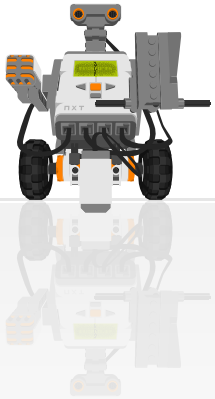
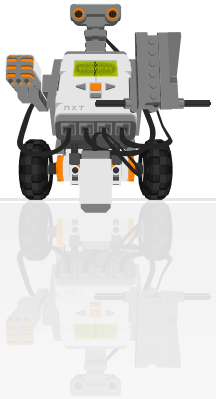


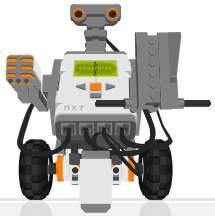
実践ロボットプログラミング

LEGO Mindstorms NXT で目指せロボコン！

WEB：http://robot-programming.jp/
著者：藤吉弘亘，藤井隆司，鈴木裕利，石井成郎
E-mail：support@robot-programming.jp

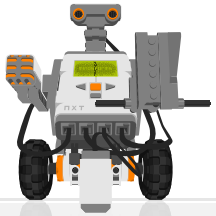


■LEGO Mindstorms NXTについて



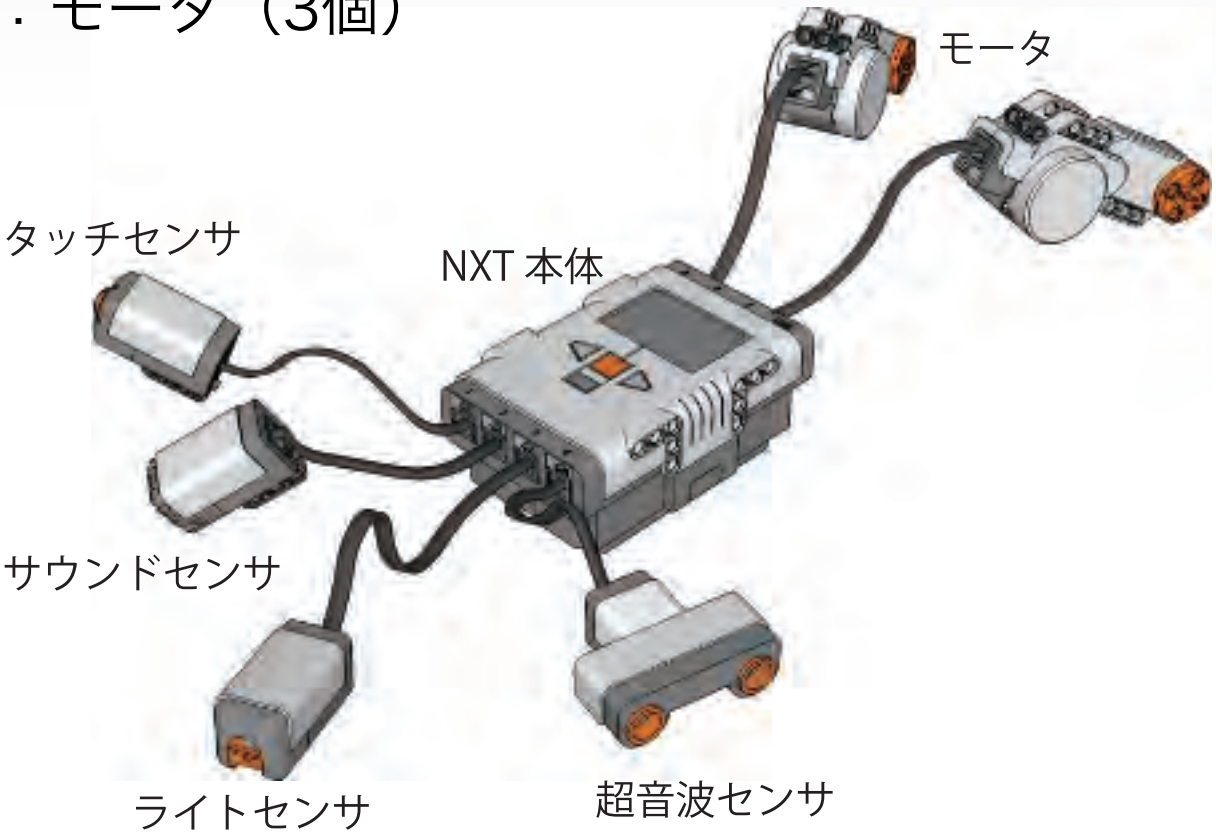
VS.

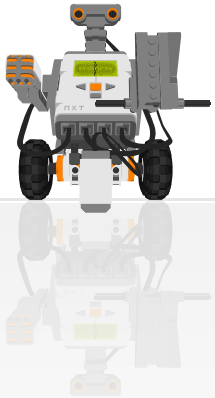
	RIS	NXT
発売時期	1998年	2006年
CPU	H8（8 bit）	ARM7（32 bit）
クロック周波数	16MHz	48MHz
RAM	32KB	64KB
フラッシュメモリ	なし	256KB
転送方法	赤外線通信	USB/Bluetooth
ポート数	入力:3 出力:3	入力:4 出力:3
駆動	電池	電池／バッテリーパック



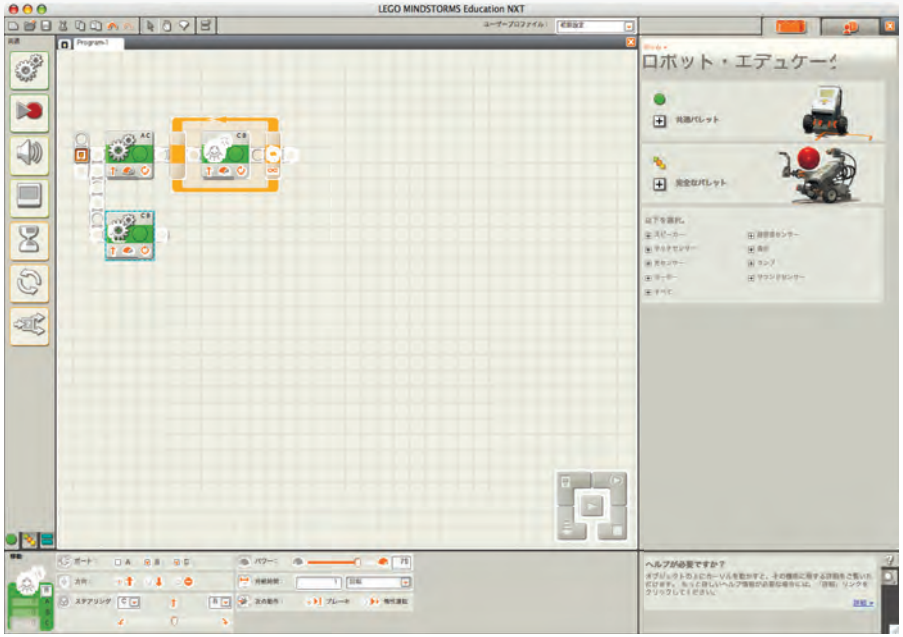
LEGOロボット構成

- 入力：タッチセンサ（２個）ライトセンサ、超音波センサ、サウンドセンサ
- 出力：モータ（3個）

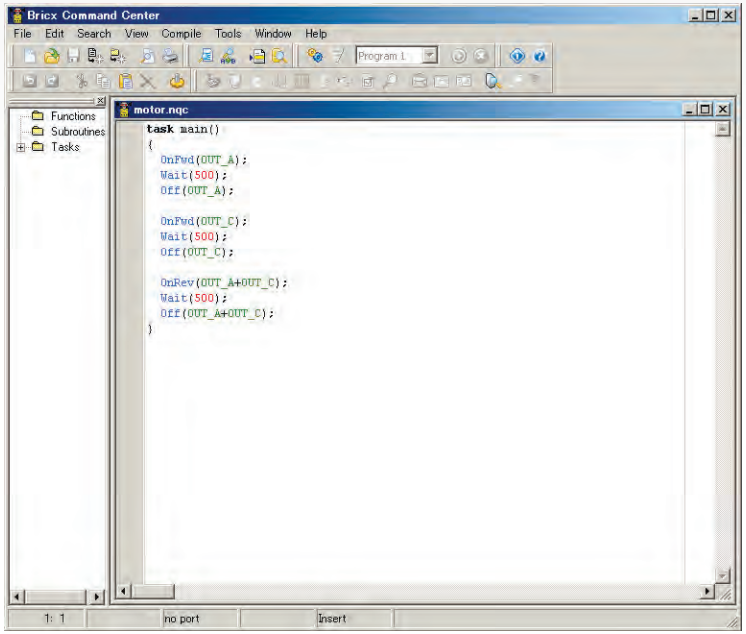




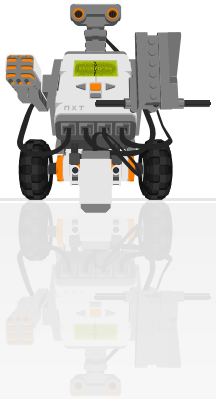
プログラミング環境



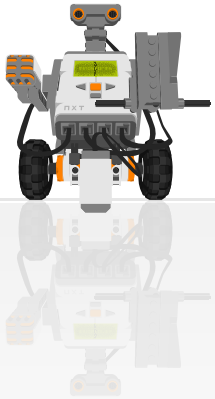
NXT-SW



NXC

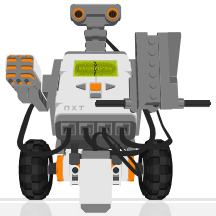


■プログラムを作成するには



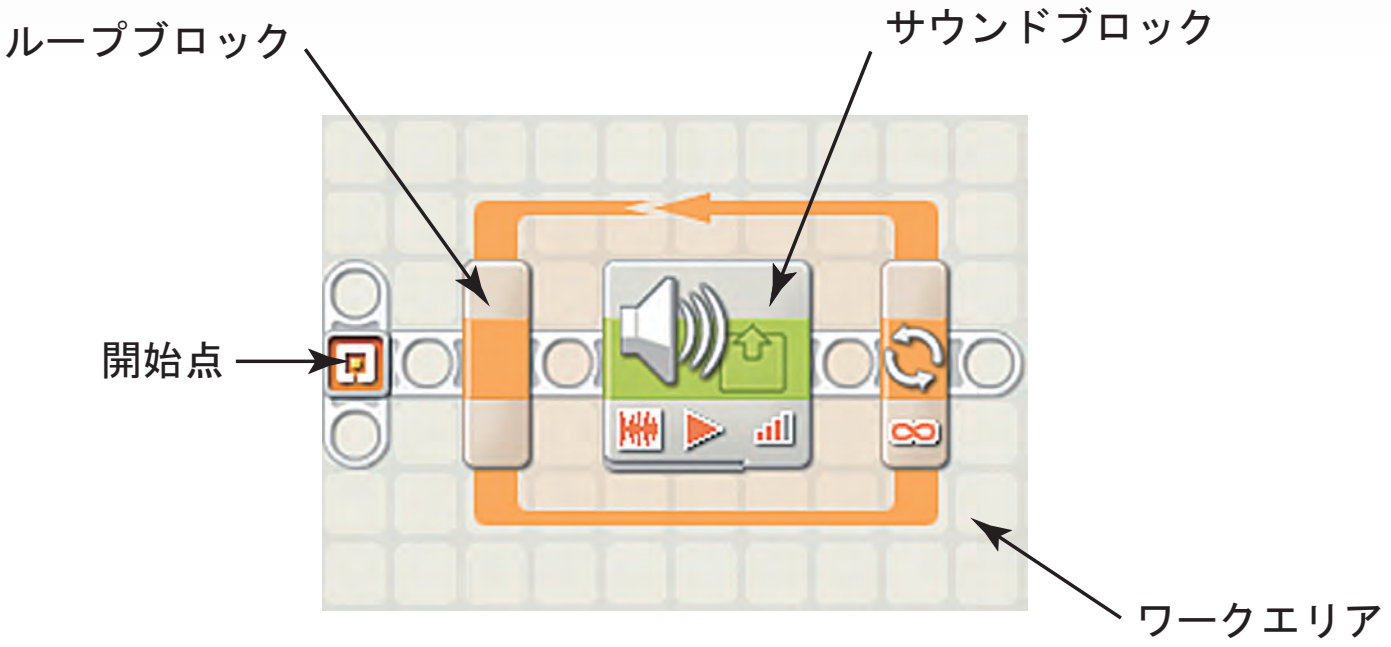
プログラム実行までの流れ

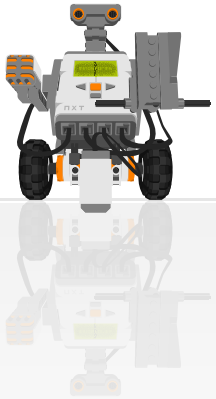
1. PC上でプログラム(NXT-SW)を作成
2. USB経由でロボットへダウンロード
3. ロボット上でプログラムを実行



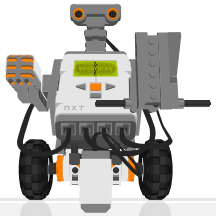
NXT-SW

- NXT-SW
  - ブロックを並べてプログラムを作成



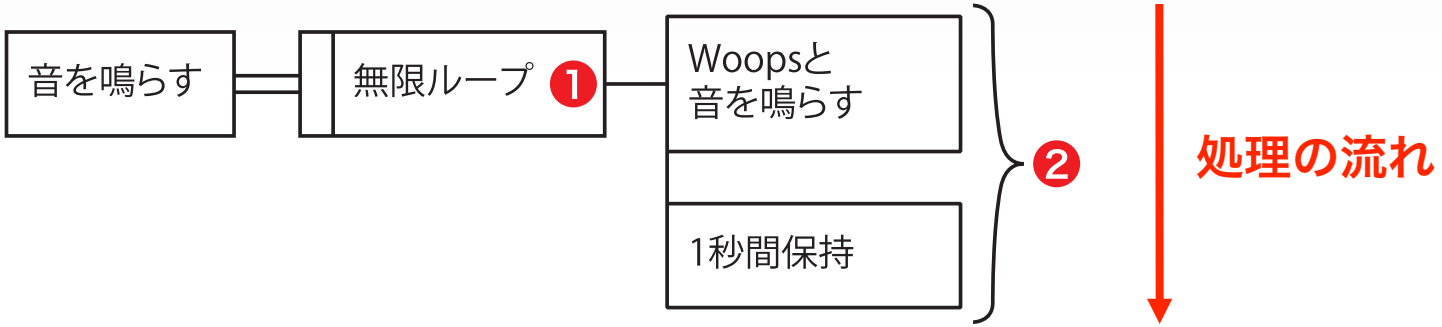


■音を鳴らしてみよう



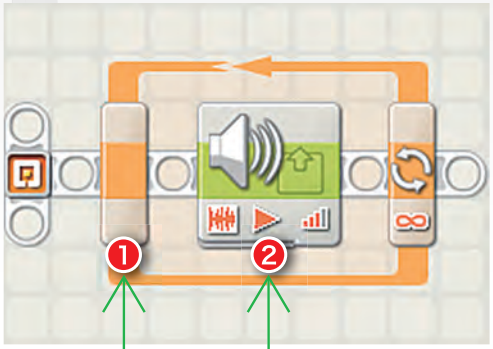
音を鳴らすプログラムのPAD(p.30)

- 音を鳴らすアルゴリズム



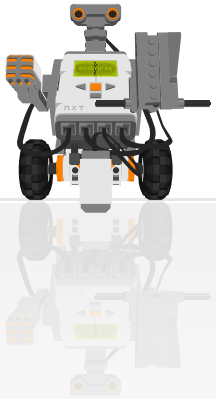
NXT-SWプログラム (p.32: sound.rbt)

sound.rbt

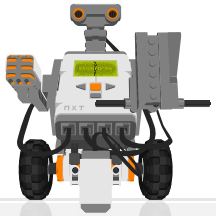


無限にループ内のブロックを
繰り返す

音声ファイル”Woops”を再生
音が重ならないように「完了待ち」
にチェックを入れる



■プログラムを実行してみよう



プログラムの転送

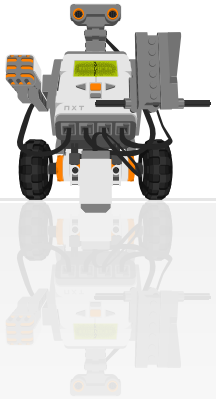
ダウンロードして実行：
転送した後、自動的に実行

NXTウインドウ：
NXTウインドウの表示

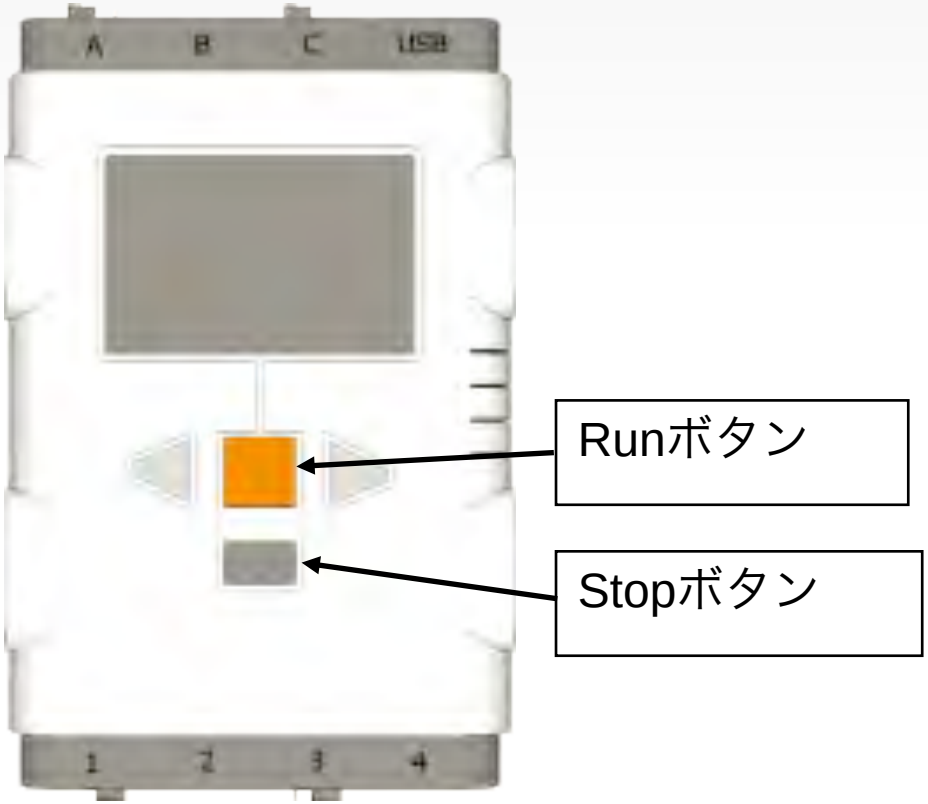
一部実行：
選択した部分を転送して実行

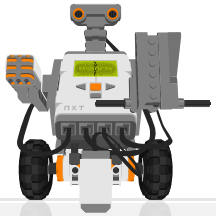
ダウンロード：
プログラムを転送

停止：
実行しているプログラムを停止



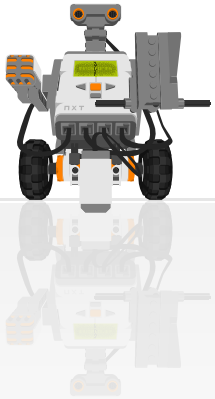
プログラムの実行



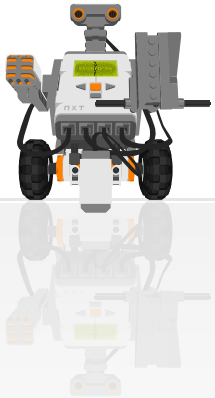


プログラムの実行

- 実行時の注意
  - ロボットの動作より作成したプログラムのアルゴリズムが実現できているか確認
  - ロボットが目的に応じた動作をしないときは、ロボットの動きをよく観察しデバッグすること
  - NXT本体の空きメモリが足りなくなったら不要なファイルを消す



■ロボットの組み立て



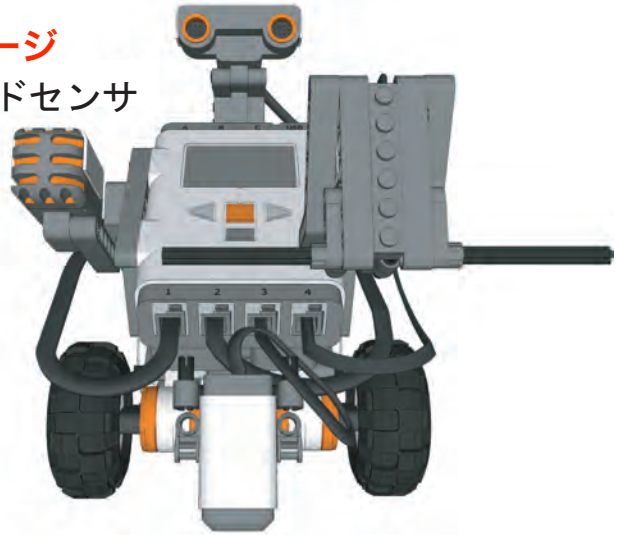
ロボットの組み立て(p.39)

Mindstorms組み立て説明書の8-22ページを
参考にローヴァーボット型ロボットを作成

超音波センサ 28-31ページ

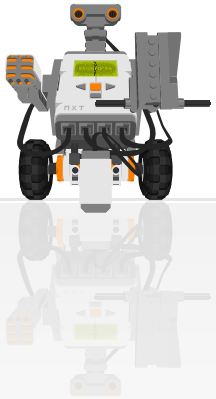
24-27ページ
サウンドセンサ

40-45ページ タッチセンサ

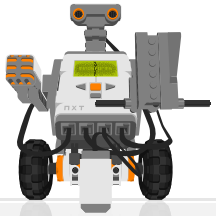


ライトセンサ
32-35ページ

ローヴァーボット型ロボット

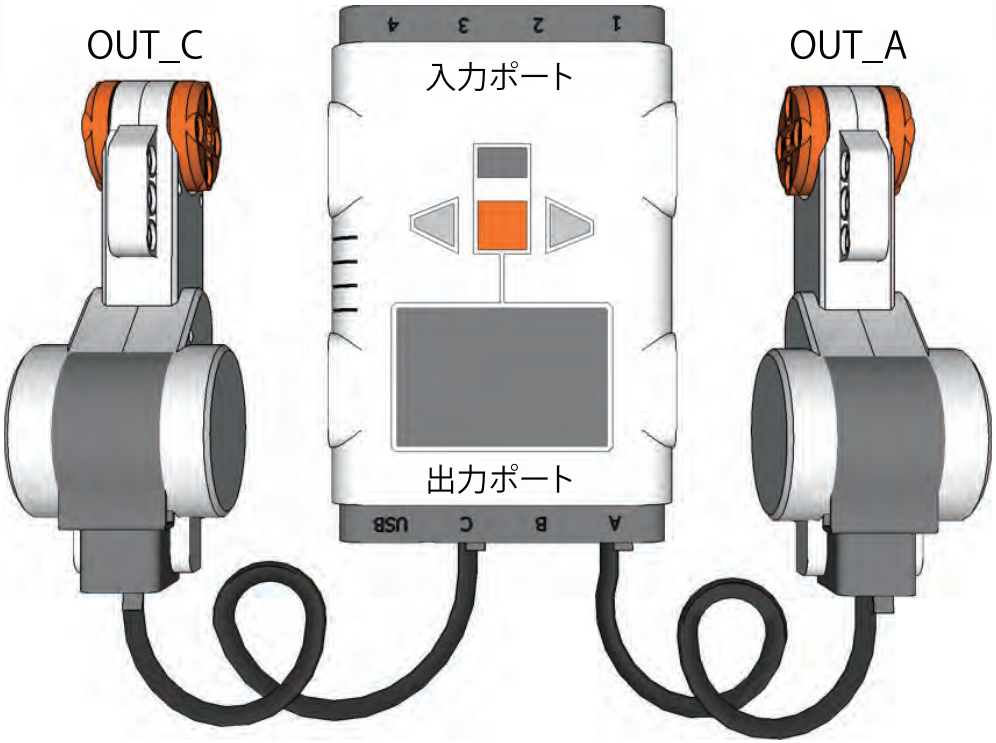


■ロボットを前進させるには(モータ制御1)



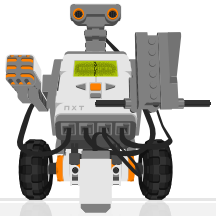
モータの接続

- NXTの出力ポートAとCにモータを接続



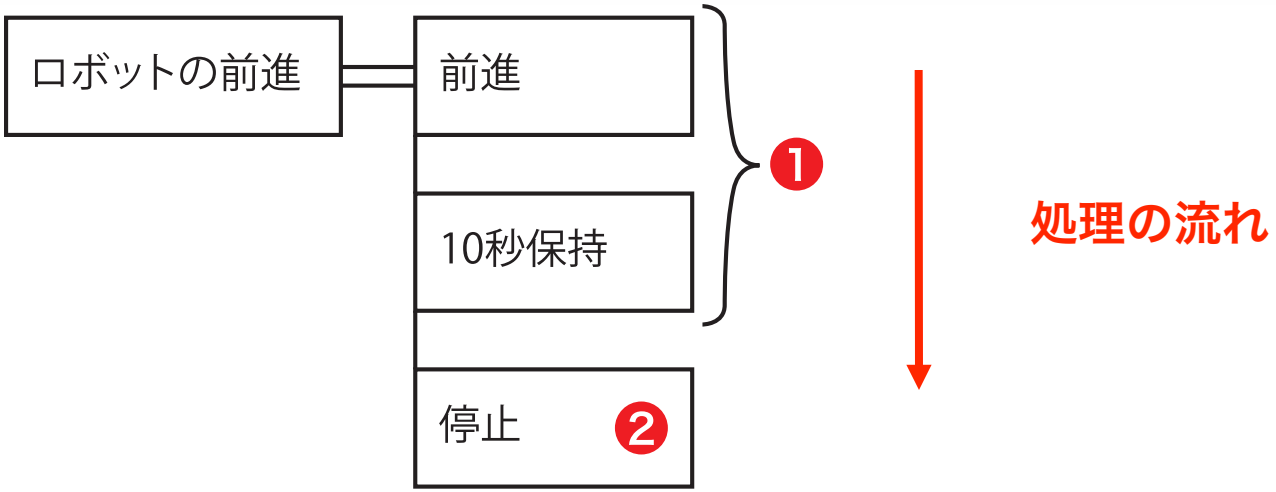
左のモータ：OUT_C

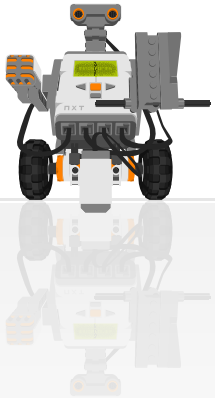
右のモータ：OUT_A



前進プログラムのPAD

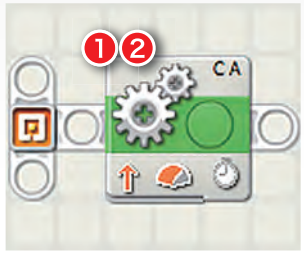
・ 10秒前進するアルゴリズム



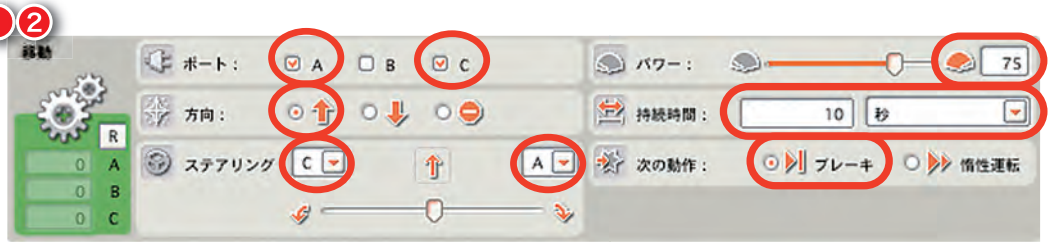


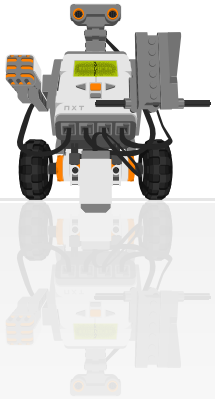
モータ制御によるロボットの前進 (p.42: forward.rbt)

forward.rbt

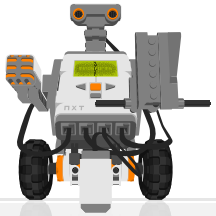


ポート A , C のモータを
パワー 75 で順方向に回
転（前進）



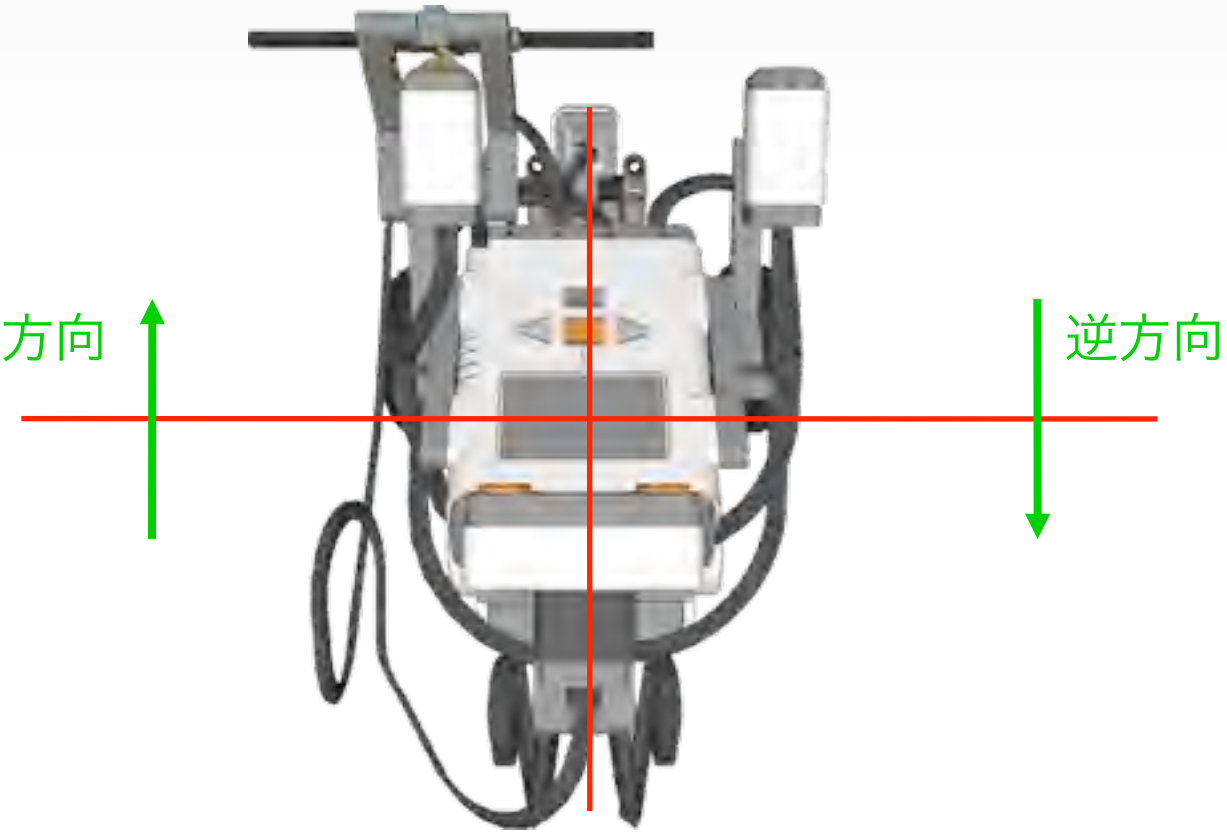


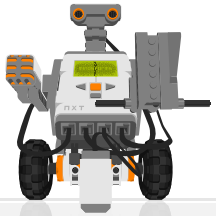
■ロボットを回転させるには (モータ制御2)



回転

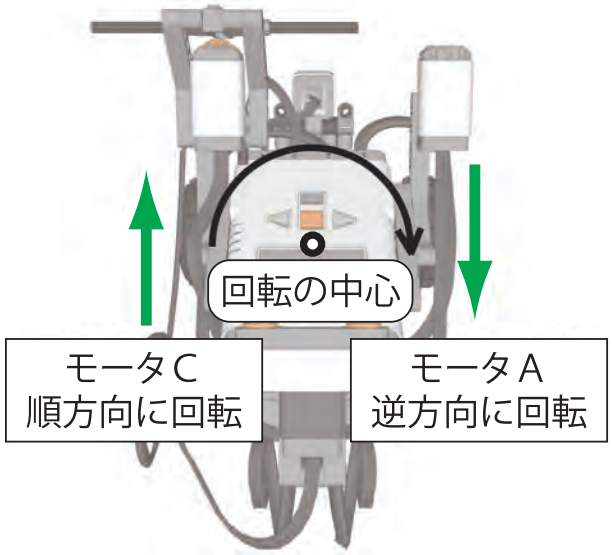
- ロボットを右に回転させるには

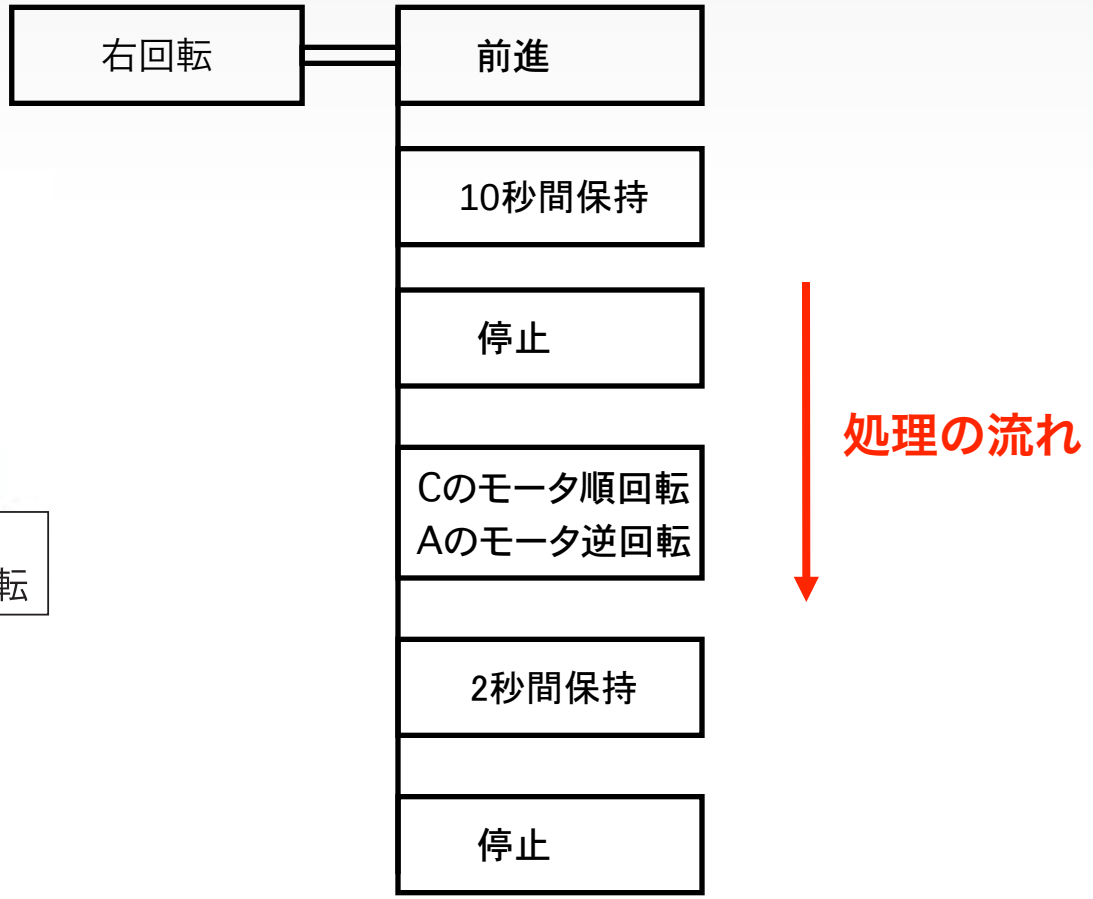


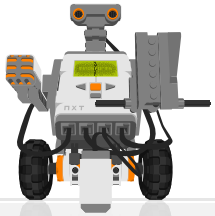


右回転プログラムのPAD

・ 右回転するアルゴリズム

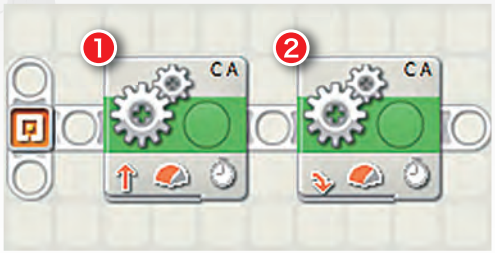






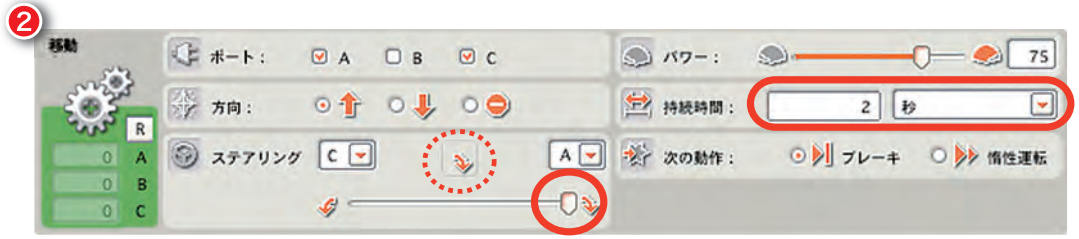
モータの制御２ (p.45: rotation.nxc)

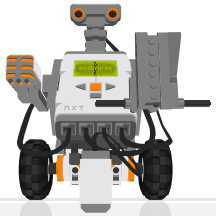
rotation.rbt



ポート A , C のモータを
パワー 75 で順方向に回
転（前進）

右回転



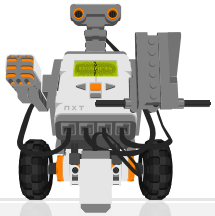


ロボットを90度回転させるには

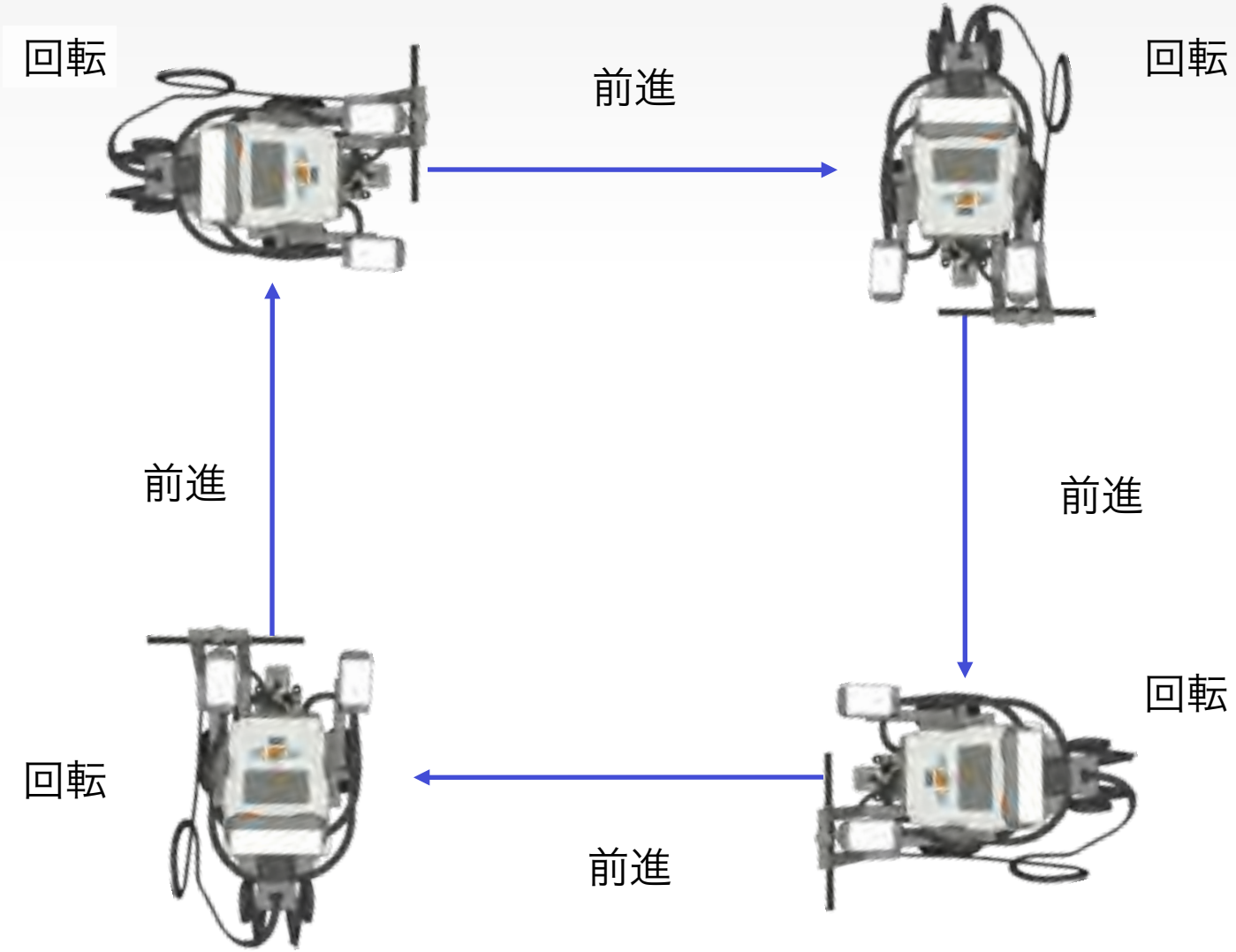
- 90度回転させるには？

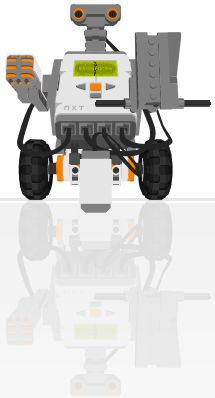
  →保持させる時間とモーターパワーを調節する

0.5秒に変更

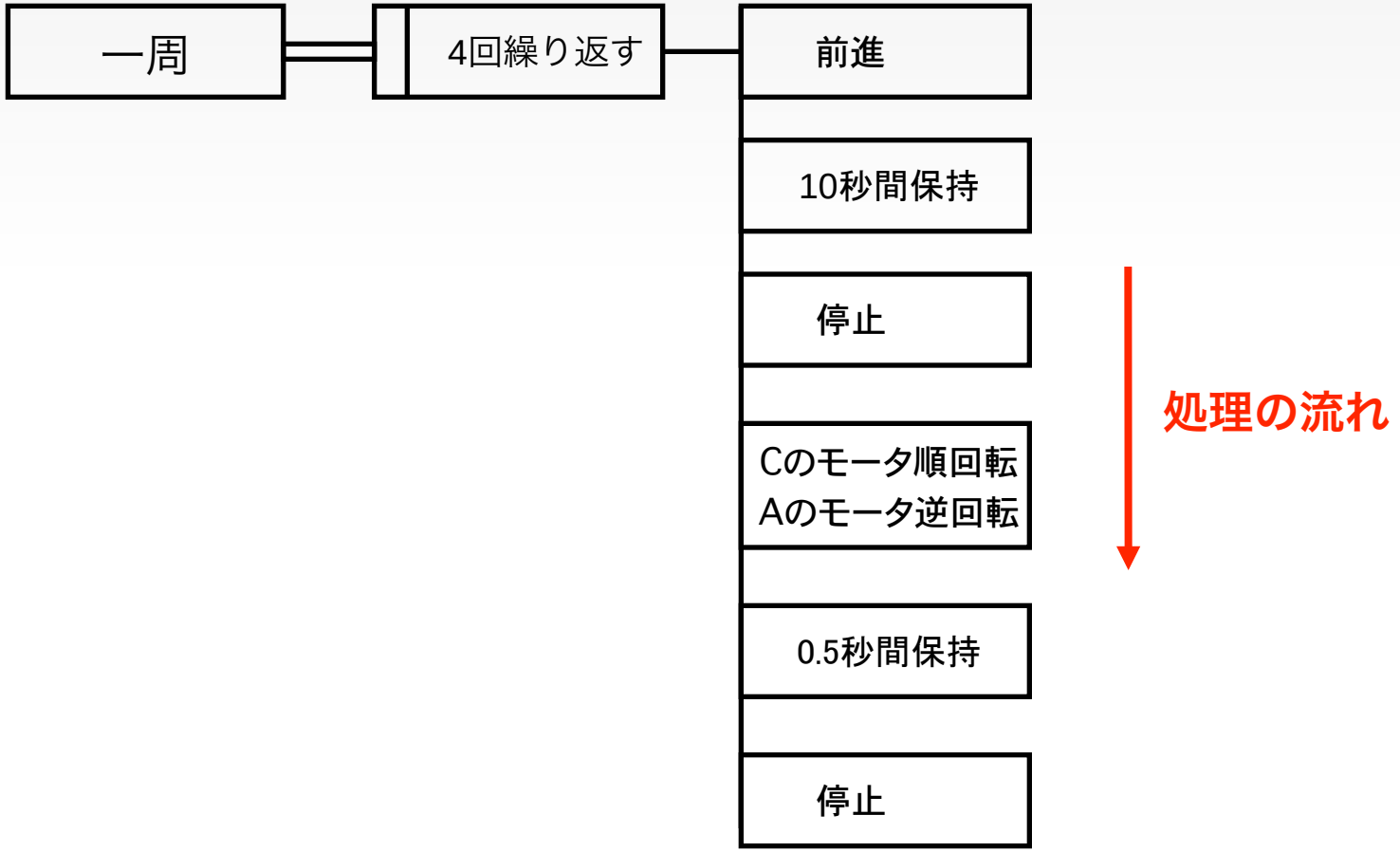


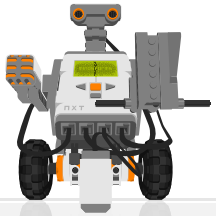
一周するには？





一周するプログラムのPAD (p.47)





プログラム（一周）

- 一周するには

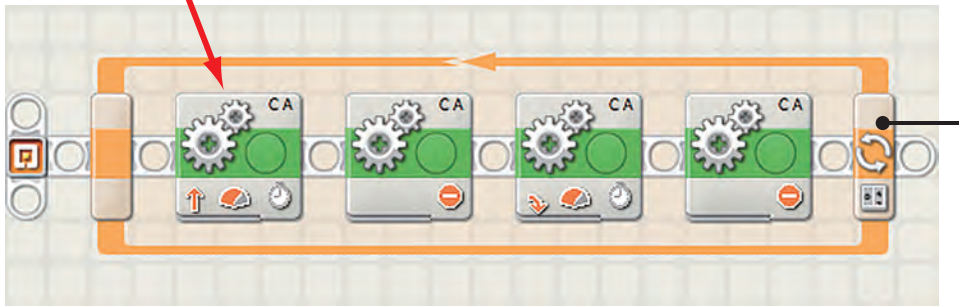
  →前進と回転を4回繰り返せばよい

(a)

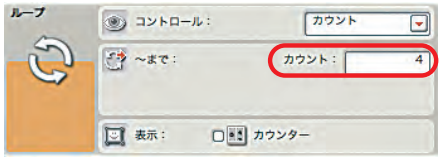
100 周するには 1600 個の
ブロックを並べる必要あり

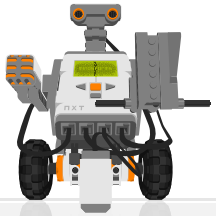
=

(b)



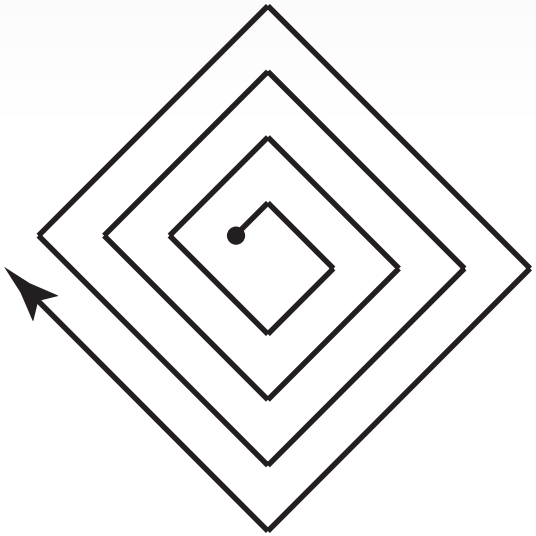
ループブロック

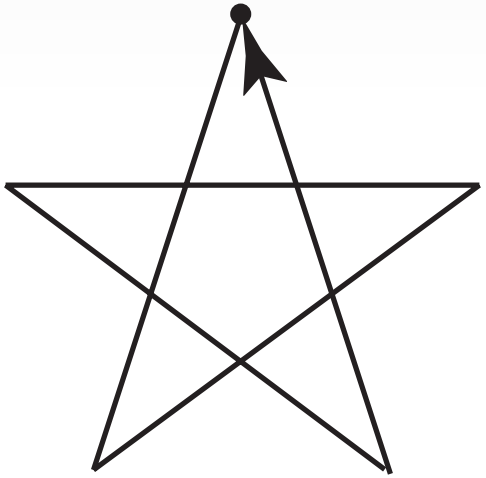


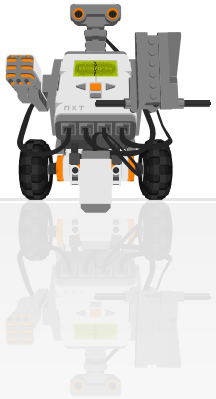


演習問題4-5 (p.49)

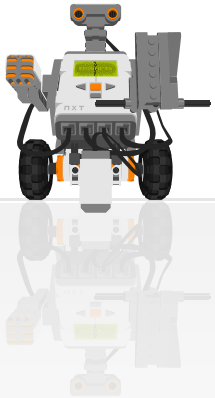
- スパイラルや星形の軌跡を描くロボットの動きを実現



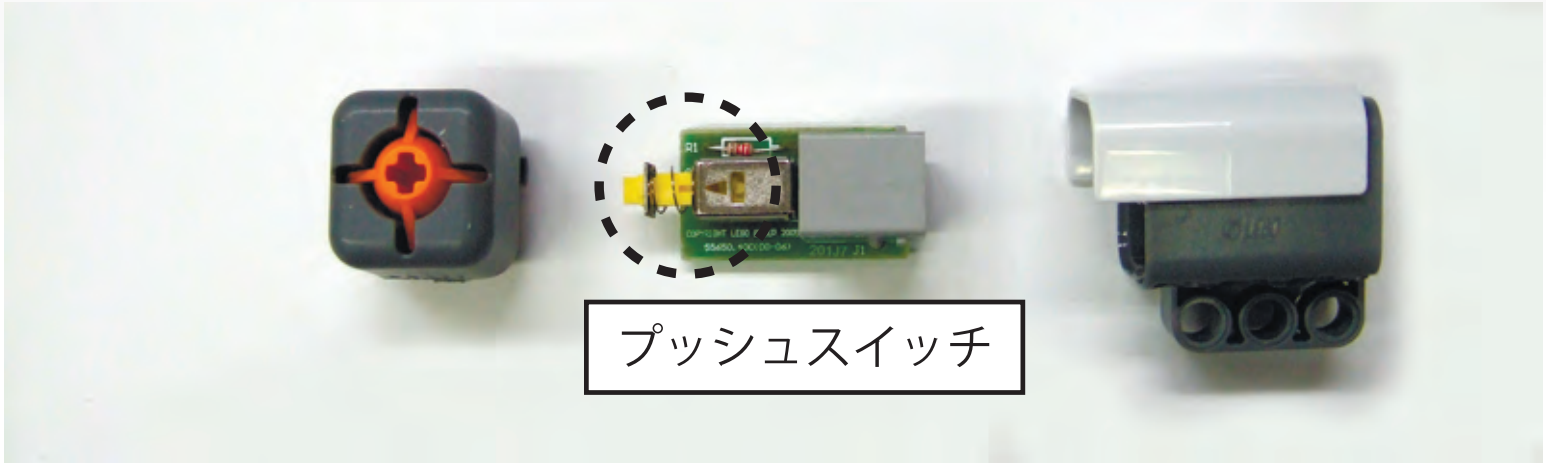




■障害物回避(タッチセンサ)

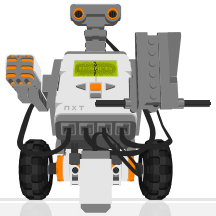


タッチセンサ



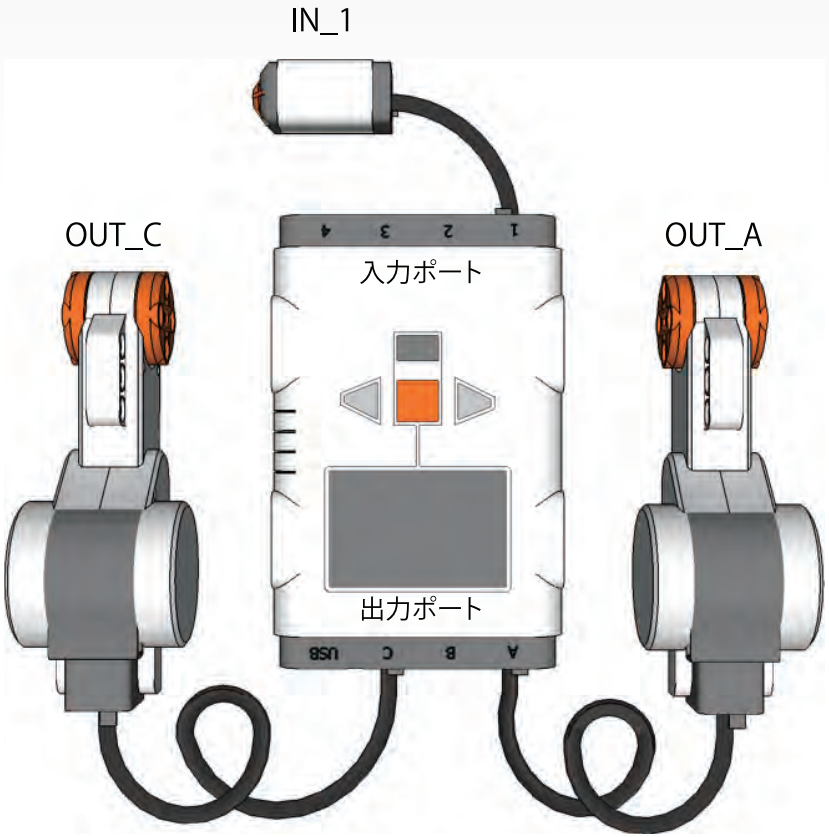
タッチセンサの原理

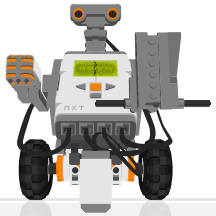
対象物にぶつかりタッチセンサが押されると中のプッシュスイッチが押されスイッチがＯＮとなる



タッチセンサの接続

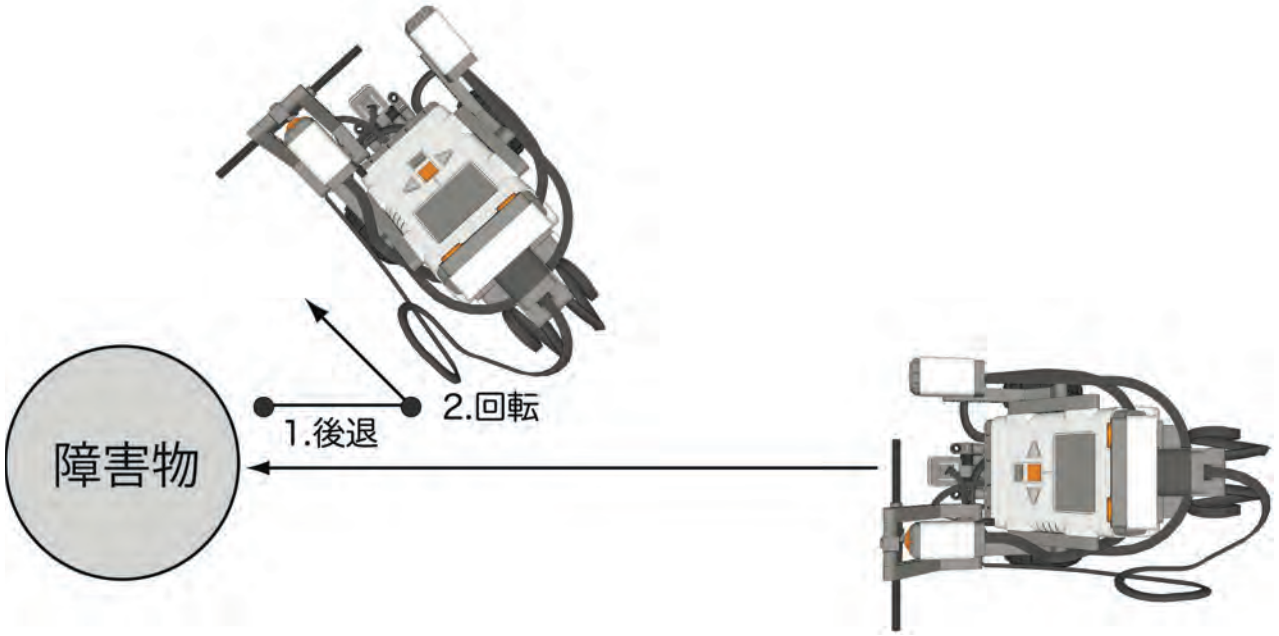
- NXTの入力ポート１タッセンサを接続

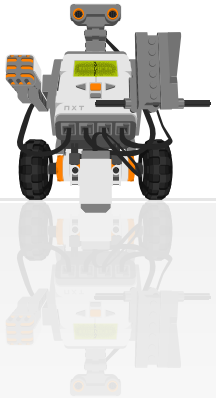




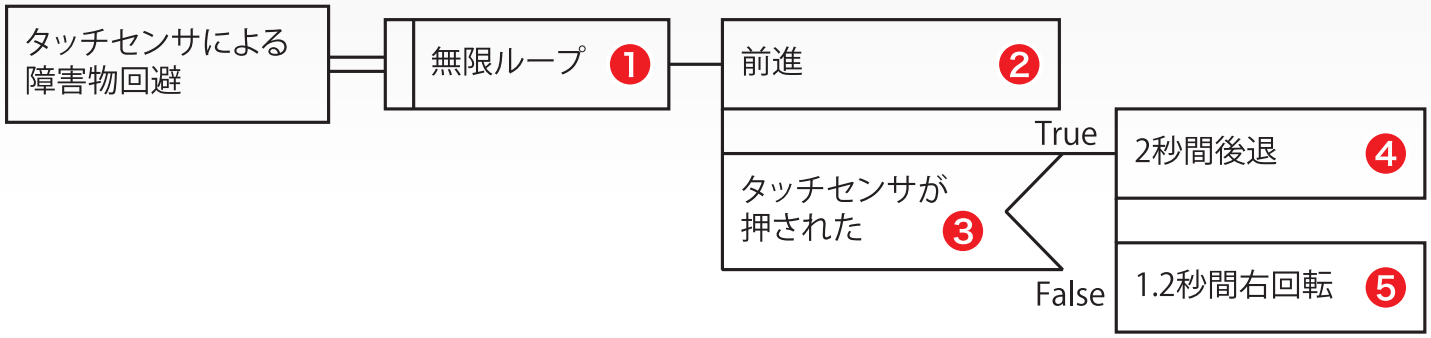
タッチセンサによる障害物回避

1. 常にロボットを前進　→　無限ループの利用
2. タッチセンサが押されたら，障害物と判定　→　条件分岐
3. 衝突と判定したら，一定時間後退し，右回転．
   その後1. に戻る

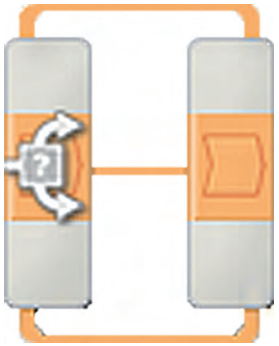




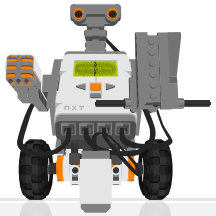
PADによるアルゴリズムの図示 (p.50)



条件分岐：スイッチブロック

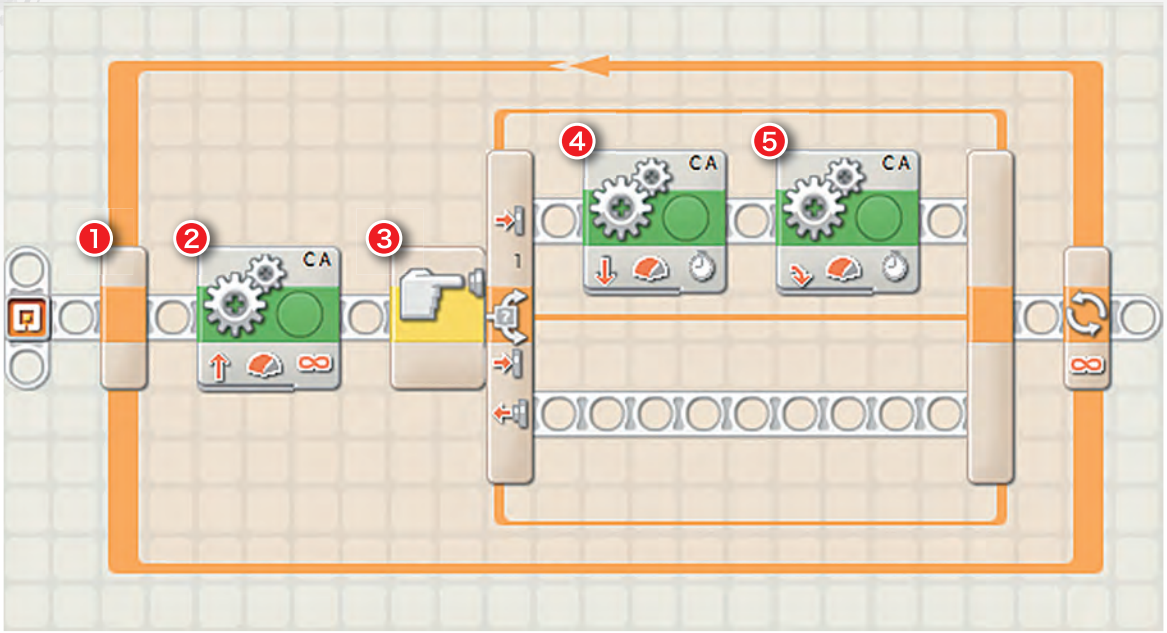


条件を満たす（真）と上段を実行
それ以外は下段を実行



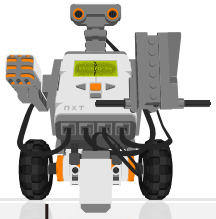
タッチセンサによる障害物回避 (p.52: touch.rbt)

touch.rbt



ループ内のブロックを無限に繰り返す
(無限ループ)

ポート A , C のモータを一瞬だけ順方向に回転して次のブロックへ移る

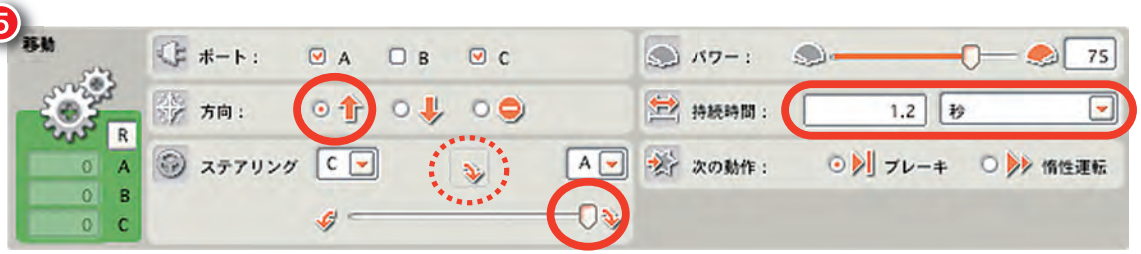


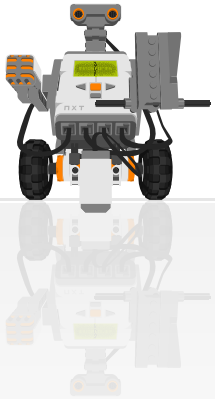
タッチセンサによる障害物回避 (p.52: touch.rbt)

ポート１のタッチセンサが押されたか押されていないかで動作を変える（条件分岐）

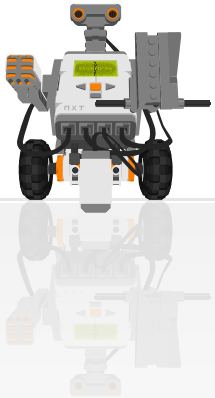
ポート A, C のモータを逆方向に回転（後退）

その場で右回転

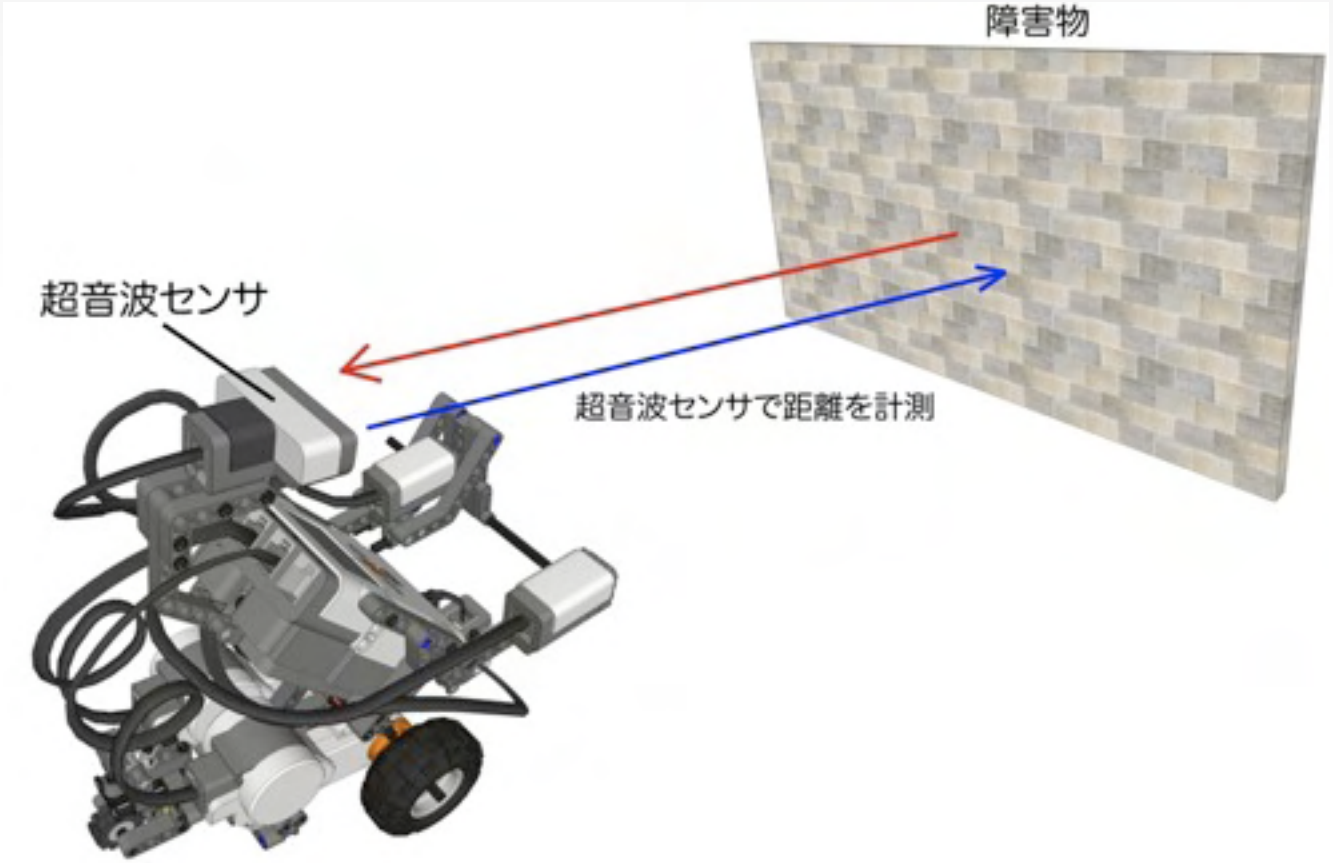


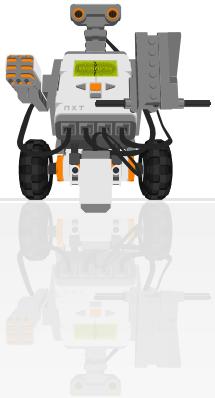


■障害物回避(超音波センサ)

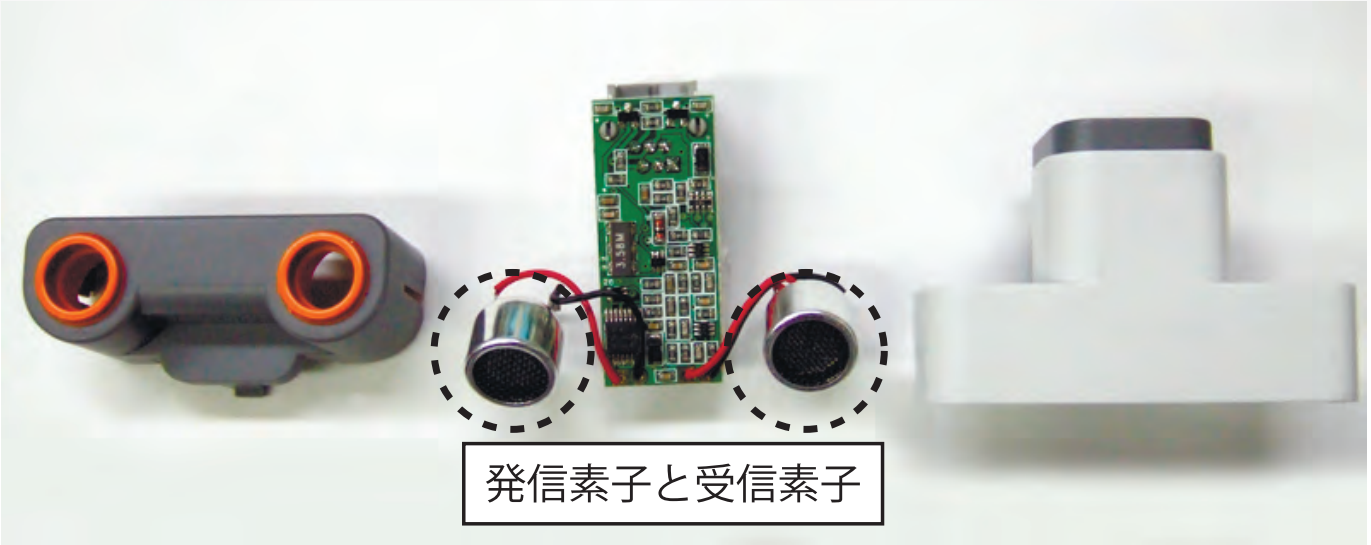


超音波センサによる障害物回避



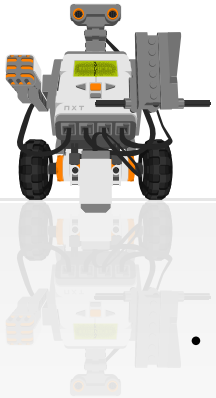


超音波センサ



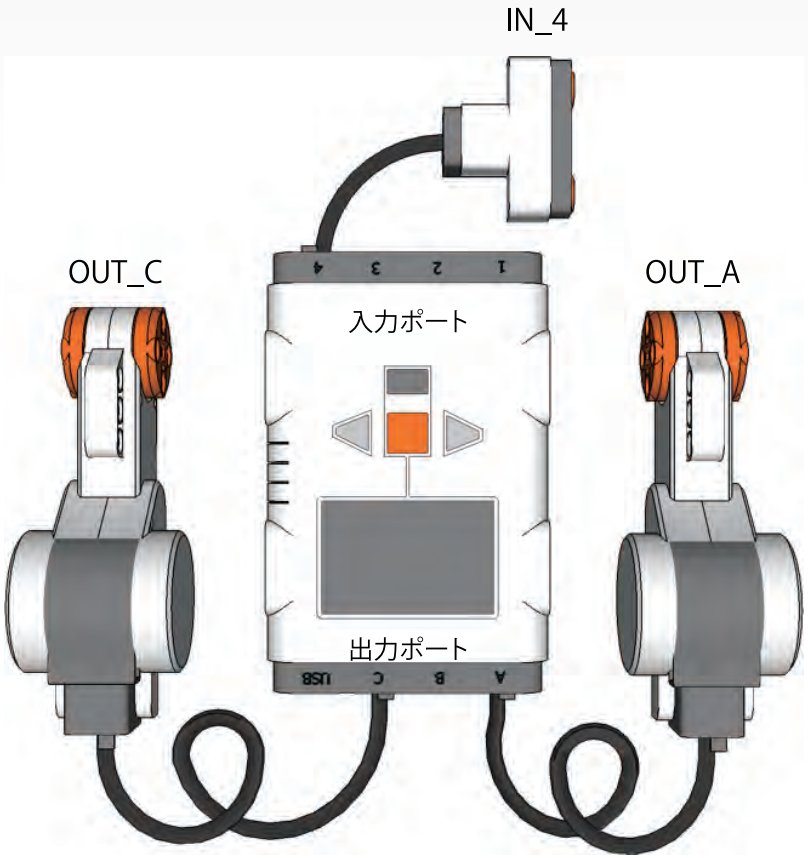
超音波センサの測定原理

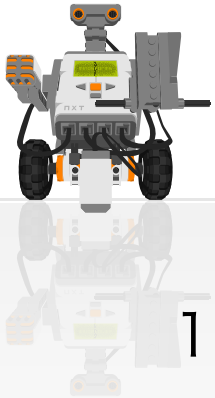
超音波を発信し、対象物で反射した超音波を受信し、この音波の発信から受信までの時間を計測することで対象物までの距離を計測



超音波センサの接続

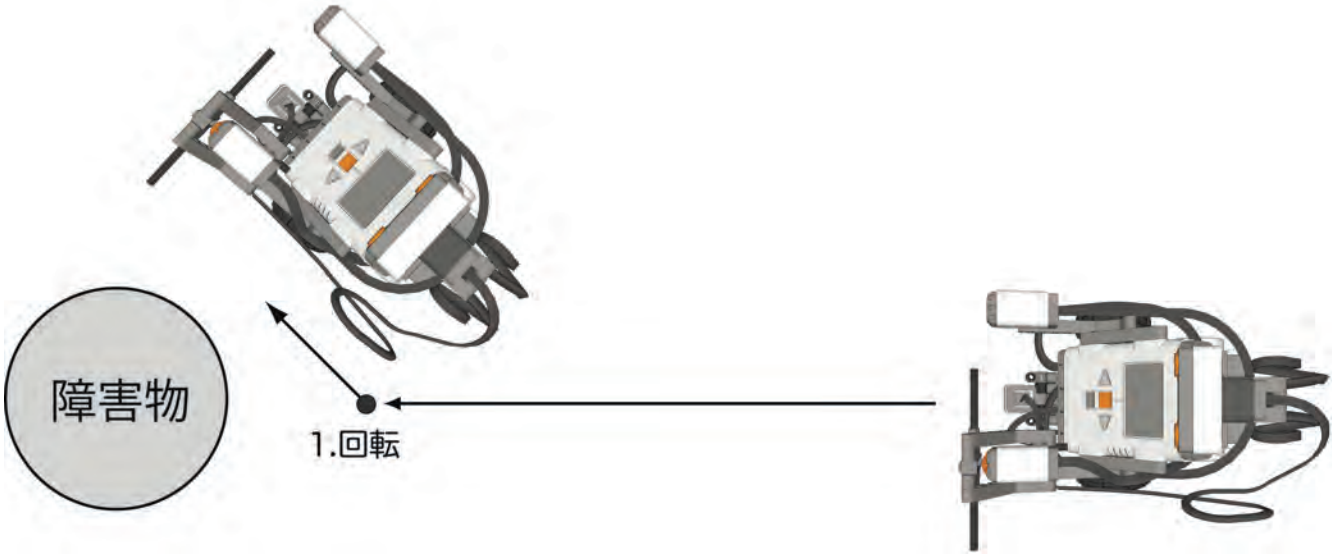
- NXTの入力ポート4に超音波センサを接続



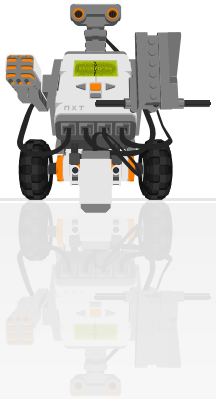


超音波センサによる障害物回避

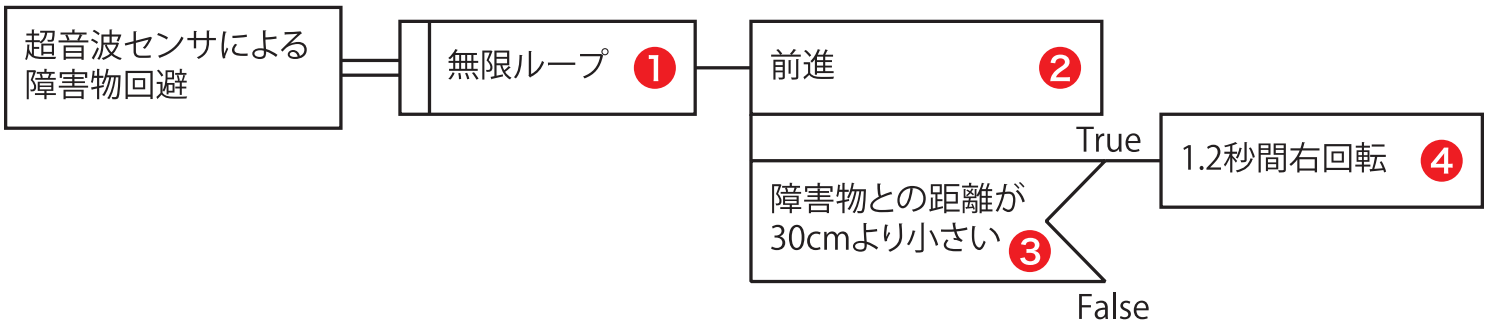
1. 常にロボットを前進　→　無限ループの利用
2. 障害物との距離が30cmより小さいとき，右回転，その後
    1. に戻る

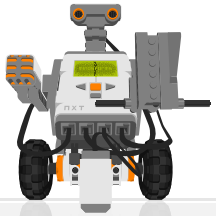


→後退する動作を必要としない



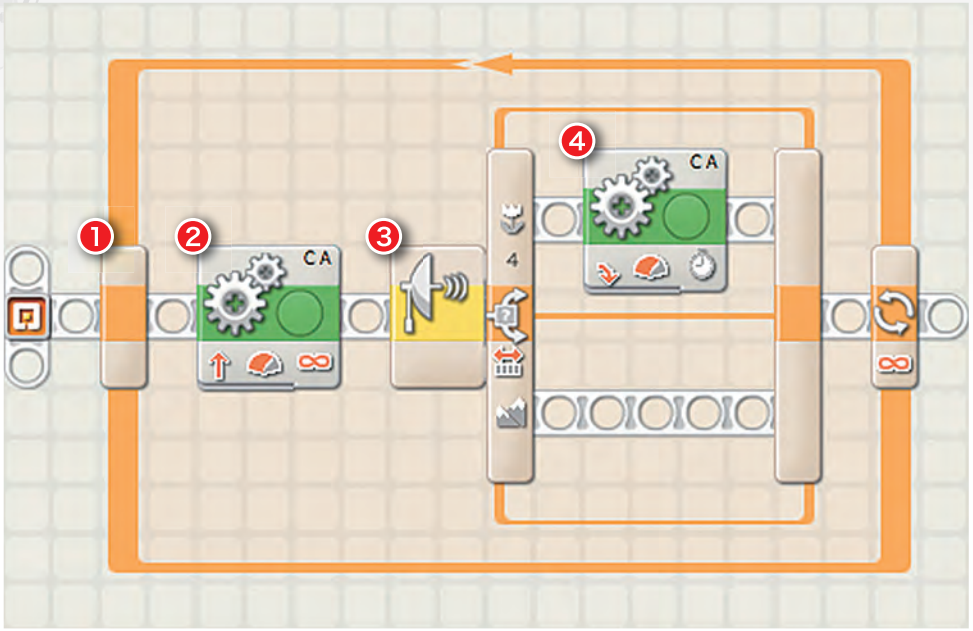
超音波センサによる障害物回避のPAD





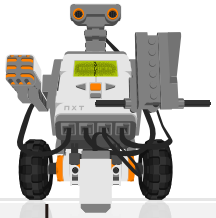
超音波センサによる障害物回避 (p.56: usonic.nxc)

usonic.rbt



無限ループ

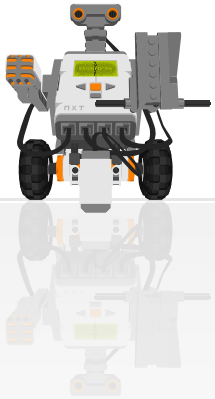
ポート A , C のモータを
一瞬だけ順方向に回転し
て次のブロックへ移る



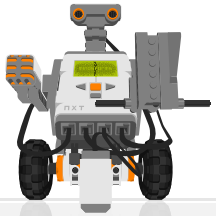
超音波センサによる障害物回避 (p.56: usonic.nxc)

ポート 4 の超音波センサで測った値が 30cm より小さいかどうか

1.2 秒間右回転

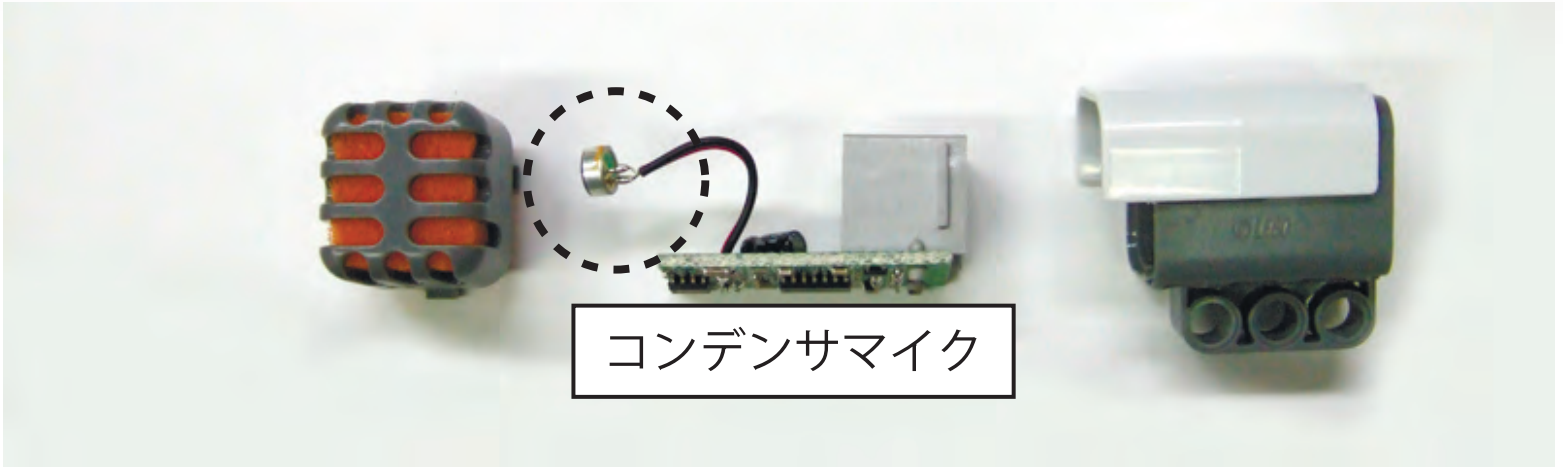


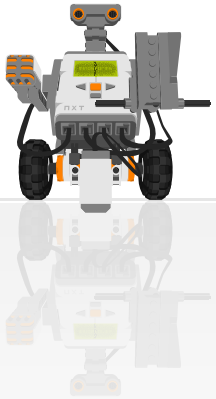
■サウンドセンサによる制御



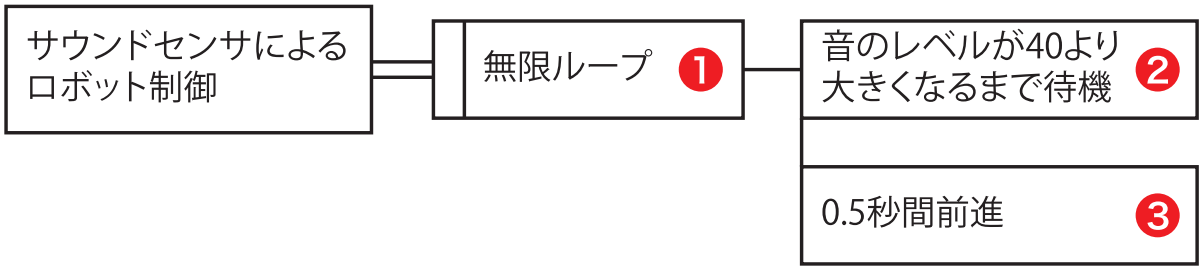
サウンドセンサ

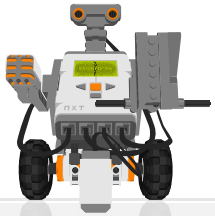
- サウンドセンサの出力
  - 音のない状態を0として0～100までの値





サウンドセンサによる制御のPAD





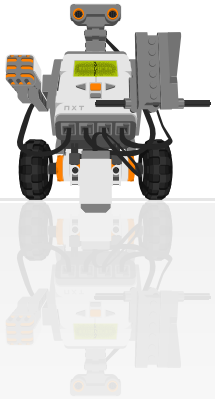
サウンドセンサによる制御 (p.58: mic.nxc)

mic.rbt

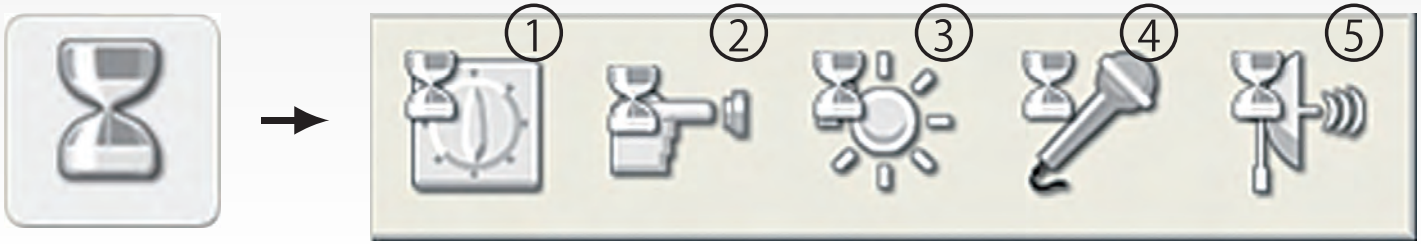
無限ループ

サウンドセンサの読み取り値が 40 をこえるまでなにもしない．読み取り値が 40 を超えたら③を実行.

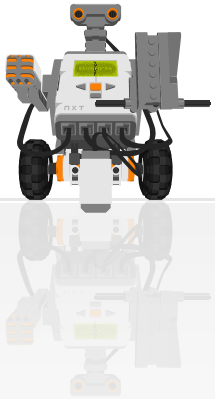
0.5 秒間前進



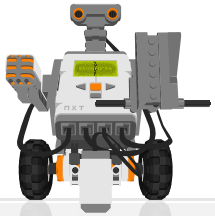
待機ブロックの種類



① 時間ブロック　：設定した時間だけ待機
② タッチブロック：設定した条件をタッチセンサが満たすまで待機
③ 照明ブロック　：設定した条件をライトセンサが満たすまで待機
④ 音ブロック　　：設定した条件をサウンドセンサが満たすまで待機
⑤ 距離ブロック　：設定した条件を超音波センサが満たすまで待機

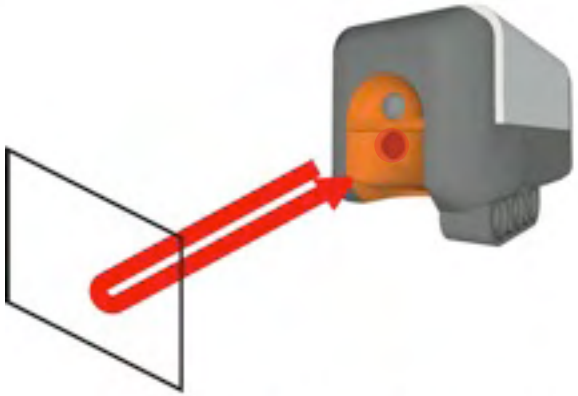


■ライントレース

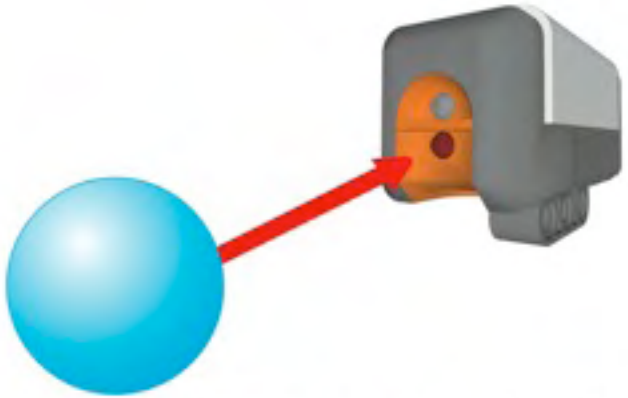


ライトセンサ

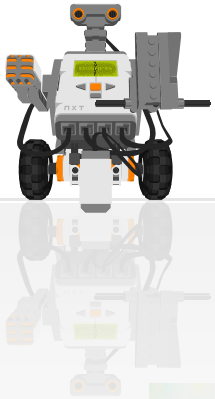
- 赤色LEDの反射光の量を数値で表す



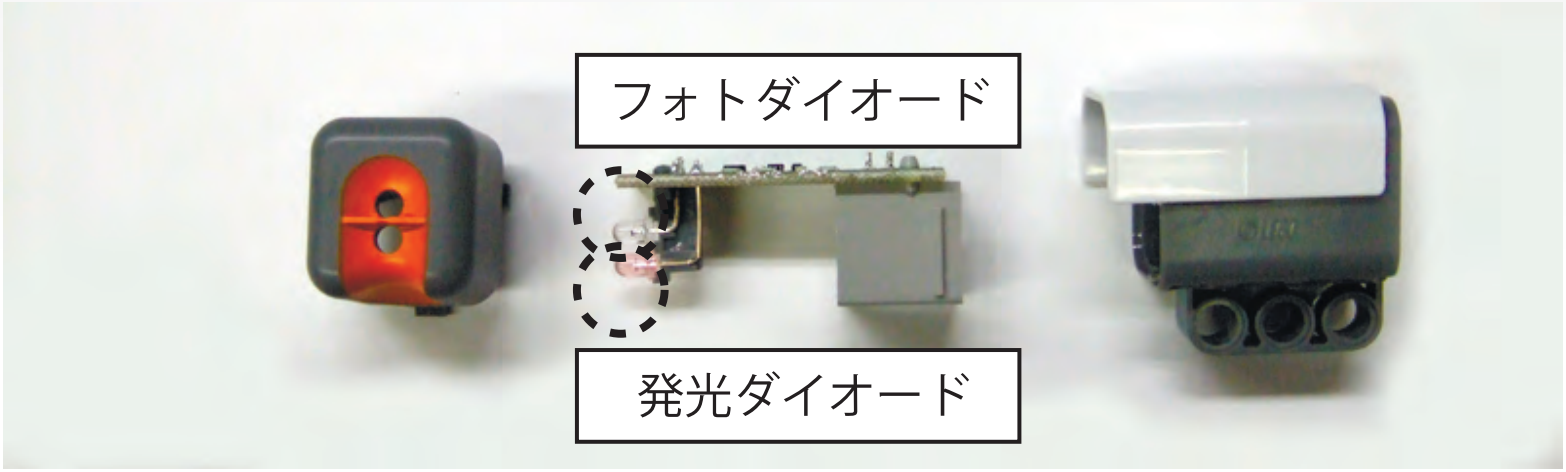
(a) 反射の光量を読みとる



(b) 発行球の光量をよみとる

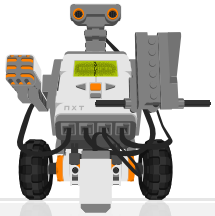


ライトセンサ



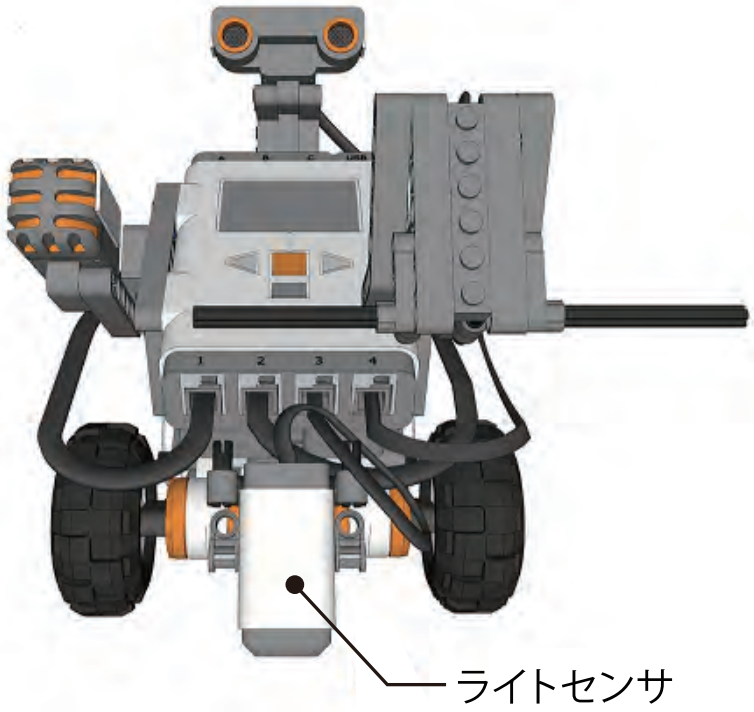
ライトセンサの原理

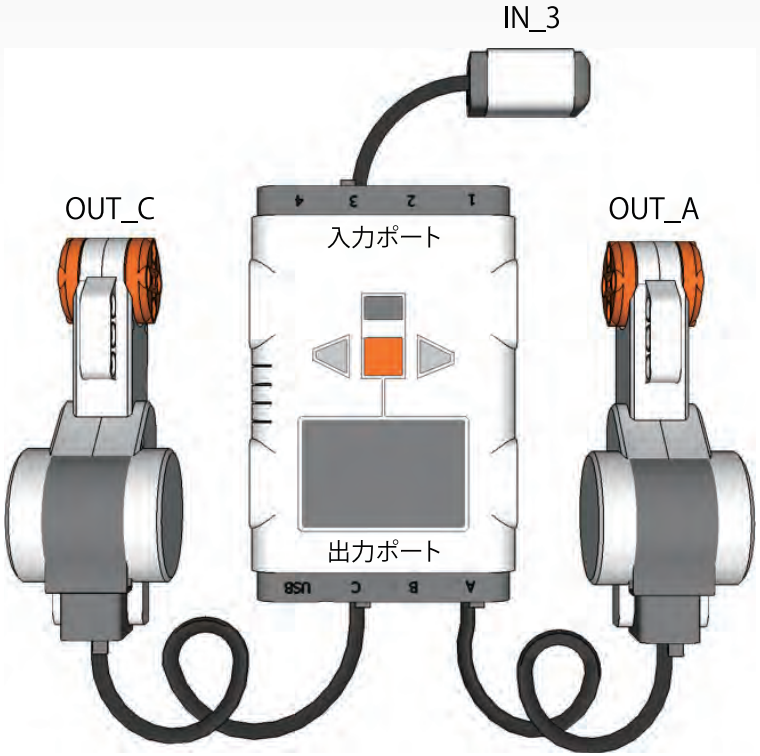
発光ダイオードの光が対象物にぶつかり反射する。その反射の光量をフォトダイオードが検知し計測する

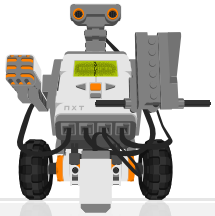


ライトセンサの接続

・ NXTの入力ポート3にライトセンサを接続

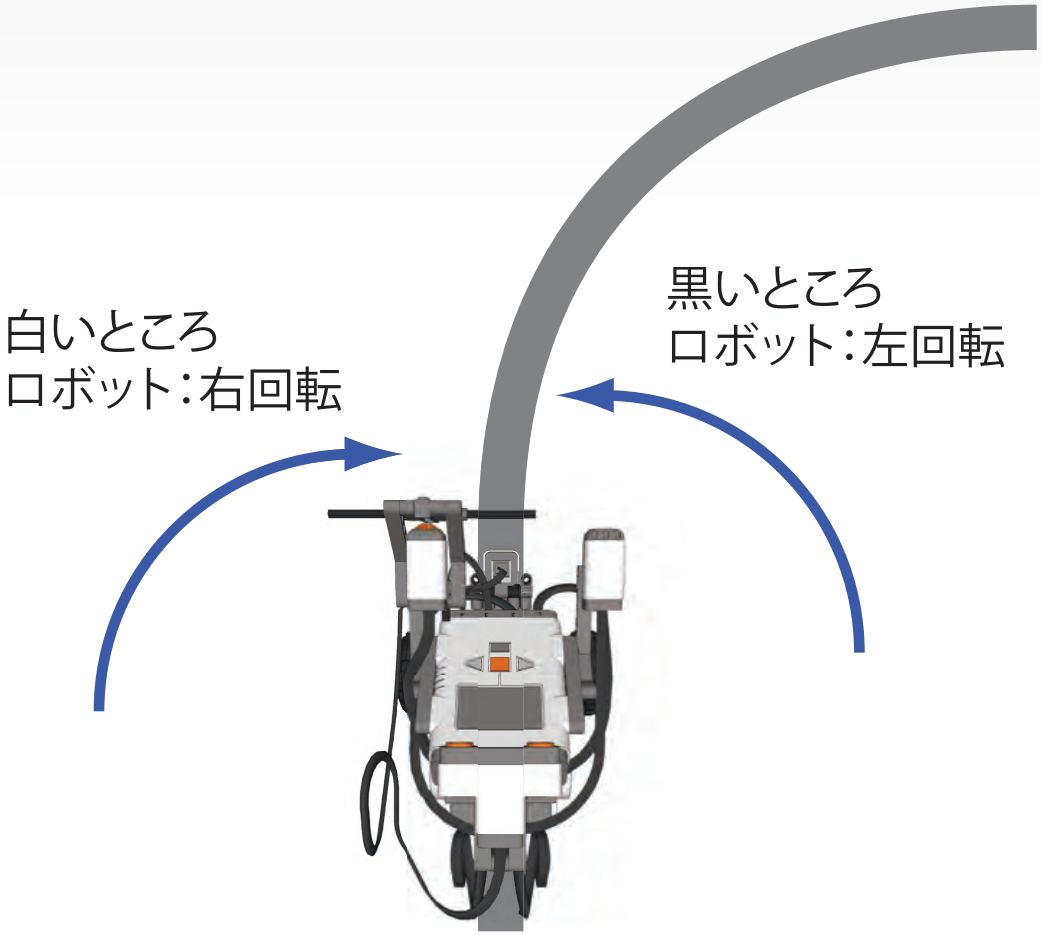


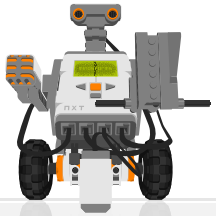




ライントレースの考え方

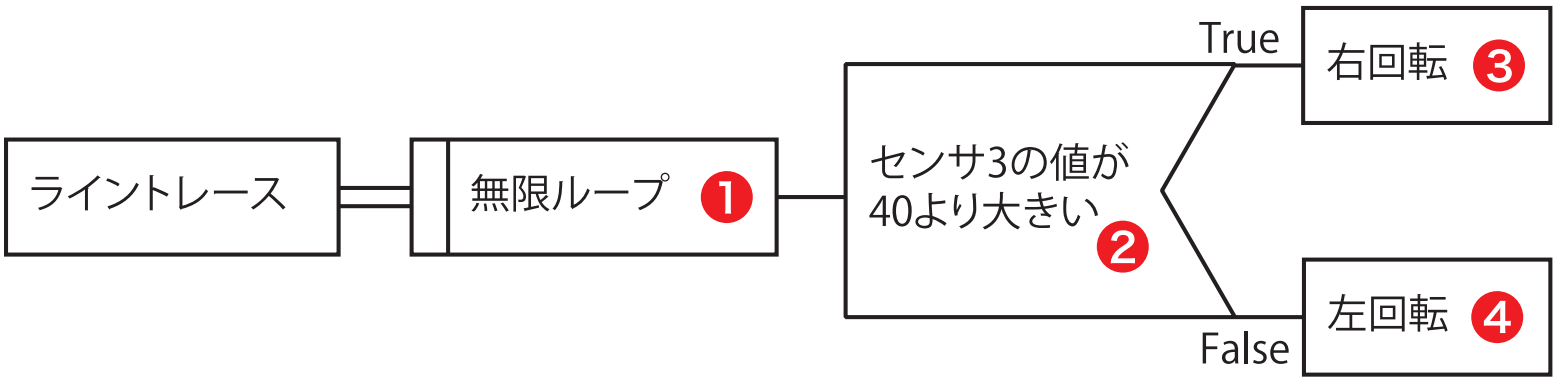
- 白(明るい)ところでは右回転，黒(暗い)ところでは左回転

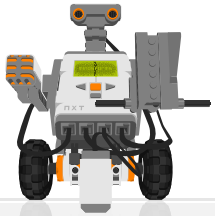




ラインとレースのPAD

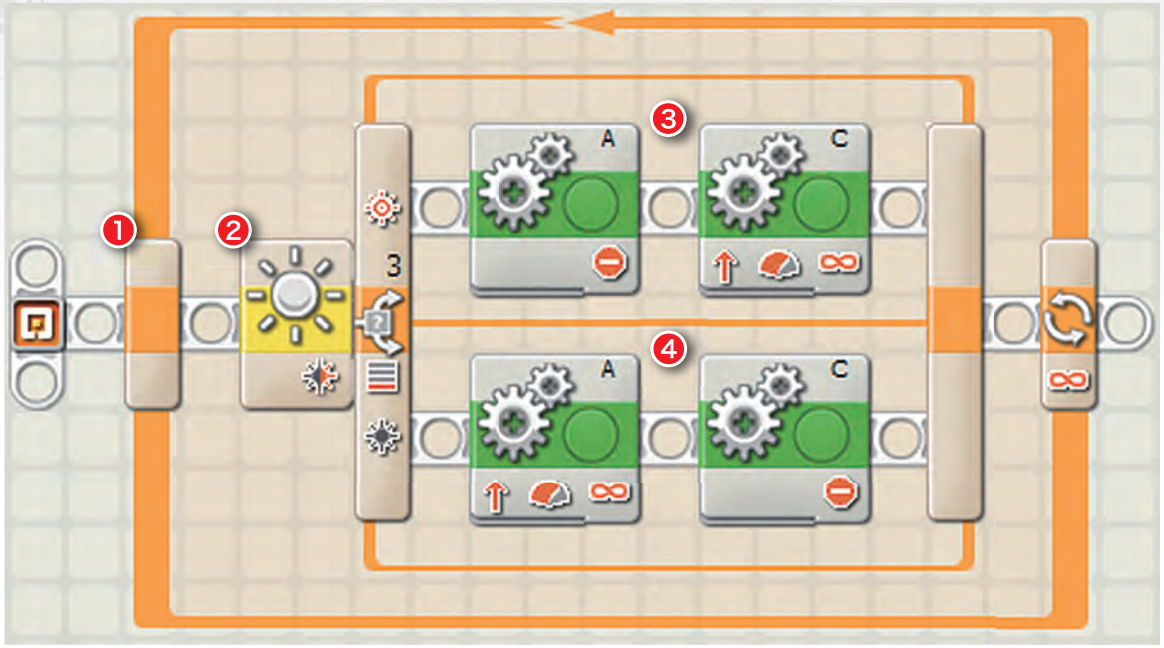
- 白(明るい)ところでは右回転，黒(暗い)ところでは左回転をするプログラムのアルゴリズム





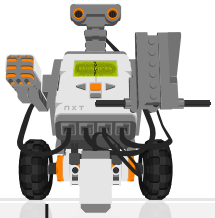
ライトセンサによるラインレース (p.62: line-tracer.rbt)

line_tracer.rbt



無限ループ

ポート 3 のライトセンサで床の明るさを測り，明るさが 40 より大きいか小さいかで動作を分岐



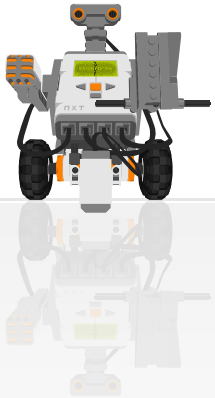
ライトセンサによるライントレース (p.62: line-tracer.rbt)

床の明るさが 40 より大きいとき，モータ A を停止し，モータ C のみ回転して右回転

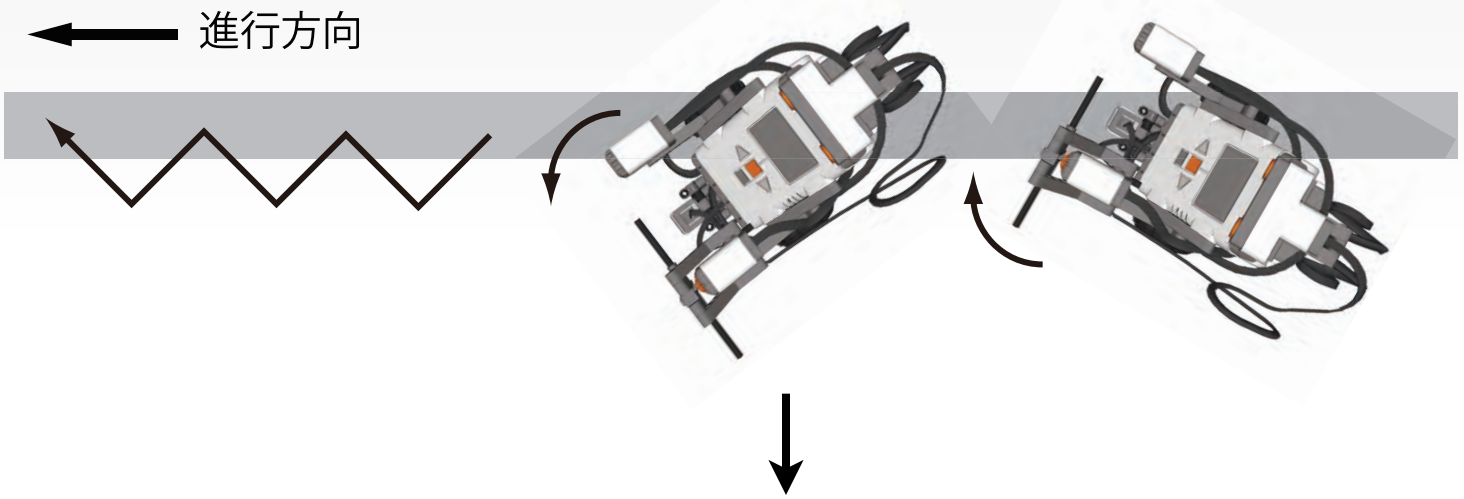
3

4

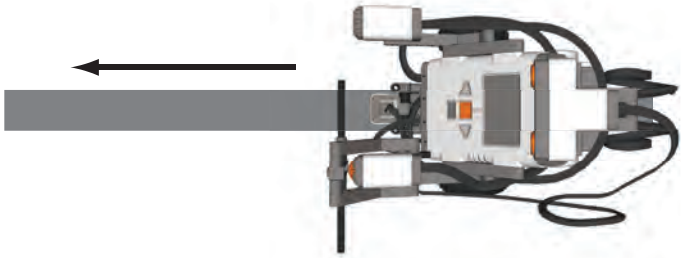
床の明るさが 40 以下のとき，モータ A のみ回転し，モータ C を停止して左回転



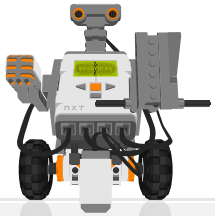
ライントレースの改良



① 黒のライン上
→ 前進

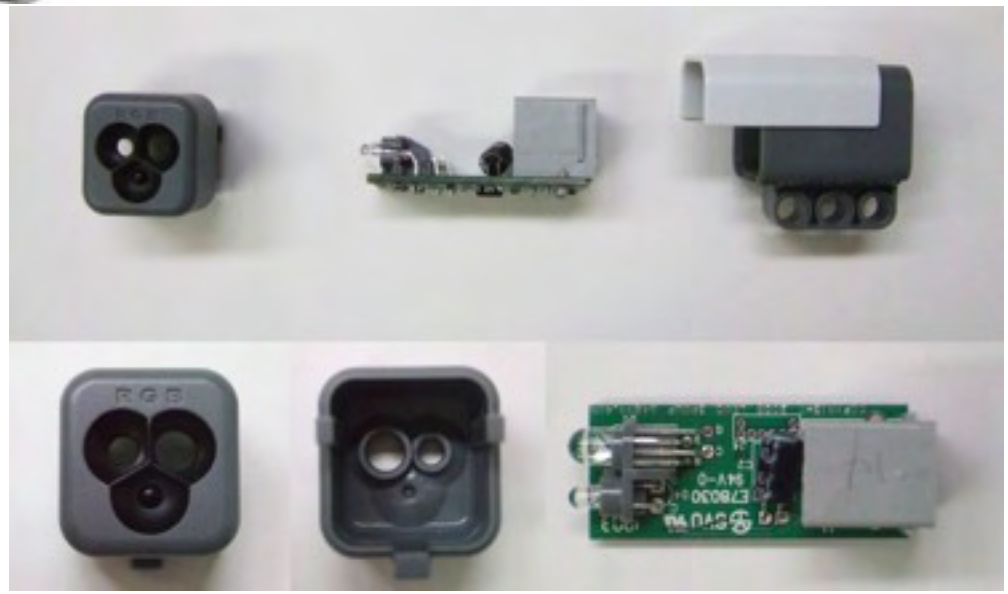


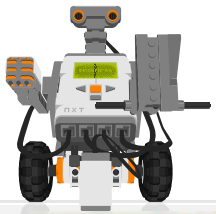
② 白の場合
→ 左右に回転して
黒のラインを探索



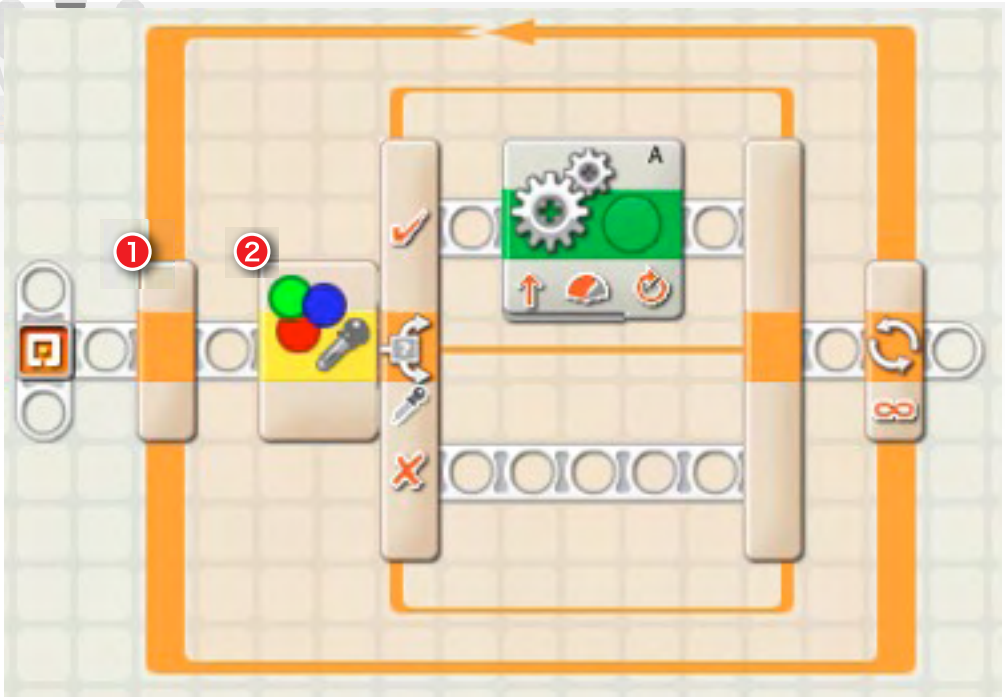
カラーセンサ（LEGO社）

・ 赤・青・緑の照明を照射して、反射光を読み取る



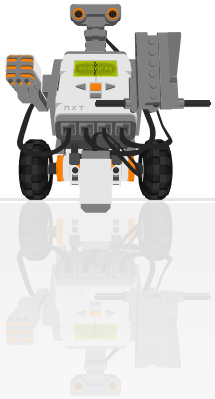


カラーセンサ（LEGO）によるモータ制御

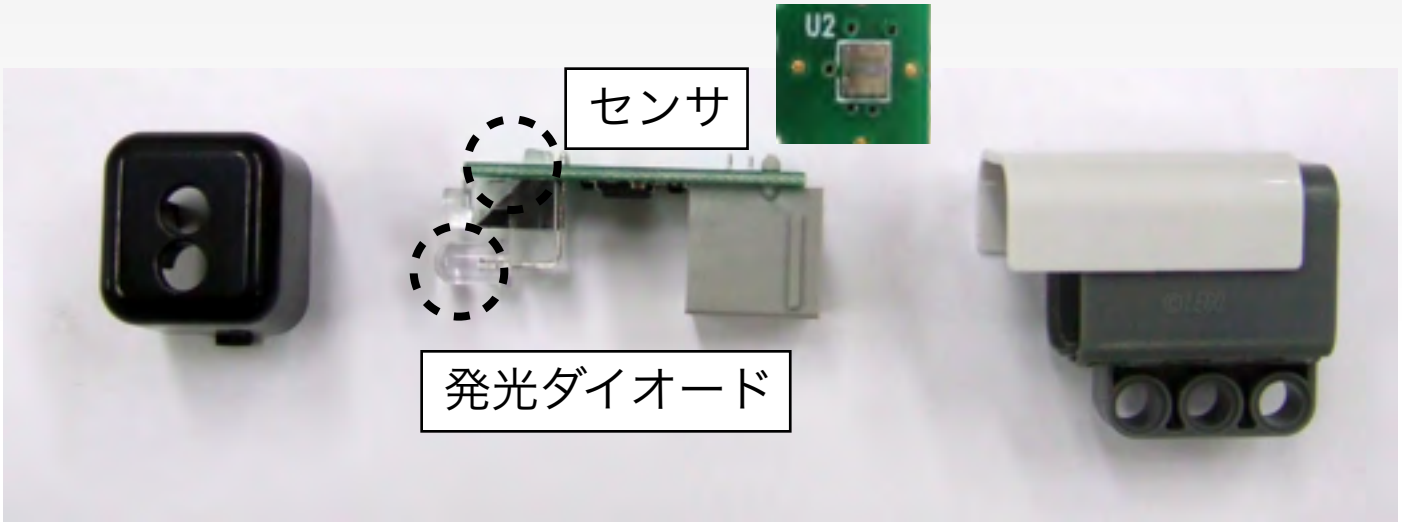


無限ループ

ポート3のカラーセンサで
黄色を判断して分岐

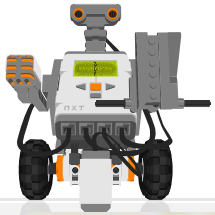


カラーセンサ（HiTechnic社）

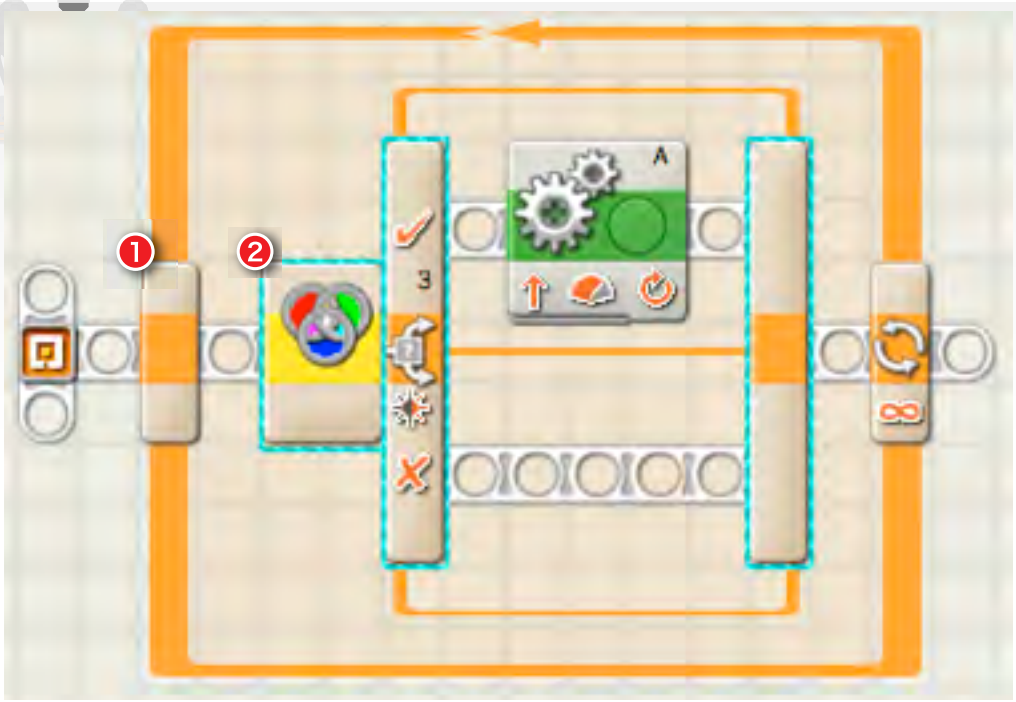


カラーセンサの原理

発光ダイオードの光が対象物にぶつかり反射する。その反射の光をセンサが検知し計測する



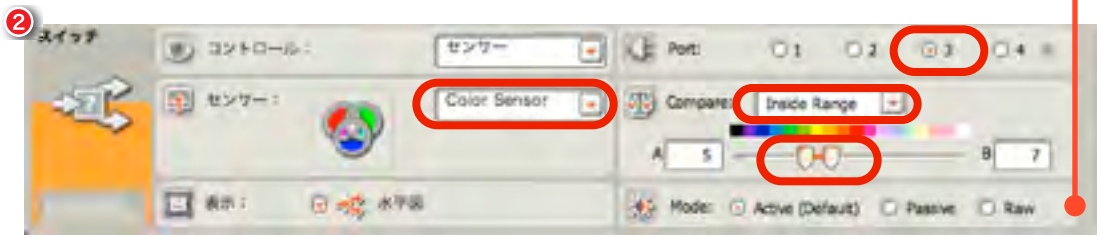
カラーセンサ（HT）によるモータ制御

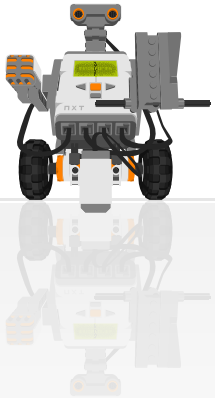


Mode: Passive(光源を使用0-17)
Passive(光源を使用しない0-17)
Raw(光源を使用、読み取り値)

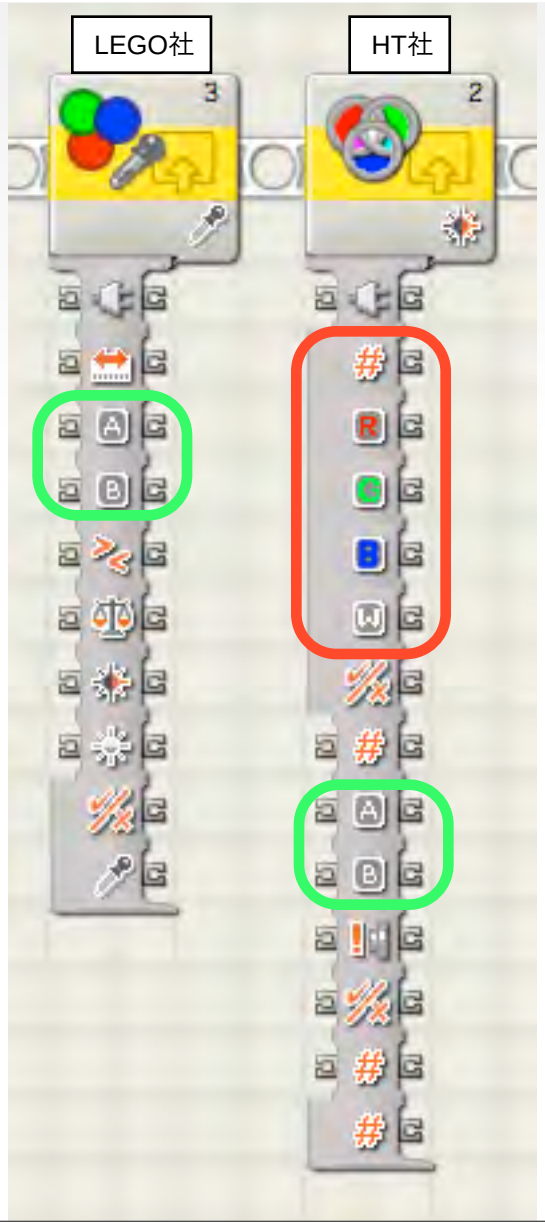
無限ループ

ポート3のカラーセンサで
黄色を判断して分岐





カラーセンサ比較



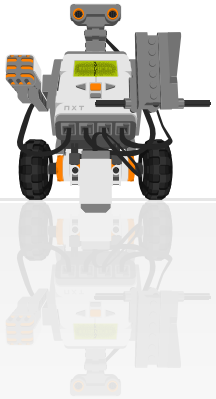
色の値（0-17）

赤色の値

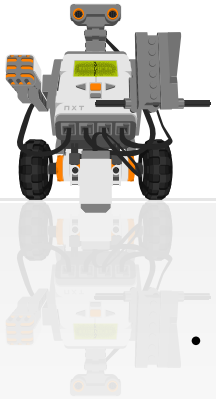
緑色の値

青色の値

白色の値

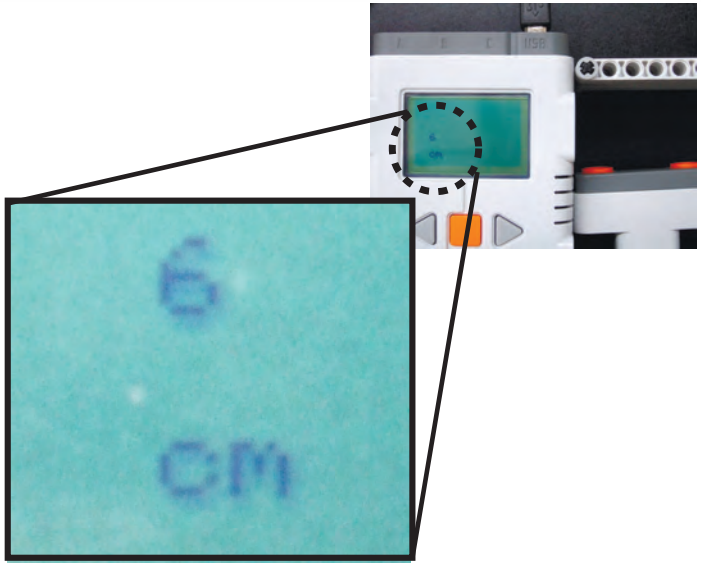


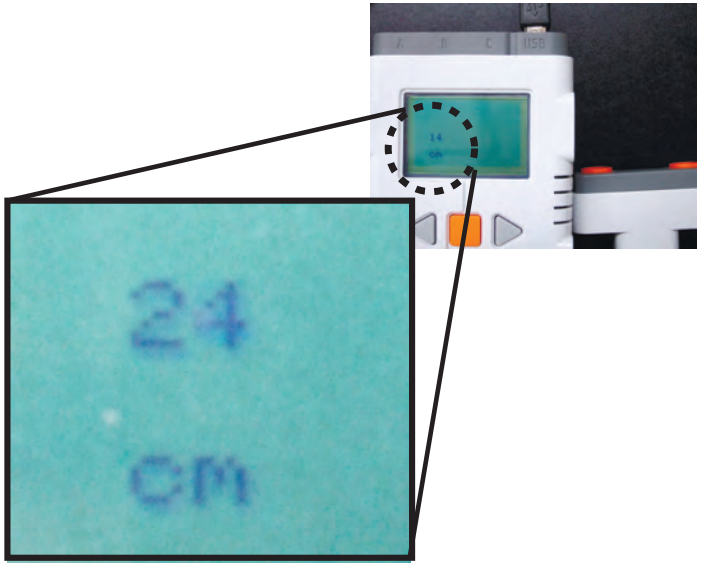
■ディスプレイ表示

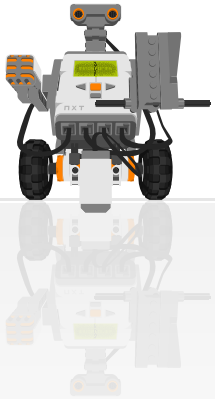


テキストの表示

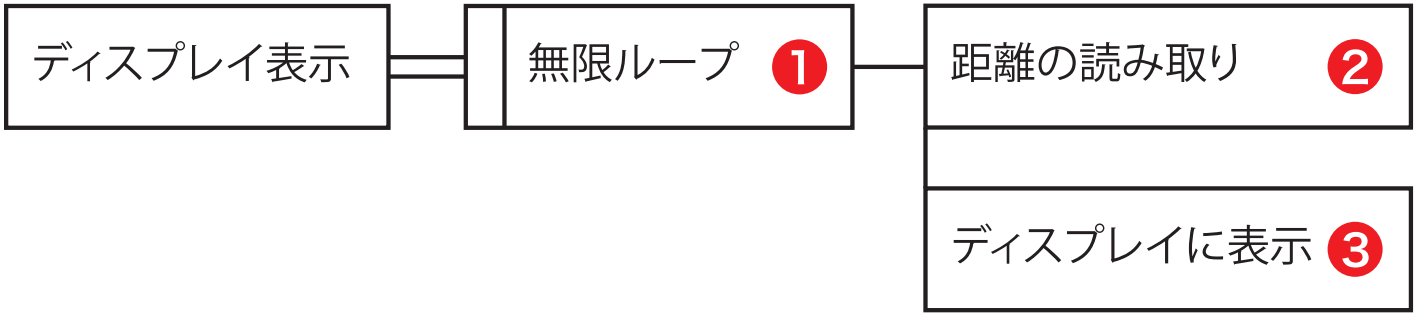
- 超音波センサの値を液晶ディスプレイに表示
  - デバッグに重要

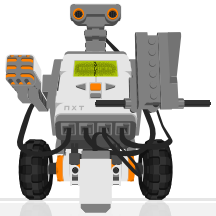






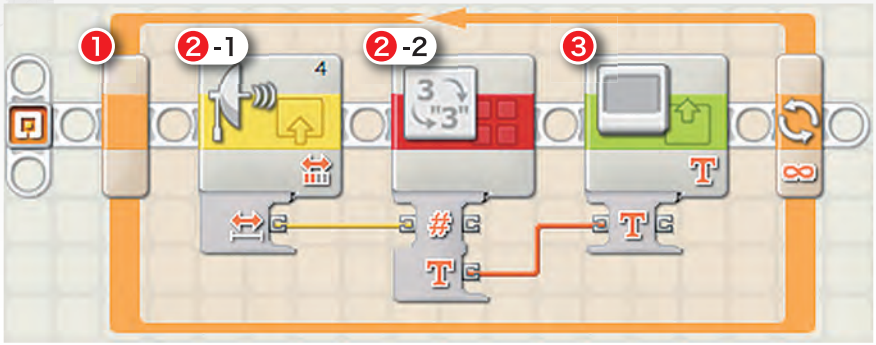
超音波センサ値を表示





超音波センサ値を表示 (p.67: display-text.rbt)

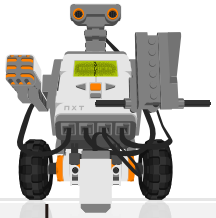
display_text.rbt



無限ループ

ポート 4 の超音波センサで前方の物体までの距離を測定

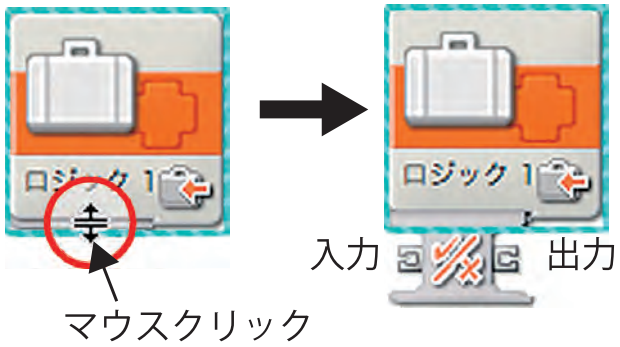
超音波センサで測った数値をテキストに変換



超音波センサ値を表示 (p.67: display-text.rbt)

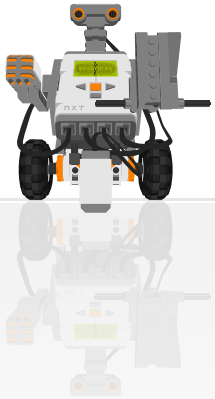
テキストを表示
（データワイヤで受け取ったテキストを表示するので入力欄の文字は無視される.）

データワイヤとデータハブ



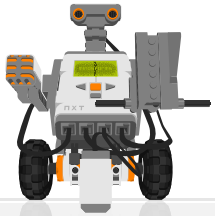
データワイヤの色と意味

黄	：数値
緑	：ロジック（正偽値）
オレンジ	：テキスト（文字）



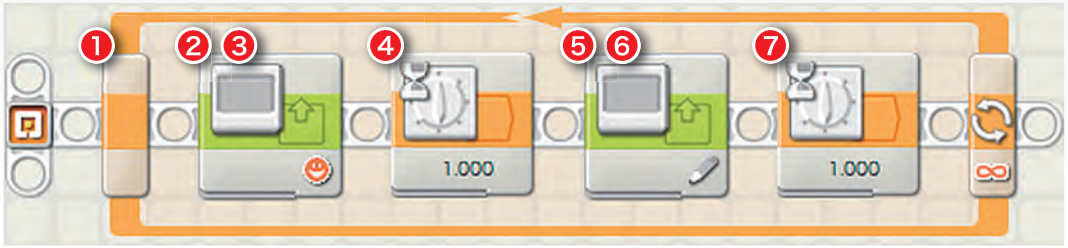
図形の表示

- ディスプレイ表示
  - 無限ループ ❶
    - ディスプレイのクリア ❷
    - 画像をディスプレイに1秒表示 ❸ ❹
    - ディスプレイのクリア ❺
    - 図形をディスプレイに1秒表示 ❻ ❼



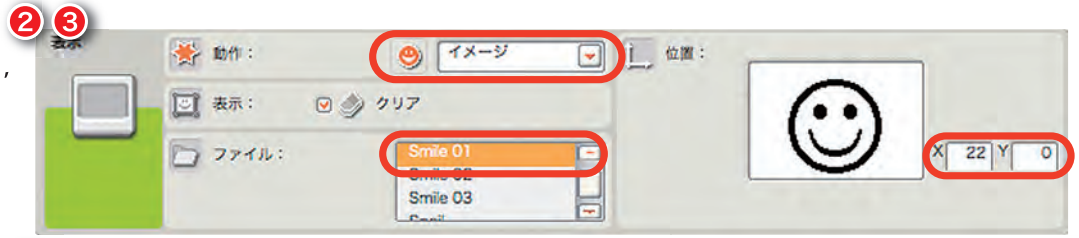
図形を表示 (p.70: display_fig.rbt)

display_fig.rbt

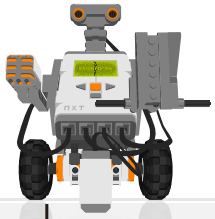


無限ループ

現在の画面の表示状態をクリアして，
NXT-SW に標準で用意されている画像ファイルを表示



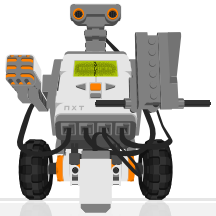
表示状態を 1 秒間保持



図形を表示 (p.70: display_fig.rbt)

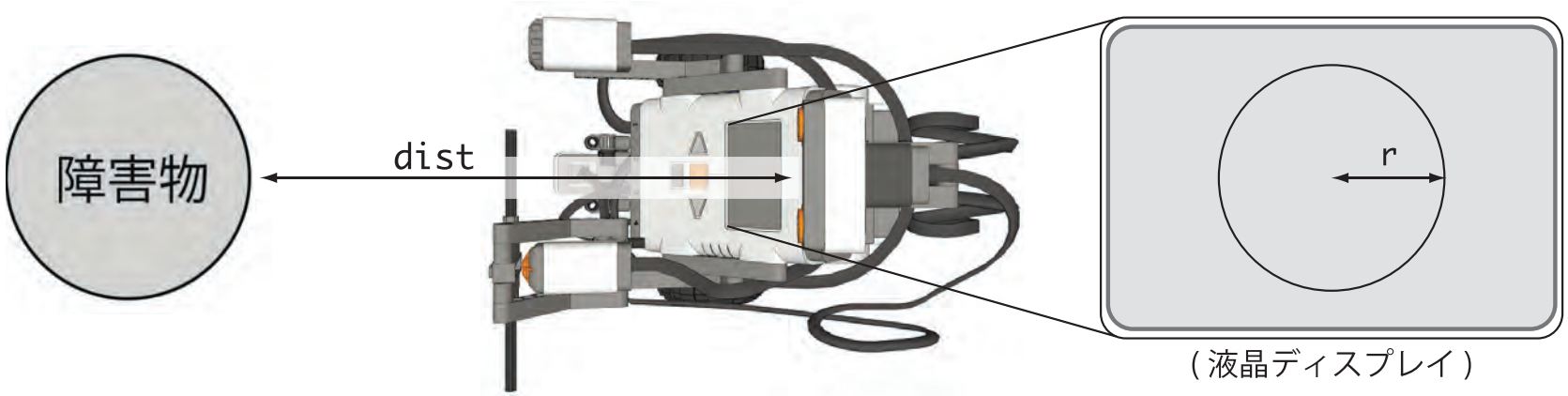
現在の画面の表示状態をクリアして，円を表示 (X，Y で設定する値は表示する円の中心)

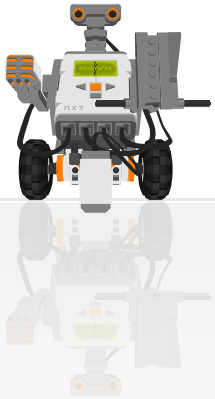
表示状態を 1 秒間保持



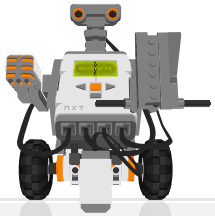
■■　演習問題5-1　■■

- 障害物までの距離が小さくなるほど大きくなる円を表示

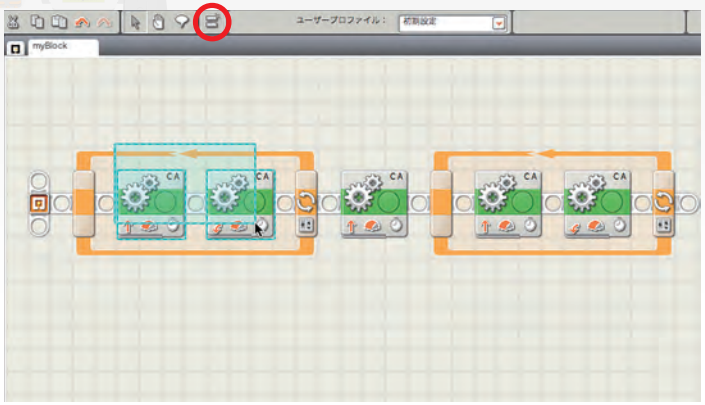




■マイブロック



マイブロック化(p.74)

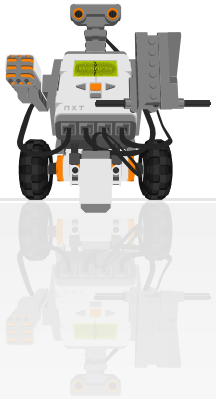


(a) ブロックを選択

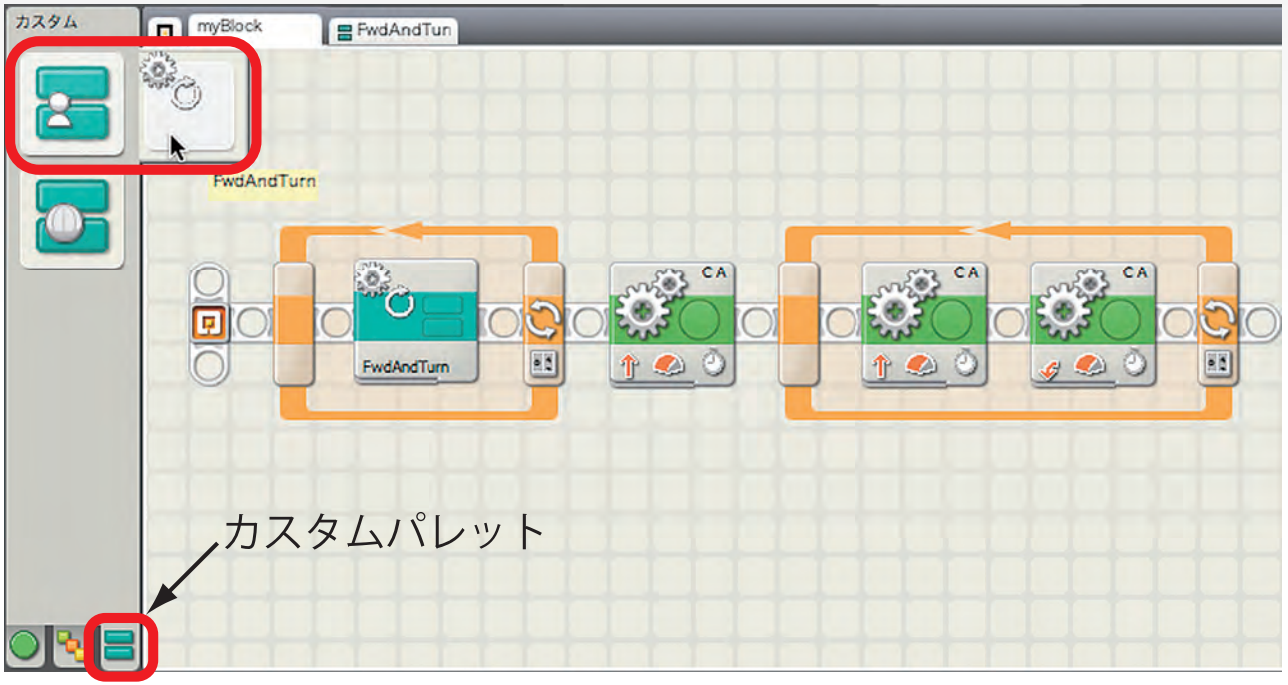
(b) マイブロック・ビルダー

(c) アイコンの作成

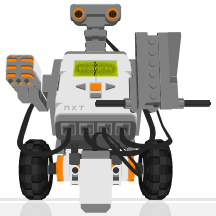
①マイブロック化したい複数のブロックをドラッグにより選択
②”マイブロック作成”というボタンを押す
③マイブロックの名前やアイコンを決定



マイブロックの利用(p.74)

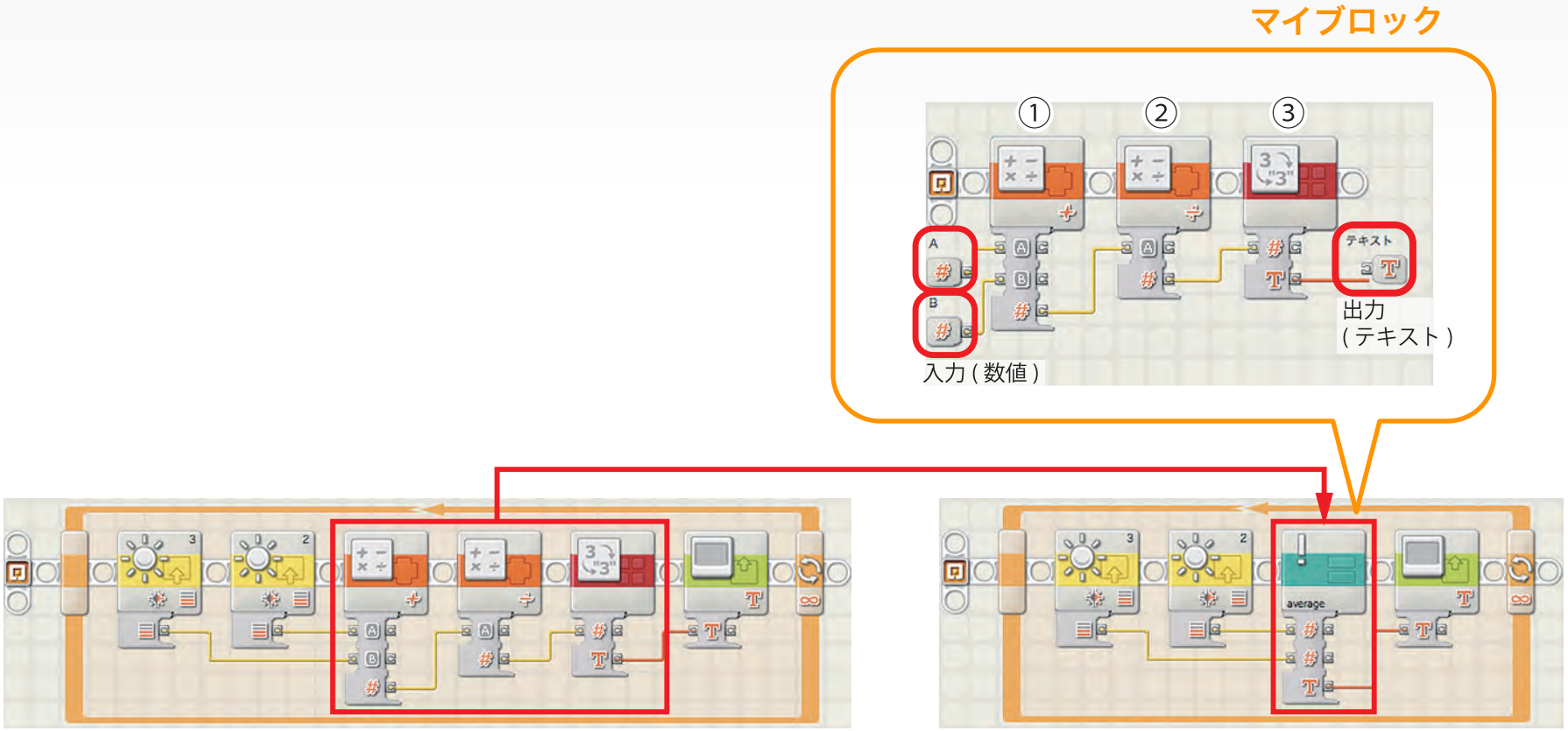


カスタムパレットからマイブロックをワークエリアにドロップ



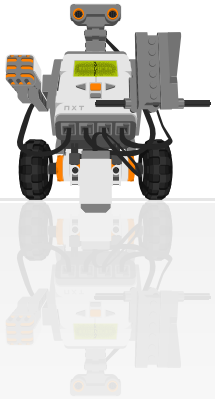
マイブロック間のデータのやりとり(p.75)

マイブロックへの入力：ライトセンサ2とライトセンサ３の値
マイブロックの出力　：テキスト（平均値）

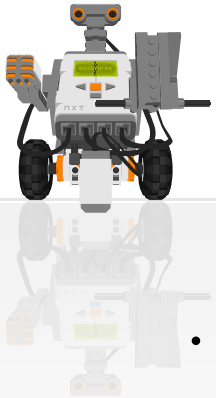


(a) 選択

(b) 入力と出力を持つマイブロック

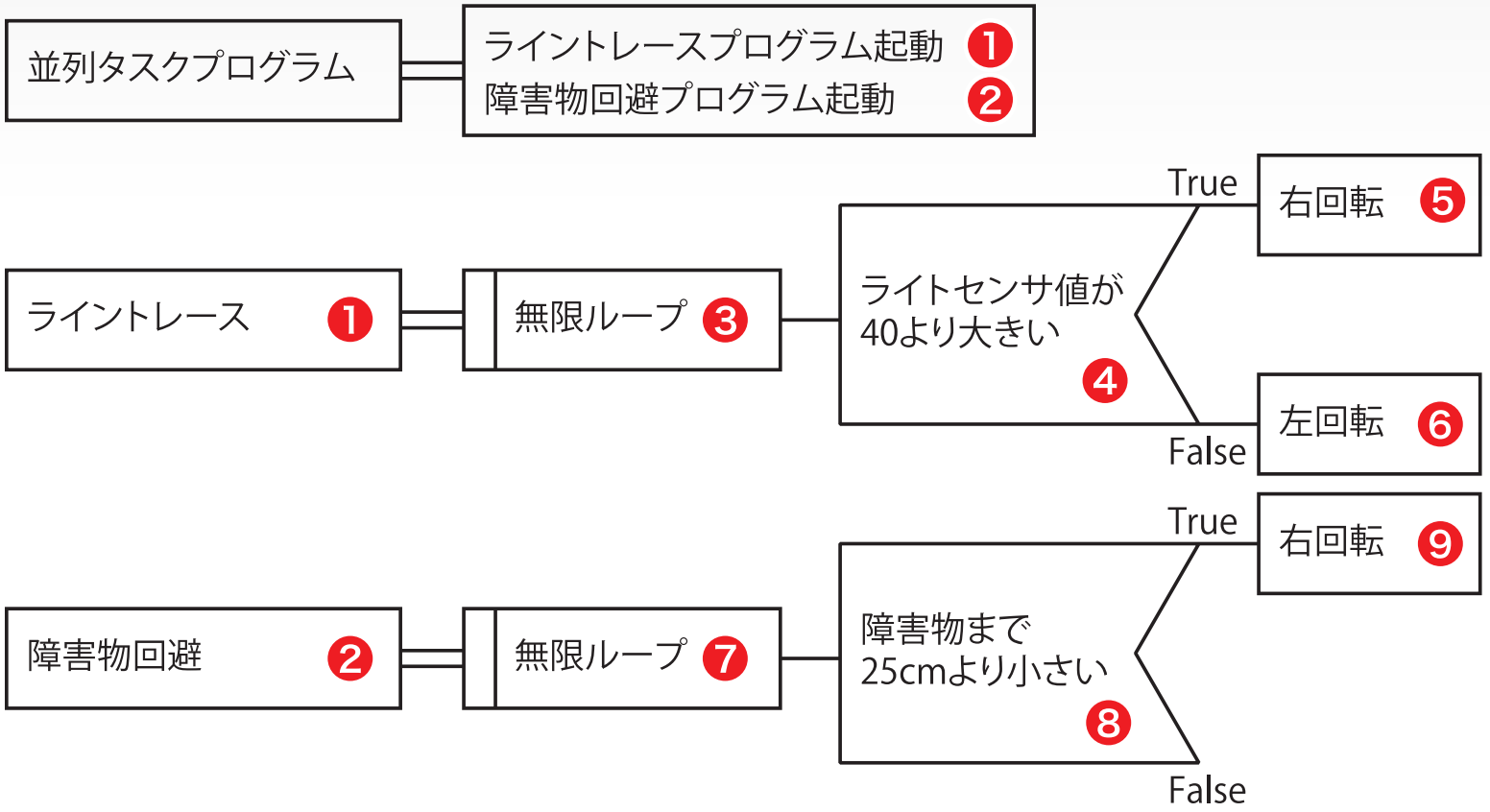


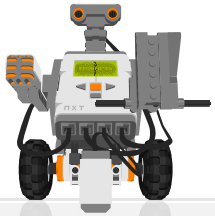
■並列タスク



PADによる並列タスクの図示

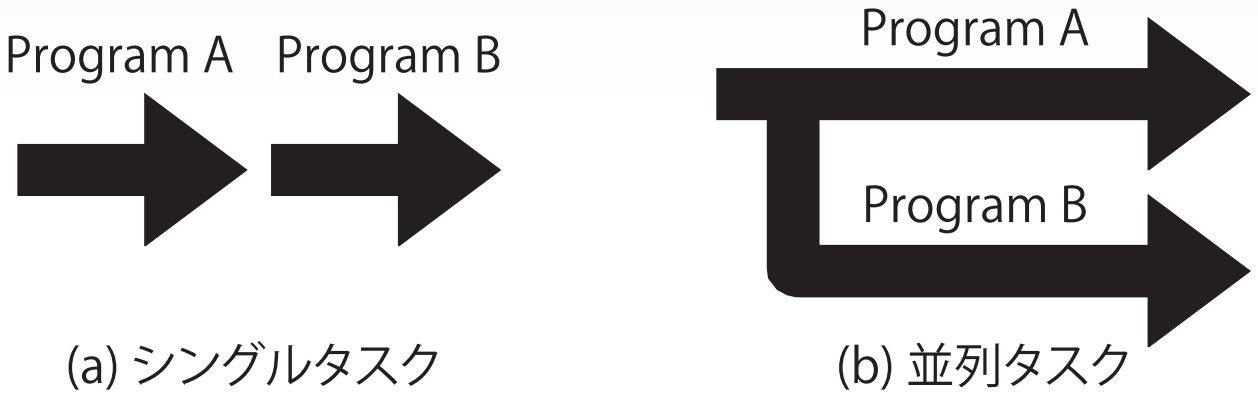
- 並列タスク





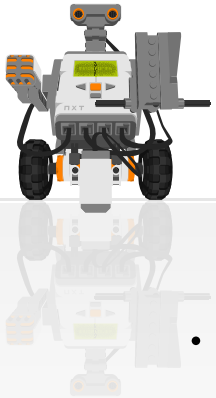
シングルタスク

- シングルタスクと並列タスク



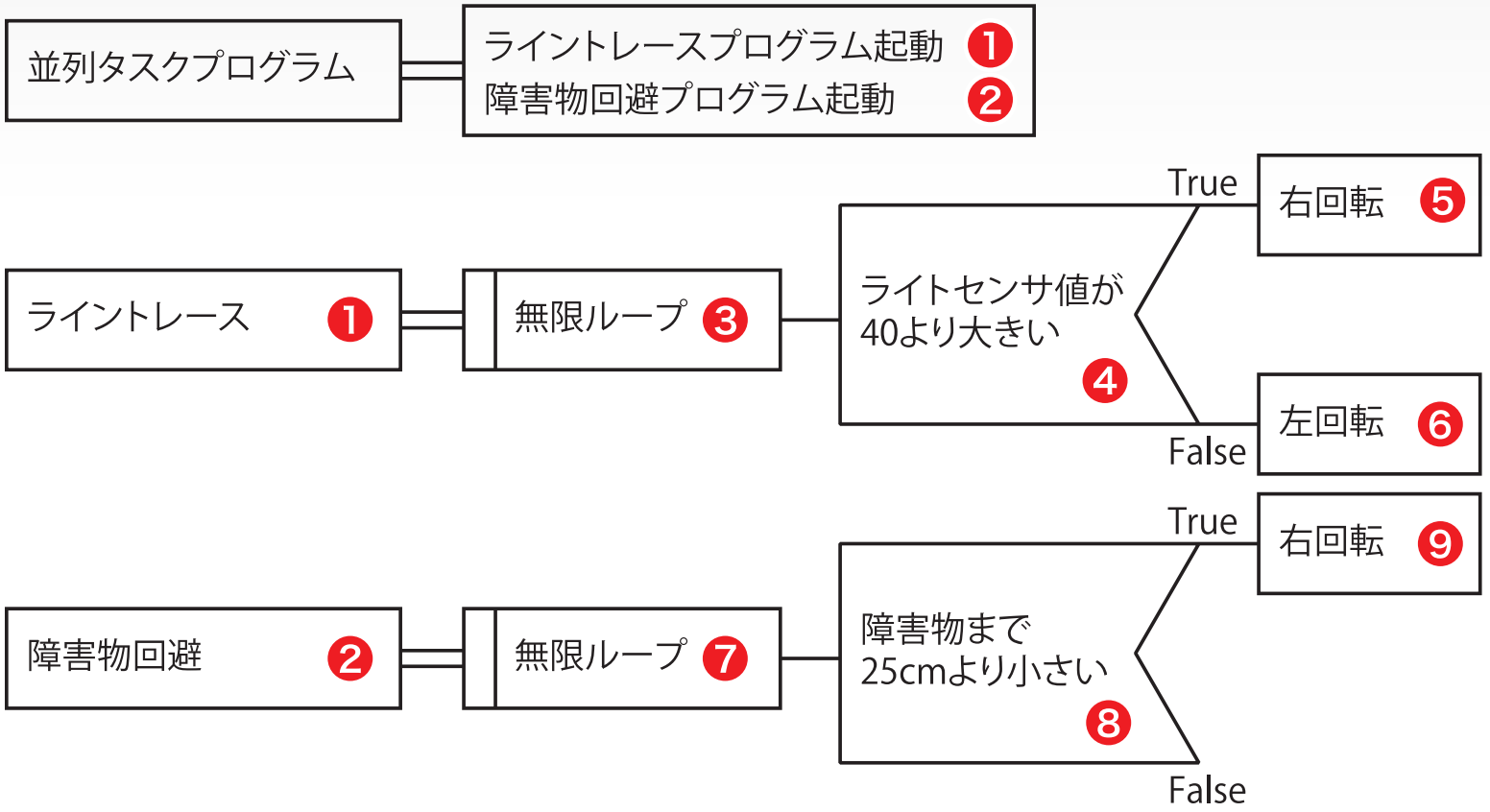
(a) シングルタスク　　(b) 並列タスク

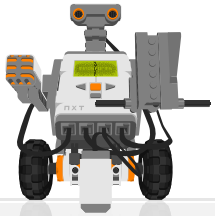
→ライントレースと障害物回避を同時に行うには並列タスク化する



PADによる並列タスクの図示

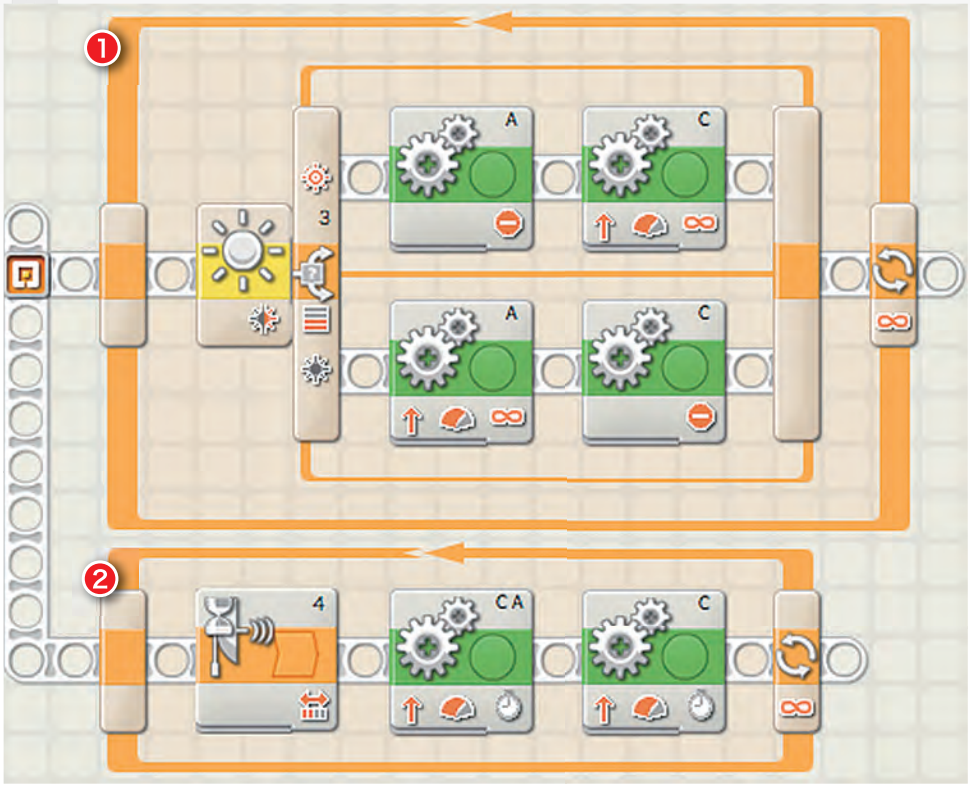
- 並列タスク



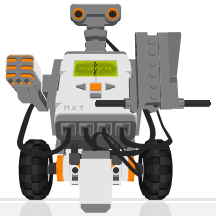


並列タスク (p.84: parallel_task1.rbt)

parallel_task1.rbt

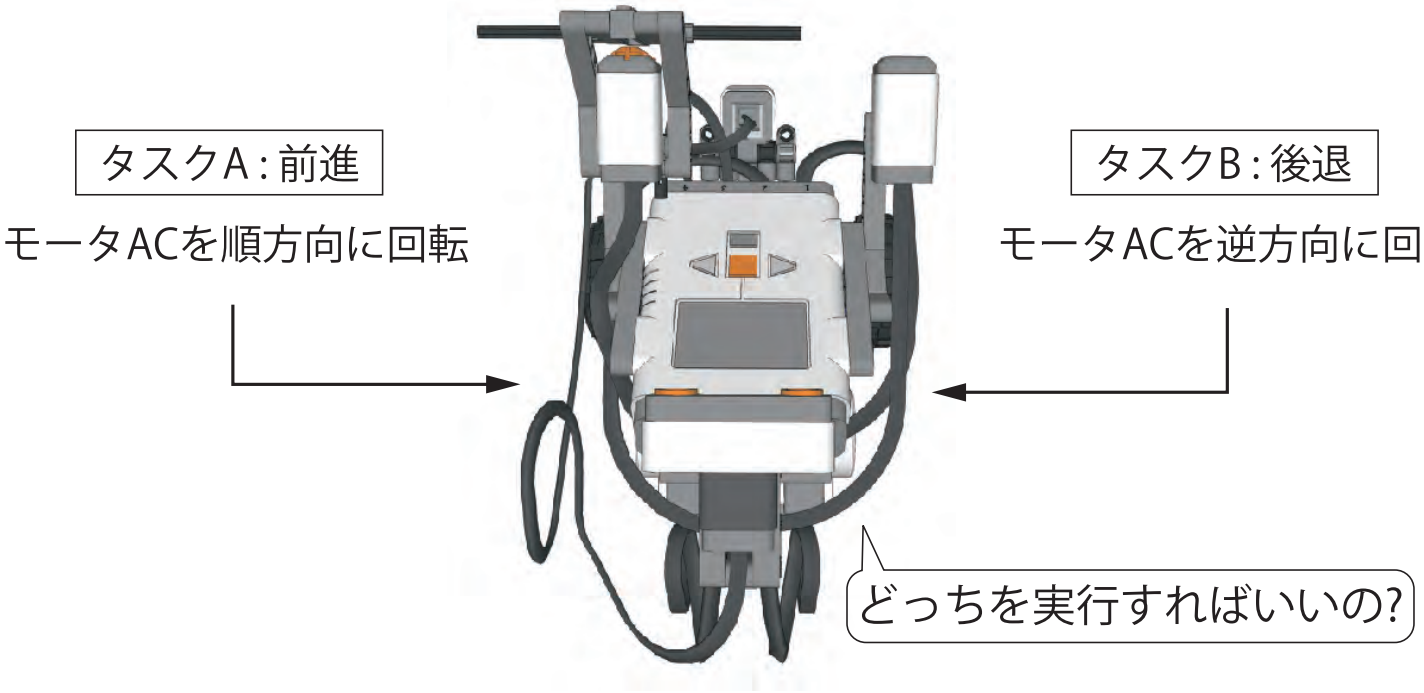


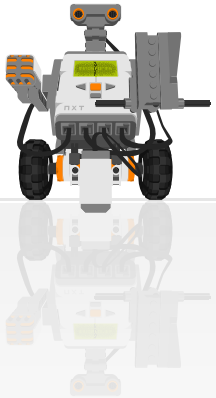
プログラムの並列化：シーケンスビームを並行に並べる



並列タスクにおける問題

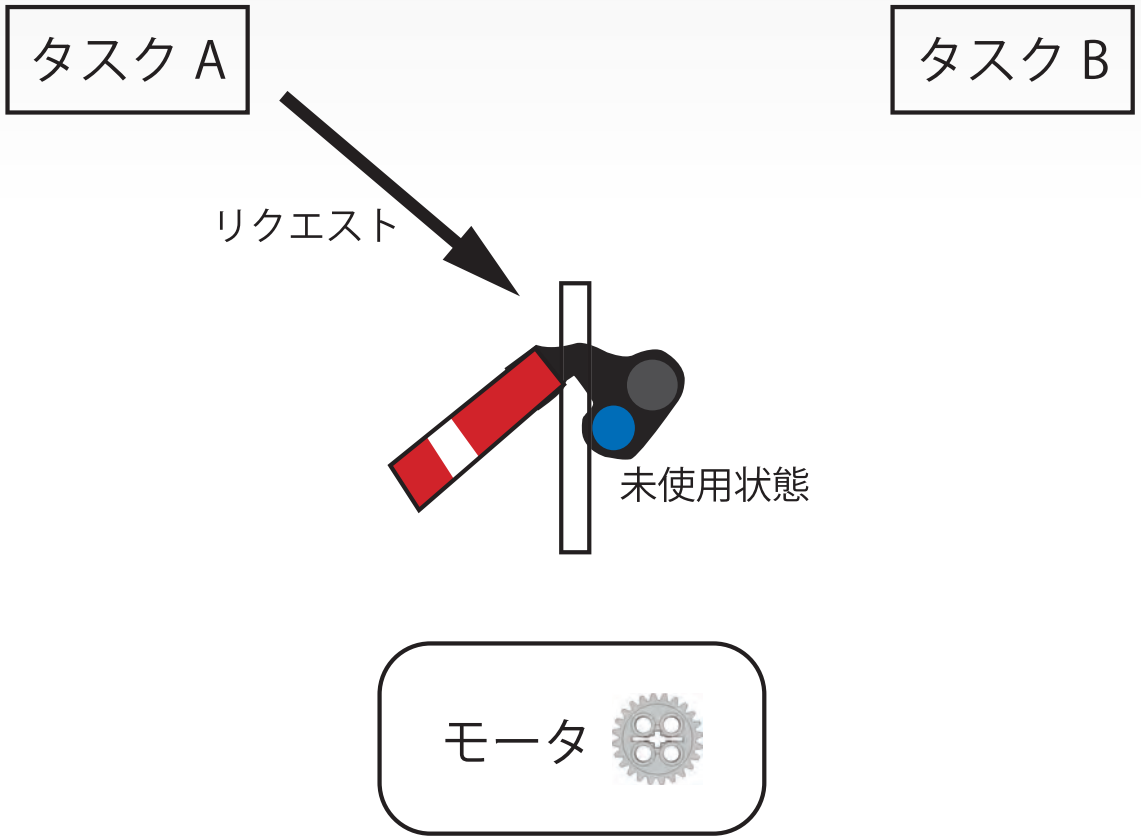
- 命令の衝突（コンフリクト）
  - タスクAがタスクBの実行を妨害
  - 同時にモータを制御

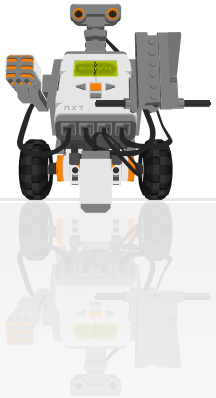




セマフォ（信号灯）

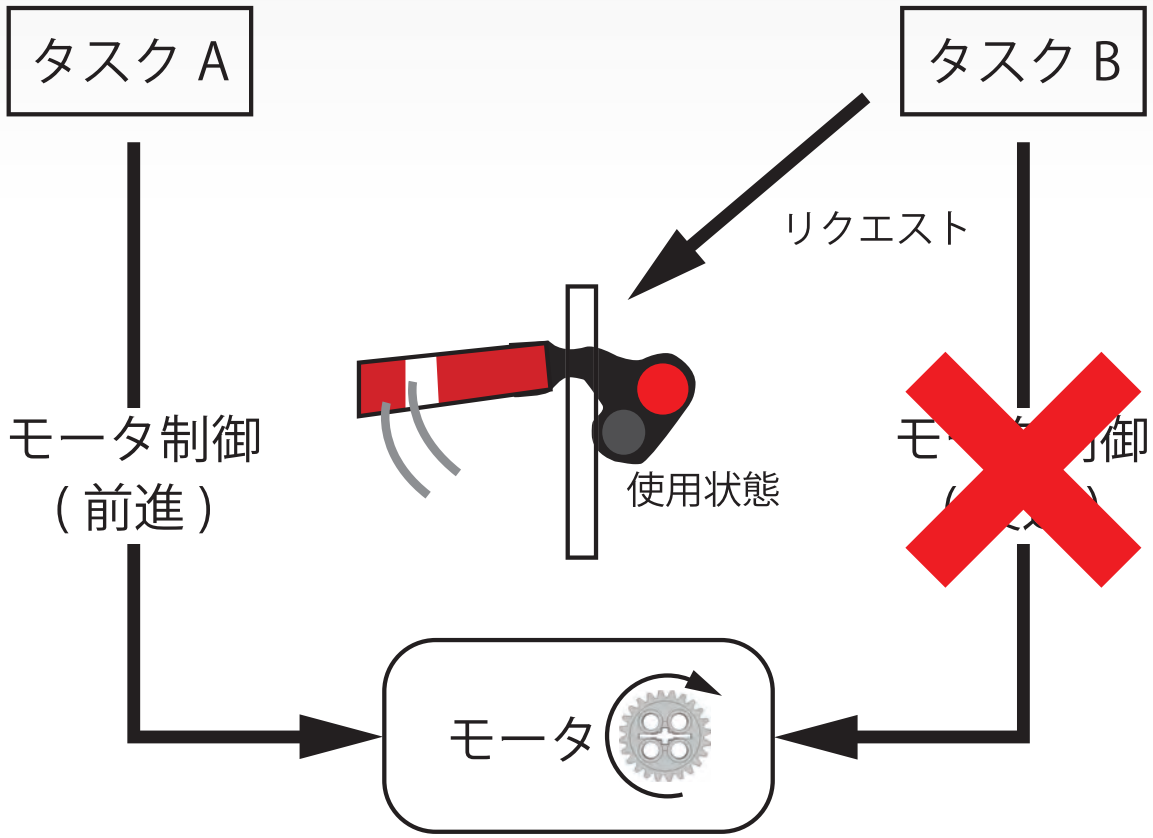
① モータの未使用状態

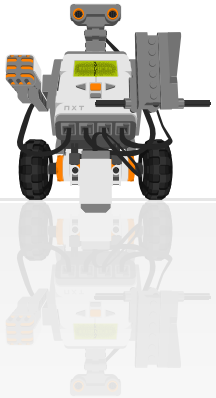




セマフォ（信号灯）

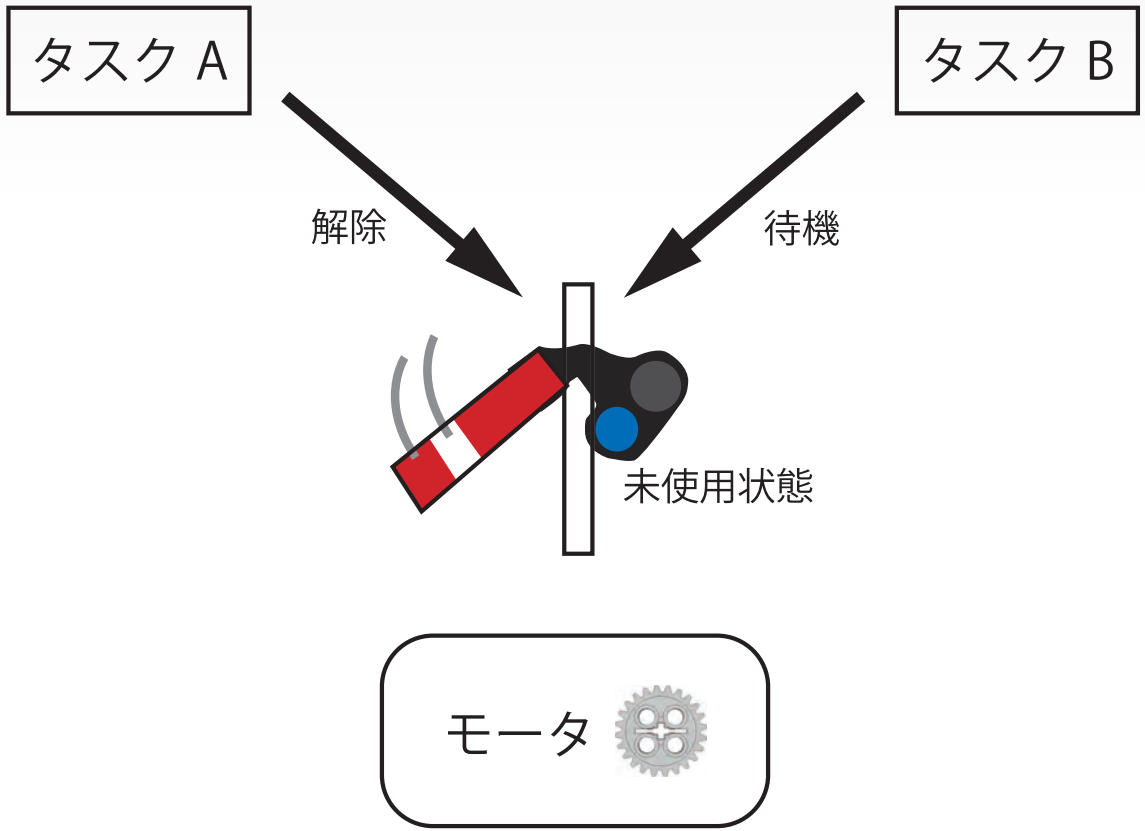
② タスク A によるモータ制御

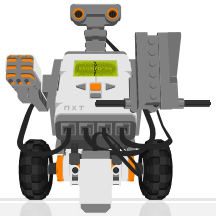




セマフォ（信号灯）

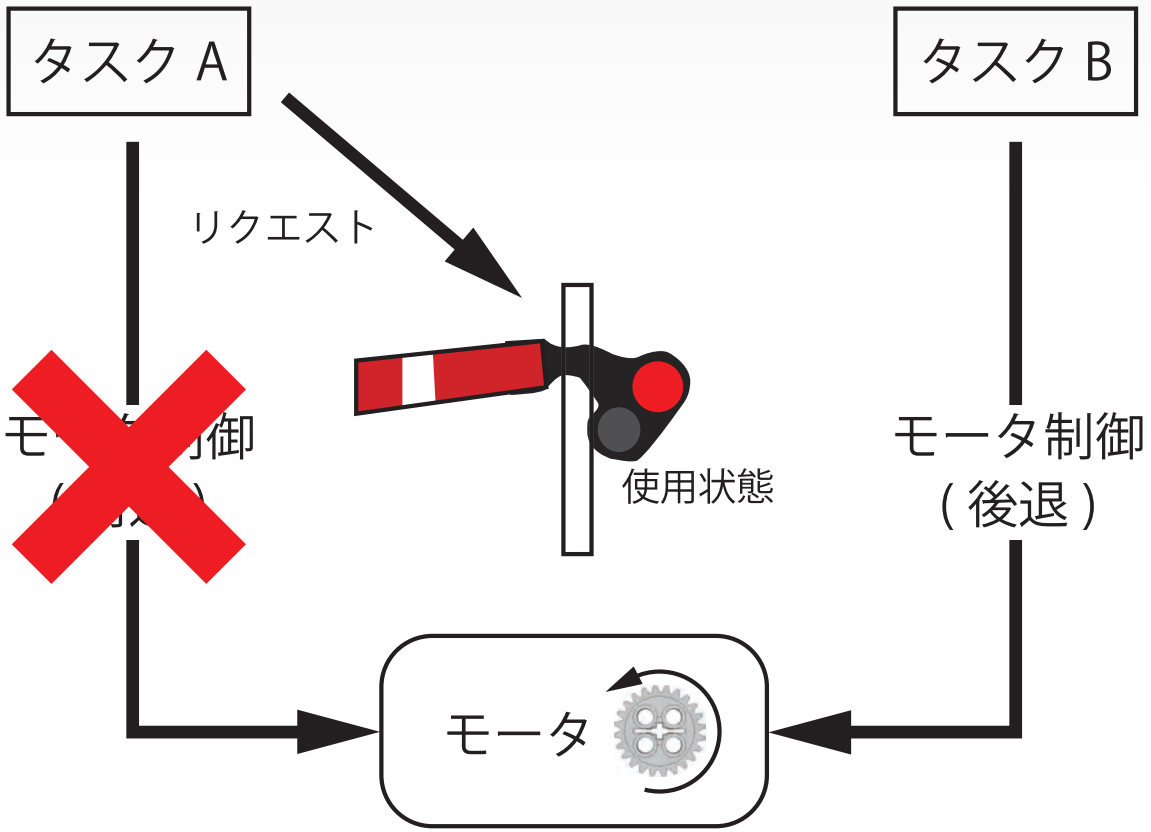
③ タスク A によるモータ制御終了

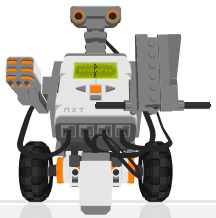




セマフォ（信号灯）

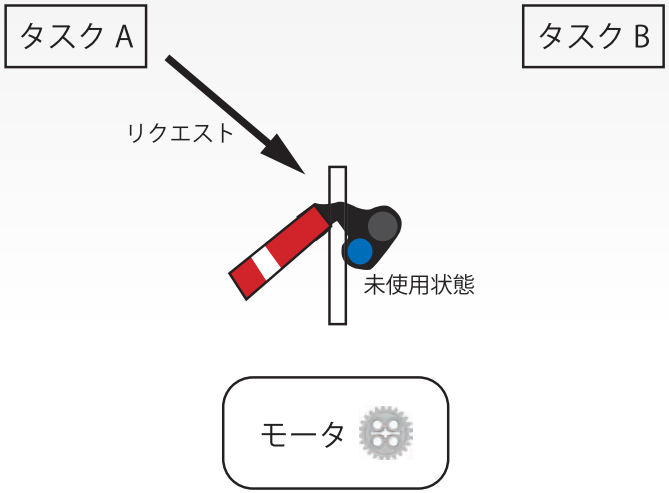
④ タスク B によるモータ制御



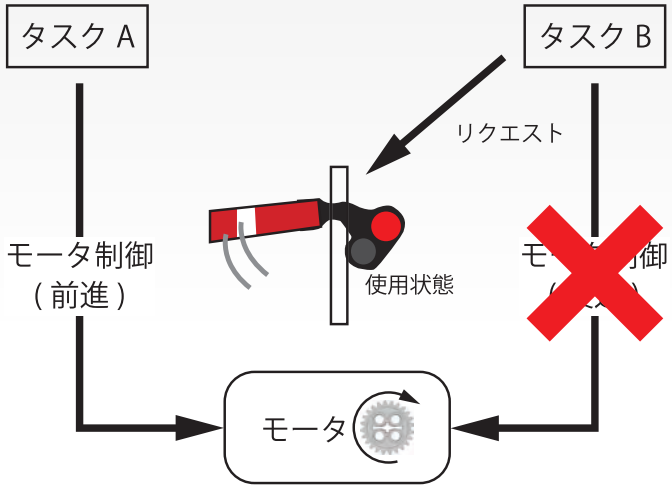


セマフォによる並列タスクのコンフリクトを回避

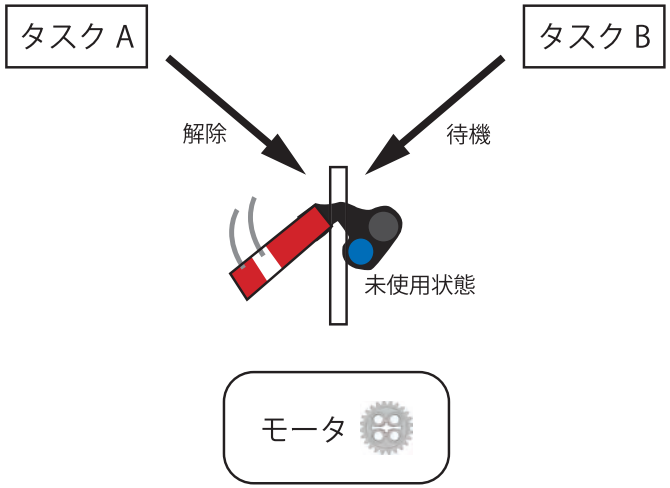
① モータの未使用状態



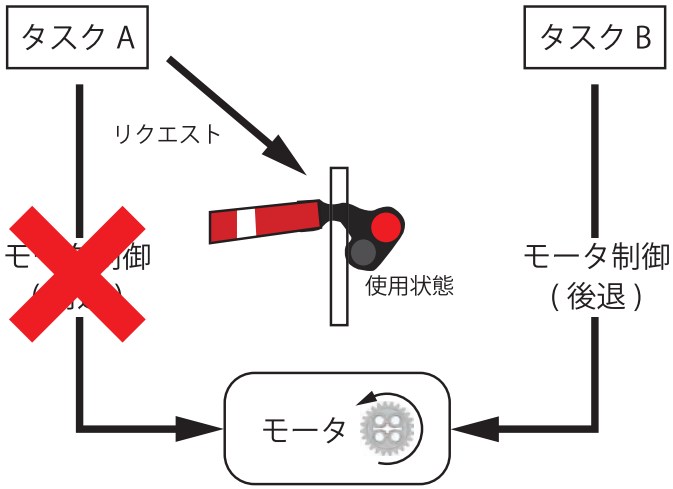
② タスク A によるモータ制御

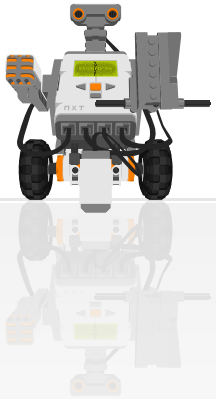


③ タスク A によるモータ制御終了

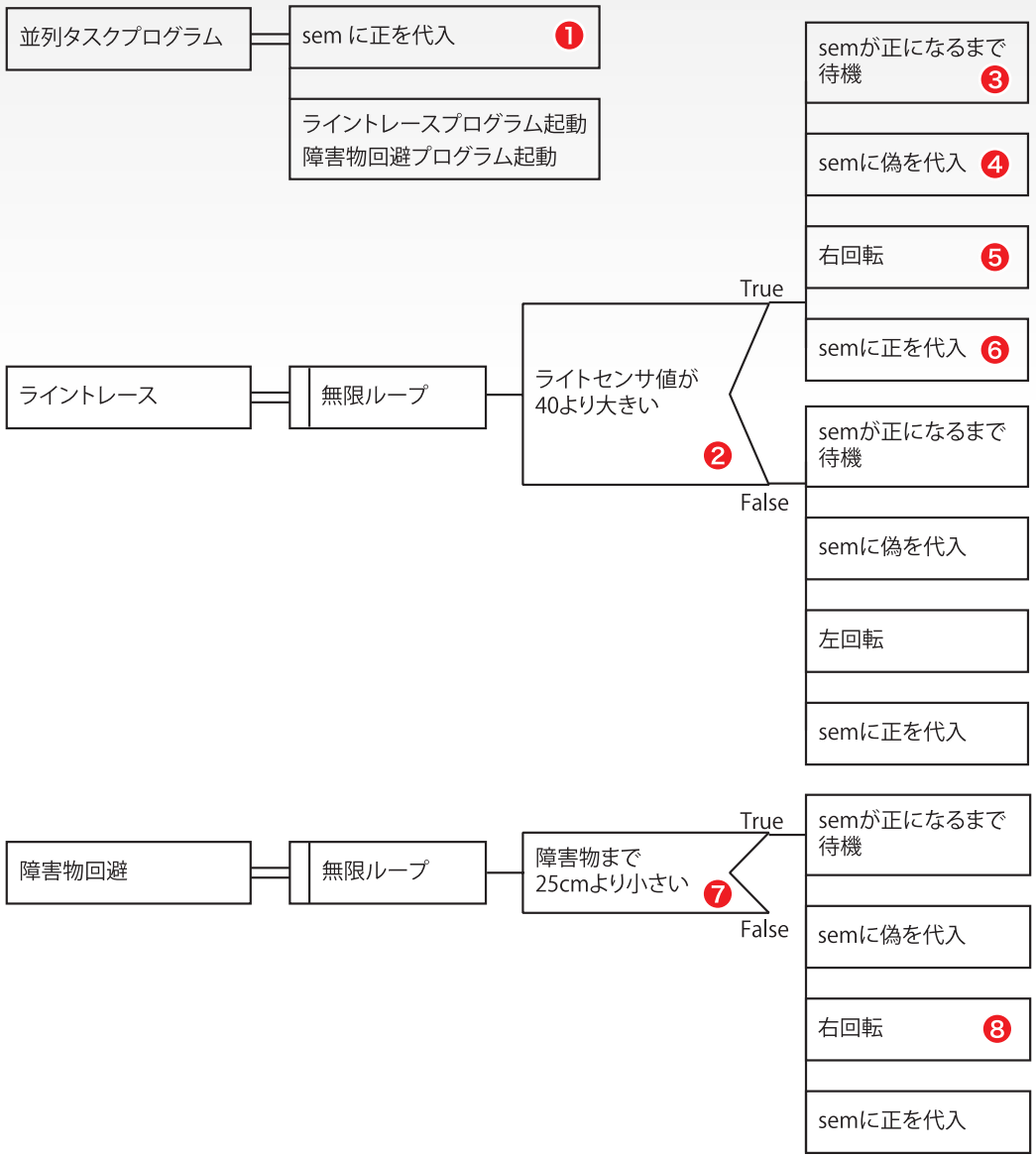


④ タスク B によるモータ制御



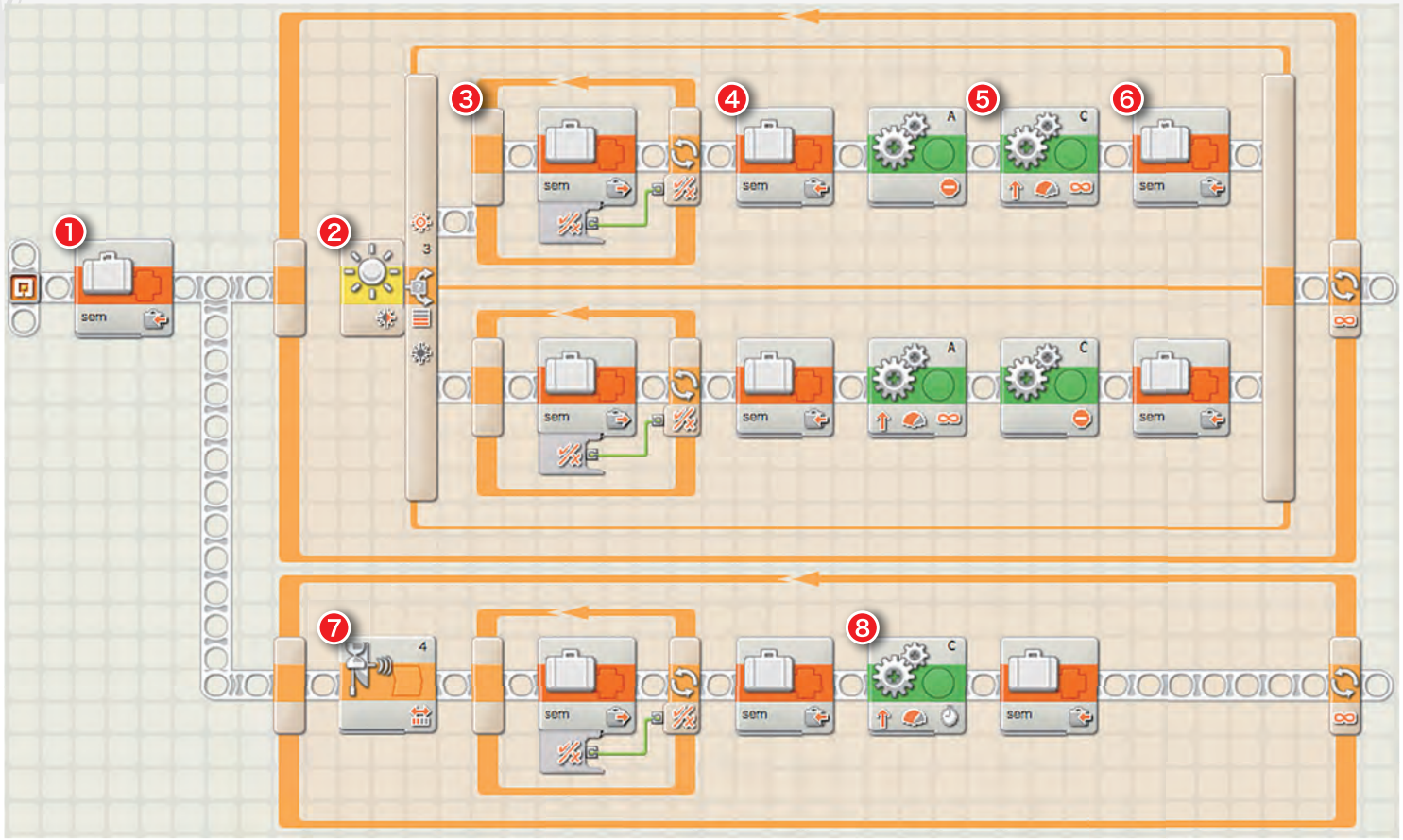


セマフォによるコンフリクト回避

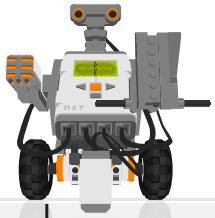


セマフォによるコンフリクト回避 (p.90: parallel_task2.rbt)

parallel_task2.rbt



ロジックタイプの変数
sem に正の値を書き込む



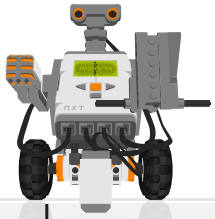
セマフォによるコンフリクト回避 (p.90: parallel_task2.rbt)

ライトセンサの値が 40 より大きいとき上段を実行，小さいときは下段を実行

sem が正になるまで待機

sem に偽の値を書き込む

モータ A を停止し，モータ C のみを回転して右回転

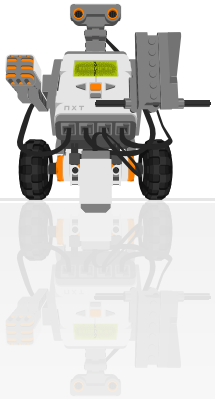


セマフォによるコンフリクト回避 (p.90: parallel_task2.rbt)

sem に正の値を書き込む

超音波センサが 25 より小さいとき障害物回避を実行

回転



■See you CU-Roboocon !