Microprocessors and Microsystems 61 (2018) 11–20

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

A survey of Open-Source UAV flight controllers and flight simulators

Emad Ebeid*, Martin Skriver, Kristian Husum Terkildsen, Kjeld Jensen, Ulrik Pagh Schultz

SDU UAS Center, University of Southern Denmark, Campusvej 55, Odense, Denmark

ARTICLE INFO

Keywords:
Unmanned Aerial Vehicle (UAV)
Drones
Flight controllers
Drone simulators
Open platforms
Survey

ABSTRACT

The current disruptive innovation in civilian drone (UAV) applications has led to an increased need for research and development in UAV technology. The key challenges currently being addressed are related to UAV platform properties such as functionality, reliability, fault tolerance, and endurance, which are all tightly linked to the UAV flight controller hardware and software. The lack of standardization of flight controller architectures and the use of proprietary closed-source flight controllers on many UAV platforms, however, complicates this work: solutions developed for one flight controller may be difficult to port to another without substantial extra development and testing. Using open-source flight controllers mitigates some of these challenges and enables other researchers to validate and build upon existing research.

This paper presents a survey of the publicly available open-source drone platform elements that can be used for research and development. The survey covers open-source hardware, software, and simulation drone platforms and compares their main features.

1. Introduction

The current disruptive innovation in civilian drone (Unmanned Aerial Vehicle, UAV) applications and the associated growth in the drone industry has led to an increased need for research and development in UAV technology. While applications such as aerial filming, mapping, and inspection can be accomplished within the current legal framework and using technology already available on the market, the industry is pushing to break down the barriers which will enable entirely new markets in logistics, e-commerce, surveillance etc. Examples of these barriers are Beyond Visual Line of Sight (BVLOS) flights, which by many stakeholders is considered to be the next game-changer in the UAV business, autonomous operations not supervised by a pilot, and long range flights using small UAVs. Industry and academia are collaborating on solving the associated challenges, which are related to UAV platform properties such as functionality, reliability, fault tolerance, and endurance, which are all tightly linked to the UAV flight controller hardware and software. The lack of standardization of flight controller architectures and the use of proprietary closed-source flight controllers on many UAV platforms, however, complicates this work, as solutions developed for one flight controller may be difficult to port to another without substantial extra development and testing if at all legally possible. Using open-source flight controllers mitigates some of these challenges and enables other researchers to validate and build upon existing research.

In this work, we present a survey on open-source flight controllers designed for UAVs. The survey is based on 20+ flight controllers publicly available on the web. We relate flight controller hardware (Section 2) to flight controller software (Section 3) and compare features, specifications, license types etc. of each controller and software. Simulation is often used for software functionality experiments tests before performing actual flight tests, especially in flight controller research and development which can minimize the risk of loss of control during test flights and save time. We include a survey of open-source simulation systems (Section 4), some of which support hardware-in-the-loop or software-in-the-loop simulation. Currently, UAVs are under constant and rapid technological development, and the reader should note that the information described in this paper is as of February 2018.

1.1. Related work

In 2010 Cao et al. presented a survey of autopilot systems for small or micro UAVs systems with the objective of providing a summary of the available commercial, open-source and research autopilot systems in the market at that moment for potential users of small UAV [1]. In 2011 Mészarós surveyed open-source hardware and software for fixed-wing UAV platforms, including a design of a small fixed-wing UAV based on one of the presented systems with its first field test [2]. In 2012 Lim et al. surveyed publicly available open-source projects for quadrotor UAV systems, presenting eight quadrotors with descriptions of their avionics, sensor composition, analysis of attitude estimation and control algorithms, and comparison of their additional features [3].

* Corresponding author.
E-mail address: esme@mmmi.sdu.dk (E. Ebeid).

https://doi.org/10.1016/j.micpro.2018.05.002
Received 20 February 2018; Accepted 7 May 2018
Available online 14 May 2018

In 2016 Sabikan and Nawawi presented a general view of the implementation of an open-source quadcopter platform to develop a quadcopter research testbed [4], with the aim of making the proposed platform expendable and usable in any application area; they briefly mention a couple of open-source flight controller platforms, of which one of them has been used in their work

1.2. Unmanned Aerial Vehicles

For an introduction to the basics of UAV systems, we refer the interested reader to the textbook by Fahlstrom and Gleason [5]. We, however, note that compared to control systems for other advanced autonomous systems such as robotics, the control of UAVs presents two unique challenges. First, UAVs have severe constraints in terms of Size, Weight, and Power (SWaP) [6] not found to the same extent in robotics in general. Second, UAV control systems are in practice almost always safety-critical systems due to the primary purpose of the UAV being operation in open-ended environments where humans may be present [7].

In addition, each UAV platform comes with a license for governing the use or redistribution. We refer to the reader to the article by Christopher Vendome [8] for its coverage of the software license (e.g., Berkeley Software Distribution (BSD), The GNU General Public License (GPL), Massachusetts Institute of Technology (MIT), etc.).

1.3. Methodology

The information presented in this paper is to a large extent based on publicly available information from the web pages of the corresponding open-source projects developing the software. This includes manual inspection of schematics and source code by the authors in many cases. This approach, however, implies that missing information about the presence of a specific feature in a specific artifact is a threat to our validity. Manually experimenting with all of the documented projects was deemed untraceable.

2. Open-Source Hardware platforms

This section explores different Open-Source Hardware (OSH) platforms and summarizes their features. For OSH platforms, the hardware design such as mechanical drawings, schematics, bills of materials, Printed Circuit Board (PCB) layouts, Hardware Description Language (HDL) source code, and integrated circuit layout data are all released under free/libre terms [9].

2.1. Overview

Table 1 shows a comparison of the OSH flight controller platforms in terms of the MCU, built-in sensors, license, and interfaces; Table 2 compares the platforms in terms of dimensions, weight, power consumption, and fault tolerance features. With the exception of APM, all platforms use 32-bit MCUs based on the ARM architecture (ignoring discontinued platforms). All platforms have IMUs and support UART, PWM, and I2C interfaces. There is significant variation in the interfaces supported and the physical dimensions. Only a few platforms from the Pixhawk series of platforms support fault tolerance features.

2.2. Pixhawk series

The Pixhawk flight controller started at the Computer Vision and Geometry Lab of ETH Zurich, before becoming an independent OSH platform project [21]. Pixhawk collaborates with several partners, including the Linux Foundation DroneCode project [22]. It is based on the PX4-Flight Management Unit (FMU) and multiple versions have been developed.

Table 1
Comparison of MCUs, sensors and licenses for OSH flight controller platforms. Sensors: all platforms have IMUs. Interfaces: all platforms support UART, PWM, and I2C.

Platform	MCU	Sensors	License	Interfaces
Pixhawk [10]	STM32F427	b, m	BSD	c, s, a, pp, sb, ds
Pixhawk 2 [11]	STM32F427	b, m	CC-BYSA-3.0	c, s, a, pp, sb, ds
PixRacer [12]	STM32F427	b, m	CC-BY 4.0	c, pp, sb, ds
Pixhawk 3 Pro [13]	STM32F427	b, m	CC BY 4.0	c, s, pp, sb, ds
PX4 FMUv5 and v6 [14]	STM32F427	b, m	CC BY 4.0	c, s, a, pp, sb, ds
Sparky2 [15]	STM32F405	b, m	CC BY-NC-SA 4.0	c, pp, sb, ds, da
Chimera [16]	STM32F767	b, m, p	GPLv2	c, s, a, da, pp, sb, ds, x, au
CC3D [17]	STM32F103	None	GPLv3	pp, ds, sb
Atom [17]	STM32F103	None	GPLv3	pp, ds, sb
APM 2.8 [18]	ATmega2560	b	GPLv3	pp, a
FlyMaple [19]	STM32F103	b, m	GPLv3	None
Erle-Brain 3 [20]	Raspberry Pi	b, m	CC BY-NC-SA	a
PXFmini [20]	Raspberry Pi	b, m	CC BY-NC-SA	a
AeroQuad[d]	STM32F407	b, m	GPLv2	-
Mikrokopter[d]	ATmega644	b	-	s, pp
MatrixPilot[d]	dsPIC33FJ256	None	GPLv3	None

b: barometer, m: magnetometer, p: pitot tube sensor, c: CAN, s: SPI, a: ADC, pp: PPM, sb: S.BUS, ds: DSM, da: DAC, x: XBEE, au: AUX, [d]: discontinued.

Table 2
Comparison between OSH flight controller platforms.

Platform	Dimensions (mm)	Weight (g)	Fault tolerance features
Pixhawk [10]	81.5 × 50 × 15.5	38	c, IMU
Pixhawk 2 [11]	38.5 × 38.5 × 23	39	trp
PixRacer [12]	36 × 36*	10.9	c, trp, tri
Pixhawk 3 Pro [13]	71 × 49 × 23	45	i
PX4 FMUv5 and v6 [14]	71 × 49 × 23	45	–
Sparky2 [15]	36 × 36*	13.5	None
Chimera [16]	89 × 60*	–	tsp
CC3D [17]	36 × 36*	8	None
Atom [17]	15 × 7*	4	None
APM 2.8 [18]	70.5 × 45 × 13.5	31	None
FlyMaple [19]	50 × 50 × 12	15	None
Erle-Brain 3 [20]	95 × 70 × 23.8	100	None
PXFmini [20]	31 × 73*	15	None
AeroQuad [d]	–	–	None
Mikrokopter [d]	–	–	None
MatrixPilot [d]	–	–	None

*: Board without an enclosure. Height is the height of the PCB components.
c: co-processor, trp: triple-redundant power supply, tri: triple-redundant IMU, i: IMU, tsp: twin switching power supply, [d]: discontinued.

2.2.1. Pixhawk autopilot

The original Pixhawk combines the PX4 FMUv2 with the PX4 IOv2 IO board. It runs on an STM32F427 processor with an STM32F103 failsafe co-processor and has 256 KB RAM. A binary-compatible derivative called Pixfalcon has also been developed. It is a more lightweight, FPV optimized version with less IOs. Fig. 1a shows the Pixhawk along with its derivative shown in Fig. 1b.

2.2.2. Pixhawk 2 autopilot

The Pixhawk 2 was originally designed for the quadcopter 3DR Solo, while a standalone version was later released. It is built on the PX4 FMUv3 and designed by the Pixhawk community in collaboration with 3D Robotics. The form factor has changed into a cube, which features triple redundant IMUs, two barometers and the ability to connect up to three GNSS modules. The cube connects to a carrier board

(a) PIXHAWK (b) Pixfalcon (c) Pixhawk 2

(d) Pixhawk mini (e) PixRacer (f) Pixhawk 3 Pro

Fig. 1. Autopilot flight controllers.

through a single DF17 connector. The standard carrier board provides several IOs, similar to the original Pixhawk mentioned above. In addition, a version of the carrier board is available which has an interface for the Intel Edison, which is meant to run as a companion computer. Other companion computers can be connected through the IOs. A derivative of the Pixhawk 2, called Pixhawk Mini has been made, which is an evolution of the Pixfalcon. Fig. 1c shows the Pixhawk 2 and the Pixhawk Mini is seen on Fig. 1d.

2.2.3. PixRacer autopilot and Pixhawk 3 Pro

The PixRacer is built on the PX4 FMUv4 and is designed for racing UAVs. It has built-in WiFi, which enables wireless firmware updating, system setup, and telemetry. The PixRacer was evolved into the Pixhawk 3 Pro, which has received an upgrade of processors and sensors for better performance. Fig. 1e shows the PixRacer and the Pixhawk 3 Pro-is shown in Fig. 1f.

2.2.4. PX4 FMUv5 and v6

With more than 390 contributors, the PX4 hardware has evolved to FMUv5 at the end of 2017. For PX4 FMUv5, the processor is updated to STM32F7 with double-precision floating point unit (FPU) and higher accuracy attitude calculations based on the Bosch BMI055 IMU. In addition, Dronecode has announced that the PX4 FMUv6 will be released in 2018 with an upgrade of the main processor to the STM32H7.

2.3. Sparky2

The Sparky2 is an OSH platform from TauLabs [23]. It is based on the STM32F405 Microcontroller Unit (MCU) and published under the CC BY-NC-SA 4.0 license. Fig. 2 shows the Sparky2 flight controller.

2.4. Paparazzi

Paparazzi is an OSH UAV platform which was founded in 2003. Paparazzi is developed at Ecole Nationale de l'Aviation Civil (ENAC) UAV Lab [24]. It encompasses autopilot systems and ground station

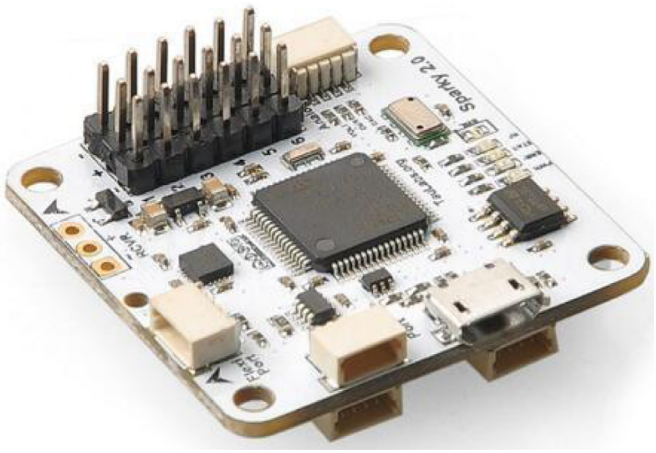


Fig. 2. The Sparky2 flight controller from TauLabs.

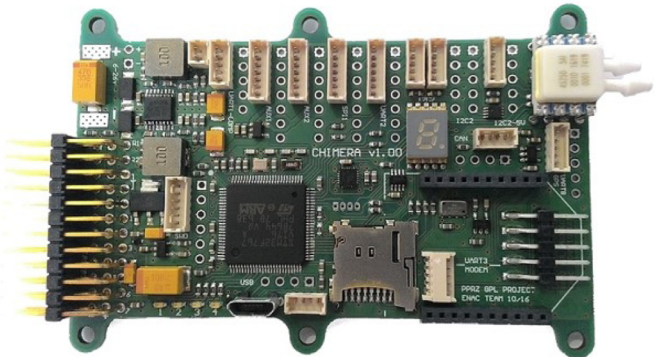
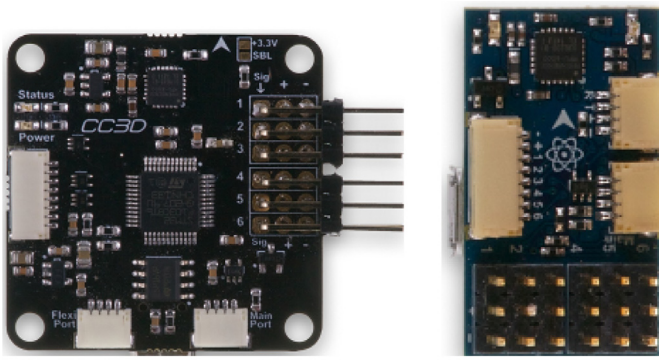


Fig. 3. The STM-based Paparazzi Chimera flight controller.

software for multi-rotors, fixed-wing, helicopters and hybrid aircraft [25]. In March 2017, ENAC Lab released a new autopilot named *Chimera* which is based on the latest STM32F7 MCU. Fig. 3 shows the Paparazzi Chimera circuit board.

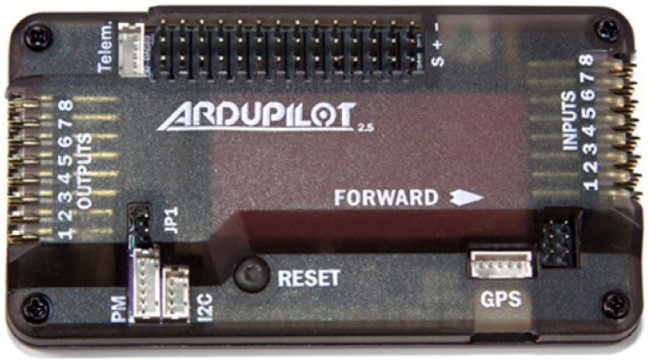
2.5. CC3D & Atom

The CC3D and Atom flight controllers have the same functionality, but differ in form factor. They are developed by LibrePilot, previously known as OpenPilot. They run the LibrePilot firmware, which is further discussed in Section 3.3.1. The flight controllers support multiple airframes, and their hardware is available under the GPLv3 license. Fig. 4 shows the CC3D and Atom flight controllers.



(a) (b)

Fig. 4. Flight controller boards: (a) CC3D (b) Atom.



(a)

Fig. 5. ArduPilot Mega (APM) 2.8 autopilot.

2.6. ArduPilot Mega (APM)

The APM is an Arduino Mega-based autopilot, which has been developed by the DIY Drones community. APM runs the ArduPilot software, which is described in Section 3.4 and supports multiple vehicles. Fig. 5 shows the ArduPilot Mega (APM) v2.8.

2.7. FlyMaple

FlyMaple is a flight controller which is based on the Maple project. FlyMaple is based on an Arduino style APM processor. Fig. 6 shows Flymaple flight controller board.

2.8. Erle-Brain 3

The Erle-Brain is a Linux-based flight controller developed by Erle Robotics located in Spain. Erle-Brain combines an embedded Linux board (a Raspberry Pi), with a PXFmini board, which contains sensors, IO and power electronics. The PXFmini is an open hardware shield designed for the Raspberry Pi. The Erle-Brain builds on the DroneCode foundation (see Section 3.5). Fig. 7 shows the Erle-Brain 3 autopilot and its accessory parts.

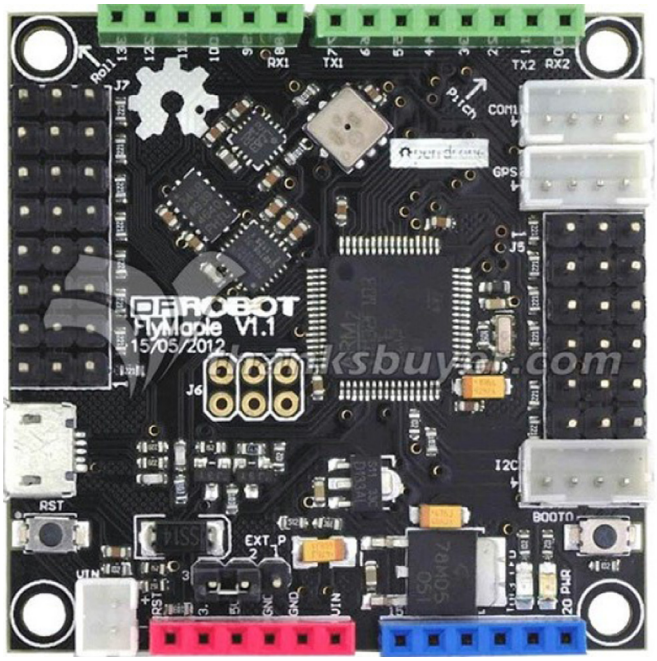


Fig. 6. FlyMaple flight controller board.

(a) (b)

Fig. 7. Erle-Brain autopilot: (a) Erle-Brain 3 (b) PXFmini on top of a Raspberry Pi.

2.9. Inactive OSH projects

AeroQuad was an Arduino-based platform, which was discontinued in 2015. However, the software is still available [26]. Mikrokopter v2.5 is based on the ATmega1284P-AU MCU. The schematics are availabe at [27]. The v3.0 is focusing on redundant systems. However, at the time of writing, the schematics are not open-source. MatrixPilot was designed for fixed-wing UAV and controlled by the UAV Development Board v3 from Sparkfun. The board has since been discontinued.

3. Open-Source Software platforms

This section explores selected Open-Source Software flight controllers. We observe that a significant number of projects have been discontinued (detailed in Section 3.7), but that in general the codebase of discontinued projects has been forked and is used in active projects.

3.1. Overview

Tables 3 and 4 show a comparison of OSS flight controller platforms in terms of key properties such as the kind of airframes supported, support for autonomous flight, availability of links for C2 and telemetry, fault detection and handling, licensing, Source Lines Of Code (SLOC), programming language, and operating system. We note to the reader that the projects no longer in active development are not included. All active platforms support multirotor airframes, similarly, fixed-wing airframes are supported by all but one (Hack flight). Autonomous flight is supported by most platforms, as are C2 and telemetry links. Fault detection and handling features are present on most platforms, with PX4 as a notable exception. There is a strong preference for GPL-based licenses, with PX4 as a notable exception. The PX4 use of the BSD 3-Clause license is a significant advantage for its use in commercial systems.

3.2. Multiwii series

The first flight control software [34] of the Multiwii series (Fig. 8) was developed by Alexandre Dupus for low-cost hardware. The initial goal was to control a tricopter with an Arduino Pro-Mini using Nintendo Wii motion sensors [35]. The original implementation language was Arduino C but later changed to C++; comparing its 12.5k SLOC with the platform in Table 4, we observe that it is a relatively small software package. The development has stopped but the source code is still available along with a Java-based configuration tool [34]. Baseflight [36] changed the Multiwii implementation language to C, evolved to processor support from 8-bit to 32-bit, and made an internet-browser-based configuration tool [37,38]. Raceflight [39] also has its roots in the Multiwii flight controller and have been optimized for UAV racing and acrobatics, but development has also stopped.

Table 3
Collection of but not limited to, supported features for OSS flight controller platforms.

Platform	Airframes	Autonomous Flight	Communication	Fault detection and handling	Features
Hack flight	mr(4)	–	pm,sb,ds,msp	–	–
Cleanflight	mr(1–8)/fw	sph,swp	sb,ib,sd,sh,pm,pw,cf,jeb,ds,xbu,fs,sp,ht,lt,ml,sl,msp	c2k,bvw	g,bm,dl,rth,hf,ts,at
Betaflight	mr(1–8)/fw	sph,swp	sb,ib,sd,sh,pm,pw,cf,jeb,ds,xbu,fs,sp,ht,lt,ml,sl,msp	c2l,c2k,bvw	g,bm,dl,rth,hf,ts,at
INAV	mr(1–8)/fw	fm,swp,sph	sb,ib,sd,sh,pm,pw,cf,jeb,ds,xbu,fs,sp,ht,lt,ml,msp	c2l,c2k,bvw,c2rl	g,at,bm,dl,rth,at
LibrePilot****	mr(3–8)/fw	sph, swp, atl	sb,ib,sd,sh,pm,pw,xbu,fs,ht,ml,sl,msp	c2k	g,rth,bac,bm
dRonin****	mr(*)/fw	sph	sb,fs,ht,ml,ut,msp	pa	bm,dl,at,rth
ArduPilot	mr(1–8)/fw	svnf,sph,swp	sb,pm,ds,fs,ml	ar,sw,apr,sc,c2rl,c2l,c2k,bll,gf	g,rlg,loa,bm,rth,***
PX4	mr(1–8)/fw	sph,swp,fm	sb,sd,ds,fs,sp,ht,ml,ir	gf,c2k,c2l,c2rl,sc,apr	g,rth,bm,dl,rlg,loa,**
Paparazzi	mr(+)/fw	sph,swp,atl,fw,svnf	pm,sb,xbe,ml	ar,sw,apr,sc,c2rl,c2l,c2k,bll,gf	g,pa,rth,bm,dl,hf,ts,at,rlg,loa,bac

*: Not found in documentation; **: See parameters for more features [28]; ***: See parameters for more [29]; ****: Low source reliability; +: Limited by hardware; Airframes mr: multirotor (number of propellers); fw: fixed wings; Autonomous Flight sph: Satelite position hold; swp: Satelite way point navigation; atl: automatic take off and landing; fm:: Follow me; svnf: Stereo vision navigation functions; Communication sb: SBus; ib: iBus; sd: SumD; sh: SumH; pm: PPM; pw: PWM; cf: CRSF; jeb: JetiExBus; ds: DSM; xbu: XBUS; xbe: XBee; fs: FrSky; sp: SmartPort (s.port); ht: HoTT; lt: LTM; ml: MavLink; sl: SRXL; ut: UAVTalk; msp: Multiwii serial protocol; ir: Iridium SBD; Fault detection and handling ar: ADS-B receiver support; c2l: Landing when missing C2 link; c2rl: Return to launch or home after missing C2 link; c2k: Kill supply for motors after missing C2 link; bvw: Battery voltage warning; ada: Auto disarm after timeout; apr: Automatic parachute release; sw: Safety switch to en-/disable output to motors; bll: Land or return to launch if battery low; gf: Geofence; sc: Pre-Arm Safety Check for sensor error; Features g: gimbal compatible; pa: Prevent arming when battery low; rth: Return to home; bm: Battery monitoring; dl: Data logging to flash; hf: Head free moving (yaw fixed); ts: Transponder support; at: PID autotune; rlg: Supports retractable landing gear; loa: Lidar obstacle avoidance; bac: Battery mAh count;

Table 4
Comparison of source code and availability for OSS flight controller platforms.

Platform	License **FC/doc/ conf. tool**	SLOC+	Language	OS
Hack flight	g/-/lg	3.3k	*C++	None
Cleanflight	g/-/g	96.3k	C	Scheduler
Betaflight	g/-/g	107k	C	Scheduler
INAV	g/-/g	93.1k	C	Scheduler
LibrePilot	g/cbs/g	**87.6k	C	FreeRTOS [30]
dRonin	g/tbd[31]/	**97.6k	C	***PiOS
ArduPilot	g/cbs/g	****242.1k	C/C++	ChibiOS [32]/NuttX [33]/Linux
PX4	b/CC BY 4.0/g	****211.8	C/C++	NuttX [33]
Paparazzi	g/gf/g	2106k(C)	C/Python	ChibiOS [32]/ Scheduler

*: C++ header only code, **:flight dir without PiOS; ***:PiOS Pilot Operating System; ****:ArduPilot repository(cpp only) without Ardusub and APMrover; +:Source Lines Of Code. License b: BSD; lg: GPLv3; lg: Lesser GPLv3; cbs: CC-BY-SA 3.0; tbd: To be determined; gf: GNU Free Documentation License (GFDL);

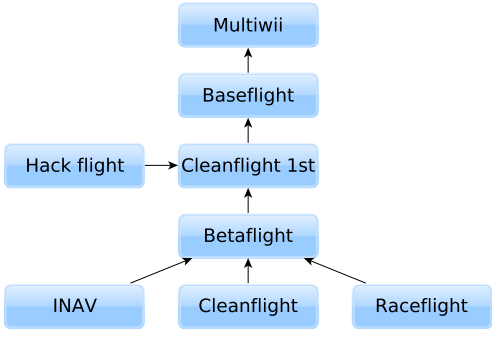


Fig. 8. Development history originating from the original Multiwii flight controller software.

3.2.1. *Hack flight*

Hack flight is derived from Baseflight, but the source [40] seems completely rewritten. It supports the STM32L4 based Ladybug flight controller and it has a wiki [41] that contains information about how to setup the controller with receivers and motors. In addition to the flight code, the repository consists of a Python-based Ground Control Station (GCS) for reading sensor values and an Android application for showing altitude. Based on statements in the README file, this platform seems suitable for smaller projects where people want hands-on development without too much complexity.

3.2.2. *Cleanflight*

The first version of the Cleanflight [42] source code [43] was forked from Baseflight, but since the functionality has been inherited from INAV and Betaflight or developed in Cleanflight. Cleanflight has documented setup guides for elements such as sensors, flight setup, and for 15 different flight controller boards, all with ARM-based STM32 F1, F3, F4 or F7 MCUs, although the F1 series will not be supported in future releases. The software contains a browser-based configuration tool [44] from where the UAV can be configured, sensors can be read, and parameter tuning performed. Cleanflight has its origin in Baseflight and the configuration tool can be added to Google Chrome [45].

3.2.3. *Betaflight*

Betaflight source code [46] and documentation are related to INAV (Section 3.2.4) and Cleanflight (Section 3.2.2) [47]. This is visible if comparing the documents and the structure of their repositories, but Betaflight differs according to the readme file, by developing with focus on creating high performance, leading-edge feature additions, and wide target support [46]. Cleanflight is suggesting this platform for people who want to work with experimental new features [43]. The documentation section describes 17 flight controller boards among those the CC3D (Section 2.5). The configuration tool is like for Cleanflight, available for Google Chrome add [48] and the source code is available [49].

3.2.4. *INAV*

INAV [50] is forked from an earlier version of Cleanflight (Section 3.2.2). INAV has improved navigation functionalities [51] compared to Betaflight and Cleanflight with “follow me” functionality and support for Ground Control Station (GCS) with mission planner running on Windows, Linux, iOS, and Android. The documentation directory describes setup for 25 flight controller boards. The source repository [52] has a Google Chrome browser add-on configuration tool [53] similar to Betaflight and Cleanflight.

3.3. *OpenPilot series*

OpenPilot is a flight controller software package that has been developed through many projects and by different developers, as illustrated in Fig. 9. OpenPilot was the original flight controller software in this series, but development was stopped in 2015. Parts of its wiki [54] is still functional and the source code [55] is still available. The flight

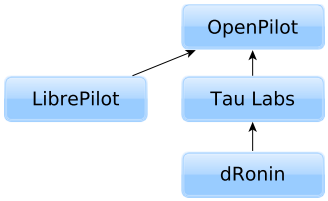


Fig. 9. Development history from the original OpenPilot flight controller software.

control software supports among others the CC3D (Section 2.5) board and it can control fixed-wing and multirotor UAVs. Similarly, Tau Labs [56] is an inactive flight controller platform from this series where the source code remains available [57]. The individual projects that still are in active development are described in this section.

3.3.1. LibrePilot

The LibrePilot [58] project started in July 2015 with focus on research and development of software and hardware to be used in a variety of applications, including vehicle control and stabilization of unmanned autonomous vehicles and robotics [58]. The source code [59] builds on the OpenPilot project and runs on various platforms such as Sparky2 (Section 2.3) and CC3D (Section 2.5). There is online documentation [60] for setting up the software for the supported boards, UAV configuration, and using sensors and the Ground Control Station (GCS).

3.3.2. dRonin

dRonin [61] forked their source code [62] from the Tau Labs repository. dRonin has online documentation [31], an online forum [63], a wiki [64], and additional documentation as repository files. The code can be used with several flight controller boards, including Sparky2 (Section 2.3), AeroQuad, and Raspberry Pi with FlyingPIO sensor extension [65]. dRonin can be linked to an Android application [66] for sensor readings and parameter tuning.

3.4. ArduPilot

ArduPilot [67] is capable of controlling a wide range of vehicle including fixed-wing airplanes, multirotors, helicopters, boats, and submarines [68]. The software was originally developed for 8-bit ARM-based MCUs running on its own board *ArduPilot*. This board was however replaced by ArduPilot Mega (APM) (Section 2.6) and the software has evolved to be optimized for use with 32-bit ARM-based MCUs. Nevertheless, the controller can run under Linux, enabling it to be used on a large class of electronics from single-board computers all the way up to a full PC system. ArduPilot has a desktop Ground Control Station (GCS) for mission planning [69,70], calibration, and vehicle setup for Windows, Linux and Mac OSX. Different setups of the AutoPilot system and installation manuals are available from the ArduPilot web page [71].

3.5. PX4

The PX4 flight stack and autopilot [72], are a part of the DroneCode collaborative project [22] which covers both Ground Control Station (GCS), hardware platforms and simulation [73]. PX4 works with many different airframes for using multirotor, fixed-wing, gliders, copters and VTOL technology. PX4 is compatible with the QGroundControl GCSs [74], from where parameters are set, sensors read, and autonomous flight configured. Performance of the flight stack is presented by Meier et al.[75]. The use of the BSD license and the high level of completeness makes this flight stack attractive for commercial companies [76]. The documentation is online [77] for developers and users and covers both flight controller and Ground Control Station (GCS) setup.

3.6. Paparazzi software

The Paparazzi project works on a complete UAV system including electronics (Section 2.4), flight controller software [78], autopilot, Ground Control Station (GCS) and simulation (Section 4.6) [16]. The Paparazzi flight controller can be configured to control fixed-wing, flapping-wing, hybrid and multirotor airframes in different configurations, as well as multiUAV systems [79]. From the Paparazzi Ground Control Station (GCS), it is possible to setup parameters, flight type and configuration, read sensor, data setup a flight plan including custom guidance, navigation and cohntrol algorithms [80]. Paparazzi has a source code documentation [81] for developers as well as a wiki-based guide [82] for setting up the flight controller.

3.7. Inactive OSS projects

AutoQuad [83] is a product series that includes UAV components like Electronic Speed Controller (ESC)s which are based on open-source hardware and flight controllers based on open-source software [84]. The flight controller has been developed through product generations for more than 6 years. The firmware is written for the STM32F4 series MCUs and supports up to 14 Brushless DC motor (BLDC)s, and is compatible with QGroundControl. The source code is still available but future development seems to be on hold or stopped.

JAviator [85] was a research project of the Computational Systems Group at the Department of Computer Sciences of the University of Salzburg, Austria [86,87]. The project was running from 2006 to 2013 and the project delivered three layers of software; JAviator Plant (JAP), Flight Control System (FCS), and Ground Control Station (GCS). The FCS software is written in C, runs on a Robostix-Gumstix stack, and implements the flight control algorithms. However, this project does not seem to be active anymore.

The AeroQuad [26] flight controller software is written in Arduino/C [26] and supports Atmel powered boards, among those their own developed AeroQuad (Section 2.9). The flight controller has a desktop configuration tool for setting up parameters. AeroQuad supports multirotors UAVs, but there have not been updates to the source code since 2013.

As described earlier, the MultiWii, Baseflight, RaceFlight, OpenPilot, and TauLabs are also among the OSS projects where development has stopped.

4. Open-Source Simulation platforms

This section explores selected Open-Source Simulator platforms and summarizes their features.

4.1. Overview

Table 5 shows a comparison of selected Open-Source Simulator (OSSIM) UAV platforms, including the implementation language, the supported operating system, and the licensing terms. C and C++ are the most popular implementation languages, with Python and Java as notable exceptions for Morse and jMAVSim. All projects support Linux, some additionally support MacOS and/or Windows.

4.2. AirSim

Aerial Informatics and Robotics Platform (AirSim) was developed by Microsoft in 2017 [94]. AirSim aims to support the development and testing of algorithms for autonomous vehicle applications, such as deep learning, computer vision, and reinforcement learning algorithms. The AirSim physics engine is based on Unreal Engine 4 (UE4) [95] and can operate at a high frequency for real-time Hardware-in-the-Loop (HIL) simulations with support for popular protocols (e.g. MavLink). The simulator follows modular design architecture with an emphasis on

Table 5
Comparison between OSSIM drone platforms.

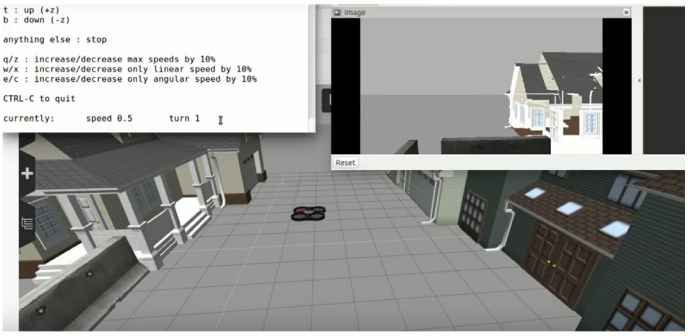
Platform	Implementation Language	Supported Operating System	License
AirSim [88]	C++	Windows, Linux	MIT
Gazebo [89]	C++	Linux, MacOS	Apache V2.0
Morse [90]	Python	Linux	BSD
jMAVSim [91]	Java	Linux, MacOS, Windows	BSD 3
New Paparazzi Simulator [92]	C	Linux, MacOS	GPLv2
HackflightSim [93]	C++	Windows, Linux	GPL

extensibility. The only supported UAVs are Iris in a MultiRotor model and in an X-configuration for PX4 quadrotor. Fig. 10a shows a screenshot of AirSim.

4.3. Gazebo

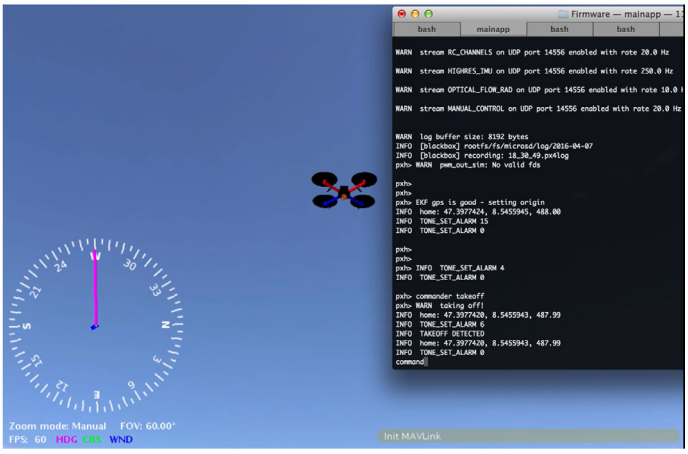
Gazebo [96] was originally developed at the University of Southern California, the USA in 2002 and later at Open Source Robotics Foundation (OSRF). Gazebo is the default simulator included with Robot Operating System (ROS), making it one of the most popular 3D dynamics multi-robot simulators with a very active community [97]. Gazebo allows the use of different physics engines and sensor models, and supports easy creation of 3D worlds, enabling testing of robot

(a) AirSim

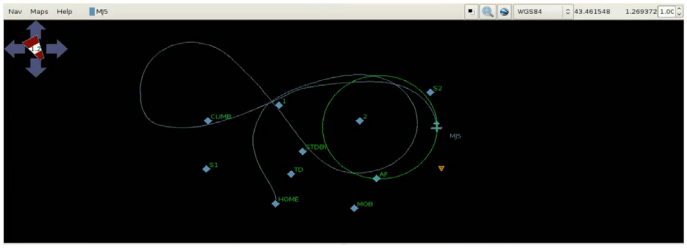


(b) Gazebo

(c) Morse



(d) jMAVSim



(e) New Paparazzi Simulator

(f) HackflightSim

Fig. 10. Simulator screenshots.

designs and algorithms, regression testing, and training of AI systems using realistic scenarios. Gazebo uses a distributed architecture with separate libraries for physics simulation, rendering, user interface, communication, and sensor generation. In order to simulate UAVs in Gazebo, J. Meyer et al. presented a framework to simulate a quadrotor using ROS. The framework provides a Gazebo ROS package named Hector Quadrotor [98] that aims at simulating diverse quadrotor UAV aspects such as flight dynamics, onboard sensors like IMUs, external imaging sensors and complex environments. Supported by UAVs include Quad (Iris and Solo), Hex (Typhoon H480), Generic quad delta VTOL, Tailsitter, Plane, and Rover. Fig. 10b shows a snapshot of a quadrotor UAV outdoor simulation in Gazebo. Although Gazebo is a feature-rich platform, the rendering techniques are not as advanced as, e.g., Unreal Engine [99] or Unity [100]. In addition, the Hector package lacks support for protocols such as MavLink and Hardware platforms such as Pixhawk (Section 2.2).

4.4. Morse

Modular Open Robots Simulation Engine (MORSE) is a generic simulator for research in robotics that is developed jointly at Laboratory for Analysis and Architecture of Systems (LAAS) and Office National d'Etudes et de Recherches Arospatiales (ONERA) [101] since 2011. MORSE relies on Blender [102] for physics simulation and for a realistic display of the simulated environment. MORSE is designed as a general purpose, modular system simulation of multiple moving robots in many different kinds of environments [101]. MORSE relies on a component-based architecture. Each MORSE component consists of a Blender and a Python file. The Blender file expresses the physical and visual properties of the object in the simulated world. The Python file defines an object class for the component type [101]. The MORSE component library provides `quadrotor_dynamic` which is a simple definition of a quadrotor with a rigid body. Fig. 10c shows a screenshot of a quadrotor UAV simulated in MORSE. MORSE lacks graphical user interface since it only provides a command line one. MORSE does not embed any advanced algorithms (e.g., path planning) since it expects to run such algorithms in the simulated software stack of the desired object.

4.5. jMAVSim

Java Micro Air Vehicle Simulator (jMAVSim) is a simple and lightweight multirotor simulator developed by the PIXHAWK engineering team (Section 2.2) [75,103]. jMAVSim supports the MAVLink protocol, uses the Java3D library for visualization, and connects directly to the HIL via a serial connection or to Software-in-the-Loop (SIL) via User Datagram Protocol (UDP) communication to the autopilot. Fig. 10d shows a screenshot of a quadrotor UAV application in jMAVSim. The implementation of jMAVSim is as a single component based on a relatively simple object-oriented design with no explicitly documented architecture [104].

4.6. New Paparazzi Simulator

New Paparazzi Simulator (NPS) is an advanced simulator with sensor models developed at Ecole Nationale de l'Aviation Civil (ENAC) UAV Lab (Section 2.4). The default flight model in NPS is JSBSim which supports a range of relatively complex airframes. NPS allows the use of different Flight Dynamic Model (FDM) backends, enabling developers to choose their own FDM model, for example connecting the simulator to an FDM expressed in MATLAB Simulink. Different visualization options are available, including FlightGear and the Morse (Section 4.4) simulator. Fig. 10e shows a screenshot of the simulator running using the Paparazzi GCS. Supported UAVs include rotorcraft and fixed-wing airframes.

4.7. HackflightSim

HackflightSim is a simple cross-platform quadcopter simulator developed by Simon D. Levy, Washington and Lee University Lexington, the USA in 2017. HackflightSim is implemented in C++, uses Unreal Engine 4, and is based on the Hackflight firmware which is a Simple C++ quadcopter flight control firmware for Arduino. The project is similar to AirSim (Section 4.2) however, it focuses only on quadcopter firmware. Fig. 10f shows a screenshot of a quadrotor UAV application in HackflightSim. The implementation of HackflightSim is as a single component that uses the Unreal Engine for visualization, similarly to jMAVSim (Section 4.5) the main component is based on a relatively simple object-oriented design with no explicitly documented architecture [105].

4.8. Inactive OSSIM projects

Python Quad Simulator (PyQuadSim) is an open-source quadrotor simulator in Python for Linux that is stopped in 2015.

5. Conclusions and future work

This paper has reviewed more than 20 different open-source hardware, software, and simulation UAV platforms appropriate for use in a wide range of settings, ranging from basic academic research to serve as the foundation for a commercial product. Overall we observe that open-source flight controller hardware is generally based on 32-bit ARM MCUs but otherwise varies significantly in terms of interfaces supported and physical dimensions. Support for fault tolerance features is in general limited, and there is little documentation regarding power consumption. For open-source flight controller software most projects use a GPL license, and although some projects have been discontinued, their codebase is generally still being used in other active projects. We note that similarly to the case for field robots [106], there is a lack of use of relevant industrial standards for safety-critical software in the development of open-source flight controller software, which is an issue for use in real-world scenarios. Open-source flight simulator software is generally implemented in C or C++ and in general supports Linux, with some use of languages such as Python and Java, and some support for MacOS and Windows.

Future work will be focusing on a deep analysis of the power consumption of a combined software and hardware platforms along with relative costs calculations of the hardware ones.

Acknowledgment

This work is supported and partly funded by the European Union's Horizon2020 research and innovation programme under grant agreement No. 779882 (TeamPlay) and the Innovationfund Denmark (IFD) Free the Drones (FreeD) project.

References

[1] H. Chao, Y. Cao, Y. Chen, Autopilots for small unmanned aerial vehicles: a survey, Int. J. Control Autom. Syst. 8 (1) (2010) 36–44, http://dx.doi.org/10.1007/s12555-010-0105-z. https://doi.org/10.1007/s12555-010-0105-z
[2] J. Mészarós, Aerial surveying UAV based on open-source hardware and software, Int. Arch. Photogram., Remote Sen. Spatial Inf. Sci. 37 (2011) 1.
[3] H. Lim, J. Park, D. Lee, H.J. Kim, Build your own quadrotor: open-source projects on unmanned aerial vehicles, IEEE Robot. Autom. Mag. 19 (3) (2012) 33–45.
[4] S. Sabikan, S.W. Nawawi, Akademia baru open-source project (OSPs) platform for outdoor quadcopter, Akademia Baru 24 (1) (2016) 13–27.
[5] P. Fahlstrom, T. Gleason, Introduction to UAV Systems, John Wiley & Sons, 2012.
[6] C. Howard, Uav command, control & communications, Military Aerospace Electron. Militaryaerospace. com (2013).
[7] I. Sadeghzadeh, Y. Zhang, A review on fault-tolerant control for unmanned aerial vehicles (UAVs), Infotech@ Aerospace 2011, (2011), p. 1472.
[8] C. Vendome, A large scale study of license usage on github, 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 2, (2015), pp.

772–774.
[9] D.M. German, M.D. Penta, J. Davies, Understanding and auditing the licensing of open source software distributions, 2010 IEEE 18th International Conference on Program Comprehension, (2010), pp. 84–93, http://dx.doi.org/10.1109/ICPC.2010.48.
[10] px4dev, Pixhawk web page, www.pixhawk.org.
[11] ProfiCNC, ProfiCNC, http://www.proficnc.com/.
[12] N. Arsov, P. Kocmoud, L. Meier, D. Sidrane, L. Hall, Pixracer autopilot, https://pixhawk.org/modules/pixracer.
[13] P.D. Team, Pixhawk 3 pro, https://docs.px4.io/en/flight_controller/pixhawk3_pro.html.
[14] pixhawk.org, Px4fmu autopilot / flight management unit, https://pixhawk.org/modules/px4fmu.
[15] T. Labs, Sparky2, https://github.com/TauLabs/TauLabs/wiki/Sparky2.
[16] P. development team, Paparazzi web page, https://wiki.paparazziuav.org/.
[17] L. community, CC3D web page, http://opwiki.readthedocs.io/en/latest/user_manual/cc3d/.
[18] A.M. team, ArduPilot Mega, www.ardupilot.co.uk.
[19] DFRobot, Flymaple — a flight controller with 10 DOF IMU, https://www.dfrobot.com/product-739.html.
[20] E.R.S. L., Erlerobotics web page, www.erlerobotics.com.
[21] L. Meier, P. Tanskanen, L. Heng, G.H. Lee, F. Fraundorfer, M. Pollefeys, Pixhawk: a micro aerial vehicle design for autonomous flight using onboard computer vision, Auton. Robots 33 (1) (2012) 21–39.
[22] DroneCode, DroneCode project, https://www.dronecode.org/.
[23] T. Labs, What is tau labs, http://taulabs.org/.
[24] P. Brisset, A. Drouin, M. Gorraz, P.-S. Huard, J. Tyler, The paparazzi solution, MAV 2006, 2nd US-European Competition and Workshop on Micro Air Vehicles, (2006).
[25] E. Baskaya, G. Manfredi, M. Bronz, D. Delahaye, Flexible open architecture for UASs integration into the airspace: Paparazzi autopilot system, 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), (2016), pp. 1–7, http://dx.doi.org/10.1109/DASC.2016.7778016.
[26] AeroQuad, AutoQuad Flight Software Platform, https://github.com/AeroQuad.
[27] H. GmbH, MikroKopter, http://wiki.mikrokopter.de/en/FlightCtrl.
[28] P.D. Team, PX4 setup parameters, https://docs.px4.io/en/advanced_config/parameter_reference.html/.
[29] ArduPilot, Ardupilot parameter list, http://ardupilot.org/copter/docs/parameters.html/.
[30] A.W.S. (AWS), FreeRTOS web page, https://www.freertos.org/.
[31] dRonin documentation, http://dronin.org/docs/.
[32] G.D. Sirio, ChibiOS wiki, http://www.chibios.org/dokuwiki/doku.php/.
[33] G. Nutt, NuttX web page, http://nuttx.org/.
[34] A. Dupus, Multiwii source code, https://code.google.com/p/multiwii/.
[35] A. Dupus, MultiWii wiki, http://www.multiwii.com/wiki/index.php?title=MultiWii/.
[36] timecop, Baseflight source code, https://github.com/multiwii/baseflight/.
[37] timecop, Baseflight configuration tool add for Google Chrome, https://chrome.google.com/webstore/detail/baseflight-configurator/mppkgnedeapfejgfimkdoninnofofigk?hl=en.
[38] timecop, Baseflight configuration tool source code, https://github.com/multiwii/baseflight-configurator/.
[39] RaceFlight, RaceFlight source code, https://github.com/rs2k/raceflight/.
[40] S. Levy, Hackflight: Simple quadcopter flight control firmware and simulator for c ++ hackers, https://github.com/simondlevy/hackflight.
[41] S. Levy, Hack flight wiki, https://github.com/simondlevy/Hackflight/wiki.
[42] C. team, Cleanflight web page, http://cleanflight.com/.
[43] D.C. (hydra), Cleanflight source code, https://github.com/cleanflight/cleanflight/.
[44] Cleanflight gui add source, https://github.com/cleanflight/cleanflight-configurator.
[45] Cleanflight gui add, https://chrome.google.com/webstore/detail/cleanflight-configurator/enacoimjcgeinfnnnpajinjgmkahmfgb.
[46] B.B. (borisbstyle), Betaflight source code, https://github.com/betaflight/betaflight/.
[47] B.B. (borisbstyle), Betaflight wiki, https://github.com/betaflight/betaflight/wiki/.
[48] B.B. (borisbstyle), Betaflight gui add source, https://chrome.google.com/webstore/detail/betaflight-configurator/kdaghagfopacdngbohiknlhcocjccjao/.
[49] B.B. (borisbstyle), Betaflight gui add source, https://github.com/betaflight/betaflight-configurator/.
[50] INAV source code, https://github.com/iNavFlight/inav/.
[51] INAV wiki, https://github.com/iNavFlight/inav/wiki/.
[52] INAV Chrome add source, https://github.com/iNavFlight/inav/.
[53] INAV Chrome add, https://chrome.google.com/webstore/detail/inav-configurator/fmaidjmgkdkpafmbnmigkpdnpdhopgel/.
[54] OpenPilot, OpenPilot wiki, https://opwiki.readthedocs.io/en/latest/.
[55] OpenPilot, OpenPilot source code, https://github.com/openpilot/OpenPilot/.
[56] T.L. developer team, Tau Labs web page, http://taulabs.org/.
[57] T.L. developer team, Tau Labs source code, https://github.com/TauLabs/TauLabs/.
[58] L. Team, LibrePilot Project, http://librepilot.org/.
[59] L. Team, LibrePilot source code, https://github.com/librepilot/LibrePilot/.
[60] L. Team, LibrePilot documentation, https://librepilot.atlassian.net/wiki/spaces/LPDOC/pages/2818105/Welcome/.
[61] dRonin executive committee, dRonin web page, http://dronin.org/.
[62] t. d. e. Volunteers and committee, dRonin source code, https://github.com/d-ronin/dRonin/.
[63] dRonin forum, https://forum.dronin.org/forum/.
[64] dRonin wiki, https://github.com/d-ronin/dRonin/wiki/.
[65] dRonin Raspberry Pi hat, https://github.com/d-ronin/dronin-hw/tree/master/flyingpio/revB/.
[66] dRonin android application, https://play.google.com/store/apps/details?id=org.dronin.androidgcs/.
[67] ArduPilot, Ardupilot source code, www.github.com/ArduPilot/.
[68] A.D. Team, ArduPilot, http://ardupilot.org/about/.
[69] ArduPilot, Ardupilot mission planner, http://ardupilot.org/planner/index.html/.
[70] ArduPilot, Ardupilot mission planner 2, http://ardupilot.org/planner2/index.html/.
[71] ArduPilot, Ardupilot documentation, http://ardupilot.org/ardupilot/.
[72] P.D. Team, PX4 source code, https://github.com/PX4/.
[73] P.D. Team, PX4 web page, http://px4.io/.
[74] QGroundControl, QGroundControl web page, http://qgroundcontrol.com/.
[75] L. Meier, D. Honegger, M. Pollefeys, Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms, 2015 IEEE International Conference on Robotics and Automation (ICRA), (2015), pp. 6235–6240, http://dx.doi.org/10.1109/ICRA.2015.7140074.
[76] PX4 commercial drone use, https://www.cnet.com/news/linux-foundation-fuels-open-source-drone-efforts/.
[77] P.D. Team, PX4 documentation, https://docs.px4.io/en/.
[78] P. development team, Paparazzi source code, https://github.com/paparazzi/.
[79] H.G. de Marina, Z. Sun, M. Bronz, G. Hattenberger, Circular formation control of fixed-wing UAVs with constant speeds, 2017 IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS) (2017) 5298–5303.
[80] H.G. de Marina, Y.A. Kapitanyuk, M. Bronz, G. Hattenberger, M. Cao, Guidance algorithm for smooth trajectory tracking of a fixed wing UAV flying in wind flows, 2017 IEEE International Conference on Robotics and Automation (ICRA), (2017), pp. 5740–5745, http://dx.doi.org/10.1109/ICRA.2017.7989674.
[81] P. development team, Paparazzi source code documentation, http://docs.paparazziuav.org/latest/.
[82] P. development team, Paparazzi wiki, https://wiki.paparazziuav.org/wiki/Main_Page/.
[83] AutoQuad web page, http://autoquad.org/.
[84] AutoQuad, AutoQuad source code, www.github.com/mpaperno/aq_flight_control/.
[85] U. of Salzburg, JAviator Project, https://github.com/cksystemsgroup/JAviator/.
[86] U. of Salzburg, JAviator web, http://javiator.cs.uni-salzburg.at/.
[87] S.S. Craciunas, C.M. Kirsch, H. Röck, R. Trummer, The JAviator: A high-payload quadrotor UAV with high-level programming capabilities, Proc. of the AIAA Guidance, Navigation, and Control Conference (GNC), (2008). Honolulu, HI, USA
[88] Microsoft, AirSim web page, https://github.com/Microsoft/AirSim.
[89] O.S.R.F. (OSRF), Gazebo web page, https://bitbucket.org/osrf/gazebo.
[90] O. LAAS and labs, Morse web page, https://github.com/morse-simulator/morse.
[91] P. engineering team, jMAVSim web page, https://github.com/PX4/jMAVSim.
[92] E.N. de l'Aviation Civil, Paparazzi web page, https://github.com/paparazzi/paparazzi/tree/master/sw/simulator.
[93] S.D. Levy, HackflightSim web page, https://github.com/simondlevy/HackflightSim.
[94] S. Shah, D. Dey, C. Lovett, A. Kapoor, Airsim: High-fidelity visual and physical simulation for autonomous vehicles, Field and Service Robotics, (2017) arXiv:1705.05065. https://arxiv.org/abs/1705.05065
[95] B. Karis, E. Games, Real Shading in Unreal Engine 4, (2013). Proc. Physically Based Shading Theory Practice
[96] N. Koenig, A. Howard, Design and use paradigms for gazebo, an open-source multi-robot simulator, Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, volume 3, IEEE, 2004, pp. 2149–2154.
[97] O.S.R. Foundation, Gazebo robot simulation made easy, http://gazebosim.org/.
[98] J. Meyer, Hector Quadrotor, http://wiki.ros.org/hector_quadrotor.
[99] E. Games, Unreal Engine, https://www.unrealengine.com/.
[100] U. Technologies, Unity, https://unity3d.com/.
[101] G. Echeverria, N. Lassabe, A. Degroote, S. Lemaignan, Modular open robots simulation engine: MORSE, IEEE International Conference on Robotics and Automation, (2011), pp. 46–51, http://dx.doi.org/10.1109/ICRA.2011.5980252.
[102] B. Wallace, Blender Game Engine, https://www.blender.org/.
[103] pixhawk.org, jMAVSim, https://pixhawk.org/dev/hil/jmavsim.
[104] github.com, Px4/jmavsim, https://github.com/PX4/jMAVSim/tree/master/src/me/drton/jmavsim.
[105] S.D. Levy, Hackflightsim/source, https://github.com/simondlevy/HackflightSim/tree/master/Source/HackflightSim.
[106] J.T.M. Ingibergsson, U.P. Schultz, M. Kuhrmann, On the use of safety certification practices in autonomous field robot software development: a systematic mapping study, Product-Focused Software Process Improvement, Springer, 2015, pp. 335–352. http://dl.acm.org/citation.cfm?id=2972500

Emad Ebeid is an Assistant Professor at the Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, Denmark. Ebeid received his Ph.D. in Computer Science from the University of Verona, Italy, in 2014. His research interests are in the areas of networked embedded systems, hardware/software co-design of drone technologies and smart grid applications. He is or was involved in several EU projects in embedded systems and smart grid applications. Currently, he is special session chair of System Design for Collaborating Intelligent Systems session at EUROMICRO Conference on Digital System Design and a guest editor of a special issue of MDPI Energies journal. He is a senior IEEE member.

Martin Skriver is a research assistant at the Mærsk Mc-Kinney Møller Institute, University of Southern Denmark where he has been a part of the SDU UAS Center since August 2015. He has a M.Sc. degree in Engineering - Robotic System from the University of Southern Denmark. His main research areas have been with embedded electronics within battery management systems, GSM/GNSS based drone tracking and FPGA based systems.

Kristian Husum Terkildsen is a research assistant at the Mærsk Mc-Kinney Møller Institute, University of Southern Denmark. Terkildsen has a Master's degree in Robot Systems Engineering with specialization in Unmanned Aerial Systems. He has been a part of the SDU UAS Center since June 2017. His main focus areas are within system integration and navigation in GPS denied environments.

Kjeld Jensen is an Associate Professor since 2007 at the SDU UAS Center, University of Southern Denmark. His research interest covers experimental development of embedded systems and software to support a wide range of applications in unmanned systems. He has designed and developed a number of mobile robots for outdoor tasks and he is the architect and principal programmer behind the open source-software platform FroboMind. His recent work focus on novel applications requiring unobserved autonomous operation of UAS and mobile robots.

Ulrik Pagh Schultz is an Associate Professor since 2005 at the SDU UAS Center, MMMI, University of Southern Denmark. Schultz received his B.Sc. and M.Sc. degrees in Computer Science from the University of Aarhus, in 1995 and 1997, respectively, and a Ph.D. degree in Computer Science from University of Rennes, in 2000. From 2000 until 2005, he was a faculty member at University of Aarhus. His research interest covers software engineering for robotics of all kinds (aerial or not) and domain-specific languages. Currently he is Chair of the IFIP Working Group 2.11 on Program Generation (since 2013). He is a member of ACM and IEEE.