地車APIの概要説明書

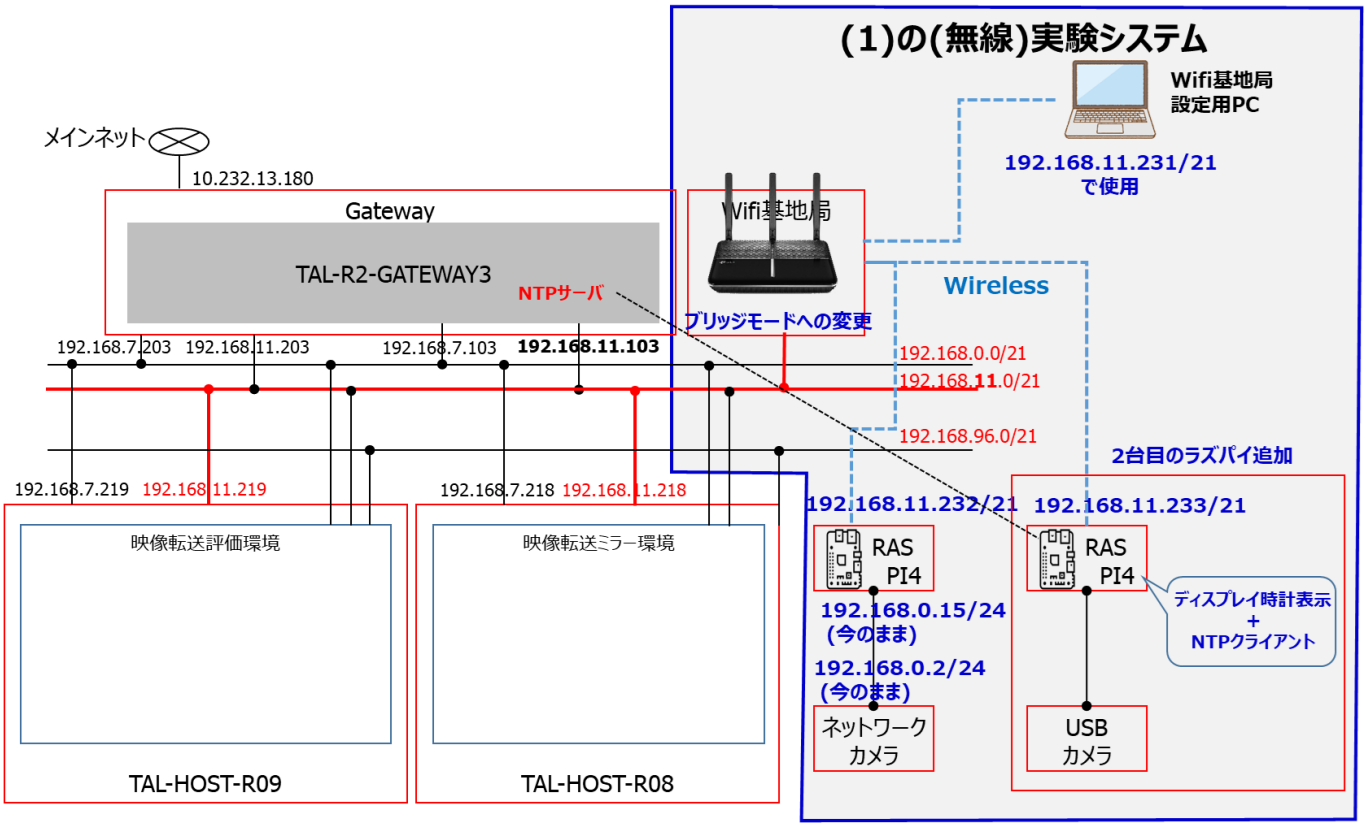
2024/02/05

江端智一

このファイルの本体の格納場所は、C:\Users\ebata\fastapi6です。

ideo_stream_requests.json](#73-video_stream_requestsjson)

1. 前提システム

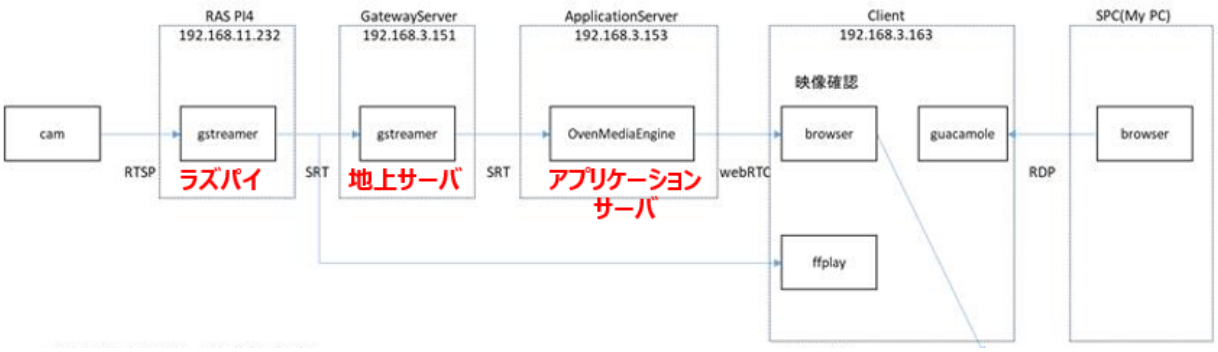


2. 作業環境

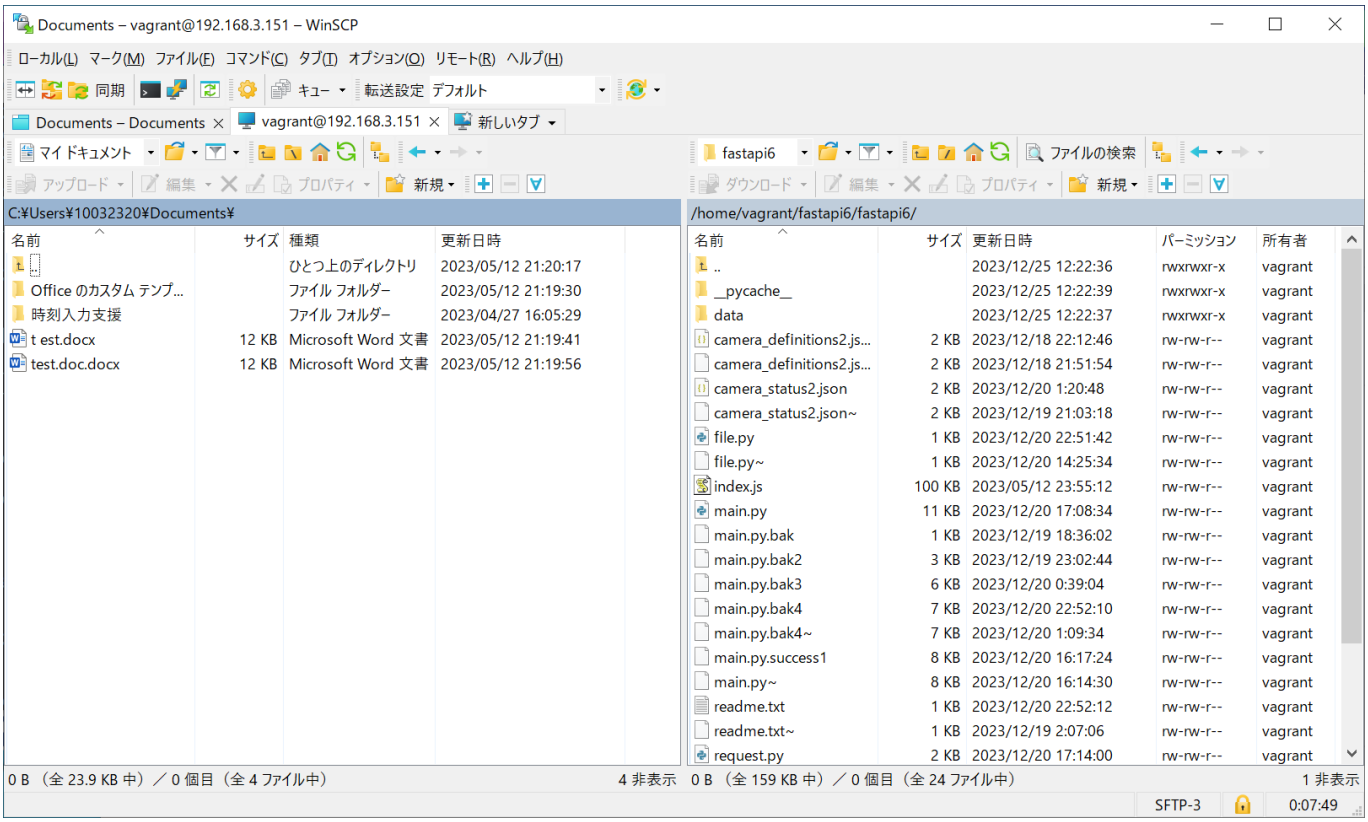
2.1. ファイル転送

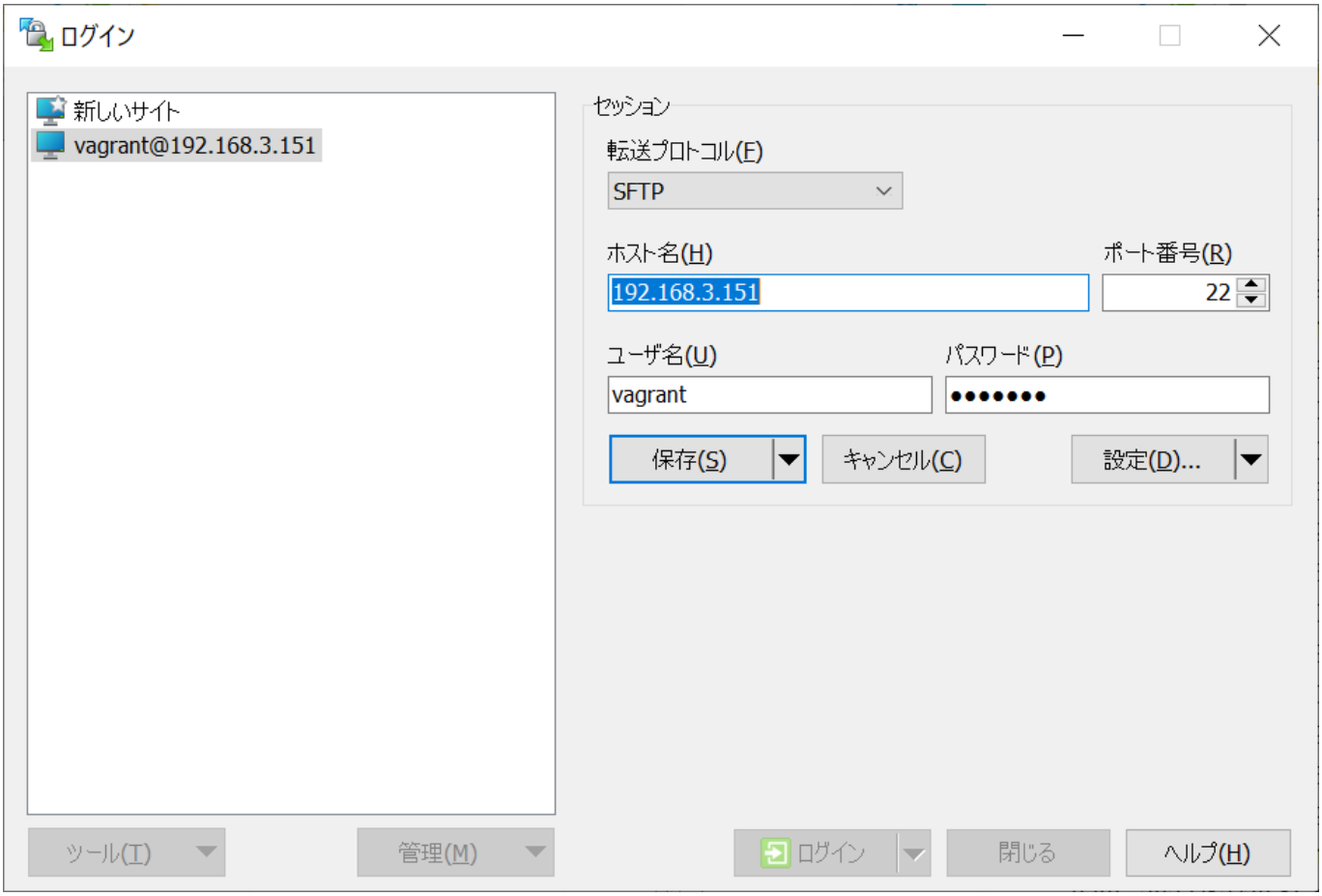
WinSCPによる192.168.3.151(地上サーバ)と、192.168.11.232(車上サーバ(ラズパイ))への接続方法

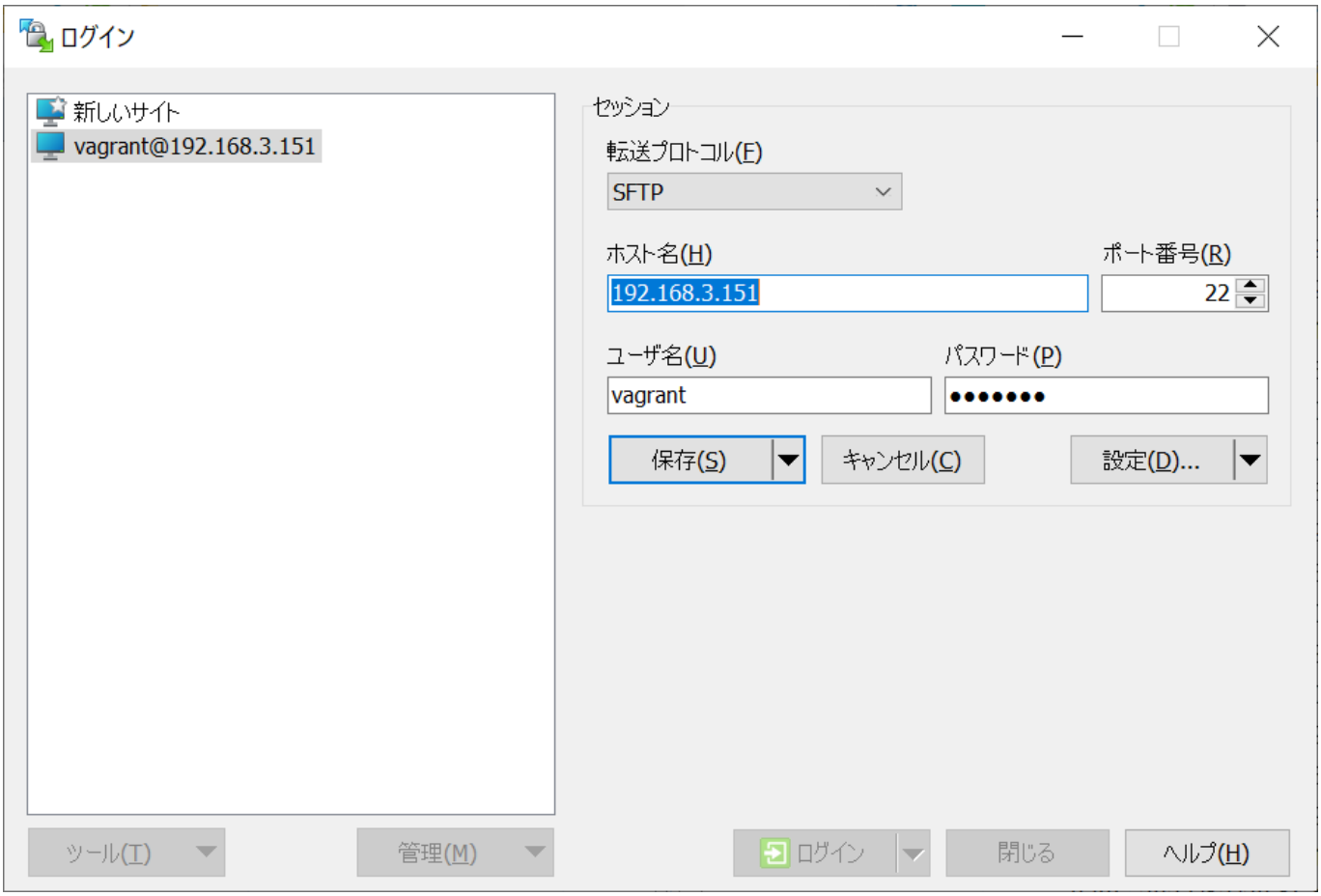
■前提システム(上記の図との連携に不明点あり)



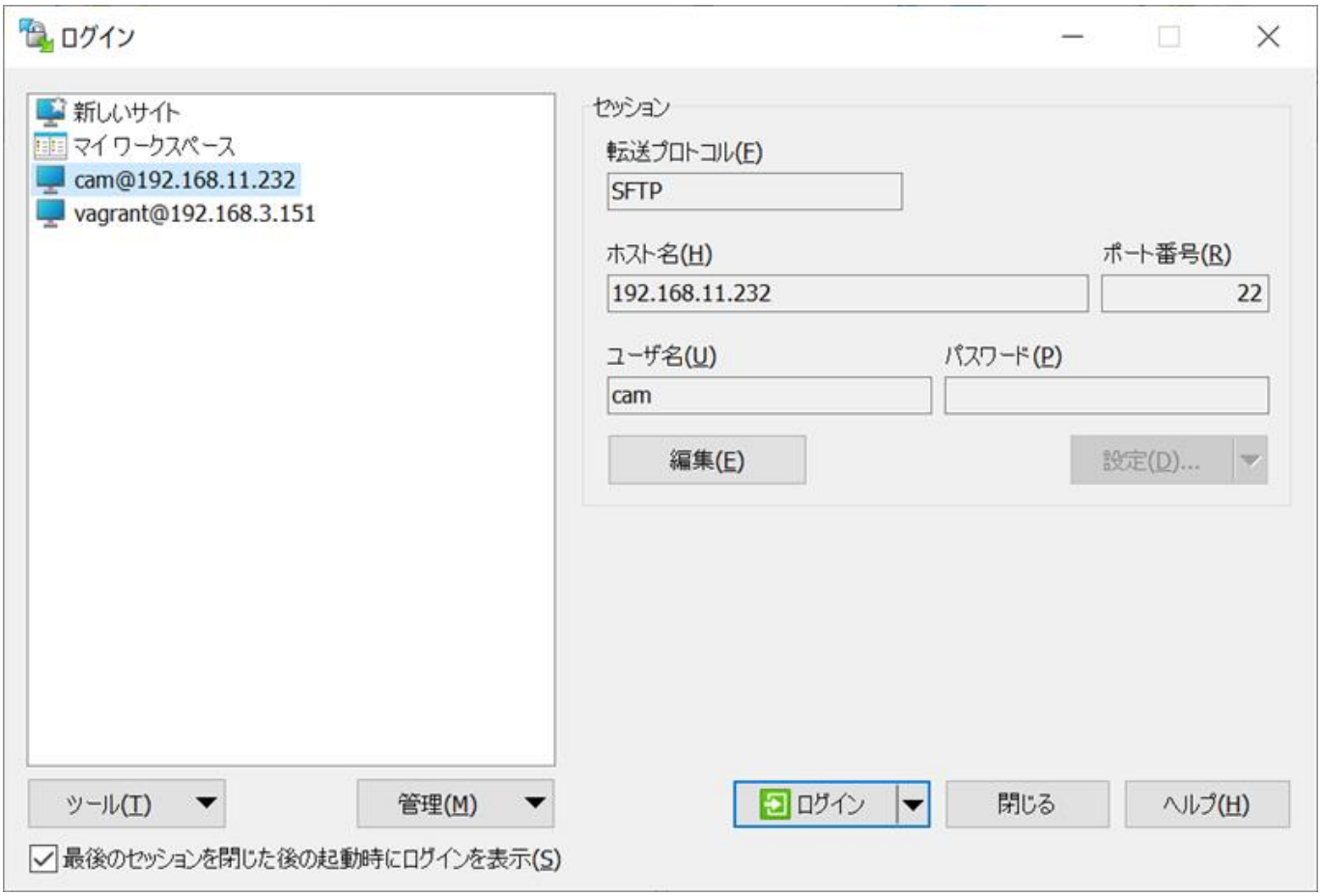
2.1.1. (地上サーバ)へのファイルの送付方法

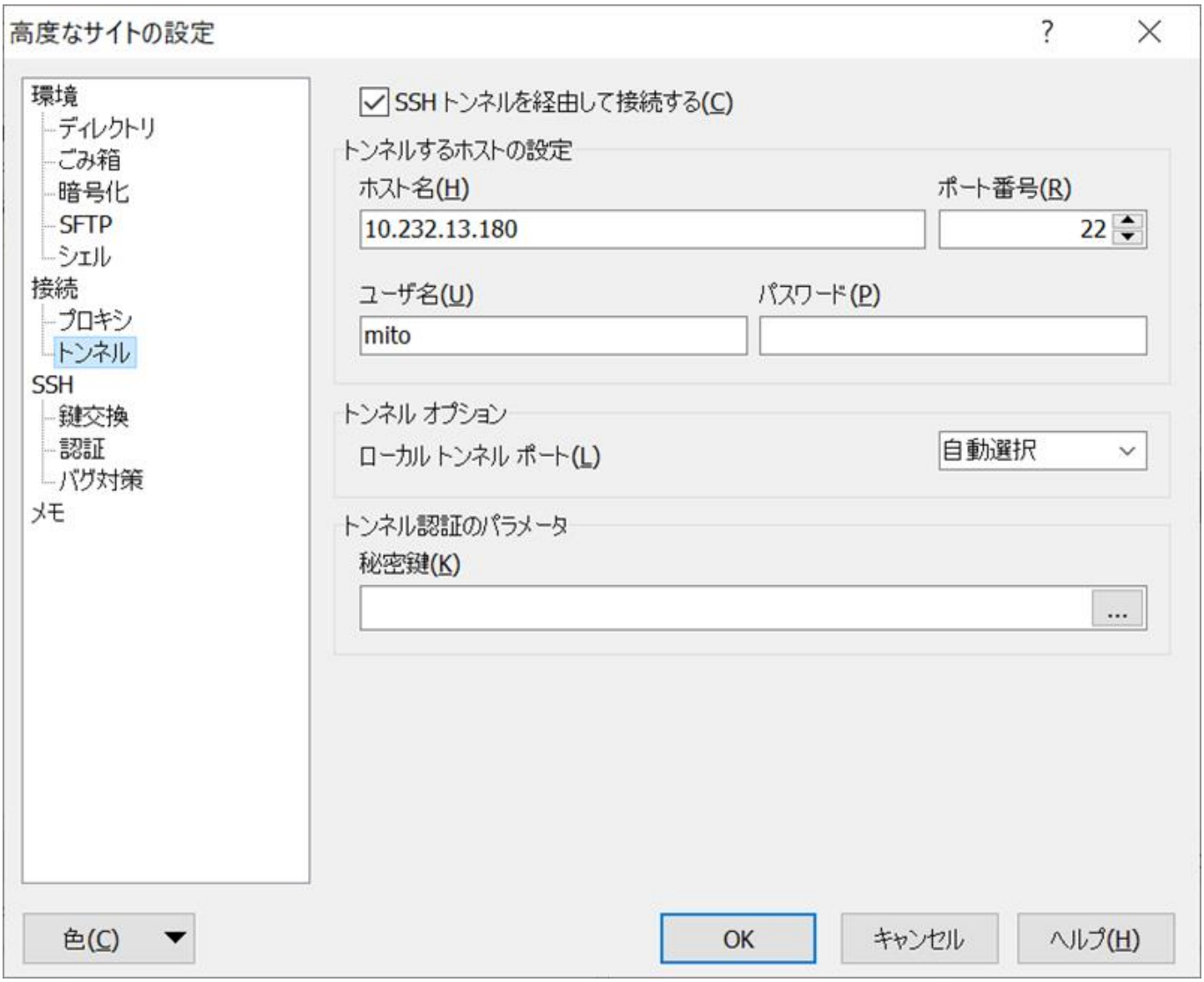

Documents – vagrant@192.168.3.151 – WinSCP
ローカル(L) マーク(M) ファイル(F) コマンド(C) タブ(T) オプション(O) リモート(R) ヘルプ(H)
同期 キュー 転送設定 デフォルト
Documents – Documents
vagrant@192.168.3.151
新しいタブ
マイ ドキュメント
アップロード 編集 プロパティ 新規
C:¥Users¥10032320¥Documents¥
名前 サイズ 種類 更新日時
ひとつ上のディレクトリ 2023/05/12 21:20:17
Office のカスタム テンプ... ファイル フォルダー 2023/05/12 21:19:30
時刻入力支援 ファイル フォルダー 2023/04/27 16:05:29
t est.docx 12 KB Microsoft Word 文書 2023/05/12 21:19:41
test.doc.docx 12 KB Microsoft Word 文書 2023/05/12 21:19:56
0 B （全 23.9 KB 中）／0 個目（全 4 ファイル中）
4 非表示
fastapi6
ファイルの検索
ダウンロード 編集 プロパティ 新規
/home/vagrant/fastapi6/fastapi6/
名前 サイズ 更新日時 パーミッション 所有者
.. 2023/12/25 12:22:36 rwxrwxr-x vagrant
__pycache__ 2023/12/25 12:22:39 rwxrwxr-x vagrant
data 2023/12/25 12:22:37 rwxrwxr-x vagrant
camera_definitions2.js... 2 KB 2023/12/18 22:12:46 rw-rw-r-- vagrant
camera_definitions2.js... 2 KB 2023/12/18 21:51:54 rw-rw-r-- vagrant
camera_status2.json 2 KB 2023/12/20 1:20:48 rw-rw-r-- vagrant
camera_status2.json~ 2 KB 2023/12/19 21:03:18 rw-rw-r-- vagrant
file.py 1 KB 2023/12/20 22:51:42 rw-rw-r-- vagrant
file.py~ 1 KB 2023/12/20 14:25:34 rw-rw-r-- vagrant
index.js 100 KB 2023/05/12 23:55:12 rw-rw-r-- vagrant
main.py 11 KB 2023/12/20 17:08:34 rw-rw-r-- vagrant
main.py.bak 1 KB 2023/12/19 18:36:02 rw-rw-r-- vagrant
main.py.bak2 3 KB 2023/12/19 23:02:44 rw-rw-r-- vagrant
main.py.bak3 6 KB 2023/12/20 0:39:04 rw-rw-r-- vagrant
main.py.bak4 7 KB 2023/12/20 22:52:10 rw-rw-r-- vagrant
main.py.bak4~ 7 KB 2023/12/20 1:09:34 rw-rw-r-- vagrant
main.py.success1 8 KB 2023/12/20 16:17:24 rw-rw-r-- vagrant
main.py~ 8 KB 2023/12/20 16:14:30 rw-rw-r-- vagrant
readme.txt 1 KB 2023/12/20 22:52:12 rw-rw-r-- vagrant
readme.txt~ 1 KB 2023/12/19 2:07:06 rw-rw-r-- vagrant
request.py 2 KB 2023/12/20 17:14:00 rw-rw-r-- vagrant
0 B （全 159 KB 中）／0 個目（全 24 ファイル中）
1 非表示
SFTP-3
0:07:49

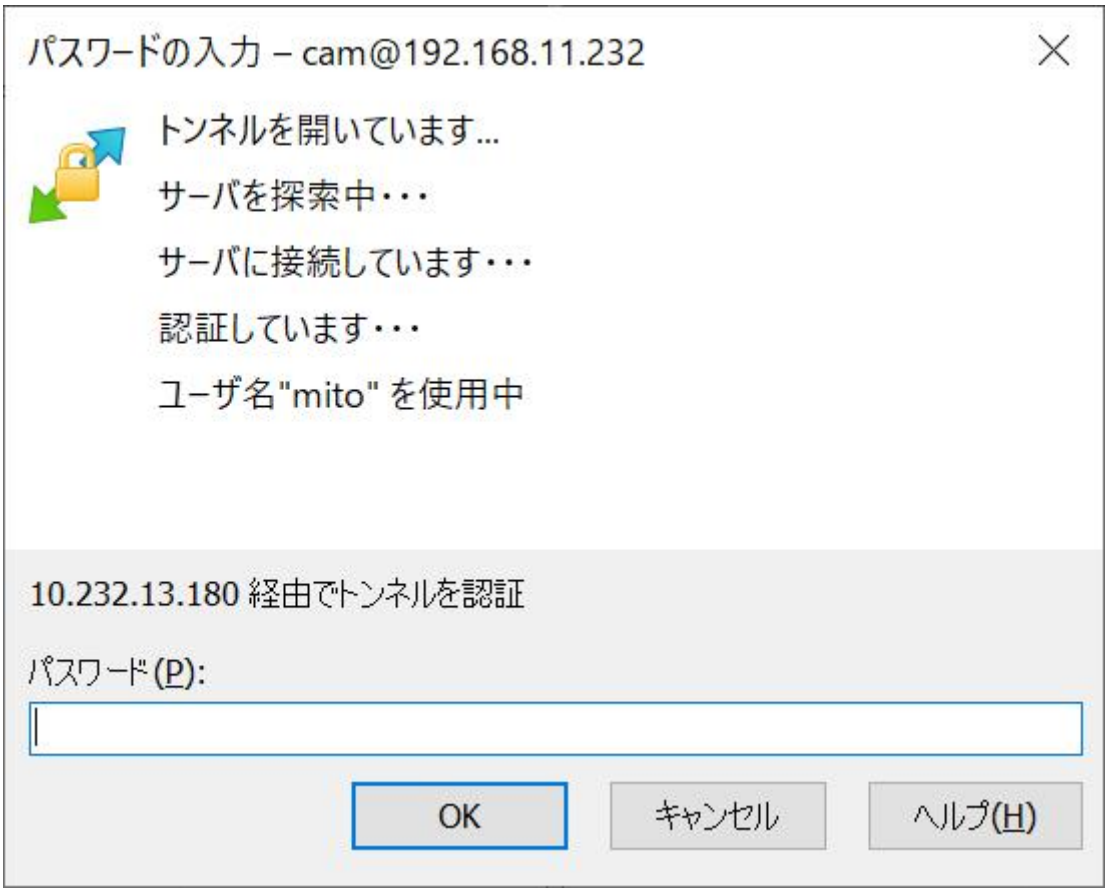

ログイン
新しいサイト
vagrant@192.168.3.151
セッション
転送プロトコル(F)
SFTP
ホスト名(H)
192.168.3.151
ポート番号(R)
22
ユーザ名(U)
vagrant
パスワード(P)
●●●●●●●
保存(S)
キャンセル(C)
設定(D)...
ツール(T)
管理(M)
ログイン
閉じる
ヘルプ(H)



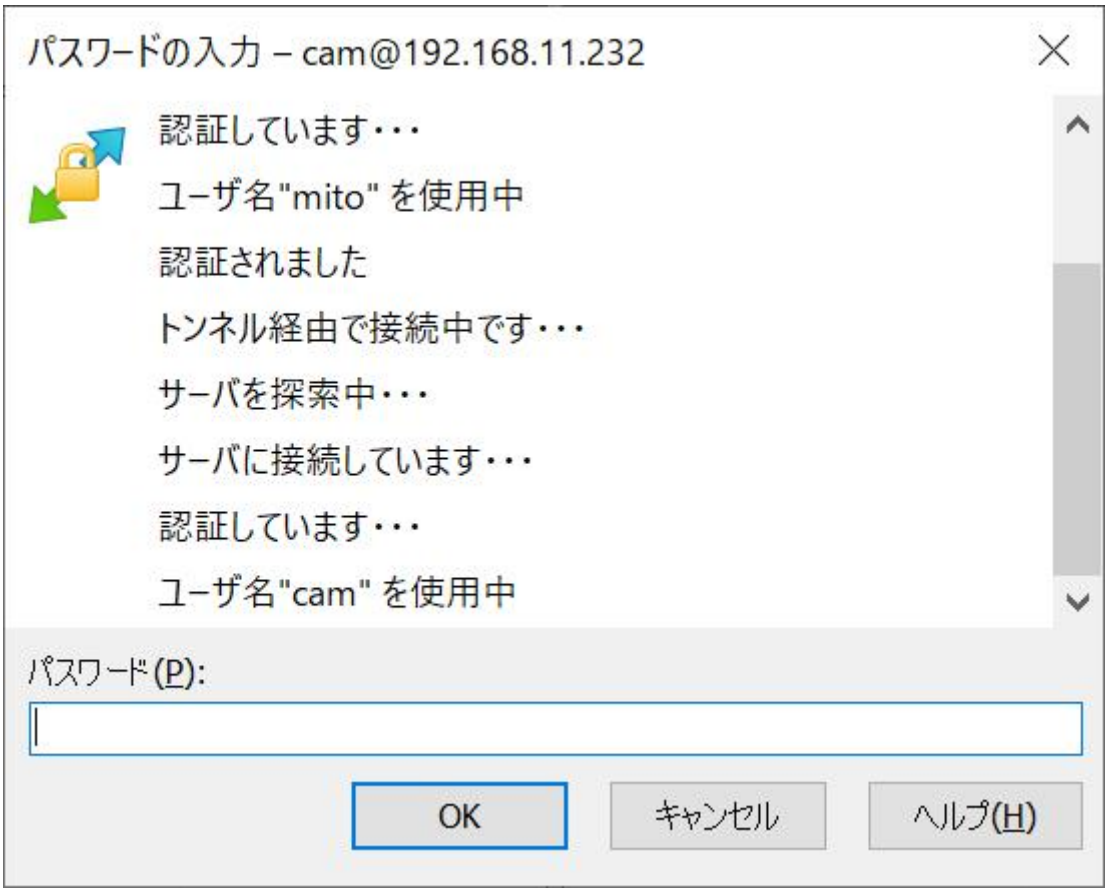
2.1.2. (車上サーバ(ラズパイ))へのファイルの送付方法


ログイン
新しいサイト
マイ ワークスペース
cam@192.168.11.232
vagrant@192.168.3.151
セッション
転送プロトコル(F)
SFTP
ホスト名(H)
192.168.11.232
ポート番号(R)
22
ユーザ名(U)
cam
パスワード(P)
編集(E)
設定(D)...
ツール(T)
管理(M)
ログイン
閉じる
ヘルプ(H)
最後のセッションを閉じた後の起動時にログインを表示(S)


高度なサイトの設定
環境
ディレクトリ
ごみ箱
暗号化
SFTP
シェル
接続
プロキシ
トンネル
SSH
鍵交換
認証
バグ対策
メモ
SSH トンネルを経由して接続する(C)
トンネルするホストの設定
ホスト名(H)
10.232.13.180
ポート番号(R)
22
ユーザ名(U)
mito
パスワード(P)
トンネル オプション
ローカル トンネル ポート(L)
自動選択
トンネル認証のパラメータ
秘密鍵(K)
色(C)
OK
キャンセル
ヘルプ(H)



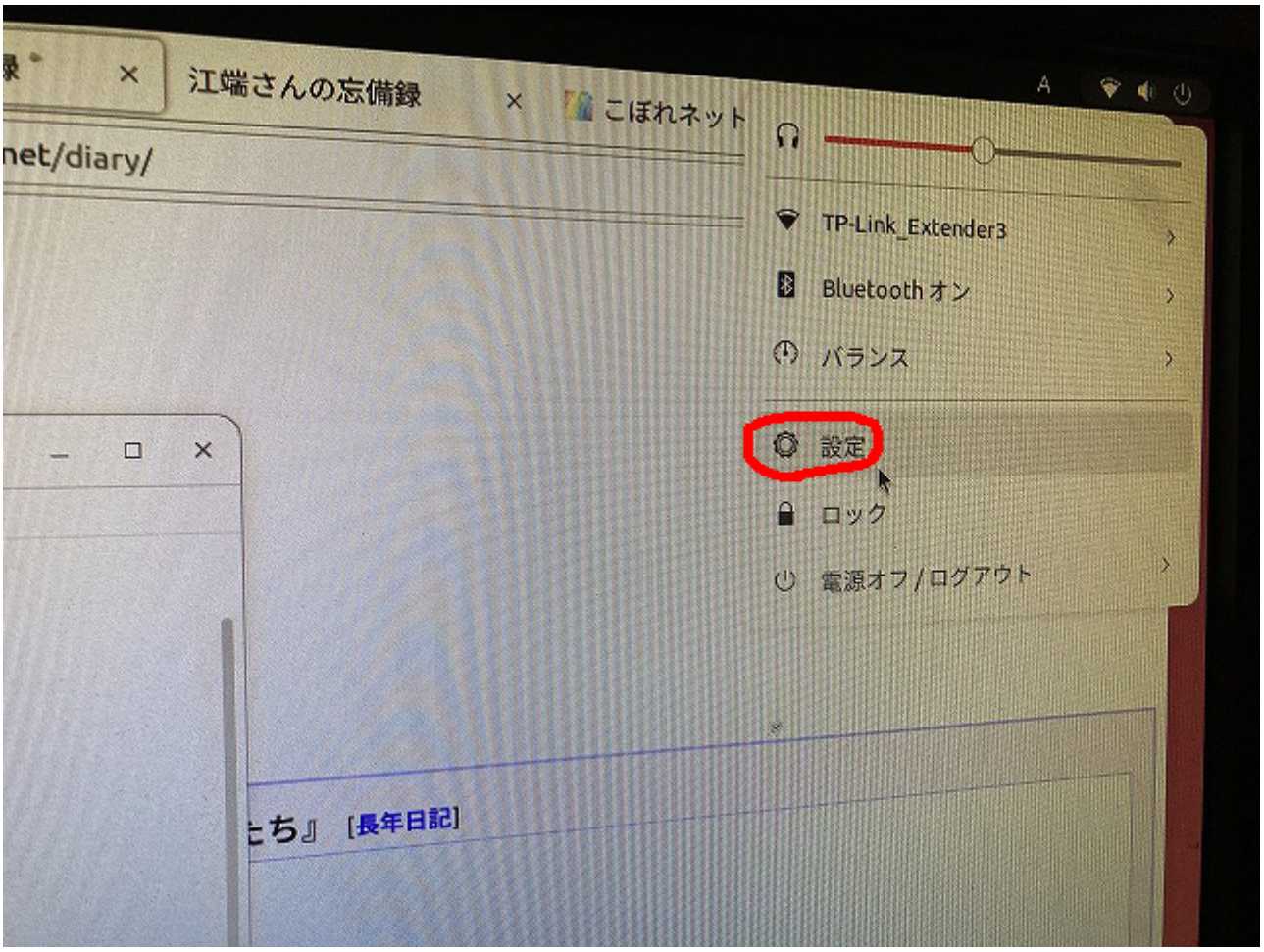
"mito0"を入力

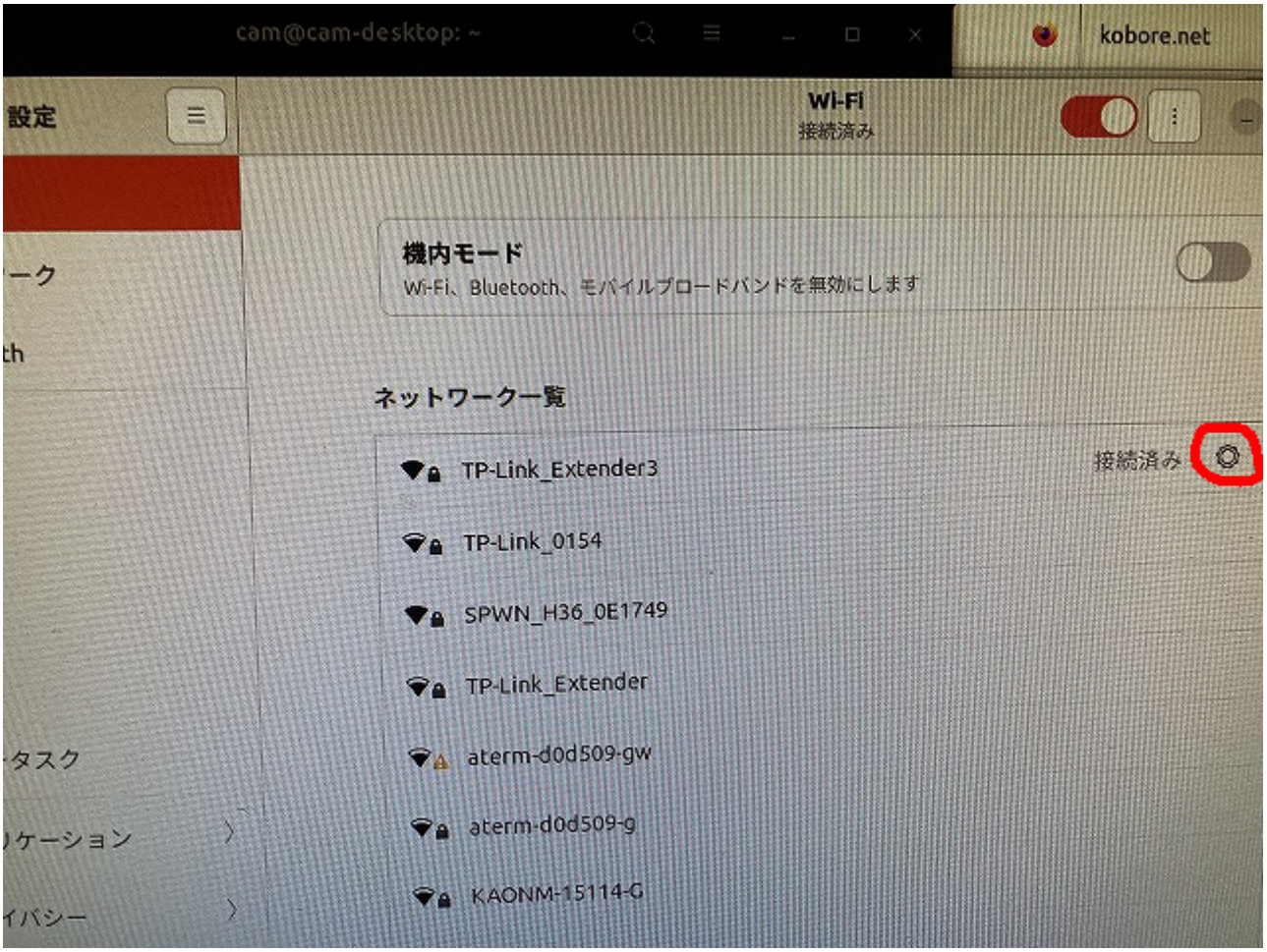


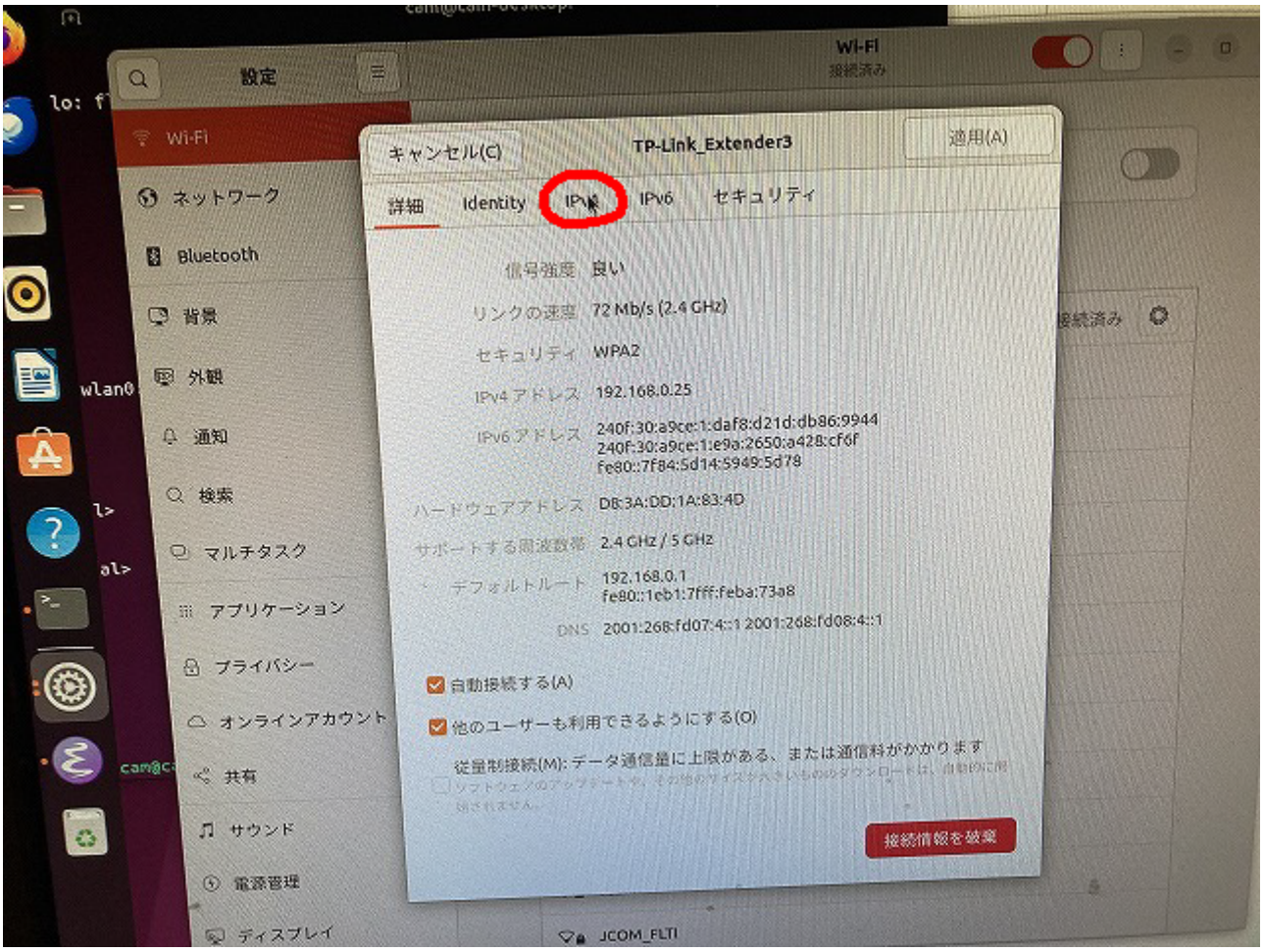
"cam"を入力

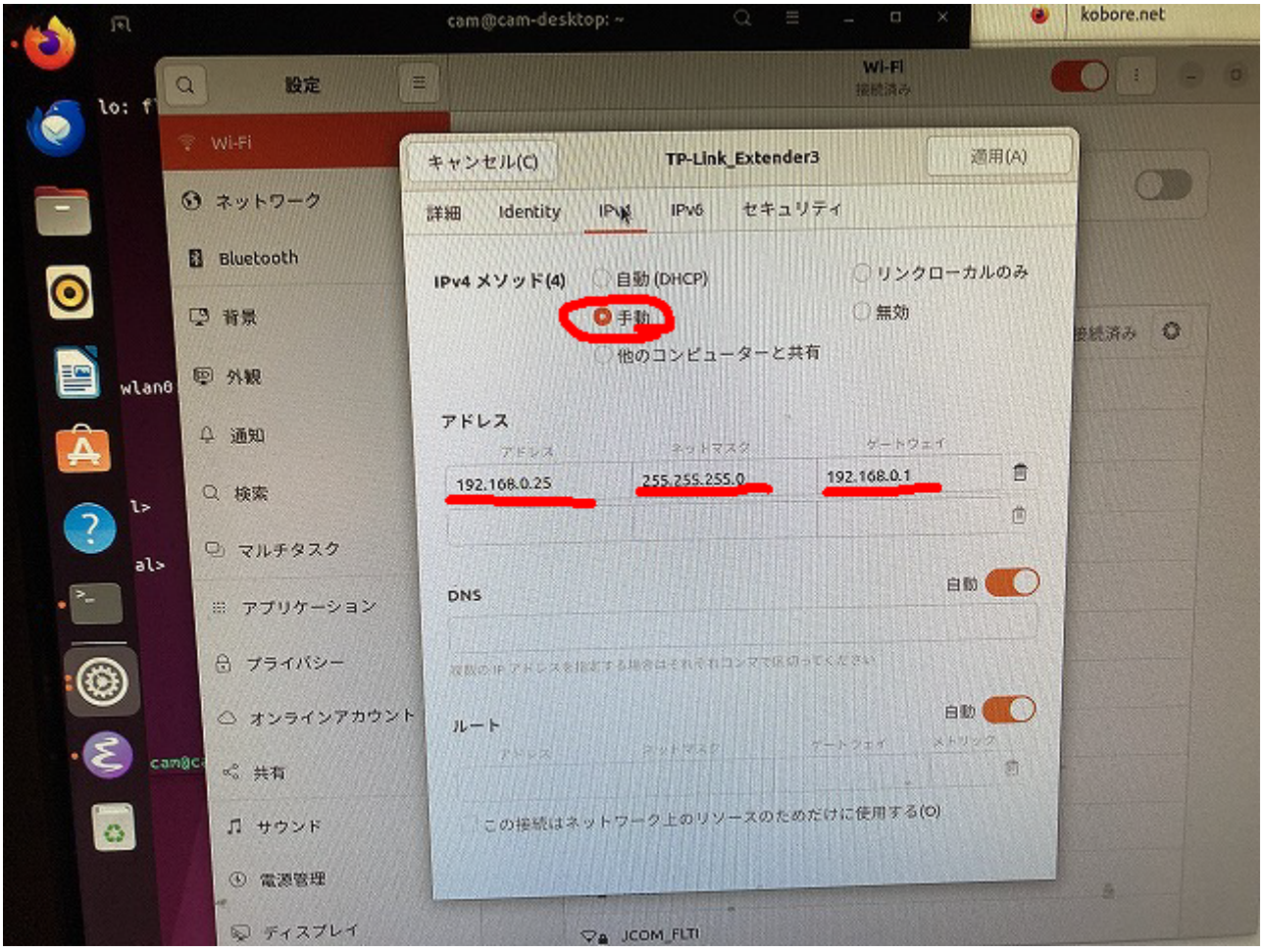
3. 準備

3.1. ラズパイ4のGUIから無線ネットワークを固定IPアドレスにする方法


江端さんの忘備録
こぼれネット
net/diary/
TP-Link_Extender3
Bluetooth オン
バランス
設定
ロック
電源オフ / ログアウト
ち』 [長年日記]


cam@cam-desktop: ~
kobore.net
設定
Wi-Fi
接続済み
機内モード
Wi-Fi、Bluetooth、モバイルブロードバンドを無効にします
ネットワーク一覧
TP-Link_Extender3
接続済み
TP-Link_0154
SPWN_H36_0E1749
TP-Link_Extender
aterm-d0d509-gw
aterm-d0d509-g
KAONM-15114-G





3.2. ラズパイ4のGUIから優先ネットワークを固定IPアドレスにする方法

/etc/netplan/　以下の設定ファイルを書き換えてDHCP接続をする形としました。

元々の設定ファイルは99_xxxx.yaml.tal3　という形で残してあります。

3.3. インストール

3.3.1. 最新のPIPをアップグレードする

```
> pip install --upgrade pip
> python -m pip install --upgrade pip
> python3 -m pip install --upgrade pip
```

のいずれかでできる。

3.3.2. uvicorn, fastapi, pydantic, pandas, requests, jsonのモジュールをインストールする

```
> pip3 install fastapi
> pip3 install uvicorn[standard]
```

は必須だが、後はプログラムを動かせば、プログラムの方から、インストールを指示されると思うので大丈夫(だろう)。

3.4. ログイン

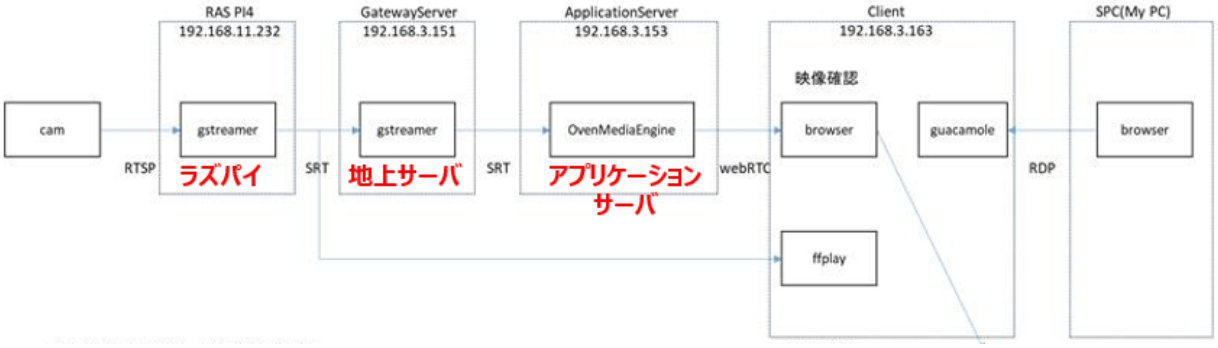
192.168.3.151(applicationServer)にsshログインして下さい。

江端の場合

- C:\Users\10032320>ssh video_application
    - mito@10.232.13.180's password:("mito0")
    - vagrant@192.168.3.153's password:("vagrant")
- vagrant@VideoApplication:~$ ssh 192.168.3.151
    - vagrant@192.168.3.151's password:("vagrant")
    - (地上サーバのログイン。さらに以下に続く)
- vagrant@CloudGateway:~$ ssh cam@192.168.11.232
    - password:("cam")
    - (車上サーバ(ラズパイ)のログイン)

3.5. ネットワークを使ったプログラムをする場合の注意

実際のところ、このIPアドレスでは、通信が成立しないことがあります。 192.168.3.*では、SRTのストリームが通りません(理由は不明ですが)



車上サーバ(ラズパイ)	地上サーバ	アプリケーションサーバ	(補足)
192.168.**11**.232	192.168.3.151	192.168.3.13	(SRTの通信に失敗することがある)
(同上)	192.168.**13**.151	192.168.**13**.153	SRTの通信には成功する様子
(同上)	192.168.**11**.105	192.168.**11**.103	完全に同じセグメント

4. APIサーバ起動方法

4.1. 地上APIサーバ(192.168.3.151)起動方法

- $ cd ~
- $ cd **fastapi6d**
- **$ sudo -E uvicorn main:app --host 0.0.0.0 --reload --port 8001** ("--port 8001"は車上サーバと同じホストに配置する場合に必要だが普通は不要)
- **sudo socat udp-listen:38089,reuseaddr,fork udp:192.168.3.153:38089** (これでSRTストリームをフォワードする)

4.2. 車上APIサーバ(192.168.11.232(ラズパイ))起動方法

- $ cd ~
- $ cd **fastapi7**
- **$ sudo -E uvicorn test:app --host 0.0.0.0 --reload**

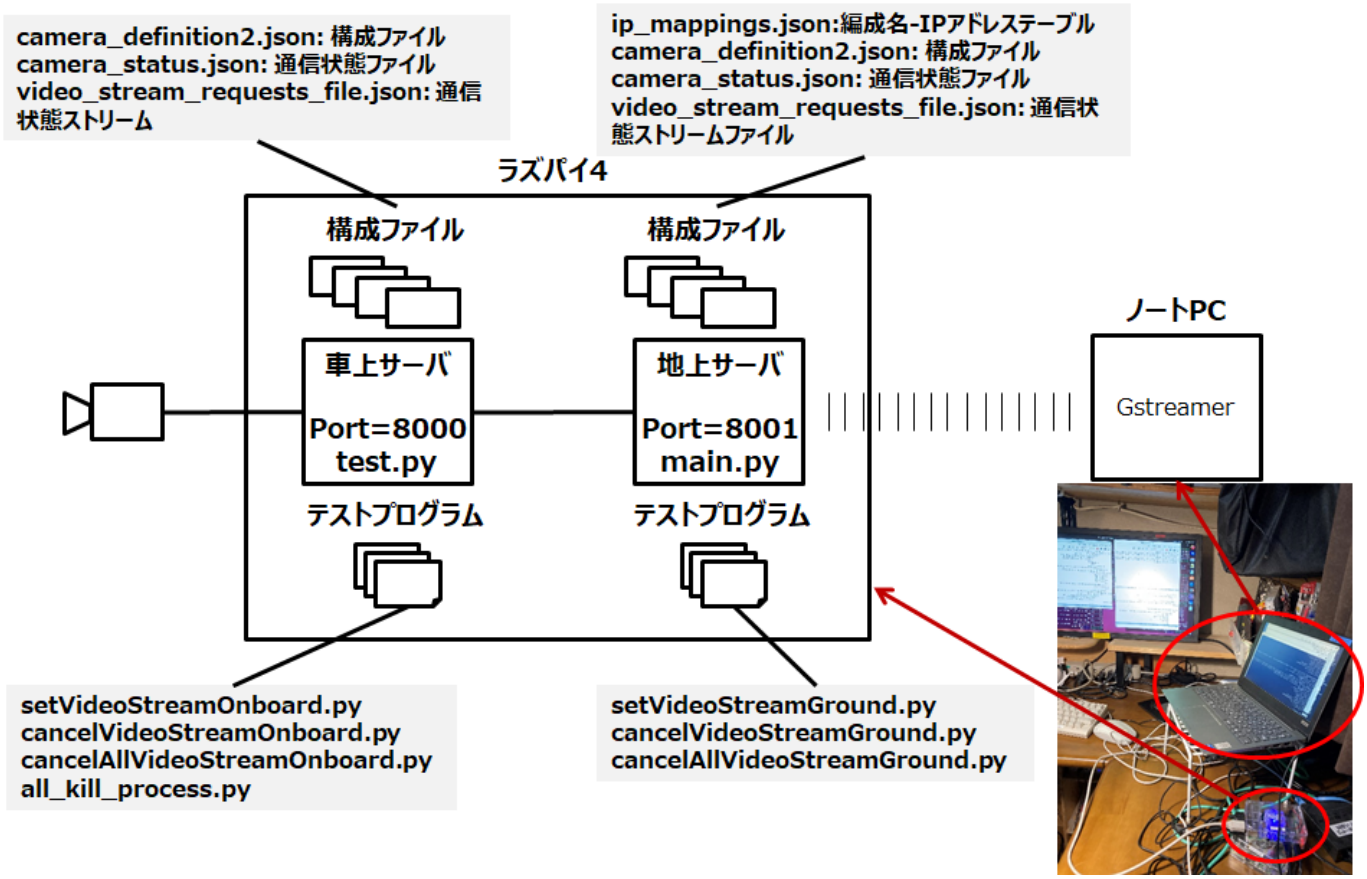
4.3. 注意事項

- 上記の"**--host 0.0.0.0**"がないと、リモートからのアクセスができない
- Windowsの場合、インバウントでポートオープンする必要がある場合がある。
    - ファイアウォールのポートを開く: FastAPIアプリケーションがリッスンしているポートを開く必要があります。デフォルトではポート8000が使用されますが、FastAPIアプリケーションがリッスンしているポートに合わせてポート番号を指定してください。
    - スタートメニューから「Windows セキュリティ」または「Windows Defender セキュリティセンター」を開きます。
    - 「ファイアウォールとネットワーク保護」を選択します。
    - 「詳細設定」をクリックします。
    - 「インバウンド ルール」を選択し、「新しいルールの作成」をクリックします。
    - ルールの種類として「ポート」を選択し、「次へ」をクリックします。

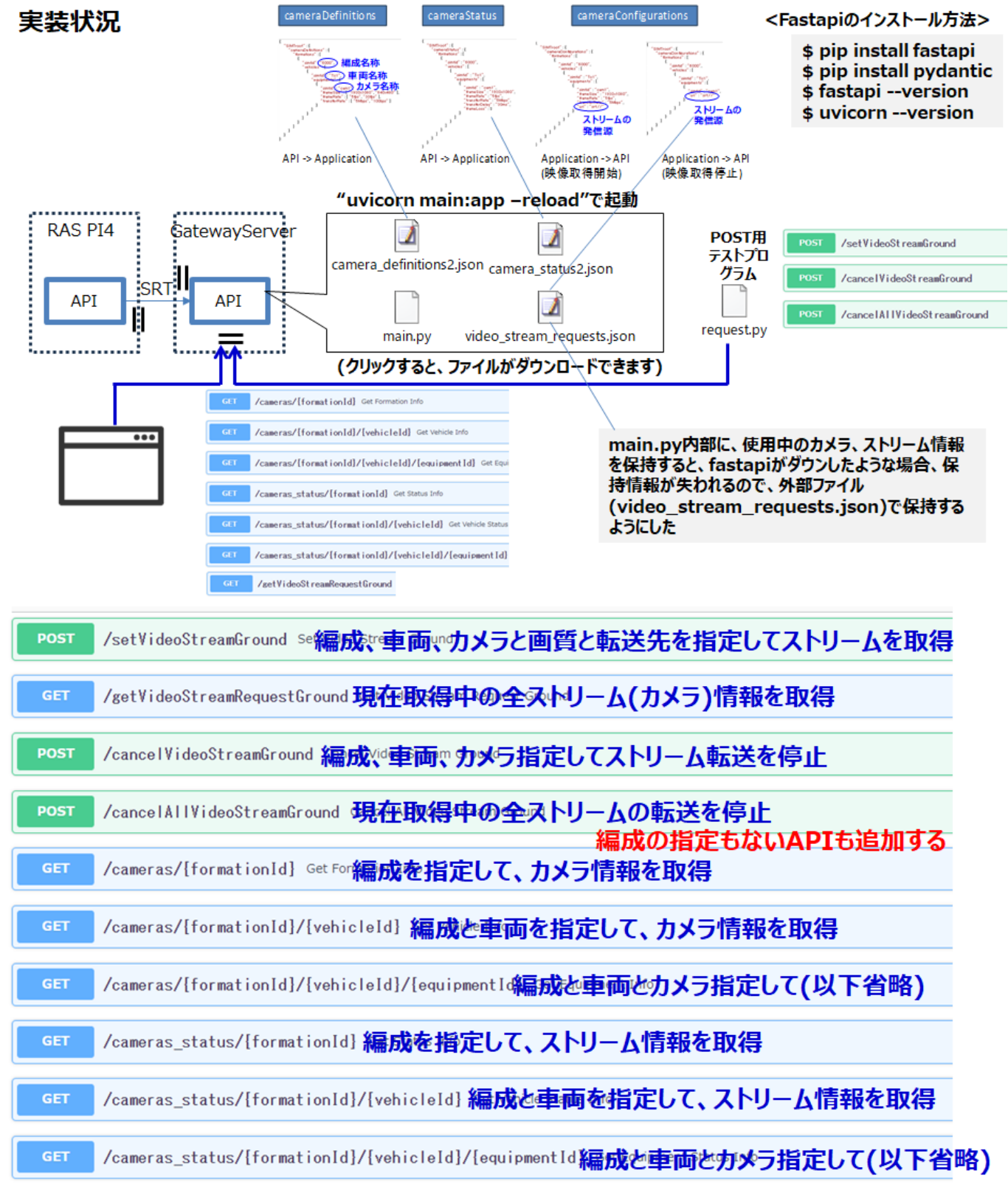
- 「TCP」を選択し、ポート番号を指定します（例: 8000）。「次へ」をクリックします。
- アクションとして「接続を許可」を選択し、「次へ」をクリックします。
- ルールの名前を入力し、必要に応じて説明を追加します。次に「完了」をクリックします。

5. APIの概要

5.1. 開発環境を例とした構成ファイルの表示



5.2. 地上APIの概要



5.3. 車上APIの概要

6. APIの動作確認方法

6.1. setVideoStreamGroud

```
$ python setVideoStreamGround.py
```

```
import requests

# FastAPIのエンドポイントURLを指定
endpoint_url = "http://localhost:8000/setVideoStreamGround"  # エンドポイントのURLを適切に設定

# テストメッセージのデータを作成
test_data = {
    "priority": 1,
    "formationId": "6000",
    "vehicleId": "Tc1",
    "equipmentId": "cam1",
    "width": 1280,
    "framerate": 10,
    "url": "http://example.com/video_stream",
    "port": 8080
}

# POSTリクエストを送信
response = requests.post(endpoint_url, json=test_data)

# レスポンスを表示
print("Response Status Code:", response.status_code)
print("Response JSON Data:", response.json())
```

6.2. cancelVideoStreamGroud

```
$ python cancelVideoStreamGround.py
```

```
import requests
import json

# サーバーのベースURL（エンドポイントを含まない部分）
# base_url = "http://サーバーのアドレス:ポート番号"
base_url = "http://localhost:8000"

# リクエストボディのデータを作成
request_data = {
    "priority": 1,
    "formationId": "6000",
    "vehicleId": "Tc1",
    "equipmentId":"cam1",
    "width": 1280,
    "framerate": 10,
```

```
    "url": "10.2.0.4",
    "port": 38089
}

# リクエストボディの値を表示
print("リクエストボディのデータ:")
print(json.dumps(request_data, indent=4))

# HTTP POST リクエストを送信
try:
    response = requests.post(f"{base_url}/cancelVideoStreamGround",
json=request_data)
    response_data = response.json()

    # ステータスコードによる応答の処理
    if response.status_code == 200:
        print("ビデオストリーミングの要求が成功しました。")
        print("応答データ:", response_data)
    else:
        print("ビデオストリーミングの要求が失敗しました。")
        print("エラーメッセージ:", response_data)
except requests.exceptions.RequestException as e:
    print("リクエストの送信中にエラーが発生しました:", e)
```

6.3. cancelAllVideoStreamGroud

```
$ python cancelAllVideoStreamGround.py
```

```
import requests
import json

# サーバーのベースURL（エンドポイントを含まない部分）
# base_url = "http://サーバーのアドレス:ポート番号"
base_url = "http://localhost:8000"

# リクエストボディのデータを作成
request_data = {
    "priority": 1,
    "formationId": "6001",
    "vehicleId": "Tc22",
    "equipmentId":"cam2",
    "width": 1280,
    "framerate": 10,
    "url": "10.2.0.4",
    "port": 38089
}

# リクエストボディの値を表示
print("リクエストボディのデータ:")
```

```
print(json.dumps(request_data, indent=4))

# HTTP POST リクエストを送信
try:
    response = requests.post(f"{base_url}/cancelAllVideoStreamGround",
json=request_data)
    response_data = response.json()

    # ステータスコードによる応答の処理
    if response.status_code == 200:
        print("ビデオストリーミングの要求が成功しました。")
        print("応答データ:", response_data)
    else:
        print("ビデオストリーミングの要求が失敗しました。")
        print("エラーメッセージ:", response_data)
except requests.exceptions.RequestException as e:
    print("リクエストの送信中にエラーが発生しました:", e)
```

6.4. getVideoStreamRequestGround

```
$ curl http://localhost:8000/getVideoStreamRequestGround
```

6.5. cameras

```
$ curl http://localhost:8000/cameras
```

```
$ curl http://localhost:8000/cameras/6000
```

```
$ curl http://localhost:8000/cameras/6000/Tc1
```

```
$ curl http://localhost:8000/cameras/6000/Tc1/cam1
```

6.6. cameras_status

```
$ curl http://localhost:8000/cameras_status/6000
```

```
$ curl http://localhost:8000/cameras_status/6000/Tc1
```

```
$ curl http://localhost:8000/cameras_status/6000/Tc1/cam1
```

7. 構成用ファイル

7.1. camera_definitions2.json

編成、車両に設置されているデバイス(カメラ)のスペック情報が記載されている構成情報ファイル。

```
{
  "SIMTroot": {
    "cameraDefinitions": {
      "formations": [
        {
          "simtid": "6000",
          "vehicles": [
            {
              "simtid": "Tc1",
              "equipments": [
                {
                  "simtid": "cam1",
                  "frameSize": ["1280x720", "640x360"],
                  "frameRate": ["2fps", "5fps", "10fps"],
                  "transferRate": ["5Mbps", "100bps"]
                },
                {
                  "simtid": "cam2",
                  "frameSize": ["1280x720", "640x360"],
                  "frameRate": ["2fps", "5fps", "10fps"],
                  "transferRate": ["5Mbps", "100bps"]
                }
              ]
            },
            {
              "simtid": "Tc2",
              "equipments": [
                {
                  "simtid": "cam3",
                  "frameSize": ["1920x1080", "720x480"],
                  "frameRate": ["3fps", "7fps", "15fps"],
                  "transferRate": ["10Mbps", "200bps"]
                }
              ]
            }
          ]
        },
        {
          "simtid": "6001",
```

```
          "vehicles": [
            {
              "simtid": "Tc11",
              "equipments": [
                {
                  "simtid": "cam22",
                  "frameSize": ["1280x720", "640x360"],
                  "frameRate": ["2fps", "5fps", "10fps"],
                  "transferRate": ["5Mbps", "100bps"]
                }
              ]
            },
            {
              "simtid": "Tc12",
              "equipments": [
                {
                  "simtid": "cam23",
                  "frameSize": ["1280x720", "640x360"],
                  "frameRate": ["2fps", "5fps", "10fps"],
                  "transferRate": ["10Mbps", "200bps"]
                }
              ]
            }
          ]
        }
      ]
    }
  }
}
```

7.2. camera_status2.json

現状のカメラの稼動状態を記録している。

```
{
  "SIMTroot": {
    "cameraDefinitions": {
      "formations": [
        {
          "simtid": "6000",
          "vehicles": [
            {
              "simtid": "Tc1",
              "equipments": [
                {
                  "simtid": "cam1",
                  "frameRate": ["2.2fps"],
                  "transferRate": ["5.5Mbps"],
                  "frameDropRate": ["3.33%"]
                },
                {
```

```
                "simtid": "cam2",
                "frameSize": ["N/A"],
                "frameRate": ["N/A"],
                "transferRate": ["N/A"],
                "frameDropRate": ["N/A"]
              }
            ]
          },
          {
            "simtid": "Tc2",
            "equipments": [
              {
                "simtid": "cam3",
                "frameSize": ["640x360"],
                "frameRate": ["7.7fps"],
                "transferRate": ["5.5%"],
                "frameDropRate": ["3.33%"]
              }
            ]
          }
        ]
      },
      {
        "simtid": "6001",
        "vehicles": [
          {
            "simtid": "Tc11",
            "equipments": [
              {
                "simtid": "cam22",
                "frameSize": ["N/A"],
                "frameRate": ["N/A"],
                "transferRate": ["N/A"],
                "frameDropRate": ["N/A"]
              }
            ]
          },
          {
            "simtid": "Tc12",
            "equipments": [
              {
                "simtid": "cam23",
                "frameSize": ["640x360"],
                "frameRate": ["10.1fps"],
                "transferRate": ["8.8Mbps"],
                "frameDropRate": ["5.55%"]
              }
            ]
          }
        ]
      }
    ]
  }
```

```
    }
}
```

7.3. video_stream_requests.json

現在稼動中のストリームを記録してしているもの。ストリームを停止すると自動的に消去される。

setVideoStreamGroudの発行で登録され、cancelVideoStreamGroud, cancelAllVideoStreamGroudの発行で削除される。

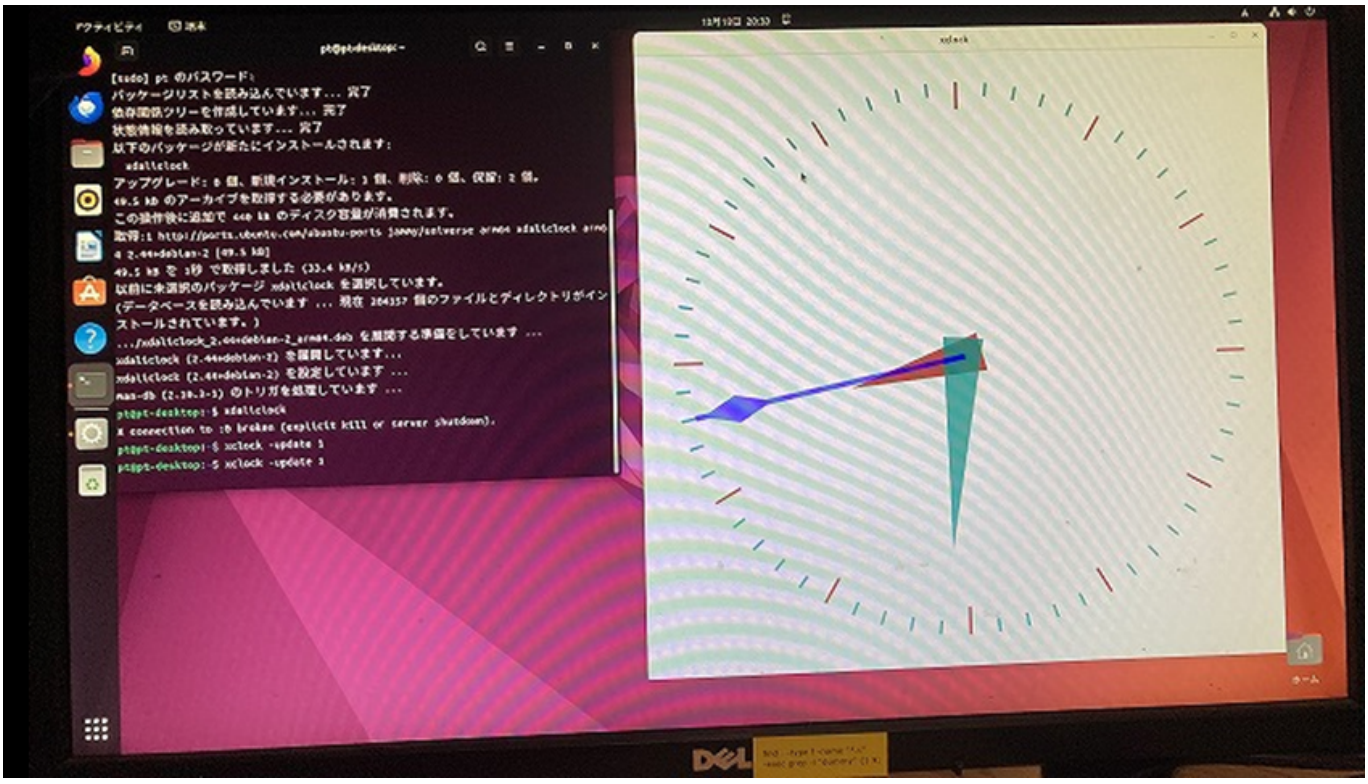
```
{"priority": 1, "formationId": "6000", "vehicleId": "Tc1", "equipmentId": "cam1",
"width": 1280, "framerate": 10, "url": "http://example.com/video_stream", "port":
8080}
```

8. その他

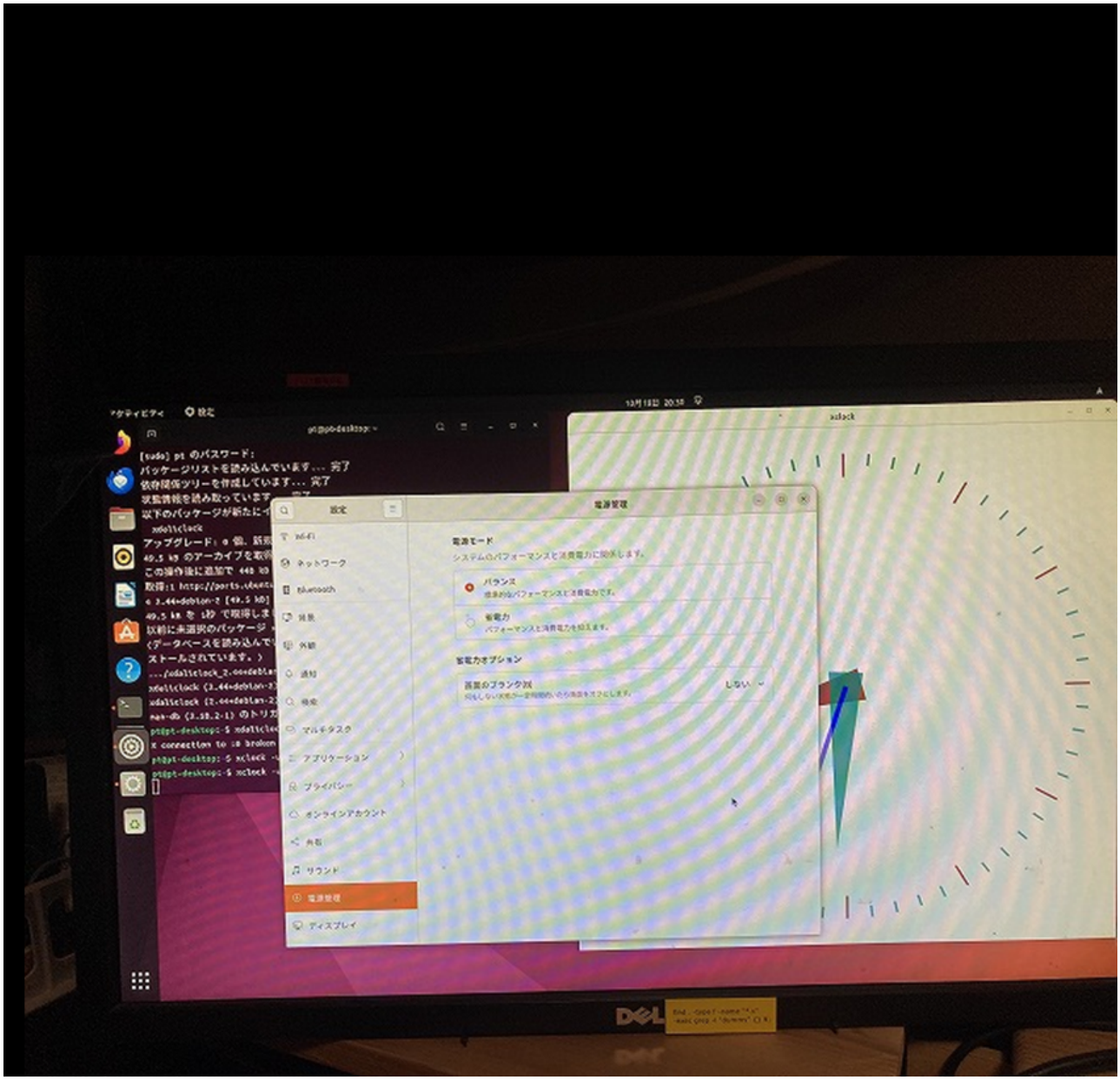
8.1. ubuntuのGUIでxclockでアナログ時計を表示し、表示し続ける方法

- 映像の遅延を測定する為に時計を表示する方法

```
xclock -update 1
```



- 時計を表示し続ける方法



- 「Linux でミリ秒まで表示するワンライナー時計」 私の場合、コンマ秒までを表示したいので、

```
while true ; do printf "\r%.10s" `date +%T.%N`; sleep 0.01 ; done
```

となります。 Ubuntuの端末で文字を大きくするには、以下の方法があります。

端末フォントサイズの変更: 通常、端末のフォントサイズを変更することで文字を大きくできます。以下の手順を実行します：

端末を開きます（通常、Ctrl+Alt+Tで起動します）。 メニューバーから「編集」をクリックし、「プロファイルの設定」を選択します。 「テキスト」タブを選択し、フォントサイズを変更します。

で、まあ、こんな風に簡単に作りました。