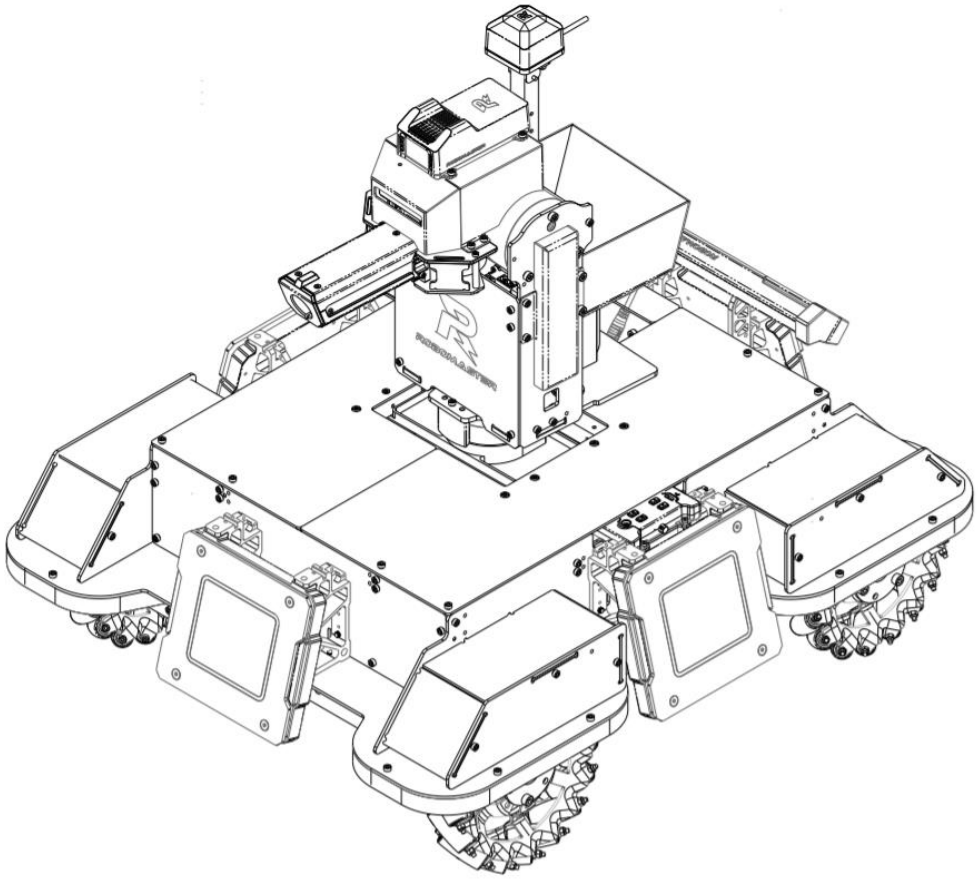
RoboMaster
2020 Standard A Unassembled kit
Development Manual

V2.0 2020.2

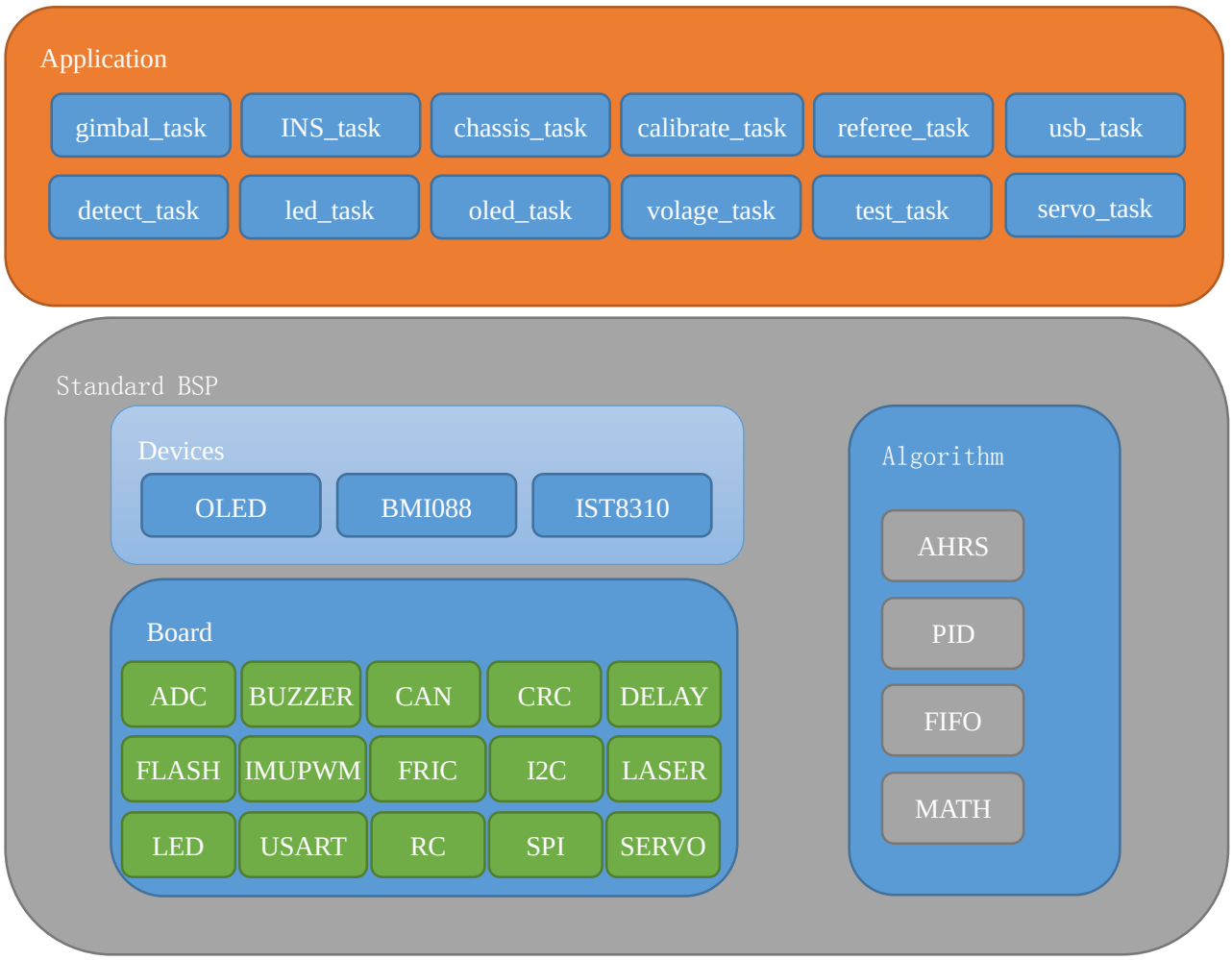


ROBOMASTER

Contents

Software Block Diagram....3
File Directory....3
Software Environment....4
Programming Specifications....4
Functional Description....4
Functional Implementation Framework....5

Software Block Diagram



File Directory

1. application: task function and interrupt function
2. bsp: to realize the encapsulation of the underlying layer, the migration standard library needs to re-implement this level
3. components\algorithm: attitude calculation algorithm and DSP library
4. components\controller: implementation of PID correlation function
5. components\devices: driving of gyroscopes BMI088, IST8310 and OLED
6. components\support: check functions of CRC8 and CRC16 as well as fifo correlation function
7. doc: related to document

8. Drivers: driver library automatically generated by cubeMX
9. Inc: h file automatically generated by cubeMX
10. MDK-ARM: engineering file generated by cubeMX
11. Middlewares: middlewares generated by cubeMX
12. Src: c file automatically generated by cubeMX

Software Environment

Toolchain/IDE	MDK-ARM V5
STM32F4xx_DFP Packs	2.13.0
STM32CubeMx	5.2.1
package version	STM32Cube FW_F4 V1.21.1
FreeRTOS version	10.0.1
CMSIS-RTOS version	1.02

Programming Specifications

Variables and functions are named following the UNIX/Linux style
Tasks that do not require accurate timing are implemented by a self-implemented software timer, and the timing accuracy can be affected by task scheduling

Functional Description

1. Calibration function: provide functions including gimbal calibration, gyroscope zero drift calibration and chassis ID reset
2. Chassis control function: complete motion control of the Mecanum wheels and power control of the chassis, with 4 control modes available as follows: closed-loop control by gimbal angle, closed-loop control by chassis angle, angle-free closed-loop rotation control of the chassis and native CAN control.
3. Offline judgment function: judge whether the device is offline or not according to the timestamp of data feedback.
4. Gimbal control function: control the gimbal angle. 3 control modes are provided: gyroscope angle control, motor encoder disc angle control and native CAN control.

5. Attitude calculation function: complete the angle fusion of gyroscope accelerometer and solve the Euler angle.
6. RGB switching of LED: use three-color LED to achieve RGB display with breathing light effect. It can be used to show whether the program has crashed.
7. OLED display function: display the battery percentage and the device error information to facilitate locating the issue.
8. Data analysis of the referee system: use single byte to analyze the referee system data, which is applicable to the 2019 referee system. The system has to be updated to the Final Tournament version.
9. Remote control data analysis: use serial port idle interrupt function to analyze the data sent by the receiver.
10. Servo control: use the keys to control 4 idle PWMs to output servo signals to facilitate the addition of cartridge control or simple mechanical device at a later stage.
11. Firing control: control the lower projectile feeding device and complete the launching logic.
12. Power supply sampling: sample the power supply voltage to estimate the current battery percentage as a simple reference to determine the battery level. This can be used when the battery is inside the robot and inconvenient to observe.

Functional Implementation Framework

Global variables are passed by pointers to reduce extern use. For example, the remote control pointer provides const RC_ctl_t *get_remote_control_point(void) to return the constant pointer, creates a new structure pointer of local_rc_ctrl in the task requiring remote control parameters, and passes the global variables by obtaining the pointer.

All global variables are in the app layer, so that personnel can easily view the parameters without jumping to other layers to find sources. The driver layer and hardware layer only provide processing functions and hardware initialization functions.

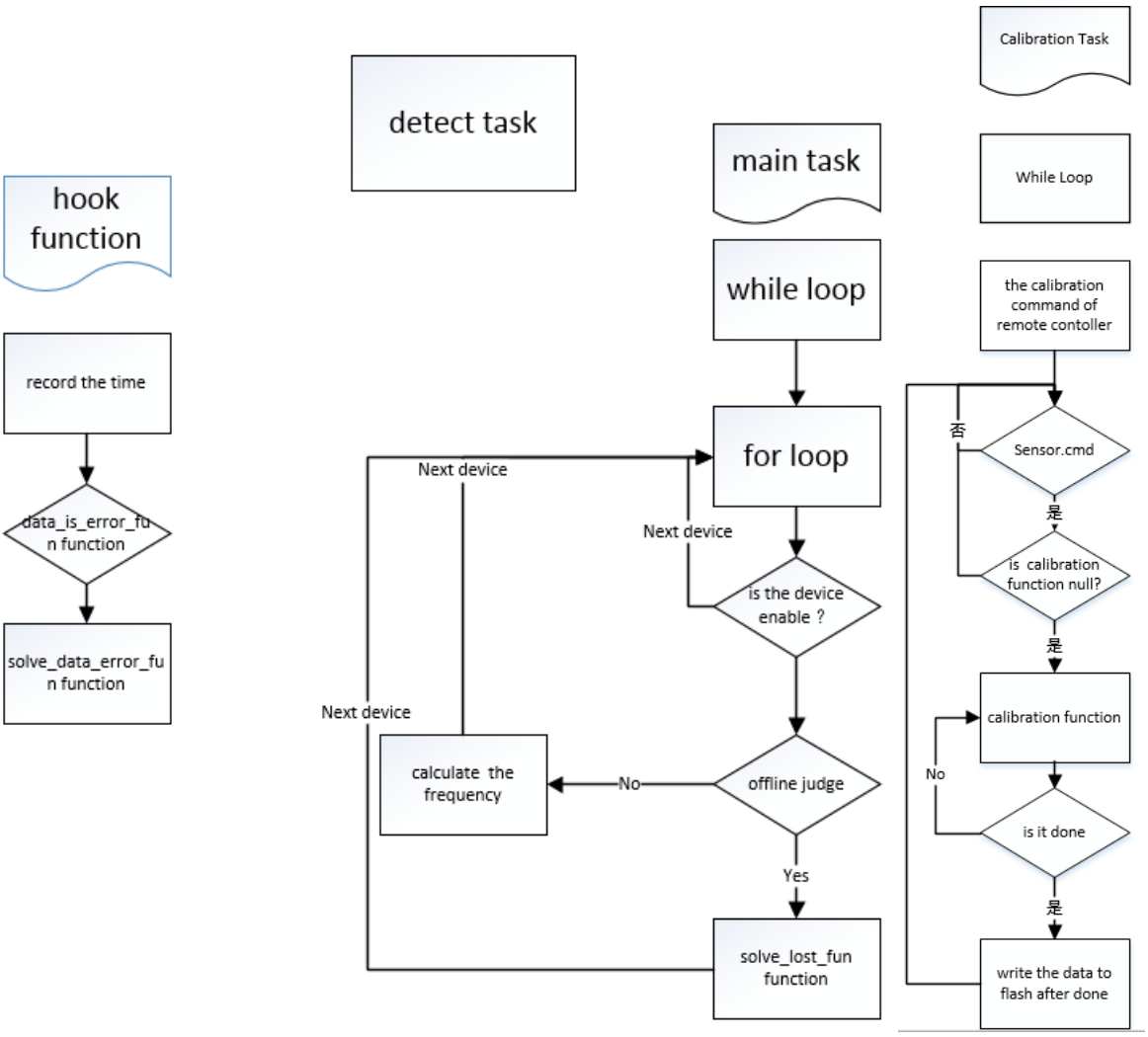
For gimbal control tasks, chassis control tasks, etc., there are corresponding variables needed for the integration of structural control variables for the users to view.

Here are the implementation workflows of some common tasks.

a. Calibration Task and Module Offline Judgment Task

The calibration task is mainly to conduct zero drift calibration of the gyroscope, median calibration of the gimbal, and fast ID setting of the chassis. The module offline judgment task is mainly to judge whether

disconnection occurs by judging the difference between the data transmission time of the module and the current system time. These two tasks are mainly completed through pointer functions.



Adding New Devices to Offline Tasks

If you need to add a device to an offline judgment task, you can take the following steps.

1.Add the device name at the end of "errorList" in "detect_task.h" as below

```
enum errorList
{
    ...
    XXX_TOE,      //new device
    ERROR_LIST_LENGHT,
};
```

2.Add offlinetime, onlinetime and priority parameters to the "detect_init" function

```
 uint16_t set_item[ERROR_LIST_LENGHT][3] =
  {
        ...
     {n,n,n}, //XX_TOE
};
```

3.If there is a “data _ is _ error _ fun, solve _ lose _ fun, solve _ data _ error _ fun” function, assign the value to the function pointer

4.When the “XXX_TOE” device data arrives, add “detect_hook (XXX_TOE)” function.

Adding New Devices to Calibration Tasks

If you need to add a device to a calibration task, you can follow the following steps.

1.Add the device name to “cali_id_e of calibrate_task.h” as below

```
typedef enum
{
      ...
      //add more...
      CALI_XXX,
      CALI_LIST_LENGHT,
} cali_id_e;
```

2. Add data structure to “calibrate_task.h”, which should be a multiple of 4 bytes as below

```
typedef struct
{
      uint16_t xxx;
      uint16_t yyy;
      fp32 zzz;
} xxx_cali_t; //Length: 8 bytes, which should be 4, 8, 12, 16...
```

3.Add "sizeof(xxx_cali_t)" to "FLASH_WRITE_BUF_LENGHT",

to implement the new function “bool_t cali_xxx_hook(uint32_t *cali, bool_t cmd)”,

Add a new name to "cali_name[CALI_LIST_LENGHT][3]"

to declare the variable “xxx_cali_t xxx_cail”,

Add the variable address to “cali_sensor_buf[CALI_LIST_LENGHT]”

Add the data length to “cali_sensor_size[CALI_LIST_LENGHT]”,

Finally, add a function to “cali_hook_fun[CALI_LIST_LENGHT]”.

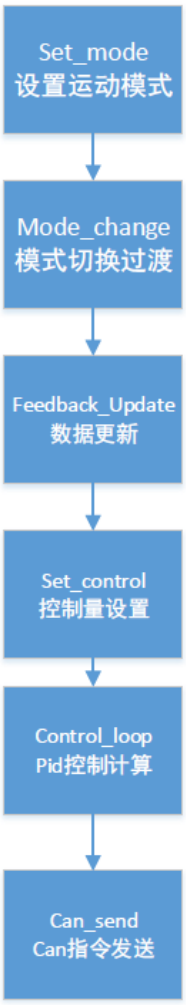
b. Gimbal, Chassis and Shooting Task Framework

For the gimbal and chassis, a two-layered structure is adopted. Layer 1 is the control layer to achieve different control objectives, and Layer 2 is the behavior layer to achieve the setting of corresponding control objectives for different functions.

For the gimbal, there are 3 control modes: gyroscope angle control, encoder angle control and original CAN control, with 6 behavior modes: powerless behavior, initialization behavior, calibration behavior, absolute angle control behavior, relative angle control behavior, stationary position behavior.

For chassis, there are 4 control modes: control by gimbal angle, control by chassis angle, angle-free rotation control and original CAN control, with 6 behavior modes: powerless behavior, non-moving behavior, standard-like behavior following gimbal, engineering-like behavior following chassis angle, behavior without following gimbal, open-loop behavior.

The whole processes of shooting, chassis and gimbal are similar, and the framework flow is as follows.



Process	Function	Notes
set_mode	Setting the state machine via remote control	
mode_change	Updated control value after state machine change	For example, when the gimbal changes the encoder from gyroscope, it is necessary to reset the control target value
feedback_update	Feedback data update	
set_control	Setting control quantity	
control_loop	Controller calculation	
can_send	Can sending command	

Adding New Modes to Gimbal Control Tasks

For the gimbal, if you would like to add a new behavior mode, please refer to the following steps

1.Add a new behavior name to “gimbal_behavior_e” in the “gimbal_behavior.h” file,

```
erum
{
    ...
    ...
    GIMBAL_XXX_XXX, // newly added
}gimbal_behaviour_e,
```

2. Implement a new function: gimbal_xxx_xxx_control(fp32 *yaw, fp32 *pitch, gimbal_control_t *gimbal_control_set);

The "yaw, pitch" parameters are the motion control input of the gimbal

The first parameter: “yaw” controls the movement of yaw axis, usually in angular increment, with positive value moving counterclockwise and negative value clockwise

The second parameter: “pitch” controls the movement of the pitch axis, usually in angular increments, with positive values moving counterclockwise and negative values clockwise

In this new function, you can assign the desired parameters to "yaw" and "pitch"

3. In the function "gimbal_behavour_set", add a new logical judgment and assign values to “gimbal _ behavior” as “GIMBAL_XXX_XXX”

Add "else if(gimbal_behaviour == GIMBAL_XXX_XXX)" at the end of the function “gimbal_behaviour_mode_set”, and select a gimbal control mode

GIMBAL_MOTOR_RAW: Use “yaw” and “pitch” as motor current setting values and send them directly to CAN bus.

GIMBAL_MOTOR_ENCONDE: “yaw” and “pitch” are angle increments, which control the relative angle of coding.

GIMBAL_MOTOR_GYRO: “yaw” and “pitch” are angle increments, which control the absolute angle of gyroscope.

4. At the end of the function "gimbal_behaviour_control_set", add

```
else if(gimbal_behaviour == GIMBAL_XXX_XXX)
{
    gimbal_xxx_xxx_control(&rc_add_yaw, &rc_add_pit, gimbal_control_set);
}
```

Adding New Modes to Chassis Control Tasks

If you want to add a new behavior mode

1.Add a new behavior name to “chassis_behaviour_e” in the “ chassis_behaviour.h” file

```
erum
{
    ...
    ...
   CHASSIS_XXX_XXX, // newly added
}chassis_behaviour_e,
```

2. Implement a new function “chassis_xxx_xxx_control(fp32 *vx, fp32 *vy, fp32 *wz, chassis_move_t * chassis)”

where the parameters "vx,vy,wz" are the movement control input of the chassis

The first parameter: “vx” normally controls longitudinal movement, with positive values for forward movement and negative values for backward movement

The second parameter: “vy” normally controls lateral movement, with positive values for leftward movement and negative values for rightward movement

The third parameter: “wz” may be angle control or rotational speed control

In this new function, you can assign the desired speed parameters to "vx", "vy", and "wz"

3. In the function "chassis_behaviour_mode_set_set", add a new logical judgment and assign values to “chase_behavior_mode” as “chase_XXX_XXX”

At the end of the function, add "else if(chassis_behaviour_mode == CHASSIS_XXX_XXX)", and select a chassis control mode

There are 4 modes available:

CHASSIS_VECTOR_FOLLOW_GIMBAL_YAW: “vx” and “vy” for speed control, and “wz” for angle control, which is the relative angle between the gimbal and chassis

You can name it as "xxx_angle_set" instead of “wz”

CHASSIS_VECTOR_FOLLOW_CHASSIS_YAW: “vx” and “vy” for speed control, and “wz” for angle control, which is the absolute angle calculated by the gyroscope of the chassis

You can name it as "xxx_angle_set"

HASSIS_VECTOR_NO_FOLLOW_YAW: “vx” and “vy” for speed control, and “wz” for rotational speed control

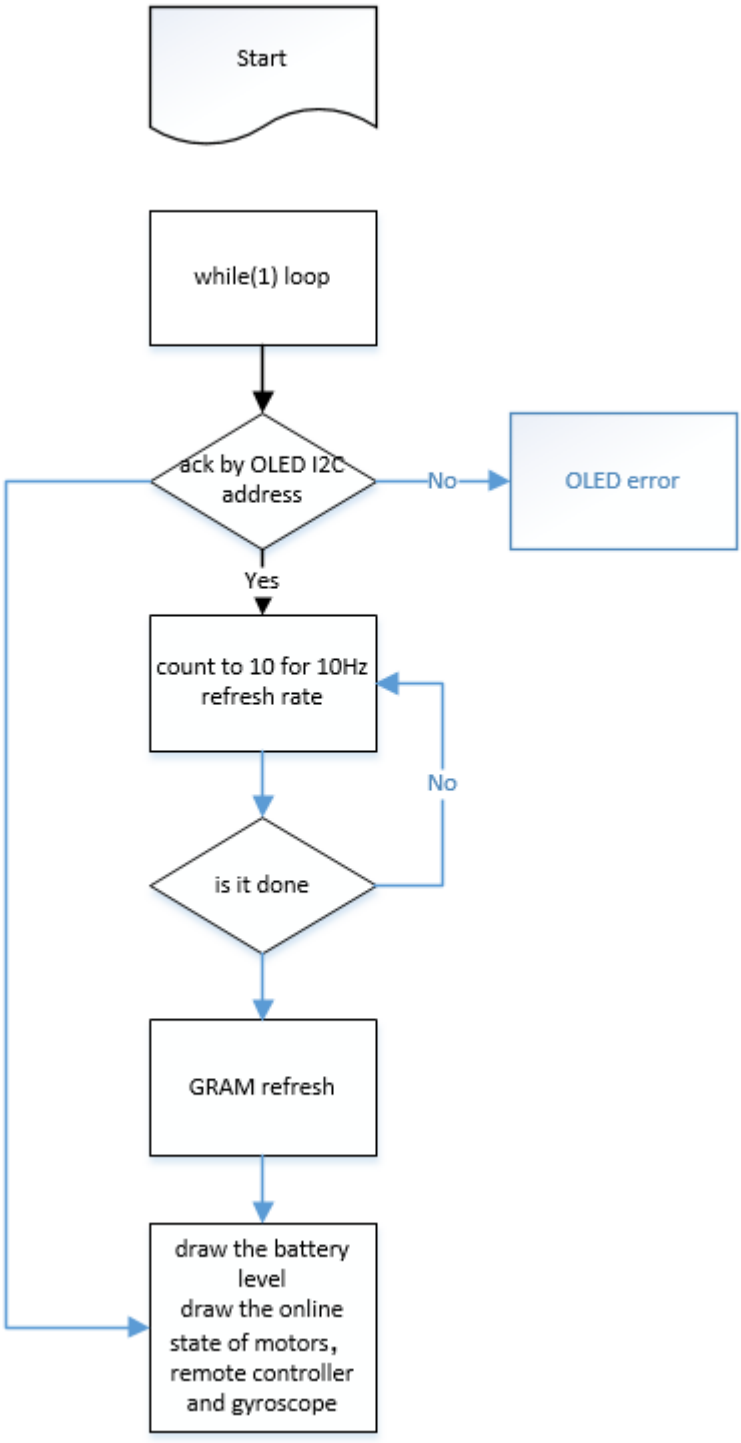
CHASSIS_VECTOR_RAW: Use “vx”, “vy” and “wz” to directly and linearly calculate the current value of the wheel, with the current value to be directly sent to the can bus

4. At the end of the function "hassis_behaviour_control_set", add

```
else if (chassis_behaviour_mode == CHASSIS_XXX_XXX)
 {
     chassis_xxx_xxx_control(vx_set, vy_set, angle_set, chassis_move_rc_to_vector);
 }
```

c. OLED Task Framework

The OLED task aims to confirm the connection status by querying the I2C address of the OLED at 100Hz and refresh the OLED screen at 10Hz. OLED is compatible with SSD1306-driven monochrome and bicolor modules.For monochrome and bicolor modules, please modify the corresponding macro definition in the file OLED.C.


Start
while(1) loop
ack by OLED I2C address
No
OLED error
Yes
count to 10 for 10Hz refresh rate
No
is it done
GRAM refresh
draw the battery level
draw the online state of motors, remote controller and gyroscope

邮箱：robomaster@dji.com
论坛：http://bbs.robomaster.com
官网：http://www.robomaster.com
电话：0755-36383255（周一至周五10:00-19:00）
地址：广东省深圳市南山区西丽镇茶光路1089号集成电路设计应用产业园2楼202