■ ロボットの組み立て

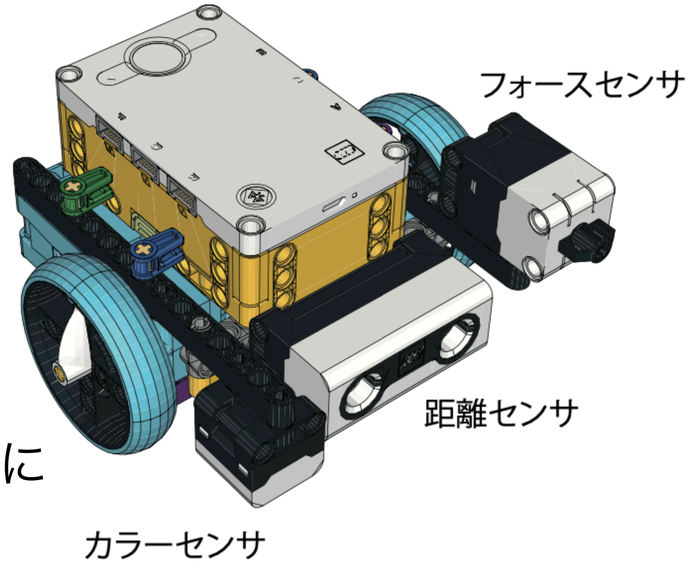
ロボットの組み立て

SPIKE App ⇒ 組み立てガイド ⇒ **ドライビングベース３＋センサ**

組み立てガイドを参考にドライビングベースとセンサを組み立てる

① **ベース・ロボット：手順 1～26**
② **カラーセンサ：手順 1～4**
③ **フォースセンサ**※
④ **距離センサ**※

を組み立てよう！



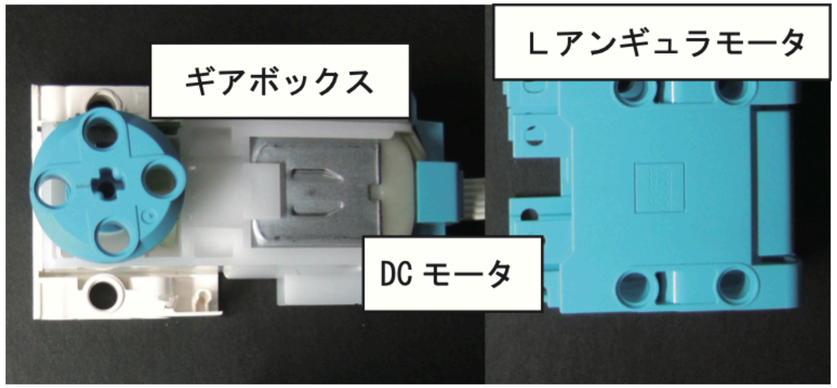
※ フォースセンサと距離センサはセンサ裏に

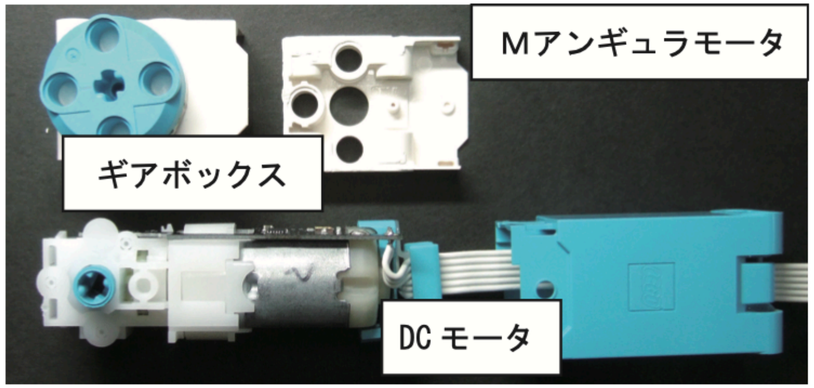
をはめてイラストのように取り付ける

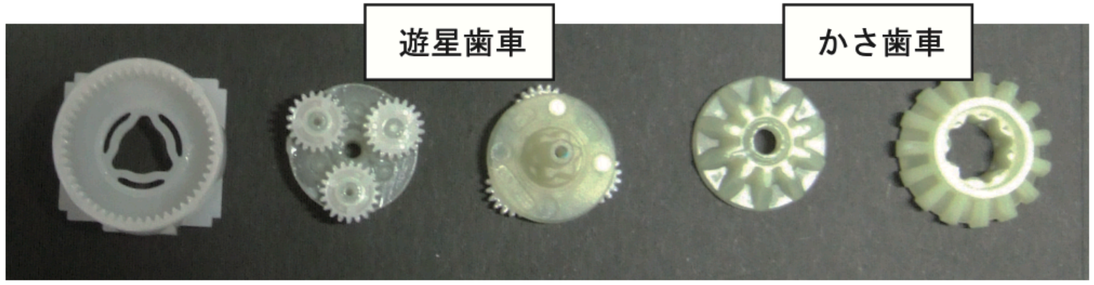
フォースセンサは**ポートA**に
距離センサは**ポートE**に接続

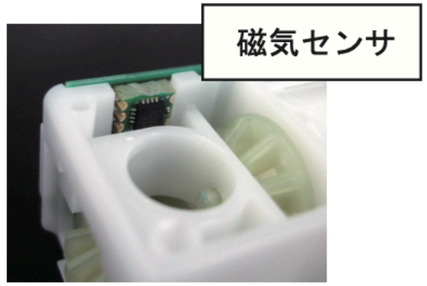
■ロボットを前進させるには(モータ制御１)

アンギュラモータ









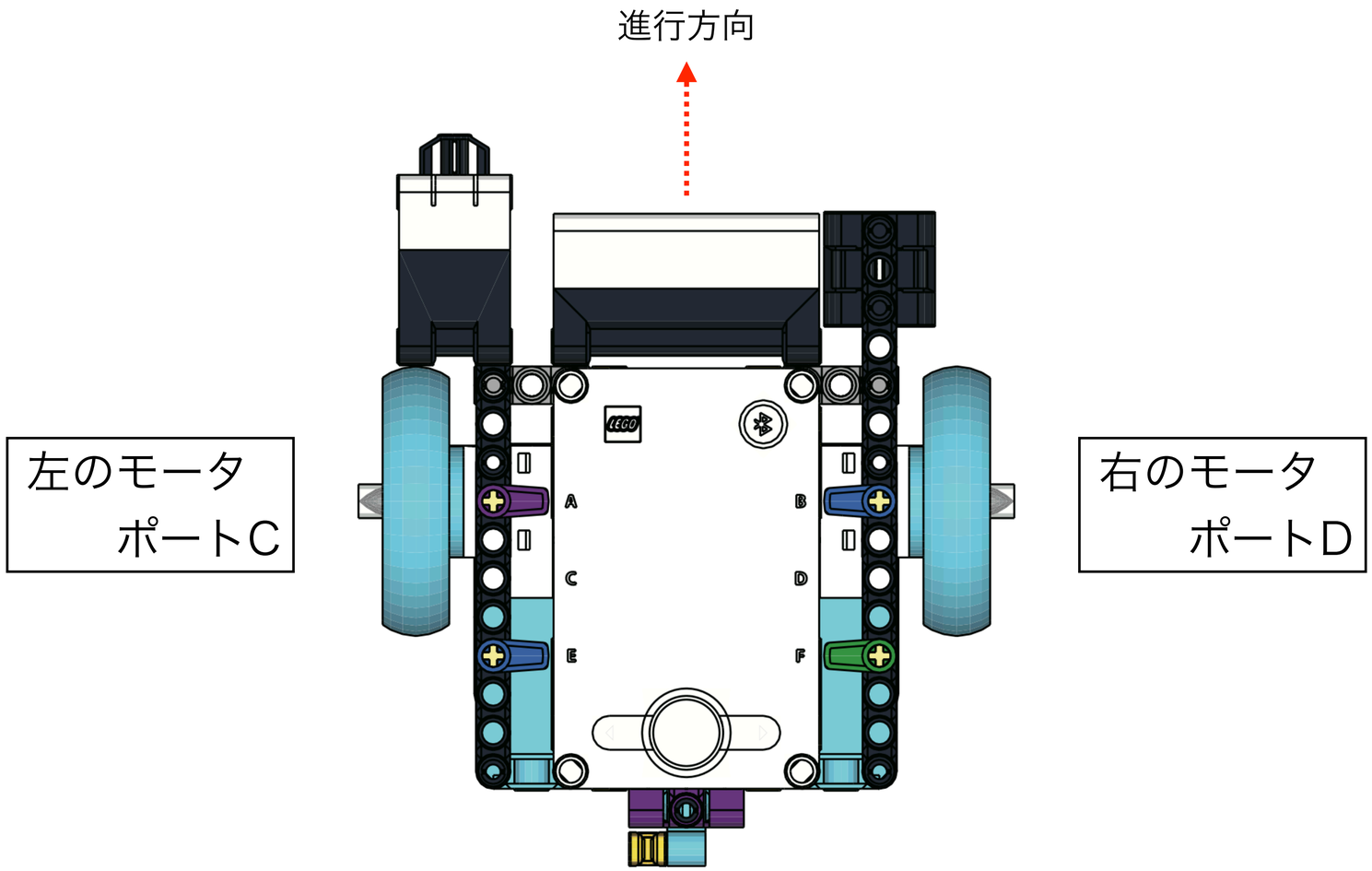
Lアンギュラモータ：
停動トルク 25Ncm
回転数 175RPM(無負荷時)

Mアンギュラモータ：
停動トルク 18Ncm
回転数 185RPM(無負荷時)

遊星歯車とかさ歯車を組み合わせて使用(Lモータ、Mモータ同じものを使用)
磁気センサを使用して回転数と回転角度の情報を得る(1回転で360カウント)

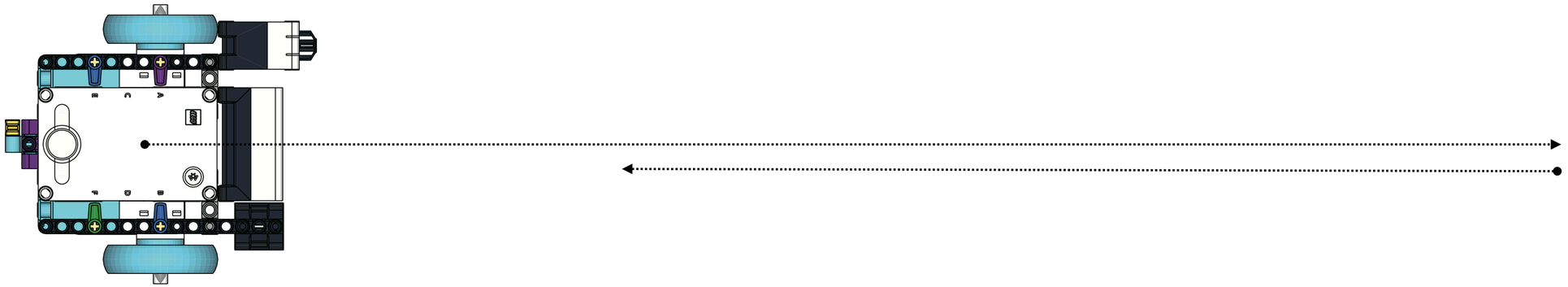
モータの接続

- SPIKEのどのポートにモータが接続されているか確認



前進プログラムのアルゴリズム

・　ロボットを3秒前進、その後2秒後退

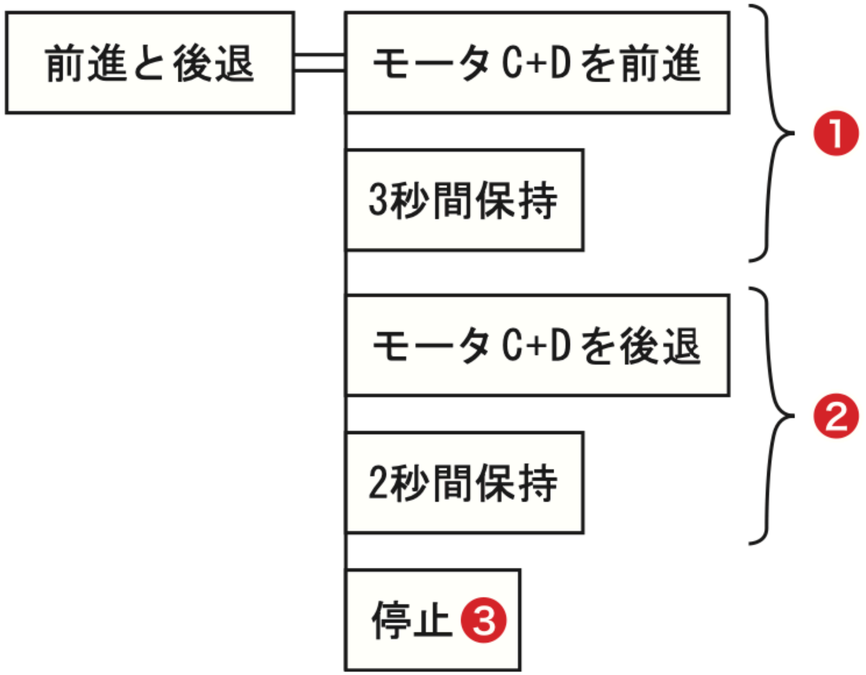


モータCと**モータD**を３秒間回転

モータCと**モータD**を２秒間逆回転

アルゴリズムを考えよう (SPIKE-WB)

- 簡単なアルゴリズム表現を使う
  - 速度50%で３秒間前進して2秒後退



タンクブロック

今回は拡張機能の中の移動拡張にあるタンクブロックを使用

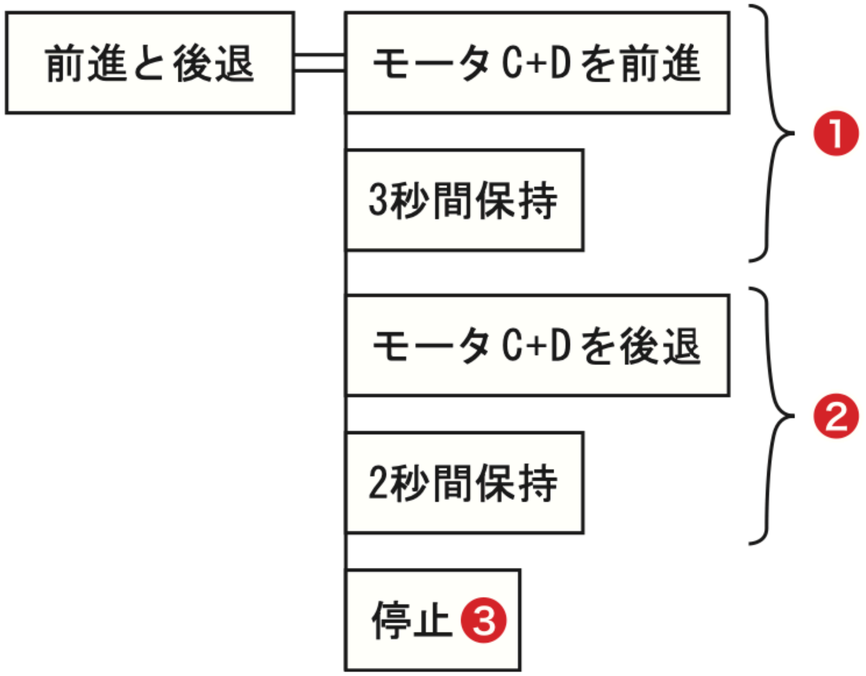
モータ制御によるロボットの前進 (タンクブロック)

・ ロボット(モータB+C)を3秒前進、その後2秒後退

p.40：motor.llsp3

アルゴリズムを考えよう（Python）

- 簡単なアルゴリズム表現を使う
  - 速度500（回転角度/秒）で３秒間前進して2秒後退



速度＝回転角度/秒

モータ制御によるロボットの前進 （Python）(モータペア)

・ ロボット(モータB+C)を3秒前進、その後2秒後退

p.41：motor.py

```
from hub import port
import motor_pair
import time

motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)
motor_pair.move_tank(motor_pair.PAIR_1, 500, 500)
time.sleep_ms(3000)
motor_pair.move_tank(motor_pair.PAIR_1, -500, -500)
time.sleep_ms(2000)
motor_pair.stop(motor_pair.PAIR_1)
```

1
2
3

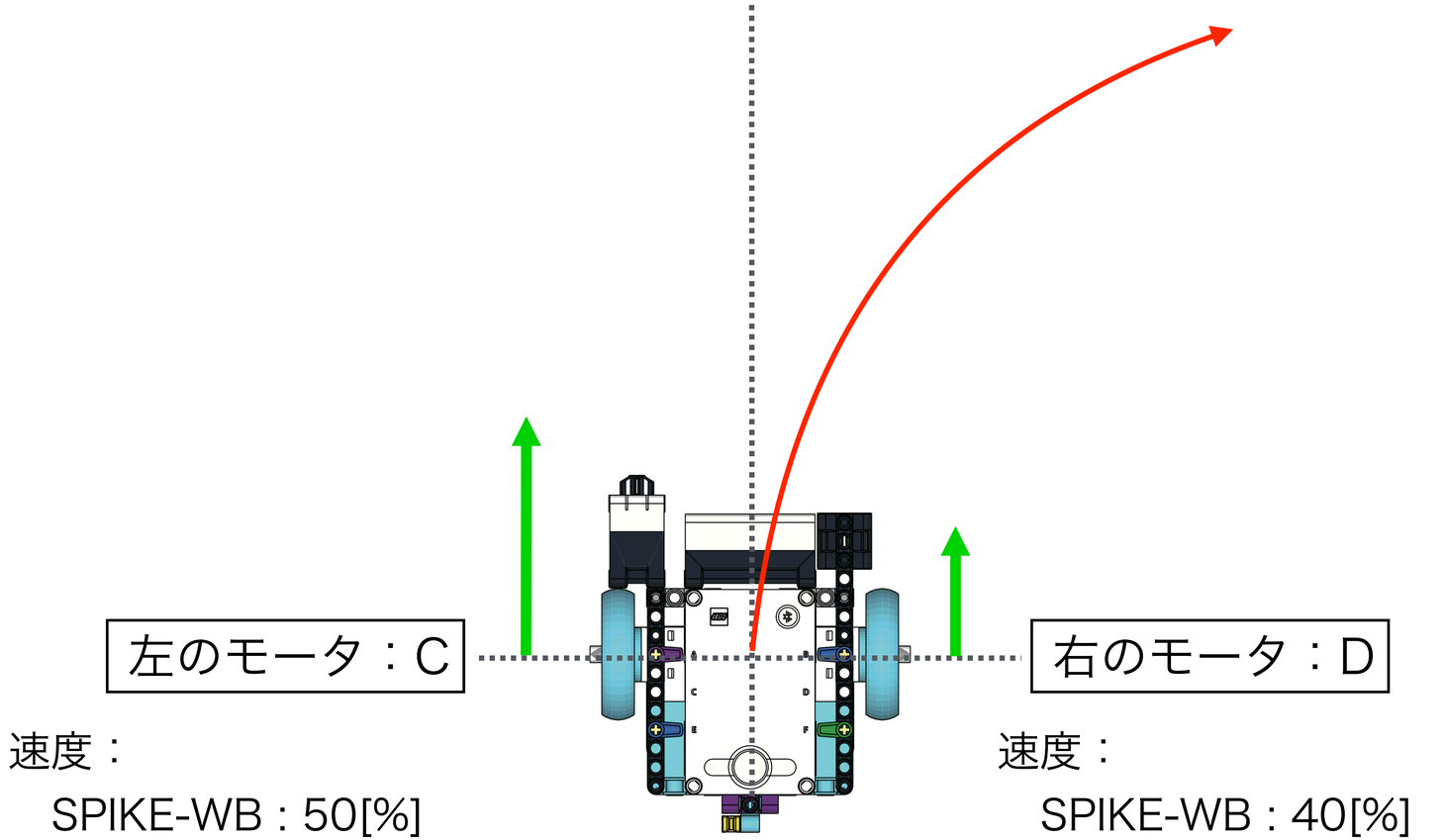
motor_pair.pair()　　：　モータのペアの設定

motor_pair.move_tank()　：　ペアのモータを動かす（速度を指定：-1,110~1,110）

time.sleep_ms()　　：　直前の状態の保持(単位はミリ秒)

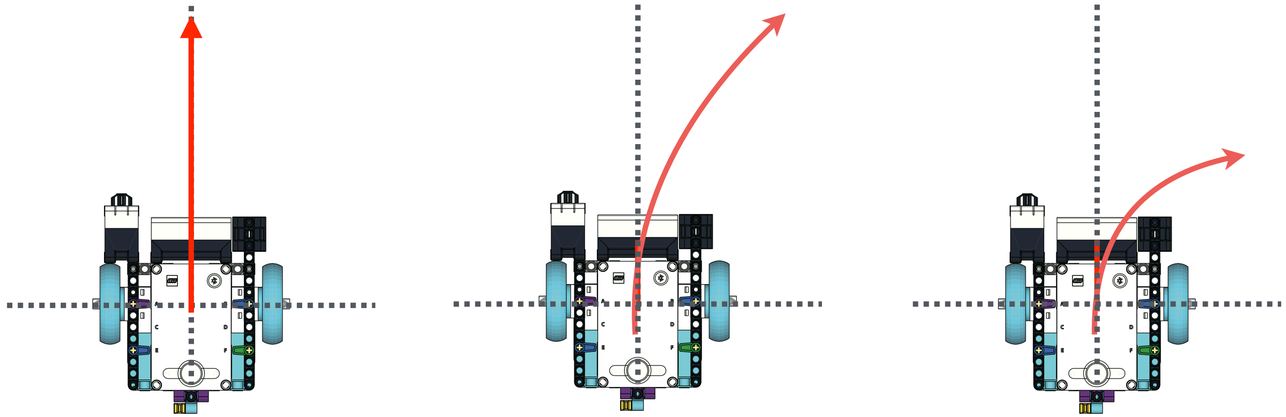
曲線の動きを実現するには

- モータCとDの速度の値のバランスを変えてみよう



曲線の動きを実現するには (SPIKE-WB)

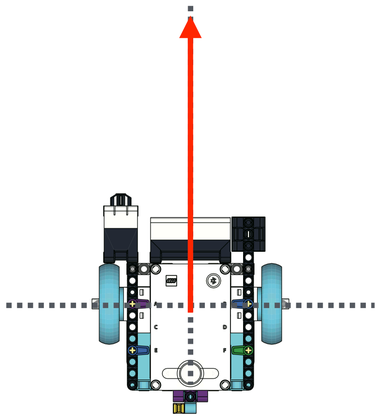
- モータCとDの速度の値を変えてみよう

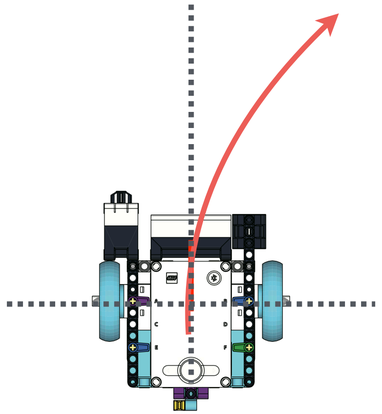


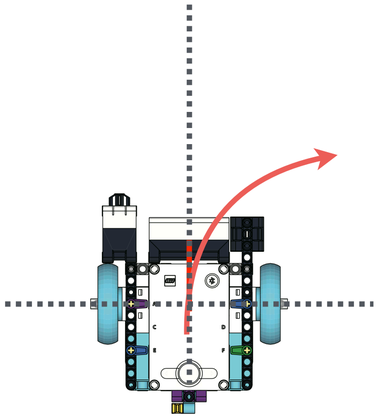
タンクブロック 	C: 50　D: 50	C: 50　D: 30	C: 50　D: 10
ステアリングブロック 	ステアリング: 0	ステアリング: 20	ステアリング: 40

曲線の動きを実現するには (Python)

- モータCとDの速度の値を変えてみよう





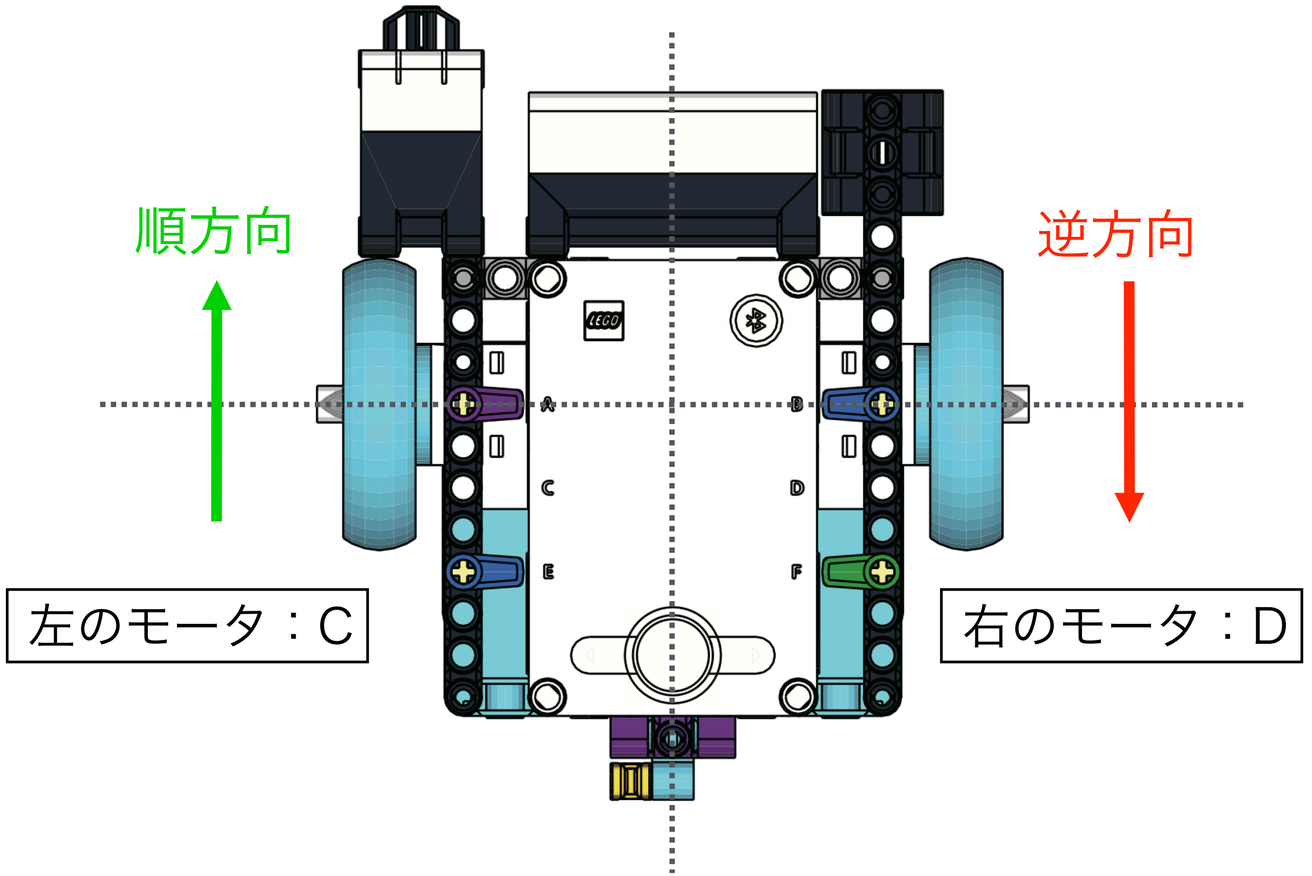


motor_pair.move_tank( )	C: 500　D: 500	C: 500　D: 300	C: 500　D: 100
move( )	steering: 0	steering: 20	steering: 40

■ロボットを旋回させるには(モータ制御2)

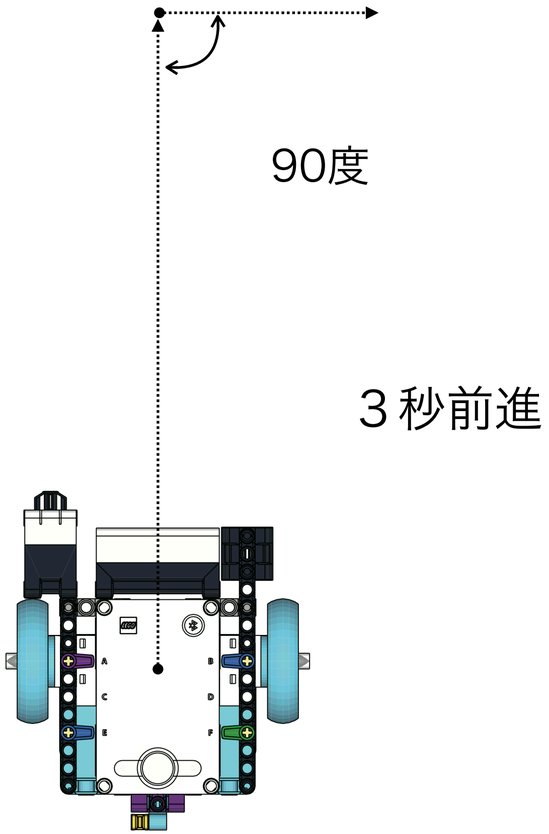
ロボットの旋回

・ ロボットを(その場で)右旋回させるには



ロボットを90度右回転させるには？

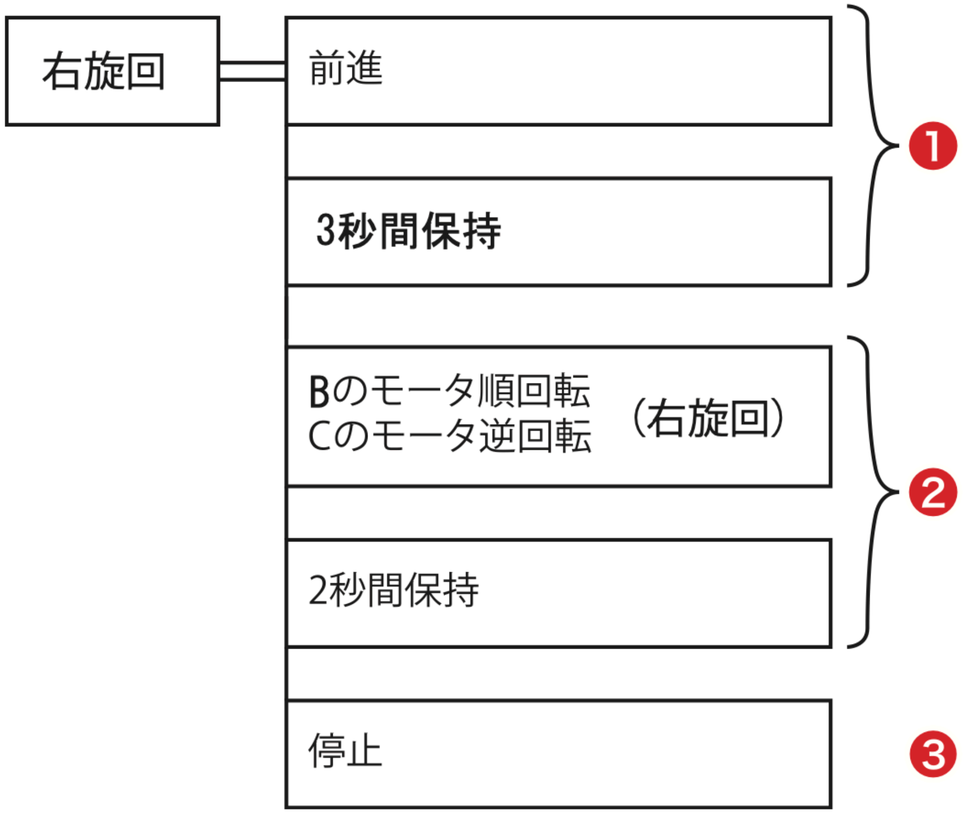
- どのように実現するか考えてみよう！



アルゴリズムを考えよう

- 簡単なアルゴリズム表現を使う
  - 3秒前進して90度旋回して止まろう
    - 90度旋回はやってみないとわからない

      (とりあえず2秒間旋回にしておく)



ロボットの右旋回

・ 左のモータBを順回転、右のモータCを逆回転

p.43：rotation.llsp3

前進 (Cの速度：50、Dの速度：50)

時間：3 秒

右回転 (Cの速度：50、Dの速度：-50)

（順回転）　（逆回転）

時間：2 秒

モータ停止

ロボットの右旋回

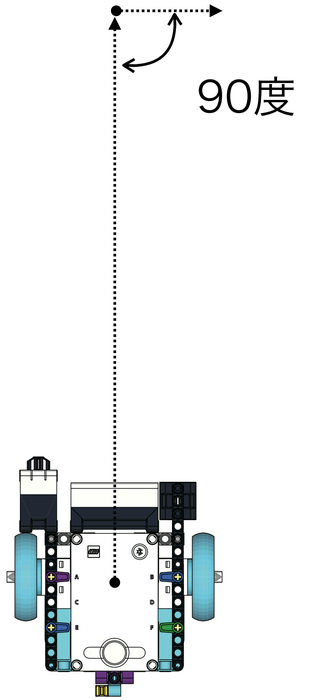
- 左のモータBを順回転、右のモータCを逆回転

p.44：rotation.py

```
from hub import port
import motor_pair
import time

motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)
motor_pair.move_tank(motor_pair.PAIR_1, 500, 500)  # 1
time.sleep_ms(3000)
motor_pair.move_tank(motor_pair.PAIR_1, 500, -500)  # 2
time.sleep_ms(2000)
motor_pair.stop(motor_pair.PAIR_1)  # 3
```

ロボットを90度右旋回させるには？

・ どのように実現するか考えてみよう！



ヒント

1. 時間制御
モータのパワーと持続時間を調整

2. 回転制御
モータの回転数もしくは角度を調整

→複数の問題解決方法があるので、いろいろと試してみよう！(試行錯誤しよう)

ロボットを90度右旋回させるには

1. 時間制御

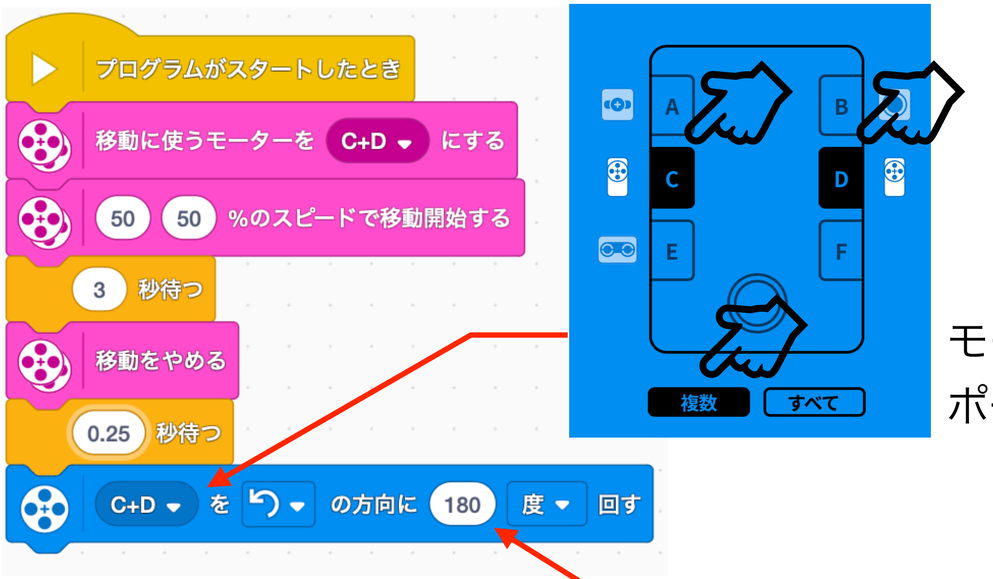
モータのスピードと持続時間を調整

持続時間：0.4秒

速度を変更すると
→持続時間の再調整が必要

2. 回転制御

モータの回転角度を調整



モータブロックで
ポートを複数選択する

回転角：180度

速度を変更しても
→回転角の再調整は必要なし

→ 回転制御の方が便利(同じ動きでロボットを早くしたいとき等)

ロボットを90度右旋回させるには

1. 時間制御

モータのパワーと持続時間を調整

```
motor_pair.move_tank(motor_pair.PAIR_1, 500, -500)
time.sleep_ms(1500)
```

持続時間：1.5秒

速度を変更すると
→持続時間の再調整が必要

2. 回転制御

モータの回転数もしくは角度を調整

```
motor_pair.move_tank_for_degrees(motor_pair.PAIR_1,180, 500, -500)
```

回転角：180度

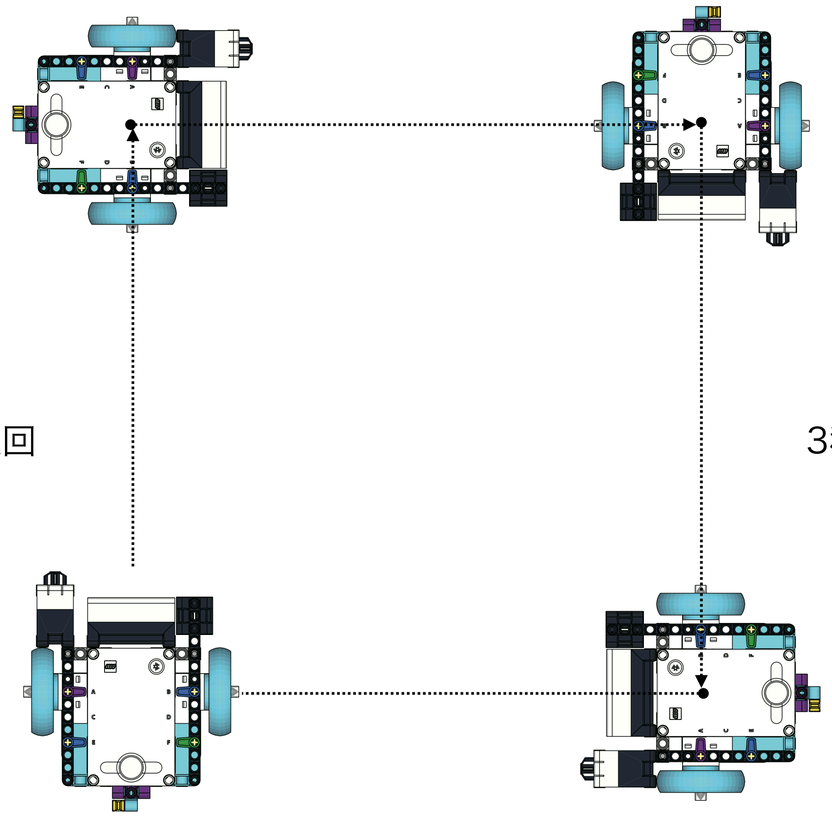
速度を変更しても
→回転角の再調整は必要なし

→ 回転制御の方が便利（同じ動きでロボットを早く動作させたいとき）

ロボットを一周させるには？

2回目

3秒前進→90度旋回



1回目

3秒前進→90度旋回

3回目

3秒前進→90度旋回

4回目

3秒前進→90度旋回

ロボットを一周させるには？

- 3秒前進と90度右旋回を4回繰り返せばよい

1回目
3秒前進→90度旋回

2回目
3秒前進→90度旋回

3回目
3秒前進→90度旋回

4回目
3秒前進→90度旋回

100周するには？
→800個のブロックを並べる必要があり！

→ 繰り返し処理(ループブロック)を利用

ロボットを一周させるには？

- 3秒前進と90度右旋回を4回繰り返せばよい

```
motor_pair.move_tank(motor_pair.PAIR_1, 500, 500)
time.sleep_ms(3000)
motor_pair.move_tank(motor_pair.PAIR_1, 500, -500)
time.sleep_ms(2000)
motor_pair.stop(motor_pair.PAIR_1)
```

1回目
3秒前進→90度旋回

```
motor_pair.move_tank(motor_pair.PAIR_1, 500, 500)
time.sleep_ms(3000)
motor_pair.move_tank(motor_pair.PAIR_1, 500, -500)
time.sleep_ms(2000)
motor_pair.stop(motor_pair.PAIR_1)
```

2回目
3秒前進→90度旋回

```
motor_pair.move_tank(motor_pair.PAIR_1, 500, 500)
time.sleep_ms(3000)
motor_pair.move_tank(motor_pair.PAIR_1, 500, -500)
time.sleep_ms(2000)
motor_pair.stop(motor_pair.PAIR_1)
```

3回目
3秒前進→90度旋回

```
motor_pair.move_tank(motor_pair.PAIR_1, 500, 500)
time.sleep_ms(3000)
motor_pair.move_tank(motor_pair.PAIR_1, 500, -500)
time.sleep_ms(2000)
motor_pair.stop(motor_pair.PAIR_1)
```

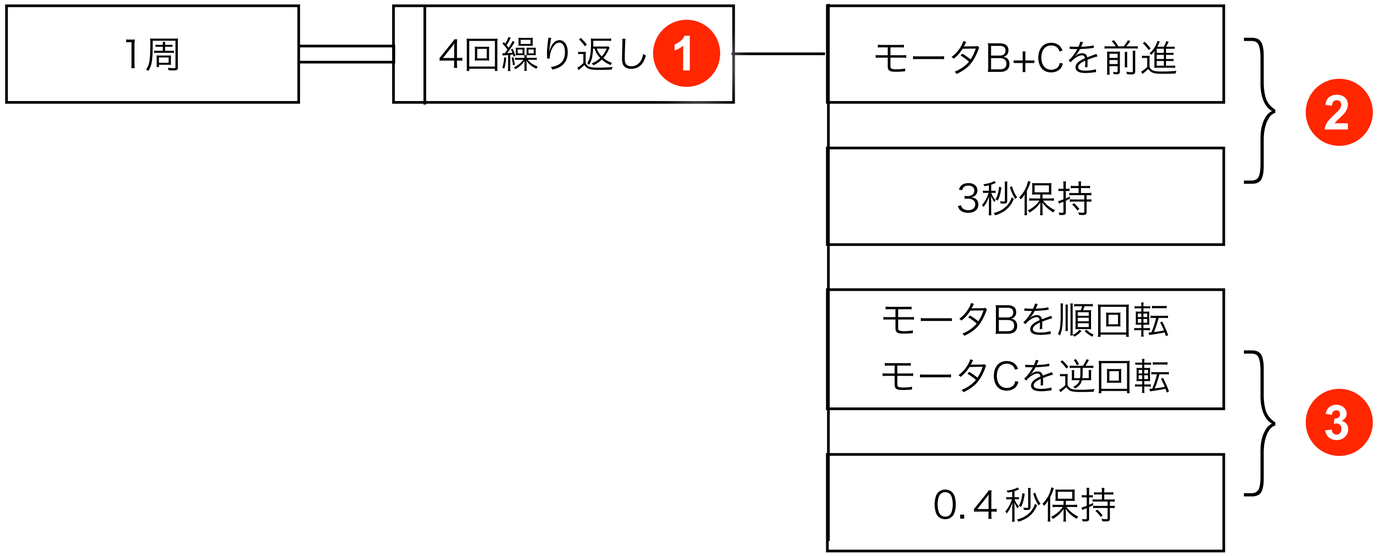
4回目
3秒前進→90度旋回

100周するには？
→ 800個のブロックを並べる必要があり！

→ 繰り返し処理（for文）を利用

一周するプログラムのPAD

- 3秒前進と90度右旋回を4回繰り返す



PADの構成部品：

処理：

選択：

反復：

ロボットを一周させるには

- 3秒前進と90度右旋回を4回繰り返す

p.48：square.llsp3

1. ループ(4回繰り返す)
2. 3秒前進
3. 0.4秒保持
(90度右旋回)

ロボットを一周させるには

- 3秒前進と90度右旋回を4回繰り返す

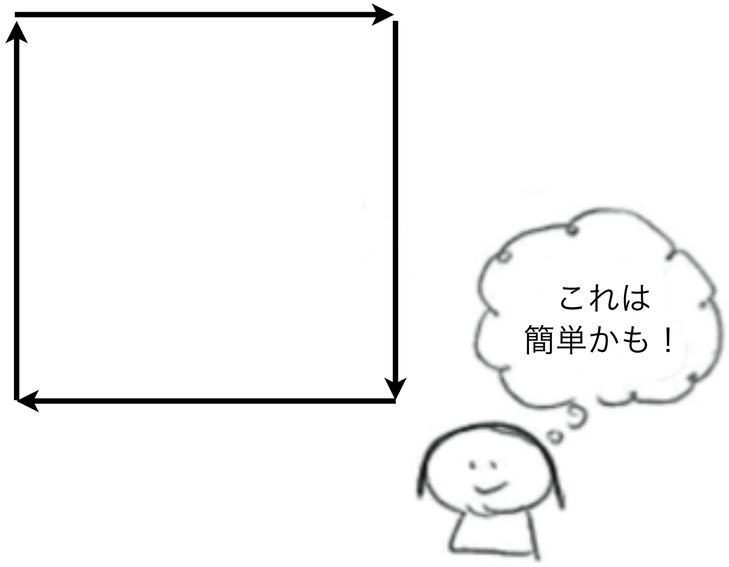
p.49：square.py

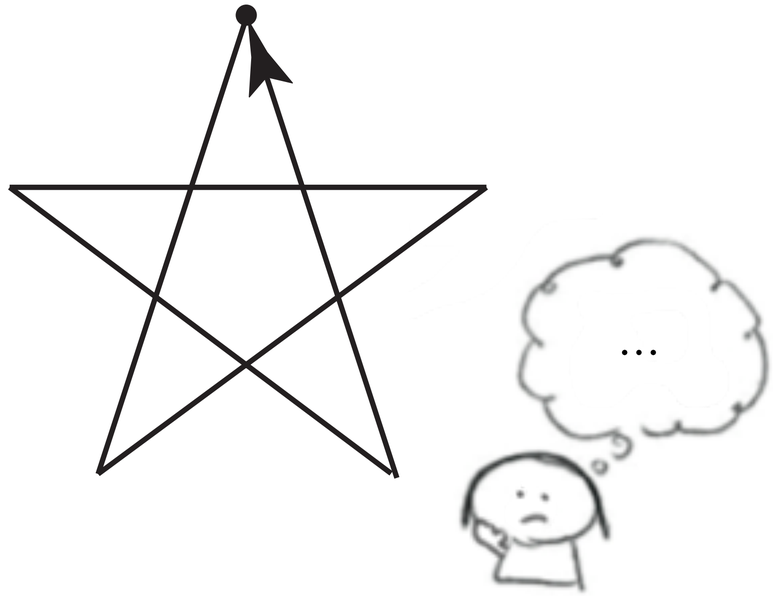
```
from hub import port
import motor_pair
import time

motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)

for i in range(4):
    motor_pair.move_tank(motor_pair.PAIR_1, 500, 500)
    time.sleep_ms(3000)
    motor_pair.move_tank(motor_pair.PAIR_1, 500, -500)
    time.sleep_ms(350)
    motor_pair.stop(motor_pair.PAIR_1)
```

練習問題3-1

- ３秒前進して、方向を変えて自分のところへ戻ってくるプログラムをつくろう

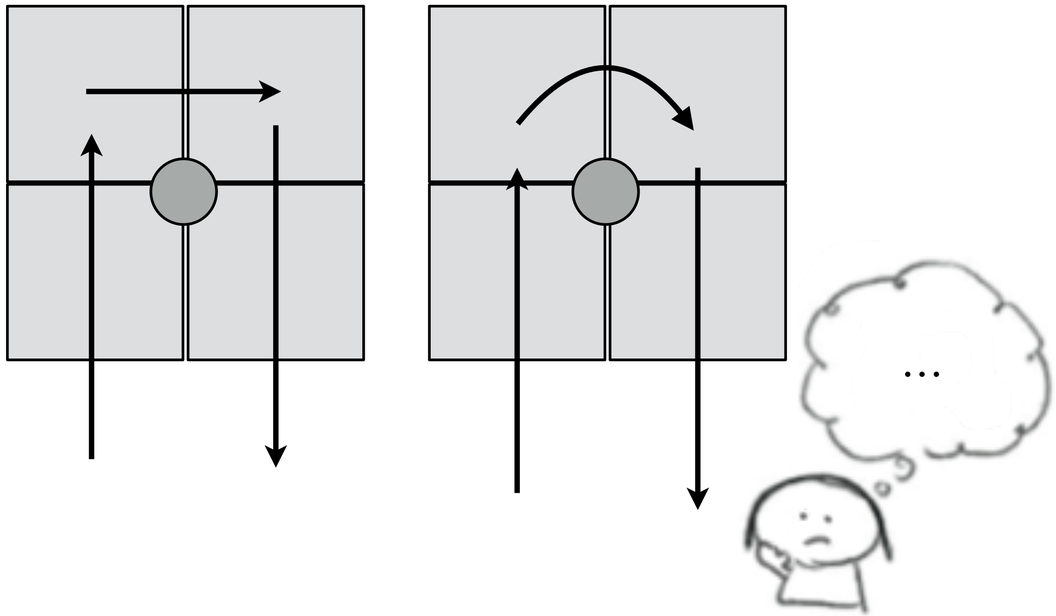
- 思い通りにロボットを動かしてみよう

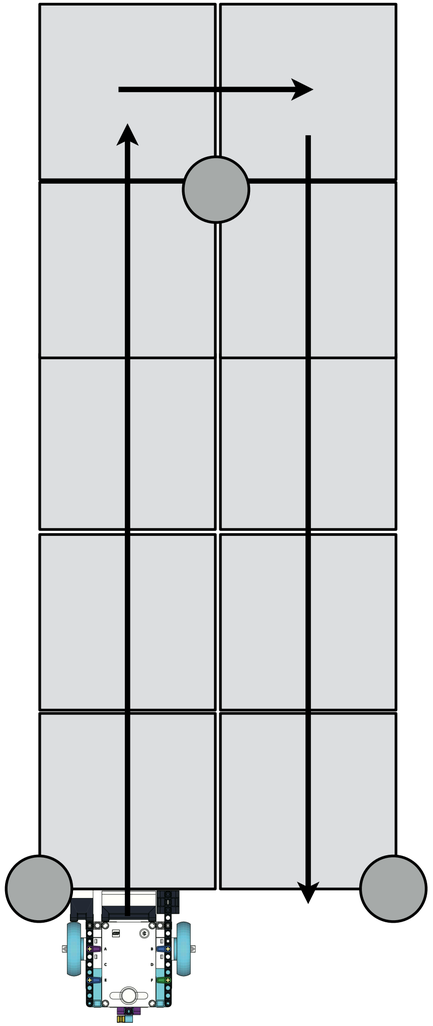




練習問題3-2

- 前進して、方向を変えて自分のところへ戻ってくるプログラムをつくろう
  - ヒント：１秒前進するとどれくらい進む？？？
  - なるべく「早く」戻ってくるようにするには？



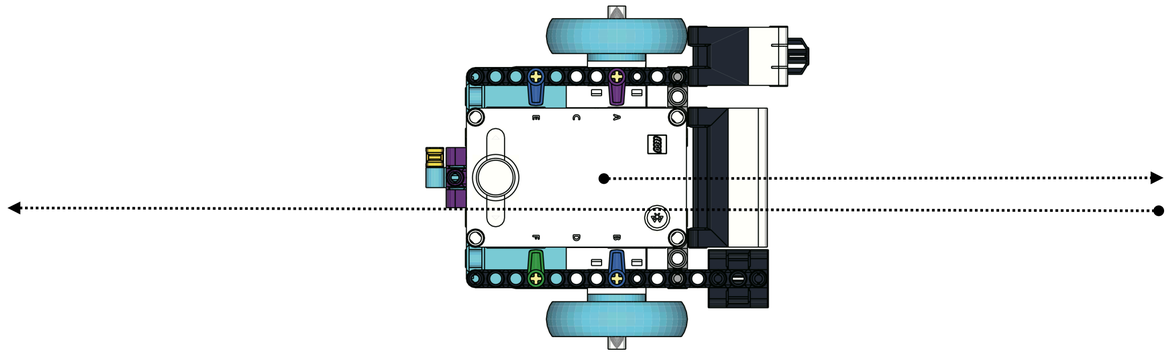


■関数化

動作のプログラムをまとめる

- 同じ処理を何度も使用する → **関数化**

ロボットを1秒前進、その後2秒後退



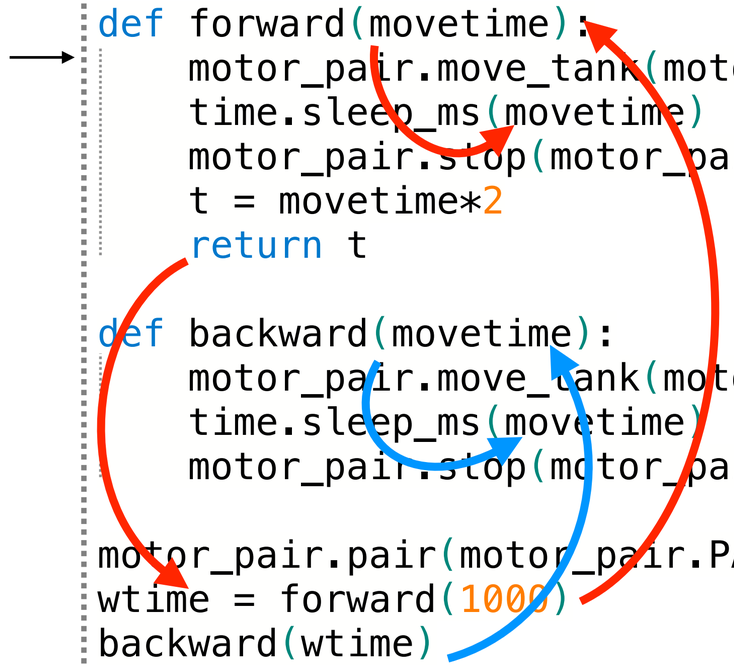
関数化（戻り値も利用）

p.53：function.py

```
from hub import port
import motor_pair
import time

motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)
motor_pair.move_tank(motor_pair.PAIR_1, 500, 500)
time.sleep_ms(1000)
motor_pair.move_tank(motor_pair.PAIR_1, -500, -500)
time.sleep_ms(2000)
motor_pair.stop(motor_pair.PAIR_1)
```

→

```
from hub import port
import motor_pair
import time

def forward(movetime):
    motor_pair.move_tank(motor_pair.PAIR_1, 500, 500)
    time.sleep_ms(movetime)
    motor_pair.stop(motor_pair.PAIR_1)
    t = movetime*2
    return t

def backward(movetime):
    motor_pair.move_tank(motor_pair.PAIR_1, -500, -500)
    time.sleep_ms(movetime)
    motor_pair.stop(motor_pair.PAIR_1)

motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)
wtime = forward(1000)
backward(wtime)
```



前進や旋回を何度も使用する場合は関数化しておくとよい

変数による定義

p.54：variable.py

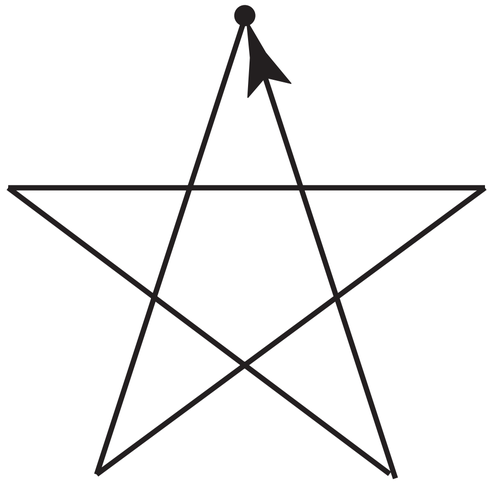
```
from hub import port
import motor_pair
import time

MOVE_TIME = 3000
TURN_TIME = 2000
VEL = 500

motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)
motor_pair.move_tank(motor_pair.PAIR_1, VEL, VEL)
time.sleep_ms(MOVE_TIME)          # 3秒間進んで停止

motor_pair.move_tank(motor_pair.PAIR_1, VEL, -VEL)
time.sleep_ms(TURN_TIME)          # 2秒間回転

motor_pair.stop(motor_pair.PAIR_1) # 停止しなさい
```

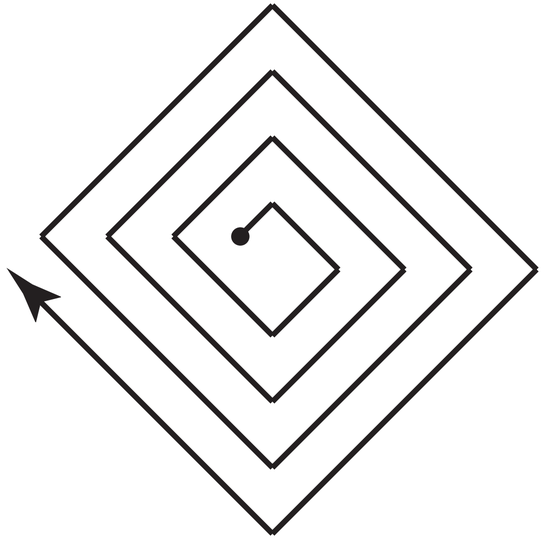
旋回角度（時間制御）を調整するには、TURN_TIMEの値を変るだけで全てに反映

■演算と変数

スパイラル軌跡に挑戦しよう

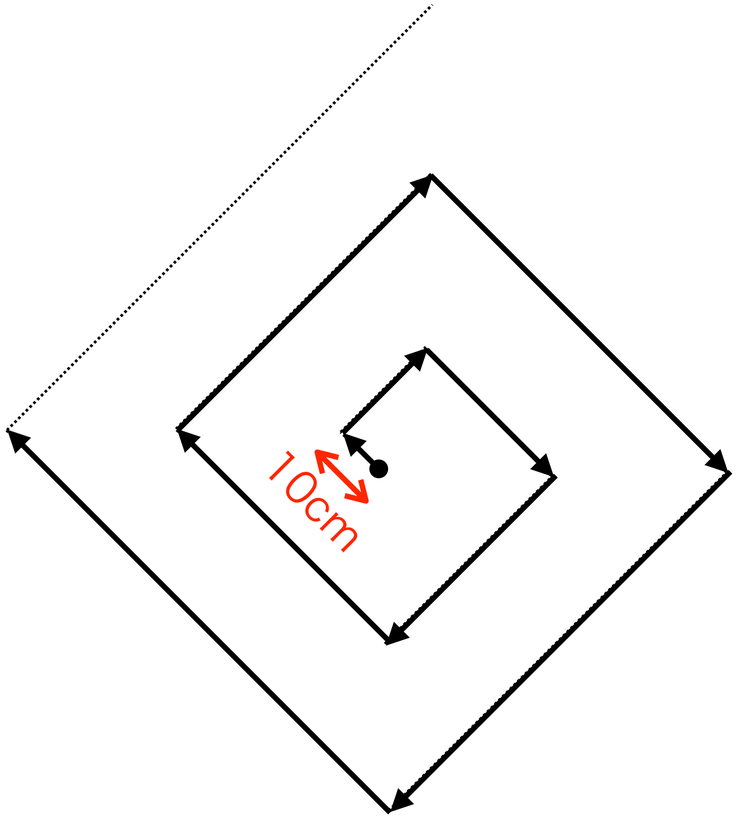
- 星形やスパイラルの軌跡を描くロボットの動きを実現してみよう！





実現したい動きの規則性

・ 繰り返し回数ごとのロボットの動きを表にしてみよう



回数	前進	右旋回
1	10cm	90度
2	20cm	90度
3	30cm	90度
4	40cm	90度
5	50cm	90度
6	60cm	90度
7	70cm	90度
8	80cm	90度
9	90cm	90度

実現したい動きの規則性

- 規則性を数式で表現

回数	前進	右旋回
1	10cm	90度
2	20cm	90度
3	30cm	90度
4	40cm	90度
5	50cm	90度
6	60cm	90度
7	70cm	90度
8	80cm	90度
9	90cm	90度

1

10

回数←1

前進する距離 = 10cm×回数

回数←回数+1

50cm前進するには？

・ モータの回転角を変化させたときの直進距離を測定し、法則をみつける

```
motor_pair.move_tank_for_degrees(motor_pair.PAIR_1, 180, 500, 500)
```

角度：
180～1440度

角度[°]	距離[cm]	1cmあたりの回転角度
180	9.0	20.0
360	17.5	20.6
720	34.5	20.8
1080	52.5	20.5
1440	69.5	20.4

中央値

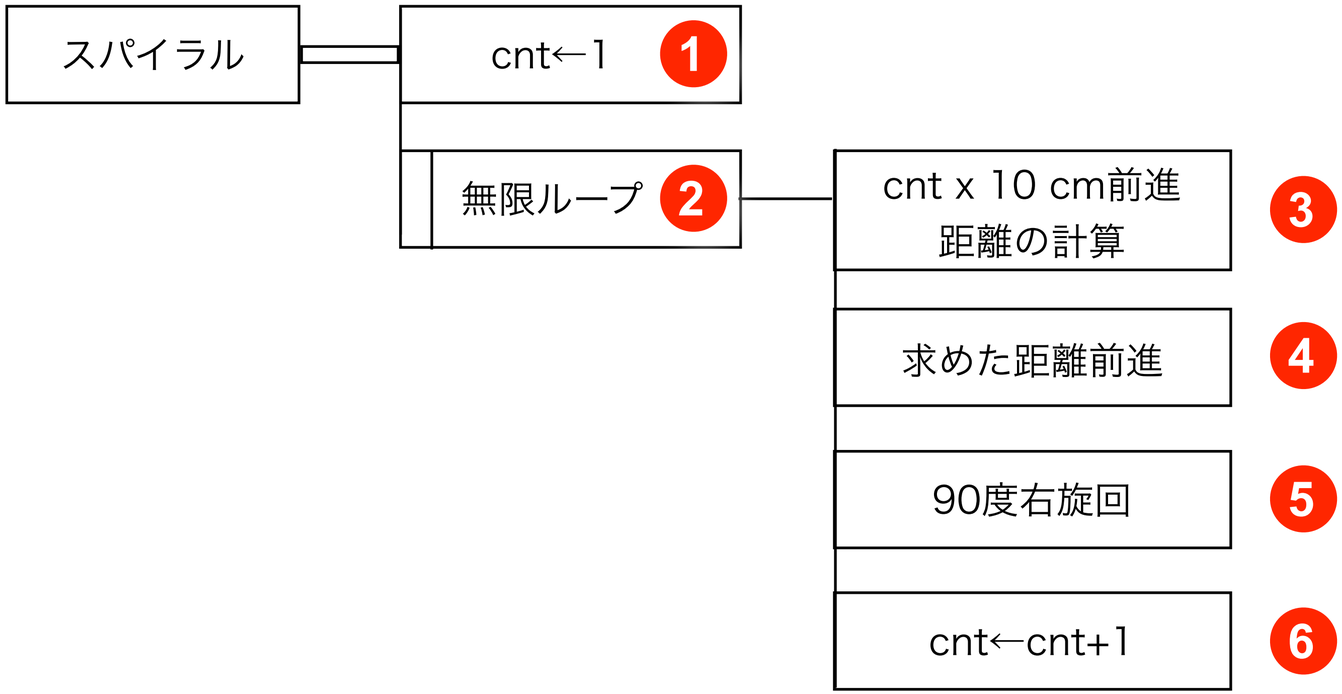
回転角度 = 直進したい距離 × 1cmあたりの回転角度

20.57度

※1cmあたりの回転角度 ＝回転角度[°] / 距離[cm]

スパイラル状の軌跡のPAD

- 繰り返し回数cntと演算によりスパイラル軌跡を実現



スパイラル状の軌跡（Python）

p.58：spiral.py

```
from hub import port
import motor_pair
import time
import runloop

TURN_ANGLE = 180
VEL = 500

async def forward(cm):
    move_angle = int(cm*20.57)
    await motor_pair.move_tank_for_degrees(motor_pair.PAIR_1,move_angle, VEL, VEL)
    motor_pair.stop(motor_pair.PAIR_1)

async def turn_right(turn_angle):
    await motor_pair.move_tank_for_degrees(motor_pair.PAIR_1,turn_angle, VEL, -VEL)
    motor_pair.stop(motor_pair.PAIR_1)

async def main():
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)
    cnt = 1.0
    while True:
        await forward(cnt*10.0)
        await turn_right(TURN_ANGLE)
        cnt=cnt+1

runloop.run(main())
```

応用問題1

- 円形スパイラルを描くようにしてみよう