UDC 004.042

A. Akhaladze, O. Lisovychenko

SYNCHRONIZATION OF FLIGHT TRAJECTORIES BASED ON THE ARCHITECTURE OF THE "INTERNET OF THINGS" IN THE IMPLEMENTATION OF SWARM MANAGEMENT

Abstract: An implemented approach using the architecture of the "Internet of Things" (IoT) [1] to solve the problem of loss of synchronization by members of the drone swarm when working out a determined flight plan at the same time. To synchronize the execution of trajectories, the proposed approach uses the analysis of telemetry data of each drone with a given discreteness as feedback swarm control. In addition, the basic requirements for the speed of feedback from drones and the impact of feedback time on the dynamics of swarm control, in general, are identified.

Keywords: Internet of Things, IoT, swarm problem, trajectory target point, trajectory synchronization, feedback

Introduction

The IoT architecture has technical prospects for use in building a swarm management system, as the main advantage of using this approach in implementation is the speed of data exchange between end devices and the decision node, which will provide feedback faster.

The problem that needs to be solved is the loss of synchronization during the flight of unmanned devices (hereinafter, drones). This can lead to an asynchronous swarm trajectory (several drones flying at the same time) and lead to abnormal control situations. The use of IoT to solve swarm problems allows to organize of only a technically limited number of drones, and drones are sent the same commands, executed by each individual drone. However, the large number of different models of drones that can be combined in a swarm creates the problem of its desynchronization.

When the condition is set to execute commands by all participants in the swarm synchronously, ie when all drones have a determined position in space at a certain point in time (synchronization point), there is a significant error at a given point, which accumulates over time.

Problem statement

It is necessary to implement a mechanism to solve the problem of loss of synchronization of flight trajectories of drones united in a swarm, working out a determined flight plan at the same time, through the use of IoT architecture and logical synchronization node. Eliminate the accumulation of cumulative exit error at a given point to the value of the error of the elementary trajectory through the use of synchronization points of flight trajectories of the whole swarm.

Solving the tasks

Implementation of synchronization service in architecture using IoT will solve the problem of drone synchronization loss in the swarm, and the decomposition of the flight plan on the trajectory and the introduction of synchronization points will avoid the accumulation of cumulative error [3] when working out the flight plan

The magnitude of the error depends on the technical characteristics of each drone, their number in the swarm, and the duration of the trajectory, the more individual teams are included in the trajectory, the faster the error increases. The problem of swarm desynchronization leads to the impossibility to program the synchronous behavior of the whole swarm [6], and the accumulated error of exit to a given point creates a risk of the intersection of flight trajectories in motion - collisions. The drones combined through the IoT hub receive commands at the same time, start working out the commands when they are received and return the hub data of their telemetry every second [4]. In real-time control mode, the error of behavior of three drones in one swarm is not visible to the naked eye due to the delay before the next movement of the joystick, and through sight, the feedback of swarm control is realized. When used in a swarm of 9 drones of different models, the trajectories of drones and their time of exit to each point of the trajectory show the effect of "waves". When programming the trajectories for a swarm, it is impossible to predict the magnitude of the error of exit to a given point of a particular drone, as it depends on external factors and characteristics of other drones in the swarm.

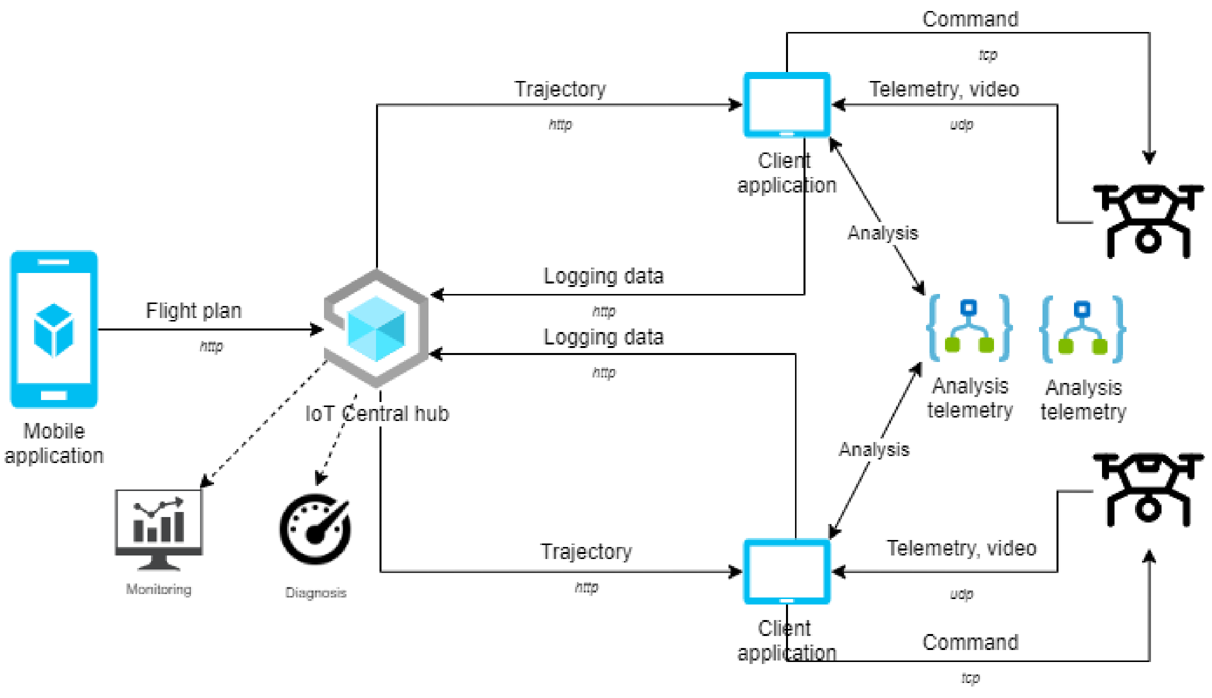

Figure 1. Architectural scheme of the swarm management system

IoT Central Hub (Fig. 1) combines two drones that receive a list of parts of the trajectory simultaneously.

Client application in Fig. 1 is a running instance of a node that transforms the sectors of the trajectory into drone commands of a particular model and the function of routing data between the node and the drone. The experiment was performed on two Tello Talent Robomaster drones from DJI, which, receiving a simple trajectory (Fig. 2), was performed iteratively and demonstrated the accumulation of error for 3 iterations in 1 second.

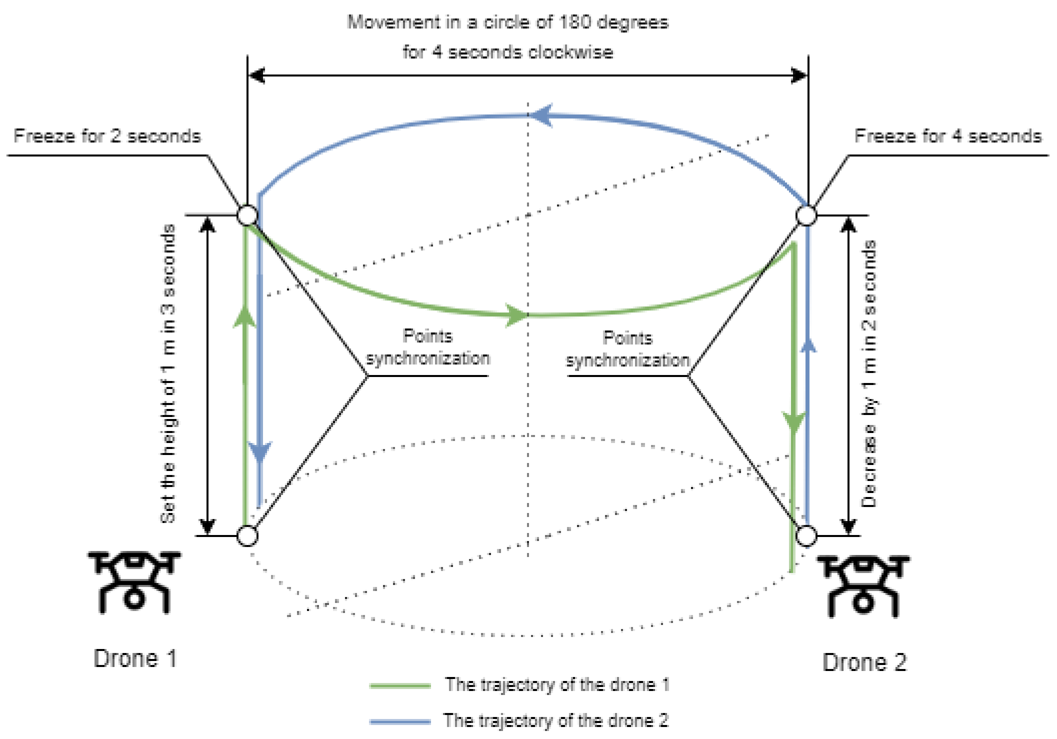
Movement in a circle of 180 degrees for 4 seconds clockwise
Freeze for 2 seconds
Freeze for 4 seconds
Set the height of 1 m in 3 seconds
Decrease by 1 m in 2 seconds
Points synchronization
Points synchronization
Drone 1
Drone 2
The trajectory of the drone 1
The trajectory of the drone 2

Figure 2. Flight plan for execution by drone

The drones are simultaneously sent a given trajectory through the hub, which the proxy application transforms into drone commands of a certain model [2] and sends on a timer with a specified execution time. For the used drones the algorithm is shown in fig. 3.

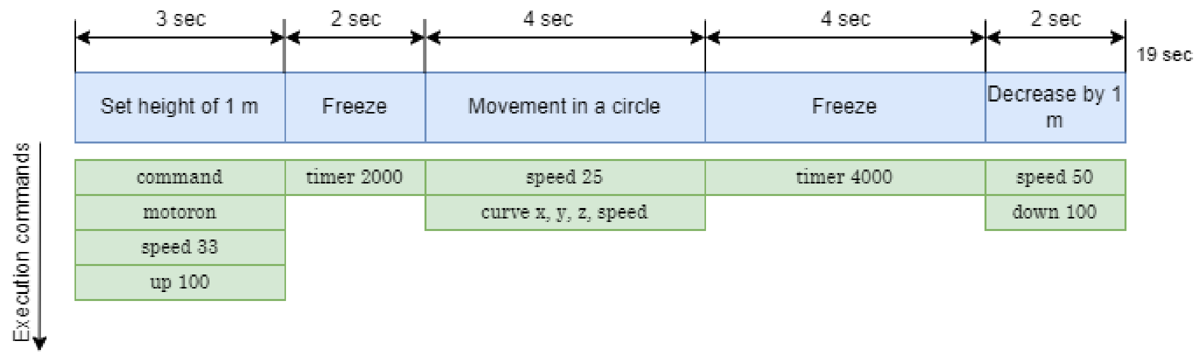

Figure 3. Decomposition of a given trajectory

Following a given trajectory cyclically by two drones connected to the IoT hub, real-time telemetry data is recorded in the log, which is registered and stored by the IoT hub. Already on the third iteration, the accumulated error leads to the reproduction of the problem of desynchronization. That is, one drone accumulates an exit error at a given point, which cannot be compensated without the use of a synchronization mechanism.

We visualize the data obtained on the basis of the analysis of the recorded logs by the hub after the trajectory for 3 iterations.

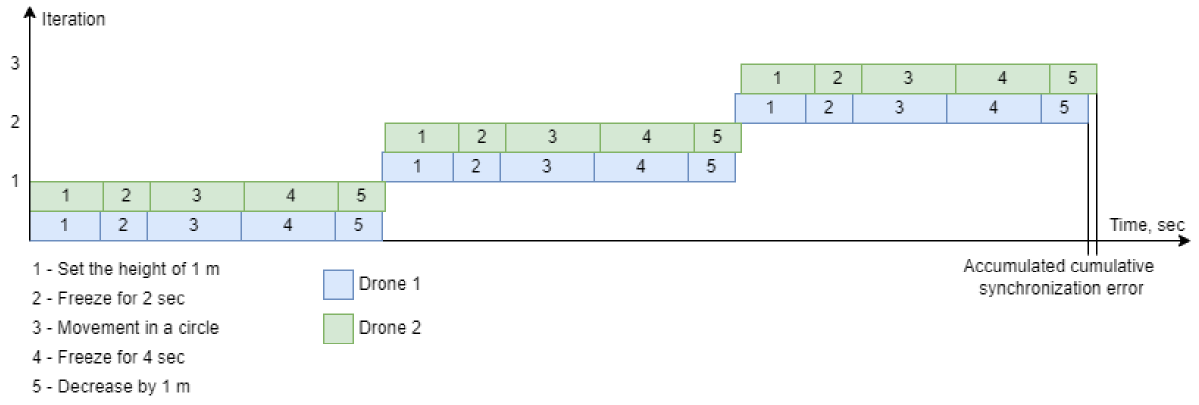

Figure 4. Time synchronization error

Cumulative synchronization error accumulates as a result of trajectory execution in a swarm with two drones of the same model (Fig. 4) and leads not only to asynchronous swarm trajectory execution but also to trajectory intersection when performing on more iterations. Adding more drones to a swarm increases the likelihood of trajectory intersections and therefore collisions. The graph of the accumulated time error of synchronization of two drones combined in a swarm using the IoT architecture is shown in Fig.

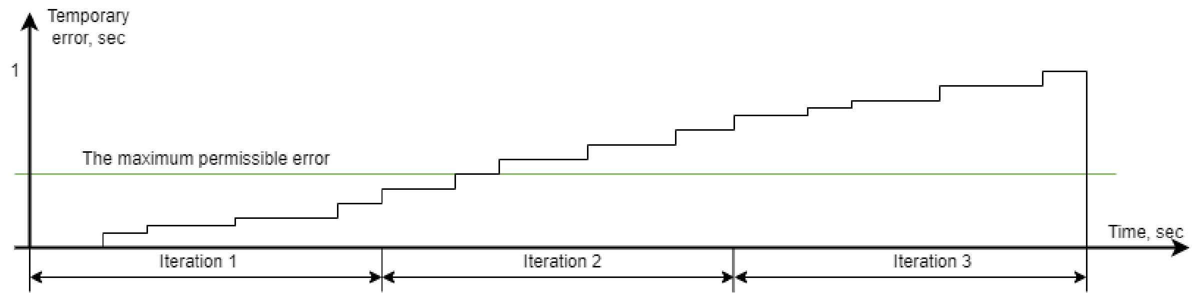

Figure 5. Time error accumulation graph

To implement the synchronization mechanism, we will add a message broker and its client [5], which receives the results of real-time telemetry analysis to confirm the exit to a given point of each swarm member, and sends the next part of the trajectory for execution only when all swarm members reach the point of the previous part of the trajectory. The architectural scheme of the system is shown in Fig. 6.

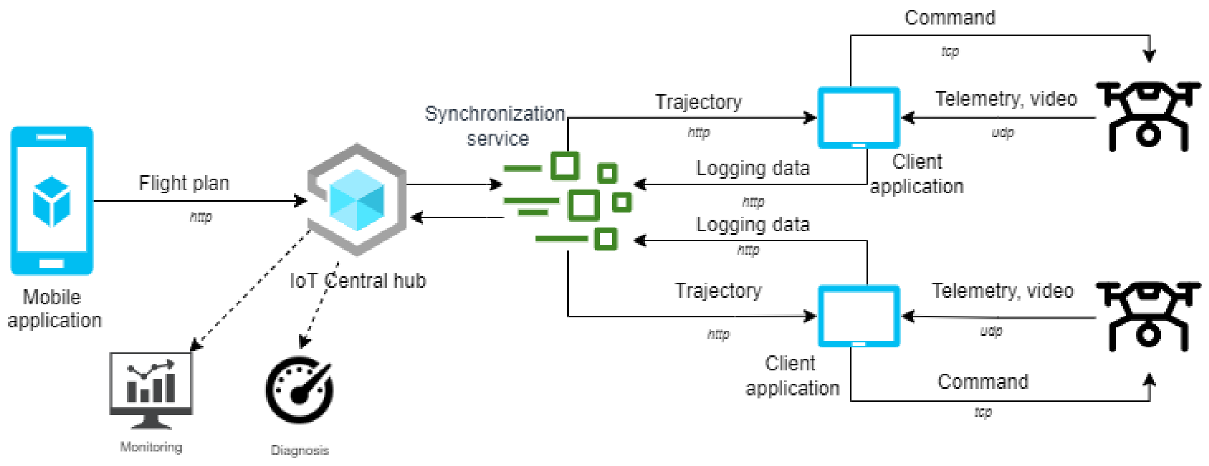

Figure 6. Architectural scheme of the swarm management system with the trajectory synchronization service

In fig. 6, initiates the task of the mobile application, which transmits the IoT hub selected scenario of swarm behavior (flight plan). The flight plan may consist of a large number of trajectories that must be performed by the swarm synchronously. Next, the IoT Hub, receiving the flight plan, sends it to the synchronization service, which implements the logic of dividing the plan into components of the trajectory and creates a list of synchronization points at the boundaries of the trajectory. The use of the hub allows you to save logging data in real time, which allows not only to build dashboards but also to diagnose manually ex post facto or automated in real time. Monitoring and diagnostic mechanisms will be considered in further work.

The client application, receiving the trajectory, transforms each of them into a list of commands that are sent directly to the drone. The use of an intermediate software level in the form of a client application allows you to encapsulate the logic of the transformation of the trajectory directly into the list of commands that can perform the drone. In addition, this approach allows you to have specific implementations for different types of drones, as the communication API in different models of drones may be different. We will add that in case of entering into a swarm of various models of the drone it will be rational to bring the list of commands to abstraction and to place it in the public assembly. In this case, when registering a new member of the swarm, it will only be necessary to describe the mechanism of transformation of public library commands into commands of a specific drone model [7]. In this example, we use two drones of the same model, so the client application in the schema is implemented as two separate instances with identical logic, deployed on two physical servers. The client application is also responsible for collecting and transforming logging data consisting of telemetry data in text format and video stream data. The command delivery channel is tcp, because the guarantee of message delivery with the command is critical, and the log data is used udp channel, as the system is more important than the relevance of the drone, and the probable loss of the packet is offset by the next.

By analyzing telemetry data in real time and using the synchronization service, we can solve the problem of swarm desynchronization. The flight plan is split on the trajectory by the synchronization service and the next part refers to testing through the client application to each drone only when all members of the swarm have reached the target point. This will increase the time to complete the flight plan (it will be equal to the time of the slowest drone), but this way we solve the problem of desynchronization of the swarm and avoid the intersection of trajectories due to the accumulation of error at the end point of each trajectory.

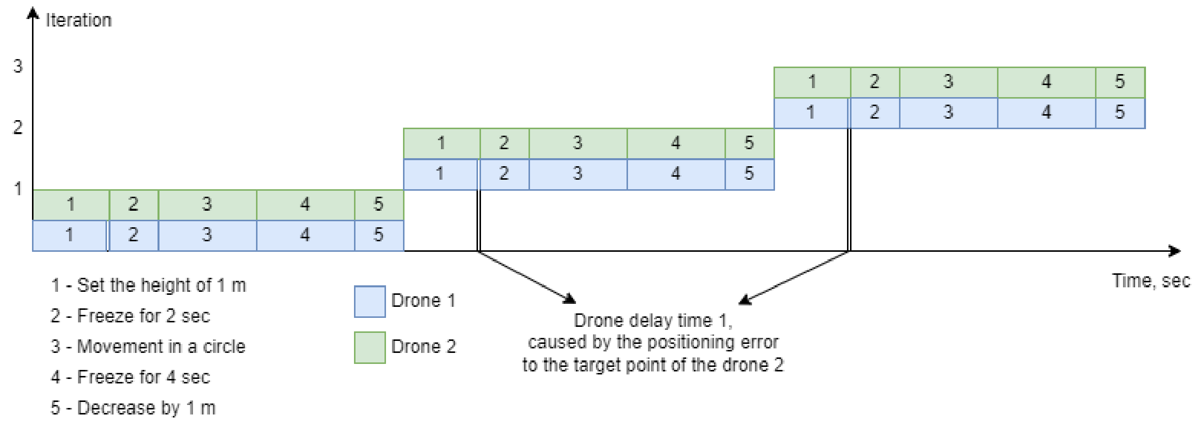

Figure 7. Temporal division of synchronized trajectory execution by two drones

In fig. 7 we have a distribution that shows the absence of accumulation of exit error at a given point when working out a flight plan in a swarm of two drones in three iterations.

In Figure 8 the total error when working out the flight plan for three iterations does not exceed the maximum allowable.

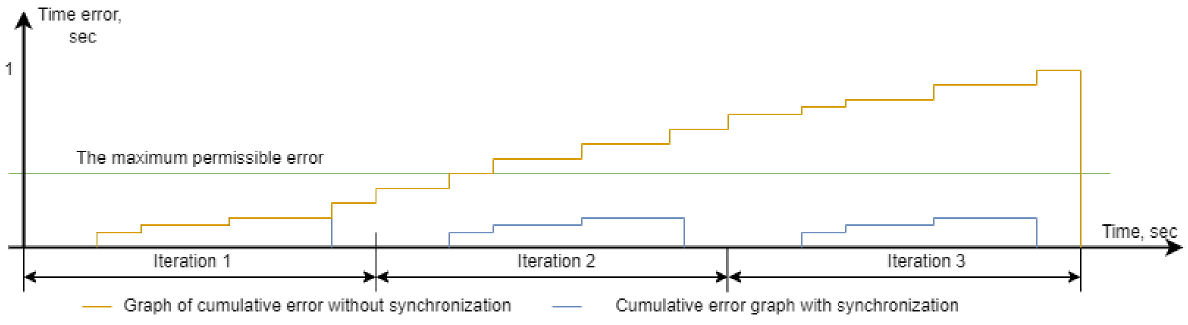

Figure 8. Graph of time error accumulation when using the synchronization service

The diagram does not show the time required to analyze telemetry data. The empirically established requirement for the speed of a mathematical apparatus that analyzes telemetry data must be less than the time of the discrete time interval of analysis of telemetry data. Such requirements are satisfied by the use of server-free functions, which in addition to runtime have advantages in the use of energy at unevenly distributed load, which we have in this system.

Conclusions

The use of IoT architecture to control a swarm of drones has technical advantages [1], but the physical implementation of the system revealed the problem of loss of synchronicity of flight trajectories by each member of the swarm due to cumulative error of trajectory endpoint, depending on the technical characteristics of each drone. The accumulated error does not allow to work out a trajectory synchronously, and at the performance of a trajectory cyclically can lead to the collision of drones in a swarm. To solve this problem, a synchronization service has been added to the swarm management system architecture, which performs the function of decomposing the flight plan into elementary trajectories by introducing synchronization points and validating the endpoint of each swarm member.

The proposed approach to solving the problem of drone desynchronization avoids the accumulation of the error of exit to a given point by individual members of the swarm and avoids collisions when working out complex trajectories.

Given the requirements for the speed of the mathematical apparatus for calculating the target point of the trajectory, the use of serverless functions looks promising.

REFERENCES

1. Akhaladze A. E. Use of IoT for synchronization of drone flight trajectories // Adaptive automatic control systems. 2021. № 39. C.20-26 URL: http://asac.kpi.ua/article/view/247381
2. DJI TelloTallent drone API specifications
3. Lianos, M. and Douglas, M. (2000) Dangerization and the End of Deviance: The Institutional Environment. British Journal of Criminology, 40, 261-278. http://dx.doi.org/10.1093/bjc/40.2.261 [Citation Time(s):1]
4. Ferguson, T. (2002) Have Your Objects Call My Object. Harvard Business Review, June, 1-7. [Citation Time(s):1]
5. Nunberg, G. (2012) The Advent of the Internet: 12th April, Courses. [Citation Time(s):1]
6. L. Roselli, N. B. Carvalho, F. Alimenti, P. Mezzanotte, G. Orecchini, M. Virili, et al., "Smart surfaces: Large area electronics systems for Internet of Things enabled by energy harvesting", *Proc. IEEE*, vol. 102, pp. 1723-1746, Nov. 2014.
7. L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A survey", *Comput. Netw.*, vol. 54, no. 15, pp. 2787-2805, Oct. 2010.