

多種多様なシステムをRSNP通信可能にする汎用ユニット サービス利用マニュアル Ver. 1.0

芝浦工業大学 理工学研究科 機械工学専攻 知能機械システム研究室 岡野 憲, 松日楽 信人

改善点などのご意見があれば、下記までご連絡ください。

連絡先：

芝浦工業大学 機械機能工学科 知能機械システム研究室

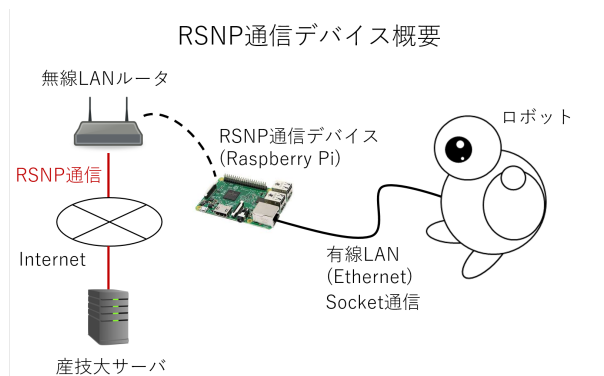
〒135-8548 東京都江東区豊洲3-7-5

機械工学専攻 修士2年 岡野 憲 Okano Satoshi

E-mail:md18020@shibaura-it.ac.jp

1. はじめに

近年、労働人口の減少等から、その補完としてロボットの活用が期待されている。平成22年に発表されたNEDOの資料[1]では、ロボット市場の拡大がされている。ゆえに、今後、ロボットの台数が増加するのも必然である。そこで、増加した多数のロボットを管理、監視するためのシステムが必要になってくる。また、各ロボットからデータを取得することで、そのデータを活用した様々なサービスを期待することができる。そこで、本提案で開発した汎用ユニット(以下、「RSNPユニット」と記載)を、多種多様なロボットやデバイスに外付けで接続することで、取得したデータをRSNP(Robot Service Networking Protocol)[2]通信でインターネット経由でサーバにアップロードして蓄積し、Webブラウザ等のGUI上で各ロボットの状態を管理、監視することができる。以下の図のようにRSNPユニットをロボットに接続して使用することが可能である。



※現状、RSNPユニットは、産業技術大学院大学(品川)サーバの次のエンドポイントへ接続します。

<http://robots.aiit.ac.jp:8080/UpdateNotificationState/services>

2. ユニットを使用するための準備

ユニットを使用するためにいくつかのソフトを予め、ダウンロード、インストール、設定する必要があります。ご了承ください。

2.1 RSNPユニットの電源投入

まず、RSNPユニットの電源を入れます。電源ボタンは搭載していないため以下の図に示すように、microUSBにusbケーブルを接続します。

2.2 RSNPユニットとPCとの接続

RSNPユニットの初期設定を行うために、PCと有線で接続します。
現状、LANケーブルとUSBケーブルで接続する2通りの方法があります。

ケース1-LANケーブルでの接続

LANケーブルでPCに接続するために、以下の図に示すように配線します。ケーブルの種類は、クロスカストレートのどちらでも接続可能です。

ケース2-USBケーブルでの接続 USBケーブルでPCに接続するために、以下の図に示すように配線します。

2.3 RSNPユニットにSSH接続する

ケース1-Linux, Mac OSの場合

Linuxを使用している場合、次のコマンドを実行することで、SSH接続することができます。

```
~$ ssh raspberrypi
```

ケース2-Windowsの場合

2.3.1 Tera Termのダウンロード&インストール

RSNPユニットにリモートでSSH接続するためのソフトウェアが必要になります。
今回は、クライアントソフトウェアとして**Tera Term**を使用します。
以下のサイトより、ダウンロードとインストールを行ってください。

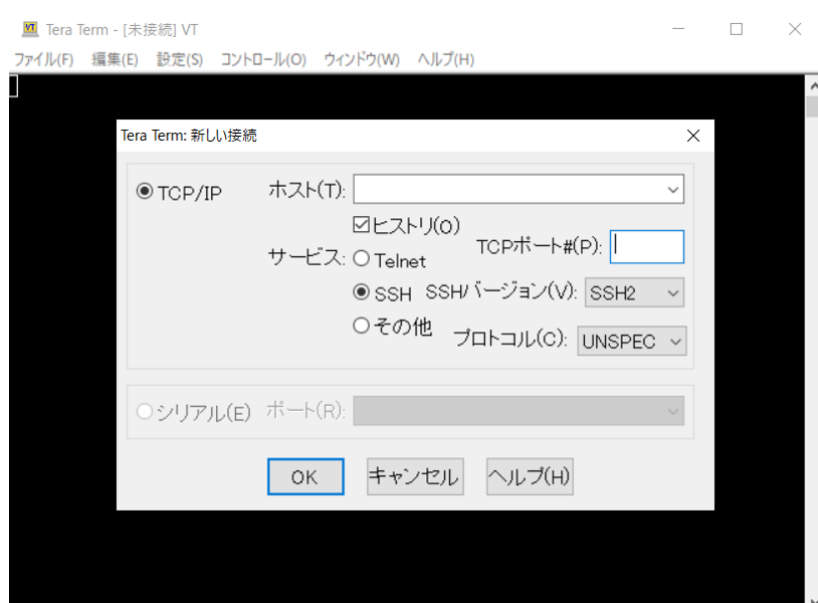
窓の杜 Tera Term

<https://forest.watch.impress.co.jp/library/software/utf8teraterm/>

2.3.2 RSNPユニットに接続

次に、インストールしたTera Termを立ち上げます。

以下の図のような画面が表示されます。

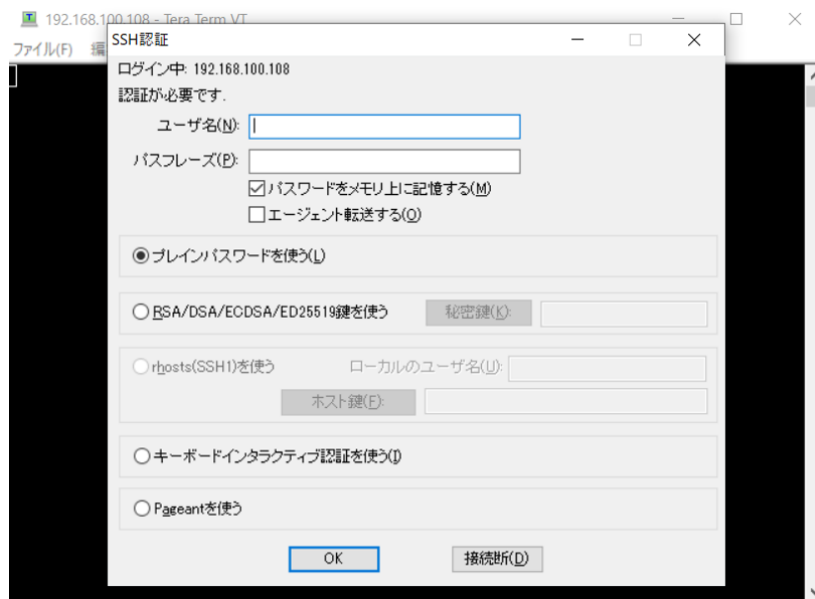


ホストに"raspberrypi"と、TCPポートに"22"と入力し、「OK」をクリックします。

2.3.3 Raspberry Piにログイン

次にRaspberry Piにログインをします。

上記で「OK」をクリック後に以下のような画面が表示されます。



ユーザ名に"pi"とパスフレーズに"8073"と入力し、「OK」をクリックします。

ケース1, ケース2-共通

RSNPユニットにSSH接続すると以下のような画面が表示されます。

2.4 java(jdk)のインストール

java(jdk)をインストールします。以下のようにコマンドを入力し実行します。

```
~$ sudo apt-get install java-1.8.0-openjdk
```

java(jdk)がインストールされたか念のため確認します。以下のようにコマンドを入力し実行します。

```
~$ java -version
```

これでバージョンが表示されれば、インストール完了です。

2.5 Githubからリポジトリをクローンする

必要なファイルをダウンロードします。

任意のディレクトリ以下に次のようにコマンドを入力し実行します。

```
~$ git clone https://github.com/SatoshiOkano/RSNPUnit.git
```

次に、ダウンロードされたか確認するため、以下のようにコマンドを入力します。

```
~$ ls
```

ダウンロードされていれば、"RSNPUnit"というディレクトリが存在します。

2.6 propertiesファイルの設定

ダウンロードしたディレクトリ内に"DataLog"というディレクトリがあるので、そこに移動します。

以下のようにコマンドを入力し実行します。

```
~$ cd /RSNPUnit/DataLog
```

移動すると, "**Config.properties**" というファイルがあります.

次に, 以下のようにコマンドを入力します.

```
~$ sudo nano Config.properties
```

※ファイルを編集するためのエディタとして今回は"nano"を使用していますが, 好みのものを使用してください.

デフォルトでは, 以下のように記述されています.

```
#Configuration
robot_id  = 1
end_point = http://robots.aiit.ac.jp:8080/UpdateNotificationState/services
send_interval = 10000
ip_address = 169.254.183.9
port = 8000
```

各パラメータの意味は, 次のようになっています.

- **robot_id** : ロボットの識別ID
- **end_point** : サーバのアドレス
- **send_interval** : 送信時間間隔
- **ip_address** : RSNPユニット本体のIPアドレス
- **port** : Socket通信のポート番号

必要に応じて, これらの各パラメータを変更します.

3.7 プログラムの実行

まず, 「**RSNPcomms.jar**」を実行します. 「**RSNP_lecture**」ディレクトリに戻るため, 以下のようにコマンドを入力します.

```
~$ cd..
```

ファイルが存在しているか確認するため, 以下のコマンドを入力します.

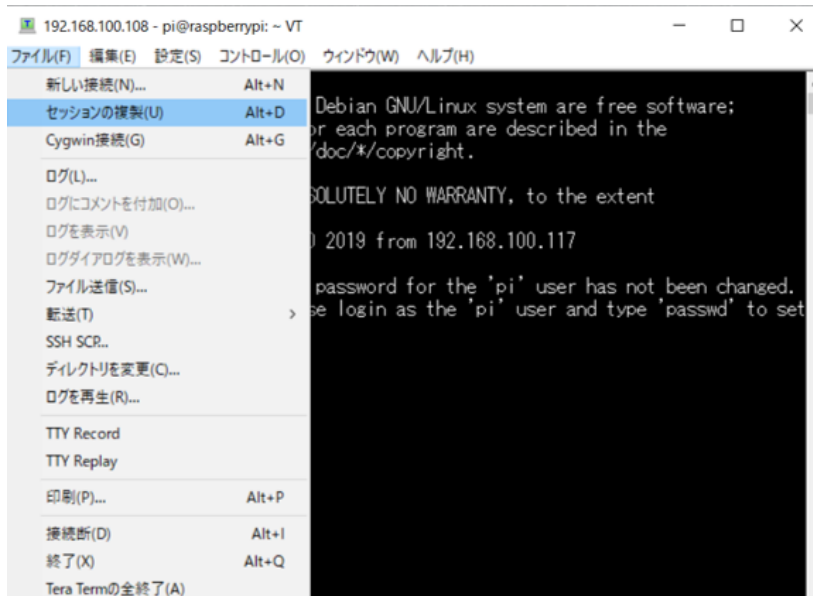
```
~$ ls
```

「**RSNPcomms.jar**」, 「**Sensing.py**」が有ります.

次に, 実行するために以下のようにコマンドを入力します.

```
~$ java -jar RSNPcomms.jar
```

実行が完了したら, 別のウィンドウでコマンドを打つために, セッションの複製を行います. Tera Termの左上のファイルをクリックし, 以下の図のように「セッションの複製」をクリックします.



すると、新たにウィンドウが開かれます。そこに、「Sensing.py」を実行するために以下のようにコマンドを入力します。

```
~$ python Sensing.py
```

実行するとウィンドウに以下のように文字列が定期的に表示されます。

```
switch OFF
{"data":[{"ac_id":1,"ac":"sensor status","re_id":1,"re":"off","co":""}]}
```

表記されているデータに関しては**4. 通信仕様**に記述されています。

3.8 状態の確認

センサの状態がサーバに送信され反映されているか確認します。






以下のURLにアクセスします。

<http://robots.aiit.ac.jp:8080/Robomech2019/>

以下のようにブラウザ上で表示されていれば、確認完了です。

Raspberry Pi②を稼働させた例。

Raspberry Pi稼働状況

【最終更新日時】	2019年6月3日(月)19時21分	【更新時間】	3秒後
	Raspberry Pi①の稼働状況：未稼働 センサ：no data		
	Raspberry Pi②の稼働状況：稼働中 センサ：off		
	Raspberry Pi③の稼働状況：未稼働 センサ：no data		
	Raspberry Pi④の稼働状況：未稼働 センサ：no data		
	Raspberry Pi⑤の稼働状況：未稼働 センサ：no data		

タクトスイッチを押し続け、センサ状態が**on**になるか、確認してください。

4. ネットワーク設定

RSNP通信は外部のインターネットに接続するのが前提です。

そのため、ここでは無線LANネットワーク設定をします。

まず、接続するルータ等のSSIDとパスワードを調べます。

次に、以下のconfファイルをエディタで編集します。

```
~$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

wpa_supplicant.confを次のように記述します。

```
network={
    ssid="SSIDを記述する"
    psk="パスワードを記述する"
    key_mgmt=WPA-PSK
}
```

次に、Raspberry Piの無線LANを再起動します。以下のようにコマンドを入力します。

```
~$ sudo ifdown wlan0
```

数秒すると無線LANはオフになるので、以下のようにコマンドを入力します。

```
~$ sudo ifup wlan0
```

以上で、有線ネットワークが接続可能になります。

5. 通信データ仕様

RSNP通信デバイスとロボット間のデータのやり取りはsocket通信で行います。

ただし、以下の5種類のデータで送信を行う必要があります。

- **Action_id**
- **Action名**
- **Result_id**
- **Resultデータ**
- **コメント**

Action_idとは、**Action名**に対する紐づけ番号である。

Action名とは、ロボットが行った動作名などである。

Result_idとは、**Resultデータ**に対する紐づけ番号である。

Resultデータとは、ロボットから得たデータ(変数)などである。

コメントとは、コメント記述を入れたい場合に用いる。

例えば、挨拶を3回、人数カウントを5人としたロボットがあったとする。この場合、データの仕様は次のようになる。

仕様名	データ1	データ2
Action_id	1	2
Action名	挨拶回数	人数
Result_id	1	2

仕様名	データ1	データ2
Resultデータ	3	5
コメント	無し	無し

ここで、実際のデータ形式は以下のようなjson形式としています。 {...}内において、先頭に「"data":」があり、その次に配列のカッコ([])内において、1種類のデータが配列の1つの要素に入ります。ダブルクォーテーション(")で囲んだ仕様名と値をカンマ(:)で区切ります。3点(...)には、対応するデータ等が入ります。見やすいように改行してありますが、実際は1行でデータ送信してください。これ以外の仕様でのデータを送信するとRaspi側で受信できないのでご注意ください。

```
{
  "data":
  [
    {
      "ac_id": ... ,
      "ac": ... ,
      "re_id": ... ,
      "re": ... ,
      "co": ...
    },
    {...},
    ...
  ]
}
```

仕様名は以下の表のように短縮形となっているのでご注意ください。

仕様名	省略形
Action_id	ac_id
Action名	ac
Result_id	re_id
Resultデータ	re
コメント	co

上記のロボットの例の場合は、

```
{"data":[{"ac_id":1,"ac":"挨拶回数","re_id":1,"re":3,"co":""},{ "ac_id":2,"ac":"人数","re_id":2,"re":5,"co":""}]}
```

となります。(コメントは無しのため、空欄(""))となっている)

複数種類の場合は、配列の成分が増加し、

```
{"data":[{"...},{...},{...},...]}
```

となる。今回は5種類まで対応している。