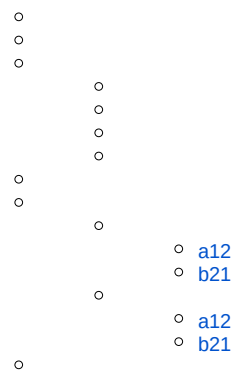
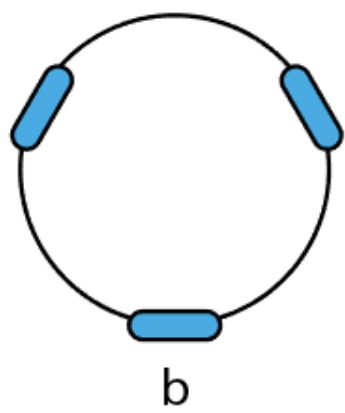
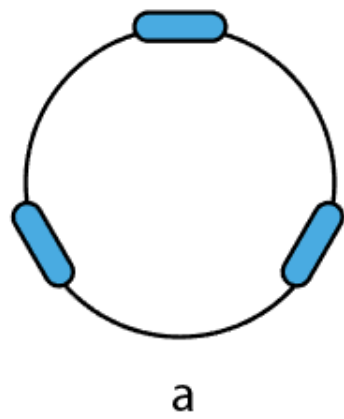


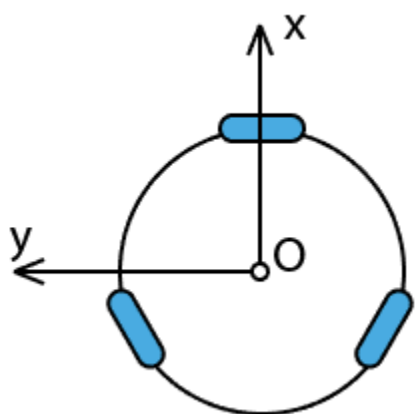
KBSW180148 Instruction for Integrating SLAMWARE Solution in Tri-omini-wheeled Base



SLAMWARE

212a21b

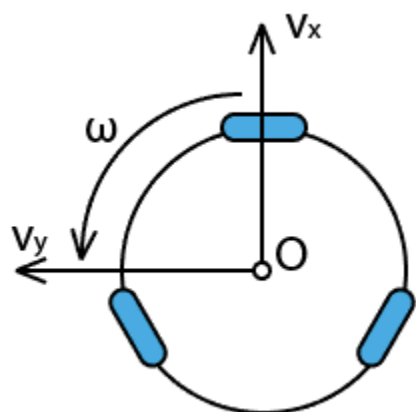




↑ 机器人前进方向

xyO

SLAMWARE Core

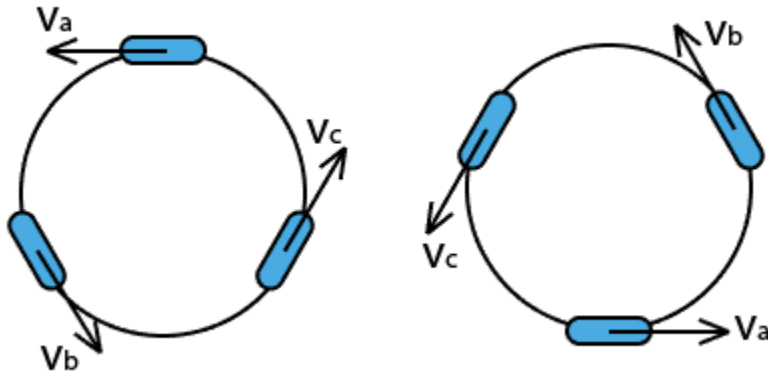


↑ 机器人前进方向

$v_x v_y$

$v_x v_y$

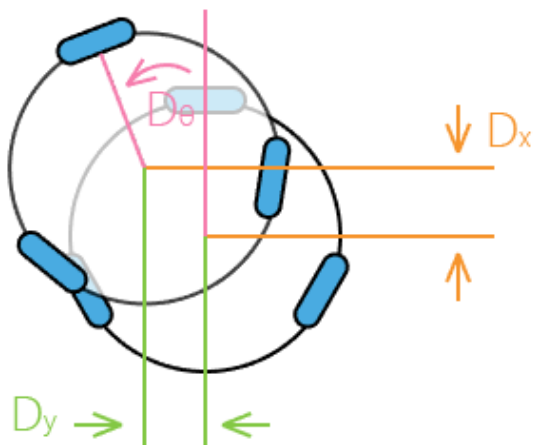




VaVbVc

SLAMWARE

DxDyD



DxDyD

LaLbLc

SLAMWARECORECB_CMD_GET_BASE_MOTOR_DATA SLAMWARECORECB_CMD_SET_BASE_MOTOR

SLAMWARECORECB_CMD_SET_V_AND_GET_DEADRECKON0x41

base_set_velocity_request_t

```
typedef struct _base_set_velocity_request
{
    _s32 velocity_x_q16;
    _s32 velocity_y_q16;
    _s32 angular_velocity_q16;
} __attribute__((packed)) base_set_velocity_request_t;
```

```
base_set_velocity_request_t req;

float vx = req.velocity_x_q16 / 65536.f;
float vy = req.velocity_y_q16 / 65536.f;
float omega = req.angular_velocity_q16 / 65536.f;
```

(dx ,dymm, dtheta)

base_deadreckon_response_t

```
typedef struct _base_deadreckon_response
{
    _s32 base_dx_mm_q16;
    _s32 base_dy_mm_q16;
    _s32 base_dtheta_degree_q16;
    _u8 status_bitmap;
} __attribute__((packed)) base_deadreckon_response_t;
```

```
#ifndef PIf
#    define PIf 3.14159265f
#endif

float dx, dy, dtheta;

base_deadreckon_response_t resp;

resp.base_dx_mm_q16 = (_s32)(dx * 1000 * 65536);
resp.base_dy_mm_q16 = (_s32)(dy * 1000 * 65536);
resp.base_dtheta_degree_q16 = (_s32)(dtheta * 180 / PIf * 65536);
resp.status_bitmap = 0; // 0
```



R

SLAMWARE Core

a12

$$V_a = V_y + R *$$

$$V_b = -V_x * \cos(30^\circ) - V_y * \sin(30^\circ) + R *$$

$$V_c = V_x * \cos(30^\circ) - V_y * \sin(30^\circ) + R *$$

b21

$$V_a = -V_y + R *$$

$$V_b = V_x * \cos(30^\circ) + V_y * \sin(30^\circ) + R *$$

$$V_c = -V_x * \cos(30^\circ) + V_y * \sin(30^\circ) + R *$$

DaDbDc

a12

$$D = (Da + Db + Dc) / R / 3$$

$$Dx = (- Db * \cos(30^\circ) + Dc / \cos(30^\circ)) / 2$$

$$Dy = (- Db * \sin(30^\circ) - Dc / \sin(30^\circ) + Da) / 3$$

b21

$$D = (Da + Db + Dc) / R / 3$$

$$Dx = (Db / \cos(30^\circ) - Dc / \cos(30^\circ)) / 2$$

$$Dy = (-Da + Db / \sin(30^\circ) + Dc / \sin(30^\circ)) / 3$$

```
// a

#ifndef PIf
#    define PIf 3.14159265f
#endif

#define COS30f    0.86602540f
#define SIN30f    0.5f
#define R        0.18f // robot radius: 18cm

static float wheel_odometer[3];

static void on_set_v_and_get_deadreckon(const base_set_velocity_request_t& req, base_deadreckon_response_t& resp)
{
    // calculate response
    float current_wheel_odometer[3];
    motion_get_wheel_odometer(current_wheel_odometer);

    float da = current_wheel_odometer[0] - wheel_odometer[0];
    float db = current_wheel_odometer[1] - wheel_odometer[1];
    float dc = current_wheel_odometer[2] - wheel_odometer[2];

    memcpy(wheel_odometer, current_wheel_odometer, sizeof(current_wheel_odometer));

    float dx = (-db / COS30f + dc / COS30f) / 2
    float dy = (da - db / SIN30f - dc / SIN30f) / 3
    float dtheta = (da + db + dc) / R / 3

    resp.base_dx_mm_q16 = (_s32)(dx * 1000 * 65536);
    resp.base_dy_mm_q16 = (_s32)(dy * 1000 * 65536);
    resp.base_dtheta_degree_q16 = (_s32)(dtheta * 180 / PIf * 65536);
    resp.status_bitmap = 0; // 0

    // calculate wheel speed
    float vx = req.velocity_x_q16 / 65536.f;
    float vy = req.velocity_y_q16 / 65536.f;
    float omega = req.angular_velocity_q16 / 65536.f;

    float va = vy + R * omega;
    float vb = - vx * COS30f - vy * SIN30f + R * omega;
    float vc = vx * COS30f - vy * SIN30f + R * omega;

    motion_set_wheel_speed(va, vb, vc);
}
```

