

【B-0(2)】IchigoJam で LED を光らせよう (2)

●外付けの LED を光らせる

今度は IchigoJam に付いている LED ではなく、別の LED を外付けして光らせてみましょう。今回は「ブレッドボード」を使って電気回路を作ります。

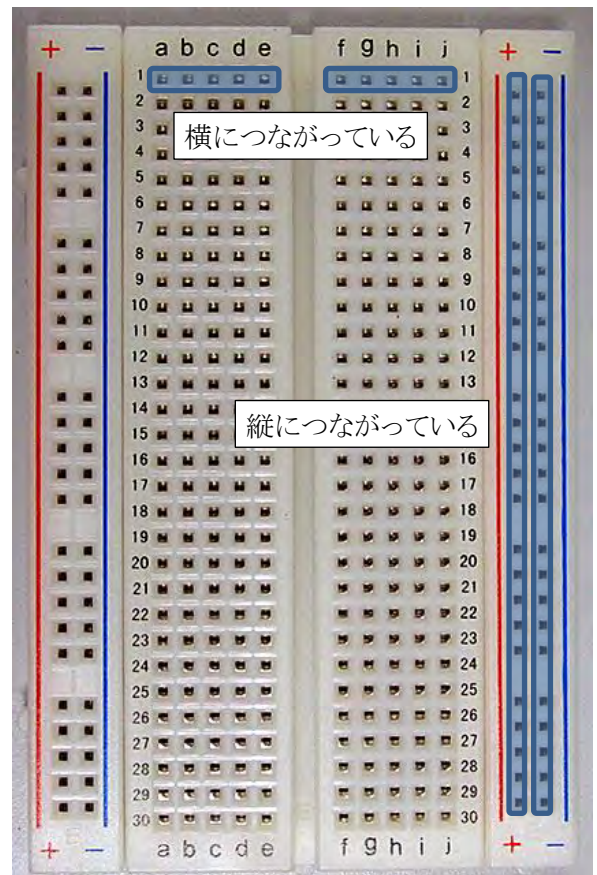
はんだ付けをしなくても電気回路が作れる、便利なボードです。

ボードには、穴がいくつも開いています。

この穴に、部品をさしていきます。

中央の「abcde」「fghij」の穴は、ボードの中で横につながっています。

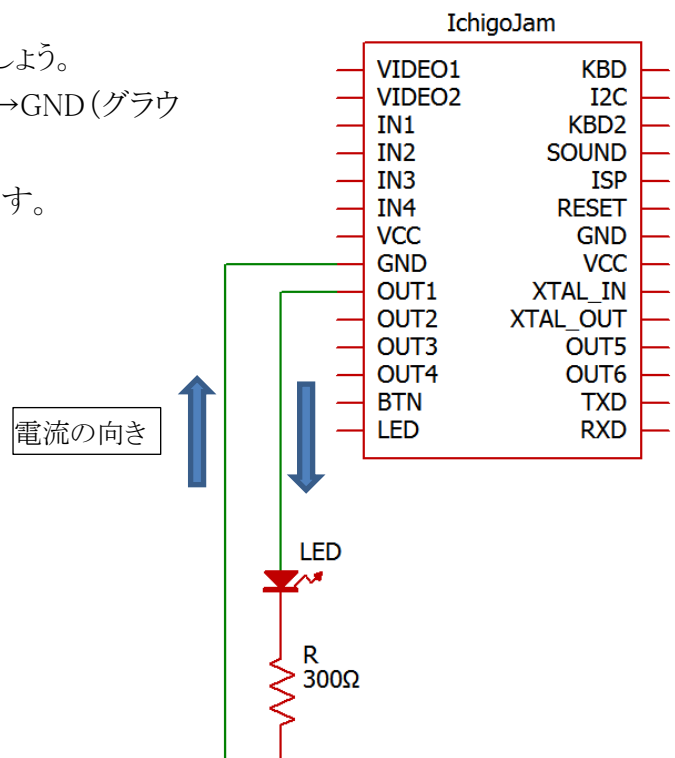
左右にある「+」「-」の穴は、ボードの中で縦につながっています。



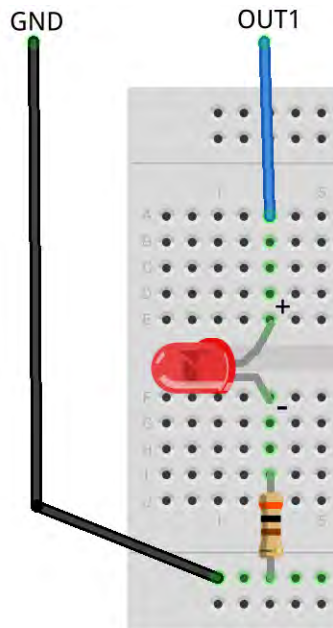
それでは、LED を光らせる回路を作りましょう。

IchigoJam の OUT1 端子→LED→抵抗→GND (グラウンド) 端子、という回路を作ります。

この順番で電流が流れて、LED が光ります。

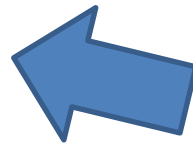


図のように回路を配線します。

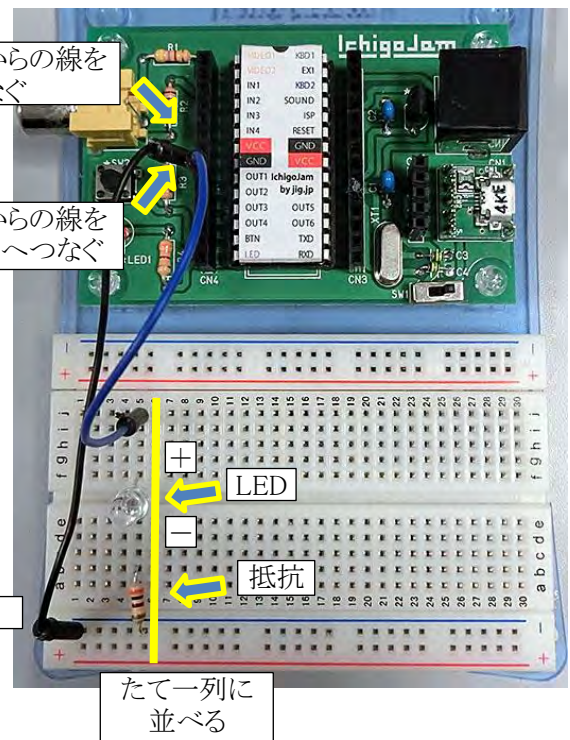


ーのラインからの線を
GND へつなぐ

LED からの線を
OUT1 へつなぐ



ーのライン

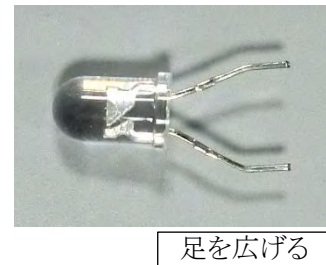
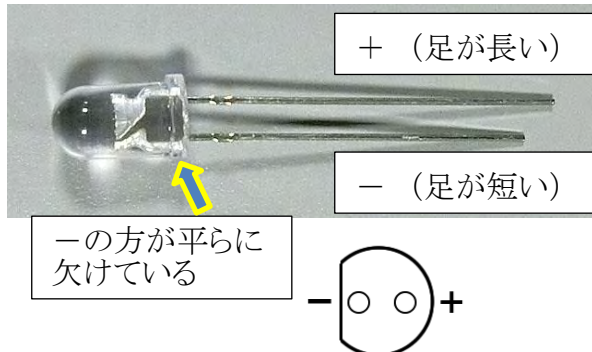


たて一列に
並べる

LED は、電流を流すと光る部品です。

端子は、「+」と「-」があります。向きを間違えると光らないので、注意してください。

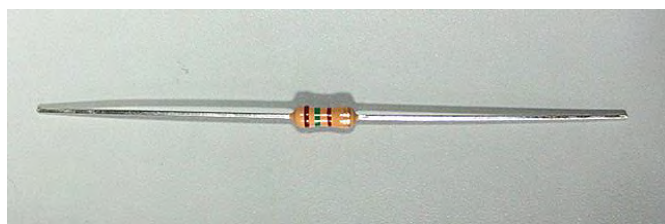
ブレッドボードにさすために、足をみじかく切って広げます。



抵抗は、電流を流れにくくする部品です。LED に流れる電流を少なくするために入れます。

向きはどちら向きでもいいです。

ブレッドボードにさすために、足をみじかく切って曲げます。



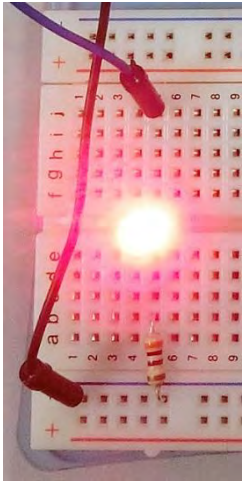
外付け LED の回路ができたら、LED を光らせてみましょう。
まずはダイレクトモードで、以下のプログラムを打ちます。

```
OUT 1,1
```

LED が光ります。

```
OUT 1,0
```

LED が消えます。



OUT (アウト) 命令は、出力ポートに値を出力する命令です。

OUT 1 , 1
 ポート 値
 番号

ポート番号	出力するポートの番号(1～6)。
値	デジタル出力なので、「0」か「1」を指定する。

LED を光らせるには「**OUT 1 , 1**」と入力して、OUT1 ポートに「1」(電圧が高い)を出力します。

GND ポートはいつも「0」(電圧が 0)の状態です。

電圧が高い方から低い方へ電流は流れるので、OUT1→GND の方向へ電流が流れて、LED が光ります。

「**OUT 1 , 0**」と入力して、OUT1 ポートに「0」を出力すると、GND ポートも「0」なので、電流が流れず、LED は消えます。

外付け LED を光らせるプログラムを作ります。

今あるプログラムを消します。

```
NEW
```

続いて、プログラムを入力します。

```
100 OUT 1,1 LED を光らせる
110 WAIT 30 0.5 秒待つ
120 OUT 1,0 LED を消す
```

※あとのプログラムの都合で、行番号を 100 行～にしています。

RUN 命令でプログラムを実行してみましょう。LED が 0.5 秒光ります。

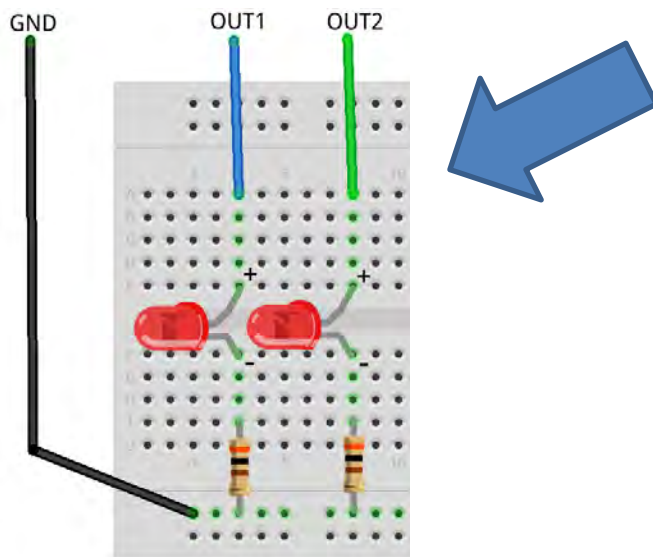
WAIT 命令の数字を変えて、光る時間をいろいろ変えてみましょう。

●LED を 2 個にする

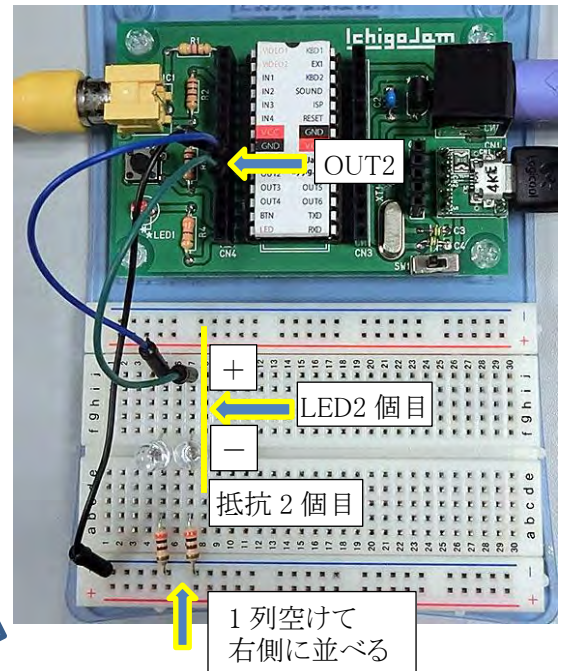
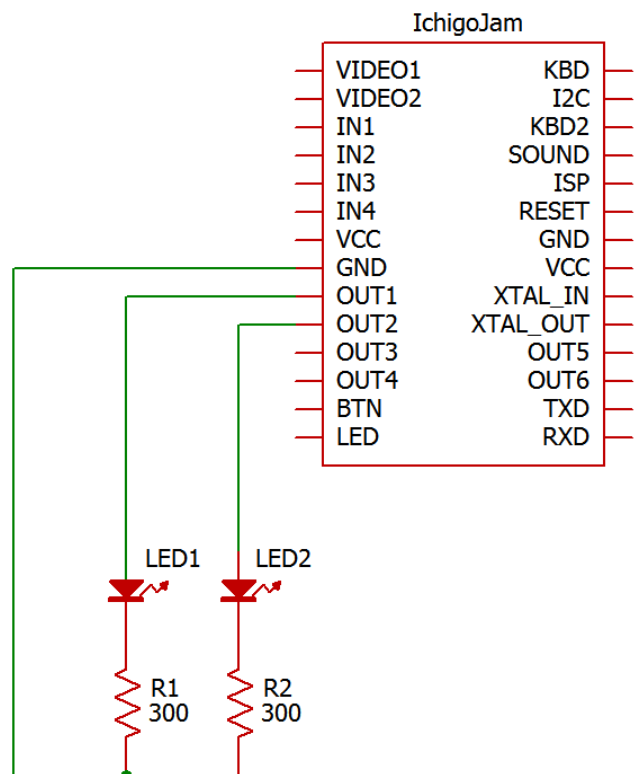
外付けの LED を、2 個に増やしてみましょう。
IchigoJam の OUT2 端子へ、もう 1 個の LED と抵抗をつなげます。

ブレッドボードでは、1 個目の LED のラインから
何列か空けて右側に、2 個目の LED のラインを
並べます。

(列を空けないと、LED がぶつかってしまって差しこめません)



回路図では、右の図のようになります。



まずはダイレクトモードで、2 個目の LED の動作確認をしましょう。

```
OUT 2,1
```

2 個目の LED が光ります。

```
OUT 2,0
```

2 個目の LED が消えます。

次に、プログラムを追加します。

1 個目の LED と 2 個目の LED が、順番に光るようにします。

```
100 OUT 1,1  
110 WAIT 30  
120 OUT 1,0  
130 OUT 2,1  
140 WAIT 30  
150 OUT 2,0
```

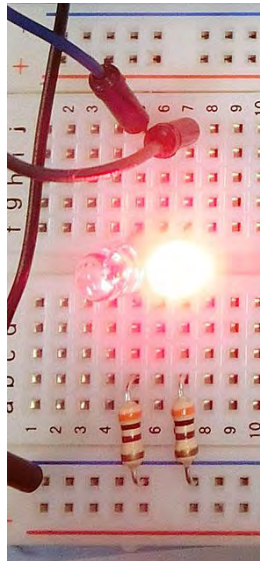
LED2 を光らせる

0.5 秒待つ

LED2 を消す

プログラムを実行してみましょう。

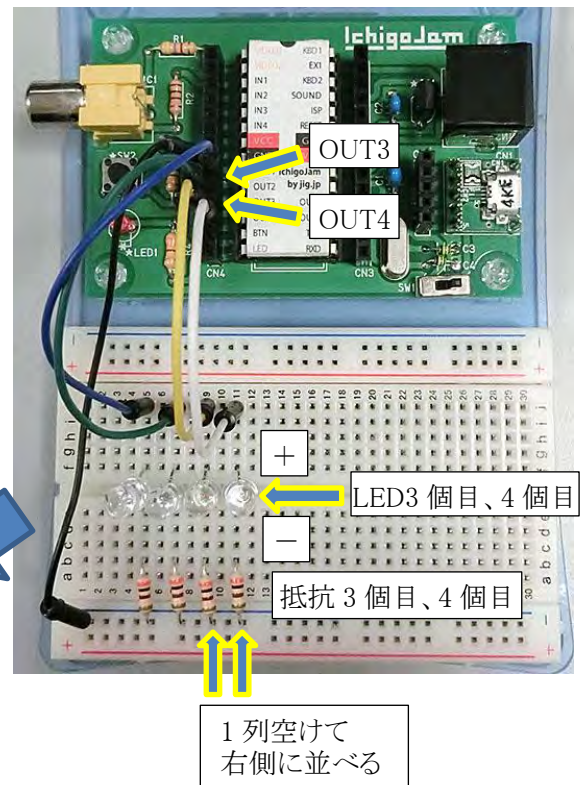
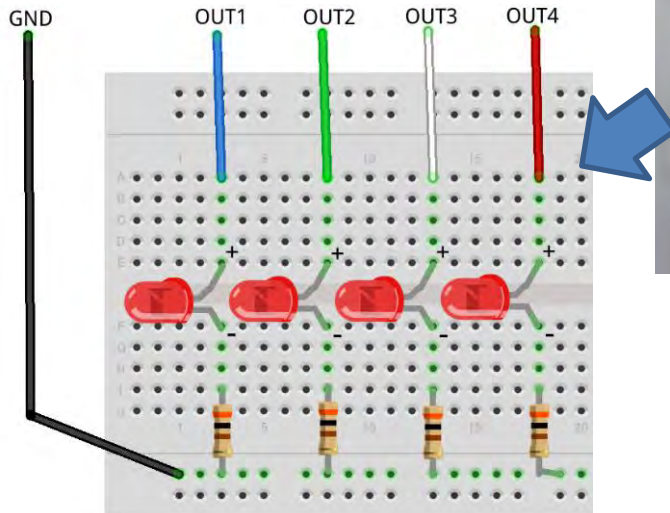
2 個の LED が順番に光ります。



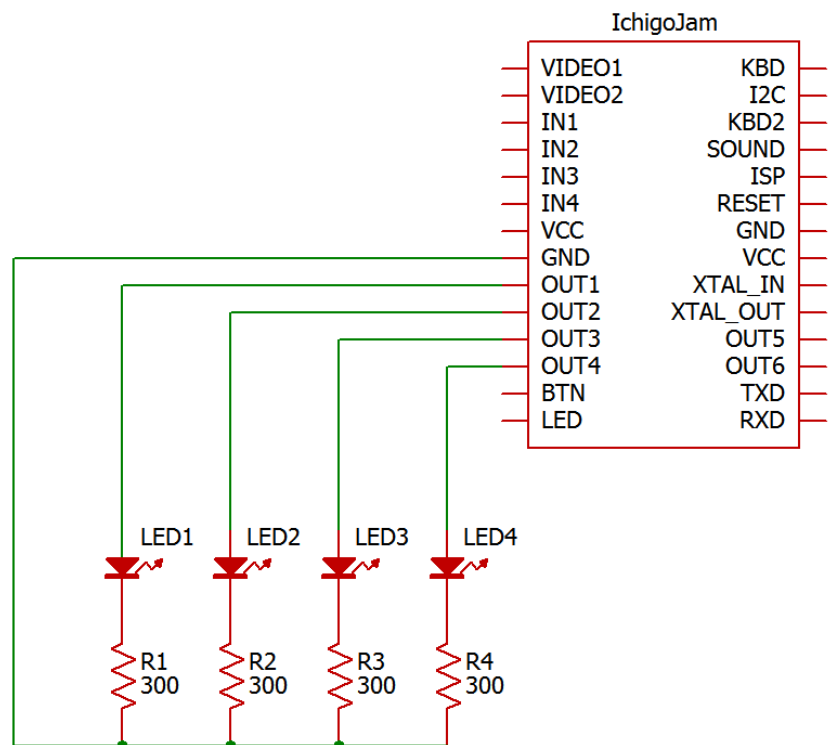
LED1 と LED2 の WAIT の待ち時間をいろいろ変えて、試してみましょう。

●LED を 4 個にする

外付けの LED を、倍の 4 個に増やしましょう。
IchigoJam の OUT3・OUT4 端子へ、2 個の
LED と抵抗をつなげます。
LED がぶつからないように、列を空けて右側へ
並べます。



回路図では、右の図のようになります。



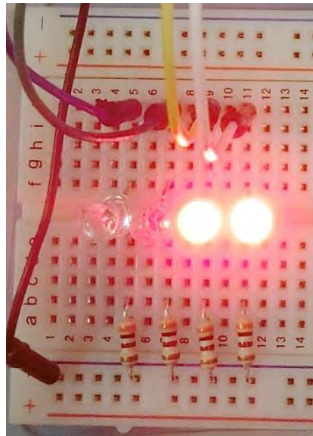
まずはダイレクトモードで、3 個目・4 個目の LED の動作確認をしましょう。

```
OUT 3,1
OUT 4,1
```

3 個目・4 個目の LED が光ります。

```
OUT 3,0
OUT 4,0
```

3 個目・4 個目の LED が消えます。



次に、プログラムを改造します。

2 個目の LED の時と同じように、3 個目・4 個目の LED を光らせるプログラムを追加してもいいのですが、同じようなプログラムを何度も打つのは大変です。

LED を光らせる部分を**サブルーチン**にして、繰り返し呼び出すようにします。

まず、130 行～150 行のプログラム(2 個目の LED を光らせる部分)を、いったん消します。

```
130
140
150
```

行番号だけ打って Enter キーを押すと、その行は削除されます。

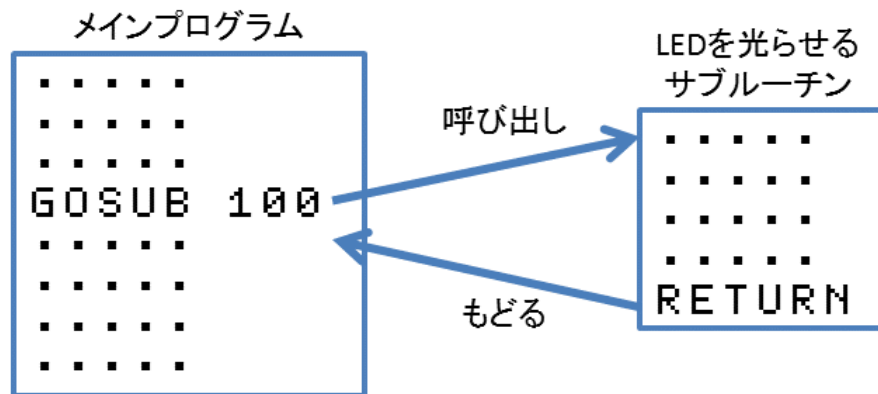
LED を光らせる部分をサブルーチンにして、メインプログラムから繰り返し呼び出します。

10 FOR P=1 TO 4	ポート番号 P を 1 から 4 まで繰り返す
20 GOSUB 100	100 行のサブルーチンを呼ぶ
30 NEXT	FOR へもどって繰り返し
40 END	プログラムを終了する
100 OUT P,1	ポート P の LED を光らせる
110 WAIT 30	0.5 秒待つ
120 OUT P,0	ポート P の LED を消す
130 RETURN	メインプログラムへもどる

プログラムを実行してみましょう。

LED1～LED4 が順番に光ります。

新しく、GOSUB (ゴーサブ) 命令と RETURN (リターン) 命令が出てきました。



「GOSUB 100」で、100 行からのサブルーチンへジャンプします。

サブルーチンのプログラムの処理をして、「RETURN」で、メインプログラムの GOSUB の次の命令へもどります。

このプログラムでは、LED を光らせるプログラムをサブルーチンにして、何度も呼び出しています。こうすると、同じようなプログラムを何回も打たなくて済みます。

また、ポート番号を変数 P にして、FOR～NEXT でくり返して、LED1～LED4 を順番に光らせています。

光らせる順番を逆にしてみましょう。

```

10 FOR P=4 TO 1 STEP -1
20 GOSUB 100
30 NEXT
40 END
  
```

ポート番号 P を 4 から 1 までくり返す

プログラムを実行してみましょう。今度は LED4→LED1 の順で光ります。

今は 1 回ずつ光って終了しますが、ずっとくり返すようにしてみましょう。

```

10 FOR P=4 TO 1 STEP -1
20 GOSUB 100
30 NEXT
40 GOTO 10
  
```

10 行へもどる

プログラムを実行してみましょう。LED4→LED1 の順で、ずっとくり返して光ります。

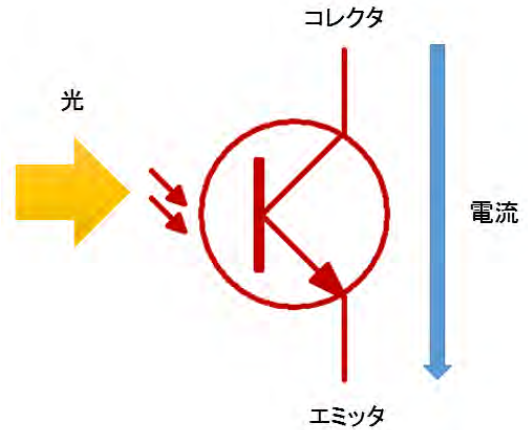
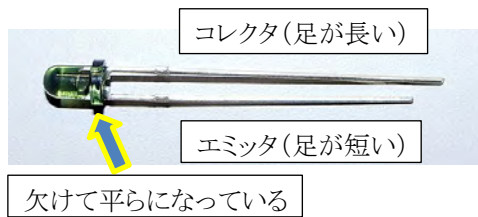
★できる人は

LED を光らせる順番を 1→3→2→4 にするなど、いろいろな順番で光らせるプログラムを考えてみましょう。

●明るさセンサーを使う

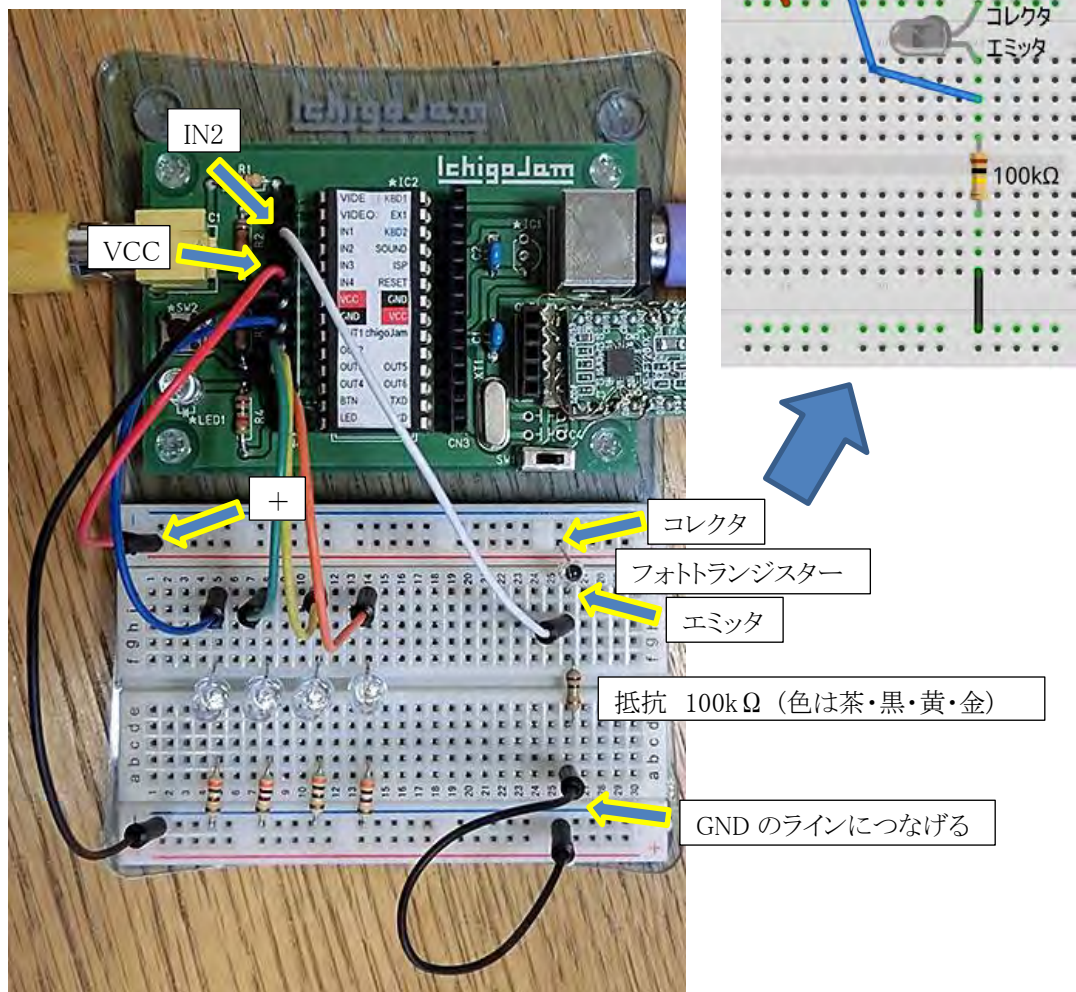
明るさセンサーを使って、周りの明るさによって LED を ON/OFF してみましょう。
今回は明るさセンサーとして、**フォトランジスタ**を使います。

フォトランジスタは、光が当たると
電流が流れる部品です。

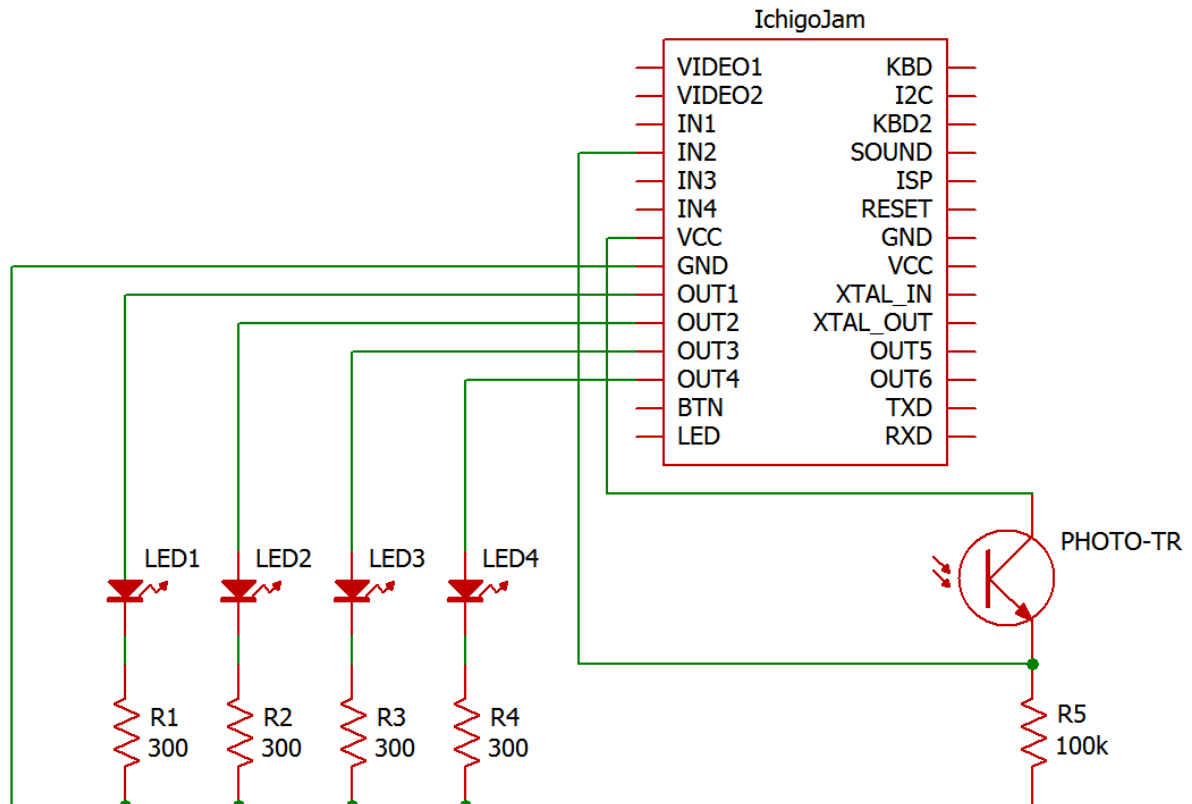


IchigoJam の IN2 に、フォトランジスタと
100k Ω の抵抗をつなげます。

VCC(電源、+)と GND(グラウンド、-)のラインも
必要なので、配線します。



回路図は下のようになります。



まず、フォトトランジスタの回路がちゃんと動作するか確認しましょう。

以下のプログラムを入力します。

```
1 PRINT ANA(2):GOTO 1
```

プログラムを実行してみましょう。画面に数字が連続で表示されます。まわりの明るさによりますが、だいたい 900 を越えたくらいの値になるはずです。

```
945
930
944
:
```

フォトトランジスタに指をおいて、かげにしてみましょう。

数字が 100 以下くらいに小さくなります。(まわりの明るさによって数字は変わります)

全然ちがう数字が表示されたり、かげにしても数字が変化しない場合は、どこか配線をまちがえています。よく見直しましょう。

確認ができれば、

```
1 (Enter)
```

と入力して、1 行目を消しましょう。

この光センサー回路を使って、「周囲が暗くなったら LED が光る」プログラムにしてみましょう。
LED を光らせるサブルーチンを改造します。

```
100 IF ANA(2)<300 THEN OUT P,1
110 WAIT 30
120 OUT P,0
130 RETURN
```

もしポート2のアナログ入力値が300より小さかったら
LEDを光らせる

プログラムを実行してみましょう。そのままだと LED は全然光りません。
フォトリジスタにゆびをおいてかげにすると、LED1～LED4 が順番に光ります。

ANA 関数は、アナログ入力を読み取る関数です。

ANA(2)
ポート
番号

ポート番号	アナログ入力するポートの番号。 「2」=IN2 端子 「5」=BTN 端子 数字を省略すると BTN
返る値	0～1023 の範囲の値が返る。

ここでは「IN2 の入力値が 300 より小さかったら、LED を光らせる」という条件にしています。
「300」の数値をいろいろ変えて試してみましょう。
周囲の明るさによっては、値を調整しないと、LED を ON/OFF できません。

★プログラムをスロット1に保存しましょう。

SAVE 1

★できる人は

フォトリジスタに当たる光の明るさによって、光る LED が移動する速度が変わるプログラムを考えてみましょう。(例:かげにして暗くするほど移動速度が速くなる)

●7 セグメント LED を光らせる回路

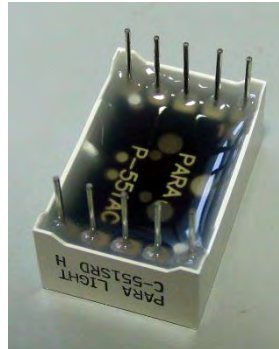
「7 セグメント LED」を光らせる回路を作って、数字を表示してみましょう。

7 セグメント LED は、7 個の LED を数字の形に並べたものです。

小数点も合わせて、全部で 8 個の LED がパッケージに収められています。



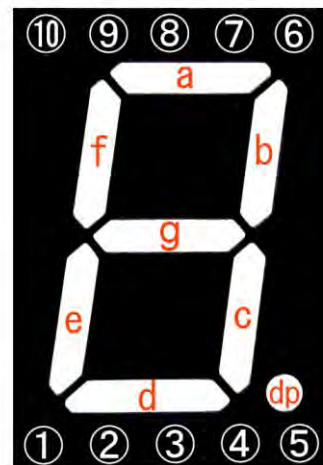
7 セグメント LED



裏側。10 本のピンがあります。

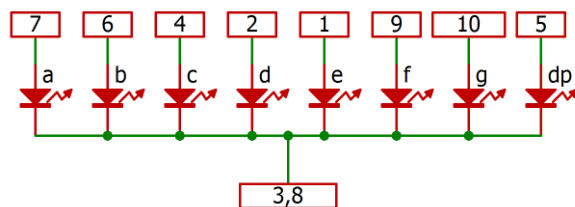
LED は、数字を表示する「a」～「g」の 7 個、小数点を表示する「dp」、合計 8 個あります。

裏側の 10 本のピンは、○数字のように番号が付いています。



内部では、8 個の LED がこの回路図のようにつながっています。

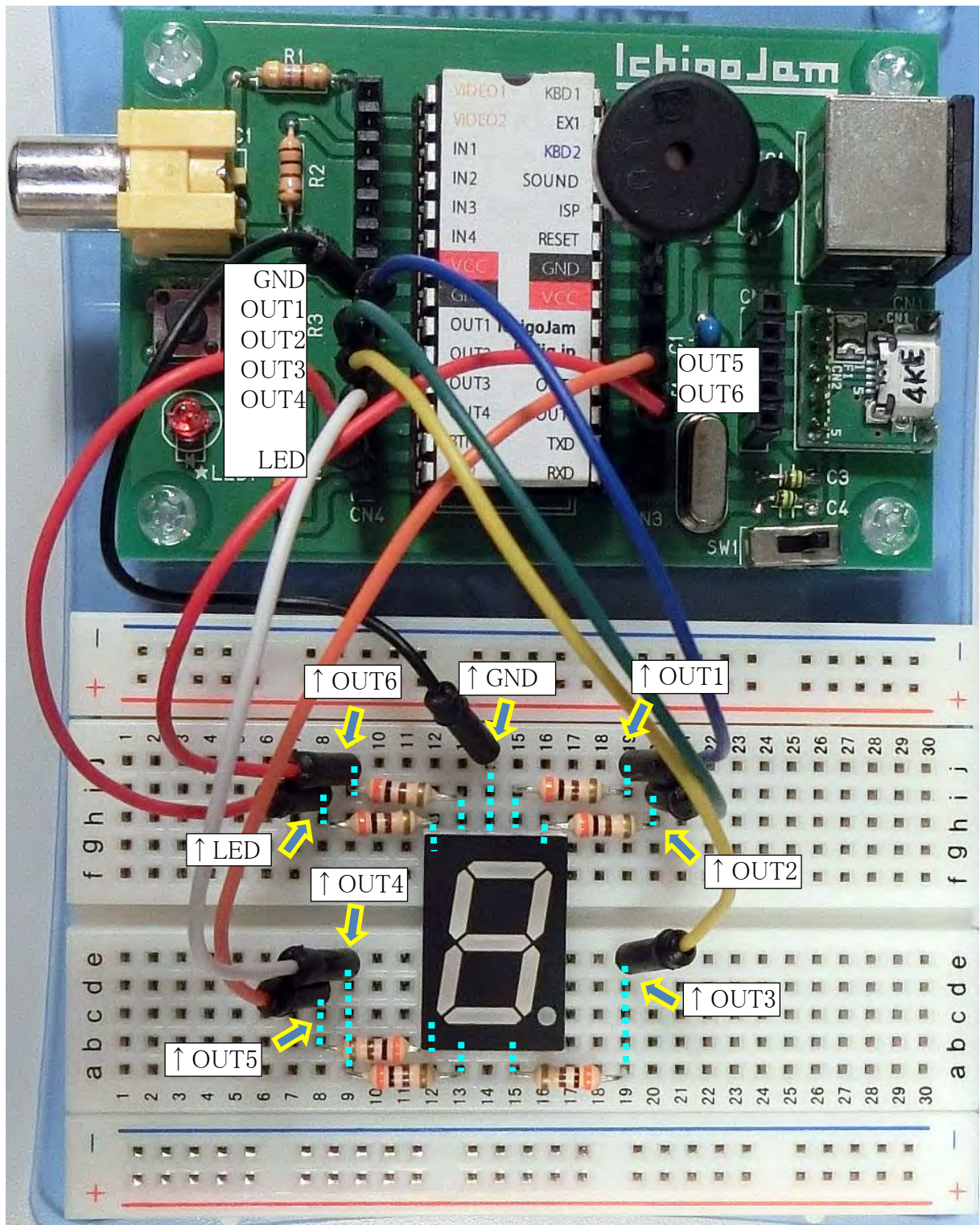
回路図の番号は、ピンの番号です。



今回は、上の 7 番ピン～5 番ピンを IchigoJam の出力端子につなぎ、下の 3 番ピンまたは 8 番ピンを GND(グラウンド)端子につなぎます。

そして、例えば 7 番ピンへ対して「1」を出力すると、「a」の LED に電流が流れて光ります。

ブレッドボードに 7 セグメント LED と抵抗を差して、IchigoJam と配線します。
7 セグメント LED から横に出すように抵抗を差して、ワイヤーをつなぐといいでしょう。

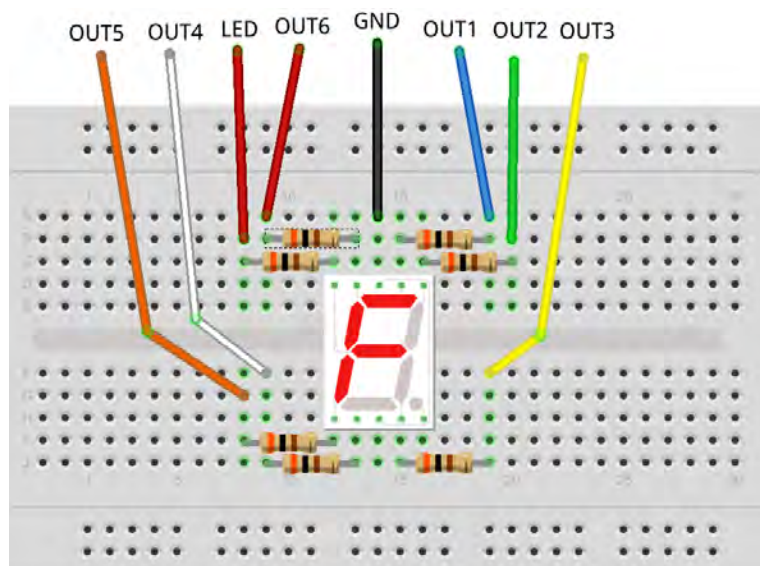


【IchigoJam と 7 セグメント LED のつながり方】

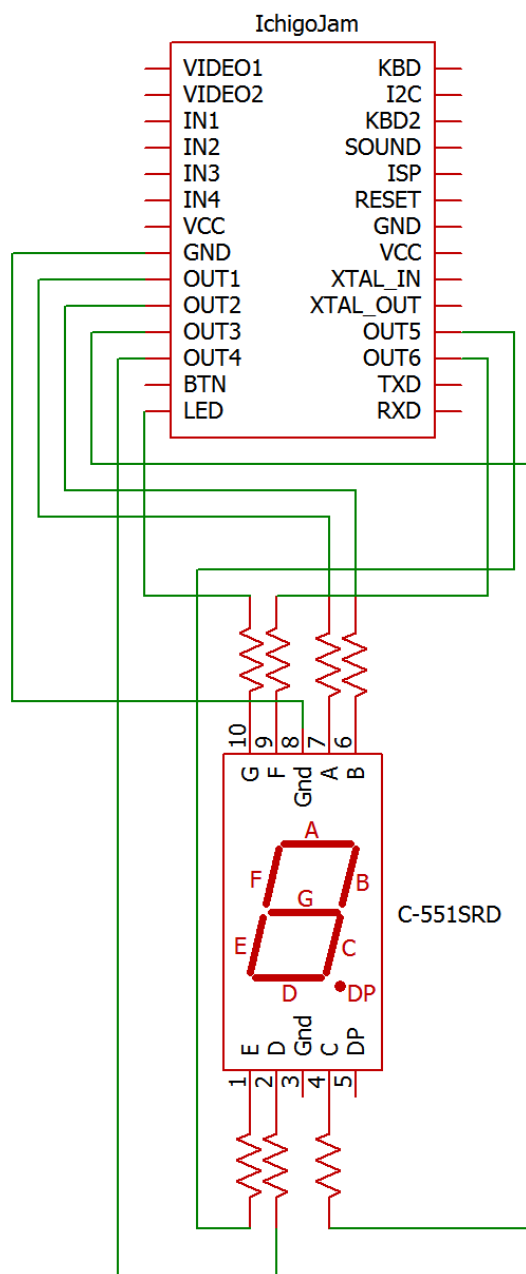
OUT1—a(7 ピン) OUT2—b(6 ピン) OUT3—c(4 ピン)
OUT4—d(2 ピン) OUT5—e(1 ピン) OUT6—f(9 ピン) LED—g(10 ピン)

LED は 7 個あるのですが、IchigoJam の出力は OUT1～OUT6 の 6 個しかないので、7 個目は LED 端子につなぎます。

7 セグメント LED のあたりの配線は、右図を見てください。



回路図に書くと、右のようになります。



回路ができたら、ちゃんとLED が光るか、ダイレクトモードで確認しましょう。

```
OUT 1,1
OUT 2,1
OUT 3,1
OUT 4,1
OUT 5,1
OUT 6,1
OUT 7,1
```

7 セグメント LED の a～g の LED が光ります。

もし光らない LED があったら、どこか配線をまちがえています。よく見直しましょう。

```
OUT 1,0
OUT 2,0
OUT 3,0
OUT 4,0
OUT 5,0
OUT 6,0
OUT 7,0
```

a～g の LED が消えます。

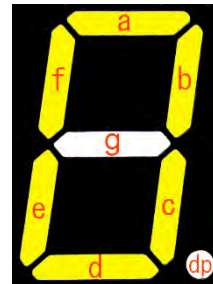
いろいろなパターンで LED を光らせてみましょう。

●7 セグメント LED で数字を表示する

回路ができたら、「NEW」でプログラムをクリアして、新しいプログラムを作ります。

まず、LED で「0」(ゼロ)を表示してみましょう。

「0」を表示するには、このようなパターンになればいいので、
a,b,c,d,e,f の 6 個の LED を光らせ、g の LED は消せばいいです。



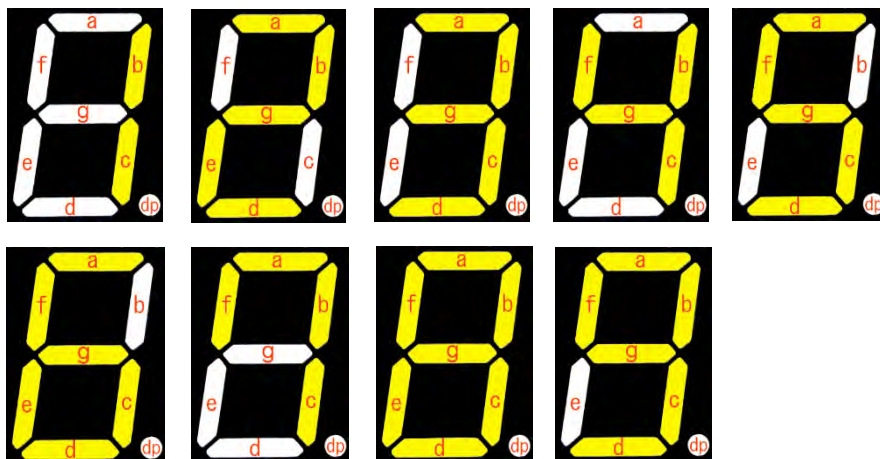
200	OUT	1, 1	ポート1～7に1と0を出力する
210	OUT	2, 1	
220	OUT	3, 1	
230	OUT	4, 1	
240	OUT	5, 1	
250	OUT	6, 1	
260	OUT	7, 0	

LED 端子は OUT7 ポートとしても指定できます。

※あとでプログラムを改造するために、行番号を 200 からにしています。

プログラムを実行してみましょう。7 セグメント LED に「0」が表示されます。

それぞれの「1」「0」の出力をいろいろ変えて、1～9 の数字を表示してみましょう。

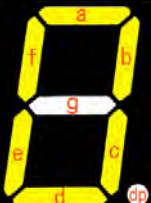
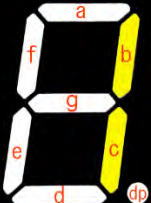
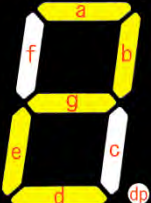
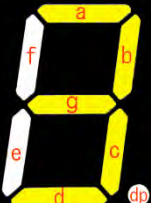
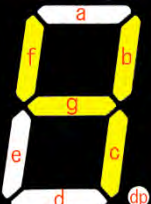
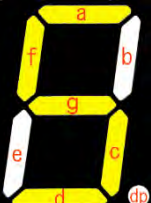
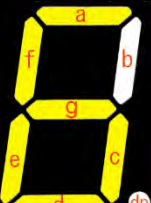


●自動的に 0～9 を表示

数字の表示を変えるのに、いちいちプログラムを書きかえるのは面倒です。

自動的に 0～9 の数字を表示するには、どうしたらいいでしょうか。

表示する数字 N と、各 LED の ON/OFF を、一覧表で考えてみます。

N	パターン	各 LED の ON/OFF (1=ON,0=OFF)						
		a	b	c	d	e	f	g
0		1	1	1	1	1	1	0
1		0	1	1	0	0	0	0
2		1	1	0	1	1	0	1
3		1	1	1	1	0	0	1
4		0	1	1	0	0	1	1
5		1	0	1	1	0	1	1
6		1	0	1	1	1	1	1

N	パターン	各 LED の ON/OFF (1=ON,0=OFF)						
		a	b	c	d	e	f	g
7		1	1	1	0	0	1	0
8		1	1	1	1	1	1	1
9		1	1	1	1	0	1	1

表示したい数字 N に合わせて、a～g の LED へ出力する値「1」または「0」を、変数 A～G にセットすることになります。

```

100 IF N=0 THEN A=1 : B=1 : C=1 : D=1 : E=1 : F=
1 : G=0
110 IF N=1 THEN A=0 : B=1 : C=1 : D=0 : E=0 : F=
0 : G=0
120 IF N=2 THEN A=1 : B=1 : C=0 : D=1 : E=1 : F=
0 : G=1
130 IF N=3 THEN A=1 : B=1 : C=1 : D=1 : E=0 : F=
0 : G=1
140 IF N=4 THEN A=0 : B=1 : C=1 : D=0 : E=0 : F=
1 : G=1
150 IF N=5 THEN A=1 : B=0 : C=1 : D=1 : E=0 : F=
1 : G=1
160 IF N=6 THEN A=1 : B=0 : C=1 : D=1 : E=1 : F=
1 : G=1
170 IF N=7 THEN A=1 : B=1 : C=1 : D=0 : E=0 : F=
1 : G=0
180 IF N=8 THEN A=1 : B=1 : C=1 : D=1 : E=1 : F=
1 : G=1
190 IF N=9 THEN A=1 : B=1 : C=1 : D=1 : E=0 : F=
1 : G=1

```

数字が 0 の時、A～G に「0」のパターンの 1 と 0 を出力する。以下同じ。

```

200 OUT 1, A
210 OUT 2, B
220 OUT 3, C
230 OUT 4, D
240 OUT 5, E
250 OUT 6, F
260 OUT 7, G

```

OUT 1～7 に A～G を出力する。

さらにこのプログラムをサブルーチンにして、メインプログラムから呼び出すようにします。
まずは N を 0 にして、サブルーチンを呼び出します。

```

50 N=0
60 GOSUB 100
90 END
100 IF N=0 THEN A=1:B=1:C=1:D=1:E=1:F=1:G=0
    (中略)
260 OUT 7, G
270 RETURN

```

N を 0 にする

100 行のサブルーチンを呼ぶ

プログラムを終了

メインプログラムへもどる

プログラムを実行してみましょう。7 セグメント LED に「0」が表示されます。
50 行目の N の値を 1～9 まで変えて、その数字が表示されるか試してみましょう。

N をいちいち変えるのは面倒なので、FOR～NEXT を使って、0～9 まで変化させます。

```

50 FOR N=0 TO 9
60 GOSUB 100
70 NEXT
90 END
    (後略)

```

N を 0～9 まで変化させる

くり返し

プログラムを実行してみましょう。IchigoJam の実行速度が速いので、あっという間に 9 まで行ってしまう。

WAIT で時間待ちを入れて、少し遅くしましょう。

```

50 FOR N=0 TO 9
60 GOSUB 100:WAIT 20
70 NEXT
90 END
    (後略)

```

60 分の 20 秒(=3 分の 1 秒)待つ

●ルーレットを作る

0～9 の数字を順番に表示するだけではおもしろくありません。

いろいろな数字をランダムに表示する、ルーレットを作ってみましょう。

ランダムな数字を表示するように、メインプログラムを改造します。

```
50 FOR I=0 TO 9
```

ループ変数をIに変更

```
55 N=RND(10)
```

Nを0～9のランダムな数字にする

```
60 GOSUB 100:WAIT 20
```

```
70 NEXT
```

```
90 END
```

(後略)

プログラムを実行してみましょう。

「0」～「9」までの数字がランダムに表示されます。

新しく RND (ランダム) 関数が出てきました。乱数(らんすう、ランダムな数)を出力します。

RND(10)

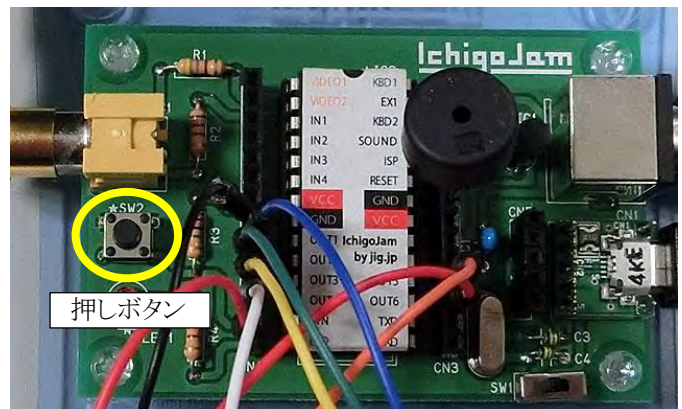
乱数の最大値

乱数の最大値	0～最大値-1 の乱数が出てきます。
--------	--------------------

「RND(10)」と指定すると、0～9 の乱数が出てきます。

本物のルーレットは、人間が手で回すと、だんだん遅くなって止まります。

ルーレットをスタートさせた後に、IchigoJam の押しボタンを押すと、だんだん遅くなって止まるようにしてみましょう。



```

10 ' *ROULETTE      タイトルコメント
20 N=RND(10)        WAIT なしで高速にランダムな数字を表示
30 GOSUB 100
40 IF BTN()=0 THEN GOTO 20  ボタンが押されていなかったらもどる
50 FOR I=0 TO 4      ループ変数 I の終値を 4 に変更 (5 回くりかえし)
55 N=RND(10)
60 GOSUB 100:WAIT 20
70 NEXT
90 END
(後略)

```

プログラムを実行してみましょう。

最初はルーレットが高速に回ります。押しボタンを押すと、ゆっくりになって止まります。

10 行目では、先頭に「'」(アポストロフィ、キーボードでは Shift キーを押しながら「7」を押す)を付けて、タイトルコメントを入れています。

「'」を付けると、その行はコメントとなり、何も実行されません。プログラムを後で見た時にわかりやすくするために、プログラムにいろいろコメントを入れるといいでしょう。

40 行目で、押しボタンが押されているかを判断するのに、BTN(ボタン)関数を使います。

BTN()	返り値	ボタンが押されている=1、押されていない=0
---------------	-----	------------------------

IF 命令で、BTN 関数の値が 0 だったら(=ボタンが押されていない)、20 行目に戻って高速ルーレットを続けます。ボタンが押されていたら次へ進み、ルーレットが遅くなります。

このままだとルーレットが止まると終わりなので、ボタンを押すとまた回るようにしましょう。

90 行の END を消して、BTN 関数を使った条件判断に変えます。

```

(前略)
50 FOR I=0 TO 4
55 N=RND(10)
60 GOSUB 100:WAIT 20
70 NEXT
90 IF BTN()=0 THEN GOTO 90 ELSE WAIT 30:
RUN

```

ボタンが押されていなかったら、この行の先頭へもどってくりかえし
押されていたら、時間待ちの後、RUN でプログラムを最初から実行

★プログラムをスロット 2 に保存しましょう。

SAVE 2

●効果音を出す

ルーレットが回るときに、音が出るようにしてみましょう。

```

10 ' *ROULETTE
20 N=RND(10)
30 BEEP 10,2:GOSUB 100  BEEP 音を出す
40 IF BTN()=0 THEN GOTO 20
50 FOR I=0 TO 4
55 N=RND(10)
60 BEEP 10,2:GOSUB 100:WAIT 20
70 NEXT
90 IF BTN()=0 THEN GOTO 90 ELSE WAIT 30:
  RUN
  (後略)
  
```

プログラムを実行してみましょう。ルーレットの数字が回ると一緒に音が鳴ります。

音を鳴らすには **BEEP** (ビーブ) 命令を使います。文法は以下のとおりです。

BEEP 30 , 30
 音の高さ 音の長さ

音の高さ	1～255 で指定する。省略可能。
音の長さ	60 分の 1 秒単位で指定する。「60」で 1 秒。省略可能。

BEEP 命令の数字を変えると、音の高さや長さが変わります。試してみましょう。

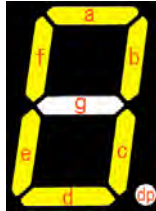
★プログラムをスロット 2 に保存しましょう。

SAVE 2

★できる人は

いろいろテクニックを使って、シンプルにしたルーレットのプログラムが、以下のとおりです。
20 行で、**配列変数**[0]～[9]に、7 桁の **2 進数**で、0～9 の数字フォントデータ(LED を ON/OFF するデータ。0=OFF、1=ON)を設定しています。

``0111111`
 g f e d c b a



※逆順なので注意

30 行・70 行・120 行で、OUT 命令を使って LED を光らせています。
20 行で設定した配列変数の値を出力することで、OUT ポート 1～7(LED ポートは OUT7 ポートとしても使えます)を制御しています。

プログラムを打ち込んで、動かしてみてください。
(20 行がとても長いのですが、途中で改行しないで連続で入力してください)
これまでのプログラムより、かなり高速に数字が表示されます。

```

10 `*ROULETTE
20 LET [0], `0111111, `0000110, `101101
1, `1001111, `1100110, `1101101, `111110
1, `0100111, `1111111, `1101111
30 OUT [0]
40 `@START
50 IF BTN()=0 THEN GOTO 40
60 FOR I=1 TO 10
70 OUT [RND(10)]
80 BEEP 20, 2
90 WAIT 6
100 NEXT
110 FOR I=1 TO 5
120 OUT [RND(10)]
130 BEEP 20, 2
140 WAIT 12
150 NEXT
160 GOTO 40
  
```

「`」は、Shift キーを押しながら「@」キーを押すと入力できます。

最初に LED に「0」を表示

ボタンが押されるまで待つ

10 回くりかえし

乱数で 0～9 の数字を表示

音を鳴らす

1/10 秒待つ

5 回くりかえし

乱数で 0～9 の数字を表示

音を鳴らす

1/5 秒待つ

ボタン入力へ戻る

配列変数に 2 進数で「0」～「9」の数字フォントデータを設定