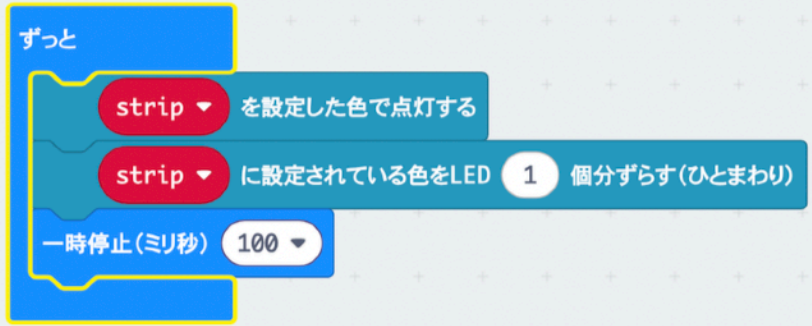
1 きらきらイルミネーション・カー

Maqeenは走るだけじゃない！！
走りながらイルミネーションを光らせることもできるのだ！
拡張機能「neopixel」ネオピクセルを追加して、挑戦しよう

「最初だけ」ブロックと・・・・

「ずっと」ブロックを追加して作る！



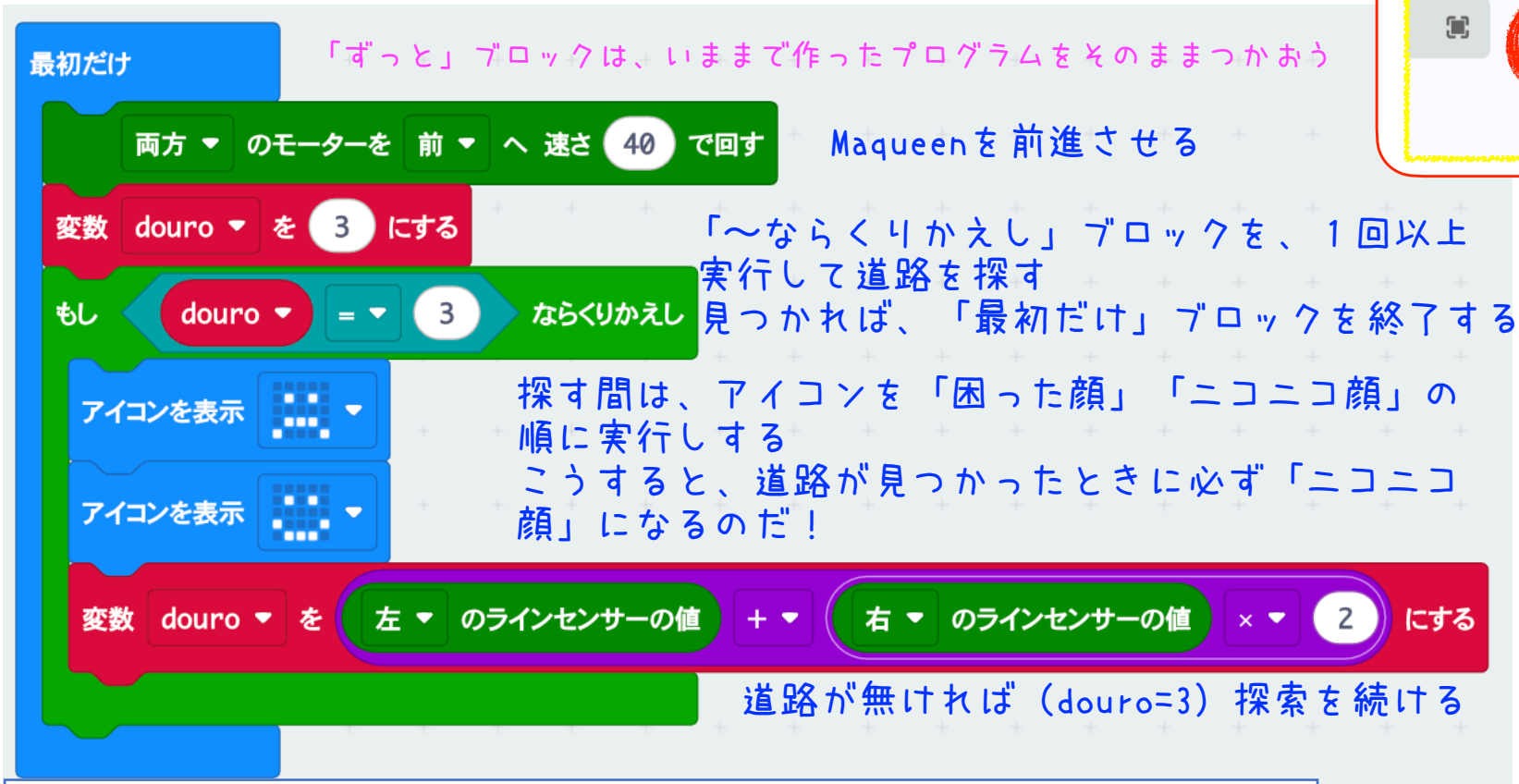
テキストで作った
プログラムはそのまま

「元に戻す」ボタンで間違いをやり直しできる❣
ー＋ボタンで、ブロックを拡大・縮小して見やすくしよう！！

micro:bitでのプログラミングでは「ずっと」ブロックを複数作ることができます。
こうすると、２つの以上のプログラムを同時に実行することができるようになります。
しかし、それぞれの処理速度は、CPUの能力制限によって、どんどん遅くなっていきます。
このしくみを、タスク並列処理（マルチタスクプロセス）とよびます。

2 道路を探してスタート！

テキスト通りだと、スタートは道路上に置く必要がある
でもこれな、コース外からスタートしても道路を見つけるぞ



「ずっと」ブロックは、いままで作ったプログラムをそのままつかおう

Maqueenを前進させる

「〜ならくりかえし」ブロックを、１回以上実行して道路を探す
見つかれば、「最初だけ」ブロックを終了する

探す間は、アイコンを「困った顔」「ニコニコ顔」の順に実行しする
こうすると、道路が見つかったときに必ず「ニコニコ顔」になるのだ！

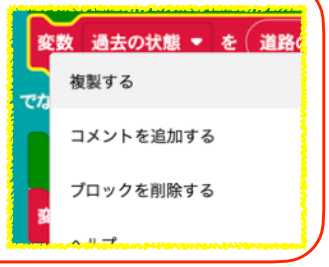
道路が無ければ（douro=3）探索を続ける

「最初だけ」は、電源を入れたときに最初の１回だけ実行するブロックです。
そして、プログラムの中では１つだけおくことができます。
チャレンジ１のプログラムも、実行する場合は、
両方のプログラムを続けて並べて作ります。

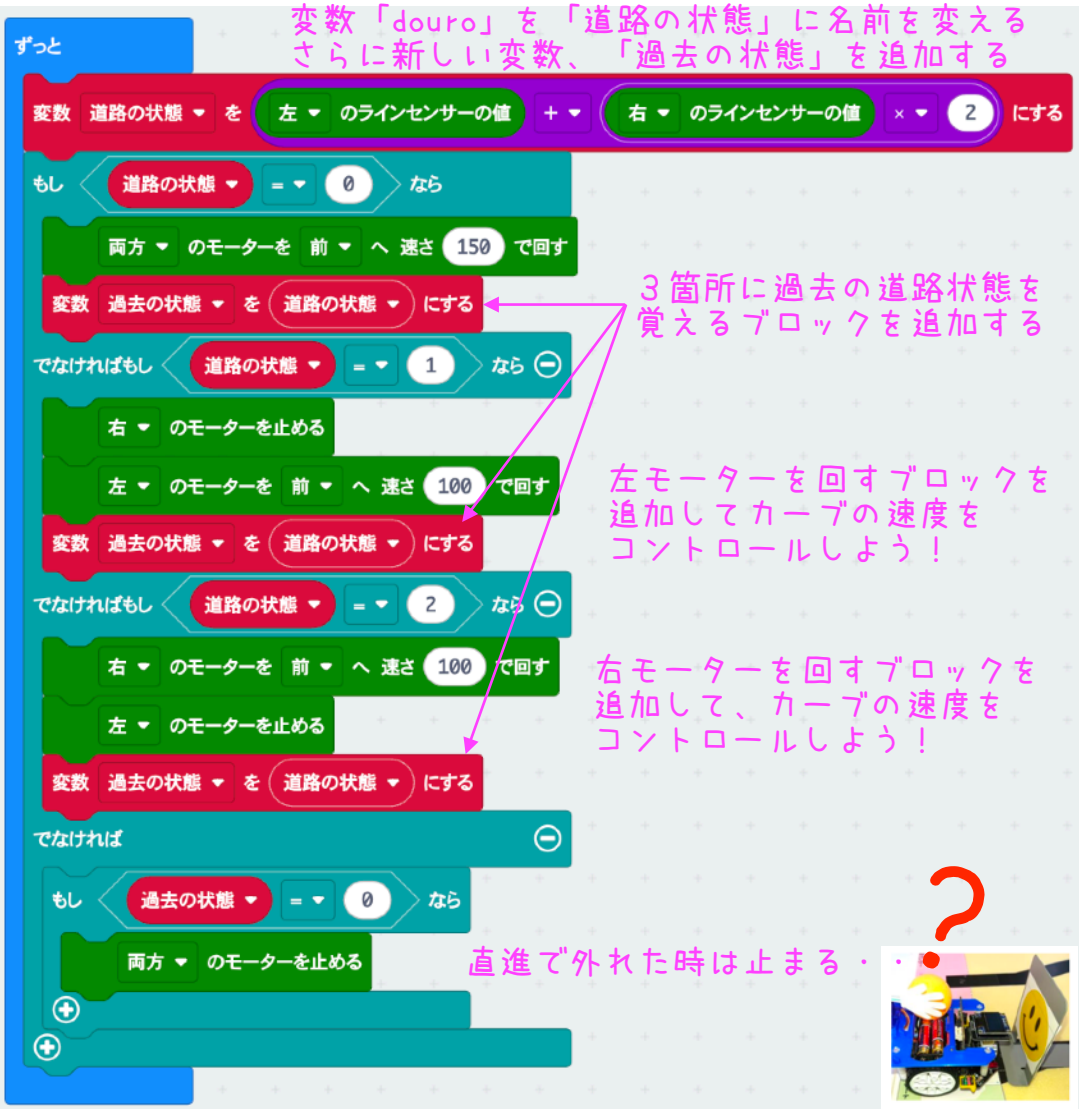
3 コースへ復帰！

道路から外れても自動でコースに復帰！

右クリック、または二本指タップをつかおう！
ブロックのコピーや削除が、一発でできるワザだ！



変数「douro」を「道路の状態」に名前を変える
さらに新しい変数、「過去の状態」を追加する



3箇所に過去の道路状態を覚えるブロックを追加する

左モーターを回すブロックを追加してカーブの速度をコントロールしよう！

右モーターを回すブロックを追加して、カーブの速度をコントロールしよう！

直進で外れた時は止まる・・・？

maqueenが止まったままで動かなくなるのは、直進中に道路から外れたときと、左右のカーブ途中で外れたときの、二つの場合なのだ

しかし、カーブ途中はそのまま走り続ければ、もとの道路に戻れるはず

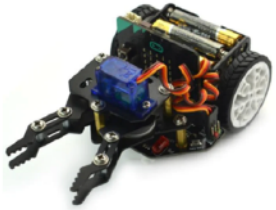
なので、「過去の状態」が直進中のときだけ、停止するように改良した

また、カーブでスピードを落とせば、スムーズに走行できるはずだ！

※チャレンジ１、２のプログラムはそのまま残しておいてもOK

3 のつづき！

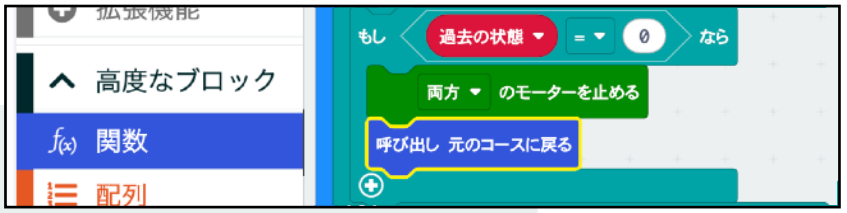
無敵Maqueen



道に迷ったら脱出するぞ

ぜったいに
コースを見つける！

関数ブロック「呼び出し 元のコースに戻る」を追加する




```
function 元のコースに戻る
一時停止(ミリ秒) 500
もし 道路の状態 = 3 ならくりかえし
    右 のモーターを 後ろ へ 速さ 50 で回す
    左 のモーターを 後ろ へ 速さ 20 + 0 から 3 までの乱数 × 10 で回す
    変数 道路の状態 を 左 のラインセンサーの値 + 右 のラインセンサーの値 × 2 にする
一時停止(ミリ秒) 100
両方 のモーターを止める
一時停止(ミリ秒) 500
両方 のモーターを 前 へ 速さ 30 で回す
```

ちょっと休む・・このブロックはなくても大丈夫かもしれない

チャレンジ２を挑戦したなら、ブロックをコピーしてもってこよう！

道路が見つかるまでバックで探す　バック以外の脱出方法も考えてみよう

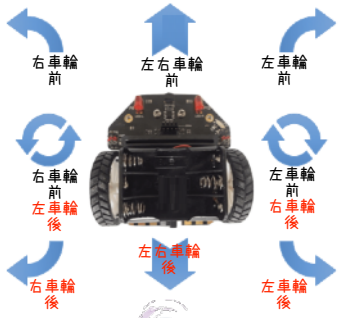
計算の順序に注意！！
20 + (randint(0, 3) * 10)
この式の結果は、20, 30, 40, 50, のいずれかになる

ちょっと休む・・このブロックはなくても大丈夫かもしれない

道路の上にいるはずなので、ここでMaqueenを止める

ちょっと休む・・このブロックはなくても大丈夫かもしれない

ゆっくり前に進めて、
ライントレース（ずっとブロック）処理にもどる



関数はプログラミングを行う上で、たいへん重要な要素です。
たとえば、何度も同じ処理を使いたい時や、独立した機能として処理したい場合などには必ず使われます。
実は「モーターを回す」などのブロックも関数であることが多いのです。
関数を使うと、プログラムの構造もが見やすくなり、デバックやテストが効率的になります。

NEXTDAY
http://www.nextday.jp/

jpw2024v6G