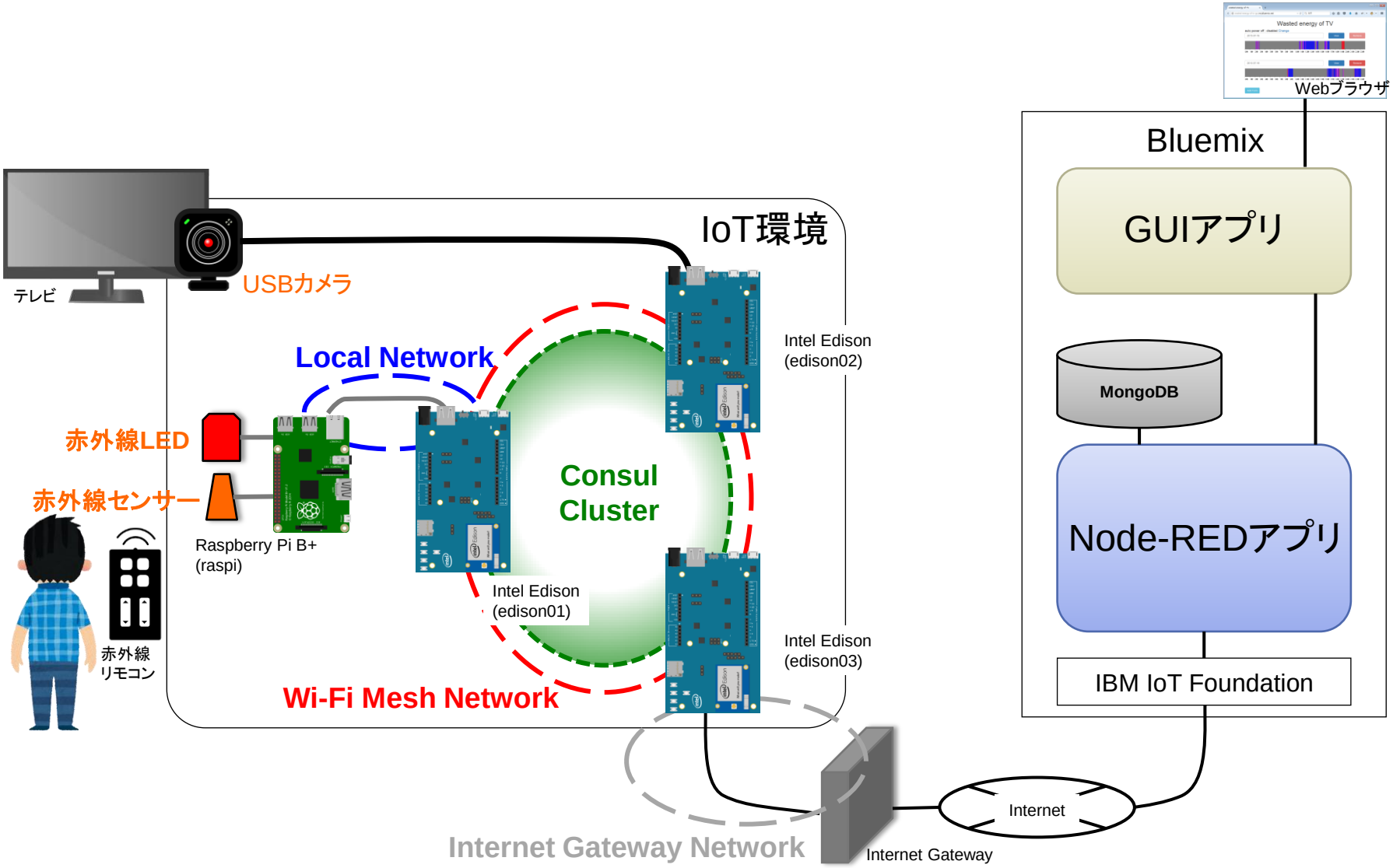
BLUEMIX CHALLENGE 2015

WASTED ENERGY OF TV 環境構築手順書

松井暢之

ID: nobuyuki.matsui@gmail.com

WASTED_ENERGY_OF_TVの全体像



IOT環境の構築

IOT環境の構築に必要なパーツ

Intel Edison Kit for Arduino x 3

- カーネルバージョン　3.10.17-yocto-standard（カスタム）
- Pythonバージョン　2.7.3

Raspberry Pi Model B+ x 1

- カーネルバージョン　3.18.11+
- Pythonバージョン　2.7.3

USBカメラ x 1

- 今回はLOGICOOL ウェブカム C270を利用

USB LANアダプタ x 2

- 今回はLogitec LAN-TX/U2H3BとLogitec LAN-TXU2Cを利用

IOT環境の構築に必要なパーツ

赤外線リモコン受信モジュール x 1

- 今回はキャリア周波数 38kHzのPL-IRM0101を利用した

カラーLED 緑 x 1 赤 x 1黄 x 1

- 今回は順電圧 2.0V、順電流 20mAのカラーLEDを利用

赤外線LED x 1

- 今回は順電圧 1.35V、順電流 50mAのOSI5FU5111Cを利用

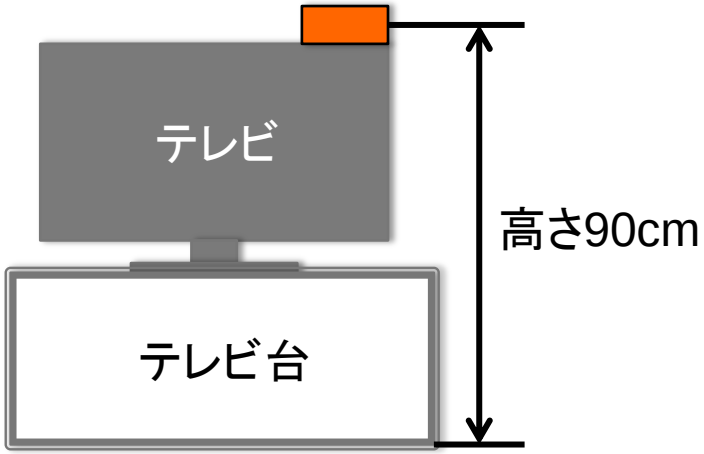
カーボン抵抗

- カラーLED用 150Ω x 3、赤外線LED用 75Ω x 1

その他ブレッドボードやジャンパワイヤ等

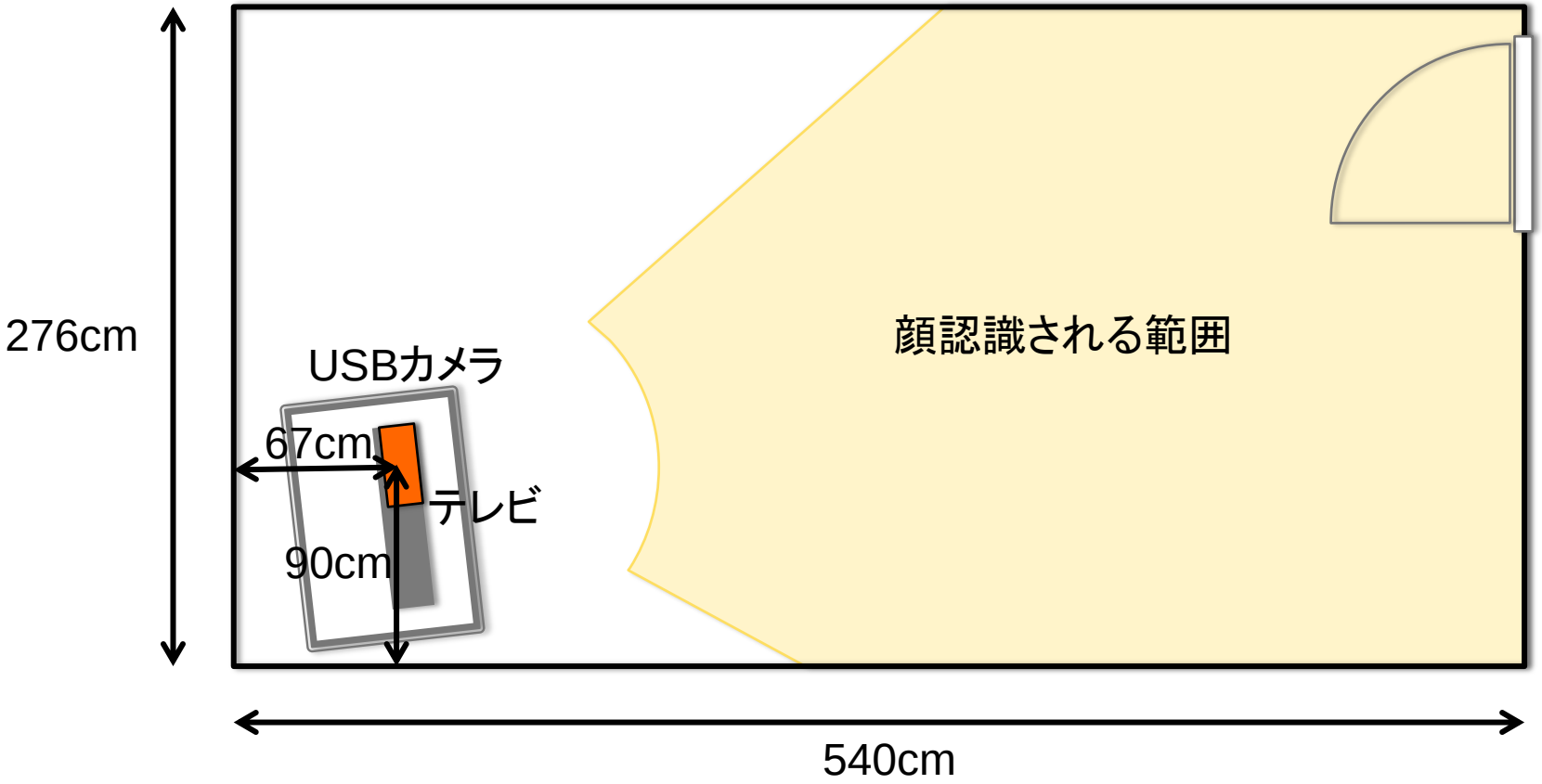
IOT環境の物理的配置

テレビとUSBカメラの配置



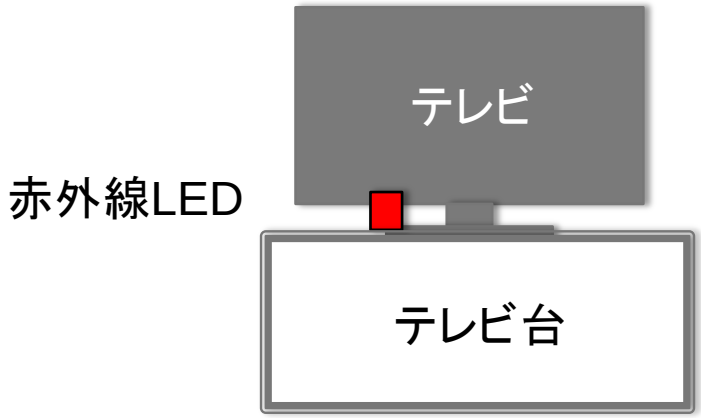
IOT環境の物理的配置

部屋の中でのUSBカメラの有効視界



IOT環境の物理的配置

テレビと赤外線LEDの配置



赤外線LEDとテレビの赤外線受光部の
距離は10cm以内

INTEL EDISONの準備

カスタムカーネルのビルドと導入

- 複数のEdisonをWi-Fi メッシュネットワークで接続するために、B.A.T.M.A.N.-advを有効にしたカスタムカーネルを作成し、Edisonに焼きこむ。
- カーネルを焼きこんだ後、パスワードやホスト名、Wi-Fiの設定を行う。

- カーネルビルドの詳細な手順は、下記URLを参照のこと。
  http://qiita.com/nmatsui/items/cb7020929654a128cf84

INTEL EDISONの準備

すべてのEdisonでOSの設定

- 以下のコマンドでパッケージをアップデートする。

```
root@edison0x:~# opkg update
root@edison0x:~# opkg upgrade
```

- 追加ライブラリのためにbase_feedを追記する。

```
root@edison0x:~# cat /etc/opkg/base-feeds.conf
src/gz all http://repo.opkg.net/edison/repo/all
src/gz edison http://repo.opkg.net/edison/repo/edison
src/gz core2-32 http://repo.opkg.net/edison/repo/core2-32
```

- base_feedを読み込む。

```
root@edison0x:~# opkg update
```

- Timezoneを設定する。

```
root@edison0x:~# timedatectl set-timezone Asia/Tokyo
```

INTEL EDISONの準備

すべてのEdisonでgitのインストール

- アプリケーションをgit cloneするため、gitをインストールする。

```
root@edison0x:~# opkg install git
root@edison0x:~# git config --global user.name "名前"
root@edison0x:~# git config --global user.email メールアドレス
root@edison0x:~# git config --global color.ui false
```

INTEL EDISONの準備

すべてのEdisonでbatctlのインストール

- ビルドサーバから、カーネルビルド時に生成されたbatctlコマンドのipkをすべてのEdisonに転送する。

```
user@build-server:$ cd edison-src/build/tmp/deploy/ipk/core2-32
user@build-server:$ scp ./batctl_2015.0-r0_core2-32.ipk root@edisonのIPアドレス:/tmp
```

- batctlをインストールする。

```
root@edison0x:~# opkg install /tmp/batctl_2015.0-r0_core2-32.ipk
```

- batctlのビルドとインストールの詳細な手順は、下記URLを参照のこと。
  http://qiita.com/nmatsui/items/cb7020929654a128cf84

RASPBERRY PIの準備

Raspbian Wheezyの導入

- Raspbian Wheezy（2015-05-15版）を導入し、下記のように設定する。
  - ユーザ名　　pi
  - パスワード　　任意
  - ホスト名　　raspi
  - ロケール　　ja_JP.UTF-8
  - Timezone　　Asia/Tokyo
  - （今回はユーザー名をpi、ホスト名をraspiに設定）
- 有線LANに接続し、インターネットに通信できることを確認する。
- 以下のコマンドでパッケージをアップデートする。

```
pi@raspi ~ $ sudo apt-get update
pi@raspi ~ $ sudo apt-get upgrade
```

RASPBERRY PIの準備

avahi-daemonのインストール

- ホスト名でアクセスできるようにavahi-daemonをインストールする。

```
pi@raspi ~ $ sudo apt-get install avahi-daemon
```

lircのインストール

- 赤外線センサーや赤外線LEDを扱うためにlircをインストールする。

```
pi@raspi ~ $ sudo apt-get install lirc
```

再起動

- Raspberry Piを一度再起動する。

RASPBERRY PIの準備

lircの設定

- Raspberry Pi起動時にlircを適切に読み込むために、/etc/modulesと/boot/config.txtに設定を追加する。

```
pi@raspi ~ $ sudo su –
root@raspi:~# echo 'lirc_dev' >> /etc/modules
root@raspi:~# echo 'dtoverlay=lirc-rpi, gpio_in_pin=14, gpio_out_pin=15' >> /boot/config.txt
```

- lircを起動スクリプトに登録する。

```
root@raspi:~# update-rc.d lirc defaults
```

RASPBERRY PIの準備

lircの設定

- ハードウェア設定を登録するために、/etc/lirc/hardware.confの該当箇所を修正する。

```
LIRCD_ARGS="--uinput"
LOAD_MODULES=true
DRIVER="default"
DEVICE="/dev/lirc0"
MODULES="lirc_rpi"
```

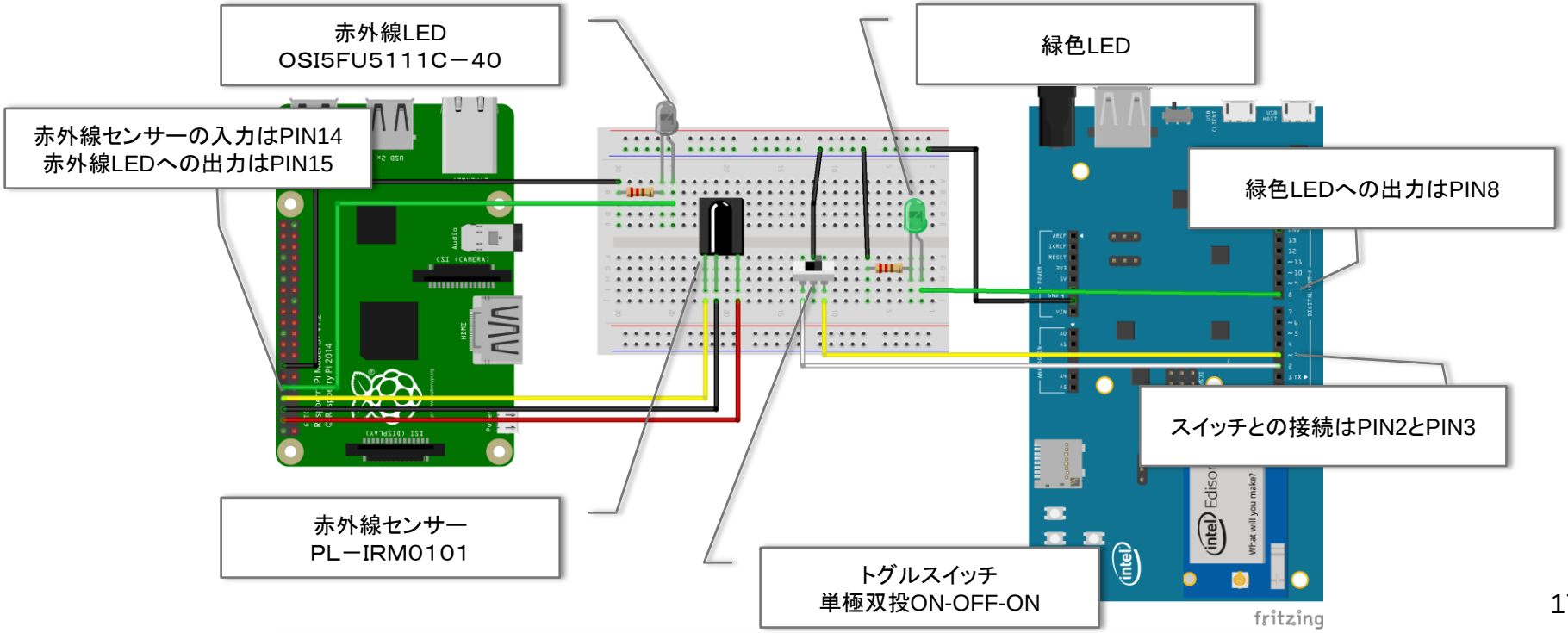
- 再起動した後、lirc関連モジュールが認識されていることを確認する。

```
root@raspi:~# reboot
pi@raspi ~ $ lsmod | grep lirc
lirc_rpi                8582  3
lirc_dev               11060  1 lirc_rpi
rc_core                23526  1 lirc_dev
pi@raspi ~ $ ls -l /dev/lirc*
crw-rw---T 1 root video 246, 0 Jan  1  1970 /dev/lirc0
```

赤外線センサー ＆ 赤外線LED の回路

Edison（edison01）とRaspberry Piを用いて下記の回路を構成する

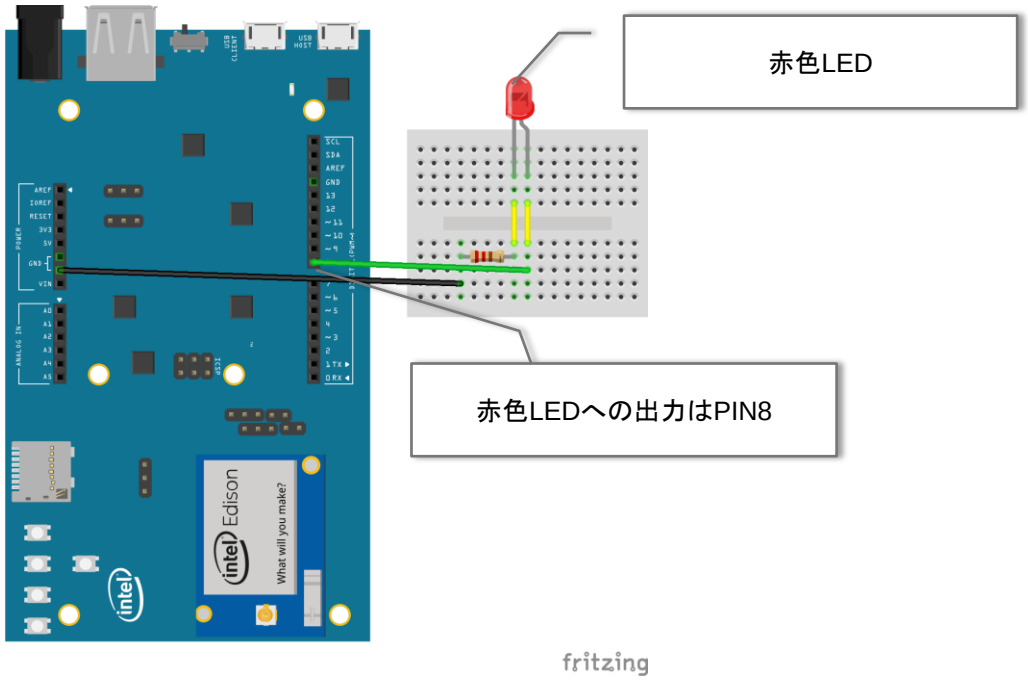
- Intel EdisonのArduinoボードにUSB LANアダプタを差し込み、Raspberry Piの有線LANポートとクロスケーブルで直結する。
- 今回の赤外線LEDには75Ω、緑色LEDには150Ωの抵抗を付ける。



USBカメラの回路

Edison（edison02）で下記の回路を構成する

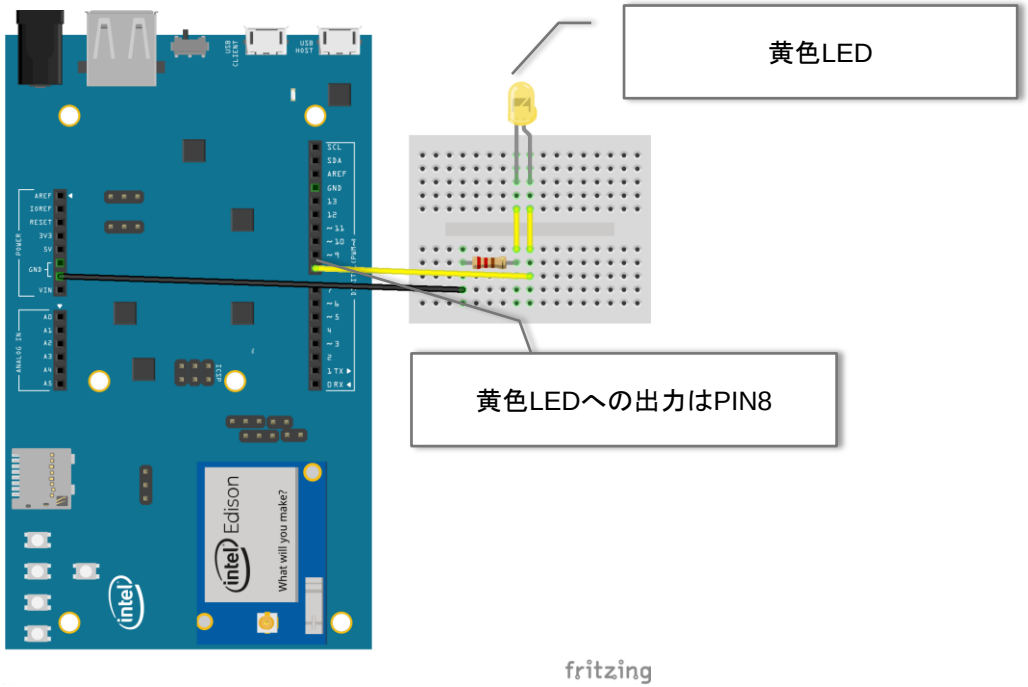
- Intel EdisonのArduinoボードにUSBカメラを差し込む。
- 今回の赤色LEDには150Ωの抵抗を付ける。



MQTTクライアントの回路

Edison（edison03）で下記の回路を構成する

- Intel EdisonのArduinoボードにUSB LANアダプタを差し込み、インターネットへ接続可能なルータへ接続する。
- 今回の黄色LEDには150Ωの抵抗を付ける。



赤外線リモコンの学習

lircを用いて赤外線リモコンの「電源ON/OFFボタン」を学習する

- lircを停止する。

```
pi@raspi ~ $ sudo service lirc stop
```

- irrecordを用い、画面の指示に従って赤外線パターンを学習する。
  （ボタン名にはpowerと入力する）

```
pi@raspi ~ $ irrecord -n -d /dev/lirc0 ~/lircd_tv.conf
```

- 生成されたファイル（~/lircd_tv.conf）を開き、nameをtvに修正する。

```
…
   name  tv
…
```

- 生成されたファイルの内容で/etc/lirc/lircd.confを置き換える。

```
pi@raspi ~ $ sudo sh -c "cat $HOME/lircd_tv.conf > /etc/lirc/lircd.conf"
```

- lircを起動する。

```
pi@raspi ~ $ sudo service lirc start
```

赤外線リモコンの学習

学習した赤外線リモコンパターンをテストする

- 学習させた赤外線リモコンパターンが認識されていることを確認する。

```
pi@raspi ~ $ irsend LIST tv ""
irsend: 0000000000000001 power
```

- irwを起動した後に赤外線センサーへ向けてリモコンの「電源ボタン」を押し、赤外線パターンを認識したことを確認する。

```
pi@raspi ~ $ irw
0000000000000001 00 power tv
```

- irsendを用いて赤外線LEDから赤外線パターンを送出し、テレビの電源が操作できることを確認する。

```
pi@raspi ~ $ irsend SEND_ONCE tv power
```

赤外線受信時のアクションを定義する

赤外線センサーが信号を受信した際のアクションを定義する

- powerパターンを受信した際のアクションを/etc/lirc/lircrcに定義する。

```
pi@raspi ~ $ cat /etc/lirc/lircrc
begin
    button = power
    prog = irexec
    config = python /home/pi/wasted_energy_of_tv_iot/power_ir_recv_client.py
end
```

- lircを再起動して、追加したアクションを認識させる。

```
pi@raspi ~ $ sudo service lirc restart
```

EDISONのPYTHON環境を整備する

すべてのEdisonのpyhon環境を整備する

- pipを利用できるようにする。

```
root@edison0x:~# echo 'export PATH=/home/root/.local/bin:$PATH' >> .profile
root@edison0x:~# echo 'export PYTHONPATH=/home/root/.local/lib/python2.7/' >> .profile
root@edison0x:~# source ~/.profile
root@edison0x:~# echo -e "[install]¥nuser=1" >> .pydistutils.cfg
root@edison0x:~# wget https://bootstrap.pypa.io/ez_setup.py --no-check-certificate -O - | python
root@edison0x:~# easy_install pip
```

EDISONにCONSULを導入する

すべてのEdisonにconsulを導入する

- Intel Edisonは32bitのx86バイナリに互換性があるため、Edison上でコンパイルせずともプレビルドされたconsulをそのまま導入できる。
  - ※紙面の都合上折り返しているが、wgetは1行で実行すること

```
root@edison0x:~# mkdir -p /usr/local/bin
root@edison0x:~# wget https://dl.bintray.com/mitchellh/consul/0.5.2_linux_386.zip
  -qO /tmp/consul_0.5.2_386.zip --no-check-certificate
root@edison0x:~# unzip /tmp/consul_0.5.2_386.zip -d /usr/local/bin
root@edison0x:~# chmod +x /usr/local/bin/consul
```

- pythonのconsulライブラリをpipで導入する。

```
root@edison0x:~# pip install python-consul
```

OPENCVとMQTTクライアントを導入する

USBカメラを接続するEdison（edison02）にopencvを導入する

- opkgを用いてopencvとpythonライブラリを導入する。

```
root@edison02:~# opkg install opencv
root@edison02:~# opkg install python-opencv
```

Bluemixと接続するEdison（edison03）にMQTTクライアントを導入する

- pipを用いてpaho-mqttを導入する。

```
root@edison03:~# pip install paho-mqtt
```

プログラムを取得する

すべてのEdison及びRaspberry PiへIoT部分のアプリを導入する

- githubからwasted_energy_of_tv_iotをcloneする。

```
root@edison0x:~# git clone https://github.com/nmatsui/wasted_energy_of_tv_iot.git
```

```
pi@raspi ~ $ git clone https://github.com/nmatsui/wasted_energy_of_tv_iot.git
```

Wi-Fiメッシュネットワークを構成する

すべてのEdisonでWi-Fiメッシュネットワークを構成する

- screenコマンドで接続し、内蔵Wi-Fiをad-hocモードにする。

```
root@edison0x:~# wpa_cli -iwlan0 disconnect
root@edison0x:~# wpa_cli -iwlan0 remove_network all
root@edison0x:~# wpa_cli -iwlan0 add_network
root@edison0x:~# wpa_cli -iwlan0 set_network 0 frequency 2412
root@edison0x:~# wpa_cli -iwlan0 set_network 0 mode 1
root@edison0x:~# wpa_cli -iwlan0 set_network 0 ssid ¥"batman¥"
root@edison0x:~# wpa_cli -iwlan0 set_network 0 auth_alg OPEN
root@edison0x:~# wpa_cli -iwlan0 set_network 0 key_mgmt NONE
root@edison0x:~# wpa_cli -iwlan0 set_network 0 scan_ssid 1
root@edison0x:~# wpa_cli -iwlan0 select_network 0
root@edison0x:~# wpa_cli -iwlan0 enable_network 0
```

- B.A.T.M.A.N.-advを用いてメッシュネットワークを構成する。

```
root@edison0x:~# modprobe batman-adv
root@edison0x:~# batctl if add wlan0
root@edison0x:~# ip link set dev bat0 up
```

Wi-Fiメッシュネットワークを構成する

edison01にIPv4アドレスを与える

- B.A.T.M.A.N.-advによって構成されたbat0にIPv4アドレスを与える。

```
root@edison01:~# ip addr add 10.0.0.1/24 dev bat0
```

edison02にIPv4アドレスを与える

- B.A.T.M.A.N.-advによって構成されたbat0にIPv4アドレスを与える。

```
root@edison02:~# ip addr add 10.0.0.2/24 dev bat0
```

edison03にIPv4アドレスを与える

- B.A.T.M.A.N.-advによって構成されたbat0にIPv4アドレスを与える。

```
root@edison03:~# ip addr add 10.0.0.3/24 dev bat0
```

edison01, edison02, edison03がお互いに通信できることを確認する

- 10.0.0.0/24のアドレス帯でお互いに通信できることを確認する。

EDISON03の有線IFを設定する

edison03から有線LANでインターネットに接続する

- ip addrを用いてUSB LANアダプタのインタフェースを特定する。
- USB LANアダプタのインタフェースを起動し、インターネットゲートウェイからDHCPでIPv4アドレスを払いだしてもらう。

```
root@edison03:~# ip link set dev enp0s17u1u4 up
root@edison03:~# udhcpc -i enp0s17u1u4
```

- edison03からインターネットに接続できることを確認する。

```
root@edison03:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=56 time=9.242 ms
64 bytes from 8.8.8.8: seq=1 ttl=56 time=9.193 ms
```

CONSULクラスタを起動する

edison01でconsulクラスタを起動する

- bootstrapノードとしてconsulクラスタを起動する。

```
root@edison01:~# rm -rf /tmp/consul
root@edison01:~# nohup consul agent -data-dir=/tmp/consul -server -bootstrap-expect=3 -bind=10.0.0.1 &
```

edison02でconsulクラスタにJOINする

- edison01で起動させたconsulクラスタにJOINする。

```
root@edison02:~# rm -rf /tmp/consul
root@edison02:~# nohup consul agent -data-dir=/tmp/consul -server -join=10.0.0.1 -bind=10.0.0.2 &
```

edison03consulクラスタにJOINする

- edison01で起動させたconsulクラスタにJOINする。

```
root@edison03:~# rm -rf /tmp/consul
root@edison03:~# nohup consul agent -data-dir=/tmp/consul -server -join=10.0.0.1 -bind=10.0.0.3 &
```

CONSULクラスタを起動する

consulクラスタの状態を確認する

- edison01, edison02, edison03でconsul membersを実行し、3台のIntel Edisonがすべてconsulクラスタに参加しており、ステータスがaliveであることを確認する。

```
root@edison0x:~# consul members
Node      Address         Status  Type    Build  Protocol  DC
edison01  10.0.0.1:8301  alive   server  0.5.2  2         dc1
edison02  10.0.0.2:8301  alive   server  0.5.2  2         dc1
edison03  10.0.0.3:8301  alive   server  0.5.2  2         dc1
```

EDISON01の有線IFを設定する

edison01とRaspberry Piの閉じたネットワークの設定を行う

- ip addrを用いてUSB LANアダプタのインタフェースを特定する。
- USB LANアダプタのインタフェースを起動し、10.10.0.1/24のIPv4アドレスを与える。

```
root@edison01:~# ip link set dev enp0s17u1 up
root@edison01:~# ip addr add 10.10.0.1/24 dev enp0s17u1
```

RASPBERRY PIの有線IFを設定する

有線LANインタフェースの設定

- Intel Edisonの１台とRaspberry Piで閉じたネットワークを構成するために、Raspberry Piの有線LANインタフェースを静的に設定する。

```
pi@raspi ~ $ cat /etc/network/interfaces
auto lo

iface lo inet loopback

iface eth0 inet static
address 10.10.0.2
netmask 255.255.255.0
gateway 10.10.0.1

iface default inet dhcp
```

- Raspberry Piを再起動し、Raspberry Piとedison01をクロスケーブルで接続する。

EDISON03とIOT FOUNDATIONを接続する

Bluemix IoT Node-REDアプリを作成する

- 下記URLを参照に、BluemixにIoT Node-REDアプリを作成する。
  http://qiita.com/nmatsui/items/379b64b9a3766df21549

edison03とIBM IoT Foundationと接続する

- `ip addr`を用いてedison03のUSB LANアダプタのMACアドレスを調べ、デバイスIDを生成する。
- IBM IoT Foundationに接続設定を行い、認証トークンを払い出す。
- 下記フォーマットでデバイス情報を書き出し、edison03に保存する。

```
org=与えられたorganization
Type=登録時に設定したデバイスタイプ
id=登録したデバイスID
auth-method=token
auth-token=払いだされた認証トークン
```

WASTED_ENERGY_OF_TVのIOTアプリを起動する

Raspberry Piで赤外線LED送信サーバを起動する

- Raspberry Piにcloneされたwasted-energy-of-tv-iotディレクトリで以下のコマンドを起動する。

```
pi@raspi ~ $ nohup ./power_ir_send_server.py &
```

edison01で赤外線LED送信クライアントをconsul watchに登録する

- edison01で以下のコマンドを起動する。
  - ※紙面の都合上折り返しているが、consul watchは1行で実行すること

```
root@edison01:~# nohup consul watch -type=event -name="power"
  "/home/root/wasted_energy_of_tv_iot/power_ir_send_client.py" &
```

edison01で赤外線センサー受信サーバを起動する

- edison01にcloneされたwasted-energy-of-tv-iotディレクトリで以下のコマンドを起動する。

```
root@edison01:~# nohup ./power_ir_recv_server.py &
```

WASTED_ENERGY_OF_TVのIOTアプリを起動する

edison02で顔認識アプリを起動する

- edison02にcloneされたwasted-energy-of-tv-iotディレクトリで以下のコマンドを起動する。

```
root@edison02:~# nohup ./face_detect.py face_features/haarcascade_frontalface_default.xml &
```

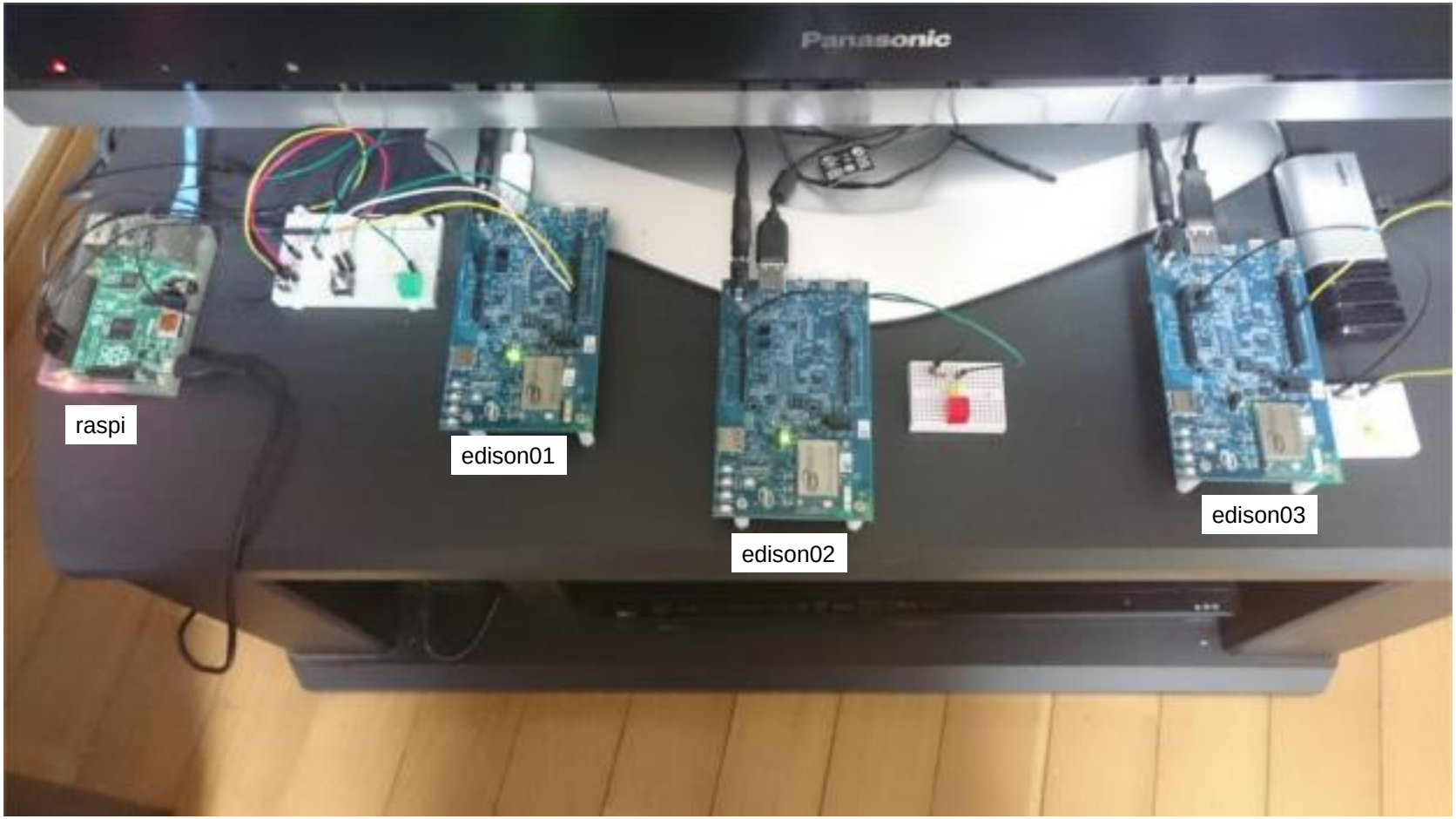
edison03でMQTT送受信アプリを起動する

- edison03にcloneされたwasted-energy-of-tv-iotディレクトリで以下のコマンドを起動する。

```
root@edison03:~# nohup ./watch_rate_publish.py デバイス情報ファイル &
```

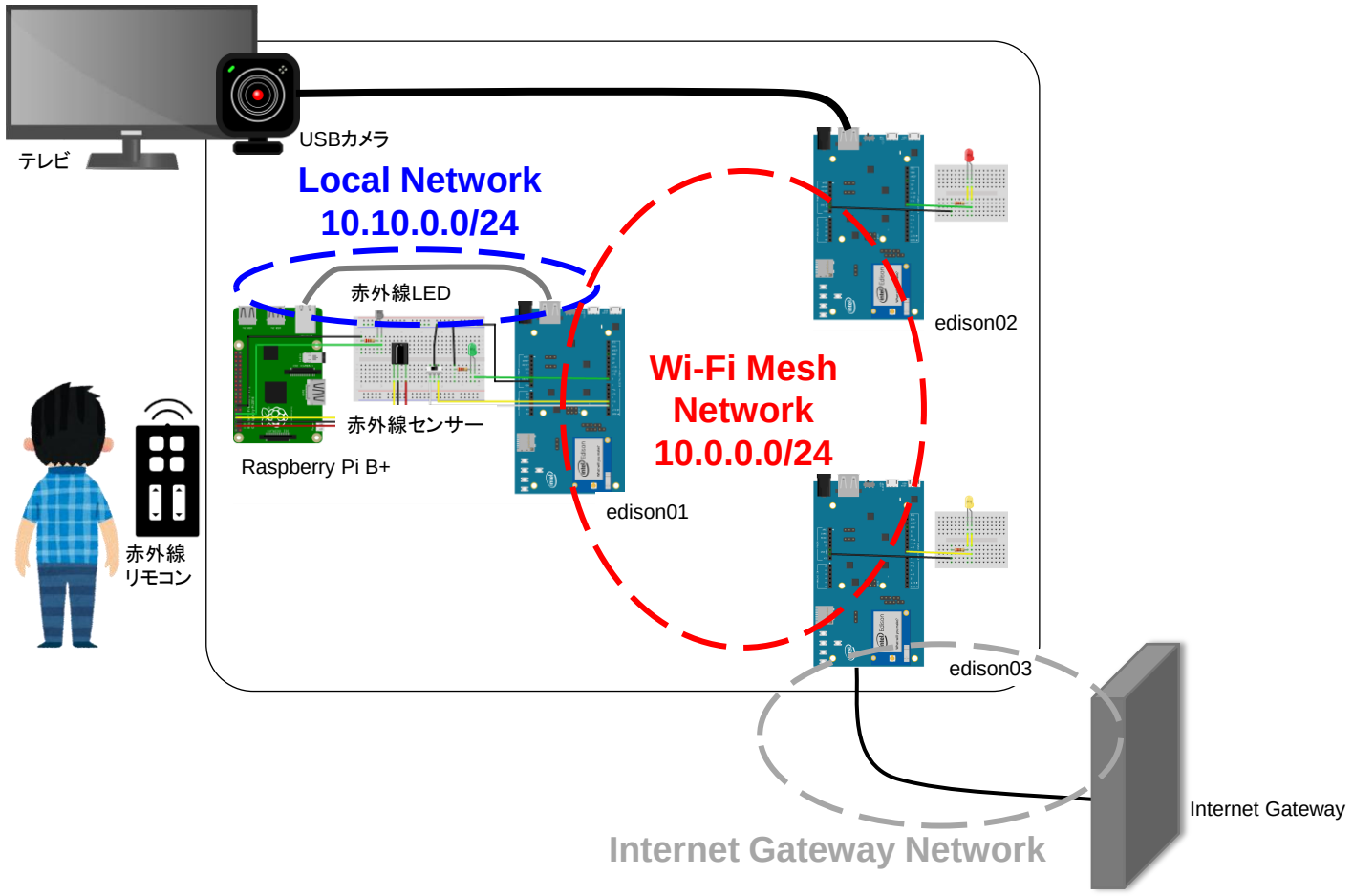
WASTED_ENERGY_OF_TVの IOTアプリを起動する

構築されたWasted_energy_of_TV　IoT環境



WASTED_ENERGY_OF_TVの IOTアプリを起動する

構築されたwasted-energy-of-tv　IoT環境



NODE-REDアプリの構築

WASTED_ENERGY_OF_TVのNODE-REDアプリを変更する

Node-REDアプリのフロー定義を取得する

- githubからwasted_energy_of_tv_noderedをcloneする。

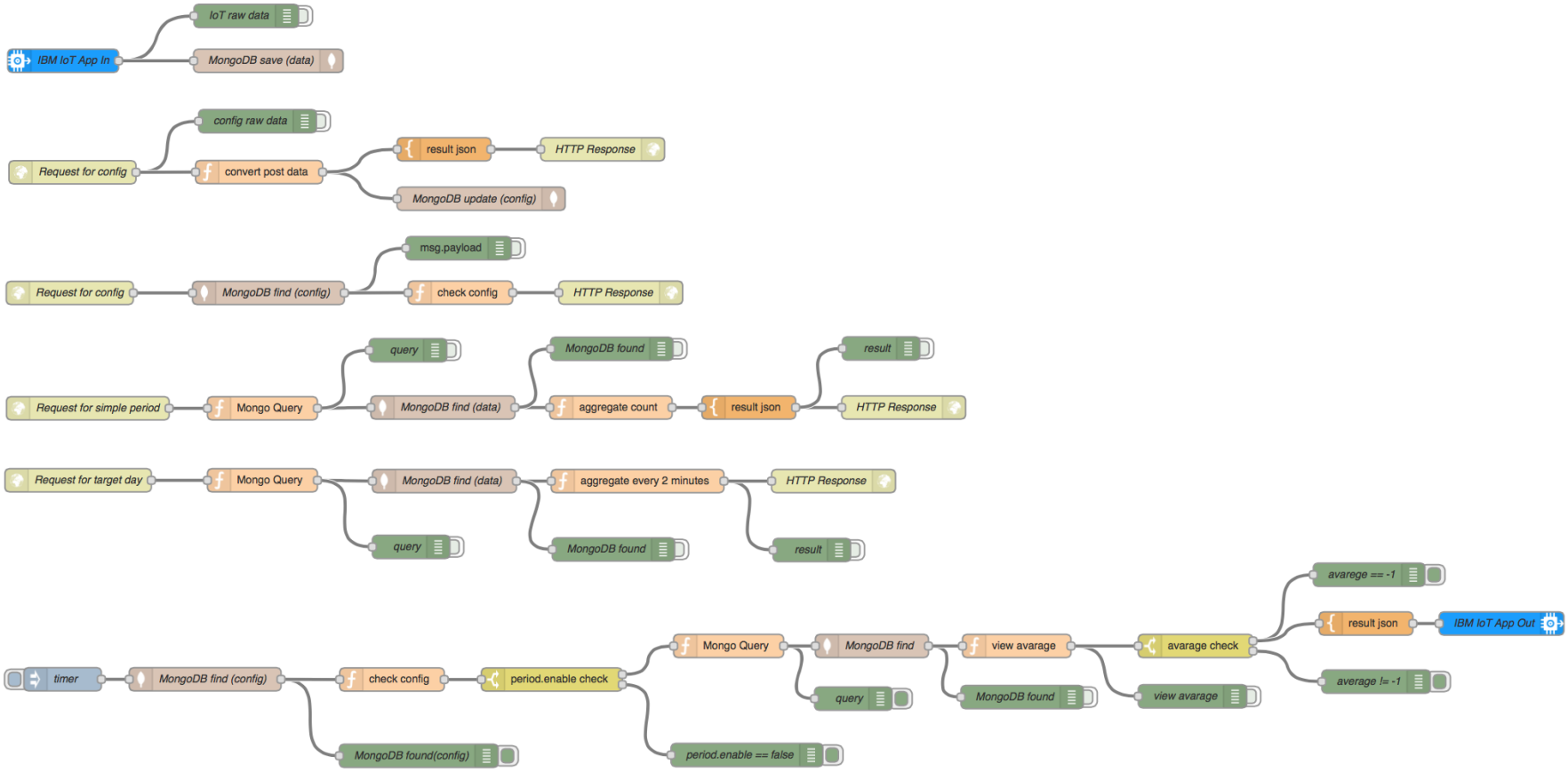
```
# git clone https://github.com/nmatsui/wasted_energy_of_tv_nodered.git
```

BluemixのNode-REDアプリへ取得したフローを登録する

- 34ページで作成したNode-REDアプリのフローをすべて消去する。
- メニューから、「Import」→「Clipboard」と選択し、取得したフロー定義をコピーして「OK」する。
- デバイスIDやMongoDBのサービス名はアプリケーションごとに異なるため、読み込まれたフローの当該ノードをクリックして設定値を変更する。
    - MongoDBが認識されない場合、サービス名を一旦「External service」に変更して「OK」し、再度MongoLabのサービス名に変更し直すとよい
- 「Deploy」ボタンをクリックしてフローを登録する。

WASTED_ENERGY_OF_TVのNODE-REDアプリを起動する

登録されたNode-REDアプリのフロー全体



GUIアプリの構築

WASTED_ENERGY_OF_TVのGUIアプリを起動する

GUIアプリを取得する

- githubからwasted_energy_of_tv_guiをcloneする。

```
$ git clone https://github.com/nmatsui/wasted_energy_of_tv_gui.git
```

nginxのビルドパックを用いてBluemixへGUIアプリを登録する

- Bluemix APIのエンドポイントを設定し、Bluemixへログインする。

```
$ cf api https://api.ng.bluemix.net
$ cf login
```

- nginxのbuildpackを用いてGUIアプリをBluemixへ登録する。
    - 今回はGUIアプリ名をwasted-energy-of-tv-guiにした

```
$ cf push -b https://github.com/cloudfoundry-community/nginx-buildpack wasted-energy-of-tv-gui
```

WASTED_ENERGY_OF_TVのGUIアプリを起動する

Wasted energy of TVのGUI

- http://wasted-energy-of-tv-gui.mybluemix.net にアクセスして画面が表示されることを確認する、