Test platform for Simulation-In-Hardware of unmanned aerial vehicle on-board computer

Arsen R. Petrosian[1], Ruslan V. Petrosian[1] and Oleksandra M. Svintsytska[1]

[1]Zhytomyr Polytechnic State University, 103 Chudnivsyka Str., Zhytomyr, 10005, Ukraine

Abstract

Recent years have been marked by the rapid development of unmanned aerial vehicles, which is of interest not only to ordinary citizens but also to military, industrial and civilian spheres of activity. Designing and developing software to control the orientation and movement of an unmanned aerial vehicle in space takes a long time, including time for debugging, testing, and conducting flight tests. Errors made in the development of the software can lead to emergencies or other unforeseen failures during operation, which can lead to the destruction of the vehicle or cause harm to people and the environment. In the course of analyzing recent studies of software testing methods for embedded systems, their advantages and disadvantages were identified. It was found that both mechanical test benches and simulation modeling in a 3D environment are used: SIL, HIL and SIH simulation. Mechanical test benches allow testing and calibration of the parameters of real models of unmanned aerial vehicles, but have some limitations. Simulation modeling does not require additional equipment due to the use of virtual models of unmanned aircraft and does not have the limitations inherent in mechanical test benches. The most successful implementation of the test platform is the implementation using SIH simulation. The proposed method is an improved version of SIH simulation. The basis of this variant is a sensor and actuator simulator that ensures the operation of the original firmware of an unmanned aerial vehicle and allows you to organize the interconnection of on-board and personal computers. To test the idea, the test platform was implemented in practice. A configuration utility for the simulator of sensors and actuators has also been developed. The results obtained in the work will allow to organize software testing of the on-board computer of an unmanned aerial vehicle even without having the source code.

Keywords

UAV, unmanned aerial vehicle, on-board computer, software testing, SIH simulation, HIL simulation, 3D environment, sensor and actuator simulator

1. Introduction

Unmanned aerial vehicles (UAV), which are flying robots, are an important part of scientific research in military, industrial and civilian areas of activity: aerial photography and mapping, promptly obtaining information about the consequences of emergencies, monitoring industrial and natural complexes, delivering goods, entertainment purposes, etc. Designing and developing software for controlling UAV orientation in space takes a long time, including time for debugging, testing, and flight tests.

Using classical methods of debugging and testing flight controller firmware is problematic. One way to test controller firmware is to use an oscilloscope and logic analyzer to obtain timing diagrams. This approach is effective, but it is not always possible. In our case it is not possible to use these tools, because the UAV must move in space. When the controller is operating power equipment, it must not be stopped. Stopping the controller during operation at the breakpoint will, at best, cause the equipment to stop, and at worst it may lead to equipment failure. Also further step-by-step debugging becomes impossible because the system state will change (closed-loop control system). However, many debugging tools, such as J-Link, display the change of variables in real time. As in the first case, it is impossible to connect directly, so it is necessary to use remote data transfer. this possibility is theoretically possible (figure 1), but this option is not provided.

Each of the considered options can be used to solve narrowly focused problems, but all the considered approaches do not guarantee the work of the whole system. Mistakes made in the development of UAV

doors-2024: 4th Edge Computing Workshop, April 5, 2024, Zhytomyr, Ukraine

xckaytz@gmail.com (A. R. Petrosian); e_rvs@ukr.net (R. V. Petrosian); sasha_1904@ukr.net (O. M. Svintsytska)

0000-0003-0960-8461 (A. R. Petrosian); 0000-0002-0388-8821 (R. V. Petrosian); 0000-0002-2613-2437 (O. M. Svintsytska)

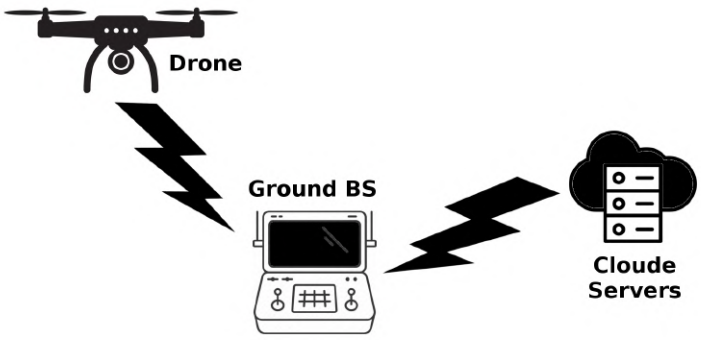

Figure 1: Simplified structure of the UAV peripheral computing network.

software can lead to emergencies or other unforeseen failures during operation. In this regard, the task of checking the reliability of the UAV's on-board computer software arises, which will simplify the maintenance, modernization and optimization of the software.

2. Theoretical background

Debugging and testing UAV software is a challenging task. The problem lies in the difficulty of generating a complete set of sensor signals and the impossibility of creating emergency situations on a real UAV, so special software and hardware are used to develop, test and debug UAV software.

Let us consider modern approaches used to verify the reliability of software during development, namely testing. According to the degree of code isolation, there are 4 levels of testing [1]:

- unit testing;
- integration testing;
- system testing;
- acceptance testing.

Unit testing is a level in which individual modules or components of a program (functions, methods, or classes) are tested independently of the rest of the program. This level of testing is intended to verify that each component works correctly. An important aspect of unit testing is the isolation of the component under test from other components of the application to ensure that bugs are detected and corrected locally within the component.

Integration testing is a level that is aimed at checking the interaction between different modules or components of the application. Integration testing checks how components interact with each other and how they interact with external systems, if any.

System testing – a level that is carried out at the final stage of development, when all components of the application are already integrated together and ready for testing in real conditions before the application is released into operation

Acceptance testing is a level aimed at verifying that the application meets the requirements of the customer or end user. It is usually performed by the customer or their representatives.

Testing ensures that:

- code quality;
- compliance with the stated requirements;
- optimization of computing resources;
- data security and integrity;
- time saving (searching for errors can take a lot of time);

- etc.

In many cases, the use of testing during the development of, for example, computer application software is sufficient to ensure the required level of reliability of the software, but not for firmware.

For effective and efficient testing of embedded software, a large number of testing methods, approaches, and tools are used, so in [2], a review and systematization of literature sources in this area was carried out.

Embedded systems often need to work in interaction with the environment, using sensors to obtain input data (temperature, pressure, etc.) [3], and UAVs are no exception. In [4], the main idea is to develop a test platform for mechanical flight simulation and better stabilization of multi-rotor UAVs using various feedback control algorithms, including PID controllers [5, 6]. This stand allows you to check and calibrate model parameters and perform real-time control of a multi-rotor UAV. A more functional stand for educational purposes is presented in [7]. As in the previous work, the test stand is a gyroscopic structure with three degrees of freedom, which provides pitch, roll, and yaw motion of the quadrotor. However, unlike the previous test stand, it allows translational movement, albeit with limitations (figure 2).

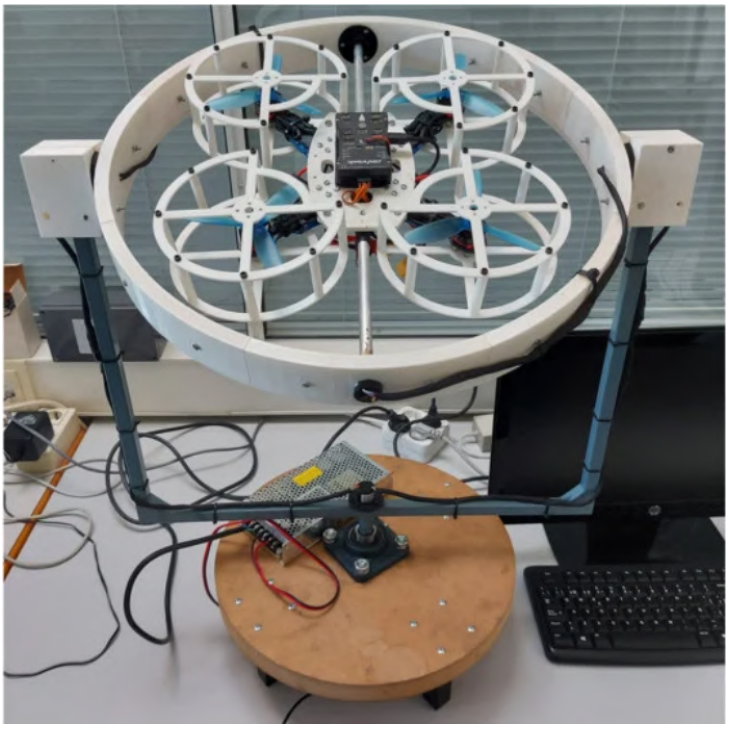

Figure 2: Experimental prototype with 3 degrees of freedom for testing control algorithms in a quadcopter.

One of the effective methods of testing embedded systems is simulation modeling. Simulation modeling is a research method that uses models that describe real-world processes. Using such models, it is possible to test the real system without exposing it to danger. Simulation modeling is most often used when it is impossible to conduct an experiment on a real object or it is necessary to simulate the behavior of a real system in time.

The firmware test in UAV simulation can be performed in three places: on a host computer, using code compilation for the source platform; in an emulator on a host computer, such as QEMU; on a real flight controller, using cross-compilation of code for the target hardware. The first two options are used when performing SIL (Software in the Loop) simulation [8, 9]. The advantage of SIL is that it is easy to organise, as no additional hardware is required. SIL allows developers to perform firmware simulation in the early stages of development, even before it is integrated into the target hardware. The third option is used when performing HIL (Hardware in the Loop) simulation [8, 10, 11]. HIL simulation

involves the use of target equipment that brings the system's operation as close as possible to real conditions. Recently, another approach to testing has emerged – SIH simulation [12]. SIH simulation is an alternative to HIL simulation. In this case, everything works on the onboard computer, and the PC is used only to display the virtual UAV.

3. Result

As mentioned above, various testing methods and tools are used to ensure the reliability of embedded system software. The main difficulty is testing the hardware components of the embedded system, so additional hardware and/or software tools are being developed. Figure 2 shows a test stand that provides verification of the functioning of algorithms in the on-board computer, the operation of sensors, motors, etc. However, it is obvious that this stand limits the UAV's movement in space. Another approach that allows you to test a full-fledged UAV flight is to use HIL simulation. The flight is performed in a 3D environment on a computer (e.g., Gazebo, jMavSim, or another). Figure 3 shows a diagram of such a HIL simulation system of the PX4 autopilot software [11].

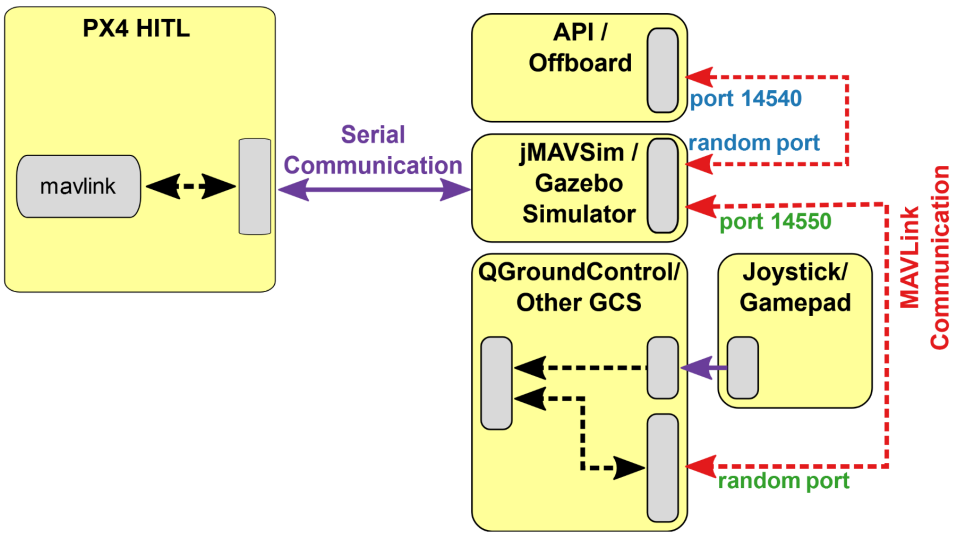

Figure 3: Block diagram of HIL simulation in PX4.

Obviously, this approach allows testing only the operation of the on-board computer, while other hardware nodes are virtual. These virtual nodes interact with the on-board computer using the MAVLink protocol [13]. Figure 4 shows the appearance of the jMavSim 3D environment.

To ensure firmware testing in the on-board computer, you need to use mock objects. The purpose of mock objects is to replace the objects that are tested in the program code with equivalent debugging objects. Creating mock objects can be associated with some difficulties, for example, using interrupts from a specific communication interface that a real sensor uses. The disadvantage of this approach is that the firmware during testing will not correspond to its final implementation. The use of SIH simulation (figure 5) improves the situation when testing UAVs [12], but does not eliminate most of the problems that HIL simulation has.

To address these shortcomings, it is proposed to implement a test platform that uses the original firmware of the on-board computer. How to implement it? Consider a simplified diagram of any sensor (figure 6).

The sensor consists of:

- a sensing element that is directly related to the measured value;
- transducer, which serves to convert the measured value into another value convenient for transmission, processing and storage.

Figure 4: jMavSim 3D environment window.

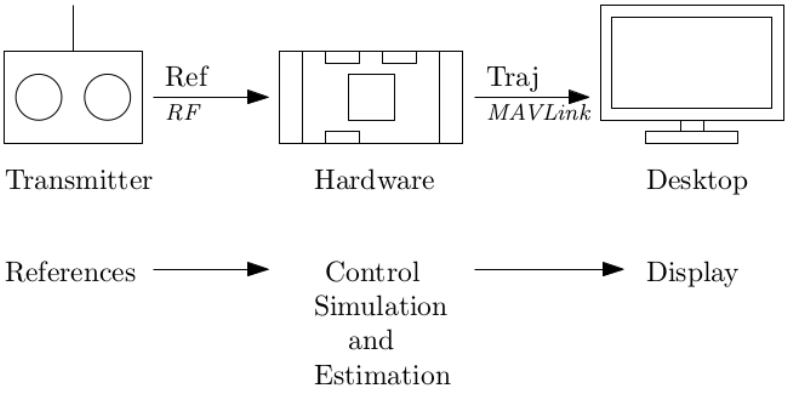

Figure 5: Block diagram of SIH simulation in PX4.

When using any simulation option (HIL, SIH), the virtual sensor is fully implemented in a 3D environment [14, 15]. In this case, the question arises: how to transmit information to the on-board computer? The MAVLink protocol is mainly used, although it is not necessary, for which mock objects are implemented in the firmware of the on-board computer. Thus, the data of the virtual sensor is processed on the side of the on-board computer. Obviously, in this case, it is necessary to complicate the architecture of the autopilot software, as well as to provide for different compilation scenarios. However, the KISS design principle states that there is no need to overcomplicate, each node should perform only a specific task.

The on-board computer already uses communication interfaces with sensors (mainly I2C, SPI), so it is

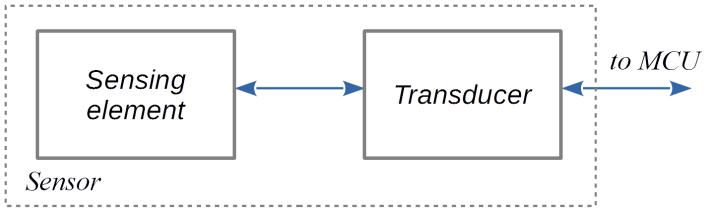

Figure 6: Simplified block diagram of a sensor.

logical to use them to solve the problem. In this case, the on-board computer will access real hardware nodes, but this is problematic because these interfaces are not available in a personal computer. On the other hand, it is necessary to use a 3D environment to organise safe flights when testing UAV algorithms.

Obviously, a node is needed to connect the on-board computer with the 3D environment on a PC to transmit navigation and control information to the UAV. From the above, it follows that a sensor and actuator simulator (SAS) is needed, where the software model will correspond to the hardware nodes. The architecture of the proposed test platform is shown in figure 7.

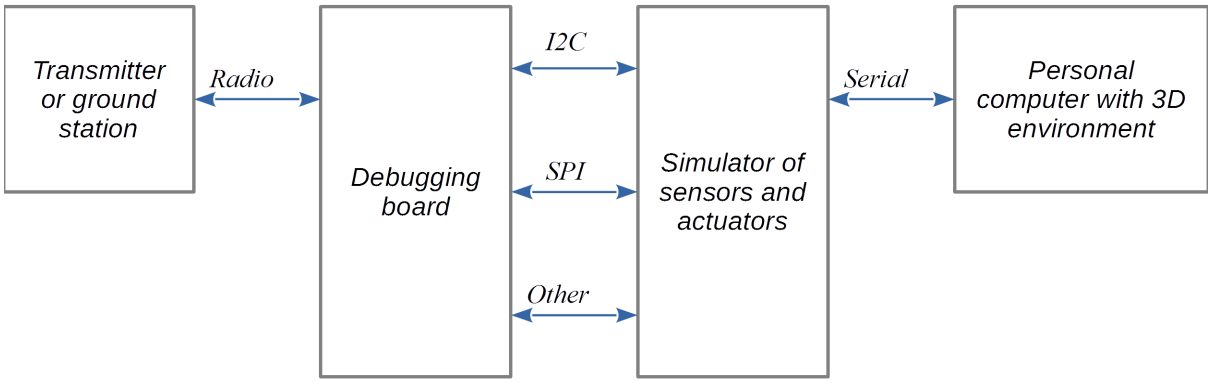

Figure 7: Architecture of the test platform.

When using this approach, in fact, the SAS will perform the functions of a sensor converter (figure 6), and only the sensing element will be used in the 3D environment. The information transmitted from the 3D environment will be wrapped in a software model of the sensor (in our case, MPU-6050 [16, 17]). Figure 8 shows some registers of the MPU-6050 accelerometer and gyroscope.

A debugging board is required for UAV simulation. The most popular microcontrollers for the on-board computer are STM32F405RGT6 and STM32F722RET6. Unfortunately, STMicroelectronics does not produce a debugging board based on the STM32F405RGT6 microcontroller, so you need to use a third-party debugging board, such as the Core405R from Waveshare or the STM32F405 Core Board from WeAct. The NUCLEO-F722ZE debug board can be used to simulate an on-board computer based on the STM32F722RET6 microcontroller. The SAS can be performed on any debugging board that has three I2C and SPI interfaces each (the number of interfaces is determined by the number of interfaces in the above-mentioned microcontrollers). In our case, we used the STM32G431 Core Board from WeAct. The configuration of the SAS for each sensor is performed similarly to its prototype, and the data registers are filled with information coming from a personal computer with a 3D environment.

The software of the SAS is implemented in the C++ programming language. For sensors with an I2C interface, the basis is an abstract class shown in the screen (figure 9).

To configure the SAS, we developed a simple utility in the C++ programming language using the QT framework (figure 10). For each channel, the sensor model, interface, and variable part of the address,

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0D	13	SELF_TEST_X	R/W	XA_TEST[4-2]			XG_TEST[4-0]				
0E	14	SELF_TEST_Y	R/W	YA_TEST[4-2]			YG_TEST[4-0]				
0F	15	SELF_TEST_Z	R/W	ZA_TEST[4-2]			ZG_TEST[4-0]				
10	16	SELF_TEST_A	R/W	RESERVED		XA_TEST[1-0]		YA_TEST[1-0]		ZA_TEST[1-0]	
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	-	-	-	FS_SEL [1:0]		-	-	-
1C	28	ACCEL_CONFIG	R/W	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]				
23	35	FIFO_EN	R/W	TEMP _FIFO_EN	XG _FIFO_EN	YG _FIFO_EN	ZG _FIFO_EN	ACCEL _FIFO_EN	SLV2 _FIFO_EN	SLV1 _FIFO_EN	SLV0 _FIFO_EN
24	36	I2C_MST_CTRL	R/W	MULT _MST_EN	WAIT _FOR_ES	SLV_3 _FIFO_EN	I2C_MST _P_NSR	I2C_MST_CLK[3:0]			
25	37	I2C_SLV0_ADDR	R/W	I2C_SLV0 _RW	I2C_SLV0_ADDR[6:0]						
26	38	I2C_SLV0_REG	R/W	I2C_SLV0_REG[7:0]							
27	39	I2C_SLV0_CTRL	R/W	I2C_SLV0 _EN	I2C_SLV0 _BYTE_SW	I2C_SLV0 _REG_DIS	I2C_SLV0 _GRP	I2C_SLV0_LEN[3:0]			
28	40	I2C_SLV1_ADDR	R/W	I2C_SLV1 _RW	I2C_SLV1_ADDR[6:0]						
29	41	I2C_SLV1_REG	R/W	I2C_SLV1_REG[7:0]							
2A	42	I2C_SLV1_CTRL	R/W	I2C_SLV1	I2C_SLV1	I2C_SLV1	I2C_SLV1	I2C_SLV1_LEN[3:0]			

Figure 8: MPU-6050 register map.

```
class I2C {
private:
    // Sensor address on the I2C bus
    uint32_t i2c_address;
public:
    // Constructor
    I2C(uint32_t address) : i2c_address(address) {}
    // Destructor
    virtual ~I2C() {}
    // Function returns the address of the sensor on the I2C bus
    uint32_t get_i2c_address() {return i2c_address;}
    // Function sets the address of the sensor on the I2C bus
    void set_i2c_address(uint32_t address) {i2c_address=address;}
    // Function is called when accessing the sensor via I2C bus
    virtual bool i2c_connect() = 0;
    // Function is called when reading data bytes from the sensor
    virtual void i2c_disconnect() = 0;
    // Function is called when a data byte is written to the sensor
    virtual uint8_t i2c_read() = 0;
    // Function is called when communication with the sensor is completed
    virtual bool i2c_write(uint8_t data) = 0;
};
```

Figure 9: Abstract class of sensors with I2C interface.

as well as noise parameters are configured in accordance with expression (1) [14]:

$$y_m = y + b + w_y, \tag{1}$$

where y is the original value, b is the current offset, and w_y is a white Gaussian noise with zero mean.

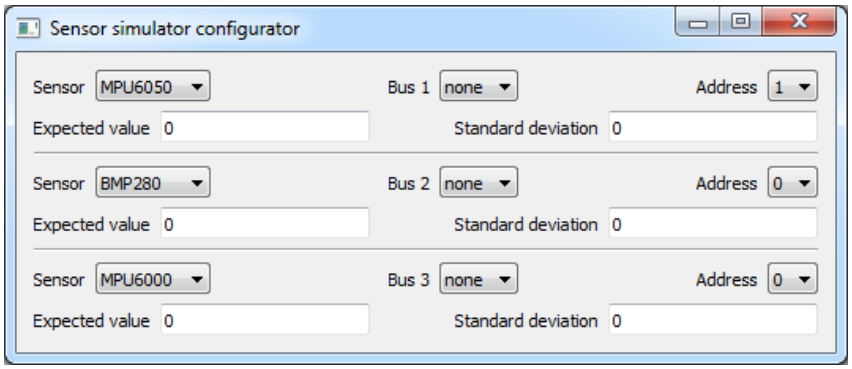

Figure 10: Appearance of the configurator of the sensor and actuator simulator.

4. Conclusion

The article considers a test platform for hardware modelling of an unmanned aircraft's onboard computer. Various testing methods and tools are used to ensure the reliability of embedded system software. The main difficulty is testing the hardware components of the embedded system, so additional hardware and software tools are being developed.

To accomplish this task, a comparative analysis of existing testing approaches was carried out to identify advantages and disadvantages. It was found that both mechanical test stand and simulation modelling in a 3D environment are used: SIL and HIL. A mechanical test rig is heavy and restricts the UAV's movement in space. Existing SIL and HIL simulators do not require additional equipment, but they do not take into account the actual operation of hardware components.

An improved version of SIH simulation is proposed. The proposed variant is based on a sensor and actuator simulator that ensures the operation of the original UAV firmware and allows for the interconnection of the onboard and personal computer. The sensor and actuator simulator was built on the basis of the Core Board STM32G431 debugging board from WeAct. To configure the sensor and actuator simulator, a small utility was developed in C++ using the QT framework.

Acknowledgments

Conceptualization, Arsen Petrosian and Ruslan Petrosian; software, Arsen Petrosian; hardware, Ruslan Petrosian; investigation, Arsen Petrosian; resources, Oleksandra Svintsytska; data curation, Oleksandra Svintsytska; writing-review and editing, Arsen Petrosian.

We would like to express our gratitude to our relatives who supported and helped us. We would also like to express our gratitude to all the organizers of the conference "doors-2024: 4rd Edge Computing Workshop" conference, especially Tetiana Vakaliuk.

References

[1] M. Aniche, Effective Software Testing: A developer's guide, Simon and Schuster, 2022.

[2] V. Garousi, M. Felderer, Ç. M. Karapıçak, U. Yılmaz, Testing embedded software: A survey of the literature, Information and Software Technology 104 (2018) 14–45. doi:10.1016/j.infsof.2018.06.016.

[3] A. Banerjee, S. Chattopadhyay, A. Roychoudhury, On testing embedded software, in: Advances in Computers, volume 101, Elsevier, 2016, pp. 121–153. doi:10.1016/bs.adcom.2015.11.005.

[4] M. Hancer, R. Bitirgen, I. Bayezit, Designing 3-DOF hardware-in-the-loop test platform controlling multirotor vehicles, IFAC-PapersOnLine 51 (2018) 119–124. doi:10.1016/j.ifacol.2018.06.058.

[5] R. V. Petrosian, I. A. Pilkevych, A. R. Petrosian, Algorithm for optimizing a PID controller model based on a digital filter using a genetic algorithm, CEUR Workshop Proceedings 3374 (2023) 97–111. URL: https://ceur-ws.org/Vol-3374/paper07.pdf.

[6] M. Zhang, C. Xu, D. Xu, G. Ma, H. Han, X. Zong, Research on improved sparrow search algorithm for PID controller parameter optimization, Bulletin of the Polish Academy of Sciences Technical Sciences 71 (2023) e147344–e147344. doi:10.24425/bpasts.2023.147344.

[7] U. Veyna, S. Garcia-Nieto, R. Simarro, J. V. Salcedo, Quadcopters Testing Platform for Educational Environments, Sensors 21 (2021) 4134. doi:10.3390/s21124134.

[8] C. Coopmans, M. Podhradský, N. V. Hoffer, Software- and hardware-in-the-loop verification of flight dynamics model and flight control simulation of a fixed-wing unmanned aerial vehicle, in: 2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), 2015, pp. 115–122. doi:10.1109/RED-UAS.2015.7440998.

[9] K. D. Nguyen, C. Ha, J. T. Jang, Development of a New Hybrid Drone and Software-in-the-Loop Simulation Using PX4 Code, in: D.-S. Huang, V. Bevilacqua, P. Premaratne, P. Gupta (Eds.), Intelligent Computing Theories and Application, Springer International Publishing, Cham, 2018, pp. 84–93. doi:10.1007/978-3-319-95930-6_9.

[10] K. D. Nguyen, C. Ha, Development of Hardware-in-the-Loop Simulation Based on Gazebo and Pixhawk for Unmanned Aerial Vehicles, International Journal of Aeronautical and Space Sciences 19 (2018) 238–249. doi:10.1007/s42405-018-0012-8.

[11] Hardware in the Loop Simulation. PX4 Autopilot User Guide, 2023. URL: https://docs.px4.io/main/en/simulation/hitl.html.

[12] Simulation-In-Hardware. PX4 Autopilot User Guide, 2023. URL: https://dev.px4.io/v1.9.0_noredirect/en/simulation/simulation-in-hardware.html.

[13] Protocol Overview. MAVLink Developer Guide, 2023. URL: https://mavlink.io/en/about/overview.html.

[14] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, O. von Stryk, Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo, in: I. Noda, N. Ando, D. Brugali, J. J. Kuffner (Eds.), Simulation, Modeling, and Programming for Autonomous Robots, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 400–411. doi:10.1007/978-3-642-34327-8_36.

[15] A. I. Hentati, L. Krichen, M. Fourati, L. C. Fourati, Simulation Tools, Environments and Frameworks for UAV Systems Performance Analysis, in: 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), 2018, pp. 1495–1500. doi:10.1109/IWCMC.2018.8450505.

[16] TDK, MPU-6000 and MPU-6050. Product Specification, 2013. URL: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[17] TDK, MPU-6000 and MPU-6050. Register Map and Descriptions, 2013. URL: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf.