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

リンクロボット②

第4回

リンクロボットを自動で走行させよう

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。

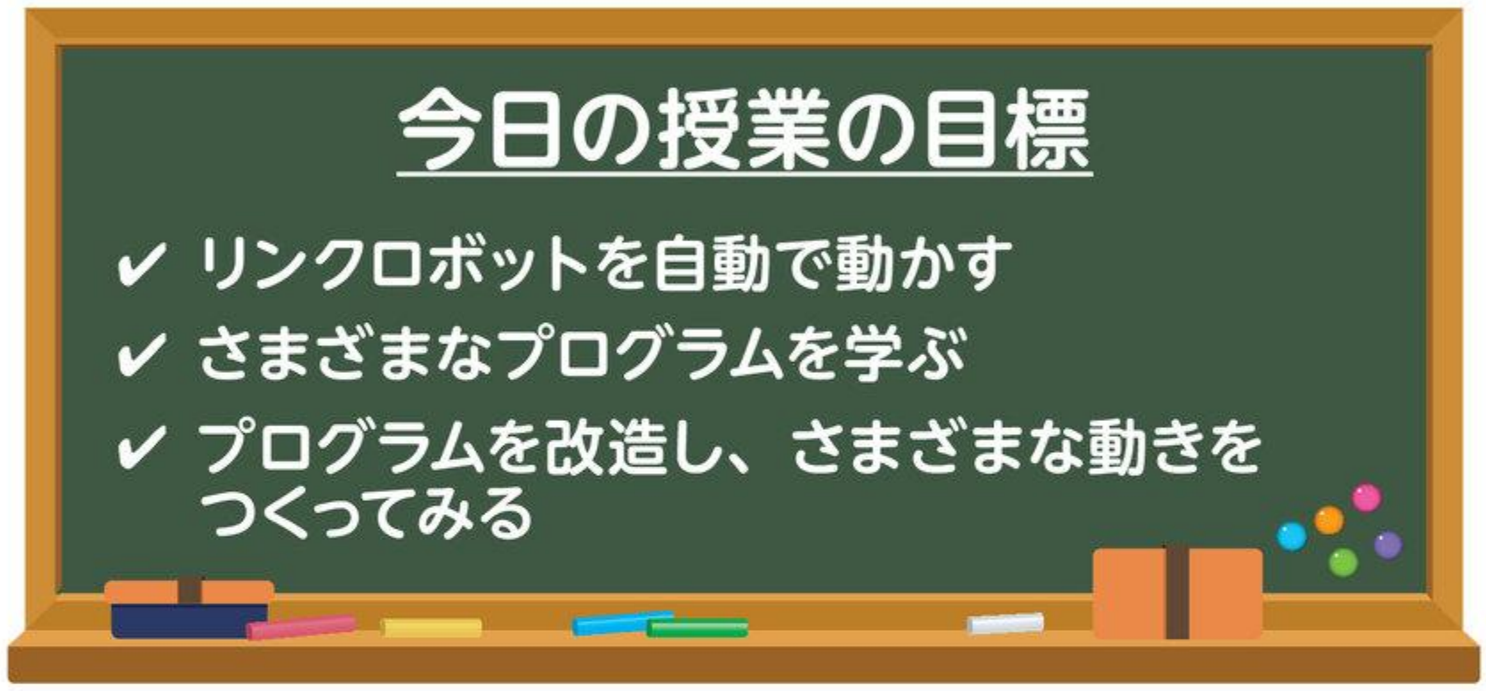
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（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. リンクロボットを自動で走行させよう（目安 5 分）

0.0. 「リンクロボットを自動で走行させよう」でやること



今回の授業では、前回コントローラーで操縦（そうじゅう）したリンクロボットを、自動で動かします。そこで活用するのがプログラムです。プログラムについても改造しながら学んでいき、最後は好きな動きをつくってみましょう！

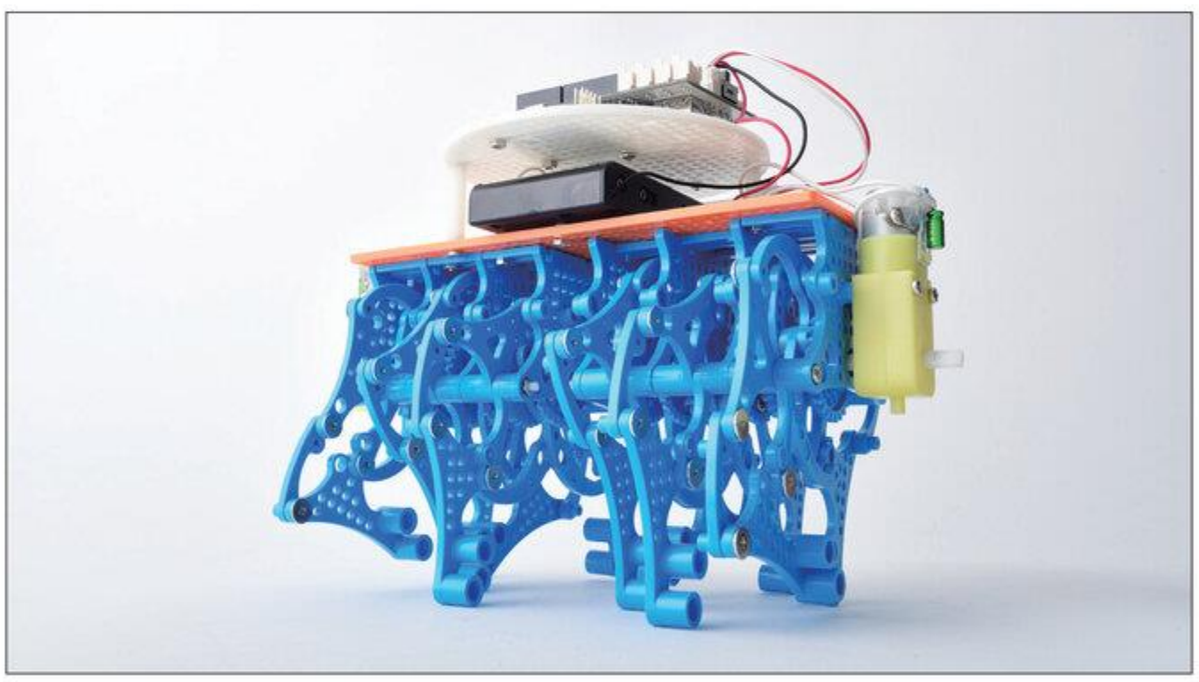
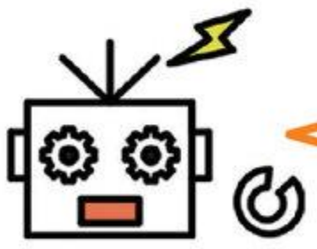


図0-0　リンクロボット本体



自動走行も可能サ。そう、プログラムならネ！

0.1. 必要なもの

前回つくったリンクロボット本体と、以下のパーツを準備しておきましょう。コントローラーは単4電池を3本使用します。あらかじめセットしておきましょう。

USBケーブル 1	コントローラー 1
	

図0-1　必要なもの

1．左右の脚の動きからロボット本体の動きを予測する（目安 45 分）

前回、リンクロボットがどのような動きができるのかを、実際に操縦してみておおよそ理解できたと思います。今回はもう少しくわしく、左右の脚の動きとロボット本体の動きの関係について理解を深めていきましょう。

1.0. リンクロボットの脚の動き

1）　左右のアナログスティックと脚の動き

前回使ったプログラム「LinkRemote」では、**図1-0**のように、コントローラーの左右のアナログスティックの操作が、リンクロボットの脚の動きに対応していました。たとえば、左スティックを倒すと左脚が、右スティックを倒すと右脚が、それぞれ動いたはずです。
左右のスティックを同時に上に倒すとロボット本体は直進し、下に倒せば後退しました。

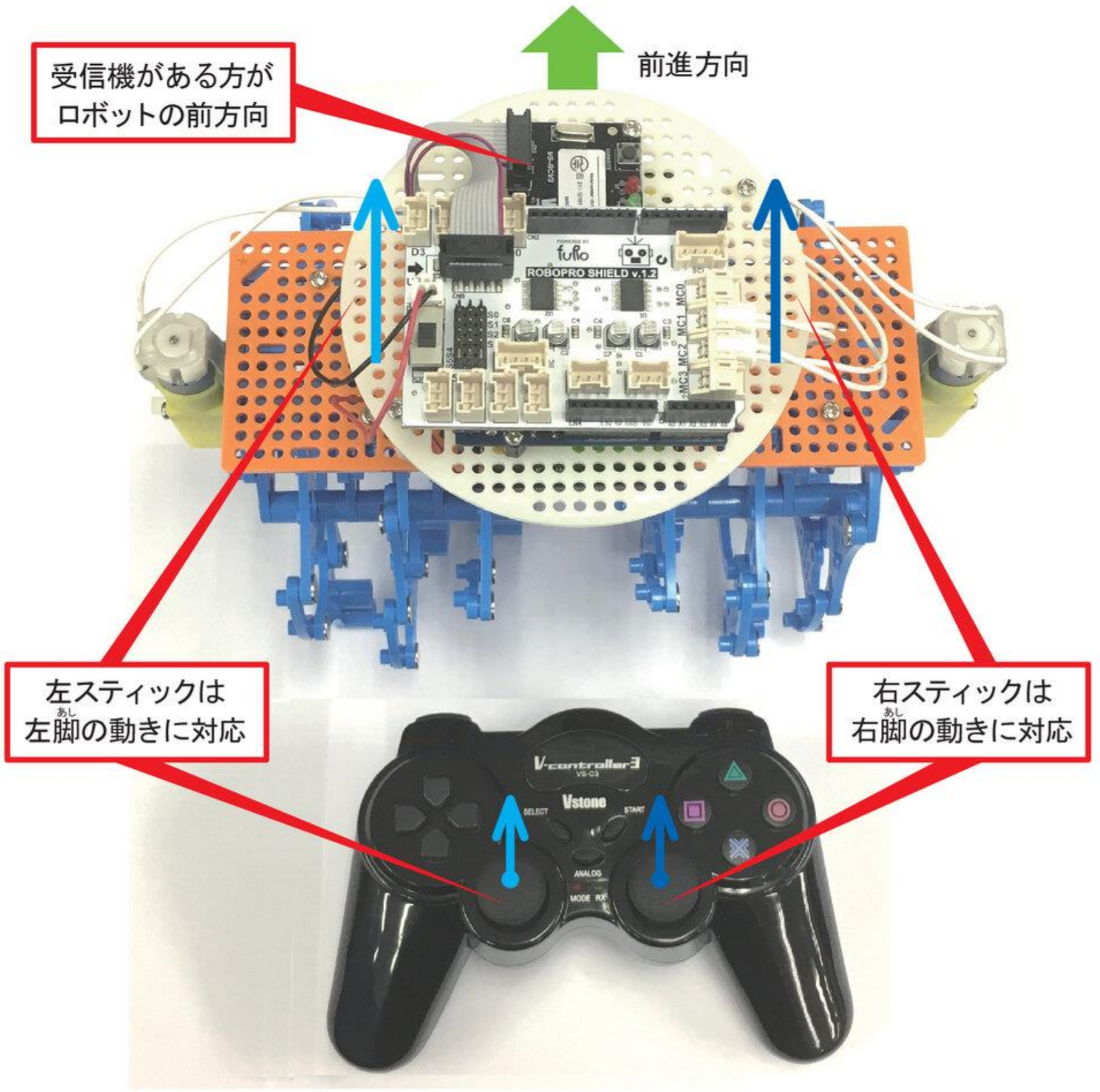


図1-0　左右アナログスティックと脚の動き

2）　モーターと脚の動き

リンクロボットでは、ギアドモーターの軸が1回転する間に、６本の脚先で地面をけって進みます。このとき、脚先が地面をける向きは前か後ろのみで、ななめなどに力を出すことはできません。つまり、リンクロボットでは、左右の脚の動き（向きと速さ）だけでロボット本体の動きが決まっているのです。

1.1.　左右の脚の動きとロボット本体の動き

では、左右の脚の動き（向きと速さ）をかえた場合、ロボット本体の動きはどうなるのでしょうか？

まずはリンクロボットの動きがわかりやすくなるように、左右の脚の動き、ロボット全体の動きをそれぞれ矢印で表すことにしましょう。**図1-1**のように、左右の脚の動きはオレンジの矢印で、ロボット全体の動きは赤の矢印で示します。左右の脚を同じ向き、同じ速さで動かすことで、ロボットが前方に直進していることがよくわかりますね。

このように「左右の脚が同じ向きに同じ速さで動く場合は、ロボット本体は直進する」、ということは直感的に理解できると思います。それでは、向きと速さが異なる場合はどうでしょうか？

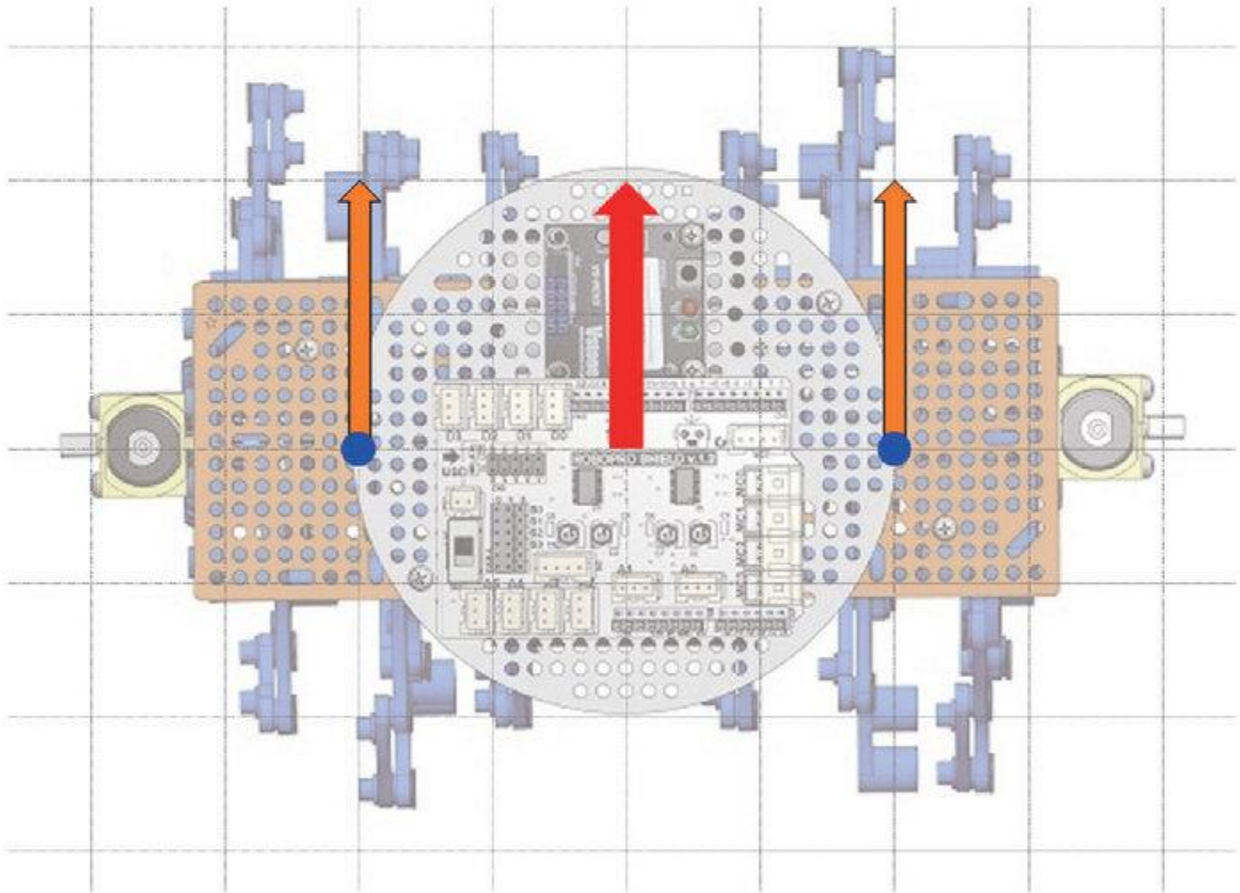


図1-1　リンクロボットの脚の動き

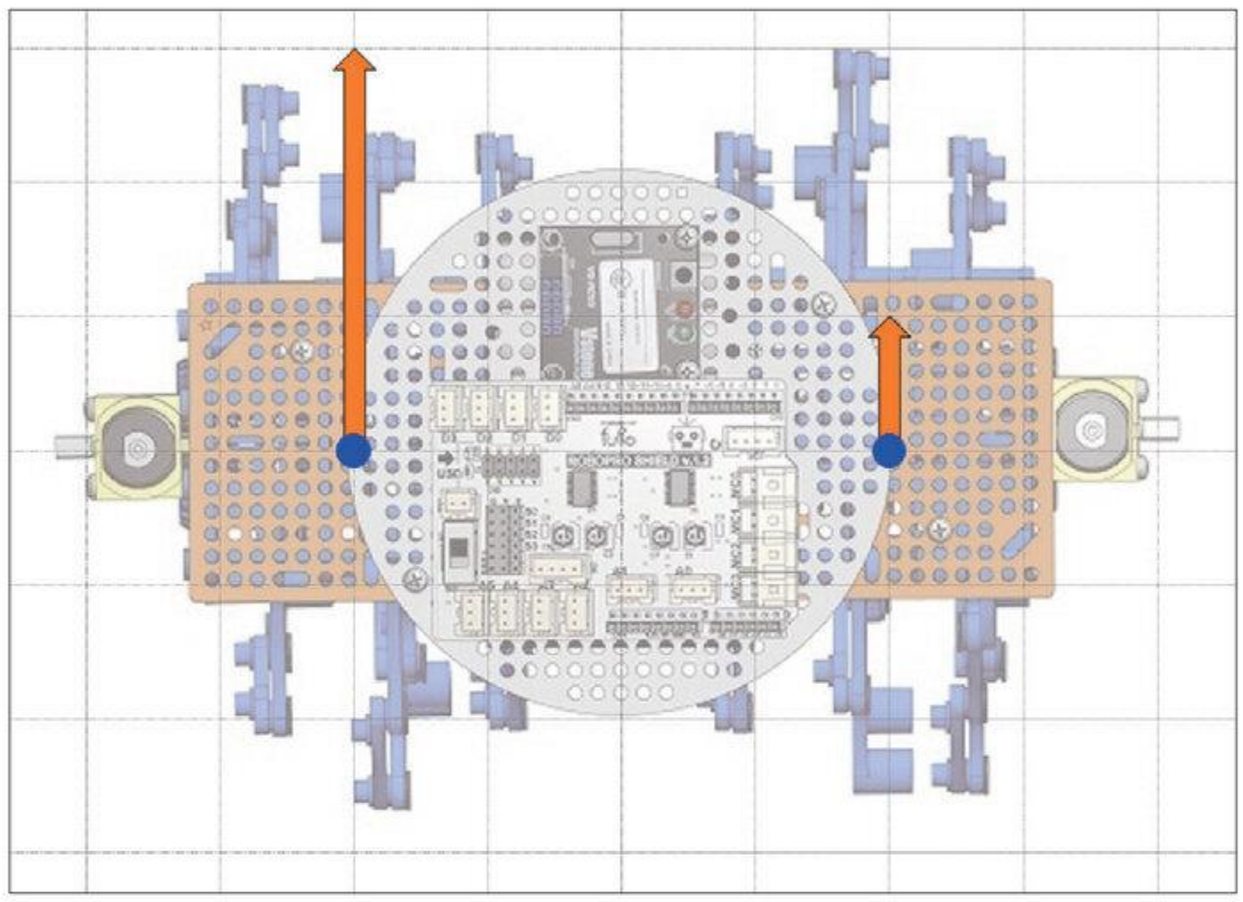
1.2. ロボット本体の動きを予測する

さて、左右の脚(あし)の動きに差をつけた場合を考えてみましょう。

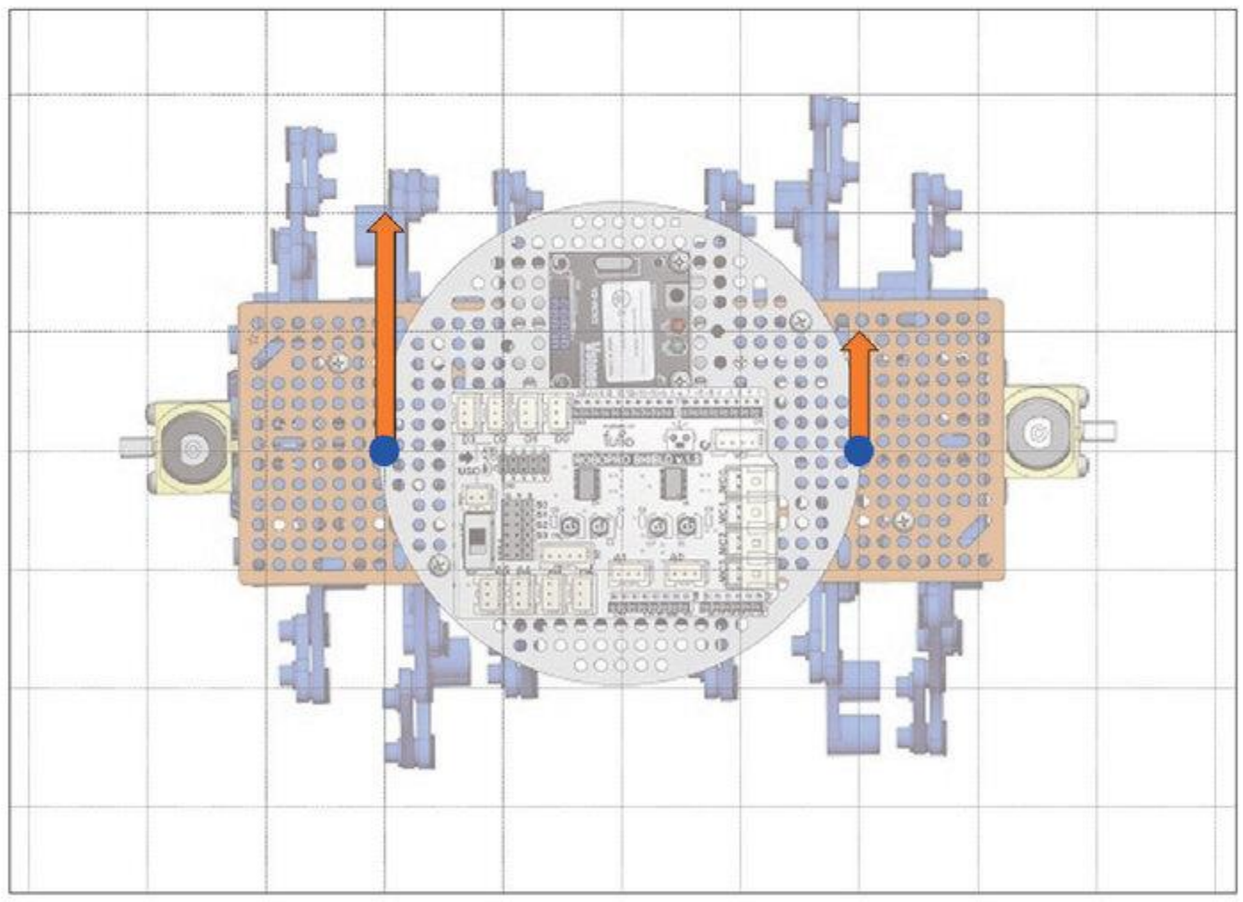
やってみよう！

次の1.と2.それぞれのとき、ロボット本体はどのように進んでいくかな？　図に矢印をかき込(こ)んでみよう！　長さや向きはだいたいで大丈夫だよ！

1.

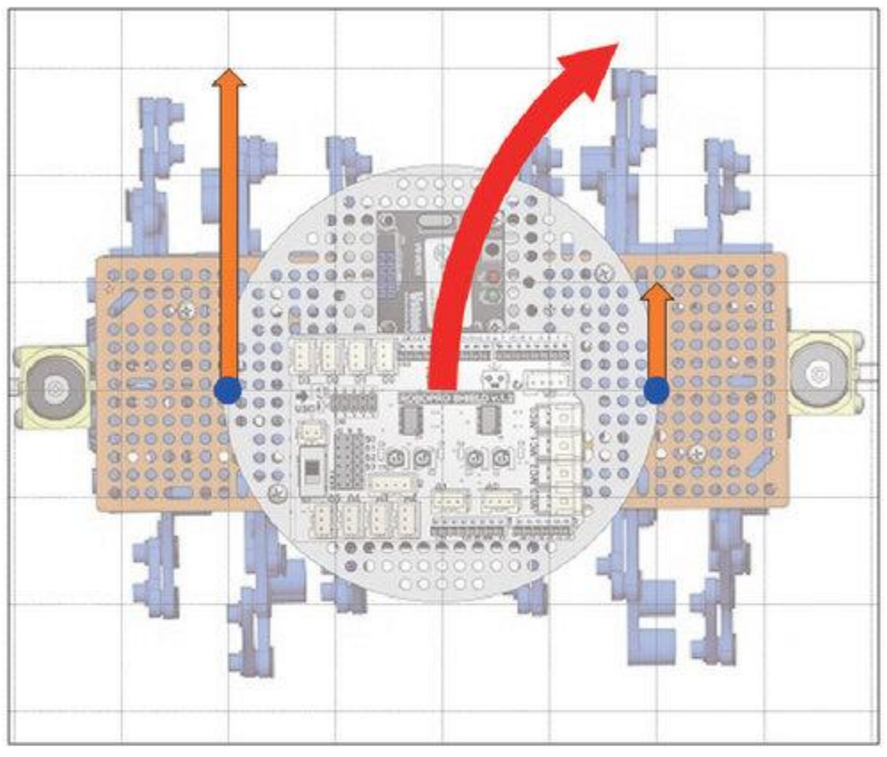


2.



予想はうまくできましたか？　答えは次のようになります。

1.



2.

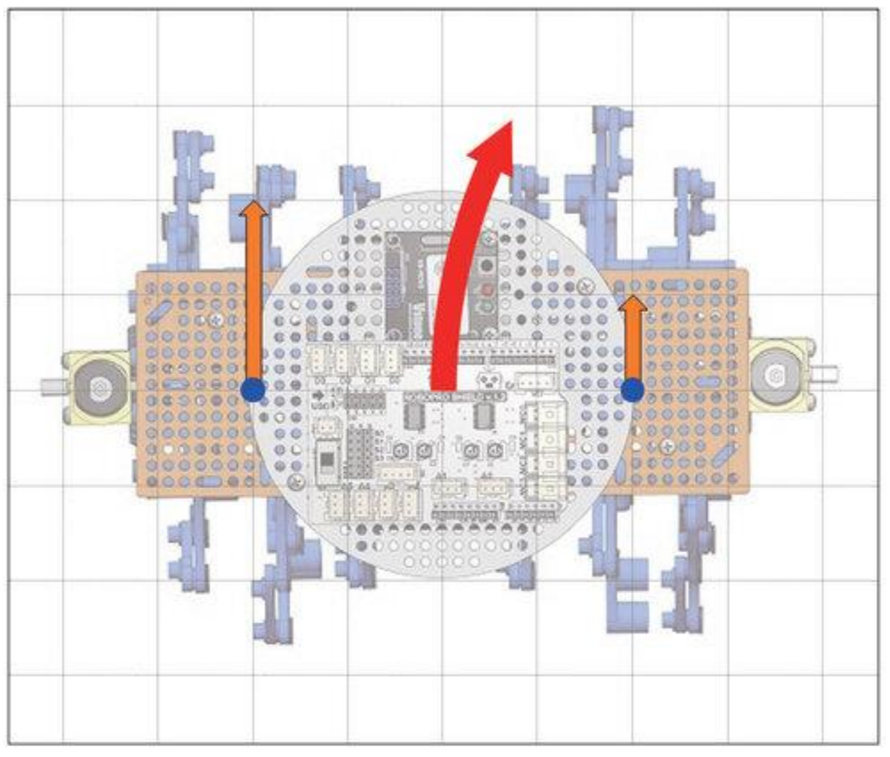


図1-2　左右の脚の速さに差があるときのロボットの動き

どちらも、右に向かって円をえがくようにカーブしていきます。ただし、左右の脚の速さに差があるほど急カーブになります。つまり、えがく円の中心と半径が変化していくのです。
えがく円の中心がどこにくるかは、以下の図のように「左右の脚を通る直線」と「2本の矢印の先端を通る直線」を引くとわかります。2本の直線が交わるところが、ロボットがえがく円の中心（回転中心）になるのです。

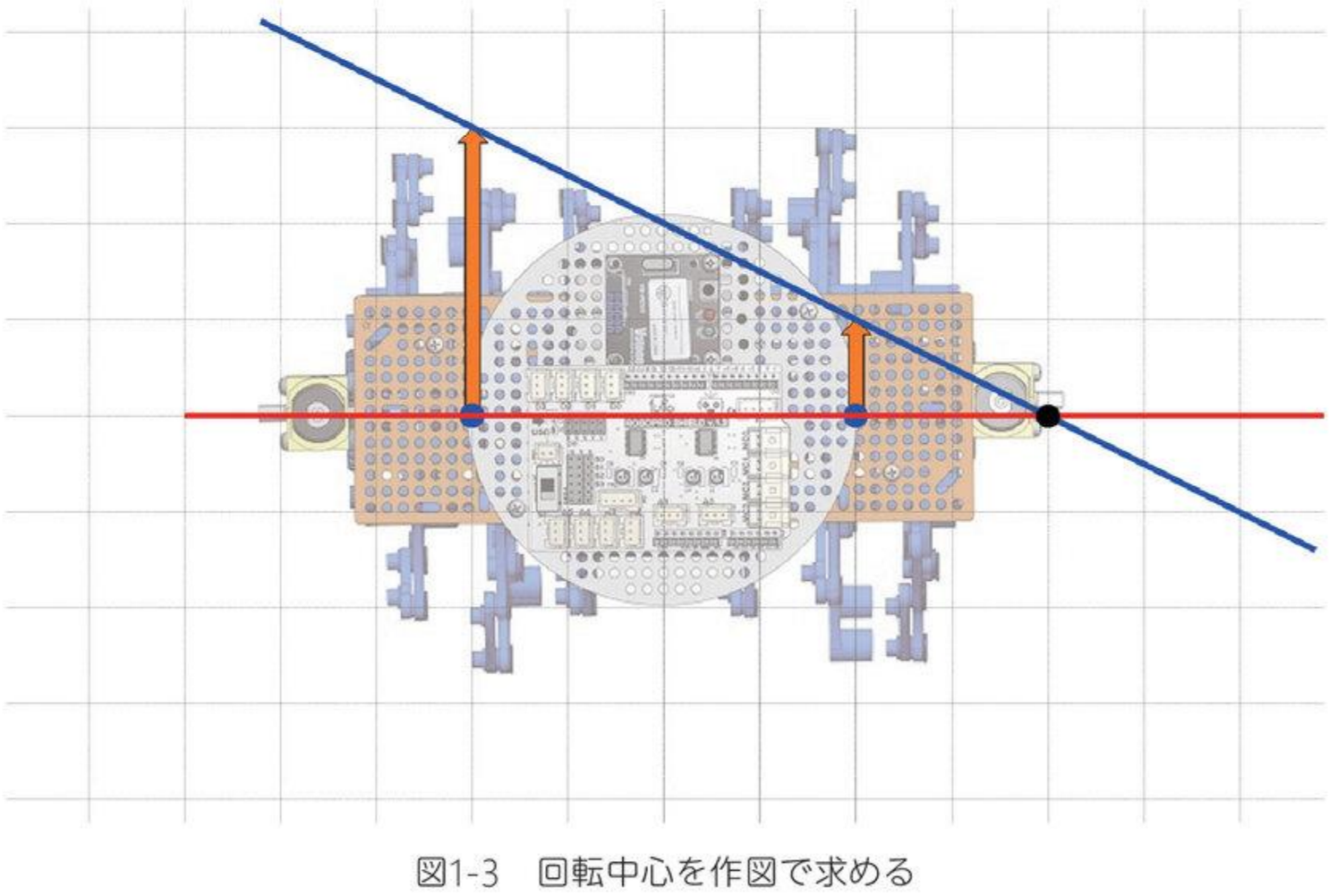
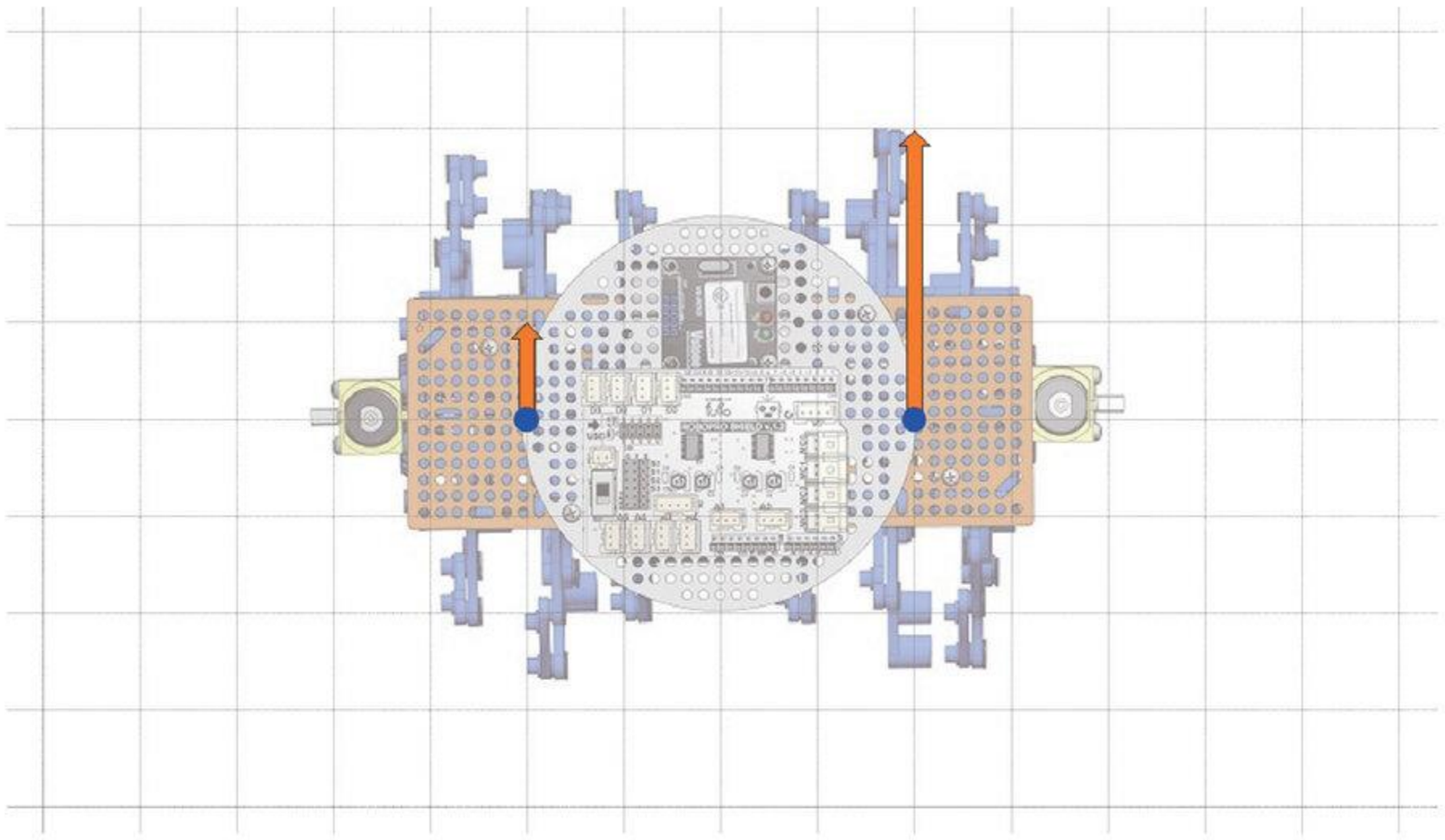


図1-3　回転中心を作図で求める

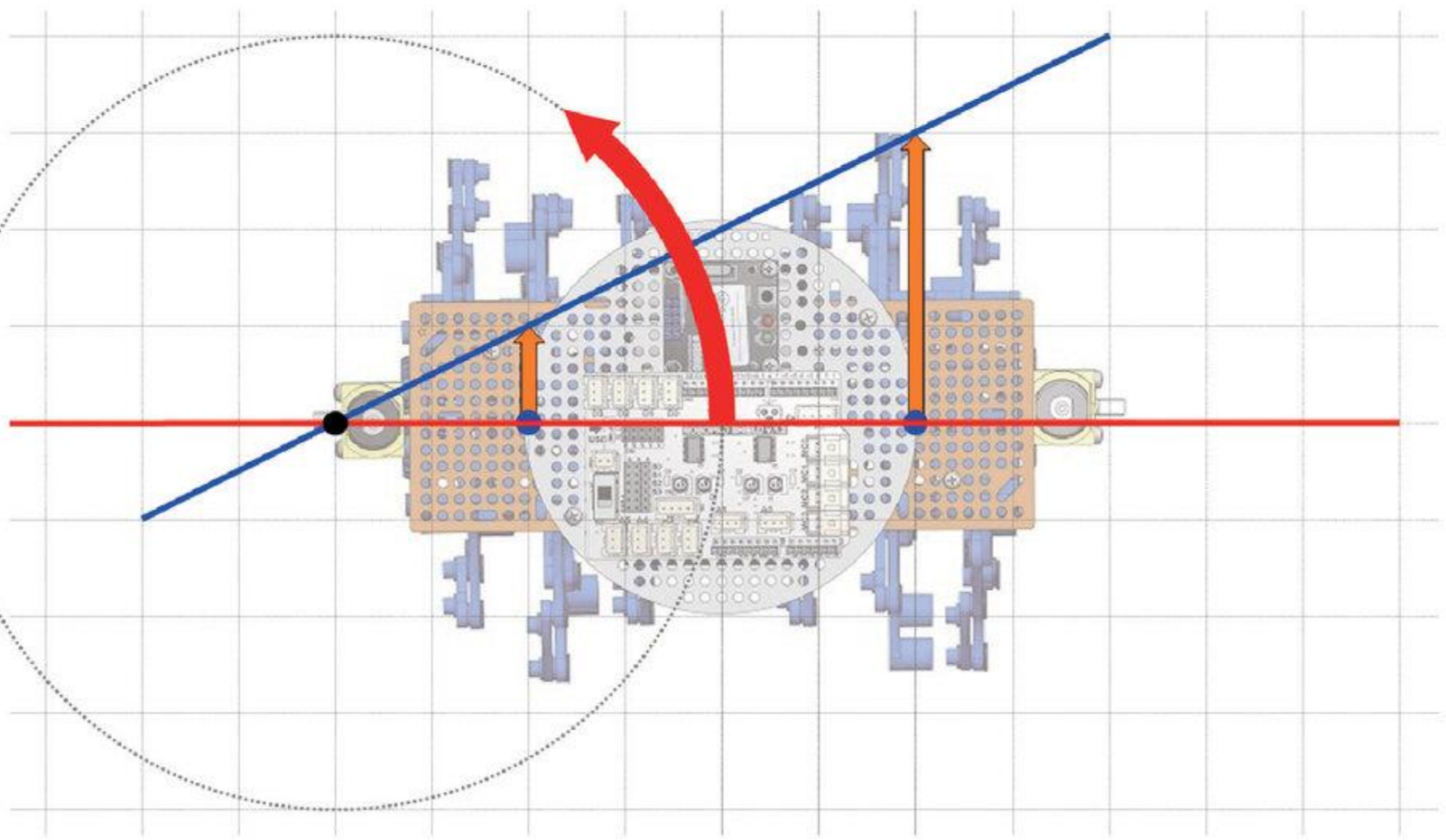
やってみよう！

図1-3のやり方で回転中心を求め、ロボット本体がえがく円をえがいてみよう！　フリーハンドで大丈夫だよ！

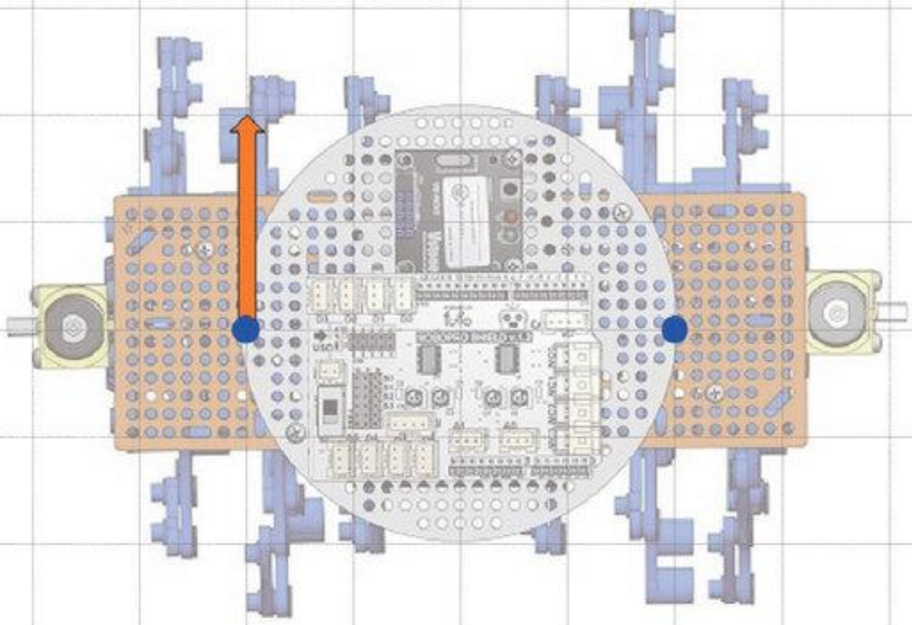
1.

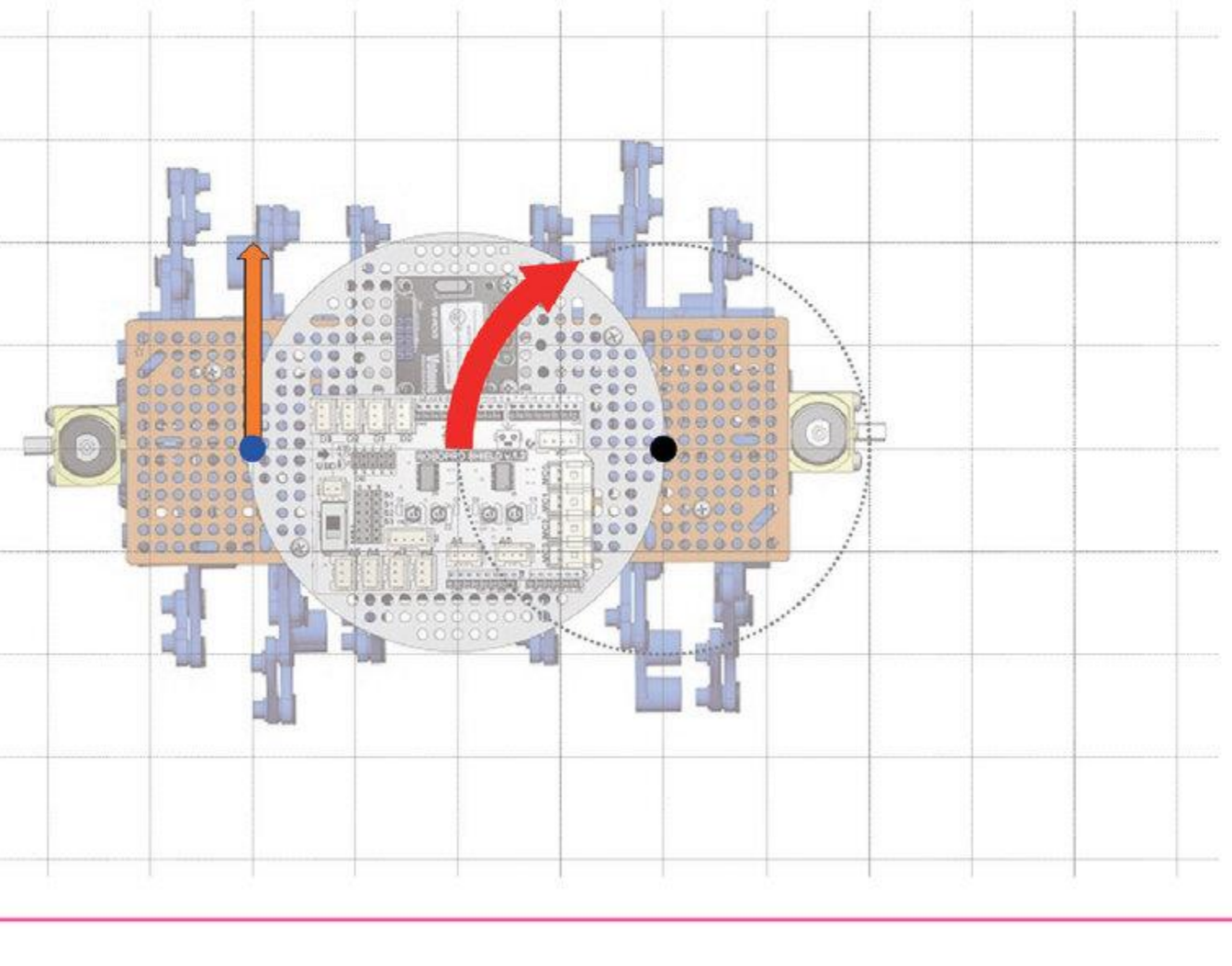


講



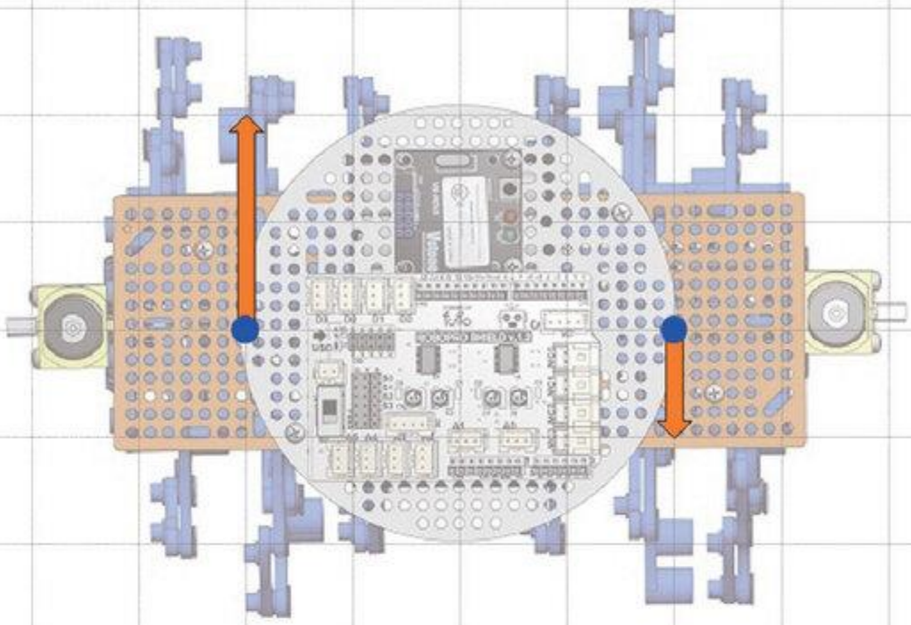
2.

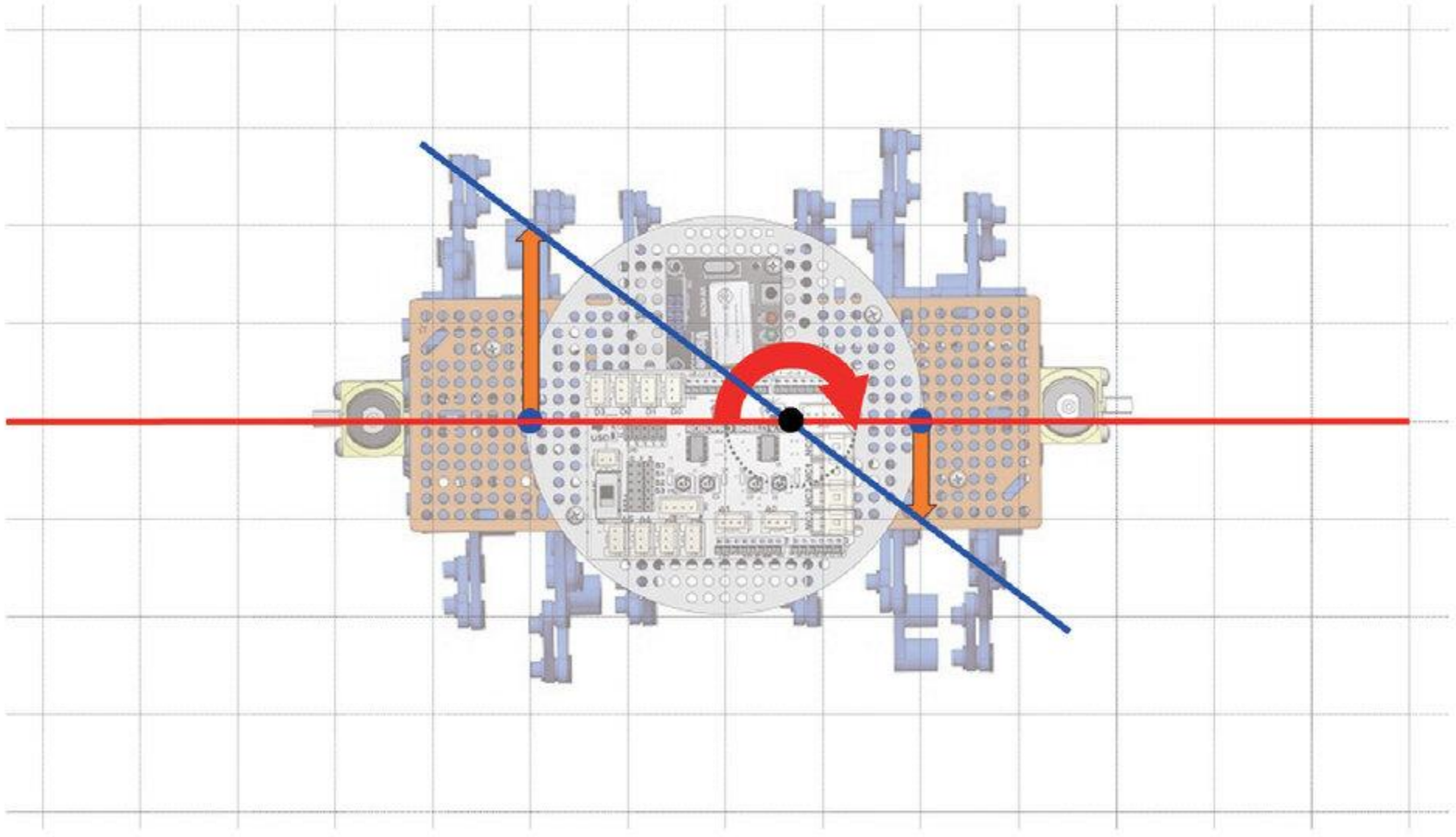




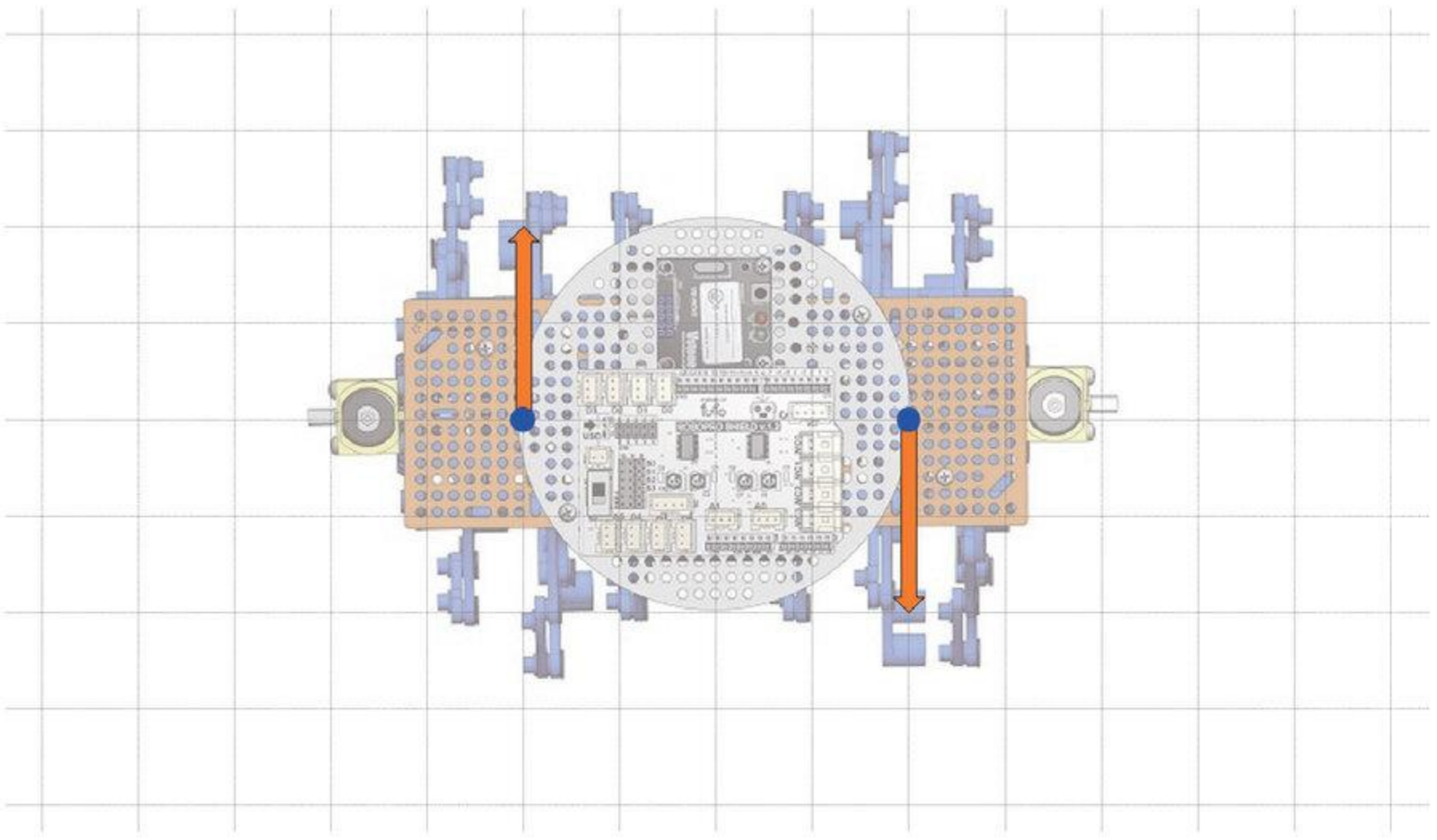
講

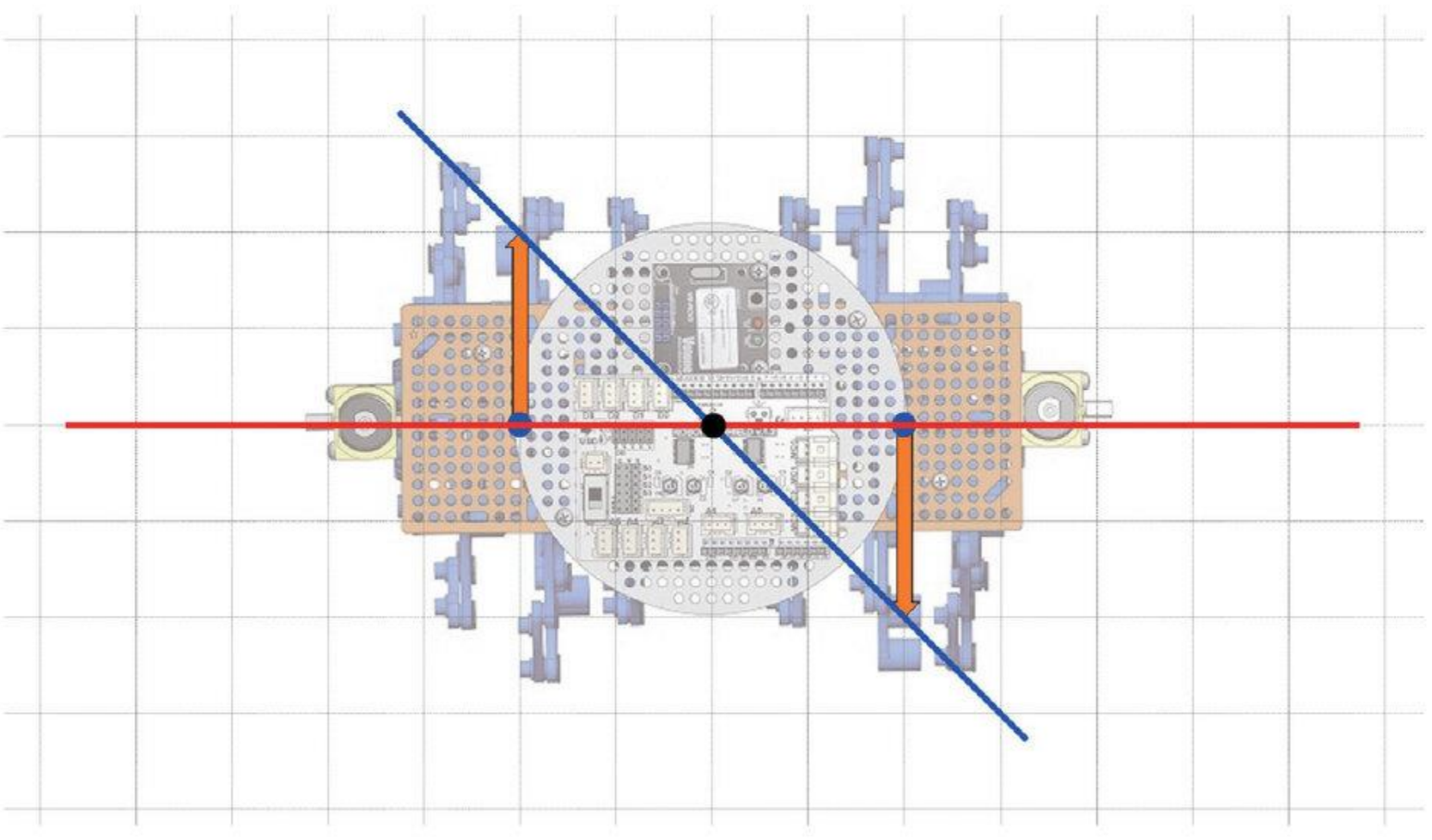
3.





4.





講

コラム　交通標識の「R=XXX」表記

交通標識を見ていると、右の図のように、カーブを示す標識の下に「R=200」などとかかれているものがあります。この「R=XXX」は、「この先に半径XXX[m]のカーブがありますよ」という意味で、道のカーブ具合を示しています。もちろん、数字が小さい方がより急カーブということになります。ちなみに、Rは半径の英単語「Radius」の頭文字をとったものです。実は、自動車もハンドル操作(そうさ)によって、回転半径をかえてカーブ走行をしています。４輪の自動車では、なめらかに曲がるために、「アッカーマン・ステアリング」とよばれるリンク機構が用いられています。興味のある人は調べてみるとよいでしょう。

2．リンクロボットの自動走行（目安 60 分）

2.0.　モーターを動かすプログラム

1）プログラムの実行

リンクロボットの動きについてだいたい理解できたところで、今度は、自動で動かすことにチャレンジしましょう。その前に、動作確認もかねて、基本となるプログラムを見ておきましょう。
まずは、以下のプログラムを実行してください。
なお、実行後、すぐに動き出すので脚（あし）を上に向けるようにしておきましょう。

プログラムの書き込み
RoboticsprofessorCourse1 ＞ LinkRobot3 ＞ MotorTest

動作確認を終えたら、プログラムの以下の赤枠の部分を見てみましょう。このプログラムでは、以下の①と②の部分でモーターを動かすための命令を出しています。

プログラム「MotorTest」より抜粋（ばっすい）

```
//おまじない(ピン接続設定)
RPmotor mc1(MC1);     //MC1に繋がっているモーターを指定する
RPmotor mc2(MC2);     //MC2に繋がっているモーターを指定する

void setup(){
}

void loop(){
  mc1.rotate(100);     //100の速度で正方向に回す(範囲は-255から255)
  mc2.rotate(100);     //100の速度で正方向に回す(範囲は-255から255)
}
```

①

②

2）動かすモーターを指定する命令

まず、①の部分を見てみると RPmotor mc1(MC1); RPmotor mc2(MC2); とかかれています。すでに、リンクロボットを組み立てる際に動かしているのでわかるかもしれませんが、この MC1 や MC2 がロボプロシールドのコネクターの記号にあたります。この記号を MC0 などと変更を加えると、動かすモーターをかえることができます。

3）モーターを動かす命令

今度は、②の部分を見てみます。

mc1.rotate(100); mc2.rotate(100); とかかれています。

mc1.rotate(); とは、mc1 のモーターを回転させる命令です。( ) 内の数字でモーターの速度を決めていて、プラスとマイナスの符号で回転させる向きを指定しています。

なお、変更できる範囲は、「-255 ～ 255」です。

では、プログラムが理解できたところで、実際に数値を変更したうえで再びプログラムを実行し、どのようにロボットの動きが変化するのかを確かめてみましょう。

やってみよう！

プログラム「MotorTest」の mc1.rotate(100); 、mc2.rotate(100); の 100 の部分を、「-255 ～ 255」の範囲で変更して、脚の動きの変化を観察しよう。

さらに、リンクロボットが前進するようにプログラムをかえてみよう。

講

前進するプログラムの解答例は以下です。

```
mc1.rotate(100);
mc2.rotate(-100);
```

「mc1.rotate() 」と「mc2.rotate() 」の「()」内の数値をそれぞれプラスとマイナスに変更します。ただし、モーターのつき方やコネクターの差し込み位置がテキストと異なる場合は、この解答で正しく動作するとは限りません。また、プログラムのかき方によっては、ほかのやり方も存在しますので、基本的な模範解答だと考えてください。

2.1. 自動走行プログラム

続いて、ロボットに「前進」と「後退」という2つの動作をさせてみましょう。プログラム「MotorTest」のvoid loop()内を以下のように変更(へんこう)してみましょう。

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

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

実行結果：ロボットが小刻(こきざ)みに振動(しんどう)するか、まったく動かない。

mc1.rotate(100); と mc2.rotate(-100); は前進、mc1.rotate(-100); と mc2.rotate(100); は後退の命令ですよね。正しい命令がかけているはずなのに、思った通りの動きをしてくれません。

この問題を解決するには、さらに以下の2行を追加します。

```
void loop(){
    mc1.rotate(100);
    mc2.rotate(-100);
    delay(1000);

    mc1.rotate(-100);
    mc2.rotate(100);
    delay(1000);
}
```

delay(1000); とは、「直前の命令を1,000ミリ秒間（1秒間）続ける」という命令です。delay( ); がないとロボットは命令された動作を一瞬(いっしゅん)で終わらせてしまうため、このように処理(しょり)を続ける時間を指示する必要があるのです。

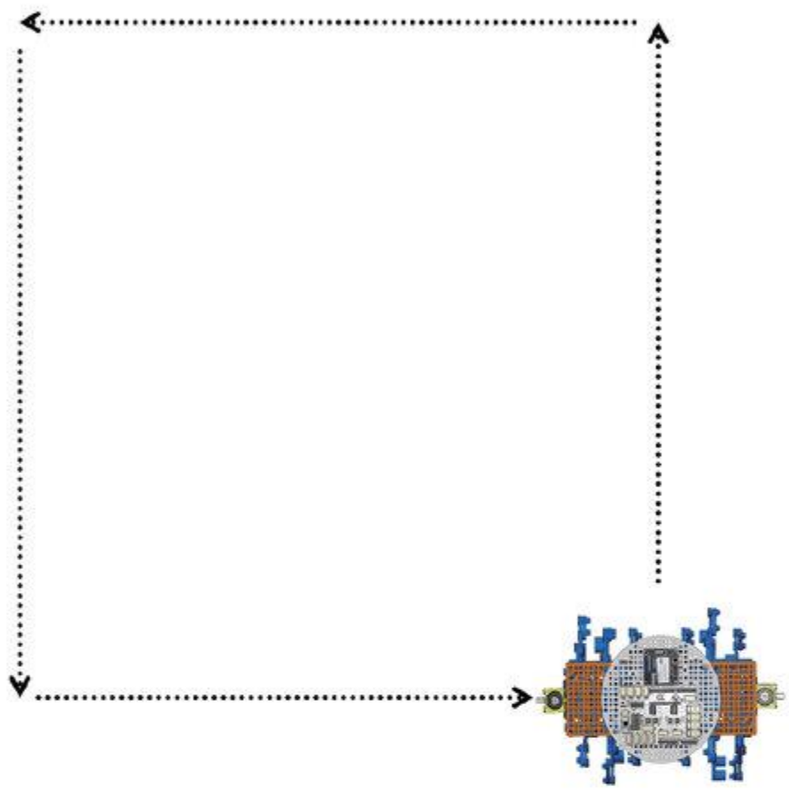
なお、「ミリ秒」とは「秒」をさらに1,000分割した時間の単位です。1,000ミリ秒で1秒、2,000ミリ秒で2秒です。

では、先ほどの命令をうまく組み合わせて、次のような動きをつくってみましょう。

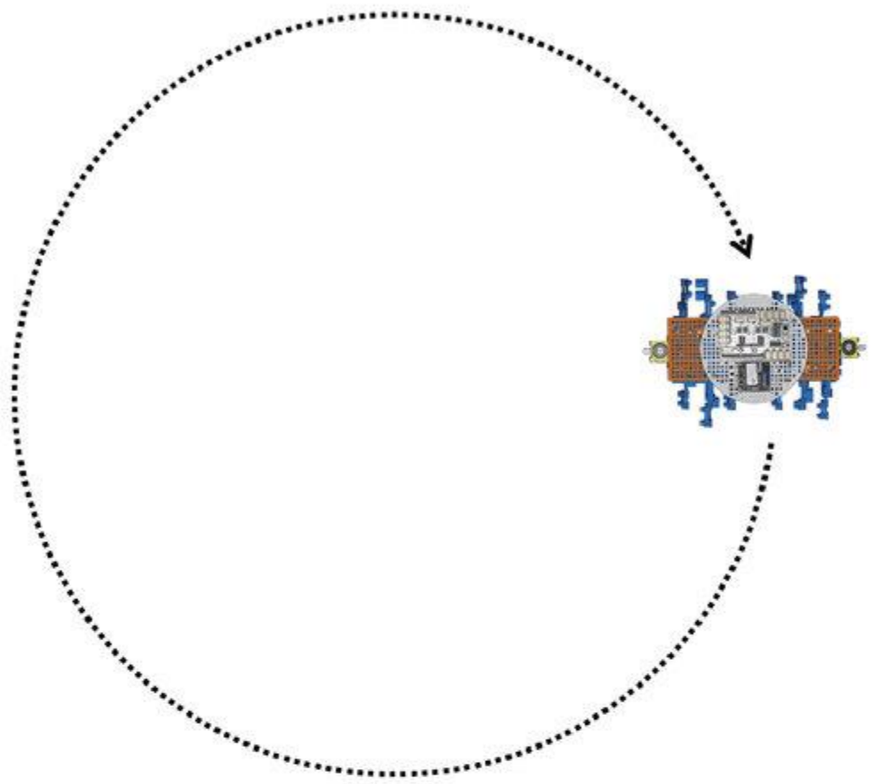
やってみよう！

プログラム「MotorTest」をかきかえ、次の動きをするようにしてみよう！

1. 正方形をえがく



2. 円をえがく



3. スラローム走行。

講

解答例プログラムはそれぞれ以下です。
1. RoboticsProfessorCourse1 ＞ LinkRobot4 ＞ LinkPattern
2. RoboticsProfessorCourse1 ＞ LinkRobot4 ＞ LinkPattern1
3. RoboticsProfessorCourse1 ＞ LinkRobot4 ＞ LinkPattern2

解答例は巻末にも記載します。

次に、プログラムをかくのが少し楽になるテクニックを紹介します。
プログラム「MotorTest」を開きなおして、以下のようにかきかえてみましょう。

```
void setup(){
}

int i;
void loop(){
    for(i = 0; i < 255; i = i + 20){
        mc1.rotate(i);        //100の速度で正方向に回す(範囲は-255から255)
        mc2.rotate(i);        //100の速度で正方向に回す(範囲は-255から255)
        delay(1000);
    }
}
```

実行結果：左右の脚(あし)がだんだん速く動くようになっていく。
ここで登場したforという命令について、くわしく説明します。すでにほかのタームでfor文についてしっかり理解できた人は、読み飛ばして次の課題に進んでもかまいません。

命　　令　「for()」
実 行 結 果：{}　で指定された部分を決めた回数だけくり返し実行する
使　い　方：for(はじめの決まり ; 終わりの決まり ; くり返し実行すること){
　　くり返す内容
　　}

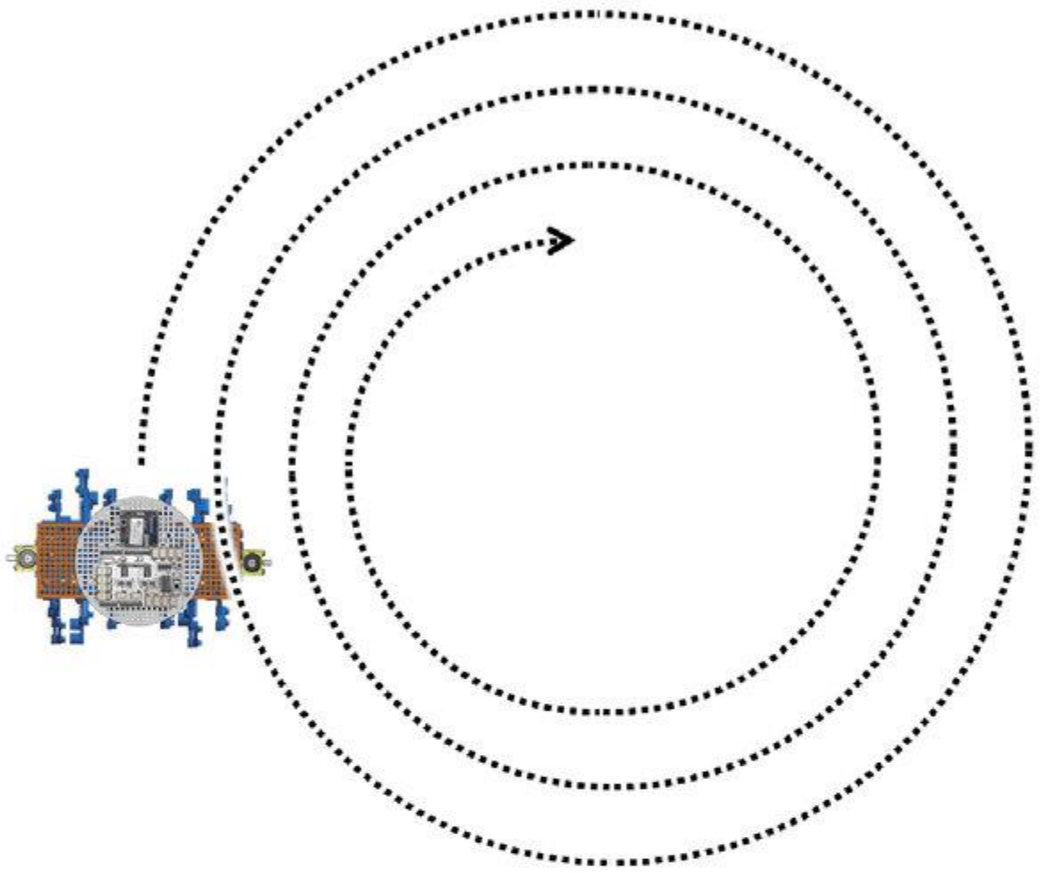
先ほどまで、左右の脚(あし)の速さは mc1.rotate(100); のように数字で指定をしていました。しかし、今回はアルファベットを使い mc1.rotate(i); とかかれています。このiとは変数とよばれる機能で、ループのたびにちがう数値に置きかえることができるのです。
たとえば、for(i = 0; i < 255; i = i + 20)とかいた場合、「はじめの決まり」がi = 0とあるので、まずは変数iが0になり、mc1.rotate(0); 、mc2.rotate(0); の動作を1秒間続けます。それが終わると、「くり返し実行すること」i = i + 20 の式にしたがい、iの値に20が足されます。つまり mc1.rotate(20); 、mc2.rotate(20); となるわけです。この動作を1秒間続けると、今度はiが40になって次の動作を行います。
こうしてiの値が20ずつ増えていき、「終わりの決まり」i < 255 から外れたら終了(しゅうりょう)です。

このかき方はfor文とよばれます。今回のように「だんだん変化する」様子をプログラムするときや、何回か決まった数だけ処理をくり返させたいときに覚えておくと便利でしょう。

では、さっそくfor文を活用して、最終課題にチャレンジしてみましょう！

ステップアップ

プログラム「MotorTest」をかきかえ、渦巻きをえがくようにしてみよう！

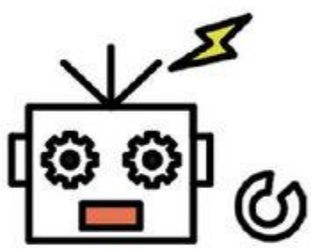


講

解答例プログラムは以下です。
RoboticsProfessorCourse1 > LinkRobot4 > LinkPatternChallenge
解答例は巻末にも記載します。
なお、解答例プログラムでは末尾に `while(1);` という命令を入れています。この命令を入れるとプログラムの処理がそこで止まるため、2 回目以降のループに入ることがなくなります。つまり、いったん渦巻きの中心に辿りついたらそこでロボットが停止します。
ただしテキスト内ではこの命令は解説しておりませんので、くり返し渦巻きをえがくロボットになったとしても問題ありません。

3. まとめ（目安５分）

今回の授業では、リンクロボットの制御（せいぎょ）プログラムをかきかえて、自分の思うように自動走行させることができるようにもなりました。
ただ、今回の自動走行では、みなさんがロボットの進路を考えて、一生懸命（いっしょうけんめい）モーターの回転速度や回転時間を微調整（びちょうせい）しながらプログラムをかきかえましたね。それは、どんな動きをしているのかを知る術（すべ）がロボット自身になかったから、言いかえれば「感じる」ことができなかったからといえるわけです。
ですから、次回の授業では、リンクロボットに「感じる」ためのセンサーを追加しますよ。ロボットをよりカシコクし、ロボット自身で「感じて」、「考えて」、「動く」ようにバージョンアップさせます！



リンクロボットをもっとカシコクしていこう！

≪次回必要なもの≫

次回は、今回使ったロボットのほかに以下のパーツを持ってきてください。

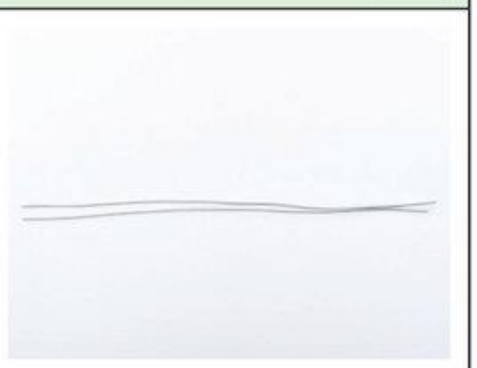
ラジオペンチ 1	ドライバー 1	USBケーブル 1	タッチセンサー 2
			
100mmビニールチューブ 1	M2.6L8タッピングネジ(B) 4	センサー L字ステイ 2	200mm針金 2
			
M3ナット 8	M3L8ネジ 8	ユニバーサルバー 1	
			

図3-0　次回必要なもの

講

○以下の授業の目標を再確認します。
- ・リンクロボットを自動で動かす
- ・さまざまなプログラムを学ぶ
- ・プログラムを改造し、さまざまな動きをつくってみる

○次回テーマは「リンクロボットをカシコクしよう」であることを告知します。

P.15　やってみよう！　解答例（LinkPattern）

```
void loop(){
    //前進
    mcl.rotate(-200);    //左脚モーターを逆方向(マイナス)に回す(範囲は-255 ～ 255)
    mcr.rotate(200);     //右脚モーターを正方向(プラス)に回す(範囲は-255 ～ 255)
    delay(1000);         //1秒(1000ミリ秒)待つ

    //左旋回
    mcl.rotate(200);     //左脚モーターを正方向(プラス)に回す(範囲は-255 ～ 255)
    mcr.rotate(200);     //右脚モーターを正方向(プラス)に回す(範囲は-255 ～ 255)
    delay(500);          //0.5秒(500ミリ秒)待つ
}
```

P.15　やってみよう！　解答例（LinkPattern1）

```
void loop(){
    //円をえがいて旋回する
    mcl.rotate(-200);    //左のモーターを回転(-255 ～ 255の範囲で指定)
    mcr.rotate(100);     //右のモーターを回転(-255 ～ 255の範囲で指定)
}
```

P.15　やってみよう！　解答例（LinkPattern2）

```
void loop(){
    mcl.rotate(-200);          //左のモーターを回転(-255 ～255の範囲で指定)
    mcr.rotate(100);           //右のモーターを回転(-255 ～255の範囲で指定)
    delay(4000);
    mcl.rotate(-100);          //左のモーターを回転(-255 ～255の範囲で指定)
    mcr.rotate(200);           //右のモーターを回転(-255 ～255の範囲で指定)
    delay(4000);
}
```

P.17　ステップアップ　解答例（LinkPatternChallenge）

```
void loop(){
    for(int i = 200; i >= 0; i -= 2){
        mcl.rotate(-200); //左のモーターを回転(-255 ～ 255の範囲で指定)
        mcr.rotate(i);    //右のモーターを回転(-255 ～ 255の範囲で指定)
        delay(500);
    }

    mcl.rotate(0);            //渦の中心で最終的にモーターを停止
    mcr.rotate(0);            //渦の中心で最終的にモーターを停止
    while(1);                 //プログラムの挙動を停止する
}
```