メカナムホイールにおける運動方程式と応用

5207D 菅谷　光慶（Mitsuyoshi SUGAYA）

概要　RCJ レスキューラインには正確なライントレースを行うことが求められている。そこで全方向移動が可能なメカナムホイールを使用するためのアルゴリズムを開発し，運動方程式を算出することが必要になった。運動方程式を算出することは比較的容易であるが，角度走行や超信地旋回，角度走行と超信地旋回の並列制御を行う時に必要な各パラメーターの算出には多くの試行を要し，メカナムホイールの制御関数に合致したライン検出関数も開発することも可能になった。この研究により，メカナムホイールを柔軟に制御することによって正確なライントレースを行うことができた。また，メカナムホイールはライントレースをだけではなく福祉機器や狭い空間における搬送システムとしての活用が期待できると考える。

キーワード：　メカナムホイール

1. はじめに

1.1 開発構想

これまでロボカップジュニア（RCJ）レスキューラインに様々な移動機構を搭載したロボットで出場してきたが，どの移動機構も複雑なライントレースや被災者救助，障害物回避などを完璧に遂行することができなかった。なぜなら，いわゆる一般的な車輪を使って細かく車体の移動方向を変えることは構造上困難なためである。以上のような問題点を解決するための移動方法を模索した結果，移動機構として今までの制御機構よりも優れている全方向移動が可能なメカナムホイールを使ったロボットの開発構想が提案され，本作を製作した。このロボットは RCJ レスキューラインに特化した構造になっているが，「福祉機器」や「狭い空間における搬送システム」としての活用が期待される。

1.2 RCJ レスキューラインの概要

レスキューラインとは，「そこは，人が被災者にたどりつくにはあまりにも危険な場所である。あなたのチームは難しい課題を与えられた。あなたのチームのロボットは人の補助無く，完全な自律モードで被災者救出作戦を実行できなければならない。そのロボットは丘やでこぼこした地面や瓦礫の上などの危険な場所で動き続けられるよう，十分知性的で耐久性が有るものでなければならない。また，ロボットは経路に置かれているレスキューキットを探し，回収し，生きている被災者に届けることも必須である。ロボットは被災者を見つけたら，人に引き継ぐ為，安全な避難場所へ穏やかかつ慎重に被災者とレスキューキットを運ばなければならない。ロボットは被災者の救助が完了したら避難ゾーンを出て，災害現場を脱するまで現場でのミッションを継続しなければならない。時間と技術力を結集し，もっとも成功したレスキューチームとなるよう準備を始めよう。」1）というシナリオに基づいた競技コート（図 1）上を自立制御ロボットが走行しライントレースや障害物の回避，被災者の救助などの動作の正確性や速さを競う競技である。

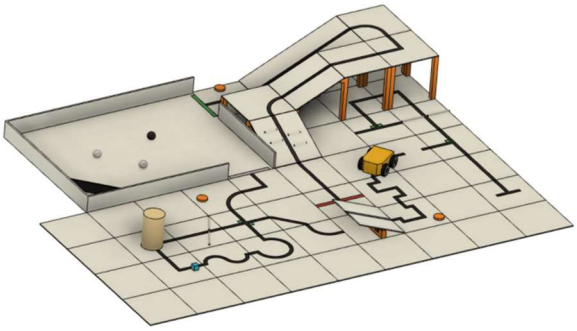

図 1　レスキューライン競技コート

2. メカナムホイールの概要

2.1 全方位移動機構

現在，二足歩行ロボットなどの開発のほか，工場，病院，飲食店などで働く各種の移動ロボットの開発が盛んである。こうした作業を行うロボットには衝突回避や協調動作など，従来よりも優れた運動性能が必要とされている。そのため狭い床面上を切り返し運動によらず任意の方向に自由自在に動き回ることができ，目的の場所に正確に移動することができるロボットの開発が求められる。このような屋内における平面上の運動性能を実現するには車輪型の全方向移動方式が有効である。全方向移動車の機構として各種の方式が開発されているが，実証

を行うロボットはRoboCupJunior Rescue Line の大会に出場するために制作されているため，Rescue Line の競技性において一般的なホイールと比べて運動性能が優れていると考えることができるメカナムホイールを採用することにした。ここで言う全方向移動車とは前後，左右，斜め，回転，などが自由にできる特殊な車両のことである。

2.2 メカナムホイールを用いた全方向移動の原理

メカナムホイール方式は 1975 年頃 Swedish 社によって開発された方式で図 2-1，2-2 のように車輪の外周の車軸に対して 45°傾けてビア樽型の複数のフリーローラーを取り付けたものである。各フリーローラーは車軸方向に投影すると，オーバーラップし完全に円形となる。これを車体に四輪取り付けたものを用いる。

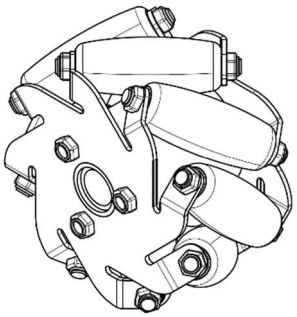

図 2-1

図 2-2

四輪すべてを正転（前進方向）させたとき，ローラーが 45 度傾斜して接地しているため，前輪は「ハ」の字，後輪は「V」の字に配置される。前輪では内側の推力が打ち消され，後輪では外側の推力が打ち消される。前後輪とも前進方向の推力が残り，前に進むことができる。横方向移動は，斜向かいの車輪を対として一方を正転，もう片方を逆転させると推力のベクトルは左右方向になる。前後輪の右側および左側の車輪を対として正転と逆転をさせると右あるいは左に超信地旋回する。超信地旋回とは油圧ショベルや戦車のように履帯（クローラー）をもつ車両が，左右の履帯を互いに逆方向に等速回転させることによって，車体の中心を軸としてその場で旋回することを示すが，メカナムホイールにおいても疑似的に可能である。すなわち各ホイールの回転速度と方向を変化させることにより全方向移動が可能となる。また，全方向移動の制御と超信地旋回の制御を組み合わせることによって，前進しながら車体の向きを変更するといった移動が可能である。

2.3 メカナムホイールの運動方程式

図 3 はメカナムホイールとフリーローラーの接地状態を示す。

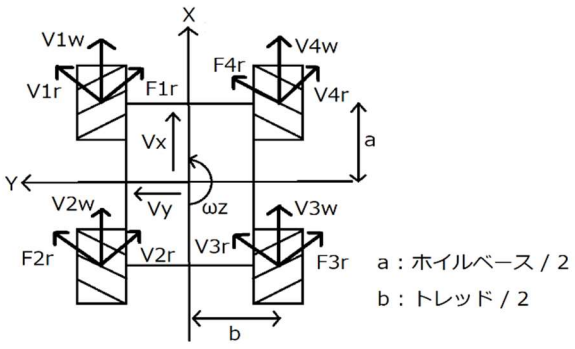

図 3　フリーローラーとメカナムホイールの配置

Viw（i = 1，2，3，4;ホイール番号）はホイール回転における速度ベクトル，Rw はホイール半径，φiw はホイールの回転速度，Vir（i = 1，2，3，4）は床に接触しているフリーローラーの接線方向の速度ベクトル，Fir（i = 1，2，3，4）は床から受ける反力，Vix，Viy（i = 1，2，3，4）は各ホイールの XY 方向の速度成分，Vx，Vy は車体本体の X 成分，Y 成分，ωz は車体の超信地旋回角速度とすると，Viw = Rw × φiw と表すことができる。これにより，Vix，Viy は Vx，Vy と ωz で表現できる。また，フリーローラーの取付角を 45 度とする。

各ホイールの X，Y 方向の速度成分は

$$V_{1x} = V_{1w} + \frac{V_{1r}}{\sqrt{2}},\ V_{1y} = \frac{V_{1r}}{\sqrt{2}}$$

$$V_{2x} = V_{2w} + \frac{V_{2r}}{\sqrt{2}},\ V_{2y} = \frac{-V_{2r}}{\sqrt{2}}$$

$$V_{3x} = V_{3w} + \frac{V_{3r}}{\sqrt{2}},\ V_{3y} = \frac{V_{3r}}{\sqrt{2}}$$

$$V_{4x} = V_{4w} + \frac{V_{4r}}{\sqrt{2}},\ V_{4y} = \frac{-V_{4r}}{\sqrt{2}}$$

式（1-1)

$$V_{1x} = V_x - b_1 \times \omega_z,\ V_{1y} = V_y + a_1 \times \omega_z$$
$$V_{2x} = V_x - b_2 \times \omega_z,\ V_{2y} = V_y + a_2 \times \omega_z$$
$$V_{3x} = V_x - b_3 \times \omega_z,\ V_{3y} = V_y + a_3 \times \omega_z$$
$$V_{4x} = V_x - b_4 \times \omega_z,\ V_{4y} = V_y + a_4 \times \omega_z$$

式（1-2）

式（1-1）(1-2)より，各ホイール速度と本体の速度成分との関係は

$$V_{1w} = V_x - V_y - (a_1 + b_1)\omega_z$$
$$V_{2w} = V_x + V_y - (a_2 + b_2)\omega_z$$
$$V_{3w} = V_x - V_y + (a_3 + b_3)\omega_z$$
$$V_{4w} = V_x + V_y + (a_4 + b_4)\omega_z$$

式（2）

ホイールの回転速度は

$$\begin{bmatrix} V_{1w} \\ V_{2w} \\ V_{3w} \\ V_{4w} \end{bmatrix} = \begin{bmatrix} R_w \times \varphi_{1w} \\ R_w \times \varphi_{2w} \\ R_w \times \varphi_{3w} \\ R_w \times \varphi_{4w} \end{bmatrix} = \begin{bmatrix} 1 & -1 & -(a+b) \\ 1 & 1 & -(a+b) \\ 1 & -1 & a+b \\ 1 & 1 & a+b \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \omega_z \end{bmatrix}$$

式（3）

すなわち，本体の移動方向の速度成分を与えることにより，そのとき必要な各ホイールの速度を得られる。

$$\begin{bmatrix} \varphi_{1w} \\ \varphi_{2w} \\ \varphi_{3w} \\ \varphi_{4w} \end{bmatrix} = \frac{1}{R_w} \begin{bmatrix} 1 & -1 & -(a+b) \\ 1 & 1 & -(a+b) \\ 1 & -1 & a+b \\ 1 & 1 & a+b \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \omega_z \end{bmatrix}$$

式（4）

また，式（4）の 4 個の解はあくまでもモーターの速度の比であり，速度そのものではない。そこで，速度係数を Pd（Pd = デューティー比 × 90）とする。これによりモーター本来の速度が得られる。

$$\begin{bmatrix} \varphi_{1w} \\ \varphi_{2w} \\ \varphi_{3w} \\ \varphi_{4w} \end{bmatrix} = \frac{P_d}{R_w} \begin{bmatrix} 1 & -1 & -(a+b) \\ 1 & 1 & -(a+b) \\ 1 & -1 & a+b \\ 1 & 1 & a+b \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \omega_z \end{bmatrix}$$

式（5）

しかし，各モーターの回転速度はすべて均一ではなく個体差があるため修正する必要がある。4 個のモーターの回転速度の比を Pi（i = 1，2，3，4）とする。よって，モーターの個体差を考慮した各ホイールの速度は

$$\begin{bmatrix} \varphi_{1w} \\ \varphi_{2w} \\ \varphi_{3w} \\ \varphi_{4w} \end{bmatrix} = \frac{P_d}{R_w} \begin{bmatrix} P_1 & 0 & 0 & 0 \\ 0 & P_2 & 0 & 0 \\ 0 & 0 & P_3 & 0 \\ 0 & 0 & 0 & P_4 \end{bmatrix} \begin{bmatrix} 1 & -1 & -(a+b) \\ 1 & 1 & -(a+b) \\ 1 & -1 & a+b \\ 1 & 1 & a+b \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \omega_z \end{bmatrix}$$

式（6）

3. 装置の概要

3.1 ハードウェア設計

3.1.1 ハードウェア設計の概要

このロボットは RoboCup Asia-Pacific 2021（RCAP 2021）の RCJ レスキューラインに出場するために制作されたものである。このロボットのハードウェア設計は，「大きな衝撃を加えても壊れず，メンテナンスがしやすく，軽い」ということを目標にして設計している。しかし一つ目と三つ目の目標は矛盾しているため，ロボットの重量を支える構造の素材を模索する必要があり，今回は強化プラスチックの一種である炭素繊維強化プラスチック（Carbon Fiber Reinforced Plastics）を採用した。炭素繊維強化プラスチックとは，FRP の中でも，炭素繊維（カーボンファイバー）を強化材として加えたものであり，高剛性，高強度といった特徴以外に「導電性・耐熱性・低熱膨張率・自己潤滑性・X 線透過性」といった特徴も兼ね備えており，それらの特徴を生かすことで軽量化・大型化・小型化・省エネ化などが期待でき，様々な用途で幅広く使われている。

また，このロボットの主要な部品は 3DCad で設計し 3D プリンターで製造されている。3D プリンターで製作した部品の材質は PETG-CF である。PETG-CF とはグリコール変性ポリエチレンテレフタレートに炭素繊維を混ぜたものである。PETG-CF は一般的な 3D プリンターのフィラメントの材質である ABS よりも強度が高いため，モーターマウントなどの負荷がかかる部品の強度を高めることができる。

全体概要を図 4 に示す。

図 4　ロボットの外観

主な仕様

ホイール	外径　60 ㎜　幅　30 ㎜
車体寸法	
車体重量	2。3 ㎏（バッテリーを含む）
モーター	Pololu 195:1 Metal Gearmotor 20D（4 個）
	Pololu 99:1 Metal Gearmotor 25D（2 個）
	Pololu 180:1 Mini Plastic Gearmotor（2 個）
	pololu 250:1 Micro Metal Gearmotor（1 個）
マイコン	Arduino Due（AT91SAM3X8E）
	Arduino Nano（ATmega328p-aur）
	M5Stack Core2（ESP32D0WDQ6）
センサー	VL53L0X（レーザー距離センサ：8 個）
	MPU6886（ジャイロセンサ：1 個）
	Pixy2（カメラ：1 個）
	NJL7502L（フォトトランジスタ：25 個）
電池	XPOWER R-SPEC Li-Po 7.4V 3800mAh

3.1.2 レスキューメカニズム

このロボットでは被災者の救助のために 5 個のモーターを使用している。

被災者の救助の方法は，第 1 に，図 5-1，図 5-2，図 5-3，図 5-4 の順番でアームを展開する。第 2 に，アームを展開した後，ロボットは距離センサを用いて壁との距離を計測し，被災者を壁に押し当てて被災者を籠の中に押し入れる。第 3 に，被災者を回収した後アームは第 1 と逆の動きをして元の位置に移動する。第 4 に，ロボットは他の被災者の救助に向かう。アームを元に戻すときのアームの位置，角度などのデータは常にジャイロセンサで

読み取り監視しているため，アームについている籠は被災者を落とさないように常に地面と平行になるようにプログラムされている。被災者をリリースするときはアームの位置を図 5-3 の状態に移行した後，籠を傾けることで被災者を避難ゾーンに入れる。またロボットは最大 3 人まで被災者を運ぶことができる。このようにして被災者を短時間で効率的に避難させることができる。

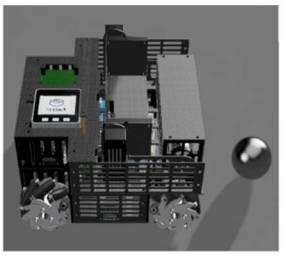

図 5-1

図 5-2

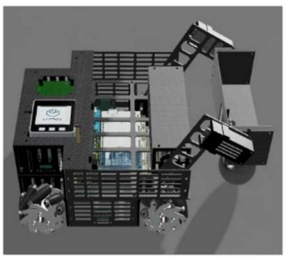

図 5-3

図 5-4

3.2 一般的なソフトウェアアーキテクチャ

3.2.1 ソフトウェアアーキテクチャの概要

このロボットに搭載されているソフトウェアは主に 3 つの部分に分かれている。

第 1 にライントレースのプログラムである。ライントレースとは，床面に描いたラインをロボットがセンサを利用して読み取り，ラインに沿って走行することである。これまで，ラインの情報を読み取るためにラインセンサとカラーセンサを使用していたが，このロボットにはカメラを搭載しているため，ラインセンサ及びカメラのデータを同時に処理できるようになっており，より正確なラインの情報を得ることができるようになっている。

第 2 にレスキューキットを回収するためのプログラムである。レスキューキットとは，50g 以下の 3cm×3cm×3cm の青く軽いブロックである。レスキューキットの判別にはカメラを使い，回収にはジャイロセンサを用いてアームの角度を常時監視しながら慎重に回収することができる。

第 3 に避難者の救助プログラムである。レスキューゾーンでは 8 個のレーザー距離センサとカメラが常時動いており，ロボットの位置を常に把握している。また 3 次元の地図を生成することにより，どこにどの種類のボールがあるかを常に把握しているため，効率的に被災者及び死者を回収することができる。

3.2.2 プログラム実行時間の最小化

Arduino 系列のマイコンをプログラムするにあたって，ArduinoIDE（Arduino 統合開発環境）が指定している関数，つまり IO ピン（Input/Output 端子）などの入出力関数は，Arduino の API（アプリケーション・プログラミング・インタフェース）を見る限りプログラムの実行時間を短くするようには作られていない。マイコンとは演算・制御装置（CPU），メモリ装置（RAM や ROM），入出力回路（I/O），タイマー回路などを一つの集積回路に実装した製品で，単体でコンピュータとしての一通りの機能を有するものである。ラインセンサを 1 サイクル動かすには IO ピンの入出力関数を 10 回以上使用することになるが，この入出力関数は私たちが期待するほど高速には動かないため，時間当たりに何サイクル動かせられるかが重要になるラインセンサのプログラムにはあまり向いていないと考えることができる。そこで，Arduino のレジスタを直接操作し IO ピンの入出力を行うことで既存の関数よりもプログラムの実行時間を短縮することができ，時間当たりで実行できるプログラムの回数を増やすことができる。レジスタとはマイクロプロセッサ（MPU/CPU）内部にある，演算や実行状態の保持に用いる記憶素子のことである。しかしレジスタのピン番号はマイコンに搭載されている CPU（中央演算処理装置）の種類によって異なるため，他のマイコンでレジスタを直接操作しているプログラムが動かせなくなってしまうといったデメリットも存在する。このようにマイコンのレジスタを直接操作することによってプログラムの実行時間を最小化しライントレースの精度を高めている。

3.2.3 カメラによるオブジェクト検出

このロボットには Pixy2 というカメラを搭載している。Pixy2 は，オブジェクト検出にカラーベースのフィルタリングアルゴリズムを使用しており，この方法は高速で効率よく，比較的安定性もあるポピュラーなものである。イメージセンサを通して各 RGB ピクセルの色相と彩度を計算し，それらをプライマリフィルタリングパラメータとして使用している。通常照明や露出を変化させると，オブジェクトの色相は大きく変化しないけれども，カラーフィルタリングアルゴリズム上では深刻な影響を与える可能性がある。しかしながら Pixy2 のフィルタリングアルゴリズムは照明や露出の変化に対しても強く，安定性が高いと考えている。このカメラを利用して距離センサでは判別できないレスキューキットを検知することができる。

3.3 電子回路の設計と構造

3.3.1 電子回路の設計概要

このロボットに搭載されている回路基板はすべてプリント基板で構成されている。プリント基板とは，集積回路，抵抗器，コンデンサー等の多数の電子部品を表面に固定し，その部品間を配線で接続することで電子回路を構成する板状またはフィルム状の部品であり，主に基材に対して絶縁性のある樹脂を含浸した基板上に，銅箔など導電体で回路（パターン）配線を構成する，いわゆるプリンテッドエレクトロニクスの一種である。これにより，プリント基板を用いることでユニバーサル基盤に部品を実装する時よりも高密度な配線をすることが可能である。なぜなら図 6 のように SMD 部品（表面実装部品）を用いることが可能だからである。また，プリント基板を用いることによってジャンパワイヤの使用を少なくすることができ，配線ミスなどのロボットの駆動に関わる重大な欠陥を防ぐことができる。

図 6

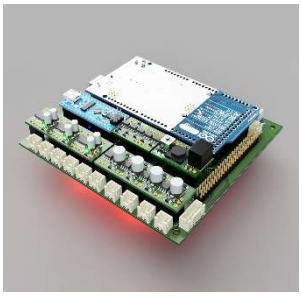

図 7

また，図 7 のように基盤は主に 7 個のモジュールで構成されており，モジュールごとに分けることによってどこかが壊れたとしても，壊れたモジュールだけを交換することによって修理できるという利点がある。そしてロボットを作り変えたとしても各モジュールを再利用することができる。

3.3.2 ラインセンサとカラーセンサの構造

このロボットに搭載されているラインセンサは 25 個のフォトトランジスタ，16 個の白色 LED，4 個の RGBLED で構成されており，ラインセンサはメイン基盤の裏側に直接ピンヘッダで接続されている。なぜなら 25 個ものデータと LED の信号線をコードでつなぐとなるととても多くのコードが必要な上に，コードのコネクタをつける面積がメイン基板の面積を圧迫してしまうからである。また LED を制御する LED ドライバモジュールもメイン基板に搭載されており，16 個の LED を一括操作することができる。フォトトランジスタのうち 12 個はアナログ読込み，他の 13 個はデジタル読込みをしている。アナログでデータを読み込む理由は 25 個の内 12 個はラインセンサとカラーセンサの両方の機能を持っているからである。ラインセンサの場合，半固定抵抗で閾値を決めることができデジタルで読み込んでも支障はないが，カラーセンサの場合 RGB それぞれの色を順番に当ててその反射光の強さを読み込むためアナログである必要があるからである。このようにこのロボットは 25 個のラインセンサと 12 個のカラーセンサでラインを線としてではなく面として読み取っている。

3.3.3 モータードライバの構造

このロボットに搭載されているモータードライバは二種類あり，一種類目は TB67H450FNG，二種類目は TB6612FNG を使用したモジュールとなっている。モータードライバを使う理由としては，メインマイコンである Arduino Due の IO ピンは 1 ピンあたり 40mA が電流出力値の定格電流であるが，モーターは最大 4A 程の電流を消費するためモーターを直接 Arduino につないでしまうと定格電流を超過し Arduino が壊れる可能性があるからである。またモータードライバを中継することによってモーターの回転する速さや，回転の向きを制御することができる。モータードライバを二種類使っている理由は，ロボットのホイールにつながっているモーターと被災者やレスキューキットを回収するモーターでは要求される電流の大きさが違うためであり，モータードライバを使うことによって Arduino などのマイコンを過電流から守ることもできるからである。

4. 走行実験

4.1 角度走行

プログラムにより角度走行させ，ロボットに搭載されたレーザー距離センサで移動前と移動後のフィールド座標を読み取らせた。移動時間指示を 5000ms，モータードライバに入力する PWM （パルス幅変調）のデューティー比（Arduino Due の場合は 0 から 255）を 50 とし，横方向を 0 度として，10 度間隔で斜行させたときの誤差角度を計測したものを 10 回試行し，平均した結果を図 8 に示す。

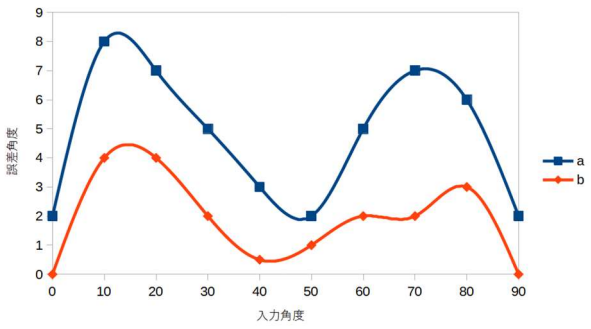

図 8　斜行時の誤差角度

グラフ a の時の制御は次による。

式（5）において，Vx = X，Vy = Y，Pd = 150，ωz = 0（斜行）とすると，次のように計算できる。

$$\begin{bmatrix}\varphi_{1w}\\ \varphi_{2w}\\ \varphi_{3w}\\ \varphi_{4w}\end{bmatrix}=\frac{P_d}{R_w}\begin{bmatrix}1 & -1 & -(a+b)\\ 1 & 1 & -(a+b)\\ 1 & -1 & a+b\\ 1 & 1 & a+b\end{bmatrix}\begin{bmatrix}X\\ Y\\ 0\end{bmatrix}$$

式（7）

$$\varphi_{1w}=\varphi_{3w}=\frac{P_d}{R_w}(X-Y)$$

$$\varphi_{2w}=\varphi_{4w}=\frac{P_d}{R_w}(X+Y)$$

式（8）

式（8）より，斜向かいのホイールの速度が等しい。また，45 度方向の場合，|Vx| = |Vy|であるから，どちらかの組み合わせのホイールは停止し，フリーローラーが回転して移動する。任意の角度の場合，その角度を x 軸方向と y 軸方向に分解して Vx，Vy に代入すればよい。

そこでグラフ a を見ると，入力角度が 10°～30°の時，60°～80°の時にかけて誤差角度が増大しているが，0°，40°～50°，90°の時は比較的誤差が少ない。このような結果になった原因は，入力角度が 10°～30°の時，60°～80°の時のモーターの動力よりも減速ギアや床からの抵抗のほうが大きいためモーターが停止し，どちらかの斜向かいのホイールが動いていない状態になっているからである。しかし，この問題は単にモーターのトルクが足りないのではなくモーターに供給する電源電流が少ないことや，モータードライバの周波数がモーターの周波数とずれているために起因するものでもある。また，デューティー比を上昇させたとしても，斜向かいのホイールの速度差が大きくなるだけであり根本的解決にはならない。そのためプログラム修正のみで対応することは困難であり，現状この問題を解決することは難しい。

また，0°，40°～50°，90°の時に角度誤差がある原因は各モーターの個体差によるものだと考える。そこで個体差を修正するための項を式（5）に追加したものが式（6）である。4 個のモーターの回転速度の比（RPM の比）を Pi（i = 1，2，3，4）とする。モーターの個体差を考慮した各ホイールの速度は，

$$\begin{bmatrix}\varphi_{1w}\\ \varphi_{2w}\\ \varphi_{3w}\\ \varphi_{4w}\end{bmatrix}=\frac{P_d}{R_w}\begin{bmatrix}P_1 & 0 & 0 & 0\\ 0 & P_2 & 0 & 0\\ 0 & 0 & P_3 & 0\\ 0 & 0 & 0 & P_4\end{bmatrix}\begin{bmatrix}1 & -1 & -(a+b)\\ 1 & 1 & -(a+b)\\ 1 & -1 & a+b\\ 1 & 1 & a+b\end{bmatrix}\begin{bmatrix}V_x\\ V_y\\ \omega_z\end{bmatrix}$$

式（6）

となる。式（6）を使って角度走行をすると，モーターの個体差による速度の違いは解消され，誤差角度はグラフ b のようにグラフ a と比べて少なくなっている。

これらの計算により X，Y の 2 つのデータから様々な方向への移動が可能になっている。

4.2 超信地旋回

プログラムにより超信地旋回させ，ロボットに搭載されたジャイロセンサで回転角度を読み取らせた。式（6）において，計算して出力される 4 個の解の単位はモーターの回転角度である。しかし，ロボットに搭載されているモーターは速さと時間を指定して駆動することはできるが，回転角を指定して駆動することはできない。そのためモーターの回転スピードが一定である時の回転角を駆動時間に変換する必要がある。

Tm = h × デューティー比（h は 1ms かつデューティー比が 1 の時にモーターが回転する角度）とすると，

$$\begin{bmatrix}\varphi_{1w}\\ \varphi_{2w}\\ \varphi_{3w}\\ \varphi_{4w}\end{bmatrix}=\frac{P_d}{R_w}\begin{bmatrix}P_1 & 0 & 0 & 0\\ 0 & P_2 & 0 & 0\\ 0 & 0 & P_3 & 0\\ 0 & 0 & 0 & P_4\end{bmatrix}\begin{bmatrix}1 & -1 & -(a+b)\\ 1 & 1 & -(a+b)\\ 1 & -1 & a+b\\ 1 & 1 & a+b\end{bmatrix}\begin{bmatrix}V_x\\ V_y\\ \omega_z\\ \overline{T_m}\end{bmatrix}$$

式（9）

となり，モーターの回転時間を算出することができる。

この時，ω = 2π，a = 86，b = 76，R = 60 とし，横方向を 0 度として，10 度間隔で超信地旋回させたときの誤差角度を計測したものを 10 回試行し，平均した結果を図 9 に示す。

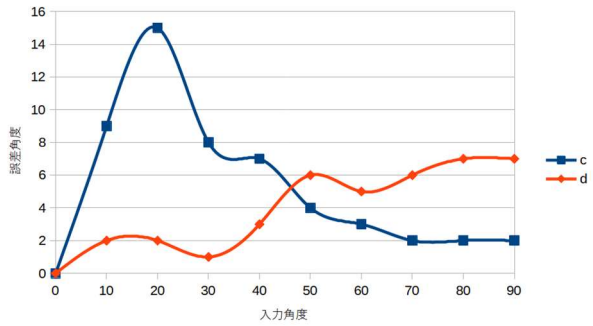

図 9　超信地旋回時の誤差角度

図 9 のグラフ c のデューティー比は 50，グラフ d のデューティー比は 100 としている。

この時の制御は次による。

$$\begin{bmatrix}\varphi_{1w}\\ \varphi_{2w}\\ \varphi_{3w}\\ \varphi_{4w}\end{bmatrix}=\frac{P_d}{60}\begin{bmatrix}P_1 & 0 & 0 & 0\\ 0 & P_2 & 0 & 0\\ 0 & 0 & P_3 & 0\\ 0 & 0 & 0 & P_4\end{bmatrix}\begin{bmatrix}1 & -1 & -162\\ 1 & 1 & -162\\ 1 & -1 & 162\\ 1 & 1 & 162\end{bmatrix}\begin{bmatrix}0\\ 0\\ \frac{2\pi}{T_m}\end{bmatrix}$$

式（10）

式（10）を計算して，

$$\begin{bmatrix}\varphi_{1w}\\ \varphi_{2w}\\ \varphi_{3w}\\ \varphi_{4w}\end{bmatrix}=\frac{2\pi P_d}{60T_m}\begin{bmatrix}P_1 & 0 & 0 & 0\\ 0 & P_2 & 0 & 0\\ 0 & 0 & P_3 & 0\\ 0 & 0 & 0 & P_4\end{bmatrix}\begin{bmatrix}-162\\ -162\\ 162\\ 162\end{bmatrix}$$

式（11）

すなわち，2π（1 回転）超信地旋回するときのホイールの回転時間は，

$$\pm 2\pi(86+76)P_iP_d \div 60T_m$$

式（12）

グラフ c を見ると，入力角度が 10°～20°の時の誤差角度が特に大きく，ほとんど入力角度と同じである。つまり，入力角度が 30°以下であるときロボットは旋回できないということが言える。この問題の原因として，角度走行の時と同じようにモーターの速度が小さいため，モーターがロボットを回転させる力よりも減速ギアや床からの抵抗のほうが大きいためモーターが停止しロボットが動かなくなると考える。角度走行の場合はこの問題を解決することは難しいが，超信地旋回の場合，左右 2 個ずつのモーターの速度は絶対値をとると等しくなるため，モーターの速度を上昇させたとしても左右のモーターの速度の絶対値の差は変わらない。また，回転角を制御しているのはモーターの駆動時間であるため，モーターの速度を上昇させ，駆動時間を減少させれば入力角度が 30°以下でも正確に超信地旋回をすることができると考える。

グラフ d はグラフ c の時と比べてデューティー比を 2 倍にした時の結果である。グラフ d を見てわかるように入力角度が 30°以下の場合誤差角度は 3°以下に収まっており，正確に超信地旋回をしている。しかし，グラフ c では 70°から 90°までの誤差は 2°であったのに対し，グラフ d では 6°以上の誤差が出てしまっている。これはデューティー比が大きいためにモーターが止まった状態でも慣性力が働いてロボットが止まり切れていないために引き起こされると考える。モーターが停止したときに逆進させて慣性力を打ち消すことは可能であるが，ライントレースにおいて 70°や 90°などの大きな角度の超信地旋回をすることはあまりなく，どちらかと言えば 10°，20°の小さな角度の超信地旋回のほうが使用回数は多い。そのためモーターの逆進制御のような不確定要素が存在する制御を導入ことよりも無視してしまったほうがメリットは大きいと考える。

これらの計算により回転角のデータから様々な角度に超信地旋回をすることができる。

4.3 角度走行と超信地旋回の並列制御

4.1 項，4.2 項では角度走行と超信地旋回を個別で制御してきたが，ライントレースにおいて個別に制御することは非効率である。なぜなら，ライントレースにおいてロボットの位置はほぼ常時ラインからずれている状態にあり，常にラインとのずれを修正しなければならず，修正時に行う制御において角度走行と超信地旋回を単体で使用することは例外処理を除きありえないからである。そこで角度制御と超信地旋回を並列制御することによって 1 回の制御の時間を少なくすることができると考える。

この時，式（9）において，Vx = 1，Vy = 1，ω = 1/4π，a = 86，b = 76，R = 60，デューティー比を 100°，移動量を 4000mm とし，横方向を 0 度としたときの，角度走行と超信地旋回をさせたときのロボットの軌跡を計測したものを座標平面上に表した結果を図 9 のグラフ h に示す。

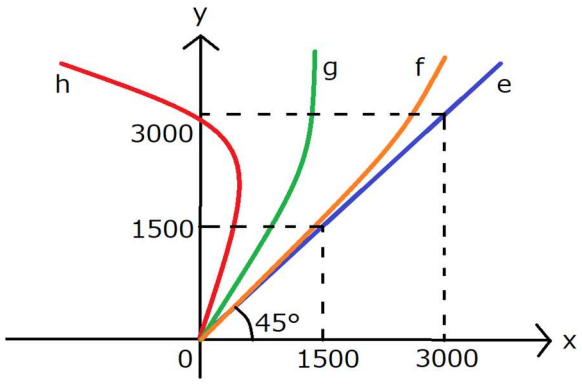

図 9　並列制御時の誤差

この制御は右な斜め 45 度方向に 4000 ㎜進みながら 45 度左超信地旋回をするものである。目標軌跡を図 9 のグラフ e とする。このデューティー比は角度走行のパラメーターであり，角度走行の駆動時間をアルゴリズムで求め，駆動時間内で 45 度回転することができる速さを超信地旋回のパラメーターとして利用している。

図 9 のグラフ h の時の制御は次による。

$$\begin{bmatrix}\varphi_{1w}\\ \varphi_{2w}\\ \varphi_{3w}\\ \varphi_{4w}\end{bmatrix}=\frac{P_d}{60}\begin{bmatrix}P_1 & 0 & 0 & 0\\ 0 & P_2 & 0 & 0\\ 0 & 0 & P_3 & 0\\ 0 & 0 & 0 & P_4\end{bmatrix}\begin{bmatrix}1 & -1 & -162\\ 1 & 1 & -162\\ 1 & -1 & 162\\ 1 & 1 & 162\end{bmatrix}\begin{bmatrix}1\\ 1\\ \frac{1}{4}\pi\\ \overline{T_m}\end{bmatrix}$$

式（13）

グラフ h を見ると，目標の軌跡であるグラフ e と比べて軌跡が大きくずれていることがわかる。まして角度走行の特徴である直線的な軌跡も残していない。このような軌跡になる原因として次のことが考えられる。

角度走行において走行中にロボットの回転軸が動かないことが必要であるが，超信地旋回の制御と並行して行っているため，走行中に回転軸がずれてしまっている。角度走行の式（12）では，回転軸の動きが考慮されておらず，超信地旋回の制御によって回転軸が動いたとしてもそのままのパラメーターで計算しているため，回転軸のずれが積み重なってグラフ h のような軌跡になってしまったと考える。

そこで角度走行と超信地旋回を分割して制御する。角度走行と超信地旋回の制御を交互に微小時間ずつ動作させ，1 回の微小時間で変化した超信地旋回によるロボットの回転軸の動きを角度走行の入力角度から引き，その値を使って微小時間，角度走行をさせる。この動作を繰り返すことにより角度走行時の回転軸のずれは補正されると考える。

角度走行の入力角度を θ，回転軸のずれを Δθ とし，

ΔVx = sin(θ - Δθ)，ΔVy = cos(θ – Δθ)，微小時間を Δt，Δt で回転する角度を Δωz とする。図 9 のグラフ g の時の制御は次による。

$$\begin{bmatrix}\varphi_{1w}\\ \varphi_{2w}\\ \varphi_{3w}\\ \varphi_{4w}\end{bmatrix}=\frac{P_d}{60}\begin{bmatrix}P_1 & 0 & 0 & 0\\ 0 & P_2 & 0 & 0\\ 0 & 0 & P_3 & 0\\ 0 & 0 & 0 & P_4\end{bmatrix}\begin{bmatrix}1 & -1 & -162\\ 1 & 1 & -162\\ 1 & -1 & 162\\ 1 & 1 & 162\end{bmatrix}\begin{bmatrix}\Delta V_x\\ \Delta V_y\\ 0\end{bmatrix}$$

式（14）

$$\begin{bmatrix}\varphi_{1w}\\ \varphi_{2w}\\ \varphi_{3w}\\ \varphi_{4w}\end{bmatrix}=\frac{P_d}{60}\begin{bmatrix}P_1 & 0 & 0 & 0\\ 0 & P_2 & 0 & 0\\ 0 & 0 & P_3 & 0\\ 0 & 0 & 0 & P_4\end{bmatrix}\begin{bmatrix}1 & -1 & -162\\ 1 & 1 & -162\\ 1 & -1 & 162\\ 1 & 1 & 162\end{bmatrix}\begin{bmatrix}0\\ 0\\ \frac{\Delta\omega_z}{T_m}\end{bmatrix}$$

式（15）

式（14）は角度走行，式（15）は超信地旋回の制御である。Arduino due のプログラムにおける最小時間は 4μs であるが，微小時間をこのようなとても小さい値にしてしまうとモーターが動かなくなると考える。よって微小時間を 1ms（1/1000s）つまり Δt = 1ms とした。式（14），式（15）を Δt 秒ごとに交互に動作させ，式（15）が Δt 秒動作したときに回転した角度を Δθ に代入し，超信地旋回による回転軸のずれを補正している。また Δt 秒で超信地旋回する角度は，

$$\Delta\omega_z = \omega_z \div (t \div \Delta t) \times 2$$

式（16）

しかし，図 9 のグラフ g では目標の軌跡であるグラフ e とのずれの差が大きい。原因として，微小時間 Δt の値が小さすぎ，モーターが動いていないことが考えられる。そこで微小時間 Δt を 100ms とする。この時の軌跡は図 9 のグラフ f が示している。グラフ f を見ると，近距離では目標の軌跡とほとんど一致しているが，長距離では少しずれていることがわかる。ただし，ライントレースにおける並列制御時の最大移動量は 30 ㎜であるからこの誤差はあまり関係ないと言える。

これらの計算により，角度走行と超信地旋回を並列制御することで移動の効率化を図っている。

5. 制御対象装置としての活用

5.1 ラインセンサの読み取り

ラインセンサは，縦 5 個，横 5 個の正方形で 25 個のフォトトランジスタで構成されている。フォトトランジスタは，フォトダイオードとトランジスタが一体化した構造になっており，フォトダイオードの出力電流（光電流）をトランジスタで増幅してから出力する素子である。フォトダイオードの光電流は，μA 台で，通常このレベルの微小電流をそのまま扱うのは困難なため，フォトトランジスタはそれを mA 台に増幅して出力する。また，増幅により照度が低い＝光電流が小さい場合でも十分な出力が得られるので感度を上げることにもつながっている。

5.2 ラインセンサのデータ処理

ラインセンサのデータ処理では，ラインセンサのデジタルデータからラインの位置，ラインとロボットの位置関係をプログラム上のメカナムホイールの制御関数で使用することができる値に計算する。

$$y = ax + b$$

式（17）

ラインセンサのデジタルデータからラインの位置の算出には，最小二乗法を使用する。ここで，回帰曲線を式（17）とし，s_{xy} は x と y の共分散，$s_x{}^2$ は x の分散，n は 2 変数データ（x，y）の総数，x_i と y_i は個々の数値，$\overline{x}$と$\overline{y}$はそれぞれの平均値とすると，次の式を立てることができる。

$$a = \frac{s_{xy}}{s_x{}^2} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

式（18）

$$b = \overline{y} - a\overline{x}$$

式（19）

式（18），式（19）にラインセンサのデジタルデータを代入することによってラインの概形を 1 次関数で表すことができる。シミュレーションにおいて，ラインの概形を 2 次関数や 3 次関数で表すことはできるが，あくまでもこのプログラムを動作させるのは 32 ビットマイコンであるため，計算量を少なくするために概形をあえて 1 次関数で表している。

5.3 PID 制御によるライントレース

ライントレースの制御方法として一般的にフィードバック制御の一種である比例制御とその発展である PI 制御，古典制御理論の中の PID 制御が使われている。これらの制御は，様々な制御手法が開発・提案され続けている今に至っても，過去の実績や技術者の経験則の蓄積により調整を行いやすいため一般な主力の制御手法であると言われている。

5.3.1 比例制御

比例制御は基本的なフィードバック制御であり，操作量を制御量と目標値の偏差の一次関数として制御するものである。

ここで，ある制御対象の制御する量を制御量，制御量に追従させたい希望の値を目標値，目標値を得るため制御対象を操作する量あるいは制御対象に入力する量を操作量と呼ぶ。なお，ある時刻 t での操作量を u(t)，出力値を y(t)，目標値を r(t)とする。このとき，目標値と現状の制御量との差を，制御偏差と呼び e(t)で表す。この時，e(t)は式（20）で表すことができる。

$$e(t) = r(t) - y(t)$$

式（20）

これらにもとづいて式に表すと，式（21）になる。

$$u(t) = K_p e(t) = K_p\{r(t) - y(t)\}$$

式（21）

この偏差に比例して操作量を変化させる動作を，比例動作あるいはP動作という。定数Kpは，Pゲインと呼ばれる。

しかし，比例制御の問題点として，次のことがあげられる。比例制御においてはKpを変えない限り，入力値に対して出力値は常に決まっている。しかし，実際に制御を行う場合には同じ入力値に対しても周囲の環境などによって出力値を変えなければならないことがある。ライントレースでの例を挙げると，ラインとロボットの中心がずれている場合，ロボットはモーターを駆動して中心をライン上に移動しなければならない。このような状況下でロボットの中心がライン上に到達するKpの値を使用して比例制御を行うとモーターの駆動量が足りず，目標値に到達しない。このようにして生じる制御結果と目標値との偏差を定常偏差またはオフセットという。

5.3.2 PI制御

残留偏差をなくすために周囲の環境が変わるたびに最適のKpを決定しなおすのは難しい。

$$u(t) = K_p e(t) + K_i \int_0^t e\,(t)dt$$

式（22）

そこで式（22）のように2つ目の項を付け加える。この項は残留偏差が存在する場合，その偏差の時間積分に比例して入力値を変化させる動作をする。つまり，偏差のある状態が長い時間続けばそれだけ入力値の変化を大きくして目標値に近づけようとする役目を果たす。定数Kiは，Iゲインと呼ばれ，この動作をI動作と呼ぶ。

しかし，PI制御の問題点として，積分時間が小さいほどI動作の影響が大きくなり残留偏差の修正が迅速に行われるが，小さすぎると目標値を行き過ぎたり（オーバーシュート），目標値の前後を出力値が振動したり（ハンチング）する現象を起こすことがある。積分動作は制御系の位相を遅らせる効果を持つので系を不安定にしやすく，ある程度時間が経過しないと働かないため，どうしても出力値を目標値に戻すために時間がかかり，正確な制御を妨げる原因となっている。

5.3.3 PID制御

5.3.2項の問題点を解決するために，

$$u(t) = K_p e(t) + K_i \int_0^t e\,(t)dt + K_d \frac{de(t)}{dt}$$

式（23）

式（23）のように，3つ目の項を付け加える。この項は急激な出力値の変化が起こった場合（例えば直角など），その変化の大きさに比例した入力を行うことで，その変化に抗しようとする役目を果たす。定数Kdはのゲインと呼ばれる。

この偏差の微分に比例して入力値を変化させる動作を微分動作あるいはD動作という。

上記のように比例動作，積分動作，微分動作を組み合わせた制御方法をPID制御という。操作量Uが比例項，積分項，微分項の3つの和となることから，3項制御とも呼ぶ。

PID制御において適切な比例ゲイン，積分ゲイン，微分ゲインを決定するには制御対象の入力に対する応答を調べておく必要がある。このためには入力値を階段状に変動させた時に出力値が応答しはじめるまでに要する時間，応答しはじめてからの変化の速度，入力値の変化量と出力値の変化量の比などを測定し，その値から設定する。

5.3.4 PID制御のゲイン値設定

PID制御の各パラメーターの値（Kp，Ki，Kd）は設計値なので，制御対象とは独立に自由に与えることができ，要求を満たす最適な値を決定する必要がある。しかし，一般的な制御対象に対して，最適なゲイン値を与える代数的な方法は存在せず，実際の調整は，数値シミュレーションを行うことや，実際の対象物に調整器を繋いで試行錯誤的に最適値を得ている。

設定の方法として，ジーグラ・ニコラス法やCHR法などがあるが，ライントレースをするときの環境はRCJの大会の場所によって異なるため，ゲイン値の設定を確実こなすことはできない。

5.3.5 PID制御の問題点

PID制御を使用してライントレースをするときに最も重要なことは適切なゲイン値を定めることである。ゲイン値設定は代数式が存在しないため経験則から設定することになるが，実際は各ゲイン値を少しずつ増減させ最適値を見つける作業が必要になる。この作業はとても時間がかかるものであり，ライントレースをする場所が一定ではないロボットにPID制御を実装したプログラムを入れると，調整した場所との環境の違いからライントレースが適切に行うことができないことがある。また，メカナムホイールの制御で使うパラメーターは移動量と移動角度であり，PID制御により求められる出力値と一致しない。このような問題を解決するために新たな制御方法を開発した。

5.4 座標制御

5.3.5 項の問題を解決するために，5.2 項で求められたラインの 1 次関数を用いた座標制御を開発した。

PID 制御は RCJ レスキューラインにおいても頻繁に使われている制御であり急激な入力値にも対応できるが，PID 制御において使用しているラインセンサは 25 個中の 3 個であり，他の 22 個が無駄になってしまっている。また，PID 制御はあくまでも線を見ることしかできず，ロボットが移動する方向のラインをあらかじめ読み取り，制御に反映することはできない。

そこで，縦 5 個，横 5 個の 25 個で構成されたラインセンサをすべて使うためにラインをラインセンサの線ではなく面で検知し，疑似的な画像処理を行う。つまり 5。2 項の最小二乗法を用いることでロボットの下のラインをロボットが移動する方向のラインの位置を考慮した 1 次関数で表すことができ，それを用いてロボットが実行すべき角度走行や超信地旋回の各パラメーターを算出することが可能である。最小二乗法によるラインの 1 次関数とロボットの位置関係を図 10 に示す。

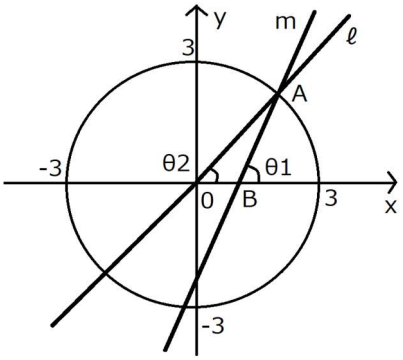

図 10　ラインとロボットの位置関係

図 10 において，原点をロボットの中心とし，$y^2 + x^2 = 9$ である半径 3 の円を描き，5.2 項で算出したラインの直線 ℓ を $y = ax + b$，直線 m と x 軸との角度を θ_1 とする。この時，円と直線 m との交点を A（α，β），x 軸と直線 m との交点を B とすると，交点 A の座標は，

$$\alpha = \frac{-ab \pm \sqrt{9a^2 - b^2 + 9}}{a^2 + 1}$$

式（24）

$$\beta = a\frac{-ab \pm \sqrt{9a^2 - b^2 + 9}}{a^2 + 1} + b$$

式（25）

となる。ただし，α は常に正とする。これらにより，交点 A と原点を結んだ直線の方程式 ℓ は，$y = \alpha / \beta \times x$ となる。直線 m と x 軸との角度を θ_2 とすると，θ_1，θ_2 は

$$\theta_1 = arctan(a)$$

式（26）

$$\theta_2 = arctan\left(\frac{\alpha}{\beta}\right)$$

式（27）

となる。以上より，メカナムホイールの制御関数のパラメーターを算出すると，

$$V_x = sin\left(arctan\frac{\alpha}{\beta}\right) = sin\theta_2$$

式（28）

$$V_y = cos\left(arctan\frac{\alpha}{\beta}\right) = cos\theta_2$$

式（29）

$$\omega_z = \left|\frac{\pi}{4} - \theta_1\right|$$

式（30）

となる。ラインセンサが図 11 のように反応しているとき，各パラメーターを計算すると，以下のようになる。

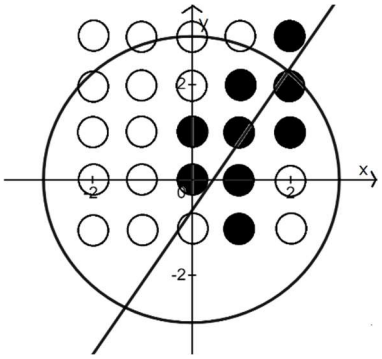

図 11　ラインセンサの反応と位置

ラインの 1 次関数：y = 1.506198x – 0.673554
角度走行時の駆動距離：30.000000 ㎜
角度走行時の移動角度：-40.714768°
角度走行時の駆動時間：540.767894ms
超信地旋回時の旋回角：-33581094°

図 11 のようにラインセンサはラインをきれいな直線で検出することは例外を除いてありえないため，最小二乗法が効果を上げていることが分かる。このように今回開発した座標制御は PID 制御とは違い，ロボットの未来位置を予測するのではなくラインセンサによって読み取ったラインを 1 次関数に変換し計算することでメカナムホイールの運動方程式を用いた制御の適したパラメーターの算出を行うことができる。また，座標制御は PID 制御よりも処理能力が高いため単位時間当たりの実行回数を増大させることが可能である。

このようなことから座標制御は，PID 制御実行時よりもライントレースの精度を高めるだけではなく，プログラム実行時間の最小化にも大きく貢献していると考える。

6. おわりに

以上にようにメカナムホイールの運動方程式を立てること自体は比較的容易であったが，モーターの個体差やホイールの誤差，角度走行や超信地旋回の目標値のずれなどの代数的に算出することのできない誤差を修正する計算式の算出には多くの試行が必要であった。角度走行と超信地旋回の並列制御においてもマイコンやモーターの性能に左右されることが多く，安定した制御を実現することは困難であったが，角度走行と超信地旋回の制御を交互に実行することにより解決できることが分かった。また，従来使用していた PID 制御に変わるものとして座標制御を導入し，運動方程式に最適なパラメーターを算出することにより正確なライントレースをメカナムホイールにより実現することができた。

そして，基礎基本のハードウェア設計，ソフトウェアルゴリズム，電子回路がロボットの底辺を支えており，これらがなければこのロボットに搭載されている先進的な各種部品や電子基板，発展的な構造，革新的なソフトウェアは成り立たない。また，ハードウェアはロボットの限界を決め，電子回路は性能を決め，ソフトウェアは動作を決めるものだと考える。これらのソフトウェア，ハードウェア，電子回路は個々に動くのではなくそれぞれが連携して動いていており，これら 3 個の要素の中でどれか 1 つの要素でもかけていたらロボットは最高の性能を出すことはできない。この 3 個の要素はそれぞれが補っているが，補える範囲もまた存在する。レスキューラインのすべてのタスクを完遂するためには，どれか 1 つの要素が特に優れていても完遂することはできず，すべての要素が一様に完璧に動くことによって完遂されるものであると考える。

7. 参考資料

超信地旋回のアルゴリズム，ソフトウェアアーキテクチャ，ロボットの 3D データ，回路データ，ロボットの詳細についての資料を下記に示す。

- ライセンサのシュミレーションプログラム
  https://github.com/kamiokannde/Rescue_project/tree/main/sensor%20sumilation
- ソフトウェアアーキテクチャ（Arduino due）
  https://github.com/kamiokannde/Rescue_project
- 超信地旋回のアルゴリズム
  https://github.com/kamiokannde/Rescue_project/tree/main/main%20flame/due
- 回路データ
  https://github.com/kamiokannde/KiCad_data
- ロボットの詳細（ROBOCUP ASIA-PACIFIC 2021 TEAM DESCRIPTION PAPER）
  https://github.com/kamiokannde/RCAP2021
  （The author was Mitsuyoshi Sugaya）
- ロボットの 3D データ
  https://github.com/kamiokannde/Fusion360_3D_data

上記のデータを複製，改変する場合，第 8 項「ライセンス及び使用上の警告」に沿って使用することを求める。

8. ライセンスおよび使用上の警告

下記の GitHub のリポジトリを「本リポジトリ」と呼称する。

GitHub：https://github.com/kamiokannde

8.1 ソースコード

本リポジトリに掲載されているソースコードは MIT ライセンスを使用している。使用者は MIT ライセンスに従ってソースコードを使用することを求める。

8.2 ハードウェア

本リポジトリに掲載されている 3D データ，KiCad データ，回路図またはそれらに準ずるものに関する情報等の掲載内容は，技術の進歩などにより予告なしに変更されることがある。

本リポジトリに掲載のデータ，図，表などに示す技術的な内容，プログラム，アルゴリズム，応用回路例，3Dデータなどの情報を使用する場合は，使用者のシステム全体で十分に評価し，使用者の責任において適用可否を判断することを求める。

本リポジトリに掲載のデータは，明示的にも黙示的にも一切の保障をしていない。

また，このデータを複製，改変しロボットなどに搭載することによって生じた不具合及び故障，動作不良などについての責任を負はない。

8.3 CA Prop 65

本リポジトリに掲載されている応用回路例中の部品には「カリフォルニア州法プロポジション 65（the Safe Drinking Water and Toxic Enforcement Act of 1986）」に掲載されている化学物質を用いた部品が使用されていることを警告する。また，応用回路例を使用する者はこれに注意し，適用ある環境関連法を遵守することを求める。

謝辞

最後に，アドバイザーの谷繋先生，ロボット部顧問の山本先生，同副顧問の中島先生，「Hidden Eagle Talon」チームメイトの小原君に感謝いたします。

引用・参考文献

1）ロボカップジュニアレスキューライン 2021 年ルール 日本語翻訳版
https://drive.google.com/file/d/1zPUsTqtEAeLpcp9eDOpt5YBlaFWkdanq/view
（図 1 の引用元）

2）メカナムホイールを用いた全方向移動車の制作
http://www2.lib.yamagata-u.ac.jp/you-campus/sangitan/kiyou-sangitan/11/p81-84.pdf
（2.2 項では細谷正廣氏の論文を参照した。）

3）ROBOCUP ASIA-PACIFIC 2021 TEAM DESCRIPTION PAPER
（The author was Mitsuyoshi Sugaya）
https://github.com/kamiokannde/RCAP2021/blob/main/RCAP2021_TDP.pdf

4）RoboCupJunior 2021 Rescue Line rubrics – Final
https://junior.robocup.org/wp-content/uploads/2022Rules/2022_RescueLine_Rules_final01.pdf

5）Open Source Initiative
https://opensource.org/licenses

6）OEHHA
https://oehha.ca.gov/proposition-65