講師用

ロボット博士養成講座

ロボティクスプロフェッサーコース

オムニホイールロボット②

（第3回/第4回テキスト）

必ず、生徒に授業日と自分の名前を記入させるようご指導をお願いいたします。

第3回授業日（だいかいじゅぎょうび） 2024年　　月　　日

第4回授業日（だいかいじゅぎょうび） 2024年　　月　　日

名前（なまえ）

Human ヒューマンアカデミー ジュニア STE"AMスクール

ロボット博士養成講座 ロボティクスプロフェッサーコース

2024年5月授業分

ロボット博士養成講座

ロボティクスプロフェッサーコース

オムニホイールロボット②

第3回

力の合成と並進運動

講師用

目　　次

○　授業開始にあたって

授業のはじめは、着席させ、大きな声であいさつしてから始めます。

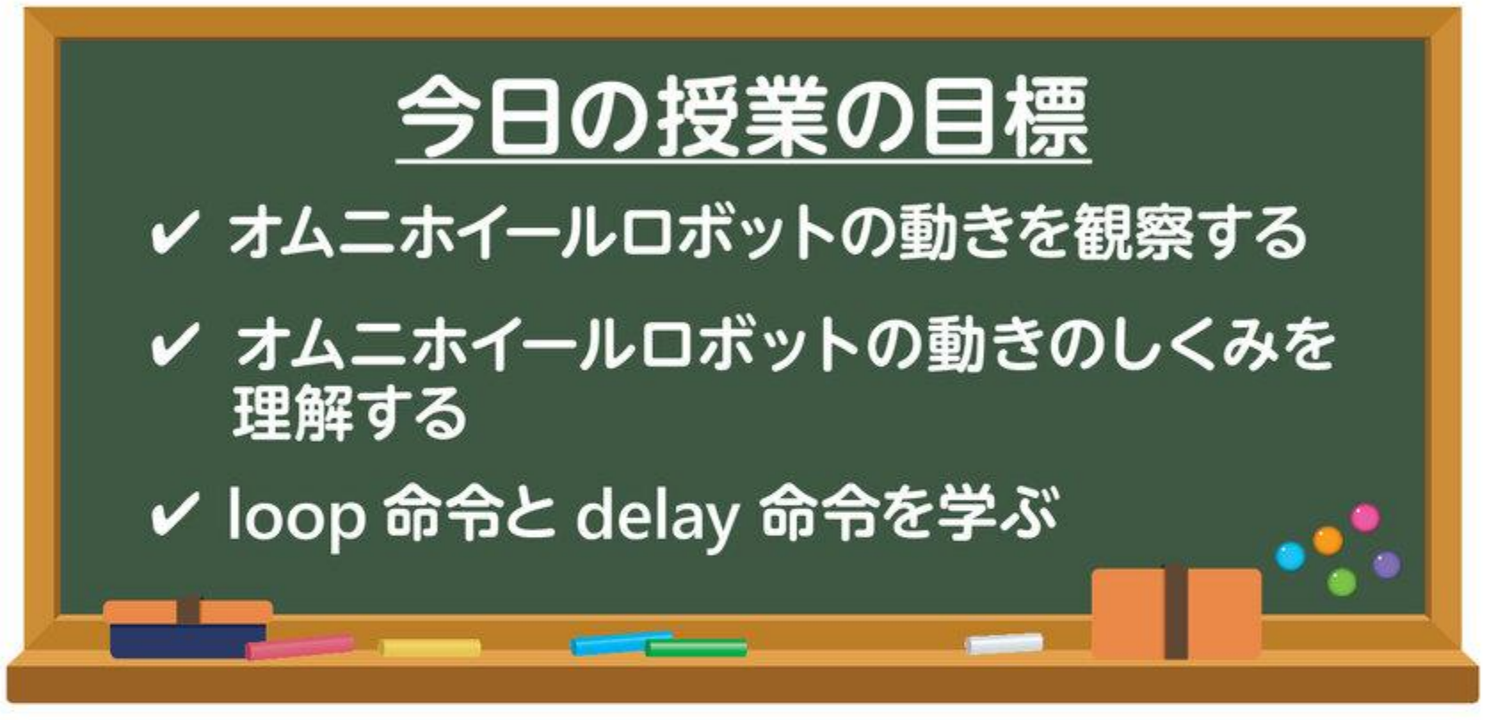
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（授業の目標を明確化することは大変重要なことですので、生徒によく理解させます）

目安時間は授業時間 120 分のうち、休憩 10 分程度を取ることを想定しています。
生徒の進捗状況により、休憩時間などを調整して授業を行ってください。

0．力の合成と並進運動（目安 10 分）

0.0. 「力の合成と並進運動」でやること



前回は、オムニホイールロボットをコントローラーで動かしながらプログラムの値を修正しました。また、オムニホイールの原理や、プログラムについても学びましたね。
今回も、オムニホイールを動かしながら、その原理やプログラムについて、さらに理解を深めていきます。
特に今回は、ロボット本体の向きを変えずに様々な方向に進むことができる原理について、実際に動かして観察しながら学びます。

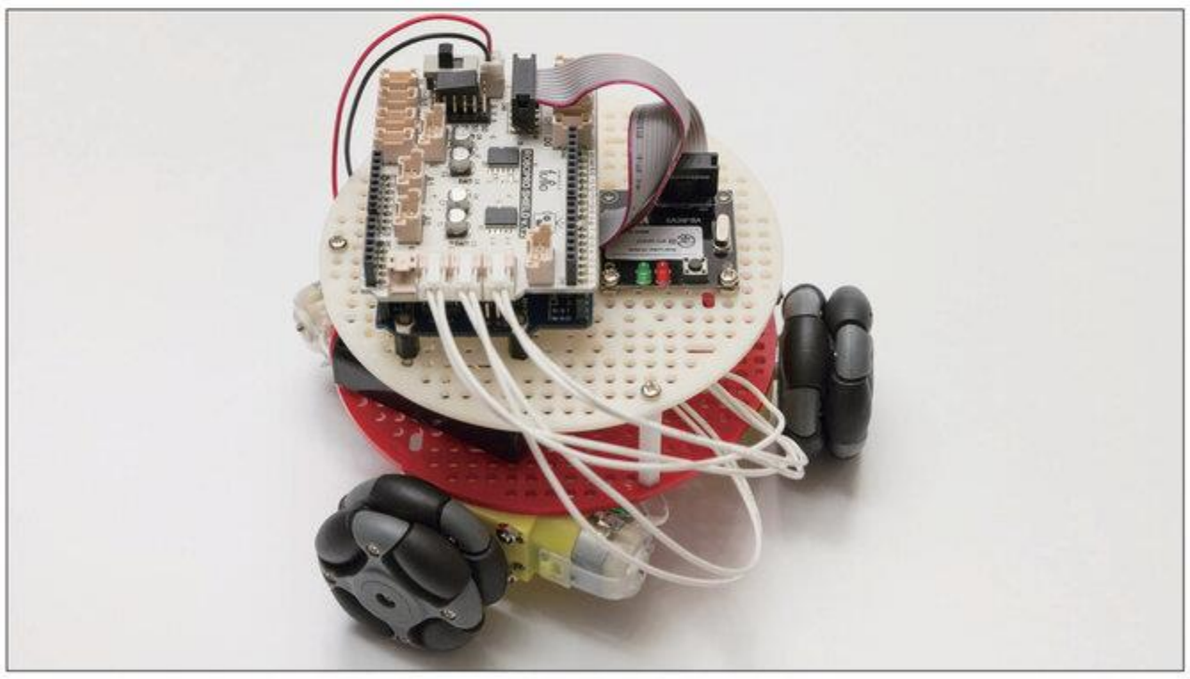


図0-0　オムニホイールロボット

0.1. 必要なもの

前回使ったロボットと、以下のパーツを準備しておきましょう。
なお、リボンケーブルやモーターなどの配線の接続や、ロボット本体やコントローラーの電池残量、ペアリング、などを事前にチェックしておきましょう。

USBケーブル 1	コントローラー 1
	

図0-1　必要なもの

0.2. モーターの接続

今回は、解説の都合上、3つのホイールを区別しておく必要があります。それぞれのホイールをロボプロシールドに接続しているコネクター名にちなんで、以下のように名づけます。各ホイールの名前を忘れたときは、ここで確認しておきましょう。

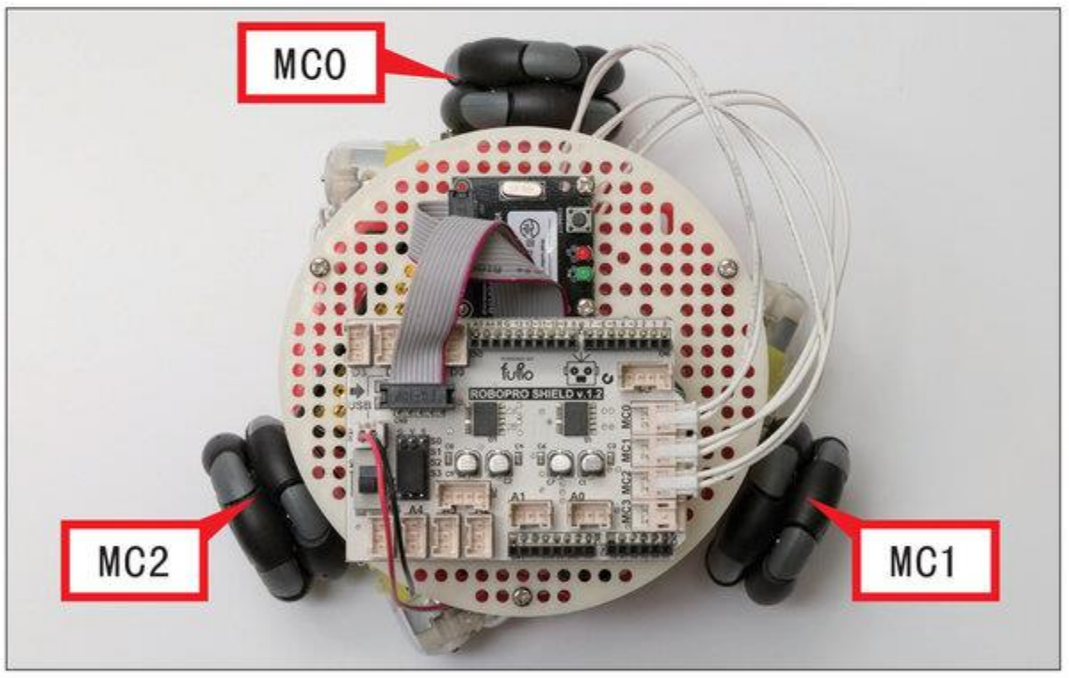


図0-2　オムニホイールロボットのホイール名

1. ロボットの動きを観察する（目安 20 分）

ここでは、コントローラーを使って、ロボットを前後・左右・ななめに移動させ、そのときのホイールの動きを観察します。

1.0. 動作確認

まずは、動作確認をしましょう。

プログラムの書き込み
RoboticsProfessorCourse1 > OmniWheelRobot2 > Remote2

前回調べた「調整値」を以下の黄色の部分に入れて、プログラムを実行しましょう。書き込みができたら、コントローラーで動かし、動作確認をしましょう。

プログラム「Remote2」より抜粋

```
// おまじない
RPomniDirect omniBot(1.0f,1.0f,1.0f,50.0f); // オムニホイール調整用パラメータ
PS2X ps2x;
```

1.1. 2 つのホイールが回転するとき

続いては、2つのオムニホイールを動かして、その様子を観察してみましょう。ここで使うプログラムでは、2本のレバーで操縦するイメージです。オムニホイールロボットの MC1 と MC2 を、コントローラーの左右のスティックで動かします。スティックの倒し方によって回転速度が調整できます。うまく直進させるには、左右のスティックをどのように操作すればよいか、ホイールの動きを観察しながらマスターしましょう。
では、以下のプログラムを実行し、実際に動かしてみましょう。

プログラムの書き込み
RoboticsProfessorCourse1 > OmniWheelRobot3 > Tank

さらに、オムニホイールロボットをひっくり返して、ホイールの動きを見てみましょう。

図1-0　タンクモードでの操縦（そうじゅう）の仕方

やってみよう！

ロボットの動作を観察して、正しいものを選ぼう。

1.ロボットがきれいに直進するとき、左右のホイールの回転速度はどうなるかな？
　①左の回転速度の方が速い　②右の回転速度の方が速い　③左右の回転速度は同じ

③

2.左右のホイールの回転速度が違うときは、どのような動きになるかな？
　①回転速度が遅い方向に曲がる　②回転速度が速い方向に曲がる　③ロボットが回転する

①

3.回転していないホイール（ MC0 ）は、どのような動きをしているかな？
　①地面から浮いている　②タル型ローラーが回転する

②

2. オムニホイールロボットの移動の仕組み（目安 70 分）

2.0. オムニホイールによる移動の仕組み

オムニホイールには2つの大きな特徴（とくちょう）がありましたね。

POINT

① ふつうのタイヤと同じように、ホイールが回転する方向へは、地面をけりながら進むことができる。
② ふつうのタイヤとは違い、ホイールの回転軸（じく）方向へも、黒色のローラーがすべることによって、進むことができる。

図2-0　オムニホイールの構造

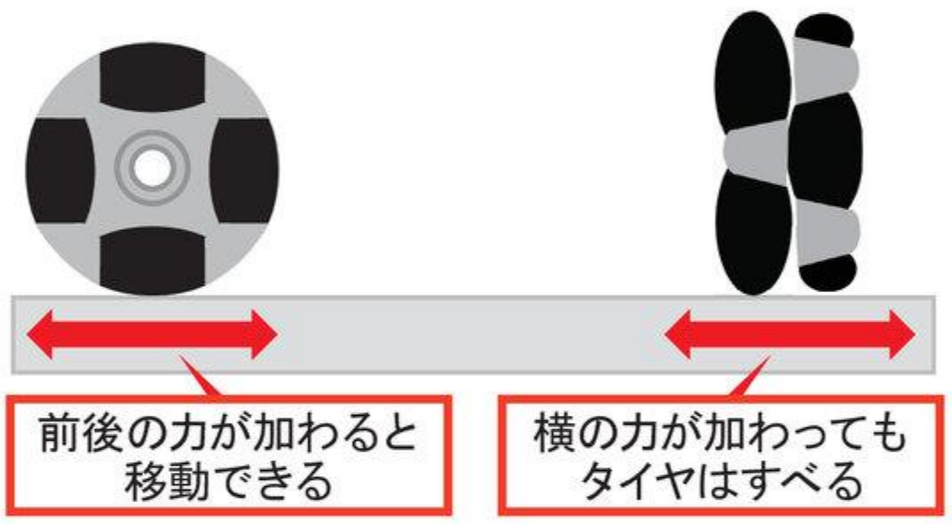


図2-1　オムニホイールの特徴（とくちょう）

このようなオムニホイールをそれぞれ違う向きに3個つけたのがオムニホイールロボットです。
3方向に配置されたオムニホイールにより、ロボットは、ホイールとローラーの回転で地面をけりながら、進みます。
そして、ロボットが進む方向は、それぞれのホイールによる力を合わせた方向によって決まります。
3個あるオムニホイールの回転をうまく配分することで、いろいろな方向へ移動（回転することも）できるようになります。

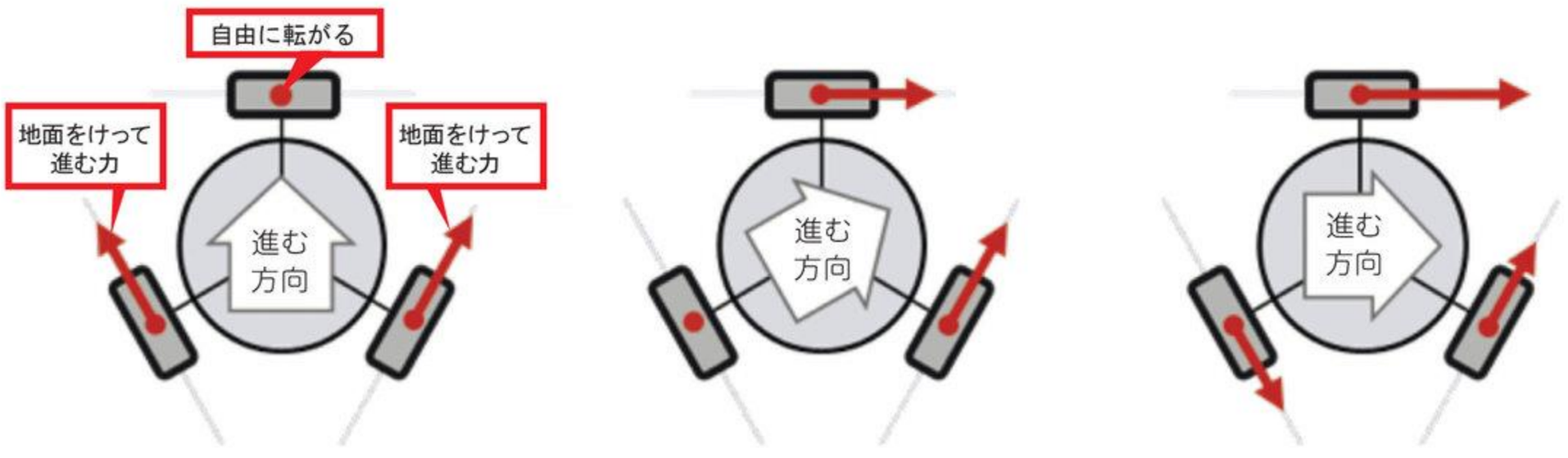


図2-2　各オムニホイールが地面をけって得る力とロボットが進む方向

コラム　全方位移動台車の利用用途と得意・不得意

オムニホイールロボットのように全方位（全ての方向）に移動できる乗り物はどのように利用できるのでしょうか？
製品化されている例では、フォークリフトや電動車いすなどがあります。向きを変えずにスライドするように移動することで、フォークリフトでは荷物の積み下ろしが、車いすではせまい場所での移動が、楽になるといったメリットがあります。将来的に全方位移動可能な自動車ができれば、せまいスペースへの縦列駐車も簡単になりますね。ただし、オムニホイールには、砂利道やでこぼこ路面が苦手といった欠点もあります。

2.1. 力の合成を学ぶ

1）力の表し方

力には3つの大事なことがあります。それは、「どこに力がはたらいているか（作用点）」、「どの向きに力がはたらいているか」、「力の大きさはどれくらいか」です。力は矢印で表し、始点、向き、長さを変えることで、それぞれをわかりやすく表します。

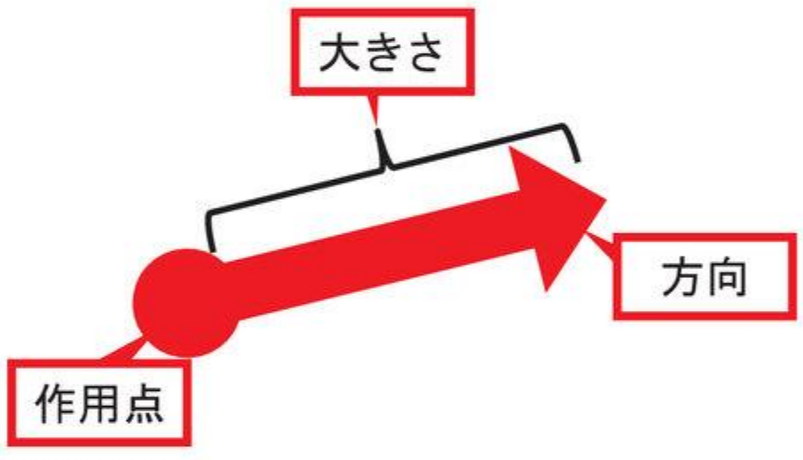


図2-3　力の表し方

講　力の表し方では、力の「作用する点」、「作用する方向」、「作用する大きさ」の３つが重要になることを、ここで覚えさせます。

オムニホイールロボットは、オムニホイールが地面をけって動くので、オムニホイールと地面が接している点が作用点になります。つまり、3つのモーターが全て動作していれば、作用点が3つできることになります。

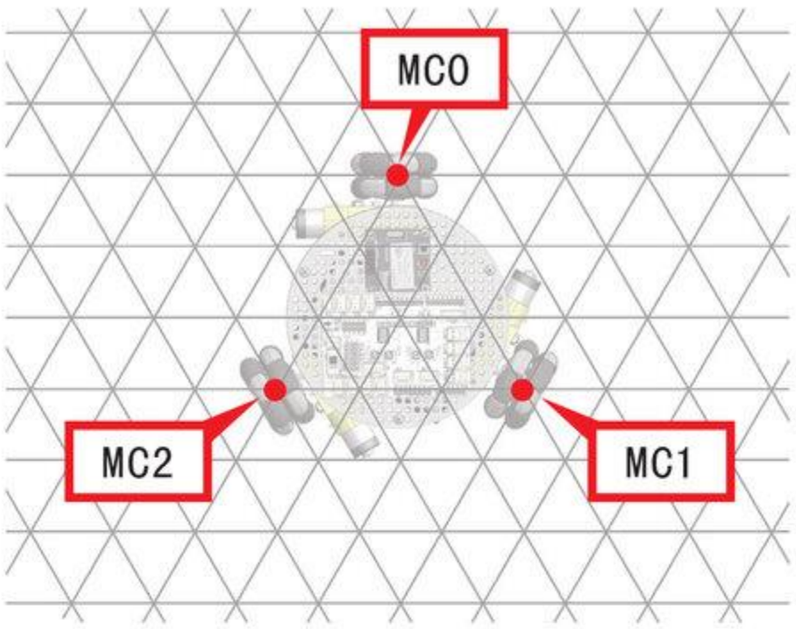


図2-4　オムニホイールロボットが地面と接する点

たとえば、mc0.rotate(100); と命令し、MC0 モーターを速度100で回転させたときに**図2-5**のような矢印ができるとしましょう。

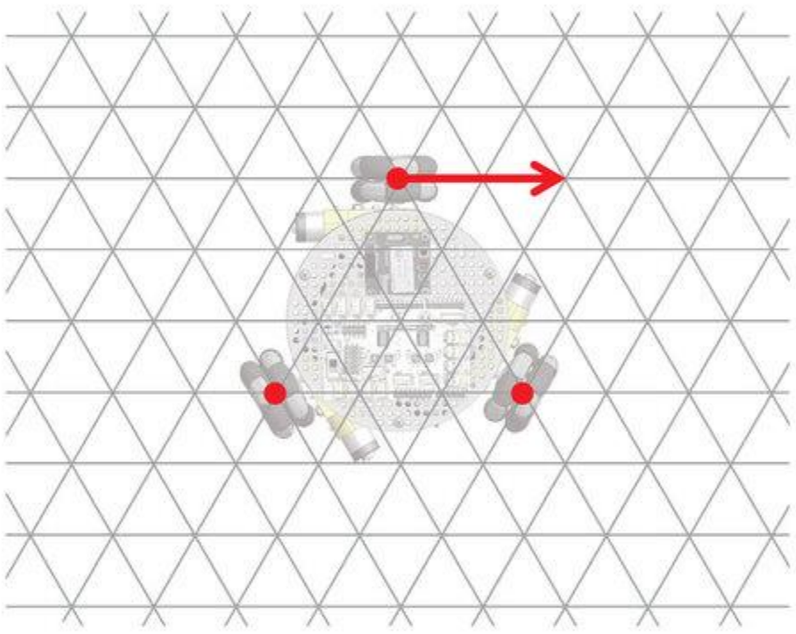


図2-5　MC0 モーターを速度100で回転させる

この場合、MC0 モーターの速度を200にすれば矢印の大きさが2倍の長さになり、-100にすれば矢印の向きが逆転しますね。

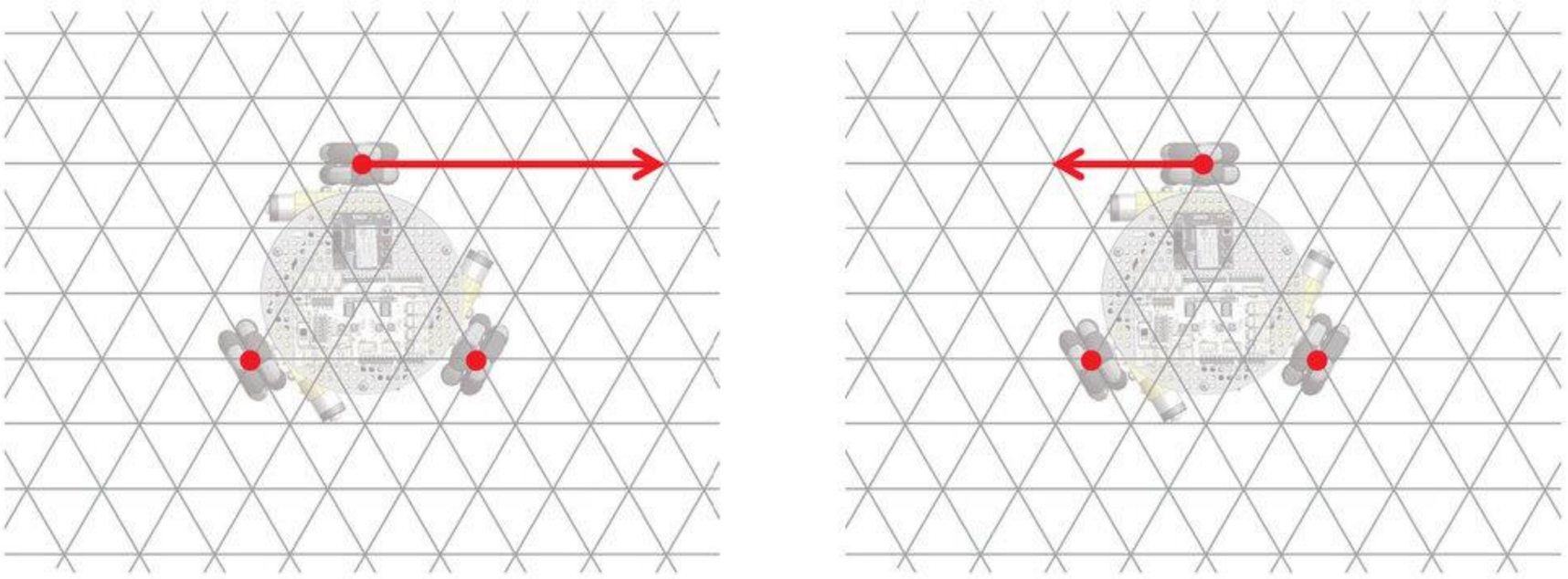


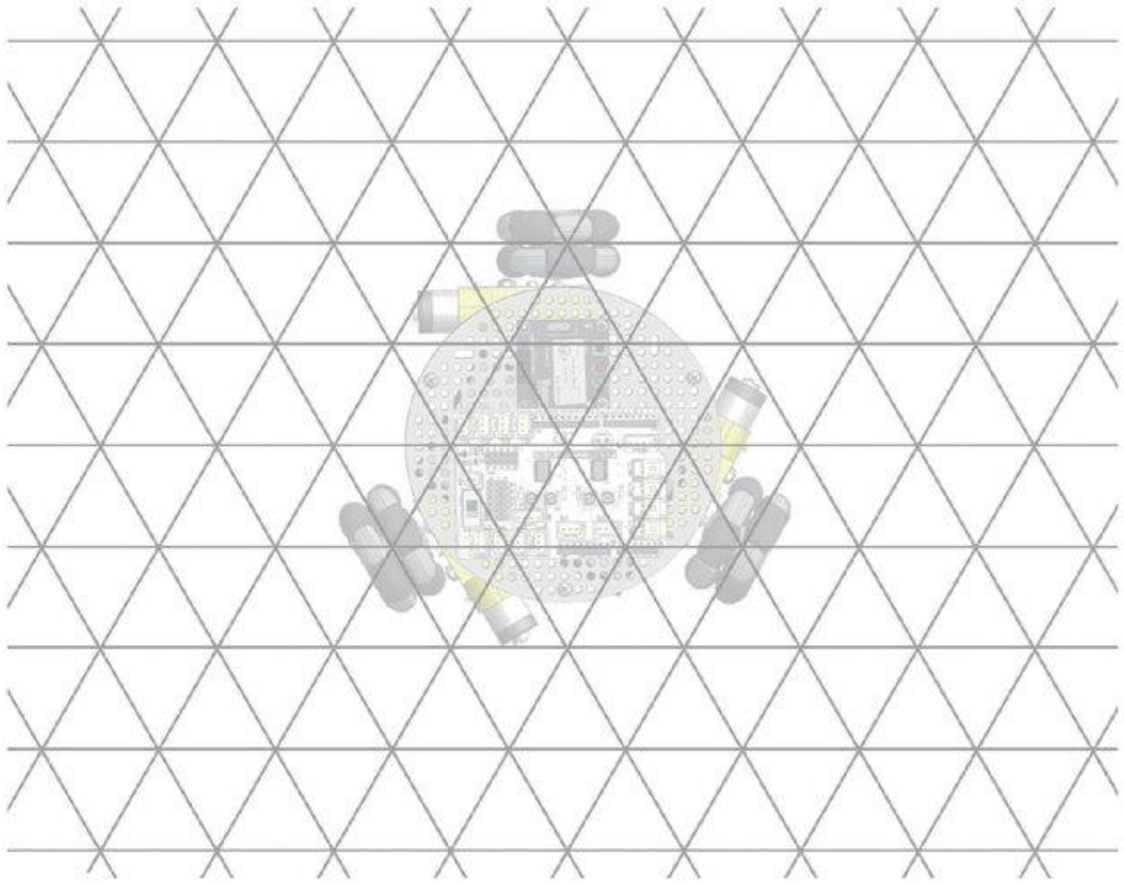
図2-6　速度200（左）と速度-100（右）

つまり矢印を見れば、どのモーターが、どれくらいの速度で、どちら向きに回転しているのかがわかるのです。

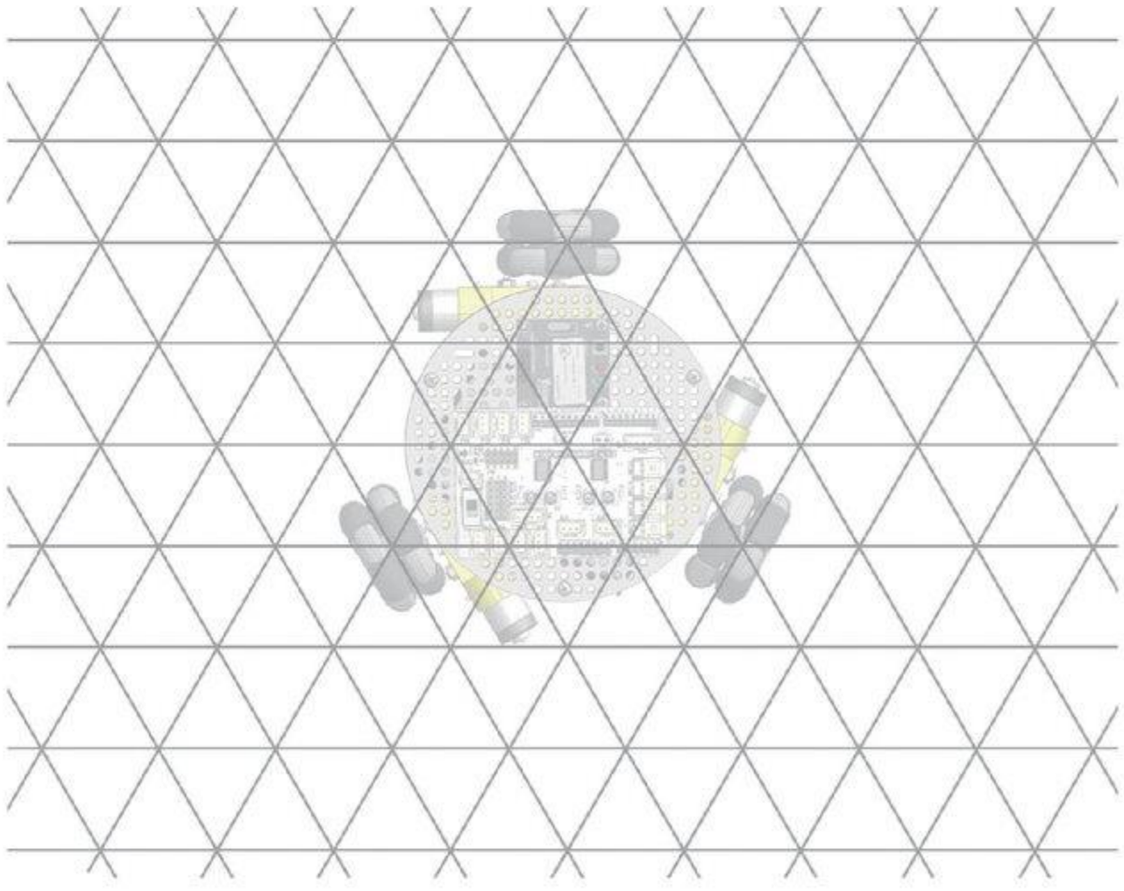
やってみよう！

次のように各モーターの速度を指定したとき、力の矢印はどのようになるかな？

1．すべてのモーターを速度100で回転させる。



2．MC1 モーターを速度-200、MC2 モーターを速度100で回転させる。



講　解答例は巻末に記載します。

2）力の合成

たとえば重いものを動かすとき、1人よりも2人で取り組んだ方がラクですよね。
これは、2人分の力が合わさり、1つの大きな力になるためです。

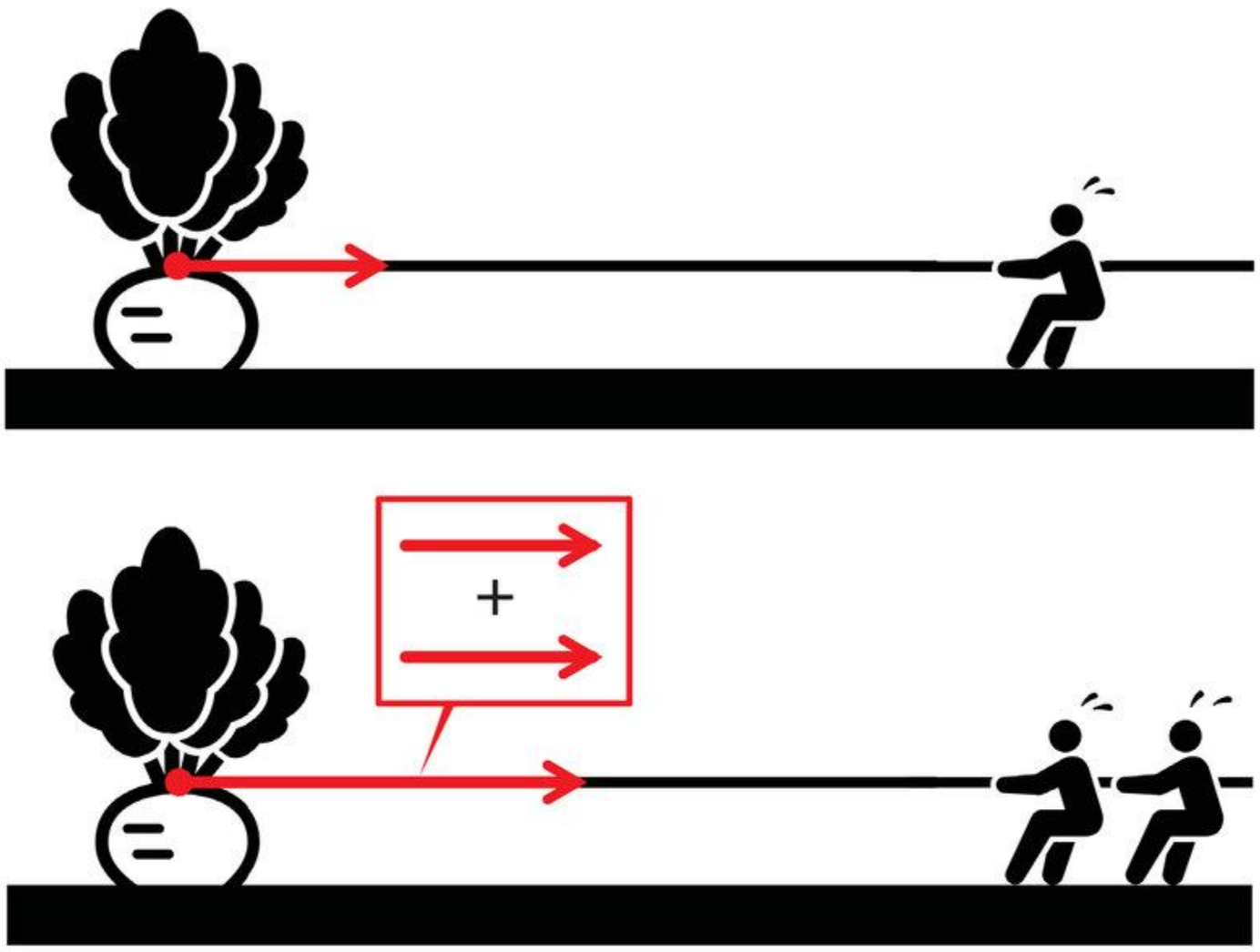


図2-7　力を合わせる

このように、複数の力をまとめて1つの力にすることを「力の合成」、合成してできた力を「合力（ごうりょく）」といいます。
上の**図2-7**の例ではまったく同じ大きさ、同じ向きの力を2つ合成しているので大きさが2倍になっただけですが、大きさや向きの異なる力を合成することもできます。

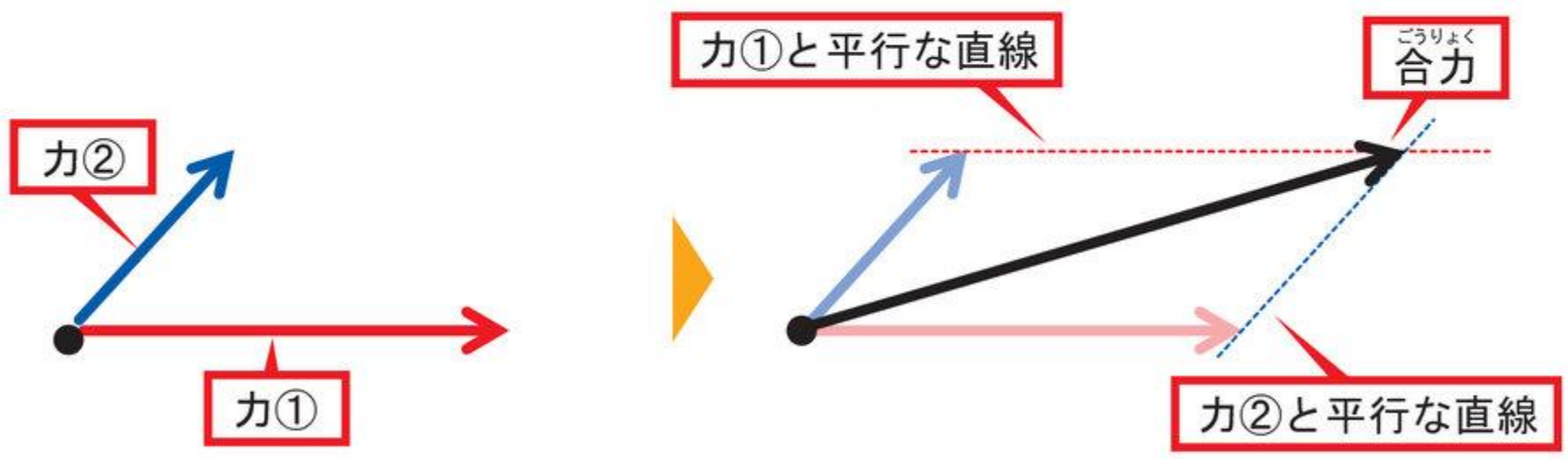


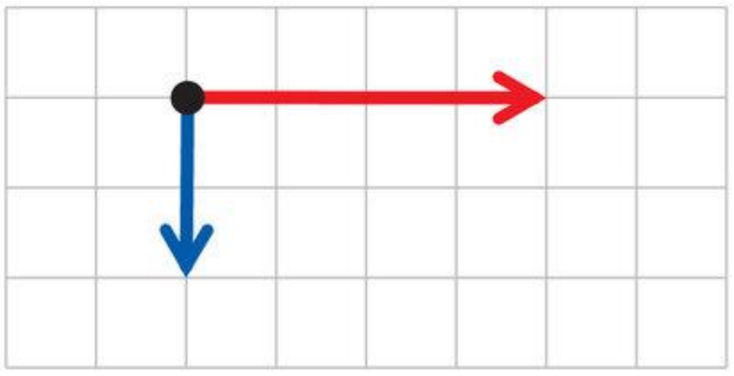
図2-8　大きさも向きもちがう力の合成

上の**図2-8**のように、力①と平行な直線、力②と平行な直線をそれぞれひいて、平行四辺形をつくるようにすると合力（ごうりょく）の矢印をえがくことができます。

やってみよう！

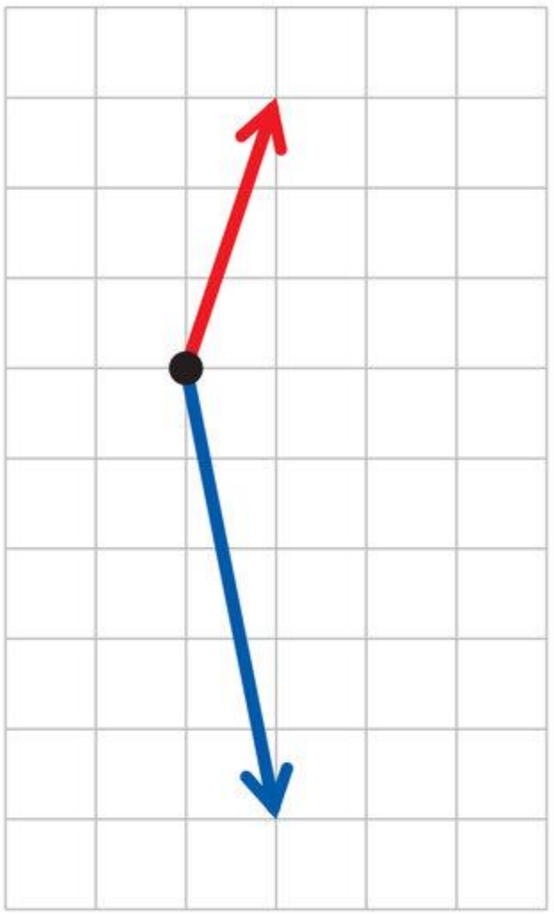
2つの力を合成し、合力（ごうりょく）の矢印を作図してみよう！

①



②

③



講

解答例は巻末に記載します。
生徒の学年によっては「平行」の概念や「平行四辺形」という図形の知識をまだ学習していない可能性がありますのでご注意ください（いずれも小４算数の内容です）。

2.2. オムニホイールロボットの進む方向

オムニホイールロボットで合力(ごうりょく)の矢印を作図しようとすると、モーターごとに作用点がことなるので思うようにいきません。

この場合は、まず作用点がロボットの中心部分に来るように矢印を移動させます。そうすれば、先ほどと同じ方法で合力(ごうりょく)を求めることができますね！

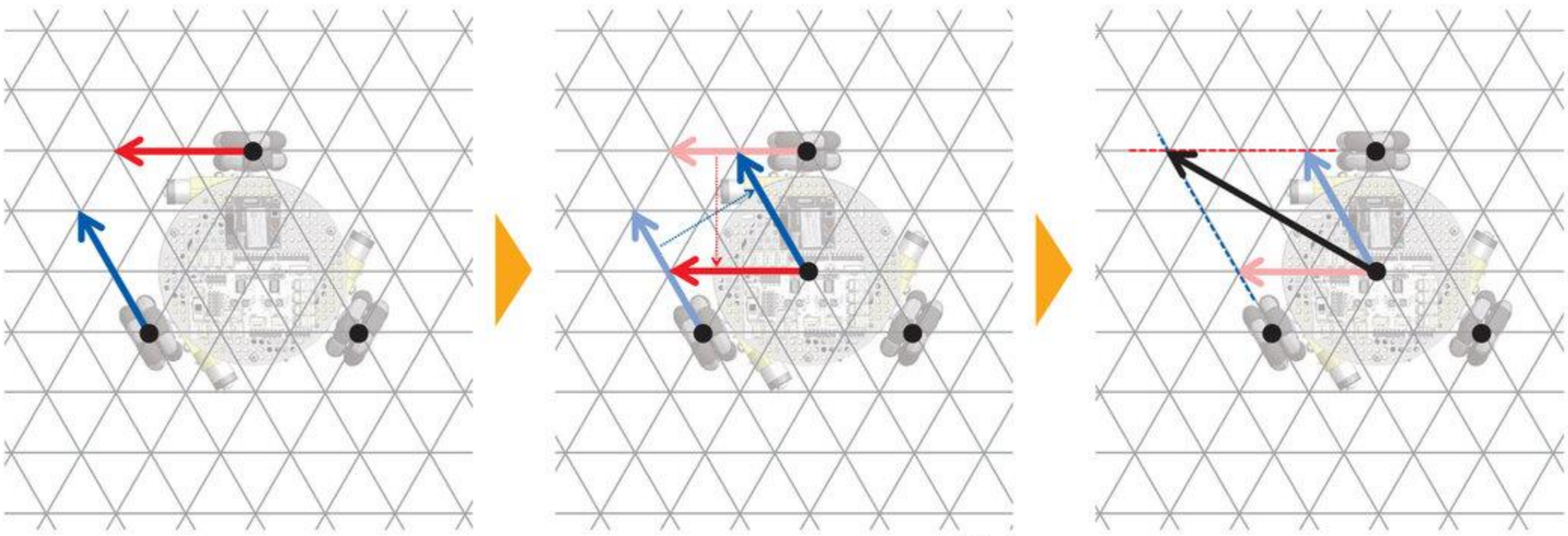


図2-9　オムニホイールロボットの合力(ごうりょく)の作図

この作図をしっかり理解できれば、実際にロボットを動かす前にどのように移動するか予測できます。

反対に、「ねらった方向に移動させるために、どのモーターをどう回転させればよいか」もわかるようになるので、プログラミングの効率がぐっと上がりますね！

図2-9の動きを再現したのが以下のプログラムです。実行して、合力(ごうりょく)の方向に移動することを確かめてみましょう。

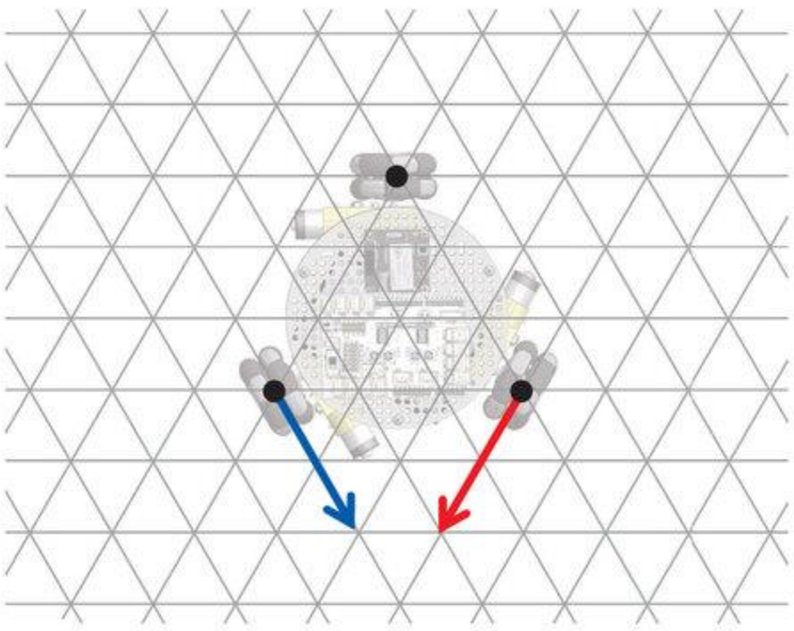
プログラムの書き込み

RoboticsProfessorCourse1 > OmniWheelRobot3 > Pattern1

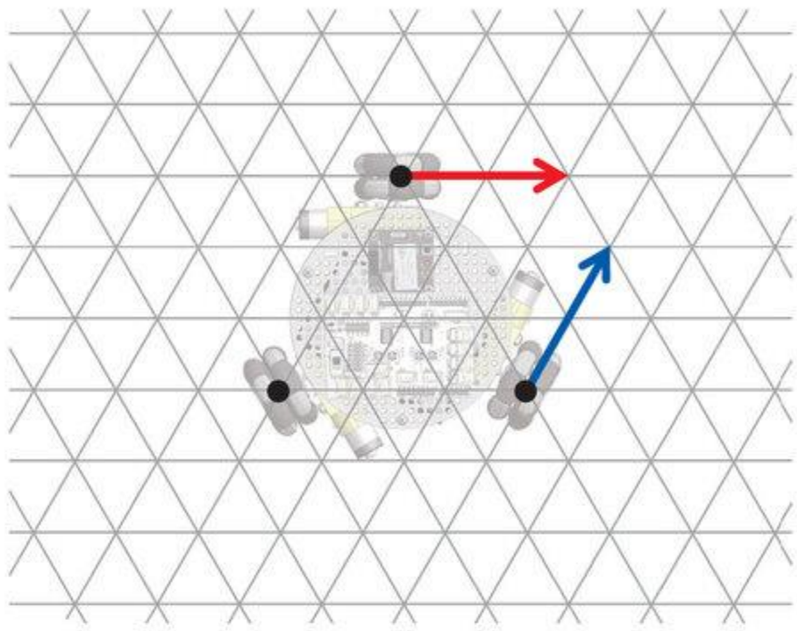
やってみよう！

1. 次の①～②のようにモーターを回転させると、ロボットはどの向きに移動するかな？合力（ごうりょく）の矢印を作図して予測を立ててから、プログラム「Pattern1」を書きかえて図の通りの動作にしてみよう！

①

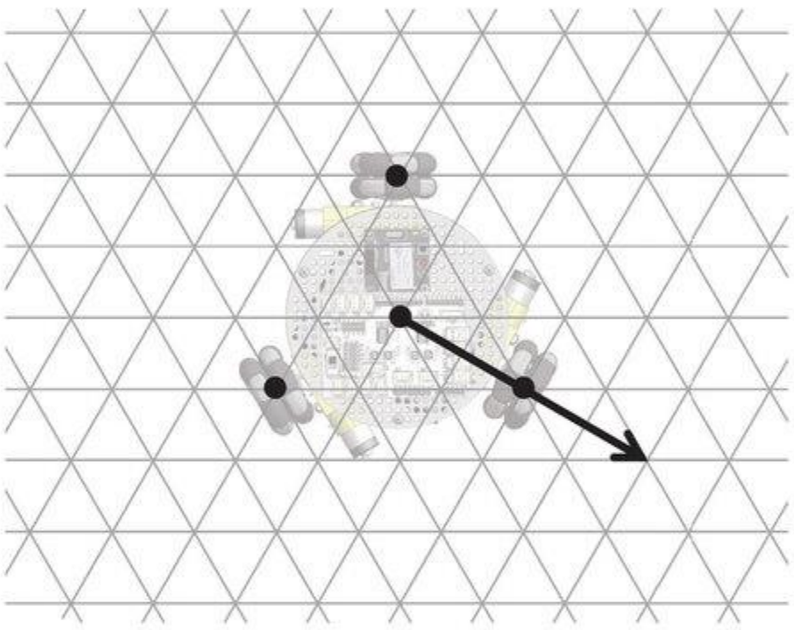


②

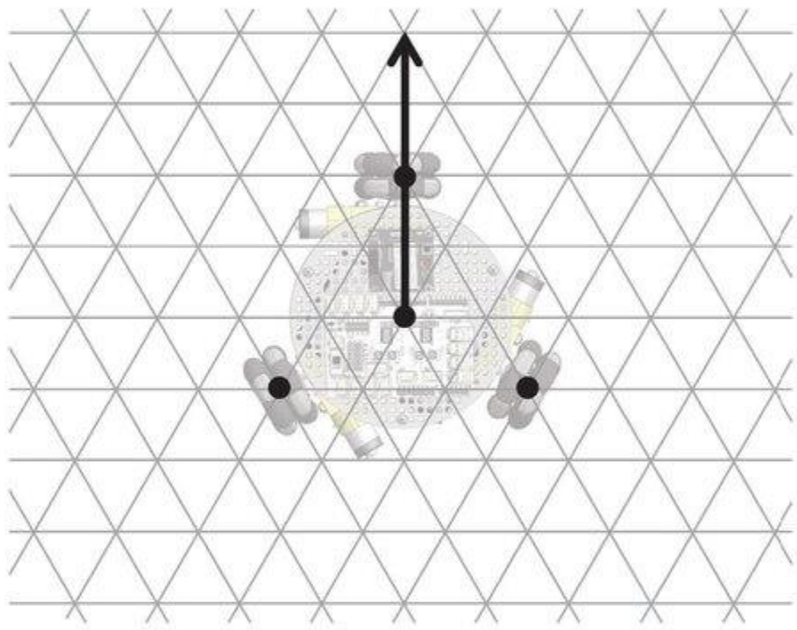


2. プログラム「Pattern1」を書きかえ、次の①～③の方向にロボットが移動するようにしてみよう！

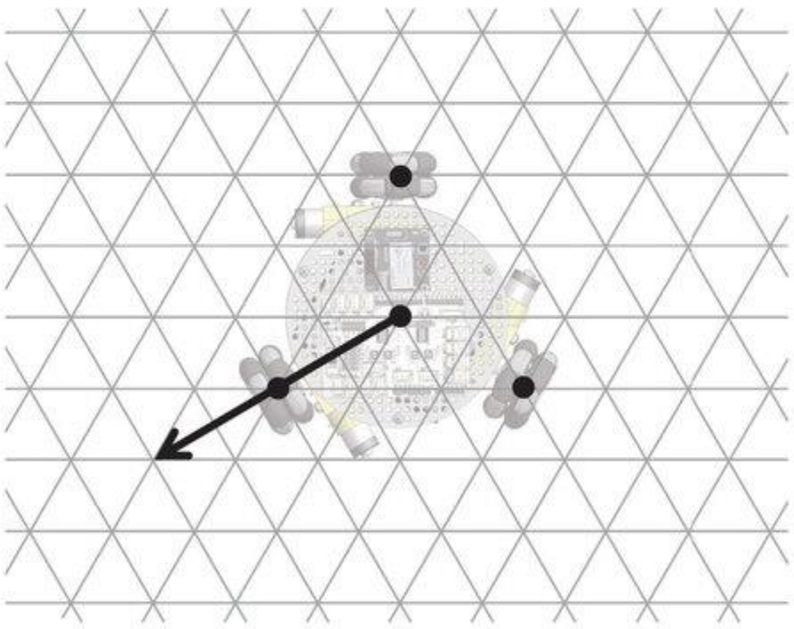
①



②



③



ヒント
①～③とも、動作させるモーターは2つだけで済むよ！

作図シート（自由に使いましょう）

講

前ページの「やってみよう！」の解答例は以下のプログラムです。

1.

① RoboticsProfessorCourse1 > OmniWheelRobot3 > Pattern2
② RoboticsProfessorCourse1 > OmniWheelRobot3 > Pattern3

2.

① RoboticsProfessorCourse1 > OmniWheelRobot3 > Pattern4
② RoboticsProfessorCourse1 > OmniWheelRobot3 > Pattern5
③ RoboticsProfessorCourse1 > OmniWheelRobot3 > Pattern6

また1．の作図の解答例は巻末に記載します。

作図シート（自由に使いましょう）

講　動きを考えるときのメモで使用させてください。

2.3. loop命令とdelay命令

1）ひし形の動きを考える

今回学んだ「力の合成」をマスターすれば、オムニホイールロボットを様々な方向に移動させることができますね！
今度は、ロボットがひし形を描くように移動するプログラムを考えてみましょう。

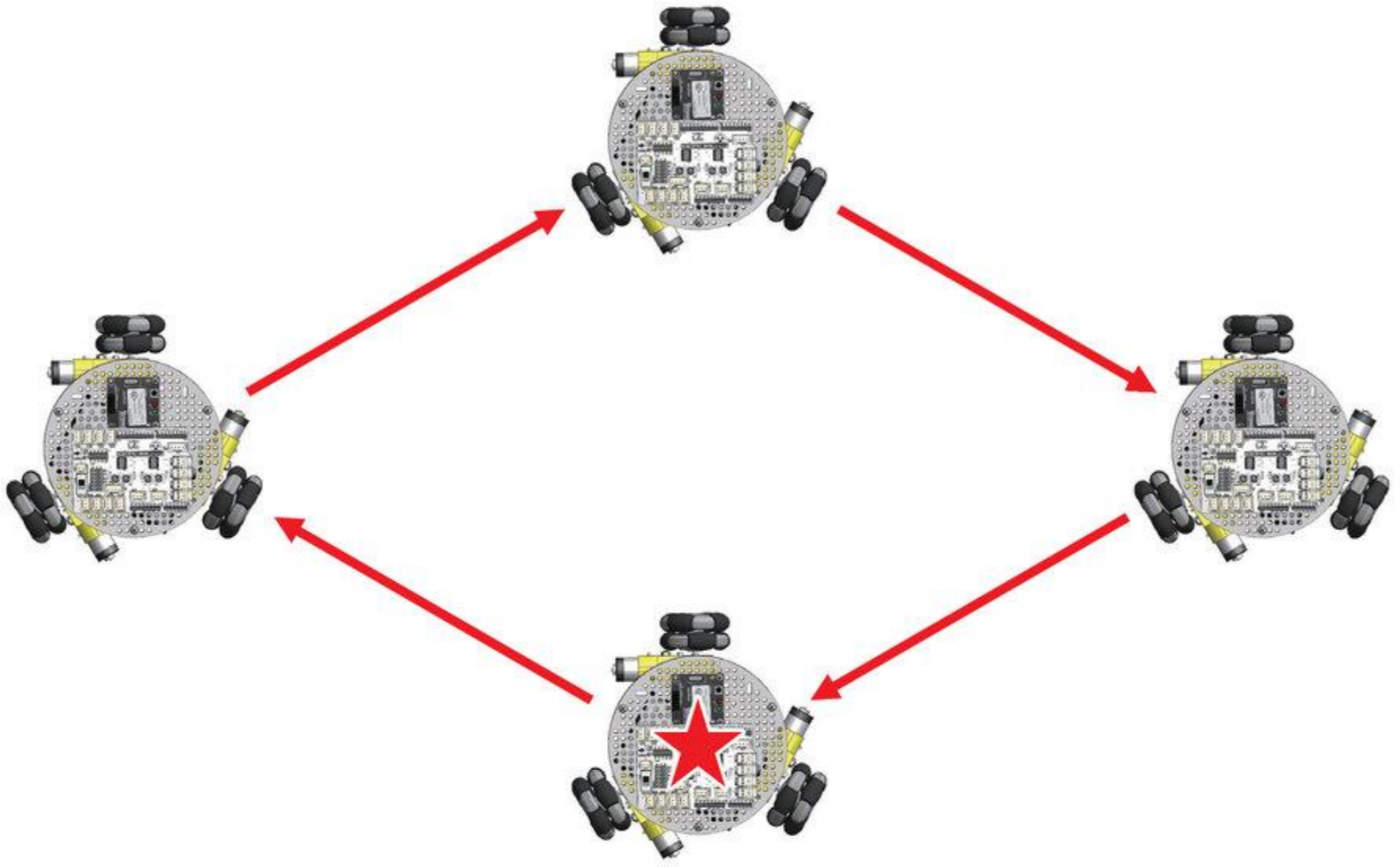


図2-10　ひし形を描くオムニホイールロボット

1周してもとの場所にもどってくるのに、4種類の動きが必要です。
図2-10の星マーク部分をスタート地点とすると、まずは左ななめ前に移動する必要がありますね。
プログラム「Pattern1」が、ちょうど左ななめ前への移動を命令していました。

プログラム「Pattern1」より抜粋（ばっすい）

```
void loop(){
    mc0.rotate(-100);    // 前(mc0)のモーターを回す
    mc1.rotate(0);       // 右(mc1)のモーターを回す
    mc2.rotate(100);     // 左(mc2)のモーターを回す
}
```

`void loop()` という命令文は、波かっこ `{ }` の中に書かれた命令を、電源を切るまでくり返すことを表しています。
つまり、`{ }` の中に他の3種類の命令も書き足していけば、ひし形を描くプログラムが完成しますね！

やってみよう！

プログラム「Pattern1」の `void loop()` の波かっこ内に、ひし形を描くのに必要な動作をすべて書いてみよう！
完成したら、プログラムを実行してロボットがひし形を描くか確認してみよう！

ヒント
「うまく書けたはずなのにロボットが動かない！」というときは次のページに進んでみよう！

講

「動作を○秒続ける」という命令を追加しないと、１つ１つの動作が一瞬で終わってしまいます。
よって `mc.rotate()` をひたすら書くだけでは、ロボットはほとんど移動しません。
次のページ以降で解説される命令をつけ足す必要があります。

2）delay命令

ロボットがひし形を描くには、以下のような4種類の動きが必要です。

```
void loop(){
    mc0.rotate(-100);
    mc1.rotate(0);
    mc2.rotate(100);

    mc0.rotate(100);
    mc1.rotate(-100);
    mc2.rotate(0);

    mc0.rotate(100);
    mc1.rotate(0);
    mc2.rotate(-100);

    mc0.rotate(-100);
    mc1.rotate(100);
    mc2.rotate(0);
}
```

左上

右上

右下

左下

しかし、このプログラムを実行しても、ロボットが動きだすことはありません。
なぜかというと、プログラムを実行しているコンピューター（マイコン）はとてもすばやく計算や処理を行うことができるので、1つ1つの動作を一瞬（いっしゅん）で済ませてしまうからです。
これまでは `void loop()` 内に1種類の動作しか書かれていなかったので、「一瞬（いっしゅん）だけ動作するのをすばやくくり返す」という形になっていました。そのため、人間の目には切れ目なく動き続けているように見えていたのです。

しかし、今回は `void loop()` の中でも動きを切りかえなければなりません。つまり、「この動きを1秒間続けたあと、次の動きに切りかえて…」などと、「動きを続ける時間」を指定してあげる必要があります。

時間を指定するのには、以下の命令を使います。

命　　令「delay」
実行内容：直前の命令を、指定した時間だけ実行し続ける
使 い 方：mc0.rotate(100);
　　delay(1000);　// 直前の命令（モーターの回転）を1秒間続ける

( )の中に入る数字は「秒」ではなく、「ミリ秒」という単位で表す必要があります。
1000ミリ秒で1秒になるので、delay(1000); と書けば1秒間命令を実行し続けます。

4種類の動きの下に delay(); を入れることで、命令を一瞬（いっしゅん）で終えることを防ぐことができますね！

ステップアップ

delay() 命令を使い、ひし形を描くプログラムを完成させよう！
完成したら、delay() の値を好きに書きかえて、動きがどう変わるか確認してみよう！

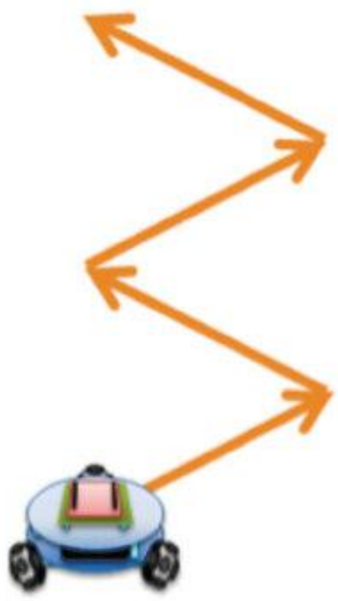
講

解答例は以下のプログラムです。
RoboticsProfessorCourse1 > OmniWheelRobot3 > Square
delay() の数字を大きくすれば１つ１つの動作時間が延びるので、より大きな図形を描くようになります。教室のスペースの広さに応じて適宜調整させて下さい。

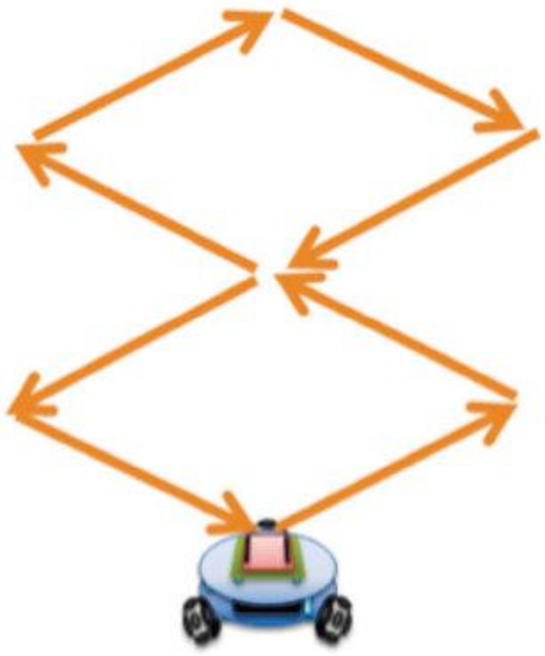
チャレンジ課題

いろいろな動きをプログラミングしてみよう。

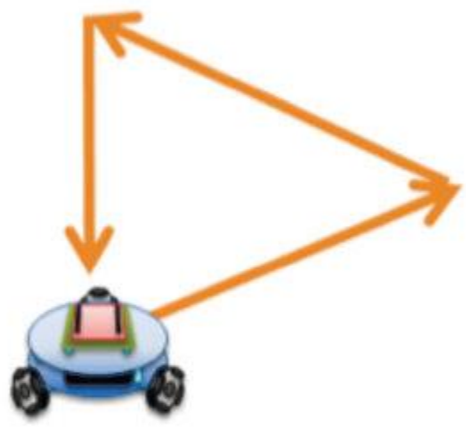
1. ジグザグ走行



2. 8の字走行



3. 三角走行

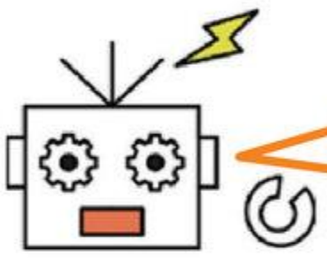


講

早く終わった生徒には、チャレンジ課題に挑戦させてください。
解答例は巻末に記載します。
ロボットを時計回りに回転させる力と、反時計回りに回転させる力が拮抗していない場合、力の合成だけで移動方向を予測することができません。
そういった場合は次回扱う「モーメント」の考え方を併せる必要があります。もし好き好きにプログラムをつくった際に疑問を持つ生徒が居た場合、そのように声掛けしてあげてください。
なお、「モーメント」を学ぶ事で真横への移動も考えられるようになります。

3. まとめ（目安５分）

今回はオムニホイールロボットの動きの仕組みについて学びました。力の合成については理解できましたか？　次回は「モーメント」とよばれるものを学習して、回る動きについて理解を深めます。



オムニホイールロボットの秘密はまだまだア・ル・ヨ！！

講

○　以下の授業の目標を再確認します。
　・オムニホイールロボットの動きを観察する
　・オムニホイールロボットの動きの仕組みを理解する
　・loop 命令と delay 命令を学ぶ
○　今回の授業で学んだ感想や面白かったことなどを、生徒から聞いてみましょう。
○　次回のテーマは「力のモーメントと回転運動」であることを告知します。

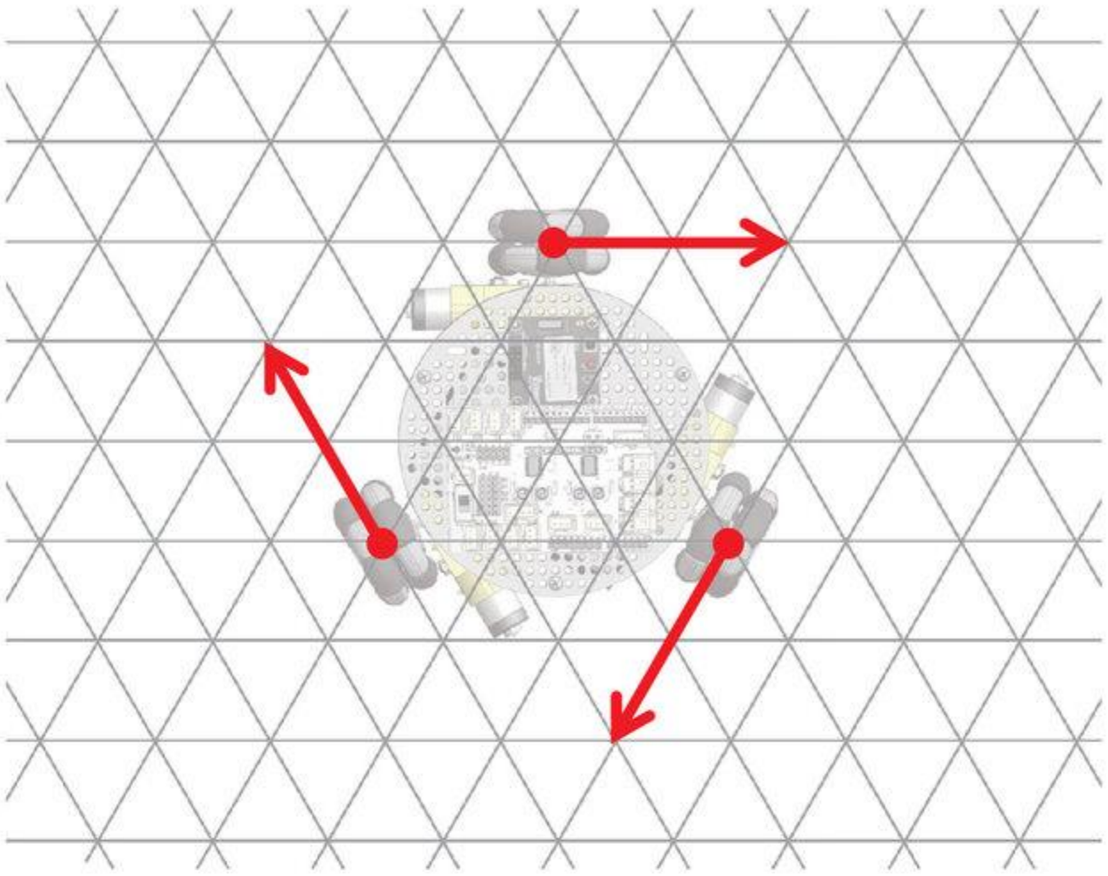
≪次回必要なもの≫

次回は、今回使ったロボットと以下のパーツを持ってきてください。

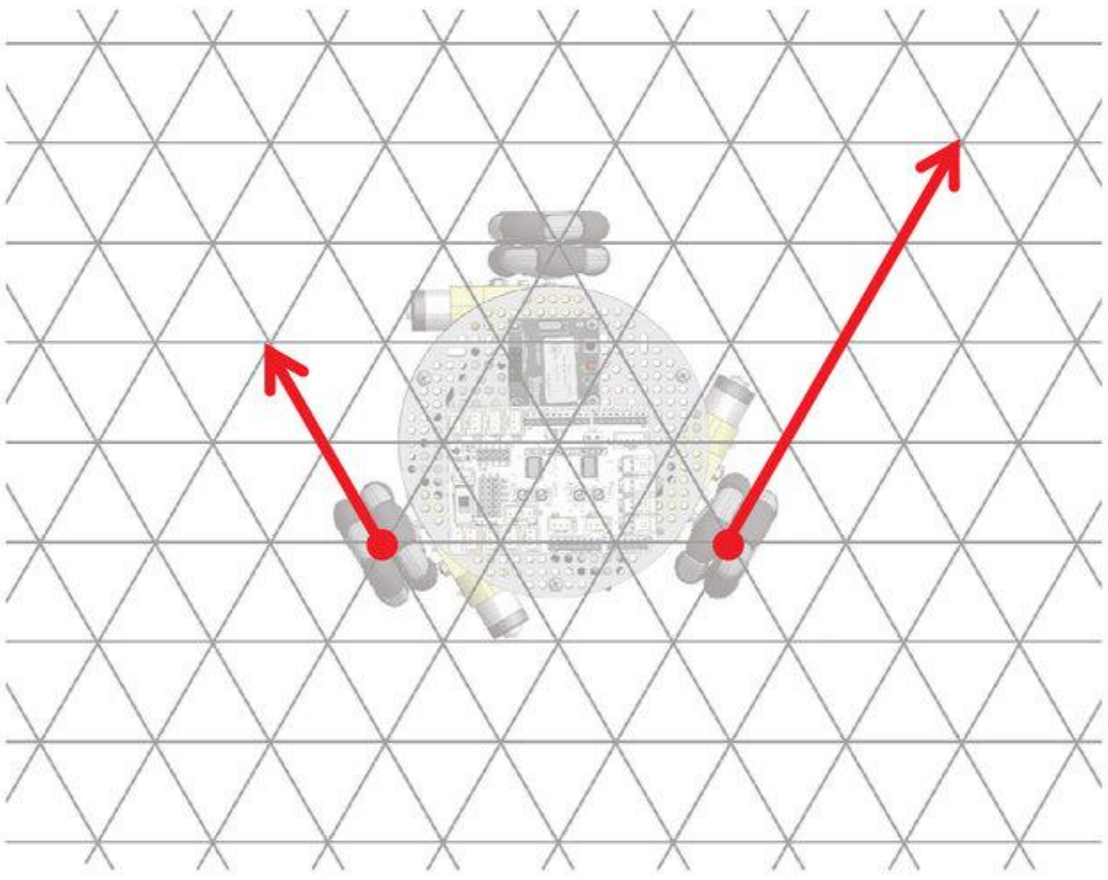
図3-0　次回必要なもの

P.8　やってみよう！　解答例

1.

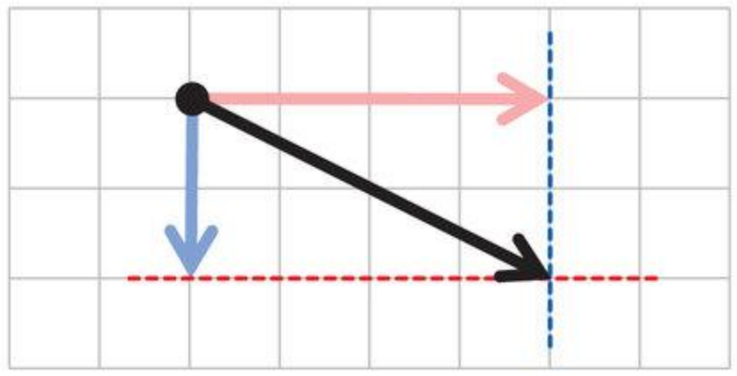


2.

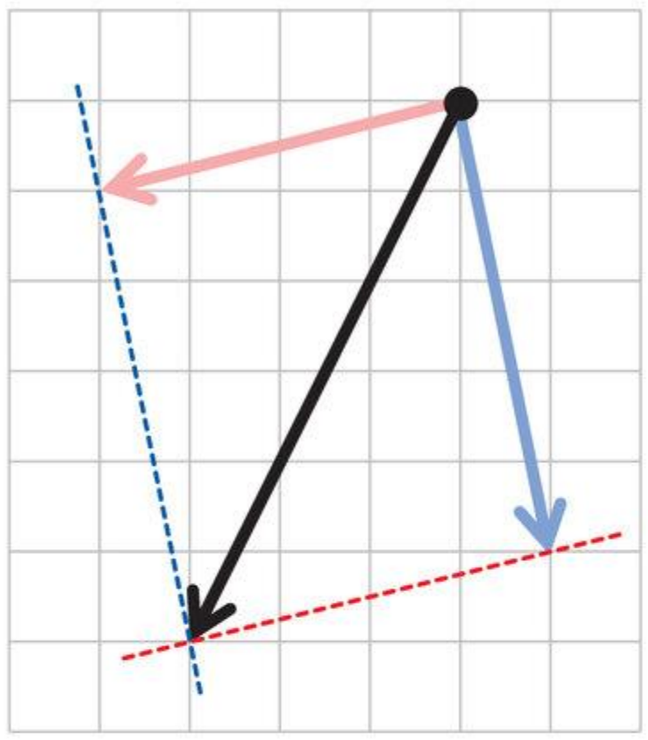


P.10　やってみよう！　解答例

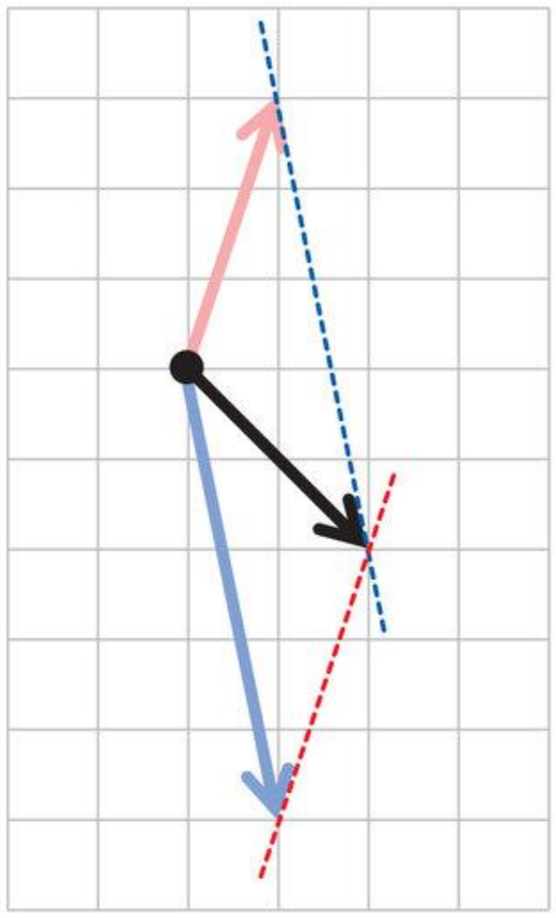
①



②

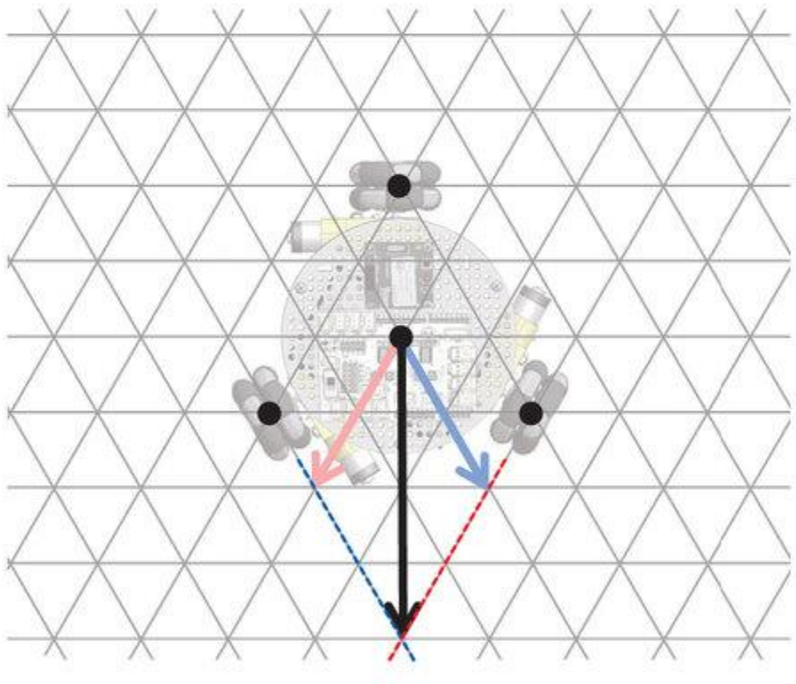


③

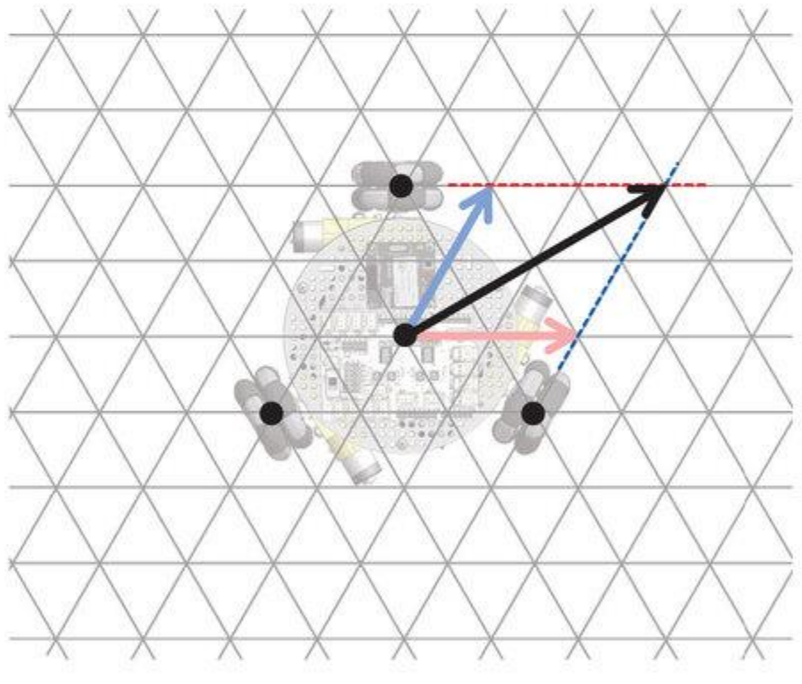


P.12　やってみよう！　解答例

①



②



P.19　チャレンジ課題1　解答例

```
#include <RPlib.h>

// ピン接続設定
RPmotor mc0(MC0);
RPmotor mc1(MC1);
RPmotor mc2(MC2);

void setup(){
}

void loop(){
    mc0.rotate(100);  // 前（mc0）のモーターを回す
    mc1.rotate(-100); // 右（mc1）のモーターを回す
    mc2.rotate(0);    // 左（mc2）のモーターを回す
    delay(1000);      // 1秒（1000ミリ秒）の間待つ

    mc0.rotate(-100); // 前（mc0）のモーターを回す
    mc1.rotate(0);    // 右（mc1）のモーターを回す
    mc2.rotate(100);  // 左（mc2）のモーターを回す
    delay(1000);      // 1秒（1000ミリ秒）の間待つ
}
```

P.19　チャレンジ課題2　解答例

```
#include <RPlib.h>

// ピン接続設定
RPmotor mc0(MC0);
RPmotor mc1(MC1);
RPmotor mc2(MC2);

void setup(){
}

void loop(){
    mc0.rotate(100);  // 前（mc0）のモーターを回す
    mc1.rotate(-100); // 右（mc1）のモーターを回す
    mc2.rotate(0);    // 左（mc2）のモーターを回す
    delay(1000);      // 1秒（1000ミリ秒）の間待つ

    mc0.rotate(-100); // 前（mc0）のモーターを回す
    mc1.rotate(0);    // 右（mc1）のモーターを回す
    mc2.rotate(100);  // 左（mc2）のモーターを回す
    delay(2000);      // 2秒（2000ミリ秒）の間待つ

    mc0.rotate(100);  // 前（mc0）のモーターを回す
    mc1.rotate(-100); // 右（mc1）のモーターを回す
    mc2.rotate(0);    // 左（mc2）のモーターを回す
    delay(1000);      // 1秒（1000ミリ秒）の間待つ

    mc0.rotate(100);  // 前（mc0）のモーターを回す
    mc1.rotate(0);    // 右（mc1）のモーターを回す
    mc2.rotate(-100); // 左（mc2）のモーターを回す
    delay(1000);      // 1秒（1000ミリ秒）の間待つ

    mc0.rotate(-100); // 前（mc0）のモーターを回す
    mc1.rotate(100);  // 右（mc1）のモーターを回す
    mc2.rotate(0);    // 左（mc2）のモーターを回す
    delay(2000);      // 2秒（2000ミリ秒）の間待つ

    mc0.rotate(100);  // 前（mc0）のモーターを回す
    mc1.rotate(0);    // 右（mc1）のモーターを回す
    mc2.rotate(-100); // 左（mc2）のモーターを回す
    delay(1000);      // 1秒（1000ミリ秒）の間待つ
}
```

P.19　チャレンジ課題3　解答例

```
#include <RPlib.h>

// ピン接続設定
RPmotor mc0(MC0);
RPmotor mc1(MC1);
RPmotor mc2(MC2);

void setup(){
}

void loop(){
    mc0.rotate(100);  // 前（mc0）のモーターを回す
    mc1.rotate(-100); // 右（mc1）のモーターを回す
    mc2.rotate(0);    // 左（mc2）のモーターを回す
    delay(1000);      // 1秒（1000ミリ秒）の間待つ

    mc0.rotate(-100); // 前（mc0）のモーターを回す
    mc1.rotate(0);    // 右（mc1）のモーターを回す
    mc2.rotate(100);  // 左（mc2）のモーターを回す
    delay(1000);      // 1秒（1000ミリ秒）の間待つ

    mc0.rotate(0);    // 前（mc0）のモーターを回す
    mc1.rotate(100);  // 右（mc1）のモーターを回す
    mc2.rotate(-100); // 左（mc2）のモーターを回す
    delay(1000);      // 1秒（1000ミリ秒）の間待つ
}
```