

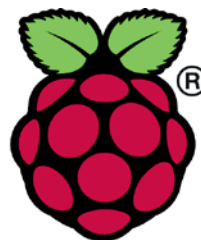
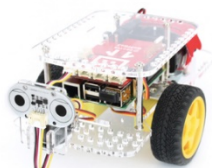
はじめてのロボット・カー開発キット

～GoPiGo3を使い、自動走行から音声制御による走行まで～

実践編（抜粋版）

GoPiGo

Build & program your own robot.



Raspberry Pi

 Google Cloud Platform



amazon alexa

スペクトラム・テクノロジー株式会社

<https://spectrum-tech.co.jp>

sales@spectrum-tech.co.jp

ロボット・カー開発キット 目次

Pi運用マニュアル

- ① RaspberryPiについて
- ② Linux基本コマンド
- ③ 基本操作
- ④ 日常運用(ウイルススキャン、更新)

GoPiGo基本操作

- ① GoPiGo概要
- ② GoPiGoソフト一覧
- ③ GoPiGoソフト開発ツール
- ④ GoPiGo起動
- ⑤ GoPiGoソフト更新
- ⑥ GoPiGoテスト&デモ
- ⑦ GoPiGoコントロールパネル
- ⑧ GoPiGo各種センサ試験
 - A) DIセンサ
 - B) カメラ(写真)
 - C) カメラ(ビデオ)
 - D) カメラ(リアルタイム動画)

Scratch学習

- ① Scratch概要
- ② Scratchプログラム学習
 - 基本操作
 - ライト点灯事例
 - 基本動作事例
 - コントロールパネル事例
 - 正方形動作事例

ページ

[5](#)
[6](#)
[7](#)
[8](#)

ページ

[11](#)
[12](#)
[13](#)
[14](#)
[15](#)
[16](#)
[17](#)

[18](#)
[19](#)
[20](#)
[21](#)

ページ

[23](#)
[24](#)
[24](#)
[26](#)
[27](#)
[28](#)
[29](#)

抜粋版のため、ページと
実際は違います

ロボット・カー開発キット 目次

画像認識開発

ページ

① Opencv

A) 顔認識(カメラ)	31
B) 顔検出(カメラ)	32
C) 顔検出2(カメラ)	33
D) 物体検出(カメラ)	34
E) 物体検出(写真)	35

② Yolo

A) 物体検出(写真)	36
B) 物体検出(ビデオ)	38

抜粋版のため、ページと
実際は違います

応用例

① キーボードコントロール	40
② マウスコントロール	41
③ 障害物検知自走	42
④ 障害物回避自走	43
⑤ Webカメラ配信	44
⑥ 自動走行(レーン追尾)	45
⑦ 火星探索	46

GCP開発

① GCP概要	48
② GCP設定	49
③ GCP開発	58
• 全体像	58
• プログラム一覧	59
• Cloud vision	60
• 文字認識	
• 顔検出	
• ラベル検出	
• ロゴ検出	

ロボット・カー開発キット 目次

Alexa開発	ページ
① 全体構成	75
② メニュー	76
③ AWS設定	78
④ Amazon. co. jp設定	85
⑤ Amazon developer設定	87
⑥ Alexa Voice Service開発	
• 概要、用途	88
• AVS製品設定	89
• マイク、スピーカ設定	94
• AVS sample APP設定	98
• AVS sample APP動作	99
• AVS sample APP試験	102
• AVS sample APPデバッグ	103
• デバイス確認	104
• 音声履歴	105
• 音声によるGoPiGo操作	106

抜粋版のため、ページと
実際は違います



Pi運用マニュアル

① Raspberry Piについて

既に全世界で1000万台以上販売された手のひらサイズのコンピュータです。LinuxベースのRaspbianOSで動作しております。今回、GoPiGoのCPUとして使用しています。

② Linux基本コマンド

A) システム関係

- 起動: 電源を入れると自動で起動します。
- 再起動: `# reboot`
又は、アプリケーション>ログアウト>再起動; 左上のメニューから
- 終了: `# shutdown`
又は、アプリケーション>ログアウト>シャットダウン; 左上のメニューから
- ログアウト `# logout`
又は、アプリケーション>ログアウト>ログアウト; 左上のメニューから
- **日本語／英語の入力切替**: キーボードの`ctl+j`を同時に押します。又は右上のアイコン(右から7個目)からプルダウンで選択



Raspberry Pi

Pi運用マニュアル

② Linux基本コマンド

B) ディレクトリ操作、コピー、移動、削除

root@:~# **cd** /root/Documents ディレクトリの切り替え
 root@:/root/Documents# **ls** ファイルとディレクトリの表示(表示したら操作したいファイルを右クリックでコピーして操作します)
 root@:~# **cp** ファイル名 ディレクトリ 配下のディレクトリのファイルを別のディレクトリへコピー
 root@:~# **mv** ファイル名 ディレクトリ 配下のディレクトリのファイルを別のディレクトリへ移動
 root@:~# **rm** ファイル名 ファイルの削除
 便利な機能 **rm -help** コマンドのオプションが分からない場合は、ヘルプで問い合わせる。すべてのコマンド共通(マイナスを2個とhelp)

C) ユーザ権限、プロセス他

root@:~ \$ **su -** スーパーユーザ(root)に切り替え、パスワードを入力
 root@:~# **ps a** 現状の動いているプロセスを表示
 root@:~# **kill** 特定のプロセスを強制終了
 root@:~# **apt-get install pkg** パッケージのインストールなどに使用
 root@:~# **date** 日付、時間の設定を行います。
 root@:~# **leafpad** /etc/network/interfaces インタフェースに記述している内容を変更します。Viよりも使いやすいです。

D) モジュール、usb、メモリ、HDDなどの表示

root@:~# **lsmod** linuxのモジュールリスト表示
 root@:~# **lsusb** usbのデバイス表示
 root@:~# **free -mt** メモリ使用状態表示
 root@:~# **df -h** HDD(マイクロSD)の使用状態表示



Raspberry Pi

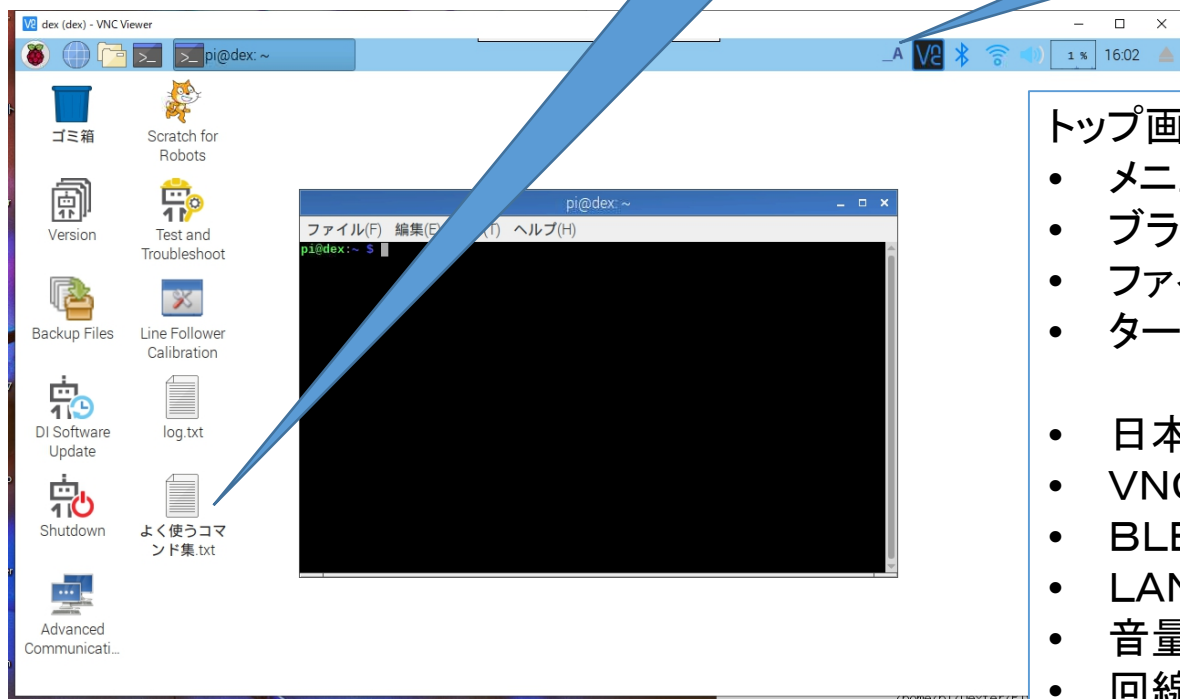
Pi運用マニュアル

③ 基本操作

A) 表示画面と内容

トップ画面によく使うコマンド.txtをアップしてます

Anthonyが出ない場合は、一度 Japaneseを選択後、再度 Anthonyを選択してください。



トップ画面(上段のタスクバーで選択)

- メニュー
- ブラウザ
- ファイルマネージャ
- ターミナル
- 日本語入力
- VNC
- BLE
- LAN/WiFi
- 音量
- 回線効率
- 時刻

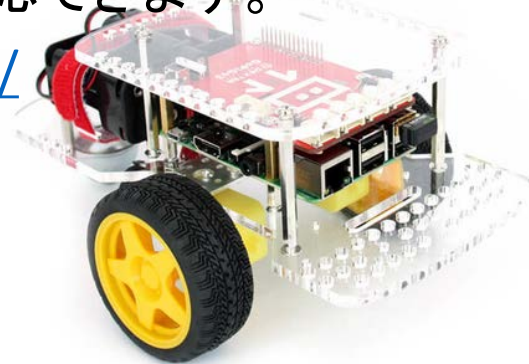
GoPiGo基本操作



GoPiGo基本操作

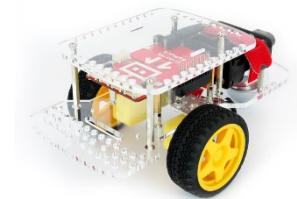
①. GoPiGo 概要

- GoPiGoは、Dexter社が、小学生から大学・社会人まで幅ひろく、ロボットを使ってプログラム習得することを目的に開発されました。既に、数多く導入され活用されています。火星探査プロジェクトでも有名です。
- 特徴として、カメラ、距離センサ、Lineセンサ、温湿度センサなど数多くのセンサがあり、そのプログラムも公開されています。
- プログラム言語もScratch, Python, Java、Cなど準備されており、OSとしては、Raspbian, DEX OSなど利用用途により使い分けできます。初心者から上級者まで幅広く対応できます。
- <https://www.dexterindustries.com/gopigo3/>



GoPiGo基本操作

②. GoPiGo ソフトウェア一覧



本開発キットのインストールしているソフトウェアの概要です。

区分	ソフト名	バージョン	備考
OS	dexter	v9	Raspbian stretch version
プログラム言語	python3	3.5.3	
	python2	2.7.13	
	scratch	1.4	
画像	Opencv	3.4	
	picamera	1.13	
	yolo	yolov3	ローカルで動作
	google-cloud-vision		グーグル・クラウド
音声 (Alexa)	AVS(alex voice service) device sdk	V1.13	
その他	Pkg, Pip関係プログラム多数		

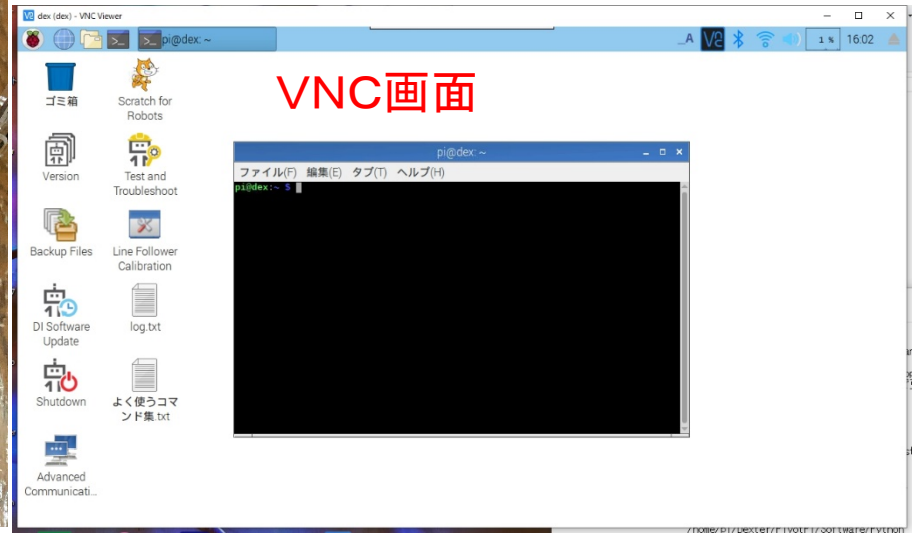
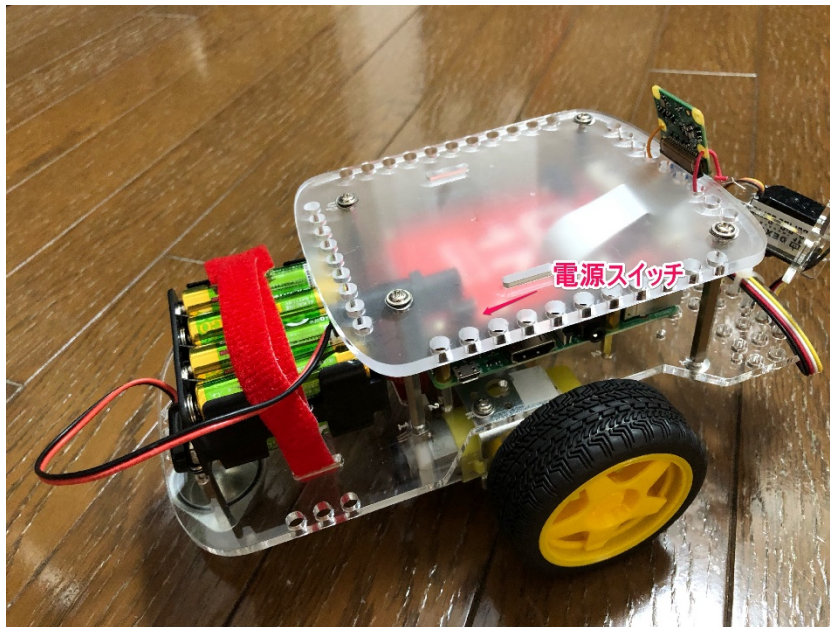
GoPiGo基本操作

④. GoPiGo起動



- GoPiGoの基本部、カメラなどの組み立て、取付が完了しましたら、電源をONにして起動します。
- Teraterm, VNC ViewerでPC等からアクセスします。

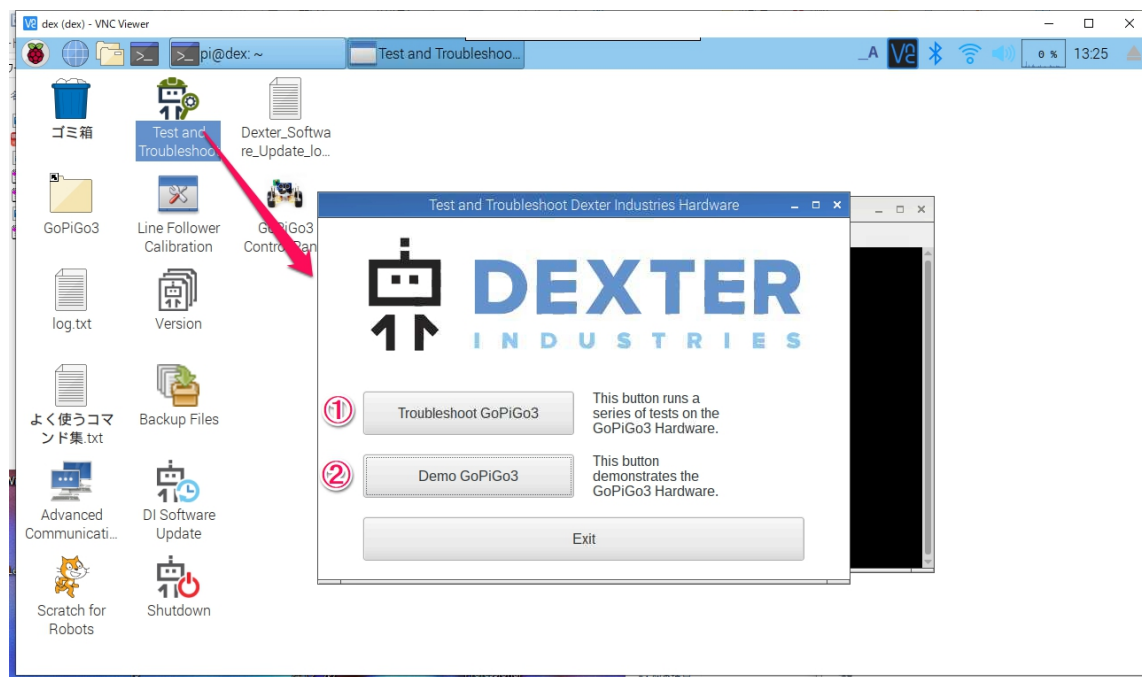
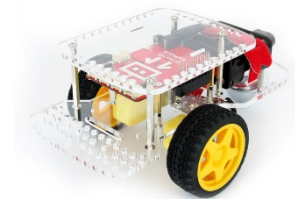
Teraterm
IPアドレス:192.168.1.xx
ユーザ名:pi
パスフレーズ:robots1234
VNC接続
パスワード:設定したもの

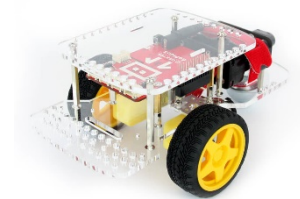


GoPiGo基本操作

⑥. GoPiGo3 テスト&デモ

- GoPiGo3のテストとデモを行います。
- ① ハードウェアのテストを行います。
- ② デモを行います。動き出しますので、注意して実施してください。



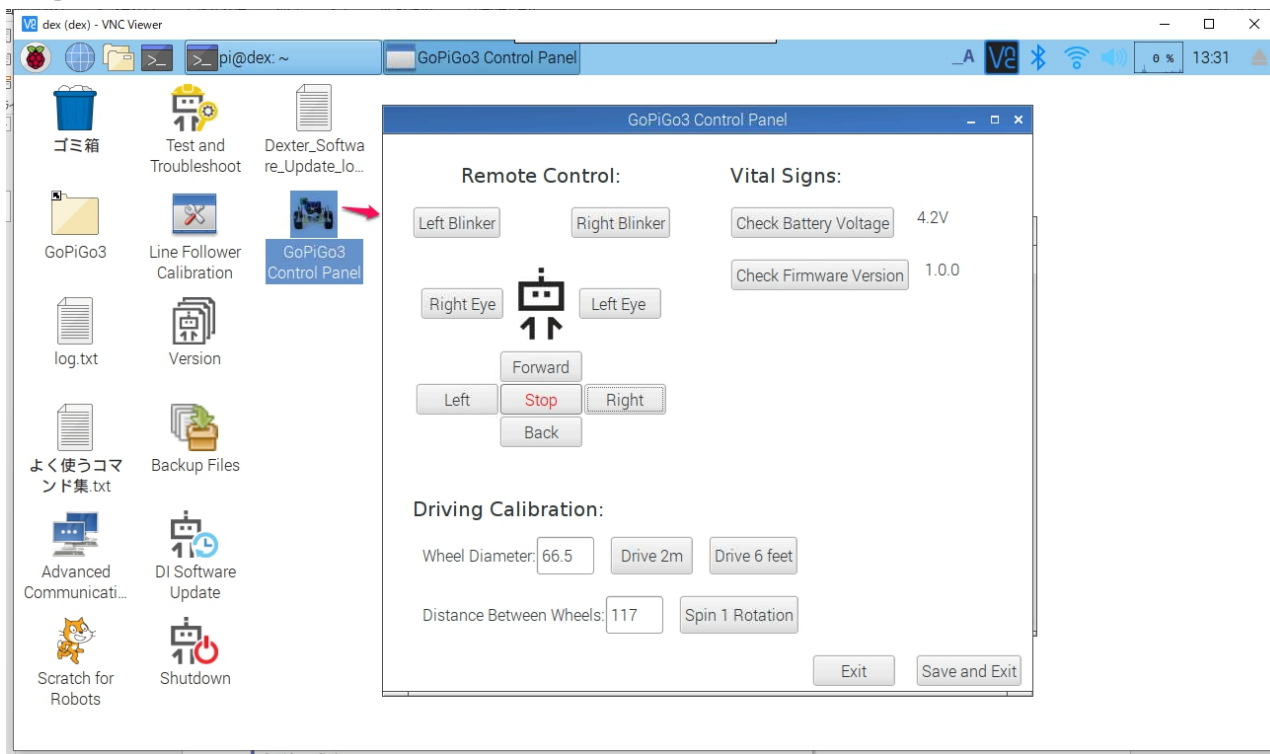


GoPiGo基本操作

⑦. GoPiGo3 コントロールパネル

• GoPiGo3のコントロールパネル。

- ① LEDライトの点灯。
- ② ロボットの動作(前進、後退、左右)
- ③ 運転補正、ファームウェアなどの確認など



GoPiGo基本操作

⑧. GoPiGo 各種センサ類試験

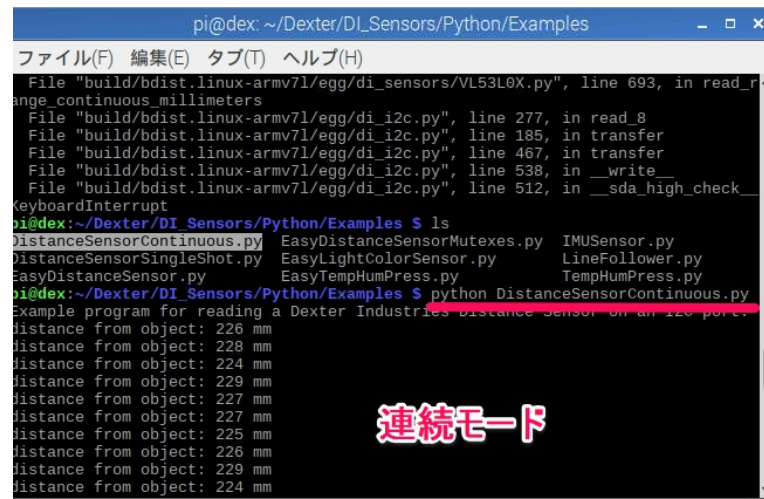
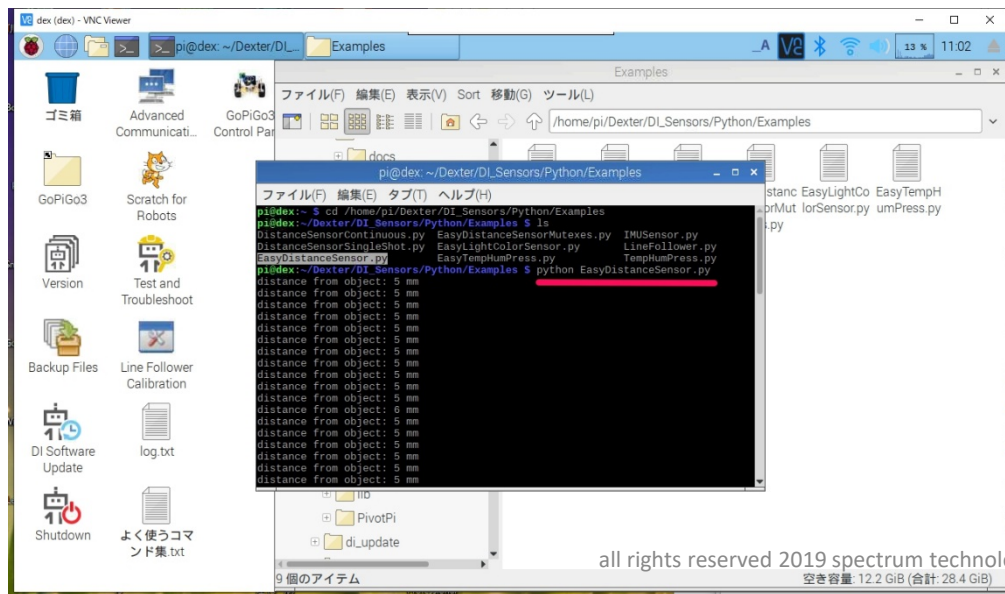
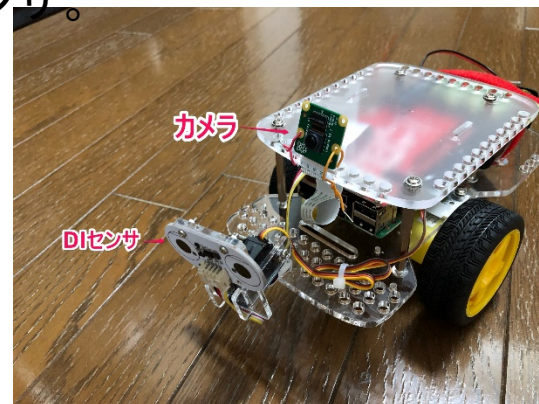
A). DIセンサ

- Distanceセンサ: センサから障害物までの距離を測るセンサ。
- <https://di-sensors.readthedocs.io/en/master/index.html>
- cd /home/pi/Dexter/DI_Sensors/Python/Examples
- python EasyDistanceSensor.py
- python DistanceSensorContinuous.py
- python DistanceSensorSingleShot.py

連続モード

シングルモード

```
$ cd
/home/pi/Dexter/DI_Sensors/Python/Examples
$ python EasyDistanceSensor.py
```



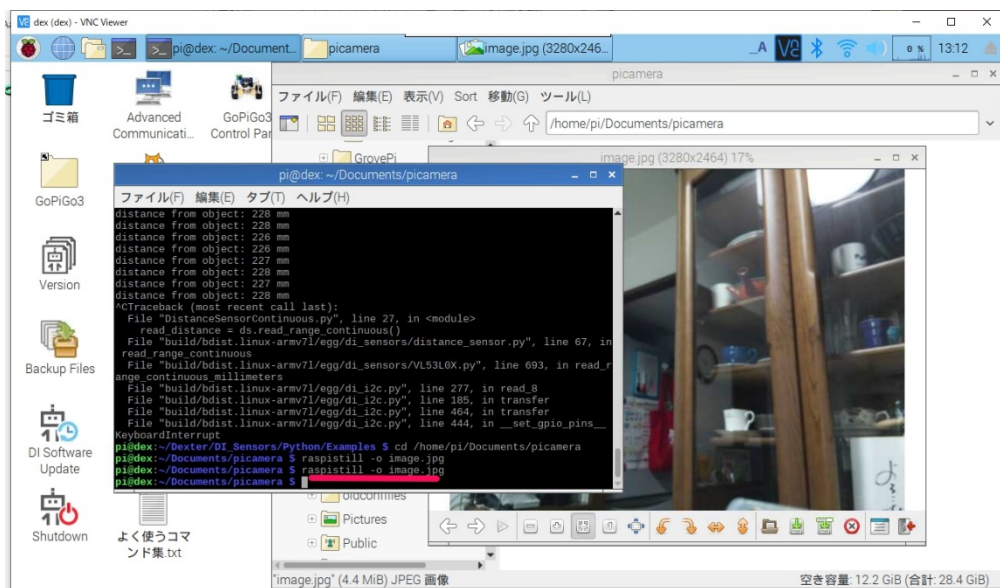
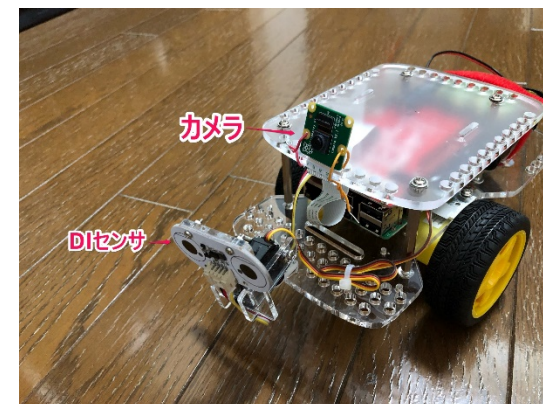
GoPiGo基本操作

⑧. GoPiGo 各種センサ類試験

B). カメラ(写真)

- カメラ: 写真及び動画撮影できます。
- <https://picamera.readthedocs.io/en/release-1.13/>
- `cd /home/pi/Documents/picamera`
- `raspistill -o image.jpg` 写真
- `python picam_img.py test1.jpg`で出力

```
$ cd
/home/pi/Documents/picamera
$ raspistill -o image.jpg
$ python picam_img.py
```



GoPiGo基本操作

⑧. GoPiGo 各種センサ類試験

D). カメラ(リアルタイム動画)

- カメラ: 写真及び動画撮影できます。

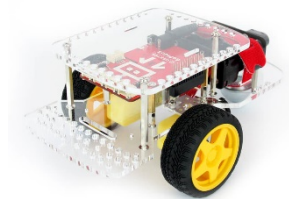
- <https://picamera.readthedocs.io/en/release-1.13/>

- `cd /home/pi/Documents/picamera`

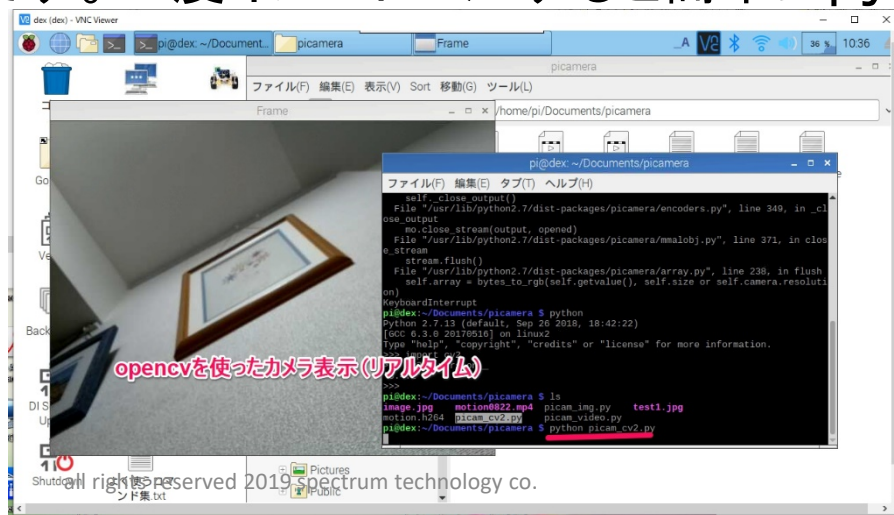
- `python picam_cv2.py` リアルタイム動画表示

- 画面サイズ: 640x480、Opencvを使って表示

- Opencvは、インテルが開発した画像用のプログラムで、顔認識、物体認識には必須です。一度インストールすると簡単にpythonで動作できます。



```
$ cd  
/home/pi/Documents/picamera  
$ python picam_cv2.py
```



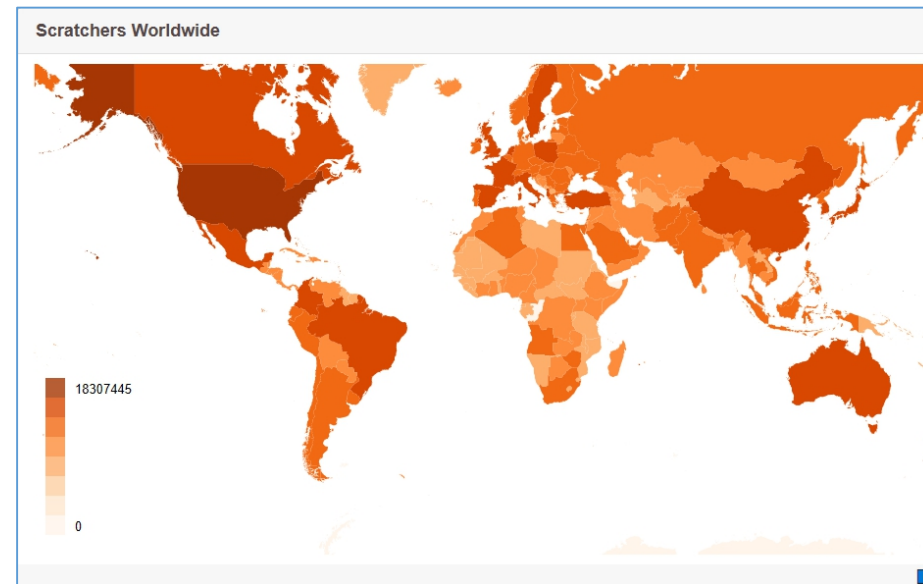
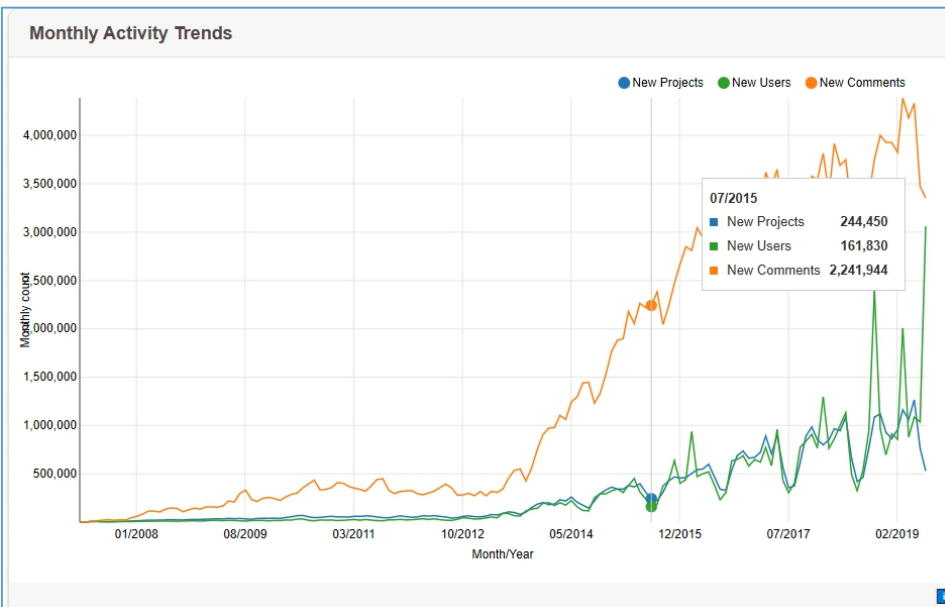
Scratch學習



Scratch学習

①. Scratch概要

- Scratchについて
- MITで開発した初心者向けのプログラミング言語。月間1百万ユーザ、グローバルに利用
- <https://scratch.mit.edu/>





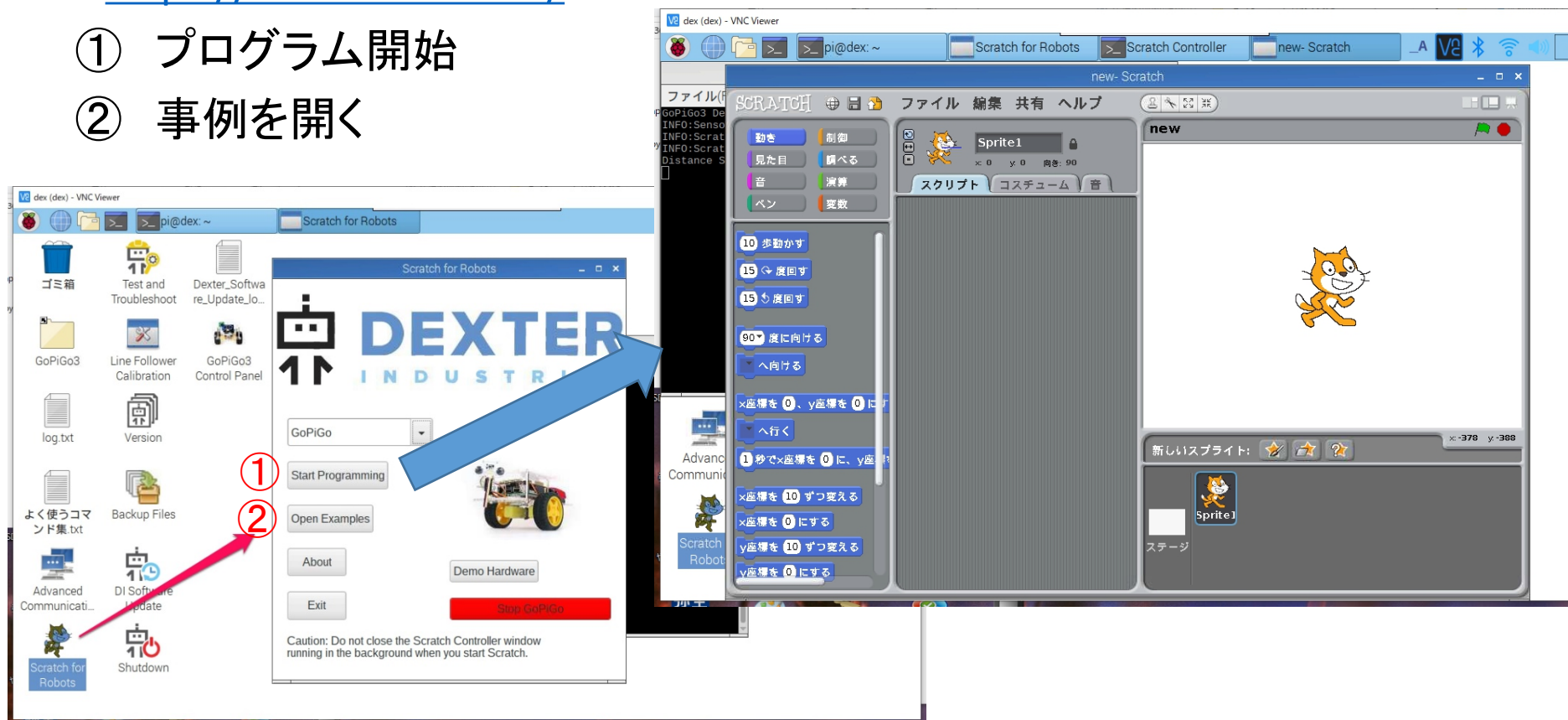
②. Scratchプログラム学習 基本操作

- Scratch言語を使ったRobotの動作

- <https://scratch.mit.edu/>

① プログラム開始

② 事例を開く





②. Scratchプログラム学習 基本操作

・ Scratch言語を使ったRobotの動作

① プログラム開始

② 事例を開く

The left screenshot shows the Scratch for Robots interface. The 'Open Examples' button is highlighted with a red arrow and a circled '2'. A blue arrow points from this button to the right screenshot.

The right screenshot shows a file explorer window displaying a list of example files. The file 'button_example.sb' is highlighted with a red box. A red arrow points to this file with the text 'ダブルクリックすると事例が現れます' (Double-click to show the example).



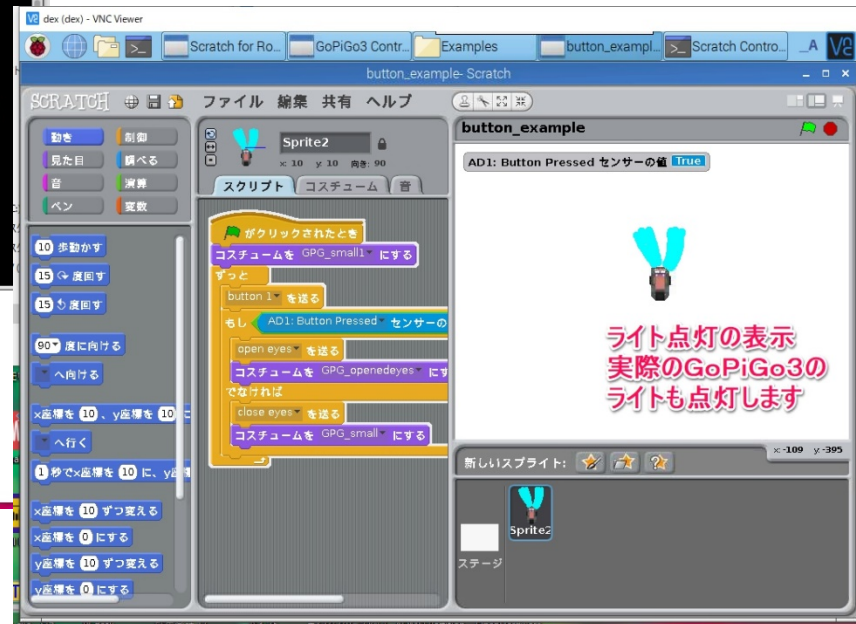
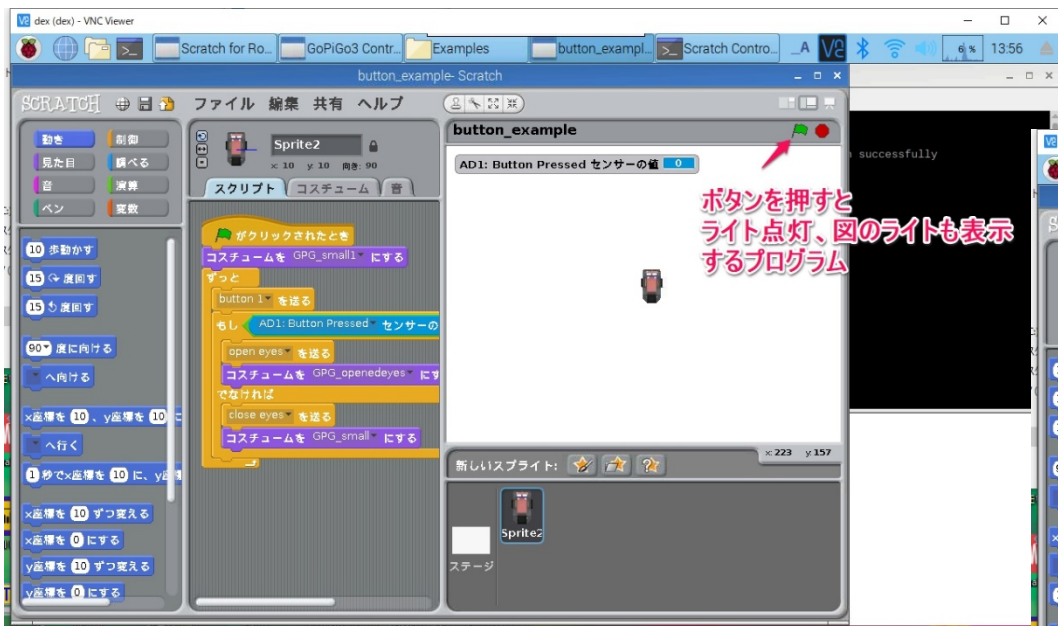
②. Scratchプログラム学習 ライト点灯事例

・ Scratch言語を使ったRobotの動作

① プログラム開始

② 事例を開く(ライト点灯事例)

```
$ cd
/home/pi/Dexter/GoPiGo3/S
oftware/Scratch/Examples
button_example.sb
```



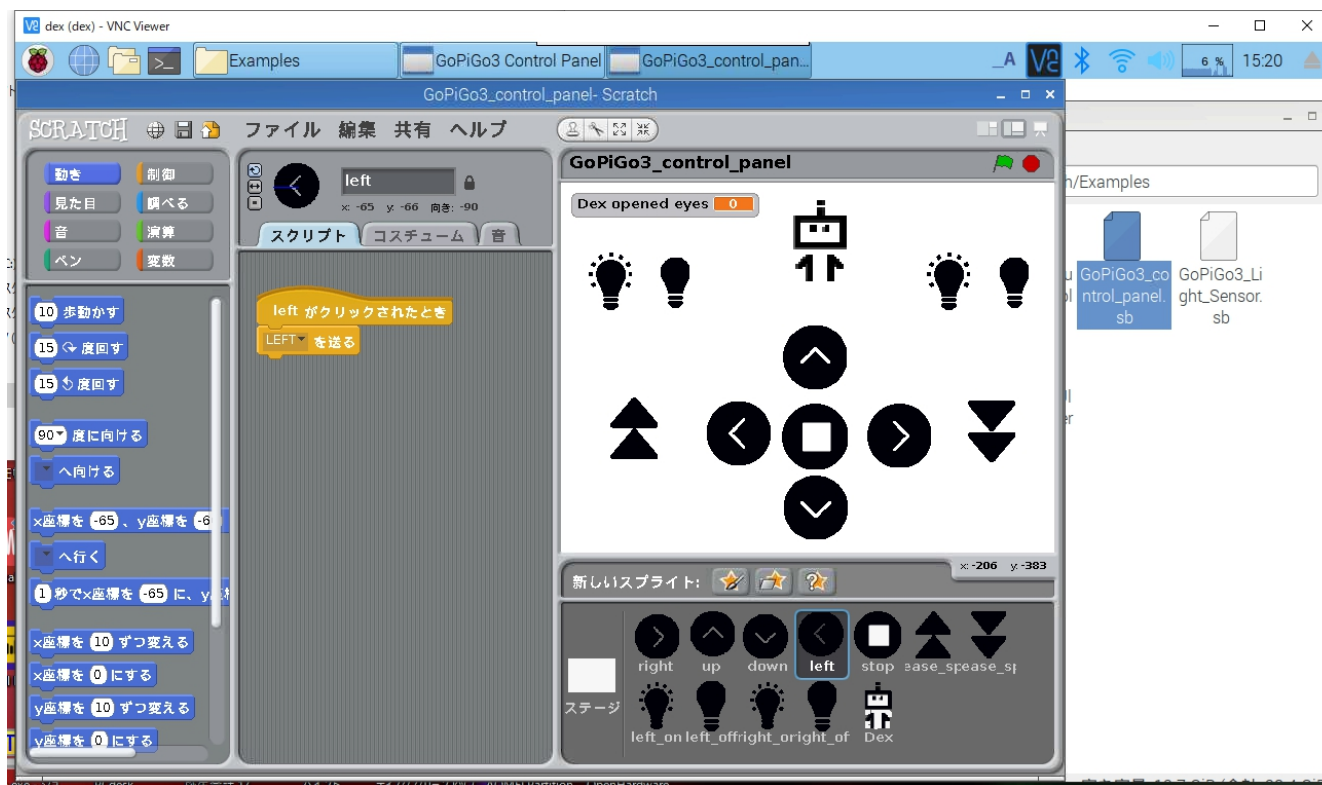


②. Scratchプログラム学習 コントロールパネル事例

- Scratch言語を使ったRobotの動作

- ① プログラム開始
- ② 事例を開く(コントロールパネル事例)

```
$ cd
/home/pi/Dexter/GoPiGo3/Soft
ware/Scratch/Examples
GoPiGo3_control_panel.sb
```





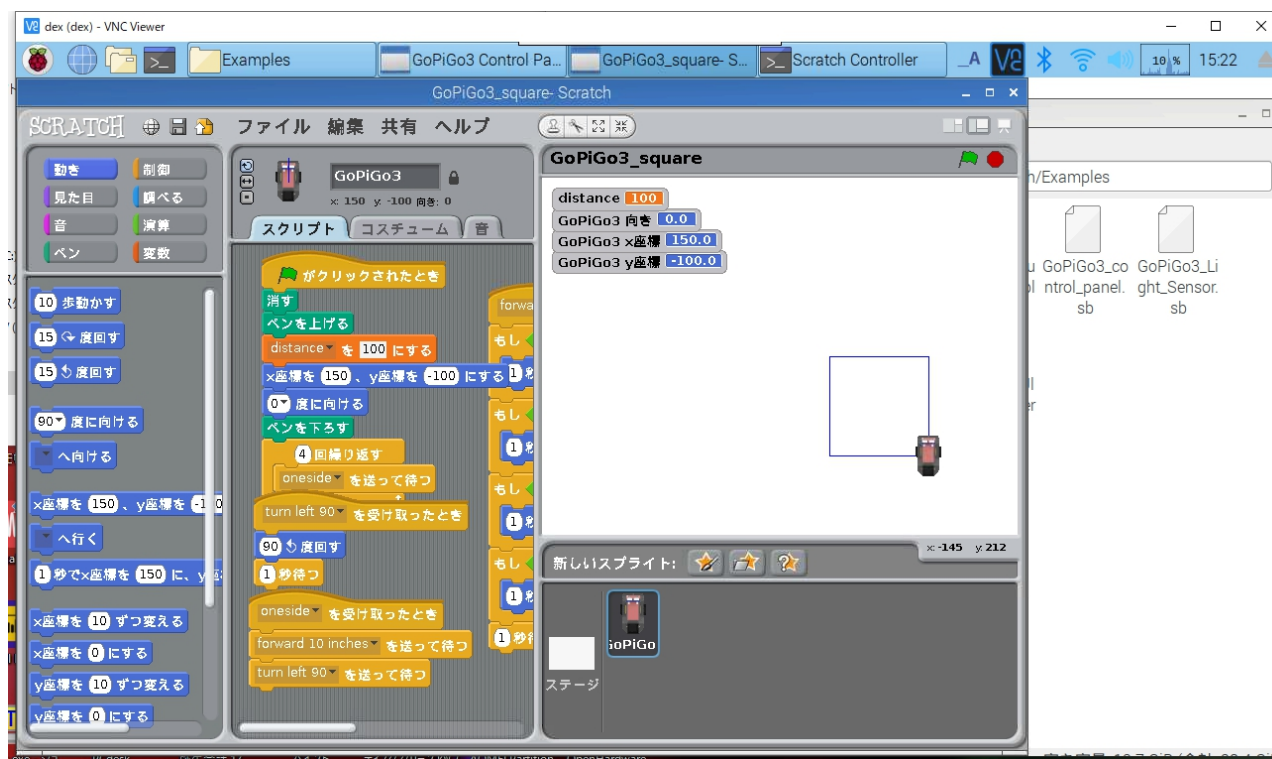
②. Scratchプログラム学習

正方形動作事例

- Scratch言語を使ったRobotの動作

- ① プログラム開始
- ② 事例を開く(正方形動作事例)

```
$ cd
/home/pi/Dexter/GoPiGo3/Software/Scratch/Examples
GoPiGo3_square.sb
```



画像認識開発

画像認識 開発

①. Opencv

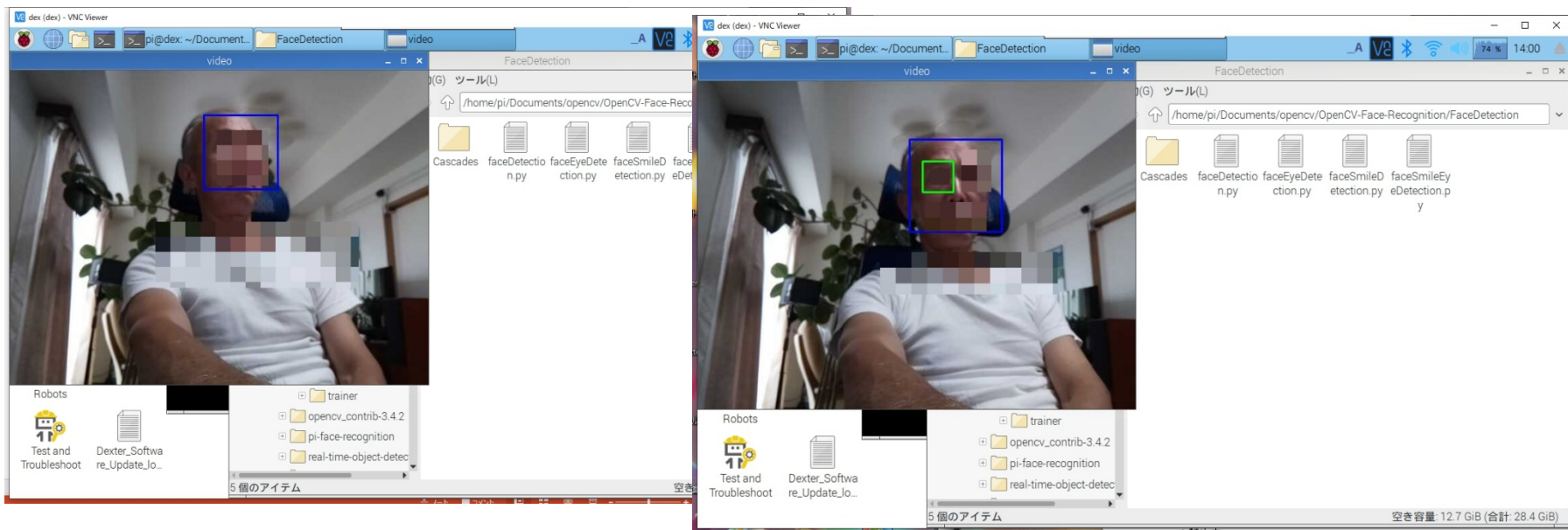
A). 顔認識(カメラ)



コマンド

```
$ cd /home/pi/Documents/opencv/OpenCV-Face-Recognition/FaceDetection
```

- Opencvを使ったwebカメラでの顔認識です。
 - <https://www.instructables.com/id/Real-time-Face-Recognition-an-End-to-end-Project/>
 - python faceDetection.py
 - python faceEyedetection.py
- 他。CPUのみで動作のため遅い。



画像認識 開発

①. Opencv

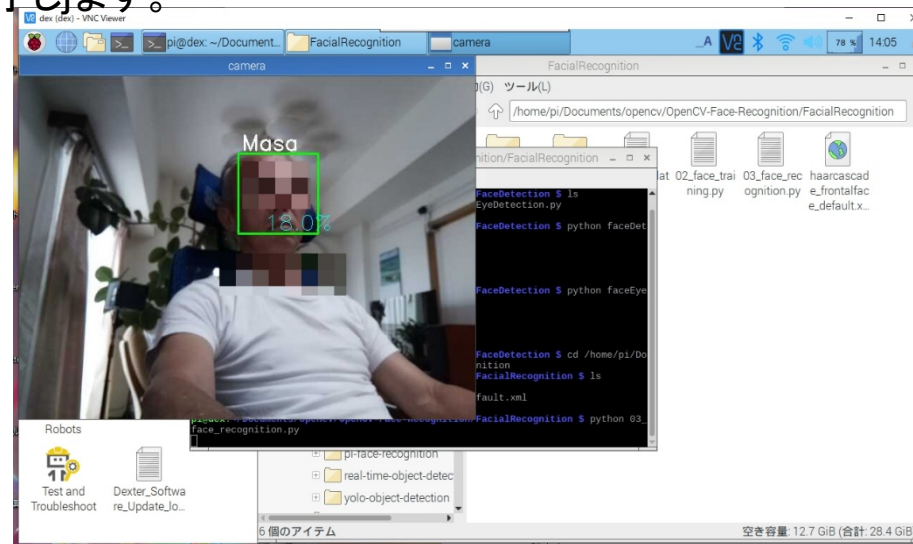
B). 顔検出(カメラ)

コマンド

```
$ cd /home/pi/Documents/opencv/OpenCV-Face-Recognition/FacialRecognition
```



- Opencvを使った、webカメラでの登録した顔を検出します。
- <https://www.instructables.com/id/Real-time-Face-Recognition-an-End-to-end-Project/>
- 顔の登録
- python3 01_face_dataset.py
 - Face id:0-3までを入力してください。
 - 自動で顔のスクランが始められます。数分で終了します。
- python3 02_face_training.py
- python3 03_face_recognition.py
 - カメラで認識します
 - 名前と確度がでます。
 - 26行目のface id=1をMasaにしています。変更してください。



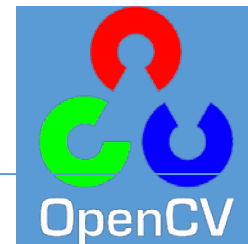
画像認識 開発

①. Opencv

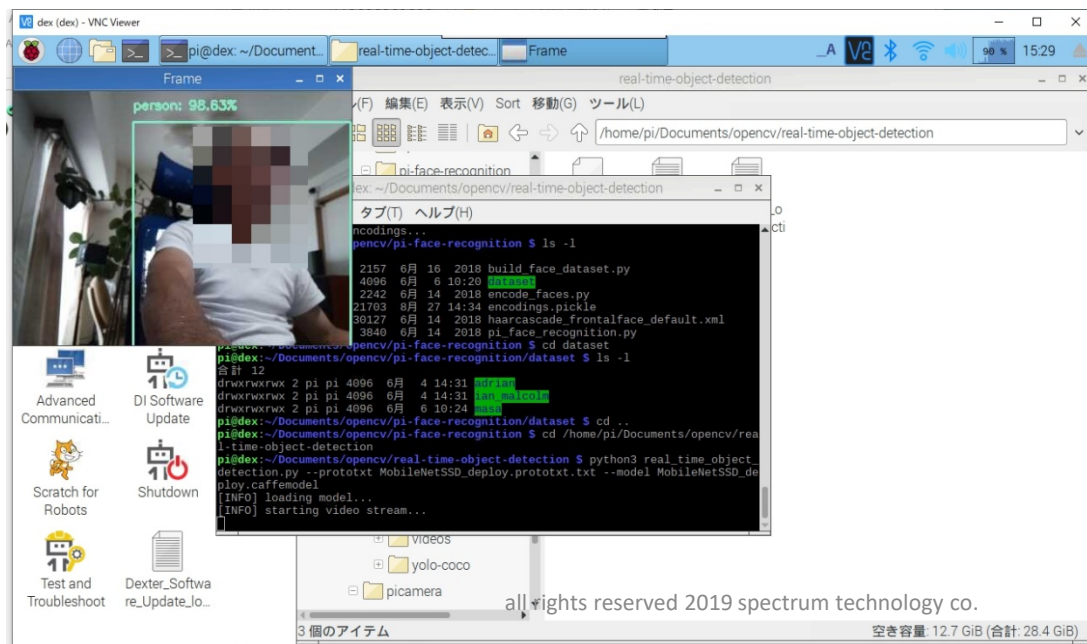
D). 物体検出(カメラ)

コマンド

```
$ cd /home/pi/Documents/opencv/real-time-object-detection
```



- <https://www.pyimagesearch.com/2017/09/18/real-time-object-detection-with-deep-learning-and-opencv/>
- 物体検出(カメラ)
- `python3 real_time_object_detection.py --prototxt MobileNetSSD_deploy.prototxt.txt --model MobileNetSSD_deploy.caffemodel`
- かなり重い



python3の
みで動作

画像認識 開発

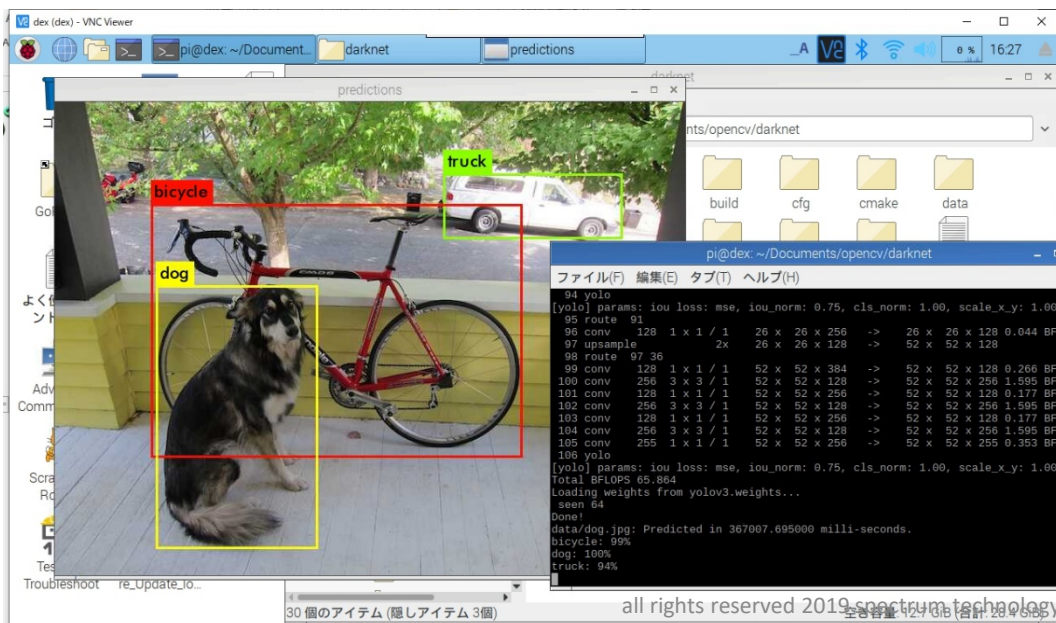
②. Yolo

A). 物体検出(写真)

コマンド

```
$ cd /home/pi/Documents/opencv/darknet
```

- Darknetとして有名なYoloです。基本c言語で書いたものをpython3でも使える用になってます。
- <https://pjreddie.com/darknet/yolo/>
- 一番基本の写真の例です。
- `./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg`
- 表示まで5分位かかります。動画は、GPU付きのもので行ってください。



自転車、車、
犬を検出

画像認識 開発

②. Yolo

C). 物体検出(ビデオ)

コマンド

```
$ cd /home/pi/Documents/opencv/yolo-object-detection
```

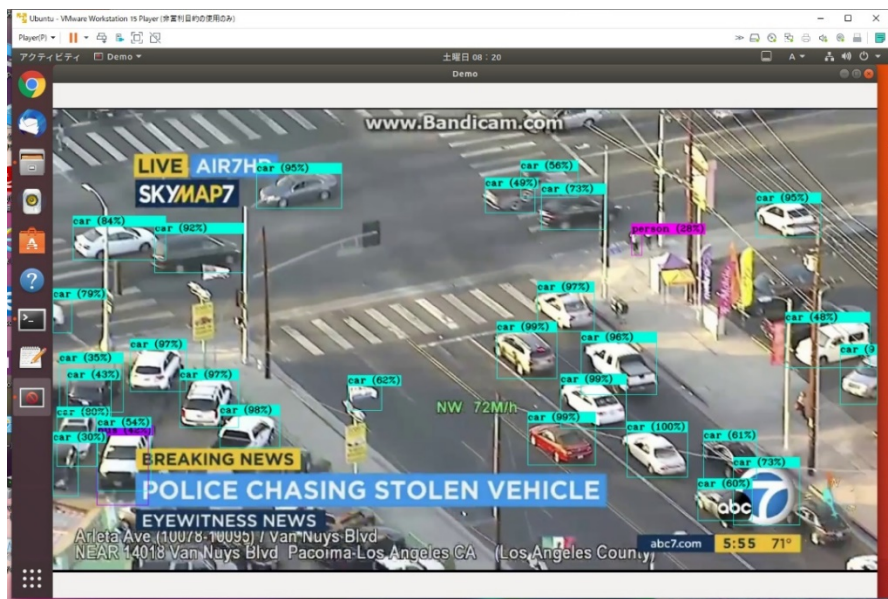
- <https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/>

- 物体検出(ビデオ)

- `python3 yolo_video.py --input videos/car_chase_02.mp4 --output output/car_chase_02.avi --yolo yolo-coco`

python3のみで動作

- GPUを使用しないと動きません「[はじめての顔認識・物体認識用AI開発キット](#)」を購入ください



応用例

応用例

②. マウスコントロール

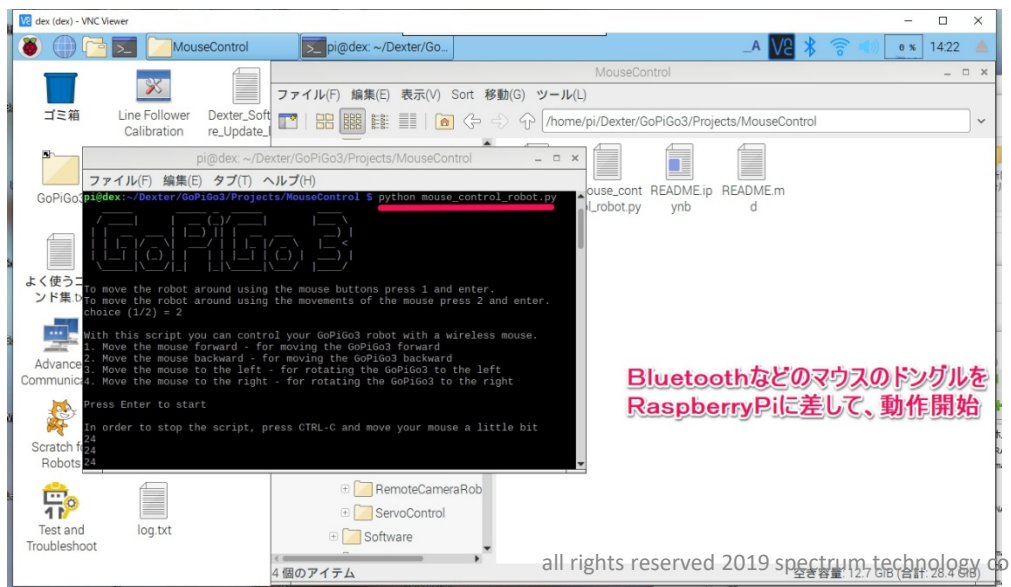
コマンド

```
$ cd
```

```
/home/pi/Dexter/GoPiGo3/Projects/MouseControl
```

```
$ python mouse_control_movement.py
```

- <https://edu.workbencheducation.com/cwists/preview/26765x>
- <https://youtu.be/aR7vXHT9pZk>
- 概要: ワイヤレスマウスを使って、GoPiGoを動かす。
- 対象: 高校、大学など
- 言語: Python
- `cd /home/pi/Dexter/GoPiGo3/Projects/MouseControl`
- `python mouse_control_movement.py`
- マウスの dongle を Raspberry Pi の USB に差しして、マウスを前後、左右に動かすと動作



④. 障害物回避自走

コマンド

\$ cd

/home/pi/Dexter/GoPiGo3/Projects/IntelligentObjectAvoider

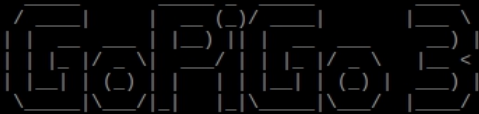
```
$ python3 robot.py
```

- 概要:DIセンサ、サーボを使って、GoPiGoを動かす。
- 対象:高校、大学など
- 言語:Python
- `cd /home/pi/Dexter/GoPiGo3/Projects/IntelligentObjectAvoider`
- `python3 robot.py`
- DIセンサ(サーボ付き)で、障害物を回避しながら自走

```
pi@dex: ~/Dexter/GoPiGo3/Projects/IntelligentObjectAvoider
```

ファイル(F) 編集(E) タブ(T) ヘルプ(H)

```
Installing collected packages: joblib, scikit-learn  
Successfully installed joblib-0.13.2 scikit-learn-0.21.3  
WARNING: You are using pip version 19.1.1, however version 19.2.3 is available.  
You should consider upgrading via the 'pip install --upgrade pip' command.  
pi@dex:~/Dexter/GoPiGo3/Projects/IntelligentObjectAvoider $ ls  
README.md devsketches robot.py  
pi@dex:~/Dexter/GoPiGo3/Projects/IntelligentObjectAvoider $ python3 robot.py
```



Let your GoPiGo3 move around and avoid any obstacles.
Pay attention to how your GoPiGo3 moves around.
Avoid sharp corners / edges as the algorithm wasn't made for advanced stuff.

```
waiting threads to fire up  
[obstacleFinder] thread fully active  
[robotController] thread fully active  
Rotating at 60.0 degrees and driving for 90.0 cm  
Rotating at 40.0 degrees and driving for 90.0 cm  
ACpi@dex:~/Dexter/GoPiGo3/Projects/IntelligentObjectAvoider $
```

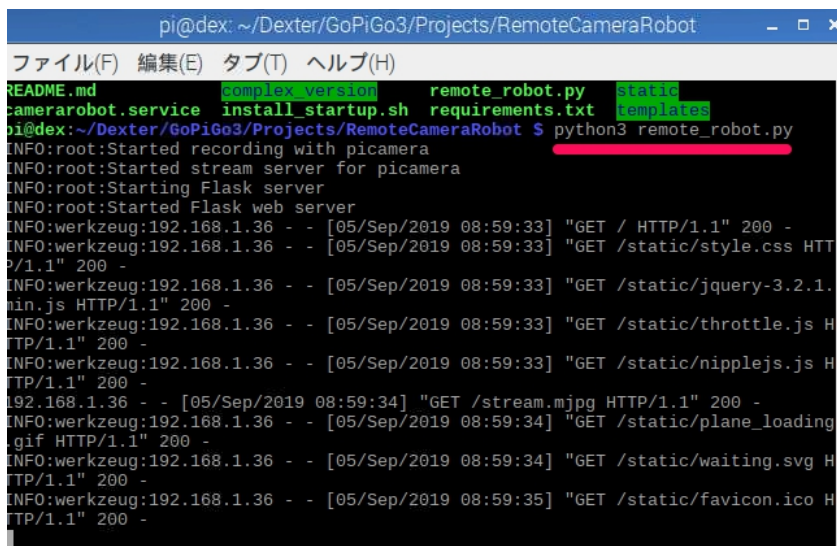
応用例

⑤. Webカメラ配信

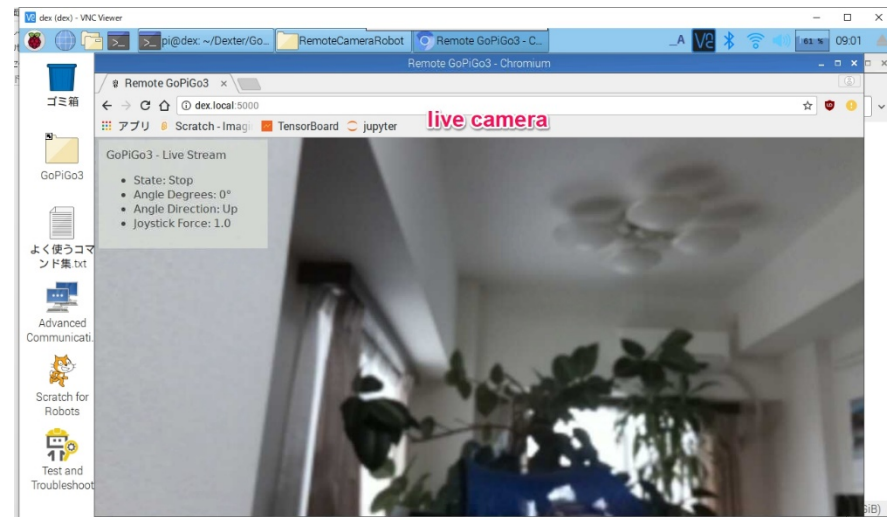
コマンド

```
$ cd /home/pi/Dexter/GoPiGo3/Projects/RemoteCameraRobot  
$ python3 remote_robot.py
```

- 概要: webカメラでライブ配信。
- 対象: 高校、大学など
- 言語: Python
- `cd /home/pi/Dexter/GoPiGo3/Projects/RemoteCameraRobot`
- `python3 remote_robot.py`
- ブラウザを立ち上げて、<http://dex.local:5000>にアクセス



```
pi@dex: ~/Dexter/GoPiGo3/Projects/RemoteCameraRobot  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
README.md complex_version remote_robot.py static  
camerarobot.service install_startup.sh requirements.txt templates  
pi@dex:~/Dexter/GoPiGo3/Projects/RemoteCameraRobot $ python3 remote_robot.py  
INFO:root:Started recording with picamera  
INFO:root:Started stream server for picamera  
INFO:root:Starting Flask server  
INFO:root:Started Flask web server  
INFO:werkzeug:192.168.1.36 - - [05/Sep/2019 08:59:33] "GET / HTTP/1.1" 200 -  
INFO:werkzeug:192.168.1.36 - - [05/Sep/2019 08:59:33] "GET /static/style.css HTTP/1.1" 200 -  
INFO:werkzeug:192.168.1.36 - - [05/Sep/2019 08:59:33] "GET /static/jquery-3.2.1.min.js HTTP/1.1" 200 -  
INFO:werkzeug:192.168.1.36 - - [05/Sep/2019 08:59:33] "GET /static/throttle.js HTTP/1.1" 200 -  
INFO:werkzeug:192.168.1.36 - - [05/Sep/2019 08:59:33] "GET /static/nipplejs.js HTTP/1.1" 200 -  
INFO:werkzeug:192.168.1.36 - - [05/Sep/2019 08:59:34] "GET /stream.mjpg HTTP/1.1" 200 -  
INFO:werkzeug:192.168.1.36 - - [05/Sep/2019 08:59:34] "GET /static/plane_loading.gif HTTP/1.1" 200 -  
INFO:werkzeug:192.168.1.36 - - [05/Sep/2019 08:59:34] "GET /static/waiting.svg HTTP/1.1" 200 -  
INFO:werkzeug:192.168.1.36 - - [05/Sep/2019 08:59:35] "GET /static/favicon.ico HTTP/1.1" 200 -
```



応用例

⑥. 自動走行(レーン追尾)

コマンド

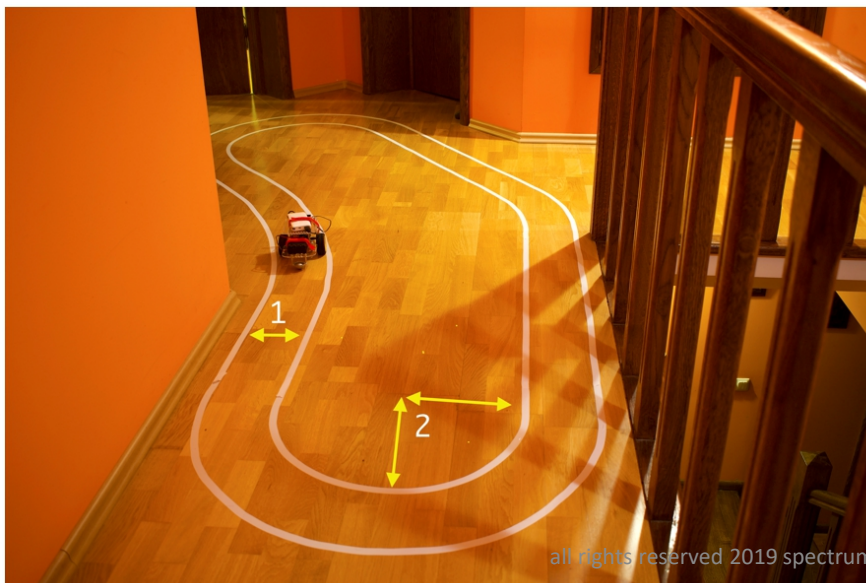
\$ cd

/home/pi/Dexter/GoPiGo3/Projects/Pixy2LaneTracker

\$ python3 pixy2_lane_follower.py

- <https://github.com/DexterInd/GoPiGo3/tree/master/Projects/Pixy2LaneTracker>
- 概要: Pixy2カメラでレーンを追尾しながら自動走行。
- 対象: 高校、大学など
- 言語: Python
- cd /home/pi/Dexter/GoPiGo3/Projects/Pixy2LaneTracker
- python3 pixy2_lane_follower.py

Pixy2カメラが必要。
オプションで提供

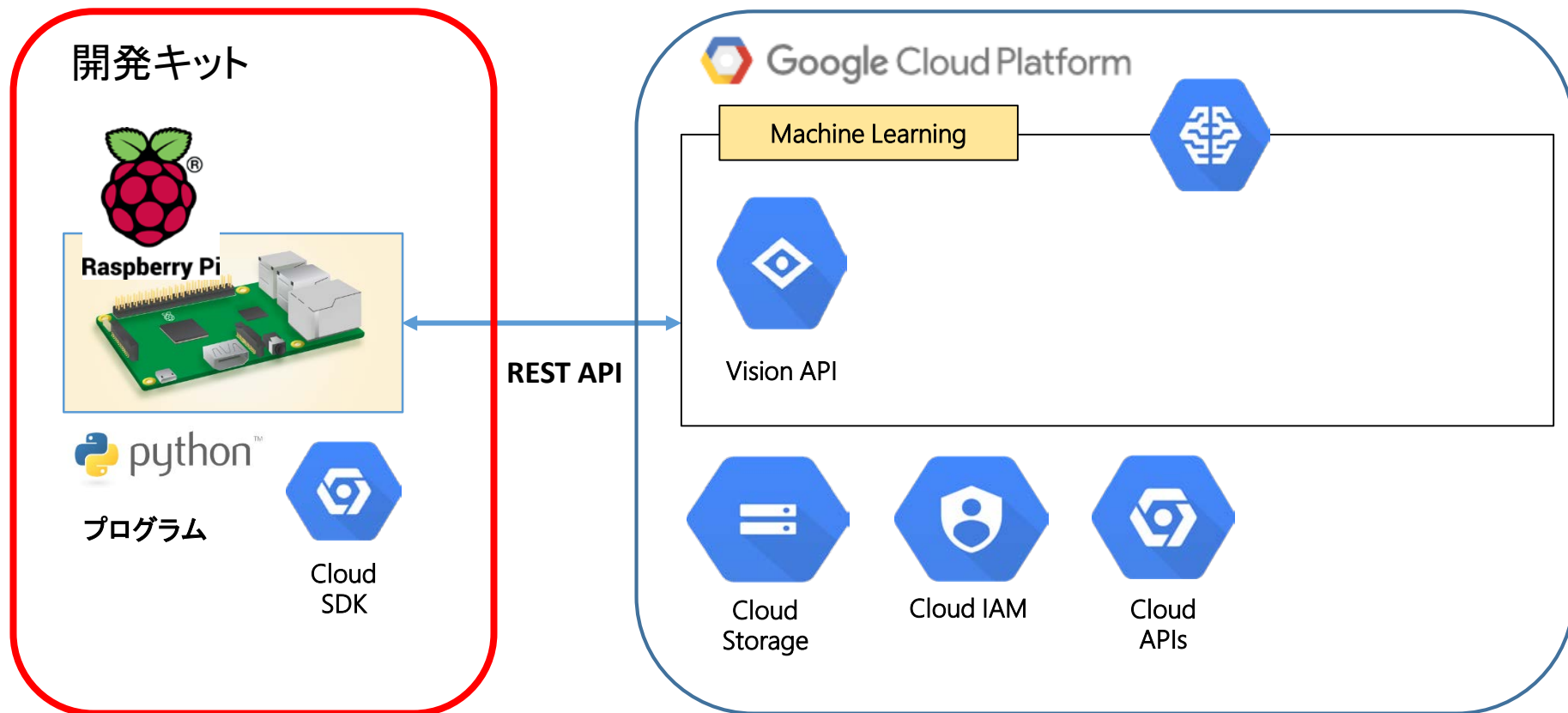


GCP開発



③. グーグル・クラウド開発

1. 開発キット全体像



③. ゴーグル・クラウド開発

2. 開発キットプログラム一覧

A:実用可能
B:チャレンジ
C:試験段階



クラウドAI名	プログラム名	機能内容	実用度 (当社評価)	信頼度 (当社評価)	備考
Cloud vision api (画像認識)	OCR(文字認識)	画像内のテキストを検出、抽出できます。幅広い言語がサポートされており、言語の種類も自動で判別されます。	A	90%	手書き文字はC:10%位
	Face(顔検出)	画像に含まれる複数の人物の顔を検出できます。感情や帽子の着用といった主要な顔の属性についても識別されます。	A	90%	特定の人顔認識はできません
	Label(ラベル検出)	乗り物や動物など、画像に写っているさまざまなカテゴリの物体を検出できます。	A	90%	
	Logo(ロゴ検出)	画像に含まれる一般的な商品ロゴを検出できます	C	10%	文字のないロゴは検出できない

③. グーグル・クラウド開発

3.Cloud vision

① 1_OCR(文字認識)

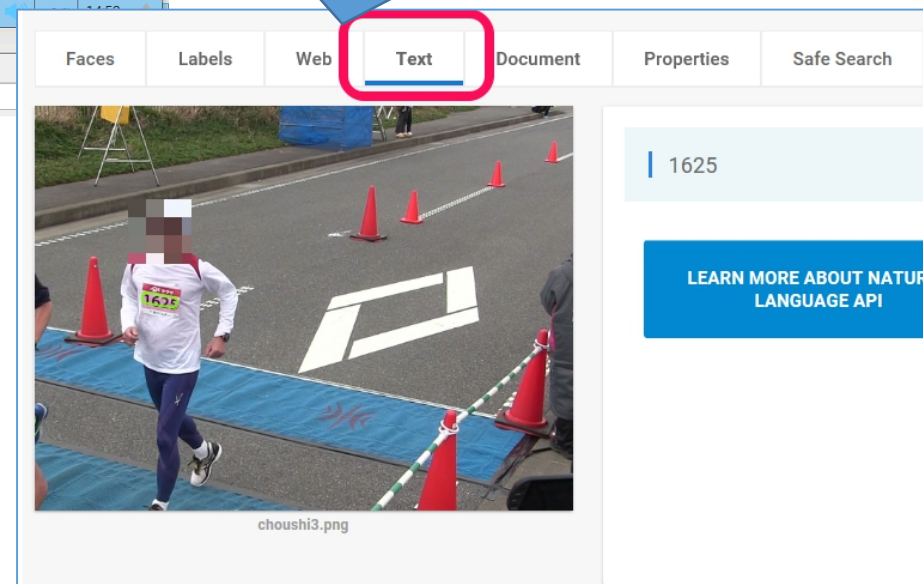
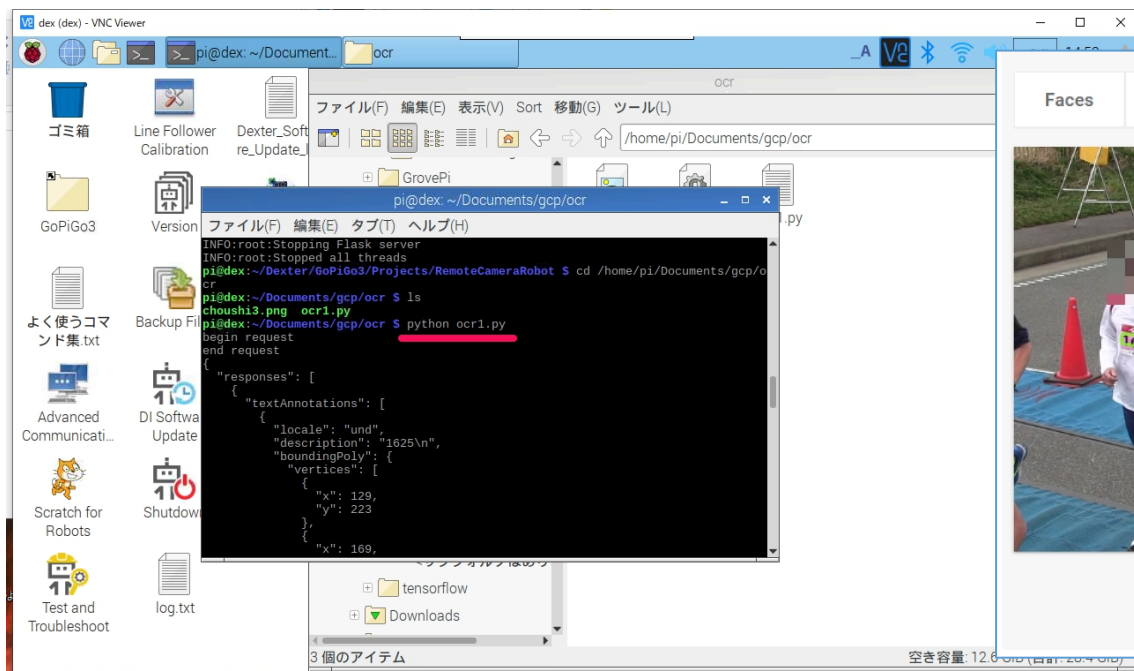
- テキスト認識: ocr1.py(画像からテキスト抽出)
 - Raspberry piからocr1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata1.jsonの双方にでます。

コマンド

```
$ cd /home/pi/Documents/gcp/1_ocr  
$ python ocr1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>



③. グーグル・クラウド開発

3.Cloud vision

② 2_face(顔検出)

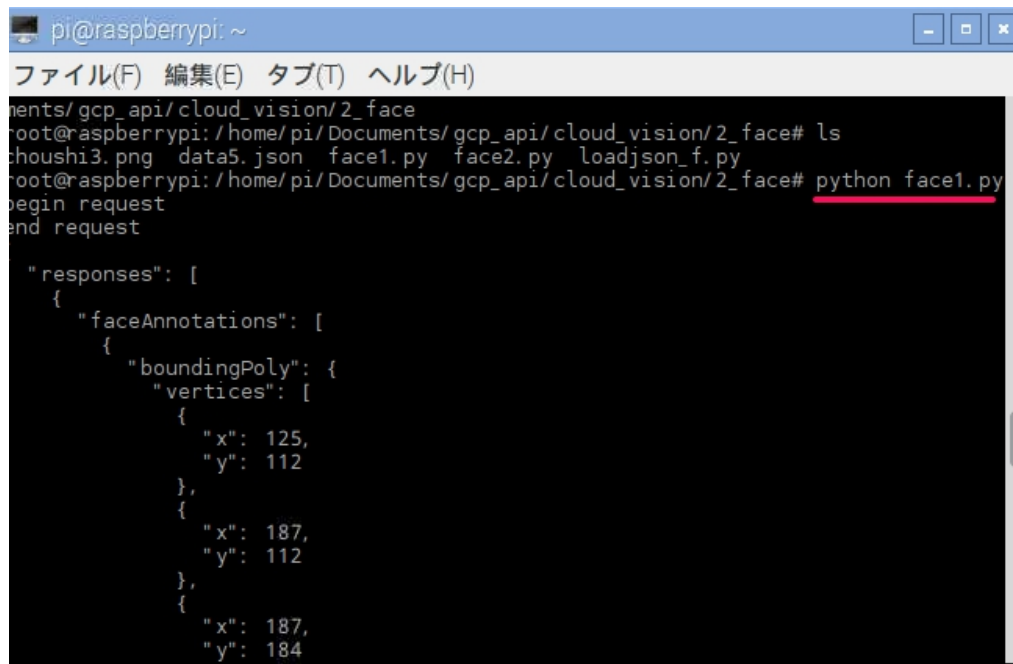
- face1.py(http使用)
 - Raspberry piからface1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata5.jsonの双方にでます。

コマンド

```
# cd /home/pi/Documents/gcp/2_face  
# python face1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>



```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
ments/gcp_api/cloud_vision/2_face  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/2_face# ls  
choushi3.png data5.json face1.py face2.py loadjson_f.py  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/2_face# python face1.py  
begin request  
and request  
"responses": [  
  {  
    "faceAnnotations": [  
      {  
        "boundingPoly": {  
          "vertices": [  
            {  
              "x": 125,  
              "y": 112  
            },  
            {  
              "x": 187,  
              "y": 112  
            },  
            {  
              "x": 187,  
              "y": 184  
            },  
            {  
              "x": 125,  
              "y": 184  
            }  
          ]  
        },  
        "confidence": 0.99999994  
      }  
    ]  
  }  
]
```



顔認識は、できません。プライバシー保護

③. グーグル・クラウド開発

3.Cloud vision

③ 3_label(ラベル検出)

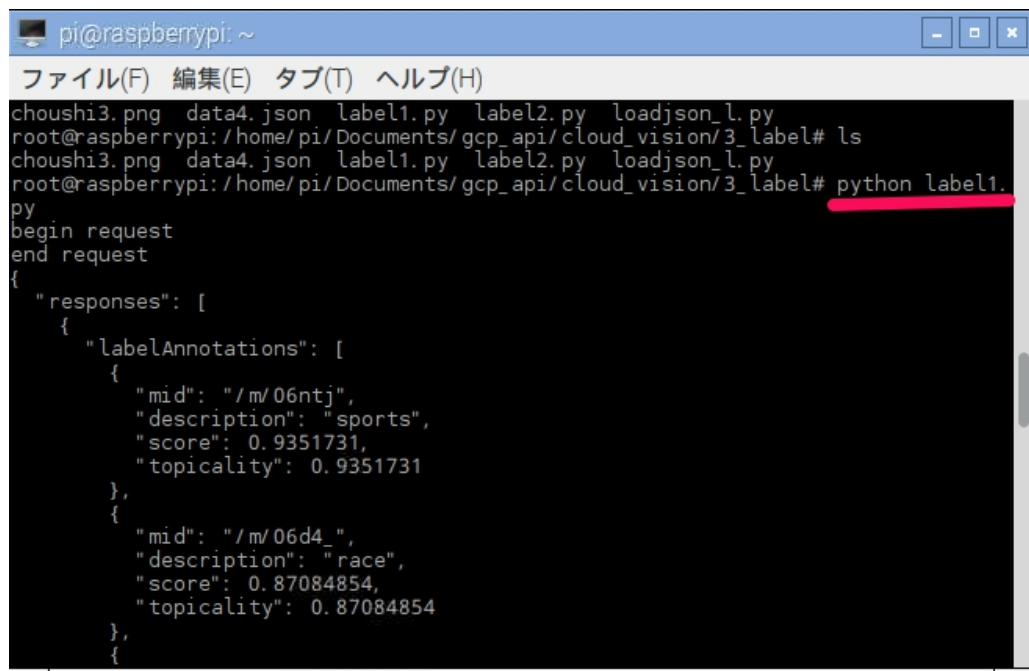
- label1.py(http使用)
 - Raspberry piからlabel1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata4.jsonの双方にでます。

コマンド

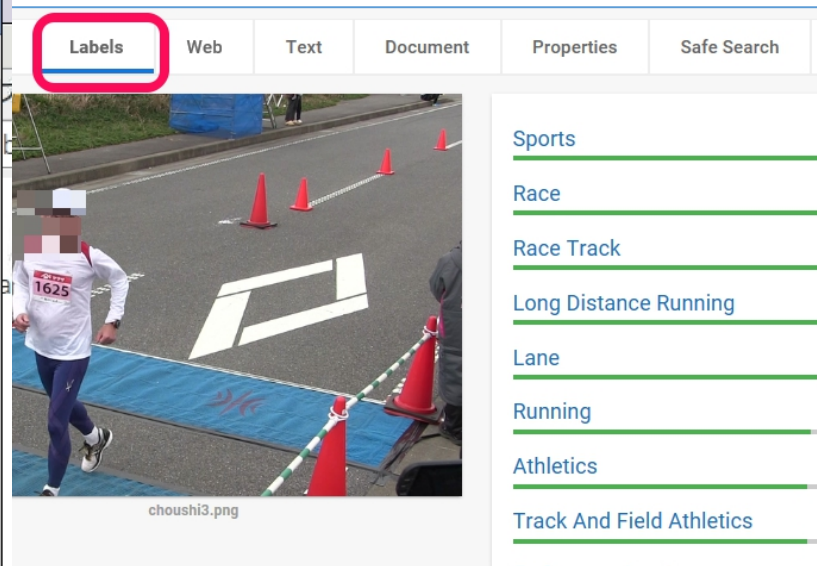
```
# cd /home/pi/Documents/gcp/3_label  
# python label1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>



```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
choushi3.png data4.json label1.py label2.py loadjson_l.py  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/3_label# ls  
choushi3.png data4.json label1.py label2.py loadjson_l.py  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/3_label# python label1.py  
begin request  
end request  
{  
  "responses": [  
    {  
      "labelAnnotations": [  
        {  
          "mid": "/m/06ntj",  
          "description": "sports",  
          "score": 0.9351731,  
          "topicality": 0.9351731  
        },  
        {  
          "mid": "/m/06d4",  
          "description": "race",  
          "score": 0.87084854,  
          "topicality": 0.87084854  
        }  
      ]  
    }  
  ]  
}
```



画像のカテゴリが出力されます

③. グーグル・クラウド開発

3.Cloud vision

④ 6_logo(ロゴ検出)

- logo1.py(http使用)
 - Raspberry piからlogo1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata9.jsonの双方にでます。

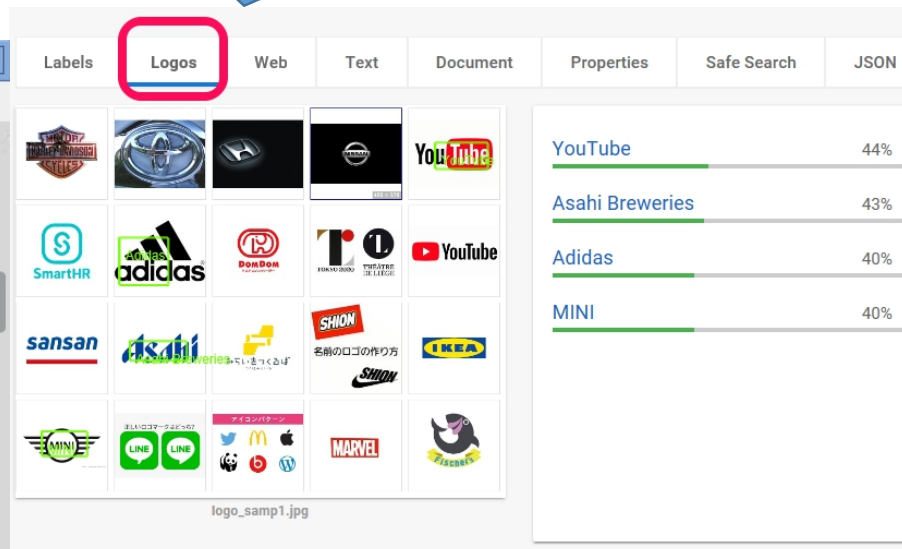
コマンド

```
# cd /home/pi/Documents/gcp/6_logo  
# python logo1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>

```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/6_logo# ls  
data9.json loadjson_lo.py logo1.py logo2.py logo_samp1.jpg  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/6_logo# python logo1.py  
begin request  
end request  
{  
  "responses": [  
    {  
      "logoAnnotations": [  
        {  
          "mid": "/g/1dv8ql4m",  
          "description": "YouTube",  
          "score": 0.44070464,  
          "boundingPoly": {  
            "vertices": [  
              {  
                "x": 797,  
                "y": 72  
              },  
              {  
                "x": 897,  
                "y": 72  
              },  
              {  
                "x": 897,  
                "y": 72  
              },  
              {  
                "x": 797,  
                "y": 72  
              }  
            ]  
          }  
        }  
      ]  
    }  
  ]  
}
```



③. グーグル・クラウド開発

3.Cloud vision

⑤ face(顔認識)

- camera-vision-face.py (ライブラリ使用)
 - Raspberry piからcamera-vision-face.pyを実行します。
 - Picameraで写真が撮られて、顔の感情分析をしてコマンド画面に出力されます。

コマンド

```
# cd /home/pi/Documents/gcp/GoogleVisionTutorials  
# python camera-vision-face.py
```

```
pi@dex: ~/Documents/gcp/GoogleVisionTutorials  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
-rw-r--r-- 1 root root 911 4月 4 13:45 README.md  
-rw-r--r-- 1 root root 27010 4月 4 13:45 camel.jpg  
-rw-r--r-- 1 root root 1904 4月 4 13:45 camera-vision-face.py  
-rw-r--r-- 1 root root 1905 4月 4 13:45 camera-vision-label.py  
-rw-r--r-- 1 root root 1915 4月 4 13:45 camera-vision-logo.py  
-rw-r--r-- 1 root root 180430 4月 4 13:45 face-google-vision.jpg  
-rw-r--r-- 1 root root 599266 4月 4 13:58 image.jpg  
-rw-r--r-- 1 root root 175829 4月 4 13:45 wheel.jpg  
pi@dex:~/Documents/gcp/GoogleVisionTutorials $ python camera-vision-face.py  
{  
  "responses": [  
    {  
      "faceAnnotations": [  
        {  
          "angerLikelihood": "VERY_UNLIKELY",  
          "blurredLikelihood": "VERY_UNLIKELY",  
          "boundingPoly": {  
            "vertices": [  
              {  
                "x": 408  
              },  
              {  
                "x": 926
```



Alexa開発



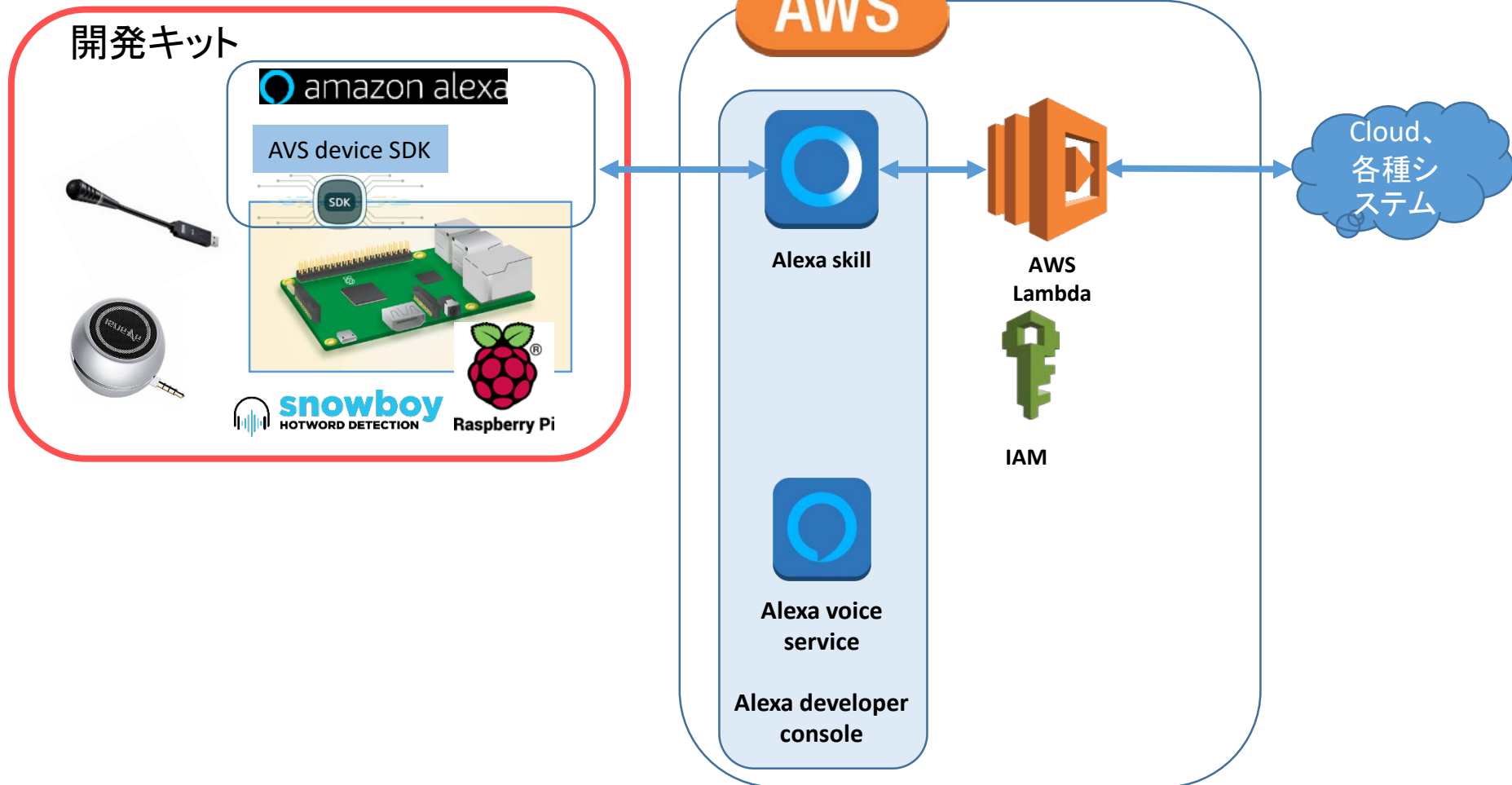
Alexa voice
service

Alexa開発

①.全体構成



Alexa voice service



Alexa開発

④. Amazon. co. jp設定



Echo

- Amazon alexaログイン
 - Amazon. co. jpのアカウントでログインします。
- <https://alexa.amazon.co.jp/>

amazon alexa

ログイン

[パスワードを忘れた方](#)

Eメールまたは携帯電話アカウントの番号

Amazonのパスワード

☐ パスワードを表示

☐ ログインしたままにする [詳細](#)

ログイン

続行することで、Amazonの利用規約およびプライバシー規約に同意するものとみなされます。

Amazonの新しいお客様ですか？

[新しいAMAZONのアカウントを作成](#)



ホーム

再生中

ミュージック・ビデオ・本

リスト

リマインダー・アラーム

スキル

スマートホーム

試してみよう!

設定

ヘルプとフィードバック

村上 正彦さんではありませんか? 設定

ホーム

日本寿町の天気
AccuWeather.com

21°

日曜日、午前12:00 JSTまで乾燥注意報を発表中です。
2019年5月9日 本曜日
曇り

最高: 23° / 最低: 11°
風速: 南南西 13.5 km/h

体感温度: 21°
降水確率: 3%

午前11:00 20° 午前12:00 21° 午前1:00 22° 午前2:00 23° 午前3:00 22° 午前4:00 22° 午前5:00 21°

日	天気	最高	最低
金 5月10日	晴	25°	12°
土 5月11日	晴	25°	12°
日 5月12日	晴	24°	13°
月 5月13日	晴	23°	14°
火 5月14日	晴	24°	14°

音声フィードバック

▶ Alexaが聞き取った内容: "エコー 今日の天気は"

Alexaは正しく応答しましたか? [はい](#) [いいえ](#)

カードを削除 [詳細はこちら](#)

元に戻る



⑥. Alexa Voice Service開発

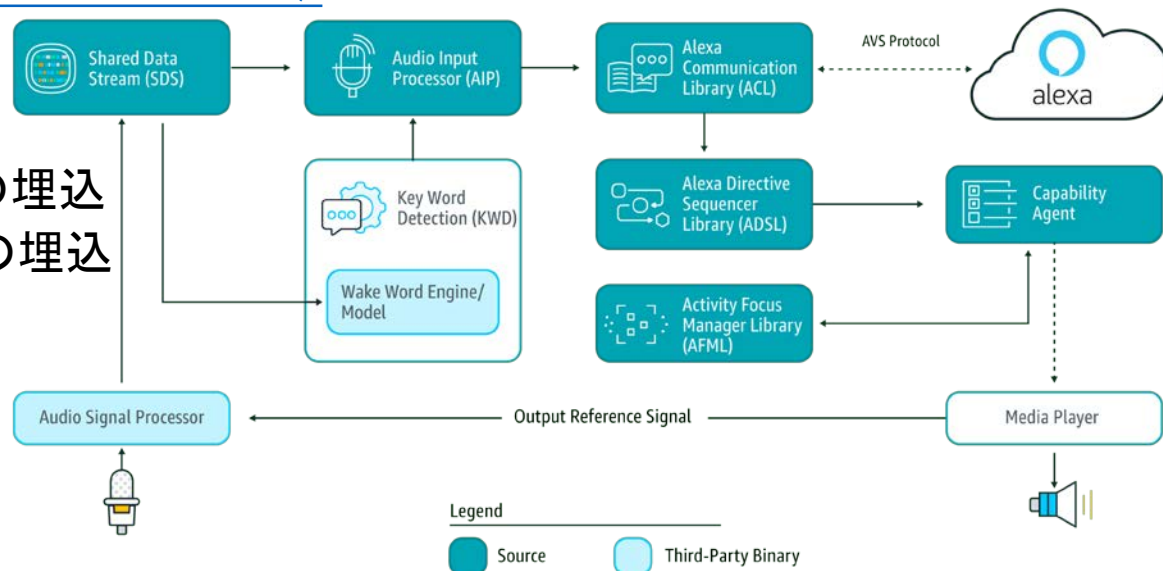
概要

- Alexa Voice Service(以下AVSと略します)を使って、自社製品にAlexaの機能を搭載可能になります。Echoなどのデバイスを使わないで、スピーカ、マイクを付けてAlexaを実現します。AVS Device SDKをRaspberry Piに搭載し、開発環境を提供します。また、スピーカとマイクを付属しており、Alexaの端末として利用できます。

- <https://developer.amazon.com/ja/alexa-voice-service>
- <https://alexa.github.io/avs-device-sdk/>

用途

- 高機能スマートスピーカ
- TVなどの家電製品への埋込
- スマートホーム製品への埋込



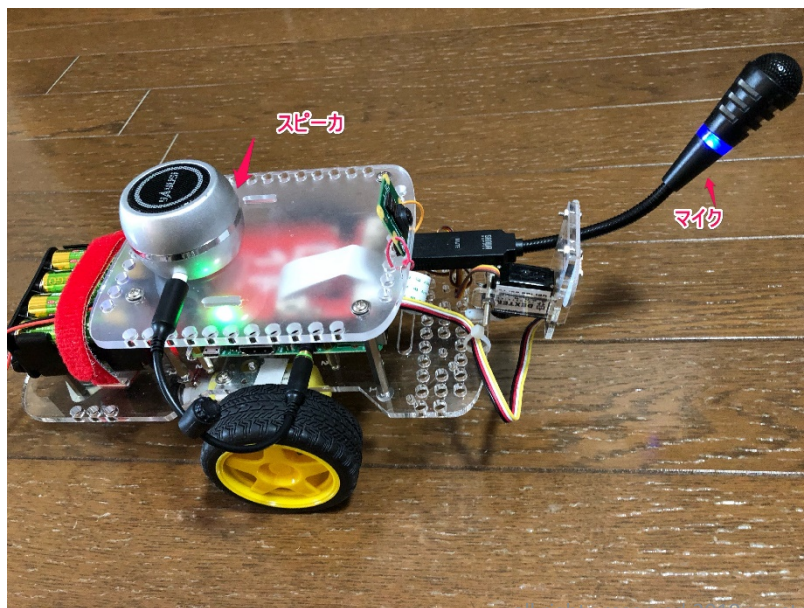
⑥. Alexa Voice Service開発

• マイク、スピーカ設定

- <https://github.com/alexavoice/avs-device-sdk/wiki/Raspberry-Pi-Quick-Start-Guide-with-Script>

• マイク、スピーカ接続

- マイク: USBにマイクを接続してください。青いLEDが点灯、赤が点灯はMuteなのでMuteを触って解除してください。
- スピーカ: イヤホンジャックに接続して、裏面のボタンを押し、青いLEDが点灯するのを確認してください。最初は、内蔵の電池をUSBミニに接続して充電してから使用してください。





⑥. Alexa Voice Service開発

Piのフォルダ位置
/home/pi/alexa/
\$ sudo bash startsnowboy.sh

• AVS sample APP動作

- コマンド画面でAuthorized, Alexa is currently idleが出れば、正常に動作中
- マイクに向かって、Alexa、今日の天気は？と呼びかけて見ましょう。(Listeningになれば、AlexaのWakeupを認識した状態になります)
- 日本語への切替、c, 1, 6と入力

```
pi@raspberrypi: ~/alexa
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
Press 'q' followed by Enter at any time to quit the application.
#####
# Connecting... #
#####
# Authorized! #
#####
# Alexa is currently idle! #
#####
# Client not connected! #
#####
# Alexa is currently idle! #
#####
```

本画面は、ログを省略したもので、実際のものとは違います。
\$ sudo bash startsnowboy.sh

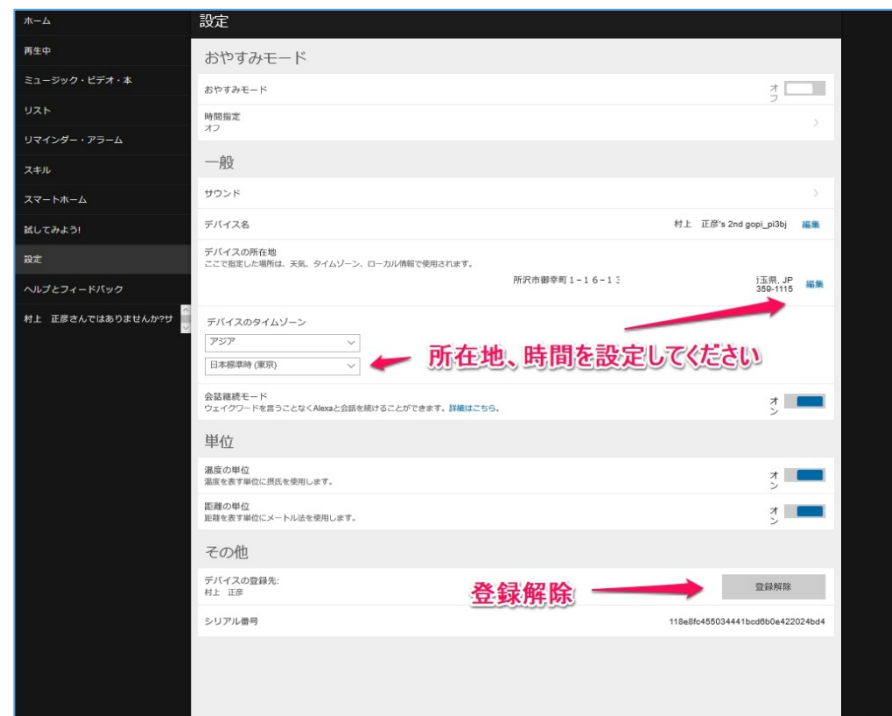
```
pi@raspberrypi: ~/alexa
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
#####
# Alexa is currently idle! #
#####
# Listening... # ← Alexaと呼びかけ
#####
# Thinking... # ← 今日の天気は？
#####
# RenderTemplateCard
#-----
# Focus State      : FOREGROUND
# Template Type    : WeatherTemplate
# Main Title       : 日本寿町
#-----
#####
# Speaking... # ← Alexaが回答中
#####
```

⑥. Alexa Voice Service開発

Alexa voice
service

• デバイス確認

- <https://alexa.amazon.co.jp/spa/index.html#settings>
- 登録したPiのAlexa端末の確認を上記のURLから実施
- デバイスがオンラインになっていることを確認
- 端末を選択して、場所、時間の設定を行ってください。
- 登録解除も可能です。



Alexa開発

⑥. Alexa Voice Service開発

- ・音声によるGoPiGo操作

音声によるGoPiGo操作

・GoPiGo動作

- ・ 音声のコマンドをサーバで受信し、その内容をPiのサーバに伝えます。

発話例

- ・ アレクサ、前方向
- ・ アレクサ、右方向

GoPiGo動作

- ・ Go aheadの場合、1秒前進。
- ・ プログラムで変更可

