音声認識とカメラを使ったセキュリティボックスの製作

石原　瑞樹　　今井　隆斗
中野　太晴　　佐藤　彩美

１.研究概要

私たちは Raspberry Pi で顔を認識して、特定の暗号で扉を開け閉めするセキュリティボックスを製作した。きっかけは、チームの中に顔認識を使ったロボットを作りたい人と音声認識を使ったロボットを作りたい人がいて、二つの要素を併せ持ったセキュリティボックスを製作することにした。

２.セキュリティボックスの構成と動作

今回製作したボックスの構成を図１に示す。

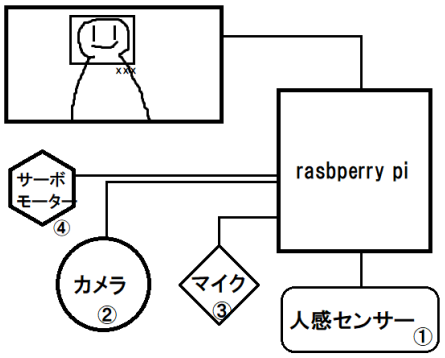

図１　セキュリティボックスの構成

（１）動作順序

セキュリティボックスの動作順序は以下の通りである。

①人感センサで人を感知するとセンサー上部の LED が発光し、プログラムが起動する。

②カメラに顔を映すと人を個別で認識し､｢合言葉を言ってください。｣と画面に表示される。

③マイクに個人に設定された特定の合言葉を話すと認識され、箱内部のサーボモータが動き鍵が外れる。

④人感センサが人を感知しなくなるとサーボモータが元の場所に戻り、箱に鍵がされる。

３．人感センサによる人の感知について

人感には人感センサを用いた。そのセンサの仕様は以下の通りである。

①人感センサが感知するまで近づく。

②感知すると LED が点灯し、プログラムを起動させる。

写真１　人感センサ

このセンサを使うために python のプログラムを使用した。そのプログラムを以下に示す。

（１）ソースコード

```
Import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
GPIO.setup(25,GPIO,OUT)
GPIO.setup(24,GPIO,IN)

while True:
      if GPIO,input(24) == GPIO,LOW
            GPIO.output(25,GPIO,HIGH)
      else:
            GPIO.output(25,GPIO.LOW)
```

「解説」

5～6 行目･･･出力するピンを 25 番、入力するピンを 24 番に設定。

9～10 行目･･･24 番に LOW が入力された場合 25 番に HIGH を出力し､LED を点灯させる。

11～12 行目･･･上記の場合以外のとき 25 番に LOW を出力し、LED を消灯させる。

４．カメラの顔認識について

画像認識には、OpenCV という python ライブラリを用いた。

（１）OpenCV とは

OpenCV とは画像や動画の処理機能をまとめたオープンソースのライブラリのことである。オープンソースであるため無料で使用することができる。また、Windows や Linux だけでなく、iOS や Android などの様々な OS に対応しており、汎用性も高いものとなっている。OpenCV を用いると、画像の認識、画像の編集、物体の検出ができるようになる。

（２）人の顔の画像認識

認識する人の顔の写真を学習させることで可能になる。その人一人につき 10～100 枚以上の顔写真を撮影し、フォルダに保存する。その後 OpnCV のダウンロードしたフォルダ内にある学習用のプログラムを実行することで、その人について登録される。

認識した人の顔をカメラがとらえると、人物の顔のに四角形が出現し、登録した情報から名前を表示する。（写真３）

複数の人の同時認識も可能。しかし、そのままの状態では認識の精度があまりよくないため、それぞれの人の写真の学習枚数を調節し、精度を上げた。

写真２　カメラ

写真３　顔認識の実例

５．音声認識について

音声認識には julius を用いた。

音声認識の辞書は事前に作成している。

動作は以下の通りである。

①マイクに向かって言葉を話す。

②話しかけたデータと辞書に登録した単語を照らし合わせて、最も近いものを認識する。

写真４　マイク

（１）Julius のセットアップ

①Julius をセットアップするために「julius-4.6.tar.gz」を下記の URL のページにある『Source code(tar.gz)』のリンクより直接ダウンロードする。

(https://github.com/julius-speech/julius/releases)

②ダウンロードした「julius-4.6.tar.gz」を「/home/kadai」フォルダ内に移動して、

と、コマンドを実行して、「/home/kadai」フォルダ内に、直接解凍する。

```
tar xvzf Julius-4.6.tar.gz
```

③

```
cd /home/kadai/Julius-4.6
```

を実行して解凍したフォルダに移動してから、

```
sudo apt-get install libasound2-dev
libesd0-dev libsndfile1-dev
```

を実行する。

「Y」キーにてインストール後に

```
./configure -with-mictype=alsa
```

を実行することで入れたライブラリーをコンパイルする。

④そして、

```
make
sudo make install
```

を実行して、インストールを行う。

⑤次に、julius-kit を導入するために

```
cd
mkdir julius-kit
cd julius-kit
```

というコマンドを実行して、
「/home/kadai/julius-kit」フォルダを作成する。
⑥下記の URL より「dictation-kit-4.5.zip」をダウンロードして、先ほど作成した「/home/kadai/julius-kit」フォルダに、解凍する。
(https://ja.osdn.net/projects/julius/releases/)

（２）マイクの設定
①マイクの設定を行う。まずは

```
sudo apt-get update
sudo apt-get upgrade
```

を実行してアップデートをする。
②マイクを USB ポートに挿入して

```
arecord -l
```

を実行し、音声入力デバイスの設定を行うために、割り振られているカード番号やデバイス番号と呼ばれる数字を確認する。

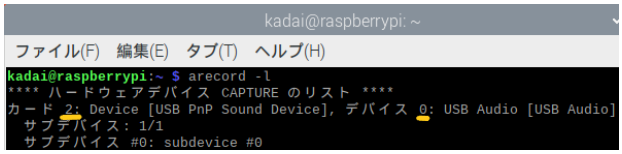

写真５　arecord -l 実行結果

③

```
export ALSADEV=” plughw:2,0”
```

を実行して音声入力デバイスの設定を行う。
「plughw:」の後の数字は、「カード番号、デバイス番号」とする。
④このまま何もしないと再起動時に何度も実行しなければならないので、

```
sudo nano /etc/profile
```

などで、最終行に
「export ALSADEV=” plughw:2,0” 」を追記する。

写真６　/etc/profile の中身

（３）辞書作成
①まず

```
cd/home/kadai/julius-
4.6/gramtools/yomi2voca
```

を実行してフォルダを開く。
②

```
nano /kenkyu.yomi
```

を実行して、図２のようにして単語と読みを空白区切りで書く。

```
開けゴマ　　ひらけごま
J04MJZ　じぇーおーふぉーえむじぇーぜっと
ymd　　わいえむでー
ymd　　わいえむでぃー
1005　　いちぜろぜろごー
1005　　いちれいれいごー
1005　　せんご
```

図２　.yomi の中身

③

```
yomi2voca.pl　　kenkyu.yomi　　>
kenkyu.phone
```

を実行して Julius が読める辞書形式に変換する。「kenkyu.phone」の中身は図２のようになる。

```
開けゴマ　　ｈｉｒａｋｅｇｏｍａ
JO4MJZ ｊｅ: ｏ: ｆｏ: ｅｍｕｊｅ: ｚｅｑ
ｔｏ
ymd　　ｗａｉｅｍｕｄｅ:
ymd　　ｗａｉｅｍｕｄｉ:
1005　　ｉｃｈｉｚｅｒｏｚｅｒｏｇｏ:
1005　　ｉｃｈｉｒｅｉｒｅｉｇｏ:
1005　　ｓｅＮｇｏ
```

図３　.phone の中身

④次に

```
nano kenkyu.grammar
```

を実行して図４のように文法を定義する。

```
S ： NS_B KADAI NS_E
KADAI ： KENKYU
```

図４　.grammar の中身

⑤

```
nano kenkyu.voca
```

を実行して、図５のように記述する。

```
% KENKYU
開けゴマ　　ｈｉｒａｋｅｇｏｍａ
JO4MJZ ｊｅ: ｏ: ｆｏ: ｅｍｕｊｅ: ｚｅｑ
ｔｏ
ymd　　ｗａｉｅｍｕｄｅ:
ymd　　ｗａｉｅｍｕｄｉ:
1005　　ｉｃｈｉｚｅｒｏｚｅｒｏｇｏ:
1005　　ｉｃｈｉｒｅｉｒｅｉｇｏ:
1005　　ｓｅＮｇｏ
% NS_B
[s] silB
% NS_E
[s] silE
```

図５　.voca の中身（先ほど記述した.phone ファイルに似ている）

⑥

```
mkdfa.pl kenkyu
```

と実行して、Julius が読み込める形式に変換する。

⑦この時、3 つのファイルが生成されるが、そのままだとエラーをはいてしまうので「kadai.dfatmp」を「kadai.dfa」に変換する。

⑧そして、

```
cd　　/home/kadai/julius-4.6/gramtools/mkdfa
mkdfa.pl　　/home/kadai/julius-4.6/gramtools/yomi2voca/kenkyu
```

を実行してコンパイルする。

（４）プログラムを作成・実行

①

```
/home/kadai/juliustest.py
```

という空のファイルを作成して、図のような Python コードを入力する。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import socket
import string
import saabo90
import saabo0

host = 'localhost'
port = 10500

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((host, port))

data = ""
while True:
```

```
        while (data.find("¥n.") == -1):
                data = data + str(sock.recv(1024), 'utf-8')

        strTemp = ""
        for line in data.split('¥n'):
                index = line.find('WORD="')

                if index != -1:
                        line = line[index + 6:line.find('"', index + 6)]
                        if line != "[s]":
                                strTemp = strTemp + line

        if strTemp != "":
                print(" 結 果 :" + strTemp)
        if strTemp == "1005":
                saabo0.test()
        if strTemp == "ymd":
                saabo0.test()
        if strTemp == "J04MJZ":
                saabo0.test()
        if strTemp == "開けゴマ":
                saabo90.test()

        data = ""
```

図６　juliustest.py の中身

②保存した後で

```
Julius -C /home/kadai/julius-kit/dictation-kit-4.5/am-gmm.jconf -nostrip -input mic -gram /home/kadai/julius-4.6/gramtools/yomi2voca/kenkyu -module
```

を実行する。

③さらにもう一つ「LXTerminal」を開いて、

```
python3 /home/kadai/juliustest.py
```

を実行することで音声認識プログラムが動作する。

写真７　音声認識の結果

６.サーボモータによる箱の鍵かけについて

箱の開閉に鍵をかける部分をサーボモータで制御する。

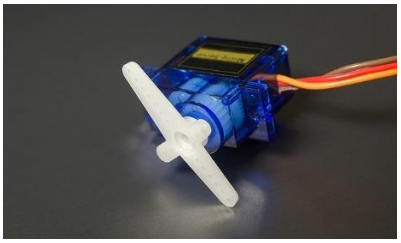

写真８　サーボモータ

このサーボモータを０°～９０°に動かすことで鍵をかける。９０°動かすプログラムを以下に示す。

（１）ソースコード

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)

GPIO.setup(18,GPIO.OUT)

def test():
        p = GPIO.PWM(18,50)
        p.start(0,0)

#def test():
        p.ChangeDutyCycle(6)
        time.sleep(1.0)

if __name__ == ‘__main__’:
         test()
```

「解説」

関数ｐを使用。

８行目、PWM 信号を出力するピン番号を 18 番、周波数を 50Hz に設定。

10 行目、信号を出力。ｐにデューティー比(0.0)を代入。つまり動かない。

13 行目､ChangDutyCycle 関数を使用すると途中でデューティー比を(6)、つまり 90° に動くように変えて動かす。-90° に動くようにするにはデューティー比を(2.5)にする。

（２）用語の解説

①PWM（Pulse Width Modulation）とは，パルス幅を変えることで，ドライバーなどの素子に流れる電流の時間を変化させ，ヒーターやモーターを制御する信号。

②デューティー比とは周期的な現象において､ある期間に占めるその期間で現象が継続される期間の割合。t=信号(関数)がゼロでない期間。T=信号(関数)の期間､周期。

デューティー比 D=t/T

７.研究のまとめ

今回の課題研究を通して、音声認識と顔認識の仕組みと作り方について学ぶことができた。また、Raspberry Pi でサーボモータや人感センサを使った電子工作の方法について理解できた。限られた時間の中で理想通りの作品を作ることはとても難しかった。しかし、メンバーと意見を交わしたり失敗しても次の打開策を考えて、何とか満足度の高いセキュリティボックスを製作することができた。今後は、認識精度を上げたり指紋認証など新しいものを追加するなど、より厳重なセキュリティボックスの作成に挑戦してみたい。

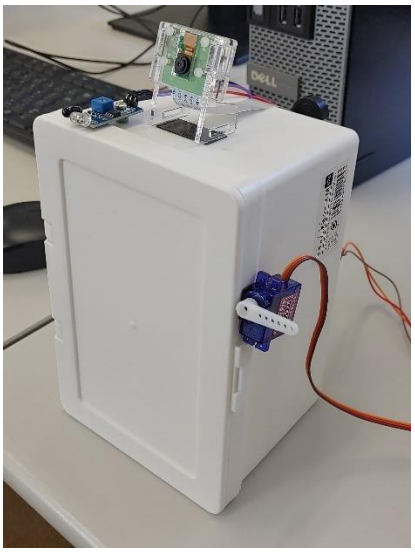

写真９　完成