

## サンプルプログラム説明書

紙面の都合により本文に含めることができなかったサンプルプログラム使用時の注意点をここにまとめましたので参考にしてください。

### ■ 全般

- ① サンプルプログラムを一式まとめてコピーできる様にプロジェクトフォルダにまとめました。
- ② **プロジェクトフォルダ**を一式で**カレントディレクトリ**(ユーザー名が□□□であれば /home/□□□) にコピーしてください。
- ③ プロジェクトフォルダ名は自由に編集してください。  
(筆者の場合は/home/kuro/kurosIRO)
- ④ 発話で使用する音声データやサウンド再生に使用する音源ファイルは著作権の関係により電子付録に含めることができないため各自で入手してください。
- ⑤ サンプルプログラム中で使用する各種ソフトウェア、Python モジュールも同様に電子付録に含めることができないため各自で入手してください。
- ⑥ プロジェクトフォルダの中にデータファイルを保存するためのフォルダをデフォルトで作成しております。
- ⑦ サンプルプログラム中にファイルの読み込みなどでファイル名を指定している部分が多いので、各々で設定したユーザー名、プロジェクトフォルダ名、各種データファイル保存場所に合わせて適宜編集してください。でないとファイルを正しく読み込めずエラーになります。

### ■ 4 章 ロボットの基本機能の実現

#### ■ 4-2 移動する

[サンプルプログラム]

SIRO\_test\_motor.py

- ① siro\_motor\_speed で SIRO の走行速度を調整してください。
- ② 前進、後退、左回転、右回転、停止を3秒毎に切り替えて動作を確認しています。他にも確認したい動作があればソースコードを適宜編集してください。

## ■ 4-3 発話する

〔サンプルプログラム〕

`SIRO_test_speak.py`

- ① `sp_vol` でスピーカー音量を調整してください。
- ② `numid=3` としていますが、環境によって変わる可能性があります。うまく動作しない場合は本書の解説：スピーカーの音量調整を参照してください。
- ③ 男性の声が良ければ `voice` で男性の声の音声ファイルを選んでください。オリジナルではコメントアウトしています。
- ④ `option` を適宜編集して声を変えてみてください。
- ⑤ 関数 `siro_speak` の引数（文字列）をアレンジして自由にテストしてください。
- ⑥ スピーカー音量が小さい場合は、ゲインを調整すればうまくいく可能性があります。本書の解説：サウンドカードのゲイン調整（スピーカー）を参照してください。

## ■ 4-4 サウンド再生

〔サンプルプログラム〕

`SIRO_test_sound.py`

- ① サンプルプログラムを動かす前に音源ファイル（MP3）を入手する必要があります（筆者はインターネット上のフリー音源をダウンロードして使用しています）。
- ② 音源を入手したらプロジェクトフォルダの中に保存します。  
（筆者の場合は `/home/kuro/kurosSIRO/data/sound/□□□.mp3`）
- ③ サンプルプログラム中の音源ファイルのファイル名の部分を適宜編集してください。
- ④ 音源が MP3 ではなく WAV の場合は `mpg321` のかわりに `aplay` を使用してください。

## ■ 4-5 顔認識①：人の顔を見つける

〔サンプルプログラム〕

`SIRO_test_opencv_facerecognition1.py`

- ① 顔認識学習モデル（カスケード分類器）のファイル名の部分を適宜編集してください。  
（筆者の場合は `/home/kuro/kurosSIRO/haarcascades/haarcascade_frontalface_alt.xml`）
- ② 動作が重い場合はカメラ画像の解像度、FPS を落としてみてください。
- ③ 誤検知が多い場合は認識可能な顔のサイズ大きくしてみてください。

## ■ 4-6 顔認識②:家族の顔を判別する

[サンプルプログラム]

`SIRO_test_opencv_facedataget.py`

`SIRO_test_opencv_facerecogtrain.py`

`SIRO_test_opencv_facerecognition2.py`

- ① 顔画像データを保存するフォルダ名の部分を適宜編集してください。  
(筆者の場合は/home/kuro/kurosSIRO/data/face/)
- ② 機械学習結果の出力ファイル名の部分を適宜編集してください。  
(筆者の場合は/home/kuro/kurosSIRO/trainer.yml')
- ③ 顔判別結果の出力部分 `print("筆者:", face_counter[0] . . . . .)` 各々の  
IDNo.の割り当てに従い適宜編集してください。

## ■ 4-7 音声認識

[サンプルプログラム]

`julius_dictation-kit-version.sh`

`julius_mydic-version.sh`

`julius_module.sh`

- ① Julius ディクテーションキットの設定ファイルのファイル名の部分を適宜編集してください。筆者の場合 /home/kuro/kurosSIRO/julius/dictation-kit-4.5/\*\*\*\*\*)
- ② オリジナル辞書ファイルのファイル名の部分を適宜編集してください。  
(筆者の場合 /home/kuro/kurosSIRO/julius/mydic/\*\*\*\*\*)
- ③ 音声認識がうまくいかない場合はオプション設定を試みてください。オプション設定の詳細については GitHub の情報を参照してください。
- ④ 筆者が作ったオリジナル辞書を元ネタにして自由に辞書を作成してみてください。

[サンプルプログラム]

`SIRO_test_voice_recognition.py`

- ① Julius 起動シェルスクリプトのファイル名の部分を適宜編集してください。  
(筆者の場合 /home/kuro/kurosSIRO/julius/julius\_module.sh")
- ② 音声認識がうまくいかない場合、マイクのゲイン調整でうまくいく可能性があります。  
本書の解説：サウンドカードのゲイン調整（マイク）を参照してください。

## ■ 4-8 ロボットの記憶

### [データファイル]

`personal.csv`  
`schedule1.csv`  
`schedule2.csv`

- ① データファイル（ひな形）を各々の ID No.の割り当てに従い適宜編集してください。
- ② エクセルで開くと文字化け（文字コードがシフト JIS になります）するので注意が必要です。メモ帳などのテキストエディターであれば UTF-8 で開いてくれますのでそちらで編集してください。

### [サンプルプログラム]

`SIRO_test_memory.py`

- ① データファイルのファイル名の部分を適宜編集してください。  
（筆者の場合 `home/kuro/kurosSIRO/data/****.csv`）
- ② 読み込んだデータを表示したり、編集してファイルに書き込んだり、読み込み、書き込みともに問題ないことを確認していますが、他にも確認したいことがあれば適宜ソースコードをアレンジしてください。

## ■ 4-9 障害物の検知

### [サンプルプログラム]

`SIRO_test_uss.py`

- ① まずは本などの平らな物体を使って問題なく検知できることを確認してください。問題なければ他の物体も試してみてどんな物が検知し易くてどんな物体が検知し難いのか確認してみてください。

## ■ 4-10 気温と湿度の検知

### [サンプルプログラム]

`SIRO_test_ondo.py`

- ① GitHub 公開の Python モジュールは必ずプロジェクトフォルダ（実行ファイルと同じ階層）に保存してください。実行ファイルと同階層に置いておくことで読み込みが可能となります。（筆者の場合 `home/kuro/kurosSIRO/Adafruit_BME280.py`）
- ② 家の中に信頼できる温湿度計があればそれと比較してみてください。差が大きいようで

あればオフセットをつけて調整してみてください。

#### ■ 4-11 ジェスチャー認識

[サンプルプログラム]

**SIRO\_test\_gesture.py**

- ① GitHub 公開の Python モジュールは必ずプロジェクトフォルダ（実行ファイルと同じ階層）に保存してください。実行ファイルと同階層に置いておくことで読み込みが可能となります。（筆者の場合 home/kuro/kurosSIRO/grove\_gesture\_sensor.py）
- ② ジェスチャーセンサーが確実に動作していることが確認できたら以下の様に 18 行目と 55 行目をコメントアウトすればセンサーの反応速度を確認できます。

```
# print("ジェスチャー認識中")  
# time.sleep(1)
```

#### ■ 4-12 バッテリー残量の検知

[サンプルプログラム]

**SIRO\_test\_battery.py**

- ① GitHub 公開の Python モジュールは必ずプロジェクトフォルダ（実行ファイルと同じ階層）に保存してください。実行ファイルと同階層に置いておくことで読み込みが可能となります（筆者の場合 home/kuro/kurosSIRO/upspackv2.py）。

#### ■ 4-13 CPU の異常検知

[サンプルプログラム]

**SIRO\_test\_cpu\_temp.py**

- ① 筆者は”vcgencmd measure\_temp”というコマンドを使って CPU 温度を検出しましたが、他にもっとスマートな方法があるかもしれません。

[サンプルプログラム]

**SIRO\_test\_cpu\_clockdown.py**

- ① 本文にも記載の通り、本当はビット演算を使ってビットごとのステータスを確認するのが正しいのですが、内容が難しくなりそうだったのであえて簡易なやり方に置き換えています。

## ■ 4-14 IP アドレスの確認

[サンプルプログラム]

`SIRO_test_localipaddress.py`

- ① 筆者は” hostname -I”というコマンドを使ってローカル IP アドレスを確認しましたが、他にもっとスマートな方法があるかもしれません。

`SIRO_test_globalipaddress.py`

- ① 筆者は” curl ifconfig.me“というコマンドを使ってグローバル IP アドレスを確認しましたが、他にもっとスマートな方法があるかもしれません。

## ■ 4-15 写真撮影

[サンプルプログラム]

`SIRO_test_globalipaddress.py`

- ① 撮影した写真を保存するフォルダ名の部分を適宜編集してください。  
(筆者の場合は /home/kuro/kurosSIRO/data/picture/)

## ■ 4-16 音声録音

[サンプルプログラム]

`SIRO_test_voice_record.py`

- ① 録音した音声データを保存するフォルダ名の部分を適宜編集してください。  
(筆者の場合は /home/kuro/kurosSIRO/data/message/)

## ■ 4-17 radiko 聴取

[サンプルプログラム]

`SIRO_test_radiko.py`

- ① 本文にも記載の通り radiko 再生用シェルスクリプトが現在うまく動作しませんが、今後、何らかの方法で使えるようにしたいためサンプルプログラムをそのまま残しておくことにしました。
- ② サンプルプログラムの動作確認はダミーのシェルスクリプト (play\_radiko.sh) を使って行っております。ダミーのシェルスクリプトの中身は1秒毎に日付を表示する簡単なシェルスクリプトです。

- ③ radiko 再生用シェルスクリプトのファイル名の部分を適宜編集してください。  
(筆者の場合 /home/kuro/kurosSIRO/play\_radiko.sh)

#### ■ 4-18 LINE 通知

〔サンプルプログラム〕

**SIRO\_test\_LINE.py**

- ① サンプルプログラム実行前に LINE Notify の設定を行ってください。LINE のマイページから通知用トークンを確認し、サンプルプログラム中のトークンの部分 (□□□としています) を編集してください。
- ② 画像ファイル送信テスト用のサンプル画像を準備、保存してください (筆者はインターネット上のフリー画像を使用しています)。
- ③ 画像ファイル名の部分を適宜編集してください。  
(筆者の場合 /home/kuro/kurosSIRO/neko.png)

#### ■ 4-19 ウィキペディア検索

〔サンプルプログラム〕

**SIRO\_test\_wikipedia.py**

- ① 検索ワードの部分を自由に編集してください。  
(筆者は猫が好きなので“猫”を検索してみました)

#### ■ 4-20 Web アプリ

〔サンプルプログラム〕

**SIRO\_test\_webapplication\_flask.py**

- ① 本プログラムは SIRO の Web アプリの一部 (ログイン画面とメニュー画面) を抽出した Web アプリの概要を理解するための動作確認用プログラムです。
- ② サンプルプログラム実行前に HTML テンプレートファイルをプロジェクトフォルダに保存してください。  
(筆者の場合 /home/kuro/kurosSIRO/templates/\*\*\*\*.html)
- ③ サンプルプログラム実行前に id とパスワードの部分 (\*\*\*\*\*としています) を適宜編集してください。
- ④ サンプルプログラム実行前にログイン画面に表示する画像を入手し (筆者はインターネット上のフリー画像を使用しています)、プロジェクトフォルダ内に保存してください。

(筆者の場合 /home/kuro/kurosSIRO/data/img/login.jpg)

- ⑤ サンプルプログラムを実行したら、Web ブラウザから Raspberry Pi のローカル IP アドレスを入力して Web アプリを開いてみてください。

(筆者の場合 <http://192.168.10.100:5000>)

- ⑥ メニュー画面を表示するところまでのサンプルプログラムになります。そのため各メニューボタンを押すと“Not found”とエラーになります。各メニューは SIRO の制御プログラム (SIRO\_all.py) で確認してください。

## ■5 章 SIRO の制御プログラム

[サンプルプログラム]

SIRO\_all.py

- ① 実際に使用している SIRO の制御プログラムです。こちらは各サンプルプログラムで動作を確認できた機能を統合し、さらに、応用機能、行動判断ロジックを追加したものになります。
- ② サンプルプログラムと同様に、ファイルの読み込みなどでファイル名を指定している部分が多数ありますのでそれらを適宜編集してください。
- ③ 発話（サウンド再生）と音声認識を同時に行うため、SIRO が自らの声（サウンド）にリアクションしないように工夫しています。具体的には、発話（サウンド再生）中であることを示す変数 `speaking` を作って、発話（サウンド再生）中にフラグを立てるようにしています。そして、音声認識の方では、フラグが立っている間は、音声認識結果を一旦キャンセルし、リアクションしないようにしています。
- ④ まだ製作途中の部分もありますがプログラムの骨子は完成させています。各々自由にアレンジしながら使用してください。
- ⑤ SIRO が勝手に走行すると危険なのでお散歩は禁止にしています。具体的には、お散歩禁止パラメータ `siro_dontmove` というものを作って、デフォルト設定で 1（禁止）にしています。SIRO にお散歩させたい場合は、`siro_dontmove = 0` としてください。
- ⑥ サンプルプログラム実行前に、Web アプリで使用する画像を各自で入手してください。具体的にはログイン画面に表示する写真、radiko 画面に表示する放送局のロゴなど。

(筆者はインターネット上の画像を使用しています)

入手した画像はプロジェクトフォルダ内に保存してください。

(筆者の場合 /home/kuro/kurosSIRO/data/img/\*\*\*\*.jpg)