Dronecode FOUNDATION

BUILD • LAUNCH • SOAR

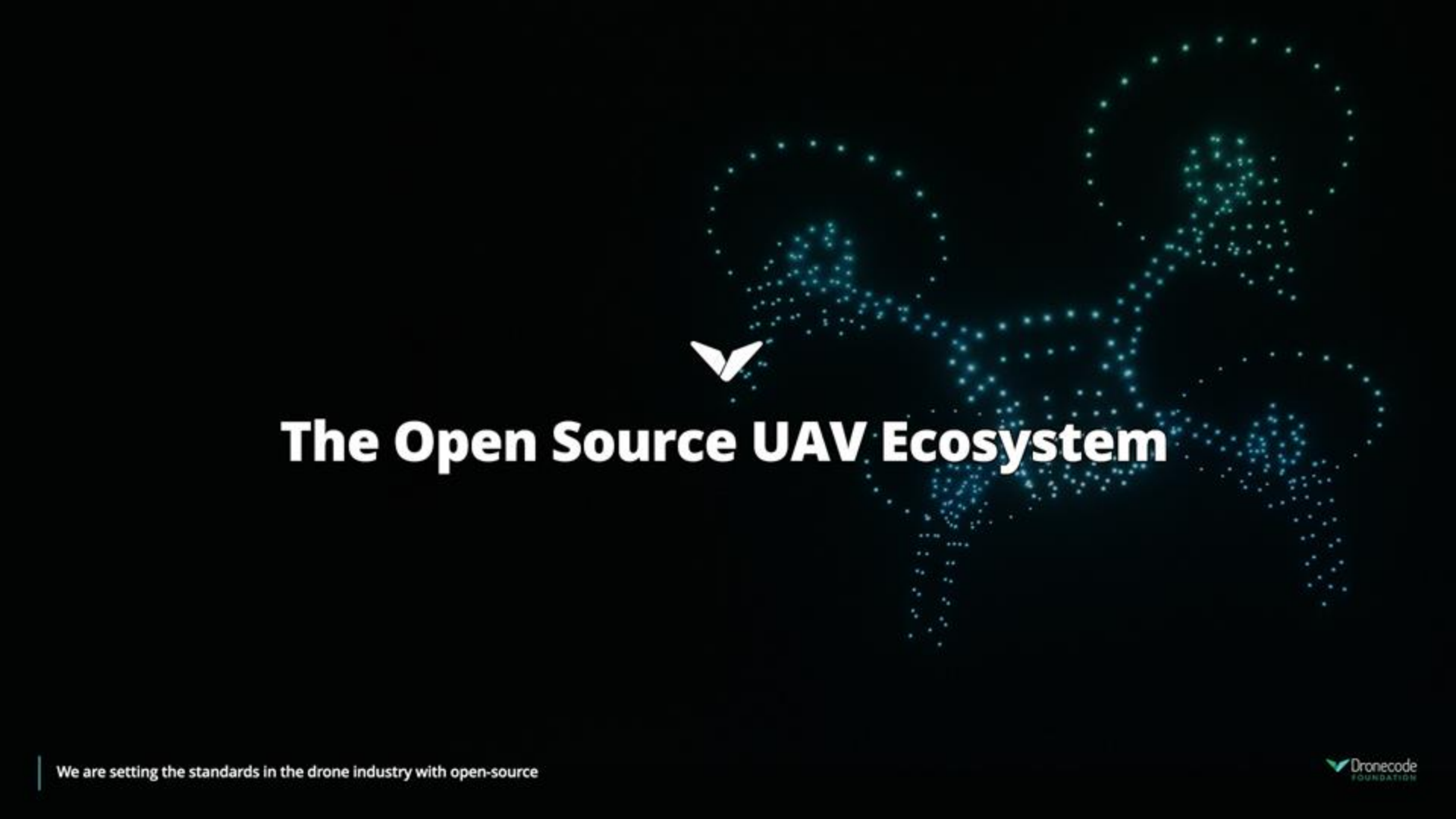
The infrastructure ecosystem for the development of autonomous aerial robotics

Ramón Roche
rroche@linuxfoundation.org

whoami

Ramón Roche
General Manager
Dronecode / Linux Foundation

- (Still) an individual contributor
- 10+ years working in aerial robotics
- Co-Lead ROS Aerial Robotics CWG
- Co-Lead Space Grade Linux SIG



The Open Source UAV Ecosystem

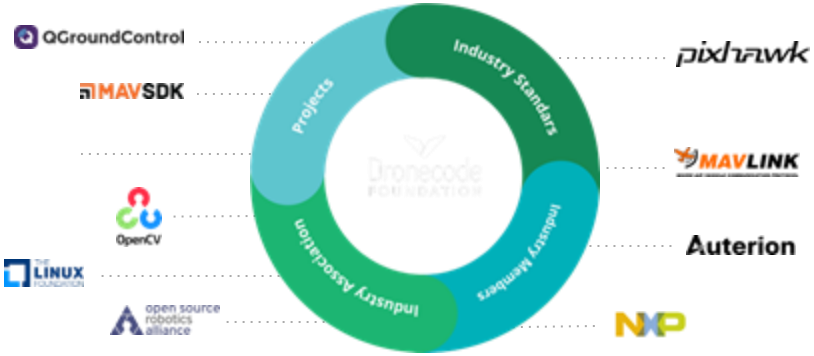
We are setting the standards in the drone industry with open-source

Dronecode FOUNDATION

The Dronecode Ecosystem

- **Open source** aerial robotics ecosystem
- **Global community** of 13,000+ contributors
- **Broad hardware, sensor and vehicle support** across air, land, and sea platforms
- **Vendor neutral governance** under the Linux Foundation
- **Permissively licensed** - BSD and Apache 2.0
- **Production ready** with industry-setting standards used by over 1 million drones worldwide

Celebrating 10 Years!



Member Community

These are the member companies that play an integral role in shaping the future of the industry.

Dronecode FOUNDATION

Platinum

Auterion

Silver

3DR

Ascend Engineering

ARK ELECTRONICS

CUAV

DRONEBLOCKS

DroneDeploy

FREEFLY

HHLA Sky

Holybro

ROBU.IN Your Ideas, Our Parts

MODAL AI ROBOTIC PERCEPTION

NEROS

NXP

RIIS Research Into Internet Systems

SIYI

sunflower labs

TII Technology Innovation Institute

UVify

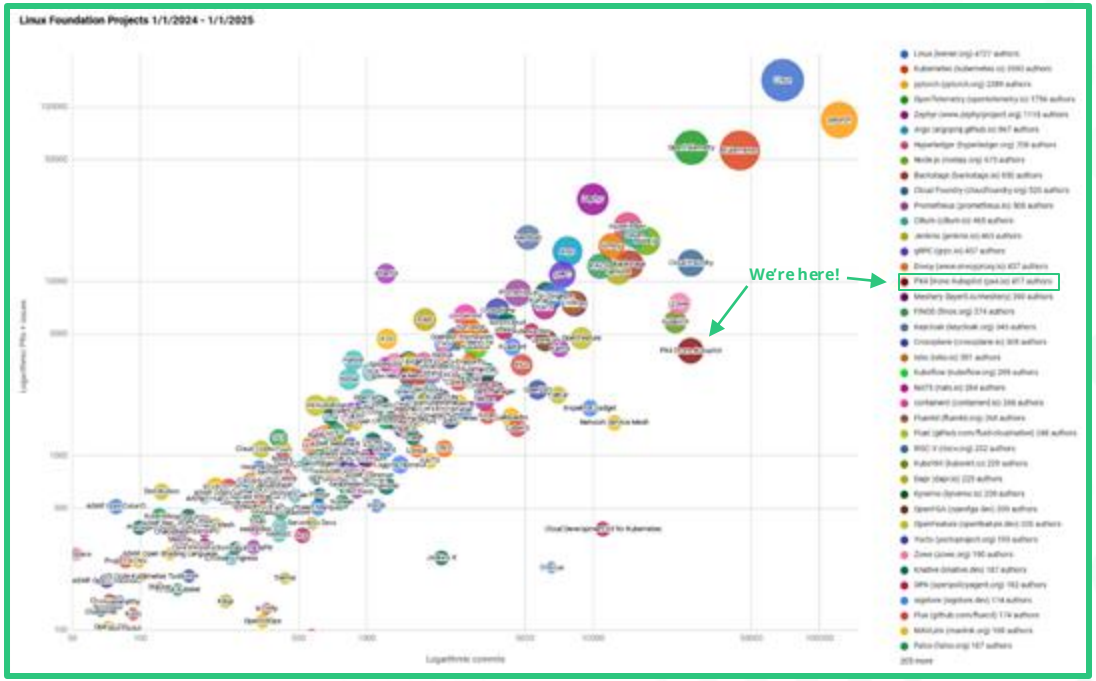
VAYDYN

ZERO ONE AeroSpace

How We Bring Value to Our Members

Dronecode FOUNDATION

PX4 Project Growth



16th Fastest Growing

(out of 240+ Linux Foundation projects)

Putting things into perspective

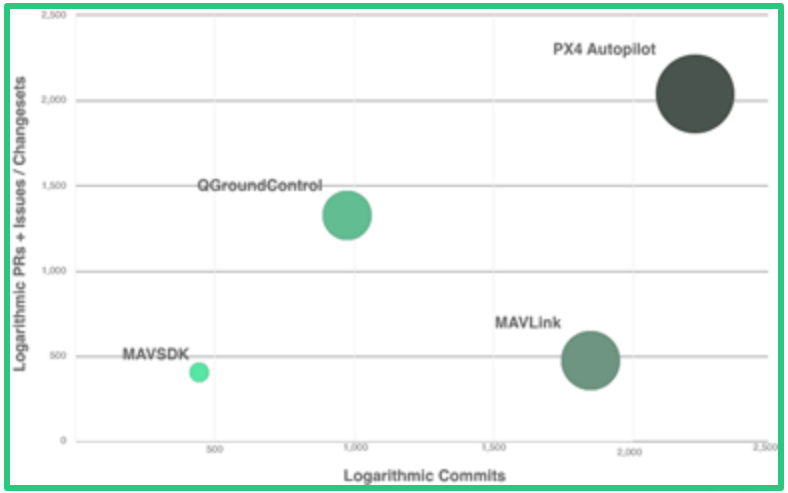
- $1B Project Value
- 59.72M Total Lines of Code
- 13,307 Total Unique Contributors

- +1,900 Contributors in 2024
- 100+ Git Repositories
- 20 Dronecode Members
- 5 Top-Level Open Source Projects

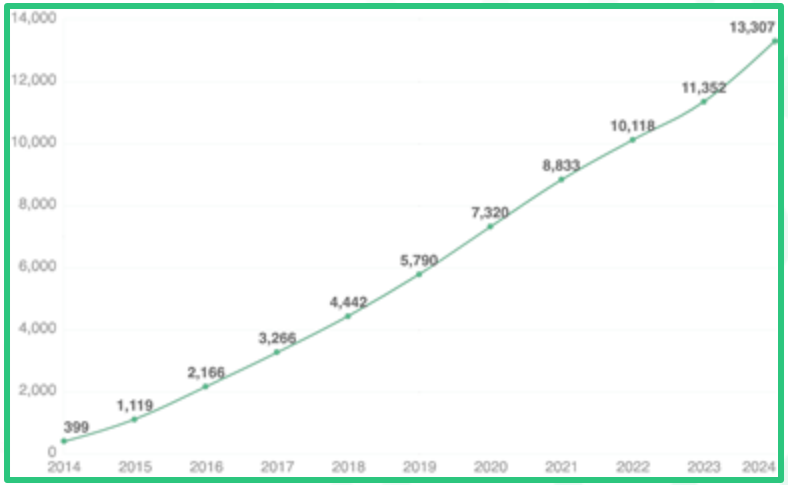
Data Provided by

LFX Insights

Dronecode Ecosystem Projects' Growth



Project Velocity 2024



Total Contributors Over Time

Data Provided by

LFX | Insights

Products Enabled by Dronecode, Built with Open Source

CUAV Pixhawk V6X

Auterion Skynode X

Holybro Pixhawk 6X-RT

ARKV6X

ModalAI Stinger

FreeFly Astro

BlueROV2

UVify IFO

DroneUp
FAA
CERTIFIED
OPERATION
STAND CLEAR

wingtra

GUINNESS
WORLD RECORDS
RECORD
HOLDER
Abbott
Elementary

See the PX4 YouTube Channel for Videos

https://www.youtube.com/@px4autopilot

Dronecode FOUNDATION

Dronecode Stack Overview

Drone Side: Main Components

Ethernet / Serial
DDS / MAVLink

STM32H7
RTOS
PX4 Autopilot

Jetson Orin NX / Nano
Linux
ROS / MAVSDK

Controller Side: Main Components

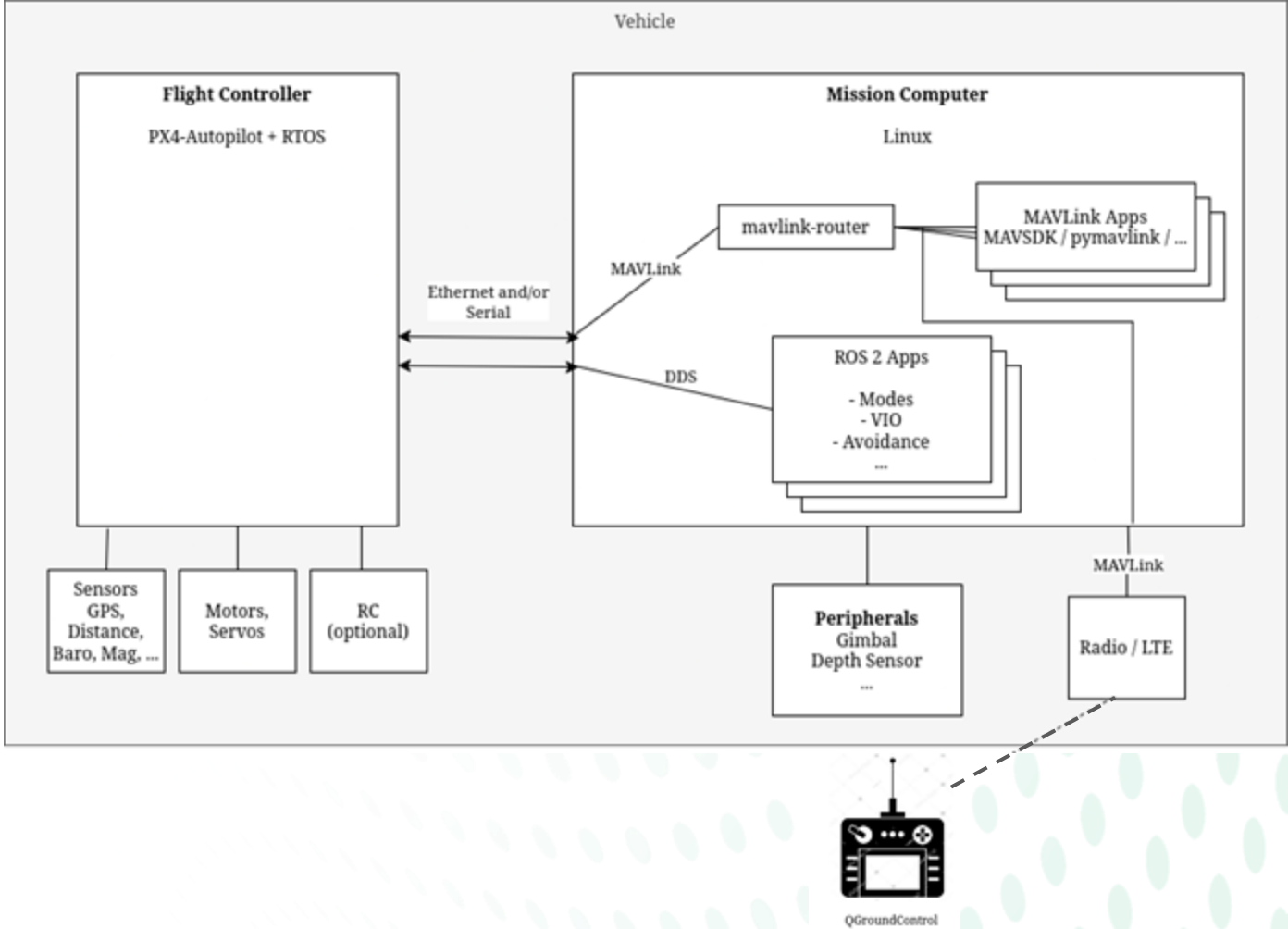
900 MHz / 2.4 GHz
DDS / MAVLink / Video

5G / LTE / WiFi / RC
Digital / Analog Links
DDS / MAVLink

Mobile Tablet
Android / iOS / Linux
DDS / MAVLink

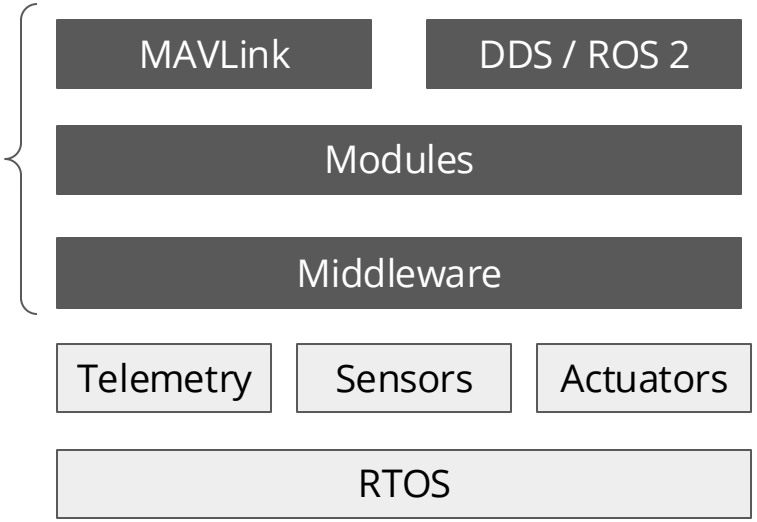
The Stack at a Glance

- Pixhawk
- PX4 Autopilot
- MAVLink
- MAVSDK
- QGroundControl
- ROS 2



PX4 Autopilot

- **Open Source** Autopilot Stack
- Developer friendly with **vibrant community** participation
- Support for a wide range of **boards** and **sensors**
- **Vendor Neutral** governance
- Business friendly license: **BSD-3**
- **Modular** architecture with a **DDS-compatible middleware** (uORB)
- Modules are fully **parallelized**, and **thread safe**
- Support for custom builds, **trim what you don't need**



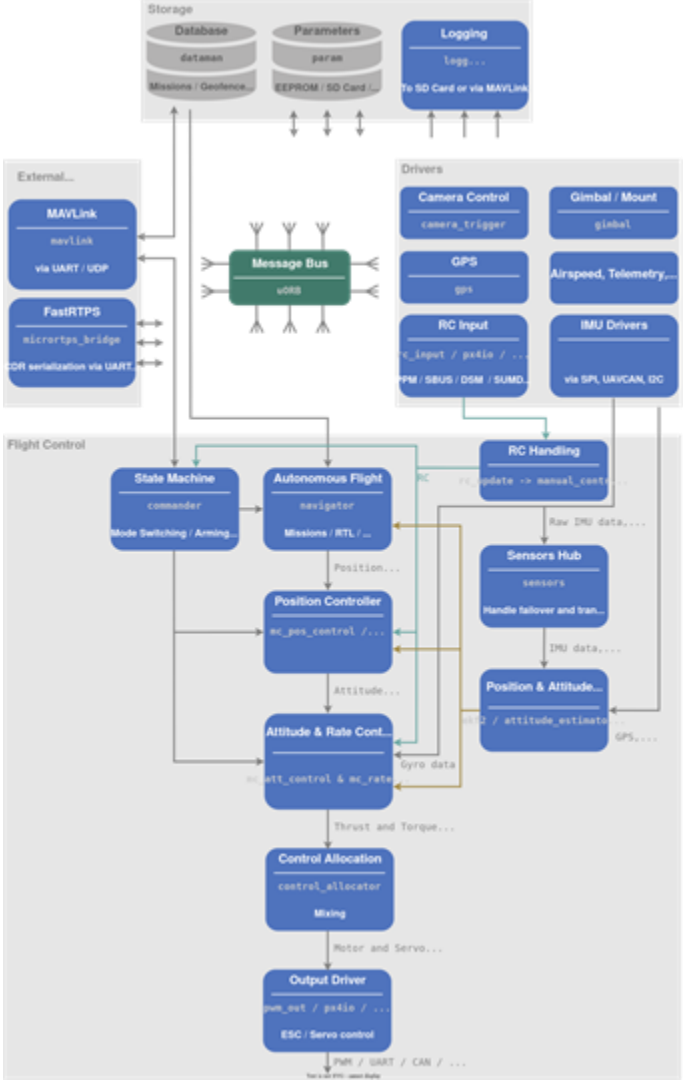
PX4 Autopilot

- **Multi-Vehicle** type support
  - **Multicopter**
  - **Fixed Wing**
  - **VTOL**
  - **Rover**
  - **Under Surface**
  - **Above Surface**
  - **Balloons**
  - **Satellites**
  - **Jetpack**

- **Developer Friendly Interfaces**
  - **Flight Modes** provide a set of helpers to control autonomy
  - **Flight Tasks** allowing developers to extend flight modes
  - **Parameter** database exposing functionality back to users
  - **Events interface** giving developers a system-wide API for notifications
- **Control allocation** translates thrust and torque commands into actuator commands which control motors and servos
  - Controllers do not require special handling for airframe geometry
- Native **ROS 2 Support** through DDS

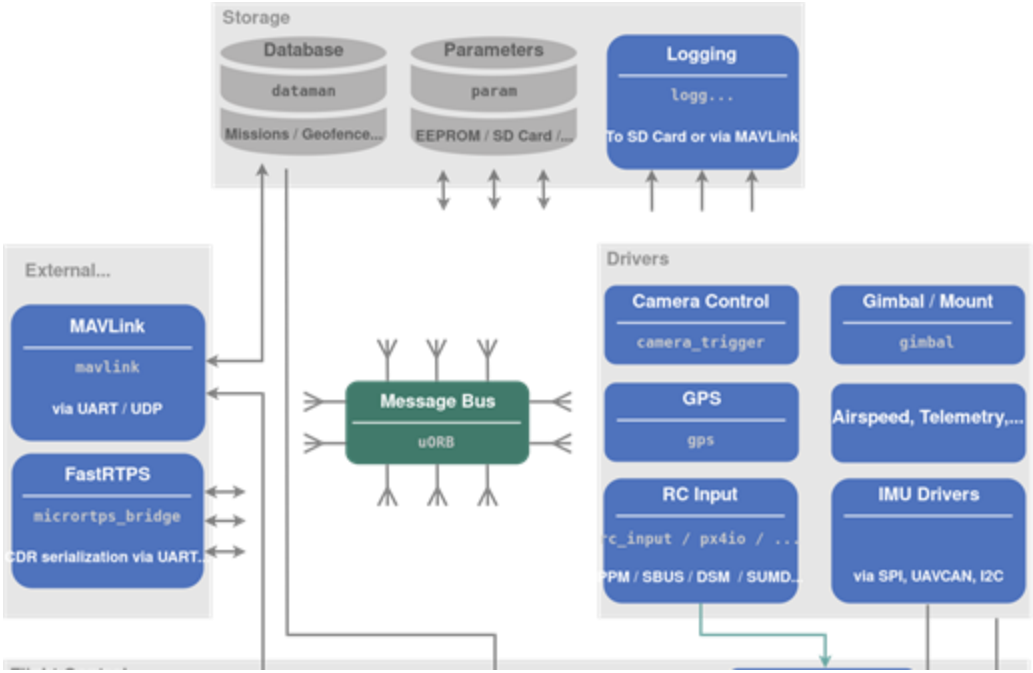
PX4's Stack Architecture

- **Developer Friendly Interfaces**
  - **Flight Modes** provide a set of helpers to control autonomy
  - **Flight Tasks** allowing developers to extend flight modes
  - **Parameter** database exposing functionality back to users
  - **Events interface** giving developers a system-wide API for notifications
- **Control allocation** translates thrust and torque commands into actuator commands which control motors and servos
  - Controllers do not require special handling for airframe geometry
- Native **ROS 2 Support** through DDS



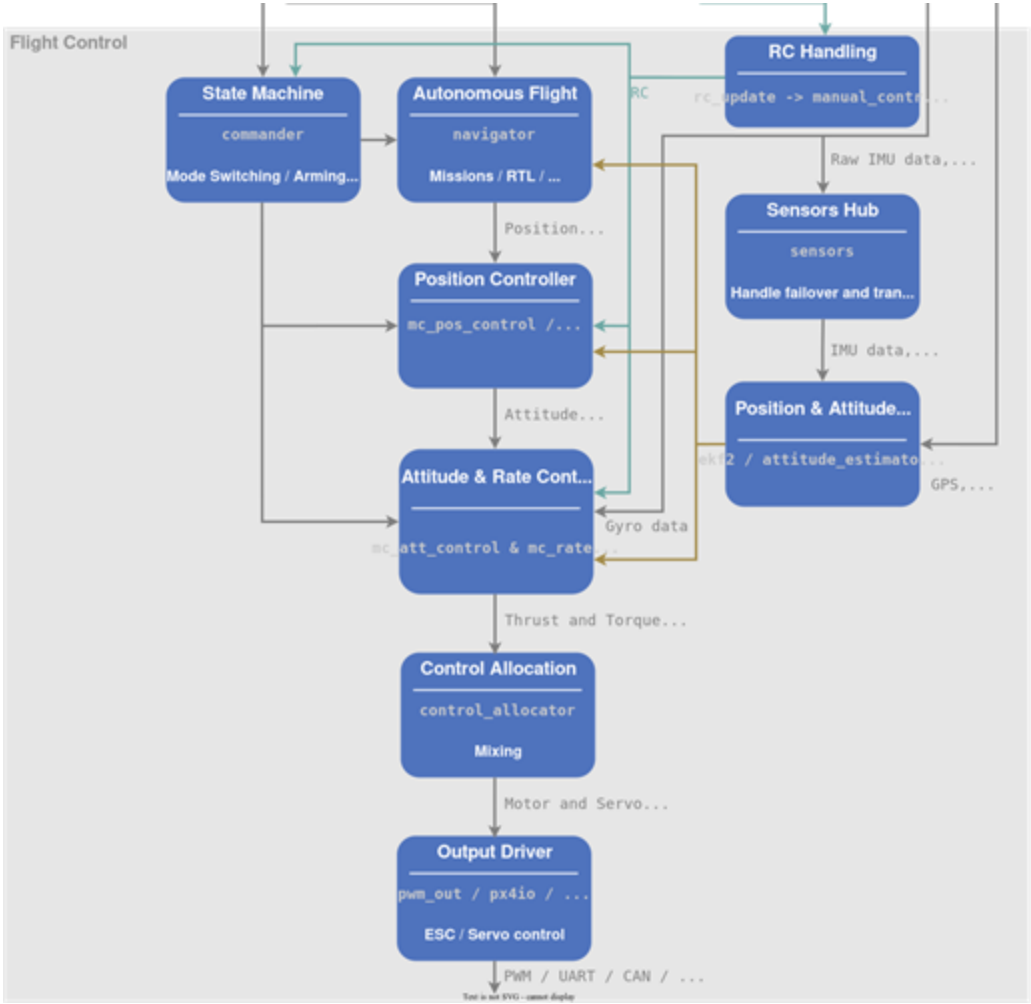
PX4's Stack Architecture

- System Storage:
  - Database
  - Parameters
  - Logging
- Drivers:
  - GPS
  - Airspeed
  - IMU
  - Baro
  - etc...
- External Interfaces:
  - MAVLink
  - XRCE-DDS for ROS2



PX4's Stack Architecture

- State Machine
- Sensor Hub
- Autonomous Flight (Navigation)
  - Position Controller
  - Attitude & Rate Controller
  - Control Allocation
- RC Handling



Parameter based configuration

The burden of such a generic platform falls upon our Parameter system. Parameters are stored in the filesystem. We have two types of parameters: Runtime and Boot time.

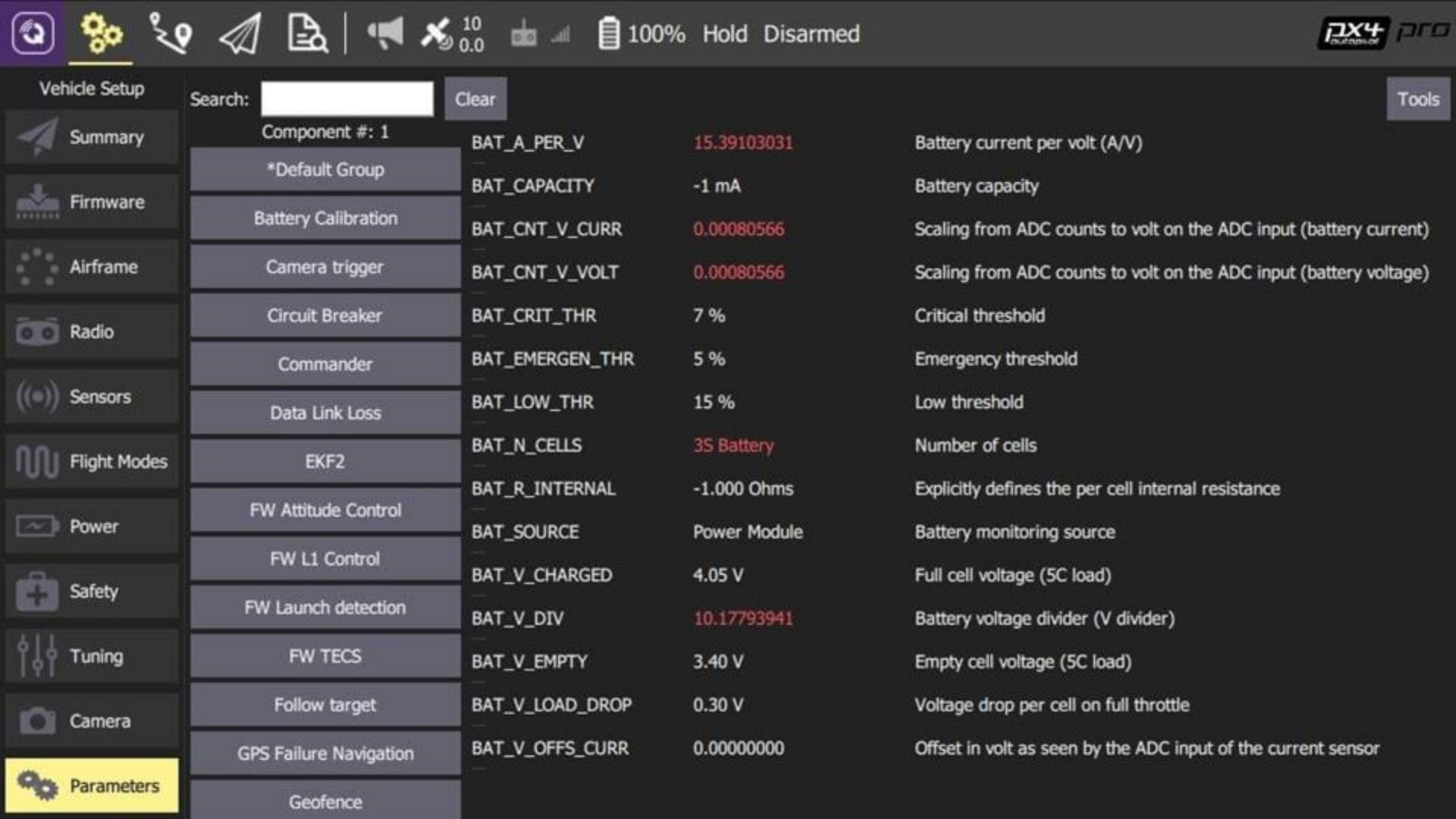
Main Concepts

- Data types: **float** & **int32_t**
- You can export / import parameters
- YAML metadata & C/C++ API

Everything can be configurable approach

What works for us?

- We give developers the tools to allow their end users to configure aspects of their system
- Think of how to a pilot would define the takeoff altitude
- Really powerful when paired with a ground station
- Thanks to metadata we can build cool online reference lists


10
0.0
100% Hold Disarmed
Vehicle Setup
Summary
Firmware
Airframe
Radio
Sensors
Flight Modes
Power
Safety
Tuning
Camera
Parameters
Search:
Clear
Tools
Component #: 1
*Default Group
Battery Calibration
Camera trigger
Circuit Breaker
Commander
Data Link Loss
EKF2
FW Attitude Control
FW L1 Control
FW Launch detection
FW TECS
Follow target
GPS Failure Navigation
Geofence
BAT_A_PER_V 15.39103031 Battery current per volt (A/V)
BAT_CAPACITY -1 mA Battery capacity
BAT_CNT_V_CURR 0.00080566 Scaling from ADC counts to volt on the ADC input (battery current)
BAT_CNT_V_VOLT 0.00080566 Scaling from ADC counts to volt on the ADC input (battery voltage)
BAT_CRIT_THR 7 % Critical threshold
BAT_EMERGEN_THR 5 % Emergency threshold
BAT_LOW_THR 15 % Low threshold
BAT_N_CELLS 3S Battery Number of cells
BAT_R_INTERNAL -1.000 Ohms Explicitly defines the per cell internal resistance
BAT_SOURCE Power Module Battery monitoring source
BAT_V_CHARGED 4.05 V Full cell voltage (5C load)
BAT_V_DIV 10.17793941 Battery voltage divider (V divider)
BAT_V_EMPTY 3.40 V Empty cell voltage (5C load)
BAT_V_LOAD_DROP 0.30 V Voltage drop per cell on full throttle
BAT_V_OFFS_CURR 0.00000000 Offset in volt as seen by the ADC input of the current sensor

Connecting apps! - uORB Middleware

u(Micro) Object Request Broker

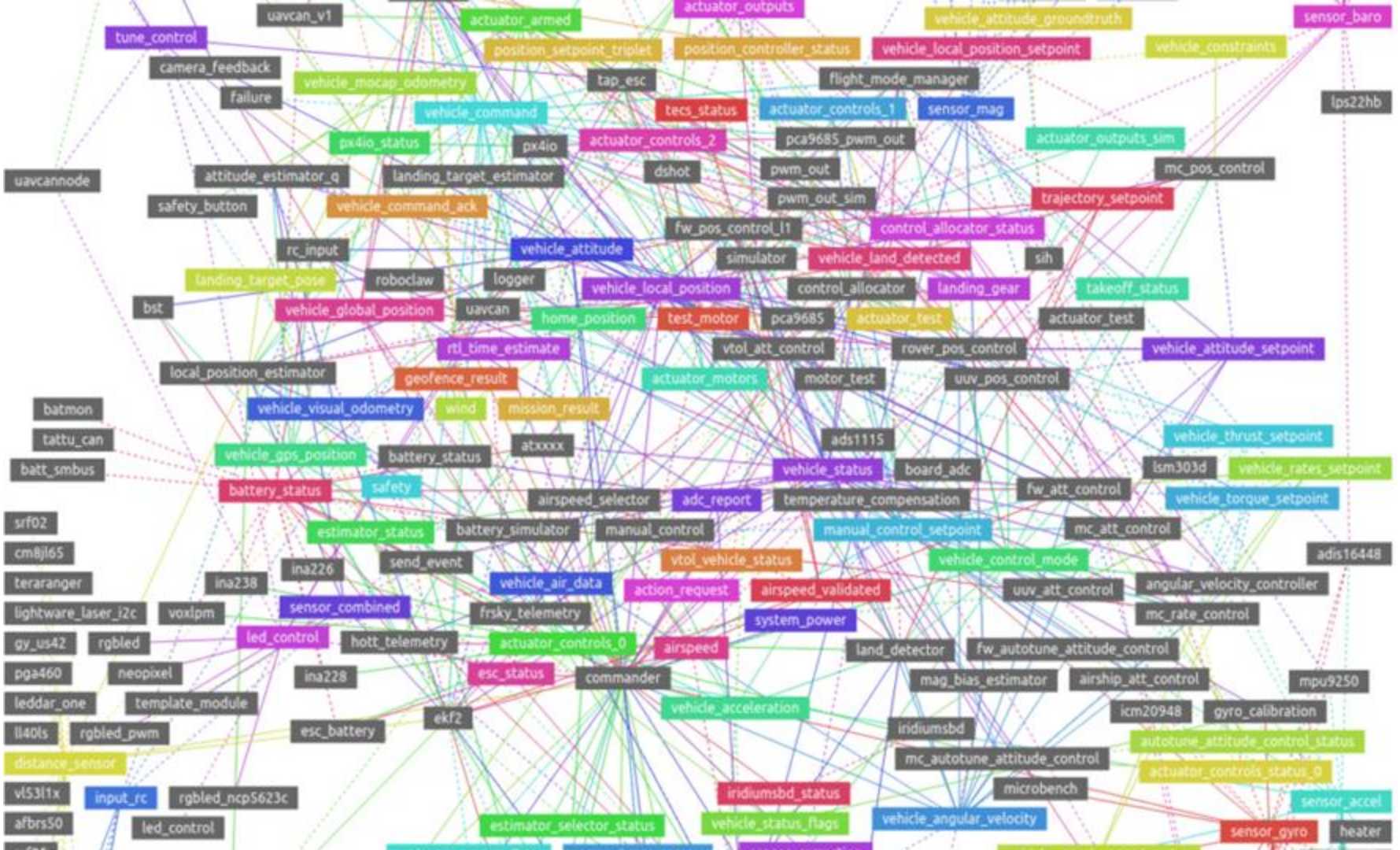
It's a really simple middleware, and arguably the top feature of PX4, at least in my opinion. It's one of the first applications that launch on boot since most of PX4 depends on it.

Main Concepts

- Messages - field information, typically reused by multiple topics
- Topics - what you subscribe and publish
- Modules - apps that depend on pub/sub of topics

What we like about uORB?

- Fast!
- DDS-like... compatible
- Offers a comprehensive way to build interfaces between modules/apps
- Priorities, rates, queues
- Messages are defined in **.msg** and generated at build time (**.h** / **.cpp**)
- NuttX Shell (NSH) Tooling
  - `ls /obj`
  - `listener <message_name> <num_messages>`
  - `uorb top`

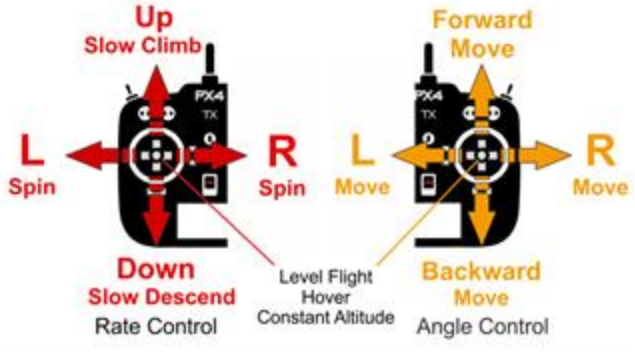

uavcan_v1
actuator_armed
actuator_outputs
vehicle_attitude_groundtruth
sensor_baro
tune_control
position_setpoint_triplet
position_controller_status
vehicle_local_position_setpoint
vehicle_constraints
camera_feedback
vehicle_mocap_odometry
tap_esc
flight_mode_manager
failure
vehicle_command
tecs_status
actuator_controls_1
sensor_mag
lps22hb
px4io_status
px4io
actuator_controls_2
pca9685_pwm_out
actuator_outputs_sim
uavcannode
attitude_estimator_q
landing_target_estimator
dshot
pwm_out
mc_pos_control
safety_button
vehicle_command_ack
pwm_out_sim
trajectory_setpoint
fw_pos_control_l1
control_allocator_status
rc_input
vehicle_attitude
simulator
vehicle_land_detected
sih
landing_target_pose
roboclaw
logger
vehicle_local_position
control_allocator
landing_gear
takeoff_status
bst
vehicle_global_position
uavcan
home_position
test_motor
pca9685
actuator_test
actuator_test
rtl_time_estimate
vtol_att_control
rover_pos_control
vehicle_attitude_setpoint
local_position_estimator
geofence_result
actuator_motors
motor_test
uuv_pos_control
batmon
vehicle_visual_odometry
wind
mission_result
tattu_can
ads1115
vehicle_thrust_setpoint
vehicle_gps_position
battery_status
atxxxx
batt_smbus
vehicle_status
board_adc
lsm303d
vehicle_rates_setpoint
battery_status
safety
airspeed_selector
adc_report
temperature_compensation
fw_att_control
vehicle_torque_setpoint
srf02
estimator_status
battery_simulator
manual_control
manual_control_setpoint
mc_att_control
cm8jl65
send_event
vtol_vehicle_status
vehicle_control_mode
adis16448
teraranger
ina226
ina238
vehicle_air_data
action_request
airspeed_validated
uuv_att_control
angular_velocity_controller
lightware_laser_i2c
voxlpm
sensor_combined
frsky_telemetry
system_power
mc_rate_control
gy_us42
rgbled
led_control
hott_telemetry
actuator_controls_0
airspeed
land_detector
fw_autotune_attitude_control
pga460
neopixel
ina228
esc_status
commander
mag_bias_estimator
airship_att_control
mpu9250
leddar_one
template_module
vehicle_acceleration
icm20948
gyro_calibration
ll40ls
rgbled_pwm
ekf2
esc_battery
iridiumsbd
autotune_attitude_control_status
distance_sensor
mc_autotune_attitude_control
actuator_controls_status_0
vl53l1x
input_rc
rgbled_ncp5623c
iridiumsbd_status
microbench
sensor_accel
afbrs50
led_control
estimator_selector_status
vehicle_status_flags
vehicle_angular_velocity
sensor_gyro
heater

PX4's *Internal* Flight Modes

Modes are Vehicle Specific. Examples:

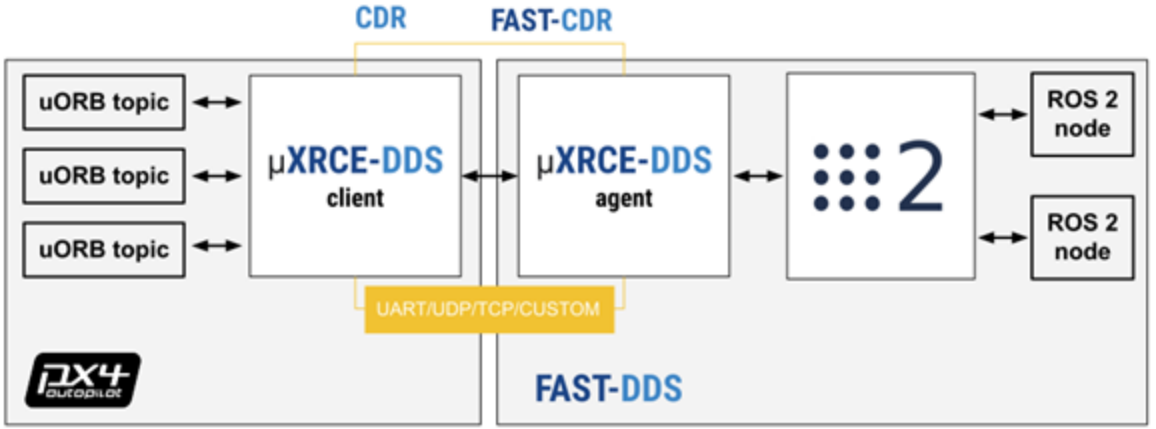
- **Position**: When the sticks are released the vehicle will remain locked to a position in 3D space.
- **Altitude**: When the sticks are released the vehicle will maintain current altitude.
- **Hold**: Vehicle stops and hovers at its current GPS position.
- **Takeoff**: Vehicle takes off to a specified height and waits further input.
- **Land**: Lands vehicle at the position where the mode was engaged.
- **Mission**: Starts a mission that was preloaded to the drone

- Modes can be Manual or Autonomous
- The API to create modes is "**Flight Task**"



ROS 2 Support

- Thanks to uORB middleware we can communicate directly with the ROS 2 middleware (XRCE-DDS based)
- PX4 internal modules can share data with ROS 2 nodes
- Agent / Client approach
- Ethernet and Serial support
- ROS 2 QoS Supported
- We are ready for the switch to Zenoh!



External Flight Modes

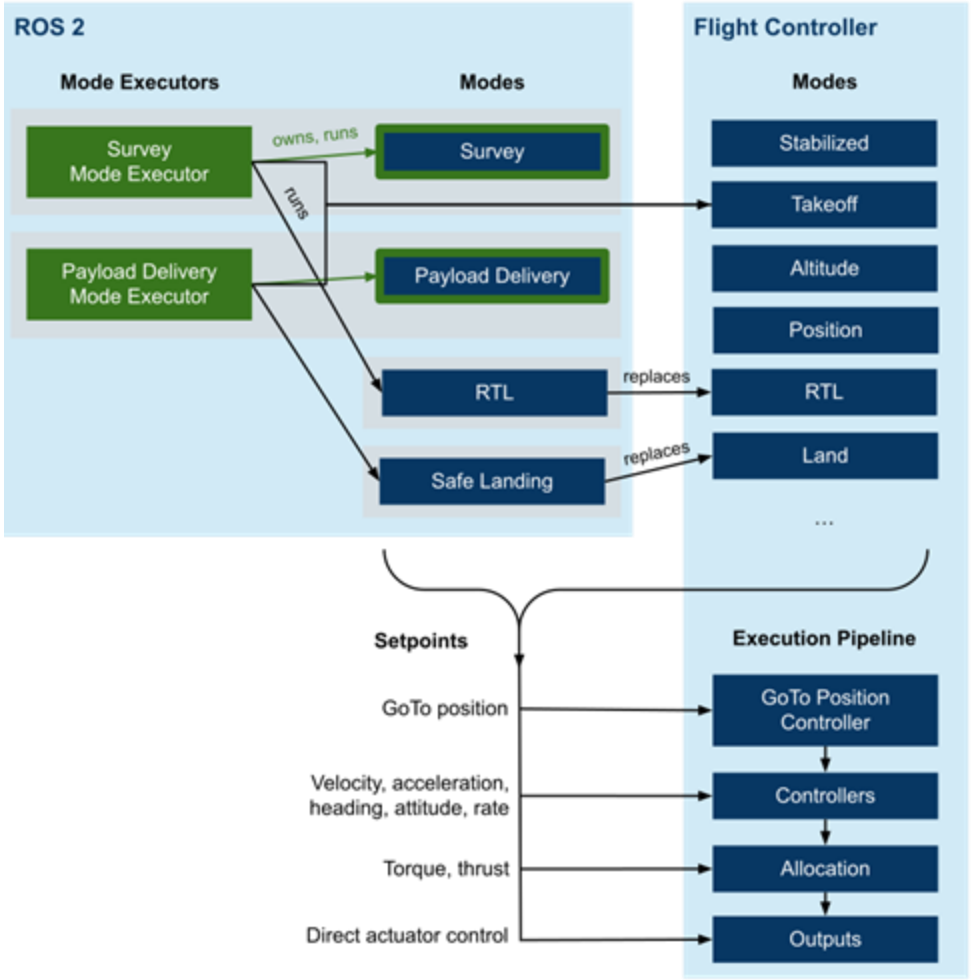
External Flight Modes are Flight Modes that can be developed and executed from outside the Flight Controller and in a non-real time system.

When do External Flight Modes matter?

- When you don't need low-level access for direct motor control
- When you have a compute module where you can run ROS 2
- When the mode is not Safety Critical

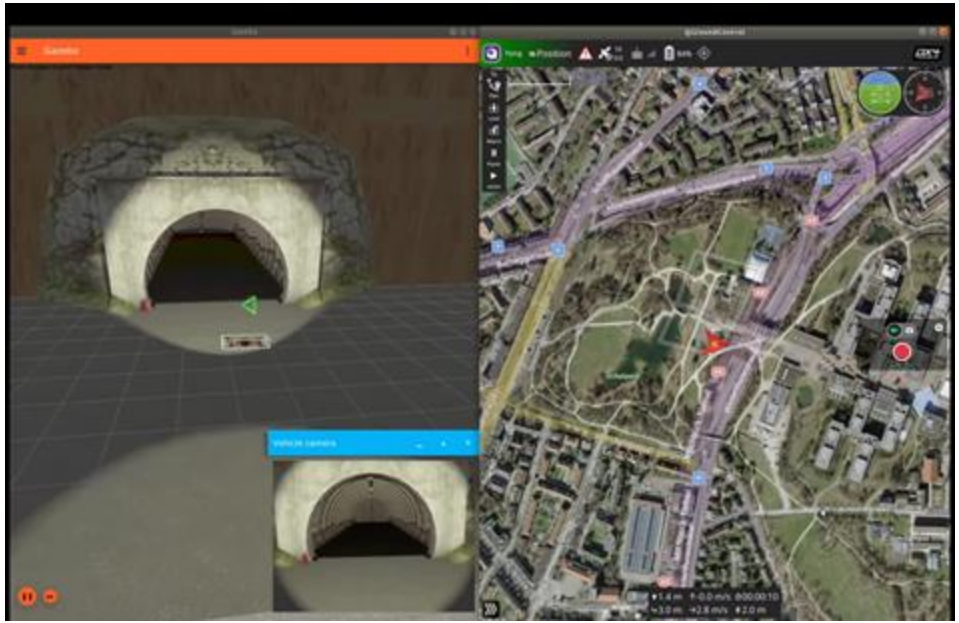
Advantages

- You don't need to deal with low level embedded constraints and requirements
- Crashing the app won't result in a catastrophic failure
- Easier to maintain outside of the PX4 codebase
- Take advantage of high-level functionality like dynamic data structures, and the ROS 2 library ecosystem.
- More advanced compute for greater processing



PX4 Simulation

Gazebo is our default simulation agent, we support both classic and modern gazebo, with multiple worlds and models to choose from.



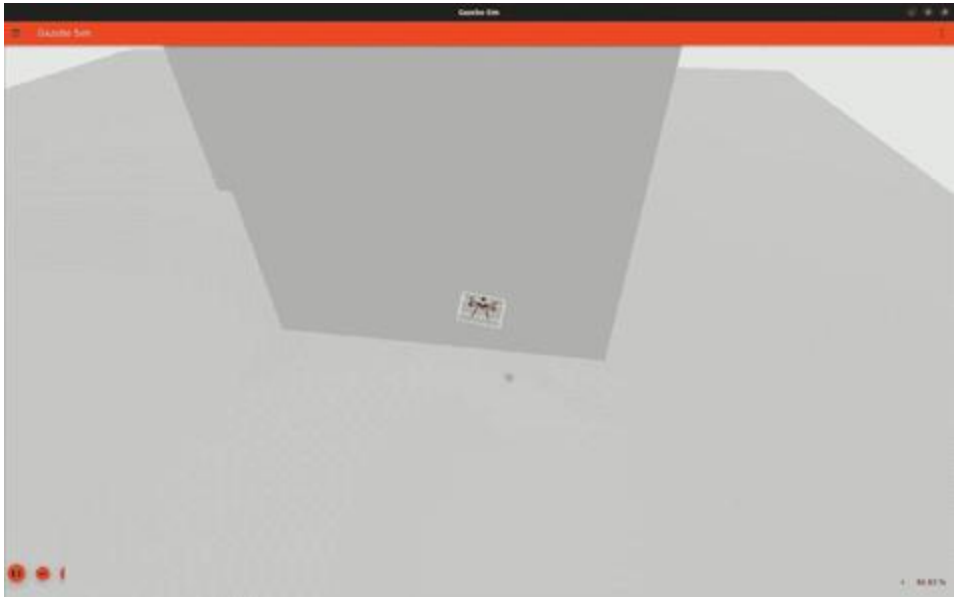
Gazebo Simulation Docs

Other Simulation Engines Supported

- Gazebo Classic
- AirSim
- Flight Gear
- jMAVSim

PX4 Simulation

Gazebo is our default simulation agent, we support both classic and modern gazebo, with multiple worlds and models to choose from.



Other Simulation Engines Supported

- Gazebo Classic
- AirSim
- Flight Gear
- jMAVSim

Gazebo Simulation Docs

Supported Hardware Architectures

Cortex-M4, Cortex-M7 &
Cortex-M7 dual-core

32 & 64 bit

Supported RTOS

NuttX RTOS

Zephyr®

(*Coming Soon)

Hardware Support

- Support for more than 80 boards from 30+ manufacturers
- Drivers for more than 100+ sensors
  - IMU, Baro, Actuators, GPS, INS, CAN, UWB... etc.
- Main Architectures Supported
  - STM32 - STMicro
  - Snapdragon - Qualcomm
  - iMX - NXP Semiconductors
  - RISCV-V

Pixhawk Project

pixhawk®

Open Hardware & Open Standards

Started as a flight controller open hardware project with the first versions of Pixhawk's

Evolved into an Open Standard for

- Flight controllers
- Payloads (Gimbals)
- Smart Batteries
- Connectors
- Debuggers

16 years of hardware

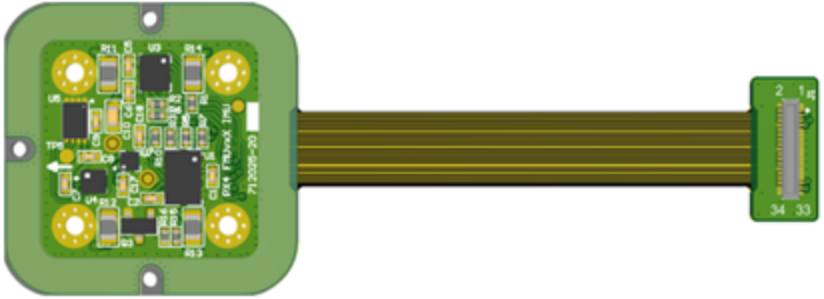
Latest Pixhawk Hardware

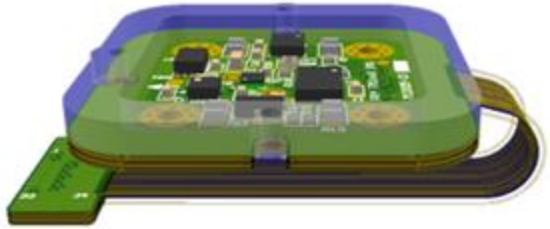
Holybro Pixhawk 6X-RT
$280

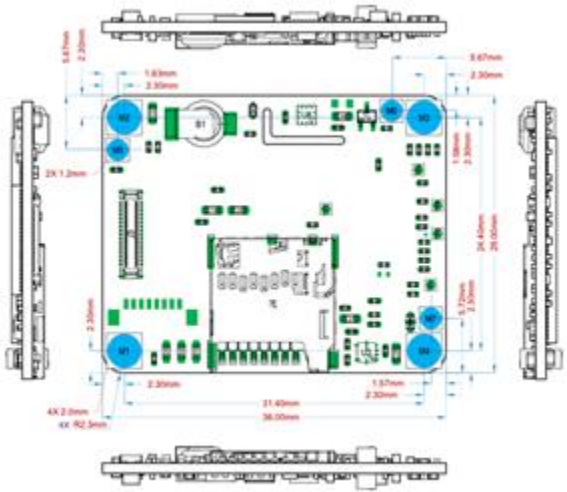
NXP MR-VMU-RT1176
$300

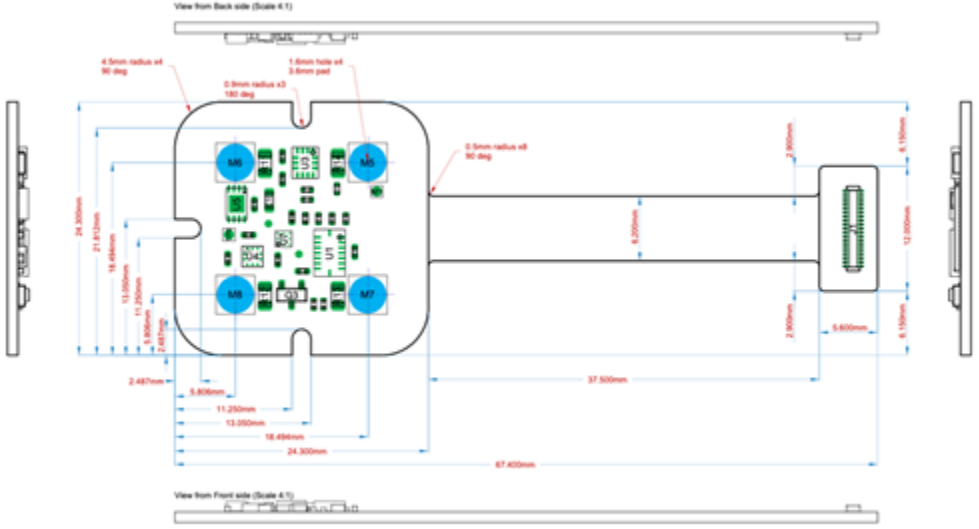
Open Standards

Pixhawk Flight Controller FMUv6X

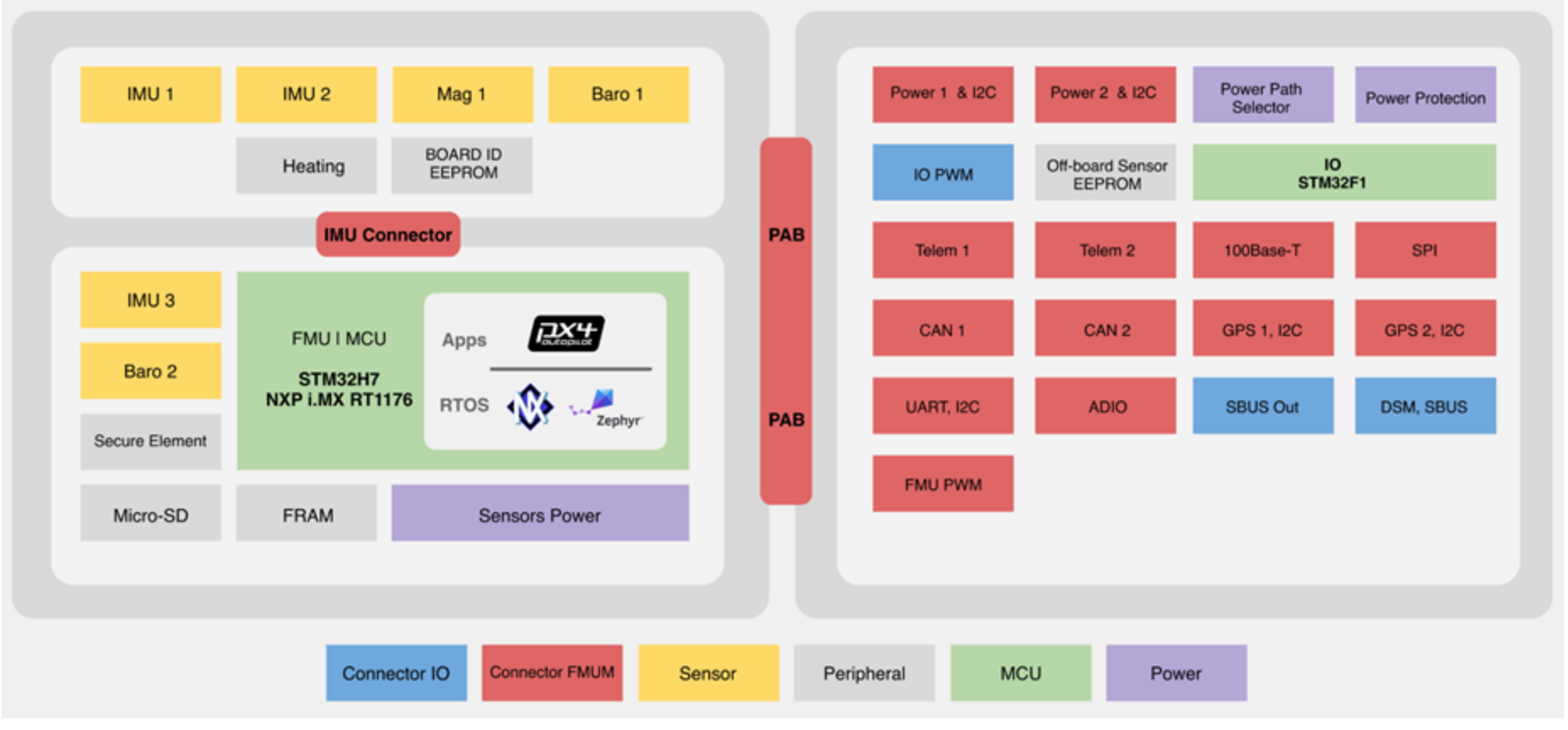








Pixhawk Flight Controller FMUv6X



Advance Compute Hardware

ARK Electronics
$1,100.00

ARK V6X
NVidia Jetson Family

ModalAI
$1,249.99

Qualcomm RB5

Holybro
$1,940

Pixhawk 6X
NVidia Jetson Family

NXP
$299.00

NavQPlus i.MX 8M Plus

Official PX4 Developer Kits

DroneBlocks
$1,999.00

ModalAI
$2,749.99

Holybro
$507.99

MAVLink

Lightweight messaging protocol that was originally intended to bridge PX4 to a Ground Station, and ended up becoming the de-facto standard to define sub-protocols of UX and UI everywhere else. It's our most used project and possibly the most underrated.

Main Concepts

- Supports unreliable low-bandwidth links
- Messages are the fundamental basis of the protocol
- XML-based messages
- Standardized messages in four groups: development, common, standard, minimal

Main Concepts (Continued)

- Microservices are high level protocols built on top of MAVLink
  - Anything that can't be defined/explained by a single message

What works for us?

- You can create custom messages
- More standards the better!
- Multiple SDK and Ground Stations

MAVLink

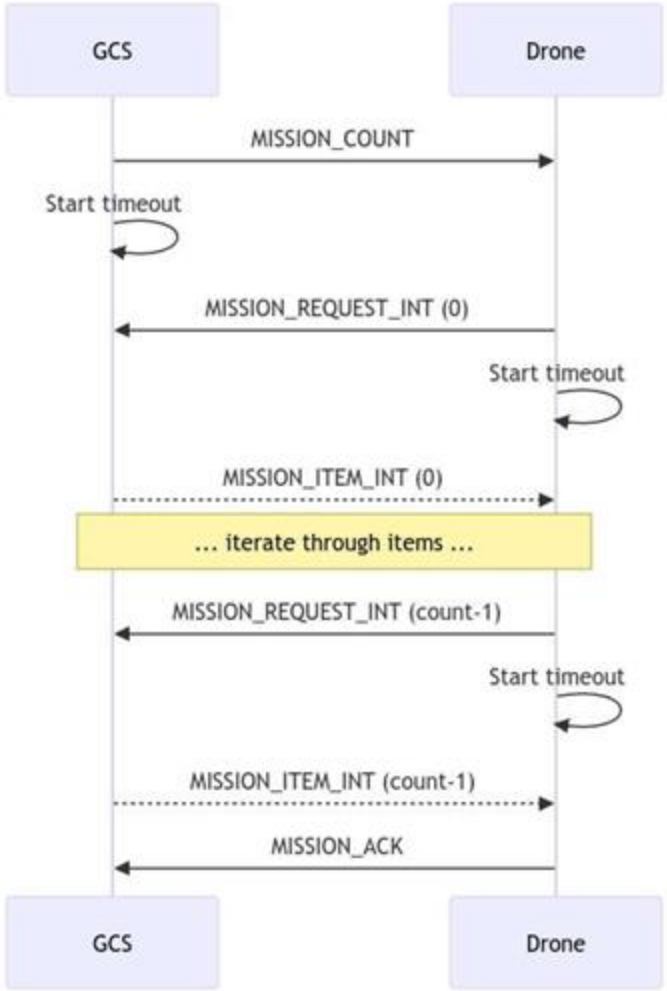
Basics of MAVLink

- Messages defined in XML
- Three main message sets
  - **Common**: Messages that have been included in at least one flight stack, including everything in Standard
  - **Standard**: The standard set of entities that are implemented by at least two core flight stacks in a compatible way.
  - **Minimal**: the minimum set of entities required to setup a MAVLink network.
- There's one more.
  - **Development**: This is where proposed messages live initially, it's an effort to allow flight stacks a way to implement messages before making them part of **Common** or **Standard**.
- The project has parsers, and tests available.

MAVLink

Microservices

- Heartbeat/Connection
- Mission Protocol
- Parameter Protocol
- Camera Protocol
- Gimbal Protocol
- Battery Protocol
- Traffic Management
- Standard Flight Modes
- Time Synchronization
- Open Drone ID
- etc..



MAVSDK

- SDK for MAVLink
- Because you writing XML is not meant for humans.
- Available for the most widely used programming languages:
  - C++, Swift, Python, Java, Go, JS, CSharp, Rust
- Server / Client architecture
  - Both can be embedded in the same library
  - Server / Client communication is through gRPC
- Widely used and adopted by industry

Top Features I like

- Plugin Architecture
- Events Interface
- Error Handling
- Shared Vehicle Control
- Multiple resources available: example code and tutorials

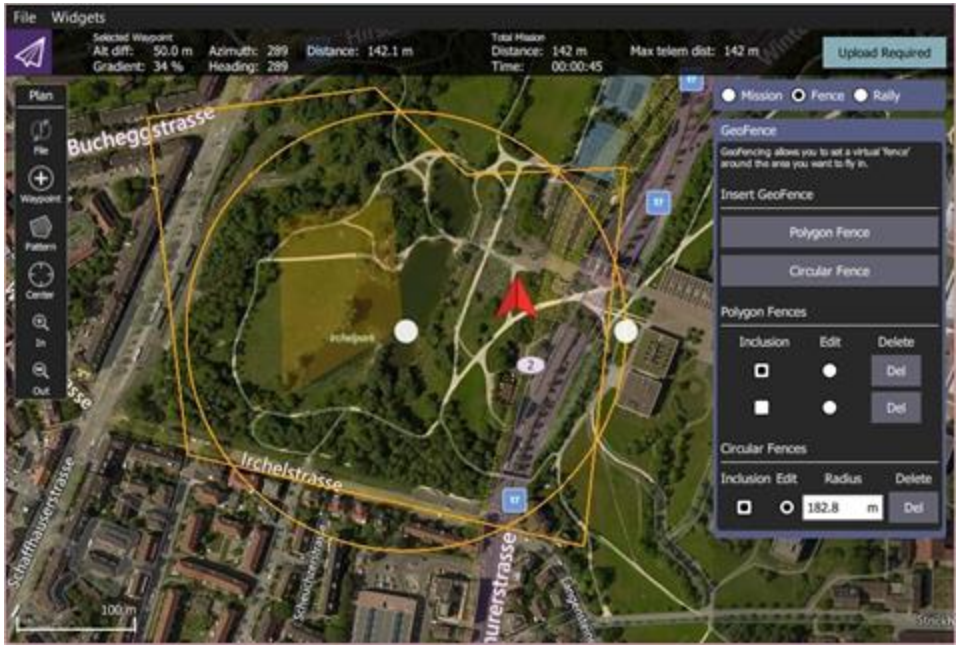
QGroundControl

Ground Station that features a wide range of functionality from all aspects of vehicle control for setup and tuning, to mission based control. All of this thanks to MAVLink

Can be deployed to multiple platforms: Linux, Windows, macOS, Android and iOS

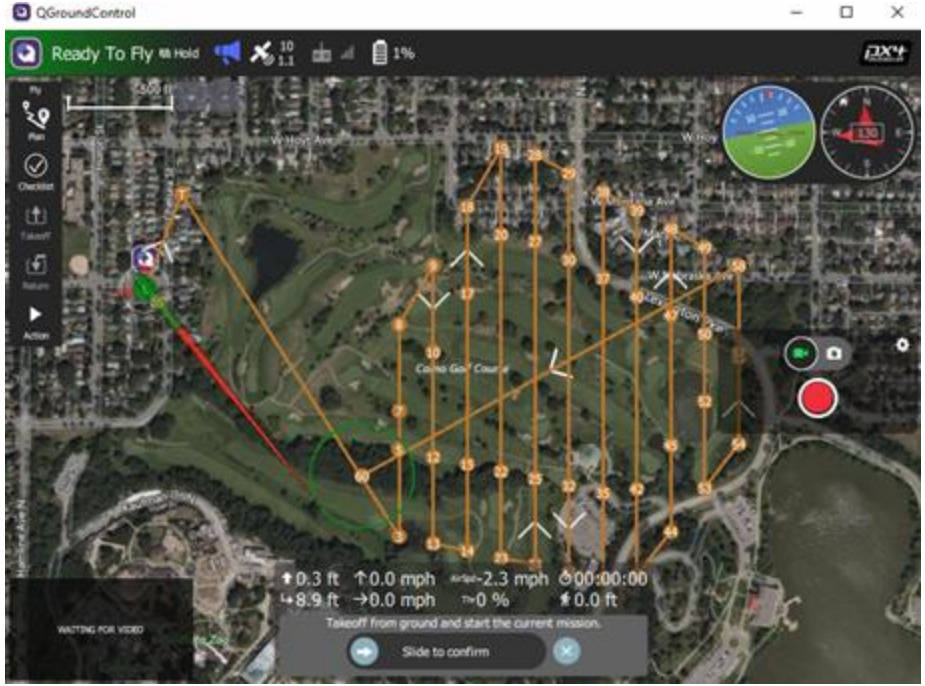
Vendors usually customize QGC for their end-users, allowing them to provide a fully featured Mission Control app.

Can be themed, you can remove any unwanted functionality.



QGroundControl

- **Open Source** Mission Planner
- Fully **themable** interface
- Support for all **MAVLink compliant** flight stacks
- **Vendor Neutral** governance
- Permissively licensed: **Apache 2.0**
- **Qt-based** application
- **Cross-platform**: Mobile, Tablets, Linux, Windows, and macOS
- Can control every aspect of a drone, **from firmware setup, to advanced mission planning**



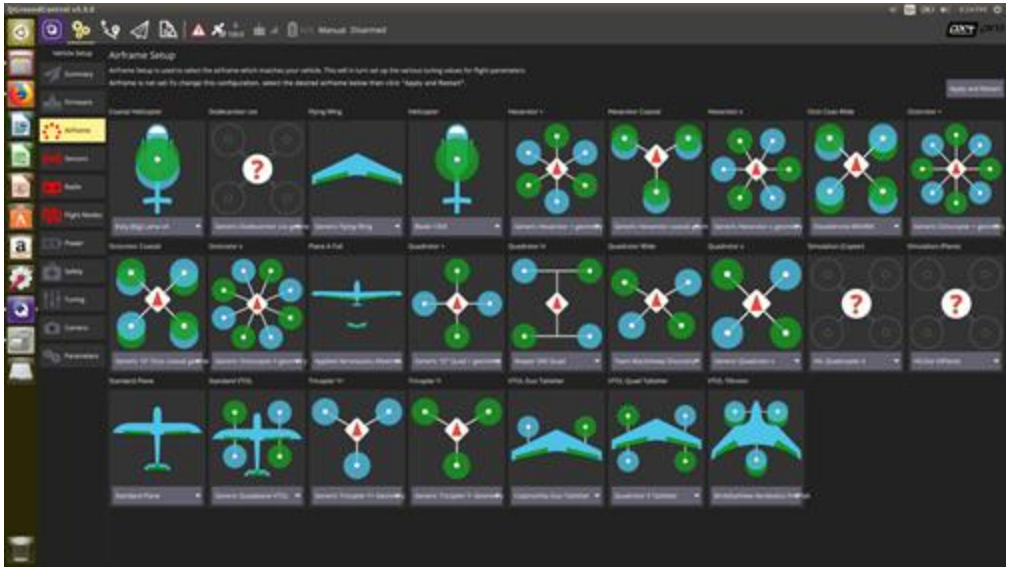
QGroundControl v. 0.9.0 (Alpha RC1) (vpn-global-
Unmanned Systems
MAV 001
AUTO MODE
WAYPOINT
ACTIVE
WP0
Active, normal operation.
Horizontal Situation Indicator
Flight Display
000.0
010.0
-004.0
Lat/Lon
Home
Edit
Set Home
Trails
Follow
Map View
Tear Trail
Reset
Sky
Day/Night
Mission Plan
NAV Waypoint
Save WPs
Load WPs
Read
Write
New current waypoint 0

3.5
2359.1

QGroundControl

Top Features I like

- Full setup and control of PX4 vehicles. (Also supports other MAVLink based platforms)
- Mission Planning for Autonomous Flight
- Flight Map with all telemetry, waypoints and vehicle instruments.
- 3D Viewer
- Video Streaming
- Support for managing multiple Vehicles
- Multi-Platform



Dronecode FOUNDATION

Dronecode Stack

Dronecode Stack

The Latest Releases from Dronecode

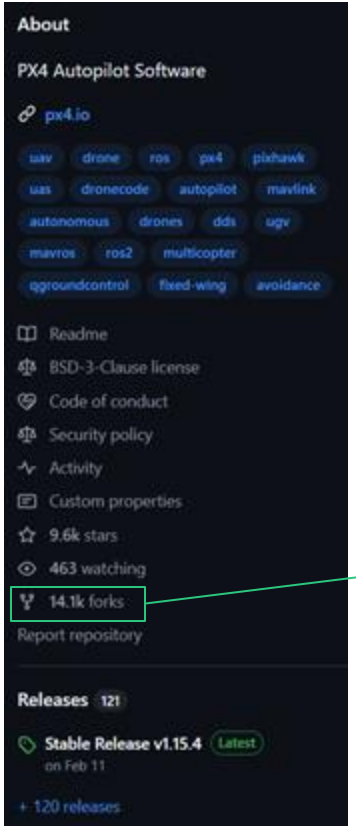
PX4 v1.16 Release Candidate- New Features

- **Expanded Hardware Support**
- **Major Rover Support Rework**
- **Enhanced State Estimation**
- **Gazebo Harmonic Sim Upgrade**
- **Improved Compass Calibration**
- **Advanced ROS 2 Integration**
- **New Sensor Support**

and more, see Release notes v1.16.

V1.16 Coming Soon!

Dronecode FOUNDATION



Widely-Used and Customized

14.1K Forks

Trusted by developers and organizations globally

Pixhawk 6 Open Hardware Standards

Pixhawk Autopilot v6X Standard

Pixhawk Autopilot v6C Standard

Pixhawk Autopilot v6U Standard

Pixhawk Autopilot Bus Standard

Pixhawk Connector Standard

Pixhawk Autopilot v5X Standard

and more, see Pixhawk Hardware Standards.

MAVSDK V3.6

Automated Cleanup for Docker Images

Core UDP Remote Connection Fixes

Third-Party Build Optimization

and more, see Full Changelog v3.6.0

Read MAVSDK documentation

Read MAVLink documentation

QGroundControl V5.0

Dronecode FOUNDATION

🎨 UI and Workflow Updates

⚙️ Streamlined Configuration and App Settings

🚁 Improved Multi-Vehicle and Telemetry Support

✈️ Configurable Flight Modes List

QGroundControl

♿ Accessibility Updates

🛠️ Modernized Build System & Developer Infrastructure

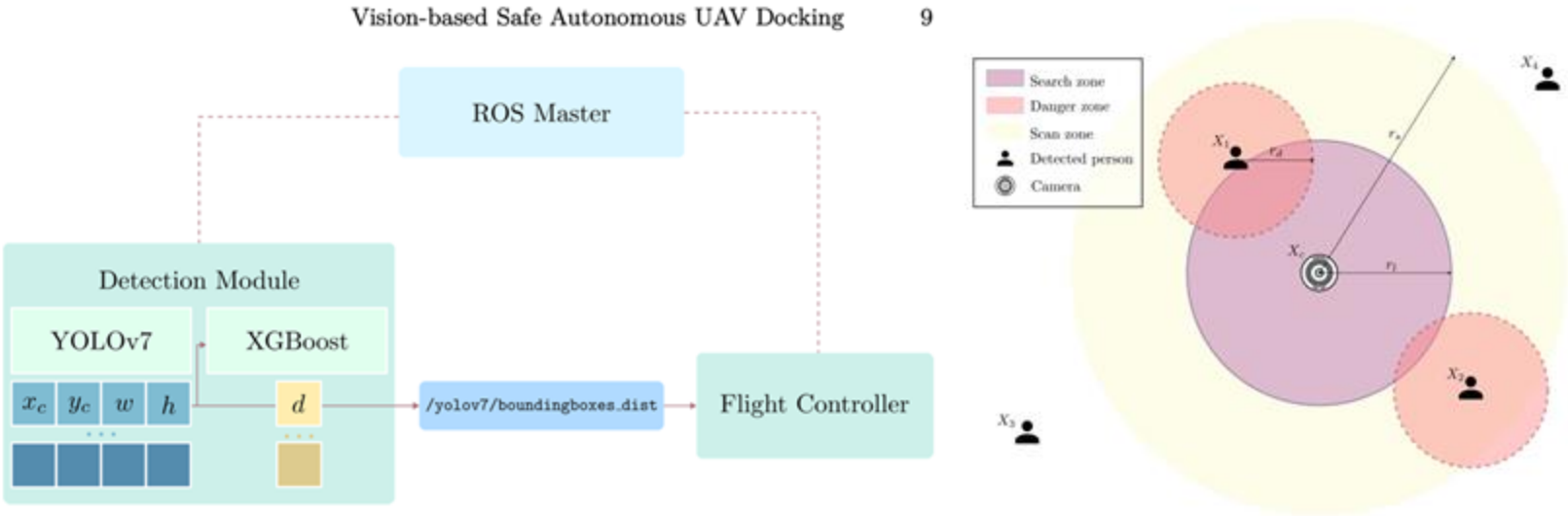
and more, see Full Changelog v5.0

Read QGC documentation

Micro Air Vehicle Message Marshalling Library.

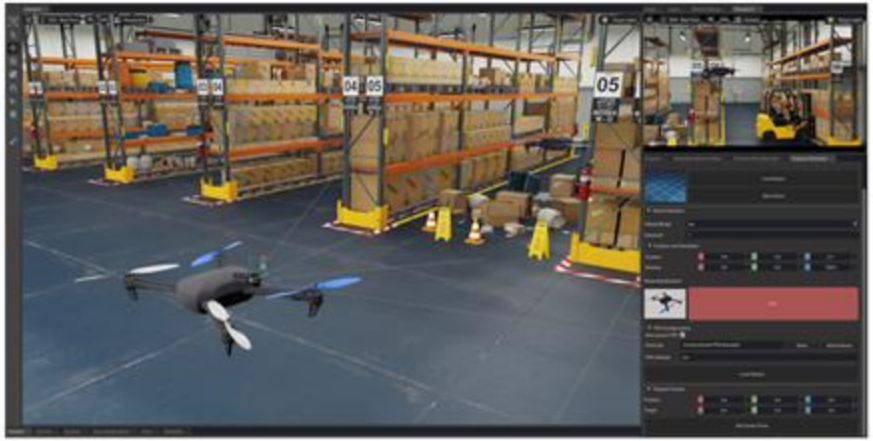
Research Projects Powered by Dronecode

Vision-based Safe Autonomous UAV Docking with Panoramic Sensors



[1]Nguyen Thuan, P., Peña Queralta, J., and Westerlund, T., “Vision-based Safe Autonomous UAV Docking with Panoramic Sensors”, arXiv e-prints, 2023. doi:10.48550/arXiv.2305.16008.

Pegasus Simulator: An Isaac Sim Framework for Multiple Aerial Vehicles



[1]Jacinto, M., “Pegasus Simulator: An Isaac Sim Framework for Multiple Aerial Vehicles Simulation”, arXiv e-prints, 2023. doi:10.48550/arXiv.2307.05263.

End-to-End UAV Simulation for Visual SLAM and Navigation

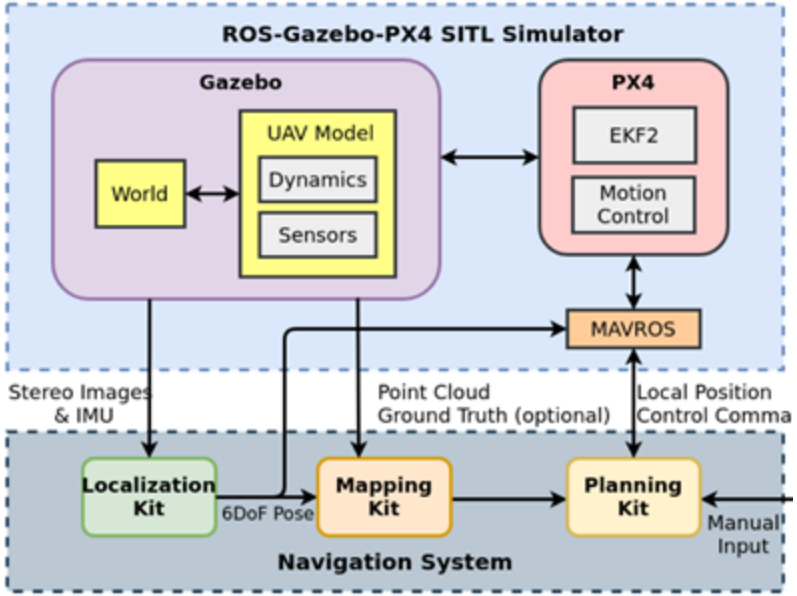
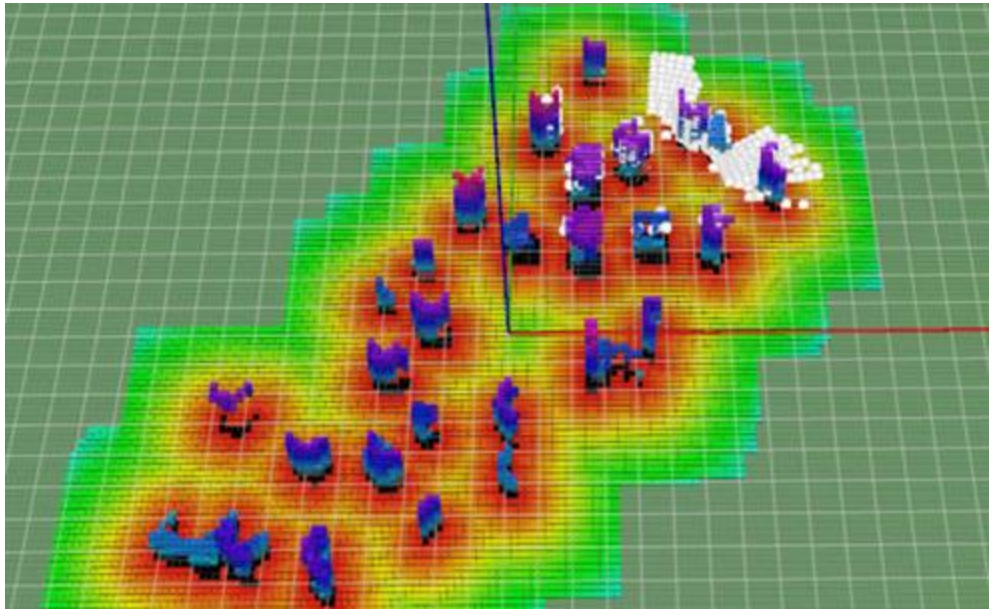
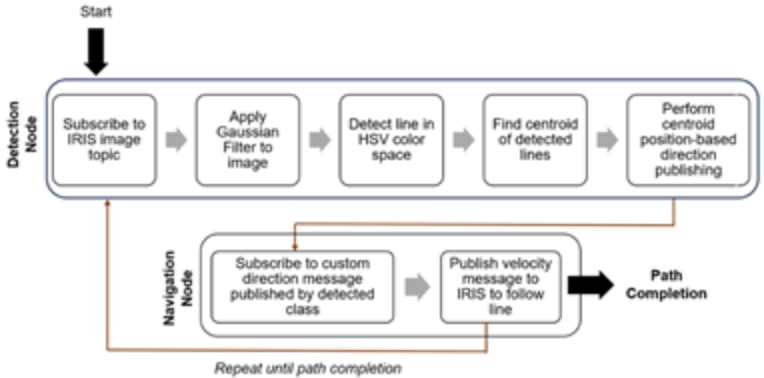


Fig. 2: Framework of simulation.



[1]Chen, S., Chen, H., Zhou, W., Wen, C.-Y., and Li, B., “End-to-End UAV Simulation for Visual SLAM and Navigation”, arXiv e-prints, 2020. doi:10.48550/arXiv.2012.00298.

Autonomous Navigation of Micro Air Vehicles in Warehouse Using Vision-based Line Following



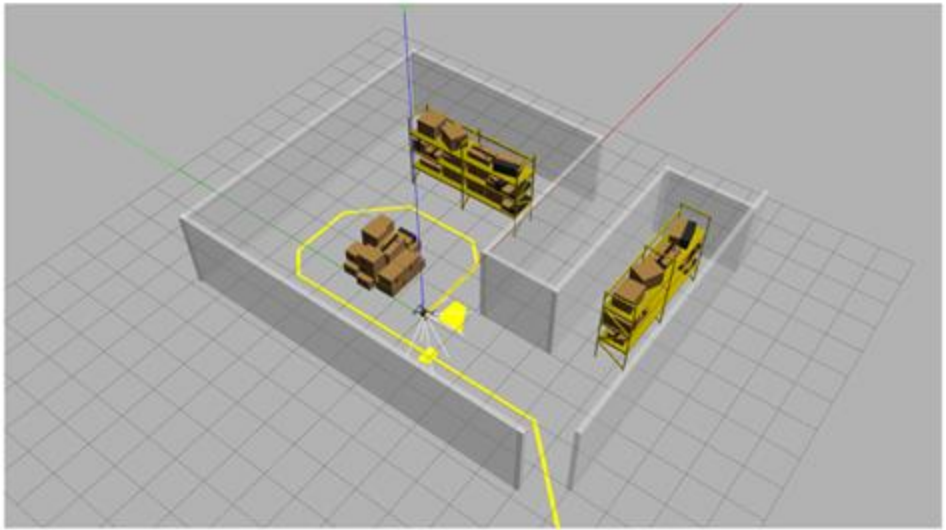


Fig. 7. Simulation Environment 1: A warehouse environment with a straight-lines path.

[1]Jacinto, M., “Pegasus Simulator: An Isaac Sim Framework for Multiple Aerial Vehicles Simulation”, arXiv e-prints, 2023. doi:10.48550/arXiv.2307.05263.

DISCOWER
Holistic approach to control and planning for space and subsea autonomous systems
KTH VETENSKAP OCH KONST
WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM
OHB SWEDEN
SAAB
NASA
FMV
WARA-PS

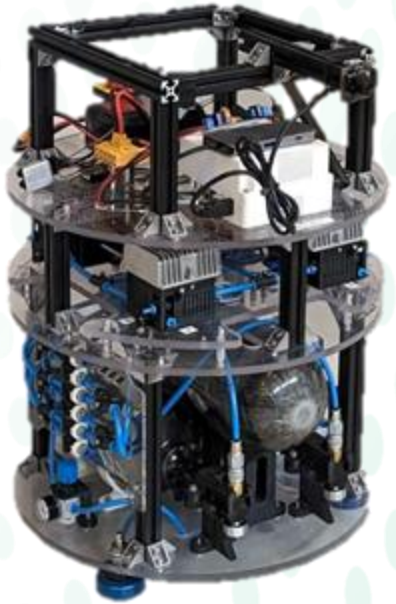
Spacecraft in PX4-Autopilot: OSS for Space Robotics

Available Functionality:

1. Control modes: Position, Stabilized, Acro, Wrench, Direct-Allocation, Offboard
2. Platform: KTH's ATMOS, atmos.discower.io OSH with BOM, build guide, community
3. Actuation: 8 Thrusters (1.4N each)

In the Future:

1. Actuation: 4 Motors with Propellers (disturbance injection, ~2N each), runs in parallel)
2. Control Modes: add velocity-based control
3. More labs/platforms using ATMOS and/or PX4: atmos.discower.io/others/atmos_in_the_wild/



ATMOS free-flyer. Available in PX4-Autopilot SITL.

Contact

Ramon Roche | General Manager
rroche@linuxfoundation.org

www.dronecode.org

The Open Source UAV Ecosystem