

自律ビーチフラッグ競技ロボットの センサー実装例とプログラム

超音波センサー + PWMサーボ編

近藤科学株式会社 開発部

目次

1 始めに.....	2
2 機体.....	2
2.1 コントロールボードを接続する.....	2
2.2 センサーの取り付け.....	3
2.3 ピポットターンユニット.....	4
2.4 その他の機体修正点.....	4
3 プログラミング例.....	4
3.1 スタート.....	6
3.2 起き上がって向きを変え、適当な距離だけフラッグへ近づく.....	6
3.3 倒れていた場合は起き上がる.....	6
3.4 壁に近づいている場合には、旋回して避ける.....	6
3.5 適当な距離だけ前進する.....	7
3.6 首を旋回させながら測距センサーで距離を計測し、目標を見つける.....	7
3.7 機体の向きを変える.....	8
3.8 目標までの距離が遠い場合は、3番へ戻る.....	8
3.9 目標の目の前まできたら特定のモーションを再生させ、フラッグを倒す.....	8
4 モーション再生について.....	9
4.1rcb3_motion_play.....	9
4.2rcb3_put_XBcode.....	9
5 まとめ.....	9

1 始めに

センサーを頼りにロボットが自律で行動する競技『自律ビーチフラッグ』はルールは簡単ですが、ロボットが自動で行うためには、いくつかポイントを押さえておく必要があります。ここではそのポイントや、実際の機体へのセンサーの実装例やプログラミングの例を紹介します。前回(PSDセンサー実装編)では、KHRの頭の部分にKRS4013サーボモーターをSIO端子に接続していましたが、今回は、KRS-788HV・PWMサーボモーターに戻して制御を行います。また、頭部のPSDセンサーは超音波センサーに付け替えています。

2 機体

自律ロボットは以下のような構成となっています。

ビーチフラッグロボット構成例			
項目	構成	項目	構成
機体	KHR - 2HV	超音波センサー	USRX - 1 x 1
コントロールボード	RCB - 3J + KCB - 1		超音波センサー取り付け金具
ユニット	ピボットターンユニット	測距センサー	PSD(位置検出素子)センサー x 2
頭部	KRS - 788	加速度センサー	RAS - 1 x 1
腕部	延長アダプターA x 4		
足	バスタブソールと田宮工作キットL字アングル		

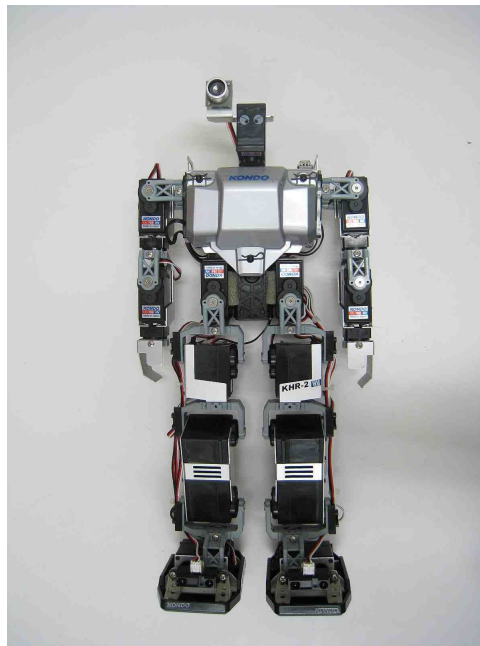


図 1: KHR-2HV+ピボットターンユニット

図 1 は弊社で作成したビーチフラッグ競技用ロボットで、ベース機体は KHR-2HV を使い、旋回を速く行うためにピボットターンユニットを取り付けています。足裏にはバスタブソールを装着しています。

2.1 コントロールボードを接続する

まず RCB-3J と KCB-1 にバッテリーから電源を供給するために、バッテリーを HV 電源スイッチハーネスに接続し、RCB-3J の電源接続端子へ接続します。次に RCB-3J で使用していないサーボ用端子(本作例では S9 端子を使用)

と KCB-1 の SIO 端子をサーボボードで接続し、KCB-1 へ電源を供給します。ここで KCB-1 の SIO 端子のシリアル通信データと RCB-3J の PWM データが混信しないように、サーボボードの白いラインを外し、電源線(赤)とグラウンド線(黒)のみを接続するようにします。くれぐれも電源ラインは間違えないようにしてください。

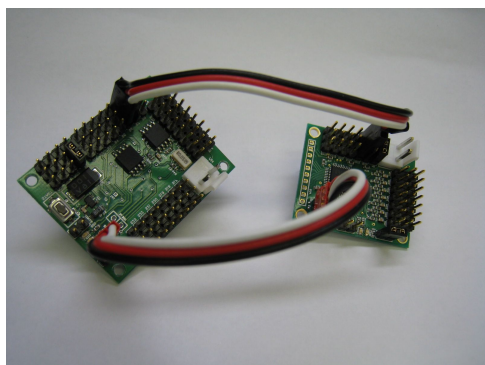


図 2: 電源ラインの接続とクロスケーブルの接続例

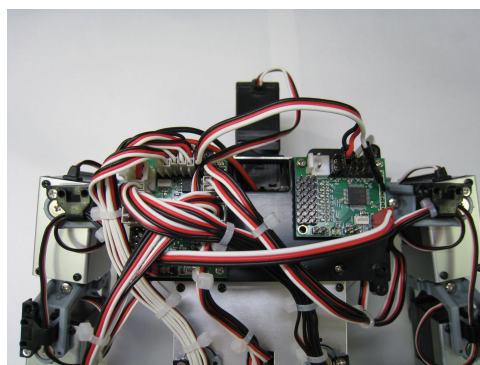


図 3: コントロールボード配置例

2.2 センサーの取り付け

センサーは以下のものを使用しました。

- KONDO 製 USRX-1 超音波センサー(受信部)(頭部×1)
- KONDO 製 PSD(位置検出素子)センサー(足部先端×2)
- KONDO 製 RAS-2 加速度センサー(肩部×1)

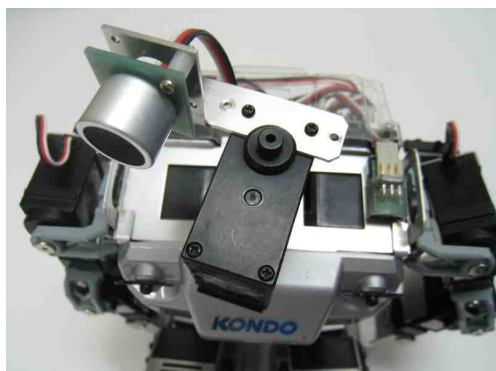


図 4: 頭部超音波センサー取り付け例

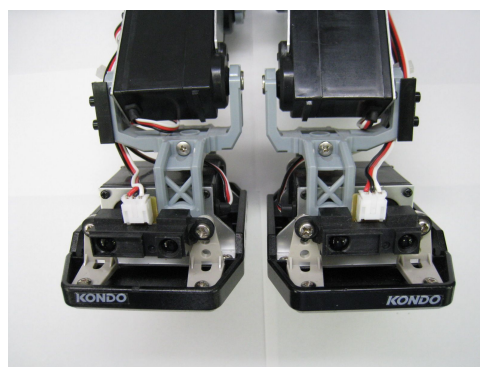


図 5: 足部 PSD センサー取り付け例

頭部測距センサーはビーチフラッグを探すために利用します。例では頭部センサーはロボスポットで取り扱っている超音波センサー取り付け金具で取り付けました。フラッグに USTX-1 超音波センサー(送信部)が取り付けられているため、フラッグからの超音波を受信することにより、フラッグとその他の障害物を間違えることはありません。

足部測距センサーはタミヤ工作キットの L 字パーツを利用し、バスタブソールの先端に取り付けて壁を検知します。

機体の姿勢を感知して自動起き上がりをするために、加速度センサーを図 6 のように取り付けておきます。

目標の距離と方向を検知するために、頭部の PWM サーボは KCB-1 に接続し、モーションに関係なくプログラムで制御できるようにしました。この変更により KCB-1 から pwm_out 命令で頭部に取り付けた超音波センサーの向きを正確に制御することが出来るようになります。

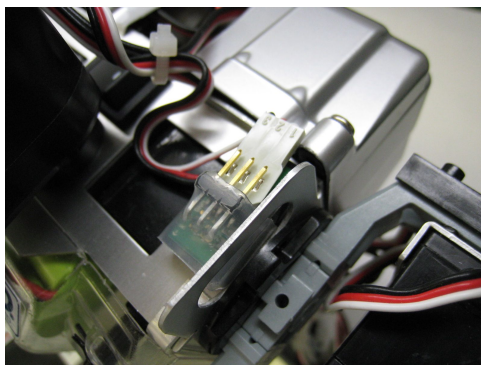


図 6: 加速度センサー取り付け例

2.3 ピボットターンユニット

目標位置を見つけた後でロボットをその方向へ正確に向けるために、股関節部分にはピボットターンユニットに変更しています。

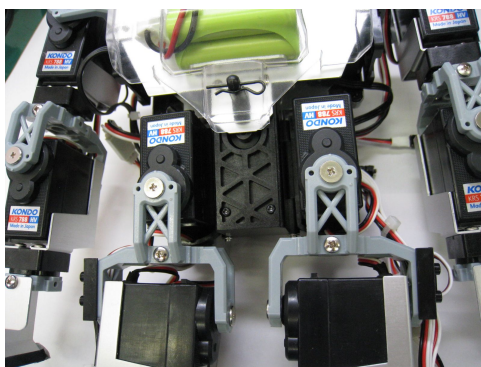


図 7: ピボットターンユニット

2.4 その他の機体修正点

起き上がりをスムーズに行うために延長アダプターAで腕を延長しました。腕の長さは大会規定に沿ったものにしてください。また、電源投入後、スイッチで自律を開始させるためにプッシュスイッチをアナログポートに接続しています。

3 プログラミング例

ここでは測距センサーと加速度センサーを実装した簡単な自律ロボットのプログラムを説明します。

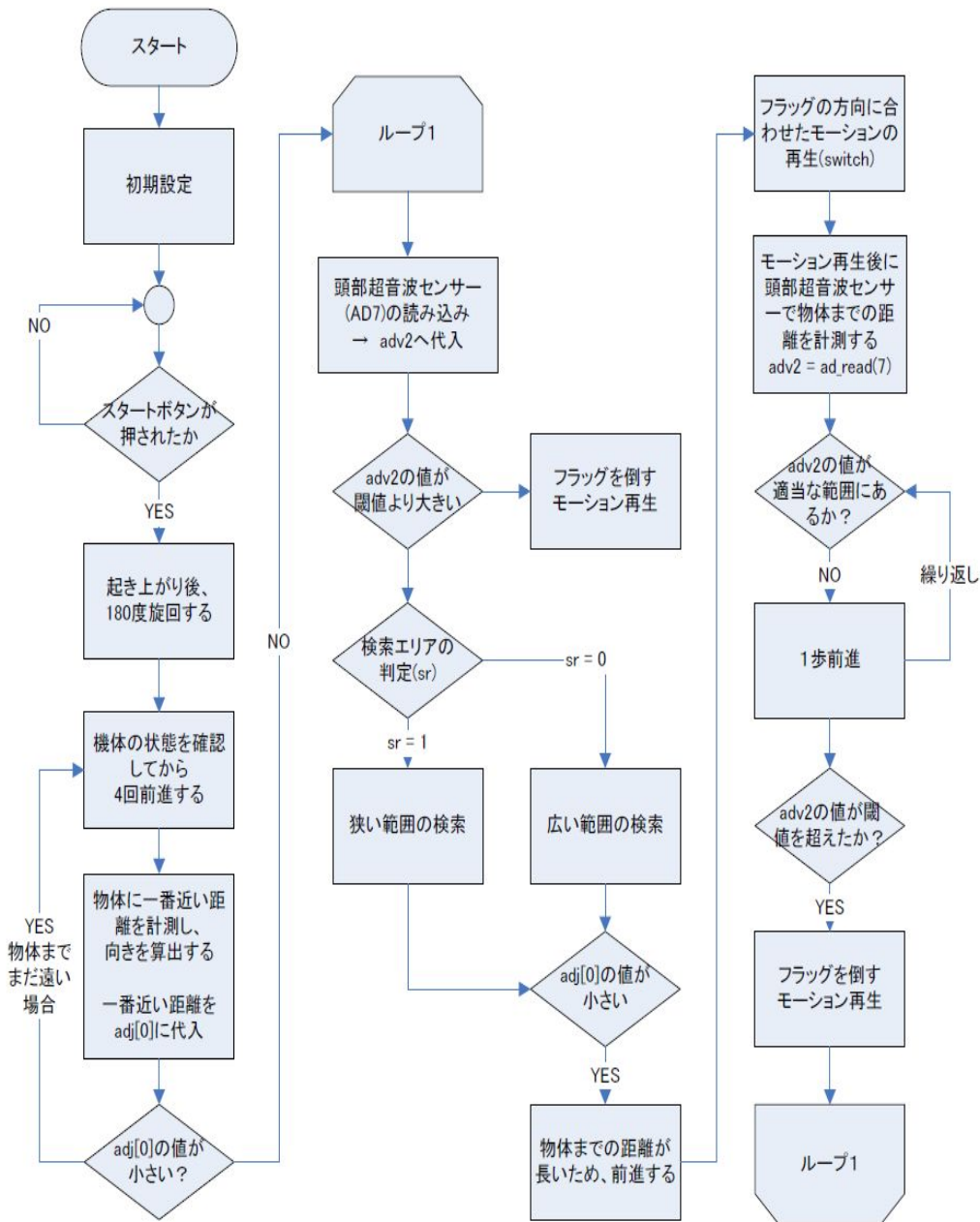


図 8: フローチャート

プログラムの基本的な処理内容は次のようになっています。

1. スタート
2. 起き上がって向きを変え、適当な距離だけフラッグへ近づく
3. 倒れていた場合は起き上がる
4. 適当な距離だけ前進する
5. 壁に近づいている場合には、旋回して避ける
6. 首を旋回させながら測距センサーで距離を計測し、目標を見つける
7. 機体の向きを変える

8. 目標までの距離が遠い場合は、3番へ戻る
9. 目標の目の前まできたら特定のモーションを再生させ、フラッグを倒す

各センサーは KCB-1 の各 AD ポートに取り付けています。

- 加速度センサー： AD1
- 脚部測距センサー： AD5(右)、AD6(左)
- 頭部測距センサー： AD7

3.1 スタート

ビーチフラッグ競技では開始後、ロボットを始動(起動ではなく)させるために一回だけ機体に触れて良いことになっています。逆に言えばそれまではロボットを待機状態にしておかななくてはなりません。そこで起動用プッシュスイッチを KCB-1 の AD2 ポートへ取り付けて、スイッチが押されるまで待機状態を保つように、次のような空ループを構成します。

```
while (ad_read(2) <= 500) {};
```

まずプッシュスイッチが押されたら、AD2 ポートの真ん中にある電圧供給線と、一番内側にあるアナログ値読み取り線が短絡(ショート)するようにスイッチを取り付けてください。KCB-1 の AD 変換は 10 ビット精度(0~1023)の値を示しますので、スイッチを押していない場合は 0、スイッチを押せば最大値を読み取ることになります。ただしアナログ値は適当にばらついた値となりますので、正確に 0 や 1023 とはならないことに注意してください。上の例ではスイッチの押し下げ状態の境界を 500 としています。

while 文は指定された条件が成り立つ間は処理を繰り返すという命令ですので、AD2 の値が 500 以下だったら何もしないという処理になり、プログラムは先へ進みません。次にタクトスイッチを押して信号線と電源線をショートさせると AD2 の値がほぼ最大値を示し、これで条件から外れるので、プログラムは次の処理に移ります。

3.2 起き上がって向きを変え、適当な距離だけフラッグへ近づく

スタート地点では、ロボットの姿勢とフラッグへの向きは決まっていますので、起き上がって向きを 180 度変え、適当な距離だけフラッグまで前進を行います。

3.3 倒れていた場合は起き上がる

加速度センサーは KCB-1 の AD1 ポートへ接続します。

加速度センサーの値が 160 を下回る場合はロボットが仰向け状態であると判断し、仰向けからの起き上がりモーション 10 番を再生します。加速度センサーの値が 400 を超える場合は、ロボットがうつぶせ状態であると判断し、うつぶせからの起き上がりモーション 11 番を再生します。加速度センサーの閾値は、センサーの取り付け位置などで変化します。必ずシリアルタームで値を確認してからプログラミングしてください。

```
adn = ad_read(1);           //加速度センサーで本体の状態を確認
if (adn < 160){             //加速度センサーの値が 160 だったら仰向けと判断し、
    rcb3_motion_play(10);   //モーション 10 番を再生する。
}
else if (adn > 400){        //加速度センサーの値が 400 だったらうつ伏せと判断し、
    rcb3_motion_play(11);   //モーション 11 番を再生する。
}
```

3.4 壁に近づいている場合には、旋回して避ける

旋回して壁から回避するプログラムは次のようになっています。

```
adv5 = ad_read(5);         //足のセンサーで状況を読み取る
adv6 = ad_read(6);
```

```

if ((adv5 >= 250) || (adv6 >= 250)){ //目標の前にいると判断
  if (adv6 <= adv5){ //右足が左足よりも壁に近かったときの処理
    if (adv6 <= 200){ //右側の値だけ上がっていたら、目標の前にいると判断
      else{
        rcb3_motion_play(7); //壁をよけるために旋回モーション（左）を三回再生
        rcb3_motion_play(7);
        rcb3_motion_play(7);
      }
    }
  }
  else{ //左足が右足よりも壁に近かったときの処理
    if (adv5 <= 200){
      else{
        rcb3_motion_play(8); //壁をよけるために旋回モーション（右）を三回再生
        rcb3_motion_play(8);
        rcb3_motion_play(8);
      }
    }
  }
}
}
}

```

左右の足に取り付けられた測距センサーの値を読み取って、左右どちらかの距離センサーの値が250を超えた場合は物体に近づいていると判断します。次にその物体が、壁かフラッグのどちらかを判断するため、片足の値が250を超えていて、もう片方が200以下であるときは、目の前にあるものが壁でない可能性が高いと判断し、回避行動は取りません。

回避すると判断した場合、左右の足と壁との距離を判断して回避する方向を決めます。モーションを三回再生しているのは、フィールドの角にはまってしまわないようにするためです。

3.5 適当な距離だけ前進する

物体を回避したり、目標までの距離がまだ遠い場合は前進モーションを繰り返し再生します。

3.6 首を旋回させながら測距センサーで距離を計測し、目標を見つける

```

//東部を移動し、センサーの値を読み取る関数
void area_radar (
  BYTE p_n, // 頭部サーボモーターのポートナンバー
  BYTE s, // 検索開始位置 (頭部サーボモーターの開始位置)
  BYTE t // 検索開始位置 (頭部サーボモーターの開始位置)
)
{
  BYTE i;
  unsigned int pos1;

  for(i = s; i <= t; i++){
    pos1 = i * SEARCH_STEP + SEARCH_START;
    pwm_out(p_n, pos1);
    wait(30000);
    adj[i]=(int)s_sonic_average(); //読み取ったAD値を代入する
  }
}

//バブルソート関数
void bubble_sort (

```

```

BYTE s, // 検索開始位置
BYTE t // 検索終了位置
)
{
BYTE i, j;
unsigned int tmp;
for(i = s; i <= t-1; i++){
for (j = i+1; j <= t; j++){
if (adj[j] > adj[i]){ //アナログの最大値を求める
tmp = adj[j];
adj[j] = adj[i];
adj[i] = tmp;

tmp = adk[j]; //ken1[t] > ken1[s]だったら ken2 も同じ処理を行う
adk[j] = adk[i]; //最大値のモーション番号は ken2[0]に代入される
adk[i] = tmp;
}
}
}
}
}

```

最初に頭部サーボモーターが旋回開始位置に移動するまで少しだけ待ちます。

次にサーボモーターの回転角度を 44 の位置から 144 の位置まで 4 ごとに分割し、頭部を旋回させながら、その度測距センサーから値を読み込み、配列へ代入しておきます。サーボが目標の地点に到着するまでしばらく待ったあと AD7 ポートからアナログ値を読み取り、配列 adj に保存します。

上記の処理を終了した後、配列 adj の値をバブルソート法で大きい順番に並べ替えます。同時に方向を示す配列 adk を並べ替えると、測距センサーの値が最大 (adj[0]) となった向き (adk[0]) を特定することが出来ます。KCB-1 で PWMサーボを制御する詳しい解説は、KCB-1 サポートページの『[KRS-788HV を KCB-1 で動かす①・②](#)』をご覧ください。

3.7 機体の向きを変える

バブルソートしたデータを元に switch でモーションを分岐させます。

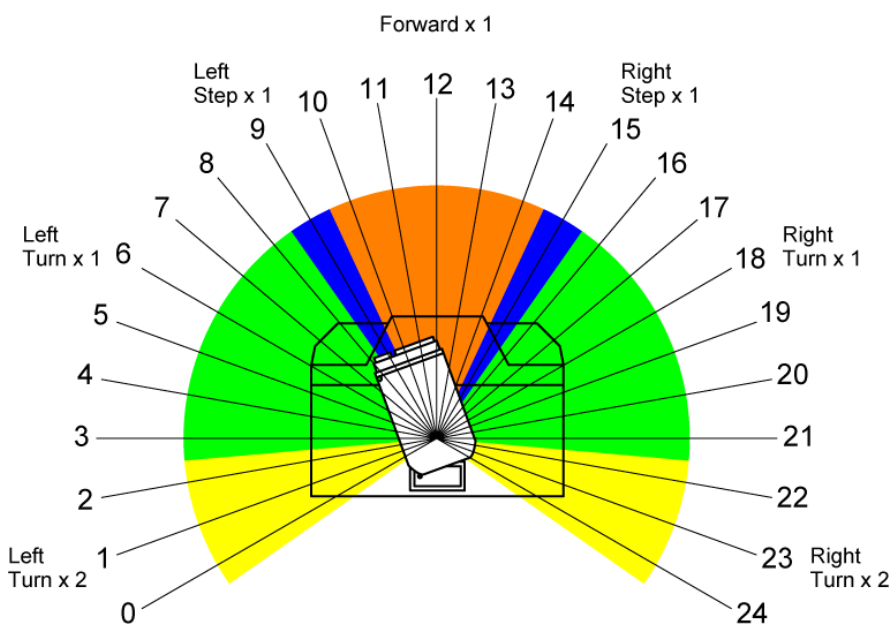


図 9: 物体が一番近い方向とモーション

3.8 目標までの距離が遠い場合は、3番へ戻る

目標までの距離が遠い場合は、繰り返し処理の先頭へ戻ります。

3.9 目標の目の前まできたら特定のモーションを再生させ、フラッグを倒す

```
adv2 = ad_read(7);           // 頭部センサーで目標までの距離を測る
if(adv2 >= 180){           // もし目標が閾値 320 以上の場所にあったら
    rcb3_motion_play(18);   // 発見モーション再生
}
```

頭部超音波センサーで目標までの距離が 180 以上の場合は、フラッグへ十分近づいたと判断して、目標を倒すモーションを再生します。

4 モーション再生について

モーション再生の仕方は二種類あります。rcb3_motion_play() を使う方法と、ロボットコントローラー KRS-1/3 で使用しているコントロールコードを送る方法です。

4.1 rcb3_motion_play —コマンド再生—

rcb3_motion_play 関数は RCB-3 に登録されたモーションの番号を指定してモーションを再生します。歩いている最中に違うモーションを再生して転んでしまったりすることの無いように、KCB-1 は命令送信後に RCB-3 がモーション終了の返事を返すまで待機状態となります。そのため rcb3_motion_play 関数を使用する際には、必ず HeartToHeart のオプション画面で「モーション再生後返事をもらう」にチェックを入れてください。また、KCB-1 の COM ポートと RCB-3 の COM ポートを **クロスケーブルで接続**してください。

この関数はモーション再生終了のお知らせを RCB-3 から受け取るまで次のモーションを再生しないように設計していますので、歩くなどの連続モーションでも 1 度だけしか再生しません。

4.2 rcb3_put_XBcode —コントロールコード再生—

rcb3_put_2Bcode, rcb3_put_7Bcode 関数は、KRC-1/3 が送信するボタン番号を命令として送る関数です。KRC ではボタン番号を 2 バイトのコントロール入力値として変換して送信しますが、これをエミュレート(模倣)したものが rcb3_put_2Bcode, rcb3_put_7Bcode 関数で、それぞれ 2 バイトコード(KRC-1)、7 バイトコード(KRC-3)をエミュレートします。

この関数では KRC と同じ信号を送信するため、連続歩行などのモーション内分岐が可能です。ただし KRC と同様にモーションの終了が分からないため、**モーション再生のタイミングを間違えると命令が無視される場合がありますので、モーション再生時間をきちんと把握し、調節する必要があります。**

```
rcb3_put_7Bcode (BTN7B_LU);   // 2 歩前進
wait (500000);               // 前進モーションを再生させる時間を指定します
rcb3_put_7Bcode (BTN7B_NP);   // コントロールコードをニュートラルに戻します
wait (250000);               // 停止するまで待つ
```

上記プログラムはコントロール入力値を使用したモーション再生プログラムです。モーションを終了する場合にはニュートラルを入力してください。なお、前進・後退などのコントロール入力値については、マニュアルを参照してください。ニュートラルを入力しても、ロボットがきちんと停止するまで待つ必要があります。

また、wait でモーションの終了を待つ時間を最小限に抑えるために、RCB-3 の空いているポートを利用してモーション終了の合図をもらう方法を、KCB-1 サポートページにて紹介しています。『[コントロールコード再生 モーションの終了を知る方法](#)』をご覧ください。

二種類のモーションに関する解説はサポートページの『[2種類のモーション設定の違い](#)』もご覧ください。

5 まとめ

ビーチフラッグ競技では、(1)フラッグへの向きと、(2)フラッグまでの距離を正しく計測することがポイントとなります。単に前進するだけでフラッグを倒してしまうとセンターサークルから出されてしまいますので、スタートからフラッグを倒すまでの一連の動作をイメージしてプログラムを作るとうまくいくでしょう。みなさんの果敢なチャレンジを期待します。