エモグラス《拡張マニュアル》　第 1 版　190507

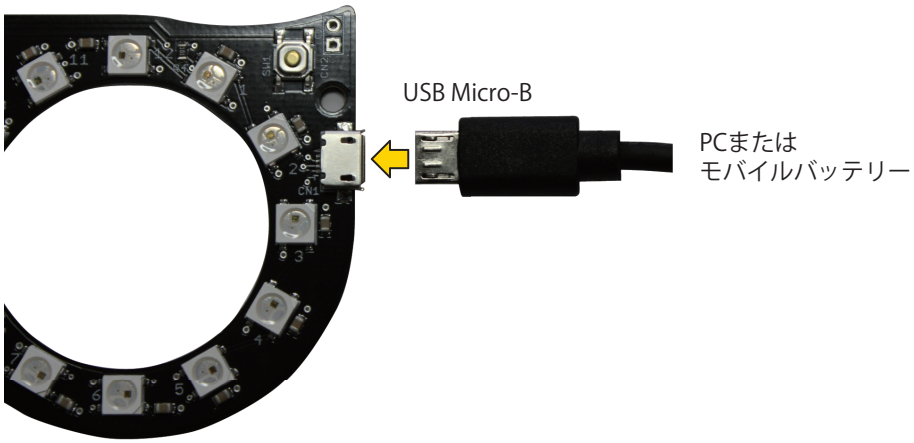
■ はじめに

エモグラスは 24 個のフルカラー LED をメガネ状に並べ、コントローラを配置した表示デバイスです。
電源を加える事であらかじめプログラムされている LED 点灯パターンを表示する事ができます。また、自作のプログラムで点灯パターンを変更できます。
（詳細は次ページ以降に「拡張機能」として記載しています）

■ 内蔵されているパターンを光らせる

エモグラス基板の向かって右側に、USB Micro B のコネクタが配置されています。
図 1 のように、このコネクタにモバイルバッテリーのような USB に対して電源を供給できる装置を接続してください。
消費電流は点灯状態で変化しますが 0.5A 以内で、殆どの USB 電源や PC の USB 端子から電源を確保する事ができます。
電源を接続して約 10 秒ほどで内蔵プログラムの動作が始まります。
（電源が接続された直後、内蔵プログラムの動作が開始されるまでの約 10 秒間は何もアクションがありません。電源接続の際は 10 秒程度お待ちください。なおこの時間が取られるのはエモグラスの互換設計のベースに採用している Digispark の仕様によるものです）

図 1: 電源の接続



＊ 1：PC 等の USB 端子は 0.5A の供給能力を持っています。
スマホに接続した OTG ケーブルや超小型ノート PC など、USB 端子の供給能力が 0.5A 未満の場合があります。これらの機器に接続した場合、最悪の場合機器を壊す恐れがあります。常用時の電源はモバイルバッテリーの利用をお勧めします。

- エモグラスは Arduino ベースのマイコンボード、「Digispark」を踏襲したプログラム（スケッチ）で動作しています。
  電源が投入された直後は、外部からプログラムが送り込まれる際の待機時間が取られています。この時間が約 10 秒で、その間は内蔵プログラムは動作せず USB 接続待ちのスタンバイ状態になっています。

- モバイルバッテリーには、負荷電流が少ないと自動で停止する機能を持った製品があります。
  エモグラスは点灯パターンで消費電流が大幅に変化します。消灯時間が長いプログラムを作る場合や、低輝度での発光時間が長いプログラムを作る場合は、モバイルバッテリーの自動停止機能が働かないかの確認をお願いします。
  （PC や USB ハブは消費電流による自動停止機能は無いので、動作確認が終わるまではこれらからの供給でお試しください）

■ 基板を固定するには

付属の「ビニール帯」を基板の左右それぞれの端にある穴に通し、お面やメガネなどに結びつけることでエモグラス基板を固定することができます。
（その他、針金などでも固定できますが、絶縁されていない剥き出しの針金を使う場合は基板を傷つけたりショートさせたりしないようご注意ください）

拡張機能に関する注意事項

本文書の以下のセクションは拡張に関する説明です。自作の表示パターンを作成する場合の参考としてご利用ください。
なお、拡張説明に記載の内容について共立電子産業は、「指定の動作にならない」あるいは「動作しない」等の障害発生に対し一切保証いたしません。またこのセクション記載の内容に関するご質問への回答はいたしかねます。
ご了解の上、自己責任にてご利用くださいますようお願いします。

■ プログラムの作成

エモグラスに搭載されているコントローラ「ATtiny85」をプログラムして、自作の点灯パターンを作成する手順です。
点灯パターンを作るには、ある程度のプログラミング知識を必要とします。
プログラムの作成は Arduino IDE を使用する場合の説明となります。

▶ エモグラスの部品構成

- コントローラ：ATtiny85 が 1 個
- フルカラー LED：WS2812B が 24 個

基本的な構成は以上です。その他の部品は補助的な役割で実装されていますが、表示の制御はこの二つのデバイスで行われています。（全体の回路構成は、本書末尾に掲載の回路図を参照してください）
WS2812B はフルカラー（RGB の 3 色発光）の LED です。各 3 色の発光は、一本のデータ線でコントロールされます。

さらにコントロール線は、全素子を個別に接続するのではなく、一本のコントロール線を数珠繋ぎに 1 番目から 2 番目、2 番目から 3 番目、…、24 番目まで順次接続する事で、全ての LED をコントロールします。エモグラスに搭載された 24 個の WS2812B をコントロールしているのは、ATtiny85 の出力線の内の一本だけです。

▶ プログラムの作成手順

エモグラスは「Arduino」と呼ばれるマイコンボードの派生品である「Digispark」基板と互換性がある構成にして出荷しています。このため、「Arduino」の開発環境があればエモグラス用のプログラムを作成する事ができます。

＊ 2：「Arduino」はオープンハードウエアと呼ぶ発想に従って作られた基板（CPU ボード）で、オープン（自由）なハードウエア（CPU ボード）を目指して作られています。
「Arduino」においては、基板の回路図が公開され、一定条件下で自由に利用する事ができます。また、ハードのみがオープンになっても使う方法が無ければ役に立たないため、自由に使えるソフトウエア開発環境も用意されています。エモグラスは「Arduino」から派生したオープンハードウエア「Digispark」の仕様に従って作られています。

手順その 1：Arduino IDE の入手とインストール

Arduino IDE は、下記ページからダウンロードする事ができます。開発環境は Windows、Mac のどちらでも動作させる事ができます。

- Arduino - Software
  `https://www.arduino.cc/en/Main/Software`

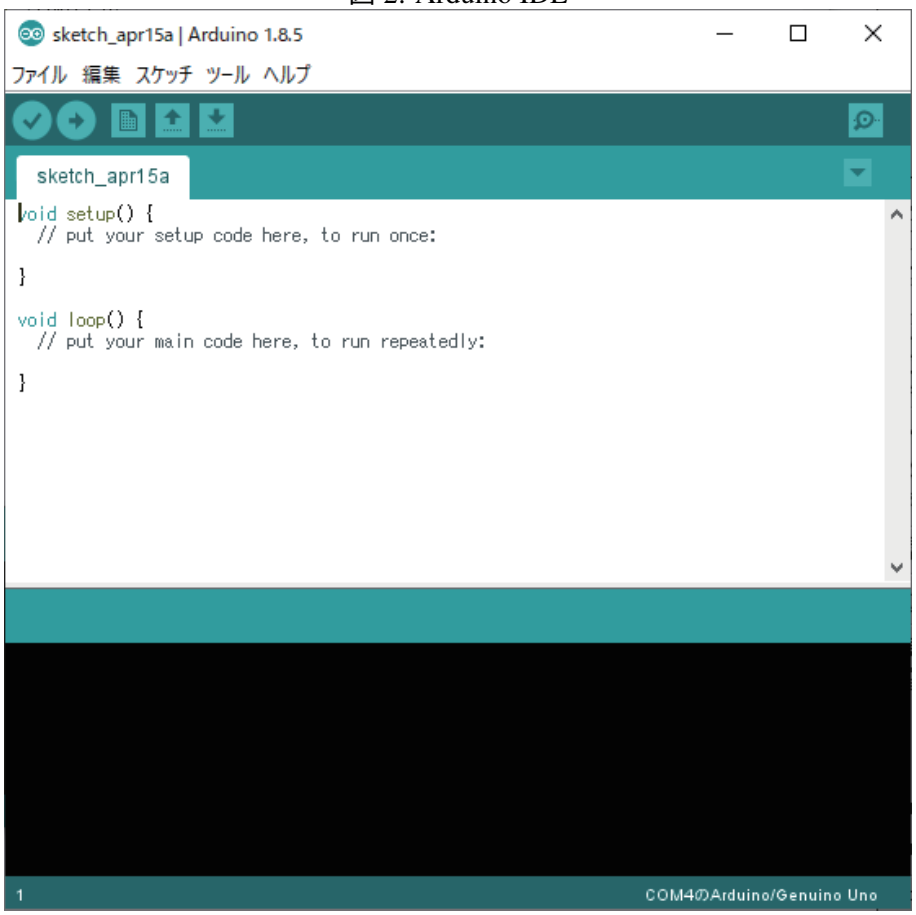
使う PC の種類にあわせて、ダウンロードする種類を右の表示から選択します。通常は「Windows Installer, for Windows XP and up」または「Mac OS X 10.8 Mountain Lion or newer」を選択します。
ダウンロードが完了すればインストールを行ってください。

＊ 3：エモグラス（Digispark）等の USB 通信ボードを Arduino から利用する場合のご注意
Arduino の通常の機種（UNO など）は、プログラムの転送にシリアルポート（COM ポート）を利用します。このため、Arduino IDE では、作成したプログラムの転送を行う際には少なくとも 1 つのシリアルポートが存在し、IDE 上でいずれか 1 つが選択されていなければならない仕様となっています。
一方、エモグラスを含めた Digispark 系列のボードは少々特殊な環境構成になっており、シリアルポートではなく USB を通信に利用して作成したプログラムを転送します。その際でも、上記の通りいずれか 1 つ無関係のシリアルポートを IDE 上から選択しなければ、プログラムの転送処理が進行されません（途中でシリアルポートが見つからない旨のエラーが表示されます）。
ノート PC などのようにハードウエアとしてシリアルポートを標準で持たない機種では、別途 USB シリアルコンバーターのようなシリアルポート（COM ポート）として認識できるデバイスをダミーとして接続する必要があります。

手順その 2：Digispark 開発環境の追加

Arduino IDE を起動させると、図 2 のような画面が表示されます。
インストール直後の Arduino IDE は色々な Arduino 基板を扱えるようになっていますが、初期状態では Digispark 基板が含まれていません。次の手順に従って Digispark 基板のサポートを追加してください。

図 2: Arduino IDE



1. 上部のファイルメニューから 環境設定 を選択してください。

2. 「環境設定」ウインドウが開きますので、追加のボードマネージャの URL の右端にあるアイコンをクリックしてください。

3. 「追加のボードマネージャの URL」ウインドウが開きますので、最終行に以下の行を追加します。
   `http://digistump.com/package_digistump_index.json`
   OK で確定させてください。(図 3)

4. 環境設定を OK で確定させてください。

5. ツールメニューのボード：XXXX の項目にマウスを乗せると右側にボードの種類が表示されます。その一番上 ボードマネージャ... を選択してください。
   「ボードマネージャ」ウインドウが開きますので、タイプのプルダウンリストで「提供された」または「contrib」を選択してください。(選択しなくても表示されますが、多数のボードが表示されます)
   表示されたボードの種類から Oak by Digistump by Digistump を選択して「インストール」をクリックしてください。完了後、閉じる で確定させてください。(図 4)

図 3: 環境設定

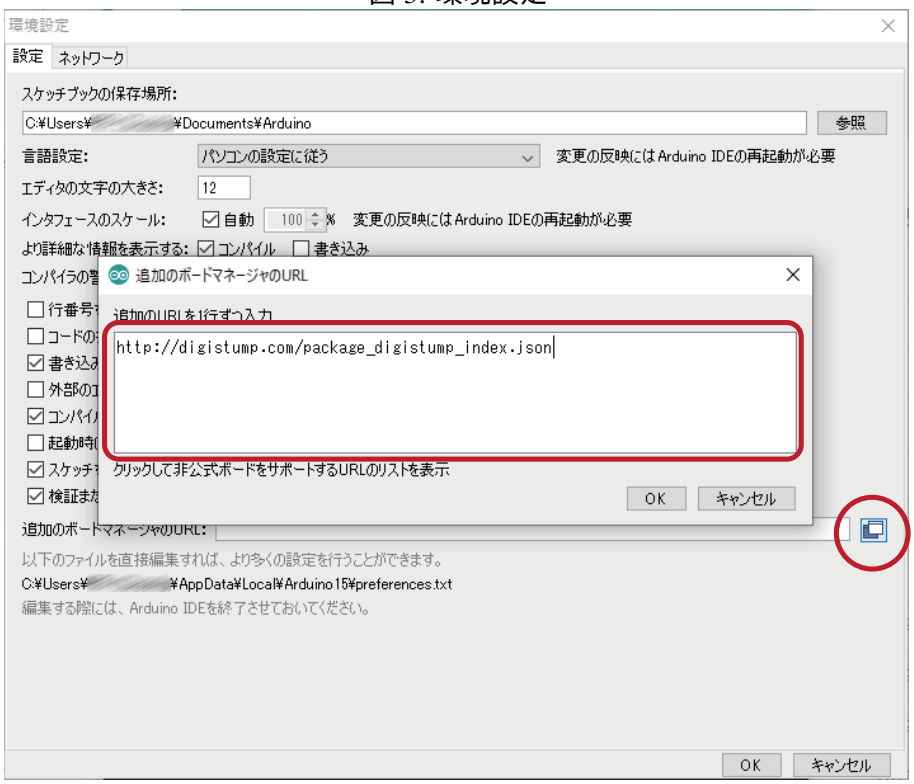
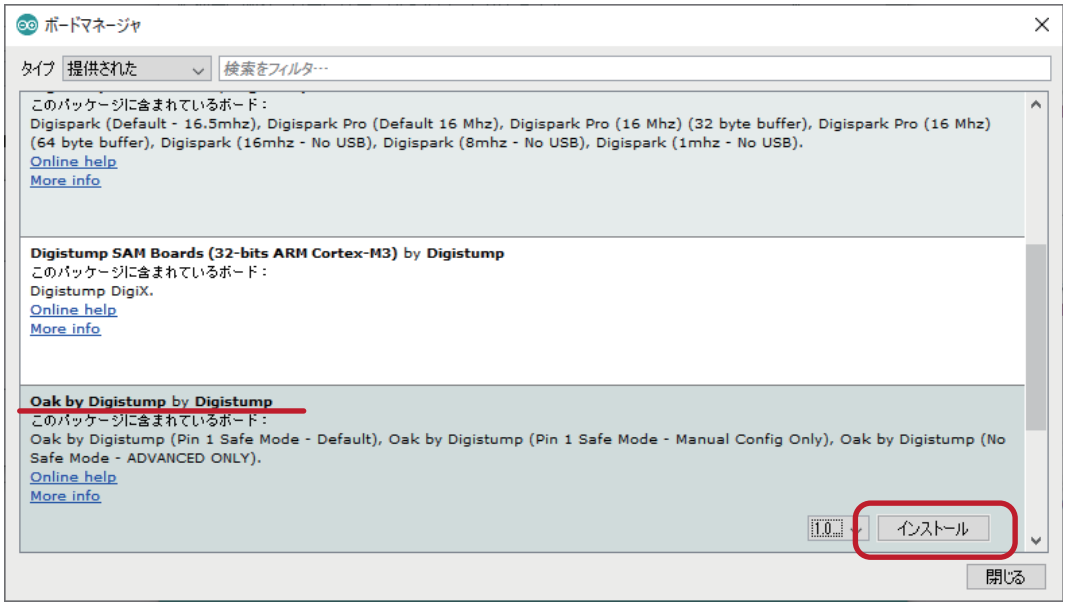


図 4: ボードマネージャ



手順その 3：エモグラスを認識させる

PC とエモグラスを USB で接続します。
新しいハードウエアとして認識され、デバイスドライバが用意されます。この操作は一度目の接続時のみ時間がかかるため、あらかじめ接続して認識を済ませる操作になります。
（10 秒程度でエモグラス内蔵のプログラムが動き始めるため、認識されている時間は一瞬です。内蔵プログラムの動作が始まると、USB の認識状態は Unknown Device に変化します）

手順その 4：WS2812B 用ライブラリの入手

フルカラー LED「WS2812B」のコントロール部分を一から作成するのはかなり大変なので、既成のライブラリを読み込んで利用します。（ライブラリが沢山ネット上に存在しているのも Arduino の特色です）

1. スケッチメニューから ライブラリをインクルード → ライブラリを管理... を選択してください。
2. 「ライブラリマネージャ」ウインドウが開きますので、右側の「検索をフィルタ…」に `neopixel` と入力します。
   リストの中の Adafruit NeoPixel by Adafruit を選択して、インストール をクリックしてください。
   NeoPixel（ネオピクセル）が WS2812B をコントロールするライブラリの名前です。完了後、閉じる で確定させてください。

手順その 5：プログラムを作る

NeoPixel で WS2812B をコントロールする方法について
WS2812B に対して輝度情報を送り込む場合は、表示素子全てのデータを停止する事なく連続して送らなければなりません。転送中の中断は許されません。
そこで NeoPixel のライブラリでは、WS2812B と 1:1 に対応する輝度情報保持メモリ（変数）に全ての情報を設定し、揃ったところで一括して転送します。
転送を指示するまでは、変数に書き込んだ輝度情報は WS2812B には反映されません。このため、表示データを作る作業以外に入れ物を準備する必要があります。

- 一つ目：表示データを作る前に入れ物を作る

```
Adafruit_NeoPixel strip = Adafruit_NeoPixel(24, 0, NEO_GRB + NEO_KHZ800);
```

  を記述します。以後 WS2812B には strip の名前でアクセスします（興味のある方や追求されたい方は C++ の解説書をご覧ください）
  最初の 24 は WS2812B の個数（エモグラス単体では 24 個）、次の 0 は WS2812B が繋がっている IO ポートの番号で、エモグラスの場合は必ずこのままにしてください。
  最後の `NEO_GRB + NEO_KHZ800` は色の並び順と動作速度設定でこのまま入れてください。
  注意 LED の個数に 100 などを指定しても、プログラム書き込み時にはメモリ不足のエラーは出ません。実行する段になってメモリ不足のエラーが発生しますが、PC とちがってエモグラスではエラー発生を外部に伝える手段がありません。外部から見た場合、単に動作しないだけとなります。

- 二つ目：使用準備を行います

```
void setup() {
  strip.begin();
  strip.show();
}
```

関数 setup() 内に begin と show を記述します。setup は Arduino 環境で作られたプログラムが実行開始した際に一度だけ呼び出される関数です。一度だけ行えば十分な初期化等の作業を行う命令を入れるところです。
`strip.begin();` で WS2812B が接続されているポートの初期化を行います。
`strip.show();` で初期の値（全ての輝度が 0）を転送して WS2812B を消灯します。
電源を加えた状態の WS2812B は偶然の値に従って発光します。このため、一度全て 0 にした輝度を転送して消灯しないと、見苦しい状態になる場合があります。

- 三つ目：プログラムで表示データを作り、入れ物（変数）に書き込み（仮置き）します
  各 WS2812B を光らせるには、色別の変数に、輝度レベルを書き込みます。輝度レベルは 0〜255 です。
  輝度レベルを書き込む命令は setPixelColor を使います。WS2812B を表す「strip」にくっつけて、例えば

```
strip.setPixelColor(4,0,0,255);
strip.setPixelColor(5,0,255,0);
strip.setPixelColor(6,255,0,0);
```

  各先頭の数値、4,5,6 は先頭から数えた WS2812B に付けられた変数の番号です。24 個で宣言したため、0〜23 が使えます。
  上記の例では 4（5 番目）,5（6 番目）,6（7 番目）に輝度情報を書きこんでいます。それぞれ、4 に対し青を 255 のレベルで、5 に対し緑を 255 のレベル、6 に対し赤を 255 のレベルでそれぞれ光らせる指示となります。
  strip.setPixelColor に設定する数値は（WS2812 の番号，赤の輝度，緑の輝度，青の輝度）の順です。

- 四つ目：WS2812B に転送
  最後に

```
strip.show();
```

  を記述する事で、変数にセットした輝度情報が一括で WS2812B に転送されます。
  `strip.setPixelColor(6,30,30,0);` とすると赤と緑で理論上は黄色に発光しますが、色味は調整を行う必要があります。また発光強度が異なる場合、`strip.setPixelColor(6,60,60,0);` とすると混合比は同じで、先の二倍の輝度で発光しますが、若干の色変化があります。

一からプログラムを作る場合は最低限以上の記述を行う必要があります。以上の方法を踏まえて、全体を通してプログラムの作成を行い、エモグラス基板への書き込みを行います。
【L チカプログラムの作成】
L チカとは、基本動作が正常に実行されるかを確かめる最低限のプログラムで、LED を点滅させるので L チカと呼ばれます。

1. Arduino IDE を起動します。
2. ツールメニューから、 ボード：xxxx （xxxx は以前使っていた基板の種類名）を選び、候補の中から Digispark (Default - 16.5mhz) を選びます。
3. ツールメニューで シリアルポート：xxxx （初回時はシリアルポートとだけ表記されています）を選択します。項目は何を選択してもかまいません。シリアルポート候補が一つも表示されていない場合は、注釈＊ 3 を参照してください。
4. ファイルメニューから スケッチ例 → 01.Basics → Blink を選択します。
   Blink 名のプログラムの雛形が作成されます。

5. スケッチメニューから ライブラリをインクルード を選び Adafruit_NeoPixel を選びます。
   この操作では先頭に #include <Adafruit_NeoPixel.h> が挿入されるだけです。

6. NeoPixel の実体を作ります。
   ソースコードの編集領域、setup() の少し上に挿入します。

```
Adafruit_NeoPixel strip = Adafruit_NeoPixel(24, 0, NEO_GRB + NEO_KHZ800);
```

7. setup() の内容を変更します。元のプログラムの setup 関数内

```
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
```

   は不要なので消去し、代わりに NeoPixel の初期化を入れます。

```
  strip.begin();
  strip.show();
```

8. 実行部分を書き換えます。
   元のプログラムにある

```
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
```

   は L チカの実行部分で接続した LED を点灯したり消灯したりする処理です。今回は右のメガネと左のメガネの各 12 個の WS2812B を交互に点滅させます。
   上記の LED 点滅を WS2812B 用にします。試験用ですので白色固定（RGB 全て同じ輝度にする）としますが 255 を入れるとまぶしいので約 1/8 の輝度の 30 を入れます。元プログラムの LED 点灯処理は、

```
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
```

   の代わりに

```
  for(int i=0;i<12;i++) strip.setPixelColor(i,30,30,30);
  for(int i=12;i<24;i++) strip.setPixelColor(i,0,0,0);
  strip.show();
```

   に置き換えます。
   次に、LED 消灯処理は、

```
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
```

   の代わりに

```
  for(int i=0;i<12;i++) strip.setPixelColor(i,0,0,0);
  for(int i=12;i<24;i++) strip.setPixelColor(i,30,30,30);
  strip.show();
```

   に置き換えます。
   最後は、点滅スピードの指定を変更します。

```
  delay(1000); // wait for a second
```

   の代わりに

```
  delay(300); // wait for 0.3 second
```

に置き換えます。2 箇所あるので両方とも変更します。delay(1000); は、ここで 1000 ミリ秒（1 秒）待機するという意味です。1000 を 300 に書き換えたので、300 ミリ秒（0.3 秒）の待機に変わります。これによって、左右の点灯・消灯のスピードが変更できます。

最終的にはこのようになります。（3〜25 行目のコメントは修正前の L チカについての内容のため、エモグラス用に変更した後の実情と合いません。）

```
1  #include <Adafruit_NeoPixel.h>
2
3  /*
4    Blink
5
6    Turns an LED on for one second, then off for one second, repeatedly.
7
8    Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
9    it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
10   the correct LED pin independent of which board is used.
11   If you want to know what pin the on-board LED is connected to on your Arduino
12   model, check the Technical Specs of your board at:
13   https://www.arduino.cc/en/Main/Products
14
15   modified 8 May 2014
16   by Scott Fitzgerald
17   modified 2 Sep 2016
18   by Arturo Guadalupi
19   modified 8 Sep 2016
20   by Colby Newman
21
22   This example code is in the public domain.
23
24   http://www.arduino.cc/en/Tutorial/Blink
25 */
26
27 Adafruit_NeoPixel strip = Adafruit_NeoPixel(24, 0, NEO_GRB + NEO_KHZ800);
28
29 // the setup function runs once when you press reset or power the board
30 void setup() {
31   strip.begin();
32   strip.show();
33 }
34
35 // the loop function runs over and over again forever
36 void loop() {
37   for(int i=0;i<12;i++) strip.setPixelColor(i,30,30,30);
38   for(int i=12;i<24;i++) strip.setPixelColor(i,0,0,0);
39   strip.show();
40   delay(300); // wait for 0.3 second
41   for(int i=0;i<12;i++) strip.setPixelColor(i,0,0,0);
42   for(int i=12;i<24;i++) strip.setPixelColor(i,30,30,30);
43   strip.show();
44   delay(300); // wait for 0.3 second
45 }
```

「ファイル」メニューから「名前を付けて保存」を選んで適当な名前で保存します。

手順その 6：プログラムの検証

Arduino IDE ウインドウ上部のスピードボタンの左側、✓ マークの付いたアイコン（検証）をクリックすると、コンパイル（検証）が行われます。

エラーが表示されなければ成功です。

手順その 7：プログラムを書き込む

上のスピードボタンの二番目、➜ マークの付いたアイコン（マイコンボードに書き込む）をクリックすると、コンパイルと転送（書き込み）が行われます。
画面下部の出力エリアに、メッセージ

```
Running Digispark Uploader...
Plug in device now...(will timeout in 60 second)
```

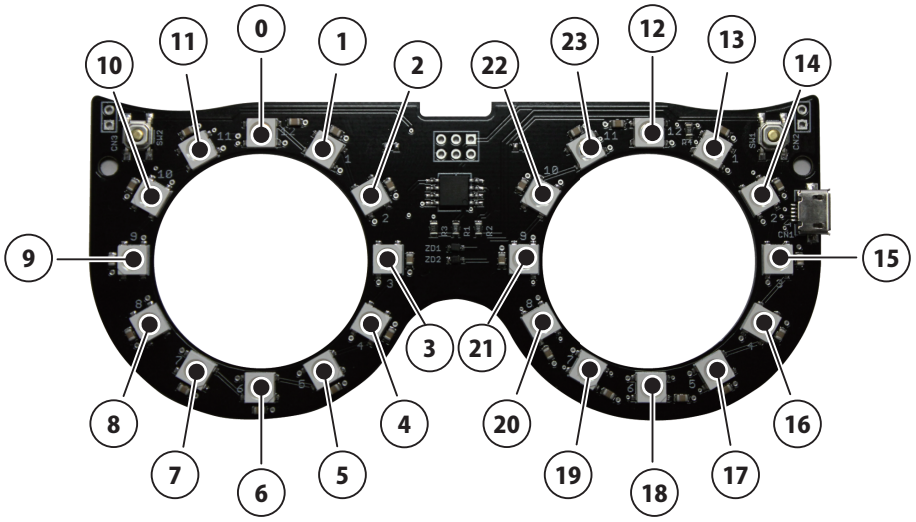
がコンパイル完了時に表示されます。このメッセージが表示されたら 60 秒以内に PC にエモグラスを接続してください（既に接続していた場合は一旦取り外して再接続します）。接続が認識されるとプログラムの書き込みが行われます。
メッセージ欄に下記の出力が現れれば、書き込み成功です。

```
>> Micronucleus done. Thank you!
```

■ WS2812B の接続形態

エモグラス基板上で ATtiny85 に接続された WS2812B は、図 5 の順番に数珠繋ぎになっています。
送り込んだデータはこの順番に順次伝送されます。

図 5: WS2812B の接続順

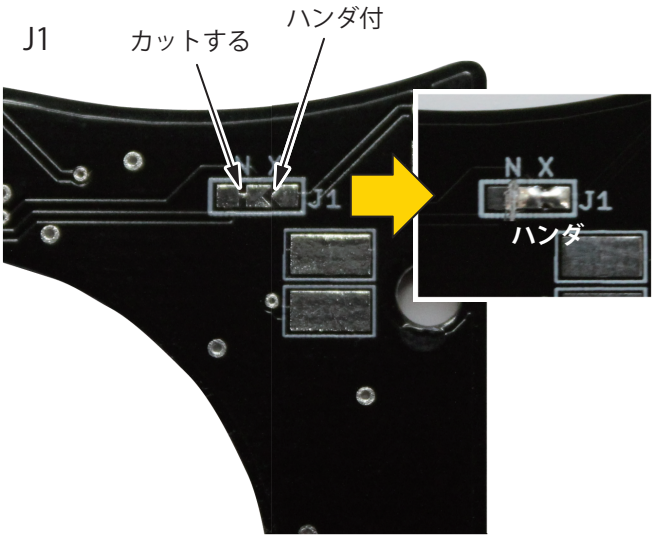


数字はATTINY85（またはデータ入力端子CN3）から見た接続順
ATTINY85／データ入力端子CN3 → **0** → **1** → **2** → … → **23** → データ出力端子CN2

■ 特殊な動作

WS2812B の最終番号のデータ出力は向かって右のコネクタ（CN2）に接続されています。
一方 WS2812B の一番目のデータ入力はエモグラス上の CPU、ATtiny85 の IO ポートに繋がっています。基板裏面にあるジャンパーを切り替える事により、ATtiny85 と繋がっていた WS2812B を切り離し、左のコネクタ（CN3）に接続する事ができます。この機能により、エモグラスの CPU を使用せず、WS2812B が 24 個実装された表示のみに特化した基板として利用できます。
切り替えは基板を裏返しにし、右上にあるジャンパー（J1）を切り替えます。出荷時は図 6 のように中央のパッドと左のパッドが接続されています。この間をカッターナイフ等で切断し、中央と右のパッド間にハンダを盛って接続します。

図 6: 信号切り替え改造

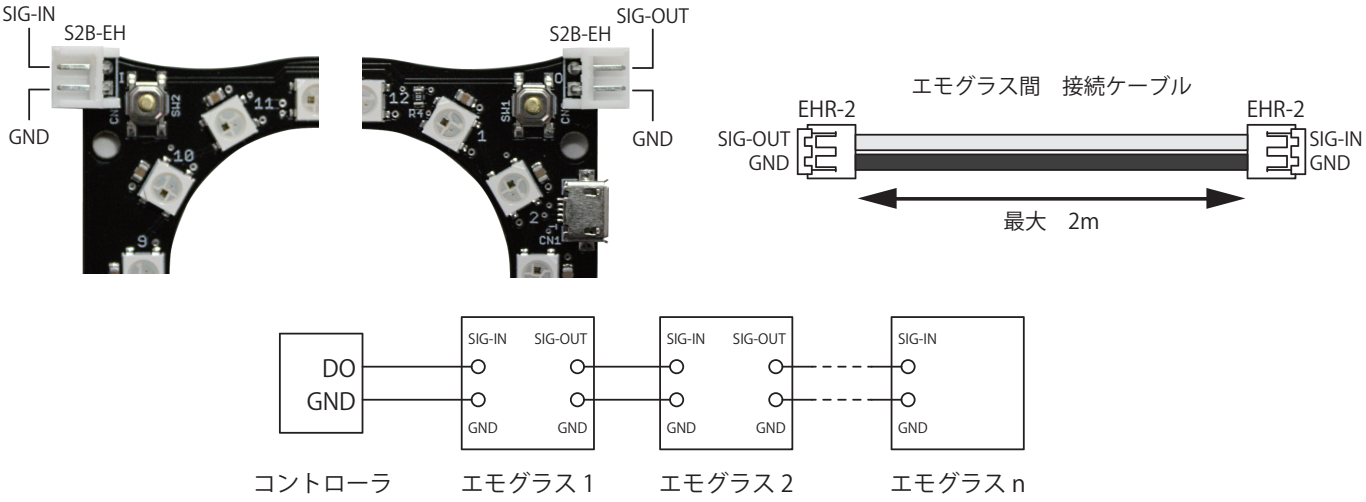


この切り替えによって、WS2812B は CPU から切断され、向かって左の CN3 コネクタ（I と印刷されています）→ WS2812B 24 個 → 向かって右の CN2 コネクタ（O と印刷されています）の順に接続されます。これを利用して、個別のエモグラスを数珠繋ぎに接続する事ができます。端子にコネクタは実装されていませんので、別途ご購入の上ハンダ付けをお願いします。また、エモグラス間の接続ケーブルの長さは 2m 以内としてください。

- コネクタ：S2B-EH
- 勘合するハウジング：EHR-2
- コンタクトピン：BEH-001T-P0.6

信号名及び接続は図 7 を参照してください。

図 7: 外部信号入力の接続



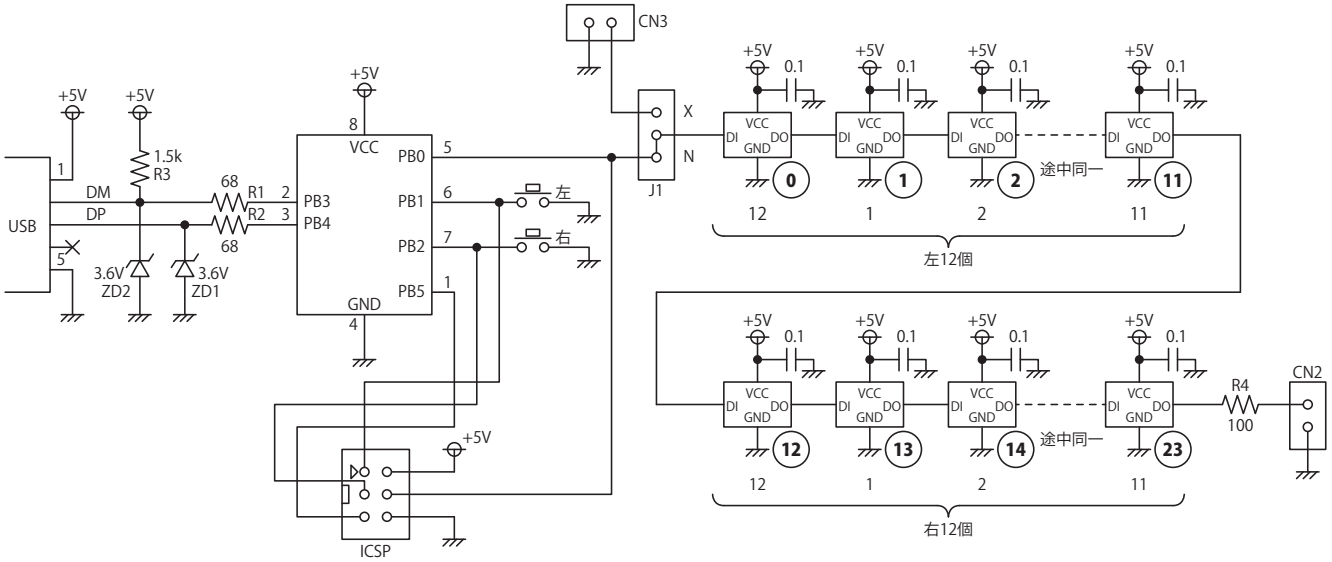
この方法によって、一枚のエモグラスで、数珠繋ぎにした別のエモグラスの WS2812B をコントロールできます。（一枚目のエモグラスが全 48 個の WS2812B を制御する）
WS2812B を制御するためには 1 個に付き 3byte のメモリを必要とします。48 個の WS2812B をコントロールするためには 144byte のメモリを使います。
エモグラスに搭載されている ATtiny85 は 256byte のメモリを持つため、プログラムの実行用メモリと 48 個分の WS2812B 用メモリ 144byte でほぼ使い切り状態になります。更に多くの WS2812B を制御（多数のエモグラスを数珠繋ぎ）したい場合はエモグラス搭載の ATtiny85 ではなく、外部に Arduino UNO 等の多くのメモリが搭載された基板を設置してください。
（Arudino UNO には 2Kbyte の RAM が実装されていますので、作るプログラム規模にもよりますが 20 枚程度のエモグラスを制御できると思われます）

■ Arduino としてのエモグラス

エモグラスに搭載された ATtiny85 のピン構成は、以下の通りになっています。

GPIO	ATtiny85	アナログ	その他	エモグラス上の接続先
P0	5 (PB0)	PWM	I^2C SDA	WS2812B コントロール用データ出力
P1	6 (PB1)	PWM		向かって右の押しボタンスイッチ
P2	7 (PB2)	A/D (A1)	I^2C SCL	向かって左の押しボタンスイッチ
P3	2 (PB3)	PWM, A/D (A3)	USB D-	USB D-
P4	3 (PB4)	PWM, A/D (A2)	USB D+	USB D+
P5	1 (PB5)	A/D (A0)	-	-

■ 回路図



・本製品およびそれらを構成するパーツ類は、改良・性能向上のため予告なく仕様・外観等を変更する場合があることをあらかじめご了承ください。
・本製品は組立キットまたは半完成品です。製作作業中の安全確保のため説明書をよくお読みになり、正しい工具の使用・手順を守ってください。
・完成品でない商品の性格上、組み立て後の完璧な性能・品質・安全運用等の保証はできません。完成後はお客様 (組立作業者) ご自身の責任のもとでご使用ください。
・本製品は機器への組込み他、工業製品としての使用を想定した設計は行っていません。また、本製品に起因する直接、間接の損害につきましては当社修理サポートの規定範囲を超えての 補償には応じられません。

Electronic Devices, Parts, Kits & Robots
KYOHRITSU
共立電子産業株式会社 共立プロダクツ事業所
〒556-0004 大阪市浪速区日本橋西2-5-1
TEL:06-6644-4447 FAX:06-6644-4448

【“共立プロダクツ”ブランドとは】
当ブランドの製品はユーザーニーズを捉えた製品をリーズナブルな価格でのご提供を目指しています。
そのためユーザーサポートはメールに限定しておりますことをご理解、ご了承ください。

Email:wonderkit@keic.jp

Twitterやblogで応用例や製品紹介を更新中です。ぜひご覧になってください。 共立プロダクツ 検索