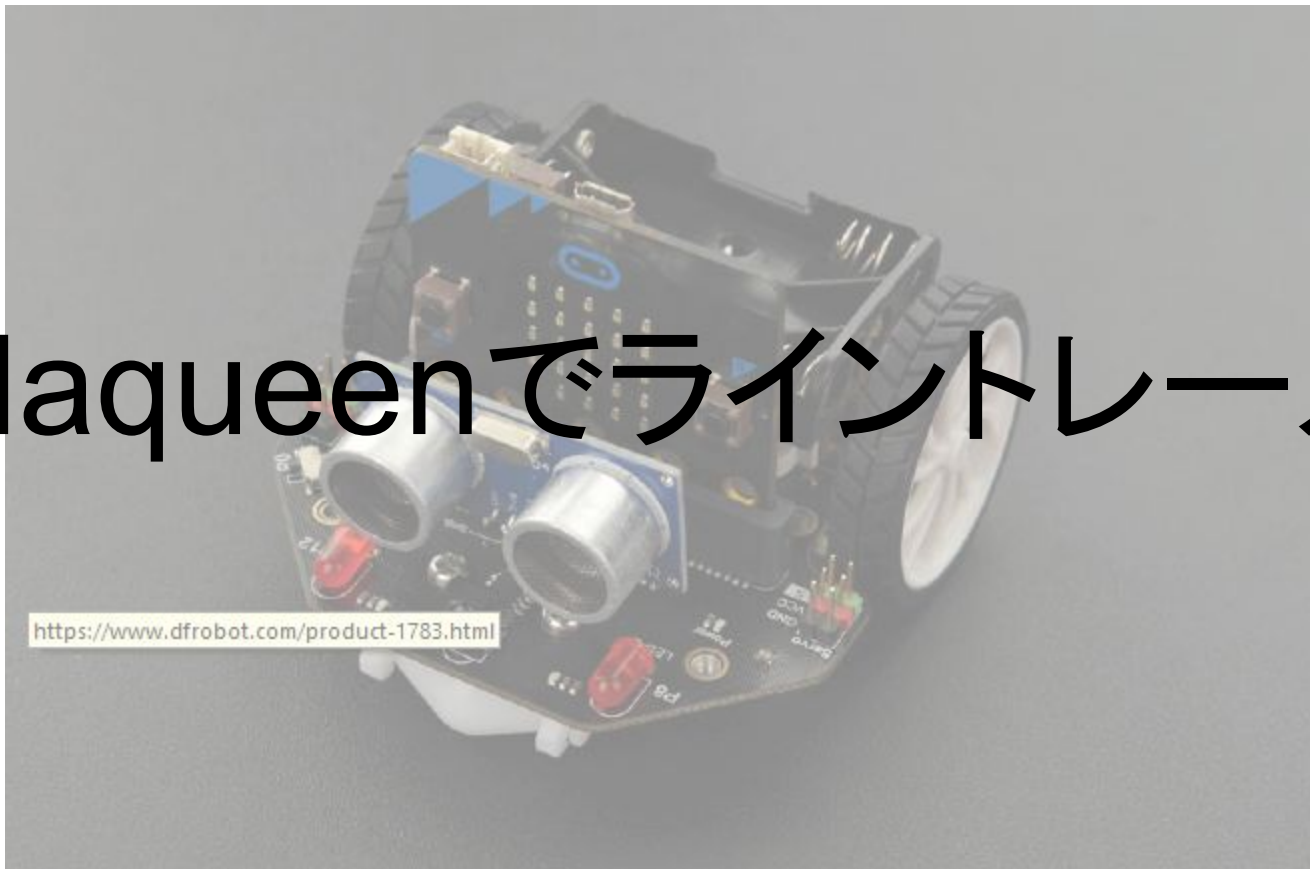


Maqueenでライントレース

<https://www.dfrobot.com/product-1783.html>



ライトレースロボットについて

ライトレースロボットは、床に描いたラインに沿って走るロボットです。

ロボットの下面にはラインを識別するための光センサーがついています。

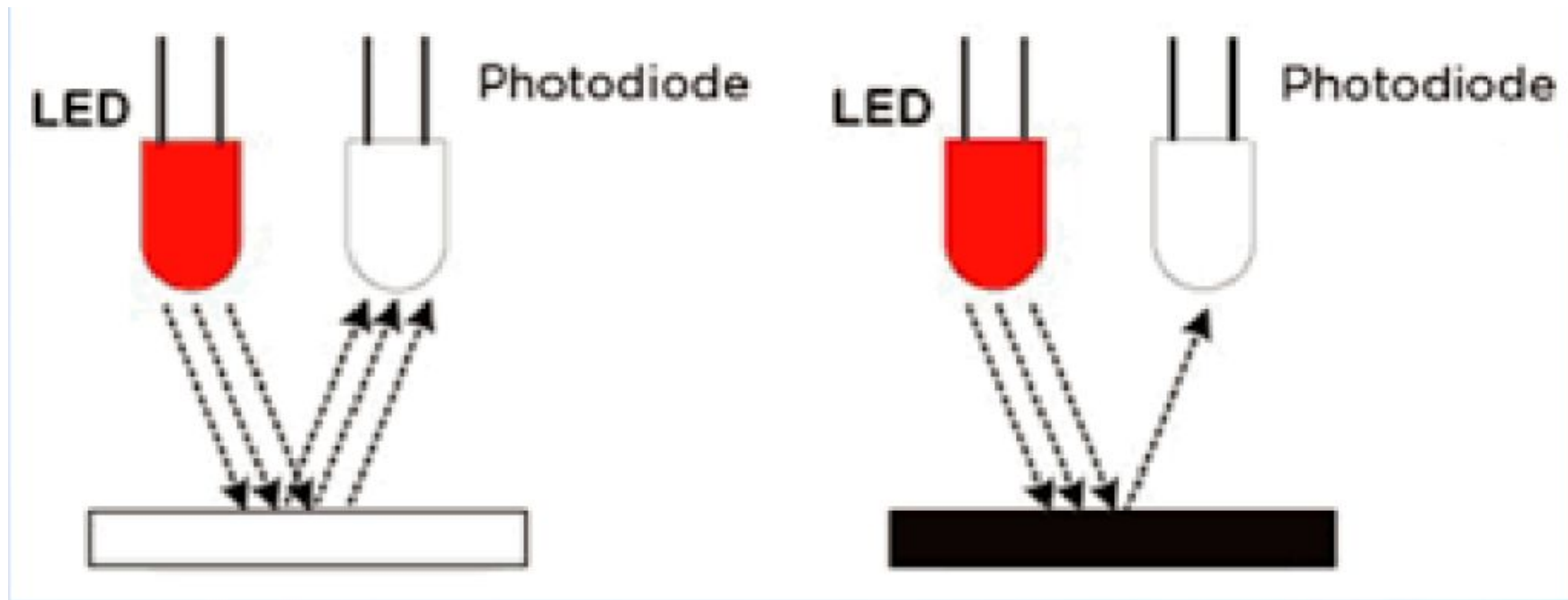
人の生活する環境には赤から紫までの可視光にあふれているため、これを感じる光センサーをライトレースロボットに使うのは、ラインを見つけるのに失敗する恐れがあります。

そこでふつうは赤外線センサー(光センサーの一種)をつかいます。

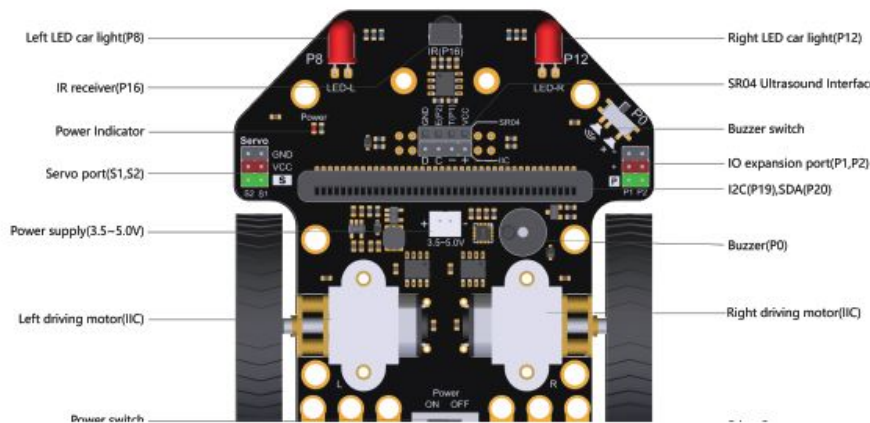
黒色は赤外線をほとんど反射しません。逆に白色は赤外線を多く反射します。

赤外線センサーはこの光の反射する量を電気信号に変えます。ロボットはこの信号をもとにラインを見つけてます。

赤外線センサー

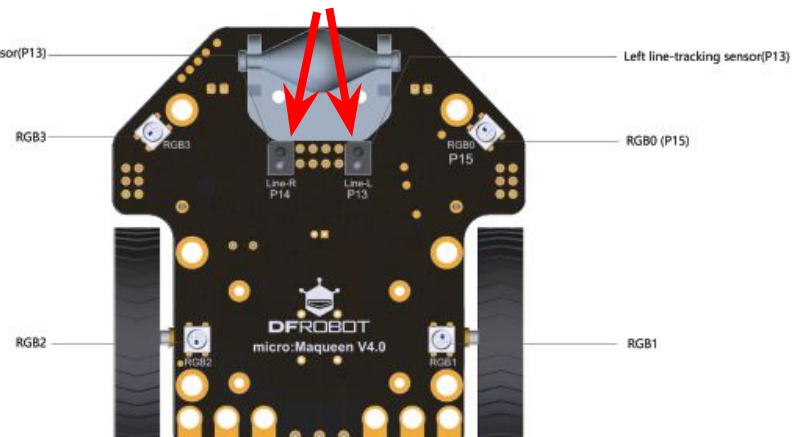


Maqueen



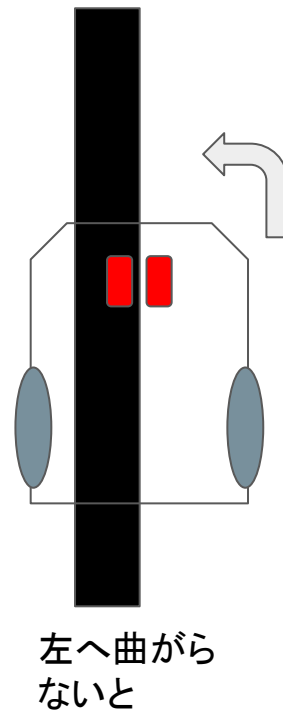
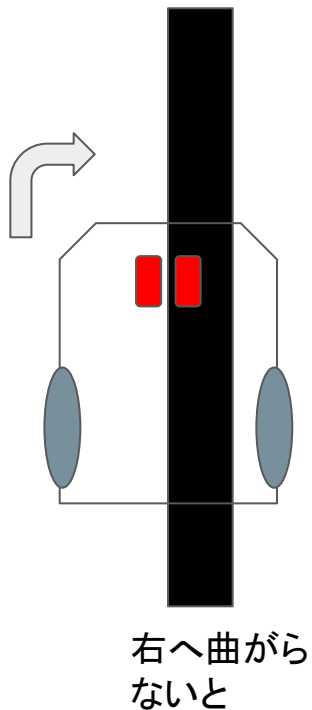
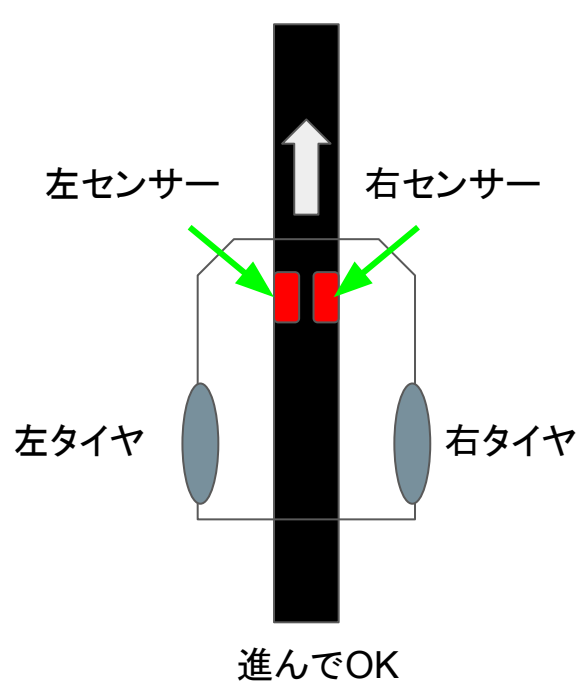
上

赤外線センサー



下

ライトレースに必要な動き



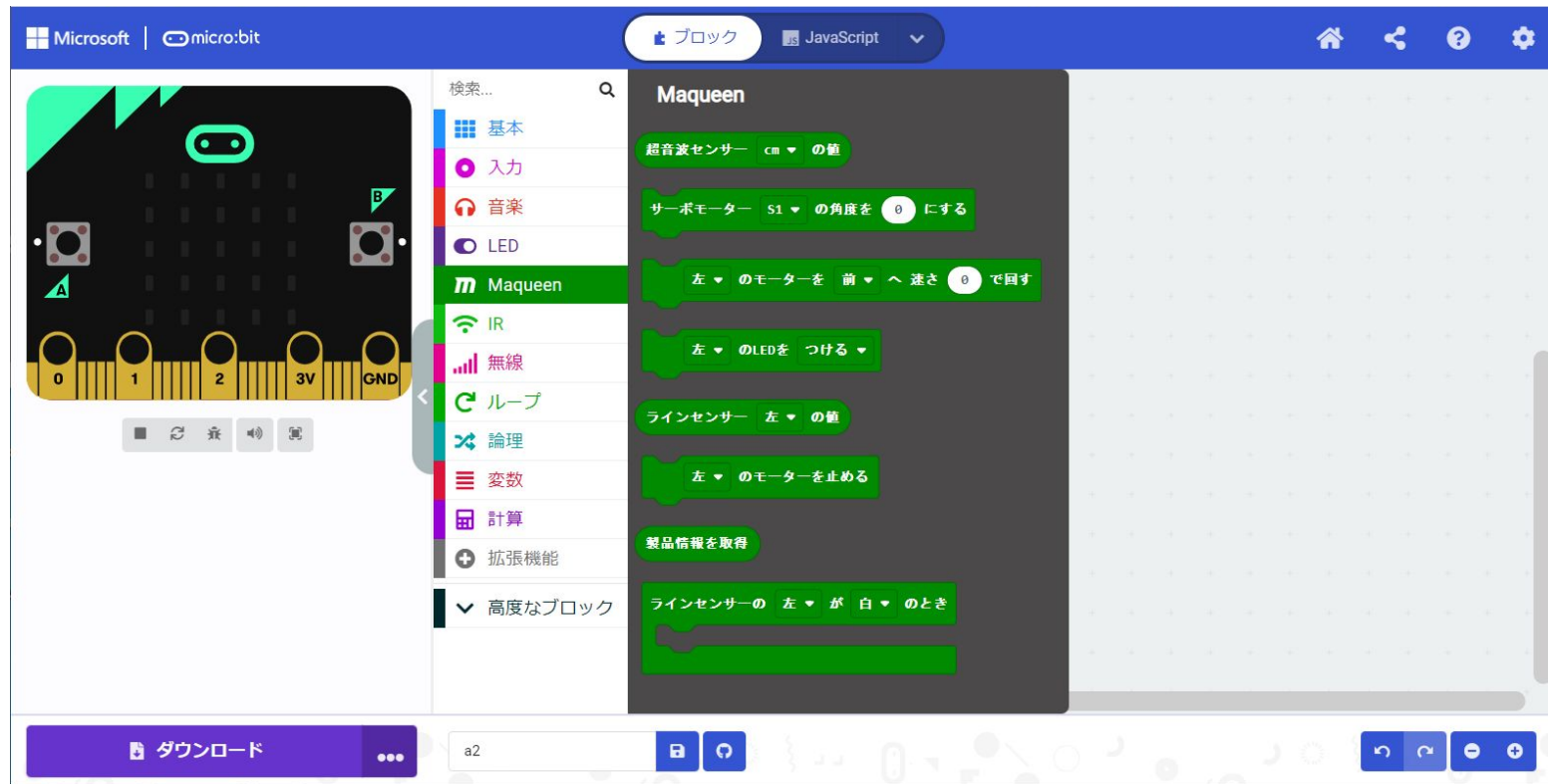
うごきを表にまとめる

うごき	左センサー	右センサー	左タイヤ	右タイヤ
前へすすむ	黒	黒	すすむ	すすむ
右にまがる	白	黒	すすむ	とめる
左にまがる	黒	白	とめる	すすむ
止まる	白	白	*	*

Maqueenをプログラムするには

The image shows the Microsoft MakeCode micro:bit IDE interface. The main workspace displays a micro:bit board with a simple program consisting of two blocks: '最初だけ' (Only at first) and 'ずっと' (Forever). The left sidebar contains a search bar and a list of categories: 基本 (Basic), 入力 (Input), 音楽 (Music), LED, 無線 (Wireless), 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 '拡張機能' section is expanded, showing a grid of available extensions. The 'Maqueen' extension, described as 'Affordable mini robot designed by DFRobot', is highlighted with a red border. Other visible extensions include 'tinkercademy-tinker-kit', 'smarthome-kit', 'RingbitCar', 'robotbit', 'cutebot', and 'MaqueenPlus'. The bottom of the interface features a 'ダウンロード' (Download) button and a search bar containing the text 'a2'.

Maqueen用のブロックが使えるようになる



表をもとにプログラムを考える

1. 左と右の両方のセンサーが黒なら左と右のタイヤを前へ
2. センサーの左が白で、右が黒なら左タイヤを前へ、右タイヤを止める
3. センサーの左が黒で、右が白なら左タイヤを止め、右タイヤを前へ
4. その他のばあい、左と右のタイヤをどうするかは適当に決める

センサーが白を見つけたばあい、プログラムではセンサーの値が1となります
逆に黒を見つけたばあい、センサーの値は0となります

2と3の曲がるばあい、片方のタイヤは止めるとしましたが、そうするとうごきがカクカクになるので、少し前にまわしておいた方がよい

モーターの速さは0~255で指定できる

ライトレースプログラム

Scratch script for a light race program:

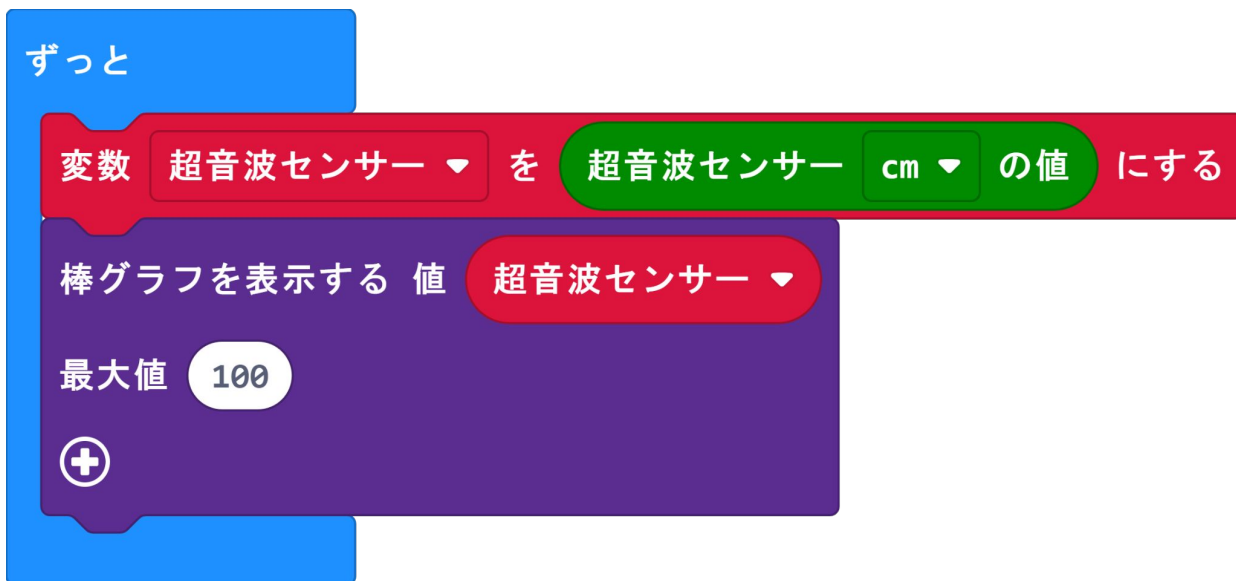
- ずっと (Forever Loop):**
 - もし (If):** `ラインセンサー 左 の値 = 0` **かつ** `ラインセンサー 右 の値 = 0` **なら**
 - `すべて のモーターを 前 へ 速さ 0 で回す`
 - でなければもし (If-Else):** `ラインセンサー 左 の値 = 0` **かつ** `ラインセンサー 右 の値 = 0` **なら** `-`
 - `左 のモーターを 前 へ 速さ 0 で回す`
 - `右 のモーターを 前 へ 速さ 0 で回す`
 - でなければもし (If-Else):** `ラインセンサー 左 の値 = 0` **かつ** `ラインセンサー 右 の値 = 0` **なら** `-`
 - `左 のモーターを 前 へ 速さ 0 で回す`
 - `右 のモーターを 前 へ 速さ 0 で回す`
 - でなければ (Else):** `-`
 - `すべて のモーターを止める`

よける

よけるにはモノまでの遠さ(距離)をしらないと

このプログラムをうごかすと、micro:bitのLED画面にモノまでの距離が表示される。2から400cmまでだが、1から3cmはエラー(失敗)となる。

ただしくはかれるのは20から80cm。



モノまでの距離が10cm以上なら前進、そうでなければ右にまがるプログラムを考えよう

The image shows a Scratch-like block diagram for a robot navigation program. The blocks are as follows:

- ずっと (Forever) loop:**
 - もし (If) block:** $\text{超音波センサー cm の値} < 10$ (Ultrasonic sensor cm value < 10).
 - かつ (And) block:** $0 \neq \text{超音波センサー cm の値}$ ($0 \neq \text{Ultrasonic sensor cm value}$).
 - なら (Then) block:** **すべて (All) of the motors forward at speed 200** (すべて のモーターを 前 へ 速さ 200 で回す).
 - でなければ (Otherwise) block:**
 - 左 (Left) motor forward at speed [input]** (左 のモーターを 前 へ 速さ [input] で回す).
 - 右 (Right) motor forward at speed [input]** (右 のモーターを 前 へ 速さ [input] で回す).
 - 一時停止 (ミリ秒) (Wait in milliseconds) block:** 500 (一時停止 (ミリ秒) 500).

A red arrow points to the "一時停止 (ミリ秒) 500" block, with the text "訂正" (Correction) written next to it.

障害物まで5cm以上ならライトレース、そうでない(障害物がある)ならUターンするプログラムを考えてみよう次のページにも続く

最初だけ

変数 LineP を 50 にする

変数 TurnP を 40 にする

変数 TurnT を 100 にする

ずっと

もし なら

もし $\langle \text{ラインセンサー 左 の値} = 0 \rangle$ かつ $\langle \text{ラインセンサー 右 の値} = 0 \rangle$ なら

すべて のモーターを 前 へ 速さ LineP で回す

でなければもし $\langle \text{ラインセンサー 左 の値} = 1 \rangle$ かつ $\langle \text{ラインセンサー 右 の値} = 0 \rangle$ なら ⊖

左 のモーターを 前 へ 速さ $\text{LineP} \times 0.8$ で回す

右 のモーターを 前 へ 速さ $\text{LineP} \times 0.2$ で回す

でなければもし $\langle \text{ラインセンサー 左 の値} = 0 \rangle$ かつ $\langle \text{ラインセンサー 右 の値} = 1 \rangle$ なら ⊖

左 のモーターを 前 へ 速さ $\text{LineP} \times 0.2$ で回す

右 のモーターを 前 へ 速さ $\text{LineP} \times 0.8$ で回す

でなければ ⊖

すべて のモーターを止める

⊕

でなければ ⊖

