Towards Safety Methods for Unmanned Aerial Systems to achieve Fail-Safe or Fail-Operational Behaviour

Philipp Stelzer
Graz University of Technology
Graz, Austria
Email: stelzer@tugraz.at

Raphael Schermann
Graz University of Technology &
Infineon Technologies Austria AG
Graz, Austria
Email: raphael.schermann@infineon.com

Felix Warmer
Graz University of Technology
Graz, Austria
Email: warmer@tugraz.at

Hannes Winkler
Graz University of Technology
Graz, Austria
Email: hannes.winkler@tugraz.at

Georg Macher
Graz University of Technology
Graz, Austria
Email: georg.macher@tugraz.at

Christian Steger
Graz University of Technology
Graz, Austria
Email: steger@tugraz.at

***Abstract*—Drones and Unmanned Aerial Systems (UAS') in general are slowly taking over more and more areas of our everyday lives. They are already being used for cinematic productions, photography, structural inspections and will take over even more areas in the future like package delivery, surveillance tasks, or rescue missions. This inevitably leads to more and more UAS' occupying the airspace and therefore pose a danger to manned aviation, buildings, infrastructure, uninvolved persons, and other UAS'. In times of exponential growth of the Unmanned Aerial System (UAS) industry, the safety and security measures for UAS' becomes more and more significant. Until now, UAS safety has largely been the responsibility of the pilot, who had to take action if the UAS experienced any kind of failure. As the UAS industry continues to drift towards autonomy, the responsibility no longer lies with the pilot but instead, the UAS must have some sort of fall-back performance or fail-safe routine implemented. In our publication, we deal with such routines to establish fail-safe respectively fail-operational behaviour in case of sensor failure in an autonomously flying UAS beyond the visual line of sight of the UAS operator.**

Keywords–UAS; automated vehicles; safety; fail-operational; fail-safe

I. INTRODUCTION

Unmanned Aerial Systems (UAS') are already used in many cases and will be increasingly important for various missions and applications in the future [1], [2]. For example, in the rescue or localisation of earthquake victims [3]. Nowadays, UAS' are still remotely controlled in many cases, but in the future they should be able to fly autonomously to complete the missions without an operator. For this purpose, however, it is also important that these UAS' are equipped with appropriate safe and reliable systems. It must be possible to assume that a Unmanned Aerial System (UAS) can complete the mission with reduced functionality or at least establish a fail-safe status in the event of failures in the systems on board, such as a failure of an important sensor for environment perception. This requires implemented safety strategies in the UAS that it is able to be fail-operational or at least fail-safe. Research on safe and robust UAS' is already being carried out at a high level of intensity. The Airborne Data Collection on Resilient System Architectures (ADACORSA) [4] project, for example, whose representative cover image can be seen in the Figure 1, is concerned with resilient architectures for flight systems, among other things. The main vision of this project is to provide technologies to render drones as a safe and efficient component of the mobility mix, with differentiated, safe and reliable capabilities in extended Beyond Visual Line Of Sight (BVLOS) operations [4]. With this publication, we also want to contribute to the improvement of the safety of autonomous flying UAS' and to establish a fail-safe respectively fail-operational behaviour in UAS' in case of sensor failures. The remainder of the publication is structured as follows. An overview of background and related work is given in Section II. Subsequently, our novel methods and proposed solutions for the reaction to a sensor failure will be introduced in detail in Section III, and the achieved results, including a short discussion, will be provided in Section V. A summary and short discussion of the findings will conclude this publication in Section VI.

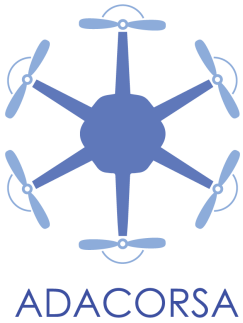

Figure 1. Airborne Data Collection on Resilient System Architectures [4].

II. BACKGROUND AND RELATED WORK

Safety, whether for automated vehicles to the ground or for automated UAS' in the air, is a very sensitive issue [5], [6]. Unlike manually operated vehicles, automated vehicles must

 ISBN: 978-1-68558-006-3

be able to rely on the sensors for environment perception and on the processing units to a certain extent. For this purpose, established measures to increase safety are briefly discussed and terminology is defined in the following subsections. UAS' are then dealt with specifically. However, several drone projects, such as from Milhouse [7], have already been realised by using open-source software and off-the-shelf hardware - this is also our approach to evaluating our methods in this publication. Drones and other UAS have already been simulated for scientific purposes by Ma et al. [8] and Megalingam et al. [9], among others. Ebeid et al. [10], for instance, have given a general survey of open-source UAS flight controllers and flight simulators in their publication and highlighted their importance. A general overview of safety approaches and some State-of-the-Art safety measures in UAS' is given below. Furthermore, the role and use of a companion computer, which can assist a flight controller with complex tasks and provide further capabilities, will be discussed in more detail.

A. General Overview of Safety Approaches

To ensure safety, automated vehicles often rely on redundancy and diversity. Whether in aviation or in the automotive domain, similar concepts are applied with regard to diversity and redundancy. In detail, these concepts usually differ in the scope of execution [11]. A redundant approach relies on the use of several components of the same type. There can be various approaches how a faulty behaviour of a component can be detected or compensated. A diverse approach is similar. Here, however, different components are used instead of the same type. For example, in the case of environment perception, a sensor fusion of data from, e.g., Radio Detection And Ranging (RADAR) and camera is used. In a redundant sensor fusion approach, only data from similar RADARs could be used [12], [13], [14]. Depending on the constellation of the chosen approach, this leads to fail-safe or fail-operational behaviour [15]. The definitions of fail-safe and fail-operational are given below in order to clarify which properties are attributed to the respective behaviour:

1) Fail-Safe: In the event of an intolerable failure, the system is brought into a fail-safe status. The system differentiates between the unimpaired continuation of operation or a stop of the system. In case of a system stop, if the system has a transient error, the system is restarted. Otherwise, the system remains in stop state [15], [16]. In the case of a UAS, the UAS is of course not immediately stopped and brought down, but must be brought to a safe state, as described in Section III, for example.

2) Fail-Operational: In contrast, with fail-operational, an failure occurrence is tolerated and the system remains operational [15], [16]. In our publication, Section III also proposes a method for UAS' to achieve fail-operational behaviour.

B. State-of-the-Art Safety Measures for UAS'

Even though the use of highly automated UAS' is only just emerging, a lot of research has already been done to strengthen the environment perception and control of these systems. For example, Saxena et al. [17] have been working on learning failure responses for autonomous, vision-based flying of robust systems, and thus manage to learn simple failure recovery manoeuvres based on experience. Another approach that may also be interesting for UAS' is the introduced competence-aware path planning via introspective perception by Rabiee et al. [18]. This is a framework that integrates introspective perception into path planning to reduce robot navigation failures. Furthermore, research was also conducted on the robust control of UAS'. Vey and Lunze, for instance, dealt with an active, fault-tolerant control framework in their publication [19]. With this framework, it should be possible to ensure that the diagnostic result is always unambiguous from the point of view of control reconfiguration. Sharifi et al. [20] also dealt with fault-tolerant control of UAS' using a sliding mode control for a quadrotor UAS. This shows that safety measures for UAS' have also been dealt with in the recent past.

C. Companion Computers on UAS'

For safety tasks on UAS', so-called companion computers are also often used. But in theory, nothing more than a flight controller is needed for the UAS to be able to fly in terms of control devices. The flight controller is responsible for operating the hardware of the UAS, for example keeping the UAS permanently levelled and stable, and must not have any time delays as this would be fatal for the flight. Although the UAS could fly without any additional hardware, with just a State-of-the-Art and common flight controller, however, it would not be capable of handling any highly complex tasks [21]. An additional companion computer is used to perform difficult tasks, send commands to the flight controller and execute higher-level scripts. Another advantage of using a companion computer, like a Raspberry Pi, is that it provides an interface between the UAS itself and other hardware like environment perception sensors. In 2019 a new method for victim detection after disastrous events like earthquakes was published [3]. After such events, victims are usually buried under debris and can therefore not be detected by UAS' that use cameras for visual identification. The idea was, that a flight controller is connected to a Raspberry Pi, which in turn will be equipped with a microphone and a speaker. The UAS hovers above the debris and uses the speaker to draw the attention of the victims for them to know that someone is listening and that they have to make aware of themselves now. In case, the UAS records a scream for help it is going to be stored with the corresponding geographical coordinates and sent to the rescue teams. In this particular use case, the Raspberry Pi is needed to activate the speaker, record everything the microphone detects, filter out the permanent noise of the UAS, read out the data from the connected GPS module and send everything to a predefined receiver. All these things could not be accomplished with just a flight controller, but the portability of the Raspberry Pi makes it possible to perform all these complex operations on the UAS and forward only the final result [3]. The computing power of the Raspberry Pi allows also to mount a camera on the UAS and perform a precision landing based on real-time visual landing pad detection. The algorithm is executed on the Raspberry Pi, it receives a snapshot from the camera as input, processes the image, tries to find the landing pad, calculates the UAS's position and orientation according to the landing pad and sends manoeuvring commands to the flight controller via the serial port. Algorithms like this have achieved landing accuracy of up to 98.5% at a frame rate of approximately 13 Hz [22]. All the above-mentioned scenarios and functions prove that companion computers add further valuable capabilities to UAS'.

 ISBN: 978-1-68558-006-3

III. Novel Safety Methods for UAS'

In this section, three methods are proposed to achieve feasibility of preventing a crash of an autonomously flying UAS in the event of a sensor failure outside the pilot's Field of View. It is assumed that a faulty behaviour of the sensors can be detected by monitors of a companion computer on board of the UAS and thus the flight controller can be informed that the sensor concerned is no longer functioning reliably. In the simplest case, the UAS is equipped with four distance sensors, one on each side of the UAS. The worst case scenario would be if the sensor in the direction of flight would fail, as shown in Figure 2. The best case would be if the sensor in the opposite

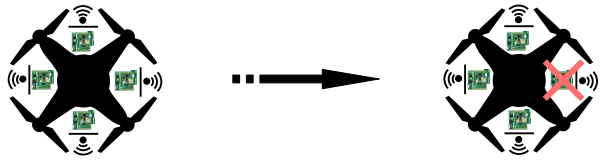

Figure 2. Simplified visualization of a sensor failure.

direction would fail. In the latter case, no immediate action would be required, as this sensor has virtually no effect on the UAS's airworthiness, under the condition that the UAS is not rotated. A reaction would be needed in case of a failure of one of the lateral sensors, but the required manoeuvre is the same as in the worst case, with only minor adjustments. During investigation, we came up with three different solutions on how the UAS should react in case of a sensor failure. The solutions are kept simple and straightforward, yet they are very effective and can even result in a fail-operational behaviour like completing the flight mission with a semi-functioning vehicle.

A. Fail-Safe and Fail-Operational Methods

In the following the three investigated safety methods for UAS' are presented.

1) Immediate Landing from UAS: The first approach we came up with is the most obvious one. In the event of a sensor failure, it does not matter which sensor fails, the UAS is simply landed right on the spot where the failure happened. A conceptual overview of this scenario is shown in Figure 3. This is the simplest solution but is not necessarily applicable at all times. For example, the failure could happen in a place where it is known nothing about the underlying surface, it could be a river where the UAS would be lost, or even worse, it could be right above a crowd of people where the rotating rotor blades of the UAS could cause serious injuries. If the UAS is used in a densely populated city, this solution would not be the first choice due to the potential risk of landing near people or in traffic. However, if the UAS is used for flights between cities over open countryside or in predefined areas, the UAS could simply land in the event of a failure to be accessible for quick repair of the sensor.

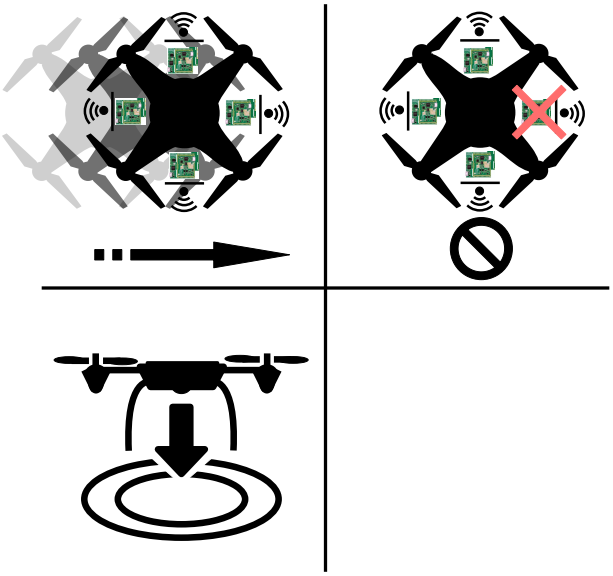

Figure 3. Immediately landing in case of sensor failure.

2) UAS Return to Launch Position: The second approach to handle a sensor failure is to return the UAS to its launch position where it started from. The applicability of this solution, like in the first one, is not always guaranteed, it works best if the sensor in direction of the flight crashes. In that case, it would not be necessary to rotate the UAS, if a lateral sensor crashes it would be necessary to turn the UAS around in order to be able to return safely to the launch position. In contrast to the option of simply landing, in this scenario, it is known everything about the condition of the landing site, since the UAS have already taken off from there. In addition, the UAS could fly back the exact same route it took on the way to the location of the failure. In that case, the UAS could be almost certain that the path should be free of obstacles unless there are other flying objects or moving parts involved. Figure 4 depicts this approach. Whether the use of this solution makes sense or

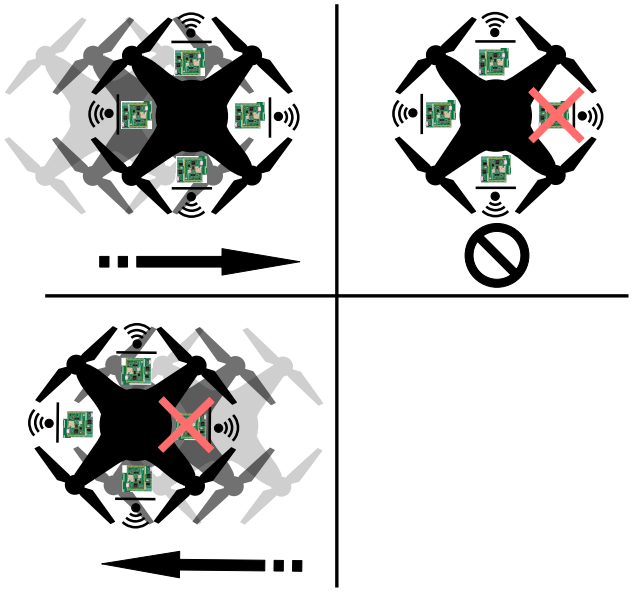

Figure 4. Return to launch position in case of sensor failure.

not is also depending on the location of the sensor failure. In case, the event occurs shortly after the departure of the UAS, it clearly would be the first choice, since the UAS would not have to travel much distance to safe position. However, when the flight has progressed to the point where you are closer to the destination than to the take-off, it should be considered if a different solution would be more practical.

3) UAS Rotation: In the third and last approach investigated, the UAS is rotated in the event of a sensor failure. This approach is a temporary solution to maintain the airworthiness

 ISBN: 978-1-68558-006-3

of the UAS in order to finish the already started mission. The idea is that depending on which sensor fails, the UAS is simply rotated to the position in which the failed sensor is looking backwards as seen in the direction of flight in Figure 5. The

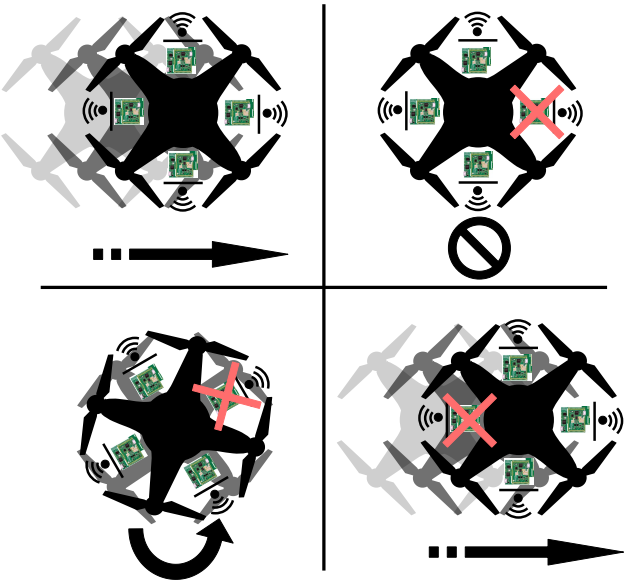

Figure 5. Rotate the UAS and finish the mission in case of sensor failure.

sensor opposite the direction of flight is not necessary most of the time, since we do not expect things or obstacles to approach the UAS from behind. If other UAS or flying vehicles get close to the according damaged UAS from behind they have their own sensors on board to become aware of the according UAS in their path. The remaining three sensors should be sufficient to complete the mission and land safely before the sensor can be replaced. This solution could be used in almost any situation, as the potential risk is kept as low as possible and thus there is no immediate danger to people or other things. With this approach, the UAS remains operable to a reduced extent, but still fail-operational.

B. Decision Graph for Method

In Figure 6, the decision flow during the mission of the UAS is depicted. It shows how the companion computer works from the definition of the way-points that the UAS should approach during the mission to a possible sensor failure and the corresponding fail-safe respectively fail-operational method or the successful completion of the mission. At the beginning, it is determined which method is to be used in the event of a sensor failure. Then the mission is started and it is continuously checked whether the sensor is still working correctly. This can be done by monitors, for example, which then notify the companion computer accordingly. As long as no sensor failure is detected, the mission can continue as usual. When the mission is completed, the mission stops and it is not checked for sensor failures until a new mission is started. However, if the mission is still in progress, it will be continuously checked for sensor failures and, in case of a sensor failure, the safety method defined at the beginning will be activated. When one of the two methods ”Immediate Landing from UAS” and ”UAS Return to Launch Position” are used, they are executed and the mission is stopped when the respective target position is reached. In case the method ”UAS Rotation” has been

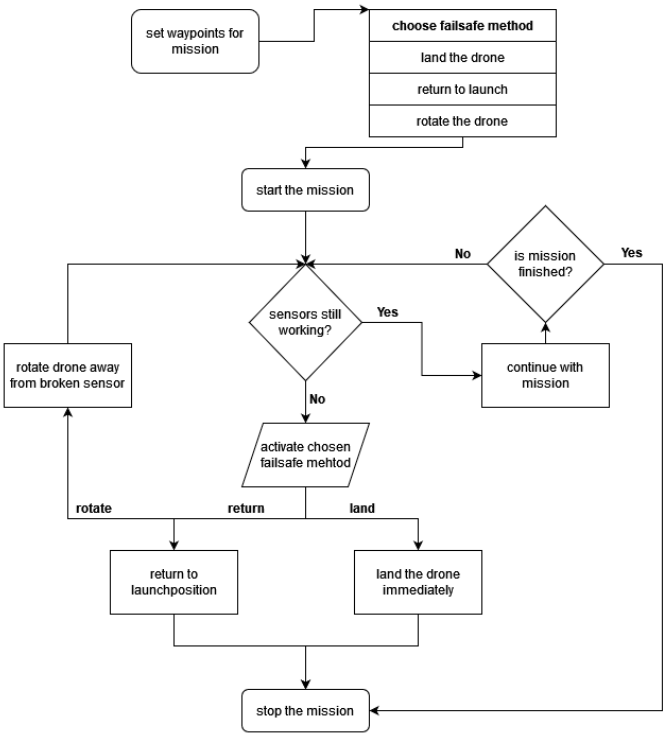

Figure 6. Decision graph of the companion computer for the three possible methods in the event of a sensor failure.

chosen, the UAS is first rotated so that a functioning sensor is positioned in front and the mission can then be continued. Here, the system will continue to check for sensor failures and if the current front sensor also fails, the method will be used again. Otherwise, the mission continues until it can be stopped.

IV. Setup of Development

In this section, the development and evaluation setup is presented. Since our intention was to investigate and test new methods to ensure safety, a corresponding concept of a development environment was designed. For this purpose, a UAS, in our case a drone, was simulated using open-source software and hardware, such as a Raspberry Pi [23] as companion computer, a Position2Go RADAR Sensor [24] from Infineon Technologies AG and a remote PC as GCS, as shown in Figure 7. The Raspberry Pi is the core of the

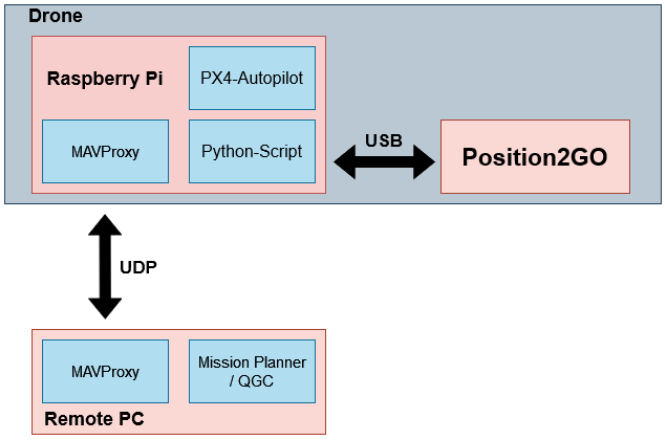

Figure 7. Concept of the development environment setup.

drone. It is running MAVProxy for the communication with the GCS, various Python scripts including the mission script and PX4 autopliot for operating the drone. The Postion2Go sensor is connected to the Raspberry Pi via USB. The mission script uses the Radar Host Communication Library for the status check of the Position2Go. The remote PC, the GCS, also runs MAVProxy and a Mission Planner/QGroundControl to track the progress of the mission in real time. The messages from MAVProxy are sent via UDP connection between the simulated drone and the remote PC. More detailed information on the hard- and software used is given in the following subsections.

A. Hardware

As already mentioned in Section II-C, a Raspberry Pi can be used as a companion computer. Due to the fact that we used a Raspberry Pi [23] and the Position2Go Development Kit [24] from Infineon Technologies AG as a RADAR sensor for our experiments, these two hardware components are described in more detail below. Although other hardware components - like the remote PC and a monitor - were also used, these two were the most essential to show that our fail-safe and fail-operational methods are relevant and essential for a companion computer for UAS'.

1) Raspberry Pi 4 Model B: For the experiments in this publication, the Raspberry Pi 4 Model B, depicted in Figure 8, was used. As with all earlier models in the Raspberry series, the operating system must be written onto a memory card. Due to the software used for our experiments, which is explained in more detail in Section IV-B, the choice of the operating system fell on a standard version of Ubuntu (Ubuntu 20.10, to be specific), which in contrast to Raspbian OS already has most of the used packages and libraries preinstalled. The

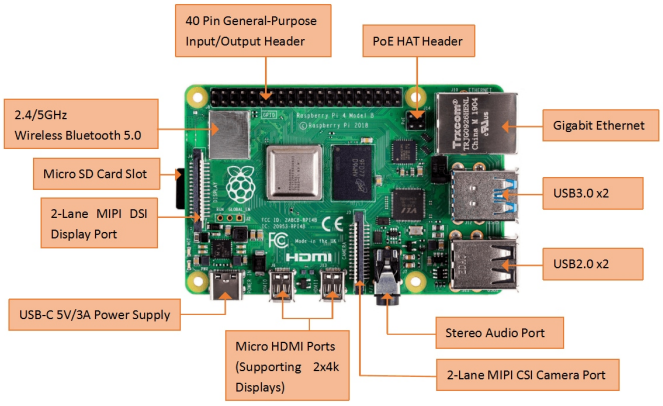

Figure 8. Raspberry Pi 4 Model B.

Raspberry Pi 4 Model B is far superior to its predecessors in terms of hardware. For this work the model with the following specifications was used [23]:

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @1.5GHz
- 4GB LPDDR4-3200 SDRAM
- 2.4GHz and 5.0GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports, 2 USB 2.0 ports
- 40 pin GPIO header
- 2 micro-HDMI ports (supporting two 4k displays)
- USB-C power supply (5V/3A)

2) Position2Go RADAR Sensor: The Position2Go RADAR sensor from Infineon Technologies AG, depicted in Figure 9, can be used to track multiple objects including humans, to detect angle and distance of moving and static targets, and to detect motion, speed, and direction of movement [24]. For

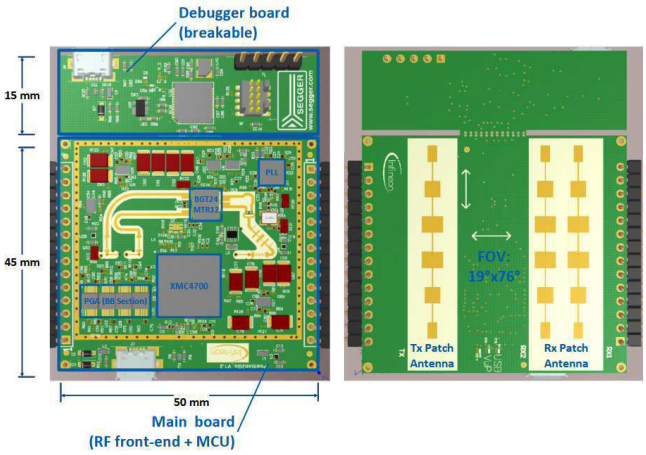

Figure 9. Position2Go board with main components and dimensions.

testing our methods, however, it is not relevant which sensor is used. In this publication, the focus is on the methods in the case of a sensor failure itself and not on how the sensor failure can be detected or what data is delivered. Another environmental perception sensor could also be used for this purpose. In our case, it is only important that we can inject a sensor failure in order to test whether our methods and whether they are applied correctly in the case of a sensor failure.

B. Software

In this section, the most important software parts that were used during our experiments are discussed. The focus is on open-source projects that are already well established for drones and UAS'.

1) MAVProxy: The originally intended use of MAVProxy was to operate as fully-functional Ground Control Station (GCS) [25]. It was designed as a minimalist, command line based software, which can be extended via add-on modules. Due to the light-weighted design and the portability to every commonly used operating system (Linux, Windows, OS X) it can run on almost every device, including smaller and less powerful ones like the Raspberry Pi. One feature is the ability to connect with other GCS, even if they are running on other devices in the same network or with the help of a VPN connection to devices outside the network.

2) PX4 Autopilot/Flight Controller: The PX4 autopilot [26] is an open-source project of the Dronecode Foundation, a US-based non-profit organization that also developed MAVLink (a communication protocol between companion computers like a Raspberry Pi, autopilots like the PX4 and GCS') and QGroundControl. The PX4 autopilot is basically a flight control software for operating UAS and other unmanned vehicles, such as rovers, fixed-wing aircraft, and many other experimental types, such as balloons and even boats and

 ISBN: 978-1-68558-006-3

submarines. The software needed to build the autopilot can be found for free in a GitHub repository [26]. It is operating system independent and highly portable. After cloning the code to the device where the autopilot will run, it can be built the software by using the provided Makefile depending on the needed configuration. No specific hardware requirements have to be met, the jMAVSim simulator has to be used to simulate the UAS respectively drone. jMAVSim is already included in the source code, it has not to be installed.

3) Ground Control Station: A GCS is usually software that can be used to monitor and control the flights of UAS [27]. It

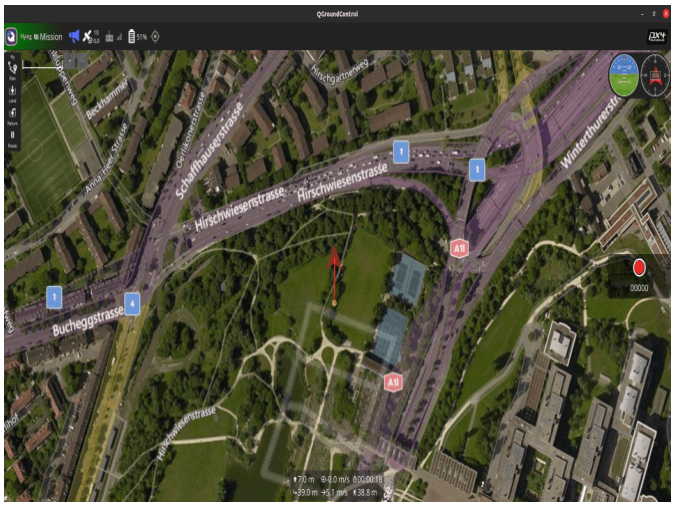

Figure 10. Screenshot from QGroundControl.

is typically linked to the vehicle via a wireless communication method but is running on a ground-based computer. On a virtual map displayed in the program, the user is able to define way-points for an autonomous mission. The user is also able to track the progress of the mission and check in on the real-time flight information like speed, altitude or distance to the next target through the program. A well known GCS is QGroundControl [28], which is developed by the same company as the PX4 autopilot. In Figure 10 a screenshot from the user interface of the QGroundControl is depicted.

V. Results

In this section, we provide the results of the implemented methods from our functionally safe system for UAS, which have been introduced in Section III.

A. Evaluation and Development Environment

The development environment used for this, according to the concept in Figure 7, was also used for the evaluation of the methods. The development and evaluation environment in use can be seen in Figure 11. The laptop (remote PC) is running the GCS, while everything on the right screen is running on the Raspberry Pi. Three different shells can be seen on the screen, one is running the PX4 autopilot, one is running the MAVProxy and one shell is running the mission script. In front of the remote PC, the Raspberry Pi can be seen, to which the Position2Go sensor is connected.

B. Method Examination

In the Figures 12, 13 and 14 parts of a single simulation run are depicted. The small window shows a shell running on the Raspberry Pi that outputs text-based information about the status of the mission. The map in the background is the GCS running on the remote PC to monitor the progress of the mission. The top red circle highlights the current output of the mission script to the Raspberry Pi, the leftmost circle shows the drone's orientation and position, and the bottom circle shows the drone's current altitude. The fall-back routine for this simulation run is described in Section III-A3, the drone will turn around in case of a failure. At the beginning, the system waits until a mission or a target position is known. This is shown in Figure 12. While waiting, the drone is on the ground. As soon as the drone receives a mission or a target position, it begins its flight. In order to check the required functionality of the predefined method, the Position2Go sensor

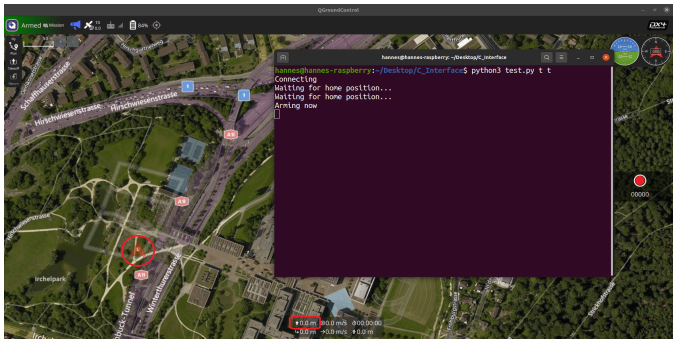

Figure 12. Drone on the ground at the beginning of the mission.

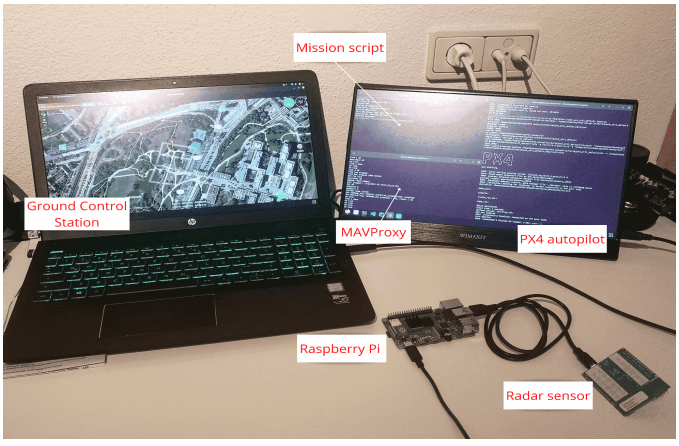

Figure 11. Evaluation and development environment setup in use.

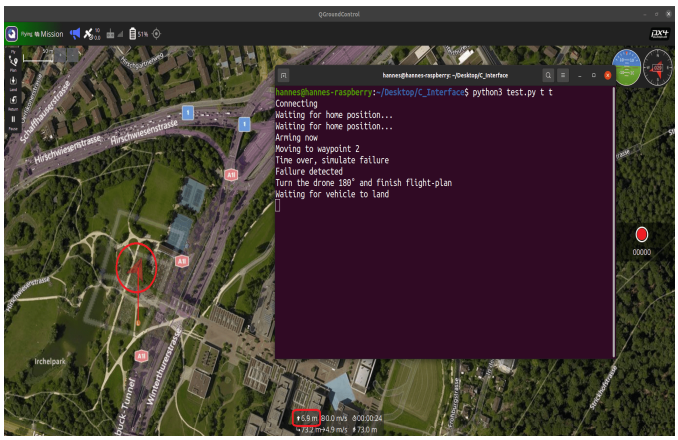

Figure 13. Drone in the air immediately after sensor failure.

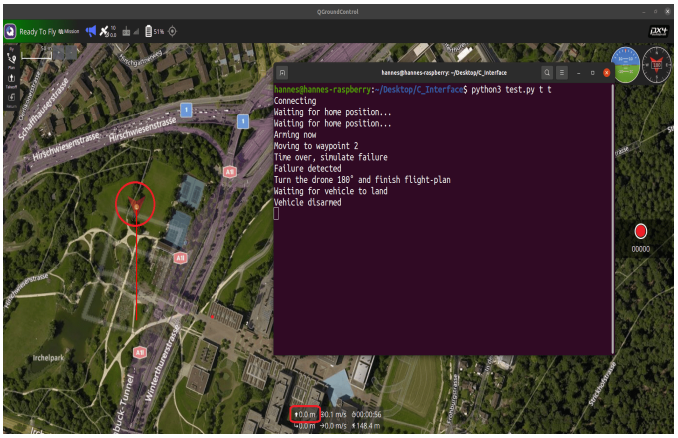

Figure 14. Rotated drone on the ground after the finished mission.

was unplugged during the simulation run and so a sensor failure was initiated. A monitor in the companion computer then detects that no more data from the sensor is arriving at the companion computer and reports a sensor fault. To detect more complex sensor faults, more sophisticated monitors must be used - but sensor fault detection was not the main focus here. However, in this case the predefined method was the "UAS rotation". Thus, in Figure 13 it can also be seen that the drone is already rotating immediately after the sensor failure. Figure 14 shows that the mission was successfully completed and the drone landed at the target point rotated 180°. If one of the two other methods is selected in advance, the simulation will run as described above in the event of a sensor failure. But in contrast to the drone rotation, the drone either lands immediately or flies back to the starting point and lands there. This depends on the chosen method at the beginning.

VI. Conclusion

As already mentioned in this publication, the research and development regarding UAS' becomes more and more important in our everyday lives and will play a major role in the future. We have discussed the given problem and presented possible solutions, in the form of investigated methods, for the scenario when a sensor fails during an ongoing mission. The simplest of them, landing immediately after sensor failure, has advantages, such as the low risk for other flying objects, but also has disadvantages like the requirement to fly over open terrain and to know the surface beneath. The second solution, returning to the launch position, is probably the least efficient solution of them all, as flying backwards with the remaining sensors is quite similar to the third solution. But with the third proposed solution, it is possible to finish the current mission. Despite that, depending on the location of the failure, the second solution is also worth to be considered. However, if strong wind or other influencing factors prevent rotation or make it impossible, the second solution must still be chosen. Considering the above, the most promising solution is probably the third and last one, where the UAS is rotated to the point where the broken sensor has the least impact on the flying behaviour of the UAS and finish the current mission. With this solution, the mission is not only finished, but also the risk of a partially uncontrolled landing for the people under the UAS or an abrupt change of direction for other flying objects in the immediate vicinity is avoided. Considering all of these aspects, the best solution for UAS' used in safety-critical applications would be a combination of all above mentioned options. However, the companion computer could either be developed in a manner that the operator of the UAS sets a fail-safe or fail-operational method when the UAS is launched, or that the UAS is developed to the point where it can apply one of the given solutions independently. This in turn would be more difficult, as the UAS would have to know everything of the surface beneath, other flying objects in the air, the already covered and remaining flight path, the task of the mission that is currently going on, and many other aspects. In conclusion, however, these methods can contribute to safe, highly automated UAS' for safety-critical applications in the future. Thus, safety in the immediate vicinity of such UAS' will also be ensured.

Acknowledgment

The authors would like to thank all national funding authorities and the ECSEL Joint Undertaking, which funded the ADACORSA project under the grant agreement number 876019.

ADACORSA is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program ICT of the Future between May 2020 and April 2023 (grant number 877585). More information: https://iktderzukunft.at/en/.

References

[1] D. Bamburry, "Drones: Designed for product delivery," Design Management Review, vol. 26, no. 1, 2015, pp. 40–48.

[2] D. Câmara, "Cavalry to the rescue: Drones fleet to help rescuers operations over disasters scenarios," in 2014 IEEE Conference on Antenna Measurements & Applications (CAMA). IEEE, 2014, pp. 1–4.

[3] Y. Yamazaki, M. Tamaki, C. Premachandra, C. Perera, S. Sumathipala, and B. Sudantha, "Victim detection using UAV with on-board voice recognition system," in 2019 Third IEEE International Conference on Robotic Computing (IRC). IEEE, 2019, pp. 555–559.

[4] ADACORSA Project. Airborne Data Collection on Resilient System Architectures. https://www.adacorsa.eu. Retrieved: July, 2022.

[5] R. Altawy and A. M. Youssef, "Security, privacy, and safety aspects of civilian drones: A survey," ACM Transactions on Cyber-Physical Systems, vol. 1, no. 2, 2016, pp. 1–25.

[6] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," IEEE access, vol. 8, 2020, pp. 87 456–87 477.

[7] M. O. Milhouse, "Framework for autonomous delivery drones," in Proceedings of the 4th Annual ACM Conference on Research in Information Technology, 2015, pp. 1–4.

[8] C. Ma, Y. Zhou, and Z. Li, "A New Simulation Environment Based on Airsim, ROS, and PX4 for Quadcopter Aircrafts," in 2020 6th International Conference on Control, Automation and Robotics (ICCAR). IEEE, 2020, pp. 486–490.

[9] R. K. Megalingam, D. V. Prithvi, N. C. S. Kumar, and V. Egumadiri, "Drone Stability Simulation Using ROS and Gazebo," in Advanced Computing and Intelligent Technologies. Springer, 2022, pp. 131–143.

[10] E. Ebeid, M. Skriver, K. H. Terkildsen, K. Jensen, and U. P. Schultz, "A survey of open-source UAV flight controllers and flight simulators," Microprocessors and Microsystems, vol. 61, 2018, pp. 11–20.

[11] V. Kharchenko, "Diversity for safety and security of embedded and cyber physical systems: Fundamentals review and industrial cases," in 2016 15th Biennial Baltic Electronics Conference (BEC). IEEE, 2016, pp. 17–26.

 ISBN: 978-1-68558-006-3

[12] J. Steinbaeck, C. Steger, G. Holweg, and N. Druml, “Next generation radar sensors in automotive sensor fusion systems,” in 2017 Sensor Data Fusion: Trends, Solutions, Applications (SDF). IEEE, 2017, pp. 1–6.

[13] Y. Luo, A. K. Saberi, T. Bijlsma, J. J. Lukkien, and M. van den Brand, “An architecture pattern for safety critical automated driving applications: Design and analysis,” in 2017 Annual IEEE International Systems Conference (SysCon). IEEE, 2017, pp. 1–7.

[14] A. Armoush, “Design patterns for safety-critical embedded systems.” Ph.D. dissertation, RWTH Aachen University, 2010.

[15] A. Kohn, M. Käßmeyer, R. Schneider, A. Roger, C. Stellwag, and A. Herkersdorf, “Fail-operational in safety-related automotive multi-core systems,” in 10th IEEE International Symposium on Industrial Embedded Systems (SIES). IEEE, 2015, pp. 1–4.

[16] P. Stelzer, A. Strasser, C. Steger, and N. Druml, “Fail-operational shock detection and correction of MEMS-based micro-scanning LiDAR systems,” in 2020 IEEE Sensors Applications Symposium (SAS). IEEE, 2020, pp. 1–6.

[17] D. M. Saxena, V. Kurtz, and M. Hebert, “Learning robust failure response for autonomous vision based flight,” in 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017, pp. 5824–5829.

[18] S. Rabiee, C. Basich, K. Wray, S. Zilberstein, and J. Biswas, “Competence-Aware Path Planning via Introspective Perception,” IEEE Robotics and Automation Letters, 2022.

[19] D. Vey and J. Lunze, “Experimental evaluation of an active fault-tolerant control scheme for multirotor UAVs,” in 2016 3rd conference on control and fault-tolerant systems (systol). IEEE, 2016, pp. 125–132.

[20] F. Sharifi, M. Mirzaei, B. W. Gordon, and Y. Zhang, “Fault tolerant control of a quadrotor UAV using sliding mode control,” in 2010 conference on control and Fault-Tolerant Systems (SysTol). IEEE, 2010, pp. 239–244.

[21] V. Marojevic, I. Guvenc, M. Sichitiu, and R. Dutta, “An experimental research platform architecture for UAS communications and networking,” in 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall). IEEE, 2019, pp. 1–5.

[22] H. Jiaxin, G. Yanning, F. Zhen, and G. Yuqing, “Vision-based autonomous landing of unmanned aerial vehicles,” in 2017 Chinese Automation Congress (CAC). IEEE, 2017, pp. 3464–3469.

[23] Raspberry Pi Foundation. Raspberry Pi 4 Model B. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. Retrieved: July, 2022.

[24] Infineon Technologies AG. Position2Go Development Kit. https://www.infineon.com/dgdl/Infineon-AN553_BGT24MTR12_XMC4700_Position2Go_DemoBoard-ApplicationNotesv01_03-EN.pdf?fileId=5546d4626cb27db2016d44631adb021f. Retrieved: July, 2022.

[25] ArduPilot Dev Team. Mavproxy documentation. https://ardupilot.org/mavproxy/. Retrieved: July, 2022.

[26] PX4 Dev Team. PX4 Autopilot: Open Source Autopilot for Drones. https://px4.io/. Retrieved: July, 2022.

[27] D. Perez, I. Maza, F. Caballero, D. Scarlatti, E. Casado, and A. Ollero, “A ground control station for a multi-uav surveillance system,” Journal of Intelligent & Robotic Systems, vol. 69, no. 1, 2013, pp. 119–130.

[28] Dronecode Project. QGroundControl. http://qgroundcontrol.com/. Retrieved: July, 2022.

 ISBN: 978-1-68558-006-3