PIC マイコンとジャイロセンサを使った倒立振子

2016 年 7 月 29 日

蚊野　浩

概要

PIC マイコンと 3 軸ジャイロ加速度センサ MPU6050 を使って，二輪倒立振子を試作した．6 個のセンサ出力の中で，一つの角速度値だけを使って倒立させることができた．

１．PIC マイコン

PIC マイコンは，米国マイクロチップ・テクノロジー社が製造するマイクロコントローラ製品群である．非常に多くの製品があり，組み込み機器用マイクロコントローラに適したアーキテクチャを有している．今回は PIC16F886 という製品を利用した．その理由は次の点である．

① 文献[1]で詳細に説明されているので情報を得やすい．

② ２セットの PWM に利用できる CCP1・CCP2 モジュールを持っており，二輪倒立振子の２個の DC モータを容易に PWM 制御できる．

③ 28 ピンの DIP であるため実装が容易．

今回利用した PIC マイコンの開発環境を説明する．PIC マイコンの統合開発環境 MPLAB X IDE を Windows10 PC にインストールした．利用したバージョンは v3.35 である．これに，CCS 社の C コンパイラ（PCM）を組み込み，C 言語のプログラムを開発した．C コンパイラにはマイクロチップ・テクノロジー社の純正品など，幾つかの製品がある．その中で，CCS 社の製品は比較的安価でライブラリが豊富である．PIC の豊富な機能を利用するには，充実したライブラリが重要である．CCS の C は，今回の開発で便利に利用することができた．

PIC マイコンへのプログラムライターとして PICKit3，書き込みアダプターとしてマルツエレックの MPIC-DPPA を利用した（図 1）．

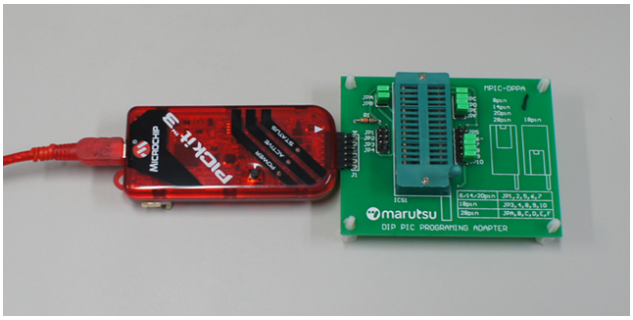

図 1　PICkit3 と MPIC-DPPA

PIC マイコンのソフト開発におけるデバッグ手段として次の方法がある．

① PICkit3 のインサーキットデバッガを使う．文献[2]に記述されているように PICkit3 と PIC マイコンを接続する．この状態で，パソコンから PIC マイコンにプログラムをダウンロードできる．続いて，リアルタイムエミュレーションが可能になる．ステップ実行や変数の値を確認することが可能である．

② PIC マイコンとパソコンを UART で接続する．CCS の関数 printf を使うと，書式付き文字列が UART に出力される．これをパソコン側で表示することで，プログラムの状態を確認できる．

２．　３軸ジャイロ加速度センサ MPU6050

MPU6050 は InvenSense 社の 3 軸加速度ジャイロセンサである．これをモジュール化した GY-521 を利用した（図 2）．GY-521 と PIC マイコンを I2C インタフェースで接続する．接続の詳細については後述する．MPU6050 は 3 軸回りの角速度を 3 つの数値として，3 軸方向の加速度を 3 つの数値として，合計 6 個の数値を出力する．今回は，x 軸回りの角速度だけを利用する．

図 2　MPU6050（中央のチップ）と GY-521

３．　二輪倒立振子メカ

タミヤの楽しい工作シリーズ No.168「ダブルギアボックス（左右独立 4 速タイプ)」をギア比 38.2:1 で組み立てた．これに，同シリーズ No.193「スリムタイヤセット」の 55mm 径のタイヤを装着し，同シリーズ No.98「ユニバーサルプレートセット」にネジ止めした．図 3 に回路とバッテリーも実装した状態での外観を示す．

なお，本研究で参考にした Web サイト[4]は，ダブルギアボックス（左右独立 4 速タイプ)」をギア比 114.7:1 で利用している．今回の実験でも，当初はこのギア比で組み立てたが，減速しすぎで，応答が遅れているようであった．

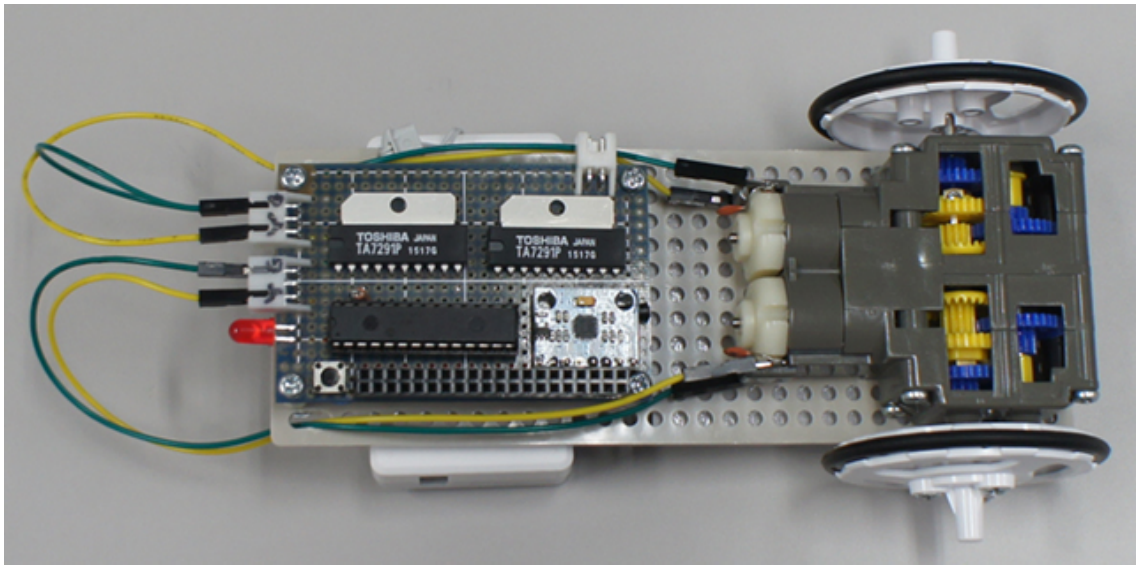

図 3 二輪倒立振子

4． 回路

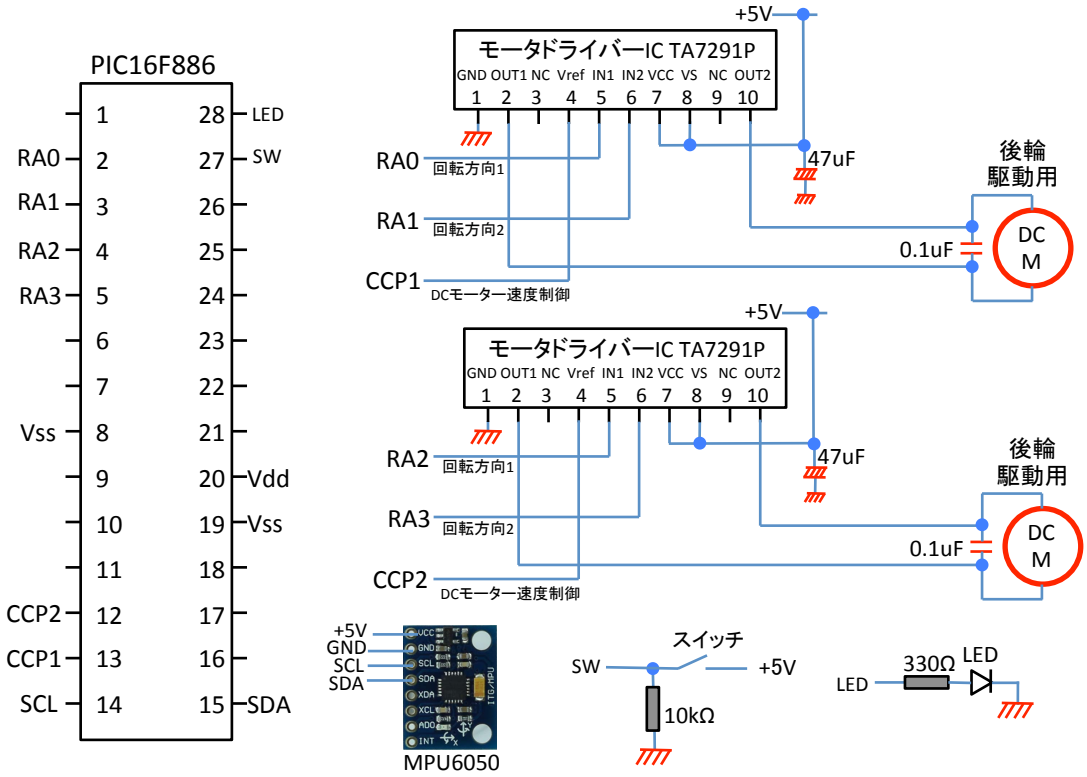

図 4 回路図

図 4 に回路図を示す．モータードライバーIC TA7291P の機能は，5・6 番ピンに回転方向を指示するパターンを与え，4 番ピンに PWM のパルス信号を加えることで，回転方向と回転力を制御することである．このように，PIC マイコンを用いることで，デジタル制御に必要な回路部分のほとんどを PIC マイコン内部に取り込むことができる．

５． 倒立振子の制御

今回試作する二輪倒立振子は，図 5 のように車体を振り上げた状態で，二輪を前進・後退させながら倒れないようにバランスをとる装置である．

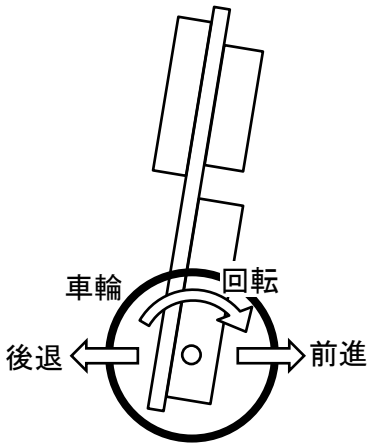

図 5　二輪倒立振子の運動

時刻 t における装置の運動状態を表す変数を次のように定義する．

angle(t)： 車体がバランスした状態からの回転角度

angle_velocity(t)： 車体の角速度

velocity(t)： 車軸の移動速度

distance(t)： 車軸の移動量

この時，二輪倒立振子を静止させるために車輪に加える制御量を u(t)とすると，

u(t) = F1*angle + F2*angle_velocity + F3*velocity + F4*distance　　・・・(1)

となることが知られている[3].

式(1)右辺は４つの項の和になっている．それぞれの項の働きは，次のようなものであると理解している．

F1*angle は車体をバランスする方向に車輪を進めるための項である．小さく回転しておればそれを補正するための制御量は小さく, 大きく回転しておればそれを補正するための制御量は大きくなる. この項だけで概ね倒立させることが可能であるように思える. しかし実際には安定せず, 振動する. その理由は, 角度 0 の位置を行き過ぎる時に, それを止めるための補正が働かないからである.

F2*angle_velocity は車体の回転運動を抑制するための項である．角度 0 の位置をある程度の角速度で行き過ぎ場合に，それを止めるための補正項として働く．

このように考えると，最初の２項で倒立振子を立てることができる．しかし，二輪倒立振子が安定している状態（角度 0，角速度 0）であっても，車体が一定速度で移動している状態がありえる．この状態を解消して，車速 0 で安定させるための項が

F3*velocity である．さらに最後の項 F4*distance は，制御を開始した位置で静止させるための項であると考えられる．

６． 二輪倒立振子のプログラム

次ページ以降のリストを参照しながら二輪倒立振子のプログラムを説明する．

4 行目から 8 行目は，PIC マイコンのコンフィグレーションを設定するための，CCS の書き方である．これらの意味は文献[1]に説明されている．5 行目は PICkit3 のインサーキットデバッガを使う場合の書き方の例，8 行間は通常の使い方をする場合の例である．“BROWNOUT”は電源電圧が低下した時にリセットすることを意味するパラメータである．3.3V で動作させるには“NOBROWNOUT”を指定する必要がある．

11 行目で内部クロックとして 8MHz を使うことを宣言している．内部クロックの種類はコメントで記述しただけの周波数がある．二輪倒立振子は，ある程度高速に制御する必要があるので，結局，8MHz のクロックを使うことになった．

14 行目で I2C をマスターとして使うことを宣言している．

16 行目から 27 行目は MPU6050 自身のアドレスとその内部レジスタのアドレスを宣言している．MPU6050 の I2C アドレスは 0x68 である．それを PIC マイコン側で設定する場合，7 ビットアドレスが 8 ビットの上位に置かれるので，0xD0 になる．

30 行目は PIC のタイマー0 のカウント値となる定数を宣言している．この例では，後で説明するように、タイマー0 のクロックが 32 μ sec になっており，その状態で 4msec をカウントするための数値が 130 である．なお，4msec は二輪倒立振子の制御周期である．

38・39 行目は IO ポートの使い方に関する宣言である．PIC16F886 は A ポート，B ポート, C ポートの 3 つの IO ポートを持っている．これらは，端子ごとに入出力を指定することができる．fast_io(x)と指定すれば，一旦，IO ポートの入出力の方向を設定すれば（set_tris_x()関数で），それ以降，方向を変えないため，IO ポートを高速にアクセスできる．IO 端子へのアクセスは，output_low(PIN_B7)などでできるが，fast_io()をしておけば処理が高速になるということである．

42 行目から 45 行目で，運動状態を表す 4 の変数をグローバル変数として宣言する．

51 行目の最初に int8 i; と宣言している．CCS の C 言語が通常の C 言語と大きく異なることは，データ型の制約が大きいことである．int8 は 8 ビット整数という意味である．データ型の制約については文献[1]を参照のこと．

53 行目から 59 行目で A ポートと B ポートの初期設定を行っている．

61 行目から 67 行目で MPU6050 を起動している．

69 行目から 73 行目で LED を 1 秒間隔で 3 回点滅させている．この間に装置を平坦な場所に静置する．

75 行目から 80 行目で x 軸回りの角速度を 255 回読み取り，その平均を av_offset としている．装置は静止させているので，av_offset は静止時にも発生する角速度のオフセットである．

82 行目から 86 行目で LED を 0.2 秒間隔で 3 回点滅させている．これで，角速度のオフセットを計測し終わったことを示す．

88 行目から 90 行目は，PIC16F886 が内蔵する 2 つの CCP モジュールを PWM で利用することを宣言する．CCP モジュールの説明は文献[1]を参照のこと．

94 行目は, PWM に用いるタイマー2 の動作を設定する．システムクロックが 8MHz（0.125μsec 周期）のとき，タイマー2 に入力されるクロックはその 4 倍の 0.5μsec になる．これをさらに分周したものをカウントして PWM の周期とする．94 行目の設定は 16 分周して 100 カウントするという意味であるから, コメントのように 800μsec が PWM の周期である．

96 行目と 97 行目は PWM のデューティ比を設定する．ここで設定する数値は 94 行目の setup_timer_2()関数の第 2 引数の数値（この例では 100）よりも小さくする．

101 行目は二輪倒立振子の制御周期を決めるタイマー0 の動作を設定する．setup_timer_0 関数で，カウンタへの入力クロックの周期 0.5μsec を 64 分周したもの（したがって 32μsec）を 8 ビットの数値でカウントすることを指定している．102 行目の数値を設定することで，タイマー0 の周期は 4msec となる．103 行目から 105 行目でタイマー0 による割り込みを許可している．

107 行目から 118 行目の無限ループの間に，後で説明する割り込みが 4msec 周期で発生する．この無限ループの中でスイッチが押されるとモータの動作をストップし，幾つかの運動パラメータを 0 にリセットする．

120 行目から 183 行目が二輪倒立振子の制御を行う割り込み処理である．126 行目と 185 行目で LED を ON/OFF し，処理時間などを確認できるようにしている．

129 行目で，次の割り込みのためにタイマー0 に値をセットする．

132 行目で MPU6050 から生の角速度値を読み込み, オフセット値を引くことで高精度化している．

135 行目で角速度を積算することで角度を求めている．

138 行目から 142 行目で 4 つの運動パラメータに対するゲインを設定し, 149 行目でそれらを足し合わせることで制御量を求める．

制御量の数値を PWM のデューティ比に換算する. デューティ比の値は 1 から 99 までを用いることにした．制御量が大きければデューティ比を大きくする．152 行目の変数 max_pwr に制御量が飽和する値を設定し，153 行目の変数 min_pwr に制御を有効にする最小の制御量を設定する．max_pwr を大きな値にすれば，制御はゆるやかになり，小さい値にすれば制御が急速に働く．

170 行目から 183 行目で新たなデューティ比を設定し，velocity，distance の値を更新する．

180 行目から 222 行目は MPU6050 にアクセスするための関数である．ここに記述するように I2C の制御信号をコントロールする．

```
#include <16f886.h>
#include <math.h>

// Pickit3 のインサーキットデバッグ機能を使う場合の CONFIG, DEBUG（NODEBUG ではなく）, MCLR（NOMCLR ではなく）
//#fuses NOPROTECT,DEBUG,NOWRT,NOCPD,NOLVP,BROWNOUT,PUT,MCLR,NOWDT,INTRC,NOIESO,NOFCMEN,BORV40

// Pickit3 のインサーキットデバッグ機能を使わない場合の CONFIG, NODEBUG, MCLR
#fuses NOPROTECT,NODEBUG,NOWRT,NOCPD,NOLVP,NOBROWNOUT,PUT,MCLR,NOWDT,INTRC,NOIESO,NOFCMEN,BORV40

//内部クロック 8MHz で利用する。
#use delay(internal = 8MHZ) //(8MHZ,4MHZ,2MHz,1MHZ,500kHz,250kHZ,125kHz,31kHz)

// I2C の定義。16F886 の場合にはハードウエアで実行する。
#use i2c(MASTER, SCL=PIN_C3, SDA=PIN_C4, FAST, FORCE_HW)

//MPU6050 とその内部レジスタのアドレス
#define MPU6050         0xD0 //0x68 を 1 ビット左シフト
#define MPU6050R        0xD1 //Read する場合、最下位ビットを 1 にする
#define POWER_MGMT      0x6B
#define CONFIG_R        0x1A
#define GYRO_CONFIG     0x1B
#define AX_ADR          0x3B
#define AY_ADR          0x3D
#define AZ_ADR          0x3F
#define GX_ADR          0x43
#define GY_ADR          0x45
#define GZ_ADR          0x47

//タイマー 0 のカウント値。例えば 4ms 周期の場合、255 - (4msec/32) = 130
#define T0COUNT         130

//関数のプロトタイプ宣言
void intTimer0();
int8 mpu6050_write(int adr, int data);
int8 mpu6050_read(int adr);
float mpu6050_rawdata(int8 reg);

#use fast_io(A)
#use fast_io(B)

//グローバル変数
float av_offset;//角速度のオフセット
float angle = 0.0f;//x 軸回りの角度
float velocity = 0.0f;//車速の推定値
float distance = 0.0f;//移動距離の推定値

#define M_PI 3.1415926f

/**** main 関数 ****/
void main() {
    int8 i;

    //ポート A の下位 4 ビットを出力、上位 4 ビットを入力に設定する。fast_io(x)を使う場合、最初に、set_tri_x()
で端子の入出力方向を決める。
    set_tris_a(0xf0);
    output_a(0x00);//0x06:順, 0x09:逆, 0x00:ストップ, 0x0f:ブレーキ
    //ポート B の下位 7 ビットを入力、B7 を出力に設定する。
    set_tris_b(0x7f);
    output_low(PIN_B7);//LED を消灯
```

```
60
61      //MPU6050 のパワーオンでエラーが発生すれば B7 端子を 0.1sec でトグルする
62      if (mpu6050_write(POWER_MGMT, 0x00) != 0) {
63          while (1) {
64              output_toggle(PIN_B7);
65              delay_ms(100);
66          }
67      }
68
69      //MPU6050 のパワーオンでエラーがない場合、LED を 3 回点滅させる。この間に、装置を平坦な場所に置く。
70      for (i = 0; i < 6; i++) {
71          output_toggle(PIN_B7);
72          delay_ms(1000);
73      }
74
75      //平坦な場所に置いた状態で、角速度のオフセットを求める
76      av_offset = 0.0f;
77      for (i = 0; i < 255; i++) {
78          av_offset += mpu6050_rawdata(GX_ADR);
79      }
80      av_offset = av_offset/255.0f;
81
82      //角速度のオフセットを求めたことを示すために、素早く LED を 3 回点滅させる。
83      for (i = 0; i < 6; i++) {
84          output_toggle(PIN_B7);
85          delay_ms(200);
86      }
87
88      //ccp1 と ccp2 を PWM で使う。ccp1 と ccp2 はモータの PWM 制御に利用する。
89      setup_ccp1(CCP_PWM);
90      setup_ccp2(CCP_PWM);
91
92      //PWM に使うタイマー 2 を(T2_DIV_BY_16,100,1)と設定すると、PWM の周期は 0.5usec×16×100=800usec になる
93      //ここで 0.5usec は 8MHz の 1 周期 0.125usec の 4 倍の値。
94      setup_timer_2(T2_DIV_BY_16,100,1);
95      //次の関数で設定するデューティ比の数値は、setup_timer_2()の第二引数よりも小さい数値に設定する。
96      set_pwm1_duty(0);
97      set_pwm2_duty(0);
98
99      //周期的に発生する割込み用にタイマー 0 を使用する
100     //タイマー 0 のクロックは 0.5us×64=32us、カウンタは 8bit。ここで 0.5usec は 8MHz の 1 周期 0.125usec の 4 倍
101 の値。
102     setup_timer_0(RTCC_INTERNAL | RTCC_DIV_64 | RTCC_8_BIT);
103     set_timer0(T0COUNT);
104     //タイマー 0 による割込みを許可する
105     enable_interrupts(INT_TIMER0);
106     enable_interrupts(GLOBAL);
107
108     while(1) {
109         //スイッチを押すとモーターを止め、角度・速度・移動距離を 0 にする。
110         if (input(PIN_B6)) {
111             output_a(0x00);//0x06:順，0x09:逆，0x00:ストップ，0x0f:ブレーキ
112             set_pwm1_duty(0);
113             set_pwm2_duty(0);
114             angle = 0.0f;
115             velocity = 0.0f;
116             distance = 0.0f;
117         }
118     }
```

```
}

//タイマー０で周期的に割込みを発生させ、MPU6050 の値を読みとり、メカを制御する。
//2016 年 7 月 29 日のバージョンでは、割込み周期を 4msec に設定し、intTimer0()の処理を 2msec で終了している。
#INT_TIMER0
void intTimer0() {

    //タイマーによる割込みが発生していることを確認するために B7 をトグルする
    output_high(PIN_B7);

    //次の割込みのためにカウンタをセットする。
    set_timer0(T0COUNT);

    //角速度（度/sec 単位）を求める。
    float angular_velocity = mpu6050_rawdata(GX_ADR) - av_offset;

    //角速度を積算して角度を求める。
    angle += angular_velocity;

    //制御量を求める
#define F1 1.0f //角度に対するゲイン(1.0 に固定。第一にこれを固定して、max_pwr を調整する。)
#define F2 4.0f //角速度に対するゲイン（4.0。第二に調整する。小さいと制御が弱く、大きいと振動する。）
#define F3 50.0f //車速に対するゲイン（50。小さいと大きく前後移動。大きくすると前後幅が小さくなるが、大きすぎると破たんする。）
#define F4 0.0f //移動距離に対するゲイン。うまく動作しないので、0 に設定している。

    float f1angle    = F1*angle;
    float f2angvel   = F2*angular_velocity;
    float f3velocity = F3*velocity;
    float f4distance = F4*distance;

    float pwr = f1angle + f2angvel + f3velocity + f4distance;//制御量

    //制御量の絶対値の最大（F1=1.0 で、この値を適切に設定する。小さいと制御が遅くなり、大きいと振動する。）
    float max_pwr = 200000.0f;
    float min_pwr =  1.0f;//制御量の絶対値の最小（この値以下ではモータを駆動しない。あまり意味がない。）

    //pwm のデューティ比を設定する
    int8 duty;//pwm のデューティ比
#define DUTY_MAX 99.0f //PWM の最大デューティ比
#define DUTY_MIN 1.0f //PWM の最小デューティ比

    if ((pwr >= max_pwr)||(pwr <= -max_pwr)) {//制御量の絶対値が max_pwr を超えた場合
        duty = DUTY_MAX;
    } else if (pwr >= min_pwr) {//制御量が min_pwr?max_pwr の場合
        duty = (int8)(DUTY_MIN + (DUTY_MAX-DUTY_MIN)*pwr/max_pwr);
    } else if (pwr <= -min_pwr) {//制御量が-min_pwr?-max_pwr の場合
        duty = (int8)(DUTY_MIN + (DUTY_MAX-DUTY_MIN)*(-pwr)/max_pwr);
    } else {
        duty = 0;
    }

    if (pwr >= 0) {
        output_a(0x06);//0x06:順，0x09:逆，0x00:ストップ，0x0f:ブレーキ
        set_pwm1_duty(duty);
        set_pwm2_duty(duty);
        velocity += (float)duty;
        distance += (float)velocity;

```

```
178         } else {
179             output_a(0x09);//0x06:順，0x09:逆，0x00:ストップ，0x0f:ブレーキ
180             set_pwm1_duty(duty);
181             set_pwm2_duty(duty);
182             velocity -= (float)duty;
183             distance -= (float)velocity;
184         }
185 
186         output_low(PIN_B7);
187     }
188 
189     int8 mpu6050_write(int adr, int data) {
190         int ack = 0;
191 
192         i2c_start();
193         ack += i2c_write(MPU6050);
194         ack += i2c_write(adr);
195         ack += i2c_write(data);
196         i2c_stop();
197 
198         return (ack);
199     }
200 
201     int8 mpu6050_read(int adr) {
202         int val;
203 
204         //MPU6050 と読み出しレジスタを指定する
205         i2c_start();
206         i2c_write(MPU6050);
207         i2c_write(adr);
208 
209         //指定したレジスタから読み出し
210         i2c_start();
211         i2c_write(MPU6050R);//読み出しの場合には、最下位ビットを 1 にする。
212         val = i2c_read(0); //0 を指定するとデータを正しく受け取るようである。
213         i2c_stop();
214 
215         return val;
216     }
217 
218     float mpu6050_rawdata(int8 reg) {
219         int8 high = mpu6050_read(reg);
220         int8 low  = mpu6050_read(reg+1);
221 
222         return (float)((signed int16)make16(high,low));
223     }
224 
```

文献

[1] 改訂版 C 言語による PIC プログラミング入門，後閑哲也，技術評論社，2009 年.

[2] http://ww1.microchip.com/downloads/jp/DeviceDoc/52010A_JP.pdf

[3] フィードバック制御による倒立ロボットの製作，川村伸司，CQ 出版 Interface，pp.70-77，2006 年 7 月号.

[4] http://www.instructables.com/id/倒立振子の研究/