ジャイロ＋加速度センサー
基板製作キット
C 言語倒立制御プログラム
解説マニュアル

第 1.02 版
2015 年 4 月 20 日
株式会社日立ドキュメントソリューションズ

注 意 事 項 (rev.6.0H)

著作権

・本マニュアルに関する著作権は株式会社日立ドキュメントソリューションズに帰属します。
・本マニュアルは著作権法および、国際著作権条約により保護されています。

禁止事項

ユーザーは以下の内容を行うことはできません。
・第三者に対して、本マニュアルを販売、販売を目的とした宣伝、使用、営業、複製などを行うこと
・第三者に対して、本マニュアルの使用権を譲渡または再承諾すること
・本マニュアルの一部または全部を改変、除去すること
・本マニュアルを無許可で翻訳すること
・本マニュアルの内容を使用しての、人命や人体に危害を及ぼす恐れのある用途での使用

転載、複製

本マニュアルの転載、複製については、文書による株式会社日立ドキュメントソリューションズの事前の承諾が必要です。

責任の制限

本マニュアルに記載した情報は、正確を期すため、慎重に制作したものですが万一本マニュアルの記述誤りに起因する損害が生じた場合でも、株式会社日立ドキュメントソリューションズはその責任を負いません。

その他

・本マニュアルに記載の情報は本マニュアル発行時点のものであり、株式会社日立ドキュメントソリューションズは、予告なしに、本マニュアルに記載した情報または仕様を変更することがあります。製作に当たりましては、最新の内容を確認いただきますようお願いします。
・すべての商標および登録商標は、それぞれの所有者に帰属します。

連絡先

株式会社 日立ドキュメントソリューションズ
〒135-0016 東京都江東区東陽六丁目 3 番 2 号 イースト 21 タワー
E-mail：himdx.m-carrally.dd@hitachi.com

目次

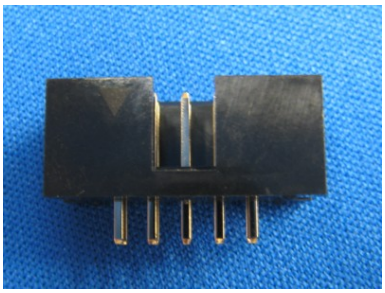
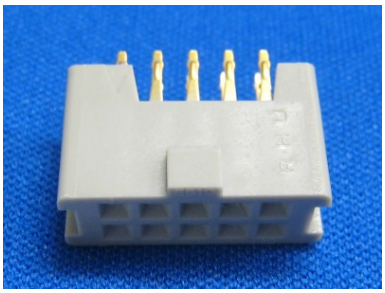
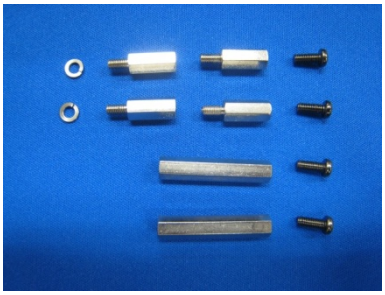
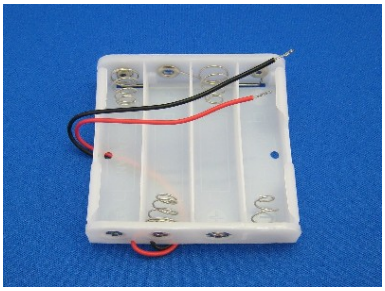
1. 概要

本書では、ジャイロ＋加速度センサー基板製作キットを使用した、倒立制御プログラムの解説を行います。
開発環境の構築方法やプログラムのビルド、書き込みについては、「ミニマイコンカー製作キット Ver.2　C 言語走行プログラム解説マニュアル」の「3. インストール」「4. ミニマイコンカーVer.2 の動作確認」を参照してください。

2. 倒立制御ロボットの作成

倒立制御ロボットは、ミニマイコンカー製作キット Ver.2 を加工し、ジャイロ＋加速度センサー基板製作キットを取り付けて作成します。

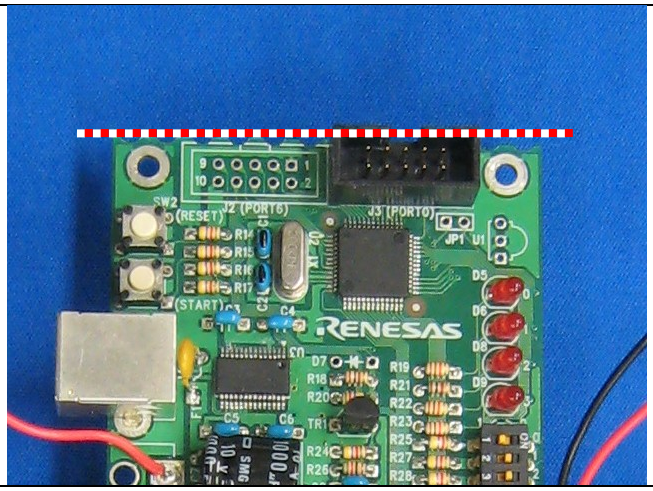
2.1 必要な部品

部品名	型名	写真
10P オスコネクター	HIF3FC-10PA 2.54DSA (M-S42)	
10P メスコネクター	8510-4500PL (M-S209)	
固定用部品	スプリングワッシャ×2 枚 スタッド 13mm×4 本 スタッド 30mm×2 本 丸ビス 8mm×4 本	
電池ボックス	(M-S104)	

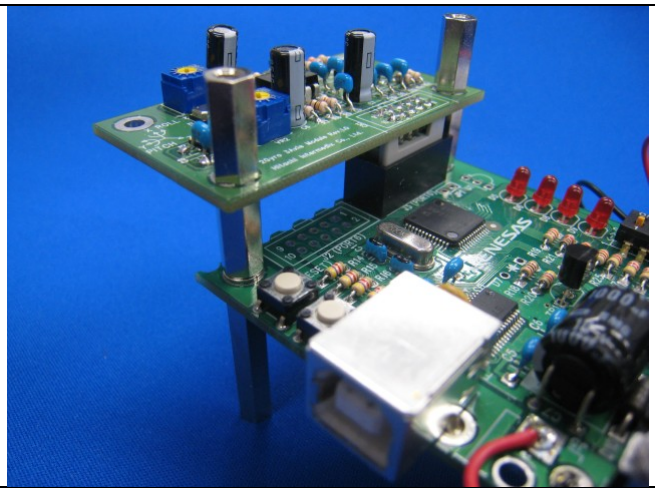
2.2　ミニマイコンカー製作キット Ver.2 の加工

ジャイロ＋加速度センサー基板を使用するためには、ミニマイコンカー基板製作キット Ver.2 の加工を行う必要があります。

2.2.1　アナログ信号入力部分

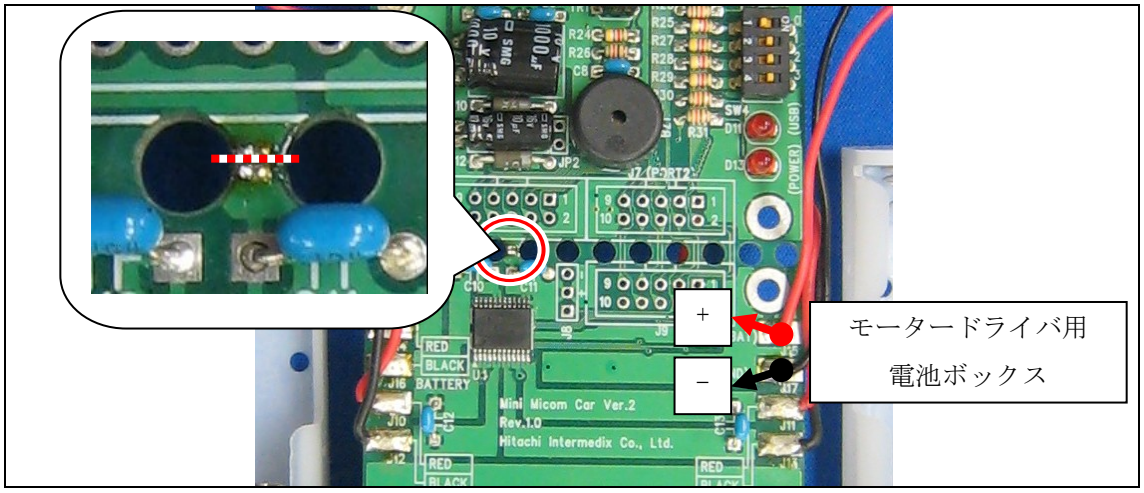


ミニマイコンカー基板製作キット Ver.2 のセンサー部分を切り離し、A/D 変換が割り当てられているポートを使用できるようにします。写真のように、ミニマイコンカー基板製作キット Ver.2 を点線部分でカットします。

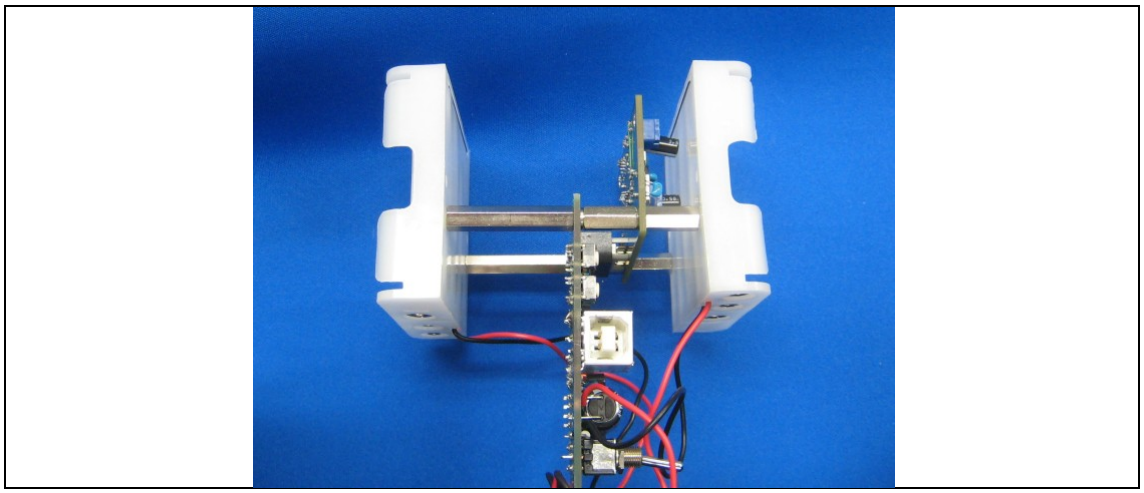


写真のように、ミニマイコンカー製作キット Ver.2 の J3 の表面に 10 ピンオスコネクターを取り付け､ジャイロ＋加速度センサー基板製作キットの CN1 の裏面に 10 ピンメスコネクターを取り付け、固定用部品を使用して固定します。

2.2.2　電源部分

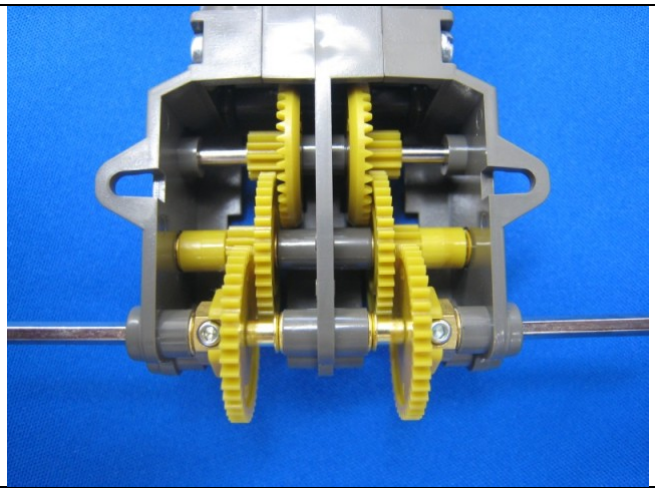


電源を CPU 部分とモータードライバ部分で分け、モーターからのノイズがセンサーの A/D 変換に影響しないようにします。写真のように、パターンを点線部分でカットして、モータードライバ用電池ボックスを追加します。

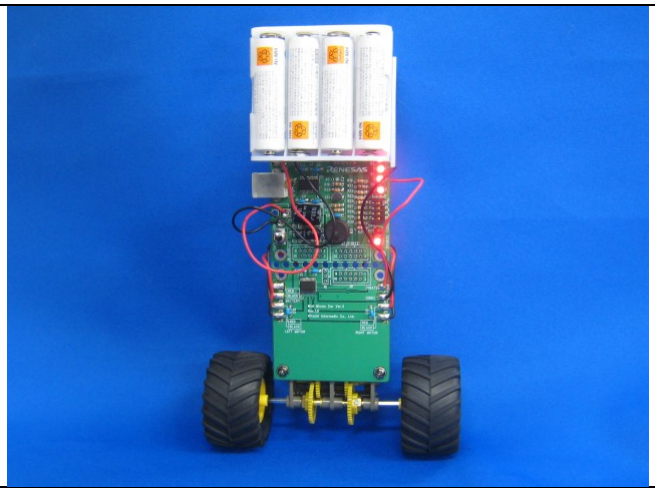


ジャイロ＋加速度センサー基板製作キットを固定している部品に電池ボックスを取り付けます。新たに購入した電池ボックスには穴があいていないので、ミニマイコンカー製作キット Ver.2 に付属している電池ボックスと同じ位置に穴をあけます。

2.2.3　ギヤボックスのギヤ組み換え



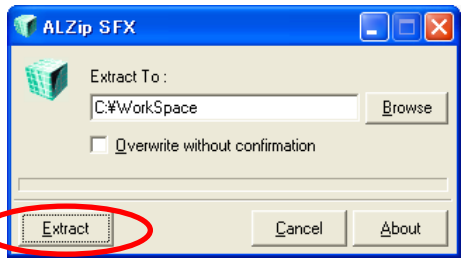
写真のように、ギヤを組み換えます。



以上で、部品の加工は終了です。

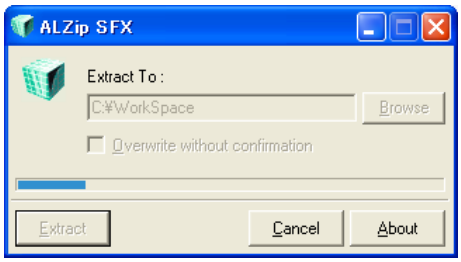
3. ワークスペースのインストール

株式会社日立ドキュメントソリューションズのマイコンカーラリー販売ページからワークスペースのインストーラー「mini_mcr2_gyro_vxxx.exe」（xxx はバージョン）をダウンロードします。
ダウンロードした「mini_mcr2_gyro_vxxx.exe」をダブルクリックし、インストーラーを実行します。

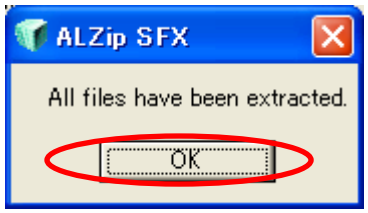


表示されたデフォルトのインストール先のフォルダ「c:¥WorkSpace」を確認して、「Extract」をクリックします。

《**補足**》 別のフォルダを選択する場合は、「Browse」をクリックしてください。



インストールが開始されます。



ワークスペースのインストールが完了しました。「OK」をクリックします。

以上でワークスペースのインストールは完了です。

4. プログラム解説「mini_mcr.c」

ミニマイコンカー製作キット Ver.2 の倒立制御を行います。

4.1 プログラムリスト

```
  1 :  //---------------------------------------------------------------------------
  2 :  // 対象マイコン R8C/35A
  3 :  // ﾌｧｲﾙ内容     倒立制御プログラム
  4 :  // バージョン   Ver.1.00
  5 :  // Date         2010.12.08
  6 :  // Copyright    ルネサスマイコンカーラリー事務局
  7 :  //              日立インターメディックス株式会社
  8 :  //---------------------------------------------------------------------------
  9 :  //---------------------------------------------------------------------------
 10 :  // インクルード
 11 :  //---------------------------------------------------------------------------
 12 :  #include "sfr_r835a.h"
 13 :  #include "printf_lib.h"
 14 :  #include "stdio.h"
 15 :
 16 :  //---------------------------------------------------------------------------
 17 :  // シンボル定義
 18 :  //---------------------------------------------------------------------------
 19 :  #define TIMER_CYCLE     155             // 1ms:0.001/(1/(20000000/128))-1
 20 :  //#define PWM_CYCLE       39999           // 16ms:0.016/(1/(20000000/8))-1
 21 :  #define PWM_CYCLE       2499           // 1ms:0.001/(1/(20000000/8))-1
 22 :
 23 :  #define Def_500Hz       4999            // 500Hz:(1/500)/(1/(20000000/8))-1
 24 :  #define Def_1000Hz      2499            // 1000Hz:(1/1000)/(1/(20000000/8))-1
 25 :
 26 :  #define Def_C3          19083           // ド:(1/131)/(1/(20000000/8))-1
 27 :  #define Def_D3          17006           // レ:(1/147)/(1/(20000000/8))-1
 28 :  #define Def_E3          15151           // ミ:(1/165)/(1/(20000000/8))-1
 29 :  #define Def_F3          14285           // ファ:(1/175)/(1/(20000000/8))-1
 30 :  #define Def_G3          12754           // ソ:(1/196)/(1/(20000000/8))-1
 31 :  #define Def_A3          11362           // ラ:(1/220)/(1/(20000000/8))-1
 32 :  #define Def_B3          10120           // シ:(1/247)/(1/(20000000/8))-1
 33 :  #define Def_C4          9541            // ド:(1/262)/(1/(20000000/8))-1
 34 :
 35 :  #define DI()            asm("FCLR I")   // 割り込み禁止
 36 :  #define EI()            asm("FSET I")   // 割り込み許可
 37 :
 38 :  //---------------------------------------------------------------------------
 39 :  // 関数プロトタイプの宣言
 40 :  //---------------------------------------------------------------------------
 41 :  void init( void );
 42 :  unsigned char sensor( void );
 43 :  void motor( int data1, int data2 );
 44 :  void timer( unsigned long timer_set );
 45 :  void beep( int data1 );
 46 :  unsigned char dipsw( void );
 47 :  unsigned char pushsw( void );
 48 :
 49 :  //---------------------------------------------------------------------------
 50 :  // グローバル変数の宣言
 51 :  //---------------------------------------------------------------------------
 52 :  unsigned long   cnt0 = 0;               // timer 関数用
 53 :  unsigned long   cnt1 = 0;               // main 内で使用
 54 :  int             pattern = 0;            // パターン番号
 55 :
 56 :  double          AccSum = 0;
 57 :  double          AccOfs = 0;
 58 :  double          AngSum = 0;
```

```
 59 :  double          AngOfs = 0;
 60 :  double          AngSpd = 0;
 61 :  double          AccAcc = 0;
 62 :  double          Ang = 0;
 63 :  double          Pwm = 0;
 64 :  double          PwmSum = 0;
 65 :
 66 :  double          KAng = 0.04000;
 67 :  double          KAngSpd = 0.02000;
 68 :  double          KPwmSum = 0.00400;
 69 :  double          KPwm = 0.00800;
 70 :
 71 :  int             State = 0;
 72 :  int             AddPwmL = 0;
 73 :  int             AddPwmR = 0;
 74 :
 75 :  //---------------------------------------------------------------------------
 76 :  // メインプログラム
 77 :  //---------------------------------------------------------------------------
 78 :  void main(void)
 79 :  {
 80 :          int i;
 81 :          char c;
 82 :
 83 :
 84 :          // 初期化
 85 :          init();
 86 :          init_uart0_printf( SPEED_9600 );
 87 :
 88 :          // 起動音
 89 :          beep(Def_500Hz);
 90 :          timer(100);
 91 :          beep(Def_1000Hz);
 92 :          timer(100);
 93 :          beep(0);
 94 :
 95 :
 96 :          while( pushsw() == 0 ){
 97 :          }
 98 :
 99 :          beep(Def_1000Hz);
100 :          timer(100);
101 :          beep(0);
102 :
103 :          State = 1;
104 :          while( State == 1 );
105 :
106 :          beep(Def_500Hz);
107 :          timer(100);
108 :          beep(0);
109 :
110 :
111 :          while(1){
112 :                  printf( "%6d ",(int)ad6 );
113 :                  printf( "%6d ",(int)ad3 );
114 :                  printf( "%6d ",(int)( AccAcc * 100 ) );
115 :                  printf( "%6d ",(int)( AngSpd * 100 ) );
116 :                  printf( "%6d ",(int)( Ang * 10 ) );
117 :                  printf( "%6d ",(int)( KAng    * 100000 ) );
118 :                  printf( "%6d ",(int)( KAngSpd * 100000 ) );
119 :                  printf( "%6d ",(int)( KPwmSum * 100000 ) );
120 :                  printf( "%6d ",(int)( KPwm    * 100000 ) );
121 :                  printf( "¥n" );
122 :
123 :
124 :                  switch( dipsw() ){
125 :                  case 0:
126 :                          AddPwmL = 0;
127 :                          AddPwmR = 0;
128 :                          break;
129 :                  case 1:
130 :                          AddPwmL = 10;
```

```
131 :                           AddPwmR = 10;
132 :                           break;
133 :                   case 2:
134 :                           AddPwmL = 0;
135 :                           AddPwmR = -10;
136 :                           break;
137 :                   case 3:
138 :                           AddPwmL = -10;
139 :                           AddPwmR = 0;
140 :                           break;
141 :                   default:
142 :                           break;
143 :                   }
144 :
145 :
146 :                   i = get_uart0( &c );
147 :                   if( i == 1 ) {
148 :                           switch( c ) {
149 :                           case '1':
150 :                                   KAng = KAng + 0.0001;
151 :                                   break;
152 :                           case 'q':
153 :                                   KAng = KAng + 0.00001;
154 :                                   break;
155 :                           case 'a':
156 :                                   KAng = KAng - 0.00001;
157 :                                   break;
158 :                           case 'z':
159 :                                   KAng = KAng - 0.0001;
160 :                                   break;
161 :
162 :                           case '2':
163 :                                   KAngSpd = KAngSpd + 0.0001;
164 :                                   break;
165 :                           case 'w':
166 :                                   KAngSpd = KAngSpd + 0.00001;
167 :                                   break;
168 :                           case 's':
169 :                                   KAngSpd = KAngSpd - 0.00001;
170 :                                   break;
171 :                           case 'x':
172 :                                   KAngSpd = KAngSpd - 0.0001;
173 :                                   break;
174 :
175 :                           case '3':
176 :                                   KPwmSum = KPwmSum + 0.0001;
177 :                                   break;
178 :                           case 'e':
179 :                                   KPwmSum = KPwmSum + 0.00001;
180 :                                   break;
181 :                           case 'd':
182 :                                   KPwmSum = KPwmSum - 0.00001;
183 :                                   break;
184 :                           case 'c':
185 :                                   KPwmSum = KPwmSum - 0.0001;
186 :                                   break;
187 :
188 :                           case '4':
189 :                                   KPwm = KPwm + 0.0001;
190 :                                   break;
191 :                           case 'r':
192 :                                   KPwm = KPwm + 0.00001;
193 :                                   break;
194 :                           case 'f':
195 :                                   KPwm = KPwm - 0.00001;
196 :                                   break;
197 :                           case 'v':
198 :                                   KPwm = KPwm - 0.0001;
199 :                                   break;
200 :
201 :                           default:
202 :                                   break;
```

```
203 :                           }
204 :                   }
205 :
206 :
207 :                   if( -1 < Ang && Ang < 1 ){
208 :                           p1 = ( p1 & 0xf0 ) | ( ~0x06 & 0x0f );
209 :                   }else if( -3 < Ang && Ang < -1 ){
210 :                           p1 = ( p1 & 0xf0 ) | ( ~0x0c & 0x0f );
211 :                   }else if( Ang < -3 ){
212 :                           p1 = ( p1 & 0xf0 ) | ( ~0x08 & 0x0f );
213 :                   }else if( 1 < Ang && Ang < 3 ){
214 :                           p1 = ( p1 & 0xf0 ) | ( ~0x03 & 0x0f );
215 :                   }else if( 3 < Ang ){
216 :                           p1 = ( p1 & 0xf0 ) | ( ~0x01 & 0x0f );
217 :                   }
218 :
219 :
220 :                   timer(100);
221 :
222 :           }
223 :
224 :   }
225 :
226 :   //--------------------------------------------------------------------------------
227 :   // R8C/35A の内蔵周辺機能の初期化
228 :   //--------------------------------------------------------------------------------
229 :   void init( void )
230 :   {
231 :           unsigned char i = 0;
232 :
233 :           // 割り込み禁止
234 :           DI();
235 :
236 :           // クロック発生回路の XIN クロック設定
237 :           prc0 = 1;
238 :
239 :           cm13 = 1;
240 :           cm05 = 0;
241 :           while(i <= 50) i++;
242 :           ocd2 = 0;
243 :
244 :           prc0 = 0;
245 :
246 :           // I/O ポートの入出力設定
247 :           prc2 = 1;                       // pd0 レジスタへの書き込み許可
248 :   //        pd0 = 0xe0;                     // P0_0～P0_3:センサー
249 :   //                                        // P0_4:マイクロスイッチ
250 :   //                                        // P0_5～P0_7:LED
251 :           pd0 = 0x00;                     // P0_0～P0_7:センサー
252 :           prc2 = 0;                       // pd0 レジスタへの書き込み禁止
253 :           pd1 = 0xdf;                     // P1_0～P1_3:LED
254 :                                           // P1_4:TXD0
255 :                                           // P1_5:RXD0
256 :           pd2 = 0xfe;                     // P2_0:スイッチ
257 :                                           // P2_1:AIN1
258 :                                           // P2_2:PWMA
259 :                                           // P2_3:BIN1
260 :                                           // P2_4:PWMB
261 :                                           // P2_5:SERVO
262 :                                           // P2_6:AIN2
263 :                                           // P2_7:BIN2
264 :           pd3 = 0xfb;                     // P3_2:赤外線受信
265 :                                           // P3_4:ブザー
266 :           pd4 = 0x80;                     // P4_2:VREF
267 :                                           // P4_3～P4_5:DIPSW
268 :                                           // P4_6:XIN
269 :                                           // P4_7:XOUT
270 :           pd5 = 0x40;                     // P5_7:DIPSW
271 :           pd6 = 0xff;                     //
272 :
273 :
274 :
```

```
275 :         mstcr = 0x00;                   // モジュールストップ解除
276 :
277 :
278 :
279 :         // タイマ RB の 1ms 割り込み設定
280 :         trbmr = 0x00;                   // カウントソースは f1
281 :         trbpre = 128 - 1;               // プリスケーラ
282 :         trbpr = TIMER_CYCLE;            // プライマリカウンタ
283 :         trbic = 0x01;                   // タイマ RB の割り込みレベル設定
284 :         trbcr = 0x01;                   // カウントを開始
285 :
286 :         // タイマ RC の PWM モード
287 :         trccr1 = 0xb0;                  // カウントソースは f8
288 :         trcgra = 0;                     // 圧電サウンダの周期
289 :         trcgrc = 0;                     // 圧電サウンダのデューティ比
290 :         trccr2 = 0x02;                  // TRCIOC 端子はアクティブレベル H
291 :         trcoer = 0x0b;                  // TRCIOC 端子の出力許可
292 :         trcpsr1 = 0x02;                 // TRCIOC 端子を P3_4 に割り当て
293 :         trcmr = 0x8a;                   // カウントを開始
294 :
295 :         // タイマ RD のリセット同期 PWM モード
296 :         trdpsr0 = 0x08;                 // TRDIOB0 端子を P2_2 に割り当て
297 :         trdpsr1 = 0x05;                 // TRDIOB1 端子を P2_5 に割り当て
298 :                                         // TRDIOA1 端子を P2_4 に割り当て
299 :         trdmr = 0xf0;                   // レジスタをバッファ動作にする
300 :         trdfcr = 0x01;                  // リセット同期 PWM モードに設定
301 :         trdoer1 = 0xcd;                 // TRDIOB1 の出力許可
302 :                                         // TRDIOA1 の出力許可
303 :                                         // TRDIOB0 端子の出力許可
304 :         trdcr0 = 0x23;                  // カウントソースは f8
305 :         trdgra0 = trdgrc0 = PWM_CYCLE;  // 周期
306 :         trdgrb0 = trdgrd0 = 0;          // TRDIOB0 端子（左モータ）
307 :         trdgra1 = trdgrc1 = 0;          // TRDIOA1 端子（右モータ）
308 :         trdgrb1 = trdgrd1 = 0;          // TRDIOB1 端子（サーボ）
309 :         trdstr = 0x0d;                  // カウントを開始
310 :
311 :         // AD 変換機の設定
312 :         admod = 0x33;                   // 繰り返し掃引モード
313 :         adinsel = 0x30;                 // 8 端子を使用
314 :         adcon1 = 0x30;                  // 10 ビットモード、AD 動作可能
315 :         adst = 1;                       // A/D 変換スタート
316 :
317 :         // 割り込み許可
318 :         EI();
319 : }
320 :
321 : //-----------------------------------------------------------------------
322 : // 割り込み
323 : //-----------------------------------------------------------------------
324 : #pragma interrupt intTRBIC (vect=24)
325 : void intTRBIC( void )
326 : {
327 :         static int cnt = 0;
328 :
329 :         cnt0++;
330 :         cnt1++;
331 :
332 :         switch( State ){
333 :
334 :         case 1:
335 :                 AngSum += ( double )ad6;
336 :                 AccSum += ( double )ad3;
337 :
338 :                 cnt++;
339 :                 if( 100 <= cnt ){
340 :                         cnt = 0;
341 :                         State = 2;
342 :
343 :                         AngOfs = AngSum / 100;
344 :                         AccOfs = AccSum / 100;
345 :                 }
346 :
```

```
347 :                   break;
348 :
349 :           case 2:
350 :                   AngSpd = ( ( double )ad6 - AngOfs ) * 5 / ( 1024 * 0.0067 );
351 :                   AccAcc = -( ( double )ad3 - AccOfs ) * 5 / ( 1024 * 1.000 );
352 :
353 :                   // 切り捨て
354 :                   if( -0.5 < AngSpd && AngSpd < 0.5 ){
355 :                           AngSpd = 0;
356 :                   }
357 :                   if( -0.01 < AccAcc && AccAcc < 0.01 ){
358 :                           AccAcc = 0;
359 :                   }
360 :
361 :                   Ang += AngSpd * 0.002 - Ang * 1.25 * 0.002 + AccAcc * 90.0 * 1.25 * 0.002;
362 :
363 :                   State = 3;
364 :                   break;
365 :
366 :           case 3:
367 :                   Pwm += KAng * Ang + KAngSpd * AngSpd + KPwmSum * PwmSum + KPwm * Pwm;
368 :
369 :                   if( Ang < -45 || 45 < Ang ){
370 :                           Pwm = 0;
371 :                   }
372 :
373 :                   if( 100 < Pwm ){
374 :                           Pwm = 100;
375 :                   }
376 :                   if( Pwm < -100 ){
377 :                           Pwm = -100;
378 :                   }
379 :
380 :                   PwmSum += Pwm * 0.002;
381 :
382 :                   motor( -Pwm + AddPwmL, -Pwm + AddPwmR );
383 :
384 :                   State = 2;
385 :                   break;
386 :
387 :           default:
388 :                   break;
389 :           }
390 :
391 : }
392 :
393 : //--------------------------------------------------------------------------------
394 : // センサー状態検出
395 : // 引数         なし
396 : // 戻り値       センサ値
397 : //--------------------------------------------------------------------------------
398 : unsigned char sensor( void )
399 : {
400 :           volatile unsigned char  data1;
401 :
402 :           data1 = ~p0;                    // ラインの色は白
403 :           data1 = data1 & 0x0f;
404 :
405 :           return( data1 );
406 : }
407 :
408 : //--------------------------------------------------------------------------------
409 : // モーター速度制御
410 : // 引数         左モータ:-100～100、右モータ:-100～100
411 : //              0 で停止、100 で正転 100%、-100 で逆転 100%
412 : // 戻り値       なし
413 : //--------------------------------------------------------------------------------
414 : void motor( int data1, int data2 )
415 : {
416 :           volatile int    motor_r;
417 :           volatile int    motor_l;
418 :           volatile int    sw_data;
```

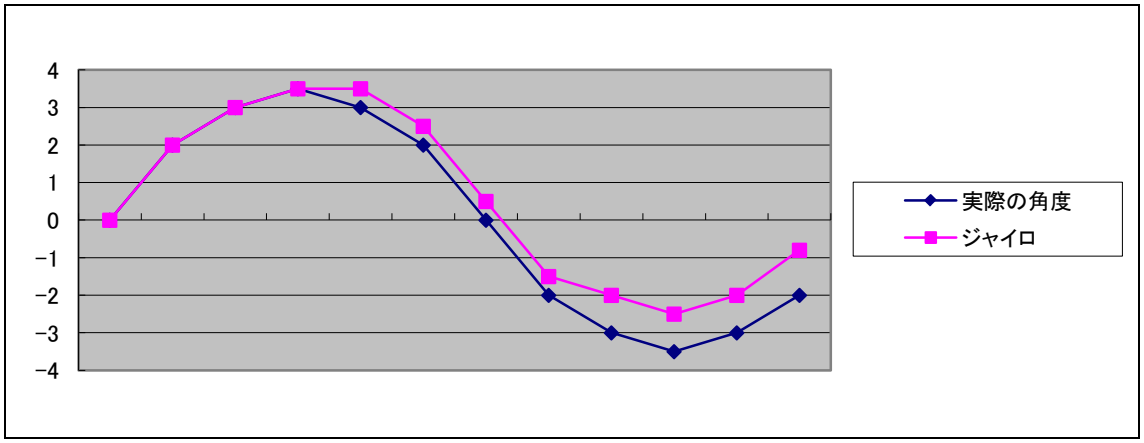
```
419 :
420 :  //        sw_data = dipsw() + 5;
421 :  //        motor_l = (long)data1 * sw_data / 20;
422 :  //        motor_r = (long)data2 * sw_data / 20;
423 :          motor_l = (long)data1;
424 :          motor_r = (long)data2;
425 :
426 :          if( motor_l >= 0 ) {
427 :                  p2_1 = 0;
428 :                  p2_6 = 1;
429 :                  trdgrd0 = (long)( PWM_CYCLE - 1 ) * motor_l / 100;
430 :          } else {
431 :                  p2_1 = 1;
432 :                  p2_6 = 0;
433 :                  trdgrd0 = (long)( PWM_CYCLE - 1 ) * ( -motor_l ) / 100;
434 :          }
435 :
436 :          if( motor_r >= 0 ) {
437 :                  p2_3 = 0;
438 :                  p2_7 = 1;
439 :                  trdgrc1 = (long)( PWM_CYCLE - 1 ) * motor_r / 100;
440 :          } else {
441 :                  p2_3 = 1;
442 :                  p2_7 = 0;
443 :                  trdgrc1 = (long)( PWM_CYCLE - 1 ) * ( -motor_r ) / 100;
444 :          }
445 :  }
446 :
447 :  //---------------------------------------------------------------------------
448 :  // 時間稼ぎ
449 :  // 引数         タイマ値 1=1ms
450 :  // 戻り値       なし
451 :  //---------------------------------------------------------------------------
452 :  void timer( unsigned long data1 )
453 :  {
454 :          cnt0 = 0;
455 :          while( cnt0 < data1 );
456 :  }
457 :
458 :  //---------------------------------------------------------------------------
459 :  // 音を鳴らす
460 :  // 引数         (1/音の周波数)/(1/(クロック周波数/8))-1
461 :  // 戻り値       なし
462 :  //---------------------------------------------------------------------------
463 :  void beep( int data1 )
464 :  {
465 :          trcgra = data1;                 // 周期の設定
466 :          trcgrc = data1 / 2;             // デューティ 50%のため周期の半分の値
467 :  }
468 :
469 :  //---------------------------------------------------------------------------
470 :  // DIP スイッチ状態検出
471 :  // 引数         なし
472 :  // 戻り値       0～15、DIP スイッチが ON の場合、対応するビットが 0 になります。
473 :  //---------------------------------------------------------------------------
474 :  unsigned char dipsw( void )
475 :  {
476 :          volatile unsigned char  data1;
477 :
478 :          data1 = ( ( p5 >> 4 ) & 0x08 ) | ( ( p4 >> 3 ) & 0x07 );
479 :
480 :          return( data1 );
481 :  }
482 :
483 :  //---------------------------------------------------------------------------
484 :  // プッシュスイッチ状態検出
485 :  // 引数         なし
486 :  // 戻り値       スイッチが押されていない場合:0、押された場合:1
487 :  //---------------------------------------------------------------------------
488 :  unsigned char pushsw( void )
489 :  {
490 :          unsigned char data1;
```

```
491 :
492 :           data1 = ~p2;
493 :           data1 &= 0x01;
494 :
495 :           return( data1 );
496 :  }
```

4.2　角度の計算について

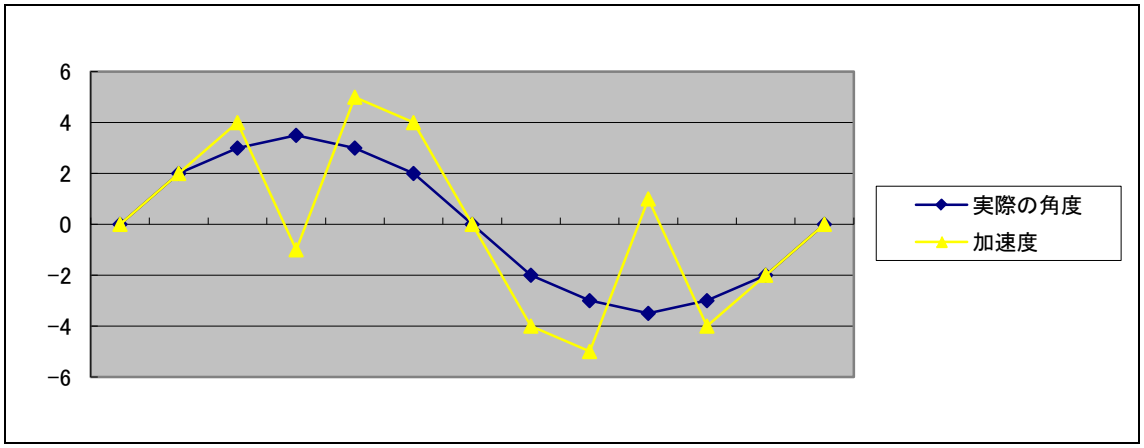
角度は、ジャイロセンサーからの角速度を積分した角度と、加速度センサーからの角度を合成して求めています。

4.2.1　ジャイロセンサーから角度を計算



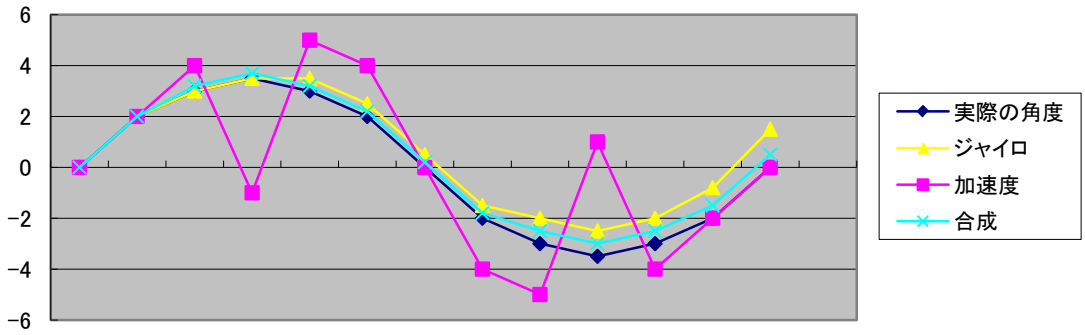
ジャイロセンサーからの角速度を積分した角度のみを使用した場合、積分誤差が蓄積してしまい、角度がオフセットしていきます。

4.2.2　加速度センサーから角度を計算



加速度センサーからの角度のみを使用した場合、静止時の角度は求まりますが、ロボットが移動中には加速度により正しい角度が求まりません。

4.2.3　ジャイロ＋加速度センサーから角度を計算

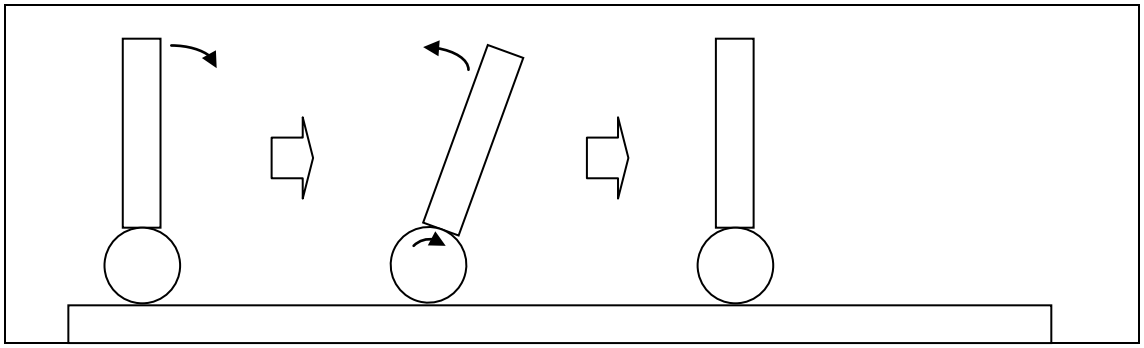


そこで、ジャイロセンサーからの角速度を積分した角度をハイパスフィルタ、加速度センサーからの角度をローパスフィルタに通して合成することで、角度がオフセットすることなく、ロボットが移動中も正しい角度が求まります。

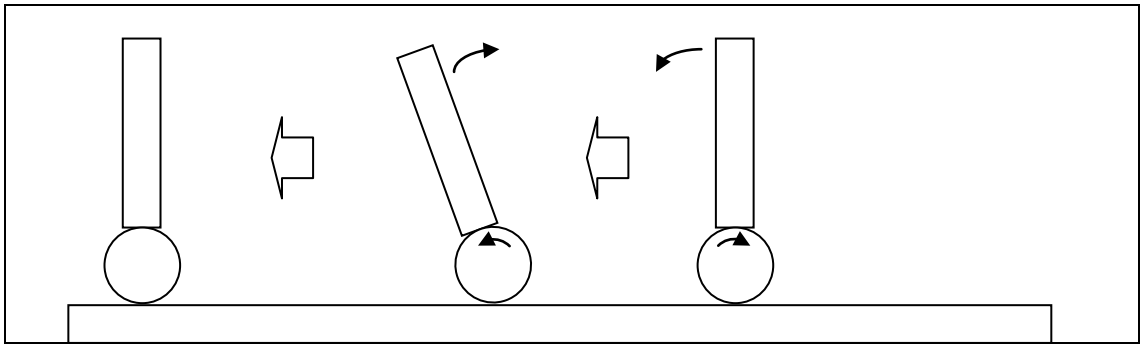
ジャイロセンサーの角速度：$\dot{\theta}_g$　加速度センサーの角度：θ_a
前回の角度：$\theta_{(n-1)}$　角度：θ　角周波数：ω　カットオフ周波数：f_0
サンプリング時間：Δt

$\omega = 2\pi f_0$
$\theta = \theta_{(n-1)} + \dot{\theta}_g \Delta t - \omega\theta_{(n-1)}\Delta t + \omega\theta_a \Delta t$

4.3　倒立制御について



ロボットが倒れる方向に進むことによって、倒立をすることができます。
ですが、この制御だけでは最初の位置には止まらず、あちこちに移動してしまいます。



元の位置に止まるには、さらに倒れる方向に進め、逆方向に倒れるようにすると元の位置に戻ります。

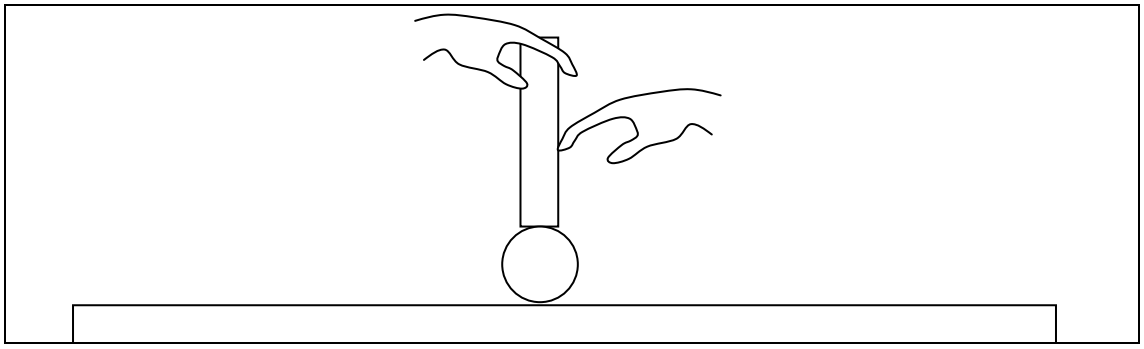
角度：θ　角速度：ω　距離：d　前回の速度：$v_{(n-1)}$　速度：v　加速度：a
サンプリング時間：Δt
角度ゲイン：$K\theta$　角速度ゲイン：$K\omega$　距離ゲイン：Kd　速度ゲイン：Kv

$$a = K\theta \times \theta + K\omega \times \omega + Kd \times d + Kv \times v_{(n-1)}$$
$$v = v_{(n-1)} + a$$

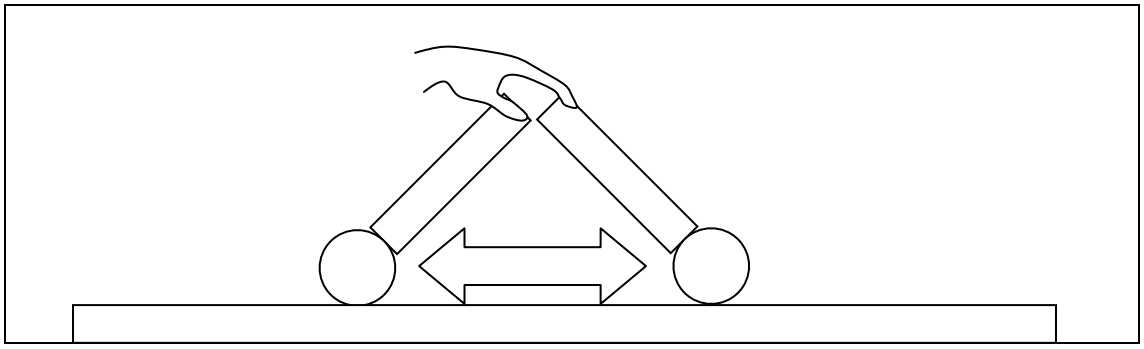
ミニマイコンカー製作キット Ver.2 にはロータリーエンコーダーが付いていませんので、距離と速度についてはフィードバック制御ができません。距離は PWM のデューティ値の積算値を、速度は PWM のデューティ値を、おおよその値としてロータリーエンコーダーの代わりにしています。

4.4　ゲイン調整について

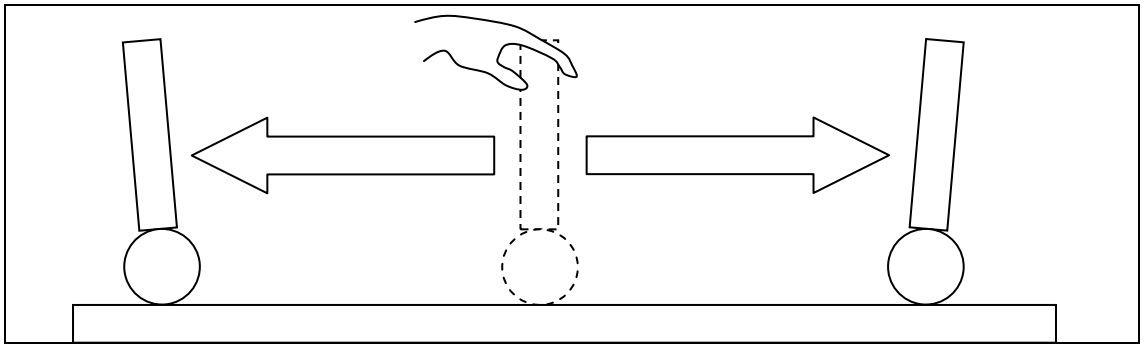
角度ゲイン、**角速度ゲイン**、**距離ゲイン**、**速度ゲイン**を調整して、ミニマイコンカー製作キット Ver.2 が倒立するようにします。



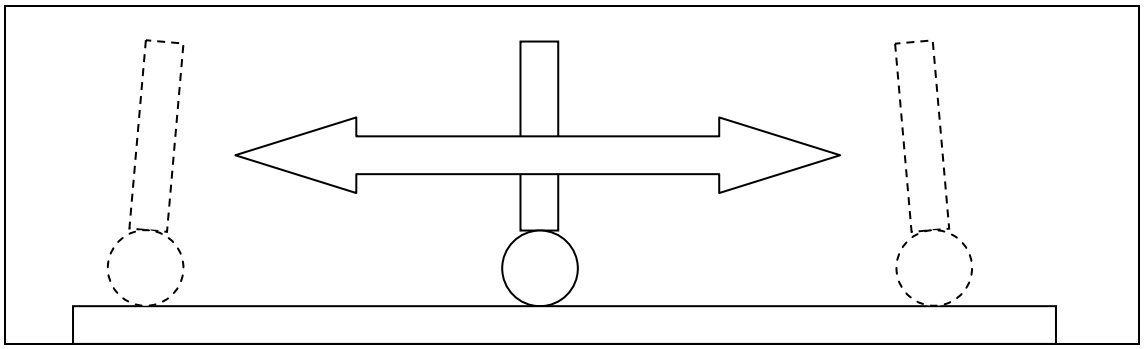
ミニマイコンカー製作キット Ver.2 の上部を持ったままバランスさせた状態でスタートスイッチを押します。



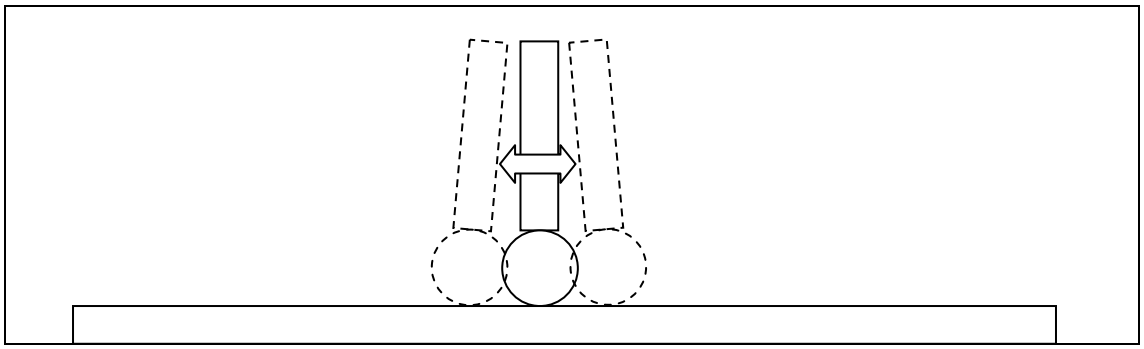
角度ゲインを上げていくと、素早い前後移動をするようになります。
この状態では、まだバランスを取っていません。



角速度ゲインを上げていくと、素早い前後移動は収まり、バランスを取るようになります。
この状態では、傾いた方向に移動し続けます。



距離ゲインを上げていくと、移動し続けるのをやめ、大きく前後移動しながらバランスするようになります。



速度ゲインを上げていくと、大きく前後移動するのをやめ、その場でバランスを取るようになります。

4.5　シンボル定義

走行プログラムでは、PWM の周期は 16ms でしたが、倒立制御の周期が 2ms なので、それより速い 1ms に変更しています。

プログラム

```
  20 :  //#define PWM_CYCLE       39999           // 16ms:0.016/(1/(20000000/8))-1
  21 :  #define PWM_CYCLE       2499          // 1ms:0.001/(1/(20000000/8))-1
```

4.6　グローバル変数の宣言

倒立制御に必要な変数を追加しています。

プログラム

```
  56 :  double          AccSum = 0;
  57 :  double          AccOfs = 0;
  58 :  double          AngSum = 0;
  59 :  double          AngOfs = 0;
  60 :  double          AngSpd = 0;
  61 :  double          AccAcc = 0;
  62 :  double          Ang = 0;
  63 :  double          Pwm = 0;
  64 :  double          PwmSum = 0;
  65 :
  66 :  double          KAng = 0.04000;
  67 :  double          KAngSpd = 0.02000;
  68 :  double          KPwmSum = 0.00400;
  69 :  double          KPwm = 0.00800;
  70 :
  71 :  int             State = 0;
  72 :  int             AddPwmL = 0;
  73 :  int             AddPwmR = 0;
```

名称	型	説明
AccSum	double	加速度センサー出力の 100 回分の合計
AccOfs	double	加速度センサー出力の 100 回分の平均
AngSum	double	ジャイロセンサー出力の 100 回分の合計
AngOfs	double	ジャイロセンサー出力の 100 回分の平均
AngSpd	double	角速度
AccAcc	double	加速度
Ang	double	角度
Pwm	double	モーターに加える PWM のデューティ（速度）
PwmSum	double	モーターに加える PWM の積分値（距離）
KAng	double	角度定数
KAngSpd	double	角速度定数
KPwmSum	double	距離定数
KPwm	double	速度定数
State	int	割り込み内での処理のステート
AddPwmL	int	左モーターに加える PWM のデューティのオフセット値
AddPwmR	int	右モーターに加える PWM のデューティのオフセット値

4.7 メインプログラムを説明する前に

main 関数は、main 関数の後に記載されている関数を組み合わせてプログラムしていますので、先に main 関数以外の関数の解説を行います。

本マニュアルで解説されている以外の関数については「ミニマイコンカー製作キット Ver.2　C 言語走行プログラム解説マニュアル」の「5. プログラム解説「mini_mcr.c」」を参照してください。

4.8 R8C/35A の内蔵周辺機能の初期化：init 関数

init 関数は「ミニマイコンカー製作キット Ver.2　C 言語走行プログラム解説マニュアル」で解説されています。ここでは、init 関数の変更部分のみを解説します。

4.8.1 ポートの設定

ミニマイコンカー製作キット Ver.2 のセンサー部分をカットしたことにより、センサー部分の LED は使用できなくなります。空きポートもアナログ入力端子として割り当て、センサーを拡張できるようにします。

プログラム

```
248 :  //          pd0 = 0xe0;                    // P0_0～P0_3:センサー
249 :  //                                         // P0_4:マイクロスイッチ
250 :  //                                         // P0_5～P0_7:LED
251 :            pd0 = 0x00;                    // P0_0～P0_7:センサー
```

レジスタ	ビット	シンボル	説明	設定値
PD0	7	PD0_7	-	0x00
	6	PD0_6	-	
	5	PD0_5	-	
	4	PD0_4	加速度センサーZ 軸	
	3	PD0_3	加速度センサーY 軸	
	2	PD0_2	加速度センサーX 軸	
	1	PD0_1	ジャイロセンサーPITCH 軸	
	0	PD0_0	ジャイロセンサーROLL 軸	

4.8.2 A/D コンバータの設定

ポート 0 のアナログ端子 8ch を、繰り返し掃引モードで A/D 変換できるように設定します。

プログラム

```
312 :       admod = 0x33;              // 繰り返し掃引モード
313 :       adinsel = 0x30;            // 8 端子を使用
314 :       adcon1 = 0x30;             // 10 ビットモード、AD 動作可能
315 :       adst = 1;                  // A/D 変換スタート
```

レジスタ	ビット	シンボル	説明	設定値
ADMOD	7	ADCAP1	A/D 変換トリガをソフトウェアトリガにするため“00”にします。	0x33
	6	ADCAP0		
	5	MD2	A/D 動作モードを繰り返し掃引モードにするため“110”にします。	
	4	MD1		
	3	MD0		
	2	CKS2	クロック源を f1 にするため“0”にします。	
	1	CKS1	分周比を分周なしにするため“11”にします。	
	0	CKS0		
ADINSEL	7	ADGSEL1	A/D 入力グループをポート P0 グループにするため“00”にします。	0x30
	6	ADGSEL0		
	5	SCAN1	A/D 掃引端子数を 8 端子にするため“11”にします。	
	4	SCAN0		
	3	-	何も配置されていないので“0”にします。	
	2	CH2	アナログ入力端子を AN0～AN7 にするため“000”にします。	
	1	CH1		
	0	CH0		
ADCON1	7	ADDDAEL	A/D 断線検出アシストを変換前ディスチャージにするため“0”にします。	0x30
	6	ADDDAEN	A/D 断線検出アシストを禁止にするため“0”にします。	
	5	ADSTBY	A/D を動作可能にするため“1”にします。	
	4	BITS	分解能を 10 ビットにするため“1”にします。	
	3	-	何も配置されていないので“0”にします。	
	2	-		
	1	-		
	0	ADEX0	拡張アナログ入力端子を非選択にするため“0”にします。	

4.9 割り込みプログラム：intTRBIC 関数

intTRBIC 関数は、1ms ごとに割り込みで実行されます。

プログラム

```
 324 :  #pragma interrupt intTRBIC (vect=24)
 325 :  void intTRBIC( void )
 326 :  {
 327 :          static int cnt = 0;
 328 :
 329 :          cnt0++;
 330 :          cnt1++;
 331 :
 332 :          switch( State ){
 334～ 347 :  「4.9.1 オフセット電圧の計算」を参照
 349～ 364 :  「4.9.2 ジャイロセンサーの角速度計算＆加速度センサーの加速度計算＆角度計算」を参照
 366～ 385 :  「4.9.3 モーターPWM デューティの決定」を参照
 387 :          default:
 388 :                  break;
 389 :          }
 390 :
 391 :  }
```

4.9.1　オフセット電圧の計算

State 変数が 1 の場合、オフセット電圧の計算が行われます。
100 回分の A/D 変換結果から平均を計算し、この値をゼロ点とします。

プログラム

```
334 :          case 1:
335 :                  AngSum += ( double )ad6;
336 :                  AccSum += ( double )ad3;
337 :
338 :                  cnt++;
339 :                  if( 100 <= cnt ){
340 :                          cnt = 0;
341 :                          State = 2;
342 :
343 :                          AngOfs = AngSum / 100;
344 :                          AccOfs = AccSum / 100;
345 :                  }
346 :
347 :                  break;
```

```
335 :                  AngSum += ( double )ad6;
336 :                  AccSum += ( double )ad3;
```

ジャイロセンサーと加速度センサーの A/D 変換値を AngSum 変数と AccSum 変数に、それぞれ積算していきます。

```
338 :                  cnt++;
339 :                  if( 100 <= cnt ){
340 :                          cnt = 0;
341 :                          State = 2;
342 :
343 :                          AngOfs = AngSum / 100;
344 :                          AccOfs = AccSum / 100;
345 :                  }
346 :
347 :                  break;
```

cnt 変数をインクリメントしていき、100 回以上になった場合、AngSum 変数と AccSum 変数を、それぞれ 100 で割り平均値を計算し、AngOfs 変数と AccOfs 変数に代入します。
State 変数は 2 にして、「4.9.2 ジャイロセンサーの角速度計算＆加速度センサーの加速度計算＆角度計算」を行うようにします。

4.9.2　ジャイロセンサーの角速度計算＆加速度センサーの加速度計算＆角度計算

State 変数が 2 の場合、A/D 変換の値を、角速度、加速度、角度に変換します。

プログラム

```
349 :          case 2:
350 :                  AngSpd = ( ( double )ad6 - AngOfs ) * 5 / ( 1024 * 0.0067 );
351 :                  AccAcc = -( ( double )ad3 - AccOfs ) * 5 / ( 1024 * 1.000 );
352 :
353 :                  // 切り捨て
354 :                  if( -0.5 < AngSpd && AngSpd < 0.5 ){
355 :                          AngSpd = 0;
356 :                  }
357 :                  if( -0.01 < AccAcc && AccAcc < 0.01 ){
358 :                          AccAcc = 0;
359 :                  }
360 :
361 :                  Ang += AngSpd * 0.002 - Ang * 1.25 * 0.002 + AccAcc * 90.0 * 1.25 * 0.002;
362 :
363 :                  State = 3;
364 :                  break;
```

```
350 :                  AngSpd = ( ( double )ad6 - AngOfs ) * 5 / ( 1024 * 0.0067 );
```

ジャイロセンサーの A/D 変換値から AngOfs 変数を引くことによって、ゼロ点からの変化量を計算します。

ジャイロセンサーの感度が、6.7mV/deg/sec（オペアンプで約 10 倍にしています）、電源電圧が 5V、A/D 変換の分解能が 10 ビットなので、上記の式により角速度[deg/sec]が計算でき、AngSpd 変数に代入します。

```
351 :                  AccAcc = -( ( double )ad3 - AccOfs ) * 5 / ( 1024 * 1.000 );
```

加速度センサーの A/D 変換値から AccOfs 変数を引くことによって、ゼロ点からの変化量を計算します。

加速度センサーの感度が、1000mV/G（電源電圧 5V）電源電圧が 5V、A/D 変換の分解能が 10 ビットなので、上記の式により加速度[G]が計算でき、AccAcc 変数に代入します。

```
354 :                  if( -0.5 < AngSpd && AngSpd < 0.5 ){
355 :                          AngSpd = 0;
356 :                  }
357 :                  if( -0.01 < AccAcc && AccAcc < 0.01 ){
358 :                          AccAcc = 0;
359 :                  }
```

角速度が±0.5deg/sec 未満の場合は、切り捨てを行います。

加速度が±0.01G 未満の場合は、切り捨てを行います

```
361 :                   Ang += AngSpd * 0.002 - Ang * 1.25 * 0.002 + AccAcc * 90.0 * 1.25 * 0.002;
```

ジャイロセンサーと加速度センサーの出力から角度を計算します。

0.002 はサンプリング時間で、この式が呼び出される間隔を入力しています。

1.25 は角周波数で、カットオフ周波数を 0.2Hz に設定しているため、この値になります。

90.0 は加速度から角度に変換するための定数です。加速度センサーは 90° 傾けた時に 1G となるので、加速度×90.0 で角度が求まります（加速度センサーの角度は X 軸と Z 軸の加速度の合成で求める必要がありますが計算量が多くなるため X 軸のみで計算しています)。

```
363 :                   State = 3;
```

State 変数は 3 にして、「4.9.3 モーターPWM デューティの決定」を行うようにします。

4.9.3　モーターPWM デューティの決定

State 変数が 3 の場合、モーターを動かします。

プログラム

```
366 :          case 3:
367 :                  Pwm += KAng * Ang + KAngSpd * AngSpd + KPwmSum * PwmSum + KPwm * Pwm;
368 :
369 :                  if( Ang < -45 || 45 < Ang ){
370 :                          Pwm = 0;
371 :                  }
372 :
373 :                  if( 100 < Pwm ){
374 :                          Pwm = 100;
375 :                  }
376 :                  if( Pwm < -100 ){
377 :                          Pwm = -100;
378 :                  }
379 :
380 :                  PwmSum += Pwm * 0.002;
381 :
382 :                  motor( -Pwm + AddPwmL, -Pwm + AddPwmR );
383 :
384 :                  State = 2;
385 :                  break;
```

```
367 :                  Pwm += KAng * Ang + KAngSpd * AngSpd + KPwmSum * PwmSum + KPwm * Pwm;
```

角度、角速度、距離、速度にそれぞれ係数をかけて、モーターのデューティを計算します。

```
369 :                  if( Ang < -45 || 45 < Ang ){
370 :                          Pwm = 0;
371 :                  }
372 :
373 :                  if( 100 < Pwm ){
374 :                          Pwm = 100;
375 :                  }
376 :                  if( Pwm < -100 ){
377 :                          Pwm = -100;
378 :                  }
```

角度が 45° を超える場合は、倒立状態を維持できていないと判断しますので、モーターを停止します。また、モーターのデューティが 100%を超える場合にも 100%に制限します。

```
380 :                  PwmSum += Pwm * 0.002;
381 :
382 :                  motor( -Pwm + AddPwmL, -Pwm + AddPwmR );
383 :
384 :                  State = 2;
385 :                  break;
```

PwmSum 変数に Pwm 変数の値を積算していきます。0.002 はサンプリング時間です。

motor 関数を呼び出して、モーターを動かします。AddPwmL 変数と AddPwmR 変数は、意図的にモーターのデューティをオフセットさせて、その場で倒立、前進、旋回をさせるためのものです。

State 変数は 2 にして、「4.9.2 ジャイロセンサーの角速度計算＆加速度センサーの加速度計算＆角度計算」に戻ります。

4.10 メインプログラム：main 関数

main 関数は、スタートアップルーチンから呼び出され、最初に実行される C 言語のプログラムです。

```
 78 :  void main(void)
 79 :  {
 80 :          int i;
 81 :          char c;
 82 :
 83 :
 84 :          // 初期化
 85 :          init();
 86～  87 :  「4.10.1 シリアルの初期化」
 88 :          // 起動音
 89 :          beep(Def_500Hz);
 90 :          timer(100);
 91 :          beep(Def_1000Hz);
 92 :          timer(100);
 93 :          beep(0);
 94 :
 95 :
 96 :          while( pushsw() == 0 ){
 97 :          }
 98 :
 99～ 110 :  「4.10.2 オフセット電圧の計算待ち」
111 :          while(1){
112～ 123 :  「4.10.3 シリアル出力」
124～ 145 :  「4.10.4 DIP スイッチの設定」
146～ 206 :  「4.10.5 シリアル入力」
207～ 219 :  「4.10.6 LED 出力」
222 :          }
223 :
224 :  }
```

4.10.1 シリアルの初期化

```
 86 :          init_uart0_printf( SPEED_9600 );
```

パラメーターの調整にシリアル通信をする必要がありますので、シリアルを初期化します。

4.10.2 オフセット電圧の計算待ち

```
 99 :          beep(Def_1000Hz);
100 :          timer(100);
101 :          beep(0);
102 :
103 :          State = 1;
104 :          while( State == 1 );
105 :
106 :          beep(Def_500Hz);
107 :          timer(100);
108 :          beep(0);
```

State 変数を 1 にして、割り込みルーチン内でオフセット電圧の計算を行います。

計算が終了すると 2 になるので、それまではループして待ちます。

4.10.3 シリアル出力

```
112 :                   printf( "%6d ",(int)ad6 );
113 :                   printf( "%6d ",(int)ad3 );
114 :                   printf( "%6d ",(int)( AccAcc * 100 ) );
115 :                   printf( "%6d ",(int)( AngSpd * 100 ) );
116 :                   printf( "%6d ",(int)( Ang * 10 ) );
117 :                   printf( "%6d ",(int)( KAng    * 100000 ) );
118 :                   printf( "%6d ",(int)( KAngSpd * 100000 ) );
119 :                   printf( "%6d ",(int)( KPwmSum * 100000 ) );
120 :                   printf( "%6d ",(int)( KPwm    * 100000 ) );
121 :                   printf( "¥n" );
```

内部の各変数をシリアル出力します。

本プログラムの printf 関数では、小数を扱うことができませんので、100 などの数値を掛けて、整数化します。

4.10.4 DIP スイッチの設定

```
124 :                   switch( dipsw() ){
125 :                   case 0:
126 :                           AddPwmL = 0;
127 :                           AddPwmR = 0;
128 :                           break;
129 :                   case 1:
130 :                           AddPwmL = 10;
131 :                           AddPwmR = 10;
132 :                           break;
133 :                   case 2:
134 :                           AddPwmL = 0;
135 :                           AddPwmR = -10;
136 :                           break;
137 :                   case 3:
138 :                           AddPwmL = -10;
139 :                           AddPwmR = 0;
140 :                           break;
141 :                   default:
142 :                           break;
143 :                   }
```

DIP スイッチの状態によって、その場で倒立、前進、旋回を切り替えます。

4.10.5 シリアル入力

```
146 :                   i = get_uart0( &c );
147 :                    if( i == 1 ) {
148 :                           switch( c ) {
149 :                           case '1':
150 :                                   KAng = KAng + 0.0001;
151 :                                   break;
152 :                           case 'q':
153 :                                   KAng = KAng + 0.00001;
154 :                                   break;
155 :                           case 'a':
156 :                                   KAng = KAng - 0.00001;
157 :                                   break;
158 :                           case 'z':
159 :                                   KAng = KAng - 0.0001;
160 :                                   break;
161 :
162 :                           case '2':
163 :                                   KAngSpd = KAngSpd + 0.0001;
164 :                                   break;
165 :                           case 'w':
166 :                                   KAngSpd = KAngSpd + 0.00001;
167 :                                   break;
168 :                           case 's':
169 :                                   KAngSpd = KAngSpd - 0.00001;
170 :                                   break;
171 :                           case 'x':
172 :                                   KAngSpd = KAngSpd - 0.0001;
173 :                                   break;
174 :
175 :                           case '3':
176 :                                   KPwmSum = KPwmSum + 0.0001;
177 :                                   break;
178 :                           case 'e':
179 :                                   KPwmSum = KPwmSum + 0.00001;
180 :                                   break;
181 :                           case 'd':
182 :                                   KPwmSum = KPwmSum - 0.00001;
183 :                                   break;
184 :                           case 'c':
185 :                                   KPwmSum = KPwmSum - 0.0001;
186 :                                   break;
187 :
188 :                           case '4':
189 :                                   KPwm = KPwm + 0.0001;
190 :                                   break;
191 :                           case 'r':
192 :                                   KPwm = KPwm + 0.00001;
193 :                                   break;
194 :                           case 'f':
195 :                                   KPwm = KPwm - 0.00001;
196 :                                   break;
197 :                           case 'v':
198 :                                   KPwm = KPwm - 0.0001;
199 :                                   break;
200 :
201 :                           default:
202 :                                   break;
203 :                           }
204 :                   }
```

キーボード入力によって、定数を増減させることができます。

4.10.6 LED 出力

```
207 :                   if( -1 < Ang && Ang < 1 ){
208 :                           p1 = ( p1 & 0xf0 ) | ( ~0x06 & 0x0f );
209 :                   }else if( -3 < Ang && Ang < -1 ){
210 :                           p1 = ( p1 & 0xf0 ) | ( ~0x0c & 0x0f );
211 :                   }else if( Ang < -3 ){
212 :                           p1 = ( p1 & 0xf0 ) | ( ~0x08 & 0x0f );
213 :                   }else if( 1 < Ang && Ang < 3 ){
214 :                           p1 = ( p1 & 0xf0 ) | ( ~0x03 & 0x0f );
215 :                   }else if( 3 < Ang ){
216 :                           p1 = ( p1 & 0xf0 ) | ( ~0x01 & 0x0f );
217 :                   }
218 :
```

ミニマイコンカー製作キット Ver.2 の CPU 部分に付けた LED の点灯を制御します。

4.10.7 時間待ち

```
220 :                   timer(100);
```

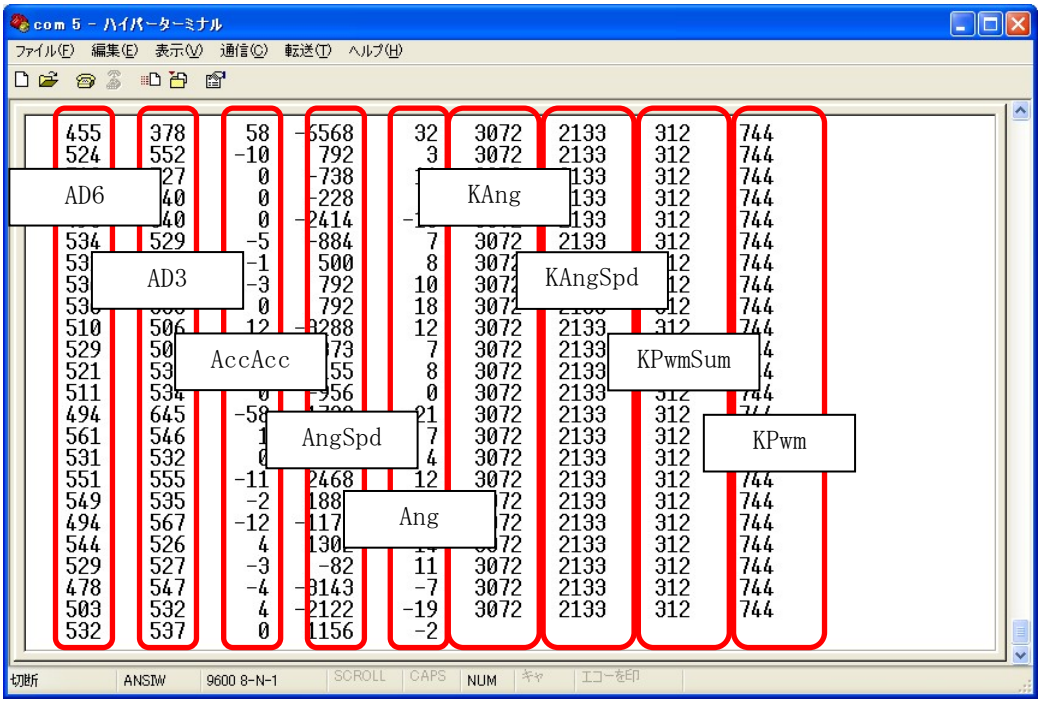
シリアルポートの通信負荷低減のため、待ち時間を設けています。

4.11 ボリューム・パラメーターの調整（必須事項）

ボリューム・パラメーターの調整を行う場合は、ミニマイコンカー製作キット Ver.2 の電源を入れた状態で、ミニマイコンカー製作キット Ver.2 と PC を USB ケーブルで接続し、ハイパーターミナルなどの通信ソフトで設定を行うことができます。

ポートの設定

項目	設定値
ビット／秒	9600
データビット	8
パリティ	なし
ストップビット	1
フロー制御	なし



スタートボタンを押すと、内部の各変数の値が表示されます。

《**重要**》　倒立をさせる前に、ジャイロ静止時の電圧がオペアンプの飽和電圧の中間（約 1.75V）になるように、VR1 のボリュームを調整する必要があります。
モータードライバ側の電源は入れずに、AD6 の数値を、

1024÷5×1.75＝358

になるように調整をしてください。

PC のキーを押すことにより倒立制御の各定数を変更することができます。状態がより安定する数値を見つけ、プログラムの各定数に入力することによって、次回からは安定した状態で倒立できるようになります。

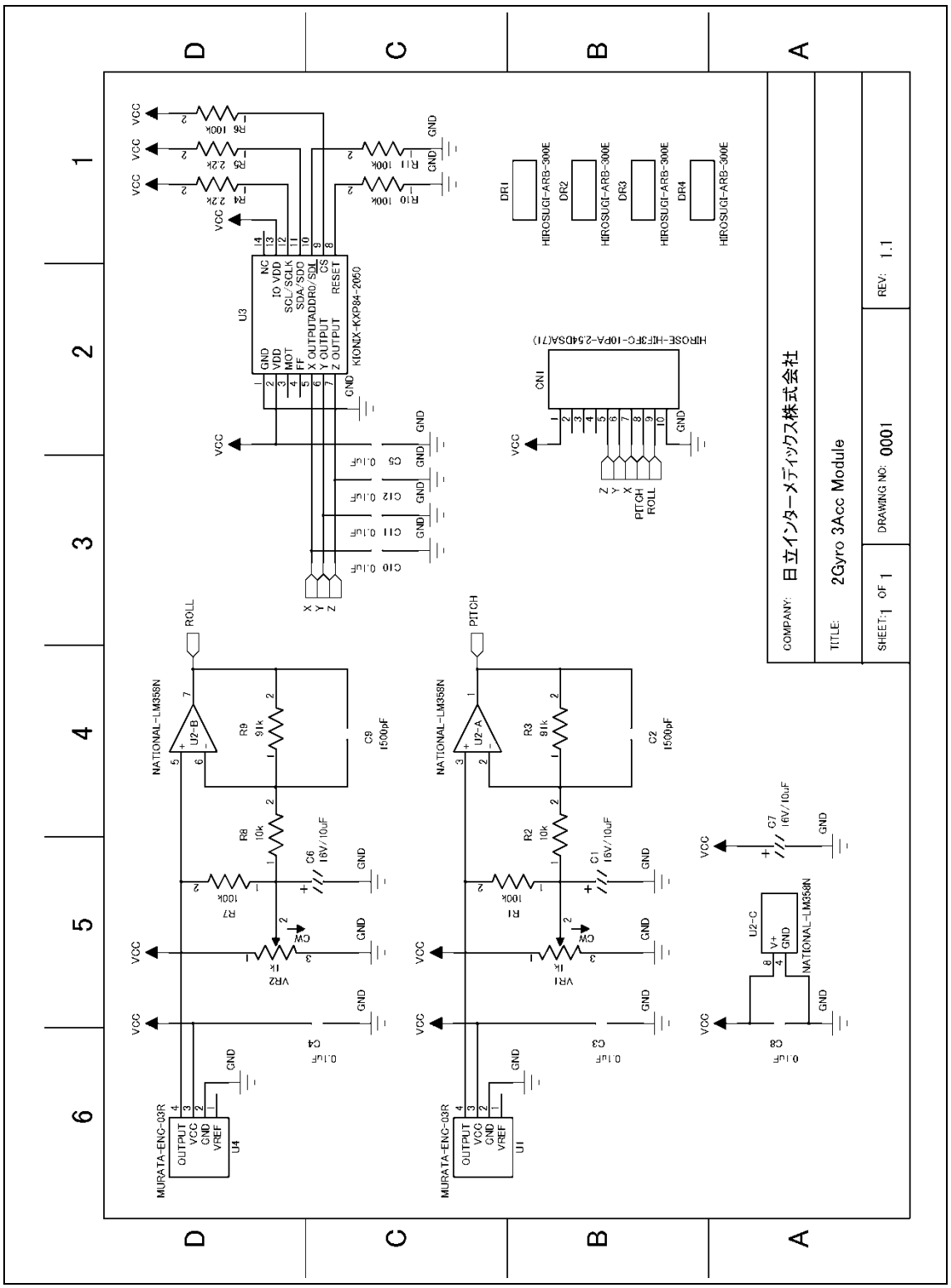
	KAng	KAngSpd	KPwmSum	KPwm
+0.0001	1	2	3	4
+0.00001	Q	W	E	R
-0.00001	A	S	D	F
-0.0001	Z	X	C	V

5. 仕様

5.1 仕様

内容	詳細
電源	DC+5V
ジャイロセンサー	村田製作所 ENC-03RC/ENC-03RD (0.67mV/deg/sec)
3 軸加速度センサー	KIONIX KXP84-2050 (1000mV/G)
I/O	センサー信号出力用コネクタ×1 個

5.2 回路図


COMPANY: 日立インターメディックス株式会社
TITLE: 2Gyro 3Acc Module
DRAWING NO: 0001
REV: 1.1
SHEET: 1 OF 1
U3
KIONIX-KXP84-2050
CN1
U2-A
U2-B
U2-C
NATIONAL-LM358N
U1
U4
MURATA-ENG-03R
DR1
DR2
DR3
DR4
HIROSUGI-ARB-300E
ROLL
PITCH
VCC
GND

5.3 ポート表

コネクタ	番号	端子名	機能
CN1	1	VCC	
	2		-
	3		-
	4		-
	5		加速度センサーZ 軸
	6		加速度センサーY 軸
	7		加速度センサーX 軸
	8		ジャイロセンサーPITCH 軸
	9		ジャイロセンサーROLL 軸
	10	GND	

5.4 ピン配置図

10 ピンコネクタのピン配置