株式会社 ビズライト・テクノロジー

BiZduino
テクニカルノート

最終更新日: 2017/04/04
バージョン: 1.1

（株）ビズライト・テクノロジー

1. BiZduino の特徴

この度は BH シリーズ、BiZduino をご購入いただき、誠にありがとうございます。
BiZduino は、長い実績を持ち世界中で使われている 8Bit マイコンボード「Arduino (UNOR3)」の上位互換機です。
CPU も Arduino と同じ ATmega328P を採用し、基板形状やサイズも合わせています。このためこれまで開発されてきた Arduino のスケッチ(アプリケーションプログラム)だけではなく、多くのシールド(オプションボード)もそのまま利用することができます。これに加えて Wi-Fi(※1)や RTC(※2)など Arduino では外付けにするしかなかった重要モジュールを標準搭載しているので、単なる試作や実験機ではなく実用的なシステムをこれ 1 枚で構築することが可能です。

※1　Wi-Fi モジュールとして Espressif Systems 社製 ESP-WROOM-02 を搭載しています。
※2　RTC モジュールとして MAXIM 社製 DS1388 を搭載しています。
RTC 用リチウムコイン電池は CR2032 をご利用ください。

2. BiZduino の 3 つの動作モード

BiZduino には 3 つの動作モードがあります。

(1) Arduino 互換モード
Wi-Fi モジュールは切り離され、ATmega328P のみで動作します。

(2) Wi-Fi モジュール専用モード
ATmega328P は切り離され、Wi-Fi モジュール単独で動作します。ファームウェアの書き換えも可能です。

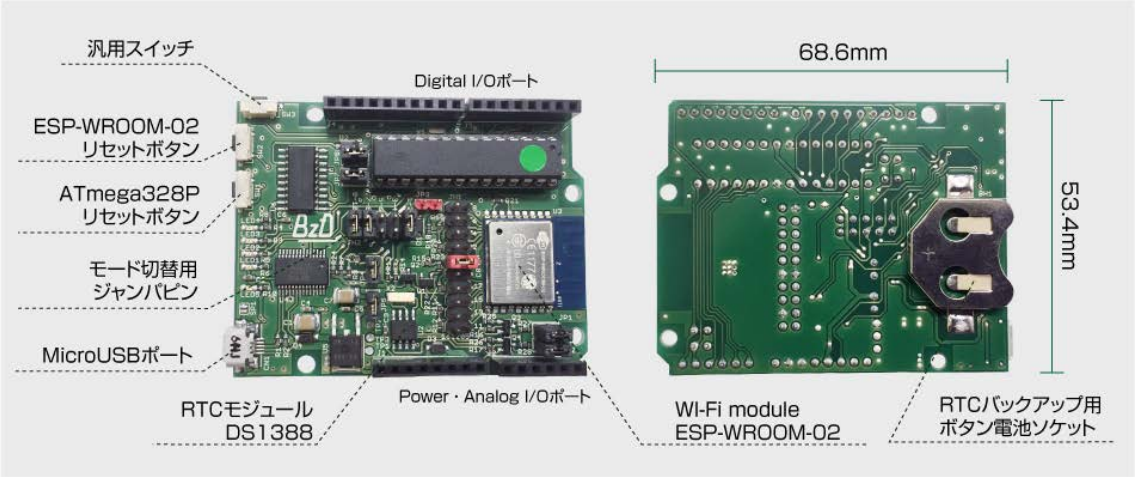
(3) Arduino 互換 + Wi-Fi モード
ATmega328P から Wi-Fi モジュールを AT コマンドで制御し、シリアルモニタから両方の TXD の監視が可能です。

これらはピンヘッダ(JH2)のジャンパーピンの組み合わせと、スケッチ(ソフトウェア)からのポートアクセスによって実現されています。

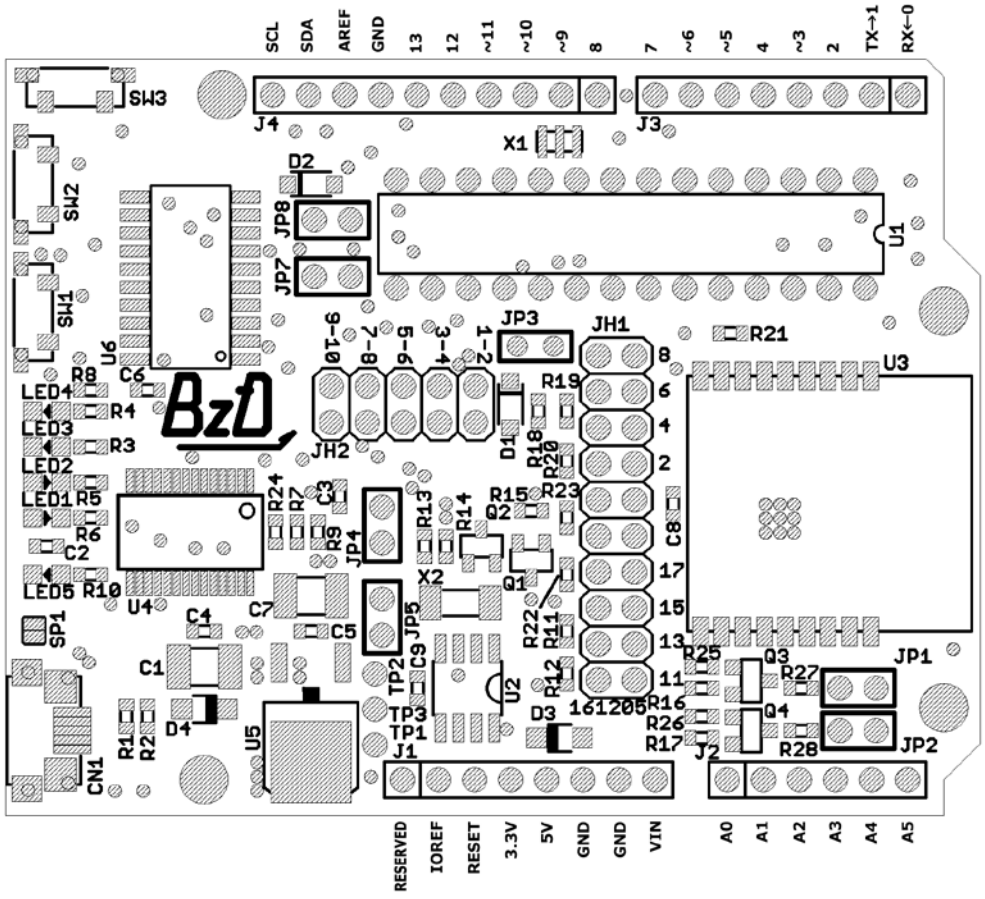
3. ハード仕様

CPU	ATmega328P 8-bit AVR 16MHz
Flash Memory	32Kbytes (プログラム書き込み用)
SRAM	2Kbytes
EEPROM	ATmega328P 内蔵　1024bytes DS1388 内蔵　512bytes
ネットワーク	ESP-WROOM-02 Wi-Fi モジュール搭載 802.11b/g/n (2.4GB)　Wireless LAN
RTC	DS1388 ※ATmega328P と I2C 接続（アドレス 0x68)
USB	Micro USB (給電及びシリアル通信用)
I/O コネクタ	アナログ 6 ピン / デジタル 14 ピン(うち PWM6 ピン) ※デジタル 4 ピン(うち PWM1 ピン)をモジュール制御用に予約済み
電源	マイコン動作電圧: 5V
外寸	W53.4mm × H68.6mm (Arduino UNO R3 と同様)
重量	27g
環境	温度　0～40℃ / 湿度 10～80% (非結露)

4. 外寸と各部名称



5. 基板レイアウト図



6. I/O コネクタと設定ピンの説明

1. Arduino Uno 互換ピンソケット（J1～4）

BiZduino の両端にあるピンソケットは Arduino Uno 互換です。
ただし、以下のピンソケットは内部で使用しています。

ピン名称	説明
D0 (Rx)	RX (Wi-Fi モジュール → ATmega328P) ※1
D1 (Tx)	TX (ATmega328P → Wi-Fi モジュール) ※1
D8	HIGH = Wi-Fi モジュールリセット (HIGH にして約 100ms 後に LOW にする) ※2
D9	HIGH = Arduino + Wi-Fi モード ※1 LOW = Arduino モード

※1 Arduino + Wi-Fi モードにすると Wi-Fi モジュールと ATmega328P の通信が可能になります。
スケッチで以下のように D9 を HIGH にします。

```
pinMode(9, OUTPUT);
digitalWrite(9, HIGH);
```

Wi-Fi モジュールと ATmega328P の通信を遮断するには、
スケッチで以下のように D9 を LOW にします。

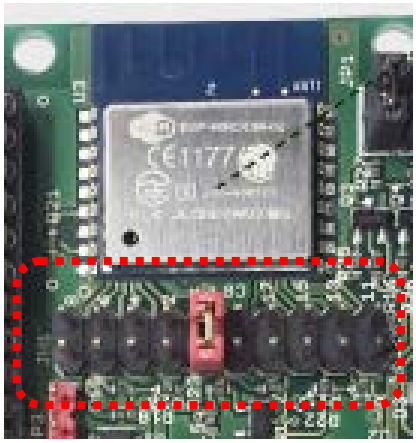
```
pinMode(9, OUTPUT);
digitalWrite(9, LOW);
```

※2 リセットするにはスケッチを以下のようにします。

```
pinMode(8, OUTPUT);
digitalWrite(8, HIGH);
delay(100);
digitalWrite(8, LOW);
```

2. Wi-Fi モジュール 端子（JH1）

BiZduino に搭載している Wi-Fi モジュールの各端子と以下のように繋がります。



Wi-Fi モジュール

IO0	IO15	IO12	EN	3.3V	GND	IO16	RST	GND	RXD
GND	IO2	IO13	IO14	3.3V	GND	TOUT	IO5	TXD	IO4

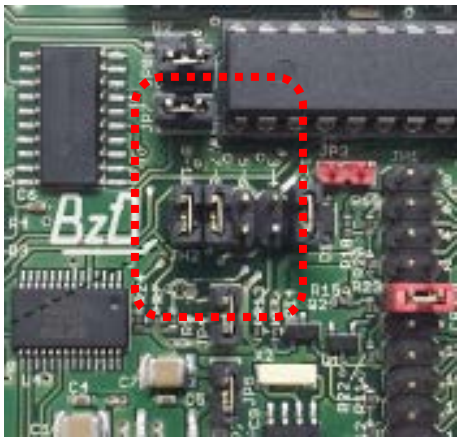
※ 各ピンの詳細については URL からアクセスできます。

「2. Pin Description」をご覧ください。

https://www.espressif.com/sites/default/files/documentation/0c-esp-wroom-02_datasheet_en.pdf

3. 動作モード設定（JH2）

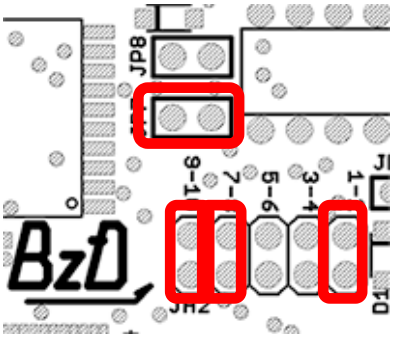
BiZduino の動作モードを設定できます。



ピン番号	説明
1-2	Wi-Fi モジュール通信モニタ接続（※1）
3-4	USB RX と Wi-Fi TX を接続（※2）
5-6	USB TX と Wi-Fi RX を接続（※2）
7-8	USB RX と ATmega328P TX を接続（※1,3）
9-10	USB TX と ATmega328P RX を接続（※1,3）

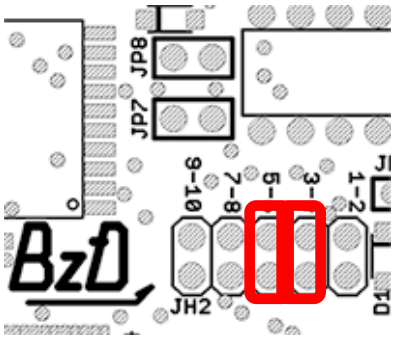
※1 Arduino + Wi-Fi モード

この他に JP7 にジャンパーピンを付けた状態でスケッチ上で D9 を HIGH にする必要があります。



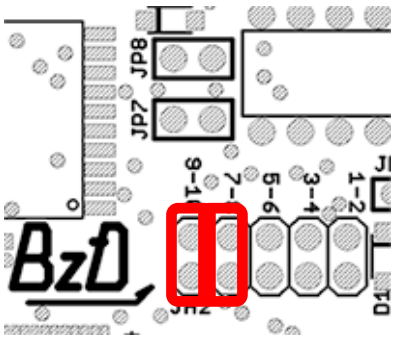
※2　Wi-Fi ダイレクトモード

この他に JP7 のジャンパーピンを外す必要があります。JP7 を外すと ATmega328 と ESP-WROOM-02 の接続が強制的に切断されます。



※3　Arduino モード

この他に JP7 のジャンパーピンを外すかスケッチ上で D9 を LOW にする必要があります。JP7 を外すと ATmega328 と ESP-WROOM-02 の接続が強制的に切断されます。



スケッチ上での D9 の制御については P6 をご参照ください。

4. その他（JP1～5）

その他のピンについては以下のとおりです。

ピン番号	説明
JP1	RTC SCL (DS1388 - ATmega328P) ジャンパーピンを外すと信号線を切り離します。
JP2	RTC SDA (DS1388 - ATmega328P) ジャンパーピンを外すと信号線を切り離します。
JP3	Wi-Fi モジュールへスケッチ書き込み時、ファームウェア書き換え時に添付の赤色ジャンパーピンを付けます。
JP4	USB DTR (ATmega328P RESET_PC6) ジャンパーピンを外すとATmega328リセット端子への信号線を切り離します。
JP5	3.3V (RTC モジュール VCC) ジャンパーピンを外すとモジュールへ供給する 3.3V を切り離します。外した場合は、ESP-WROOM-02 の電源も切れます。
JP7	D9 (Arduino + Wi-Fi モード) ジャンパーピンを外すと、ATmega328P - Wi-Fi モジュール間の通信を切り離します。
JP8	D8 (Wi-Fi モジュールリセット) ジャンパーピンを外すと ESP-WROOM-02 リセット端子への信号線を切り離します。

7. 各スイッチの説明

BiZduino には以下のスイッチがあります。

スイッチ番号	説明
SW1	ATmega328P リセット
SW2	Wi-Fi モジュールリセット
SW3	汎用スイッチ (ATmega328P D12 に接続、ON の時に LOW) ※

※ SW3 の状態を読み取るにはスケッチで以下のようにします。

```
pinMode(12, INPUT_PULLUP);
if (digitalRead(12) == LOW) {
    // SW3 が ON になったときの処理をここに書く
} else {
    // SW3 が OFF になったときの処理をここに書く
}
delay(100);     // チャタリング防止
```

8. 各 LED の説明

BiZduino には以下の LED があります。

LED 番号	説明
LED1	TX (USB → ATmega328P) 通信時に点灯
LED2	RX (USB ← ATmega328P) 通信時に点灯
LED3	D13 が HIGH のときに点灯 ※
LED4	D9 (点灯 = Arduino + Wi-Fi モード、 消灯 = Arduino モード)
LED5	POWER (5V) 通電時に点灯

※ LED3 を点灯・消灯するにはスケッチで以下のようにします。

```
pinMode(13, OUTPUT);
digitalWrite(13, HIGH);     // 点灯
delay(3000);                // 消灯まで 3 秒待つ
digitalWrite(13, LOW);      // 消灯
```

(delay は目視で LED の点灯・消灯が分かるようにするため)

9. Wi-Fi モジュールの設定

Arduino IDE のシリアルモニタ等で AT コマンドを送信することで Wi-Fi モジュールの設定をします。事前に開発環境をセットアップしてください。
（セットアップ手順は開発環境構築手順書をご覧ください）

AT コマンドの詳細資料は以下よりダウンロードしてください。
https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf
https://www.espressif.com/sites/default/files/documentation/4b-esp8266_at_command_examples_en.pdf

動作確認:

1. ArduinoIDE（以下 IDE）を起動します。
2. BiZduino の JH2 の 3-4,5-6 にジャンパーピンを付けて JP7 のジャンパーピンを外し(または J4 の D9 を LOW)、PC と BiZduino を接続します。
3. IDE にてメニューから「ツール」→「シリアルポート」で、BiZduino を接続したポートを選択します。
4. メニューから「ツール」→「シリアルモニタ」を開きます。
5. シリアルモニタ画面上部に”AT”と入力して送信ボタンをクリックします。
6. 応答を受信して”OK”と表示されると、正常に動作しています。

```
AT
OK
```

バージョン確認:

1. 動作確認の 1.～4.と同様に IDE のシリアルモニタを開きます。
2. シリアルモニタ画面上部に”AT+GMR”と入力して送信ボタンをクリックします。
3. 応答を受信してバージョン情報が表示されます。

```
AT+GMR
AT version:1.3.0.0(Jul 14 2016 18:54:01)
SDK version:2.0.0(656edbf)
compile time:Jul 19 2016 18:44:22
OK
```

IP、MAC アドレス確認:

1. 動作確認の 1.～4.と同様に IDE のシリアルモニタを開きます。
2. シリアルモニタ画面上部に”AT+CIFSR”と入力して送信ボタンをクリックします。
3. 応答を受信して IP、MAC アドレスが表示されます。

```
AT+CIFSR
+CIFSR:APIP,"192.168.4.1"
+CIFSR:APMAC,"XX:XX:XX:XX:XX:XX"
+CIFSR:STAIP,"192.168.XX.XX"
+CIFSR:STAMAC,"XX:XX:XX:XX:XX:XX"
OK
```

Wi-Fi モード切替:

1. 動作確認の 1.～4.と同様に IDE のシリアルモニタを開きます。
2. station mode（クライアント動作）に切り替えるには、シリアルモニタ画面上部に“AT+CWMODE_CUR=1”と入力して送信ボタンをクリックします。

```
AT+CWMODE_CUR=1
OK
```

3. softAP mode（アクセスポイント動作）に切り替えるには、シリアルモニタ画面上部に“AT+CWMODE_CUR=2”と入力して送信ボタンをクリックします。

```
AT+CWMODE_CUR=2
OK
```

4. softAP + Station mode（アクセスポイント+クライアント動作）に切り替えるには、シリアルモニタ画面上部に“AT+CWMODE_CUR=3”と入力して送信ボタンをクリックします。

```
AT+CWMODE_CUR=3
OK
```

セキュリティ設定:

Wi-Fi アクセスポイントとして使用する場合の設定です。

1. 動作確認の 1.～4.と同様に IDE のシリアルモニタを開きます。
2. Wi-Fi モード切替の 3.や 4.と同様に Wi-Fi モードの softAP(アクセスポイント動作)を有効にします。

```
AT+CWMODE_CUR=3
OK
```

3. 現在のセキュリティ設定を確認します。シリアルモニタ画面上部に“AT+CWSAP_CUR?”と入力して送信ボタンをクリックします。

```
AT+CWSAP_CUR?
+CWSAP_CUR:"ESP8266","",1,0,4,0
OK
```

(応答データの説明)
+CWSAP_CUR:[SSID],[パスワード],[チャンネル ID],[方式],
[最大接続数],[SSID ブロードキャスト]
SSID=ESP8266、パスワード=なし、チャンネル ID=1、方式=0"OPEN"、
最大接続数=4、SSID ブロードキャスト=0"有効"

4. デフォルトのセキュリティ設定を変更します。シリアルモニタ画面上部に以下のように入力して送信ボタンをクリックします。

```
AT+CWSAP_DEF="BiZduino_AP","password",1,3,1,0
OK
```

(送信データの説明)
AT+CWSAP_DEF=[SSID],[パスワード],[チャンネル ID],[方式],
[最大接続数],[SSID ブロードキャスト]
SSID=BiZduino_AP、パスワード=password、チャンネル ID=1、
方式=3"WPS2-PSK"、最大接続数=1、SSID ブロードキャスト=0"有効"

※ パスワードは 8～64 バイトの ASCII 文字が設定できます。
※ Wi-Fi モードで softAP が有効のときのみ成功します。

10. HTTP サーバー動作のサンプルスケッチ

HTTP サーバーとして動作させて、接続した PC やスマートフォンのブラウザに“Hello world”を表示するサンプルスケッチです。
このサンプルスケッチは、以下からダウンロードできます。
http://dl.bizright.jp/bd/httpServer.ino

httpServer.ino

```
#define WIFI_RESET     8
#define WIFI_ENABLE    9
#define LED3           13
#define BUFF_SIZE      256

char _rxBuffer[BUFF_SIZE];
int _indexBuf = 0;

void setup() {

    pinMode(WIFI_RESET, OUTPUT);
    pinMode(WIFI_ENABLE, OUTPUT);
    pinMode(LED3, OUTPUT);

    // Wi-Fi モジュールとの通信を開始
    Serial.begin(115200);

    // 受信バッファ初期化
    _indexBuf = 0;
    _rxBuffer[0] = '¥0';

    delay(3000);

    // Wi-Fi モジュール有効化
    digitalWrite(WIFI_ENABLE, HIGH);

    // AT コマンドのエコーを無効にする
    Serial.print(F("ATE0"));
```

```
    Serial.print("¥r¥n");
    waitResponse(500);

    // Wi-Fi モードの設定 (3 : softAP + station mode)
    Serial.print(F("AT+CWMODE_CUR=3"));
    Serial.print("¥r¥n");
    waitResponse(500);

    // マルチコネクションモード設定
    Serial.print(F("AT+CIPMUX=1"));
    Serial.print("¥r¥n");
    waitResponse(500);

    // サーバー設定
    Serial.print(F("AT+CIPSERVER=1,80"));
    Serial.print("¥r¥n");
    waitResponse(500);
}

void loop() {
    // Wi-Fi モジュールからの受信を読み取る
    if (receive()) {
        if (strstr(_rxBuffer, "GET") != NULL) {
            // GET 要求を受信
            waitReceiveWiFiData(1000);

            // 送信準備 (リンク ID、データサイズ)
            int dataSize = 237;
            Serial.print(F("AT+CIPSEND=0,"));
            Serial.print(dataSize);
            Serial.print("¥r¥n");
            if (waitResponse(500)) {
                // HTML 送信 (実データ)
                Serial.print(F("<!DOCTYPE html>"));
                Serial.print(F("<head>"));
                Serial.print(F("<meta http-equiv=¥"Content-Type¥" content=¥"text/html;
```

```
 charset=UTF-8¥">"));
                Serial.print(F("<meta name=¥"viewport¥" content=¥"width=device-width,
 initial-scale=1.0, maximum-scale=1.0, user-scalable=no¥" />"));
                Serial.print(F("</head>"));
                Serial.print(F("<body>"));
                Serial.print(F("<h1>Hello world</h1>"));
                Serial.print(F("</body>"));
                waitResponse(1000);
            }

            // TCP コネクション切断
            Serial.print(F("AT+CIPCLOSE=0"));
            Serial.print("¥r¥n");
            waitResponse(1000);

            // 受信バッファ初期化
            _indexBuf = 0;
            _rxBuffer[0] = '¥0';

            // LED3 点灯
            digitalWrite(LED3, HIGH);
        }
    }
}

// 対 Wi-Fi モジュールのシリアル通信受信
boolean receive() {
    char c;
    if (Serial.available()) {
        c = Serial.read();
        if (c >= 0 && c <= 0x80 && c != '¥0' && c != '¥t' && c != '¥v') {
            if ((_indexBuf + 1) >= BUFF_SIZE) {
                _indexBuf = 0;
                _rxBuffer[0] = '¥0';
            }
            _rxBuffer[_indexBuf] = c;
```

```
            _rxBuffer[_indexBuf + 1] = '¥0';
            _indexBuf++;
            return true;
        }
    }
    return false;
}

// Wi-Fi モジュールの応答を指定した時間まで待つ
boolean waitResponse(unsigned long waitMillis) {
    boolean result = false;
    char c;
    unsigned long currentMillis, previousMillis;
    currentMillis = millis();
    previousMillis = currentMillis;
    while (currentMillis - previousMillis < waitMillis) {
        if (receive()) {
            if ((strstr(_rxBuffer, "OK") != NULL)
                 || (strstr(_rxBuffer, "SEND OK") != NULL)) {
                // 正常応答受信
                result = true;
                break;
            }
            else if ((strstr(_rxBuffer, "ERROR") != NULL)
                      || (strstr(_rxBuffer, "FAIL") != NULL)) {
                // 異常応答受信
                break;
            }
        }
        currentMillis = millis();
    }
    _indexBuf = 0;
    _rxBuffer[0] = '¥0';
    delay(10);
    return result;
}
```

```
// ネットワークからのデータ受信を待ち受ける
int waitReceiveWiFiData(unsigned long waitMillis) {
    char c;
    unsigned long currentMillis, previousMillis;
    currentMillis = millis();
    previousMillis = currentMillis;
    while (currentMillis - previousMillis < waitMillis) {
        receive();
        currentMillis = millis();
    }
    return _indexBuf;
}
```

11. HTTP クライアント動作のサンプルスケッチ

HTTP クライアントとして動作させて、SW3 を押すと Web ページの html をシリアルモニタに表示するサンプルスケッチです。
このサンプルスケッチは、以下からダウンロードできます。
http://dl.bizright.jp/bd/httpClient.ino

※ Wi-Fi アクセスポイント環境に合わせてスケッチにある SSID とパスワードを変更してください。

httpClient.ino

```
#define WIFI_RESET     8
#define WIFI_ENABLE    9
#define SW3            12
#define LED3           13
#define BUFF_SIZE      256
#define HOST_NAME_LEN  18

char _rxBuffer[BUFF_SIZE];
int _indexBuf = 0;

const char WIFI_SSID[] = "SSID";
const char WIFI_PASS[] = "password";

const char HOST_NAME[HOST_NAME_LEN + 1] = "www.bizright.co.jp";

void setup() {

    pinMode(WIFI_RESET, OUTPUT);
    pinMode(WIFI_ENABLE, OUTPUT);
    pinMode(SW3, INPUT_PULLUP);
    pinMode(LED3, OUTPUT);

    // Wi-Fi モジュールとの通信を開始
    Serial.begin(115200);
```

```
    // 受信バッファ初期化
    _indexBuf = 0;
    _rxBuffer[0] = '¥0';

    delay(3000);

    // Wi-Fi モジュール有効化
    digitalWrite(WIFI_ENABLE, HIGH);

    // AT コマンドのエコーを無効にする
    Serial.print(F("ATE0"));
    Serial.print("¥r¥n");
    waitResponse(500);

    // Wi-Fi モードの設定 (1 : station mode)
    Serial.print(F("AT+CWMODE_CUR=1"));
    Serial.print("¥r¥n");
    waitResponse(500);

    // マルチコネクションモード設定
    Serial.print(F("AT+CIPMUX=1"));
    Serial.print("¥r¥n");
    waitResponse(500);
}

void loop() {
    if (digitalRead(SW3) == LOW) {
        // Wi-Fi AP へ接続
        Serial.print(F("AT+CWJAP_CUR=¥""));
        Serial.print(WIFI_SSID);
        Serial.print(F("¥",¥""));
        Serial.print(WIFI_PASS);
        Serial.print(F("¥""));
        Serial.print("¥r¥n");
        waitResponse(10000);
```

```
    // TCP コネクション
    Serial.print(F("AT+CIPSTART=0,¥"TCP¥",¥""));
    Serial.print(HOST_NAME);
    Serial.print(F("¥",80"));
    Serial.print("¥r¥n");
    waitResponse(10000);

    // 送信準備（リンク ID、データサイズ）
    int dataSize = 47 + HOST_NAME_LEN;
    Serial.print(F("AT+CIPSEND=0,"));
    Serial.print(dataSize);
    Serial.print("¥r¥n");
    if (waitResponse(500)) {
        // HTML リクエスト送信（実データ）
        Serial.print(F("GET / HTTP/1.1¥r¥nHost: "));
        Serial.print(HOST_NAME);
        Serial.print(F("¥r¥nUser-Agent: arduino¥r¥n¥r¥n"));
        waitResponse(1000);

        // HTML 受信
        waitReceiveWiFiData(10000);
        // LED3 点灯
        digitalWrite(LED3, HIGH);
    }
    else {
        // TCP コネクション切断
        Serial.print(F("AT+CIPCLOSE="));
        Serial.print(0);
        Serial.print("¥r¥n");
        waitResponse(1000);
    }

    // Wi-Fi AP 接続を切断
    Serial.print(F("AT+CWQAP"));
    Serial.print("¥r¥n");
    waitResponse(1000);
```

```
    }
}

// 対 Wi-Fi モジュールのシリアル通信受信
boolean receive() {
    char c;
    if (Serial.available()) {
        c = Serial.read();
        if (c >= 0 && c <= 0x80 && c != '¥0' && c != '¥t' && c != '¥v') {
            if ((_indexBuf + 1) >= BUFF_SIZE) {
                _indexBuf = 0;
                _rxBuffer[0] = '¥0';
            }
            _rxBuffer[_indexBuf] = c;
            _rxBuffer[_indexBuf + 1] = '¥0';
            _indexBuf++;
            return true;
        }
    }
    return false;
}

// Wi-Fi モジュールの応答を指定した時間まで待つ
boolean waitResponse(unsigned long waitMillis) {
    boolean result = false;
    char c;
    unsigned long currentMillis, previousMillis;
    currentMillis = millis();
    previousMillis = currentMillis;
    while (currentMillis - previousMillis < waitMillis) {
        if (receive()) {
            if ((strstr(_rxBuffer, "OK") != NULL)
                 || (strstr(_rxBuffer, "SEND OK") != NULL)) {
                // 正常応答受信
                result = true;
                break;
```

```
            }
            else if ((strstr(_rxBuffer, "ERROR") != NULL)
                    || (strstr(_rxBuffer, "FAIL") != NULL)) {
                // 異常応答受信
                break;
            }
        }
        currentMillis = millis();
    }
    _indexBuf = 0;
    _rxBuffer[0] = '¥0';
    delay(10);
    return result;
}

// ネットワークからのデータ受信を待ち受ける
int waitReceiveWiFiData(unsigned long waitMillis) {
    char c;
    unsigned long currentMillis, previousMillis;
    currentMillis = millis();
    previousMillis = currentMillis;
    while (currentMillis - previousMillis < waitMillis) {
        receive();
        currentMillis = millis();
    }
    return _indexBuf;
}
```

12. BiZduino2 台で通信するサンプルスケッチ

BiZduino を、1 台はサーバーモード、1 台はクライアントモードとして 2 台使用し、クライアントモードの BiZduino の SW3 を押すたびにサーバーモードの BiZduino の LED が点灯/消灯を繰り返すサンプルスケッチです。SW3 を押す際は、クライアントモードの BiZduino の LED3 が消えていることを確認してから押してください。
このサンプルスケッチは、以下からダウンロードできます。
http://dl.bizright.jp/bd/wifi1.ino
http://dl.bizright.jp/bd/wifi2.ino

wifi1.ino（サーバーモード）

```
#define WIFI_RESET    8
#define WIFI_ENABLE   9
#define LED3          13
#define BUFF_SIZE     256

char _rxBuffer[BUFF_SIZE];
int _indexBuf = 0;

void setup() {

    pinMode(WIFI_RESET, OUTPUT);
    pinMode(WIFI_ENABLE, OUTPUT);
    pinMode(LED3, OUTPUT);

    Serial.begin(115200);

    _indexBuf = 0;
    _rxBuffer[0] = '¥0';

    delay(3000);
    digitalWrite(WIFI_ENABLE, HIGH);

    Serial.print(F("ATE0"));
    Serial.print("¥r¥n");
    waitResponse(500);

    Serial.print(F("AT+CWMODE_CUR=3"));
    Serial.print("¥r¥n");
    waitResponse(500);

    Serial.print(F("AT+CWSAP_DEF=¥"BiZduino_AP¥",¥"password¥",1,3,1,0"));
    Serial.print("¥r¥n");
    waitResponse(500);

    Serial.print(F("AT+CIPMUX=1"));
    Serial.print("¥r¥n");
    waitResponse(500);

    Serial.print(F("AT+CIPSERVER=1,80"));
```

```
    Serial.print("¥r¥n");
    waitResponse(500);
}

void loop() {
  if (receive()) {
    if (strstr(_rxBuffer, "GET") != NULL) {
      if(digitalRead(LED3) == LOW){
        digitalWrite(LED3, HIGH);
      } else {
        digitalWrite(LED3, LOW);
      }
      waitReceiveWiFiData(1000);

      int dataSize = 1;
      Serial.print(F("AT+CIPSEND=0,"));
      Serial.print(dataSize);
      Serial.print("¥r¥n");
      if (waitResponse(500)) {
          Serial.print(F("1"));
          waitResponse(1000);
      }

      Serial.print(F("AT+CIPCLOSE=0"));
      Serial.print("¥r¥n");
      waitResponse(1000);
      _indexBuf = 0;
      _rxBuffer[0] = '¥0';
    }
  }
}

boolean receive() {
    char c;
    if (Serial.available()) {
        c = Serial.read();
        if (c >= 0 && c <= 0x80 && c != '¥0' && c != '¥t' && c != '¥v') {
            if ((_indexBuf + 1) >= BUFF_SIZE) {
                _indexBuf = 0;
                _rxBuffer[0] = '¥0';
            }
            _rxBuffer[_indexBuf] = c;
            _rxBuffer[_indexBuf + 1] = '¥0';
            _indexBuf++;
            return true;
        }
    }
    return false;
}

boolean waitResponse(unsigned long waitMillis) {
    boolean result = false;
    char c;
    unsigned long currentMillis, previousMillis;
    currentMillis = millis();
    previousMillis = currentMillis;
```

```
    while (currentMillis - previousMillis < waitMillis) {
        if (receive()) {
            if ((strstr(_rxBuffer, "OK") != NULL) || (strstr(_rxBuffer, "SEND OK") != NULL))
{
                result = true;
                break;
            }
            else if ((strstr(_rxBuffer, "ERROR") != NULL) || (strstr(_rxBuffer, "FAIL") !=
NULL)) {
                break;
            }
        }
        currentMillis = millis();
    }
    _indexBuf = 0;
    _rxBuffer[0] = '¥0';
    delay(10);
    return result;
}

int waitReceiveWiFiData(unsigned long waitMillis) {
    char c;
    unsigned long currentMillis, previousMillis;
    currentMillis = millis();
    previousMillis = currentMillis;
    while (currentMillis - previousMillis < waitMillis) {
        receive();
        currentMillis = millis();
    }
    return _indexBuf;
}
```

wifi2.ino(クライアントモード)

```
#define WIFI_RESET    8
#define WIFI_ENABLE   9
#define SW3           12
#define LED3          13
#define BUFF_SIZE     256
#define HOST_NAME_LEN 11

char _rxBuffer[BUFF_SIZE];
int _indexBuf = 0;

const char WIFI_SSID[] = "BiZduino_AP";
const char WIFI_PASS[] = "password";

const char HOST_NAME[HOST_NAME_LEN + 1] = "192.168.4.1";

void setup() {

    pinMode(WIFI_RESET, OUTPUT);
    pinMode(WIFI_ENABLE, OUTPUT);
    pinMode(SW3, INPUT_PULLUP);
```

```
    pinMode(LED3, OUTPUT);

    Serial.begin(115200);

    _indexBuf = 0;
    _rxBuffer[0] = '¥0';

    delay(3000);

    digitalWrite(WIFI_ENABLE, HIGH);

    Serial.print(F("ATE0"));
    Serial.print("¥r¥n");
    waitResponse(500);

    Serial.print(F("AT+CWMODE_CUR=1"));
    Serial.print("¥r¥n");
    waitResponse(500);

    Serial.print(F("AT+CIPMUX=1"));
    Serial.print("¥r¥n");
    waitResponse(500);
}

void loop() {
    if (digitalRead(SW3) == LOW) {
        digitalWrite(LED3, HIGH);
        Serial.print(F("AT+CWJAP_CUR=¥""));
        Serial.print(WIFI_SSID);
        Serial.print(F("¥",¥""));
        Serial.print(WIFI_PASS);
        Serial.print(F("¥""));
        Serial.print("¥r¥n");
        waitResponse(10000);

        Serial.print(F("AT+CIPSTART=0,¥"TCP¥",¥""));
        Serial.print(HOST_NAME);
        Serial.print(F("¥",80"));
        Serial.print("¥r¥n");
        waitResponse(10000);

        int dataSize = 47 + HOST_NAME_LEN;
        Serial.print(F("AT+CIPSEND=0,"));
        Serial.print(dataSize);
        Serial.print("¥r¥n");
        if (waitResponse(500)) {
            Serial.print(F("GET / HTTP/1.1¥r¥nHost: "));
            Serial.print(HOST_NAME);
            Serial.print(F("¥r¥nUser-Agent: arduino¥r¥n¥r¥n"));
            waitResponse(1000);

            waitReceiveWiFiData(10000);
        }
        else {
            Serial.print(F("AT+CIPCLOSE="));
            Serial.print(0);
```

```
            Serial.print("¥r¥n");
            waitResponse(1000);
        }

        Serial.print(F("AT+CWQAP"));
        Serial.print("¥r¥n");
        waitResponse(1000);
        digitalWrite(LED3, LOW);
    }
}

boolean receive() {
    char c;
    if (Serial.available()) {
        c = Serial.read();
        if (c >= 0 && c <= 0x80 && c != '¥0' && c != '¥t' && c != '¥v') {
            if ((_indexBuf + 1) >= BUFF_SIZE) {
                _indexBuf = 0;
                _rxBuffer[0] = '¥0';
            }
            _rxBuffer[_indexBuf] = c;
            _rxBuffer[_indexBuf + 1] = '¥0';
            _indexBuf++;
            return true;
        }
    }
    return false;
}

boolean waitResponse(unsigned long waitMillis) {
    boolean result = false;
    char c;
    unsigned long currentMillis, previousMillis;
    currentMillis = millis();
    previousMillis = currentMillis;
    while (currentMillis - previousMillis < waitMillis) {
        if (receive()) {
            if ((strstr(_rxBuffer, "OK") != NULL) || (strstr(_rxBuffer, "SEND OK") != NULL))
{
                result = true;
                break;
            }
            else if ((strstr(_rxBuffer, "ERROR") != NULL) || (strstr(_rxBuffer, "FAIL") !=
NULL)) {
                break;
            }
        }
        currentMillis = millis();
    }
    _indexBuf = 0;
    _rxBuffer[0] = '¥0';
    delay(10);
    return result;
}

int waitReceiveWiFiData(unsigned long waitMillis) {
```

```
        char c;
        unsigned long currentMillis, previousMillis;
        currentMillis = millis();
        previousMillis = currentMillis;
        while (currentMillis - previousMillis < waitMillis) {
            receive();
            currentMillis = millis();
        }
        return _indexBuf;
    }
```

13.RTC に NTP から時刻取得・設定サンプルスケッチ

5 秒おきに現在時刻をシリアルモニタへ表示し、SW3 を押すと、RTC に NTP から取得した時刻を設定するサンプルスケッチです。
このサンプルスケッチは、以下からダウンロードできます。
http://dl.bizright.jp/bd/rtcNtpAdjust.ino

※ Wi-Fi アクセスポイント環境に合わせてスケッチにある SSID とパスワードを変更してください。

rtcNtpAdjust.ino

```
#include <Wire.h>

// DS1388 アドレス
#define DS1388_ADDRESS          0x68

// DS1388 EEPROM レジスタ ビット
#define DS1388_EEPROM_0         0x01
#define DS1388_EEPROM_1         0x02

#define WIFI_RESET      8
#define WIFI_ENABLE     9
#define SW3             12
#define LED3            13
#define BUFF_SIZE       256
#define HOST_NAME_LEN   18
#define TIME_ZONE       9 * 60 * 60

#define SECONDS_FROM_1970_TO_2000 946684800

char _rxBuffer[BUFF_SIZE];
int _indexBuf = 0;

const char WIFI_SSID[] = "SSID";
const char WIFI_PASS[] = "password";
```

```
unsigned int localPort = 2390;

const char* ntpServerName = "ntp.nict.jp";

// NTP パケットサイズ
const int NTP_PACKET_SIZE = 48;

// NTP パケットバッファ
byte packetBuffer[NTP_PACKET_SIZE];

const byte NTP_VERSION_4    = 0b00100000;
const byte NTP_MODE_CLIENT  = 0b00000011;

const char _IPD[5] PROGMEM = "+IPD";

unsigned long currentMillis, previousMillis;

uint8_t yOff, m, d, hh, mm, ss;
const uint8_t daysInMonth [] PROGMEM = { 31,28,31,30,31,30,31,31,30,31,30,31 };


void setup() {

  pinMode(WIFI_RESET, OUTPUT);
  pinMode(WIFI_ENABLE, OUTPUT);
  pinMode(SW3, INPUT_PULLUP);
  pinMode(LED3, OUTPUT);

  // Wi-Fi モジュールとの通信を開始
  Serial.begin(115200);

  // 受信バッファ初期化
  _indexBuf = 0;
  _rxBuffer[0] = '¥0';
```

```
// RTC モジュールとの通信を開始
Wire.begin();

delay(3000);

// Wi-Fi モジュール有効化
digitalWrite(WIFI_ENABLE, HIGH);

// AT コマンドのエコーを無効にする
Serial.print(F("ATE0"));
Serial.print("¥r¥n");
waitResponse(500);

// Wi-Fi モードの設定 (1 : station mode)
Serial.print(F("AT+CWMODE_CUR=1"));
Serial.print("¥r¥n");
waitResponse(500);

// マルチコネクションモード設定
Serial.print(F("AT+CIPMUX=1"));
Serial.print("¥r¥n");
waitResponse(500);

// RTC 時刻調整 (調整しない場合は以下をコメント化)
adjustRtc(2017, 1, 1, 0, 0, 0);

// シリアルモニタへ出力のため Wi-Fi モジュールとの通信を切り離す
digitalWrite(WIFI_ENABLE, LOW);
// RTC 時刻取得
if (!printRtc()) {
  Serial.println("printRtc error.");
}
currentMillis = millis();
previousMillis = currentMillis;
// Wi-Fi モジュールとの通信を有効に戻す
delay(50);
```

```
  digitalWrite(WIFI_ENABLE, HIGH);
}

void loop() {
  if (digitalRead(SW3) == LOW) {

    // シリアルモニタへ出力のため Wi-Fi モジュールとの通信を切り離す
    digitalWrite(WIFI_ENABLE, LOW);

    Serial.println(F(">>>>> NTP start"));

    // Wi-Fi モジュールとの通信を有効に戻す
    delay(50);
    digitalWrite(WIFI_ENABLE, HIGH);

    // Wi-Fi AP へ接続
    Serial.print(F("AT+CWJAP_CUR=¥""));
    Serial.print(WIFI_SSID);
    Serial.print(F("¥",¥""));
    Serial.print(WIFI_PASS);
    Serial.print(F("¥""));
    Serial.print("¥r¥n");
    waitResponse(10000);

    // LED3 点灯
    digitalWrite(LED3, HIGH);

    // UDP ポート登録
    Serial.print(F("AT+CIPSTART=0,¥"UDP¥",¥""));
    Serial.print(ntpServerName);
    Serial.print(F("¥",123¥r¥n"));
    waitResponse(10000);

    // 送信準備（リンク ID、データサイズ）
    Serial.print(F("AT+CIPSEND=0,"));
    Serial.print(NTP_PACKET_SIZE);
```

```
  Serial.print("¥r¥n");
  if (waitResponse(500)) {
    // NTP リクエスト送信（実データ）
    sendNTPpacket();
    waitResponse(1000);
    // NTP データ受信
    if (waitReceiveNtpData(10000) == NTP_PACKET_SIZE) {
      readNTPdata();
    }
  }

  // コネクション切断
  Serial.print(F("AT+CIPCLOSE="));
  Serial.print(0);
  Serial.print("¥r¥n");
  waitResponse(1000);

  // Wi-Fi AP 接続を切断
  Serial.print(F("AT+CWQAP"));
  Serial.print("¥r¥n");
  waitResponse(1000);

  // LED3 点灯
  digitalWrite(LED3, LOW);
}

currentMillis = millis();
if (currentMillis - previousMillis > 5000) {

  // シリアルモニタへ出力のため Wi-Fi モジュールとの通信を切り離す
  digitalWrite(WIFI_ENABLE, LOW);

  // RTC 時刻取得
  if (!printRtc()) {
    Serial.println("printRtc error.");
  }
```

```
    // Wi-Fi モジュールとの通信を有効に戻す
    delay(50);
    digitalWrite(WIFI_ENABLE, HIGH);

    previousMillis = currentMillis;
  }
}

// 対 Wi-Fi モジュールのシリアル通信受信
boolean receive() {
  char c;
  if (Serial.available()) {
    c = Serial.read();
    //if (c >= 0 && c <= 0x80 && c != '¥0' && c != '¥t' && c != '¥v') {
    if ((_indexBuf + 1) >= BUFF_SIZE) {
      _indexBuf = 0;
      _rxBuffer[0] = '¥0';
    }
    _rxBuffer[_indexBuf] = c;
    _rxBuffer[_indexBuf + 1] = '¥0';
    _indexBuf++;
    return true;
  }
  return false;
}

// Wi-Fi モジュールの応答を指定した時間まで待つ
boolean waitResponse(unsigned long waitMillis) {
  boolean result = false;
  char c;
  unsigned long currentMillis, previousMillis;
  currentMillis = millis();
  previousMillis = currentMillis;
  while (currentMillis - previousMillis < waitMillis) {
    if (receive()) {
```

```
      if ((strstr(_rxBuffer, "OK") != NULL)
          || (strstr(_rxBuffer, "SEND OK") != NULL)) {
        // 正常応答受信
        result = true;
        break;
      }
      else if ((strstr(_rxBuffer, "ERROR") != NULL)
               || (strstr(_rxBuffer, "FAIL") != NULL)) {
        // 異常応答受信
        break;
      }
    }
    currentMillis = millis();
  }
  _indexBuf = 0;
  _rxBuffer[0] = '¥0';
  delay(10);
  return result;
}

// NTP データ受信を待ち受ける
// <w_time> 待ち時間
// <buf>    受信データの格納場所
// <lenBuf> buf で確保しているサイズ
// <return> 受信したデータのサイズ（len 以下）
int waitReceiveNtpData(unsigned long waitMillis)
{
  char * pIPD = NULL;
  char * pData = NULL;
  int dataSize = -1, lenDataSize = -1;
  unsigned long currentMillis, previousMillis;
  _indexBuf = 0;
  _rxBuffer[0] = '¥0';
  currentMillis = millis();
  previousMillis = currentMillis;
  while (currentMillis - previousMillis < waitMillis) {
```

```
    if (receive()) {
      if (pIPD == NULL) {
        pIPD = strstr_P(_rxBuffer, _IPD);
      }
      else if (pData == NULL) {
        pData = strchr(pIPD, ':');
      }
      else if (dataSize <= 0) {
        int lenDataSize = pData - (pIPD + 7) + 1;
        char * strDataSize = (char *) malloc(lenDataSize + 1);
        strlcpy(strDataSize, pIPD + 7, lenDataSize);
        dataSize = atoi(strDataSize);
        free(strDataSize);
      }
      else if ((_rxBuffer + _indexBuf) >= (pData + 1 + dataSize)) {
        memcpy(packetBuffer, (byte*)(pData + 1), NTP_PACKET_SIZE);
        break;
      }
    }
    currentMillis = millis();
  }
  return dataSize;
}

// NTP パケット送信
void sendNTPpacket()
{
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  packetBuffer[0] = NTP_VERSION_4 | NTP_MODE_CLIENT;  // LI, Version, Mode
  Serial.write(packetBuffer, NTP_PACKET_SIZE);
}

void setTime(uint32_t t) {
    t -= SECONDS_FROM_1970_TO_2000;    // bring to 2000 timestamp from 1970

    ss = t % 60;
```

```
    t /= 60;
    mm = t % 60;
    t /= 60;
    hh = t % 24;
    uint16_t days = t / 24;
    uint8_t leap;
    for (yOff = 0; ; ++yOff) {
        leap = yOff % 4 == 0;
        if (days < 365 + leap)
            break;
        days -= 365 + leap;
    }
    for (m = 1; ; ++m) {
        uint8_t daysPerMonth = pgm_read_byte(daysInMonth + m - 1);
        if (leap && m == 2)
            ++daysPerMonth;
        if (days < daysPerMonth)
            break;
        days -= daysPerMonth;
    }
    d = days + 1;
}

// NTP 受信データ読み取り
void readNTPdata()
{
  // シリアルモニタへ出力のため Wi-Fi モジュールとの通信を切り離す
  digitalWrite(WIFI_ENABLE, LOW);

  Serial.println();   // 改行 x2
  Serial.println();

  unsigned long highWord = word(packetBuffer[40], packetBuffer[41]);
  unsigned long lowWord = word(packetBuffer[42], packetBuffer[43]);
  unsigned long secsSince1900 = highWord << 16 | lowWord;
  const unsigned long seventyYears = 2208988800UL;
```

```
  unsigned long epoch = secsSince1900 - seventyYears;   // UNIX タイムスタンプ
  unsigned long jstTime = epoch + TIME_ZONE;
  setTime(jstTime);

  // RTC 設定前に待機 (BiZduino 試作用)
  delay(500);

//  if (adjustRtc(year(), month(), day(), hour(), minute(), second())) {
  if (adjustRtc(2000 + yOff, m, d, hh, mm, ss)) {
    Serial.println(F(">>>>> RTC adjust success."));
  } else {
    Serial.println(F(">>>>> RTC adjust error."));
  }

  // Wi-Fi モジュールとの通信を有効に戻す
  delay(50);
  digitalWrite(WIFI_ENABLE, HIGH);
}

uint8_t bcd2bin(uint8_t val) {
    return val - 6 * (val >> 4);
}

uint8_t bin2bcd (uint8_t val) {
    return val + 6 * (val / 10);
}

// RTC 時刻調整
boolean adjustRtc(uint16_t year, uint8_t month, uint8_t day, uint8_t hour, uint8_t min, uint8_t
sec) {
    int res = 0;
    Wire.beginTransmission(DS1388_ADDRESS);
        Wire.write((byte) 0);
        Wire.write(bin2bcd(0));
        Wire.write(bin2bcd(sec));
        Wire.write(bin2bcd(min));
```

```
        Wire.write(bin2bcd(hour));
        Wire.write(bin2bcd(0));
        Wire.write(bin2bcd(day));
        Wire.write(bin2bcd(month));
        Wire.write(bin2bcd(year - 2000));
    res = Wire.endTransmission();
    if (res == 0) {
        Wire.beginTransmission(DS1388_ADDRESS);
        Wire.write((byte) 0x0b);
        Wire.write((byte) 0x00);
        res = Wire.endTransmission();
    }
    return res == 0;
}

// RTC 時刻取得
boolean printRtc() {
    boolean result = false;
    int res = 0;
    Wire.beginTransmission(DS1388_ADDRESS);
        Wire.write((byte) 0);
    res = Wire.endTransmission();
    if (res == 0) {
        Wire.requestFrom(DS1388_ADDRESS, 8);
        if (Wire.available()) {
            uint8_t hs = bcd2bin(Wire.read() & 0x7F);
            uint8_t ss = bcd2bin(Wire.read() & 0x7F);
            uint8_t mm = bcd2bin(Wire.read());
            uint8_t hh = bcd2bin(Wire.read());
            Wire.read();
            uint8_t d = bcd2bin(Wire.read());
            uint8_t m = bcd2bin(Wire.read());
            uint16_t y = bcd2bin(Wire.read()) + 2000;
            Serial.print(F("RTC: "));
            printDateTime(y, m, d, hh, mm, ss);
            result = true;
```

```
        }
    }
    return result;
}

void printDateTime(uint16_t year, uint8_t month, uint8_t day, uint8_t hour, uint8_t min, uint8_t
sec) {
    Serial.print(year);
    Serial.print('/');
    print2d(month);
    Serial.print('/');
    print2d(day);
    Serial.print(' ');
    print2d(hour);
    Serial.print(':');
    print2d(min);
    Serial.print(':');
    print2d(sec);
    Serial.println();
}

void print2d(uint8_t val) {
    if (val < 10) {
        Serial.print("0");
    }
    Serial.print(val);
}
```

14.RTC の EEPROM 書き込み/読み取りサンプルスケッチ

起動時に EEPROM 内 512byte 全体のデータを読み込みシリアルモニタに表示して、SW3 を押すたび 0x00、0x01、…0xFF のデータを 512byte 全体に書き込みと読み込みをするサンプルスケッチです。
このサンプルスケッチは、以下からダウンロードできます。
http://dl.bizright.jp/bd/rtcEEPROM.ino

rtcEEPROM.ino

```
#include <Wire.h>

// DS1388 アドレス
#define DS1388_ADDRESS          0x68

// DS1388 EEPROM レジスタ ビット
#define DS1388_EEPROM_0         0x01
#define DS1388_EEPROM_1         0x02

#define SW3   12
#define LED3  13

int writeData = 0;
boolean dataWrite = false;

byte writeBuff[8];
byte readBuff[8];
char header[8];
char record[49];

void setup() {
    pinMode(SW3, INPUT_PULLUP);
    pinMode(LED3, OUTPUT);

    Serial.begin(115200);
    Wire.begin();
```

```
    // RTC EEPROM 全領域を読み取り
    digitalWrite(LED3, HIGH);
    readAll();
    digitalWrite(LED3, LOW);
}

void loop() {
    if (digitalRead(SW3) == LOW) {
        if (!dataWrite) {
            dataWrite = true;
            digitalWrite(LED3, HIGH);
            Serial.println();
            // RTC EEPROM 全領域へ書き込み
            if (writeAll(writeData)) {
                delay(500);
                Serial.println();
                // RTC EEPROM 全領域を読み取り
                if (readAll()) {
                    writeData++;
                    if (writeData > 255) {
                        writeData = 0;
                    }
                }
            }
            digitalWrite(LED3, LOW);
        }
        delay(200);
    }
    else {
        dataWrite = false;
        delay(200);
    }
}

// RTC EEPROM のアドレスを取得
uint8_t getAddressRtcEPPROM(uint16_t pos) {
```

```
    if (pos > 255) {
        return DS1388_ADDRESS | DS1388_EEPROM_1;
    } else {
        return DS1388_ADDRESS | DS1388_EEPROM_0;
    }
}

// RTC EEPROM 全領域へ書き込み
boolean writeAll(byte wdata) {
    Serial.print(F("writeAll: "));
    Serial.println(wdata, HEX);
    for (int i = 0; i < 8; i++) {
        writeBuff[i] = wdata;
    }
    for (int j = 0; j < 64; j++) {
        Serial.print(F(">"));
        if (!writeRtcEEPROMPage(j, writeBuff)) {
            Serial.println(F(" error."));
            return false;
        }
        delay(10);
    }
    Serial.println(F(" success."));
    return true;
}

// RTC EEPROM 全領域を読み取り
boolean readAll() {
    Serial.println(F("readAll:"));
    for (int i = 0; i < 64; i++) {
        if (i % 2 == 0) {
            sprintf(header, "%04X : ", (i / 2) * 16);
            Serial.print(header);
        }
        if (!readRtcEEPROMPage(i, readBuff)) {
            Serial.println();
```

```
            Serial.println(F("> error."));
            return false;
        }
        sprintf(record, "%02X %02X %02X %02X %02X %02X %02X %02X ",
            readBuff[0], readBuff[1], readBuff[2], readBuff[3],
            readBuff[4], readBuff[5], readBuff[6], readBuff[7]);
        Serial.print(record);
        if (i % 2 != 0) {
            Serial.println();
        }
    }
    Serial.println(F("> success."));
    return true;
}

boolean writeRtcEEPROMPage(uint8_t page, uint8_t *data) {
    if (page >= 64) {
        return false;
    }
    int res = 0;
    Wire.beginTransmission(getAddressRtcEPPROM((uint16_t)page * 8));
    uint8_t rel_pos = ((uint16_t)page * 8) % 256;
    Wire.write((byte)rel_pos);
    for(uint8_t i = 0; i < 8; i++) {
        Wire.write((byte)data[i]);
    }
    res = Wire.endTransmission();
    return res == 0;
}

boolean readRtcEEPROMPage(uint8_t page, uint8_t *data) {
    if (page >= 64) {
        return false;
    }
    int res = 0;
    uint8_t devAddr = getAddressRtcEPPROM((uint16_t)page * 8);
```

```
    Wire.beginTransmission(devAddr);
    uint8_t rel_pos = ((uint16_t)page * 8) % 256;
    Wire.write((byte)rel_pos);
    res = Wire.endTransmission(true);
    if (res != 0) {
        return false;
    }
    Wire.requestFrom(devAddr, (uint8_t)8);
    for(uint8_t i = 0; i < 8; i++) {
        data[i] = Wire.read();
    }
    res = Wire.endTransmission();
    return res == 0;
}

// RTC EEPROM へ 1byte 書き込み
boolean writeRtcEEPROM(uint16_t pos, uint8_t c) {
    if(pos >= 512){
        return false;
    }
    int res = 0;
    uint8_t rel_pos = pos % 256;
    Wire.beginTransmission(getAddressRtcEPPROM(pos));
        Wire.write((byte)rel_pos);
        Wire.write((byte)c);
    res = Wire.endTransmission();
    if (res != 0) {
        Serial.print(F(" error :"));
        Serial.println(res);
    }
    return res == 0;
}

// RTC EEPROM から 1byte 読み取り
uint8_t readRtcEEPROM(uint16_t pos) {
    if(pos >= 512){
```

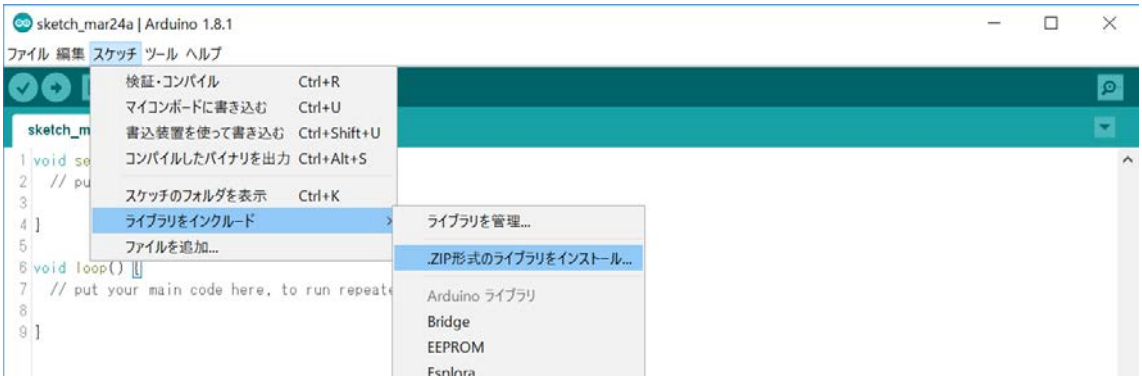
```
        return 0;
    }
    int res = 0;
    uint8_t rel_pos = pos % 256;
    uint8_t c = 0xFF;
    uint8_t devAddr = getAddressRtcEPPROM(pos);
    Wire.beginTransmission(devAddr);
    Wire.write((byte)rel_pos);
    Wire.endTransmission(true);
    Wire.requestFrom(devAddr, (uint8_t)1);
    if (Wire.available()){
        c = Wire.read();
    }
    return c;
}
```

15. BiZduino ライブラリ

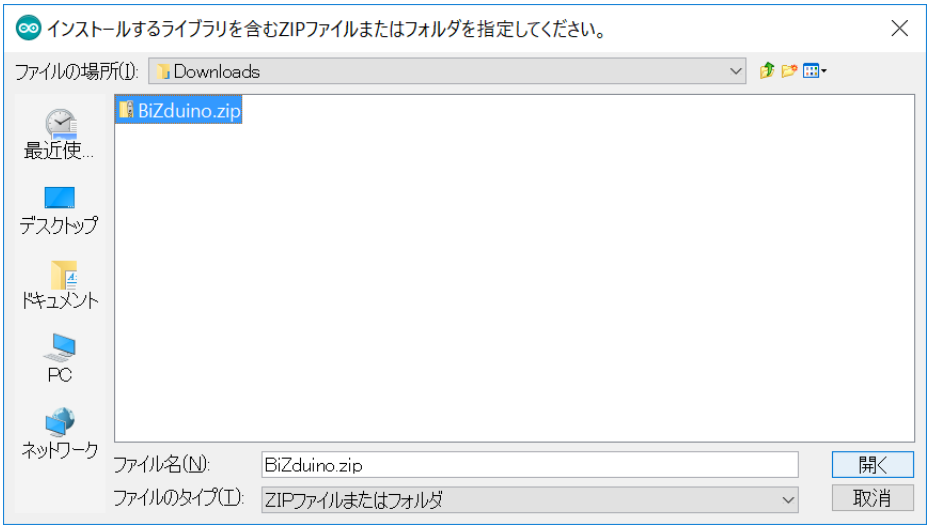
BiZduino の LED3、SW3、Wi-Fi モジュール、RTC モジュールを扱うための BiZduino ライブラリを用意しています。
BiZduino ライブラリを IDE へ追加する方法を以下に記します。

1. 以下の URL から BiZduino ライブラリをダウンロードします。
   http://dl.bizright.jp/bd/BiZduino.zip

2. IDE を起動して、メニューから「スケッチ」→「ライブラリをインクルード」→「.ZIP 形式のライブラリをインストール」をクリックします。



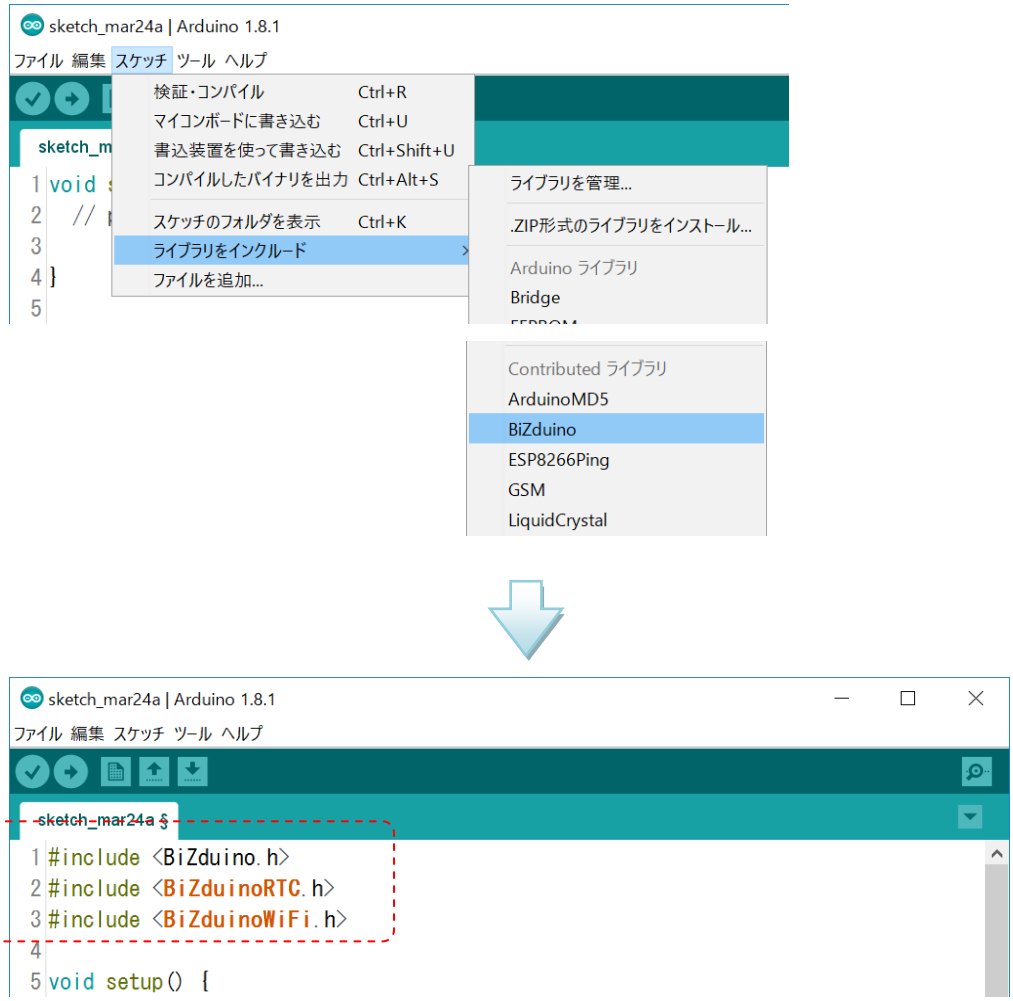
3. 以下の画面で先ほどダウンロードした ZIP ファイルを選択します。



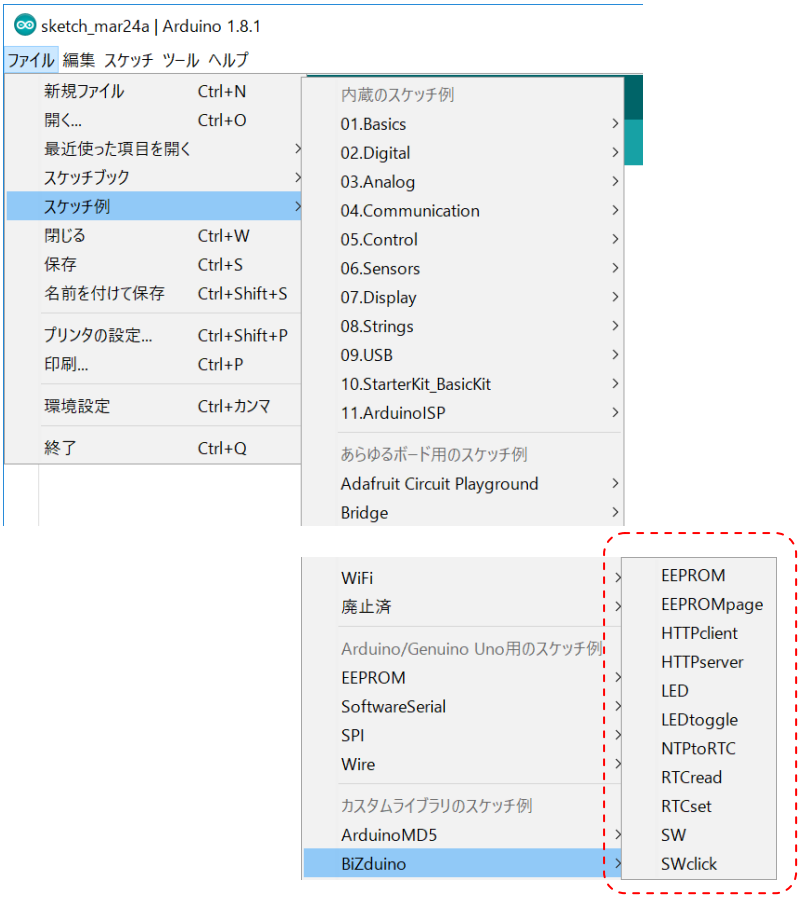
4. インストールされると“ライブラリが追加されました。「ライブラリをインクルード」メニューを確認してください。”とメッセージが表示されます。
   IDE のメニューから「スケッチ」→「ライブラリをインクルード」→「BiZduino」が追加されていることを確認します。

5. 以上で BiZduino ライブラリのインストール完了です。

6. ライブラリを使用するには、IDE のメニューから「スケッチ」→「ライブラリをインクルード」で "Contributed ライブラリ" 以下の「BiZduino」を選択します。
   スケッチに#include の行が追加されます。



※ サンプルスケッチは、IDE のメニューから「ファイル」→「スケッチ例」で "カスタムライブラリのスケッチ例" 以下の「BiZduino」にあります。
各スケッチ例の説明は別紙の BiZduino ライブラリ リファレンスを参照してください。



16. Wi-Fi モジュールへのスケッチ書き込み方法

Wi-Fi モジュール ESP-WROOM-02 はスケッチを書き込むことで単独でもマイコンとして動作でき、モジュールの GPIO が使用できます。
開発環境のセットアップ手順は開発環境構築手順書をご覧ください。

※ スケッチを書き込むと AT コマンドは使用できなくなります。
AT コマンドが使用できるように戻すにはファームウェア書き換えが必要です。

※ ATmega328P は 5V、ESP-WROOM-02 は 3.3V で動作します。

※ Wi-Fi モジュールと RTC は接続されていません。

※ Wi-Fi モジュールの公式資料は以下の URL からアクセスできます。
https://github.com/esp8266/Arduino

※ サンプルスケッチは、メニューから「ファイル」→「スケッチ例」で "Generic ESP8266 Module 用のスケッチ例" 以下にあります。

1. HTTP サーバー動作のサンプルスケッチ
"Generic ESP8266 Module 用のスケッチ例" - ESP8266WiFi > WiFiWebServer

2. HTTP クライアント動作のサンプルスケッチ
"Generic ESP8266 Module 用のスケッチ例" - ESP8266WiFi > WiFiClient

※ Wi-Fi アクセスポイント環境に合わせてスケッチにある SSID とパスワードを変更してください。

3. NTP クライアント動作のサンプルスケッチ
"Generic ESP8266 Module 用のスケッチ例" - ESP8266WiFi > NTPClient

※ Wi-Fi アクセスポイント環境に合わせてスケッチにある SSID とパスワードを変更してください。取得時刻は UTC のため日本時間の場合は+9:00:00 してください。

17. Wi-Fi モジュール用スケッチの例外エラー調査方法

Wi-Fi モジュール用スケッチで動作中に例外エラーが発生するとシリアルモニタ上で以下のように例外コードを表示します。

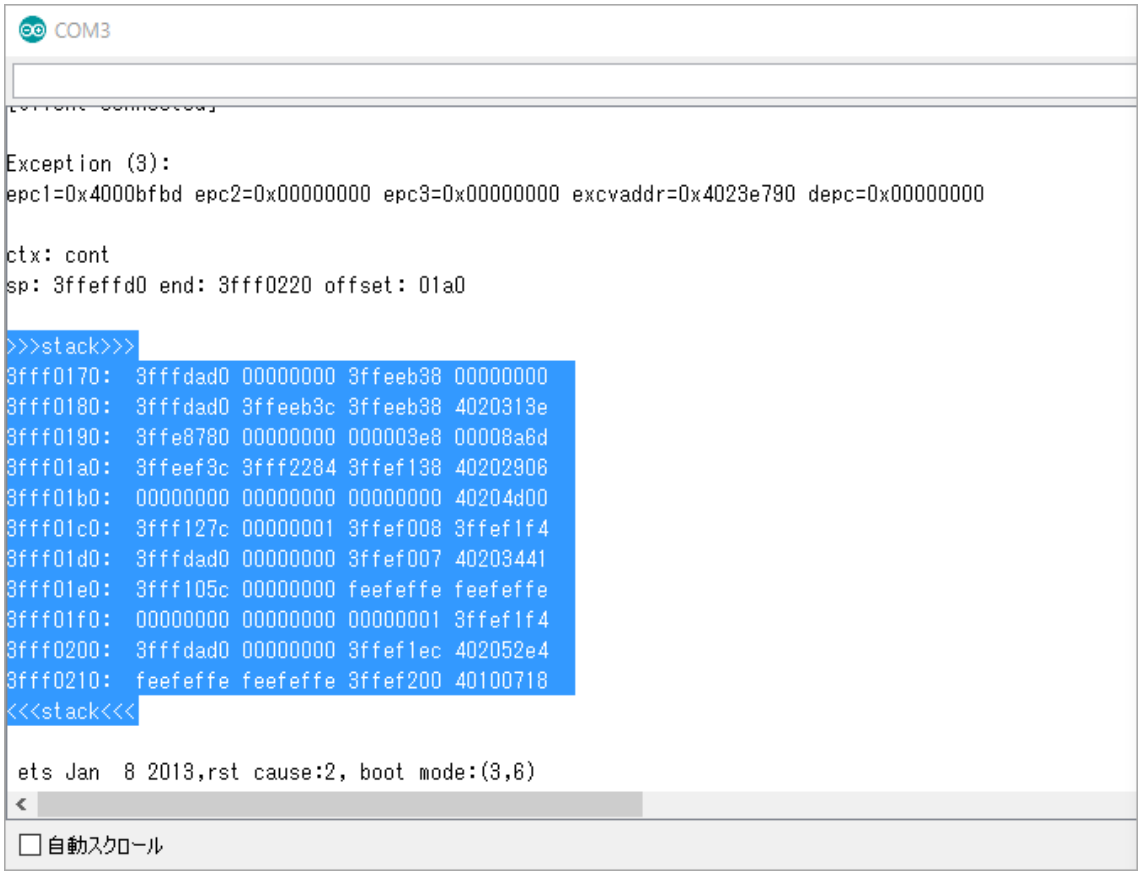
例外コード:

```
Exception (29):
```

例外コードの詳細は以下の URL で調べることができます。
https://github.com/esp8266/Arduino/blob/master/doc/exception_causes.md

例外エラーの発生箇所は、Exception Stack Trace Decoder で以下のように調べることができます。

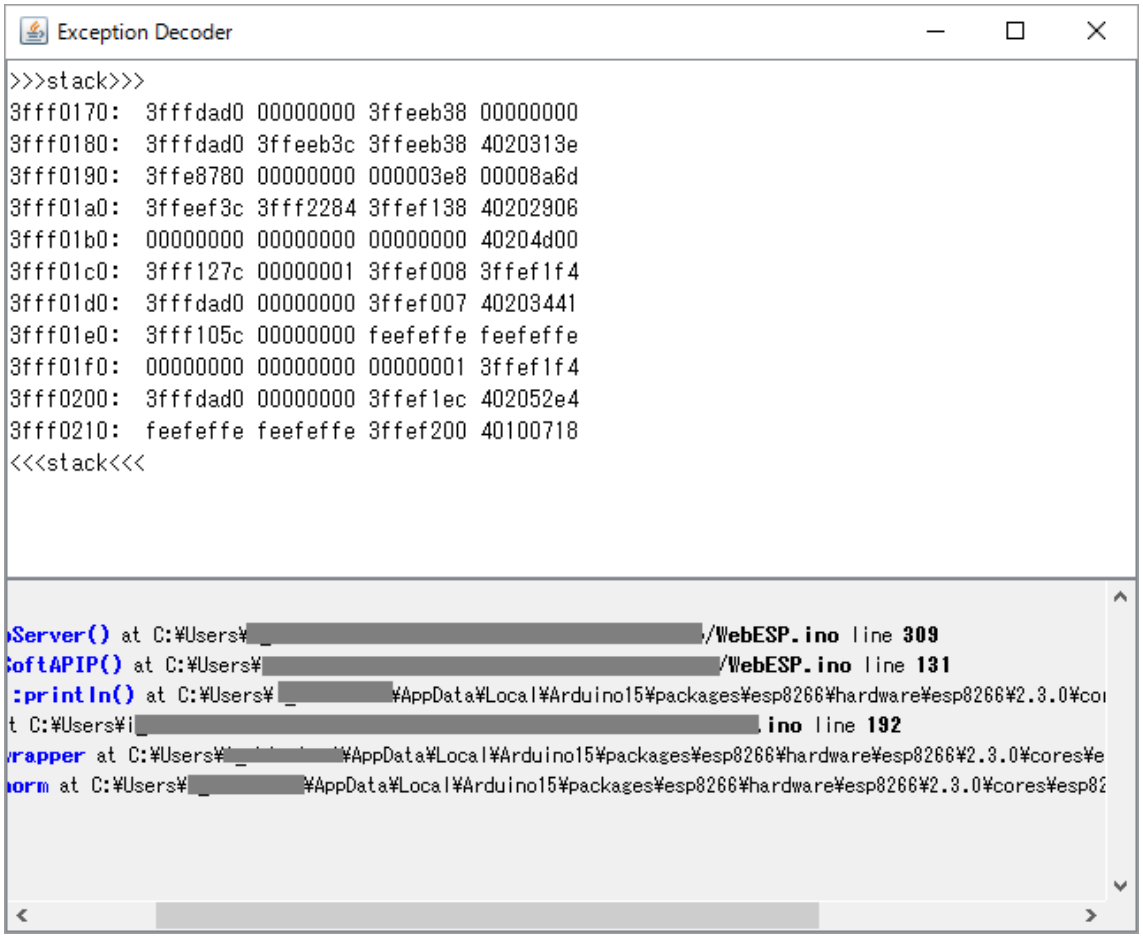
1. シリアルモニタに表示された ">>>stack>>>" と "<<<stack<<<" で囲まれた部分をコピーします。



2. Arduino IDE メニューから「ツール」→「ESP Exception Decoder」をクリックします。

3. Exception Decoder 画面上部にコピーした内容を貼り付けます。

4. Exception Decoder 画面下部に例外エラーが発生した箇所の情報が表示されます。



※ 詳細な資料は以下の URL からアクセスできます。
https://github.com/esp8266/Arduino/blob/master/doc/faq/a02-my-esp-crashes.md

18. Wi-Fi モジュールのファームウェア書き換え方法

Wi-Fi モジュールのファームウェアは、チップの供給時に最新バージョンではない場合があります。供給時のバージョンに無い新機能を使用する場合にはファームウェア書き換えを行ってください。
また、Wi-Fi モジュールにスケッチ書き込み後、AT コマンドが使用できるように戻す場合もファームウェア書き換えを行ってください。

1. 以下の URL から NONOS SDK の最新バージョンをダウンロードします。
https://www.espressif.com/en/support/download/sdks-demos?keys=&field_type_tid%5B%5D=14

2. ダウンロードしたファイルを展開し、ESP8266_NONOS_SDK¥bin 内を C:¥esp8266_bin フォルダへ移動します。

3. 以下の URL から書き込み用ツール Flash Download Tools をダウンロードし、任意の場所へ展開します。
https://www.espressif.com/en/support/download/other-tools?keys=&field_type_tid%5B%5D=14

4. BiZduino の JH2 の 3-4,5-6 にジャンパーピンを付けて JP7 のジャンパーピンを外し、Wi-Fi ダイレクトモードにして PC と USB ケーブルで接続します。

5. Arduino IDE のシリアルモニタから以下の AT コマンドでファームウェアのバージョンを確認します。

```
AT+GMR
AT version:0.52.0.0(Jan  7 2016 18:44:24)
SDK version:1.5.1(e67da894)
compile time:Jan  7 2016 19:03:11
OK

AT+CWMODE_CUR=3
OK

AT+CIPAPMAC_CUR?
+CIPAPMAC:"XX:XX:XX:XX:XX:XX"
OK

AT+CIPSTAMAC_CUR?
+CIPSTAMAC:"XX:XX:XX:XX:XX:XX"
OK
```

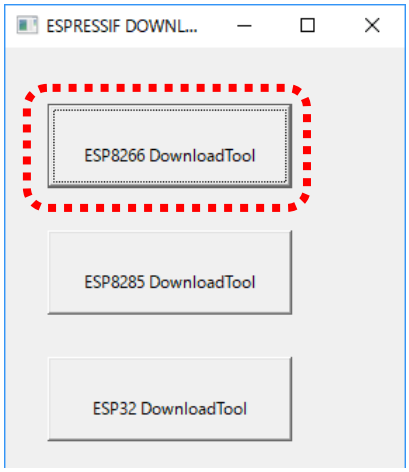
※ MAC アドレスは後の手順で必要になるので紙などにメモしてください。

6. シリアル通信の設定を変更している場合、以下の AT コマンドで 115200 bps に戻します。

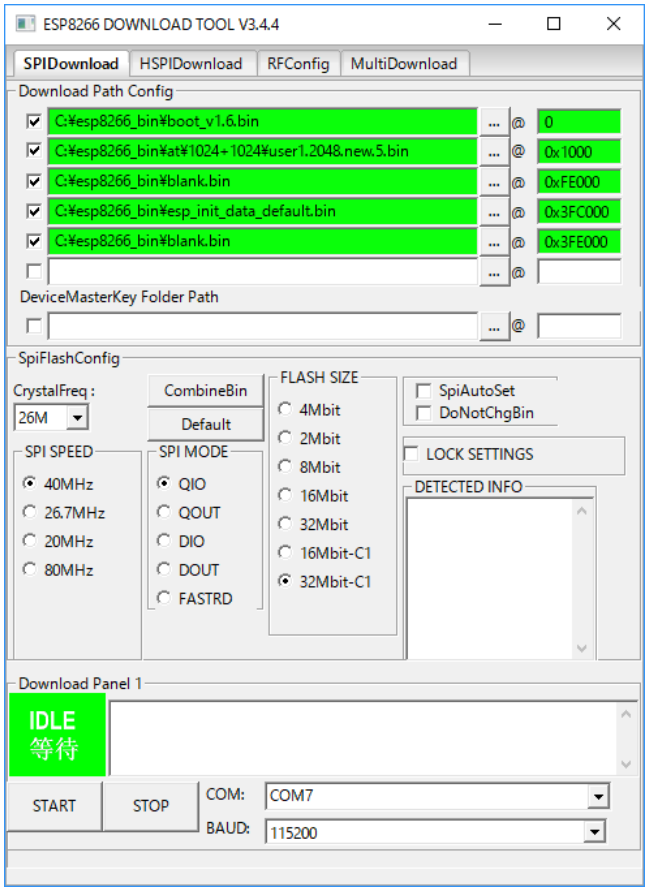
```
AT+UART_DEF=115200,8,1,0,0
OK
```

7. JP3 にジャンパーピンを付けて、SW2 を押すとファームウェア書き換えモードになります。Arduino IDE は終了します。

8. 3.で追加した Flash Download Tools を起動し、"ESP8266 DownloadTool"をクリックします。



9. SPIDownload タブ内で以下のように設定して "START" をクリックしてください。



Download Path Config

Path	Address
C:¥esp8266_bin¥boot_v1.6.bin	0
C:¥esp8266_bin¥at¥1024+1024¥user1.2048.new.5.bin	0x1000
C:¥esp8266_bin¥blank.bin	0xFE000
C:¥esp8266_bin¥esp_init_data_default.bin	0x3FC000
C:¥esp8266_bin¥blank.bin	0x3FE000

SpiFlashConfig

Config	Value
CrystalFreq	26M
SPI SPEED	40MHz
SPI MODE	QIO
FLASH SIZE	32Mbit-C1

Download Panel 1

Config	Value
COM	(使用するシリアルポート番号を選択)
BAUD	115200

10. 書き込みが終わるまでお待ちください。書き込みが終わると "FINISH" と表示されます。

11. 書き込み用ツール Flash Download Tools を終了します。

12. JP3 のジャンパーピンを外してから SW2 を押して Wi-Fi モジュールをリセットします。

13. Arduino IDE のシリアルモニタから以下の AT コマンドでファームウェアのバージョンを確認します。

```
AT+GMR
AT version:1.3.0.0(Jul 14 2016 18:54:01)
SDK version:2.0.0(656edbf)
compile time:Jul 19 2016 18:44:22
OK

AT+CWMODE_CUR=3
OK

AT+CIPAPMAC_CUR?
+CIPAPMAC:"XX:XX:XX:XX:XX:XX"
OK

AT+CIPSTAMAC_CUR?
+CIPSTAMAC:"XX:XX:XX:XX:XX:XX"
OK
```

※ MAC アドレスが書き換わった場合は AT コマンドの AT+CIPAPMAC_DEF や AT+CIPSTAMAC_DEF でメモしていたアドレスに戻してください。

14. 以上でファームウェア書き換え完了です。

19.トラブルシューティング

以下にトラブルシューティングを記載します。

1. 動作が不安定になる。
   電源供給が足りない場合に不安定な動作になることがあります。
   USB の給電が 5V 以上になっていることを確認してください。
   開発用 PC と USB 接続していて電圧が低い場合は、セルフパワーの USB ハブを経由して接続してください。

2. RTC の I2C 通信が不安定、もしくは通信できない。
   電池ホルダーに手で触れると RTC の I2C 通信に悪影響を与えることがあります。
   動作中は電池ホルダーに触れないようにしてください。

20. 改訂履歴

更新日	バージョン	内容
2017/02/15	1.0	初版
2017/04/04	1.1	BiZduino ライブラリの説明を追加