LOGI – Scientific Journal on Transport and Logistics
*Vol. 13 No. 1 2022 DOI: 10.2478/logi-2022-0018*
© 2022 R. Kampf *et al.* This is an open access article licensed under the Creative Commons Attribution-NonCommercial-NoDerivs License (http://creativecommons.org/licenses/by-nc-nd/3.0/).

# Creation of SW for Controlling Unmanned Aerial Systems

Rudolf Kampf[1*], Jakub Soviar[2], Ladislav Bartuška[1] and Milan Kubina[2]

[1]*The Institute of Technology and Business in České Budějovice, Faculty of Technology, České Budějovice 37001, Czech Republic; Email: kampf@mail.vstecb.cz, bartuska@mail.vstecb.cz*
[2]*University of Zilina, Department of Management Theories, Faculty of Management Science and Informatics, Slovakia; Email: jakub.soviar@fri.uniza.sk, milan.kubina@fri.uniza.sk*

***Corresponding Author:** Rudolf Kampf

Received: 20 August 2022; Revised: 21 September 2022; Accepted: 13 October 2022; Published: 21 November 2022

**Abstract:** This paper deals with an analysis of software functions that can be used to control Unmanned Aerial Vehicles (UAV), and particular design of a simple application (program) for flight planning with the use of predetermined points (the so-called Waypoints). The paper contains a description of how to set up the development environment as well as a brief description of the MAVLink protocol and its use by means of statements of several methods used in the program. Ardupilot Mega was preferred as a control unit which supports the MAVLink protocol. Although the application is fully functional, it is designed in its current form rather as a demonstration of possibilities of using the C# programming language and the MAVLink protocol parser than being for a regular use. It may be seen (and that is how it was intended) as a guide for anyone wishing to try to program their own application.

**Keywords:** UAV, drone, unmanned aerial vehicle, waypoint, autopilot

## 1. Introduction

After the necessary technologies were released into commercial operations at the beginning of the 21st century, Unmanned Aerial Vehicles (UAV) and Unmanned Aerial Systems (UAS) have become available for a variety of uses. The field of UAV (referred to as "drones") is continuously developing and being applied in relation to diverse industries. At present, UAV are used not only in the military, but also in scientific and commercial areas, the film industry or merely for fun [1]. There are publications dealing with UAV in general, describing their history, technical details and even constructing them on one´s own, e.g. “Unmanned Aircraft Systems: UAVS Design, Development and Deployment” [2], “Introduction to UAV systems” [3] or “Drony” in Czech [1].

The authors' intention is to present a part of the research, namely the issue of software design so that the application concerned is easy to understand and use even for regular UAV users. Therefore, the objective of the paper is to analyse functions of software controlling UAV, and program the application for controlling such vehicles with the use of waypoints.

Recently, the trend in the field of UAV has been to move from manual control to semi-automatic or fully automatic control. The semi-automatic mode or the fully automatic mode are not always guaranteed when considering UAV. There are many systems, particularly the cheaper and recreational ones, which can only be operated manually. On the other hand, automatic flight systems are supplied with software for controlling UAV that comes directly from their manufacturer, or it is possible to use third-party products both for personal computers and (nowadays mainly) for mobile devices. Not every software can yet be used with all UAV [4]. The proprietary ones provided by their manufacturers are intended only for their own products, whereas open-source programs support only a certain type of hardware related to developers and therefore focus on a limited range of devices.

## 2. Software for Controlling UAV

First, it is necessary for a drone to fly its entire route and record the position of waypoints in a newly created mission that can be saved for later reuse. After starting its mission, the drone will automatically fly according to the set route and its pilot can thus fully concentrate on the given task.

Very interesting is DJI application, namely Ground Station Pro (DJI GS Pro), designed to control or plan automatic flights. It has a considerable number of functions and, unlike the previous programs, allows, among other things, to plan missions in advance. It is free except for PhotoMap, 3DMap POI and Import KML modules [4]. A minor drawback of the application is its limited availability only for the iOS system and its compatibility only with some DJI products.

As regards Windows, DJI offers the PC Ground Station program. Its most interesting features are predefined flight path patterns, including triangle, square and circle or scanning of a marked area. Using these templates, the desired flight path can be planned literally with a few mouse clicks. However, the option of setting only 16 waypoints for the Naza-M, Naza-M V2 and Phantom 2 units is somewhat limiting. On the other hand, the WKM and A2 units can manage 50 waypoints, whilst the Ace unit can handle up to 200 waypoints. The station uses Google maps as its basis. They can be saved in the local storage and the station can therefore be used even in the offline mode.

Yet another software product is the UgCS (Universal ground Control Software) application. Due to the closed platform, DJI drones must be connected to the UgCS ground station via the UgCS for DJI application installed on an Android device [5]. Drones made by other manufacturers are connected directly using radio modules. The application allows to transfer a pre-planned mission to a drone, display telemetry data or replay the mission in the telemetry player.

An equally interesting application intended for controlling and setting parameters of UAV is the Mission Planner open-source product. Unlike the previous ones, this software is intended exclusively for the Windows system as well as ArduPilot, Pixhawk, Px4 control units and others compatible with the MAVLink protocol [6]. In addition to drones, it can be used with airplanes and ground vehicles as well. It has been developed by several core developers and a very large community of other contributors. This application also allows to plan an air mission on a map basis with a simple mouse click and choose how a drone will react at each specified point. With connected telemetry modules, it can provide its users with information about the state of UAV during and after their mission, log files for download and further analyses. It can be used to automatically download and upload the latest autopilot firmware and set up, configure and "tune" UAV to optimal performance.

## 3. Hardware Required for Automatic Flights

In addition to control units and devices with an installed ground station, at least two other modules are required to ensure automatic flights, navigation or even driving. The first of them is a GPS module, with the help of which it is possible to determine the geographical position of a receiver located anywhere on the Earth with an accuracy of units of meters. The second one is a telemetry module, by means of which the ground station communicates with the controlled device and enables to obtain current information about the status of the device or send commands to it. In this respect, the position of the controlled device can be monitored on a map basis, when the control unit processes information from the GPS module and sends it through the telemetry module to the ground station, which then displays it.

Units that can be used to pre-program a route to a ground station involve A3/A3 PRO, N3, Naza-M v2 or Wookong-M as well as Phantom, Inspire and Matrix platforms or ArduPilot, Pixhawk, Px4, Mikrokopter and others [7]. While, for example, Naza-M v2 is intended exclusively for multicopters (it is therefore suitable for building one´s own drone on the DJI platform), the ArduPilot control unit can be used for various devices after uploading appropriate firmware [8].

## 4. Software Used for Application Development

### 4.1 Development Environment

The discussed application was developed using Microsoft Visual Studio Community Edition, which is free full-featured Integrated Development Environment (IDE) for students, open-source developers and individual developers, and which is available for download from the Microsoft website. It can be used to create applications for Android, iOS, Windows, web and cloud. Also included in the application is the C# (C Sharp) programming language, a high-level object-oriented programming language developed by Microsoft, which can be used to create database programs, web applications

and pages, web services, form applications in Windows or software for mobile devices (tablets and mobile phones) [9].

### 4.2 MAVLink Communication Protocol

Communication between both elements of the telemetry module is ensured by the MAVLink protocol. It was originally developed in 2009 to transmit parameters required to control flights of unmanned aircraft, and is currently widely used by a number of open-source autopilots, including APM.

It is licensed under the Lesser General Public License (LGPL) courtesy of the Free Software Foundation, which means that an application that is compiled with linking of the MAVLink library is not considered a derivative work of the MAVLink library [6,9,10]. Thus, the protocol can be used without restrictions in any open-source or proprietary application.

## 5. Programming Own Waypoints Application

This section comprises instructions for programming the considered application that was simply called Waypoints. Due to the assumed use of this application in the theoretical part of the research and in order to understand principles of communication of UAV with ground stations, the application is not primarily intended for real use. All codes are executed directly in the events of the respective control units. The application is programmed to be used with the ArduPilot control unit that supports the MAVLink protocol.

### 5.1 MAVLink Protocol

Micro Aerial Vehicle Link (MAVLink) is a protocol used by ArduPilot to communicate with a particular ground station (GCS). It is stateless consisting of defined messages, enumeration types and a set of prescribed and service functions. The actual transmission takes place over a serial line using USB or wirelessly using a 3DR radio module. It defines a set of messages, including messages for device status, command entry, parameter setting and telemetry transmission messages, and also contains a variety of messages, of which some are common to all devices, whereas some are specific and applicable only to a particular type (land vehicle, ship, aircraft, multi-engine system) [11].

Furthermore, each message is typically encoded and sent from the GCS in a computer or a tablet to a given device. The control computer in the device decodes it, activates message handling events and takes appropriate actions. It may or may not follow a reply back to the ground station. The communication works both ways. ArduPilot automatically generates a number of messages and sends them (e.g. telemetry data) to the ground station. The messages are distinguished through an identification number, the so-called Message ID, as defined by the protocol [12].

Moreover, each message contains a header (6 bytes), a body and a checksum (2 bytes). The minimum message length is 8 bytes (without data), whilst the maximum one is 263 bytes. Regarding the header, one should draw attention to the System ID, which represents an identification of the source/message recipient, the Component ID, which represents an identification of the subsystem as the source/message recipient, and the Message ID as an identification of the message type. The data part changes according to the Message ID and it is exactly the part containing the data required.

In simple terms, data reading works in the following manner: according to the System ID and the Component ID, the software determines whether a particular message sent by ArduPilot is intended for it or not [13]. If not, it discards the message. If the message is valid, the data are extracted from the message according to the Message ID into a packet of the relevant data structure, from which one can already read the transmitted parameters according to the data type. This also works exactly in the same manner in the reverse direction.

Additionally, messages are generated by service functions created with the use of an XML file. To illustrate this on an example, the authors shall select the GLOBAL_POSITION_INT message, which offers latitude, longitude, altitude, relative altitude, velocity relative to the ground and compass direction calculated from GPS and accelerometers [14]. The Message ID is 33, its total size is 28 bytes and it belongs to the MAVLINK_MSG_ID message type. Then, its XML definition is as follows:

```
public struct mavlink_global_position_int_t
{
public  uint time_boot_ms; (number of milliseconds since system power on)
public  int lat; (latitude [°])
public  int lon; (longitude [°])
public  int alt; (absolute altitude [m])
public  int relative_alt; (relative altitude above the ground [m])
public  short vx; (velocity relative to the ground in x-axis (ground length) [m/s])
public  short vy; (velocity relative to the ground in y-axis (ground latitude) [m/s])
public  short vz; (velocity relative to the ground in z-axis (altitude) [m/s])
public  ushort hdg; (rotation angle of the front of the drone [°])
};
```

The MAVLink protocol contains a large number of messages that are constantly being developed and complemented, or on the contrary, removed and replaced by newer ones. To understand the protocol options and to allow for possible programming of one´s own application, it is necessary to study it in detail. There are a considerable number of sources on the Internet describing MAVLink's capabilities in great detail.

### 5.2 Application Method Description – ReadData Method

In this section, the main function of a selected method in the application shall be described in order to understand operations in the MAVLink protocol. There is no need to list all of them here, since dealing with other messages is similar, but they only differ in their parameters. It should be noted that

it is not necessary to know the MAVLink message encoding and decoding mechanism, as the MAVLink library added to Visual Studio will provide that [15].

Specifically, the method concerned is the "readData" method. As its name suggests, it is used to read data from the serial port. The input parameters are SystemID, ComponentID, timeout fixed at 4500 ms (APM has a maximum limit of 5000 ms) and the MAVLink message type, hence e.g. MAVLink.mavlink_global_position_int_t. [16]

### 5.3 Navigation Commands

At this point, one can already read data that the APM autopilot sends, and can also send the APM a command to perform an action desired. Thus, it is possible to capitalize on this knowledge and use it to program the drone's flight path according to specified waypoints, which is done by the MAV_CMD_NAV navigation commands. Of all available commands, only three (TAKEOFF, WAYPOINT and RETURN_TO_LAUNCH) are to be used here. They are sufficient for the drone to take off, fly the specified route and land at the take-off point.

The TAKEOFF command, which should also be the first command of every mission, is used for the drone to automatically take off from the ground or from one´s hand to specified altitude. Before the actual take-off It was necessary to activate the drone to the ArduCopter firmware version 3.3, set its engine power with the use of transmitter controls and switch it to the automatic mode. As regards the ArduCopter version 3.3, a new MISSION_START command is available, which allows an autonomous mission to be started only by switching the drone to the AUTO mode without having to manipulate the remote control in any way [17].

The WAYPOINT command is intended for the drone to fly along a straight path to set coordinates at specified altitude. In addition to the geographical coordinates and the altitude, the DELAY parameter can be set as well, which causes the drone to wait for a specified number of seconds (the maximum is 65535) at the reached waypoint. There is also the SPLINE_WAYPOINT command used when the flight path between individual waypoints is not straight, but rounded so that it is smoothly followed.

The last command used is RETURN_TO_LAUNCH. By executing the command, the drone being at the specified altitude will return to its initial position, i.e. where it was activated and where it saved the first GPS position after launch. It will land on its own at that location, turn off its engines and end the mission. This should be the last command of every mission.

Furthermore, a brief theoretical overview is required to understand the use of navigation commands to plan a mission in the automatic mode. While reading waypoints stored in the APM unit, the software first sends the MISSION_REQUEST_LIST message to the control unit, which requests a list of all stored waypoints. The control unit responds to this message by sending the total number

of waypoints in the MISSION_COUNT message. Then, the software requests information about each waypoint with the MISSION_REQUEST message, and the control unit sends it with the MISSION_ITEM message. Individual waypoints are numbered from zero, with waypoint (0) representing HOME, i.e. the position where the drone was turned on and where it will return to on the RETURN_TO_LAUNCH command. Having received the last waypoint, the software sends the MISSION_ACK confirmation packet to the control unit. The entire procedure is shown in Figure 1 where the word waypoint is used instead of the word mission, since the picture is taken from the QGroundControl website using own implementation of the MAVLink protocol [18].

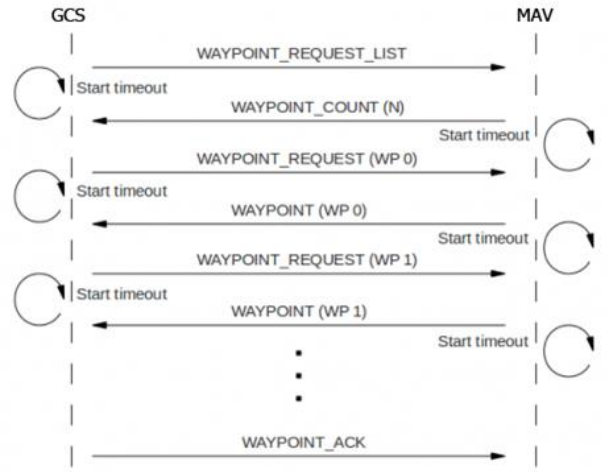



**Fig. 1** Procedure for reading waypoints stored in control unit. Source: Mavlink waypoint protocol

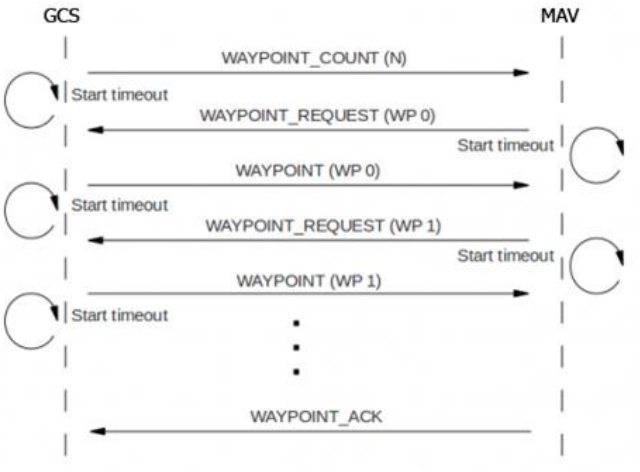



**Fig. 2** Procedure for writing waypoints to control unit. Source: Mavlink waypoint protocol

Correspondingly, all waypoints are recorded (written) in the control unit in a similar way. The program sends the APM the MISSION_COUNT (N) message, where N represents the number of waypoints being sent. The control unit prepares for their receipt and gradually requests one waypoint after another with the MISSION_REQUEST message, starting with the number zero as in the case of reading. After successfully receiving the last waypoint, the control unit sends the MISSION_ACK confirmation and thus ends the transmission. The entire procedure is shown in Figure 2.

The program uses the btnReadWP method to read waypoints from the control unit and the btnWriteWP method to write them. Although they are the program´s crucial parts, their detailed description shall not be given here since their statement would be too long. Moreover, they are sufficiently commented in the source code. The principle of sending and receiving messages is still the same, only their content changes.

## 6. Functions of Waypoints Application

The Waypoints application is designed to be connected to the APM autopilot. The main functions include downloading points stored in APM, uploading new waypoints to the autopilot, saving points

to a text file and reloading them into the application. The application is written for the Windows system, and in order to run the program it is required to have the .NET Framework v4.6.1 installed on one´s computer [19].

### 6.1 Connection to Autopilot

Connection to the APM autopilot is done in the Connection section by selecting a serial port and connection speed. The program will automatically detect all available serial ports and display them after clicking on the first drop-down menu. With the port having been selected, connection speed in the second drop-down menu is then to be set. Since there are essentially two connection methods, i.e. using either a USB cable directly to the autopilot or a 3DR radio telemetry module, there are merely two options to choose from.

Provided that the autopilot is connected and serial line parameters are correctly selected, the application will try to load three HEARTBEAT messages (which is also indicated by the progress bar in the status bar) that the autopilot automatically sends when it is turned on, and only after that the application is connected to the autopilot. As in the previous case, the user is informed about the connection process [20]. In the status bar, the Disconnected text changes to Connected, and information about the battery status and the number of visible satellites is displayed. For practice reasons, the current position of the drone is not automatically filled in, although it would be done so in the application´s full ready to use version. After turning on the drone, one is to wait a short while for the GPS position to be fixed and then upload particular coordinates to the program by pressing the Home position link in the application section of the same name.

### 6.2 Inserting Waypoints

Inserting waypoints manually or loading them from a file works even without connected autopilot. Latitude is entered in the Latitude field and longitude in the Longitude field (both in the Google format, decimal system). Altitude, being relative given the Home position, is selected in the Altitude field. By pressing the Add WP button an individual waypoint is saved to a table and is automatically assigned the WAYPOINT type when inserted. Regarding an automatic mission, the first sequence command should be TAKEOFF and the last RETURN_TO_LAUNCH. However, it is not necessary, as all points can be of the WAYPOINT type, but only in that case it will be necessary for the drone to take off and land manually. Commands can be set in the drop-down menu when planning a mission. After inserting the first point, the Save WPs button is activated, with which the waypoints can be saved to a file.

### 6.3 Loading/Writing Waypoints from/to Autopilot

After successful connection to the drone´s control unit, stored data can be read from it or created data can be written into it. Loading the mission saved in the autopilot is done by clicking the Read WPs button. The progress bar in the status bar informs the user about the loading process and the uploaded waypoints are gradually displayed in a table. If some data have already been displayed in the table, the program informs the user that such data will be deleted. Finally, a dialog box is displayed to the user with information about an option to set the HOME position coordinates. When clicking the No button, coordinates stored in the currently uploaded mission will be displayed in the Home position field. With the Yes button, these coordinates will be replaced by coordinates of the position in which the drone was first activated (turned on). This can be convenient if one wishes to start and end the current mission in a different location. The Home position coordinates can be updated at any time by pressing the Home position link [21].

Saving waypoints to the autopilot is done by pressing the Write WPs button. If there are no data in the table, the application will notify one of that. Prior to writing data to the control unit, the program performs several checks and settings of the data entered. If the TAKEOFF command is selected, the application will set both geographic coordinates to zero. This will ensure that the drone takes off from the current position. The same will be done in relation to RETURN_TO_LAUNCH, with altitude being set to zero as well. Having flown through the last waypoint, the drone will therefore return to the Home position, land, turn off its engines and end the mission. As for the standard waypoints, the program checks the minimum altitude set at 10 meters in the program. However, if the altitude is lower, the program will not allow writing and the altitude must be changed. It should also be noted that this altitude is relative to the Home position altitude and must therefore be entered taking into account the mission area elevation. Then, the coordinate format is checked. If there are no errors, the mission is ready to be written.

### 6.4 Saving/Loading Waypoints to/from File

Saving waypoints to a file is done by clicking the Save WPs button in the standard manner that Windows users are used to. In a user dialog box, a path and a file name are selected and the Save button is pressed. Data are saved in a regular text file that can be edited in other applications as well, e.g. Notepad in Windows. The application saves not only the waypoints entered, but also the Home position. In the case of a completely new data set, it stores zero values in the Home position. If one wishes to save the current position of the drone, it is necessary to activate it, wait for GPS coordinates to be fixed, load them by pressing the Home position link, and only then save the file.

## 7. Conclusion

At the beginning of the paper, listed are several of the large number of available applications used to control UAV. Each of them has its pros and cons and it is impossible to state which one is the best. Additionally, it is important to select a software platform for which the given program is designed, and indeed a hardware platform, which in this case involves UAV with different control units. This also determines a range of use the application will have. The answer to the question of using software is therefore not, and cannot be, completely unambiguous.

Using the aforementioned simple example in the paper related to the programmed Waypoints application for the ArduPilot Mega autopilot, it was shown that communication between one´s own software and UAV using the MAVLink protocol, which was developed specifically for this purpose, is not a complicated matter. In a few hundred lines of the program code, the basics of controlling flights of UAV and completing missions with the use of predefined waypoints were illustrated.

It is possible to read web presentations of manufacturers, discussions and opinions of users or browse contribution forums. However, no such actualities can replace and balance the benefit of practical testing with the use of the given program and its options. There is no other recommendation for software selection than to realistically try out several possible applications that can be used with one´s own device and choose one according to personal discretion and needs and on the basis of expected functionalities.

In order to be able to program own application to control not only a drone, but basically any other device, certain knowledge of programming is necessary [22]. The Waypoints application, as it is written in its current form (and mentioned in the paper), should not be used in heavy traffic (nor was it intended to be), since it does not contain any safety features, e.g. setting the FAILSAFE mode safe return or drone landing, etc. Moreover, it is written in the still used and yet already outdated concept of Windows Forms and with the absence of all modern methods of today´s programming. For the sake of easy understanding along with no necessity to jump from place to place in the code, as well as for its integrity, all of it was inserted directly into button handlers, and some parts are thus repeated. However, such code would become confusing and difficult to maintain when expanding its functions.

The current programming uses the modern Windows Presentation Foundation (WPF) framework where all codes are enclosed in classes for better maintainability and clarity. Calculations, writing to files, databases, etc. are performed in the classes. On the contrary, event error displays are performed outside the classes in forms that communicate with the user. When regarding its present version, the Waypoints program was intended to be as simple a guide as possible for those who would like to experiment and write their own program. For a more serious use, however, it would be necessary to write the application in the spirit of today´s modern programming methods. In order to sufficiently

understand the MAVLink protocol and use it at its maximum, the authors recommend that its complete documentation, which is presented in abundance on the Internet, is thoroughly studied.

**Acknowledgments**

This publication was realized with support of the Operational Programme Integrated Infrastructure in frame of the project: Intelligent systems for UAV real-time operation and data processing, code ITMS2014+: 313011V422 and co-financed by the European Regional Development Found.

**References**

[1] Karas, J. & Tichý, T. (2016). Drones (in Czech). Brno: Computer Press. ISBN 978-80-251-4680-4.

[2] Austin, R. (2010). Unmanned Aircraft Systems: Uavs Design, Development and Deployment. ISBN: 978-0-470-05819-0.

[3] Fahlstrom, P.G. & Gleason, T.J. (2012). Introduction to UAV Systems: Fourth Edition. DOI: 10.1002/9781118396780.

[4] Peter, P. (2012). Implementing an unmanned aerial vehicle (UAV) for surveillance tasks. Saarbrücken. Bachelor thesis. Saarland University.

[5] ArduPilot. MAVLink Commands. (2016). MAVLink Interface. Retrieved 2022 from http://ardupilot.org/dev/docs/mavlink-commands.html

[6] Fuller, B., Kok, J., Kelson, N. & Gonzalez, L. (2014). Hardware Design and Implementation of a MAVLink Interface for an FPGA-Based Autonomous UAV Flight Control System. Retrieved 2022 from https://www.researchgate.net/publication/303450554_Hardware_Design_and_Implementation_of_a_MAVLink_Interface_for_an_FPGA-Based_Autonomous_UAV_Flight_Control_System/citation/download

[7] ArduPilot. MAVLink Step by Step. (2016, February). MAVLink Step by Step. Retrieved 2022 from https://discuss.ardupilot.org/t/mavlink-step-by-step/9629

[8] DJI. GS Pro User Manual v2.0. (2017). DJI. GS Pro User Manual v2.0. Retrieved 2022 from https://dl.djicdn.com/downloads/groundstation_pro/20171121/GS_Pro_User_Manual_EN_V2.0.pdf

[9] Kwon, Y.-M., Yu, J., Cho, B-M., Eun, Y. & Park, K-J. (2018). Empirical Analysis of MAVLink Protocol Vulnerability for Unmanned Aerial Vehicles. IEEE, 1-1. DOI: 10.1109/access.2018.2863237.

[10] DJI. Naza-M V2. (2017). Compact designed multirotor autopilot system. Retrieved 2022 from https://www.dji.com/naza-m-v2

[11] DJI. PC Ground Station. (2017). PC Ground Station. Retrieved 2022 from https://www.dji.com/pc-ground-station

[12] Dronecode. MAVLink Developer Guide. (2017). MAVLink Developer Guide. Retrieved 2022 from https://mavlink.io/en/

[13] Domin, K., Marin, E. & Symeonidis, I. (2016). Security Analysis of the Drone Communication Protocol: Fuzzing the MAVLink protocol. Proc. Symposium on Information Theory in the Benelux, Louvain-la-Neuve, Belgium, 2016 (pp. 198–204).

[14] Liu, Q., Wang, H., Sun, Y. & Han, T. (2020). A Multi-UAVs Communication Network Simulation Platform using OPNET Modeler, 1-6. DOI: 10.1109/ICC40277.2020.9149416.

[15] Dronecode. QGroundControl User Guide. (2018) QGroundControl User Guide. Retrieved 2022 from https://docs.qgroundcontrol.com/en/

[16] GMap.NET Beginners Tutorial. (2016). Independent Software. GMap.NET Beginners Tutorial. Technical Articles by Independent Software in Category C#. Retrieved 2022 from http://www.independent-software.com/category/c/

[17] LibrePilot. (2016) LibrePilot. Retrieved 2022 from http://www.librepilot.org/site/index.html

[18] Mission Planner. (2017). Mission Planner Home. Retrieved 2022 from http://ardupilot.org/planner/index.html

[19] MAVLink Micro Air Vehicle Communication Protocol (2018). MAVLink Micro Air Vehicle Communication Protocol. QGroundControl. Retrieved 2022 from http://qgroundcontrol.org/mavlink/start

[20] UgCS. (2017). UgCS Desktop & UgCS for DJI App. Retrieved 2022 from https://www.ugcs.com/en

[21] U-blox. NEO-M8 series. (2018). U-blox. NEO-M8 series. Retrieved 2022 from https://www.u-blox.com/en/product/neo-m8-series

[22] GMap.NET Beginners Tutorial. (2016). GMap.NET Beginners Tutorial. Independent Software. Technical Articles by Independent Software in Category C#. 31. Retrieved 2022 from http://www.independent-software.com/category/c/