PID control robot movement

1. Experimental purpose

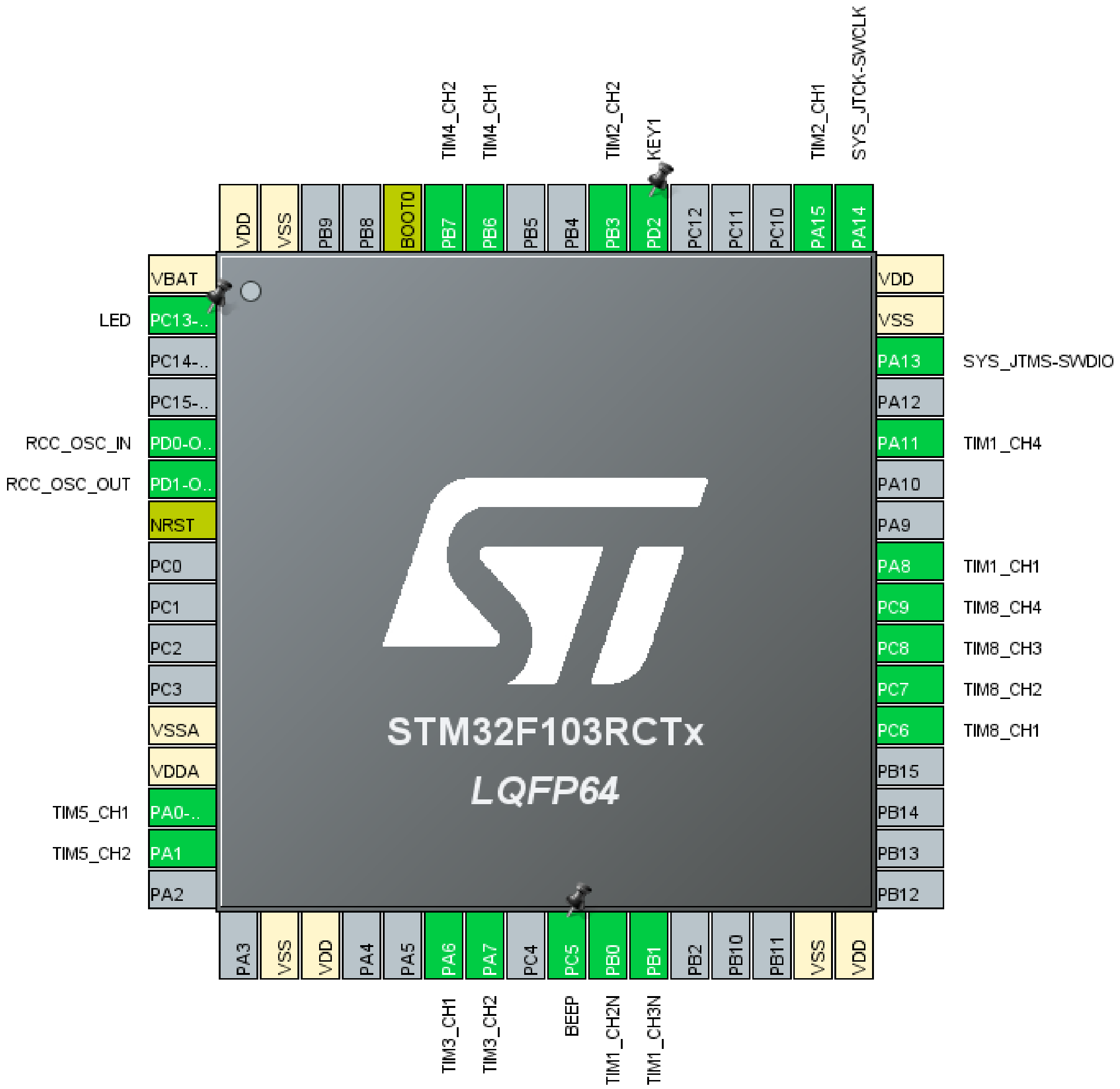
Control the movement of the robot, and control the running speed of the robot through the PID algorithm.

Since different models require different drive codes, here we only take the Mecanum wheel trolley as an example.

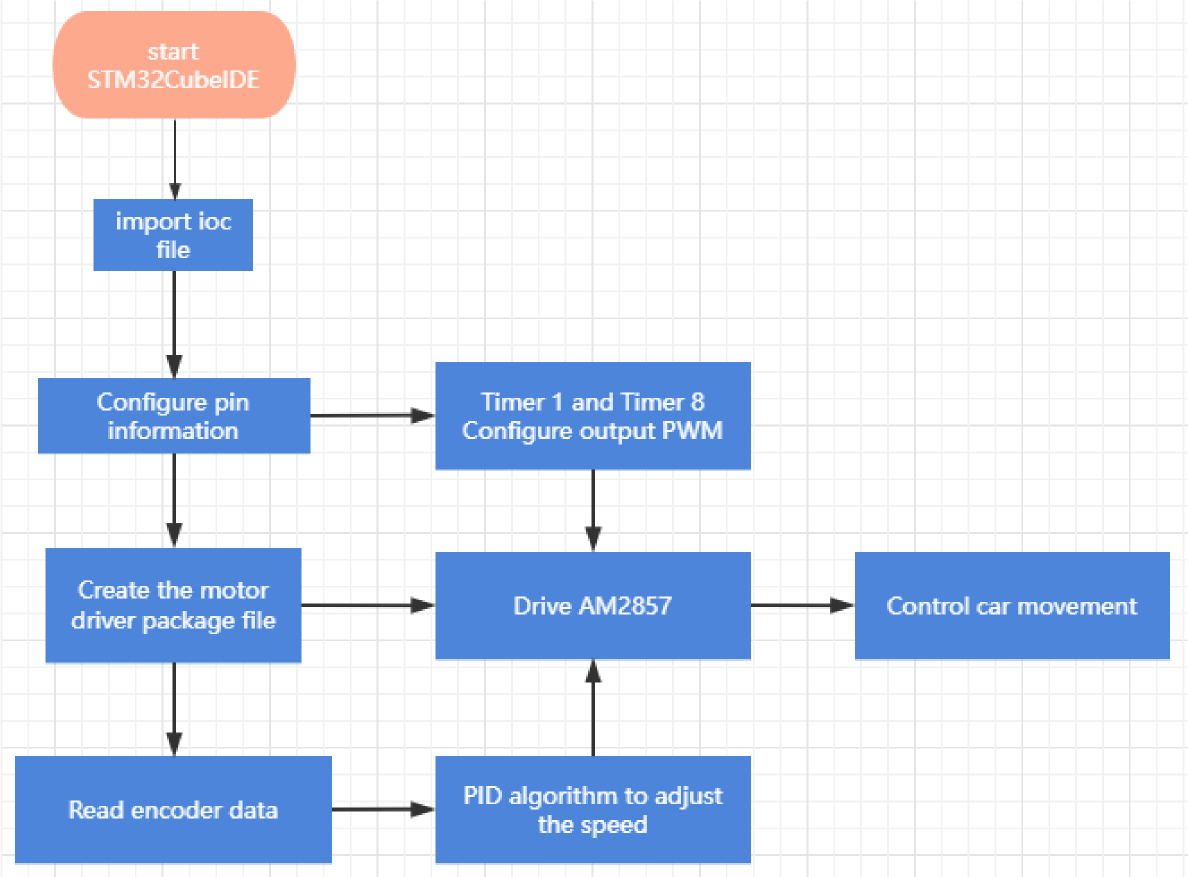
2. configuration pin information

1. Import the ioc file from the Encoder project and name it Car_Motion.

No additional content is required, and the final chip configuration pins are shown in the following figure:

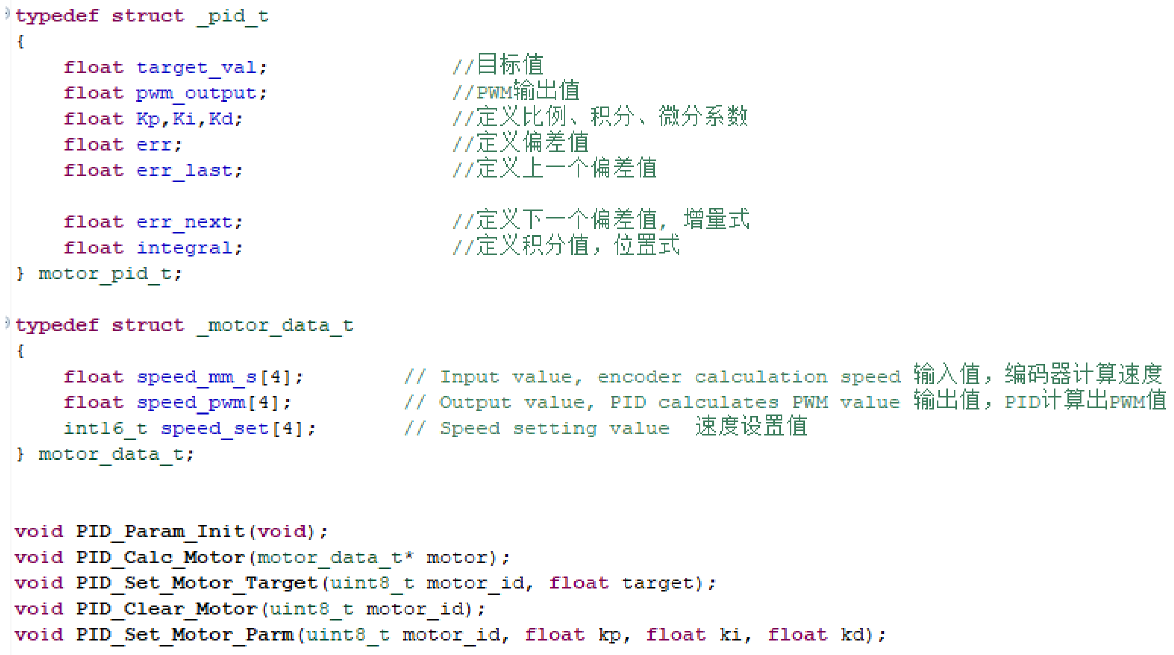


3. Analysis of the experimental flow chart


start
STM32CubeIDE
import ioc file
Configure pin information
Timer 1 and Timer 8 Configure output PWM
Create the motor driver package file
Drive AM2857
Control car movement
Read encoder data
PID algorithm to adjust the speed

4. core code explanation

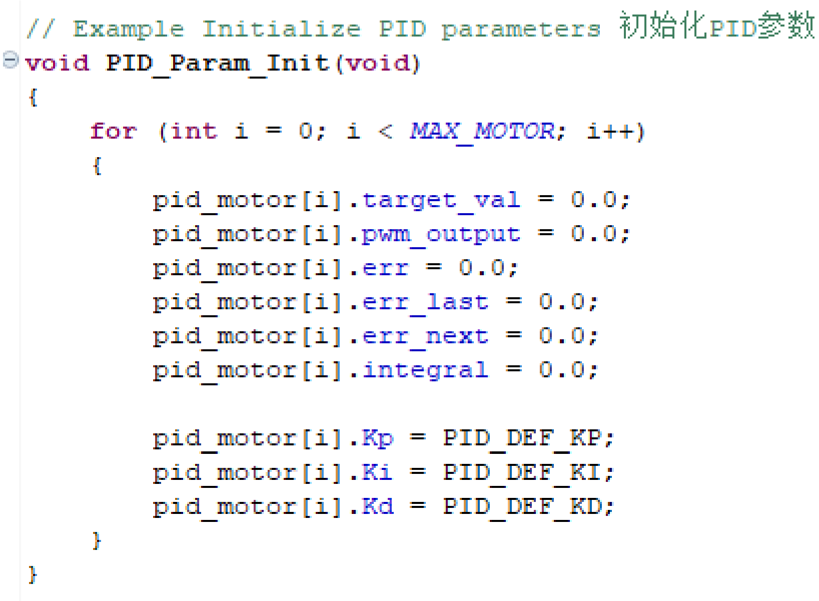
1. Create new bsp_pid.h and bsp_pid.c, and add the following content to bsp_pid.h:



```
typedef struct _pid_t
{
    float target_val;               //目标值
    float pwm_output;               //PWM输出值
    float Kp,Ki,Kd;                 //定义比例、积分、微分系数
    float err;                      //定义偏差值
    float err_last;                 //定义上一个偏差值

    float err_next;                 //定义下一个偏差值, 增量式
    float integral;                 //定义积分值，位置式
} motor_pid_t;

typedef struct _motor_data_t
{
    float speed_mm_s[4];        // Input value, encoder calculation speed 输入值，编码器计算速度
    float speed_pwm[4];         // Output value, PID calculates PWM value 输出值，PID计算出PWM值
    int16_t speed_set[4];       // Speed setting value  速度设置值
} motor_data_t;


void PID_Param_Init(void);
void PID_Calc_Motor(motor_data_t* motor);
void PID_Set_Motor_Target(uint8_t motor_id, float target);
void PID_Clear_Motor(uint8_t motor_id);
void PID_Set_Motor_Parm(uint8_t motor_id, float kp, float ki, float kd);
```

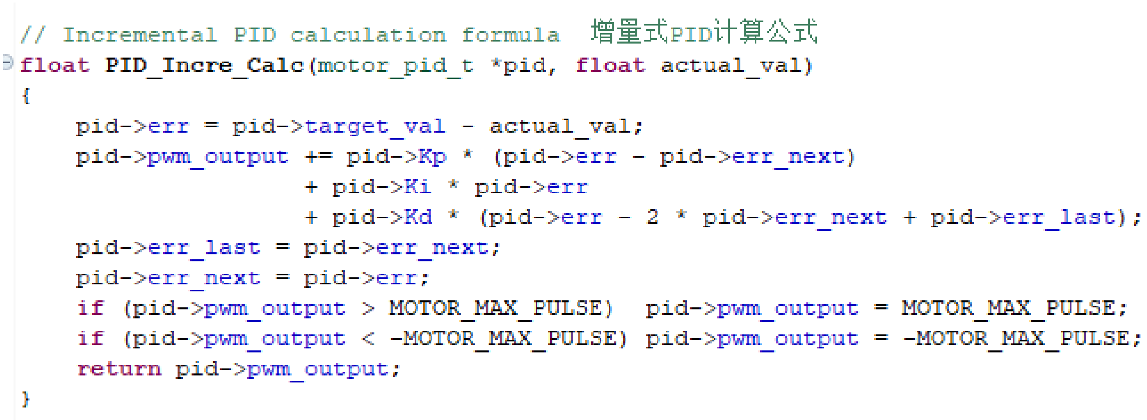
2. Create the following content in the bsp_pid.c file:

Initialize PID parameters:



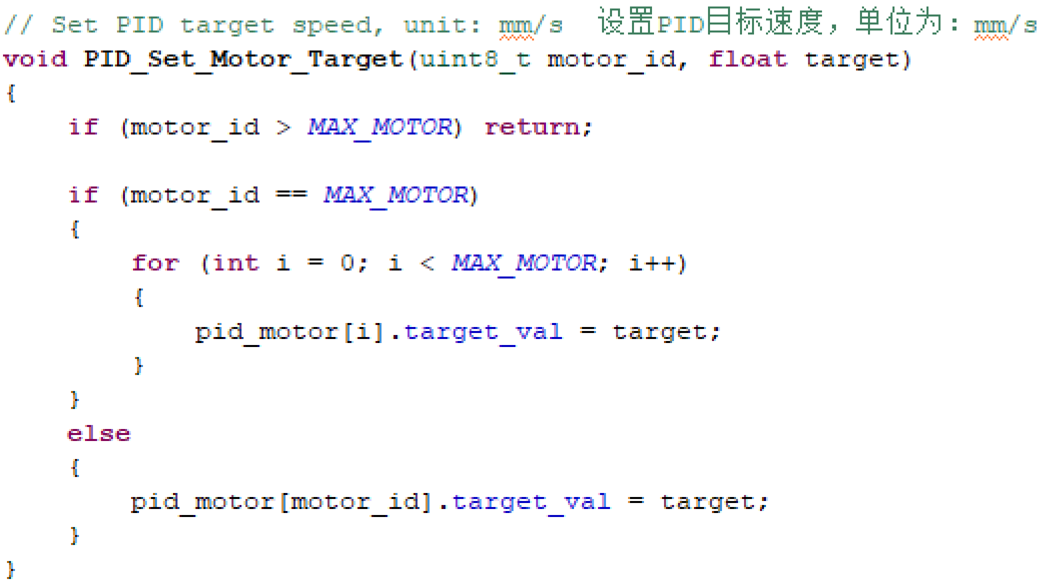
```
// Example Initialize PID parameters 初始化PID参数
void PID_Param_Init(void)
{
    for (int i = 0; i < MAX_MOTOR; i++)
    {
        pid_motor[i].target_val = 0.0;
        pid_motor[i].pwm_output = 0.0;
        pid_motor[i].err = 0.0;
        pid_motor[i].err_last = 0.0;
        pid_motor[i].err_next = 0.0;
        pid_motor[i].integral = 0.0;

        pid_motor[i].Kp = PID_DEF_KP;
        pid_motor[i].Ki = PID_DEF_KI;
        pid_motor[i].Kd = PID_DEF_KD;
    }
}
```

3. Calculation formula of incremental PID.



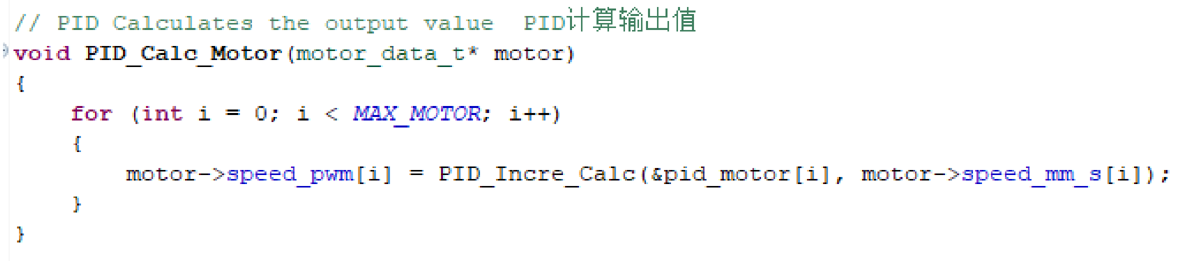
```
// Incremental PID calculation formula  增量式PID计算公式
float PID_Incre_Calc(motor_pid_t *pid, float actual_val)
{
    pid->err = pid->target_val - actual_val;
    pid->pwm_output += pid->Kp * (pid->err - pid->err_next)
                    + pid->Ki * pid->err
                    + pid->Kd * (pid->err - 2 * pid->err_next + pid->err_last);
    pid->err_last = pid->err_next;
    pid->err_next = pid->err;
    if (pid->pwm_output > MOTOR_MAX_PULSE)  pid->pwm_output = MOTOR_MAX_PULSE;
    if (pid->pwm_output < -MOTOR_MAX_PULSE) pid->pwm_output = -MOTOR_MAX_PULSE;
    return pid->pwm_output;
}
```

4. Set the target value.



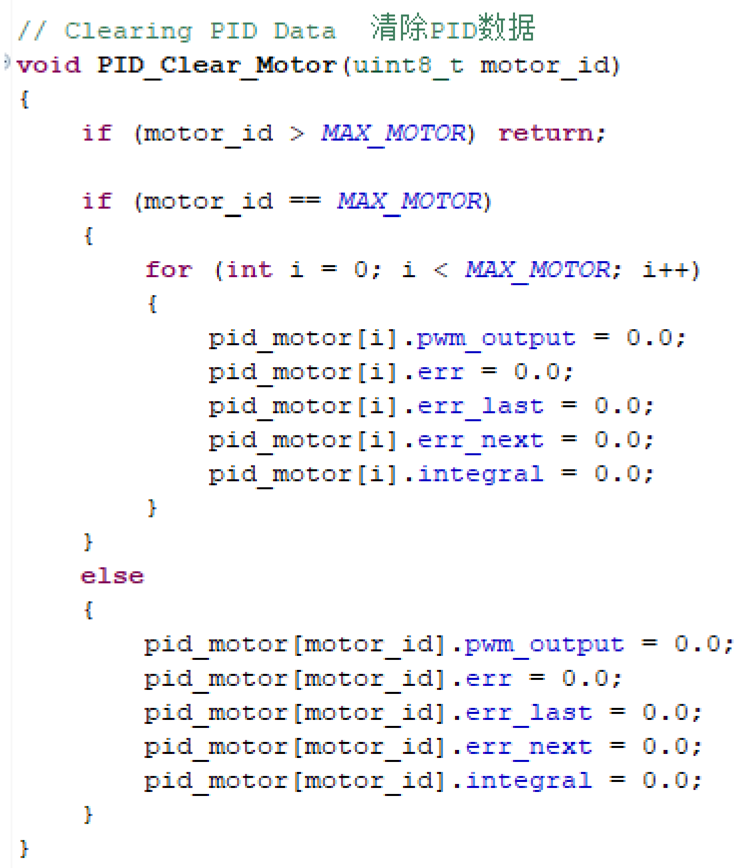
```
// Set PID target speed, unit: mm/s  设置PID目标速度，单位为：mm/s
void PID_Set_Motor_Target(uint8_t motor_id, float target)
{
    if (motor_id > MAX_MOTOR) return;

    if (motor_id == MAX_MOTOR)
    {
        for (int i = 0; i < MAX_MOTOR; i++)
        {
            pid_motor[i].target_val = target;
        }
    }
    else
    {
        pid_motor[motor_id].target_val = target;
    }
}
```

5. Calculate the PWM output value through PID calculation.



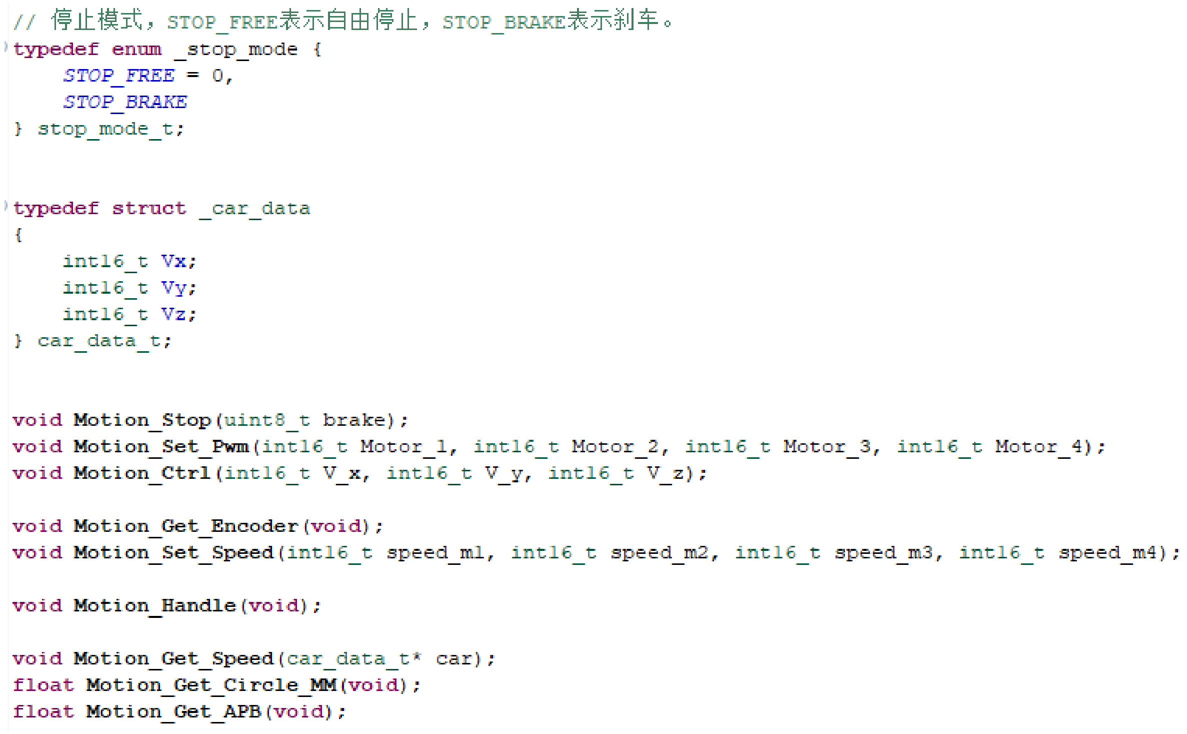
```c
// PID Calculates the output value  PID计算输出值
void PID_Calc_Motor(motor_data_t* motor)
{
    for (int i = 0; i < MAX_MOTOR; i++)
    {
        motor->speed_pwm[i] = PID_Incre_Calc(&pid_motor[i], motor->speed_mm_s[i]);
    }
}
```

6. Clear the PID parameter data.



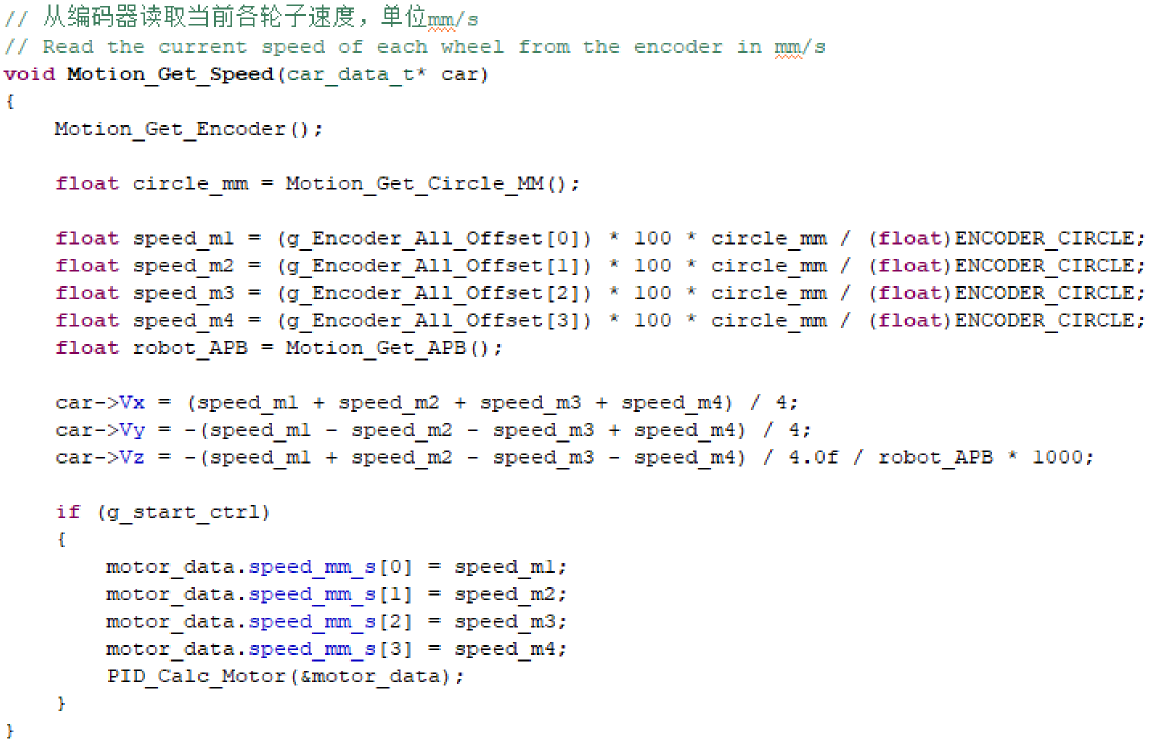
```c
// Clearing PID Data  清除PID数据
void PID_Clear_Motor(uint8_t motor_id)
{
    if (motor_id > MAX_MOTOR) return;

    if (motor_id == MAX_MOTOR)
    {
        for (int i = 0; i < MAX_MOTOR; i++)
        {
            pid_motor[i].pwm_output = 0.0;
            pid_motor[i].err = 0.0;
            pid_motor[i].err_last = 0.0;
            pid_motor[i].err_next = 0.0;
            pid_motor[i].integral = 0.0;
        }
    }
    else
    {
        pid_motor[motor_id].pwm_output = 0.0;
        pid_motor[motor_id].err = 0.0;
        pid_motor[motor_id].err_last = 0.0;
        pid_motor[motor_id].err_next = 0.0;
        pid_motor[motor_id].integral = 0.0;
    }
}
```

7. Create new bsp_motion.h and bsp_motion.c files.



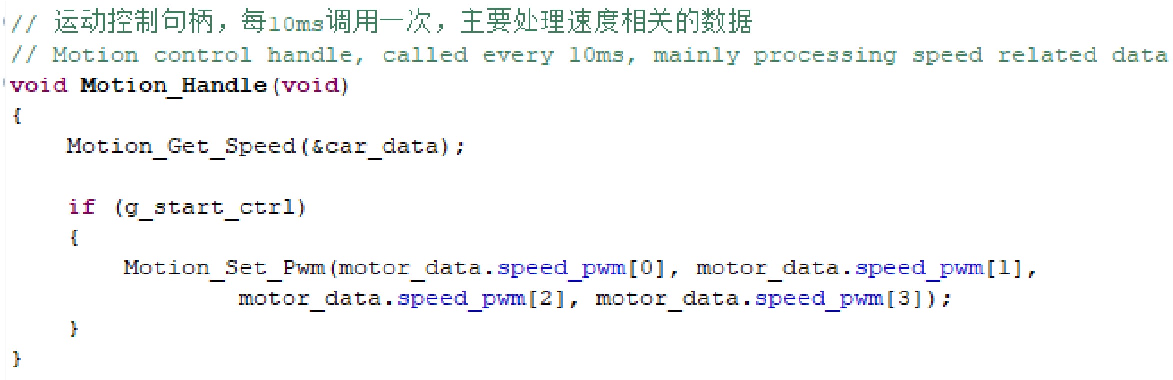
```c
// 停止模式，STOP_FREE表示自由停止，STOP_BRAKE表示刹车。
typedef enum _stop_mode {
    STOP_FREE = 0,
    STOP_BRAKE
} stop_mode_t;


typedef struct _car_data
{
    int16_t Vx;
    int16_t Vy;
    int16_t Vz;
} car_data_t;


void Motion_Stop(uint8_t brake);
void Motion_Set_Pwm(int16_t Motor_1, int16_t Motor_2, int16_t Motor_3, int16_t Motor_4);
void Motion_Ctrl(int16_t V_x, int16_t V_y, int16_t V_z);

void Motion_Get_Encoder(void);
void Motion_Set_Speed(int16_t speed_m1, int16_t speed_m2, int16_t speed_m3, int16_t speed_m4);

void Motion_Handle(void);

void Motion_Get_Speed(car_data_t* car);
float Motion_Get_Circle_MM(void);
float Motion_Get_APB(void);
```

8. Get and calculate the speed of the trolley from the encoder.



```
// 从编码器读取当前各轮子速度，单位mm/s
// Read the current speed of each wheel from the encoder in mm/s
void Motion_Get_Speed(car_data_t* car)
{
    Motion_Get_Encoder();

    float circle_mm = Motion_Get_Circle_MM();

    float speed_m1 = (g_Encoder_All_Offset[0]) * 100 * circle_mm / (float)ENCODER_CIRCLE;
    float speed_m2 = (g_Encoder_All_Offset[1]) * 100 * circle_mm / (float)ENCODER_CIRCLE;
    float speed_m3 = (g_Encoder_All_Offset[2]) * 100 * circle_mm / (float)ENCODER_CIRCLE;
    float speed_m4 = (g_Encoder_All_Offset[3]) * 100 * circle_mm / (float)ENCODER_CIRCLE;
    float robot_APB = Motion_Get_APB();

    car->Vx = (speed_m1 + speed_m2 + speed_m3 + speed_m4) / 4;
    car->Vy = -(speed_m1 - speed_m2 - speed_m3 + speed_m4) / 4;
    car->Vz = -(speed_m1 + speed_m2 - speed_m3 - speed_m4) / 4.0f / robot_APB * 1000;

    if (g_start_ctrl)
    {
        motor_data.speed_mm_s[0] = speed_m1;
        motor_data.speed_mm_s[1] = speed_m2;
        motor_data.speed_mm_s[2] = speed_m3;
        motor_data.speed_mm_s[3] = speed_m4;
        PID_Calc_Motor(&motor_data);
    }
}
```

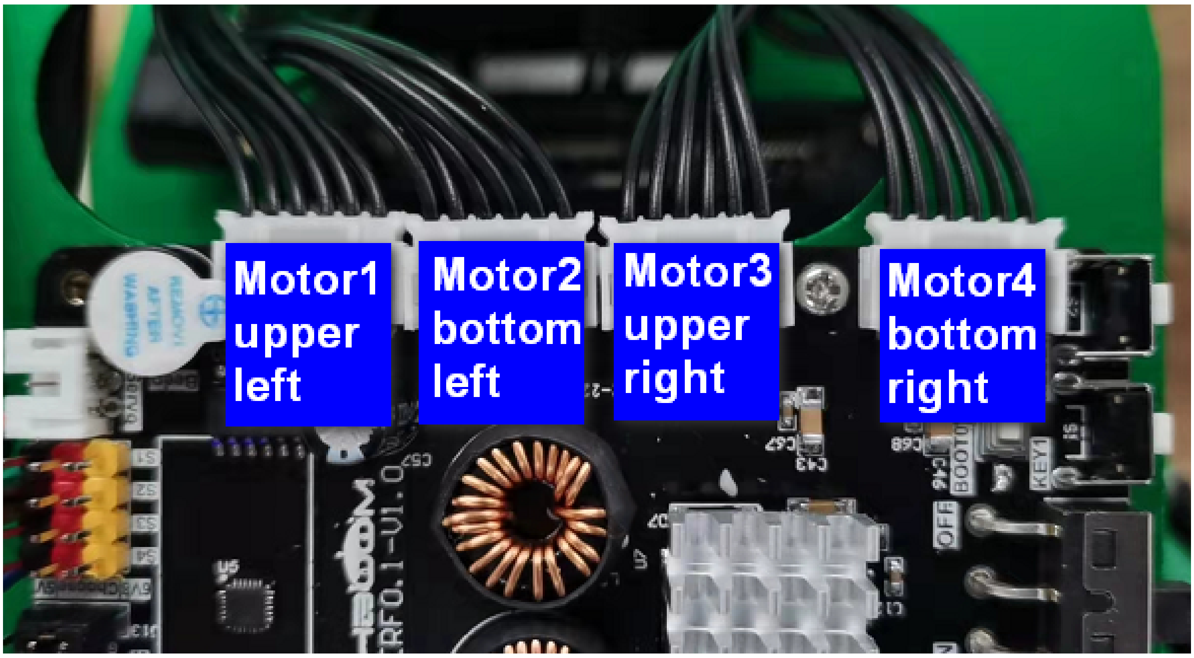
9. Re-update the PWM data of the motor according to the speed value, so as to achieve the effect of speed regulation.



```
// 运动控制句柄，每10ms调用一次，主要处理速度相关的数据
// Motion control handle, called every 10ms, mainly processing speed related data
void Motion_Handle(void)
{
    Motion_Get_Speed(&car_data);

    if (g_start_ctrl)
    {
        Motion_Set_Pwm(motor_data.speed_pwm[0], motor_data.speed_pwm[1],
                motor_data.speed_pwm[2], motor_data.speed_pwm[3]);
    }
}
```

5. hardware connection

The motor connecting line needs to be connected to the corresponding motor as shown in the figure below, otherwise it may cause the problem that the program does not match the phenomenon. Motor 1 corresponds to the motor in the upper left corner of the body, Motor 2 corresponds to the motor in the lower left corner, Motor 3 corresponds to the motor in the upper right corner, and Motor 4 corresponds to the motor in the lower right corner.



Since the power of the motor is relatively large, the expansion board should not be powered by USB 5V directly, but must be powered by DC 12V.

6. Experimental effect

Since the motor will turn when started, please stand up the trolley before the experiment, and the motor wheels are suspended in the air to avoid rampage.

After the program is programmed, the LED light flashes every 200 milliseconds. Press the button KEY1 once, the car moves forward, and press the KEY1 button again to stop the car.