修士学位論文

論文題目　画像と音声情報を用いた
自律型ロボットに関する研究

ふりがな　なかごみ　きょうへい
氏　名　中込　恭平

専攻　工学研究科　機械工学専攻

指導教員　金丸　隆志　准教授

修了年月(西暦)　2013 年 3 月

工学院大学大学院

概要

近年，カメラや距離センサ，赤外線センサといった様々なセンサを搭載した自律行動型のロボットや家電が多く開発されており，これらの自律的に行動するロボットに関する分野は今後さらに発展していくと考えられる．本研究では種々のセンサの中でも，物体の色や形状といった多くの情報を取得できるカメラを搭載することによって，我々の生活環境の情報を取得し，その情報によって自律的に行動するロボットを製作できないかと考えた．

またロボットのコントローラとして，近年普及・発展が著しい Android スマートフォンを採用することにより，より手頃かつ安価に自律型ロボットのシステムを構築することを目指した．

対象を追従する方法として，対象物の画像上の位置座標と，目標となる座標の偏差を無くすために PI 制御を用い，対象物の追従制御を行った．

以上の方法を用いることによって，画像処理・音声認識により対象を取得，自律的に追従するロボットの開発を目指した．

1章 序論

1.1 研究背景

1.1.1 ロボットと音声認識・画像処理技術

近年自律型のロボットや車など，多くの自律技術が開発されている．東京工業大学で開発された『人混みでも環境地図を学習して稼働する自律移動ロボット [1]』など，周囲の環境を画像処理などで取得し行動するというものから，1.1.2 で説明するルンバ，COCOROBO と言った産業レベルの物まで様々である．しかし一口に自律と言っても，ほぼ完全に自律行動するロボットもあれば，人とのコミュニケーションを頻繁にとる必要があるロボットもある．産業用ロボットのような決められた動作を繰り返す物ならばコミュニケーションはあまり必要とされない．しかし例えば介護用に作られたロボットならばその動作の複雑さから，より多くの情報を欲するため頻繁にコミュニケーションが必要となる．

その方法としてタッチパネルやマウス，キーボードなど接触型のヒューマンインターフェースが用いられてきた．それは長らく機械が人間の行動を補佐するツールであったためである．しかしコミュニケーションをとる相手が『ツールとしての機械』ではなく『自律した行動をとる機械』となった場合，人間と機械のより適したコミュニケーションの手段は，従来とは異なるものとなるだろう．例えば音声認識技術がそれにあたる．4 章で解説する Google 社の開発した音声認識 API や Apple 社の開発した Siri などがそれにあたり，既に人と機械のコミュニケーションの手段は変わりつつある．

音声認識の利点はジェスチャやボタン操作のようにインプット時の身体の動作を要求せず，ストレスも少ない点が挙げられる．人間同士は対話する際相手との間に最適な空間(パーソナルスペース)をとり，言葉や身振りといった情報によってコミュニケーションをとるが，相手が近すぎると感じれば離れ，見つめあいすぎていると感じれば視線を逸らすという，パーソナルスペースに入り込んだ異分子に無意識に拒否反応を示す．これは人間とコミュニケーションをとるロボットを製作するにあたり考慮すべき点である．自律型ロボットとパーソナルエリアに関する研究は既に行われている．音声認識を用いる事は，自律型ロボットが必要以上に人間のパーソナルスペースへ侵入することを防ぐ事に繋がるだろう．自律型ロボットがより我々の生活空間に適応するためには，パーソナルスペースの様な繊細な問題にも気を使わなければならないだろう．

しかし音声認識だけでは周囲の環境変化に伴った行動ができない．自律型ロボットが行動するには，周囲の環境をリアルタイムに取得しなければならないからである．周辺環境の情報の取得は自律ロボットと人間のコミュニケーションにおいて重要な意味を持ち，任意の対象を自律でロボットに追尾させたい場合『何』に対して追従制御を行うかを決定するための根拠になる．例えばそれは人の体色であり，目や鼻，口と言った特徴であり，あるいは発光している，四角い，丸いといった特徴である．これらの情報をセンシングするためには赤外線センサやレーザーセンサ，輝度センサといった様々なセンサが必要となる．しかし画像処理を行うことで，理想的にはこれらのセンサの多くをカバーできるだろう．他のセンサと複合的な処理を行うこともできる．画像処理を制御に取り入れることにより，自律型ロボットは周囲の環境情報を一元的に取得し，音声認識を取り入れる事で，よりスムーズに人と対話できるようになるだろう．

現在では音声と画像情報を複合的に用いた研究は行われているが，『口唇領域の画像処理による音声認識補助システムの開発 [2]』のように音声をより正確に認識するための補助技術として画像処理が用いられていることが多い．一方画像と音声情報を用いて，より人間に近い形の命令をロボットの研究もされつつある．電気通信大学の『ユーザからの曖昧さを伴った指示に基づく実環境内のオブジェクト探索 [3]』や埼玉大学の『対話と視覚情報を用いた環境情報の取得 [4]』がその例として挙げられる．

ユーザからの曖昧さを伴った指示に基づく環境内のオブジェクト探索は，人間がロボットに『〜を取って』などという命令を出し，画像処理と音声認識によりその命令をロボットが遂行する事を想定した研究である．対話と視覚情報を用いた環境情報の獲得も，同様の想定をしているが，物体の位置情報を検出し，命令に反映することに焦点をあてている．これらの研究は人間がロボットへ音声情報を入力し，その指示に従いロボットが行動するというものである．これらの研究のように，様々な物が存在する生活環境内で動作する自律型ロボットがより適切な行動をとるためには，画像処理と音声認識が必要である．

1.1.2 ロボット家電

ルンバや COCOROBO，コーボルトのようなロボット家電製品が多く開発されている．これらのロボット家電の特徴は，カメラや距離センサなど搭載されたセンサにより様々な環境をセンシングし，自律的に行動する点である．

例えば i-Robot 社製のロボット家電であるルンバ(図 1.1)は掃除能力に特化した製品で, 赤外線センサで様々な障害物を避けて室内を隅々まで能動的に掃除してくれる．しかし赤外線センサはガラスや黒い物体を検知することができないため，これらの障害物はバンパに取り付けられた触覚センサで検知する事になる．また他の多くのロボット家電は乗り越えることができる段差の上限が 1.5cm となっているが，ルンバは 2cm まで乗り越える事ができ，他のロボットよりも走破性が高いと言える．さらにリビングとキッチンが一体化している間取りの部屋で，リビングとキッチンを仮想的に別の部屋と認識するためのヴァーチャルウォール機能が搭載されている．この機能はルンバに付属した赤外線発生装置を境界としたい場所に設置することにより，ルンバが認識する部屋を仮想的に 2 つに隔てる事ができる機能である．また自動で充電のためのクレイドルへ戻るなどの機能も有しており，バッテリーの持つメモリ効果[1]を解消するためのリフレッシュ機能なども搭載している，掃除機としての機能に特化したロボット家電と言えるだろう．

図 1.1　ルンバの外観

[1] 二次電池の充電回数と共に充電容量が減少したように見える現象. 放電が終わらない内に充電を行う事によって発生する．十分に放電させることでこれを解消することができる．

一方 SHARP 社製の COCOROBO(図 1.2)は掃除機能の他にプラズマクラスターや音声認識，無線カメラを搭載しているなどより多くの機能を有しており，スマートフォンからの操作や，スマートフォンの画面上で COCOROBO が撮影している映像を確認することも可能となっている．また COCOROBO は他のロボット家電が搭載している赤外線センサではなく，超音波センサを利用する事によってガラスや黒い物体を認識する事が可能となっている．

図 1.2　COCOROBO の外観

またフォアベルク社製のコーボルト(図 1.3)はルンバや COCOROBO とは異なり，カメラの他にレーザーセンサを搭載しており，秒間に 1800 回の 360°のスキャンを行うことにより周囲の環境を検知する．また壁面に 80cm～100cm の隙間があるとそこをドアとして認識し，一つの部屋の掃除を終えると，ドアを出て次の部屋を掃除するなどの機能を搭載している．

図 1.3　コーボルトの外観

このように自律型ロボットは次世代の家電製品として製品化されているが，掃除機としての機能といったような他の家電の機能を持った物がほとんどである．

本研究ではこのような自律型のロボット家電を，掃除用としてではなく我々の生活環境をセンシングするロボットとして，スマート家電のシステムに取り入れることが出来ないかと考えた．

1.1.3 情報端末の OS　Android[2]

近年はスマートフォンやタブレット PC などの小型情報端末の普及･発展が著しく, 我々の生活圏には以前より小型で高性能な情報端末の見られる機会が多くなった(図 1.4). 本研究ではその中でも，Android という OS を搭載したスマートフォンを採用した.

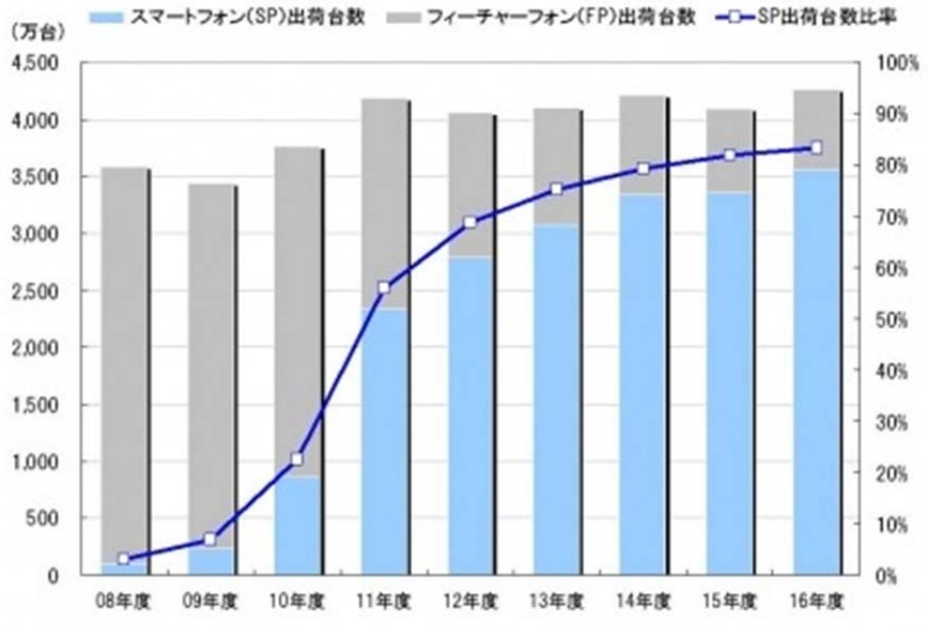

図 1.4　スマートフォン出荷台数の推移・予測 [5]

[2] ハードウェアとアプリケーションを仲介するシステム. アプリケーションがオペレーティングシステムを通しハードウェアに命令を送信することで実際の処理動作が発生する.

Android は Android Phone に搭載された OS であり，スマートフォンに限らず様々なデバイスをサポートしている．図 1.5 はスマートフォンのソフトウェアプラットホーム市場の各シェア率であるが，これを見ても解るように Android は多くの市場を占めていることがわかる．

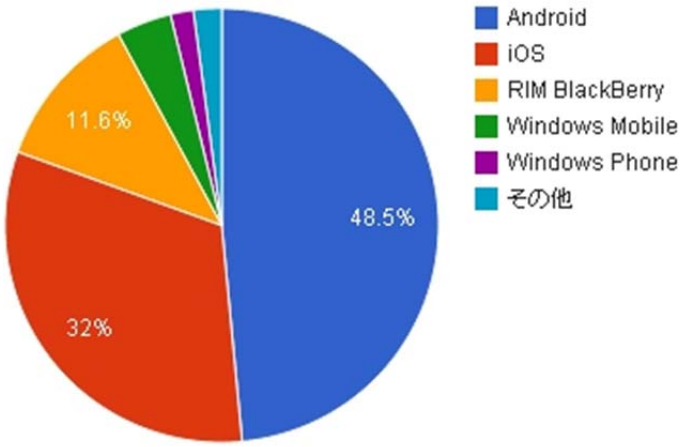

図 1.5　ソフトウェアプラットホームのシェア率 [6]

Android がここまでの発展を見せた要因の一つとして，オープンソースで，さらにアプリケーションの開発を無料で行うことが出来るという点が挙げられる．オープンソースであるため，様々なアーキテクチャ上で動作可能なようにユーザによって手が加えられ，さらにその上で動くアプリケーションの開発も出来る．

この Android を搭載したスマートフォンをプラットホームにすることにより，専用の処理装置が不要になり，次節で紹介するスマートハウスをより手軽に手に入れることが出来る．また開発者にもオープンソースであることや，開発環境が無償で整えられるという利点がある．

1.1.4 スマートハウスとロボット家電

スマートハウスとは個々の家電が持つ情報を相互にやりとりすることによって，ユーザのニーズに合った生活環境の提供，最適化を行うという概念である．スマート家電とスマートハウスは『複数の家電を接続し情報のやり取りを行いつつ制御する』という共通点があるが，近年開発が進んでいるスマート家電についてはスマートフォンと個々の家電の連携という面が強く押し出されている．例えば東芝社製のエアコン X シリーズのように遠隔地からの電源の ON/OFF といった用途が主として取り上げられ，スマートハウスのように種々の家電を統合的に最適化するというアプローチの仕方を主軸とした製品は未だ普及していない．しかし今後，家の環境を家電製品が能動的に取得し，人間の要求に合った制御を自律して行うスマートハウスのような技術が発展するだろうと考えられる．そしてその技術は今までの家電製品だけではなく，ルンバや COCOROBO といった新しいタイプのロボット家電にも適用されるだろう．

例えばロボットが部屋にいる人間を検知し，その人が過ごしやすいような室温を提供するためにエアコンに情報を送り，室内の不要な照明を自動で消灯するなどの技術がそれにあたる．このようなシステムは東芝社が発表している『人と共存するロボット』で示されている．図 1.6 のようにロボットが情報を取得し，それぞれの家電に情報を送信し動作の最適化を行うようなシステムである．このホームロボットと汎用情報端末を，自律走行するロボットとスマートフォンで構築できないかと考えた．

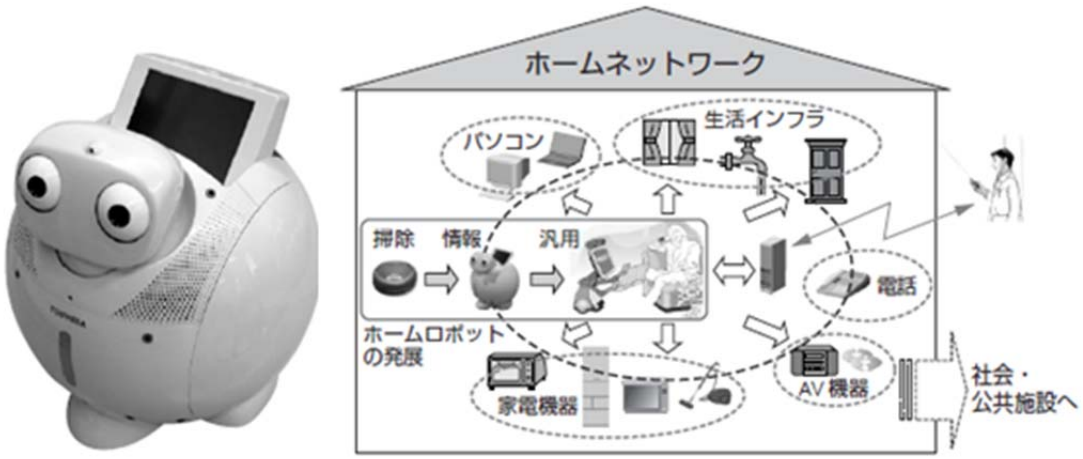

図 1.6　ホームネットワークの要となるホームロボット Apri AttendaTM [7]

1.2 研究目的

1.1 で説明したように，自律型のロボット家電が持つ高性能なセンサと，そのプラットホームとしてスマートフォンを利用することで，我々の生活環境をロボット家電が一括でセンシングし，その他の家電を生活スタイルに合わせてスマートフォンから各家電を制御・最適化するというシステムを提案したい.

よって本論文ではその中でも，スマートフォンとロボットによる生活環境のセンシングと自律制御，特には画像と音声情報を用いた自律型ロボットに焦点をあてている.

1.3 論文の構成

本論文の構成を述べる．まず 1 章，序論では本研究を行うに至った背景，及び目的を説明する．次に 2 章ではロボットを製作するにあたり必要なデバイス，および開発環境を示し，3 章でロボットの駆動部，およびその制御回路について触れる．4 章，及び 5 章ではそのロボットを制御するための音声認識，画像処理で制御するためのカメラや，アプリケーションについて詳細を述べ，6 章ではロボットの動作実験やその考察を記述し，7 章で総括へ至る. このような流れで論文を進めていく.

2章 ロボットの構成と開発環境

研究目的を受けて，本章では画像処理と音声認識によって行動可能なロボットの開発に必要と考えられる構成や，ソフトウェアとその開発環境について説明する．

2.1 ロボットの概要

2.1.1 ロボットの構成

本研究は画像情報と音声認識を用いたロボットに関する研究となっている．この目的を実現するためには，画像情報の入力デバイスとなるカメラ，音声情報の入力デバイスとなるスマートフォンに内蔵されたマイク，それらの情報を処理しロボットを制御する演算処理部，さらにそれを動力としてロボットに伝えるアクチュエータが必要となる．これらの構成を満たす，図 2.1 のような機器を用いてロボットを構築した．

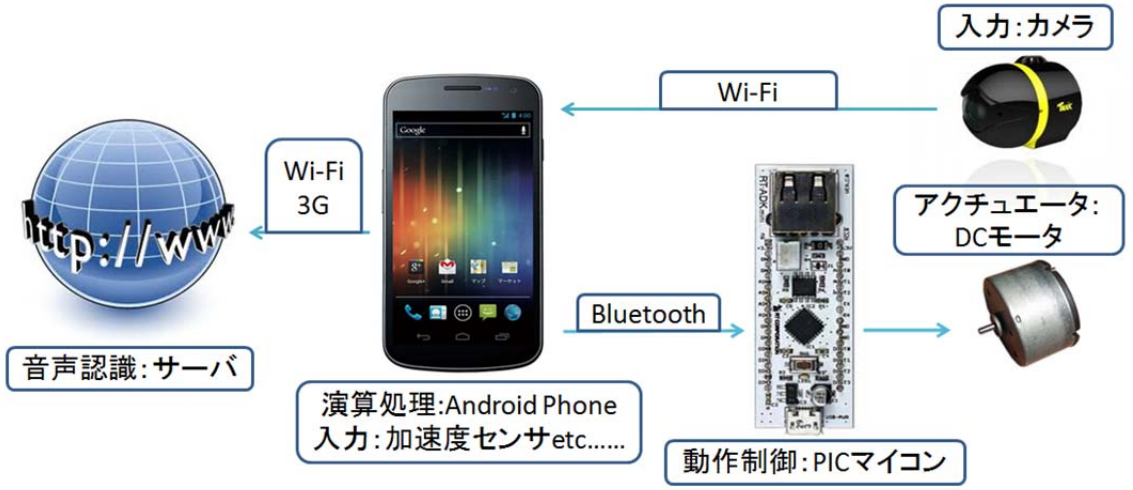

図 2.1　ロボットの構成

Wi-Fi 通信によってカメラから周囲の環境を動画像として取得し，音声認識により『あか』，『四角』，『明るい』などというような目標となる物体の特徴がスマートフォンに入力される．そしてその情報をもとに，画像上でその情報に近い物体を捜索し位置を計算する．その結果算出された DC モータの回転方向や速度の命令を，Bluetooth 通信を介して PIC マイコンに送信する．PIC マイコンはその命令を受信し，制御回路を通して DC モータを駆動する．なお，画像処理や音声認識などはスマートフォンのプログラムであるアプリケーション[3]が行うが，DC モータの制御は PIC マイコンのプログラムであるファームウェア[4]が行う．

---

[3] コンピュータ上でユーザが任意の作業を実施するための機能を持つソフトウェア．本研究ではロボットの動作モードの変更などの機能を提供する．

[4] 電子機器を搭載したハードウェアに任意の動作を行わせるための機能を持つソフトウェア．本研究ではロボットを動作させるための PIC 用ソフトウェアを指す．

2.1.2 ロボットの外観と詳細

図 2.2 がロボットの外観である．土佐電子社製のオムニキットを駆動部として，その上にカメラ，回路，バッテリーが搭載されている．ハードウェアやソフトウェアなどの構成要素については 3 章，4 章，5 章にて解説しているので，ここではロボットの大きさなどの詳細を表　2-1 に紹介する．

図 2.2　Wi-Fi カメラ搭載，三輪ロボット

表　2-1 ロボットの詳細

総重量	760g
全幅×奥行き×全高	220×220×90mm
連続動作時間	3 時間

2.2 ソフトウェアの開発環境

本研究のロボットのソフトウェアは，前章で述べたように PIC マイコンのファームウェアとスマートフォンのアプリケーションの 2 つで構成されている．スマートフォンは DC モータやセンサ類などといった，外部デバイスを直接制御することをサポートしておらず，これを行う為の制御回路が別途必要であることが多い．ファームウェアとアプリケーションはそれぞれ C 言語と Java 言語で記述する．

2.2.1 PIC ファームウェアの開発環境　MPLAB

ロボットのファームウェアの開発に使用した MPLAB という IDE[5]について説明する．PIC マイコンはロボットの駆動制御を行うために用いるマイクロチップ社製のマイクロコンピュータである．PIC マイコンはアセンブラ，あるいは C 言語で記述されたプログラムを書き込み，動作させることができる．そのプログラムの開発，及び書き込み機能が搭載されたソフトウェアが MPLAB である．

MPLAB はマイクロチップ社が開発した IDE で，PIC マイコンのファームウェアの開発環境を提供する．ソフトウェアは図 2.3 のように『エディタ』『プロジェクトマネージャ』『デバッガ』で構成されており，基本的にはアセンブラによる開発の機能を持っている．本研究ではアセンブラではなく C 言語で開発を行うので，それに対応するコンパイラを MPLAB に加えた．PIC24family の開発に用いる C30 がそれにあたる．

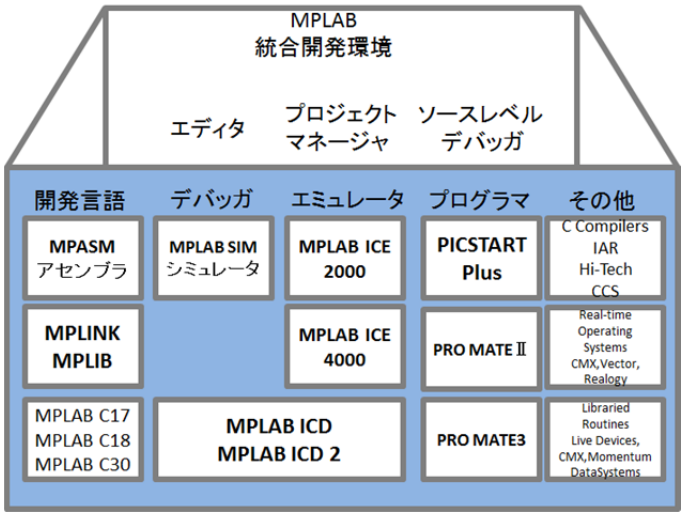

図 2.3　MPLAB の内部構成［8］

[5] Integrated Development Environment の略，ソフトウェアの開発環境を指す．

2.2.2 PICKit3

図 2.4 の PICKit3 は Microchip 社が開発した PIC マイコン専用のプログラマ[6]で, デバッガ機能も搭載している．通常 PIC マイコンにプログラムを書き込むとき一度チップを回路から取り外し，専用のソケットに接続しなければならない．PICKit3 はその手間をかけず，PIC マイコンから伸びる 6 本の端子にこれを接続するだけで書き込みを行うことができる．チップを回路に接続，または解除するときに端子を破損する可能性があるため，それを行わずにプログラムの書き込みができるため, PIC マイコンの損傷の危険をより低減できることから，PICKit3 を使用した．

図 2.4　PICKit3 の外観

2.2.3 アプリケーションの開発環境 eclipse

スマートフォンアプリケーション開発は eclipse を用いて行った．eclipse は IBM 社によって開発されたオープンライセンスの IDE であり，Java 言語でのプログラム製作を行うことができる他，プラグイン[7]を追加することによって C#，C++などの言語での開発も可能となる．Android アプリケーションも eclipse によって開発することができ，スマートフォンなどの端末のドライバをインストールすれば USB 接続によって書き込みも行うことができる．Android アプリケーションを製作するには，Google の発行する ADT と呼ばれるプラグインが必要となる．ADT とは Android Development Tool の略で，Android 端末で動作するプログラムの製作に必要なシステムを提供している．

6 ファームウェアを PIC マイコンに書き込む機能を持った機械の総称.
7 アプリケーションに特定の機能を追加するためのプログラム.

3章 ロボットの駆動制御

マスタであるスマートフォンに対して本ロボットの構成では，スレーブとして行動する．本章では駆動部の制御法と，スマートフォンと制御回路の接続法についてのみ取り扱い，画像処理などについては 5 章で解説する．

3.1 オムニキットの概要

オムニキット(図 3.1)は土佐電子が製造･販売しているロボットで，3 輪，または 4 輪タイプのものがあり自由度の高い駆動を実現することができる．本ロボットでは 3 輪タイプのものを採用したが，これは制御するモータの増加に伴い，その回路の規模が大きくなってしまうためである．以下にオムニキットの詳細(表 3-1)とオムニキットに付属されているタミヤ製のギヤボックスについての詳細(表 3-1)を示す．

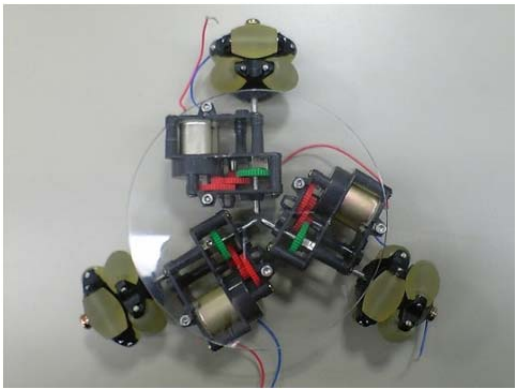

図 3.1　オムニキットの外観 [9]

表 3-1　オムニキットの詳細 [9]

サイズ	220mm×220mm×100mm
重量	660g
耐荷重	約 1kg
勾配移動	約 10 度（滑りにくい路面状態時）

表 3-2　タミヤ製ギヤボックスの詳細 [9]

ギヤ比	64.8：1	
トルク	784	（g*cm）
回転数	156	（rpm）

3.2 オムニキットの制御方法

我々が用いるオムニキットは 3 輪型の車両で，3 つの DC モータ，ギヤボックス，ホイール(図 3.2)，及びそれらを接続するベースで構成されている．このような駆動部を制御するには，それぞれの車輪の速度と回転方向を制御する必要があるが，駆動部の特性を理解し，制御モデルを考える必要がある．

3.2.1 オムニキットの特性

本研究で取り扱ったオムニキットは 3 つの DC モータが中心から 60° 毎に取り付けられており，一番の特徴は車輪に取り付けられたオムニホイールである．オムニホイールには車輪の回転軸に垂直な軸を持つローラが取り付けられており，車輪はその回転軸方向に滑ることができるため，通常の三輪車両では実現できない駆動，例えばその場での回転や，進行方向に対する斜め横への駆動，ななめ後ろへの駆動が可能となっている．

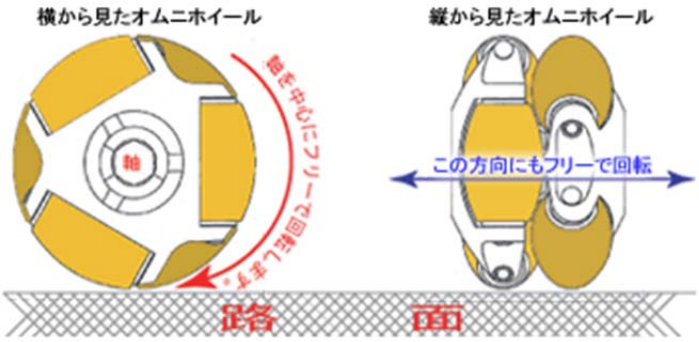

図 3.2　オムニホイールの外観［10］

3.2.2 オムニキットの制御モデル

オムニキットの駆動は 3 つの車輪の回転方向と回転速度で決定される．

図 3.3 のように X 座標の移動速度，Y 座標の移動速度，角速度をそれぞれ$v_x\ v_y\ v_\theta$とし，ホイール 1,2,3 の回転速度をそれぞれ$v_s\ \ v_t\ \ v_u$ とすると移動速度は次式 3.1 が成立する．

$$\mathrm{V} = \begin{pmatrix} v_x \\ v_y \\ v_\theta \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} & -1 \\ \frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} & 0 \\ l & l & l \end{pmatrix} \begin{pmatrix} v_s \\ v_t \\ v_u \end{pmatrix} \qquad \text{式 3.2.1}$$

この式に従い DC モータの回転制御を行うことでロボットの駆動方向及び移動速度を制御する．

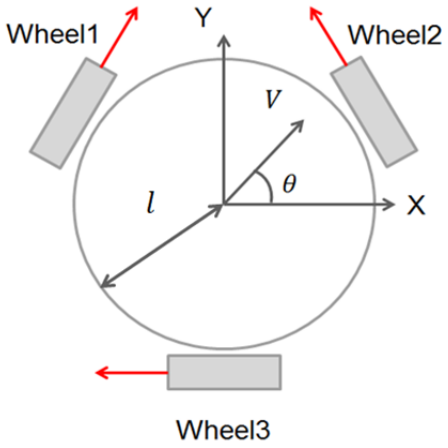

図 3.3　ホイールの回転とロボットの駆動［11］

3.3 ドライバ回路によるオムニキットの駆動

モータを駆動するための回路の制御には，PIC マイコンという種類の IC を用いている．PIC マイコンが行うのは『スマートフォンからの命令の受信』，それもとにした DC モータの『回転方向』と『回転速度』の制御であり，移動方向やその速度の指定はスマートフォンが Bluetooth 経由で行う．その 3 つの制御に対応する機能が無線通信機能, I/O 機能, PWM 機能の 3 つである．PIC マイコンは I/O 機能と PWM 機能を標準で搭載しており，また無線通信機能も新たに Bluetooth ドングルなど専用の送受信機を接続することによって可能となっている．回路に搭載された PIC マイコンを用いてこの制御を行うことにより，オムニキットを駆動する．これらの PIC マイコンを中枢とするモータ駆動のための回路全体を，以後はドライバ回路と呼ぶ.

3.3.1 PIC マイコンの概要

PIC マイコンには幾つかの種類があり，その内部レジスタ[8]の構成や機能数などにより幾つかのグレードに分けられている．本研究で使用した PIC24FJ64GB002(図 3.4)は 16bit 構成(表 3-3)のマイクロコンピュータで，実質的に使用可能な機能数の上限となる端子数も 28 本と多い．電源電圧は 3.3V，出力電流は 700mA となっている．この PIC マイコン以下のグレードを用いようとすると，モータ制御用に多くの端子を割いてしまい，ロボットの最低限の機能が果たせなく可能性があるため，このグレードを採用した.

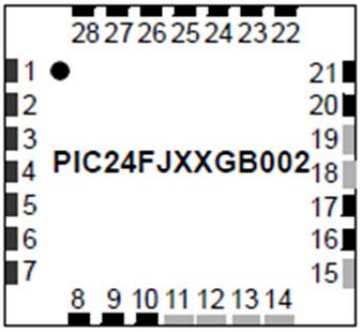

図 3.4　PIC24FJ64GB002 の外観と端子番［12］

表 3-3　PIC24FJ64GB002 の内部構成［12］

PIC24FJ Device	Pins	Program Memory (Bytes)	SRAM (Bytes)	Remappable Peripherals						I²C™	10-Bit A/D (ch)	Comparators	PMP/PSP	RTCC	CTMU	USB OTG
				Remappable Pins	Timers 16-Bit	Capture Input	Compare/PWM Output	UART w/ IrDA®	SPI							
64GB002	28	64K	8K	15	5	5	5	2	2	2	9	3	Y	Y	Y	Y

[8] プロセッサが内蔵する記憶回路．プロセッサ内部の状態・動作を決定するための機能を持つ.

3.3.2 RT-ADKmini の概要

RT-ADKmini(図 3.5)は RT 社が開発したマイコンボードである．通常 PIC マイコンを動作可能な状態で回路に実装するには，電源電圧を供給するためのレギュレータや，クロックを正確に動作させるための水晶発振子が必要になる．この RT-ADKmini にはそれらの PIC に必須な周辺回路が既に搭載されており，電源供給のための micro USB 端子，さらに Bluetooth ドングルを接続可能な USB 端子を搭載している．これらの回路が小さくまとまっており，基板に実装する回路の規模を縮小できることから，このマイコンボードを使用した．

図 3.5　RT-ADKmini の外観［13］

3.3.3 モータドライバの概要

ドライバ回路に求められる機能のうち，回転方向の制御はモータドライバという IC で行う事ができる．モータドライバは内部に H ブリッジと呼ばれる回路を搭載している．H ブリッジ回路(図 3.6)は 4 つのトランジスタにより構成された回路で，それらのトランジスタに電圧をかけることでモータへ流れる電流の方向の制御を行う．しかしこの回路をそのまま基板に実装するとなれば 1 つのモータに対して 4 つのトランジスタが必要となり，制御するモータの数が増える程回路の規模が大きくなってしまう．またバイポーラトランジスタを使ったモータドライバでは誘導起電力や発熱の問題が発生してしまう．これらの問題を解決するため，ユニポーラトランジスタの一種である電界効果型トランジスタ(以後 FET)で組まれた H ブリッジ回路を使う．

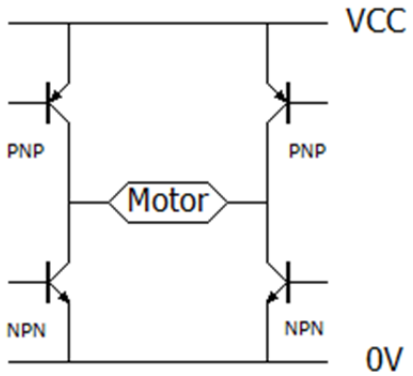

図 3.6　バイポーラトランジスタとそれによる H ブリッジ回路［8］

バイポーラトランジスタで組まれたドライバ IC を用いてモータを駆動すると出力飽和電圧[9]により電源電圧をそのままモータに流すことができず，電圧の損失がある．またその損失によりトランジスタが発熱し，場合によっては IC の正常な動作を阻害する，損傷に繋がるなどの問題が危惧される．一方 FET では動作させたときの抵抗が極めて低く発熱も少ないので，トランジスタより高い電圧・効率で DC モータを駆動できる．また今回は DC モータを駆動させるために，後述する PWM という機能を用いる．この機能は ON/OFF の矩形波を出力するので高速でのスイッチング動作を伴うが，FET はバイポーラトランジスタに比べスイッチ動作に要する時間が少ない．

よってモータドライバには，FET で組まれた H ブリッジ回路が内蔵された物を用いる事が望まれる．以上の条件を満たしたドライバ IC として，制御回路には東芝製モータドライバ IC MP4207 を使用した．この IC を用いることで，1 つの電源でモータの回転方向の制御を行う．図 3.7 はドライバの内部構造と端子構成である．

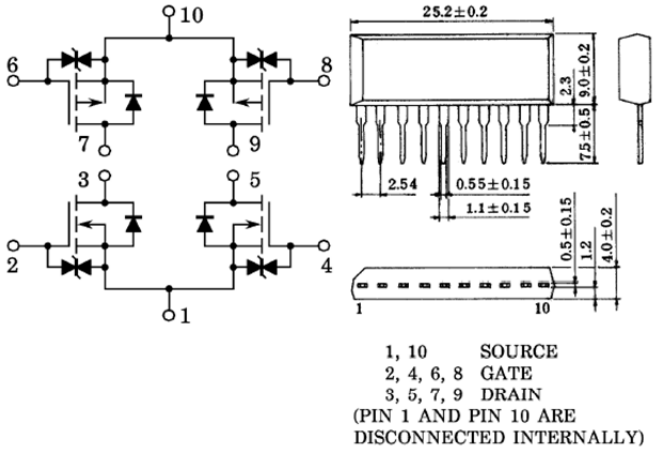

図 3.7　MP4207 の内部構造と外観 [14]

[9] バイポーラトランジスタの特性の 1 つ. 電圧降下によって出力側と入力側に電圧差が発生する現象.

3.3.4 モータドライバの制御

モータドライバにはソースとグランドに加え，4 つのドレインと 4 つのゲートによって構成されている．この IC の制御は 4 つのゲート端子にかける電圧のパターンを変えることによって行う．このモータドライバが行えるのは　正転/反転/空転/ブレーキ　の 4 つで，それらの条件を表 3-4 に示す.

表 3-4　ドライバ駆動の為の信号 [8]

	Gate2	Gate4	Gate6	Gate8
正転	0	1	0	1
反転	1	0	1	0
空転	0	0	0	0
ブレーキ	1	1	1	1

表 3-4 のような 4 つの信号を PIC マイコンで制御するには，GateN に接続された x ポートの LATx レジスタに High/Low を書き込むことによって行う．LATx レジスタとは出力機能を制御するためのもので，このレジスタに書き込みがあると対応した端子に 3V，18mA の信号が出力される.

LATx レジスタは表 3-5 のようなレジスタ構成となっている．PIC24family は 16bit 構成なのでレジスタは Bit15~0 となっている．例えば LATA の 10bit 目を High にした場合，A ポートの 10 番ピンに 3V が出力される．この機能を持つ端子を，1 つのモータにつき 4 つ接続することでモータドライバを利用した DC モータの制御を行うことが可能になる. 以下はレジスタに直接アクセスする命令法(図 3.8)と，C 言語の関数的記法を用いた命令法(図 3.9)である.

レジスタに直接アクセスする命令ではアクセスしたいポートに対応したレジスタを指定し，書き込むという方法をとる．この場合 LATA レジスタのビットに直接アクセスするという命令が LATAbits という部分に当たり，LATA3 という部分が書き込み対象となる具体的なビットの指定部分となる．この場合は表 3-5 の bit3 にアクセスしている.

```
LATAbits.LATA3 = 0;//Din1-RA3-OFF
LATAbits.LATA2 = 1;//Din0-RA2-ON
```

図 3.8　レジスタに直接アクセスするコード

表 3-5　ポートレジスタの内部構成［12］

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10(1)	Bit 9(1)	Bit 8(1)
TRISA	02C0	—	—	—	—	—	TRISA10	TRISA9	TRISA8
PORTA	02C2	—	—	—	—	—	RA10	RA9	RA8
LATA	02C4	—	—	—	—	—	LATA10	LATA9	LATA8
ODCA	02C6	—	—	—	—	—	ODA10	ODA9	ODA8
File Name	**Addr**	**Bit 7(1)**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit2**	**Bit 1**	**Bit 0**
TRISA	02C0	TRISA7	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
PORTA	02C2	RA7	—	—	RA4	RA3	RA2	RA1	RA0
LATA	02C4	LATA7	—	—	LATA4	LATA3	LATA2	LATA1	LATA0
ODCA	02C6	ODA7	—	—	ODA4	ODA3	ODA2	ODA1	ODA0

一方 C 言語による記述の仕方では，関数的な記述となる．関数名があり，引数があるという文になるが，関数名で対応する処理がある程度わかるようになっている．例えば下図 3.9 のプログラムならば，書き込むレジスタを mPORTA で指定し，Write で書き込み動作を行う事を指定しているような関数名となる [15]．実際に引数として渡されるのは図 3.9 の括弧内の数値であり，この中身は 16 ビットの数値が入る．例えば mPORTAWrite(0x000f) ならばポート A の下位 4 ビット全てに 1 が，上位 12 ビットには 0 が出力される．

```
//PortA,B
mPORTAWrite(0x000f);//ポート下位4ビット出力
mPORTBWrite(0x0000);//ポートB全初期化
```

図 3.9　関数を用いたコード

```
#define mPORTAWrite(lat) { LATA = (unsigned int)lat;} /*Write PortA*/
```

図 3.10　mPORTAWrite 関数の定義

実際にはこの関数は，図 3.10 のようにマクロによって実現されている．この方法では C 言語の関数と似た記述方法によって LATx レジスタへの書き込みを意識することなく信号を出力することができるが，これはレジスタ全体に書き込みを行ってしまうため，他の機能と書き込もうとしているポートが競合している場合，動作が不安定になる可能性がある．そのため制御信号はこの LATx レジスタに直接書き込む記述方法をとった．

3.3.5 PWM 機能による速度制御

DC モータの速度を変化させるにはモータに印加する電圧を ON/OFF といった 2 値的にではなくではなく，連続的に変化させる必要がある．しかし PIC マイコンを用いた回路上で電圧を連続的に変化させることは難しい. PIC マイコンは ON/OFF の 2 つの出力しか持っておらず，0V か 3.3V(製品によっては 5V)の固定値しか出力できないからである．そこで DC モータの速度を制御する方法の 1 つのとして PWM 制御を採用した.

PWM 制御とは Pulse Width Modulation 制御の略で, パルス幅を変調することによって様々なデバイスを制御するための機能である．PIC マイコンにはこの波形を出力する機能が標準で搭載されており，特別な回路を組むことなくこのパルスを出力できる．PWM はサーボモータの操作角を制御するために用いられるが, これを用いて本研究では DC モータの速度制御を行う.

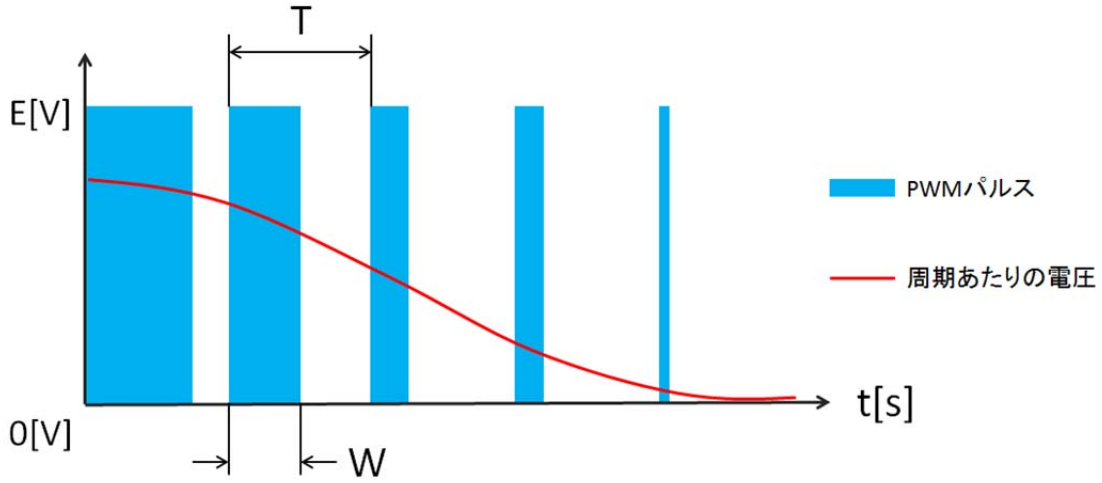

図 3.11　PWM パルスの波形

PWM パルスの波形を図 3.11 に表す. PWM 変調においてパルスの ON/OFF の周期 T とパルス幅 W を変化させることが出来る. 図 3.11 のように周期中のパルス幅が伸びると 1 周期あたりにモータが回転する時間が増加し，逆に 縮むと回転する時間が減少する．これを連続して行う事によってあたかもモータの回転速度を制御しているように見せる事ができる. 制御回路では PIC マイコンに搭載されたこの機能を用いる事により，DC モータの速度制御を行っている.

3.3.6 PWM パルスの出力方法

PIC マイコンで PWM パルスを出力するには Output Compare モジュールを用いる(以降 OC モジュール)．OC モジュールとはタイマ機能を利用して出力ピンの ON/OFF を制御するためのモジュールで，単一一致モード，二重一致モード，PWM モードなどの動作方法があるが今回は PWM モードを用いる．なお単一一致モードなどの他のモードについては本研究では使用しない機能の解説は割愛する．

PWM 制御はその周期とパルス幅により動作が決定される．ON/OFF の周期についてはタイマでの設定が必要になるが，タイマを利用するか，どのタイミングで ON になり OFF になるかなど，PWM 出力に関する動作の基本設定は OCxCON レジスタのみで行うことが可能である．OCxCON は表 3-6 のような構成となっているが，この設定も LATx レジスタ同様にレジスタを指定し直接書き込む方法と，関数を用いて書き込む方法が存在する．PWM の出力設定についてはこのレジスタが他の機能と競合を起こすことは考えにくいため，後者の方法で設定を行った．

OCSIDL はプロセッサへのクロック供給が停止され，動作をしていない間にも継続して PWM を出力するかどうかを決定するレジスタである． OCFLT は過電流などハードウェアが損傷してしまうような状態になったとき書き込まれるビットで，ユーザがこれを書き込むことはほぼない．ENFLT はフォルト機能を有効，もしくは無効にするビットで，これによってフォルト制御をするか否かを決定する．TRIGMODE は PWM が Low から High になるか，もしくは High から Low になるかを設定するためのビットである．

表 3-6　OC1CON レジスタの内部構成［12］

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	OCSIDL	OCTSEL2	OCTSEL1	OCTSEL0	ENFLT2(2)	ENFLT1
bit 15							bit 8

R/W-0	R/W-0, HCS	R/W-0, HCS	R/W-0, HCS	R/W-0	R/W-0	R/W-0	R/W-0
ENFLT0	OCFLT2	OCFLT1	OCFLT0	TRIGMODE	OCM2(1)	OCM1(1)	OCM0(1)
bit 7							bit 0

これらのレジスタのうち，PWM の波形を決定するために重要な設定を持っているのは 10,11bit 目の OCTSEL レジスタと 0,1,2bit 目の OCM レジスタである．OCTSEL レジスタは PWM パルスの周期を決定するタイマの設定を行う．OC モジュールで利用できるタイマはタイマ 2，3 であるため，これを選択する必要がある．OCM レジスタは動作モードの決定という重要な設定を行うレジスタで，ここでは PWM モードを指定するように設定する．

また PWM 機能のレジスタの設定も，レジスタに直接アクセスする方法と関数を用いた設定方法があるが，本研究では関数を用いた方法を採用した．

まず PPSOutput 関数によって OC モジュールの機能を割り付けるピンを選択し，OpenTimer2 関数でタイマ 2 を開始する．次に OpenOC 関数で PWM の動作モードの設定を行い，SetDCOC1 関数で PWM の波形を決定している．なお PPSOut 関数や OpenTimer 関数，OpenOC 関数，SetDCOC 関数などの実体は図 3.12~図 3.15 のようにマクロである．その内容については表 3-7~表 3-9 に示す．

これらの関数は PIC の種類によって異なり，自分の必要とする関数を調べる場合専用のライブラリを調べる必要がある．本研究の場合 Microchip PIC24F Peripheral Library がそれにあたり Microchip のフォルダの中にある．例えば本研究の開発を行った PC なら C:¥Program Files (x86)¥Microchip¥mplabc30¥v3.30¥docs¥periph_lib　にそのファイルが存在している．

```
#define PPSOutput(pin,fn) iPPSOutput(OUT_PIN_##pin,OUT_FN_##fn)
```

図 3.12　PPSOutput 関数の定義［15］

```
OpenTimer2(
    unsigned int config,
    unsigned int period
);
```

図 3.13　OpenTimer2 関数の定義［15］

表 3-7　OpenTimer2 関数の詳細［15］

config	This contains the parameters to be configured in the T2CON register as defined below Timer Module On/Off • T2_ON • T2_OFF Timer Module Idle mode On/Off • T2_IDLE_CON • T2_IDLE_STOP Timer Gate time accumulation enable • T2_GATE_ON • T2_GATE_OFF Timer prescaler • T2_PS_1_1 • T2_PS_1_8 • T2_PS_1_64 • T2_PS_1_256 Timer 32bit mode enable/disable • T2_32BIT_MODE_ON • T2_32BIT_MODE_OFF Timer clock source • T2_SOURCE_EXT • T2_SOURCE_INT
period	This contains the period match value to be stored into the PR register

```
OpenOC1_GB(
    unsigned int config1,
    unsigned int config2,
    unsigned int value1,
    unsigned int value2
);
```

図 3.14　OpenOC1 関数の定義 [15]

表 3-8　OpenOC1 関数の詳細 [15]

config1	This contains the parameters to be configured in the OCxCON1 register
config2	This contains the parameters to be configured in the OCxCON2 register
value1	This contains the value to be stored into OCxRS Secondary Register.[In single compare mode, user may set this parameter as 0x0000]
value2	This contains the value to be stored into OCxR Main Register

```
SetDCOC1PWM_GB(
    unsigned int dutycycle,
    unsigned int period
);
```

図 3.15　SetDCOC1 関数の定義 [15]

表 3-9　SetDCOC1 関数の詳細 [15]

dutycycle	This is the duty cycle value to be stored into Output Compare Main Duty Cycle register (OCxR).
period	This is the period stored into Secondary register (OCxRS)

3.3.7 周辺回路 1　DC モータの電源

制御回路ではモータを駆動させるにあたり，PIC マイコンとは別に DC モータを駆動させるための電源が必要になる．PIC マイコンの消費電力は極めて小さく，モバイルバッテリーで電源の供給を行っても安定して動作するが，DC モータは駆動に 700mA の電流を必要とし，さらに単位時間当たりの電流出力を多く必要とする．充電器として製造されたモバイルバッテリーでは単位時間当たりの出力電圧が足りず，モータを十分に駆動させることができない．そこでモータの電源としてモバイルバッテリーよりも時間あたりの出力電流に勝る乾電池を採用した．

3.3.8 周辺回路 2　レベルシフト IC の導入

このモータドライバは 4VGate ドライブなので Gate に 4V の電圧をかける必要があるが，PIC が出力可能な電圧は 3.3V であるのでこれを上回るように出力信号を昇圧する必要がある．またモータドライバの出力する電圧はゲート - ソース電圧で決定される．このゲート - ソース間電圧が増加する程ドレインに出力される電流も，図 3.16 で示したように増加する．

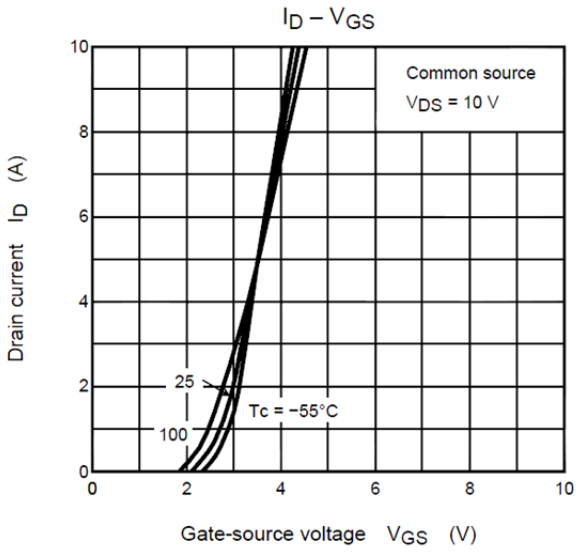

図 3.16　ドレイン-ゲートソース間電圧のグラフ [14]

レベルシフト IC(図 3.17)とは 3.3V で動作する PIC などのマイコンと, 5V で動作する GPS など動作電圧が異なる装置を繋ぐための IC であり，任意の電圧を昇圧，またその逆に降圧することができる．これによって PIC マイコンからの信号を 3.3V から 5V に昇圧した．レベルシフト IC の機能を表 3-10 に示す.

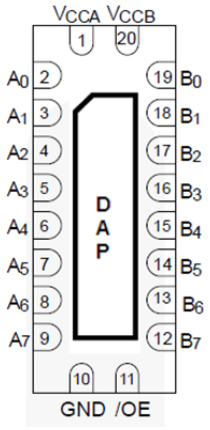

図 3.17　レベルシフト IC のピン構成［16］

表 3-10　レベルシフト IC の機能［16］

Pin #	Name	Description
1	V_{CCA}	A-Side Power Supply
2	A0	A-Side Inputs or 3-State Outputs
3	A1	A-Side Inputs or 3-State Outputs
4	A2	A-Side Inputs or 3-State Outputs
5	A3	A-Side Inputs or 3-State Outputs
6	A4	A-Side Inputs or 3-State Outputs
7	A5	A-Side Inputs or 3-State Outputs
8	A6	A-Side Inputs or 3-State Outputs
9	A7	A-Side Inputs or 3-State Outputs
10	GND	Ground
11	/OE	Output Enable Input
12	B7	B-Side Inputs or 3-State Outputs
13	B6	B-Side Inputs or 3-State Outputs
14	B5	B-Side Inputs or 3-State Outputs
15	B4	B-Side Inputs or 3-State Outputs
16	B3	B-Side Inputs or 3-State Outputs
17	B2	B-Side Inputs or 3-State Outputs
18	B1	B-Side Inputs or 3-State Outputs
19	B0	B-Side Inputs or 3-State Outputs
20	V_{CCB}	B-Side Power Supply
DAP	NC	No Connect

3.3.9 周辺回路 3　ボルテージフォロアによる出力電流の補填

モータドライバを通して DC モータを動作させるためにはある程度の電流量が必要になってくる．しかし PIC マイコンの IO 機能で出力できるのは 25mA 程度の電流であるので，これを補わなければならない．そこでボルテージフォロアを使用した．

ボルテージフォロアはオペアンプを用いた回路の一種で，図 3.18 の 1，2 間の抵抗を無限大にすると出力と入力の電圧がほぼ等しくなることを利用している回路である．この回路を用いる事によって回路間の電圧分離を行い，同時に動作に必要な電流を出力した．

この回路はオペアンプを用いた非反転増幅回路の一種として見做すことができる．図 3.18 の回路において入力電圧 V_i と出力電圧 V_o の関係は式 3.2 で得られる．

$$V_0 = \left(1 + \frac{R_2}{R_1}\right) V_i \qquad \text{式 3.3.1}$$

このとき$R_2 = 0$，$R_1 = \infty$ とすると $V_0 = V_i$ が成り立ち，入力と出力の電圧が常に一定に保たれるような回路となることがわかる．

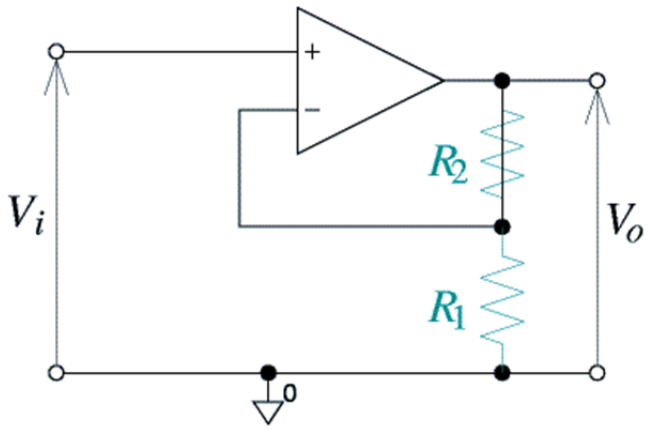

図 3.18　オペアンプによる増幅回路［17］

3.3.10 電源

今回バッテリーに使用したのは Panasonic 社製，QE-PL102 である．図 3.19 と表 3-11 にその外観と詳細を示す.

図 3.19　QE-PL102 の外観 [18]

表 3-11　QE-PL102 の詳細 [18]

スマートフォンフル充電回数の目安	約 1 回※1
電池容量	Li-ion min.2,700mAh/3.7V※2
充電時間	無接点充電パッド:5 時間※3
AC 電源	約 3.5 時間
USB	約 7 時間
USB ポート数 充電用/供給用	1(MicroUSB)/1(USB-A)
USB 出力	DC5V 1A(MAX)
本体寸法	約高さ 24×幅 70×奥行 40mm
本体質量	約 85g

3.4 Bluetooth 通信によるスマートフォンとの連携

図 3.20 に示したようにこのロボットはカメラから取得した映像をスマートフォン上で画像処理し，決定された移動方向を制御回路が受け取る，という構成になっている．今回制御回路とスマートフォンの無線通信には Bluetooth 通信を採用した．

図 3.20　ロボットの無線ネットワーク構成

3.4.1 Bluetooth 通信の概要

Bluetooth 通信は近距離無線通信に特化した通信規格で，主にワイヤレスウォークマンやヘッドホンなどの製品に搭載されている．これは Bluetooth が他の無線通信規格に比べて省電力であるためである. 通信容量や有効範囲については Wi-Fi などその他の無線通信に一歩劣るものの，近距離での通信で大容量の通信を必要としない場合，そして省電力を求められる場合，Bluetooth 通信はその要求に適した通信方法の 1 つと言えるだろう．表 3-12 は Bluetooth と Wi-Fi 通信の性能を比較したものだが，そちらに詳しく載っている.

ただし PIC マイコンを用いた回路で Wi-Fi や Bluetooth などの無線通信を可能にするには，新たに専用の回路を増設して通信機能を実装し，専用のプログラムを書きこむ必要がある．Bluetooth は既に無線通信機能として普及しており，それらの機能を実装するモジュールは Bluetooth ドングル(図 3.21)と呼ばれる．しかしモジュールを接続すればそのまま使えるというわけではなく，そのためのプログラムが必要となる．今回使用する BUFFALO 社製の Bluetooth ドングル，BSHSBD02 は Bluetooth2.1+EDR という通信規格を採用している．ここで Bluetooth2.x +EDR と Wi-Fi 802.11 の比較表を表 3-12 に示す.

図 3.21　Bluetooth ドングルの外観

表 3-12　Bluetooth と Wi-Fi 通信の比較 [19]

	Bluetooth2.x +EDR	Wi-Fi 802.11
通信速度	1.3 Mbps	54 Mbps
有効範囲	100 m	80 m

3.4.2 Bluetooth 通信のファームウェアとアプリケーション

PIC マイコンで Bluetooth 通信を行うには, 3.4.1 で紹介したように Bluetooth ドングルを接続する必要があるが，同時にその通信のためのプログラムを PIC マイコンに書き込まなければならない．通常 Bluetooth 通信を行うには専用のライブラリ[10]やプロファイル[11]を調べてプログラムを製作するが，これは膨大な作業である．そこで WEB 上で hrdakinori 氏が無償公開している Android 用アプリケーション『DroidControl』，及び PIC マイコン用ファームウェアの『BT_DROID』を利用した [20].

このプログラムはスマートフォンと PIC マイコンを Bluetooth で繋ぐことのできるファームウェアとアプリケーションを提供しており，本来ならば必要なプロファイルなどの設定作業をこちらで実装する必要がない．ドライバ制御のプログラムはこの通信プログラムを基にして製作した．

3.4.3 制御回路の回路図

以上の要素によって構成された制御回路を図 3.22 に示す．PIC マイコンから出力された信号がレベルシフトで昇圧され，オペアンプ，NAND 回路を通ってモータドライバに供給されていることがわかる．回路図に電源供給用の端子や Bluetooth ドングルが載っていないのは，制御回路で使用している RT-ADKmini ボードの USB 基板に USB 接続端子があり，そこに接続しているためである．

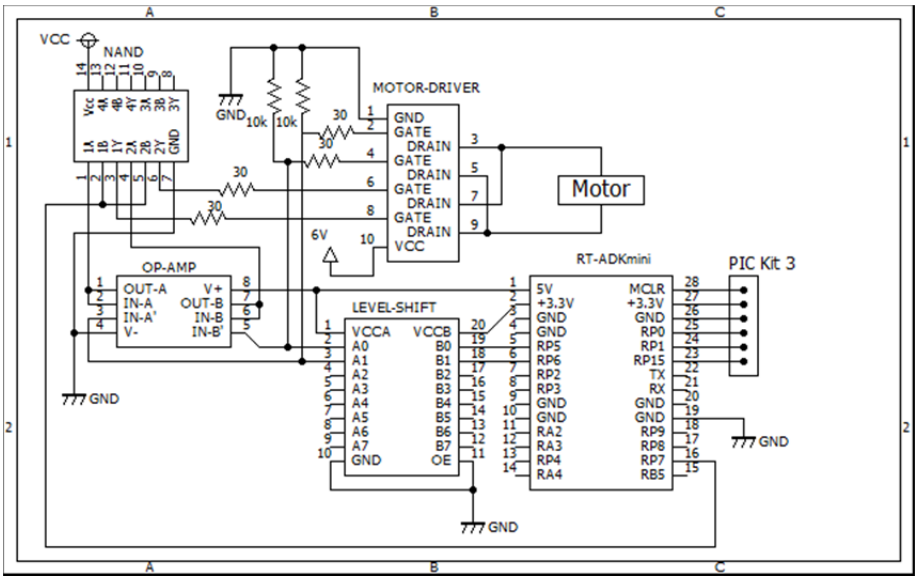

図 3.22　ドライバ回路の回路図

---

[10] 汎用性の高い複数のプログラムをまとめたもの．

[11] Bluetooth 通信におけるプロトコールの一種．ヘッドホンや健康機器など様々なモジュールに接続するため，各モジュールに特化したプロトコールが組まれている．端末間のプロファイルに互換性がないと接続できないなどの問題が生じる．

3.5 PIC マイコンのプログラムとスマートフォンからの命令

PIC マイコンに書き込まれたプログラムは『スマートフォンからの命令を受け取り，DC モータを駆動させる』ためにある．Bluetooth の接続処理に関しては，3.4.2 で示したように『BT_Droid』が処理してくれるが，その他の動作は新しく書き込む必要がある．そこでまず PIC マイコンが行う処理のフローチャートを図 3.23 に示す．

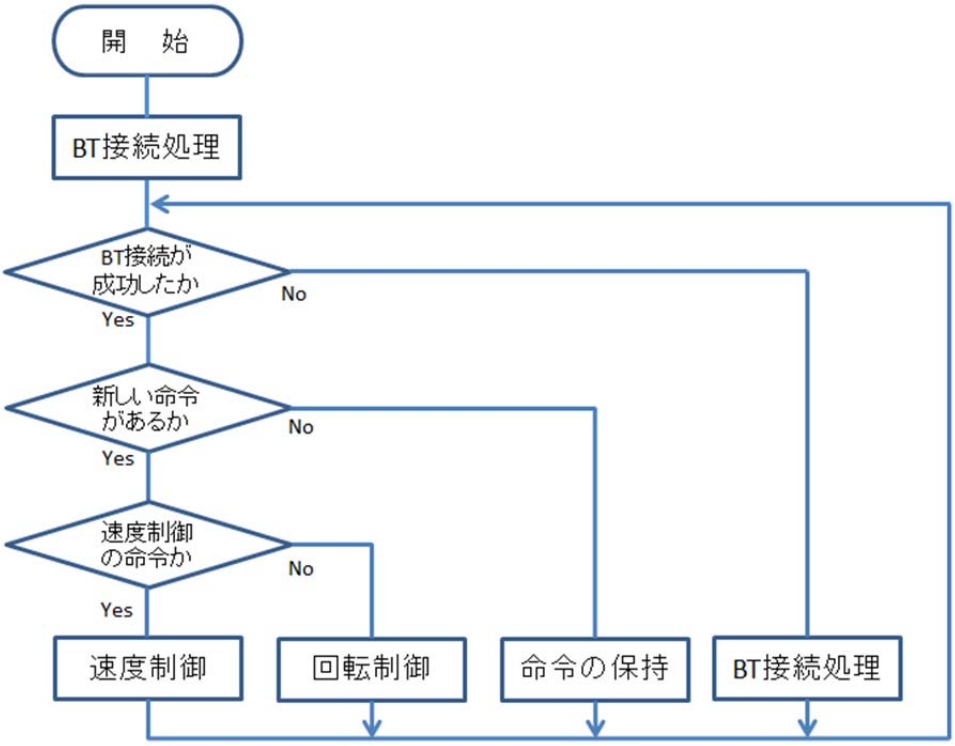

図 3.23　接続処理の追加プログラムのフローチャート

スマートフォンから送信される信号は 2 つの命令で構成されている．それはその命令が回転方向の制御のための情報か，速度制御のための情報かを伝えるための情報かを判別するための接頭文字と，それぞれの具体的内容を持った部分の 2 つである．命令の頭にくる文字により回転方向か速度制御かを見分け，その後に来る文字列でその具体的内容を計算する．

回転方向の制御の場合，l(**)のように必ず 3 文字で構成された命令が送信され，速度制御の場合は s(00~ff)，t(00~ff)，u(00~ff)という，文字の後に 8bit の数値が送信されることになる．これらの信号を常に受け取り，通信のバッファに値の入力がなくなるまで同じ動作を繰り返す． なお，表 3-13 の数値は回転制御が 2 進数で表され，速度制御は 16 進数で表されている．

表 3-13　モータドライバの制御信号

制御動作		接頭文字	数値
回転制御	正転	l	10
	反転		01
	停止		00
	ブレーキ		11
速度制御	停止～中速	s,t,u	0～7e
	中速		7f
	中速～最大速		80～ff

4章 スマートフォンアプリケーション

スマートフォンアプリケーションは常にカメラから画像を取得し，画像処理から得た結果を命令として制御回路に送る．またその内部の処理では音声認識により目標物の変更が発生する．3.4.2 で述べたように，このアプリケーションは hrdakinori 氏の『DroidControl』を基に記述されている．このアプリケーションはスマートフォンと制御回路の通信はサポートしているが，カメラからの画像の取得，音声認識など，ロボットの制御に必要な部分を書き加える必要がある．本章では主に書き加えた部分について取り扱う．

4.1 アプリケーションの構成

スマートフォンに搭載するアプリケーションは幾つかのクラスによって成り立っている(図 4.1).『音声認識』『制御回路に送信する値の処理』を行うメインクラス.『動画取得』を行うクラス.『画像処理』を行うクラス.『BT 接続』を行うクラスである.

シングルスレッドのアプリケーションでは同時に 2 つの処理を動作させることはできないが，マルチスレッド化を行うことによって複数の処理を同時に動作させることができる．ここでは動画取得を行うスレッドと，画面の UI の更新を行うスレッドを同時に稼働させている．Bluetooth 通信を行うクラスはメインクラスから適時呼び出され，命令を制御回路へ送信する．これらの 3 つのクラスが連携して動作することでアプリケーションのメイン動作が行われる．

実際にはこれらの他に，Wi-Fi，Bluetooth 通信の設定変更を行うためのクラスが幾つか存在するが，これらのクラスの詳細な解説は割愛する．

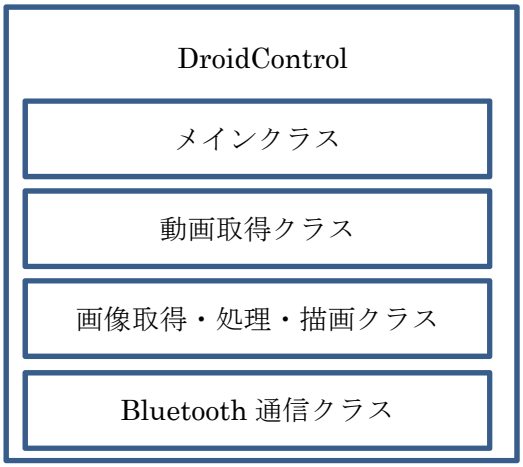

図 4.1　アプリケーションの構成

4.1.1 マルチスレッド化による並列動作

通常のプログラムは記述された通りの順序でコードが実行されていく．多くの命令を条件分岐などでおこなっていくが，その処理は常に 1 つの流れ(スレッド)で実行される．よって本ロボットのように画像取得とその他の動作を同時に行いたい場合，何らかの方法で処理を並列化する必要がある．それを可能にするのがマルチスレッドである．Java 言語はマルチスレッドに対応している．図 4.2 はシングルスレッドとマルチスレッドの処理を比較したイメージ図である．

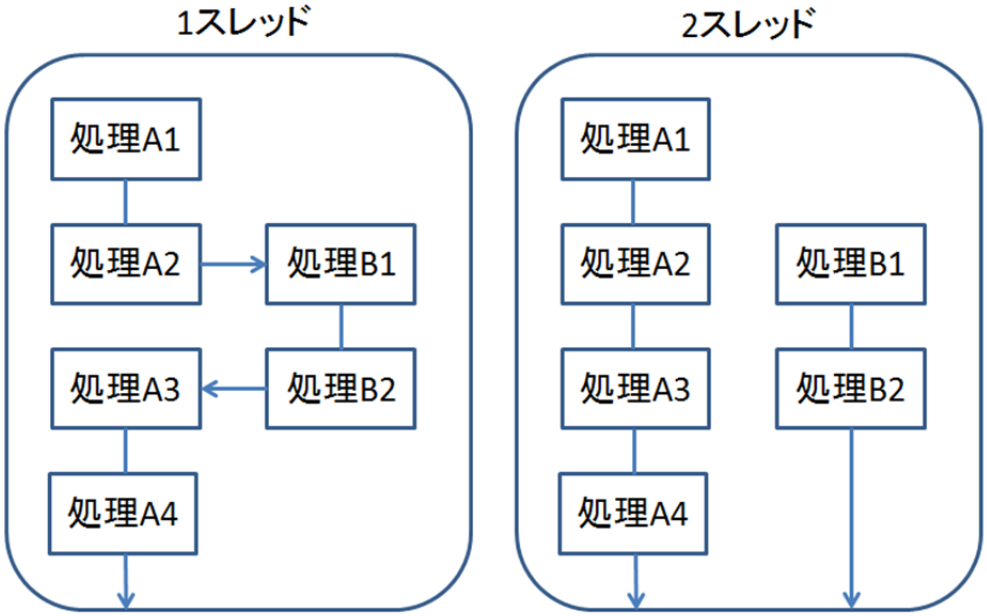

図 4.2　シングルスレッドとマルチスレッドの比較図

4.2 メインクラスの動作

4.2.1 音声認識 API

音声認識の実際の処理は Google の保有するサーバ上で行われるので，サーバに音声情報を送信する必要がある．音声情報をサーバへ送信するには，Google 社が提供している音声認識アプリケーションを仲介して行われる．本アプリケーションが行っているのは，そのアプリケーションの呼び出しと，返ってきた文字列からキーワードを検出，取得する部分である．

音声認識アプリケーションを別のアプリケーションから呼び出す為には, Intent を用いる．Intent とはアプリやそれを構成する機能同士を跨いで情報のやりとりを行うための API[12]である．音声認識では Recognizer Intent を用いて，インターネット経由で音声認識の結果を取得する．

この Intent が実際に行っている動作は図 4.3 のようになっている．アプリケーションから Intent が発行され，音声認識アプリケーションが別に立ち上がる．その音声認識アプリケーションが音声をサーバ上に音声情報を送信し，結果を受け取り，アプリケーションに返す．本研究のアプリケーションでは小型ロボットの制御にこの音声認識を用いるが，インターネット上のサーバに接続しなければならないので時間がかかり，素早い動作が求められるような制御を行うのは好ましくない．よって目標とする物体を識別するための色の選択を音声認識で行う．

図 4.3　音声認識処理の流れ

[12] Application Programming Interface の略．アプリケーションからプログラミングを行うために用意された，一連の処理をまとめ，簡潔に記述できるようにした機能．

4.2.2 色の種類

一口に色と言っても我々の生活圏には様々な色が遍在している．それらの色を分類する方法として，スペクトルによる分光や，人が色を認識するための処理など種々の方法がある．しかしそれらの色の認識を行えば処理に時間を裂かれてしまう．本ロボットが行うようなリアルタイムに画像を処理し，その結果によって動作を決定するというような処理速度を要求する動作を，複雑な色の認識処理と同時に行えば，ロボットの動作に遅れが生じる．詳しい内容については後述の画像処理についての章にて取り扱うが，本研究では画像上の色は全て図 4.4 に示すような，あか：(255,0,0)　みどり：(0,255,0) あお：(0,0,255) の 3 原色をもとに処理を行っている．よって音声認識で目標とする色の選択を行う場合，あか，みどり，あおの 3 パターンを抽出すればよい．

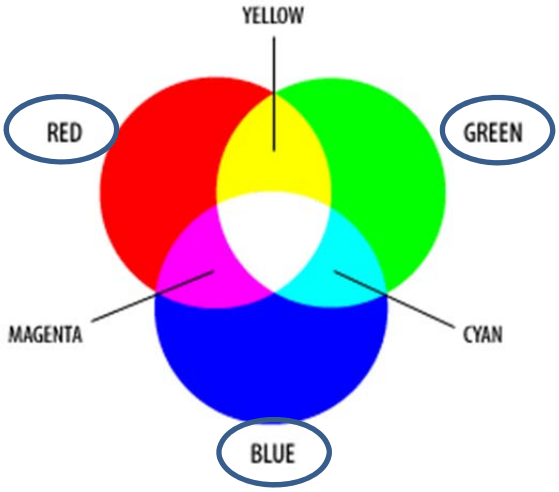

図 4.4　色の 3 原色と中間色 [21]

4.2.3 音声認識の結果と命令の照合

Recognizer Intent により返ってきた値はで文字列としてアプリケーションに渡される．この値は一つの結果のみではなく，類似する候補も全て返す．例えば『あか』という音声を入力した場合，『あか』という文字列だけではなく『赤』や『朱』という文字列も同時に返される可能性がある．本アプリケーションにおいて音声認識は使われる状況が限定されており，選択肢も物体の色，赤・青・黄だけなので，誤認識の可能性を考慮するよりもより確実に言葉を拾いたい．よって返ってきた『あか』『アハ』『はか』などの複数の結果を『あかアハはか』というように 1 つの文字列に合成し，『あか』『みどり』『あお』という文字列と照合していくという方法をとった．

4.3 カメラと画像取得クラス

4.3.1 カメラの概要

画像処理を行うに当たりロボットにカメラを搭載した．本研究では無線通信の問題から，Ai‐Ball と呼ばれる Wi-Fi カメラと，スマートフォンに搭載されたカメラを利用している．

4.3.2 Ai‐Ball の詳細

Ai‐Ball は Wi-Fi 通信で動画をストリーミングすることができる Wi-Fi カメラで，図 4.5 で示すようにサイズも小さく携帯性に富む．電源はカメラ用の電池で供給されるが，別売りの専用クレイドルと接続し，その mini USB 端子から電源を供給することも可能である．動画は Ai‐Ball 専用に割り振られた IP アドレスに接続し，専用の URL を指定することにより取得する．送られてくる映像の解像度は 640×480 の 2Mpixel で，この画像から画像処理を行う．Ai‐Ball の詳細については表 4-1 に示す．

図 4.5　Ai-Ball の外観 [22]

表 4-1　Ai-Ball の詳細 [23]

通信規格	Wi-Fi 802.11 b/g
最大フレームレート	30fps
バッテリー連続動作時間	1.5 時間
視認可能距離	25m
視野範囲	30mm × 35mm

4.3.3 カメラのネットワーク

カメラのネットワークは Ai-Ball，スマートフォンのカメラ両者とも Wi-Fi 通信で構成されている．Wi-Fi 通信によりネットワークを設けたい場合には，アクセスポイントと呼ばれる通信の拠点を設置しなければならない．Ai-Ball に関してはそれ自体がアクセスポイントとなるが，スマートフォンのカメラの場合テザリング機能を用いてコントローラとなるスマートフォン自身がアクセスポイントとなり，カメラスマートフォンからの接続を受け入れるか，別途ルータを用意しカメラとコントローラのスマートフォンがそこに接続するという方法をとることになる．図 4.6~図 4.8 はネットワークの構成を示している．

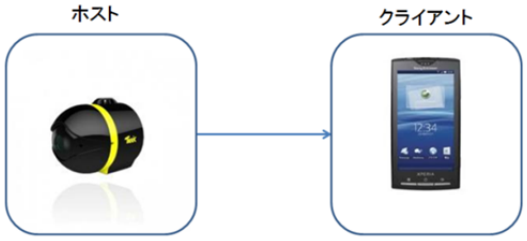

図 4.6　Ai-Ball の通信の図

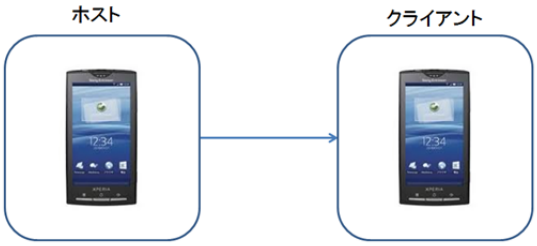

図 4.7　カメラーコントローラの図

図 4.8　カメラ ― ルータ ― コントローラの図

4.3.4 スマートフォン内蔵のカメラ

スマートフォン内蔵のカメラを使う場合，その動画をネットワーク上に乗せる必要がある．それを行うものとして IPWebcam と呼ばれるアプリケーションを導入した．

IPWebcam は GooglePlay で無償公開されているスマートフォン用アプリケーションである．このアプリを導入することにより，Ai-Ball と同様に専用の IP アドレスに接続することで動画を取得できるようになる．

5章 画像処理

画像情報を用いてロボットを制御するためには，画像上から意味のある情報を取得し，それを適切に処理しなければならない．本章では画像から意味ある情報，特徴量を取得しそれを元にロボットの制御を行う方法について説明する．

5.1 視覚フィードバックによるロボット制御

本研究ではカメラで取得した画像情報を二値化し，ロボットとの相対的な位置，姿勢を制御している．このように画像処理を用いた制御を視覚フィードバックという．視覚フィードバックを用いたロボット制御の方法には，対象の特徴量，例えば抽出した領域の面積やその形状，中心座標などを抽出し制御を行う方法と，ロボットアームの先端と観測対象との距離を画像から三次元，あるいは二次元に再構成して取得し，制御を行う方法がある．

本研究では画像処理専用に製造されたプロセッサではなく，スマートフォン内蔵の汎用プロセッサによる処理を前提としているので，大きな処理能力を要求する方法を排除する必要がある．よって画像から三次元の位置を推定するというような処理ではなく，より単純化された方法で制御を行う．その方法として，二値化された画像上の対象物が持つ『特徴量の重心座標』を用いて，対象物の追従運動を行う制御を行った．

ただしスマートフォンの OS は画像処理によりロボットを動作させることを想定しておらず，制御アプリケーションをそのまま記述しても処理速度に限界がある．これは Java がマルチプラットホームでの動作を前提としており，ヴァーチャルマシンを介してプロセッサを動作させている事に起因している．そこで本研究では Java アプリケーション上からヴァーチャルマシンを介さずプロセッサを動作させることができる Java Native Interface とよばれる API を採用した．

5.1.1 Java Native Interface(JNI)の導入

画像処理は当初 Java 言語による記述のみで完結していたが，これによる画像処理には時間がかかり，ロボットの動作に命令が追いつかないといった深刻な時差が発生してしまう．そのためこれを高速化する方法が必要になる．その手段として Java Native Interface を採用した．

Android アプリケーションは Java 言語で記述されており，仮想マシン[13](以降 VM)を通してプロセッサに命令が送られ，その命令をもとに動作する．これは Android のカーネルやライブラリといったコアとなる部分のほとんどが C 言語，もしくは C++で記述されていることに起因する．Java 言語で記述された命令は C 言語で記述されたマシンの上で動作することはできず，VM により C 言語の命令に置き換えられ実際の処理に移る．この VM を通した変換作業の有無により純粋に C 言語で記述された命令と Java で記述された命令の間に処理速度の差が発生する．また VM を介することにより，そのプロセッサの得意とするネイティヴコードを呼び出すこともできなくなり VM を使う場合とそうでない場合，アプリケーションによってはその処理能力の差が顕著に表れてしまう．さらに Java 言語は持っていないが，C 言語ならば持っているというような命令が存在し，JNI による高速化は主にそのような命令を利用することによって高速化をはかる事ができる．

ただし，JNI を用いる事によって処理が高速化されることもあるが，一方で JNI が使用する命令に対応したシステムを持つプロセッサ上でなければ動作する事ができず，Java 言語が持つプラットホームの柔軟性を損なってしまうという面も存在するので注意が必要である．ただし本研究では画像処理を行うスマートフォンは固定したので，この問題は発生しない．そこで画像処理は Java Native Interface を用いて，C 言語で記述する方法をとった．

[13] ある言語で記述されたプログラムを，別の言語で記述されたプラットホームで実行可能にするソフトウェア．ここでは Java で記述されたプログラムを Android 上で動かすためのソフトウェア．

5.2 色認識のアルゴリズム

色認識は図 5.1 の流れで行われる.

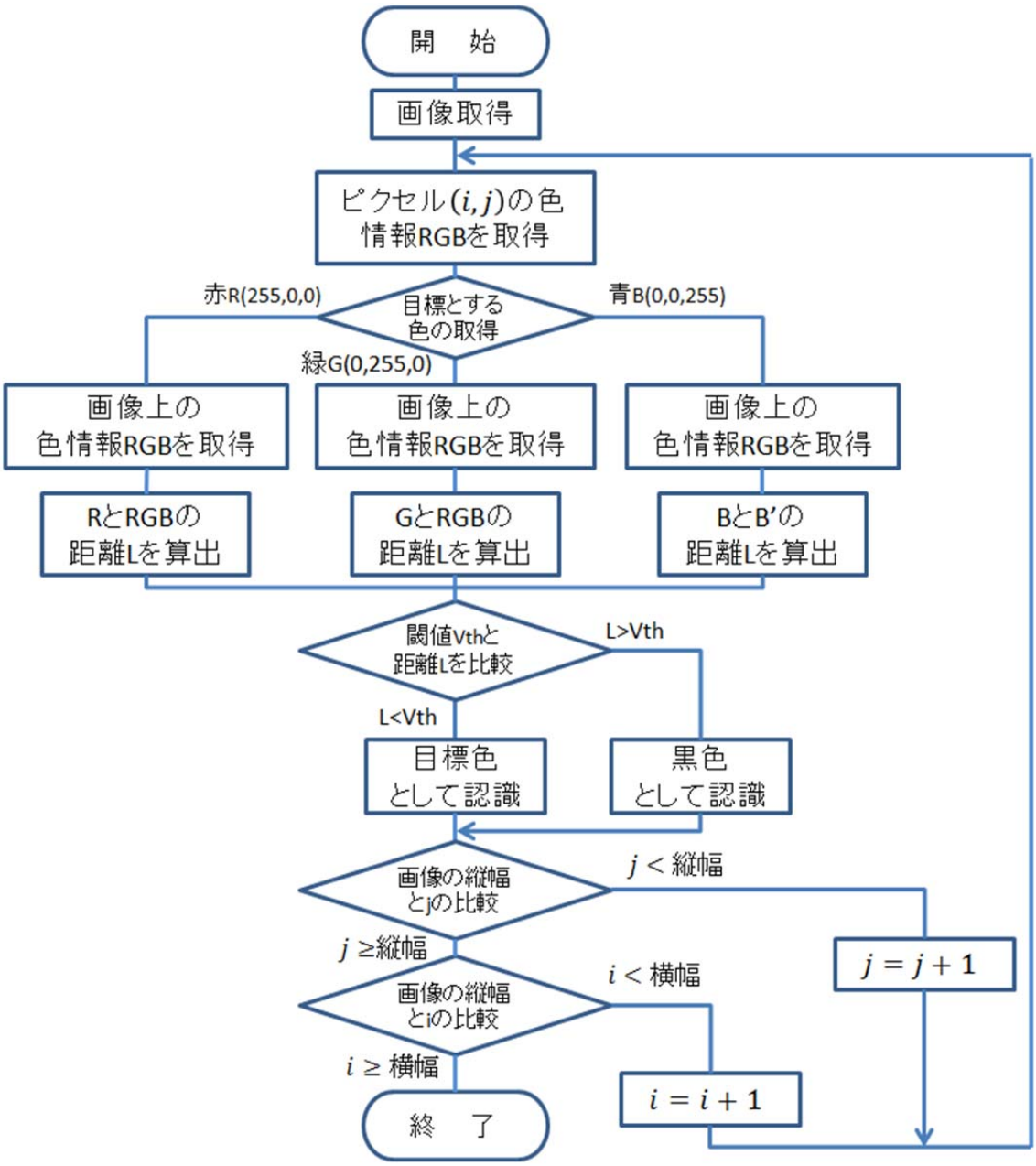

図 5.1　色認識のフローチャート

5.2.1 RGB モデル

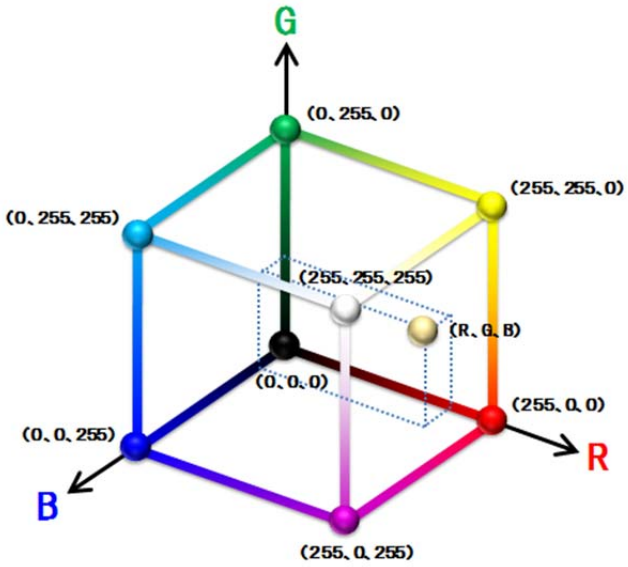

図 5.2　RGB 色空間［24］

RGB モデルとは色の三原色と呼ばれる RGB の三要素を合成して, 様々な色を再現するための物である．画像上の色を取得するとき，この RGB によって色情報が形成される．

RGB 値による色認識を行うにあたり，画像上の色を赤(Red)，緑(Green)，青(Blue)の 3 つに分類し，ある点での RGB 値を取得する．図 5.2 のような RBG 色空間において，取得した点の RGB 値

取得値 RGB(R = 0~255 ， G = 0~255 ， B = 0~255)

と目標とする色の RGB 値

R($R_r = 255$ ， $G_r = 0$ ， $B_r = 0$)　　G($R_g = 0$ ， $G_g = 255$ ， $B_g = 0$)

B($R_b = 0$ ， $G_b = 0$ ， $G_b = 255$)

とのユークリッド距離 L_r ，L_g ，L_b を定義し，この値が閾値に収まった場合目標とする色として認識する．この値が閾値 V_{th}よりも小さければ目標の色であると認識し，それ以外は黒と判断する．

$$L_r = \sqrt{(R_r - R)^2 + (G_r - G)^2 + (B_r - B)^2} \qquad \text{式 5.0}$$

$$L_g = \sqrt{(R_g - R)^2 + (G_g - G)^2 + (B_g - B)^2} \qquad \text{式 5.1}$$

$$L_b = \sqrt{(R_b - R)^2 + (G_b - G)^2 + (B_b - B)^2} \qquad \text{式 5.2}$$

5.2.2 RGB モデルの問題点

RGB モデルでの画像認識は単純な計算で認識を行う事ができる. ただし RGB での二値化の場合，動作環境の明暗によって誤差が発生する．同じ赤でも明るい赤と暗い赤があり，これを赤と認識するために閾値を調整すると目的としない色の要素まで赤と認識してしまう恐れがある．これは RGB モデルではそれぞれの色要素の合計が明るさや鮮やかさを決定しているためである．そのため，色の種類，色の鮮やかさ，色の明るさという 3 つの要素を持つ HSV モデルによる二値化を行った.

5.2.3 HSV モデルの導入

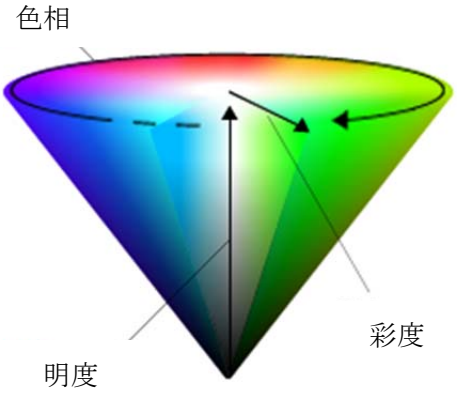

図 5.3　HSV 色空間［25］

HSV モデル(図 5.3)は『色相：Hue』『彩度：Saturation』『明度：Value』の 3 成分から構成される色の表現法をさす. RGB から HSV モデルへは式 5.3～5.7 で変換できる．これを RGB モデルのように目標とする色情報と画像上の色情報との距離 L を定義し，閾値と比較し画像処理を行う.

$$\mathrm{H} = 60 \times \frac{G - B}{MAX - MIN} \qquad \text{If MAX} = \mathrm{R} \qquad \text{式 5.3}$$

$$\mathrm{H} = 60 \times \frac{B - R}{MAX - MIN} + 120 \qquad \text{If MAX} = \mathrm{G} \qquad \text{式 5.4}$$

$$\mathrm{H} = 60 \times \frac{R - G}{MAX - MIN} + 240 \qquad \text{If MAX} = \mathrm{B} \qquad \text{式 5.5}$$

$$\mathrm{S} = \frac{MAX - MIN}{MAX} \qquad \text{式 5.6}$$

$$\mathrm{V} = \mathrm{MAX} \qquad \text{式 5.7}$$

5.2.4 RGB モデルによる二値化と HSV モデルによる二値化の認識比較実験

RBG モデルによる二値化と HSV モデルによる二値化の比較を行った. まず, RGB と HSV モデルによる二値化で, 同じ領域を検出する状態に閾値を設定. 次に図 5.4 の右側の画面のように二値化の対象となる赤色の円が印刷された立方体を手で覆い, 屋内で蛍光灯を消灯した状態でどれ程のピクセルを検知できるかを実験した.

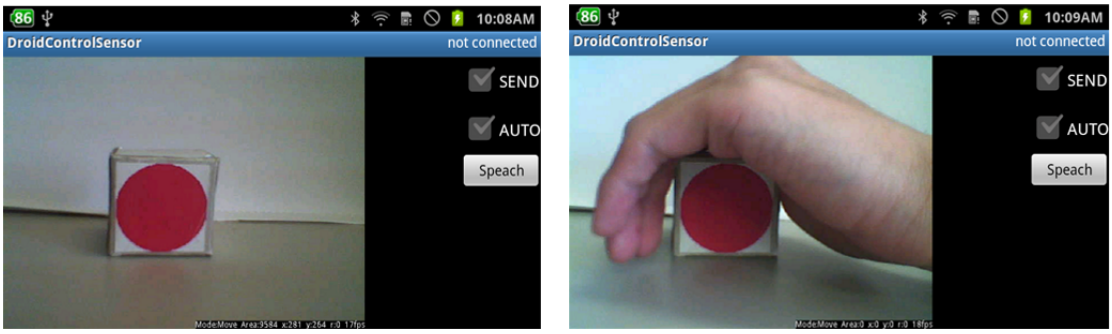

図 5.4　実験の様子. 実験前(左)　実験中(右)

RGB モデルを用いた二値化(図 5.5)では, 明度が下がると赤色と検出される領域が極端に狭くなるが, 一方で HSV モデルによる二値化(図 5.6)を行った場合はノイズが乗ってしまっているが検出される領域が円である事が理解できる程度の大きさになっている. ピクセル数では, RGB モデルの場合, 200 ピクセル前後の領域を検出したが, HSV はその 18 倍程度の 3700 ピクセル程の領域を検出することができた.

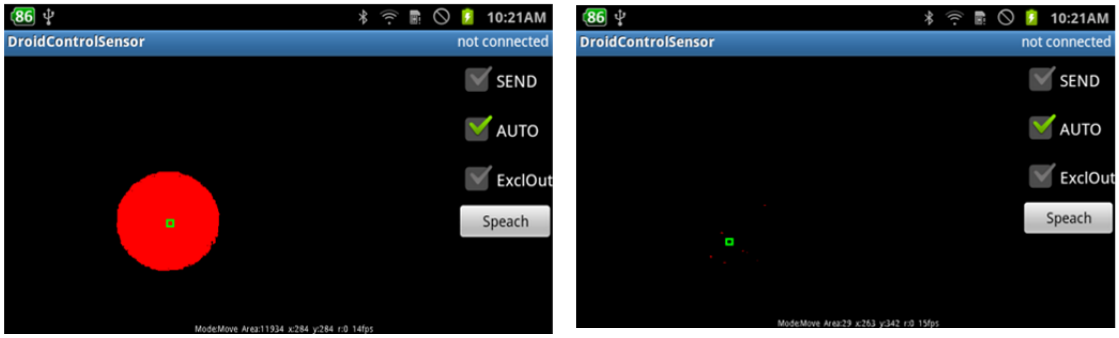

図 5.5　RGB モデルでの実験. 実験前(左)　実験中(右)

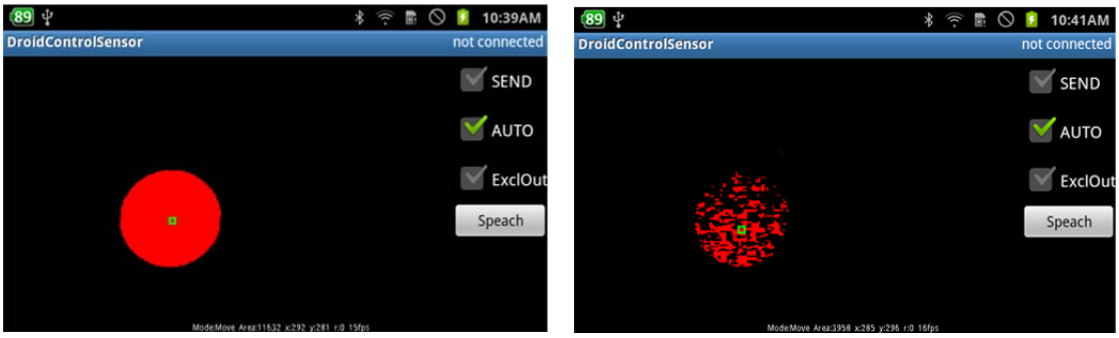

図 5.6　HSV モデルでの実験. 実験前(左)　実験中(右)

5.3 画像処理による視覚情報のフィードバック

追従動作は対象をロボットの正面に捉え続ける必要がある．そのため対象を左右に動かしそれを追う視点制御を行う．これらの処理を行うために，抽出した対象の特徴量について計算する．詳しい方法については以下に説明する．

5.3.1 PID 制御

PID 制御はフィードバック制御の一種で，Proportional Integral Derivative 制御の略である．制御を行った結果と目標値との　偏差：Proportional　その積分：Integral　微分：Derivation から次の入力値を推定し制御する．本研究では，これらの要素のうち，PI 要素を用いた PI 制御によってロボットを自律制御する．

比例要素は常に目標値との偏差をとりそれを制御に反映するが，目標値が周囲の環境などによって変化し，本来収束する筈であった値と一定値離れた値に収束してしまうことがある．これを残留偏差(図 5.8)という．この残留偏差を 0 にするために必要な要素が，積分要素である．

積分要素は偏差の積分で，偏差がある状態が続くと制御値を増やす働きがある．この要素を追加すると，残留偏差が発生していた場合，その偏差を無くすように制御値が増減される．この PI 制御を用いる事によって，画像処理によるロボット制御を行う．PI 制御の式は，制御量をC，偏差をΔV，比例要素のゲイン，積分要素のゲインをそれぞれK_p ，K_iとすると，式 5.8　で与えられる．

$$C = K_p \Delta V + K_i \int \Delta V \, dt \qquad \text{式 5.8}$$

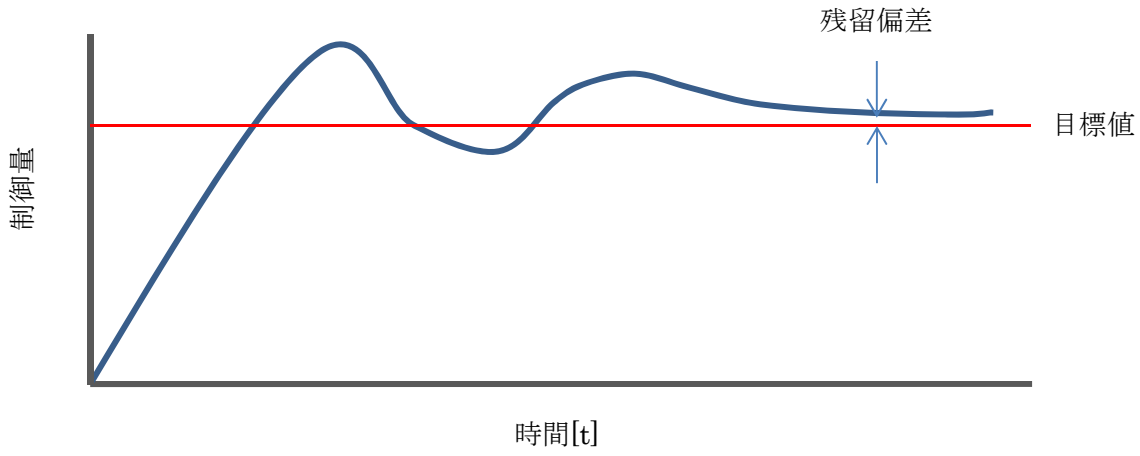

図 5.7　PI 制御における制御値と残留偏差

5.3.2 視点制御

対象をロボットの正面に捉え続ける為には，その対象の中心となる座標を検出する必要がある．二値化された対象の重心を取得し，その座標が常にカメラ画像の中心に位置するように制御を行う．図 5.8 はそのイメージ図である．画像の中心となる緑色の垂直な線と，対象を抽出した赤い円の中心座標との偏差を取得し，これが 0 となるような制御を行う．この制御は対象との距離は関係なく，その重心の方向に常にロボットの姿勢を制御することになるため，オムニキットを回転させることによってこれを実現する．

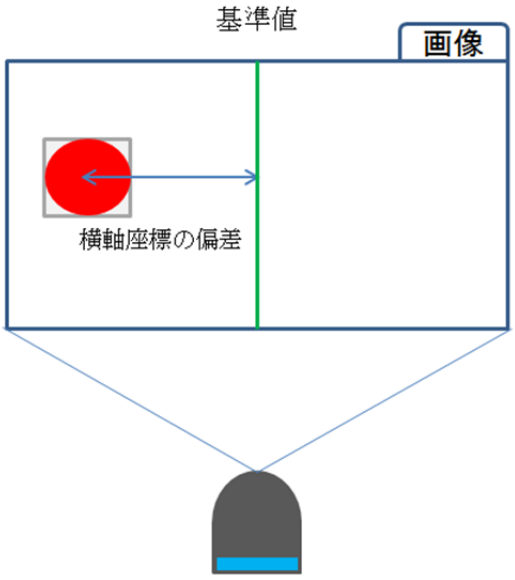

図 5.8　視点制御の基準値と目標の偏差

6章 動作実験

6.1 実験方法

動作実験は対象をロボットから見て左右に動かしそれを追う視点制御とする(図 6.2). 色情報による二値化を行うので周辺から走査する色と似通った色や，その他の彩度の高い物を排除し，平坦な場所で動作実験を行う．追従対象は図 6.1 のような側面に赤い色が描かれた 45mm×45mm×45mm の立方体とする．これを手動で動かし，ロボットに追従させる実験(図 6.2)を行った．

実験の環境は，周囲から彩度の高い物や誤検知の可能性がある色のものは排除し，地面は平坦な場所を選んだ．照明は室内を想定しているので，蛍光灯を点灯しその下で実験を行った．

またこの実験はロボットの動作の確認に加え，PI 制御のパラメータを決定する事も目的としているため，比例ゲインと積分ゲインの数値を変え実験を行い，適切な値を探った．

図 6.1　追従対象

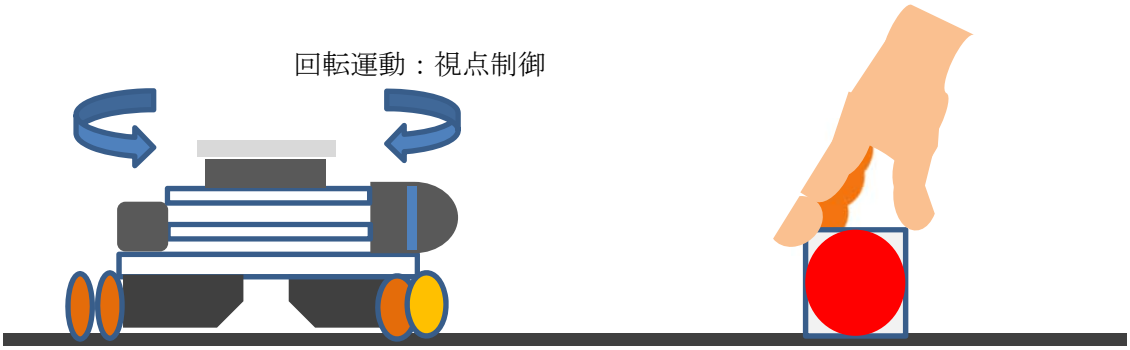

図 6.2　実験方法

6.2 視点制御実験

図 6.3 は対象をカメラの正面に捉える視点制御の実験方法を示している．追従対象はロボットから200mm離れた正面に据え，その状態で制御を開始する．そして追従対象を50mm真横にずらし，後方の車輪で機体を回転させることによりそれを追従させる．制御ログはスマートフォンのアプリケーションによって対象と目標値との偏差，及び経過時間をcsvファイル[14]として SD カードに出力した．また実験結果として出力された csv ファイルは Microsoft 社製の Excel で編集しグラフにおこした．

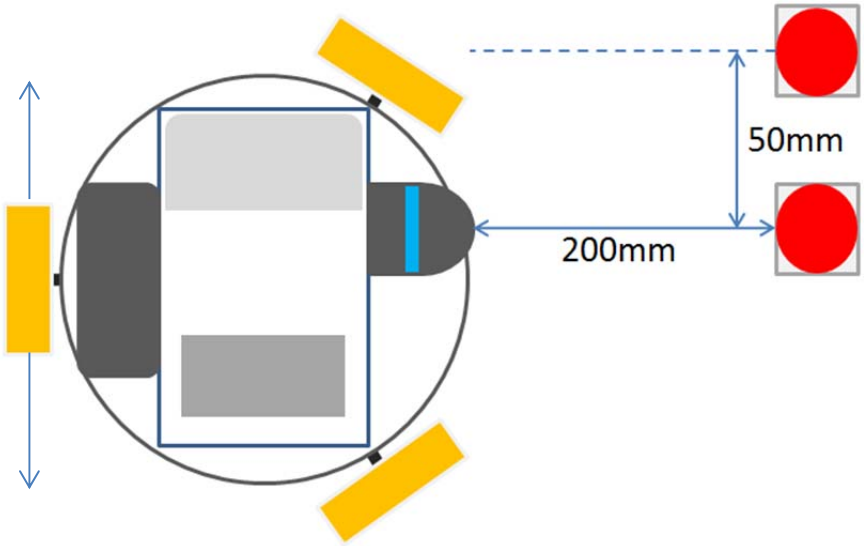

図 6.3　視点制御の実験方法

[14] Comma-Separated Values の略．いくつかの項目をカンマで区切ったテキストファイル．

6.3 視点制御実験　実験結果と考察

6.3.1 実験結果及びその考察

視点制御の結果を以下のグラフに示す．グラフは横軸に経過時間を，縦軸には目標値との偏差[pixel]を表している．

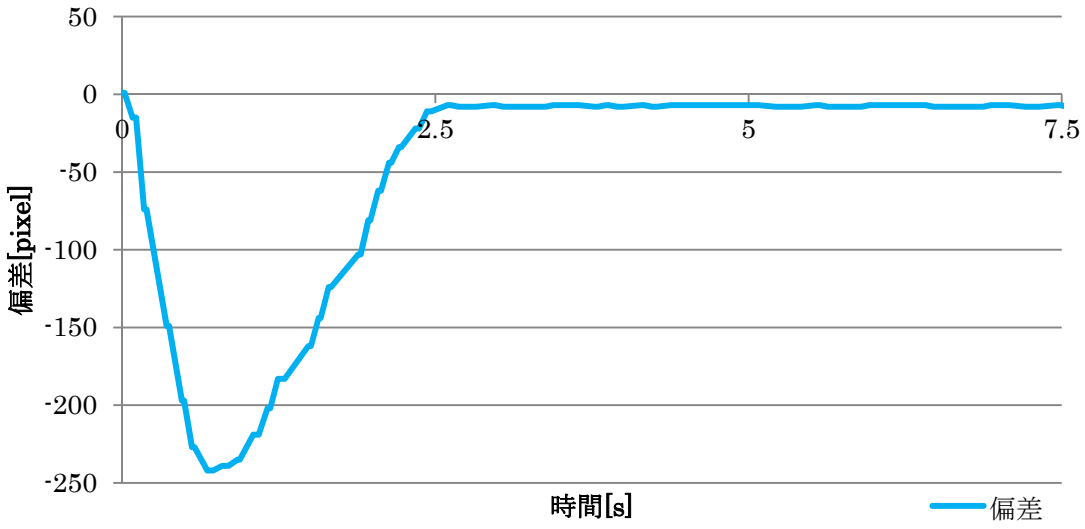

図 6.4　K_p: 0.2 K_i: 0.0　の実験

図 6.4 のグラフは積分要素を排除して制御を行った結果である．この時のゲインは比例値が 0.2，積分値が 0 となっており比例要素だけの制御を行っている．対象を移動してから約 2.5 秒で値が一定値に到達し，その後一定の偏差を残し停止してしまっている．これはこの偏差の時，モータに印加される電圧が起動トルクを下回るほど微弱であるためであると考えられる．こうなってしまうと以降偏差が変化することはなく，比例制御によって出力される電圧も変化しないため目標値との偏差を残し定常値をとる．このような偏差を収束させるためには，偏差を時間で積分した値を制御に取り入れる必要がある．積分要素を取り入れると，一定の偏差を残した状態が続いた場合制御量が増大する．ただし積分要素のゲインが大きすぎると収束せず，逆に図 6.5 のように目標値の周りで振動してしまう．

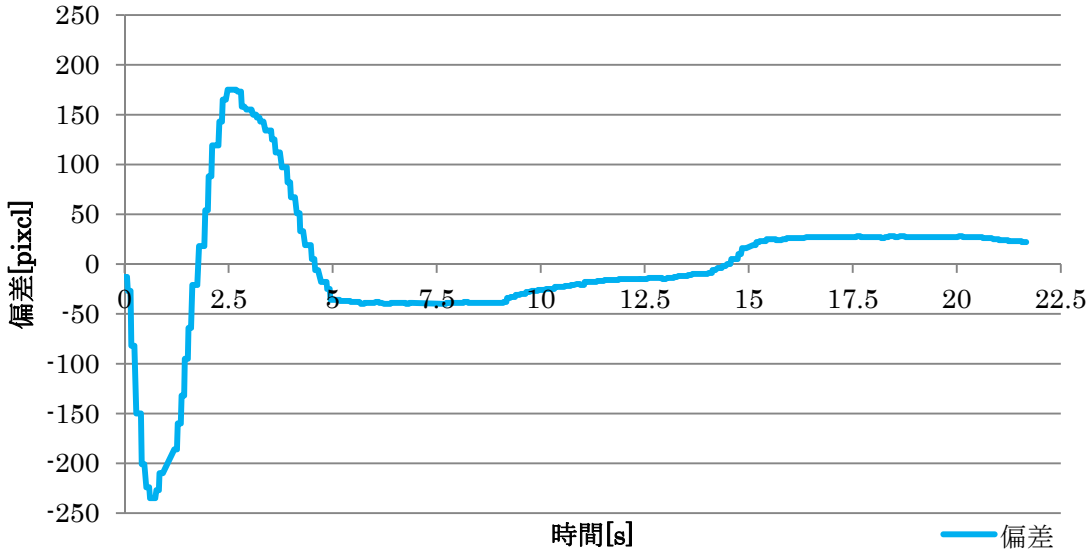

図 6.5 K_p: 0.35 K_i: 0.15 の実験

図 6.5 では目標値に到達するまでの時間こそ図 6.4 よりも早くなっているが，比例や積分要素のゲインが適正でないことにより制御量の超過により目標値を大きく上回ってしまい，制御開始から動作が安定するまでに約 5 秒もの時間が経過してしまっている．積分制御を取り入れたことにより目標値まで制御運動が継続されているが，このようなゲイン値をとると大きなオーバーシュオートが発生してしまう．

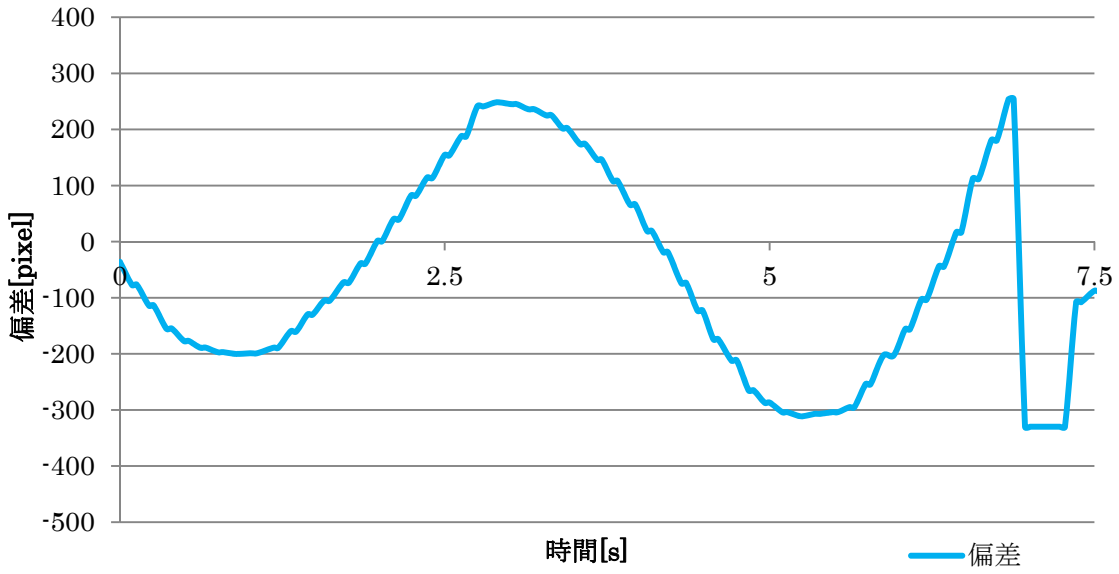

図 6.6 K_p: 0.6 K_i: 0.0 の実験

図 6.6 は極端に比例要素のゲイン値を大幅に上昇させたもので，徐々に振動が激しくなり，グラフが発散してしまっているのがわかる．また 7 秒あたりでグラフが急激に変化しているのは，画面上から対象が消えて，見失ってしまっている状態となっている．比例要素のゲイン値を極端に上昇させるとこのような挙動をしてしまう．

そこで比例ゲインを大幅に下げると図 6.7 の制御実験のように応答時間が極端に長くなり，目標値に到達するまでに長時間が経過してしまう．このように極端なゲイン値をとると制御が発散してしまう，逆になかなか目標値に到達しないなどの不安定なものとなってしまう．

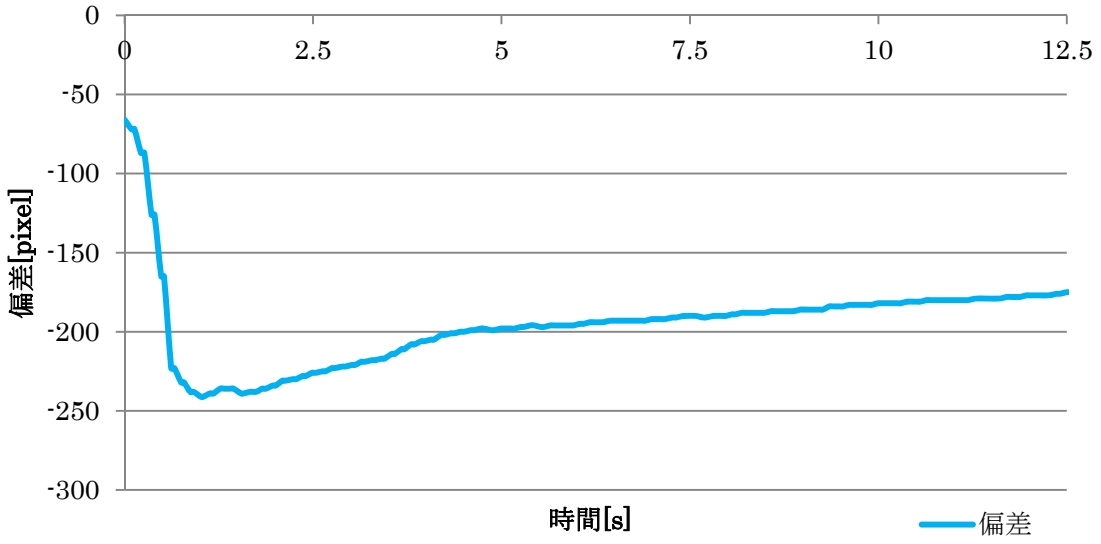

図 6.7　K_p: 0.15　K_i: 0.0　の実験

一方図 6.8 の制御実験では図 6.5 と比べ，より緩やかに少しずつ値が目標値へ収束しようとしている事がわかる．比例要素は図 6.8 より大きく積分要素は小さくなっているが，オーバーシュートが発生していることは変わらず，その値も大きくなってしまっている．これらのことからオーバーシュートは比例要素のゲイン値により発生していると考えられる．

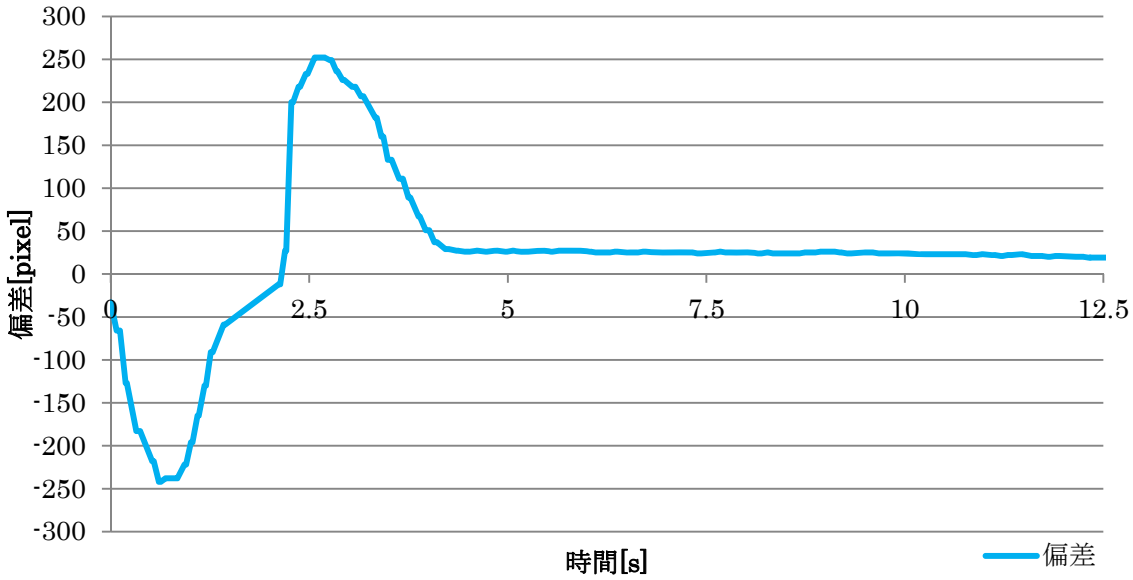

図 6.8　K_p: 0.4　K_i: 0.05　の実験

比例要素，積分要素それぞれのゲイン値を抑えた図 6.9 の制御実験では一度機体を振るだけで，オーバーシュートも発生せず，そこから目標値まで徐々に収束しようとしていることがわかる．よって左右に対象を回転，追従するような制御ではこのときの比例ゲイン，積分ゲインの値をとるのが良いと考えられる．

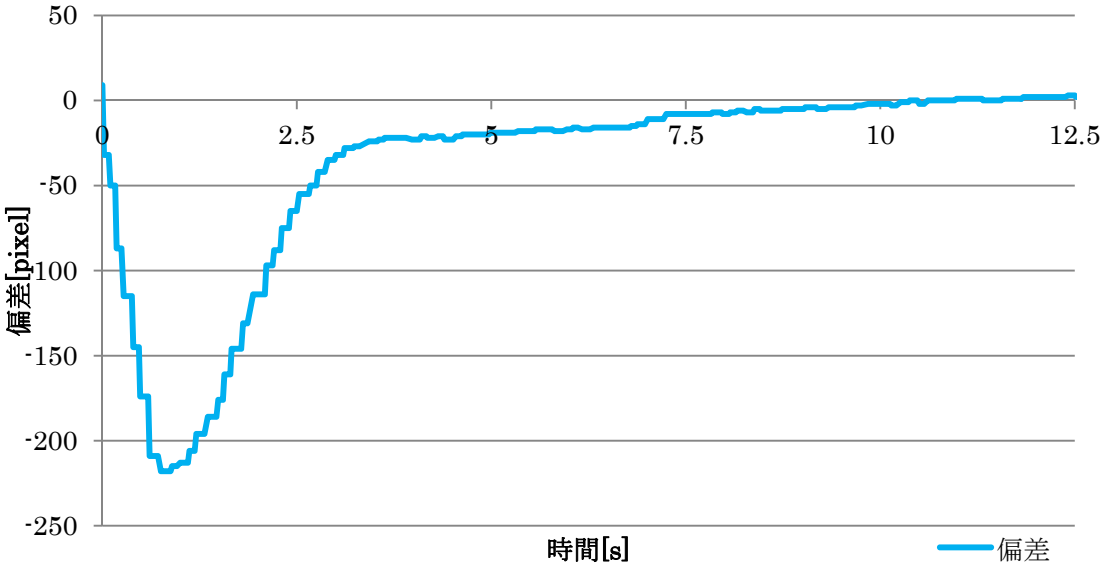

図 6.9　K_p: 0.2　K_i: 0.08　の実験

7章 総括

本研究では画像と音声情報を用いた自律型ロボットに関する研究を行った．掃除用のロボット家電やスマートハウスなどの高性能な家電製品が普及・発展し始めている現状を見るに，それらの機能を持ったロボットが家庭に普及していくだろうと考えた．これらのロボットは画像や音声などといったセンサを用いて制御を行えると考え，専用の API を用いて音声情報によって色情報を取得することに成功し，また視覚フィードバックによる PI 制御を利用した，色情報によるロボットの視点制御を実現した．

ロボットの駆動部にはオムニホイールという特殊な車輪を接続した三輪車両，オムニキットを用い，ロボットの安定性と，その場での回転動作などの機動性を両立した．このオムニキットを動作させるには 3 つの車輪の回転速度，及び回転方向を制御する事が要求されるため，PWM 制御とモータドライバを併用し，これを実現した．

ソフトウェアの基となる，ロボットとスマートフォンを繋ぐ通信部分には hrdakinori 氏が公開されている『DroidControl』『BT_DROID』というソフトウェアを元に，Wi-Fi カメラによって取得した画像の色情報を用いた二値化を行い，抽出した色の面積の中心を求め，常に対象をロボットの中心に捉えるような PI 制御を行った．PI 制御のゲイン値は実験により経験的に決定し，オーバーシュートもなく，目標値に収束するような値を用いて制御を行い，これを実現した．

しかし現在このロボットには自身の状態をセンシングするための内界センサが搭載されておらず，車輪の回転数や機体の傾き，速度，加速度を取得することができない．スロープのような接地面が斜面である環境での制御や，移動速度が高く周囲の人間に危険が及ぶような速度になった場合などの制御ができない状態となっている．またブレーキ機構などの減速機能も搭載しておらず，周囲に人がいる環境での自律走行を考えた場合危険である．カメラによる画像の取得はロボット前方の情報しかとらえることが出来ず，周囲環境に何らかの変化が発生してもこれを取得する方法がなく，非常に外乱に弱くいシステムである．

また画像処理にも問題が残っており，二値化処理を行った後もノイズが発生しているため，安定した対象物認識を行う為にはこれを除去する必要があるが，そのシステムも搭載されていない．HSV モデルにより認識能力は上がったが，あくまでこれは簡易的なものなので，実用性を求めるならばさらに何らかの画像処理を行い，対策を施す必要がある．

上記のようにロボットを PI 制御によって自律的に視点制御することを実現したが，前進・後退などの追従制御はできず，外乱に弱く，画像処理も簡素なものとなっている．自律行動により対象を追従移動するようなロボットにはまだ遠く，より多くの機能を搭載しなければならないだろう．

8章 謝辞

本研究を進めていくにあたり多くの人々にお力添えを頂いた事への感謝を，謝辞という形で記します.

まず初めに多くの助言を賜り，また多大なる迷惑をおかけしてしまった金丸隆志准教授にお詫びと深い感謝の意を表したいと思います．また本論文の執筆にあたりましてご助言を賜りました濱根洋人准教授，並びに見崎大悟准教授に厚く御礼申し上げます.

そしてプログラムの通信部分にあたる DroidControl・BT_DROID を配布されておりますhrdakinori 氏，参考にさせて頂いた書籍や論文の著者の方々にも深く御礼をいたします.

最後になりましたが，実験や研究を行うにあたり様々な協力を頂いた塙暁成氏，野崎祐基氏に感謝いたします.

参考文献

1. 長谷川修　森岡博史　李想揆　Tongprasit　Noppharit. 人混みでも環境地図を学習して稼働する自律移動ロボットを開発. 東京工業大学, 2010. 学内広報.
2. 南山尚久　安藤吉伸　水川真. 口唇領域の画像処理による音声認識補助システムの開発. 日本機械学会, 2007. 2P1-D11(1), ロボティクス・メカトロニクス講演会講演概要集.
3. 西田寿雄　金子正秀. ユーザからの曖昧さを伴った指示に基づく実環境内のオブジェクトの探索. 電子情報通信学会, 2003. P41, 電子情報通信学会技術研究報告.
4. 吉崎充敏　久野義徳. 対話と視覚情報処理を用いた環境情報の獲得. 電子情報通信学会コンピュータビジョンとイメージメディア, 2001. P157.
5. MM 総研. スマートフォン出荷台数の推移・予測. (http://www.m2ri.jp/newsreleases/main.php?id=010120120313500) MM 総研, 2012 年 3 月.
6. Nielsen 株式会社. ソフトウェアプラットフォームのシェア率. (http://www.itmedia.co.jp/promobile/articles/1205/08/news046.html) (編) 佐藤由紀子. 2012 年 3 月.
7. 松日楽信人　小川秀樹　吉見卓. 人と共存する生活支援ロボット. 東芝, 2005. SPECIAL REPORTS.
8. 後閑哲也. PIC マイコンではじめる作って遊べるロボット工作. 技術評論社, 2003.
9. 株式会社土佐電子. ロボットキット>22cm オムニキット. 株式会社　土佐電子　WEB SHOPPING. 株式会社土佐電子. http://www.tosadenshi.co.jp/cargo/goodslist.cgi?in_kate=10-5.
10. 土佐電子. オムニホイール> ウレタンオムニ. 株式会社　土佐電子　WEB SHOPPING. 土佐電子. http://www.tosadenshi.co.jp/cargo/goodslist.cgi?in_kate=20-3.
11. 荒垣龍馬. 正四面体全方向移動ロボットの走行制御. 日本機械学会, 2011. No.11-2, 日本機械学会発表論文　CD-ROM 論文集.
12. Microchip. PIC24FJFamilyReference. Microchip 社, 2010 年.
13. RT. RT CORPORATION. RT 公式ホームページ. 株式会社 RT. http://rt-net.jp/.
14. 東芝. MP4207　Reference. 2001 年.
15. MICROCHIP. Microchip PIC24F Peripheral Library. 2011 年.
16. FAIRCHILD. FXMA108　Reference. 2010 年.
17. 横溝淳. 強化学習を用いた二足歩行ロボットのための制御回路とソフトウェア環境の構築. 工学院大学, 2008. 修士論文.
18. Panasonic. 無接点対応 USB モバイル電源　QE-PL102. Panasonic. http://ctlg.panasonic.jp/product/info.do?pg=04&hb=QE-PL102.
19. BUFFALO. BSHSBD02　取扱い説明書. 2010 年.
20. hrdakinori. hrdakinori のいろいろ.

http://d.hatena.ne.jp/hrdakinori/searchdiary?word=Bluetooth.
21. Adobe. Technical Guides. Adobe 公式ホームページ. Adobe.
http://www.adobe.com/jp/support/techguides/color/colormodels/rgbcmy.html.
22. Trek. How　Ai-Ball　works. Ai-Ball 公式ホームページ. Trek.
http://www.thumbdrive.com/aiball/.
23. −. Ai-Ball　Reference.
24. Akira. 色相、彩度、明度の計算方法. イメージングソリューション.
http://imagingsolution.net/imaging/hue-saturation-brightness-calc/.
25. Microsoft. 色. Microsoft 公式ホームページ.
http://msdn.microsoft.com/ja-jp/library/aa511283.aspx.
26. 昌達慶仁. 詳解　画像処理プログラミング. SoftBankCreative, 2009.
27. 後閑哲也. C 言語ではじめる PIC24F 活用ガイドブック. 技術評論社, 2007.
28. MICROCHIP. 16 ビット言語ツールライブラリ. 2007 年.
29. 布留川英一. Android1.5　プログラミングバイブル. ソシム株式会社, 2009.
30. 藤田政之. 視覚フィードバック制御の現状と今後の展望. 金沢大学, 2003. 制御技術部会研究会.
31. 板谷友彰　金子正秀. 聞き手の立ち位置関係を調節するロボット. 社団法人映像情報メディア学会, 2011. vol.35　P107.
32. 光永法明　クリスチャン・スミス　神田崇行　石黒浩　萩田紀博. 方策勾配型強化学習によるロボットの対人行動の個人適応. 日本ロボット学会誌, 2006. vol.24　No.7　P820.
33. 村松光浩　大川茂樹　小林哲則　白井克彦. 読唇の併用による音韻認識. 電子情報通信学会, 1994. P47, 電子情報通信学会技術研究報告.
34. 明日香 Moonlight. moonlight_aska. Android プログラマへの道 〜
http://wiki.livedoor.jp/moonlight_aska/d/%a5%dc%a5%bf%a5%f3%a4%ac%b2%a1%a4%b5%a4%ec%a4%eb%a4%c8
http://wiki.livedoor.jp/moonlight_aska/d/%a5%aa%a5%d7%a5%b7%a5%e7%a5%f3%a5%e1%a5%cb%a5%e5%a1%bc%a4%f2%c9%bd%bc%a8%a4%b9%a4%eb.
35. adakoda. 逆引き Android 入門.
http://www.adakoda.com/android/000128.html
http://www.adakoda.com/android/000079.html
http://www.adakoda.com/android/000085.html.

図表目次