技術・研究報告

電子工作キットを利用した音声認識ロボットの製作

福庭隆弘[a], 北村達也[b]

[a] 甲南大学大学院 自然科学研究科 情報システム工学専攻
神戸市東灘区岡本８－９－１, 658-8501
[b] 甲南大学 知能情報学部 知能情報学科
神戸市東灘区岡本８－９－１, 658-8501

（受理日 2010 年 5 月 17 日）

概 要

オープンソースの音声認識エンジン Julius と USB 制御キット EZUSB-FX2 を用いてリモコンロボット (電子工作キット) を制御するシステムを開発した．この音声認識ロボットでは,「進め」や「止まれ」などの単語音声を Julius で認識し，その結果に基づいてロボットの 2 つのモーターを制御することによって前後左右への移動を実現する．EZUSB-FX2 は音声認識の結果に基づいてモーターへの電流の On/Off 及び方向を制御する役割を担っている．Julius で実現されている不特定話者を対象とした高速な音声認識によって，このロボットは音声指令に対して正確かつほぼ実時間で動作し，音声認識技術に馴染みのない人々へその可能性を示す格好のデモンストレーションになっている．

キーワード: 音声認識，Julius, USB 制御キット, EZUSB-FX2，リモコンロボット

1 はじめに

長年の研究により，ノイズや歪みのない音声に対する音声認識技術はすでに実用レベルに達している．また，初期の研究は特定話者が発話した単語音声を対象にしていたが，現在では不特定話者を対象とした大語彙連続 (文章) 音声認識が可能になっている．さらに，その処理時間も短縮され，コンピュータの性能向上と相まって実時間に近い処理の実現に至っている．このような背景に基づいて，携帯電話を対象とした音声認識技術及び自動翻訳技術の商用利用も始まっている．とはいえ，一般の人々にとっては音声認識技術はまだ身近なものとはいえないのが現状である．そこで，本研究では，この技術を目に見える形でわかりやすく示すことができる音声認識ロボットの開発を行う．

現代の音声認識技術は確率統計モデルの一種である隠れマルコフモデル (Hidden Markov Model, HMM) に基づいている．これは，時系列信号の確率モデルであり, 複数の定常信号源の間を遷移することで, 非定常な時系列信号をモデル化するものである [1]. Julius はこの隠れマルコフモデルに基づいて音声認識を行うオープンソースの汎用大語彙連続音声認識エンジンである [2]．本研究で開発するシステムでは，この Julius を利用してロボットへの音声指令を認識する．

2 使用ソフトウェア及びキット

2.1 Julius

Julius は，名古屋工業大学徳田, 李研究室と京都大学河原研究室を中心に研究, 開発が行われているオープンソースの汎用大語彙連続音声認識エンジン, ソフトウェアである．1996 年に探索アルゴリズムを原点として開発をスタートし, 現在も持続的に新機能や安定性を追求し, バージョンアップを繰り返している. Julius は標準キットを利用すると数万語の語彙を対象とした文章発声の認識を行う能力がある.

Julius の特徴は汎用性と可搬性である. 汎用性は, 発音辞書や言語モデルや音響モデルなどの音声認識の各モジュールの組み替えが可能な点である. 商用のソフトウェアでは音響モデルや言語モデルが一体化されているのに対し, Julius では音響モデルや言語モデルは完全に独立している. 従って, これらのモジュールを差し替えることにより様々な入力環境やタスク, 言語に適用することができる。また, 各モジュールとインターフェースがオープンで, 誰でもモジュールを作成することができる. もう一つの特徴は, 様々なプラットフォームで動作する可搬性である. 実際に, Linux, Windows, MacOS などで動作が確認されている.

2.2 EZUSB-FX2

EZ-USB は, Cypress 社が製造している USB 制御用のマイコンチップである. 本研究で使用した EZ-USB の特徴は, Cypress 社が提供する動作クロック 24 MHz の CY7C68013 が採用されている点である. また, このチップ内の CPU は, 8051 (Intel 製 8 ビット CPU) を拡張したものが搭載されている. このマイコンチップを中心に, USB 端子などの最小限の機能をセットにしたものが Strawberry Linux 社の EZUSB- FX2 マイコンボード Ver. 2 (図 1) [3] であり, 本研究はこれを利用した. EZUSB-FX2 の仕様を表 1 に示す.

このマイコンボードは I/O ポートを有し, USB ケーブルを介した PC とのデータのやりとりや外部機器への電源供給が可能である. ポートの制御はマイコンボードのメモリに書き込まれるファームウェアにより行われる. PC 上のプログラムからこのファームウェアにデータを送ることによって I/O ポートを制御することができる. ファームウェアは書き換え可能であるため, EZUSB-FX2 が 1 つあれば様々な外部機器と組み合わせることができる.

2.3 ロボクラフト

リモコンロボットとしてタミヤ製のロボクラフト 2 チャンネルリモコン・インセクト「昆虫」(図 2) を利用した. この製品は, プラモデルのように自身でロボットを組み立てる電子工作キットである. このロボットは, 6 足の昆虫の形をしており, 2 つのモータで歩行する. 1 つのモータにつき 3 足が連動して動作する. 本体の全長は 116 mm である.

ロボットはケーブルによって電池 2 個が入ったコントローラとつながっており, コントローラ上にある 2 つのレバーを操作することでロボットを前後左右に移動させることができる. モータとコント

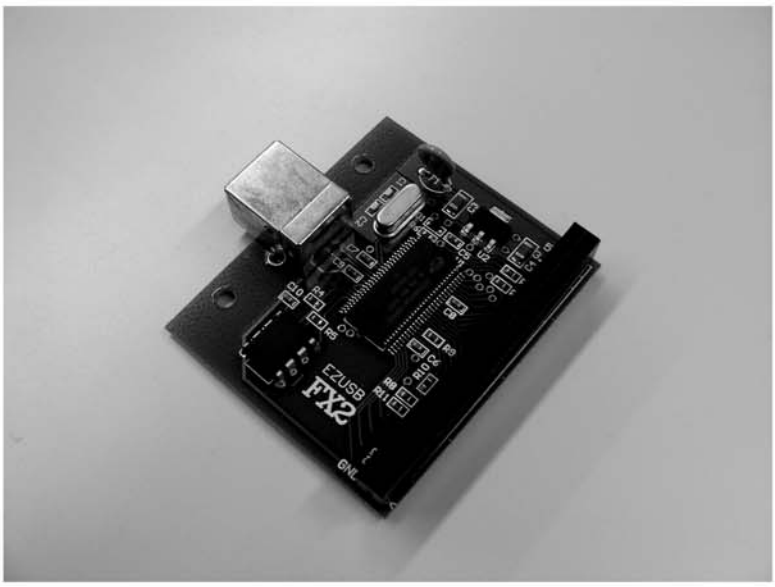

図 1: Strawberry Linux 製 EZUSB-FX2 マイコンボード

表 1: EZUSB-FX2 (CY7C68013) の仕様

項目	内容
RAM 容量	約 8.5 k バイト
エンドポイント用バッファ容量	64 バイト× 3 個, 1024 バイト× 4 個
発振子の周波数	約 24 MHz (マイコン内部は 48 MHz で動作)
動作電圧	約 3.3 V
I/O ポート	PA0〜7, PB0〜7, PD0〜7 (計 24 個 56 pin タイプの場合)
その他の機能	シリアルポート 2 個, タイマ 3 個, 割り込みソース 13 種

ローラのレバーは 1 対 1 で対応しており, レバー操作によってモータの On/Off と回転方向を制御する. 両方のレバーを同じ方向へ動かせばロボットが前進, 後退し, 左のレバーのみを動かせば右方向へ, 右のレバーのみを動かせば左方向へ方向転換する仕組みとなっている.

3 音声認識ロボットの製作及び評価

3.1 全体構成

本システムの全体構成を図 3 に示す. 本研究では製作を簡単にするため, すべて有線の接続によって構成した. PC と EZUSB-FX2 は USB ケーブルによって接続した. EZUSB-FX2 には PC からの指令を受けてモータ制御を行うための電子回路ユニットがピンヘッダを介して接続されている. その電子回

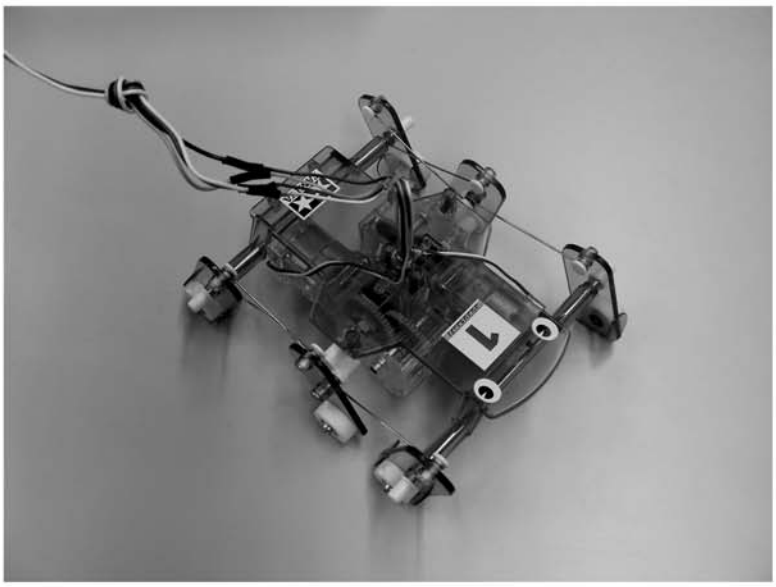

図 2: タミヤ製ロボクラフト 2 チャンネルリモコン・インセクト「昆虫」

路ユニットの出力はロボットのモータと接続されている. 電子回路ユニット, EZUSB-FX2, ロボクラフトを含む部品を表 2 に示す. これらの部品にかかった費用は 8000 円程度であった. PC には Julius がインストールされており, 音声入力のためのマイクロフォンが接続されている. この PC は, CPU が Intel Core 2 Duo E6750 (2.66 GHz) , メモリ 2 GB で構成されている.

システムの実行においては, はじめに PC から EZUSB-FX2 のメモリにファームウェアを転送する. EZUSB-FX2 は PC の電源を切るたびにメモリ内の情報が消去される仕様になっているため, 毎回この手続きが必要である. 次に Julius を起動し, マイクロフォンからの音声入力を待つ状態にする. 音声入力があれば Julius が音声を認識し, 用意した命令セットと照合し, 命令セットに対応する引数を EZUSB-FX2 に送信する. すると, EZUSB-FX2 のファームウェアが引数を受け取り, ポートの電圧制御を行うことによって電子回路ユニット上のリレーを On/Off し, モータを制御する. 命令セットは, 2 つのモータの On/Off, 回転方向を組み合わせてできる 7 種類である. モータの回転方向とロボットの動きに関する対応関係を表 3 に示す. ロボットの頭の方向へモータが回転することを順回転と呼ぶとすると, 両方のモータが停止, 順回転, 逆回転, 片方のモータのみ順回転, 逆回転の命令セットとなる.

3.2 Julius の変更及び設定

本システムは 7 種類程度の命令セットしか持たないため音声認識に利用する単語が非常に少ない. また, 文章単位の認識をせずに単語単位で音声認識を行う. そこで, 軽量化版の Julius である julius-simple.c (Julius バージョン 4.0 に同梱) に手を加えて利用した.

プログラムのコンパイルは PC にインストールした MSYS 上の gcc で行った. また, プログラムの実行は Cygwin 上で行った. MSYS と Cygwin は共に Windows 上で動作し, フリーで利用できる UNIX 風

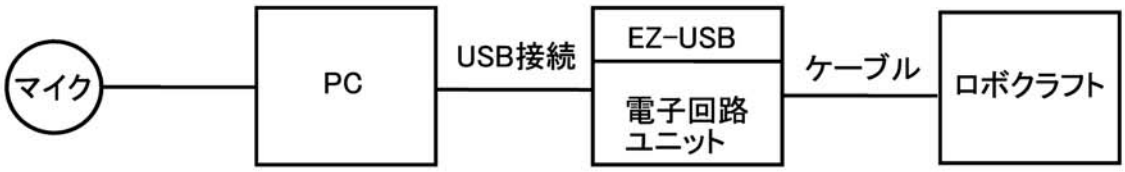

図 3: 本システムの全体構成

表 2: 部品リスト

部品名	規格	個数
USB マイコンボード	EZUSB-FX2 マイコンボード Ver. 2	1
6 足歩行ロボット	ロボクラフト 2 ch リモコン・インセクト「昆虫」	1
ユニバーサル基板	95 × 72 mm	1
リレー	Y14H-1C-3DS	4
抵抗	200 Ω	4
単三乾電池	1.5 V	4
電池ボックス	単三乾電池 1 本用	4

のシェル環境である.

オリジナルの Julius からの変更点は, 大別してメインプログラムにあたる julius-simple.c の変更と辞書ファイル (dict ファイル) の変更の 2 点である.

julius-simple.c は, 後述する辞書ファイル (dict ファイル) 内で指定した認識結果を受け取り, 条件分岐などを経てファームウェアを引数付きで実行するように変更した. 認識結果は, julius-simple.c 内の変数 `winfo->woutput[seq[i]]` に渡される. この変数を条件分岐し, 命令ごとに異なる引数を与えてファームウェアを実行させるように記述した. これらの変更点を付録に示す.

辞書ファイルに関しては, voca ファイルと grammar ファイルを作成し, DFA (Deterministic Finite Automation) ネットワークファイル (dfa ファイル) と dict ファイルに変換した. voca ファイルは単語の表記と発音音素列を登録するもので, grammar ファイルは単語間の接続に関する制約を記述した文法ファイルである. voca ファイルは, 「前」や「右」などのロボクラフトの動作に関する命令単語を登録した. 1 行に単語の表記と発音音素列を順にスペースで区切って記述する. 発音音素列は, Julius の公式 Web ページ [4] に掲載されている音素リストに従った. 例えば,「うしろ」という音声入力に対して標準出力に「後ろ」と出力するためには,

```
後ろ  u sh i r o
```

と記述する.

grammar ファイルは前述の通り, 単語間の接続に関する制約を記述した文法ファイルである. 本研究では単語間の接続を必要としないため, 複数の単語から成る文法を記述しなかった. 従って, grammar ファイルには以下に示すコマンド 1 行のみを記述した.

表 3: モータの回転方向とロボットの動き

左モータ	右モータ	ロボットの動き
停止	停止	停止
停止	順回転	左回転
順回転	停止	右回転
順回転	順回転	前進
停止	逆回転	右回転
逆回転	停止	左回転
逆回転	逆回転	後退

```
S : NS_B COMMAND NS_E
```

S, NS_B, COMMAND, NS_E はそれぞれ開始記号, 文頭の無音区間, 単語, 文末の無音区間に対応する.

作成した voca ファイルと grammar ファイルの 2 つを文法コンパイラ mkdfa を用いて dfa ファイルと dict ファイルに変換した. また, 様々な言い回しに対応するために 1 パターンのモータの制御に対して「前進」,「前」,「進め」等の複数の単語を dict ファイルに登録した.

Julius を実行するには, 使用するモジュール等を設定ファイル (jconf ファイル) に従って引数で指定する必要がある. 本研究では, dfa ファイル, dict ファイル, jconf ファイル, 音声入力ソース, 出力方法を指定した. dfa ファイルと dict ファイルは独自で作成したもの, jconf ファイルは Julius のパッケージに含まれる hmm_ptm.jconf を用いた. 音声入力ソースは, マイクロフォン入力なので `-input mic`, 出力方法は逐次的に出力するために `-demo` とした. 以上を引数で指定すると次のようなコマンドとなる. このコマンドは, 図 4 に示すディレクトリ, ファイル構成の下, juliu-4.0/にて実行することを想定している.

```
./julius-simple/julius-simple -dfa SampleGrammars/tamiya/tamiya.dfa
 -v SampleGrammars/tamiya/tamiya.dict -C hmm_ptm.jconf -input mic -demo
```

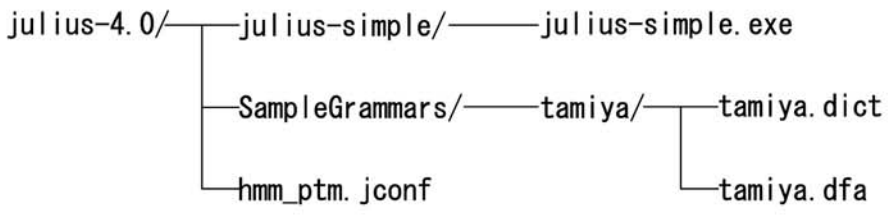

図 4: ディレクトリとファイル構造

3.3 EZ-USB 上のプログラム

EZ-USB のファームウェアの開発環境には, Cypress 社が提供する Visual Studio 風の keil μvison2 (図 5) を用いた. この開発環境にてビルドを行うことによって hex ファイル (ファームウェア) が生成される. この hex ファイルを EZ-USB Control Panel (図 6) にて EZ-USB に転送した.

ファームウェアの作成は文献 [5], [6], [7] を参考にして bulkloop.U2v という bulk 転送用のサンプルのプロジェクトファイルを変更した. bulk 転送とは, 非周期的に大量のデータを扱う転送方式で, プリンタやスキャナなどに利用される転送方法である. 変更点は, julius-simple.c から渡される引数に従ってコネクタのポートの On/Off を制御するようにした点である. julius-simple.c から渡される引数は bulkloop.U2v に関連付けられている bulkloop.c の `EP2FIFOBUF[0]` という変数に格納される. この変数を利用して命令別に処理を分岐し, 変数 `IOA` に値を与えて 4 つのポート (PA0, PA1, PA2, PA3) を制御した. 変数 `IOA` に規則に従って 16 進数を代入することによって 4 つのポートの On/Off を決定する. この規則は, 以下のようになっている. PA0 から PA3 までを昇順に 2 進数の各桁とみなし, 電流を流すポートを 1, 流さないポートを 0 とする. できあがった 4 桁の 2 進数を 16 進数表記に変換する. 例えば, PA2 と PA3 のみを ON にする場合は 2 進数表示で 1100, 16 進数表示で C となるのでこれを以下のように代入する.

```
IOA = 0x0c;
```

bulkloop.c は付録に示す. さらに, julius-simple.c から引数を受け取り, ファームウェアに送信するためのプログラム bulkloopctrl.c を作成した. このプログラムは Cygwin 上の gcc でコンパイルを行った. bulkloopctrl.c は付録に示す.

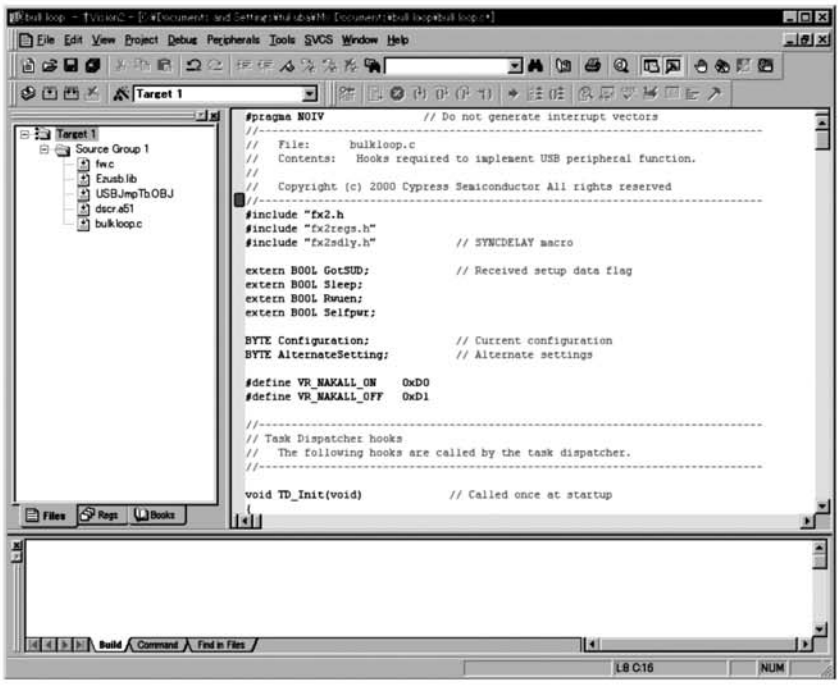

図 5: keil μvision2

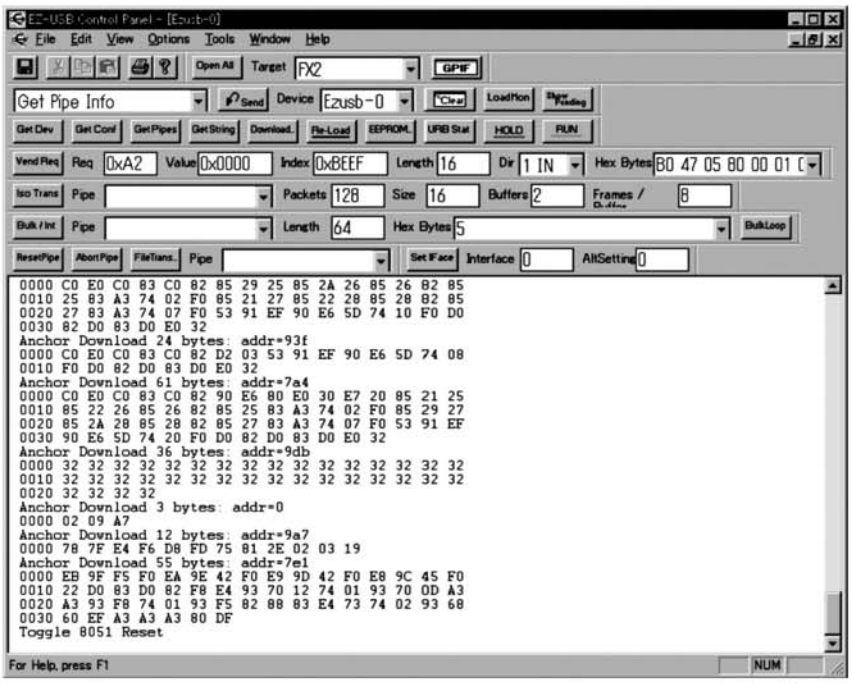

図 6: EZ-USB Control Panel

3.4 電子回路ユニット

EZUSB-FX2 のコネクタ制御によってコネクタに流れる電流を 7 種類のモータ制御に変換するために図 7 の電子回路ユニットを製作した. ユニバーサル基盤上にリレー, 抵抗をハンダ付けし, リレーからケーブルを延ばして単三乾電池及びモータに接続した. この電子回路ユニットと EZUSB-FX2 のコネクタの両方にピンヘッダを挿入することで 2 つを接続した. EZUSB-FX2 のコネクタ配置を図 8 に示す. コネクタの PA0 から PA3 の 4 ポート, GND, 5 V のポートを利用して, 4 つのリレーのスイッチングを行った. リレーのスイッチングにより電池とモータが接続され, モータが回転する. また, 選択的にリレーをスイッチングすることによって電流の向きを制御し, それによってモータの回転方向を制御する. リレーから単三乾電池, ロボクラフトのモータへと接続し, 順回転, 逆回転の両方ができるように配線を行った. このとき, リレーのスイッチングは EZUSB-FX2 からの電源供給で行った.

3.5 評価

完成したシステムに音声を入力すると, ほとんどタイムラグなくロボットが動作を開始した. また, 不特定話者の音声であっても正確に認識できる. これは命令セットの少なさも寄与していると考えられる. ただし, 子供の音声に対しては認識精度が低い. これは, Julius に用意されている音響モデルが成人の音声から作られたものだからであろう. 経験的には明瞭な発音を心がけることによってより精度が向上する. その他, PC のマイク入力音量の適切な設定, 利用者の声量などが認識精度に影響する. 本システムでは, 移動速度の遅い 6 足歩行ロボットを採用したため, 利用者が音声指令入力を急ぐ必要も

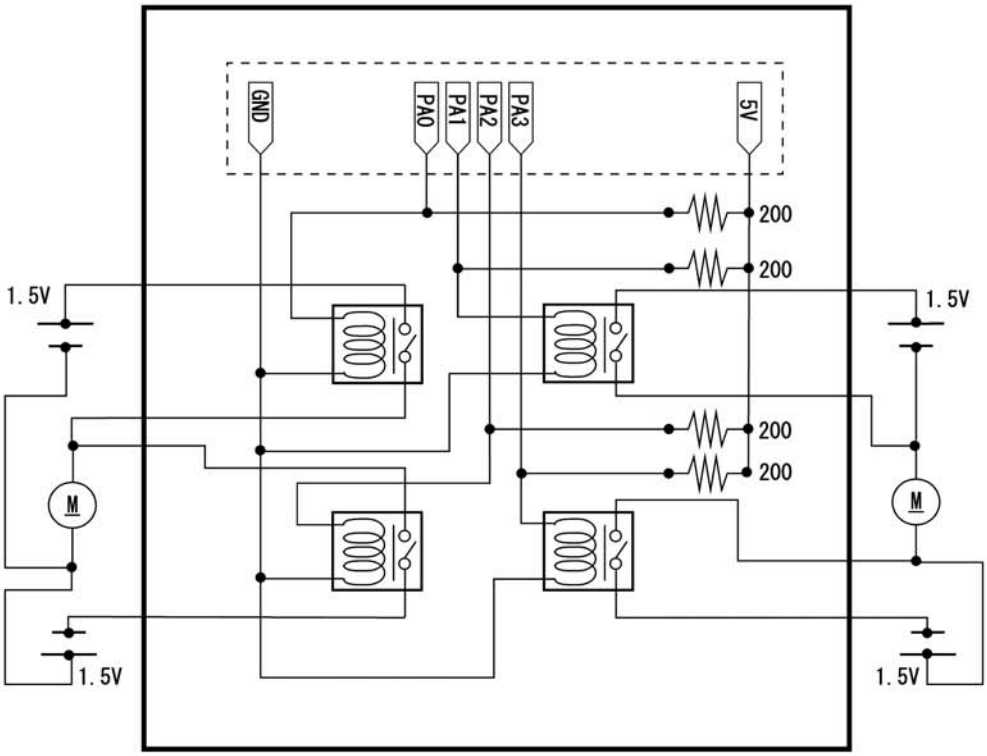

図 7: 電子回路ユニットの回路図

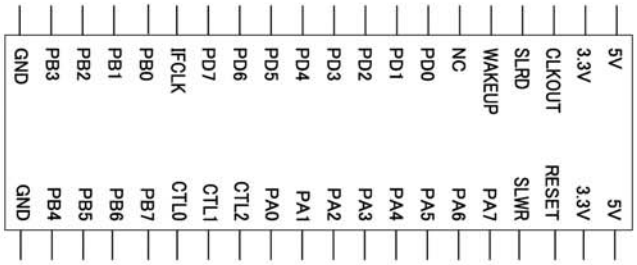

図 8: EZUSB-FX2 のコネクタ配置

なく, また多少の誤認識があっても問題が生じないという利点もある.

4 おわりに

オープンソースの音声認識エンジン Julius, USB 制御キット EZUSB-FX2, 電子工作キットを組み合わせて音声認識ロボットを開発した. このロボットは, 本学オープンキャンパス等を通じて音声認識の技術を広く一般の方々に体験してもらうことに役立っている. さらに, 本学部において学生が身につけられるスキルに関して, 1 つの具体的イメージを高校生及びその保護者等に持ってもらうことにも役立っていると考えている.

参考文献

[1] 鹿野清弘, 伊藤克旦, 河原達也, 武田一哉, 山本幹雄, IT Text 音声認識システム. オーム社, 2001.

[2] 河原達也, 李晃伸, “連続音声認識ソフトウエア Julius,” 人工知能学会誌, vol. 20, no. 1, pp. 41–49, 2005.

[3] 松原拓也, EZ-USB 電子工作入門 Visual Basic でラクラク USB コントロール. ソフトバンククリエイティブ, 2006.

[4] 大語彙連続音声認識エンジン Julius, http://julius.sourceforge.jp/

[5] 松原拓也, “パソコンを使ったマイコン制御 自分だけの USB 機器を作ろう,” C MAGAZINE, 12 月号, pp. 62–71, 2005.

[6] 松原拓也, “自分だけの USB 機器を作ろう! パート 2 EZ-USB による「USB ラジオ」「USB リレー」の製作,” C MAGAZINE, 3 月号, pp. 61–67, 2006.

[7] 松原拓也, “自分だけの USB 機器を作ろう! パート 3 EZ-USB による「USB 温度計」「USB 湿度計」の製作,” C MAGAZINE, 4 月号, pp. 80–83, 2006.

[8] Cypress Semiconductor, http://www.cypress.com/

付録　bulkloopctrl.c, bulkloop.c, julius-simple.c

本システムの開発にあたり, オリジナルのプログラムに変更を加えた部分を以下に示す. bulkloopctrl.c のオリジナルは C MAGAZINE [6] である. bulkloop.c のオリジナルは, Cypress 公式 Web ページ [8] のサンプルプログラムで C MAGAZINE [5], [7] を参考にして変更した. julius-simple.c のオリジナルは, Julius-4.0 のパッケージ同梱されているものである.

```
/* bulkloopctrl.c */

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <winioctl.h>
#include "ezusbsys.h"

#define OUTPACKETSIZE 1
#define OUTPIPE 0

int main(int argc, char *argv[])
{
  HANDLE hDevice;
  BOOLEAN bResult;
  DWORD nBytes = 0;
  UCHAR outBuffer[OUTPACKETSIZE];
  BULK_TRANSFER_CONTROL bulkControl;

  // Open the device
  hDevice = CreateFile("\\\\.\\Ezusb-0",
       GENERIC_WRITE,
       FILE_SHARE_WRITE,
       NULL,
       OPEN_EXISTING,
       0,
       NULL
       );

  // Check
  if (hDevice == INVALID_HANDLE_VALUE)
    {
      fprintf(stderr, "Failed to execute CreateFile\n");
      return;
    }

  // Send data
  outBuffer[0] = (UCHAR)atoi(argv[1]); // off
  bulkControl.pipeNum = OUTPIPE;

  bResult = DeviceIoControl(hDevice,
    IOCTL_EZUSB_BULK_WRITE,
    &bulkControl,
    sizeof(BULK_TRANSFER_CONTROL),
    outBuffer,
    OUTPACKETSIZE,
```

```
    &nBytes,
    NULL
    );

  // Check
  if (bResult != TRUE)
    {
      fprintf(stderr, "Failed to send data\n");
    }

  // Close the device
  CloseHandle(hDevice);
}


/* bulkloop.c */
void TD_Poll(void) // Called repeatedly while the device is idle
{
  // C Magazine 2006/03, p. 64
  if(!(EP2468STAT & bmEP2EMPTY)){
    // EP2 FIFO is empty
    switch(EP2FIFOBUF[0]){ //TF
      case 10: //stop
          IOA = 0x00;// Output PortA bit0 = 0
          IOB = 0x00;// Output PortB bit0 = 0
          IOD = 0x00;// Output PortD bit0 = 0
          break;

      case 20://go
        IOA = 0x03;
        break;

      case 30://left
        IOA = 0x02;
        break;

      case 40://right
        IOA = 0x01;
        break;

      case 50://left back
        IOA = 0x08;
        break;

      case 60://right back
        IOA = 0x04;
        break;

      case 70://back
        IOA=0x0C;
        break;

      case 80://red LED ON
```

```
      IOD=0x01;
      break;

    case 90://blue LED ON
      IOD=0x02;
      break;

    case 95://red and blue LEDS ON
      IOD=0x03;
      break;

    case 100://portA and portD OFF(only portB ON)
      IOA = 0x00;// Output PortA bit0 = 0
      IOD = 0x00;// Output PortD bit0 = 0
   }
  EP2BCL = 0x80; // re(arm) EP2OUT
  }
}
```

```
/*julius-simple.c*/

for(i=0;i<seqnum;i++) printf(" %s", winfo->woutput[seq[i]]);
  pre=num;
  num=atoi(winfo->woutput[seq[1]]);//文字列を int に変換

  switch(num){
    case 0:
      printf("\n終了します");system(sp);exit(1);
      break;

    case 10:
    case 11:
    case 12:
    case 13:
      printf("\n止まれ");system(sp);
      break;

    case 20:
    case 21:
    case 22:
    case 23:
    case 24:
      printf("\n 前");system(go);system(red);//同時に赤色 LED 点灯
      break;

    case 30:
      printf("\n左");system(le);system(randb);//赤と青 LED 点灯
      break;

    case 40:
      printf("\n右");system(randb);
      system(ri);
```

```
      break;

    case 50:
      printf("\n左後ろ");
      system(lbk);
      break;

    case 60:
      printf("\n右後ろ");
      system(rbk);
        break;

    case 70:
    case 71:
    case 72:
    case 73:
    case 74:
      printf("\n後ろ");system(bk);system(blue);//同時に青色LED点灯
      break;
  }
}
```