平成 26 年度卒業論文

Raspberry Pi を用いた
図書館の騒音調査

Investigation of noise in a library using Raspberry Pi

函館工業高等専門学校　情報工学科 5 年

東海林研究室　西谷寧々

目次

付録

1.序論

1.1 はじめに

本校図書館は、本の貸し出しを始め、授業での利用や読書、放課後の自習など、利用する学生があとを絶たない。PC の利用、論文の検索なども行うことができるため、学生には必要不可欠な施設であると言える。特に試験前やレポート作成時などは利用する学生で溢れ返り、それに伴って騒がしさ（騒音）も増す。その日の利用者数や環境によっては、他の利用者の集中力に違いが生じると考える。

また、最近の論文[1]では、騒音が必ずしも集中力にマイナスの影響を与えるわけではないという研究結果が出ている。図書館などの静かな環境で作業をするよりも、物音や人の声のような適度な騒音のある環境の方がクリエイティブになるというものである。脳の働きには、無音だと集中できないという性質があるようだ。

そこで本研究では、図書館利用者の利便性の向上を目的として挙げ、図書館の騒音レベルを自動で日常的に調査するシステムを開発する。その後は、実際にそのシステムを利用し、騒音レベルやそのときの図書館の環境が、ユーザの集中力にどのような影響を与えるかを学習的に探る。

1.2 英文アブストラクト

Abstract： This school Library is necessary facility for a student. A main reason are renting on a book and reading, self-teaching after school, using in a lesson. But, when there are a lot of users, there is noisy. I think the user's concentration is different by environment of the library. By a recent thesis, the noisy don't always have a minus influence to the concentration. The main purpose of this research is improving convenience for users of the library. Develop the system to investigate the noise level in automatically. And, we investigate that have an influence in the user's concentration at the environment of the library to using a system.

Key words： environment of the library, Raspberry Pi

2.システムの概要

本章では、システムの仕様、開発環境について説明する。

2.1 システム仕様

本研究では、図書館に設置する騒音レベルの測定器には、小型 PC である Raspberry Pi を用いた。録音は Raspberry Pi に接続した USB マイクで行い、録音した音は一度 WAVE ファイルで保存する。WAVE ファイルからは音の音圧（振幅）データだけを取り出し、測定器はその音圧データから騒音レベルを求める。その後、測定器はさらに図書館の温度と湿度も測定する。測定は、Raspberry Pi に付属している GPIO ピンとセンサモジュールを接続することで行い、WEB サーバに騒音レベルとともに送信する。次に WEB サーバは受信した測定データを参照し、一定時間ごとに騒音レベルや温度、湿度を WEB ブラウザで表示する。

システムのユーザは可視化したすべての情報を WEB ページからリアルタイムに確認できる(図 1)。

さらに、WEB ページには入力ボタンを設け、現在の図書館の環境において、ユーザが集中できるかどうかの可否を学習できる機能の追加を図る。また、それによって得た集中と騒音に関する情報も WEB ページで表示する。

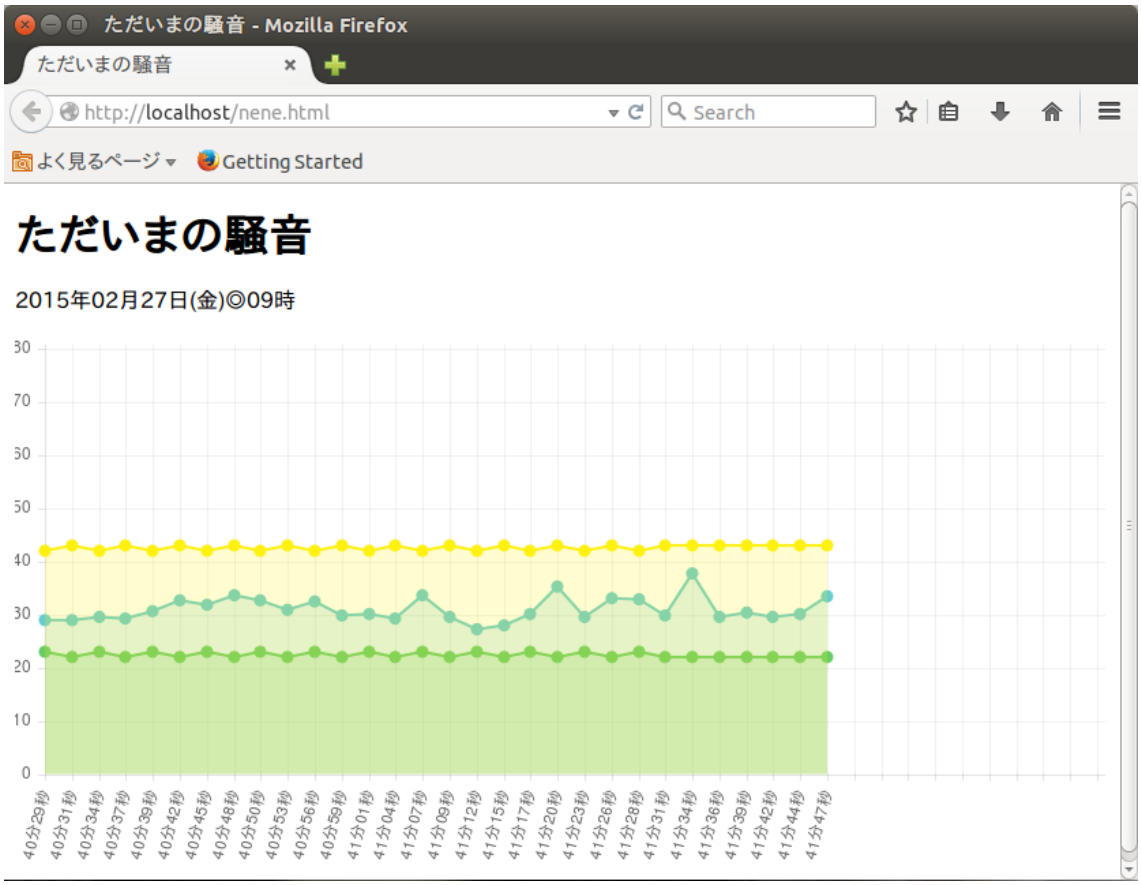
ただいまの騒音 - Mozilla Firefox
ただいまの騒音
http://localhost/nene.html
Search
よく見るページ
Getting Started
ただいまの騒音
2015年02月27日(金)◎09時
30
70
30
50
40
30
20
10
0
40分29秒
40分31秒
40分34秒
40分37秒
40分39秒
40分42秒
40分45秒
40分48秒
40分50秒
40分53秒
40分56秒
40分59秒
41分01秒
41分04秒
41分07秒
41分09秒
41分12秒
41分15秒
41分17秒
41分20秒
41分23秒
41分26秒
41分28秒
41分31秒
41分34秒
41分36秒
41分39秒
41分42秒
41分44秒
41分47秒

図 1 結果画面

2.2 Raspberry Pi について

本研究で測定器として使用する Raspberry Pi[3][4]について述べる(図 2)。

Raspberry Pi とは、英国ラズベリーパイ財団（Raspberry Pi Foundation）が開発した小型コンピュータである。低価格で、持ち運びしやすく、誰でも気軽に触れることができるシンプルなコンピュータとして、次世代 IT 社会の発展へと寄与することを目的に、2012 年 3 月に発売された。Raspberry Pi のサイズは 85.6[mm]*53.98[mm]*17[mm]と非常にコンパクトであるが、CPU が搭載しており、入出力操作は一般的な PC と同様に行うことができる。しかしハードディスクなどは搭載しておらず、OS は、別途用意した SD カードにインストールし、スロットにセットして起動させる必要がある。また、OS は、Linux のオープンソースを採用している。Raspberry Pi には、メインメモリが 256[MB]の Model A と、512[MB]の Model B の 2 タイプがあり、それぞれ 25 ドル、35 ドルという低価格で購入できる。

なお、Model B は、USB ポートを 2 つ搭載するとともに、10/100BASE-T Ethernet による有線 LAN ネットワーク接続が可能であるため、本研究の測定器には、Model B を採用した。Raspberry Pi Model B の OS には、Raspbian を用いる。公式ページ[2]からイメージファイルをダウンロードし、インストールを行った後、IP アドレスの割り振りや、SSH の許可、Linux コマンドのインストールなどといった初期設定[5]をした。

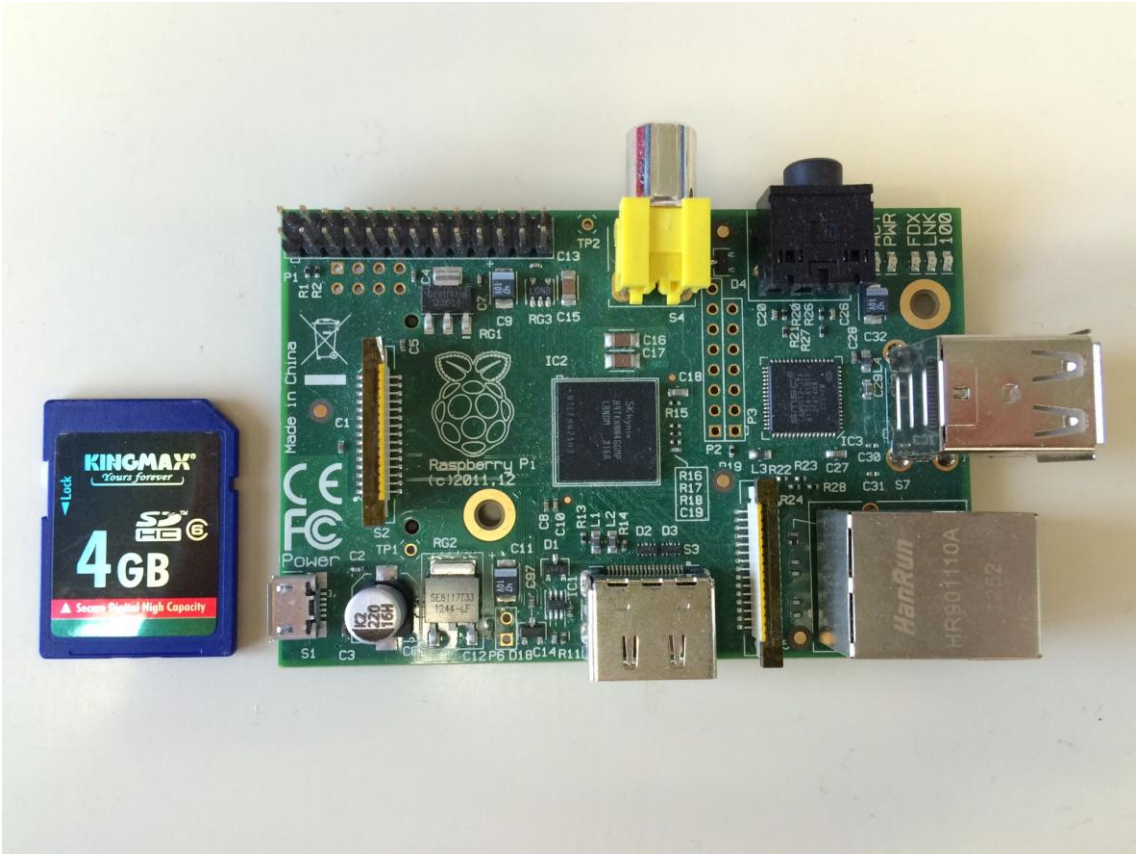

図 2 Raspberry Pi Model B

2.3 接続機器について

次に、測定に用いる機器について説明する。

図書館の音の録音には、ELECOM USB スタンドマイク HS-MC02UBK という USB マイクを用いた(図 3)。フレキシブルアームを採用したもので、マイク位置を自在に調節できる。また、ドライバが不要なため、Raspberry Pi に接続するだけで使用が可能である。

図 3 USB マイク

温度と湿度の測定には、DHT11 Humidity & Temperature Sensor というセンサモジュールを使用した(図 4)。測定範囲は、温度が 0～50℃、湿度が 20～90%である。300 円と安価だが、測定誤差は温度が±2℃、湿度が±5%と非常に多く、データ精度も温度 8[bit] (1℃単位)、湿度[8bit](1%単位)と小数点以下を計ることができない。なお、動作電圧は約 3.3V～5.5V であり、出力は単線双方向シリアルで得られる。

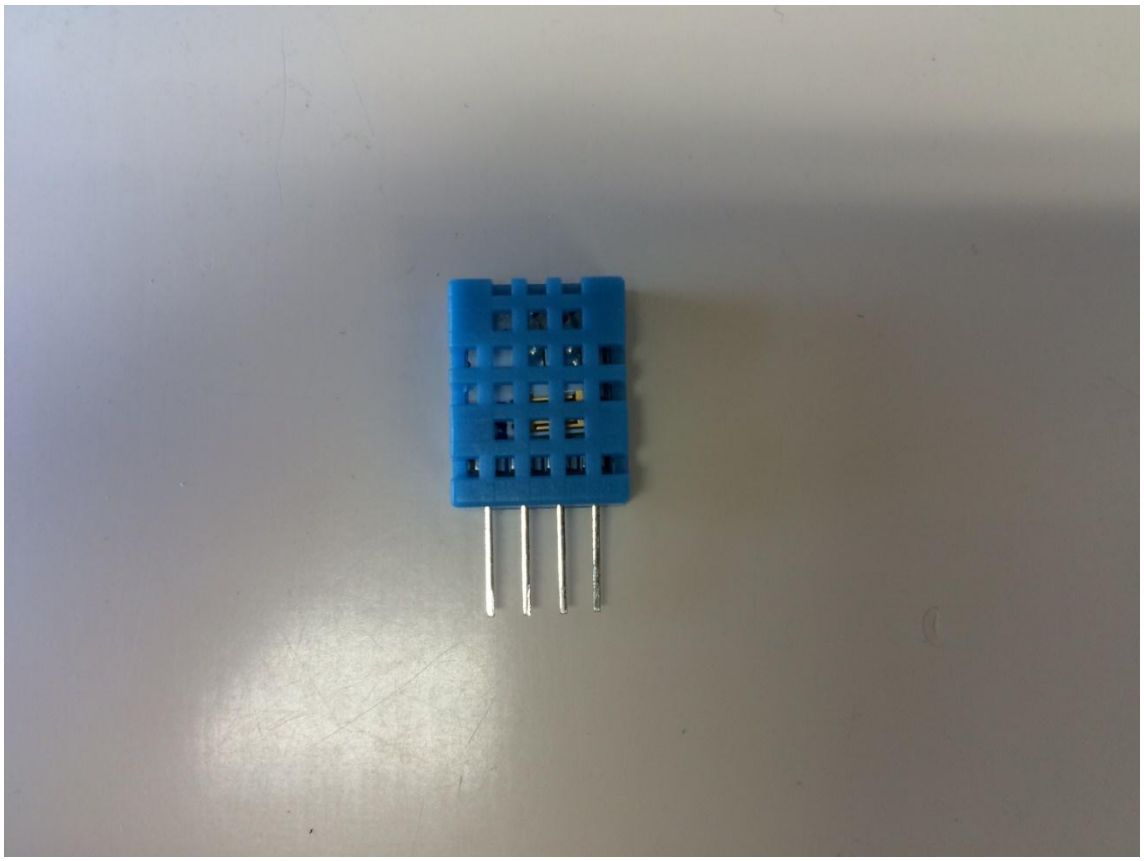

図 4 センサモジュール

上記の機器を、初期設定を終えた Raspberry Pi に接続し、測定器を完成させる。USB マイクは Raspberry Pi に搭載された USB ポートに、センサモジュールも同様に付属の GPIO ピンに接続し、それぞれ接続の確認を行う。USB マイクに関しては、USB ポートとの接続に USB ハブを噛ませると、どうにもまともな音が取れなかったため、Raspberry Pi 付属の USB ポートに直接接続することを進める。

図 5 に接続時の写真、図 6 に、GPIO ピンとセンサモジュールの接続回路図、図 7 に完成した測定器の全体図を添付する。

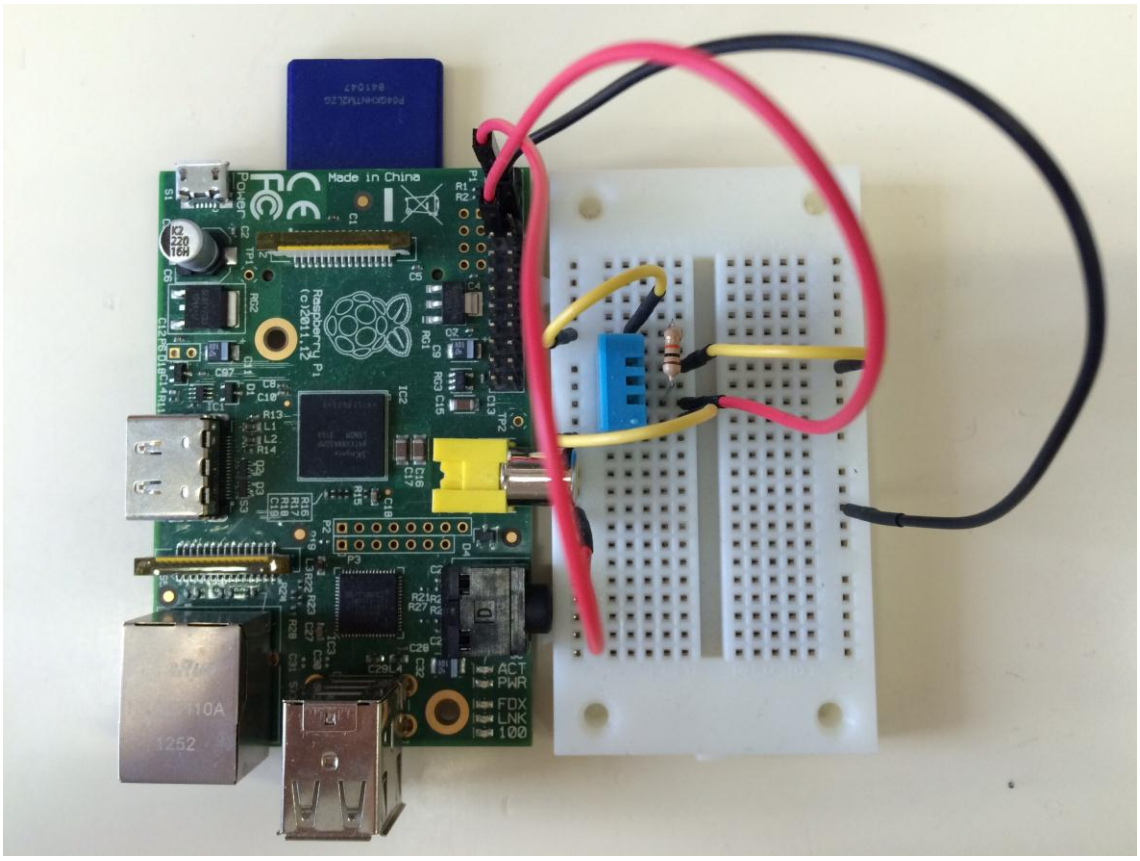

図 5 湿温度センサモジュールの接続

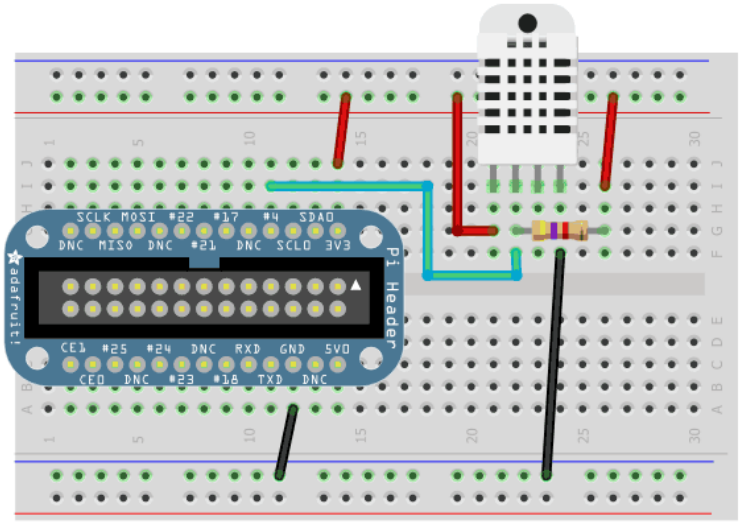

図 6 湿温度センサモジュールと回路図

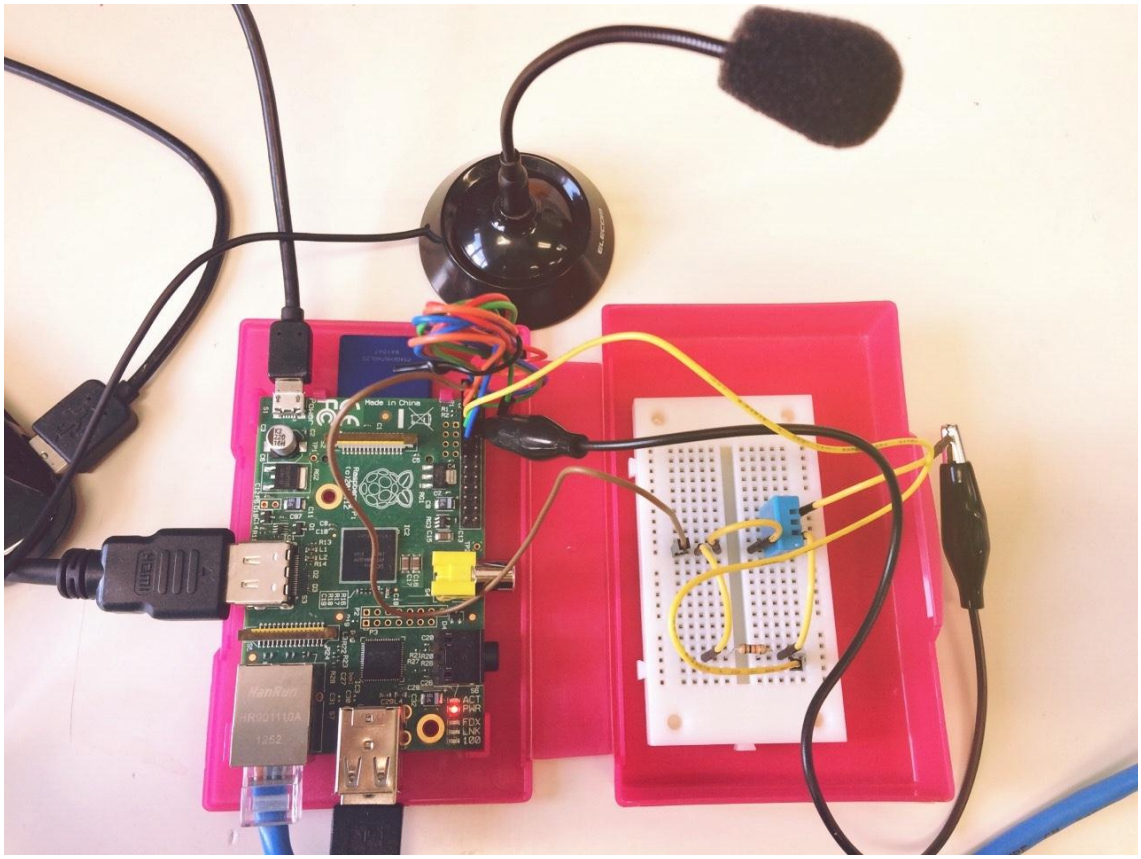

図 7 測定器

2.4 サーバの仕様

続いて、サーバ機の設定を行う。サーバ機として用いる PC の OS には、Ubuntu14.04 Linux をインストールした。そして、WEB サーバの構築に伴い、WEB サーバソフトウェアである apache2[6]をインストールし、同時に PHP[7]の使用環境も整えた。

また、WEB サーバは測定器である Raspberry Pi から受信したデータを参照し、WEB ページへの出力を行う。そのため、サーバ機に FTP サーバの構築も行うことで、Raspberry Pi と WEB サーバとのデータの送受信を、FTP を用いて行うことができる。本研究での FTP サーバーには vsftpd[8]を採用し、そのときのサーバ機と Raspberry Pi の通信には、Ethernet による有線 LAN ネットワーク接続を用いる。

さらにサーバ機の SSH も許可することで、サーバ機から Raspberry Pi へのリモートログインを可能にした。

2.5 開発言語

最後に、使用した開発言語の説明を述べる。

測定器である Raspberry Pi が WAVE ファイルからデータを取得し、騒音レベルを求めるまでの処理と、その他データの CSV ファイルへの出力には、C++言語を用いる。温度と湿度の測定に当たっては、WiringPi[9]という C++ライブラリを用いて、センサモジュールからそれぞれデータを得る。なお、CSV ファイルには、測定時刻、騒音レベル、温度、湿度の 4 種類のデータを出力する。

また、WEB サーバは Raspberry Pi から受信した CSV ファイルのデータを参照し、PHP を用いて HTML 化する。WEB ブラウザは Chart.js という JavaScript ライブラリを用いて、測定時刻ごとにリアルタイムに変化するグラフを描画する。その他 WEB ページのレイアウトには、HTML と JavaScript を使用する。

以上で、システムの概要と処理に用いる機器や言語の説明を終える。

3.騒音レベルの導出

本章では、騒音レベルの導出に関して述べる。

3.1 録音方法

初めに、Raspberry Pi には、PulseAudio[10]をインストールする。PulseAudio とは、GNOME などのデスクトップ環境で一般的に使用されるサウンドサーバである。インストールを行うことで、Rasbian 既存のサウンドアプリケーションである ALSA に PulseAudio の機能が追加される。それにより、すべてのサウンドが同じひとつのサウンドサーバを経由できるため、制御しやすく、衝突が起こりにくくなる。alsamixer を用いて、USB マイクの音量調整を行い、録音に移る。

図書館の録音には接続／調整した USB マイクと、ALSA サウンドドライバのための、コマンドラインサウンドレコーダである arecord を用いる。arecord は、録音から WAVE への保存までを行うことが可能で、その保存形式も自由に変更することができる。本研究での WAVE ファイルのフォーマットは、サンプリング周波数 8000[Hz]、量子化ビット数 16[bit]とし、そうすることで、音圧の大きさを－32768～＋32767 の範囲の整数データで表現することが可能となる。また、録音は 10 秒ごとに 1 秒間行う。

図 8 に、取得した音圧データのグラフを添付する。

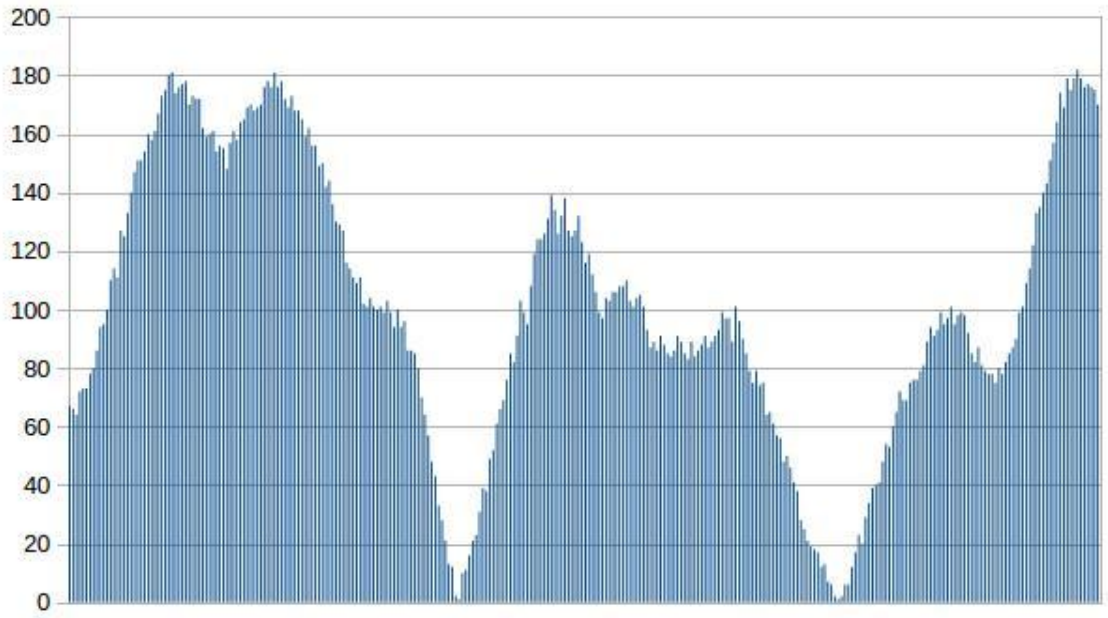

図 8 音圧のグラフ

3.2 騒音レベルの計算

続いて、騒音レベルの導出を行う。

初めに、測定器である Raspberry Pi は、保存された WAVE ファイルの音圧データだけを取り出し、その音圧データの平均を求める。さらに、その平均音圧の絶対値を計算し、最終的には 0～＋32768 の整数データを用いて、騒音レベルを得る。

騒音レベルの計算には、以下の式を用いる。なお、本研究では、騒音レベルの単位にはデシベル（[dB]）を採用する。

$$\text{式 } Lp = 20\log(P/P0) \text{ [dB]}$$

ここで、Lp はそのときの 1 秒間の騒音レベル[dB]を表す。P には、WAVE ファイルから求めた最終的な整数データを用い、P0 には音圧の最小値である 1 を用いる。

デシベルを用いることで、倍率で表示するよりも桁数を抑えることができるため、システムのユーザにも直感的に分かり易い値であると言える。

3.3 結果の保存

導出した騒音レベルはテキストファイルに出力する。

4.温度・湿度の取得

本章では、温度／湿度の導出に関して述べる。

4.1 測定方法

温度と湿度の測定は、Raspberry Pi に接続したセンサモジュール[11]で行う。騒音レベルの導出と同様に、データは 10 秒ごとに取得する。センサモジュールの出力はシリアルで得られるため、取得したデータは C++言語を用いてパラレルに変換する。

なお、その処理のソースコードは付録に添付する。

4.2 結果の保存

導出した温度と湿度は、テキストファイルに出力する。

5.測定結果の送信

本章では、データの出力を終えたテキストファイルの送信に関して説明する。

測定器である Raspberry Pi は、FTP を用いてサーバ機 PC にテキストファイルを送信する。open コマンドでサーバ機にリモートログインし、get コマンドでテキストファイルを指定し、受信する。

なお、以上までに述べたデータの測定からテキストファイルの送信までの一連の動作は、測定器である Raspberry Pi でスクリプトを動作させることで行う。スクリプトは、実行時刻をテキストファイルに出力する処理を始め、3～4 章で行った arecord を用いた録音、C++で記述した実行ファイルの実行、そして本章のテキストファイルの送信までの動作を記述し、10 秒ごとに繰り返し実行することとする。送信するテキストファイルは、実行時刻、騒音レベル、温度、湿度の 4 種類である。

スクリプトのソースコードを図 9 に載せる。

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ cat nene.command
open 192.168.1.106
user nishiya tagamogohan0802
cd /var/www/html
put result.txt
put log.txt
put h.txt
put t.txt
pi@raspberrypi ~ $ cat nene.sh
#!/bin/sh
while true ; do
sed -e '1,1d' log.txt > gomi.txt
cp gomi.txt log.txt
date +"'%M分%S秒'" >> log.txt
sudo ./sensor
arecord -r 8000 -f S16_LE -d 1 a.wav
./a.out a.wav
ftp -n< nene.command
sleep 1
done
pi@raspberrypi ~ $
```

図 9 スクリプトのソースコード

6.WEB ブラウザでの表示

本章では、WEB ページの作成について説明する。

6.1 受信データの参照

測定結果がすべて保存されたテキストファイルを Raspberry Pi から受信した後、サーバ機は PHP を用いて、そのデータを参照し、HTML 化する。

6.2 WEB ページの作成

データの参照後は、データに基づいて実行時刻ごとに、騒音レベル、温度、湿度を同一のグラフに描画する処理を、JavaScript を用いることで行う。JavaScript のライブラリである Chart.js を使用し、グラフの種類は折れ線を選択する。

また、その他レイアウトは PHP と HTML を使用し、現在時刻の表示や、10 秒ごとのページの自動更新などの処理も行うことで、リアルタイムで変化グラフを作成することができる。

なお、HTML ソースを、付録に添付する。

7.まとめと今後の展望

7.1 まとめ

図書館の騒音レベルや温度、湿度などのデータを、WEB 上でリアルタイムに可視化できるシステムの開発を行うことができた。

温度、湿度の測定に関して、使用したセンサモジュールのスペックに問題があり、小数点以下のデータが取得できなかったため、図書館の正確な温度と湿度を計測できなかった。

さらに、測定時に不規則的なデータの取りこぼしなども多く見られるうえに、その調整のための処理にも多く時間が掛かった。そのため、騒音レベルのみだと 2 秒でスクリプトのすべての処理を終えることができるが、温度と湿度の測定を含めるとその 6 倍以上の時間が必要になってしまった。

また、最終目的である本研究のシステムを利用した騒音と集中力との関連は、探ることができなかった。

7.2 今後の展望

今後は、測定に用いたセンサモジュールの改良を行う。また、グラフの描画に関して、折れ線の動きの変化だけでなく、ある一定の騒音レベルを超えた時点でグラフの色に変化を加えるなど、色覚的な可視化も増やす。

騒音と集中力に関しては、温度や湿度だけでなく、時間帯や日付によっても集中力に影響があるのかを知る要素のひとつとして加える。

さらに図書館で検証実験を行う。

参考文献

[1] Journal of Consumer Research 12 月号,
http://www.jstor.org/stable/full/10.1086/665048

[2] Raspberry Pi,
http://www.raspberrypi.org/

[3] Raspberry Pi Type B,
https://www.switch-science.com/catalog/1268/

[4] Introduction Raspberry Pi,
http://thinkit.co.jp/story/2013/08/06/4162

[5] Raspberry Pi の 12 の使い方,
http://readwrite.jp/archives/4888

[6] Ubuntu に Apache2 をインストールしたときの記録,
http://mizupc8.bio.mie-u.ac.jp/pukiwiki/index.php?Ubuntu/Server/Web/Apache2

[7] PHP マニュアル,
http://php.net/manual/ja/install.unix.apache2.php

[8] vsftpd のインストールと設定,
http://www.server-world.info/query?os=Ubuntu_14.04&p=ftp

[9] Raspberry Pi の GPIO 制御 WiringPi のインストール,
https://tool-lab.com/2013/12/raspi-gpio-controlling-command-2/

[10] pulseaudio のインストールと設定,
http://d.hatena.ne.jp/penkoba/20130928/1380351297

[11] Interfacing Temperature and Humidity Sensor DHT11 With Raspberry Pi,
http://www.rpiblog.com/2012/11/interfacing-temperature-and-humidity.html

付録

付録 1 WAVE ファイル操作・騒音レベル計算プログラム

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include <linux/soundcard.h>
#include <math.h>
#define BUFSIZE 64000
#define N 16000
#define DATA 30

typedef struct
{
          FILE*     fp;
          short     format;
          short     channel;
          int                 rate;
          short     bit;
          long      offset;
          int                 length;
}
WAVE;

// main 関数

static int wave_read_file( char* fname, WAVE* wave );

int main( int argc, char** argv )
{
          WAVE      wave;
          char      buf[ BUFSIZE ];
```

```
	int		len;

	if( argc != 2 )
	{
		fprintf( stderr, "usage : wave filename¥n" );
		return 1;
	}

	if( wave_read_file( argv[1], &wave ) != 0 )
	{
		fprintf( stderr, "%s は読み込めない。仕方ないね。¥n", argv[1] );
		return 1;
	}


	fclose( wave.fp );
	return 0;
	}

static int wave_read_file( char* fname, WAVE* wave )
{
	char	buf[32];
	int		len;

	if( ( wave->fp = fopen( fname, "r" ) ) == NULL )
	{
		fprintf( stderr, "%s を開けない。仕方ないね。¥n", fname );
		return -1;
	}

	fread( buf, 8, 1, wave->fp );

	if( strncmp( buf, "RIFF", 4 ) != 0 )
	{
		fprintf( stderr, "RIFF ヘッダがない。仕方ないね。¥n" );
		fclose( wave->fp );
```

```
		return -1;
	}

	fread( buf, 4, 1, wave->fp );

	if( strncmp( buf, "WAVE", 4 ) != 0 )
	{
		fprintf( stderr, "WAVE ヘッダがない。仕方ないね。¥n" );
		fclose( wave->fp );
		return -1;
	}

	while( 1 )
	{
		fread( buf, 8, 1, wave->fp );
		len = *( int* )( &buf[4] );
		if( strncmp( buf, "fmt ", 4 ) != 0 )
		{
			if ( fseek( wave->fp, len, SEEK_CUR ) == -1 )
			{
				fprintf( stderr, "fmt チャンクがない。仕方ないね。.¥n" );
				fclose( wave->fp );
				return -1;
			}
		}
		else
		{
			break;
		}
	}

	fread( buf, len, 1, wave->fp );

	wave->format		= *( ( short* )( &buf[0] ) );
	wave->channel		= *( ( short* )( &buf[2] ) );
	wave->rate = *( ( int* )( &buf[4] ) );
```

```
wave->bit = *( ( short* )( &buf[14] ) );

if( wave->format != 1 )
{
	wave->format = 0;
}


while( 1 )
{
	fread( buf, 8, 1, wave->fp );
	len = *( int* )( &buf[4] );

	if( strncmp( buf, "data", 4 ) != 0 )
	{
		if( fseek( wave->fp, len, SEEK_CUR ) == -1 )
		{
			fprintf( stderr, "data チャンクがない。仕方ないね。¥n" );
			fclose( wave->fp );
			return -1;
		}
	}
	else
	{
		break;
	}
}

wave->length = len;

if( ( wave->offset = ftell( wave->fp ) ) == -1 )
{
	fprintf( stderr, "PCM のオフセットが読み込めない。仕方ないね。¥n" );
	fclose( wave->fp );
	return -1;
}
```

```
int        i;
short      sound;
int        average = 0;
double     level = 0;
char       data[N];


fread( data, N, 1, wave->fp );

for( i = 0; i < N; i=i+2 )
{
           sound = abs( *( ( short* )( &data[i] ) ) );
           printf( "%d¥n", sound );
           average = average + sound;
}

average = average/8000;

//騒音レベル
level = 10*log10( average*average );

fclose( wave->fp );

char       log[DATA][256] = {0};
FILE       *fpfp;

if( ( fpfp = fopen( "result.txt", "r" ) ) == NULL )
{
        fprintf( stderr, "ファイルオープン失敗。仕方ないね。¥n" );
        return -1;
}

for( i = 0; i < DATA; i++ )
{
           fgets( log[i], 256, fpfp );
```

```
        }

        fclose( fpfp );

        if( ( fpfp = fopen( "result.txt", "w" ) ) == NULL )
        {
                fprintf( stderr, "ファイルオープン失敗。仕方ないね。¥n" );
                return -1;
        }

        for( i = 1; i < DATA; i++ )
        {
                  fprintf( fpfp, "%s", log[i] );
        }
        fprintf( fpfp, "%f¥n", level );

        return 0;
}
```

付録 2 温度湿度測定プログラム

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#define MAX_TIME 85
#define DHT11PIN 7
#define DATA 30
int dht11_val[5] = {0, 0, 0, 0, 0};

void dht11_read_val()
{
	uint8_t lststate = HIGH;
	uint8_t counter = 0;
	uint8_t j = 0,i;
	float farenheit;

	for( i = 0; i < 5; i++ ) dht11_val[i]=0;

	pinMode( DHT11PIN, OUTPUT );
	digitalWrite( DHT11PIN, LOW );
	delay( 18 );
	digitalWrite( DHT11PIN, HIGH );
	delayMicroseconds( 40 );
	pinMode( DHT11PIN, INPUT );

	for( i = 0; i < MAX_TIME; i++ )
	{
		counter=0;
		while( digitalRead( DHT11PIN ) == lststate )
		{
			counter++;
			delayMicroseconds( 1 );
			if( counter == 255 ) break;
		}
```

```
            lststate = digitalRead( DHT11PIN );
            if( counter == 255 ) break;

            if( ( i >= 4 ) && ( i%2 == 0 ) )
            {
                        dht11_val[j/8] <<= 1;
                        if( counter > 16 ) dht11_val[j/8] |= 1;
                        j++;
            }
}

FILE *fp_h, *fp_t;
char h[DATA][256] = {0};
char t[DATA][256] = {0};

if( ( fp_h = fopen( "h.txt", "r" ) ) == NULL )
{
        fprintf( stderr, "失敗¥n" );
        return -1;
}
if( ( fp_t = fopen( "t.txt", "r" ) ) == NULL )
{
        fprintf( stderr, "失敗¥n" );
        return -1;
}

for( i = 0; i < DATA; i++ )
{
            fgets( h[i], 256, fp_h );
            fgets( t[i], 256, fp_t );
}

fclose( fp_h );
fclose( fp_t );
```

```
        if( ( fp_h = fopen( "h.txt", "w" ) ) == NULL )
        {
                fprintf( stderr, "失敗¥n" );
                return -1;
        }
        if( ( fp_t = fopen( "t.txt", "w" ) ) == NULL )
        {
                fprintf( stderr, "失敗¥n" );
                return -1;
        }

        for( i = 1; i < DATA; i++ )
        {
                  fprintf( fp_h, "%s", h[i] );
                  fprintf( fp_t, "%s", t[i] );
        }

        if( ( j >= 40 ) && ( dht11_val[4] == ( ( dht11_val[0]+dht11_val[1]+dht11_val[2]+dht11_val[3]) &
0xFF ) ) )
        {
                  printf( "温度 = %d.%d *C¥n", dht11_val[2],dht11_val[3] );
                  printf( "湿度 = %d.%d %%¥n", dht11_val[0],dht11_val[1] );
                  fprintf( fp_h, "%d.%d", dht11_val[2],dht11_val[3] );
                  fprintf( fp_t, "%d.%d", dht11_val[0],dht11_val[1] );

        }
        else
        {
                  fprintf( fp_h, "0.0¥n");
                  fprintf( fp_t, "0.0¥n");
        }
        fclose( fp_h );
        fclose( fp_t );
}

int main(void)
```

```
{
        if(wiringPiSetup()==-1)
        exit(1);
        dht11_read_val();
        return 0;
}
```

付録 3 HTML ソースコード

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja" lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
<title>ただいまの騒音</title>
<script src="Chart.js"></script>
</head>
<body>

<h1>ただいまの騒音</h1>

<p>
<?php
$week = array("日", "月", "火", "水", "木", "金", "土");
echo date("Y 年 m 月 d 日", time())."(".$week[date("w", time())].")".date("◎H 時", time());
?>
</p>
<script type="text/javascript">
<!--
function sample() {
var canvas = document.getElementById('sample1');
if (canvas.getContext) {

var context = canvas.getContext('2d');
context.fillRect(20,40,50,100);
context.strokeStyle = 'rgb(00,00,255)';
context.fillStyle = 'rgb(255,00,00)';
context.strokeRect(200,80,100,50);
context.arc(150,75,60,Math.PI*1,Math.PI*2,true);
context.fill();

}
}
```

```
-->
</script>
<canvas id="lineChartCanvas" width="800" height="400"></canvas>
<script>
	var lineChartData =
	{
		<?php

		$fpfp = fopen('log.txt', 'r');
		if( $fpfp )
		{
			printf( "labels : [" );
			$N = 29;
			for( $i = 0; $i < $N; $i++ )
			{
			$line = fgets( $fpfp );
				echo $line;
				printf(",");
			}
			$line = fgets( $fpfp );
			echo $line;
			printf( ",'  ','  ','  ','  ','  ','  ','  ','  ','  ','  ', ]¥n," );
		}
		$flag = fclose( $fpfp );

		?>

		datasets:
		[
		{
			fillColor:		"rgba(102,204,204,0.2)",
			strokeColor:			"rgba(102,204,204,1.0)",
			pointColor:			"rgba(102,204,204,1.0)",
			pointStrokeColor:	"rgba(102,204,204,1.0)",

			<?php
```

```
$fp = fopen('result.txt', 'r');
if( $fp )
{
        printf( "data : [" );
        $NN = 29;
        for( $i = 0; $i < $NN; $i++ )
        {
        $line = fgets( $fp );
                echo $line;
                printf(",");
        }
        $line = fgets( $fp );
        echo $line;
        printf( "]" );
}
$flag = fclose( $fp );

?>
},
{
        fillColor:          "rgba(102,204,102,0.2)",
        strokeColor:                "rgba(102,204,102,1.0)",
        pointColor:                 "rgba(102,204,102,1.0)",
        pointStrokeColor:   "rgba(102,204,102,1.0)",

        <?php

        $fp_h = fopen('h.txt', 'r');
        if( $fp_h )
        {
        printf( "data : [" );
        $NN = 29;
        for( $i = 0; $i < $NN; $i++ )
        {
        $line = fgets( $fp_h );
```

```
                    echo $line;
                    printf(",");
                }
                $line = fgets( $fp_h );
                echo $line;
                printf( "]" );
            }
            $flag = fclose( $fp_h );

            ?>
        },
        {
            fillColor:          "rgba(255,255,204,0.0)",
            strokeColor:                "rgba(13,8,22,1.0)",
            pointColor:                 "rgba(13,8,22,1.0)",
            pointStrokeColor:   "rgba(13,8,22,1.0)",

            <?php

            $fp = fopen('t.txt', 'r');
            if( $fp )
            {
                printf( "data : [" );
                $NN = 29;
                for( $i = 0; $i < $NN; $i++ )
                {
                $line = fgets( $fp );
                    echo $line;
                    printf(",");
                }
                $line = fgets( $fp );
                echo $line;
                printf( "]" );
            }
            $flag = fclose( $fp );
```

```
                        ?>
                }

                ]
        }


        var options =
        {

scaleOverlay : true,
scaleOverride : true,
scaleSteps : 8,
scaleStepWidth : 10,
scaleStartValue : 0,
bezierCurve : false,
animation : false
        }
var chart = new Chart(document.getElementById("lineChartCanvas").getContext("2d")).Line(lineChartData,
options);

</script>

<script>
<!--
var timer = "500";
function ReloadAddr(){
        window.location.reload();
}
setTimeout(ReloadAddr, timer);
-->
</script>

</body>
</html>
```