

図2

します。

1. LEDのアース端子(短い方の脚)をワイヤーに接続します。
2. このワイヤーの端を、Raspberry Piの拡張ヘッダーのピン6(Ground)に接続します。Raspberry Piのピン構成図を参考にしてください。
3. LEDの長い方の脚を抵抗に接続します。
4. この抵抗の反対側の端を2本目のワイヤーに接続します。
5. 2本目のワイヤーの反対側の端を、Raspberry Pi 拡張ヘッダーのGPIO 1(ピン12)に接続します。

組立て後の全体の様子は図3のようになります。

注意点として、正しいピン番号体系を使用するようにしてください。現状ではさまざまなピン番号体系が利用されています。BlueJはPi4Jライブラリを使用するため、ここではPi4Jライブラリで用いられる番号体系に合わせます。かならず正しい番号体系を参照するようにしてください(先ほどの図を確認してください)。この番号体系で

は、GPIO 1はピン12です。

**BlueJでの作業**

現時点でLEDはGPIOピンのひとつに接続されています。次に、BlueJでGPOutputクラスのオブジェクトを作成します(パート1で説明したとおり、BlueJではクラスを右クリックして、メニューでコンストラクタを選択することによって、オブジェクトを作成できます)。

表示されるコンストラクタには、使用するGPIO番号を選択できるものと、デフォルトのGPIO 1を使用するものの2種類あります。前項でLEDをGPIO 1に接続したので、ここではデフォルトのコンストラクタを使用できます。この操作により、オブジェクト・ベンチに、LEDを表すオブジェクトが表示されます(通常は、このオブジェクトはGPIO 1に接続されているあらゆる部品を表しますが、本記事ではLEDがその部品にあたります)。

以上で、オブジェクトとの対話によってLEDを制御できるようになりました。本当に簡単です。gPOutput



図3

オブジェクトを右クリックしてon()メソッドを呼び出す(図4)ことによって、LEDが点灯します。次に、LEDの消灯を試してください。さらに、blink()メソッドも試すことができます。さまざまなパラメータ値を指定してこのメソッドを呼び出してみてください。

**ボタンの使用**

次に、Raspberry Piにボタンを接続し、Raspberry Piの状態を読み取ります。ボタンを以下のように接続してください(図5)。

1. 3本目のワイヤーを使用して、ボタンの一方のコネクタを、Raspberry Pi 拡張ヘッダーの3.3V

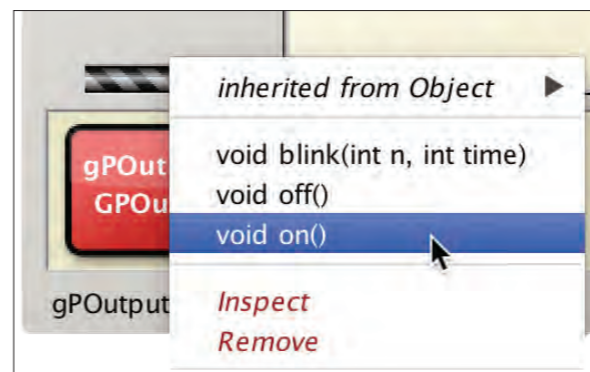


図4

2. 4本目のワイヤーを使用して、ボタンのもう一方のコネクタを、Raspberry Pi 拡張ヘッダーのGPIO 4(ピン16)に接続します。

BlueJで、Button型のオブジェクトを作成して、このボタンを表すことができますので、実際に試してください。先ほどと同様に、デフォルトのコンストラクタと、使用するGPIO番号を指定するためのパラメータを持つ第2コンストラクタがあります。デフォルトで使用されるGPIO(この場合GPIO 4)にボタンを接続したので、デフォルトのコンストラクタを使用すれば簡単にできます。

buttonオブジェクトには、ボタンの状態を確認するための2つのメソッド、isUp()とisDown()があります。ボタンに触らずに、つまりボタンが押されていない状態でこれらのメソッドを呼び出した場合、isUp()はtrueを返し、isDown()はfalseを返します。

ボタンを押したままの状態でも、これらのメソッドを呼び出してみてください。buttonオブジェクトによって、ボタンの状態が正しく報告されるはずですが、

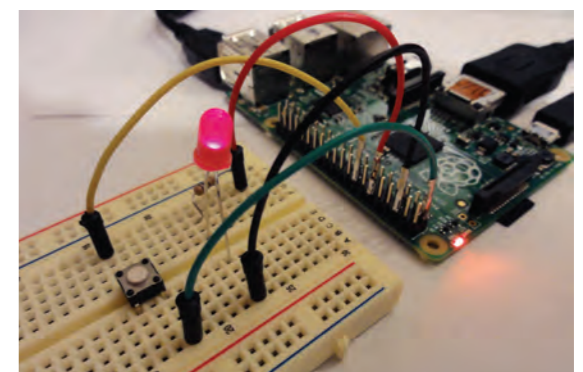


図5





