研究テーマ　「オムニホイールとジャイロセンサを採用した自立型ロボット」

長野県岡谷工業高等学校　電気部
情報技術科　２年　髙林　蓮、武井　慶次郎、宮澤　絆希
顧問　電気科　小井土政範

1.　研究目的・概要

あらかじめ動作内容をプログラミングして動作する自立型ロボットについて、従来のメカナムホイールを用いたロボットでは、床面の摩擦係数によって横滑りするなど課題があった。そこでオムニホイールを採用することで横滑りせず移動させるロボットを製作した。また、ジャイロセンサを用いて自己姿勢把握をさせることで、より精度の高い移動を実現させた。

2.　競技内容

2.1 競技概要と課題

本研究では「ROBOCON IN 信州　キャリー競技」のルールに基づいた自立型ロボット競技を例に説明する。

自立型ロボットは保管庫から出発し、ふじ置き場に配置されたアイテムを取得し、ジュース出荷台に納品する。その後、保管庫へ戻ってくるというシンプルな競技である。

自立型ロボットが動作する領域（これをジュース工場エリアと呼ぶ）の床面は「パネコート」と呼ばれる摩擦係数がかなり少ない材料が用いられている。そのため、駆動輪が滑って移動できない・その場で旋回してしまうなどの課題がある。このことから、滑らずにまっすぐに走るというハードウェア的な工夫が求められる。

また、アイテム取得位置が各所にあるため、自立型ロボットは各種センサを駆使した自己位置把握によって取得場所および納品場所を往復することとなる。このことから、精度の高い自己姿勢把握から自己位置を推定するソフトウェア的な工夫も求められることとなる。なお、アイテム取得機構については本研究では割愛する。

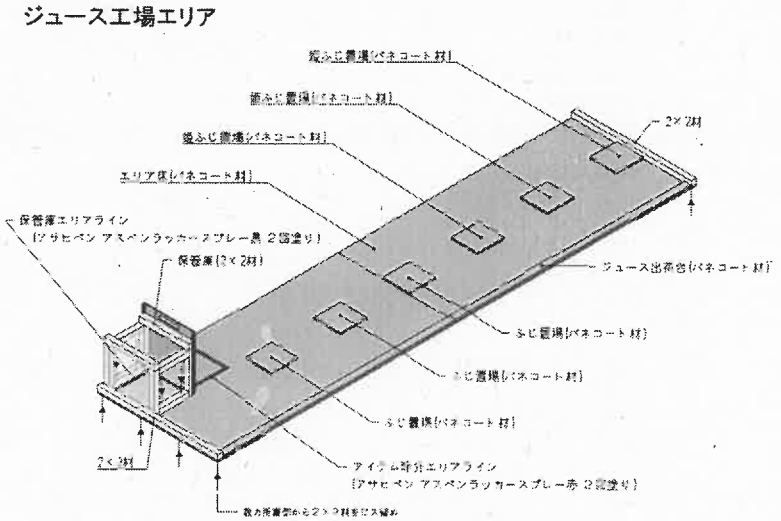

図１．コースイメージ図

3. ハードウェア的アプローチ

3.1 オムニホイール

過去に製作した自立型ロボットはすべてメカナムホイールを用いたロボットであった。メカナムホイールの利点は4輪の回転方向を制御することで、前後左右回転と自由自在に動作させることができることである。また、段差などに強く、旋回軸を多く持っている。ただし、45° に傾いたタル部分が空転することで横滑りをしたり、床面から浮いてしまうなど実際の動作では課題がある。操縦型のロボットであれば、修正操作を人間が指示できるため問題はないが、自立型ロボットには不向きであると考えられる。

そこで、オムニホイールを用いたロボットを設計した。オムニホイールはメカナムホイールと同様に4輪の回転方向の制御によって、前後左右回転と動作できる。また、駆動系を小型化でき、動作時の振動が少なく、タイヤが滑りにくいことが利点である。段差には弱いというデメリットもあるが、今回の競技では段差がない。これらのことから、オムニホイールを用いた自立ロボットを製作することとした。

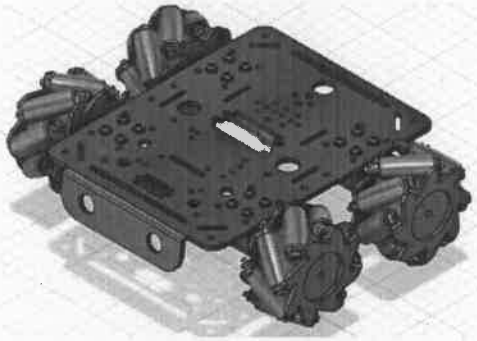

図2．メカナムホイール車体の設計

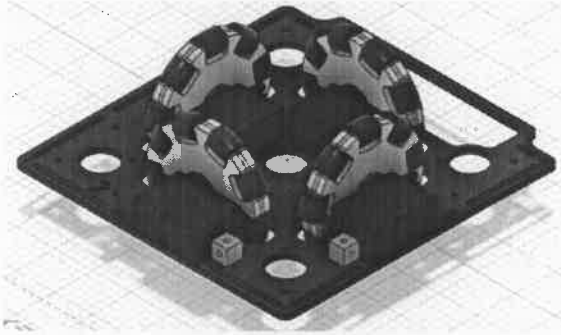

図3．オムニホイール車体の設計

3.2 連続回転サーボモータ

ホイールを回転させるためにはこれまで DC ギアードモータを利用してきた。しかし、DC モータには同じ型番（同じメーカー）であっても特性に違いがあり、同条件で動作させてもまったく同じ回転数では回転しないという大きな問題（個体差）があった。この状況で動作させると、回転数の違いから機体がわずかに旋回しながら走るため、直進することは出来ない。回転量を調整するために電圧を細かく制御する方法もあるが、DC モータは低電圧での動作時にトルクが低くなるため不向きである。

そこで、連続回転サーボモータを用いることとした。サーボモータにも特性の違いがあるもののその差は僅かである。また、制御回路（ドライバとエンコーダ）が内蔵されており、低速度の回転時にも適切なトルクで回転させることが可能である。

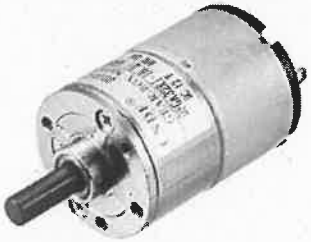

図4．DC ギアードモータ

図5．連続回転サーボモータ

4.　ソフトウェア的アプローチ

4.1 ジャイロセンサの実装とアルゴリズム

自立型ロボットの自己姿勢把握のため用いるジャイロセンサは「MPU6050」を用いた。このジャイロセンサは、I²C通信により XYZ 軸方向の加速度、XYZ 軸周りの角速度が計測できる。また、安価であり、Arduino 用のライブラリも用意されているため使用するための敷居が低い。実際に使用してみると、静止状態でジャイロ起動時からおおよそ 10 秒間で 3° ほど角度がずれていくドリフト現象が発生した。そのため、オフセット値を計算して角度を補正するようプログラミングした。

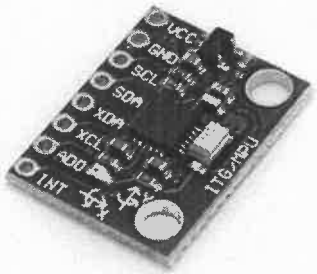

図6．6軸ジャイロセンサ「MPU6050」GY-521

この角度情報をもとに自動で補正動作をさせるために、自立制御のアルゴリズムを大きく変更した。従来はプログラムから直接モータ制御電圧を指定して出力していた。これを、「移動量変数」を作成し、行動内容とジャイロ補正の２つから移動量を積算させ、その値から制御電圧を決めるアルゴリズムとした。こうすることで、それぞれのプログラムを独立化させることができ、個別に処理内容を考えることが可能となる。また、ジャイロ機能を ON,OFF したり、追加でコントローラ入力を受け付けられるなど改造しやすく汎用性が高いプログラムとなった。

また、これらをオブジェクト指向に基づいてクラス化して実装することとした。クラス化することで、行動内容とジャイロ関係を別々のプログラマが同時進行で開発することができるため、効率的なプログラム開発が可能となった。

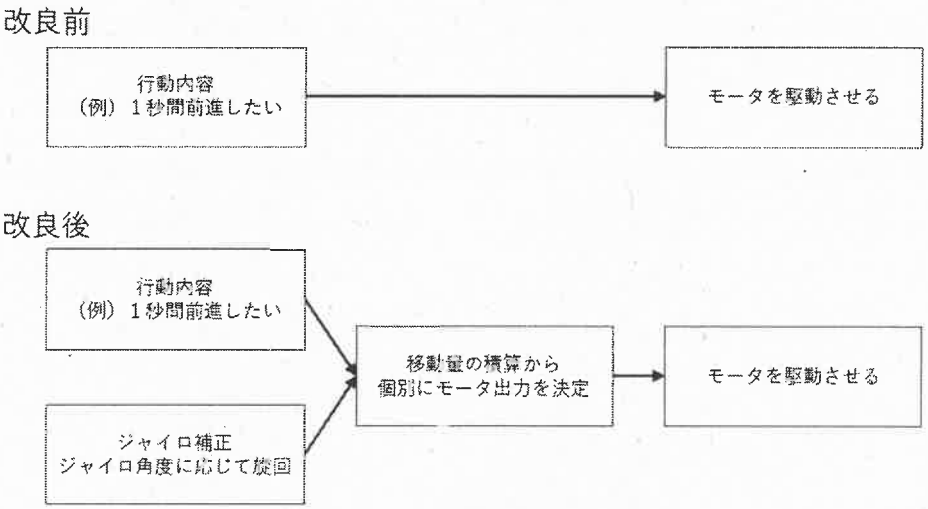

図7．自立制御アルゴリズム

4.2 ジャイロセンサによる機体角度の補正

ジャイロセンサによる角度に基づき、旋回する命令（移動量の設定）をする。

目標角度（基本は0°）に向かって旋回する制御をかけている。微分・積分制御も試したが、あまり効果が感じられなかったため比例制御のみとしている。

走行時に障害物に引っかかって機体角度が変わったときも、角度が戻るように動作するようになった。

```
//------ジャイロによる角度補正（係数操作）------
void rotateGyro() {
  //目標角度との差を計算
  turn_angle = ((int)(yprGyro[0]) - terget_angle);

  //旋回方向の設定
  if (turn_angle > 0) {
    //右に傾いていたら左旋回
    if (turn_angle > TURN_SENS_MAX) {
      input[R_L] += ROTAT_MAX * ROTAT_GAIN;
    }
    else {
      input[R_L] += map(turn_angle, 0, TURN_SENS_MAX, 0, ROTAT_MAX) * ROTAT_GAIN;
    }
  }
  else if (turn_angle < 0) {
    turn_angle = -turn_angle;
    //左に傾いていたら右旋回
    if (turn_angle > TURN_SENS_MAX) {
      input[R_R] += ROTAT_MAX * ROTAT_GAIN;
    }
    else {
      input[R_R] += map(turn_angle, 0, TURN_SENS_MAX, 0, ROTAT_MAX) * ROTAT_GAIN;
    }
  }
}
```

図８．角度補正プログラム

4.3 ジャイロセンサによる直線移動の補正

現在角度に応じて直線移動したときのずれに基づき、平行移動する命令（移動量の設定）をする。

角度と移動量（行動内容）から、平行移動誤差量を求め、その値だけ補正をかけている。

角度補正のみでは、平行移動してしまった分は戻らないのではないかと考え補正機能を追加した。障害物に引っかかった場合に、その障害物から抜け出すような動作が確認できた。

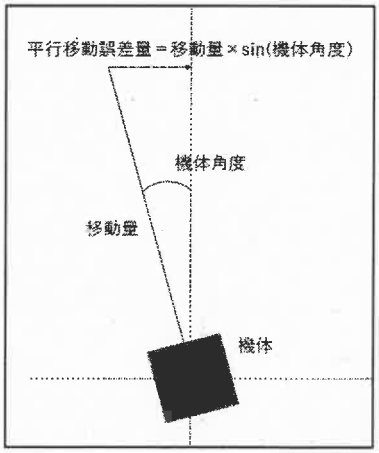

図９．直線移動の補正

```
//------ジャイロによる平行移動補正（係数操作）------
void translationGyro() {
  int distance;   //大きさ
  int direction;  //方向
  for (int i = M_F; i <= M_R; i++) {
    //進行方向に対する、位相90度方向の大きさ
    distance = (int)((float)input[i] * sin(yprGyro[0] / 360.0 * M_PI) * TRANSLATION_GAIN);

    //位相90度方向の大きさによって入力方向を切り替え
    if (distance < 0) {
      direction = i + 1;
      if (direction > M_R)  direction = M_F;
      input[direction] += abs(distance);
    }
    else if (distance > 0) {
      direction = i - 1;
      if (direction < M_F)  direction = M_R;
      input[direction] += abs(distance);
    }
  }
}
```

5. 動作検証

5.1 ハードウェア的アプローチの実装

パネコート上をジャイロ補正をせずに「メカナムホイールと DC ギアードモータ」および「オムニホイールとサーボモータ」で 1000mm 直進した場合の理想到達点からの横方向誤差は表 1 のとおりである。

メカナムホイールの特性から滑りにより進行方向が変わってしまい（横移動や旋回）全く違う方向へ動作してしまうことがあった。メカナムホイールを使用する場合は、より精度の高い補正の工夫が必要だと考えられる。また、オムニホイールの誤差の原因はサーボモータの個体差か機体の重量バランスにあると考えられる。

表 1．直線 1000mm 移動時の誤差（試走回数 20 回の絶対値の平均）

機体の仕様	横方向誤差
メカナムホイール+DC ギアードモータ（ジャイロ補正なし）	371.25 mm
オムニホイール+サーボモータ（ジャイロ補正なし）	259.75 mm

5.2 ソフトウェア的アプローチの実装

「オムニホイール+サーボモータ」の機体を用いて、ジャイロ補正の有無で 1000mm 直進した場合の理想到達点からの横方向誤差は表 2 のとおりである。

結果、特に機体角度の補正効果が大きいことがわかった。直線移動の補正については、角度のズレによる進行方向のズレが補正しきれていないことから効果が低いものと思われる。このことから、ジャイロ（角度と直線）補正をかけることにより、モータの個体差や重量バランスによる影響が軽減できることが確認された。

表 2．直線 3500mm 移動時の誤差（試走回数 20 回の絶対値の平均）

機体の仕様	横方向の誤差
ジャイロ補正なし	259.75 mm
ジャイロセンサによる機体角度の補正のみ	38.60 mm
ジャイロセンサによる直線移動の補正のみ	119.50 mm
機体角度の補正　＋　直線移動の補正	23.25 mm

この研究を通じて、「想像上の結果」と「現実的に起こる結果」には大きな隔たりがあり、その差をいかにして少なくするかが技術力なのだということがわかった。特に設計段階ではモータの個体差や、床面との滑りはまったく考慮しておらず、製作してから大きな課題としてぶつかった。解決のためのアプローチについて様々な方法を考えながら徐々に誤差が小さくなっていくことにものづくりの楽しさを感じることができた。この経験を自身の進路決定に役立てたい。また、本研究が後輩たちのロボット製作の糧となればと思います。