＜第 2 週目：超音波センサと DC モータを用いた制御実験＞

1　使用機材

・PC
・BeagleBone Black(BBB)
・USB ケーブル
・超音波センサ(PARALLAX 社，PING))) Ultrasonic Distance Sensor)＋接続回路
・DC モータ(AO-8041　ギヤヘッド用　380 モータ)
・モータドライバ回路(モータドライバ IC：東芝　TA8429HQ)
・モータ駆動用蓄電池

1.1 モータドライバ IC

BBB の GPIO からの出力は小さいため，モータと BBB を直接繋いでも動かない．本実験で使用するモータの動作電圧は，7.2[V]である．そのため，BBB から「H(1)」が出力されている時に，モータへ 7.2[V]の電圧がかかる必要がある．さらに，「ブレーキ」，「正転」，「逆転」等のモータの回転方向を制御するために，「モータドライバ IC」と呼ばれる IC を使用する．このような IC は，モータの制御においては，一般的に使用されている．本実験では，東芝製 TA8429H を使用する．表 1.1 に，本実験で使用するモータドライバ IC の入出力ファンクションを示す. モータドライバ IC は, 比較的高い電圧を使用するため，発熱するので，手で触る時には注意が必要となる．

表 1.1　TA8429H の入出力ファンクション

入力			出力		出力モード
IN1	IN2	ST	OUT1	OUT2	
H	H	H	L	L	ブレーキ
L	H	H	L	H	逆転(正転)
H	L	H	H	L	正転(逆転)
L	L	H	OFF(ハイ　インピーダンス)		ストップ
H/L	H/L	L	OFF(ハイ　インピーダンス)		スタンバイ

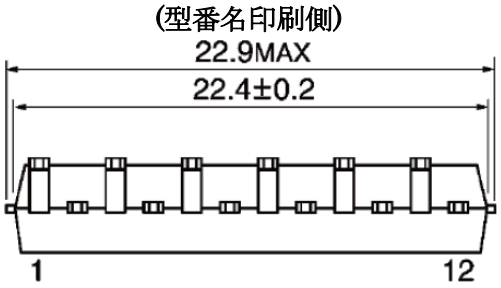

図 1.1　TA8429H のピン配置

本実験で使用するモータドライバ IC は，12 本のピンを有している．図 1.1 にピン配置，表 1.2 にそれぞれの端子の説明を示す．BBB の GPIO と接続する端子は，「IN1」，「IN2」，「ST」である．モータと接続する端子は，「OUT1」と「OUT2」である．「Vs」と「Vcc」，「GND」は蓄電池と接続する．

表 1.2　TA8429H の端子説明

端子番号	端子記号	端子説明
1	***IN1***	**出力の状態を制御する入力端子**
2	***IN2***	**PNP タイプの電圧コンパレータを内蔵**
3	N.C	Non Connection
4	OUT1	DC モータがつながる端子で Sink、Source とも 3[A]の電流容量 また, モータの逆起電圧吸収用のダイオードを VCC 側と GND 側に内蔵
5	N.C	Non Connection
6	GND	接地端子
7	N.C	Non Connection
8	OUT2	④ピンとの間にモータがつながる端子 ④ピンと同等の機能を持ち，①，②ピンにより制御
9	N.C	Non Connection
10	Vs	出力部電源端子
11	Vcc	制御部電源端子，Vs とは完全に分離
12	***ST***	**OPEN または GND にすると出力は OFF**

2　DC モータ駆動実験

2.1 DC モータ駆動実験 1

BBB，モータドライバ回路，DC モータを用いて，DC モータを「正転」，「逆転」，「ブレーキ」させなさい．この時，モータの回転速度の制御は不要である．BBB からは，「H(1)」と「L(0)」の情報を GPIO から, モータドライバ回路へ送ることでモータの駆動を制御させる. モータドライバ IC の「IN1」と「IN2」に BBB の GPIO から信号を送る．モータドライバ IC の「ST」には，BBB の「SYS_5V」を接続させる．プログラムは，次のプログラムの空欄を埋めて完成させなさい．

```
1：  //DC モータの正転，逆転，停止
2：  #include <stdio.h>
3：  #include <unistd.h>
4：  #include <stdint.h>
5：  #include <stdlib.h>
6：  #include <string.h>
```

```
 7 :  #include <dirent.h>
 8 :  #include <fcntl.h>
 9 :  #include <sys/mman.h>
10 :  #include <poll.h>
11 :
12 :
13 :  //gpio の有効化関数
14 :  void gpio_export(int n){
15 :            int fd;
16 :            char buf[40];
17 :
18 :            sprintf(buf, "%d", n);
19 :
20 :            fd = open("/sys/class/gpio/export", O_WRONLY);
21 :            write(fd, buf, strlen(buf));
22 :            close(fd);
23 :  }
24 :
25 :  //gpio の有効化解除の関数
26 :  void gpio_unexport(int n){
27 :            int fd;
28 :            char buf[40];
29 :
30 :            sprintf(buf, "%d", n);
31 :
32 :            fd = open("/sys/class/gpio/unexport", O_WRONLY);
33 :            write(fd, buf, strlen(buf));
34 :            close(fd);
35 :  }
36 :
37 :  //gpio の設定ファイルを開く関数
38 :  int gpio_open(int n, char *file, int flag){
39 :            int fd;
40 :            char buf[40];
41 :
42 :            sprintf(buf, "/sys/class/gpio/gpio%d/%s", n, file);
43 :
44 :            fd = open(buf, flag);
45 :            return fd;
46 :  }
47 :
48 :  int main(void)
49 :  {
50 :            int i, fd;
51 :            FILE *fp;
52 :            int gpio1=          ;          //IN1 への GPIO 番号
53 :            int gpio2=          ;          //IN2 への GPIO 番号
54 :            char path2[60], path3[60];/*デバイスファイルのパス*/
55 :            int in1=0, in2=0;
56 :
57 :
58 :            //gpio1 の有効化
59 :                                        ;
60 :
61 :            //gpio1 を出力(out)に設定
62 :            fd=gpio_open(gpio1, "                ", O_WRONLY);
63 :            write(fd, "          ", 3);
```

```
 64 :            close(fd);
 65 :
 66 :            //gpio1 から「0」を出力
 67 :            sprintf(path2, "/sys/class/gpio/gpio%d/value", gpio1);
 68 :            fp=fopen(path2, "wb");
 69 :            fprintf(fp, "%d", 0);
 70 :            fclose(fp);
 71 :
 72 :
 73 :            //gpio2 の有効化
 74 :            [                                  ];
 75 :
 76 :            //gpio2 を出力(out)に設定
 77 :            fd=gpio_open(gpio2, "[                  ", O_WRONLY);
 78 :            write(fd, "[        ", 3);
 79 :            close(fd);
 80 :
 81 :            //gpio2 から「0」を出力
 82 :            sprintf(path3, "/sys/class/gpio/gpio%d/value", gpio2);
 83 :            fp = fopen(path3, "w");
 84 :            fprintf(fp, "[        ", [        ]);
 85 :            fclose(fp);
 86 :
 87 :            sleep(1);
 88 :
 89 :            //モータ回転の無限ループ（「5」を入力で終了）
 90 :            while(1){
 91 :                     printf("In1 を入力してください．(1 or 0 で回転方向指定 , 5 でプログラム終了)¥n");
 92 :                     scanf("[        ", [        ]);          //キーボードから，in1 へ数値代入
 93 :                     printf("in1= %d¥n", in1);
 94 :
 95 :                     sprintf(path2, "/sys/class/gpio/gpio%d/value", gpio1);
 96 :                     fp=fopen(path2, "wb");
 97 :
 98 :                     //in1 が「5」の時，GPIO から「0」を出力してループ終了
 99 :                     if([                ){
100 :                              fprintf(fp, "%d", 0);
101 :                              fclose(fp);
102 :                              [                ];
103 :                     }
104 :                     //in1 が「0」,「1」,「5」以外の時は，in1 を「0」にする
105 :                     else if([                        ){in1=0;}
106 :                     fprintf(fp, "%d", [        ]);
107 :                     fclose(fp);
108 :                     usleep(200);
109 :
110 :                     printf("In2 を入力してください．(1 or 0 で回転方向指定 , 5 でプログラム終了)¥n");
111 :                     scanf("[        ", [        ]);          //キーボードから，in2 へ数値代入
112 :                     printf("in2= %d¥n", in2);
113 :
114 :                     sprintf(path3, "/sys/class/gpio/gpio%d/value", gpio2);
115 :                     fp=fopen(path3, "wb");
116 :
117 :                     //in2 が「5」の時，GPIO から「0」を出力してループ終了
118 :                     if([                ){
119 :                              fprintf(fp, "%d", 0);
120 :                              fclose(fp);
```

```
121 :                                   ;
122 :                       }
123 :                       //in2 が「0」,「1」,「5」以外の時は，in2 を「0」にする
124 :                       else if(                          ){in2=0;}
125 :                       fprintf(fp, "%d",          );
126 :                       fclose(fp);
127 :                       usleep(200);
128 :
129 :             }
130 :             printf("プログラム終了¥n");
131 :
132 :             //使用した GPIO の無効化
133 :                                   ;
134 :                                   ;
135 :
136 :             return 0;
137 :  }
```

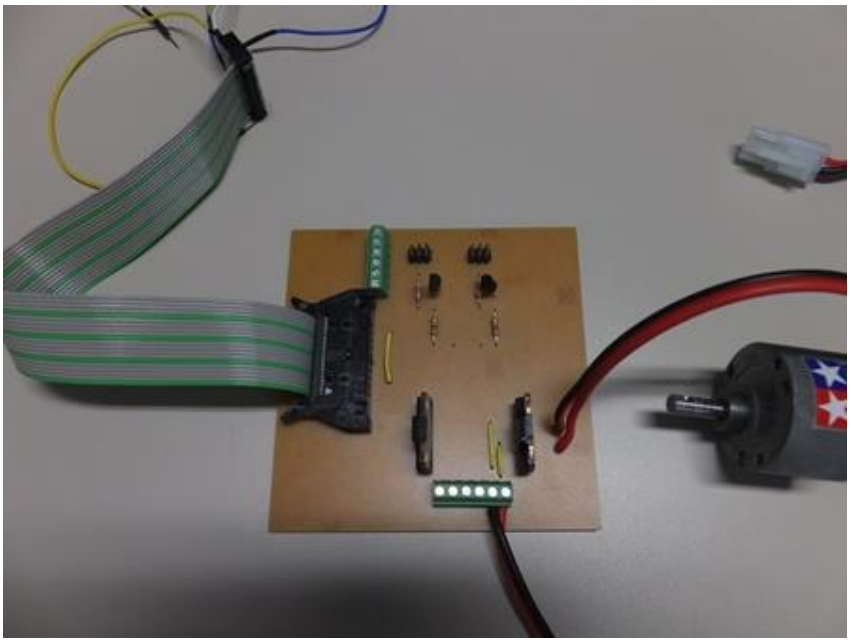

図 2.1　モータドライバ回路例

実験当日に配布

図 2.2　BBB 接続コネクタのピン配置

本実験では，ST を SYS_5V に接続，IN1 と IN2 に，それぞれ異なる GPIO のピンを接続する．ただし，**使用するボードによって，図 2.2 のピン配置と異なる場合があるため注意が必要**.

2.2 PWM 制御

PWM(pulse width modulation)制御とは，パルス幅変調制御と呼ばれるように，パルスの ON と OFF の幅を変化させることにより，モータを制御する方法である．サーボモータでは，角度制御に使用され，本実験で使用する DC モータにおいては，速度制御で使用される．PWM 制御は，ON と OFF の時間を制御することで，DC モータの速度制御する．ON の時間が長いと速度は速くなり，ON の時間が短いと速度は遅くなる．ON と OFF の時間が同じの場合には，速度が常に ON 状態の半分になる．

PWM 制御において，生成されるパルス電圧は一定の周期 T で出力される．周期の中で，ON 状態にある時間を T_{ON} と呼び，1 周期における ON 状態と OFF 状態の比率をデューティー比 D($D=T_{ON}/T$)と呼ばれる．BBB では，周期 T を「period」，ON 状態にある時間 T_{ON} を「duty」と定義している．

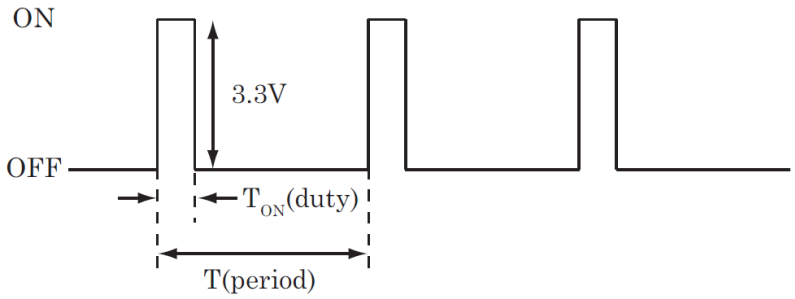

図 2.3　PWM 制御におけるパルス出力

2.3 BBB における PWM 制御

第 1 週目のテキストの図 2.2 における「GPIO_番号(PWM)」と書かれたピンが，PWM 制御に使用することが可能なピンである．ただし，「GPIO_50(PWM)」と「GPIO_51(PWM)」，「GPIO_2(PWM)」と「GPIO_3(PWM)」，は同一系統のピンとなっているため，それぞれ片方のみしか PWM 制御には使用することができない．また，先に PWM 制御と使用すると有効化したピンが優先される．そのため，P9 において，PWM 制御に使用可能なピンは，3 個である．

PWM 制御を使用するためには，次の手順でピンの設定を行う．

(1) 使用するピンを決定する．

(2) PWM の有効化を行う．

「/sys/devices/bone_capemgr.●/slots」に「am33xx_pwm」を書き込む．

＊「bone_capemgr.●」の●には，ボードにより決められた数字が入る．

⇒「/sys/devices/」のディレクトリのデータを参照することで確認可能．

(3) 使用するピンの設定「P■_×」

決定したピン番号を「P9_ピン番号」として記録する．例えば，P9 の 14 番ピンであれば「P9_14」．

「/sys/devices/bone_capemgr.●/slots」に「bone_pwm_P9_14」を書き込む．

書き込む場所は，(2)と同じである．

(4) (3)で書き込みを行うと，「/sys/devices/ocp.▲/pwm_test_P9_14.◎」というディレクトリが作成される．「ocp.▲」の「▲」にもボードで決められた数が入るため，「ls コマンド」で確認すること．また，

「pwm_test_P9_14.◎」の「◎」にも番号が入る．こちらは，<u>有効化する順番で変わるため，毎回確認が必要</u>である．基本的に，1 つのみであれば，BBB の電源 ON 状態では変わらない．

(5) PWM のパルス周期(period)[ns]の設定

「/sys/devices/ocp.▲/pwm_test_P9_14.◎/period」に「周期[ns]」を書き込む．

例えば，10[ms]の場合，周期を「10000000」とする．

(6) PWM のパルスの ON 時間(duty)[ns]の設定

「/sys/devices/ocp.▲/pwm_test_P9_14.◎/duty」に「ON 時間[ns]」を書き込む．

(7) パルスの極性(polarity)の設定

「/sys/devices/ocp.▲/pwm_test_P9_14.◎/polarity」に「極性（今回は「0」）」を書き込む．

(8) PWM 出力の開始

「/sys/devices/ocp.▲/pwm_test_P9_14.◎/run」に「1」を書き込む．

逆に停止する場合には，「0」を書き込む．

(9) PWM 条件の変更(DC モータの速度変更)

「/sys/devices/ocp.▲/pwm_test_P9_14.◎/duty」に「変更した ON 時間[ns]」を書き込む．

PWM 出力中には，「duty」のみ変更可能．

2.4 DC モータ駆動実験 2

実験 1 では，モータを最高速度で回転させていた．この実験では，PWM 制御を用いることで，モータの回転速度を制御しなさい．モータドライバへは，「IN1」に PWM 制御の信号，「IN2」に GPIO からの出力(本実験では「0」)を入力することで，DC モータの速度制御を実現しなさい．プログラムは，記載してあるプログラムの空欄を埋めて完成させなさい．そして，DC モータが回転するギリギリの「duty」を「duty」を様々に変更することで求めなさい．

```
 1 : //DC モータの PWM 制御
 2 : #include <stdio.h>
 3 : #include <unistd.h>
 4 : #include <stdint.h>
 5 : #include <stdlib.h>
 6 : #include <string.h>
 7 : #include <dirent.h>
 8 : #include <fcntl.h>
 9 : #include <sys/mman.h>
10 : #include <poll.h>
11 :
12 : #define PIN_PWM "            "  //P■_×
13 :
14 :
15 : //gpio の有効化関数
16 : void gpio_export(int n){
17 :         int fd;
18 :         char buf[40];
19 :
20 :         sprintf(buf, "%d", n);
21 :
```

```
22 :            fd = open("/sys/class/gpio/export", O_WRONLY);
23 :            write(fd, buf, strlen(buf));
24 :            close(fd);
25 :    }
26 :
27 :    //gpio の設定ファイルを開く関数
28 :    int gpio_open(int n, char *file, int flag){
29 :            int fd;
30 :            char buf[40];
31 :
32 :            sprintf(buf, "/sys/class/gpio/gpio%d/%s", n, file);
33 :
34 :            fd = open(buf, flag);
35 :            return fd;
36 :    }
37 :
38 :    int main(void)
39 :    {
40 :            int i, fdd;
41 :            FILE *fp, *fp1, *fp2;
42 :            int gpio2=            ;       //IN2 への GPIO 番号
43 :            int pwm_no=           ;       //pwm 番号,「P■_×．◎」の◎に該当する番号
44 :            int ocp_num=          ;       //ocp.▲の▲に該当する番号
45 :            char path[60],path2[60], path3[60];/*デバイスファイルのパス*/
46 :            int in1=0, in2=0;
47 :            int period, duty;
48 :
49 :
50 :            //gpio2 の有効化
51 :            gpio_export(gpio2);
52 :
53 :            //gpio2 を出力(out)に設定
54 :            fdd=gpio_open(gpio2, "direction", O_WRONLY);
55 :            write(fdd, "out", 3);
56 :            close(fdd);
57 :
58 :            //gpio2 から「0」を出力
59 :            sprintf(path3, "/sys/class/gpio/gpio%d/           ", gpio2);
60 :            fp = fopen(path3, "w");
61 :            fprintf(fp, "%d",    );
62 :            fclose(fp);
63 :
64 :            /*PWM 機能の有効化*/
65 :            fp = fopen("/sys/devices/bone_capemgr.  /slots","w");
66 :            fprintf(fp, "                              ");
67 :            fclose(fp);
68 :
69 :            /*ピンの設定（PIN_PWM 指定のピン）*/
70 :            sprintf(path, "bone_pwm_%s",PIN_PWM);
71 :            fp = fopen("/sys/devices/bone_capemgr.  /slots", "w");
72 :            fprintf(fp, path);
73 :            fclose(fp);
74 :
75 :            /*安全のため，PWM 出力の停止*/
76 :            sprintf(path, "/sys/devices/ocp.  /pwm_test_           /run", ocp_num, PIN_PWM, pwm_no);
77 :            fp = fopen(path, "wb");
78 :            fprintf(fp, "%d", 0);
```

```
 79 :            fclose(fp);
 80 :            /*PWM 周期の設定*/
 81 :            period = 10000000;
 82 :            sprintf(path, "/sys/devices/ocp.  /pwm_test_           /period", ocp_num, PIN_PWM, pwm_no);
 83 :            fp = fopen(path, "wb");
 84 :            fprintf(fp, "%d", period);
 85 :            fclose(fp);
 86 :
 87 :            /*PWM 極性の設定*/
 88 :            sprintf(path, "/sys/devices/ocp.  /pwm_test_           /polarity", ocp_num,
 89 :                     PIN_PWM, pwm_no);
 90 :            fp=fopen(path, "wb");
 91 :            fprintf(fp, "%d", 0);
 92 :            fclose(fp);
 93 :
 94 :            /*PWM  ON 状態時間の初期化*/
 95 :            sprintf(path, "/sys/devices/ocp.  /pwm_test_           /duty", ocp_num, PIN_PWM, pwm_no);
 96 :            fp=fopen(path, "wb");
 97 :            fprintf(fp, "%d", 0);
 98 :            fclose(fp);
 99 :
100 :            /*PWM 出力の開始*/
101 :            sprintf(path, "/sys/devices/ocp.  /pwm_test_           /       ", ocp_num,
102 :                     PIN_PWM, pwm_no);
103 :            fp=fopen(path, "wb");
104 :            fprintf(fp, "%d",    );
105 :            fclose(fp);
106 :
107 :            sleep(1);
108 :
109 :            //モータ回転の無限ループ（「-1」を入力で終了）
110 :            while(1){
111 :                     printf("duty を入力してください．(period=%d)¥n", period);
112 :                     scanf("%d", &              );
113 :                     printf("duty = %d¥n",              );
114 :
115 :                     if(duty==-1){break;}
116 :
117 :                     //入力した duty で PWM 信号を出力
118 :                     sprintf(path, "/sys/devices/ocp.  /pwm_test_           /duty", ocp_num,
119 :                              PIN_PWM, pwm_no);
120 :                     fp=fopen(path, "wb");
121 :                     fprintf(fp, "%d", duty);
122 :                     fclose(fp);
123 :                     usleep(200);
124 :
125 :            }
126 :            printf("プログラム終了¥n");
127 :
128 :            //GPIO を out にして，無効化するとそのピンから電圧が出力されるため，
129 :            //GPIO は無効化しない．⇒無効化するとプログラム終了後にモータが回転．
130 :
131 :            /*PWM  ON 状態時間の初期化*/
132 :            sprintf(path, "/sys/devices/ocp.  /pwm_test_           /duty", ocp_num, PIN_PWM, pwm_no);
133 :            fp=fopen(path, "wb");
134 :            fprintf(fp, "%d", 0);
135 :            fclose(fp);
```

```
136 :            /*PWM 出力の停止*/
137 :            sprintf(path, "/sys/devices/ocp.□/pwm_test_□□□□□□□□□/run", ocp_num, PIN_PWM, pwm_no);
138 :            fp=fopen(path, "wb");
139 :            fprintf(fp, "%d",□□□);
140 :            fclose(fp);
141 :
142 :            sleep(1);
143 :
144 :            return 0;
145 :  }
```

3　応用実験

第 1 週目の最後に作成した超音波センサから距離を求めるプログラムと今回の実験 2 のプログラムを組み合わせることで，超音波センサから得られた距離に応じて，モータの回転速度が変化するプログラムを作成し，DC モータを制御しなさい．また，プログラムをキーボードからあるキーを入力した際に終了するように，次の関数とプログラムを追加しなさい．

「距離 50[mm]未満：モータ停止」

「距離 50～500[mm]：停止から最高速まで線形的に増加」

「距離 500[mm]を超える：最高速で回転」

＊超音波センサのプログラムに DC モータのプログラムを追加する方がプログラミングしやすい．

＜追加関数：キー入力判断関数＞　＊時間がない場合は追加不要

```
 1 :  //キー入力判断関数，「termios.h」を追加でインクルード
 2 :  int kbhit(void)
 3 :  {
 4 :      struct termios oldt, newt;
 5 :      int ch;
 6 :      int oldf;
 7 :
 8 :      tcgetattr(STDIN_FILENO, &oldt);
 9 :      newt = oldt;
10 :      newt.c_lflag &= ~(ICANON | ECHO);
11 :      tcsetattr(STDIN_FILENO, TCSANOW, &newt);
12 :      oldf = fcntl(STDIN_FILENO, F_GETFL, 0);
13 :      fcntl(STDIN_FILENO, F_SETFL, oldf | O_NONBLOCK);
14 :
15 :      ch = getchar();
16 :
17 :      tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
18 :      fcntl(STDIN_FILENO, F_SETFL, oldf);
19 :
20 :      if (ch != EOF) {
21 :          ungetc(ch, stdin);
22 :          return 1;
23 :      }
24 :
25 :      return 0;
26 :  }
```

＜キー入力判断関数を用いた無限ループの脱出＞　＊時間がない場合は追加不要

```
1：  //キーボードの「q」を押すとプログラム終了
2：  if (kbhit()) {
3：            if(getchar()=='q') //「q」で終了
4：                      break;
5：  }
```

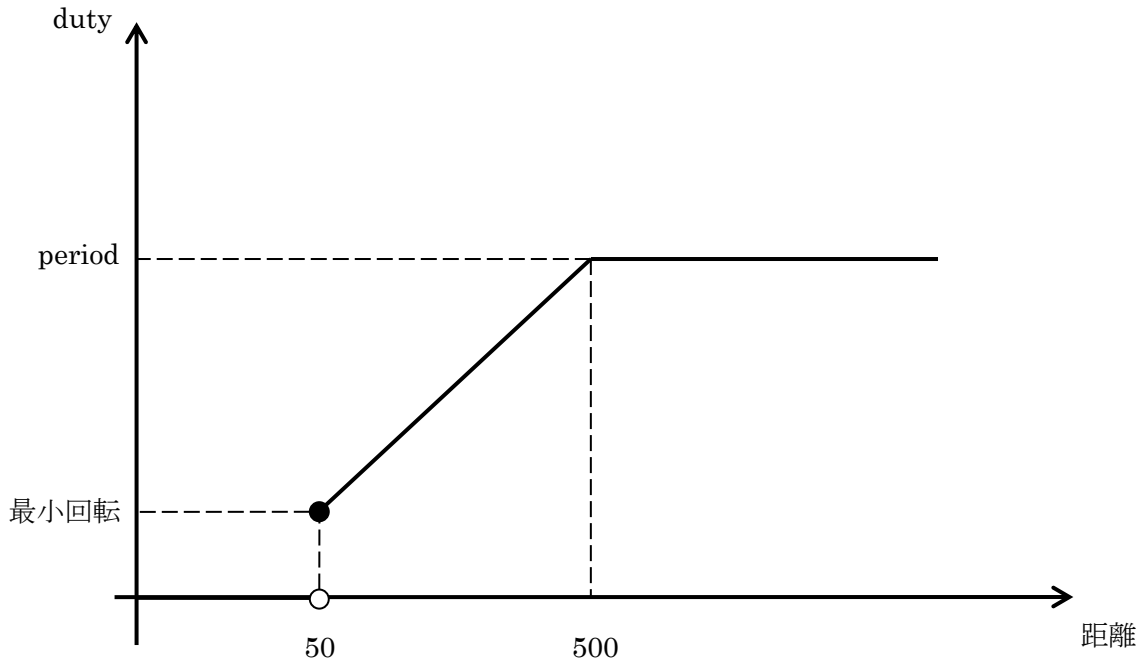

図 3.1　duty の出力イメージ

4　レポート課題

（課題 1）

PWM 制御の実験では，一方向のみにモータを回転させたが，実際に使用する際には，正転と逆転を行わせる必要がる．そこで，2 個の GPIO と 1 個の PWM 制御信号を用いて，どのようにすれば，正転と逆転を実現できるかを考え，実現方法を説明しなさい．分からない場合，表 1.1 を参考にしなさい．

（課題 2）

世の中に存在している自律移動ロボットが壁にぶつからずに動作している仕組みを使用されているセンサや制御の面から調べて説明しなさい．

***調べた際の参考文献は必ず記載すること．**

5　レポートについて

実験終了後 1 週間以内に提出して下さい．作成したプログラムソースと，実行結果の出力画面のキャプチャは必ず載せて下さい．参考文献は，インターネットを用いて調べた場合でも必ず記載して下さい．