「地域におけるIoTの学び推進事業」実証事業

デジタル寺子屋@つづき

講座テキスト

つづきIoT学習推進協議会

「デジタル寺子屋＠つづき」って？何をするの？

デジタル寺子屋＠つづきは、デジタル技術革新がもたらす大きな社会変化に対応し、誰もがクリエイティブに生きるために必要な知識を学ぶ、地域のICTクラブです。2018年総務省IoTの学び推進事業の実証実験のひとつとして、横浜市都筑区のNPO法人 I Loveつづきと、デジタル学習支援に実績のあるNPO法人ブロードバンドスクール協会、日本マイクロソフト 株式会社などが、地域のみなさまと連携して、つづきIoT学習推進協議会を発足し、実施、運営にあたりました。

デジタル寺子屋＠つづきでは、10 歳から 13 歳までの児童を対象に、教育用に開発された “IchigoJam” と “micro:bit” を使ってプログラミングの初歩を学びました。
なお、このテキストの著作権は総務省に帰属します。出版や販売等の商用利用、内容の改変等は出来ませんので、ご了承ください。

IoT の学び推進事業

総務省
MIC Ministry of Internal Affairs and Communications

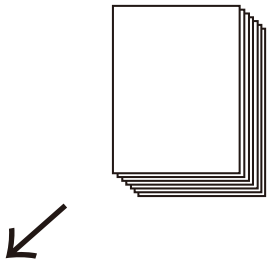
この事業は、総務省の「地域における IoT の学び推進事業」実証事業のひとつとして、つづき IoT 学習推進協議会が実施・運営しています。

つづき IoT 学習推進協議会

特定非営利活動法人 I Love つづき／ NPO 法人 ブロードバンドスクール協会 / 日本マイクロソフト株式会社 / PCN 秋葉原 / ONE / Arioso 学習教室 / イクミンズ / 一般社団法人 横浜もの・まち・ひとづくり / NPO 法人 テレワークセンター横浜

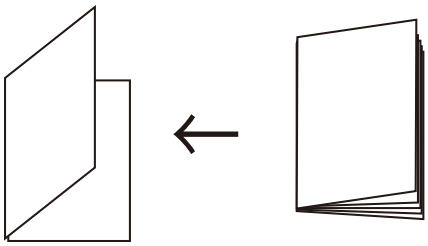
このテキストの使い方

プログラミング講座を運営するときのポイントをわかりやすくまとめたテキストと、人生の先輩からのメッセージを小冊子を1つのフォルダーに入れました。対象となるこどもたちの進捗状況に合せてご利用ください。



「IchigoJam でプログラミング」

プログラミング言語 IchigoJam を使っての講座用テキスト。カード形式なので、単元毎、テーマ毎に取り出してコピー編集利用ができます。アイロンビーズ作品でのプログラミング例も載っています。



「みんなで仲良くプログラミング」

教育用に開発されたマイコンボードの、micro:bit を様々に活用してプログラミングを学ぶテキスト。

「何かをつくるって楽しいね」
＋
「みんなで楽しくプログラミング」

「コンピューターおばあちゃん」と呼ばれる若宮 正子さんからAI(人工知能)時代に生きるあなたへのメッセージと、IchigoDake & IchigoIgai スクールセットの使い方と電子工作のテキスト。アイロンビーズでつくる花の説明もあります。

デジタル寺子屋 @ つづき　目次

「IchigoJam でプログラミング」

「みんなで仲良くプログラミング」

「何かをつくるって楽しいね」
＋
「みんなで楽しくプログラミング」

■プログラミングとは？

数字と文字でコンピューターにしてほしいことを伝えることです。

■ IchigoJam とは？

すべてのこどもにプログラミングのきっかけを提供するべく誕生した、1500 円で買えるプログラミング専用こどもパソコン。
IchigoJamBASIC というプログラミング言語でコンピューターにしてほしいことを伝えます。
福井県鯖江市で誕生。
IchigoJam(イチゴジャム) のほかにも、MapleSyrup(メープルシロップ)、PanCake(パンケーキ)、MixJuice(ミックスジュース) なんておいしそうな関連製品がそろっています。

関連製品各種

・MapleSyrup：モーター制御基板
・PanCake：グラフィック・サウンド制御基板
・MixJuce：ネットワーク制御基板
など

リファレンス（コード一覧と説明書）
https://ichigojam.net/IchigoJam.html

※ IchigoJam は jig.jp <http://jig.jp> の登録商標です。

point

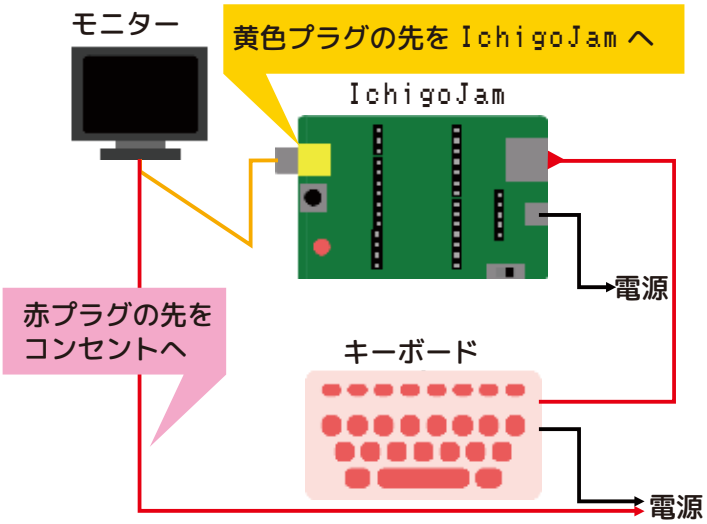
・プログラミングをやる人がみんなすべてのコードを覚えているわけではないです。

・リファレンスを手元に置いてコードを書くこともありますし、誰かほかの人が同じことやってないかなーと調べたり人に聞いたりするのは日常です。
→例えば「IchigoJam　複数の LED をつなぐ」とか「IchigoJam　LED　抵抗の計算」など気になったことをインターネットで調べると今は結構いろいろ出てきます。

・まずは「誰か」「何か」のマネをすることがスタートで問題ないです。

・英語なんかではよくある、なんとなく「聞ける」「読める」けど「書けない」「話せない」で基本的なプログラミングは何とかなります。

■プログラミングに必要なものをつないでみる



IchigoJam：U・T・S など各種バージョンあり
キーボード：「PS/ 2 対応」の USB 接続のもの。
モニター：「黄色のビデオ入力端子」がついているもの。
ケーブル①：「黄色のビデオ入力ケーブル」赤や白でも可。
ケーブル②：「microUSB ケーブル」スマホ充電用（Type C は不可）
AC アダプタ①：「モニター」を家庭用コンセントにつなぐためのアダプタ。
AC アダプタ②：「microUSB ケーブル」を家庭用コンセントにつなぐためのアダプタ。

■正常に動くか確認

電源を ON にして、モニターに「IchigoJam　BASIC　～　OK」と出れば接続完了

```
IchigoJam BASIC 1.x.x by jig.jp
OK
```

point

・キーボードは使えるキーボードが限られるので、購入する際は注意が必要。PS/ 2 対応で USB 接続が可能であるもの。また、IchigoJam を購入する際に、「英語キーボード用」か「日本語キーボード用」かを確認の上、それに合わせてキーボードを購入しましょう。

・ビデオ入力端子があればテレビも使えます。ただし、たまに「相性の悪いもの」がある。ポータブル DVD プレーヤーなどでも機種によっては使えます。モニターの種類によっては、AC アダプタが別売りだったり、ビデオ端子の「オス（とんがってるタイプ）」が標準でついていたりすることもある。

・microUSB ケーブルは 100 円ショップのもので OK。ただし、こちらもたまに「相性の悪いもの」がある。

・AC アダプタは一部 100 円ショップで 200 ～ 300 円程度。

■基本のプログラム

```
LED 1 ↲
```

```
LED 0 ↲
```

LED を光らせて基本のプログラムをまなびます。

「↲」は「Enter」(エンター) キーを押すことを意味している。

LED　だけだと「点灯」する命令か「消灯」する命令かがわからない。

「1」で出力、「0」で出力停止。

■ LED を点滅させてみる

```
LED 1:LED 0 ↲
```

「:」(コロン) をつかって、命令をつなげることができる。

←一瞬ついた？？？

```
LED 1:WAIT 60:LED 0 ↲
```

いろいろなプログラムで重要になるのが、「WAIT」ウェイト（待て）「60」でおよそ「1 秒」

「↑」「↓」でカーソル移動し、書いたプログラムに戻って再度「エンター」を押すと、再実行できる。

■別の書き方で LED を点滅させてみる

```
10 LED 1 ↲
20 WAIT 60 ↲
30 LED 0 ↲
```

行番号を使ったプログラムでは

- 「一時的」に Ichigojam にプログラムを記憶させることができる。
- 「エンター」を押しても実行されない。
- ここでは「エンター」は「決定」の意味を持つ。

```
RUN ↲
```

RUN(ラン) と入力し、「エンター」を押すと実行する。

←「F5」ボタンでも代用できる

■2 回点滅させてみる

```
10 LED 1 ↵
20 WAIT 60 ↵
30 LED 0 ↵
40 WAIT 60 ↵
50 LED 1 ↵
60 WAIT 60 ↵
70 LED 0 ↵
```

行番号は「2 桁以上」の数字が使われることが多い。プログラムの追加や改造のとき、行番号が詰まっていると修正がたいへんなため。「10」ずつ増やしておくと、間にプログラムを追加できる。

「F5」で実行

■新しいプログラムを書く

```
NEW ↵
```

NEW(ニュー) と入力すると、IchigoJam が一時的に記憶したプログラムを消去できる。

```
10 LED 1:WAIT 60 ↵
20 LED 0:WAIT 60 ↵
30 LED 1:WAIT 60 ↵
40 LED 0 ↵
```

行番号とコロンは組み合わせができる。組み合わせることで、行数を減らしたり、時にはプログラムの実行そのものに変化をつけられる。

point

・プログラムの書き方は 2 種類
1) 1 行で書く　→　「Enter」を押すと「実行」
2)「行番号」を使って複数行で書く　→　「Run」もしくは「F5」ボタンで「実行」

・プログラミングの基本は「コンピュータが理解できるように書く」こと
1)「LED をつけて、消して」だけだと、「コンピュータ」のスピードで実行するだけ
→「LED をつけて、○○秒待って、LED を消して」のように「WAIT」を使い「時間」も指定して命令すると狙い通りにうごいてくれる
2)「行番号」は「10」ずつ増やしてつけるのが一般的
→「10」ずつ増やしておくと、間にプログラムを追加できる
3) 行番号を使って新しいプログラムを書くときは必ず「NEW」を実行
→「NEW」を実行しておかないと、不要なプログラムが残ってしまうこともある
4) 画面がいっぱいになってきたら「F1」ボタンで画面をキレイに
→行番号を使ったプログラムは IchigoJam がすこしだけ覚えているから、「F4」ボタンで復活できる

■プログラムを「保存」する

```
SAVE 0 ↵
```

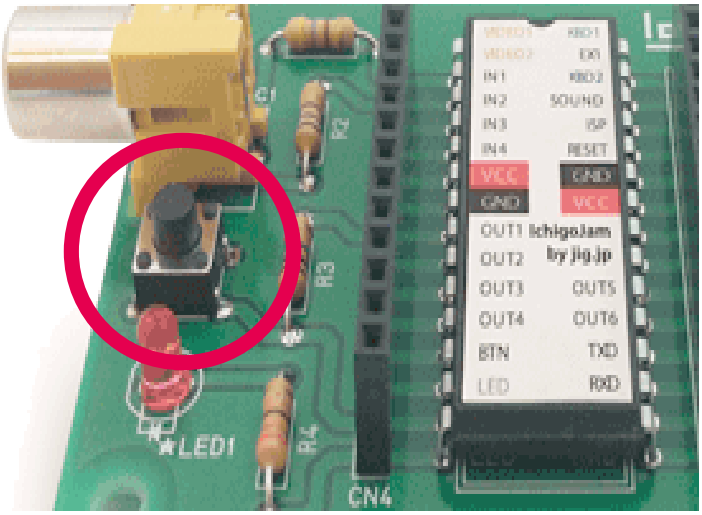
```
Saved ~byte ↵
OK ↵
```

SAVE(セーブ) コマンドで IchigoJam にプログラムを「保存」する。

IchigoJam の「保存領域」は「0」～「3」番までの 4 領域。

Saved ~byte と容量と OK が出れば成功
「0」だけは特別な意味を持つ。

■保存できているか確認する



「0」番に保存したら、電源 OFF →ケーブル類（ビデオケーブル・USB）を抜く。microUSB ケーブルは「給電」のためそのままにしておく。

黄色いビデオ端子横の黒い「ボタン」を押しながら再度電源を ON してみる。

SAVE0 に保存されたプログラムは、電源 ON したときに「自動起動」される(キーボードやテレビをはずしても OK)
電源をいれてLED が 2 回点滅したら成功

point

・行番号を使ったプログラムは、NEW を実行したり、電源を切るまでは覚えてくれる。
1) 電源を切っても覚えてもらう方法として「SAVE」（保存）を使う。
→ SAVE は「0 ～ 3 番」、4 つ保存できる。
2)「0 番」は特別
→ IchigoJam のボタンを押しながら電源を入れると、「0 番」のプログラムが自動実行される。
→電源を入れただけで実行されると、「実行したくないとき」にも実行されてしまうので「ボタン」を押しながら電源を入れることで「実行したいよ」と IchigoJam におしえてあげる。
・黒いボタンは「タクトスイッチ」と呼ばれます。対して電源スイッチのような形状のものを「スライドスイッチ」と呼びます。「ボタン」と「スイッチ」で区別しよう。

■保存したプログラムを「呼び出す」

ケーブル類（ビデオケーブル・USB）をもういちどつないで、電源を ON する。

```
LOAD 0 ↲
```

LOAD（ロード）と入力
この状態ではまだ、「ほんとうに呼び出されているか」わからない。

```
Loaded ~byte ↲
OK ↲
```

「Loaded ～ byte」とでれば成功。

```
LIST ↲
```

LIST で「一時記憶」を呼び出すことができる。「F4」でも代用できる。

■くりかえし点滅させてみる

```
10 LED 1:WAIT 60 ↵
20 LED 0:WAIT 60 ↵
30 LED 1:WAIT 60 ↵
40 LED 0 ↵
```

もし「くりかえす」というプログラムがあるとしたら・・・

行番号「30」と「40」はいらないことになる。

```
30 ↵
```

「行番号のみ」で「エンター」を押すと、その行がそのまま消去される

```
40 ↵
```

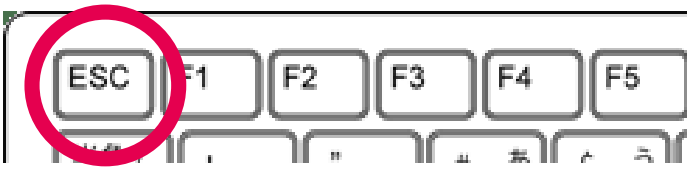
30 行と 40 行が消えたかを「F1」→「F4」でチェック

■ちょっとレベルアップ

```
10 LED 1:WAIT 60 ↵
20 LED 0:WAIT 60 ↵
30 GOTO 10 ↵
```

30 GOTO と入力
「F5」で実行

ずっと点滅をくりかえすので、プログラムを止めるために「ESC」（エスケープ）キーを押す



実行中のプログラムを止めるほうほうは「ESC」か「電源 OFF」

point

・一般的な PC と違い、構造が単純なので「シャットダウン」という操作はなし

・ESC の後、LED がついたままなら「LED0」で消す

■くりかえす回数を指定する

```
10 LED 1:WAIT 60 ↲
20 LED 0:WAIT 60 ↲
```

30 行をいったん消去

```
10 LED 1:WAIT 60 ↲
20 LED 0:WAIT 60 ↲
5 FOR A=1 TO 3 ↲
30 NEXT ↲
```

5 FOR A=1 TO 3
30 行に NEXT と入力

「F1」→「F4」で確認

```
5 FOR A=1 TO 3 ↲
10 LED 1:WAIT 60 ↲
20 LED 0:WAIT 60 ↲
30 NEXT ↲
```

下に続けて小さい行番号入りのプログラムを入力しても OK。行番号順に自動で並び変わる。

「F5」で実行

3 回 LED が点滅したら成功

point

- 「A」のようなアルファベットを「変数」という。A ～ Z までなんでも OK。今回は A を 1 から 1 づつ増やして、3 まで行ったら終わるというプログラム。
- For 変数 = 初期値 To 終了値 NEXT 構文

■画面に指定した数字を表示する

```
NEW ↵
```

一時的に記憶したプログラムを消去する
「F1」→「F4」でプログラムが消去されているか確認。

```
PRINT 3 ↵
```

PRINT~ は「～と表示しろ」という命令
「? 」3 でもおなじ結果になる。

```
3
OK
```

「3」と表示されたら成功

■文字を表示させる

```
PRINT HELLO↵
```

「HELLO」と入力

```
0Syntax error
```

Syntax error(シンタックスエラー) と表示される→まちがいがある

数字と文字は別物。文字は「文字列」と呼ばれる

```
PRINT "HELLO"↵
```

「" " 」(ダブルクォート) で囲むことで「文字列」を表示できる。

```
HELLO
OK
```

FOR ～ NEXT　で出てきた「変数」との区別をするため、アルファベット 1 文字は変数としてつかう場合と文字としてつかう場合でつかい分けが必要。

point

- 工夫をしないと、アルファベット 1 文字（変数）とアルファベット 2 文字以上の区別がつかなくなる。
- 「0Syntax error」と頭に「0」が入っている。これは、頭の「H」を変数として認識したため、「0」を表示した。

■変数と文字のちがい

```
A=5 ↵
```

「変数」とは「箱」のようなもので、数を 1 つだけ格納できる。

A=5 と設定すると、名前が「A」という箱には「5」という値が入っていることになる。

```
PRINT A ↵
```

```
5
OK
```

変数 A を表示しろという命令には、その値 5 が表示される。

```
PRINT "A" ↵
```

```
A
OK
```

文字 A を表示するには「" "」で区別する。

point

・こういった違いが出るので、数字と文字（文字列）をしっかりと区別して使用する必要がある

■変数で計算

「変数」は「計算」にもつかえる。

```
A=5 ↵
```

```
A=A+3 ↵
```

```
?A ↵
```

「?」は「PRINT」のかわり
「PRINT A ↵」と同じ

```
8
OK
```

「等式」としての意味ではなく、元の自分に 3 をたしてして更新という意味。

point

・変数は自分自身をアップデートすることができる。

■変数のいろいろなつかいかた

```
B=RND(5)  ↵
```

RND (ランダム) と入力する。

```
?B  ↵
```

どうなった？

```
0
OK
```

RND は「ランダム」の略。とくに決まりがなく、予測ができないこと。

```
1
OK
```

結果は 0 からカッコのなかの数字までの整数のどれかになる
ただし、カッコなかの数字は含まない
今回は 0・1・2・3・4 のどれかになる

```
2
OK
```

```
3
OK
```

```
4
OK
```

point

・RND はいろんなところで使用される。ゲームプログラミングでは重要なプログラム。

・ランダム　RND(数字) は、() 内の数字の 1 つ前までの整数がランダムで選ばれる。
1) RND(5) の場合　0,1,2,3,4 のうちのどれか

■ちょっとしたゲームプログラミング

```
10 C=RND(2) ↲
20 IF C==0 THEN ?"ZERO" ELSE ?"ICHI"↲
```

```
ZERO
```

「F5」で実行(じっこう)

```
ICHI
```

どうなった？
何度(なんど)も実行(じっこう)して違(ちが)いをしらべよう。

■ IF~THEN~ELSE の意味(いみ)

IF ●● THEN ◆◆ ELSE ▲▲　は
「もし●●だったら◆◆して、そうじゃなかったら▲▲して」
という命令

```
C==0 "ZERO"

C<>0 "ICHI"↲
```

もし C が 0 だったら ZERO と表示(ひょうじ)して、
そうじゃなかったら ICHI と表示(ひょうじ)して
という命令(めいれい)

point

- 「THEN」は省略可能
- 「ELSE」以降も特に命令を必要としなければ省略して OK。
- IF ●● THEN ■■ ELSE ▲▲の基本は日本語で考えるとわかりやすい
  1) もし（IF)、●●が正解だったら、■■して、もしハズレだったら▲▲して
  2) ELSE ～は省略できる
  →省略すると、
  もし（IF)、●●が正解だったら、■■して、もしハズレだったら先に進んで　となる

■ボタンを押すとじゃんけんの「手」が決まるプログラム

```
10 A=RND(3) ↵
20 IF A==0 THEN ?"GU-" ↵
30 IF A==1 THEN ?"CHOKI" ↵
40 IF A==2 THEN ?"PA-" ↵
```

「F5」で実行(じっこう)

どうなった？
何度(なんど)も実行(じっこう)して違(ちが)いをしらべよう。

A==0 のとき

```
GU-
```

A==1 のとき

```
CHOKI
```

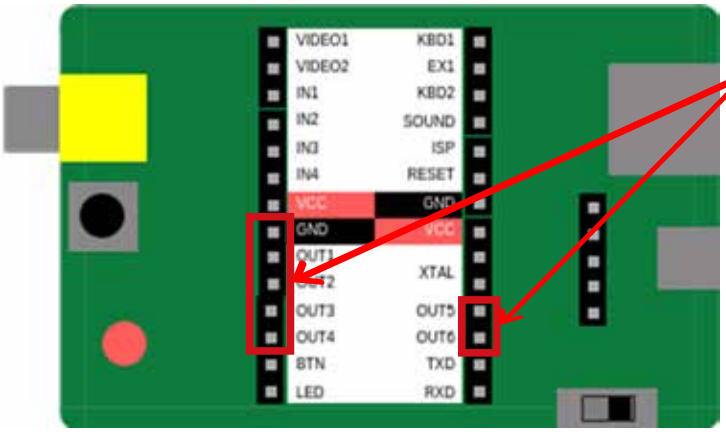
A==2 のとき

```
PA-
```

point

・IF 数１　THEN　次１　の意味は
　→もし数１のとき、次１を実行する

■ LED をふやそう



初めからついているの LED のほかにも LED を増やせる

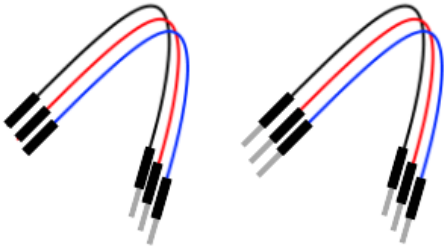
LED を増やす場合には基本的に 7 つの「ポート」をつかう

OUT1 ～ OUT6
→電池のプラス

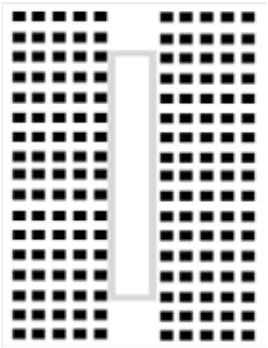
GND
→電池のマイナス

■ LED をふやためにつかうものの例

LED
色によって
とくちょうがちがう



ジャンパーワイヤー
ピンが両方出ているものと
ピンが片方だけ出ているもの



ブレッドボード
穴がいっぱいあいている
簡単に「回路」ができる

抵抗（ていこう）
大きな電流が流れないようにする
線の色の組み合わせで性能が変わる

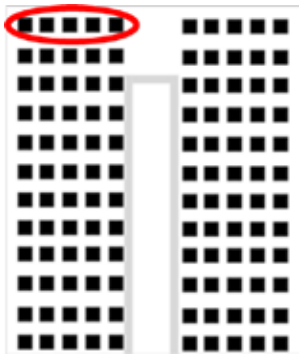
大きな電流が流れると LED が壊れてしまうことがあります。そのため、電流を流れにくくする「抵抗」をつかいます。

LED は色ごとに特徴が違います。本来は特徴に合わせて電流を調整する必要があります。電流を調整するための「抵抗」についてはしらべてみましょう。

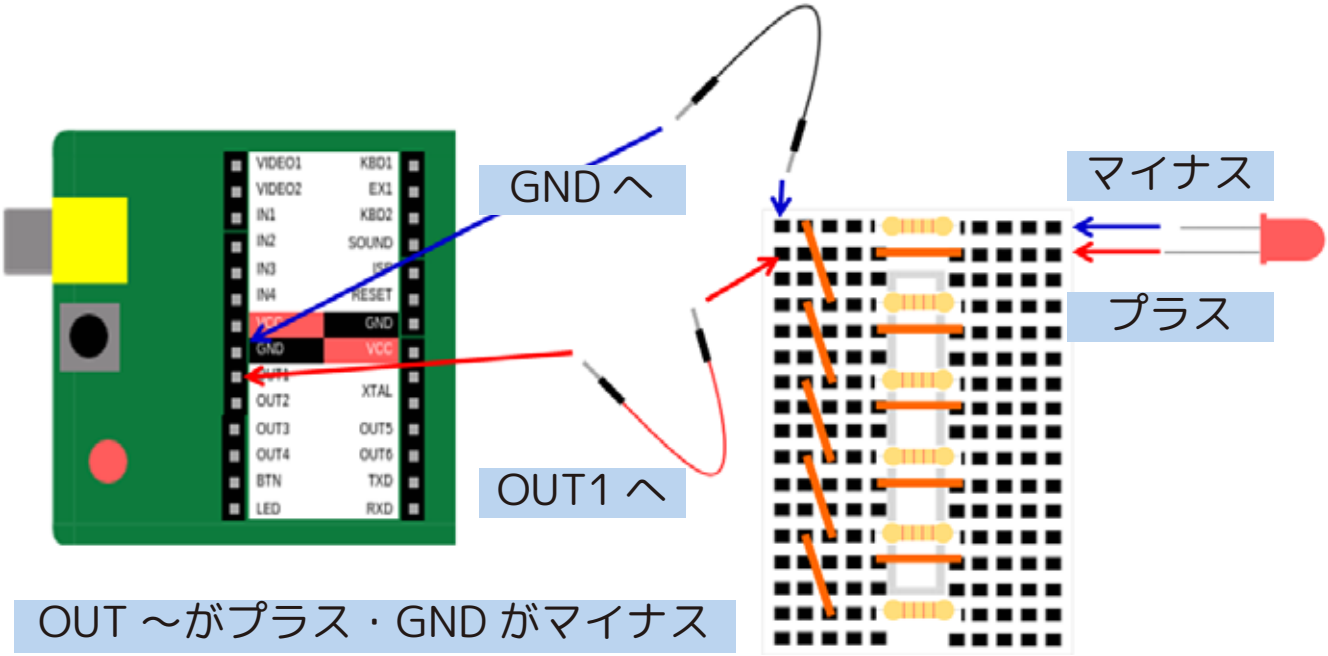
■ LED を「1 つ」つないでみよう

まずは IchigoJam の電源を OFF

LED は脚の長いほうを
「プラス」につなぐ



ブレッドボードは
丸で囲んだ部分が
「つながっている」



■ OUT1 につないだ LED をつけて消す

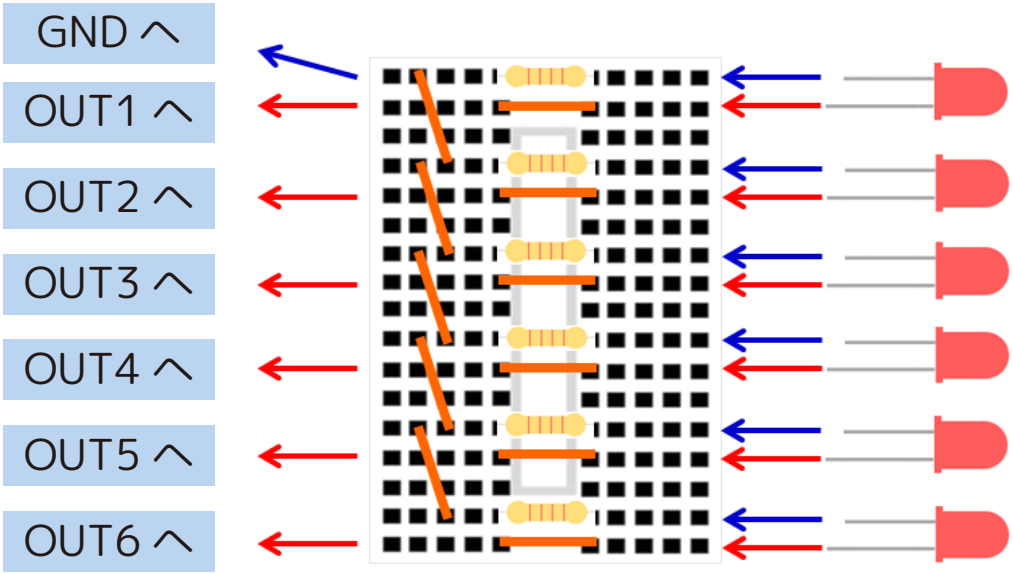
```
OUT 1,1:WAIT 60:OUT 1,0↵
```

point

・これまでとの違いは「LED」が「OUT」に変わったことと、ON・OFF の命令だけではなくなったこと

・今回は「LED つけて」ではなく「OUT1 に電気を流して」という命令に変わったイメージ

■ LED を「6 つ」に！！




```
OUT 1,1:WAIT 60:OUT 1,0↵
```

ここの数字を「1」～「6」まで
順に変えて点滅するか確認しよう

LED は電流の流れる方向が決まっている
ので、もし点滅しない LED があったら、
接続を見直してみよう

■ LED を順番に光らせる

```
10 FOR L=1 TO 6↵
20 OUT L,1:WAIT 60:OUT L,0↵
30 NEXT ↵
```

動いているプログラムを止めると
きは「ESC」ボタン

■ LED をランダムに光らせる

```
10 L=RND(6)+1↵
20 OUT L,1:WAIT 60:OUT L,0↵
30 GOTO 10↵
```

■ LED でルーレットをつくろう

```
NEW ↵
```

```
5 ?"PUSH BTN START" ↵
10 OUT0:IF BTN()=0 GOTO 10 ↵
20 F=0:W=5:T=0:L=RND(6)+1 ↵
30 IF F=0 AND BTN()=0 F=1 ↵
40 IF F=1 AND BTN()=1 F=2:T=RND(15)+15 ↵
50 IF F=2 AND W>120 F=3:GOTO 100 ↵
60 W=W+T:BEEP:OUT L%6+1,1:WAIT W:OUT
L%6+1,0:L=L+1:GOTO 30 ↵

100 FOR X=1 TO 3 ↵
110 BEEP 5,20:OUT L%6+1,0:WAIT 20:OUT
L%6+1,1:WAIT 20 ↵
120 NEXT ↵
130 ?"PUSH BTN RESTART" ↵
140 IF BTN()=0 GOTO 140 ELSE CLS:GOTO 10 ↵
```

「F5」で実行(じっこう)　どうなりました？

```
PUSH BTN START
```

↑黒(くろ)い「ボタン」を押(お)してみよう

BTN は「ボタン」の略(りゃく)

BTN()==0 はボタンが押(お)されていない状態(じょうたい)

10 行目(ぎょうめ)のプログラムは「もしボタンが押(お)されていなければ、ずっと 10 行目(ぎょうめ)をくりえして」という命令(めいれい)

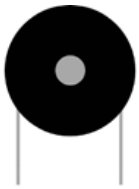
point

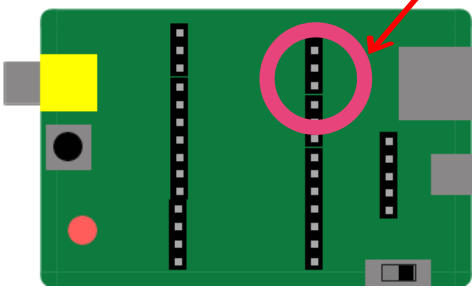
- 「ボタン」は黒い「タクトスイッチ」のこと。このプログラムも様々なところで使用される。
- CLS(クリアスクリーン) は画面を全部消す命令
- 式 AND(アンド) 式はどちらの式も 1 の時に 1、それ以外で 0 を返す
- L%6+1 の %(パーセントあるいはモッド) は割り算した余りを返す

2-4.IchigoJam ミュージック 1

■「Ichigojam」で音を鳴らそう

圧電サウンダ





圧電サウンダを「SND」「GND」と書かれている黒いソケットに差し込む

```
BEEP ↵
```

「ピッ」と音が鳴れば接続 OK

```
BEEP 5,60 ↵
```

音の高さ　　音の長さ

■ドレミも出せる？

```
PLAY "CDEFGAB"↵
```

「ドレミファソラシ」まで

```
PLAY "O4 CDEFGAB O5 C2"↵
```

標準の「4 オクターブ」指定　「5 オクターブ」指定　2 分音符

point

・でも機械の特徴として「ちょっと音痴」です

Ichigojam でピアノプログラミング

```
10 CLV:CLS ↵
20 ?"ﾄﾞ ﾚ ﾐ ﾌｧ ｿ ﾗ ｼ ﾄﾞ" ↵
30 ?"1 2 3 4 5 6 7 8" ↵
40 P="C ;D ;E ;F ;G ;A ;B ;<C"↵
50 K=INKEY():IF !K GOTO 50 ↵
60 PLAY P+(K-49)*3 ↵
70 GOTO 50↵
```

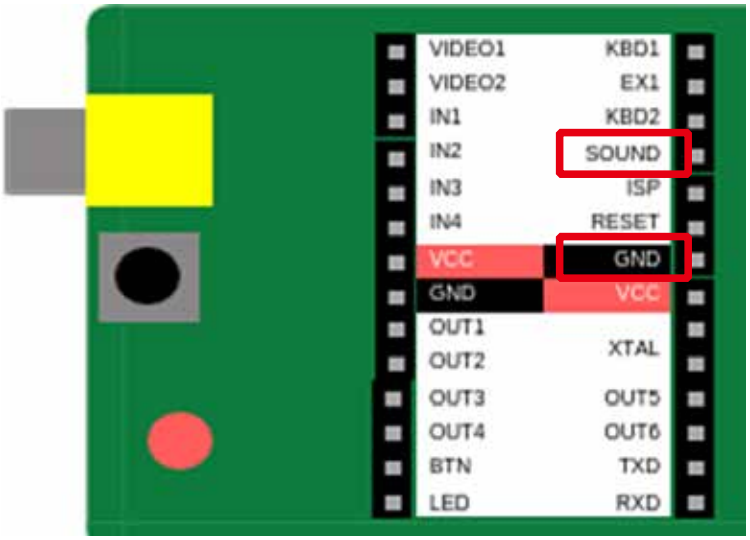
「F5」で実行(じっこう)
数字(すうじ)ボタンで好(す)きな音(おと)をだしてみよう

Ichigojam でおとゲーム

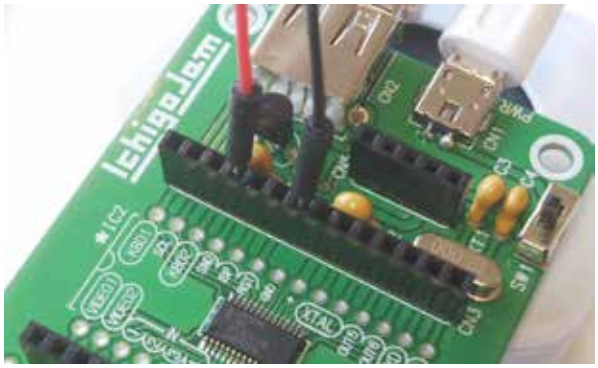
```
10 CLV:CLS:CLT ↵
20 P="C ;E ;G ;<C"↵
30 M=RND(20)+1 ↵
40 LC M-1,23:IF M<5 ?M ELSE ? ↵
50 LC 4,0:?"< SCORE=";S ↵
60 FOR T=1 TO 40 ↵
70 K=INKEY() ↵
80 IF K>0 AND SCR(K-49,0)=K GSB 120 ↵
90 NEXT ↵
100 IF TICK()/60<60 GOTO 30 ↵
110 LC 0,1:END ↵
120 PLAY P+(K-49)*3:S=S+1 ↵
130 LC K-49,0:?" ":RTN ↵
```

point

■ちょっと「良い音」にする



「SOUND」と「GND」にスピーカーをつなぐ



「2-4-1. IchigoJam ミュージック」のプログラムを実行してみよう

```
PLAY "O4 CDEFGAB O5 C2"↵
```

■LED を順番に光らせている間に「ジングルベル」を鳴らす

ここで「エンター」は押さない

```
5 LED 1
10 IF BTN()==0 GOTO 10 ELSE LED0↵
20 PLAY "T180 L8 O5 ERERER4R4ERERE4R4
ERGRC4.DE2R2 FRFRF4RFF4ERE4R
EE4DRDRCRDR4RG4R4" ↵
25 X=0 ↵
30 FOR A=1 TO 3 ↵
40 OUT A,1:WAIT 20:OUT A,0 ↵
50 NEXT ↵
55 X=X+1 ↵
60 IF X<10 GOTO 30 ↵
70 GOTO 5 ↵
```

O1 ～ O5　まで自由に変更してみる

■LED をランダムに光(ひか)らせている間(あいだ)に「ジングルベルを鳴(な)らす」

```
5 LED 1
10 IF BTN()==0 GOTO 10 ELSE LED0↵
20 PLAY "T180 L8 O5 ERERER4R4ERERE4R4
ERGRC4.DE2R2 FRFRF4RFF4ERE4R
EE4DRDRCRDR4RG4R4" ↵
25 X=0 ↵
30 A=RND(3)+1↵
40 OUT A,1:WAIT 15:OUT A,0:WAIT 5↵
55 X=X+1↵
60 IF X<30 GOTO 30↵
70 GOTO 5 ↵
```

ここで「エンター」は押(お)さない

■LED を点滅(てんめつ)させている間(あいだ)に「ジングルベルを鳴(な)らす」

```
5 LED 1
10 IF BTN()==0 GOTO 10 ELSE LED0↵
20 PLAY "T180 L8 O5 ERERER4R4ERERE4R4
ERGRC4.DE2R2 FRFRF4RFF4ERE4R
EE4DRDRCRDR4RG4R4" ↵
25 X=0 ↵
30 OUT 1,1:OUT 2,1:OUT 3,1:WAIT 10↵
40 OUT 1,0:OUT 2,0:OUT 3,0:WAIT 10↵
55 X=X+1 ↵
60 IF X<30 GOTO 30↵
70 GOTO 5 ↵
```

3-1. アイロンビーズ作品づくり手順 1

■用意するもの

1）ビーズ（商品名はパーラービーズ）
2）ピンセット状のもの
3）プレート
4）アイロン（スチームがないほうがつかいやすい）
5）アイロンペーパー
6）重しになるもの（本など）

1.
つくりたいものを決めます。好きな形のプレートと色を決めましょう。
ここでは緑色のビーズでハートの形をつくります。
手のひらにのせると、つまみやすいようです。

2
ビーズを並べ終えたら、アイロンペーパーをかぶせてアイロンをかけます。

3
ゆっくりアイロンをかけましょう。温度は中温に設定していますが、ゆっくり、ちょっと低めでかけると、ビーズがつぶれません。
ビーズは、色によってとける温度や時間に差があります。
透明のビーズのほうが半透明なものより時間がかかるようです。

4
アイロンペーパーのむこうで、ビーズが溶け出しているのがわかりますね。

5
ビーズ全体が溶け出したのを確認したら、本などの重しをのせて変形するのを防ぎます。
さめるまで待ちます。
室温の高い夏はなかなかさめませんが、あせらずに十分にさめるまで待ちましょう。

6
ビーズがさめて、片面はとけて、ちゃんとつながりました。

7
プレートからハート型のビーズを慎重にはずして反対側にも同じようにアイロンをかけ、ビーズがとけるようすを確認して、重しをのせ、さめるまで待ちます。

8
緑のビーズハートの完成です。

3-2.LED でキラキラあそび (きほんへん)1

■ざいりょう

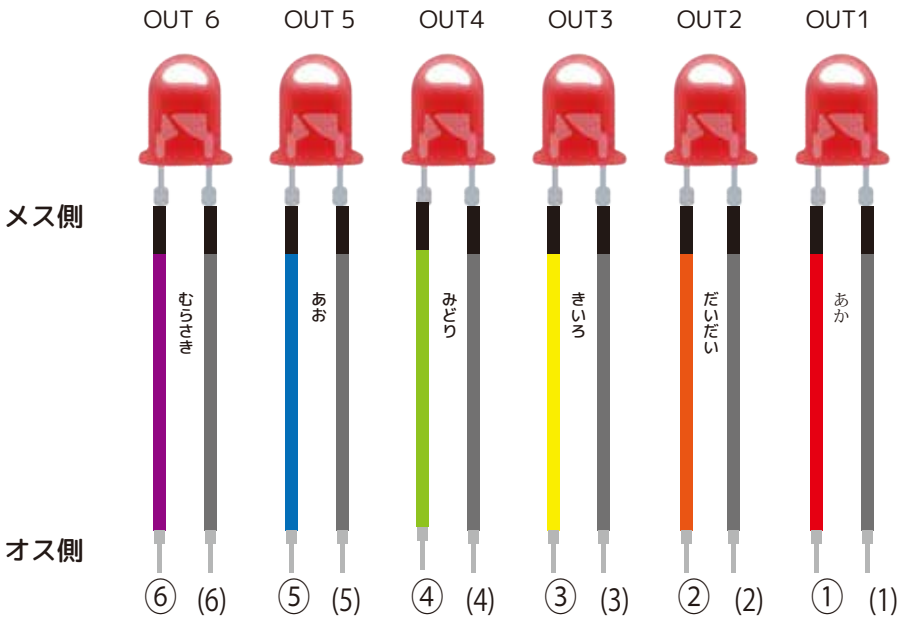
- ■ IchigoJam
- ■ LED　６つ　　３mm Φ・5mm Φ・10mm Φ
  おこのみにあわせてつかってください。
- ■ ジャンパーワイヤー　　メスーオス　１２本
  オスーオス　　２本
- ■ ブレッドボード　１枚　本説明書では aitendo の mini ブレッドボード BBD-ZY25 を使用しています。

■つなぎかた

LED について

※ニッパーであらかじめてきとうな長さにＬＥＤの足を切るとさしやすいです。

LED にメスーオスのジャンパーワイヤーをさします

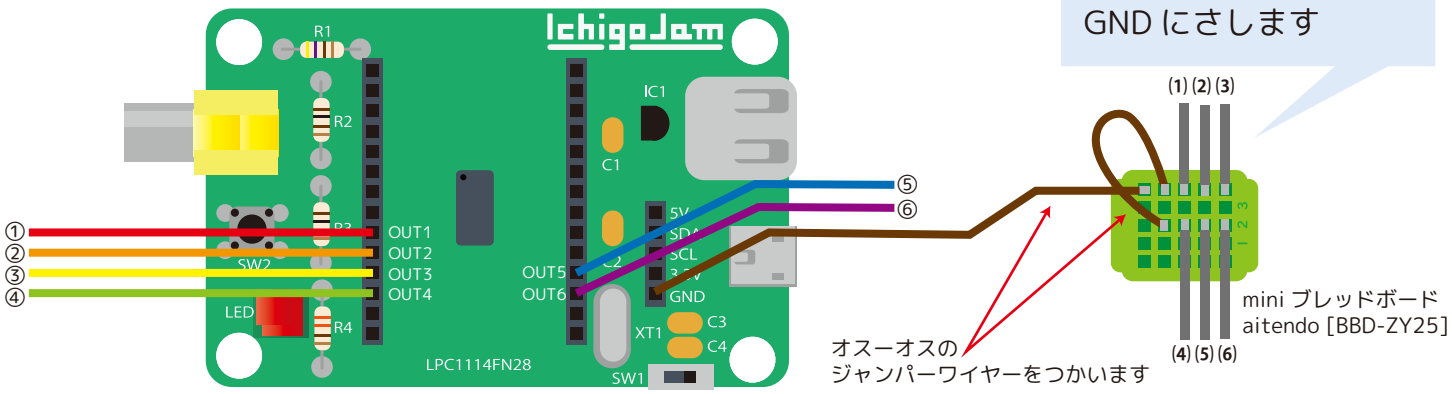


さす向きにちゅういしよう！

※ジャンパーワイヤーはつかうポートごとに色をかえるととてもわかりやすいです。

■ IchigoJam に LED をとりつけたジャンパーワイヤーをさします

カソード側はブレッドボードでひとまとめにしてから GND にさします



■６つのＬＥＤがすべて点灯(てんとう)するか確認(かくにん)してみよう

```
OUT`111111
```

※ (`) の文字は [shift]+@ と打つとでるよ

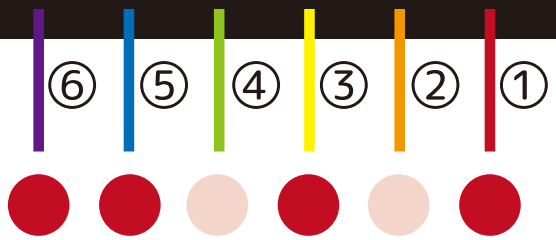
■しくみ

１がＯＮ（光る）、０がＯＦＦ（きえる）だ

２進数でかくことで同時に命令をおくることができる

（例）

```
OUT`110101
```



この場合は、
ＯＵＴ１とＯＵＴ３・ＯＵＴ５・ＯＵＴ６がＯＮ
ＯＵＴ２とＯＵＴ４がＯＦＦになる

■サンプルプログラム

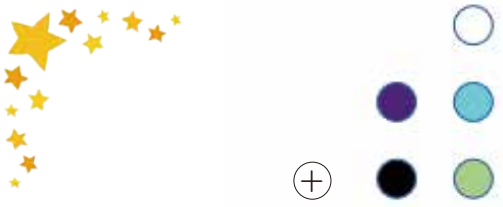
```
5  T=30
10 OUT`00001:WAIT T
20 OUT`00011:WAIT T
30 OUT`00111:WAIT T
40 OUT`01111:WAIT T
50 OUT`11111:WAIT T
60 OUT`00000:WAIT T
70 GOTO 10
```

アレンジレシピ

どんなふうに光るかな？
アレンジレシピためしてみてね！

3-3. ちょうちょ 1

ふわふわキラキラ光ります



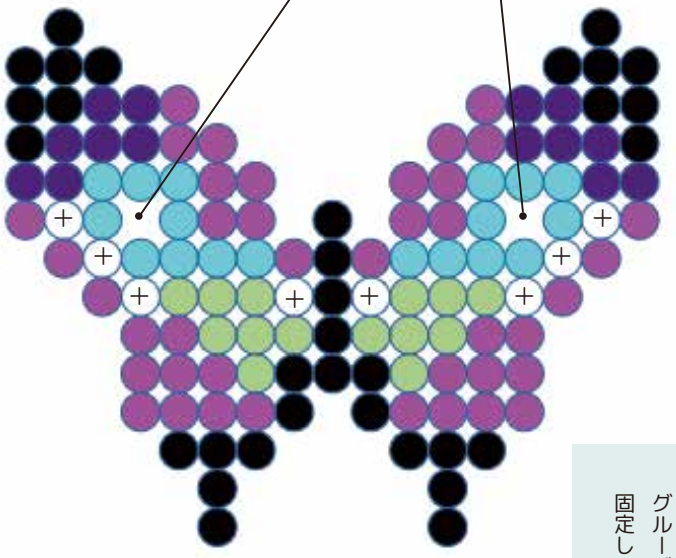
ちょうちょの色や LED の場所
などステキにアレンジしてね！

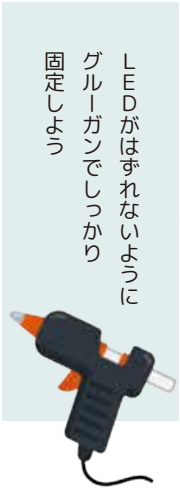
パーラービーズ配置例

夜光あお	5072
夜光むらさき	5086
ラメむらさき	5046
夜光しろ	5075
くろ	5018
＋ラメとうめい	5041

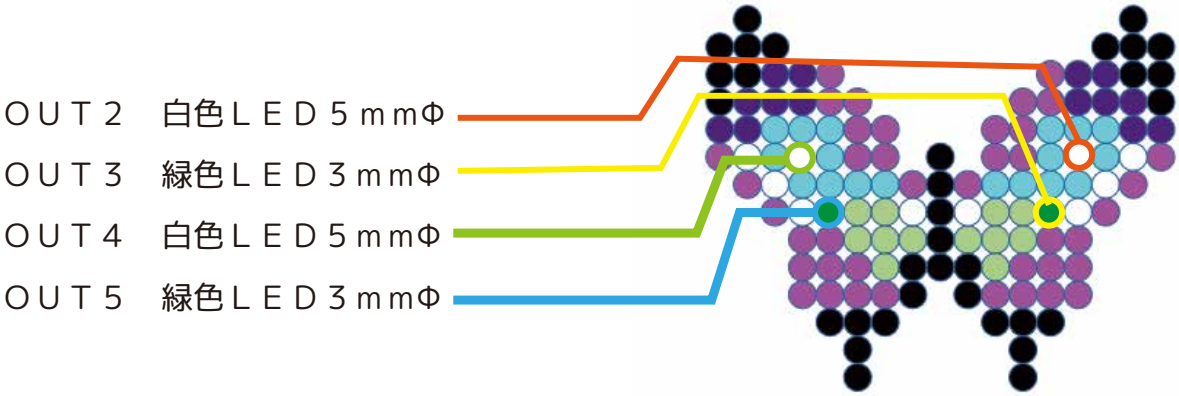
表 面

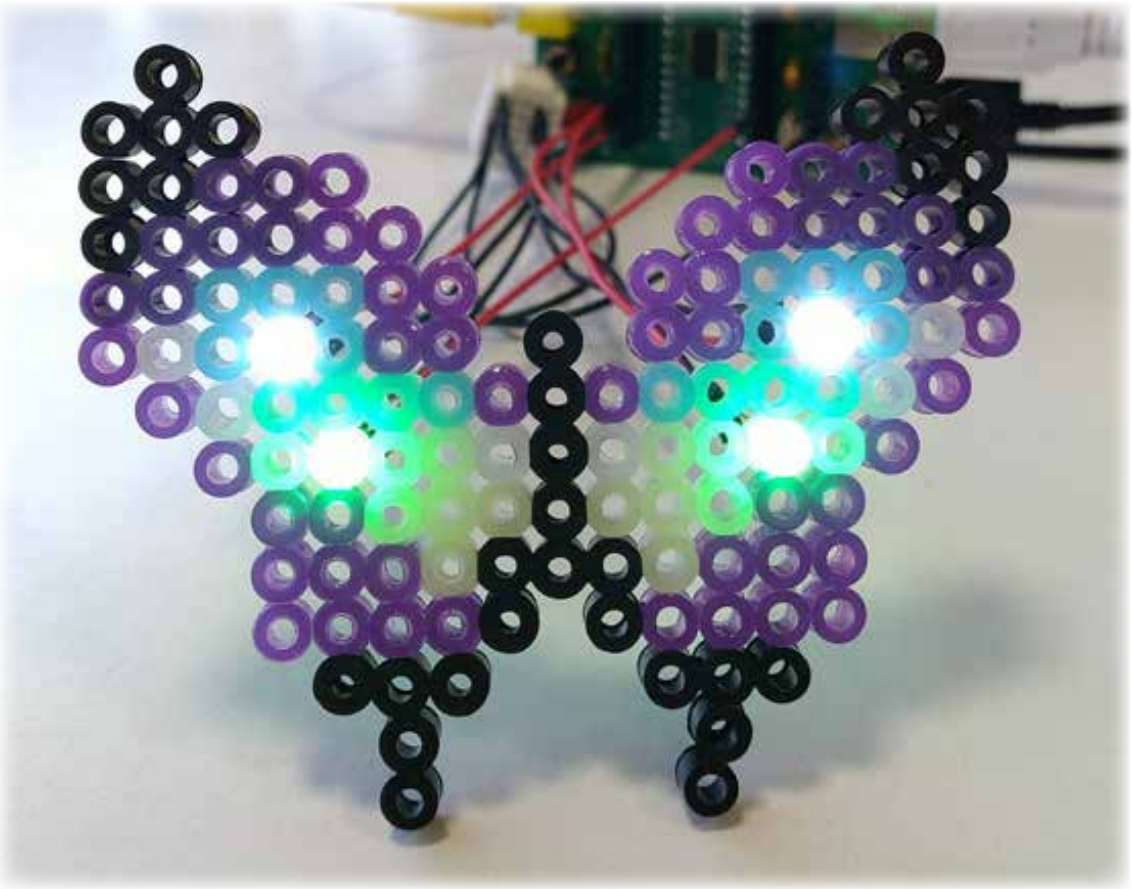
5mm のＬＥＤを
入れるので
１マスあける





裏 面



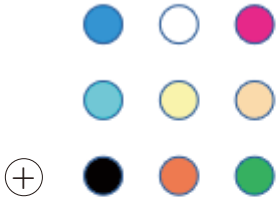


■サンプルプログラム

```
10 FOR J=0 TO 4 : FOR I=-10 TO 10
20 PWM2,ABS(I*20),500
30 PWM3,ABS(I*20-200),500:WAIT 3
40 PWM4,ABS(I*20),500
50 PWM5,ABS(I*20-200),500:WAIT 3
60 NEXT:NEXT:GOTO 100
100 CLO:FOR I=0 TO 10
110 OUT RND(64):WAIT 10:NEXT
120 OUT`1111:WAIT 180:GOTO 10
```

[PWM 命令を使うとＬＥＤをほわほわ光らせることができます]
ＰＷＭはパルス幅変調を行う命令です。
書き方
PWM < 端子 >,< パルス幅 > , < パルス周期 >
< 端子 > は 2 ～ 5 で OUT2 ～ 5 に対応します。
< パルス幅 > は 0.01 ミリ秒単位で HIGH になる時間を設定します。
< パルス周期 > は 0.01 ミリ秒単位でパルスの周期を指定します。

3-4. おひめさま 1

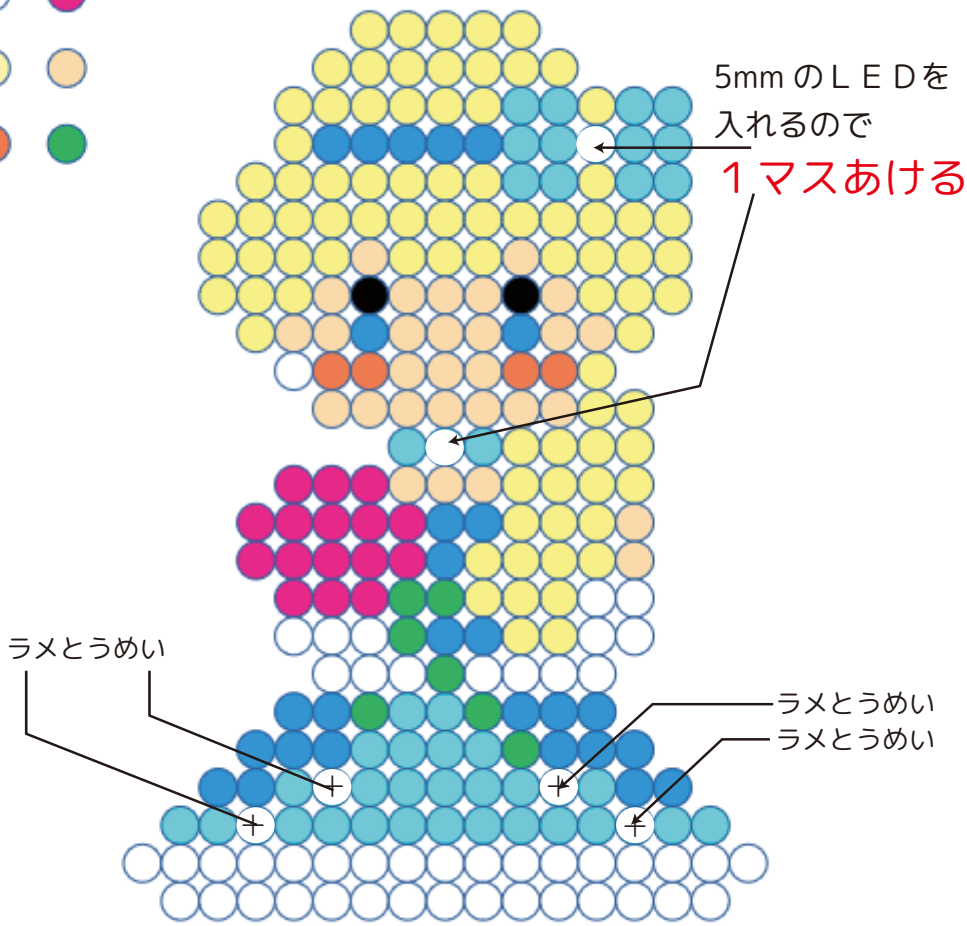


ドレスの色やかみがたなど
ステキにアレンジしてね！

パーラービーズ配置例

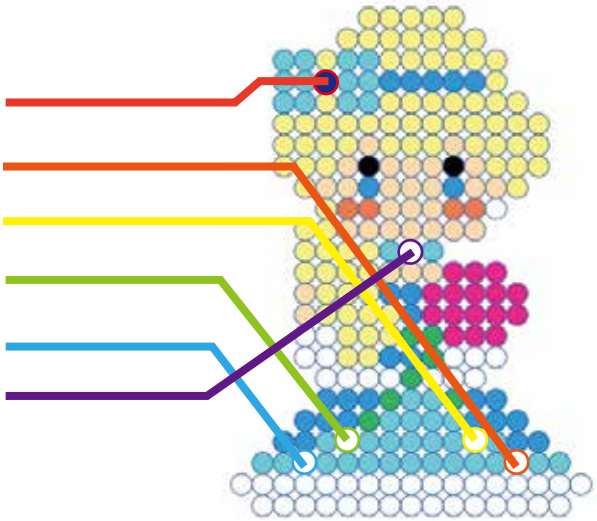
●	よもぎいろ	5080
●	パステルあお	5042
●	夜光あお	5072
●	しろ	5001
●	つつじいろ	5083
●	クリーム	5002
●	ピーチ	5033
●	きいろ	5003
○	ラメとうめい	5041

表　面



裏　面

ＯＵＴ１　リボン　　青色ＬＥＤ 5mmΦ
ＯＵＴ２　スカート　白色ＬＥＤ３ｍｍΦ
ＯＵＴ３　スカート　白色ＬＥＤ３ｍｍΦ
ＯＵＴ４　スカート　白色ＬＥＤ３ｍｍΦ
ＯＵＴ５　スカート　白色ＬＥＤ３ｍｍΦ
ＯＵＴ６　チョーカー 白色ＬＥＤ５ｍｍΦ



ＬＥＤがはずれないように
グルーガンでしっかり
固定しよう

3-4. おひめさま 2

■サンプルプログラム

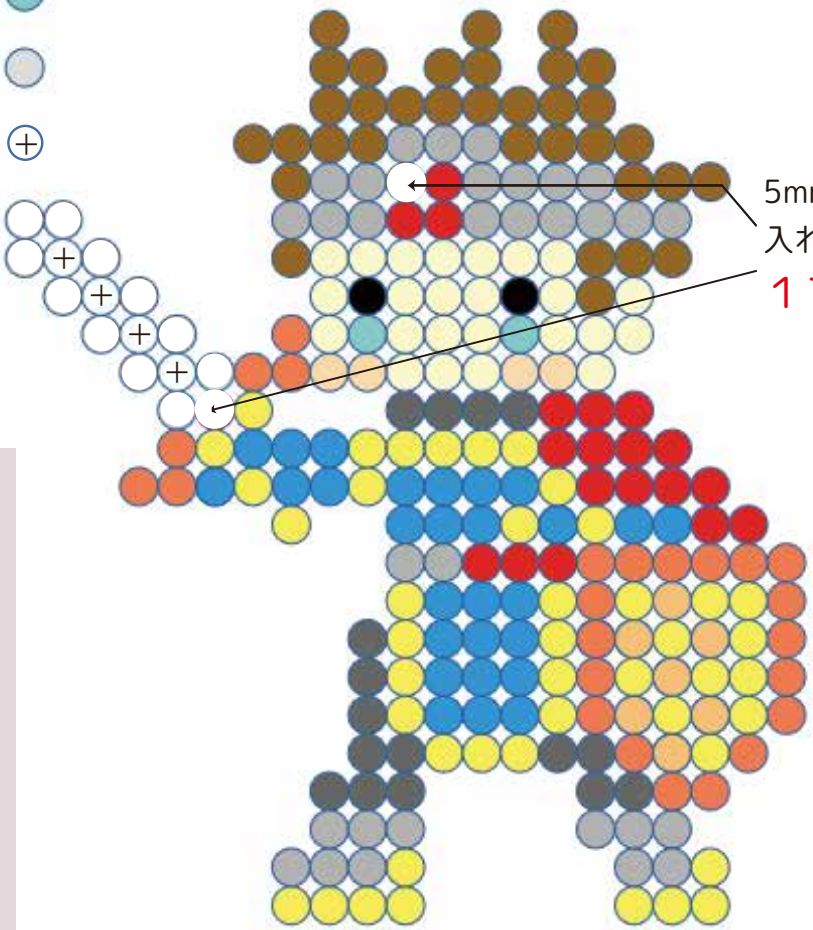
```
5 W=60
10 OUT`100001:WAIT W
20 OUT`110011:WAIT W
30 OUT`101101:WAIT W
40 OUT`111111:WAIT W*3
50 FOR J=0 TO 5
60 FOR I=-10 TO 10
65 A=ABS(I*50)
70 PWM2,A,500
80 PWM3,ABS(A-500),500
90 PWM4,A,500
100 PWM5,ABS(A-500),500
110 WAIT 3:NEXT:NEXT
120 CLO:GOTO 10
```

[PWM 命令を使うとＬＥＤをほわほわ光らせることができます]
ＰＷＭはパルス幅変調を行う命令です。
書き方
PWM < 端子 >,< パルス幅 > , < パルス周期 >
< 端子 > は 2 ～ 5 で OUT2 ～ 5 に対応します。
< パルス幅 > は 0.01 ミリ秒単位で HIGH になる時間を設定します。
< パルス周期 > は 0.01 ミリ秒単位でパルスの周期を指定します。

3-5. ゆうしゃ 1

力をためて剣が光りだすような
えんしゅつにしてみました

表　面

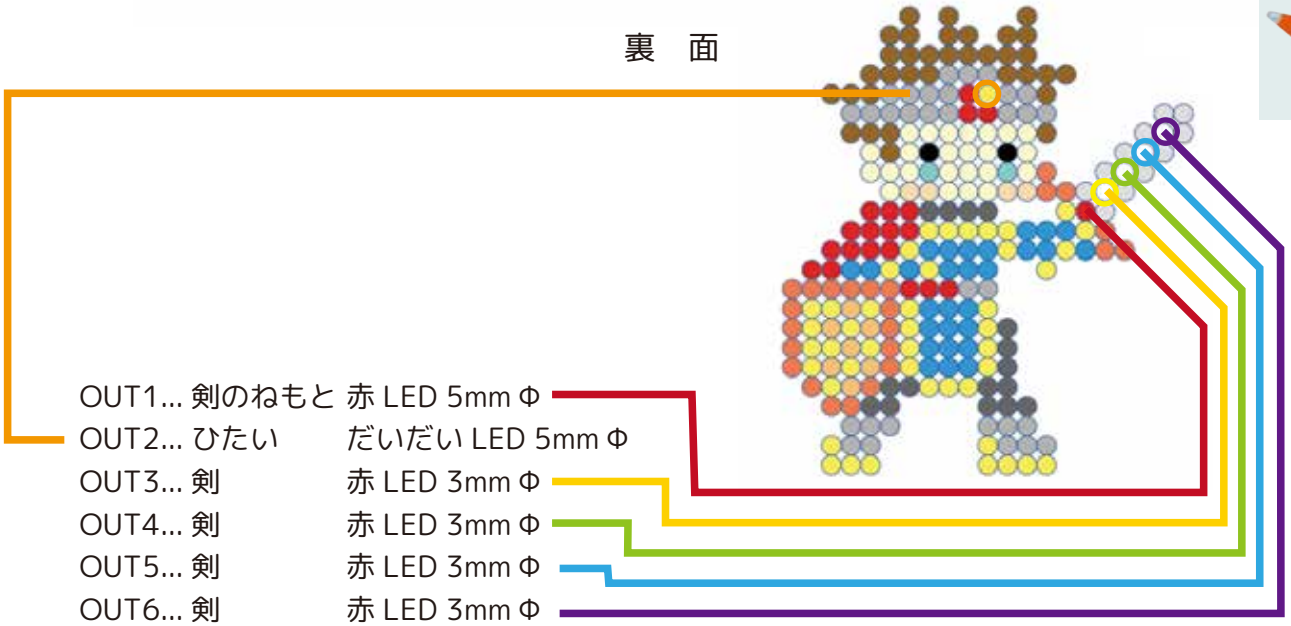


ゆうしゃの服や光る剣の色など
アレンジして楽しんでね！

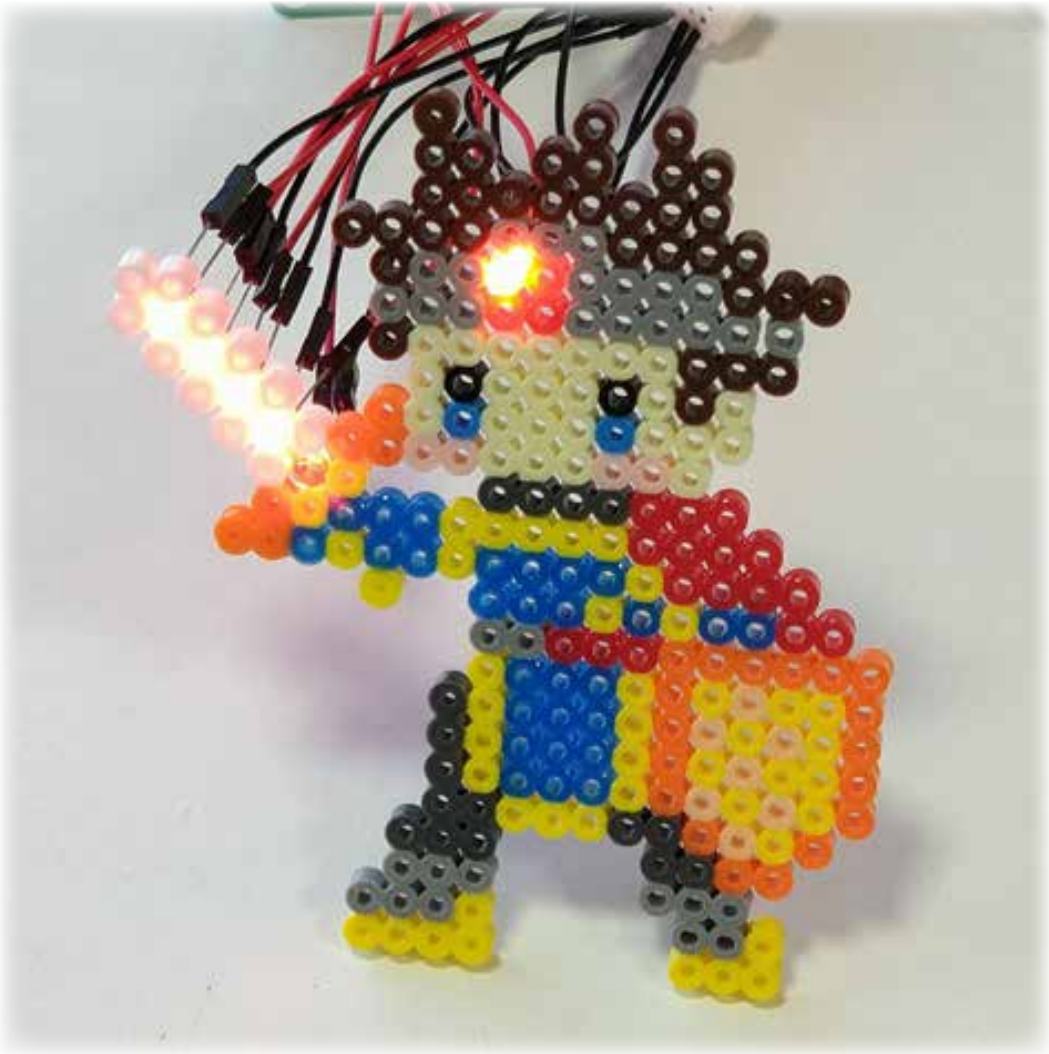
クリーム	5002
ピーチ	5033
オレンジ	5004
きいろ	5003
アプリコット	5098
ラメあお	5044
ラメあか	5042
しろ	5001
ラメとうめい	5041
こげちゃいろ	5012
ダークグレー	5092
はいいろ	5017
くろ	5018
みずいろ	5009

ＬＥＤがはずれないように
グルーガンでしっかり
固定しよう

裏　面



3-5. ゆうしゃ 2



■サンプルプログラム

```
5 CLO
10 OUT`000101:WAIT 20
20 OUT`001101:WAIT 20
30 OUT`011101:WAIT 20
40 OUT`111101:WAIT 20
50 OUT`111111:WAIT 60:OUT 3
60 FOR K=0 TO 9:FOR I=2 TO 6
70 OUT I,1:WAIT(10-K)
80 NEXT:OUT3:NEXT
90 FOR J=0 TO 2 :FOR I=-10 TO 10
100 PWM2,ABS(I*20),500:WAIT 3
110 NEXT:NEXT:GOTO 5
```

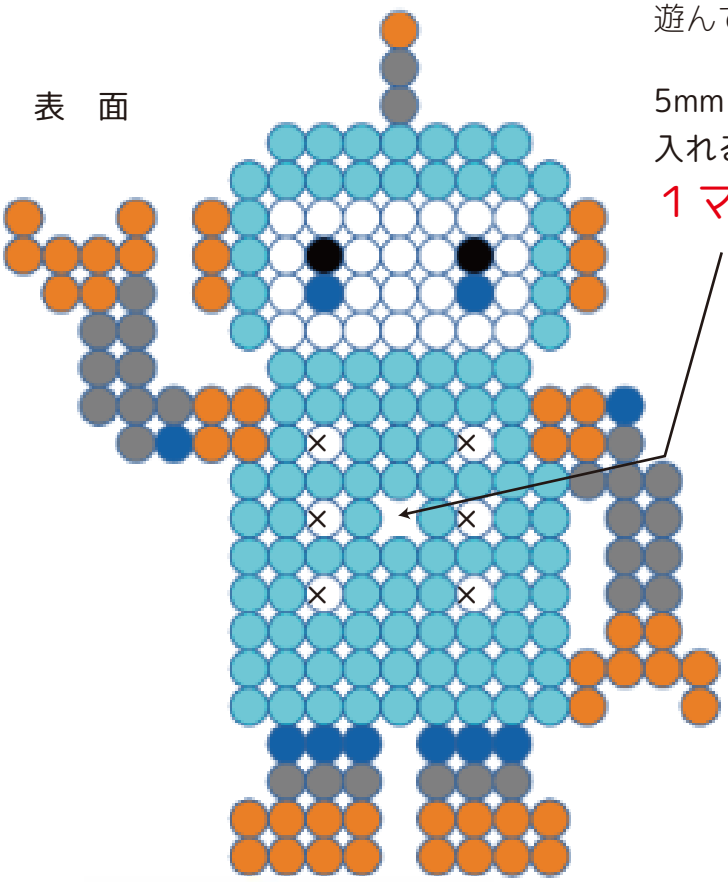
3-6. サイコロロボット 1

ボタンをおすとサイコロの目を出してくれる
楽しいロボットを考えてみました。
遊んでみてね。

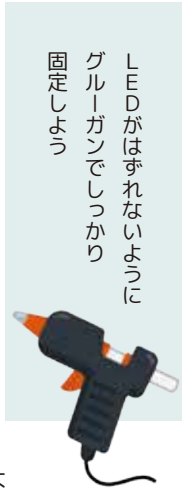
ロボットの色や形など
ステキにアレンジしてね！

- みずいろ　5009
- はいいろ　5017
- オレンジ　5004
- しろ　5001
- くろ　5018
- ラメあお　5044
- × ラメとうめい　5041

表　面

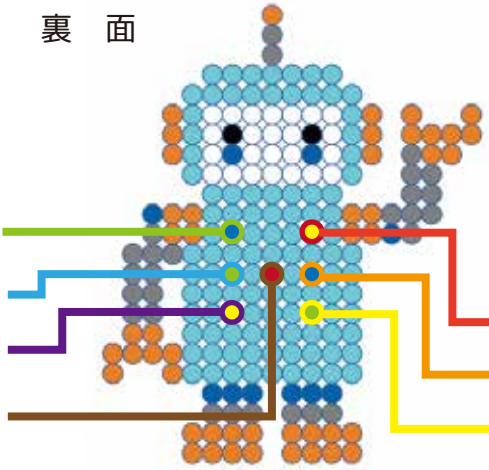


5mm のＬＥＤを
入れるので
１マスあける

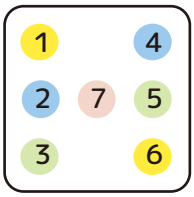


ＬＥＤがはずれないように
グルーガンでしっかり
固定しよう

裏　面



表がわからは
この順番になるよ



LED のとりつけかた

ＯＵＴ４　青色ＬＥＤ３ｍｍΦ
ＯＵＴ５　緑色ＬＥＤ３ｍｍΦ
ＯＵＴ６　黄色ＬＥＤ３ｍｍΦ
ＯＵＴ７　赤色ＬＥＤ５ｍｍΦ
[LED] のポートにさす

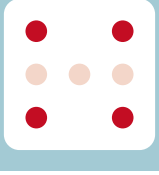
ＯＵＴ１　黄色ＬＥＤ３ｍｍΦ
ＯＵＴ２　青色ＬＥＤ３ｍｍΦ
ＯＵＴ３　緑色ＬＥＤ３ｍｍΦ

光らせてみよう！

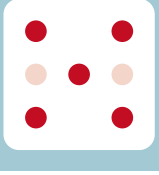
一の目
OUT`1000000

二の目
OUT`0100001

三の目
OUT`1100001



四の目
OUT`0101101



五の目
OUT`1101101

六の目
OUT`0111111

サイコロロボットのプログラミング

■①サイコロの目を一から六までじゅんばんに表示してみよう

```
10 OUT`1000000:WAIT 60
20 OUT`0100001:WAIT 60
30 OUT`1100001:WAIT 60
40 OUT`0101101:WAIT 60
50 OUT`1101101:WAIT 60
60 OUT`0111111:WAIT 60
```

■②サイコロをふれるようにしてみよう

```
?RND(6)
```

```
?RND(6)+1
```

どんなすうじがでたかな？
RND（ランダム）は（）（かっこ）の中の数だけ面があるサイコロです。 あれ？でも６が出ない？
これは０をあわせた６つの数字が出るといういみです。
（）（かっこ）のなかが６のばあいは０、１、２、３、４、５の６つになります。
０を出したくないな、というときは RND(6)+1 にかきかえてみよう

```
5 IF BTN()=0:WAIT 6:GOTO 5
8 A=RND(6)+1:GOTO A*10
10 OUT`1000000:GOTO 100
20 OUT`0100001:GOTO 100
30 OUT`1100001:GOTO 100
40 OUT`0101101:GOTO 100
50 OUT`1101101:GOTO 100
60 OUT`0111111:GOTO 100
100  ?A:WAIT 60:GOTO 5
```

BTN( ) のコマンド
ボタンがおされていれば１(ひかる) そうでないとき０(きえる)
じっけんしてみて

```
LED BTN()
```

ボタンをおすとサイコロの目がでるようになったかな？

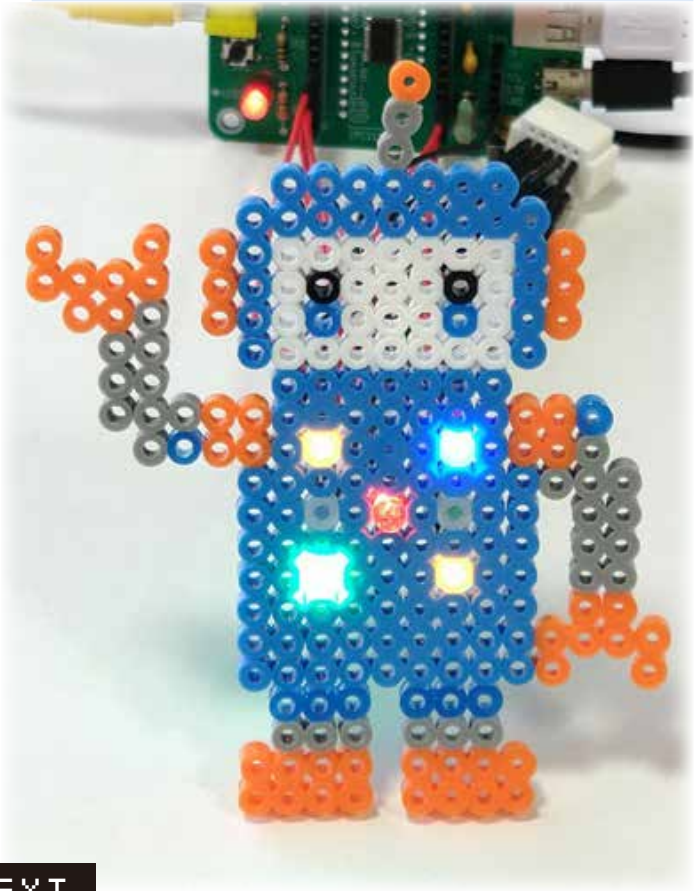
■③楽しいえんしゅつを追加しよう

```
OUT RND(64)
```

ロボットの７つの LED がめちゃくちゃに光るよね

```
6 FOR I=0 TO 5:OUT RND(64)
7 WAIT 10:NEXT
```

これを加えると、ボタンをおしてからサイコロの目がでるまでめちゃくちゃに光るから何が出るかわからなくてドキドキしちゃう。



■④ブザーからピコピコ音もだしちゃおう

圧電ブザーを SOUND と GND 端子にさしてね

```
BEEP 5
BEEP 20
BEEP 20,30
```

←たかいおと
←ひくいおと
←ながいおと

７ぎょうめをこんなふうにかきかえるとピコピコ楽しい！

```
7 BEEP RND(20),10:WAIT 10:NEXT
```

文字、数字の読み方と意味

コマンド	解　説	例
LED 数 / エルイーディー	数が 1 なら光り、0 なら消える	LED 1
WAIT 数 1{, 数 2} / ウェイト	数 1 の数値フレーム分待つ 60 で約 1 秒	WAIT 60
: / コロン	コマンドを連結する	WAIT 60:LED 1
行番号 コマンド	プログラムとしてコマンドを記録する	10 LED1
行番号	指定した行番号のプログラムを消す	10
RUN / ラン	プログラムを実行する [F5]	RUN
LIST { 行番号 1{, 行番号 2}} / リスト	プログラムを表示する [F4]	LIST 10,300
GOTO 行番号 / ゴートゥー	指定した行番号へ飛ぶ（式も指定可能）	GOTO 10
END / エンド	プログラムを終了する	END
IF 数 {THEN} 次 1 {ELSE 次 2} / イフ · ゼン · エルス	数が 0 でなければ次 1 を実行し、0 であれば次 2 を実行する（THEN,ELSE 以降は省略可）	IF BTN() END
BTN({ 数 }) / ボタン	ボタンが押されていれば 1、そうで無いとき 0 を返す	LED BTN()
NEW / ニュー	プログラムを全部消す	NEW
PRINT { 数や文字列 } / プリント	文字を表示する（文字列は " で囲む、";" で連結できる）省略形：?	PRINT "HI!"
LOCATE 数 1, 数 2{, 数 3} / ロケート	次に文字を書く位置を横、縦の順に指定する（左上が 0,0、縦 =-1 で無表示）。数 3 が 0 でなければ指定した場所にカーソルを表示する。 省略形：LC	LOCATE 3,3
CLS / クリア スクリーン	画面を全部消す	CLS
RND(数) / ランダム	0 から数未満の正数をランダムに返す	PRINT RND(6)
SAVE { 数 } / セーブ	プログラムを保存する（0 ～ 3 の 4 つ）ボタンを押した状態で起動すると 0 番を読み込み自動実行	SAVE 1
LOAD { 数 } / ロード	プログラムを読み出す（0 ～ 3 の 4 つ省略で前回使用した数）	LOAD
FILES { 数 1{, 数 2}} / ファイルズ	数 1(省略可) ～数 2 のプログラム一覧を表示する（EEPROM 内ファイル表示に対応、0 指定ですべて表示、ESC で途中停止）	FILES
BEEP { 数 1{, 数 2}} / ビープ	BEEP を鳴らす 周期 (1-255) と長さ (1/60 秒単位) は省略可 ※ SOUND(EX2)-GND に圧電サウンダーなどの接続必要	BEEP
PLAY {MML} / プレイ	MML で記述した音楽を再生する MML 省略で停止 ※ SOUND(EX2)-GND に圧電サウンダーなどの接続必要（次項の MML 参照）	PLAY "$CDE2CDE2"
TEMPO 数 / テンポ	再生中の音楽のテンポを変更する	TEMPO 1200
数 + 数	足し算する	PRINT 1+1
数 - 数	引き算する	PRINT 2-1
数 * 数	掛け算する	PRINT 7*8
数 / 数	割り算する（小数点以下は切り捨て）	PRINT 9/3
数 % 数	割り算した余りを返す	PRINT 10%3
(数)	カッコ内を優先して計算する	PRINT 1+(1*2)
LET 変数 , 数 / レット	アルファベット 1 文字を変数として数の値を入れる（配列に連続代入可能 LET[0],1,2）省略形：変数 = 数	LET A,1
INPUT { 文字列 ,} 変数 / インプット	キーボードや UART からの入力で数値を変数にいれる（文字列とコンマは省略可）	INPUT "ANS?",A
TICK() / ティック	CLT からの時間を返す（約 1/60 秒で 1 進む）	PRINT TICK()
CLT / クリア ティック	時間をリセットする	CLT
INKEY() / インキー	キーボードや UART から 1 文字入力する（入力がない時は 0、UART から 0 が入力された時は #100）	PRINT INKEY()
定数	LEFT=28, RIGHT=29, UP=30, DOWN=31, SPACE=32	IF INKEY()=SPACE LED1
CHR$(数) / キャラ	文字コードに対応する文字を返す（コンマ区切りで連続表記可）	PRINT CHR$(65)
ASC(" 文字 ") / アスキー	文字に対する文字コードを返す	PRINT ASC("A")
SCROLL 数 / スクロール	指定した方向に 1 キャラクター分スクロールする（0/UP: 上、1/RIGHT: 右、2/DOWN: 下、3/LEFT: 左）	SCROLL 2
SCR({ 数 , 数 }) / スクリーン	画面上の指定した位置に書かれた文字コードを返す（指定なしで現在位置）別名：VPEEK	PRINT SCR(0,0)
数 = 数	比較して等しい時に 1、それ以外で 0 を返す (== でも可)	IF A=B LED 1
数 <> 数	比較して等しくない時に 1、それ以外で 0 を返す (!= でも可)	IF A<>B LED 1
数 <= 数	比較して以下の時に 1、それ以外で 0 を返す	IF A<=B LED 1
数 < 数	比較して未満の時に 1、それ以外で 0 を返す	IF A<B LED 1
数 >= 数	比較して以上の時に 1、それ以外で 0 を返す	IF A>=B LED 1
数 > 数	比較してより大きい時に 1、それ以外で 0 を返す	IF A>B LED 1
式 AND 式 / アンド	どちらの式も 1 の時に 1、それ以外で 0 を返す (&& でも可)	IF A=1 AND B=1 LED 1
式 OR 式 / オア	どちらかの式が 1 の時に 1、それ以外で 0 を返す (|| でも可)	IF A=1 OR B=1 LED 1
NOT 式 / ノット	式が 0 の時に 1、それ以外で 0 を返す (! でも可)	IF NOT A=1 LED 1
REM / リマーク	これ以降の命令を実行しない（コメント機能）省略形：'	REM START
FOR 変数 = 数 1 TO 数 2{STEP 数 3}　NEXT / フォー · トゥー · ネクスト	変数に数 1 をいれ、数 2 になるまで数 3 ずつ増やしながら NEXT までをくりかえす（STEP は省略可、6 段まで）	FOR I=0 TO 10:?I:NEXT
OUT 数 1{, 数 2} / アウト	外部出力 OUT1-11 に 0 または 1 を出力する 数 2 を省略でまとめて出力できる（数 2 に -1 指定で IN へ切り替え、-2 指定でプルアップ付き IN へ切り替え ※ IN3 は除く）	OUT 1,1
IN({ 数 }) / イン	IN0-10 から入力する（0 または 1）数を省略してまとめて入力できる（IN0,1,4,9 はプルアップ、IN5-8,10-11 は OUT で切り替え時使用可能、IN0,9 はボタン）	LET A,IN(1)
ANA({ 数 }) / アナログ	外部入力の電圧 (0V-3.3V) を 0-1023 の数値で返す (2:IN2、5-8:IN5-8(OUT1-4)、0,9:BTN、省略で 0)	?ANA()
PWM 数 1, 数 2{, 数 3} / ピーダブリューエム	外部出力 OUT2-5 に数 2 で 0.01msec 単位で指定するパルスを出力する（0-2000、周期 20msec）、数 3 で周期を指定（省略時 2000=20msec、マイナス値指定で周期 1/480）	PWM 2,100