

# NEC 無線通信可視化ソフトウェア 構築マニュアル

2024 年 3 月 15 日  
日本電気株式会社

改版履歴

版数	日付	改版内容
1.0.0	2024/3/15	初版発行

## 目次

1.	はじめに .....	4
1.1.	本書の位置づけ .....	4
1.2.	参照ドキュメント .....	4
2.	システム構成 .....	5
2.1.	サーバ動作環境 .....	5
2.2.	センサ動作環境 .....	6
3.	サーバ構築手順 .....	7
3.1.	バックエンドプログラム インストール .....	7
3.2.	解析プログラム インストール .....	28
3.3.	フロントエンドプログラム インストール .....	32
3.4.	プロセス監視 設定 .....	37
3.5.	時刻同期サーバ設定 .....	39
3.6.	ファイアウォール設定 .....	39
3.7.	詳細設定 .....	40
3.8.	バックエンドプログラム Tomcat 更新時の注意事項 .....	43
4.	センサ構築手順 .....	44
4.1.	時刻同期クライアント設定 .....	44
4.2.	センサプログラム インストール .....	45
4.3.	センサプログラム 起動・停止方法 .....	52
4.4.	センサ 高可用化 .....	53
4.5.	ファイアウォール設定 .....	68
5.	問い合わせ窓口 .....	70
5.1.	お問い合わせ先 .....	70
5.2.	受付時間 .....	70

## 1. はじめに

### 1.1. 本書の位置づけ

本書は、NEC 無線通信可視化ソフトウェアのインストール手順について説明します。  
本書は Linux について基本的な知識(特にネットワーク関連)を有している方を想定しています。

本文書は日本電気株式会社(以下、NEC)の許可なくコピー及びその配布、ホームページへの掲載を禁じます。  
また、当社は本ソフトウェア仕様について随時変更することができるものとします。  
本書記載の内容は 2024 年 3 月時点の内容を元に作成しています。今後のソフトウェアのアップデートや個別の設定により画面や文言が変更になることがあります。

### 1.2. 参照ドキュメント

参照ドキュメント

ドキュメント名	内容
NEC 無線通信可視化ソフトウェアユーザーズマニュアル	NEC 無線通信可視化ソフトウェアのインストール後、設定を行う際に参照してください。

## 2. システム構成

無線通信可視化ソリューションは、無線 LAN 通信を収集するセンサと、センサが解析した情報を蓄積・集計・解析しブラウザを使用して無線通信状況を視覚的に表示するサーバで構成されます。

無線通信状況を可視化したい環境に複数台のセンサを設置していただきます。各センサは LTE、無線 LAN、有線 LAN などのネットワークを通じてサーバに情報を通知します。サーバで解析された情報はブラウザで確認できるためサーバから離れた場所でも確認することができます。



構成図

サーバの構築をしてから、センサを構築してください。システム構築後の可視化するために必要な設定情報をユーザ様に提供してください。

### 2.1. サーバ動作環境

サーバソフトウェア動作環境は以下のとおりです。

動作環境(SW)

OS	Ubuntu 20.04
OpenJDK	11
Tomcat	9
MongoDB	4.0 以上
BaaS	7.5.6
RabbitMQ	3.9 以上
Nodejs	18
python	3.9
nginx	1.24

動作環境(HW)

Series	Mate & VersaPro
CPU	Core i5 以上
Memory	16GB 以上
Storage	500GB 以上

## 2.2. センサ動作環境

サーバソフトウェア動作環境は以下のとおりです。

動作環境(SW)

OS	Raspberry Pi OS 64-bit Lite (Bullseye)
Nodejs	18
python	3.9

動作環境(HW)

Series	Raspberry Pi 4 model B
CPU	Quad-core 64-bit SoC
Memory	2GB 以上
Storage	32GB 以上
Wireless LAN	NEC Aterm WL900U TP-Link Archer T9UH TP-Link Archer TX20UH ※1 BUFFALO WI-U3-1200AX2 ※1 ELECOM WDC-X1201DU3 ※1 ASUS USB-AX56 ※1 NETGEAR A8000 ※2

※1：2024 年 3 月現在、4.2.2 章に例示したドライバ(lwfinger/rtl8852au)では受信強度(RSSI)情報の取得がマネジメントフレームのみに限定されることを確認しています。そのため端末位置推定機能が制限されます。

※2：2024 年 3 月現在、本ソフトウェアでセンシングに利用可能な周波数帯は 2.4GHz 帯および 5GHz 帯のみになります。

### 3. サーバ構築手順

サーバは大きく分けて3種類のプログラムで構成されています。1つ目データの蓄積や蓄積したデータベースへのインターフェースを提供するバックエンドプログラム、2つ目はセンサから収集・蓄積したデータを解析処理する解析プログラム、3つ目は解析したデータを視覚的に表示するためのフロントエンドプログラム、これら3種類のプログラムが連携して動作することでWeb サービスを提供します。

バックエンド、解析、フロントエンドの順番でプログラムのインストールを実施します。構築はroot権限が必要なコマンドを“sudo”コマンドで実行でき、またインターネットに接続できる環境で実施してください。

#### 3.1. バックエンドプログラム インストール

ここではバックエンドプログラムをインストールする手順を説明します。

##### 3.1.1. ulimit の設定

システムリソースの制限に関する設定をします。以下の設定ファイルを新規作成してください。

```
$ sudo vi /etc/security/limits.d/99-mongodb-nproc.conf
```

/etc/security/limits.d/99-mongodb-nproc.conf

```
*      soft  nproc   64000
*      hard  nproc   64000
*      soft  nofile  64000
*      hard  nofile  64000
```

##### 3.1.2. OpenJDK のインストール

OpenJDK をインストールします。

```
$ sudo apt update
$ sudo apt install -y default-jdk
```

##### 3.1.3. Tomcat のインストール

Tomcat を実行するユーザを作成します。

```
$ sudo useradd -m -U -d /opt/tomcat -s /bin/false tomcat
```

Tomcat のバイナリを取得し、/opt/tomcat にインストールします。Tomcat のバージョン番号は Tomcat 9 の最新バージョンに読み替えてください。Tomcat9 の最新バージョンは[公式サイト](#)で確認できます。

インストール手順の例を記載しますが、手順は変更されることがあるため最新のインストール手順は[公式サイト](#)を参照してください。

```
$ VERSION=9.0.87
$ wget https://archive.apache.org/dist/tomcat/tomcat-9/v${VERSION}/bin/apache-
tomcat-${VERSION}.tar.gz -P /tmp
$ sudo tar -xf /tmp/apache-tomcat-${VERSION}.tar.gz -C /opt/tomcat/
$ sudo chown -R tomcat:tomcat /opt/tomcat
$ sudo ln -s /opt/tomcat/apache-tomcat-${VERSION} /opt/tomcat/latest
```

Tomcat の起動設定をするため、systemd 用の Unit ファイルを作成します。以下のファイルを新規作成してください。

```
$ sudo vi /etc/systemd/system/tomcat.service
```

/etc/systemd/system/tomcat.service

```
[Unit]
Description=Tomcat 9 application server
After=network.target

[Service]
Type=forking

User=tomcat
Group=tomcat

Environment="JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64"
Environment="JAVA_OPTS=-Djava.security.egd=file:///dev/urandom -
Djava.awt.headless=true"

Environment="CATALINA_BASE=/opt/tomcat/latest"
Environment="CATALINA_HOME=/opt/tomcat/latest"
Environment="CATALINA_PID=/opt/tomcat/latest/temp/tomcat.pid"
Environment="CATALINA_OPTS=-Xmx6144M -Xms1024M -server -
XX:+UseParallelGC"

ExecStart=/opt/tomcat/latest/bin/startup.sh
ExecStop=/opt/tomcat/latest/bin/shutdown.sh
ExecReStart=/opt/tomcat/bin/shutdown.sh;/opt/tomcat/bin/startup.sh

[Install]
WantedBy=multi-user.target
```

Tomcat を有効化、および起動します。

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable tomcat.service
$ sudo systemctl start tomcat.service
```



### 3.1.4. MongoDB のインストール

MongoDB をインストールします。

インストール手順の例を記載しますが、手順は変更されることがあるため最新のインストール手順は[公式ドキュメント](#)を参照してください。

MongoDB 専用のリポジトリからダウンロードするため、最初にパッケージの検証に利用する gpg キーを取得し、その後 apt ソースにリポジトリを追加し、インストールを実行します。

MongoDB の最新安定バージョンの公開 gpg キーをインポートします。インストールするバージョンに合わせて、URL のバージョン部分を置き換えてください。

```
$ curl -fsSL https://www.mongodb.org/static/pgp/server-7.0.asc | ¥  
sudo gpg -o /usr/share/keyrings/mongodb-server-7.0.gpg --dearmor
```

続いて、MongoDB のリポジトリを追加し、インストールします。

インストール環境のアーキテクチャ、インストールする MongoDB のバージョンに合わせて置き換えてください。アーキテクチャが “x86\_64” の場合は “amd64” を、“aarch64” の場合は “arm64” を指定してください。以下は アーキテクチャ “x86\_64”、MongoDB のバージョン “7.0” の場合の手順になります。

```
$ echo "deb [ arch=amd64 signed-by=/usr/share/keyrings/mongodb-server-7.0.gpg ]  
https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/7.0 multiverse" | ¥  
sudo tee /etc/apt/sources.list.d/mongodb-org-7.0.list  
$ sudo apt update  
$ sudo apt install -y mongodb-org=7.0.7
```

MongoDB を有効化、および起動します。

```
$ sudo systemctl enable mongod  
$ sudo systemctl start mongod
```

### 3.1.5. unzip のインストール

unzip をインストールします。

```
$ sudo apt install -y unzip
```

### 3.1.6. OpenSSL のアップデート

OpenSSL を OpenSSL3 にアップデートします。最新バージョンは[公式サイト](#)を確認してください。最新の手順はダウンロードしたソースコード内の “INSTALL.md” を参照してください。

```
$ cd /usr/local/src/
$ sudo wget https://www.openssl.org/source/openssl-3.2.1.tar.gz
$ sudo tar xvf openssl-3.2.0.tar.gz
$ sudo apt install build-essential
$ sudo ./config
$ sudo make
$ sudo make test
$ sudo make install
$ sudo cat <<- __EOF__ | tee -a /etc/ld.so.conf.d/openssl-3.conf
/usr/local/lib64
__EOF__
$ sudo ldconfig
```

### 3.1.7. BaaS のデプロイ

NEC モバイルバックエンド基盤をデプロイします。

REST API サーバ api.war と デベロッパーコンソールサーバ console.war を Tomcat にデプロイし、Tomcat を再起動します。

```
$ sudo wget -qO /opt/tomcat/latest/webapps/api.war ¥
https://github.com/nec-baas/baas-server/releases/download/v7.5.6/api.war
$ sudo wget -qO /opt/tomcat/latest/webapps/console.war ¥
https://github.com/nec-baas/baas-server/releases/download/v7.5.6/console.war
$ sudo chown tomcat:tomcat /opt/tomcat/latest/webapps/api.war
$ sudo chown tomcat:tomcat /opt/tomcat/latest/webapps/console.war
$ sudo systemctl restart tomcat
```

### 3.1.8. RabbitMQ のインストール

RabbitMQ をインストールします。インストール手順の例を記載しますが、最新の手順は[公式サイト](#)を参照してください。

```
$ curl -1sLf "https://github.com/rabbitmq/signing-keys/releases/download/3.0/rabbitmq-release-signing-key.asc" | sudo gpg --dearmor | ¥
sudo tee /usr/share/keyrings/com.github.rabbitmq.signing.gpg > /dev/null
$ curl -1sLf
"https://keyserver.ubuntu.com/pks/lookup?op=get&search=0xf77f1eda57ebb1cc" | ¥
sudo gpg --dearmor | ¥
sudo tee /usr/share/keyrings/net.launchpad.ppa.rabbitmq.erlang.gpg > /dev/null
$ sudo apt-get install apt-transport-https
$ sudo tee /etc/apt/sources.list.d/rabbitmq.list <<EOF
deb [signed-by=/usr/share/keyrings/net.launchpad.ppa.rabbitmq.erlang.gpg]
http://ppa.launchpad.net/rabbitmq/rabbitmq-erlang/ubuntu focal main
deb-src [signed-by=/usr/share/keyrings/net.launchpad.ppa.rabbitmq.erlang.gpg]
http://ppa.launchpad.net/rabbitmq/rabbitmq-erlang/ubuntu focal main
EOF
$ sudo apt-get update -y
$ sudo apt-get install -y erlang-base ¥
erlang-asn1 erlang-crypto erlang-eldap erlang-ftp erlang-inets ¥
erlang-mnesia erlang-os-mon erlang-parsetools erlang-public-key ¥
erlang-runtime-tools erlang-snmp erlang-ssl ¥
erlang-syntax-tools erlang-tftp erlang-tools erlang-xmerl
```

rabbitmq ユーザを追加し、administrator 権限を付与します。

```
$ sudo rabbitmqctl add_user "rabbitmq" "rabbitmq"
$ sudo rabbitmqctl set_permissions -p / rabbitmq ".*" ".*" ".*"
$ sudo rabbitmqctl set_user_tags rabbitmq administrator
```

### 3.1.9. SSE Push サーバのインストール

SSE Push サーバ ssepush.war を Tomcat にデプロイします。

```
$ sudo wget -qO /tmp/ssepush-server-7.5.1.tar.gz ¥
https://github.com/nec-baas/ssepush-server/releases/download/v7.5.1/ssepush-server-
7.5.1.tar.gz
$ tar zxvf /tmp/ssepush-server-7.5.1.tar.gz -C /tmp/
$ sudo mv /tmp/ssepush-server-7.5.1/ssepush.war ¥
/opt/tomcat/latest/webapps/ssepush.war
$ sudo chown tomcat:tomcat /opt/tomcat/latest/webapps/ssepush.war
```

SSE Push サーバの設定ファイル /etc/ssepush/config.xml を作成します。

```
$ sudo mkdir /etc/ssepush
$ sudo vi /etc/ssepush/config.xml
```

/etc/ssepush/config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <entry key="amqpUri">amqp://rabbitmq:rabbitmq@localhost:5672</entry>
  <entry key="heartbeatIntervalSec">30</entry>
  <!--
  <entry key="hazelcastConfig">file:/etc/ssepush/hazelcastConfig.xml</entry>
  -->
</properties>
```

バックエンドプログラムとの接続設定ファイル /etc/baas/development.xml を作成します。

```
$ sudo mkdir /etc/baas
$ sudo vi /etc/baas/development.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<!--
```

プロパティ設定 XML ファイル。

本ファイルは、BaaS の設定を行うためのものである。

以下いずれかのディレクトリに **production.xml**, **development.xml**, **test.xml** の名称で配置する。  
複数置いた場合は、下のものが優先される。

- 1) /etc/baas/
- 2) c:/NEC/BaaS/etc/ (Windows)
- 3) ~/.baas/
- 4) [デプロイディレクトリ]/WEB-INF/classes/ (クラスパス)

なお設定値はサーバ起動時のシステムプロパティや環境変数、およびデベロッパーコンソールでも設定できる。複数設定した場合は、下のものが優先される。

- 1) XML ファイル
- 2) 環境変数
- 3) システムプロパティ
- 4) デベロッパーコンソールのシステム設定値

なお、Spring のプロファイル(production/development/test)の切り替えはここではできない。

WAR ファイル内の **web.xml** ファイルを変更するか、

JVM オプション **-Dspring.profiles.active** で起動時に指定する。

logback の設定は、このファイルではなく **logback.xml** ファイルで行う。

-->

<properties>

<!-- API サーバ ベース URI デフォルト設定 -->

<!-- <entry key="api.baseUrl">http://baas.example.com/api</entry> -->

<!-- API サーバ 内部向けベース URI デフォルト設定 -->

<!-- 省略時は api.baseUrl の値が使用される -->

<!-- <entry key="api.internalBaseUrl"></entry> -->

<!-- Console サーバ ベース URI デフォルト設定 -->

<!-- 注: デベロッパーコンソール システム設定の値のほうが優先される -->

<!-- <entry key="console.baseUrl">http://baas.example.com/console</entry> -->

<!--

MongoDB 設定

-->

<!--

MongoDB サーバ。MongoDB Connection String URI で指定する。

("mongodb://" は省略可。)

例 1) ローカル接続

mongodb://localhost:27017

例 2) レプリカセットの場合

```
mongodb://server1:27017,server2:27017,server3:27017
-->
<entry key="mongo.servers">mongodb://localhost:27017</entry>

<!-- MongoDB 認証ユーザ名 -->
<entry key="mongo.username"></entry>
<!-- MongoDB 認証パスワード -->
<entry key="mongo.password"></entry>

<!-- MongoDB ホスト当たり最大コネクション数 -->
<!-- <entry key="mongo.maxConnectionsPerHost">200</entry> -->

<!--
AMQP サーバ設定 (SSE Push/カスタムロジック に必要)。
amqp.{addr,username,password,vhost} を設定する、amqp.uri を設定するか、いずれ
か選択。
両方設定した場合は前者(addr,...)が優先される。
-->
<!-- <entry key="amqp.addr"></entry>
<entry key="amqp.username"></entry>
<entry key="amqp.password"></entry>
<entry key="amqp.vhost"></entry> -->
<entry key="amqp.uri">amqp://rabbitmq:rabbitmq@rabbitmq.localhost:5672</entry>

<!-- API カウント対象外キー : サーバマネージャ側と設定をあわせること -->
<!-- <entry key="system.noChargeKey">sAmPlENoChargeKey12345678</entry> -->

</properties>
```

### 3.1.10. ログ出力先設定

ログ出力先ディレクトリを作成し、所有者を tomcat に変更します。

```
$ sudo mkdir /var/log/baas
$ sudo chown tomcat:tomcat /var/log/baas
$ sudo mkdir /var/log/ssepush
$ sudo chown tomcat:tomcat /var/log/ssepush
```

バックエンドプログラムのログのプロパティファイル /etc/baas/logback.properties を作成します。

```
$ sudo vi /etc/baas/logback.properties
```

/etc/baas/logback.properties

```
logback.level=WARN

logback.types=STDOUT,FILE

logback.logdir=/var/log/baas
```

SSE PUSH サーバのログのプロパティファイル /etc/ssepush/logback.properties を作成します。

```
$ sudo vi /etc/ssepush/logback.properties
```

/etc/ssepush/logback.properties

```
logback.level=WARN  
  
logback.types=STDOUT,FILE  
  
logback.logdir=/var/log/ssepush
```

Bass のログローテーション保存数を変更します。保存数は必要なログ保存期間に合わせて設定してください。以下では 120 日から 7 日に変更する例を記載します。設定ファイルは 3 つとなります

```
$ sudo vi /opt/tomcat/latest/webapps/api/WEB-INF/classes/logback-common.xml  
$ sudo vi /opt/tomcat/latest/webapps/console/WEB-INF/classes/logback-common.xml  
$ sudo vi /opt/tomcat/latest/webapps/ssepush/WEB-INF/classes/logback.xml
```

/opt/tomcat/latest/webapps/api/WEB-INF/classes/logback-common.xml

※3 ファイルとも共通です

```
<!-- ログファイル名パターン : 日時ローテーション -->  
<fileNamePattern>${logback.logdir}/${logFileName}.%d{yyyy-MM-dd}.log</fileNamePattern>  
<!-- 保存ログ数 -->  
<maxHistory>7</maxHistory>    ★この行を変更します
```

tomcat のログローテーション機能を無効にします。

```
$ sudo vi /opt/tomcat/latest/conf/server.xml
```

/opt/tomcat/latest/conf/server.xml

```
<!-- Access log processes all example.  
Documentation at: /docs/config/valve.html  
Note: The pattern used is equivalent to using pattern="common" -->  
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"  
prefix="localhost_access_log" suffix=".txt"  
rotatable="false"    ★この行を追加します  
pattern="%h %l %u %t &quot;%r&quot; %s %b" />
```

Tomcat を再起動します。

```
$ sudo systemctl restart tomcat
```

logrotate.d でログのローテーション設定をします。保存数は必要なログ保存期間に合わせて設定してください。以下では 7 日に変更する例を記載します。

```
$ sudo vi /etc/logrotate.d/tomcat
```

```
/etc/logrooatate.d/tomcat
```

```
/opt/tomcat/latest/logs/catalina.out
/opt/tomcat/latest/logs/localhost_access_log.txt
{
    copytruncate
    daily
    rotate 7
    #rotate 1
    compress
    missingok
    create 0644 tomcat
}
```

### 3.1.11. バックエンドプログラム設定

バックエンドプログラムの設定を行います。

バックエンドプログラムは baas console から設定します。baas console は現時点ではブラウザから `<http://[ServerIpAddress:8080]/console>` にアクセスすることで表示できます。サーバの IP アドレスが 192.168.11.132 の場合、<http://192.168.11.132:8080/console> にアクセスします。

3.3 章でフロントエンドプログラムをインストールした後は、そこで設定したリバースプロキシの設定に従ってアクセスすることになるのでご注意ください。



(画面例)サーバの IP アドレスが 192.168.11.132 の場合

初回は以下の初期システム管理者アカウントでログインしてください。初回ログインに成功するとパスワード設定画面が表示されますので、新しいパスワードを設定してください。パスワードは類推されにくいパスワードとし、厳重に管理、また定期的に変更するようにしてください。

メールアドレス (E-mail)	<a href="mailto:admin@example.com">admin@example.com</a>
初期パスワード (Password)	admin

バックエンドプログラムを動作させるにはライセンス認証が必要になります。  
ライセンス情報は NEC モバイルバックエンド基盤サーバ のライセンスファイルとして提供されます。  
"licenseKey="以降の文字列がラインセンスキーになります。

# (コメント)  
licenseKey=.....

baas console にシステム管理者アカウントでログインし、画面上部メニューの「管理」⇒「ライセンス」を選択します。

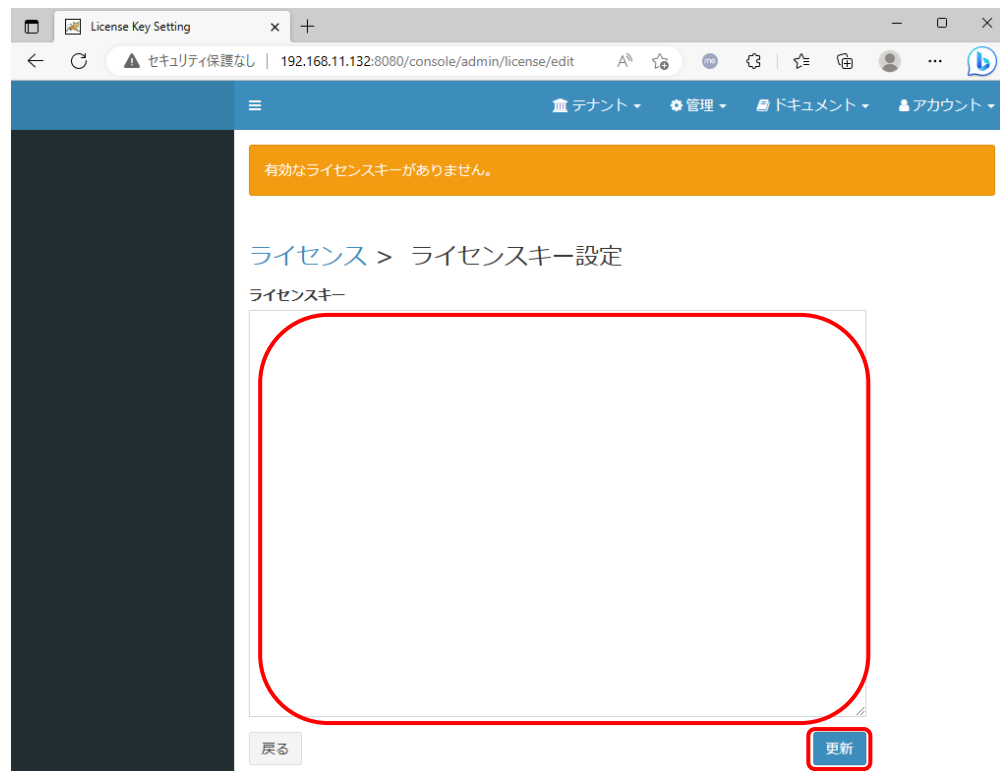




ライセンスキー設定を選択します。



ライセンスキーの入力欄にライセンスファイルのライセンスキー("licenseKey="以降の文字列)を入力して、更新を選択しライセンス認証を実行します。





baas console 上で 新規テナントの追加、アプリケーション追加、グループ追加、ユーザ登録を実施します。

新規テナントを追加します。

画面上部メニューの「管理」⇒「テナント管理」からテナント管理画面を開きます。テナント管理画面で「追加」を選択し、新規テナント追加画面を開きます。



「テナント名」は任意です。以降は 「01」 というテナント名で説明します。「テナント名」入力後、「保存」を選択します。

テナント管理 > テナント一覧 > 新規テナント追加

一般設定

テナント名 必須

01

説明

認証方式

通常認証

テナント有効/無効設定

有効

MongoDB個別接続設定

MongoDBサーバ

テナント固有のMongoDBサーバ(接続URI("mongodb://"は省略可)。無指定時はデフォルトサーバ。(例: mongodb://host1:27017,host2:27017,host3:27017)

認証ユーザ名

認証パスワード

戻る

保存

アプリケーションを追加します。

左メニューの「アプリケーション」を選択し、アプリケーション一覧画面を表示します。

「追加」ボタンを選択し、新規アプリケーション登録画面を開きます。

任意の「アプリ名」を入力し、「保存」ボタンを押下します。

## アプリケーション一覧 >

一般設定

アプリ名 必須

WirelessCommunicationVisualization

説明

☐ クライアントPush許可

アプリ有効/無効設定 有効

アプリケーションの追加が完了すると、アプリ詳細画面が表示されます。ここで表示されている「テナントID」「アプリケーションID」「アプリケーションキー」「マスターキー」は後で使用するので記録しておいてください。

01

アプリケーション

ユーザ

グループ

オブジェクトバケット

ファイルバケット

API Gateway

Cloud Functions

Event Subscription

API統計情報

管理者設定

テナント設定

テナント管理

追加しました

アプリケーション一覧 > アプリ詳細

アプリ名	WirelessCommunicationVisualization
テナントID	642bf3a23bad863a6b4cbd92
アプリケーションID	642bf8743bad863a6b4cbd96
アプリケーションキー	WqlfXutRvxiNhgrBPRjQqhftYCY8wRXNfEiwRxs
マスターキー	0qFZPtheSAfunf6nvse4sJtWpmlN7ihmUQRrWdT
説明	
クライアントPush	禁止
サーバーキー	未設定
APNs認証種別	Token
Team ID	
Key ID	
APNs 種別	Development
APNs topic	

戻る

グループ、オブジェクトバケット、インデックス、ファイルバケットを追加します。Baas 提供のシェルスクリプトと“etc\_wirelessvisualization.tar.gz” に格納されている “01.yaml” を使って設定します。“01.yaml” は 3.2.4 章の手順の中で、“ /etc/nec/wirelessvisualization/01” に展開されます。

そのため、ここで一度バックエンドプログラムの設定を中断し、3.2 章 解析プログラムのインストールの 3.2.4 章までを実施した後に、以降の手順を実施してください。

3.2.4 章までの設定が完了したらバックエンドプログラムの設定を再開します。

テナント名を「01」という名前から変更した場合は 01.yaml の tenant 名を修正してください。

```
$ wget https://github.com/nec-baas/baas-server/releases/download/v7.5.6/baas-server-7.5.6.tar.gz
$ sudo tar zxvf baas-server-7.5.6.tar.gz
$ cd baas-server-7.5.6/server
$ sudo vi /etc/nec/wirelessvisualization/01/01.yaml
```

/etc/nec/wirelessvisualization/01/01.yaml

```
tenant:
  name: "01"
```

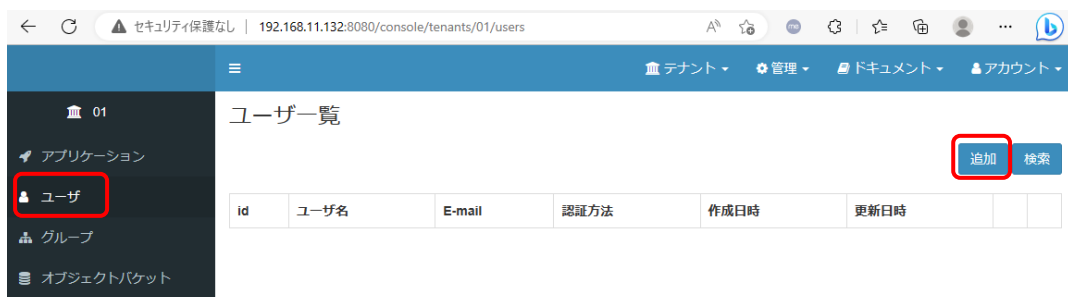
baas-admin.sh を実行します

```
$ sudo ./baas-admin.sh -c import -f /etc/nec/wirelessvisualization/01/01.yaml
```

左メニューの「ユーザ」を選択し、ユーザー一覧画面を表示します。「追加」を選択し、新規ユーザ追加画面を開きます。ユーザは定期的に棚卸を実施してください。

任意の「ユーザ名」、「E-mail」、「パスワード」を入力、役割に応じた「グループ」を選択し、「保存」選択してください。

管理ユーザアカウントと、無線センサアカウントは最低でも 1 つは必要になります。ここで作成した「E-mail」と「パスワード」はサーバへのログインに利用しますので、記録しておいてください。



「ユーザ名」は任意の文字列を入力してください。

「E-mail」は“@”と、“@”よりも後に“.”を含む100文字以下の文字列であれば、実際には利用できないメールアドレスでも問題ありません。

「パスワード」は8文字以上、100文字以下の1バイト文字列である必要があります。

「グループ」は“\_clientuser”、“\_necadmin”と“\_normaluser”の3種類があります。

“\_clientuser”はセンサプログラム用のグループで、センサがサーバ接続する時に使用します。登録した「E-mail」「パスワード」は4.2.4 センサプログラム インストールのconfig.yamlで設定します。

“\_necadmin”はWebUIの管理者グループで、UI画面にログインする時に使用します。登録した「E-mail」「パスワード」はWebUIのログイン画面で入力してください。管理権限でWebUIの設定操作が可能になります。

“\_normaluser”はWebUIの一般グループで、UI画面にログインする時に使用します。登録した「E-mail」「パスワード」はWebUIのログイン画面で入力してください。WebUI閲覧用ユーザであり、設定操作ができません。

## ユーザー一覧 > ユーザ追加

ユーザ名 必須

 sensor

E-mail 必須

 sensor@example.com

属性情報(オプション)[JSONフォーマット]

パスワード 必須

 .....

パスワード(確認) 必須

 .....

☐ パスワードの自動生成

☐ ユーザの追加をE-mail宛に通知

グループ

☒ \_clientuser ☐ \_necadmin ☐ \_normaluser

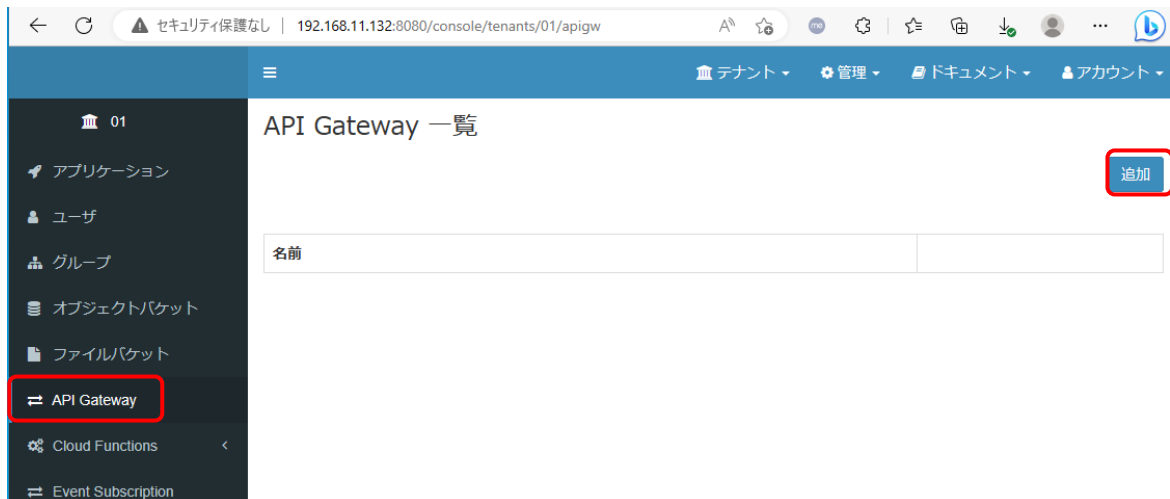
戻る

保存



API Gateway を追加します。

左メニューの「API Gateway」を選択し、API Gateway 一覧画面を表示し、「追加」を選択し追加画面を開きます。



「名前」に "wirelessVisualization" を設定し、「API 定義」に "etc\_wirelessvisualization.tar.gz" に格納されている "wirelessVisualization.yaml" の内容を貼り付けて「保存」を選択してください。

各 uri のポート番号はのちほど 3.2.5 解析プログラムの設定をする際に必要になります。テナントを複数作成する場合はテナント毎に異なるポート番号になるように設定してください。



## API Gateway > 追加

**名前** 必須

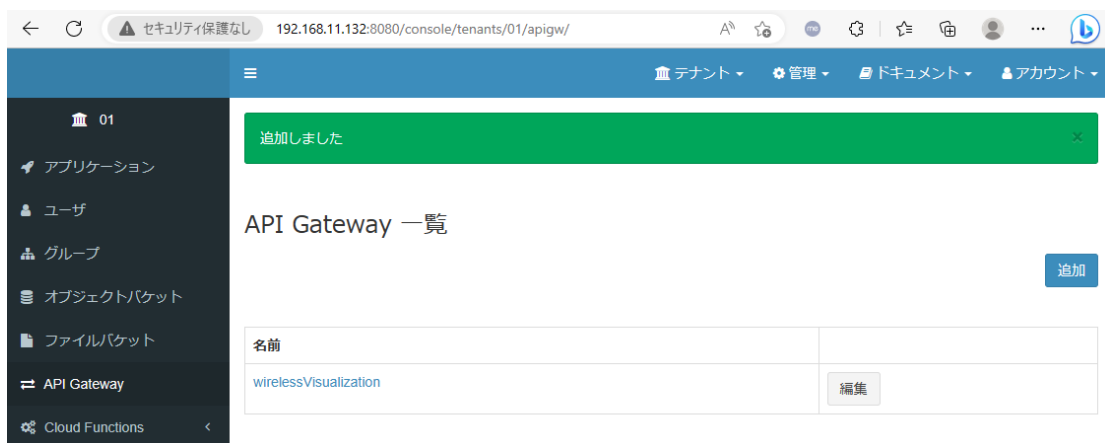
wirelessVisualization

**API 定義** 必須

```
1 ---
2 swagger: "2.0"
3 basePath: "/wirelessVisualization"
4 produces:
5   - "application/json"
6 x-acl:
7   - "g:_necadmin_"
8   - "g:_normaluser_"
9 paths:
10  /logs/analyze/data:
11    get:
12      x-proxy:
13        uri: http://localhost:3000/logs/analyze/data
14        method: GET
15  /logs/analyze/nrData:
16    get:
17      x-proxy:
18        uri: http://localhost:3000/logs/analyze/nrData
19        method: GET
20  /logs/analyze/srfData:
21    get:
22      x-proxy:
23        uri: http://localhost:3000/logs/analyze/srfData
24        method: GET
25  /logs/analyze/time:
26    get:
27      x-proxy:
28        uri: http://localhost:3000/logs/analyze/time
29        method: GET
30
```


戻る

保存



SSE Push サーバ公開 URI を設定します。無線センサからアクセスするための URI になるため、無線センサからアクセス可能なサーバの IP アドレスを設定してください。

画面上部メニューの「管理」⇒「システム設定」からシステム設定画面を開きます。



システム設定

一般設定

APIサーバ ベースURI	http://localhost:8080/
APIサーバ 内部向けベースURI	http://localhost:8080/api
SSE Push サーバ 公開URI	
システムキー	

システム設定画面で「編集」を選択し、編集画面を開きます。

## システム設定

### 一般設定

APIサーバ ベースURI	http://localhost:8080/api
APIサーバ 内部向けベースURI	http://localhost:8080/api
SSE Push サーバ 公開URI	
システムキー	

編集画面で「SSE Push サーバ公開 URI」に “https://<サーバ IP アドレス>/ssepush/events” を入力し、画面下部の「更新」を選択します。末尾の “/events” まで必要です。

## システム設定 > 編集

**一般設定**

**APIサーバ ベースURI**

**APIサーバ 内部向けベースURI**

**SSE Push サーバ 公開URI**  
  
SSE Push サーバの URI。末尾の “events” を含む。(例:  
https://push.example.com/ssepush/events)

**システムキー**

**ファイルストレージ設定**

**ファイルの最大サイズ(メガバイト)**

戻る

更新

システム設定画面の「SSE Push サーバ公開 URI」の欄に入力した URI が表示されます。



## システム設定

編集

### 一般設定

APIサーバベースURI	http://localhost:8080/api
APIサーバ 内部向けベースURI	http://localhost:8080/api
SSE Push サーバ 公開URI	https://192.168.11.132/ssepush/events
システムキー	

バックエンドプログラムの設定は以上です。

3.2.5 章の解析プログラムの設定から再開してください。

## 3.2. 解析プログラム インストール

ここでは解析プログラムをインストールする手順を説明します。

解析プログラムは pm2 を使ってプロセス管理をします。そのために必要なパッケージをインストールします。

### 3.2.1. Node.js インストール

Node.js をインストールします。apt リポジトリのバージョンは最新ではないため、パッケージ管理ツールである npm と Node.js の管理を行う n 使って Node.js 18 にアップデートします。

```
$ sudo apt install -y nodejs npm --no-install-recommends
$ sudo npm install -g n
$ sudo n 18
```

Node.js のバージョンは次のコマンドで確認できますが、アップデートした直後は古いバージョン情報が出力されます。その場合、一度コンソールをログアウトして再度確認すると、アップデート後のバージョンを確認することができます。

```
$ node -v
v18.15.0
```

### 3.2.2. pm2 インストール

npm を使って pm2 をインストールします。

```
$ sudo npm install -g pm2
```

### 3.2.3. 関連モジュール インストール

必要なモジュールをインストールします。

```
$ sudo apt install -y libatlas-base-dev gfortran libgtk-3-dev python3-pip
```

### 3.2.4. 解析プログラム インストール

解析プログラムはバックエンドプログラムの1テナントにつき、1つ起動するプログラムになります。

テナント毎に設定ファイルを保存するディレクトリが必要になります。ディレクトリ名は任意ですが、“etc”と“opt”の配下で共通の名称で作成してください。

解析プログラム用のディレクトリを作成します。共通のディレクトリ名を“01”として例示します。違う名称にする場合は適宜読み替えてください。

```
$ sudo mkdir -p /etc/nec/wirelessvisualization/01
$ sudo mkdir -p /opt/nec/wirelessvisualization/01
```

サーバに“etc\_wirelessvisualization.tar.gz”と“opt\_wirelessvisualization.tar.gz”、“wirelessvisualization-1.0.0.tgz”を格納してください。

設定ファイルと実行ファイルを解析プログラム用のディレクトリに展開します。

```
$ sudo tar zxvf etc_wirelessvisualization.tar.gz -C /etc/nec/wirelessvisualization/01/
$ sudo tar zxvf opt_wirelessvisualization.tar.gz -C /opt/nec/wirelessvisualization/01/
```

ここで中断していたバックエンドプログラムの設定に戻って、グループ、オブジェクトバケット、インデックス、ファイルバケットを追加から設定を再開してください。

3.1.11 章のバックエンドプログラム設定が全て完了したら、以下の手順から解析プログラムのインストールを再開してください。

解析プログラムをインストールします。

```
$ sudo npm install -g wirelessvisualization-1.0.0.tgz --omit=dev
```

必要なライブラリをインストールします。

```
$ sudo pip3 install numpy==1.20.3
$ sudo pip3 install Cython==0.29.33
$ sudo pip3 install -r /opt/nec/wirelessvisualization/01/requirements.txt
```

### 3.2.5. 解析プログラム 設定

baas ファイルを更新します。

```
$ sudo cp -f /opt/nec/wirelessvisualization/01/baas.js ¥
/usr/local/lib/node_modules/wirelessvisualization/node_modules/¥@nec-baas/jssdk/dist/
```

3.1.11 章のバックエンドプログラム設定の中で記録した「テナント ID」「アプリケーション ID」「マスターキー」と、「API Gateway」で設定したポート番号を

“/etc/nec/wirelessvisualization/01/config.json” 内の “baasInfo” の「tenant」「appld」「appKey」「port」に設定します。

バックエンドプログラムで作成されたキーには「アプリケーションキー」と「マスターキー」が存在します。この「appKey」には「マスターキー」を設定するので間違えないように注意してください。

```
$ sudo vi /etc/nec/wirelessvisualization/01/config.json
```

/etc/nec/wirelessvisualization/01/config.json

```
{
  "baasInfo": {
    "tenant": "xxxxxxxxxxxxxxxx",      ★「テナント ID」を設定
    "appld": "xxxxxxxxxxxxxxxx",      ★「アプリケーション ID」を設定
    "appKey": "xxxxxxxxxxxxxxxx",     ★「マスターキー」を設定
    "baseUri": "http://localhost:8080/api",
    "debugMode": "release",
    "timeout": 300,
    "port": 3000                      ★「API Gateway」で設定したポート番号を設定
  },
}
```

起動設定ファイル “/etc/nec/wirelessvisualization/01/srv.config.js” でテナント毎のディレクトリ名を編集します。「const dirName = 」にデフォルトで “01” が入力されていますので、“srv.config.js” の 1 つ上の階層のディレクトリ名に合わせて変更してください。

```
$ sudo vi /etc/nec/wirelessvisualization/01/srv.config.js
```

/etc/nec/wirelessvisualization/01/srv.config.js

```
// コンフィグファイルディレクトリ
const configDir = '/etc/nec/wirelessvisualization';
// ログ出力ディレクトリ
const logDir = '/var/log/wirelessvisualization';
// 実行ファイルディレクトリ
const execDir = '/opt/nec/wirelessvisualization';
// テナント毎のディレクトリ名
const dirName = '01';    ★ここを“srv.config.js” の 1 つ上の階層のディレクトリ名に変更
(以下省略)
```

サービスファイル “wlanlocation\_01.service” のファイル内のパスのテナント毎のディレクトリ名の部分を編集します。デフォルトではテナント毎のディレクトリ名を “01” として設定されていますので、設定した内容に合わせて変更してください。

```
$ sudo vi /opt/nec/wirelessvisualization/01/wlanlocation_01.service
```

/opt/nec/wirelessvisualization/01/wlanlocation\_01.service

```
# This file copy to /etc/systemd/system folder.
[Unit]
Description=wlanlocation for wireless visualization.

[Service]
User=root
WorkingDirectory=/opt/nec/wirelessvisualization/01/wlanlocation
ExecStart=/opt/nec/wirelessvisualization/01/wlanlocation.sh
Environment="LOGFILE=/var/log/wirelessvisualization/01/wlanlocation.log"
Environment="CONFIGFILE=/etc/nec/wirelessvisualization/01/config.json"
Restart=always
Type=forking

[Install]
WantedBy=multi-user.target
```

サービスファイルを systemd に登録します。登録する際にファイル名をテナント毎のディレクトリ名に合わせて変更してください。

```
$ sudo cp /opt/nec/wirelessvisualization/01/wlanlocation_01.service ¥
/etc/systemd/system/wlanlocation_01.service      ★テナント毎のディレクトリ名に変更
$ sudo systemctl enable wlanlocation_01.service  ★テナント毎のディレクトリ名に変更
```

### 3.2.6. 解析プログラム 起動

解析プログラムの起動と有効化をします。

```
$ sudo pm2 start /etc/nec/wirelessvisualization/01/srv.confii.js
$ sudo pm2 startup ubuntu
$ sudo pm2 save
$ sudo systemctl start wlanlocation_01.service      ★テナント毎のディレクトリ名に変更
```

解析プログラムのインストール手順は以上です。

### 3.3. フロントエンドプログラム インストール

ここではフロントエンドプログラムをインストールする手順を説明します。  
フロントエンドプログラムは nginx を利用します。そのために必要なパッケージをインストールします。

#### 3.3.1. nginx インストール

nginx の公開 gpg キーをインポートし、最新の安定化バージョンのリポジトリを登録後、インストールします。

```
$ curl -1sLf https://nginx.org/keys/nginx_signing.key | ¥
sudo gpg --dearmor | ¥
sudo tee /usr/share/keyrings/nginx-archive-keyring.gpg > /dev/null
$ echo "deb [signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg] ¥
http://nginx.org/packages/ubuntu `lsb_release -cs` nginx" | ¥
sudo tee /etc/apt/sources.list.d/nginx.list
$ sudo apt update
$ sudo apt install nginx
```

#### 3.3.2. リバースプロキシ 設定

リバースプロキシの設定を行います。設定ファイル “/etc/nginx/conf.d/default.conf” に以下の内容を追記してください。

```
$ sudo vi /etc/nginx/conf.d/default.conf
```

/etc/nginx/conf.d/default.conf

```
server {
    listen      443 ssl;
    server_name localhost;

    #charset koi8-r;

    location / {
        root    /usr/share/nginx/html;
#         root    /var/www/html;
        index   index.html index.htm;
    }

    location ~ ^/api/1/(.*)/api/srf/login {
        proxy_pass http://localhost:8080/api/1/$1/login;
    }

    #error_page 404                /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root    /usr/share/nginx/html;
    }

    # リバースプロキシ用の HTTP ヘッダ設定
    proxy_set_header Host $host:443;
    proxy_set_header X-Real-IP $remote_addr;
```



```

proxy_set_header X-Forwarded-Proto https;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Host $host:443;
proxy_set_header X-Forwarded-Server $host;

# login
#location ~ ^/api/1/(.*)/api/srf/login {
#    #return 301 http://localhost:8080/api/1/$1/login;
#    proxy_pass http://localhost:8080/api/1/$1/login;
#}

# installations
#location ~ ^/api/1/(.*)/api/srf/push/installations {
#    proxy_pass http://localhost:8080/api/1/$1/push/installations;
#}

# API サーバ
location /api/ {
    proxy_pass http://localhost:8080/api/;
    proxy_redirect http:// https://;
}

# コンソールサーバ
location /console/ {
    proxy_pass http://localhost:8080/console/;
    proxy_redirect default;
    proxy_http_version 1.1;
    proxy_set_header Connection "";
    proxy_connect_timeout 5m;
    proxy_read_timeout 5m;
    proxy_send_timeout 5m;
}

# SSE PUSH サーバ
location /ssepush/ {
    proxy_pass http://localhost:8080/ssepush/;
    proxy_redirect default;
    proxy_buffering off;
    proxy_cache off;
    proxy_set_header Connection "";
    proxy_connect_timeout 5m;
    proxy_read_timeout 5m;
    proxy_send_timeout 5m;
}

ssl_session_timeout 5m;

ssl_protocols TLSv1.2;
ssl_ciphers
ECDHE+RSAGCM:ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128
:DH+AES:!EXPORT:!DES:!3DES:!MD5:!DSS:!ADH:!AECDH;
ssl_prefer_server_ciphers on;

#ssl on;
ssl_certificate /etc/pki/tls/certs/server.crt;
ssl_certificate_key /etc/pki/tls/certs/server.key;

```

```
client_max_body_size 100m;
```

```
}
```

### 3.3.3. 自己証明書の作成

通信の暗号化のため自己発行の証明書を作成します。第三者の認証局による正規の証明書を利用する場合はこの手順は不要です。

認証ファイル “/etc/pki/tls/certs/ca.conf” に適切なデフォルト値を設定してください。署名リクエストの際にデフォルト値と違う内容を設定することができますが。ただし、「commonName」と「IPv4 localhost」には サーバの IP アドレスを記入する必要があります。

```
$ sudo mkdir -p /etc/pki/tls/certs  
$ sudo cp /etc/nec/wirelessvisualization/01/ca.conf /etc/pki/tls/certs/  
$ sudo vi /etc/pki/tls/certs/ca.conf
```

/etc/nec/wirelessvisualization/01/ca.conf

```
[ req ]
default_bits      = 2048
default_keyfile   = server-key.pem
distinguished_name = subject
req_extensions    = req_ext
x509_extensions  = x509_ext
string_mask       = utf8only

[ subject ]
countryName       = Country Name (2 letter code)
countryName_default = US

stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = NY

localityName       = Locality Name (eg, city)
localityName_default = New York

organizationName   = Organization Name (eg, company)
organizationName_default = Example, LLC

commonName         = Common Name (e.g. server FQDN or YOUR name)
commonName_default = 192.168.11.132 ★サーバの IP アドレスを設定

emailAddress       = Email Address
emailAddress_default = test@example.com

[ x509_ext ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer

basicConstraints     = CA:FALSE
keyUsage             = digitalSignature, keyEncipherment
subjectAltName       = @alternate_names
nsComment            = "OpenSSL Generated Certificate"

[ req_ext ]
subjectKeyIdentifier = hash

basicConstraints     = CA:TRUE
keyUsage             = digitalSignature, keyEncipherment
subjectAltName       = @alternate_names
nsComment            = "OpenSSL Generated Certificate"

[ alternate_names ]
# IPv4 localhost
IP.1      = 192.168.11.132 ★サーバの IP アドレスを設定
# IPv6 localhost
IP.2      = ::1
```

認証ファイルをもとに鍵を作成、署名リクエストファイルを作成します。デフォルト値のままでよければ、何も入力せずに Enter を押してください。作成した鍵と署名リクエストファイルを使って証明書を作成します。作成した証明書 “server.csr” はセンサにインポートします。また、ブラウザアクセスする PC にインポートする可能性があるため、保管してユーザー様に提供してください。

```
$ sudo openssl genrsa -out /etc/pki/tls/certs/server.key 2048
```

```
$ sudo openssl req -config /etc/pki/tls/certs/ca.conf -new -key /etc/pki/tls/certs/server.key  
-out /etc/pki/tls/certs/server.csr
```

*You are about to be asked to enter information that will be incorporated  
into your certificate request.*

*What you are about to enter is what is called a Distinguished Name or a DN.*

*There are quite a few fields but you can leave some blank*

*For some fields there will be a default value,*

*If you enter '.', the field will be left blank.*

-----

*Country Name (2 letter code) [US]:***JP**

*State or Province Name (full name) [NY]:***Kanagawa**

*Locality Name (eg, city) [New York]:***Kawasaki**

*Organization Name (eg, company) [Example, LLC]:***NEC**

*Common Name (e.g. server FQDN or YOUR name) [Example Company]:***192.168.11.132**

*Email Address [test@example.com]:*

```
$ sudo openssl x509 -req -days 36500 -in /etc/pki/tls/certs/server.csr -signkey  
/etc/pki/tls/certs/server.key -out /etc/pki/tls/certs/server.crt -extfile  
/etc/pki/tls/certs/ca.conf -extensions req_ext
```

作成した証明書に署名した内容が登録されていることを確認します。

```
$ openssl x509 -noout -text -in /etc/pki/tls/certs/server.crt
```

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

ba:ab:00:c5:ff:54:4e:05

Signature Algorithm: sha256WithRSAEncryption

Issuer: C=**JP**, ST=**Kanagawa**, L=**Kawasaki**, O=**NEC**,

CN=**192.168.11.132**/emailAddress=test@example.com

Validity

Not Before: Feb 22 09:39:21 2019 GMT

Not After : Feb 19 09:39:21 2029 GMT

Subject: C=**JP**, ST=**Kanagawa**, L=**Kawasaki**, O=**NEC**,

CN=**192.168.11.132**/emailAddress=test@example.com

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

Public-Key: (2048 bit)

(以下省略)

### 3.3.4. Web ページの展開

Web ページを公開用ディレクトリ “/usr/share/nginx/html/URL 名” に展開します。

公開用ディレクトリを作成します。ディレクトリ名は任意です。ここでは “WirelessVisualization” とし

ます。

```
$ sudo mkdir -p /usr/share/nginx/html/WirelessVisualization
```

サーバに“ui.zip”を格納してください。公開用ディレクトリに展開します。

```
$ sudo unzip ui.zip -d /tmp/  
$ sudo mv /tmp/ui/* /usr/share/nginx/html/WirelessVisualization/
```

### 3.3.5. バックエンドプログラムとの接続設定

公開用ディレクトリに展開した Web ページの設定ファイル“config.js”にバックエンドプログラムとの接続情報を設定します。「tenant」「appId」「appKey」にバックエンドプログラムで作成された「テナント ID」「アプリケーション ID」「アプリケーションキー」を記載します。

「baseUri」には“https://<サーバの IP アドレス>/api”を記載します。

```
$ sudo vi /usr/share/nginx/html/WirelessVisualization/config/config.js
```

/usr/share/nginx/html/WirelessVisualization/config/config.js

```
var NebulaConfig = {  
  "tenant": "xxxxxxxxxxxxxxxx",      ★「テナント ID」を設定  
  "appId": "xxxxxxxxxxxxxxxx",      ★「アプリケーション ID」を設定  
  "appKey": "xxxxxxxxxxxxxxxx",     ★「アプリケーションキー」を設定  
  "baseUri": "https://xxxxxxxxxxxxxxxx/api" ★https://<サーバ IP アドレス>/api を設定  
};
```

### 3.3.6. フロントエンドプログラム 起動

nginx を再起動することで Web ページが公開されます。nginx はインストール時に有効化されます。以降はバックエンドプログラムのコンソール画面にアクセスするときも https 通信になるのでご注意ください。

```
$ sudo systemctl restart nginx
```

フロントエンドプログラムのインストールは以上です。

## 3.4. プロセス監視 設定

バックエンドプログラムで利用している Tomcat, MongoDB, RabbitMQ や フロントエンドプログラムで利用している nxinx に何らかの異常が発生し停止するとシステムが利用できなくため、monit を利用してプロセスを監視し、監視対象のプロセスが停止した場合には再起動を行うようにします。

### 3.4.1. monit インストール

monit をインストールします。

```
$ sudo apt install monit
```

初期設定では、システム起動から 240 秒後に監視を開始し、120 秒周期でプロセス監視を行います。必要に応じてそれぞれの時間を調整してください。

```
$ sudo vi /etc/monit/monitrc
```

/etc/monit/monitrc

```
set daemon 120          # check services at 120 seconds intervals
# with start delay 240   # optional: delay the first check by 4-minutes
```

次に監視したいプロセス用の設定ファイルを追加します。

Tomcat 監視用設定ファイルを作成します。

```
$ sudo vi /etc/monit/conf.d/tomcat.conf
```

/etc/monit/conf.d/tomcat.conf

```
check process tomcat matching "tomcat"
start program = "/bin/systemctl start tomcat"
stop program = "/bin/systemctl stop tomcat"
```

MongoDB 監視用設定ファイルを作成します。

```
$ sudo vi /etc/monit/conf.d/mongod.conf
```

/etc/monit.d/mongod.monitrc

```
check process mongod matching "mongod"
start program = "/bin/systemctl start mongod"
stop program = "/bin/systemctl stop mongod"
```

RabbitMQ 監視用設定ファイルを作成します。

```
$ sudo vi /etc/monit/conf.d/rabbitmq-server.conf
```

/etc/monit.d/rabbitmq-server.monitrc

```
check process rabbitmq matching "rabbitmq"
start program = "/bin/systemctl start rabbitmq-server"
stop program = "/bin/systemctl stop rabbitmq-server"
```

nginx 監視用設定ファイルを作成します。

```
$ sudo vi /etc/monit/conf.d/nginx.conf
```

```
/etc/monit.d/nginx.pid
```

```
check process nginx with pidfile /var/run/nginx.pid
start program = "/bin/systemctl start nginx"
stop program = "/bin/systemctl stop nginx"
```

### 3.5. 時刻同期サーバ設定

時刻同期サーバの設定をします。センサは時刻同期完了を検知してからセンサプログラムを起動するため、センサからアクセス可能な時刻同期サーバがない場合はこのサーバを時刻同期サーバとして設定してください。

NTP デーモン “chrony” をインストールし、アクセスを許可するネットワークアドレスを登録します。また、サーバ自身が上位の NTP サーバから時刻同期できない場合にもセンサからの時刻同期を許可するために階層設定 stratum を 10 に設定します。

```
$ sudo apt install -y chrony
$ sudo vi /etc/chrony/chrony.conf
```

```
/etc/chrony/chrony.conf
```

```
allow 192.168.11.0/24
local stratum 10
```

### 3.6. ファイアウォール設定

ファイアウォール用のディレクトリを作成します。

```
$ sudo mkdir -p /opt/nec/serverFw
```

サーバに “opt\_serverFw.tar.gz” を格納してください。  
実行ファイルをファイアウォール用のディレクトリに展開します。

```
$ sudo tar zxvf opt_serverFw.tar.gz -C /opt/nec/serverFw/
```

#### 3.6.1. ファイアウォール設定の変更

ファイアウォール設定は TCP の HTTPS(443)、SSH(22) と UDP の NTP(123) のみ外部からのアクセスが可能になる設定になっています。サーバで許容する通信を追加したい場合は以下を参考に修正してください

```
$ sudo vi /opt/nec/serverFw/serverFw.sh
```

/opt/nec/serverFw/serverFw.sh

```
# ssh 許可
iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# https 許可
iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# ntp 許可
iptables -A INPUT -p udp --dport 123 -j ACCEPT
```

サービスファイルを systemd に登録します。

```
$ sudo cp /opt/nec/serverFw/serverFw.service ¥
/etc/systemd/system/serverFw.service
$ sudo systemctl enable serverFw.service
```

### 3.6.2. ファイアウォール起動

ファイアウォールの起動と有効化をします。

```
$ sudo systemctl start serverFw.service
```

ファイアウォールの設定手順は以上です。

## 3.7. 詳細設定

設定可能なファイルのパラメータ情報についてファイル単位で一覧を表示します。初期値のままで使用可能ですが、必要に応じて変更してください。

解析プログラムのコンフィグ

/etc/nec/wirelessvisualization/01/config.json

分類	パラメータ	デフォルト値	内容
mBaas テナント情報 (baasInfo)	tenant		mBaas のテナント ID
	appId		mBaas のアプリケーション ID
	appKey		mBaas のマスターキー
	baseUri		mBaas の api への URI
	debugMode	"release"	
	timeout	300	
	port	3000	mBaas の待ち受けポート
ローテート時間(timeInfo)	afterTime	4	測定データのローテート処理で 0 時から設定時間分ごとにデータをローテートします。[hour] 現在時刻から設定時間分のデータは可視化データのためローテートされないようにデータを保持します。 ローテートされたデータは csv としてサーバで保



			<p>管されます。初期設定だと、下記間隔で csv を作成します。</p> <p>00-04, 04-08, 08-16, 16-20, 20-00</p> <p>設定値は 1,2,3,4,6 で設定してください。それ以外の値の場合は 4 で動作します。</p>
	deleteSplit	1000	<p>ローテート処理で過去データをバケットから 1 度にデータを削除する件数。</p> <p>件数が大きすぎるとメモリ負荷が増えます (動作例：10000 件のデータを消す時、1000 件削除を 10 回行う)</p>
	periodTime	60	<p>ローテート処理でデータ計算を行う間隔を設定します。[sec]</p> <p>初期設定だと、60 秒毎にデータを計算します。</p>
	getCount	1000	<p>CSV ファイルをバックエンドプログラムに保存できる上限数。</p> <p>初期設定だと 1000 件まで保存され、超過した場合は古いデータから削除されます。</p>
CSV ファイル保持上限 (countInfo)	monitoringTime	60000	<p>イベントログ集計実行周期時間[msec]</p> <p>初期設定だと、60 秒ごとに実行</p>
監視イベントログ設定情報 (monitoringLogInfo)	alarmCount	10000	監視イベントログ保持数
	specificTime	60	<p>監視イベント集計単位[sec]</p> <p>※設定値は秒単位ですが、分刻みで設定してください</p>
	apStopType	alert	AP 停止検知アラート種別
	apStopTime	300	<p>AP 停止検知期間[sec]</p> <p>※設定値は秒単位ですが、分刻みで設定してください</p>
	staConnectionShortType	warning	帰属失敗検知アラート種別
	staConnectionShortTime	60	<p>帰属失敗検知期間[sec]</p> <p>※設定値は秒単位ですが、分刻みで設定してください</p>
	staConnectionShortCount	10	帰属失敗検知回数閾値
	staConnectionLongType	warning	ローミング多発検知アラート種別
	staConnectionLongTime	600	<p>ローミング多発検知期間[sec]</p> <p>※設定値は秒単位ですが、分刻みで設定してください</p>
	staConnectionLongCount	10	ローミング多発検知回数閾値
	staConnectionLongErrorCount	8	ローミング多発検知エラー回数閾値
	staRetryType	warning	通信効率劣化検知アラート種別
	staRetryTime	300	<p>通信効率劣化検知期間[sec]</p> <p>※設定値は秒単位ですが、分刻みで設定してください</p>
	staRetryCount	50	通信効率劣化検知閾値
	apCongestionType	warning	混雑検知アラート種別
	apCongestionTime	300	<p>混雑検知期間[sec]</p> <p>※設定値は秒単位ですが、分刻みで設定してください</p>
	apCongestionCount	50	混雑検知閾値

	InStopType	warning	無線センサ停止検知アラート種別
	InStopTime	60	無線センサ停止検知期間[sec] ※設定値は秒単位ですが、分刻みで設定してください
	apDetectType	information	未管理 AP 検知アラート種別
	getCount	1000	REST API の GET リクエストに対して時間指定がない場合のレスポンス数の上限
id 設定情報 (idInfo)	contactTime	15	濃厚接触者リストの濃厚接触期間 [min]
	radius	1	濃厚接触者リスの濃厚接触距離 [m] ※対象からの半径範囲
	maxRegistration	10000000	Mac 紐づけ情報、顔認証情報を mBaas に保存できる上限数を設定します。 初期設定だと 10,000,000 件まで保存され、超過した場合は古いデータから削除されます。
	timeSlice	60	濃厚接触継続判定周期時間 [sec] ※設定値は秒単位ですが、分刻みで設定してください

## 解析プログラムのログ出力設定

/etc/nec/wirelessvisualization/01/log\_srv.json

分類	パラメータ	デフォルト値	内容
出力処理の指定 (appenders)	type	dateFile	file: ファイル出力 datefile: 日付毎にファイル出力 console: コンソール出力
	filename	/var/log/wirelessvisualization/srv.log	ログ出力先/ファイル名
	pattern	yyyy-MM-dd	ファイルの出力パターン (例) srv.log-2023-04-01
	dayToKeep	14	ログファイルを保持する日数
	alwaysIncludePattern	true	レイアウトパターンを読み込む。
	layout	type:pattern pattern: "[%d %5.5p - %m%]"	%p ログレベル %d ISO8601 フォーマット %m ログデータ
出力内容のカテゴリ (category)	level	info	設定したレベルより上位の内容を出力 off: 出力しない fatal: 致命的 error: エラー warn: ワーニング info: お知らせ debug: デバッグ trace: トレース

## 端末位置推定機能の設定

/opt/nec/wirelessvisualization/01/wlanlocation/config.ini

分類	パラメータ	デフォルト値	内容
mbaas 情報取得設定 (mbaas)	getData_time	60	測定データ取得時間
	getCycle_time	10	最終解析時刻取得周期時間

## Web ページの描画設定

/usr/share/nginx/html/WirelessVisualization/dist/js/visualConfig.ini

分類	パラメータ	デフォルト値	内容
RSSI 描画	QUALITY	1/10	描画解像度。1 が高解像かつ高負荷処理
	RSSI_TH_MAX	-25	RSSI の最大値
	RSSI_TH_MIN	-90	RSSI の最小値
	RSSI_TH_VAL	2	等高線間隔閾値(db)
	RSSI_OPACITY	0.9	RSSI ヒートマップ不透明度
	RSSI_FILL_OPACITY	0.25	RSSI ヒートマップ塗り不透明度
	RSSI_LINE	0.1	等高線太さ(px:0 で線無し)
	RSSI_LINE_OPACITY	0.3	等高線不透明度
	RSSI_BASE_DATA	true	true:ベースデータ(全測定点の MAX)を使用する、false:使用しない
	KRIGING_MODEL	'exponential'	Kriging Model 'exponential':指数モデル 'gaussian':ガウスモデル 'spherical':円形モデル
	KRIGING_SIGMA2	0	Kriging Sigma2
	KRIGING_ALPHA	0.5	Kriging alpha
ICON 描画	ICON_SIZE	18	AP と STA アイコンの font-size(px)
占有率描画	OCCUPANCY_CIRCLE_SIZE	200	占有率の円直径(px)
	OCCUPANCY_CIRCLE_OPACITY	0.9	占有率の円不透明度
	OCCUPANCY	[30, 50]	パーセンテージの色閾値(青、黄色、赤)
	PARCENTAGE_COLOR_TH	[30, 50]	パーセンテージの色閾値(青、黄色、赤)
グラフ	RETRYRATE_TH_MIN	2	再送率の閾値(送信最小カウント数)
	LOWER_RETRY_COUNT	1	送信数の閾値
アニメーション	PLAY_DURATION	700	再生 Wait(msec)
	DEMO_MODE	True	RSSI データ、使用率、帰属数でデータを上書き。 自動 Play 移動。(時間毎のデータが上書きされ不定となる)
測定データファイル一覧画面更新時間	NODEFILELIST_UPDATE_TIME	15	測定データファイル一覧画面更新時間(sec)
SRF デバイス関連	SRFINFO_ACQUISITION_TIME	600	SRF データ取得間隔(sec)

## 3.8. バックエンドプログラム Tomcat 更新時の注意事項

tomcat をアップデータした場合はデプロイファイルのコピーとシンボリックリンクの再作成を実施してください。

```
$ sudo cp apache-tomcat-9.0.11/webapps/api.war apache-tomcat-9.0.31/webapps/
$ sudo cp apache-tomcat-9.0.11/webapps/console.war apache-tomcat-9.0.31/webapps/
$ unlink /opt/tomcat/latest
$ sudo ln -s /opt/apache-tomcat-9.0.31 /opt/tomcat/latest
$ sudo chown -R tomcat:tomcat /opt/apache-tomcat-9.0.31
```

## 4. センサ構築手順

センサは OSS パッケージを利用して無線 LAN パケットを収集し、内容を解析・統計処理して、サーバに通知する python スクリプトとして動作します。2.2 章のセンサ動作環境に記載しているとおり、HW に要求する条件は高くありませんが、複数台設置する必要があることからコストパフォーマンスの優れる Raspberry Pi 上に構築する手順を説明します。事前準備としては、Raspberry Pi にはあらかじめ Raspberry Pi OS Lite 64bit 版が動作可能であり、SSH アクセスの許可、適切なローカル設定が完了している状態であるものとして記載しています。構築は root 権限が必要なコマンドを“sudo”コマンドで実行でき、またインターネットに接続できる環境で実施してください。

Raspberry Pi 向けに多くのソフトウェアが準備されているため、他の HW 上に構築する場合には手順が異なる場合がありますが、該当する OSS の公式サイトなどを参考に適宜読み替えてください。

無線 LAN パケットの収集精度はセンサの受信感度によって品質が変わります。100%のパケット収集率にはなりませんが、可能な限り多方面からの受信感度が良い場所や、調査したい無線 LAN パケットを送受信する機器の近くに設置することを推奨します。

### 4.1. 時刻同期クライアント設定

センサの時刻が合っていない場合、サーバでの解析処理や画面表示が実態と異なる結果になります。Raspberry Pi はハードウェアクロックを持たないこともあり、後に設定するセンサプログラムのデーモンにも時刻同期設定が完了した後に起動するように設定しているため、時刻同期設定は必ず実施してください。

Raspberry Pi OS では初期状態で“debian.pool.ntp.org”に時刻同期する設定になっています。時刻同期に関する設定ファイル“/etc/systemd/timesyncd.conf”を編集して構築したサーバに時刻同期するように設定を変更します。

```
$ sudo vi /etc/systemd/timesyncd.conf
```

/etc/systemd/timesyncd.conf

```
[Time]
NTP=192.168.11.132      ★コメントアウト“#”を削除し、サーバ IP アドレスを設定します
#FallbackNTP=0.debian.pool.ntp.org 1.debian.pool.ntp.org 2.debian.pool.ntp.org 3.debian.pool.ntp.org
#RootDistanceMaxSec=5
#PollIntervalMinSec=32
PollIntervalMaxSec=60  ★コメントアウト“#”を削除し、60 を設定します
```

次回起動時からはサーバに時刻同期するようになりますが、即時に設定を反映したい場合は、時刻同期のデーモンを再起動してください。

```
$ sudo systemctl restart systemd-timesyncd.service
```

時刻同期の完了を検知するためのデーモン“systemd-time-wait-sync.service”を有効化します。これにより時刻同期の完了を待ってからセンサプログラムを起動することを実現します。

```
$ sudo systemctl enable systemd-time-wait-sync.service
```

## 4.2. センサプログラム インストール

センサプログラムのインストール手順を説明します。

### 4.2.1. OSS パッケージ インストール

センサの構築や、動作に使用する OSS パッケージをインストールします。  
まず環境を最新にします。

```
$ sudo apt update  
$ sudo apt -y upgrade
```

python をインストールします。

```
$ sudo apt install -y python3.9 python3-pip
```

インストールしたバージョンに対してシンボリックリンクが作成されていることを確認します。

```
$ sudo ls -laF /usr/bin/python*  
lrwxrwxrwx 1 root      7 Mar  3  2021 /usr/bin/python -> python3*  
lrwxrwxrwx 1 root      9 Apr  5  2021 /usr/bin/python3 -> python3.9*  
-rwxr-xr-x 1 root 5280744 Mar  1  2021 /usr/bin/python3.9*  
lrwxrwxrwx 1 root      34 Mar  1  2021 /usr/bin/python3.9-config -> aarch64-linux-gnu-python3.9-  
config*  
lrwxrwxrwx 1 root      16 Apr  5  2021 /usr/bin/python3-config -> python3.9-config*
```

既存のシンボリックリンクがインストールしたバージョン 3.9 にリンクされていない場合や、シンボリックリンクが存在しない場合はシンボリックリンクを作成します。

既存のシンボリックリンクで適切状態な場合や、“python3.9-config”が存在しない場合は省略してください。

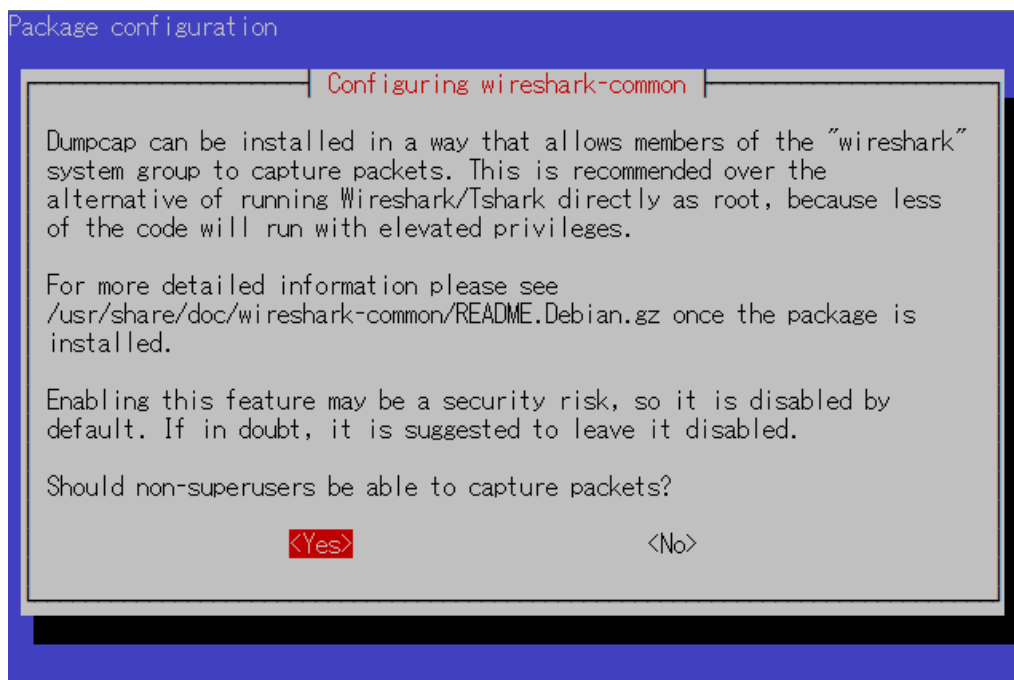
```
$ sudo ln -s /usr/bin/python3 /usr/bin/python  
$ sudo ln -s /usr/bin/python3.9 /usr/bin/python3  
$ sudo ln -s /usr/bin/python3-config /usr/bin/python-config  
$ sudo ln -s /usr/bin/python3.9-config /usr/bin/python3-config
```

パケットの取得や解析を行う “tshark” をインストールします。

```
$ sudo apt install -y tshark
```

インストール中に wireshark-common の設定として、スーパーユーザ以外のパケット取得を許可するか確

認がされるため、“Yes” を選択し、スーパーユーザ以外のパケット取得を許可してください。



もし指定を間違った場合は、インストール完了後に以下のコマンドで再度設定画面を表示し “Yes” を選択してください。

```
$ sudo dpkg-reconfigure wireshark-common
```

サーバとの連携動作をするのに使用する “nodejs” をインストールします。apt リポジトリのバージョンは最新ではないため、パッケージ管理ツールである npm を、Node.js の管理を行う n 使って Node.js 18 にアップデートします。

```
$ sudo apt install -y nodejs npm --no-install-recommends
$ sudo npm install -g n
$ sudo n 18
```

#### 4.2.2. 無線 LAN ドライバ インストール

無線 LAN パケットを収集するには、モニターモード動作に対応したチップセット・ドライバである必要があります。

参考として 推奨無線 LAN ドングルの NEC WL900U が搭載しているチップセット RTL8812AU と、TP-Link Archer T9HU が搭載しているチップセット RTL8814AU に対応した、aircrack-ng が公開する無線 LAN ドライバのインストール方法を説明します。詳細は [aircrack-ng サイト](#) の最新バージョンの “README.md” を参照してください。

```
$ sudo apt install -y raspberrypi-kernel-headers dkms git
$ git clone https://github.com/aircrack-ng/rtl8812au.git
$ cd rtl8812au
$ sed -i 's/CONFIG_PLATFORM_I386_PC = y/CONFIG_PLATFORM_I386_PC = n/g' Makefile
$ sed -i 's/CONFIG_PLATFORM_ARM64_RPI = n/CONFIG_PLATFORM_ARM64_RPI = y/g' Makefile
$ export ARCH=arm64
$ sed -i 's/^MAKE="/MAKE="ARCH=arm64¥ /' dkms.conf
$ sudo make dkms_install
```

また、参考として 推奨無線 LAN ドングルの TP-Link Archer TX20UH、BUFFALO WI-U3-1200AX2、ELECOM WDC-X1201DU3、ASUS USB-AX56NEC WL900U が搭載しているチップセット RTL8852AU に対応した、lwfinger が公開する無線 LAN ドライバのインストール方法を説明します。詳細は [lwfinger サイト](#) の最新バージョンの “README.md” を参照してください。

```
$ sudo apt install -y raspberrypi-kernel-headers build-essential git
$ git clone https://github.com/lwfinger/rtl8852au.git
$ cd rtl8852au
$ make
$ sudo make install
```

また、参考として 推奨無線 LAN ドングルの NETGEAR A8000 が搭載しているチップセット mt7921u に対応した無線 LAN ドライバを読み込む方法を説明します。mt7921u ドライバは Linux 5.18 以降のカーネルに組み込まれています。

2024 年 3 月現在では、Raspberry Pi OS Bullseye の Linux カーネルには NETGEAR A8000 の Device ID が組み込まれていないため、プラグアンドプレイを実現するための Device ID 設定と合わせて、Linux カーネルに組み込まれている mt7921u の読み込み設定を記載します。

```
$ sudo vi /etc/udev/rules.d/90-usb-08469060-mt7921u.rules
```

/etc/udev/rules.d/90-usb-08469060-mt7921u.rules

```
ACTION=="add", ¥
    SUBSYSTEM=="usb", ¥
    ENV{ID_VENDOR_ID}=="0846", ¥
    ENV{ID_MODEL_ID}=="9060", ¥
    RUN+="/usr/sbin/modprobe mt7921u", ¥
    RUN+="/bin/sh -c 'echo 0846 9060 > /sys/bus/usb/drivers/mt7921u/new_id'"
```

### 4.2.3. ネットワーク設定

センサはサーバと通信可能である必要があります。通信手段として有線 LAN や Raspberry Pi 内蔵の無線 LAN を使用できます。IP アドレスは DHCP による動的設定で問題ありませんが、無線 LAN パケットを収集する無線 LAN インターフェースを管理対象から除外する必要があるため、ネットワーク設定ファイル “/etc/dhcpd.conf” に設定を追加します。固定の IP アドレスを設定する場合などは、同ファイル内に予め記載されている設定例を参考にしてください。

```
$ sudo vi /etc/dhcpd.conf
```

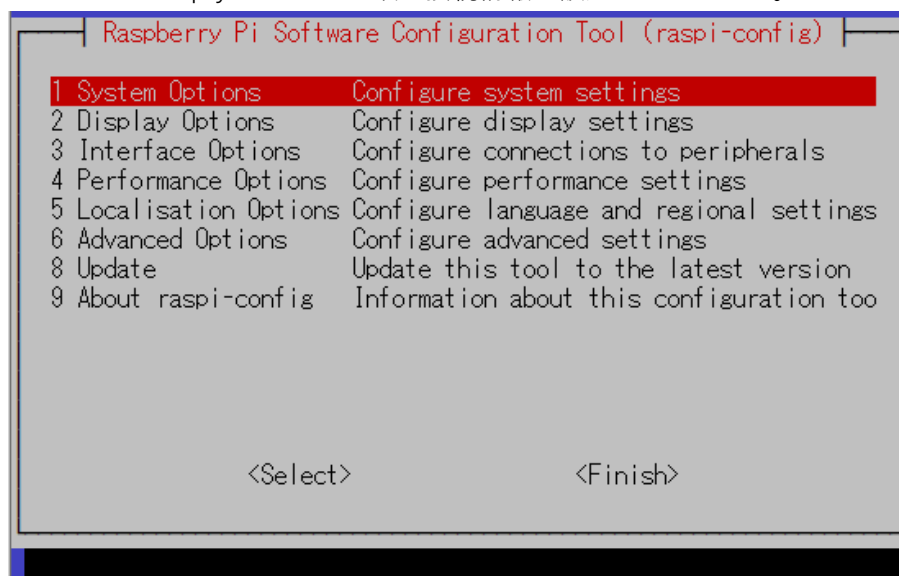
/etc/dhcpd.conf

```
interface wlan1  
nohook wpa_supplicant
```

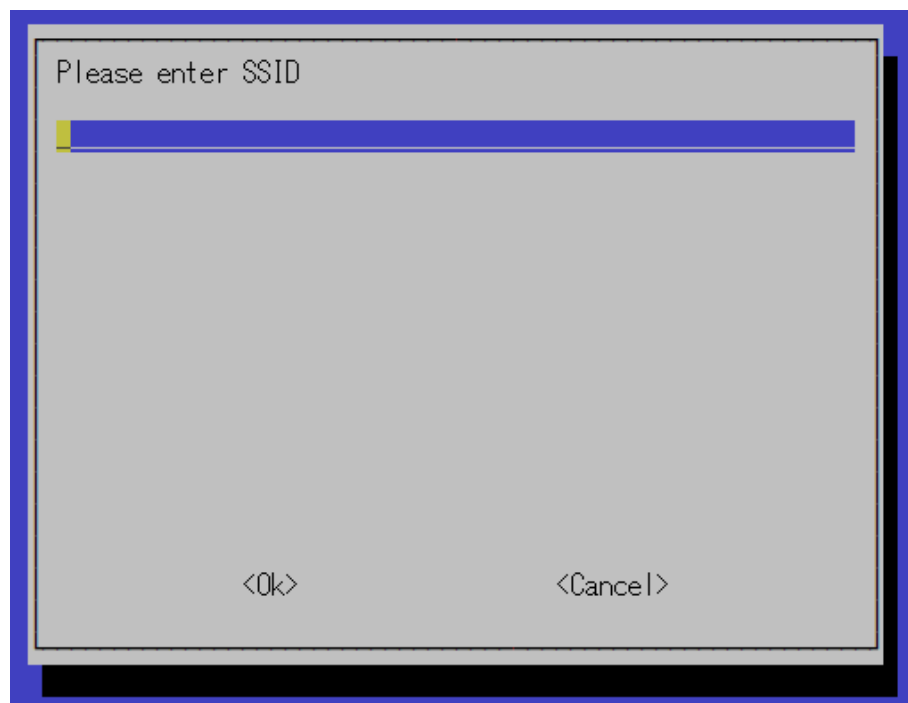
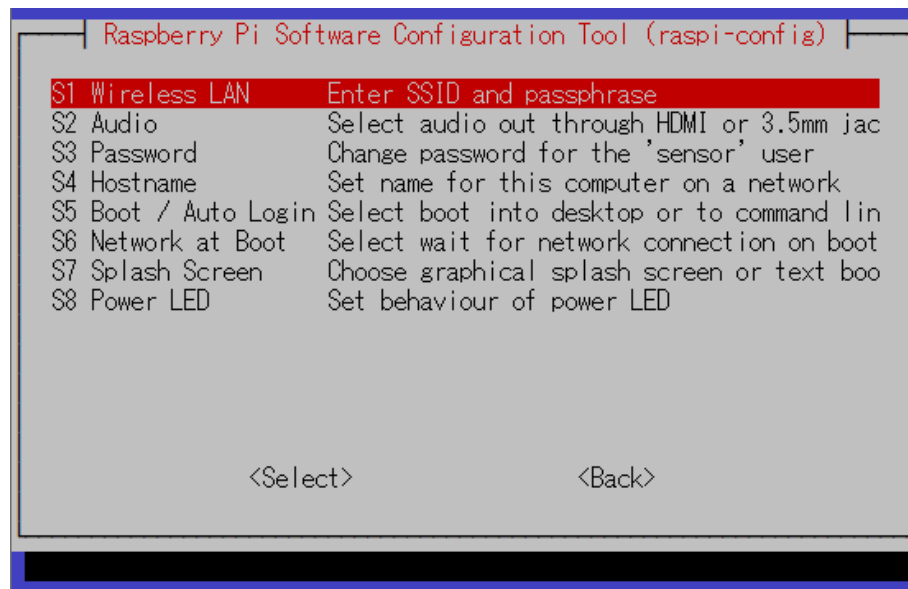
通信手段として Raspberry Pi 内蔵の無線 LAN を使用する場合は、“raspi-config” を使って無線接続設定をします。

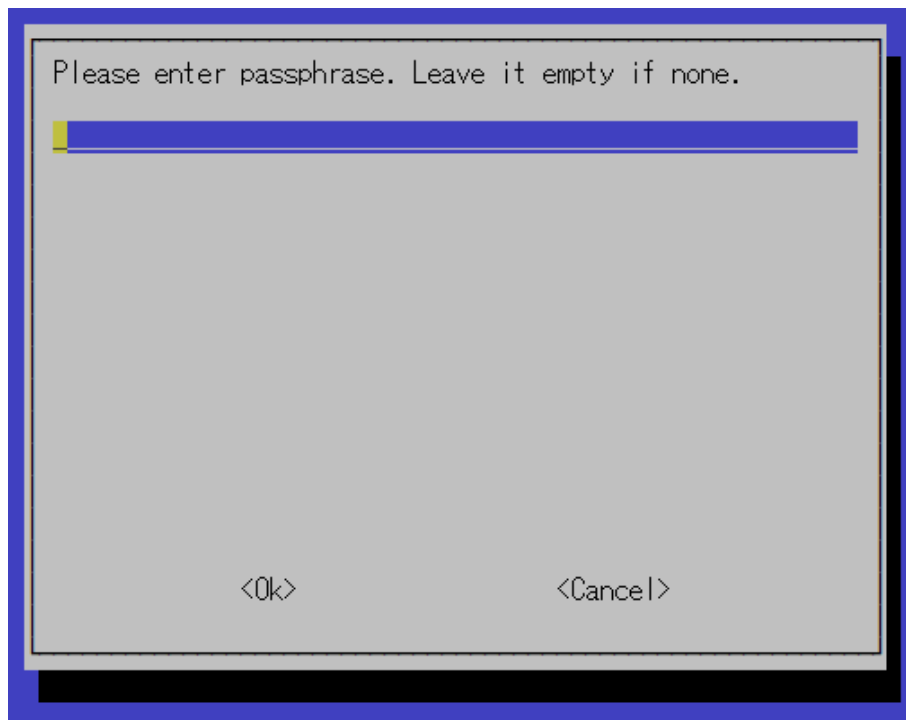
```
$ sudo raspi-config
```

設定画面が表示されるので “1 System Options”、“S1 Wireless LAN” を選択し、“Please enter SSID”、“Please enter passphrase. Leave it empty if none” の順に接続情報を設定してください。









raspi-config で設定した接続情報は “/etc/wpa\_supplicant/wpa\_supplicant.conf” に登録され、無線 LAN クライアントのデーモン “wpa\_supplicant” から参照されます。接続情報を変更する場合は、“/etc/wpa\_supplicant/wpa\_supplicant.conf” を直接修正してください。

#### 4.2.4. センサプログラム インストール

インストールする環境に “client.tar.gz” を格納してください。展開先のディレクトリを作成しセンサプログラムを展開します。

```
$ sudo mkdir /opt/nec  
$ sudo tar zxvf client.tar.gz -C /opt/nec/
```

サーバと接続するための設定をします。サーバのバックエンドプログラムで記録した情報を “tenantId”, “appId”, “appKey”, “baseUrl”, “email”, “password” を修正してください。

```
$ sudo cp /opt/nec/client/etc/config.yaml.sample /opt/nec/client/etc/config.yaml  
$ sudo vi /opt/nec/client/etc/config.yaml
```

/opt/nec/client/etc/config.yaml

```
tenantInfo:
  # Tenant ID
  # (str): require
  tenantId: "sampletenantid1234567890"
  # Application ID
  # (str): require
  appId: "sampleappid1234567890"
  # Application Key
  # (str): require
  appKey: "sampleAppKey1234567890"
  # API Server Base URI
  # (str): require
  baseUrl: https://192.168.11.132/api
```

(中略)

```
userInfo:
  # (str): require
  email: "sensor@example.com"
  # (str): require
  password: "sensor1234"
```

LED のシンボル名を “led0” として登録していますが、シンボル名が変更されている場合があります。以下のコマンドを実行して “led0” が存在しない場合には、他の LED のシンボル名を設定してください。

```
$ ls -la /sys/class/leds/
lrwxrwxrwx  1 root 0 Apr 10 17:35 ACT -> ../../devices/platform/leds/leds/ACT
xrxrwx  1 root 0 Apr 10 17:35 default-on -> ../../devices/virtual/leds/default-on
lrwxrwxrwx  1 root 0 Apr 10 17:35 mmc0 -> ../../devices/virtual/leds/mmc0
lrwxrwxrwx  1 root 0 Apr 10 17:35 PWR -> ../../devices/platform/leds/leds/PWR
$ sudo vi /opt/nec/client/etc/config.yaml
```

/opt/nec/client/etc/config.yaml

```
aliveMonitor:
  # LED Path
  # RED LED
  # /sys/class/leds/led1/trigger
  # Green LED
  # /sys/class/leds/led0/trigger
  # (str): require
  ledPath: "/sys/class/leds/ACT/trigger"
```

センサにサーバの自己証明書をインポートします。サーバで作成した自己証明書“server.crt”をセンサの“/usr/local/share/ca-certificates”に格納し、証明書情報を更新します。

```
$ sudo mv server.crt /usr/local/share/ca-certificates/
$ sudo update-ca-certificates
```

Python の仮想環境を生成します。

```
$ cd /opt/nec/client
$ sudo pip3 install pipenv
$ sudo pipenv install
```

node モジュールをインストールします。

```
$ cd /opt/nec/client/lib/ssePushReceiver
$ sudo npm install --omit=dev
$ cd ~/
```

センサソフトウェアを systemd に登録します。

```
$ sudo cp /opt/nec/client/bin/ClientAgent.service /etc/systemd/system/
$ sudo systemctl enable ClientAgent.service
$ sudo cp /opt/nec/client/bin/SsePushReceiver.service /etc/systemd/system/
$ sudo systemctl enable SsePushReceiver.service
```

再起動後、センサプログラムが自動的に起動します。

### 4.3. センサプログラム 起動・停止方法

センサプログラムは装置起動後、時刻同期を確認した後に、自動的に起動しますが、明示的に起動・停止を制御する場合は以下のコマンドで制御することが可能です。

センサプログラムの起動

```
$ sudo systemctl start ClientAgent.service
$ sudo systemctl start SsePushReceiver.service
```

センサプログラムの終了

```
$ sudo systemctl stop ClientAgent.service
$ sudo systemctl stop SsePushReceiver.service
```

センサプログラムはサーバとの接続可否を監視しており、一定回数接続が確認できない場合は再起動による接続復旧を試みます。再起動を繰り返す場合は、サーバとの接続ができていない可能性があるため、一度センサプログラムのデーモンを停止し、ネットワーク設定、センサプログラムの設定“/opt/nec/client/etc/config.yaml”を確認してください。

## 4.4. センサ 高可用化

Raspberry Pi のフリーズを検知して再起動するように watchdog を設定します。オプションのハードウェアインターフェースとして watchdog を有効化します。

```
$ sudo vi /boot/config.txt
```

/boot/config.txt

```
# Uncomment some or all of these to enable the optional hardware interfaces
#dtparam=i2c_arm=on
#dtparam=i2s=on
#dtparam=spi=on
dtparam=watchdog=on      ★この行を追加
```

watchdog モジュールの設定ファイル“/etc/modprobe.d/bcm2835-wdt.conf”を作成します。ここに設定する“heartbeat”間隔(秒)の間に、この後設定するシステムからの生存信号を受信できなければ再起動します。

```
$ sudo vi /etc/modprobe.d/bcm2835-wdt.conf
```

/etc/modprobe.d/bcm2835-wdt.conf

```
options bcm2835_wdt heartbeat=60 nowayout=0
```

システムから watchdog モジュールに生存信号を通信する間隔を設定するファイル“/etc/systemd/system.conf.d/systemd-wdt.conf”を作成します。先ほど watchdog モジュールに設定した“heartbeat”値よりも短い時間を設定してください。

```
$ sudo mkdir /etc/systemd/system.conf.d
$ sudo vi /etc/systemd/system.conf.d/systemd-wdt.conf
```

/etc/systemd/system.conf.d/systemd-wdt.conf

```
RuntimeWatchdogSec=30
```

センサプログラムを監視し、停止している場合に装置を再起動するよう設定します。

monit は “/etc/monit/monitrc” の中で 120 秒間隔でチェックをするように設定されているため、3 サイクル 6 分間センサプログラムの動作を確認できてきなければ再起動します。

```
$ sudo apt install monit
$ sudo vi /etc/monit/conf.d/ClientAgent.conf
```

/etc/monit/conf.d/ClientAgent.conf

```
check process ClientAgent matching "ClientAgent"
if does not exist for 3 cycles then exec "/bin/systemctl reboot"
```

```
$ sudo vi /etc/monit/conf.d/SsePushReceiver.conf
```

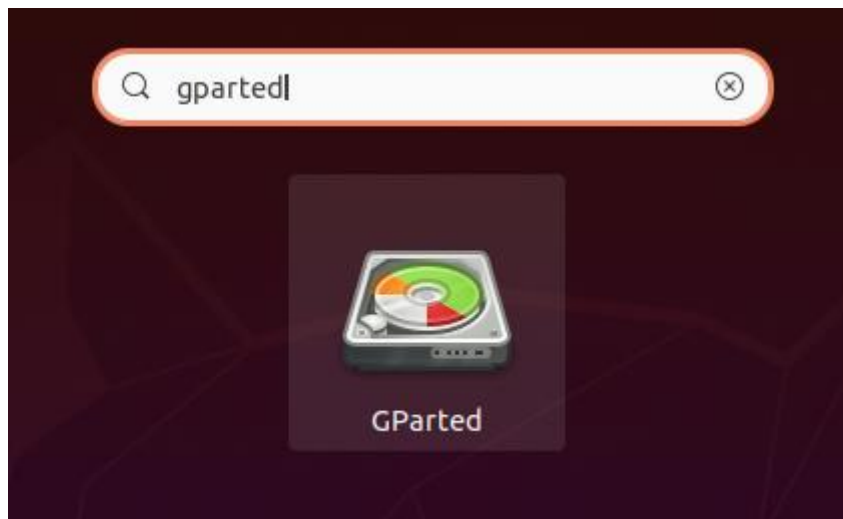
/etc/monit/conf.d/SsePushReceiver.conf

```
check process ClientAgent matching "SsePushReceiver"
if does not exist for 3 cycles then exec "/bin/systemctl reboot"
```

センサプログラムのログデータを保存するディレクトリ以外を overlayfs によって読み取り専用にすることで SD カードの寿命を延ばし、突然の電源断による SD カード破損を防止します。

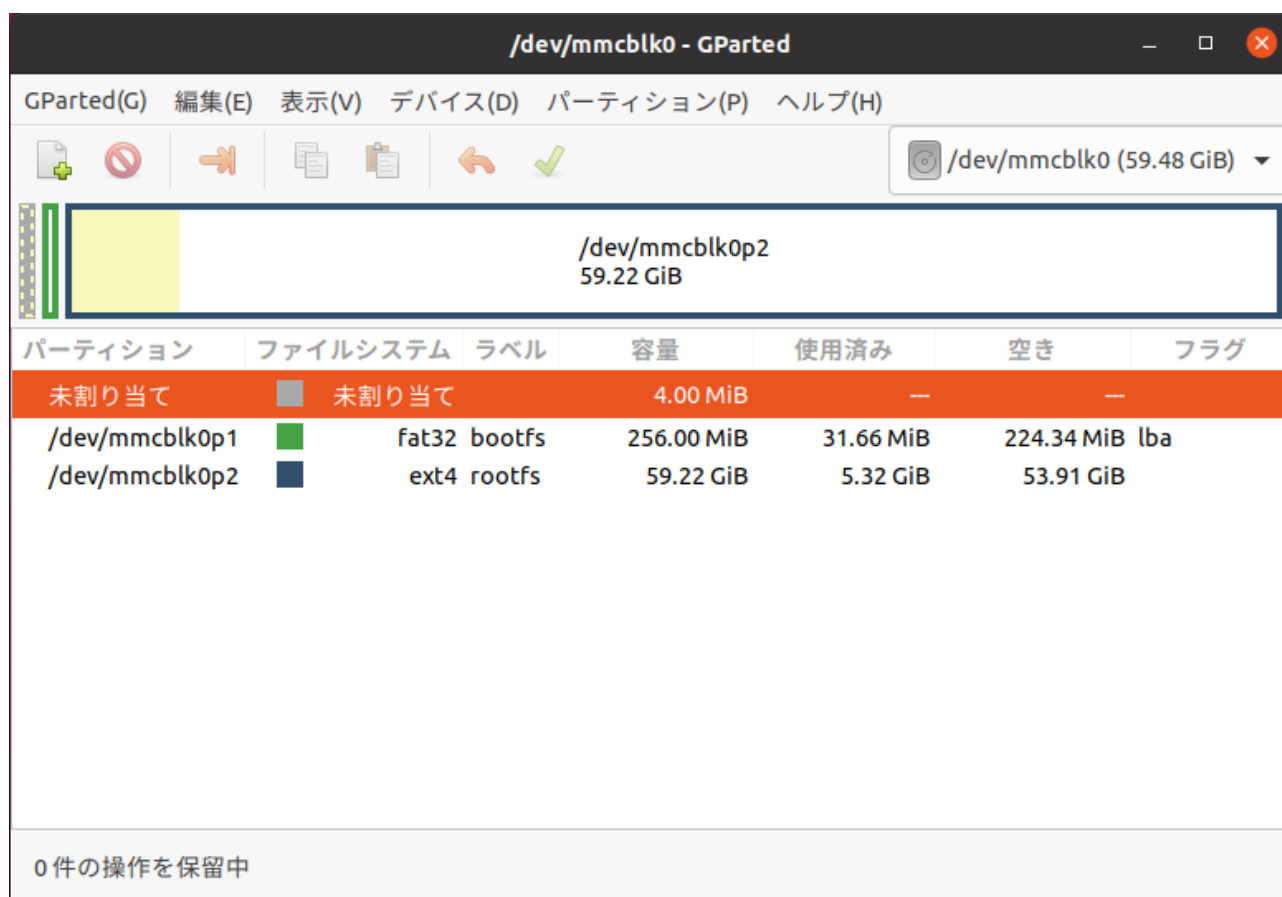
まずセンサプログラムのログデータを保存するディレクトリをマウントするための新しいパーティションを作成します。パーティションの作成方法は複数ありますが、ここでは Raspberry Pi とは別の Linux PC に Raspberry Pi の SD カードを読み込ませ GParted を使ってパーティションを作成します。

まず Raspberry Pi とは別の Linux PC を起動し、ログインします。停止した Raspberry Pi から SD カードを取り出し Linux PC に差し込んでください。SD カードを差し込んだ状態で、GParted を起動します。”アクティビティ”で”GParted”と入力するとアイコンで GParted が表示されるため、選択して起動します。



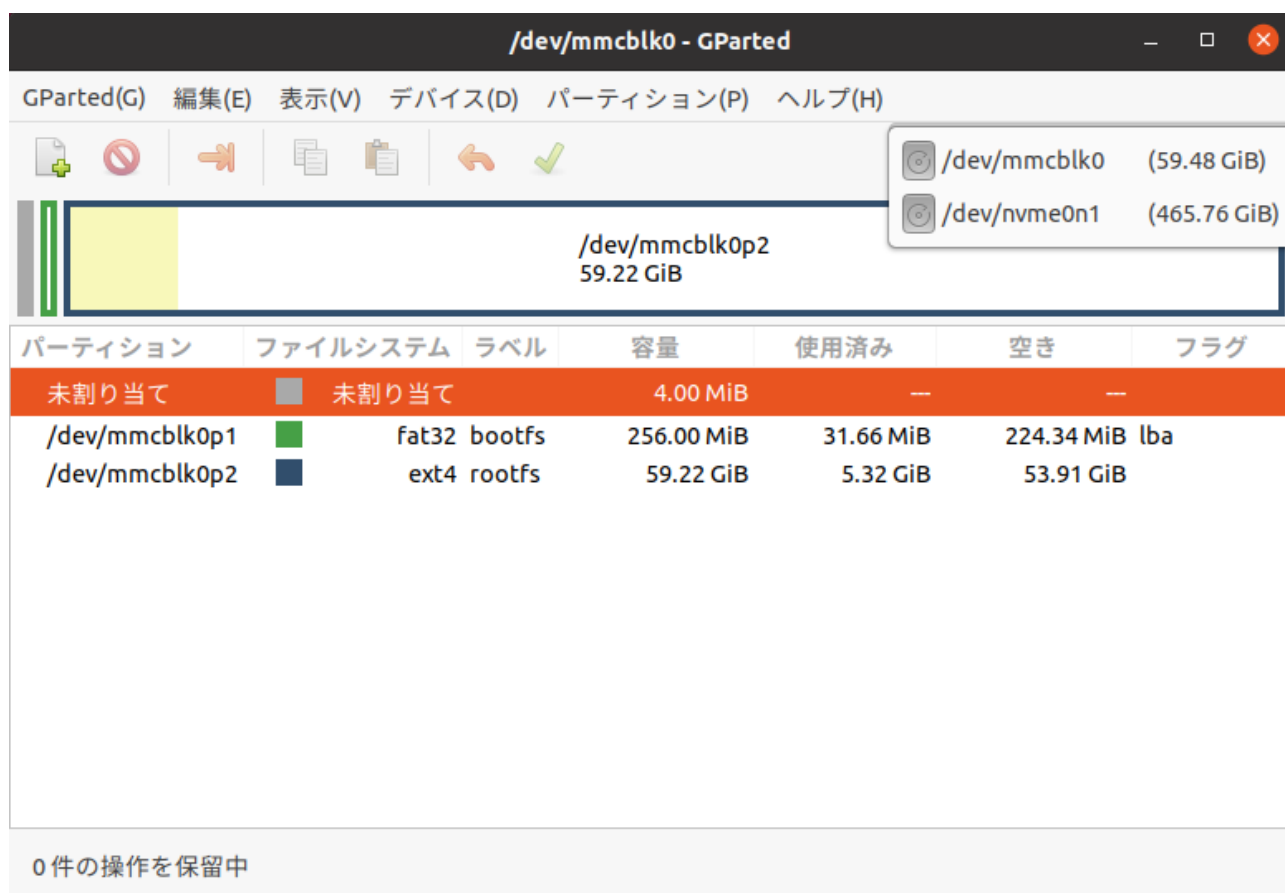
起動すると root 権限の認証画面が表示されるので Linux PC の root パスワードを入力して”認証”を選択してください。GParted が起動します。



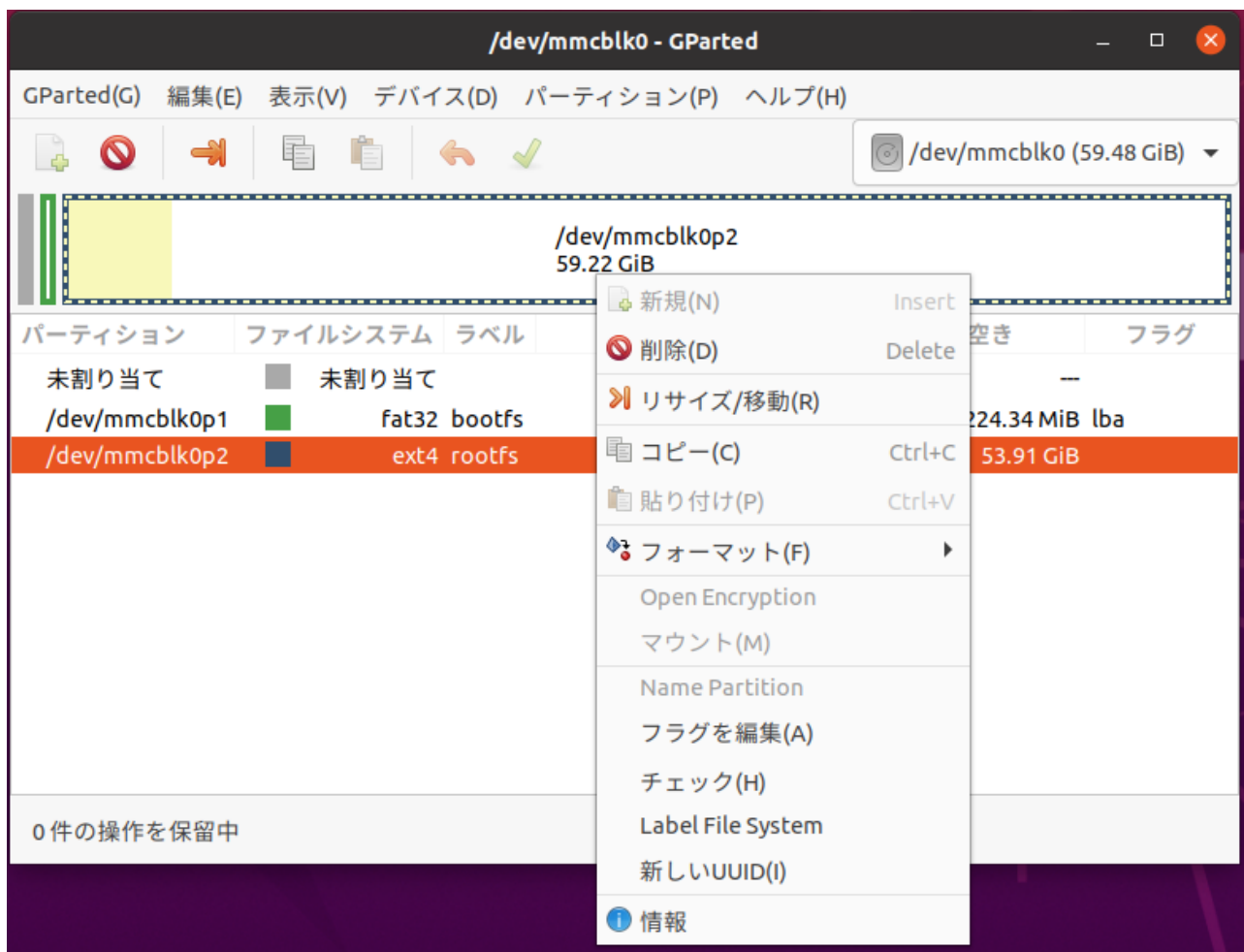


容量などから Raspberry Pi の SD カードが選択されていることを確認してください。"mmcblk0"や"sda"などのドライブ名で認識されることが多いです。容量やドライブ名からでは判断できない場合は、一度 GParted を終了し、SD カードを取り出した状態で再度 GParted を起動してください。SD カードを取り出した状態で表示されるドライブ名が SD カード以外のドライブですので、メモしておきます。その後あらためて SD カードを Linux PC に接続し GParted を起動したときに、SD カードを差し込んだ状態の時のみ表示されるドライブ名が Raspberry Pi の SD カードです。

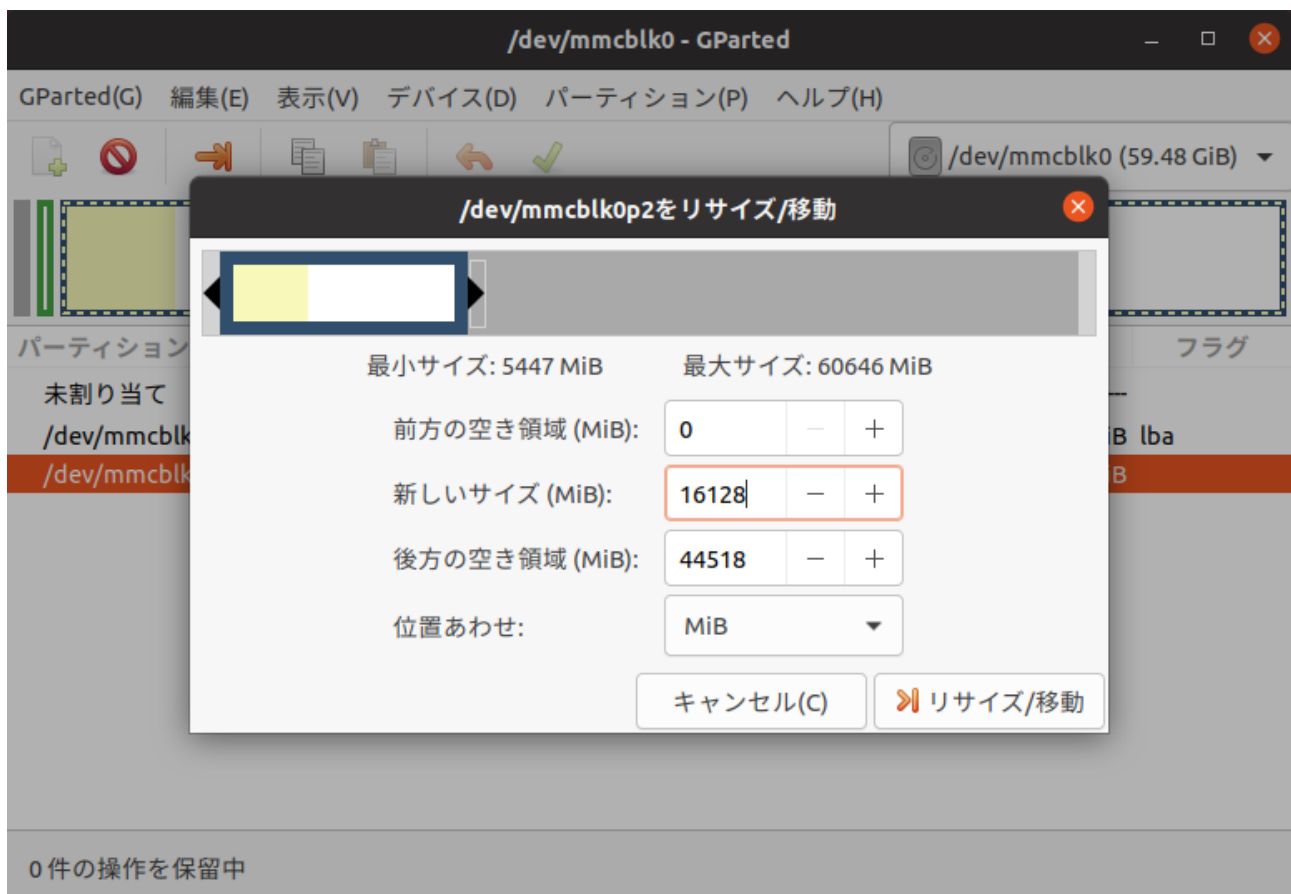




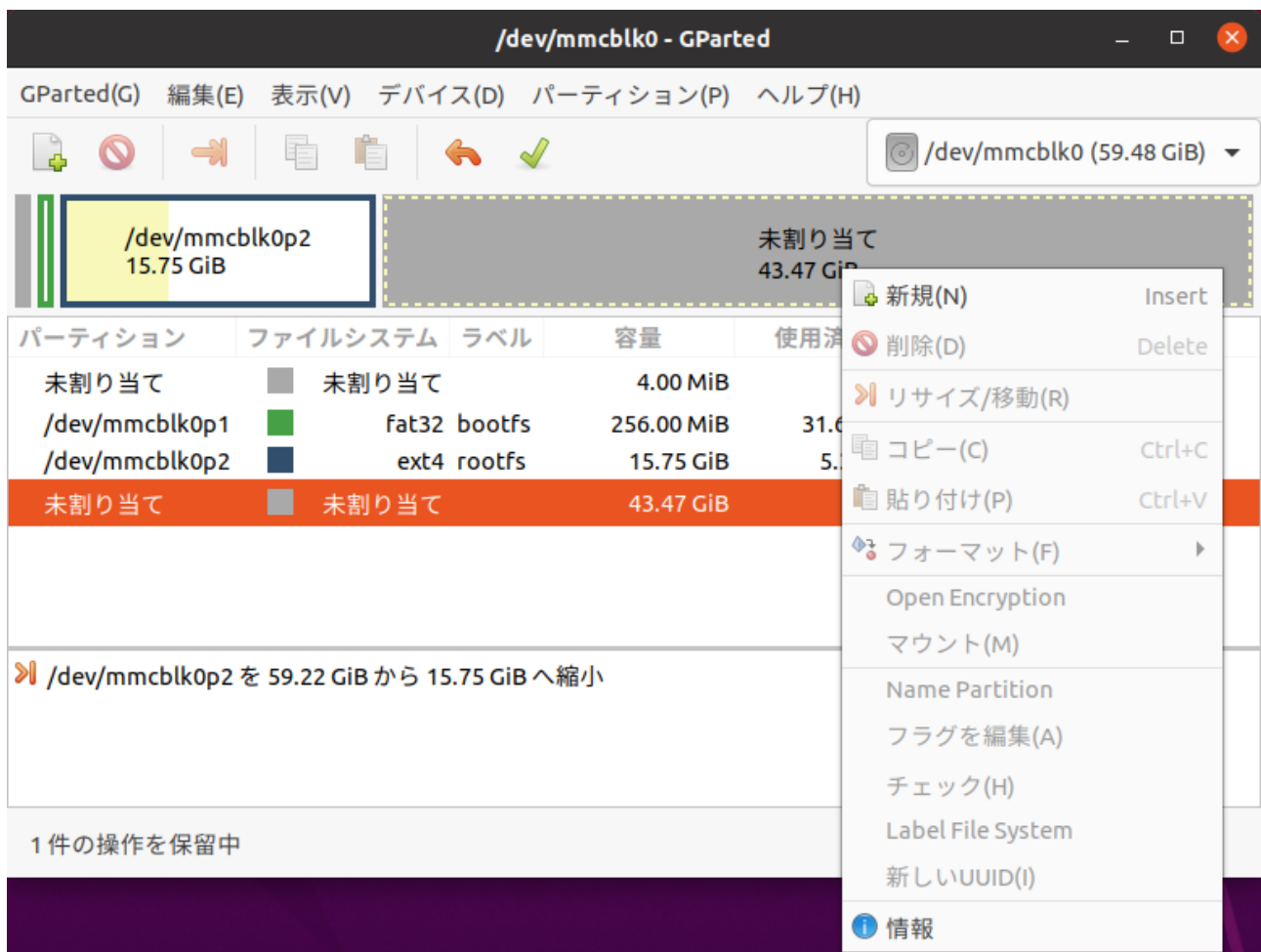
rootfs のパーティションサイズを小さくします。rootfs のパーティションを右クリックし、“リサイズ/移動”を選択します。



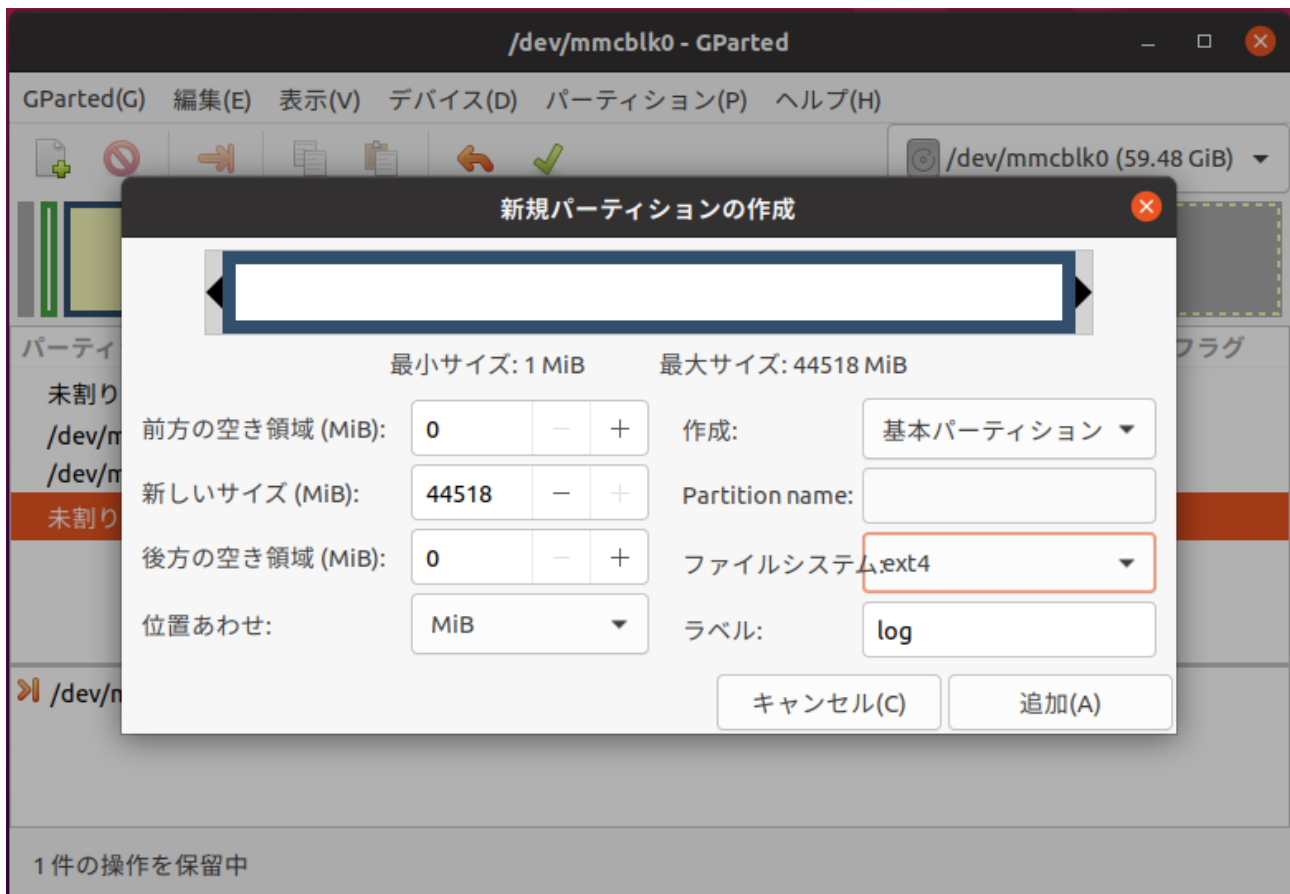
バーをドラッグするか、新しいサイズに容量を数値で入力することで割り当てる領域を設定し、“リサイズ/移動”を選択します。



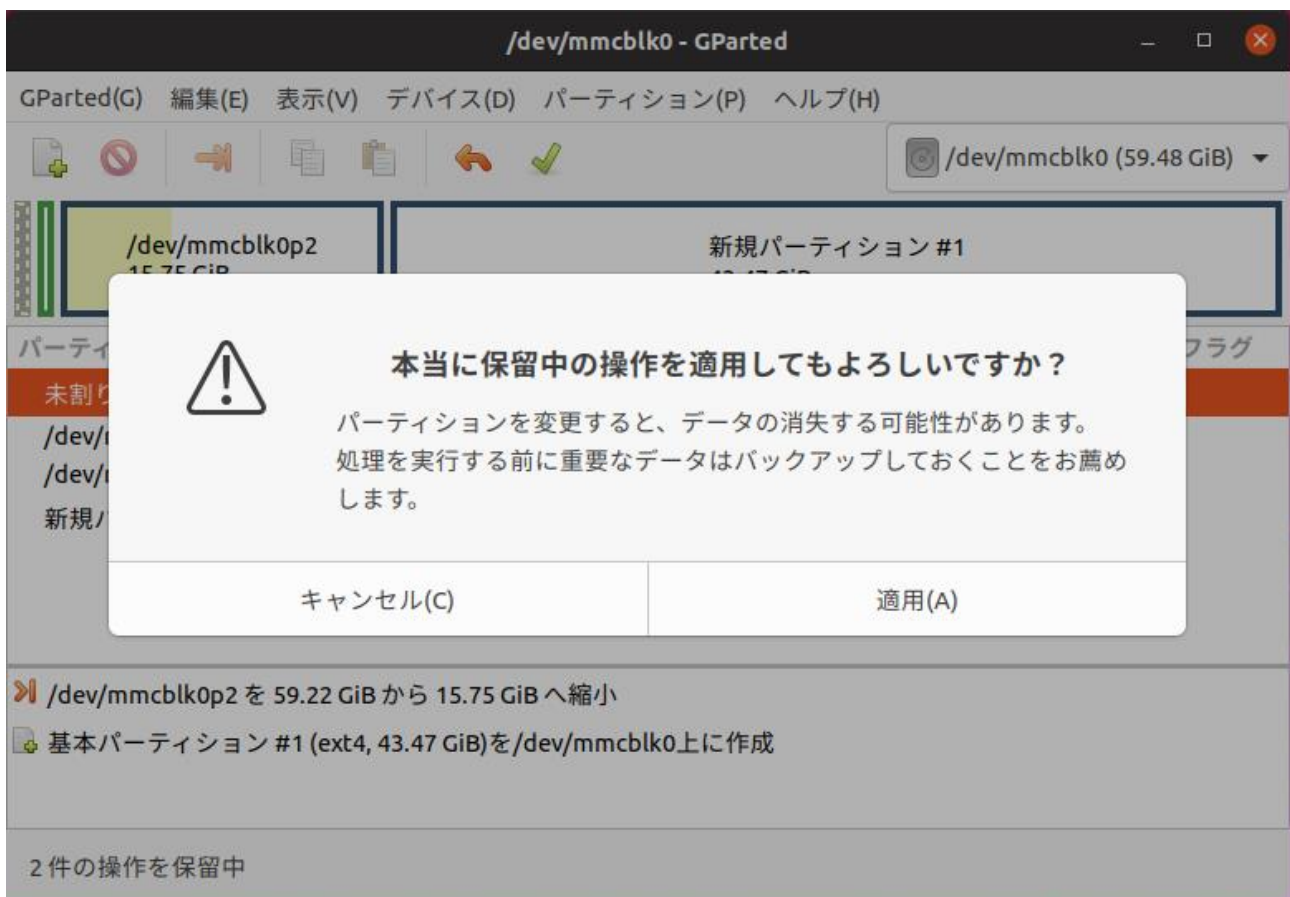
rootfs 領域をリサイズしたことでできた未割り当ての領域を右クリックで選択し、“新規”を選択します。



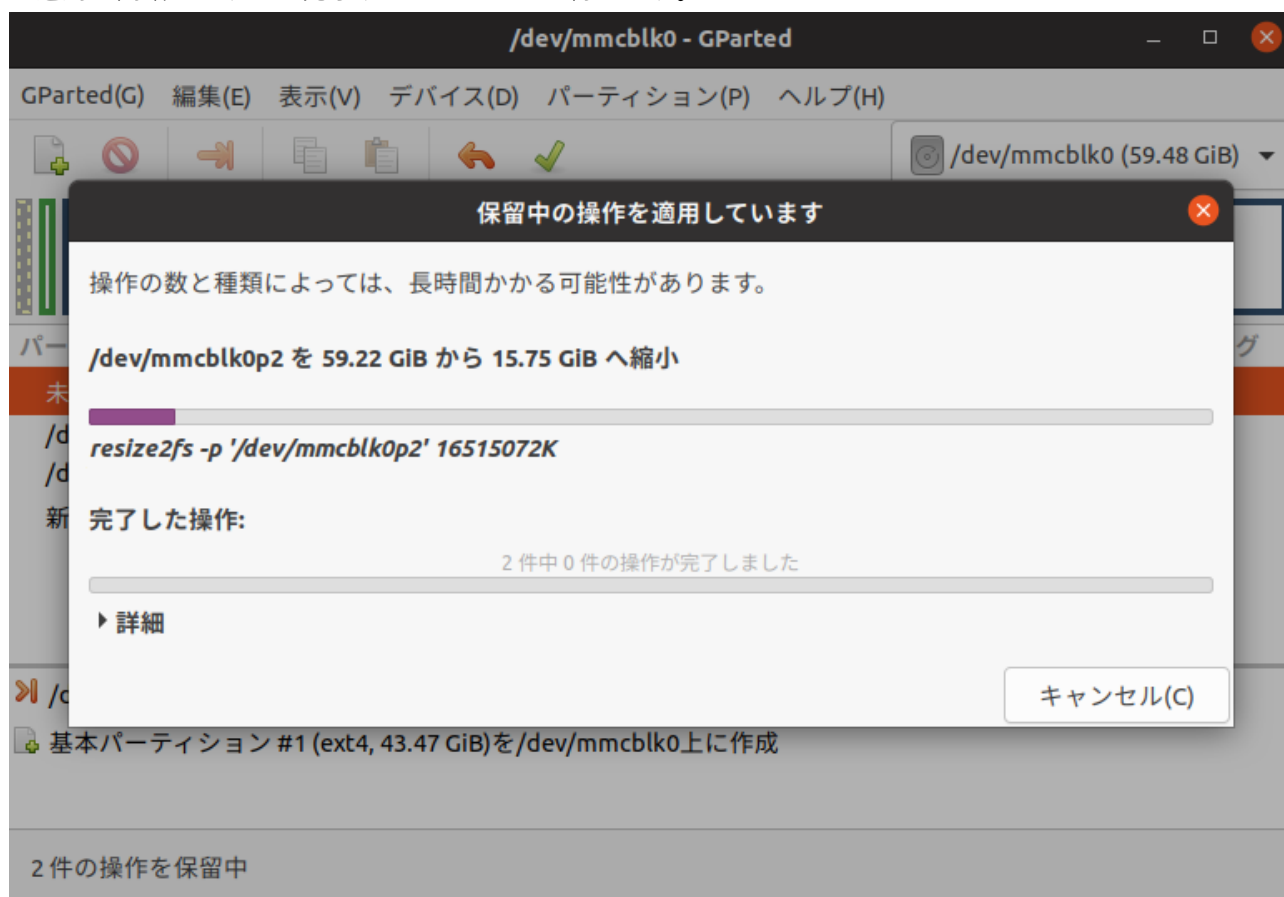
ラベルに任意の名称を入力し、“追加”を選択します。その他の項目は変更する必要はありません。



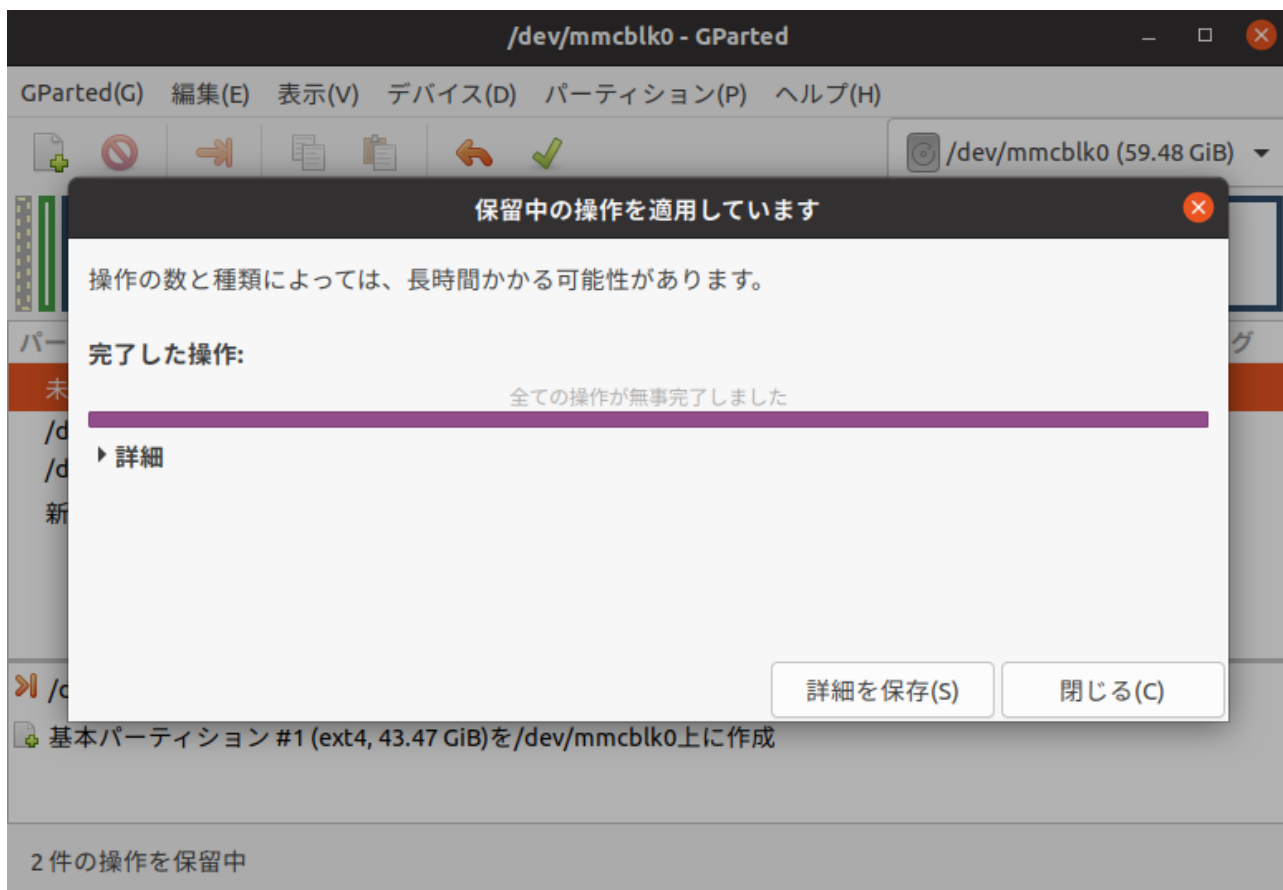
チェックマークを選択しパーティション操作を適用します。確認メッセージが表示されるため”適用”を選択します。



適用が開始しますので完了するまでしばらく待ちます。



すべての操作が無事完了しましたというメッセージが出たらパーティション作成が完了です。“閉じる”を”選択”します。



log 用のパーティションが作成されてますので、GParted を終了します。Linux PC から SD カードを取り出し、Raspberry Pi に差し込んで Raspberry Pi を起動しログインします。

新規作成したパーティションの ID を確認します。

```
$ blkid
/dev/mmcblk0p1: LABEL_FATBOOT="bootfs" LABEL="bootfs" UUID="37CA-39EC"
BLOCK_SIZE="512" TYPE="vfat" PARTUUID="82987921-01"
/dev/mmcblk0p2: LABEL="rootfs" UUID="a4af13c6-d165-4cbd-a9f6-c961fef8255d"
BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="82987921-02"
/dev/mmcblk0p3: LABEL="log" UUID="5ae99d0a-4ed9-409d-8b1d-c036bcae50da"
BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="82987921-03"
```

センサプログラムのログディレクトリを新規作成したパーティションにマウントするように設定を追加します。

```
$ sudo vi /etc/fstab
```

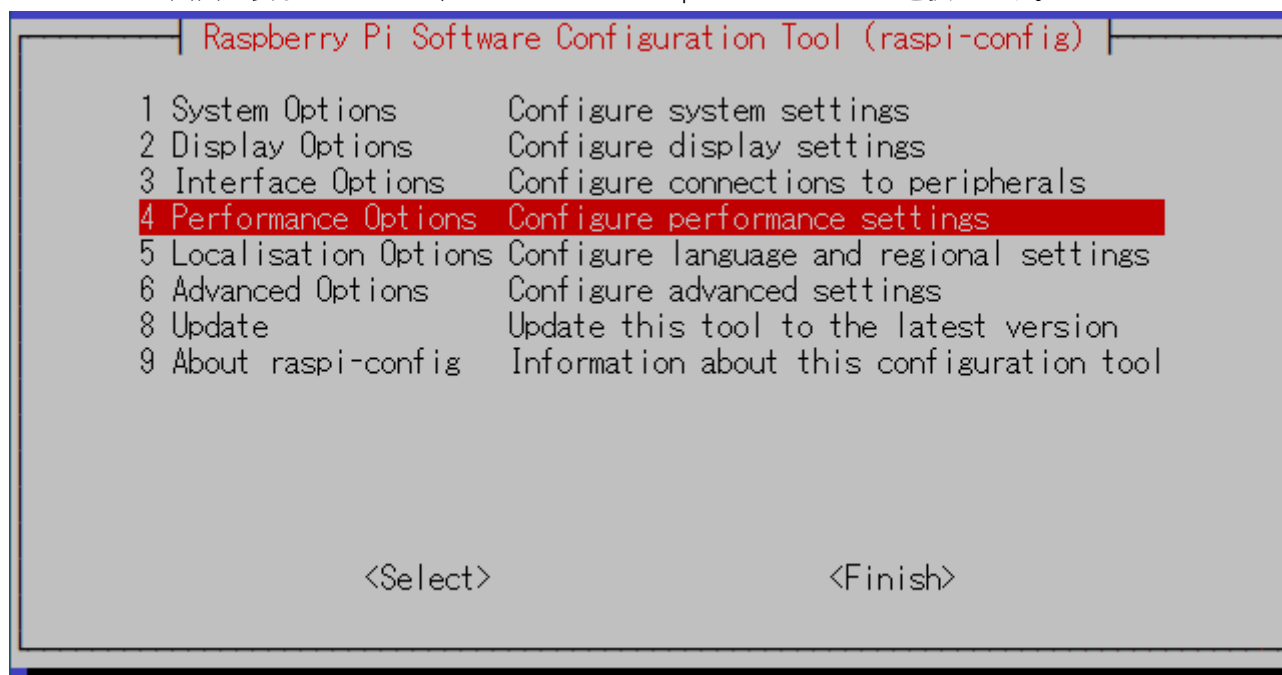
/etc/fstab

proc	/proc	defaults	0	0
PARTUUID=82987921-01	/boot	vfat defaults	0	2
PARTUUID=82987921-02	/	ext4 defaults,noatime	0	1
PARTUUID=82987921-03	/opt/nec/client/log	ext4 defaults	0	3

“raspi-config”を利用して overlay file system の設定をします。

```
$ sudo raspi-config
```

コンフィグ画面が表示されるので、“4 Performance Options”を Enter で選択します。

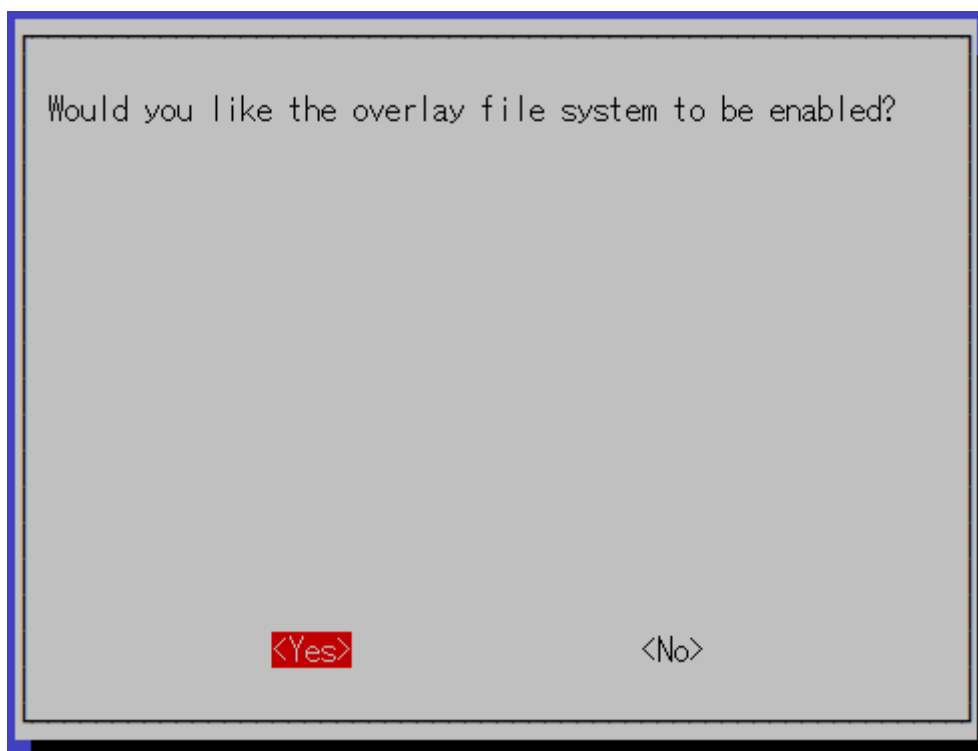


続けて “P3 Overlay File System Enable/disable read-only file system” を Enter で選択します。

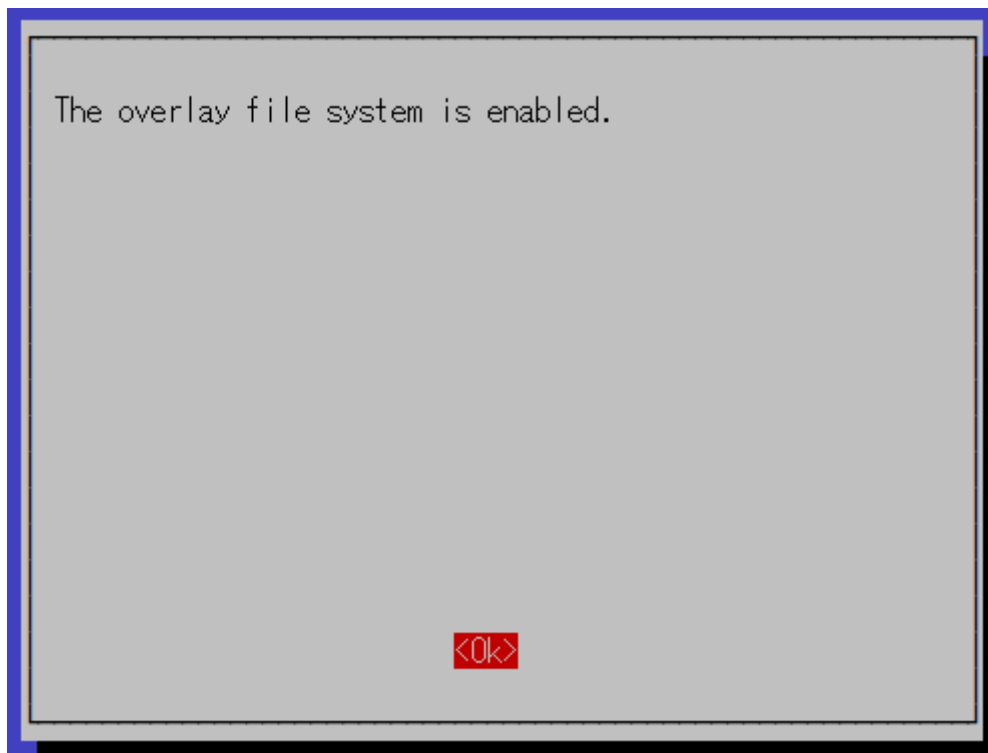




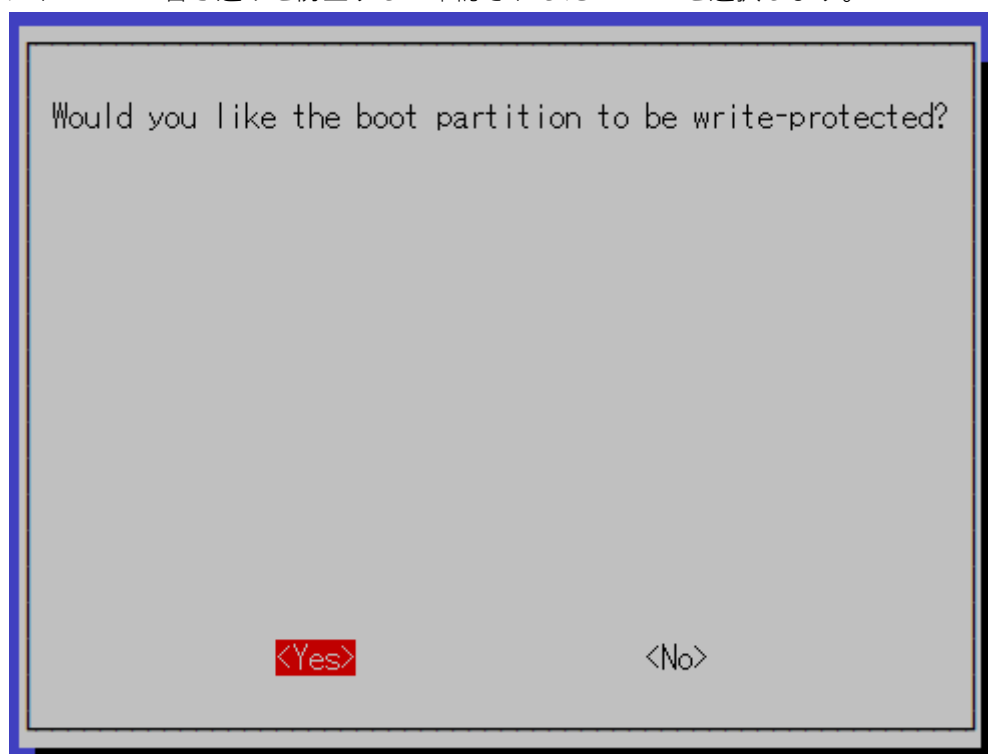
Overlay file systemd を有効にするか確認されるので、“Yes”を選択します。設定が完了するまでしばらく待ちます。



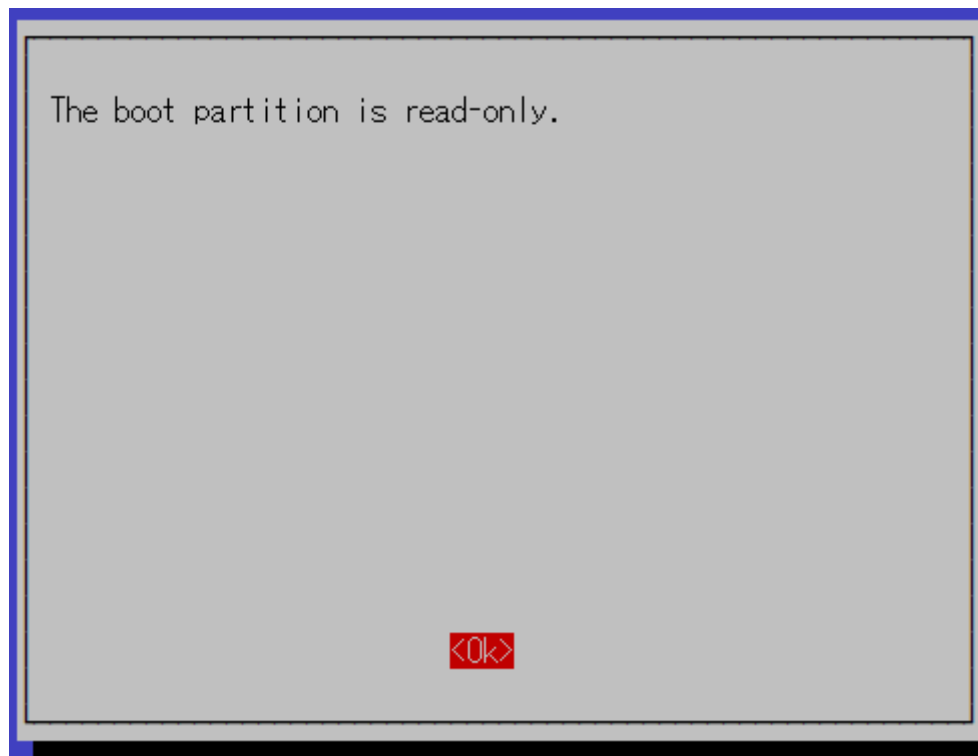
有効化が完了したら“OK”を選択します。



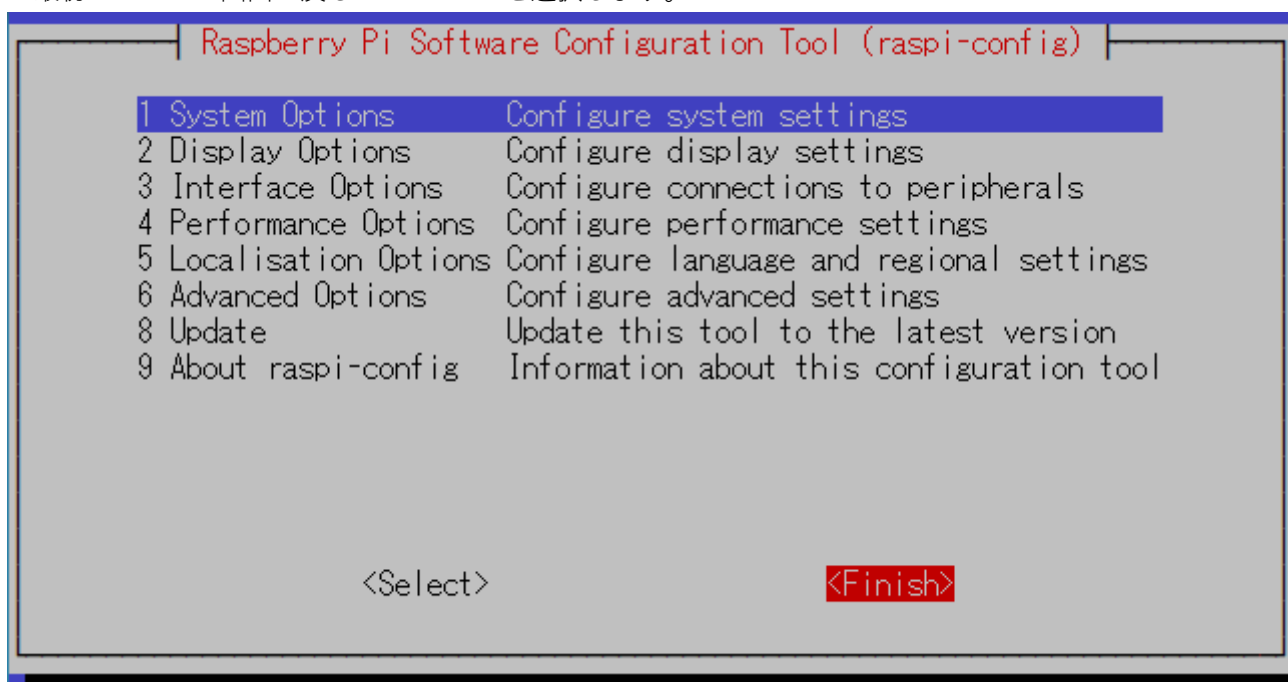
boot パーティションの書き込みを防止するか確認されるため"Yes"を選択します。



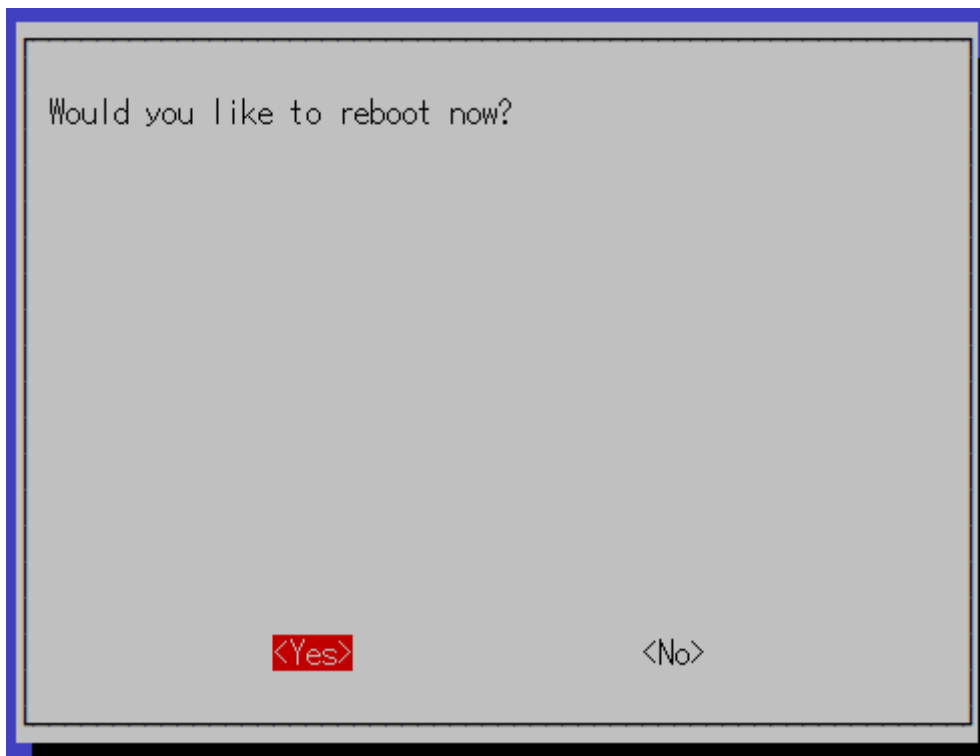
boot パーティションが読み込み専用になったことが表示されるので"OK"を選択します。



最初のメニュー画面に戻るので"Finish"を選択します。



すぐに再起動するか確認されるため "Yes" を選択して再起動します。



再起動後、Overlay Filesystem に root(/)が、新規作成したパーティションにセンサプログラムのログディレクトリがマウントされていれば成功です。この状態では/etc などを含めて書き込んだ内容が OS の再起動で消えるため、/etc などの設定変更をする場合は、raspi-config を利用して overlay 設定をしたときと同様の手順で overlay を無効化し、再起動する必要がありますのでご注意ください。

## 4.5. ファイアウォール設定

センサで外部ネットワークからのアクセスを拒否したい場合は、ファイアウォール設定をしてください。設定手順は以下になります

### 4.5.1. ファイアウォール設定の変更

サンプルのファイアウォール設定は TCP の SSH(22)のみ外部からのアクセスが可能になる設定になっています。サーバで許容する通信を追加したい場合は以下を参考に修正してください。また SSH を拒否したい場合は下記の内容をコメントアウトしてください

```
$ sudo vi /opt/nec/client/bin/clientFw.sh
```

```
/opt/nec/client/bin/clientFw.sh
```

```
# ssh 許可  
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

サービスファイルを systemd に登録します。

```
$ sudo cp /opt/nec/client/bin/clientFw.service ¥  
/etc/systemd/system/clientFw.service  
$ sudo systemctl enable clientFw.service
```

#### 4.5.2. ファイアウォール起動

ファイアウォールの起動と有効化をします。

```
$ sudo systemctl start clientFw.service
```

ファイアウォールの設定手順は以上です。

## 5. 問い合わせ窓口

### 5.1. お問い合わせ先

日本電気株式会社 プラットフォーム・テクノロジーサービス事業部門  
fr-contact@iot.jp.nec.com

### 5.2. 受付時間

平日 9：00～17：00

土曜・日曜・祝日、年末年始を除く