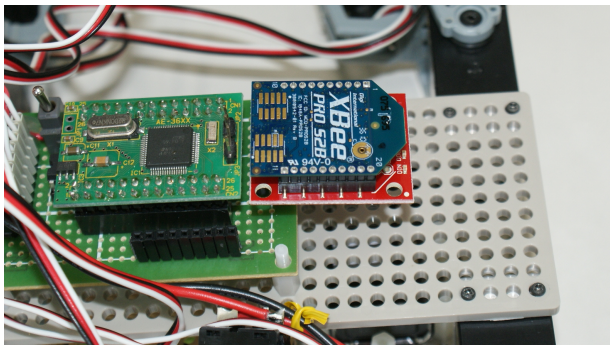

図 15.7　XBee を利用する場合

15.3　モーション (動き) を考える

8 軸 4 足歩行ロボットの動作を考えましょう。

4 軸の場合と異なるのは、常に胴体を浮かせた状態で動くことができるということです。

そのためには少なくとも対角の脚が接地していなくてはなりません。1 本ずつ脚を動かすことによってこれは常に可能ですが、ここではすべての動作を 4 ポーズで作れるように、すべての脚を動かしながら、どちらかの対角の脚 2 本は常に接地するように動きを作りましょう。

ポーズ数を制限したのは、マイコンのメモリの関係です。

なお、前後の脚を同じようにバタフライ歩行させることで、4 足歩行ロボットでもバタフライ形の歩行をさせることが可能です。ここではプログラムに入れませんでしたが、興味のある方はやってみてください。

ここで掲載した動作は、今回のロボットで実現できる動きの一部です。他にもどのような動作が可能か考えてみてください。

15.3.1 前進 (クロール形)

まずは、左右の前脚を交互に前に出し、対角の後ろ脚を交互に前に出すことによって前進させることを考えましょう。

クロール形の前進は、以下の図 15.8 のように 2 本の前足を動かすことで実現できます。

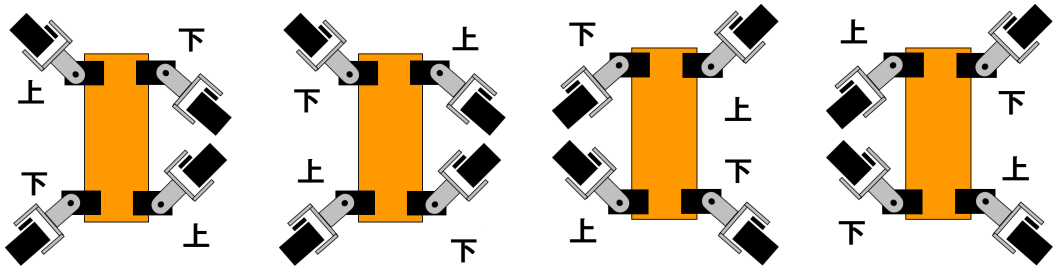

図 15.8　前進 (クロール形) 上から見た図 (画面上が前)

図 15.8 は以下のような動作になります。

- 左前脚と右後脚を上げ、前に出す。右前脚と左後脚は下にし、後にさげる。
- 左前脚と右後脚を下にし、右前脚と左後脚は上げる。
- 左前脚と右後脚を後ろにさげ、右前脚と左後脚は前に出す。
- 左前脚と右後脚を上げ、右前脚と左後脚はさげる。

15.3.2　後退 (クロール形)

後退は図 15.8 を逆から (右図から) 再生すれば実現できます。

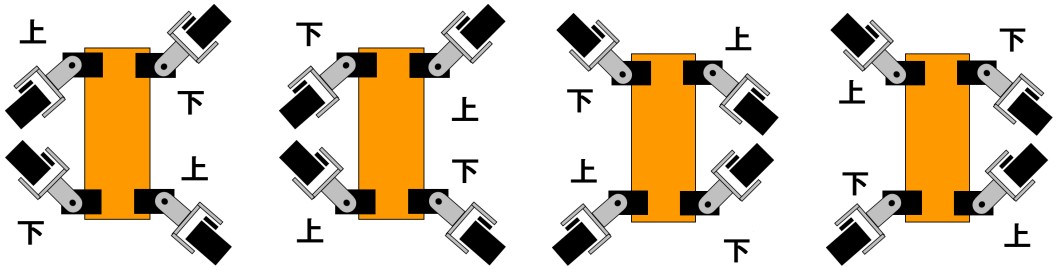

図 15.9　後退 (クロール形) 上から見た図 (画面上が前)

図 15.9 は以下のような動作になります。

- 左前脚と右後脚を上げ、後ろにさげる。右前脚と左後脚は下にし、前に出す。
- 左前脚と右後脚を下にし、右前脚と左後脚は上げる。
- 左前脚と右後脚を上げ、前に出す。右前脚と左後脚は、後にさげる。
- 左前脚と右後脚を下にし、右前脚と左後脚は下にする。

15.3.3 前進 (バタフライ形)

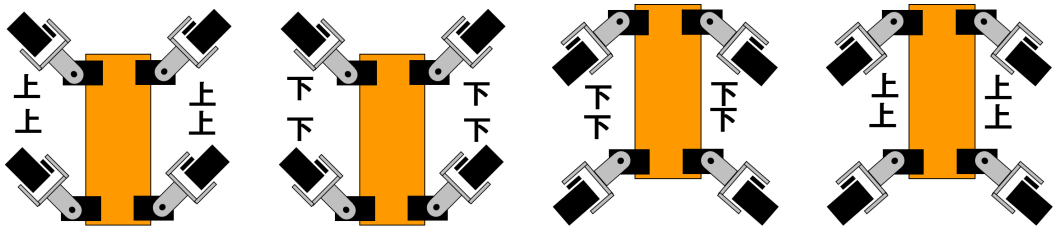

図 15.10　前進 (バタフライ形) 上から見た図 (画面上が前)

バタフライ型は、体を上下させながら進んでいきます。

図 15.10 は以下のような動作になります。

- 全ての脚を上げ、前に出す。
- 全ての脚を下にする。
- 全ての脚を、後にさげる。
- 全ての脚を上にする。

後退は逆から (右図から) 再生することで実現できます。

15.3.4 右旋回

旋回は、対角の脚が接地するように、作りました。

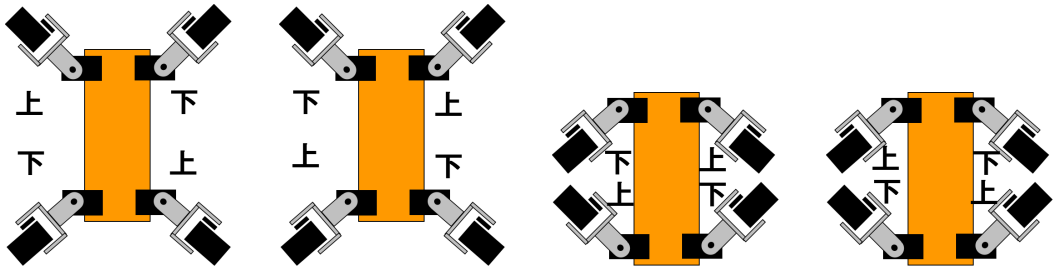

図 15.11　右旋回 上から見た図 (画面上が前)

15.3.5 左旋回

左旋回は、右旋回を逆から (右図から) 再生すると実現できます。

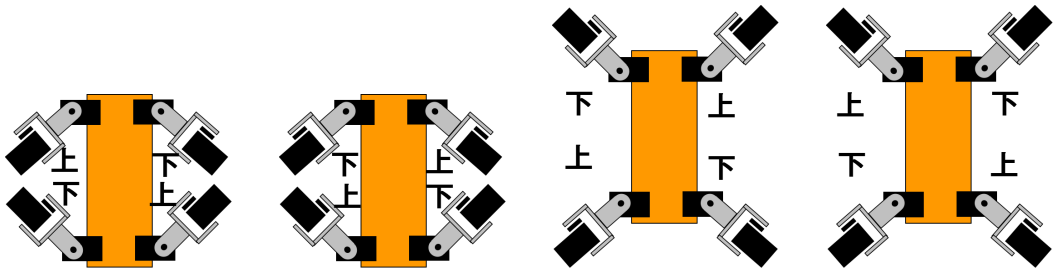

図 15.12　左旋回 上から見た図 (画面上が前)

15.4　自律して動かす

では、P.364 で紹介したサンプルプログラム、「シリアルからサーボモータを動かす (IC で割り振り)」を用いてモータのデータを作成してみましょう。

以下に、作ったデータをもとに動きを再生するプログラムを示します。

データはそのまま配列に代入するとメモリが足りなくなるようですので、100 分の 1 にして利用することにしました。再生する際に 100 倍して元の値に戻します。

今回のプログラムを動作する際は、シリアルのケーブルを外しても結構です。

自律して動きます。

データは 4 つのモーションが入っていますが、動作は右旋回だけです。

07_TMRW07.c

```
/************************************************************************/
/*                                                                      */
/*  FILE        :07_TMRW07.c                                            */
/*  IC(TC4051B) を用いる 4 足ロボットデータ再生                         */
/*                                                                      */
/*  COM P82/FTIOB                                                       */
/*  A P10                                                               */
/*  B P11                                                               */
/*  C P12                                                               */
/************************************************************************/
#include "iodefine.h"
#include<machine.h>

#define P1_MASK 0x07

#define PWM_MAX_DUTY 11500 //2.3mS
#define PWM_MIN_DUTY  3500 //0.7mS
#define PWM_MID_DUTY ((PWM_MAX_DUTY+PWM_MIN_DUTY)>>1) //1.5mS
#define PWM_PERIOD   (12500-1) //2.5mS
#define maX  90 //1.8mS の 1/100
#define miN  60 //1.2mS の 1/100
#define miD  75 //1.5mS の 1/100

#define SRV_CH_NUM 8 /* モータ数 */
#define MOTION_NUM 4 /* モーション数 */

int srv_ch;     /* PWM の出力先 */
int srv_flg;    /* 20ms をカウント */
unsigned short pwm_duty[SRV_CH_NUM]; /* PWM のデューティのデータ */

/* PWM のデューティのデータ */
unsigned char forward_data[MOTION_NUM][SRV_CH_NUM]=
    {{ miD, miN, miN, miN, maX, maX, maX, miD},
```

```
34        { maX, miN, miN, miD, miD, maX, maX, miN},
35        { maX, maX, maX, miD, miD, miN, miN, miN},
36        { miD, maX, maX, miN, maX, miN, miN, miD}};
37  unsigned char backward_data[MOTION_NUM][SRV_CH_NUM]=
38      {{ miD, maX, maX, miN, maX, miN, miN, miD},
39        { maX, maX, maX, miD, miD, miN, miN, miN},
40        { maX, miN, miN, miD, miD, maX, maX, miN},
41        { miD, miN, miN, miN, maX, maX, maX, miD}};
42  unsigned char left_data[MOTION_NUM][SRV_CH_NUM]=
43      {{ miD, maX, miN, miN, maX, miN, maX, miD},
44        { maX, maX, miN, miD, miD, miN, maX, miN},
45        { maX, miN, maX, miD, miD, maX, miN, miN},
46        { miD, miN, maX, miN, maX, maX, miN, miD}};
47  unsigned char right_data[MOTION_NUM][SRV_CH_NUM]=
48      {{ miD, miN, maX, miN, maX, maX, miN, miD},
49        { maX, miN, maX, miD, miD, maX, miN, miN},
50        { maX, maX, miN, miD, miD, miN, maX, miN},
51        { miD, maX, miN, miN, maX, miN, maX, miD}};
52
53
54  /*******************************
55     PWM 出力端子を初期化
56  *******************************/
57  void PwmInit(void)
58  {
59    srv_ch=0;
60    srv_flg=0;
61    IO.PCR1|=0x07;
62    IO.PDR1.BYTE|=(srv_ch & P1_MASK);
63    IO.PDR1.BYTE&=(srv_ch | (~P1_MASK));
64    set_imask_ccr(1);
65    TW.TMRW.BYTE=0x49;  /* FTIOB 端子を PWM 出力に設定 */
66    TW.TCRW.BYTE=0xA0;  /* 内部クロックの 1/4, コンペアマッチ B で 1 出力 */
67    TW.TIERW.BYTE=0x71; /* A の割り込みを可能にする */
68    TW.TSRW.BYTE=0x70;  /* 割り込みフラグをクリア */
69    TW.TCNT=0x0000;     /* TCN の初期化 */
70    TW.GRA=PWM_PERIOD;  /* 周期 (2.5mS) */
71    TW.GRB=PWM_PERIOD-PWM_MID_DUTY;
72    TW.TMRW.BIT.CTS=1;  /* TCNT カウンタスタート */
73    set_imask_ccr(0);
74
75  }
76
77  /*********************************
78    サーボ動作開始
79   *********************************/
80  void PwmStart(void)
81  {
82    TW.TMRW.BIT.CTS=1;
83  }
84
85  /*********************************
86    サーボ動作停止
87   *********************************/
88  void PwmStop(void)
89  {
90    TW.TMRW.BIT.CTS=0;
91  }
92
93  /*********************************
94    デューティの設定
95   *********************************/
96  void PwmSetDuty(int ch, unsigned char duty)
97  {
98    unsigned short c_duty;
99
100   c_duty=(unsigned short)(duty*100);
101
102   /* サーボチャンネルが範囲外なら無視 */
103   if(ch >= SRV_CH_NUM){
104     return ;
105   }
106
107   if(c_duty < PWM_MIN_DUTY){          /* デューティを最小値以上に限定 */
108     c_duty = PWM_MIN_DUTY;
109   }else if(c_duty > PWM_MAX_DUTY){    /* デューティを最大値以下に限定 */
110     c_duty = PWM_MAX_DUTY;
111   }
112   /* デューティを設定 */
113   pwm_duty[ch] = c_duty;
114 }
115
116 /*********************************
117   main 関数
118  *********************************/
119 void main(void)
120 {
121   int i, j=0;
122
123   PwmInit(); /* PWM の初期化 */
124
```

```
125    while(1){
126      /* 次のモーションデータを設定 */
127      for(i=0; i<SRV_CH_NUM; i++){
128        PwmSetDuty(i, right_data[j][i]);
129      }
130      while(srv_flg<20){
131        ;
132      }
133      srv_flg=0;
134      j++;
135      if(j>=MOTION_NUM){
136        j=0;
137      }
138    }
139  }
140
141  /*********************************
142    割り込み関数
143   *********************************/
144  __interrupt(vect=21) void INT_TimerW(void)
145  {
146    /* 2.5mS ごとのコンペアマッチ */
147    TW.TSRW.BIT.IMFA=0;
148    srv_ch++;
149    if(srv_ch>= SRV_CH_NUM){
150      srv_ch=0; /* モータ出力先を 0 に戻す */
151      srv_flg++;
152    }
153    IO.PDR1.BYTE|=(srv_ch & P1_MASK);
154    IO.PDR1.BYTE&=(srv_ch | (~P1_MASK));
155    TW.GRB=PWM_PERIOD-pwm_duty[srv_ch];
156  }
```

End Of List

実行結果

自律して動作。右旋回する。

課題 15.4.1　(提出) 4 つの動作を順番の再生する

前進、後退、右旋回、左旋回を順番に再生するようにしてください。

各動作の再生時間は約 10 秒とし、左旋回の後は停止してください。

プロジェクト名：e07_TMRW07

15.5　シリアルからコントロールする

ここでは、ロボットを外部からコントロールすることを考えてみましょう。まずは、シリアル通信を用いてロボットの制御をしてみます。

シリアル通信の関数は、第 11 章で作った関数を利用しません。受信バッファがデータをためてしまうと、シリアルからの指令がたまってしまい、現在の指令がなかなか再生されないからです。

今回の目的のためには、受信バッファサイズは 1 バイトが良いでしょう。第 11 章で作った関数でそのように定義し直しても良いのですが、ヘッダファイルの中身を書き換えてしまうと前に作ったプログラムに影響が出ます。

バッファサイズを定義できるように関数を拡張するのが良いかもしれませんが、今回はプログラム内に書き込むことにしました。

ターミナルソフトから送信するデータの仕様は、以下のように簡単なものにしました。

モーション再生ソフト仕様

- f : 前進。
- b : 後退。
- r : 右旋回。
- l : 左旋回。

ボーレート 19200bps。

07_TMRW08.c

```
/************************************************************************/
/*                                                                      */
/*  FILE        :07_TMRW08.c                                            */
/*  IC(TC4051B) を用いる 4 足ロボットデータ再生                         */
/*                                                                      */
/*  COM P82/FTIOB                                                       */
/*  A P10                                                               */
/*  B P11                                                               */
/*  C P12                                                               */
/************************************************************************/

#include "iodefine.h"
#include<machine.h>

#define TXD1F SCI3.SSR.BIT.TDRE //送信データフラグ
#define TD1D SCI3.TDR           //送信データ

#define P1_MASK 0x07
#define WAIT_LOOP 0x200000
#define CHECK_LOOP 0x200000
#define CR 0x0D
#define LF 0x0A

#define PWM_MAX_DUTY 11500 //2.3mS
#define PWM_MIN_DUTY  3500 //0.7mS
#define PWM_MID_DUTY ((PWM_MAX_DUTY+PWM_MIN_DUTY)>>1) //1.5mS
#define PWM_PERIOD   12500 //2.5mS
#define maX  90 //1.8mS の 1/100
#define miN  60 //1.2mS の 1/100
#define miD  75 //1.5mS の 1/100

#define SRV_CH_NUM 8 /* モータ数 */
#define MOTION_NUM 4 /* モーション数 */

int srv_ch;     /* PWM の出力先 */
int srv_flg;    /* 20ms をカウント */
unsigned short pwm_duty[SRV_CH_NUM]; /* PWM のデューティのデータ */
char rx1_buff;

/* PWM のデューティのデータ */
unsigned char forward_data[MOTION_NUM][SRV_CH_NUM]=
    {{ miD, miN, miN, miN, maX, maX, maX, miD},
     { maX, miN, miN, miD, miD, maX, maX, miN},
     { maX, maX, maX, miD, miD, miN, miN, miN},
     { miD, maX, maX, miN, maX, miN, miN, miD}};
unsigned char backward_data[MOTION_NUM][SRV_CH_NUM]=
    {{ miD, maX, maX, miN, maX, miN, miN, miD},
     { maX, maX, maX, miD, miD, miN, miN, miN},
     { maX, miN, miN, miD, miD, maX, maX, miN},
     { miD, miN, miN, miN, maX, maX, maX, miD}};
unsigned char left_data[MOTION_NUM][SRV_CH_NUM]=
    {{ miD, maX, miN, miN, maX, miN, maX, miD},
     { maX, maX, miN, miD, miD, miN, maX, miN},
     { maX, miN, maX, miD, miD, maX, miN, miN},
     { miD, miN, maX, miN, maX, maX, miN, miD}};
unsigned char right_data[MOTION_NUM][SRV_CH_NUM]=
```

```
57      {{ miD, miN, maX, miN, maX, maX, miN, miD},
58       { maX, miN, maX, miD, miD, maX, miN, miN},
59       { maX, maX, miN, miD, miD, miN, maX, miN},
60       { miD, maX, miN, miN, maX, miN, maX, miD}};
61  unsigned char home_data[MOTION_NUM][SRV_CH_NUM]=
62      {{ maX, miD, miD, miN, maX, miD, miD, miN},
63       { maX, miD, miD, miN, maX, miD, miD, miN},
64       { maX, miD, miD, miN, maX, miD, miD, miN},
65       { maX, miD, miD, miN, maX, miD, miD, miN}};
66
67  typedef enum{
68    br19200=32,
69    br38400=15,
70    br57600=10
71  }BaudRate;
72
73  /*********************************
74     シリアルポートを初期化 :Sci1Init
75  *********************************/
76  void Sci1Init(BaudRate b)
77  {
78    SCI3.SCR3.BYTE = 0x00;  /* TE、RE クリア   CKE 初期化 */
79    SCI3.SMR.BYTE  = 0x00;  /* 調歩同期、8 ビット、パリティなし、ストップ 1 ビット n=0 */
80    SCI3.BRR  = b;
81    IO.PMR1.BIT.TXD = 1;  /* TXD ポートイネーブル */
82    SCI3.SCR3.BYTE = 0x70;  /* TE RE 受信割込み */
83  }
84
85  /*******************************
86     1 バイト受信関数 (割り込み有り)
87  *******************************/
88  __interrupt(vect=23) void INT_SCI3(void)
89  {
90    if(SCI3.SSR.BIT.RDRF){
91      rx1_buff=SCI3.RDR;
92    }
93  }
94
95  /************************************************
96  *    1 文字受信 : Sci1Read
97  ************************************************/
98  char Sci1Read(void)
99  {
100   return(rx1_buff);
101 }
102
103 /*******************************
104    PWM 出力端子を初期化
105 *******************************/
106 void PwmInit(void)
107 {
108   srv_ch=0;
109   srv_flg=0;
110   IO.PCR1|=0x07;
111   IO.PDR1.BYTE|=(srv_ch & P1_MASK);
112   IO.PDR1.BYTE&=(srv_ch | (~P1_MASK));
113   set_imask_ccr(1);
114   TW.TMRW.BYTE=0x49;  /* FTIOB 端子を PWM 出力に設定 */
115   TW.TCRW.BYTE=0xA0;  /* 内部クロックの 1/4, コンペアマッチ B で 1 出力 */
116   TW.TIERW.BYTE=0x71; /* A の割り込みを可能にする */
117   TW.TSRW.BYTE=0x70;  /* 割り込みフラグをクリア */
118   TW.TCNT=0x0000;     /* TCN の初期化 */
119   TW.GRA=PWM_PERIOD;  /* 周期 (2.5mS) */
120   TW.GRB=PWM_PERIOD-7500;        /* 1.5mS */
121   TW.TMRW.BIT.CTS=1;  /* TCNT カウンタスタート */
122   set_imask_ccr(0);
123
124 }
125
126 /*********************************
127   サーボ動作開始
128  *********************************/
129 void PwmStart(void)
130 {
131   TW.TMRW.BIT.CTS=1;
132 }
133
134 /*********************************
135   サーボ動作停止
136  *********************************/
137 void PwmStop(void)
138 {
139   TW.TMRW.BIT.CTS=0;
140 }
141
142 /*********************************
143   デューティの設定
144  *********************************/
145 void PwmSetDuty(int ch, unsigned char duty)
146 {
147   unsigned short c_duty;
```

```
148
149    c_duty=(unsigned short)(duty*100);
150
151    /* サーボチャンネルが範囲外なら無視 */
152    if(ch >= SRV_CH_NUM){
153      return ;
154    }
155
156    if(c_duty < PWM_MIN_DUTY){          /* デューティを最小値以上に限定 */
157      c_duty = PWM_MIN_DUTY;
158    }else if(c_duty > PWM_MAX_DUTY){    /* デューティを最大値以下に限定 */
159      c_duty = PWM_MAX_DUTY;
160    }
161    /* デューティを設定 */
162    pwm_duty[ch] = c_duty;
163  }
164
165  /*********************************
166    main 関数
167   *********************************/
168  void main(void)
169  {
170    int i, j=0;
171    unsigned long wl;
172    char r_data;
173    unsigned char *data;
174
175     for(wl=0; wl<WAIT_LOOP; wl++){ /* XBee 使用時のための待ちループ */
176       ;
177     }
178
179    Sci1Init(br19200);
180    PwmInit(); /* PWM の初期化 */
181    Sci1Write('A');
182    data=home_data[0];
183    while(1){
184      /* 次のモーションデータを設定 */
185      for(i=0; i<SRV_CH_NUM; i++){
186        PwmSetDuty(i, *(data+j*SRV_CH_NUM+i));
187      }
188      while(srv_flg<20){ /* 0.4s 待ち */
189        ;
190      }
191      srv_flg=0;
192      j++;
193      if(j>=MOTION_NUM){
194        j=0;
195        r_data=Sci1Read();
196        switch(r_data){
197          case 'f':  /* 前進 */
198            data=forward_data[0];
199            break;
200          case 'b':  /* 後退 */
201            data=backward_data[0];
202            break;
203          case 'l':  /* 左旋回 */
204            data=left_data[0];
205            break;
206          case 'r':  /* 右旋回 */
207            data=right_data[0];
208            break;
209          default:  /* その他はホームポジション */
210            data=home_data[0];
211          break;
212        }
213      }
214    }
215  }
216
217  /*********************************
218    割り込み関数
219   *********************************/
220  __interrupt(vect=21) void INT_TimerW(void)
221  {
222    /* 2.5mS ごとのコンペアマッチ */
223    TW.TSRW.BIT.IMFA=0;
224    srv_ch++;
225    if(srv_ch>= SRV_CH_NUM){
226      srv_ch=0; /* モータ出力先を 0 に戻す */
227      srv_flg++;
228    }
229    IO.PDR1.BYTE|=(srv_ch & P1_MASK);
230    IO.PDR1.BYTE&=(srv_ch | (~P1_MASK));
231    TW.GRB=PWM_PERIOD-pwm_duty[srv_ch];
232  }
```

End Of List

実行結果

シリアルケーブルを通して、ターミナルソフトから指令を送る。
前進、後退、右旋回、左旋回の動作を指令可能。

課題 15.5.1　(提出) 他の動作を登録する

ここで紹介した前進 (f)、後退 (b)、右旋回 (r)、左旋回 (l) 以外の動作を再生するように変更してください。

プロジェクト名：e07_TMRW08