株式会社 ビズライト・テクノロジー

BH3-FA
テクニカルノート

最終更新日: 2019/12/16
バージョン: 1.0

（株）ビズライト・テクノロジー

1. 本製品について

この度は BH シリーズ、BH3-FA をご購入いただき、誠にありがとうございます。
本製品は下記を組み込んだセットとなっております。

・Raspberry Pi3 Model B
・(株)ビズライト・テクノロジー　オリジナルHAT基板
・冷却性の高いアルミ筐体ケース(放熱用 FAN＆ヒートシンク)
・OSインストール済み microSD カード

※ACアダプタ、もしくは、電源ユニットを別途ご用意ください。

2. 付属品について

・DIN レールアダプタ×1
・DIN レールアダプタ取り付け用ネジ×4　※1
・ゴム足×4
・FA用途向けセットアップ済み microSD カード　※2 （本体に装着済みです。）

※1. DIN レールアダプタは、必ず付属の取り付け用ネジを使用してください。
長さの異なるネジで取り付けた場合、内部の基板や実装部品とショートする恐れがあります。

※2. BH3-FA では、microSD カードの標準添付品として、サンディスク「Industrial」 シリーズを採用しています。

3. BH3-FA の概要

BH3-FA は Raspberry Pi3 Model B をFA向けにカスタマイズした製品です。
オリジナルのHAT基板を搭載し、電源の安定化や放熱のためのFAN制御など、機能性もアップしております。
Raspberry Pi は特に IoT 分野などに多く使われはじめております。しかし、プロトタイプを終えて実際にフィールドで利用するためには、システム時刻保持、電源の安定化、耐ノイズ、放熱、ウォッチドックタイマーなどクリアしなければならない問題がたくさん出てきます。
BH3-FA は特に堅牢性に重点を置き、フィールドでも使える STB を目指して開発されました。

4. 機能説明

(1)RTC(リアルタイムクロック)

データロギングなど大抵のアプリケーションには、正しいシステム時刻が必要です。BH3-FA はバッテリでバックアップされた RTC を搭載しています。

(2)電源供給の安定化

Raspberry Pi は通常 MicroUSB コネクタから電源を供給しますが、消費電流が増加したときに、このコネクタの内部抵抗が問題になり、電源が不安定になることがありました。BH3-FA は I/O ポートから電源を供給していますので、この問題が発生しません。内部に DC/DC コンバータを搭載し、安定した動作電圧を確保しています。

(3)CPU の放熱対策

BH3-FA は CPU にヒートシンクを装着し、冷却用ファンや熱伝導シートを搭載することにより、放熱対策をしています。55℃の外部環境での動作テストをクリアしていますので、フィールドで安心してご利用になれます。

(4)ノイズ、静電対策

アルミケースの採用や、グランド処理などにより、静電気、雷サージ、GPIO ラインノイズなどの対策を施しています。

(5)ウォッチドッグタイマー

Raspberry Pi に搭載されたウォッチドッグタイマーを有効にし、万が一 CPU が暴走した場合でも、ハードウェア的に CPU をリブートさせます。システムがハングアップしたままになる可能性を減らすことが可能です。

(6)GPIO

GPIO を利用したプログラミングで、LED、ブザーをはじめ、外部に接続した機器をカスタマイズ制御することができます。使用可能なピンについては、「8. I/O コネクタへのアクセスと GPIO ポート」を参照してください。

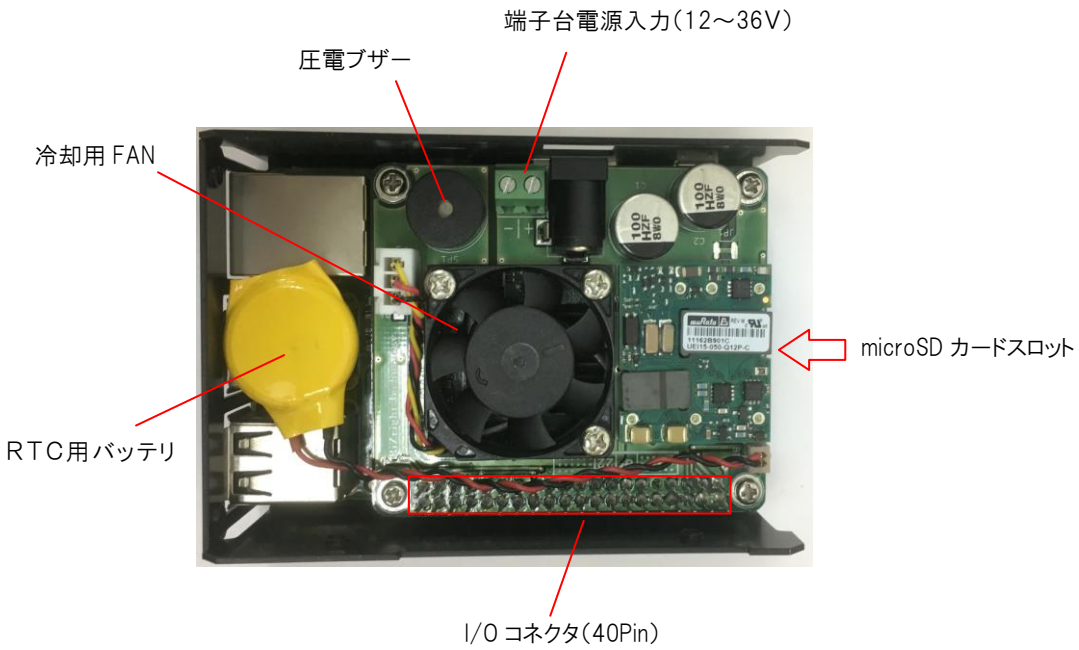
(7)SD カードの長寿命化

耐久度が高い SD カードを採用し、NAND フラッシュメモリ向けに最適化された f2fs ファイルシステムで運用することで、SD カードの長寿命化を実現します。

5. ハード仕様

OS	Raspbian Stretch Desktop BH3-FA セットアップツール
CPU	Raspberry Pi3 Model B 1.2GHz / 64-bit quad-core ARMv8 CPU
GPU	250 MHz / Broadcom VideoCore IV
メモリ	LPDDR2 SDRAM　1GB
RTC	NXP PCF2127T I2C 接続（GPIO2、3 を使用）
ビデオ出力	HDMI(1.3 / 1.4)
音声出力	3.5mm ジャック / HDMI
USB ポート	USB2.0×4
I/O コネクタ	40 ピン（ピンアサインは Raspberry Pi3 と同様）
ネットワーク	10 / 100Mbps イーサネット 802.11n Wireless LAN
Bluetooth	Bluetooth 4.1 Bluetooth Low Energy (BLE)
カードスロット / ストレージ	Micro SD カードスロット ※16GB　OS セットアップ済み SD カードを付属
電源	別売。 推奨電源は下記の通り。 ・端子台入力（24W） 12V（2A） ～ 36V（0.6A） ・AC アダプタ入力（24W） 12V（2A） ～ 15V（1.5A。DC ジャックの耐圧による制限） ・microUSB 端子入力 （15W） 5V(3A)
外寸	高さ 35mm×幅 62mm×奥行き 93mm
重量	150g
動作保障温度	-10 度～55 度
動作保障湿度	10%～80%（非結露）

6. 各部名称



Micro SD カードは、上カバーを開けて、Raspberry Pi に直接挿します。

7. 電源について

(1)電源供給について

※ACアダプタと端子台は内部で直結されていますので、両方同時に接続しないようにしてください。

①電源入力端子台からの電源供給(24W)

入力:12V 2A ～ 36V 0.6A

②AC アダプタからの電源供給(24W)

ジャック:Φ2.1 センタープラス

入力:12V 2A ～ 15V 1.5A (DC ジャックの耐圧による)

③microUSB 端子からの電源供給(～15W)

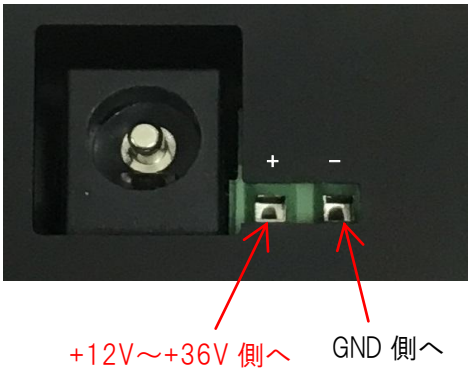
入力:5V Max3A

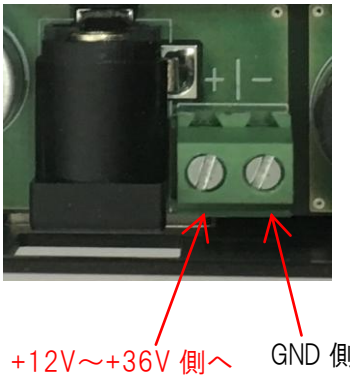
(2)外部電源取り付け手順(端子台を使用する場合)

①ネジ 3 か所を外し、上カバーを取り外します。

上カバーは、横方向にスライドさせることで、簡単に外せます。

②外部電源が OFF になっていることを確認し、極性に注意してケーブルを取り付けます。端子台のねじ穴を傷めないよう、精密ドライバーをご使用ください。

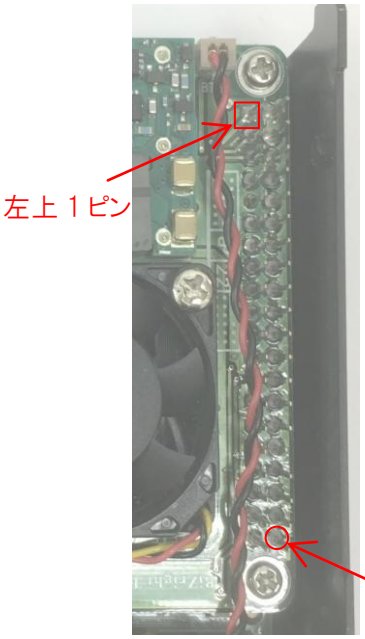




③本体カバーを取り付け、外部電源の出力設定が 12V～36V になっていることを確認してから、電源を入れてください。

8. I/O コネクタへのアクセスと GPIO ポート

上カバーをはずすと、I/O コネクタにアクセスできます。
GPIO ポート番号は下図のようになっております。



1	+3.3V			+5V	2
3	RTC_SDA(GPIO2)			+5V	4
5	RTC_SCL(GPIO3)			GND	6
7	GPIO4(GCLK)			GPIO14(TXD)	8
9	GND			GPIO15(RXD)	10
11	GPIO17(GEN0)			GPIO18(GEN1/PWM0)	12
13	GPIO27(GEN2)			GND	14
15	GPIO22(GEN3)			GPIO23(GEN4)	16
17	+3.3V			GPIO24(GEN5)	18
19	GPIO10(SPI0_MOSI)			GND	20
21	GPIO9(SPI0_MISO)			BUZZER_OUT(GPIO25)	22
23	GPIO11(SPI0_SCK)			GPIO8(SPI0_CE_N)	24
25	GND			GPIO7(SPI1_CE_N)	26
27	ID_SD			ID_SC	28
29	GPIO5			GND	30
31	FAN_OUT(GPIO6)			LED_OUT(GPIO12)	32
33	FAN_PWM(GPIO13)			GND	34
35	GPIO19(SPI1_MISO)			GPIO16	36
37	FAN_PULSE(GPIO26)			GPIO20(SPI1_MOSI)	38
39	GND			GPIO21(SPI1_SCK)	40

9. BH3-FA のピンアサインについて

BH3-FA では、以下のピンを利用しています。(回路上でつながっています。)

対象	信号名	PIN 番号	GPIO 番号	備考
汎用 LED	LED_OUT	32	12	**OUTPUT:** HIGH で LED が点灯
ファン	FAN_OUT	31	6	**OUTPUT:** HIGH でファンが回転
	FAN_PULSE	37	26	**INPUT:** ファンの回転信号を検知
	FAN_PWM	33	13	現在未使用(未接続状態)
RTC (PCF2127T)	RTC_SDA	3	2	RTC(PCF2127T)設定のための I2C 通信
	RTC_SCL	5	3	
ブザー	BUZZER_OUT	22	25	**OUTPUT:** HIGH で鳴動

10. LED について

(1)右側:電源 LED (色: 赤)

電源が供給されている場合、常に点灯します。

(2)左側:汎用 LED (色: 赤)

ユーザープログラムにより制御可能な 汎用 LED が 1 つ搭載されています。

汎用 LED は GPIO 12 に接続されています。

11. ファンについて

Raspberry Pi の仕様により CPU 温度が上昇すると段階的にクロック数が低下するため、
主に CPU を冷却する用途で使用されます。

OS起動時の動作確認を除き、ファンの稼働条件を満たさない場合は常に停止しています。
FAN_PULSE (GPIO26)によりファンの回転数を計測しており、異常を検知した場合は断続的にブザーが鳴ります。

ファンの稼働条件(工場出荷時)
・CPU の温度が 60.5℃ 以上

12. ブザーについて

自励式ブザーが搭載されています。
BH3-FA の OS 起動時や稼働中に FAN_PULSE (GPIO26) 信号を検出し、ファンの回転数を計算していますが、ファンの回転数が異常だった場合に断続的にブザーが鳴ります。

13. 電源の入れ方と初期設定

本機に電源スイッチはありません。
Raspbian Stretch OS に FA 用途向けのセットアップを行った microSD カードが装着されています。
ACアダプタや端子台からの電源供給が開始されると電源LEDが点灯し、そのまま OS が起動します。
※AC アダプタと端子台は内部で直結されていますので、両方同時に接続しないようにしてください。

OS 起動中に、FAN動作のチェックのため数秒間FANが回転します。
工場出荷時設定では、デスクトップモードで起動しますので、必用に応じて wifi などのネットワーク設定や SSH の設定をしてください。
RaspberryPi についての詳しい情報は、インターネットで情報を集めたり、「RaspberryPi 公式ドキュメント」をご確認ください。

※工場出荷時のユーザー名は「pi」、パスワードは「raspberry」となっております。

14. FA 用途向けセットアップ済み OS について

下記の機能が、FA用途向けにインストールされています。

- CPU 温度を監視し、自動的にファンをコントロール
- ファン回転数異常時にブザーで警告
- OS 起動時に RTC 時刻をシステム時刻に設定
- chrony を利用して現在時刻を取得し定期的に RTC に保存
- ウォッチドッグタイマーの有効化

工場出荷時の SD カードイメージにリカバリーしたい場合、下記のページから SD カードイメージをダウンロードできます。
また、お客さまご自身でクリーンインストールし、FA用途向け OS をセットアップすることも可能です。

https://bh.bizright.co.jp/bh/bh3-fa.html

15. 追加されるパッケージ・ファイル

追加されるパッケージおよびファイルは以下の通りです。

パッケージ	用途
i2c-tools	I2C 通信ライブラリ
chrony	NTP クライアント/サーバ
f2fs-tools	f2fs ファイルシステムツール
watchdog	ウォッチドッグタイマー
anacron	日単位のスケジュール実行
xdotool	GUI 操作自動化ツール
unclutter	マウスカーソル表示/非表示切り替えツール
ファイルパス	用途
/usr/lib/systemd/system/bh-init-rtc.service	RTC 初期化サービス
/usr/lib/systemd/system/bh-buzzer.service	ブザーサービス
/usr/lib/systemd/system/bh-fan-state.service	ファン状態サービス
/usr/lib/systemd/system/bh-fan-speed.service	ファン回転数サービス
/usr/lib/systemd/system/bh-healthd.service	状態管理サービス
/usr/bin/bh-init-rtc	RTC の初期化
/usr/bin/bh-timekeeper	時刻を正確にする
/usr/bin/bh-buzzer	ブザーを鳴動
/usr/bin/bh-fan-state	ファンを回転/停止
/usr/bin/bh-fan-speed	ファンの RPM を取得
/usr/bin/bh-healthd	状態の管理とログ出力

FA用途向けセットアップで変更している設定

工場出荷時設定では、以下の様に設定を変更しています。

(1)I2C

RTC と I2C 通信するための設定です

内容	設定ファイル	工場出荷時設定
I2C の有効化	/boot/config.txt	dtparam=i2c_arm=on
	/etc/modules	i2c-dev
RTC(PCF2127) の I2C 接続	/boot/config.txt	dtoverlay=i2c-rtc,pcf2127
I2C 通信の設定	/boot/config.txt	dtparam=i2c_baudrate=20000

(2)timesyncd

RaspberryPi デフォルトの時刻同期の仕組みですが 、FA 用途向けセットアップで追加される chrony(NTP クライアント/サーバ)と競合するので無効にします

内容	工場出荷時のコマンド設定
timesyncd を無効	$ sudo systemctl disable systemd-timesyncd

(3)fake-hwclock

デフォルトでは fake-hwclock の日時を過去に設定することが出来ないので、設定できるように変更します

内容	設定ファイル	工場出荷時設定
日時を過去に設定できるように変更	/etc/default/fake-hwclock	FORCE=force
	/lib/systemd/system/fake-hwclock.service	ExecStop=/sbin/fake-hwclock save $FORCE

(4) crontab

時刻を正確に保つためのスクリプトを定期的に実行する設定です。

工場出荷時設定では、bh-timekeeper を 5 分ごとに実行します。

内容	root ユーザーの cron の設定(工場出荷時)
bh-timekeeper を追加	*/5 * * * * /usr/bin/bh-timekeeper

変更例

bh-timekeeper を 10 分ごとに実行する場合、ターミナルから下記を実行します。

```
$ sudo crontab -e
```

エディターが立ち上がりますので、

```
*/10 * * * * /usr/bin/bh-timekeeper
```

と書き換えて保存します。

crontab: installing new crontab と表示され、設定が有効になります。

※上記のコマンドを実行すると、

```
Select an editor.  To change later, run 'select-editor'.
  1. /bin/ed
  2. /bin/nano        <---- easiest
  3. /usr/bin/vim.tiny

Choose 1-3 [2]:
```

と表示される場合があります。

そのまま Enter キーを押して nano エディターを使用するか、任意のエディターを選んでください。

(5)bh-healthd

ファンの動作条件や CPU 温度をチェックする間隔、ブザーの鳴動間隔を変更できます。

内容	工場出荷時設定
ファンを ON にする CPU の温度	FAN_ON_THRESHOLD = 60.5
ファンを OFF にする CPU の温度	FAN_OFF_THRESHOLD = 58.0
ファン ON を継続させる最低時間(0.1 秒単位)	FAN_ON_SEC_MIN = 12.0
ファン OFF を継続させる最低時間(0.1 秒単位)	FAN_OFF_SEC_MIN = 5.0
ファンの回転を検知する回転数の設定	FAN_ON_RPM = 6000
ファンの停止を検知する回転数の設定	FAN_OFF_RPM = 2000
CPU の温度をチェックする間隔 (0.1 秒単位)	CPU_TEMP_INTERVAL_SEC = 3.0
ブザーの鳴動間隔 (0.1 秒単位)	FAN_BUZZER_INTERVAL_SEC = 1.0

bh-healthd は、Python 言語で書かれています。

設定したい項目を直接書き換えて保存し、BH3-FA を再起動します。

```
$ sudo nano /usr/bin/bh-healthd
$ sudo reboot
```

16. RTC (PCF2127T) について

オフライン環境でも時刻を正確に保つため、温度補償のある RTC が搭載されています。

(1)温度補償の範囲と精度

温度補償の範囲は −30℃ ～ +80℃、精度は ±3ppm (月差±8 秒 相当) です。

(2)状態確認

①動作中か否か

RTC の動作を確認するには、以下のコマンドを実行します

```
$ sudo hwclock
```

エラー無く日時が表示されれば動作中です。

②内部設定値の確認

内部設定値は I2C 通信で確認することができます。PCF2127 のI2Cアドレスは「0x51」です。

以下では i2cget コマンドで確認する方法を記載します。実行には root 権限が必要です。

詳しい仕様は PCF2127 のマニュアルを参照ください。

準備: RTC のドライバをアンロードする

```
$ sudo modprobe -r rtc_pcf2127
```

例: CLKOUT control register (0x0f) を読み出す

```
$ sudo i2cget -y 1 0x51 0x0f c
```

→ 初期化されていれば、8 ビットを 16 進表示した「0x87」が表示されます。

※本機は、回路上 CLKOUT と接続されていませんので、下位 3bit(CLKOUT frequency (Hz))を 111 (CLKOUT = high-Z)に設定し、消費電力を削減しています。

詳しくは PCF2127 のマニュアルを参照ください。

確認が終わったら RTC のドライバをロードしなおす

```
$ sudo modprobe rtc_pcf2127
```

(3)初期設定と設定値の変更方法について

初期設定は以下のようになっています

内容	初期設定値	備考
RTC の TCXO 温度を測定する間隔	1 分	4 分、2 分、1 分、30 秒から選択

設定を変更する場合:

準備: RTC のドライバをアンロードする

```
$ sudo modprobe -r rtc_pcf2127
```

例: RTC の TCXO 温度の測定間隔を 30 秒にする。

```
$ sudo i2cset -y 1 0x51 0x0f 0xc7
```

→0xc7 の部分を下記のように変更することで、測定間隔を変更できます。

0x07:4 分、 0x47:2 分、 0x87:1 分、 0xc7:30 秒

設定が終わったら RTC のドライバをロードしなおす

```
$ sudo modprobe rtc_pcf2127
```

(4)NTP との連携動作

時刻調整スクリプトが NTP サーバのオンライン/オフラインを判定し、NTP と RTC が連携して可能な限り時刻を正確に保ちます。

①オンラインの場合

NTP サーバの時刻を基準として、システムクロックの時刻と RTC の時刻を調整します。

②オフラインの場合

RTC の時刻を基準として、システムクロックの時刻を調整します。

17. ウォッチドッグタイマーのセットアップ内容

工場出荷時設定で、ウォッチドッグタイマーを有効化しています。
OS 起動時に bcm2835_wdt モジュールを自動的にロードします。
下記の設定ファイルを変更すると、設定をカスタマイズすることができます。

```
$ sudo nano /etc/watchdog.conf
```

(1)工場出荷時設定:
max-load-1= 24
watchdog-timeout = 10

(2)設定項目の内容:
max-load-1 → このロードアベレージを超えたらリセット
watchdog-timeout → ウォッチドッグがタイムアウトと判定する時間

18. 汎用 LED を利用したプログラムのサンプルソース

BH3-FA 本体の汎用 LED を制御するプログラムのサンプルソースです。
このサンプルソースは、以下からダウンロードできます。
http://dl.bizright.jp/bh/bh3fa_sample.zip

(1)C 言語で Wiring Pi を利用した場合:
bh3faLED.c

```
#include <stdio.h>
#include <wiringPi.h>

#define GPIO_LED 12

int setupGpio();

// メイン関数
int main(int argc, char *argv[]) {

        // GPIO の初期化
        if (setupGpio() == -1) return 1;

        while (1) {

                // 汎用 LED を点灯
                digitalWrite(GPIO_LED, 1);

                // 0.5 秒スリープ
                delay(500);

                // 汎用 LED を消灯
                digitalWrite(GPIO_LED, 0);

                // 0.5 秒スリープ
                delay(500);

        }
```

```
        return 0;
}

// GPIO の初期化関数
int setupGpio() {
        // WiringPi の初期化
        if (wiringPiSetupGpio() == -1) return -1;

        // 出力モードに設定
        pinMode(GPIO_LED, OUTPUT);

        return 0;
}
```

ビルドコマンド:

```
$ gcc -lwiringPi -o bh3faLED bh3faLED.c
```

実行コマンド:

```
$ ./bh3faLED
```

(2)Python で RPi.GPIO を利用した場合:

bh3faLED.py

```
# coding: UTF-8

import RPi.GPIO as GPIO
import time

GPIO_LED = 12

#  出力モードに設定
GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_LED, GPIO.OUT)

try:
        while 1:

                #  汎用 LED を点灯
                GPIO.output(GPIO_LED, 1)

                # 0.5 秒スリープ
                time.sleep(0.5)

                #  汎用 LED を消灯
                GPIO.output(GPIO_LED, 0)

                # 0.5 秒スリープ
                time.sleep(0.5)

finally:
        # GPIO の設定をリセット
        GPIO.cleanup()
```

実行コマンド:

```
$ python bh3faLED.py
```

19.改訂履歴

更新日	バージョン	内容
2019/12/16	1.0	初版