PX4 Autopilot

Firmware Guide (Spring-22)

Please keep editing if you find mistakes

.CERLAB.

Carnegie Mellon University

Multicopter Control Architecture

Multicopter Control Architecture

- http://docs.px4.io/master/en/concept/architecture.html
- http://docs.px4.io/master/en/concept/px4_systems_architecture.html
- https://www.youtube.com/watch?v=nEo4WGl4Lgc&t=118s

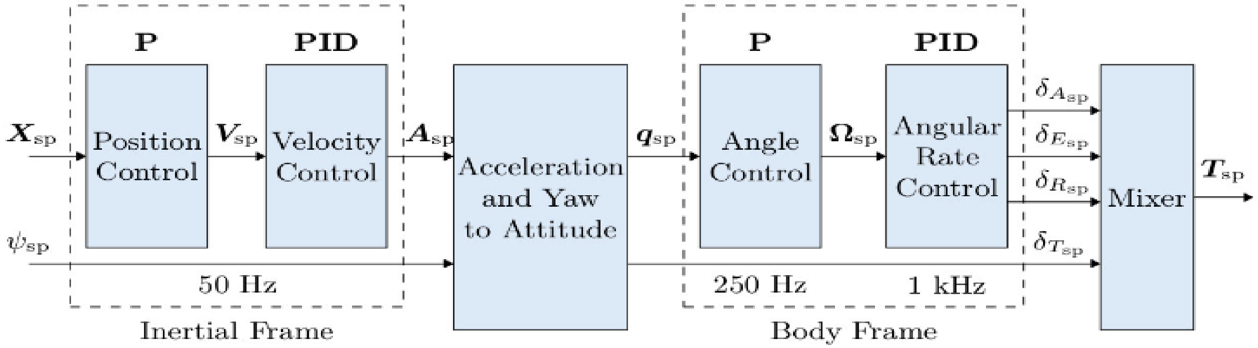
Multicopter Control Architecture

Controller Diagrams

This section contains diagrams for the main PX4 controllers.

The diagrams use the standard PX4 notation (and each have an annotated legend).

Multicopter Control Architecture



- This is a standard cascaded control architecture.
- The controllers are a mix of P and PID controllers.
- Estimates come from EKF2.
- Depending on the mode, the outer (position) loop is bypassed (shown as a multiplexer after the outer loop). The position loop is only used when holding position or when the requested velocity in an axis is null.

http://docs.px4.io/master/en/flight_stack/controller_diagrams.html

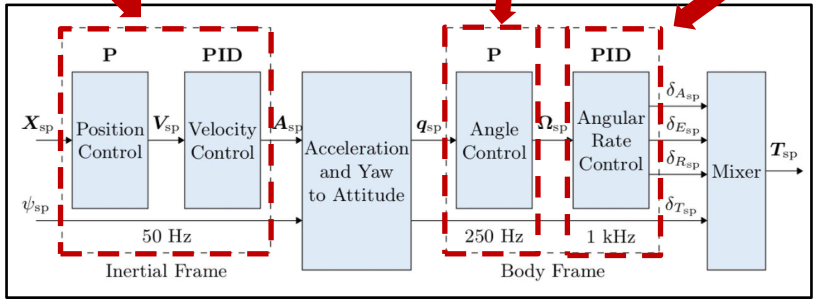
Github Repo

https://github.com/PX4/PX4-Autopilot/blob/master/src/modules

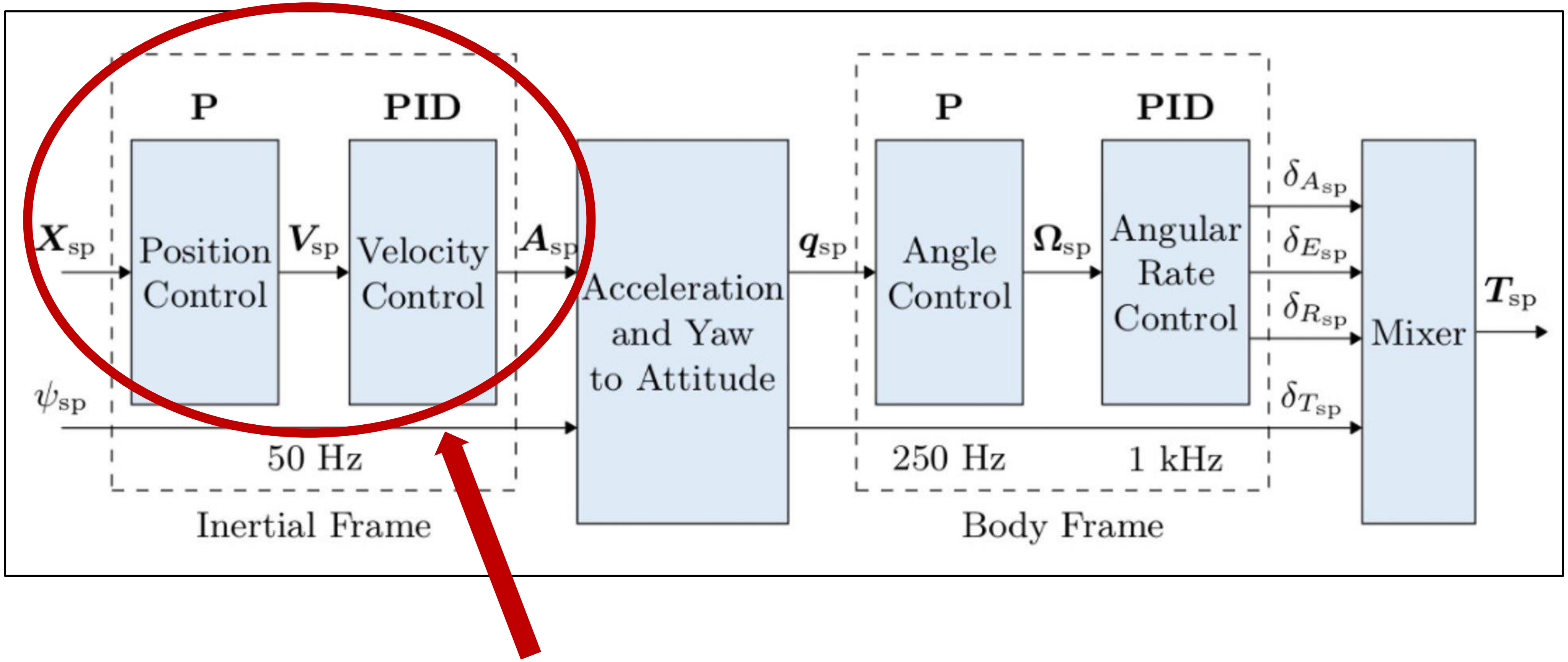
- mc_pos_control
  - PositionControl
    a. PositionControl.cpp
    b. PositionControl.hpp
  - MulticopterPositionControl.cpp
  - MulticopterPositionControl.hpp

- mc_att_control
  - AttitudeControl
    a. AttitudeControl.cpp
    b. AttitudeControl.hpp
  - mc_att_control.cpp
  - mc_att_control.hpp

- mc_rate_control
  - RateControl
    a. RateControl.cpp
    b. RateControl.hpp
  - MulticopterRateControl.cpp
  - MulticopterRateControl.hpp

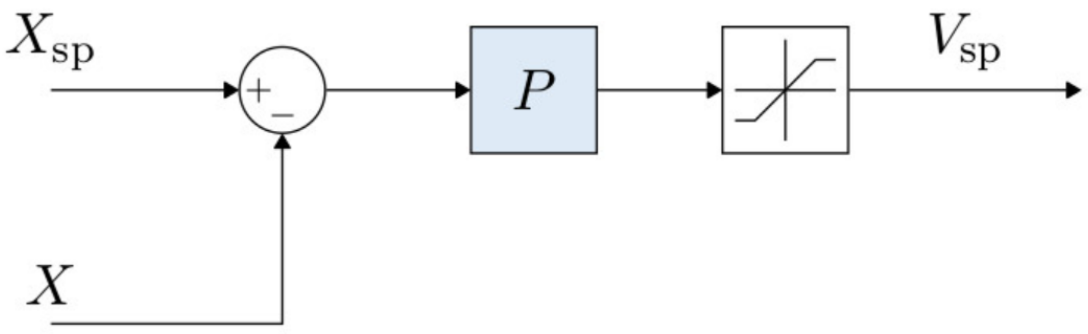


Position & Velocity Control



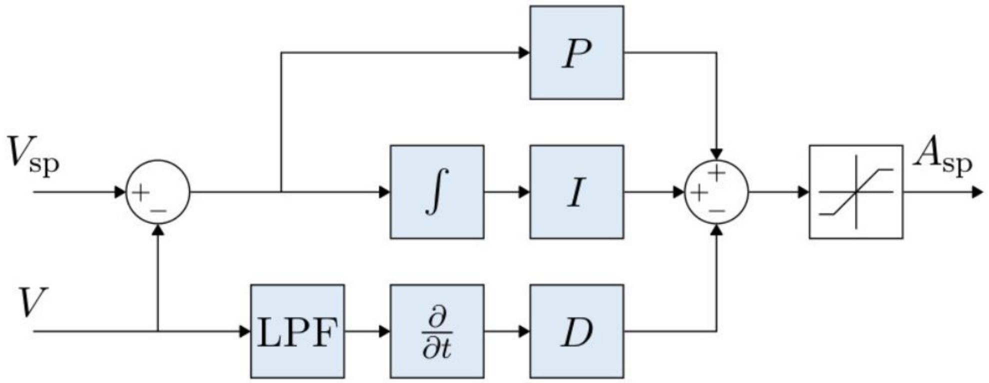
Position & Velocity Control

Multicopter Position Controller



- Simple P controller that commands a velocity.
- The commanded velocity is saturated to keep the velocity in certain limits.

Multicopter Velocity Controller

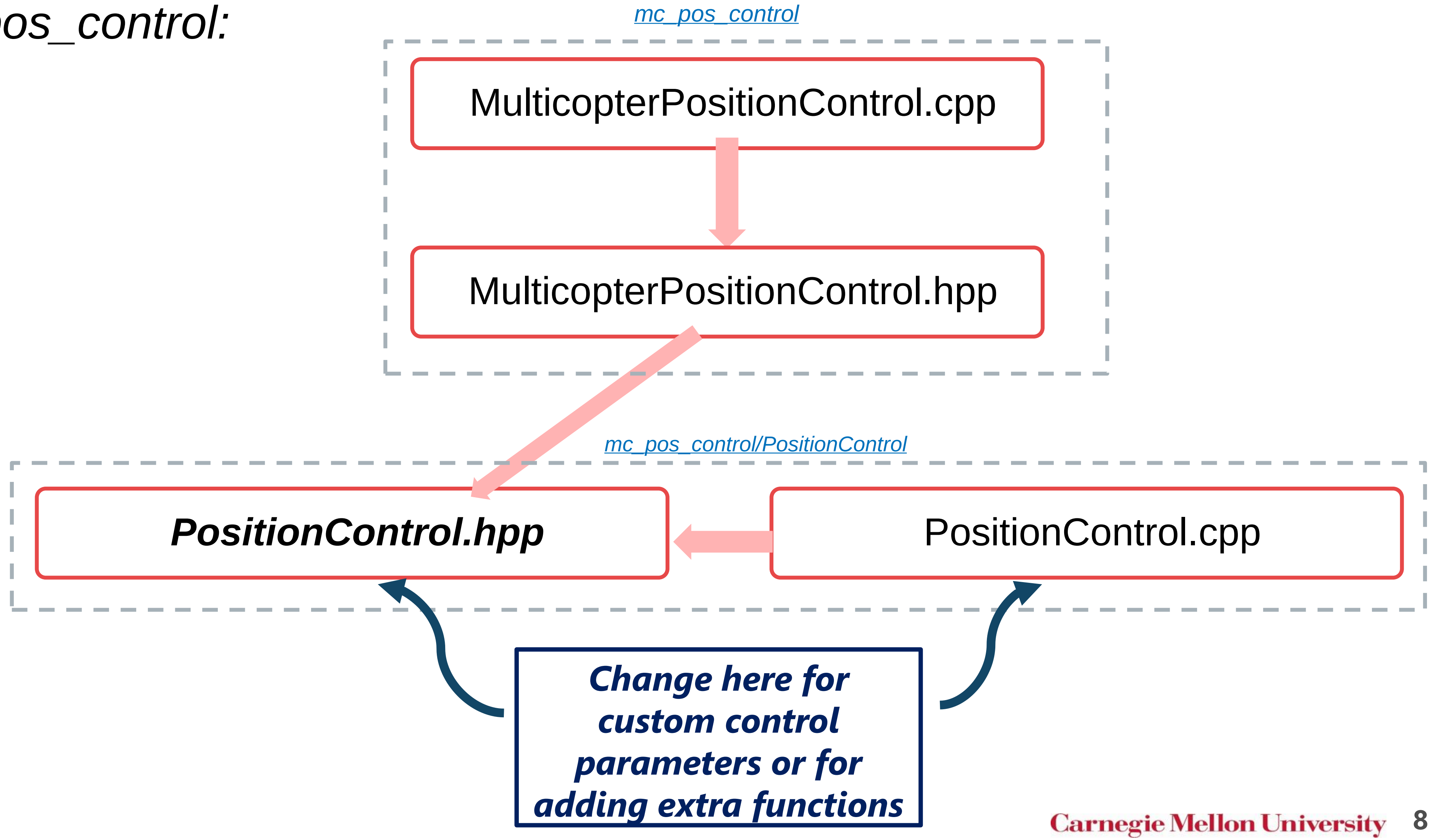


- PID controller to stabilise velocity. Commands an acceleration.
- The integrator includes an anti-reset windup (ARW) using a clamping method.
- The commanded acceleration is saturated.

http://docs.px4.io/master/en/flight_stack/controller_diagrams.html

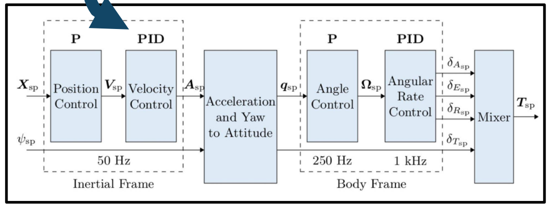
Codeflow

- *mc_pos_control:*



Setpoint Hierarchy

- If position-setpoint && velocity-setpoint true:
  - (Velocity component of P-Controller) >>> (feed-forward component from velocity-setpoint)
- If position/velocity-setpoint && thrust-setpoint true:
  - thrust-setpoint omitted and recomputed from next cascade/step in the architecture (i.e. Velocity PID controller)

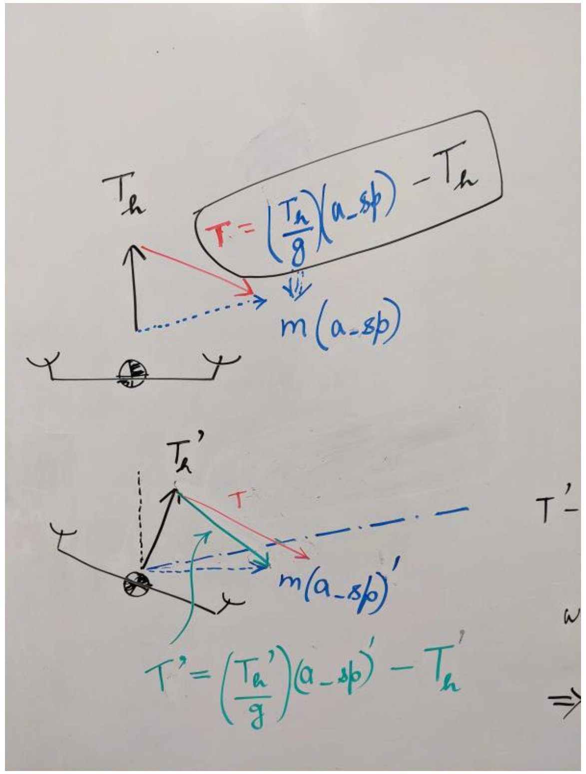


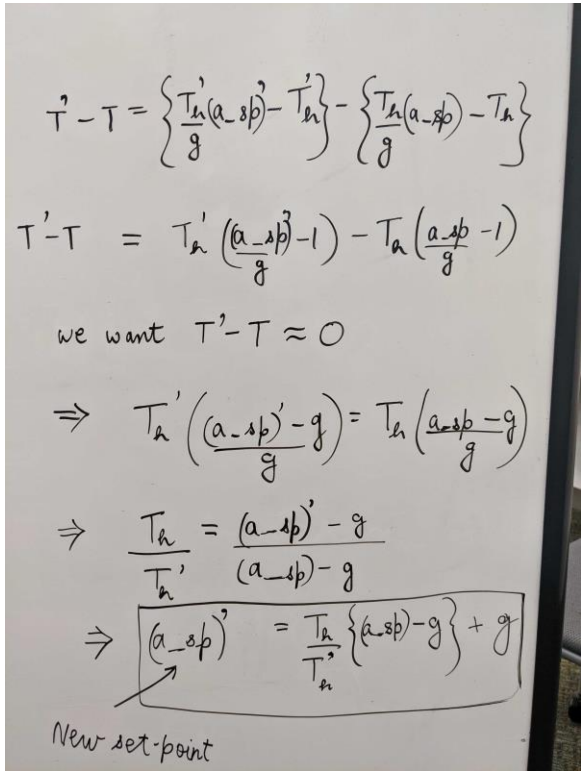

```
};

/**
 *      Core Position-Control for MC.
 *      This class contains P-controller for position and
 *      PID-controller for velocity.
 *      Inputs:
 *              vehicle position/velocity/yaw
 *              desired set-point position/velocity/thrust/yaw/yaw-speed
 *              constraints that are stricter than global limits
 *      Output
 *              thrust vector and a yaw-setpoint
 *
 *      If there is a position and a velocity set-point present, then
 *      the velocity set-point is used as feed-forward. If feed-forward is
 *      active, then the velocity component of the P-controller output has
 *      priority over the feed-forward component.
 *
 *      A setpoint that is NAN is considered as not set.
 *      If there is a position/velocity- and thrust-setpoint present, then
 *  the thrust-setpoint is ommitted and recomputed from position-velocity-PID-loop.
 */
class PositionControl
```

mc_pos_control/PositionControl/PositionControl.hpp

Hover Thrust Update

- void PositionControl::updateHoverThrust(const float hover_thrust_new)
    - Line 73-85

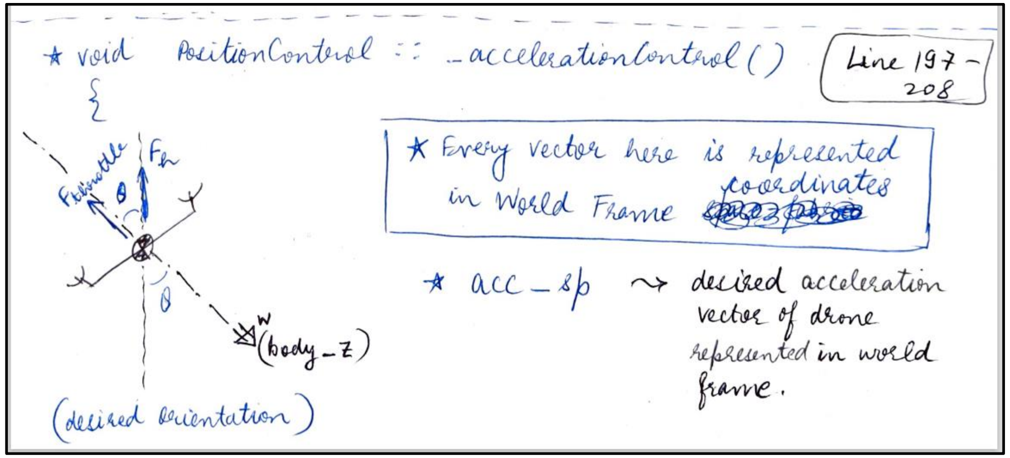


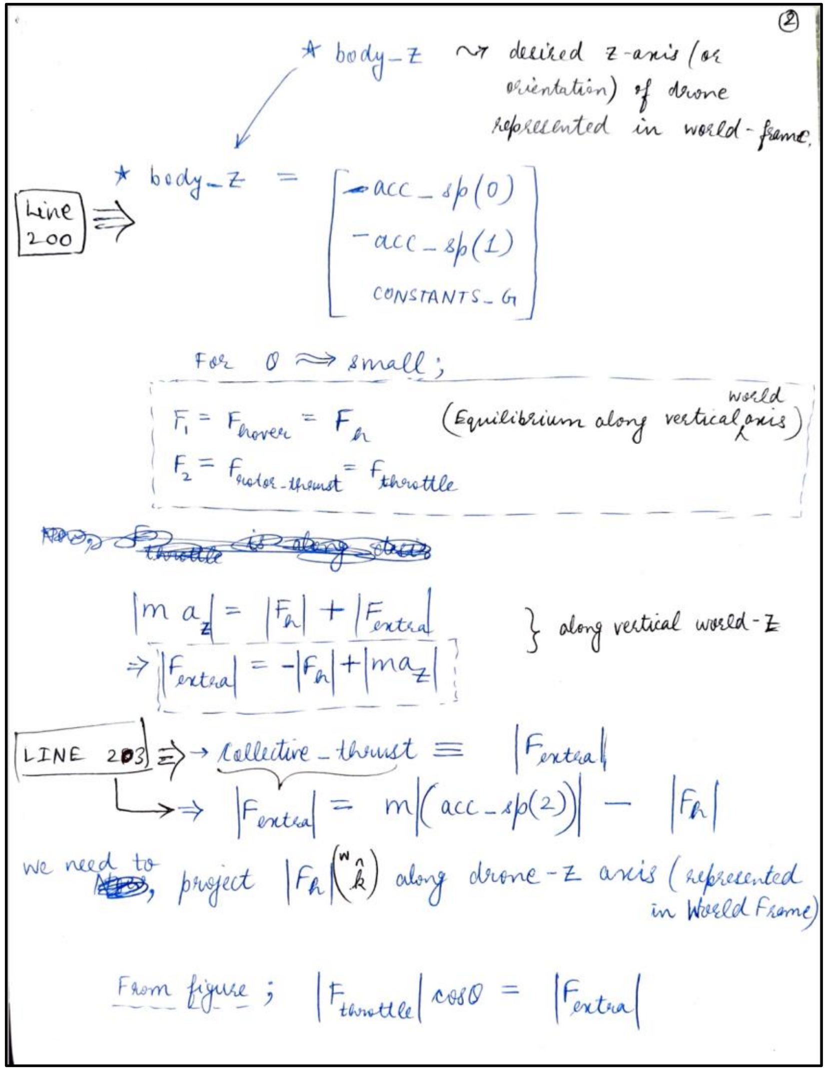


mc_pos_control/PositionControl/PositionControl.cpp

Thrust Setpoint and Acceleration Setpoint (1)

- Output of Velocity-PID block
- void PositionControl::_accelerationControl
  - Line 197-208
  - This is called earlier in void PositionControl::_velocityControl()

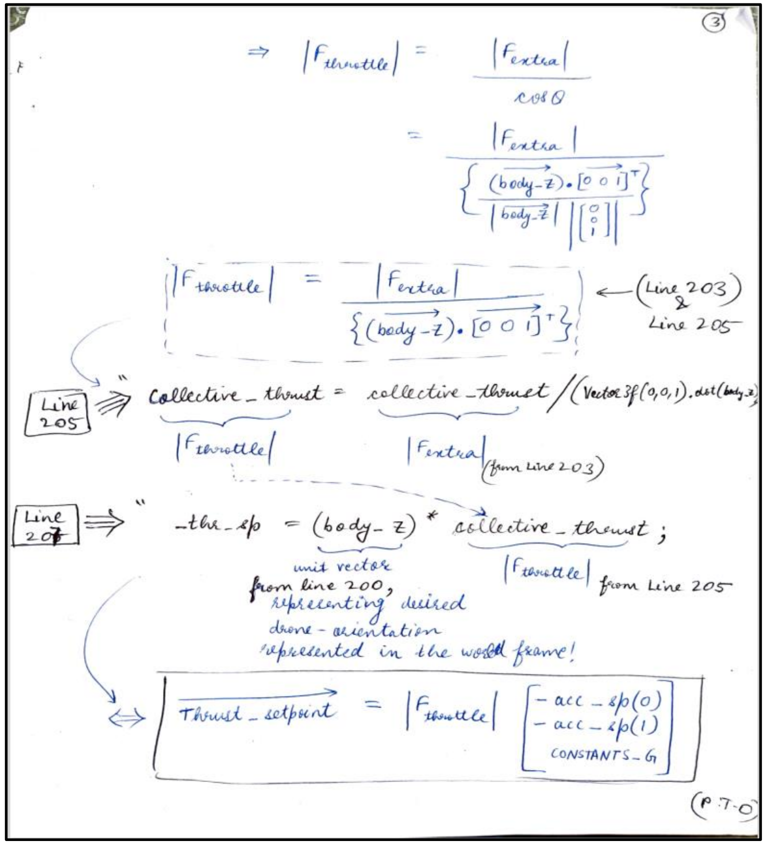




mc_pos_control/PositionControl/PositionControl.cpp

Thrust Setpoint and Acceleration Setpoint (2)

- Output of Velocity-PID block
- void PositionControl::_accelerationControl
  - Line 197-208
  - This is called earlier in void PositionControl::_velocityControl()



3

$$\Rightarrow |F_{throttle}| = \frac{|F_{extra}|}{\cos\theta} = \frac{|F_{extra}|}{\left\{\frac{(\overrightarrow{body\_z})\cdot\overrightarrow{[0\ 0\ 1]}^T}{|\overrightarrow{body\_z}|\ \left|\begin{bmatrix}0\\0\\1\end{bmatrix}\right|}\right\}}$$

$$|F_{throttle}| = \frac{|F_{extra}|}{\{(\overrightarrow{body\_z})\cdot\overrightarrow{[0\ 0\ 1]}^T\}}$$ ← (Line 203 & Line 205)

Line 205 ⇒ " $\underbrace{\text{collective\_thrust}}_{|F_{throttle}|} = \underbrace{\text{collective\_thrust}}_{|F_{extra}|\ \text{(from Line 203)}} / (\text{Vector3f(0,0,1).dot(body\_z)})$

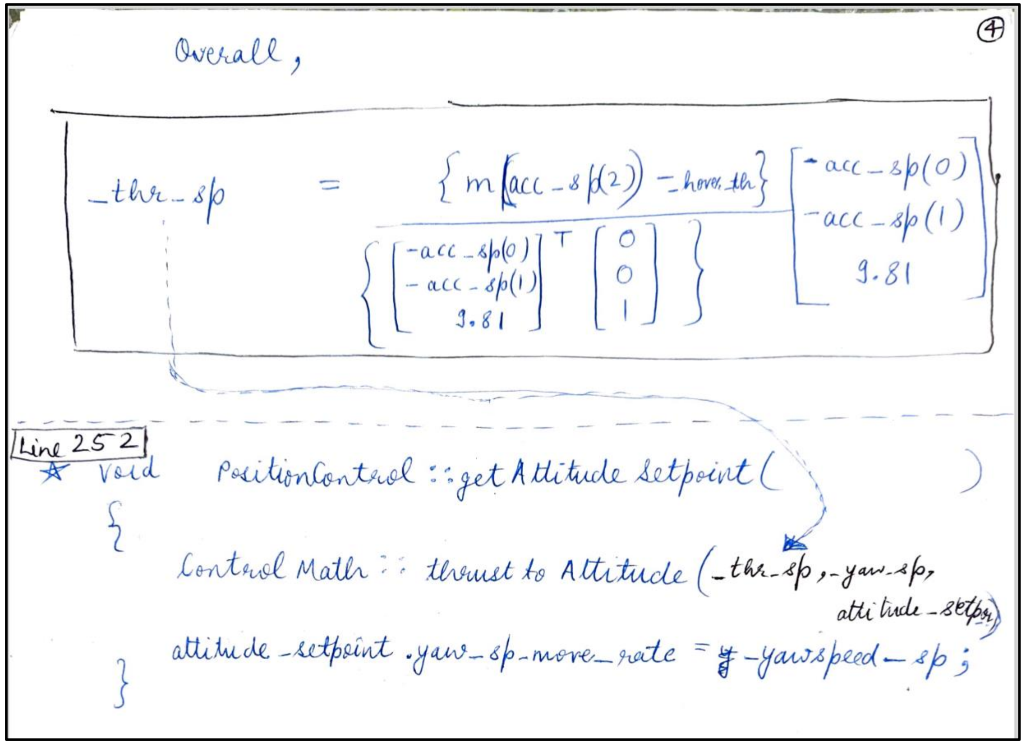
Line 207 ⇒ " $\text{\_thr\_sp} = (\text{body\_z}) * \underbrace{\text{collective\_thrust}}_{|F_{throttle}|}$;

unit vector from line 200, representing desired drone-orientation represented in the world frame!

$|F_{throttle}|$ from Line 205

$$\Leftrightarrow \boxed{\overrightarrow{\text{Thrust\_setpoint}} = |F_{throttle}| \begin{bmatrix}-acc\_sp(0)\\-acc\_sp(1)\\\text{CONSTANTS\_G}\end{bmatrix}}$$

(P.T.O)



4

Overall,

$$\boxed{\text{\_thr\_sp} = \frac{\{m(acc\_sp(2)) - \text{\_hover\_thr}\}}{\left\{\begin{bmatrix}-acc\_sp(0)\\-acc\_sp(1)\\9.81\end{bmatrix}^T\begin{bmatrix}0\\0\\1\end{bmatrix}\right\}}\begin{bmatrix}-acc\_sp(0)\\-acc\_sp(1)\\9.81\end{bmatrix}}$$

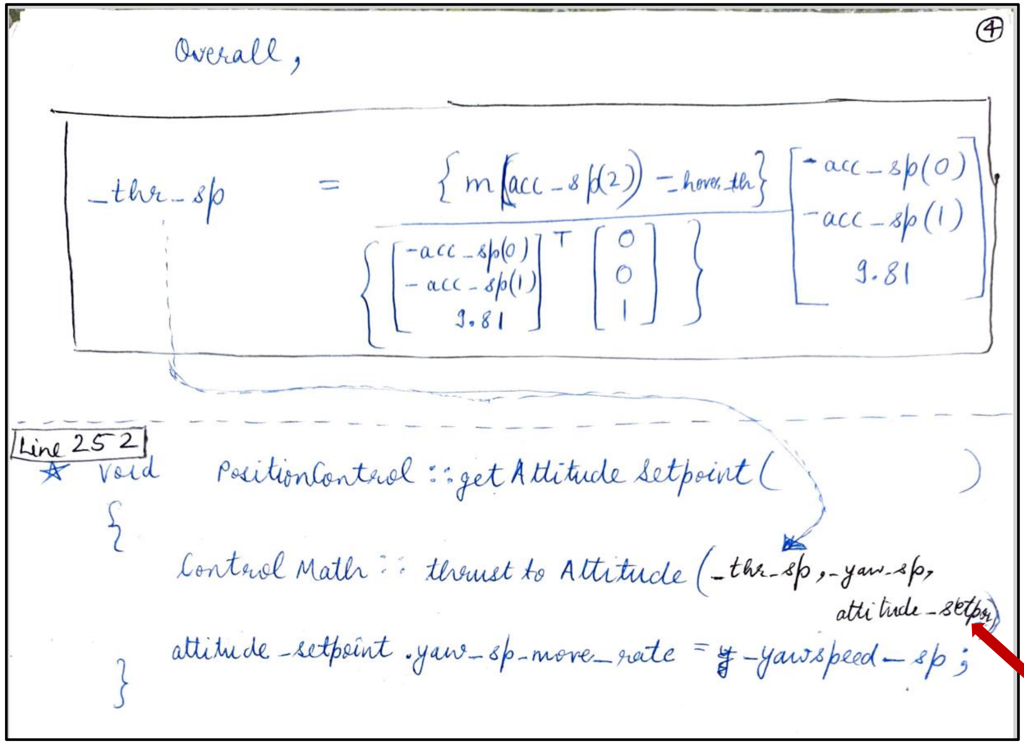
Line 252

★ void PositionControl::getAttitudeSetpoint( )
{
ControlMath::thrustToAttitude(_thr_sp, _yaw_sp, attitude_setpoint);
attitude_setpoint.yaw_sp_move_rate = _yawspeed_sp;
}

mc_pos_control/PositionControl/PositionControl.cpp

Thrust Setpoint To Rotation Matrix

- /PositionControl/ControlMath.cpp
- void thrustToAttitude()
  - Line 49
- void bodyzToAttitude()
  - Line 70 onwards
  - **"R" -> Line 104 – 108:**
    - ***Mapping from body-frame coordinates to world-coordinates....according to Zac's notation -> X_world = R*X_body.***



This is probably just a message topic

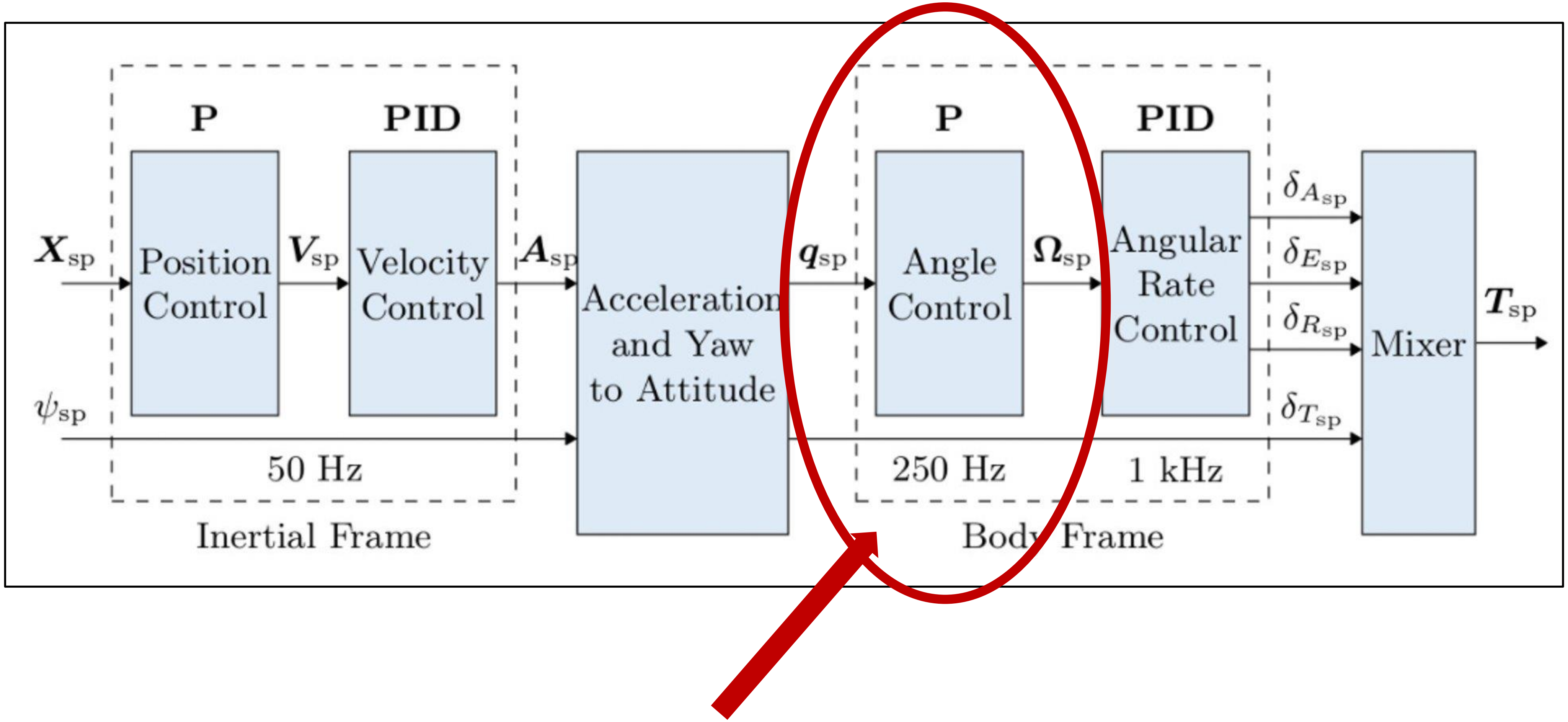
mc_pos_control/PositionControl/PositionControl.cpp

How is all of this called?

- */mc_pos_control/*MulticopterPositionControl.cpp
- void MulticopterPositionControl::Run()
  - _**control** object, throughout this file, refers to *"/PositionControl/PositionControl.cpp"*
  - Line 474 and 495 calls _**control.update** that triggers the Position-Control algorithm!

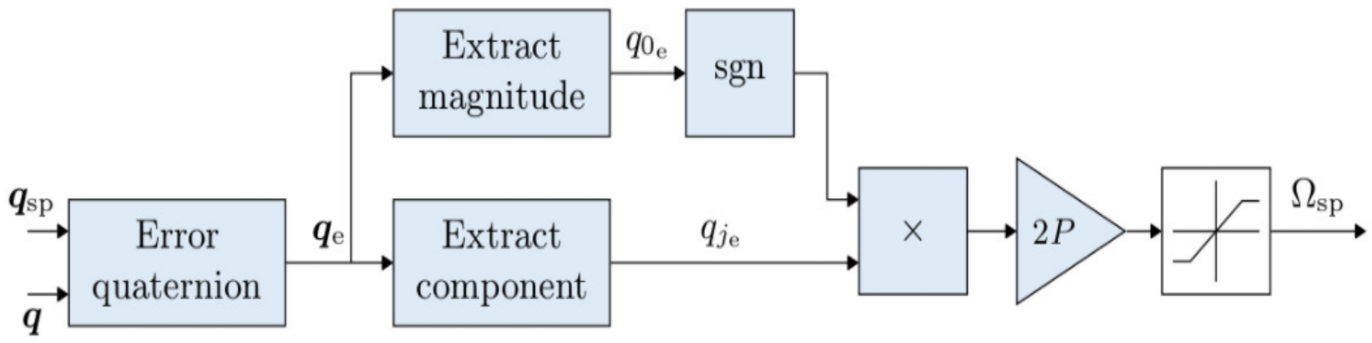
*mc_pos_control/*MulticopterPositionControl.cpp

Attitude Control



Attitude Control

Multicopter Attitude Controller

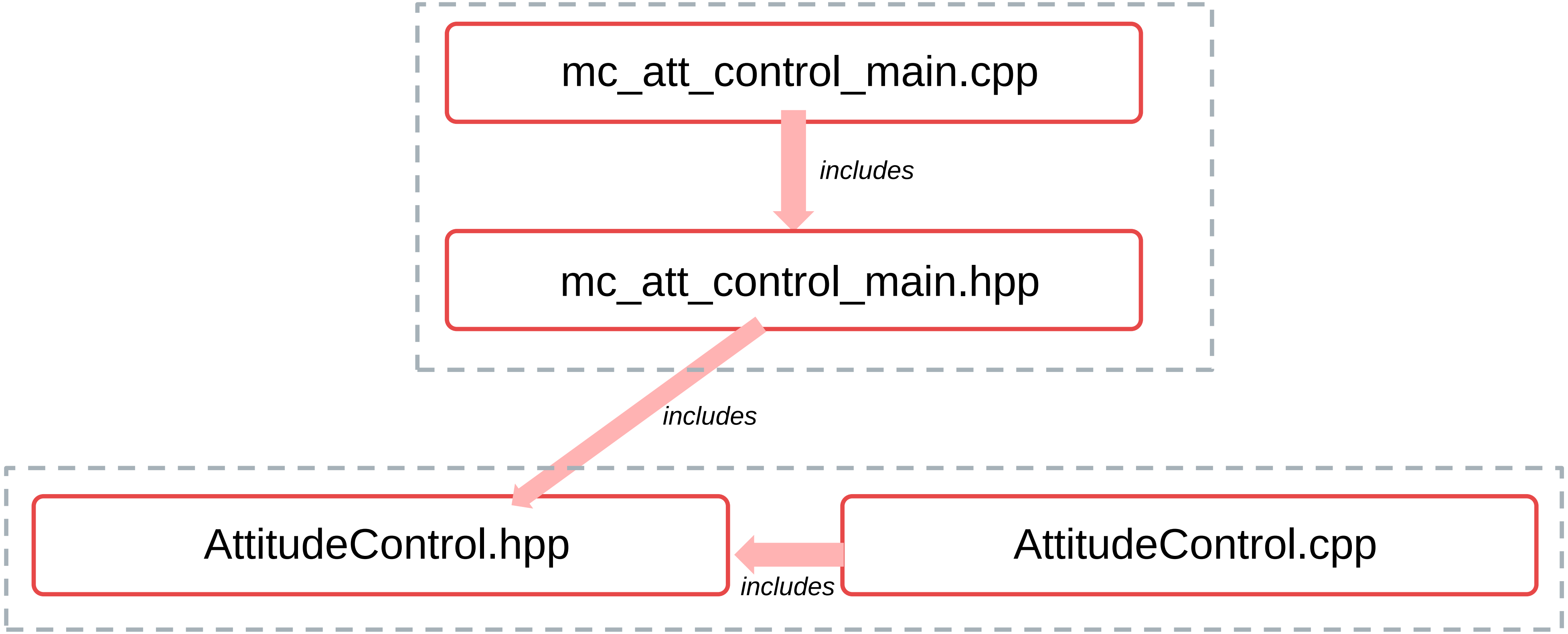


- The attitude controller makes use of quaternions .
- The controller is implemented from this article .
- When tuning this controller, the only parameter of concern is the P gain.
- The rate command is saturated.

http://docs.px4.io/master/en/flight_stack/controller_diagrams.html
***Refer this link for an additional note on IMU pipeline*

Codeflow

- *mc_att_control:* *(quaternion based control :* https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/154099/eth-7387-01.pdf*)*



Mc_att_control_main.cpp

generate_attitude_setpoint

- */mc_att_control/mc_att_control_main*.cpp
- void MulticopterAttitudeControl::generate_attitude_setpoint()
  - Line 155 – 157
    - ✓ *For axis angle representation, refer this: 1)* https://en.wikipedia.org/wiki/Axis%E2%80%93angle_representation
    - ✓ 2) https://github.com/Optimal-Control-16-745/lecture-notebooks-2021/blob/main/Lecture%2013/Lecture%2013.pdf
  - Line 165, 166, 177 spits out the Roll, Pitch and Yaw setpoints -->

- attitude_setpoint.roll_body
- attitude_setpoint.pitch_body
- attitude_setpoint.yaw_body

- void MulticopterAttitudeControl::Run()
  - **Line 304 – 308**
    - **Generates attitude setpoints if we are in Manual/Stabilized mode!**
  - Line 310 calls "*generate_attitude_setpoint()"* mentioned above
  - Line 318 calls "*_attitude_control.update(q)"* . This belongs to "AttitudeControl.cpp" and is discussed in next slides

AttitudeControl.cpp

AttitudeControl.cpp

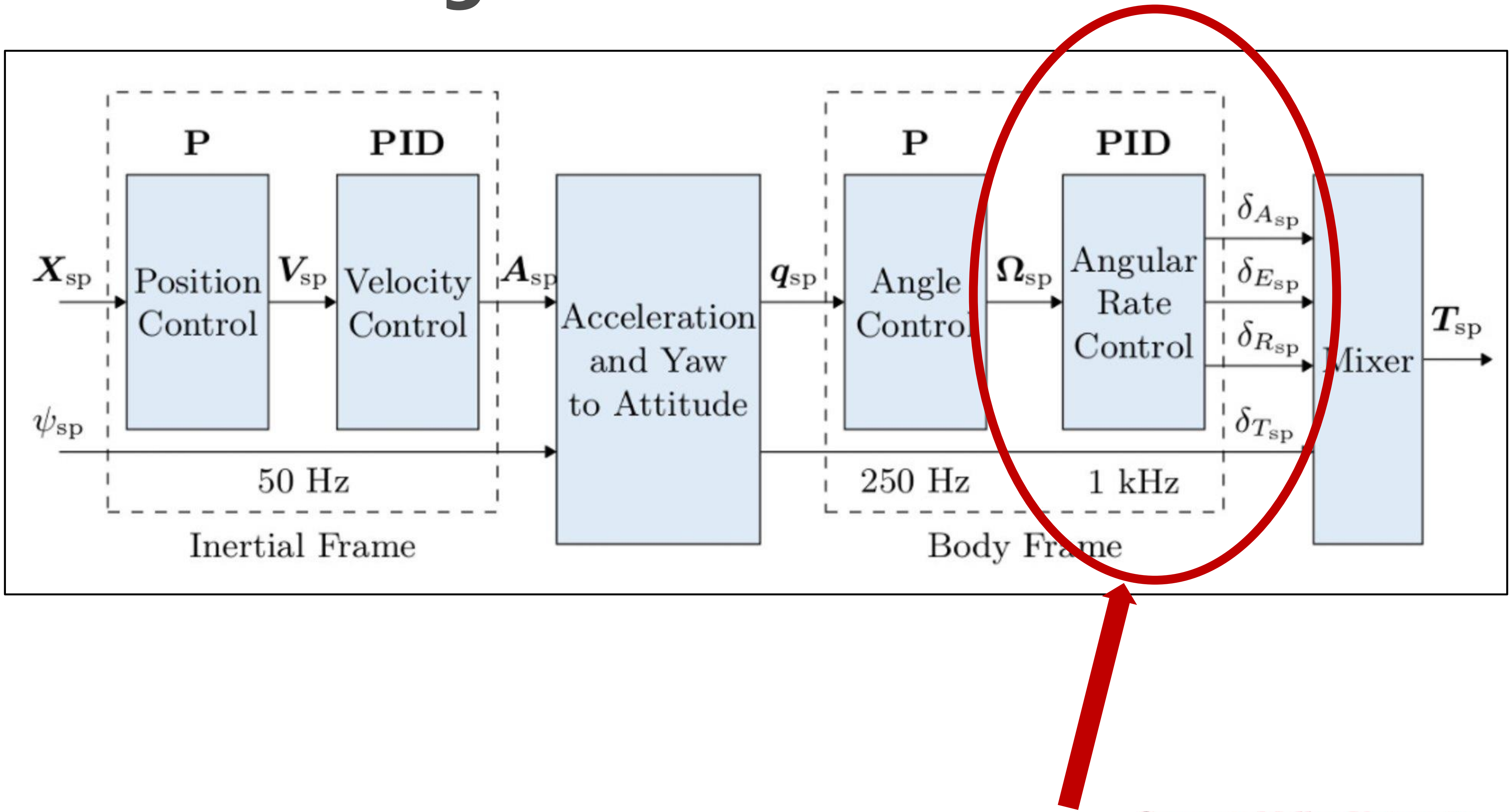
- */mc_att_control/AttitudeControl/AttitudeControl.cpp*
- void AttitudeControl::setProportionalGain()
- void AttitudeControl::update()
  - Refer https://github.com/PX4/PX4-Autopilot/blob/master/src/lib/matrix/matrix/Quaternion.hpp

*mc_att_control/*M.cpp

Quaternion Starter-Pack

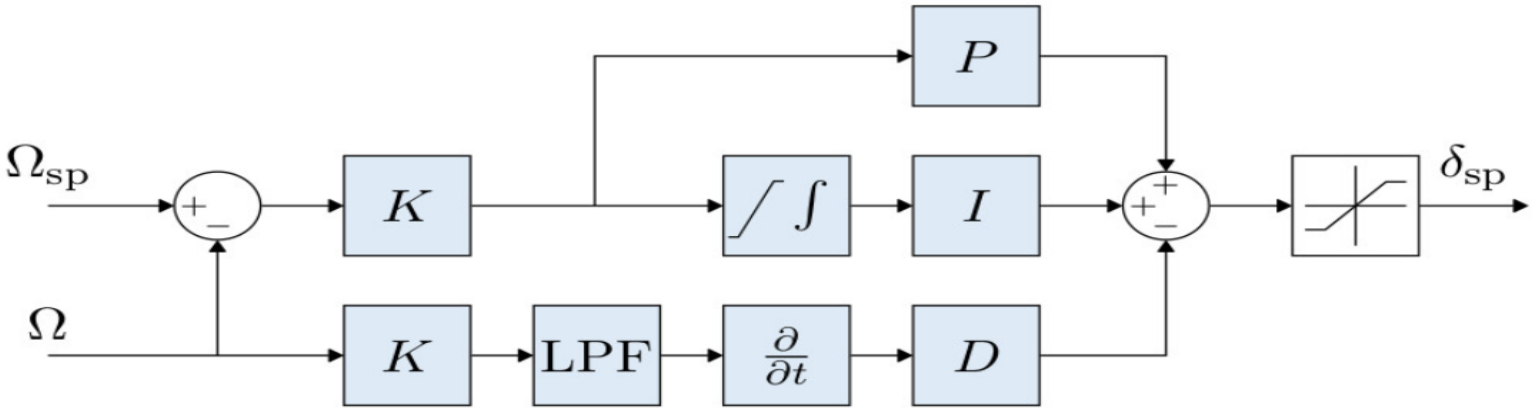
- https://www.youtube.com/watch?v=zjMuIxRvygQ
- https://github.com/Optimal-Control-16-745/lecture-notebooks-2021/blob/main/Lecture%2013/Lecture%2013.pdf
- https://github.com/Optimal-Control-16-745/lecture-notebooks-2021/blob/main/Lecture%2014/Lecture%2014.pdf
- https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/154099/eth-7387-01.pdf
- https://ieeexplore.ieee.org/document/9326337

Angular Rate Control



Angular Rate Control

Multicopter Angular Rate Controller



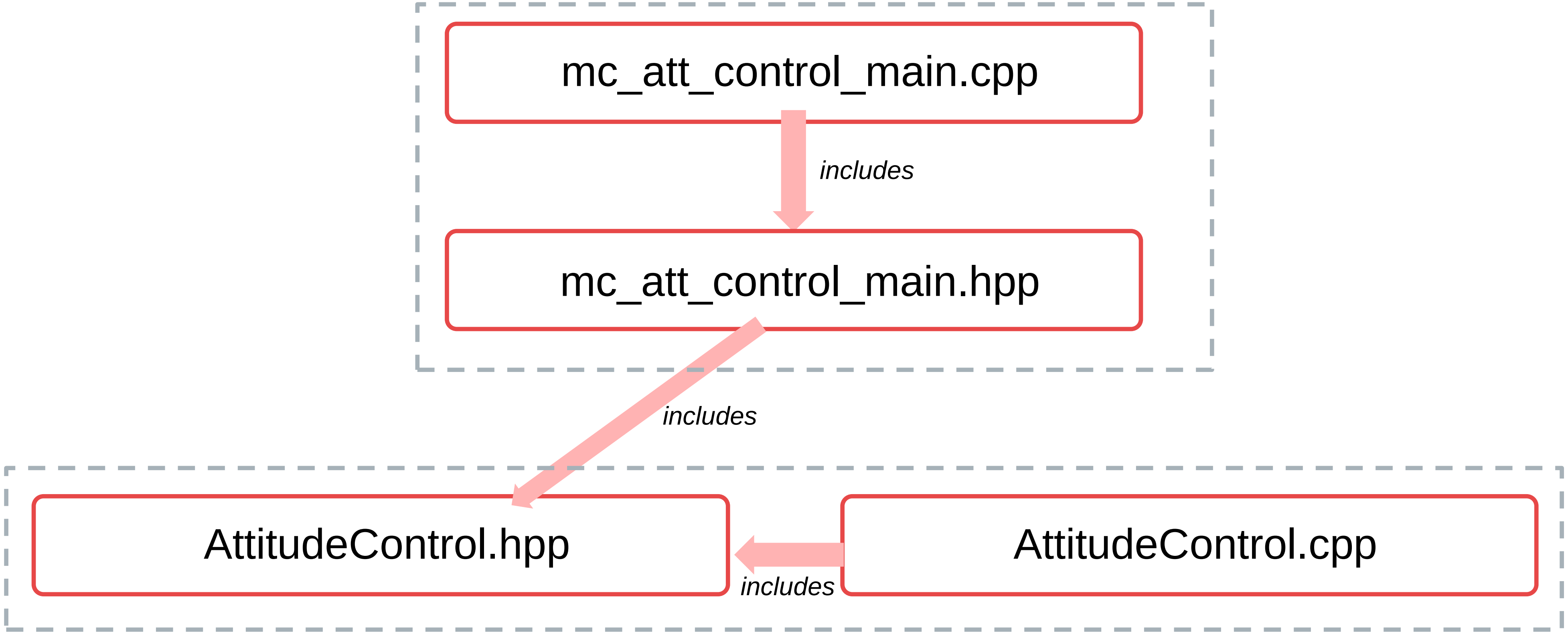
- K-PID controller. See Rate Controller for more information.
- The integral authority is limited to prevent wind up.
- The outputs are limited (in the mixer), usually at -1 and 1.
- A Low Pass Filter (LPF) is used on the derivative path to reduce noise (the gyro driver provides a filtered derivative to the controller).

http://docs.px4.io/master/en/flight_stack/controller_diagrams.html

***Refer this link for an additional note on IMU pipeline*

Codeflow

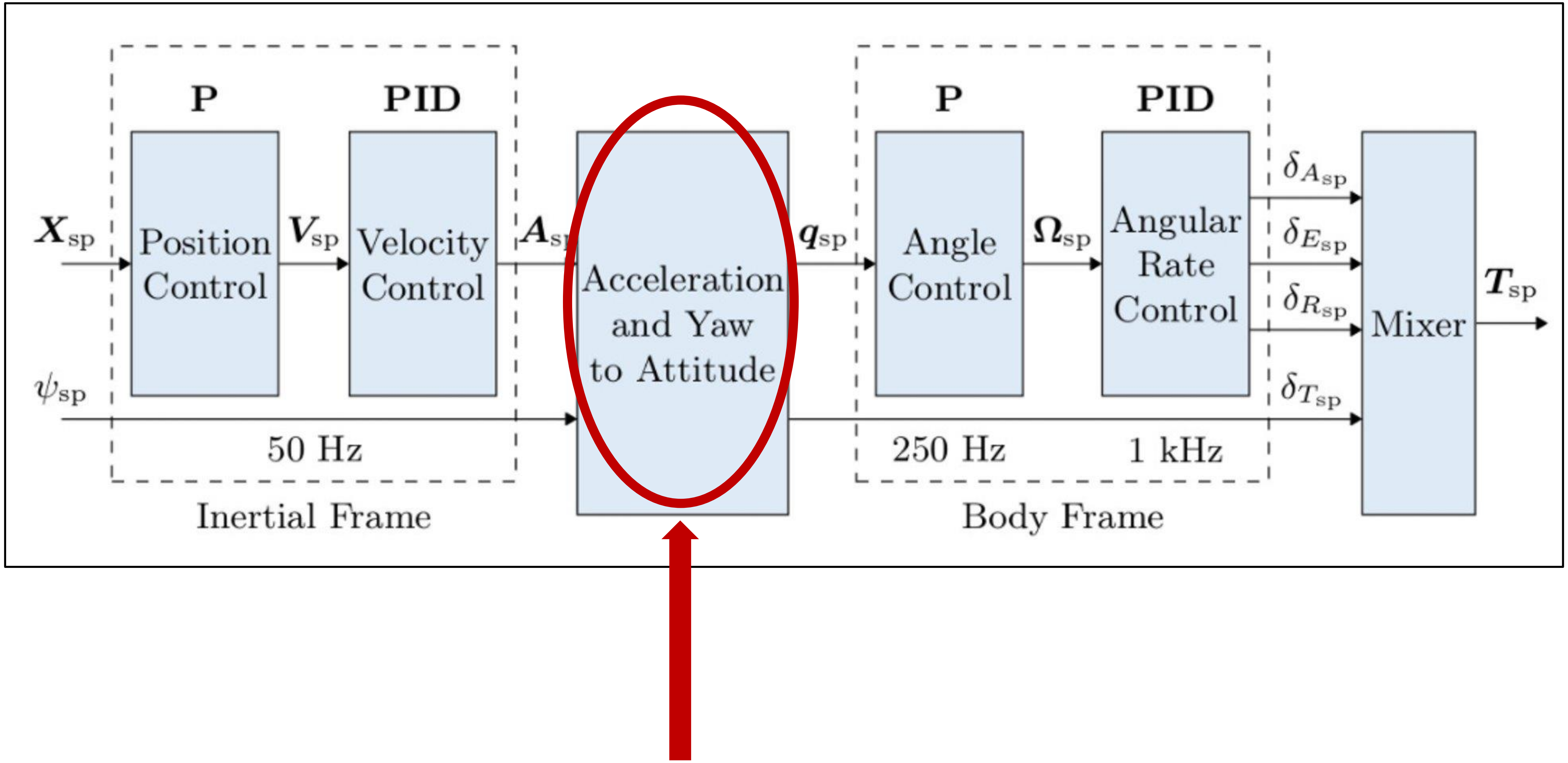
- *mc_rate_control:* *(quaternion based control :* https://www.research-collection.ethz.ch/bitstream/handle/20.500.11850/154099/eth-7387-01.pdf *)*



MulticopterRateControl.cpp

RateControl.cpp

Attitude Setpoint



UUV Control Architecture *(coming soon…)*