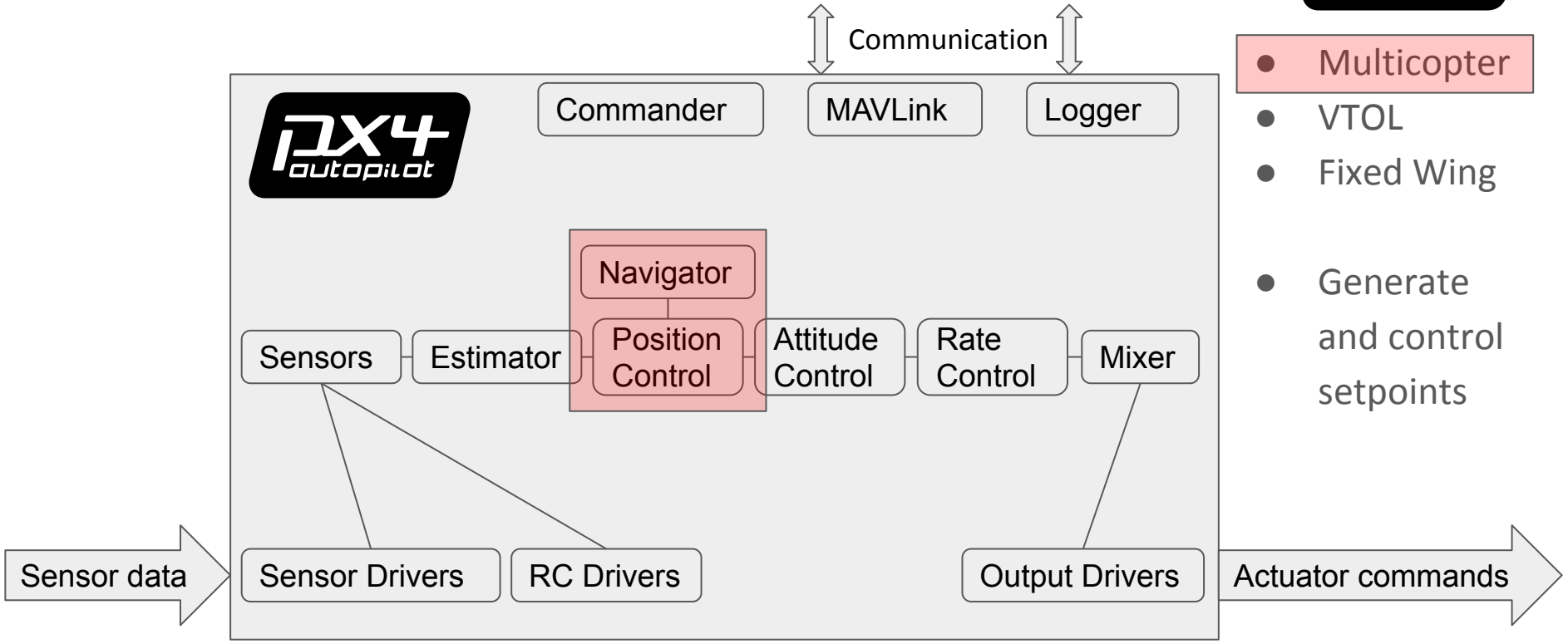
FlightTask Architecture

Introduction / QA

Dennis Mannhart
YUNEEC RESEARCH

Matthias Grob
auterion

Entire System Overview



- Multicopter
- VTOL
- Fixed Wing

- Generate and control setpoints

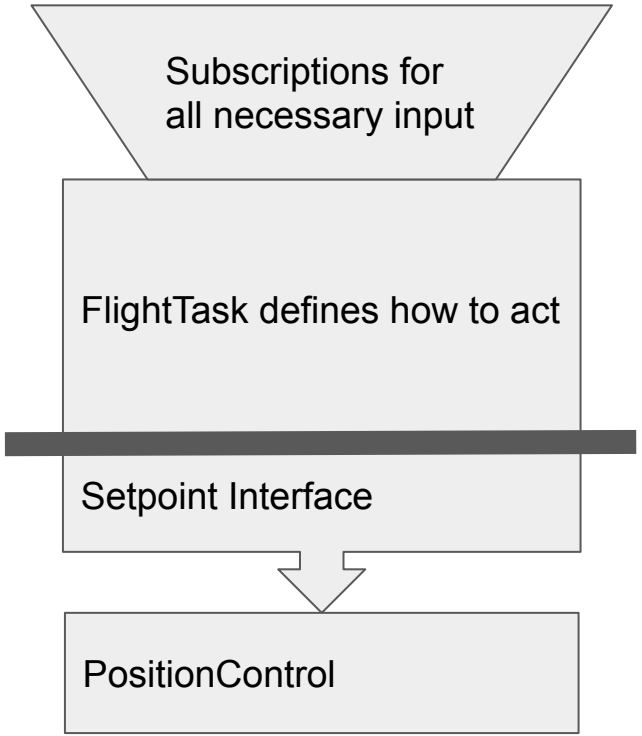
Why change anything?

- Position controller
  center of behavior?
- Module class
  reached 3.5k lines
- Several flight modes scattered
  all over the file
- Very hard to debug

```
1488  void
1489  MulticopterPositionControl::control_non_manual()
1490  {
1491          /* select control source */
1492          if (_control_mode.flag_control_offboard_enabled) {
1493                  /* offboard control */
1494                  control_offboard();
1495                  _mode_auto = false;
1496
1497          } else {
1498                  _hold_offboard_xy = false;
1499                  _hold_offboard_z = false;
1500
1501                  /* AUTO */
1502                  control_auto();
1503          }
1504
1505          // guard against any bad velocity values
1506          bool velocity_valid = PX4_ISFINITE(_pos_sp_triplet.current.vx) &&
1507                                PX4_ISFINITE(_pos_sp_triplet.current.vy) &&
1508                                _pos_sp_triplet.current.velocity_valid;
1509
1510          // do not go slower than the follow target velocity when position tracking is active (set to valid)
1511          if (_pos_sp_triplet.current.type == position_setpoint_s::SETPOINT_TYPE_FOLLOW_TARGET &&
1512              velocity_valid &&
1513              _pos_sp_triplet.current.position_valid) {
1514
1515                  math::Vector<3> ft_vel(_pos_sp_triplet.current.vx, _pos_sp_triplet.current.vy, 0);
1516
1517                  float cos_ratio = (ft_vel * _vel_sp) / (ft_vel.length() * _vel_sp.length());
1518
1519                  // only override velocity set points when uav is traveling in same direction as target and vector component
1520                  // is greater than calculated position set point velocity component
```

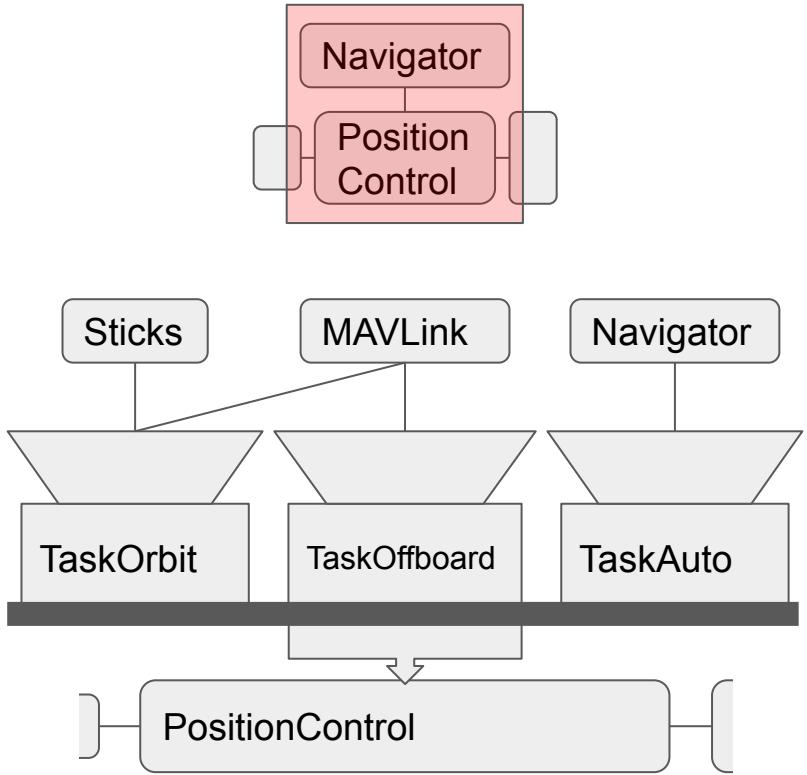
Idea behind FlightTask Architecture

- Simplify addition of new behavior
  - Modularity, debuggability
- ⟶ Library with one class defining one **task**
  - Problems have limited scope
- Clear output **interface**
- Separate setpoint generation from core PID control
  - PositionControl class for core
- Task can report error
  - Implicit failsafe
- Limit overhead on MCU



Where does it go?

- Library in `Firmware/src/lib/FlightTasks`
- Instantiated in position control module
- Before core position controller
- Interfaces are uORB messages



FlightTask Output - PositionControl Input

Setpoint
`vehicle_local_position_setpoint`
Local world frame

- **3D position**
- **3D velocity**
- **3D acceleration [WIP]**
- 3D jerk [log]
- 3D thrust
- Yaw (heading)
- Yawspeed

Constraints
`vehicle_constraints`

- **Horizontal speed**
- **Speed up**
- **Speed down**
- Yawspeed
- Tilt (roll & pitch)
- Minimum distance to ground
- Maximum distance to ground
- Takeoff trigger

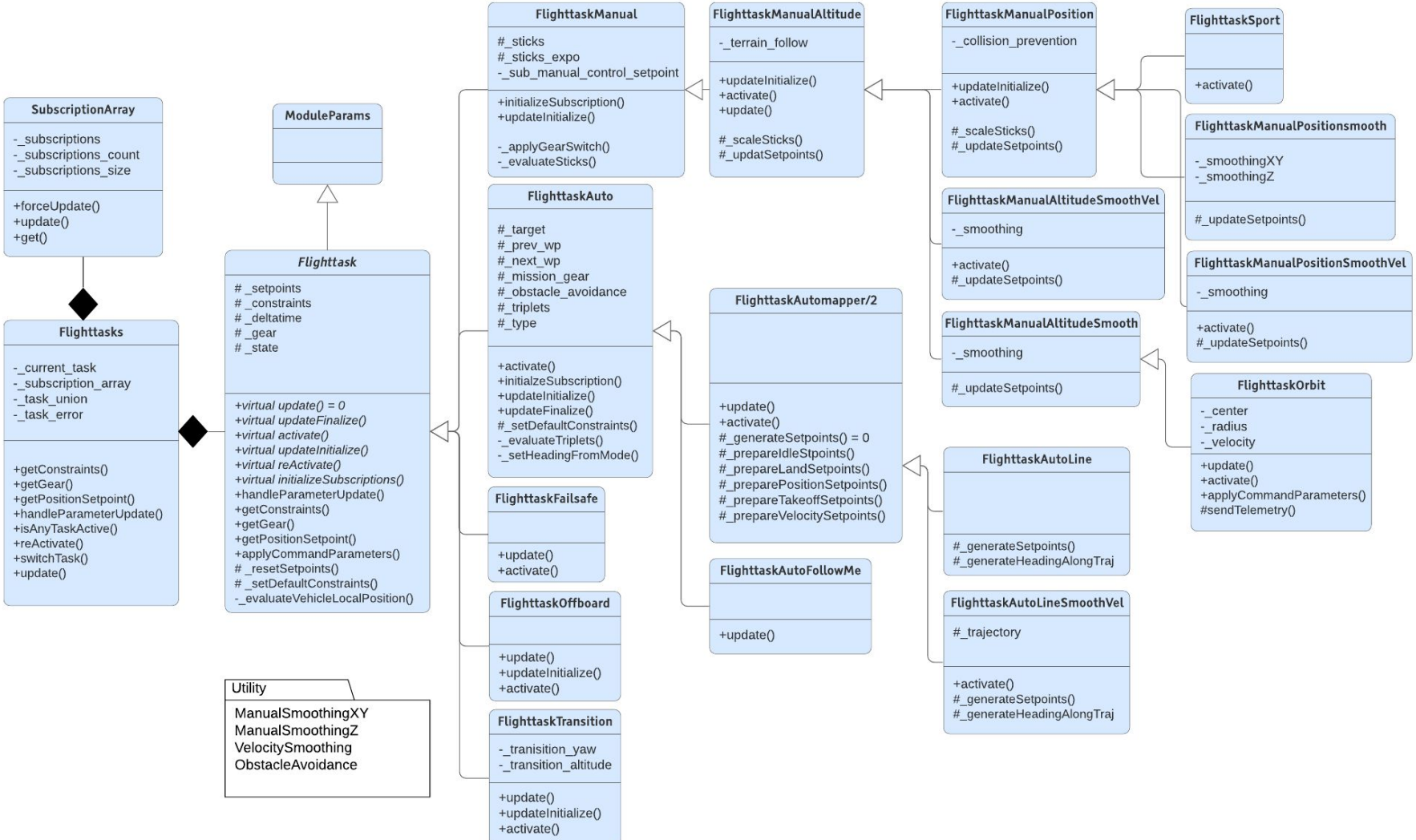
- Enables trajectories
- Any setpoint combination
- NAN - not set

- Setpoints logged before and after execution

Flighttasks Library Key Concepts

- Only one flight-task can run at a time
- One memory slot for all flight-tasks: size = largest flight-task
- Factory-Class: **Flighttasks**
- Base Class: **Flighttask**
- Core flight-tasks: Fully integrated into PX4 with dedicated PX4-flight modes
- Added flight-tasks: can be triggered via MAVLink Commands (MAV_CMD) (example: Orbit)
- Creating new flight-tasks: inheritance and utility classes

Architecture


SubscriptionArray
-_subscriptions
-_subscriptions_count
-_subscriptions_size
+forceUpdate()
+update()
+get()
ModuleParams
Flighttasks
-_current_task
-_subscription_array
-_task_union
-_task_error
+getConstraints()
+getGear()
+getPositionSetpoint()
+handleParameterUpdate()
+isAnyTaskActive()
+reActivate()
+switchTask()
+update()
Flighttask
_setpoints
_constraints
_deltatime
_gear
_state
+virtual update() = 0
+virtual updateFinalize()
+virtual activate()
+virtual updateInitialize()
+virtual reActivate()
+virtual initializeSubscriptions()
+handleParameterUpdate()
+getConstraints()
+getGear()
+getPositionSetpoint()
+applyCommandParameters()
_resetSetpoints()
_setDefaultConstraints()
-_evaluateVehicleLocalPosition()
Utility
ManualSmoothingXY
ManualSmoothingZ
VelocitySmoothing
ObstacleAvoidance
FlighttaskManual
#_sticks
#_sticks_expo
-_sub_manual_control_setpoint
+initializeSubscription()
+updateInitialize()
-_applyGearSwitch()
-_evaluateSticks()
FlighttaskManualAltitude
-_terrain_follow
+updateInitialize()
+activate()
+update()
#_scaleSticks()
#_updatSetpoints()
FlighttaskManualPosition
-_collision_prevention
+updateInitialize()
+activate()
#_scaleSticks()
#_updateSetpoints()
FlighttaskSport
+activate()
FlighttaskManualPositionsmooth
-_smoothingXY
-_smoothingZ
#_updateSetpoints()
FlighttaskManualAltitudeSmoothVel
-_smoothing
+activate()
#_updateSetpoints()
FlighttaskManualPositionSmoothVel
-_smoothing
+activate()
#_updateSetpoints()
FlighttaskManualAltitudeSmooth
-_smoothing
#_updateSetpoints()
FlighttaskOrbit
-_center
-_radius
-_velocity
+update()
+activate()
+applyCommandParameters()
#sendTelemetry()
FlighttaskAuto
#_target
#_prev_wp
#_next_wp
#_mission_gear
#_obstacle_avoidance
#_triplets
#_type
+activate()
+initialzeSubscription()
+updateInitialize()
+updateFinalize()
#_setDefaultConstraints()
-_evaluateTriplets()
-_setHeadingFromMode()
FlighttaskFailsafe
+update()
+activate()
FlighttaskOffboard
+update()
+updateInitialize()
+activate()
FlighttaskTransition
-_tranisition_yaw
-_transition_altitude
+update()
+updateInitialize()
+activate()
FlighttaskAutomapper/2
+update()
+activate()
#_generateSetpoints() = 0
#_prepareIdleStpoints()
#_prepareLandSetpoints()
#_preparePositionSetpoints()
#_prepareTakeoffSetpoints()
#_prepareVelocitySetpoints()
FlighttaskAutoFollowMe
+update()
FlighttaskAutoLine
#_generateSetpoints()
#_generateHeadingAlongTraj
FlighttaskAutoLineSmoothVel
#_trajectory
+activate()
#_generateSetpoints()
#_generateHeadingAlongTraj

Receipt for adding a flight-task to library


```
FlightTasks/
├── CMakeLists.txt
├── FlightTasks.cpp
├── FlightTasks.hpp
├── generate_flight_tasks.py
├── tasks
│   ├── Auto
│   ├── AutoFollowMe
│   ├── AutoLine
│   ├── AutoLineSmoothVel
│   ├── AutoMapper
│   ├── AutoMapper2
│   ├── Failsafe
│   ├── FlightTask
│   ├── Manual
│   ├── ManualAltitude
│   ├── ManualAltitudeSmooth
│   ├── ManualAltitudeSmoothVel
│   ├── ManualPosition
│   ├── ManualPositionSmooth
│   ├── ManualPositionSmoothVel
│   ├── Offboard
│   ├── Orbit
│   ├── Sport
│   ├── Transition
│   └── Utility
└── Templates
```

```
├── tasks
│   ├── Auto
│   │   ├── CMakeLists.txt
│   │   ├── FlightTaskAuto.cpp
│   │   └── FlightTaskAuto.hpp
│   ├── AutoFollowMe
│   │   ├── CMakeLists.txt
│   │   ├── FlightTaskAutoFollowMe.cpp
│   │   └── FlightTaskAutoFollowMe.hpp
│   ├── AutoLine
│   │   ├── CMakeLists.txt
│   │   ├── FlightTaskAutoLine.cpp
│   │   └── FlightTaskAutoLine.hpp
│   ├── AutoLineSmoothVel
│   │   ├── CMakeLists.txt
│   │   ├── FlightTaskAutoLineSmoothVel.cpp
│   │   └── FlightTaskAutoLineSmoothVel.hpp
│   ├── AutoMapper
│   │   ├── CMakeLists.txt
│   │   ├── FlightTaskAutoMapper.cpp
│   │   └── FlightTaskAutoMapper.hpp
```

```
FlightTasks/
├── tasks
│   ├── Orbit
│   │   ├── CMakeLists.txt
│   │   ├── FlightTaskOrbit.cpp
│   │   └── FlightTaskOrbit.hpp
```

FlightTasks/tasks/Orbit/CMakeLists.txt:

```
px4_add_library(FlightTaskOrbit
        FlightTaskOrbit.cpp
)
target_link_libraries(FlightTaskOrbit PUBLIC
FlightTaskManualAltitudeSmooth)
target_include_directories(FlightTaskOrbit PUBLIC
${CMAKE_CURRENT_SOURCE_DIR})
```

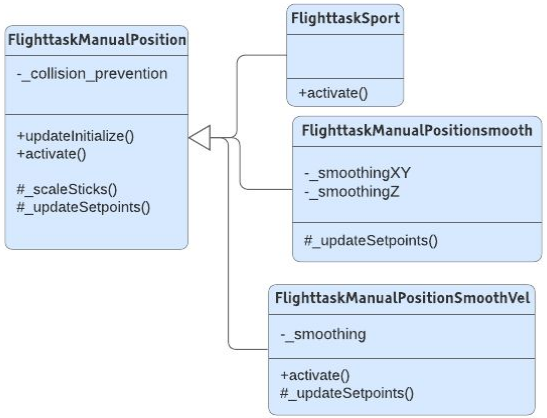
FlightTasks/CMakeLists.txt

```
list(APPEND flight_tasks_to_add
        Orbit
)
```

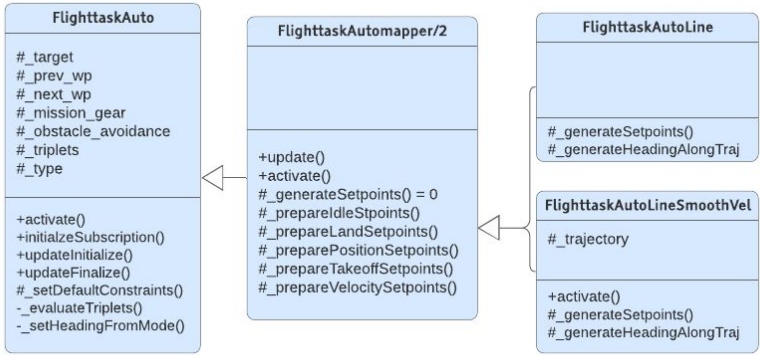
Receipt for triggering new flight-task

Extend Existing PX4-flight modes via Parameters

MPC_POS_MODE



MPC_AUTO_MODE



Receipt for triggering new flight-task

New PX4-flight mode via MAVLink Commands (example: Orbit)

1. Create a new Mavlink MAV_CMD command: MAV_CMD_DO_ORBIT (https://mavlink.io/en/messages/common.html#MAV_CMD_DO_ORBIT)
2. Add new mode to uORB messages:

- *vehicle_command.msg*: VEHICLE_CMD_DO_ORBIT
- *commander_state.msg*: MAIN_STATE_DO_ORBIT
- *vehicle_status.msg:* NAVIGATION_STATE_DO_ORBIT

3. *Commander.cpp* ***handle_command()***: add case vehicle_command_s::VEHICLE_CMD_DO_ORBIT

```
case vehicle_command_s::VEHICLE_CMD_DO_ORBIT:
        main_state_transition(*status_local, commander_state_s::MAIN_STATE_ORBIT, status_flags, &internal_state);
        break;
```

Receipt for triggering new flight-task

4. *statemachine_helper.cpp* ***main_state_transition()***: case MAIN_STATE_ORBIT

```
case commander_state_s::MAIN_STATE_ORBIT:
        if (status.vehicle_type == vehicle_status_s::VEHICLE_TYPE_ROTARY_WING) {
                ret = TRANSITION_CHANGED;
        }
        break;
```

5. *statemachine_helper.cpp* ***set_nav_state()***: case commander_state_s::MAIN_STATE_ORBIT

```
case commander_state_s::MAIN_STATE_ORBIT:
        if (status->engine_failure) {
                // failsafe: on engine failure
                status->nav_state = vehicle_status_s::NAVIGATION_STATE_AUTO_LANDENGFAIL;
        } else {
                // no failsafe, RC is not mandatory for orbit
                status->nav_state = vehicle_status_s::NAVIGATION_STATE_ORBIT;
        }
        break;
```

Receipt for triggering new flight-task

6. *Commander.cpp* ***set_control_mode()***: case vehicle_command_s::VEHICLE_CMD_DO_ORBIT

```
case vehicle_status_s::NAVIGATION_STATE_ORBIT:
        control_mode.flag_control_manual_enabled= false;
        control_mode.flag_control_auto_enabled= false;
        control_mode.flag_control_rates_enabled= true;
        control_mode.flag_control_attitude_enabled= true;
        control_mode.flag_control_rattitude_enabled= false;
        control_mode.flag_control_altitude_enabled= true;
        control_mode.flag_control_climb_rate_enabled= true;
        control_mode.flag_control_position_enabled= true;
        control_mode.flag_control_velocity_enabled= true;
        control_mode.flag_control_acceleration_enabled= false;
        control_mode.flag_control_termination_enabled= false;
        break;
```

7. *mc_pos_control_main.cpp* ***start_flight_task()****:* add case vehicle_status_s::NAVIGATION_STATE_ORBIT

```
if (_vehicle_status.nav_state == vehicle_status_s::NAVIGATION_STATE_ORBIT) {
        should_disable_task =false;
}
```

Example: Continuous yaw (trigger via MPC_AUTO_MODE)

- Trigger in Auto-mode
- Fly up and down 8 meters starting with upward flight
- Origin is set at trigger time
- Keep horizontal position constant
- Rotate with +-45 deg/s

Example: Continuous yaw (trigger via MPC_AUTO_MODE)

```
FlightTasks/
├── tasks
│   ├── Auto
│   ├── AutoFollowMe
│   ├── AutoLine
│   ├── AutoLineSmoothVel
│   ├── AutoMapper
│   ├── AutoMapper2
│   ├── ContinuousYaw
│   ├── Failsafe
│   ├── FlightTask
│   ├── Manual
│   ├── ManualAltitude
│   ├── ManualAltitudeSmooth
│   ├── ManualAltitudeSmoothVel
│   ├── ManualPosition
│   ├── ManualPositionSmooth
│   ├── ManualPositionSmoothVel
│   ├── Offboard
│   ├── Orbit
│   ├── Sport
│   ├── Transition
│   └── Utility
└── Templates
```

```
│   │   ├── ContinuousYaw
│   │   │   ├── CMakeLists.txt
│   │   │   ├── FlightTaskContinuousYaw.cpp
│   │   │   └── FlightTaskContinuousYaw.hpp
```

Firmware/src/lib/FlightTasks/tasks/ContinuousYaw/CMakeLists.txt

```
px4_add_library(FlightTaskContinuousYaw
        FlightTaskContinuousYaw.cpp
)
target_link_libraries(FlightTaskContinuousYaw PUBLIC FlightTask)
target_include_directories(FlightTaskContinuousYaw PUBLIC${CMAKE_CURRENT_SOURCE_DIR})
```

Firmware/src/lib/FlightTasks/CMakeLists.txt

```
# add core flight tasks to list
list(APPEND flight_tasks_all
            ManualAltitude
            ManualAltitudeSmooth
            ManualAltitudeSmoothVel
            ManualPosition
            ManualPositionSmooth
            ManualPositionSmoothVel
            Sport
            AutoLine
            AutoLineSmoothVel
            AutoFollowMe
            Offboard
            Failsafe
            Transition
            ContinuousYaw
            ${flight_tasks_to_add})
```

- Trigger in Auto-mode
- Fly up and down 8 meters starting with upward flight
- Origin is set at trigger time
- Keep horizontal position constant
- Rotate with +-45 deg/s

FlightTaskContinuousYaw.hpp

```
#pragma once

#include "FlightTask.hpp"

class FlightTaskContinuousYaw : public FlightTask
{
public:
	FlightTaskContinuousYaw() = default;
	virtual ~FlightTaskContinuousYaw() = default;

	bool update() override;
	bool activate() override;
private:
	float _origin_z = 0.0f;
};
```

FlightTaskContinuousYaw.cpp

```
#include "FlightTaskContinuousYaw.hpp"

bool FlightTaskContinuousYaw::activate()
{
	bool ret = FlightTask::activate();

	_position_setpoint(0) = _position(0);
	_position_setpoint(1) = _position(1);

	_origin_z = _position(2);

	_yawspeed_setpoint =45.0f * 3.142f / 180.f;
	_velocity_setpoint(2) = -1.0f; //NED frame

	return ret;
}

bool FlightTaskContinuousYaw::update()
{
	float diff_z = _position(2) - _origin_z;

	if (diff_z <= -8.0f) { //NED frame
		_velocity_setpoint(2) = 1.0f;
		_yawspeed_setpoint =45.0f * 3.142f / 180.f * -1.0f;

	} else if (diff_z >= 0.0f ) {
		_velocity_setpoint(2) = -1.0f;
		_yawspeed_setpoint =45.0f * 3.142f / 180.f;
	}
	return true;
}
```

Example: Continuous yaw (via Parameter)

Firmware/src/modules/mc_pos_control/mc_pos_control_params.c

```
/**
* Auto sub-mode
*
* @value 0 Default line tracking
* @value 1 Jerk-limited trajectory
* @value 2 Continuous Yaw
* @group Multicopter Position Control
*/
PARAM_DEFINE_INT32(MPC_AUTO_MODE, 1);
```

Firmware/src/modules/mc_pos_control/mc_pos_control_main.cpp

```
// Auto related tasks
switch (_param_mpc_auto_mode.get()) {
case 1:
        error =  _flight_tasks.switchTask(FlightTaskIndex::AutoLineSmoothVel);
        break;

case 2:
        error =  _flight_tasks.switchTask(FlightTaskIndex::ContinuousYaw);
        break;

default:
        error =  _flight_tasks.switchTask(FlightTaskIndex::AutoLine);
        break;
}
```

Todos

- Switching
  - Following commander navigation state vs following vehicle commands
  - Setpoint discontinuity can occur
- Move from inheritance towards libraries
  - Inheritance in this case less readable
  - Libraries allow completely free combination
- Simplify process of adding new task
  - Not depend on commander module changes
- Acceleration setpoint getting executed
  - Currently work in progress

Questions?

Answers

Dennis Mannhart
YUNEEC RESEARCH

Matthias Grob
auterion