# EducarSpiconLib

合同会社 S-coRT 技研

2024 年 04 月 12 日

# 目次


改訂履歴 ..... 4
依存ライブラリ ..... 5
構造体 ..... 6
関数 Reference ..... 7
セットアップ関数 ..... 7
ロボットのセットアップ--- void robotSetup(void) ..... 7
車輪用基本関数 ..... 7
左車輪 pwm 出力--- void wh_motorOutputL(int) ..... 7
右車輪 pwm 出力--- void wh_motorOutputR(int) ..... 7
左車輪角度計算--- void wh_calAngleL(void) ..... 7
右車輪角度計算--- void wh_calAngleR(void) ..... 8
左車輪角速度計算--- void wh_calAngVelL(void) ..... 8
右車輪角速度計算--- void wh_calAngVelR(void) ..... 8
左車輪角度取得--- float wh_getAngleL(void) ..... 8
右車輪角度取得--- float wh_getAngleR(void) ..... 8
左車輪角速度取得--- float wh_getAngVelL(void) ..... 9
右車輪角速度取得--- float wh_getAngVelR(void) ..... 9
左車輪目標角速度取得--- float wh_getGoalAngVelL(void) ..... 9
右車輪目標角速度取得--- float wh_getGoalAngVelR(void) ..... 9
車輪用制御関数 ..... 10
左車輪 PID 角度制御--- int wh_pidAngleL(float, float, float, float) ..... 10
右車輪 PID 角度制御--- int wh_pidAngleR(float, float, float, float) ..... 10
左車輪 PI 角速度制御--- int wh_piAngVelL(float, float, float) ..... 10
左車輪 PI 角速度制御--- int wh_piAngVelR(float, float, float) ..... 10
左車輪定速軌道生成--- int wh_calConstVelTrajectoryL(float) ..... 10
右車輪定速軌道生成--- int wh_calConstVelTrajectoryR(float) ..... 10
移動ロボット用運動学関数 ..... 11
ロボット前方速度計算--- void rb_calFowardVelocity() ..... 11
ロボット旋回速度計算--- void rb_calSteeringVelocity() ..... 11
ロボット前方速度取得--- float rb_getFowardVelocity() ..... 11
ロボット旋回速度取得--- float rb_getSteeringVelocity() ..... 11
ロボット前方移動距離計算--- void rb_calFowardDistance() ..... 11
ロボット旋回角度計算--- void rb_calSteeringRadan() ..... 12
ロボット前方移動距離取得--- float rb_getFowardDistance() ..... 12
ロボット旋回角度取得--- float rb_getSteeringRadian() ..... 12

ロボットX方向速度計算--- void rb_calVx() ..... 12
ロボットY方向速度計算--- void rb_calVy() ..... 12
ロボットX方向速度取得--- float rb_getVx() ..... 12
ロボットY方向速度取得--- float rb_getVy() ..... 13
ロボットX位置計算--- void rb_calPx() ..... 13
ロボットY位置計算--- void rb_calPy() ..... 13
ロボットX位置取得--- float rb_getPx() ..... 13
ロボットY位置取得--- float rb_getPy() ..... 13
逆運動学計算--- float rb_calDiffWheelGoalAngularVelocity(float, float) ..... 14

この説明書は、教育用移動ロボット EduCar-spicon 移動ロボット用のライブライである.

# 改訂履歴

・ 2024.04.12: mobileRobotLib beta.0.4

# 依存ライブラリ

1. エンコーダ用

RotaryEncoder class

- getPosition();
- tick();

2. I2C 通信用

Wire class

- begin();
- beginTransmission();
- write();
- endTransmission();
- read();

# 構造体

Wheel 構造体

```
typedef struct __wheel_type{
   float newAng;                 //車輪現在角度(deg)
   float newAngVel;              //車輪現在角速度(deg/s)
   float ang_d;                  //車輪目標角度(deg)
   float angVel_d;               //車輪目標角速度(deg/s)
   float duty;                   //車輪駆動用 duty 比
   float ang_g;                  //車輪最終目標角度(deg)
   float angVel_g;               //車輪最終目標角速度(deg/s)
}Wheel;
```

DiffWheelRobot 構造体

```
typedef struct __diffwheelrobot_type{
   float pf;                     //ロボット前方移動距離(m)
   float vf;                     //ロボット前方移動速度(m/s)
   float px;                     //ロボット X 位置(m)
   float vx;                     //ロボット X 方向速度(m/s)
   float py;                     //ロボット y 位置(m)
   float vy;                     //ロボット y 方向速度(m/s)
   float phi;                    //ロボット旋回角度(rad)
   float dphi;                   //ロボット旋回角速度(rad/s)
   float vf_d;                   //ロボット目標前方移動速度(m/s)
   float dphi_d;                 //ロボット目標旋回速度(rad/s)
}DiffWheelRobot;
```

# 関数 Reference

## セットアップ関数

### ロボットのセットアップ--- void robotSetup(void)

説　明：　ロボットの各種セットアップ関数．PWM 用の pin モード設定，エンコーダ用の設定，慣性センサ MPU6050 の設定を行う．ArduinoIDE のプログラムの void setup()内に記述する．

引　数：　なし

返り値：　なし

使い方：

```
robotSetup();
```

## 車輪用基本関数

### 左車輪 pwm 出力--- void wh_motorOutputL(int)

説　明：　制御入力(duty 比)を受けて左側車輪のモータドライバに pwm を出力する．内部で-100～100%に制限される．

引　数：　int 型の duty 比

返り値：　なし

使い方：

```
int mDuty = 50;
wh_motorOutputL(my_lw.duty);
```

### 右車輪 pwm 出力--- void wh_motorOutputR(int)

説　明：　制御入力(duty 比)を受けて右側車輪のモータドライバに pwm を出力する．内部で-100～100%に制限される．

引　数：　int 型の duty 比

返り値：　なし

使い方：

```
int mDuty = 50;
wh_motorOutputR(my_rw.duty);
```

### 左車輪角度計算--- void wh_calAngleL(void)

説　明：　左車輪の角度(degree)を計算して float 型の lwh.newAng 変数に格納する．

引　数：　なし

返り値：　なし

使い方：

```
wh_calAngleL();
```

### 右車輪角度計算--- void wh_calAngleR(void)

説　明：　右車輪の角度(degree)を計算して float 型の rwh.newAng 変数に格納する.

引　数：　なし

返り値：　なし

使い方：

```
wh_calAngleR();
```

### 左車輪角速度計算--- void wh_calAngVelL(void)

説　明：　左車輪の角速度(deg/s)を計算して float 型の lwh.newAngVel 変数に格納する.

引　数：　なし

返り値：　なし

使い方：

```
wh_calAngVelL();
```

### 右車輪角速度計算--- void wh_calAngVelR(void)

説　明：　右車輪の角速度(deg/s)を計算して float 型の lwh.newAngVel 変数に格納する.

引　数：　なし

返り値：　なし

使い方：

```
Wh_calAngVelR();
```

### 左車輪角度取得--- float wh_getAngleL(void)

説　明：　左車輪の角度(degree)を取得する．wh_calAngleL()を先に実行して角度を更新しておく必要がある.

引　数：　なし

返り値：　float 型の角度(degree)

使い方：

```
float lw_newAng;
wh_calAngleL();
my_lw.newAng = wh_getAngleL();
```

### 右車輪角度取得--- float wh_getAngleR(void)

説　明：　右車輪の角度(degree)を取得する．wh_calAngleR()を先に実行して角度を更新しておく必要がある.

引　数：　なし

返り値：　float 型の角度(degree)

使い方：

```
float rw_newAng;
```

```
wh_calAngleR();
my_rw.newAng = wh_getAngleR();
```

### 左車輪角速度取得--- float wh_getAngVelL(void)

説　明：　左車輪の角速度(deg/s)を取得する．wh_calAngVelL()を先に実行して角度を更新しておく必要がある．

引　数：　なし

返り値：　float 型の角速度(deg/s)

使い方：

```
float lw_newAngVel;
wh_calAngVelL();
my_lw.newAngVel = wh_getAngVelL();
```

### 右車輪角速度取得--- float wh_getAngVelR(void)

説　明：　右車輪の角度(degree)を取得する．wh_calAngleR()を先に実行して角度を更新しておく必要がある．

引　数：　なし

返り値：　float 型の角速度(deg/s)

使い方：

```
wh_calAngVelR();
my_rw.newAngVel = wh_getAngVelR();
```

### 左車輪目標角速度取得--- float wh_getGoalAngVelL(void)

説　明：　左車輪の目標角速度(rad/s)を取得する
rb_calDiffWheelGoalAngularVelocity()の結果をリターンする．

引　数：　なし

返り値：　float 型の目標角速度(deg/s) lwh.angvel_d

使い方：

```
rb_calDiffWheelGoalAngularVelocity(my_rb.vf_d, my_rb.dphi_d);
my_lw.angVeld_d = wh_getGoalAngVelL()*Rad2Deg;
```

### 右車輪目標角速度取得--- float wh_getGoalAngVelR(void)

説　明：　右車輪の目標角速度(rad/s)を取得する
rb_calDiffWheelGoalAngularVelocity()の結果をリターンする

引　数：　なし

返り値：　float 型の目標角速度(deg/s) rwh.angvel_d

使い方：

```
rb_calDiffWheelGoalAngularVelocity(my_rb.vf_d, my_rb.dphi_d);
my_rw.angVeld_d = wh_getGoalAngVelR()*Rad2Deg;
```

## 車輪用制御関数

### 左車輪 PID 角度制御--- int wh_pidAngleL(float, float, float, float)

説　明：　左車輪を目標角度に追従するよう PID 制御する.

引　数：　(目標角度、P ゲイン、D ゲイン、I ゲイン )

返り値：　制御入力用 duty 値

使い方：

```
my_lw.duty = wh_pidAngleL(my_lw.ang_d, 1, 0.1, 0);
```

### 右車輪 PID 角度制御--- int wh_pidAngleR(float, float, float, float)

説　明：　右車輪を目標角度に追従するよう PID 制御する.

引　数：　(目標角度、P ゲイン、D ゲイン、I ゲイン )

返り値：　制御入力用 duty 値

使い方：

```
my_rw.duty = wh_pidAngleR(my_rw.ang_d, 1, 0.1, 0);
```

### 左車輪 PI 角速度制御--- int wh_piAngVelL(float, float, float)

説　明：　左車輪を目標角速度に追従するよう PI 制御する.

引　数：　(目標角度、P ゲイン、I ゲイン )

返り値：　制御入力用 duty 値

使い方：

```
my_lw.duty = wh_piAngVelL(my_lw.angVel_d, 1, 0.1);
```

### 左車輪 PI 角速度制御--- int wh_piAngVelR(float, float, float)

説　明：　右車輪を目標角速度に追従するよう PI 制御する.

引　数：　(目標角度、P ゲイン、I ゲイン )

返り値：　制御入力用 duty 値

使い方：

```
my_rw.duty = wh_pidAngVelR(my_rw.angVel_d, 1, 0.1);
```

### 左車輪定速軌道生成--- int wh_calConstVelTrajectoryL(float)

説　明：　左車輪を定速回転させるための角度を計算する.

引　数：　最終目標角速度 vel_f

返り値：　目標角度 angle_d

使い方：

```
my_lw.ang_d = wh_calConstVelTrajectoryL(my_lw.vel_f);
```

### 右車輪定速軌道生成--- int wh_calConstVelTrajectoryR(float)

説　明：　右車輪を定速回転させるための角度を計算する.

引　数：　最終目標角速度 vel_f

返り値：　目標角度 angle_d

使い方：

```
my_rw.ang_d = wh_calConstVelTrajectoryL(my_rw.vel_f);
```

## 移動ロボット用運動学関数

### ロボット前方速度計算--- void rb_calFowardVelocity()

説　明：　左右車輪の角速度から前方速度(m/s)を計算する.

引　数：　なし

返り値：　なし

使い方：

```
rb_calFowardVeloity();
```

### ロボット旋回速度計算--- void rb_calSteeringVelocity()

説　明：　左右車輪の角速度から旋回角速度(rad/s)を計算する.

引　数：　なし

返り値：　なし

使い方：

```
rb_calSteeringVeloity();
```

### ロボット前方速度取得--- float rb_getFowardVelocity()

説　明：　移動ロボットの前方速(m/s)度を取得する.

引　数：　なし

返り値：　ロボットの前方速度 dwr.vf

使い方：

```
rb_calFowardVeloity();
my_rb.vf = rb_getFowardVelocity();
```

### ロボット旋回速度取得--- float rb_getSteeringVelocity()

説　明：　左右車輪の角速度から旋回角速度(rad/s)を計算する.

引　数：　なし

返り値：　ロボットの旋回速度 dwr.dphi

使い方：

```
rb_calSteeringVeloity();
my_rb.dphi = rb_getSteeringVelocity();
```

### ロボット前方移動距離計算--- void rb_calFowardDistance()

説　明：　前方速度と旋回速度から前方移動距離(m)を計算する.

引　数：　なし

返り値：　なし

使い方：

```
rb_calFowardDistance();
```

### ロボット旋回角度計算--- void rb_calSteeringRadan()

説　明：　前方速度と旋回速度から旋回角度(rad)を計算する.

引　数：　なし

返り値：　なし

使い方：

```
rb_calSteeringRadian();
```

### ロボット前方移動距離取得--- float rb_getFowardDistance()

説　明：　移動ロボットの前方移動距離(m)を取得する.

引　数：　なし

返り値：　ロボットの前方速度 dwr.pf

使い方：

```
rb_calFowardDistance();
my_rb.pf = rb_getFowardDistance();
```

### ロボット旋回角度取得--- float rb_getSteeringRadian()

説　明：　左右車輪の角速度から旋回角速度を計算する.

引　数：　なし

返り値：　ロボットの旋回速度 dwr.phi

使い方：

```
rb_calSteeringRadian();
my_rb.phi = rb_getSteeringRadian();
```

### ロボット X 方向速度計算--- void rb_calVx()

説　明：　前方速度と旋回速度から X 方向速度(m/s)を計算する.

引　数：　なし

返り値：　なし

使い方：

```
rb_calVx();
```

### ロボット Y 方向速度計算--- void rb_calVy()

説　明：　前方速度と旋回速度から Y 方向速度(m/s)を計算する.

引　数：　なし

返り値：　なし

使い方：

```
rb_calVy();
```

### ロボット X 方向速度取得--- float rb_getVx()

説　明：　X 方向速度(m/s)を取得する.

引　数：　なし

返り値：　ロボットの前方速度 dwr.vx

使い方：

```
rb_calVx();
my_rb.vx = rb_getVx();
```

### ロボット Y 方向速度取得--- float rb_getVy()

説　明：　Y 方向速度(m/s)を取得する.

引　数：　なし

返り値：　ロボットの旋回速度 dwr.vy

使い方：

```
rb_calVy();
my_rb.vy = rb_getVy();
```

### ロボット X 位置計算--- void rb_calPx()

説　明：　原点座標系に対する X 位置(m)を計算する.

引　数：　なし

返り値：　なし

使い方：

```
rb_calPx();
```

### ロボット Y 位置計算--- void rb_calPy()

説　明：　原点座標系に対する Y 移動距離(m)を計算する.

引　数：　なし

返り値：　なし

使い方：

```
rb_calPy();
```

### ロボット X 位置取得--- float rb_getPx()

説　明：　原点座標系に対する X 位置(m)を取得する.

引　数：　なし

返り値：　ロボットの前方速度 dwr.px

使い方：

```
rb_calPx();
my_rb.px = rb_getPx();
```

### ロボット Y 位置取得--- float rb_getPy()

説　明：　原点座標系に対する Y 位置(m)を取得する.

引　数：　なし

返り値：　ロボットの旋回速度 dwr.py

使い方：

```
rb_calPy();
my_rb.py = rb_getPy();
```

### 逆運動学計算--- float rb_calDiffWheelGoalAngularVelocity(float, float)

説　明：　目標前方速度(m/s)と目標旋回速度(rad/s)から両車輪の目標角速度(rad/s)を計算する

引　数：　(目標前方速度 vf_d, 目標旋回速度 omega_d)

返り値：　なし

使い方：

```
rb_calDiffWheelGoalAngularVelocity(my_rb.vf_d, my_rb.dphi_d);
my_lw.angVel_d = wh_getGoalangVelL()*Rad2Deg;
my_rw.angVel_d = wh_getGoalangVelR()*Rad2Deg;
```