修士学位論文

論文題目　カメラとセンサを用いた

半自律型自動車ロボットの制御

ふりがな　のざき　ゆうき
氏　　名　野崎　祐基

専　　攻　工学研究科　機械工学専攻

指導教授　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

目次

概要

本研究は自動車の運転手がヒューマンエラーによって引き起こす事故を減らすことを目指している．そのような事故の例として，対向車線への逸脱が原因で起こる衝突事故，狭い高速道路のカーブで隣り合う車線にはみだすことが原因で起こる接触事故，前方不注意から起こる追突事故や人身事故などが挙げられる．この問題に対し，近年自動車安全技術や防犯カメラ等様々な分野で利用されている画像処理，および普及を続けているスマートフォンに着目し，これらの事故を減らすシステムを構築することを着想した．スマートフォンを画像処理エンジンとすることで高価な専用 LSI を用意することなく画像処理を行うことができるメリットがある．さらに，人や物などの接触から起こる事故に対しては自動車前面にセンサを設けることで前方監視を行い，人や物が一定距離以下に接近した場合に自動でブレーキをかけ停止させるシステムや，カメラを画像処理エンジンであるスマートフォンに接続し，走行車線判別を行うことで走行車線を認識し車線逸脱を防ぐシステムなどを考案した．

これらのシステムを実車を想定してラジオコントロールカーに実装し動作を確認することができた．

第1章　序論

1.1　　研究背景

自動車にはさまざまな電子制御やコンピュータ技術が取り入れられており，カメラやセンサが取り付けられたものも今や珍しいものではなくなってきている．交通安全白書によると 23 年の交通事故死数は 11 年連続で減少し 4,612 人となり，昭和 45 年の 3 割以下になっている [1]．これは交通安全に対する意識の向上，信号機や道路標識や防護柵といった設備が充実してきた点だけでなく，自動車技術の発達とともに車両周辺監視技術等が高められてきている点も原因であると推測される [2]．このことから今後の自動車技術は事故が起きてからの被害を少なくすることよりも事故を未然に防ぐことに重点が置かれるだろうと考えられる．

しかし夜間の視覚補助を行う赤外線カメラや車両後方に付けられたコーナーセンサ等，現存する各種安全装備は運転手の周囲確認を補助するに留まっており，最終的には運転手依存になるため事故防止の確実性はない．

1.1.1　　世界が取り組んでいる自律走行車

現在，世界中の自動車メーカー各社が交通事故を無くすという課題に取り組んでいるが，その中の筆頭が自律走行自動車である．情報技術の発達によりコンピュータが周辺を判断し走行する自律走行自動車が可能になりつつある．これらは主に画像処理や GPS，レーザーセンサなどを複合することで自律走行する．例えば米 Google はトヨタ自動車のプリウスをベースとした自律走行車を試験開発している（図 1.1）[3]. 2012 年 5 月にはネバダ州において公道での運転試験が許可され，更にはカリフォルニア州でも実験中の自律走行車が目撃されている．また，2012 年 8 月には自律走行車の試験運転距離が 48 万キロを無事故で走破しており，特許の出願もしている [4].

図 1.1　Google 社の自律走行車

また，英国リカルド社主導により自動伴走技術 SARTRE（環境に配慮した安全なロードトレイン）というプロジェクトが行われている．ロードトレインは，図 1.2 のようにプロのドライバーが運転する先導車があり，その後ろに数台の自律型自動車が続く形がとられており，まさしく路上の電車である．安全で環境にやさしい交通システムを目指すための EU 第 7 次枠組み計画の一環として欧州委員会によって出資も受けており，自動車メーカーのボルボ社も参加している．ロードトレインの公道実験では，バルセロナ郊外の高速道路で一般車が走行する中で実験を行い 200km 走破している [5].

随伴してくる後続車にはカメラやレーダー，レーザーセンサなどの機能を含む既存の安全システムを搭載し，前後の自動車をモニターする．また，各車両でワイヤレス通信を行うことで先導車の加速度やブレーキ状況をモニターし，その動きを模範することで正確な随伴走行を行っている．

図 1.2　SARTRE プロジェクトのロードトレイン

また，図 1.3 はトヨタ自動車系列のレクサスが，同社の大型セダン LS をベースとして開発した自律走行車であり，アメリカ･ラスベガスで開催されたコンシューマエレクトロショーで披露された．これはステレオ型ハイビジョンカメラを取り付け前方監視を行い，GPS で車両の向きや角度を検出する．さらに 360 度レーザートラッキング技術を採用し，周囲の状況を監視する [6].

図 1.3　レクサスの自律走行車

この自律走行車は，これらのセンサ類で自律走行しつつ地図データを取って技術の蓄積をしており，現在も開発を続けている．

このように自動車業界だけではなく様々な企業が自動車の安全に対して自律走行車両の研究開発を行っているが，これらは実験段階であり実用化にはいたっていない．

1.1.2　半自律型自動車

本研究で取り扱う半自律型自動車は，自律走行自動車の機能を簡易化したものである．通常はドライバーが運転しているが，何らかの危険が起きた時に自動車自身が判断しブレーキをかける等，ドライバーの意思とは関係なく自律的な行動をとるものを半自律型自動車呼ぶことにする．

半自律型自動車の例として，近年知名度が上がった富士重工業のアイサイトと呼ばれるシステムがある．これはステレオカメラを使用した衝突回避システムであり，ドライバーの補助を行いつつ必要があれば自動車が自律でブレーキ操作を行う．このシステムはセンサによる物体感知や夜間の視覚補助用カメラなどよりも確実にヒューマンエラーによる事故を減らせるものである．富士重工業は 20 年近く前からこのようなシステムを取り入れている [7].

1.2 目的

1.1.1 や 1.1.2 で紹介したように， Google 社やトヨタ，富士重工業等多くの企業が自律型，半自律型の自動車を研究開発しているが，これらのシステムには高度な画像処理を行う為の専用 LSI が搭載されている．しかしこの LSI は，高い処理能力を有するが高コストなため，そのシステム全体のコストも上がってしまうことがある．それを解決する方法として本研究では，小型でもパソコンに近い計算能力を有し，それ単体に様々なセンサや無線通信機能を搭載したデバイスを画像処理エンジンとして代用する．それにより，画像処理専用の LSI より劣らない性能を持たせつつ，安価なシステムを構築することを目指す．完全な自律走行車を目的としないのは，ドライバーがシステム依存になり安全に対しての意識が薄れてしまうことを防ぐためである．また，主観的な意見にはなるが，自動車は自らが運転する方が楽しいものであってほしいと考えた. 1.1.1 で紹介したレクサスの自律走行車の発表時に登壇したマーク･テンプリン氏は，このシステムはあくまでドライバーが主で，システムは副操縦士のような立場にあり，ドライバーは運転に慎重でなければならならず，安全に対してより意識を高めるべきだと述べている．これらのことから通常時はドライバーが運転を行いつつ，システムは前方監視を行い衝突回避や危険回避を行う半自律型の自動車を作成する．

しかし，本物の自動車を使用して半自律型のシステムを作成する場合，実験を行う際に広いスペースが必要になることや，安全性の問題をクリアすることは困難であると考えた．そこで本研究では実車を 1/10 スケールで再現したラジコンカーを使用し研究を進める．ラジコンカーであれば加工が容易であり，実車のスケールモデルであるため実車を想定した実験環境を再現しやすく，安全面も確保できるといえる．

既存の自律型，半自律型の自動車と同様に画像処理を取り入れるが，この画像処理エンジンには近年発達を続けているスマートフォン用いる．これらは小型ながら高い計算能力を有しており，それ単体で様々な無線通信手段やセンサ類が搭載されたものである．

さまざまなスマートフォンが存在するなか，本研究では Android スマートフォンを使用する． これは，Android スマートフォンは開発環境が無償であり，アプリケーション開発が容易であるという利点があるためである．

図 1.4 は本研究で構築するシステムの全体イメージである. 小型 WiFi カメラを車体に搭載し，WiFi 接続にてスマートフォンに画像データを無線送信する．受け取った映像をスマートフォン上で画像処理行い得られた結果を Bluetooth で送信しラジオコントロールカーを制御する．また，距離センサを搭載することで前方の物体を感知し，一定距離以下までに停止しなければ衝突回避のために自律で停止する．コントローラ部はスマートフォンのみとなっており，制御部はすべてラジコンに搭載する．

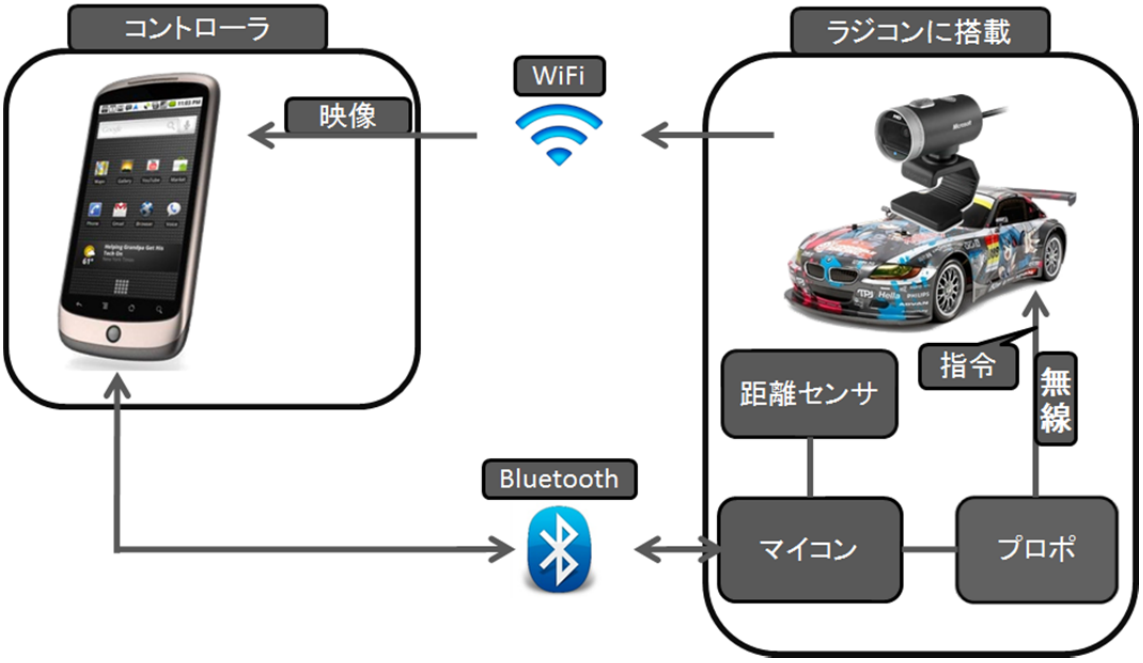

図 1.4　システム全体イメージ

1.3 本論文の構成

本論文の構成を以下に示す.

まず，第 1 章では本研究を行うに至った背景とその目的について述べる．自動車の安全技術やスマートフォンを取り入れることについての可能性などを示す.

第 2 章では本研究で自動車に代わり使用するラジコンカーについての詳細な仕様や特徴を述べる．実車での実験が困難なため，実車に近い挙動を取ることのできるラジコンカーを採用した.

第 3 章ではラジコンカーを制御するための制御部について詳しく述べる．マイクロコンピュータからシリアル通信で DA コンバータに命令を送信し電圧を出力する過程を記述する.

第 4 章では AndroidOS 搭載のスマートフォンからラジコンカーを制御する方法を述べる．マイクロコンピュータとスマートフォンを Bluetooth 接続し無線で命令のやり取りを行う.

第 5 章ではラジコンカーの自律停止機能の追加について触れる．低速走行中に一定距離以内に近づいた場合に自動で停止する機能である.

第 6 章では小型の WiFi カメラを利用し画像処理でラジコンカーを制御する手法について述べ，実験を通して得られた結果を述べる.

第 7 章では本論文で得られた成果をまとめる.

第2章　ラジオコントロールカーの構成

2.1　　RC モデルの特徴

RC モデルとはタミヤ社製のラジオコントロールカー[1]の名称である．RC モデルとはタミヤの商標であり，同じくラジオコントロールモデルを作成している模型メーカーの京商は，R/C モデルという名前で商標登録を行っている．この RC モデルにも種類があり，燃料と専用エンジンで走行するエンジン RC モデル，バッテリーとモータで走行する電動 RC モデル，テールスライドを可能にするドリフト RC モデル，オフロードや断崖絶壁を駆け上がるビッグタイヤ&バギーといった様々な種類がある．これらはアクセルの加速度やステアリング角をプロポと呼ばれるコントローラで細かな操作が可能である [8]．また，サスペンションやディファレンシャルギアと呼ばれるカーブなどで外輪の取る半径と内輪の取る半径から生じるシャフトのねじれを解消し，内輪･外輪で回転差を生むことでエンジンのトルクをそれぞれに伝えることのできる差動歯車も装備されている．これらは近代の自動車であれば装備されており，サイズや重量等実車と異なる点も多いが，ラジオコントロールカーの中では RC モデルの取る挙動は実車に近いものになると考えられる．これを実験に使用することで実車に結果が得られると考える．

図 2.1　RC モデルとプロポ

[1] 一般的に無線操縦を行うものをラジコンと呼ぶが，これは株式会社増田屋コーポレーションの商標である．

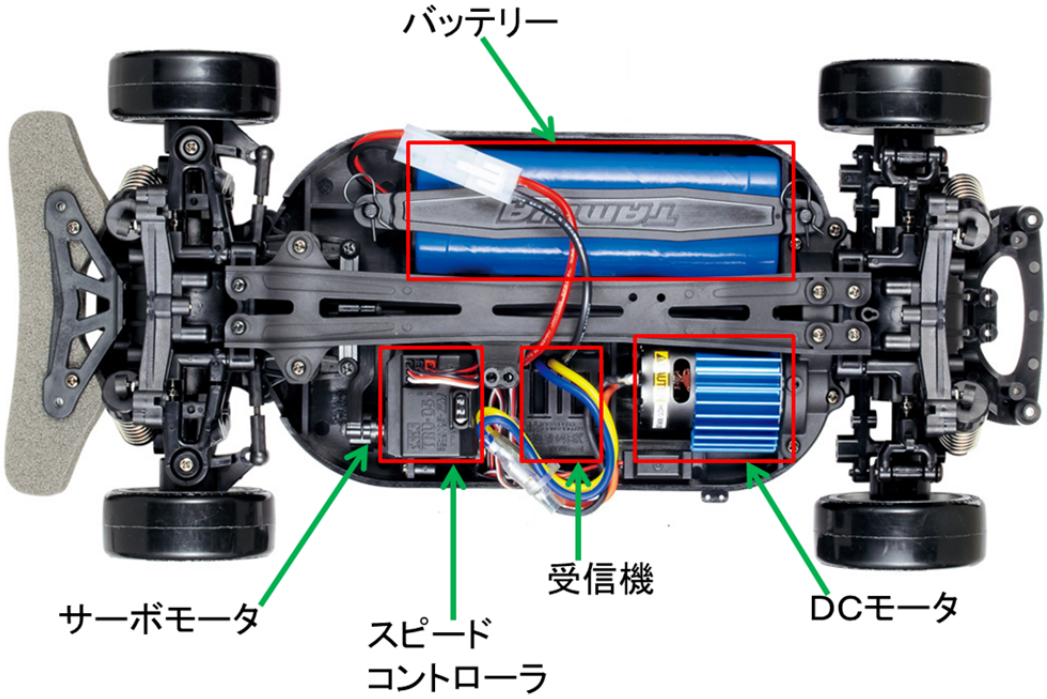

図 2.2　RC モデルの搭載する制御部

使用した RC モデルは図 2.1 のような 1/10RC ツーリング, 電動 RC モデルシリーズの組み立てキットである．ステアリングはサーボモータ，アクセルには DC モータを採用しそれぞれが独立して稼働する．またコントローラはガンタイププロポでアクセル，ステアリングの 2 チャンネルである．図 2.2 は RC モデルの搭載している装置であり，制御部としてサーボモータ，スピードコントローラ（以下 ESC），受信機の 4 つで 1 セットである．

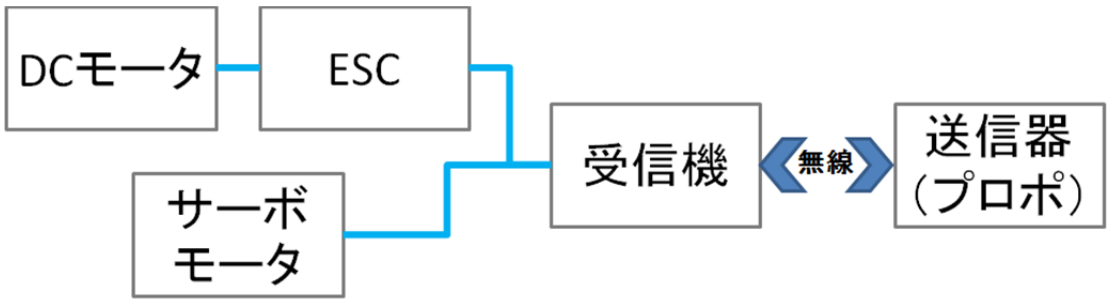

図 2.3　ラジコン制御系統の簡易図

電動ラジコンカーは図 2.3 のように，プロポから受信機に制御命令が送信され受信機で受け取った後，サーボモータと DC モータを制御する．2 チャンネル・1 サーボ・1ESC の制御部から動作している．

コントローラと受信機の通信規格は FM ラジオ無線方式であり，RC モデルに使用されている電波は 27MHz 帯，40MHz 帯の 2 波である．この帯域を細かに分割して使用することで同時に複数の RC モデルを走行させることが可能である[2]．今回使用した周波数バンドは FM27MHz 帯 26.995MHz である．

このコントローラの信号を受信機が受け取りステアリングとアクセルの制御を行う．ステアリングは受信機から直接サーボモータを制御するが，アクセルに関しては ESC を介して制御を行う．これは前進後退に使用している DC モータの制御に PWM 制御を行っている為である．図 2.4 のように PWM 制御を行うことで回転数を制御し前進，後退，加速，減速を可能にしている [9].

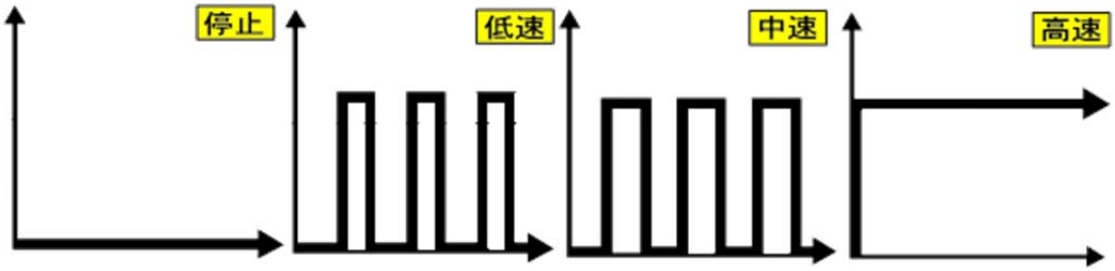

図 2.4　モータの PWM による速度制御

このような RC モデル等の速さや周回数を競うレースのルールとして一切のバック走行が禁止されていることが多いため，ESC 本体にバック走行そのものをキャンセルできる機能を持つものもあり，さらにメーカーによってはそもそもバック走行というものがない ESC もある [10].

[2] なお，40MHz タイプの周波数帯を使用する場合は(財)日本ラジコン電波安全協会に規定料金を納めて登録を行う必要がある．

本研究で使用した ESC はタミヤ社製の TEU-104BK（図 2.5）でありバック走行，バックキャンセル機能付きで，現在の受信状況を知らせる LED がついているものである． この LED はスロットル操作と連動しており，ニュートラル状態で消灯，前進か後退にスロットルを操作すると点灯，それぞれの最高速度になるとまた消灯する機能がある．

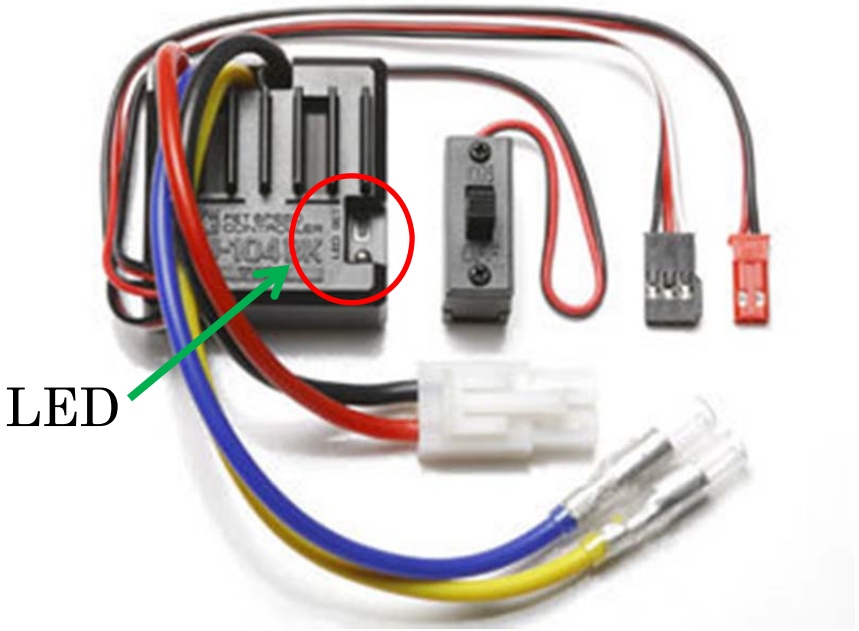

図 2.5　タミヤ社製 ESC　TEU-104BK

2.1.1　RC モデルの仕様

以下は本研究に使用した RC モデルの詳細である．

表 2.1　1/10RC モデル詳細

全長	431mm	デフギヤ	FRとも3ベベル
全幅	187mm	ステアリング	3分割タイロッド式
全高	125mm	サスペンション	4輪ダブルウィッシュボーン
ホイールベース	251mm	ダンパー	FRともフリクション
タイヤ幅/径	FRとも27/67mm	ギヤ比	8:35:01
フレーム	バスタブタイプ	モータ	540タイプ
駆動方式	シャフトドライブ4WD	スピードコントローラ	ESC仕様

表 2.1 にあるフレームのバスタブタイプとは，シャシー部をバスタブのように箱状にすることで，バッテリーやモータなどをより低い位置に設置でき車高を落とすことなく低重心を実現できるものである．ディファレンシャルギアに使用しているギアはフロント，リアともにベベルギアである，また，サスペンションダンパーはフリクションと呼ばれる乾式摩擦の抵抗で振動を減衰させるものである．サスペンション形式は上下 2 本のアームでタイヤを支持するダブルウィッシュボーン式のサスペンションである．

2.2　プロポーショナルシステム

プロポーショナルシステムはラジコンカーを人間の思うままに操るために必須なコントローラシステムであり，コントローラ，受信機，ESC，ステアリングサーボをまとめたものである．また，コントローラ単体でプロポと呼ぶこともある．

左右に曲がる，走る，止まるなどの単純な操作以外に，曲がる角度の微調整，速度の微調整，加速の加減など，実車のような複雑な動作も可能にするものである．これは一般的にアクセル，ステアリングともに ON/OFF 制御しかできないトイラジと呼ばれるものとの最大の違いである．

2.2.1　プロポの特徴

このプロポはスロットルトリガーを引く，ステアリングホイールをねじるといった簡単な操作で RC モデルの複雑な動きを可能にしている．このプロポを制御部に組み込むことで既存のシステムを生かしたまま RC モデルの制御ができる．

図 2.6　ガンタイププロポ

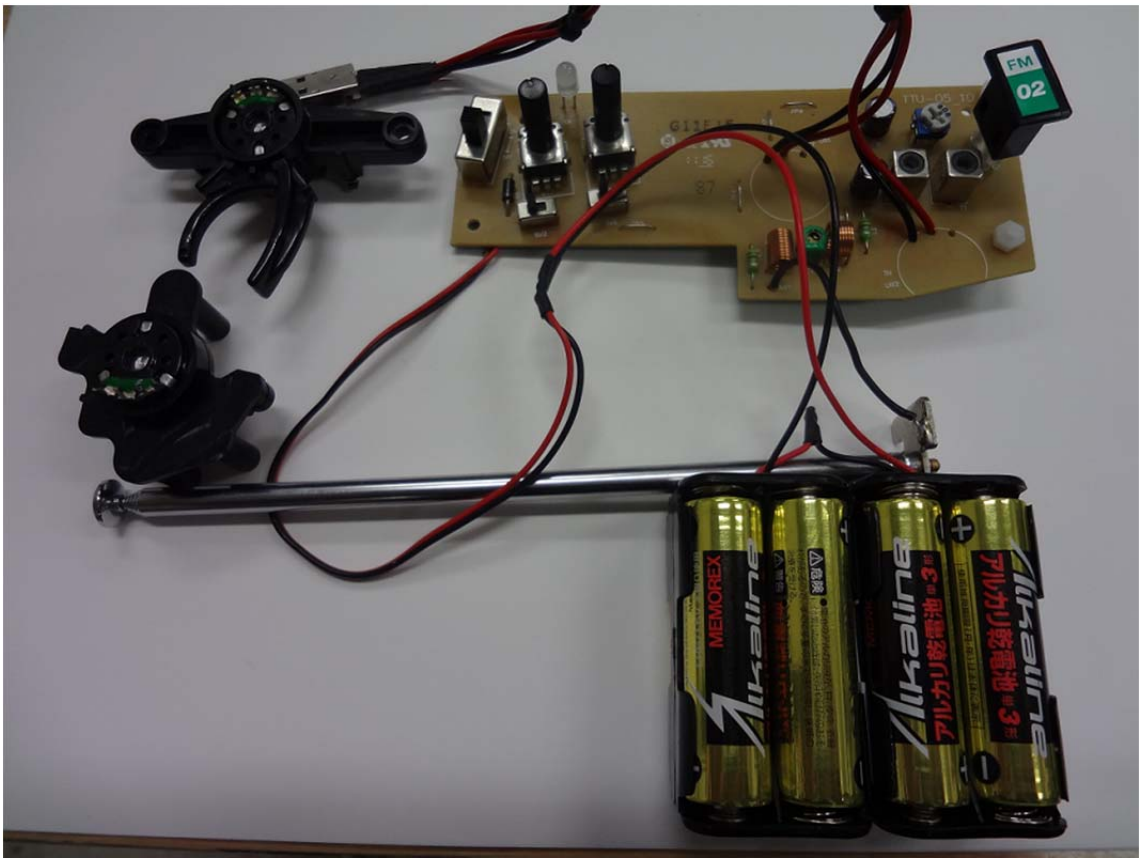

図 2.7　分解したプロポ

まず，プロポの内部構造は，図 2.7 のように外装，回路部，電池ボックスの 3 部分で構成される．回路部にはスロットルトリガー，ステアリングホイール，アンテナ，電池ボックスが接続されている．RC モデルを操作するためのスロットルトリガー，ステアリングホイール部は内部に可変抵抗が組み込まれており，電圧変化で RC モデルを制御している．

2.2.2　プロポの検証実験

まず，プロポを分解した状態でスロットルトリガーの前進，停止，後退時とステアリングトリムの左，ニュートラル，右時において図 2.9 中の赤い円で囲った部分の電圧を測定した．測定はプロポの電源を入れ，図 2.9 中赤い円プラス部と図 2.9 赤い内マイナス部（どちらでも可）の電圧をテスタで測定する．測定を行う時のスロットルトリガーとステアリングトリムの状態はそれぞれ，前進はスロットルトリムを最大まで引いた状態，停止は触らない状態，後退は最大まで押した状態である．また，ステアリングも同様に左はステアリングホイールを左に最大までねじった状態，ニュートラルは触らない状態，右はステアリングホイールを右に最大までねじった状態である（図 2.8）．

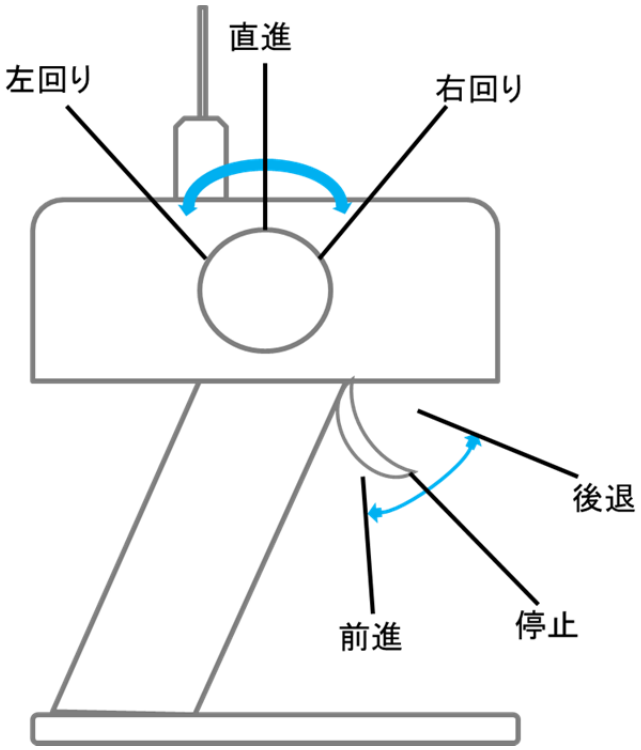

図 2.8　プロポの操作法

表 2.2 に測定値をまとめたものを示す[3]．

表 2.2　アクセルとステアリングの電圧関係

アクセル		ステアリング	
前進	2.74V	左	4.16V
停止	2.98V	ニュートラル	2.57V
後退	3.12V	右	1.09V

[3] なお，これらの値は実測値でありプロポの製造メーカー，各個人のプロポセッティングにより変化することがある．

この結果をもとにスロットルトリガー，ステアリングホイールの操作ではなく，その部分に直接電源から電圧変化を与えた場合に RC モデルが操作できるのかを実験した．まず，プロポ回路部分に取り付けられているスロットルトリガーとステアリングトリムを取り外し，実験で得られた電圧を一か所ずつ入力する．そして電圧を連続的に変化させ，RC モデルがトリムで操作した時と同様に動作するか検証した．

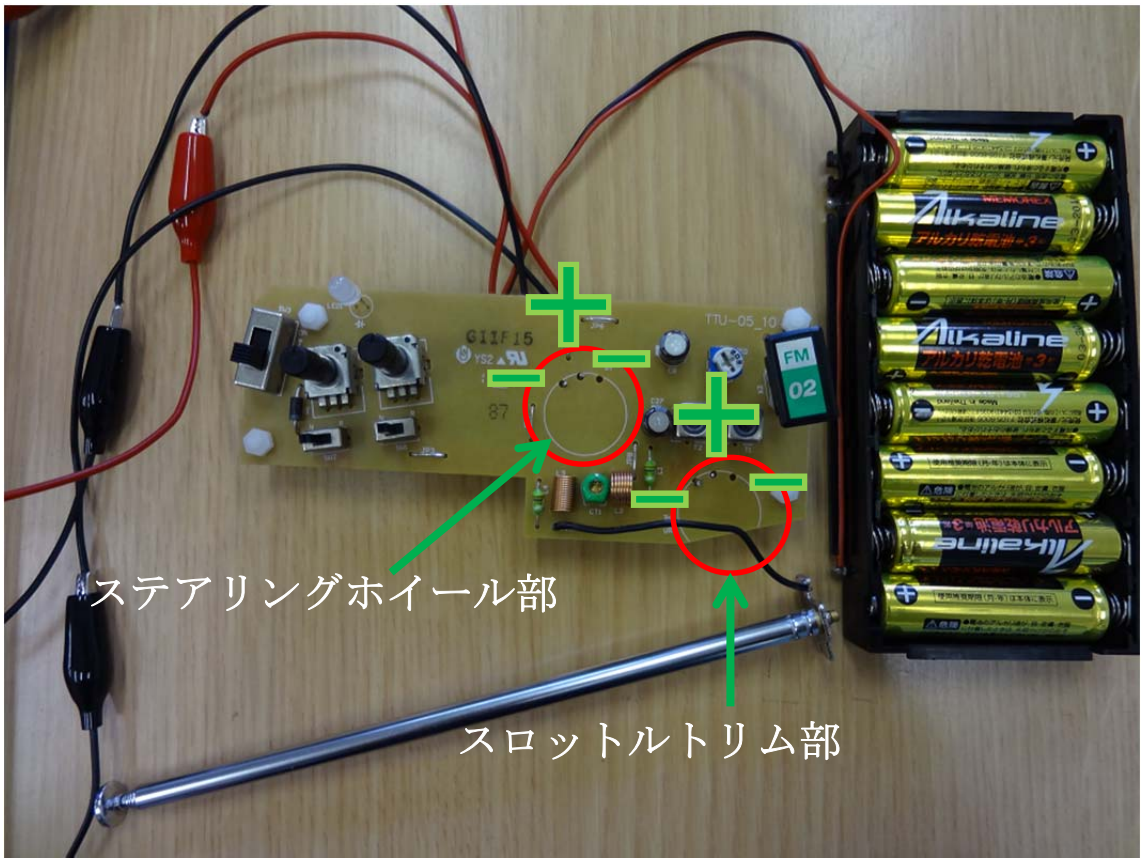

図 2.9　電源を直接接続した部分

図 2.9 はプロポの回路からスロットルトリガーとステアリングトリムを外した状態の写真である．トリム部の 3 つある端子は中央がプラス，左右がマイナスとなっている．実験はプロポと RC モデルの電源を入れ，この取り外したスロットルトリム，ステアリングホイール部に安定化電源を接続し，表 2.2 で測定した値を出力する．

まず，アクセルの停止状態である 2.9V を電源から出力し RC モデルの動作を見た．この時，電圧が正しく認識されているならば RC モデルの駆動は停止しており，そうでなければノーコントロール状態になる[4]．

[4] ラジオコントロールで動作する機器において操作している人間の意思に反した動作，または勝手に動作してしまう状態のことを指す．

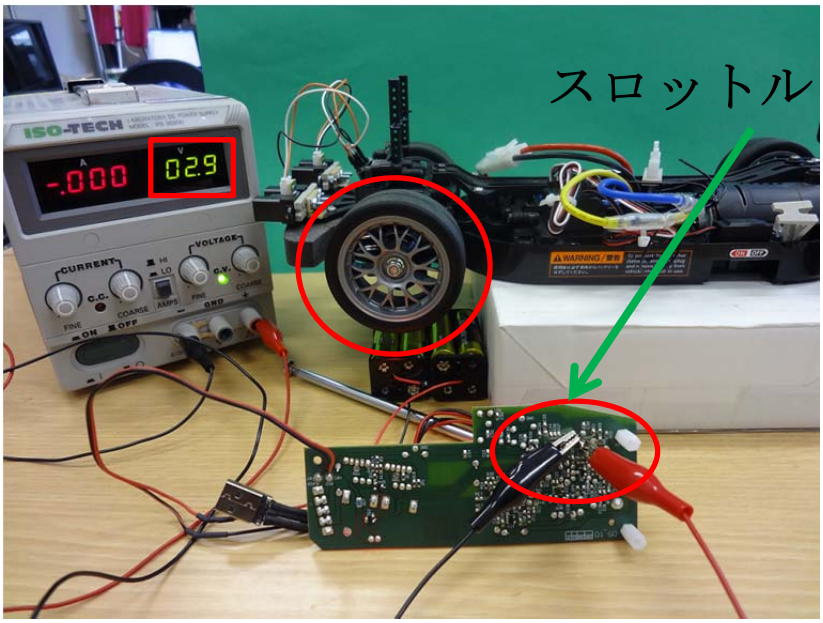

図 2.10　停止状態の電圧をプロポに入力

図 2.10 のように停止状態である 2.9V をプロポに入力した場合，RC モデルのタイヤは停止している為，外部からプロポに電圧を入力しても正しく反応することがわかった．この状態から電圧を前進の最高速度である 2.7V まで下げていく．

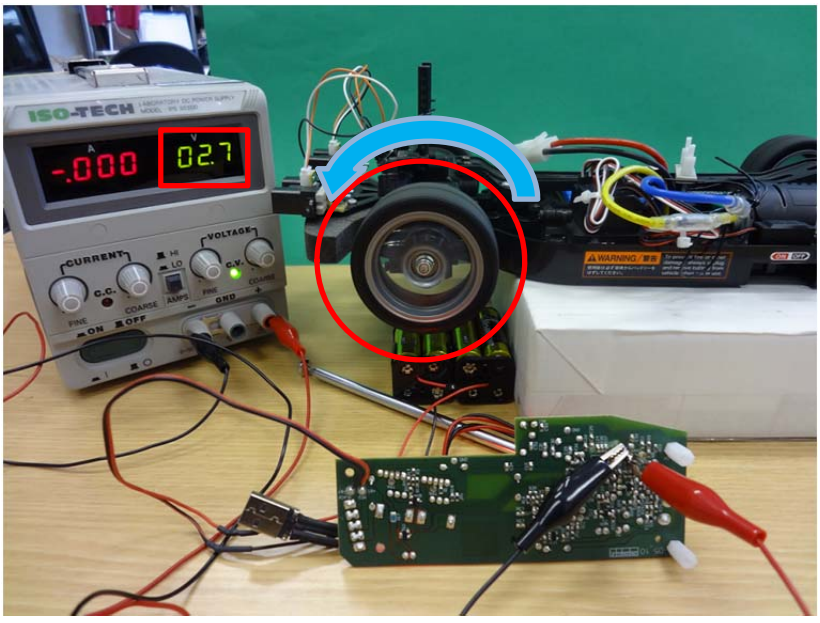

図 2.11　前進最高速度の電圧をプロポに入力

するとタイヤは徐々に前進方向に回転（加速）し始め，図 2.11 のように 2.7V で最高速度となった．最高速度の確認は 2.1 で解説した ESC に搭載されている LED で行った．次に，この状態から後退最高速度である 3.1V まで電圧を徐々に上げていく．

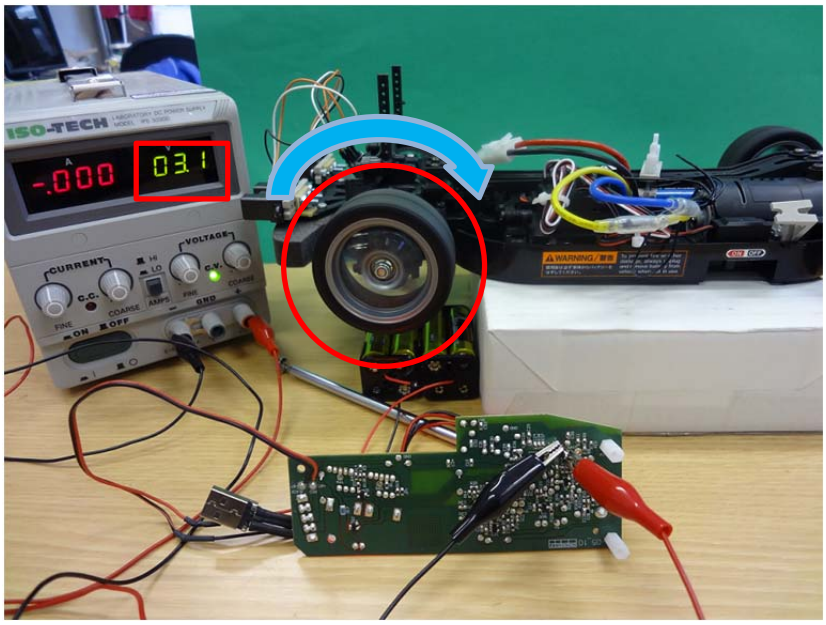

図 2.12　後退最高速度の電圧をプロポに入力

タイヤは徐々に減速し始め 2.9V で停止，その後後退方向にタイヤが回転（加速）し始め，図 2.12 のように 3.1V で後退方向最高速度となった．次にステアリング部にも同様の操作で実験を行う．

まず，プロポのステアリングホイール部に電源を接続し，ステアリングニュートラル状態である 2.5V を出力する．

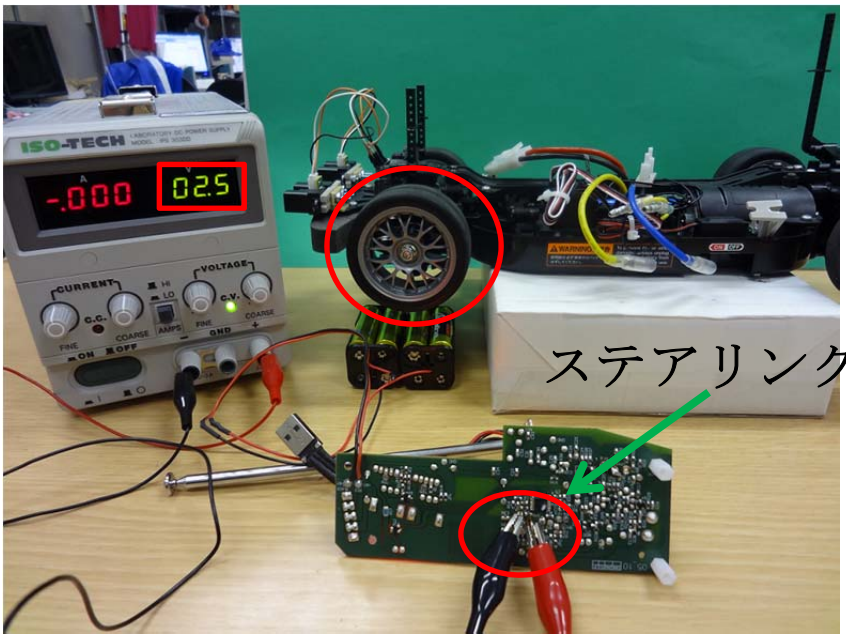

図 2.13　ステアリングニュートラル状態の電圧をプロポに入力

図 2.13 は，ステアリングのニュートラル状態である 2.5V をステアリングトリム部に入力した写真である．図のようにステアリングはニュートラル状態（直進方向）であり，これも正しく動作していることが確認できた．この状態から電圧を増減させステアリングの挙動を見る．まず左方向にステアリングが切れるか確認するため，4.1V まで徐々に電圧を増やしていった．

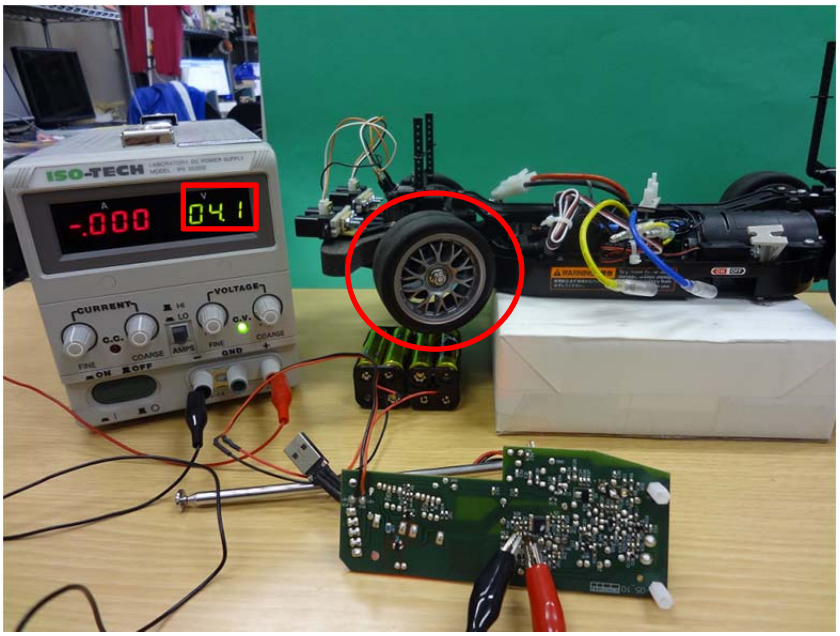

図 2.14　ステアリング左方向の電圧をプロポに入力

すると，電圧の増加に伴いステアリングは左方向に切れ始め，図 2.14 のように 4.1V で最大切れ角になった．次に，ここから右方向に切れるかを確認するために 1.0V まで電圧を下げていった．

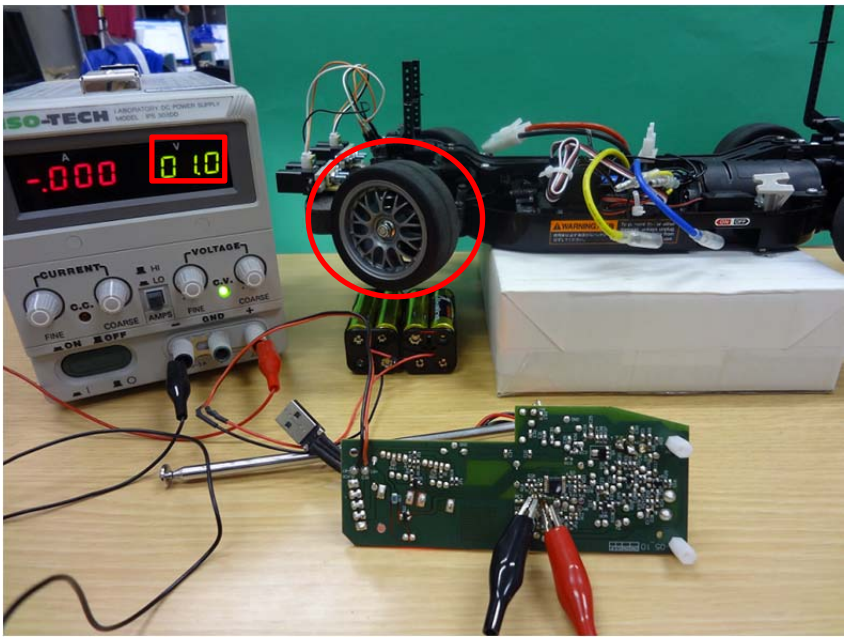

図 2.15　ステアリング右方向の電圧をプロポに入力

するとステアリングは徐々にニュートラル状態に戻り，電圧が 2.5V より下がると右に切れ始めた. そして図 2.15 のように電圧が 1.0V 付近で最大切れ角となった. なお，ステアリングの状態はアクセルのように ESC の LED では確認できないため，実際にステアリング（サーボモータ）が動かなくなる場所が最大切れ角となる．

これらの実験より，プロポのスロットルトリガー，ステアリングトリム部に表 2.2 の電圧値を入力することで RC モデルを制御できることが分かった．

第3章　制御回路

3.1　　本研究におけるシステム全体像

図 3.1 は本研究で作成するシステムの全体像である.

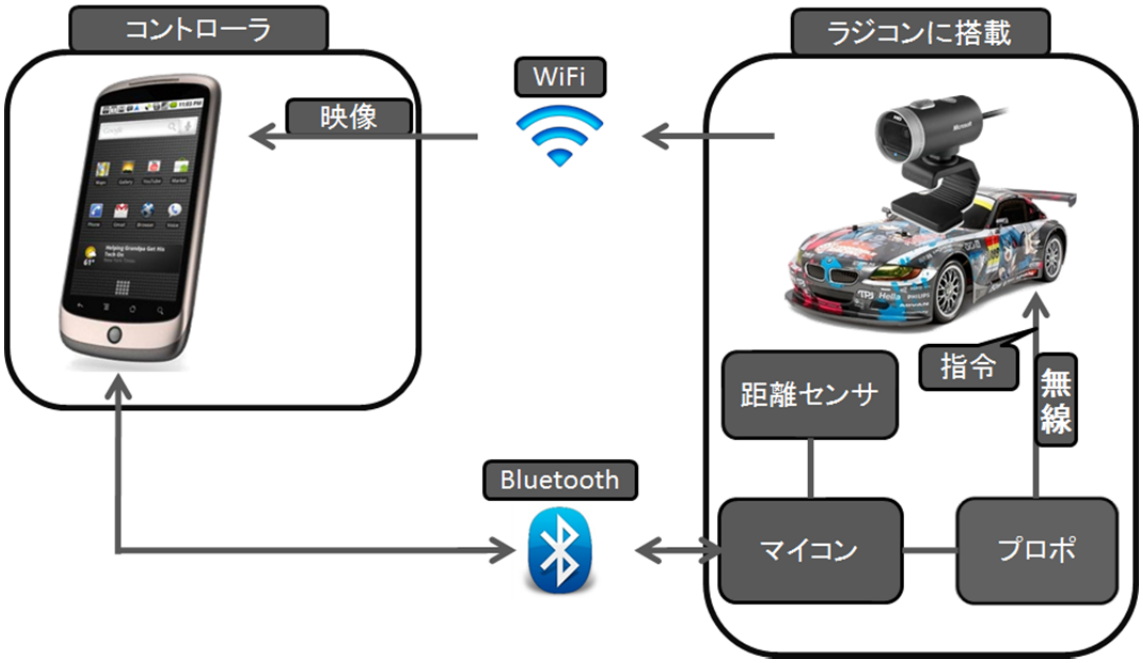

図 3.1　システム全体像

ドライバーによる手動運転，センサによる前方衝突回避，画像処理で白線を認識し走行車線中央を走行するようステアリングを補助する機能の 3 つの機能の実装を目指す.

まず，WiFi カメラを使用しスマートフォンに映像を送信し，取得した映像を画面上に表示する. この映像を確認しながらスマートフォンで RC モデルを操作するモードを手動運転モードとする．これはスマートフォンとマイコンを Bluetooth 接続し，マイコンはスマートフォンから取得した命令を基に電圧変化をプロポに入力することで実現する.

衝突回避は全てのモードで動作する．距離センサをマイコンに接続し，スマートフォンの命令とは一切関係なく常に前方監視を行い，一定距離以下に物体が近づいた場合に強制的に RC モデルを停止させる.

白線認識走行モードは，取得した映像を画像処理し白線を認識し，そこで得た結果を基に車線中央を走行するよう制御をマイコンへ送信する．受けとった命令から電圧変化をプロポのステアリングトリム部に入力することで RC モデルのステアリングを自律制御する.

3.2　RT-ADKmini の利用

RT-ADKmini は CPU に PIC24FJ64GB002-I/ML を搭載した RT 社製のリファレンスボードである．スマートフォンとマイコンを Bluetooth 接続するために USB ポートが付いているこのボードを選択した．

表 3.1　RT-ADKmini の仕様

RT-ADKminiRT-ADKmini	
サイズ	21(W)×62(D)×18(H)
重量	11g
CPU	PIC24FJ64GB002-I/ML
I/Oピン	15ピン
スマートフォンとの接続	USB Aタイプ

表 3.1 は RT-ADKmini の仕様である [11]．このボードは ADK と呼ばれる Android 端末と USB で有線接続することによりボードと Android 端末双方でのやり取りが可能な開発キットであるが，本研究ではこの ADK 機能は使用しない．また 28 ピンという豊富なピン数を有しており MPLAB を使用してプログラムを実装することで様々な機能を割り当てることが可能である．

3.2.1　RT-ADKmini と Android スマートフォンの連携

AndroidOS 搭載の機器と回路との連携方法は有線接続や Bluetooth 接続などいくつかある．本研究では図 3.2 のように RT-ADKmini に搭載されている ADK 接続用に設けられた USB ポートに Bluetooth ドングルを挿しこみ hrdakinori 氏のプログラムを使用することでスマートフォンと回路を Bluetooth 接続する [12].

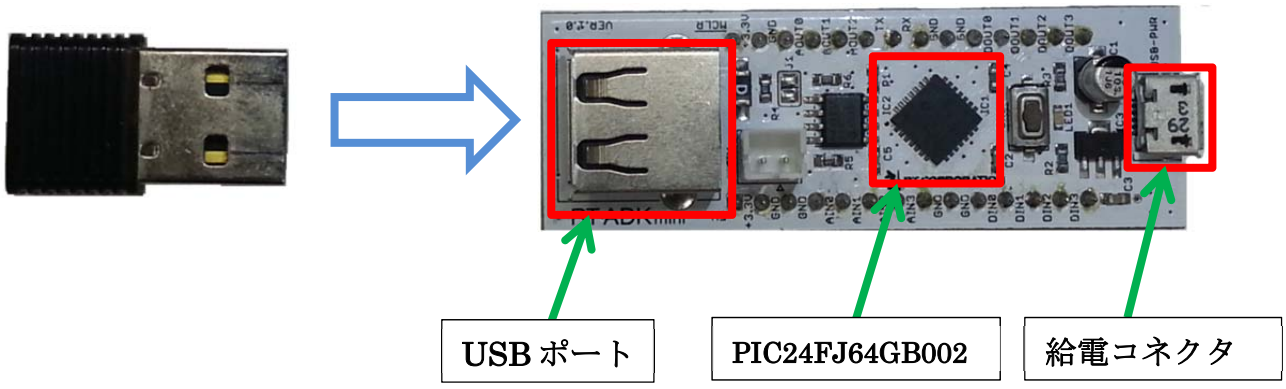

図 3.2　Bluetooth ドングル（左）と RT-ADKmini（右）

また，Android スマートフォンと連携するためには RT-ADKmini 側だけでなく，スマートフォン側のアプリケーションも必要であり，こちらも hrdakinori 氏のアプリケーションを使用し，本研究で必要になる機能を付け加えていく．

スマートフォンが送信した信号を RT-ADKmini が受け取りプロポに電圧変化を送信させることができれば RC モデルを制御することができる．そのためにはアナログ信号を出力しなければならないが，RT-ADKmini に搭載されている PIC にはその機能がない．そこで RT-ADKmini から DA コンバータを通すことでアナログ値を出力する．DA コンバータと RT-ADKmini とのデータの受け渡しには SPI 通信を行う．

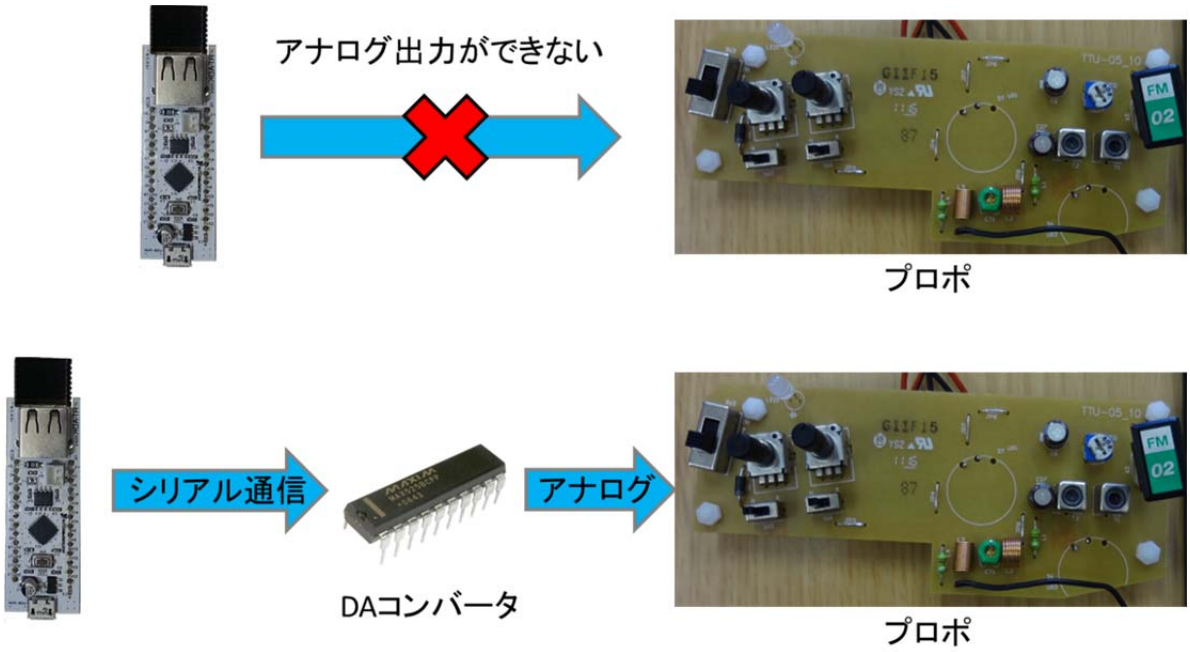

図 3.3　DA 変換によるアナログ出力

3.2.1　DA 変換

DA 変換にはマキシマム社製 MAN525ACPP という DA コンバータを使用する．これは 12 ビットデジタル―アナログコンバータであり，RT-ADKmini から MAX525ACPP へ SPI 通信を使いデータの受け渡しを行い DA 変換をする．MAX525ACPP に特別な設定は必要なく，SPI 通信で使用する決められたピンに接続し，ファームウェアで設定することで DA 変換が行われる．この MAX525ACPP の利用法については [13]を参考にした．

3.2.2　PIC によるシリアル通信

RT-ADKmini に搭載された PIC と外部の DA コンバータとのデータ通信はシリアル通信を使用する．

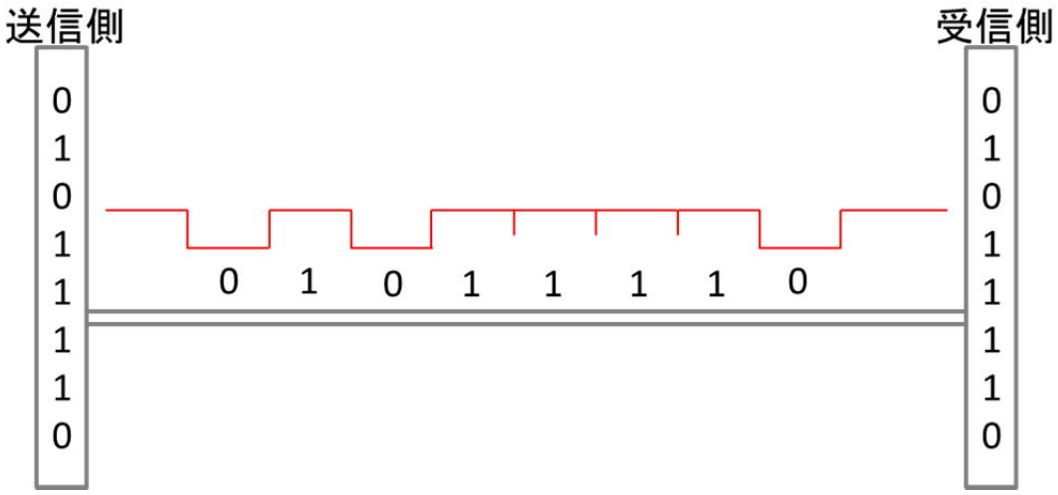

図 3.4　シリアル通信イメージ

シリアル通信とは 1 本の信号線や回線を使って図 3.4 のように 1 ビットずつ一定時間間隔で順番にデータを送受信する通信，転送方式のことである．コンピュータ内部の伝送路やコンピュータと周辺機器の接続などで用いられる [14].

このシリアル通信を利用して RT-ADKmini から DA コンバータに通信を行う．これには SPI（Serial Peripheral Interface）通信というシリアル通信の規格を利用した．

3.2.3　SPI 通信

MAX525ACPP とデータの受け渡しを行うために使う SPI 通信は，モトローラ社が開発したマイコン間やマイコンと外付け部品などで使われる同期式全二重シリアル通信である. 使用している RT-ADKmini に搭載されている PIC は PIC24F ファミリであるので SPI インターフェースが 2 組ある.

SPI 通信は 4 つの信号が必要であり，それらを通信したい相手と接続して通信を行う．この 4 つの信号は Serial Clock（SCK），Serial Data Input（SDI），Serial Data Output（SDO），Serial Slave（SS）である．これらそれぞれの役割は表 3.2 に示す.

表 3.2　SPI 通信における信号名とその役割

信号名（略称）	役割
Serial Clock（SCK）	マスタとスレーブのクロックを同期
Serial Data Input（SDI）	データ送信
Serial Data Output（SDO）	データ受信
Serial Slave（SS）	送信するスレーブの選択

また，SPI モジュールの内部構成は図 3.5 のようになっている.

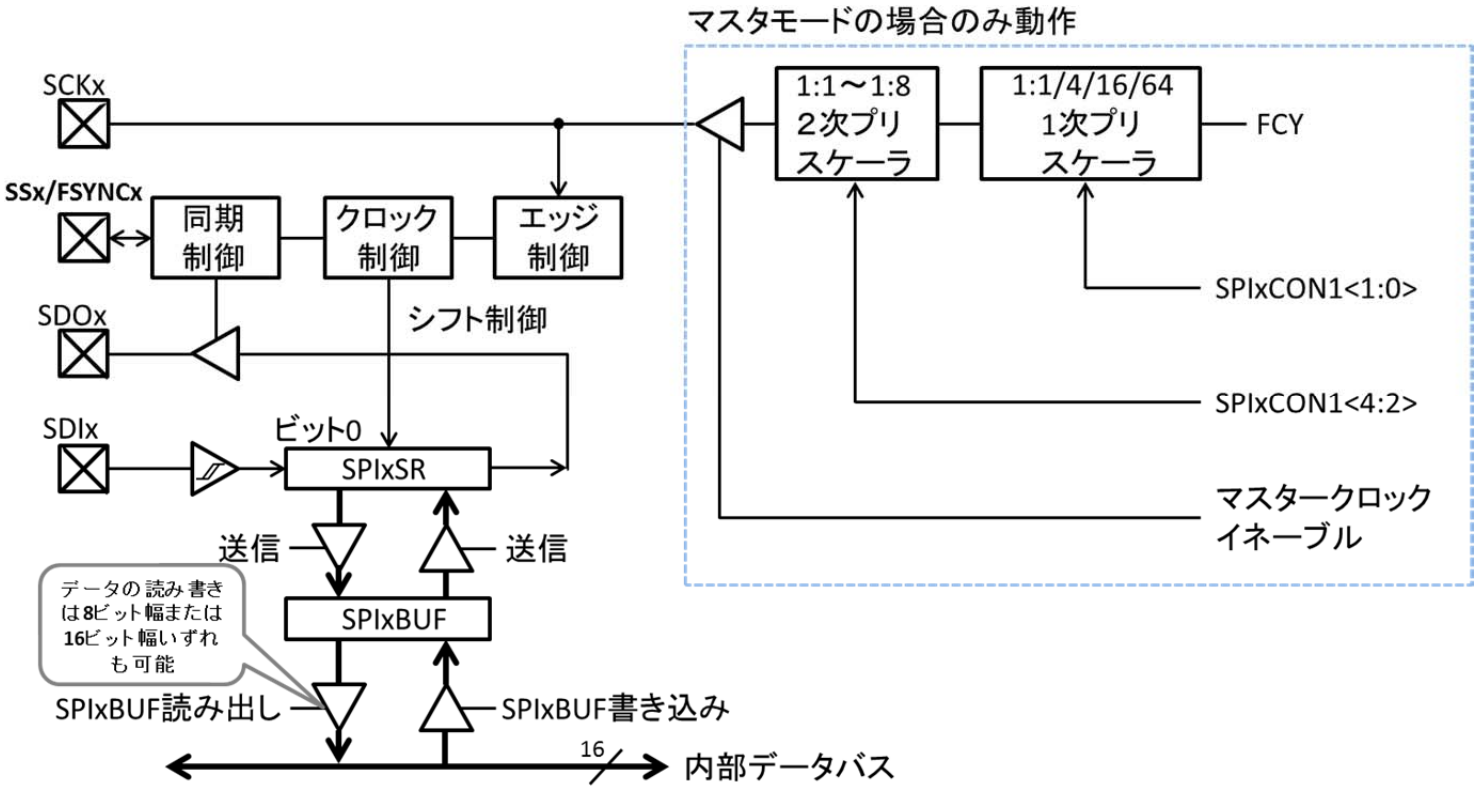

図 3.5　SPI モジュールの内部構成［15］

PIC24F ファミリの SPI モジュールの転送単位は 8 ビットと 16 ビット両方が可能となっており，選択もできるが，本研究では 8 ビットを用いる．

SPI 通信はマスタとスレーブの間で行われ，マスタは，内部システムクロックをプリスケーラで分周して SCKx ピンに SPI 用クロックとして出力し，スレーブの場合は SCKx をクロック入力ピンとして扱う．本研究では PIC がマスタ，DC コンバータがスレーブである．また，スレーブの場合には，SSx ピンをデバイス選択ピンとして使えるようになっており，必要な場合に SSx ピン機能を有効にして複数デバイスを区別することもできるが，本研究では使用していない．

マスタもスレーブも常に送信と受信が同時に行われ，SPIxSR レジスタがシフトレジスタとして動作する．このレジスタが SCKx のクロックに従って動作することで，シリアルデータが SDIx と SDOx ピンに同時に入出力される．

送受信バッファの SPIxBUF は送信と受信で兼用され，受信の場合は SPIxRXB と呼ばれ，送信の場合は SPIxTXB と呼ばれる．

送信の場合，SPIxBUF にデータを書き込むと送信中でない限りすぐに SPIxSR レジスタに転送され，クロックにより送信を開始する．この時受信も同時に行われ，送信完了と同時に受信完了となり，受信した SPIxSR レジスタの内容が即 SPIxRXB つまり SPIxBUF に転送され割り込みを発生する．このように送信データと受信データが同時に保持されることはないので SPIxBUF を兼用しても問題はない [15].

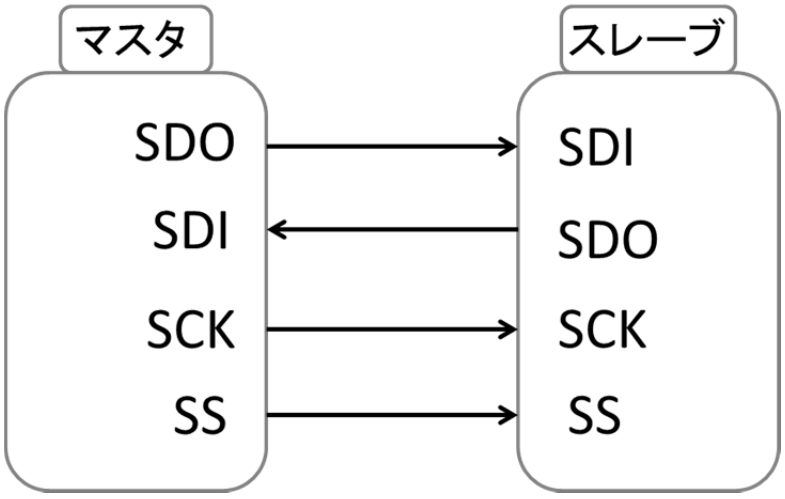

図 3.6　4 つの信号の接続関係

また，図 3.6 のように SPI 通信でマスタ側とスレーブ側で接続するとき，それぞれの信号は対応したものに接続しなければならない．

以上のことから，図 3.7 のような RT-ADKmini と MAX525ACPP を使用した AD 変換回路を作成した.

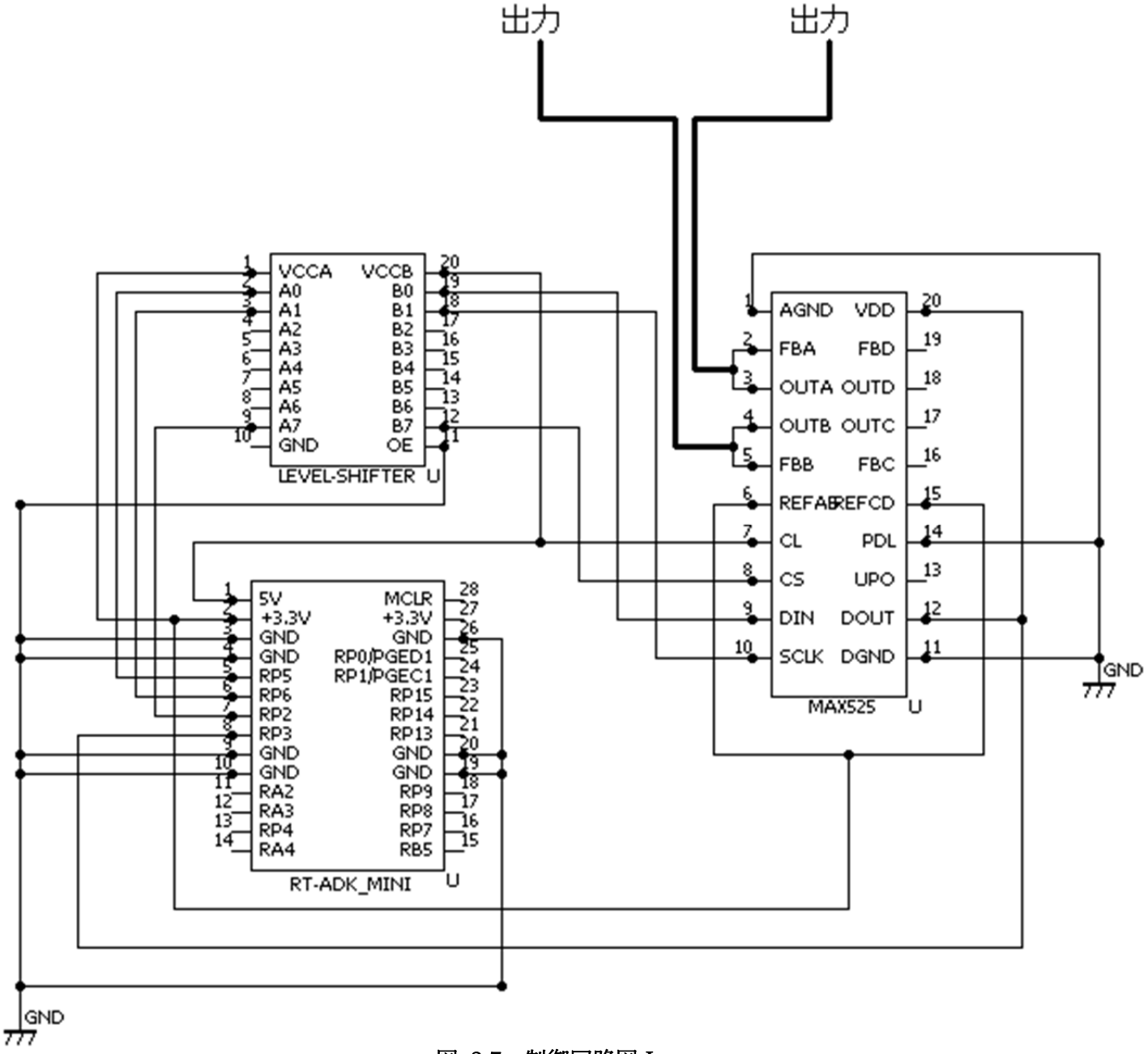

図 3.7　制御回路図 I

この回路はアナログ電圧を出力する回路である．RT-ADKmini がレベルシフト回路に接続してあるが，これは MAX525ACPP が 5V 電源で動作するため信号を増幅させる必要があるためである．なお，この回路が出力するアナログ値を変更する方法については第 4 章 Android スマートフォンによる RC モデル制御にて詳しく記述する.

図 3.8 は本研究における RT-ADKmini の SPI 通信の設定と機能の割り付けを行っている部分のソースコードである．

```
mPORTAOutputConfig(0x3);//PortA RA2 (RA1) Output///ADC分
mPORTBOutputConfig(0x1C);//PortB RB14 RB4 PB8 Output///ADC分


mPORTAWrite(0x0);//PORTA=0x0 ポートA初期化///ADC分
mPORTBWrite(0x0);//PORTB=0x0 ポートB初期化///ADC分


PPSOutput(PPS_RP5, PPS_SDO1);//RPOR SDO1割り付け RA0:RP5:AIN0
PPSOutput(PPS_RP6, PPS_SCK1OUT);//RPOR SCK1OUT割り付け RA1:RP6:AIN1
// ^SSはRB2:RP2:AIN2を手動で制御する
```

図 3.8　SPI 通信に使用する各ポートの設定

青字で表記されている 16 進数はピンが出力設定である場所が 1，そうでない場所を 0 と 2 進数で決定し，それらをまとめて 16 進数に変換したものである．図 3.7 で SPI 通信に使用するピンは RP2，RP3，RP5，RP6 であり，それらの設定を行う．例えば，赤枠内の 0x3 であれば二進数に直すと 0011 となり，PortA の RA0 及び RA1 が出力設定となる．ここで各ポートのアウトプット設定，初期化， 表 3.2 のピン割り付けの設定を行っている．この場合，RP5 が SDO，RP6 が SCK，RP3 が SS となっている．また，本研究では SDI は使用していない．

```
OpenSPI1(
    ENABLE_SCK_PIN      |
    ENABLE_SDO_PIN      |
    SPI_MODE16_ON       |
    SPI_SMP_OFF         |
    SPI_CKE_OFF         |
    SLAVE_ENABLE_OFF    |
    CLK_POL_ACTIVE_LOW  |
    MASTER_ENABLE_ON    |
    SEC_PRESCAL_1_1     |
    PRI_PRESCAL_16_1
    ,
    FRAME_ENABLE_OFF
    ,
    SPI_ENABLE          |
    SPI_IDLE_CON        |
    SPI_RX_OVFLOW_CLR
);
ConfigIntSPI1(SPI_INT_DIS & SPI_INT_PRI_6);

LATBbits.LATB2 = 1;
```

図 3.9　SPI 通信における機能の割り付け

図 3.9 は図 3.8 で割り付けた機能の内容を設定している．SPI 通信における各機能の割り付けのソースコードである．この機能内容の設定は [16]や [17]を参考にした．例えば，CLK_POL_ACTIVE_LOW でクロックの極性を選択し，MASTER_ENABLE_ON で RT-ADKnimi をマスタに設定している．以上で SPI 通信を行うためのピン設定と機能の割り付けが完了し，MAX525ACPP との SPI 通信が行われる．

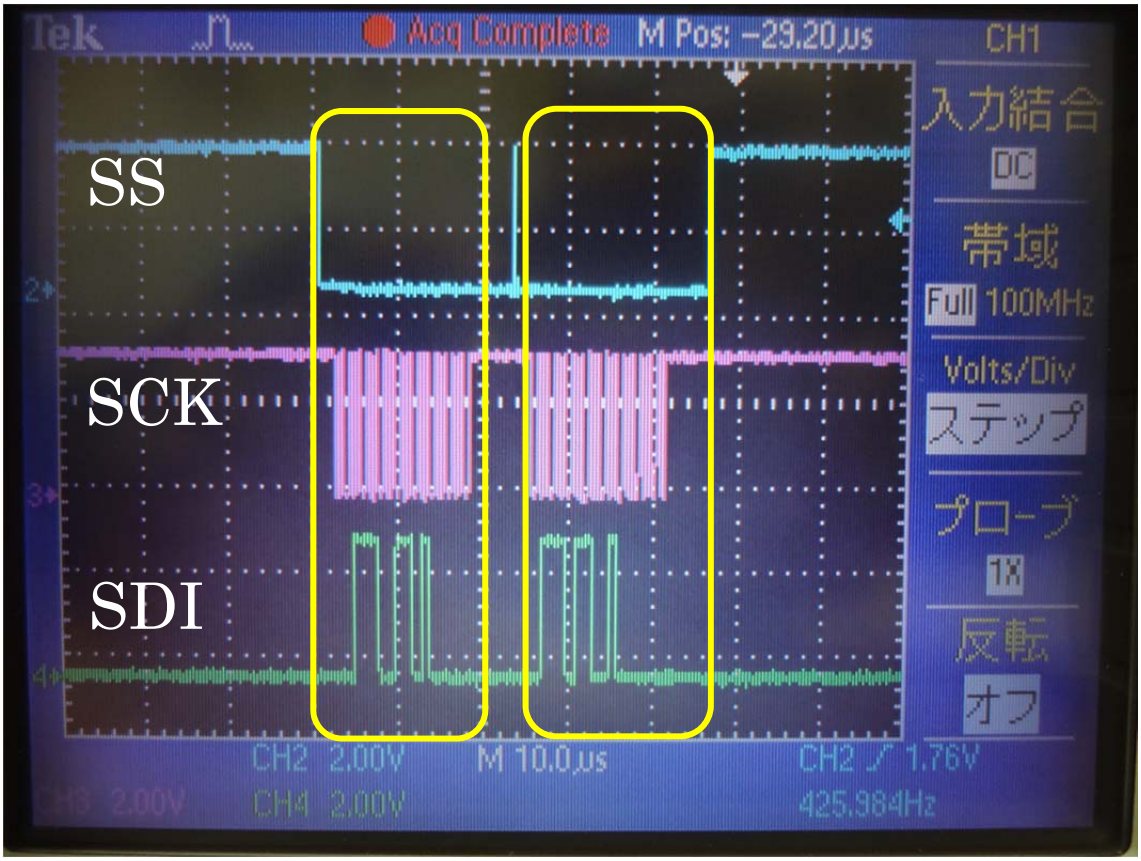

図 3.10　SPI 通信で出力されるディジタル信号

図 3.10 は出力されている SPI 通信の信号である. 青いチャネルが命令送信タイミングを同期するための信号であり，この信号が立ち下がる（LOW）瞬間から命令が 2 回送信される．

これらが正しく動作しているかを確認する動作実験を行った．作成した回路とスマートフォンを Bluetooth 接続し，スマートフォンの操作に応じて MAX525ACPP の出力ピン 2 つからそれぞれアナログ値が出力されているかをオシロスコープで確認した．

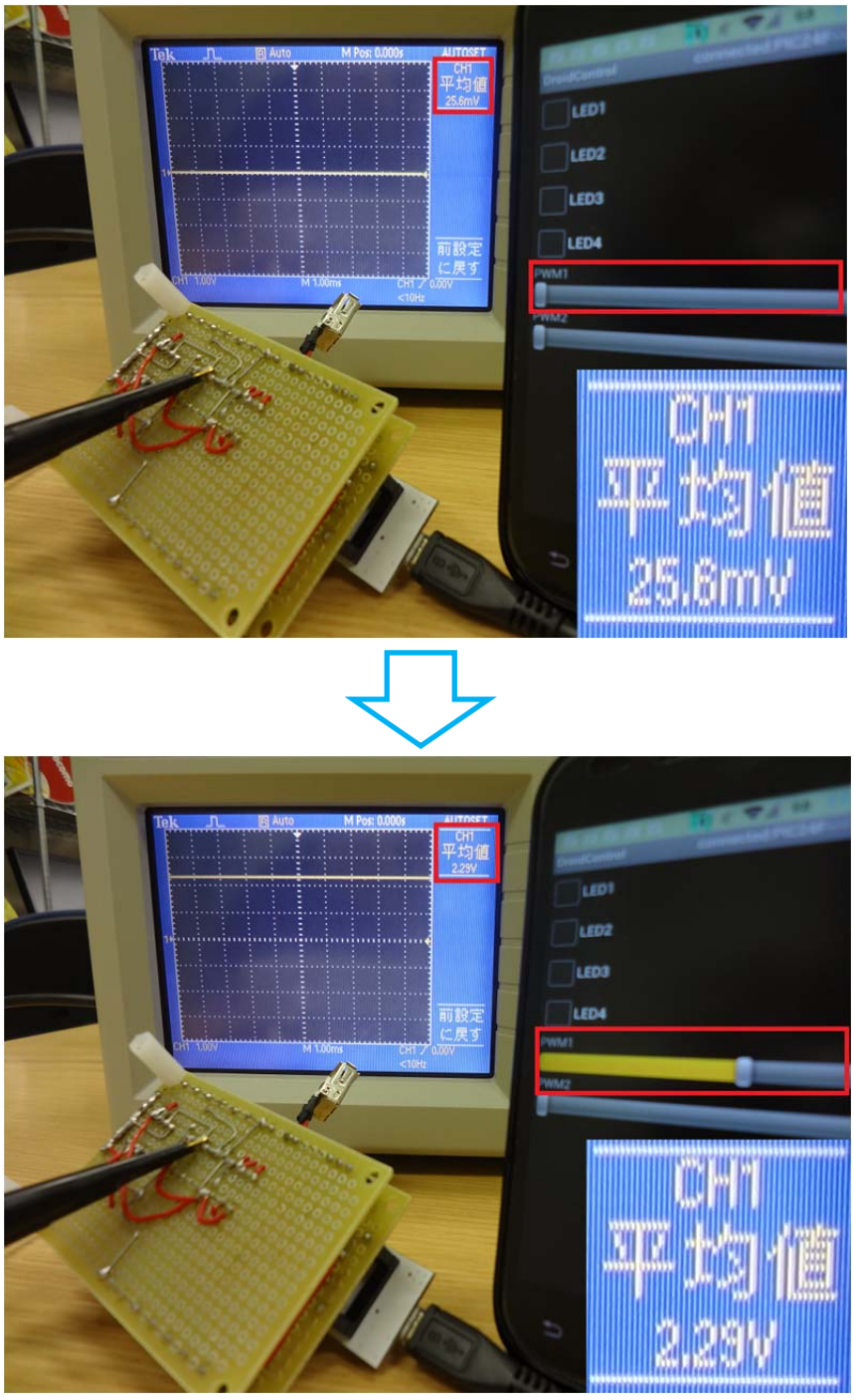

図 3.11　スマートフォン操作とアナログ出力 I

図 3.11 は実験の写真であり，スマートフォン上の上段シークバーを右に動かす．写真右下の電圧表示はその時のオシロスコープの値を拡大したものである．

すると図 3.11 下の写真のように，シークバーを動かした量に応じて電圧が 25.6mV から 2.29V まで上昇した．次にもう一方の出力ピンも確認した．下段のシークバーを右に動かしていき電圧を確認する．

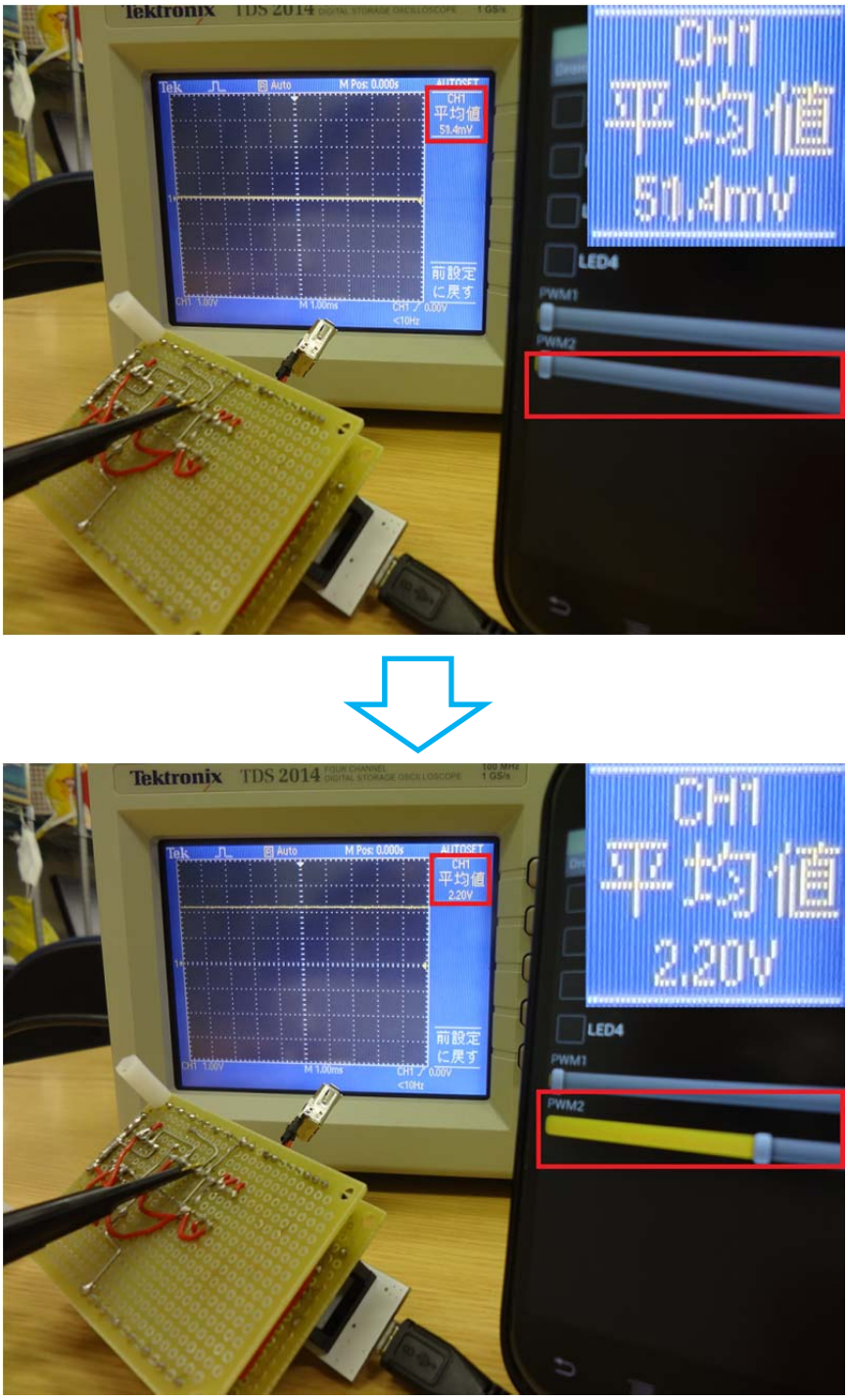

図 3.12　スマートフォン操作とアナログ出力Ⅱ

こちらも同様にシークバーの動きと連動して電圧が上昇することを確認することができた．（図 3.12）

しかし，MAX525ACPP から出力された電圧と表 2.2 で測定した電圧とは一致しない．そこでオペアンプを使用してプロポが認識する電圧値に合わせ増幅させるために非反転増幅回路を用いた．これはオペアンプを利用した増幅回路であり，回路の抵抗比（$1+R_2/R_1$）で増幅率が決まる．このことから，抵抗の 1 つを可変抵抗とすることで最も適した電圧になるよう増幅率を変更できるようにした．

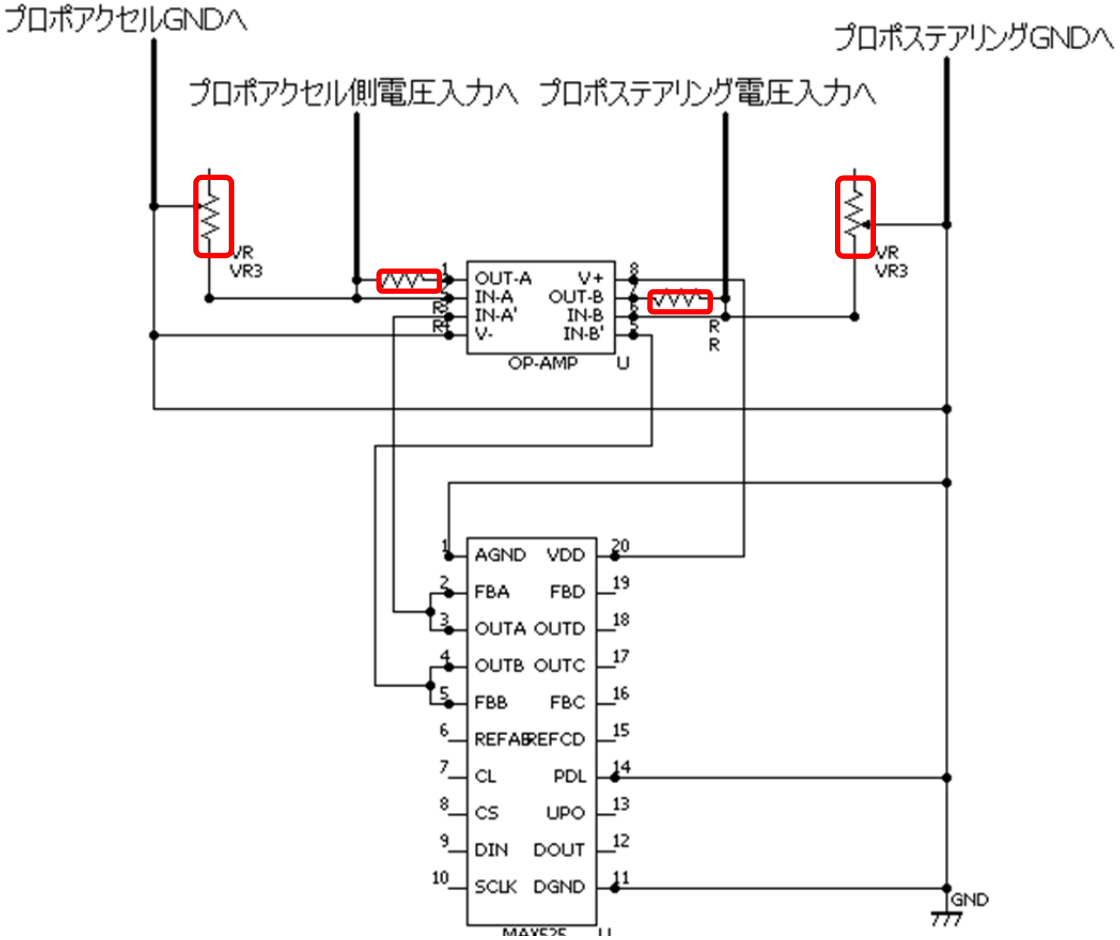

図 3.13　非反転増幅を行っている部分の回路図

図 3.13 は非反転増幅を行っている部分の回路図である．図中の赤枠で囲んだ可変抵抗の比で増幅率を決定している．本研究では固定抵抗に 560Ω，可変抵抗に 1kΩ のものを使用し増幅率は最大 1.56 倍になる．

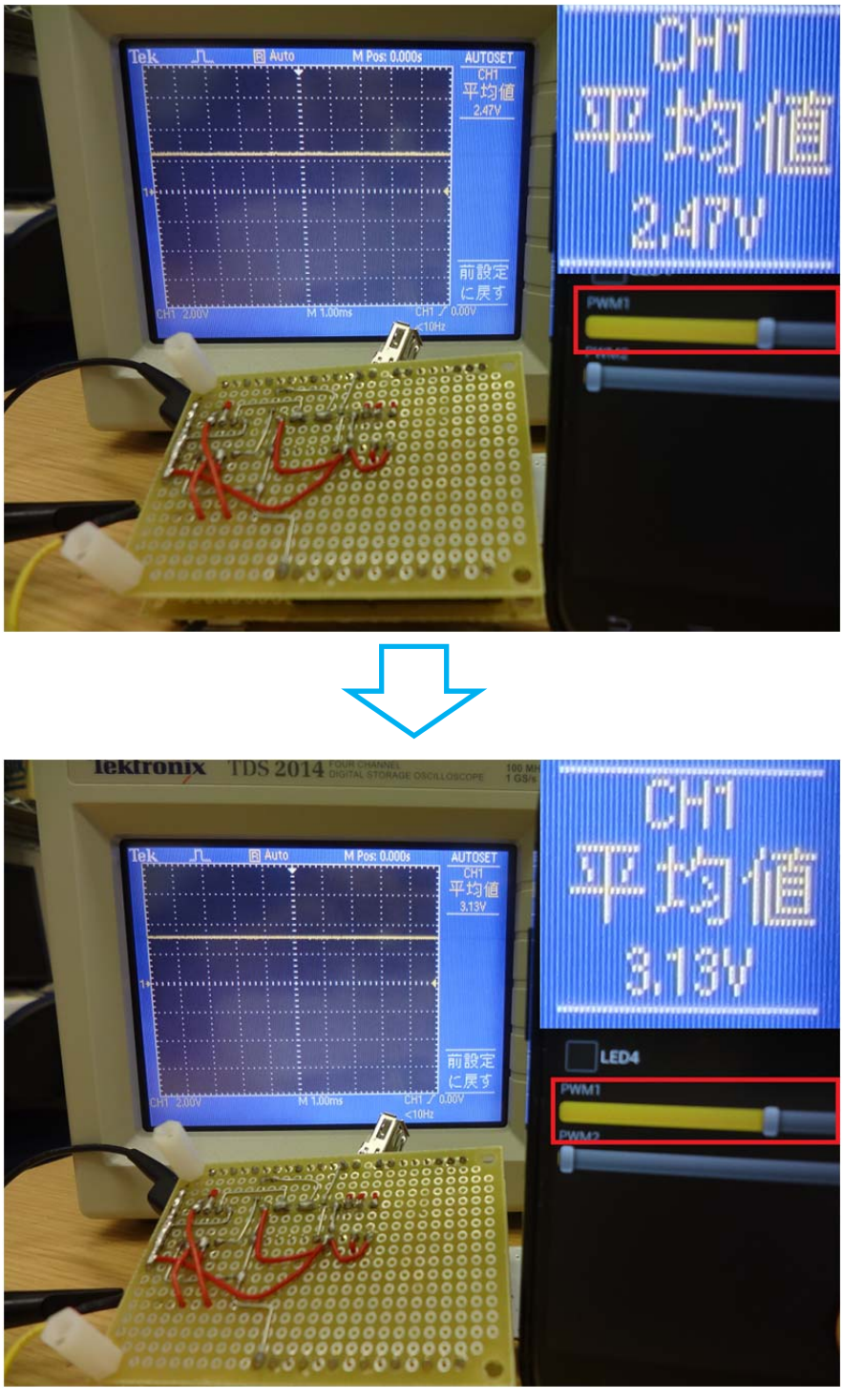

図 3.14　増幅後のアナログ出力 I

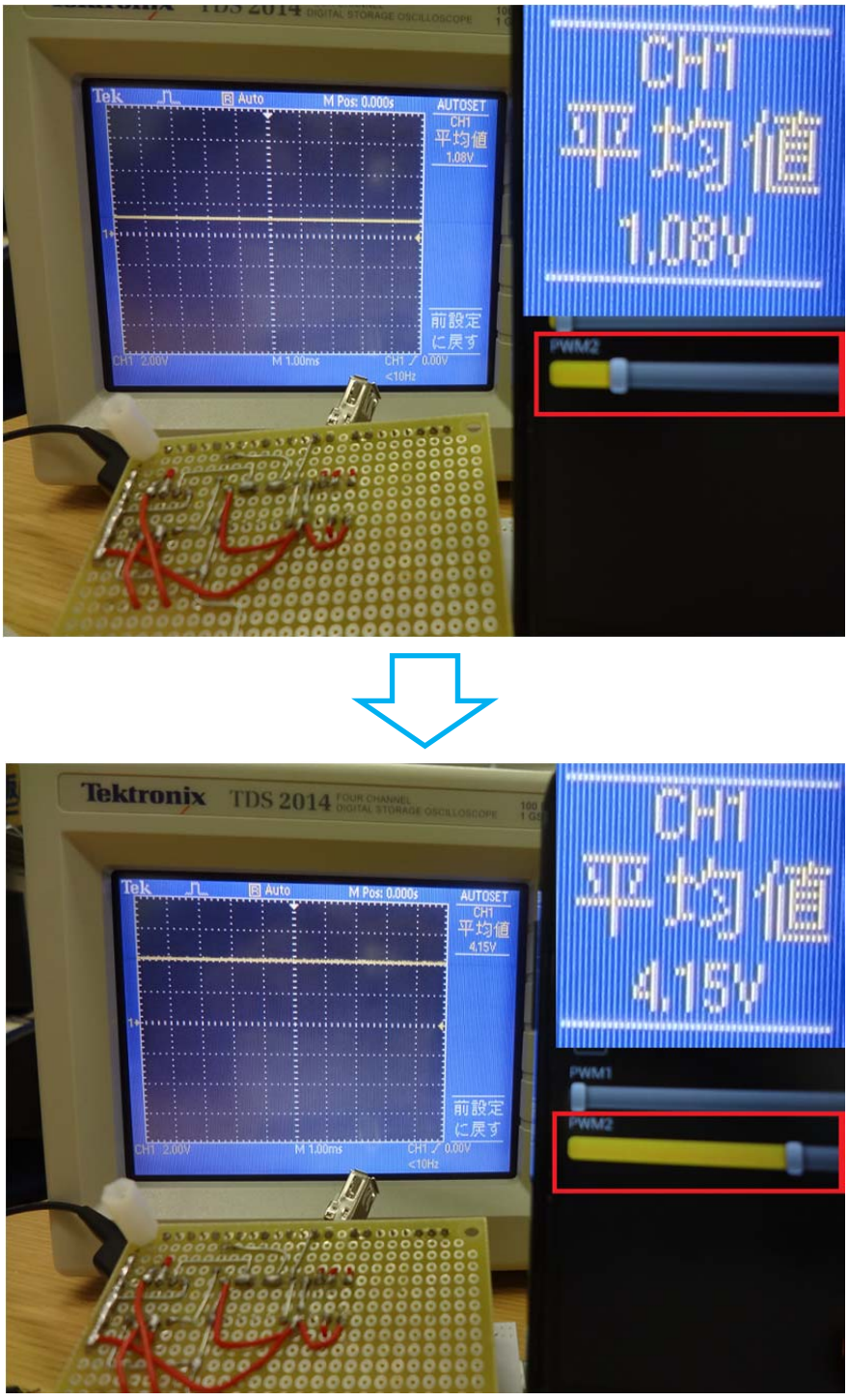

図 3.15　増幅後のアナログ出力Ⅱ

図 3.14，図 3.15 は増幅後のアナログ出力である．このように表 2.2 に示した電圧幅に対応することができた．

第4章　Android スマートフォンによる RC モデル制御

4.1　Android とは

Android とは Google 社が発表したスマートフォンやタブレット端末用のオペレーティングシステムである．世界の携帯電話端末メーカー数十社が共同で Open Handest Alliance(OHA)という業界団体を設立し，関連技術の開発や普及を推進している．現在は多くのスマートフォン端末が AndroidOS 搭載のものであり増え続けている．Android の特徴として OS がオープンソースであることあげられる．また，誰でも無償でアプリケーションソフトを開発でき，Android 端末で動作させることができる．

図 4.1　AndroidOS 搭載のスマートフォン

4.1.1　開発環境

Android のアプリケーション開発には Java 統合開発環境としてサポートされている Eclipse を使用する．

まず，Java 言語で書かれたプログラムソースをコンパイル，実行するために必要な JDK（Java Developer Toolkit）をインストールする必要がある．

次に AndroidSDK をインストールする．これは Android SDK Tool から Android の各バージョンパッケージをインストールし Eclipse でそのバージョンに対応した開発環境を整えることができる．

この Eclipse 上でアプリケーションのプログラミングを行い，スマートフォンにダウンロードする．

4.2　　コントローラアプリケーション

本節ではスマートフォンから回路への制御系統について述べる．PIC から出力されている信号をスマートフォンに内蔵されている傾きセンサと連動させ，前に倒すと前進，後ろは後退，左右に傾ければ左右に曲がる直感型のコントローラとする．スマートフォンには傾きセンサや加速度センサ，方位センサ等様々なセンサが搭載されている．これらの中から傾きセンサを利用し，スマートフォンの傾きで RC モデルを制御する．

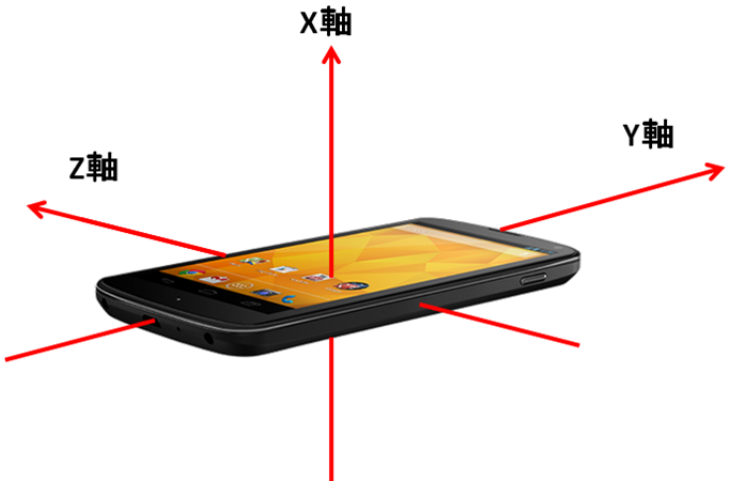

図 4.2　スマートフォンの傾きセンサがとる軸

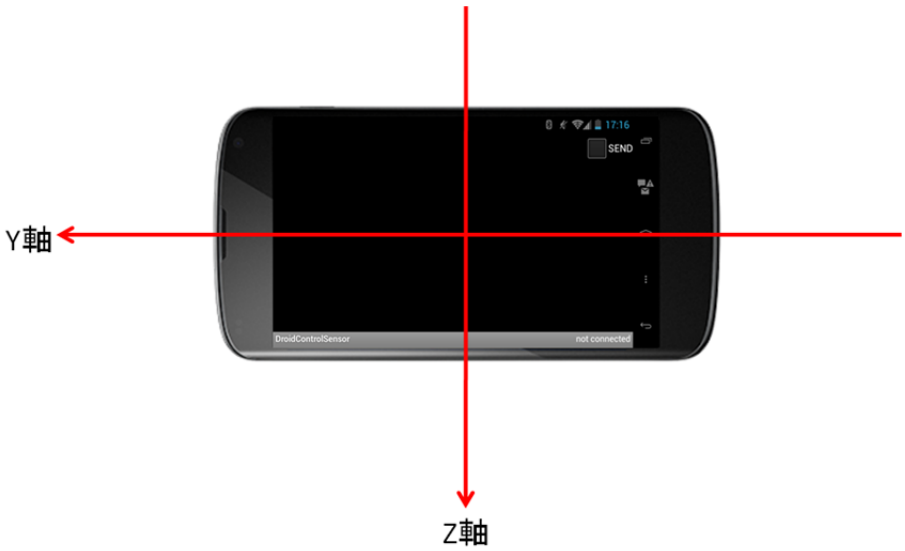

図 4.3　コントローラアプリケーションにおける軸

図 4.2 はスマートフォンの傾きセンサの軸を表している．X 軸は方位角，Y 軸は傾斜角，Z 軸は回転角をそれぞれ表している．アクセルとステアリングの 2 つを制御する本研究では X 軸に相当する方位は不要であるため，Y 軸と Z 軸 2 軸を使用する．図 4.3 は実際にコントローラアプリケーションをとして使用する場合，使用者から見たスマートフォンの向きとその軸の取り方である．

傾きセンサを用いたコントローラを作成する準備として，シークバーで動作するアプリケーションを作成した．シークバーを上下に 2 本表示させ，上段のシークバーを操作すると前進，停止，後退を行い，下段のシークバーを操作するとステアリングが左右に切れるものである．

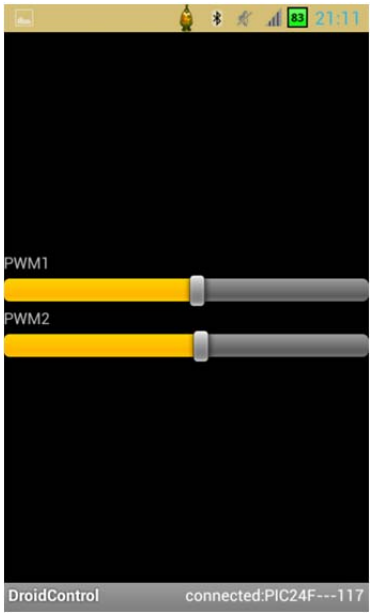

図 4.4　シークバーコントローラ

このとき，シークバーは 0～255 の整数値を取っており，プロポのトリム部の電圧値をこれに対応させることで操作した．表 4.1 に対応表を示す．

表 4.1　0～255 の値の対応表

アクセル		ステアリング	
前進	91	左	41
停止	153	ニュートラル	100
後退	186	右	161

Java のプログラミングコードで Log を出力すると Eclipse の LogCat ウィンドウで確認することができる．

```
170             // トラッキング開始時に呼び出されます
171             @Override
172             public void onStartTrackingTouch(SeekBar seekBar) {
173                 Log.v("SeekBar", "Start : " + String.valueOf(seekBar.getProgress()));
174             }
175             // トラッキング中に呼び出されます
176             @Override
177             public void onProgressChanged(SeekBar seekBar, int progress, boolean fromTouch) {
178                 Log.v("SeekBar", "Changed : " + String.valueOf(seekBar.getProgress()));
179
180                 //シークバーから値を取り出し
181                 ThresholdSeek = seekBar.getProgress();
182
183             }
184             // トラッキング終了時に呼び出されます
185             @Override
186             public void onStopTrackingTouch(SeekBar seekBar) {
187                 Log.v("SeekBar", "Stop : " + String.valueOf(seekBar.getProgress()));
188             }
189         });
```

図 4.5　シークバーと Log のソースコード

図 4.5 はシークバーのソースコードの一部である．赤枠で囲われた部分に書かれているのが Log のプログラミングコードであり，それぞれシークバーをタップした時，シークバーを動かしている時，シークバーを止めた時の 3 つのタイミングで Log を出力している．この Log を Eclipse 上の LogCat で表示したのが図 4.6 となる．

```
SeekBar          Start : 151
SeekBar          Changed : 97
SeekBar          Changed : 87
SeekBar          Changed : 85
SeekBar          Changed : 82
SeekBar          Changed : 79
```

図 4.6　LogCat での取得した Log の確認

4.2.1　　傾きセンサによるコントローラ

これらの値からスマートフォンの傾きセンサを使用したコントローラに作成する．図 4.7，図 4.8 は傾きセンサを使用したコントローラの使い方のイメージである．

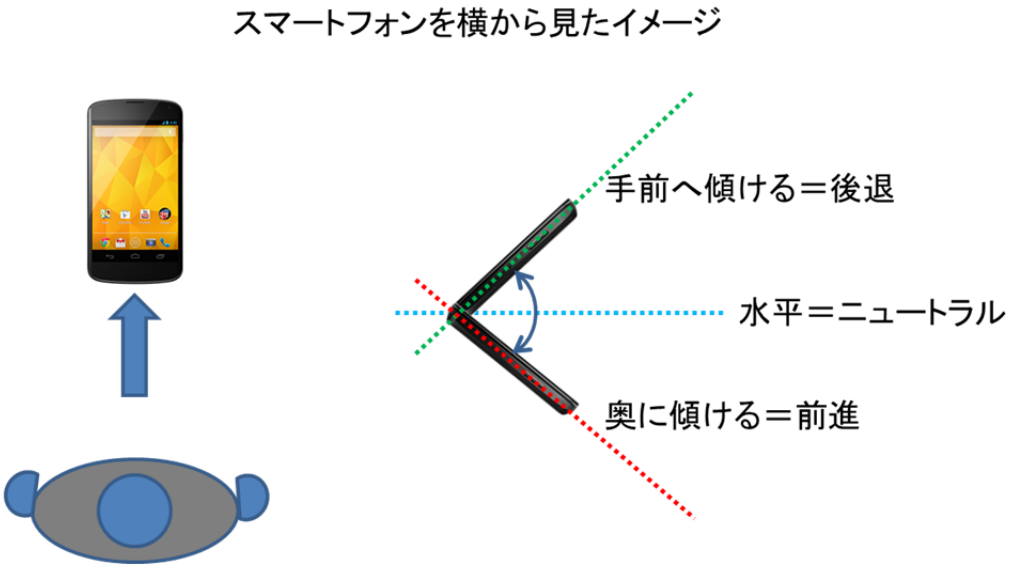

図 4.7　傾きセンサによるアクセル制御イメージⅠ

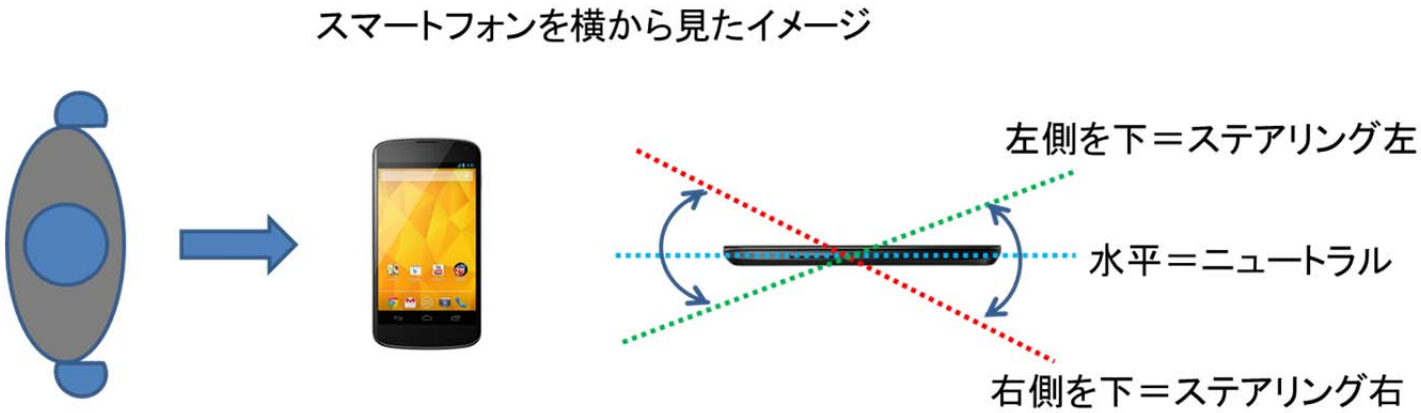

図 4.8　傾きセンサによるステアリング制御イメージⅡ

このように傾けることで前進，後退，左，右の制御を行う．この動作を傾きセンサの値を基に適切な PWM の幅を計算することで傾きセンサによるコントローラアプリケーションを作成する．

```
651         if(DEBUG) Log.d(TAG,""+event.values[2]);
652         float angle = event.values[2];
653         float slope = event.values[1];

664         pwm_width1 = (int)(95*angle/165+154);      //アクセル0度が0点(停止)のときの式
665         pwm_width2 = (int)(2*slope/3+98);          //ステアリング現行セッティング(可変抵抗)の式
```

図 4.9　傾き角度を取得後式変換

angle が Z 軸でアクセル制御, slope が Y 軸でステアリング制御に使用している. event values でそれぞれの傾きセンサの値を取得している．傾きセンサの値と表 4.1 の値より一次関数を作成する（図 4.9）．この場合 pwm_width1 が angle を含む式なのでアクセル，pwm_width2 が slope を含む式なのでステアリングをそれぞれ表している．

```
809         sendMessage(String.format("s%02x", pwm_width1));
824         sendMessage(String.format("t%02x", pwm_width2));
```

図 4.10　コントローラアプリケーションの傾き情報送信

これら傾きセンサから得られた情報を回路に Bluetooth を使用して送信するソースコードが図 4.10 である．

以下では作成したアプリケーションの利用法について述べる．図 4.11 は，スマートフォンのスクリーンショットである．

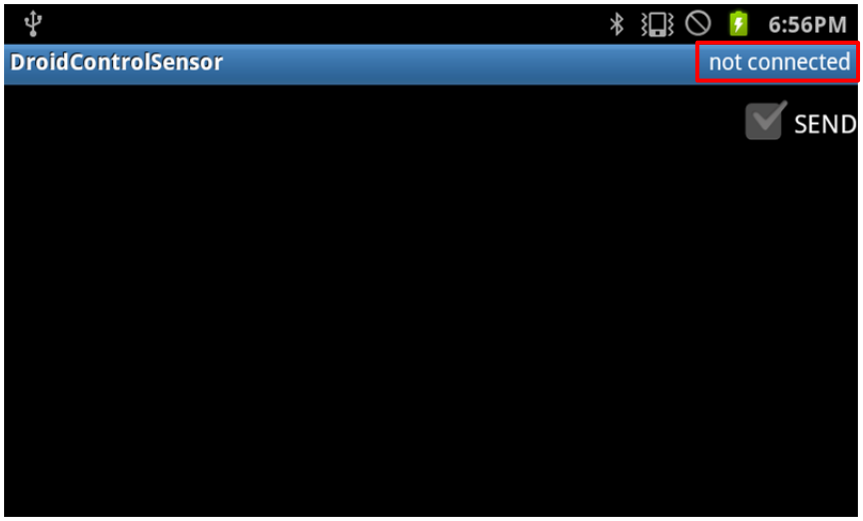

図 4.11　コントローラアプリケーションのキャプチャ画像

このとき，上部赤枠内は not connected と表示されており，回路側と Bluetooth 接続が完了していないことを示している．以下はスマートフォンと回路を Bluetooth 接続する為の手順である．

図 4.12　アプリケーションメニュー画面

メニューから図 4.12 にある Connect device をタップするとスマートフォンとペアリング設定が完了している Bluetooth の番号が出てくる[5]．

[5] あらかじめコントローラアプリケーションをダウンロードした端末の Bluetooth 接続を ON にしておき，Bluetooth ドングル（回路側）とペアリング設定を行う必要がある．

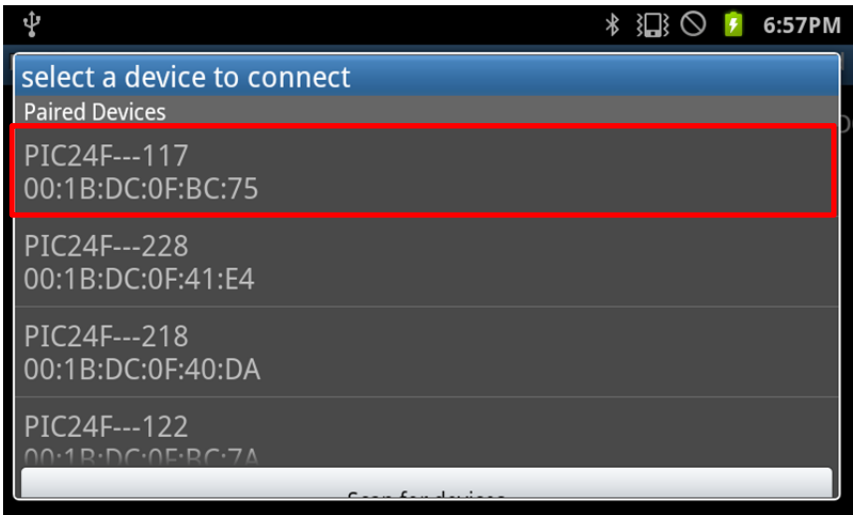

図 4.13　ペアリング済みの Bluetooth リスト

図 4.13 中から現在回路に使用している Bluetooth 番号の書かれているものをタップすると接続を行う．図 4.13 では PIC24F---117 を選択している．

図 4.14　回路との接続状態

完了すると図 4.14 のように not connected と書かれていた場所に現在接続している Bluetooth ドングルの番号が表示される．

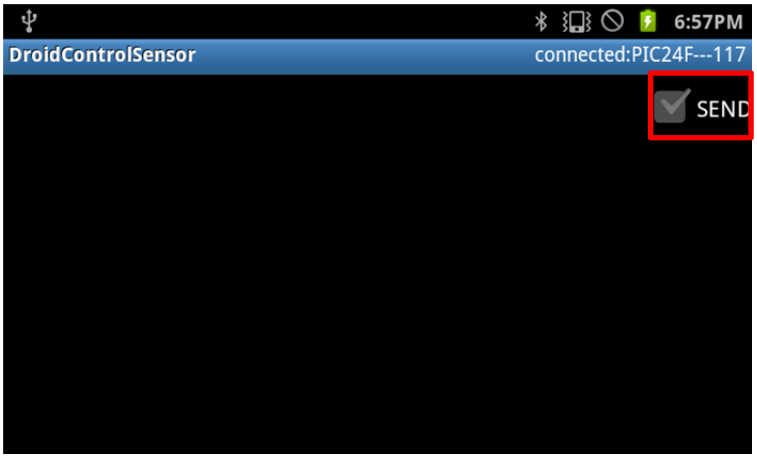

図 4.15　Bluetooth 接続の完了したアプリケーション画面

その後，図 4.15 赤枠内の SEND チェックボックスをタップすることで傾きセンサの値を回路側に送信し始める．

4.3　バック走行用コントローラ

傾きセンサコントローラは, スマートフォンを水平に置いた状態で RC モデルが静止し，これをニュートラル状態とする．そして水平状態から奥，つまり Z 軸のマイナス側に倒していくと加速，Z 軸プラス側向に倒していくと後退する．また Y 軸マイナス側に傾けていくと右にステアリングが切られ, Y 軸プラス側に傾けていくと左にステアリングが切られる．

しかし，このままでは RC モデルを後退させることはできない．電動 RC モデルは前進時に突然後退しないようになっているためである．前進中，または前進命令を 1 度でも受信した後であれば，1 回目のバック命令はブレーキ命令となる．そして，その後ニュートラルに戻し[6]，もう一度バック命令を送信することで初めて後退する仕様になっている（図 4.16）．

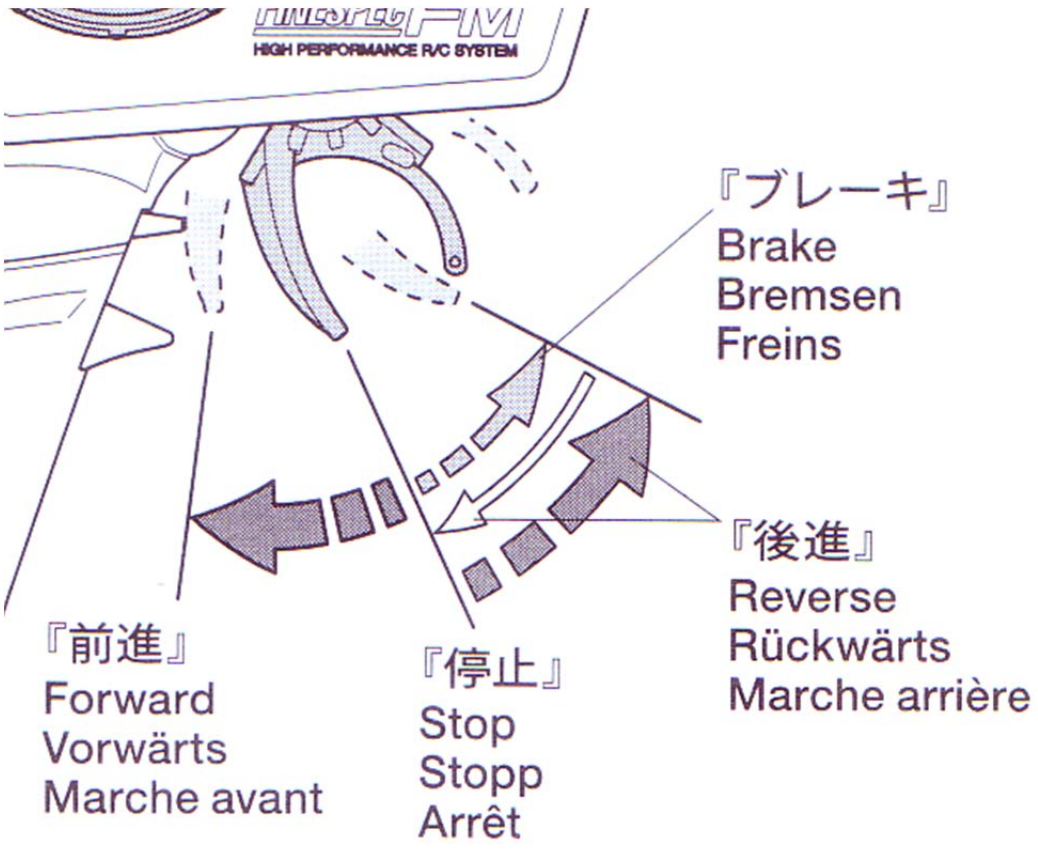

図 4.16　スロットルトリムでの後退操作 [18]

[6] ニュートラルの信号を一度送信できればいいので RC モデルを停止させる必要はない．

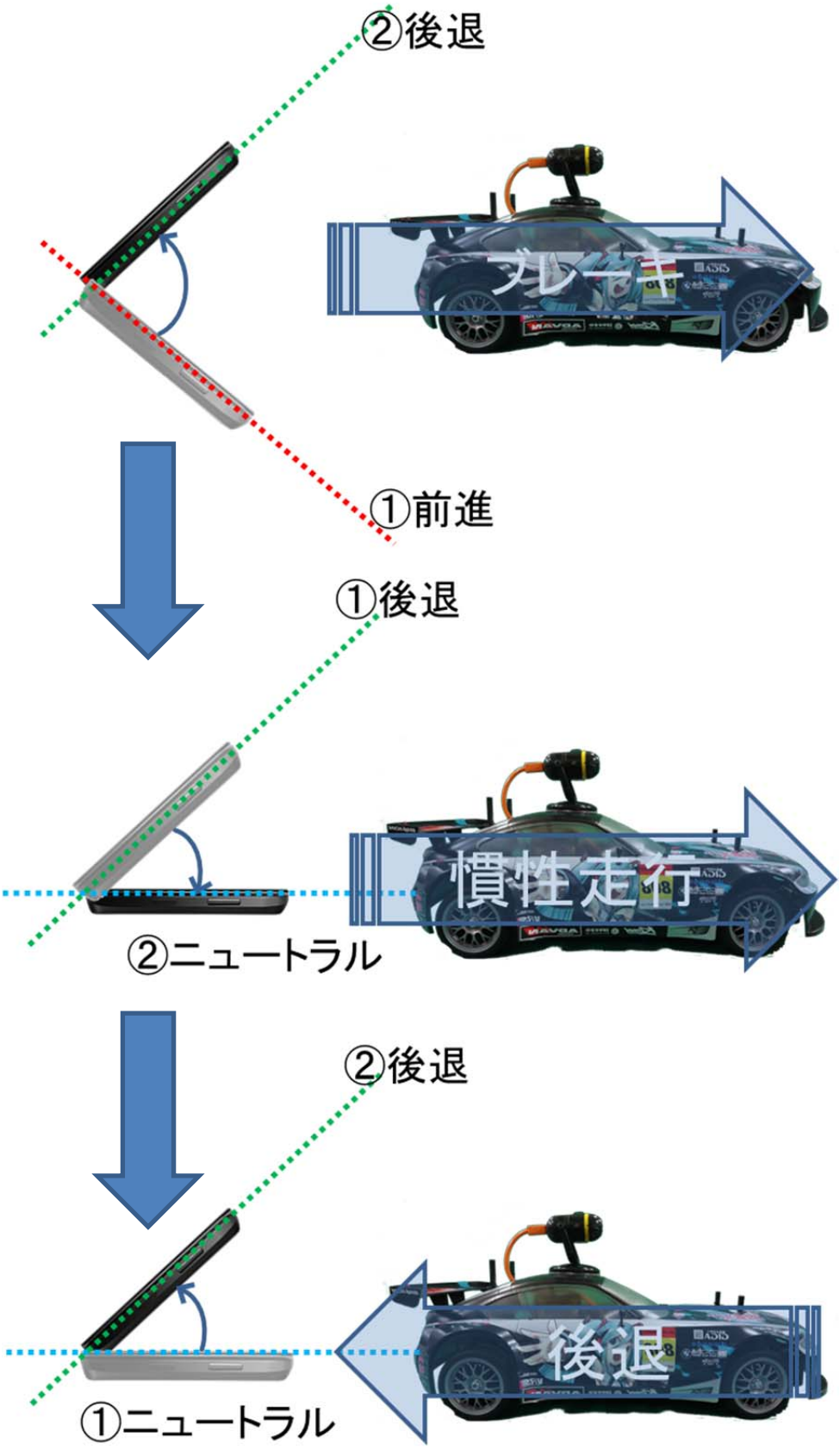

図 4.17　コントローラアプリケーションによる後退時の操作流れ I

この動きをコントローラアプリケーションでこの操作をする場合，図 4.17 のような一連のコントローラアプリケーションの動作が必要である．前進走行中に後退方向に傾けた後，ニュートラル位置（スマートフォンが水平）に戻し，もう一度後退方向に傾けることで初めて後退を開始する．

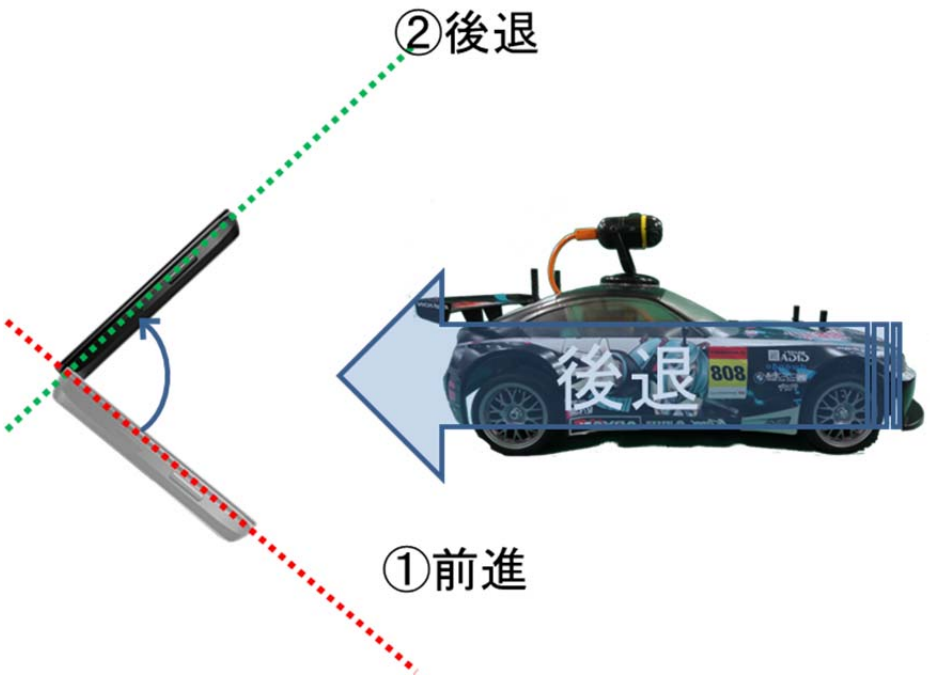

図 4.18　コントローラアプリ―ションによる後退時の操作流れⅡ

この操作は本来の用途であるレースを行うのであればこの機能は必須だが本研究では高速走行することはなくその限りではない．　そのため図 4.18 のように，一度後退方向に傾けるだけで後退を開始するようにコントローラアプリケーションを変更する．

図 4.19 が変更下プログラムである．スマートフォンを後ろに傾けると，スリープ→ニュートラル値を送信→スリープ後退，という図 4.17 の流れの通りの処理が行われる．

図 4.19　バック走行用ソース，スリープ処理

また，ファームウェア側で RC モデルのスイッチが入った時のノーコントロール状態を防ぐ為に初期値を設定する必要がある．コントロール部の制御はすべてスマートフォンから送信されるが，RC モデルを起動したとき静止している為には、ステアリング，アクセルともにある一定の数値を送信し続けるよう図 4.20 を RT-ADKmini の Main 関数冒頭に記述することでニュートラル値を出力させる．

```
// 初期値はここで定める
unsigned bar_val1=0x9A0;
unsigned bar_val2=0x640;
```

図 4.20　初期値の設定

第5章　RC モデルの自律停止機能の追加

5.1　　赤外線距離センサの利用

近年自動車にセンサ類が搭載されていることは珍しくなくなってきている．本研究では RC モデル前方に 2 つの赤外線距離センサを搭載し，一定距離以内に物体が接近した場合に制動をかけ停止させる．これは運転手の前方不注意から発生する接触，あるいは人身事故を回避する目的がある．図 5.1 は赤外線距離センサの搭載位置である．

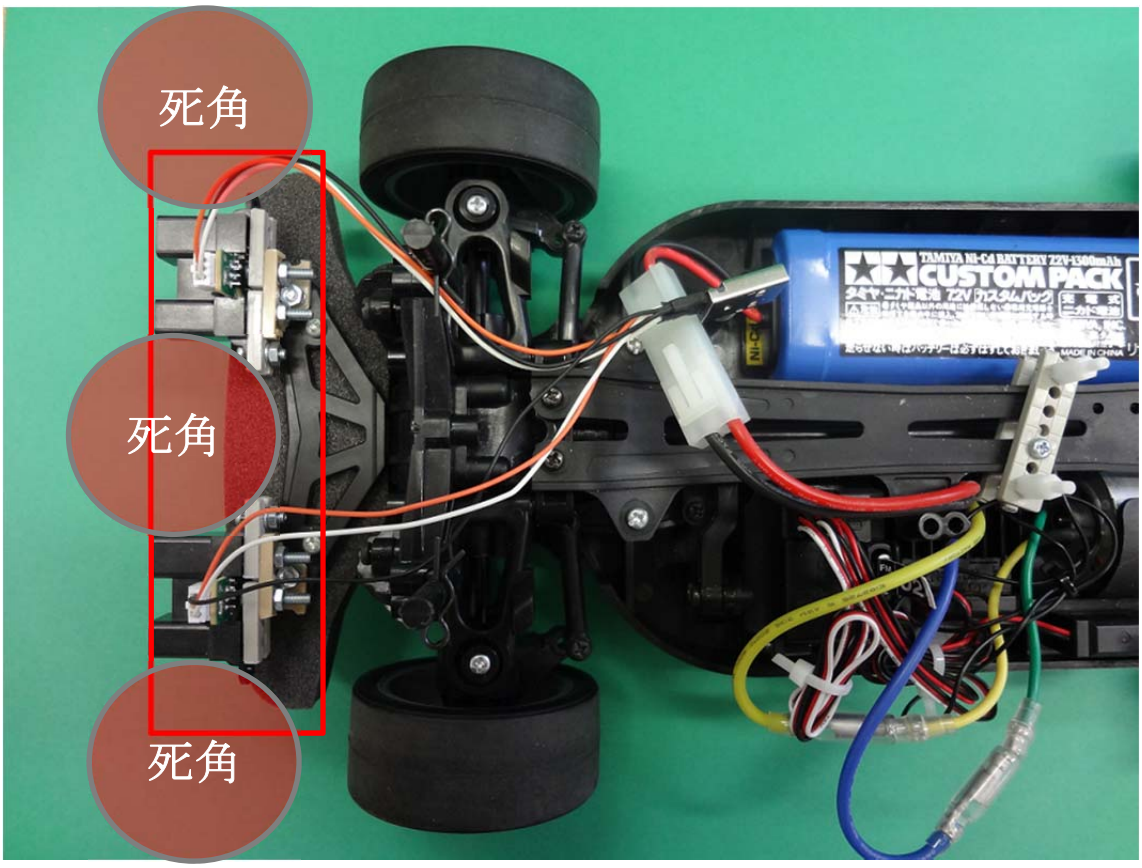

図 5.1　センサ取り付け位置

この赤外線距離センサの検出範囲は正面 80cm である．そのため赤外線距離センサを 2 つ搭載することで検出範囲を増やした．しかし 2 つのセンサの設置間隔距離以下の物体は検知できない為図 5.1 にある部分は死角になってしまうが，本研究では赤外線距離センサは補助的な意味合いが強いため死角を完全に無くすことは行わなかった．

使用した赤外線距離センサは SHARP 社製の GP2Y0A02YK というアナログ電圧出力タイプのセンサである．図 5.2 は使用した赤外線距離センサのデータシートの距離と電圧の関係を表したグラフの抜粋である．

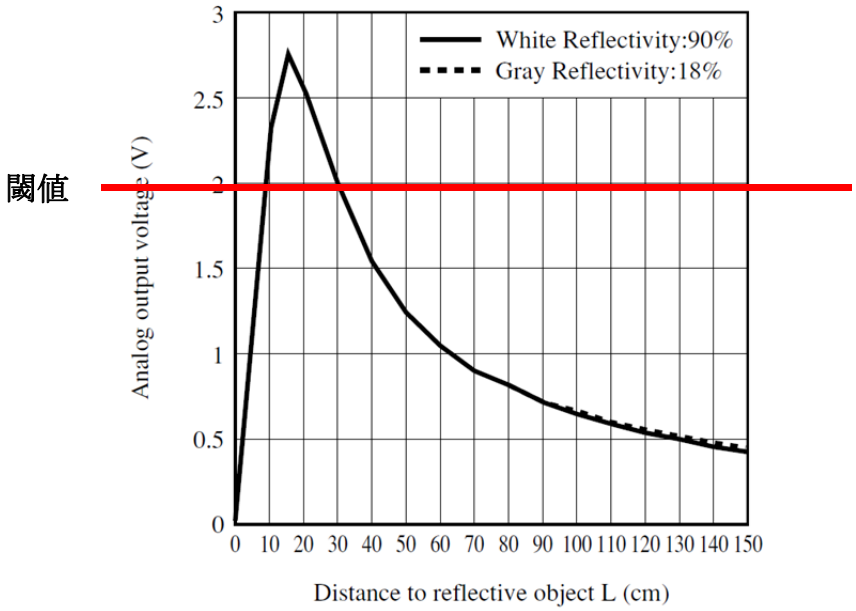

図 5.2　出力電圧と距離の関係［19］

このように距離に応じて電圧が出力される為，ある閾値以上の電圧が得られたときに PIC でアクセルを制御することで RC モデルを停止させることができる．例えば 30cm で制動をかけたい場合，図 5.2 のように 2V に閾値を設けると閾値以上の電圧，前方 30cm~10cm の距離間で制動をかける．

5.1.1　AD 変換

距離センサから出力された電圧（アナログ値）を読み取るため，PIC の AD 変換機能を使用する．AD コンバータの構成は図 5.3 のようになっている．サンプルホールドアンプと AD コンバータ本体はそれぞれ 1 個だけなので常に 1 入力ごとの AD 変換となる．この AD コンバータは，逐次変換型であり最大変換速度は 500ksps，分解能は 10 ビットとなっている．また最大 16 チャネルの入力ピンがあり，上下両端のリファレンス電源として外部入力を用いることが可能である[7] [15].

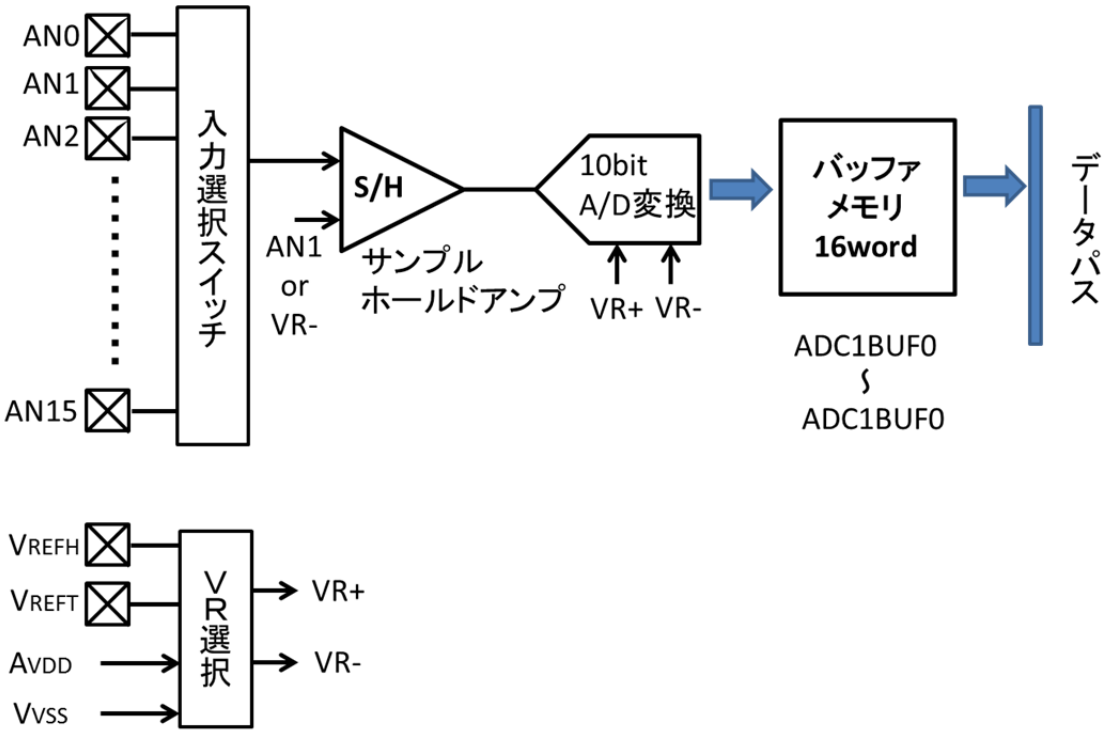

図 5.3　AD コンバータの構成

[7] ただし PIC のピン数によって実装チャネル数は異なる．

5.1.2　制御系統

赤外線距離センサの情報を AD 変換し RC モデルのアクセル制御を行う. まずピンの入出力設定と AD 変換を行う為の設定を行う.

```
//A/D変換
CLKDIV = 0;           //1:1クロック分周

///Timer3の設定 内部クロック 1/1 FOSC=16MHz
PR3 = 0xc35;                  //50msecインターバル
T3CON = 0x8000;

//ADCの設定
AD1CON1 = 0x8044;             //ADオン
AD1CON2 = 0x0404;
AD1CON3 = 0x1F05;
AD1CHS = 0x0000;
AD1PCFG = 0xFFF3;             //RA2,3をアナログに
AD1CSSL = 0x000C;             //An0,1自動スキャン

IEC0bits.AD1IE = 1;           //タイマーの割り込み許可
```

図 5.4　AD コンバータの機能設定

RC モデルのアクセル制御だが, 今回はスマートフォン側ではなくファームウェア側で行う. これはセンサが読み取った情報は緊急性が高いものでありスマートフォンの命令に割り込んで制御を行う必要があるためである.

```
if(itp_data1 >= 650 && bar_val1 <= 2463){
    bar_val1 = 2464;
}else if(itp_data2 >= 650 && bar_val1 <= 2463){
    bar_val1 = 2464;
}else if(itp_data1 >= 650 && itp_data2 >= 650 && bar_val1 <= 2463){
    bar_val1 = 2464;
}
```

図 5.5　センサによる RC モデル制動プログラム

図 5.5 はある一定距離以下になった時に RC モデルを停止させるプログラムである. itp_data1, 2 が左右それぞれの赤外線距離センサのデータ名である. この値を変更することで反応する距離を変更できる. また, bar_val1 は PIC におけるアクセル制御に用いる変数であり, 今回はこれに直接値を格納することで停止制御を行う.

5.1.3　制御回路

図 5.6 は赤外線距離センサ部の回路図である．RP0，RP1 を AD 変換ポートとして赤外線距離センサからの情報を処理する．

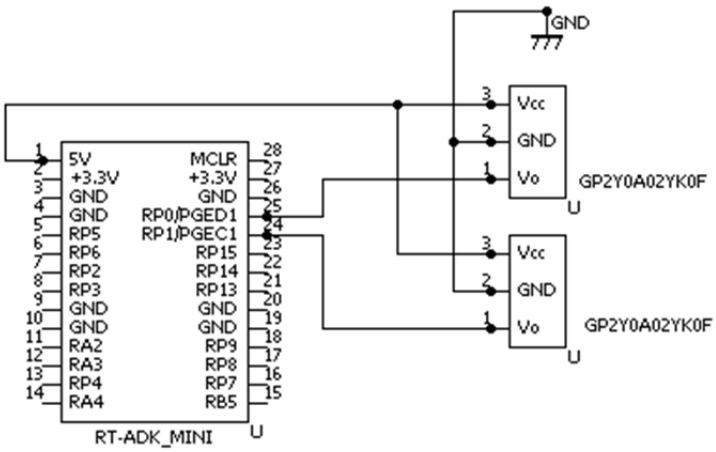

図 5.6　赤外線距離センサ部の回路図

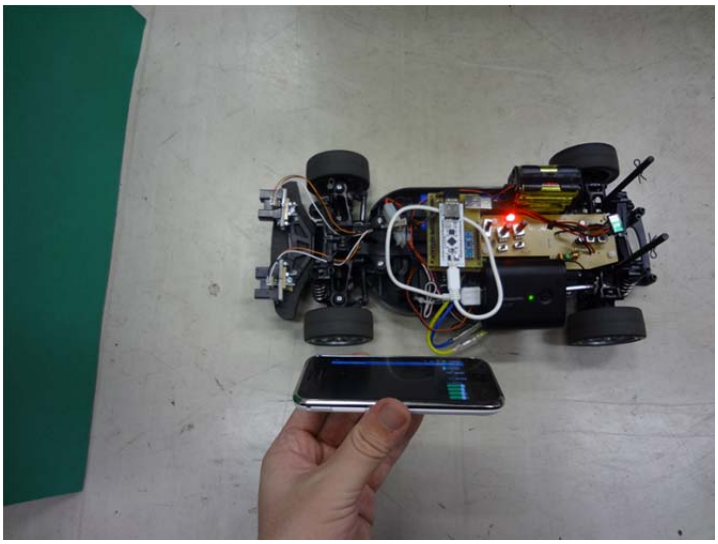

図 5.7　赤外線距離センサの動作実験

図 5.7 は赤外線距離センサの動作確認実験の写真である．スマートフォンをコントローラ状態にし，前進方向に傾け RC モデルを前進させながら壁に近づけていき，前方の壁を検知すると RC モデルは停止した．

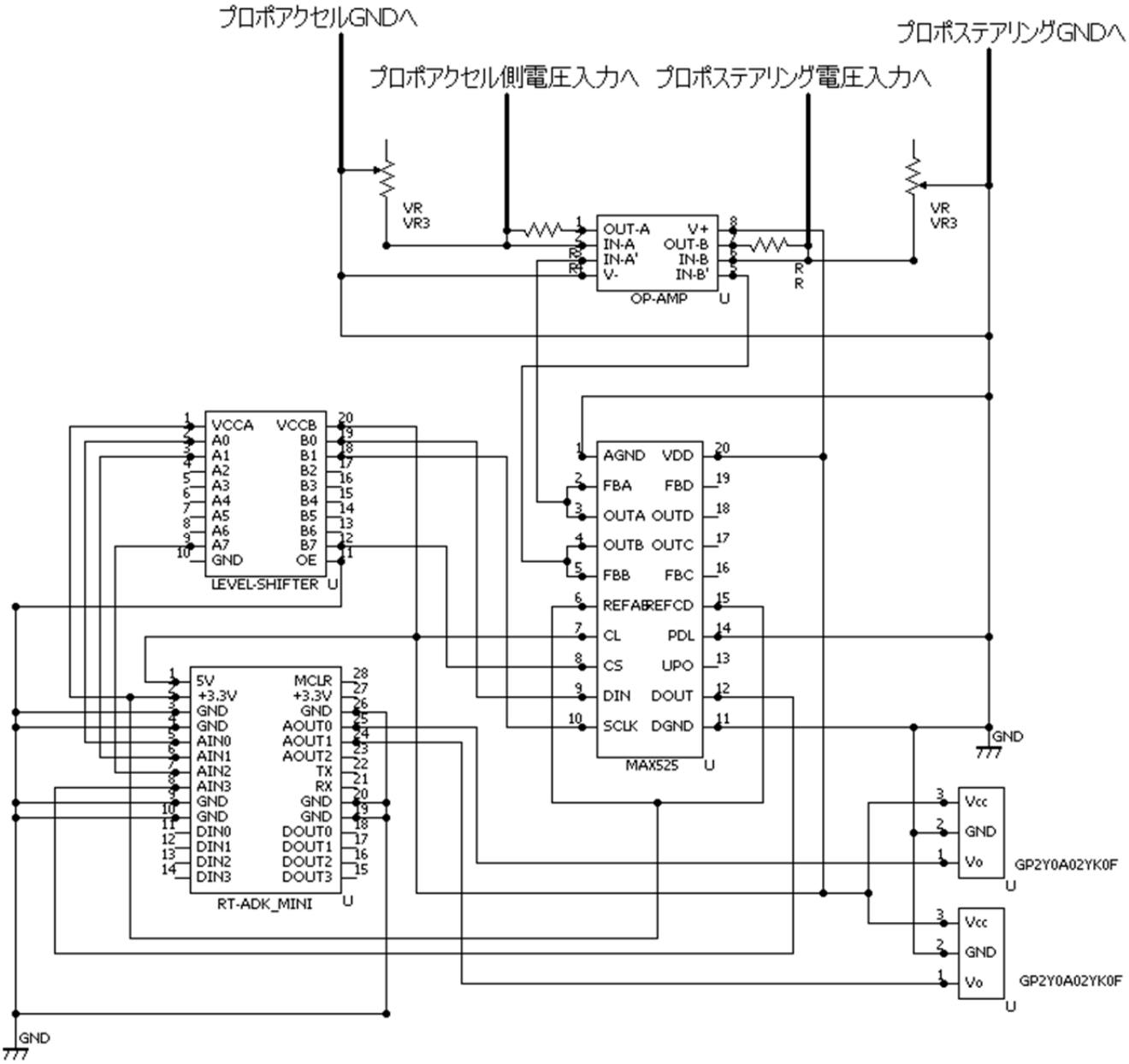

図 5.8　制御回路Ⅱ

図 5.8 は図 3.7，図 3.13，図 5.6 を統合した最終的な制御回路である．これでスマートフォンの傾きセンサによるコントロール機能，赤外線距離センサによる衝突回避が可能になり，基本的な制御形態が完成した．

第6章　画像処理による白線認識走行

6.1　　小型 WiFi カメラの利用

スマートフォンは RC モデルに対して Bluetooth による無線接続で命令を送信する．そのため，RC モデルから映像を受信する際も無線経由でなければならない．そこで，RC モデルに直接装備できる小型の無線カメラとして，プラネックスコミュニケーション社製の CS-W07G-CY と呼ばれる小型 WiFi カメラ[8]を使用した．カメラから直接映像を取得でき，また，無線 LAN ルータを介してインターネット上から映像を取得することもできる．

図 6.1　プラネックスコミュニケーションズ社製 WiFi カメラ

このカメラはリチウム電池でも稼働することのできるので携帯性や設置する場所を選ばない点で優れている．しかしリチウム電池で稼働させていると稼働時間が少ないという問題点があった．そこでモバイルブースタと呼ばれる携帯型 USB 給電バッテリーを RT-ADKmini の電源と共有することでスペースの確保と安定した給電時間を両立させた．

[8] 現在プラネックスコミュニケーションズはこの商品を取り扱っておらず，Trek 社が同型のものを扱っている．

6.1.1　プラネックスコミュニケーション社製小型 WiFi カメラの仕様

以下はプラネックスコミュニケーションズ社製 WiFi カメラの詳細な仕様である.

表 6.1　WiFI カメラの仕様 I

カメラ部仕様	
映像素子	1/9インチCMOS
レンズ	F:2.8, フォーカス:オートフォーカス(20cm以上)
視野角/画素数	垂直:60°/30万画素
解像度	VGA(640×480), QVGA(320×240), QQVGA(160×120)
ホワイトバランス	自動
ゲインコントロール	自動/自動
基本機能	
画像圧縮方式(動画)	Motion-JPEG
画像圧縮方式(静止画)	JPEG
フレームレート	30fps
画質設定	明るさ, コントラスト
ネットワーク設定	固定IPアドレス, DHCPクライアント
無線部仕様	
対応規格	IEEE802.11g, IEEE802.11b
チャンネル数	1～13ch
周波数帯(中央周波数)	2.4GHz帯(2,412～2,472MHz)
伝搬速度	IEEE802.11g:54, 48, 36, 24, 18, 12, 9, 6Mbps IEEE802.11b:11, 5.5, 2, 1Mbps
伝搬方式	IEEE802.11g:直行周波数分割多重(OFDM方式) IEEE802.11b:直接拡散型スペクトラム拡散(DSSS方式)
アンテナ	内蔵アンテナ1本(1T1R)
アクセス方式	インフラストラクチャモード, アドホックモード
到達距離	インフラストラクチャモード:約20m, アドホックモード:約7.5m
セキュリティ	WEP(オープン/共有), WPA-PSK/WPA2-PSK

表 6.2　WiFi カメラの仕様Ⅱ

ハードウェア仕様	
LED/インターフェース	Power/モード切替スイッチ（ON/OFF/セットアップ）
電源	リチウム電池（CR2）×1本：3V　750Ah　　シガープラグ：5V　1A
消費電力	300mA
外形寸法	本体：約30（W）×30（H）×35（D）※突起部除く クレードル：約56（W）×63（H）×56（D）※水平設置時 シガープラグ：約32（W）×93（H）×20（D）※突起部除く
重量	本体：約10g　　クレードル：約17g　　シガープラグ：約30g
動作時環境	温度：0～50℃　　湿度5～90%（結露なきこと）
保管時環境	温度：-10～60℃　　湿度5～90%（結露なきこと）
ソフトウェア仕様	
対応OS	Windows7（32bit/64bit）　Vista（32bit/64bit）　XP（32bit） Mac　OSX　10.6/10.5/10.4（Intel/PowerPC対応） iOS　4以上　　Android　2.2以上
動作環境	Internet Explorer 6.0以上，Safari 4.0以上，Firefox 3.6以上，専用アプリ

表 6.1，表 6.2 から最大解像度は VGA の 640×480 であり，本研究ではこの最大解像度で映像を取得する．また，フレームレートは 30fps であるが WiFi での接続なため周囲の無線環境や受信側の端末との距離によって受信状況が変動する．

6.2　Android スマートフォン上での画像処理

本章では Android スマートフォン上での画像処理について述べる．Android スマートフォンが WiFi カメラから取得してきた画像に Java で直接画像処理を行う方法と，JNI インターフェースを使用して画像処理を行う方法の 2 種類である．

画像処理を行うに当たり，カメラ視点はなるべく高い位置にあることが理想とされる．仮にカメラが低い位置にある場合，地面よりも風景が多く映ってしまうと考えられる．これでは路面状況を判断するための画像処理には適していない為，高い位置から地面を見下ろすことで路面面積を増やし，画像処理に適した視点を確保する．

図 6.2　WiFi カメラの設置位置

図 6.3　車体ルーフに設置した WiFi カメラからの視野

図 6.2 はカメラの設置位置であり，図 6.3 はカメラがとらえる視界である．このように画面の半分以上を地面にすることができた．この WiFi カメラは 1 秒間に 30 フレーム画像をスマートフォンに送信するが，図 6.3 のときは 14fps で受信している．これはフレーム落ちや WiFi の受信感度などが考えられる．

6.2.1 Java 上での画像処理

本章では道路の白線の中央を自律走行させることを画像処理により実現する．

まず取得画像を二値化し，白線を白，走行車線を黒に変換する．次に画像処理内の白いピクセルの重心を計算し，この重心点を追うように RC モデルを制御する（図 6.4）．

図 6.4 画像処理による重心計算イメージ

本研究における画像処理は，重心計算を行うことが主であるため図 6.4 のように画面全体を画像処理するのではなく，シークバーで画像処理範囲を調整変更できるよう図 6.5 のようにソースコードを記述した．

```
183         for( int XX = 0; XX < width2; XX+=2 ){
184           for( int YY = activity.IPRangeSeeki; YY < activity.IPRangeSeekj; YY+=2 ){
185
186             int bitmapColor = pixels[( XX + YY * width2 )];
187
188             int rr = Color.red( bitmapColor );
189             int gg = Color.green( bitmapColor );
190             int bb = Color.blue( bitmapColor );
191
192
193             int X, Y;
194             Y = ( rr + gg + bb ) / 3;
195
196             if( Y < activity.getThreshold() ){
197               X = 0;//輝度値
198             } else {
199               X = 255;//輝度値
200             }
201
202             if(X==255 && YY>3*height2/4){
203                 n = n +1;
204                 Sx = Sx + XX;
205                 Sy = Sy + YY;
206             }
207
208             rr = X;
209             gg = X;
210             bb = X;
211
212             pixels[( XX + YY * width2 )] = Color.rgb( rr, gg, bb );
213
214                 }
215             }
```

図 6.5　Java 上での画像処理ソースコード

画像処理範囲を画面一部に限定するが，実験を行う際，カメラの視界に合わせて画像処理範囲を変更できるようにシークバーから範囲を設定できるようにした．また，画像処理を行うピクセルを for 文内の XX+=2，YY+=2 で縦，横 2 ピクセルごとに処理を行うことで全てのピクセルを処理した時よりも 1/4 の処理量になる．図 6.5 下の赤枠内は，二値化した画像の白いピクセルを数え Sx，Sy に格納している．

```
217     if(n>=1){
218
219         Gx = Sx / n;//CoG center of gravity
220         Gy = Sy / n;//CoG center of gravity
221
222          activity.setCoG(Gx);
223     }
```

図 6.6　Java 上での重心座標の計算

図 6.6 は図 6.5 で数えた白いピクセルの数から重心座標を計算し，重心の x 座標である Gx を Java プログラムにある他の Activity に渡している．

図 6.7　Java 上での画像処理のスクリーンキャプチャ

図 6.7 は Java で画像処理を行った時のスクリーンショットである．赤枠内が限定した画像処理範囲である．網目状に見えるのは縦横 2 ピクセルごとに二値化処理しているからである．この画像処理範囲は右側赤枠内のシークバーから変更可能になっている．また右下の青枠は現在の fps 値[9]は 2fps であることがわかる．さらに，遅延時間が 5 秒ほど発生した．これでは，リアルタイムで画像処理を行い RC モデルを制御することはできない．そこで画像処理の高速化を行う．

[9] この fps 値はあくまで参考程度の意味合いであり，実際の処理速度を表しているわけではない．また処理速度自体も WiFi カメラを使用している都合上，周囲の電波環境にも左右されることがある．

6.2.2　JNI の利用

JNI とは Java Native Interface の略で，Java で記述されたプログラミングソースと他の言語（C 言語等）で記述されたプログラミングソースを連携するためのインターフェースである [20].

JNI を使用することにより各プラットフォームで実行可能な形式に変換されたコードを Java 上に呼び出すことが可能になる．これにより、各プラットフォームに適した形のプログラムが実行できるためパフォーマンスが格段に向上する. しかしながら JNI にも問題があり，呼び出されるプログラミングソースは各プラットフォームに依存した形に変換されるため，同じバイナリでは異なるプラットフォーム上では実行できない.

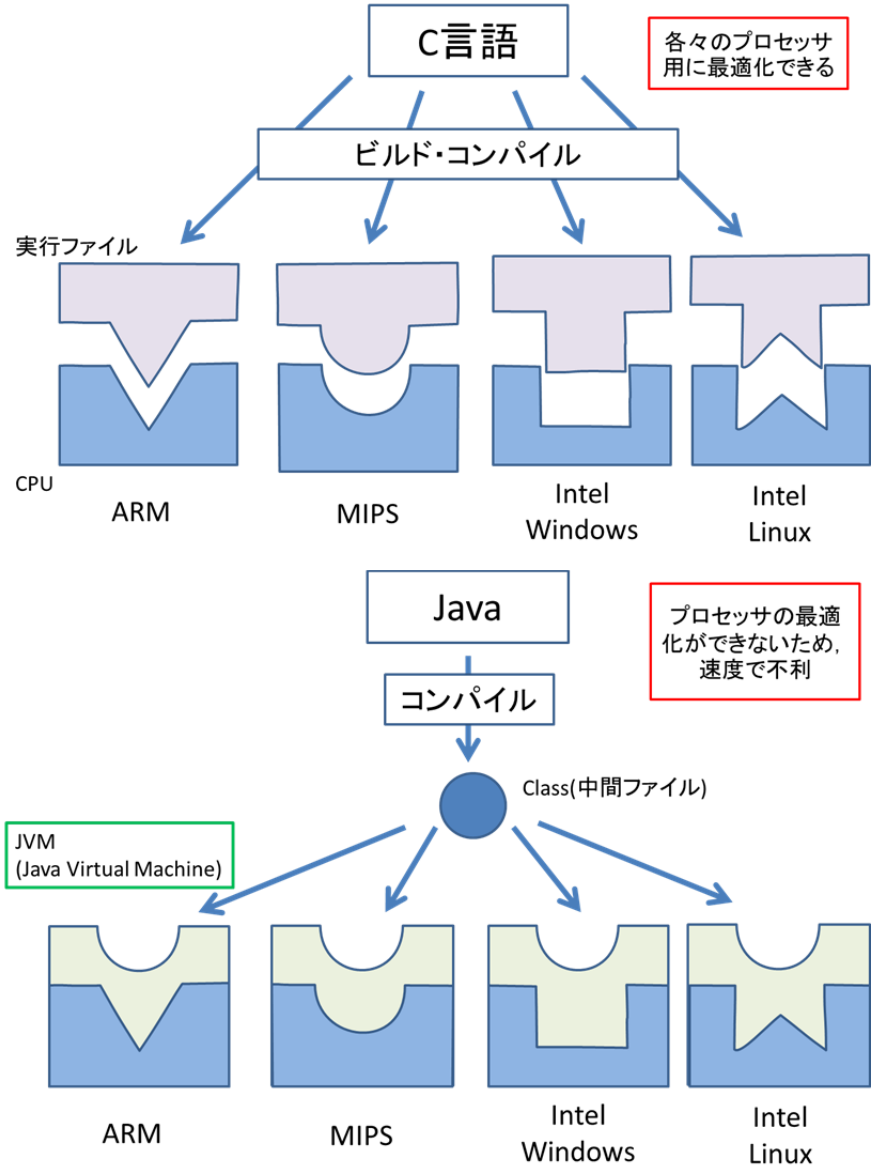

図 6.8　C 言語と Java のコンパイル簡易図

図 6.8 は C 言語コンパイルの簡易図と Java コンパイルの簡易図である．この図からわかるように C 言語は個々のプロセッサ用に最適化している形をとることができるので処理時間が短くなる．しかしJavaは個々のプロセッサに最適化することはできず，Java Virtual Machine を上のみ実行できる．そのためどのプロセッサでも動作させることはできるものの速度では劣ってしまうことが多い．

本研究では WiFi カメラから画像データを取得する部分で既にこれを使用している．ここに画像処理を追加することで画像処理を高速化できる．なお，JNI 利用にあたっては [21]を参考にした．

```
160     if(bmp==null){
161         bmp = Bitmap.createBitmap(IMG_WIDTH, IMG_HEIGHT, Bitmap.Config.ARGB_8888);
162     }
163     mIn.readMjpegFrame(bmp);
164     int Gx = imageprocessing(bmp,activity.getThreshold(),
165                              activity.getThresholdi(),activity.getThresholdj());
166     destRect = destRect(bmp.getWidth(),bmp.getHeight());
167     activity.setCoG(Gx);
```

図 6.9　Java 側の JNI の設定

本研究では重心座標は x 座標のみ必要である為，JNI から取得するのは重心の x 座標 Gx のみである． 図 6.9 の int Gx 以下の文は，JNI で画像処理した値を取得している．また赤枠下段は JNI 上で計算した重心の x 座標を Java の他の Activity に受け渡すための文である．

次に JNI を利用して画像処理を行う為に C 言語でプログラミングソースを記述する．

```
419  for(i=0 ; i<IMG_WIDTH ; i+=2){//i<IMG_WIDTH
420    for(j=IPRangeSeeki ; j<IPRangeSeekj ; j+=2){      1
421      int blue =(0xff0000 & pixels[getIndex(i, j)])>>16;
422      int green =(0xff00 & pixels[getIndex(i, j)])>>8;
423      int red =0xff & pixels[getIndex(i, j)];
424
425      // 浮動小数演算を行うとパフォーマンスが悪いので整数演算で
426      //int Y = (int)(0.299f*red + 0.587f*green + 0.114f*blue);
427      int Y = (299*red + 587*green + 114*blue)/1000;
428      blue = green = red = Y;
429
430      int XX,YY;
431      YY = (blue + green + red)/3;
432      if(YY < ThresholdSeek){  2
433        XX = 0;
434      }else{
435        XX = 255;
436      }
437
438    red = XX;
439    green= XX;
440    blue = XX;
441
442    if(XX == 255){
443        n = n+1;
444        Sx = Sx+i;    3
445        Sy = Sy+j;
446    }
447      pixels[getIndex(i, j)] = 0xff000000 | blue<<16 | green<<8 | red ;//白黒の情報を打っている
448
449  }
450 }
```

図 6.10　JNI を利用した画像処理 C 言語ソースコード

図 6.10 は C 言語で記述した二値化と重心計算を行っているソースコードである．1 は画像処理範囲をシークバーから変更できるようにしてあり，また Java 上での画像処理と同様に i+=2, j+=2 として 2 ピクセルごとに画像処理を行うことで画像処理の高速化を図っている．また，図中 2 も同様に，シークバーから閾値の変更ができるようになっている．シークバーの詳細は第 6 章 4 項にて説明する．図中 3 は白ピクセルの重心を計算するために画像処理範囲中の白ピクセルの個数と座標を算出している．

```
452     if(n >= 1){
453      Gx = Sx/n;
454      Gy = Sy/n;
455      }
456 pixels[getIndex(Gx, Gy)] = 0xffff00ff;
457 return(Gx);//Gxの値を返す
```

図 6.11　重心座標の計算

図 6.11 で図 6.10 中の 3 で得られた白ピクセルの情報から重心座標を計算し，重心の X 座標である Gx の値を返すことで Java 上で Gx の値を使用することが可能になる．

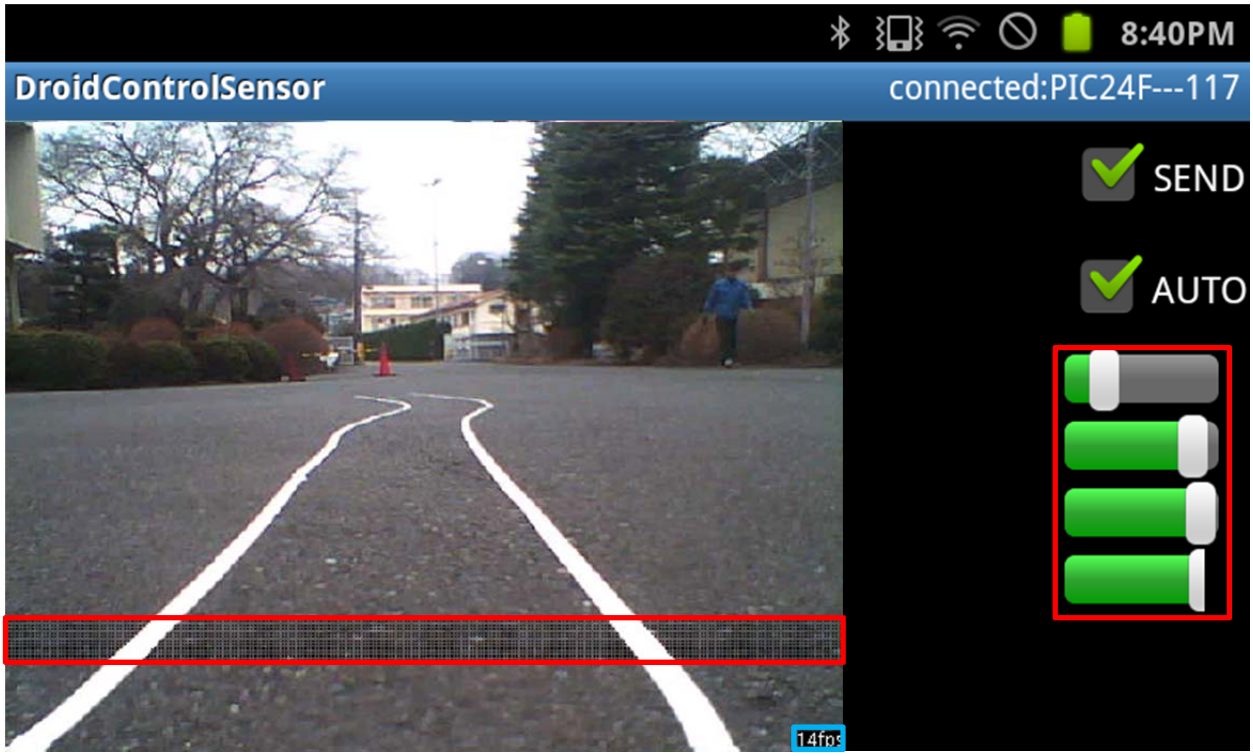

図 6.12　JNI を利用した画像処理のスクリーンキャプチャ

図 6.12 は JNI に画像処理を書いて実行した場合のアプリケーションのスクリーンショットである．赤枠で囲われた部分が画像処理範囲であり，この範囲と二値化の閾値はアプリ右手にあるシークバーで変更可能である．この時のフレームレートは 14fps となっており，全く同様の条件でキャプチャした Java 上で画像処理を行った図 6.7 の 2fps よりも格段に早くなっていることがわかる．また，遅延時間もほぼ無くなり，画像処理の高速化は成功したといえる．

6.3　　画像処理による RC モデル制御法

画像処理で得られた白線の水平方向の重心をもとに RC モデルのステアリングを制御することで車道中心を走行させる．まず，WiFi カメラは車体の天井部中央に設置する．これは車体の中心線と WiFi カメラの中心がそろうようにするためである．また，WiFi カメラの取得映像サイズは 640×480 のため，X 軸中央の座標は 320 である．

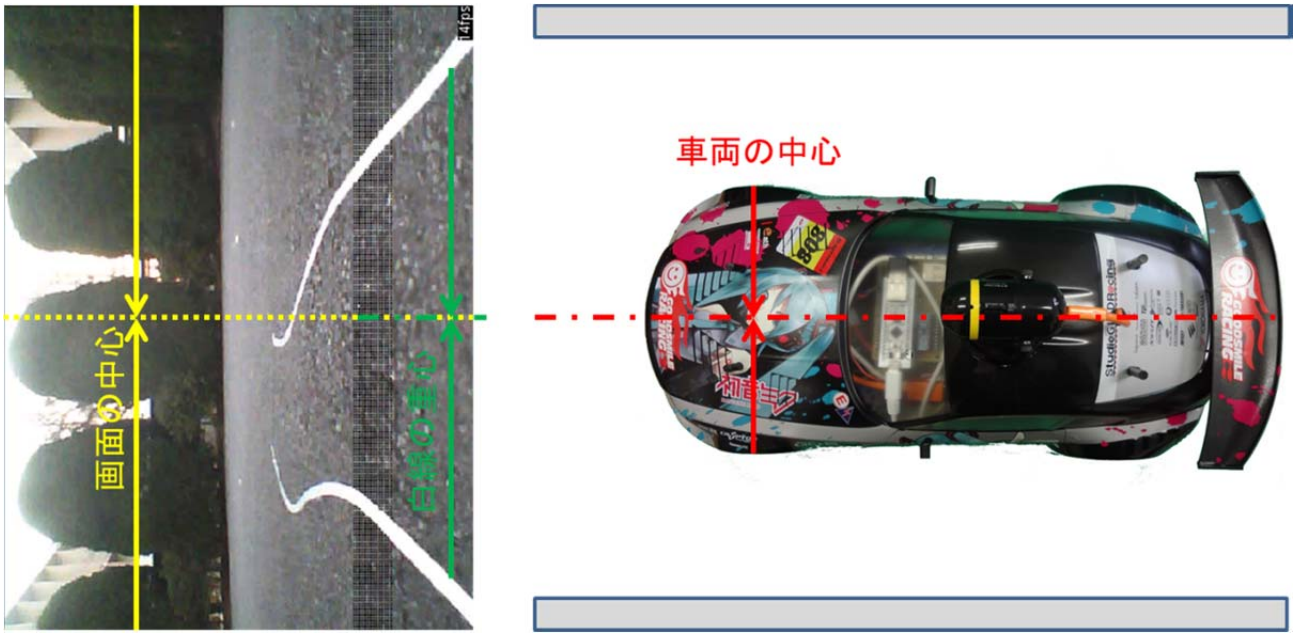

図 6.13　中心線の一致

つまり，WiFi カメラが取得してきた映像の中心が車体の中心となる．また，車道外側線に囲まれた車線の中心は車道外側線の重心座標を結んだ線と等しい．よってこの WiFi カメラから取得した映像の X 軸の中心座標と，車道外側線の重心を重ねるようにステアリングを制御することで車道の中央を走行させる．

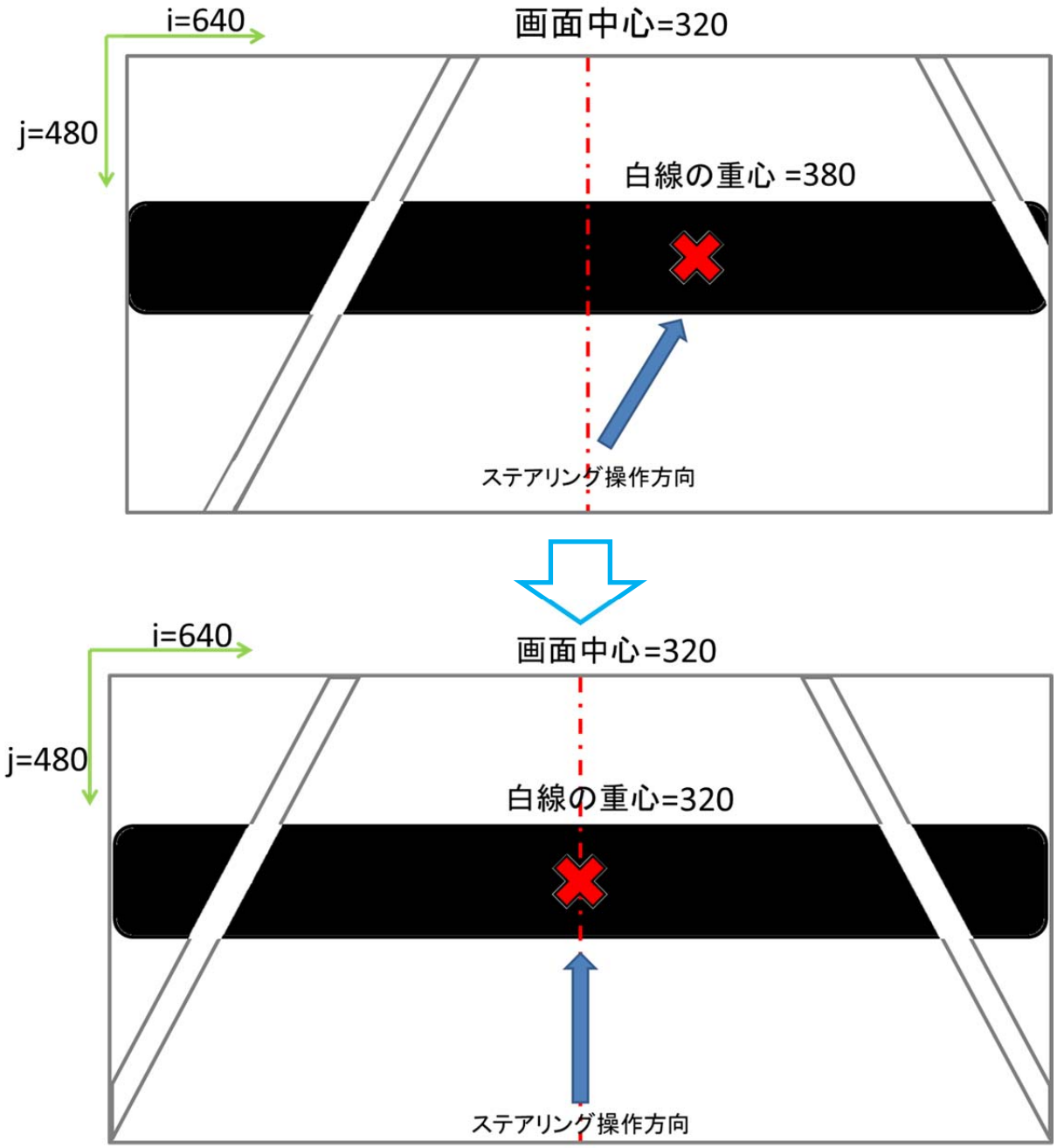

図 6.14　重心のずれとステアリングの向きの関係

図 6.14 は重心とステアリングの向きとの関係図である．まず，重心の X 座標が 320 より大きい場合は右，小さい場合は左，という単純な ON/OFF 制御プログラムを作成した．しかし，このプログラムで走行実験をした場合，白線の中央を維持しようとステアリングが動作するが，ON/OFF 制御なため蛇行してしまい，とても実用的とは言えない．そこで目標値である画面中心と重心との偏差に応じてステアリング角を決めるために PID 制御を用いる．

6.3.1　PID 制御

速度制御や温度制御などをはじめとした各種制御に用いられる制御方式として PID 制御がある．PID とは Proportional（比例），Integral（積分），Differential（微分）の頭文字であり，それぞれが表すように比例要素，積分要素，微分要素がある [22].

6.3.2　P 要素

本研究では，画像処理によるステアリング制御が単純な ON/OFF 制御の場合，ステアリング操作量が目標である重心座標の間を行き来してしまい蛇行してしまう．

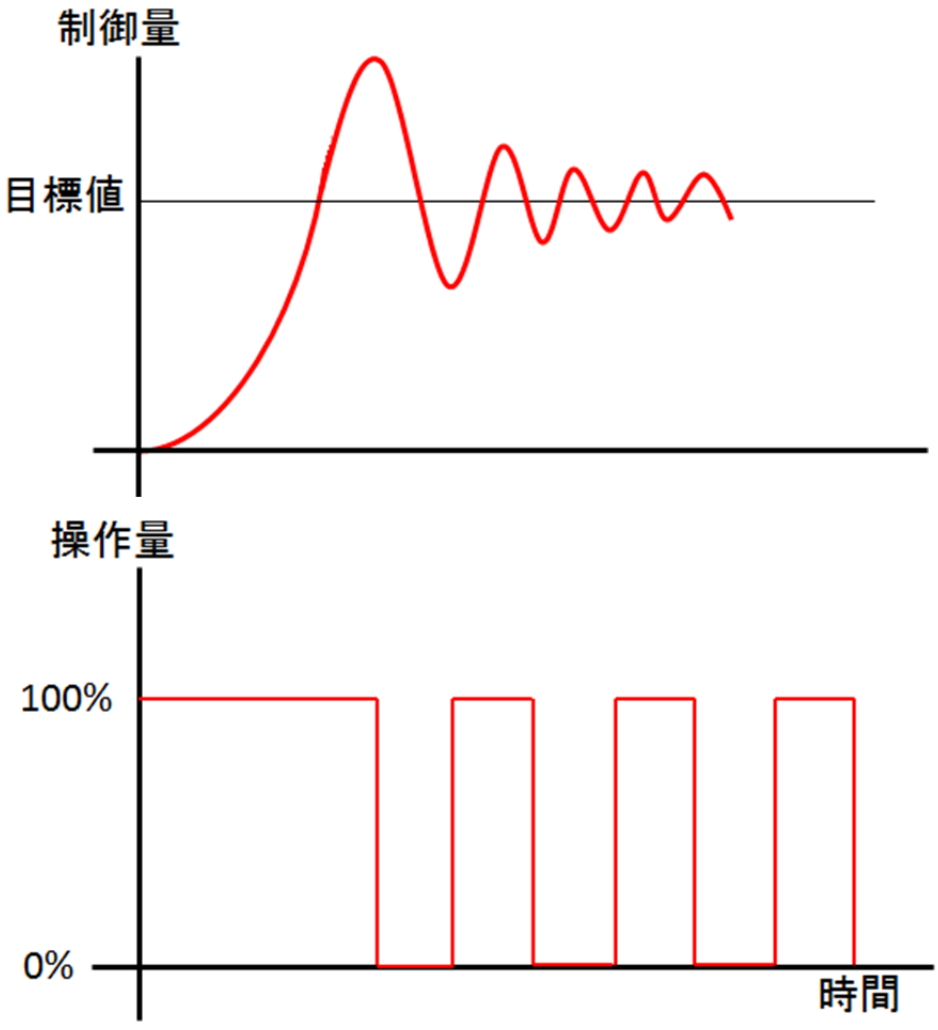

図 6.15　ON/OFF 制御の特性

これに対し, PID 制御の P 要素では目標値からの偏差に比例した量を制御量とする．そのため偏差が大きいほど制御量は増え，小さいほど操作量は減少する．重心座標が車道中央から離れるほどステアリング角を大きくとり，近ければ小さくとることとする．P 制御は偏差×定数で操作量を決定する．

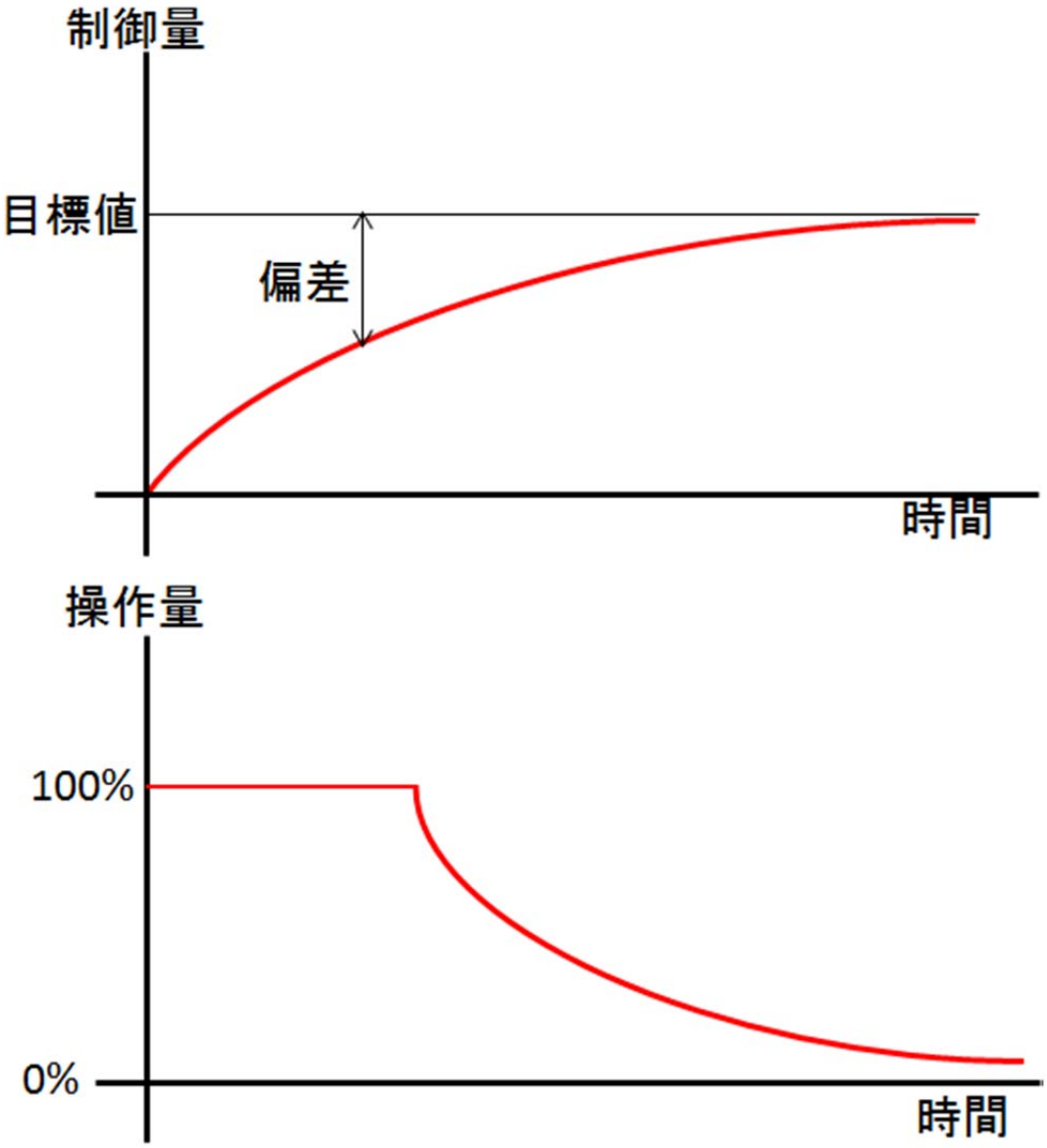

図 6.16　P 要素の特性

P 要素の欠点として，偏差が小さいときに制御量が小さくなりすぎてしまい，制御量が目標値とずれた値で安定してしまうことがある．これをオフセット，または定常偏差と呼ぶ.

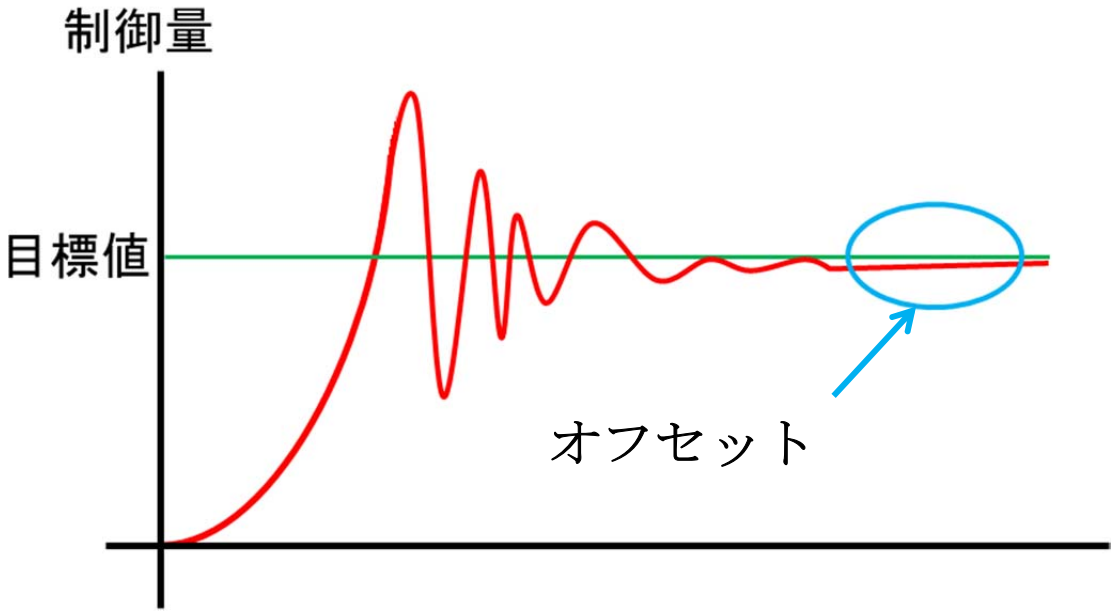

図 6.17　P 要素の欠点

図 6.17 のように P 要素だけではオフセットが発生してしまい，目標値付近を走行し続ける．また，直線でもカーブでもその時点の制御量に応じた操作を行い走行するため，オフセットが発生し，カーブ走行中に偏差が小さいと制御量も小さくなりコースアウトしてしまう可能性がある. P 要素にかける定数を大きくすることである程度防げるが，カーブ中の走行を安定させることはできても直線走行の安定性はなくなってしまう．そこで，オフセットを無くしカーブを走行している間のみ制御量を上げるために I 要素を用いる.

6.3.3　I 要素

I 要素は積分項であり，偏差の累積×定数で制御量を決定する．また，偏差の累積であるため，小さな偏差でも徐々に大きくなり，P 要素で対応できなかったオフセットを解消することができる．

6.3.4　PI 制御

6.3.2 と 6.3.3 を合わせた制御が PI 制御である．PI 制御は以下の式から求まる．

$$制御量 = K_P e + K_I \int edt$$

e は目標値と現在の重心との偏差，K_P，K_Iは定数でこれをパラメータといい，このパラメータで P 要素，I 要素の加減を調整する．これらをステアリング制御に反映させることで直線，カーブともに白線の中心を走行させる．

```
985         Pval = cog - dispcenter;//重心 - 目標値 = 偏差を取得 比例項
986         Ival += Pval;//2*0.04;//偏差の累積値 積分項
987         PIval = (Pseekf*Pval + 0.01*Iseekf*Ival);//0.01かけvar
```

図 6.18　P 要素，I 要素 PI 制御の式

図 6.18 は P 要素，I 要素，PI 制御のソースコードである．これは Java 上に記述した．P 要素は JNI で求めた重心の x 座標 cog から目標値である画面中央の x 座標 dispcenter との偏差である．I 要素は P 要素で求めた偏差の累積である．P，I 要素に定数をかけ足すことで PI 制御を行っているが，本研究では実験により定数を求めるためこの定数はシークバーで変更できるようにした．Pseekf と Iseekf がそれである．また，I 要素には PIval の式で 0.01 をかけているが，これは I 要素がステアリング制御に対して大きすぎたため，制御に反映することのできる値に収めるためである．

```
1011                if(autoMode){
1012                    pwm_width1 = ThresholdSeekD;
1013                    sendMessage(String.format("s%02x", pwm_width1));
1014                }
1015                if(autoMode){
1016                    pwm_width2 = (int)(-PIval+100);
1017
1018
1019                    sendMessage(String.format("t%02x", pwm_width2));
1020                }
```

図 6.19　PI 要素の制御部

図 6.19 は PI 制御をステアリング制御に使用しているソースコードである．緑色の枠で囲われている箇所は，自律モードになった際にシークバーで指定した速度に固定している．赤枠で示した個所で PI 制御で定めたステアリング角を定めている．ステアリングのニュートラル値である 100 を足すことで PI 制御が 0 の時, つまり制御量 0 の時にまっすぐ走行する．

以上でステアリングの PI 制御化が完了し，P 要素，I 要素の定数を実験から決定する．

6.4　コントローラアプリケーションのオプション機能

画像処理による半自律走行を含んだアプリケーションにシークバーを追加し，アプリケーション上から様々な数値を変更ができるようにした．

変更できる値は，二値化の閾値，RC モード，ステアリング自律モードの時のアクセルの最高速度，画像処理の範囲，P 要素，I 要素の定数の 5 種類である．

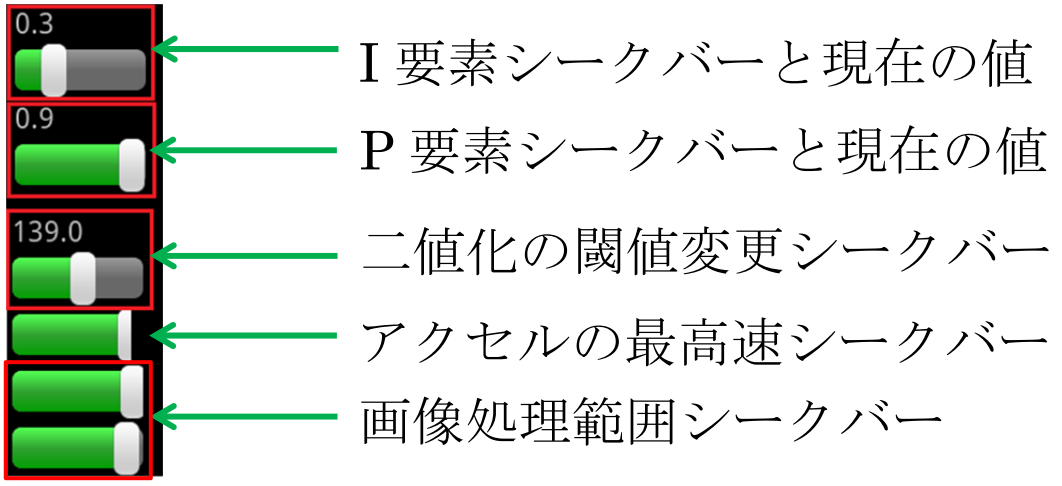

図 6.20　コントローラアプリケーションの各種シークバー

これにより実験中に現在の周辺状況や天候に合わせた各種値の変更が可能となった．このシークバー機能やその他のアプリケーション作成にあたっては [23]を活用した．

6.5　実装

制御回路，プロポ，モバイルブースタ，WiFi カメラをすべて RC モデルに実装する．

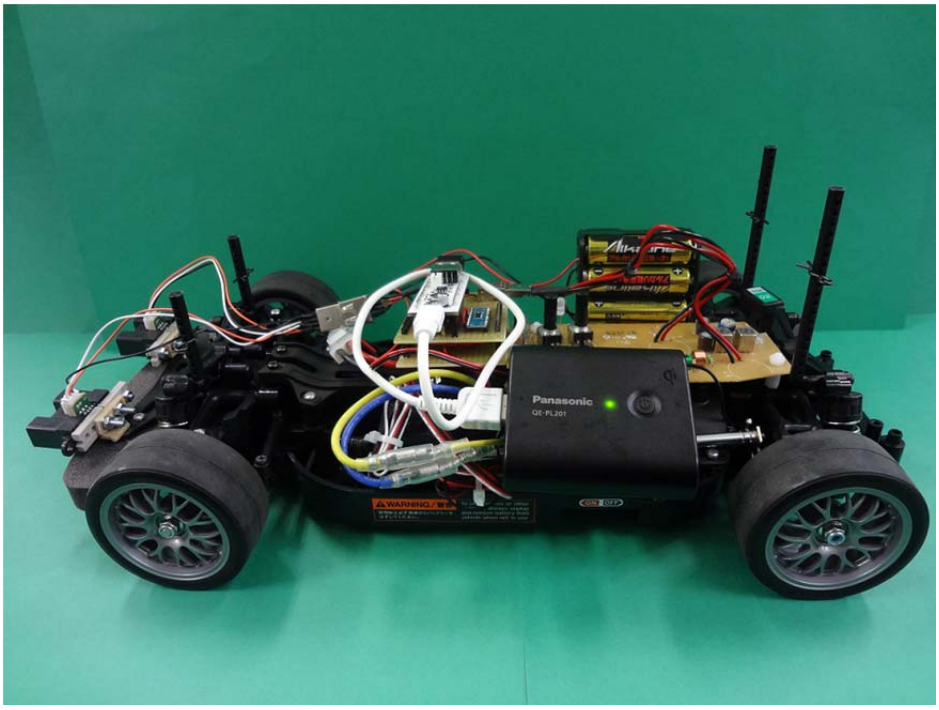

図 6.21　シャシーへの回路実装

RT-ADKmini と WiFi カメラの電源には 5V のモバイルブースタを使用する．ただし，プロポには 12V の電源が必要なため乾電池を用いる．これらは走行実験で落下しないようにネジやスペーサーを使用して固定する．

WiFi カメラはボディの天井部分に設置した．これはもっと視点が高くなる位置に取り付けることでカメラの視野を広くとることができ，画像処理時に遠くの白線まで見渡せるようにするためである．

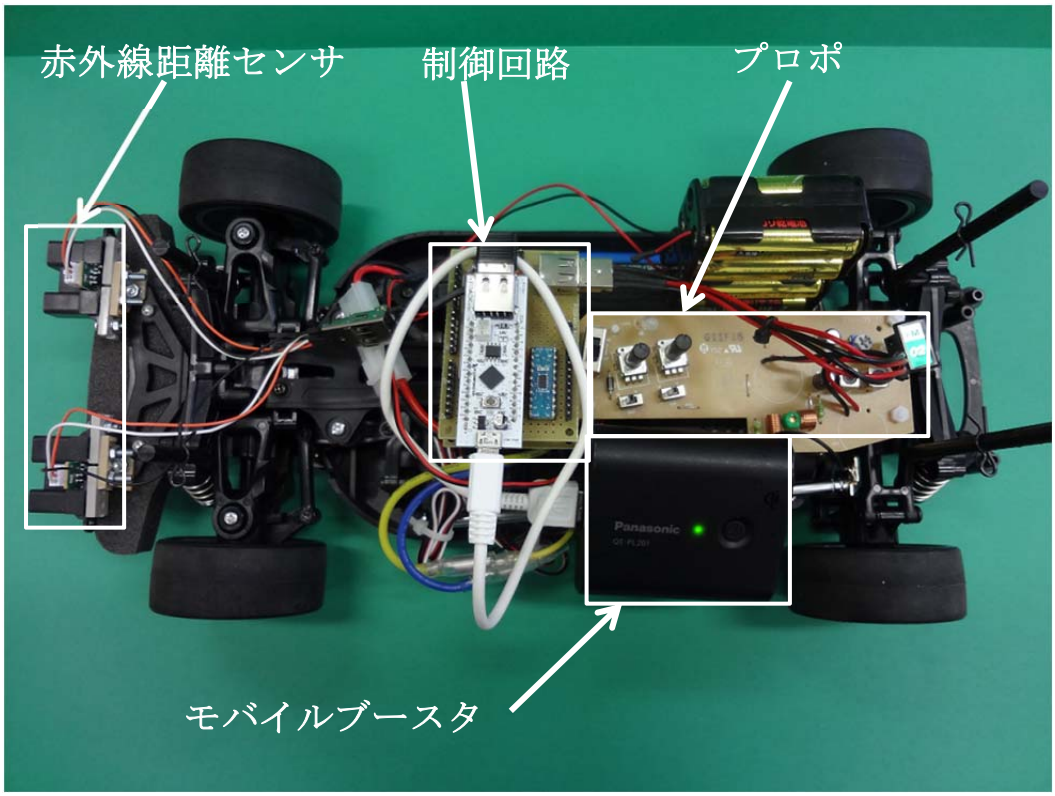

図 6.23　シャシーへの回路実装

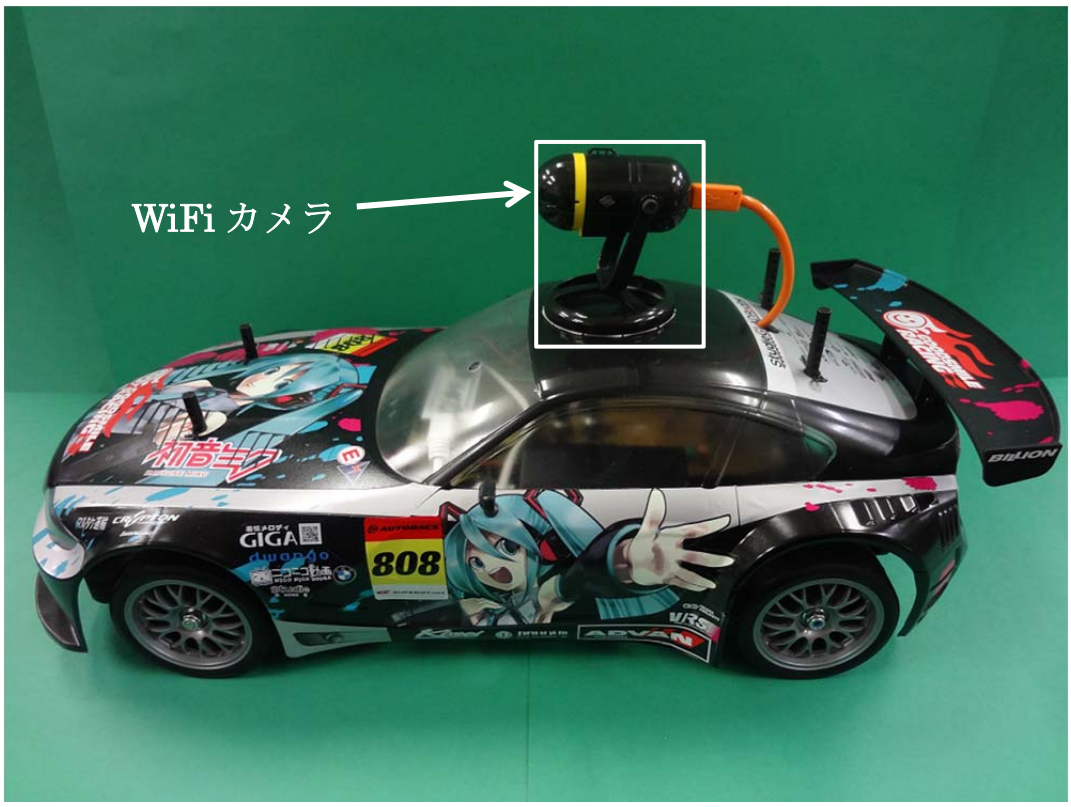

図 6.22　実装外観

6.6　　走行実験

実験では実際の環境を想定し，アスファルトの地面に白いビニールテープで直線のコースを作り，車道外側線や車線境界線に見立てた．

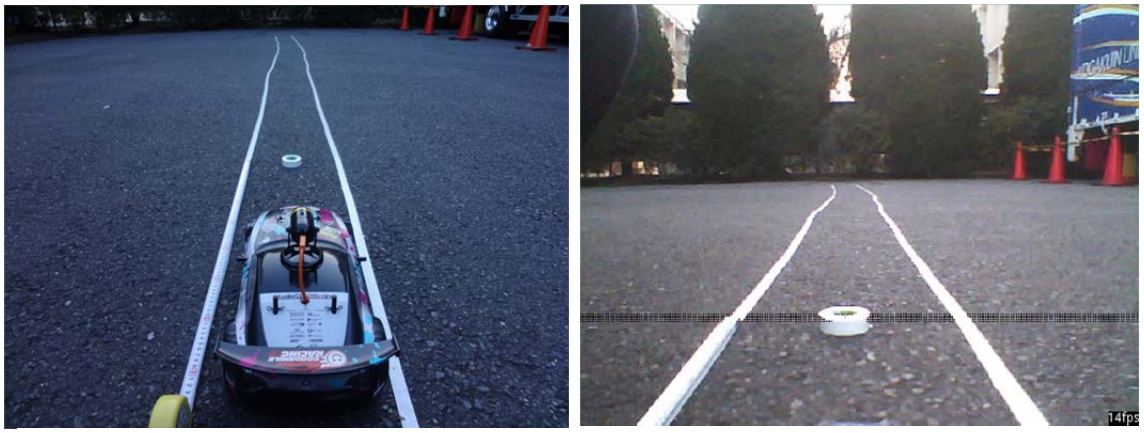

図 6.24　実験風景と WiFi カメラから見える視界

RC モデルを車線中央からずらした位置に置き，P 要素，I 要素の定数を 0.0~0.3 まで 0.1 刻みで増加させていき，車線中央に戻る速さ，中央を維持しながら走行できるかを実験した．I 要素は図 4.18 で示したように 0.01 をかけるので実際には 0~0.0003 である．初めから偏差を与えた状態で走行し始めることで P 要素，I 要素の働きを調べ，最適なパラメータを割り出すことが目的である．

図 6.25　車線中央より右よりに RC モデルを置く

また，実験環境は天候の荒れた日や夜間以外とした．これは夜間になると WiFi カメラの性能上，周囲を撮影することができなくなるためである．

実験中の目標値の変化を SD カードに書き出す機能をアプリケーションに実装することで, 実験中の状況をグラフで確認できるようにした. PI 制御のパラメータK_P，K_Iを様々に変えて実験を行い最適なパラメータを探った．それぞれのパラメータごとに 10 回ずつの走行実験を行っている.

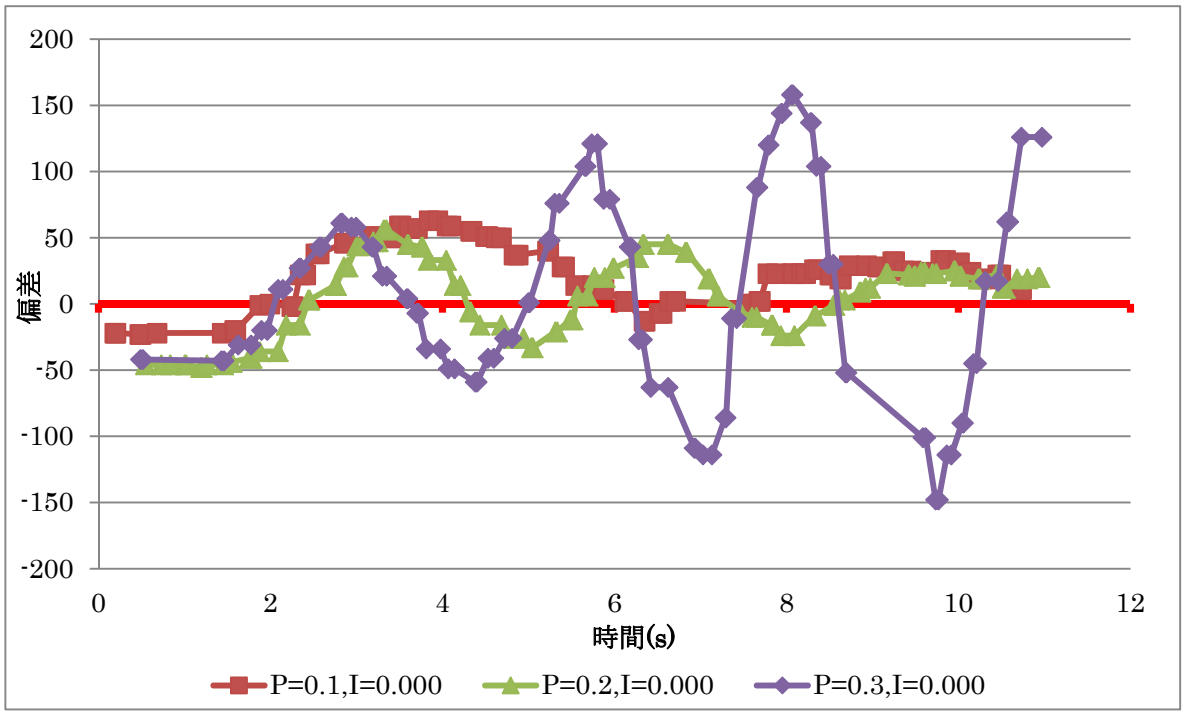

図 6.26　P 要素のみを変化させたときの偏差

図 6.26 は P 要素のみで走行させた場合のグラフである．P=0.1 の場合，緩やかな減衰振動をしており目標値に近づいているが，定数が小さいため目標値へ修正する速度が遅い.

P=0.2 の場合，P=0.1 に比べ振動する回数は増えたが減衰振動をしており，目標値に修正する速度も速い．しかし，これら 2 つは最終的に目標値付近で安定しておりオフセットが発生している.

P=0.3 は目標値へ修正する速度は最も早いが, 発散振動してしまい目標値から離れていってしまっている．これは走行しているコースに対して定数が大きいためである.

次に, P 要素を固定し I 要素を 0.001~0.003 までそれぞれ変化させ I 要素の働きを探る.

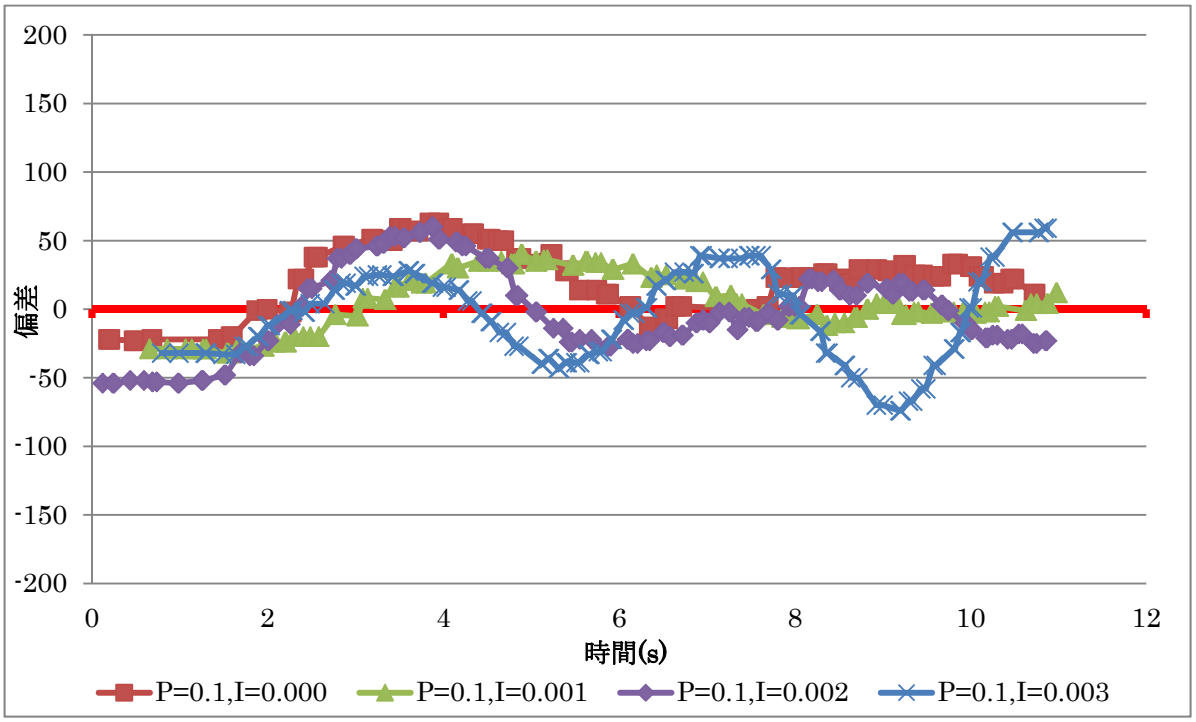

図 6.27　P=0.1 で固定し I 成分を変化

図 6.27 は P=0.1 として I 要素を 0.000~0.003 まで変化させ実験を行ったときのグラフである.

I=0.001 の場合，偏差が少なく目標値に近い位置で減衰振動をしている．また 8 秒から先を見ると, P 要素のみの場合と比べオフセットを無くす働きを確認することができる. I=0.002 の場合 I=0.001 の時よりも目標値への修正が早く減衰振動をしており, 6~8 秒付近でオフセットを無くす働きが見られる．しかし目標値への修正速度は遅いため, 直線での安定性は高いがカーブを走行した場合曲がりきれないことが考えられる. I=0.003 の場合，発散振動をしてしまっている．これは P 要素と同様に I 要素が大きいためである．よって，P=0.1 の時は I=0.001 または 0.002 が適当だと言える.

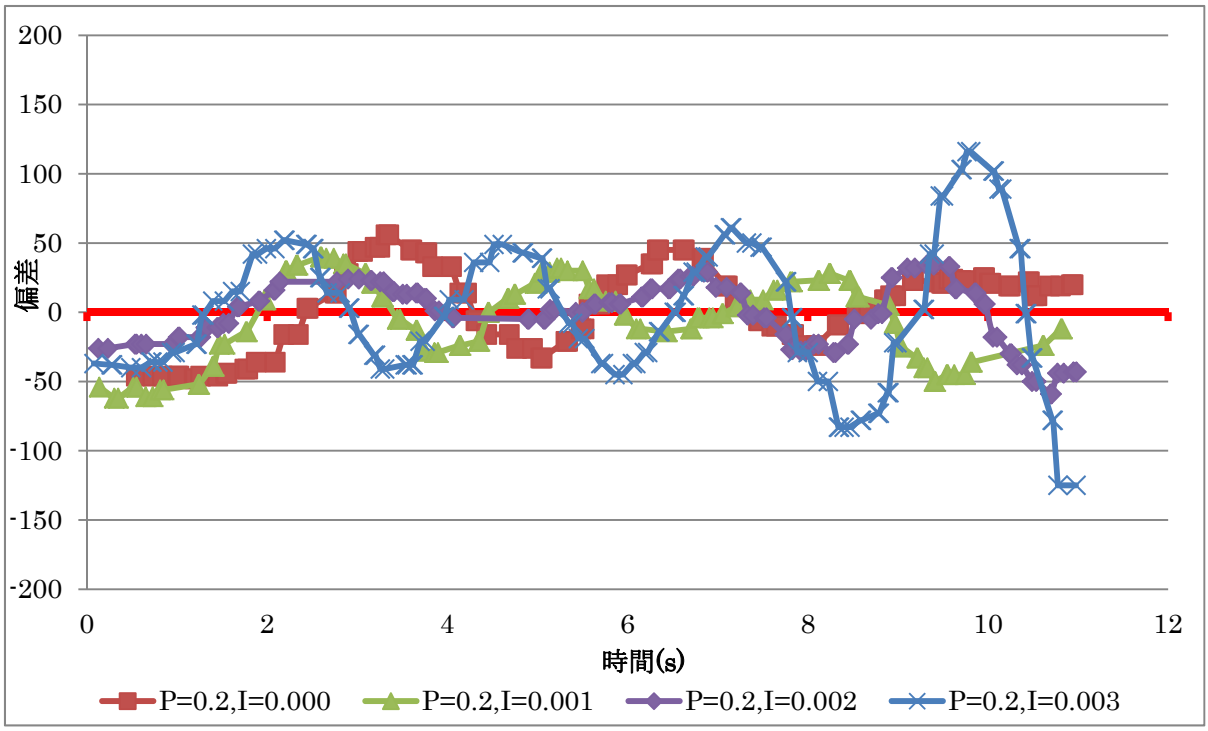

図 6.28　P=0.2 で固定し，I 成分を変化

図 6.28 は P=0.2 として I 要素を 0.000~0.003 まで変化させ実験を行ったときのグラフである．

I=0.001，0.002 ともに持続振動を取っているが I=0.002 は I=0.001 に比べ目標値に近い位置で振動している．これは目標値への修正速度が速いが，パラメータがコースに対して適当であるためである．I=0.003 になると I 要素がコースに対して大きくなってしまい発散振動をしており適当ではないことがわかる．このことからよって，P=0.2 の時は I=0.002 が適当だと言える．

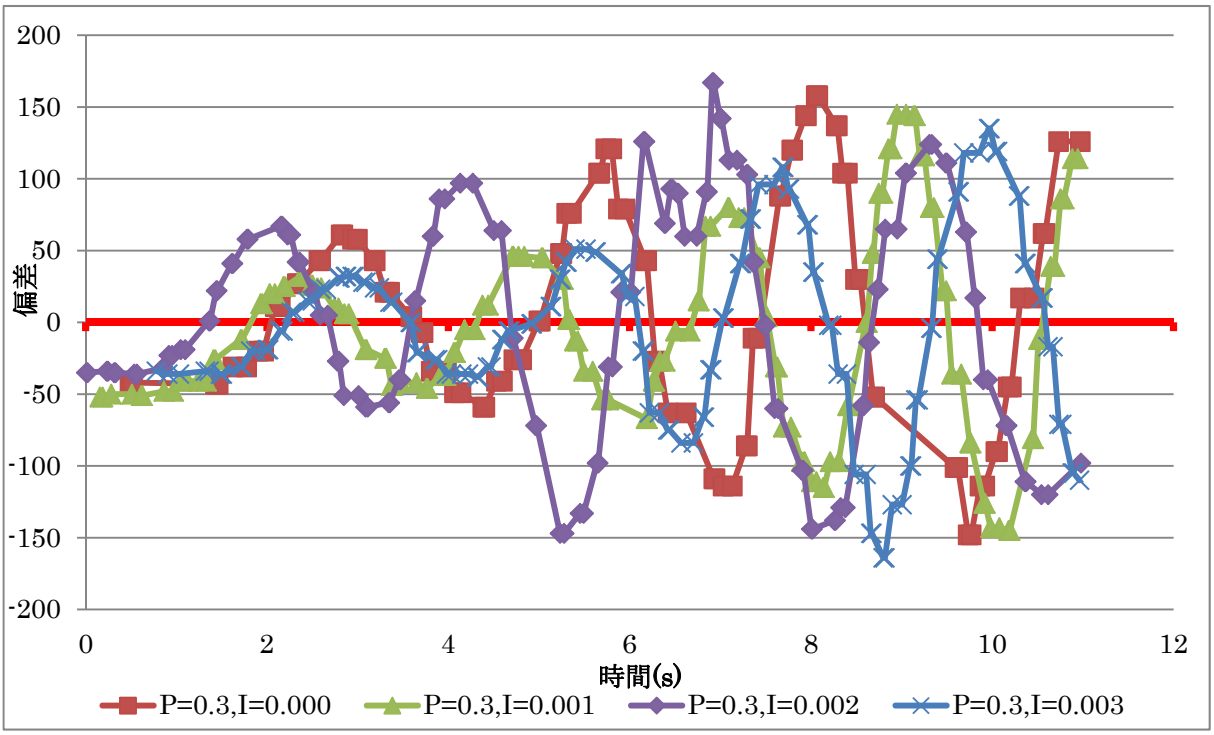

図 6.29　P=0.3 で固定し，I 成分を変化

図 6.29 P=0.3 として I 要素を 0.000~0.003 まで変化させ実験を行ったときのグラフである．

P 要素のみの場合ですでに発散振動をしてしまっており， I 要素を導入しても変化することなく発散振動強いてしまっている．このことから P=0.3 のパラメータは本研究では適切ではないと言える．

直線での安定性は P=0.1，I=0.001，0.002 の各パラメータが最も良いが，カーブを走行させる場合 P=0.2，I=0.002 が発散振動しない程度に目標値への修正速度が速いため適している．よって，直線とカーブ両方あるコースを走行する場合 P=0.2，I=0.002 が適切であると考えられる．このことから P=0.2，I=0.002 のパラメータを用いて直線とカーブの混合コースを走行させた．

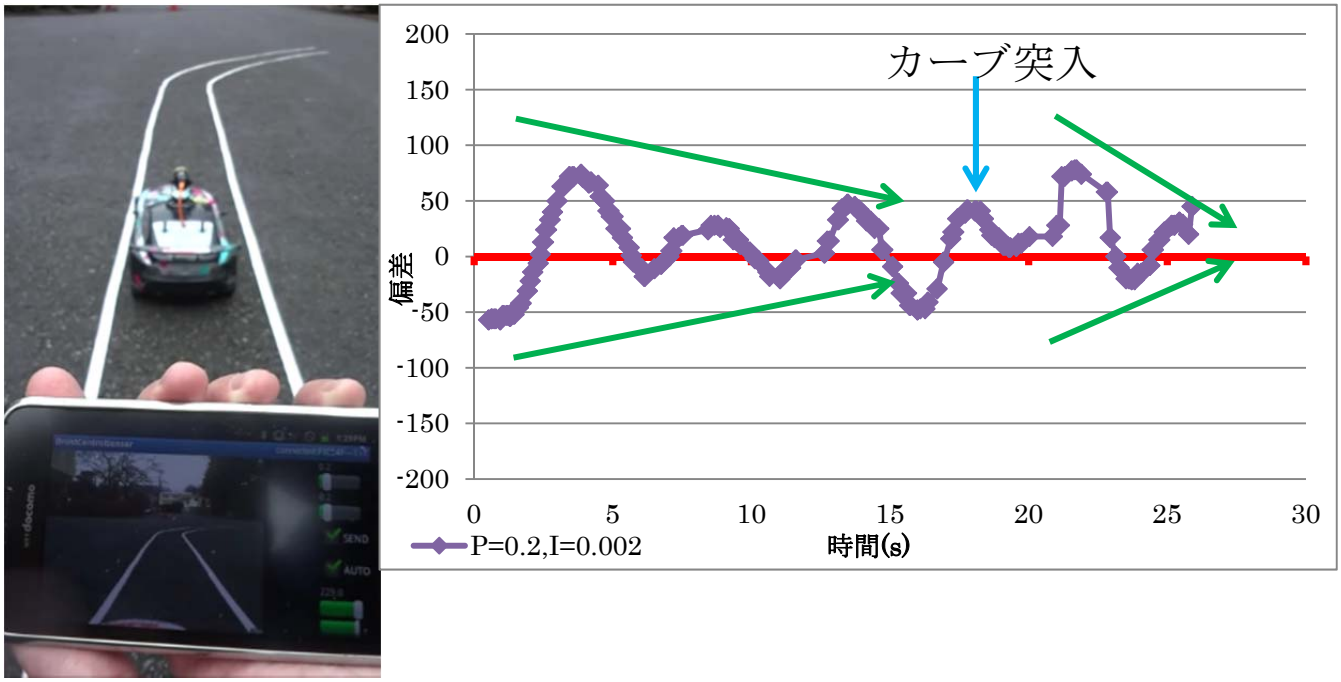

図 6.30　実験風景と実験時の偏差の変化

図 6.30 のように直線からカーブするコースでの走行実験を行った．10 秒付近までは直線を走行しており，減衰振動をしながら走行している．このまま走行することで目標値に収束してくことが考えられる．また， 15 秒付近でカーブに入り偏差が大きくなるがこれもすぐ修正しカーブ走行中も減衰振動を取っている．これらの事から PI 制御のパラメータは直線，カーブの両方を走行する場合は P=0.2，I=0.002 が適当だったと言える．

第7章　結論

7.1　スマートフォンによるRCモデルコントロール機能

まず, スマートフォンの傾きにより RC モデルのコントロールを行うシステムを開発した．コントローラは同時に WiFi カメラからの映像を受信している．これにより，映像を見ながらスマートフォンを傾けることでコントロールすることが可能である．

問題点として，スマートフォン内蔵の傾きセンサが Y 軸方向±80 度以上，Z 軸方向±80 度以上を同時に傾けると誤感知してしまい，その範囲ではコントロールが乱れてしまうことが挙げられる．また，スマートフォンと WiFi カメラは無線接続なためスマートフォンと RC モデルの距離が離れてしまうと受信感度が下がってしまい映像にコマ落ちが多くなる．しかし，これらは使用しているハードウェアの問題点であり，本研究におけるコントロール機能は人間が操作（運転）する機能であり，他の機能と両立，切り替え可能なことが重要である．

7.2　赤外線センサによる前方衝突回避機能

前方に取り付けた赤外線距離センサにより衝突回避を行った．これにより走行しているときは常に前方の障害物に対して一定距離以下まで近づいた場合に RC モデルを停止させ衝突回避のため自律的に制動をかける．

問題点として障害物を感知して停止した場合，前進することができなくなる．そのため，狭い駐車場などでもう少し距離を詰めたいといった時に前進ができず微調整が不可能になってしまう．そのため，赤外線距離センサを使用するかしないかを切り替えることができるようにする必要がある．また，自動車は走行速度により制動距離が変化する．これに対応するように走行速度に応じて検知距離を変化させることで様々な速度域での衝突防止が可能になる．

7.3 画像処理による白線認識自律走行機能

WiFi 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，車道中央を自律走行させることに成功した．画像処理に JNI を使用することで処理速度を向上させ，ステアリング制御に PI 制御を用いた．これにより重心からの偏差に応じてステアリグを操作させたが，P 要素，I 要素のパラメータはまだ検討の余地がある．また，シークバーから様々なパラメータを変更できるようにし，周辺環境に最適なものを探ることができる．

画像処理を行う際に，カメラから映像を取得→処理→命令送信のサイクル中にも RC モデルは走行し続けており，このサイクル中は命令が来ていない空白の時間である．この命令のない間を埋める処理を行うことが必要という提案があり，検証するべきである．また，本研究では白線認識は一定速度で行ったが，実際の自動車が一定速度で走行し続けることはまれであり，速度変化によって画像処理範囲を変更する必要がある．速度が遅いときは画像処理範囲を車両に近い範囲に，速度が速いときは画像処理範囲を車両から離れた範囲にすることでどの速度域でも白線認識が行える．

7.4 結言

コントロール機能，衝突回避機能，自律走行機能の 3 つの機能を実装し，人間の制御と自律制御を両立させることで，本研究の目的としていた半自律型自動車のモデルを構築した．しかし，赤外線センサによる衝突回避機能や画像処理による白線認識はまだ改良，検証するべき点が多い．

最後に，今後も自動車事故が少しでも減少し続け，ドライバーの自動車安全に対する意識や運転は楽しいと思える安全技術の発達に期待したい．

謝辞

本研究を行うに当たり，懇切丁寧なご指導を賜りました金丸隆志准教授に深く感謝し、本論文の執筆に当たり，ご指導賜りました濱根洋人准教授ならびに見崎大悟准教授に心より御礼申し上げます．また，本研究の要となった RT-ADKmini と Android スマートフォンの Bluetooth 通信連携のためのソフトウェアを公開している hrdakinori 氏に深く感謝します．これがなければ本研究はここまで進むことは困難であったと思っております．

学部生のころから 4 年間，金丸研のメンバーとして在籍できたことを誇りに思っております．とりたてて勉学のできる学生でもないにもかかわらず，遊んでいることが多く決して真面目に研究活動を行っているとはお世辞にも言えませんでした．もともとプログラミングなどの知識もなく，かといって勉強をするわけでもなかったため，修士 2 年になっても知識が身につかず基礎的なことまで常に金丸先生や研究室の同輩に頼ることになっていました．そんな私でも丁寧にご指導くださったことに大変感謝しております．また，就職活動時が長引いてしまい，学会発表を交代することになってしまい，金丸先生ならびに中込恭平氏に多大なご迷惑をおかけしてしまいました．特に中込氏には発表者という大役を突然任せることになってしまっても快く引き受けていただき感謝しております．おかげさまで無事就職活動を終えることができました．

4 年間という長いようで短い時間でしたが人として学ぶべき点を多く見つけることができ成長できたと思っております．また，様々なイベントへの出展や出展者様がたとの交流など，普段ならとても経験できないようなことに参加させていただけたことも貴重な体験であり，勉強になりました．改めて金丸隆志准教授に心より感謝の意を表します．ありがとうございました．

そして 4 年間，いつも私を支えてくださった研究室の先輩，同輩，後輩たちに感謝します．とても周囲に恵まれた 4 年間でした．なかでも，学部からの友人であり，ともに学んだ中込恭平氏，塙暁成氏，山口和也氏にこの場で感謝の意を表します．

最後に，精神面，通学等様々な面から私を支え，応援してくださった両親，祖母に感謝し謝辞といたします．

I　　付録

ここでは付録として MPLAB の使用方法や，JNI 導入における環境設定や利用法を説明する．

I.I　　PIC の特徴と開発環境について

前章で述べた電圧変化を出力するために PIC を使用した．PIC は Microchip Technology 社製のマイクロコントローラと呼ばれる IC である．CPU や ROM 等が小さな IC に収められており小型である．またさまざまなピン数のものが発売されており目的に応じて選択できる．また動作電圧が 2.5V～6.0V と低く，消費電力が小さく電池で動作させることも可能である．これらの特徴から様々な周辺機器を搭載した PIC のマイコンボードも作成されている．画像処理や Linux などの OS を載せることのできるほどのパワーは持っていないが限られた範囲での使用では真価を発揮する．

また，PIC の開発には MPLAB IDE（Integrated Development Environment）というマイクロチップテクノロジ社が提供している無料の統合開発環境ソフトと PIC にプログラムを書き込む PICkit3 というライタが必要である．すべての PIC シリーズのプログラムがこの MPLAB IDE で開発できるようになっており図 I.1 [24]のようなプログラム群で構成されている．プロジェクトと呼ばれる開発単位ですべてが統合管理され，エディタやアセンブラ等が内蔵された開発環境である．

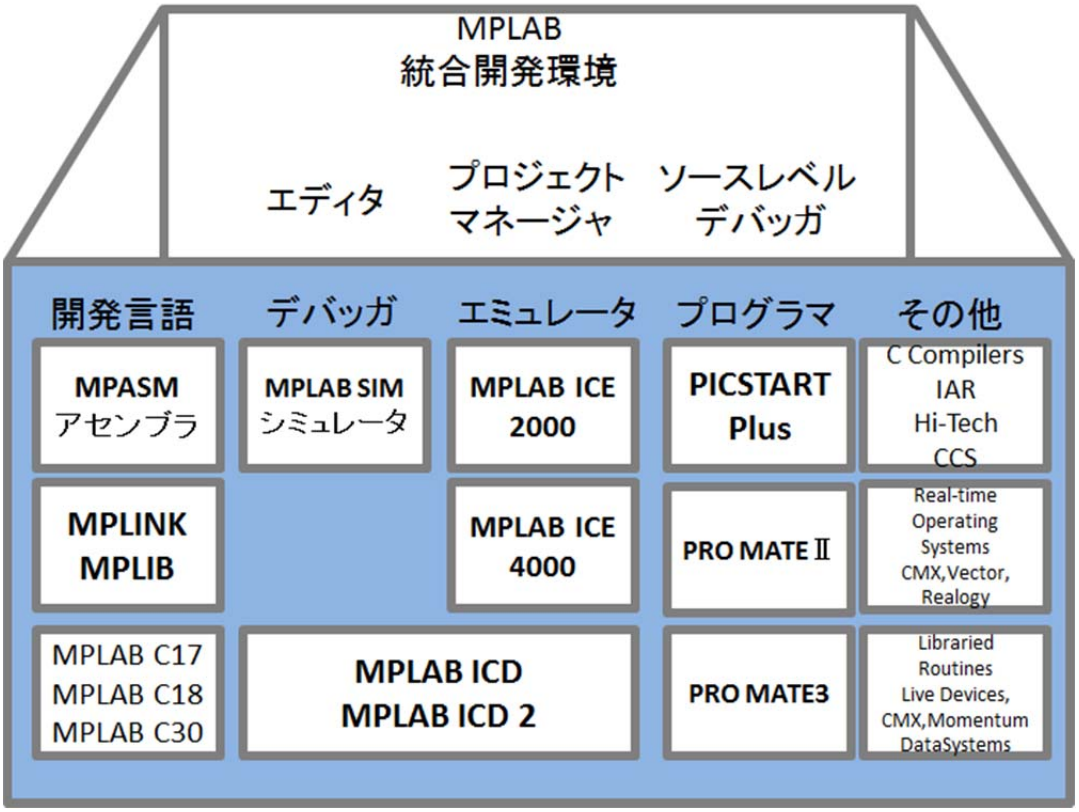

図 I.1　MPLAB の内部構成

PIC のプログラム開発はまずプロジェクトの作成から始める．これに合わせ，ディレクトリが生成されプログラムのファイルや自動生成されたファイルが格納される．

エディタはプログラムのソースファイルそのものを編集するテキストエディタである．シンタックス毎に色や字体が自動的に変化するようになっており，シミュレーションデバッグもこのエディタでソースレベルに落とし込むことができる．

MPLINK はリンカと呼ばれるものであり，アセンブラをリロケータブルな形式で複数分割して作成し，それらを一つに接続するとき，その接続を行うプログラムである．

MPLIB はライブラリアンと呼ばれ，作成したプログラムを共有，呼び出しするために保存しておくためのツールである．シミュレータはデバッグツールであり，パソコン上でPICをシミュレーションし動作を確認できるようにしたものである．多くのPIC内部情報を確認することや，疑似的な入力を行いながらプログラム動作を確認できる．

次にコンパイルしたソースコードを PIC に書き込むためのライタについて記述する．本研究で使用したライタは MICROCHIP 社製の PICkit3 というものである．これは MPLAB IDE の仕様を前提としたデバッガプログラマであり，各種リアルタイムデバッグが可能になる．また PICkit3 単体で過電圧，ショート検出機能や現在の状態（POWER，ACTIVE，STATUS）を示すアナログ LED が搭載されている．

図 I.2　PICkit3 の外観

PICkit3 を使用するには PICkit3 のアプリケーションソフトウェアをインストールする必要があるがインストール手順は割愛する

ここで MPLAB の使用法について記述する.

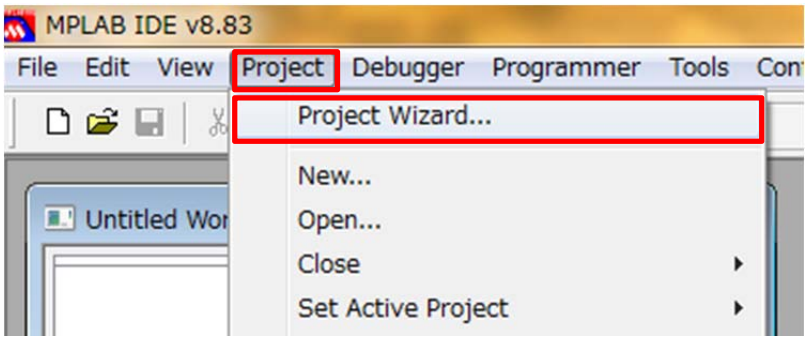

図 I.3　MPLAB での新規プロジェクト作成

まず MPLAB を起動すると図 I.3 にある画面上部のメニューバーから Project を選択し Project Wizard に進むと Project Wizard の設定が始まる.

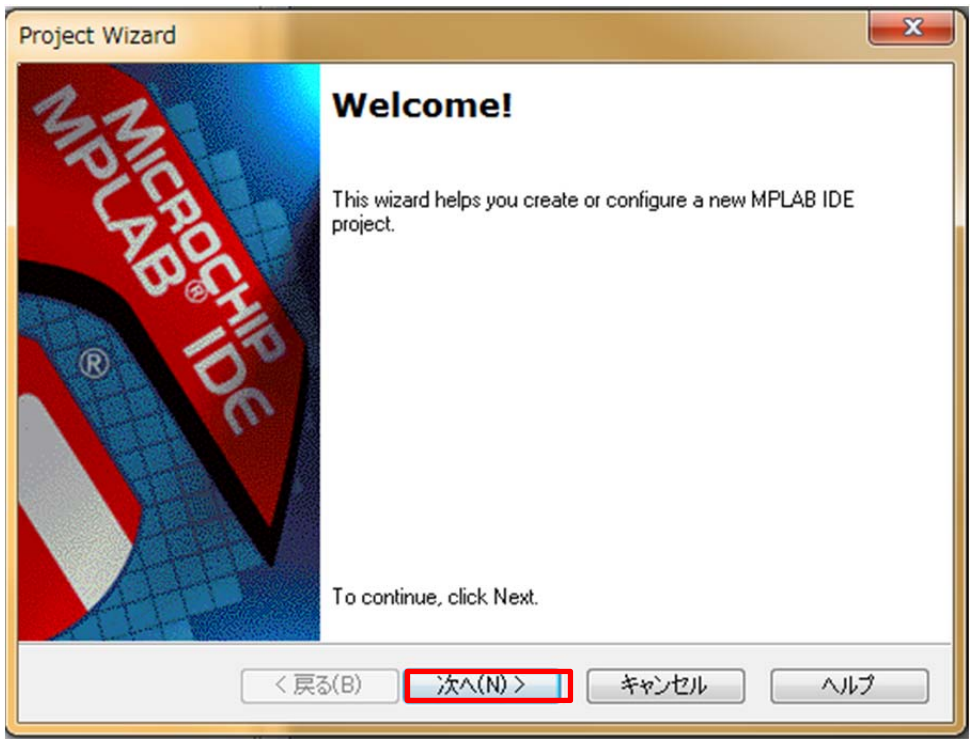

図 I.4　Project Wizard の設定

図 I.4 のウィザード画面が表示され　次へ　をクリックする.

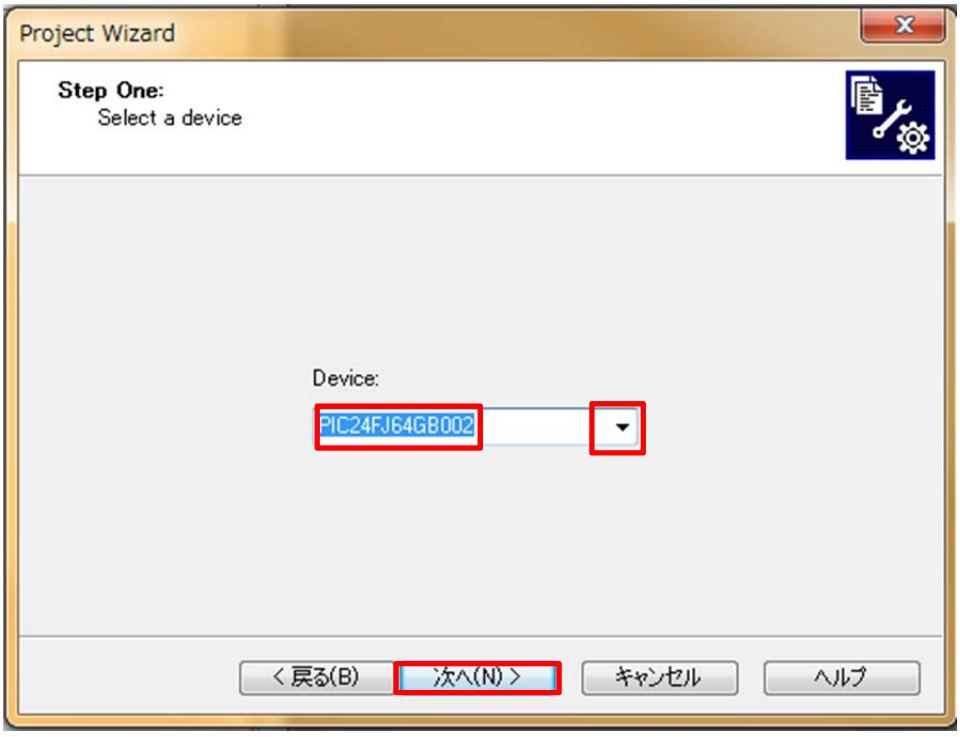

図 I.5　使用するデバイスの選択

図 I.5 では使用する対象となる PIC の種類を選択する．▼をクリックすると PIC のリストが表示されるので今回開発に使用する PIC24FJ64GB002 を選択し　次へ　をクリックする．

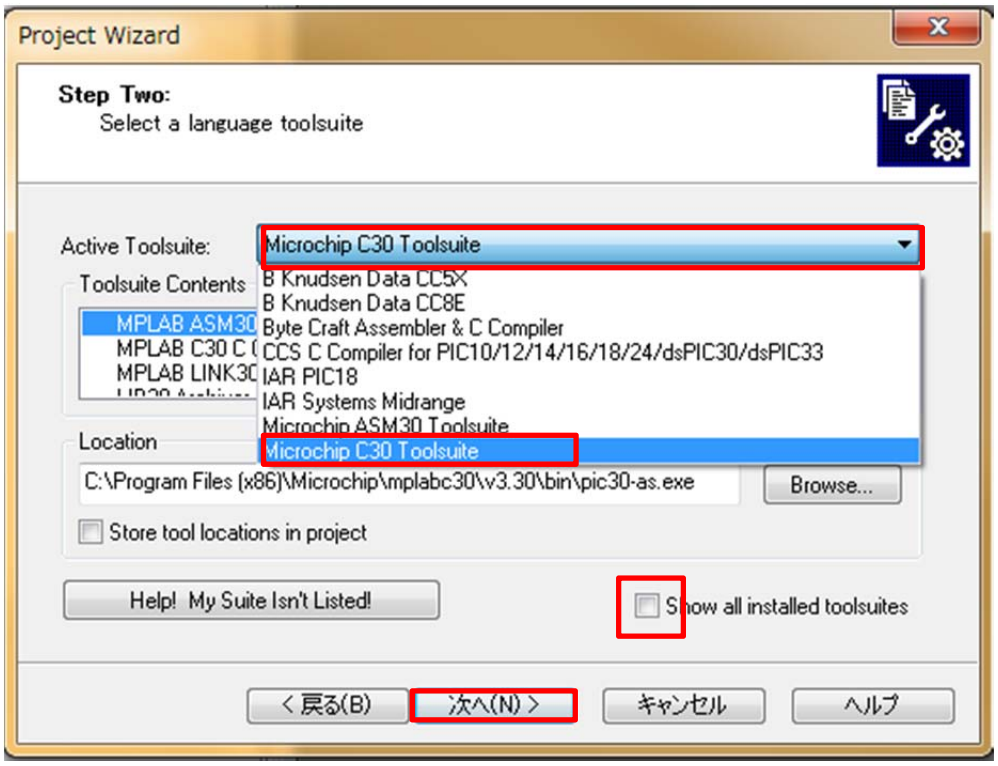

図 I.6　コンパイラ選択画面

図 I.6 で使用するコンパイラを選択する．▼で使用可能なコンパイラ一覧が表示されるので今回は Microchip C30 Toolsuite を選択し　次へ　をクリックする．もし使用したいコンパイラが表示されない場合は右下の Show all installed toolsuites のチェックボックスにチェックすることで表示されることがある．

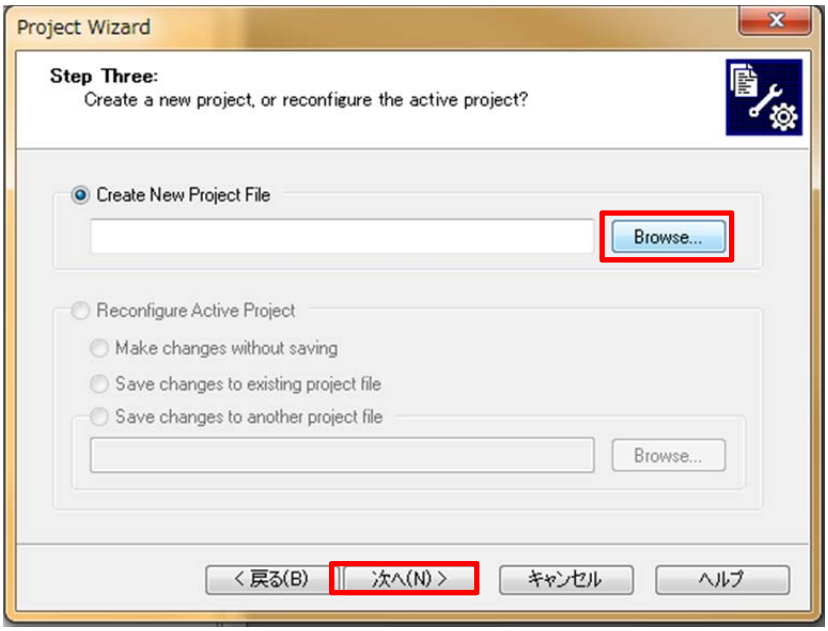

図 I.7　フォルダ選択画面

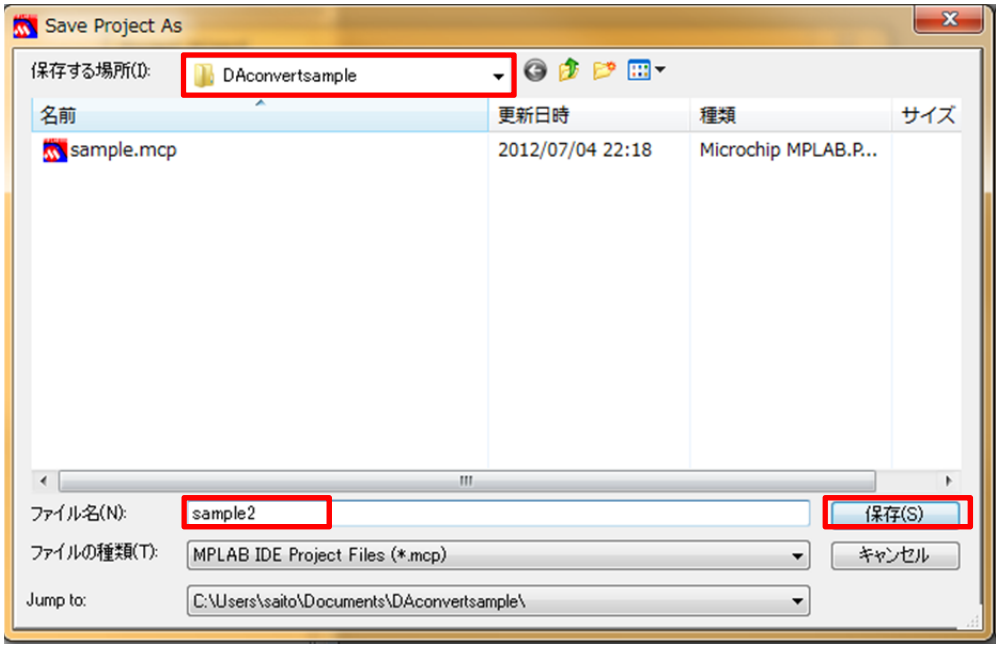

図 I.8　保存するフォルダの選択

図 I.7 でプロジェクトファイルを作成するフォルダを選択する．Browse をクリックすると図 I.8 のような画面が立ち上がる．ここで保存したいフォルダを選択しファイル名を記入し保存をクリックする．この時，フォルダ名やファイル名に漢字を使用すると MPLAB が認識できないので注意が必要である．

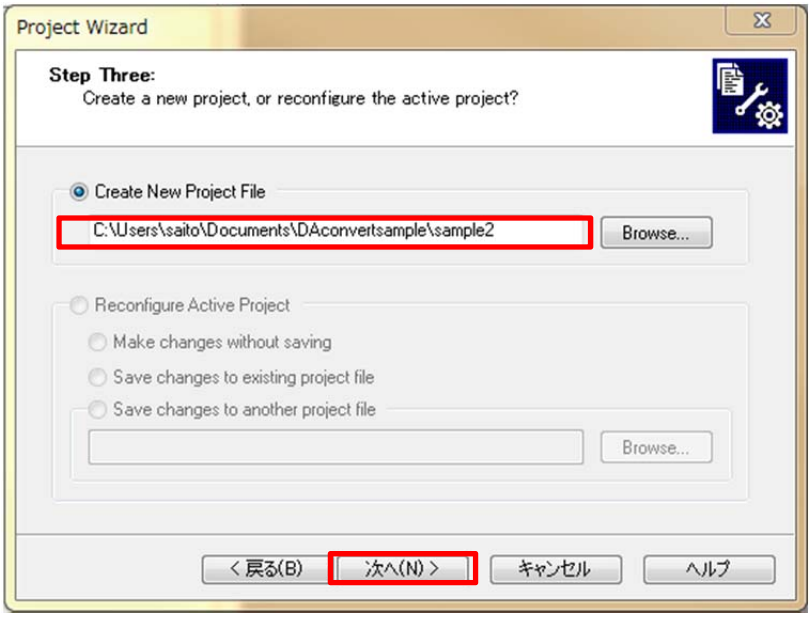

図 I.9　選択したフォルダの確認

図 I.9 は図 I.8 で選択したフォルダやファイル名が正しく選択されているかを確認し　次へ　をクリックする．

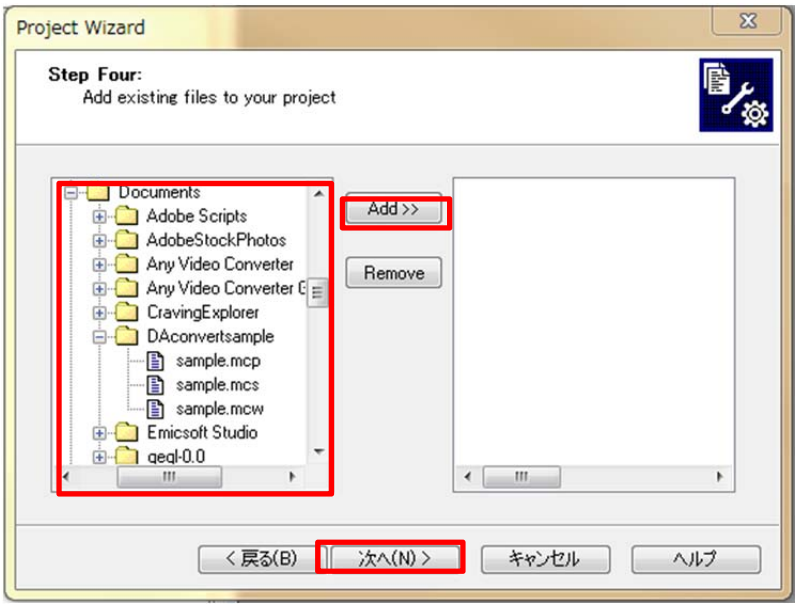

図 I.10　ソースファイル選択画面

図 I.10 では任意でプロジェクトファイルに加えたいソースプログラムを左側から選択し Add をクリックすることで追加できる．この操作は後でも可能である．特にないか追加が完了であれば　次へ　をクリックする．

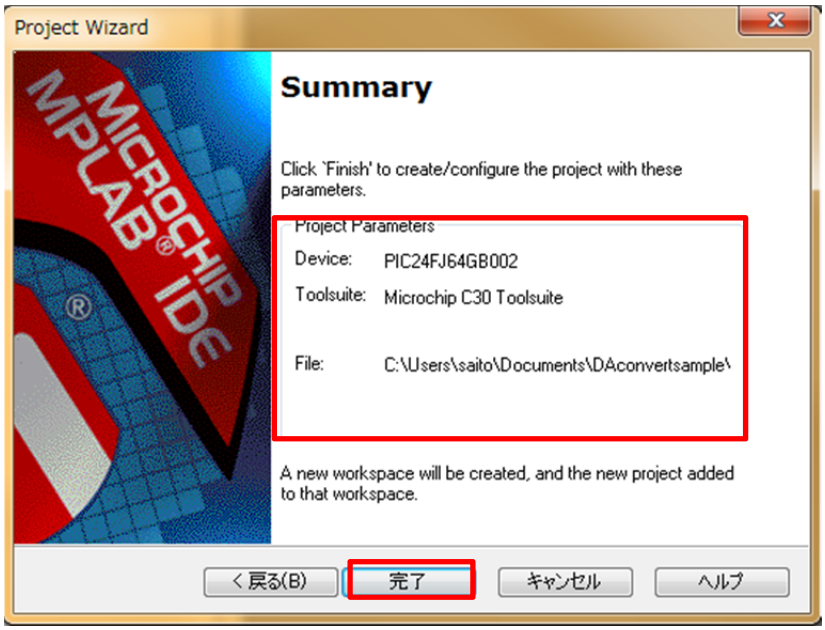

図 I.11　プロジェクトファイル設定の完了

ここまでの設定内容が上部に表示される．あっていれば　完了　をクリックすることで完了しプロジェクトが立ち上がる．

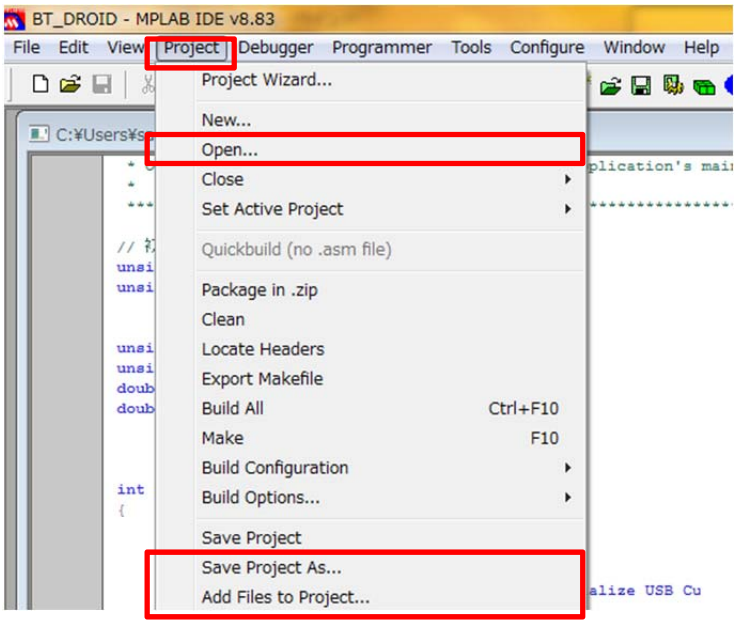

図 I.12　プロジェクトファイルを開くまたは保存

プロジェクトファイルを開く場合はMPLABのメニューバーからProjectをクリックしOpenから任意のフォルダを開く．保存する場合は同じくProjectからSave Projectをクリックする．

次に MPLAB 上で書いたプログラムソースをコンパイルする方法を示す．ソースを書き終えた段階で MPLAB メニューバーの Project から Build All をクリックする．

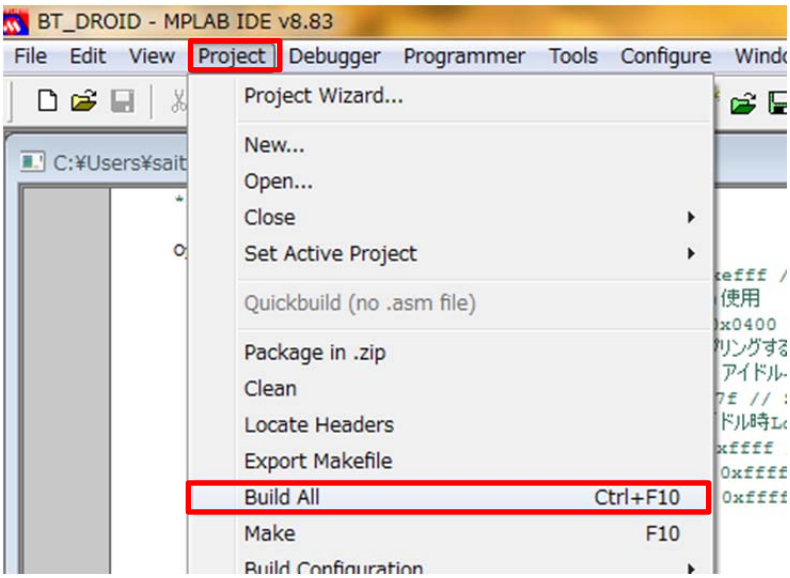

図 I.13　プログラムのコンパイル

ソースコード上で Ctrl+F10 のショートカットキーを押すことでもコンパイルが可能である．

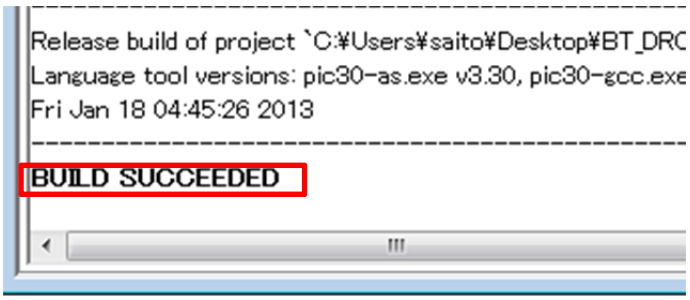

図 I.14　ビルドの成功

コンパイルが成功すると図 I.14 のように BUILD SUCCEEDED と表示される．

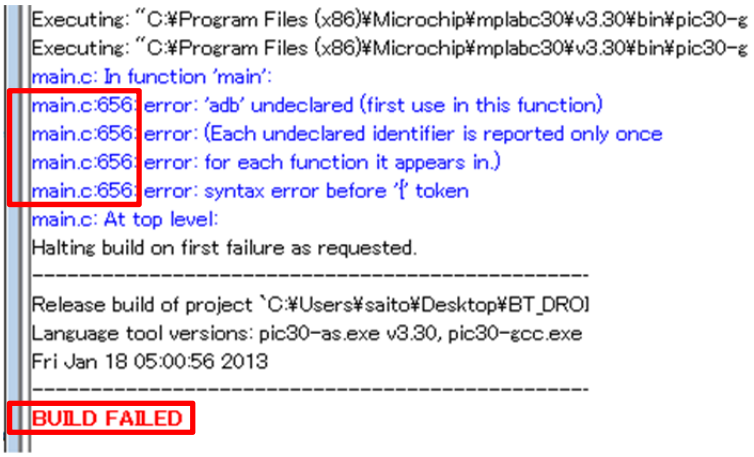

図 I.15　エラーの行数とビルド失敗

また，コンパイルに失敗すると図 I.15 のように BUILD FAILED という文字とともに上段青文字でプログラムソースのエラーが発生している行数が表示される．

ビルド完了後 MPLAB 側から PICkit3 をライタとして使用するための設定を行う．

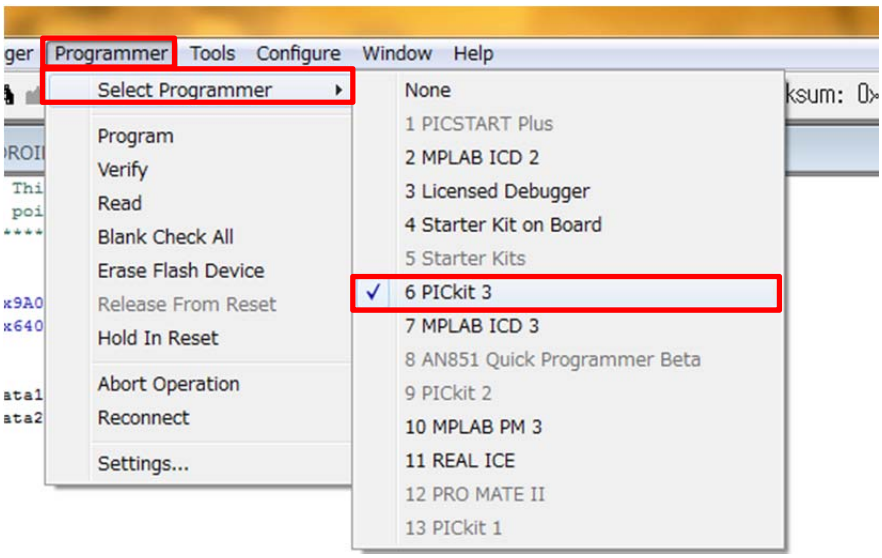

図 I.16　MPLAB 側の設定

MPLAB メニューバーの Programmer をクリックし，Select Programmer の一覧から PICkit3 を選択する．

これでライタの選択が終了しうまくいっているならば図 I.17 のように Output ウィンドウ上に Device ID Revision　=　XXXX　と表示される．XXXX の部分は使用する PIC により変化する．

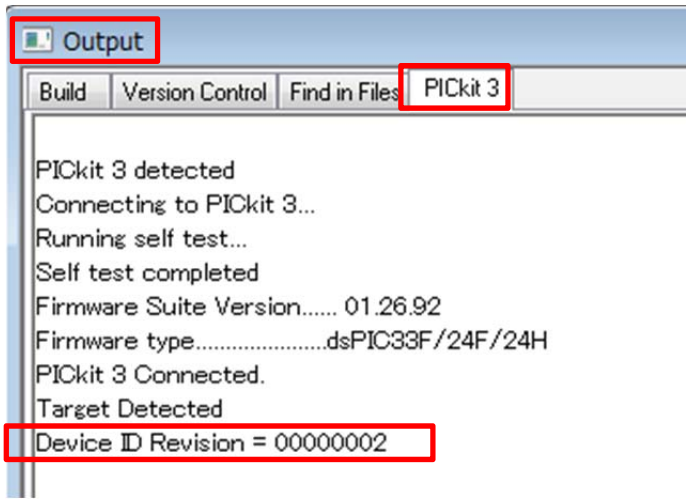

図 I.17　PICkit3 と使用する PIC の接続状態

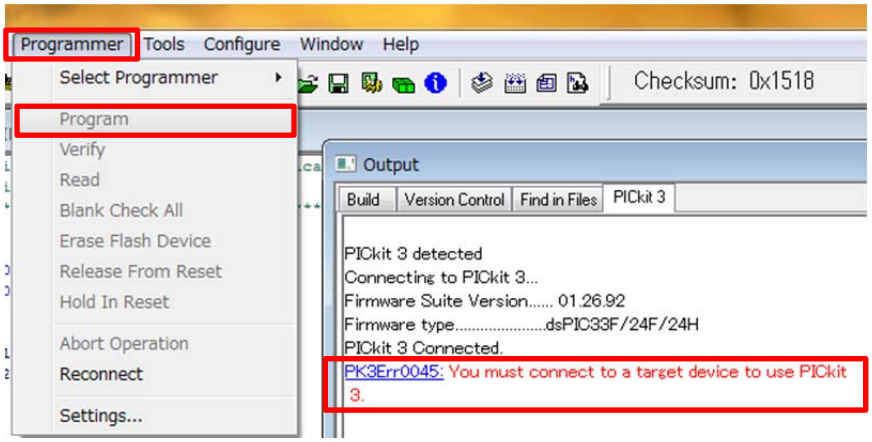

図 I.18　PICkit3 の接続状態 I

ビルドが成功し，PICkit3 が使用する PIC に接続している状態で書き込みを行う．書き込みは Programmer の Program をクリックするのだが，図 I.18 は PICkit3 が対象の PIC に接続されていない状態の時のものであるため Program がクリックできない．図 I.18 の Output ウィンドウに赤文字で You must connect a target device to use PICkit と表示されており，対象と PICkit3 が接続できていないことがわかる．

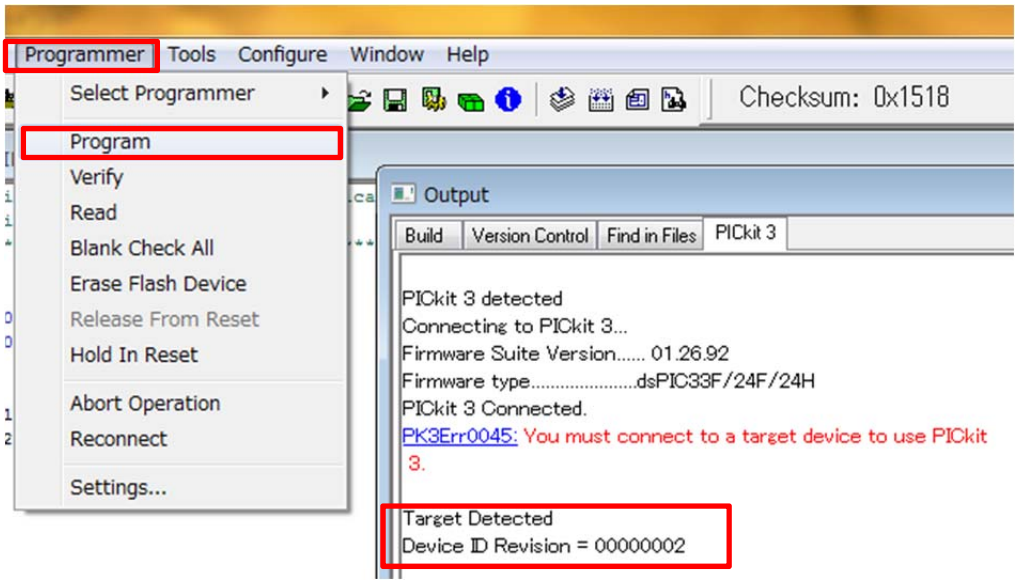

図 I.19　PICkit3 の接続状態Ⅱ

図 I.19 は PICkit3 が書き込む対象の PIC に接続さている状態である．Output ウィンドウ上では Target Detected と表示されており Programmer の Program もクリックできるようになっている．この状態で Programmer から Program をクリックすることで PIC にプログラムソースを書き込むことができる．

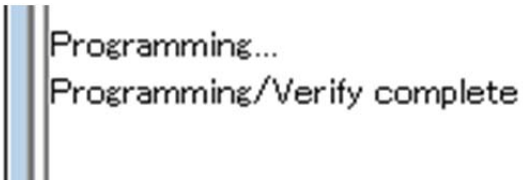

図 I.20　プログラム書き込み完了

図 I.20 はプログラムの書き込みが完了すると Output ウィンドウに表示される．書き込みが完了すれば PICkit3 は対象の PIC との接続を切っても問題ない．

参考文献

1. 内閣府　共生社会政策統括官．交通安全白書．共生社会政策．
http://www8.cao.go.jp/koutu/taisaku/h24kou_haku/pdf/zenbun/gen1_1_1_01.pdf.
2. −．道路交通環境の整備．交通安全白書．
http://www8.cao.go.jp/koutu/taisaku/h24kou_haku/pdf/zenbun/gen1_1_2_01.pdf.
3. 米グーグルの自律走行車，カリフォルニアでも公道走行が可能に. AFPBBNews.
http://www.afpbb.com/article/environment-science-it/science-technology/2903873/9584103.
4. グーグル，自律型自動車の制御で特許. CNET.
http://japan.cnet.com/news/business/35012071/.
5. 自動運転技術「SARTRE」のロードトレイン，行動にて初公開. VOLVO.
http://www.volvocars.com/jp/top/about/news-events/pages/default.aspx?itemid=102.
6. [CES 13]レクサス，自律走行車を公開.
http://response.jp/article/2013/01/08/188367.html.
7. TECHNOLOGY STORY. SUBAARU.
http://www.subaru.jp/about/technology/story/eyesight/eyesight01.html.
8. RC モデル・トップページ．株式会社タミヤ.
http://www.tamiya.com/japan/rc/index.html.
9. 株式会社タミヤ．タミヤ RC スタートガイド. TAMIYA INC.　株式会社タミヤ.
http://www.tamiya.com/japan/cms/rcstartguide.html.
10. RC カーのブレーキに関して. RCT 入門者フォーラム.
http://www.rct.jp/cgi/bbs/beginner/raib.cgi?md=tv&pn=99&ln=2161.
11. RT-ADK. RT CORPORATION.
http://rt-net.jp/product/rtadkseries/.
12. hrdakinori/BT_DROID.
https://github.com/hrdakinori/BT_DROID.
13. 横溝惇（2008）「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」『工学院大学修士論文』
14.「シリアル通信」とは？ 電子工作室.
http://www.picfun.com/cpu12.html.
15. 後閑哲也（2007）『C 言語ではじめる PIC24F 活用ガイドブック』技術評論社
16. m はげ. SPI の罠. PIC の罠.
http://www2.ocn.ne.jp/~mhage/PIC_Trap/SPI.html.
17. PIC 間　SPI 通信．ｙｓ電子工作ラボ.
http://www.ys-labo.com/pic/pic%20chips/pic%20chips%20contets/061009%20SPI%20html.html#1CHAR%20%20INT%20%20C30%20dsPIC.

18. TAMIYA. タミヤ RC システム　ファインスペック FM・電動 RC ドライブセット取扱説明書.
19. SHARP GP2Y0A02YK.
http://sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a02_e.pdf.
20. 高木佑介（2012）「AndroidOS とカメラによる対象物追跡における処理の高速化」『工学院大学修士論文』
21. neuralassembly. Android で WiFi から受け取った動画を画像処理する（1）ピクセルを直接操作する方法. neuralassembly のメモ.
http://neuralassembly.blogspot.jp/2012/12/androidwifi.html.
22. モータの PID 制御法. 電子工作室.
http://www.picfun.com/motor05.html.
23. 布留川英一（2012）『Android プログラミングバイブル』ソシム

図表目次

付録図表目次