PX4 Developer Summit 2019

Presented by **Dr. Lorenz Meier**

WELCOME!

Planned: 40 - Effective: 200 Attendees

Quick introduction: Dr. Lorenz Meier

2008 pixhawk

2009 MAVLINK
MICRO AIR VEHICLE COMMUNICATION PROTOCOL

2009 QGroundControl

2011 px4 autopilot

lorenz@px4.io

LorenzMeier

LorenzMeier

ETH zürich

PX4 Developer Summit 2019

#px4summit

“Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live.”

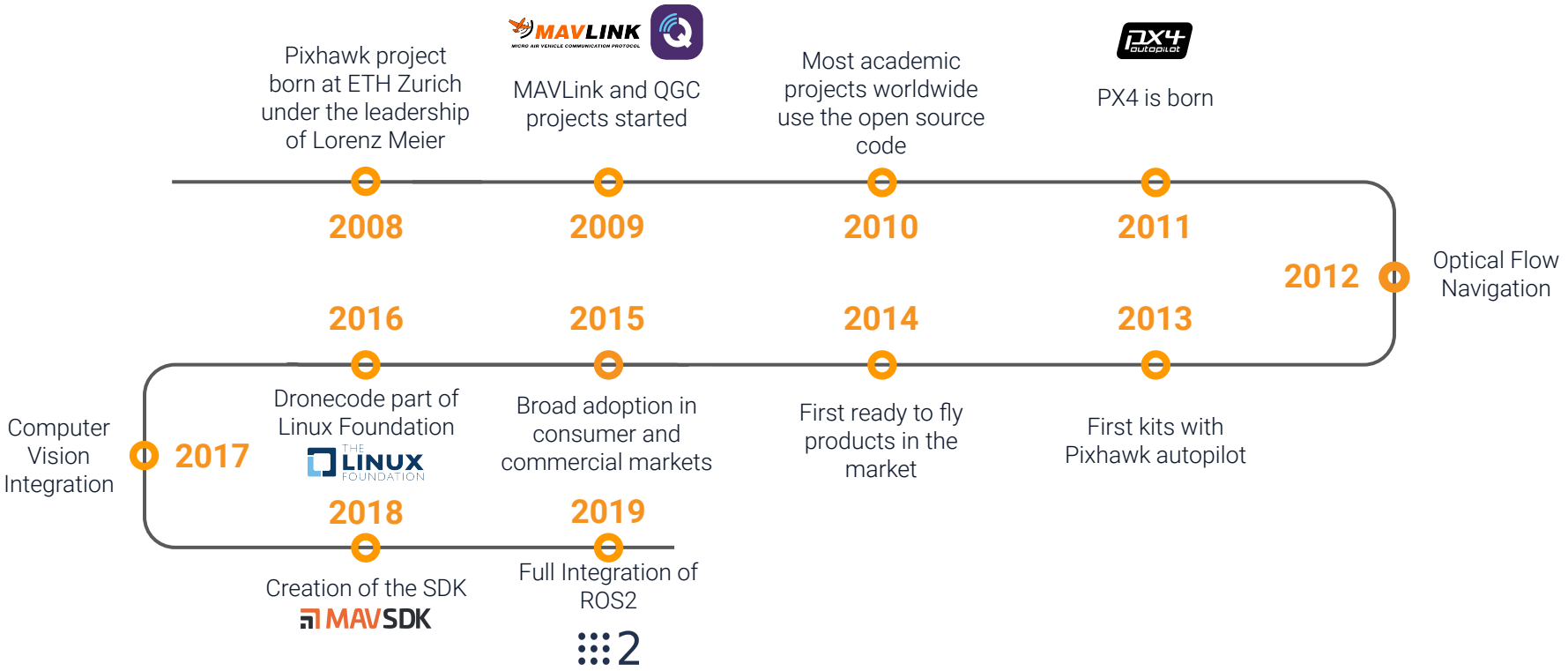
- Martin Golding

Quick introduction: Daniel Agar

daniel@agar.ca

dagar

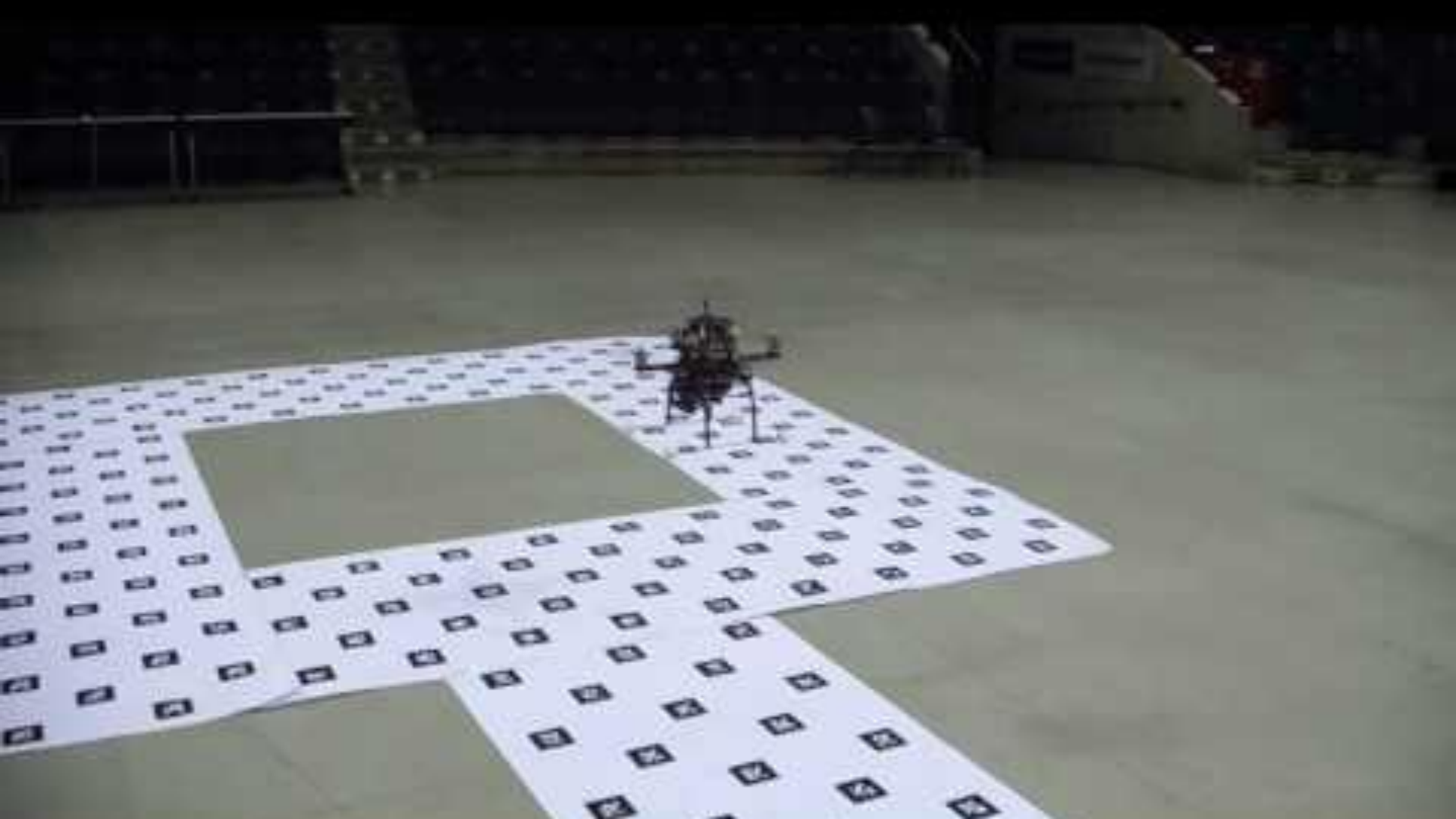
On the same mission for the last decade.



9 years ago..

Control
Connected to MAV 220
Takeoff
Land
Kill
MODE LOCKED
Set Mode
No actions executed so far
Drone Details
Battery
Packet Loss
CPU Load
Top Rotor
Bot Rotor
Left Servo
Right Servo
Object Recognition
No objects recognized
Unmanned Systems
GCS
MAV 220
AUTO MODE
ACTIVE
WPX
Waypoint List
Communication Console
Enter data/text below to send

8 years ago..

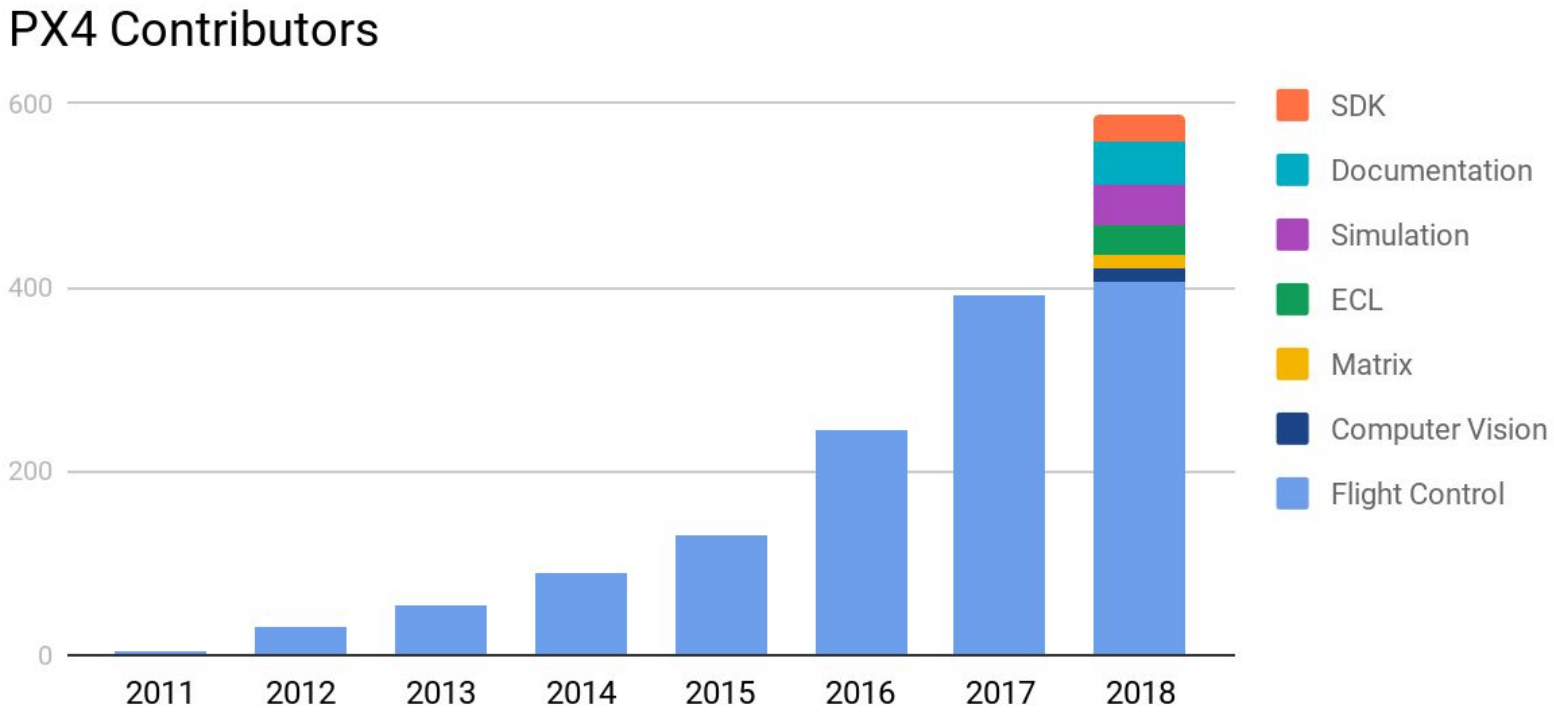


7 years ago..

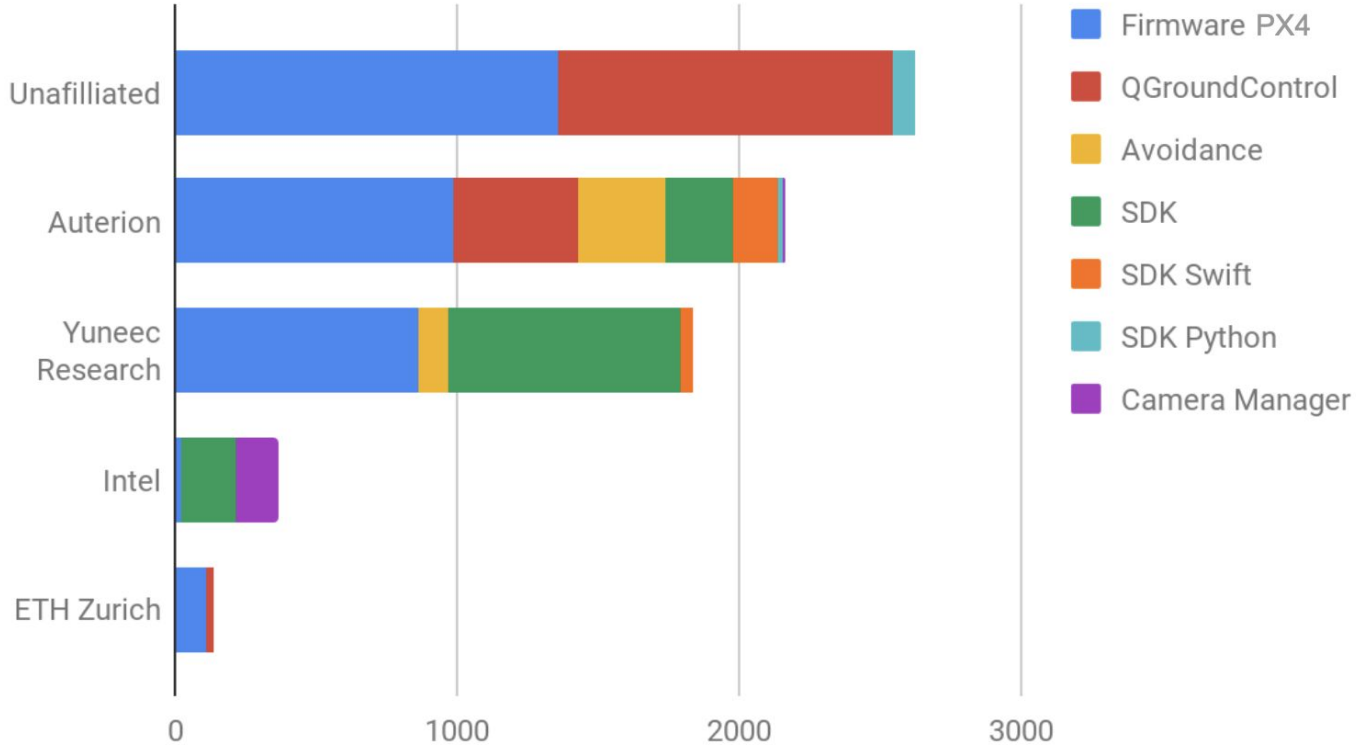
4x speed

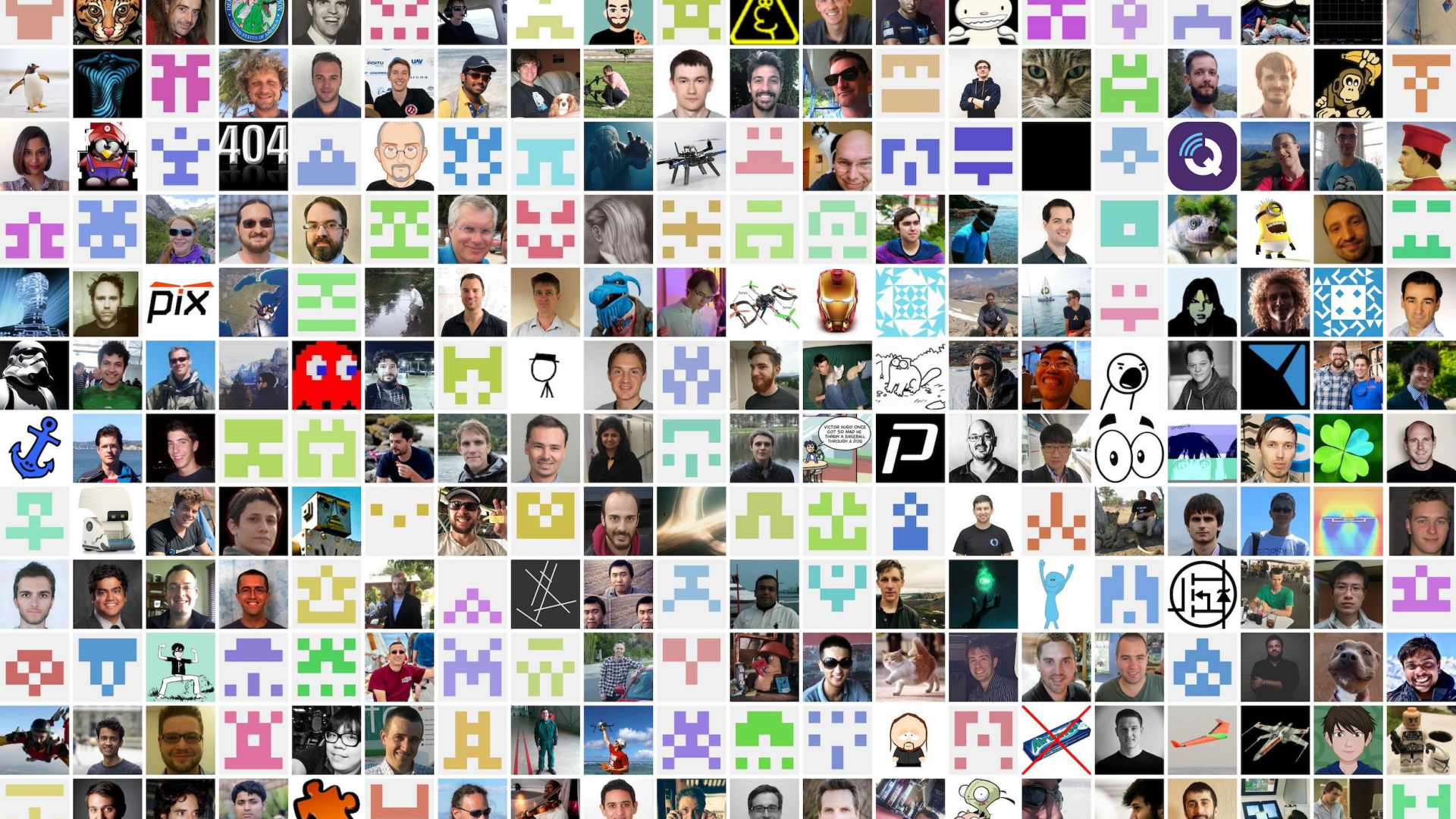
PX4 Today

PX4 has a fast growing community of developers



Who builds PX4 today?



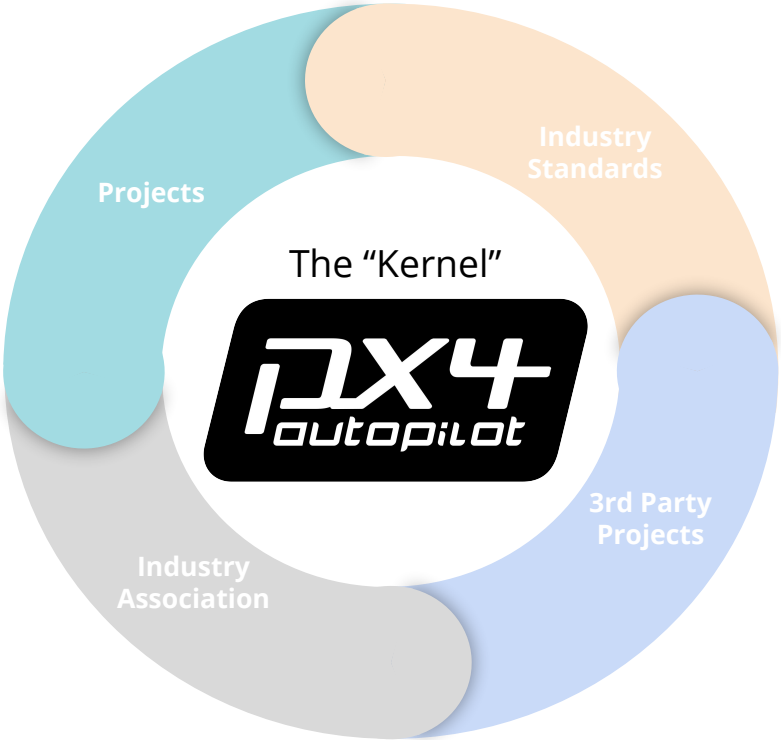

404
pix
VICTOR HUGO ONCE GOT SO MAD HE THREW A BASEBALL THROUGH A DOG

What is the relationship between the parts?

QGroundControl

<dronecode/sdk>

Dronecode run by THE LINUX FOUNDATION



pixhawk

MAVLINK
MICRO AIR VEHICLE COMMUNICATION PROTOCOL

ROS

PX4 is an integrated software ecosystem

Vehicle **Firmware**	**Communication** MAVLINK MICRO AIR VEHICLE COMMUNICATION PROTOCOL	**Ground Control Station** QGroundControl	**Hardware** pixhawk Qualcomm Snapdragon Flight Intel® Aero
Autonomy Vision based localization and avoidance	**Middleware** **ROS2** 2	**API** <dronecode/sdk>	

Leading products and dev platforms based on PX4

YUNEEC

YUNEEC

IMPOSSIBLE AEROSPACE

UVify

FREEFLY

NXP

DHL

amazon Prime Air

QUALCOMM

INSPIRED FLIGHT

TEAL

SONY

sentera

VERTICAL technologies

ELROY AIR

FlightWave

AIRDOG

wingtra

Main Usage of PX4 in the drone industry

Consumer

- Need for reliable base
- Significant customization per product
- Contributions on specific aspects

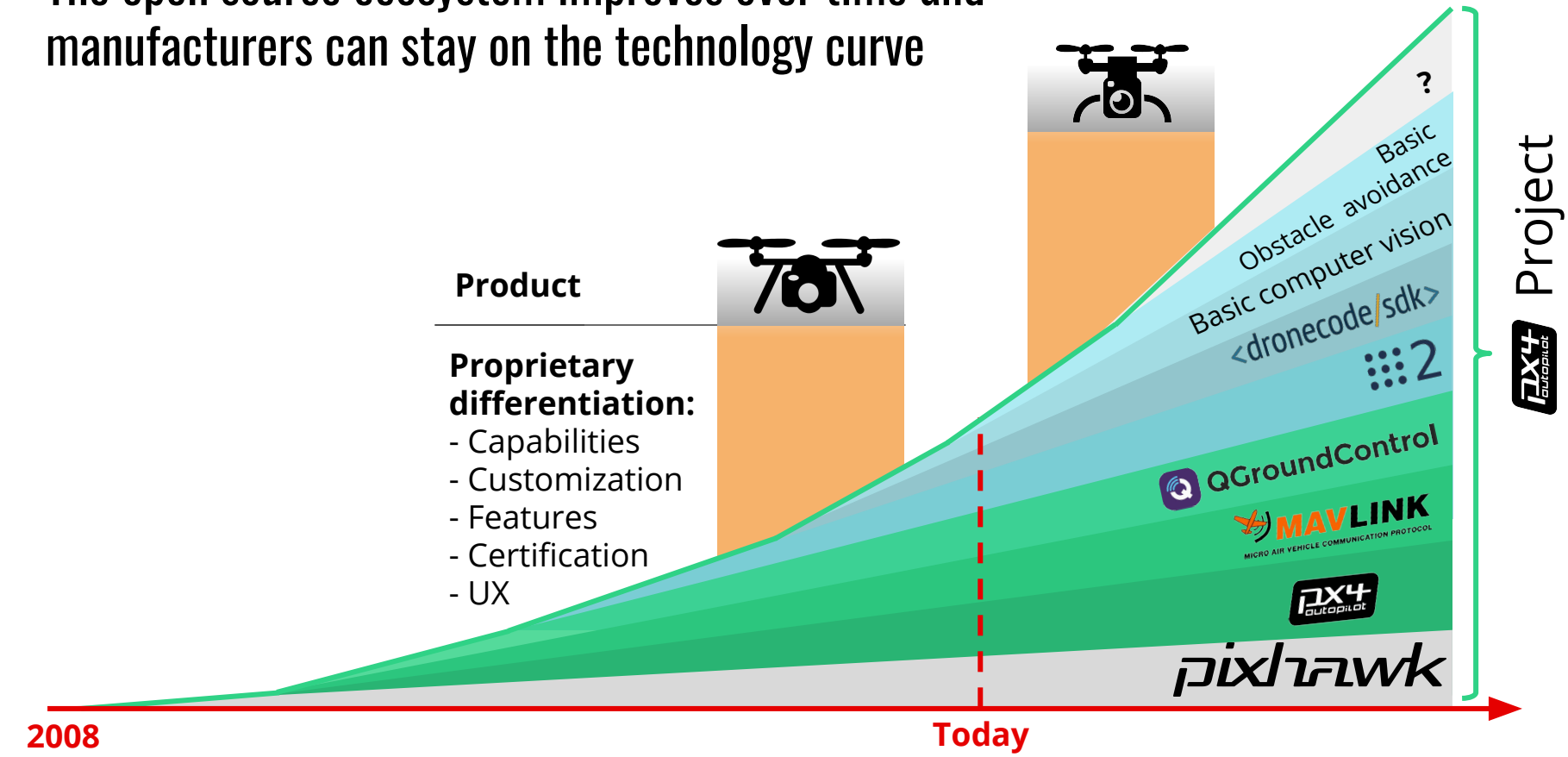
Commercial

- Open system is key value add
- Significant overall contributions to project
- Wide adoption

Industrial

- Capability to deploy additional software is key (apps)
- Deep SW integration
- Mostly VTOL and fast growing segment

The open source ecosystem improves over time and manufacturers can stay on the technology curve



The automotive industry went through a transformation

of manufacturers

1899 1908 1920 1960

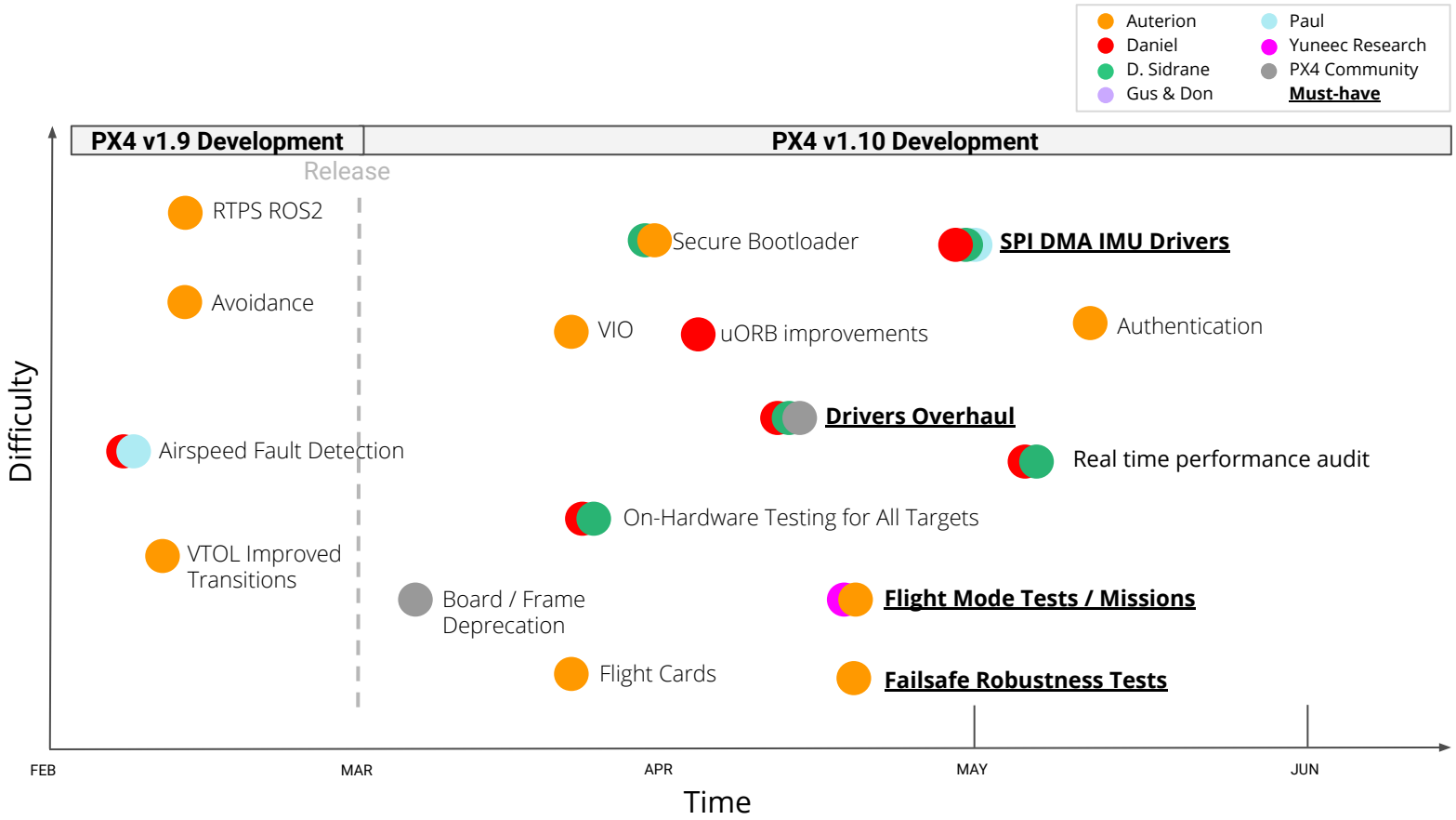
Standards enabled it to scale.

1917

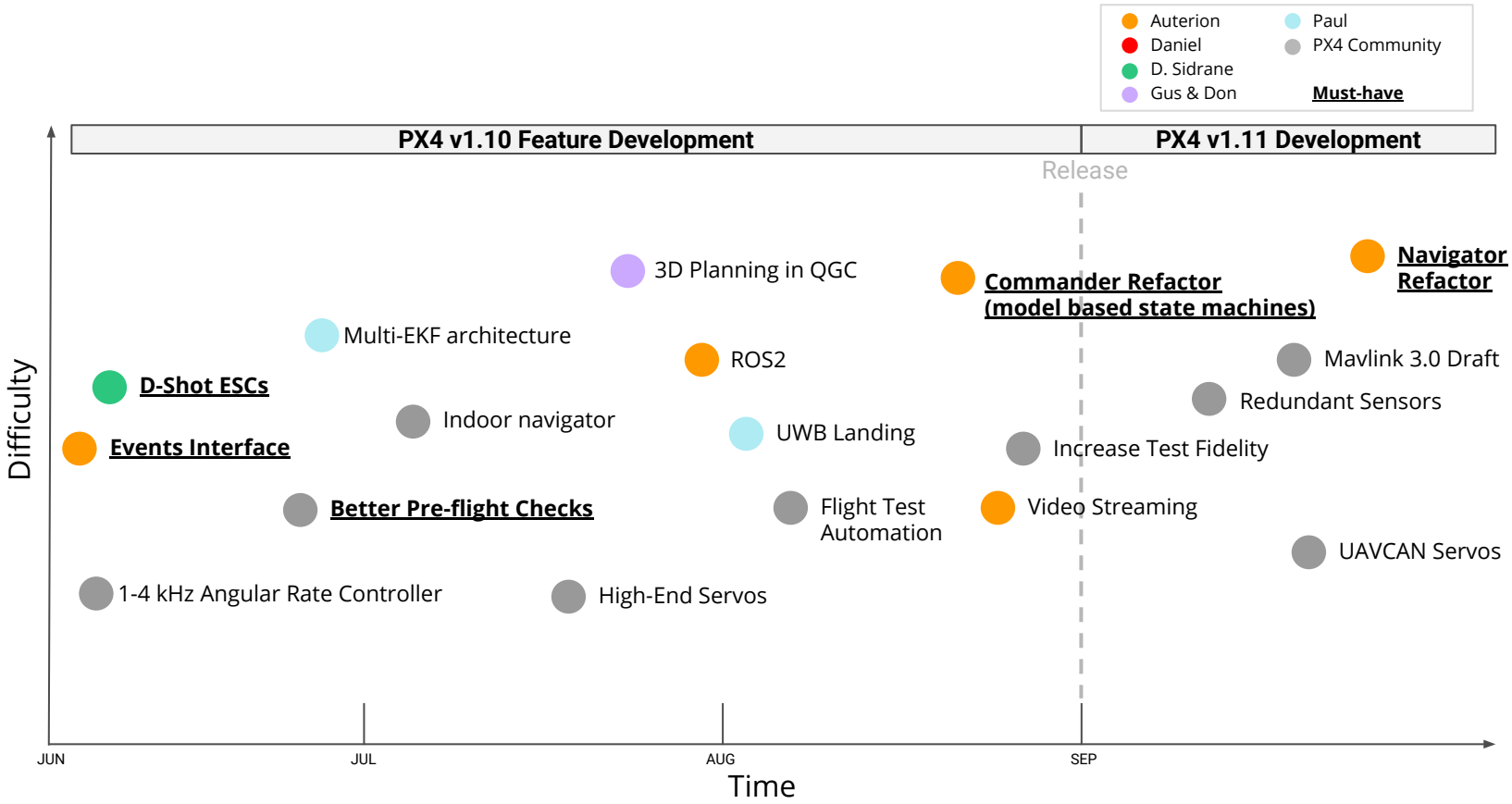
1918

Open Source is the agile approach to standardization

PX4 v1.9: Release June, 2019



PX4 v1.10: Release September, 2019

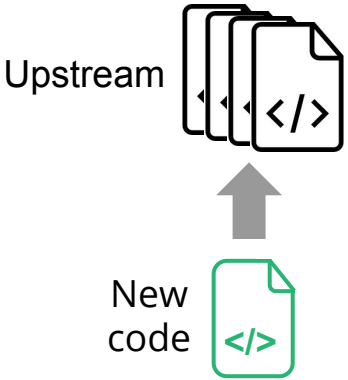


Highlights

Quality Assurance baked into the development process

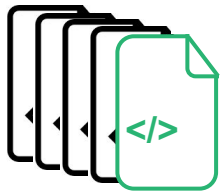
1 → 2 → 3 → 4

Request to pull new code in mainstream over GitHub.



Code maintainer receives the request and starts the review process

1) **Code review**
2) **Automated module tests**
3) **Automated flight simulation**
4) **Physical flight tests by Dronecode team in Mexico**



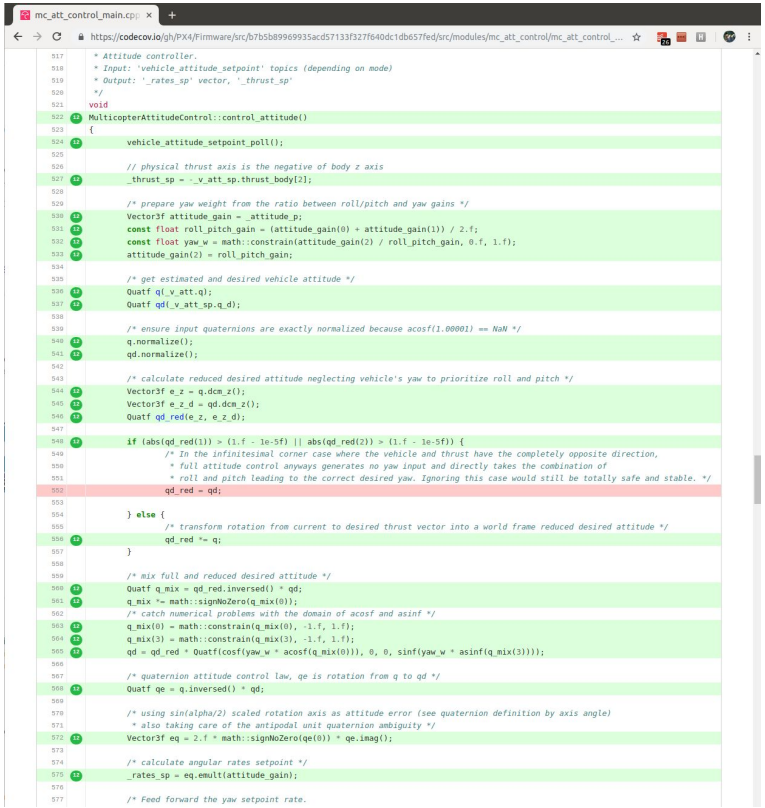
Code merged

Type Certification of PX4 vehicles with FAA

Testing Testing Testing

PX4 official flight controller support means.

- Board part of the hardware test rack, testing every contribution
- Board with the Dronecode test team



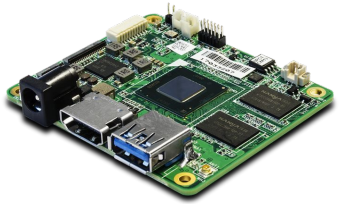
mc_att_control_main.cpp

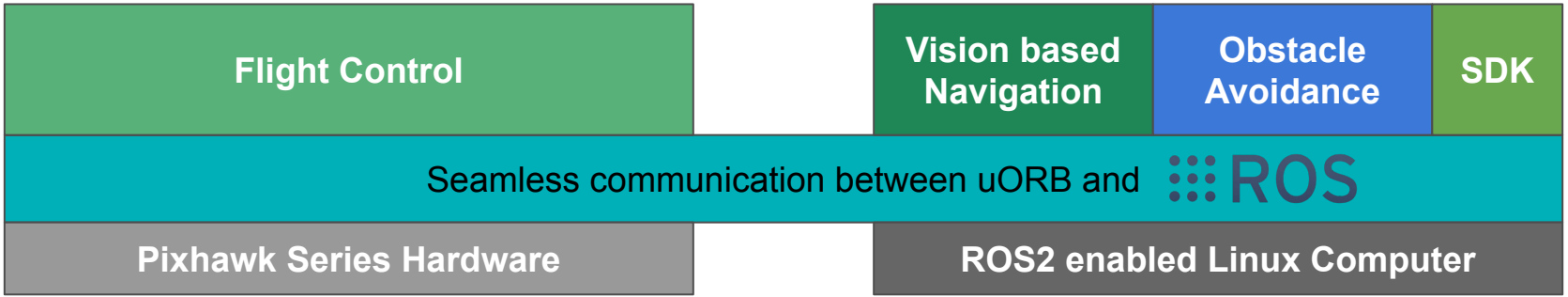
https://codecov.io/gh/PX4/Firmware/src/b7b5b8996993 5acd57133f327f640dc1db657fed/src/modules/mc_att_control/mc_att_control_...

```
517   * Attitude controller.
518   * Input: 'vehicle_attitude_setpoint' topics (depending on mode)
519   * Output: '_rates_sp' vector, '_thrust_sp'
520   */
521  void
522  MulticopterAttitudeControl::control_attitude()
523  {
524          vehicle_attitude_setpoint_poll();
525
526          // physical thrust axis is the negative of body z axis
527          _thrust_sp = -_v_att_sp.thrust_body[2];
528
529          /* prepare yaw weight from the ratio between roll/pitch and yaw gains */
530          Vector3f attitude_gain = _attitude_p;
531          const float roll_pitch_gain = (attitude_gain(0) + attitude_gain(1)) / 2.f;
532          const float yaw_w = math::constrain(attitude_gain(2) / roll_pitch_gain, 0.f, 1.f);
533          attitude_gain(2) = roll_pitch_gain;
534
535          /* get estimated and desired vehicle attitude */
536          Quatf q(_v_att.q);
537          Quatf qd(_v_att_sp.q_d);
538
539          /* ensure input quaternions are exactly normalized because acosf(1.00001) == NaN */
540          q.normalize();
541          qd.normalize();
542
543          /* calculate reduced desired attitude neglecting vehicle's yaw to prioritize roll and pitch */
544          Vector3f e_z = q.dcm_z();
545          Vector3f e_z_d = qd.dcm_z();
546          Quatf qd_red(e_z, e_z_d);
547
548          if (abs(qd_red(1)) > (1.f - 1e-5f) || abs(qd_red(2)) > (1.f - 1e-5f)) {
549                  /* In the infinitesimal corner case where the vehicle and thrust have the completely opposite direction,
550                   * full attitude control anyways generates no yaw input and directly takes the combination of
551                   * roll and pitch leading to the correct desired yaw. Ignoring this case would still be totally safe and stable. */
552                  qd_red = qd;
553
554          } else {
555                  /* transform rotation from current to desired thrust vector into a world frame reduced desired attitude */
556                  qd_red *= q;
557          }
558
559          /* mix full and reduced desired attitude */
560          Quatf q_mix = qd_red.inversed() * qd;
561          q_mix *= math::signNoZero(q_mix(0));
562          /* catch numerical problems with the domain of acosf and asinf */
563          q_mix(0) = math::constrain(q_mix(0), -1.f, 1.f);
564          q_mix(3) = math::constrain(q_mix(3), -1.f, 1.f);
565          qd = qd_red * Quatf(cosf(yaw_w * acosf(q_mix(0))), 0, 0, sinf(yaw_w * asinf(q_mix(3))));
566
567          /* quaternion attitude control law, qe is rotation from q to qd */
568          Quatf qe = q.inversed() * qd;
569
570          /* using sin(alpha/2) scaled rotation axis as attitude error (see quaternion definition by axis angle)
571           * also taking care of the antipodal unit quaternion ambiguity */
572          Vector3f eq = 2.f * math::signNoZero(qe(0)) * qe.imag();
573
574          /* calculate angular rates setpoint */
575          _rates_sp = eq.emult(attitude_gain);
576
577          /* Feed forward the yaw setpoint rate.
```

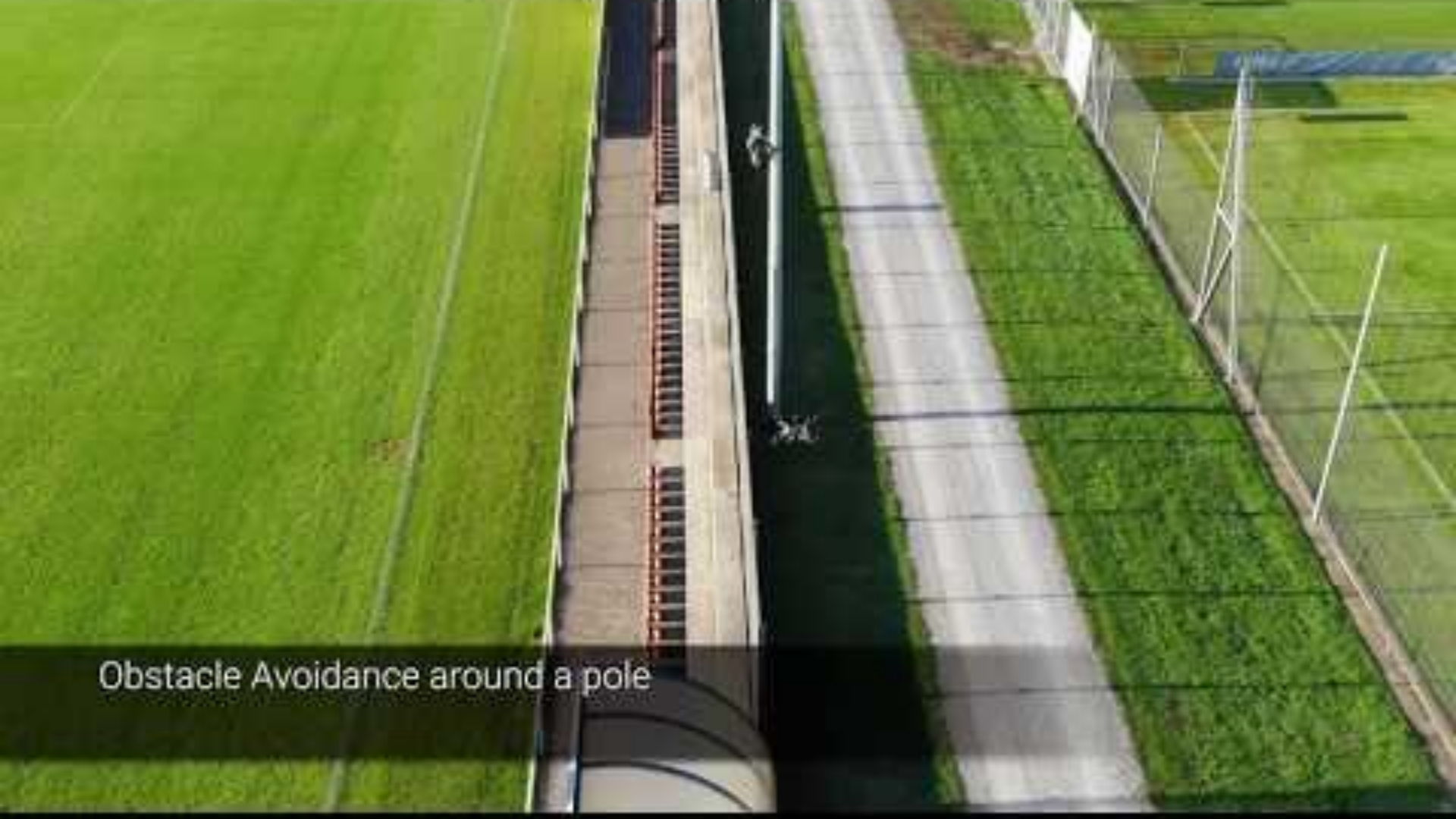
Futureproof Hybrid System architecture

PX4 supports native communication with ROS2 via FastRTPS. This enables integration of a companion computer using industry-standard middleware with safety-critical implementations available. ROS nodes can communicate **today** natively with PX4 without the need for a translation layer like MAVROS.



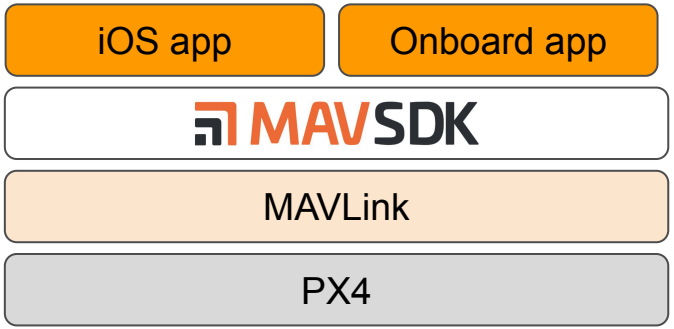


https://dev.px4.io/en/middleware/micrortps.html


Obstacle Avoidance around a pole

MAVSDK is the SDK for MAVLink vehicles

- Language bindings to Python, Swift (iOS), Java (Android), C++
- High-performance core written in C++
- Simple, easy to use API
- For mobile apps and onboard applications
- Extensible by plugins
- Compiles for Linux, macOS, Windows, iOS, Android



Documentation:

- http://github.com/mavlink/MAVSDK

SDK Webinar recording: https://auterion.com/sdk-webinar/

Airspace integration

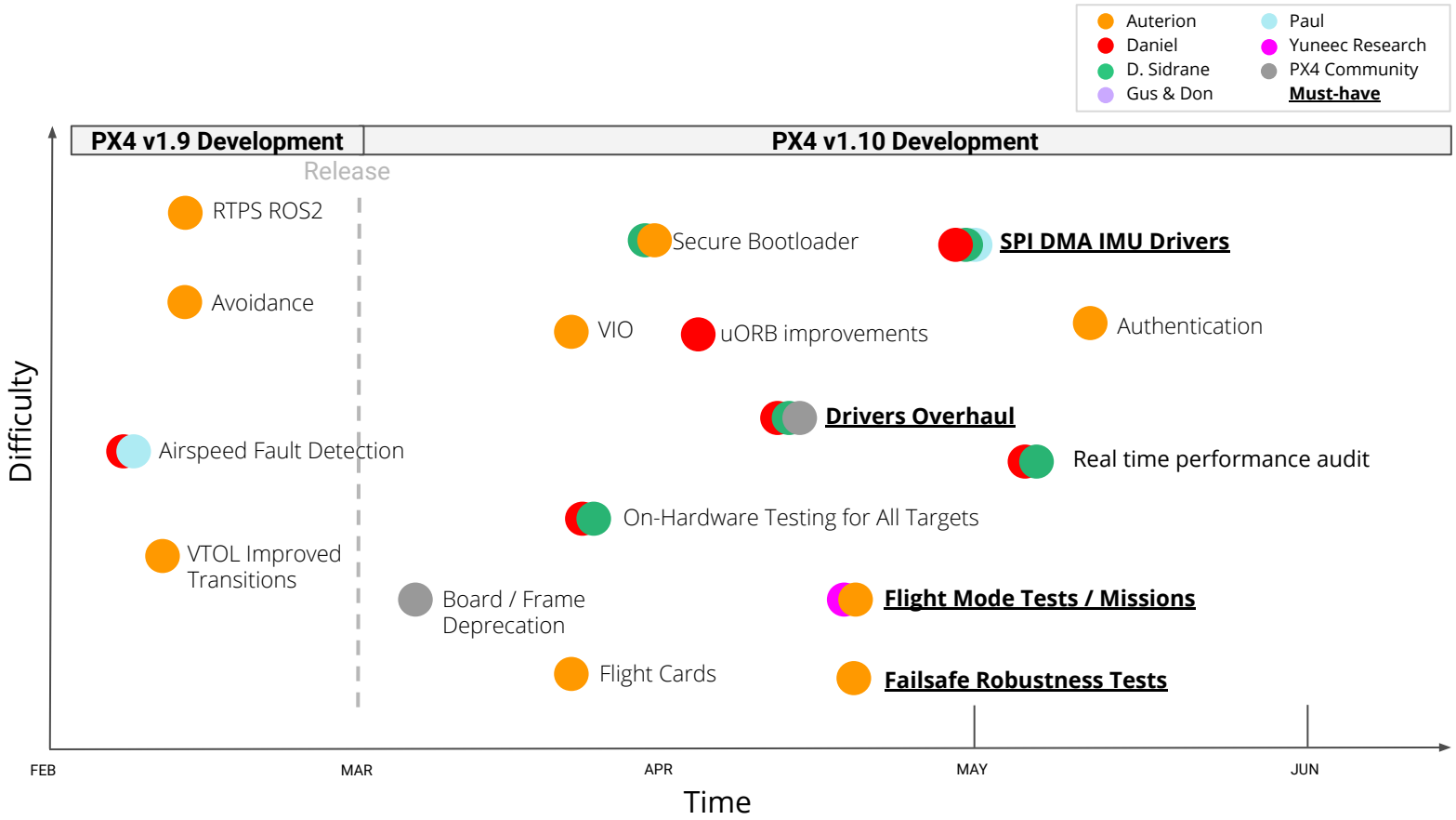
- PX4 + QGC have full airspace integration
  - Airspace maps
  - Geofence upload to the flight controller
  - Telemetry upload to air traffic control (if enabled)
- Field-tested in Geneva in the traffic control zone (u-space demo)
- Based on standard available to all airspace management providers
- Implementation in QGC implemented against Airmap service
  - Generic interface
  - Airmap service implementation of interface

Safety and Certification

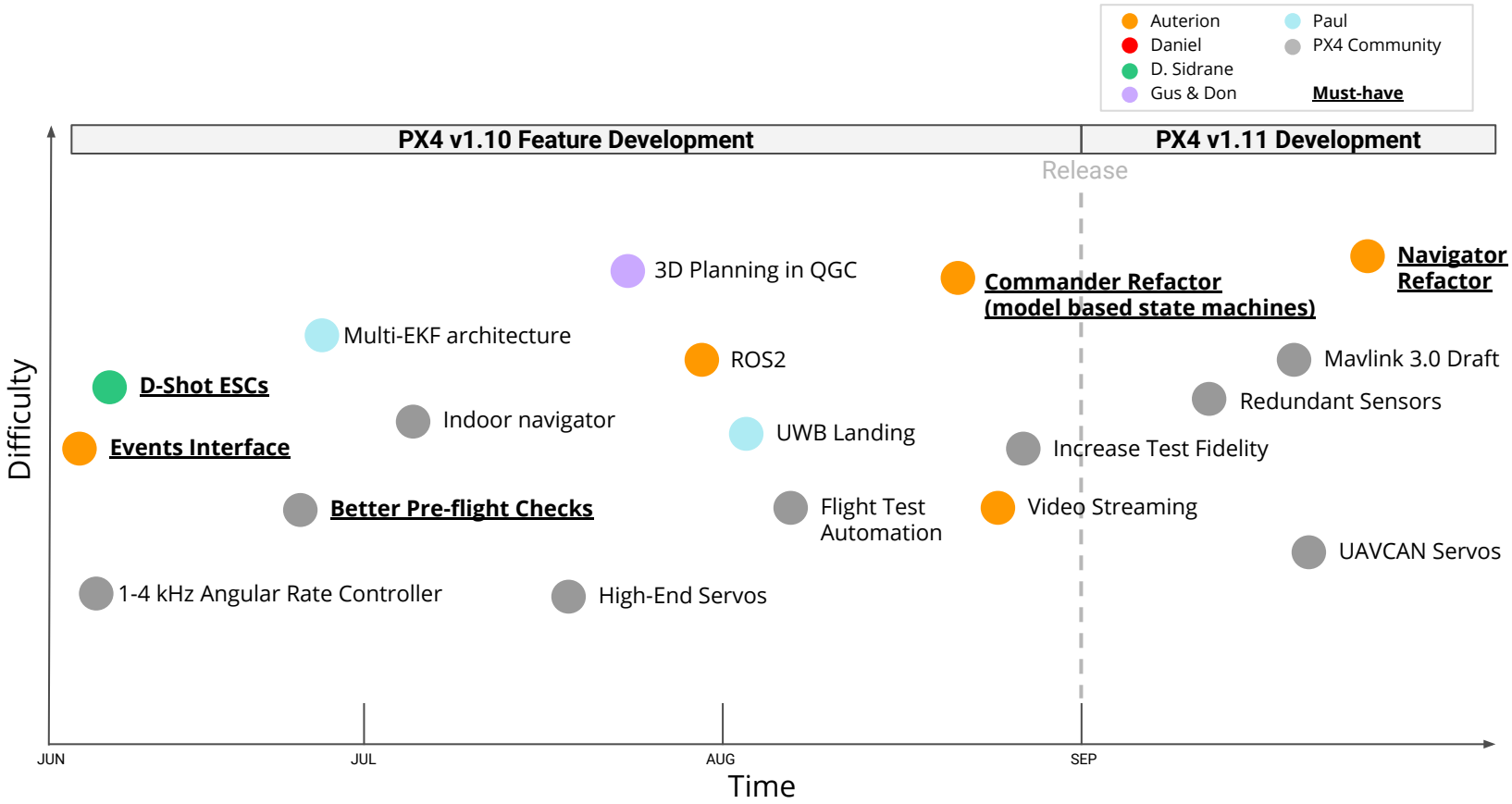
- Initial failure mode analysis of PX4 completed
- Safety analysis project with the functional safety lab at ZHAW
- Discussions on Dronecode level with FAA
- Redundant sensor design and power management
- Automotive-grade sensors and power supply in next-generation hardware reference designs
- Different standards we are looking at for inspiration: ISO26262, DO178

Roadmap

PX4 v1.9: Release June, 2019



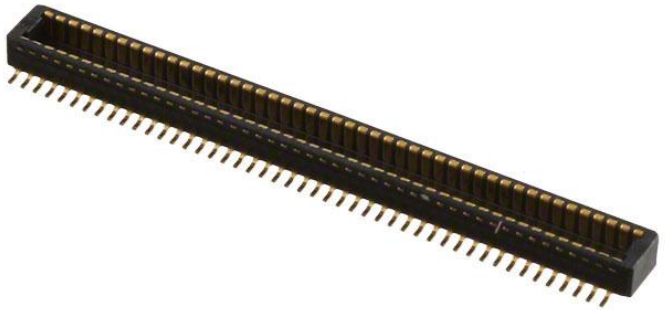
PX4 v1.10: Release September, 2019

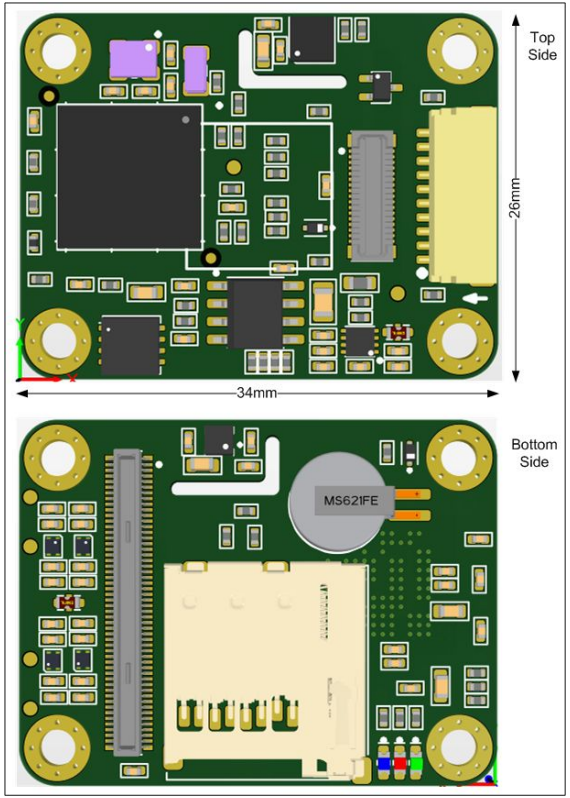


Upcoming Hardware Reference Designs

New hardware generation: Robustified design: FMUv5X

- Mechanical and electrical definition for Autopilot-on-Module (AoM)
- X1 connector: 100-pin connector
    - 34x26 mm core module
    - M2 screws





Thanks!