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

不思議アイテムⅢ－1②

第4回

赤外線を追従（ついじゅう）する1

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。

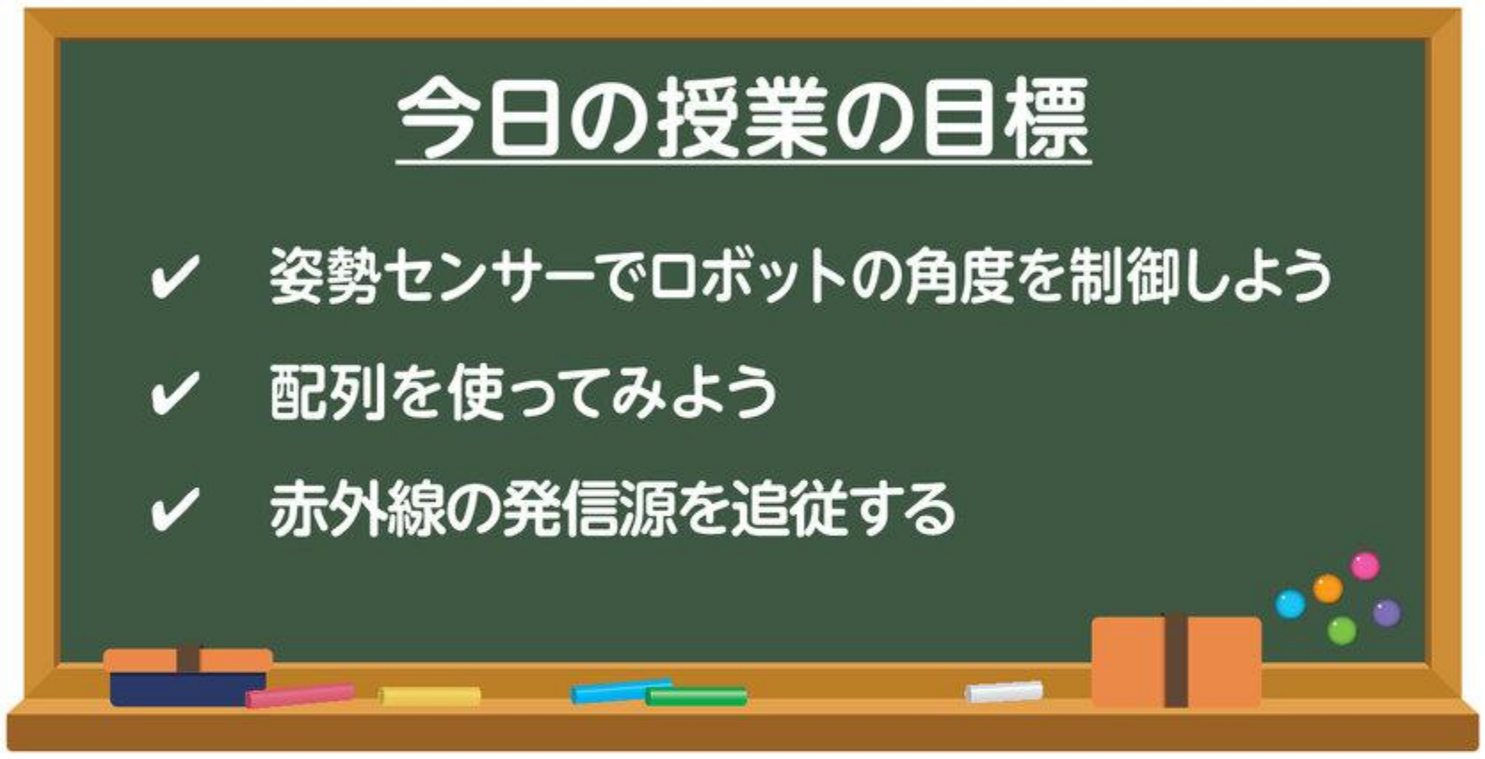
○　今回の目標をパネルで用意するか、黒板に予め書いておきます。

（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。

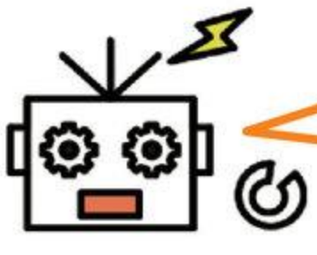
0.　赤外線を追従する１（目安５分）

0.0.　「赤外線を追従する１」でやること



今回、赤外線の発信源の場所を特定し、記録してその方向を向くロボットをつくります。そのためにいくつかのプログラミングの方法を学びましょう。そのひとつは、配列変数の宣言です。これは今まで使ったプログラムにも出てきていましたが、詳しい解説はしていなかった部分ですので疑問に思っていた人も多かったかもしれませんね。こうしたテクニックを使って、発信源を特定するアルゴリズムを考え、実際にロボットを動かすという目的を達成します。このアルゴリズムはいろいろなものが考えられると思いますが、「赤外線LED」「赤外線受光素子」の性質を無視はできないので、その性質を踏まえた上でできることを考えて、アルゴリズム化します。このあたりがロボットのアルゴリズムを考えていく醍醐味ですね。使える部品が決まっていてどうやって目的のものへと組み上げていくか、非常に面白い部分です。今回の授業の方法以外にもたくさん方法は考えられますので、自由に考えてみましょう。

図0-0　追従イメージ



探偵のように追跡していくぞ～！

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。

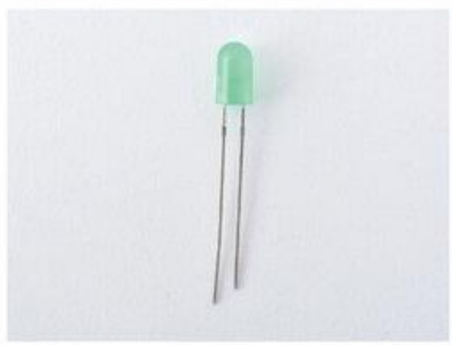
USB ケーブル　1	緑色 LED　1	姿勢検出シールド　1
		

図 0-1　必要なもの

講　授業のはじめにジャンパー線の導通チェックを行わせましょう。

0.2.　ロボットの準備

前回に引き続き基本形は赤外線センサーロボットです。
今回は赤外線の発信源の位置情報を得るために、姿勢検出シールドをつけ加えます。
今回、スピーカーは使いませんので取り外しておきましょう。

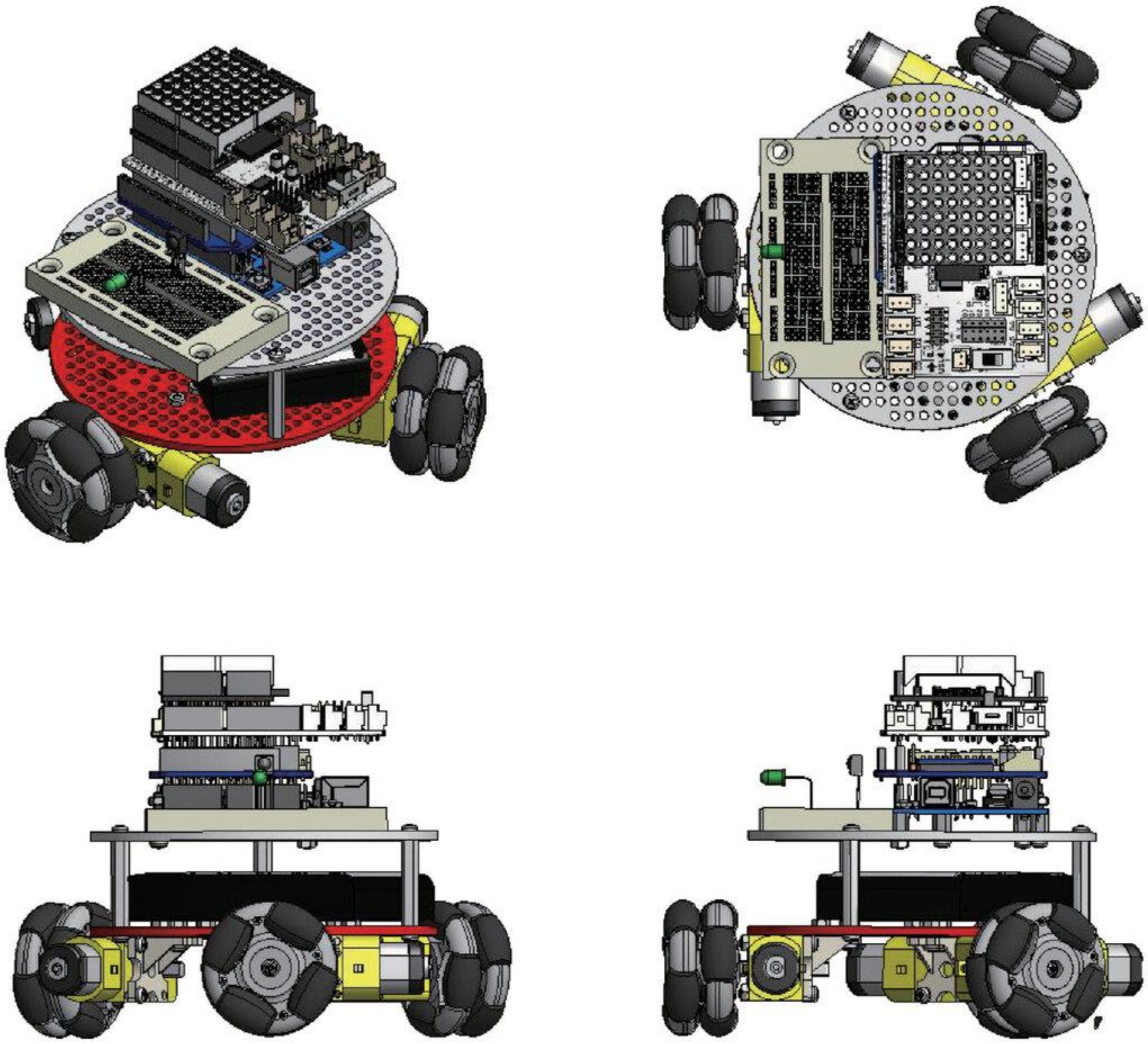


図0-2　赤外線センサーロボットに姿勢検出シールドを接続する

POINT

姿勢検出シールドは、**図0-2**の通りマイコンボードとロボプロシールドの間に取りつけましょう。

１．アルゴリズムについて考えよう（目安 100 分）

1.0. 「アルゴリズム」とは？

アルゴリズムとは問題を解くための手順のことを意味します。ある問題を解くために複数の解決方法がある場合があります。それは、私たちの日常生活においてもあてはまることがあると思います。
アルゴリズムの説明するときにこんなたとえがあります。
下の図のような人参の飾り切りをつくりたいとき、いくつかの方法があります。
たとえば、30個の星型の飾り切りをつくる場合、どのような方法で切り、包丁を何回入れればいいでしょうか。一緒に考えてみましょう。

図1-0　人参の飾り切り

１）方法①：人参を輪切りにしてから星型の飾り切りにする

１つ目の方法は人参を輪切りにしてから図のようにそれぞれの角を落として星型にする方法です。
この方法ではまず人参の両端を取るのと、輪切りを30個つくるので31回包丁を入れて、さらにそこから星型にするのに図のように輪切り１つにつき10回ずつ包丁を入れればつくれます。

31回包丁を入れる

星は辺が10あるので、星１個につき10回包丁を入れる

図1-1　方法①

2）方法②：人参一本をあらかじめ星型にしてから輪切りにする

2つ目の方法は人参の断面を星型にしてから輪切りにする方法です。
この方法では人参一本を星型の断面になるように切ってから30等分にします。最初に両端を取るのに2回、星型にするのに10回、30等分するのに29回包丁を入れればつくれます。

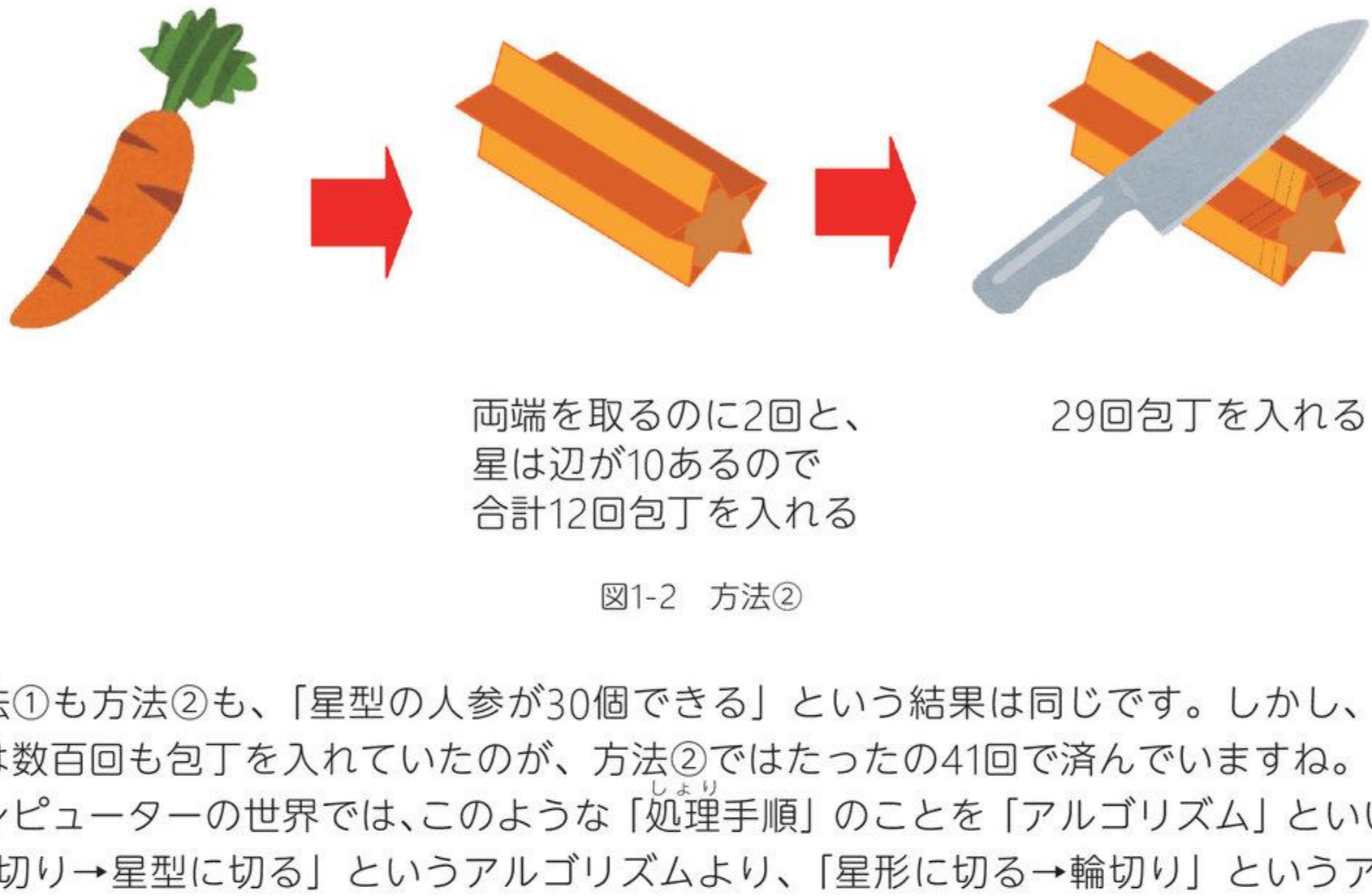


図1-2　方法②

方法①も方法②も、「星型の人参が30個できる」という結果は同じです。しかし、方法①では数百回も包丁を入れていたのが、方法②ではたったの41回で済んでいますね。
コンピューターの世界では、このような「処理(しょり)手順」のことを「アルゴリズム」といいます。「輪切り→星型に切る」というアルゴリズムより、「星形に切る→輪切り」というアルゴリズムのほうが、より少ない作業ですばやくゴールにたどり着けることがわかります。

プログラミングをする際にも、よりシンプルで効率的なアルゴリズムを考えることが重要です。
プログラムがシンプルにできている方が、マイコンの負担が少なく済みますし、ミスやエラーも発生しにくくなるからです。

1.1.　赤外線を追従(ついじゅう)するロボット

今回の目標は、赤外線センサーロボットが、「赤外線の発信源を探し出して、その方向を向く」という内容です。いろいろと手順が考えられると思いますが、まずはロボットの機能を再確認して少しずつ手順を考えていきましょう。

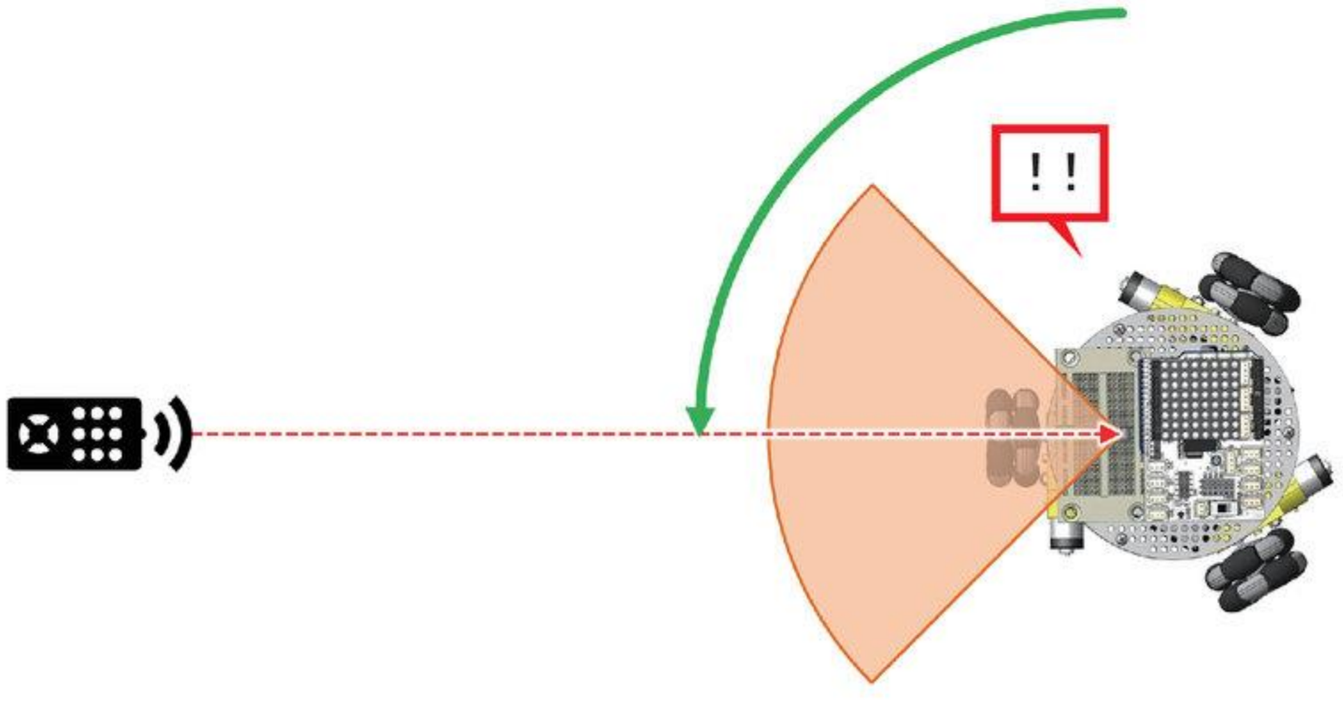


図1-3　「赤外線センサーロボット」の動作イメージ

1.2. ブレッドボード配線の変更

ここまでに「シーカーが赤外線の発信源の位置を推定するアルゴリズム」を説明しました。ここからは実際にロボットを動かして実証していきましょう。ビーコンとシーカーのブレッドボードの配線を**図1-4**の回路を参照して変更しましょう。この回路では赤外線LEDから発信される信号の様子を可視化できるように緑色LEDを**図1-5**のように直列に取りつけます。ビーコン・シーカーとも同じ配線です。

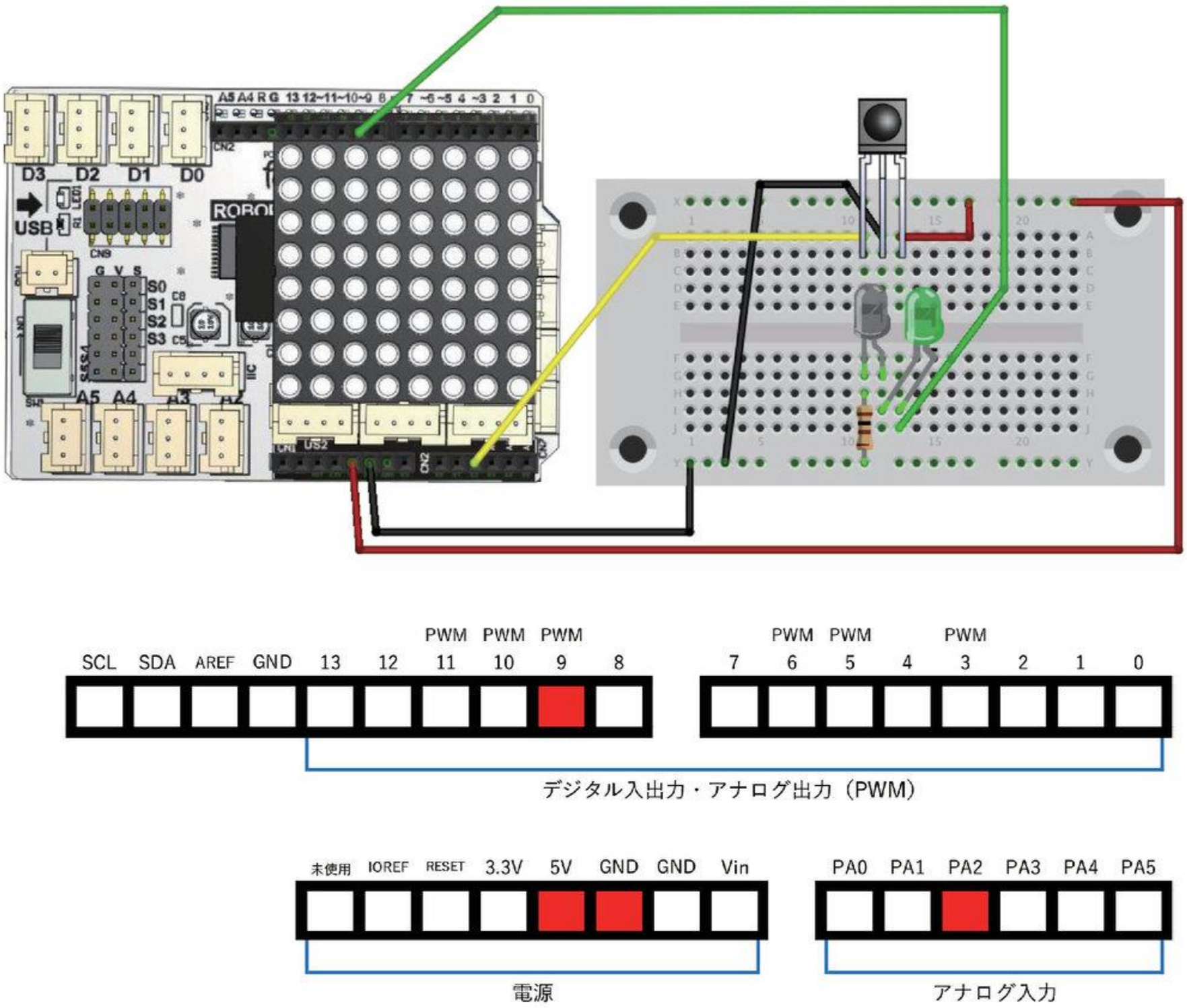


図1-4　ビーコン・シーカーの回路およびマイコンボードの入出力詳細図

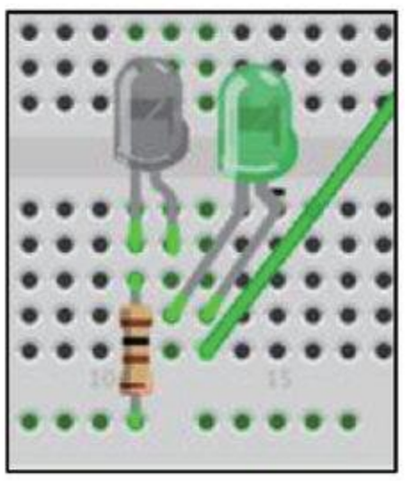


図1-5　緑色LEDの取りつけ

1.3.　赤外線LEDと赤外線受光素子の調整

図 1-6 を参照して、ブレッドボード上の赤外線 LED、赤外線受光素子は前方を向いていることを確認しておきます（シーカー、ビーコン共通です）。前を向けることで、発信源の位置を特定しやすくなります。

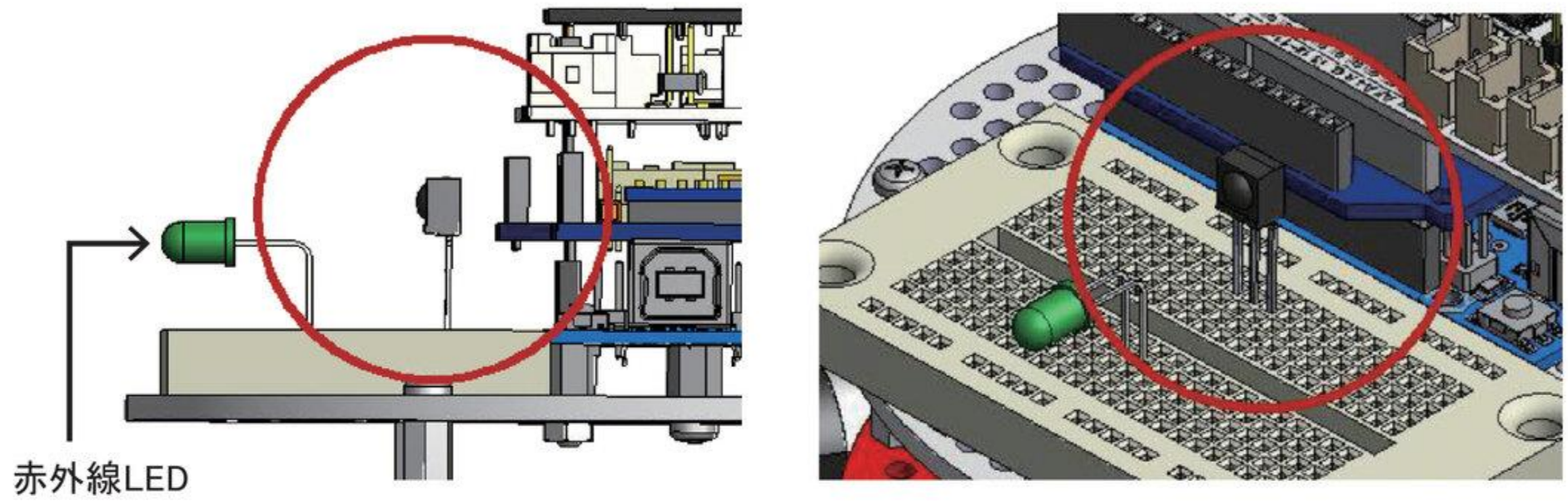


図1-6　赤外線LEDと赤外線受光素子の状態

1.4.　ビーコンの動作確認

それでは、ビーコンから発信テストを行います。以下のプログラムを実行すると、赤外線 LED から発信される信号と同じテンポで緑色 LED が点滅します。また、マトリクス LED のドットもグルグルとまわりはじめます。

プログラムの書き込み

RoboticsProfessorCourse3 > infraRed4 > IRBeacon

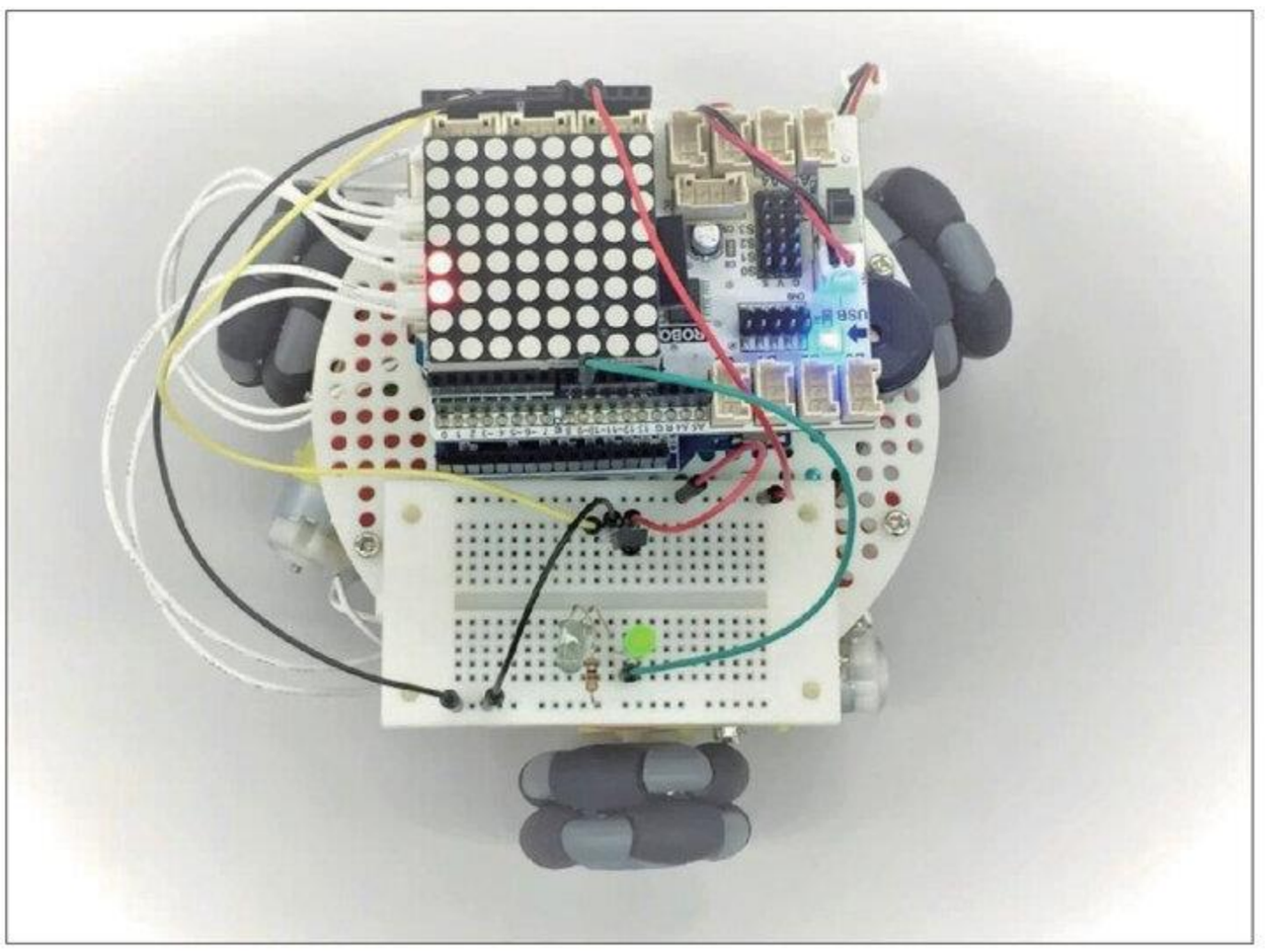


図1-7　ビーコンの動作

1.5. 赤外線の検知範囲

赤外線を追従するロボットはシーカーの方です。
まず、ロボットを動かす前にシーカーの機能を再確認して、アルゴリズムを考えていきましょう。
赤外線受光素子の受光範囲は左右30度くらいです。
しかし、これはしっかりと信号を受信できる範囲を示していて、検知するだけならもっと広角になります。環境差はありますが、左右45度くらいになる場合が多いです。

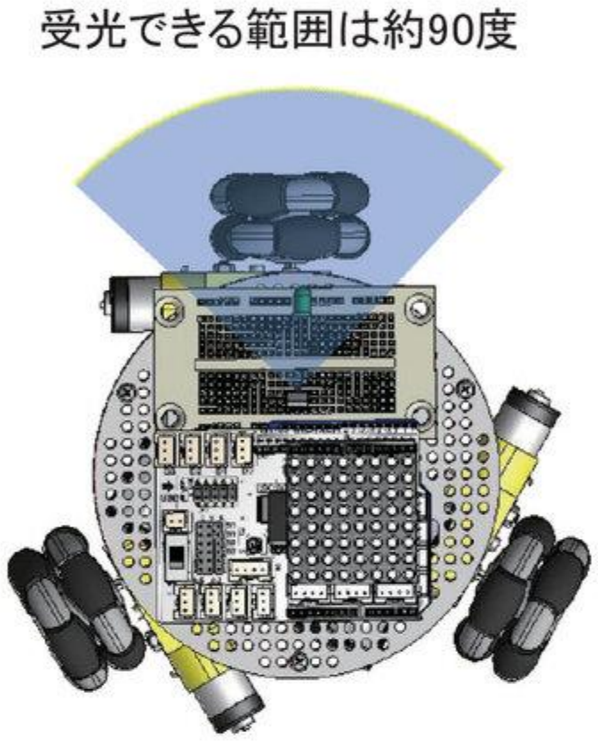


図1-8　赤外線受光素子の受光範囲

つまり、人間の視野は両目で180 ～ 200度くらいといわれていますが、このロボットの視野は90度と人間の半分くらいであるということができます。また、人間はまわりの状況を確認するときに首を動かしたり、姿勢をかえて周囲を見渡します。ロボットには、この行動を回転動作にかえて行わせることにします。

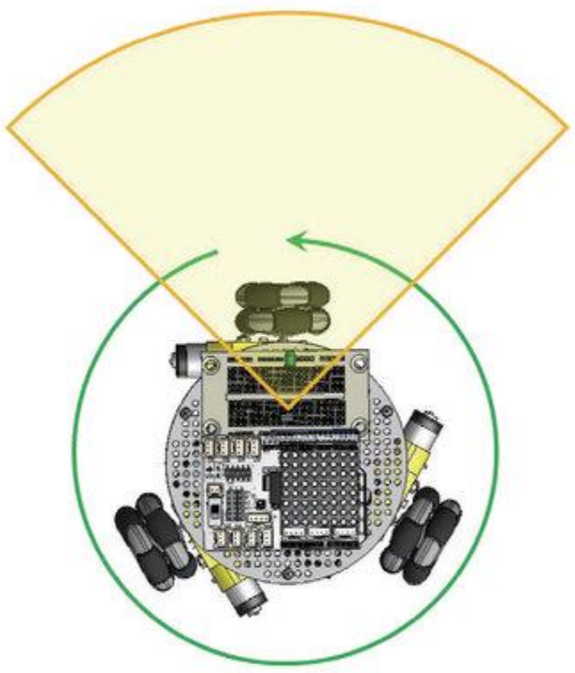


図1-9　赤外線を追従するアルゴリズム

1.6.　赤外線を検出するアルゴリズム（手順）

1）シーカーが回転して、周囲の赤外線を探す

今回姿勢検出シールドを取りつけたことで、ロボットを一周だけ回転させる、という動作をさせることができるようになりました。
以下のプログラムを書き込んでみましょう。

プログラムの書き込み

RoboticsProfessorCourse3 > infraRed4 > ZAngleCtl

実行結果：ロボットが反時計回りに回転し、一周したところで止まる。

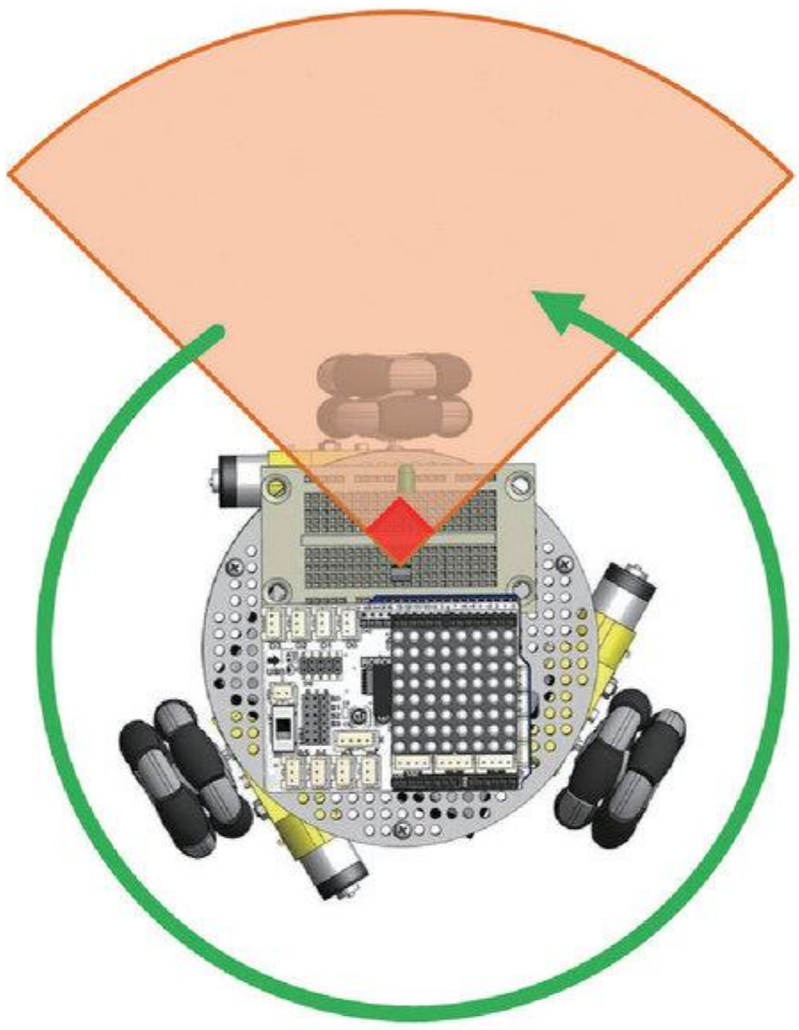


図1-10　回転運動をしながら360度を見渡す

プログラム「ZAngleCtl」より抜粋

```
Err = ang - Ref;
spd = constrain(Kp * Err, -40, 40);
omniBot.move(0, 0, spd);
```

変数 ang が現在の角度、 Ref が目標の角度です。つまり、 ang は初期値0から少しずつ増加していくのに対して、 Ref は360のままずっと固定です。
この2つの差、つまり「一周まであと何度か」の値を変数 Err としています。ちなみに、Errという変数名はError（誤差）という英単語から来ています。
あとは、 Err の値をもとに回転速度 spd を決めるだけです。こうすることで、ロボットがちょうど一周、つまり変数 Err が0になったときに停止することになります。あまり高速にならないように、constrain文を用いて回転速度が±40におさまるようにしています。
なお、 spd は倒立振子ロボットでも扱った「比例制御」で求めています。変数 Kp は比例ゲインでしたね。

2）赤外線を検知していた範囲をチェックする

赤外線受光素子で検知をつづけながらロボットを一回転させると、はじめは赤外線を受信せず、ある瞬間から赤外線を受信しはじめ、しばらく回転するとまた赤外線を受信しなくなる、という場合などが考えられます。
受信開始と受信終了、それぞれ何度くらい回転したときのことだったかを記録しておけば、発信源の特定に役立ちそうですね。

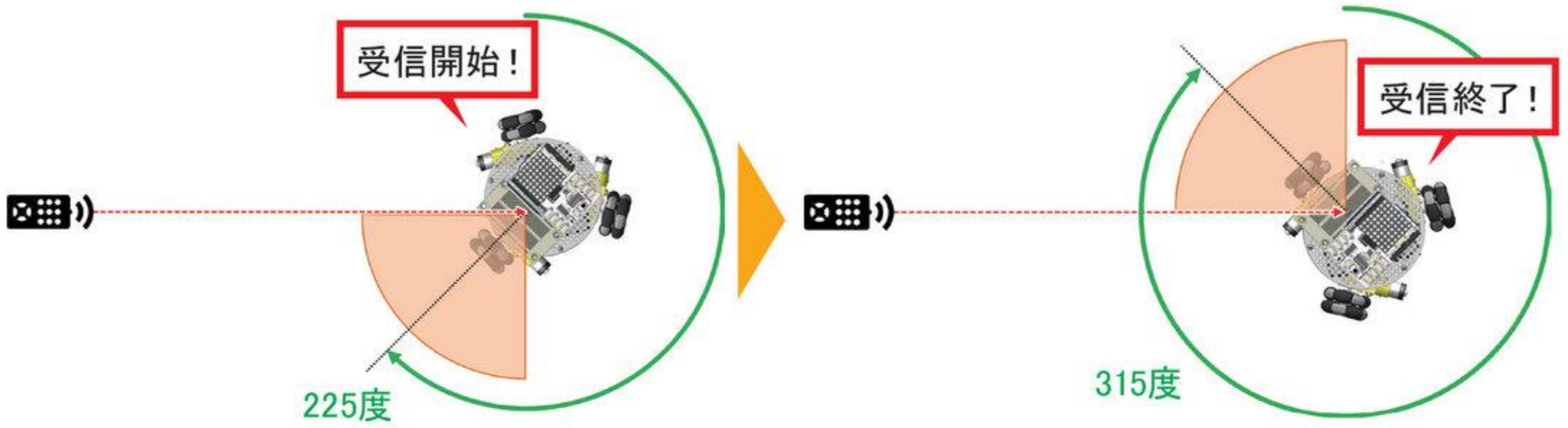


図1-11　赤外線の検知方法

図1-11のような状況なら、発信源は受信開始角度と受信終了角度のちょうど中間あたりにあったと推測できます。

やってみよう！

受信開始角度を変数 min 、受信終了角度を変数 max とすると、発信源の角度を求めるための式はどのようになるかな？

(min + max) ÷ 2

やってみよう！

プログラム「ZAngleCtl」を書きかえ、「回転中、赤外線を受信したら受信データ数を変数に記録する」という処理を追加してみよう！

次のようなif文を追加すれば、赤外線を受信した際の角度を変数に記録しておけますね。

```
if (irrecv.decode(&results)){
    count = results.rawlen;
}
```

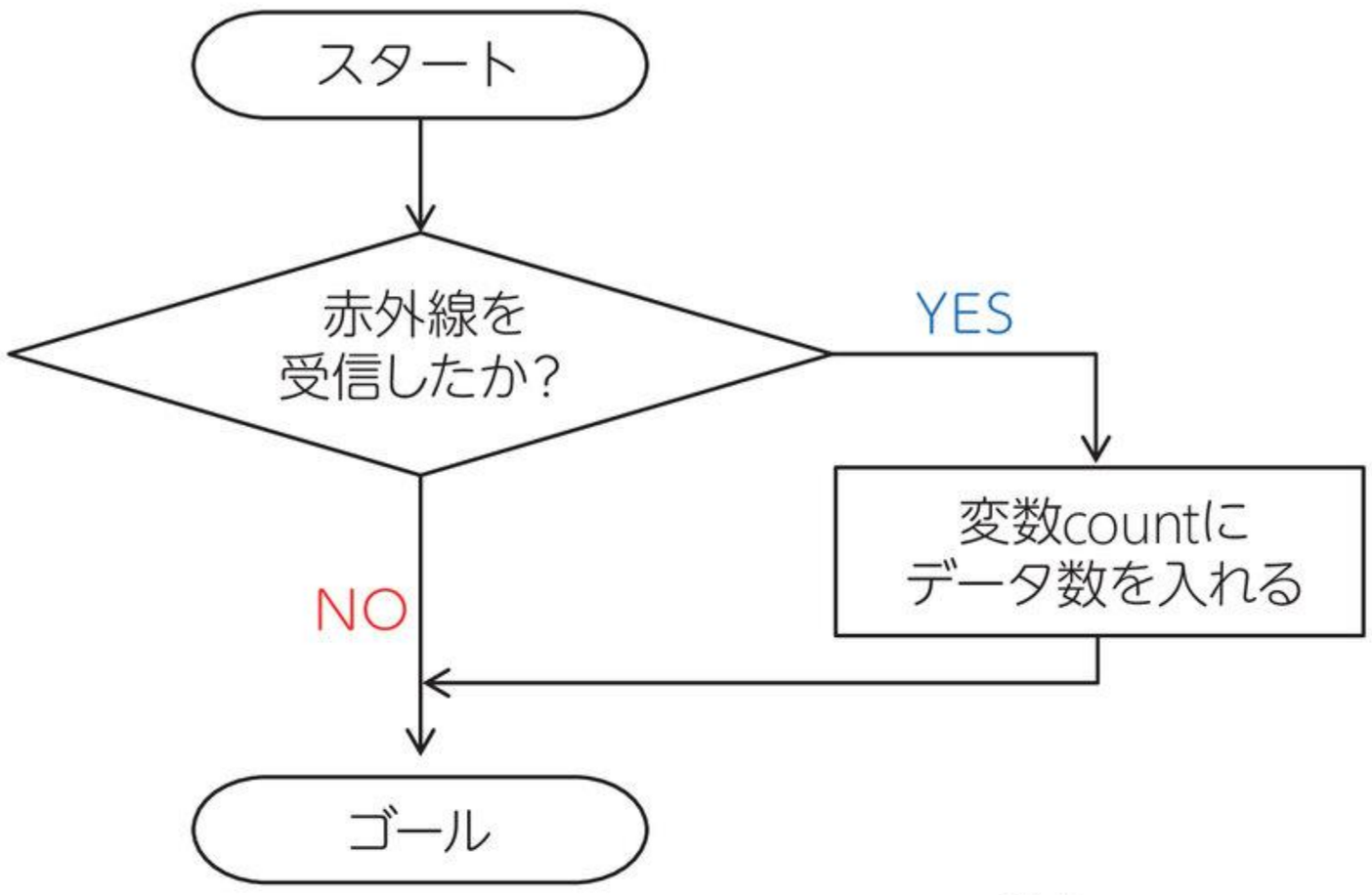


図1-12　データ数を変数に格納する条件分岐

フローチャートにすると**図1-12**のような処理です。
実は、この処理にはある問題点があります。フローチャートから読み取れるでしょうか？

正解は、赤外線を受信するたびに変数 count の値が上書きされていくため、一番最後に受信したデータ数以外の記録は消えてしまうという点です。
これを防ぐには、赤外線を受信するたび、新たな変数にデータ数を入れていく必要があります。

やってみよう！

プログラムをさらに書きかえ、上記の問題を解決してみよう！

ヒント

たとえば count0 ～ count360 の変数を宣言して1つ1つ使っていくやりかたでもデータ数の記録はできるけど、宣言部分だけでもかくのが大変すぎるね。
1回の宣言で、変数を何個もつくって使用したい。そんなときに役立つテクニックを以前学んだことを覚えているかな？

```
if (irrecv.decode(&results)){
    count = results.rawlen;
    range[(int)ang] = count;
}
```

range という変数を配列化しました。添え字に変数 ang を使っているので、一周する間に count[0] ～ count[360] の配列に1つずつ reluts.rawlen の値が入っていきます。それぞれの角度のときの受信データ数を、別々に記録しておけるようになりましたね。ただ、実際に添え字を何百個も使うとマイコンの負担が大きくなってしまうので、10度回転するごとに1回記録するものとして count[36] までの配列にしておきましょう。

```
if (irrecv.decode(&results)){
    count = results.rawlen;
    range[(int)ang / 10] = count;
}
```

講

上記の処理（各方角からのデータ受信数を配列に収める）を実装したものが以下のプログラムです。
RoboticsProfessorCourse3 > infraRed4 > IRSeekerSt1
格納した数字をマトリクスLEDに表示させる等の命令が無いので、外見上の動作は変化がありません。

POINT

ang はfloat型の変数として宣言されていますが、添え字に小数が使えないため、int型として宣言しなおしています。

1.7. 赤外線を追従するロボットのアルゴリズム（まとめ）

ここまでのプログラムをまとめると、以下のような処理が行われています。

POINT

① ロボットが360度回転する。
② 回転中、赤外線を受信しているかチェックする。
③ 赤外線を受信していたら、変数 `range[]` にデータ数の値を入れる。

これに加えて、「赤外線を追従するロボット」に必要な機能は以下のようなものが考えられますね。

POINT

④ 赤外線受信の開始角度、終了角度を `range[ang / 10]` の中から特定する。
⑤ 受信開始角度、受信終了角度を用いて、発信源の推定方向を算出する。
⑥ ⑤で求めた方角を向くように、ロボットを回転させる。

まず、④および⑤の処理からつくっていきましょう。

ステップアップ

プログラムをさらに書きかえ、以下のような処理を追加してみよう。

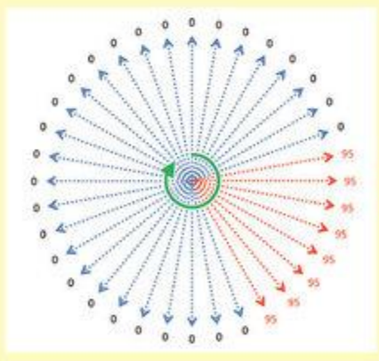
POINT

・range[ang / 10]のうち、赤外線受信が始まったときの `ang / 10` の値を `min_ang` 、受信を終えたときの `ang / 10` の値を `max_ang` に入れる。
・ `min_ang` の値と `max_ang` の値を使い、発信源の推定方向を算出する。
・算出した値をマトリクスLEDに表示する。

ヒント

`results.rawlen` の最大値は、前回までのように赤外線で文字やデータをやりとりするなら68だったけど、今回はただ赤外線を発信するだけなので100になるよ！

たとえば「 `results.rawlen > 90` なら赤外線を受信しているものとする」のような基準にするといいね！



上の図のような検知をしたとすると、受信を開始したのが80度のとき、終了したのは160度のときなので、 `max_ang = 8` 、 `min_ang = 16` となればいいんだね！

赤外線を受信し始めると、(誤検知でなければ) そこからしばらく赤外線を受信し続けます。また、赤外線受信が終了すれば、当然ながらそのあとは赤外線を受信しない状態が続きます。

よって、「何回か連続で赤外線を受信している」という範囲を探せば、その中でもっとも角度が小さいときが min_ang になりますし、それ以降で「何回か連続で赤外線を受信していない」という範囲を探せば、その中でもっとも角度が小さいときが max_ang となります。

```
for(int i = 0; i < 33; i++){
    if(range[i] > 90 && range[i + 1] > 90 && range[i + 2] > 90 && range[i +
3] > 90){
        min_ang = i;
        break;
    }
}

for(int i = min_ang; i < 33; i++){
    if(range[i] < 90 && range[i + 1] < 90 && range[i + 2] < 90 && range[i +
3] < 90){
        max_ang = i;
        break;
    }
}
```

あとは、min_ang と max_ang を足して2で割れば、発信源の推定方向が算出できます。ただし、 min_ang や max_ang は10倍しないと角度の値にならないので注意が必要です。

算出が終わったら、ロボットを停止させるために回転速度 spd を0にしておきましょう。ついでに、求めた値をマトリクスLEDに表示させるようにしておくとわかりやすくなります。

```
source = 10.0 * ( max_ang + min_ang ) * 0.5;
myMatrix.putd2(0, 0, source / 10);
delay(500);
spd = 0;
```

講

解答例は以下のプログラムです。
RoboticsProfessorCourse3 > infraRed4 > IRSeekerSt2
ただし、このプログラムは後ほど出題される「赤外線検知範囲が0度を含む場合」の特殊処理を既に含んでしまっているため、最後のチャレンジ課題にも取り組ませたい場合は使用を避けてください。

これで、向くべき方向を求めることはできるようになりました。
あとは、オムニホイールロボットを回転させ、実際にその方向へ向き直るような処理(しょり)を追加してあげるだけですね。

ステップアップ

プログラムをさらに書きかえ、算出した方向にロボットが向き直って停止する処理(しょり)を追加しよう！

ヒント

一周するときの回転速度は変数 spd で制御していたね。この変数がどのように求められていたか、もう一度確認してみると役に立つかもしれないね！

講

解答例は以下のプログラムです。
RoboticsProfessorCourse3 > infraRed4 > IRSeekerSt3

検知の際にはロボットが360度だけ回転しますが、これは変数 Ref を360に設定していたためでした。

つまり、変数 Ref はロボットの目標角度を表していることになりますね。

発信源の方向を計算するころにはすでにロボットは一回転し終えているので、 Ref の値は360でなくても構いません。

むしろ、Refに発信源の方向を入れてあげれば、今度はそちらに向けて回転してくれることになりますね。

```
Ref = 10.0 * ( max_ang + min_ang ) * 0.5;
```

これなら、 spd を0にすることなく、発信源の方を向いたら自動的にロボットが停止してくれます。

ただし、この処理にはある問題があります。

たとえば、以下のように発信源がはじめから正面にあると、ロボットが正反対を向いて止まってしまいます。

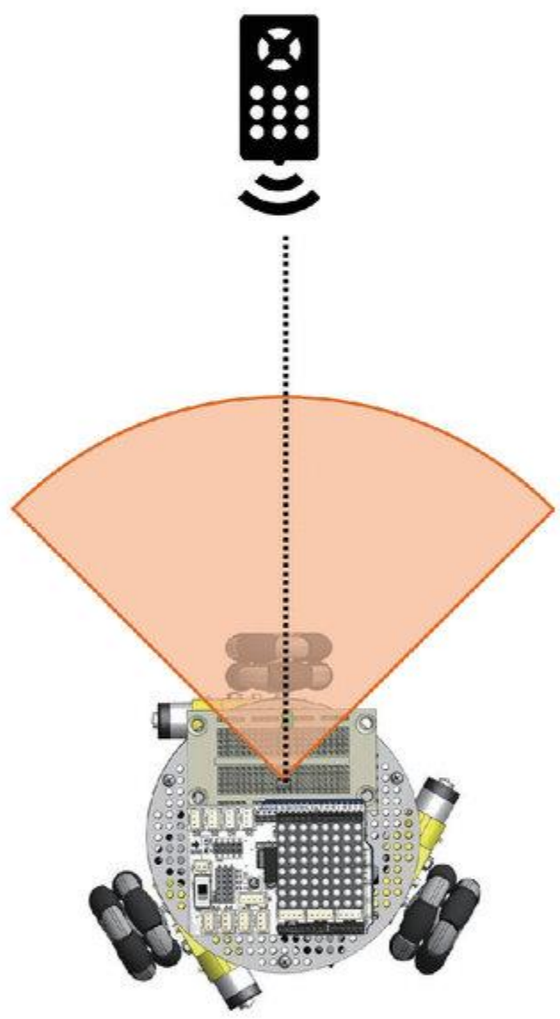


図1-13　発信源が正面にある場合

チャレンジ課題

プログラムをさらに書きかえ、上記の問題を修正してみよう！

講

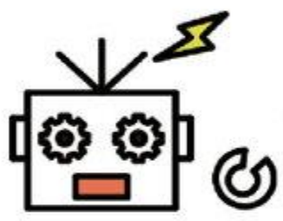
ステップアップの解答例「IRSeekerSt3」は、すでに上記の問題を解消したつくりになっています。

赤外線を受信した範囲に0度を含むと、 min_ang が0、 max_ang が360となり、目標角度 Ref が180になってしまうため、反対側を向いてしまいます。

解答例では、もし範囲に0度を含むならフラグ用変数 overZero を1にし、 overZero が1のときのみ発信源算出用の数式を専用のものに切りかえるような処理が追加されています。

2．まとめ（目安５分）

今回は、赤外線の発信源の方向を推定(すいてい)して、そちらの方向を向くということができるようになりました。
第6回では、ロボカップサッカー大会のように、赤外線ボールに見立てたビーコンを追いかけることを行います。今回行ったことは、赤外線ボールを探すのに役立つアルゴリズムです。
次回は、サッカーが行えるようなロボットにするために、赤外線センサーをバージョンアップして、もっと精度の高いロボットにしていきましょう。



いよいよ、自律ロボットらしくなってきたね～。
次回、さらにスゴイロボットにしよう！

《次回必要なもの》
今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。

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 1
			
センサー L字ステイ 3	カラーセンサー 1	センサーケーブル 1	200㎜針金 1
			
M3 ナット 12	M3L8 ネジ 8	ユニバーサルバー 1	M3L6 フラットヘッドビス 2
			

図 2-0　次回必要なもの①

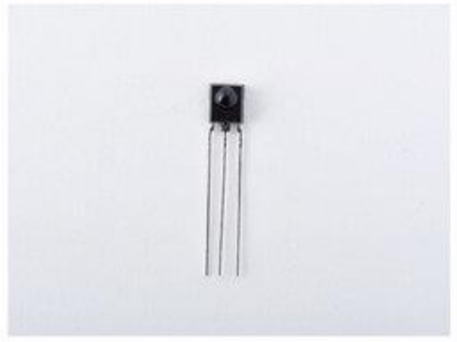
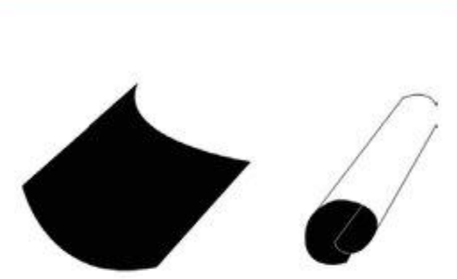
M3L10 ネジ 2	25mm角スペーサー 6	ジャンパー線 65	赤外線受光素子 1
			
コンデンサー 0.022uF 2	**1kΩ抵抗 2**	**15㎜ビニールチューブ 1**	**色紙　円筒 1**
			
セロハンテープ 1			
			

図 2-1　次回必要なもの②

講

○ 今回の授業の目標を再確認します。
　・姿勢センサーでロボットの角度を制御しよう
　・配列を使ってみよう
　・赤外線の発信源を追従する
○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、「赤外線を追従する2」であることを告知します。