメタをGUIに表示

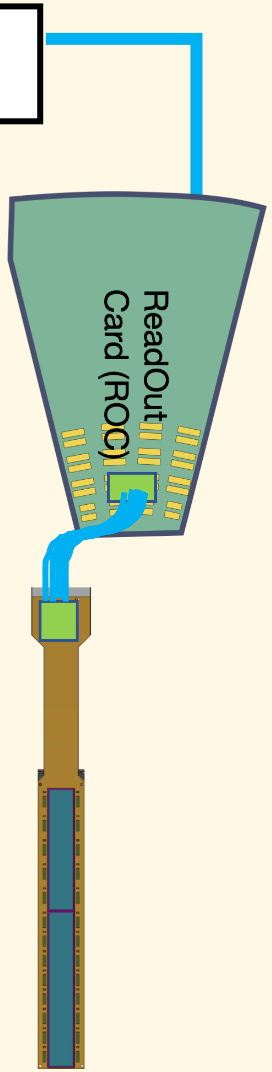
データベース（DB）

Expert
GUI

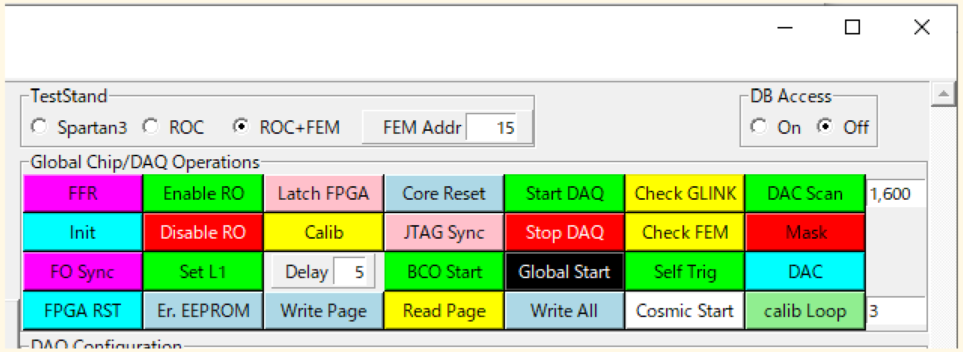
Felixに命令

応答（Read Backなど）
をGUIに表示

Felix



1. ボタンや入力欄などの設置して、
GUIの**フロントエンド**をデザインする必要がある。



TestStand: ○ Spartan3 ○ ROC ● ROC+FEM FEM Addr 15

DB Access: ○ On ● Off

Global Chip/DAQ Operations

Delay 5

1,600

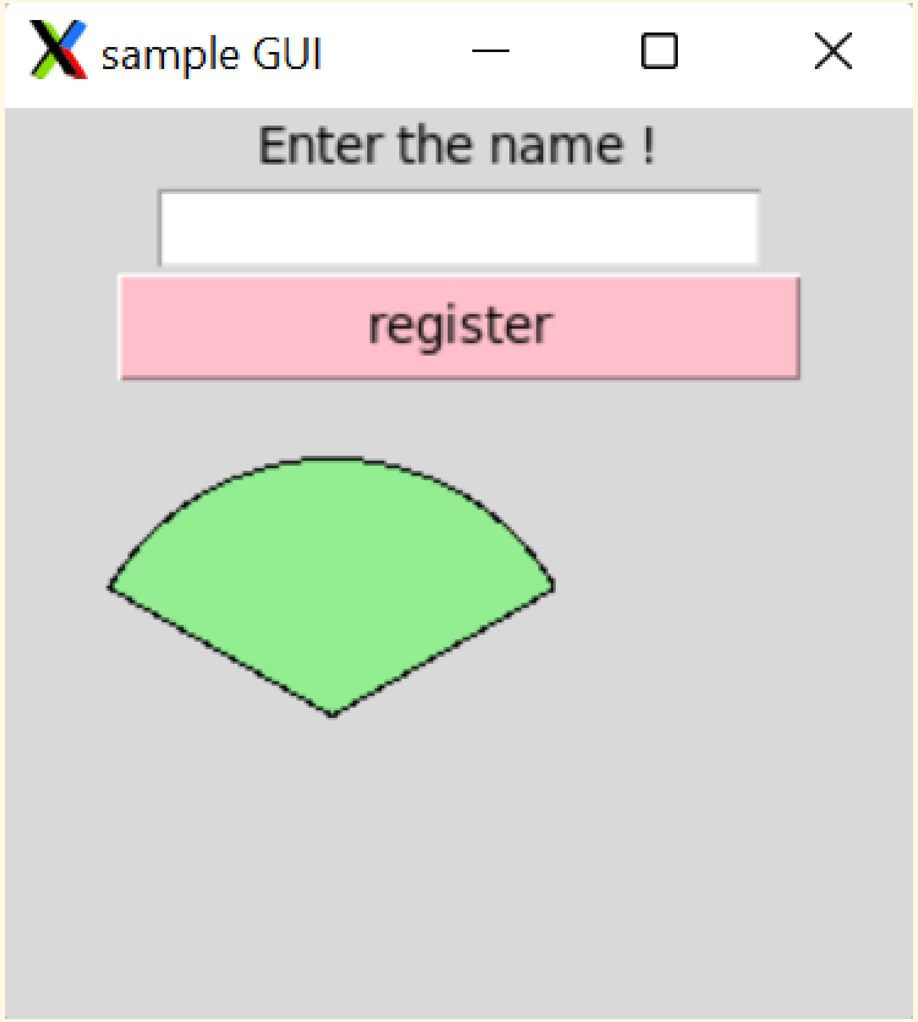
3

2. DAC値やチャンネルマスク
などのパラメタを保存する
データベースをつくる必要がある

Python3 tkinter

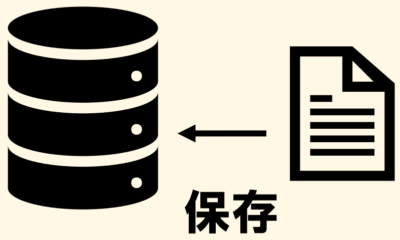
- Python3のGUI開発ツールキットであるTkinterを活用する。

```python
import tkinter as tk

# define main window
main_window = tk.Tk()
main_window.geometry("250x250")
main_window.title("sample GUI")

#########################
# set the widget ( Ex. label , entrybox , button .... )

sample_label = tk.Label(master=main_window , text= ' Enter the name ! ' )
sample_label.pack()

sample_entrybox = tk.Entry(master = main_window , width = 20  )
sample_entrybox.pack()

sample_button = tk.Button( master=main_window, width= 20 , text= "register" , bg= 'pink' )
sample_button.pack()

sample_canvas = tk.Canvas(master=main_window)
sample_canvas.create_arc(20 , 20 , 160, 160, start = 30, extent = 120 , fill = 'lightgreen' )
sample_canvas.pack()

#########################
#display main_window
main_window.mainloop()

```



データベース(PostgreSQL)

- 大切なデータを効率的、集中的かつ保守的に管理する仕組みをデータベースと呼ぶ。



- sPHENIXではデータベースを管理するシステムとしてPostgreSQLを使うように指定されている。

- PostgreSQL ではデータをテーブルにまとめ、その集合体として管理する。

名前	年齢	学年
今井	23	M2
藤木	23	B4
加藤	21	B4
宍倉	8	小2

名前	担当仕事
今井	PHENIX解析、ExpertGUI開発、DAC Scan、4年生の世話
藤木	ROCのデバッグ
加藤	マイクロ同軸の放射線耐性評価
宍倉	FakeHit 問題

どのようなデータベースをつくるべきか

- どのようなデータベースをつくるべきか → どのような**テーマ**のテーブルをつくるかべきか

1. DAC値やLVDSカレントなどのFPHXチップのパラメタをテーマにするテーブル

ladder position	direction	chip	DAC0	DAC1	LVDS
B0L000	North	chip1	15	30	3
B1L116	North	chip26	20	30	255

2. チャンネルのマスクの状態をテーマにするテーブル

ladder position	direction	chip	mask status
B0L000	North	chip1	111111111111
B1L116	North	chip26	001111111110

3. ラダーとROCの接続関係をテーマにするテーブル

（ラダーがどこのROCに、どのポートに繋がっているか）

ladder position	direction	roc position	roc name	column	port
B0L000	North	RC_01N	ROC19	A	2
B1L116	North	RC_06N	ROC21	B	1

4. ROCとFelixボードの接続関係をテーマにするテーブル

（ROCがどのFelixに繋がっているか）

Roc	Felix
RC_01N	Felix1
RC_02N	Felix1

データベースのデザイン

- それぞれのテーブルで現在の情報と過去の情報を分けて保存することにした。

present_fphx_parameters

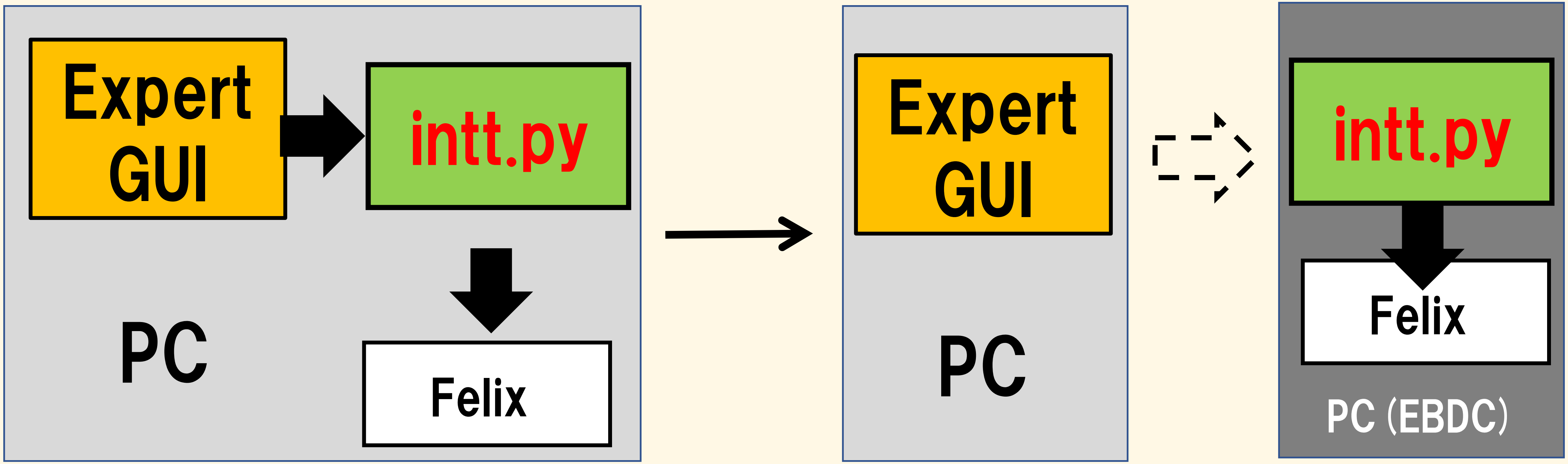
ladder position	direction	chip	DAC0	DAC1	LVDS	start time
B0L000	North	chip1	15	30	4	2023/1/23/ 9:00
B1L116	North	chip26	20	30	255	2023/1/20/ 14:00

history_fphx_parameters

ladder position	direction	chip	DAC0	DAC1	LVDS	start time
B0L000	North	chip1	15	30	3	2023/1/20/ 14:00

もちろん、これら2つのテーブルをごちゃまぜにしても構わない。
しかし、ExpertGUIは現在のDAC値などを表示するので、相性が良い気がする。

RPCについて



現在のBNLでのセットアップ

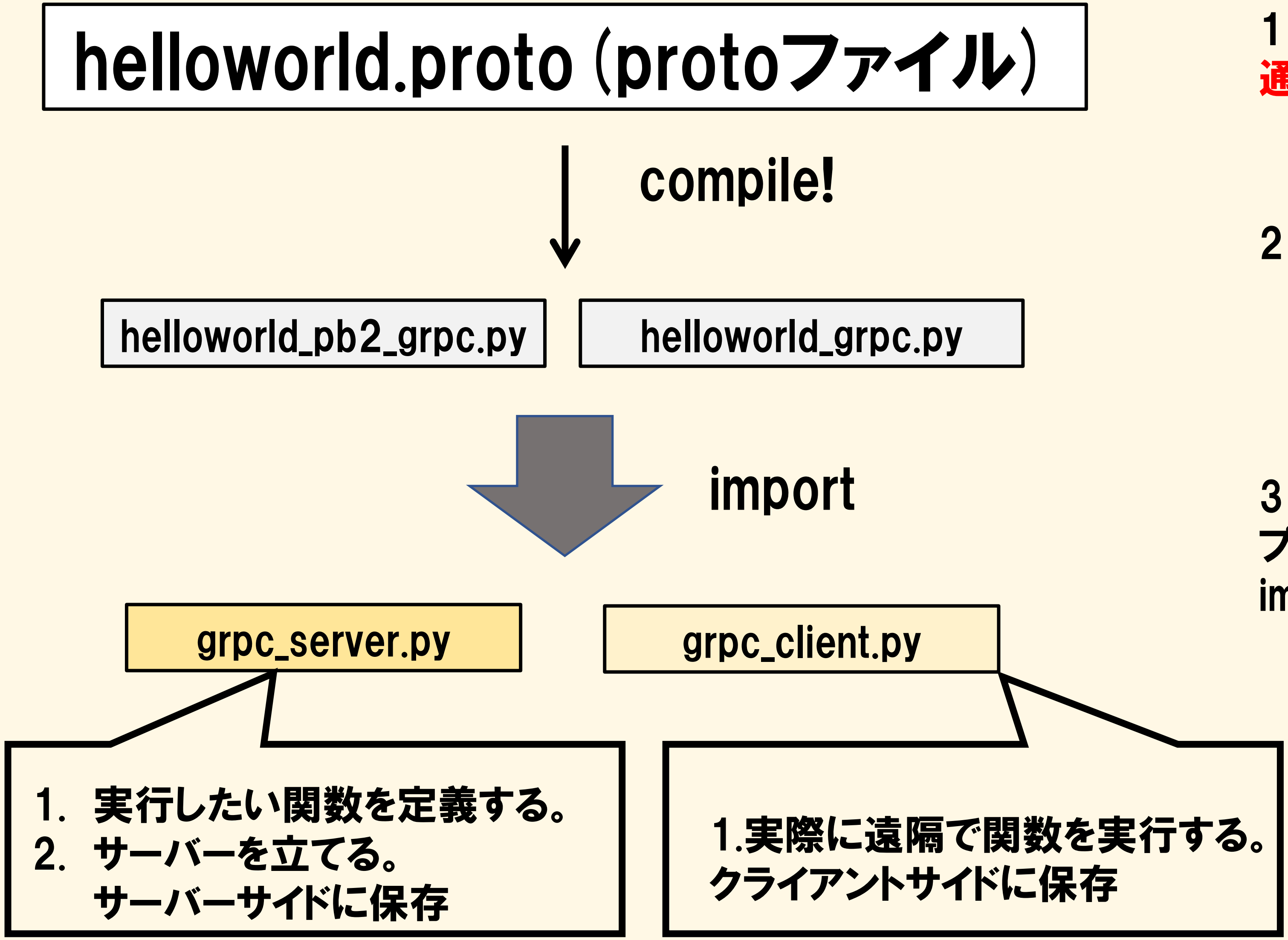
sPHENIXでのセットアップ

- ExpertGUIから、物理的に離れているPCにあるintt.pyの関数を実行する必要がある。 → 別のPCにある関数を遠隔で実行する必要がある。
- そこで、リモート・プロシージャ・コール（RPC）と呼ばれる技術を使うことで、この課題を開発する。（sPHENIXから指定は無い）

RPC （gRPC）を使う準備

gRPC公式ページ

- 今回はgRPCと呼ばれるGoogleが開発したオープンソースのRPCフレームワークを使う。



helloworld.proto（protoファイル）

compile!

helloworld_pb2_grpc.py

helloworld_grpc.py

import

grpc_server.py

grpc_client.py

1. 実行したい関数を定義する。
2. サーバーを立てる。
   サーバーサイドに保存

1.実際に遠隔で関数を実行する。
クライアントサイドに保存

1 protoファイルと呼ばれるファイルを作り、
通信のインターフェースを定義する。

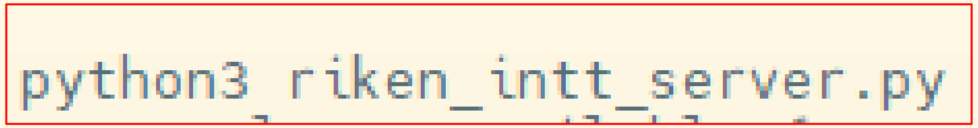
2 protoファイルをコンパイルして、2つの
　ファイルを生成する。（絶対に編集しない）

3 serverサイドとclientサイドで動かす2つの
プログラムを自分で用意し、2で生成したファイルを
importすることで、実行したい内容を実装する。

gRPCのテスト@理研

テストベンチのPC（サーバーサイド）

自分のPC（クライアントサイド）



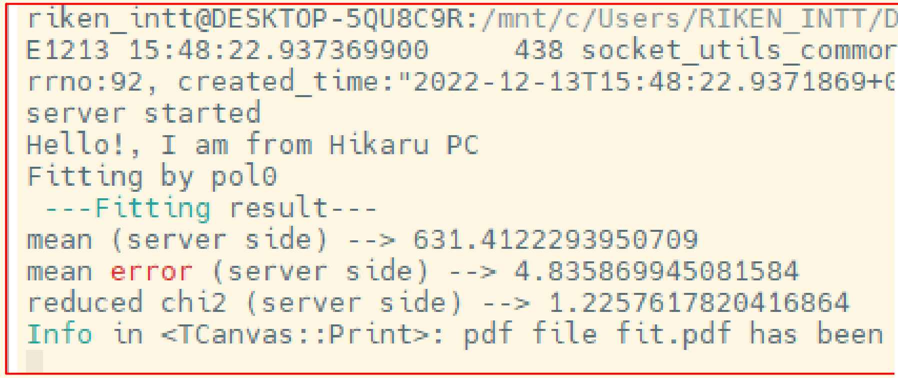
```
python3 riken_intt_server.py
```

1. gRPCサーバーを起動する。
   このテストでは、サーバーサイドにあるデータを　fitting
   してfittingパラメタを返す関数を登録

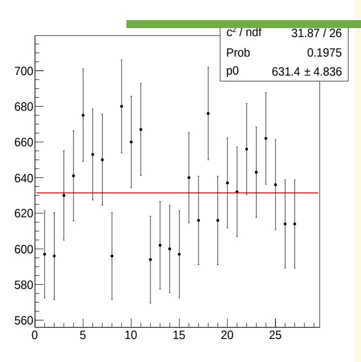
```
[hikaru@hikaru helloworld]$ python3 riken_intt_client.py
```

2. クライアント側のプログラムを実行して、
   RPCサーバーに登録されている関数を実行

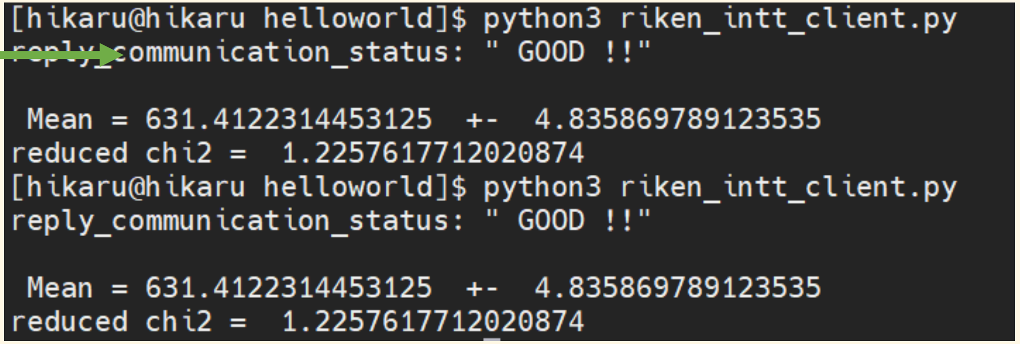
3. クライアント側から要求が来ると
   サーバー側で計算処理。



```
riken_intt@DESKTOP-5QU8C9R:/mnt/c/Users/RIKEN_INTT/Desktop/Hi
E1213 15:48:22.937369900     438 socket_utils_commor
rrno:92, created_time:"2022-12-13T15:48:22.9371869+0
server started
Hello!, I am from Hikaru PC
Fitting by pol0
 ---Fitting result---
mean (server side) --> 631.4122293950709
mean error (server side) --> 4.835869945081584
reduced chi2 (server side) --> 1.2257617820416864
Info in <TCanvas::Print>: pdf file fit.pdf has been
```



4. サーバーからの計算結果をクライアントに
   送信して、表示

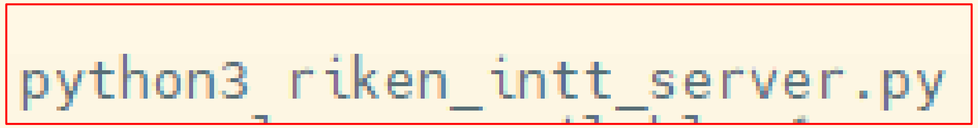


```
[hikaru@hikaru helloworld]$ python3 riken_intt_client.py
reply_communication_status: " GOOD !!"

 Mean = 631.4122314453125  +-  4.835869789123535
reduced chi2 =  1.2257617712020874
[hikaru@hikaru helloworld]$ python3 riken_intt_client.py
reply_communication_status: " GOOD !!"

 Mean = 631.4122314453125  +-  4.835869789123535
reduced chi2 =  1.2257617712020874
```

gRPCのテスト@理研

テストベンチのPC（サーバーサイド）

自分のPC（クライアントサイド）



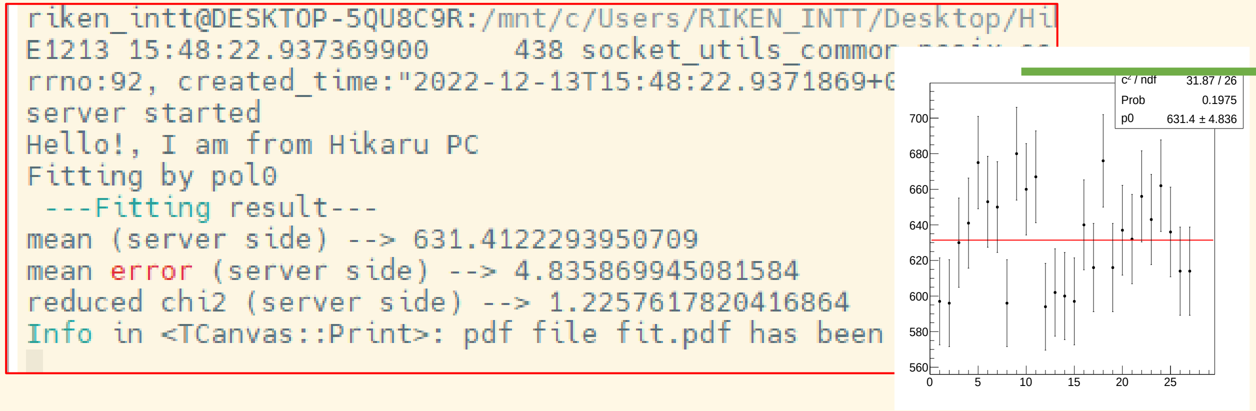
```
python3 riken_intt_server.py
```

1. gRPCサーバーを起動する。
   このテストでは、サーバーサイドにあるデータを fitting
   してfittingパラメタを返す関数を登録

```
python3 riken_intt_client.py
```

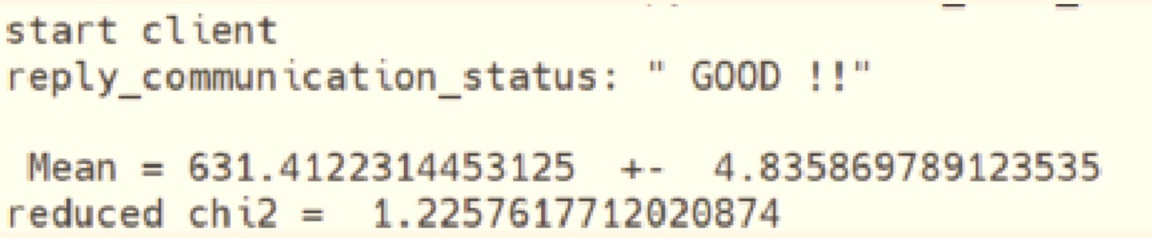
2. クライアント側のプログラムを実行して、
   RPCサーバーに登録されている関数を実行

3. クライアント側から要求が来ると
   サーバー側で計算処理。



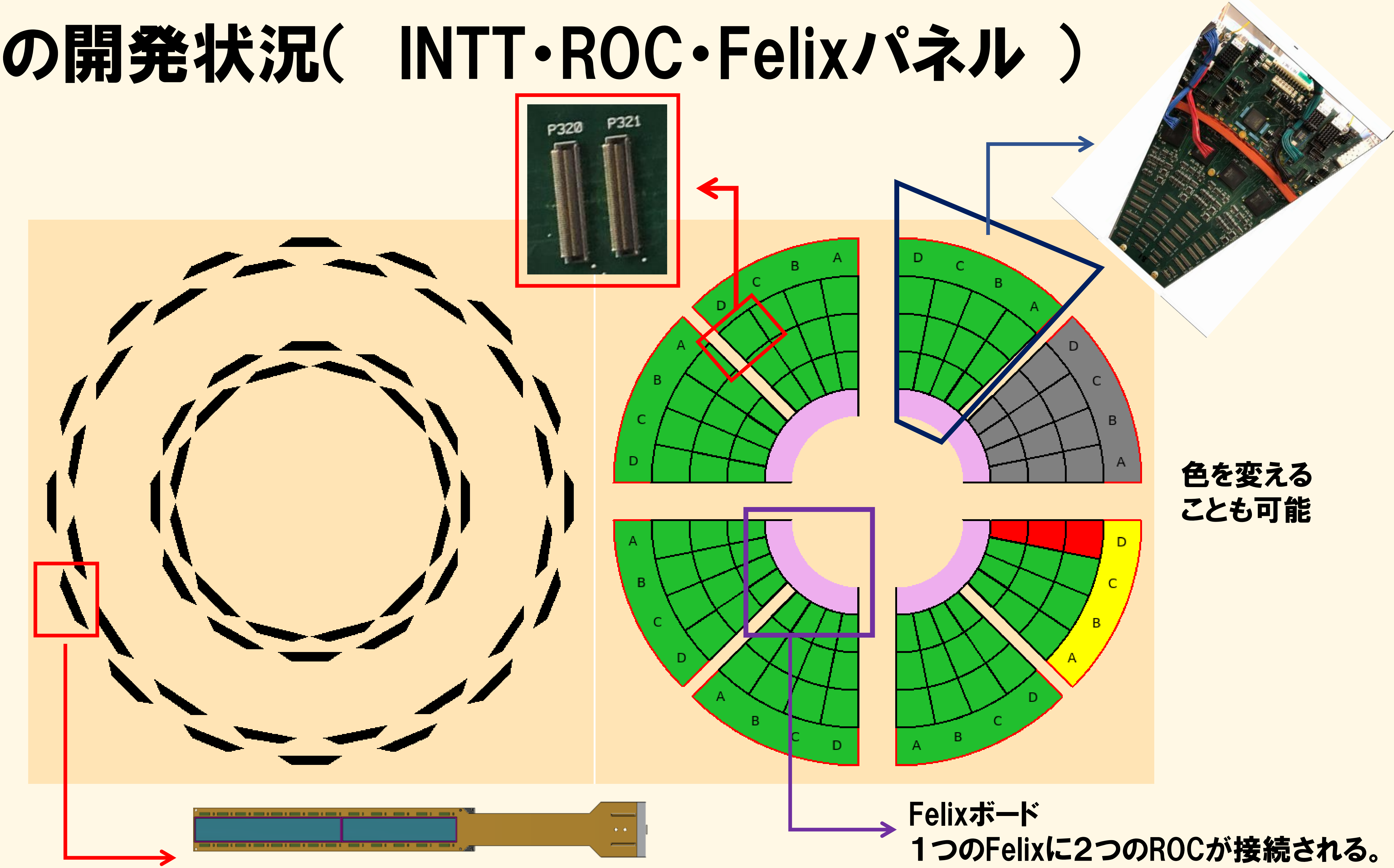
```
riken_intt@DESKTOP-5QU8C9R:/mnt/c/Users/RIKEN_INTT/Desktop/Hi
E1213 15:48:22.937369900     438 socket_utils_common
rrno:92, created_time:"2022-12-13T15:48:22.9371869+0
server started
Hello!, I am from Hikaru PC
Fitting by pol0
 ---Fitting result---
mean (server side) --> 631.4122293950709
mean error (server side) --> 4.835869945081584
reduced chi2 (server side) --> 1.2257617820416864
Info in <TCanvas::Print>: pdf file fit.pdf has been
```

4. サーバーからの計算結果をクライアントに
   送信して、表示



```
start client
reply_communication_status: " GOOD !!"

 Mean = 631.4122314453125  +-  4.835869789123535
reduced chi2 =  1.2257617712020874
```

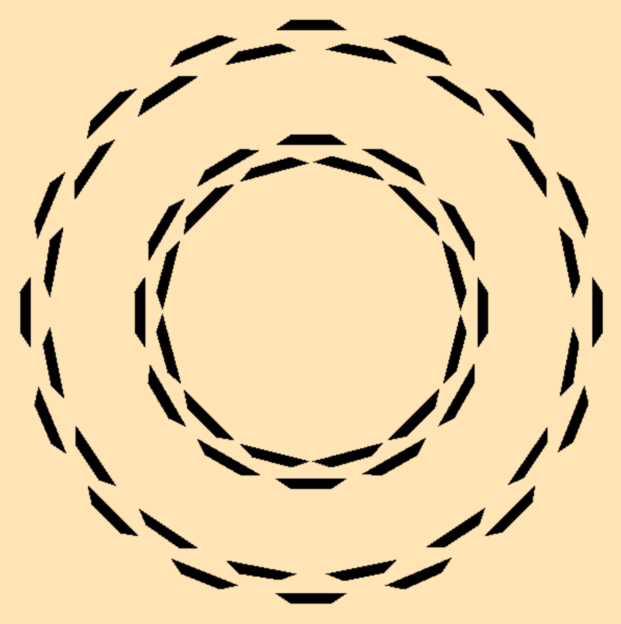
現在の開発状況（ INTT・ROC・Felixパネル ）

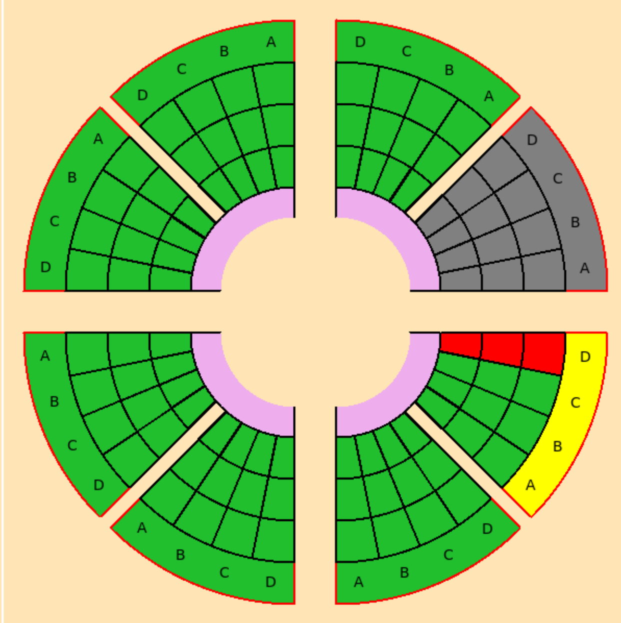


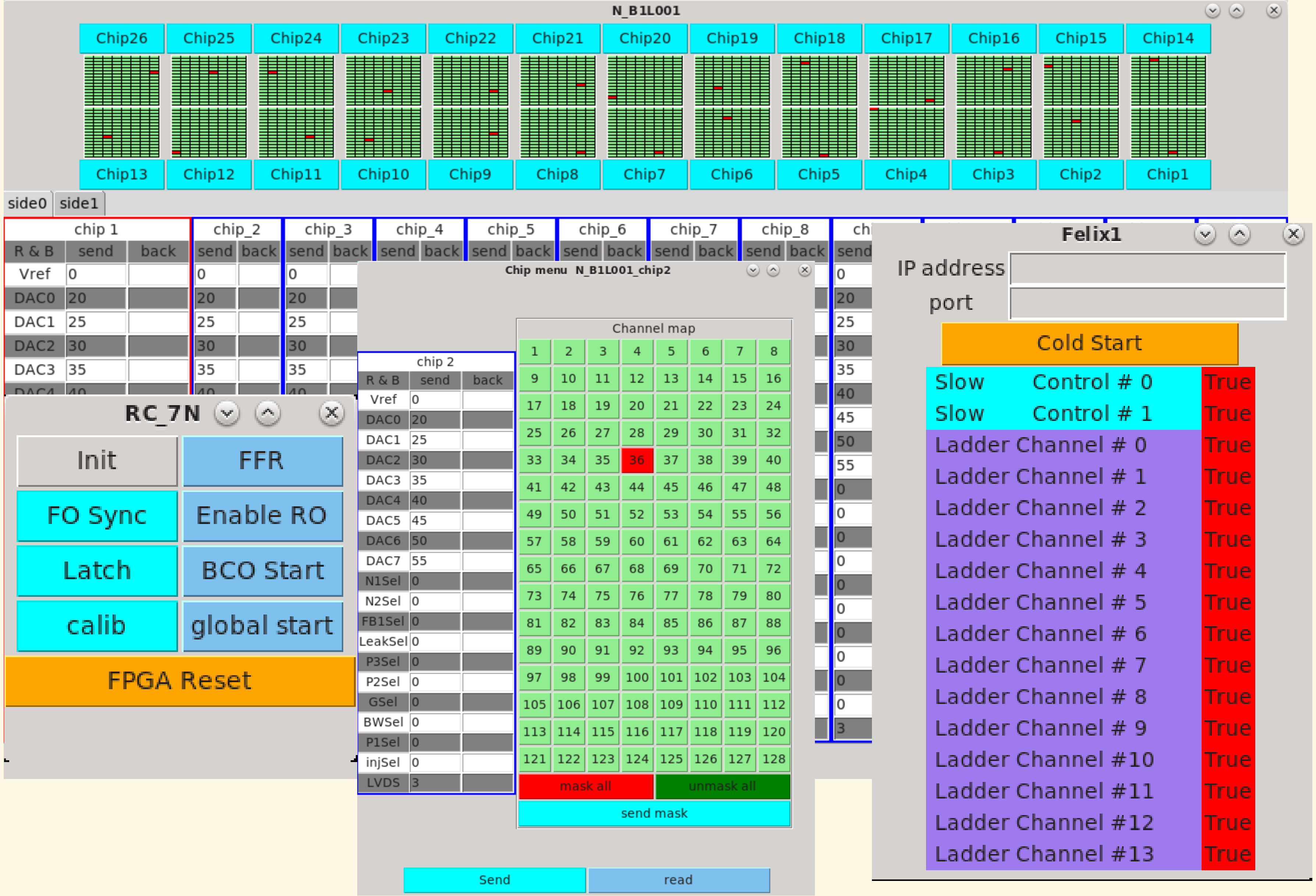
色を変える
ことも可能

Felixボード
１つのFelixに２つのROCが接続される。

現在の開発状況（ ラダー・チップ・ROC・Felixメニュー ）







chip 1

R & B	send	back
Vref	0	
DAC0	20	
DAC1	25	
DAC2	30	
DAC3	35	

chip 2

R & B	send	back
Vref	0	
DAC0	20	
DAC1	25	
DAC2	30	
DAC3	35	
DAC4	40	
DAC5	45	
DAC6	50	
DAC7	55	
N1Sel	0	
N2Sel	0	
FB1Sel	0	
LeakSel	0	
P3Sel	0	
P2Sel	0	
GSel	0	
BWSel	0	
P1Sel	0	
injSel	0	
LVDS	3	

Channel map

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88
89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104
105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128

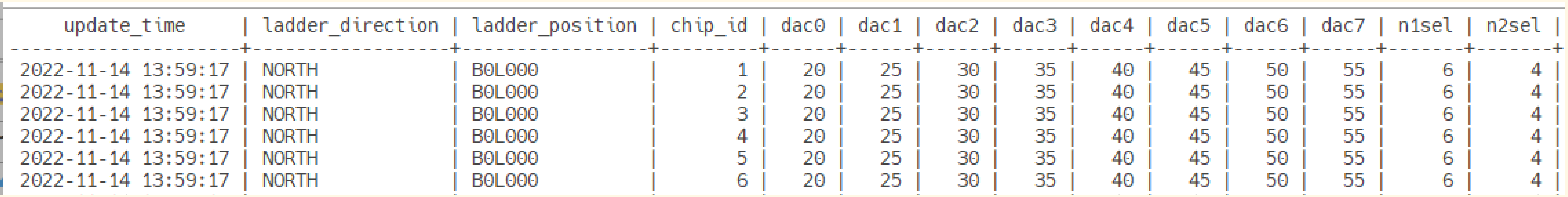
IP address

port

Slow Control # 0	True
Slow Control # 1	True
Ladder Channel # 0	True
Ladder Channel # 1	True
Ladder Channel # 2	True
Ladder Channel # 3	True
Ladder Channel # 4	True
Ladder Channel # 5	True
Ladder Channel # 6	True
Ladder Channel # 7	True
Ladder Channel # 8	True
Ladder Channel # 9	True
Ladder Channel #10	True
Ladder Channel #11	True
Ladder Channel #12	True
Ladder Channel #13	True

現在の開発状況(データベース)

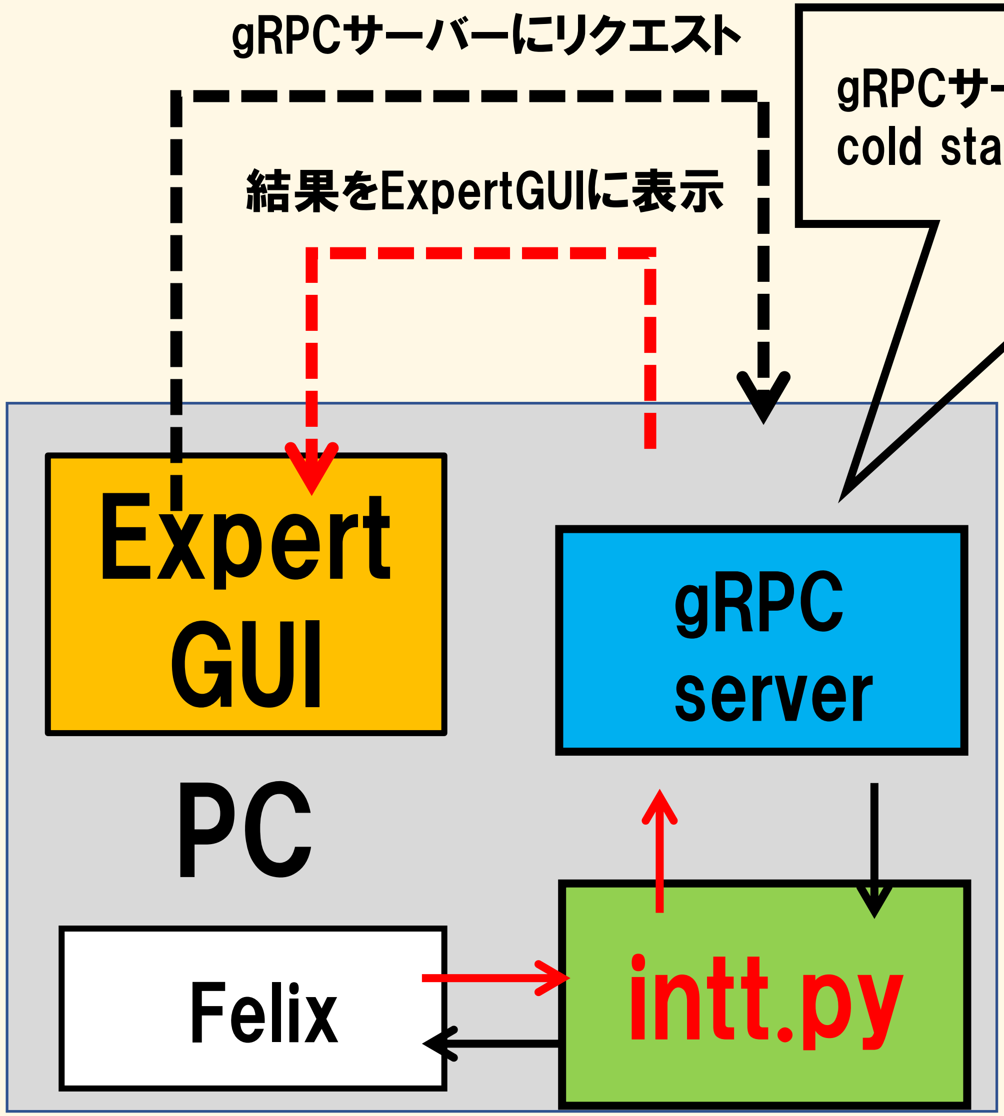
- 全ラダー分のデータを1つ1つ登録するのは、現実的では無いのでPostgreSQLのデータベースをつくるPythonスクリプトを書いた。



```
     update_time     | ladder_direction | ladder_position | chip_id | dac0 | dac1 | dac2 | dac3 | dac4 | dac5 | dac6 | dac7 | n1sel | n2sel |
---------------------+------------------+-----------------+---------+------+------+------+------+------+------+------+------+-------+-------+
 2022-11-14 13:59:17 | NORTH            | B0L000          |       1 |   20 |   25 |   30 |   35 |   40 |   45 |   50 |   55 |     6 |     4 |
 2022-11-14 13:59:17 | NORTH            | B0L000          |       2 |   20 |   25 |   30 |   35 |   40 |   45 |   50 |   55 |     6 |     4 |
 2022-11-14 13:59:17 | NORTH            | B0L000          |       3 |   20 |   25 |   30 |   35 |   40 |   45 |   50 |   55 |     6 |     4 |
 2022-11-14 13:59:17 | NORTH            | B0L000          |       4 |   20 |   25 |   30 |   35 |   40 |   45 |   50 |   55 |     6 |     4 |
 2022-11-14 13:59:17 | NORTH            | B0L000          |       5 |   20 |   25 |   30 |   35 |   40 |   45 |   50 |   55 |     6 |     4 |
 2022-11-14 13:59:17 | NORTH            | B0L000          |       6 |   20 |   25 |   30 |   35 |   40 |   45 |   50 |   55 |     6 |     4 |
```

- BNLテストベンチに開発者向けのユーザーinttdevがある。
  inttdevユーザーがPostgreSQLを使えるように設定を完了した。

現在の開発状況(gRPC)



gRPCサーバーにリクエスト

結果をExpertGUIに表示

Expert GUI

gRPC server

PC

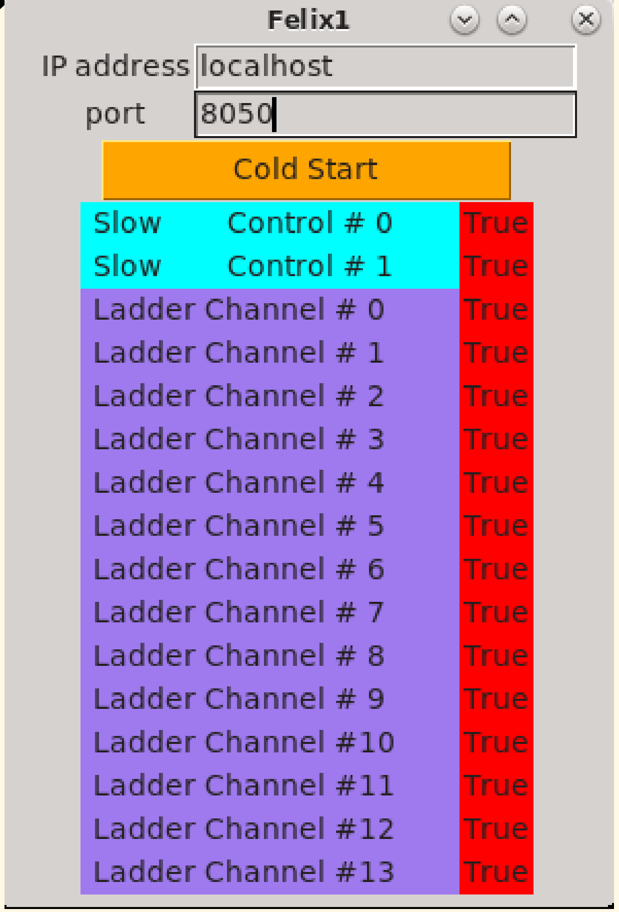
Felix

intt.py

gRPCサーバーにintt.pyの中にある
cold start 関数を実行することを登録

cold start

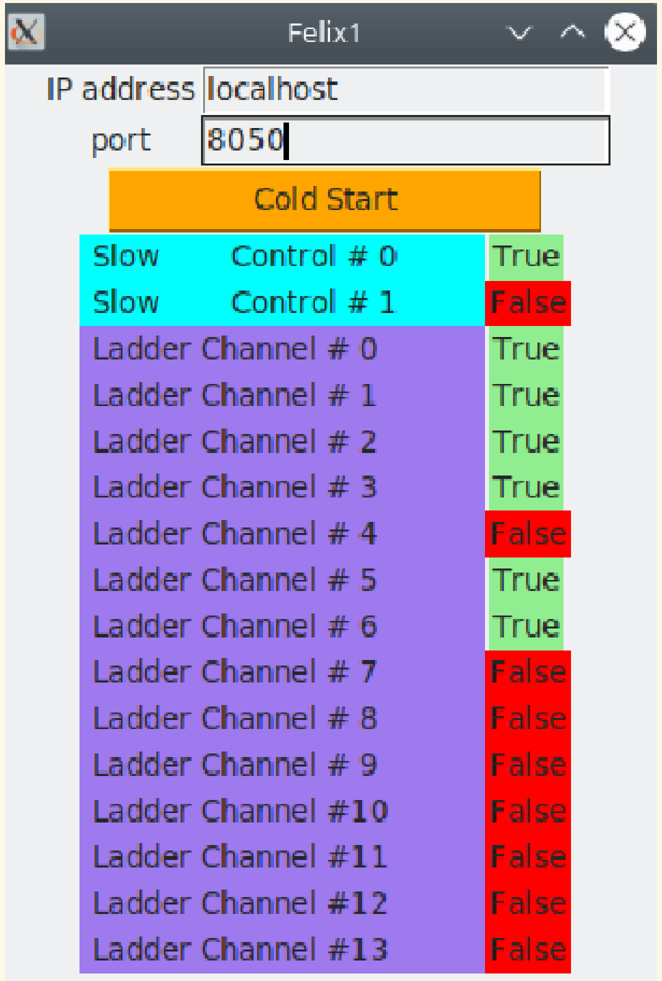
→ FelixとROCを結ぶファイバーが正しく繋がっているか確認
（ファイバーラッチ）



IP address localhost

port 8050

Slow Control # 0	True
Slow Control # 1	True
Ladder Channel # 0	True
Ladder Channel # 1	True
Ladder Channel # 2	True
Ladder Channel # 3	True
Ladder Channel # 4	True
Ladder Channel # 5	True
Ladder Channel # 6	True
Ladder Channel # 7	True
Ladder Channel # 8	True
Ladder Channel # 9	True
Ladder Channel #10	True
Ladder Channel #11	True
Ladder Channel #12	True
Ladder Channel #13	True



IP address localhost

port 8050

Slow Control # 0	True
Slow Control # 1	False
Ladder Channel # 0	True
Ladder Channel # 1	True
Ladder Channel # 2	True
Ladder Channel # 3	True
Ladder Channel # 4	False
Ladder Channel # 5	True
Ladder Channel # 6	True
Ladder Channel # 7	False
Ladder Channel # 8	False
Ladder Channel # 9	False
Ladder Channel #10	False
Ladder Channel #11	False
Ladder Channel #12	False
Ladder Channel #13	False

cold startの結果をgRPCを用いてGUIに反映されることに成功

今後の予定・やるべきこと

- Tkinter（フロントエンド）

  ラダーのポート変更などのハードウェアでの変更が生じたときに
  その変更点をデータベースに登録しなければならないが、それを行えるパネルが存在しない。

- データベース(PostgreSQL)

  1. inttdevにPostgreデータベースを作成する。
  2.遠隔のDBに接続する方法を調査する。

- gRPC

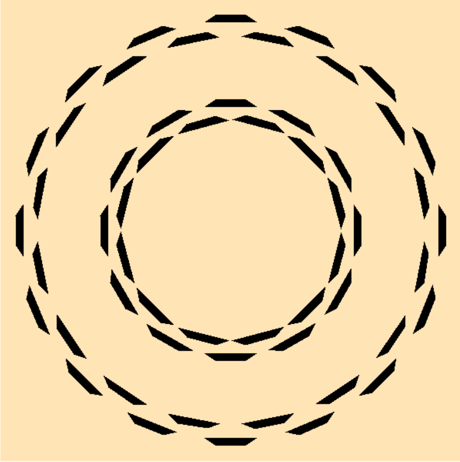
  cold start以外にもintt.pyにある関数を登録して使えるようにする。

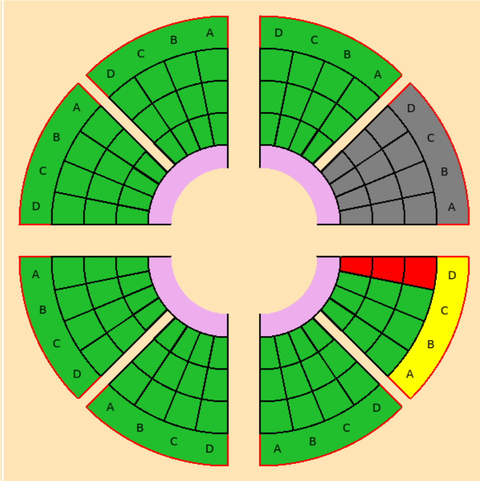
まとめ

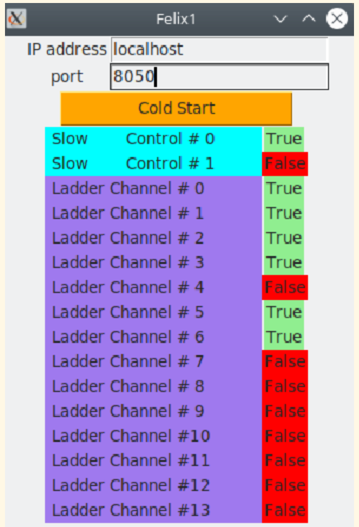
- sPHENIXで使われるすべてのラダー(バレル)やROCをコントロールできる、INTTのExpertが使うGUIアプリケーションを開発中。

- ExpertGUIに必要な基礎的な技術は何かを調査し、それぞれについて学習をした。

- intt.pyにある関数の1つ、cold start関数を実装しgRPCを用いて正しく動作することが確認された。



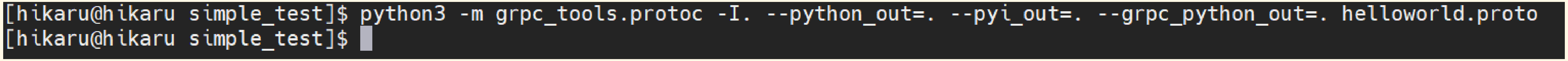




IP address localhost
port 8050

Slow Control # 0	True
Slow Control # 1	False
Ladder Channel # 0	True
Ladder Channel # 1	True
Ladder Channel # 2	True
Ladder Channel # 3	True
Ladder Channel # 4	False
Ladder Channel # 5	True
Ladder Channel # 6	True
Ladder Channel # 7	False
Ladder Channel # 8	False
Ladder Channel # 9	False
Ladder Channel #10	False
Ladder Channel #11	False
Ladder Channel #12	False
Ladder Channel #13	False

Back up



```
[hikaru@hikaru simple_test]$ python3 -m grpc_tools.protoc -I. --python_out=. --pyi_out=. --grpc_python_out=. helloworld.proto
[hikaru@hikaru simple_test]$ 
```