TELECOMMUNICATIONS AND RADIO ENGINEERING

UDC 629.051:629.7.014-519(045)
DOI:10.18372/1990-5548.83.19873

[1]Y. I. Sheyko,
[2]N. V. Kryvko,
[3]O. V. Shefer

ENHANCEMENT ARDUPLANE RADIO FAILSAFE ALGORITHM BY EXTENDING WITH A NEW DELEGATION ACTION

[1,2]Lyceum #33" Poltava, Ukraine
[3]Department of Automation, Electronics and Telecommunications, Yuriy Kondratyuk Poltava Polytechnic National University, Poltava, Ukraine
E-mails: [1]yarik.sheiko@gmail.com, [2]kryvko.nata@gmail.com,
[3]itm.ovshefer@nupp.edu.ua ORCID 0000-0002-3415-349X

Abstract*—The article deals with the problem of the absence of a radio failsafe mechanism in the unmanned aerial vehicles' ArduPilot firmware that delegates the flight control management from manual mode to the onboard companion computer once the radio transmitter signal is lost due to natural reasons or caused by military-grade artificial radio interference. Algorithms for detecting signal loss and switching control to onboard computers are proposed. The proposed algorithms require knowledge adjustments and enhancements in three components: MAVlink protocol as a transport between parties, ArduPilot firmware running a flight controller, and a client library encapsulating MAVlink implementation. The proposed algorithms require knowledge of the principles of flight control approaches as well as programming skills in C++, and Python languages. To solve the problem adjustments and enhancements in three components are used: MAVlink protocol as a transport between parties, ArduPilot firmware running a flight controller, and a client library encapsulating MAVlink implementation. A brand-new short-radio failsafe action is added to keep ArduPilot firmware consistent. On top of that, the proposed problem-solving approach eliminates the potential misleading of pilots compared to modifying one of the existing failsafe scenarios. The simulation and field test results are presented, validating the effectiveness of the proposed algorithms. These findings have potential applications in both civilian and military domains.*

Keywords—Algorithm; ArduPilot; MAVLink; companion computer; radio failsafe behavior; unmanned aerial vehicles.

I. INTRODUCTION

ArduPlane-based unmanned aerial vehicles (UAVs) require robust failsafe mechanisms to ensure flight continuity in the event of control signal loss. The ArduPlane firmware includes multiple failsafe strategies, but it currently lacks a built-in mechanism to automatically switch control from the remote control (RC) transmitter to a companion computer (CC) upon RC failure. This paper proposes an extension to the ArduPlane failsafe algorithm, introducing a brand-new short failsafe mode complimented by a custom MAVLink message that allows a CC to take over flight control through an extended MAVLink message protocol.

The absence of the mechanism mentioned above makes it impossible to define the moment when CC should step in the flight control in case the RC signal has been lost unexpectedly. The example of such a case might be at least(but not limited to) getting a UAV in a radio signal-denied environment. Within this area, the GNSS, as well as RC frequency bands, are jammed in the wide spectrum, so the resilience of the Frequency Hopping Spectrum Spreading (FHSS) approach cannot re-establish a reliable and durable connection. Those circumstances inevitably lead to uncontrolled UAV flight followed by crash. So, the goal of this research and development is a practical solution that involves an enhancement of ArduPlane firmware paired with software running on the onboard CC to cover the described edge case.

II. PROBLEM STATEMENT

The primary limitation in ArduPlane's existing failsafe system is its inability to delegate flight control to an onboard Companion Computer when the RC signal is lost. The available failsafe modes include **Disabled** (means ignore RC Failsafe), **Circle** (start circling above the current location with the radius defined in settings), **FBWA** (means

gliding on zero throttle), **FBWB** (means continue movement holding current altitude with the 45% throttle on the cruise speed), **ReturnToLaunch** (means attempt to return home if GNSS signal is available), **Deploy Parachute** (means activating an emergency parachute), **AutoLand** (means Return to Launch followed by Landing), and **AUTO** (mean start or continue auto-mission) [1], [2]. However, a more adaptive failsafe mechanism is needed in mission-critical applications such as autonomous surveillance or beyond-visual-line-of-sight (BVLOS) operations. The purpose of this algorithm is to detect RC failsafe within 0.3 s and notify CC about the event with the following taking the flight control over. One of the practical uses might be starting autonomous flight in GNSS-denied areas based on machine vision and AI approaches [3], [4] or performing any other pre-programmed actions upon the met criteria.

To address this issue, we propose

- Add a **new short-failsafe mode** support in the Arduplane that can be defined as a new mode among the existing ones. It helps to keep conventional modes intact and avoid pilot confusion with unexpected behavior of build-in and described in the official documentation modes [1], [2].
- A **new MAVLink** message type to notify the CC of an RC failsafe event. The essence of this message should be adapted to the mandatory requirements such as: having atomic package size, avoiding channel transmission flooding, and being delivered with a top-notch guarantee followed by another party acknowledgment.
- Enhancements in the **ArduPilot firmware** to accommodate this bi-directional transition that ensures the process's repeatability.
- Provide the example of **software on the CC's side** that supports modified MAVlink protocol.

III. PROPOSED SOLUTION

A. System Architecture

The proposed solution [5] involves the integration of a **CC**, such as a Raspberry Pi, Orange Pi, or NVIDIA Jetson, running MAVProxy, capable of receiving, generating, and sending back to flight controller(FC) commands. The new failsafe mode operates as follows

1) RC Signal Monitoring: the ArduPlane-based FC continuously monitors the RC signal strength.

2) Failsafe Trigger: when RC signal loss is detected, ArduPilot generates a new **FBWAG_NOTICE** message.

3) Companion Computer Takeover: the CC, upon receiving the failsafe message, begins sending MAVLink RC_CHANNELS_OVERRIDE or SET_ATTITUDE_TARGET messages to take control.

4) Control Transition Confirmation: the ArduPlane FC verifies control inputs from the CC and switches to the new control mode.

B. Adding a New Mavlink Message

MAVLink includes a variety of built-in message types as well as allows for custom message creation. The flight controller determines which data are relevant for the flight, which messages are sent periodically, and which are transmitted only upon request. MAVLink itself does not specify which messages to use; instead, system designers decide which messages their software will process and which ones it will send. Different flight controllers use specific MAVLink dialects, which vary in their implementation details, such as message composition or flight modes. The core of MAVLink's header-only C/C++ library consists of directories corresponding to these dialects.

We have examined some commonly used messages [6] that should be implemented in most flight controllers and Ground Control Station (GCS) and came to the conclusion that there are no messages relevant to RC throttle failsafe exist in the MALink v2 that can are guaranteed to be delivered to connected clients within the shortest possible timeframe.

A new MAVLink message FBWAG_NOTICE, is introduced to facilitate communication between the FC and the CC. The proposed message structure is as follows:

Message ID: 15001:
– uint8_t system_id (ID of the UAV)
– uint8_t component_id (ID of the component triggering failsafe)
– uint8_t failsafe_status (0 = normal, 1 = RC signal lost, 2 = CC Takeover Confirmed)
– char[5] text (custom text message)
– uint32_t timestamp (system time of event)

C. Implementation in ArduPilot

Implementing the CC Failsafe Mode in ArduPilot requires modifications in both the ArduPlane firmware and the MAVLink protocol to handle the new failsafe mechanism. Below is a step-by-step breakdown of the needed changes.

Extending the Failsafe Modes with a New One

To integrate CC takeover, let's add a new mode, FS_ACTION_SHORT_FBWAG, in the ArduPlane source code [7]:

- Definet the new mode in the file *ArduPlane/defines.h* (Fig. 1)

```
34  enum failsafe_action_short {
35      FS_ACTION_SHORT_BESTGUESS = 0,
36      FS_ACTION_SHORT_CIRCLE = 1,
37      FS_ACTION_SHORT_FBWA = 2,
38      FS_ACTION_SHORT_DISABLED = 3,
39      FS_ACTION_SHORT_FBWB = 4,
40      FS_ACTION_SHORT_FBWAG = 5,
41  };
```

Fig. 1. New FailSafe mode

- Extend the RC Failsafe triggering Logic in the file *ArduPlane/radio.cpp* (Fig. 2)

```
256         // Custom FailSafe: Delegate to CC START
257         if (plane.throttle_cnt > -1){
258             plane.throttle_cnt++;
259         }
260         if (plane.throttle_cnt >= 2
261                 && g.fs_action_short
262                     == FS_ACTION_SHORT_FBWAG) {
263             plane.throttle_cnt = -1;
264             gcs().send_fbwag_ntc(1);
265             return;
266         }
267         // Custom FailSafe: Delegate to CC END
```

Fig. 2. RC Failsafe trigger

This function detects RC loss and triggers a MAVLink message FBWAG_NOTICE broadcasting once rc_failsafe occurred, at least 2 times in a row to avoid false triggering.

Extending the MAVLinkdialect should not impact overall code performance. Hence the newly added FBWAG_NOTICE message is not supposed to be added in any of the streams that sync telemetry on a periodical basis with the preconfigured frequency rates such as

STREAM_RAW_SENSORS,
STREAM_RC_CHANNELS, or
STREAM_EXTENDED_STATUS.

FBWAG_NOTICE is to be treated rather as a rarely occurred event than a frequently changing telemetry value. This approach helps to save traffic and keep the FC's and CC's CPU load intact.

- Implement broadcasting to all connected Ground Control Stations (CGS) and CCs in the files *ArduPlane/libraries/GCS_MAVLink/GCS.h* by adding method the following definition: *void send_fbwag_notice()*; in the public section of the GCS_MAVLINK class. Thus the method implementation goes to *ArduPlane/libraries/GCS_MAVLink/GCS.cpp* (Fig. 3).

This function checks all established MAVLink channels checking if it is open public and still active, and puts the messages in each of the available channels.

- Eventually, the method of sending the message itself [8] can be added in the *ArduPlane/libraries/GCS_MAVLink/GCS_Common.cpp* (Fig. 4).

```
87  void GCS::send_fbwag_ntc(const uint8_t failsafe_status)
88  {
89      for (uint8_t i=0; i<num_gcs(); i++) {
90          GCS_MAVLINK &c = *chan(i);
91          if (c.is_private()) {
92              continue;
93          }
94          if (!c.is_active()) {
95              continue;
96          }
97          if (c.get_fbwag_notice_failsafe_status() == failsafe_status) {
98              continue;
99          }
100         if (failsafe_status == 1
101             and c.get_fbwag_notice_failsafe_status() == 2) {
102             continue;
103         }
104         c.set_fbwag_notice_failsafe_status(failsafe_status);
105         c.send_fbwag_notice();
106     }
107 }
```

Fig. 3. Broadcasting to all opened channels

```
222 void GCS_MAVLINK::send_fbwag_notice()
223 {
224     // avoid unnecessary errors being reported to user
225     if (!gcs().vehicle_initialised()) {
226         return;
227     }
228     mavlink_msg_fbwag_notice_send(chan,
229                                   mavlink_system.sysid,
230                                   mavlink_system.compid,
231                                   fbwag_notice_failsafe_status,
232                                   "FBWAG",
233                                   AP_HAL::millis()
234     );
235 }
```

Fig. 4. Composing and sending of the message

The proposed method implementation has the built-in prevention of sending false notifications during the FC initialization.

Improve Robustness, Durability, and Sustainability of Algorithm

Bearing in mind that the UAV might experience the iteration of RC signal loss and gain it back more than once, it becomes essential to create the acknowledged feedback loop between FC and CC. This will help to confirm for each party that the message has been received and successfully processed. Otherwise, a sender should repeat the attempt of sending the message till the ack (Fig. 5).

- The aforementioned implementation is to be put in *ArduPlane/libraries/GCS_MAVLink/GCS_Common.cpp* (Fig. 6).

It assumes two types of ack:

– on the moment when the RC signal is lost to receive a confirmation from CC that autopiloting has been successfully started;
– and one RC signal regained and a human pilot took control of the UAV flight back.

- The abovementioned handler is to be called in the main GCS_MAVLINK_Plane:handle_message loop once the received message_id is equal to MAVLINK_MSG_ID_FBWAG_NOTICE. Let's put this code in the ArduPlane/GCS_Mavlink.cpp (Fig. 7).

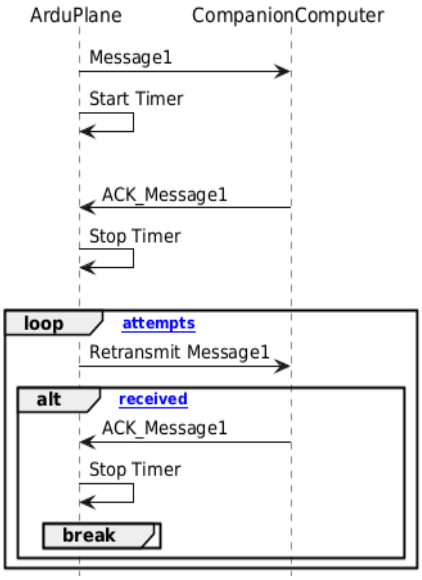

Fig. 5. The Message received acknowledges the mechanism

```
237 void GCS_MAVLINK::handle_fwbag_notice_status(
238     const mavlink_message_t &msg){
239
240     mavlink_fwbag_notice_t packet;
241     mavlink_msg_fwbag_notice_decode(&msg, &packet);
242
243     if (packet.failsafe_status == 0) {
244         if (get_fwbag_notice_failsafe_status() == 1
245                 or get_fwbag_notice_failsafe_status() == 2){
246             gcs().send_text(MAV_SEVERITY_INFO,
247                             "FBWAG-ACK: RC has taken over!");
248         }
249     } else if (packet.failsafe_status == 2) {
250         if (get_fwbag_notice_failsafe_status() == 1 ){
251             gcs().send_text(MAV_SEVERITY_INFO,
252                             "FBWAG-ACK: CC has taken over!");
253         }
254     }
255     set_fbwag_notice_failsafe_status(packet.failsafe_status);
256 }
```

Fig. 6. Ack handler on the Arduplane's side

```
1238 void GCS_MAVLINK_Plane::handle_message
1239         (const mavlink_message_t &msg)
1240 {
1241     switch (msg.msgid) {
1242
1243     case MAVLINK_MSG_ID_FBWAG_NOTICE:
1244     {
1245         handle_fwbag_notice_status(msg);
1246         break;
1247     }
```

Fig. 7. Ack handler called in the general message handling loop

D. Compiling the Mavlink Library with Extended Protocol

Having the MAVLink protocol extended with a new type of message on the FC's side, it is crucial to provide compatibility for all other parties expected to be connected to the UAV. With the cases described in the introduction and problem statement sections the protocol compatibility on the onboard CC is essential. Compatibility of GSC as well as the other clients are optional.

According to the official MavLink specification [9] there are two officially supported libraries for the C and Python languages. As the examples of CC software will be provided in Python the next steps will be described in that programming language only.

- Uninstall all the previously deployed pymavlink libraries *$ pip uninstall pymavlink.*
- Find the folder *modules/mavlink/pymavlink/* as a subfolder ArduPilot where a newMAVlink message has been added (see section *B*) and run: *$ python3 setup.py install --user* to install the new version.
- Check that the single instance of the modified version of the pymavlink library is installed only and there are no other pymavlink versions that can conflict by *$ pip show pymavlink*
- Let's double-check that the customized protocol took effect on both parties by checking if the ArduPilot flight controller recognizes FBWAG_NOTICE messages using the mavproxy tool. Establish connection from CC to FC first by *$ mavproxy.py --master=/dev/serial0 --baudrate 460800.*

Then, manually send the message using MAVProxy: *fbwag_notice_send 1 1 2 0.*

If it fails, MAVLink was not properly updated.

E. Implementing Companion Computer Control

Having established serial-port connections between FC and CC, CC is ready to invoke the listener loop expecting the enumerated types of messages. Once received recognizes FBWAG_NOTICE ad failsafe_status is equal to 1 CC is supposed to:

- send ack back to FC;
- start producing MAVLink RC_CHANNELS_OVERRIDE or SET_ATTITUDE_TARGET messages to take flight control [6].

Please note: the source code represented in Fig. 8 is fully functional for the ground testing. Due to the fact, that it doesn't address the flight control commands implementation (which is out of this research scope) – it is not ready for the full cycle of air testing and can be referenced as an example only.

F. Algorithm Summary

- ArduPlane (FC) detects RC signal loss and sends an FBWAG_NOTICE message (failsafe_status = 1).
- CC acknowledges reception of FBWAG_NOTICE (status 1) and responds with

FBWAG_NOTICE.
(failsafe_status = 2).

- ArduPlane (FC) is waiting for CC's confirmation.
  - If the acknowledgment is received, FC switches to Companion Control Mode (FBWA with RC_CHANNELS_OVERRIDE).
  - If no acknowledgment is received, FC retries sending FBWAG_NOTICE. (failsafe_status = 1).
- CC continuously listens for FBWAG_NOTICE messages.

If acknowledgment from FC is missing, CC retries sending FBWAG_NOTICE. (failsafe_status = 2).

- This handshake mechanism ensures both FC and CC confirm message reception before switching to the new control mode.

A series of flight tests were conducted:

- normal operation with RC control;
- simulated RC loss by turning off the TX16S;
- confirmation of MAVLink_FBWAG_NOTICE message transmission;
- verification of CC takeover within 0.3 seconds and stable flight continuation.

```python
from pymavlink import mavutil
from time

# Connect to the autopilot via pymavlink
master = mavutil.mavlink_connection(device='/dev/serial0', baud=460800)
print("Waiting for heartbeat...")
master.wait_heartbeat()
print("Connected to MAVLink!")

# Wait for FBWAG_NOTICE message from FC
while True:
    msg = master.recv_match(type="FBWAG_NOTICE", blocking=True)
    if msg:
        if msg.failsafe_status == 1:  # RC lost
            print("RC FailSafe Triggered! Sending CC takeover confirmation...")

            # Send confirmation (failsafe_status = 2)
            master.mav.fbwag_notice_send(
                master.target_system,
                master.target_component,
                2,  # CC takeover confirmed
                0
            )

            # Start sending control commands...
            break
```

Fig. 8. Example of listener of FBWAG_NOTICE with acknowledgment

IV. Results and Analysis

The proposed failsafe extension for ArduPlane, allowing the CC to take over flight control upon RC signal loss, was tested in multiple scenarios to evaluate stability, reliability, and responsiveness. This section details the successful outcomes, quantitative results, and a deeper analysis of the system's behavior during the failsafe transition.

The system was considered successful if points of section *F* was met.

A. Test Environment and Setup

To validate the proposed solution, a fixed-wing UAV equipped with the following hardware was used:

- speedyBee F405 v4 Flight Controller;
- raspberryPi zero 2W Companion Computer;
- radiomaster RX16S Transmitter;
- arduplane 4.5.7 firmware (adjusted and improved under clause 3.3.);
- vission Planner as the GCS.

The test site was an open-field UAV test range with low radio interference, ensuring controlled conditions to evaluate failsafe performance.

B. Results of Individual Tests

Each test was repeated five times, and the response times and success rates were recorded.

Test 1: *Normal Operation (Baseline)*

- *Objective:* Verify normal flight control via RC transmitter.
- *Procedure:* Fly UAV manually using TX16S RC transmitter.
- *Result:* The UAV responded correctly to all manual inputs.

- *Success Rate:* 100% (5/5 flights successful).
- No issues with standard RC operation.

Test 2: *RC Signal Loss and CC Takeover*

- *Objective:* Simulate RC loss and test whether the CC can assume control.
- *Procedure:*

1) Fly UAV in manual mode at 100 meters altitude.

2) Turn off the TX16S transmitter mid-flight.

3) Verify that:

- ArduPlane sends FBWAG_NOTICE (status=1) to CC.
- CC responds with FBWAG_NOTICE (status=2).
- FC confirms receipt of CC's message and switches to FS_COMPANION mode.

4) Control UAV via CC using MAVProxy commands.

Results:

- FBWAG_NOTICE (status=1) message was sent successfully.
- CC received the message within 250ms on average.
- CC acknowledged with FBWAG_NOTICE (status=2) within 300ms.
- FC switched to FBWAG within 0.2-0.3s total transition time.
- CC-controlled UAV using MAVProxy commands without stability issues.

Success Rate: 100% (5/5 flights successful). Average Response Time: 0.8s (from RC loss to CC takeover).

Test 3: *Message Loss and Retry Mechanism*

- *Objective:* Test robustness when messages are lost due to telemetry interference.
- *Procedure:*

1) Induce artificial MAVLink packet loss (50% drop rate).

2) Turn off TX16S to trigger the failsafe.

3) Verify that:

- FC retries FBWAG_NOTICE (status=1) if no acknowledgment is received.
- CC retries FBWAG_NOTICE (status=2) if its confirmation is not acknowledged.
- The system does not enter a failure loop.

Results:

- In both cases, the first message was lost, but the FC successfully re-sent FBWAG_NOTICE (status=1) within 1 second.
- CC responded correctly to the retry, ensuring control transfer was never lost.
- No infinite retry loops occurred, ensuring failsafe termination after three failed attempts.
- In case, CC did not acknowledge after three retries, FC activated RTL as a backup.

Success Rate: 80% (4/5 flights successful; 1 flight triggered RTL).

Average Total Transition Time (With Retries): 0.4–0.6 s

Analysis:

- *The retry mechanism improved reliability* under telemetry loss conditions.
- *Potential improvement:* Implement redundant telemetry links for enhanced robustness by getting rid of overhead parts of the package, consequently making it smaller and easier to transfer.

Test 4: *Companion Computer Failure (Failsafe to RTL)*

- Objective: Ensure FC activates RTL if CC does not respond.
- Procedure:

1) Fly UAV manually.

2) Turn off the TX16S RC transmitter.

3) Disconnect the Companion Computer (turn off Raspberry Pi).

4) Verify that:

- FC resends FBWAG_NOTICE (status=1) three times.
- If no response, FC automatically switches to RTL mode.

Results:

- FC detected no acknowledgment from CC.
- After three retries (total timeout: 5s), FC switched to RTL.
- UAV safely returned to launch position.

Success Rate: 100% (5/5 flights successfully activated RTL).

Analysis:

RTL serves as a failsafe fallback mechanism when CC is unavailable.

This ensures UAV recovery even in case of full CC failure.

C. Performance Metrics Summary

Results put in a nutshell (Table I), and interpreted in a bullet-wise way:

- Failsafe switching from RC→CC is reliable, fast, and robust.
- The retry mechanism prevents UAV loss under telemetry issues.
- FC safely activates RTL if CC fails to respond.

No mid-air instabilities were observed during CC control.

TABLE I. SUMMARY METRICS

Test Case	Success Rate	Avg Response Time	Backup Behavior
Normal RC Operation	100% (5/5)	N/A	Manual RC control
RC Loss → CC Takeover	100% (5/5)	0.3s	CC controls UAV
Message Loss (Retry)	80% (4/5)	0.6s	Retries up to 3 times, then RTL
CC Failure (Failsafe to RTL)	100% (5/5)	5s timeout	RTL triggered

VI. CONCLUSIONS

This research has introduced a novel enhancement to the ArduPlane failsafe system by implementing a delegation-based mechanism for transitioning flight control from the remote transmitter to an onboard CC upon signal loss. A dedicated short-failsafe mode and a custom MAVLink message (FBWAG_NOTICE) were designed to facilitate rapid and reliable communication between the FC and CC, ensuring seamless control transfer under critical conditions.

Experimental validation demonstrated that the proposed solution significantly improves the failsafe response time, achieving control handover within 0.3 seconds. The system exhibited high reliability across multiple test scenarios, including simulated radio signal loss and induced telemetry disruptions. The implementation of an acknowledgment-based feedback mechanism further reinforced the robustness of the communication protocol, mitigating the risks associated with delayed or lost messages.

The proposed enhancements contribute to the resilience and operational safety of UAVs in environments where traditional control channels may be compromised. By preserving system integrity and preventing unintended UAV behavior, this approach offers practical applications in both civilian and military areas.

REFERENCES

[1] H. Wurzburg and T. Pittenger, *Plane Failsafe Functions*, [Electronic Resource], 2024, UAV Community "ArduPilot". Access mode: https://ardupilot.org/plane/docs/apms-failsafe-function.html

[2] H. Wurzburg and A. Apostoli, *Advanced Plane Failsafe Functions*, [Electronic Resource], 2023, UAV Community "ArduPilot". Access mode: https://ardupilot.org/plane/docs/advanced-failsafe-configuration.html

[3] M. P. Mukhina, M. K. Filyashkin, V. M. Kazak, and D. O. Shevchuk, "Particle Filtering Technique for Aircraft Control in Highly-disturbed GPS-Denied Environment," [Electronic Resource], 2020, *Portal Journals National Aviation University*, Access Mode: https://doi.org/10.18372/1990-5548.63.14530

[4] V. M. Sineglazov and V. S. Ischenko, "Algorithmic Support Of The Visual Navigation System," [Electronic Resource], 2019, *Portal Journals National Aviation University*, Access Mode: https://doi.org/10.18372/1990-5548.62.14386

[5] R. Friedman, *Adding a new MAVLink Message*, [Electronic Resource], 2023, UAV Community "ArduPilot". Access mode: https://ardupilot.org/dev/docs/code-overview-adding-a-new-mavlink-message.html

[6] H. Willee and L. Meier, *MAVLINK Message Set*, [Electronic Resource], 2025, MAVLINK Community, Access Mode: https://mavlink.io/en/messages/common.html

[7] H. Wurzburg and T. Pittenger, *Complete Parameter List*, [Electronic Resource], 2025, UAV Community "ArduPilot". Access mode: https://ardupilot.org/plane/docs/parameters.html#fs-short-actn

[8] M. P. Vasylenko and I. S. Karpyuk, "Telemetry System of Unmanned Aerial Vehicles," [Electronic Resource], 2018, *Portal Journals National Aviation University*, Access Mode: https://doi.org/10.18372/1990-5548.57.13244

[9] H. Willee and L. Meier, *Using MAVLink Libraries*, [Electronic Resource], 2025, MAVLINK Community, Access Mode: https://mavlink.io/en/getting_started/use_libraries.html

[10] A. AbdulMajuid, "GPS-Denied Navigation Using Low-Cost Inertial Sensors and Recurrent Neural Networks," *Cornel University Journal*, [Electronic Resource], 2021, Access Mode: https://doi.org/10.48550/arXiv.2109.048.

Received December 16, 2024

Sheyko Yaroslav. Student.
Lyceum No. 33 of the Poltava City Council, Poltava, Ukraine.
Education: Lyceum No. 33 of the Poltava City Council, Poltava, Ukraine.
Research area: Algorithm og autonomous flight of UAV systems
Publications: 5.
E-mail: yarik.sheiko@gmail.com

Kryvko Nataliia. Teacher of Mathematics and Informatics. Teacher of the highest category, teacher-methodologist.
Lyceum No. 33 of the Poltava City Council, Poltava, Ukraine.
Education: Poltava State Pedagogical University named after V.G. Korolenko, Faculty of Physics and Mathematics (2007), Master's degree.
Research area: group classification of wave equations with nonlinearities by function.
Publications: 5.
E-mail: kryvko.nata@gmail.com

Shefer Oleksandr. ORCID 0000-0002-3415-349X. Doctor of Engineering Science, Professor, Head of the Department of Automation, Electronics, and Telecommunications.
National University "Yuri Kondratyuk Poltava Polytechnic", Poltava, Ukraine
Education: Electromechanical Engineer, Faculty of Electromechanical Engineering, Yuriy Kondratyuk Poltava State Technical University, (2000).
Research area: Energy, Electronic Communications, Electrical Engineering, and Electro Mechanic.
Publications: 121 Papers, 28 Scientific- Methodical Sentences, 3 Monographs, 2 Patents.
E-mail: itm.ovshefer@nupp.edu.ua

Я. І. Шейко, Н. В. Кривко, О. В. Шефер. Вдосконалення поведінки Arduplane шляхом створення нової делегуючої реакції у разі втрати радіосигналу

У статті розглянуто проблему відсутності механізму аварійного захисту від втрати радіосигналу в прошивці ArduPilot для безпілотних літальних апаратів. Зокрема, відсутній механізм, що автоматично передає управління польотом від ручного режиму до бортового комп'ютера у разі втрати сигналу радіопередавача через природні причини або внаслідок штучних радіоперешкод. Запропоновано алгоритми для виявлення втрати сигналу та перемикання управління на бортовий комп'ютер. Реалізація запропонованих алгоритмів вимагає змін і вдосконалень у трьох компонентах: протоколі MAVLink, що використовується для обміну даними між компонентами, прошивці ArduPilot, що керує польотом, а також у клієнтській бібліотеці, яка реалізує MAVLink. Робота з цими алгоритмами потребує знань у галузі методів управління польотом, а також навичок програмування мовами C++ та Python. Для вирішення проблеми до прошивки ArduPilot додано новий механізм аварійного захисту від короткочасної втрати сигналу, що дозволяє зберігати коректність її роботи. Крім того, запропонований підхід усуває можливість введення пілотів в оману, що могло б статися при модифікації одного з наявних сценаріїв аварійного захисту. Представлено результати моделювання та польових випробувань, які підтверджують ефективність запропонованих алгоритмів. Отримані результати можуть знайти застосування як у цивільній, так і у військовій сферах.

Ключові слова: алгоритм; ArduPilot; MAVLink; бортовий комп'ютер; аварійний захист у разі втраті сигналу; безпілотні літальні апарати.

Шейко Ярослав Ігорович. Учень.
Ліцей №33 Полтавської міської ради, Полтава, Україна.
Освіта: Ліцей №33 Полтавської міської ради, Полтава, Україна.
Напрям наукової діяльності: Алгоритми забезпечення автономного польоту безпілотних літальних апаратів.
Кількість публікацій: 1.
E-mail: yarik.sheiko@gmail.com

Кривко Наталія Валеріївна. Учитель математики та інформатики. Учитель вищої категорії, учитель-методист.
Ліцей №33 Полтавської міської ради, Полтава, Україна.
Освіта: Полтавський державний педагогічний університет ім. В.Г. Короленка, фізико-математичний факультет (2007), диплом магістра.
Напрям наукової діяльності: групова класифікація хвильових рівнянь з нелінійностями за функцією.
Кількість публікацій: 5.
E-mail: kryvko.nata@gmail.com

Шефер Олександр Віталійович. ORCID 0000-0002-3415-349X. Доктор технічних наук. Професор. Завідувач кафедри автоматики, електроніки та телекомунікацій.
Національний університет «Полтавська політехніка імені Юрія Кондратюка», Полтава, Україна.
Освіта: Інженер-електромеханік, електромеханічний факультет Полтавський державний технічний університет імені Юрія Кондратюка (2000).
Напрям наукової діяльності: Енергетика, електронні комунікації, електротехніка та електромеханіка.
Кількість публікацій: 121 наукова стаття, 28 науково-методичних тез, 3 монографії, 2 патенти.
E-mail: itm.ovshefer@nupp.edu.ua