

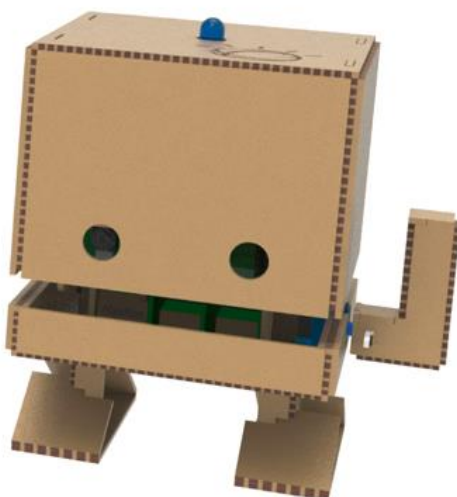


写真を撮って内容を分析

Watson、「Text to Speech」「Visual Recognition」と Raspberry Pi、Bluetooth スピーカー、Node-Red を使って、撮った写真について話します。

<使用する Watson サービス>

「Text to Speech」、 「Visual Recognition」



こんにちは、私の名前は TJBot です！

私は、楽しい方法で Watson サービスを理解するために、お手伝いをするオープンソースプロジェクトです。

TJBot は、IBM 最初の会長兼最高経営責任者（CEO）であるトーマス・J・ワトソンの名前を愛称としています。

TJBot は、IBM Research の Maryam Ashoori によって、認知対象の設計と実装におけるベストプラクティスを見つけるための実験として作成されました。

もくじ

- 【1】Web Site 情報
- 【2】内容
- 【3】H/W セットアップ
- 【4】Watson「Text to Speech」、「Visual Recognition」サービスの作成
- 【5】Rasbian の最新化と Node.js、Node-Red のインストール
- 【6】コードのダウンロードとインストール
- 【7】ダウンロードしたコードを Node-Red ヘインポート
- 【8】「Text to Speech」、「Visual Recognition」、構成情報の設定
- 【9】写真イメージが表示されない。プログラムの実行前に修正する箇所
- 【10】プログラムの実行
 - 【11】日本語対応に変更
 - 【12】日本語対応版プログラムの実行

材料 (価格は変わる可能性があります。消費税、郵送料は含まれません)

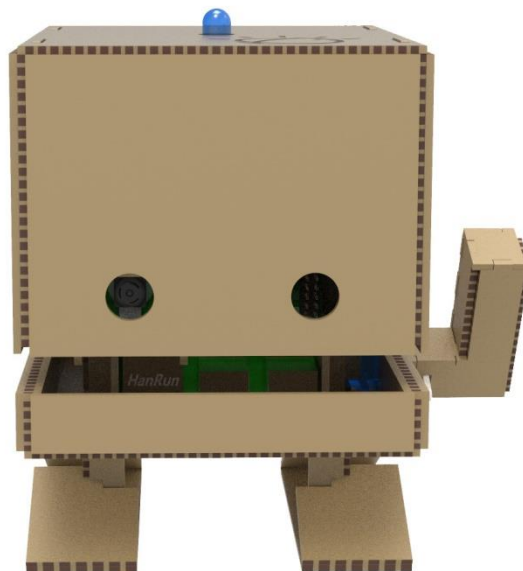
材料名	補足	価格 (消費税、送料含まず)	購入先例
厚紙	2mm厚	¥650	東急ハンズ
レーザーカット費用	1000円/10分	¥500	coromoza
Raspberry Pi3 Model B	Bluetooth、無線LAN含む。	¥5,600	秋月電子通商
電源 (Raspberry Pi3用)	スイッチングACアダプター5V2.5A AD-B50P250、USBケーブル Aオス-マイクロBオス 0.15m A-microB	¥1,210	秋月電子通商
microSD Card 8GB Class10	Transcend microSDHCカード 8GB Class10 (無期限保証)	¥1,200	アマゾン
Bluetoothスピーカー	Anker SoundCore mini コンパクト Bluetoothスピーカー	¥2,399	アマゾン
Raspberry Pi3 カメラ	サインスマート Raspberry Pi 用 カメラモジュール Camera Module for ラズベリーパイ	¥1,599	アマゾン
小計		¥13,158	
HDMIディスプレイ	Raspberry Pi3初期設定時のみ使用。テレビで代用		
HDMIケーブル	Raspberry Pi3初期設定時のみ使用。HDMI 1.4ケーブル	¥250	秋月電子通商
USBマウス	Raspberry Pi3初期設定時のみ使用。Logicool ロジクール 有線光学式3ボタンマウス M100r ブラック	¥475	アマゾン
USBキーボード	Raspberry Pi3初期設定時のみ使用。サンワサプライ USBキーボード(ブラック) SKB-L1UBK	¥664	アマゾン
小計		¥1,389	
合計		¥14,547	

【1】Web Site 情報

<https://github.com/samuelvogelmann/visualtj>

【2】内容

Watson、「Text to Speech」「Visual Recognition」と Raspberry Pi、Bluetooth スピーカー、Node-Red を使って、撮った写真について話します。




```
load-module module-always-sink

### Automatically suspend sinks/sources that become idle for too long
load-module module-suspend-on-idle

### Enable positioned event sounds
load-module module-position-event-sounds

### Automatically load driver modules for Bluetooth hardware
.ifexists module-bluetooth-policy.so
load-module module-bluetooth-policy
.endif

.ifexists module-bluetooth-discover.so
load-module module-bluetooth-discover
.endif

^G Get Help   ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify   ^W Where Is  ^V Next Page  ^U UnCut Text ^T To Spell
```

・pulseaudio サービスを起動

```
$ sudo systemctl start pulseaudio.service
```

・pulseaudio サービスを自動起動するよう設定

```
$ sudo systemctl enable pulseaudio.service
```

・root、pi ユーザが PulseAudio で音を出せるようにします

```
$ sudo gpasswd -a root pulse-access
```

```
$ sudo gpasswd -a pi pulse-access
```

```
pi@raspberrypi:~$ sudo systemctl start pulseaudio.service
pi@raspberrypi:~$ sudo systemctl enable pulseaudio.service
Created symlink from /etc/systemd/system/multi-user.target.wants/pulseaudio.service to /etc/systemd/system/pulseaudio.service.
pi@raspberrypi:~$ sudo gpasswd -a pi pulse-access
ユーザ pi をグループ pulse-access に追加
```

ここまでで PulseAudio 関連の設定が完了しましたので、次に bluetoothctl で Bluetooth デバイスを接続します。

スピーカーの電源をオン。

```
$ sudo bluetoothctl
```

```
[bluetooth]# power on
```

デバイスをスキャンし、スピーカーを探します。

```
[bluetooth]# scan on
```

Bluetoothスピーカーが見つかったら、スキャンを停止します。

```
[bluetooth]# scan off
```

```
pi@raspberrypi:~$ sudo bluetoothctl
[NEW] Controller B8:27:EB:C1:8F:AF raspberrypi [default]
[bluetooth]# scan on
Discovery started
[CHG] Controller B8:27:EB:C1:8F:AF Discovering: yes
[NEW] Device 00:E0:4C:58:6C:EC SoundCore mini
[NEW] Device CF:E9:B1:E0:D6:98 ID107 HR
[CHG] Device 00:E0:4C:58:6C:EC Class: 0x240404
[CHG] Device 00:E0:4C:58:6C:EC Icon: audio-card
[bluetooth]# scan off
[CHG] Device CF:E9:B1:E0:D6:98 RSSI is nil
[CHG] Device 00:E0:4C:58:6C:EC RSSI is nil
[CHG] Controller B8:27:EB:C1:8F:AF Discovering: no
Discovery stopped
[bluetooth]#
```

[NEW] Device 00:E0:4C:58:6C:EC SoundCore mini

で Bluetooth スピーカーのデバイス情報が入手できました。

```
[bluetooth]# trust 00:E0:4C:58:6C:EC
```

```
[bluetooth]# pair 00:E0:4C:58:6C:EC
```

```
[bluetooth]# connect 00:E0:4C:58:6C:EC
```

「ピロリン」という音が鳴ると、接続完了です。

```
[bluetooth]# exit
```

で `bluetoothctl` から抜けます。

・なかなか接続できない場合

<https://raspberrypi.stackexchange.com/questions/44497/having-an-issue-with-bluetooth-manager-pairing-on-the-pi>

Syslog を確認します。

```
$ cat /var/log/syslog
```

“bluetoothd[794]: a2dp-sink profile connect failed for 00:E0:4C:58:6C:EC: Protocol not available”というエラーの場合下記手順を試してください。

(1) PulseAudio Bluetooth ライブラリーがインストールされているか確認

```
$ sudo apt-get install pulseaudio-module-bluetooth
```

(2) Pulse Audio server が稼働しているか確認

まず停止します。

```
$ sudo killall pulseaudio
```

起動します。

```
$ pulseaudio --start
```

(3) bluetoothctl で connect します。

```
[bluetooth]# connect 00:E0:4C:58:6C:EC
```

それでもだめだったら、[スピーカのモード切替](#)を行う。

(2) カメラ

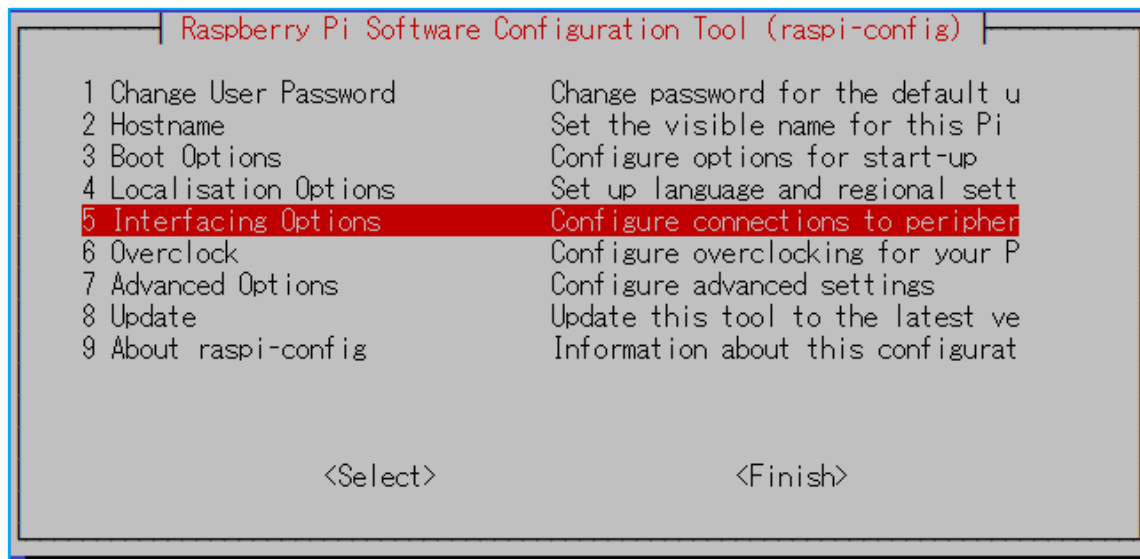
サインスマート Raspberry Pi 用 カメラモジュール Camera Module for ラズベリーパイを使用しました。



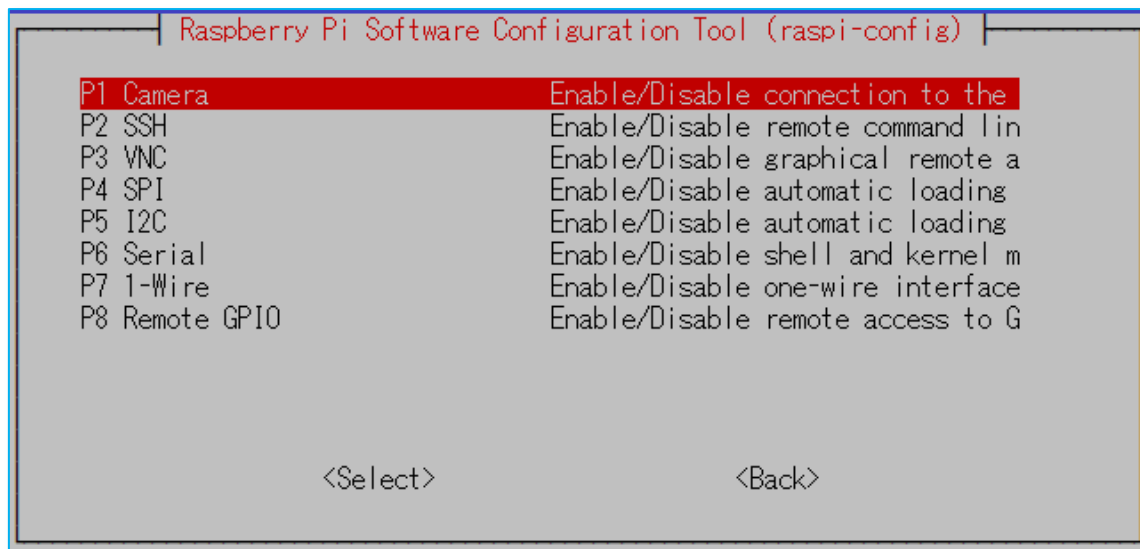
カメラが使えるように、Raspberry Pi を設定します。

```
$ sudo raspi-config
```

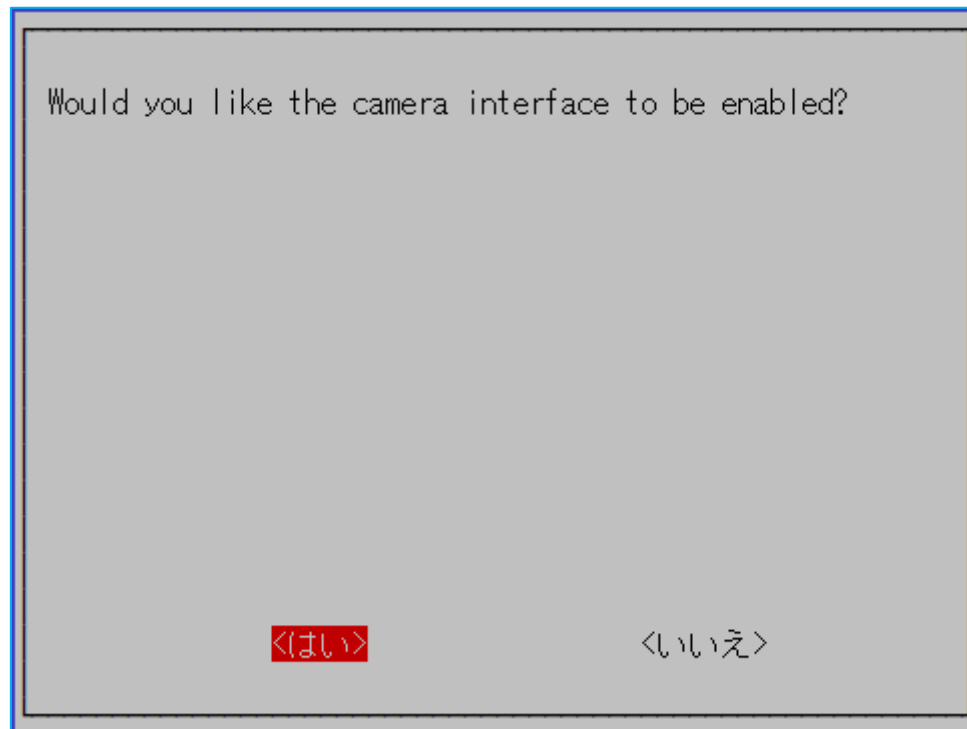
「5 Interfacing Options」選択



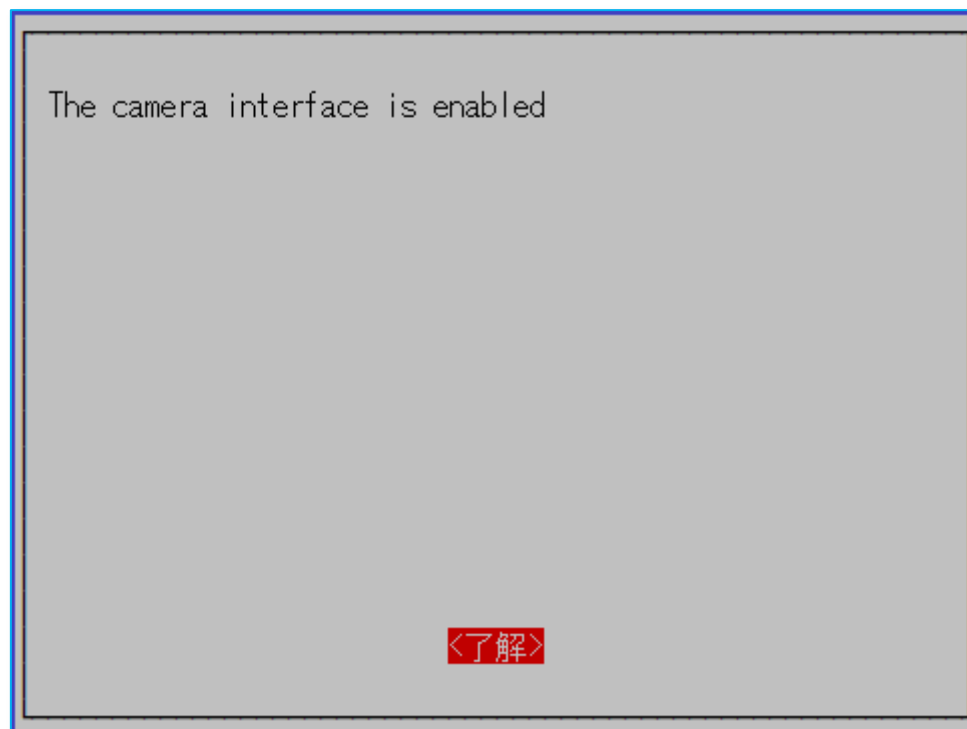
「P1 Camera」選択



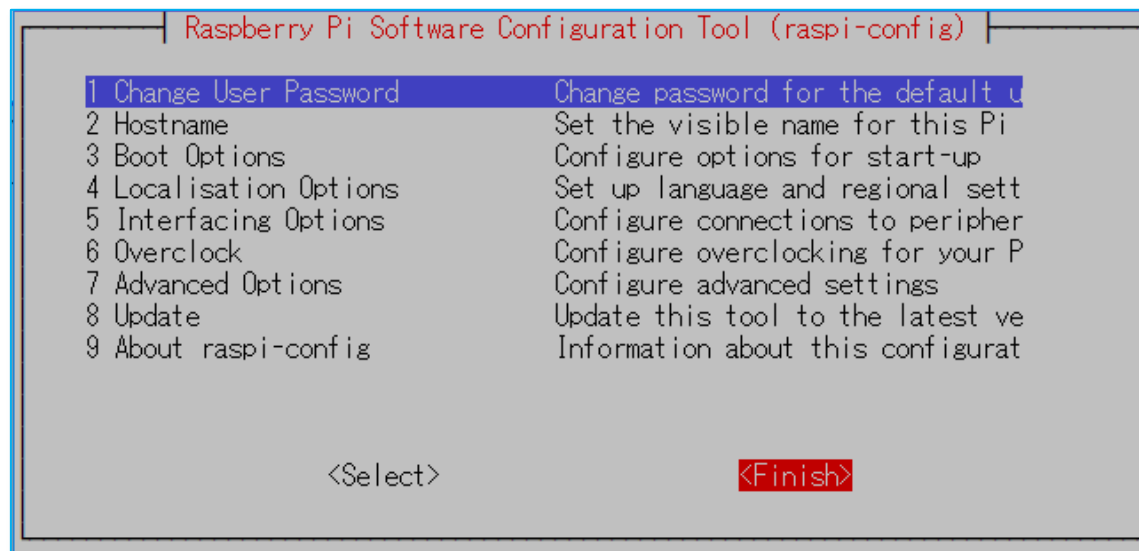
「はい」選択



「了解」選択



「Finish」選択



【4】Watson「Text to Speech」、「Visual Recognition」サービスの作成

Bluemix にログイン

<https://console.bluemix.net/>

Log into IBM Bluemix

IBMID またはメール・アドレスを入力してください
IBM ID をお忘れですか?

New to Bluemix? [Sign Up](#)



Log into IBM Bluemix

IBM ID:

パスワード パスワードをお忘れの場合

[別の IBM アカウントまたは E メールを使用してください](#)

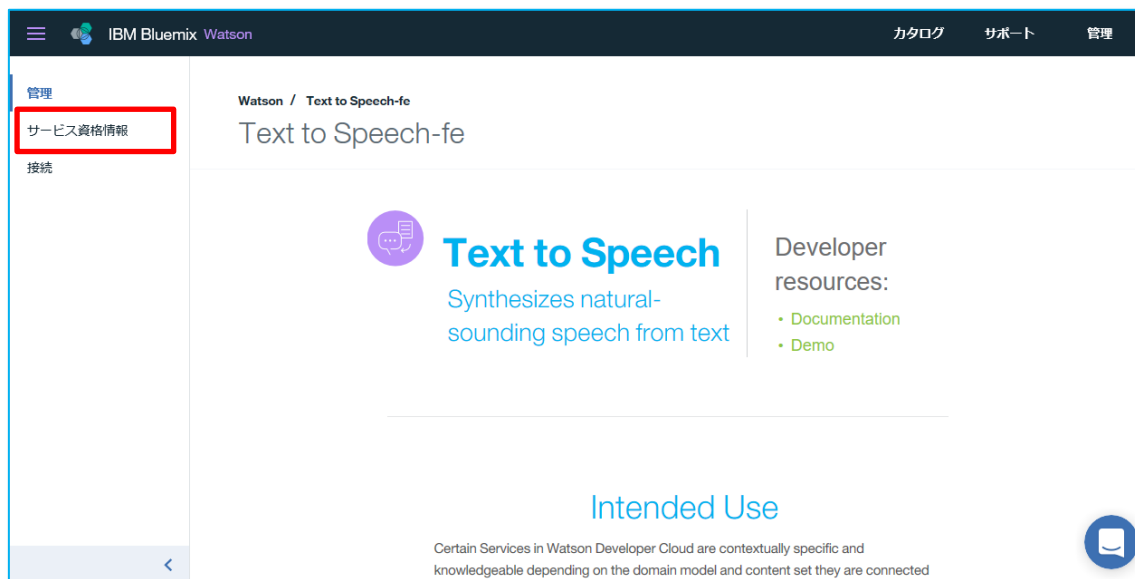
(1) カタログより「Text to Speech」を選択



「作成」をクリック



「サービス資格情報」をクリック



「資格情報の表示」をクリック



「作成」をクリック

IBM Bluemix カタログ

← すべて表示

Visual Recognition

画像コンテンツに含まれる意味を検出します。場面、対象物、顔のイメージ、およびその他のコンテンツを分析します。既製のデフォルト・モデルを選択するか、独自のカスタム種別を作成します。コレクションに含まれる類似のイメージを検出します。イメージやビデオ・フレームの画像コンテンツを解析し、何が起きている場面なのかを理解するためのスマート・アプリケーションを作成します。

サービス名:
Visual Recognition-5I

資格情報名:
Credentials-1

フィーチャー

- 一般種別
イメージを説明するクラス・キーワードを生成します。独自のイメージを使用するか、一般に公開されている Web ページから関連するイメージの URL を解析用に抽出します。
- 顔検出
イメージ内の人物の顔を検出します。また、このサービスでは、顔の一般的な年齢層と性別も示されます。
- 画像トレーニング
- 類似イメージ検索 (ベータ)

ヘルプが必要ですか?
Bluemix 営業担当へのお問い合わせ

月額費用の計算
費用計算

作成

「サービス資格情報」をクリック

IBM Bluemix Watson

管理

サービス資格情報

接続

Watson / Visual Recognition-I5

Visual Recognition-I5

Visual Recognition

Understand the contents of images. Create custom classifiers to develop smart applications. Create custom collections to search for similar images.

Visual Recognition Tool (Beta)

Developer resources:

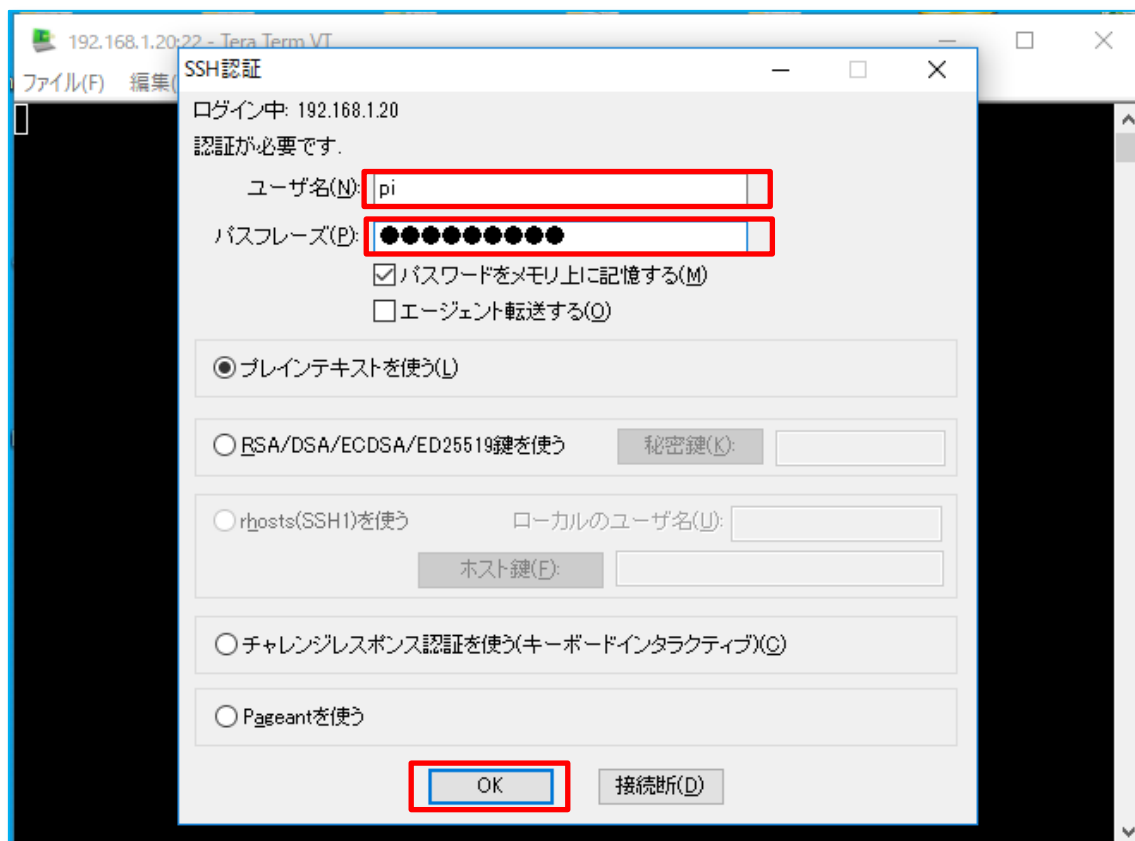
- Documentation
- Demo

【 5 】Rasbian の最新化と Node.js、Node-Red のインストール

Teraterm などでも SSH 接続します。ID、パスワードの初期値は、

ID:pi

Password: raspberry です。



以下のコマンドを実行してください。

```
$ sudo apt-get update  
$ sudo apt-get dist-upgrade  
$ curl sL https://deb.nodesource.com/setup_6.x | sudo -E bash -  
$ sudo apt-get install -y nodejs  
$ sudo npm install -g node-red
```

一度 Node-Red を起動してから停止します。

```
$ node-red
```

Ctl + c で停止します。

Watson Node-Red サービスをインストールします。

```
$ cd ~/.node-red  
$ npm install node-red-node-watson
```

Node-Red を起動します。

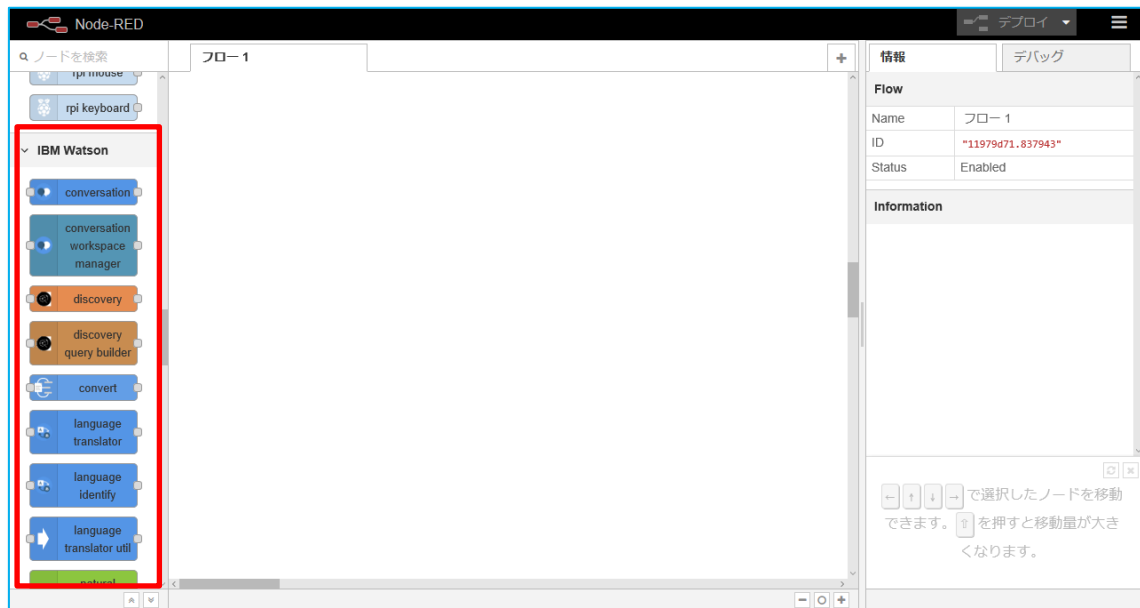
```
$ node-red
```

```
pi@raspberrypi: ~/.node-red $ node-red  
24 Jul 21:31:46 - [info]  
  
Welcome to Node-RED  
=====  
  
24 Jul 21:31:46 - [info] Node-RED version: v0.17.5  
24 Jul 21:31:46 - [info] Node.js version: v6.11.1  
24 Jul 21:31:46 - [info] Linux 4.9.35-v7+ arm LE  
24 Jul 21:31:47 - [info] Loading palette nodes  
24 Jul 21:31:55 - [info] Settings file : /home/pi/.node-red/settings.js  
24 Jul 21:31:55 - [info] User directory : /home/pi/.node-red  
24 Jul 21:31:55 - [info] Flows file : /home/pi/.node-red/flows_raspberrypi.j  
son  
24 Jul 21:31:55 - [info] Creating new flow file  
24 Jul 21:31:55 - [info] Starting flows  
24 Jul 21:31:55 - [info] Started flows  
24 Jul 21:31:55 - [info] Server now running at http://127.0.0.1:1880/
```

終了するのは、**Ctrl+c** です。

ブラウザを使って、**IP アドレス:1880** でアクセスします。

Watson ノードも登録されています。



【6】コードのダウンロードとインストール

```
$ git clone https://github.com/samueltvogelmann/visuالتj.git
```

```
$ cd ~/visuالتj
```

```
pi@raspberrypi:~/visuالتj $ ls -al
合計 56
drwxr-xr-x  4 pi pi  4096  7月 24 21:51 .
drwxr-xr-x 22 pi pi  4096  7月 24 21:51 ..
drwxr-xr-x  8 pi pi  4096  7月 24 21:51 .git
-rw-r--r--  1 pi pi  1073  7月 24 21:51 LICENSE
-rw-r--r--  1 pi pi  6032  7月 24 21:51 README.md
-rw-r--r--  1 pi pi 25294  7月 24 21:51 flow.json
drwxr-xr-x  2 pi pi  4096  7月 24 21:51 images
```

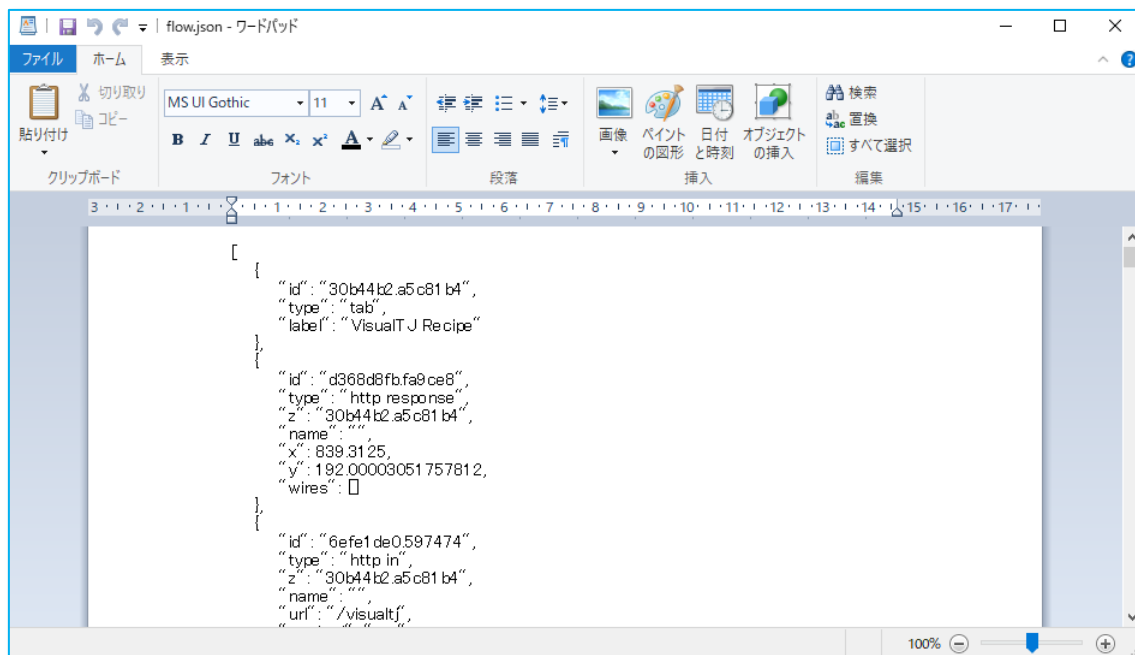
「flow.json」がダウンロードした Node-Red のコードです。

【7】ダウンロードしたコードを Node-Red ヘインポート

インポートはクリップボードを使用します。

Winscp などを使って、「flow.json」をクライアント PC へ転送します。

Wordpad で「flow.json」を開き、**Ctrl+a** ⇒ **Ctrl+c** でクリップボードへコピーします。



Raspberry 側で Node-Red が停止している場合、Node-Red 起動しておきます。

```
$ node-red
```

```
pi@raspberrypi: ~/visualtj $ node-red
24 Jul 22:10:23 - [info]

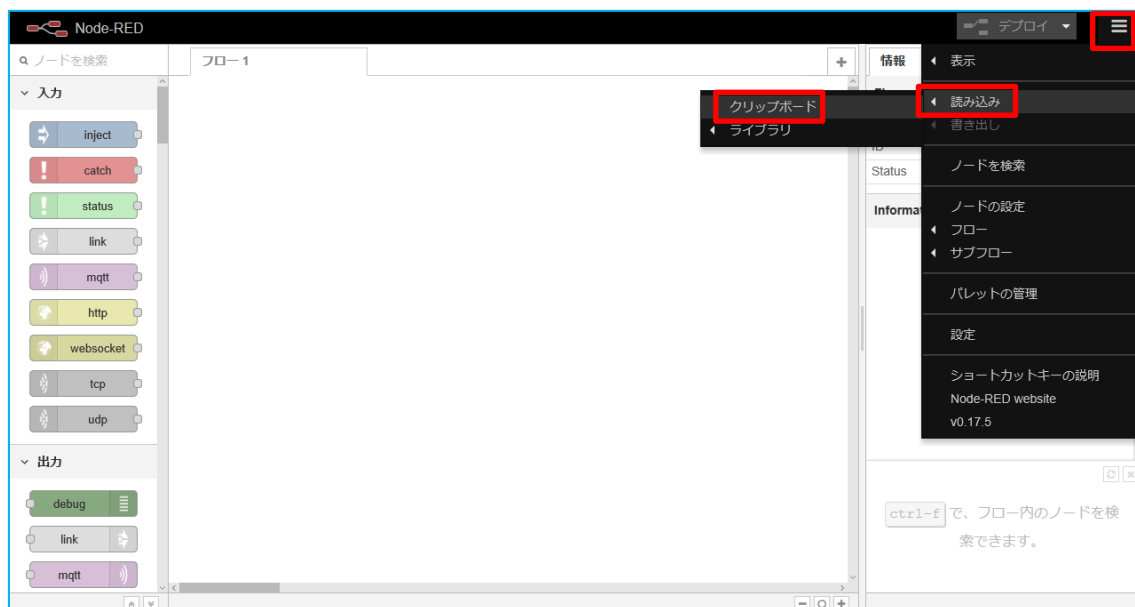
Welcome to Node-RED
=====

24 Jul 22:10:23 - [info] Node-RED version: v0.17.5
24 Jul 22:10:23 - [info] Node.js version: v6.11.1
24 Jul 22:10:23 - [info] Linux 4.9.35-v7+ arm LE
24 Jul 22:10:25 - [info] Loading palette nodes
24 Jul 22:10:30 - [info] Settings file : /home/pi/.node-red/settings.js
24 Jul 22:10:30 - [info] User directory : /home/pi/.node-red
24 Jul 22:10:30 - [info] Flows file : /home/pi/.node-red/flows_raspberrypi.json
24 Jul 22:10:30 - [info] Creating new flow file
24 Jul 22:10:30 - [info] Starting flows
24 Jul 22:10:30 - [info] Started flows
24 Jul 22:10:30 - [info] Server now running at http://127.0.0.1:1880/
```

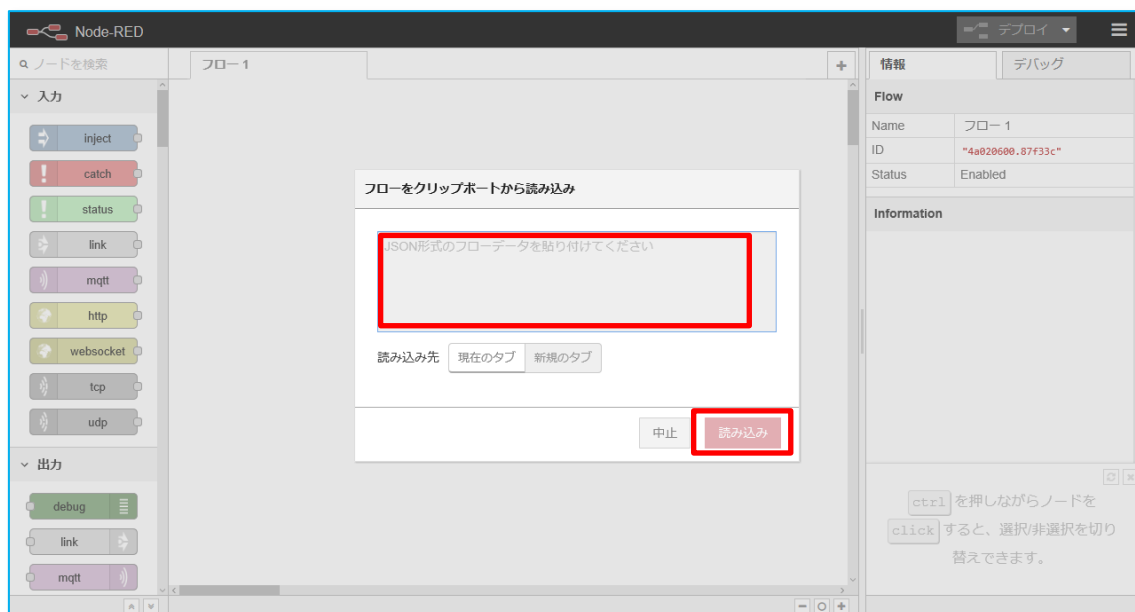
クライアント PC 側のブラウザで Node-Red にアクセス。

<http://RaspberryPIのIPアドレス:1880>

右上の☰「ハンバーガーメニュー」⇒「読み込み」⇒「クリップボード」



貼り付けて、「読み込み」



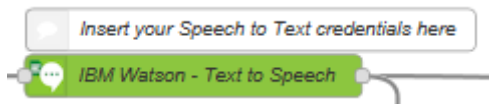
「VisualTJ Recipe」が読み込まれました。

The screenshot shows the Node-RED web interface. The main workspace displays a flow titled "VisualTJ Recipe", which is highlighted with a red rectangular box. The flow starts with a "WebApp" node connected to a "VisualTJ" node. It then branches into two paths: one for image analysis (including "Take or analyze picture", "img upload", "Create image checker", "Take and save the image to the archive folder", "Take a picture", "Picture status", and "img download") and another for speech-to-text conversion (including "Take or analyse picture", "Add Custom Classifier", "insert your VoIP API key here", "MSI Watson - Visual Recognition", "Analyze the image", "select classification", and "switch"). The right sidebar shows the "Flow" information panel with details for "VisualTJ Recipe" (ID: "fd4681e4-c9703", Status: "Enabled") and a "ctrl-f" search tip.

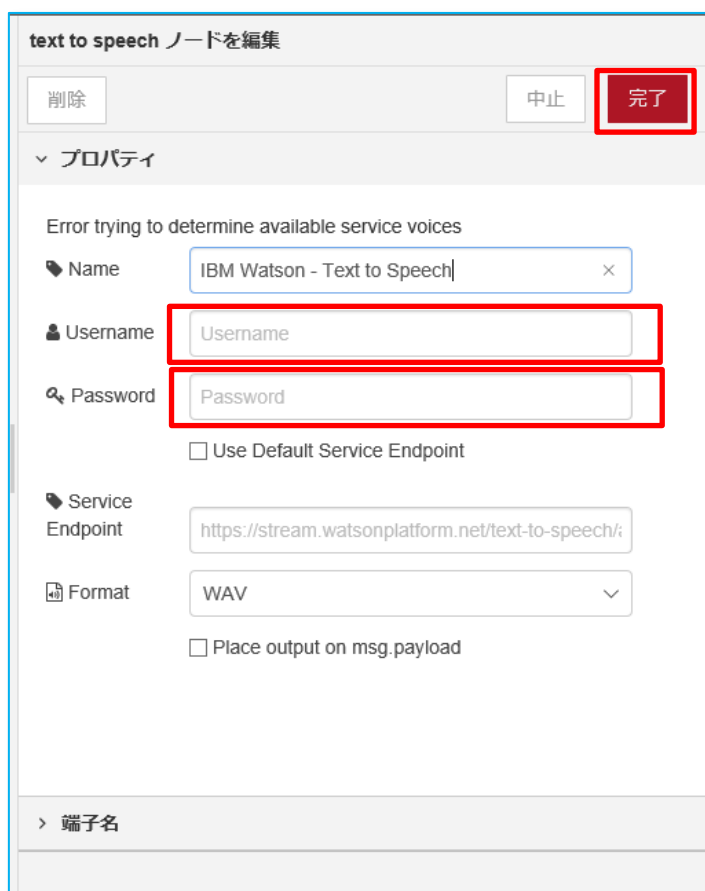
【8】「Text to Speech」、「Visual Recognition」、構成情報の設定

(1) 「Text to Speech」

「Text to Speech」ノード（緑色）をダブルクリック



「username」「password」を入力後、「完了」ボタンを押す。



text to speech ノードを編集

削除 中止 完了

▼ プロパティ

Error trying to determine available service voices

Name IBM Watson - Text to Speech

Username Username

Password Password

Use Default Service Endpoint

Service Endpoint https://stream.watsonplatform.net/text-to-speech/

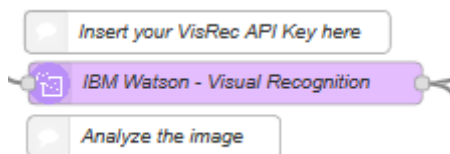
Format WAV

Place output on msg.payload

> 端子名

(2) 「Visual Recognition」

「Text to Speech」ノード（紫色）をダブルクリック



「API Key」を入力後、「完了」ボタンを押す。

The image shows a dialog box titled "visual recognition ノードを編集". At the top, there are three buttons: "削除" (Delete), "中止" (Cancel), and "完了" (Complete). The "完了" button is highlighted with a red box. Below the buttons is a section titled "プロパティ" (Properties) with a downward arrow. It contains four rows of configuration options:

- API Key:** A text input field containing "API Key", highlighted with a red box.
- Detect:** A dropdown menu with "Classify an image" selected.
- Name:** A text input field containing "IBM Watson - Visual Recognition".
- Language:** A dropdown menu with "English" selected.

At the bottom of the dialog, there is a section titled "端子名" (Terminal Name) with a rightward arrow.

The screenshot displays the Node-RED web interface. The main workspace shows a flow titled "VisualTJ Recipe" with several nodes connected. The flow starts with an "inject" node, followed by "http" and "websocket" nodes. It then branches into two paths: one for image processing (using "Create image directory", "Take a picture", "Picture status", and "img upload") and another for text-to-speech (using "insert your speech to text credentials here", "RMF Watson - Text to Speech", and "img upload"). The image path includes a "select classification" node and a "switch" node. The right sidebar shows the "Node" information for a "websocket out" node, including its ID and a description of its behavior.

Node Information:

Node	
型	websocket out
ID	"11Fd84Fd_b02ceb"
show more ▾	
Information	
WebSocket out node.	
By default, <code>msg.payload</code> will be sent over the WebSocket. The socket can be configured to encode the entire <code>msg</code> object as a JSON string and send that over the WebSocket.	
If the message arriving at this node started at a WebSocket in node, the message will be sent back to the client that triggered the flow. Otherwise, the message will be broadcast to all connected clients.	
If you want to broadcast a message that	

`ctrl-space` で、サイドバーの表示非表示の切り替えができます。

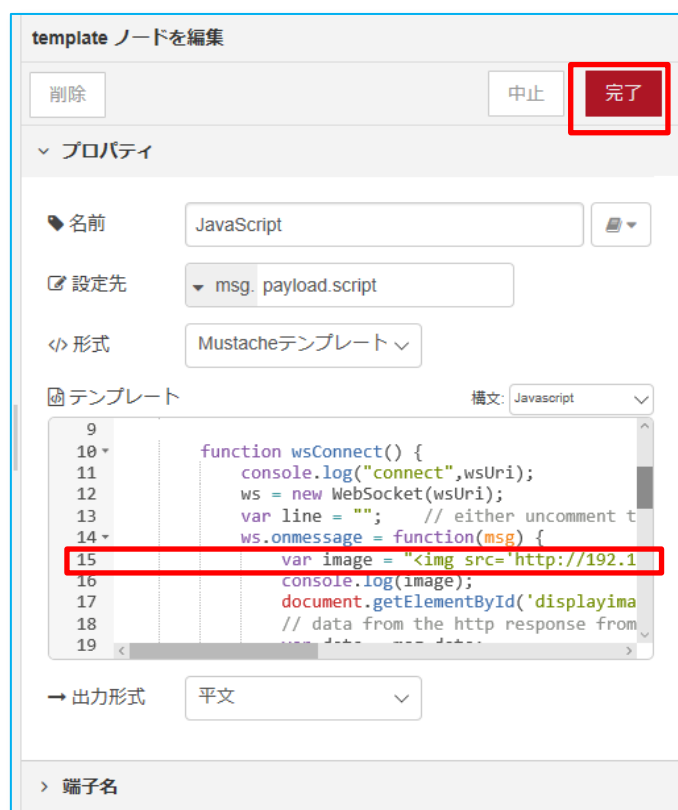
【9】写真イメージが表示されないときに、修正する箇所

Node-Redが実行されている、Raspberryのブラウザでプログラムを実行するには問題ないですが、別のPCのブラウザで実行すると撮影した写真イメージが表示されません。

これは、イメージの保存場所指定のURLがlocalhostになっているためです。

ここでは、RaspberryのIPアドレスを「192.169.1.20」としてプログラムを修正します。

「JavaScript」ノードの修正。



15行目の「localhost」を「192.168.1.20」に変更します。

```
var image = "<img src='http://localhost:1880/getimage?'+ new Date().getTime()  
+ "' alt='Picture' width='480px'/>";
```

⇒

```
var image = "<img src='http://192.168.1.20:1880/getimage?'+ new  
Date().getTime() + "' alt='Picture' width='480px'/>";
```

「完了」ボタンを押す。

【10】プログラムの実行

Node-Red が起動していない場合は、「\$ node-red」で起動しておきます。

```
pi@raspberrypi:~$ node-red
25 Jul 18:48:19 - [info]

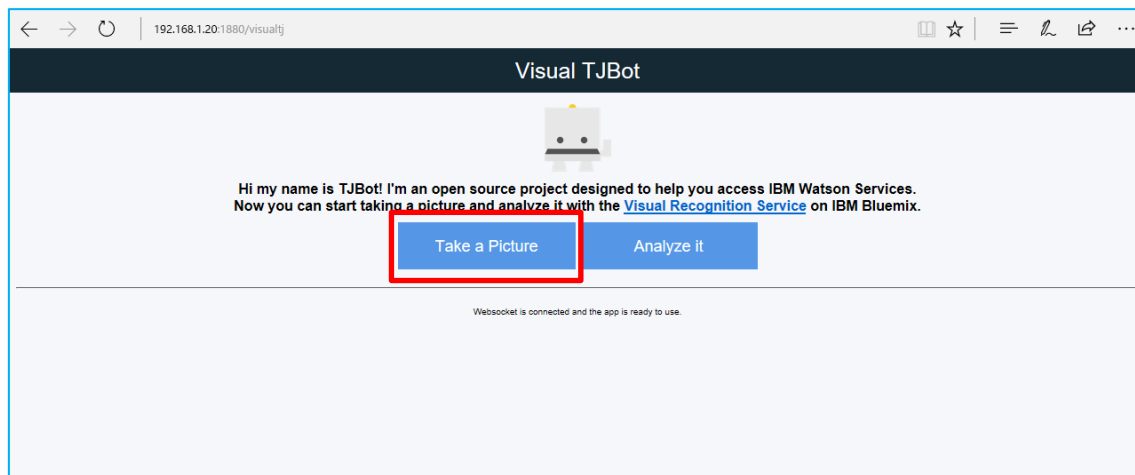
Welcome to Node-RED
=====

25 Jul 18:48:19 - [info] Node-RED version: v0.17.5
25 Jul 18:48:19 - [info] Node.js version: v6.11.1
25 Jul 18:48:19 - [info] Linux 4.9.35-v7+ arm LE
25 Jul 18:48:20 - [info] Loading palette nodes
25 Jul 18:48:26 - [info] Settings file : /home/pi/.node-red/settings.js
25 Jul 18:48:26 - [info] User directory : /home/pi/.node-red
25 Jul 18:48:26 - [info] Flows file    : /home/pi/.node-red/flows_raspberrypi.j
son
25 Jul 18:48:26 - [info] Server now running at http://127.0.0.1:1880/
25 Jul 18:48:26 - [info] Starting flows
25 Jul 18:48:28 - [info] Started flows
```

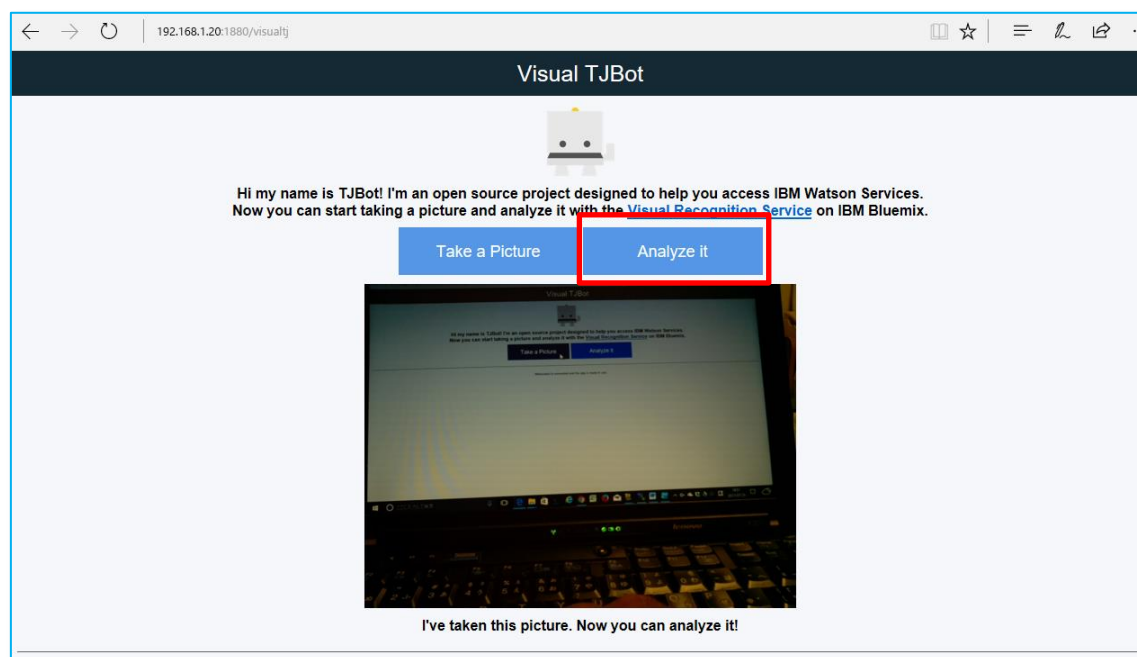
PC のブラウザからアクセス

<http://192.168.1.20:1880/visualtj>

ここでは、Rasberry の IP アドレスを「192.168.1.20」としました。




「Take a Picture」ボタンをクリックして、写真を撮ります。



「Analyze it」ボタンをクリック、撮影した写真イメージが何かを分析します。

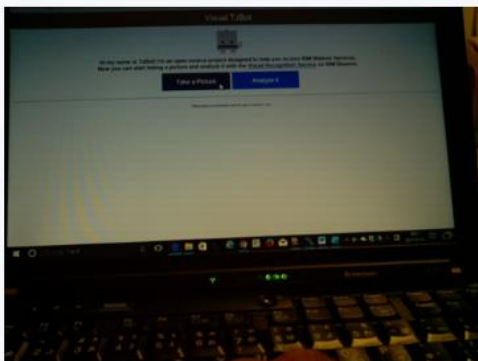
192.168.1.20:1880/visualtj

Visual TJBOT



Hi my name is TJBOT! I'm an open source project designed to help you access IBM Watson Services. Now you can start taking a picture and analyze it with the [Visual Recognition Service](#) on IBM Bluemix.

Take a Picture Analyze it



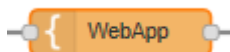
Here are my results:

default	
Class	Score
computer	0.855
machine	0.855
computer screen	0.664
video display	0.673

分析結果を表示と音声で答えます。

【 1 1 】日本語対応に変更

(1) 「WebApp」ノード



・14 行目

```
<p><b>Hi my name is TJBot! I'm an open source project designed to help you  
access IBM Watson Services.</b></p>
```

⇒

```
<p><b>ハイ！ TJBot です。私は IBM Watson Services へのアクセスを支援するように設計され  
たオープンソース・プロジェクトです.</b></p>
```

・15 行目

```
<p><b>Now you can start taking a picture and analyze it with the <a  
target="_blank"  
href="http://www.ibm.com/watson/developercloud/visual-recognition.html">Visu  
al Recognition Service</a> on IBM Bluemix.</b></p>
```

⇒

```
<p><b>さあ！ <a target="_blank"  
href="http://www.ibm.com/watson/developercloud/visual-recognition.html">Visu  
al Recognition Service</a> on IBM Bluemix.で写真を撮って分析しましょう！ </b></p>
```

・19 行目

```
<div id="divbutton"><button type="button" onclick='takepic("take  
picture");'>Take a Picture</button></div>
```

⇒

```
<div id="divbutton"><button type="button" onclick='takepic("take picture");'>写  
真を撮る</button></div>
```

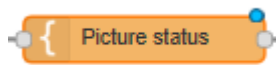
・20 行目

```
<div id="divbutton"><button type="button" onclick='analyze("analyze  
picture");'>Analyze it</button></div>
```

⇒

```
<div id="divbutton"><button type="button" onclick='analyze("analyze  
picture");'>写真を分析</button></div>
```

(2) 「Picture status」ノード



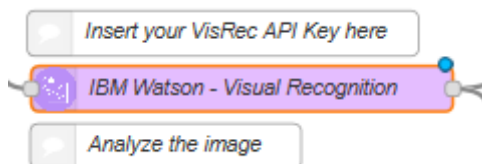
・1行目

<p>I've taken this picture. Now you can analyze it!</p>

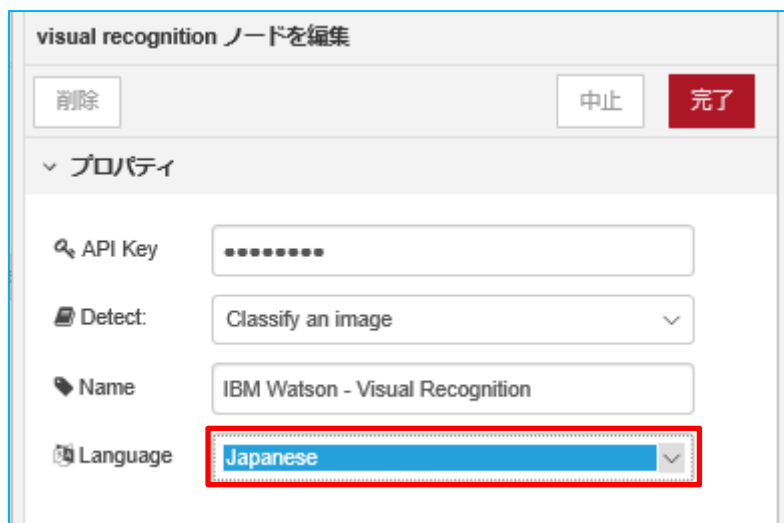
⇒

<p>写真を撮りました。分析できますよ！</p>

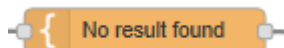
(3) 「Visual Recognition」ノード



・“Language”を「English」から「Japanese」へ変更



(4) 「No result found」ノード



<p>I don't know what this is!</p>

⇒

<p>なんだか？わかりません！</p>

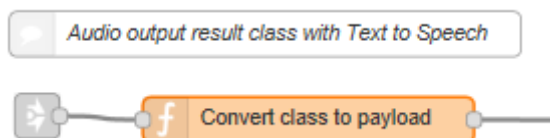
(5) 「Convert class to payload」ノード

・1行目

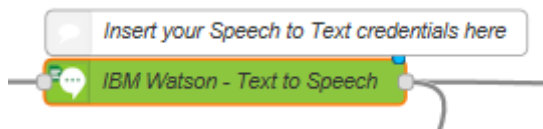
```
msg.payload="This is a "+msg.class;
```

⇒

```
msg.payload="これは"+msg.class;
```



(6) 「Text to Speech」ノード



- ・“Language” を「US English」から「Japanese」へ変更
- ・“Voice”を「Michaeli」から「Emi」へ変更

text to speech ノードを編集

削除 中止 完了

▼ プロパティ

Name IBM Watson - Text to Speech

Username ed2075ac-5d96-4362-8a19-9c435151e630

Password

Use Default Service Endpoint

Service Endpoint https://stream.watsonplatform.net/text-to-speech/

Language Japanese

Voice Emi

Format WAV

Place output on msg.payload

【 1 2 】日本語対応版プログラムの実行

Node-Red が起動している状態で、PC のブラウザから

<http://192.168.1.20:1880/visualtj>

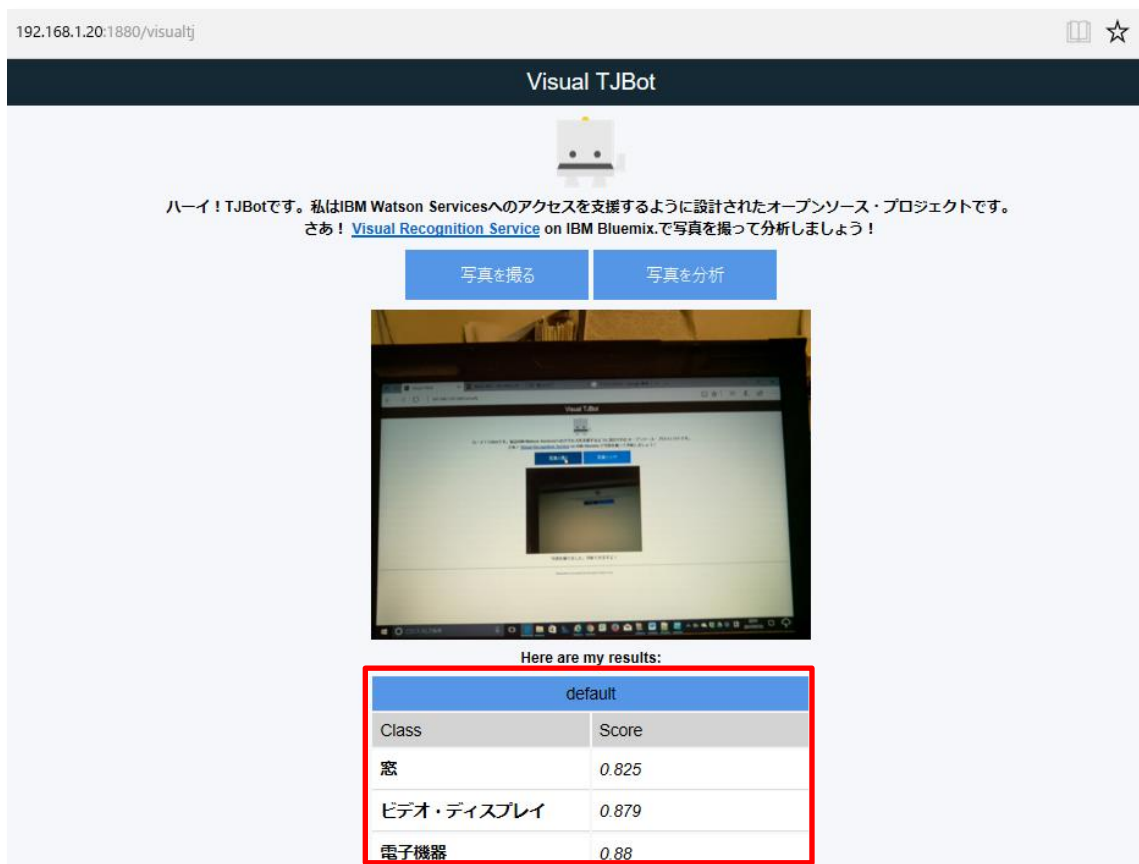
にアクセスします。

「写真を撮る」をクリック



「写真を分析」をクリック



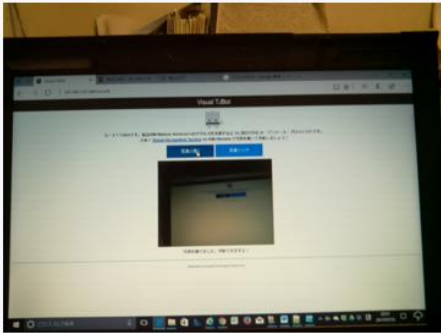


192.168.1.20:1880/visualtj

Visual TJBot

ハイ！TJBotです。私はIBM Watson Servicesへのアクセスを支援するように設計されたオープンソース・プロジェクトです。
さあ！ [Visual Recognition Service](#) on IBM Bluemix.で写真を撮って分析しましょう！

写真を撮る 写真を分析



Here are my results:

default	
Class	Score
窓	0.825
ビデオ・ディスプレイ	0.879
電子機器	0.88

分析結果を日本語表示と音声で答えます。