AI/IoTセンサのしくみを知ろう(応用編)

徳島大学技術支援部常三島技術部門
技術専門職員　辻　明典　博士(工学)
E-mail: a-tsuji@is.tokushima-u.ac.jp

2019年12月21日(土)

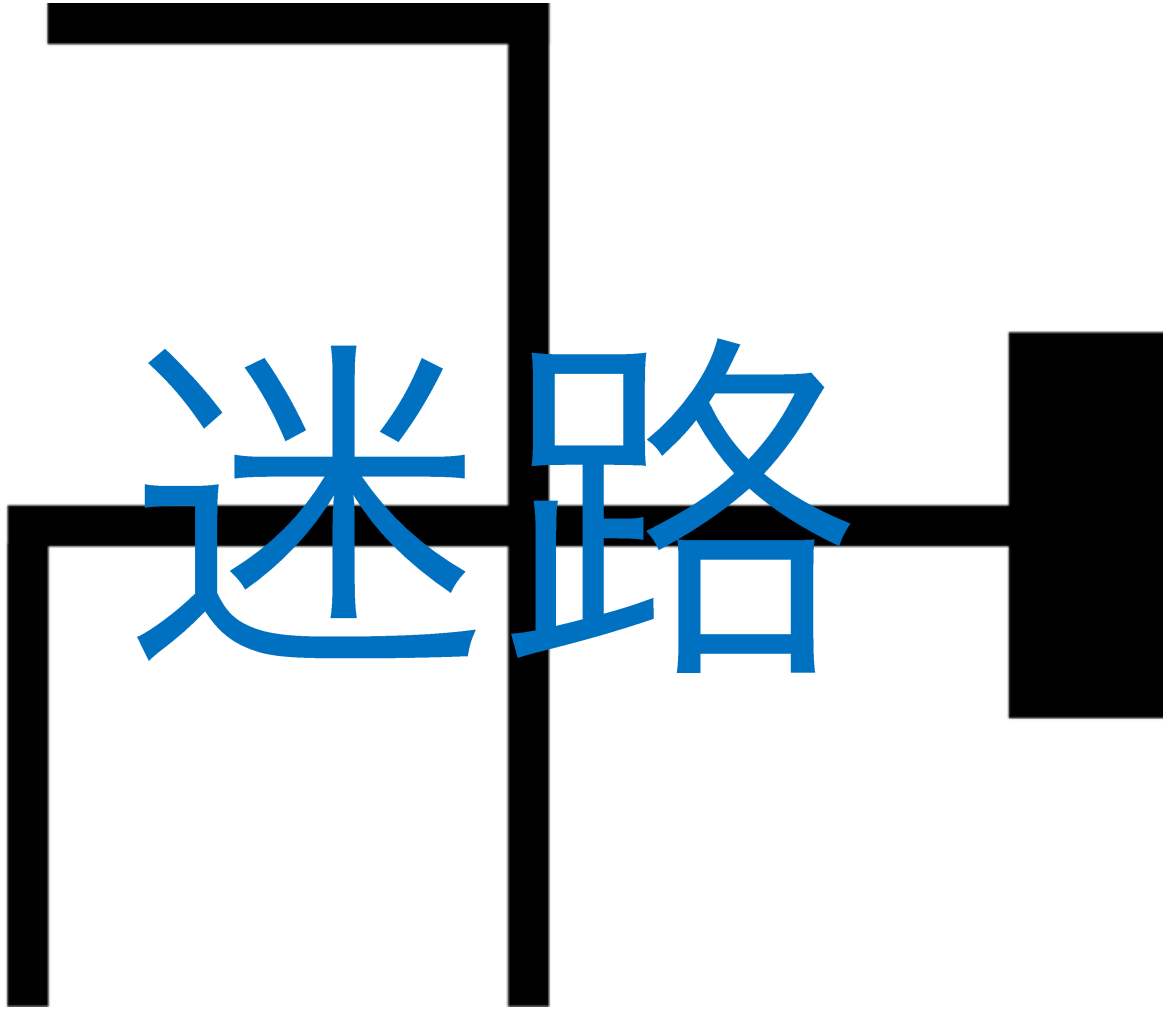
講座内容

- 講師：辻　　明典（徳島大学技術支援部）
  桑折　範彦（徳島大学名誉教授）
  川上　博　　（徳島大学名誉教授）
- 土曜日：10:00～11:30
- 日程：

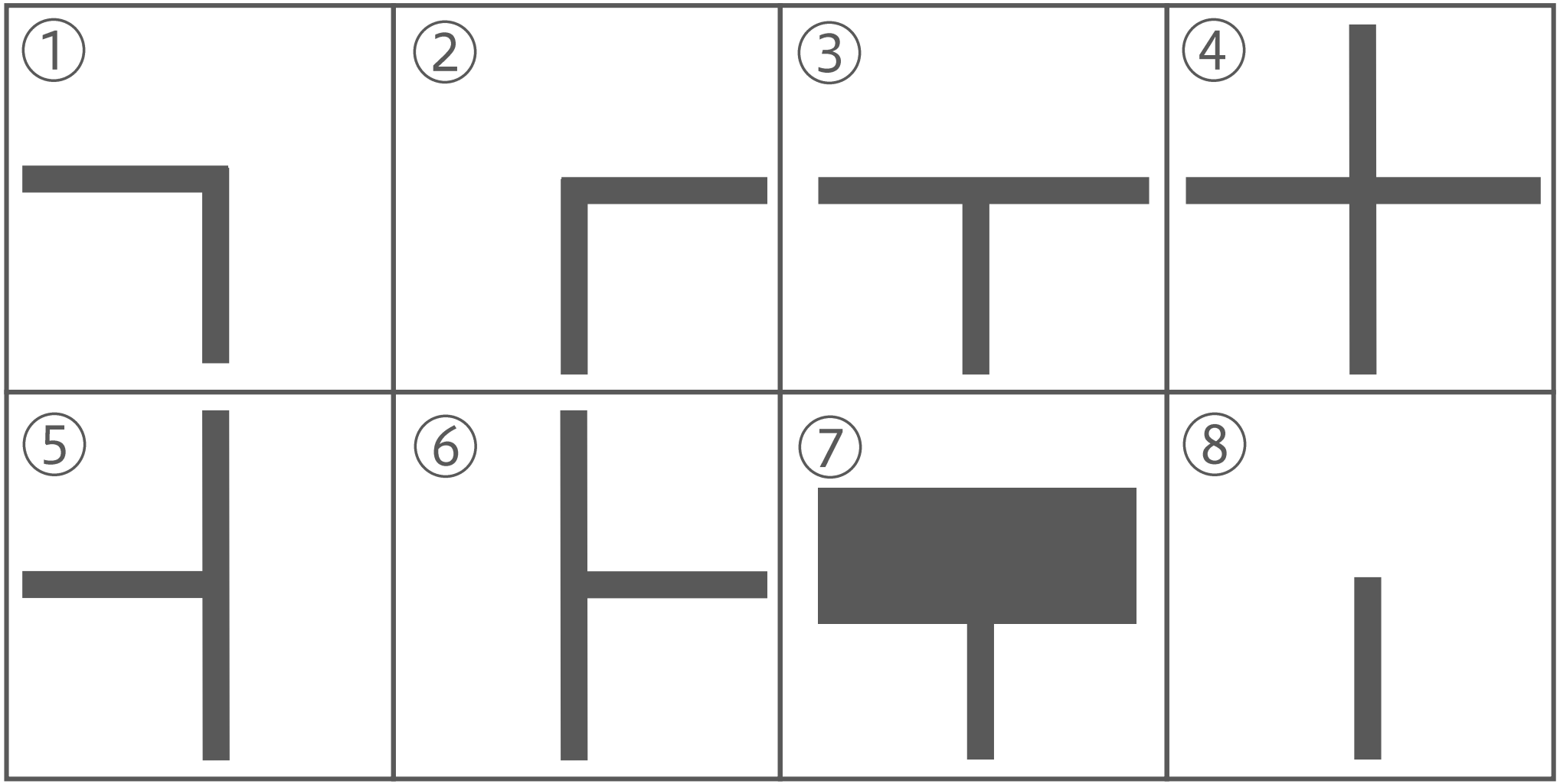
① 10 / 5　概要，環境設定，配布部品の確認
② 10 / 19 復習
③ 10 / 26 ロボットのモーター1（基本動作）
④ 11 / 9　ロボットのモーター2（応用動作）
⑤ 11 / 16 ロボットの制御1
（モーター，センサの協調動作）
⑥ 11 /30　ロボットのセンサ1
（フォトリフレクタ）
⑦ 12 / 7　ロボットのセンサ2
（ライントレース1）
⑧ 12 / 14　ロボットの制御2
（ライントレース2）
⑨ 12 / 21　ロボットの制御3
（迷路課題）

迷路探索

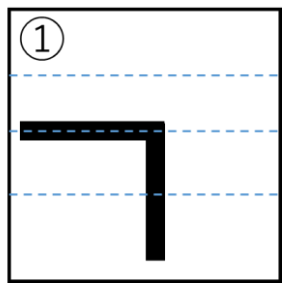
- 迷路探索
  - ライントレースの応用
- 必要なロボットの動き
  - “正確な”ライントレース
  - “正確な”回転
  - “正確な”ライン判定
- 机の上を片付ける！

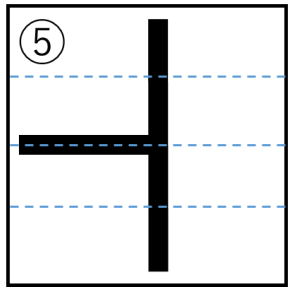


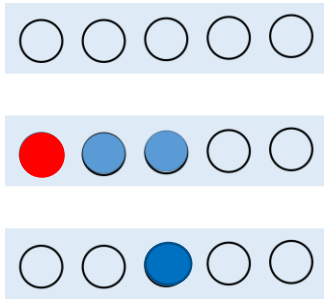
迷路パターン(8種類)

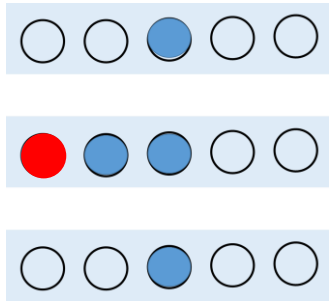


Ex0901：迷路パターンの判別



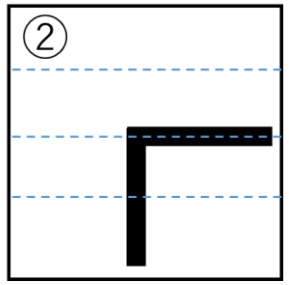


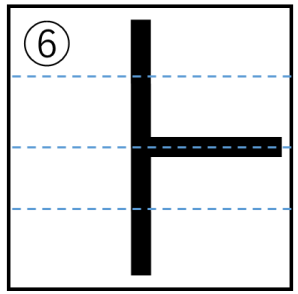


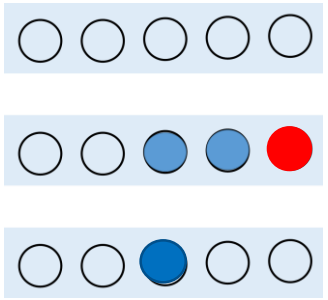


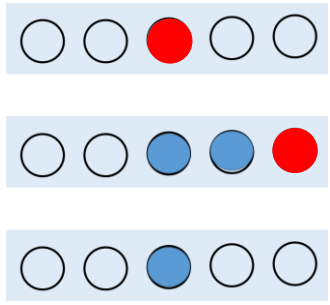
左折(L)

左折(L)
(直進)



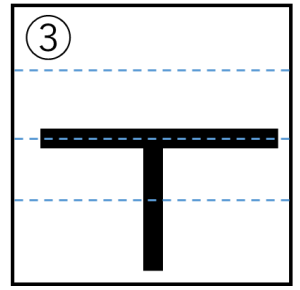


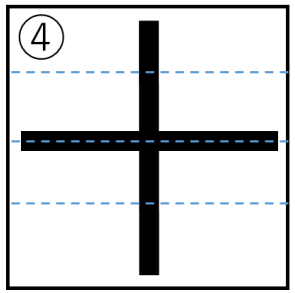


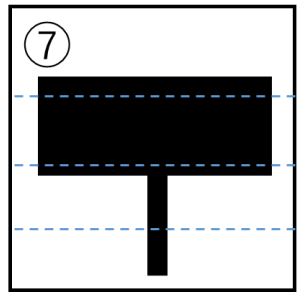


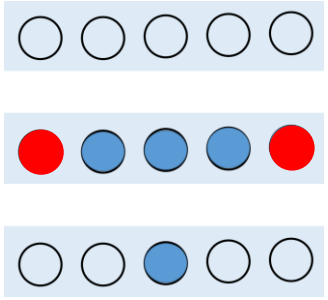
右折(R)

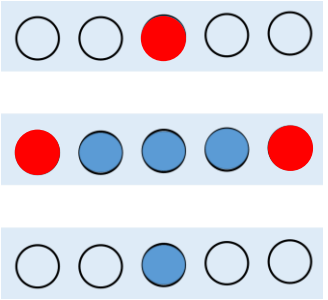
直進(S)
(右折)

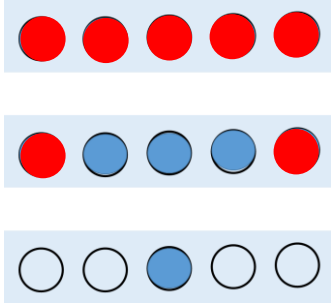








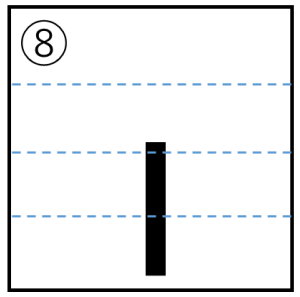


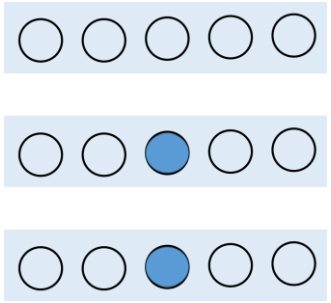


左折(L)
(右折)

左折(L)
(直進)
(右折)

ゴール





旋回(T)

交差点：左端，右端のセンサで検出
左端：pr_bin[0] == 1
右端：pr_bin[4] == 1

交差点の判定

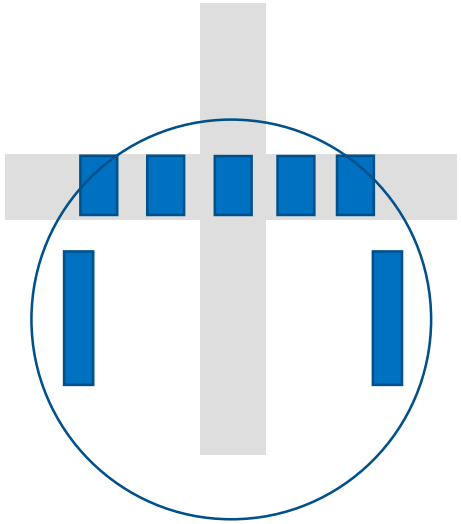
- T字，四角の判定
  - 左端，右端のセンサで交差点判定
  - ロボットがまっすぐ交差点に進入しない

交差点の判定

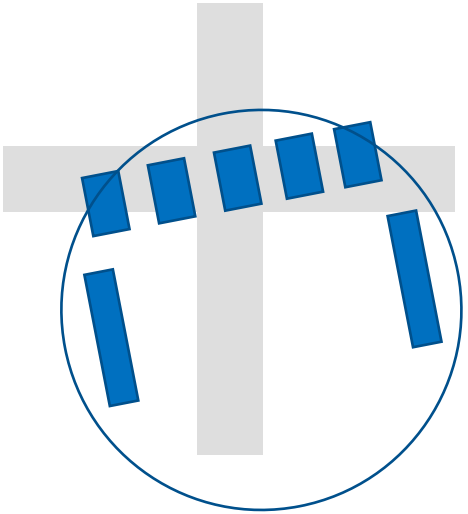
- 少し進んでラインの上で再判定
  - 左端または右端のセンサが反応したら，少し進んで，再度，ライン検出

```
if (pr_bin[0] == 1 || pr_bin[4] == 1) {
  delay(I_TIME);
  get_pr();
}
```

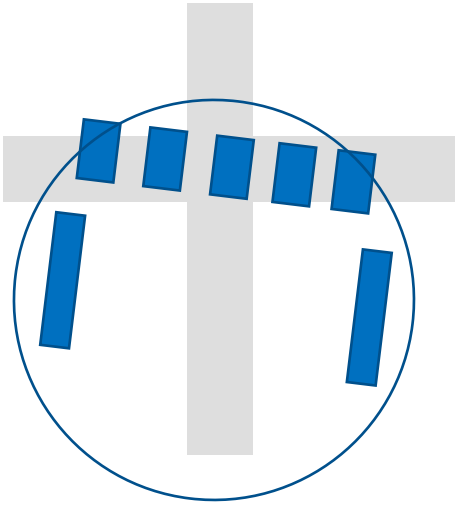
OK！



OK！



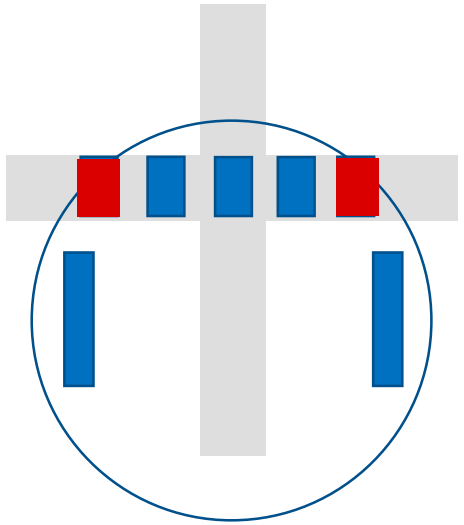
OK！



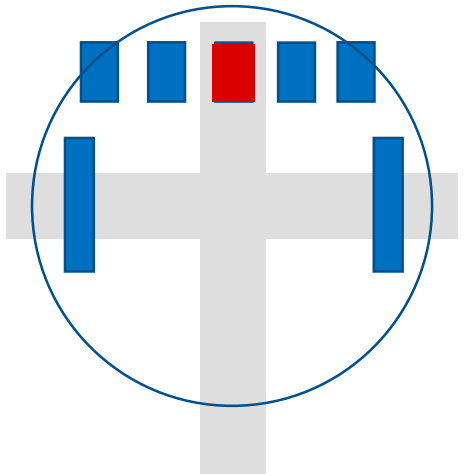
交差点の旋回

- 交差点での旋回
  - 左旋回 turn_L
  - 右旋回 turn_R

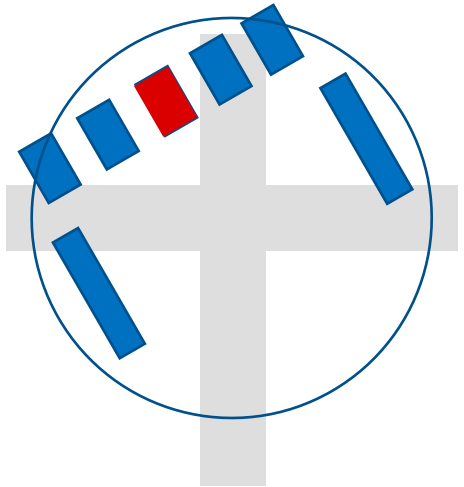
交差点の判定
左端・右端のセンサ反応



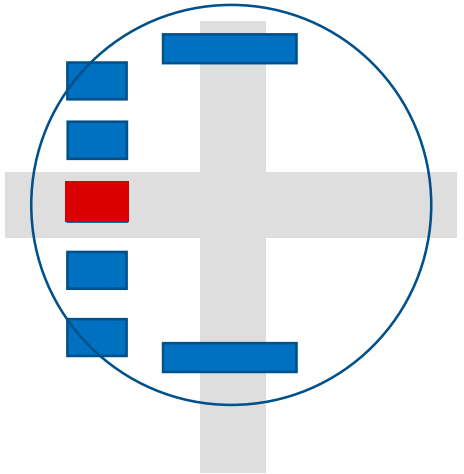
交差点から少し進む
中央のセンサ反応！



ラインをよける
中央のセンサが
反応しなくなるまで旋回

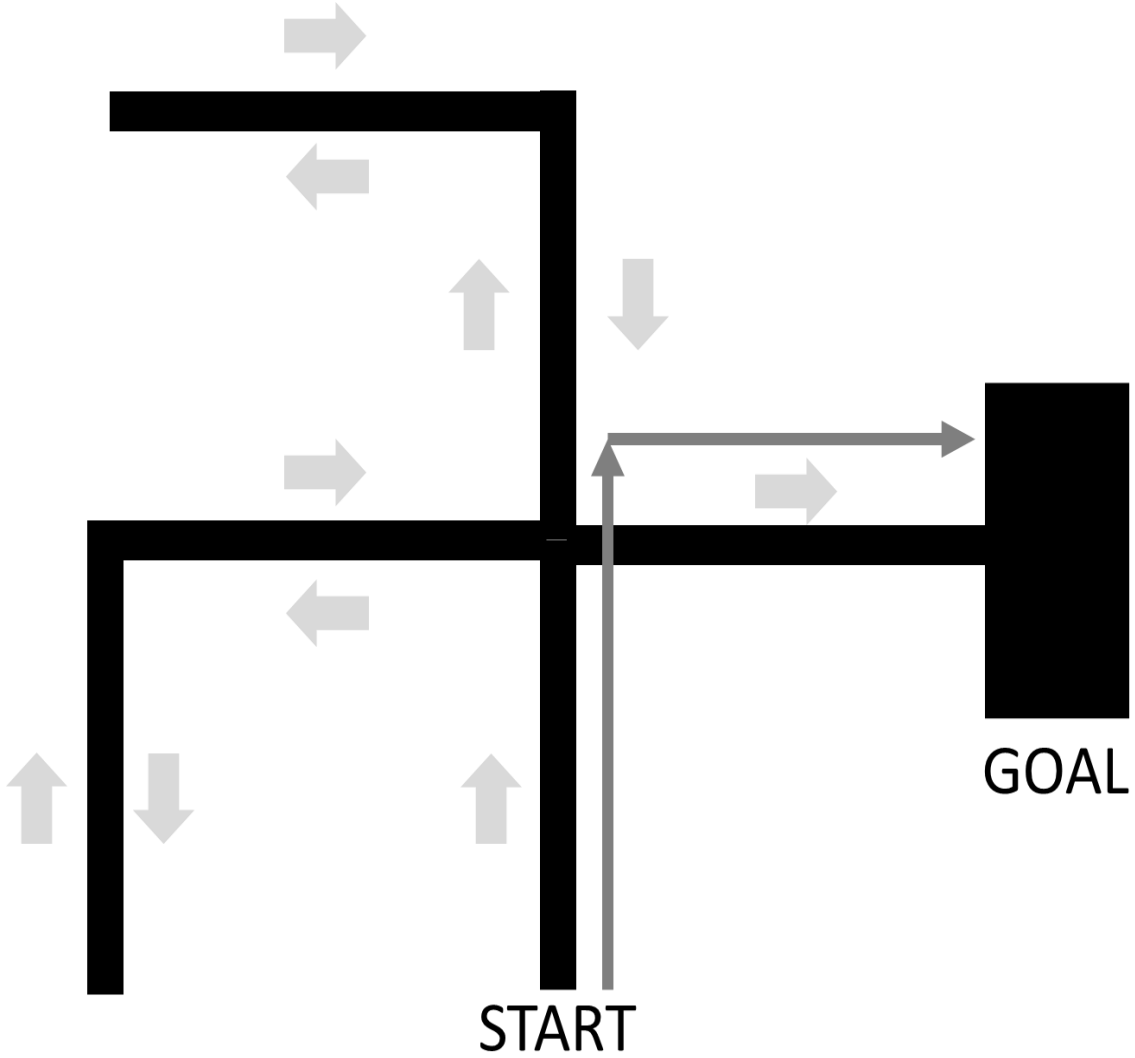


ラインの検出
中央のセンサが反応
するまで旋回



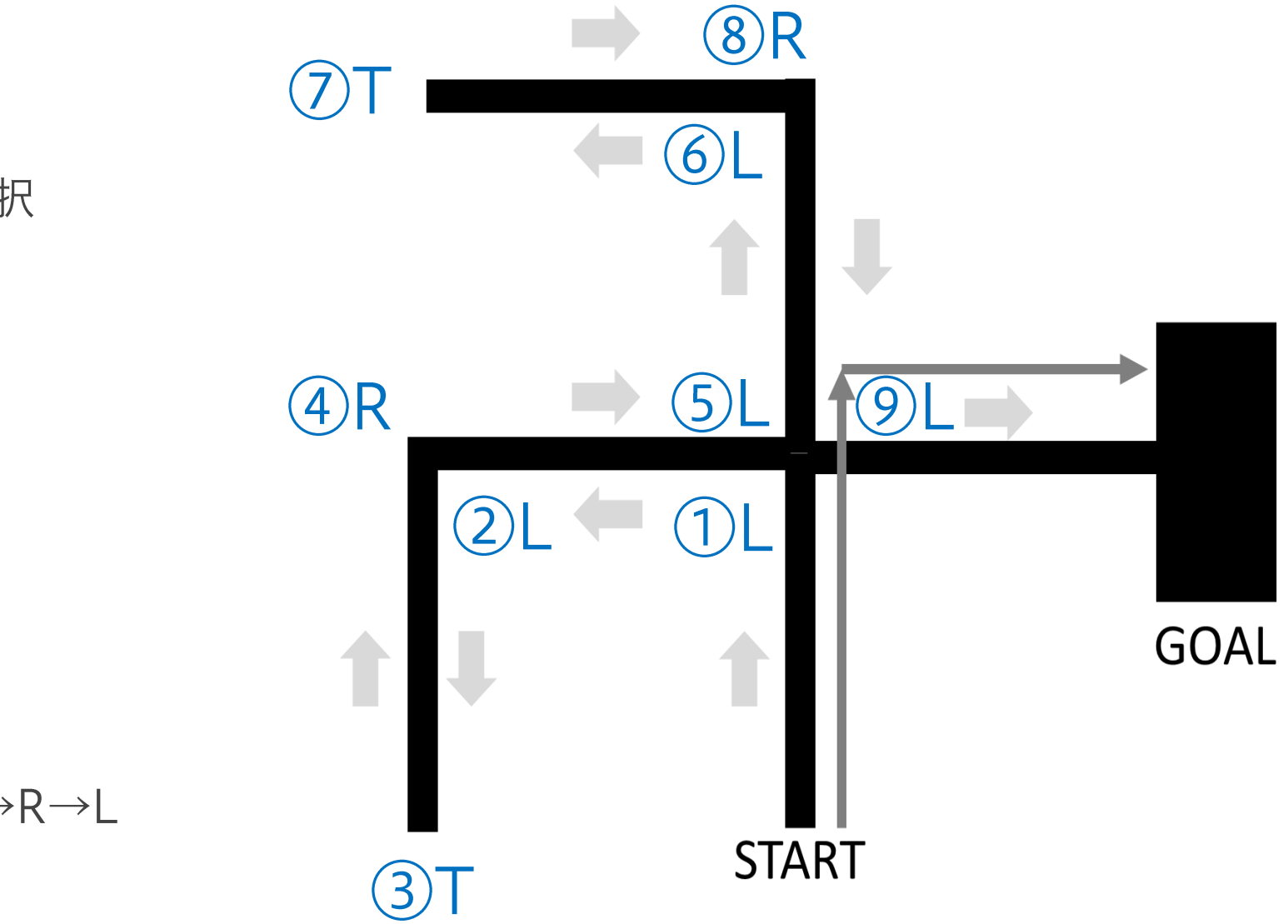
迷路探索(左手法)

- 迷路探索
  - START－＞GOALまでの
    最短経路を求める
- 左手法
  - 左手を壁伝いに迷路を回ると
    ゴールに到達
  - 無駄な経路(パス)が多数あり
- 最短経路
  - START→四つ角(右折)→GOAL



左手法の経路

- 左手法の経路
  - 交差点で判定
  - 交差点では「左折」を選択
  - 行き止まりは旋回
- パスの記号
  - L：左折(Left turn)
  - T：旋回(Turn)
  - R：右折(Right turn)
  - S：直進(Straight)
- ゴールまでの経路
  - L→L→T→R→L→L→T→R→L
  - 経路長：9



Ex0902：迷路の経路を記録

- 交差点：各パターンにパスの記号を割り当てる
- 通過記録：maze_path[] 配列
- 経路長：path_len

(例) 行き止まり('T')

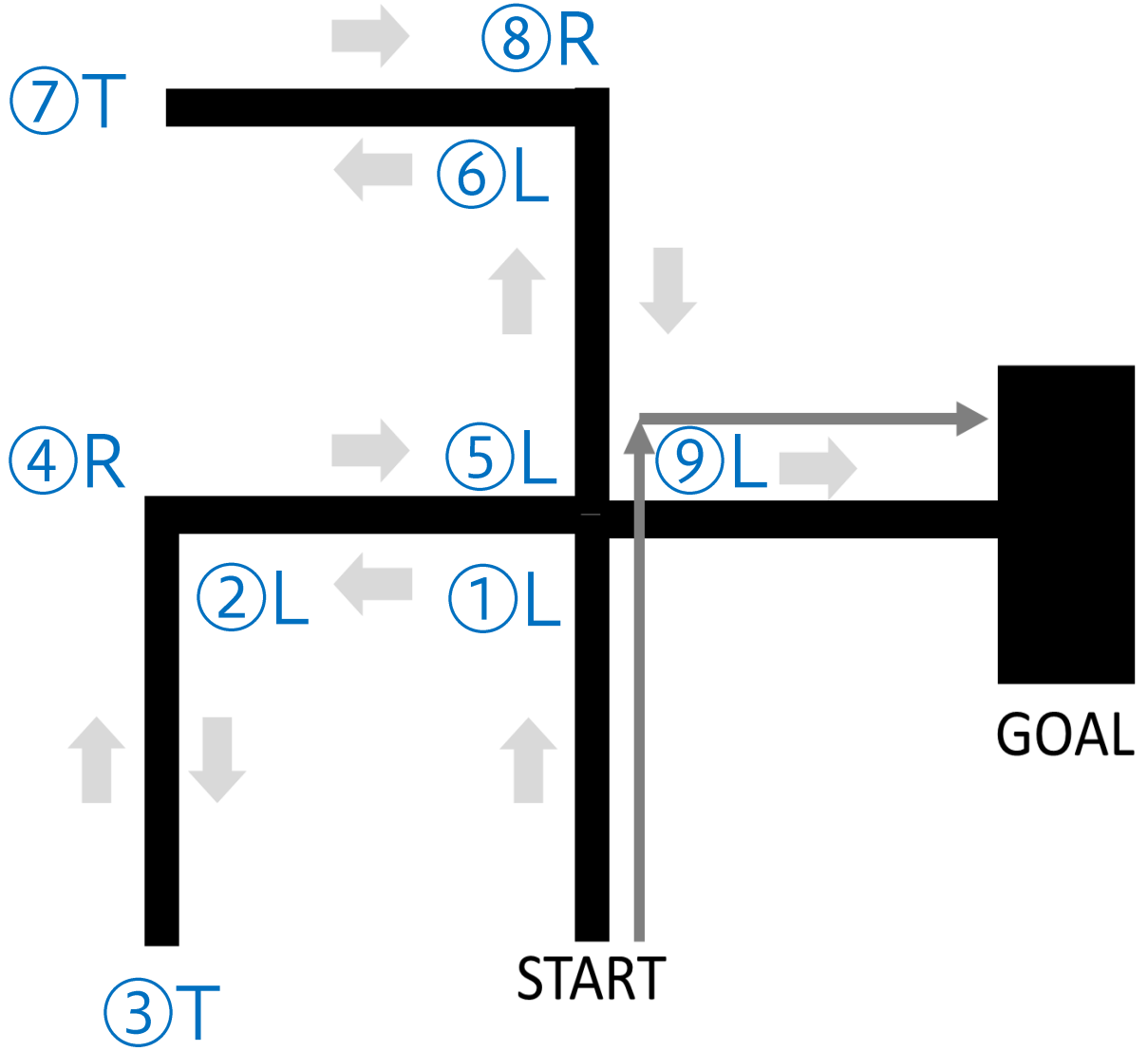
```
if (pr_bin[0] == 0 && pr_bin[1] == 0 && // ラインなし
    pr_bin[2] == 0 && pr_bin[3] == 0 &&
    pr_bin[4] == 0) {
  maze_path[path_len] = 'T';
  path_len++;
  go_D(D_TIME);
  turn_L(T_TIME);
 }
```

基本パターンで経路の記録を確認

1. ボタンを押してスタート
2. 経路長＝5・・・交差点5回通過
3. USBケーブルを接続
4. シリアルモニターを起動
5. 経路を確認

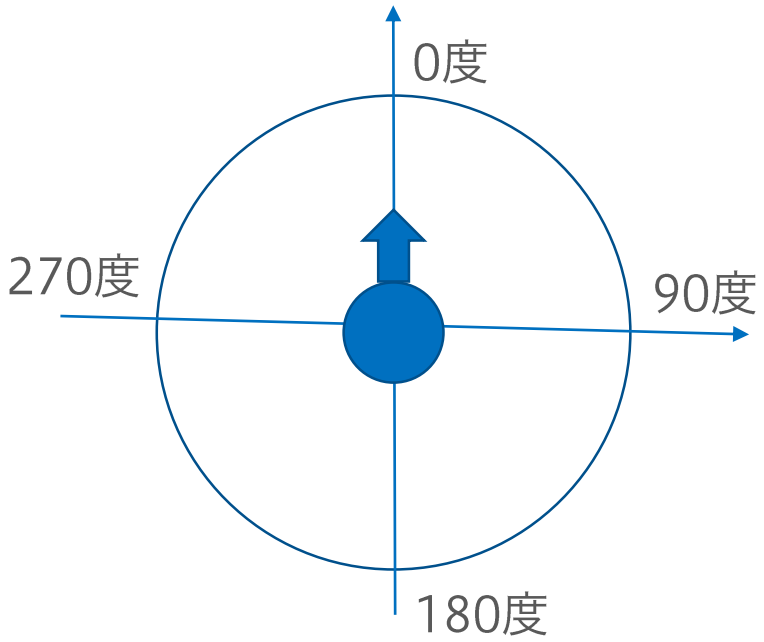
最短経路の探索

- 左手法の経路
  - L→L→T→R→L→L→T→R→L (path_len = 9)
- 最短経路を求める
  - L[LTR]（LTR=>T）
  - L[T]L　（LTL=>S）
  - [S]LTR（LTR=>T）
  - [S][T]L（STL=>R）
  - [R]　(path_len=1)
- 旋回(T)を含むと無駄経路？
  - ?[T]?



最短経路の短縮ルール

- ロボットの状態
  - S: 0度
  - R: 90度
  - T: 180度
  - L: 270度



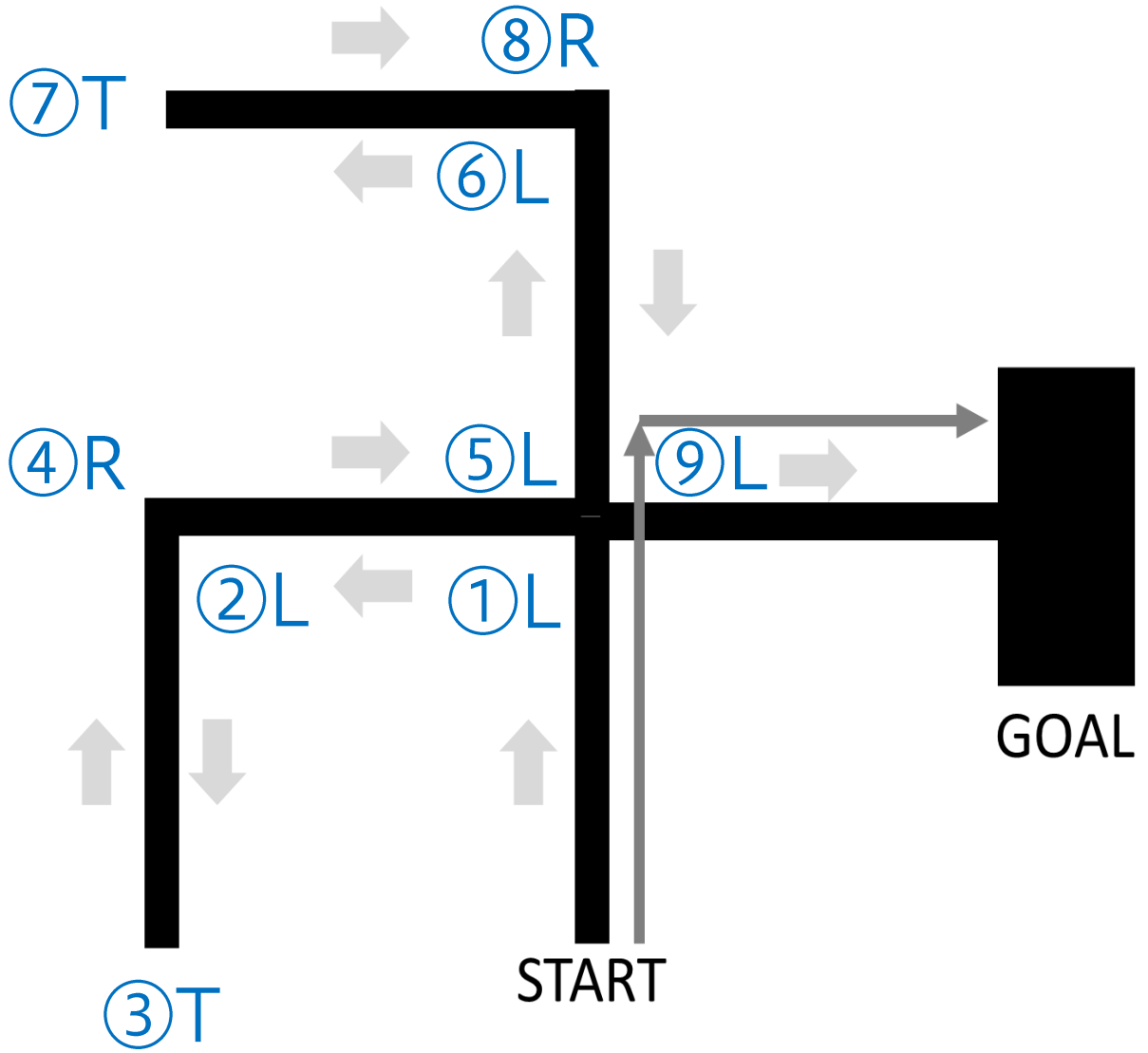
ルール：経路の短縮

LTS => R (270+180+0 =450)：90(R)
LTR => T (270+180+90 =540)：180(T)
LTL => S (270+180+270=720)：0(S)
RTS => L (90+180+0 =270)：270(L)
RTR => S (90+180+90 =360)：0(S)
RTL => T (90+180+270 =540)：180(T)
STS => T (0+180+0 =180)：180(T)
STR => L (0+180+90 =270)：270(L)
STL => R (0+180+270 =450)：90(R)

余りを求める：X % Y = A･･･a
(例) 450 % 360 = 1･･･90
720 % 360 = 2･･･0

Ex0903：最短経路を探索

- 最短経路の探索(find_path)
- ゴールまでの流れ(maze_path[])
  - ①L
  - ②LL
  - ③LLT
  - ④L[LTR]　LTR=>T
  - ⑤L[T]L
  - ⑥[S]L
  - ⑦[S]LT
  - ⑧[S][LTR]　LTR => T
  - ⑨[S][T]L　STL => R
- 最短経路：R, path_len = 1



Ex0904：ゴールアクション

- ゴールに到達したときのアクションを考える
- goal_action()
  - LEDを光らせる
  - 音を鳴らす
  - ロボットを回転させる
  - など
- 自由に追加してください。

```
void loop() {
  while (maze_pattern()); // 迷路のゴールに到達するまで
  goal_action(); // ゴールアクション
  check_path(); // 迷路の経路確認
}
```

Ex0905：迷路探索の後，最短経路で走行

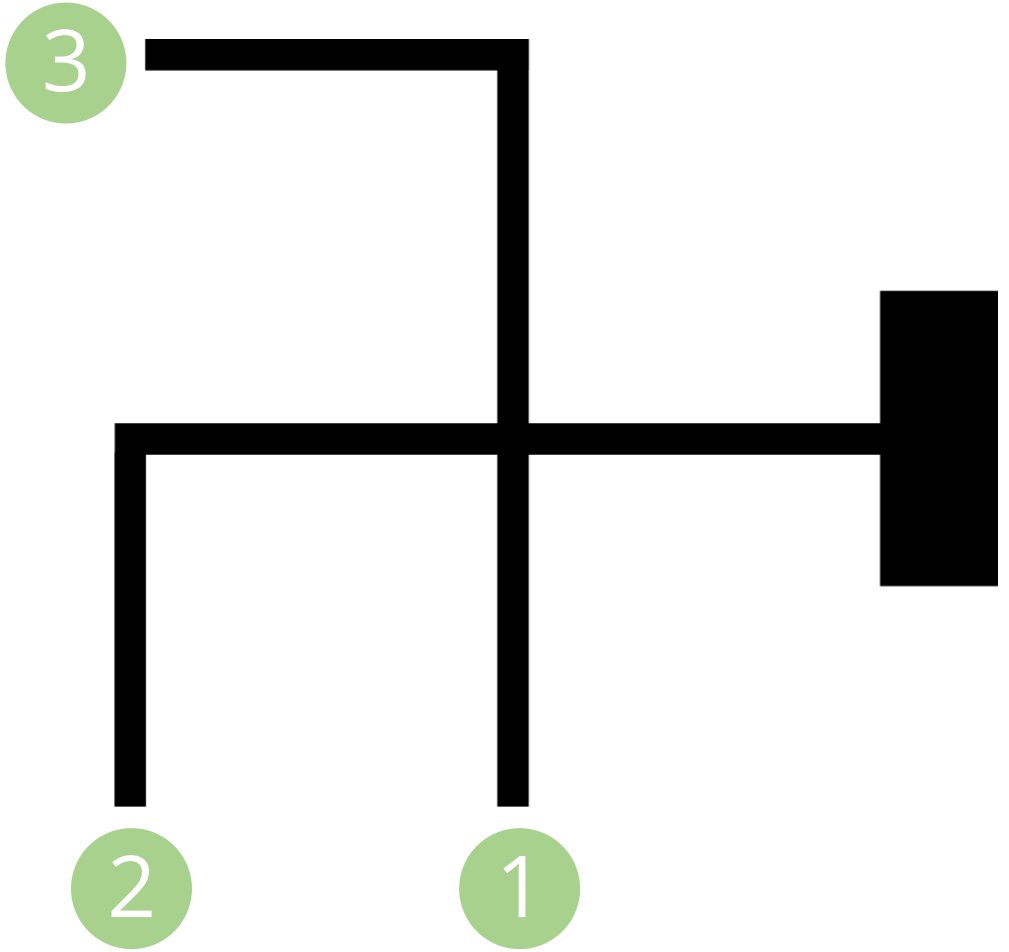
- スタート地点からゴールまで走行 maze_pattern()
- スタート地点に戻す
- 最短経路でゴールまで走行 maze_solve()

```
void loop() {
  while (maze_pattern()); // 迷路のゴールに到達するまで(左手法)
  goal_action(); // ゴールアクション
  check_path(); // 迷路の経路確認

  while (maze_solve()); // 迷路の探索(最短経路)
  goal_action(); // ゴールアクション
  check_path(); // 迷路の経路確認
}
```

課題1：最短経路の探索

- ①から始めて最短経路を探索
- ②から始めて最短経路を探索
- ③から始めて最短経路を探索



課題2：迷路探索のスピードアップ

- go_Dのmotor_stop時間を減らす
- 各待ち時間変数の調整
- モータースピードspを変更
- PD制御パラメタ調整
- など

- まず，基本パターンで動作確認

お疲れさまでした！

- この講座について

感想

課題

改善

?