

プログラミングをしてみよう

# Micro bitにおけるプログラミング

Micro bit では、プログラミングをやったことがない人でもブロックを組み合わせることで簡単にプログラムを作成して学ぶことができます。

プログラミング言語はむずかしい言葉ですが、実際に操作してみると楽しみながらプログラミングの基本を学習していきましょう

# Micro bitに接続する

インターネットに接続して、  
Micro bitのサイトに行こう

ここをクリック

ここに入力して検索

10:00  
2021/06/04

# Micro bitに接続する

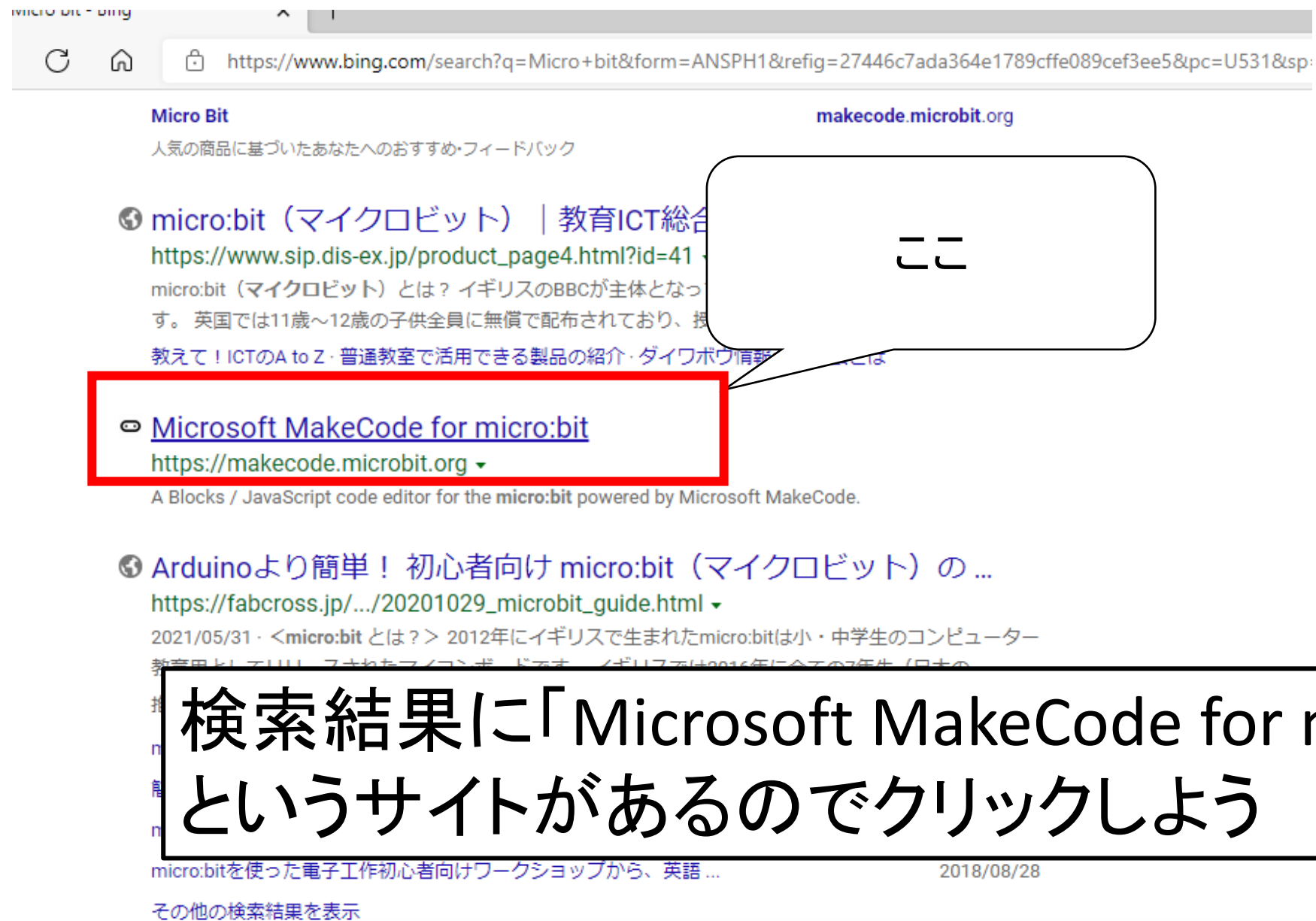
ここに入力



入力したら虫眼鏡をクリック  
かエンターキーを押そう

ブラウザが開いたら、上の白いバーに  
「Micro bit」と入力しよう

# Micro bitに接続する



The image shows a screenshot of a Bing search results page for the query "Micro bit". The browser's address bar shows the URL: <https://www.bing.com/search?q=Micro+bit&form=ANSPH1&refid=27446c7ada364e1789cffe089cef3ee5&pc=U531&sp:>

The search results include:

- A link to "makecode.microbit.org" with the text "人気の商品に基づいたあなたへのおすすめ・フィードバック".
- A search result for "micro:bit (マイクロビット) | 教育ICT総合" with a URL [https://www.sip.dis-ex.jp/product\\_page4.html?id=41](https://www.sip.dis-ex.jp/product_page4.html?id=41). A speech bubble with the text "ここ" (here) points to this result.
- A search result for "Microsoft MakeCode for micro:bit" with a URL <https://makecode.microbit.org>. This result is highlighted with a red rectangular box.
- A search result for "Arduinoより簡単！ 初心者向け micro:bit (マイクロビット) の ..." with a URL [https://fabcross.jp/.../20201029\\_microbit\\_guide.html](https://fabcross.jp/.../20201029_microbit_guide.html).

At the bottom of the page, there is a large black-bordered box containing the text: "検索結果に「Microsoft MakeCode for micro:bit」というサイトがあるのでクリックしよう".

Other visible text includes "micro:bitを使った電子工作初心者向けワークショップから、英語 ..." and "2018/08/28".



# Micro bitに接続する

micro:bit

ホーム

Microsoft

ここ

マイプロジェクト すべて表示

読み込む

+

新しいプロジェクト

このサイト上でプログラミングが出来る  
「新しいプロジェクト」をクリックしてみよう

チュートリアル

初めて? Start Here!

点滅するハート

Name Tag

Smiley Buttons

Dice

Love Meter

Micro (

# 基本操作を学ぼう



The screenshot shows the bit.io interface. At the top, there's a navigation bar with 'ホーム' (Home) and a settings gear. Below it is a banner with a 'show leds' button. A modal dialog titled 'プロジェクトを作成する' (Create Project) is open in the center. It contains the instruction 'プロジェクトに名前をつけてください。' (Please name your project.) and an empty text input field. A green '作成' (Create) button with a checkmark is at the bottom right of the dialog. A speech bubble with the characters 'ここ' (here) points to the input field. On the left, there's a purple button with a plus sign and the text '新しいプロジェクト' (New Project). At the bottom, there are several project thumbnails: 'おはようハート', 'Name Tag', 'Smiley Buttons', 'Dice', and 'Love Meter'.

プロジェクトを作成する

プロジェクトに名前をつけてください。

作成 ✓

ここ

新しいプロジェクト

おはようハート Name Tag Smiley Buttons Dice Love Meter

1つのプログラムを「プロジェクト」と呼ぶ  
「テスト」と名前を付けて作成ボタンを押そう

# 基本操作を学ぼう

The screenshot shows the micro:bit online programming environment. At the top, there is a navigation bar with 'micro:bit', 'ホーム', '共有', 'ブロック', and 'JavaScript'. Below this is a search bar and a list of block categories: 基本, 入力, 音楽, LED, 無線, ループ, 論理, 変数, 計算, and 高度なブロック. To the right is a workspace with two blue blocks labeled '最初だけ' and 'ずっと'. A red box highlights the block palette and workspace. A white callout box points to the workspace, and another points to the block palette. A third callout box points to the workspace.

micro:bit ホーム 共有 ブロック JavaScript

検索...

- 基本
- 入力
- 音楽
- LED
- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック

最初だけ

ずっと

実際にプログラムがどう動くか見れる

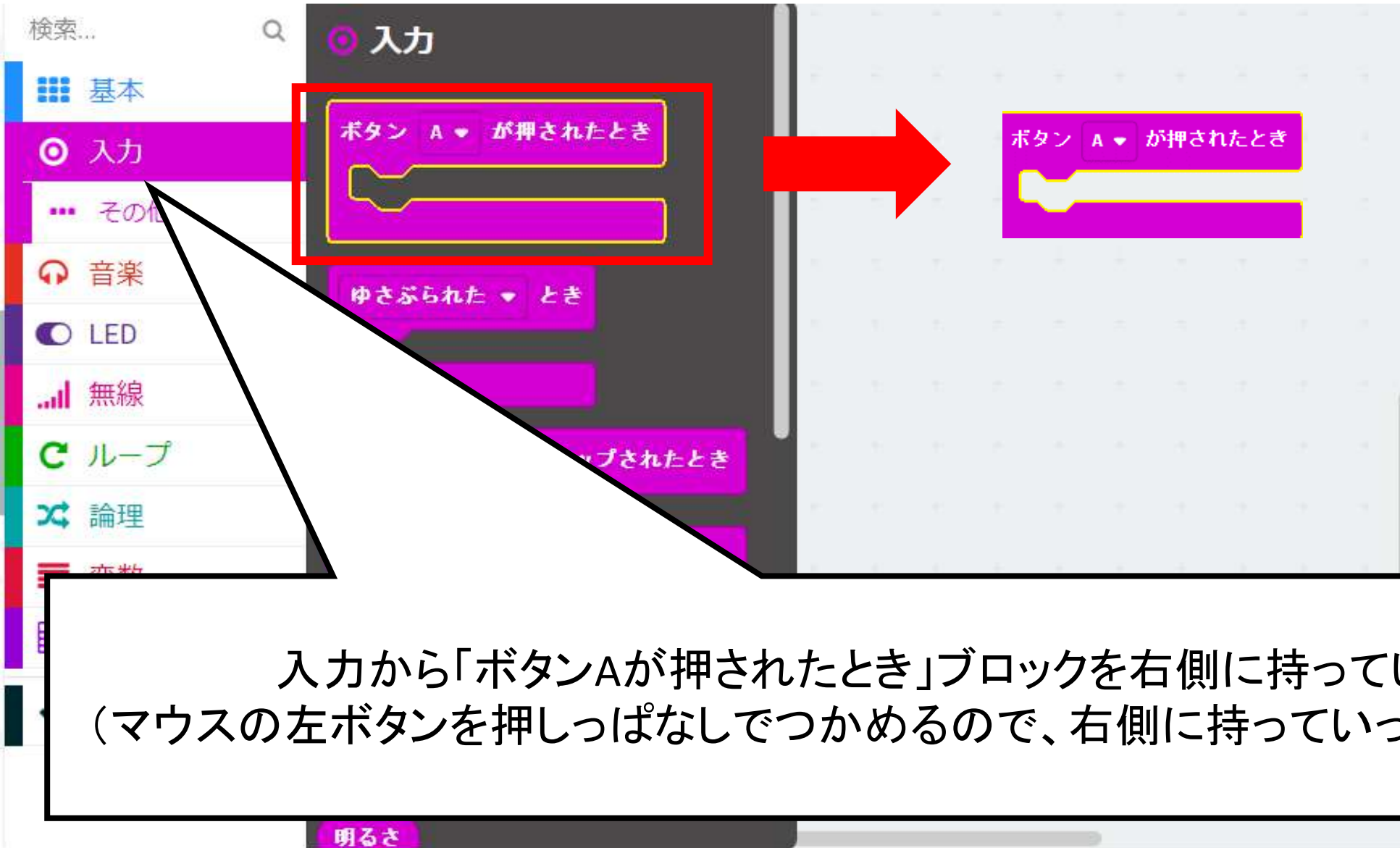
このエリアにブロックを配置  
⇒組み合わせでプログラムを作成する

この中からプログラムブロックを取り出して

ダウンロード



# 基本操作を学ぼう



The image shows a software interface with a sidebar on the left containing various categories: 基本 (Basic), 入力 (Input), その他 (Others), 音楽 (Music), LED, 無線 (Wireless), ループ (Loop), 論理 (Logic), and 変数 (Variables). The '入力' category is selected. In the main workspace, a block titled 'ボタン A が押されたとき' (When button A is pressed) is highlighted with a red box. A red arrow points from this block to a larger version of the same block on the right side of the workspace. A black callout box with a white background and a black border is positioned at the bottom, containing Japanese text. At the bottom of the interface, there is a slider labeled '明るさ' (Brightness).

検索...

基本

入力

その他

音楽

LED

無線

ループ

論理

変数

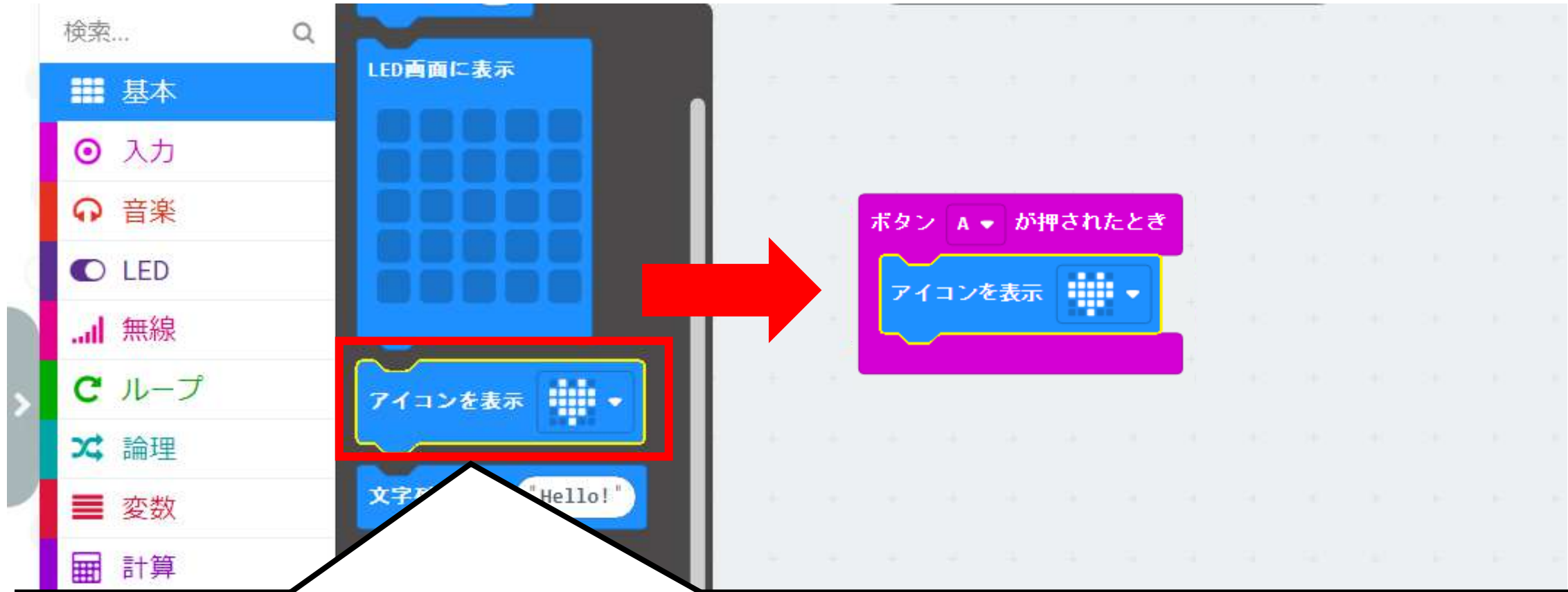
ボタン A が押されたとき

ボタン A が押されたとき

明るさ

入力から「ボタンAが押されたとき」ブロックを右側に持っていこう。  
(マウスの左ボタンを押しっぱなしでつかめるので、右側に持っていったら指を放そう)

# 基本操作を学ぼう

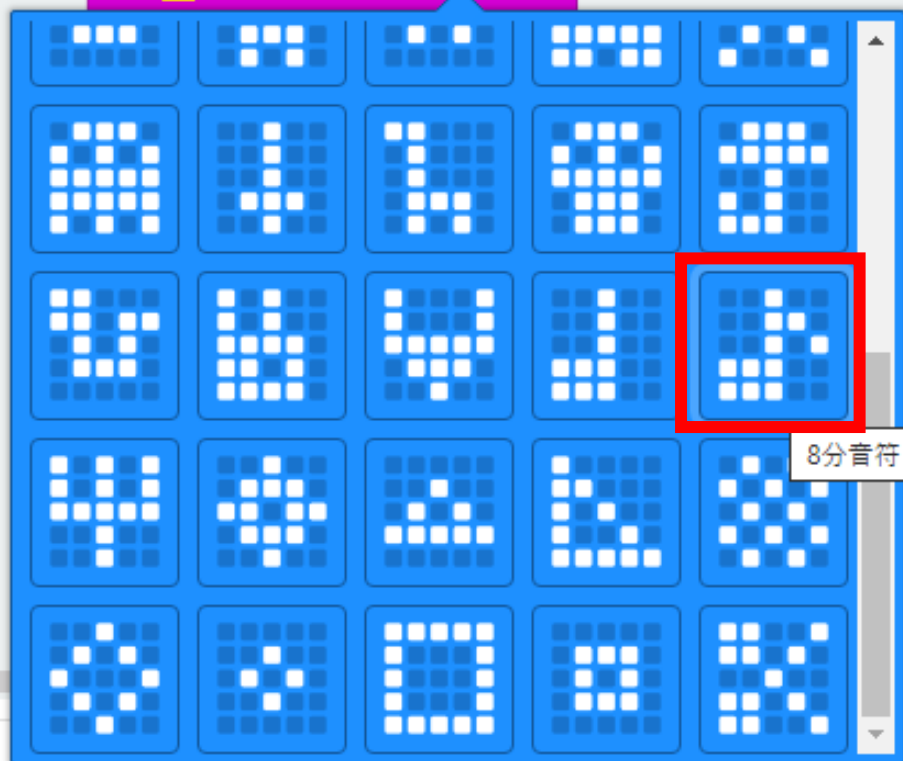


基本から「アイコンを表示」ブロックを右側に持っていこう。  
「ボタンAが押されたとき」ブロックと組み合わせよう。

# 基本操作を学ぼう

ハートマークのところをクリックして音符マークに変更しよう

ボタン A ▼ が押されたとき  
アイコンを表示



8分音符

# 基本操作を学ぼう



左側にあるMicro bitのAボタンをクリックしてみよう  
これで、電源を入れてAボタンを押したらLEDが光る仕組みが出来たので、  
次はBボタンを押して表示を消す仕組みを作ってみよう

# 基本操作を学ぼう

The image shows a block-based programming environment. On the left is a sidebar with a search bar and a list of categories: 基本 (Basic), 入力 (Input), その他 (Others), 音楽 (Music), LED, 無線 (Wireless), ループ (Loop), 論理 (Logic), 変数 (Variables), 計算 (Calculation), and 高度なブロック (Advanced Blocks). The main workspace is titled '入力' (Input) and contains several blocks. A red box highlights a block labeled 'ボタン A ▼ が押されたとき' (When button A is pressed). A white arrow points from this block to a zoomed-in view on the right. In the zoomed view, a red box highlights the dropdown menu for the button, which shows three options: 'A' (with a checkmark), 'B', and 'A+B'. The text 'ボタン A ▼ が押されたとき' is also highlighted in yellow in the background.

もう一度「ボタンAが押されたとき」ブロックを右側に持ってきて、「A▼」を押して「ボタンBが押されたとき」に変更しよう



# 基本操作を学ぼう

検索...

- 基本
- 入力
- 音楽
- LED
- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック

基本から「表示を消す」ブロックを持ってきて  
「ボタンBが押されたとき」ブロックと組み合わせよう

The image shows a block-based programming environment. On the left is a palette with various block categories. A 'Clear display' block (blue with a yellow border) is highlighted with a red box. A red arrow points from this block to the workspace. In the workspace, a 'When button B is pressed' block (purple) is already present, and the 'Clear display' block is being placed into its 'do this' slot. Below the workspace, a text box says 'すべてのLEDをオフにします。' (Turn off all LEDs.)

LED画面

ボタン B が押されたとき

表示を消す

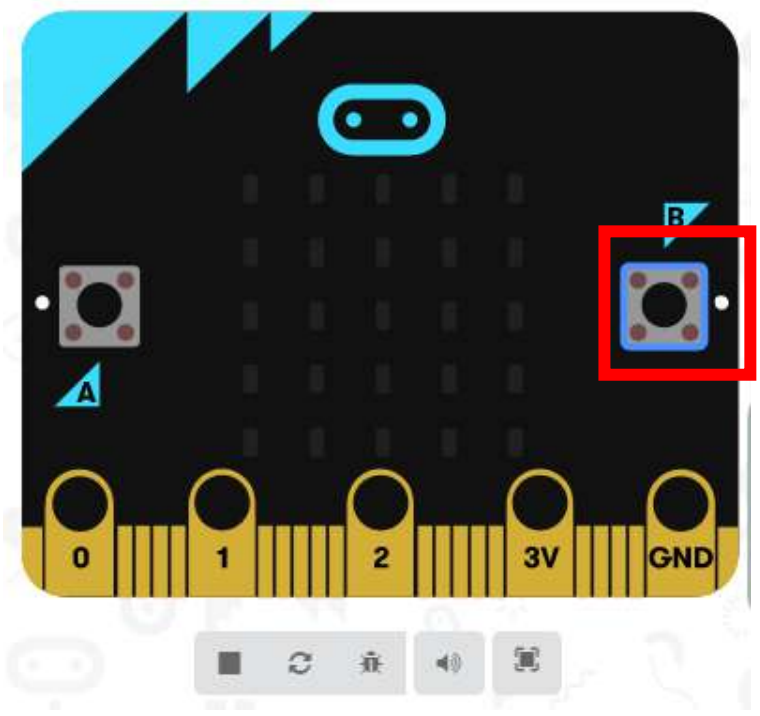
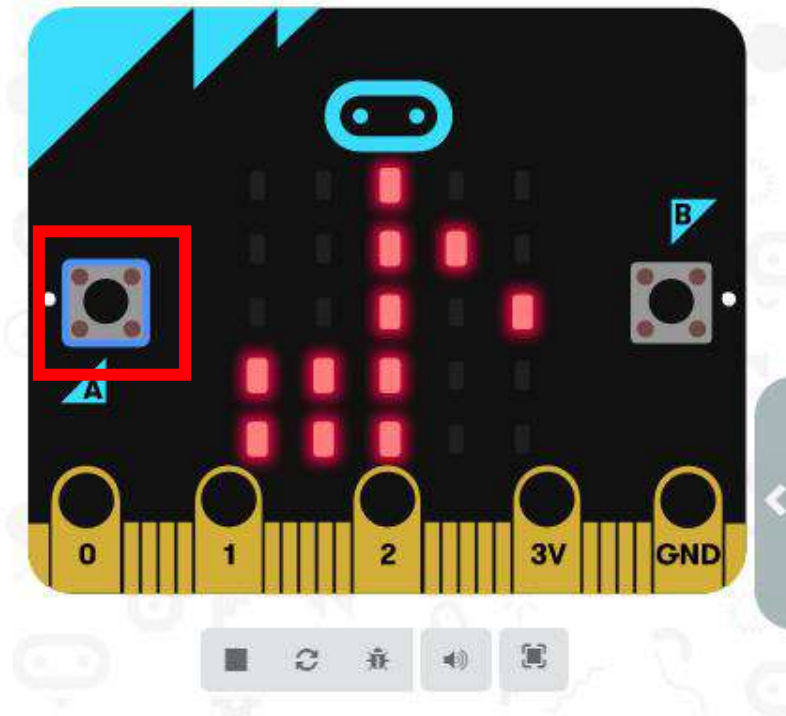
アイ...

文字を表示 "Hello!"

表示を消す

すべてのLEDをオフにします。

# 基本操作を学ぼう



Aボタンを押すとLEDが光ってBボタンで消える仕組みが完成

# 基本操作を学ぼう

検索...

- 基本
- 入力
- 音楽
- LED
- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック

音楽

メロディ

メロディを鳴らす [Musical notes icon] テンポ 120 (bpm)

トーン

音を鳴らす 高さ (Hz) 真ん中のド 長さ 1 拍

音を鳴らす 高さ (Hz) 真ん中のド

休符 (ミリ秒) 1 拍

真ん中のド

ボタン A が押されたとき

アイコンを表示 [Icon icon]

メロディを鳴らす [Musical notes icon] テンポ 120 (bpm)

ボタン B が押されたとき

表示を消す

今度は音楽から「メロディを鳴らす」ブロックをもってきて「アイコンを表示」ブロックの下に組み合わせよう


# 基本操作を学ぼう



音符マークをクリックして、ギャラリーから好きなメロディを選んでみよう

右側のテンポは数字が大きければ早くなる

# 基本操作を学ぼう



The image shows a Scratch script on a light gray grid background. It consists of two event blocks and two action blocks. The first event block is 'ボタン A が押されたとき' (When button A is pressed), which triggers a 'アイコンを表示' (Show icon) block with a 3x3 grid icon and a 'メロディを鳴らす' (Play melody) block with a tempo of 250 bpm. The second event block is 'ボタン B が押されたとき' (When button B is pressed), which triggers a '表示を消す' (Hide icon) block.

ボタン A が押されたとき

アイコンを表示

メロディを鳴らす テンポ 250 (bpm)

ボタン B が押されたとき

表示を消す

Micro bitの本体にこのプログラムを入れてみよう！



Micro bitで温度を測ろう

Micro bitの画面から、新しいプロジェクトをクリックしてプログラムを作成する

マイプロジェクト [すべて表示する](#)

空の新しいプロジェクトを作成します。

A purple rectangular button with a white circle containing a plus sign and the text "新しいプロジェクト" (New Project) below it. The button is highlighted with a red border.

新しいプロジェクト

探知機生徒用

A small icon of a person.

16時間前

探知機

A small icon of a person.

21時間前

チュートリアル

A red heart with black dots and a hand pointing to it.

初めて? Start Here!

点減するハート

A hand holding a card with the word "MICRO" written on it.

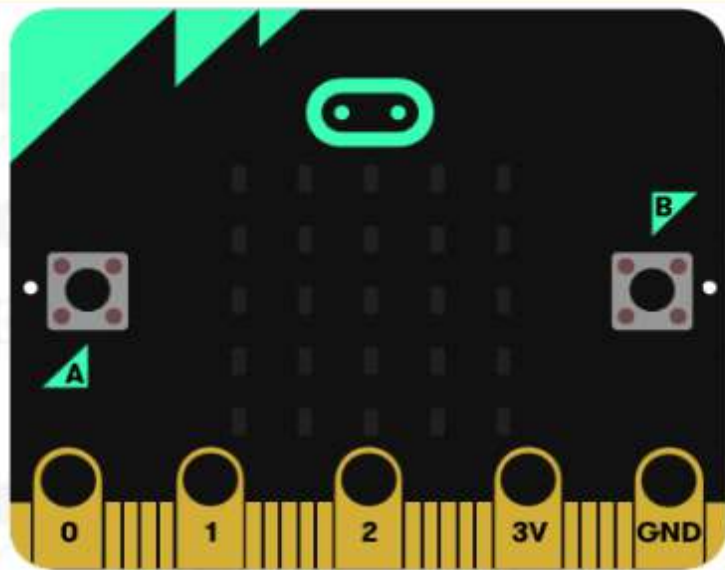
Name Tag

A yellow smiley face with a hand pointing to it.

Smiley Buttons

A white die with a face.

Dice



検索...



基本

入力

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

最初だけ

ずっと

ダウンロード

温度計

ブロックを配置してプログラムを作る画面が出てくる

ここからボタンを押したときに温度を測る機能を用意する

検索...



## 入力

基本

入力

その他

音楽

LED

無線

ループ

論理

ボタン A が押されたとき

ゆさぶられたとき

端子 P0 がタッチされたとき

ボタン (A、B、C、D、E、F、G、H、I、J、K、L、M、N、O、P、Q、R、S、T、U、V、W、X、Y、Z) の両方が押されてはなされたときに実行されます。

ボタン A が押されたとき

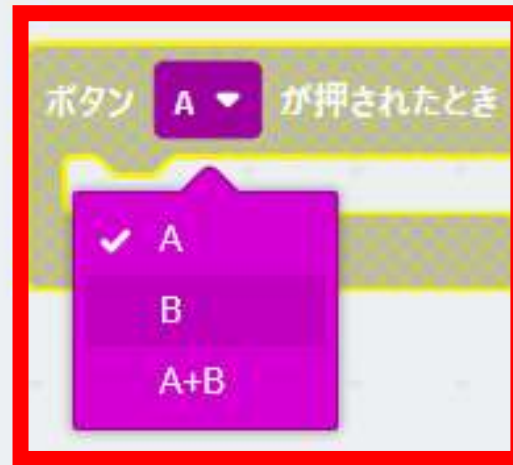
ボタン A が押されたとき

「入力」をクリックして、「ボタンAが押されたとき」ブロックを2つ右のエリアに持ってくる

端子 P0 がタッチされている

明るさ

方角 (°)



片方の「ボタンAが押されたとき」ブロックの▼をクリックして、Bにチェックを付ける



検索...



## 基本

基本

入力

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

数を表示 0

画面上の数字をスクロールします。数が1桁で、画面上に収まる場合、スクロールしません。

ボタン A が押されたとき

ボタン B が押されたとき

アイコンを表示

文字列を表示 "Hello"

表示を消す

ずっと

「基本」をクリックして  
「数を表示」ブロックをAに  
「表示を消す」ブロックをBに合体させる

検索...



基本

入力

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

ずっと

ボタン A が押されたとき

数を表示

0

ボタン B が押されたとき

表示を消す

これでボタンAを押したときに  
0を表示するようになり、ボタンBを押したときは  
0の表示が消えるプログラムが出来た。

温度計



検索...

基本

入力

その他

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

ボタン A が押されている

加速度 X

端子 P0 がタッチされている

明るさ

方角 (°)

温度 (°C)

ゆさぶられた 動き

マイクロビット (v2)

大音量 音がしたとき

ロゴが 押された とき

ずっと

ボタン A が押されたとき

0

ボタン B が押されたとき

表示を消す



「入力」をクリックして、  
「温度」ブロックを数を表示の0のところに  
持っていき合体させる



- 基本
- 入力
- その他
- 音楽
- LED
- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック

ずっと

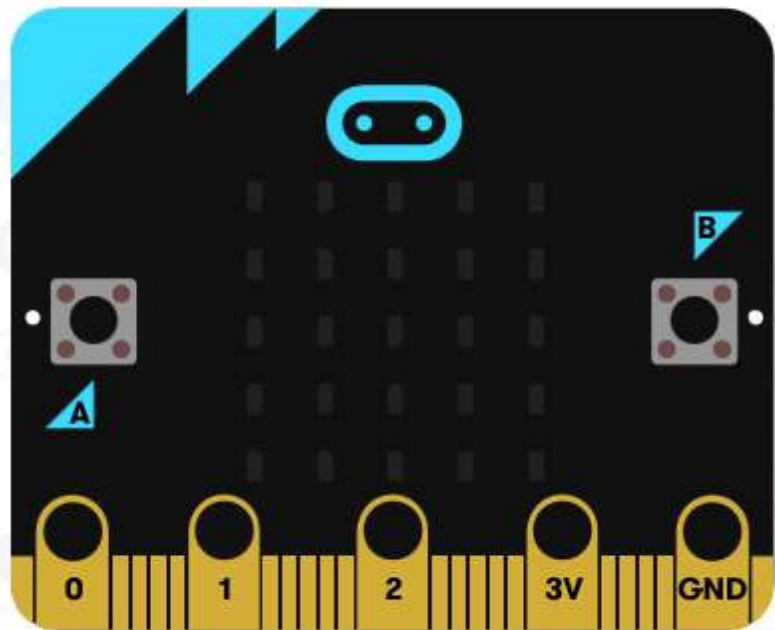
ボタン A が押されたとき

数を表示 温度 (°C)

ボタン B が押されたとき

表示を消す

これでボタンAを押すとLEDがその場所の温度を表示してくれる



検索...



基本

入力

その他

音楽

LED

無線

ループ

論理

ずっと

ボタン A が押されたとき

数を表示 温度 (°C)

ボタン B が押されたとき

表示を消す

Micro bitをパソコンとつないでダウンロード  
実際にプログラムを動かしてみよう！

プログラムをmicro:bitに書き込む。

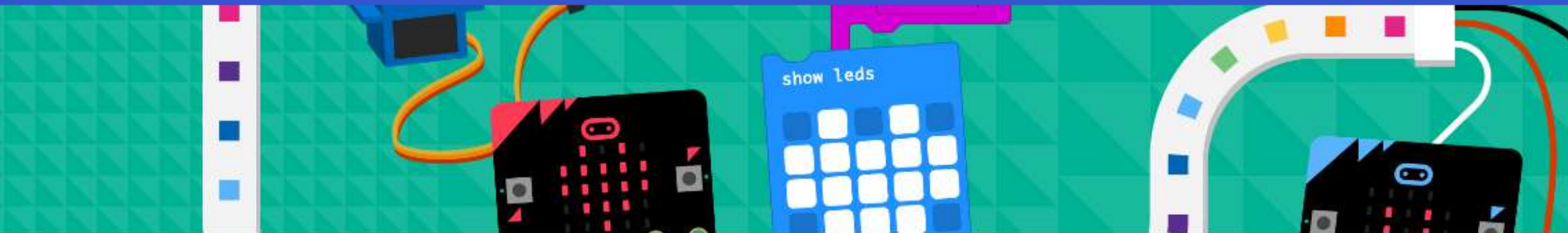
ダウンロード

温度計



じゃんけんしてみよう





マイプロジェクト

新しいプロジェクト

A purple rectangular button with a white plus sign icon and the text '新しいプロジェクト' (New Project) in white. The button is highlighted with a red border.

新しいプロジェクトをクリックして  
プログラムを作成していく

5分前

17時間前

22時間前

チュートリアル

初めて? Start Here!

点減するハート

Name Tag

Smiley Buttons

Dice

Love Meter

検索...

基本

入力

その他

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

## 入力

ボタン A が押されたとき

ゆさぶられたとき

端子 P0 がタッチされたとき

ボタン A が押されている

加速度 X

端子 P0 がタッチされている

明るさ

方角 (°)



新しいプロジェクトを起動したら、  
入力をクリックして「ゆさぶられたとき」  
ブロックを持ってくる

検索...



## 論理

基本

入力

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

### 条件判断

もし 真 ▼ なら



もし 真 ▼ なら

でなければ (−)



### くらべる

0 = 0

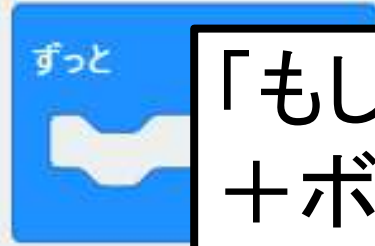
0 < 0

"" = ""

ずっと

ゆさぶられた ▼ とき

「論理」をクリックして、  
「もし真なら～でなければ」ブロックを  
「ゆさぶられたとき」ブロックと合体させる



「もし真なら～でなければ」ブロックの  
+ボタンを押して画像のように条件を3つにする



最初の値が真の場合には、最初のかたまりのブロックを動かします。そうでない場合で、二番目の値が真なら、二番目のかたまりのブロックを動かします。どちらの値も真でない場合には、最後のかたまりのブロックを動かします。





ここから条件の分岐とLEDの表示を設定していく

Scratch script for a button click event:

- Event: **ゆさぶられた** (Clicked) **とき** (When)
- Condition 1: **もし** (If) **0** **=** **0** **なら** (then)
- Condition 2: **でなければもし** (If not) **0** **=** **0** **なら** (then) **-**
- Condition 3: **でなければ** (If not) **-**
- Condition 4: **+**





「計算」をクリックして、  
「ランダムな数字を選択」ブロックを  
もし～ならの左側の0に合体させる

検索...

- 基本
- 入力
- 音楽
- LED
- 無線
- ループ
- 論理
- 変数
- 計算**
- 高度なブロック

0 と 0 のうち 小さい方 ▾

0 と 0 のうち 大きい方 ▾

0 の絶対値

平方根 ▾ 0

小数点以下四捨五入 ▾ 0

ランダムな数字を選択: 0 から 10 まで

0 を 0 以上 0 以下の範囲に制限

数値をマップする 0 元の下限 0 元の上限 1023 結果の下限 0 結果の上限 4

ランダムに真か偽に決める

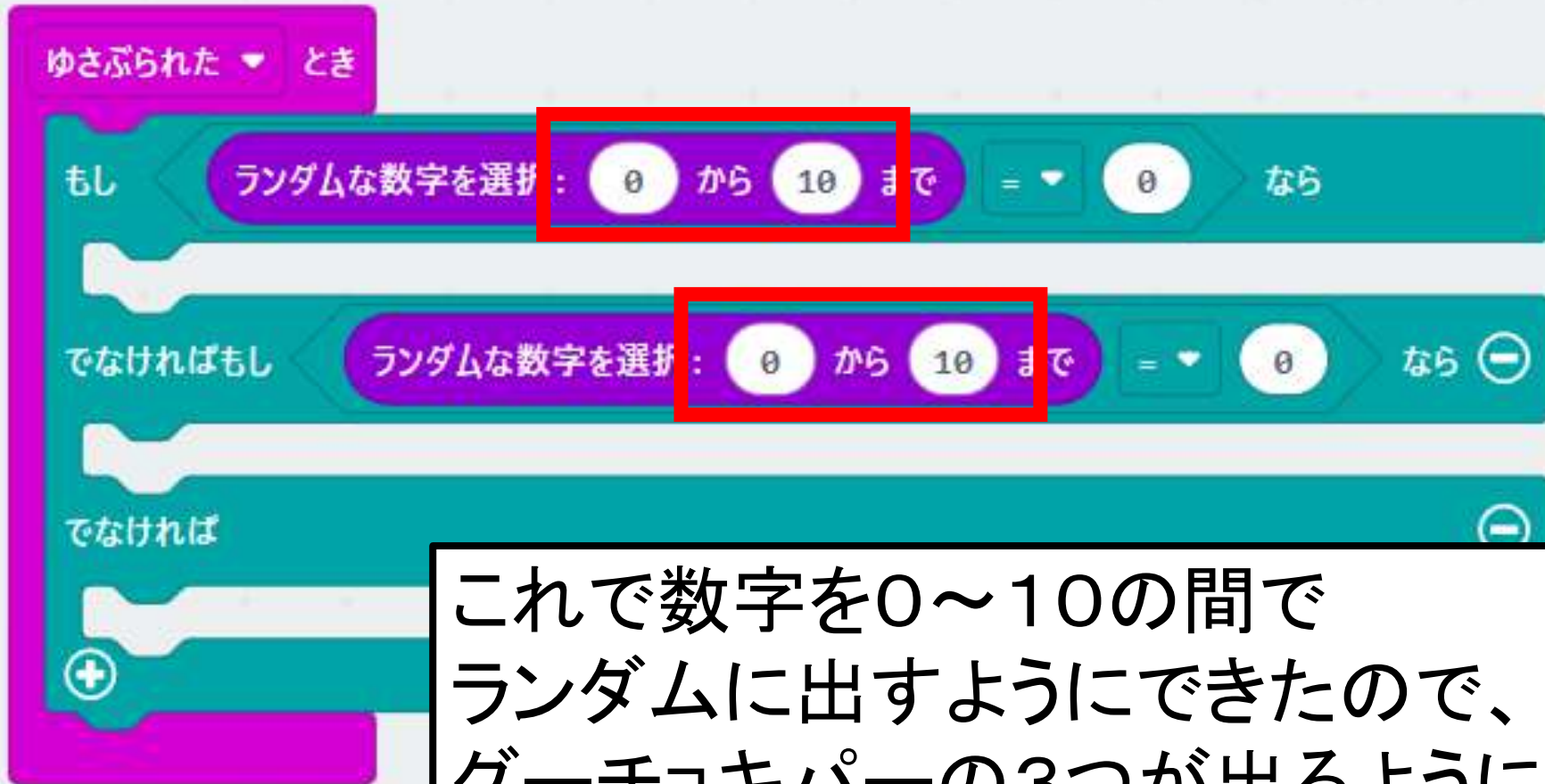
ゆさぶられた ▾ とき

もし 0 = ▾ 0 なら

でなければもし 0 = ▾ 0 なら -

でなければ -

+ -



これで数字を0～10の間でランダムに出すようにできたので、グーチョキパーの3つが出るように数字を変更する

ゆさぶられた ▾ とき

もし

ランダムな数字を選択:

0

から

2

まで

= ▾

0

なら

でなければもし

ランダムな数字を選択

0

から

1

まで

= ▾

0

なら



でなければ



条件を整理すると...

- ①最初にランダムに0・1・2から選ぶようにする
- ②0が選ばれたら結果を出し、そうでないときは0・1のどちらかを選ぶ
- ③2回目の条件で1が選ばれたら対応する結果を表示する

- 基本
- 入力
- 音楽
- LED
- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック

数を表示 0

LED画面に表示

アイコンを表示

文字列を表示 "Hello!"

表示を消す

ずっと

ゆさぶられた とき

もし ランダムな数字を選択: 0 から 2 まで = 0 なら

でなければもし ランダムな数字を選択: 0 から 1 まで = 0 なら

LED画面に画像を表示します。

「基本」をクリックして、「LED画面に表示」ブロックを3つの条件と合体させる

ゆさぶられた ▾ とき

もし

ランダムな数字を選択: 0 から 2 まで = ▾ 0 なら

LED画面に表示



でなければもし

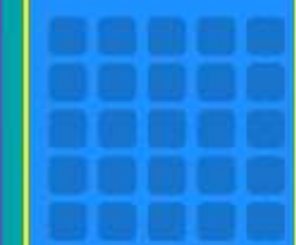
ランダムな数字を選択: 0 から 1 まで = ▾ 0 なら

LED画面に表示



でなければ

LED画面に表示



このような形に合体させたら、  
あとはゲーチャヨキパーを割り当てる



ゆさぶられた ▾ とき

もし

ランダムな数字を選択: 0 から 2 まで = ▾ 0 なら

LED画面に表示



でなければもし

ランダムな数字を選択: 0 から 1 まで = ▾ 0 なら ↻

LED画面に表示



でなければ

LED画面に表示



濃い青色のマスをクリックして  
白色になったところが実際に光るので  
画像のように設定してみよう



検索...



基本

入力

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

もし ランダムな数字を選択: 0 から 2 まで = 0 なら

LED画面に表示



でなければもし ランダムな数字を選択: 0 から 1 まで = 0

LED画面に表示



でなければ

LED画面に表示



ダウンロード

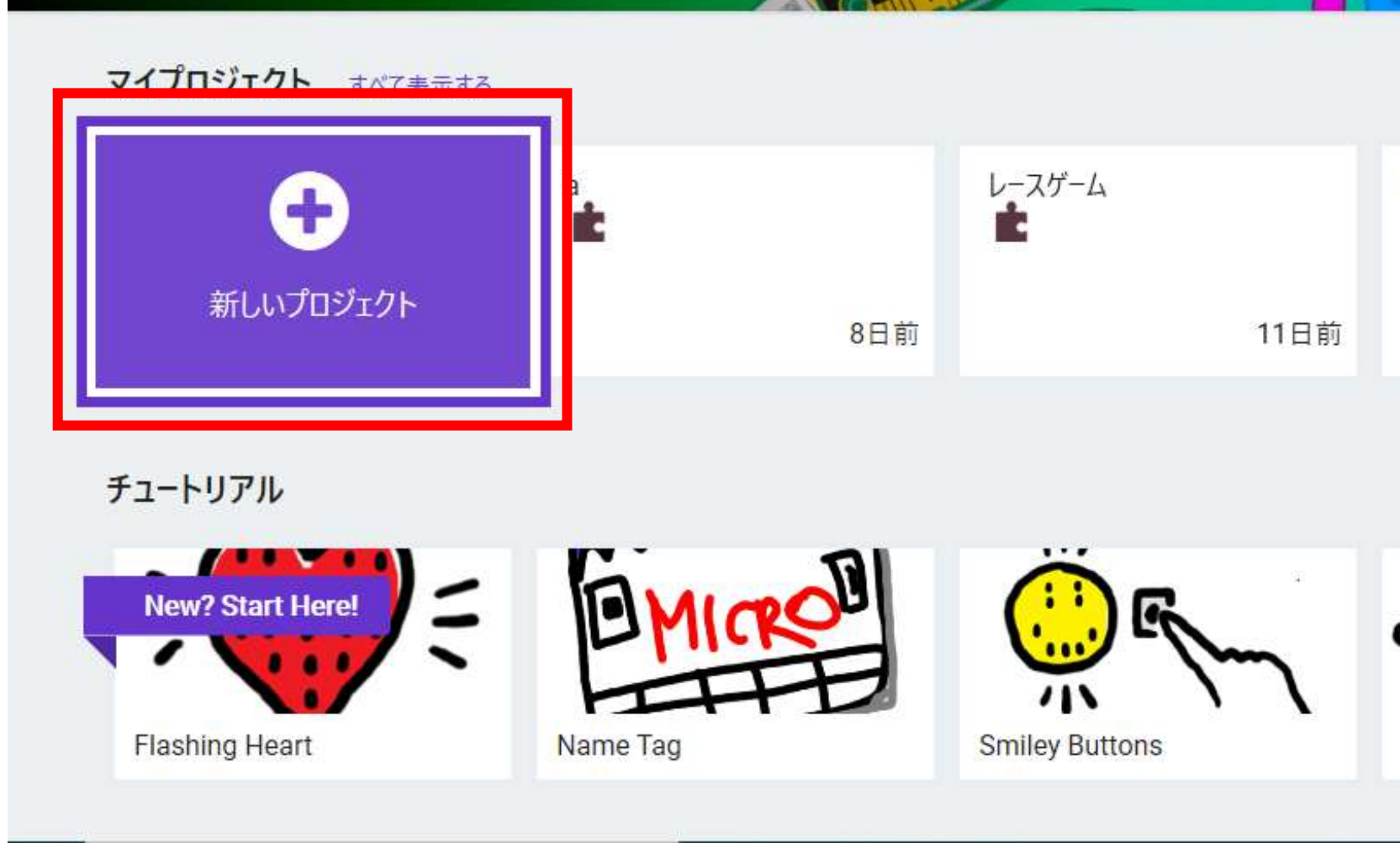
じゃんけん



PCとMicro bitをつないで  
プログラムを入れてみよう!

明るさセンサーをつくらう

まずは新しいプロジェクトをクリックしよう  
今回は、部屋の明るさを数字で出すプログラムを組むよ



# プロジェクトの名前を決めよう

プロジェクトを作成する 🤖



プロジェクトに名前をつけてください。

明るさセンサー



▶ コードのオプション

作成



29日前

検索...



基本

入力

その他

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

入力

ボタン A が押されたとき

ゆさぶられたとき

端子 P0 が短くタップされたとき

ポ

加速

端子 P0 がタッチされている

明るさ

右角 (°)

ボタン A が押されたとき

プロジェクトが開いたら、「入力」から「ボタンAが押されたとき」を右側に持っていこう

検索...

基本

数を表示 0

LED画面に表示

アイコンを表示

ボタン A が押されたとき

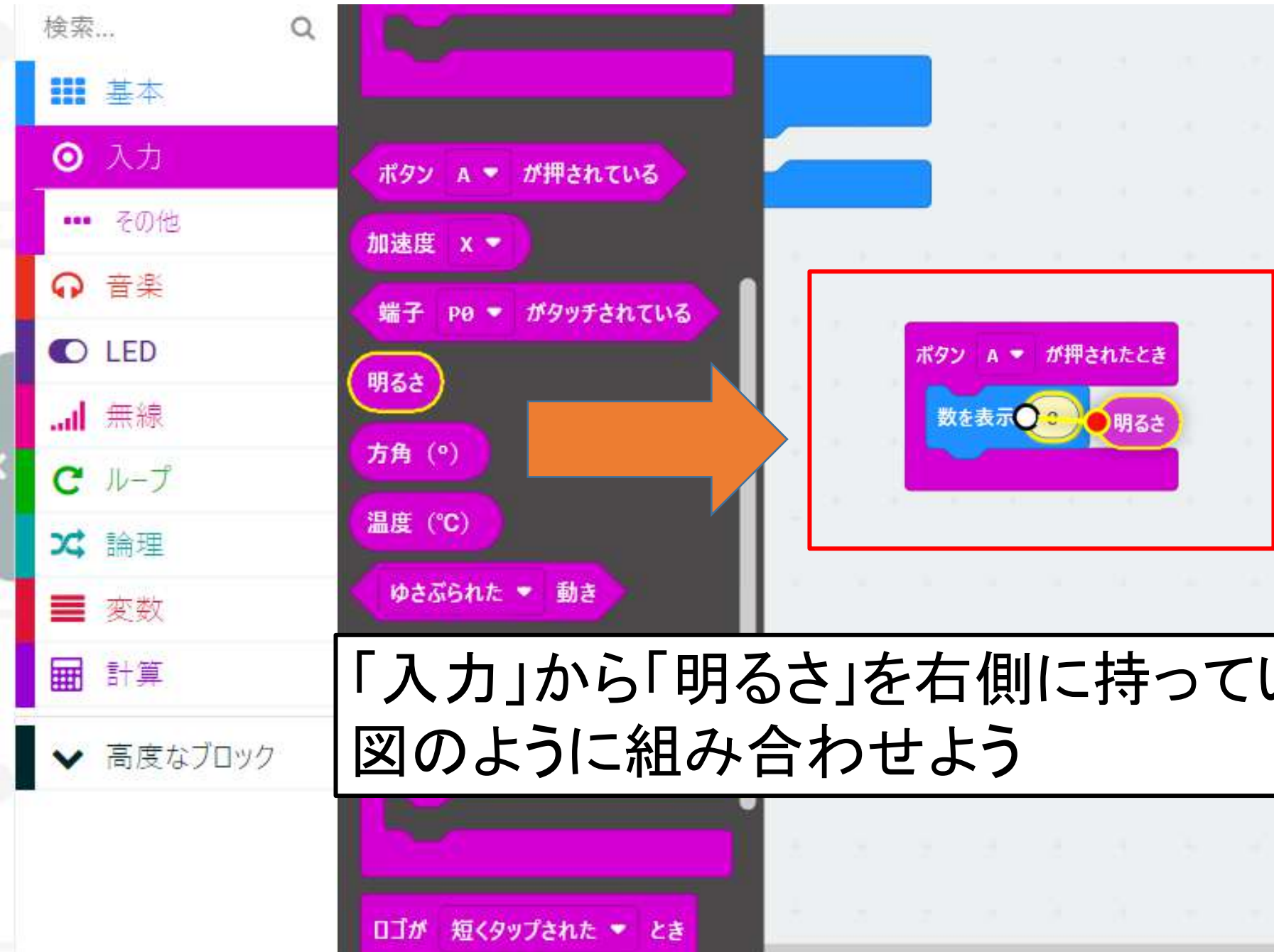
数を表示 0

ずっと

ずっと

「基本」から「数を表示」を右側に持って行って図のように組み合わせよう





「入力」から「明るさ」を右側に持って行って図のように組み合わせよう

検索...

基本

入力

その他

音楽

LED

無線

ループ

論理

変数

計算

入力

ボタン A が押されたとき

ゆさぶられたとき

端子 P0 が短くタップされたとき

ボタン A が押されている

ボタン (A、B、またはAとBの両方) が押されてはなされたときに実行されます。

数を表示 明るさ

ボタン A が押されたとき

明るさ

「入力」から「ボタンAが押されたとき」を右側に持っていこう

検索...

基本

入力

最初だけ

ずっと

「ボタンAが押されたとき」をクリックして  
「ボタンBが押されたとき」に変更しよう

無線

ループ

論理

変数

計算

高度なブロック

ボタン A が押されたとき

数を表示

明るさ

ボタン B が押されたとき

A

✓ B

A+B

「基本」から「表示を消す」を  
右側に持って行って組み合わせよう

これでAボタンを押すと明るさを数字で  
表示してくれるようになる

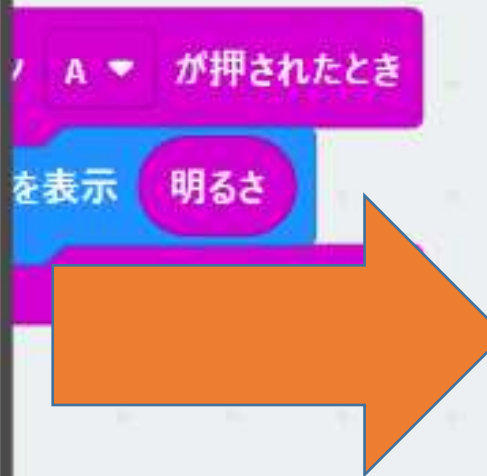
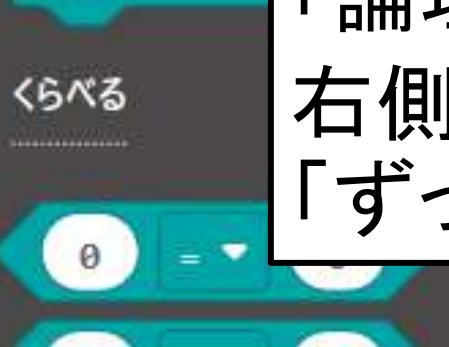
The image shows a block editor interface. On the left, a palette lists categories: ループ (Loop), 論理 (Logic), 変数 (Variables), 計算 (Math), and 高度なブロック (Advanced Blocks). A blue block labeled "表示を消す" (Clear display) is highlighted with a red box in the palette. An orange arrow points from this block to a script area on the right. In the script area, there are two event blocks: "ボタン A が押されたとき" (When button A is clicked) and "ボタン B が押されたとき" (When button B is clicked). The "表示を消す" block is being moved into the "ボタン B が押されたとき" block, also highlighted with a red box. The "表示を消す" block is highlighted with a yellow border. In the background, a mobile app interface is visible with a "Hello!" button and a slider.

検索...



## 論理

### 条件判断



「論理」から「もし真なら」を  
右側に持って行って、最初からある  
「ずっと」ブロックに組み合わせよう

基本

入力

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック



検索...



基本

入力

音楽

LED

無線

ループ

論理

変数

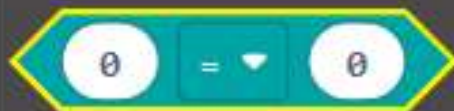
計算

高度なブロック

でなければ



くらべる



真偽値



A が押されたとき

を表示 明るさ

B

示を消

ずっと

もし 真 なら



つづけて「論理」から「0=0」を  
右側に持っていきこう



検索...

基本

入力

その他

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

ボタン A が押されている

加速度 X

端子 P0 がタッチされている

明るさ

方角 (°)

温度 (°C)

ゆさぶられた

micro:bit (V2)

まわりの音がうる

が押されたとき

明るさ

ずっと

もし 真 なら

明るさ

<

0

「入力」から「明るさ」を  
右側に持って行って「0=0」に  
組み合わせよう



組み合わせたブロックの右側  
を「100」に変更しよう



組み合わせたら図のように  
「もし真なら」ブロックに合体させよう

検索...

- 基本
- 入力
- 音楽
- LED
- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック

**基本**

数を表示 0

LED画面に表示

アイコンを表示

文字列を表示

が押されたとき

明るさ

消す

ずっと

もし 明るさ < 100 なら

数を表示 0

「基本」から「数を表示」を  
右側に持って行って図のように  
組み合わせよう

- 検索...
- 基本
  - 入力**
  - その他
  - 音楽
  - LED
  - 無線
  - ループ
  - 論理
  - 変数
  - 計算
  - 高度なブロック

ボタン A が押されている

加速度 X

端子 P0 がタッチされている

**明るさ**

方角 ( )

温度 ( )

ゆさぶられた 動き

micro:bit (V2)

まわりの音が うるさく

ロゴが 短くタップされたとき

LED画面に当たる光の明るさを  
るい) の範囲で返します。



ずっと

もし 明るさ < 100 なら

数を表示 明るさ

+

「入力」から「明るさ」を  
右側に持って行って図のように  
組み合わせよう



検索...



基本

入力

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

アイコンを表示



文字列を表示

"Hello!"

表示を消す

ずっと

最初だけ

一時停止 (ミリ秒)

100

矢印を表示

上向き ↑

が押されたとき

明るさ

が押されたとき

消す

ずっと

もし

明るさ

<

100

なら

数を表示

明るさ

一時停止 (ミリ秒)

100

「基本」から「一時停止 (ミリ秒)」を右側に持って行って図のように組み合わせよう





「一時停止 (ミリ秒)」の数字を  
クリックして 1 second (1秒) に  
設定しよう

設定した「一時停止（ミリ秒）」の  
ブロックを右クリックして、「複製する」を  
クリックしよう

クリックしたら下のように組み合わせよう

複製する

コメントを追加する

ブロックを削除する

ヘルプ

数を表示 明るさ

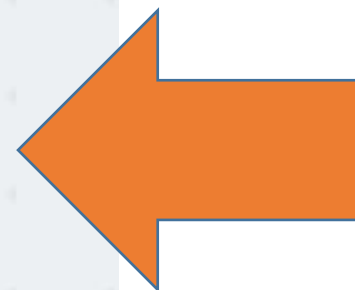
一時停止（ミリ秒）

1000

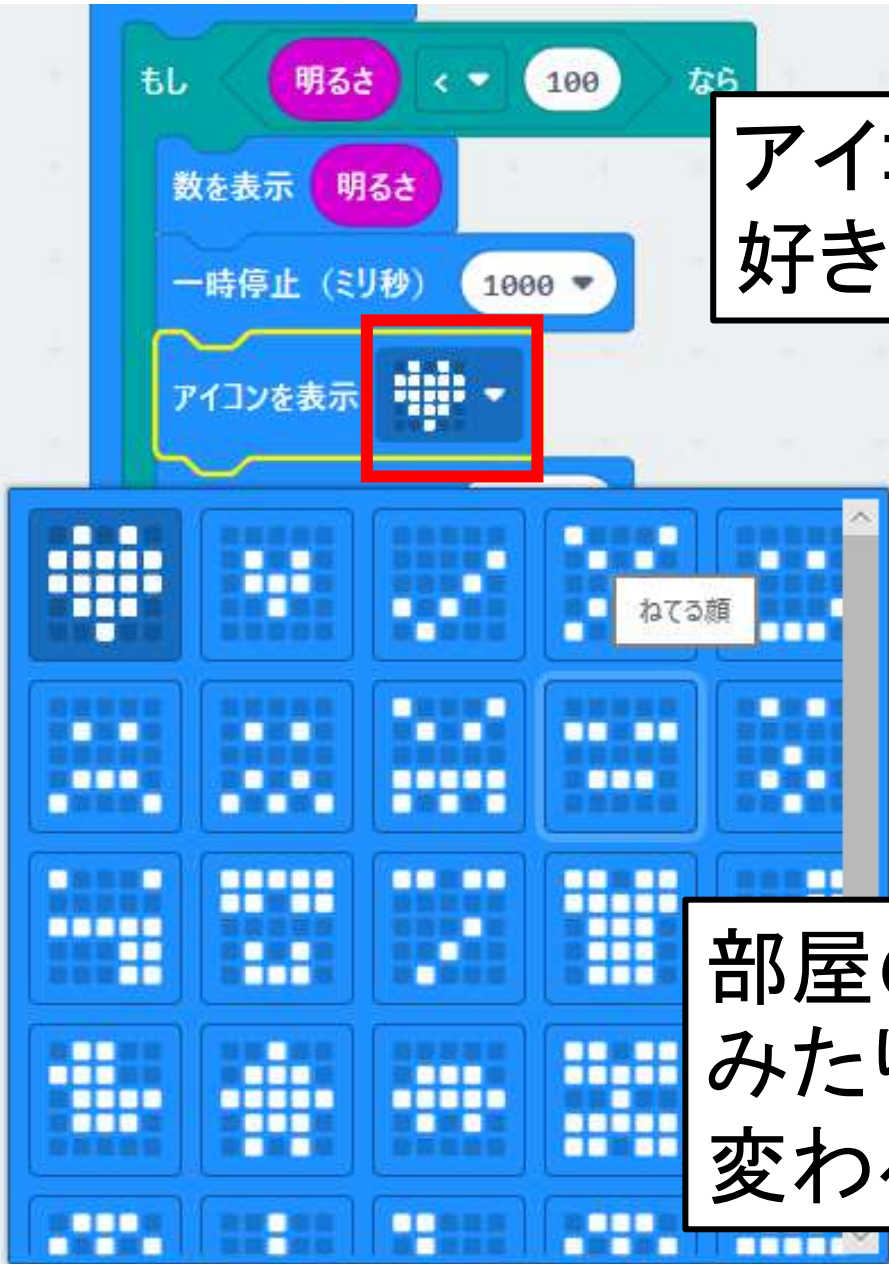
一時停止（ミリ秒）

1000

「基本」から「アイコンを表示」を  
右側に持って行って図のように  
組み合わせよう

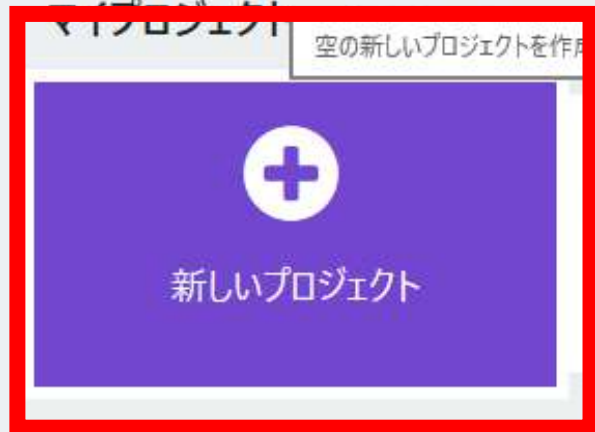


アイコンのハートマークをクリックして、好きなアイコンに変えてみよう



部屋の電気を消したり、机の下に置いてみたりして、明るさの数字がどれくらい変わるか見てみよう！

宝探しゲームを作ろう



Microbitの画面から、新しいプロジェクトをクリック

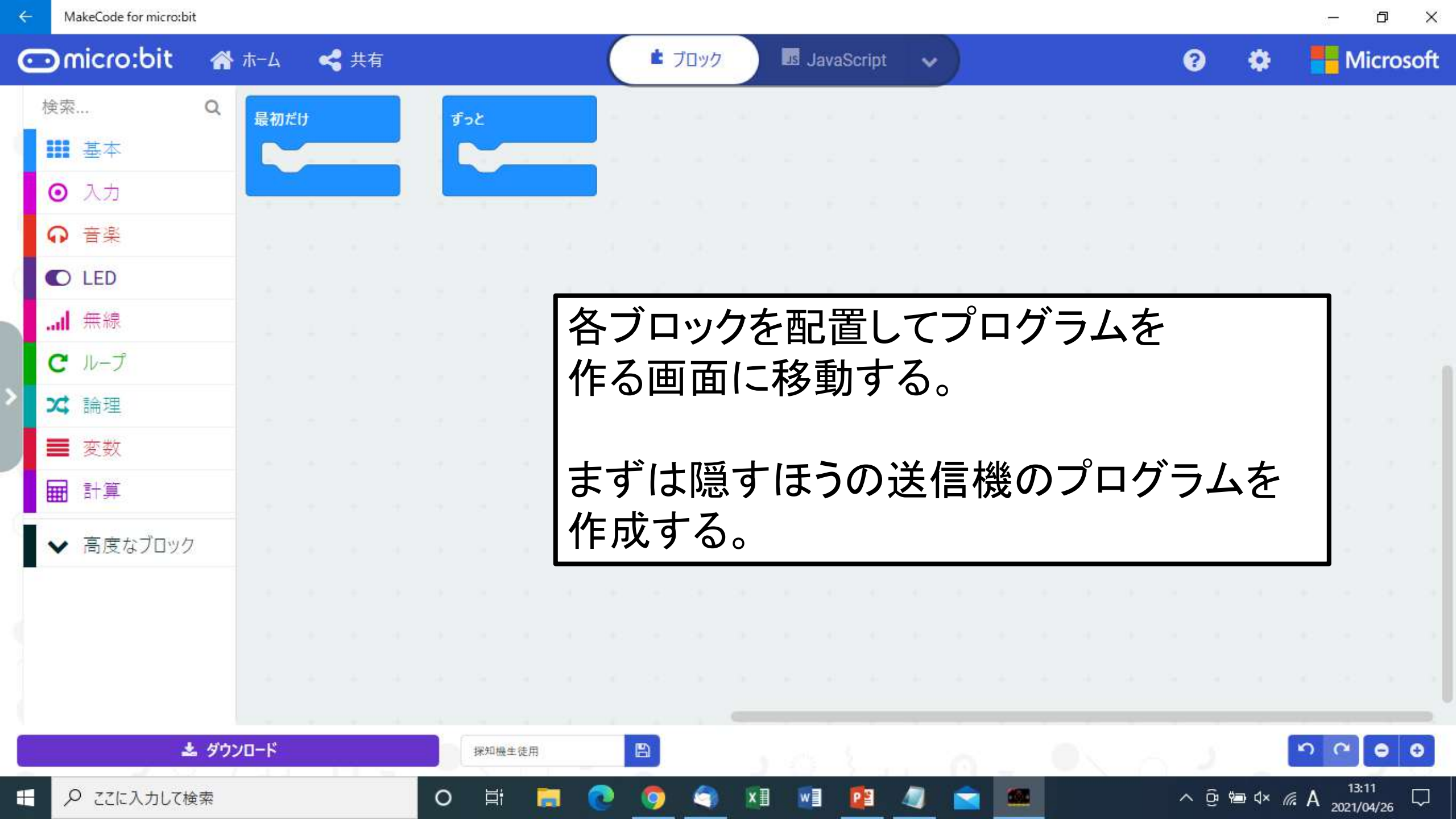
20分前

32分前

### チュートリアル







各ブロックを配置してプログラムを作る画面に移動する。

まずは隠すほうの送信機のプログラムを作成する。

検索...



基本

入力

音楽

LED

無線

その他

ループ

論理

変数

計算

高度なブロック

## 無線

無線のグループを設定 1

無線で数値を送信

無線で送信 "name" = 0

無線で文字列を送信 ""

無線で受信したとき

無線で受信したとき

無線で受信したとき receivedNumber

最初だけ

ずっと

「無線」から「無線のグループを設定」の  
ブロックを持ってきて「最初だけ」に合体させる  
※「ずっと」ブロックは使わないため消しても良い

- 基本
- 入力**
- その他
- 音楽
- LED
- 無線
- ループ
- 論理
- 変数
- 計算

### 入力

- ボタン A が押されたとき
- ゆさぶられたとき
- 端子 P0 がタッチされたとき
- ボタン A が押されている
- 加速度 X
- 端子 P0 がタッチされている
- 明るさ
- 方角 (°)

最初だけ

無線のグループを設定 1

ずっと

ボタン A が押されたとき

「入力」から「ボタンAが押されたとき」のブロックを持ってくる

検索...



基本

基本

入力

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

数を表示 0

LED画面に表示

アイコンを表示

文字列を表示 "Hello!"

表示を消す

ずっと

最初だけ

無線のグループを設定 1

ずっと

ボタン A が押されたとき

ボタン A が押されたとき

LED画面に表示

この状態

「基本」から「LED画面に表示」の  
ブロックを持ってきて「ボタンAが押されたとき」  
と合体させる

青いマスをクリックして形を「！」にする



検索...

- 基本
- 入力
- 音楽
- LED
- 無線
- ループ**
- 論理
- 変数
- 計算
- 高度なブロック

### ループ

くりかえし 4 回

もし 真 ならくりかえし

変数 index を 0 ~ 4 に変えてくりかえす

配列 list の値を変数 value に入れてくりかえす

くりかえしを終わる

「ループ」から「もし真ならくりかえし」のブロックを持ってきて「ボタンAが押されたとき」と合体させる

※LED画面に表示の下に持っていく

条件が1つ限り、同じ動作をくりかえします。

A が押されたとき

LED画面に表示

ボタン A が押されたとき

LED画面に表示

もし 真 ならくりかえし

この状態



- 基本
- 入力
- 音楽
- LED
- 無線**
- その他
- ループ
- 論理
- 変数
- 計算
- 高度なブロック

**無線**

無線のグループを設定 1

無線で数値を送信 0

無線で送信 "name" が押されたとき

無線で文字列を送信 "name" が押されたとき

無線で受信したとき receivedString

無線で受信したとき name value

無線で受信したとき receivedNumber

「無線」から「無線で数値を送信」のブロックを持ってきて「もし真ならくりかえし」と合体させる

信号を送った時に！が出て分かりやすくなる

LED画面に表示

ボタン A が押されたとき

LED画面に表示

もし 真 ならくりかえし

無線で数値を送信 0

この状態



検索...

- 基本
- 入力
- 音楽
- LED
- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック

アイコンを表示

文字列を表示 "He

表示を消す

ずっと

最初だけ

一時停止 (ミリ秒) 100

矢印を表示 上向き ↑

「基本」から「一時停止(ミリ秒)」の  
ブロックを持ってきて「もし真ならくりかえし」  
と合体させる



最初だけ

無線のグループを設定 1

ずっと

もし 真 ならくりかえし

無線で数値を送信 0

続けて受信機の仕組みをプログラムする。

「無線」から「無線で受信したとき (received Number)」のブロックを持ってくる

検索...

基

入

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

「論理」から「もし真なら～でなければ」の  
ブロックを持ってきて「無線で受信したとき」ブロック  
と合体させる

もし 真 ▼ なら

+

もし 真 ▼ なら

でなければ -

+

くらべる

0 = 0

0 < 0

" " = " "



無線で受信したとき receivedNumber

無線で受信したとき receivedNumber

もし 真 ▼ なら

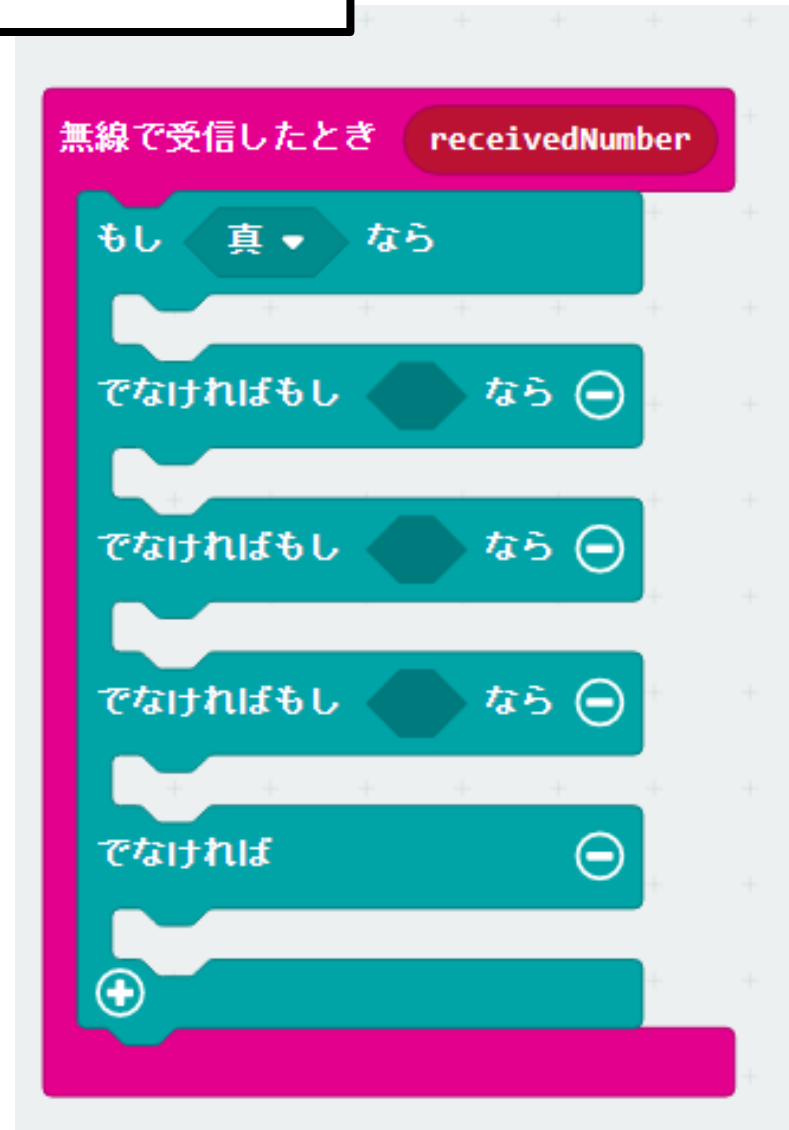
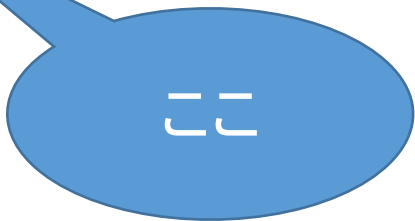
でなければ -

+



この状態

「もし真なら～でなければ」ブロックの左下にある  
+ボタンを**3回**押す





検索... 🔍

基本

入力

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

でなければ



くらべる



真偽値



「論理」から「 $0 < 0$ 」のブロックを4つ持ってくる

※1つ持ってきたら右クリック⇒複製でもOK



「無線」から「受信したパケットの信号強度」ブロックを4つ持ってくる

※1つ持ってきたら右クリック⇒複製でもOK

持ってきたら「0<0」ブロックの右側に合体させる

- ループ
- 論理
- 変数
- 計算
- 高度なブロック

The screenshot shows a programming environment with a block palette on the left and a workspace on the right. The palette contains several blocks, including a pink '無線で受信したとき' (When received wirelessly) block with 'name' and 'value' fields, and a '受信したパケットの 信号強度' (Received packet signal strength) block. The workspace shows a sequence of blocks: a pink '無線で受信したとき' block, followed by three teal 'でなければもし' (If not) blocks, and finally a comparison block '0 < 0'. A blue arrow points from the '無線で受信したとき' block in the palette to the '0 < 0' block in the workspace, indicating the process of attaching the block to the comparison operator.

無線で受信したとき receivedNumber

もし  真  偽 なら

< ▼ 受信したパケットの 信号強度 ▼

でなければもし  真  偽 なら ⊖

でなければもし  真  偽 な ← ⊖

でなければもし  真  偽 なら ⊖ ←

でなければ  真  偽 ⊖

+

< ▼ 受信したパケットの 信号強度 ▼

< ▼ 受信したパケットの 信号強度 ▼

< ▼ 受信したパケットの 信号強度 ▼

合体した「0<受信したパケット」ブロックを「もし～なら」ブロックにそれぞれ合体させる

無線で受信したとき

receivedNumber

もし



受信したパケットの 信号強度 なら

でなければもし



受信したパケットの 信号強度 なら

でなければもし



受信したパケットの 信号強度 なら

でなければもし



受信したパケットの 信号強度 なら

でなければ

「0<受信したパケット」ブロックを  
上から-45、-55、-65、-70に書きかえる

- 基本
- 入力
- 音楽
- LED
- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック

LED画面に表  
「基本」から「アイコンを表示」ブロックを4つ  
「もし~なら」ブロックの条件の部分に合体させる  
「でなければ」の部分には表示を消すを入れる

アイコンを表示



選択されたアイコンを、LED画面に表示します。

表示を消す

ずっと



でなければもし < -70 < 受信したパケットの 信号強度 > なら

でなければもし < -100 < 受信したパケットの 信号強度 > なら

でなければ

+





無線で受信したとき receivedNumber

もし -45 < 受信したパケットの 信号強度 なら

アイコンを表示



でなければもし -55 < 受信したパケットの 信号強度 なら

アイコンを表示



でなければもし -65 < 受信したパケットの 信号強度 なら

アイコンを表示



でなければもし -70 < 受信したパケットの 信号強度 なら

アイコンを表示

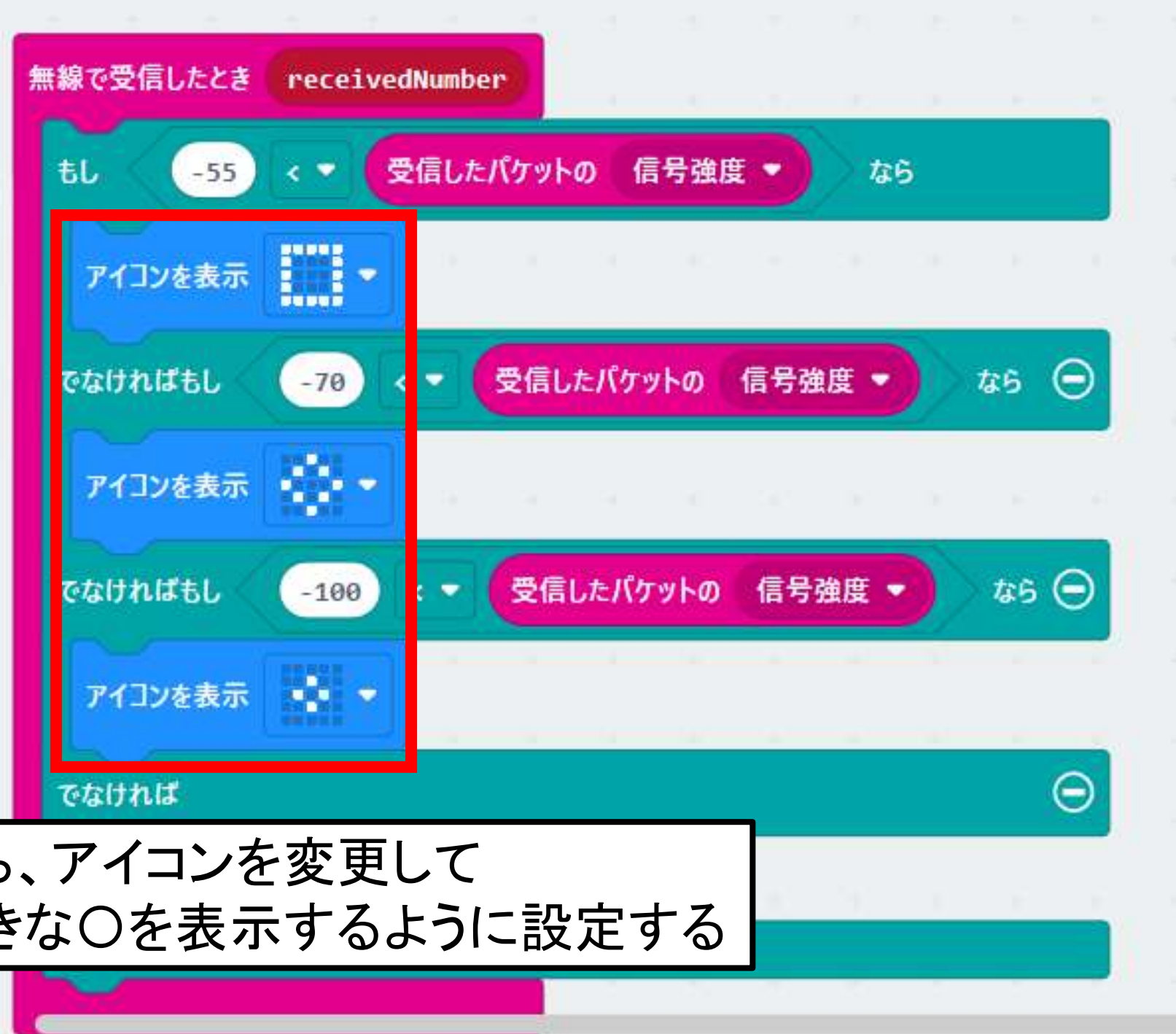


でなければ

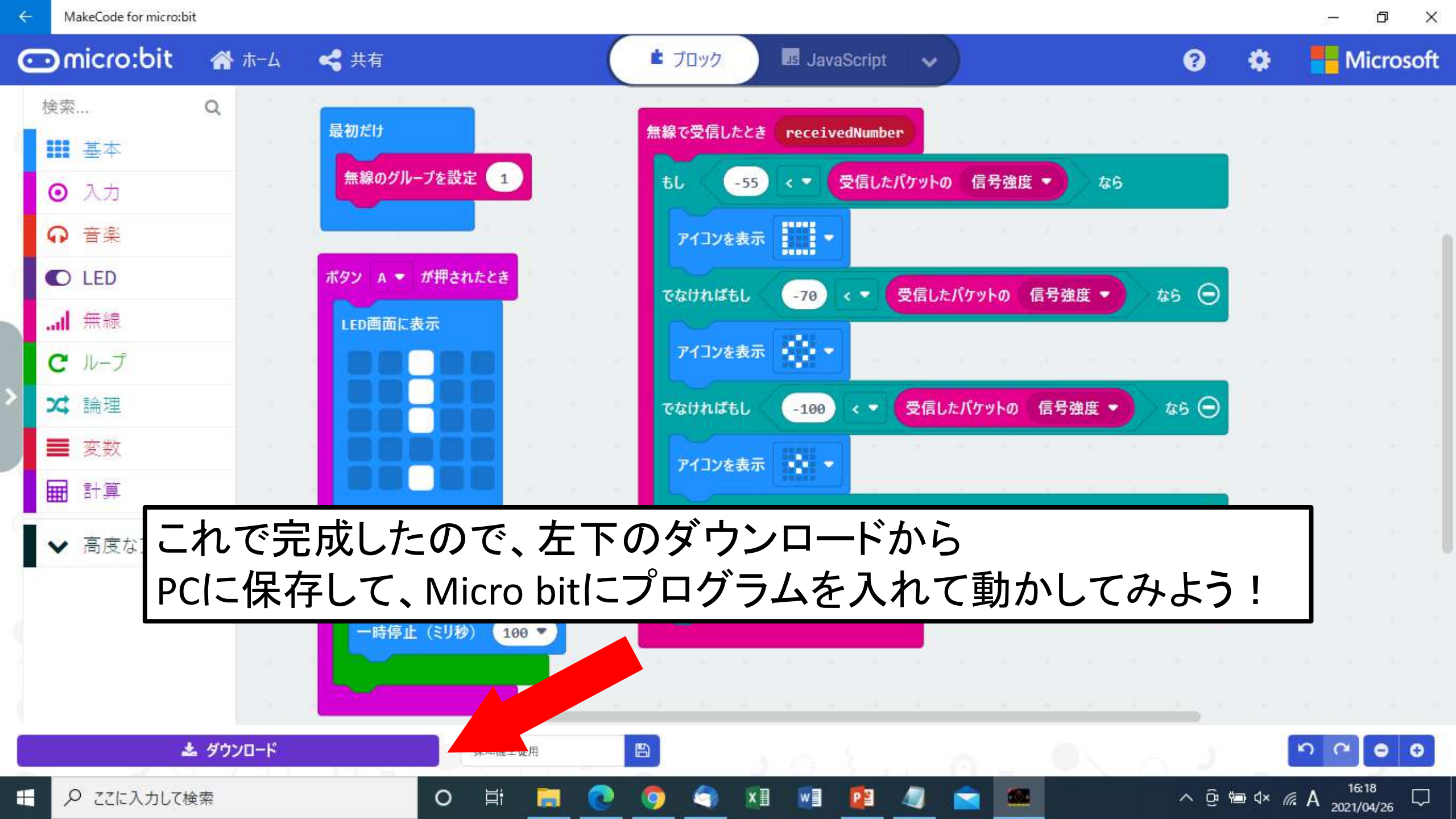
表示を消す

Micro bitでは、信号の強さが-128~-42の間で表されることになっているため、これで送信機に近づくほどアイコンが表示されるようにすることができた





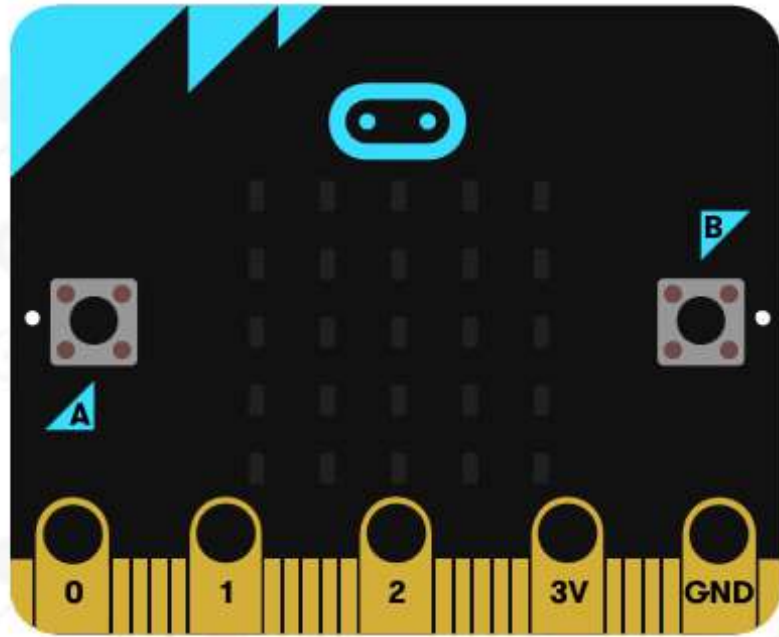
アイコンを表示の▼から、アイコンを変更して送信機に近づくほど大きな○を表示するように設定する



これで完成したので、左下のダウンロードから  
PCに保存して、Micro bitにプログラムを入れて動かしてみよう！

ダウンロード

ドライブゲームを作ろう



検索...



基本

入力

音楽

LED

無線

ループ

論理

変数

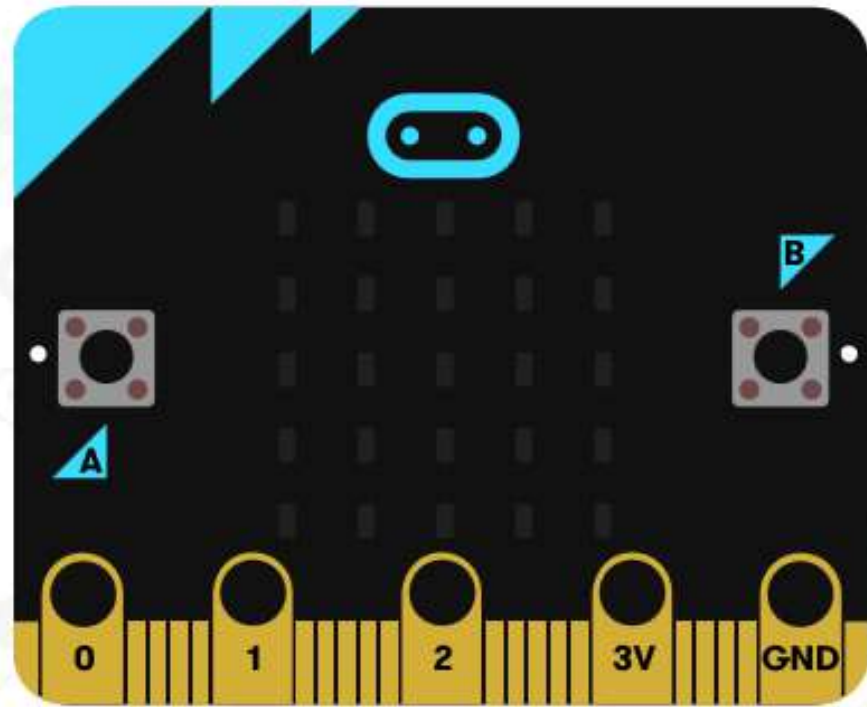
計算

最初だけ

ずっと

LED機能を使って、上から迫ってくる車を避ける  
ゲームを作ろう





検索...



基本

入力

音楽

LED

無線

ループ

論理

変数

計算

変数

変数を追加する...

まずは、プログラムの入れ物になる「変数」を作成する

左側のブロック一覧から「変数」をクリックして、「変数を追加する...」をクリックしよう



# 「自機」「敵機」「スピード」「得点」の4つの変数を作成しよう

作成する変数の名前：



自機



OK



作成する変数の名前：



敵機



OK



作成する変数の名前：



スピード



OK



作成する変数の名前：



得点



OK



下の図のように、4つの変数とブロックが出来上がるので、「変数 得点を0にする」を2つ右側に持っていきこう

The image shows a block editor interface with a sidebar on the left containing categories: 入力 (Input), 音楽 (Music), LED, 無線 (Wireless), ループ (Loop), 論理 (Logic), 変数 (Variables), 計算 (Math), and 高度なブロック (Advanced Blocks). The main workspace contains several red blocks: 'スピード' (Speed), '得点' (Score), '敵機' (Enemy), '自機' (Self), and two '変数 得点 を 0 にする' (Set Score to 0) blocks. A blue arrow points from the workspace to a red-bordered inset box containing two '変数 得点 を 0 にする' blocks. A tooltip below the inset box reads: '変数の値を、与えられた値と同じにします。' (Set the value of the variable to the given value).

持ってきた変数のブロックを、それぞれ下の図のように変更しよう



「ゲーム」から、「スプライトを作成 x:○ y:○」を  
2つ右側に持っていきこう

ゲーム

- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック
- 関数
- 配列
- 文字列
- ゲーム
- その他
- 画像
- 入出力端子

スプライトを作成 x: 2 y: 2

スプライト を削除

スプライト が削除済み

スプライト を 1 ドット進める

スプライト 方向転換 右 に 45 °

スプライト の x を 1 だけ増やす

スプライト の x に 0 を設定する

スプライト の x

最初だけ

ずっと

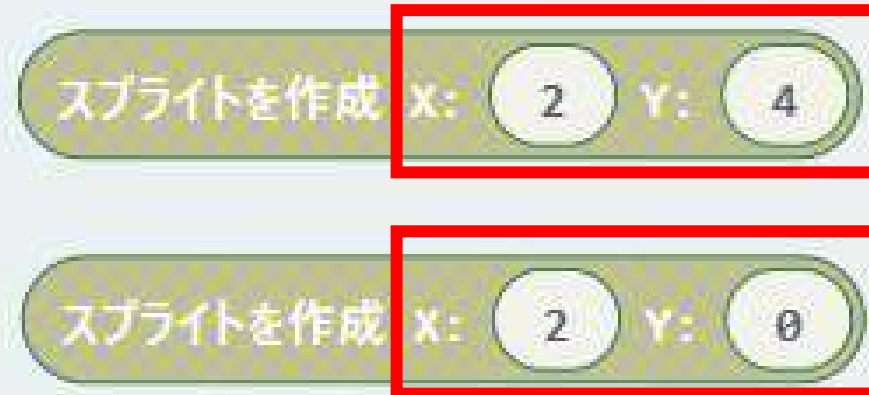
変数 自機 を 0 にする

変数 敵機 を 0 にする

スプライトを作成 x: 2 y: 2

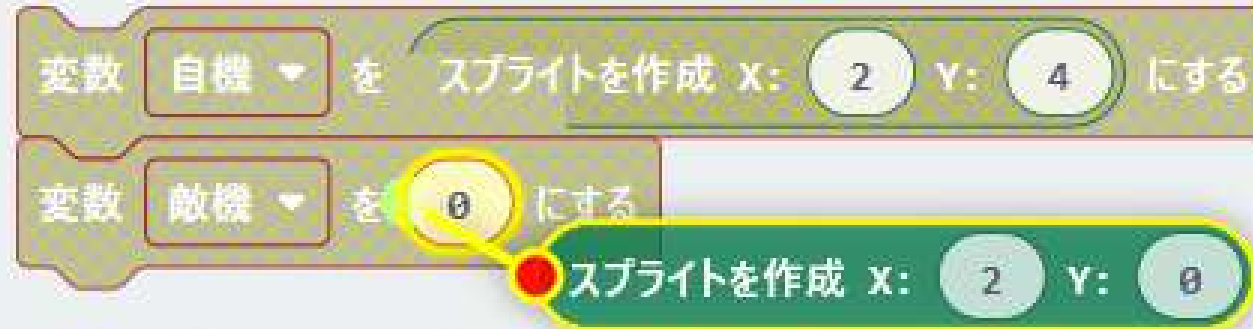
スプライトを作成 x: 2 y: 2

持ってきたブロックを、  
「スプライトを作成 X:2 Y:4」と  
「スプライトを作成 X:2 Y:0」に変更しよう





変更したら、先ほど持ってきた変数ブロックに  
それぞれ組み合わせよう



次に、「変数」からもう一度「変数 得点を0にする」ブロックを右側に持って行って組み合わせよう

The image shows a programming interface with a sidebar on the left and a workspace on the right. The sidebar contains several categories: 変数 (Variables), 計算 (Math), 高度なブロック (Advanced Blocks), 関数 (Functions), 配列 (Arrays), 文字列 (Text), ゲーム (Games), 画像 (Images), 入出力端子 (Input/Output), シリアル通信 (Serial Communication), and 制御 (Control). The '変数' category is selected, showing a list of variables: スピード (Speed), 得点 (Score), 敵機 (Enemy), and 自機 (Self). The '得点' variable is highlighted with a yellow box, and the block '変数 得点 を 0 にする' (Set variable Score to 0) is also highlighted. Below it, another block '変数 得点 を 1 だけ増やす' (Increase variable Score by 1) is visible. A blue arrow points from the highlighted block to the workspace. In the workspace, there are two blue blocks labeled '最初だけ' (Only at the start) and 'ずっと' (Forever). Below them, a script is being assembled, consisting of three blocks: '変数 自機 を スプライトを作成 x: 2 y: 4 にする' (Set variable Self to create sprite at x: 2 y: 4), '変数 敵機 を スプライトを作成 x: 2 y: 0 にする' (Set variable Enemy to create sprite at x: 2 y: 0), and '変数 得点 を 0 にする' (Set variable Score to 0).

変数 自機 ▼ を スプライトを作成 x: 2 y: 4 にする

変数 敵機 ▼ を スプライトを作成 x: 2 y: 0 にする

変数 **スピード ▼** を 0 にする

- ✓ スピード
- 得点
- 敵機
- 自機
- 変数を作成する...
- 変数の名前を変更...
- この変数「スピード」を削除する



変数 自機 ▼ を スプライトを作成 x: 2 y: 4 にする

変数 敵機 ▼ を スプライトを作成 x: 2 y: 0 にする

変数 スピード ▼ を **100** にする

組み合わせたら、「得点」の部分をクリックして「スピード」に変更しよう。

続けて、0と数字が入っているところを**100**に変更しよう

さらに「変数」から「変数 得点を0にする」  
ブロックを右側に持って行って組み合わせよう

The image shows a programming environment with a sidebar on the left and a workspace on the right. The sidebar contains various category icons: 音楽 (Music), LED, 無線 (Wireless), ループ (Loop), 論理 (Logic), 変数 (Variables), 計算 (Math), 高度なブロック (Advanced Blocks), 関数 (Functions), 配列 (Arrays), 文字列 (Strings), ゲーム (Games), and 画像 (Images). The '変数' (Variables) category is highlighted in red. In the workspace, a script is being built with the following blocks from top to bottom: '変数 自機' (Variable Self Ship) set to 'スプライトを作成 x: 2 y: 4' (Create Sprite at x: 2 y: 4), '変数 敵機' (Variable Enemy Ship) set to 'スプライトを作成 x: 2 y: 0' (Create Sprite at x: 2 y: 0), '変数 スピード' (Variable Speed) set to '100' (100), and '変数 得点' (Variable Score) set to '0' (0). The '変数 得点' block is highlighted with a red border. The workspace background is a light gray grid.

組み合わせた変数ブロックを、「最初だけ」ブロックと合体させよう





- 基本
- 入力
- その他
- 音楽
- LED
- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック
- 関数
- 配列

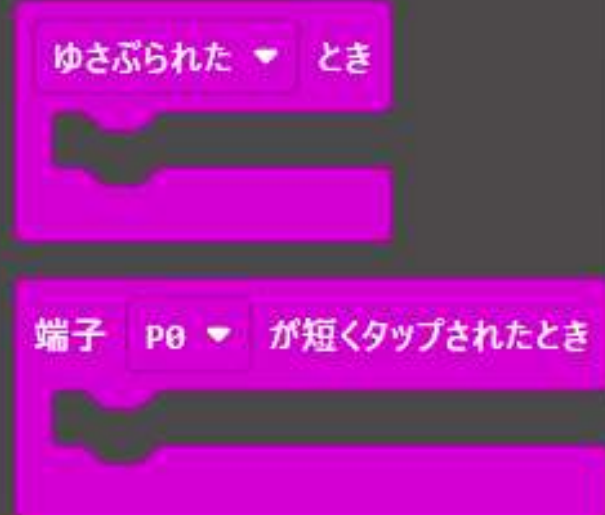
## 入力

ボタン A が押されたとき



ゆさぶられたとき

端子 P0 が短くタップされたとき



ボタン A が押されたとき

加速度 X

端子 P0 がタッチされている



変数 敵機 を スプライトを作成

変数 スピード を 100 にする

変数 得点 を 0 にする



ボタン A が押されたとき



「入力」から「ボタンAが押されたとき」を右側に持っていこう

- ループ
- 論理
- 変数
- 計算
- 高度なブロック
- 関数
- 配列
- 文字列
- ゲーム
- その他
- 画像
- 入出力端子
- シリアル通信
- 制御
- 拡張機能

### ゲーム

スプライトを作成 x: 2 Y: 2

スプライト を削除

スプライト が削除済み

スプライト を 1 ドット進める

スプライト 方向転換 右 に 45 °

スプライト の x を 1 だけ増やす

スプライト の x

スプライト が他のスプライト にさわっている

変数 敵機 を スプライトを作成 x: 2 Y:

変数 スピード を 100 にする

変数 得点 を 0 にする

ボタン A が押されたとき

スプライト を 1 ドット進める

「ゲーム」から「スプライトを1ドット進める」を右側に持って行って組み合わせよう



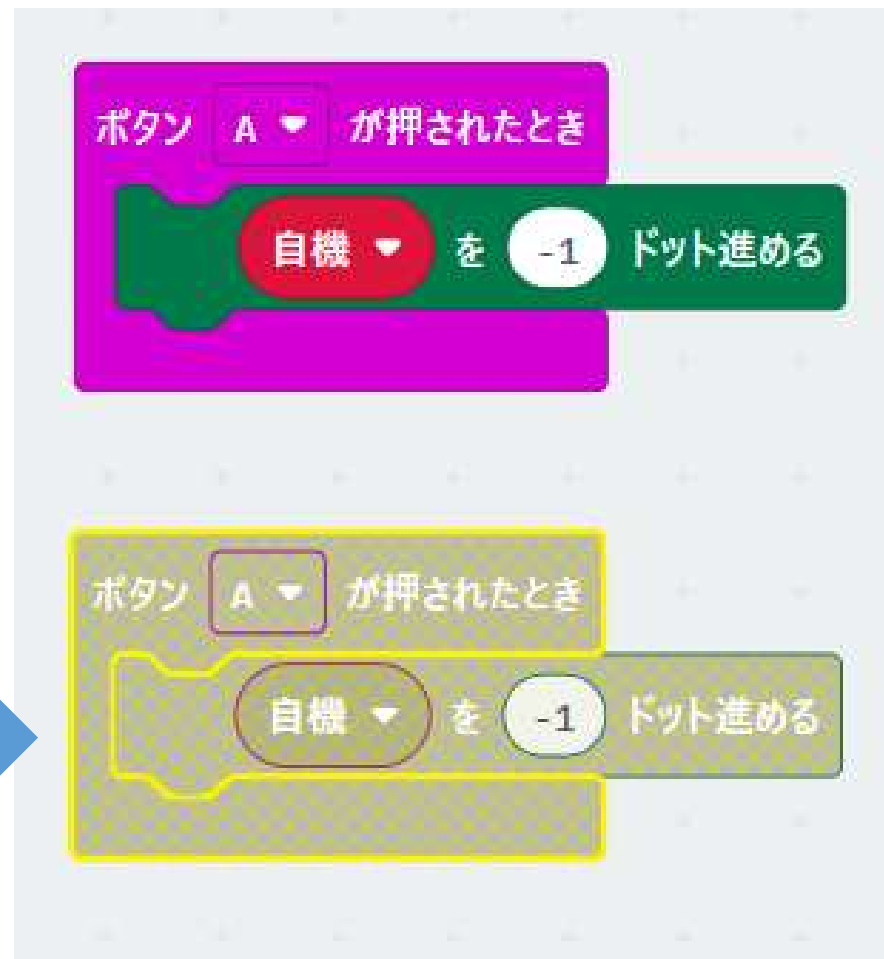
- スピード
- スプライト
- 得点
- 敵機
- ✓ 自機
- 変数を作成する...
- 変数の名前を変更...
- この変数「自機」を削除する



「スプライト」をクリックして「自機を1ドット進める」に変更しよう

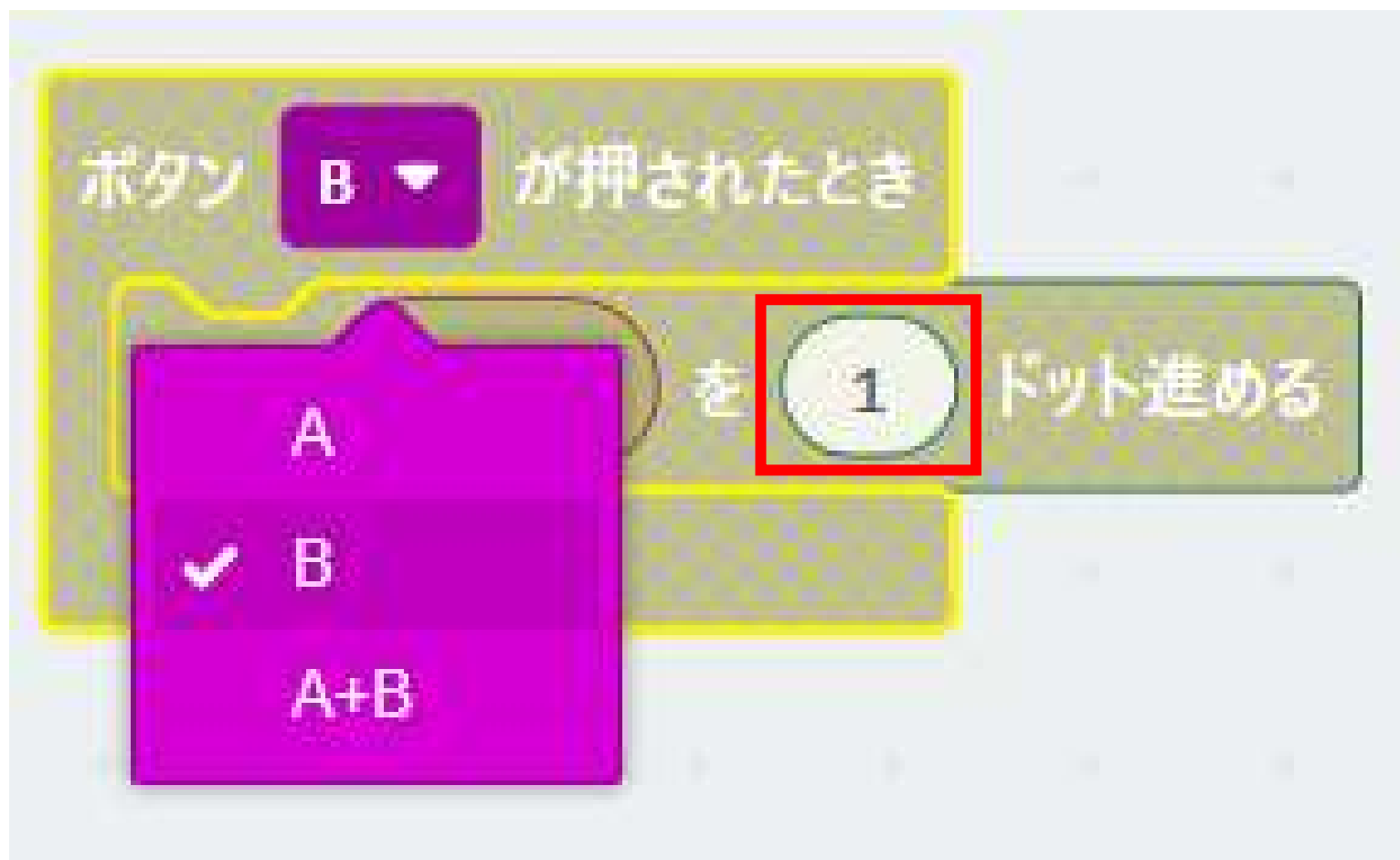
右側の数字を1から-1に変更しよう

「ボタンAが押されたとき」を右クリックして  
「複製する」をクリックしよう



コピーされた方のブロックを  
「ボタンBが押されたとき」に変更しよう

右側の数字を-1から**1**に変更しよう





検索...



基本

入力

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

関数

配列

文字列

アイコンを表示



文字列を表示 "Hello!"

表示を消す

ずっと

最初だけ

一時停止 (ミリ秒) 100

矢印を表示 上向き ↑

作成 X: 2 Y: 4 にする

作成 X: 2 Y: 0 にする

ずっと

一時停止 (ミリ秒) 100

「基本」から「一時停止 (ミリ秒)」を持ってきて「ずっと」ブロックに組み合わせよう

検索...



基本

入力

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

関数

配列

変数

変数を追加する...

スピード ▾

スプライト ▾

得点 ▾

敵機 ▾

自機 ▾

変数 スプライト

変数 スプライト ▾ を 1 だけ増やす

にする

にする

ずっと

一時停止 (ミリ秒)

スピード ▾

「変数」から「スピード」を  
持ってきて「一時停止」ブロックに組み合わせよう

「ゲーム」から「スプライトのXを1だけ増やす」を2つ持ってきて図のように組み合わせよう

The image shows a programming environment with a block palette on the left and a workspace on the right. The palette includes categories like 音楽 (Music), LED, 無線 (Wireless), ループ (Loop), 論理 (Logic), 変数 (Variables), 計算 (Math), 高度なブロック (Advanced Blocks), 関数 (Functions), 配列 (Arrays), and 文字列 (Strings). The workspace contains a sequence of blocks: 'スプライトを削除' (Delete Sprite), 'スプライトが削除済み' (Sprite Deleted), 'スプライトを1ドット進める' (Move Sprite 1 Dot), 'スプライト方向転換 右に45度' (Turn Sprite Right 45 Degrees), 'スプライトのXを1だけ増やす' (Increase Sprite X by 1), 'スプライトのXに0を設定する' (Set Sprite X to 0), and 'スプライトのX' (Sprite X). A red box highlights the 'スプライトのXを1だけ増やす' block in the palette. On the right, a custom block definition is shown, consisting of a blue 'ずっと' (Forever) loop block containing two green 'スプライトのXを1だけ増やす' blocks, one with a warning icon. A red box highlights the entire custom block definition, and a yellow box highlights the second block within it.



「スプライト」をクリックして、  
「敵機のXを1だけ増やす」にどちらも変更しよう

Xのところをクリックして、  
「敵機のYを1だけ増やす」に変更しよう

ずっと

一時停止 (ミリ秒)

スピード ▼

敵機 ▼ の Y ▼ を 1 だけ増やす

敵機 ▼ の X ▼ を 1 だけ増やす

- X
- ✓ Y
- 方向
- LED画面の明るさ
- 点滅



「計算」から、「0から10までの乱数」を持ってきて、「敵機のXを～」と組み合わせよう

- 検索...
- 基本
- 入力
- 音楽
- LED
- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック
- 関数
- 配列
- 文字列
- ゲーム

0 ÷ 0

0

0 を 1 で割ったあまり

0 と 0 のうち 小さい方

0 と 0 のうち 大きい方

0 の絶対値

平方根 0

小数点以下四捨五入 0

**0 から 10 までの乱数**

0 を 0 以上 0 以下の範囲に制限

数値をマップする 0 元の下限 0 元の上限 1023 結果の下限 0 結果の上限 4

ランダムに真か偽に決める

と

一時停止 (ミリ秒) スピード

敵機 の Y を 1 だけ増やす

敵機 の X を **0 から 10 までの乱数** だけ増やす

乱数を「-1から1までの乱数」に変更しよう

The image shows a Scratch script on a light blue grid background. The script is contained within a blue 'ずっと' (Forever) loop block. Inside the loop, there are three blocks: a blue '一時停止 (ミリ秒) スピード' (Pause in milliseconds) block, a green '敵機 の Y を 1 だけ増やす' (Increase enemy Y by 1) block, and another green '敵機 の X を -1 から 1 までの乱数 だけ増やす' (Increase enemy X by a random number between -1 and 1) block. The second green block is highlighted with a red rectangular border. The numbers '-1' and '1' in the second green block are each enclosed in a white circle, and the entire range '-1 から 1 までの乱数' is highlighted with a yellow oval border.

検索...

- 基本
- 入力
- 音楽
- LED
- 無線
- ループ
- 論理**
- 変数
- 計算
- 高度なブロック
- 関数
- 配列
- 文字列

論理

条件判断

もし 真 なら

でなければ

くらべる

0 = 0

0 < 0

0 = 0

ずっと

一時停止 (ミリ秒) スピード

敵機 の Y を 1 だけ増やす

敵機 の X を -1 から 1 までの乱数 だけ増やす

もし 真 なら

「論理」から「もし真なら」を  
持ってきて図のように組み合わせよう

- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック
- 関数
- 配列
- 文字列
- ゲーム
- その他
- 画像
- 入出力端子
- シリアル通信
- 制御

Sprite の X を 1 だけ増やす

Sprite の X に 0 を設定する

Sprite の X

**Sprite が他のSprite にさわっている**

Sprite が端にある

Sprite が端にあれば反射させる

ライフ数を 0 だけ減らす

ライフ数

ライフ数

点数を 0 にする

ずっと

一時停止 (ミリ秒) Speed

敵機 の Y を 1 だけ増やす

敵機 の X を -1 から 1 までの乱数 だけ増やす

もし **Sprite が他のSprite にさわっている** なら

+

「ゲーム」から「Spriteが他のSpriteにさわっている」を持ってきて図のように組み合わせよう

「スプライト」をクリックして、  
「自機が他のスプライトにさわっている」に変更しよう

敵機 の Y を 1 だけ増やす

敵機 の X を -1 から 1 までの乱数 だけ増やす

もし 自機 が他のスプライト にさわっている なら

スピード

スプライト

得点

敵機

✓ 自機

変数を作成する...

変数の名前を変更...

この変数「自機」を削除する



- 無線
- ループ
- 論理
- 変数**
- 計算
- 高度なブロック
- 関数
- 配列
- 文字列
- ゲーム
- 画像
- 入出力端子
- シリアル通信
- 制御
- 拡張機能

## 変数

変数を追加する...

スピード ▾

スプライト ▾

得点 ▾

**敵機 ▾**

自機 ▾

変数 スプライト ▾ を 0 にする

変数 スプライト ▾ を 1 だけ増やす

ずっと

一時停止 (ミリ秒) スピード ▾

敵機 ▾ の Y ▾ を 1 だけ増やす

敵機 ▾ の X ▾ を -1 から 1 までの乱数 だけ増やす

スプライト 敵機 ▾ にさわっている なら

+

「変数」から「敵機」を持ってきて、  
図のように組み合わせよう

- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック
- 関数
- 配列
- 文字列
- ゲーム
- その他
- 画像
- 入出力端子
- シリアル通信

## ゲーム

スプライトを作成 x: 2 Y: 2

スプライト を削除

スプライト が削除済み

スプライト を

スプライト 方向転換 右 に 45 °

スプライト の x を 1 だけ増やす

スプライト の x に 0

スプライト の x

ゲームエンジンからスプライトを削除します。スプライトは画面に表示されたり、他のスプライトと連動したりすることはなくなります。

ずっと

一時停止 (ミリ秒) スピード

敵機 の Y を 1 だけ増やす

敵機 の X を -1 から 1 までの乱数 だけ増やす

もし 自機 が他のスプライト 敵機 にさわっている なら

スプライト を削除

「ゲーム」から「スプライトを削除」を持ってきて、図のように組み合わせよう

- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック
- 関数
- 配列
- 文字列
- ゲーム
- その他
- 画像
- 入出力端子
- シリアル通信

スプライト が端にあれば反射させる

ライフ数を 0 だけ減らす

ライフ数を 0 だけ増やす

ライフ数を 0 にする

点数を 0 にする

点数を 1 だけ増 現在の点数を設定します。

カウントダウンを開始 (ミリ秒) 10000

点数

ゲームオーバーにする

ゲームオーバーである

ずっと

一時停止 (ミリ秒) スピード

敵機 の Y を 1 だけ増やす

敵機 の X を -1 から 1 までの乱数 だけ増やす

もし 自機 が他のスプライト 敵機 にさわっている なら

スプライト を削除

点数を 0 にする

「ゲーム」から「点数を0にする」を持ってきて、図のように組み合わせよう



- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック
- 関数
- 配列
- 文字列
- ゲーム
- その他
- 画像
- 入出力端子
- シリアル通信
- 制御

ライフ数を 0 だけ減らす

ライフ数を 0 だけ増やす

ライフ数を 0 にする

点数を 0 にする

点数を 1 だけ増やす

カウントダウンを開始 (ミリ秒) 10000

点数

ゲームオーバーにする

ゲームオーバーである

一時停止中である

ゲーム中である



ずっと

一時停止 (ミリ秒) スピード

敵機 の Y を 1 だけ増やす

敵機 の X を -1 から 1 までの乱数 だけ増やす

もし 自機 が他のスプライト 敵機 にさわっている なら

スプライト を削除

点 0

ゲームオーバーにする

「ゲーム」から「ゲームオーバーにする」を持ってきて、図のように組み合わせよう

スピード

スプライト

得点

敵機

✓ 自機

変数を作成する...

変数の名前を変更...

この変数「自機」を削除する

1 だけ増やす

-1 から 1 までの乱数 だけ増やす

スプライト 敵機 にさわっている なら

自機 を削除

点数を 0 にする

ゲームオーバーにする

「スプライトを削除」を「自機を削除」  
に変更しよう



検索...

基本

入力

音楽

LED

無線

ループ

論理

変数

計算

高度なブロック

関数

配列

文字列

## 変数

変数を追加する...

スピード ▾

スプライト ▾

得点 ▾

敵機 ▾

自機 ▾

変数 スプライト ▾ を 0 にする

変数 スプライト ▾ を 1 だけ増やす

この変数の値を返します。

「変数」から「得点」を持ってきて、「点数を0にする」に組み合わせよう

ずっと

一時停止 (ミリ秒) スピード ▾

敵機 ▾ の Y ▾ を 1 だけ増やす

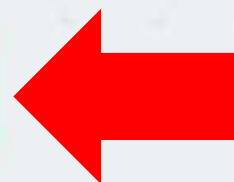
敵機 ▾ の X ▾ を -1 から 1 までの乱数 だけ増やす

もし 自機 ▾ が他のスプライト 敵機 ▾ にさわっている なら

自機 ▾ を削除

点数を 得点 ▾ にする

ゲームオーバーにする



一度クリックしてこのブロックだけ選択

複製する

コメントを追加する

ブロックを削除する

ヘルプ

「一時停止（ミリ秒）」ブロックをクリックしてから  
右クリックして、3つ複製を作ろう

一時停止（ミリ秒）

スピード ▾

一時停止（ミリ秒）

スピード ▾

一時停止（ミリ秒）

スピード ▾

一時停止 (ミリ秒)

スピード ▼

敵機 ▼ の Y ▼ を 1 だけ増やす

敵機 ▼ の X ▼ を -1 だけ減らす

もし 自機 ▼ が他のスプライト 敵機 ▼ にさわっている なら

自機 ▼ を削除

点数を 得点 ▼ にする

ゲームオーバーにする

一時停止 (ミリ秒)

スピード ▼

コピーした「一時停止 (ミリ秒)」ブロックを一番下に組み合わせよう

一時停止 (ミリ秒)

スピード ▼

「論理」から「もし真なら」を  
持ってきて、図のように組み合わせよう

The image shows a block editor interface with a sidebar on the left and a workspace on the right. The sidebar contains various block categories: 入力 (Input), 音楽 (Music), LED, 無線 (Wireless), ループ (Loop), 論理 (Logic), 変数 (Variables), 計算 (Math), 高度なブロック (Advanced Blocks), 関数 (Functions), 配列 (Arrays), and 文字列 (Text). The 'Logic' category is highlighted in teal. In the workspace, a script is being built. A red box highlights a 'もし 真 なら' (If True) block being dragged from the sidebar. Another red box highlights a completed script segment: a 'もし 自機 が他のスプライト 敵機 にさわっている なら' (If self sprite touches enemy sprite) block, followed by '自機 を削除' (Delete self sprite), '点数を 得点 にする' (Set score to points), 'ゲームオーバーにする' (Game over), a '一時停止 (ミリ秒) スピード' (Pause in milliseconds speed) block, and another 'もし 真 なら' (If True) block. The '論理' category in the sidebar also has a red box around it.



「ゲーム」から「スプライトのX」を右側に持っていこう

```
スプライトを作成 x: 2 Y: 2
スプライト を削除
スプライト が削除済み
スプライト を 1 ドット進める
スプライト 方向転換 右 に 45 °
スプライト の x を 1 だけ増やす
スプライト の x に 0 を設定する
```

スプライト の X

スプライトのX  
スプライトのプロパティを取得します。

```
一時停止 (ミリ秒) スピード
を 1 だけ増やす
敵機 の x を -1 から 1 までの乱数 だけ増やす
もし 自機 が他のスプライト 敵機 にさわっている なら
自機 を削除
点数を 得点 にする
ゲームオーバーにする
一時停止 (ミリ秒) スピード
もし 真 なら
```

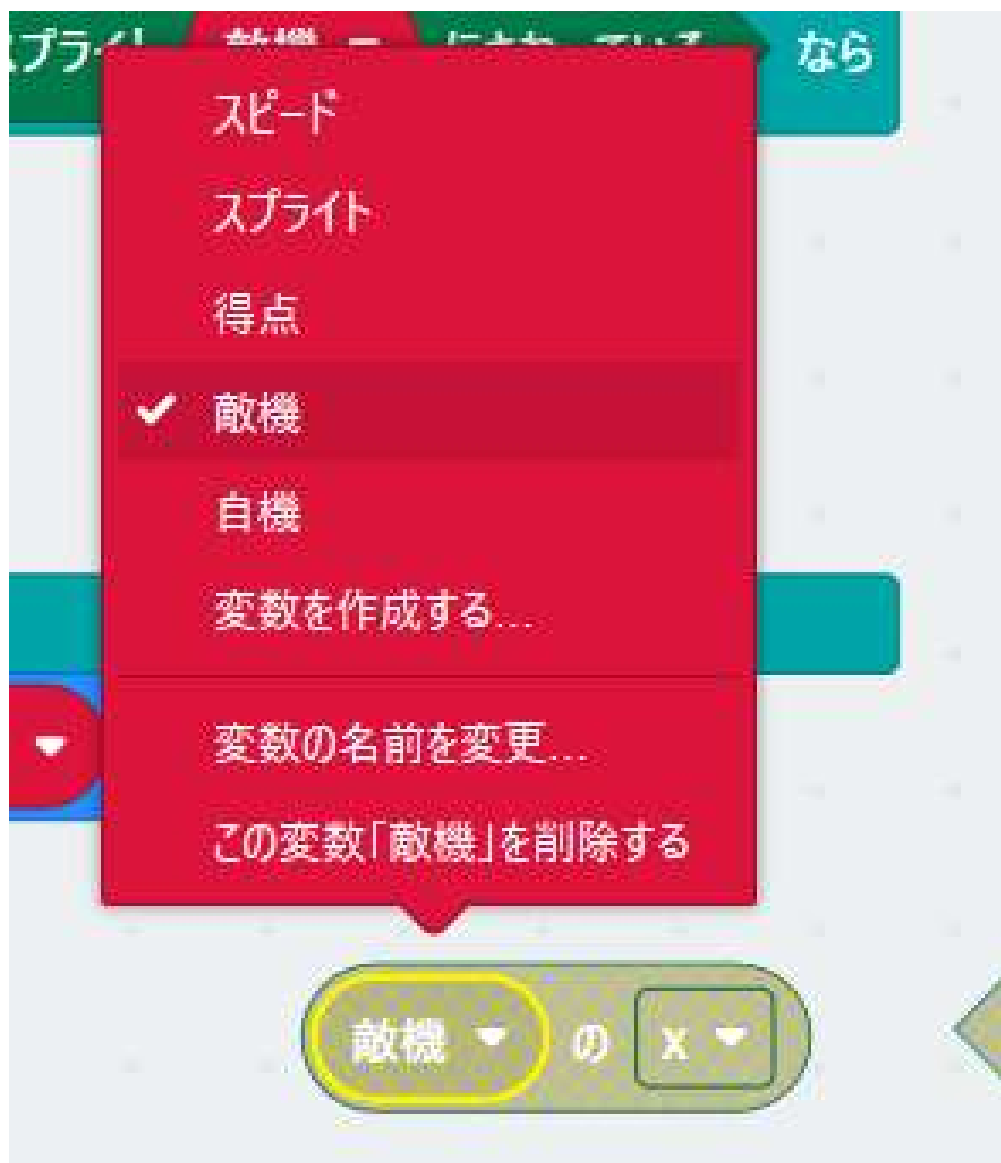
スプライト の X

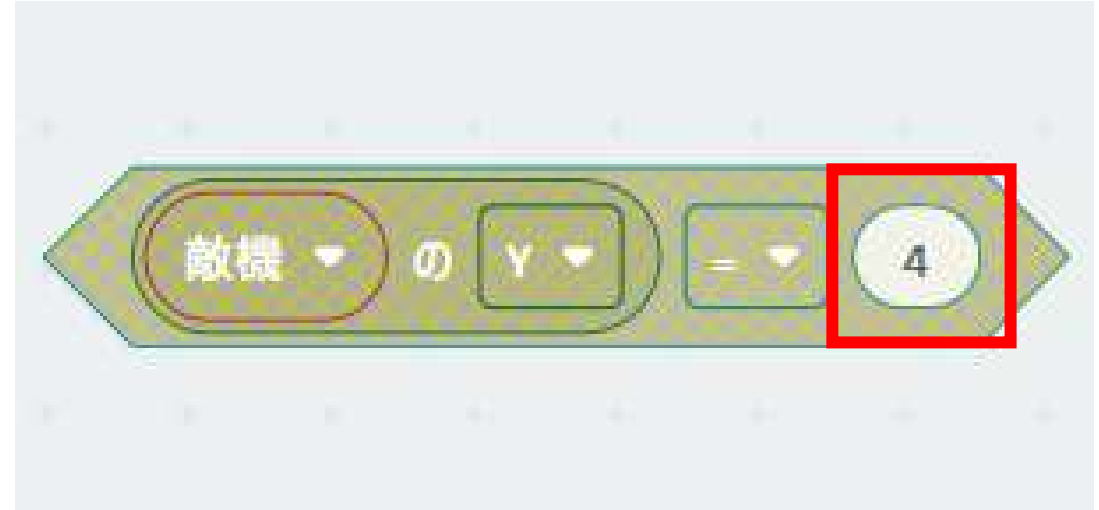
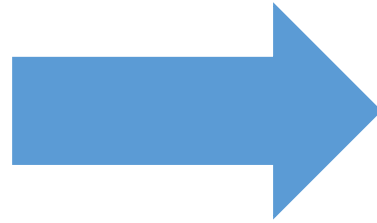
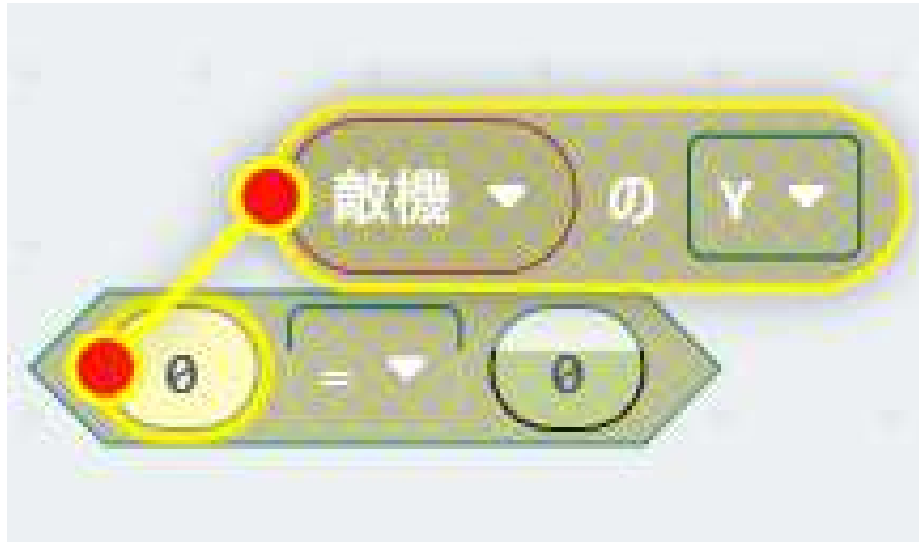


「論理」から「0=0」を右側に持っていきこう

The image shows the Scratch block palette on the left and a workspace on the right. The palette is organized into categories: 入力 (Input), 音楽 (Sound), LED, 無線 (Wireless), ループ (Loops), 論理 (Logic), 変数 (Variables), 計算 (Math), 高度なブロック (Advanced Blocks), 関数 (Functions), 配列 (Arrays), 文字列 (Text), and ゲーム (Games). The 'Logic' category is selected, showing various logical blocks. A red box highlights the '0 = 0' block in the 'Logic' category. In the workspace, a script is visible with several blocks. A red box highlights the '0 = 0' block in the workspace, which is being moved from the palette. The workspace also contains other blocks like '一時停止 (ミリ秒)' (Wait for milliseconds), '敵機' (Enemy), 'スピード' (Speed), and 'X' coordinate blocks.

# 「スプライトのX」ブロックを「敵機のY」に変更しよう



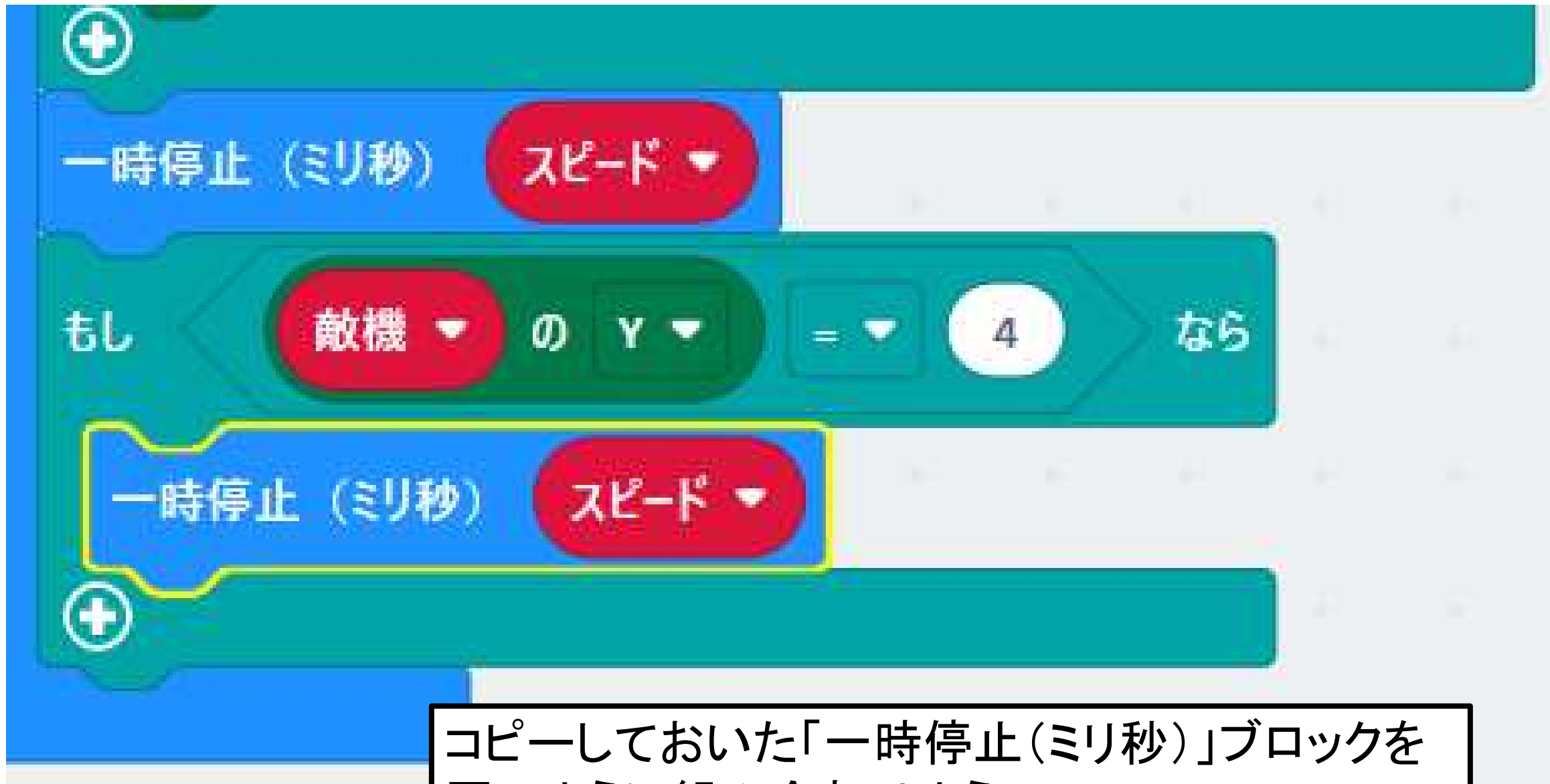


「敵機のY」と「0=0」ブロックを合体させよう

右の数字を4に変更して、「敵機のY=4」にしよう



合体させたブロックを「もし真なら」ブロックに  
組み合わせよう



コピーしておいた「一時停止 (ミリ秒)」ブロックを  
図のように組み合わせよう



ゲーム

無線

ループ

論理

変数

計算

高度なブロック

関数

配列

文字列

ゲーム

その他

制御

Sprite を削除

Sprite が削除済み

Sprite を 1 ドット進める

Sprite 方向転換 右 に 45 °

Sprite の x を 1 だけ増やす

Sprite の x に 0 を設定する

Sprite が他のSprite にさわっている

敵機 の Y を 1 だけ増やす

敵機 の X を -1 から 1

もし 自機 が他のSprite 敵機

自機 を削除

点数を 得点 にする

ゲームオーバーにする

一時停止 (ミリ秒) スピード

もし 敵機 の Y = 4

一時停止 (ミリ秒) スピード

Sprite を削除

「ゲーム」から「Sprite を削除」を持ってきて、図のように組み合わせよう



「スプライトを削除」を「敵機を削除」に変更しよう

検索...



変数

「変数」から「変数 スプライトを1だけ増やす」を持ってきて、図のように組み合わせよう

LED

無線

ループ

論理

変数

計算

高度なブロック

関数

配列

文字列

スプライト

得点

敵機

自機

変数 スプライト を 0 にする

変数 スプライト を 1 だけ増やす

点数を 得点 にする

ゲームオーバーにする

停止 (ミリ秒) スピード

もし 敵機 の Y = 4 なら

一時停止 (ミリ秒) スピード

敵機 を削除

変数 スプライト を 1 だけ増やす



敵機 ▼ を削除

変数 得点 ▼ を 1 だけ増やす

スピード

スプライト

✓ 得点

敵機

自機

変数を作成する...

変数の名前を変更...

この変数「得点」を削除する

「スプライト」を「得点を1だけ増やす」  
に変更しよう

検索...

変数

「変数」から「変数 スプライトを0にする」を持ってきて、図のように組み合わせよう

LED

無線

ループ

論理

変数

計算

高度なブロック

関数

配列

スプライト

得点

敵機

自機

変数 スプライト を 0 にする

変数 スプライト を 1 だけ増

変数の値を、与えられた値と同じにします。

点数を 得点 にする

ゲームオーバーにする

一時停止 (ミリ秒) スピード

もし 敵機 の Y = 4 なら

一時停止 (ミリ秒) スピード

敵機 を削除

変数 得点 を 1 だけ増やす

変数 スプライト を 0 にする



点数を 得点 ▾ にする

ゲームオーバーにする

スピード

スプライト

得点

✓ 敵機

自機

変数を作成する...

変数の名前を変更...

この変数「敵機」を削除する

「スプライト」を「敵機を0にする」に変更しよう

= ▾ 4 なら

ード ▾

だけ増やす

変数 敵機 ▾ を 0 にする



- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック
- 関数
- 配列
- 文字列
- ゲーム
- その他
- 画像
- 入出力端子
- シリアル通信

### ゲーム

スプライトを作成 x: 2 y: 2

スプライト を削除

スプライト が削除済み

スプライト を 1 ドット進める

スプライト 方向転換 右 に 45 °

スプライト の x を 1 だけ増やす

スプライト の x に 0 を設定する

スプライト の x

スプライト が他のスプライト に重なっている

点数を 得点 にする

ゲームオーバーにする

一時停止 (ミリ秒) スピード

もし 敵機 の y = 4 なら

一時停止 (ミリ秒) スピード

敵機 を削除

変数 得点 を 1 だけ増やす

変数 敵機 を スプライトを作成 x: 2 y: 2 にする

「ゲーム」から「スプライトを作成 x:2 y:2」を持ってきて、図のように組み合わせよう

「計算」から「0から10までの乱数」を持ってきて、図のように組み合わせよう

- 無線
- ループ
- 論理
- 変数
- 計算
- 高度なブロック
- 関数
- 配列
- 文字列
- ゲーム
- 画像
- 入出力端子
- シリアル通信
- 制御

0

0 を 1 で割ったあまり

0 と 0 のうち 小さい方

0 と 0 のうち 大きい方

0 の絶対値

平方根 0

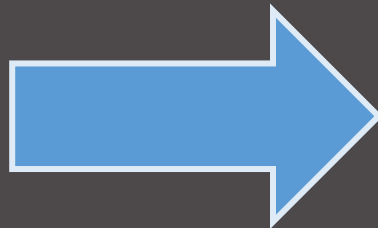
小数点以下四捨五入 0

0 から 10 までの乱数

0 を 0 以上 0 以下の範囲に制限

数値をマップする 0 元の下限 0 元の上限 1023 結果の下限 0 結果の上限 4

ランダムに真か偽に決める



敵機 を スプライトを作成 x: 0 から 10 までの乱数 y: 2

敵機 を削除

得点 を 1 だけ増やす

敵機 の Y = 4 なら

停止 (ミリ秒) スピード

上 (ミリ秒) スピード

オーバーにする

得点 にする



「0から10までの乱数」を「0から4までの乱数」に変更しよう

自機 ▼ を削除

コピーしておいた最後の「一時停止（ミリ秒）」を  
図のように組み合わせて完成！

実際に動かしてみよう！

もし 敵機 ▼ の Y ▼ = ▼ 4 なら

- 一時停止（ミリ秒） ▼ スピード ▼
- 敵機 ▼ を削除
- 変数 得点 ▼ を 1 だけ増やす
- 変数 敵機 ▼ を スプライトを作成 x: 0 から 4 までの乱数 y: 2 にする
- 一時停止（ミリ秒） ▼ スピード ▼

The script is a Scratch 'when green flag clicked' event. It contains a loop: 'if enemy's Y coordinate equals 4, then: pause for a certain number of milliseconds (with a speed dropdown), delete the enemy, increase the score by 1, create a new enemy sprite at a random X position between 0 and 4 and Y position 2, and then pause for another certain number of milliseconds (with a speed dropdown). The final pause block is highlighted with an orange box.