Raspberry Pi による音声認識制御の製作

松林晃，丸川龍二
水口鳳太朗

1. 研究概要

音声認識による制御装置の作成を行いRaspberry Piを用いて学ぶとともにLinux OSについての理解を深める。

２．研究の具体的内容

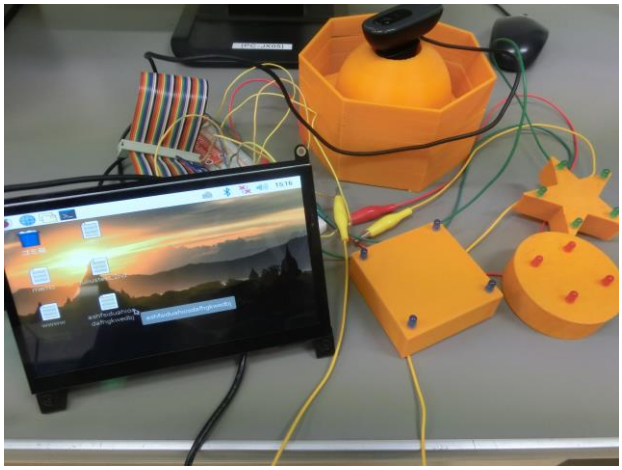

図１　完成図

（１）OSのインストール

あらかじめ学校の PC 上にフォルダを作っておき，その中にOSをインストールする。それをSDカードの中に入れ，Raspberry Piに差し込みOSを導入した。今回の課題研究ではNoobsを使用した。

（２）Raspberry Piの環境設定をする

初めに，使用するソフトウェアのダウンロードを行い，プログラムを実行するためにネットワーク設定をした。

まずセキュリティの強化のためSSHの有効化を行った。

```
$ sudo raspi-config
```

上のコマンドを実行し，コンフィグのメニューを開きSSHを有効にする。

図２　SSHの有効化完了画面

(a)有線LANの設定

ラズパイをインターネットに接続するために，dhcpcd.cofに学校のIPアドレスを打ち込んだ。

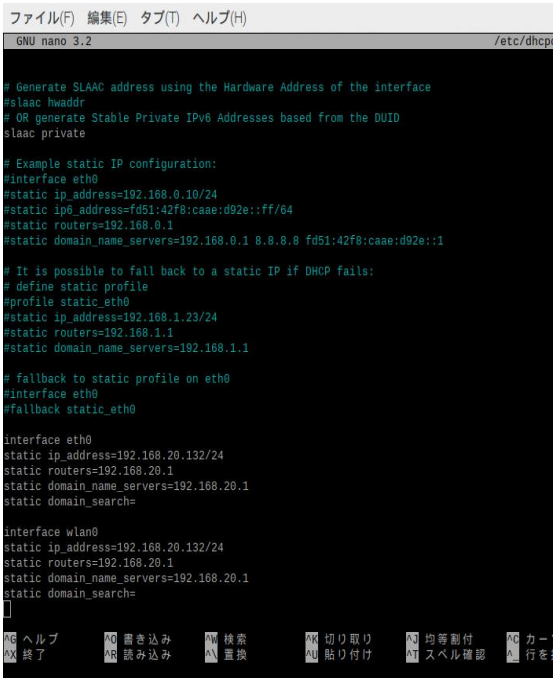

図３　IPアドレスを書き込んだ画面

(b)マイクの設定

まずマイクの優先順位を確認するために以下のコマンドを入力した。

```
$ cat /proc/asound/modules
```

上記のファイルを作成しUSBマイクのindexの値を0に設定することで最も優先順位を高くすることができる。

（3）Juliusのインストール

音声認識をしていくうえで一番重要となる音声認識エンジンをインストールしていく。まずJuliusで使用するモジュールを起動時に毎回起動するように設定する。

```
$ sudo sh -c "echo snd-pcm-oss
>> /etc/modules"
```

次にJulius本体のインストールを行う。今回はバージョン4.4.2.1を使用した。

```
jpi@raspberrypi:~ $julius -version
JuliusLib rev.4.4.2.1 (fast)

Engine specification:
 -  Base setup   : fast
 -  Supported LM : DFA, N-gram, Word
 -  Extension    : LibSndFile
 -  Compiled by  : gcc -g -O2

Library configuration: version 4.4.2.1
 - Audio input
    primary A/D-in driver   : alsa (Advanced Linux Sound Architecture)
    available drivers       : alsa
    wavefile formats        : various formats by libsndfile ver.1
    max. length of an input : 320000 samples, 150 words
 - Language Model
    class N-gram support    : yes
    MBR weight support      : yes
    word id unit            : short (2 bytes)
 - Acoustic Model
    multi-path treatment    : autodetect
 - External library
    file decompression by   : zlib library
 - Process hangling
    fork on adinnet input   : no
 - built-in SIMD instruction set for DNN

    NONE AVAILABLE, DNN computation may be too slow!

Try `-help' for more information.
```

図4　Juliusのバージョン

最後に記述文法音声認識キットのダウンロードを行う。音声認識には汎用的なディクテーションキットと文法を定義して使う記述文法音声認識キットの2つがあり今回は後者を使用した。

記述文法音声認識は自分で辞書と文法ファイルを作成して，音声認識を行う。これにより，認識精度を上げることができ，処理速度も速くなるのでこちらのキットを使用することにした。

（4）「語彙」，「読み」，「音素」，「構文」，のファイルを作成し，辞書ファイルの中に格納する

単純な単語を認識できる辞書を作る。構文の設定次第では複雑な文章を認識することも可能ではあるが，今回はLEDライトの点灯と消灯と点滅だけの制御なので，比較的単純な文章を認識させることになる。

まずは辞書に登録したい語彙に対して，みの情報を定義した読みファイルを
作成する。

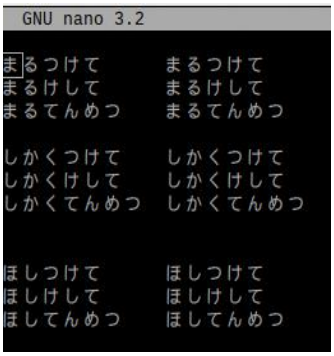

図5　作成した読みファイル

次にJuliusに含まれているスクリプト（yomi2voca.pl）を使用し，読みファイルから音素ファイルを作成する。

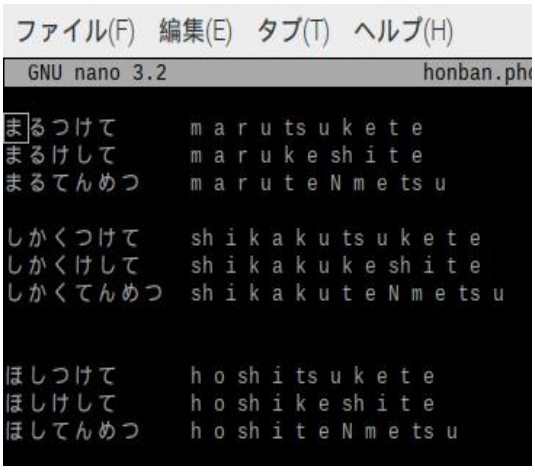

図6　音素ファイル

音素ファイルが作成できたら構文定義を設定した構文ファイルを作成する。設定する文字列は，音素ファイルで出力された文字列で文章を組み立てる。

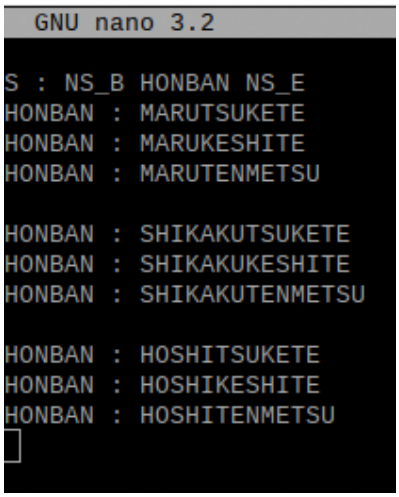

図７　構文ファイル

次に語彙ファイルを作成する。
語彙ファイルも同様に作成した音素ファイルを使って作成した。

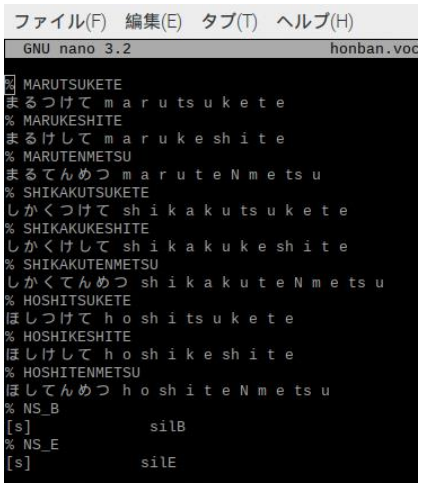

図８　語彙ファイル

最後に音素ファイル，構文ファイル，語彙ファイルと Julius のスクリプト(mkdfa.pl)を使用して辞書ファイルを作成する。

（５）出力先の配線を行う

今回の課題研究では LED ライトは，3D プリンターで作成した円柱と四角柱と星形の柱に穴をあけてそこにはめ込むので，柱の内側にはんだ付けで配線をした。

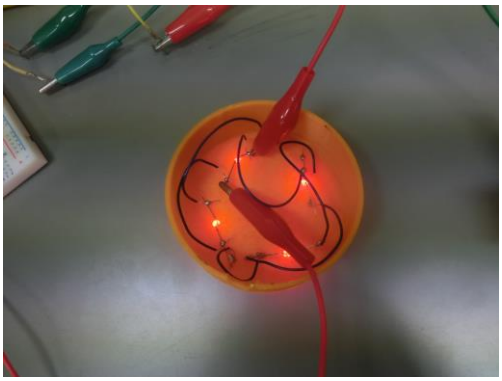

図９　円柱の台の配線

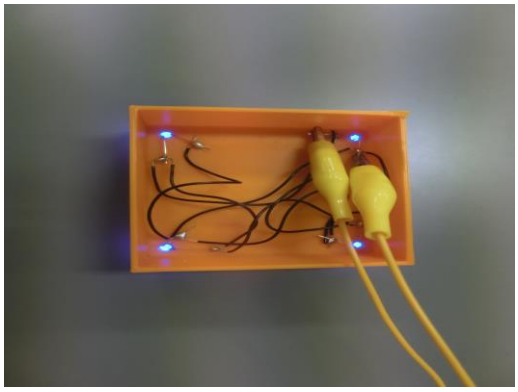

図１０　四角柱の台の配線

図１１　星の柱の台の配線

３．研究を終えての感想

この課題研究で私たちは，Raspberry Pi を使うことによって音声認識の制御装置を作ることができ音声認識についての関心を深めることができた。

また Linux OS を使ったプログラミングをすることによって Linux に関する知識を深めることができた。

Raspberry Pi をここまでしっかりと使ったのも Linux OS を使ったプログラミングもこの課題研究が初めてだったので，インタ

ーネットで色々なサイトを見て学びながら作業を進めていた。

そのため，他のグループと比べ作業の進行が遅くなってしまう問題があった。

しかし，三人のそれぞれ得意な作業の分野で基本的なプログラミングや 3D プリンターでの LED ライトの台の作成，コードとライトのはんだ付け，発表などで使うワードやパワーポイントの作成などの作業をそれぞれ得意な作業を分担して個々で進めることによって作業効率を上げ元々の進行の遅さをカバーし，スムーズに進められるように努力した。

私たちは音声認識に関する知識だけでなく，グループ内での連携の取り方を学ぶことが出来たのではないかと思う。

４．　使用機器

・Raspberry Pi 3B
・マイク(Logicool)
・3D プリンター(XYZ printing pro)

５．　参考文献

・Julius の独自辞書を使って音声を認識させる
https://qiita.com/mininobu/items/322a49856e6665bc8a30

・ラズパイと音声認識でＬチカ
https://qiita.com/mooriii/items/6a16663ce9a80b2e2b92

・Raspberry Pi 3 (RASPBIAN JESSIE) SSH・ネットワーク設定・ユーザー変更
https://hirazakura.hatenablog.com/entry/raspberrypi/setup/last