Magicalight マニュアル

◆ ハードウェア

- 使用部品
- 制作回路

◆ ソフトウェア

- 開発したコンポーネント群

2016 年 11 月 14 日版

中沢真太郎，猪瀬将也，片桐大地，小山拓馬，伏見学，神戸菜緒（芝浦工業大学），
土屋彩茜（東京工業大学），佐々木毅（芝浦工業大学）

更新履歴

2016 年 10 月 31 日	第 1 版.
2016 年 11 月 1 日	第 2 版. 誤字及び内容の修正と不足内容の追記.
2016 年 11 月 14 日	第 3 版. コンポーネント修正による該当箇所の修正.

目次

1. 作品紹介 1
2. 機能説明 2
2.1. 火の揺らぎ 2
2.2. 息による切り替え 2
2.3. 連動 3
3. ハードウェア 5
3.1. 使用部品 5
3.1.1. RaspberryPi 5
3.1.2. マイクロホン 6
3.1.3. AD コンバータ 6
3.2. 制作回路 6
4. ソフトウェア 7
4.1. 開発環境 7
4.2. 開発したコンポーネントの説明 7
4.2.1. RaspberryPi 上で起動するコンポーネント群 7
4.2.1.1. アナログデータ受取コンポーネント(AnalogInput) 7
4.2.1.2. LED 揺らぎコンポーネント(PWMFluctuations) 8
4.2.2. WindowsPC 上で起動するコンポーネント群 8
4.2.2.1. 閾値判別コンポーネント(ThresholdSwitch) 8
4.2.2.2. LED 状態管理コンポーネント(LEDStateManager) 10
5. コンポーネント群を起動する前に 11
5.1. RaspberryPi の設定 11
5.1.1. SPI 通信の有効化の方法 11
5.1.2. WiringPi のインストール方法 12
5.2. Windows の設定 12
6. 参考文献 13
7. お問い合わせ 13

1. 作品紹介

今回私たちは「メカと自然の融合」というテーマの下,「Magicalight」という名前の LED キャンドルを制作しました. 機能は「LED の光が火のように揺らぐ」,「息を吹きかけると点灯/消灯する」,「複数の Magicalight が連動する」の 3 つを実装しています．これらの機能の実装によりテーマの達成を目指しました.

また，実機として使用しているアクリルパイプの上部に曇り加工を施し，LED の光を曇り加工部分で透過させることにより，自然の火に感じられるように表現しました．また下部には曇り加工を敢えて施さないことで，内部の回路の形状を見やすくし，より機械的な印象を与えることを目指しました．この上部と下部の違いにより作品の外形でメカと自然の融合の表現を目指しました.

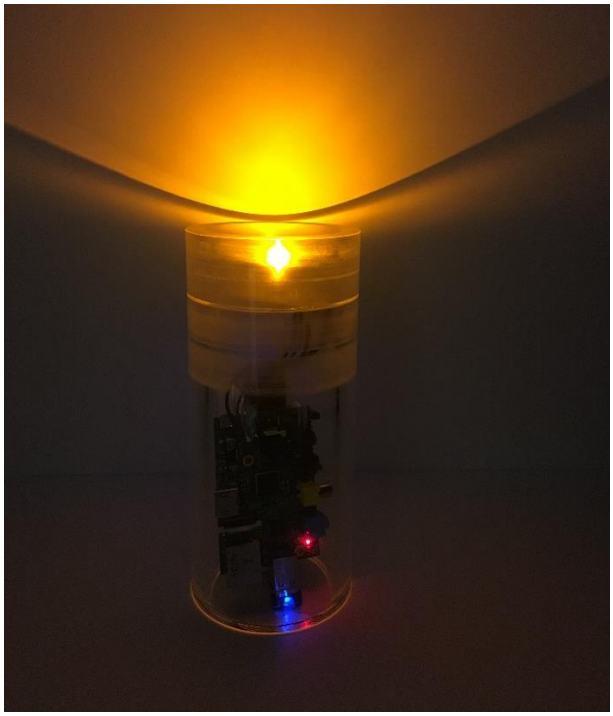

Fig.1 Magicalight

本作品は実機が複数あるときに通信を行うことが可能ですが, 無線通信を用いることで有線通信の場合に必要となるコードを排除しました．さらに，実機への電力供給をモバイルバッテリーで行い，そのバッテリーをアクリルパイプの内部に組み込むことにより，実機の無線化を図りました.

Fig.2 Magicalight（複数版）

2. 機能説明

2.1. 火の揺らぎ

LED の光がより自然の火に感じられるように火の揺らぎ機能を実装しました．今回は RaspberryPi という小型コンピュータを用いて LED の制御を行っています．しかし，RaspberryPi ではアナログデータ出力ができないため PWM（パルス幅変調）を用いて LED の明るさに強弱を持たせました．さらに LED の明るさの変更を不規則に行うことで自然の火特有の不規則な揺らぎの再現を目指しました．

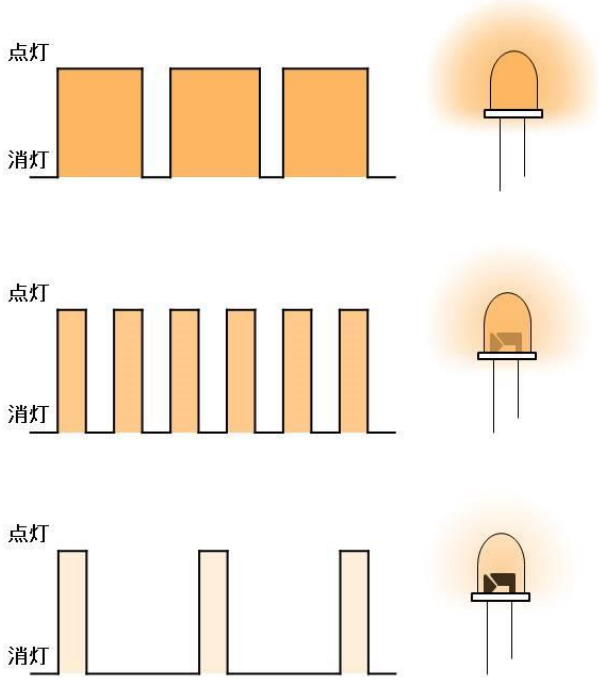

Fig.3 揺らぎの仕組み

2.2. 息による切り替え

息を吹いたら光が消えるようにすることで自然の火が持つ特性を表現しました．また，息を吹いても光が点くというメカトロニクスならではの機能を実装することで，「メカと自然の融合」を目指しました．

本作品の回路にはマイクアンプキットを搭載しており，この部品が息を検知してくれます．息などの入力がない場合のマイクロホンからの出力は Fig.4 のようになっています．この時の出力値は閾値(Threshold)を超えていないので，Magicalight の LED に変化はありません．息が入力された場合の出力は Fig.5 のようになっていています．この時の出力値は閾値を超えているので，コンポーネント ThresholdSwitch で設定した基準値を満たしたとき，Magicalight の LED の状態は変化します．

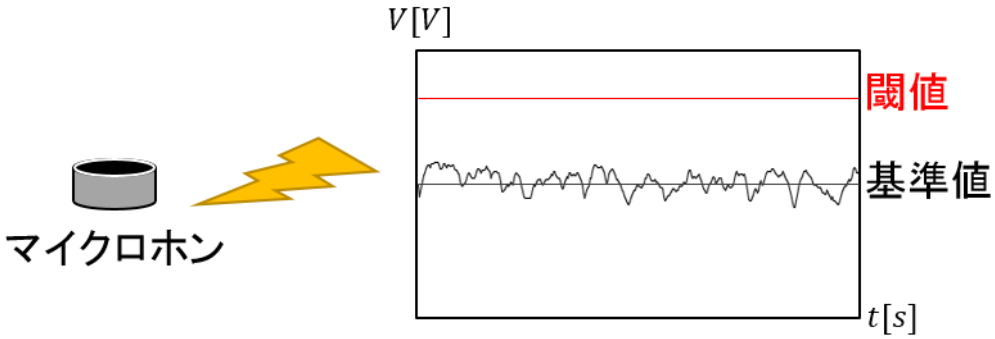

Fig.4 息を吹いていない時のマイクロホンからの出力値

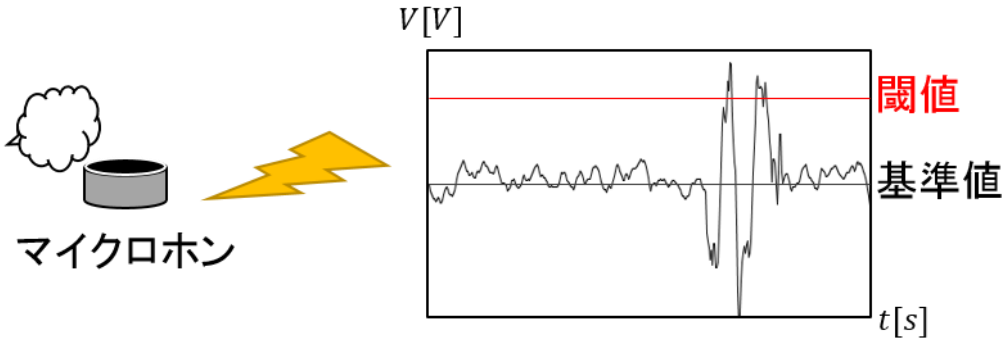

Fig.5 息を吹いた時のマイクロホンからの出力値

2.3. 連動

Magicalight は 4 つのモードを実装しています．以下では，各モードの動作について説明します．

1 つ目のモードは連動モードです．まず初めに，Magicalight の中で任意の一台を親機として設定します．今回制作した Magicalight はマイクロホンに息を吹きかけたときに，それがトリガーとなって内部の LED が点灯/消灯するようになっています．連動モードでは親機に息が吹きかけられると，子機もそれに応じて LED の状態が変化します．動作の様子は Fig.6 のようになります．このとき，子機は息を吹きかけられても変化はせず，子機の点灯/消灯状態は親機の点灯/消灯状態に依存します．

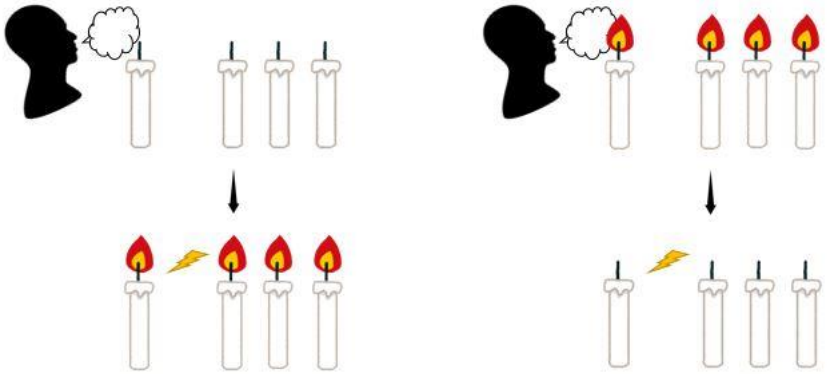

Fig.6 連動モード

2 つ目のモードはイエスマンモードです．1 つ目のモードと同様に，親機に息を吹きかけると子機の状態も同様に変化しますが，1 つ目のモードと違って子機自体に息を吹きかけるとその子機の状態は変化します．親機に息を吹きかけて子機全体の LED の状態を変えても，子機に息を吹きかけるとその子機の状態は変化します．動作の様子は Fig.7 のようになります．

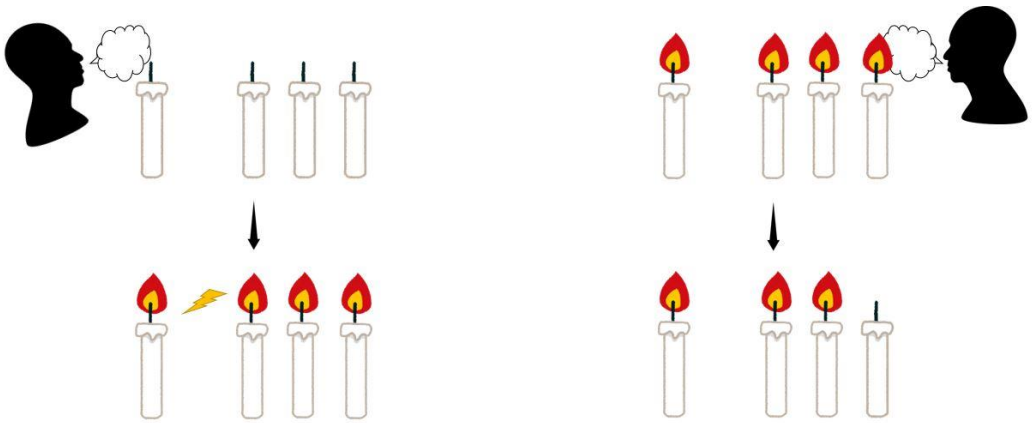

Fig.7 イエスマンモード

3 つ目のモードはランダムモードです．親機を設定する必要がなく，親機と子機の関係もありません．接続されている Magicalight のどれかに息を吹きかけると，全ての Magicalight の状態が不規則に変化します．動作の様子は Fig.8 のようになります．

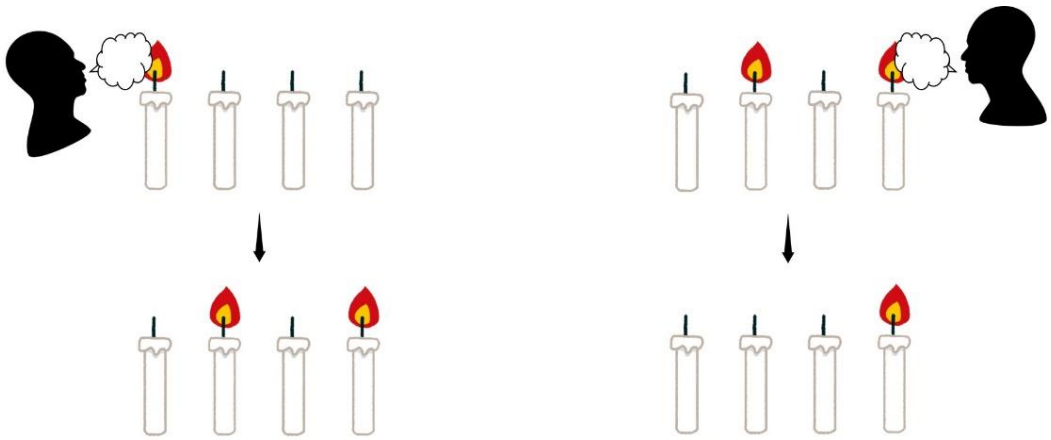

Fig.8 ランダムモード

4 つ目のモードは非連動モードです．接続した Magicalight が連動せず，それぞれ独立して動作します．親機と子機の設定もありません．動作の様子は Fig.9 のようになります．

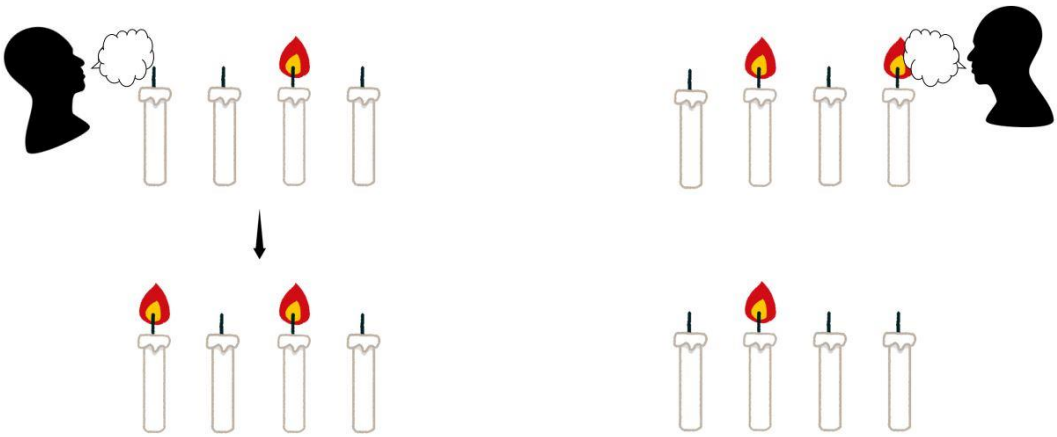

Fig.9 非連動モード

3. ハードウェア

3.1. 使用部品

今回 Magicalight 制作に使用した部品を Table.1 にまとめました．また，外形に使用したアクリルパイプはアクリ屋様[1]で外形 90mm，厚さ 10mm，長さ 1000mm のものを購入しました．この大きさですと RaspberryPi とモバイルバッテリーがアクリルパイプの中にちょうど良く収納できます. またアクリルパイプの加工ですが，芝浦工業大学の工作室で長さ 200mm に切断し，旋盤でアクリルパイプを固定し内部にヤスリ掛けを行いました．以下の項では使用した部品の詳細な説明をしていきます.

Table.1 使用部品

部品名	部品内容	型番	個数
抵抗	50[Ω]		1
可変抵抗	50[kΩ]		1
LED	色：Yellow 定格電流：60[mA]	OS5YKE56C1A	1
AD コンバータ		MCP3002	1
マイクアンプキット AE-MICAMP	マイクロホン(P-01810)， 基板（部品実装済）， リード線，制作資料	K-05757	1
RaspberryPi		Model B	1

3.1.1. RaspberryPi

RaspberryPi とは英国ラズベリーパイ財団が開発した名刺サイズのコンピュータです. RaspberryPi は低価格で持ち運びやすい教育用の小型コンピュータとして提供されており, コンピュータやプログラミングの基礎知識を学ぶ道具としてしばしば用いられています[2].

通常 RaspberryPi を使用するためには，画面出力用のディスプレイとそれを繋ぐ HDMI ケーブル，文字入力を行うための USB 接続のキーボートとマウス, それに加え起動用の電源が必要となります. 接続例は Fig.10 に示します．また，ハードディスクは搭載していないので，保存容量は別途取り付ける SD カードに依存します．OS は別途用意する SD カードにインストールし，搭載されている SD カードスロットにセットして起動させます．OS は Windows ではなく Linux の Debian を基にした Raspbian が一般的です.

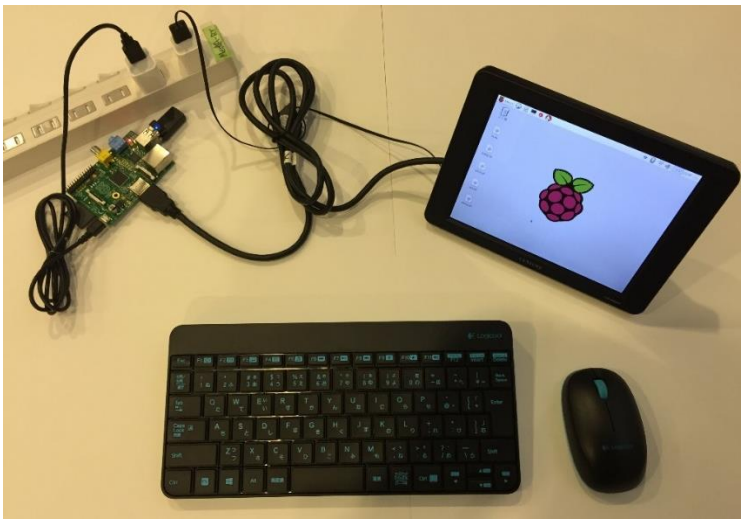

Fig.10 RaspberryPi と周辺機器

RaspberryPi を使用する際，SD カードに予め NOOBS の展開ファイルを配置しておく必要があります．NOOBS とは，RaspberryPi に Raspbian をインストールさせるためのファイルです．また，本体の初期設定とは別に RTM を使用するための環境設定を行わなければ, RTM を扱うことができないため注意が必要です.

さらに本作品では SPI 通信を行うため，予め SPI 通信を有効化させる必要もあります．SPI 通信の設定方法は 5 章 1 節 1 項に記述します．

本作品で使用する RaspberryPi の GPIO ピンは，GPIO17 の他に 5V，GND，さらに SPI 通信のための MOSI，MISO，SCLK，CE0 です．それぞれ 3 章 2 節の Fig.11 の回路図に示したように接続します．

3.1.2. マイクロホン

音の検出を行うセンサをマイクロホンと呼びます．マイクロホンはその変換方式の違いにより，動電型・圧電型・静電型などに分類されます．静電型ではコンデンサの原理を応用し，音声入力に応じて静電容量が変化することで音を電気信号として出力します．本作品では静電型マイクロホンを使用しています．また，マイクロホンは変換方式の他に指向特性や周波数特性などが異なり多くの種類があるため，目的にあったマイクロホンを選択する必要があります[3].

3.1.3. AD コンバータ

アナログ信号からデジタル信号への変換を AD 変換と呼び，AD 変換を行う回路・IC を AD コンバータと呼びます．AD コンバータにアナログ信号が入力されると，それを 2 進数の数値に変換します．このときの精度はそれぞれのコンバータの規格にある bit 数によって決まります．また，サンプリング間隔を外部から指定する必要があります．AD コンバータは変換方法によって複数の種類にわかれるため，扱う際に用途に合った種類を選択することが大切です[4].

3.2. 制作回路

Fig.11 は，本作品で制作した回路の回路図です．また，Fig.11 内部に用いられているマイクアンプキット[5]の回路図詳細を示したものが Fig.12 で，Fig.11 の回路に接続する時に使用する RaspberryPi のピン配置をまとめたものが Fig.13 です．

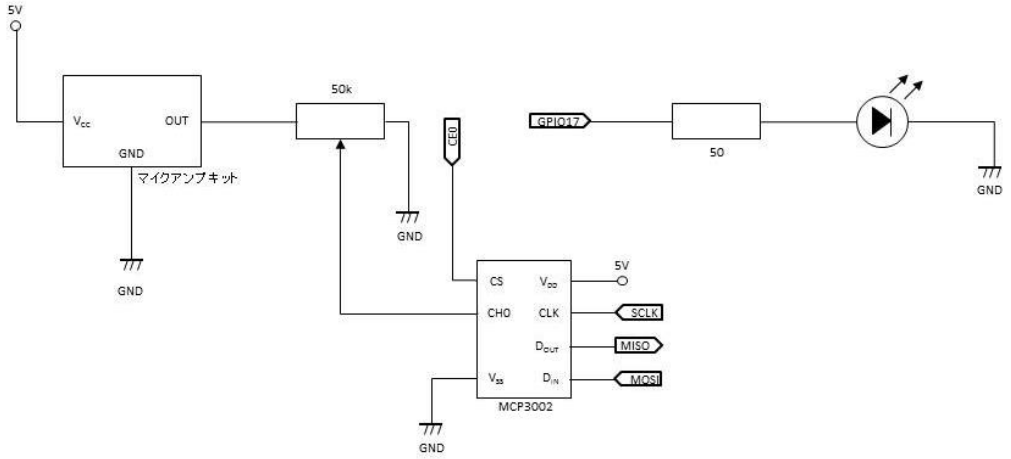

Fig.11 制作した回路

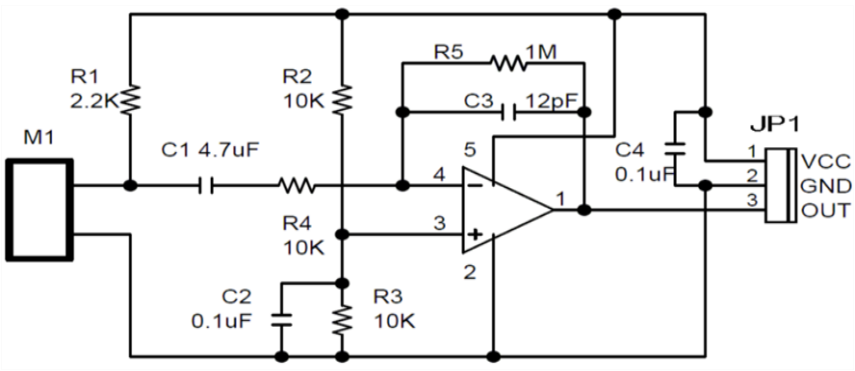

Fig.12 マイクアンプキット [5]

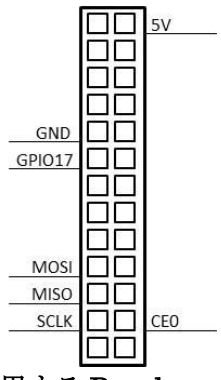

Fig.13 使用する RaspberryPi のピン

Fig.11 の回路は，センサから取得したデータを RaspberryPi に伝える回路と RaspberryPi の出力に応じて LED の点灯/消灯が行われる回路によって構成されています.

本作品では，マイクロホンを用いて周囲の音を検出し，検出されたデータに応じて LED の点灯/消灯状態の制御を行います. RaspberryPi ではアナログデータを受け取ることができないため, AD コンバータを使用し，AD 変換することでマイクロホンにより検出されたデータを RaspberryPi で受け取ることができます. 受け取ったデータに応じて RaspberryPi の出力を変化させることで LED の制御を行うことができ，本作品では GPIO17 の出力変化により LED の制御を行います.

4. ソフトウェア

4.1. 開発環境

本コンポーネント群は Windows 及び RaspberryPi にて動作確認を行いました. 開発環境は以下の通りです.

- Windows 10 (64bit)
  - RTM：OpenRTM-aist-1.1.1-RELEASE（C++, 32bit 版）
  - コンパイラ：Microsoft VisualC++ 2013 Express
  - CMake：CMake 3.2.1
  - python：python-2.7.9
  - PyYAML：PyYAML-3.11
  - doxygen：doxygen-1.8.9.1
- RaspberryPi Model B
  - NOOBS：1.9.2

詳しい開発環境設定につきましては，下記ページを参照してください.

- インストーラー：http://www.openrtm.org/openrtm/ja/node/5711
- 設定方法：http://openrtm.org/openrtm/ja/content/lets_start
- RaspberryPi：http://openrtm.org/openrtm/ja/content/RaspberryPi_openrtm_installation

4.2. 開発したコンポーネントの説明

4.2.1. RaspberryPi 上で起動するコンポーネント群

4.2.1.1. アナログデータ受取コンポーネント(AnalogInput)

AnalogInput は，RaspberryPi 上でセンサからのアナログデータ受取を可能にするコンポーネントです. 今回使用したセンサはマイクアンプキット[5]に付属しているマイクロホンで，息を検知するために使用しました．また，アナログデータ受取に使用した AD コンバータは MICROCHIP 社の mcp3002[6]です.

本来デジタルデータの受取しか行えない RaspberryPi でアナログデータを受け取るためには, AD コンバータを使用してデジタルデータをアナログデータに変換した後，SPI 通信で RaspberryPi に送信します．SPI 通信を行うためには RaspberryPi 上で設定を行う必要があります．SPI 通信の設定に関しては宮城大学の小嶋様のホームページを参考にしました．詳細は以下の URL をご参照ください.

http://www.myu.ac.jp/~xkozima/lab/raspTutorial3.html(参照日 2016/10/10)

本コンポーネントでは InPort は使用せず, OutPort のみで構成されています. 使用する OutPort は 1 つで, RaspberryPi 上のセンサから受け取ったアナログデータを外部に出力するための PORT となっています.

- InPort

無し

- OutPort

名称	型	説明
OutData	TimedDouble	センサから受け取ったデータを出力します.

- Configuration

無し

4.2.1.2. LED 揺らぎコンポーネント(PWMFluctuations)

PWMFluctuations は RaspberryPi に接続されている LED の光を火のように揺らがせるコンポーネントです．ここでは光の明るさを変えることでろうそくの火のような光の揺らぎ方をさせています．このプログラム上では PWM を用いて光の明るさを変えることで揺らぎを実装しています．また，configuration の Rate によって光の明るさの変化を速めたり，遅くしたりする事ができます．さらに，MaxBrightness と MinBrightness によって明るさの最大値，最小値の割合をパーセンテージで変えることができます．

本コンポーネントでは GPIO ピンの 17 番に LED を接続することを想定した実装となっております．違う番号の GPIO ピンを使用する場合は，PWMFluctuations.h の上部にある

#define PIN 17

の 17 を任意の数字に変更してください．

- InPort

名称	型	説明
inSW	TimedBoolean	LED の点灯/消灯を決めます．true：点灯，false：消灯．

- OutPort

無し

- Configuration

名称	型	デフォルト値	説明
Mode	int	1	LED の光らせ方を選択します． 0：通常点灯モード 1：揺らぎモード その他：消灯モード
MaxBrightness	double	80	明るさの最大値を決めます．大きくなればなるほど ON の時間が長くなります．
MinBrightness	double	30	明るさの最小値を決めます．小さくなればなるほど ON の時間が短くなります．
Rate	int	2	変化の速さを変えます．大きくすると速くなり，小さくすると遅くなります．

4.2.2. WindowsPC 上で起動するコンポーネント群

4.2.2.1. 閾値判別コンポーネント(ThresholdSwitch)

ThresholdSwitch は，コンポーネントに入力された値を参照し，閾値を超えたら true を出力するコンポーネントです．コンポーネントの概要を以下で説明します．

本コンポーネントは入力された値が閾値を超えたのかを判別する部分と，2 つの configuration で設定した値と前述の判別結果から，true か false のどちらを出力するのか判別する部分の 2 つに分かれています．

1 つ目の部分には最大値，最小値，平均値を手動で設定するモードと自動で設定するモードがあります．どちらの方法も平均値を基準として最大値，最小値をそれぞれ 100 パーセントとします．閾値はパーセンテージで ThresholdPercent として設定します. 手動で設定するモードでは入力の最大値, 最小値, 平均値を決め，それぞれ InportMax，InportMin，Average に設定します.自動で設定するモードでは最大値，最小値，平均値を入力された値から自動で算出します．

2 つ目の部分では，Reference，Frequency の 2 つの configuration の値を利用します．Reference は参照するデータの個数です．Frequency は true を出力するために参照データが閾値を超えなければならない回数です．つまり，Reference で設定した個数分データを参照し，その参照データが Frequency で設定した回数以上閾値を超えたら true を出力します．閾値を超えた回数が Frequency で設定した回数未満であれば false を出力します．例えば Reference の値が 5 で Frequency の値が 3 のときは，5 つのデータを参照し，そのうちの 3 つ以上のデータが閾値を超えた場合 true を出力します．初期状態では false を出力するようになっています．また，上記の 2 つの configuration の値を両方とも 1 に設定することで，入力された値が閾値を超えると true を出力し，閾値を超えなければ false を出力するという単純な仕様になります．

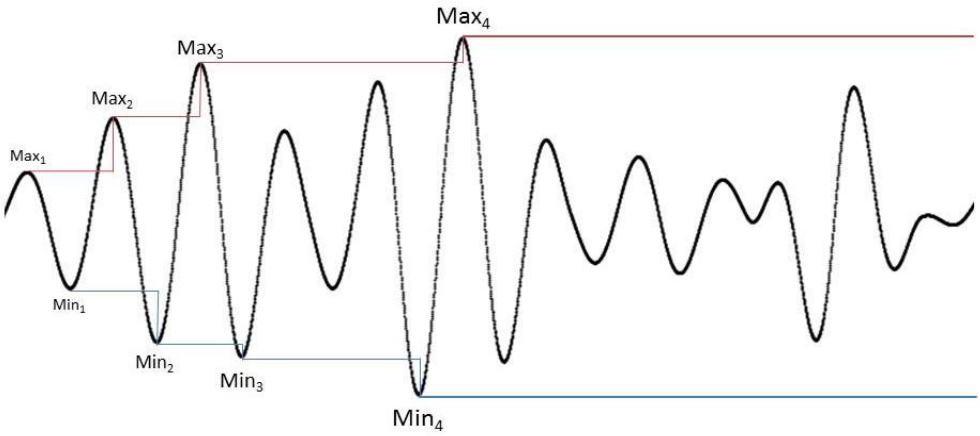

Fig. 14 自動設定モードでの最大値・最小値決定の仕組み

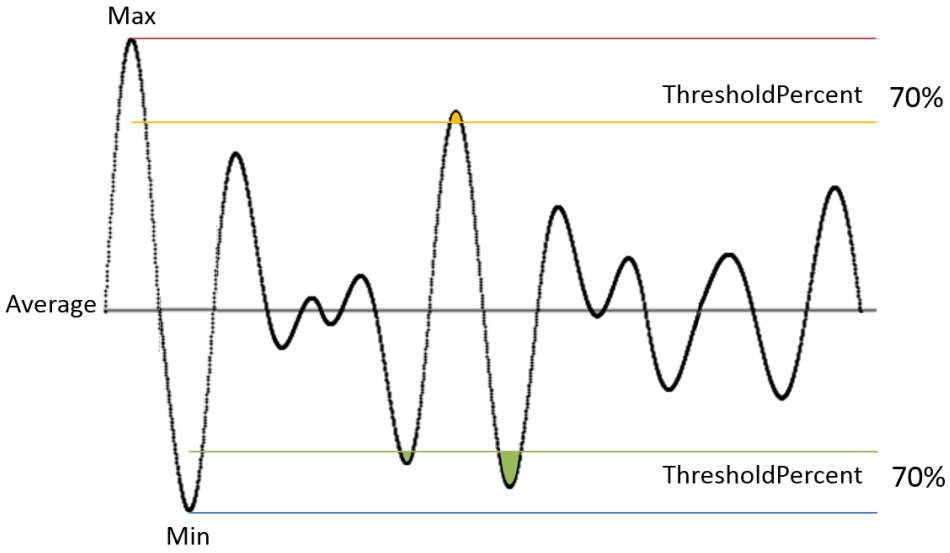

Fig.15 ThresholdPercent の仕組み

Fig.15 について，Max と Min は値が違うのですが，Average が 0 パーセント，Max が 100 パーセントとしたときの ThresholdPercent のパーセンテージを閾値としています．また，Average が 0 パーセント，Min が 100 パーセントとしたときの ThresholdPercent のパーセンテージも閾値としています. つまり閾値は Max 方向，Min 方向にそれぞれに 1 つずつ配置されています．これによって，値は違っても同じ比率で閾値を判別することができます.

- InPort

名称	型	説明
InportData	TimedDouble	入力されるデータの値です.

- OutPort

名称	型	説明
OutResult	TimedBoolean	入力されるデータと configuration の値から true または false を出力します.

- Configuration

名称	型	デフォルト値	説明
AutoMode	Int	1	自動で設定するモードにするのか手動で設定するモードにするのかを選択します. 1：自動設定モード その他：手動で設定するモード
Frequency	Int	3	true を出力するために閾値を超えなければならない回数です.
Average	Double	512	手動で設定するモードでのみ働きます．入力されるデータの平均値です.
InportMax	Double	1023	手動で設定するモードでのみ働きます．入力されるデータの最大値です.
InportMin	Double	-1	手動で設定するモードでのみ働きます．入力されるデータの最小値です.
ThresholdPercent	Double	90	平均値を基準として最大値，最小値を 100 パーセントとしたときの閾値のパーセンテージです.
Reference	Int	5	参照するデータの個数です.

4.2.2.2. LED 状態管理コンポーネント(LEDStateManager)

LEDStateManager は，接続された複数の LED の点灯/消灯状態を管理し，指定の方法で連動させるコンポーネントです．今回は 4 つのモードを指定できるようにしました．以下ではコンポーネントの動作概要を説明します.

まず Activate 時に InPort と OutPort を管理しやすいように配列に代入し，全て配列に初期値として false を代入します．この時，入力値判別に使用するフラグも初期化しておきます．デフォルトでは 4 つの Magicalight を接続させることを想定していますが，configuration の値を変えることで InPort と OutPort の数を変更することができます.

Execute 時は常に InPort の状態を確認し，新規データが入力されたらフラグの確認を行います．この時フラグが立っていたら（フラグの値が true なら）出力データの更新を行います．出力データの更新は現在 configuration で指定されているモードに則って行い，更新が終了したらフラグを下げます．フラグが立っていなかったら（フラグの値が false なら）フラグを立てます．フラグを作ることにより，入力されたデータが前回と同一だった際の出力データ更新にかかる時間を短縮することができます.

- InPort

名称	型	説明
in* (*は接続する LED の数によって変わります)	TimedBoolean	現在の LED の状態を受け取ります. true：点灯，false：消灯. 接続される LED の数が 2 つの時の例を示します. InPort は 2 つで，名称は in0，in1 とします. in0，in1 に接続されている LED の番号をそれぞれ 0 番，1 番とします．この時の番号は configuration の BaseUnitNum で設定する親機の番号の候補になります.

- OutPort

名称	型	説明
out* (*は接続する LED の数によって変わります)	TimedBoolean	連動処理後の LED の状態を送ります. true：点灯，false：消灯. 接続される LED の数が 2 つの時の例を示します. OutPort は 2 つで，名称は out0，out1 とします. out0 には 0 番の LED, out1 には 1 番の LED を対応させて接続させてください.

- Configuration

名称	型	デフォルト値	説明
Mode	int	0	値に応じて LED の動作が変わります. 0：連動モード 1：イエスマンモード 2：ランダムモード その他：非連動モード
BaseUnitNum	int	0	親機の番号を指定します. 0 は InPort の in0 に接続されている LED を指します.
inportNum	int	4	InPort の数を設定します.
outportNum	int	4	OutPort の数を設定します.

5. コンポーネント群を起動する前に

本作品は Fig.16 のように各コンポーネントを接続することにより機能します．しかし，通常の設定しか行っていない RaspberryPi では私たちが開発したコンポーネントは起動できません．また，息を吹きかけられた時の判別を行うコンポーネントである ThresholdSwitch は，接続されている Magicalight の数だけ起動してください. 以下では, RaspberryPi での必要な設定及び ThresholdSwitch の複数起動の方法を説明します.

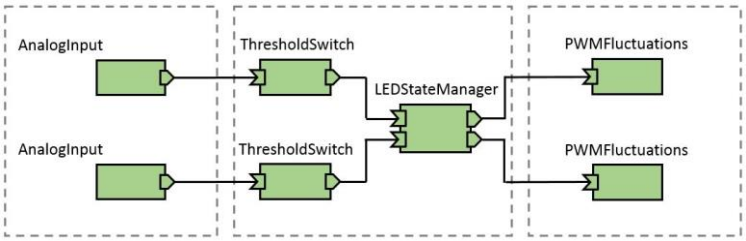

Fig.16 コンポーネント図

5.1. RaspberryPi の設定

AnalogInput と PWMFluctuation を起動するためには，RaspberryPi で WiringPi というライブラリをインストールする必要があります. WiringPi をインストールすると C 言語で容易に RaspberryPi の GPIO ピンを使用することができるので非常に便利です．また，RaspberryPi で AD コンバータによるアナログデータの受取を行うには，SPI 通信を有効化する必要があります．以下の項では，WiringPi のインストール方法と SPI 通信の有効化の方法を説明します. ここでは RaspberryPi に専用 OS である Raspbian がインストールされているものとします.

5.1.1. SPI 通信の有効化の方法

まず初めに RaspberryPi 上で terminal を起動します．次に

$sudo raspi-config

でコンフィギュレーション設定画面を出します．そこで

Advanced Options

を選択します．次に

SPI

を選択し，「はい」を選択すると，SPI 通信を有効化できるようになります.

$sudo vi /etc/modules

でエディターを起動し，modules の一番下の行に

Spidev

と記述します．最後に RapsberryPi を再起動して SPI 通信の有効化は終了です.

5.1.2. WiringPi のインストール方法

まず初めに RaspberryPi 上で terminal を起動します．次に

$sudo apt-get update

でパッケージリストを入手し

$sudo apt-get upgrade

でパッケージを最新版にします．そのあと RaspberryPi 上のブラウザで

https://git.drogon.net/?p=wiringPi;a=summary（参照日　2016 年 10 月 22 日）

にアクセスし，WiringPi のインストーラーをダウンロードします．ダウンロードは「snapshot」を押すと始まります．詳しくは Fig.17 をご参照ください．

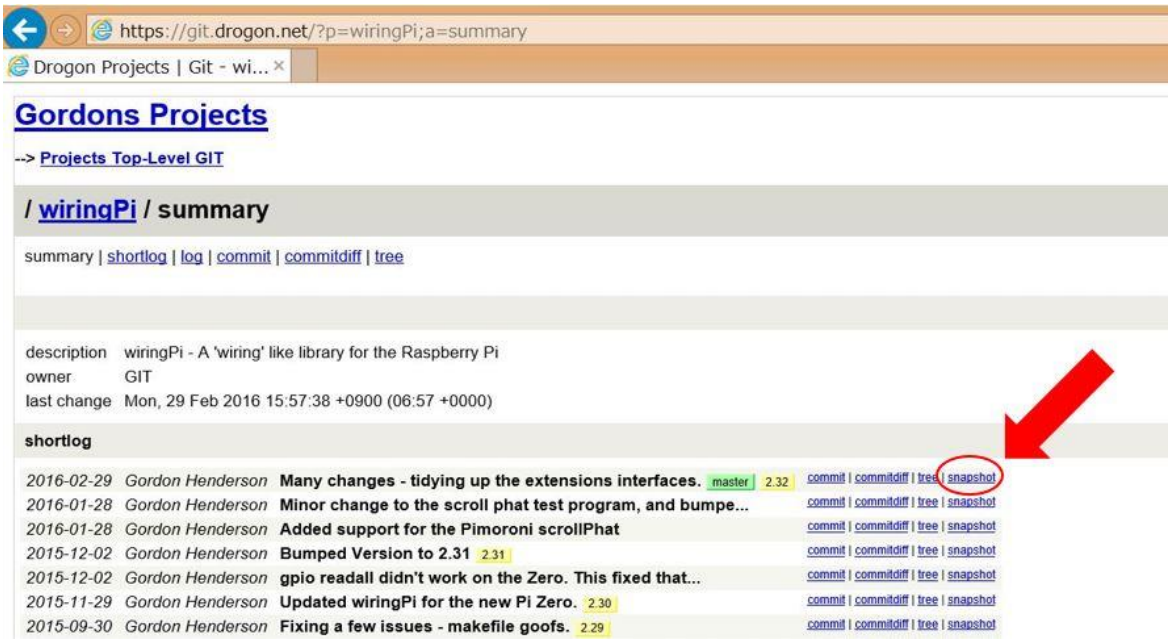

Fig.17 wiringPi のダウンロードページ

インストーラーのダウンロードが完了したら，ファイルを解凍すると「wiringPi-〇〇〇」（〇〇〇は version 名）というフォルダが展開されます．そのフォルダを「wiringPi」という名前に変えて pi の直下においてください（home/pi/wiringPi というディレクトリになるようにしてください）．次に

$cd home/pi/wiringPi

でカレントディレクトリを移動させて，

$sudo ./build

で WiringPi をビルドします．これで RaspberryPi に WirinigPi をインストールすることができました．

5.2. Windows の設定

ここでは ThresholdSwitch を複数起動させる方法を説明します．ThresholdSwitch は 4 つ同時起動用に準備されています．ThresholdSwitch/build/src/Debug を見ていただけると分かりますが，rtc_0.conf，…rtc_3.conf という名前の 4 つのファイルがあると思います．rtc_0.conf には

corba.nameservers: localhost

naming.formats: %h.host_cxt/%n_0.rtc

と記述されています．これはネームサーバに，このコンポーネントは「ホスト名_0」という名前で起動していると認識させるためのファイルです. 通常一つのネームサーバでは同名のコンポーネントは起動できませんが，起動する際に先ほどの conf ファイルをオプションで参照することによって，ネームサーバにそれぞれ別のコンポーネントを起動していると認識させることができます．ファイルの参照はコマンドプロンプトで

¥ThresholdSwitch までのパス¥build¥src¥Debug¥ThresholdSwitchComp.exe

-f ¥ThresholdSwitch までのパス¥build¥src¥Debug¥rtc_0.conf

と入力することで行えます. 上記のコマンドでは rtc_0.conf を参照して ThresholdSwitch が起動しています.

この手順を参照するファイルの番号を変えて行うことで，同じコンポーネントの複数起動が行えます．ただ，毎度コマンドプロンプトでコマンドを入力するのは大変なのでバッチファイルを作成することを推奨します．バッチファイルの作り方は，まず初めにテキストエディター（メモ帳，秀丸等）で「○○○（任意の名前）.bat」というファイルを作成します．そのファイルに

```
SET COMP_PATH=（ThresholdSwitch までのパス）¥build¥src¥Debug¥

SET TARGET_EXE0="%COMP_PATH%ThresholdSwitchComp.exe
 -f %COMP_PATH%rtc_0.conf"
        ・・・(省略)・・・
SET TARGET_EXE3="%COMP_PATH%ThresholdSwitchComp.exe
 -f %COMP_PATH%rtc_3.conf"

START "ThresholdSwitch_0" "%TARGET_EXE0%"
        ・・・(省略)・・・
START "ThresholdSwitch_3" "%TARGET_EXE3%"
```

と記述することで，今回は 4 つ同時起動の例ですが，バッチファイルの完成です．このファイルをダブルクリックするだけで，ThresholdSwitch の 4 つ同時起動が行えます．

5 以上の同時起動を行いたい場合も，任意の数 rtc_(任意の数字).conf を作成し，そのファイルを参照するようにバッチファイルを書き換えることで容易に可能になります．

また，今回は github に magicalight.bat という名前でバッチファイルをアップロードしておきますので，参考にしていただけると幸いです．このバッチファイル内では ThresholdSwitch のパスは

C:¥workspace¥ThresholdSwitch

となっておりますので，コンポーネントを起動する際にお気を付けください．

6. 参考文献

[1] アクリ屋ドットコム,<http://www.acry-ya.com/>(参照日 2016/10/10).
[2] 「RASPBERRYPI とは」,< http://raspi.uassist.co.jp/>(参照日 2016/10/17).
[3] 小野測器,「音とそのセンサについて」,
<https://www.onosokki.co.jp/HP-WK/c_support/newreport/sound/soundsensor_1.htm>
(参照日 2016/10/17).
[4] 佐藤幸男(1999)『信号処理入門』株式会社オーム社
[5] 秋月電子通商,「高感度マイクアンプキット」,
< http://akizukidenshi.com/download/ds/akizuki/AE-MICAMP.pdf>
(参照日 2016/9/26).
[6] MICROCHIP,「MCP3002」,< http://www.microchip.com/wwwproducts/en/MCP3002>
(参照日 2016/11/1).

7. お問い合わせ

本コンポーネントにつきましては，まだまだ改善の余地があるものと考えております．バグ報告やマニュアルの記述の不備等ございましたら，お手数ですが下記ホームページのメールフォームよりご連絡ください．

MediA-RT：http://media-rt.weebly.com/