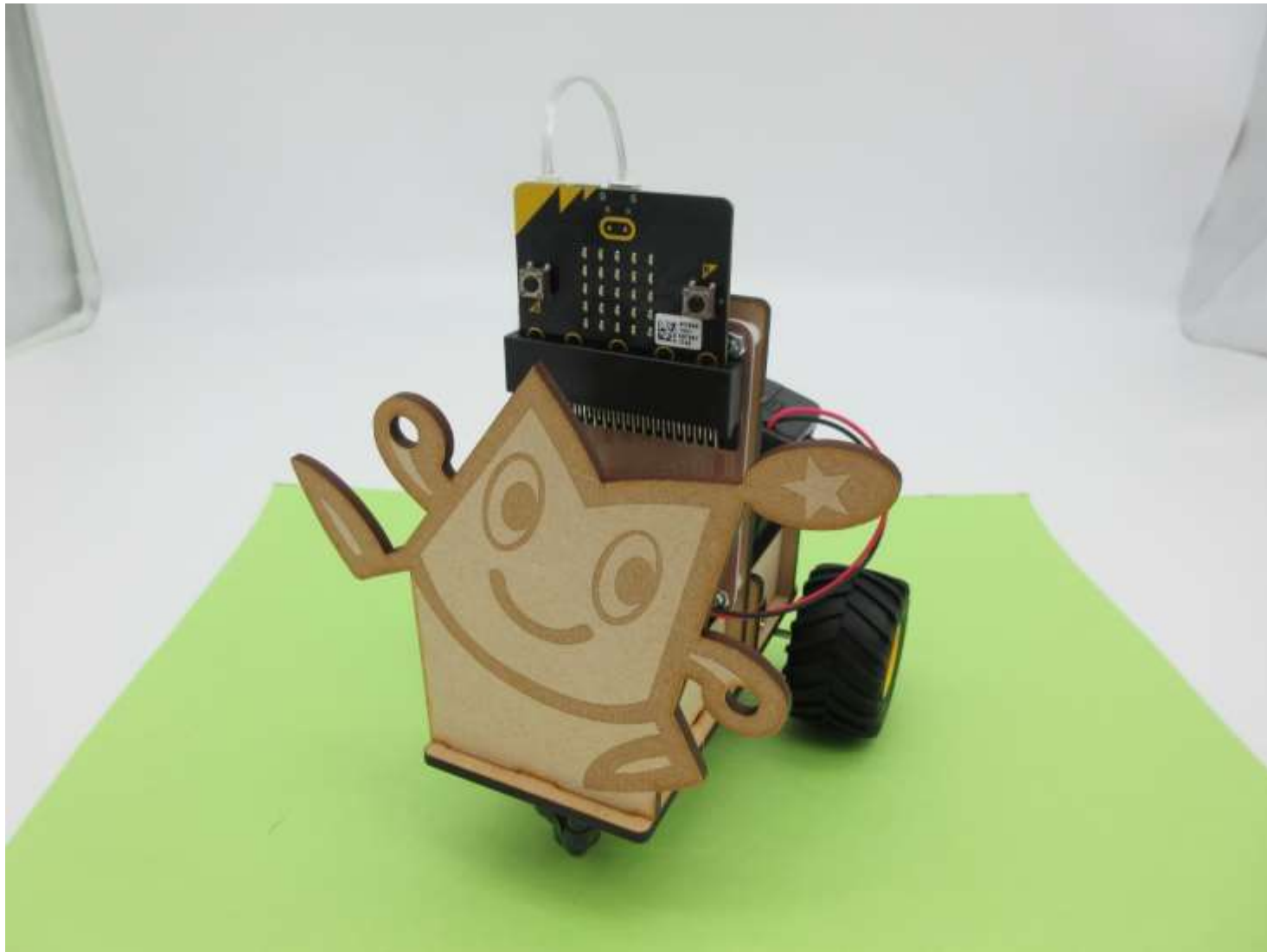


動くロボットプログラミング教室 ～マイクロビットで動く車を作ろう～

Android版







ロボットの5大要素

コンピュータ



センサ



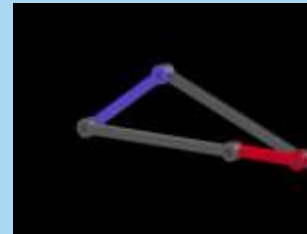
アクチュエータ



エネルギー源



機構(きこう)



ロボットの5大要素



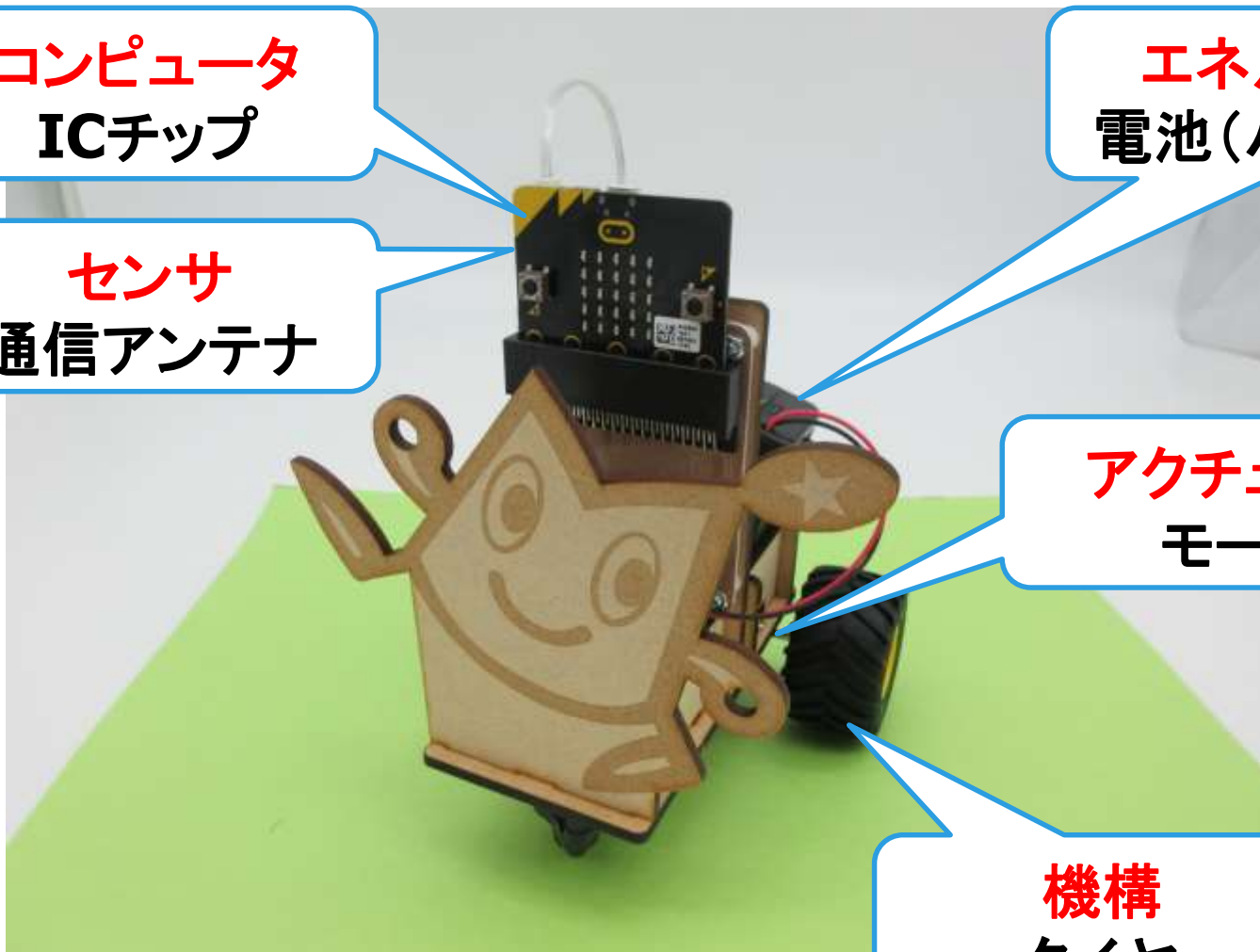
コンピュータ
ICチップ

センサ
通信アンテナ

エネルギー源
電池(バッテリー)

アクチュエータ
モーター

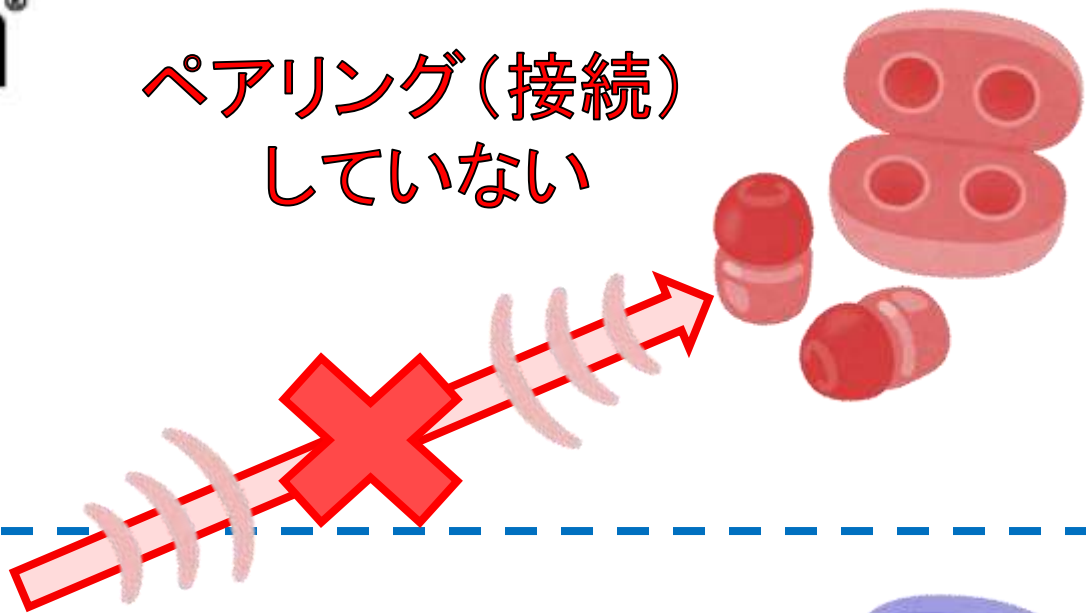
機構
タイヤ





ペアリング(接続)
していない

A

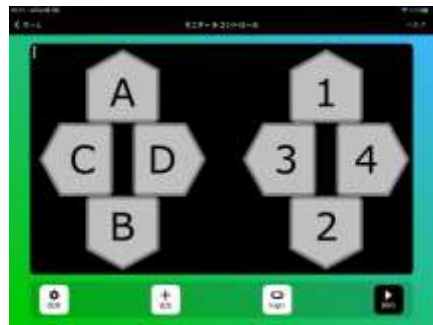


B



ペアリング(接続)済み

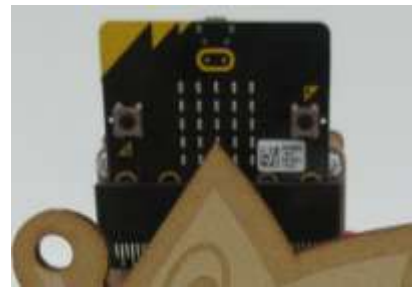
動作するための流れ



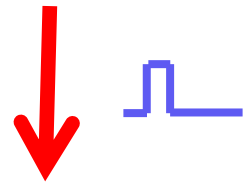
スマート
フォン



通信
アンテナ



マイクロビット



モータ
ドライバ



モータ



走る





ケガをしないように

- 集中する
- つかれたら休む
- まわりをかたづける
- あわてない



<https://www.irasutoya.com/>



作成のポイント

- 工具を正しく使う



- 説明書の写真や図をしっかりと見る

- わからないところはよく見る



<https://www.irasutoya.com/>



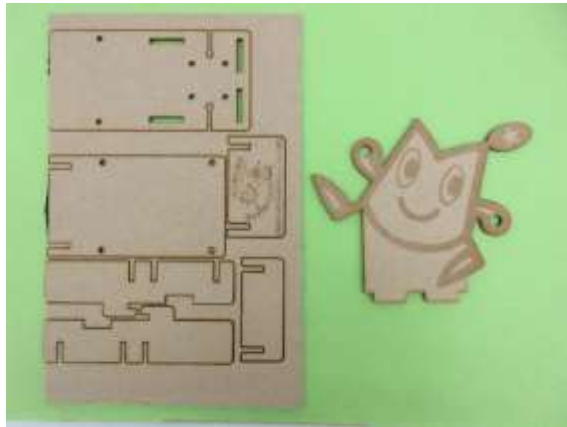
■ 道具の確認

- 紙皿
- ラジオペンチ
- ドライバー
- えんぴつ





■ 全部あるか確認しよう



車体パーツセット



ギアボックス



キャスター



基板セット



Micro:bit セット



タイヤ



■ 全部あるか確認しよう



短いネジ
6本



長いネジ
4本



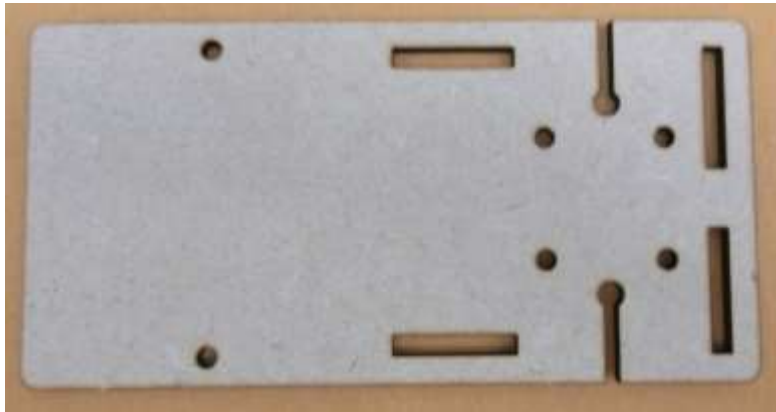
ナット
14個



スペーサー
4個



■ 用意するもの



土台パーツ



短いネジ
2本



ナット
2個



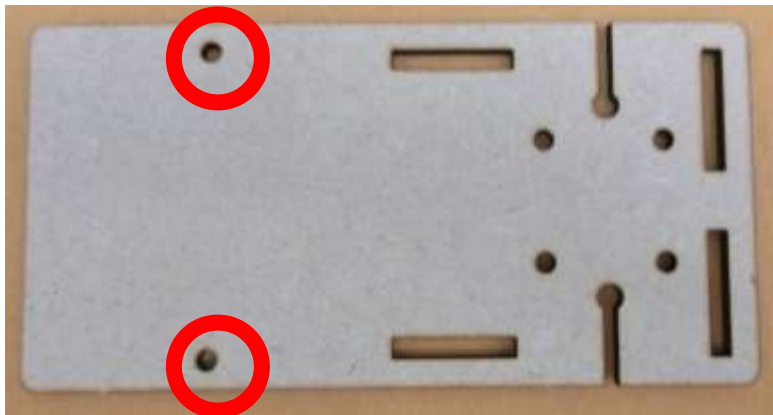
ギアボックス

①車体パーツセットから
土台を取り外す

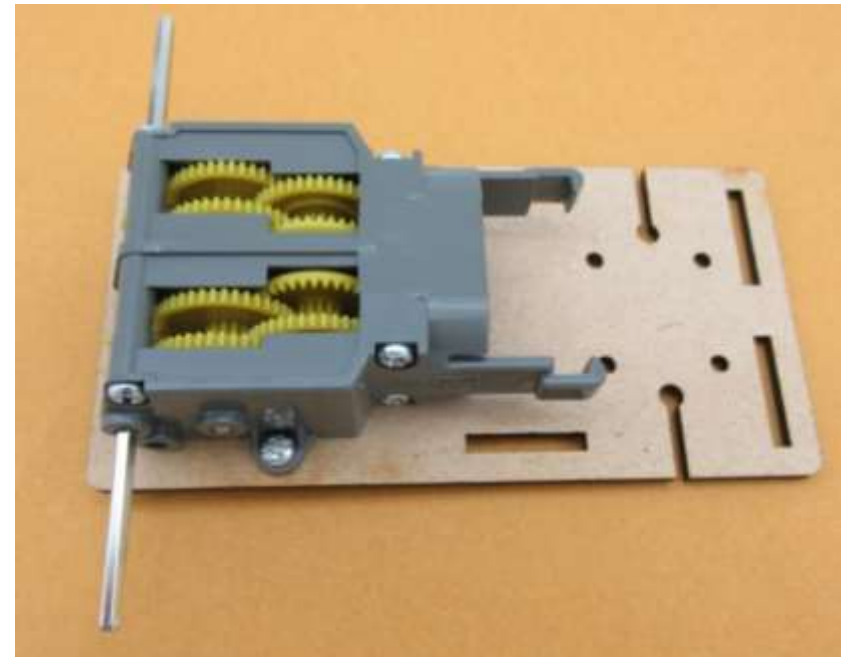
ねじの長さ注意



- 車体パーツセットから取り外した土台にギアボックスを取り付ける



丸の位置に短いねじで固定するよ



ギアボックス側からねじをさしこみ
うら側をナットでとめるよ



■ 用意するもの



基板セット



仕切り大



長いネジ



スペーサー
各 4 個

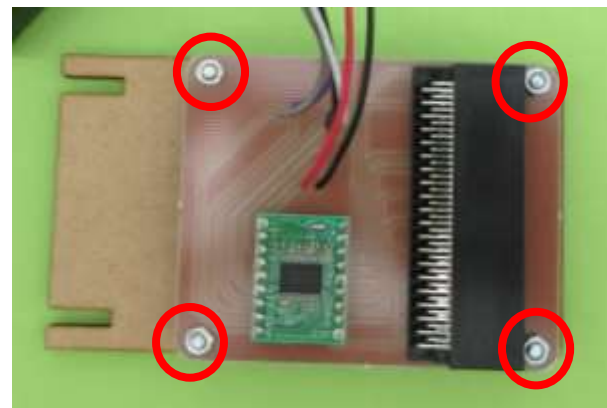


ナット
8 個

- 仕切り大に基板セットを
ねじとナットで取り付ける(4か所)

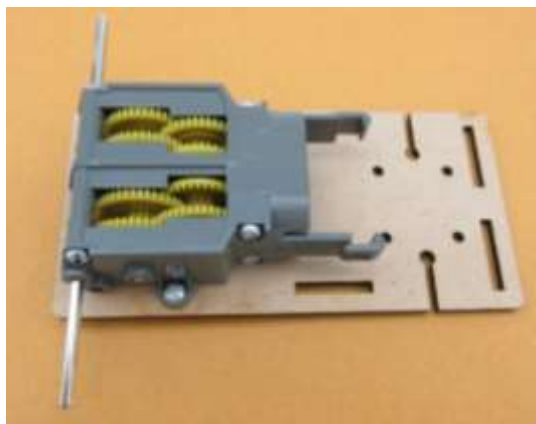


- ① **スペーサー**をナットでとめる
- ② **基盤**をナットでとめる

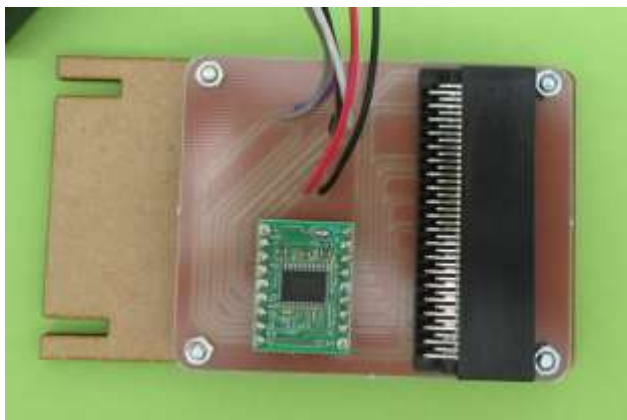




■ 用意するもの

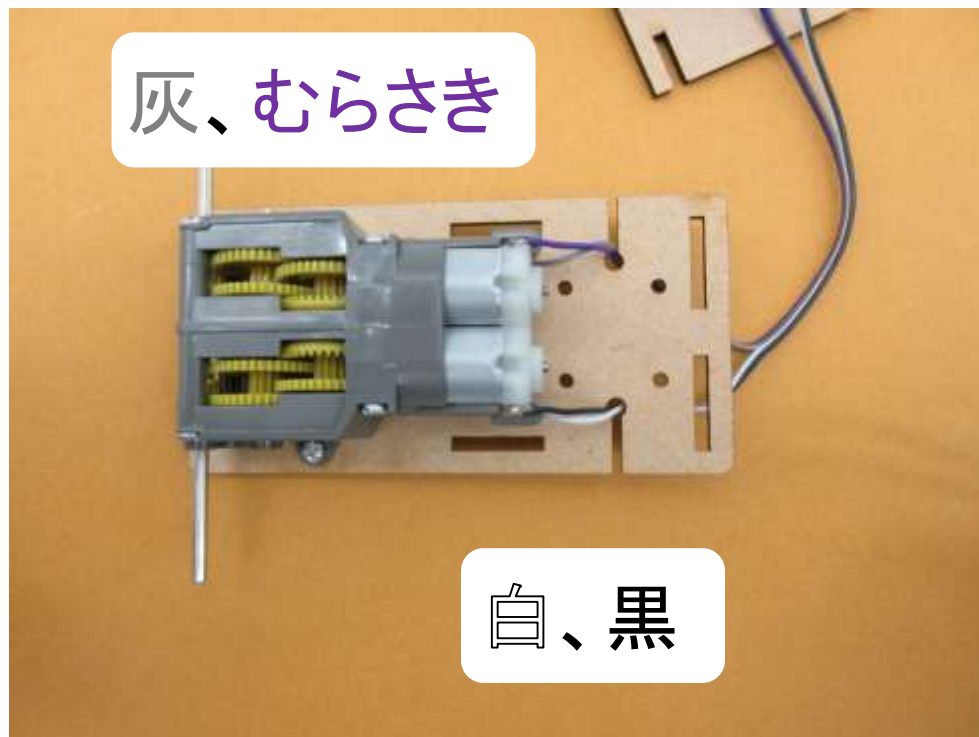


ギアボックス



基板セット

基板についているモーターを土台の切れ込みを通してギヤボックスにさしこむ線の色に注意しよう



灰、むらさき

白、黒



■ 用意するもの



キャスター



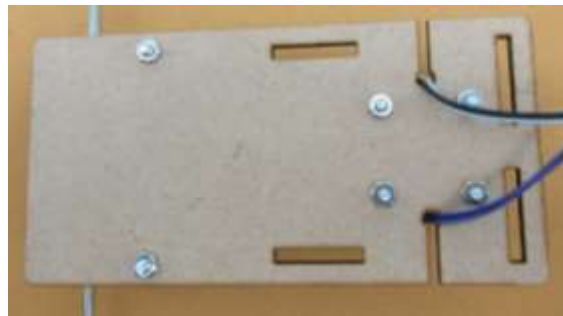
短いねじ



ナット

各 4 個

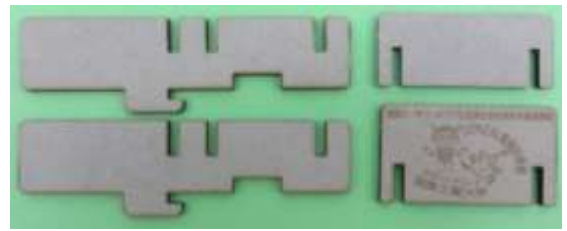
■ 土台にキャスターを取り付ける



裏側をナットで
とめよう

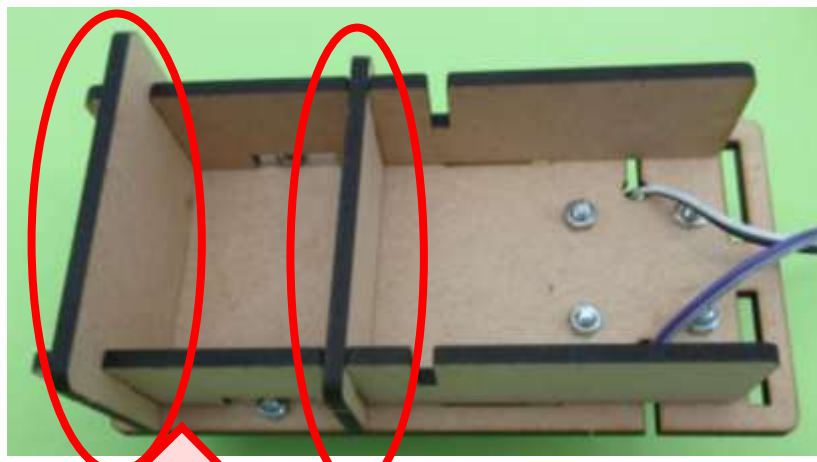


■ 用意するもの

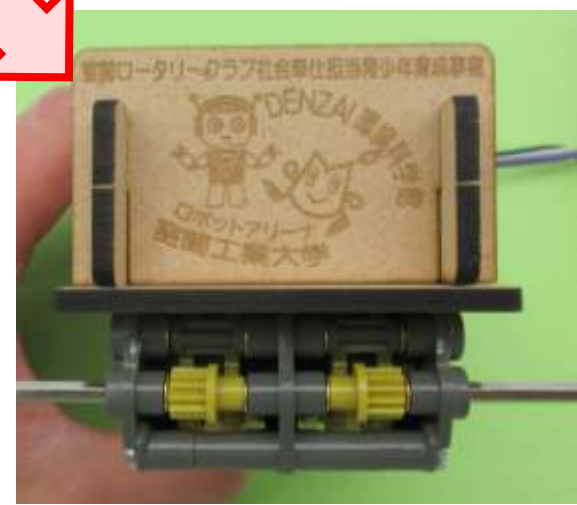
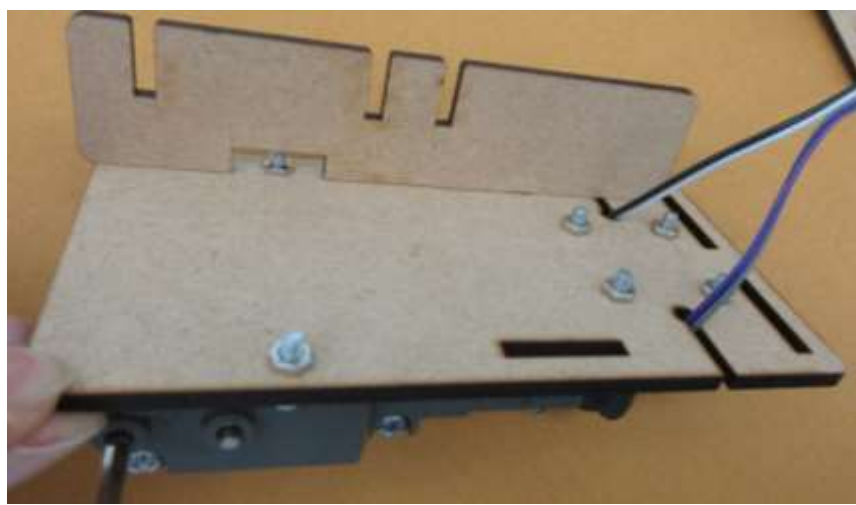


サイド 仕切り小

②仕切り小をさしこむ

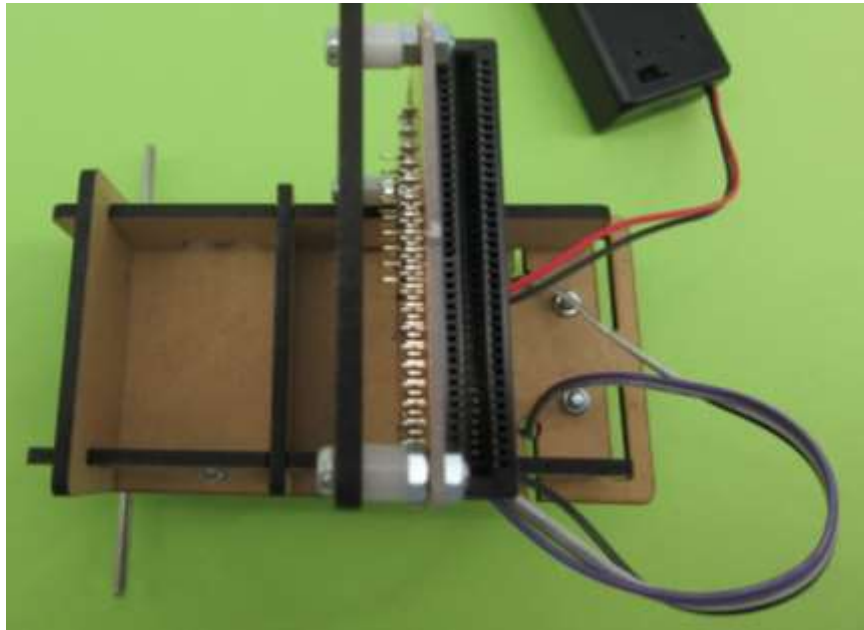


①土台にさしこみスライドする

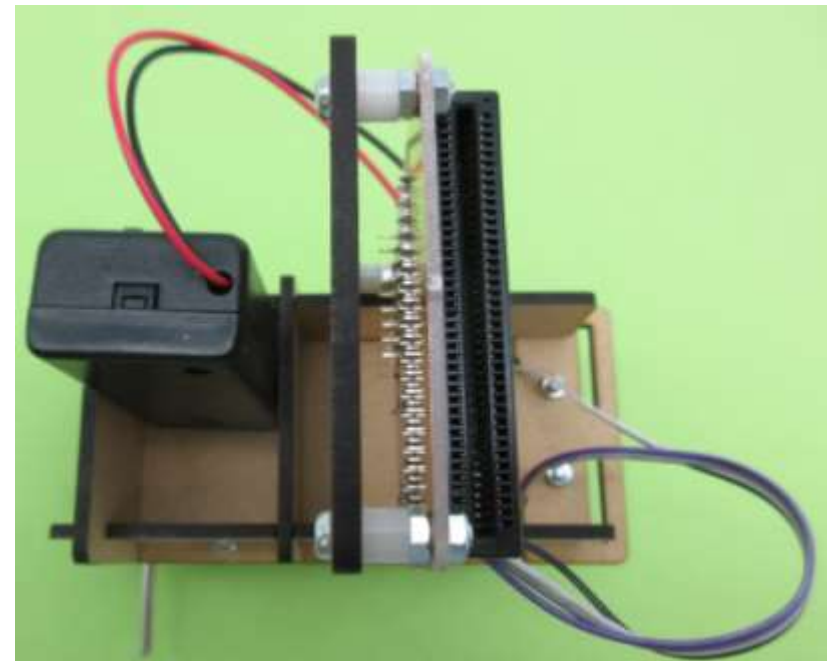




- 組み立てた土台に
基板をさしこむ



- 電池ボックスは左の
スペースにしまう





■ 用意するもの



タイヤ

タイヤを取り付けよう
しっかり奥までさしこもう





- 車体の前の穴にムロピョンを差し込む



コンピュータは
そのままでは動かないよ！

プログラムを書きこんで
ロボットが動くようにしよう

組み立て完成！！



10分



きゅうけい



micro:bit プログラム編





■ 道具の確認

□ パソコン

□ インターネット

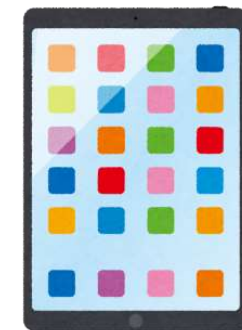
□ Webブラウザ

Google Chrome など

□ micro:bit

□ USBケーブル

□ スマホ／タブレット

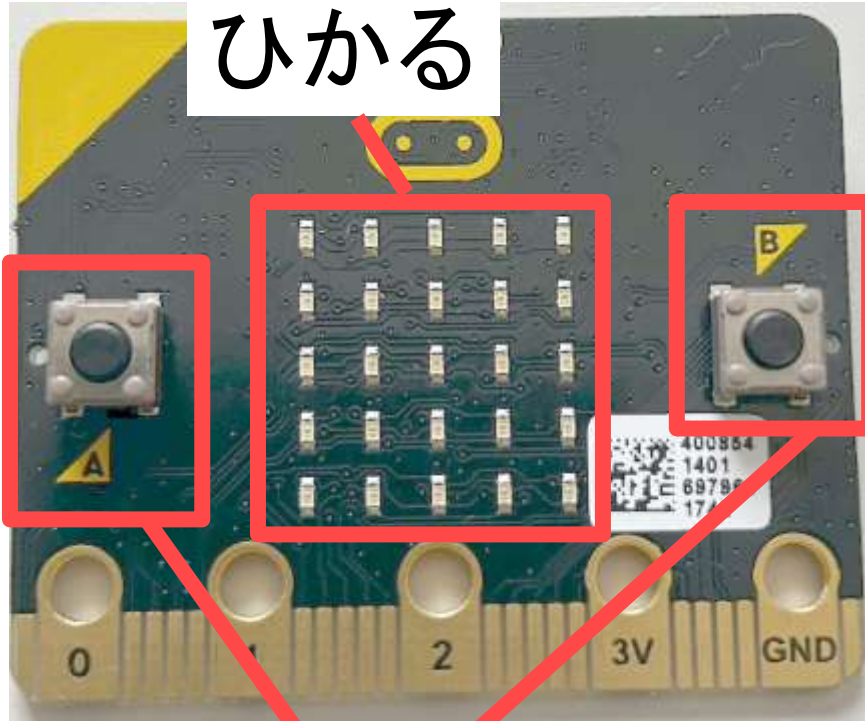




マイクロビット

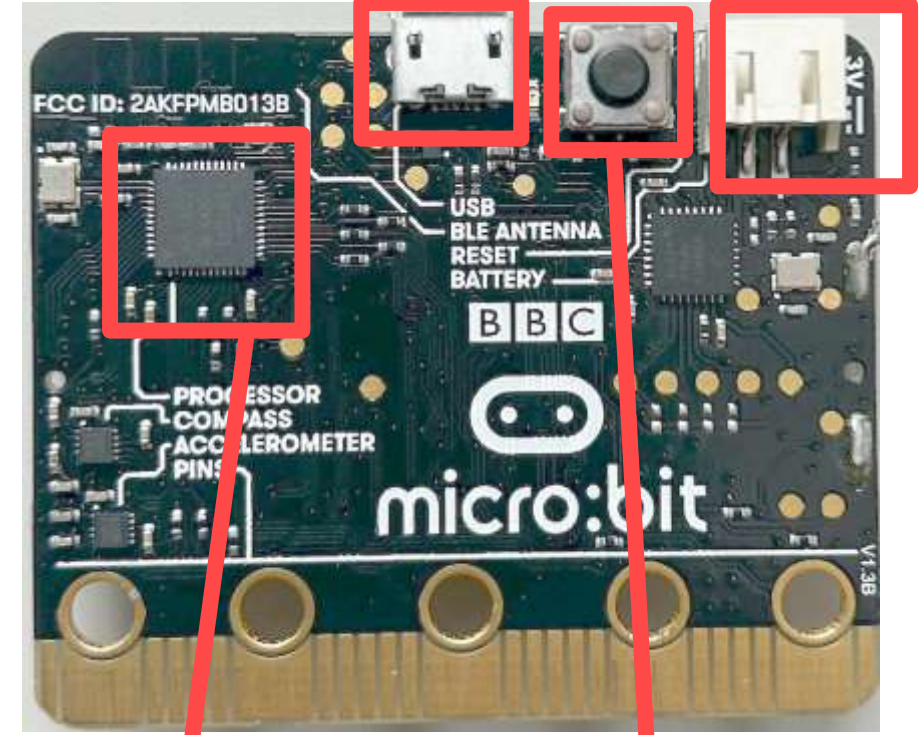
PC

でんち



ひかる

ボタン

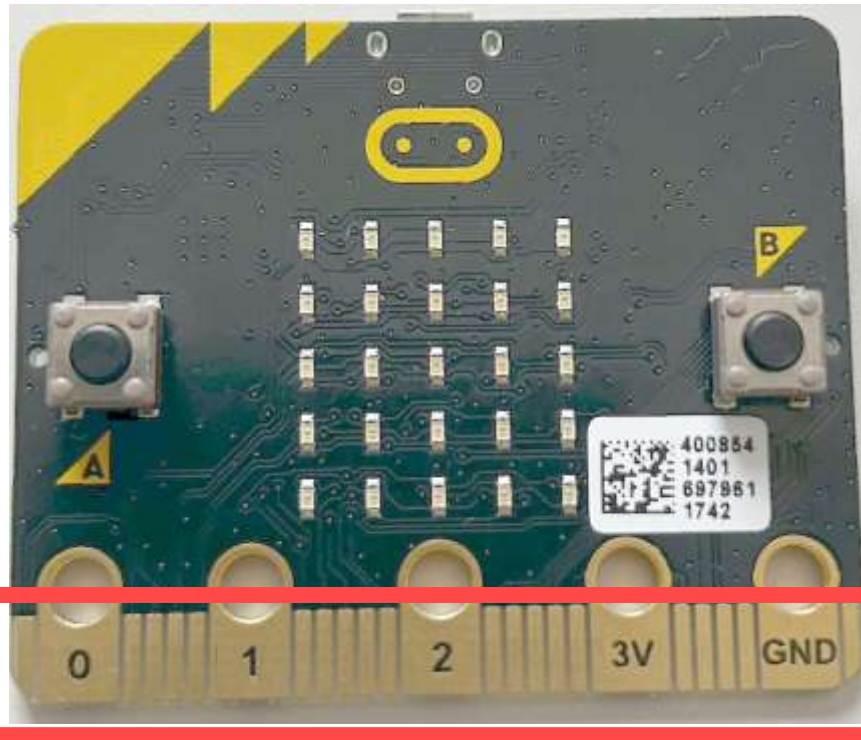


マイコン

リセット



マイクロビット



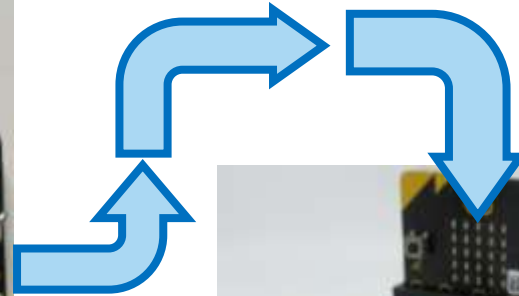
ロボットへ



パソコン



<https://www.irasutoya.com/>より転載



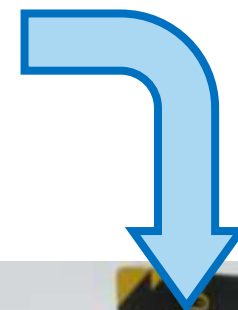
ケーブル: ユーエスビー (USB)



スマホ
タブレット



<https://www.irasutoya.com/>より転載

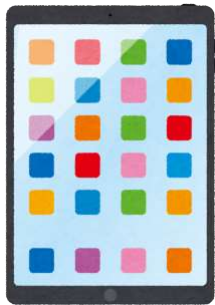


むせん: ブルートゥース (Bluetooth)



<https://www.irasutoya.com/>より転載

メイクコード
MakeCode



<https://www.irasutoya.com/>より転載

マイクロビット
micro:bit



<https://www.apple.com/jp/app-store/>



<https://play.google.com/>



シミュレータ

ぶひん

プログラム



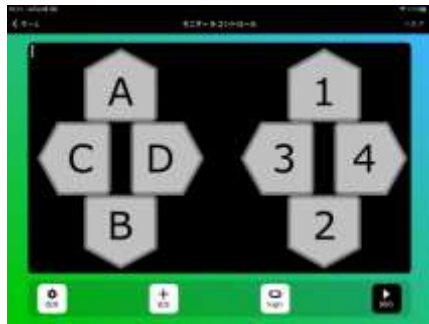
ぶひんを
くみあわせる

なまえ



プログラムのながれ

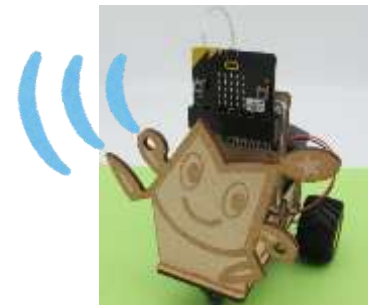
スマホからロボットをうごかす



スマート
フォン



めいれい



はしる

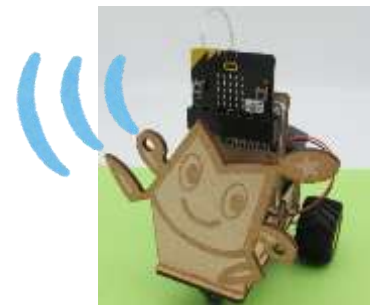
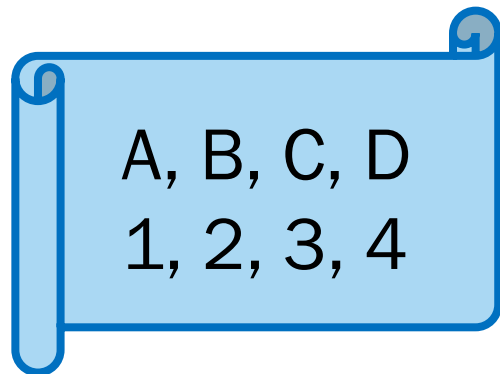
ひかる

- 「A」 → 「まえに, はしる」
- 「B」 → 「うしろに, はしる」
- ⋮
- 「1」 → 「1 と, ひかる」
- 「2」 → 「2 と, ひかる」
- ⋮



- 【1】 スマホから, めいれいを, うけとる
- 【2】 どのめいれいか, はんだんする
- 【3】 タイヤをまわす, LEDをひからせる

8このめいれい

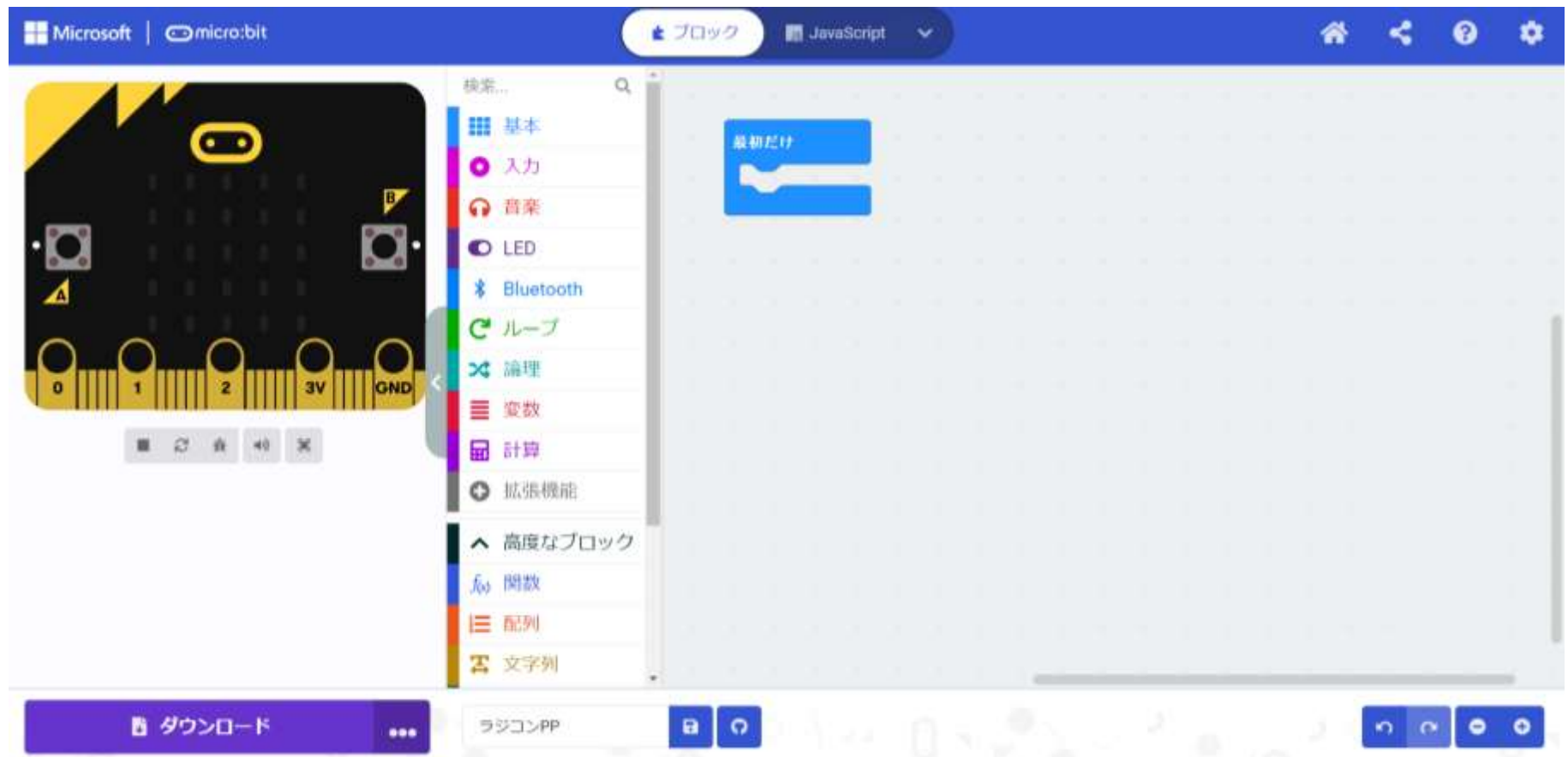


はしる

ひかる



プログラムのソフトをひらく





- 【1】 スマホから, めいれいを, うけとる (8)
- 【2】 どのめいれいか, はんだんする (12)
- 【3】 タイヤをまわす, LEDをひからせる (17)



【1】 スマホから, めいれいを, うけとる (1/8)

①Bluetooth → ②その他タブ →③ [Bluetooth UARTサービス]

The screenshot shows the Scratch programming environment. On the left is the 'Component Palette' with categories: 基本 (Basic), 入力 (Input), 音楽 (Music), LED, Bluetooth, その他 (Other), 変数 (Variables), 計算 (Math), and 拡張機能 (Extensions). The 'Bluetooth' category is selected, and the 'Bluetooth UARTサービス' block is highlighted. A red arrow points from this block to the 'Program' area on the right, where it is being dragged and dropped into the '最初だけ' (At the start) area. The 'Program' area contains several 'Bluetooth UART' blocks. A red circle with the number '1' is around the 'Bluetooth' category in the palette, a red circle with '2' is around the 'Bluetooth UARTサービス' block in the palette, and a red circle with '3' is around the 'Bluetooth UART 名前と値を書き出す' block in the program area. A white box with the text 'クリック' (Click) is positioned over the 'Bluetooth UART サービス' block in the program area. Another white box with the text 'ドラッグ & ドロップ' (Drag & Drop) is positioned over the 'Bluetooth UART サービス' block in the program area. A white box with the text 'Bluetooth UART サービス' is positioned over the 'Bluetooth UART サービス' block in the program area. A white box with the text 'Bluetooth UART サービス' is positioned over the 'Bluetooth UART サービス' block in the program area.



【1】 スマホから、めいれいを、うけとる (2/8)



③ “last data” と入力。

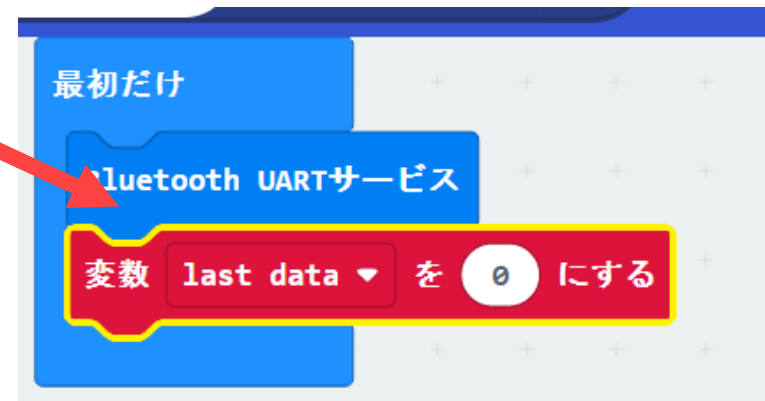




【1】 スマホから、めいれいを、うけとる (3/8)



ブロックを[]
のブロックにいれる



ドラッグ &
ドロップ



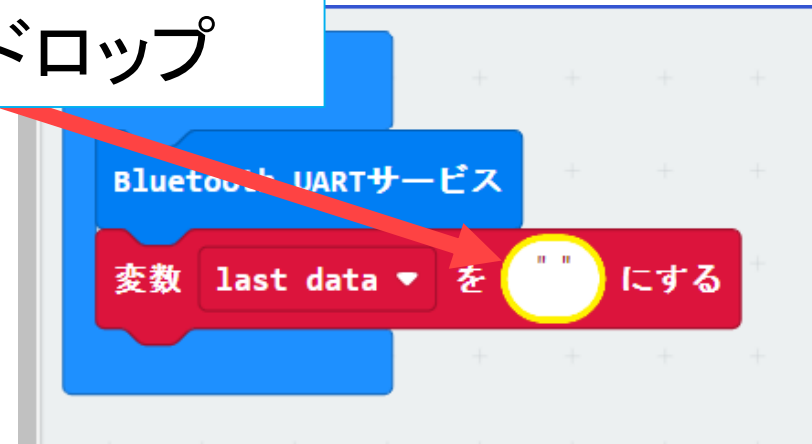
【1】 スマホから、めいれいを、うけとる (4/8)

①高度なブロック→②文字列→ ③ []



ドラッグ &
ドロップ

3



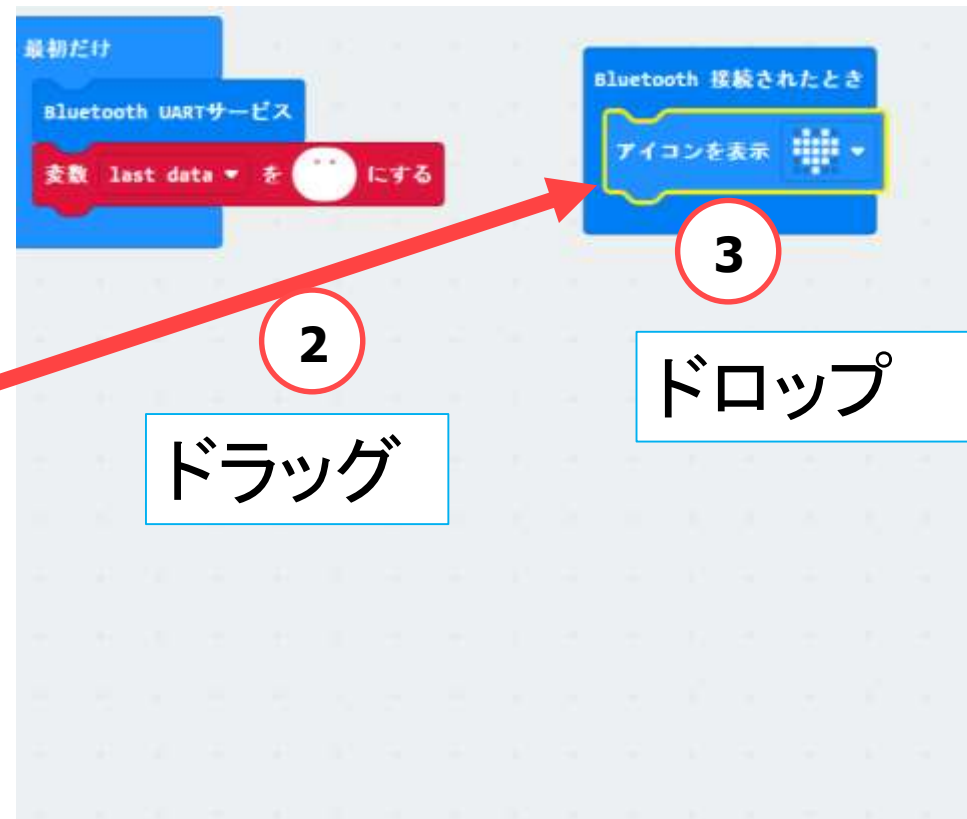


【1】 スマホから、めいれいを、うけとる (5/8)





【1】 スマホから、めいれいを、うけとる (6/8)



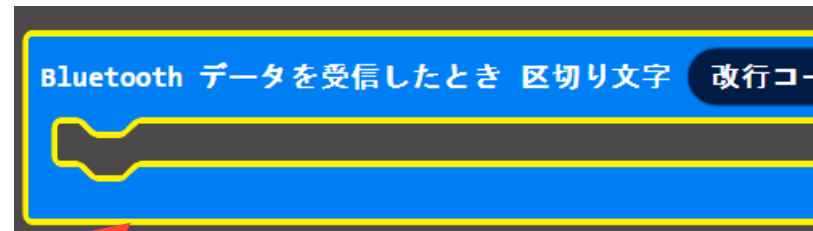


【1】 スマホから、めいれいを、うけとる (7/8)



クリック

1



2

ドラッグ &
ドロップ



【1】 スマホから、めいれいを、うけとる (8/8)

①改行コード → ② “シャープ「#」”を選択



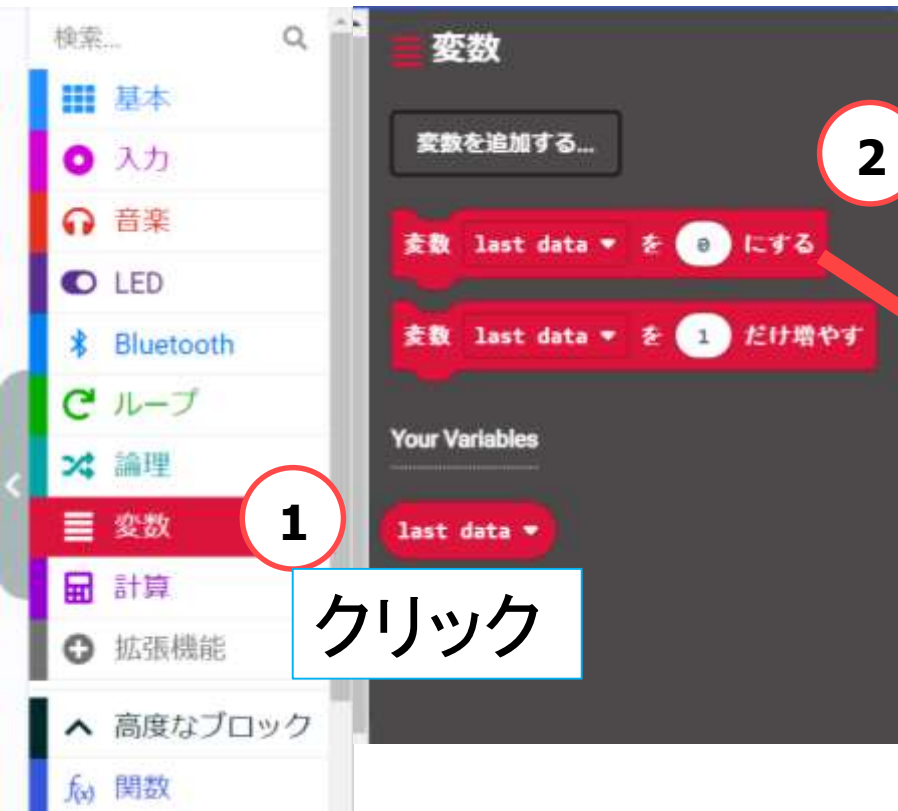
改行コードを“シャープ「#」”



- 【1】 スマホから, めいれいを, うけとる (8)
- 【2】 どのめいれいか, はんだんする (16)
- 【3】 タイヤをまわす, LEDをひからせる (17)

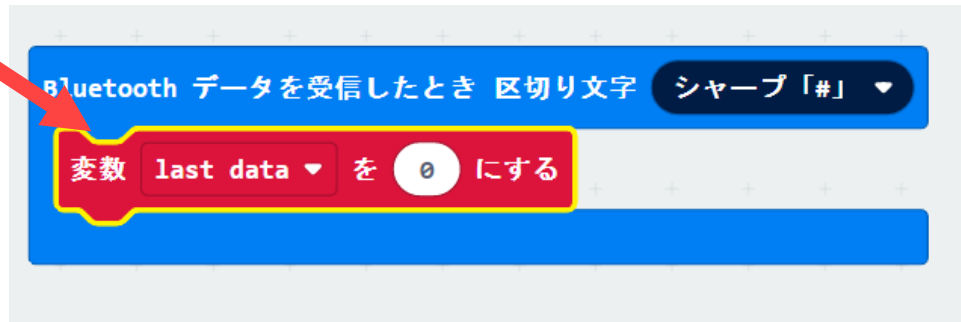


【2】 どのめいれいか, はんだんする (1/16)



クリック

ドラッグ & ドロップ





【2】 どのめいれいか, はんだんする (2/16)

- ①Bluetooth → ②その他タブ
→③[Bluetooth UART つぎのいずれかの文字の手前まで読み取る 改行コード] →④ “シャープ「#」”を選択

クリック

1

2

クリック

3

ドラッグ & ドロップ

4

クリック



【2】 どのめいれいか, はんだんする (3/16)

① [変数 last data を] → ② 変数を作成する…
→ ③ "data" と入力 → ④ [OK ✓]

The screenshot shows the Scratch IDE interface. At the top, a blue bar contains the text "Bluetooth UART つぎのいずれかの文字の手前まで読み取る" and a dropdown menu set to "シャープ「#」". Below this, a red bar contains a dropdown menu with "変数 last data" selected. A white box with the number "1" and the word "クリック" (Click) points to this dropdown. Below the red bar, a red menu is open, showing options: "last data" (checked), "変数を作成する..." (Create new variable...), "変数の名前を変更..." (Rename variable...), and "この変数「last data」を削除する" (Delete this variable). A white box with the number "2" and the word "クリック" (Click) points to the "変数を作成する..." option. To the right, a dialog box titled "作成する変数の名前:" (Name of the variable to create:) is shown. The input field contains the text "data". A white box with the number "3" and the word "クリック" (Click) points to the input field. At the bottom right, a green button with "OK ✓" is visible. A white box with the number "4" and the word "クリック" (Click) points to this button.

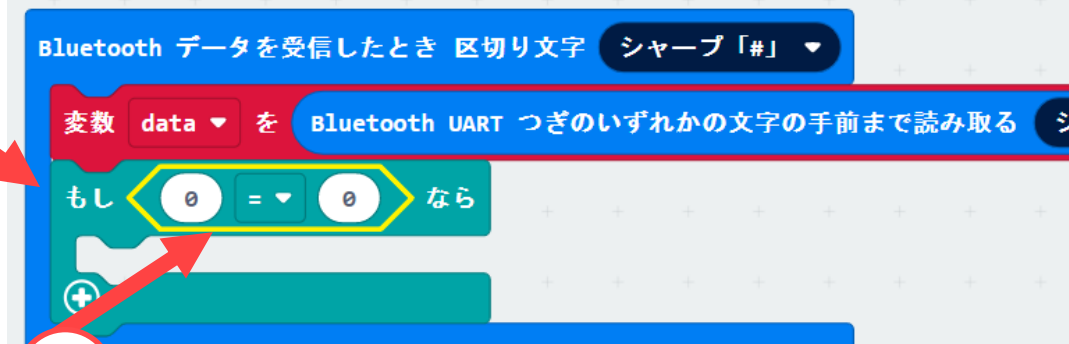


【2】 どのめいれいか, はんだんする (4/16)

①論理タブ→ ② [] → ③ []



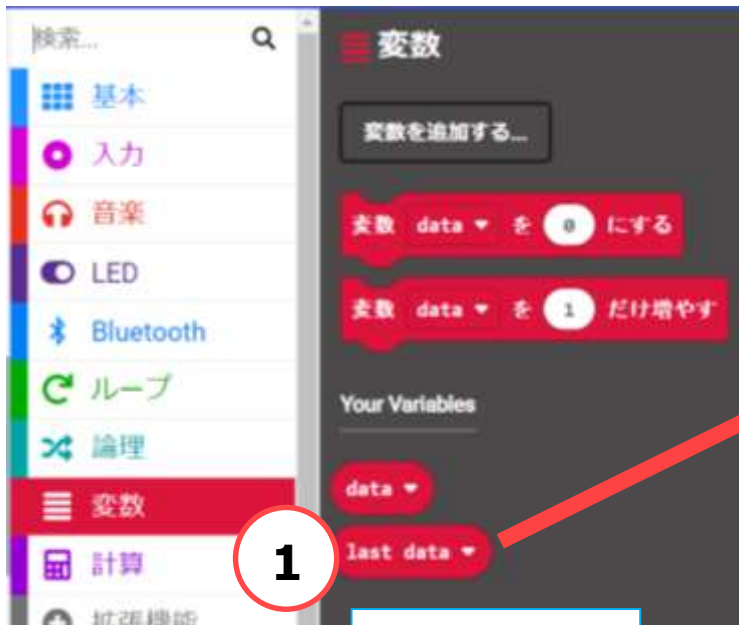
ドラッグ &
ドロップ



ドラッグ &
ドロップ

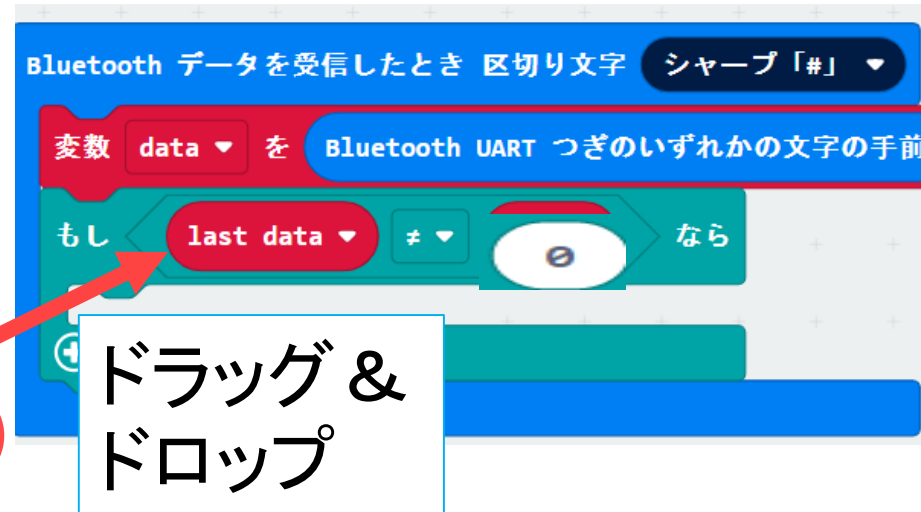


【2】 どのめいれいか, はんだんする (5/16)



1

クリック



2

ドラッグ & ドロップ

ポイント!

ガイド ●
がでた
ところでドロップ





【2】 どのめいれいか, はんだんする (6/16)

④ \neq にする

検索...

変数

変数を追加する...

変数 data を 0 にする

変数 data を 1 だけ増やす

Bluetooth データを受信したとき 区切り文字 シャープ「#」

変数 data を Bluetooth UART つぎのいずれかの文字の手前

もし last data \neq data なら

クリック

1

2

ドラッグ & ドロップ

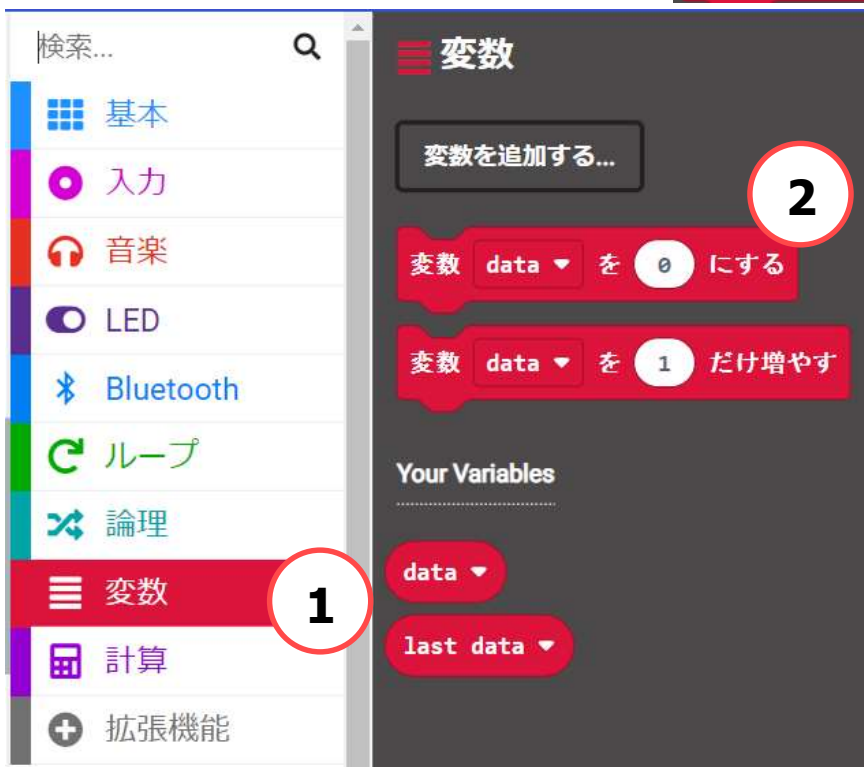
3



【2】 どのめいれいか, はんだんする (7/16)

①変数タブ→②[変数 data ▼ を 0 にする] →

③[変数 data ▼]を[変数 last data ▼]





【2】 どのめいれいか, はんだんする (8/16)

The image shows two screenshots of the Scratch programming environment. The top screenshot shows a script block with a 'data' block selected, and a context menu is open over it. The bottom screenshot shows the 'data' block pasted into a 'set variable to' block.

- 1 みぎクリック (Right-click)
- 2 クリック (Click)
- 3 コピーができる (Copy is possible)
- 4 ドラッグ (Drag)
- 5 ドロップ (Drop)



【2】 どのめいれいか, はんだんする (9/16)





【2】 どのめいれいか, はんだんする (10/16)

1 クリック

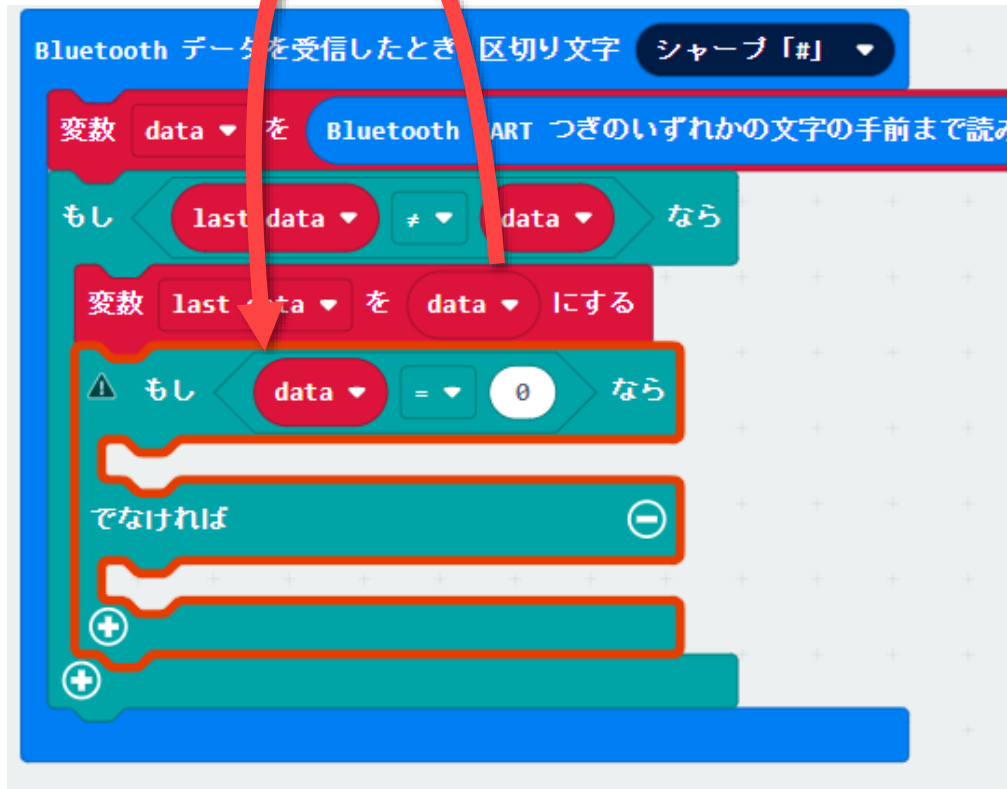
2 ドラッグ

3 ドロップ



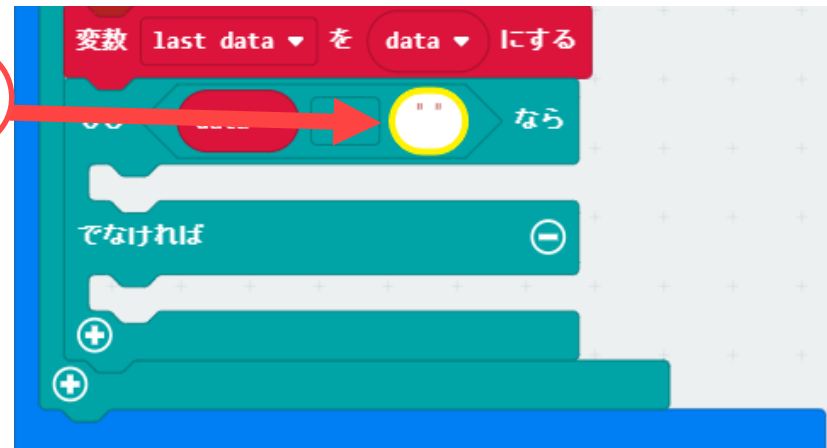
【2】 どのめいれいか, はんだんする (11/16)

- ① (5)とおなじほうほうで,
コピーして, ドラッグ&ドロップ





【2】 どのめいれいか, はんだんする (12/16)



④ “f” と入力



【2】 どのめいれいか, はんだんする (13/16)

Bluetooth データを受信したとき 区切り文字 シャープ「#」

変数 data を Bluetooth UART つぎのいずれかの文字の手前ま

もし last data ≠ data なら

変数 last data を data にする

もし data = "f" なら

でなければ

1

十を1かい
クリックすると

値が真の場合には、最初のかたまりを動かします。そうでない場合、かたまりのブロックを動かします。

Bluetooth データを受信したとき 区切り文字 シャープ「#」

変数 data を Bluetooth UART つぎのいずれかの文字

もし last data ≠ data なら

変数 last data を data にする

もし data = "f" なら

でなければもし なら

でなければ

2

「もし～なら」が1つ, 増える



【2】 どのめいれいか, はんだんする (14/16)

ポイント!

めいれいは
6こある

A, B, C, D
1, 2,



「もし～なら」を6こにする. あと, 4かい, クリックする



【2】 どのめいれいか, はんだんする (15/16)

変数 data を Bluetooth UART つぎのいずれかの文字

もし last data が data と異なるならば

変数 last data を data にする

もし data が "f" ならば

でなければもし

でなければもし

でなければもし

でなければもし

でなければもし

でなければ

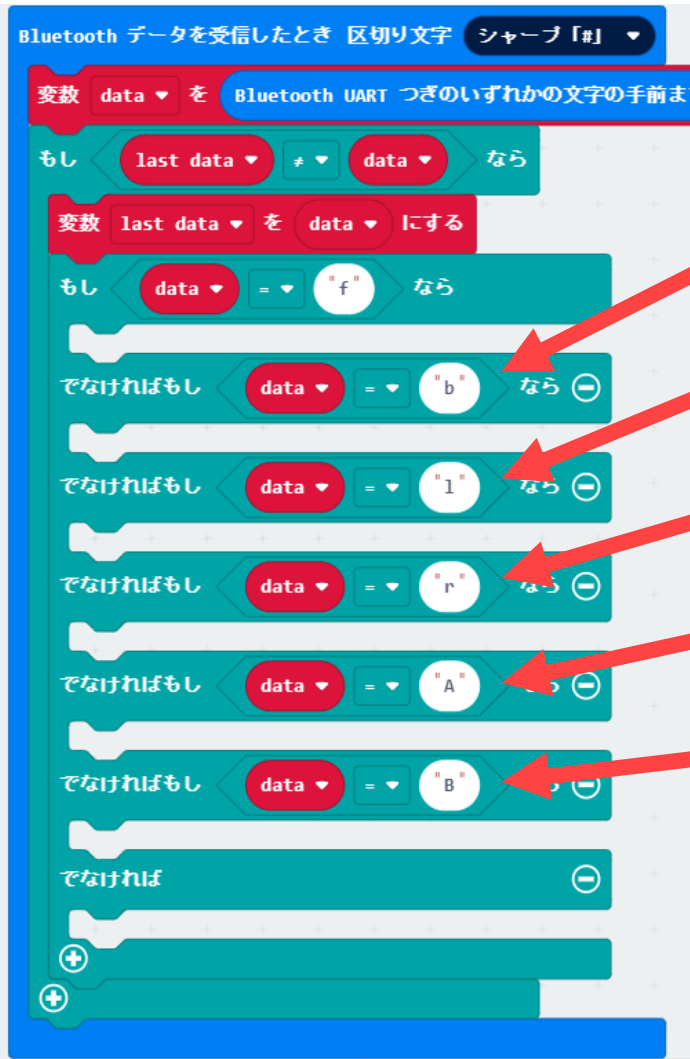
複製する
コメントを追加する
ブロックを削除する
ヘルプ

イベントの値 = 1

- 1 みぎクリック
- 2 クリック
- 3 コピーができる
- 4 ドラッグ
- 5 ドロップ
- 6 ぜんぶ, おなじに



【2】 どのめいれいか, はんだんする (16/16)



① もじをかえる

b にする

l(エル) にする

r にする

A にする

B にする



- 【1】 スマホから, めいれいを, うけとる (8)
- 【2】 どのめいれいか, はんだんする (16)
- 【3】 タイヤをまわす, LEDをひからせる (17)



【3】 タイヤをまわす, LEDをひからせる (1/17)

Bluetooth データを受信したとき 区切り文字 シャープ「#」

変数 data を Bluetooth UART つぎのいずれかの文字の手

もし last data ≠ data なら

変数 last data を data にする

もし data = "f" なら

でなければもし data = "b" なら

でなければもし data = "l" なら

でなければもし data = "r" なら

でなければもし data = "A" なら

でなければもし data = "B" なら

でなければ

まえ

うしろ

ひだり

みぎ

タイヤをまわしてすすむ

もし data = "f" なら

文字列を表示 "F"

デジタルで出力する 端子 P8 値 1

デジタルで出力する 端子 P1 値 0

デジタルで出力する 端子 P12 値 1

デジタルで出力する 端子 P2 値 0

アナログで出力する 端子 P0 値 300

アナログで出力する 端子 P16 (出力のみ) 値 300

ポイント!
ブロックのかたまりをコピー



【3】 タイヤをまわす, LEDをひからせる (2/17)



2 ドラッグ

3 ドロップ

このなかに, ブロックの
かたまりをつくる



【3】 タイヤをまわす, LEDをひからせる (3/17)

The image shows the Scratch programming environment. On the left is a vertical menu with categories: 基本 (Basic), 入力 (Input), 音楽 (Sound), LED, Bluetooth, ループ (Loops), 論理 (Logic), 変数 (Variables), 計算 (Math), 拡張機能 (Extensions), and 高度なブロック (Advanced Blocks). The 'Click' block is highlighted in the 'Basic' category with a red circle and the number '1'. A red box labeled '2 ドラッグ' (Drag) surrounds the 'Click' block in the script area. A red arrow points from this box to a 'Drop' block (labeled '3 ドロップ') in the script area. The 'Drop' block contains a 'Show text' block with the text 'Hello!'.

1 クリック

2 ドラッグ

3 ドロップ



【3】 タイヤをまわす, LEDをひからせる (4/17)

The image shows a Scratch script editor with a 'Loop' block selected. The 'Input/Output' category is highlighted in the left sidebar. A 'Digital Output' block is being dragged from the 'Input/Output' category into the script area. The block is being placed into a 'Repeat' block that is set to repeat 4 times.

1 クリック

2 クリック

3 ドラッグ

4 ドロップ

5 4かい, くりかえす



【3】 タイヤをまわす, LEDをひからせる (5/17)

The image shows a Scratch script editor with a 'Loop' block selected. The 'Input/Output' palette is open, and an 'Analog Output' block is being dragged from the palette to the script area. The 'Analog Output' block is highlighted in red in the palette and yellow in the script area. The 'Hello!' block is also visible in the script area.

- 1 クリック
- 2 クリック
- 3 ドラッグ
- 4 ドロップ
- 5 2かい, くりかえす



【3】 タイヤをまわす, LEDをひからせる (6/17)



① すうじをかえる

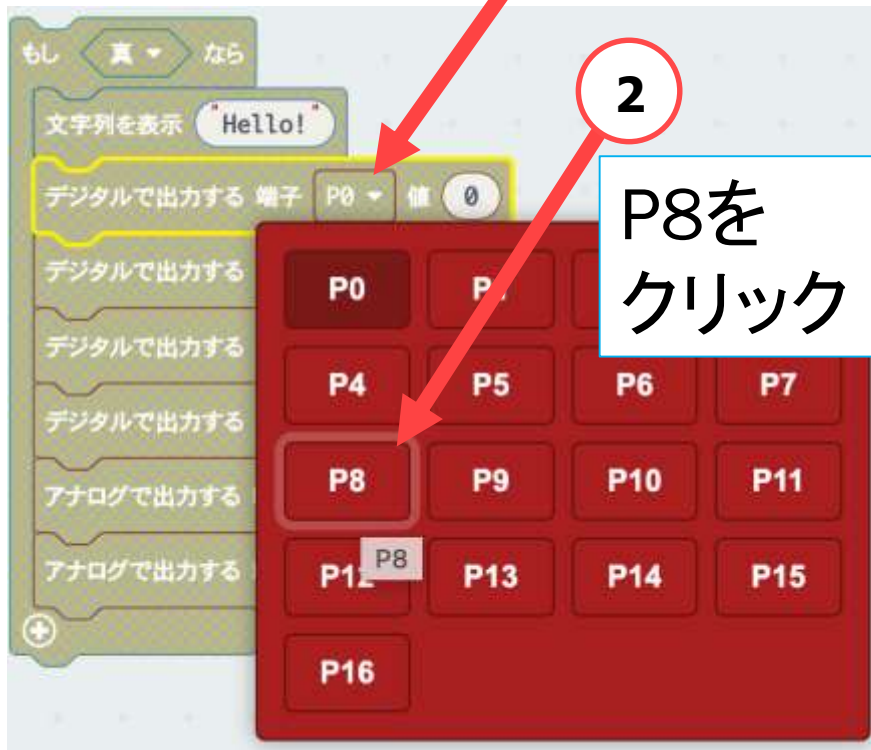
300 にする



【3】 タイヤをまわす, LEDをひからせる (7/17)

1 クリック

3 ほかのポートもかえる





【3】 タイヤをまわす, LEDをひからせる (8/17)

もし 真なら

- 文字列を表示 Hello!
- デジタルで出力する 端子 P8 値 0
- デジタルで出力する 端子 P1 値 0
- デジタルで出力する 端子 P12 値 0
- デジタルで出力する 端子 P2 値 0
- アナログで出力する 端子 P0 値 300
- アナログで出力する 端子 P16(出力のみ) 値 300

① みぎクリック

② クリック

- 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

コピー

③ 4かい, くりかえして, 5つに



【3】 タイヤをまわす, LEDをひからせる (9/17)

クリック

1



2 ドラッグ



3 ドロップ





【3】 タイヤをまわす, LEDをひからせる (10/17)



4 「イベントの値」が “f”, “b”, “l”, “r” のところに4つとも

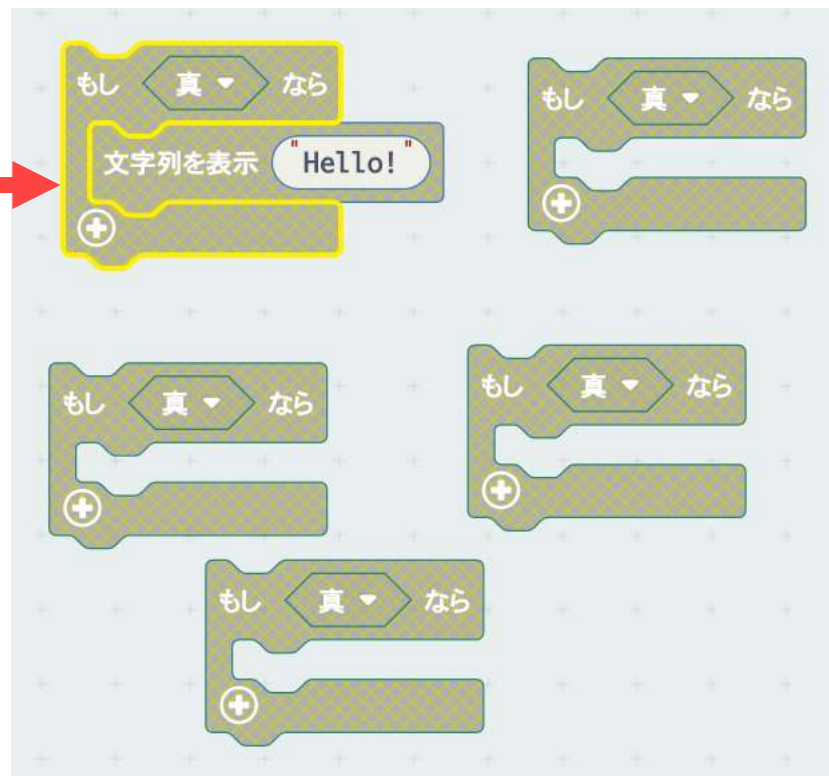


【3】 タイヤをまわす, LEDをひからせる (11/17)

1 クリック

2 del キー

ざんがいを,
さくじょ



3 くりかえして, ほかの4つも, さくじょする



【3】 タイヤをまわす, LEDをひからせる (12/17)

はじめは, とまるので, ぜんぶ 0 にする

もし last data ≠ data なら

- デジタルで出力する 端子 P8 値 0
- デジタルで出力する 端子 P1 値 0
- デジタルで出力する 端子 P12 値 0
- デジタルで出力する 端子 P2 値 0
- アナログで出力する 端子 P0 値 0
- アナログで出力する 端子 P16 (出力のみ) 値 0

P8 → 0

P1 → 0

P12 → 0

P2 → 0

P0 → 0

P16 → 0



【3】 タイヤをまわす, LEDをひからせる (13/17)

data が f (まえ)

赤のところをかえる

The screenshot shows a programming environment with a script starting with a conditional block: "もし data が 'f' なら". The value "f" is circled in red. Below the conditional block are several output blocks:

- 文字列を表示 "F"
- デジタルで出力する 端子 P8 値 1
- デジタルで出力する 端子 P1 値 0
- デジタルで出力する 端子 P12 値 1
- デジタルで出力する 端子 P2 値 0
- アナログで出力する 端子 P0 値 300
- アナログで出力する 端子 P16 (出力のみ) 値 300

文字	→	F
P8	→	1
P1	→	0
P12	→	1
P2	→	0
P0	→	300
P16	→	300



【3】 タイヤをまわす, LEDをひからせる (14/17)

dataが b (うしろ)

赤のところをかえる

でなければもし data = "b" なら

文字列を表示 "B"

デジタルで出力する 端子 P8 値 1

デジタルで出力する 端子 P1 値 0

デジタルで出力する 端子 P12 値 1

デジタルで出力する 端子 P2 値 0

アナログで出力する 端子 P0 値 300

アナログで出力する 端子 P16 (出力のみ) 値 300

文字 → B

P8 → 0

P1 → 1

P12 → 0

P2 → 1

P0 → 300

P16 → 300



【3】 タイヤをまわす, LEDをひからせる (15/17)

data が I(エル) (ひだり)

赤のところをかえる

でなければもし data ▼ = ▼ "1" なら ⊖

文字列を表示 "L"

デジタルで出力する 端子 P8 ▼ 値 1

デジタルで出力する 端子 P1 ▼ 値 0

デジタルで出力する 端子 P12 ▼ 値 0

デジタルで出力する 端子 P2 ▼ 値 1

アナログで出力する 端子 P0 ▼ 値 300

アナログで出力する 端子 P16 (出力のみ) ▼ 値 300

文字	→	L
P8	→	1
P1	→	0
P12	→	0
P2	→	1
P0	→	300
P16	→	300



【3】 タイヤをまわす, LEDをひからせる (16/17)

dataが r (みぎ)

赤のところをかえる

The screenshot shows a programming environment with a script area. At the top, there is a conditional block: "でなければもし" (if not) followed by "data" and an equals sign, and a circle containing the character "r". Below this is a blue block "文字列を表示" (display text) with the character "R". Following are six red blocks for digital and analog outputs:

- デジタルで出力する 端子 P8 ▼ 値 0
- デジタルで出力する 端子 P1 ▼ 値 1
- デジタルで出力する 端子 P12 ▼ 値 1
- デジタルで出力する 端子 P2 ▼ 値 0
- アナログで出力する 端子 P0 ▼ 値 300
- アナログで出力する 端子 P16 (出力のみ) ▼ 値 300

文字	→	R
P8	→	0
P1	→	1
P12	→	1
P2	→	0
P0	→	300
P16	→	300



【3】 タイヤをまわす, LEDをひからせる (17/17)

検索...

基本

1 クリック

基本

入力

音楽

LED

Bluetooth

ループ

論理

変数

計算

拡張機能

高度なブロック

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

すべてのLEDを消す

アナログで出力する 端子 P16 (出力のみ) 値 300

でなければもし data = "A" なら

文字列を表示 "1" 1 にする

でなければもし data = "B" なら

文字列を表示 "2" 2 にする

でなければ

表示を消す

3

ドラッグ&ドロップ



パソコンと micro:bit をケーブルでつなぐ

パソコン



<https://www.irasutoya.com/>より転載

ブラウザが
Google chrome, Edge など
グーグル クローム エッジ



(2) へ

それ以外

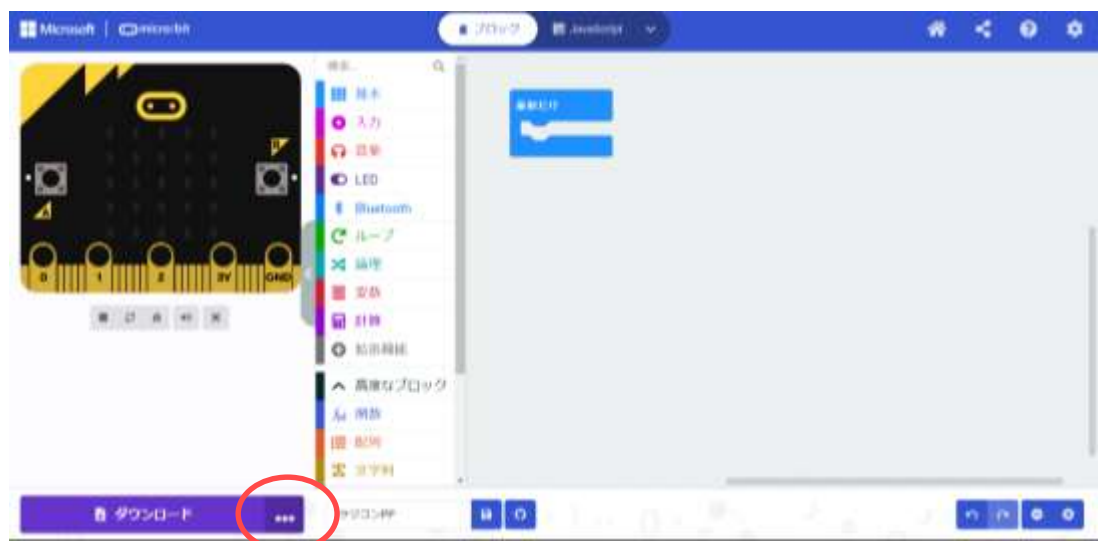


つなげる . (5) へ

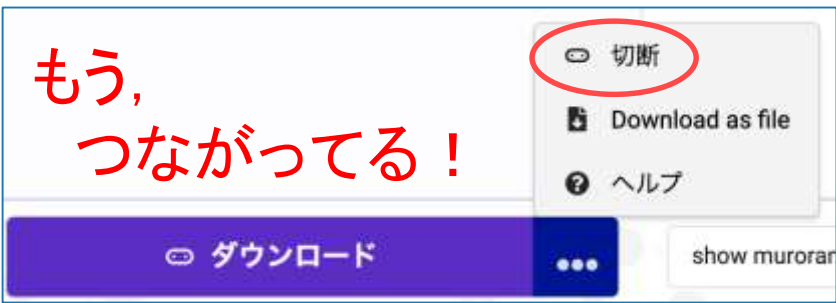
micro:bit ヘブプログラムをおくる(2)



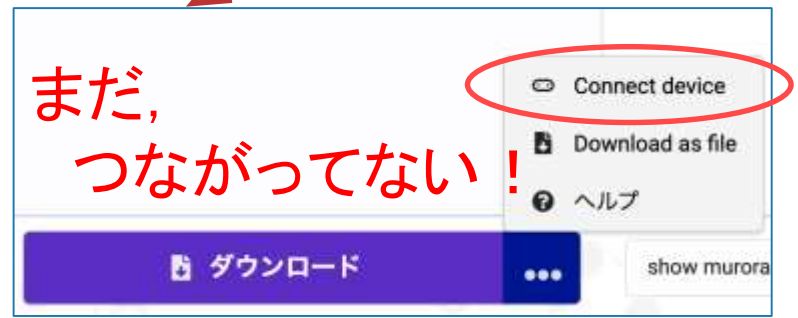
ブラウザと
つながっているか、
たしかめる



クリック



↓ (4) へ



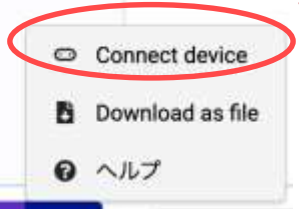
↓ つなげる。(3)へ

micro:bit ヘブプログラムをおくる(3)



ブラウザとつなげる

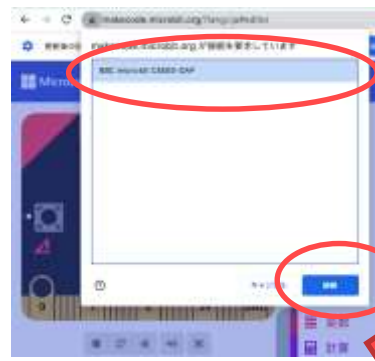
クリック



クリック



クリック



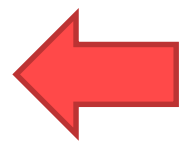
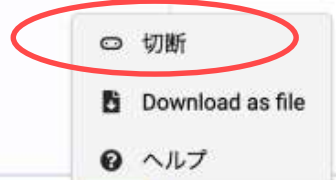
クリックでえらぶ

クリック



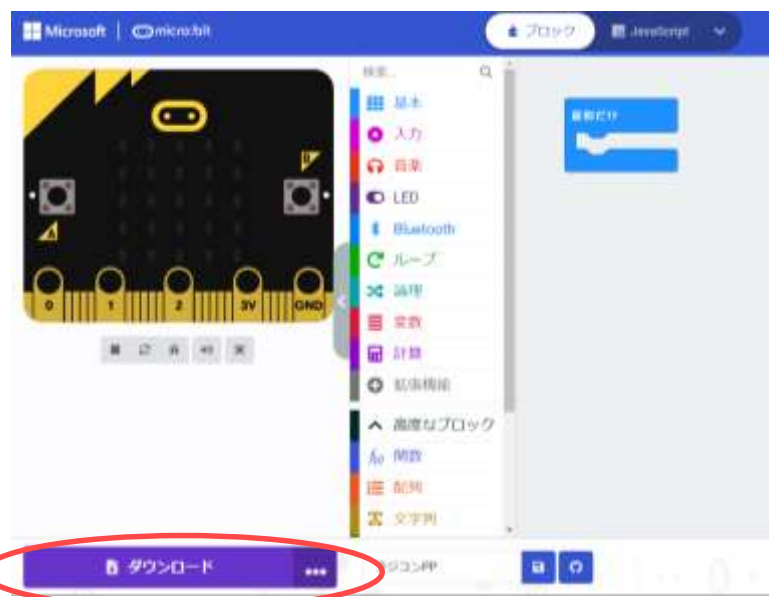
クリック

おわったら、
たしかめ！





プログラムをおくる



クリック



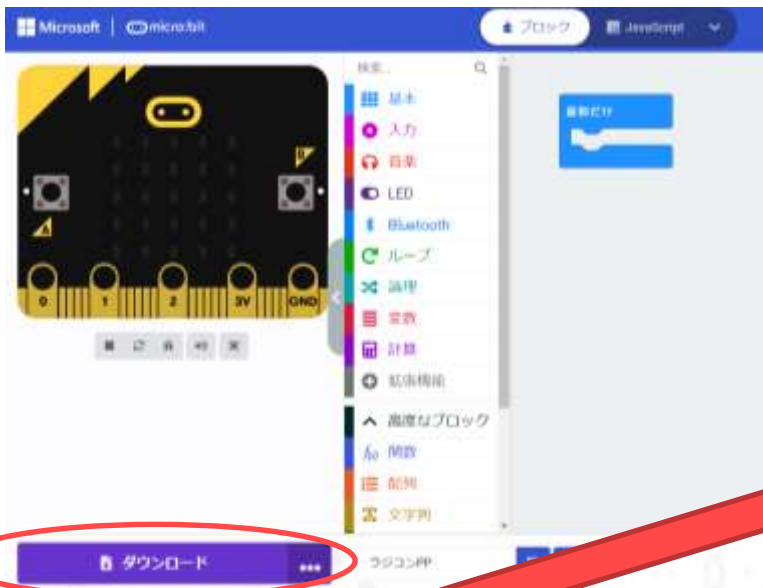
<https://www.irasutoya.com/>より転載

オレンジに
ピカピカ
ひかる

おわり



Google Chrome や Edge などではないとき



クリック



クリック

プログラムはパソコンの中に
ダウンロードしたよ。
パソコンで
micro:bit にコピーしてね



プログラムをコピー

MICROBIT ドライブに
ドラッグ & ドロップ



プログラムのなまえは
microbit- ではじまるよ



<https://www.irasutoya.com/>より転載



オレンジに
ピカピカ
ひかる

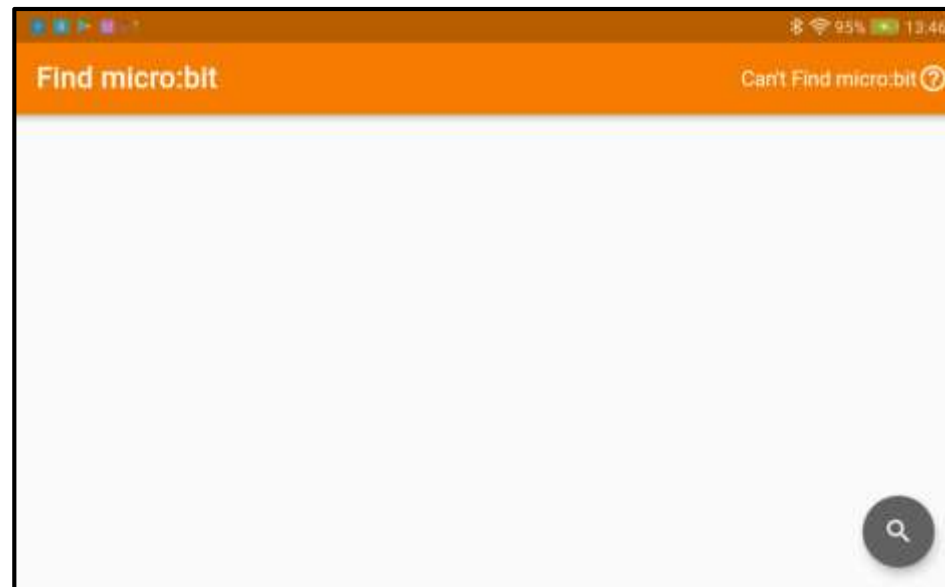
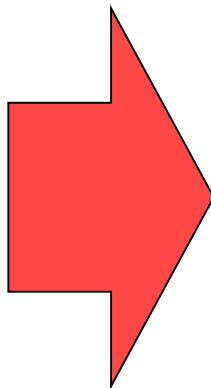
おわり



- スマホのホーム画面から「EV-micro:bit」というアプリを起動する



このアイコンをタップしてアプリを起動しよう



アプリのホーム画面



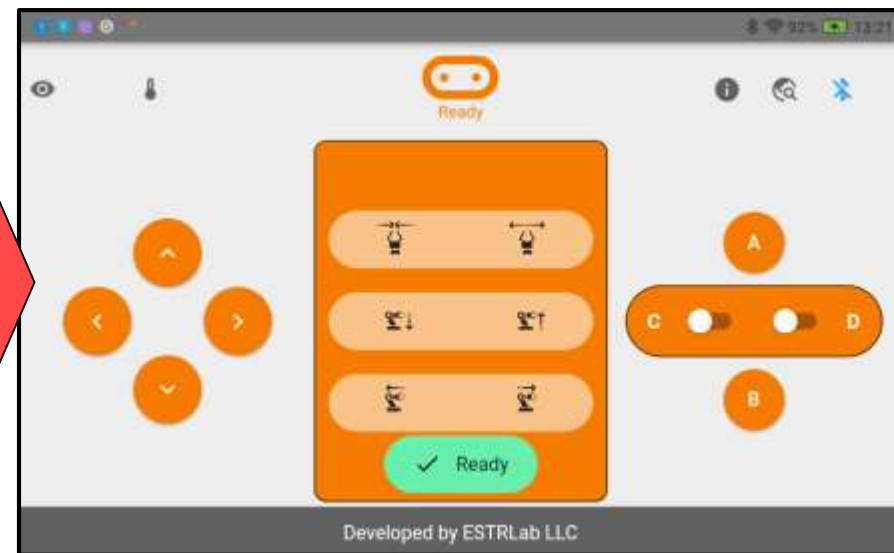
- microbitの電源をいれ、アプリのホーム画面から手元にあるmicrobitを選択し、端末と接続する



ここをタップして、
手元のmicrobitを選択



何も表示されない場合は、
ここをタップしてみよう



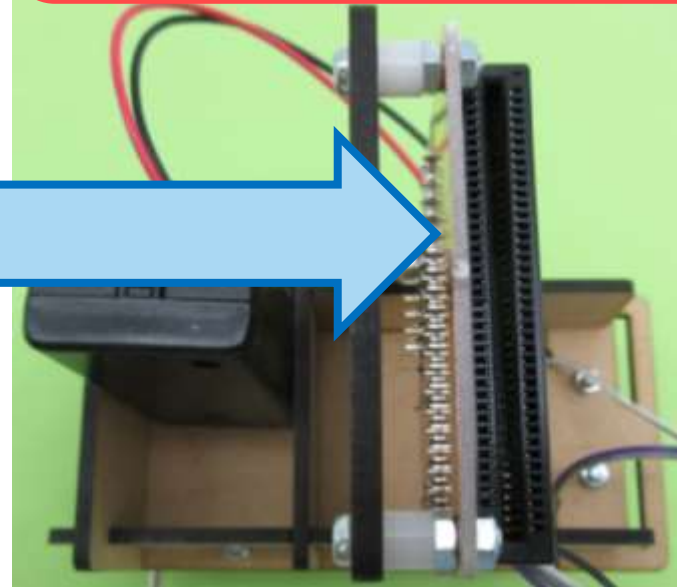
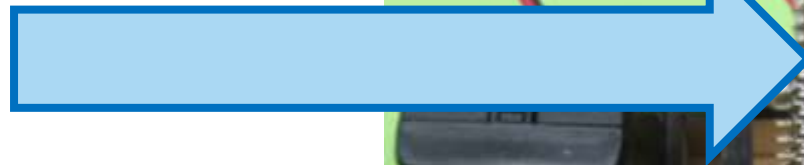
何も表示されない場合は、
ここをタップしてみよう

※もし繋がらなかったら指導員に聞こう！



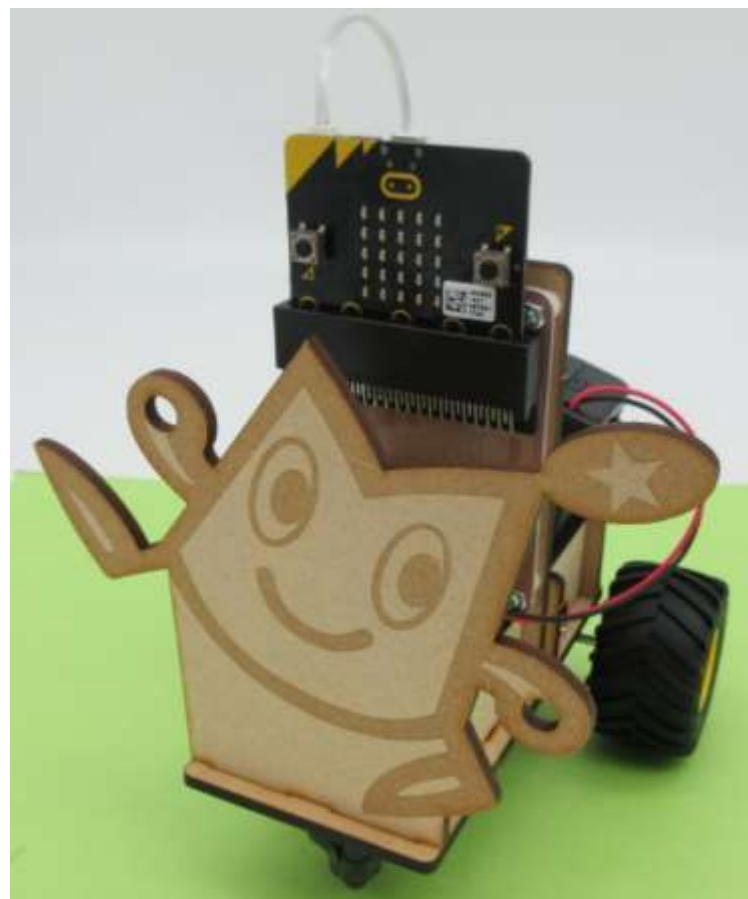
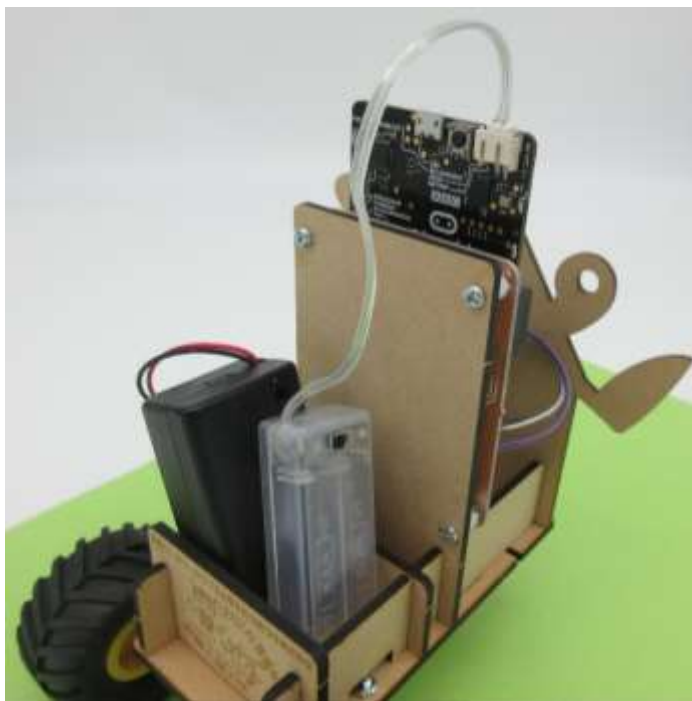
- 黒色のソケットにmicro:bitを差し込む

差し込むのが難しい場合は
指導員に声掛けて下さい。



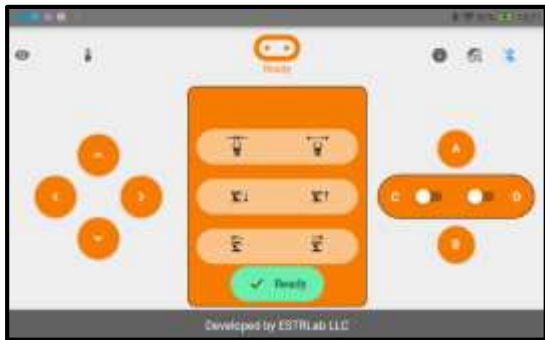


- 電池ボックス端子を接続して、後ろに入れる



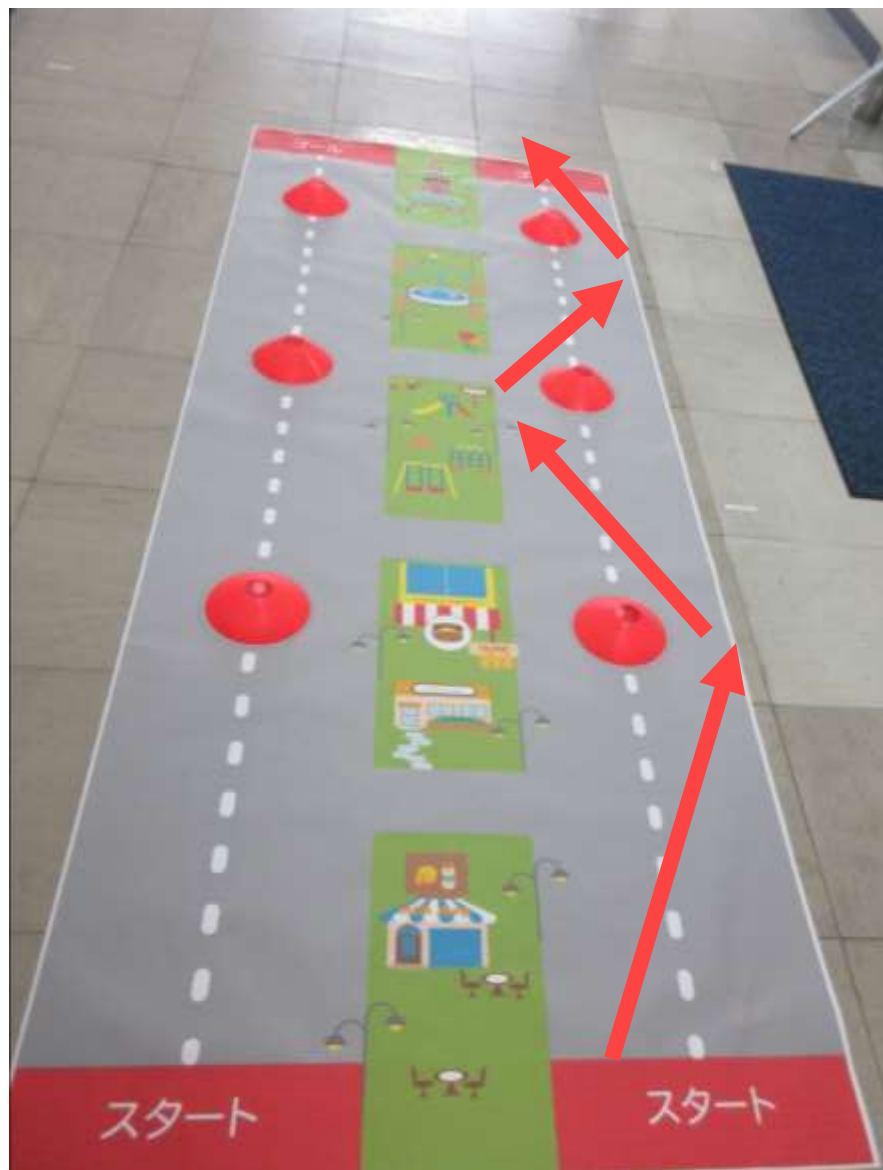
完成！！

じっさいにうごかしてみよう！

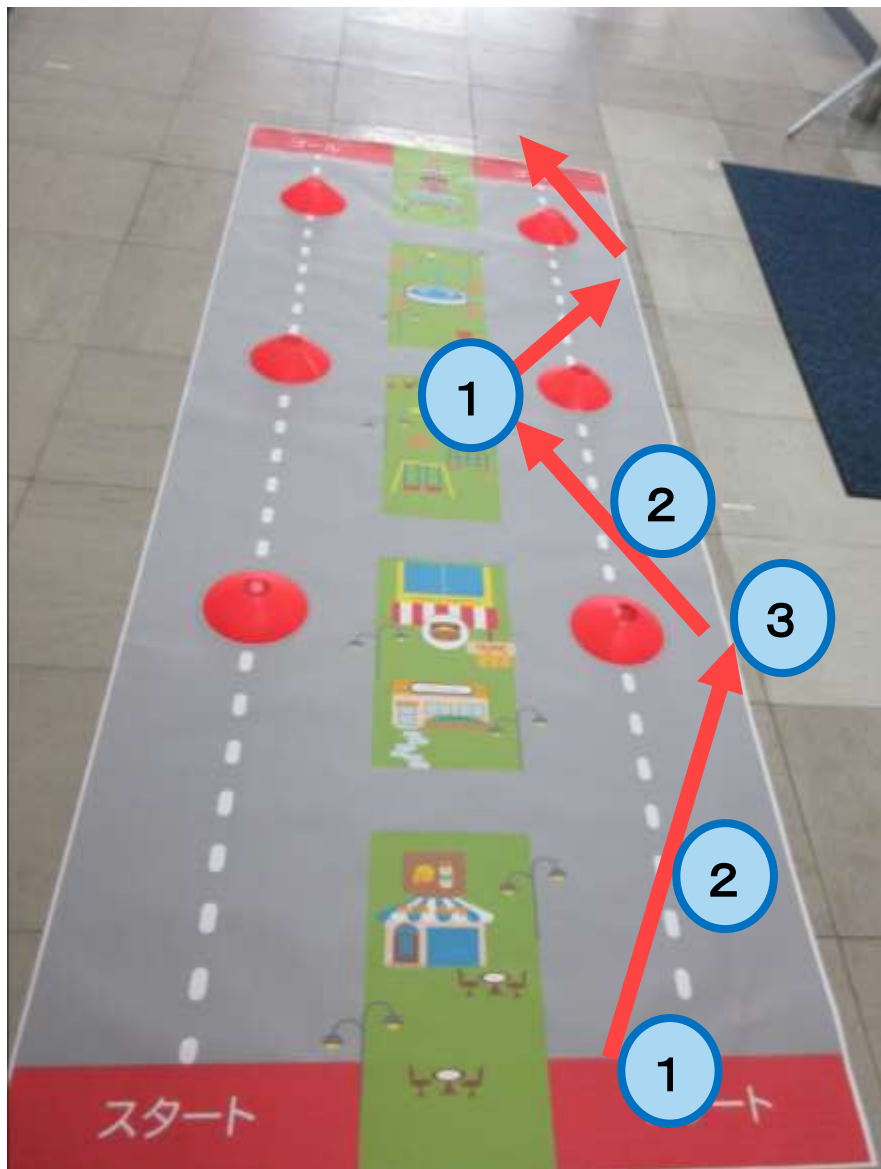




チャレンジしてみよう！



自動でギザギザ道を進む
プログラムを考えよう。



② 真っ直ぐ進む



③ 車体を左に向ける



② 真っ直ぐ進む



① 車体を右に向ける





■ 真っ直ぐ進むために補正する



左のプログラムを動かして真っ直ぐ進むか確認する。
真っ直ぐ進むようにP0とP16の値を変える。

左 P0 300 — =

右 P16 300 — =



- 秒数と角度の関係を求める。



左のプログラムを動かして
何ミリ秒で角度が何度変わる
か確認する。

100ミリ秒 → °
↓
5° → ミリ秒

45° → ミリ秒

90° → ミリ秒



でなければもし イベントの値 = 9 なら

文字列を表示 1

デジタルで出力する端子 P8 値 0

デジタルで出力する端子 P1 値 1

デジタルで出力する端子 P12 値 1

デジタルで出力する端子 P2 値 0

アナログで出力する端子 P0 値 300

アナログで出力する端子 P16 (出力のみ) 値 300

一時停止 (ミリ秒) 100

デジタルで出力する端子 P8 値 1

デジタルで出力する端子 P1 値 0

デジタルで出力する端子 P12 値 1

デジタルで出力する端子 P2 値 0

アナログで出力する端子 P0 値 300

アナログで出力する端子 P16 (出力のみ) 値 300

一時停止 (ミリ秒) 1600

45° 右に回転

初めは車体が真っ直ぐ
なので秒数を変える

前進

初めは距離が長いので
秒数を変える



くりかえし 2 回

90° 左に回転

3

デジタルで出力する 端子 P8 値 1

デジタルで出力する 端子 P1 値 0

デジタルで出力する 端子 P12 値 0

デジタルで出力する 端子 P2 値 1

アナログで出力する 端子 P16 (出力のみ) 値 300

アナログで出力する 端子 P0 値 300

一時停止 (ミリ秒) 250

デジタルで出力する 端子 P8 値 1

デジタルで出力する 端子 P1 値 0

デジタルで出力する 端子 P12 値 1

デジタルで出力する 端子 P2 値 0

アナログで出力する 端子 P0 値 300

アナログで出力する 端子 P16 (出力のみ) 値 300

一時停止 (ミリ秒) 1600

前進

2

一時停止 (ミリ秒) 1600

デジタルで出力する 端子 P8 値 0

デジタルで出力する 端子 P1 値 1

デジタルで出力する 端子 P12 値 1

デジタルで出力する 端子 P2 値 0

アナログで出力する 端子 P0 値 300

アナログで出力する 端子 P16 (出力のみ) 値 300

一時停止 (ミリ秒) 250

デジタルで出力する 端子 P8 値 1

デジタルで出力する 端子 P1 値 0

デジタルで出力する 端子 P12 値 1

デジタルで出力する 端子 P2 値 0

アナログで出力する 端子 P0 値 300

アナログで出力する 端子 P16 (出力のみ) 値 300

一時停止 (ミリ秒) 1600

90° 右に回転

1

前進

2

繰り返すことでギザギザを進めるよ