第17回 IEEE Engineer Spotlight

動く！光る！音が鳴る！アイデアをカタチにしよう！
～今年の自由研究はプログラミングと電子工作だ！～

㈱東芝 研究開発センター / IEEE Tokyo Young Professionals Vice Chair

石垣 雄太朗

2021年7月24日

自己紹介: 石垣 雄太朗

所属: ㈱東芝 研究開発センター / IEEE Tokyo Young Professionals Vice Chair

略歴

2007～2012　東京 工業高等専門学校 (電子工学科)

2012～2016　東京農工大学・大学院 (電気電子工学科)

2016～　(株)東芝 ・ 東芝デバイス＆ストレージ(株)

通称「高専」
高校と異なり5年制の学校

理系を目指したい人は、選択肢の一つとしてご参考に！

IEEE

IEEE（アイトリプルイー）とは？

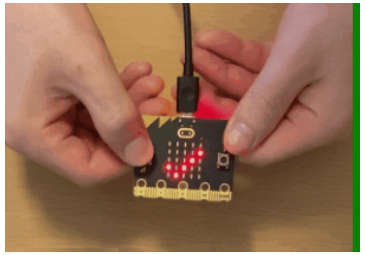
- 電気・情報工学などの分野を中心とする世界最大の技術団体
- 活動内容
  - 学術 会議の開催、論文誌の発行、技術 標準 化活動
  - セミナー開催、国際交流、教育活動

…など様々な活動

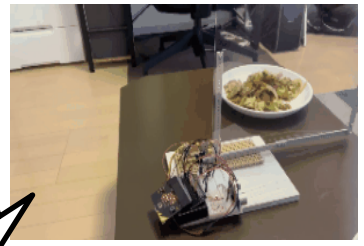
IEEE Engineer Spotlightではさまざまな講演を聴くことができます。

今日のお話

1. ミニゲームを作って
プログラミングの基本を学習！

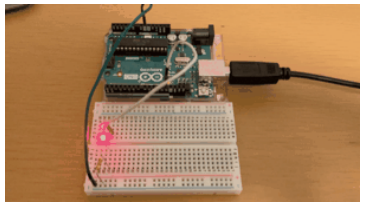


2. 自動開閉パーティション
を作って電子工作を体験！

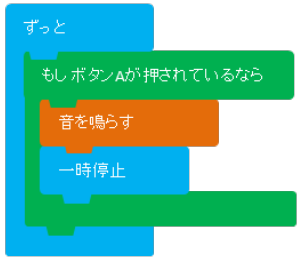


コロナ対策！

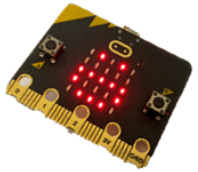
3. LED点滅実験で
回路設計体験！



自由研究にもぜひ！



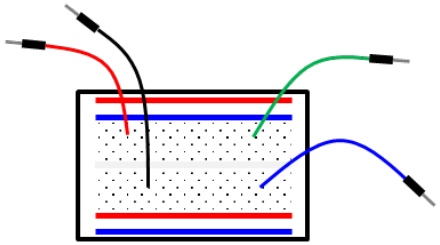
プログラミングで
自在にコントロール！



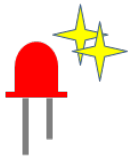
micro:bit

Arduino

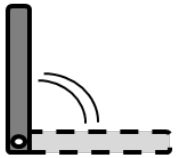
音が鳴る！



ブレッドボードで
回路を作る！
(はんだ付け不要)



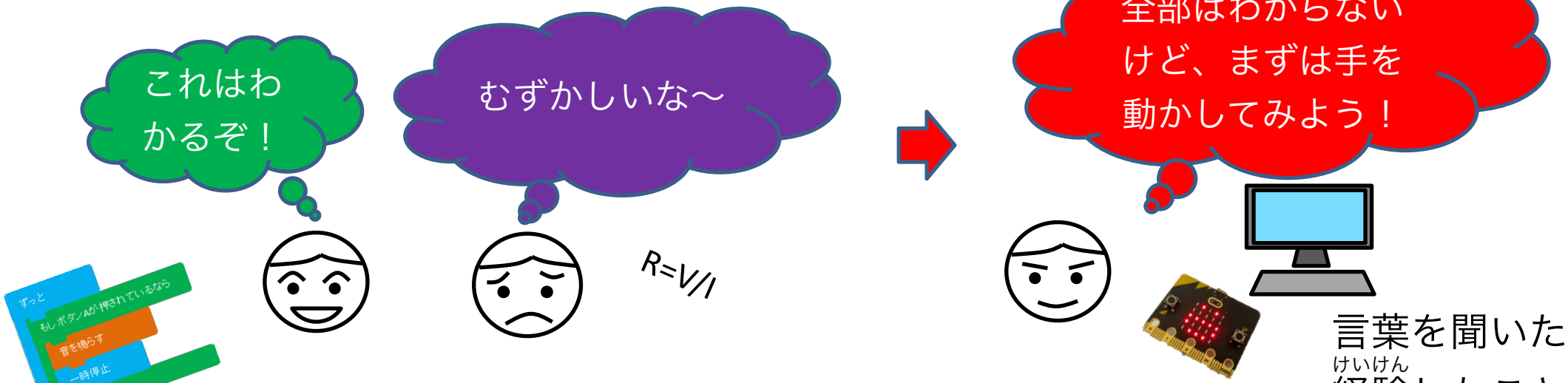
光る！



動く！

今日お話するにあたって

学年によっては難しい内容も含みます



言葉を聞いたことがある・経験したことがあるだけでも今後役に立つはず！

今日のお話し中は、手を動かしてもらう時間はないので、あとから動画や資料を見返して、ぜひチャレンジ！

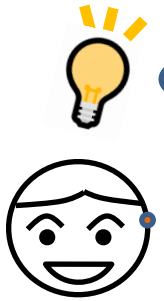
動画と資料→「エンジニアスポットライト」で検索！

アイデアをカタチにするには？

「できること」は多い方がいい

「できること」が少ないと…

アイデアがうかんでも実現できない



あんなことといいな..
できたらいいな..

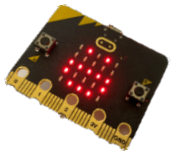
でも、どうやって作ったらいいかわからない…

現実的なアイデアがうかばない

一応できる範囲で考えたがこれは一体何の役に立つんだ…

今日はみなさんの「できること」を増やします！
アイデアのヒントになるような、工作例もいくつか示します！

マイコンプログラミング



“現実世界で”モノを動かしたり、光らせたりできる！

工具と部品について

以下のページを参考に、必要なものをそろえてください

https://www.ieee-jp.org/japancouncil/affinitygroup/R10_Spotlight/images/tools17.pdf

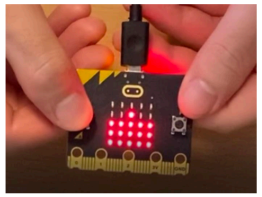
使うもの

必要に応じて部品や工具*を購入してください。オンデマンド配信も予定しているため、講演後に購入頂いても問題ありません。

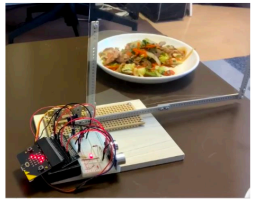
＊お近くのホームセンターやオンラインショップ等で購入する事ができます。実習される場合必要に応じてご準備ください。　(micro:bit V2はお早めの購入をおすすめします。)

講演での実習内容

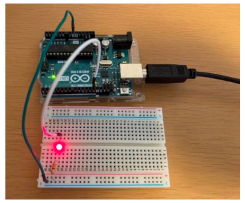
① ミニゲーム作成・プログラミングの基礎学習



② 自動開閉パーティション



③ Arduino LED点灯実験



工具表

品名	実習内容 ①	②	③
ニッパー	-	★	-
ラジオペンチ	-	★	★
プラスドライバー (No. 1, No. 2)	-	★	-
錐 (キリ)	-	★	-

★: 必須　■: 任意　- : 不要

次ページも同様

ニッパー

ラジオペンチ

プラス
ドライバー

錐 (キリ)

部品表

※価格は2021年6月末時点のもの

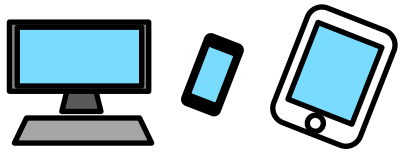
★: 必須　■: 任意　- : 不要

品名	【参考】価格(円)	個数	【参考】購入できる場所 (URL)	実習内容 ①	②	③	備考
micro:bit go V2スターターキット	2400	いずれか1個	https://akizukidenshi.com/catalog/g/gK-16019/				【おすすめ】USBケーブル、電池ボックス付き
micro:bit V2	2200		https://akizukidenshi.com/catalog/g/gM-15882/	★	★	-	【おすすめ】
micro:bit V1	2160		https://akizukidenshi.com/catalog/g/gM-12513/				
USBケーブル A-microB	120	1	https://akizukidenshi.com/catalog/g/gC-07607/	★	★	-	micro:bit go V2スターターキットを購入した場合は不要
micro:bitブレイクアウトボード	680	1	https://akizukidenshi.com/catalog/g/gP-12836/	-	★	-	
Arduino Uno Rev3	2940	1	https://akizukidenshi.com/catalog/g/gM-07385/	-	-	★	
USBケーブル (Aオス-Bオス)	130	1	https://akizukidenshi.com/catalog/g/gC-07605/	-	-	★	
ACアダプタ(9V, 2.5A)	900	1	https://akizukidenshi.com/catalog/g/gM-10877/	-	-	■	パソコンから電源供給しない場合は必要
ブレッドボード	200	1	https://akizukidenshi.com/catalog/g/gP-05294/	-	★	★	
ブレッドボード用ジャンパー線 (オス-メス) 10本入	220	1	https://akizukidenshi.com/catalog/g/gC-08932/	-	★	-	黒色 (リンク先から他の色も選べます)
ブレッドボード用ジャンパー線 (オス-オス) 10本入	350	1	https://akizukidenshi.com/catalog/g/gP-02933/	-	★	★	黒色 (リンク先から他の色も選べます)
ブレッドボード　ジャンパーワイヤ	400	1	https://akizukidenshi.com/catalog/g/gP-00288/	-	★	★	
カーボン抵抗 10kΩ 1/4W (100個入)	100	1	https://akizukidenshi.com/catalog/g/gR-25103/	-	★	-	
カーボン抵抗 20kΩ 1/4W (100個入)	100	1	https://akizukidenshi.com/catalog/g/gR-03940/	-	★	-	
カーボン抵抗 330Ω 1/4W (100個入)	100	1	https://akizukidenshi.com/catalog/g/gR-25331/	-	■	★	
赤色LED 10個入	100	1	https://akizukidenshi.com/catalog/g/gI-02320/	-	■	★	
三端子レギュレータ TA48033S (3.3V 1A)	100	1	https://akizukidenshi.com/catalog/g/gI-00534/	-	★	-	
超音波距離センサ HC-SR04	450	1	https://akizukidenshi.com/catalog/g/gM-11009/	-	★	-	
サーボモータ SG92R	500	1	https://akizukidenshi.com/catalog/g/gM-08914/	-	★	-	
スイッチ付き電池ボックス 単3×3本	90	1	https://akizukidenshi.com/catalog/g/gP-02666/	-	★	-	
電池ボックス 単3×2本 PHコネクタ付	70	1	https://akizukidenshi.com/catalog/g/gP-12665/	■	-		パソコンから電源供給しない場合は必要 (micro:bit go V2スターターキットを買った場合は単4のものが付属しているので不要)
アルカリ単3乾電池 4本入	80	1	https://akizukidenshi.com/catalog/g/gB-03256/	■	★	-	
ユニバーサルプレートセット	350	1	https://akizukidenshi.com/catalog/g/gP-09100/	-	★	-	
ユニバーサルアームセット	400	1	https://akizukidenshi.com/catalog/g/gK-10258/	-	★	-	
タッピングネジ (太さ3mm, 長さ16mm)		1	お近くのホームセンターなどで購入してください。	-	★	-	
木板 はがき倍判 (15cm×20cm, 厚さ1cm)		1		-	★	-	サイズは任意だが、縦横サイズ15cm×20cm, 厚さ1cm以上を推奨。
プラバン B4サイズ 厚み0.3mm		1		-	★	-	例: https://item.rakuten.co.jp/acrysunday/1017359/
プラスチック対応接着剤		1		-	★	-	例: https://amzn.to/3ykTtV4

1 プログラミングの基本（きほん）
～micro:bitを使って学ぼう～

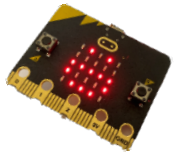
コンピュータの基本

コンピュータと聞いて何を思いうかべる？



今日使うコンピュータは“**マイコン**”

(マイクロコントローラ / マイクロコンピュータ)



あらゆるものに組み込まれている

コンピュータの主な役割は“**計算**”と“**制御**(コントロール)”

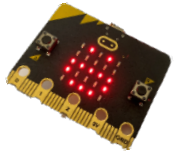
例えば、スマホゲームの裏では、複雑な計算や制御が動いている

例: 一秒間に数千万回以上の計算ができる

一般的なコンピュータはプログラム(命令)通りに動く

LEDを光らせて！

そのあと音を鳴らして！



りょうかい～

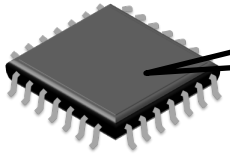
~~めんどいからやだ～~~

プログラミングとは？

- コンピュータへの命令を記述(きじゅつ)すること

プログラム(命令)

0100010…
0010101…
0100100…
0100100…



コンピュータは“1”と“0”しかあつかえない！

コンピュータが理解(りかい)できる形式でプログラミングするのはたいへん…

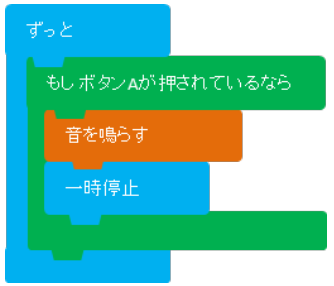
普通(ふつう)は人間が理解(りかい)しやすい言語(げんご)やブロックでプログラミングする

専用(せんよう)のソフトでコンピュータが理解(りかい)できる形式に変換(へんかん)する

言語(げんご)(文字)を使ったもの
C言語, Python…など

```
int main()
{
    printf(“HelloWorld”);
    return 0;
}
```

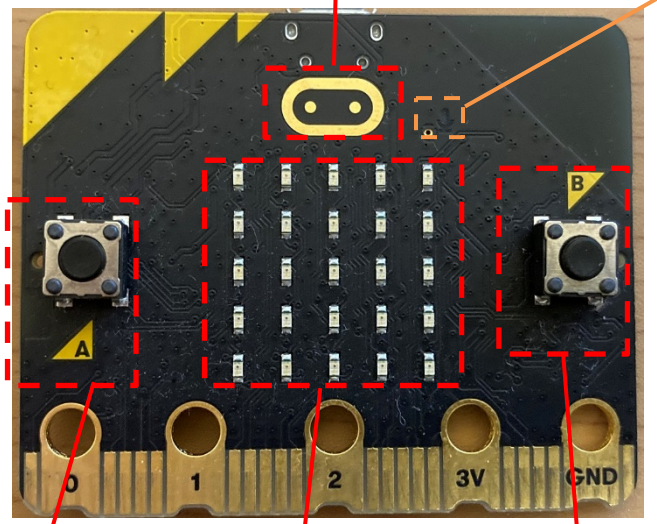
ブロックを使ったもの
Scratch, MakeCode…など



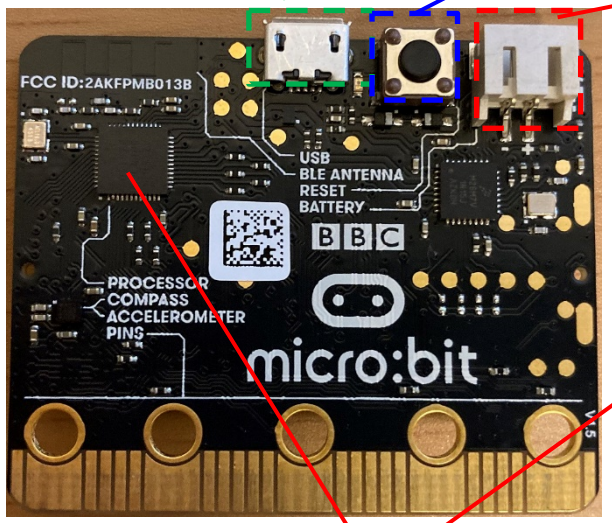
間違(まちが)った命令を与えると、間違(まちが)った通りに動く。(バグ)
正(ただ)しい命令を与(あた)えよう。

micro:bit

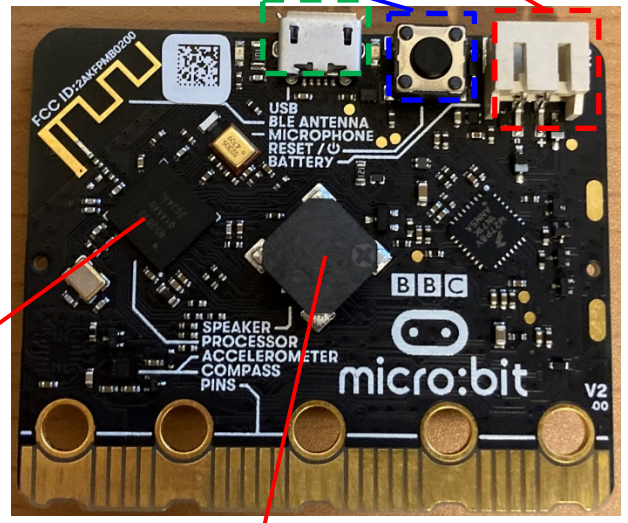
表面(ひょうめん) (共通(きょうつう))



裏面(りめん) (micro:bit V1)



裏面(りめん) (micro:bit V2)



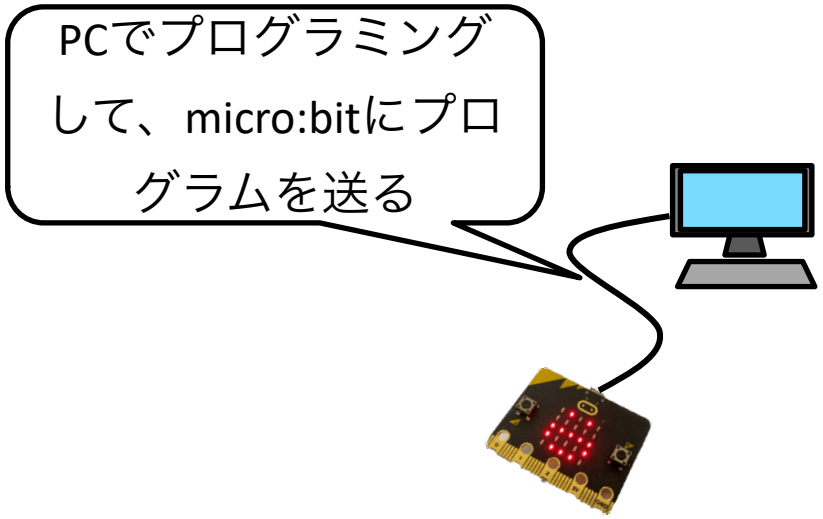
静電気(せいでんき)に非常(ひじょう)に弱いので、できるだけ端子(たんし)やマイコンなど(小さな黒い部品)を素手(すで)でさわらないように！

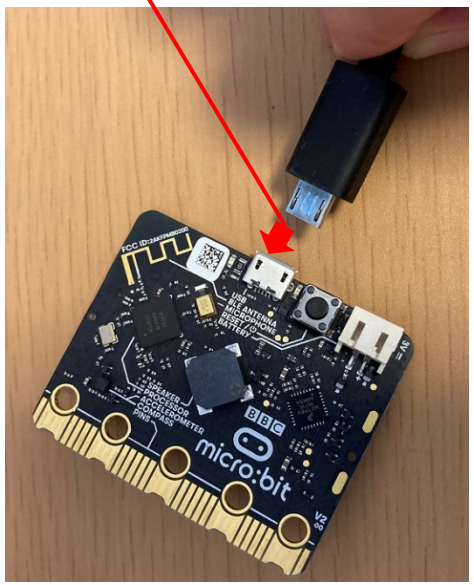
micro:bitのプログラミング方法

- パソコンとmicro:bitをUSBケーブル(A-MicroB)でつないでプログラミング

① USBケーブルのMicroB側(端子が小さい方)をmicro:bitに接続

② USBケーブルのA側(端子が大きい方)をパソコンに接続





どちらでもOK

MakeCodeでのプログラミングのはじめ方

Windows10では専用(せんよう)アプリもあるが、ここではブラウザ(**Google Chromeがおすすめ**)を使ってプログラミング

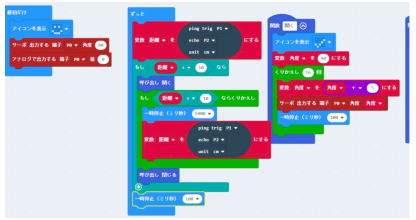
URL: https://makecode.microbit.org/　　または、検索(けんさく)エンジンで「makecode microbit」と検索(けんさく)

”プロジェクト”という単位で
プログラミングを行う

プロジェクト名「test」 -------

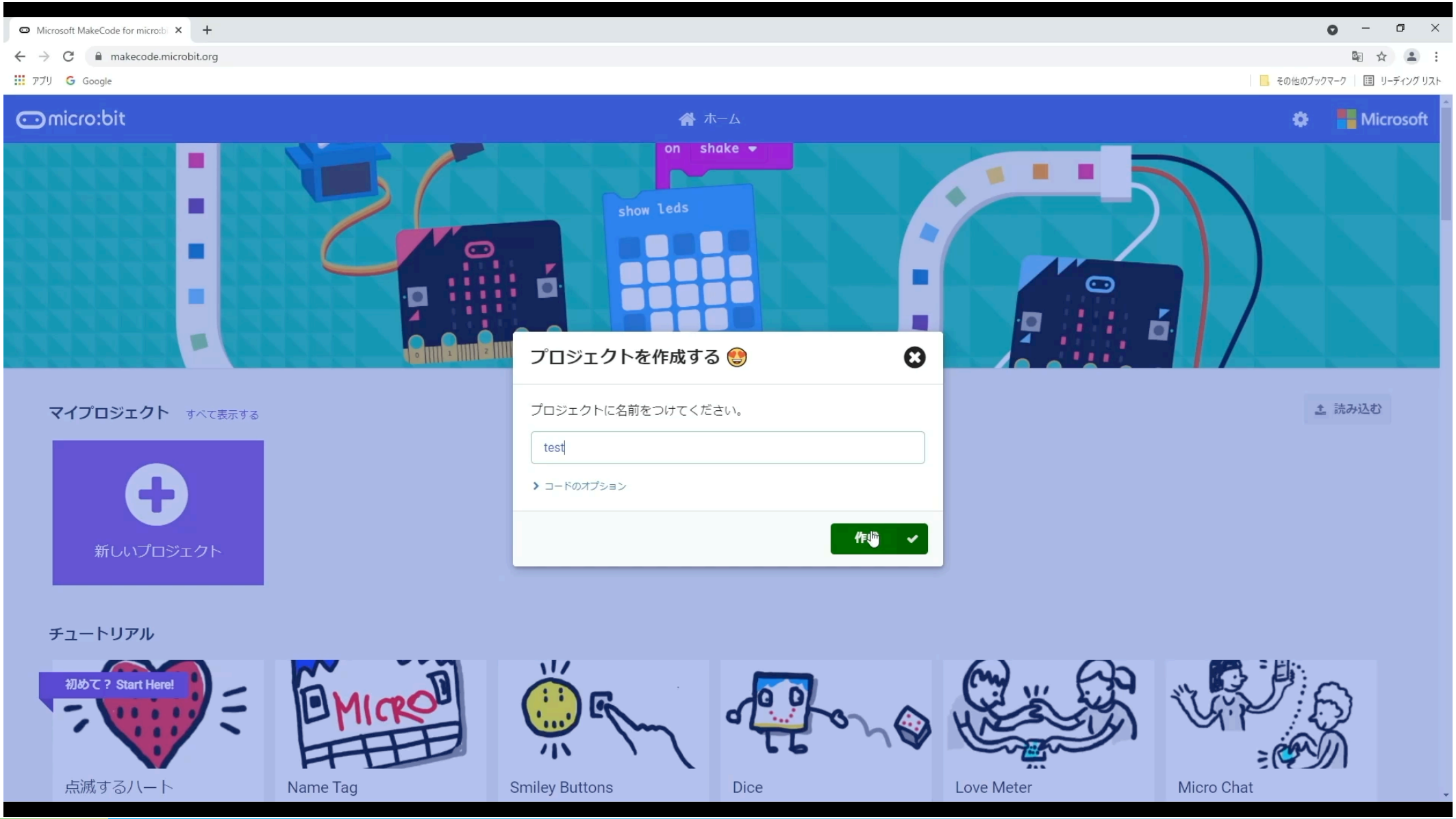
プロジェクト名「タイミングゲーム」 ----------

プロジェクト名「自動開閉パーティション」 -------



好きな名前を入力 (日本語もOK)

MakeCodeでのプログラミングのはじめ方 動画

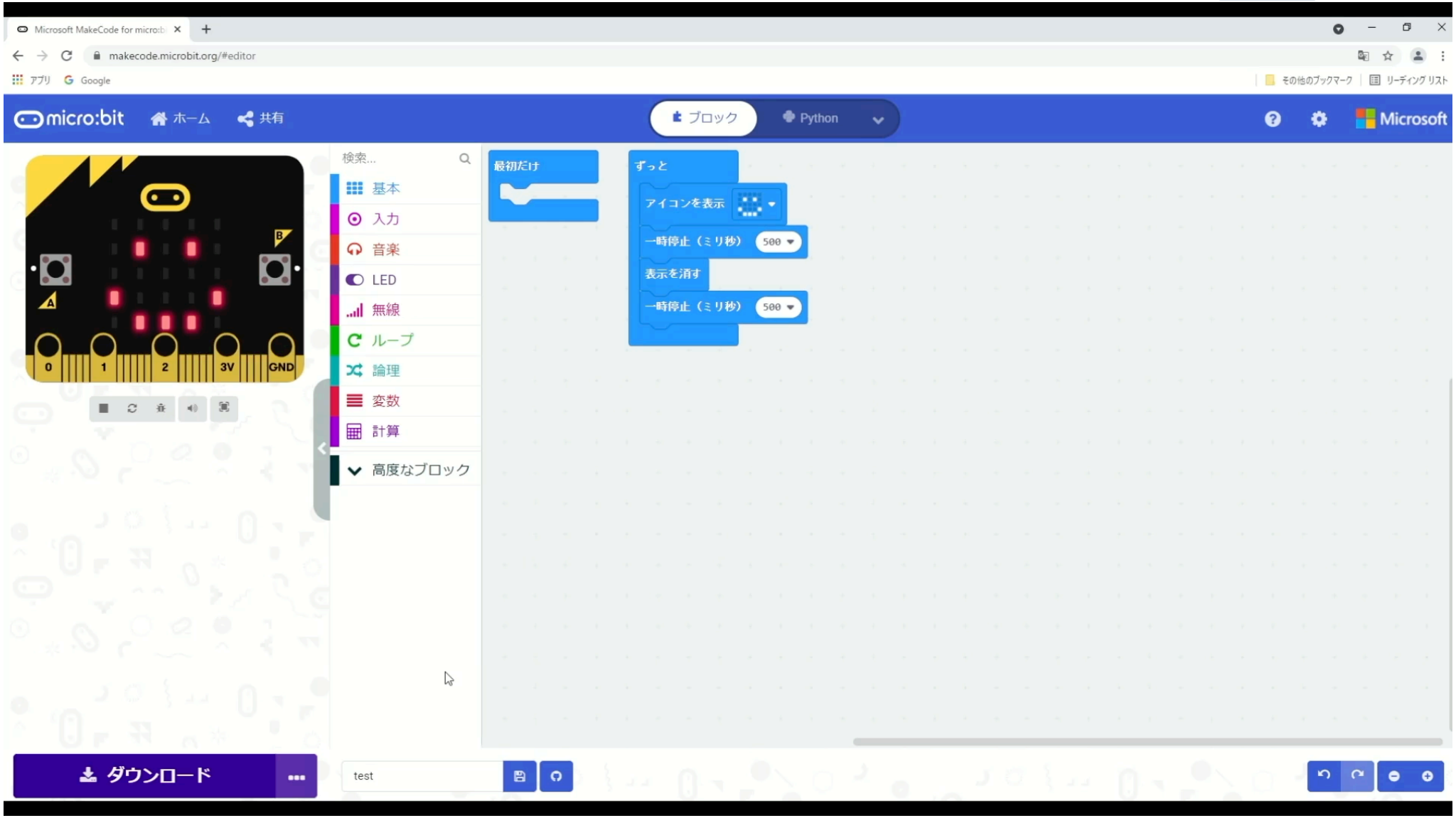


画面の見方


ホームに戻る
Microsoft
micro:bit
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
プログラミング画面
シミュレータ
プログラムの
ダウンロード
ブロックリスト
戻るボタン
ダウンロード
test

まずはLEDを点滅させてみよう！

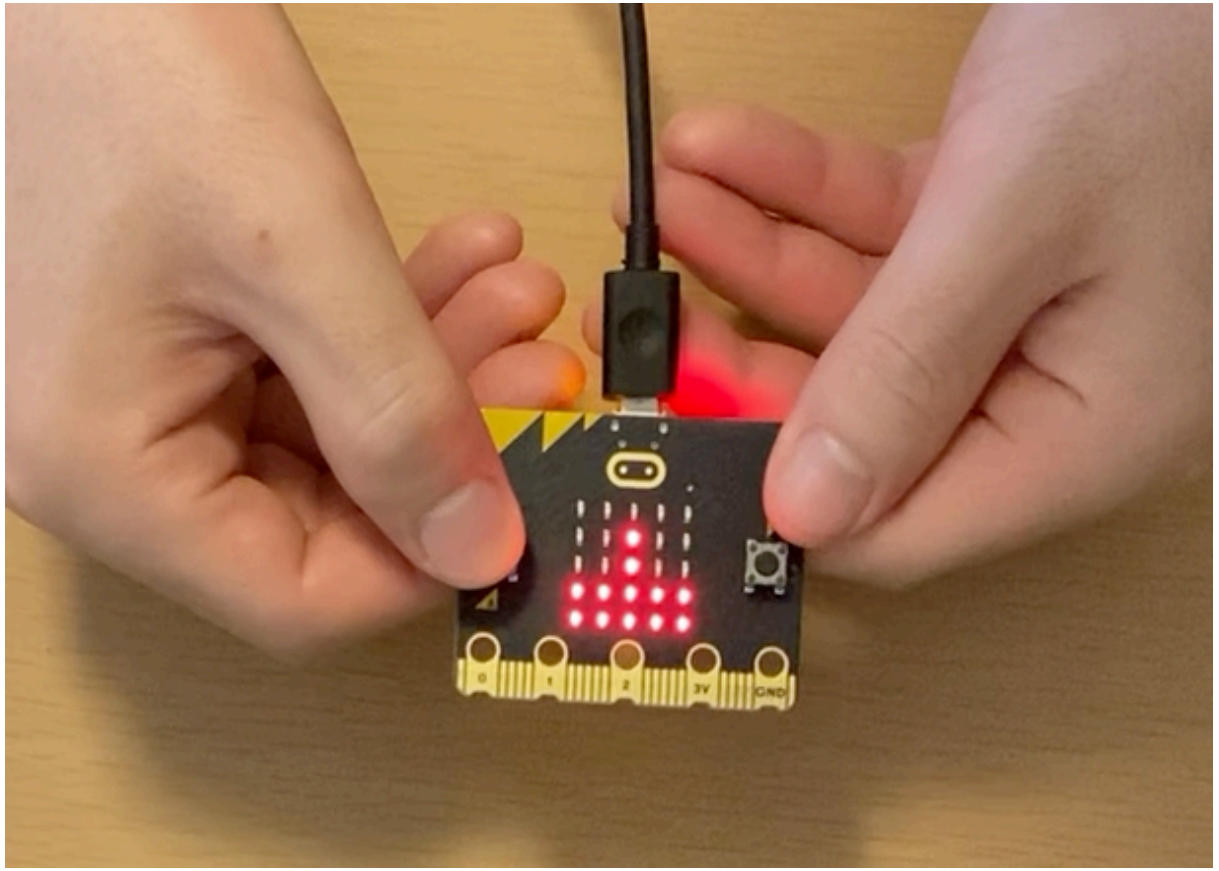
動画



※動画はWindows10だが、Macでも基本的に同様の手順

かんたんなゲームを作りながらプログラミングの基本を学ぼう

- タイミングゲーム！
  - LEDがすべて光っているときにAボタンを押すとクリア
  - クリアするまでのボタンを押す回数で競う (少ない方が勝ち)



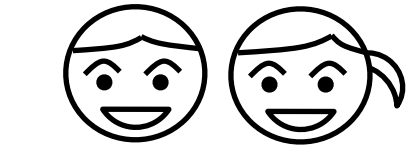
クリアすると「✔」マーク
失敗すると「×」マーク

動画

プログラミングの基本①: 処理の順番

- **上から順番に**実行される
- イベントが発生すると処理の流れが移ることも

処理の順番 → 上から



順番に
やっていこう

もともとのプラン

お昼ごはんを食べる

宿題をやる

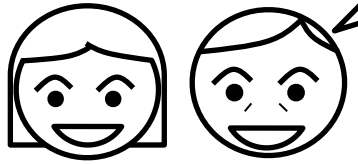
ゲームをやる

急におつかいをたのまれると…

店に向かう

買い物をする

家に帰る



おつかい行ってきて～

micro:bitの処理の流れ

このブロックの中は、**電源を入れたときに一回だけ**実行

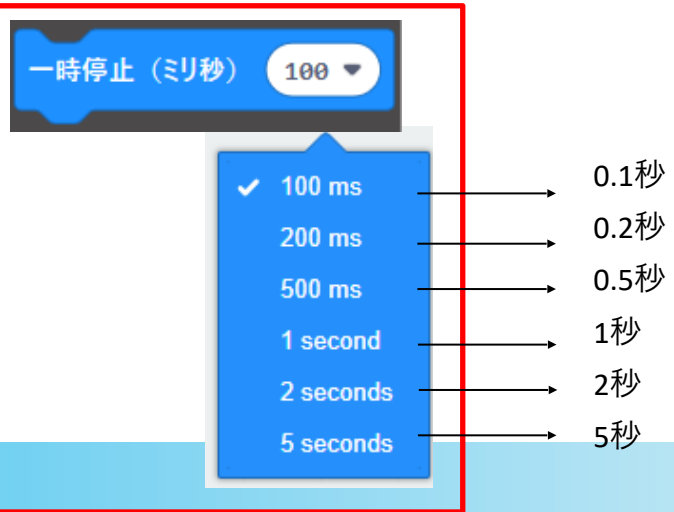
このブロックの中は**ずっとくり返される**

そのほかの四角いブロックの中は、**対応するイベントが起こったときのみ**実行

コンピュータの処理速度はとても速い！

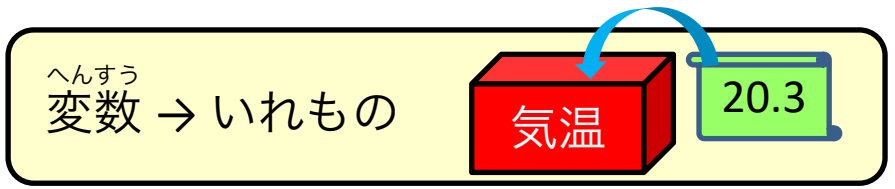
多くの処理は、**人間にとって一瞬**！

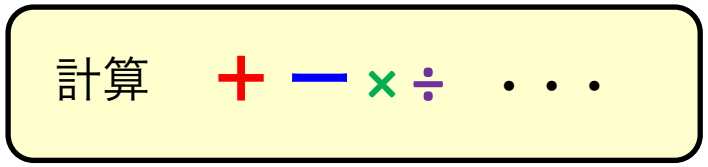
適切に「一時停止ブロック」を入れよう！

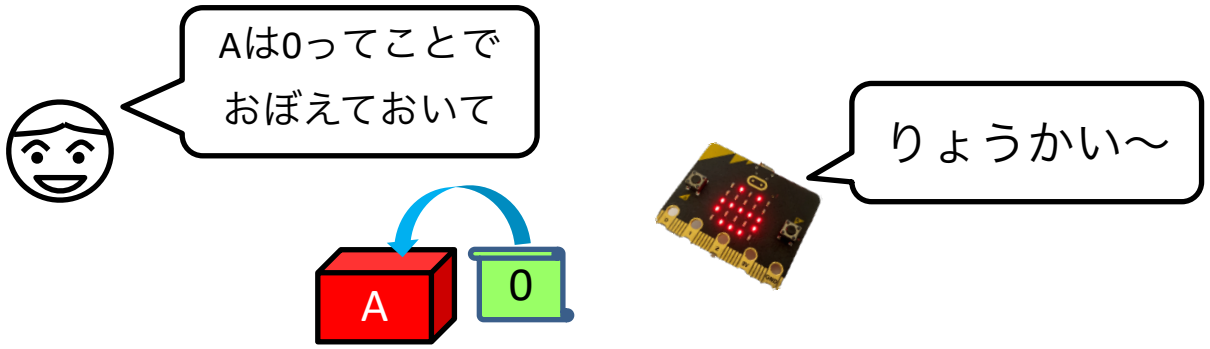


プログラミングの基本②: 変数と計算

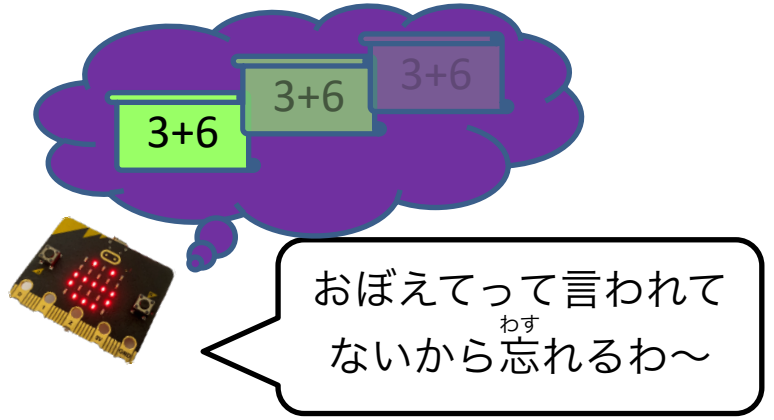
- 変数とは？
  - データのいれもの (データ: 数値、文字など)
  - 好きな名前をつけられる
- どんな計算ができる？
  - 足し算、引き算、かけ算、わり算など





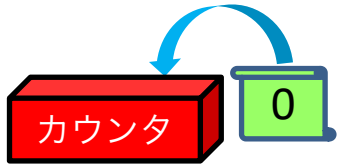


変数に入れないと…



変数と計算のブロック

変数にデータをセットするとき

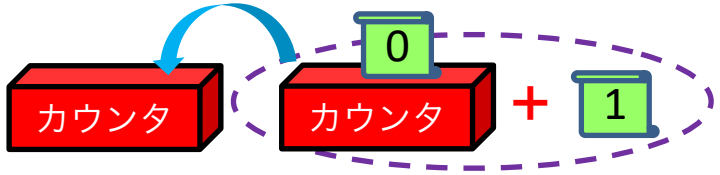


変数を使うとき

計算するとき

計算式に変数も入れることができる

組み合わせると…

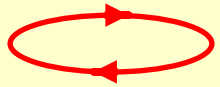


“カウンタ”の中身が**0**から0+1(=**1**)に変化

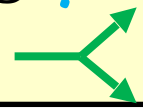
計算結果で“カウンタ”変数を書きかえる

プログラミングの基本③: ループと条件分岐

- ループとは？
  - 同じ処理を何回もくり返すこと
- 条件 分岐とは？
  - 設定した条件によって処理を変えること

ループ → くり返し 

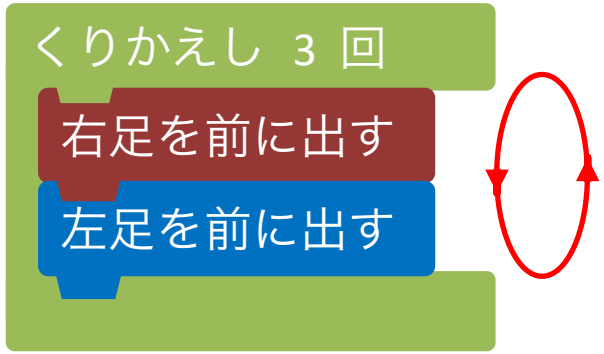
条件 → もし…だったら ?

分岐 → 枝分かれ 

ループの例

6歩 歩く

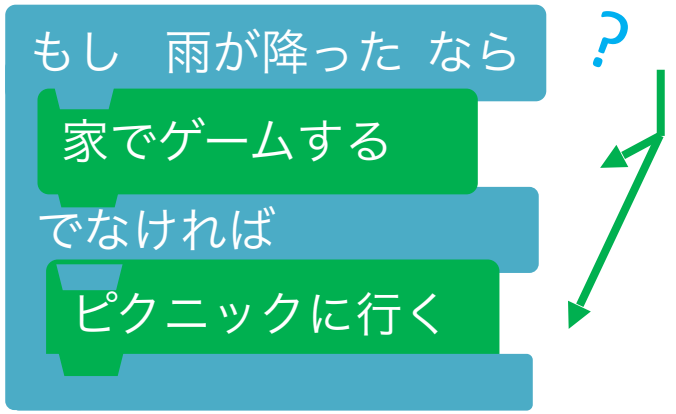
右足を前に出す
左足を前に出す
右足を前に出す
左足を前に出す
右足を前に出す
左足を前に出す



条件 分岐の例

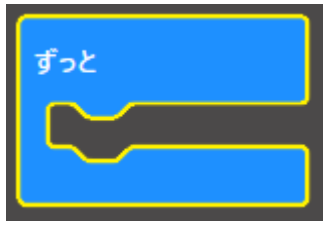
休日の予定

もし雨が降ったら
家でゲームする
そうじゃなかったら
ピクニックに行く

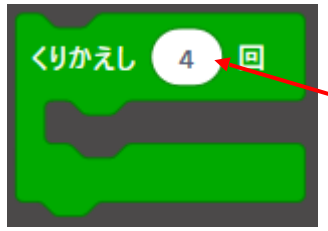


ループのブロック

ループのブロックの例



micro:bitの電源をONにしてからずっと中の処理をくり返す



ここで決めた回数、中の処理をくり返す

条件が成り立っているときだけループが回る

条件 分岐のブロック

条件を表すブロックの例

左と右のデータが同じか？

左のデータの方が右のデータより小さいか？

ボタンが押されているか？

条件を組み合わせることもできる

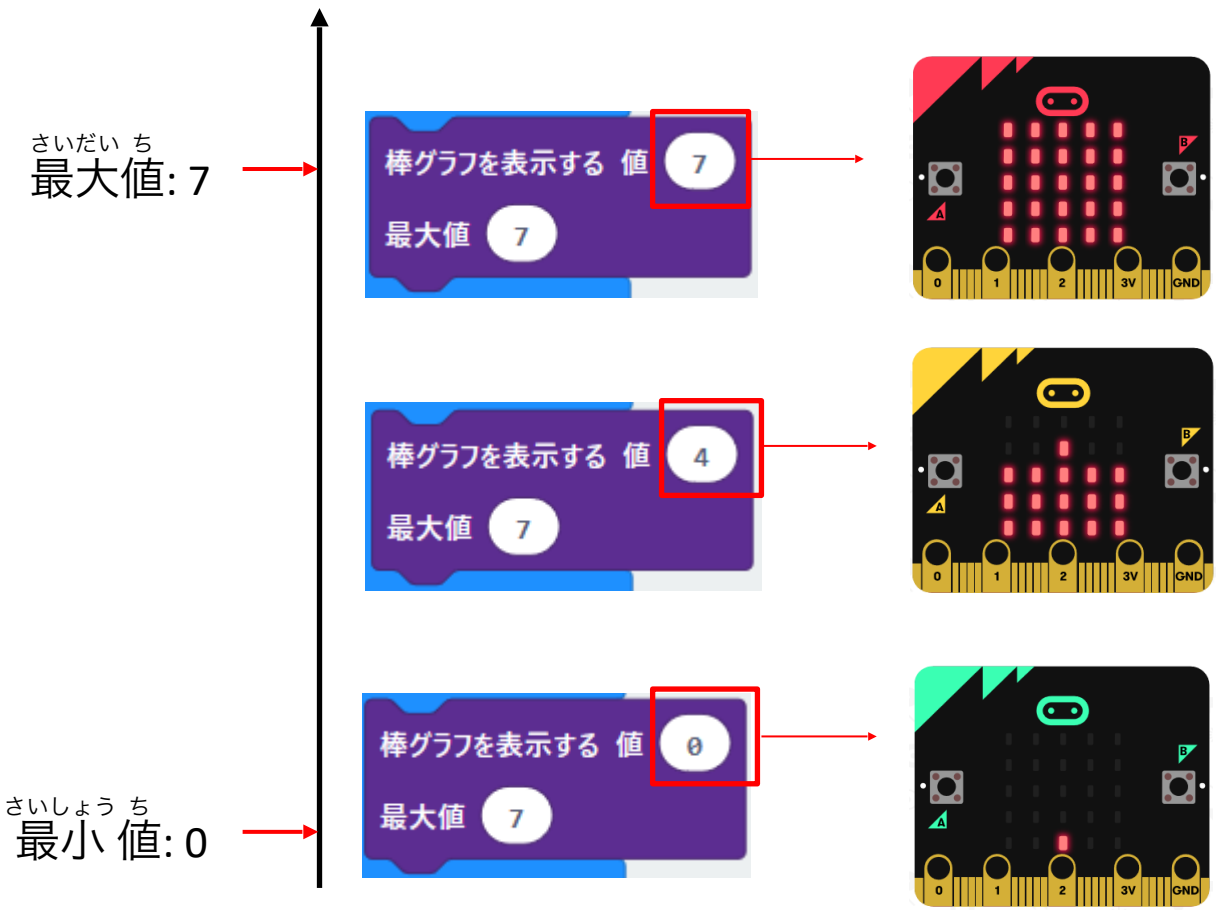
条件 分岐のブロック

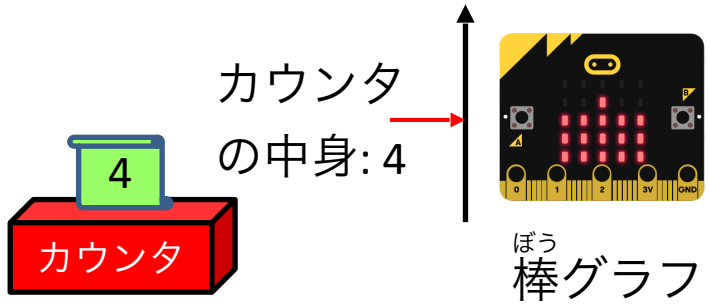
【補足】
MakeCodeでは「条件判断」と呼んでいる

条件が成り立っている場合、中の処理を行う

ここをおすと条件が増やせる

（予備知識）LED棒グラフの使い方

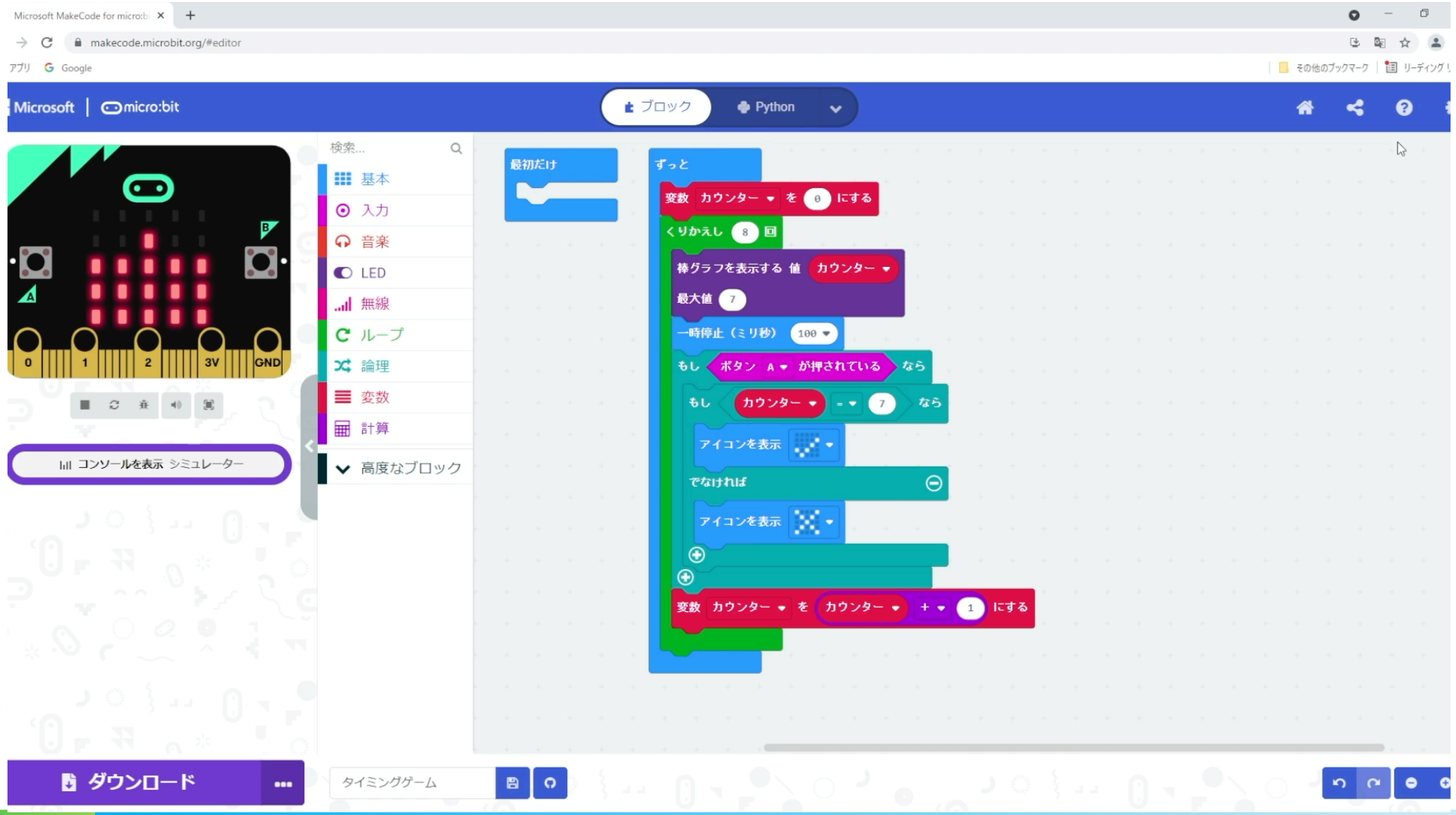




0～最大値までの数字をLEDの光る量で表現

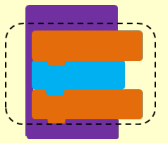
ここまでの知識(ちしき)を使ってゲームを作ってみよう

動画



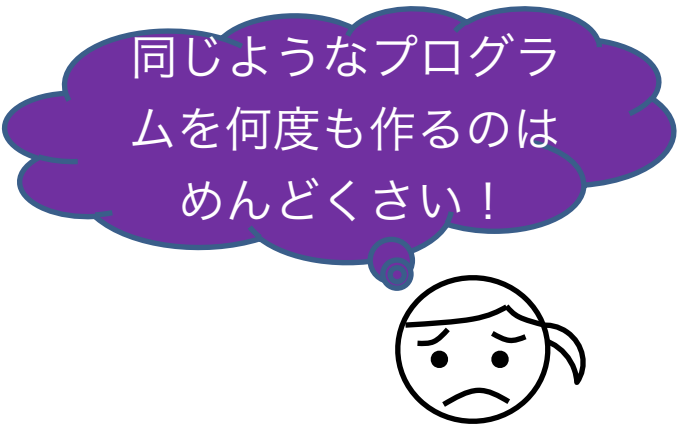
プログラミングの基本④: 関数

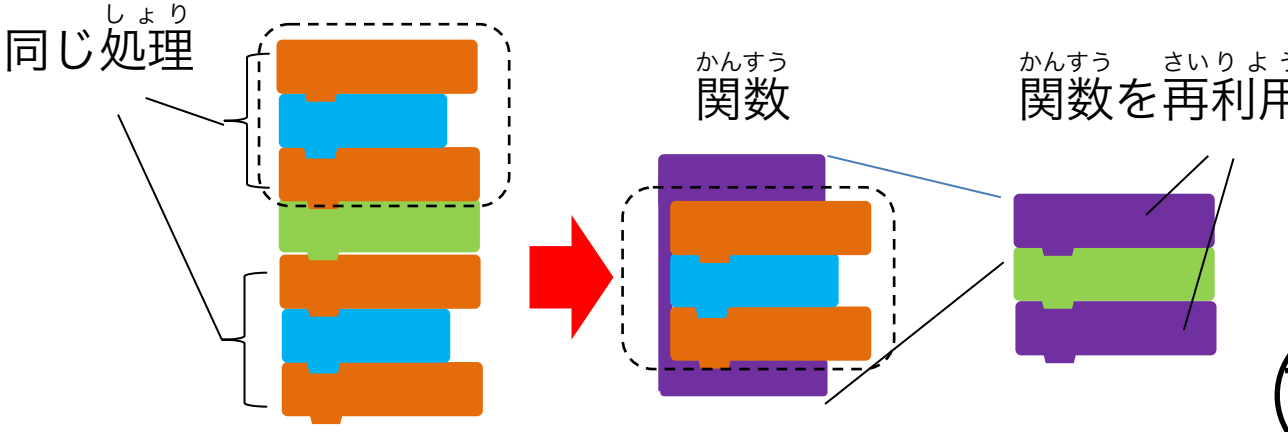
関数 → まとめる

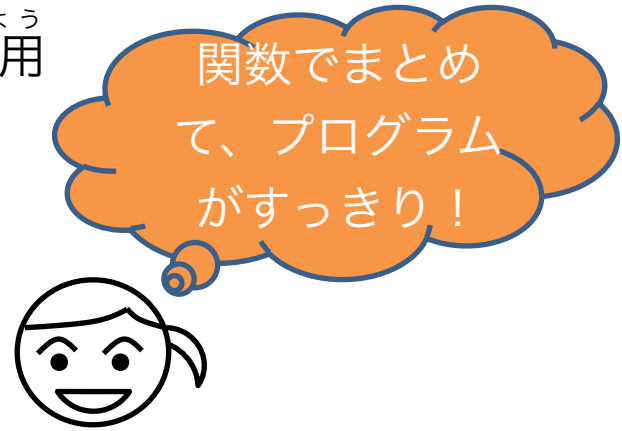


関数とは？

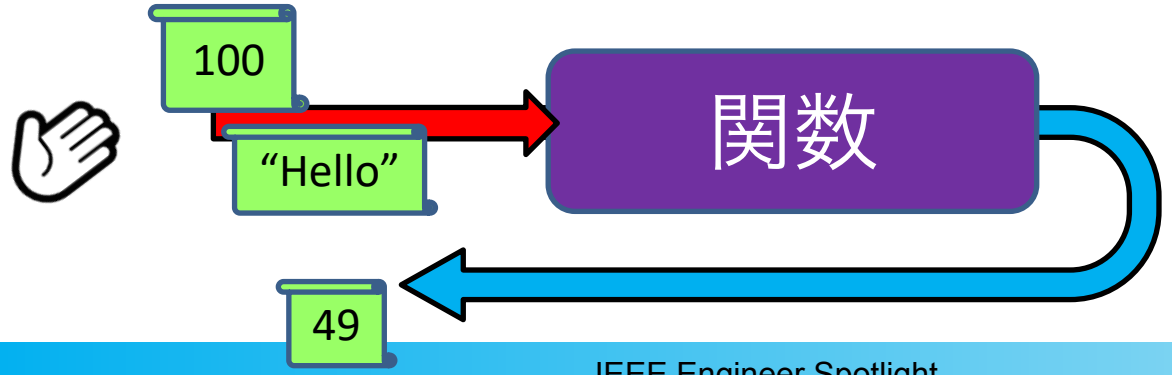
- 機能ごとにプログラムをまとめ、使いやすく・わかりやすくするもの







データを関数に渡したり、関数からデータを返したりできる (渡す / 返すデータがなくてもOK)



処理内容は同じだが、処理するデータ自体は変えたいというときに便利

関数を使ってゲームの機能をまとめよう

関数のブロック

関数の名前

関数に渡すデータ
(関数の中では**変数**として扱う)

関数を実際に使うときのブロック

関数に渡すデータ

（ちょっと難しい）予備知識

真偽値

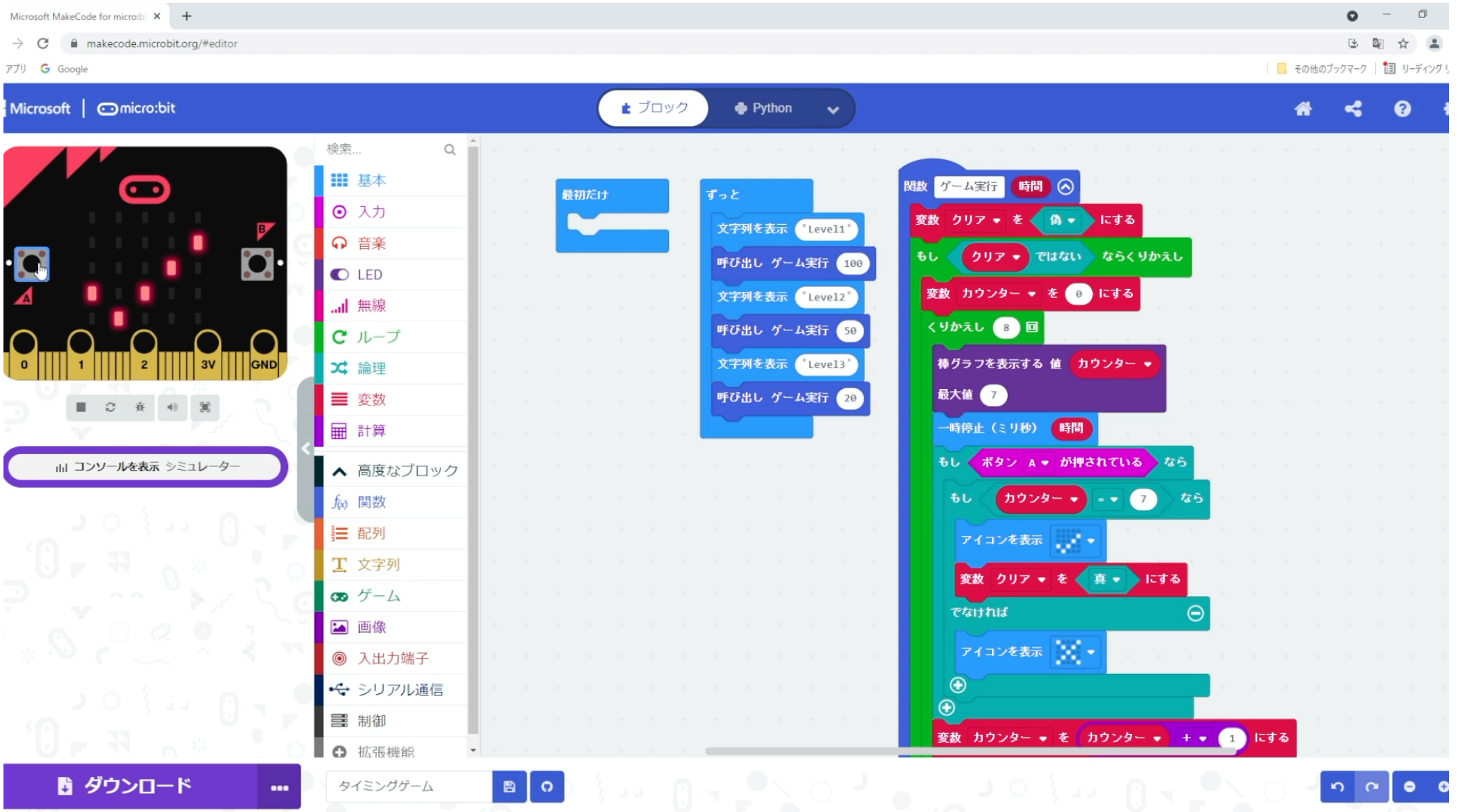
「条件が成り立っている」ことを表すデータ

「条件が成り立っていない」ことを表すデータ

変数に入れることもできる

関数を使ってゲームの機能をまとめよう

動画



プログラミングの基本の復習

- 変数とは？ (1) のいれもの
- ループとは？ (2) のくり返し
- 条件分岐とは？ (3) によって処理を変えること

A

(1): プログラム
(2): 同じ処理
(3): 計算結果

B

(1): データ(数値や文字など)
(2): 異なる処理
(3): 設定した条件

C

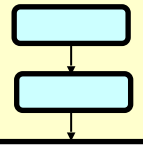
(1): データ(数値や文字など)
(2): 同じ処理
(3): 設定した条件

適切な組み合わせを選んでください！

プログラミングの基本のまとめ

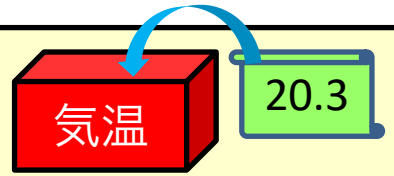
1. 処理の順番

処理の順番 → 上から



2. 変数と計算

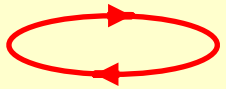
変数 → いれもの



計算　＋　－　×　÷　・・・

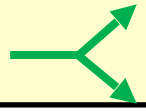
3. ループと条件分岐

ループ → くり返し



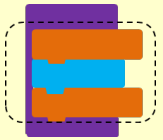
条件 → もし…だったら ？

分岐 → 枝分かれ



4. 関数

関数 → まとめる



【応用編】ゲームをさらにアレンジしてみよう

例えば…

- いまのプログラムでは、何回スイッチを押したかを人間が確認しているので、これもコンピュータにやらせる
- 各レベルでクリアまでにスイッチを押した回数から最終的なスコアを計算する
- クリアしたときに音を出す

など

プログラミング例は参考資料を見てください。
電池をつなぐ場合の情報も載せています。

2

電子工作を体験！

～自動開閉パーティションを作って学ぼう～

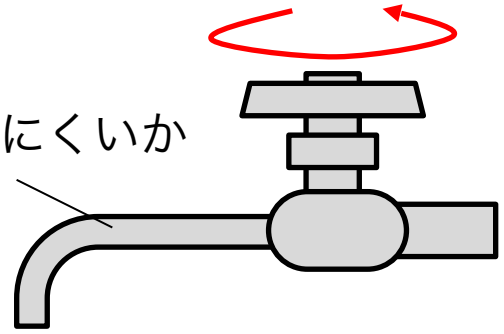
電圧・電流・抵抗

- 電圧
  - 電気を流そうとする力の量
- 電流
  - 流れる電気の量
- 抵抗
  - どれくらい電気が流れにくいかという量

【補足】単位について
電圧: V (ボルト)
電流: A (アンペア)
抵抗: Ω (オーム)

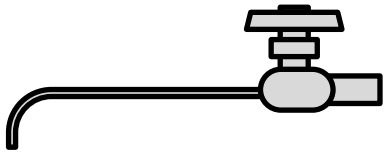
イメージ

電圧: どれくらい蛇口をひねるか

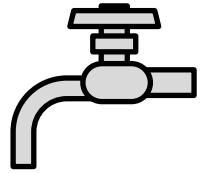


抵抗: どれくらい水が流れにくいか

抵抗が大きい



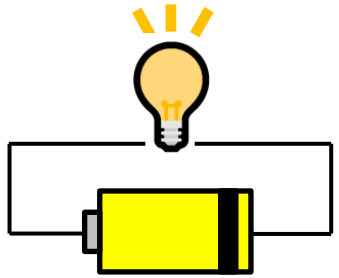
抵抗が小さい



電流: どれくらい水が流れるか

電子回路ってなんだ？

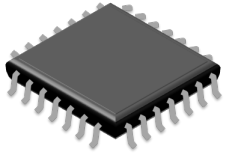
回路: 電気の通り道

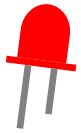


回路を作ると電気が流れる

電子回路: 一般的には半導体という部品を使った回路のこと

ダイオード、トランジスタ、LEDなど





マイコンも半導体！

今日はこれだけわかればOK

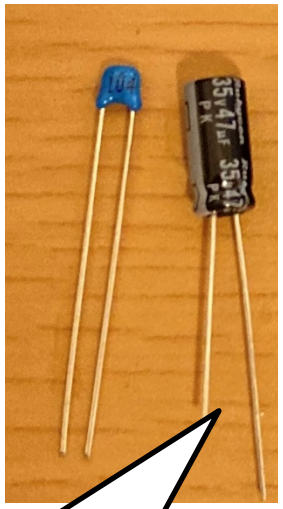
電子回路を作る → いろいろな部品をつなげて、電気の通り道を作る

部品の例

抵抗器

カラーコード: 色で抵抗値がわかる

コンデンサ

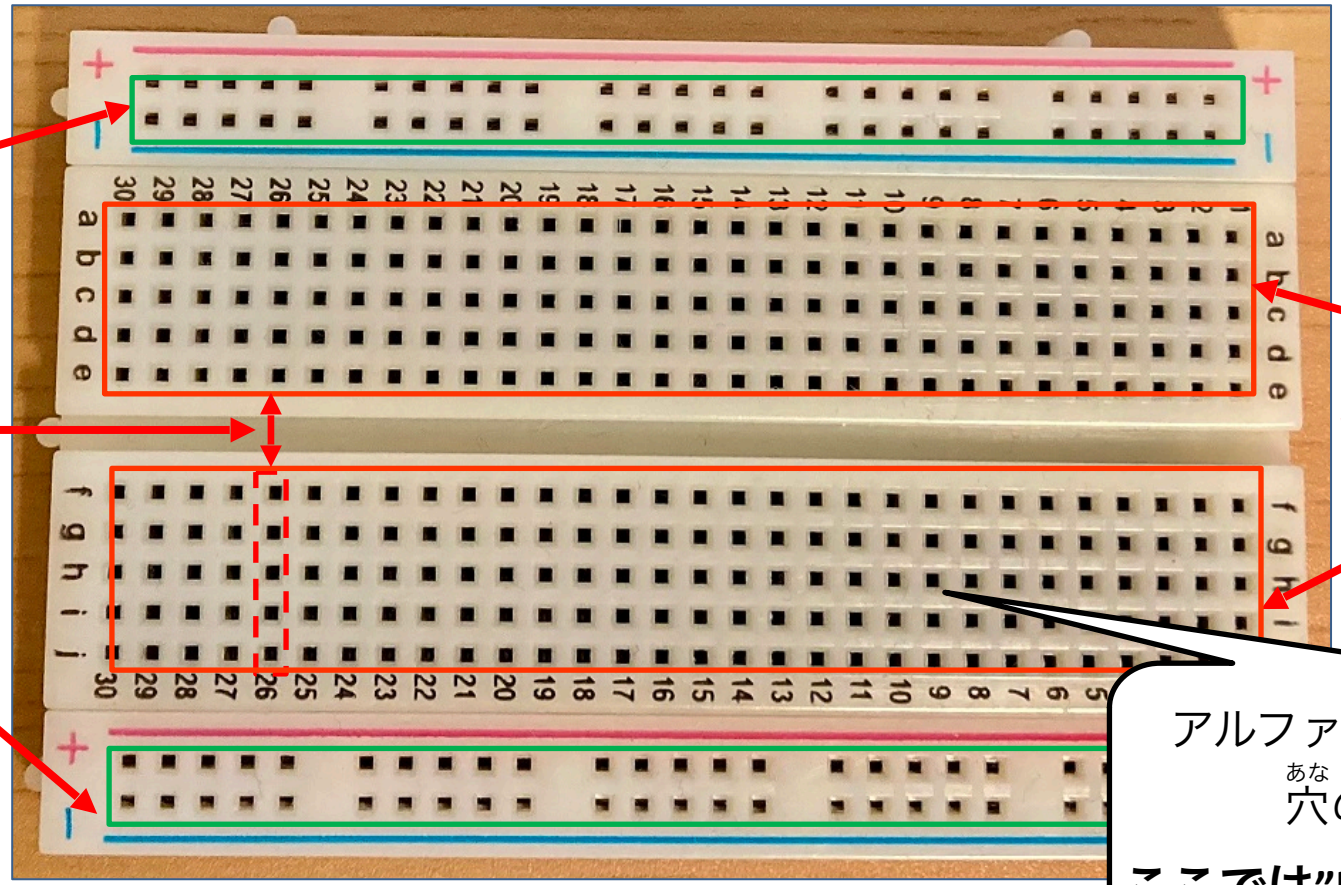


LED

取り付ける向きがある部品もあるので気をつけよう！

ブレッドボードでかんたんに回路を作成！

- ブレッドボードとは？
  - かんたんに部品の抜き差しができ、回路の試作によく使われる



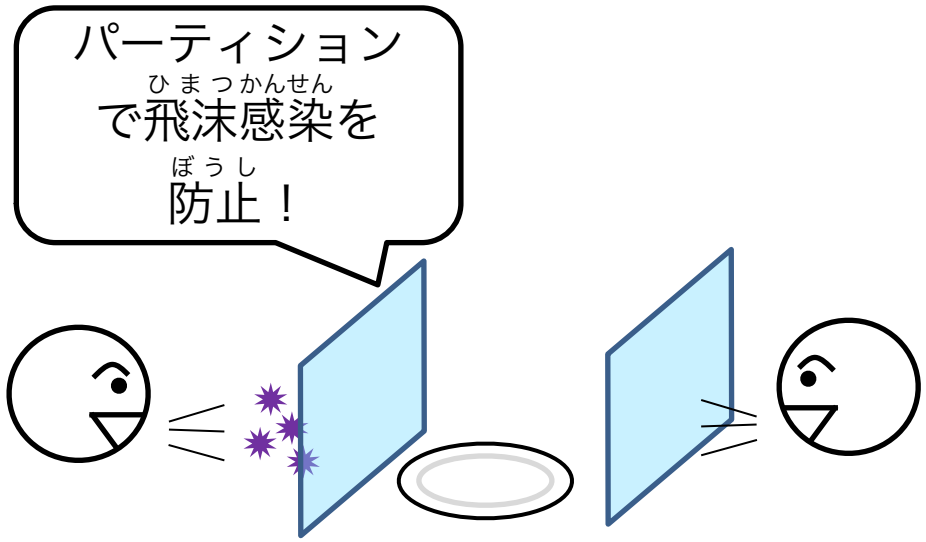
ここは、それぞれ横にすべて
つながっている
(縦にはつながっていない)

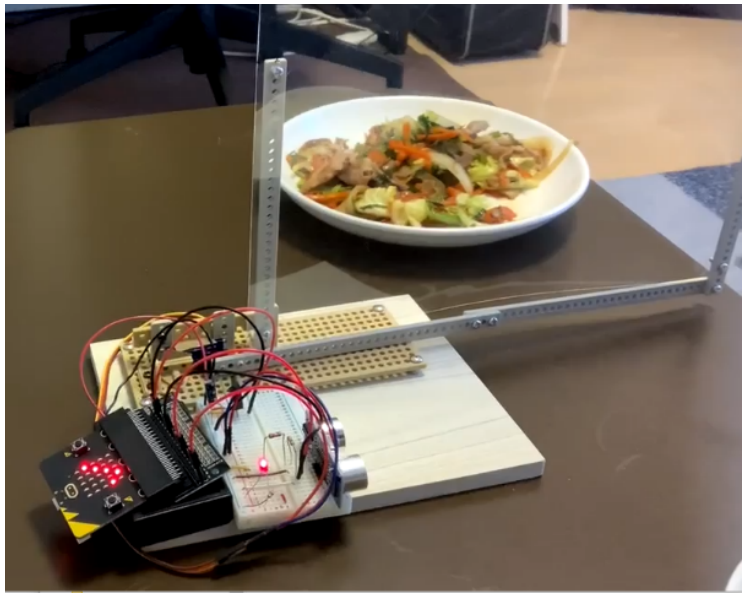
この間はつながっていない

ここは、それぞれ縦にすべて
つながっている
(横にはつながっていない)

アルファベット(縦)と数字(横)で
穴の位置を表せる。
ここでは”h-9”のように表記する。

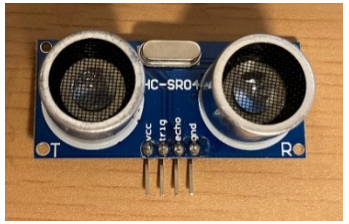
作るもの: 自動開閉パーティション



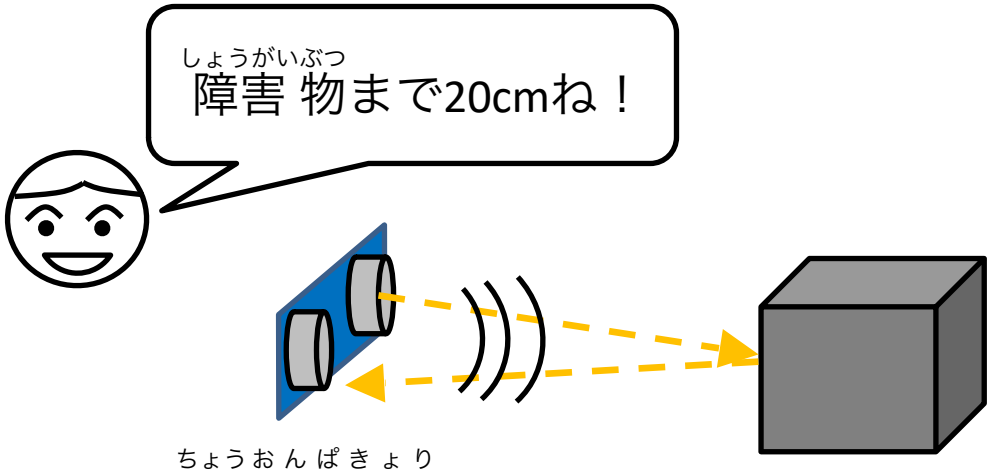


予備知識

- この工作で使う重要な部品
  - サーボモータ
    - 位置・角度制御が得意なモータ
  - 超 音波距離センサ
    - 超 音波の反射時間によって距離を測定できるセンサ



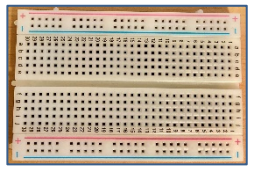
サーボモータ



超 音波距離センサ

サーボモータと超音波距離センサを動かす回路を作ってみよう

ここで使うもの



ブレッドボード

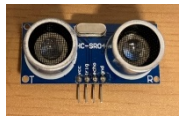
10kΩ抵抗器
(カラーコード: 茶黒橙金)

20kΩ抵抗器
(カラーコード: 赤黒橙金)

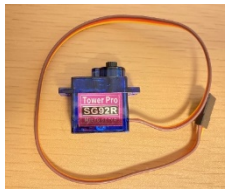
330Ω抵抗器
(カラーコード: 橙橙茶

赤色LED

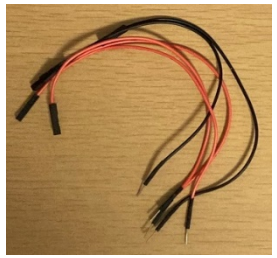
電池ボックス
(単三3本)



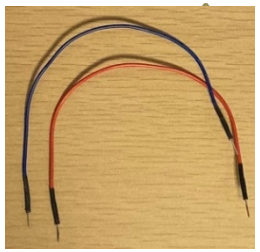
超 音波距離センサ



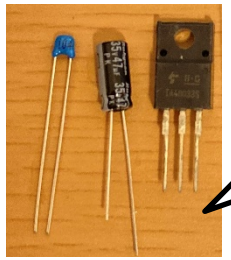
サーボモータ



オス-メスジャンパー線
(5本)



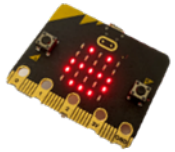
オス-オスジャンパー線
(2本)



三端子レギュレータ
と付属のコンデンサ

micro:bitに必要な電圧を作る

ジャンパー線セット



micro:bitと
ブレークアウトボード

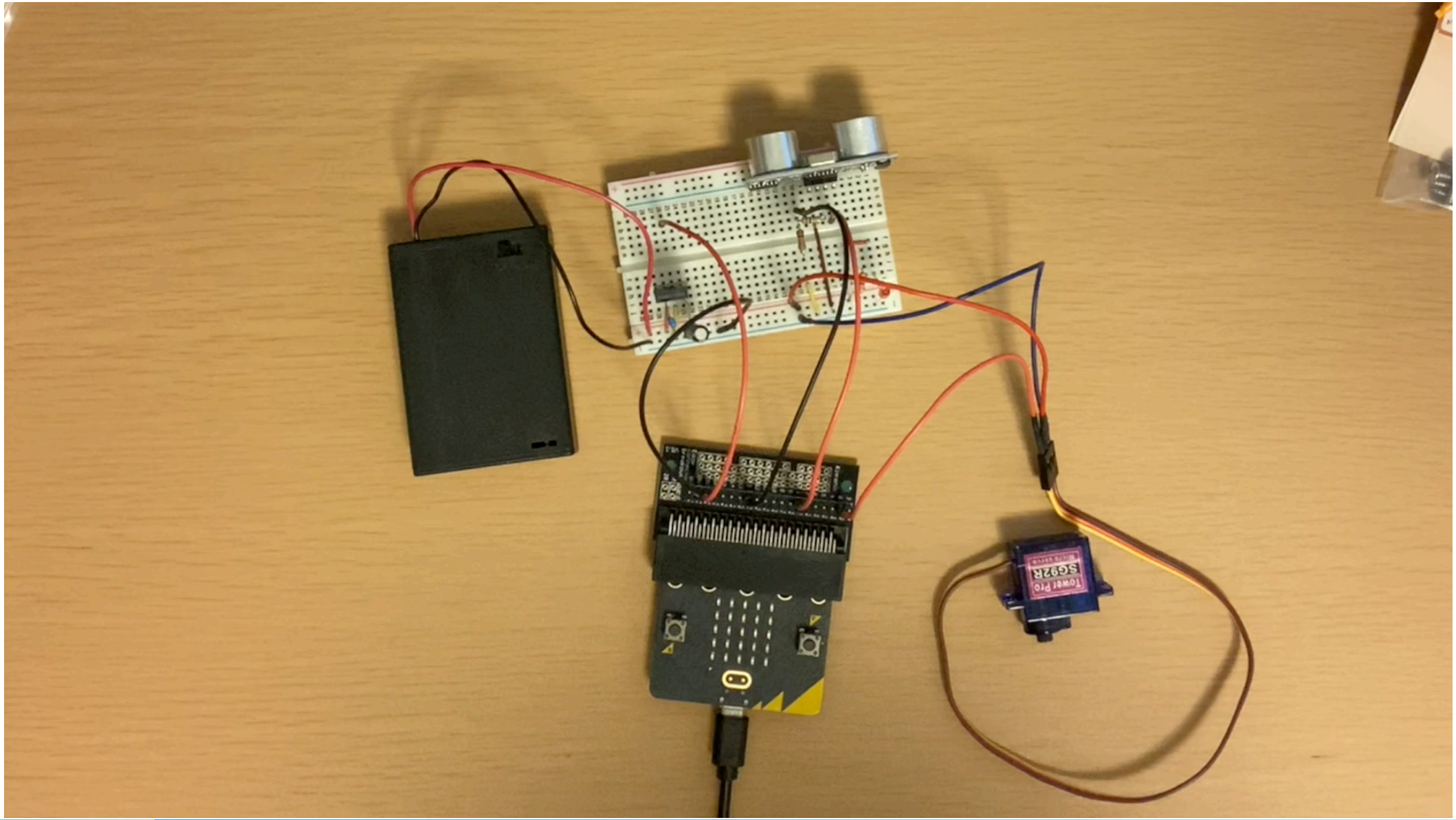
【注意】 回路作成は十分に気をつけよう！

回路を間違えると、**部品はかんたんに壊れる**。

少しでもおかしいと思ったらすぐに電源を切って、回路をチェック！

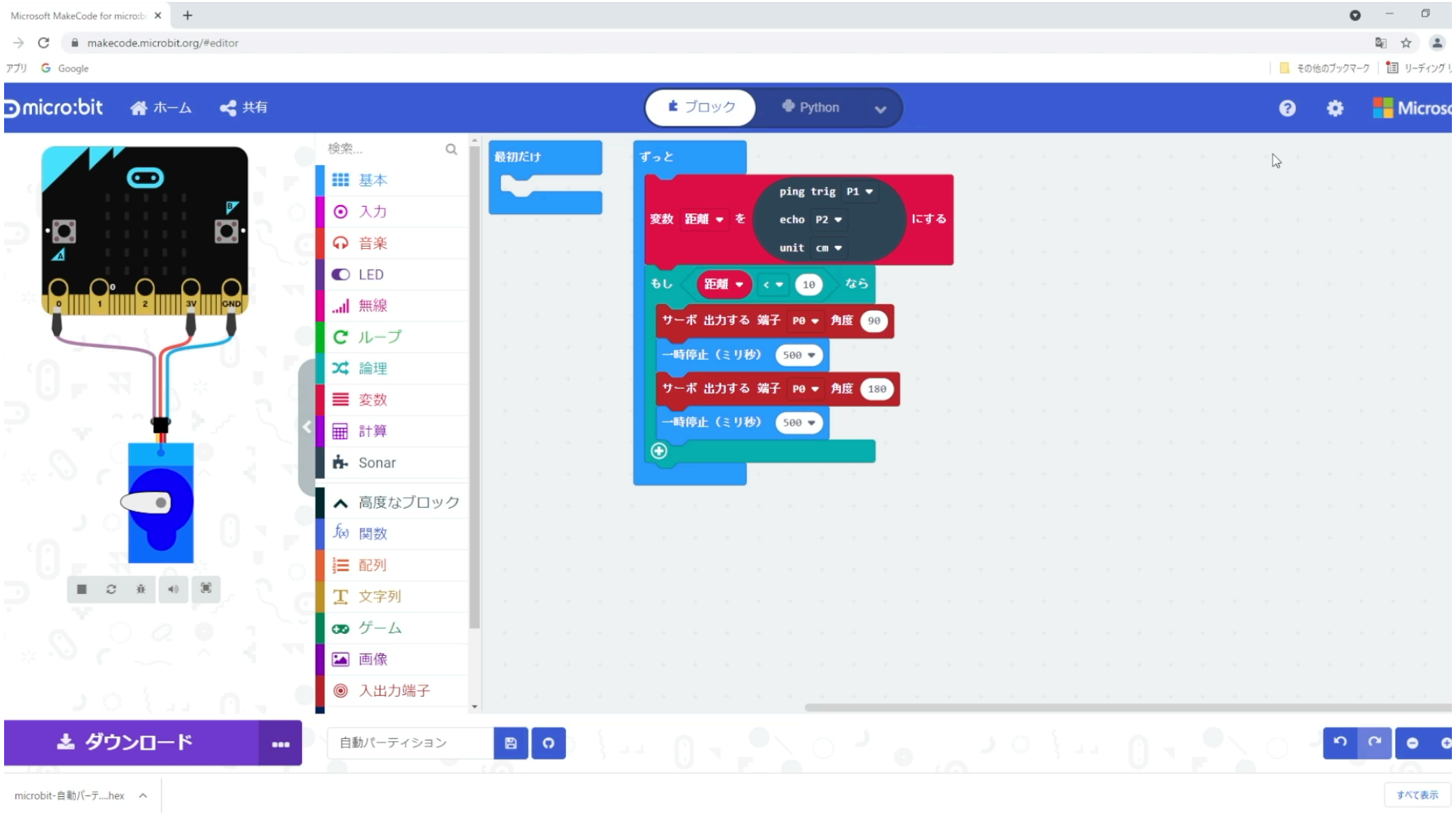
サーボモータと超音波距離センサを動かす回路を作ってみよう

動画



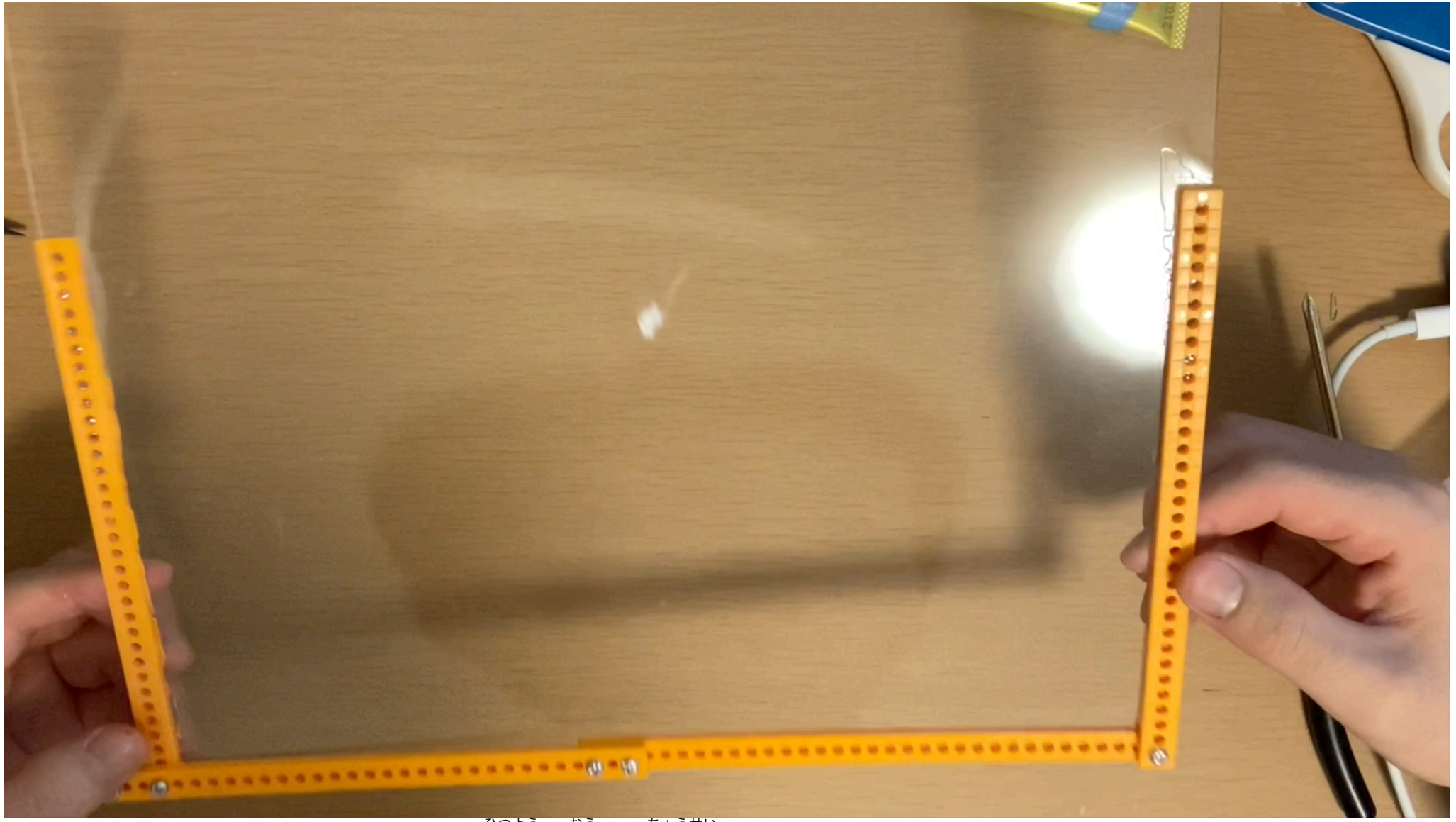
サーボモータと超音波距離センサを動かすプログラム

動画



パーティション用のアームを作る

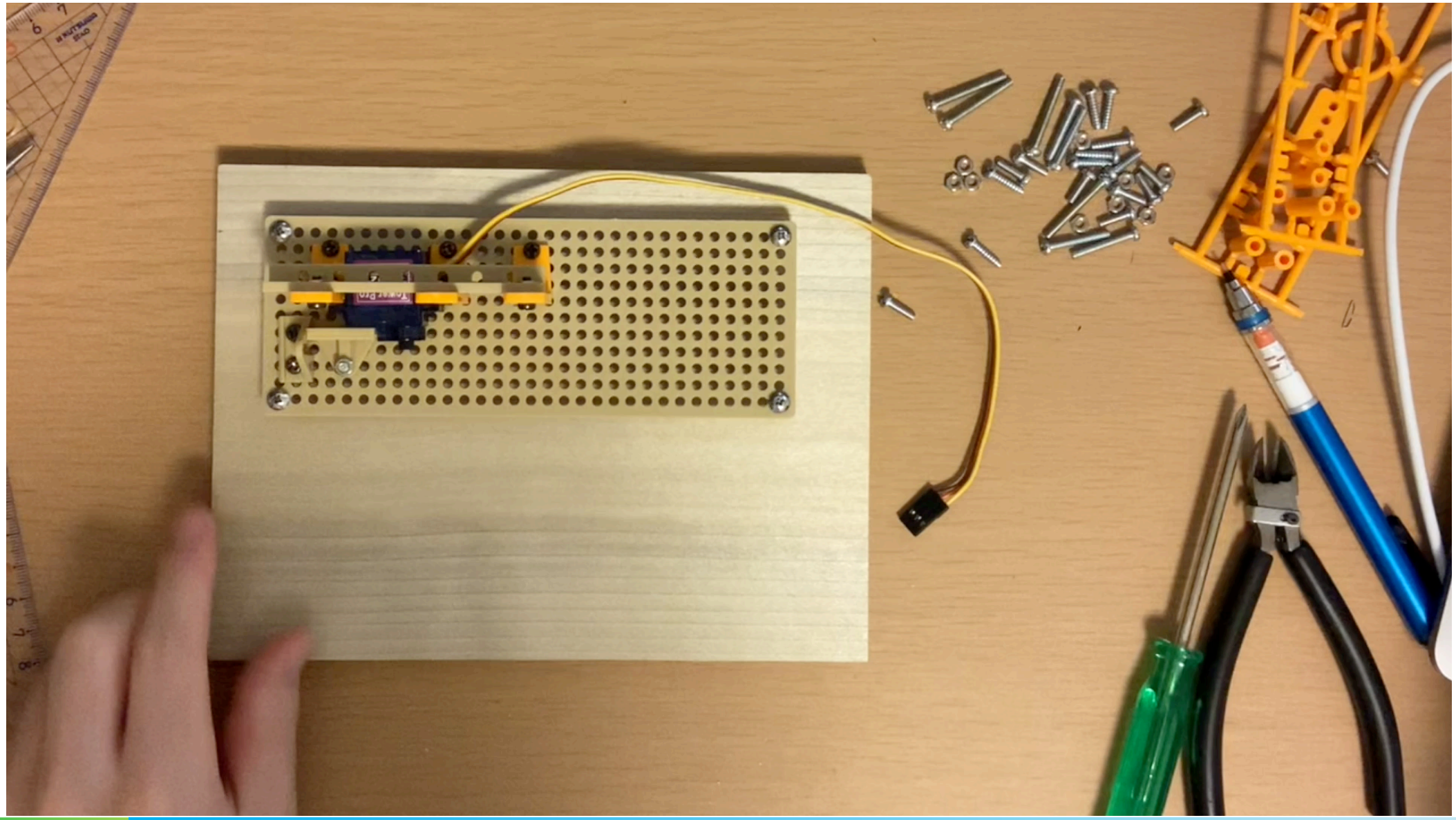
動画



パーティションの大きさは必要(ひつよう)に応(おう)じて調整(ちょうせい)してください。(このサイズより大きくはしないように)

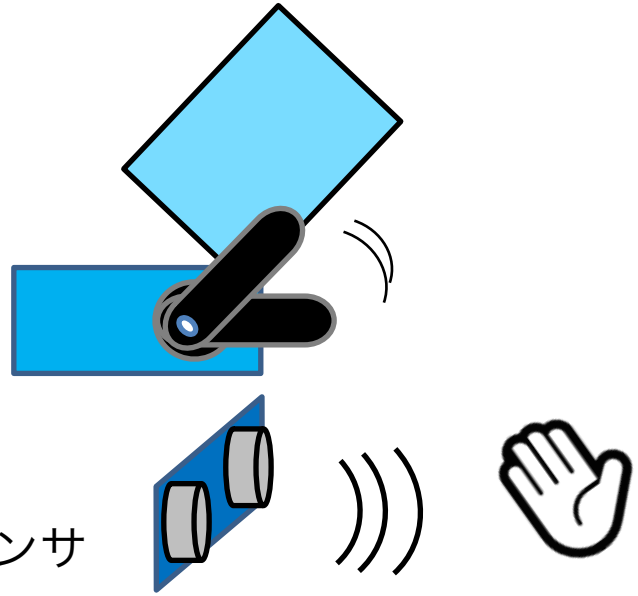
土台を作る

動画



自動開閉パーティションのプログラムを作る

基本的な仕組み



超 音波距離センサで近いものを検出

↓

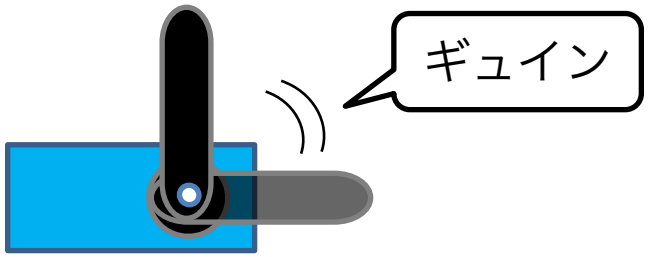
サーボモータを動かしてパーティションを開ける

↓

ある一定間隔で近くに何かあるかを確認し、なければサーボモータを動かし、パーティションを閉じる

ずっとくり返す

サーボモータはふつうに角度を与えると、動きが速すぎる

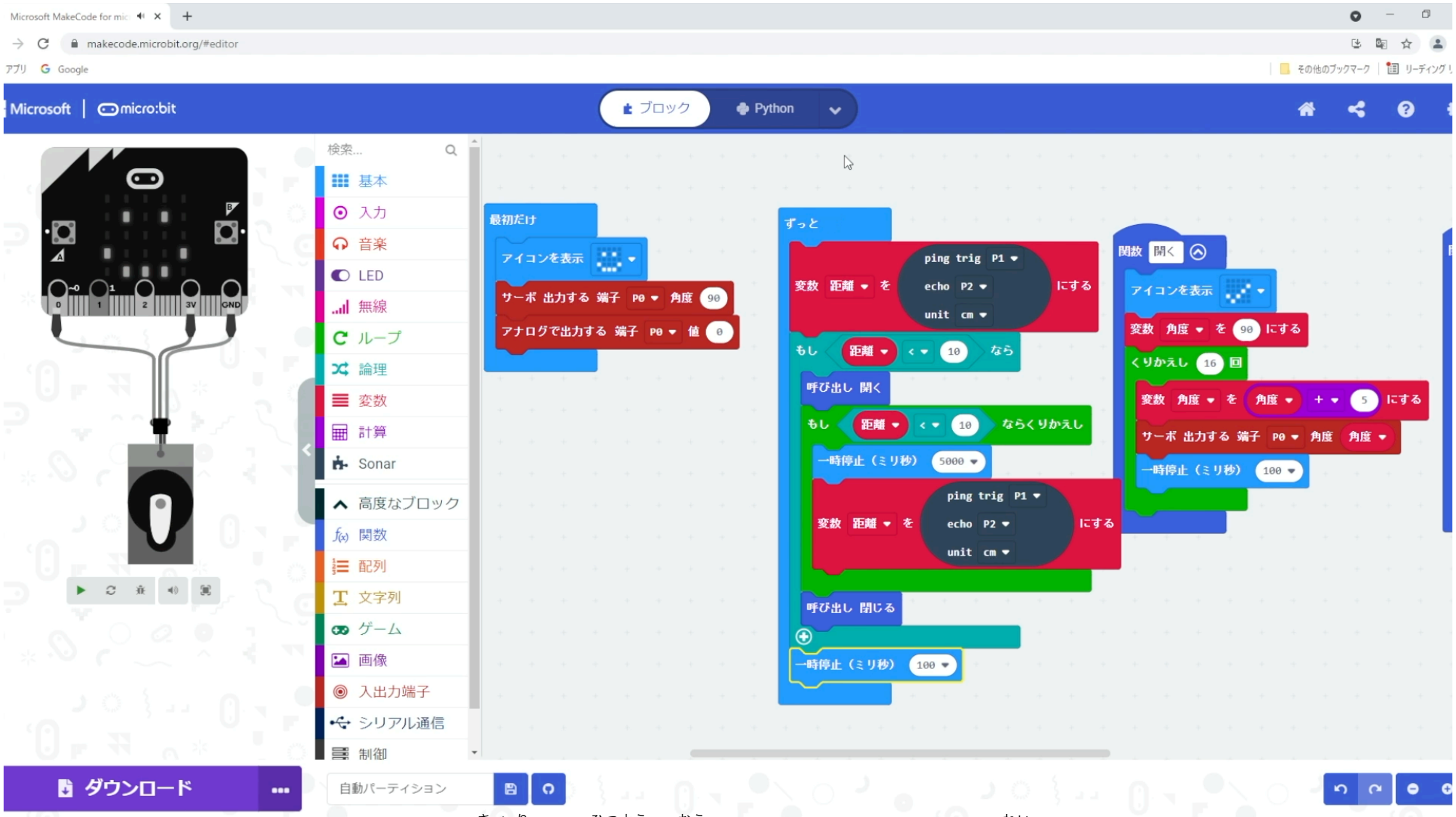


短い時間間隔で少しずつ動かし、速度を調整する

ここでは、90°～180°の間を5°ずつ約0.1秒ごとに動かす

自動開閉(かいへい)パーティションのプログラムを作る

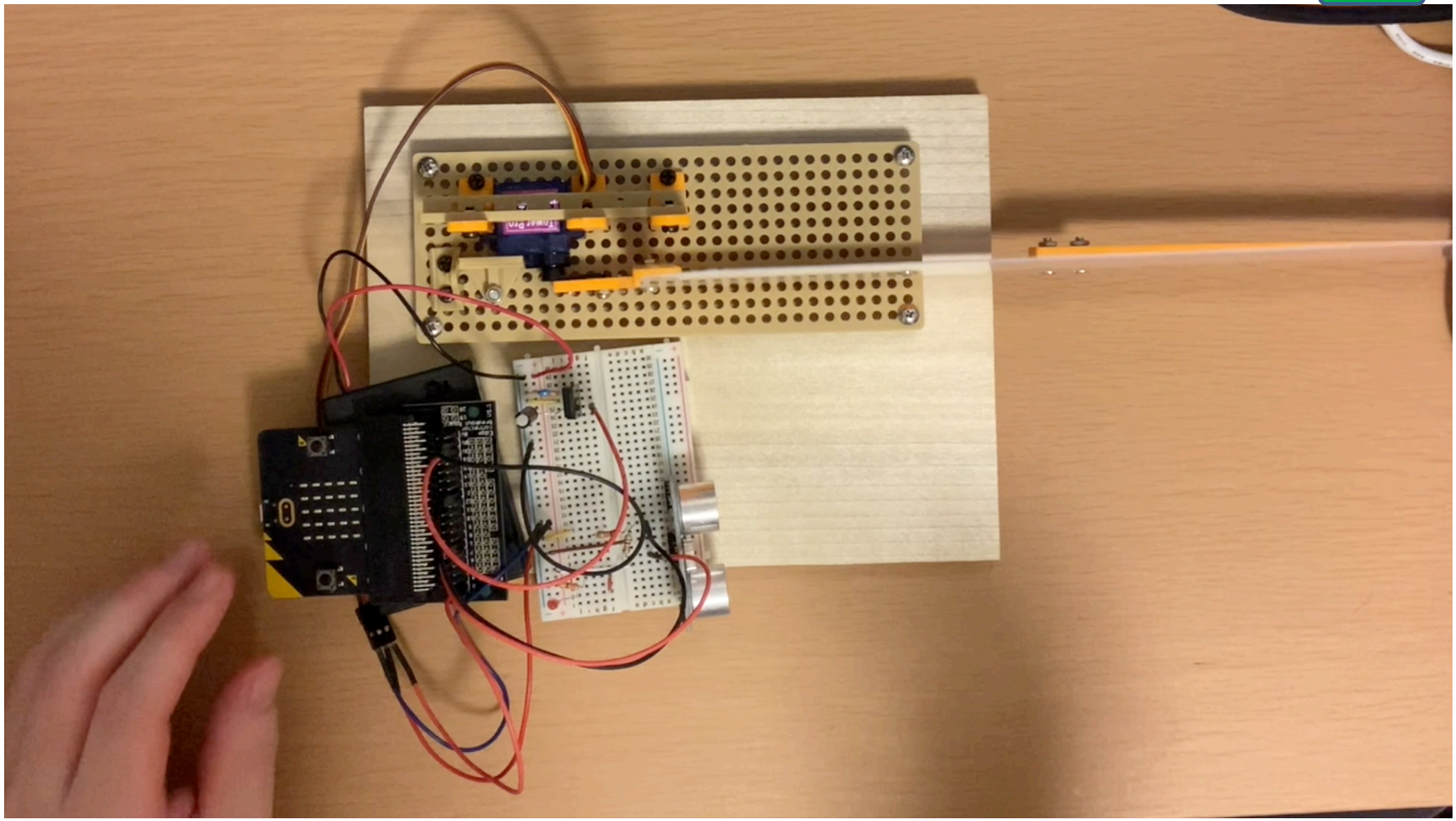
動画



パーティションを開く距離(きょり)は、必要(ひつよう)に応(おう)じて変えてください。(例(れい): 20cm)

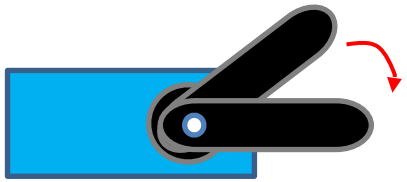
仕上げ（しあ）

動画

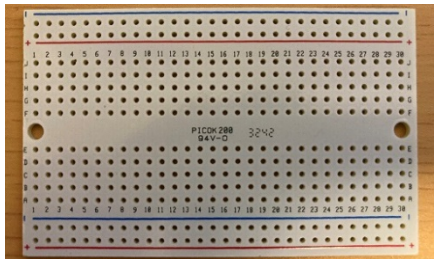


補足

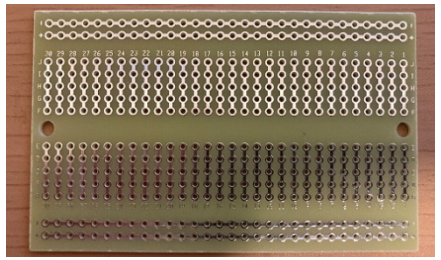
- 動きがおかしいなと思ったら
  - 回路の接続をチェック (ジャンパー線が抜けていたりなど)
  - 電池を交換してみる
- サーボモータがガタつくときは
  - 接着剤を使ったり、間に厚紙などを挟んだりする
- 急に動くパーティションに注意
  - パーティションが初期位置でないところで電源を入れると、最初に初期位置(90°)に戻ろうとするので、勢いよくパーティションが動いてしまう
  - 必要に応じて一度パーティションをサーボモータから取り外すなどの対応をする
  - 電源を切るときは、初期位置に戻った状態で
- ブレッドボードだと配線が抜けやすいので、はんだ付けしてしまうのも手
  - ブレッドボードと同じ配線の基板もある



電源を入れた瞬間、勢いよくここに戻ろうとする



表面

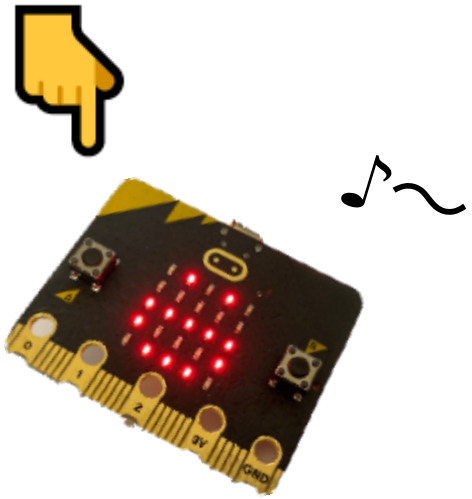


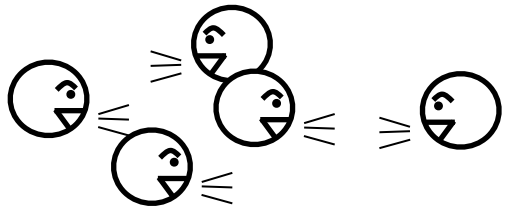
裏面

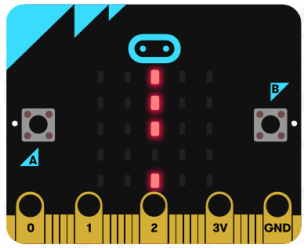
ブレッドボードと同じ配線のユニバーサル基板

オリジナルの工作をするときの発想法の例

- イベントとその結果(処理)を考えよう！
  - 例1: イベント「ボタンAが押されたとき」 → 結果「音を鳴らす」
  - 例2: イベント「まわりの音がうるさくなったとき」 → 結果 「LEDでアイコンを表示する」

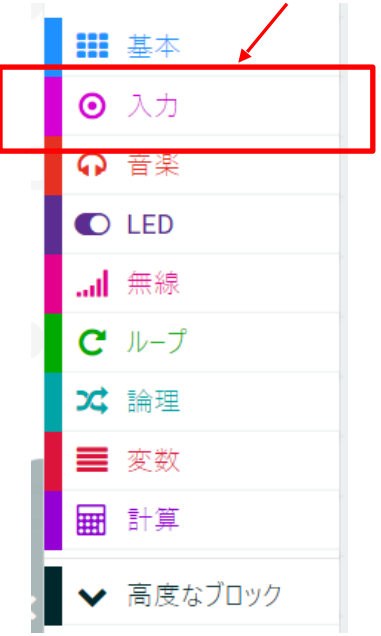






micro:bitでどんなことができるか

イベントの例



ボタン入力

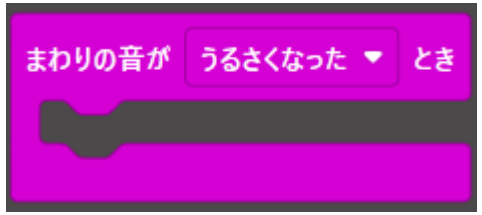
Bボタンや、AとBの同時押し(A+B)も選択できる

micro:bitの動きや傾き

傾きなども条件に設定できる

など

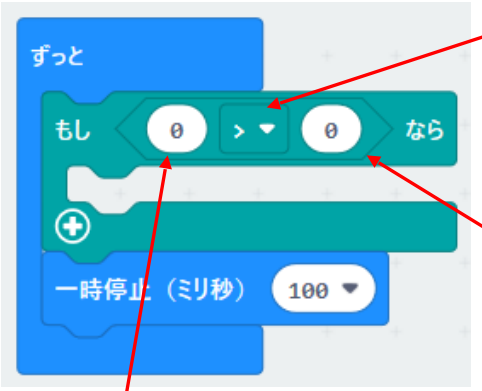
音量 (micro:bit V2のみ対応)



「静かになったとき」も選択できる

どれくらいでうるさいと判断するかはこのブロックで設定できる

条件 分岐と組み合わせる



条件を選択

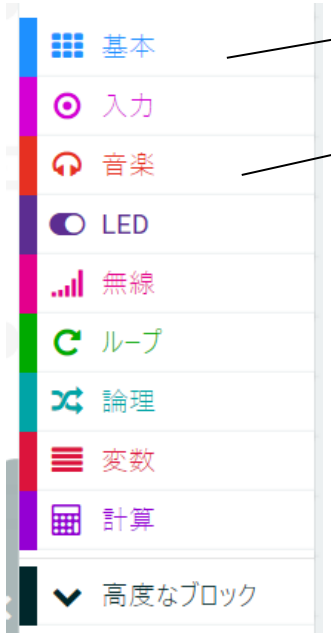
適切な数値を設定

ここに以下のようなデータを入れる

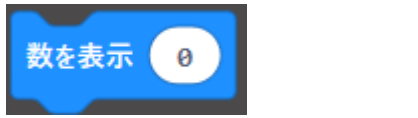
micro:bit V2のみ対応

micro:bitでどんなことができるか

処理の例



LED表示



クリックした部分のLEDが光る

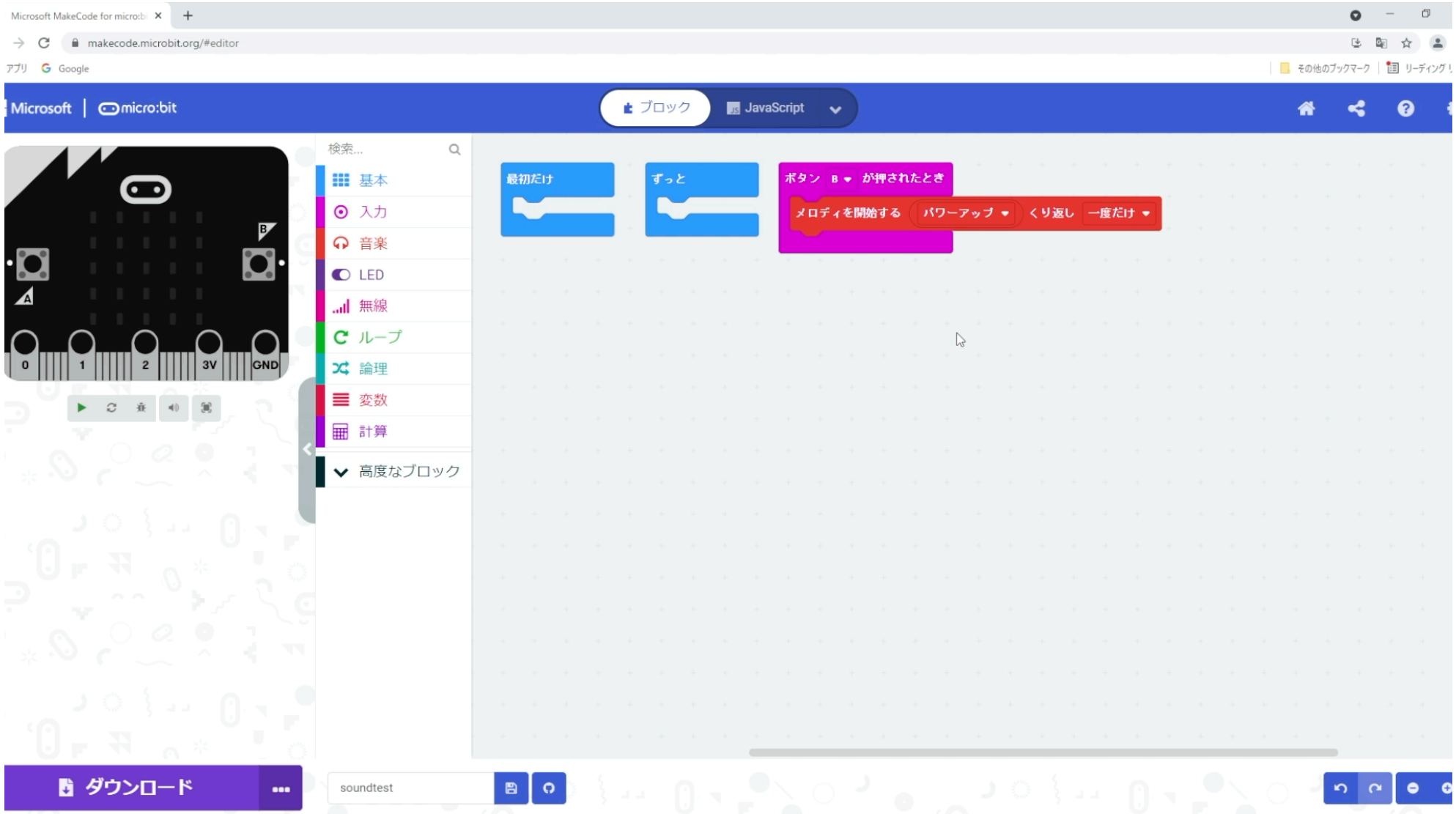
音を鳴らす

様々なメロディーがある

自分だけのメロディーを作ってみよう！

micro:bit V2では、そのまま音を鳴らせる
V1の場合は参考資料を参照

音を鳴らす例



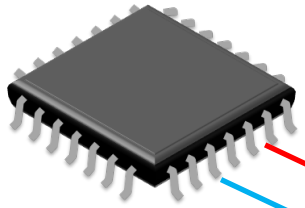
micro:bitでできることは、ここで紹介しきれないぐらいたくさんある

3 【上級(じょうきゅう) 編(へん)】 Arduinoを使(つか)ったLED点滅(てんめつ)実験(じっけん)

Arduinoとは

- マイコンボードの一つ
  - 様々な電子部品を制御できる

micro:bitとは搭載している
コンピュータが違う



マイコンは端子の
電圧を制御できる

5V (HIGH)

0V (LOW)

- プログラミングはC言語を使う

```
void setup() {
  pinMode(13, OUTPUT);  // 13番ピンを出力に設定
}
void loop() {
  digitalWrite(13, HIGH);   // 13番ピンをHIGH(5V)に
  delay(100);               // 0.1秒一時停止
  digitalWrite(13, LOW);    // 13番ピンをLOW(0V)に
  delay(100);               // 0.1秒一時停止
}
```

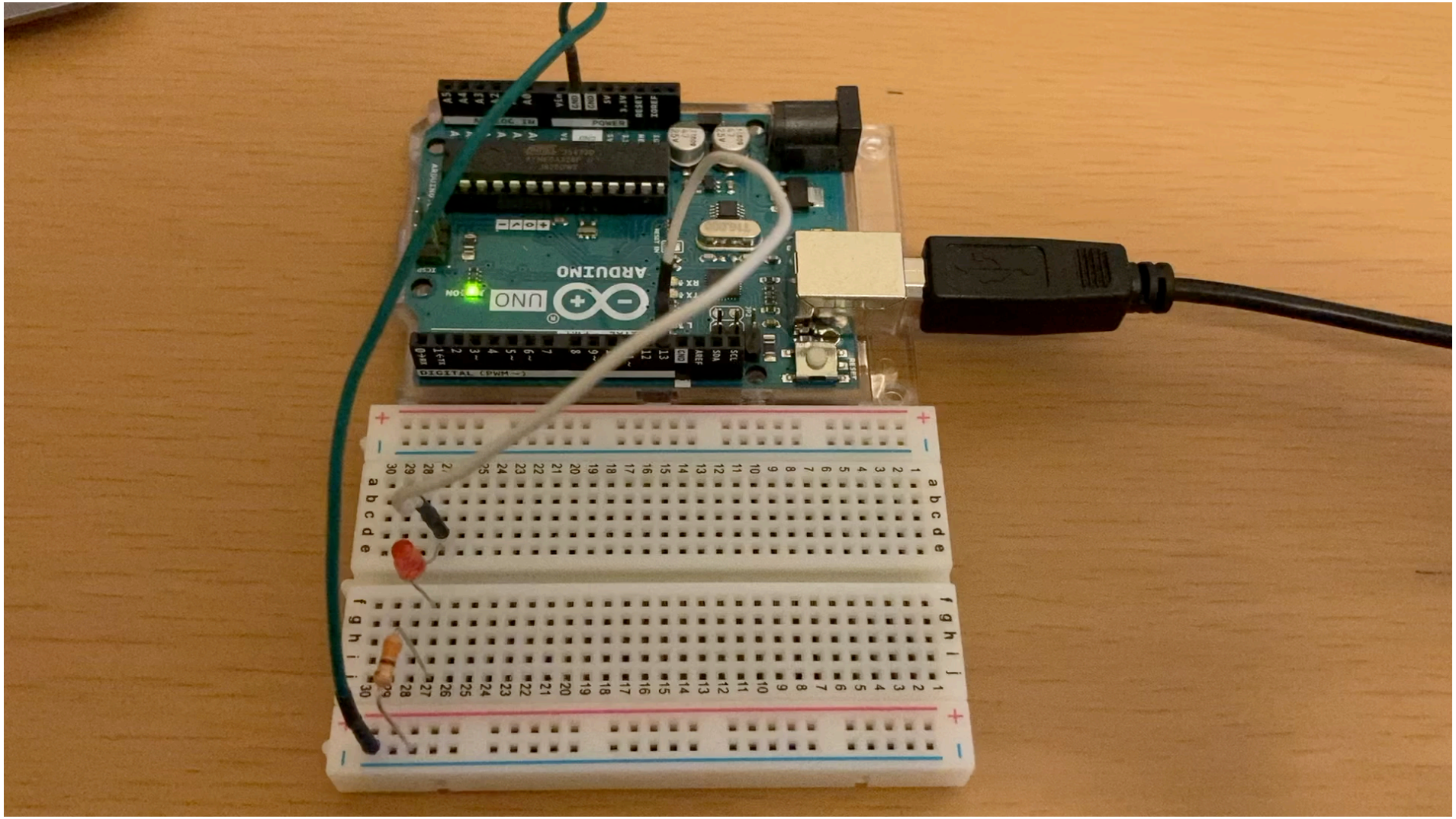
IEEE TryEngineering

- IEEEが提供するオンライン学習コンテンツ
- Arduinoを使ったLED点灯実験も掲載
  - https://tryengineering.org/teacher/arduino-blink-challenge/

LEDの点滅実験

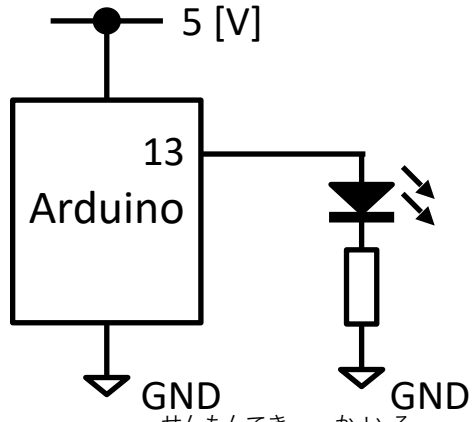
プログラミング方法などは参考資料を参照

(Webにも様々な情報が載っています。)

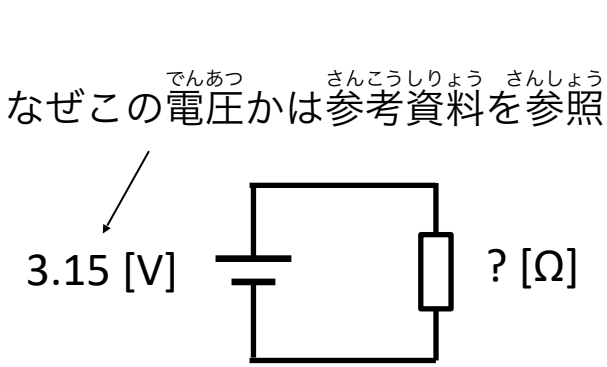


回路設計を体験！

- 中学校の理科(物理)で習う「**オームの法則**」を使ってLEDの点灯回路を設計する



ちょっと専門的な回路



LEDの影響を考慮したシンプルな回路 (端子から5V出力時)

LEDに流れる電流は10 [mA]ほどにしておきたい。
抵抗値はどのようにすればよいか？

オームの法則: 抵抗(R) = 電圧(V) / 電流(I)

R = 3.15 [V] / 0.010 [A] = 315 [Ω]

⇒ 近い値の330 [Ω]を選択

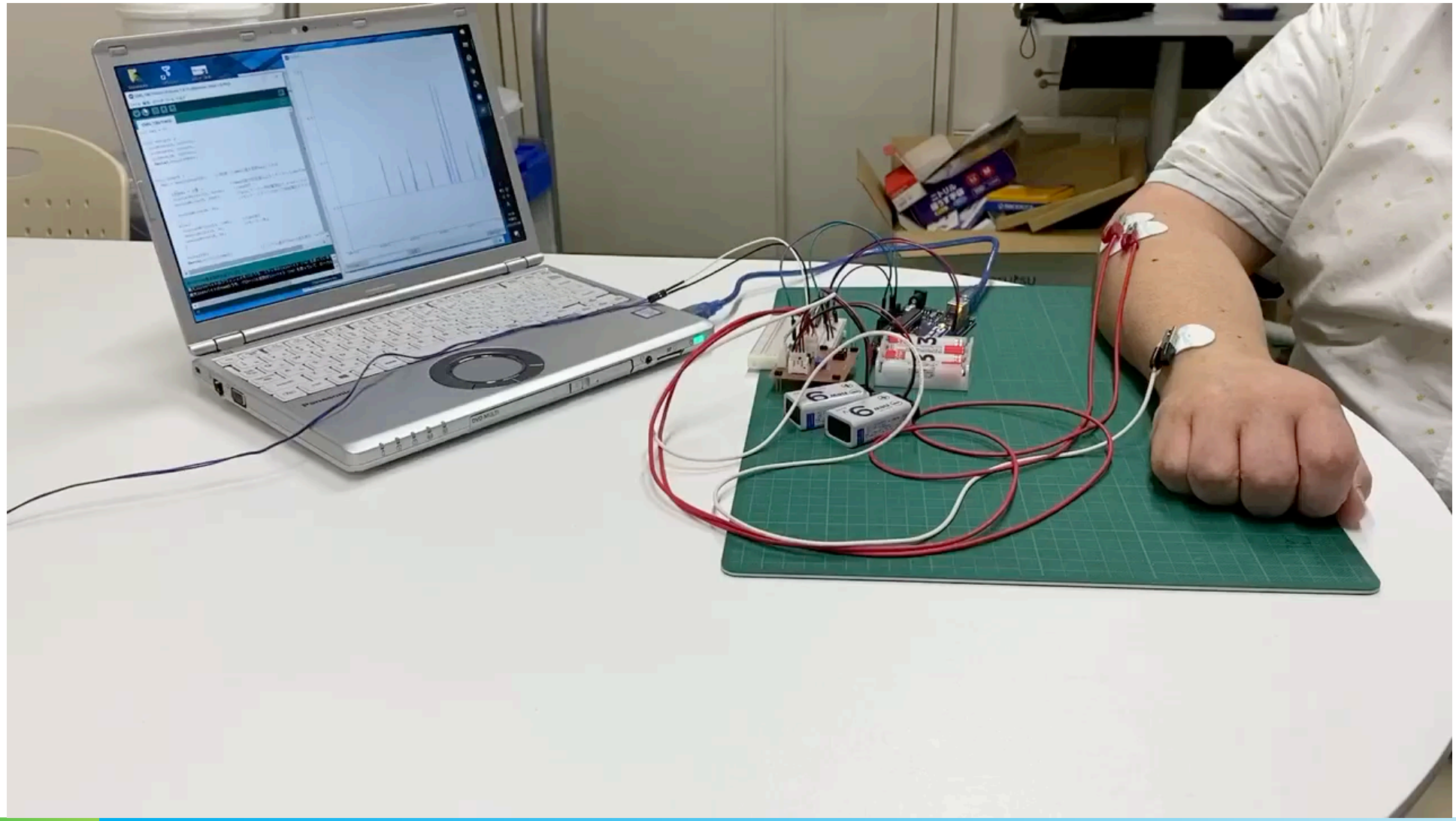
なぜ10 [mA]？

→ LEDやArduinoに流せる最大の電流を超えない＆明るすぎない程度に

4 Arduinoの応用例(おうようれい) 紹介(しょうかい)

体の電気信号を使ってムカデロボットを操作！

東京都立産業技術高等専門学校 医療福祉工学コース 吉田嵩先生からのご提供

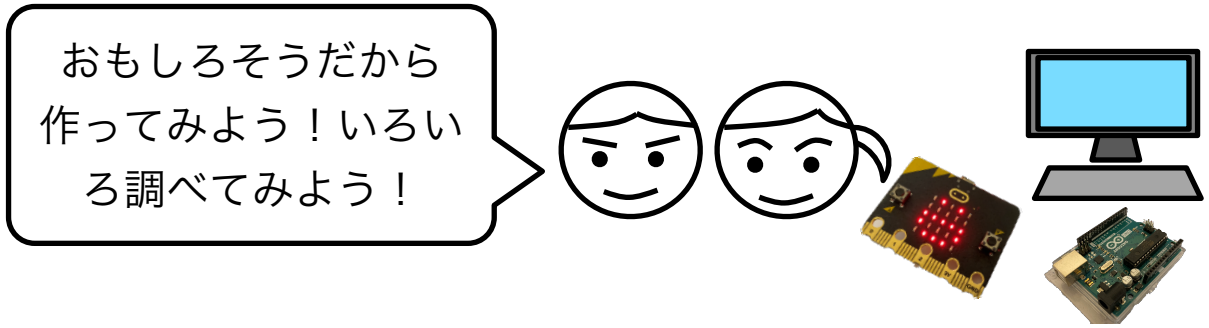


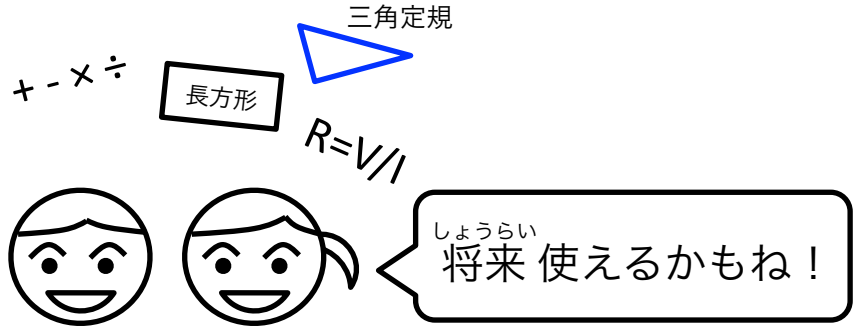
動画

5 おわりに

おわりに

- みなさんのアイデア実現のお手伝いするため、実際のプログラミングや工作例を見せながらみなさんの「できること」を増やしました
- ぜひ、この講演資料や参考資料、オンデマンド配信の動画を見返して、理解を深めてください
- みなさんが学校で習っている勉強は、決してテストや受験のためだけのものではないと実感できたのでは？ 将来 使える知識として興味を持って勉強してくれるとうれしいです





第17回 IEEE Engineer Spotlight

【参考資料】

動く！光る！音が鳴る！アイデアをカタチにしよう！

～今年の自由研究はプログラミングと電子工作だ！～

㈱東芝 研究開発センター / IEEE Tokyo Young Professionals Vice Chair

石垣 雄太朗

2021年7月24日

1 micro:bitプログラミングやタイミングゲームに関する補足情報（ほそくじょうほう）

画面の見方

ホームに戻る
他のプロジェクトを開くときなど

プログラミング画面
ここにブロックを置いてプログラミングを行う

【補足】
つながっているブロックから一つだけブロックを動かしたいときは、キーボードのCtrlキーを押しながらドラック＆ドロップする

ブロックはドラック＆ドロップで置く
(マウスを左ボタンを押したまま、マウスを動かして、左ボタンから指をはなす)

シミュレータ
micro:bit実物を動かさなくても、動作確認できる

戻るボタン
クリックすると一つ前の状態に戻る (操作を間違えてしまったときなどに利用)

ダウンロード
プログラムができたらここからmicro:bitにダウンロード

ブロックリスト
ここにあるブロックを組み合わせてプログラミングを行う

タイミングゲーム応用編

- 以下の機能を追加
  - クリアしたときや失敗したときなどに音を鳴らす
  - クリアまでボタンを押した回数をmicro:bitで数える
  - クリアまでの回数でスコアを計算して表示
    - スコアの計算式の例: Level1の回数×10 + Level2の回数×4 + Level3の回数
    - スコアが少ない方が勝ち
- 以下のURLからプログラムをコピーできる (やり方は次のページ)
  - https://makecode.microbit.org/_1uviKLM042vV

かんたんなレベルで失敗するとペナルティが大きくなるようにする

プログラムのコピー方法

Google ChromeでURLを入力し、このページで「編集」をクリックすればコピーできる

各プログラムのURLは以下の通り

タイミングゲーム(講演で解説): https://makecode.microbit.org/_hvuKFrPoDKL2

タイミングゲーム応用編(前ページ): https://makecode.microbit.org/_1uviKLM042vV

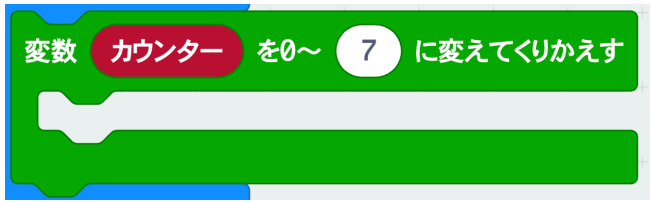
自動開閉パーティション(講演で解説): https://makecode.microbit.org/_9TFAYsiC3Ehi

プログラムをコピーして、プログラムがうまくダウンロードできない場合は、一度ブラウザ(Google Chrome)を閉じて、再びMakeCodeでコピー後のプロジェクトを開き直すとうまくいく可能性がある。(MakeCodeのバグ？)

タイミングゲーム応用編

クリアまでの回数をカウントするための変数「回数」を追加し、初期値0をセット

このループを使ってプログラムをすっきりさせる



クリアしたとき、失敗したときに音を鳴らす

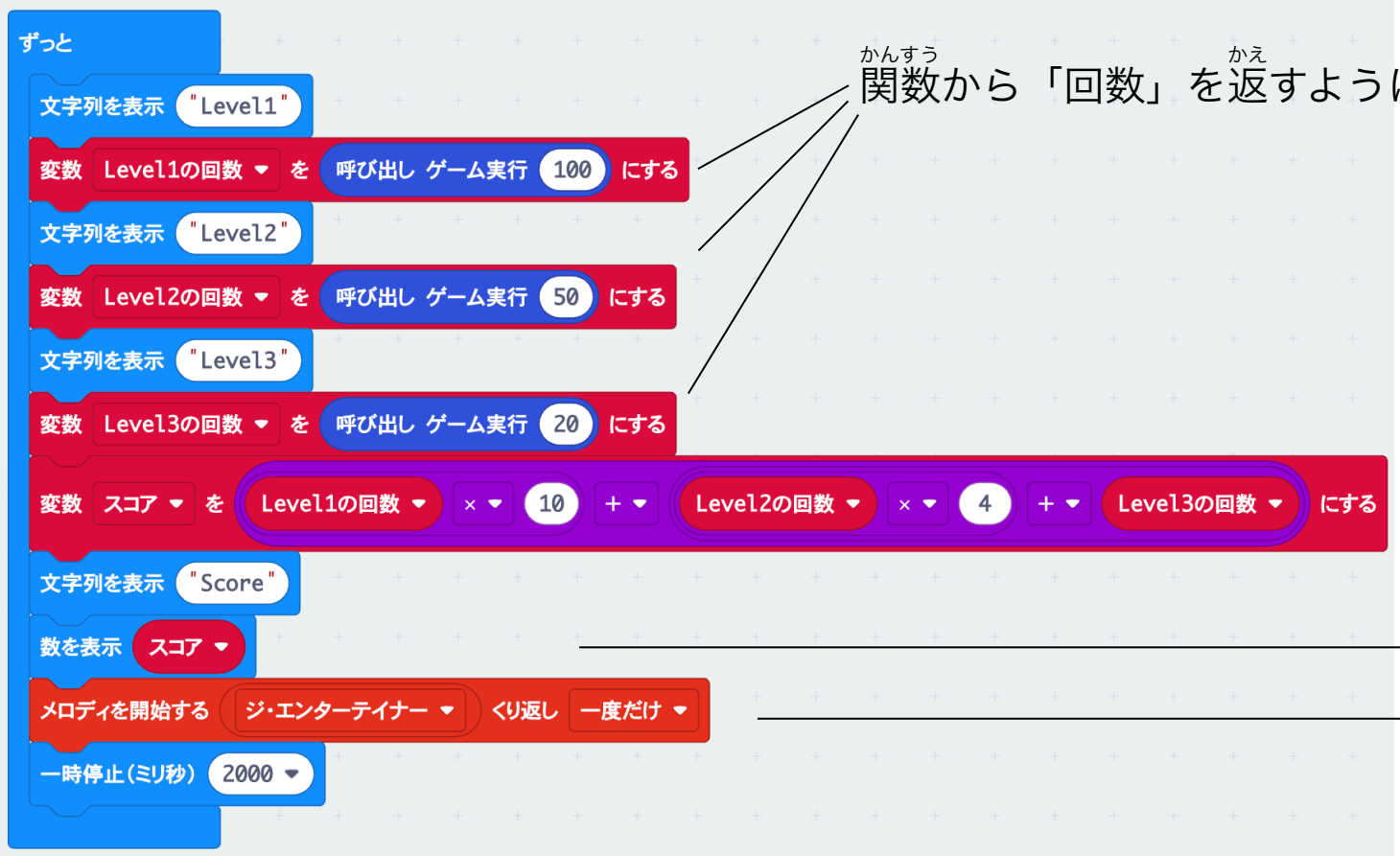
ボタンを押すたびに「回数」を1増やす

ボタンが押されたら「カウンター」を変えるループから抜ける

クリアしたら「回数」を表示する

関数から「回数」を返すようにする

タイミングゲーム応用編



関数から「回数」を返すようにしたので、返したデータを変数に入れる

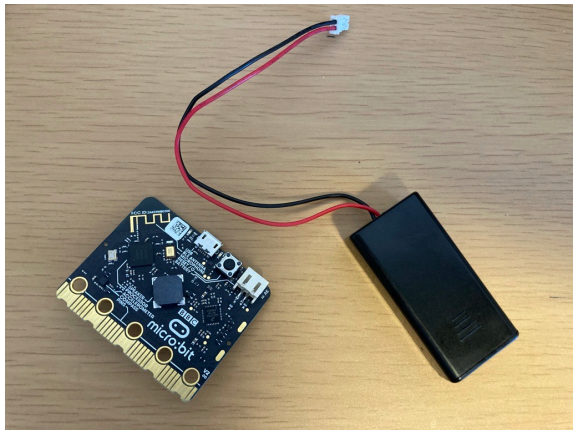
スコアを計算する

スコアを表示する

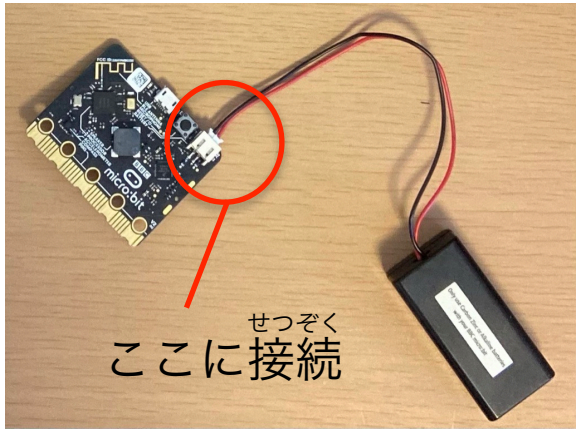
スコアを表示したら音楽を鳴らす

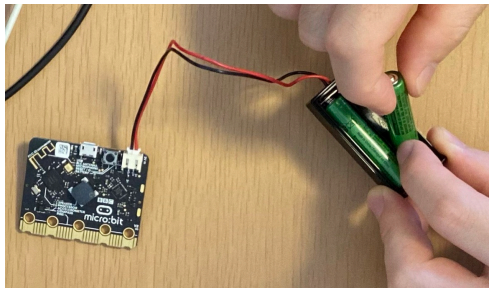
micro:bitに電池ボックスを直接つなぐ場合

- タイミングゲームのようにブレッドボードを使わない場合、USBケーブルを接続していない間は電池ボックスをつなぐ必要がある
- 取り外しは大変なので、スイッチが電池の取り外しで電源のON/OFFを行う



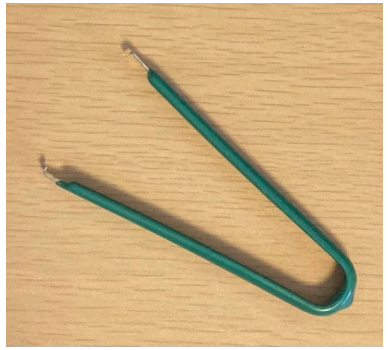
micro:bit go V2 スターターキットの場合は単４電池ボックス

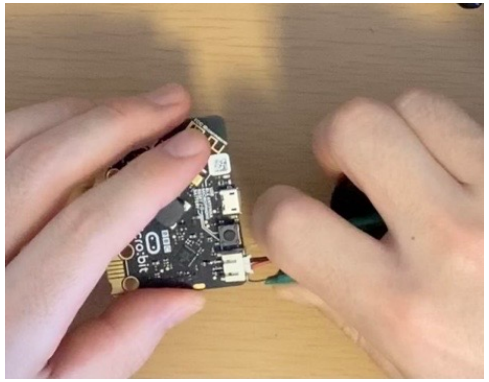




電池を入れて電源ON

※取り外すときは、ケーブルを切らないように、しっかりプラスチック部分を持つ

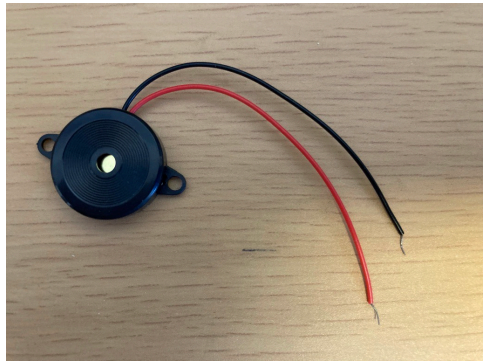




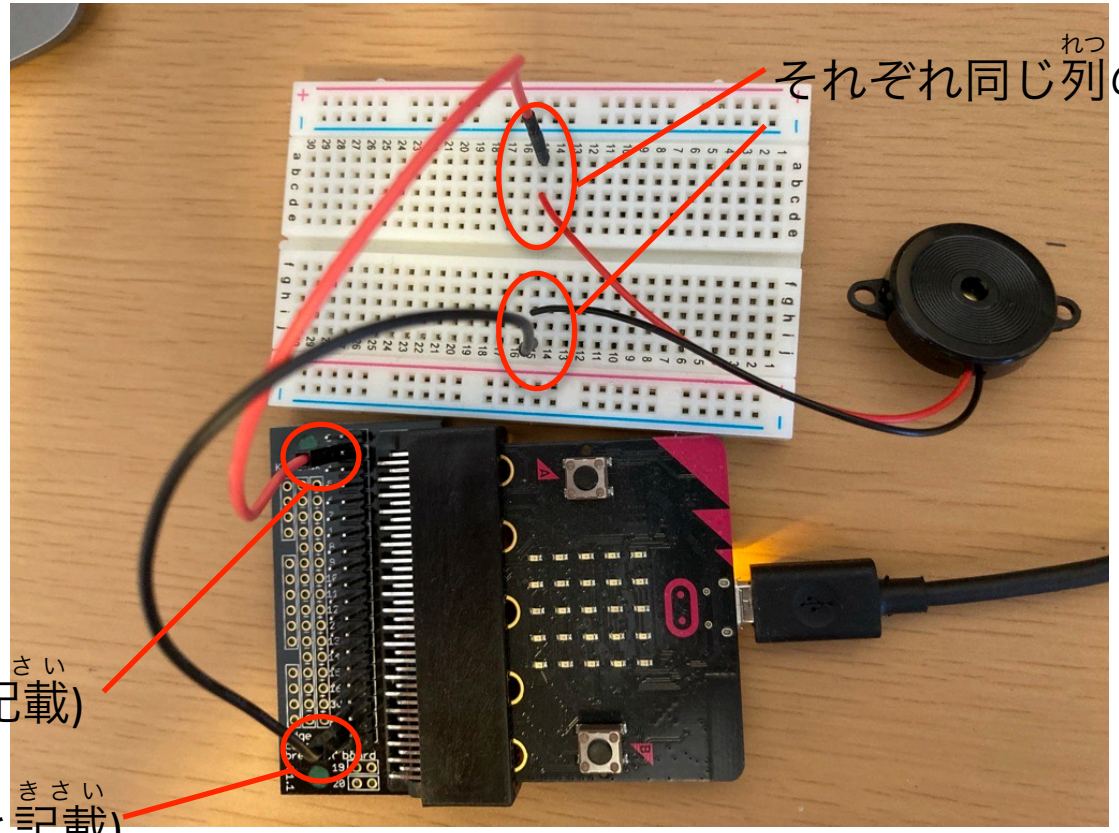
IC引き抜き工具を使うと抜きやすい
(【参考】購入できる場所の例: https://akizukidenshi.com/catalog/g/gT-13303/)

micro:bit V1で音を鳴らす方法

- ブレッドボードを使って、圧電スピーカーとmicro:bitを接続する



圧電スピーカー
(【参考】購入できる場所の例: https://akizukidenshi.com/catalog/g/gP-01251/)



それぞれ同じ列の穴に挿す

上から２番目のピン(“0”と記載)

下から２番目のピン(“0V”と記載)

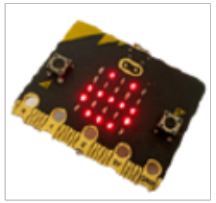
※micro:bit V2はスピーカーが付いているので上の対応は不要

2 自動開閉(かいへい)パーティションの補足(ほそく)情報(じょうほう)

回路図（かいろず）

部品がどのようにつながっているかを表す図

TA48033S



Vin Vout GND

0.1μF

47μF

3V

micro:bit

0 0V 2 1

接続点

- つながっている
- つながっていない

HC-SR04

Vcc

Trig

Echo

GND

ブレッドボード上の配線（はいせん）を見て、回路図（かいろず）と対応（たいおう）づいているか確認（かくにん）してみましょう

+

-

330Ω

LED

赤 橙

Vcc Sig GND

茶

SG92R

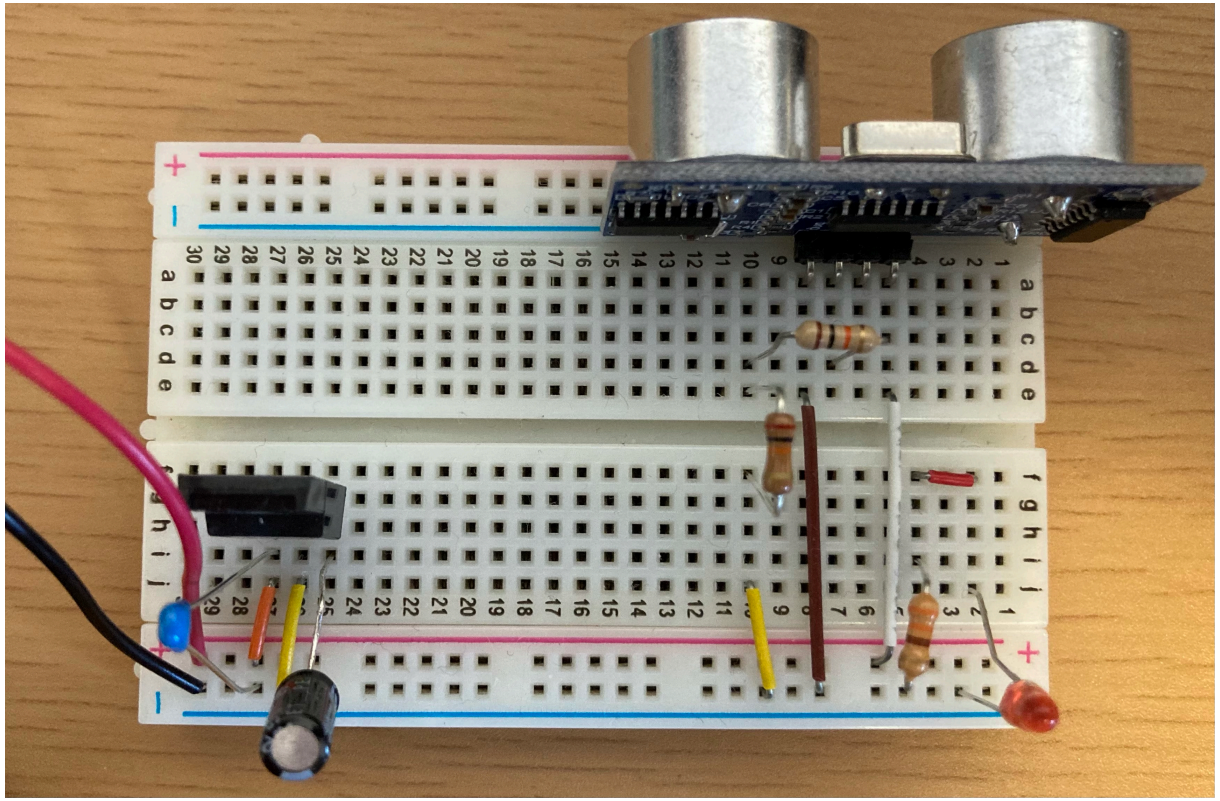
10kΩ

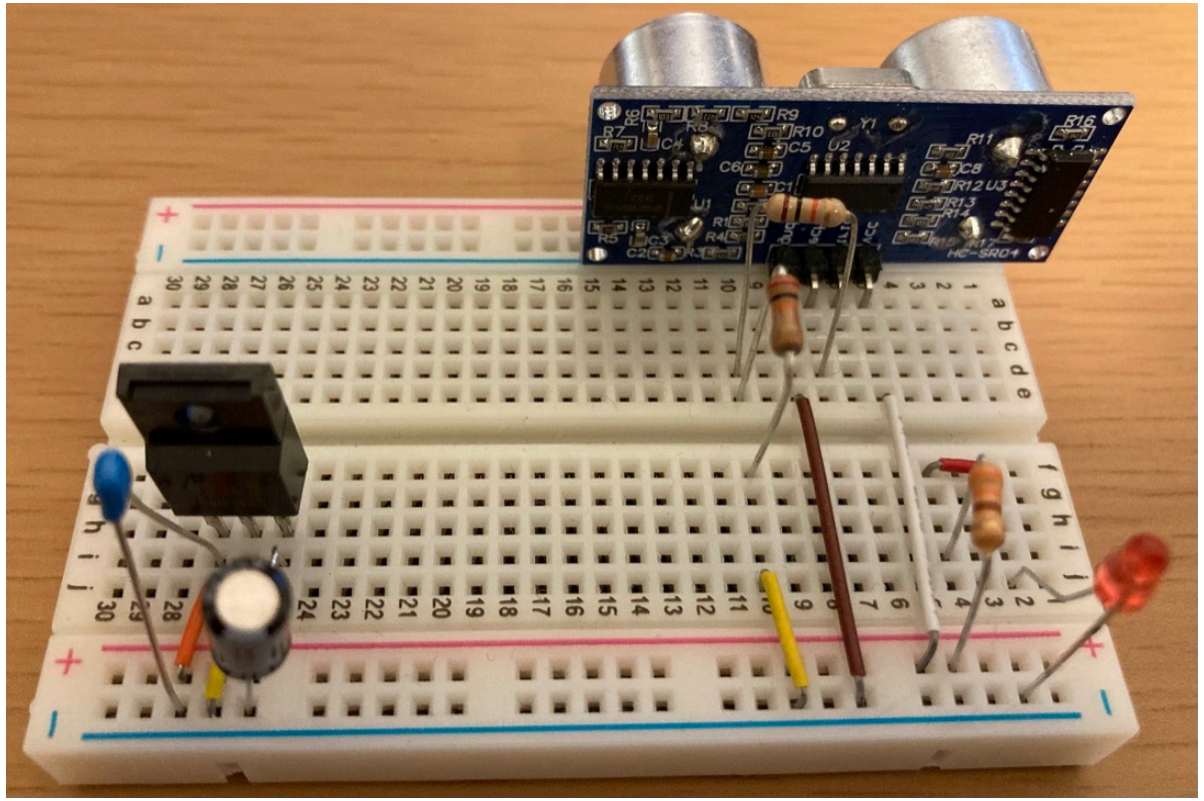
20kΩ

micro:bitとサーボモータ・超音波距離（ちょうおんぱきょり）センサは動作する電圧（でんあつ）が異（こと）なる。「電圧（でんあつ）レベル変換（へんかん）」をするのが正当なやり方だが、ここでは回路（かいろ）をかんたんにするため抵抗器（ていこうき）などで代替（だいたい）。

ブレッドボード上の回路（かいろ）

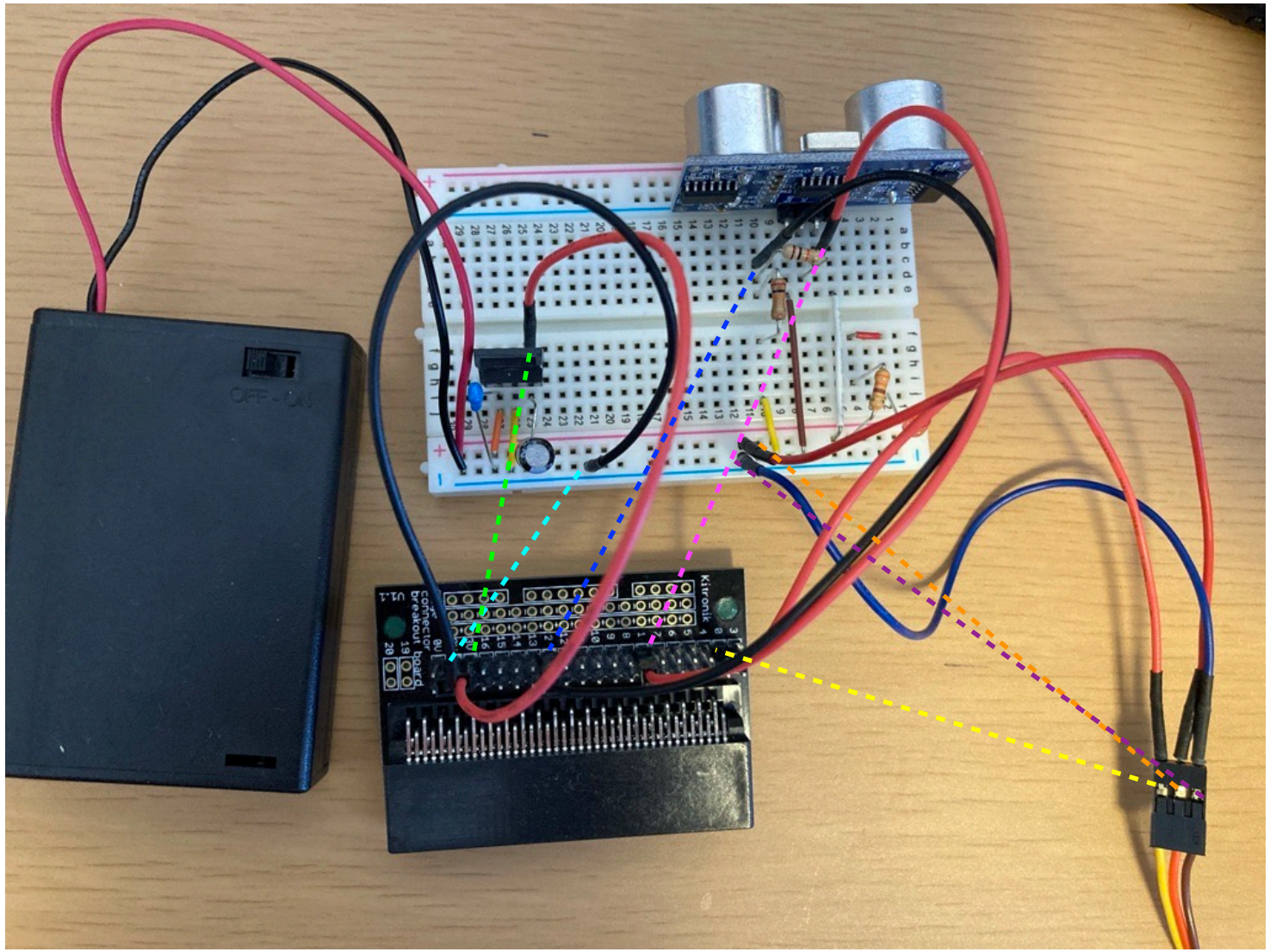
micro:bitやサーボモータと接続（せつぞく）する前の状態（じょうたい）





ブレッドボード上の回路

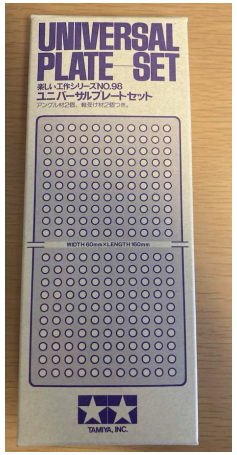
Micro:bitやサーボモータを接続したあと



※点線でジャンパー線の接続を表現

自動開閉(かいへい)パーティションの作り方

ここで使うもの



ユニバーサル
プレートセット

ユニバーサル
アームセット

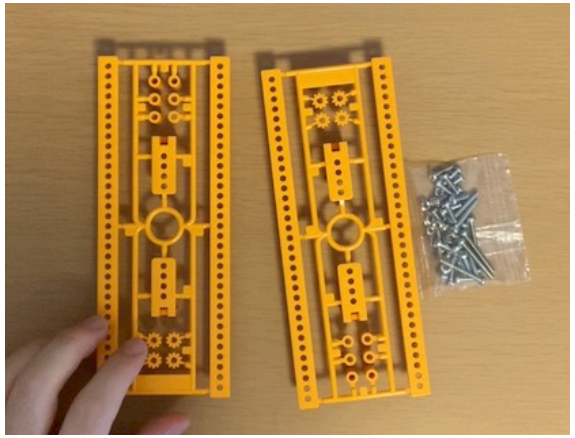
プラバン(B4 厚み0.3mm)

サーボモータ

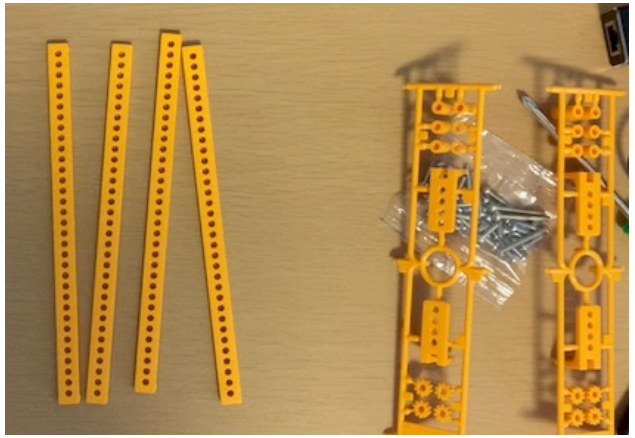
プラスチック対応 接着(せっちゃく) 剤(ざい)

木板とタッピングネジ

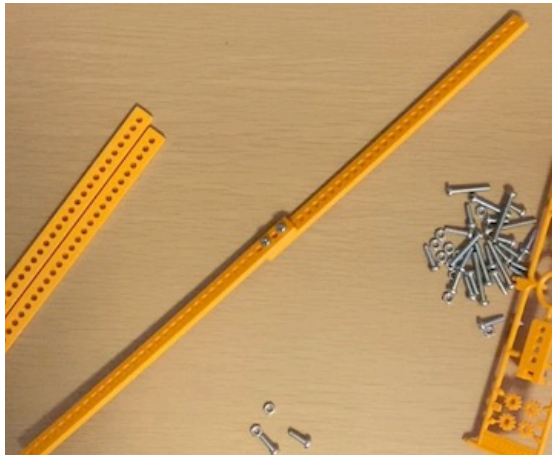
自動開閉パーティションの作り方



(1)ユニバーサルアームセットを開封

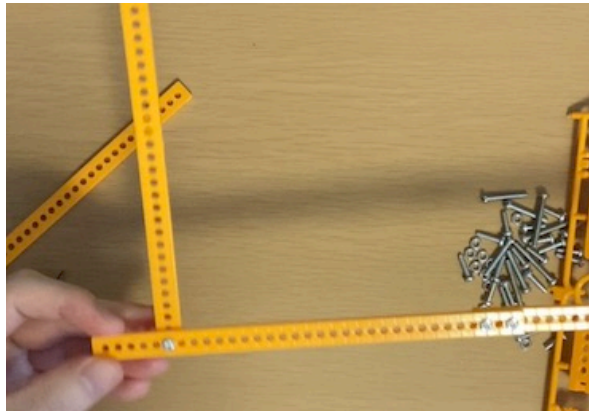


(2) 4つのアーム部品を切り取る



(3) 長さ1cmのネジと、ナットをそれぞれ2つずつ使い、2つのアーム部品を連結する

長さは必要に応じて変えてよい (ここでは穴が3つ重なるように)

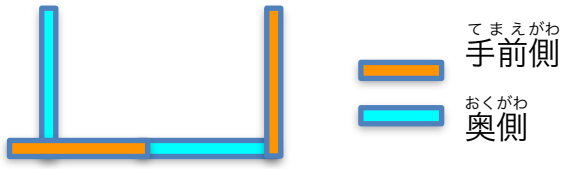


(4) 連結したアームの左から6つ目の穴にアーム部品を取り付ける

(長さ1cmのネジと、ナットを使う)

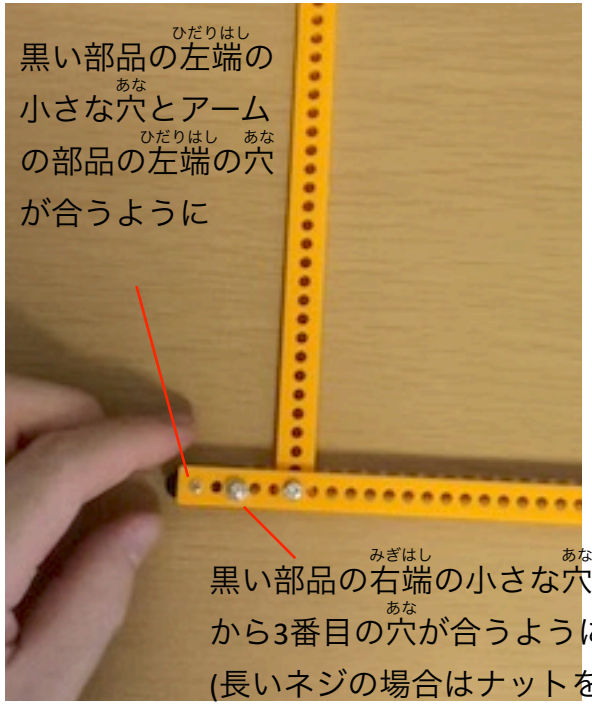
(5) 連結したアームの右端の穴にアーム部品を取り付ける

(長さ1cmのネジと、ナットを使う)

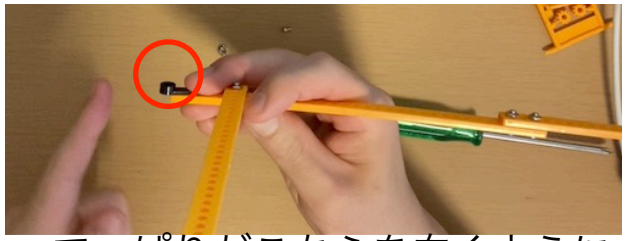


このような位置関係

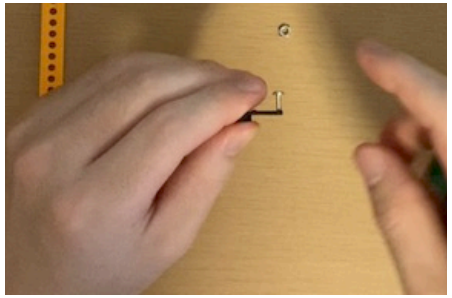
自動開閉パーティションの作り方



黒い部品の右端の小さな穴とアームの部品の左から3番目の穴が合うように
(長いネジの場合はナットを間にはさむとよい)

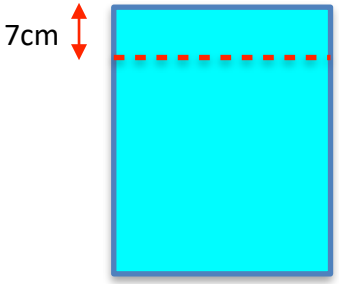


でっぱりがこちらを向くように

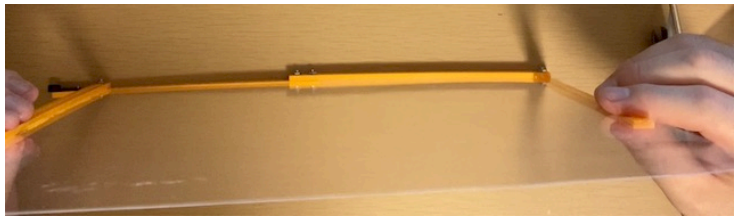


事前にサーボモータ付属のネジで少し刺さるぐらいにネジを回しておいてから、ネジを取り外し、それからアーム部品に取り付けるとやりやすい

(6) 連結したアームの左端にサーボモータ付属の黒い部品を取り付ける
(サーボモータ付属のネジ2つを使う)

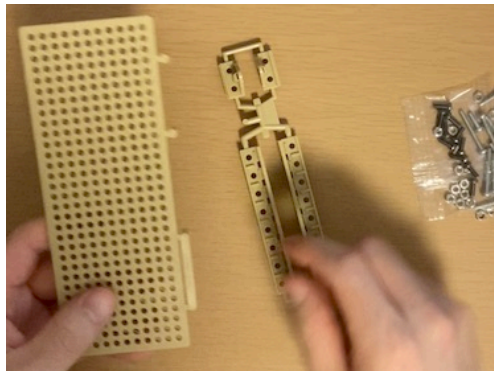


(7) パーティションの大きさに合わせてプラバンを切り取る
(ここでは上から7cmの位置で切り取る)



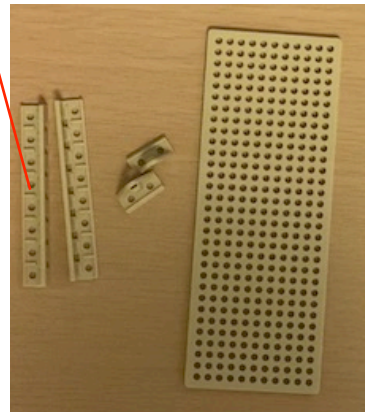
(8) 切り取ったプラバンを接着剤でアーム部品につける

自動開閉パーティションの作り方



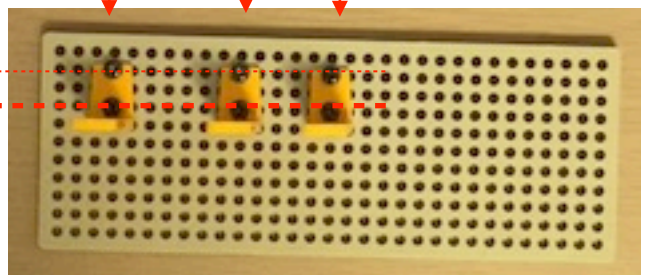
(9) ユニバーサルプレートセットを開封する

一つは使用しない

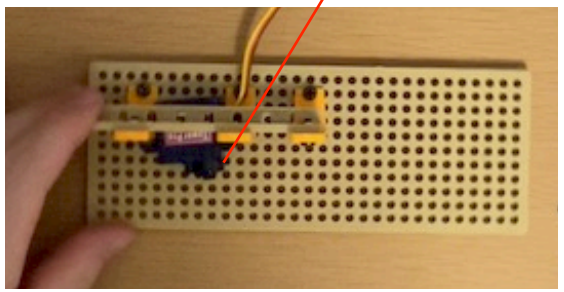


(10) 部品をニッパーで切りはなす

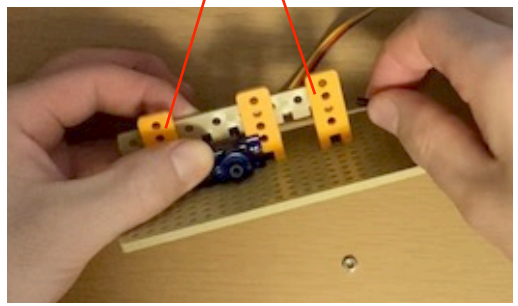
(11) ユニバーサルアームセットのL字パーツを3つ切りはなす



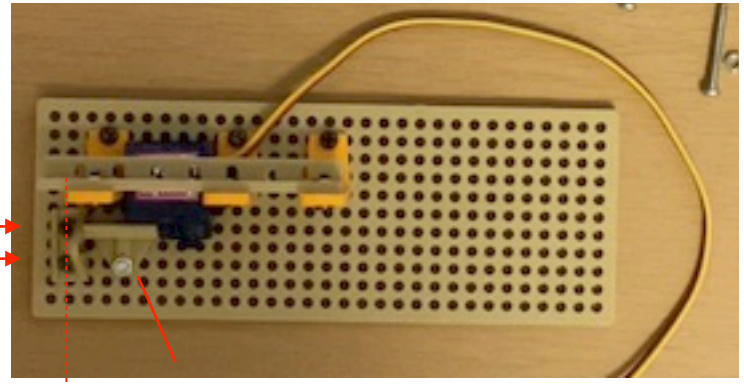
(12) L字パーツをプレートに取り付ける



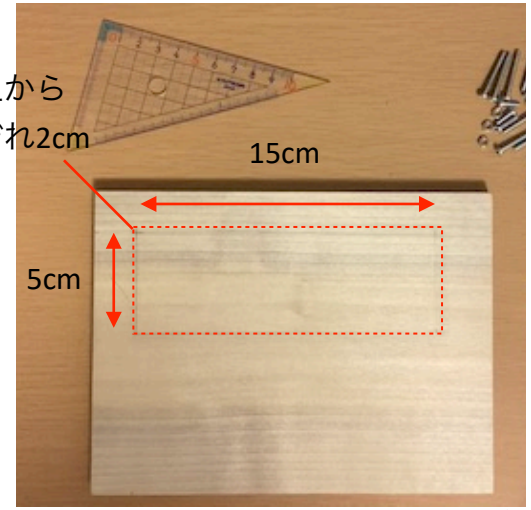
(13) サーボモータを置き、上から細長いL字パーツを取り付ける



自動開閉パーティションの作り方

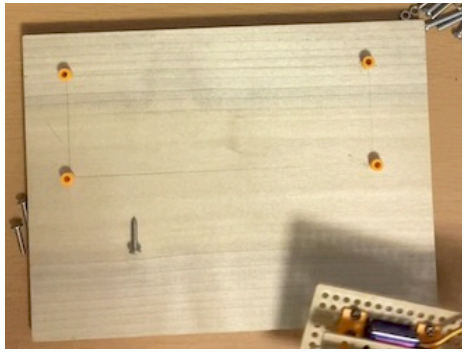


(14) 三角パーツをプレートに取り付ける

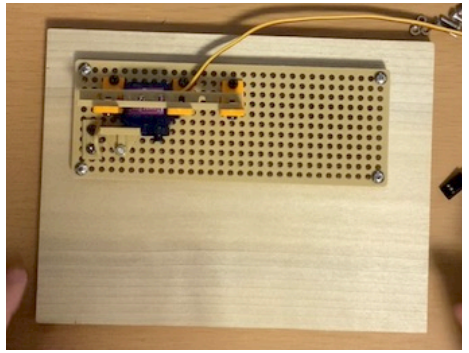


(15) 5cm×15cmの長方形を木板上に描く

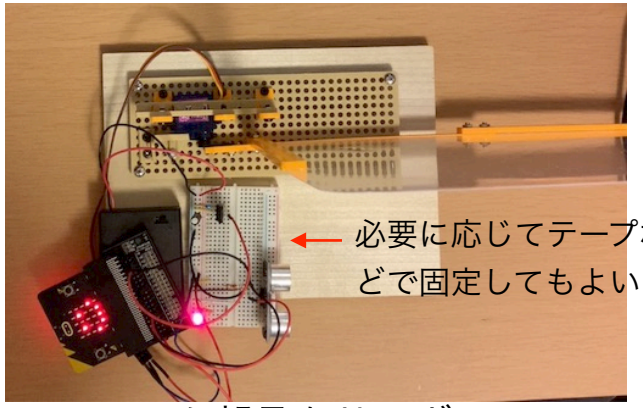
(16) 長方形の四隅をキリで軽く穴あけ



(17) 一番短いスペーサーを4つ穴の上に置く



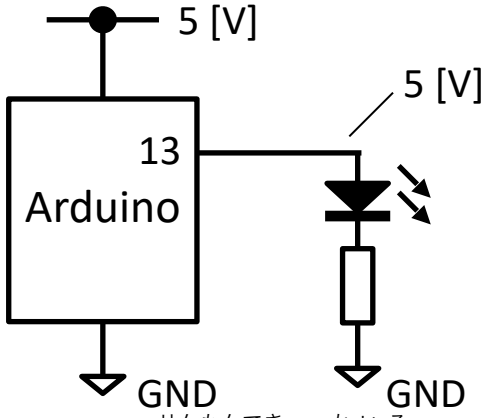
(18) プレートを上に置いて、
四隅を木ネジでとめる



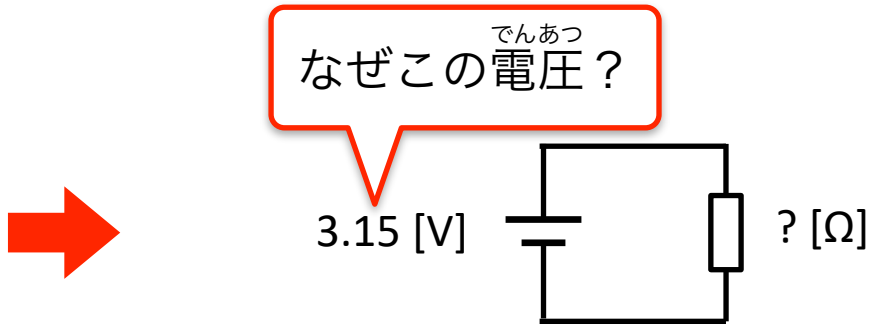
(19) アーム部品をサーボモータにつなぎ、回路をつなぐ

3 Arduinoを使ったLED点滅実験の補足情報

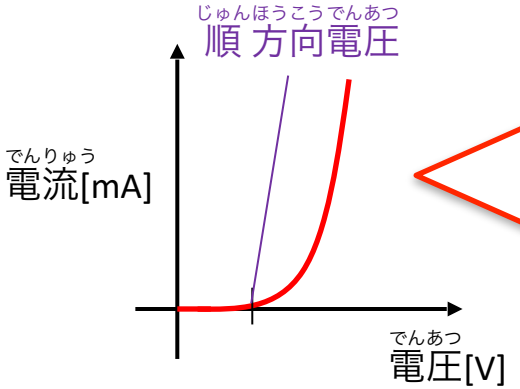
シンプルな回路にしたときの電圧について



ちょっと専門的な回路



LEDの影響を考慮したシンプルな回路 (端子から5V出力時)



LEDの電圧- 電流 特性グラフ

LEDはある程度電圧を加えないと電流が流れない！
この電圧が「順方向電圧」

今回使用するLEDの順方向電圧: 1.85～2.5V

抵抗器に加わる電圧は、順方向電圧を除いて考える

Arduinoの端子の電圧(5 [V]) – LEDの順方向電圧(1.85 [V]) = 3.15 [V]

Arduino開発環境のインストール方法

以下のページからインストーラをダウンロード

https://www.arduino.cc/en/software

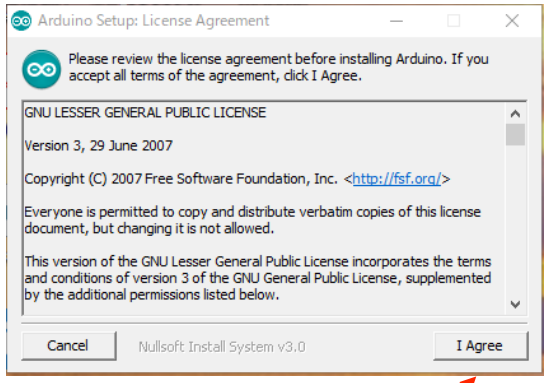
Windowsの場合は
ここをクリック

寄付する必要がなければこちらをクリック

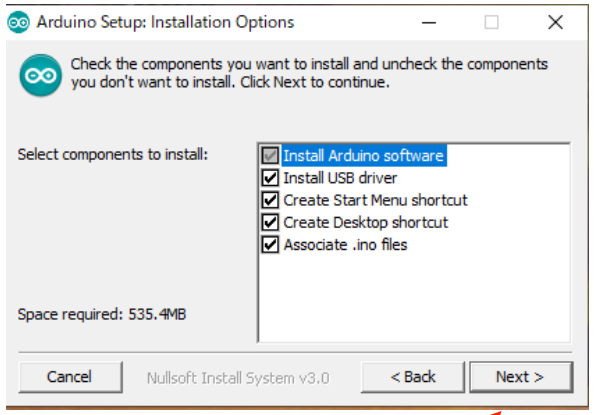
以降はWindows版で説明

Arduino開発環境(かいはつかんきょう)のインストール方法

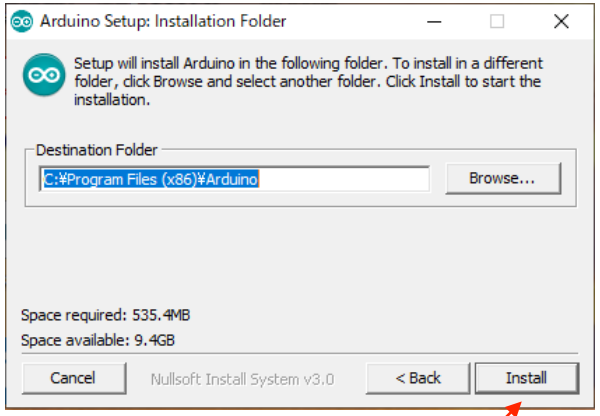
ダウンロードしたインストーラーをダブルクリックで実行し、インストールをすすめる



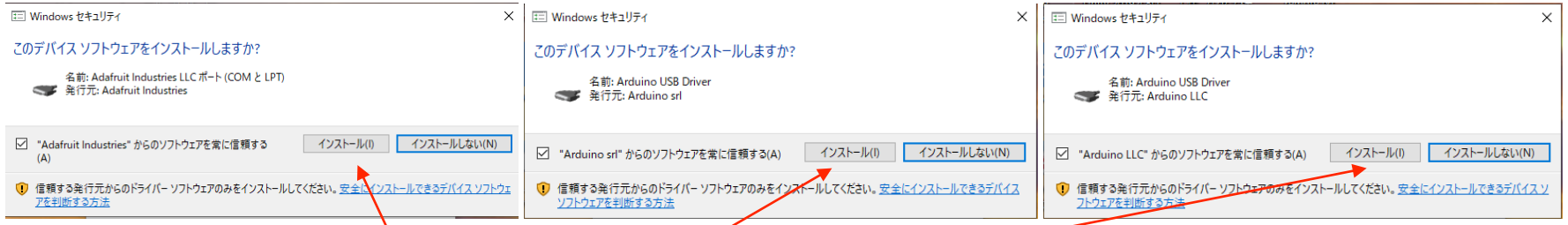
ここをクリック



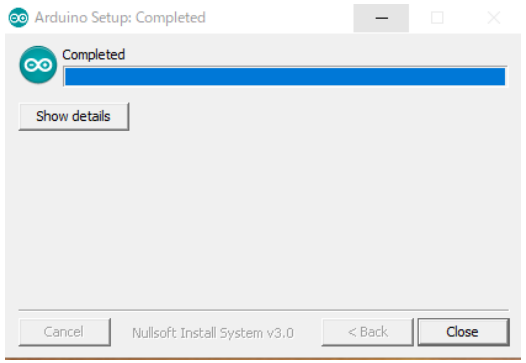
そのままでここをクリック



そのままでここをクリック



この表示が出たら「インストール」をクリック



ここをクリックして完了(かんりょう)

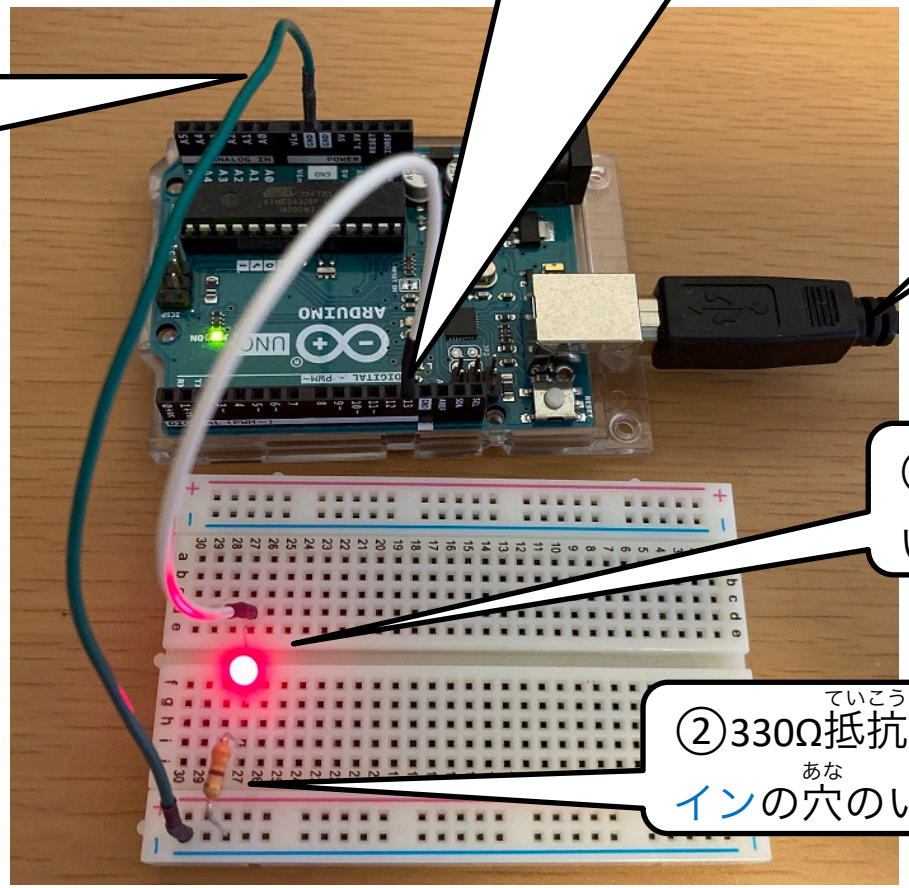
回路作成

接続例

④ジャンパー線をArduinoの13番ピンとブレッドボードのd-27に挿す。

③ジャンパー線をArduinoのGND端子とブレッドボードの青ラインの穴いずれかに挿す。

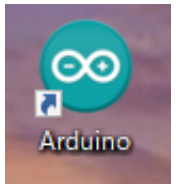
⑤USBケーブルをArduinoとパソコンに接続する。



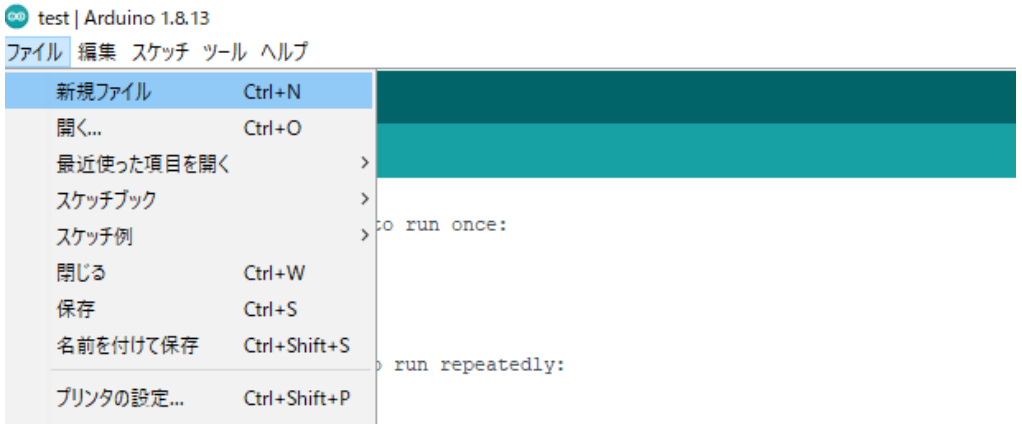
①LEDの足の長い端子をe-27、短い端子をf-27に挿す。

②330Ω抵抗器(橙橙茶金)をi-27, 青ラインの穴のいずれかに挿す。

LED点滅（てんめつ）プログラムの作成



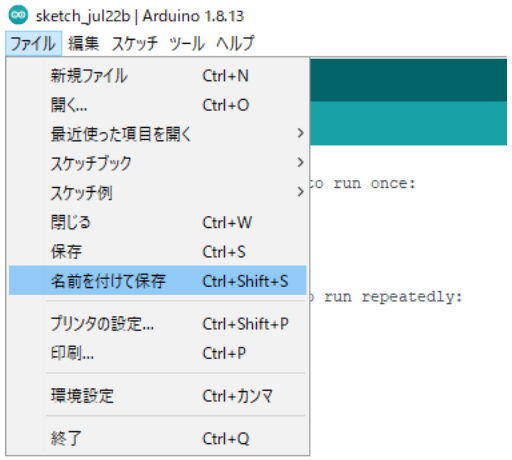
(1) ダブルクリックしてArduino IDEを起動



(2)「ファイル」→「新規ファイル」をクリック

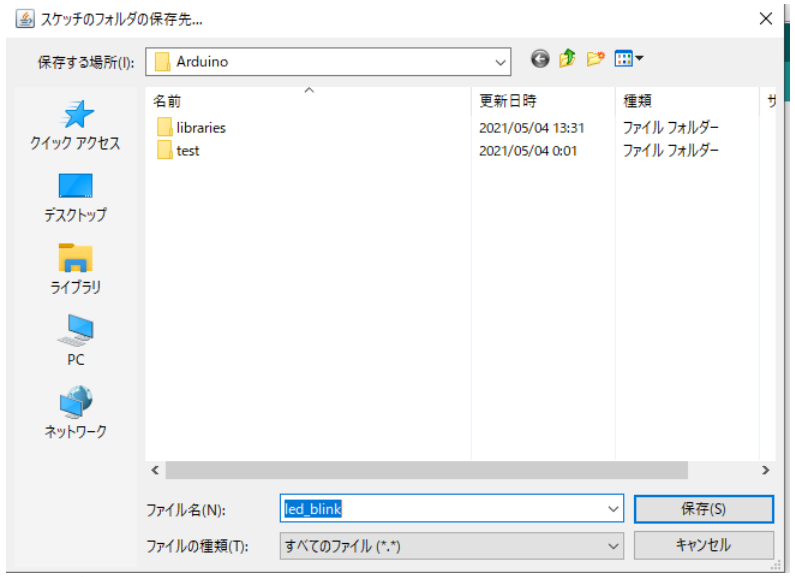
(3) プログラムを入力

LED点滅プログラムの作成



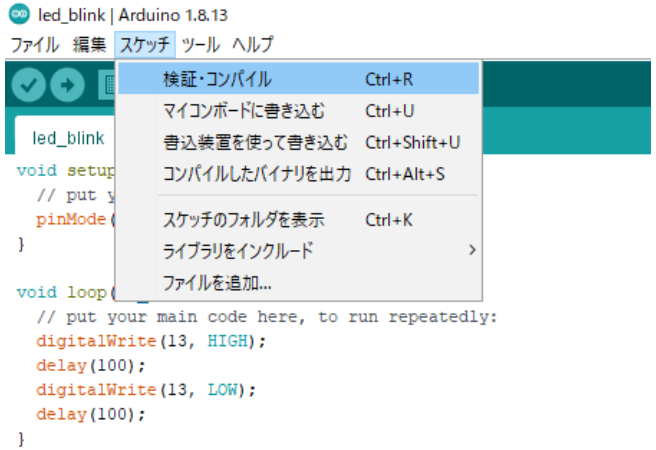
(4)「ファイル」→「名前を付けて保存」をクリック

(以降(いこう)は、同じファイルであれば「保存」をクリックして変更(へんこう)を保存(ほぞん)する)

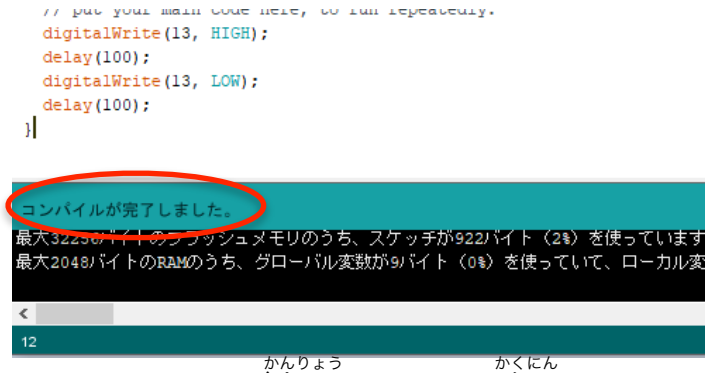


(5) 適当(てきとう)な名前をつけて「保存」をクリック

(ここでは「led_blink」とした)

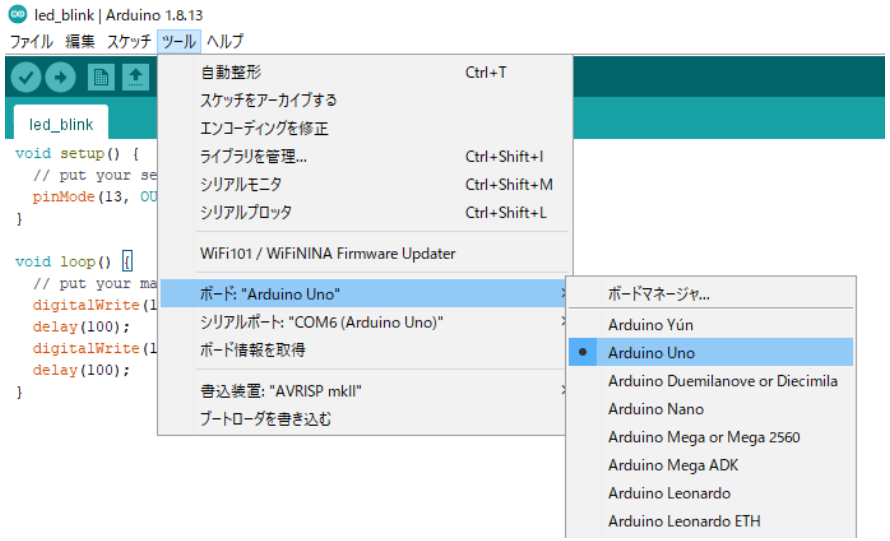


(6)「スケッチ」→「検証・コンパイル」をクリック

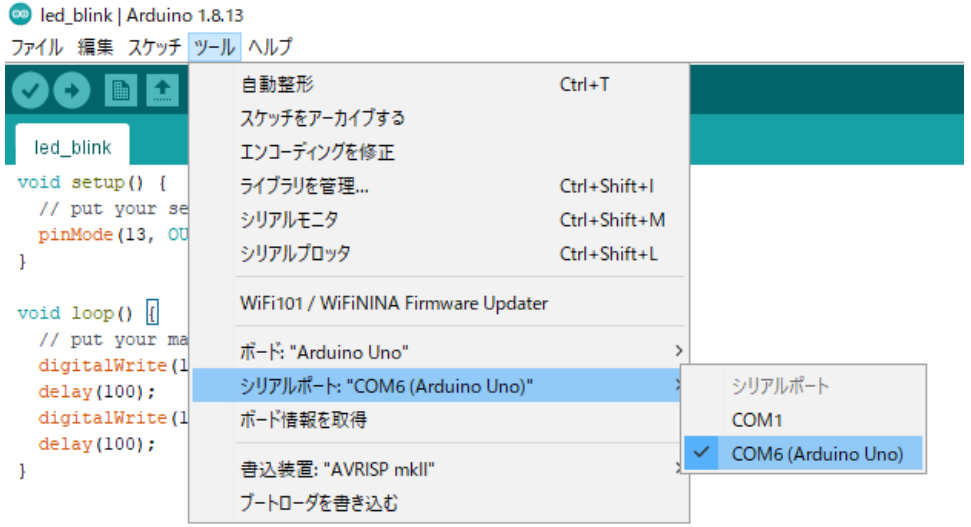


コンパイルが完了(かんりょう)することを確認(かくにん)

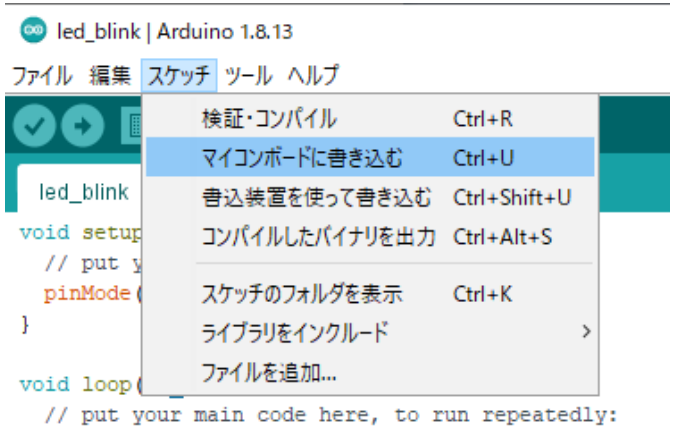
LED点滅プログラムの作成



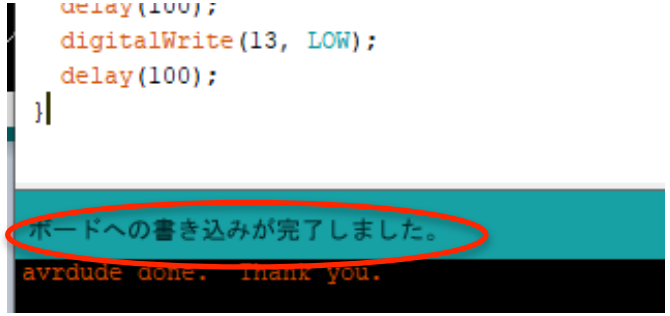
(7)「ツール」→「ボード」→「Arduino Uno」をクリック



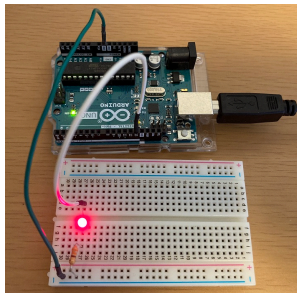
(8)「ツール」→「シリアルポート」→「(Arduino Uno)」がついているCOMをクリック
(Arduinoがパソコンに接続(せつぞく)されていないと出てこない)



(9)「スケッチ」→「マイコンボードに書き込む」をクリック



ボードへの書き込みが完了(かんりょう)することを確認(かくにん)し、LEDが点滅(てんめつ)することを確認(かくにん)する



プログラムの解説

```
void setup() {
  pinMode(13, OUTPUT);  // 13番ピンを出力に設定
}

void loop() {
  digitalWrite(13, HIGH);   // 13番ピンをHIGH(5V)に
  delay(100);               // 0.1秒一時停止
  digitalWrite(13, LOW);    // 13番ピンをLOW(0V)に
  delay(100);               // 0.1秒一時停止
}
```

← マイコンの端子は、「出力」や「入力」を設定できる
(入力の場合はスイッチが押されているかなどを判断できる)

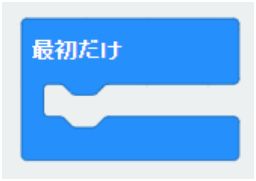
← 電流が流れるのでLEDが光る

← 電流が流れないのでLEDが消える

setupやloopは関数

setup関数の中の処理は、電源がONになったとき一回だけ実行される

MakeCodeのこのブロックと同じ



loop関数の中の処理は、ずっとくり返される

MakeCodeのこのブロックと同じ