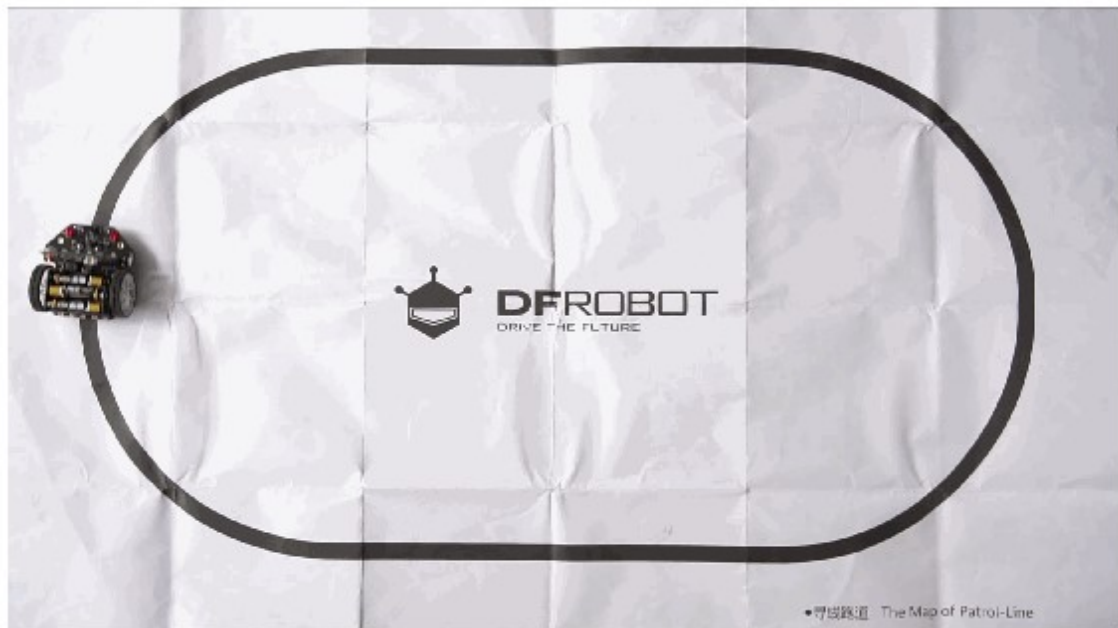


## ライントレーサーを作ろう



### 1. ライントレーサーってなに？

ライントレーサーは、ラインは線、トレーサーはトレースするもの、つまり辿る（たどる）ものまたはなぞるものという意味で、床に引かれた線を自動的に辿って走る車やロボットのことです。ラインレースはロボットの性能を競う競技でよく使われる技術です。ライン・トラッキングとも言います。

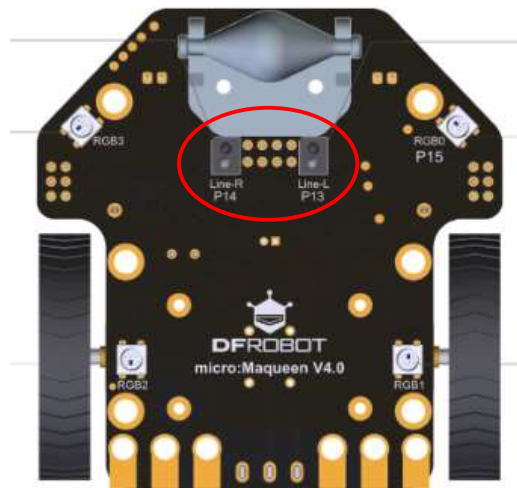


ライントレーサー

### 2. ライントレーサーの仕組み

床に引かれた線の位置を知るためにはセンサーが必要です。人であれば目がそれにあたります。あるいは点字ブロックのように表面に凹凸があれば杖の先や足元の感覚で知ることもしよう。

ロボットの競技で使われるラインレースでは黒や白の太い線が使われることが多いです。これはフォトレジスターと呼ばれるセンサーが安くて手に入りやすく、ロボットによく使われるからです。



### ロボットの裏側

今回使うロボットの裏側にもフォトレジスターが二つ付いています。フォトレジスターにも種類がありますが、ここで使われているのは赤外線フォトレジスター（または赤外線フォトダイオード）と呼ばれるもので、赤外線と呼ばれる目には見えない光（テレビのリモコンで使と同じ）を床に向かってあてて、その反射をセンサーで測るしくみです。反射の強さが弱ければ黒い線の上に、強ければ線の外に居ると判定できます。

ラインレースではこの他にカラーセンサーと呼ばれる目に見える光の色や量を計るセンサーやカメラを使う場合もあります。

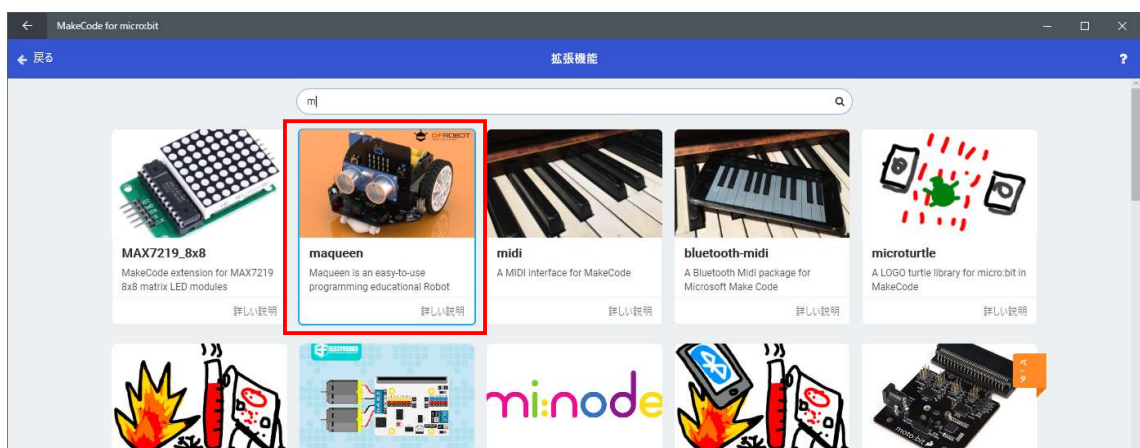
### 3. ライトレーサーを作ってみよう



ライトレースのためのセンサーのブロックは、MakeCode の拡張機能を使います。

「高度なブロック」の一番下にある「拡張機能」を選びます。

拡張機能の画面の検索欄に「m」と入力して検索すると「maqueen」という名前の拡張機能（図の赤枠のもの）が表示されるのでそれを選択します。





すると、元の画面に「MaqueenIR」と「Maqueen」とが表示されるようになります。（今回使用するロボットの名前は Micro Maqueen（マイクロ・マックイーン）といいます。）

Maqueen のモーターやセンサーなどで使う命令用のブロックはこの中にあります。

緑色の Maqueen の中にある「ラインセンサー（…側）の値」というのがライントレースで使用するセンサーの値（あたい）になります。ラインセンサーは右側と左側の2つあります。

センサーが白い（明るい）床の上にあるとこのブロックの値が1になり、青いLEDが点きます。反対に黒い（暗い）線の上にあると値が0になり、青いLEDは消えます。

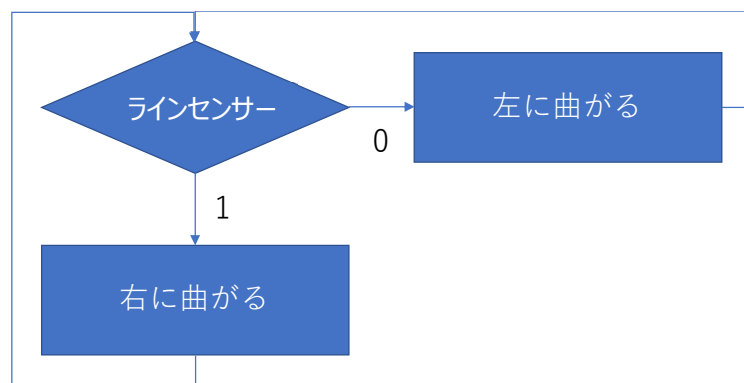


それでは最初にセンサーを一つ使ったライントレーサーをプログラミングで作ってみましょう。

その前に、今回はフローチャートと呼ばれる図を描いてみます。フローチャートは、プログラムを作る前にプログラムの動作の仕組みや順番を考えるためによく書かれる図です。

もし～という条件文はひし形の図形で表します。動作や命令は

長方形で表し、その中に命令の説明を分かりやすく書きます。ブロックとブロックは矢印の線で結び、上から下へ向かってつなげていきます。ここでは条件文でセンサーの値が0か1かを読み取るので、もし0なら Maqueen は左に曲がり、1なら右に曲がるようにします。最後に「ずっと」のループ処理を表すために矢印を最初の箇所に戻しています。



フローチャート

Maqueen は左右に一つずつモーターがあり、それぞれに車輪がついています。まっすぐ進んだり左右に曲がったりするにはモーターのブロックを使います。



「右側」、「左側」を選んで左右どちらのモーターを動かすか指定します。

「前に」、「後に」で前進か後進かを指定します。

速さは数字で表し、0 から 255 まで指定できます。0 は停止です。

左右に曲がるには左右のモーターの速さを変えて調整します。

さきほどのフローチャートをプログラムで表すと、例えば次のようになるはずです。



ラインセンサーの値は 0 か 1 しかないので、0「でなければ」1ということになります。

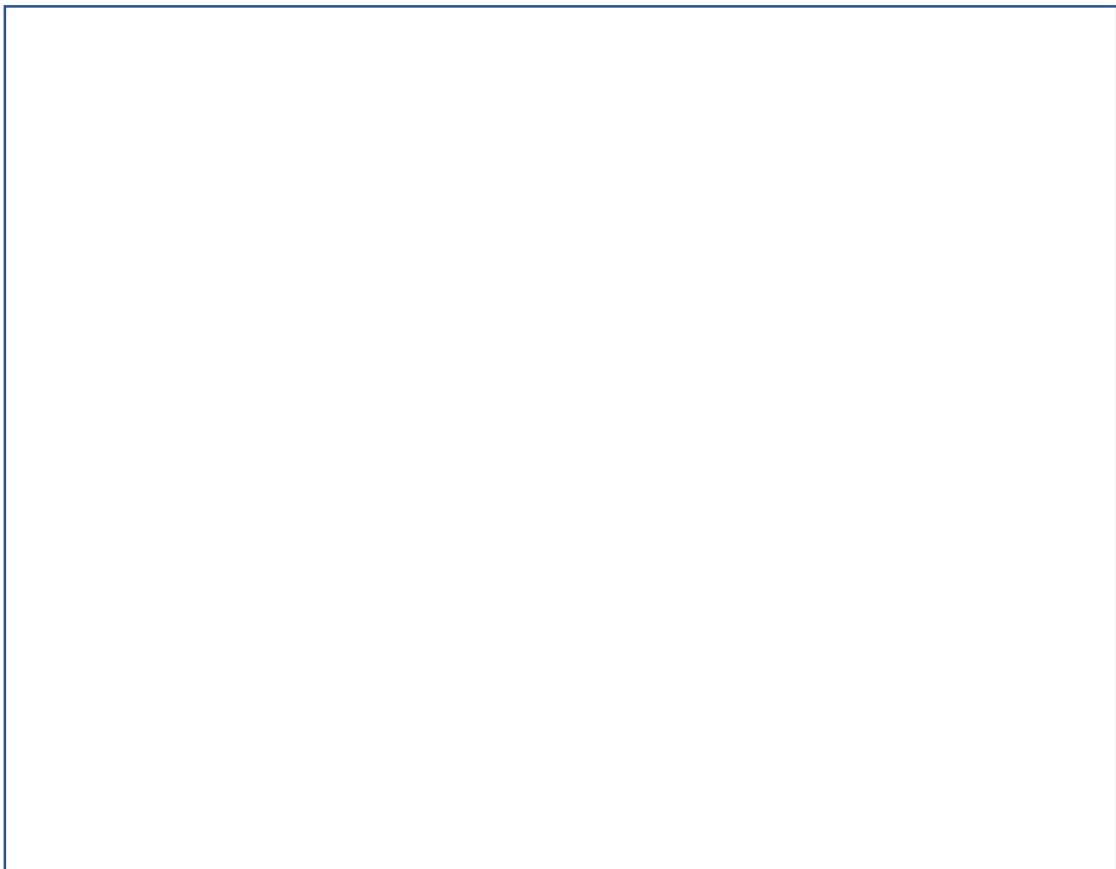
なお、上のプログラム例では左側のラインセンサーだけを使っています。

プログラムができたなら Maqueen をコースの上で走らせて、フローチャートで考えたとおりに動くか確かめてみましょう。

## 4. 自分で考えてプログラミングしてみよう

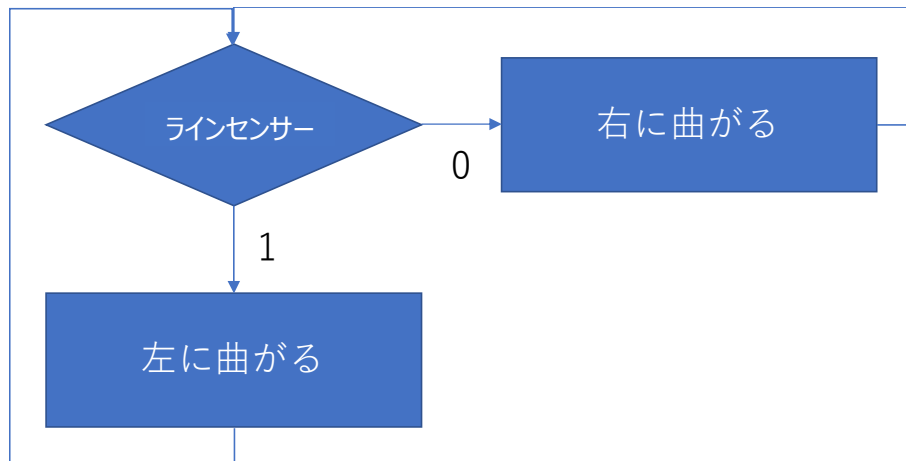
さきほどのプログラムは、Maqueen が線の上にいる間は左に曲がり、線から外れたら右に曲がるというようになっていたと思います。

今度は Maqueen が線の上にいる間は右に曲がり、線から外れたら左に曲がるプログラムを書いてみましょう。その前にまずは自分でフローチャートを下の枠の中に書いてみましょう。



### フローチャートを描いてみよう

フローチャートができたら自分で考えてプログラムを作り、考えたとおりに動くかどうか試してみましょう。



フローチャートの例



プログラムの例

最初のプログラムでは、円周のコースを線の外側ならジグザグに右回りで、線の内側なら左回りでジグザグに走るプログラムになっていたと思います。

2 番目の自分で作ったプログラムでは、線の外側を左回りで、内側なら右回りで走るようになったと思います。

なぜそうなったのか考えてみましょう。



## 5. もっと速く走るように工夫してみよう

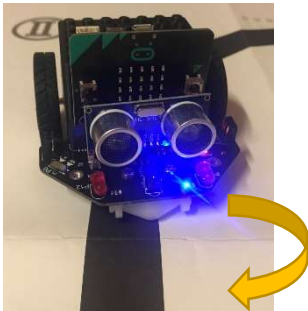
これまでのプログラムは左側のセンサーだけを使ってジグザグに進んでいたのですが、あまり速く走ることができませんでした。ラインレース用のフォトレジスタは左右にあるので、これを両方使ってもっと速くコースを走らせてみましょう。

そのためには、①まっすぐ走る、②左に曲がる、③右に曲がるの3つの場合に分けて考えてみましょう。

場合に分けてみる



- ① 両方のセンサーが 0 ⇒ まっすぐ走る  
(ラインの幅が細い場合は、両方 1 になる)



- ② 右が 1 で左が 0 ⇒ 左に曲がる

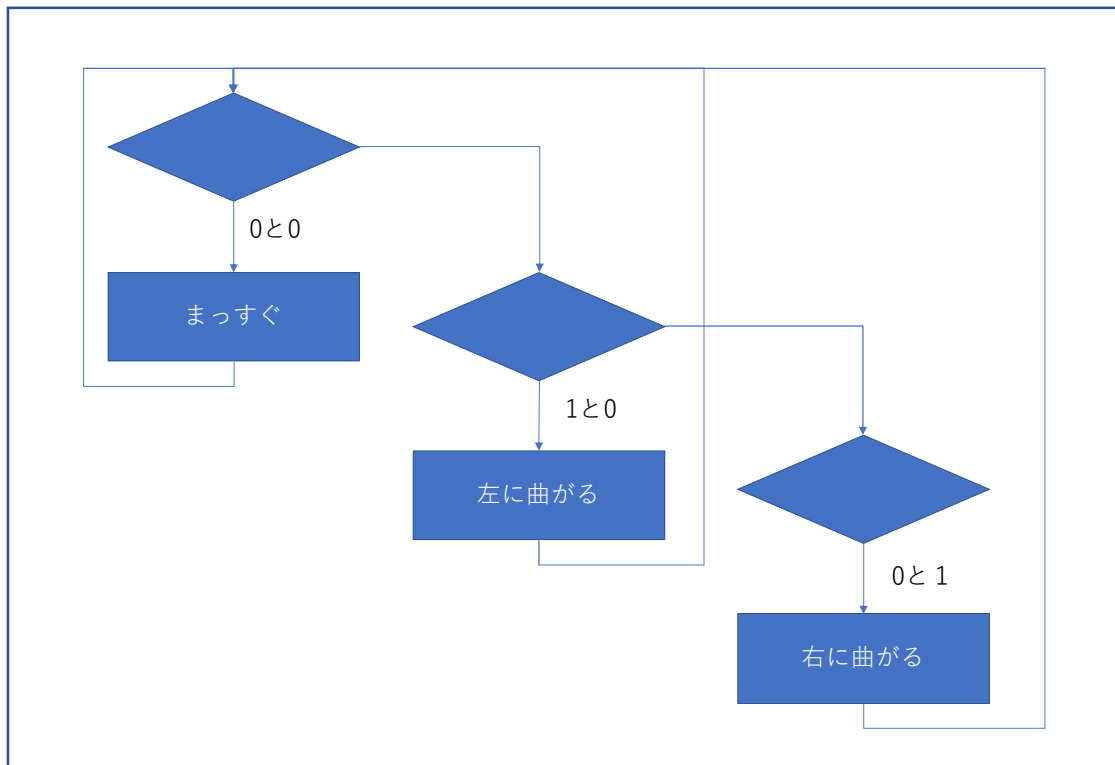


- ③ 右が 0 で左が 1 ⇒ 右に曲がる

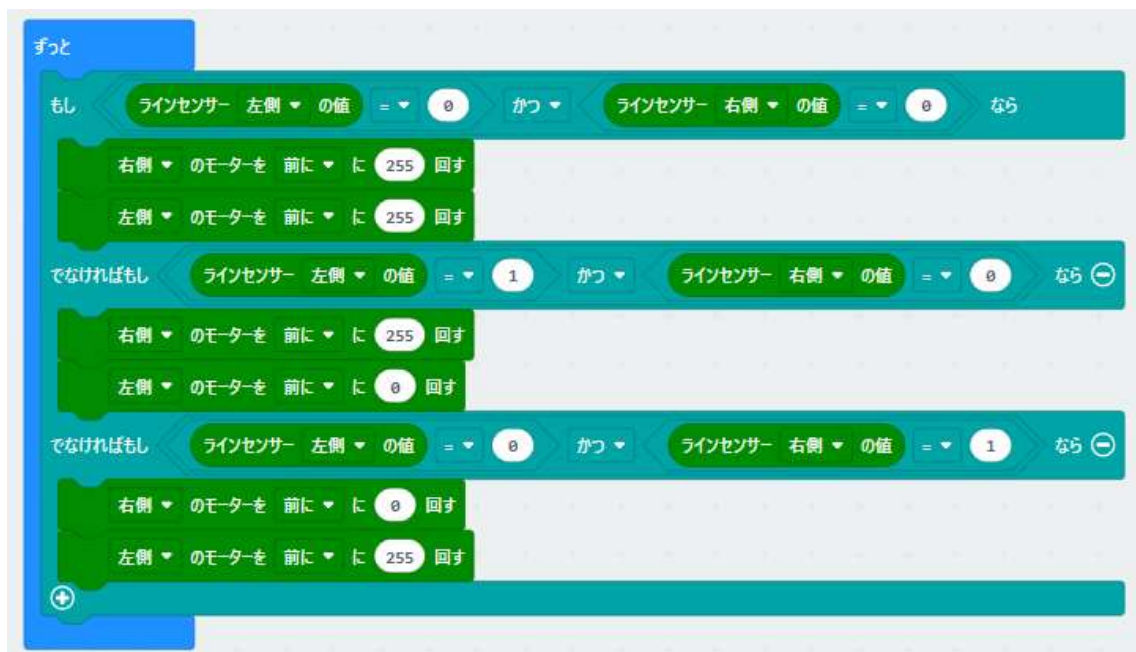
自分でフローチャートを描いてみましょう



フローチャートをプログラムにしてみましょう



フローチャートの例

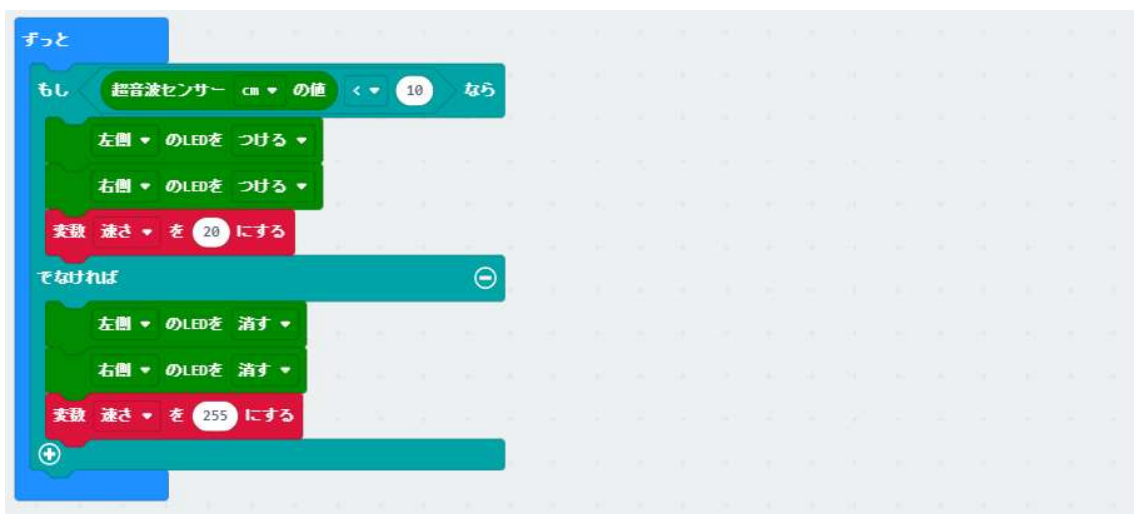


プログラムの例

## その他のプログラム例（2 台以上で）

## 1) 追いかける

超音波センサーを使って、前の車と一定の距離を保ちます。



## 2) 無線でやりとり

近づいたら止まって相手に無線で進めというメッセージを送ります。

The code is organized into several sections:

- 開発 走る speed**: The main movement loop.
  - もし** (If) condition: `ラインセンサー 左側 の値 = 0` **かつ** (and) `ラインセンサー 右側 の値 = 0` **なら** (then).
    - 左側 のモーターを 前に に `speed` 回す (Turn left motor forward at speed).
    - 右側 のモーターを 前に に `speed` 回す (Turn right motor forward at speed).
  - でなければもし** (Else if) condition: `ラインセンサー 左側 の値 = 1` **かつ** `ラインセンサー 右側 の値 = 0` **なら**.
    - 左側 のモーターを 前に に `speed` 回す.
    - 右側 のモーターを 前に に `8` 回す.
  - でなければもし** condition: `ラインセンサー 左側 の値 = 0` **かつ** `ラインセンサー 右側 の値 = 1` **なら**.
    - 左側 のモーターを 前に に `8` 回す.
    - 右側 のモーターを 前に に `speed` 回す.
  - でなければもし** condition: `ラインセンサー 左側 の値 = 1` **かつ** `ラインセンサー 右側 の値 = 1` **なら**.
    - 左側 のモーターを 前に に `speed` 回す.
    - 右側 のモーターを 前に に `8` 回す.
- 最初だけ** (Only at start):
  - 無線のグループを設定** (1) (Set wireless group to 1).
  - 変数 list を 空の配列 にする** (Initialize list as an empty array).
  - くりかえし** (8) 回 (Repeat 8 times):
    - `list の最後に 超音波センサー cm の値 を追加する` (Add ultrasonic sensor value to the end of list).
- ずっと** (Forever) loop:
  - 呼び出し 走る 速さ** (Call move block).
  - 無線で受信したとき receivedNumber** (When received number):
    - 変数 速さ を receivedNumber にする** (Set speed variable to received number).
    - 左側 のLEDを 消す (Turn off left LED).
    - 右側 のLEDを 消す (Turn off right LED).
- ずっと** (Forever) loop:
  - 変数 合計 を 0 にする** (Set total to 0).
  - `list の最後に 超音波センサー cm の値 を追加する` (Add value to list).
  - `list から最初の値を取り除く` (Remove first value from list).
  - 変数 index を 0 から 7 に変えてくりかえす** (Repeat from 0 to 7):
    - 変数 合計 を list の index 番目の値 だけ増やす** (Increase total by value at index).
  - 変数 平均 を 合計 ÷ 8 にする** (Set average to total / 8).
  - もし** condition: `平均 < 10` **なら**.
    - 左側 のLEDを つける (Turn on left LED).
    - 右側 のLEDを つける (Turn on right LED).
  - 無線で数値を送信** (255) (Send number wirelessly).
  - くりかえし** (1500) 回 (Repeat 1500 times):
    - 呼び出し 走る 20** (Call move block with distance 20).
  - 変数 速さ を 0 にする** (Set speed variable to 0).