Preprint of: Burdziakowski P., Razmjooy N., Estrela V., Hemanath J., Open-source software (OSS) and hardware (OSH) in UAVs, (2020), pp.49-66, https://digital-library.theiet.org/content/books/10.1049/pbce120g_ch3
In: Estrela V. V., Hemanth J., Saotome O., Nikolakopoulos G., Sabatini R. (ed.), Imaging and Sensing for Unmanned Aircraft Systems: Volume 2: Deployment and Applications, (Control, Robotics &amp; Sensors, 2020) DOI: 10.1049/PBCE120G, IET Digital Library, https://digital-library.theiet.org/content/books/ce/pbce120g

Open Source Software (OSS) and Hardware (OSH) in UAVs

Pawel Burdziakowski[1], Navid Razmjooy[2], Vania V. Estrela[3], Jude Hemanth[4]

[1]Faculty of Civil and Environmental Engineering, Gdansk University of Technology, Poland.
pawelburdziakowski@gmail.com
[2]Department of Electrical Engineering Tafresh University, Tafresh, Iran
navid.razmjooy@ieee.org
[3]Departamento de Engenharia de Telecomunicações, Universidade Federal Fluminense, Rio de Janeiro, Brazil
vania.estrela.phd@ieee.org
[4] Karunya University, Coimbatore, India,
jude_hemanth@rediffmail.com

ABSTRACT

The popularity of the Open Source Tool (OST) has expanded significantly. This is the case for Unmanned Aerial Vehicles (UAVs) based on Open Source Hardware (OSH) as well. Open Source Software (OSS) and OSH can be applied in a wide range of applications and can improve several technologies. The chapter begins with an introduction to OSS depicting its rationale, description of fundamental differences between OSS and Proprietary Software (PS), what benefits OSSs provide to overall UAV community, the motives leading people to pick up an OSS instead of a PS, which helps the academic and research community. Chapter covers also some OSSs used within the UAV community to support all aspects of UAV technology and the Remote Sensing (RS) and photogrammetry data post-processing chain. It is possible to build fully autonomous and operational UAV based only on OSH and OSS. The chapter describes the state of the art for OSS widely used in UAV technology, the software used in all aspects of UAV technology like: ARDUPILOT-based Autopilot firmware, MISSION PLANNER-based ground station, OPENTX transmitter software, MINIM On-Screen Data Software (OSD), OPEN DRONE MAP photogrammetry data processing suite, Web drone data processing suite WebODM. The chapter describes software, built-in functions, and features as well as platform requirements. A typical UAV photogrammetry workflow for drone construction with flight planning/execution and OSS data processing is provided.

Keywords: UAV; open source software; virtual communities; imaging; open source hardware; autopilot; benchmarking; simulation

INTRODUCTION

Micro Aerial Vehicles (MAVs) are Unmanned Aerial Vehicle (UAV) class possesses a 5-kg Maximum Takeoff Weight (MTOW), approximately 1-hour endurance and an operational range nearby 10 km. MAVs are widespread and widely applied in Photogrammetry and Remote Sensing (RS). Potential MAVs' use in classical photogrammetry demands low-cost alternatives. There are two main ideas, for MAV construction and software development: the Open Source Hardware (OSH) and the Open Source Software (OSS). Amateurs and researchers mainly make MAVs built with OSH and OSS. The most fruitful open source ventures have academical roots, and now its participation embraces a wide variety of academic initiatives. This chapter highlights open source projects' cons and pros. Purely commercial MAV solutions where the software and hardware are developed, provided and maintained by private companies may limit the UAV development, especially in the aftermath of a world's crisis although there are options with high quality and accuracy.

OPEN SOURCE SOFTWARE

Firstly, it is paramount to explain the open source concept [1], which denotes adjustable and shareable software or hardware. Its design is overtly accessible and associated to a broader set of (also known as the open source way). OSS, OSH, projects, products, and initiatives cover open exchange, cooperation, transparency, rapid prototyping, meritocracy, and community-oriented improvement. OSS is a software with source code that anyone can scrutinize, alter, and enhance. In computing science, the source code encompasses a collection of code with annotations, written with a human-friendly programming language, usually as plain text [2]. The program source code is especially designed to smooth programming of the computer-performed tasks done mostly by the written source code. A compiler often transforms the source code into binary code understood by the target machine (PC, Arduino Board, Android, etc.). Most application software is disseminated as executable files only. If the source code was included, then it would be helpful anybody that might want to study or revise the program.

Since programmers can add features to the source code or modify parts that do not always work correctly, the program can be improved by anyone who can do it. Mission Planner, Ardupilot, OpenTX, OpenDroneMap, Web ODM are examples of MAV and photogrammetry OSS.

On the opposite side, there is the so-called "closed source" or commercial or proprietary software. In that case source code is that only the individual, team, or organisation who created it, maintains, controls and modifies it. In that case, only the original authors of closed software can copy, examine, and rework that software. At this time, almost all commercial MAVs rely on private software. The companies like DJI (Dajiang) Innovations, Parrot SA, Yuneec offers commercial products, MAVs that are running closed software, and the other programmers cannot modify its source code.

In general, the OSS promotes collaboration and sharing that permit other persons to adapt the source code and integrate those modifications into their projects. A very vivid example, within the UAV public, is the ArduPilot Community and organisation. At this time, Ardupilot is the most advanced, full-featured and trustworthy existing open source autopilot software [3]. It has been developed over the years by qualified engineers, computer scientists, and communities. The autopilot software can regulate any robotic vehicle (e.g., conventional aeroplanes, multi-rotors, helicopters, boats and submarines). The concept of open-source code based implies it has rapid development always at the cutting edge of technology. Users profit from an all-encompassing ecosystem of sensors, companion computers in addition to communication systems due to the many peripheral suppliers and interfaces. In conclusion, open source code can be appraised to safeguard compliance with security and confidentiality. As stated in [1], people prefer OSS to PS for some reasons, including:

• Control. More control over software favours the choice of an OSS. The code can be tested to make sure that if the UAV is not doing anything, then the control should stay dormant.
• Training. The open source code can help people to become better programmers. Students, investigators or anybody else can efficiently study the publicly accessible code to make better software, share their work with others, and invite comment and critique, as development progresses. When some mistakes are discovered in the program's source code, it can be shared with others to circumvent the repetition of the same mistakes.
• Security. Some people prefer OSS due to its satisfactory security and stability when compared to PS. Someone might identify and fix errors or omissions not seen by the program's original authors thanks to the fact that anyone can assess and modify OSS. Programmers can correct, update, and upgrade OSS more quickly than proprietary software.
• Stability. The OSS is desired in essential and long-standing projects as the source code is publicity distributed, users can rely on that software for critical activities, and without the fear that their tools will disappear or degrade if the original developers discontinue the project. OSS also tends to incorporate as well as a function according to open standards.

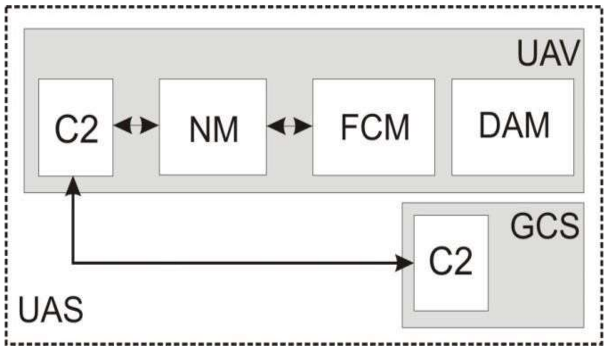

Figure 1 UAS (UAV-CPS) system basic modules [4]

OPEN SOURCE UAS

UAV is a part of a Cyber-Physical System (CPS) named UAV-CPS or UAS, which entails a drone, at least a Ground Control Station (GCS) besides a Communication and Control (C2) link between the GCS and UAV [4]. If the complete UAV-CPS uses OSH and OSS it can be named Open Source Unmanned Aerial System (OSUAS), it merely means that all main components and software used within UAS employs open source idea.

Critical UAV building blocks are onboard UAV platforms like the Navigation Module (NM), Flight Control Module (FCM), and mechanical servos. Depending on the UAV primary purpose and associated tasks, the payload may differ. For photogrammetry and RS, the payload can be a Data Acquisition Module (DAM).

The NM is the most critical module onboard a UAV that repeatedly provides the aircraft's location, speed, and altitude to the FCM. Hence, the NM feeds the FCM with critical UAV guiding data. The NM contains a Navigation Systems (NSs) to fix the position of a platform (usually via a GNSS) and orientation system with accelerometers (motion sensors) and gyroscopes (rotation sensors) to continuously estimate the orientation, direction, and speed of the Inertial Measurement Unit (IMU) stage. Autopilots encompass NM (NS+OS), and FCM integrated into one module.

The FCM is a device commanding a flight, which means leading a UAV to the elected position while setting the correct orientation and speed. The FCM has two parts: (i) the data analysis stage that receives commands from the system operator, and (ii) the current flight parameters control module from the NM to analyse and send commands for actually correcting the flight parameters with mechanical servos and an electronic engine speed controller that moves all control surfaces and regulates the engines' speeds.

At this time, most known and considered as a state of art of the OSS is an Ardupilot. Ardupliot software can operate onboard open hardware and closed hardware. The supported open hardware boards are Pixhawk, The Cube (also known as a Pixhawk 2), Pixracer, Pixhack, F4BY, Erle-Brain, OpenPilot Revolution, Beagle Bone Blue, PXFmini RPi Zero Shield, TauLabs Sparky2.

When considering OSH, it worth to remember that it consists of physical technology artefacts designed and made available by the open design effort. OSH means that hardware evidence is easily identified so that others can tie it to the maker movement. Hardware design, like mechanical drawings, diagrams, bills of material, PCB layout documents, HDL source code, integrated circuit layout documents, and the software driving the hardware can be all released under free terms and made accessible for the community.

In the case of open hardware autopilots, a manufacturer shares all necessary data to build the same hardware depending on available data. If one connects it with OSS, it makes an extensive and open UAV autopilot receptive to betterments.

The GCS comprises stationary or moveable devices to observe, command in addition to controlling the UAV. The GCS can operate from the ground, aquatic or air working as a bridge a connection among a machine and an operator. The UAV GCS design has specific functional requirements. Basic functionalities are:

(i) UAV control – a capability to successfully control and fly the drone thru mission;
(ii) payload control – the ability to operate sensors from the ground;
(iii) mission planning – functionality that aids UAVs operators to plan the mission while providing the required knowledge inputs concerning the UAV capabilities and limitations;

(iv) payload evidence analysis and broadcasting – the capacity to disseminate data from the payload to eventual users;
(v) system/air vehicle diagnostics – automatic test procedures for UAV while keeping the GCS effective maintenance and deployment;
(vi) operator training – the facility to train the drone controller as well as practising mission plans; and
(vii) emergency procedures, post-flight analysis – the capability to amass both flight and payload data to analyse it after the flight.

The most popular GCS OSS supporting the autopilot hardware and software pieces stated above are APM Planner 2.0, Mission Planner, MAVProxy, QGroudControl, Tower and AndroPilot.

The DAM module includes optical RS devices, and airborne image acquisition systems with frequencies from the visible band to the Near Infrared (NIR), the Thermal Infrared (TIR), microwave systems, active and passive ranging instruments.

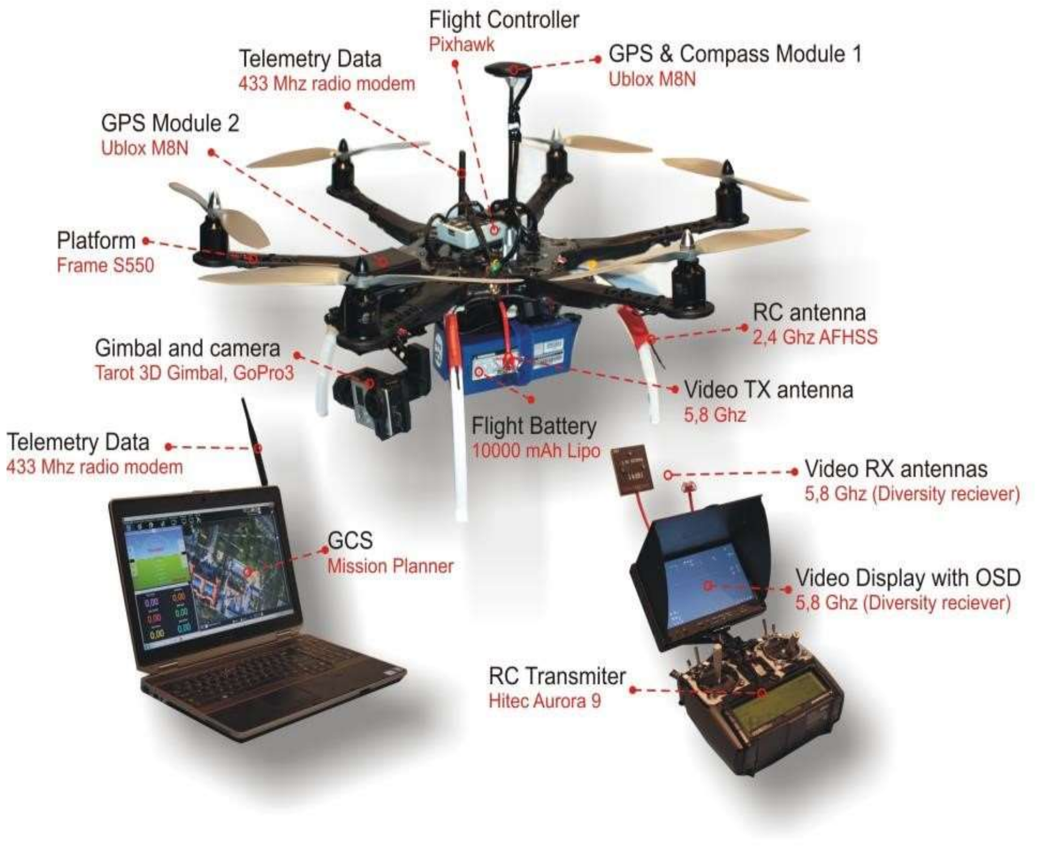

Figure 2 Open source hardware and software-based UAS [5]

Going deeper into the MAV system internal design, Figure 3 presents an advanced system architecture relying purely on the open hardware design. The system initially is designated for real-time visual navigation and photogrammetry tasks [6].

In this particular design, for real-time UAV applications, a reasonable commuting power is needed. The NVIDIA TX2 processor has been chosen to run all the required calculations in the shortest possible time. Jetson TX2 combines a 256-core NVIDIA Pascal GPU with a hex-core ARMv8 64-bit CPU platform, and an 8-Gb memory

(LPDDR4 with a 128-bit interface). The CPU platform has a dual-core NVIDIA Denver 2 together with a quad-core ARM Cortex-A57. The Jetson TX2 building block has a small size, light, low power, a size of 50 mm x 87 mm, 85 g, and 7.5-watt power consumption. These features make this credit-card-sized processor very suitable for MAV applications. Jetson TX2 is an outstanding class of GPU-enabled boards marketed today for autonomous frameworks. The central computing power comes from NVidia processor. NVidia processor, in that case, is only assigned for image calculation or other demanding commuting [7-9]. An application running in this processor should not impact flight control process. The flight control process is controlled by a flight controller (autopilot) ruining different processor. In that particular design, commands come from visual system (NVidia processor) are passed to the flight controller, only simple commands, similar to the remote user command, or some waypoint coordinates. The flight controller performs all the flight calculations. This communication between components happens via a standard protocol, fast and straightforward.

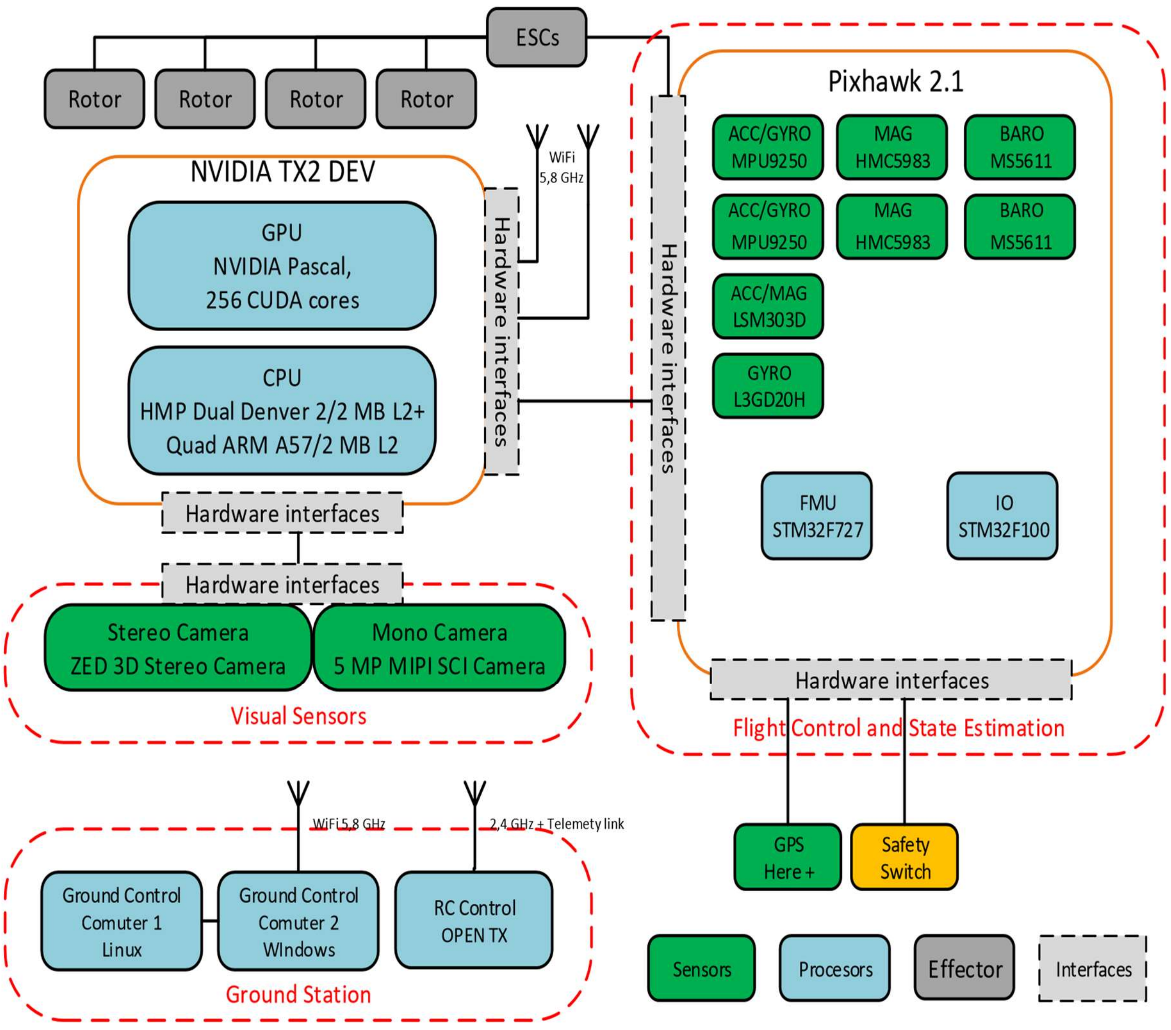

Figure 3 Open hardware MAV advanced system design [6]

UNIVERSAL MESSAGING PROTOCOL

As it can be noticed, the open source autopilot software can operate with many ground control station software and components. It would not be possible without a universal, open source, interoperable communication protocol.

The most popular, it can be considered at this time, as a standard protocol for MAVs is a protocol called MAVlink, which has an extremely lightweight messaging protocol intended for onboard MAVs' components relying on a recent hybrid publish-subscribe with a point-to-point design pattern. MAVlink sends (publishes) data streams as topics while configuration subprotocols set on a mission or parameter protocol-employing point-to-point with retransmission. This concept is very analogous to the Robot Operating System (ROS) [10], which is another prominent and influential opensource framework. ROS is a flexible framework with a group of libraries, tools, and conventions to make straightforward the task of creating sophisticated and robust automatic behaviour across a wide diversity of robotic platforms. MAVLink messages adopt XML files where each of them delineates the message group kept up by a specific MAVLink system (aka dialect). Most GCSs and autopilots implement this reference message set in common.xml (so that most dialects use this scheme). MAVs and GCSs in addition to, other MAVLink systems, employ these generated libraries to communicate [11].

MAVLink protocol is very efficient whose first version (MAVLink 1) has an 8-byte overhead per packet, together with the starting sign and packet drop recognition. The subsequent version (MAVLink 2) has 14 bytes of overhead and is a much safer and more extensible. MAVLink is well-suited for applications with limited communication bandwidth because it does not require any additional framing. Ever since 2009, MAVLink has been allowing communication among many different vehicles, GCSs, and other nodes as well over diverse and challenging channels (with high latency/noise). It is a very reliable protocol because it arranges for methods to probe' drops, corruption, and need for authentication of packets. This technology runs on many microcontrollers and OSs (e.g., ATMega, dsPic, STM32, ARM7, Windows, MacOS, Linux, Android, and iOS) while supporting many programming languages. With a maximum number of 255 concurrent subsystems on the network (like UAVs and GCSs to name a few) it facilitates both offboard as well as onboard communications (e.g., amid a GCS and a MAV, and between autopilot and a surveillance camera MAVLink).

The example of advanced software architecture (Figure 4) presents high-level structures of the system, software elements and relations among them purely based on open software [6]. The specific structural options from any available possibilities have been chosen to enable real-time computing, compatible autopilot interaction possibilities, and any custom payload integration.

On the presented drone software architecture, flight control is on FC open software (responsible for flight control and standalone flight parameters computation) and the embedded computer software. The flight controller software uses Ardupilot FC firmware. Embedding commuting (NVidia TX2) is responsible for highly demanding sensors data calculation. NVidia TX2 is working on the Ubuntu Operating System (OS), also open source. A simple MavLink command is sent to FC thanks to the calculations. Ubuntu OS can work with the ROS for sensor data calculation. The ROS is a flexible framework for creating robot software consisting of an assortment of tools, libraries, in addition to conventions to simplify the creation of multifaceted and robust drone behaviour. On top of the mentioned software, the Flyt OS is integrating ROS, MAVlink and user applications and interface. This OS lets integrate external ROS/Linux libraries with custom data helping the execution of onboard in addition to offboard applications. The custom applications can be run on any OS and hardware (smartphones, PC, MAC, wearables, portable devices) to build an interface between the drone and human operator.

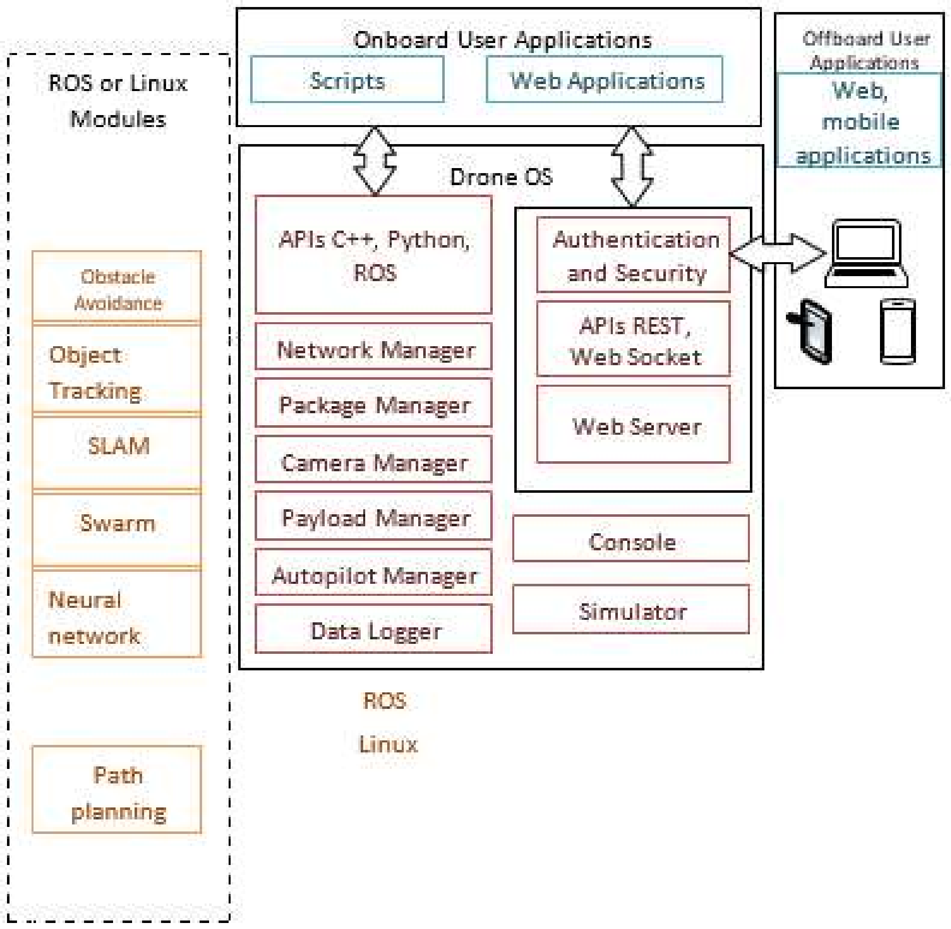

Figure 4 Advanced open-source MAV system architecture [6]

GROUND CONTROL STATION SOFTWARE

The GCS software should be able to operate with the desired autopilot and be able to run and control all necessary functionalities. This onboard software operates specific OSs including Windows, Linux, Android or iOS. The main functions entail:

(a) loading the built-in software (the firmware) into the autopilot controlling a specific vehicle;

(b) optimum performance via setup, configuring, and tuning of the drone operation;

(c) planning, saving and loading sovereign undertakings employing the autopilot together with a simple and intuitive point-and-click GUI established on Google or other supported mapping software platform;

(d) downloading and analysing mission logs; and

(e) interfaces with a PC flight simulator to build a full hardware-in-the-loop UAV simulator [5].

With additional hardware, software should enable: monitor the drone's status in action with backup copies of the telemetry logs, which enclose much more information about the onboard autopilot logs, and operating the drone in First-Person View (FPV) Mode. The software must support all autopilot functions as well as full mission planning abilities. Each facet of the autonomous or automatic mission may be programmed and controlled via the application.

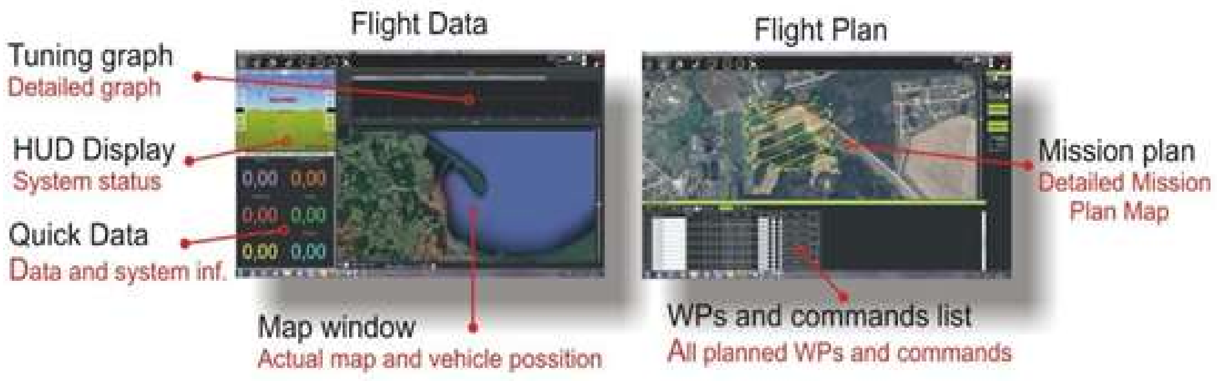

Figure 5 The Mission Planner user interface [5]

Table 1 GCS Software platforms (own elaboration)

	Mission Planner	APM Planner 2.0	MAVProxy	QGroundControl	Tower	Andropilot
Platform	Windows, Mac OS X (Using Mono)	Windows, Mac OS X, Linux	Linux	Windows, Mac OS X, Linux, Android and iOS	Android Phones and Tablets	Android Phones and Tablets
Source Code language	C# .net	Qt	Python	Qt QML	Android Studio	Scala (Java)
License	Open source (GPLv3)	Open source (GPLv3)	Open source (GPLv3)	Open source (GPLv3)	Open source (GPLv3)	Open source (GPLv3)

Currently, OSS like Mission Planner, APM Planner, MAVProxy, QGroudControl, Tower and AndroPilot are the most common software. The applications are designed for any OS and several different software architectures (Table 1). The common future, which allows operating with Ardupilot firmware is an open source communication protocol – MAVlink. GCS acts as an interface between human (operator), translates user inputs to MAVlink command, understandable for UAV's FC.

PROCESSING SOFTWARE

There are many photogrammetry software on the market, which is designed especially for UAV photogrammetry and RS applications. The most popular commercial software, despite its expensiveness, provides a wide range of image processing workflows, supporting almost all cameras and MAVs on the market [7, 12-15]. Companies guarantee full customer support, global service, training, and distribution. Software is relatively simple to use and is regularly updated. There are significant pros, which have to be mentioned here. There is no equivalent open source image processing software for MAV, which can cover all functionalities and simplicity of operation. Today, only one open source project can deliver some UAV photogrammetry product like point clouds, digital surface modes, orthorectified imagery, etc. The Open Drone Map (ODM) ecosystem is such a case, with its current subprojects WebODM and Node-ODM.

The Open Drone Map (ODM) is an open source software for processing mainly aerial drone imagery. It has to be highlighted that, the current ODM version number is 0.3.1 bet, and has started in 2011. This software handles micro air vehicle imagery taken by a non-metric camera, and is able to produce orthorectified imagery, digital surface models (DSM), digital terrain models (DTM), textured 3D models, and classified point clouds. ODM does not provide a Graphical User Interface (GUI) to interact with the user, what imposes basic knowledge as for the Linux OS terminal commands. The prepared input images are stored in the specified folder [9, 16, 17]. Terminal commands start processing the images and save the results in the desired folders structure. ODM does not provide any additional viewing result interface, and due to that fact, an additional software for viewing, editing or evaluating results is necessary. All processes are computed using a Central Processing Unit (CPU). This software does not support calculation by Graphical Processing Unit (GPU). GPU calculations, especially for processing images using NVIDIA CUDA technology [18], can be accelerated up to 8 times if we compare it with CPU computation.

As this project is named ecosystem, additional subprojects support ODM. Basing on the ODM engine, the web graphical user interface, to promote interaction with the user, was added to this ecosystem and is called WebODM. The WebODM is designed to be a user-friendly application for drone image processing. This ecosystem also delivers an application programming interface (API) for software developers. Because the WebODM is a kind of web-based user interface for ODM, it generates the same outputs as ODM.

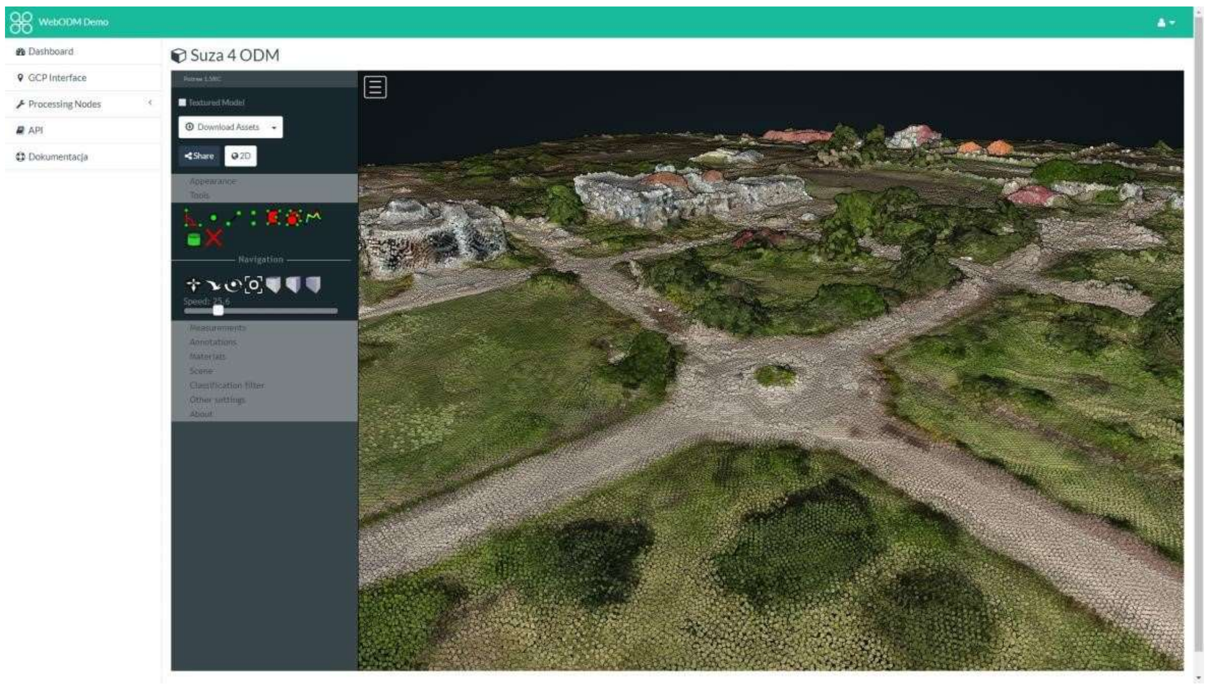

Figure 6 WEBOdm user interface: point cloud model

OPERATOR INFORMATION AND COMMUNICATION

During the UAV mission, operators should have actual information about the actual platform status and flight parameters so that there are few ways to handle data.

Visually, in case of Visual Line of Sight (VLOS) operator can observe the flying platform and control UAV behaviour. This is a primary method, but must be trained by the operator to get the UAV under control in any emergencies. In this method, only the operator can have the relative position of the platform, relative to other elements of the environment and actual platforms orientation.

GCS flight parameters display (telemetry) help during either VLOS and Beyond Visual Line of Sight (BVLOS) operation. The GCS offers the most comprehensive information about the UAV status and actual information. Telemetry data are transferred to the GCS and the GCS screen. Nevertheless, not all data are useful for the operator, and even some can disrupt the operator’s attention. Usually, the UAV data dispelled on the GCS are monitored by a second operator, in the two-person UAS configuration. In that case, when there is only one operator, the third method can be used.

Flight data overlaid on the operator's video stream and On-Screen Display (OSD). An OSD is a piece of information (image) superimposed on a screen picture. The vital information for the piloting operator is superimposed on the video stream. His method gives the one mane OS opportunity to simultaneously monitor the UAV status and flight parameters and control video operators stream, transmitted by the UAV.

MinimOSD and its modification like MinimOSD-Extra are the most popular OSS OS. This software runs on the cheap and open hardware board relying on the MAX7456 processor, which is a single-channel monochrome OSD generator that does not need an external video driver, EEPROM, sync separator, and video switch. The MAX7456 attends all national and global markets with 256 user-programmable characters (in NTSC and PAL standards). The Minim OSD Extra receives information from the flight controller via the MAVlink protocol on four configurable screens configured by the operator depending on actual needs.

On the presented example, the operator's screen (Figure 7) displays the most critical information, like (from top left) number of GPS satellites fixed (here 0), comas rose and actual heeding (240 deg.) Home Altitude (HA – 1 meter). In the middle, there is an artificial horizon line and the exact pitch and roll angle. Form the bottom left – battery status (voltage, current and used capacity), warnings section (here No GPS fix warning), and actual flight mode.

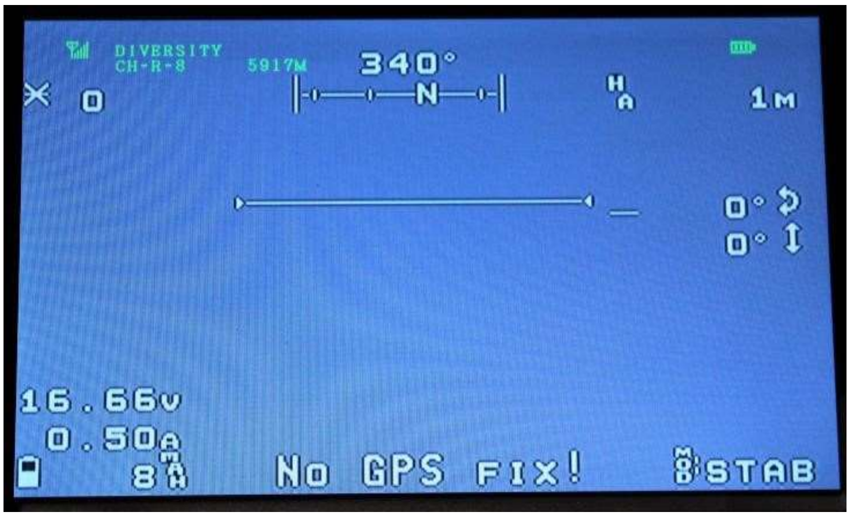

Figure 7 The MinimOSD display

To manually control or supervise the UAV, a radio link is required. Radio transmitter, on the ground and, relevant radio receiver. Radio Control (RC) transmitters are changing the operator's input to the desired radio signal and finally on the mechanical or electrical response on the UAV. Nowadays, the state of art of the open source transmitter software is the OpenTX. Precisely, OpenTX is an open source firmware for RC radio transmitters. This firmware gained popularity due to its high configurability and many more features than the ones found in traditional proprietary radios. According to the main OSS idea, the frequent feedback from the users around the word ensures the continued stability and quality of the firmware.

The OpenTX firmware and its hardware base are renowned players in the RC industry since this open source radio targets the mainstream market. The OpenTX project improves both the hardware and software sides of a product, and they are low cost and devoid of the customary marketing-driven limitations, that most manufacturers place on their offerings. OpenTX offers features that match and even exceed those of the highest end radios in the industry [19]. The most significant difference, when compared to the commercial approach is the open source community-driven firmware, so unlike with major manufacturers, if there is a need to implement a particular function or users have good improvement suggestions realised a few days later, and published.

OPEN SOURCE PLATFORM

Since OSS is shared and written by its community, for use and diffusion, it requires some platforms to share, collaborate, manage changes and versions. Many open source projects are available on the GitHub platform, which hosts code for version control and collaboration, and it allows people to work together on projects from anywhere [20]. This platform consists of user repositories and delivers particular functionalities. A repository is the essential GitHub element, and that contains a project's folder containing all its associated files with documentation and each file's historic of revisions. Regardless if they are public or private, repositories allow for multiple collaborators. The GitHub platform hosts mostly code, and additionally, it supports the subsequent formats and essential features like:

- Issues' tracking (including feature requests) with labels, milestones, assignees and a search engine to suggested improvements, tasks or questions related to the repository. For public repositories, anyone can create them, and repository members moderate them. Each issue encompasses its discussion forum, labels, and user assignments.
- Wiki - a subsite on the repository on which users collaboratively modify content and structure directly from the web browser. In a typical wiki, the text is written using a simplified markup language and often edited with the help of a rich-text editor [21],
- Pull requests with code review, and comments contain proposed changes to a repository submitted by a user with the corresponding acceptances or rejections by a repository's collaborators. Each pull request has a discussion forum.
- Commitment history
- Documentation, including automatically rendered README files,
- visualisation of repository data like contributors, commitments, code frequency, punch card, network and members,
- GitHub Pages are small websites can host public repositories on GitHub.
- Visualisation of additional data alike to 3D render files, geospatial data, Photoshop's native PSD and PDF documents.

All software mentioned above has its repository and full history on the GitHub platform.

FUTURE WORK

Most of the UAV's challenges gravitate through the dichotomy computational tasks performed onboard and offboard to decrease energy consumption. Furthermore, the tendency is to have UAVs using while also providing IoT Devices and IoT Services thru dependable Wireless Communication Networks [22].

A) OHS Challenges

The OSH concept relates to any physical technology that is it goes beyond electronics. Depending on how one interprets this notion, it can be considered even much more popular than OSS because people have been sharing recipes for quite a long time [22].

One problem is the technical expertise and the need to have tools and instruments. Programming requires a computer, reading, and typing even if the people involved are highly capable.

One situation where OSH can truly lift off is in the usage of FPGAs because designing with them is as unpretentious as conniving software. However, OSH has problems more difficult to describe and solve than the development languages available, e.g., VHDL, Verilog, and LabVIEW FPGA because they require thinking about challenges differently. There are timing and parallelisation problems that cannot be learned like in serial programming.

Moreover, the hardware is not as easily replicated as software. If one wants a piece of OSS, it is just a matter of downloading it. Hardware replication requires a little money to acquire it and transport it. OSS software can be distributed for free, but OSH (in its more accessible form) involves the VHDL/Verilog/FPGA code with some way to produce the physical system.

As much as integrated circuits and CAD projects go, it is entirely impractical to produce open-source alternatives given today's limitations.

A hardware description can be copyright, but not the physical realisation itself. The real physical implementation can only be secured by patent, which demands time and money. OSS typically can benefit from copyright because the software can be attached to its license differently from the hardware. Hence, one cannot enforce the OSH philosophy in the courts [22].

B) Open Data

A UAV-CPS cater to a variety of applications that demand different types of databases for the sake of benchmarking all its stages. However, handling Open Source Data (OSD) poses some challenges.

The Open4Citizens project [22] strategies to promote open data usage and to support the foundation of communities and practices. It addresses design complexity over multi-layered interventions going from the organisation of extensive processes to the definition of a network.

It is important to stress that there is a tendency to decoupling problems so that the workflow of people from different backgrounds is not disturbed.

C) Cloud Data Center

The development of UAV-CPSs needs to build upon cloud computing infrastructure delivered by a combination of tools resembling the Amazon Web Services (AWS). Hence, the analyses and classification procedures, which embraces the segmentation, feature computation, and labelling steps executed in computer clusters with different virtual machines' sizes. One cluster node must be reserved for the Hadoop JobTracker (aka the master node) only responsible for task scheduling and managing.

The Apache Hadoop software library is an arrangement that permits large data sets' distributed processing across computer clusters via simple programming models. It scales up from a single server to a myriad of machines, each with local computation and storage. Instead of relying on high-availability high-performance hardware, the library detects and handles failures working at the application layer. Hence, highly-available service is delivered on top of a computer cluster where each computer and each cluster may be prone to failures.

As big data augments and becomes indispensable, open source implementations for working with it will undoubtedly remain rising as well:

(i) The Apache Software Foundation (ASF) supports many of these big data projects.
(ii) Elasticsearch is an enterprise search engine built on Lucene. Its Elastic stack produces comprehensions from structured and unstructured records.
(iii) Cruise Control by LinkedIn runs Apache Kafka clusters at a massive scale.
(iv) TensorFlow has proliferated since Google open sourced it. Thus, it will increase the availability of computational intelligence tools and popularising machine learning as a result of its ease-of-use concept.

D) Crowd-Sourced Data in UAV-CPSs

UAVs can perform crowd surveillance [23] and use information mined from social networks to facilitate image comprehension. To assess the use cases, video data can be offloaded to be processed by a Mobile Edge Computing (MEC) node compared to the local video data treatment onboard UAVs. MEC-based offloading can help to save the limited energy of UAVs.

Cloud computing, big data, software-defined networks/network functions virtualisation, cloud security, sensor networking, distributed machine learning, internet of things software, 3D scanning software, etc. will all grow faster than normal and change in the next 5 years.

E) Control of UAV Swarms

Nowadays, cloud services handle a very high number of loosely-coupled UAV microservices. The paradigm shift towards microservices takes into consideration that lots of the assumptions about cloud-based UAV-CPSs are continually changing, as well as current opportunities and challenges when optimising Quality of Service (QoS) and cloud utilisation especially when UAV swarms are involved [25-33].

Open-source benchmarks built with microservices must allow simulation and verification of large end-to-end services, in a modular and extensible fashion. To incorporate suites that can permit the combination of swarm control and coordination with other more human-dependent applications such as social network, entertainment/media service, e-commerce, e-banking, and different types of IoT applications still require an OSS and OSH development. This necessity arises from the impact UAV swarms may have in networking, security, and operating systems, among other issues. They challenge cluster management and other tradeoffs regarding application design/deployment and programming frameworks. Scale impacts microservices in real frameworks due to the extremely high number of users, and poses increased pressure on the prediction of required parameters like performance.

CONCLUSIONS

In this chapter, the conception of Open Source Tools' (OSTs) has been studied. This panorama extends to this is the situation for MAVs in light of OSH too. OSS and OSH can suit a wide scope of applications and improve technologies. This text starts with a prologue to OSS portraying its method of reasoning, establishing contrasts among OSS and Proprietary Software (PS). OSS can benefit a large part of the UAV community with the intentions of driving individuals to use OSS rather than a PS, which helps the academic, amateur and research communities. Afterwards, the chapter covers some OSSs utilised inside the UAV people to help all parts of UAV innovation and the RS/photogrammetric information post-preparing chain. It is conceivable to manufacture completely self-governing and operational UAVs dependent on OSH and OSS.

This chapter highlights the multidisciplinary nature of issues related to the OSS and the OSH communities. It outlines different perspectives of the communities dedicated to their development [24]: virtual structures, collective knowledge, stimulus, innovation, shared intelligence, community organisation, learning, and achievement.

OSH requires physical equipment, while OSS involves virtual entities. Hence, the OSS product can be interminably replicated at almost no price tag and shared straightforwardly. OSH necessitates actual constituents and expenses to reproduce each unit or instance. Moreover, designing hardware has additional necessary degrees of freedom, which are not simple replacements of one another in contrast to the design of software where it is all code.

REFERENCES

[1] I. Red Hat. (2019). Available: https://opensource.com/

[2] P. Iora, M. Taher, P. Chiesa, and N. Brandon, "A one dimensional solid oxide electrolyzer-fuel cell stack model and its application to the analysis of a high efficiency system for oxygen production," *Chemical engineering science,* vol. 80, pp. 293-305, 2012.

[3] A. D. Team, ""ArduPilot," *URL: www. ardupilot. org, accessed,* vol. 2, p. 12, 2016.

[4] P. Burdziakowski and J. Szulwic, "A commercial of the shelf components for a unmanned air vehicle photogrammetry," *16th International Multidisciplinary Scientific GeoConference SGEM 2016,* vol. 2, 2016.

[5] P. Burdziakowski, "Low cost hexacopter autonomous platform for testing and developing photogrammetry technologies and intelligent navigation systems," 2017.

[6] P. Burdziakowski, "UAV design and construction for real time photogrammetry and visual navigation," in *2018 Baltic Geodetic Congress (BGC Geomatics)*, 2018, pp. 368-372.

[7] A. M. Coelho and V. V. Estrela, "Data-driven motion estimation with spatial adaptation," *International Journal of Image Processing (IJIP),* vol. 6, p. 54, 2012.

[8] D. J. Hemanth and V. V. Estrela, *Deep Learning for Image Processing Applications* vol. 31: IOS Press, 2017.

[9] B. S. Mousavi, P. Sargolzaei, N. Razmjooy, V. Hosseinabadi, and F. Soleymani, "Digital image segmentation using rule-base classifier," *American Journal of Scientific Research ISSN,* 2011.

[10] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs*, et al.*, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009, p. 5.

[11] D. Mendes, N. Ivaki, and H. Madeira, "Effects of GPS Spoofing on Unmanned Aerial Vehicles," in *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2018, pp. 155-160.

[12] V. V. Estrela and A. E. Herrmann, "Content-based image retrieval (CBIR) in remote clinical diagnosis and healthcare," in *Encyclopedia of E-Health and Telemedicine*, ed: IGI Global, 2016, pp. 495-520.

[13] P. Moallem and N. Razmjooy, "Optimal threshold computing in automatic image thresholding using adaptive particle swarm optimization," *Journal of applied research and technology,* vol. 10, pp. 703-712, 2012.

[14] B. S. Mousavi and F. Soleymani, "Semantic image classification by genetic algorithm using optimised fuzzy system based on Zernike moments," *Signal, Image and Video Processing,* vol. 8, pp. 831-842, 2014.

[15] N. Razmjooy, B. S. Mousavi, M. Khalilpour, and H. Hosseini, "Automatic selection and fusion of color spaces for image thresholding," *Signal, Image and Video Processing,* vol. 8, pp. 603-614, 2014.

[16] V. Estrela, L. A. Rivera, P. C. Beggio, and R. T. Lopes, "Regularized pel-recursive motion estimation using generalized cross-validation and spatial adaptation," in *16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003)*, 2003, pp. 331-338.

[17] N. Razmjooy, B. S. Mousavi, B. Sadeghi, and M. Khalilpour, "Image Thresholding Optimization Based on Imperialist Competitive Algorithm," in *3rd*

Iranian Conference on Electrical and Electronics Engineering (ICEEE2011), 2011.

[18] W. Błaszczak-Bąk, A. Janowski, and P. Srokosz, "High performance filtering for big datasets from Airborne Laser Scanning with CUDA technology," *Survey Review,* vol. 50, pp. 262-269, 2018.

[19] T. K. Jespersen, "Software Defined Radio," 2015.

[20] GitHub. (2019). *GitHub Help*. Available: Available: https://help.github.com/

[21] F. H. Knight, "Capital and interest," *Encyclopedia Britanica,* vol. 4, pp. 779-801, 1946.

[22] N. H. Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives," *IEEE Internet of Things Journal,* vol. 3, pp. 899-922, 2016.

[23] M. Ghazal, Y. AlKhalil, A. Mhanna, and F. Dehbozorgi, "Mobile panoramic video maps over MEC networks," in *2016 IEEE Wireless Communications and Networking Conference*, 2016, pp. 1-6.

[24] M. R. Martínez-Torres and M. d. C. Diaz-Fernandez, "Current issues and research trends on open-source software communities," *Technology Analysis & Strategic Management,* vol. 26, pp. 55-68, 2014.

[25] J. Gray. "Design and Implementation of a Unified Command and Control Architecture for Multiple Cooperative Unmanned Vehicles Utilizing Commercial Off the Shelf Components." 2015.

[26] S.G.P. Hardy. "Implementing Cooperative Behavior & Control Using Open Source Technology Across Heterogeneous Vehicles." 2015.

[27] A. Bingler, and K. Mohseni. "Dual-Radio Configuration for Flexible Communication in Flocking Micro/Miniature Aerial Vehicles." 2018.

[28] L. He, P. Bai, X. Liang, J. Zhang, and W. Wang. "Feedback formation control of UAV swarm with multiple implicit leaders." 2018.

[29] T. Larrabee, H. Chao, M. B. Rhudy, Y. Gu, and M. R. Napolitano. "Wind field estimation in UAV formation flight." 2014 American Control Conference, 2014: 5408-5413.

[30] V. Stepanyan, K.K. Kumar, and C. Ippolito. "Coordinated Turn Trajectory Generation and Tracking Control for Multirotors Operating in Urban Environment." 2019.

[31] R. Ahmadi, N. Hili, L. Jweda, N. Das, S. Ganesan, and J. Dingel. "Run-time Monitoring of a Rover: MDE Research with Open Source Software and Low-cost Hardw

[32] Robert Mies, Jean-François Boujut and Rainer Stark. "What is the "Source" of Open Source Hardware?" 2017.

[33] Y. Zhang, Y. Gan, and C. Delimitrou. "Accurate and Scalable Simulation for Interactive Microservices." 2019.