アジャイルワーク１　レポート

千葉工業大学
情報変革科学部 情報工学科

班 番 号 グループ１

学生番号 25G1026

氏　　名 大菅　海兎

1. ワークB 脈拍測定プログラム開発

1 はじめに

本ワークでは，心拍測定のためのプログラムを開発することを目的に実習を行なった．本実習では，マイコンとしてArduinoを利用し，脈波信号としてAtomLiteを利用した．また，実行結果をcsvファイルとグラフにすることで，脈拍計測システムの機能と性能の検証を行った．

2 ワークB-1

2.1 目的と実施内容

ワークB-1では， AtomLiteからの4種類の信号の中から脈波信号を出しているものを特定することが目的である．方法として，信号をArduinoを用いてAD変換し，その値で波形のグラフを描画する．波形の形を一般的な脈波波形と比較することにより，AtomLiteから出ている脈波を特定する．

2.2 脈波信号の検討

AtomLiteには4種類の信号が記録されている．この信号はAtomeLiteに付属しているLEDの色が一つずつ対応している．色は，赤，青，緑，紫がある．この4つの中の一つに脈波が記録されているため，それぞれの色の波形をグラフにすることで，脈波がどの色かを判別する．判別方法として，人間の一般的な脈拍を利用する．人間の一般的な脈拍は安静時に1分あたり60回から100回とされている[1]．また，波形は1周期の間に複数の山が出現することが知られている[2]．そのため，ワークB-1では5秒間記録を取り，グラフに表す．グラフに5回から8回程度のピークが見ることができると考えられる．よって，これを利用し，判別を行う．赤の波形は図1.1に表す．この図から，ピークが5回あることがわかる．また，1周期も複数の小さい山と大きい1つの山で構成されている．そのため，この波形は人間の脈波波形であると十分にいえると考えられる．

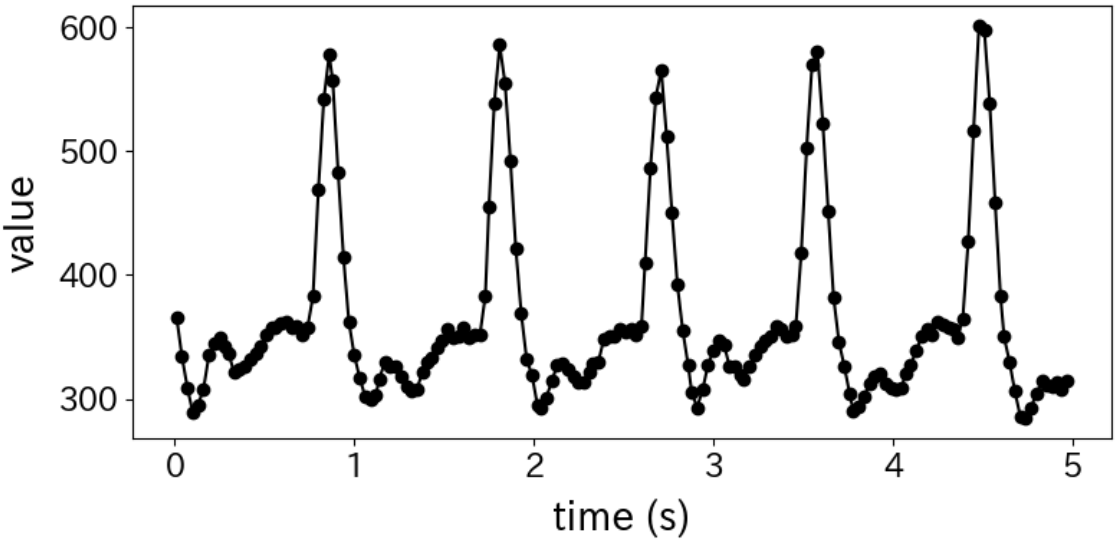

図 1.1: 赤色の波形

青の波形は図 1.2 に表す．この図から，ピークは 7 回であることがわかる．しかし，1 周期の間に複数の山が見られない．そのため，人間の脈波波形であるとは言えないと考えられる．

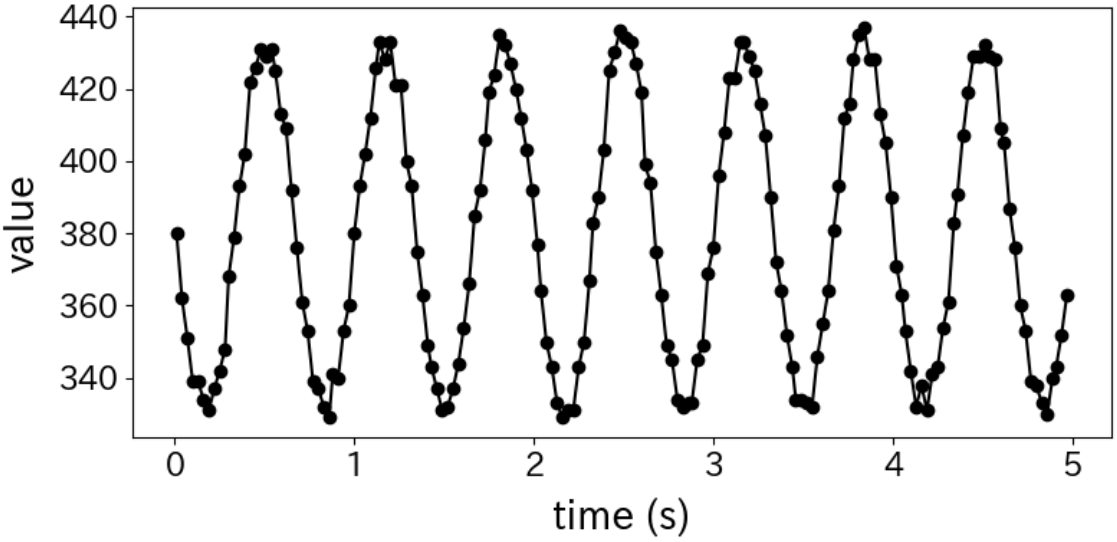

図 1.2: 青色の波形

緑の波形は図 1.3 に表す．この図から，ピークが数多く見られ，人間の脈拍にしては多いことがわかる．そのため，人間の脈波波形ではないと考えられる．

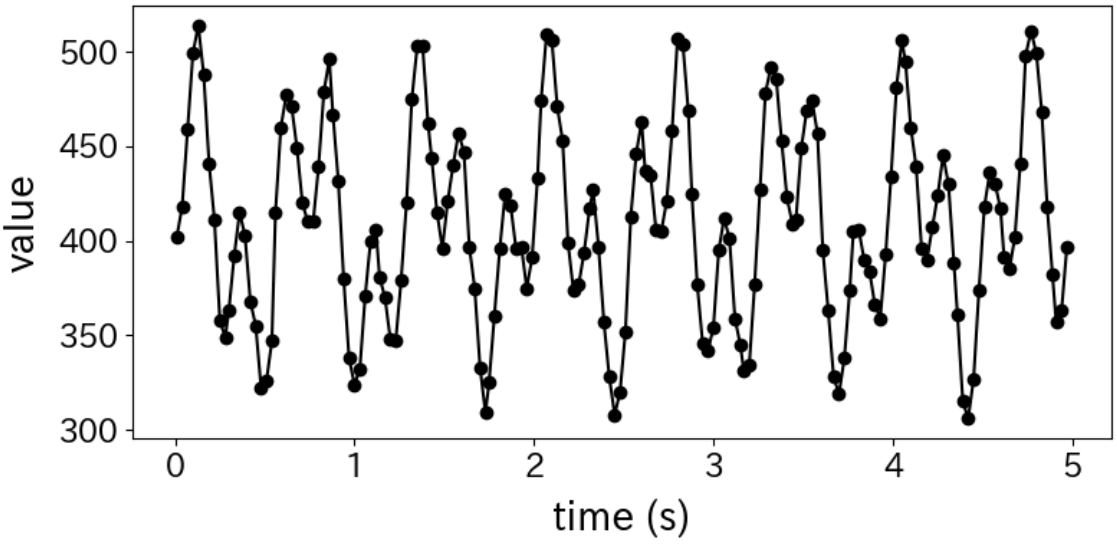

図 1.3: 緑色の波形

紫の波形は図 1.4 に表す．この図から，ピークが数多く見られ，人間の脈拍にしては多いことがわかる．そのため，人間の脈波波形ではないと考えられる．

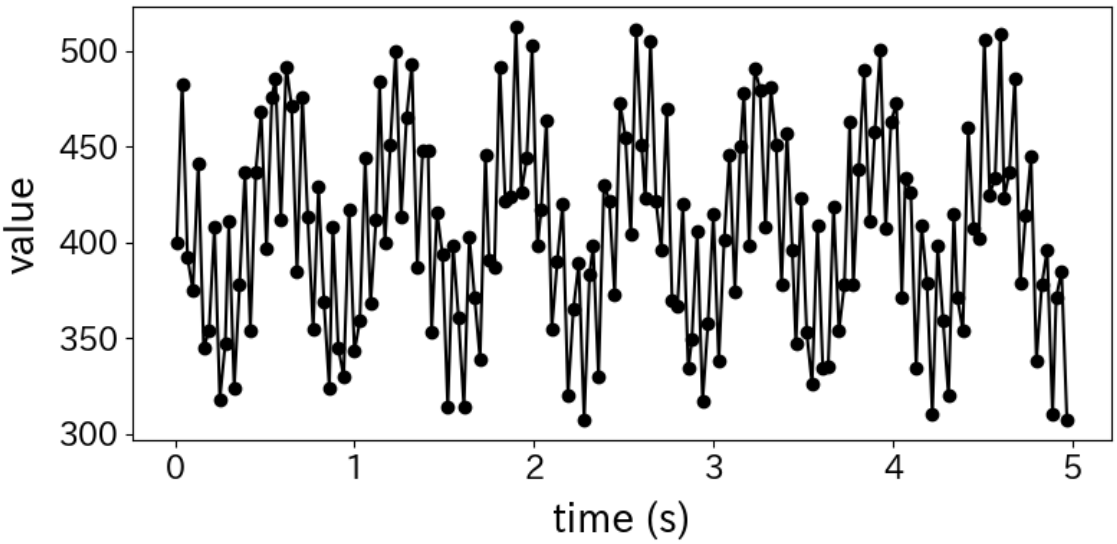

図 1.4: 紫色の波形

これら全ての波形のグラフから，赤の波形が人間の脈波であると結論づける．

2.3 実装内容

ワーク B-1 では，b1_signal_cheak.ino を実装した．プログラム内容は図 1.5 にある通りである．

3 行目では，シリアルモニターに time，value と 1 行でプログラム起動時にのみ表示するようにしている．10 から 17 行目のコードは if 文を用いて，11 から 16 行目のプログラムが 5 秒間のみ実行される．11 行目では，経過時間であるミリ秒を秒に変換している．12 から 14 行目では，秒数とアナログ値が 1 行で表示されるようになっている．

```
1  void setup() {
2    Serial.begin(9600);
3    Serial.println("time,value");
4  }
5
6  void loop() {
7    uint16_t v = analogRead(A0); // アナログ入力を読み取る
8    uint32_t t = millis(); // 起動からの時間を計測
9
10 if(t<5000){
11   float t_second=t/1000.0;
12   Serial.print(t_second); // 計測した時間をシリアル出力
13   Serial.print(",");
14   Serial.println(v); // 読み取った値をシリアル出力
15
16   delay(20); // 次のループまで20 ミリ秒待機
17 }
18 }
```

図 1.5: b1_signal_check.ino

3 ワーク B-2

3.1 目的と実施内容

ワーク B-2 では，B-1 にて特定した脈波の 1 周期の最大値を検知し，マークをつけることが目的である．

方法として，Arduino で，適当な閾値を設定することにより，Arduino が検知したピークを波形に印をつける．閾値は TOP_THRESHOLD と DOWN_THRESHOLD が用意されている．TOP_THRESHOLD は，ピーク検出プログラムを実行させる値であり，設定値を上回ったときにのみ実行される．DOWN_THRESHOLD は，ピークが存在したことを決定する値である．読み取ったアナログ値が前回値を下回った回数が DOWN_THRESHOLD に達した時に記録していた最大値をピークとする．この作業を複数回行うことにより，印がすべてのピークに適切に付けられる閾値を決定する．

3.2 閾値の検討

まず初めに，閾値は DOWN_THRESHOLD を 4 で固定し，TOP_THRESHOLD の適切な値を見つけることを行った．図 1.6 では TOP_THRESHOLD を 300 に設定したものである．

この閾値では，小さい山でもピークと判断され，印が付けられていることがわかる．そのため，TOP_THRESHOLD が 300 では足りないことがわかる．また，DOWN_THRESHOLD は全てのピークに付いているため，TOP_THRESHOLD が 300 の時は適切であることがわかる．

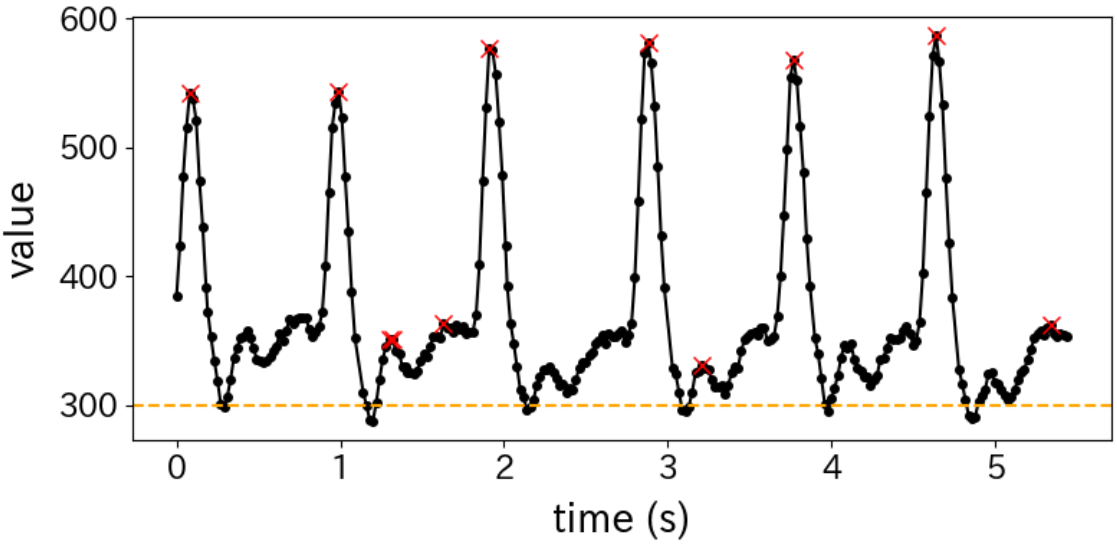

図 1.6: TOP_THRESHOLD が 300，DOWN_THRESHOLD が 4 の時の波形

図 1.7 では TOP_THRESHOLD を 350 に設定したものである．この閾値では，小さい山にもピークと判定され，印が付けられていることがわかる．そのため，TOP_THRESHOLD が 350 では足りないことがわかる．また，DOWN_THRESHOLD は全てのピークに付いているため，TOP_THRESHOLD が 350 の時は適切であることがわかる．

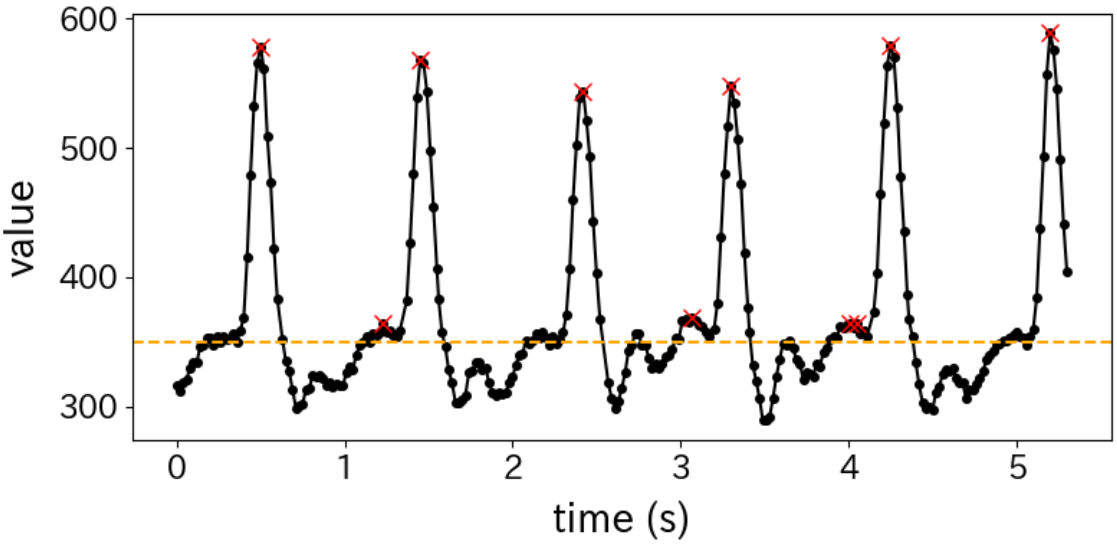

図 1.7: TOP_THRESHOLD が 350，DOWN_THRESHOLD が 4 の時の波形

図 1.8 では TOP_THRESHOLD を 400 に設定したものである．この閾値では，小さい山にピークと判定され, 印が付けられていないことがわかる．そのため, TOP_THRESHOLD は 400 が適切な値だとわかる．しかし，DOWN_THRESHOLD は全てのピークに付いていないため，TOP_THRESHOLD が 400 の時は値が大きいことがわかる．

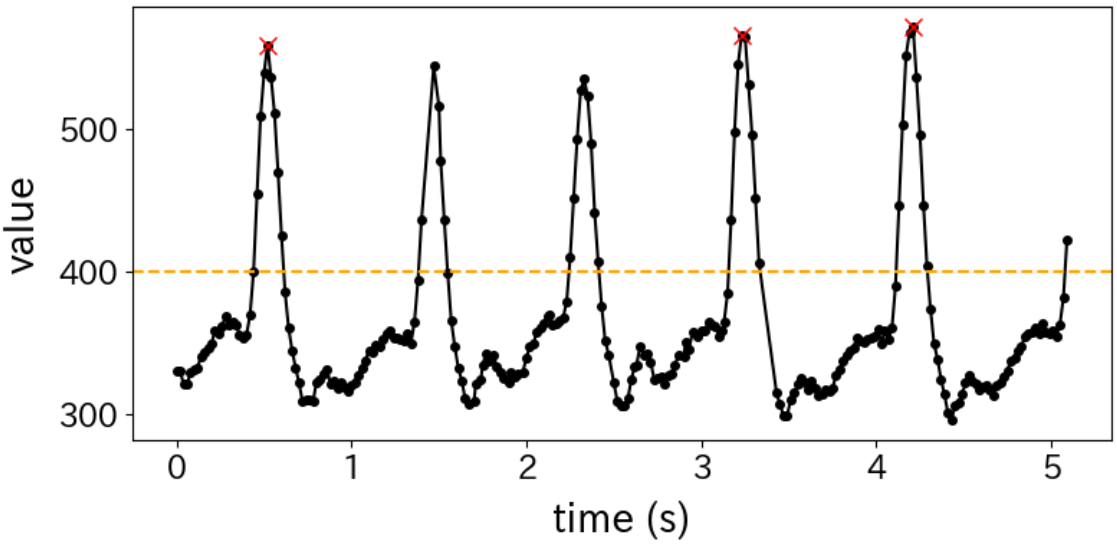

図 1.8: TOP_THRESHOLD が 400，DOWN_THRESHOLD が 4 の時の波形

TOP_THRESHOLD の適切な値が 400 と判明したため，DOWN_THRESHOLD の適切な値を設定する．

図 1.9 では DOWN_THRESHOLD を 3 に設定したものである．この閾値では，全てのピークに対してマークが付けられているため，適切のように見える．しかし，2 秒付近の山には 3 回目の下降が TOP_THRESHOLD の設定値付近であるため，ブレにより検出できない可能性がある．そのため，DOWN_THRESHOLD の値が 3 は安定性に欠けるため適切とは言えない．

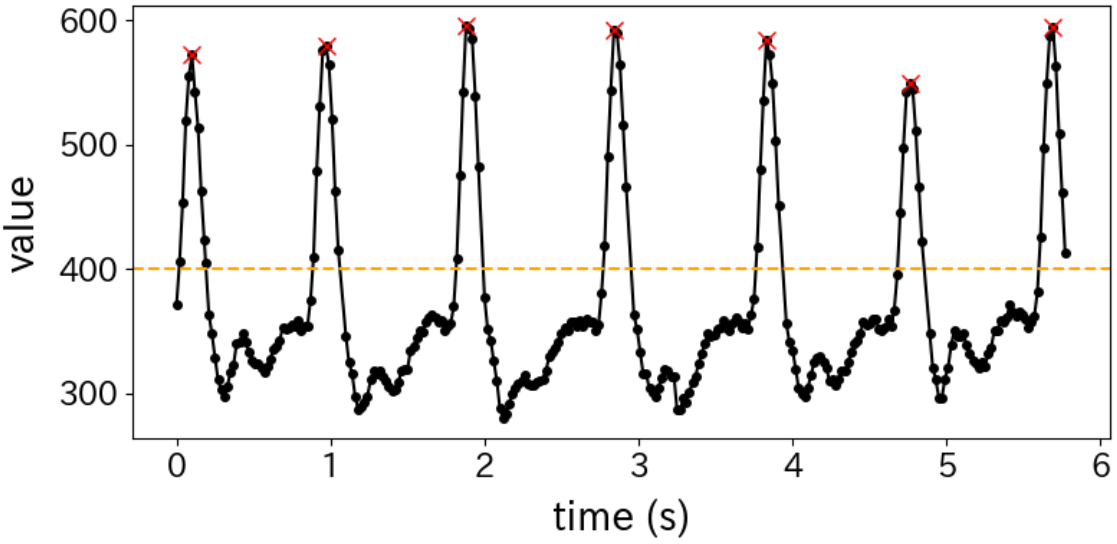

図 1.9: TOP_THRESHOLD が 400，DOWN_THRESHOLD が 3 の時の波形

図 1.10 では DOWN_THRESHOLD を 2 に設定したものである．この閾値では，全てのピークに対してマークが付けられているおり，TOP_THRESHOLD の設定値とも余裕があるため，DOWN_THRESHOLD の設定値は 2 が適切である．

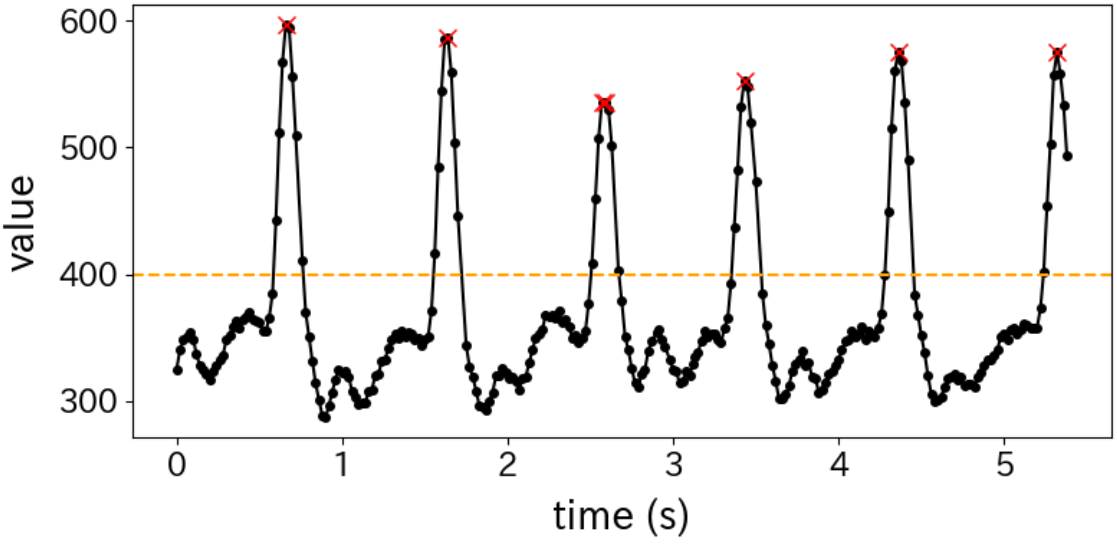

図 1.10: TOP_THRESHOLD が 400，DOWN_THRESHOLD が 2 の時の波形

これらの複数回の検討により，閾値は，TOP_THRESHOLD が 400，DOWN_THRESHOLD が 2 に設定することが適切であることが確認できた．

3.3 実装内容

ワーク B-2 では，b2_peak_detection.ino と myPulse.h と myPulse.cpp を実装した．図 1.11 は b2_peak_detection.ino を示している．このプログラムは top 関数を 5 秒間動作させるものである．5 行目では，シリアルモニターに time，value，peak をプログラム起動時に一度だけ出力している．9 行目では，起動からの時間を記録している．10 から 12 行目では，5 秒間 top 関数を呼び出し続ける．

```
1  #include "myPulse.h"
2
3  void setup() {
4    Serial.begin(115200);
5    Serial.println("time,value,peak");
6  }
7
8  void loop() {
9    uint32_t t = millis();
10   if (t < 5000) {
11     top();
12   }
13 }
```

図 1.11: b2_peak_detection.ino

図 1.12 は myPulse.h を示している．このプログラムには手を加えていないため，説明は省略する．

```
1 #ifndef MyPulse_h
2 #define MyPulse_h
3
4 #include <Arduino.h>
5
6 uint16_t top();
7
8 #endif
```

図 1.12: myPulse.h

図 1.13 と図 1.14 は myPulse.cpp を示している．17 行目では，配列の要素数を上限 300 で固定している．これは，20ms に一回記録をとるため，6 秒間記録を保持できるようになっている．18 から 20 行目では，配列を用いて，Vbox はアナログ値，Tbox は時間，Pbox はピークの判定値を記録できるようにしている．21 行目では，配列のどの場所に入れるかを保存する変数である．26 から 31 行目では，配列の保存容量を超えない限りアナログ値，時間，ピーク判定値 0 を記録する．44 から 47 行目では，配列の中身を 1 から確認し，検出したアナログ値の最大値と同じ場所をピーク判定値を 0 から 1 に変更している．48 から 52 行目では，時間，アナログ値，ピーク判定値を 1 行にしてシリアルモニターに出力している．

```
#include "myPulse.h"

# define PIN_IN A0
# define DELAY_TIME 20

# define TOP_THRESHOLD 0
# define DOWN_THRESHOLD 0
# define UP_THRESHOLD 1

uint16_t top(){
  uint32_t time = 0; // 信号の受信時間の保存用
  uint16_t maximum = 0; // 最大値の保存用
  uint16_t value = 0; // 読み込んだ信号の保存用
  uint8_t up_count = 0; // 上昇回数の保存用
  uint8_t down_count = 0; // 下降回数の保存用
  uint8_t timeout = 255; // タイムアウトまでのカウント
  const int Baffer_size=300;
  uint16_t Vbox[Baffer_size];
  uint32_t Tbox[Baffer_size];
  uint8_t Pbox[Baffer_size];
  int baffer_index=0;

  while(timeout--){
    value = analogRead(PIN_IN); // アナログ入力で値を読み取る
    time = millis(); // 起動からの時間を計測
    if(baffer_index<Baffer_size){
        Vbox[baffer_index]=value;
        Tbox[baffer_index]=time;
        Pbox[baffer_index]=0;
        baffer_index++;
      }

    if(value >= TOP_THRESHOLD){ // 読み込んだ値が山の閾値以上である場合のみピーク検
        出の処理を行う
      if(value > maximum){
        maximum = value;
        up_count++;
        down_count = 0;
      }
      else{
        down_count++;
      }
    }
```

図 1.13: myPulse.cpp

```
43      if(up_count>UP_THRESHOLD && down_count>DOWN_THRESHOLD){
44        for(int i=0;i<baffer_index;i++){
45          if(Vbox[i]==maximum){
46            Pbox[i]=1;
47          }
48      Serial.print(Tbox[i]/1000.0, 2); // 時間を秒に変換して小数点二桁までシリアル
            出力
49      Serial.print(",");
50      Serial.print(Vbox[i]); // 読み込んだ値をシリアル出力
51      Serial.print(",");
52      Serial.println(Pbox[i]);
53      }
54        return maximum;
55    }
56      delay(DELAY_TIME);
57
58    }
59    return 0;
60  }
```

図 1.14: myPulse.cpp

4 ワーク B-3

4.1 目的と実施内容

ワーク B-3 では，心拍数の算出を行うことが目的である．方法として，ワーク B-2 にてピークの判定が可能となった．そのため，ピークからピークまでの時間を利用して心拍数の計算を行う．

4.2 測定結果の考察

AtomLite からの信号を利用し，心拍数を算出した．その統計値は表 1.1 である．また，グラフは図 1.15 である．表 1.1 には，最大値，最小値，平均値，中央値，標準偏差がまとめられている．人間の心拍数は，60 から 100 回が標準値であることが知られている [1]．今回のデータでは心拍数が 60 から 69 の間に収まっている．また，標準偏差が小さいため，心拍の振れ幅も小さく，安定性が高いと言える．そのため，今回のデータは，人間の安静時の心拍数の記録として妥当である．

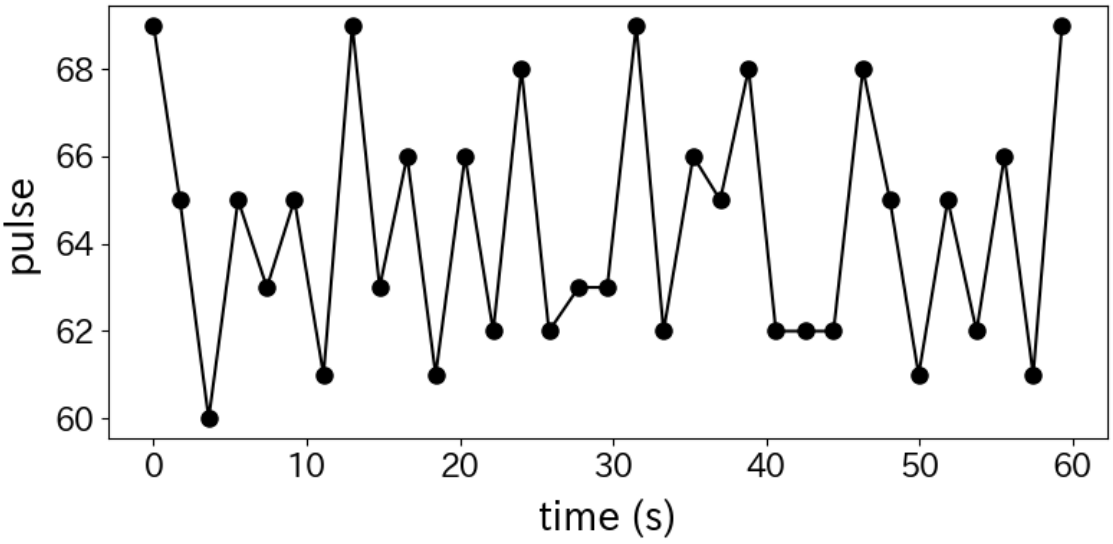

図 1.15: 計測時間と心拍数

表 1.1: 測定心拍数の統計値

統計	値
最大値	69.00
最小値	60.00
平均値	64.36
中央値	65.00
標準偏差	2.79

4.3 実装内容

ワーク B-3 では，3 つのプログラムを実装した．図 1.16 は，b3_pulse_measurement.ino の中身である．このプログラムでは，5 行目に，time，pulse を 1 行でシリアルモニターに表示している．9 行目で，起動からの時間を計測し，10 から 16 行目のプログラムが 1 分間動作するように制御されている．11 行目では，rate 関数から心拍数が返り値として出力されている．12 行目でミリ秒から秒に変換し，13 から 15 行目で，時間と心拍数を 1 行でシリアルモニターに出力している．

```
1  #include "myPulse.h"
2
3  void setup() {
4    Serial.begin(115200); シリアル通信の開始//
5    Serial.println("time,pulse");
6  }
7
8  void loop() {
9    uint32_t time=millis();
10   if(time<60000){
11     uint8_t pulse=rate();
12     float second=time/1000.0;
13     Serial.print(second);
14     Serial.print(",");
15     Serial.println(pulse);
16   }
17 }
```

図 1.16: b3_pulse_measurement.ino

図 1.17 は，myPulse.h の中身である．これは配布されたものから変更を加えていないため，説明は省略する．

```
1 #ifndef MyPulse_h
2 #define MyPulse_h
3
4 #include <Arduino.h>
5
6 uint16_t top();
7 uint8_t rate();
8
9 #endif
```

図 1.17: myPulse.h

図 1.18 と図 1.19 は，myPulse.cpp の中身である．

図 1.18 は主に top 関数が実装されている．6，7 行目では，ワーク B-2 で決定した閾値に設定が変更されている．32 行目では返り値として時間を返すようにされている．

図 1.19 は rate 関数が実装されている．39 行目では時間を記録している．40 行目では，top 関数の返り値を保存している．41 から 44 行目では，時間を保存する変数が定義されている．45 行目は心拍数を保存する変数が定義されている．47 行目から 57 行目は top 関数から値が返ってきた時に動作する．48 行目では，firstpeak に 1 回目のピークを確認できた時間を保存する．49 行目から 56 行目では，firstpeak に値が代入された時に動作し，50 行目で top 関数から返り値として 2 回目のピークの時間が secondpeak に保存される．51 行目から 55 行目では，secondpeak に値が代入された時に動作し，周期を計算し，心拍数を計算し，その値を返すようになっている．

```
#include "myPulse.h"

# define PIN_IN A0
# define DELAY_TIME 20

# define TOP_THRESHOLD 400
# define DOWN_THRESHOLD 2
# define UP_THRESHOLD 1

uint16_t top(){
  uint16_t maximum = 0; // 最大値の保存用
  uint16_t value = 0; // 読み込んだ信号の保存用
  uint8_t up_count = 0; // 上昇回数の保存用
  uint8_t down_count = 0; // 下降回数の保存用
  uint8_t timeout = 255; // タイムアウトまでのカウント
  uint32_t time=millis();
  while(timeout--){
    value = analogRead(PIN_IN); // アナログ入力で値を読み取る

    if(value >= TOP_THRESHOLD){ // 読み込んだ値が山の閾値以上である場合のみピーク検出の処理を行う
      if(value > maximum){
        maximum = value;
        up_count++;
        down_count = 0;
      }
      else{
        down_count++;
      }
    }

      if(up_count>UP_THRESHOLD && down_count>DOWN_THRESHOLD) {
      return time;
      }
    delay(DELAY_TIME);
  }
  return 0;
}
```

図 1.18: myPulse.cpp の top 関数

```
38 uint8_t rate(){
39   uint32_t time=millis();
40   uint32_t peaktime=top();
41   uint32_t lasttime;
42   uint32_t firstpeak;
43   uint32_t secondpeak;
44   uint32_t interval;
45   uint8_t bpm;
46
47   if(peaktime>0){
48     firstpeak=peaktime;
49     if(firstpeak>0){
50     secondpeak=top();
51     if(secondpeak!=0){
52       interval=secondpeak-firstpeak;
53       bpm=60000/interval;
54       return bpm;
55       }
56     }
57   }
58     return 0;
59 }
```

図 1.19: myPulse.cpp の rate 関数

参考文献

[1] 東京大学保健センター，脈拍，https://www.hc.u-tokyo.ac.jp/checkupresult/explanation/pr/．(2025/10/30 参照).

[2] 第一三共エスファ株式会社，薬剤師が知っておきたい「心電図」の基本，https://med2.daiichisankyo-ep.co.jp/cardiology/knowledge/arrhythmia03.php?certification=1．(2025/10/30 参照).