株式会社 ビズライト・テクノロジー

BiZduino ライブラリ
リファレンス

最終更新日: 2017/04/05

バージョン: 1.0

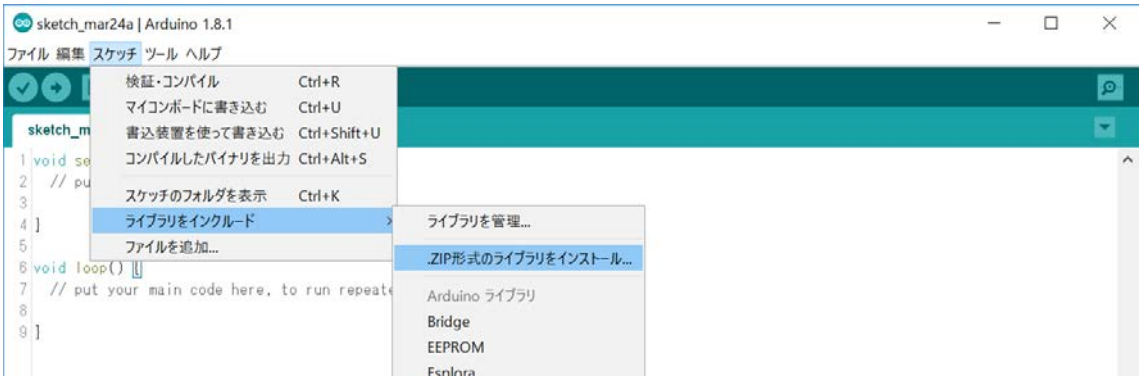
（株）ビズライト・テクノロジー

1. BiZduino ライブラリ使用方法

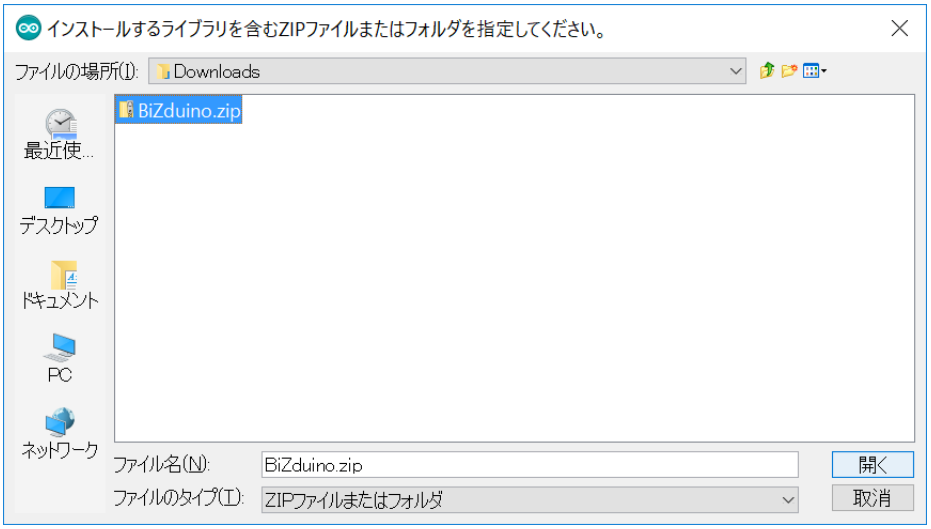
BiZduino の LED3、SW3、Wi-Fi モジュール、RTC モジュールを扱うための BiZduino ライブラリを用意しています。
BiZduino ライブラリを IDE へ追加する方法を以下に記します。

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。



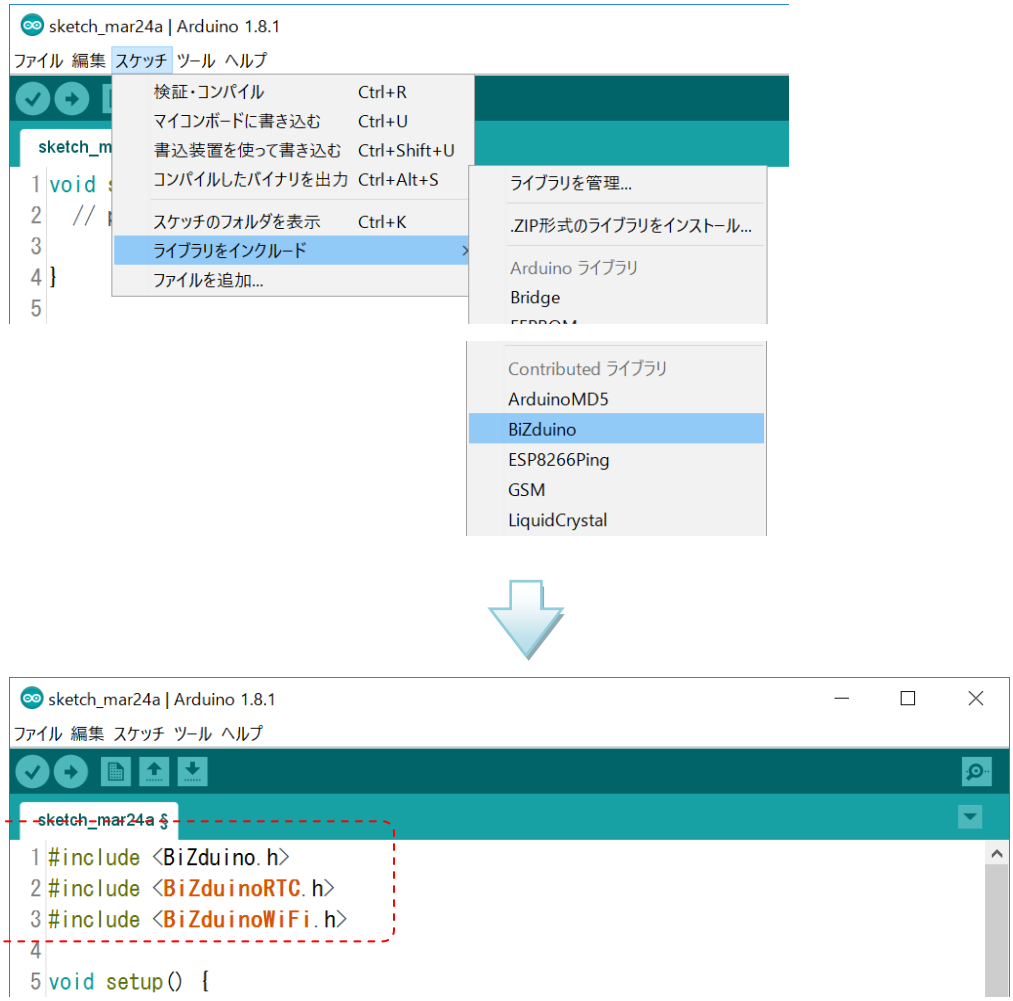
3. 以下の画面で先ほどダウンロードした ZIP ファイルを選択します。



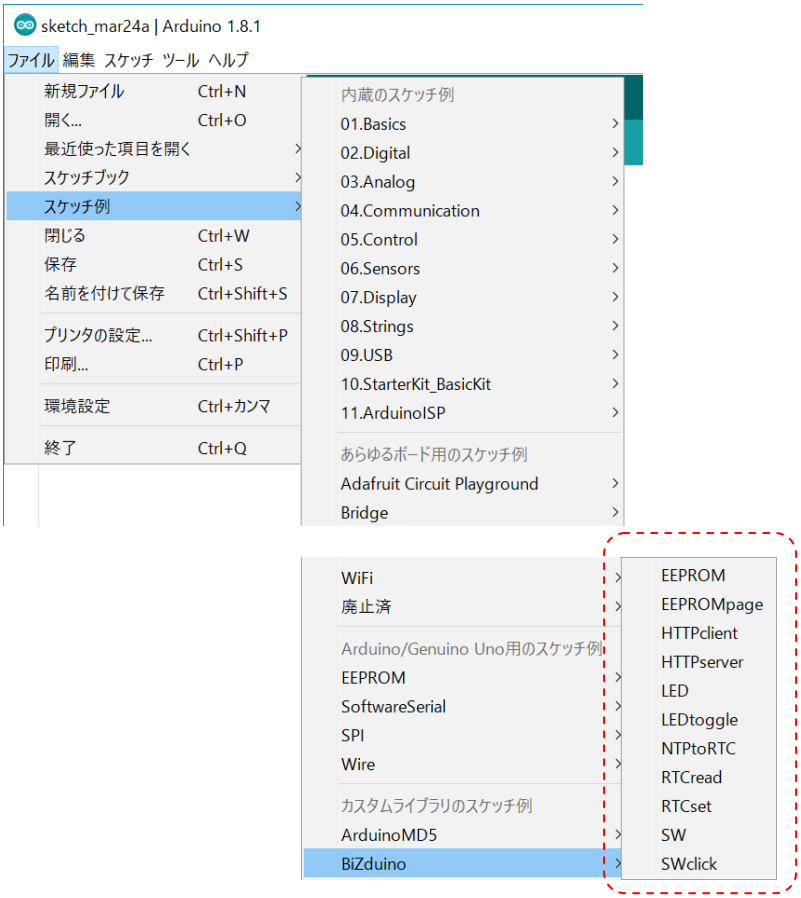
4. インストールされると“ライブラリが追加されました。「ライブラリをインクルード」メニューを確認してください。”とメッセージが表示されます。
   IDE のメニューから「スケッチ」→「ライブラリをインクルード」→「BiZduino」が追加されていることを確認します。

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次ページ以降に記します。



2. BiZduinoLED3 クラス

LED3 の点灯/消灯を行うクラスです。
(インクルードファイル:BiZduino.h)

メソッド名	説明
void on()	LED3 を点灯
void off()	LED3 を消灯
void toggle()	LED3 の点灯/消灯を反転

(1) BiZduinoLED3 スケッチ例 1
LED3 を 1 秒点灯、0.5 秒消灯を繰り返すサンプルスケッチです。

LED.ino

```
// LED3 を 1 秒点灯、0.5 秒消灯を繰り返すサンプルスケッチ

#include <BiZduino.h>

BiZduinoLED3 LED3;

void setup() {
}

void loop() {
  // LED3 を点灯
  LED3.on();

  // 1 秒待つ
  delay(1000);

  // LED3 を消灯
  LED3.off();

  // 0.5 秒待つ
  delay(500);
}
```

(2) BiZduinoLED3 スケッチ例 2

LED3 を 1 秒ごとに点灯、消灯を繰り返すサンプルスケッチです。

LEDtoggle.ino

```
// LED3 を 1 秒ごとに点灯、消灯を繰り返すサンプルスケッチ

#include <BiZduino.h>

BiZduinoLED3 LED3;

void setup() {
}

void loop() {
  // LED3 を切り替える
  LED3.toggle();

  // 1 秒待つ
  delay(1000);

}
```

3. BiZduinoSW3 クラス

SW3 の ON/OFF 状態を確認するクラスです。
(インクルードファイル:BiZduino.h)

メソッド名	説明
bool isPressed()	SW3 が押されているか判定 return: true=SW on / false=SW off
bool isClicked()	SW3 が押されたか判定 return: true=off 状態から on になった / false=その他

(1) BiZduinoSW3 スケッチ例 1
SW3 を押している間 LED3 が点灯するサンプルスケッチです。

SW.ino

```
// SW3 を押している間 LED3 が点灯するサンプルスケッチ

#include <BiZduino.h>

BiZduinoLED3 LED3;
BiZduinoSW3 SW3;

void setup() {
}

void loop() {

  // SW3 が押されているか判定
  if (SW3.isPressed()) {

    // SW3 が押されている場合は LED3 を点灯
    LED3.on();
  } else {

    // SW3 が押されていない場合は LED3 を消灯
    LED3.off();
  }

}
```

(2) BiZduinoSW3 スケッチ例 2

SW3 を押す度に LED3 の点灯と消灯を切り替えるサンプルスケッチです。

SWclick.ino

```
// SW3 を押す度に LED3 の点灯と消灯を切り替えるサンプルスケッチ

#include <BiZduino.h>

BiZduinoLED3 LED3;
BiZduinoSW3 SW3;

void setup() {
}

void loop() {

  // SW3 が押された時か判定
  if (SW3.isClicked()) {

    // LED3 を切り替える
    LED3.toggle();

  }

  // チャタリング防止ウエイト
  delay(50);
}
```

4. BiZduinoWiFi クラス

Wi-Fi モジュールを扱うクラスです。
(インクルードファイル:BiZduinoWiFi.h)

※ ITEADLIB Arduino WeeESP8266 ライブラリを元に作成しています。
https://github.com/itead/ITEADLIB_Arduino_WeeESP8266

メソッド名	説明
void begin(uint32_t baud = 115200)	Wi-Fi 設定 baud: シリアルのボーレート(デフォルト 115200)
bool enableSerial()	Wi-Fi モジュールをシリアルポートに接続 return: true=接続状態に切り替えた / false=すでに接続済
bool disableSerial()	Wi-Fi モジュールをシリアルポートから切断 return: true=切断状態に切り替えた / false=すでに切断済
void resetWiFi()	Wi-Fi モジュールをリセットする
bool kick()	Wi-Fi モジュール通信確認（"AT"） return: true=成功 / false=エラー
bool restart()	リスタートコマンド（"AT+RST"） return: true=成功 / false=エラー
String getVersion()	WiFi モジュールのバージョン情報取得 return: バージョン情報
bool setOprToStation()	ステーションモードに設定する return: true=成功 / false=エラー
bool setOprToSoftAP()	ソフト AP モードに設定する return: true=成功 / false=エラー
bool setOprToStationSoftAP()	ステーション+ソフト AP モードに設定する return: true=成功 / false=エラー

メソッド名	説明
String getAPList()	AP(アクセスポイント)リストを取得する return: AP リスト文字列 +CWLAP:(<ecn>,"<ssid>",<rssi>, "<mac>",<ch>,<freq offset>, <freq calibration>) ※AP が多いとメモリ不足で取得出来ない場合があります
bool joinAP(String ssid, String pwd)	アクセスポイントに接続する ssid: SSID pwd: パスワード return: true=成功 / false=エラー
bool enableClientDHCP(uint8_t mode, boolean enabled)	DHCP モードを設定する mode: サーバモード (0=soft AP, 1=station, 2=both) enabled:DHCP 有効 return: true=成功 / false=エラー
bool leaveAP()	アクセスポイントとの接続を切る return: true=成功 / false=エラー
bool setSoftAPParam(String ssid, String pwd, uint8_t chl = 7, uint8_t ecn = 4)	ソフト AP モードのパラメータを設定する ssid: SSID pwd: パスワード chl: チャンネル(1～13、デフォルト 7) ecn: 暗号方式 0=暗号なし / 1=WEP / 2=WPA_PSK / 3=WPA2_PSK / 4=WPA_WPA2_PSK(デフォルト) return: true=成功 / false=エラー
String getJoinedDeviceIP()	ソフト AP モードで接続しているデバイスの IP リストを取得する return: アドレスリスト文字列 "<IP addr>","<mac>"

メソッド名	説明
String getIPStatus()	接続ステータスを取得する return: 接続ステータス文字列 STATUS:<stat> +CIPSTATUS:<link ID>,<type>, <remote_IP>,<remote_port>, <local_port>,<tetype>
String getLocalIP()	ローカル IP リストを取得する return: ローカル IP リスト文字列 +CIFSR:STAIP,"<IP addr>" +CIFSR:STAMAC,"<mac>"
bool enableMUX()	マルチコネクションモードにする return: true=成功 / false=エラー
bool disableMUX()	シングルコネクションモードにする return: true=成功 / false=エラー
bool createTCP(String addr, uint32_t port)	TCP 接続(シングルコネクション) addr: アドレス port: ポート return: true=成功 / false=エラー
bool releaseTCP()	TCP 切断(シングルコネクション) return: true=成功 / false=エラー
bool registerUDP(String addr, uint32_t port)	UDP 接続(シングルコネクション) addr: アドレス port: ポート return: true=成功 / false=エラー
bool unregisterUDP()	UDP 切断(シングルコネクション) return: true=成功 / false=エラー
bool createTCP(uint8_t mux_id, String addr, uint32_t port)	TCP 接続(マルチコネクション) mux_id: マルチコネクション ID(0～4) addr: アドレス port: ポート return: true=成功 / false=エラー
bool releaseTCP(uint8_t mux_id)	TCP 切断(マルチコネクション) mux_id マルチコネクション ID(0～4) return: true=成功 / false=エラー

メソッド名	説明
bool registerUDP(uint8_t mux_id, String addr, uint32_t port)	UDP 接続(マルチコネクション) mux_id: マルチコネクション ID(0～4) addr: アドレス port: ポート return: true=成功 / false=エラー
bool unregisterUDP(uint8_t mux_id)	UDP 切断(マルチコネクション) mux_id: マルチコネクション ID(0～4) return: true=成功 / false=エラー
bool setTCPServerTimeout(uint32_t timeout = 180)	TCP サーバ時のタイムアウトを設定する timeout:タイムアウト秒 (0～28800、デフォルト 180) return: true=成功 / false=エラー
bool startTCPServer(uint32_t port = 333)	TCP サーバを開始 port: ポート(デフォルト 333) return: true=成功 / false=エラー ※マルチコネクションモードにする必要がある クライアント接続情報は getIPStatus()で取得する。サーバの受信待ち受けは uint32_t recv(uint8_t *coming_mux_id, uint8_t *buffer, uint32_t buffer_size, uint32_t timeout) を使用する
bool stopTCPServer()	TCP サーバを停止 return: true=成功 / false=エラー
bool startServer(uint32_t port = 333)	TCP サーバを開始 port: ポート(デフォルト 333) return: true=成功 / false=エラー
bool stopServer()	TCP サーバを停止 return: true=成功 / false=エラー

メソッド名	説明
bool send(const uint8_t *buffer, uint32_t len)	データ送信(シングルコネクション) buffer: データ len: データ長 return: true=成功 / false=エラー
bool send(uint8_t mux_id, const uint8_t *buffer, uint32_t len)	データ送信(マルチコネクション) mux_id: マルチコネクション ID(0～4) buffer: データ len: データ長 return: true=成功 / false=エラー
uint32_t recv(uint8_t *buffer, uint32_t buffer_size, uint32_t timeout = 1000)	データ受信(シングルコネクション) buffer: 受信バッファ buffer_size: 受信バッファ長 timeout: タイムアウト return: 受信データ長
uint32_t recv(uint8_t mux_id, uint8_t *buffer, uint32_t buffer_size, uint32_t timeout = 1000)	データ受信(マルチコネクション) mux_id: マルチコネクション ID(0～4) buffer: 受信バッファ buffer_size: 受信バッファ長 timeout: タイムアウト return: 受信データ長
uint32_t recv(uint8_t *coming_mux_id, uint8_t *buffer, uint32_t buffer_size, uint32_t timeout = 1000)	サーバデータ受信(マルチコネクション) coming_mux_id: マルチコネクション ID 格納先(0～4) buffer: 受信バッファ buffer_size: 受信バッファ長 timeout: タイムアウト return: 受信データ長

(1) BiZduinoWiFi スケッチ例 1

別の PC またはスマートフォンなど(以下、別端末)を、BiZduino と同じ Wi-Fi ネットワークに接続して、別端末から BiZduino に Web ブラウザでアクセス(※)すると「Hello world」が表示されるサンプルスケッチです。

※ シリアルモニタに URL が表示されるので、そこにアクセスしてください。

```
URL : http://192.168.XXX.XXX/
```

※ 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。

HTTPserver.ino

```
// HTTP サーバ動作のサンプルスケッチ
// 次の Wi-Fi に接続して、Web ブラウザでアクセスすると「Hello world」が表示される

#include <BiZduino.h>
#include <BiZduinoWiFi.h>

#define SSID        "<SSID>"
#define PASSWORD    "<PASSWORD>"

#define HTTP_PORT   80


BiZduinoWiFi wifi;  // WiFi インスタンス
BiZduinoLED3 LED3;  // LED3 インスタンス

void setup() {
  // WiFi 開始
  wifi.begin();

  // Station モードにする
  if (!wifi.setOprToStation()) {
    wifi.disableSerial();
    Serial.print("to station err¥r¥n");
  }

  // AP に接続する
  if (!wifi.joinAP(SSID, PASSWORD)) {
    wifi.disableSerial();
    Serial.print("Join AP failure¥r¥n");
  }
```

```
  // マルチコネクションモードにする
  if (!wifi.enableMUX()) {
    wifi.disableSerial();
    Serial.print("multiple err¥r¥n");
  }

  // TCP サーバ開始
  if (!wifi.startTCPServer(HTTP_PORT)) {
    wifi.disableSerial();
    Serial.print("start tcp server err¥r¥n");
  }

  // TCP サーバタイムアウト設定
  if (!wifi.setTCPServerTimeout(10)) {
    wifi.disableSerial();
    Serial.print("set tcp server timout err¥r¥n");
  }

  // ローカル IP アドレスを求める
  // (CIFSR 戻り値から「STAIP,"(IP addr)"」の部分を抜き出す
  String ip = "?";
  String list = wifi.getLocalIP();
  int32_t index1 = list.indexOf(F("STAIP,¥""));
  if (index1 != -1) index1 += 7;
  int32_t index2 = list.indexOf(F("¥""), index1);
  if (index1 != -1 && index2 != -1) {
    ip = list.substring(index1, index2);
  }

  wifi.disableSerial();
  Serial.print("¥r¥n¥n¥n");
  Serial.print("setup ok.¥r¥n");
  Serial.print("URL : http://");
  Serial.print(ip.c_str());
  Serial.print("/¥r¥n");

  wifi.enableSerial();
}

void loop() {
  char *header = "HTTP/1.1 200 OK¥r¥nContent-Type: text/html¥r¥n¥r¥n";
  char *content = "<html><head></head><body><h1>Hello world</h1></body></html>";
  char *notfound = "HTTP/1.1 404 Not Found¥r¥n¥r¥nNot Found!!";

  uint8_t buffer[128] = {0};
  uint8_t mux_id;
```

```
  // リクエスト受信待ち
  uint32_t len = wifi.recv(&mux_id, buffer, sizeof(buffer), 100);
  if (len > 0) {
    // LED3 点灯
    LED3.on();

    if (strncmp(buffer, "GET / ", 6) == 0) { // GET / のみ応答

      // ヘッダ送信
      wifi.send(mux_id, header, strlen(header));

      // コンテンツ送信
      wifi.send(mux_id, content, strlen(content));

    } else {

      // Not Found 送信
      wifi.send(mux_id, notfound, strlen(notfound));

    }

    // TCP コネクション切断
    wifi.releaseTCP(mux_id);

    // LED3 消灯
    LED3.off();
  }
}
```

(2) BiZduinoWiFi スケッチ例 2

SW3 ボタンを押すと HTTP 取得するサンプルスケッチです。

正常に取得できるとシリアルモニタに以下のように表示されます。

```
Received:[HTTP/1.1 200 OK
Date: Sun, 01 Jan 2017 00:00:00 GMT
Server:
Last-Modified: Sun, 01 Jan 2017 00:00:00 GMT
ETag: "00000-0-00000000"
Accept-Ranges: bytes
Content-Length: 1
Content-Type: text/plain

1]
```

※ 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

```
// HTTP クライアント動作のサンプルスケッチ
// SW3 ボタンを押すと HTTP 取得します
// Wi-Fi アクセスポイント環境に合わせて次の SSID とパスワードを変更

#include <BiZduino.h>
#include <BiZduinoWiFi.h>

#define SSID        "<SSID>"
#define PASSWORD    "<PASSWORD>"
#define HOST_NAME   "dl.bizright.jp"
#define HOST_PORT   80
#define PATH        "/bd/sample1"


BiZduinoWiFi wifi;  // WiFi インスタンス
BiZduinoLED3 LED3;  // LED3 インスタンス
BiZduinoSW3 SW3;    // SW3 インスタンス

void setup() {
  // WiFi 開始
  wifi.begin();

  // WiFi モジュールのファームバージョン取得
  String ver = wifi.getVersion();

  // Station モードにする
```

```
  if (!wifi.setOprToStation()) {
    wifi.disableSerial();
    Serial.print("to station err¥r¥n");
  }

  // AP に接続する
  if (!wifi.joinAP(SSID, PASSWORD)) {
    wifi.disableSerial();
    Serial.print("Join AP failure¥r¥n");
  }

  // マルチコネクションモードにする
  if (!wifi.enableMUX()) {
    wifi.disableSerial();
    Serial.print("multiple err¥r¥n");
  }

  // ローカルアドレスリスト取得
  String addr = wifi.getLocalIP();

  wifi.disableSerial();
  Serial.print("¥r¥n¥n");

  // ファームバージョン表示
  Serial.print("FW Version: ");
  Serial.println(ver.c_str());

  // ローカルアドレスリスト表示
  Serial.print("Address: ");
  Serial.println(addr.c_str());

  Serial.print("setup end¥r¥n");

}

void loop() {
  static uint8_t mux_id = 0;
  uint8_t buffer[256] = {0};

  // SW3 が押されたか判定
  if (SW3.isClicked()) {

    // LED3 点灯
    LED3.on();

    // TCP 接続
    if (wifi.createTCP(mux_id, HOST_NAME, HOST_PORT)) {
```

```
      // HTTP リクエスト送信
      char *req = "GET " PATH " HTTP/1.1¥r¥nHost: " HOST_NAME "¥r¥n
                        User-Agent: arduino¥r¥n¥r¥n";
      wifi.send(mux_id, (const uint8_t*)req, strlen(req));

      // HTTP レスポンス受信
      uint32_t len = wifi.recv(mux_id, buffer, sizeof(buffer), 10000);

      // TCP 切断
      wifi.releaseTCP(mux_id);

      wifi.disableSerial();
      Serial.print("Data Receive ");
      Serial.print(mux_id);
      Serial.println(" ok");
      Serial.print("data length=");
      Serial.print(len);
      Serial.println("");
      if (len > 0) {
        Serial.print("Received:[");
        for (uint32_t i = 0; i < len; i++) {
          Serial.print((char)buffer[i]);
        }
        Serial.print("]¥r¥n");
      }
    } else {
      wifi.disableSerial();
      Serial.print("create tcp ");
      Serial.print(mux_id);
      Serial.println(" err");
    }

    // マルチリンク ID 変更
    mux_id++;
    if (mux_id >= 5) {
      mux_id = 0;
    }

    // LED3 消灯
    LED3.off();
  }
}
```

5. BiZduinoRTC クラス、DateTime クラス

RTC モジュールを扱うクラスです。
(インクルードファイル:BiZduinoRTC.h)

※名前'RTC'のスタティックインスタンスが定義されます。

1. DateTime クラス
   時刻データを扱うクラスです。
   (インクルードファイル:BiZduinoRTC.h)

メソッド名	説明
DateTime (uint32_t t)	コンストラクタ t: エポックタイム(タイムゾーン込みの値)
DateTime (uint16_t year, uint8_t month, uint8_t day, uint8_t hour, uint8_t min, uint8_t sec)	コンストラクタ year: 年 month: 月 day: 日 hour: 時 min: 分 sec: 秒
DateTime (const char* date, const char* time)	コンストラクタ date: 年月日の文字列 time: 時分秒の文字列
uint16_t year()	年を取得
uint8_t month()	月を取得
uint8_t day()	日を取得
uint8_t hour()	時を取得
uint8_t minute()	分を取得
uint8_t second()	秒を取得
uint8_t dayOfWeek()	曜日を取得 return: 0=日, 1=月, 2=火, 3=水, 4=木, 5=金, 6=土

2. BiZduinoRTC クラス
   RTC モジュールを扱うクラスです。
   (インクルードファイル:BiZduinoRTC.h)

※ 名前'RTC'のスタティックインスタンスが定義されます。

メソッド名	説明
static void begin()	初期化
static bool read(DateTime& dt)	時刻取得(DateTime 型) dt: 時刻の格納先 return: true=成功 / false=エラー
static bool adjust(const DateTime& dt)	時刻設定(DateTime 型) dt: 設定時刻の格納先 return: true=成功 / false=エラー
static bool chipPresent()	データ有効判定 return: true=データ有効 / false=データ無効
static uint16_t milliSecond()	get()/read()実行時のミリ秒を取得する return: ミリ秒
static bool writeEEPROMPage(uint8_t page, uint8_t *data)	RTC EEPROM ページを指定して書き込み page: ページ位置 (8 バイト単位のブロック、0～63) data: 書き込みデータ(8 バイト) return: true=成功 / false=エラー
static bool readEEPROMPage(uint8_t page, uint8_t *data)	RTC EEPROM ページを指定して読み取り page: ページ位置 (8 バイト単位のブロック、0～63) data: 読み取りデータ格納先(8 バイト) return: true=成功 / false=エラー
static bool writeEEPROM(uint16_t pos, uint8_t c)	RTC EEPROM へ 1byte 書き込み pos: バイト位置(0～511) c: 書き込み値 return: true=成功 / false=エラー
static uint8_t readEEPROM(uint16_t pos)	RTC EEPROM から 1byte 読み取り pos: バイト位置(0～511) return: 読み取り値

(1) BiZduinoRTC スケッチ例 1

RTC の時刻を取得してシリアルモニタに 1 秒おきに表示するサンプルスケッチです。

RTCread.ino

```
// RTC の時刻を取得するサンプルスケッチ

#include <BiZduinoRTC.h>

// 曜日文字列
const char *weekName[7] = {
  "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"
};

DateTime dt;

void setup() {
  // シリアルモニタとの通信を開始
  Serial.begin(115200);
  RTC.begin();
}

void loop() {
  int wait = 1000;

  // RTC の時刻を取得
  if (RTC.read(dt)) {

    Serial.print(dt.year());     // 年の表示
    Serial.write('/');
    print2digits(dt.month());    // 月の表示
    Serial.write('/');
    print2digits(dt.day());      // 日の表示
    Serial.write(' ');
    Serial.print(weekName[dt.dayOfWeek()]); // 曜日の表示
    Serial.write(' ');
    print2digits(dt.hour());     // 時の表示
    Serial.write(':');
    print2digits(dt.minute());   // 分の表示
    Serial.write(':');
    print2digits(dt.second());   // 秒の表示
    Serial.write('.');
    print3digits(RTC.milliSecond()); // ミリ秒表示 (RTC の精度は 1/100 秒単位です)
    Serial.println();

    wait = 1000 - RTC.milliSecond(); // ミリ秒値でディレイ時間を調整
```

```
  } else {
    if (RTC.chipPresent()) {
      // RTC データエラー
      Serial.println("RTC is stopped.");
      Serial.println();
    } else {
      // RTC 通信エラー
      Serial.println("RTC read error!");
      Serial.println();
    }
  }

  delay(wait);
}

// 2 桁表示
void print2digits(int number) {
  if (number >= 0 && number < 10) {
    Serial.write('0');
  }
  Serial.print(number);
}

// 3 桁表示
void print3digits(int number) {
  if (number >= 0 && number < 100) {
    Serial.write('0');
  }
  if (number >= 0 && number < 10) {
    Serial.write('0');
  }
  Serial.print(number);
}
```

(2) BiZduinoRTC スケッチ例 2

RTC に時刻を設定するサンプルスケッチです。

設定が成功するとシリアルモニタに以下のように表示します。

```
configured Date=Jan  1 2017, Time=00:00:00
RTC.adjust()
2017/01/01 Sun 00:00:00
```

※ コンパイル日時(__DATE__と__TIME__)を設定しています。

RTCset.ino

```
// RTC に時刻を設定するサンプルスケッチ

#include <BiZduinoRTC.h>

// 曜日文字列
const char *weekName[7] = {
  "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"
};
// 月文字列
const char *monthName[12] = {
  "Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
};

DateTime dt;

void setup() {
  // シリアルモニタとの通信を開始
  Serial.begin(115200);
  RTC.begin();

  Serial.print("configured Date=");
  Serial.print(__DATE__);
  Serial.print(", Time=");
  Serial.println(__TIME__);

  // コンパイル日時を時刻に変換
  getDateTime(__DATE__, __TIME__);

  ////////////////////////////////////////////////////
  // RTC 設定

  Serial.println("RTC.adjust()");
```

```
  if (RTC.adjust(dt)) {
    // RTC の時刻を取得して表示
    printRTC();
  } else {
    // RTC エラー
    Serial.println("RTC set error!");
  }
}

void loop() {
}


// RTC の時刻を取得して表示
void printRTC() {

  // RTC の時刻を取得
  if (RTC.read(dt)) {

    Serial.print(dt.year());     // 年の表示
    Serial.write('/');
    print2digits(dt.month());    // 月の表示
    Serial.write('/');
    print2digits(dt.day());      // 日の表示
    Serial.write(' ');
    Serial.print(weekName[dt.dayOfWeek()]); // 曜日の表示
    Serial.write(' ');
    print2digits(dt.hour());     // 時の表示
    Serial.write(':');
    print2digits(dt.minute());   // 分の表示
    Serial.write(':');
    print2digits(dt.second());   // 秒の表示
    Serial.println();
  } else {
    Serial.println("RTC read error!");
  }
}

// 2 桁表示
void print2digits(int number) {
  if (number >= 0 && number < 10) {
    Serial.write('0');
  }
  Serial.print(number);
}

// 日付文字列と時刻文字列から年月日時分秒に変換
bool getDateTime(const char *strDate, const char *strTime)
```

```
{
  char Month[12];
  int Day, Year;
  uint8_t monthIndex;
  int Hour, Min, Sec;
  if (sscanf(strDate, "%s %d %d", Month, &Day, &Year) != 3) {
    return false;
  }
  if (sscanf(strTime, "%d:%d:%d", &Hour, &Min, &Sec) != 3) {
    return false;
  }
  for (monthIndex = 0; monthIndex < 12; monthIndex++) {
    if (strcmp(Month, monthName[monthIndex]) == 0) break;
  }
  if (monthIndex >= 12) {
    return false;
  }
  dt = DateTime(Year, monthIndex + 1, Day, Hour, Min, Sec);
  return true;
}
```

(3) BiZduinoRTC スケッチ例 3

NTP から時刻取得して RTC に設定するサンプルスケッチです。
通常時は RTC の時刻を取得してシリアルモニタに 1 秒おきに表示し、SW3 ボタンを押すと NTP サーバから時刻を取得して RTC に設定します。

※ 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。

NTPtoRTC.ino

```
// NTP から時刻取得して RTC に設定するサンプルスケッチ
// SW3 ボタンを押すと NTP サーバから時刻を取得して RTC に設定します
// Wi-Fi アクセスポイント環境に合わせて次の SSID とパスワードを変更

#include <BiZduino.h>
#include <BiZduinoWiFi.h>
#include <BiZduinoRTC.h>

#define SSID        "<SSID>"
#define PASSWORD    "<PASSWORD>"

// NTP のサーバ名
#define NTP_HOST    "ntp.nict.jp"
// NTP のポート番号
#define NTP_PORT    123

// タイムゾーン補正値(秒)
#define TIME_ZONE 9 * 60 * 60

// NTP パケットサイズ
#define NTP_PACKET_SIZE 48

#define NTP_VERSION_4     0b00100000
#define NTP_MODE_CLIENT   0b00000011


BiZduinoWiFi wifi;  // WiFi インスタンス
BiZduinoLED3 LED3;  // LED3 インスタンス
BiZduinoSW3 SW3;    // SW3 インスタンス

// 曜日文字列
const char *weekName[7] = {
  "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"
};
```

```
DateTime dt;

void setup() {
  // WiFi 開始
  wifi.begin();
  RTC.begin();

  // Station モードにする
  if (!wifi.setOprToStation()) {
    wifi.disableSerial();
    Serial.print("to station err¥r¥n");
  }
  // シングルコネクションモードにする
  if (!wifi.disableMUX()) {
    wifi.disableSerial();
    Serial.print("single err¥r¥n");
  }

  // WiFi モジュールのシリアルを切る
  wifi.disableSerial();

}

void loop() {
  static unsigned long lastMilli = 0;
  static uint8_t mux_id = 0;
  uint8_t buffer[256] = {0};

  // SW3 が押されているか判定
  if (SW3.isClicked()) {

    // LED3 点灯
    LED3.on();

    // AP に接続する
    if (wifi.joinAP(SSID, PASSWORD)) {

      // NTP からエポックタイムを取得
      unsigned long epoch = getNTP();

      // タイムゾーンを足して RTC に設定
      dt = DateTime(epoch + TIME_ZONE);
      if (RTC.adjust(dt)) {
        wifi.disableSerial();
        Serial.println("RTC set success!");
      } else {
        wifi.disableSerial();
```

```
        Serial.println("RTC set error!");
      }

      // AP を切断
      wifi.leaveAP();

    } else {
      wifi.disableSerial();
      Serial.print("Join AP failure¥r¥n");
    }

    // LED3 消灯
    LED3.off();

    // WiFi モジュールのシリアルを切る
    wifi.disableSerial();

  } else {

    unsigned long nowMilli = millis();
    if ((nowMilli - lastMilli) >= 1000) {  // 経過時間

      // RTC の時刻表示
      printRTC();

      lastMilli = nowMilli;
    }

  }
}

// NTP から時刻を取得する
unsigned long getNTP()
{
  unsigned long epoch = 0;

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

  if (wifi.registerUDP(NTP_HOST, NTP_PORT)) {
    memset(packetBuffer, 0, NTP_PACKET_SIZE);
    packetBuffer[0] = NTP_VERSION_4 | NTP_MODE_CLIENT;  // LI, Version, Mode
    wifi.send(packetBuffer, NTP_PACKET_SIZE);

    uint32_t len = wifi.recv(packetBuffer, NTP_PACKET_SIZE, 10000);

    if (len == NTP_PACKET_SIZE) {
      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;
      epoch = secsSince1900 - seventyYears;   // UNIX タイムスタンプ
    }

    wifi.unregisterUDP();

  } else {
    wifi.disableSerial();
    Serial.println("register udp err");
  }

  return epoch;
}


// RTC の時刻表示
void printRTC() {

  // RTC の時刻を取得
  if (RTC.read(dt)) {

    Serial.print(dt.year());    // 年の表示
    Serial.write('/');
    print2digits(dt.month());   // 月の表示
    Serial.write('/');
    print2digits(dt.day());     // 日の表示
    Serial.write(' ');
    Serial.print(weekName[dt.dayOfWeek()]); // 曜日の表示
    Serial.write(' ');
    print2digits(dt.hour());    // 時の表示
    Serial.write(':');
    print2digits(dt.minute());  // 分の表示
    Serial.write(':');
    print2digits(dt.second());  // 秒の表示
    Serial.println();
  } else {
    if (RTC.chipPresent()) {
      // RTC データエラー
      Serial.println("RTC is stopped.");
      Serial.println();
    } else {
      // RTC 通信エラー
      Serial.println("RTC read error!");
      Serial.println();
    }
  }
```

```
}


// 2 桁表示
void print2digits(int number) {
  if (number >= 0 && number < 10) {
    Serial.write('0');
  }
  Serial.print(number);
}
```

(4) BiZduinoRTC スケッチ例 4

RTC の EEPROM のサンプルスケッチです。

起動時に全領域のデータをシリアルモニタに表示します。

```
000 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
～
1F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

SW3 ボタンを押すと、A0 をサンプリングして EEPROM に書き込み、全領域に書き込んでシリアルモニタに表示します。

```
Sampling start
................................
000 : 6F 6A 6C 6B 65 6D 64 63 64 64 5F 65 5D 60 5E 5E
010 : 59 5E 5A 5A 55 58 53 59 52 55 50 56 4E 54 4D 53
～
1F0 : 4C 53 4E 50 50 50 4D 52 4F 4C 4F 4C 4F 4F 4E 4C
```

EEPROM.ino

```
// RTC の EEPROM のサンプルスケッチ
// SW3 ボタンを押すと、A0 をサンプリングして EEPROM に書き込み
// 全領域に書き込んでシリアルモニタに表示する

#include <BiZduino.h>
#include <BiZduinoRTC.h>

#define SAMPLING_TIME 50  // サンプリング間隔

BiZduinoLED3 LED3;  // LED3 インスタンス
BiZduinoSW3 SW3;    // SW3 インスタンス

void setup() {
  // シリアルモニタとの通信を開始
  Serial.begin(115200);
  RTC.begin();

  // EEPROM 全領域表示
  readAll();
}

void loop() {

  // SW3 が押された時か判定
  if (SW3.isClicked()) {

    // LED3 点灯
```

```
    LED3.on();

    Serial.println("Sampling start");

    // サンプリング開始
    for (int pos = 0 ; pos < 512 ; pos++) {
      unsigned long time = millis();

      int data = analogRead(0) / 4; // A0 の値を 0～255 にする

      // EEPROM に書き込み
      if (!RTC.writeEEPROM(pos, data)) {
        Serial.println();
        Serial.println("EEPROM write error.");
      }

      // 16 バイト毎にピリオド表示
      if (pos % 16 == 15)
        Serial.write('.');

      unsigned long elapsed = millis() - time;  // 経過時間

      // サンプリング間隔まで待つ
      if (elapsed < SAMPLING_TIME)
        delay(SAMPLING_TIME - elapsed);
    }
    Serial.println();

    // LED3 消灯
    LED3.off();

    // EEPROM 全領域表示
    readAll();

  }

  delay(50);
}

// RTC EEPROM 全領域を読取り表示
boolean readAll() {
  for (int pos = 0; pos < 512; pos++) {
    uint8_t data = RTC.readEEPROM(pos);

    if (pos % 16 == 0) {
      print2hex(pos / 16);
      Serial.print("0 :");
    }
```

```
    Serial.write(' ');
    print2hex(data);
    if (pos % 16 == 15) {
      Serial.println();
    }
  }
  return true;
}

// 2 桁 16 進表示
void print2hex(uint8_t number) {
  if (number < 16) {
    Serial.write('0');
  }
  Serial.print(number, 16);
}
```

(5) BiZduinoRTC スケッチ例 5

RTC の EEPROM の Page 書込/読込のサンプルスケッチです。

起動時に全領域のデータをシリアルモニタに表示します。

```
000 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
～
1F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

SW3 ボタンを押す度に EEPROM 全領域に同値を書き込んでシリアルモニタに表示します。(値は SW3 を押す度に+1 しています)

```
Write start
000 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
010 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
～
1F0 : 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
```

EEPROMpage.ino

```
// RTC の EEPROM の Page 書込/読込のサンプルスケッチ
// SW3 ボタンを押す度に EEPROM 全領域に同値を書き込んでシリアルモニタに表示する

#include <BiZduino.h>
#include <BiZduinoRTC.h>


BiZduinoLED3 LED3;  // LED3 インスタンス
BiZduinoSW3 SW3;    // SW3 インスタンス


void setup() {
  // シリアルモニタとの通信を開始
  Serial.begin(115200);
  RTC.begin();

  // EEPROM 全領域表示
  readAll();
}

void loop() {
  static uint8_t writeValue = 0;

  // SW3 が押された時か判定
  if (SW3.isClicked()) {

    // LED3 点灯
    LED3.on();
```

```
    Serial.println("Write start");

    // EEPROM 全領域書き込み
    writeAll(writeValue);

    // LED3 消灯
    LED3.off();

    // EEPROM 全領域表示
    readAll();

    writeValue++;
  }

  delay(50);
}

// RTC EEPROM 全領域に同じ値を書き込む
boolean writeAll(uint8_t val) {
  uint8_t buff[8];
  for (int i = 0; i < 8; i++) buff[i] = val;
  for (int page = 0; page < 64; page++) {
    if (!RTC.writeEEPROMPage(page, buff)) {
      Serial.println();
      Serial.println("EEPROM write error.");
      return false;
    }
  }
  return true;
}

// RTC EEPROM 全領域を読取り表示
boolean readAll() {
  uint8_t buff[8];
  for (int page = 0; page < 64; page++) {
    if (!RTC.readEEPROMPage(page, buff)) {
      Serial.println();
      Serial.println("EEPROM read error.");
      return false;
    }
    if (page % 2 == 0) {
      print2hex(page * 8 / 16);
      Serial.print("0 :");
    }
    for (int i = 0 ; i < 8 ; i++) {
      Serial.write(' ');
      print2hex(buff[i]);
```

```
    }
    if (page % 2 != 0) {
      Serial.println();
    }
  }
  return true;
}

// 2 桁 16 進表示
void print2hex(uint8_t number) {
  if (number < 16) {
    Serial.write('0');
  }
  Serial.print(number, 16);
}
```

6. 改訂履歴

更新日	バージョン	内容
2017/04/05	1.0	初版