Pixhawk 整体逻辑

1. commander 和 navigator 产生期望位置
2. position_estimator 是当前位置
3. 通过 pos_ctrl 产生期望姿态
4. attitude_estimator 是当前姿态
5. 通过 att_ctrl 产生 pwm 的数值
6. 最后通过 mixer 和 motor_driver 控制 4 个电机

navigator 模块

用于接受任务并将其转为底层导航的原始数据。

代码目录

- navigator_main.cpp 处理任务项目，地理栅栏和故障安全导航行为，给位置控制器发布位置设定值。
- navigator_mode.cpp 导航器中不同模式的基类
- mission_block.cpp 使用任务项目的辅助类
- mission.cpp 访问任务的辅助类
- loiter.cpp 待机的辅助类

- rtl.cpp 访问 rtl 的辅助类
- takeoff.cpp 起飞的辅助类
- land.cpp 在当前位置着陆的辅助类
- mission_feasibility_checker.cpp 检查提供导航功能的任务是否可行
- geofence.cpp 提供处理地理栅栏的函数
- datalinkloss.cpp 基于 OBC 规则的数据链路丢失模式的辅助类
- rcloss.cpp 基于 OBC 规则的 RC 丢失模式的辅助类
- enginefailure.cpp 固定式发动机故障模式的辅助类
- gpsfailure.cpp 基于 OBC 规则的 GPS 故障模式的辅助类
- follow_target.cpp 追踪和跟随给定的位置的辅助类

navigator_main.cpp 代码分析

1．根据 ORB_ID(vehicle_status)主题选择不同的导航模式。

导航模式

```
_navigation_mode_array[0] = &_mission;//任务

_navigation_mode_array[1] = &_loiter;//待机

_navigation_mode_array[2] = &_rtl;//返航模式  (RTL, return to launch)

_navigation_mode_array[3] = &_dataLinkLoss;//数据链路故障
```

```
_navigation_mode_array[4] = &_engineFailure;//引擎故障

_navigation_mode_array[5] = &_gpsFailure;//gps 故障

_navigation_mode_array[6] = &_rcLoss;//遥控故障  (RC, remote control)

_navigation_mode_array[7] = &_takeoff;//起飞

_navigation_mode_array[8] = &_land;//着陆

_navigation_mode_array[9] = &_follow_target;//跟随目标
```

2．获取 commander.cpp 发布的 ORB_ID(vehicle_status)主题

```
/* Do stuff according to navigation state set by commander */

        //按 commander 设定的导航状态工作

        switch (_vstatus.nav_state) {

        case vehicle_status_s::NAVIGATION_STATE_AUTO_MISSION:

            _pos_sp_triplet_published_invalid_once = false;

            _navigation_mode = &_mission;

            break;
```

```
case vehicle_status_s::NAVIGATION_STATE_AUTO_LOITER:
	_pos_sp_triplet_published_invalid_once = false;
	_navigation_mode = &_loiter;
	break;

case vehicle_status_s::NAVIGATION_STATE_AUTO_RCRECOVER:
	_pos_sp_triplet_published_invalid_once = false;
	_navigation_mode = &_rcLoss;
	break;

case vehicle_status_s::NAVIGATION_STATE_AUTO_RTL:
	_pos_sp_triplet_published_invalid_once = false;
	_navigation_mode = &_rtl;
	break;

case vehicle_status_s::NAVIGATION_STATE_AUTO_TAKEOFF:
	_pos_sp_triplet_published_invalid_once = false;
```

```
        _navigation_mode = &_takeoff;
        break;

    case vehicle_status_s::NAVIGATION_STATE_AUTO_LAND:
        _pos_sp_triplet_published_invalid_once = false;
        _navigation_mode = &_land;
        break;

    case vehicle_status_s::NAVIGATION_STATE_DESCEND:
        _pos_sp_triplet_published_invalid_once = false;
        _navigation_mode = &_land;
        break;

    case vehicle_status_s::NAVIGATION_STATE_AUTO_RTGS:
        _pos_sp_triplet_published_invalid_once = false;
        _navigation_mode = &_dataLinkLoss;
        break;
```

```
case vehicle_status_s::NAVIGATION_STATE_AUTO_LANDENGFAIL:
	_pos_sp_triplet_published_invalid_once = false;
	_navigation_mode = &_engineFailure;
	break;

case vehicle_status_s::NAVIGATION_STATE_AUTO_LANDGPSFAIL:
	_pos_sp_triplet_published_invalid_once = false;
	_navigation_mode = &_gpsFailure;
	break;

case vehicle_status_s::NAVIGATION_STATE_AUTO_FOLLOW_TARGET:
	_pos_sp_triplet_published_invalid_once = false;
	_navigation_mode = &_follow_target;
	break;

case vehicle_status_s::NAVIGATION_STATE_MANUAL:
```

```
case vehicle_status_s::NAVIGATION_STATE_ACRO:

case vehicle_status_s::NAVIGATION_STATE_ALTCTL:

case vehicle_status_s::NAVIGATION_STATE_POSCTL:

case vehicle_status_s::NAVIGATION_STATE_TERMINATION:

case vehicle_status_s::NAVIGATION_STATE_OFFBOARD:

case vehicle_status_s::NAVIGATION_STATE_STAB:

default:

    _pos_sp_triplet_published_invalid_once = false;

    _navigation_mode = nullptr;

    _can_loiter_at_sp = false;

    break;

}



/* iterate through navigation modes and set active/inactive for each */

//迭代导航模式，并为每个设置活动/不活动

for (unsigned int i = 0; i < NAVIGATOR_MODE_ARRAY_SIZE; i++) {
```

```
                _navigation_mode_array[i]->run(_navigation_mode ==
_navigation_mode_array[i]);
        }
```

3．之后调用 navigator 文件夹中其他文件。

4．最终目的是为了使用

mission_item_to_position_setpoint(&_mission_item, &pos_sp_triplet->current);

mission_item_to_position_setpoint(&mission_item_next_position,
&pos_sp_triplet->next);

将_mission_item 结构体的值赋给 pos_sp_triplet 结构体

之后在 navigator_main.cpp 中调用

参考：http://blog.csdn.net/czyv587/article/details/52120876