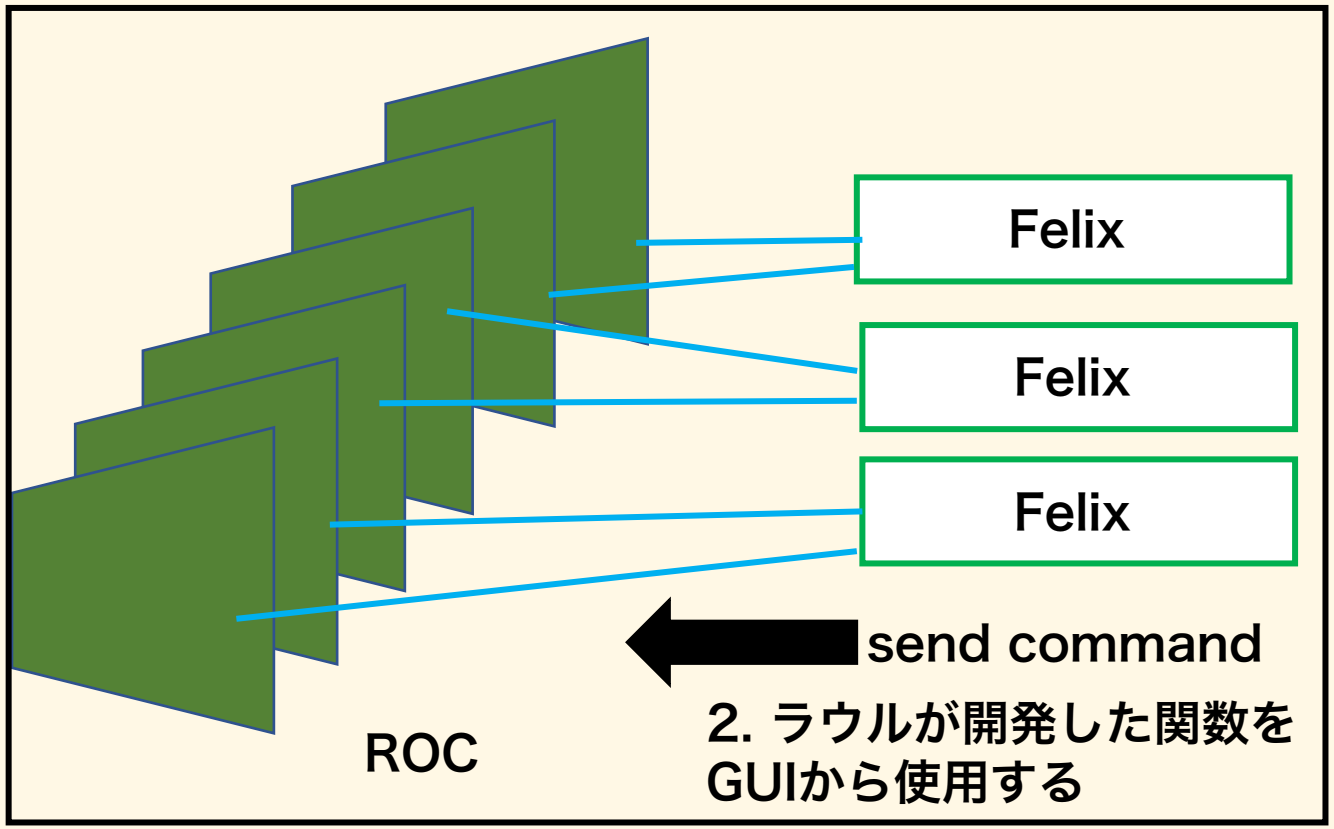
Expert GUI

Felixとの通信案

RBRC/Rikkyo

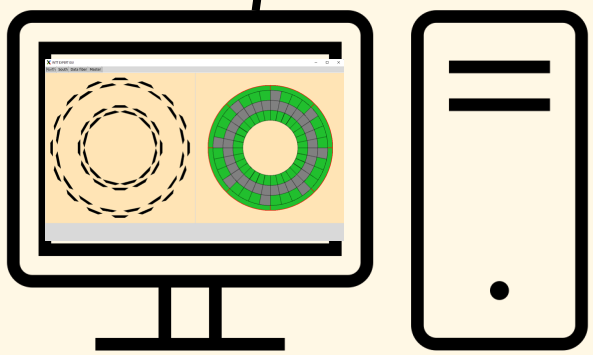
今井ひかる

やるべきこと



2. ラウルが開発した関数を
GUIから使用する

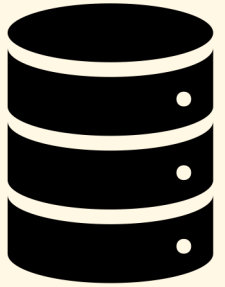
4. 外部のPCから
複数のFelixを操作する



1.フロントエンドを完成させる

5.ドキュメントを書く

データベース



3.パラメタを保存する
データベースを開発

概要

- 現在、FelixボードはPCに直接差さっているが、本番ではPCとFelixは別々。

- １つのPCから複数のFelixボードを遠隔で操作する必要がある。

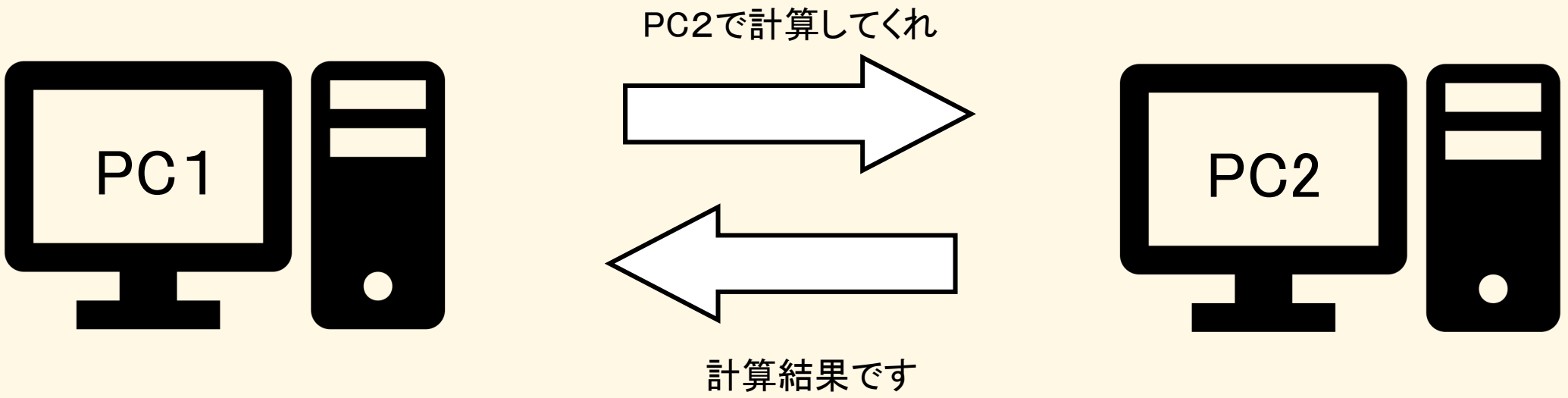
- この遠隔で操作する方法を全く知らないので、調査し、
使えそうな方法が見つかったので、提案したい。

今回試したこと

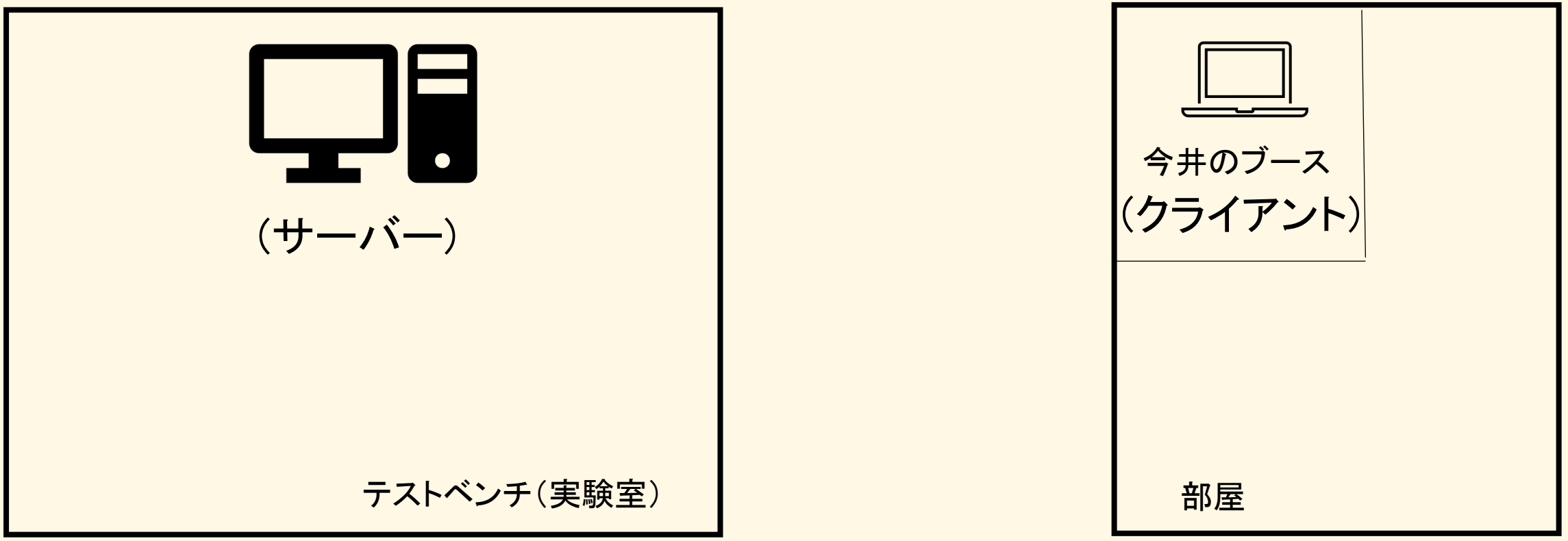
- RPC（リモートプロシージャコール）と呼ばれる技術を使う。

- RPCとは、ネットを通じて、あるPCから別のPCに処理を依頼したり、
  結果を返したりすること。

- PythonにRPCを簡単にできるライブラリがある。これを使ってみた。



今回試したこと



物理的に離れているテストベンチのPCに命令を送り、値を返してもらう実験

準備

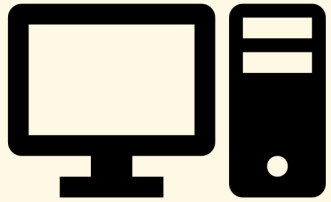
サーバー側に、左のPythonスクリプトを用意する。

```python
from xmlrpc.server import SimpleXMLRPCServer
import subprocess as sp
from Ana import Fitting_GetMean


#                          IP address     port number
with SimpleXMLRPCServer(('172.27.217.44', 8000)) as server:

    # define function which client will use

    def hello():
        return 'Hello, Pyhton!'

    def add_calc(x, y):
        return x + y


    def call_Fitting(direction):

        return Fitting_GetMean(direction)
        # PyROOT
        # return mean of gaus fitting result

    # register user function
    #                        func_val          name
    server.register_function(hello        ,"hello")
    server.register_function(add_calc     ,"add")
    server.register_function(call_Fitting,"fit")
    print("Waitting...")

    # waitting forever
    server.serve_forever()
```

あらかじめ、使う関数を定義しておく

関数を登録

rpc_server.py



（サーバー）

テストベンチ（実験室）

時間(s)	加速度(x)	加速度(y)	加速度(z)	加速度(abs)
0.0016	0.1108	0.2239	9.8649	9.8681
0.0084	0.1039	0.2147	9.8681	9.8710
0.0184	0.1061	0.2103	9.8699	9.8727
0.0284	0.1085	0.2241	9.8757	9.8789
0.0384	0.1102	0.2257	9.8643	9.8675
0.0484	0.1058	0.2200	9.8687	9.8717

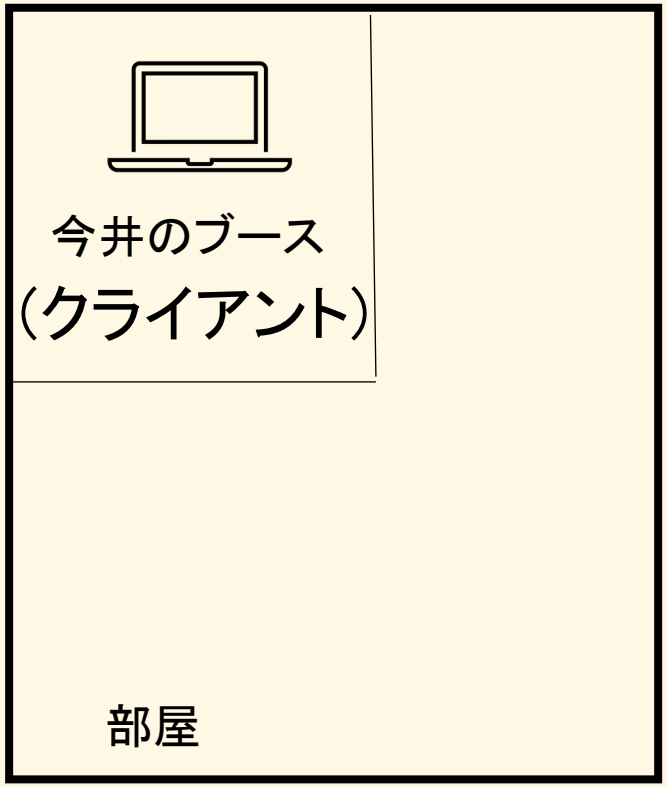
サーバにtxtデータがある。

準備

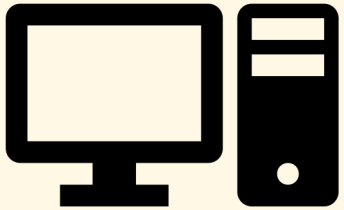
クライアント側に、左のPythonスクリプトを用意する。

```python
import xmlrpc.client

# ---- client side -----

with xmlrpc.client.ServerProxy('http://172.27.217.44:8000/') as proxy:

    # you can use function is registerd to server
    # usage --> proxy.func()

    print(proxy.hello())
    print(proxy.add(100, 200))
    print(proxy.fit("z") )
```

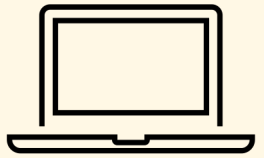
rpc_client.py



実行



サーバー側
（テストベンチPC）



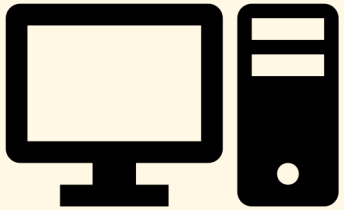
クライアント側
（今井PC）

1. $ python3 rpc_server.pyでプログラムを実行させる。
   → 準備OK

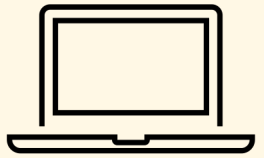
```
riken_intt@DESKTOP-5QU8C9R:/mnt/c/Users/RIKEN_INTT/Desktop/Hikaru/2022_1031$ python3 rpc_server.py
Waitting...
```

```
$ python3 rpc_server.py
```

実行



サーバー側
（テストベンチPC）



クライアント側
（今井PC）

1. $ python3 rpc_server.pyでプログラムを実行させる。
   → 準備OK

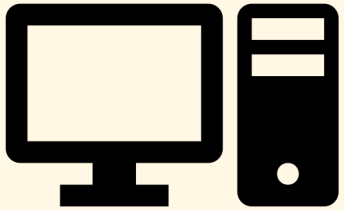
```
riken_intt@DESKTOP-5QU8C9R:/mnt/c/Users/RIKEN_INTT/Desktop/Hikaru/2022_1031$ python3 rpc_server.py
Waitting...
```

2. $ python3 rpc_client.pyでサーバー側に登録されている関数を使う

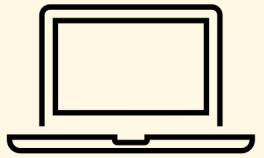
```
hikaru@hikaru:~$ python3 rpc_client.py
Hello, Pyhton!
300
9.850572973236224
```

```python
import xmlrpc.client

# ---- client side -----

with xmlrpc.client.ServerProxy('http://172.27.217.44:8000/') as proxy:

    # you can use function is registerd to server
    # usage --> proxy.func()

    print(proxy.hello())
    print(proxy.add(100, 200))
    print(proxy.fit("z") )
```

実行



サーバー側
（テストベンチPC）



クライアント側
（今井PC）

1. $ python3 rpc_server.pyでプログラムを実行させる。
   → 準備OK

```
riken_intt@DESKTOP-5QU8C9R:/mnt/c/Users/RIKEN_INTT/Desktop/Hikaru/2022_1031$ python3 rpc_server.py
Waitting...
```

```
10
11      def hello():
12          return 'Hello, Pyhton!'
13
14      def add_calc(x, y):
15          return x + y
16
17
18      def call_Fitting(direction):
19
20          return Fitting_GetMean(direction)
21          # PyROOT
22          # return mean of gaus fitting result
23
```

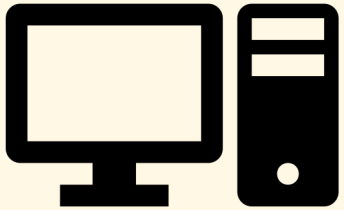
2. $ python3 rpc_client.pyでサーバー側に登録されている関数を使う

```
hikaru@hikaru:~$ python3 rpc_client.py
Hello, Pyhton!
300
9.850572973236224
```

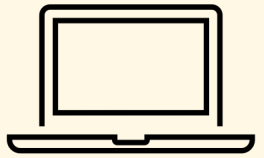
```
1  import xmlrpc.client
2
3  # ---- client side -----
4
5  with xmlrpc.client.ServerProxy('http://172.27.217.44:8000/') as proxy:
6
7      # you can use function is registerd to server
8      # usage --> proxy.func()
9
10     print(proxy.hello())
11     print(proxy.add(100, 200))
12     print(proxy.fit("z") )
```

見かけ上、サーバー側にある関数を importして
使っているように見える

実行



サーバー側
（テストベンチPC）

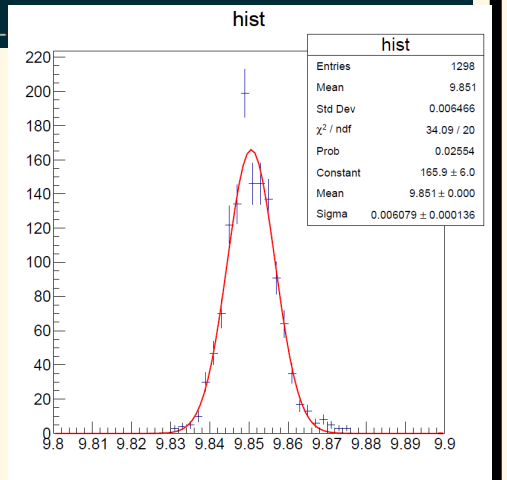


クライアント側
（今井PC）

1. $ python3 rpc_server.pyでプログラムを実行させる。
   → 準備OK

```
riken_intt@DESKTOP-5QU8C9R:/mnt/c/Users/RIKEN_INTT/Desktop/Hikaru/2022_1031$ python3 rpc_server.py
Waitting...
```

```
riken_intt@DESKTOP-5QU8C9R:/mnt/c/Users/RIKEN_INTT/Desktop/Hikaru/2022_1031$ python3 rpc_server.py
Waitting...
172.27.217.12 - - [31/Oct/2022 15:13:34] "POST / HTTP/1.1" 200 -
172.27.217.12 - - [31/Oct/2022 15:13:34] "POST / HTTP/1.1" 200 -
 FCN=7.79905 FROM MIGRAD    STATUS=CONVERGED      65 CALLS          66 TOTAL
                     EDM=7.53848e-09    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     5.40131e+02   1.88505e+01   2.17800e-02  -6.71472e-06
   2  Mean         1.05324e-01   1.08210e-04   1.53887e-07  -1.49160e-01
   3  Sigma        3.81228e-03   8.03720e-05   7.96565e-06  -2.02617e-02
Info in <TCanvas::Print>: pdf file test.pdf has been created
172.27.217.12 - - [31/Oct/2022 15:13:35] "POST / HTTP/1.1" 200 -
```



しかし、計算をしているのは、
サーバー側

サーバー側にしかデータがない。

2. $ python3 rpc_client.pyでサーバー側に登録されている関数を使う

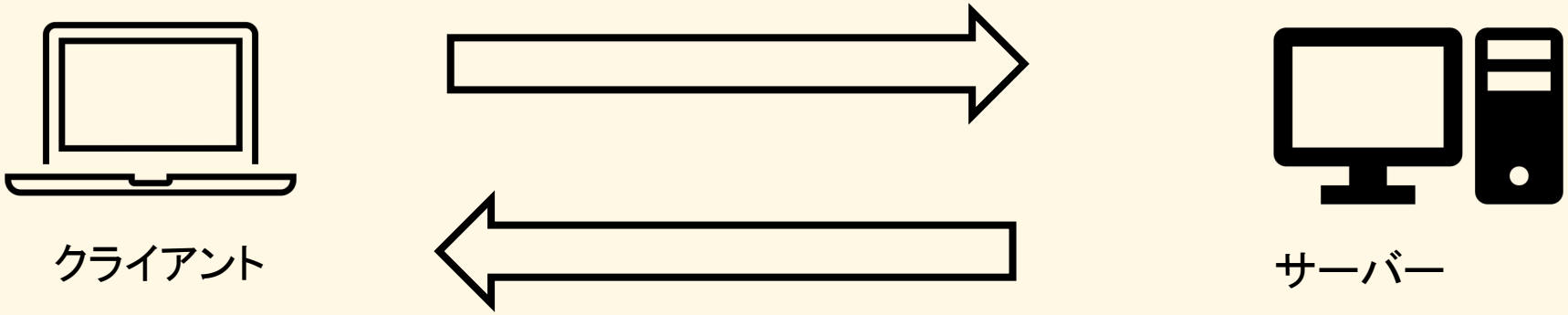
```
hikaru@hikaru:~$ python3 rpc_client.py
Hello, Pyhton!
300
9.850572973236224
```

```python
import xmlrpc.client

# ---- client side -----

with xmlrpc.client.ServerProxy('http://172.27.217.44:8000/') as proxy:

    # you can use function is registerd to server
    # usage --> proxy.func()

    print(proxy.hello())
    print(proxy.add(100, 200))
    print(proxy.fit("z") )
```

見かけ上、サーバー側にある関数を importして
使っているように見える

結果

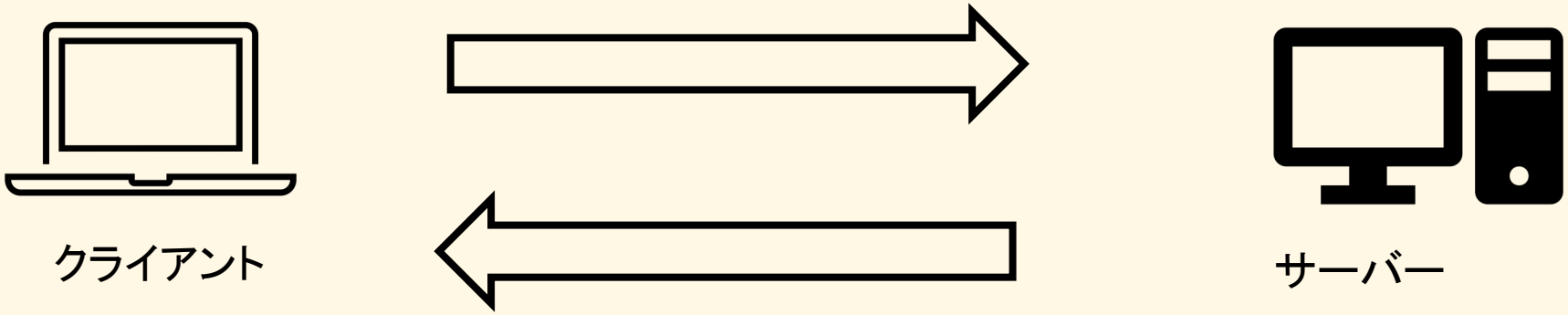
サーバーにある関数を実行（引数はクライアントが決めている）



関数の実行、リターンをクライアントに送る

結果

サーバーにある関数を実行（引数はクライアントが決めている）

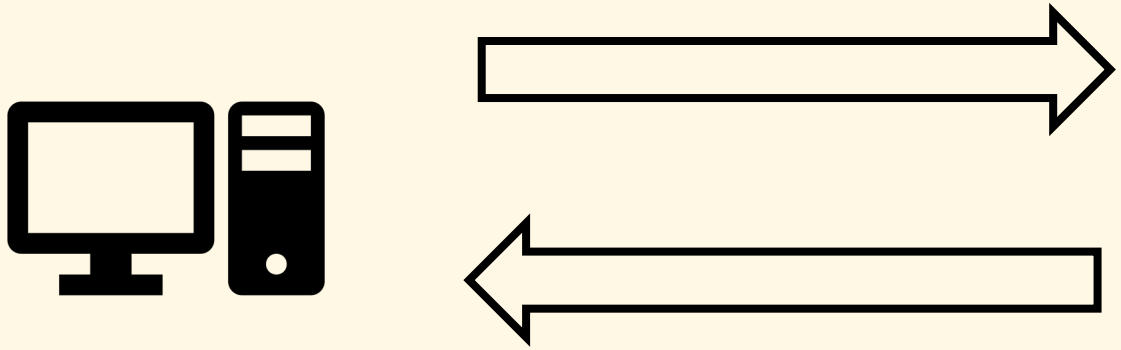


関数の実行、リターンをクライアントに送る

Expert　GUIが
入っている

例） ポート@@@に差さっているラダーのchip*** のチャンネル$$$をマスク

引数　引数　引数



様々な関数が用意されてる。
intt.py(ラウルが開発している)

- Felix
- Felix
- Felix

ReadBackなどの値を返したGUIに表示

懸念点

1番最初に、準備OKにするためのプログラムをFelixで実行する必要がある。

1. $ python3 rpc_server.pyでプログラムを実行させる。→ 準備OK

```
riken_intt@DESKTOP-5QU8C9R:/mnt/c/Users/RIKEN_INTT/Desktop/Hikaru/2022_1031$ python3 rpc_server.py
Waitting...
```

案1　Felixサーバーが立ち上がった時に自動的に、
このプログラムを実行するようにする。

まとめ

- sPHENIXでは、Felixボード複数を遠隔でコントロールする必要がある。
- そのためにRPCと呼ばれるものを使う。Pythonのライブラリを使う。
- 理研の中で試しに使ってみたところ、別のPCに計算させ、結果を返してもらうことができた。