研究テーマ

「ＲＡ４Ｍ１マイコンを使用したマイコンカーによる高速化と安定化の研究」

長野県駒ケ根工業高校
電気科　佃　洸希
情報技術科　畑　叶介

1. はじめに

私たちは駒ヶ根工業高校（以下本校）ロボット研究製作部マイコンカー班において、マイコンカーの製作に取り組んできた。マイコンカーとはマイコンを搭載したライントレーサーの一種で、コース上の白、灰色の線を読み取り走行する自立型ロボットである。

マイコンカーによる大会として「ジャパンマイコンカーラリー」がある。全国高等学校長会が主催し、今年度で 31 回目の開催となった。本校では平成 18 年度からこの大会に参加しており、平成 20 年度からは地区大会である北信越大会を勝ち抜き全国大会への出場を続けている。

以下の報告では、今年度行ってきた主要な研究、活動を通して私たちが感じたこと、学んできたことについて記す。

2. 研究動機・目的

RMC-RA4M1 マイコンに置き換えるにあたって、対応するモータドライブ基板の設計およびプログラムの開発が必要になり、研究を行った。目標としてジャパンマイコンカーラリー全国大会でベスト 16 位を目指した。

3. 研究内容

① 回路の設計（担当:畑）

今回基板設計にあたってロボテナが販売しているモータドライブ基板 type D と日立ドキュメントソリューションズが販売しているモータドライブ基板 Advanced Ver.1 を参考に基板設計を行った。まず 2 つのモータドライブ基板の回路図を比べた時、モータ部分の回路に違いがあった。モータドライブ基板 Advanced Ver.1 では、モータ部分の回路に N チャネル 2 個、P チャネル 2 個の MOSFET を使った H ブリッチ回路（図 1）が使用されている。しかし、P チャネル、N チャネルの 2 つを使用した H ブリッチ回路ではゲート駆動が比較的シンプルであるが、P チャネルは N チャネルに比べてゲート総電荷量が大きくスイッチング損失が大きくなってしまう。P チャネル、N チャネルのゲート総電荷量を調べたところ P チャネル（型番 PA2736GR）は 80nC、N チャネル（型番 RJK0354DSP）は 12nC だった。このように、P チャネルの MOSFET はゲート総電荷量が大きいため電荷をためる時間が長くなってしまい、動作が遅くなってしまう。そのため、今回のモータドライブ基板では、モータドライブ基板 type D で実装されている N チャネルの MOSFET を 4 つ使った H ブリッチ回路（図 2）を使用した。すべての MOSFET を N チャネルにすることで、MOSFET のゲートドライブは複雑になってしまうが、今回使用している N チャネル（型番 IRF8915PBF）のゲート総電荷量は 4.9nC であり、前述の MOSFET より総電荷量は小さい。そのため、電荷をためる時間が短くなるため高速スイッチングが可能になる。

また、今回新たに日立ドキュメントソリューションズから販売されているモータドライブ基板 AdvancedVer.1 に実装されている電源電圧検出回路や、フルカラーLED に加え、EEPROM を追加した。フルカラーLED には OSTA5131A を使用し、EEPROM には AT24CS64 を使用している。

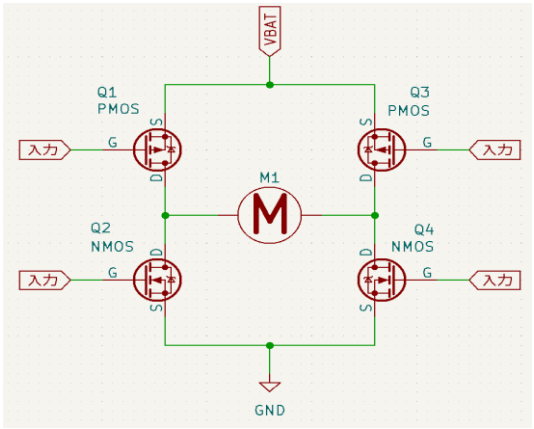

図 1　P・N チャネル H ブリッチ回路

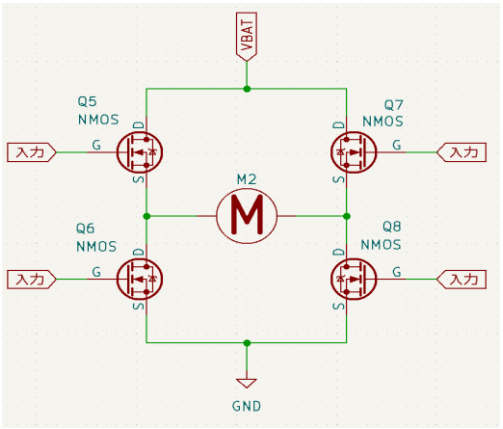

図 2　N チャネル H ブリッチ回路

② プリント基板の設計（担当:畑）

基板設計には kicad を使用した。回路設計の前にまず回路図エディターで使用する部品のシンボルを type D の部品表をもとに DigiKey などの電子部品販売サイトを利用しシンボルを探した。シンボルが見つからない場合は、既存のライブラリー内のシンボルを代用し、部品のシンボルをシンボルエディター内に保存した。集めたシンボルにフットプリントを関連づけた。フットプリントはプリント基板上に配置する部品の端子の形状・サイズ・位置などが定義されたデータである。この作業を行うことで、論理的な接続と物理的な形状が結びつけられ、PCB 設計での配線ミスの防止や未配線の検出を簡単に行うことができる。回路図の作成は、type D の回路図を元に回路図エディターに部品を配置し、配線を行なった。（図 3、図 4）

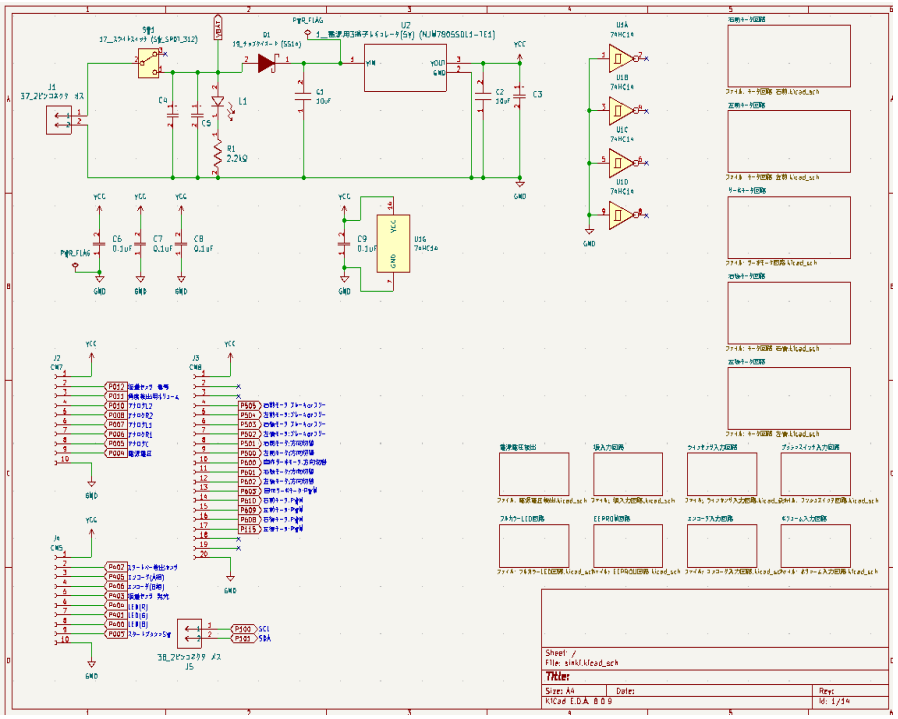

図 3　電源、コネクタ回路

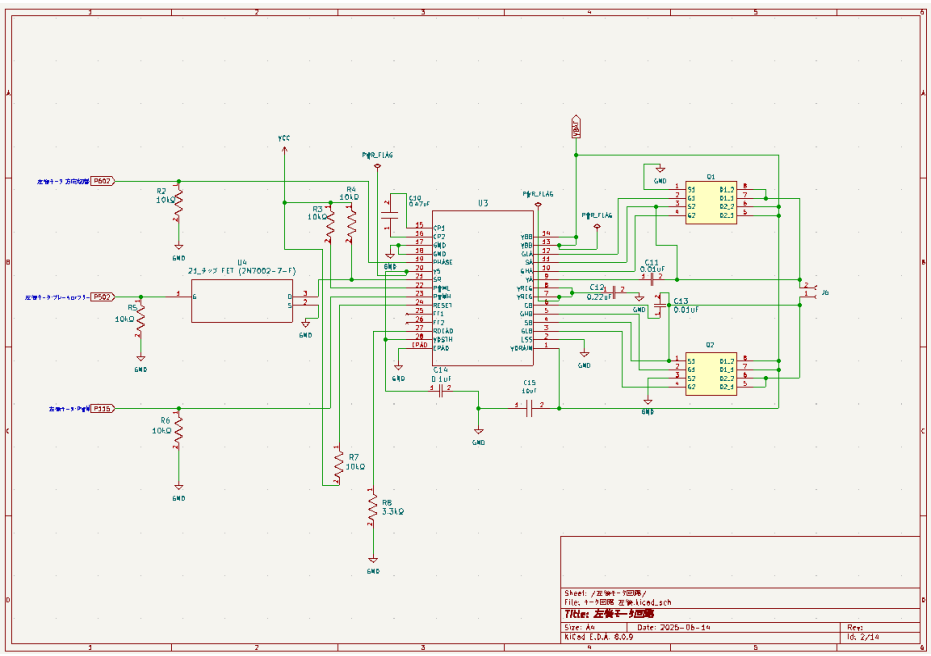

図 4　モータ回路

パターン設計は PCB エディターを使い設計を行なった。まず基板の外形線を作成した。外形線は type D とマイコンカーラリー販売のモータドライブ基板 Advanced Ver.1 の基板寸法を参考にして製作した。次に回路図から部品を更新することで回路図に対応するフットプリントが読み込まれ、部品の配置を行った。部品の配置は使いやすさや性能を決める上でとても重要な部分になるので配置するときは、電源部分やコネクタの位置、コンデンサの配置などに注意しながら配置を行った。RA4M1 マイコンのコネクタ配置は、kicad の中に RA4M1 マイコンの外形とコネクタの位置が書かれたテンプレートがあったのでテンプレートを配置し、コネクタの 1 番ピンの座標にコネクタを配置した。部品配置終了後配線作業に取り掛かった。配線はできるだけ線が長くならないように表面と裏面を使い配線を行なった。途中部品のパットや他の線があることによって配線が困難になる場面が何度かあった。その都度部品の配置や線の通す場所を変更しながら配線作業を進めた。(図 5) 完成した基板はコネクタの穴位置を確認するため基板加工機で削りだし試作品（図 6）を作った。

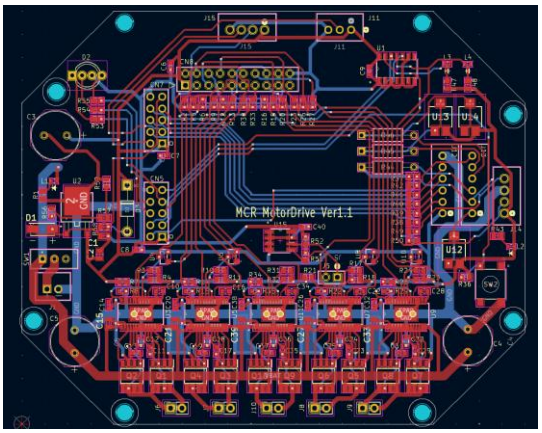

図 5　プリント基板のパターン図

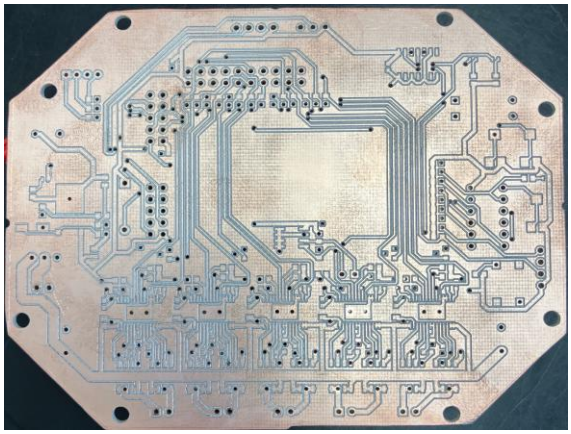

図 6　基板加工機での試作品

試作品でのコネクタの位置を確認後、回路図と見比べながら配線の確認を行なった。

③ プログラムの作成（担当:佃）

まず私たちは、走行する際に重要な、アナログセンサの値を読み取るためのプログラムから組んでいくことにした。前のプログラムでは getAnalogSensor という関数内に AD_L1,AD_R1 があり、それぞれ#define から ad0,ad4 と定義されている。それらはポート 7 から A/D 変換された値を受け取っている。仮にこれをそのままコピーしても、ポートの配置が完全に違う上に、RA 基板自体がアナログ専用ポートなども存在しているため、これらを考慮しながら置き換えを進めていった。まず使うポートに目星をつけたあとに、置き換えを進めた。マイコンカーラリー販売にある RMC-RA4M1 の実習マニュアル(参考文献.1)にあるセンサのポート(COM7)を参考に進めた。まず、私たちは Arduino 側で用意されている digitalRead、digitalWrite、analogRead、analogWrite という関数を使い、確実に値の読み取りができることを確認していきながら進めた。それぞれのセンサから読み取れることを確認し、次の工程へ移った。これらの関数を使用し、8 個ほど連続で読むようにすると、約 22[μs]程かかってしまうため、これらの関数を介さず RA マイコンのレジスタから直接読み書きできるプログラムを使用し、組み込んでいった。しかし、ここで思わぬ現象が発生してしまった。いざトレースを行ってみると、何も触れていないのに、急に発散するような挙動に遭遇した。これはトレースの強度をかなり低くしても同様な挙動をした。原因は恐らく、マイコンが前と比較して処理速度が高速化し、読み取る際に意図していない値を読み込んだ結果、ブレてしまうかもしれないと考えた。そこで、対策として、Arduino 側で既にある、何回か値を読み込んだ後に平均値をだし、その値を返すプログラムに変更したところ、発散する現象はおおよそ抑えることができた。

次に私たちは、モータ関係のプログラムを組んでいった。私たち Advanced クラスは、駆動輪とサーボの PWM の設定は 250[μs]で行っているが、提供されているプログラムの駆動輪制御はタイマ割り込み(1[ms])前提で組まれているため、特にトレース関係で走行に深刻な影響が出てしまうため、250[μs]に対応するように組んでいった。そのため、まずは割り込み内にあった駆動輪制御関係をすべて撤去した後に、1[ms]の PWM 周期の設定値である 47999 の 4 分の 1 の 17999（250[μs]）に変更した。PWM の初期設定が終わったら、モータ関数の制作に入った。前のプログラムでは速度に依存せずモータパワーの制御をする motor2 関数、ストレートやカーブの速度制御をおこなう motor3 関数、クランクやレーンチェンジの侵入の際に速度に応じたブレーキをかける motor6 関数があり、それぞれフロント_f とリア_r に分かれていて、それらを参考に組んでいった。まずは motor2 関数から行い、motor2 関数の引数は左の強さ、右の強さの順に設定し、引数をもとに計算した値を GTIOC4A、GTIOC4B, GTIOC5A, GTIOC5B に代入し、モータの正転、逆転は R_PORT6->PODR_b.PODR1、R_PORT6->PODR_b.PODR2、R_PORT5->PODR_b.PODR0、R_PORT5->PODR_b.PODR1 に正転０, 逆転 1 として代入した。それに加えてモータのフリーブレーキの関数を別個で追加した。フリーブレーキはフリー１、ブレーキ０として R_PORT5->PODR_b.PODR2、R_PORT5->PODR_b.PODR3、R_PORT5->PODR_b.PODR4、R_PORT5->PODR_b.PODR5 に代入し制御できるようにした。しかし、このまま 2 進数でも問題なく動作するが、わかりやすくするために F(1)、B(0)と define と定義し、モータ制御をできるようにした。動作確認を行った後に、motor3 関数に取り掛かり、基本的な部分は motor2 をベースに追加する形で組んでいった。速度制御を行うために、引数に制限速度を追加し、その値を超えると、引数のモータパワーに関係なく 0 にして速度制御を行えるようにした。motor3 関数の確認が終わったら、motor6

関数に取り掛かった。Motor6 関数は主にクランクなど速度をかなり落とさないといけない区間で扱うブレーキ制御で、motor3 関数をベースに制限速度を上回っていたら、-100～-50 でモータのブレーキ処理を行うようにした。

モータ関数を組んでいく過程で、vSpeed という走行プログラムとは別でモータ制御を行えるようにした。LCD でモードの制御とパワー制御を行えるようにし、4 輪個別でパワー制御を行える処理と、フロントとリアでパワー制御を行える処理、駆動輪のパワー制御に加えて、モード切り替え時の角度を 0 として、LCD で指定した切り角をもとにサーボモータを制御するプログラムをそれぞれ追加した。

④ カーブの進入速度制御（担当:佃）

私は、置き換えの後に一通りの動作確認を行った後は、新しい駒高プログラムとして、8 月あたりから、マシンの調整を進めていった。まず私はカーブの調整を行った。カーブには半径が 450[mm]のカーブと 600[mm]のカーブがあり、450 では、侵入時に約秒速 3.9[m/s]まで減速できれば、約 3.6[m/s]後半で切り角は約 27 度で安定して走行し、600 では侵入約 4.1[m/s]で約 4.0[m/s]ほどで切り角は 22 度ほどだった。しかし、問題点が一つあった。それは、侵入時減速する際に、450 基準の場合、丁度よい塩梅で減速し走行することができたが、600 に侵入する場合、過剰に減速してしまって走行に支障が出てしまった。その上、450 を走行するときと、600 を走行するときのパワーバランスもかなり異なっていたため、私はどうにか侵入する際だけでも 450,600 の判断を行い、ブレーキを切り上げるタイミングとパワーの調整を行えないかと考えた。そこで私は、iAngle という値に着目した。iAngle とは、切り角の変化量のことで、この値が大きいほど短時間のうちに大きく切っているということがわかり、逆に小さいほどあまり切るスピードが速くないことがわかる。これを用いて、カーブの侵入時に 450,600 の判断を行った。私のマシンは、ストレートから約 4.6[m/s]程で侵入すると、450 の場合の iAngle の値は、15 以上になり、600 の場合は、12～8 の間程になった。これを基準に私はプログラムに組み込んだが、一つ問題が発生した。それは、低速で 450 に侵入した場合、iAngle も穏やかに増加するので、減速が足りずにアンダーステアになってしまうことがあった。そこの対策として閾値の条件に加える形で、閾値以上の値の時の iAngle の和を使うことでこの問題を解決しようとした。450 と 600 を比較して 450 の方が小さい訳だから、閾値以上の和を使うことで、より具体的に差別化ができるのではないかと考え、変更してみたら前よりは誤検知の回数は確実に少なくなったが、完全に潰すことはできなかった。しかし、この変更のおかけで、不必要な減速を削ることができ、安定して走ることができた。

4. 考察・まとめ

［畑　叶介］

RA4M1 にマイコンを変更することになり対応するモータドライブ基板製作を通して、まず部品探しから始まった。部品探しでは、過去のモータドライブの部品表からネットで探し、中には見つからない物もいくつかあった。代わりになる部品を探し、それでもない場合は自分で作成するなど部品探しの段階でいろいろと苦戦した。回路図設計では、既に存在する回路を利用して作成したが、kicad の使用などでエラーや警告をなくすのに苦戦した。プリント基板設計では、部品の配置から始まり、配線作業時になるべく線が長くならないように部品を配置するよ

うにした。だが、実際配線を行うと違う部品の線同士が交差してしまいそのたびに部品の配置を考え直したり、線の通す位置を変えたりし、何度かやり直すことが多かった。しかし、何度もやり直したことによって段々と線を通す位置を自分の中でイメージしながら部品を配置できるようになり配線作業をスムーズに進めることができた。なんとか基板を完成させることができた。だが、先輩に完成した基板に部品をはんだ付けしたもらい動作確認を行ってもらったところ、RA4M1 マイコンの CN5 と CN7 のコネクタの配置がモータドライブ基板側で逆に配置していたことが分かった。プリント基板設計が終わった後に最終確認を行っていたが、配線のみの確認だったためコネクタ番号など確認がおろそかになっていたことが原因だと思われる。コネクタの配置、配線をやり直し、基板を作り直した。部品をはんだ付けし動作確認を行ったところ、センサ、モータなど正常に動き、基板を完成させることができた。

大会には 11 月に行われた北信越大会で記録を残すことができた。予選タイムが 13.13 [s]、決勝タイムが 13.15 [s] だった。秒速に直すと 4.1[m/s]、4.09[m/s]だった。目標にしていた 4.2[m/s]に届かず、4.1[m/s]がやっと出せたという結果で終わった。最終順位は 12 位だった。先輩に決勝タイムで 0.04[s]届かず全国大会に出場することができなかったが、RA4M1 マイコンに乗せ換えての初出場にしてはわりと良い成績を残すことができたと思う。

5. 活動を通して

［畑　叶介］

今回 RA4M1 マイコンに置き換えるということで私はモータドライブ基板の製作を行った。基板製作を通して私は基板を作る大変さを理解することができた。限られた面積の中に効率のよい部品配置を考え、配置が終わると配線作業を行い、線の長さや、大電流が流れるところは線を太くし熱がたまらないようにするなど考えなら配線を行った。モータ部分の回路は特に熱がたまりやすいのでベタを広げ、ビアを打って熱をなるべく逃がすように工夫した。基板設計はとても大変だが、自分が作った基板が完成し、始めて電源を入れうまく動作したときの達成感を感じることができた。また、基板製作を行っている中で今回はモータドライブ基板だけではなく LCD の作成も同時進行で作成していた。そのため、LCD をマイコンに付ける向きとマイコンをモータドライブ基板に付ける向きがある。基板製作を行っている中でその向きの関係を部員同士で話会うことを怠った結果、モータドライブ基板側のマイコンの向きと部員が作成したマイコンに付ける LCD の向きが逆向きになってしまった。普段の活動だけではなく、自分一人で行っていることではないので仲間と協力して活動を行っているからこそコミュニケーションがとても大切だということを改めて学んだ。

今回無事基板を完成させることができたのは先生方、先輩、家族、部員の支えがあったからだと思う。この部活に入っていたからこそ本格的な基板製作を一から行うことができとても貴重な経験になった。残り 1 年の活動でもっといろんな経験を通して得られるスキルを増やしていきたいと思う。

6. 参考文献

(1),RMC-RA4M1 の実習マニュアル:

https://j-mcr.net/download/rmc_ra4m1_rev20_zisyu.pdf

(2),Kicad Basics for 8.0 ―kicad8.0 入門実習テキスト―