Task main

2016年5月27日 10:55

订阅消息 sub.....
parameters_update(true); 初始化更新参数，参数为true，此次为强制读取
poll_subscriptions 初始化更新全部的状态
初始化矩阵、推力积分的积分量

进入循环
Px4_poll 函数，该函数内部包括一个信号量，负责读取 local_pos的数据，最多等待500ms，
如果读不到数据，则重新读取，因为之后的计算基础就是local_position的数据。

poll_subscriptions();	读取航向角： _yaw = euler_angles(2);
parameters_update(false);	参数为false，即有数据更新的时候才读

更新数据和参数

更新时间 t/dt

```
if (_control_mode.flag_armed && !was_armed) {
        /* reset setpoints and integrals on arming */
        _reset_pos_sp = true;
        _reset_alt_sp = true;
        _vel_sp_prev.zero();
        reset_int_z = true;
        reset_int_xy = true;
        reset_yaw_sp = true;
}
```

第一次解锁的时候，将所有的setpoint及积分项置零

update_ref(); 更新参考高度

if (_local_pos.timestamp > 0) {

_pos =	_local_pos.x/y/z
_vel =	_local_pos.vx/vy/vz

读取位置、速度信息，每种飞行模式都读取。

? _vel_err_d(i) = _vel_x_deriv.update(-_vel(i));
? 计算得出 _vel_err_d

在正常的控制模式下{
  ? 前馈设置为0；

重置两个参数	_run_pos_control = true; _run_alt_control = true;

非手动或非定点模式下，不进行位置控制，_pos_hold_engaged = false；
非手动或非定高模式下，不进行高度控制，_alt_hold_engaged = false;

遥控器控制模式下，调用 control manual(dt) 函数，dt为循环间隔，此函数根据摇杆的值及飞行模式设置相应的_vel_sp & _pos_sp，遥控器的死区控制也在此函数中

地面站模式调用control_offboard函数，自动模式调用control_auto

地面站IDLE模式下将所有期望姿态att_sp参数置为零

手动&landed情况下，停止姿态控制

其他情况（正常的飞行模式）{

位置控制，根据pos_sp给出vel_sp
if (_run_pos_control) {
_vel_sp(0) = (_pos_sp(0) - _pos(0)) * _params.pos_p(0);
_vel_sp(1) = (_pos_sp(1) - _pos(1)) * _params.pos_p(1);
}
对水平速度做限定，防止出现速度过大的情况；

Follow-me相关控制，跟随目标的模式下有位置跟随以及速度跟随；

高度控制，根据pos_sp(2)给出vel_sp(2)：
if (_run_alt_control) {
_vel_sp(2) = (_pos_sp(2) - _pos(2)) * _params.pos_p(2);
}

限定水平及垂直方向速度最大值；

非定高/非定点设定，如果不控制位置/高度，则将当前的位置/高度设定为setpoint值

如果为手动/定高/定点模式，将水平方向速度的setpoint置为零
_control_mode.flag_control_velocity_enabled = false:
{
_vel_sp(0) = 0.0f;
_vel_sp(1) = 0.0f;
}
如果纯手动模式，将垂直方向的速度setpoint置为零

_control_mode.flag_control_climb_rate_enabled =false：
{
	_vel_sp(2) = 0.0f;
}

自动降落/自动起飞逻辑

根据加速度极值算出来新的 _vel_sp， publish到global_vel_sp
orb_publish(ORB_ID(vehicle_global_velocity_setpoint), _global_vel_sp_pub, &_global_vel_sp);

? 定高/定点模式下：
	根据需要对积分项进行重置
	定高/定点：
		reset_int_z;
		thrust_int(2) = -0.12;
	定点：
		reset_int_xy;
		thrust_int(0) = 0.0f;
		thrust_int(1) = 0.0f;

计算控制量， vel_err = _vel_sp - _vel;

从手动/定高切入到定点时，做一个平滑，保证状态连续。
if (!control_vel_enabled_prev &&
_control_mode.flag_control_velocity_enabled)

接下来，根据vel_error进行PID控制
thrust_sp = vel_err.emult(_params.vel_p) +_vel_err_d.emult(_params.vel_d) + thrust_int;

这样，三个方向的thrust_sp就得出来了

然后如果为自动起飞，那么需要进行垂直方向推力补偿

接下来为关键逻辑：
1.在非定点模式下，将水平方向的thrust_sp置为零
2.在非定高模式下，将垂直方向的thrust_sp置为零
此处手动状态下， thrust_sp已经全部为零，定高状态下只有thrust_sp(2)有值，定点状态三方向都有值

将推力值限定为大于等于零，避免产生向下的推力

自动降落相关逻辑，通过对8秒内的速度求均值判断飞机是否处于降落状态

在定点模式下对水平方向的推力做限定；
定高或定点下，有姿态倾角的情况下做推力补偿

对推力三个方向做限定

更新积分项，定点下对XY方向的积分项累加，定高定点下对Z方向的积分项累加
接下来为重要逻辑，根据推力值计算出期望姿态矩阵，推力均为NED中的向量
1，垂直方向的推力始终平行于体轴的Z，根据这个算出body_z
2，通过期望的yaw的方向（实际为期望航向转90°）与Z轴叉乘出body_x
3，通过z与x叉乘计算出y轴的方向
4，赋值到R与Q中，并算出欧拉角pithc,roll
但是并不发布出去，因为还需要后续处理

对local_pos_sp的数据赋值并发布

至此，正常的飞行模式下，所有数据处理完毕，只是在手动模式下，没有vel_sp

在定点/定高/手动模式下，对期望的姿态角进行下一步赋值：

yaw：1，需要保持航向，将当前航向赋值给期望航向，当前航向的值在最开始的poll_subscription中读取
2，改变航向，从遥控器中读取yaw的值，根据yaw_rate_max等值修正原始值

Roll/pitch：在定高和手动模式下，根据遥控器的值给出，如果定点模式则保持之前计算得出来的值

Thrust：在手动模式下，通过油门杆的值给出来

以前面计算出来的姿态角为基础，计算出yaw之后的R_sp

将新的R_sp作为结果，发布出去

循环结束！