David Such Follow

Jun 3, 2021 · 15 min read ★ · Listen

A Review of Open-Source Flight Control Systems

The evolution of open-source flight control firmware is fascinating and involves everything from years of committed development with no reward, to convoluted betrayal from previous partners and friends. In this article, we will review the most popular open-source projects, explain their antecedents and highlight the survivors.

1. **The Flight Control Ecosystem**

These days the hardest challenge in designing your own drone is the flight control firmware. For those who can't be bothered spending the months and years crafting this code, there are other options, albeit not much targeting the Arduino platform. Most available flight controller firmware tends to target a niche application, but of course there is a fair amount of overlap as every piece of firmware needs to be able to provide the same fundamental controls. For example, as a minimum, Flight Controllers need to:

- Allow pilot input and convert this to pitch, roll, yaw and throttle speed on the drone.
- Use sensors to provide feedback on the drones' orientation and position which can be compared with the pilot input to determine the correct speed of all the motors.

Thus, any flight controller firmware can be used for any drone mission, but some will be better at it than others. The major drone applications are:

1. Cinematic Video

4. Research.

5. Indoor and Micro Drones.

Using these classifications, we can group the available flight controller firmware, which is one way to select what to use for your application (Figure 1).

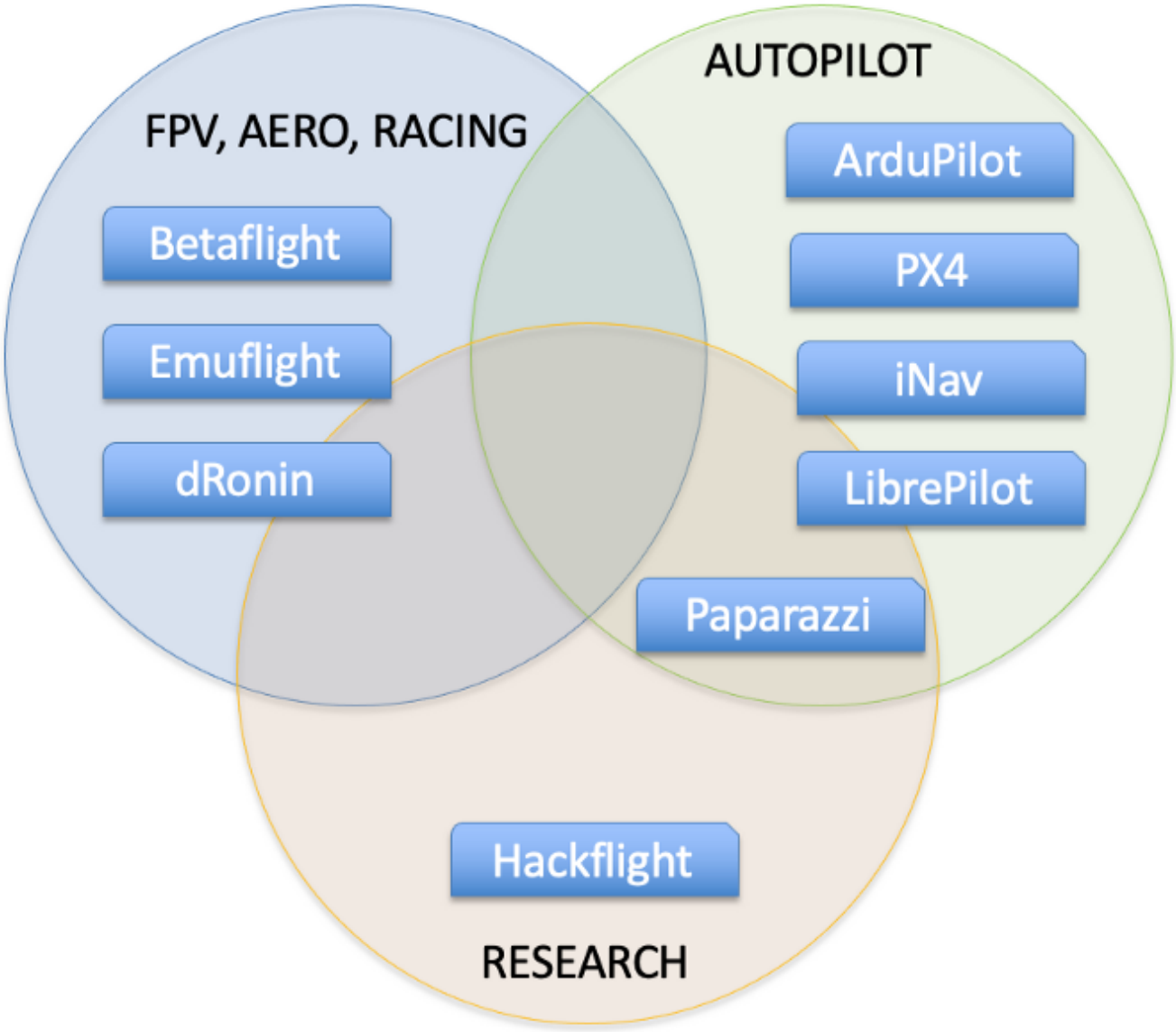
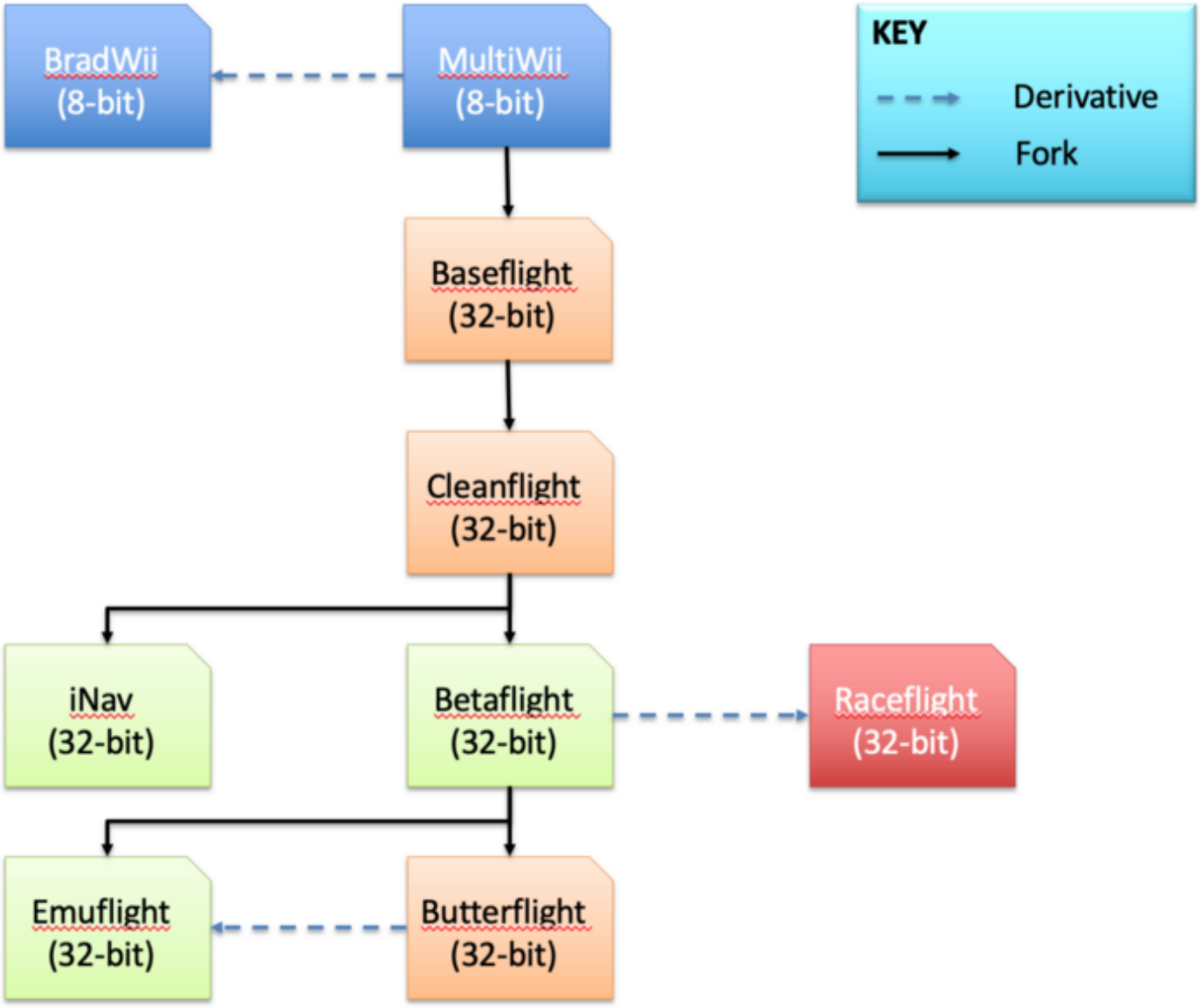


Figure 1. The Open-Source Flight Control Ecosystem

2. The MultiWii Family

It is interesting to observe the ancestry of flight controller firmware. In around 2010, MultiWii was developed by Alexandre Dubus (AKA Alexinparis) to support the Nintendo Wii console gyroscopes and accelerometers. The I2C sensors in the Nintendo Wii Motion Plus and Wii Nunchuk, were repurposed to be used in these early DIY drones. MultiWii targeted the Arduino platform and was able to control a tricopter, a quadcopter or a hexacopter. The available Arduino boards were the Pro Mini and the Mega, but it would work with any Arduino that used the ATmega 328P running at 16 MHz (e.g., Arduino Nano, Arduino Pro, and Arduino Duemilanove). MultiWii ended up being the progenitor of a line of flight controller firmware (Figure 2), those still active are highlighted in green.



Alexinparis put together a schematic for the MultiWii drone hardware layout and illustrated the speed of the various communication protocols used (Figure 3). It illustrates an important point around optimization. There is no point using fast mode for your sensors (i.e., 400 kHz) if you are just comparing this with your control input at 50 Hz.

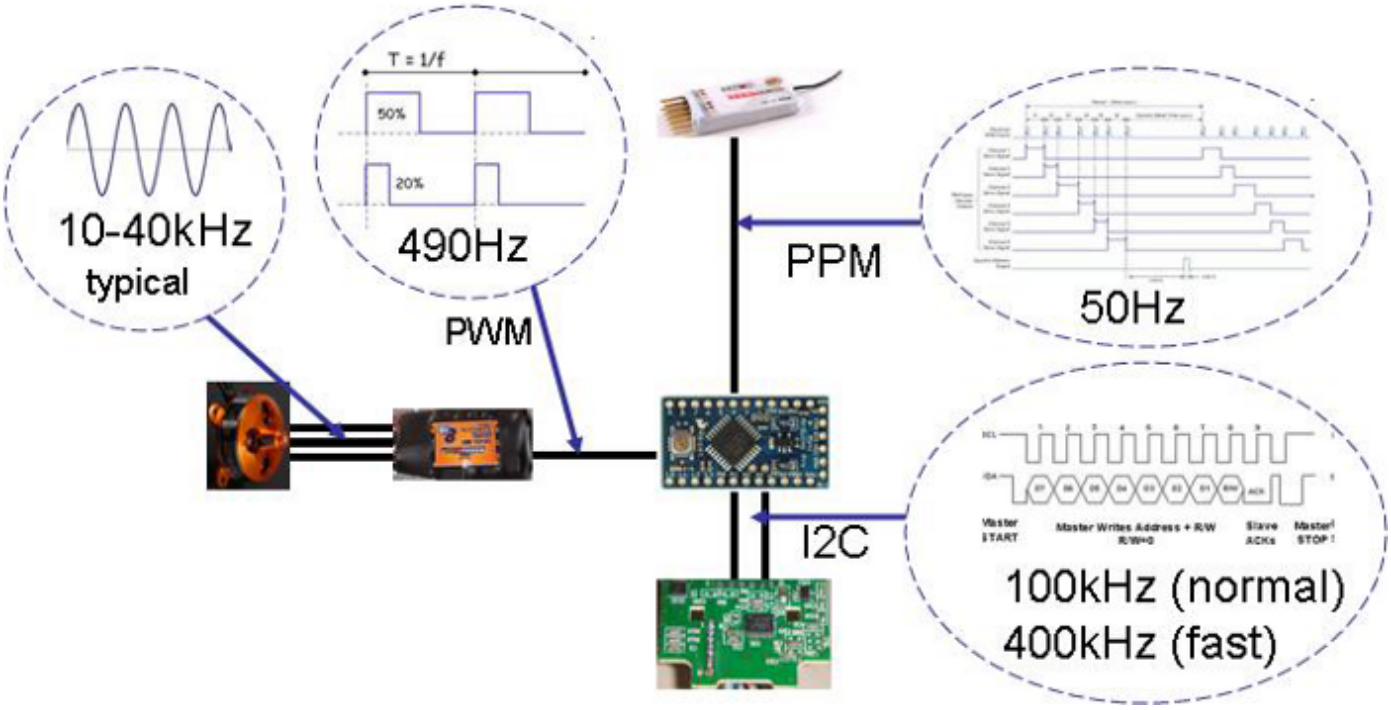


Figure 3. Communication Speeds in the MultiWii (credit: Alexinparis)

While not being maintained any longer, MultiWii is still available on GitHub *https://github.com/multiwii/multiwii-firmware*.

2.2 BradWii

BradWii is another open-source derivative of MultiWii which kicked off around 2013. It is not really a fork of MultiWii, but many of the concepts and code, were borrowed from it. The name is also a play on the original developer's name, Brad Quick. Though based on Multi-Wii, BradWii is pretty much a complete re-write. BradWii was intended to be a platform on which other projects can be built.

BradWii is not being maintained but the code is available on Brad's GitHub repository

Baseflight was a 32-bit fork of MultiWii and moved away from the Arduino platform to STM32F1 boards. Baseflight was used with Naze32 flight controllers, which were once the most popular flight controller for the mini quadcopter. The lead developer for the Baseflight fork was a chap with the pseudonym of timecop (subsequently changed to trollcop, possibly due to controversy over the Cleanflight fork). Timecop has a website *http://abusemark.com/*, where he used to sell the Naze32 and other hardware.

Baseflight is also not being maintained, but like MultiWii is available on GitHub *https://github.com/multiwii/baseflight*. By all accounts, timecop was a somewhat prickly character, but he deserves credit for being one of the people leading DIY flight controller development from 2014 to 2015.

2.4 Cleanflight

When Dominic Clifton (AKA Hydra) forked Baseflight to create Cleanflight in 2014, he really upset timecop (the lead on Baseflight) and Stefan (AKA cTn) the developer of Baseflight Configurator.

Dominic worked on Baseflight originally but after a disagreement with timecop regarding the future direction of the code, and some subsequent flaming thread discussions, was kicked from the team.

As he had a software development background, Dominic's focus was on creating maintainable code which supported a variety of hardware. History has supported his decision. He also added some new features to the code base like OneShot and RGB LED support.

There is a close relationship between Cleanflight and Betaflight. Every so often Cleanflight forks Betaflight to take advantage of their leading-edge development. As mentioned, Cleanflight was originally forked from Baseflight, then Cleanflight was forked by Betaflight, and Cleanflight was again forked from Betaflight. The process looks something like this:


Cleanflight is still being maintained but it is essentially Betaflight at the moment. Dominic took a leaf out of timecop's book and released his own flight controller hardware which runs Cleanflight software. This hardware is branded Seriously Pro Racing (SPRacing), and you can check out his boards at *http://seriouslypro.com/*.

The Cleanflight GitHub repository is *https://github.com/cleanflight/cleanflight*.

2.5 Betaflight

Betaflight was forked from Cleanflight in 2015 by Borisbstyle. However, it wasn't as simple as that. Boris started distributing modified binaries of Cleanflight for people to test, when he couldn't get his pull requests accepted. This eventually morphed into Betaflight and explains the name. If Dominic had not been distracted in his move from the UK to Spain, this fork may never have happened.

Leading an Open-Source project is hard, particularly if it becomes popular and you are trying to hold down a full-time job and have some sort of private life. It is a credit to the Betaflight developers that this fork continues to be popular, despite Boris starting to focus on other interests in 2017.

Betaflight's primary focus is leading edge performance even if this comes at the price of stability. An overview of the Betaflight system architecture is shown in Figure 4. It is currently the predominant FPV flight controller firmware, but the landscape keeps changing.

Betaflight do not have their own hardware but support a large number of STM32 based flight controllers. This may help explain the longevity of Betaflight compared to other open-source projects, as they haven't had to split their development efforts. They are also not competing with the companies that use their firmware.

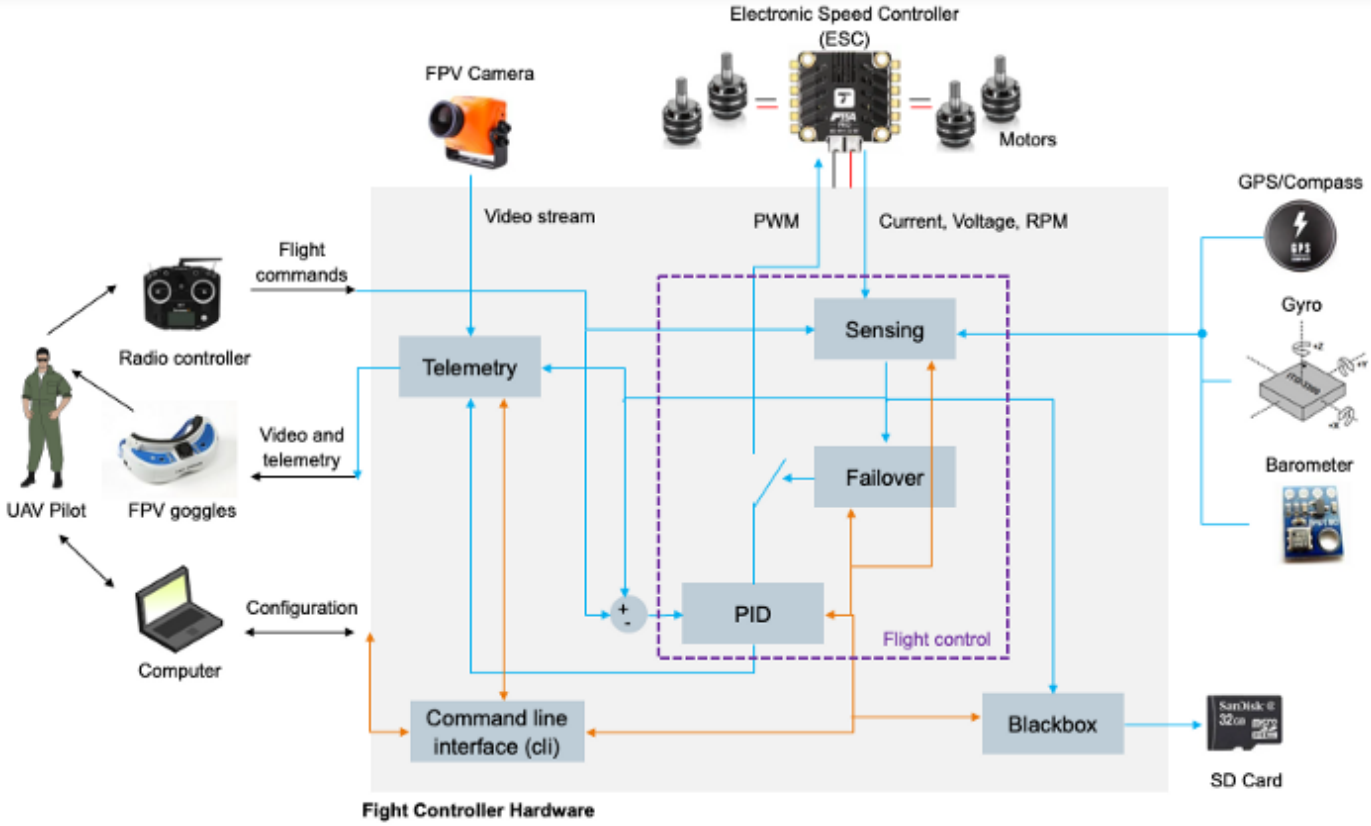


Figure 4. Typical Betaflight Technical Stack (credit)

The Betaflight GitHub repository is available at *https://github.com/betaflight/betaflight*.

2.6 iNav

iNav and Betaflight are both forks from Cleanflight (Figure 2), but they have very different objectives. While Betaflight focuses on racing, FPV and acrobatic mode, iNav is focused on navigation capabilities and a broader range of vehicle support. iNav works on quadcopters, but it also targets [bi|tri|hexa|octa]copters, fixed-wing aircraft, gliders, flying wings, rovers, cars, tanks, and boats.

Position estimation is an essential part of navigation. iNav uses data from all available sensors and fuses them together to calculate position and velocity in the local frame of reference. All navigational decisions are made based on this estimated position/velocity data. The flowchart for this process is shown in Figure 5.

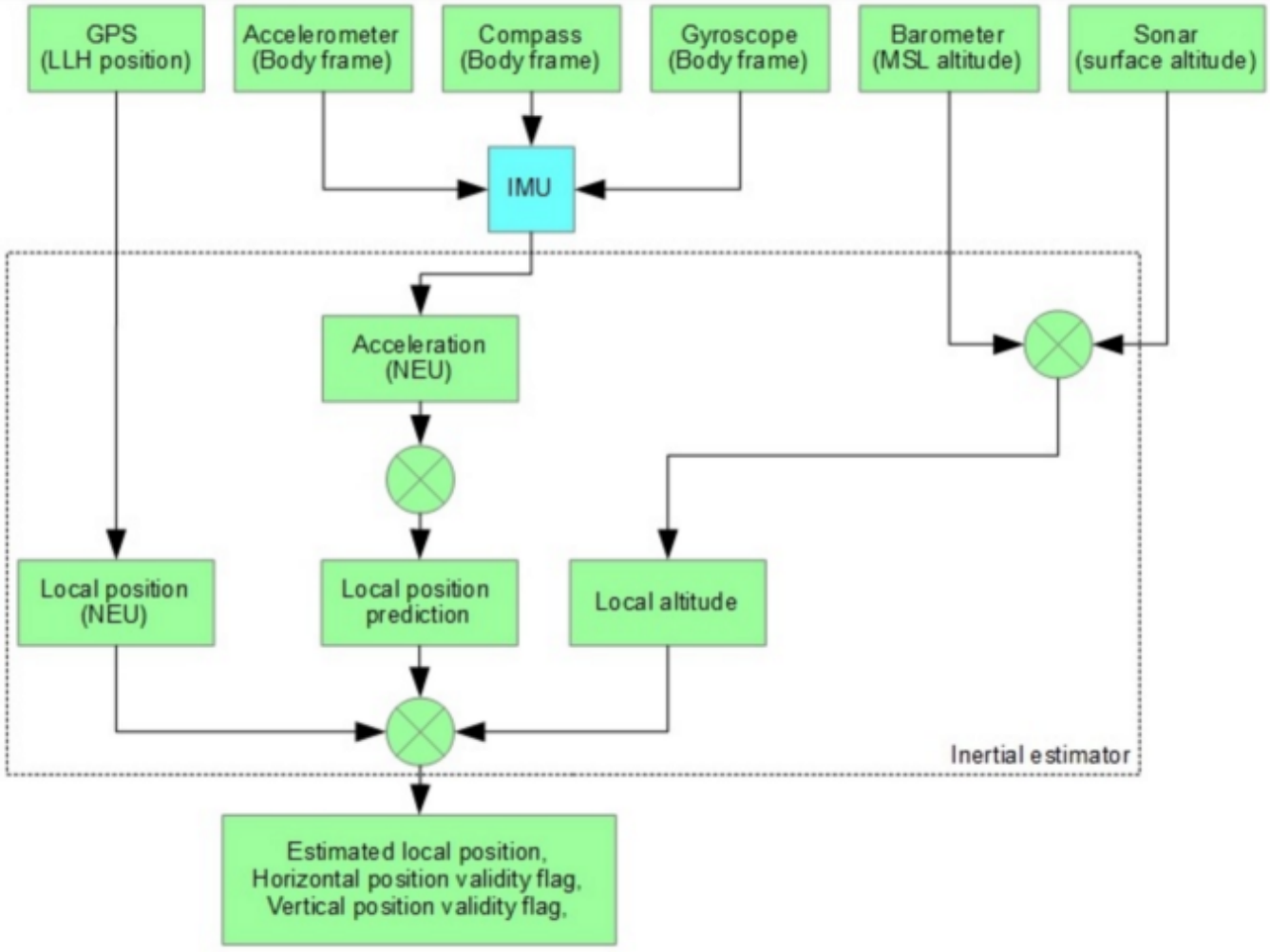


Figure 5. iNav Position Estimate Flowchart (credit)

iNav is being actively developed and the GitHub repository is at *https://github.com/iNavFlight/inav*.

2.7 Raceflight/FlightOne

The Raceflight "fork" led a brief but troubled life! It appeared on the scene as a closed source competitor of Betaflight. Kalyn Doerr started development on Raceflight in 2015. Kalyn then joined up with Preston Garrison to form a company to sell flight controller hardware that ran Raceflight firmware. Unfortunately, Kalyn and Preston had a falling out, and in 2017 Kalyn published the Raceflight proprietary source code for everyone to see. This didn't go down well with Preston, and he issued a DMCA takedown notice on the

was a big legal stoush and Raceflight changed its name to Raceflight One and then to FlightOne and everyone was suing everyone else. This brouhaha involved Helio and Butterflight as some of those involved swapped teams.

It isn't clear how this all played out in the end but today you can visit the Raceflight repository on GitHub (https://github.com/rs2k/raceflight) and see that it is published under the GPL v3 licence and stating that it is a fork of Betaflight. As with a number of other forks, this code isn't being maintained so you would be better off using the latest version of Betaflight.

2.8 Butterflight

Butterflight is all about the filters. Our protagonist from Raceflight, Kalyn Doerr, wrote a fast Kálmán filter for Betaflight as an alternative to their Biquadratic (2nd order) RC Low Pass Filter (LPF) and Finite Impulse Response (FIR2) filter. As with any Open-Source project, there was some disagreement about which filter was better. The developers who wrote the RC and FIR2 filter claimed that it was functionally equivalent to the Kálmán filter and was optimized to run on older hardware. Due to timing, space constraints, computational overhead and legacy hardware support issues, the Kálmán filter code was left out of Betaflight, and Kalyn decided to go his own way with Butterflight. This was in 2018.

The hardware associated with Butterflight was produced by HelioRC. These boards were unusual (for the MultiWii family) in that they used two processors (a STM32F4 and a STM32F3) to spread the computational load. OpenPilot used a similar hardware approach and this may be where HelioRC got the idea from.

Betaflight now includes a fast Kálmán filter but no longer supports STM32F3 and earlier processors. As with any engineering decision there are always compromises.

Butterflight is not being actively maintained but you can access the code at its GitHub repository *https://github.com/ButterFlight/butterflight*.


experience. It was released in 2019

EmuFlight was first released (v 0.1.0) in September 2019 by Kevin Plaizie. The last Butterflight based version (0.4.0) was released in 2021. Moving forward, EmuFlight has now forked Betaflight and updated that based on learnings from EmuFlight since 2019.

EmuFlight is available from its GitHub repository *https://github.com/emuflight/EmuFlight*.

3. The OpenPilot Family

3.1 OpenPilot

OpenPilot development commenced at the beginning of 2010 by David Ankers, Angus Peart and Vassilis Varveropoulos. OpenPilot was a platform aimed at civilian and research purposes. Its objective was making a platform for aerial photography and aerial video applications.

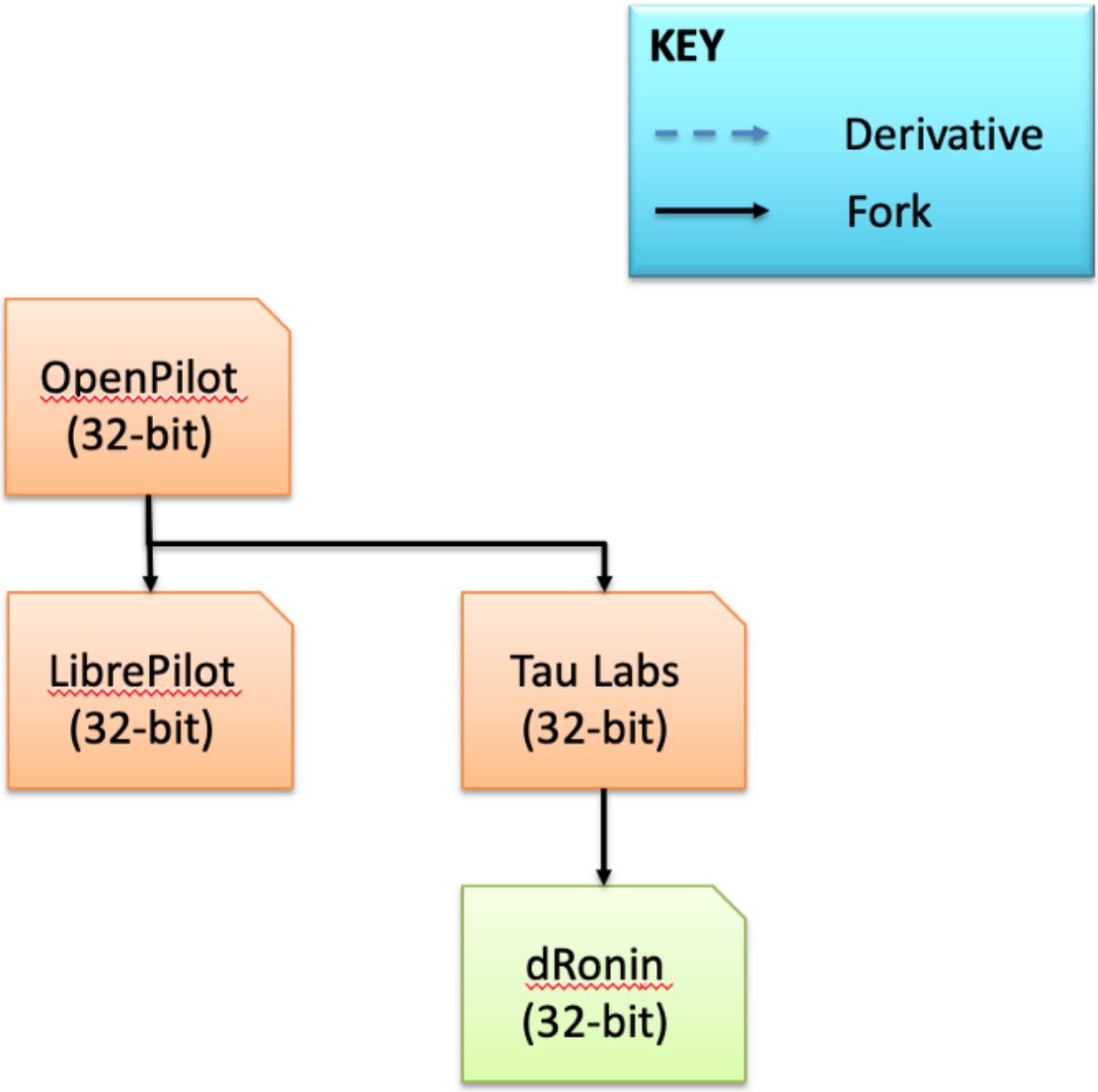



Figure 6. The OpenPilot Family Tree

OpenPilot's hardware is based on the STM32 microcontroller. There are two boards, the first contains the core microcontroller, SD socket, barometer plus servo connectors. The second contains the sensor hardware.

The OpenPilot sensor board was a 9DOF unit and contains MEMS gyroscopes,

inertial measurements are presented back to the main OpenPilot board using a SPI interface. HelioRC took a similar hardware approach with Butterflight.

Like the MultiWii family, OpenPilot software was released under the GPL version 3 license and spawned a number of other flight controller firmware (Figure 6).

While no longer supported the repository for the OpenPilot code is available at *https://github.com/openpilot/OpenPilot/*.

3.2 LibrePilot

When one of the original OpenPilot developers, David Ankers, announced that he planned to leave OpenPilot and fork the code to create OPNG (Open Pilot Next Generation), there was disagreement about what would happen with the OpenPilot assets like the forum and domain[1].

In July 2015, OpenPilot was forked by a number of the other developers to create LibrePilot. Like OpenPilot, it focused on research and development of software and hardware to be used in vehicle control, stabilization, unmanned autonomous vehicles and robotics.

LibrePilot is no longer being maintained but the source code is available at the GitHub repository *https://github.com/librepilot/LibrePilot*.

3.3 Tau Labs

Tau Labs was forked from the OpenPilot project in November 2012. Its focus was on autopilots which can be used as the basis for research projects. The Tau Labs target audience was professionals, researchers, and students.

Tau Labs is no longer being maintained but the code is still available on its GitHub repository at *https://github.com/TauLabs/TauLabs*.

3.4 dRonin

aimed at a variety of use cases including acrobatics/racing, autonomous flight, and vehicle research.

Figure 7. dRonin Configurator

For a while dRonin was used on the BrainFPV hardware but they then moved to Betaflight.

dRonin is being developed, but the pace has slowed. It seeks to improve acro/racing performance and autotuning on all targets. They also want to develop better autonomous flight and navigation functionality. As with all our other firmware, there is a configurator to assist with tuning and testing (Figure 7).

The dRonin code is available at its GitHub Repository *https://github.com/d-ronin/dRonin/*.

4. The DroneCode Foundation

In 2014, Chris Anderson was instrumental in setting up the DroneCode Foundation. This

At its peak, 3D Robotics had offices in the Bay Area, Austin, San Diego and Tijuana; employed more than 350 people; and was valued by investors — which included Qualcomm Ventures, Richard Branson and True Ventures — at more than $360 million. Unfortunately, their premier drone development called Solo, never matched the popularity of the DJI offerings, and the company no longer makes drones.

The open-source ArduPilot platform (including ArduCopter, ArduPlane, and ArduRover), grew out of collaboration within the DIY Drones community.

Around the same time, Lorenz Meier a postdoc at the Swiss Federal Institute of Technology in Zurich, built his own system: PX4, an open-source autopilot for autonomous drone control. He also led the MAVlink project.

In 2014, the ArduPilot, PX4, QGroundControl and MAVlink open-source projects all joined the DroneCode Foundation.

Then in September 2016, it all went wrong. The relationship between ArduPilot and 3DR/Chris Anderson/PX4 soured because the DroneCode Platinum board members (3DR, Intel, and Qualcomm) outvoted Silver board members to remove GPL version 3 projects including ArduPilot from the DroneCode Foundation. Ever since, ArduPilot and PX4 have gone their separate ways.

4.1 ArduPilot

The ArduPilot project commenced in late 2007 when Jordi Muñoz, wrote an Arduino program (which he called “ArduCopter”) to stabilize an RC Helicopter. In November 2009, Jordi created the ArduPilot code repository.

Early versions of ArduPilot used the APM flight controller, an AVR CPU programmed using the Arduino IDE. It has since been rewritten in standard C++ and runs on a diverse range of closed and open-source hardware.

The ArduPilot project is split into a number of sub projects which target specific platforms:


- Rover
- ArduSub
- Antenna Tracker

Figure 8. Mission Planner Screen Shot

Mission Planner is a ground station application for the ArduPilot open-source autopilot project (Figure 8). It is part of the ArduPilot suite of applications.

ArduPilot is still being developed, licensed under the GPL Version 3 and is free to download and use. The software is available at their GitHub repository *https://github.com/ArduPilot/ardupilot*.

4.2 PX4

In 2008/2009, Lorenz Meier started a research project alongside his master's degree at

Following difficulties in scaling the project, in 2011 they scrapped the software and hardware built over the previous three years and did a complete rebuild. The fourth rewrite of the PX flight control software (shorthand for Pixhawk) was called PX4. Two years later, the first stable release was unveiled.

Similar to ArduPilot, PX4 runs on a diverse range of closed and open-source hardware.

PX4 is still part of the DroneCode foundation, and the software is open-source and provided under the *BSD 3-clause license*. PX4 is actively maintained, and the source code is available at the GitHub repository *https://github.com/PX4/PX4-Autopilot*.

5. Paparazzi

Paparazzi is an open-source autopilot system oriented towards research. The project began in 2003 and is being further developed and used at École nationale de l'aviation civile (ENAC), a French civil aeronautics academy. It was initially designed for fixed wing aircraft.

It now encompasses autopilot systems and ground station software for multi-rotors, fixed-wing, helicopters and hybrid aircraft. Paparazzi UAV was designed with autonomous flight as the primary focus and manual flying was a secondary consideration.

Historically, Atmel AVR MCUs were used for the flight controller hardware. This transitioned to Philips/NXP ARM7 LPC21 microcontrollers. More recently, support for the STMicroelectronics STM32F series of microcontrollers has been introduced. These boards include one or two microcontrollers and the required connectors to handle the servos, motor controllers, sensors, RC receiver, radio modem, and a variety of payloads. In March 2017, ENAC released a flight controller board called Chimera which is based on the STM32F7 MCU. The Paparazzi hardware designs are also open-source and available at *https://github.com/paparazzi/paparazzi-hardware*.

The Paparazzi code is released under the GNU GPL Version 2 license and the GitHub repository is at *https://github.com/paparazzi/paparazzi*.


Hackflight is a simple, platform-independent, header-only C++ toolkit for building multirotor flight controllers. It is geared toward people who want to tinker with flight-control firmware and use it to teach students about ideas like inertial measurement and PID tuning. It was written by Simon Levy a Professor at the Computer Science Department at Washington and Lee University.

The project began in 2015 as an attempt to build a simple open-source flight-control firmware program for drones using the Arduino platform. Unlike the preceding flight control firmware, which supports a variety of vehicle types (multirotor, fixed-wing aircraft, ground vehicles, marine vehicles), Hackflight only supports multirotor.

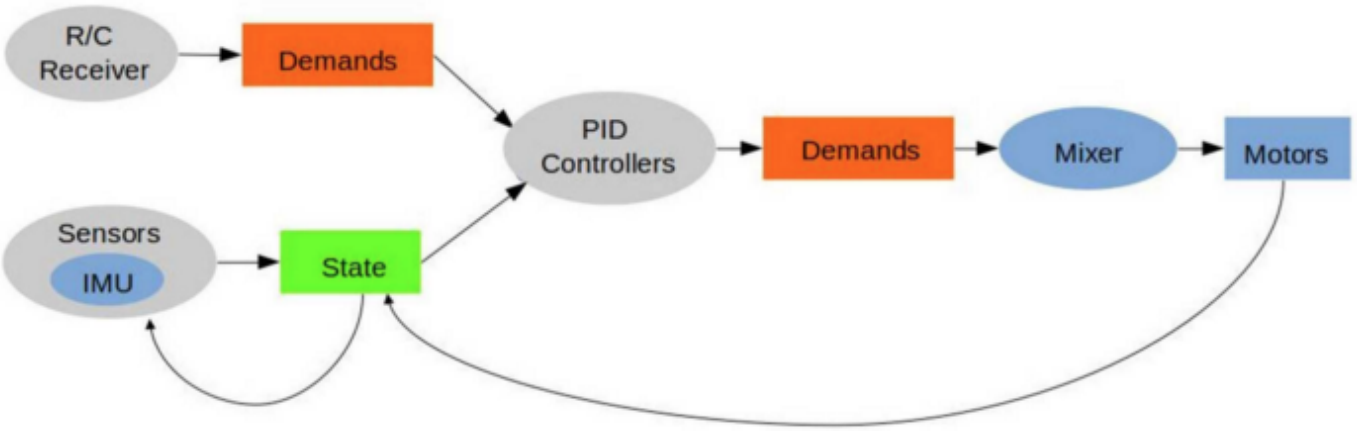


Figure 9. Hackflight Main Loop (credit)

Hackflight uses only the bare minimum of PID controllers necessary for stable flight. It matters in which order you add PID controllers, because the output of one PID controller is the input to the next. For example, to get Stabilize mode, you want the Level controller to go first, setting the desired pitch/roll angles, and the Rate controller to go next, to control the rate at which the desired angle will be reached (Figure 10).

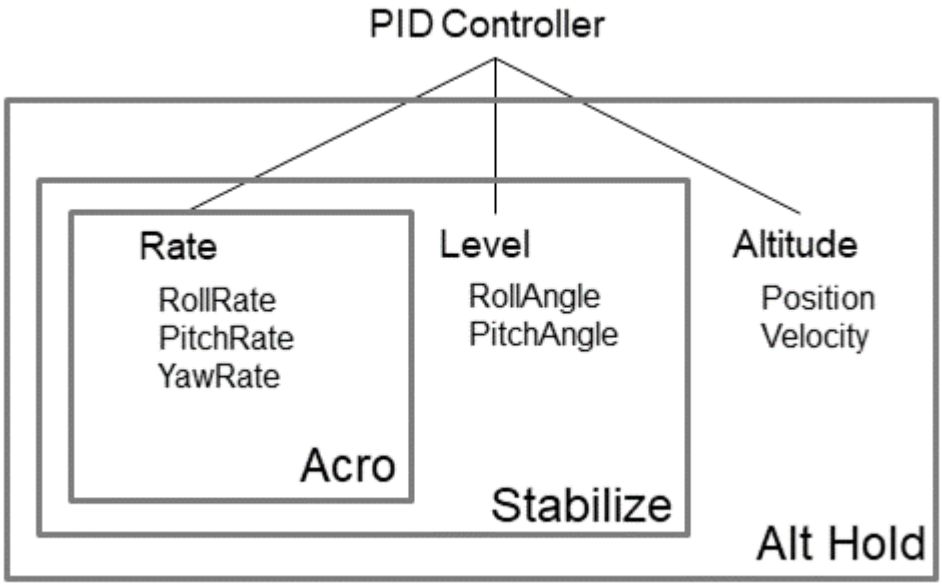


Figure 10. Hackflight PID Controllers (credit)

In addition to controlling drones, Hackflight interfaces with MulticopterSim, written by the same author. MulticopterSim is a simple multicopter flight simulator using Unreal Engine 4. It runs on Windows and is available on GitHub at *https://github.com/simondlevy/MulticopterSim*.

Hackflight is being actively developed, is published under the MIT license and the source code is available at the GitHub repository *https://github.com/simondlevy/Hackflight*.

7. Conclusion

Which flight controller firmware should you use? Well it depends on what you want to do. Assuming you want to use something that is being actively maintained we would suggest the following:

1. Cinematic Video — ArduPilot, PX4, iNav

2. FPV, Acrobatics and Racing — Betaflight, Emuflight, dRonin

5. Indoor and Micro Drones — Betaflight, Emuflight, Hackflight

We have left Cleanflight off the list because it is currently the same as Betaflight, but this could change. The above is obviously not a prescriptive list and you could, for example, use ArduPilot or PX4 for FPV if you want.

The other constraint you may have is hardware platform. Most flight controllers use the STM32 family of MCUs. Hackflight is unusual in that it is a recent firmware development focussing on some of the faster new Arduino boards.

One thing is certain, the software and hardware will continue to evolve and there will be an open source community out there, ready to take advantage of this.

[1] https://forum.librepilot.org/index.php?topic=6.0