ІНФОКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ ТА ЕЛЕКТРОННА ІНЖЕНЕРІЯ
INFORMATION AND COMMUNICATION TECHNOLOGIES, ELECTRONIC ENGINEERING

№ 5 (2), 2025

https://doi.org/10.23939/ictee2025.02.

INVESTIGATION OF UNMANNED AIRCRAFT AUTOPILOTING METHODS WITH REAL TIME ROUTE CORRECTION

I. Bernevek [ORCID: 0009-0000-2122-0545], **O. Yaremko** [SCOPUS ID: 24484282800]

Lviv Polytechnic National University, 12, S. Bandery str., Lviv, 79013, Ukraine

Corresponding author: I. Bernevek (e-mail: ivan.a.bernevek@lpnu.ua).

(Received 16 July 2025)

The article explores the prospects for the automatic control of unmanned aerial vehicles (UAVs) implementation, analyzes the results of modern research and existing implementations in this area. Special attention is paid to popular software packages for their ability to provide automatic piloting functions without any need to equip the UAV with additional modules, and also considers the features of such implementation when using third-party equipment and software. A test algorithm for extended automatic piloting of the UAV has been developed for the ArduPilot + Companion Computer assembly, which involves piloting the UAV along a given route, performing additional operations on the UAV at certain points on the route, and correcting the route in real time if necessary. The peculiarities of connecting the Companion Computer to the flight controller, using the MAVLink protocol for ArduPilot, and using the appropriate libraries for programming languages, in particular Pymavlink (mavgen) for Python, are analyzed. The corresponding stages of the test algorithm are implemented using the Python language and the Pymavlink library, in particular the stage of establishing a connection via the MAVLink protocol, receiving the coordinates of the current position from the flight controller, calculating the distance to the next route point, dynamically changing the route by setting the coordinates of the next route point, performing additional operations at certain points of the route, as well as collecting and processing UAV telemetry information. The ArduPilot + Companion Computer assembly allows you to significantly expand the functionality of the UAV and dynamically change them, however, the use of such assemblies is advisable only in cases that cannot be covered by the computational capabilities of the flight controller and standard UAV software. Compared to other software packages, ArduPilot provides the best functionality for implementing automatic piloting, both using the ArduPilot Mission Planer and in the ArduPilot + Companion Computer assembly and third-party software.

Keywords: *unmanned aircrafts, autopiloting, flight by route, ArduPilot.*
UDC: 621.126

Introduction

The functionality of any UAV is defined by its physical characteristics, flight controller and software used for control. Changing the physical characteristics of a UAV during its use is a rather resource-intensive process, whereas changing the flight controller and updating the software can significantly expand the functional and quality characteristics of the UAV with little or even no physical modification. The presence of certain functions is the key criterion for choosing software. There are a large

number of software packages for controlling UAVs. The most popular today are Betaflight, iNav and ArduPilot. In this article, using these packages as an example, we will consider the possibilities of implementing the UAV automatic piloting function within the software package itself, as well as using auxiliary modules (Companion Computer) and third-party software.

2. Recent research and existing implementation analysis

Every year, the uses of UAVs, especially automatically piloted ones, are expanding. As the range of applications grows, so do the requirements for the functional capabilities of these UAVs. Today, there is active research in the field of automatic piloting. In particular, study [1] considers the possibility of simultaneously controlling several UAVs by shifting each one a certain distance relative to the others. To implement this method, the additional Arduino modules were used to monitor the flight in real time and manage the shifting function. Study [2] explores using an automatic piloting function for UAVs in the event of a lost radio control signal. This method was implemented using an additional control module based on a Raspberry Pi, which processes the lost signal messages. Studies [3, 4] demonstrate the use of computer vision for organizing automatic piloting of UAVs in a pursuit mode. As in the previous cases, this approach requires external information processing and control modules. Study [5] presents the process of implementing a UAV autopilot function for the Betaflight software package, which in its standard version only supports a return-to-home function. The implementation used an additional Raspberry Pi module to analyze flight data and perform all UAV control functions. The results of these studies demonstrate the implementation of UAV autopilot functions for various situations, usually for flights without a predefined route. These solutions are based on the use of additional control and information processing modules, which complicates their implementation.

The implementation of UAV autopilot functions without additional modules is limited by the capabilities of the flight controller and the UAV's software. The most popular software packages used for UAVs today are Betaflight, iNav, and ArduPilot.

Betaflight is an open source flight controller software package designed primarily for multi-rotor aircraft, including quadcopters. It is by far the most popular choice for FPV drone operators, known for its wide range of features and performance-oriented design. Betaflight offers an intuitive interface and supports a wide range of hardware. An active community and ongoing development ensure regular updates introducing new features and improvements. With its high-performance customization capabilities and constant innovation, this software package is most suitable for racing and aerobatic flights. It is also suitable for long-distance flights, thanks to the GPS Rescue function, which provides automatic return to a point with given coordinates in case of loss of communication [6].

iNav is a package that focuses on GPS navigation and autonomous flight. In addition to multicopters, iNav can also be used for fixed-wing aircraft and radio-controlled cars. It offers more advanced GPS features such as waypoint navigation, return-to-home, and altitude hold. Like Betaflight, iNav is an open-source software package that is constantly updated and improved by the development community. Although iNav is not as popular for racing or aerobatic flying as Betaflight, it is a popular choice for long-range fixed-wing aircraft and autonomous applications [7].

ArduPilot is perhaps the most popular and advanced open source autopilot software package. It supports a variety of vehicles, including quadcopters, aircraft, land vehicles, and even radio-controlled submarines. ArduPilot is known for its extensive feature set and customization options, making it a good choice for experienced operators and developers. It supports both autonomous and manual control modes, as well as GPS waypoint navigation. However, the wide range of features and capabilities comes at the cost of the product's complexity [8].

The features of the above-mentioned software packages are summarized in Table 1.

Table 1

UAV software package options

Option	Betaflight	iNav	ArduPilot
Main purpose and used areas	Racing / aerobatic flights	Cruise / long-duration flights, hobby UAVs	Universal platform for UAVs, cars, boats. Industry, agriculture, mapping, military drones, autonomous platforms
Autopilot	Limited (RTH)	Basic autopilot (Loiter, RTH, Waypoints)	Full autopilot (RTH, Loiter, Mission, Follow Me etc.)
Route planning	Not supported	Limited	Fully functional Mission Planner
Support for additional modules	Camera, OSD, GPS (limited)	GPS, compass, barometer, telemetry	GPS, compass, telemetry, Lidar, camera, gimbal, ADS-B, LTE
User interface (configuration)	BetaFlight Configurator (user friendly, simple)	iNav Configurator (similar to BetaFlight)	Mission Planner, QGroundControl (complicated, fully functional)

3. Basic UAV autopiloting implementation

If we talk about manual piloting, then each of the above-mentioned software packages fully covers these needs. However, if there is a need to use autopiloting and other automated functions to perform tasks, then not each of them can provide such capabilities by default. It is also important to note that autopiloting involves the use of positioning systems.

Betaflight supports altitude hold and return-to-home functions by default, but does not support waypoint flight planning. Unlike Betaflight, iNav and ArduPilot are more advanced in this regard.

For automatic piloting, the iNav software package uses Mission Control – an extension of the iNav configurator which acts as a ground station. It allows users to create, load, and manage missions, such as waypoint flights. The process for using this function involves several steps: first, a mission must be created (defining the coordinates of waypoints, altitude, speed, and other mission options); second, the mission is loaded into the flight controller; and finally, after an initial takeoff, the mission is launched for execution. Mission Control can also be used to monitor the mission's execution. For this purpose, an additional radio channel is used for the UAV to transmit telemetry information back to Mission Control. Mission Control does not provide a way to dynamically change a mission while it is being executed. In such cases, it is necessary to create and load a new mission and launch its execution.

Similar to iNav, ArduPilot also allows for mission execution. For this purpose, the ArduPilot Mission Planner is used. Compared to iNav, ArduPilot allows for more detailed configuration of both mission and UAV parameters, which provides increased functionality. Another key feature is the ability to dynamically change the mission without needing a reboot, as well as the ability to analyze log files to understand the system's performance and correctness. The expanded list of ArduPilot functions naturally places increased requirements on flight controllers and also increases the complexity of setting up and using the system

According to the publications analyzed, another possible option for autonomous UAV control is the use of a companion computer. This computer can form a flight route based on data received from the UAV's sensors and cameras.

For communication between the flight controller and the external computer, two main protocols are used: MAVLink – supported by ArduPilot, with limited support in Betaflight and iNav; and MSP – supported by iNav and Betaflight, with limited support in ArduPilot.

When using a companion computer with Betaflight or iNav, the functionality depends on the communication protocol used. With MAVLink, its use is generally limited to reading and processing telemetry data. Betaflight does not support UAV control commands via MAVLink, while iNav supports both telemetry and control commands [7, 9, 10]. Regarding the MSP protocol, recent updates allow for the development of autopilot solutions using a companion computer with Betaflight [5]. However, these solutions currently do not integrate with existing flight planning tools like Betaflight or iNav (Mission Control) and require the implementation of a separate control system.

Compared to Betaflight and iNav, ArduPilot has limited support for MSP, using it only to receive telemetry information. However, it can fully utilize MAVLink, providing complete two-way interaction and the ability to integrate with existing flight planning tools like the ArduPilot Mission Planner [11].

4. Using ArduPilot + Companion Computer to implement advanced UAV autopiloting

Based on a preliminary analysis of recent research and implementations, it can be concluded that the basic implementation of UAV automatic piloting is possible without additional computing modules. This is achieved through the capabilities of the flight controller and its software. In cases where the requirements for automatic piloting exceed the capabilities of the flight controller or its software, using a companion computer is recommended.

Considering the advantages of ArduPilot, let's consider an example of the operation of the ArduPilot + Companion Computer assembly using the example of performing an autopilot task with performing additional operations at certain waypoints along the route, and correcting the route if necessary.

Task:

- fly over a certain area, according to the principle of a mapping mission using a predefined flight route;
- at the predefined waypoints, take a photo of the area from the current altitude;
- analyze the image for the presence of suspicious objects (transport, people, fires, etc.);
- in case of detection – move in the direction of these objects and take additional pictures, analyze again and send the conclusions to the monitoring;
- continue the flight along the route.

The algorithm of UAV operation in this configuration is presented in Fig. 1.

To perform resource-intensive computing operations, this configuration uses a Companion Computer – a separate on-board computer connected to the flight controller, which provides additional operations that are not provided by the flight controller. The Companion Computer usually runs Linux, as it is a universal platform for software development. Connection to the flight controller is carried out using a fast serial or Ethernet connection and using the MAVLink protocol. The Raspberry Pi mini computer is used as the Companion Computer. Connection and use are carried out in accordance with the official ArduPilot recommendations and the use of recommended software [12]. The MAVLink protocol is used to ensure communication between the Companion Computer and the flight controller, in particular commands for working with missions [13]. To work with the MAVLink protocol, both official and external libraries for various programming languages are available. The choice of programming language, as well as the library, will depend on the functionality and speed that must be obtained [14]. For the current task, we will use the Pymavlink library for python [15]. The process of interaction of UAV components during the execution of the algorithm is presented in Fig. 2. The general diagram of connecting UAV components to work with the Companion Computer is shown in Fig. 3.

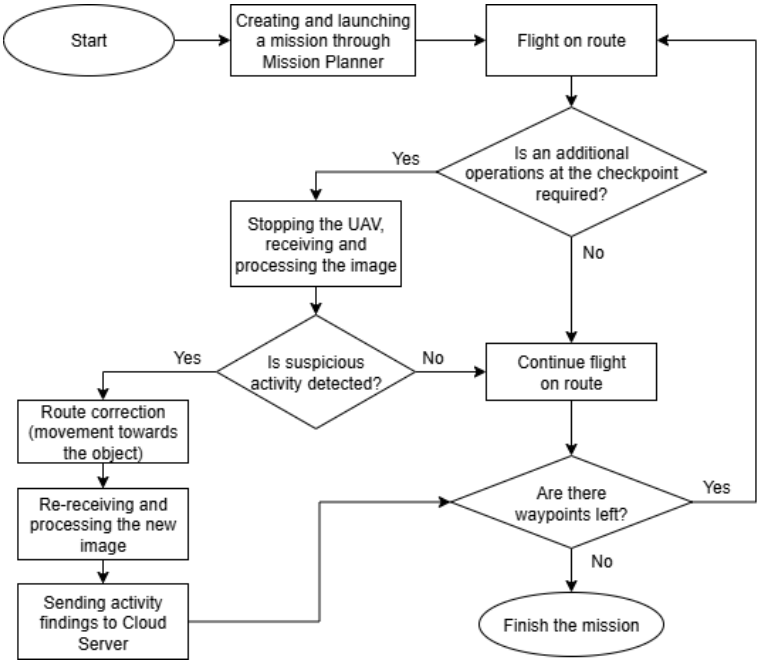

Fig. 1. Algorithm of UAV operation for mission execution in ArduPilot + Companion Computer configuration

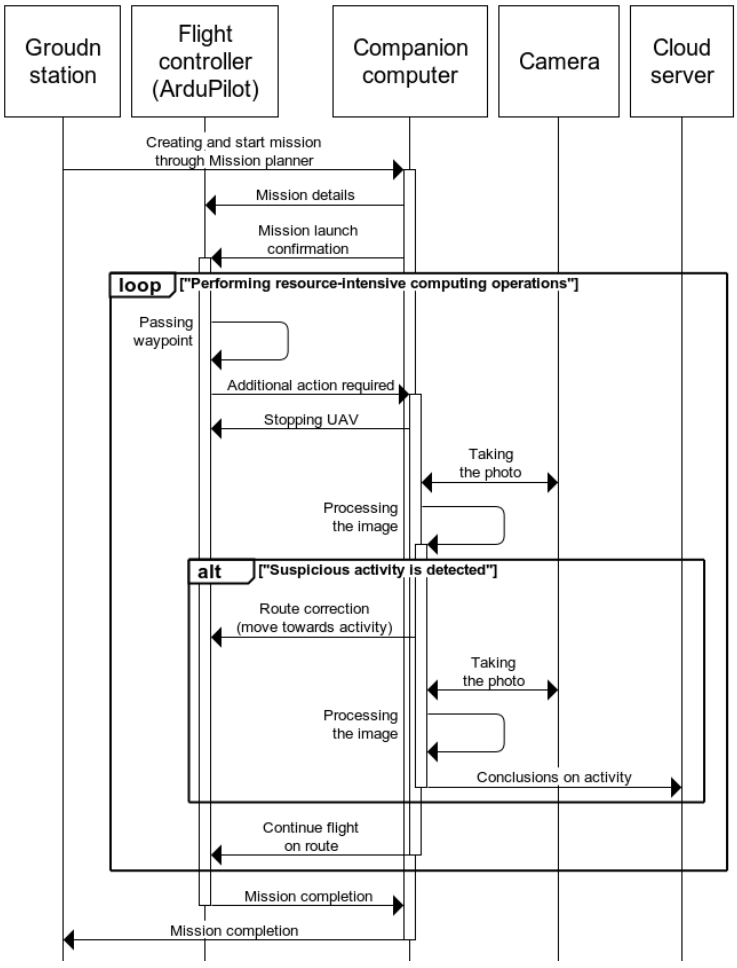

Fig. 2. UAV components interaction diagram for algorithm

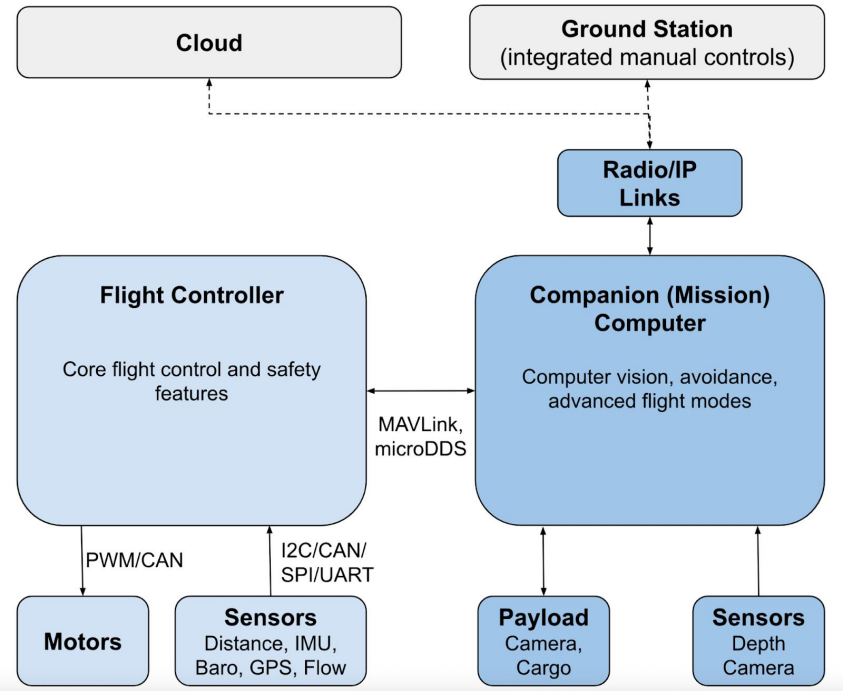

Fig. 3. UAV component connection diagram using Companion Computer

According to the constructed operating algorithm and UAV components interaction diagram, the implementation involves writing code blocks that will be responsible for performing various functions, in particular:

· Code block for establishing a connection with the flight controller (Fig. 4).

```
from pymavlink import mavutil
import time

# Connect to the drone via MAVLink
master = mavutil.mavlink_connection('/dev/ttyAMA0', baud=57600)
```

Fig. 4. Code block to establish connection to flight controller

· The function to obtain the coordinates of the current location – is used to determine the position of the UAV in the route (Fig. 5).

```
def get_position():
    msg = master.recv_match(type='GLOBAL_POSITION_INT', blocking=True, timeout=5)
    if msg:
        lat = msg.lat / 1e7
        lon = msg.lon / 1e7
        alt = msg.relative_alt / 1000.0
        return lat, lon, alt
    return None
```

Fig. 5. Code block to get the coordinates of the current location

· The function to calculate the distance to the next waypoint – is used to estimate flight time, keep a flight log, etc. (Fig. 6).

```python
def haversine(lat1, lon1, lat2, lon2):
    from math import radians, cos, sin, asin, sqrt
    R = 6371000  # Earth radius in meters
    dlat = radians(lat2 - lat1)
    dlon = radians(lon2 - lon1)
    a = sin(dlat/2)**2 + cos(radians(lat1)) * cos(radians(lat2)) * sin(dlon/2)**2
    return R * 2 * asin(sqrt(a))
```

Fig. 6. Code block to calculate the distance to the next waypoint

· The function for setting the coordinates of the next route point is used to correct the flight route or dynamically build a route (Fig. 7).

```python
def goto_waypoint(lat, lon, alt):
    master.mav.set_position_target_global_int_send(
        0, master.target_system, master.target_component,
        mavutil.mavlink.MAV_FRAME_GLOBAL_RELATIVE_ALT_INT,
        0b0000111111111000,  # Position only
        int(lat * 1e7), int(lon * 1e7), alt,
        0, 0, 0, 0, 0, 0,
        0, 0)
```

Fig. 7. Code block to set waypoint coordinates

· The code to perform a series of additional operations at defined waypoints along the route. It first determines the current location's coordinates, then takes and analyzes an image for suspicious objects. If an object is detected, the code corrects the route, moves a set distance toward the object, takes and analyzes new images, sends the results to the server, and then resumes the original route (Fig. 8).

```python
def perform_action_at_waypoint(index):
    print(f"Performing custom action at waypoint {index}...")
    pos = get_position()
    photo = take_a_photo()
    photo_processing_result = process_photo()
    send_photo_to_server(photo_processing_result)
    if photo_processing_result.additional_photo_needed
        # move toward suspicious object
        next_point = photo_processing_result.suspicious_object_position
        goto_waypoint(next_point.lat, next_point.lon, next_point.alt)
        time.sleep(5)
        # take photo in details
        photo = take_a_photo()
        photo_processing_result = process_photo()
        send_photo_to_server(photo_processing_result)
        # back to main route
        goto_waypoint(pos.lat, pos.lon, pos.alt)
```

Fig. 8. Code block to perform additional operations at defined waypoints of the route

· The overall program cycle, which includes all previous mission stages (Fig. 9).

```python
for i, (lat, lon, alt) in enumerate(waypoints):
    print(f"Going to waypoint {i+1}: {lat}, {lon}, {alt}")
    goto_waypoint(lat, lon, alt)
    # Wait until arrival
    while True:
        pos = get_position()
        if pos:
            dist = haversine(pos[0], pos[1], lat, lon)
            print(f"Distance to waypoint: {dist:.2f} meters")
            if dist < ARRIVAL_RADIUS:
                print("Arrived at waypoint.")
                break
        time.sleep(1)
    landing(i+1)
```

Fig. 9. Full cycle of algorithm execution

Testing the assembly involves the use of two configurations: the Companion Computer provides full control over the UAV operation process (guided mode) and the Companion Computer works as an auxiliary module to perform additional operations (hybrid mode).

5. Test results and conclusions

As a result of testing the assembly and analyzing logs from the flight controller and the companion computer, a slight improvement in the UAV's operation was found in guided mode compared to hybrid mode. Specifically, a decrease in the delay of the transition from flight to the execution of resource-intensive operations by an average of 120–150 ms was observed. This can be explained by the transfer of not only resource-intensive operations but also the control of their execution to the companion computer. As a result, the flight controller no longer has to monitor the performance of additional operations; instead, it only needs to send telemetry data to the companion computer and monitor control commands from it.

The transition delay change for guided and hybrid modes is presented in Fig. 10.

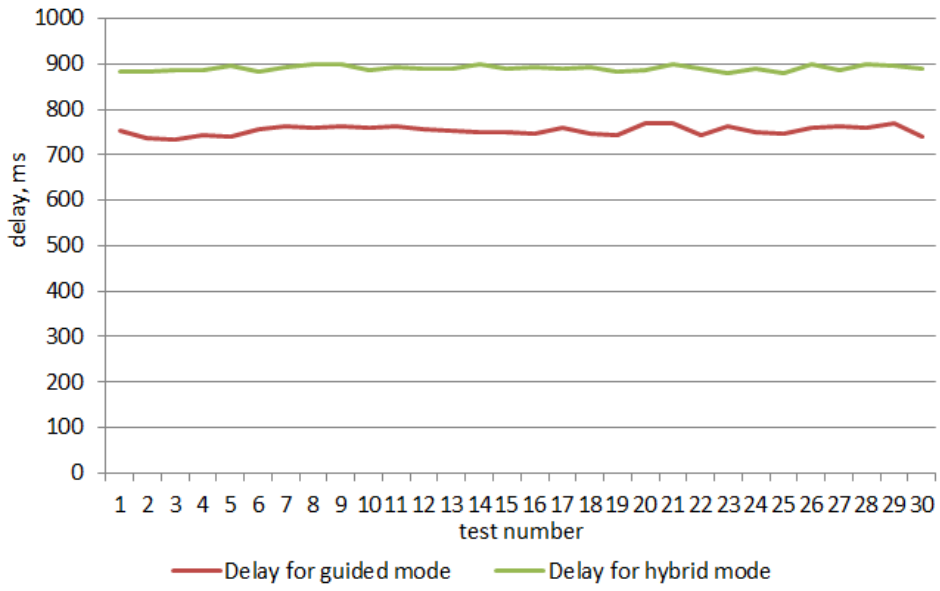

Fig. 10. The transition delay from flight to performing resource-intensive operations

Based on the results obtained, the choice between guided mode and hybrid mode depends on the mission's complexity, as well as the requirements for flexibility and reliability.

In guided mode the Companion Computer provides full control of the UAV, making it ideal for complex, flexible, and intelligent missions. Advantages: this mode offers on-board information processing, real-time decision-making, the ability to configure additional actions at specific waypoints, and integration with specialized sensors and artificial intelligence (e. g., object detection, QR code recognition). Disadvantages: it requires more development effort detailed testing and has a higher configuration complexity. Uses: autonomous inspection, precision agriculture with variable-speed actions, drone-based research, and tasks requiring visual guidance.

Hybrid ArduPilot Mission Planner + Companion Computer configuration – best suited for balanced missions that require both reliability and periodic additional operations. Advantages: preserves the reliability of ArduPilot auto mode, allowing the Companion Computer to monitor or intervene as needed (e. g., log data, trigger actions, handle exceptions). Disadvantages: more complex than ArduPilot auto mode, but safer than full ArduPilot guided missions, which are completely controlled by user code. Uses: exploration missions with post-processing, conditional actions, semi-autonomous search and rescue operations.

References

[1] Stephen Lazzaro (2015), "Flying multiple drones from 1 remote controller", available at: https://minds.wisconsin.edu/bitstream/handle/1793/72188/TR1818.pdf (Accessed 15 July 2025).

[2] Yaroslav Sheyko, Nataliia Kryvko, Oleksandr Shefer (2025), "Enhancement Arduplane Radio Failsafe Algorithm by Extending with a New Delegation Action", Electronics and Control Systems 2025. No. 1(83): 42–49. DOI: 10.18372/1990-5548.83.19873

[3] Dmytro Sazonov (2025), "How to build the Eyes of an Autopilot for FPV Combat Drone", available at: https://medium.com/illuminations-mirror/how-to-build-the-eyes-of-an-autopilot-for-fpv-combat-drone-bbf13d605a9f (Accessed 15 July 2025).

[4] Dmytro Sazonov (2025), "How to build an Autopilot with Computer Vision and Target Following for FPV Combat Drone", available at: https://ai.gopubby.com/how-to-build-an-autopilot-with-computer-vision-and-target-following-for-fpv-combat-drone-3544f482baae (Accessed 15 July 2025).

[5] Dmytro Sazonov (2025), "FPV autonomous operation with Betaflight and Raspberry Pi", available at: https://medium.com/illumination/fpv-autonomous-operation-with-betaflight-and-raspberry-pi-0caeb4b3ca69 (Accessed 15 July 2025).

[6] "Betaflight – Pushing the Limits of UAV Performance", available at: https://betaflight.com/ (Accessed 15 July 2025).

[7] "INAV Remote Management, Control and Telemetry", available at: https://github.com/iNavFlight/inav/wiki/INAV-Remote-Management%2C-Control-and-Telemetry (Accessed 15 July 2025).

[8] "Flight Controller Firmware for FPV Drone: Choosing Between Betaflight, iNav, Ardupilot", available at: https://oscarliang.com/fc-firmware/ (Accessed 15 July 2025).

[9] Dmytro Sazonov (2024), "FPV Autonomous Flight with MAVLink and Raspberry Pi. Part I", available at: https://blog.cubed.run/fpv-autonomous-flight-with-mavlink-and-raspberry-pi-part-i-f7dfa913f505 (Accessed 15 July 2025).

[10] Dmytro Sazonov (2024), "FPV autonomous flight with MAVLink and Raspberry Pi. Part II", available at: https://medium.com/illumination/fpv-autonomous-flight-with-mavlink-and-raspberry-pi-part-ii-2d55dcd8d659 (Accessed 15 July 2025).

[11] "Companion Computers – dev documentation", available at: https://ardupilot.org/dev/docs/companion-computers.html (Accessed 15 July 2025).

[12] "Communicating with Raspberry Pi via MAVLink – dev documentation", available at: https://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html (Accessed 15 July 2025).

[13] "MAVLINK common message set, MAV_CMD_NAV_WAYPOINT – dev documentation", available at: https://mavlink.io/en/messages/common.html#MAV_CMD_NAV_WAYPOINT (Accessed 15 July 2025).

[14] "Using MAVLink Libraries – dev documentation", available at: https://mavlink.io/en/getting_started/use_libraries.html (Accessed 15 July 2025).

[15] "Using Pymavlink Libraries (mavgen) – dev documentation", available at: https://mavlink.io/en/mavgen_python/#using-pymavlink-libraries-mavgen (Accessed 15 July 2025).

ДОСЛІДЖЕННЯ МЕТОДІВ АВТОМАТИЧНОГО ПІЛОТУВАННЯ БЕЗПІЛОТНИХ ЛІТАЛЬНИХ АПАРАТІВ З КОРЕКЦІЄЮ МАРШРУТУ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Іван Берневек, Олег Яремко

Національний університет "Львівська політехніка", вул. С. Бандери, 12, Львів, 79013, Україна

Досліджено перспективи впровадження автоматичного управління безпілотними літальними апаратами (БПЛА), проаналізовано результати сучасних досліджень та наявні реалізації в цій галузі. Окрему увагу приділено популярним програмним пакетам на предмет їх здатності забезпечувати функції автоматичного пілотування без потреби оснащення БПЛА додатковими модулями, а також розглянуто особливості такої реалізації із використанням стороннього обладнання та програмного забезпечення. Для поєднання ArduPilot + Companion Computer розроблено тестовий алгоритм розширеного автоматичного пілотування БПЛА, який передбачає пілотування БПЛА за заданим маршрутом, виконання БПЛА додаткових операцій у певних точках маршруту, коригування маршруту в режимі реального часу в разі необхідності. Проаналізовано особливості підключення Companion Computer до політного контроллера, використання протоколу MAVLink для ArduPilot та відповідних бібіліотек для мов програмування, зокрема Pymavlink (mavgen) для Python. Відповідні етапи тестового алгоритму реалізовано з використанням мови Python та бібліотеки Pymavlink, зокрема етап встановлення з'єднання через протокол MAVLink, отримання від політного контроллера координат поточної позиції, розрахунок відстані до наступної точки маршруту, динамічну зміну маршруту а допомогою встановлення координат наступної точки маршруту, виконання додаткових операцій у певних точках маршруту а також збирання та опрацювання телеметричної інформації БПЛА. Збірка ArduPilot + Companion Computer дає змогу істотно розширити функціональні можливості БПЛА та динамічно змінювати їх, проте використання таких поєднань доцільне лише у випадках, які неможливо реалізувати за рахунок обчислювальних можливостей політного контролера та стандартного програмного забезпечення БПЛА. Порівняно з іншими пакетами програмного забезпечення, ArduPilot надає найкращий функціонал для реалізації автоматичного пілотування, як з використанням ArduPilot Mission Planer, так і в поєднанніі ArduPilot + Companion Computer зі стороннім програмним забезпеченням.

Ключові слова: *безпілотні літальні апарати, автоматичне пілотування, політ за маршрутом, ArduPilot.*