２０２１年１１月１５日（林原，王、藤井）

ロボット速度制御の基礎と実験

１．ハードウェアを制御するプログラム

ロボットのモータを制御するためには，モータドライバＩＣに適切な信号を加える必要があります． なお、左右どちらのモータが，どの方向に回転するかは，モータへの配線によって決まります．各ロボットにより変わってきますので，今後のサンプルプログラムの動作がおかしいと思った場合は調整してください．（H/Wで調整しても，プログラムで調整してもどちらでも構いません．）

【課題１】上記に従い，ロボットが全力（最高速度）で走行するプログラムを作成し，ロボットの最高速度を計測しましょう．実験は複数回を行い，複数回のデータを記録，分析します．

（同じ条件で５回以上の実験を行い、速度を求めるために必要な距離・時間データを記録する。速度を計算した上、各実験データと共に平均値と最大誤差、もしくは分散をレポートに記入する）

以下は左のモータを動くサンプルプログラムですので，右のモータを駆動するコマンドを追加して，ロボットの両輪で全力前進をするプログラムを完成してください．

```
#include ″mbed.h″
#include ″Motor.h″
#include ″adrobo.h″

BusOut led(LED1,LED2,LED3,LED4);
Motor motor_left(MOTOR11,MOTOR12);

int main() {
    float i= 1.0f;
    motor_left.setMaxRatio( 1.0 );        /* 初期設定が最大出力は 25%か 100%に設定*/
    while(1) {
            motor_left = i;
            }
    }
```

２．ＰＷＭ制御

ロボットの全力走行に関して，非常に速いと感じたと思います．図形を書くためには，もっとゆっくり正確に移動することが必要になります．ゆっくり動くためには、PWM制御を使用します．今回は，CHOP(PWM用の端子)が Lの時にモータに電流が流れるので，図の Lの幅を調整します．ロボット電子回路で方法に関しては，学びましたので詳細は省きますが，上記のプログラムで，制御することができます．

【課題２】１mの直線を５秒間で走行するように，モータに加える PWMのデューティ比(0～1.0)を調整してください．走行データ（時間、距離）を計測し、実際の走行速度を計算します。

３．ロータリエンコーダのカウント

上記で様々な調整を行ったと思いますが，おそらくは電池やコースの路面など，少しの変化で大きく挙動が変化することに気づいたと思います．これは，回転数を計測せずに適当に電流を流しているからです．環境からの影響を少なくするためには，減速比を大きくすればいいのですが，緩慢な動きしかできなくなってしまいます．

今回の H/W には，モータの回転数を計測することができるように，ロータリエンコーダが装着されています．では，ロータリエンコーダをカウントするプログラムを実行してみましょう．

割込処理

ロボットを制御するためには，多くの場合，複数の処理を同時に行わなくてはなりません．今回の場合は，「ロータリエンコーダのパルスをカウントしながら」「結果を表示する」「モータに加える信号を変更する」などです．同じ周期で行うのであれば，順番に行えば良いのですが，違う場合は割り込み処理を使います．これは，あるイベントが起きたときに，今の処理を一時終始して，別の処理を行い，終了後にまた再開するというものです．割込処理は速やかに終了して，本来の処理に戻すことが重要です．

【課題３】エンコーダをカウントし，車輪の速度を測定するプログラムを作成して，車輪が回転時のエンコーダ値を確認しましょう．

```
#include "mbed.h"
#include "QEI.h"
#include "adrobo.h"
BusOut led(LED1, LED2, LED3, LED4);
BusIn in(GPIO1, GPIO2, GPIO3, GPIO4);
QEI qei_left(GPIO1, GPIO2, NC, 48, QEI::X4_ENCODING);
QEI qei_right(GPIO3, GPIO4, NC, 48, QEI::X4_ENCODING);

const double sampling_time = 0.020;
const double move_per_pulse = 0.0005;

int main() {
  in.mode(PullUp);

  while(1) {

            int enc_left = qei_left.getPulses();
            int enc_right = qei_right.getPulses();

            qei_left.reset();
            qei_right.reset();
```

```
            double speed_left = move_per_pulse * enc_left / sampling_time;
            double speed_right = move_per_pulse * enc_right / sampling_time;

            printf("%f %f¥r¥n",speed_left,speed_right);

            speed_left = speed_right = 0.0;

            wait(0.1);
    }
  }
```

４．速度制御

通常のロボットの速度制御には，PD 制御，PI 制御と PID 制御などがあります．その P は制御の比例項（Proportional Control Part），D は微分項（Differential Control Part），I は積分項（Integral Control Part）とそれぞれ意味をしています．今回利用する制御アルゴリズムは，エンコーダの性能上の理由で，PI 制御則に限定しています．制御則は以下の式で表せます．

$$E = V_d - V, \qquad \tau = K_P E - K_I \int E$$

ここで，V_d は速度の目標値，V は現在計測された速度とし，速度の誤差は E となります．K_P は比例制御ゲイン，K_I は積分制御ゲインと呼びます．計算された制御出力 τ をモータへ出力し，その車輪の速度制御を行います．

【課題４】PI 制御のサンプルプログラム（スタートガイドテキストの P46）の比例ゲイン Kp と積分ゲイン Ki をチューニングし，よりよい速度制御を実現しましょう．この課題に関しては，以下の四つの実験をそれぞれ複数回実行し，データを記録して，平均値や最大誤差値などを計算・分析し，レポートを作成しましょう．

実験 4-1： 制御の I の成分を入れない， P 成分のみの実験をしてみる。
（Ki=0.0，Kp=0.1） 目標速度は 20cm 毎秒
無負荷の走行と負荷あり（斜面か箱押し）の走行時のロボットの速度を計測する

実験 4-2： 制御ゲインの調整（チューニング）： 目標速度は 20cm 毎秒
P 制御ゲイン Kp を走行時車輪が発振するまで，徐々にあげ，走行時の発振直前の値を探し出す．その値の 0.8 倍ぐらいを最適な P 制御ゲインにする
そのゲインでの負荷ありの走行時のロボットの速度を計測する

実験 4-3： 制御の I 成分（ Ki ）を復活し，無負荷と負荷有りの走行の速度を計測し，
I 成分のない実験の結果を比較し，PI 制御の I 成分の効果を確認する．

実験 4-4： チューニングした制御ゲインでロボットの走行性能（異なる速度）
を確認する．

```
double speed_left_ref = 0.2, speed_right_ref = 0.2;
double speed_left_lpf = 0.0, speed_right_lpf = 0.0;
const double sampling_time = 0.020;
const double move_per_pulse = 0.0005;
double i_left = 0.0, i_right = 0.0;
const double ki = 10.0, kp = 1.0;

double low_pass_filter(double val, double pre_val, double gamma) {
        return gamma * pre_val + (1.0 - gamma) * val;
}

void control_handler() {
        int enc_left = qei_left.getPulses();
        int enc_right = qei_right.getPulses();
        qei_left.reset();
        qei_right.reset();
        double speed_left = -move_per_pulse * enc_left / sampling_time;
        double speed_right = move_per_pulse * enc_right / sampling_time;
        speed_left_lpf = low_pass_filter(speed_left, speed_left_lpf, 0.4);
        speed_right_lpf = low_pass_filter(speed_right, speed_right_lpf, 0.4);
        double delta_speed_left = speed_left_ref - speed_left_lpf;
        double delta_speed_right = speed_right_ref - speed_right_lpf;
        i_left += delta_speed_left * sampling_time;
        i_right += delta_speed_right * sampling_time;
        motor_left = kp * delta_speed_left + ki * i_left;
        motor_right = kp * delta_speed_right + ki * i_right;
}

int main() {
        in.mode(PullUp);
        motor_left.setMaxRatio(1.0);
        motor_right.setMaxRatio(1.0);
        control.attach(&control_handler, sampling_time);
        ……
}
```

速度制御のプログラムをビルドするためには、オプション　-Os balanced　が必要です。

速度制御レポート：提出期限（2021 年 12 月 20 日　23：50）

内容：　（実験目的，実験手法，実験データと考察が必要）

1．ロボットの最高速度の特性を実験で確認（課題１）

2．ロボットの定速走行特性を実験で確認（課題 2）（走行直進性，距離精度）

3．車輪エンコーダによる回転速度の計測（課題 3）（今回はカウンタ値の変化を写真記録）

4．速度制御特性に関する実験　（課題４）

P 制御の効果，I 制御の効果の確認　（実験 4-1，4-2，4-3）
ゲインチューニングにより PI 制御での特性の確認（実験 4-4）

5．軌道制御により図形作成実験　（12 月 12 日の軌道制御の回で詳細説明）