BeagleBone Black を用いた入出力実験 I

第 1 週目：超音波センサを用いた入出力実験

第 2 週目：超音波センサと DC モータを用いた制御実験

（レポート提出の注意点）
ペーパーレスを推進するために，レポートの提出は電子データとします.
提出方法は，Classroom を使用して提出して下さい.
クラスコード：lrqumzz
提出する電子ファイルは，必ず PDF（定められた表紙も付けること）として下さい.
ファイル名は「工学実験レポート_2 ケタの出席番号_氏名.pdf」として下さい.
スキル評価シートも必ず電子データで提出して下さい.

担当：以後直樹

＜第 1 週目：超音波センサを用いた入出力実験＞

1　実験目的

BeagleBone Black と超音波センサを組み合わせて，センサと対象物の距離を測定することが可能となることを目的とする．

2　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

2.1 BeagleBone Black について

本実験で使用する BeagleBone Black の写真が図 2.1 である．

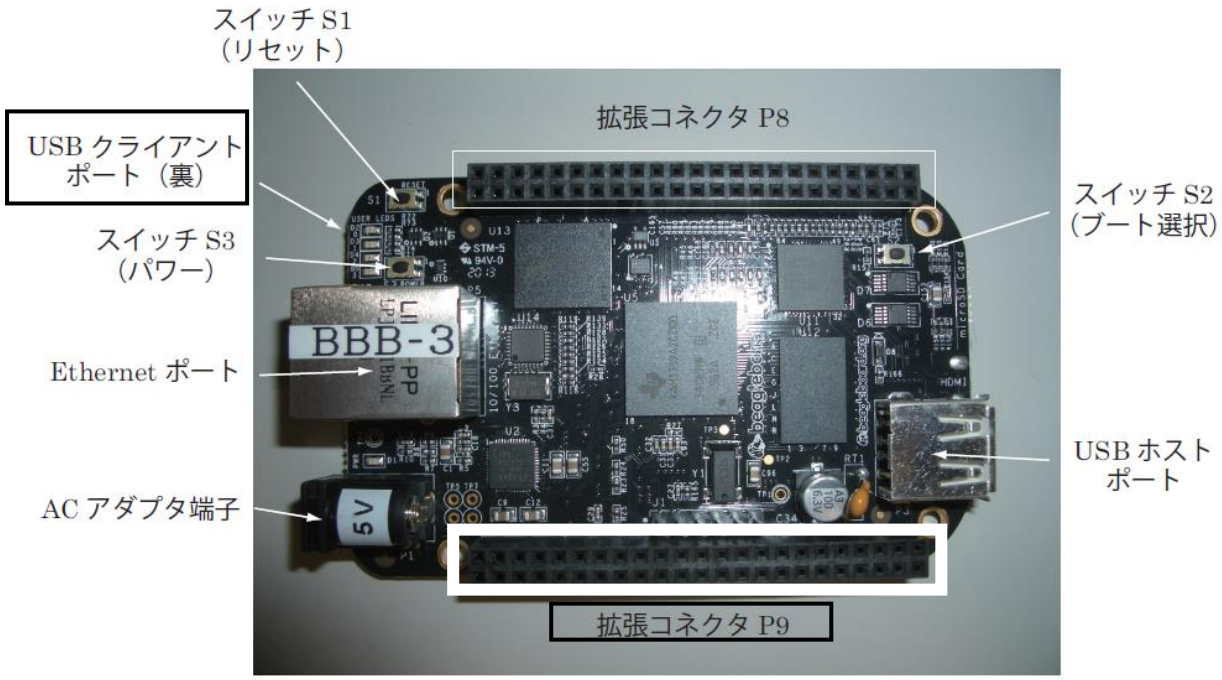

図 2.1　BeagleBone Black

2.2 BeagleBone Black の取り扱い注意点

次の注意点を守らない場合，BBB が故障する可能性が非常に高い．1 台 6000 円以上し，予備機がほとんどないため，故障させないように取り扱いには，注意すること．台数が足りなくなり，今後の実験や，後期の創造工学に影響を及ぼす可能性が大いにあり．最悪，弁償して貰う可能性もあり．

(1) I/O 端子に外部から電圧を印加したまま，BBB 本体の電源を OFF にしないこと．
(2) AIN 端子には，1.8[V]以上の電圧を入力しないこと．入力すること故障する．
(3) 出力端子として設定した GPIO 端子に電圧を印加しないこと．印加すると 1 発で故障する．

2.3 拡張コネクタ P9 のピン配置

拡張コネクタ P9 のピン配置を図 2.2 に示す．ここで，「GND」はグランド端子，「DC 3.3V」は 3.3[V]電源(最大電流 250[mA])，「SYS 5V」は 5[V]電源(最大電流 250[mA])，「GPIO～」は汎用入出力端子，「VADC」は A/D 変換用参照電圧，「AGND」は A/D 変換用 GND，「AIN～」はアナログ入力端子(最大入力電圧 1.8[V])である．　**＊GPIO_48 は，調子が悪いので使用しない方がいい．**

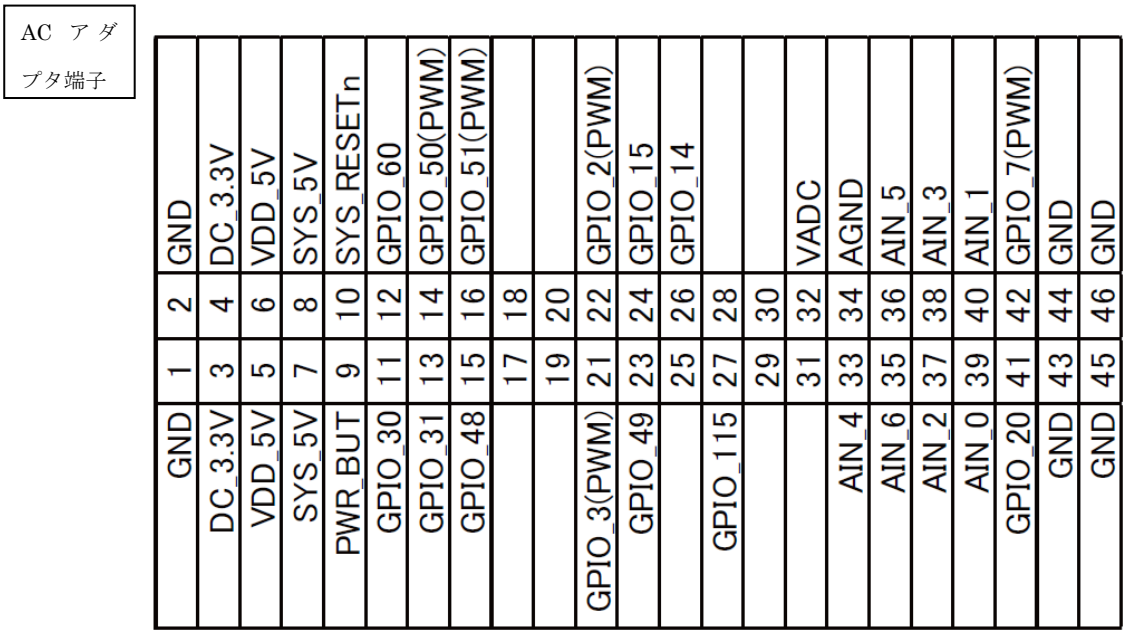

図 2.2　拡張コネクタ P9 のピン配置

2.4 BeagleBone Black の A/D 変換

コンピュータで扱うデータは，デジタルである．しかし，センサが出力する電気信号の大多数は，アナログの電圧である．そのため，センサからの出力をそのままコンピュータで扱うことはできない．コンピュータで取り扱うためには，アナログ値をデジタル値に変換する必要がある．この変換は，A/D 変換と呼ばれ，多くの電子デバイスで使用されている．

3　PC から BBB への接続の基本

3.1 Tera Term を用いた ssh 接続

ssh 接続を用いて，PC から BBB へ接続することが可能である．次の手順で，BBB へと接続する．

(1) Tera Term を起動する．起動が終わると，図 3.1 のような接続設定画面が立ち上がる．ホストを「192.168.7.2」，TCP ポートを「22」とする．その後，OK をクリックする．

図 3.1　Tera Term による接続設定画面

(2) (1)が成功すると，図 3.2 のような SSH 認証画面が起動する．ユーザ名を「root」として，OK をクリックする．

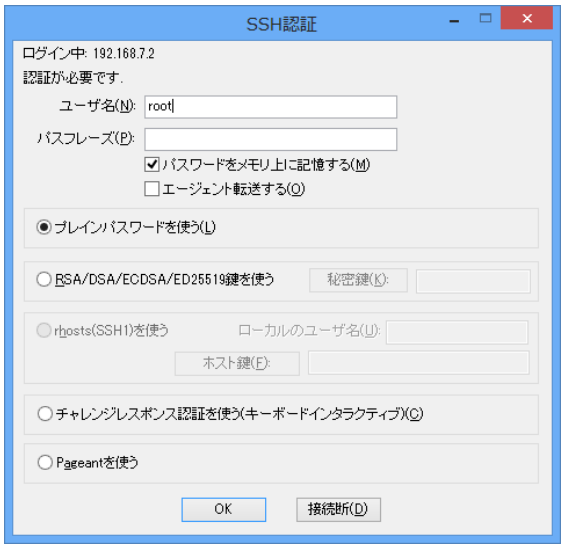

図 3.2　SSH 認証画面

(3) 認証が成功すると，図 3.3 のような画面が起動し，接続が完了する．その後は，Linux のコマンドを使用することが可能となる．

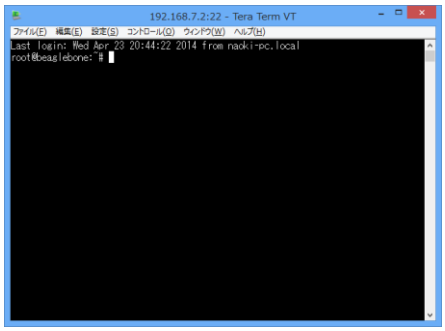

図 3.3　ssh 接続完了画面

3.2 Windows 上のファイルを BBB への転送

本実験では，プログラムのソースを Windows 上の TeraPad 等のテキストエディタを使用して作成する．ただし，BBB 内の vi を使用してもプログラムのソース作成は可能である．テキストエディタを持ち手作成したデータは，BBB へ転送する必要がある．Tera Term で ssh 接続していると，簡単にファイルの転送が行える．Windows 上で作成したファイルを Tera Term 上にドラックすれば転送ができる．ファイルを Tera Term 上に，ドラックすると図 3.4 のようなダイアログが出現する．このダイアログで「SCP」を選択し，「OK」をクリックするとファイルが転送できる．転送するディレクトリを指定する場合には，「送信先」に送り先ディレクトリの絶対アドレス（フルパス）を記入すれば，そのディレクトリに保存できる．ただし，

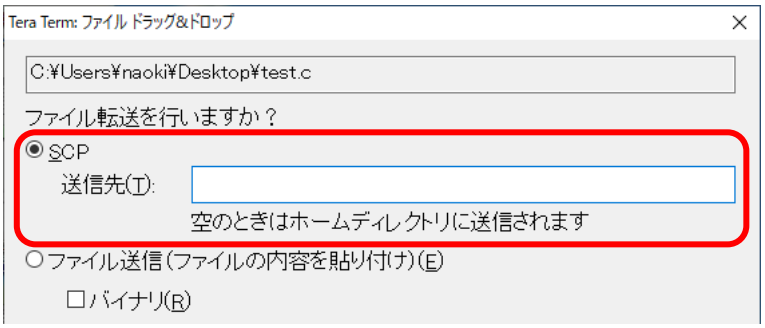

図 3.4　ファイル転送確認ダイアログ

4　BBB を用いた C 言語の基本実験

4.1 BBB を用いた C 言語の基本実験 1

ディレクトリ名「test_西暦実験日」（実験日が 2015 年 4 月 15 日場合 : test_20150415）のディレクトリを作成しなさい．また，「Hello World」と表示させるプログラムを作成し，そのソースコードを作成したディレクトリに転送しなさい．

（ディレクトリの作成）

mkdir name

4.2 BBB を用いた C 言語の基本実験 2

実験 1 のソースコードをコンパイルし，実行可能形式のファイルを作成しなさい．その後，実行可能形式のファイルを実行し，「Hello World」と表示させることを確認しなさい．実行時の画面は，レポートに使用するので，キャプチャしておくこと．

（コンパイル）

gcc filename.c　⇒　gcc filename.c -o outputname

4.3 BBB を用いた C 言語の基本実験 3

実験 1 で作成したディレクトリとその中身のデータを削除しなさい．削除前後のファイルリストを表示して，存在してディレクトリが削除されていること確認すること．この際も，画面をキャプチャしておくこと．

（必要と思われるコマンドオプション）

-r　もしくは　-rf

5　超音波センサを用いた距離測定実験

超音波センサを用いた実験では，図 5.1 のような超音波センサと図 5.2 のような接続回路を使用して，実験を行う．

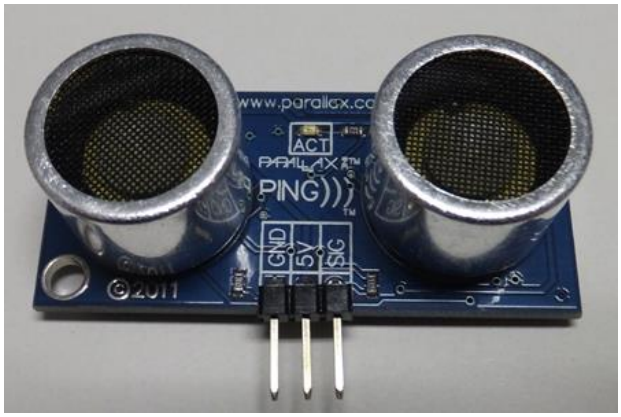

図 5.1　超音波センサ

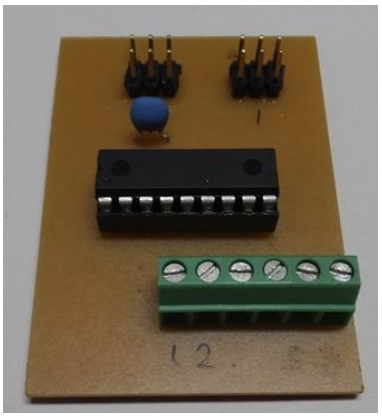

図 5.2　超音波センサ回路

5.1 使用する超音波センサについて

本実験で使用する超音波センサは，3[cm]~3.3[m]まで測定可能である．また，図 5.3 のような場合は，測定することができない．超音波センサと超音波センサ回路の接続イメージを図 5.4 に示す．回路は，2 個の超音波センサを同時に使用することができる．実験で使用するのは，1 個のみなので，1 番と 2 番の好きな方を使用すること．使用した番号の SIG と対応する出力端子を BBB と接続すること．

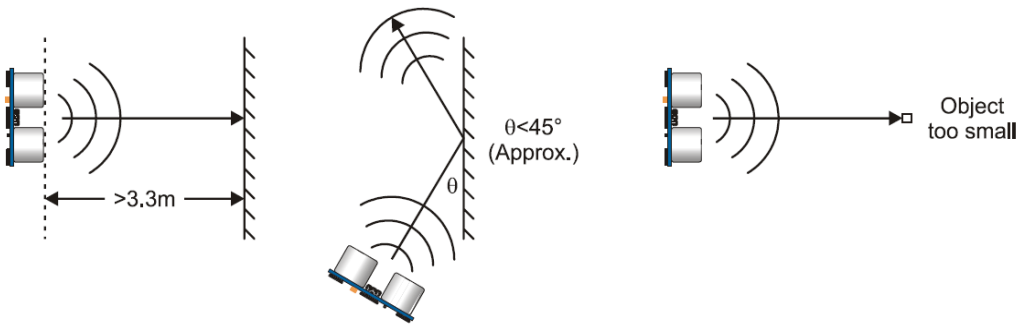

図 5.3　測定できない例

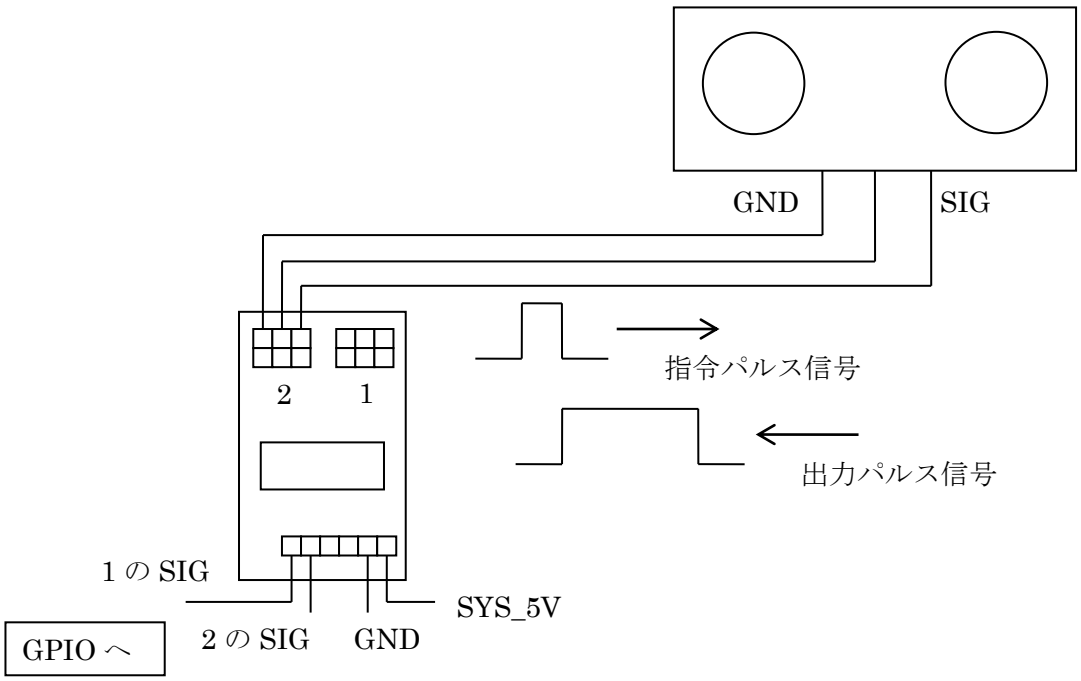

図 5.4　超音波センサ接続イメージ図

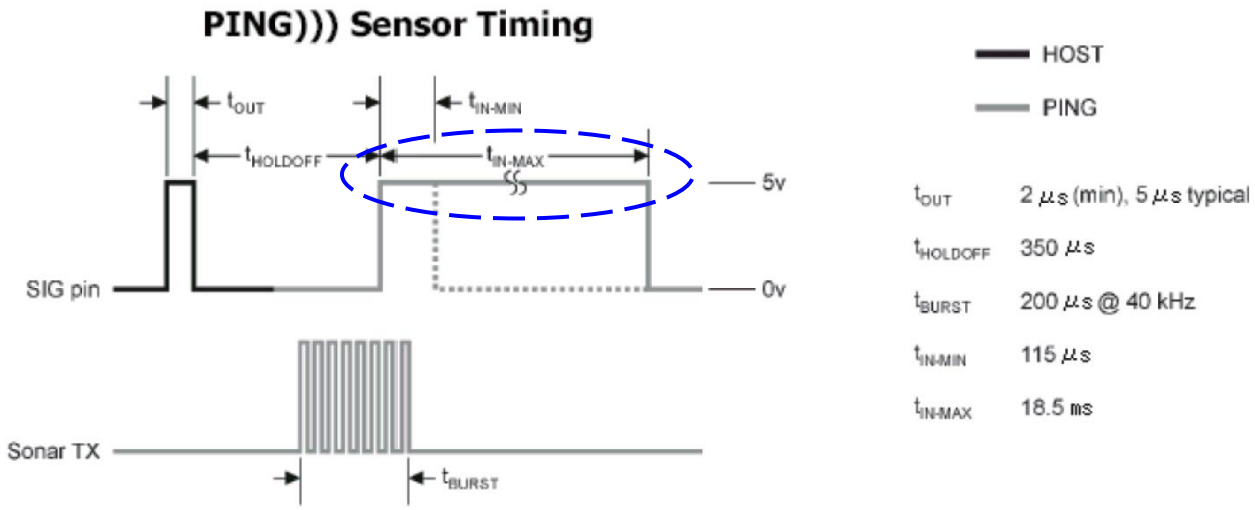

図 5.5　超音波センサの動作信号

図 5.5 に本実験で使用する超音波センサの動作信号の例を示す．基本的に，図 5.5 において破線の楕円で囲んだ箇所が，センサと対象物との距離に応じて，パルスの幅がほぼ線形的に変化する．つまり，対象物との距離が近いとパルスの幅は狭くなり，距離が遠くなるとパルスの幅は広くなる．そのため，パルス幅と対象物との関係性を求めることができると，パルス幅から距離を求めることができる．

5.2 GPIO の使用方法

BBB には，1 ビットの出力もしくは入力が可能な GPIO が豊富に搭載されている．本実験では，図 2.2

に示したように，拡張ポート P9 を使用する．図 2.2 で「GPIO_数字」と記載されている箇所が使用可能な GPIO である．この数字を GPIO 番号として使用する．GPIO を使用するためには，各ピンに，設定を行う必要がある．設定の手順は，「ピンの決定」⇒「有効化」⇒「初期化と入力 or 出力の設定」⇒「入出力」⇒「無効化」である．

(1) 図 2.2 を参考に，使用する GPIO を決定する．
(2) (1)で決めた GPIO を有効化する．使用する GPIO 番号を特殊ファイルに書き込む．
　例：GPIO_45 を使用⇒「/sys/class/gpio/export」に 45 を書き込む．
　　コマンド入力する場合：echo 45 > /sys/class/gpio/export
(3) GPIO を入力とするのか出力とするのかを決定（direction）および，edge を決定する．
　例：(2)で有効化した GPIO を入力とする場合⇒「/sys/class/gpio/gpio45/direction」に「in」を書き込む．
(4) GPIO への書き込みもしくは，読み出しを行う．
　例：「/sys/class/gpio/gpio45/value」に「1」を書き込む．
(5) GPIO の無効化をする．
　例：(2)で有効化した GPIO を無効化する場合⇒「/sys/class/gpio/unexport」に GPIO 番号の「45」を書き込む．

5.3 超音波センサを用いた距離測定実験 1

次のプログラムソースの空欄を埋めると，超音波センサから物体までの超音波の往復時間([μs])を取得することができる．プログラムを完成させて，動作を確認しなさい．ただし，コンパイルする際にはリアルタイム拡張をする必要があり，「gcc filename.c -lrt」のように「-lrt」というオプションを付ける必要があり注意すること．また，BBB と超音波センサを接続する際は，BBB と PC を接続している USB ケーブルを抜き，**BBB の電源が OFF の状態で接続すること**．BBB の電源を OFF にする場合は，「**poweroff**」とコマンド入力すること．

＜プログラムソース＞

```
 1：   #include <unistd.h>
 2：   #include <stdio.h>
 3：   #include <stdint.h>
 4：   #include <stdlib.h>
 5：   #include <string.h>
 6：   #include <dirent.h>
 7：   #include <fcntl.h>
 8：   #include <sys/mman.h>
 9：   #include <poll.h>
10：   #include <time.h>
11：
12：   //gpio の有効化関数
13：   void gpio_export(int n){
14：            int fd;
15：            char buf[40];
```

```
16 :
17 :            sprintf(buf, "%d", n);
18 :
19 :            fd = open("/sys/class/gpio/          ", O_WRONLY);
20 :            write(fd, buf, strlen(buf));
21 :            close(fd);
22 :    }
23 :
24 :    //gpio の有効化解除の関数
25 :    void gpio_unexport(int n){
26 :            int fd;
27 :            char buf[40];
28 :
29 :            sprintf(buf, "%d", n);
30 :
31 :            fd = open("/sys/class/gpio/          ", O_WRONLY);
32 :            write(fd, buf, strlen(buf));
33 :            close(fd);
34 :    }
35 :
36 :    //gpio の設定ファイルを開く関数
37 :    int gpio_open(int n, char *file, int flag){
38 :            int fd;
39 :            char buf[40];
40 :
41 :            sprintf(buf, "/sys/class/gpio/gpio          ", n, file);
42 :
43 :            fd = open(buf, flag);
44 :            return fd;
45 :    }
46 :
47 :    //パルスの ON 時間を時刻から算出する関数[マイクロ秒]：[sec]⇒[マイクロ秒]，[nsec]⇒[マイクロ秒]
48 :    //「(n.tv_sec - o.tv_sec)」パルスの立下り時刻[sec]-パルスの立ち上がり時刻[sec]
49 :    //「(n.tv_nsec - o.tv_nsec)」パルスの立下り時刻[nsec]-パルスの立ち上がり時刻[nsec]
50 :    #define RC_LAP(n, o) ((n.tv_sec - o.tv_sec)*    数字    +(n.tv_nsec - o.tv_nsec)/    数字    )
51 :
52 :    int main(void)
53 :    {
54 :            int fd;
55 :            int Ion;
56 :            int gpio_number =    GPIO 番号    ;
57 :            int ret;
58 :            char c;
59 :            struct timespec origin;// signal start time
60 :            struct timespec now;// signal change time
61 :            struct pollfd pfd;
62 :
63 :            //gpio の有効化
64 :                                    ;
65 :
66 :            //gpio を入力に設定
67 :            fd = gpio_open(gpio_number, "direction", O_WRONLY);
68 :            write(fd, "    ", 2);
69 :            close(fd);
70 :
71 :            //gpio を edge に設定
72 :            fd = gpio_open(gpio_number, "edge", O_WRONLY);
```

```
 73 :             write(fd, "both", 4);
 74 :             close(fd);
 75 :
 76 :             //gpio の value ファイルを開く
 77 :             fd = gpio_open(gpio_number, "value", O_RDONLY);
 78 :
 79 :             pfd.fd = fd;                 //監視するファイルを設定
 80 :             pfd.events = POLLPRI;        //監視する通知を設定
 81 :
 82 :             while(1){          //無条件に繰り返す
 83 :                      lseek(fd, 0, SEEK_SET);                //読み取り位置を先頭に設定
 84 :                      ret = poll(&pfd, 1, -1);     //通知を監視
 85 :                      read(fd, &c, 1);                       //通知状態を読み込む
 86 :                      if(c == '1'){                //パルスの立ち上がりの時刻を取得
 87 :                               clock_gettime(CLOCK_MONOTONIC_RAW, &origin);
 88 :                      }
 89 :                      else continue;
 90 :                      lseek(fd, 0, SEEK_SET);
 91 :                      ret = poll(&pfd, 1, -1);
 92 :                      read(fd, &c, 1);
 93 :                      if(c == '0'){                //パルスの立ち下がりの時刻を取得
 94 :                               clock_gettime(CLOCK_MONOTONIC_RAW, &now);
 95 :                      }
 96 :                      else continue;
 97 :                      Ion = RC_LAP(now, origin);   //パルスの ON 時間を算出
 98 :                      printf("interval=%d[μsec]¥n",Ion);
 99 :                      usleep(500000);
100 :             }
101 :
102 :             close(fd);
103 :             //GPIO の無効化
104 :             [                        ];
105 :
106 :             return 0;
107 :    }
```

＊sprintf について

文字列を配列やポインタに格納するために使用する．printf や fprintf のように，変数の値，文字，文字列を格納する文字列に含ませることができる．

（例）

sprintf(文字列格納変数名,”数値：%d”,a);

この場合，int 型変数 a=10 が定義されているとすると，「数値：10」が，文字列格納変数名に文字列として格納される．

5.4 超音波センサを用いた距離測定実験 2

実験 1 で作成したプログラムを元に，センサからの距離 100～250[mm]の範囲において，30[mm]刻み毎の出力データを取得し，記録しなさい．そのデータを用いて，センサの信号から，センサと対象物の距離に変換する式の変換係数を求めなさい．変換係数は，音速等を用いても理論的に求めることも可能であるが，本実験では，実験的に求めることにする．

センサと対象物の距離[mm]＝変換係数×センサからのデータ[μs]

5.5 超音波センサを用いた距離測定実験 3

実験 2 から求められた変換係数を使用して，センサの信号から，センサと対象物との距離を出力するプログラムを完成させなさい．完成後，100～250[mm]の範囲において，50[mm]刻み毎の出力データを取得し，記録しなさい．

6 レポート課題

（課題 1）

超音波センサを用いた距離の測定原理を図と数式を用いて，説明しなさい．

（課題 2）

超音波センサからの値を正確に使用するためには，温度計が必要な場合がある．どうして温度計が必要なのかを自らの力で考えて，どうして温度計が必要なのかを説明しなさい．

（課題 3）

他のシングルボード PC を 1 つ調べて, BeagleBone Black との性能等の違いについて, 説明しなさい．

***調べた際の参考文献は必ず記載すること．**

7 レポートについて

レポートは，<u>毎回提出</u>です．なので，<u>実験終了後 1 週間以内</u>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．また，考察に関しては，距離測定実験の結果等を用いて書いてください．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．

提出ファイル名は，ファイル名は「工学実験レポート_2 ケタの出席番号_氏名.pdf」として下さい．