International Journal of Transportation Engineering and Technology
2015; 1(1): 10-14
http://www.sciencepublishinggroup.com j/ijtet
doi: 10.11648/j.ijtet.20150101.12

Applying Multithreading for Multi-Rotors with FlyMaple

Nguyen Anh Quang, Ngo Khanh Hieu

Faculty of Transportation Engineering, Ho Chi Minh City University of Technology, Ho Chi Minh City, Viet Nam

Email address:
anh-quang.nguyen.1@ens.etsmtl.ca (N. A. Quang), ngokhanhhieu@hcmut.edu.vn (N. K. Hieu)

To cite this article:
Nguyen Anh Quang, Ngo Khanh Hieu. Applying Multithreading for Multi-Rotors with FlyMaple. *International Journal of Transportation Engineering and Technology*. Vol. 1, No. 1, 2015, pp. 10-14. doi: 10.11648/j.ijtet.20150101.12

Received: January 29, 2016; **Accepted**: February 13, 2016; **Published**: March 11, 2016

Abstract: With the development of science and technology, the control boards nowadays not only have a higher working clock rate but also supports multithreading. Like the effects of the multithreading processor with the development of computer sciences, control boards supporting multithread is promised to change the world of Unmanned Vehicles. This article focuses on the application of multithread for multi-rotors, a new section which has been recently researched by many universities and developers in the world. After an overview about multithread and the related projects, this article will present the utilization of multithread with FlyMaple, a new generation of control board which has some important advantages comparing to the older generations.

Keyword: Multithreading, Multi-Rotor, FlyMaple, Ardu Pilot, Free RTOS

1. Introduction

Multi-rotors is one of the most interested types of drone because of their simplicity in hardware but complexity in control algorithms. Nevertheless, solving the control algorithms to get the outputs to control the rotors is not the only task done by the control board, especially the microcontroller. The microcontroller is the heart of the control board and therefore the whole system, which is in charge for every single task, from getting the values from the sensors, communicating with other hardware systems to listening and answering the controller or the ground control station. The more requirements of the systems, the more calculation have to be done as well as the more tasks have to be taken care of. Therefore processing ability of the microcontroller have to be increased. At first, we increased the MCU frequency of the microcontroller, which will boost up the board and increase its capacity. However, there are another solution for this problem. Instead of increasing the clock rate of the microcontroller, which is limited at some rates due to technologies and other limitations [1], a microcontroller supporting multithread can increase the ability of the whole system by reducing the polling I/O time needed for every tasks. This benefits has been recognized and experienced, some of them will be discussed in the next part of this article. Multithread is promised to be the trend of the future since it will push the limitations of the control board to a higher level and open new opportunity for developers and researchers.

2. Main

2.1. Threading and Multithreading

In general, for an application like multi-rotors, there are many required tasks to make the system operate properly. In order to increase the performance and optimize the efficiency of the microcontroller, these tasks is separated into threads, which are *"small tasks that can be run independently"* [2]. There are several ways that the microcontroller handles these threads like preemptive or non-preemptive, FIFO or any other scheduling solutions. Since these threads have to be handled repeatedly in time, they are handled in a procedure called cyclic. Fig. 1 presents a simple description for a single threading.

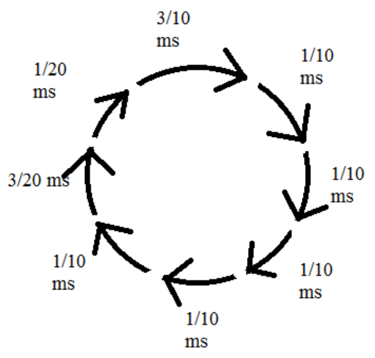

Figure 1. *Single threading cyclic.*

By default, a thread might include several steps, such as getting data, reading values or calculating… which can include or not include the using the resources of the OS. In some cases, one thread is called but then paused to wait for some events, for example the data of the sensors. This pause wastes an amount of time of the system since the microcontroller has nothing to do. The multithreading operating systems made it possible for one thread to run while the thread above is waiting [2]. In other words, multithreading system will try to fill the idle time of polling I/O with other threads, so it can optimize the efficiency of a microcontroller. Because there are tasks that can depend on the results of the other tasks, for example sending the PWM to the rotors must be proceed after calculating the PWM values, while other tasks are not, tasks and threads controlling a multi-rotors or any system can be put into different cyclics, which can run threads concurrently. Multithreading can be described as shown in Fig. 2.

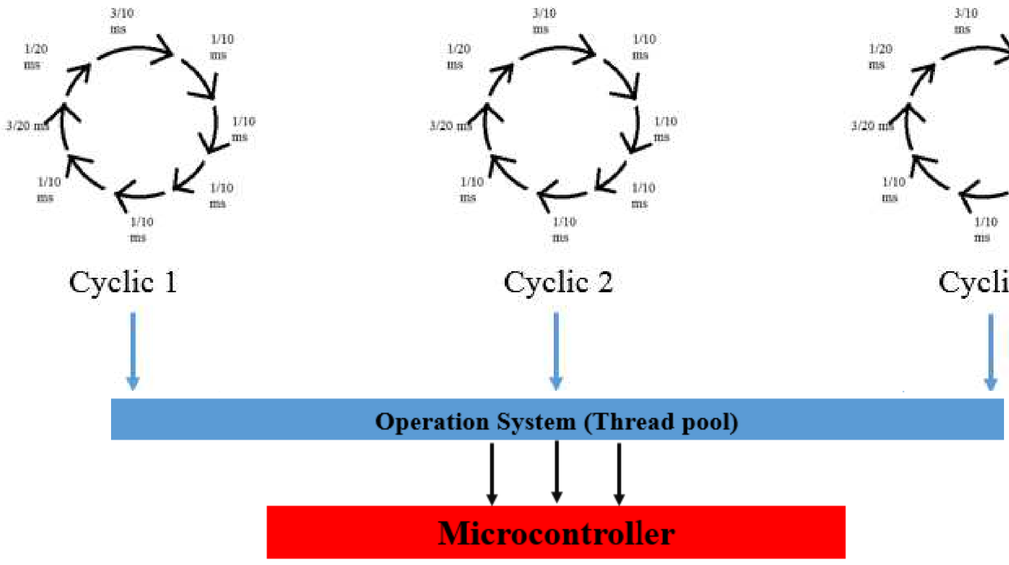

Figure 2. *Multithreading cyclic.*

Table 1. *Comparison between single thread and multithread.*

	Single Thread	Multithread
Advantages	• Easy to program and threading • Avoidable some scheduling problems	• Increase the performance of the microcontroller • Avoid overheating when increase the timer clock rate. • Safe money from hardware in case of building a system.
Disadvantages	• Cannot get the maximum efficiency of the system.	• Overhear may happen while switching between threads • Hard to program and scheduling • Some problems with scheduling might happen such as deadlock, order violation, etc. [2]

In case of controlling a multi-rotors, multithreading can be implemented as shown in Fig.3.

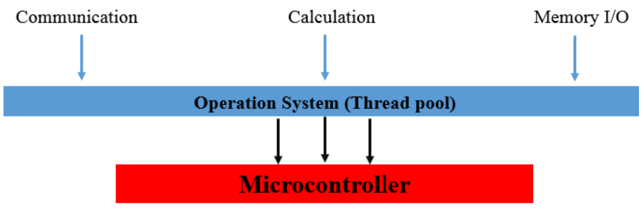

Figure 3. *Application of multithreading in controlling UAV.*

Fig. 3 introduces a simple multithreading system for controlling an UAV with 3 cyclics, each of them are independent with others. *Communication* reads values from external system such as the GPS, RC or Ground Control Station. *Calculation* handles the internal sensors and the calculating procedure. *Memory I/O* is in charge of interaction with the memory system and memorizing the flight characteristics. This structure has been successfully implemented in ArduPilot, a popular framework to create the firmware for the UAVs. Depending on the desires and the specifications of the systems, the number of cyclics can increase or decrease. However, the principle of the system remains the same.

There are both pros and cons of multithreading comparing to the original single threading. Table 1 gives some of these points.

Even though multithreading has some limits and problems as shown above, the benefits of this solution comparing to single thread has been proved in many researches [3] [4], therefore, in case of a real-time, complex system such as a multi-rotors, multithread is the key to adapt the increasing requirements of users and developers.

2.2. Multi-Rotors Projects Using Multithreading

At this time, using multithread for multi-rotors has been implemented and experimented in various projects. In order to experiment different algorithms and solutions for indoor quadrotor navigation, G. Angeletti and his team used a Gumstix board, which has a 400MHz ARM processor with Linux distribution to apply the multithread [5]. In 2013, S. H. Yoo and his team used the ChibiOS, a real-time operating system, on a STM32F4 microcontroller to create a multithreading control system for their quadrotor in the 2013 International Aerial Robotics Competition [6]. Utilization of multithreading in multi-rotors can also be found in [7] [8], etc.

In the commercial market, one of the most famous control board which can use multithread to support the system is PX4FMU/PIXHAWK. Using a Cortex-M4F microcontroller, which is supported by POSIX RTOS, this control board not only can run with the clock rate at 168MHz, extremely fast comparing with the older boards such as APM2.5 and 2.6, but also able to be multithreaded. AeroCore from Gumstix is another commercial control board supporting multithread. Unlike the PX4, AeroCore uses the Cortex-M4 running NuttX RTOS, provides user with a Linux platform, supports users with all of the facilities of a Linux system. In order to support these new generations of hardware, many group, like ArduPilot, has created the multithreading scheduler in their framework. The multithread of PIXHAWK in ArduPilot can be described as in Fig. 4.

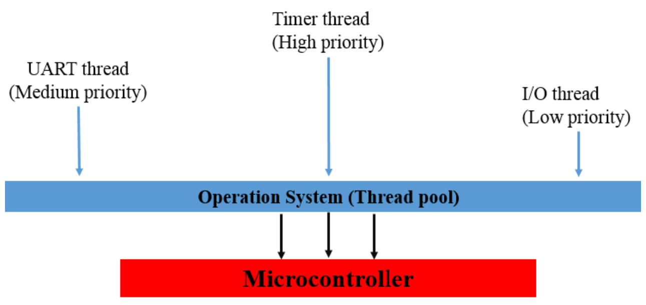

Figure 4. *PX4 multithreading in ArduPilot.*

2.3. FlyMaple Control Board and FreeRTOS

FlyMaple is also a new generation of control board, which is armed to applications in robotics and complicated system control. Using the ARM Cortex-M3, FlyMaple can operate with the clock rate at 72 MHz. This value is not as good as the PIXHAWK, however, it is also really amazing comparing to other APM control board. Although it has some limitations while comparing with the PIXHAWK, this control board is still acceptable since it has a more reasonable price, 70 USD[1] compares to nearly 200 USD of the others[2].

Since Cortex-M3 is the MCU of FlyMaple, there are two solutions to implement a multithreading firmware for this board. The first solution, which will be discussed in this section, is creating a new firmware for this board. In order to do that, multiple RTOSes will be mentioned and compared to find the most suitable one. Upgrading the existent, ready-to-used framework is the second solution and will be presented in the next part of this article.

Cortex-M3 architecture is supported by several RTOSes, including FreeRTOS, CooCox CoOS and ChibiOS. All of these operating systems are free and opened, and more importantly, supports multithread and multi-tasks. In his work, N. D. Bui summarized the compatibility of these RTOSes with FlyMaple [9]. Table 2 shows the results of his work.

Table 2. *RTOSes and FlyMaple.*

	ChibiOS	**CooCox CoOS**	**FreeRTOS**
Compatible with MapleIDE[3]	No	Yes*	Yes
Additional hardware required	Yes	No	No
Additional configurations / toolchain modifications required	Yes	Yes	No
Officially recommended	No	No	Yes
*Tested by individual			

From this result, it can be seen that FreeRTOS is the most suitable and convenient RTOS for FlyMaple. Therefore, it has been used to create the multitasking/multithreading firmware for this control board. In his project, N. D. Bui proposed a multitasking system based on the idea of Fig. 5. From this system, with some changes, a multitasking can changed into a multitasking/multithreading system.

Another project uses FreeRTOS for FlyMaple and successfully implemented the multithread is FlyMaple Project of OpenDrone[4]. This group used multithread for getting the values from the sensors and then calculating the PWM outputs.

One way or another, the projects above proved the compatibility of FreeRTOS and FlyMaple, as in the theory. However, both of them has one major limitation. In the FlyMaple project, the tasks for communication between the drone and the GCS has not been created. Even though multi-rotors can operate without the GCS in manual mode, the GCS is very important in case of auto mode. In his work, N. D. Bui created the communication tasks and then proposed a simple syntax for communicating with the self-developed GCS. His work is admirable, yet it is not standardized. Since the communicating protocol does not follow any standard, the portability and therefore the applicability is limited. However, this problem can be overcome with some simple modifications. By adding the libraries of MAVLink protocol and then construct a system for answering and responding based on this protocol, the new firmware will become compatible with any GCS supporting MAVLink such as Mission Planner.

1 http://www.dfrobot.com
2 https://store.3drobotics.com/products/3dr-pixhawk

3 MapleIDE: Maple Integrated development environment, developed in the Maple project of LeafLabs
4 https://github.com/opendrone/flymaple

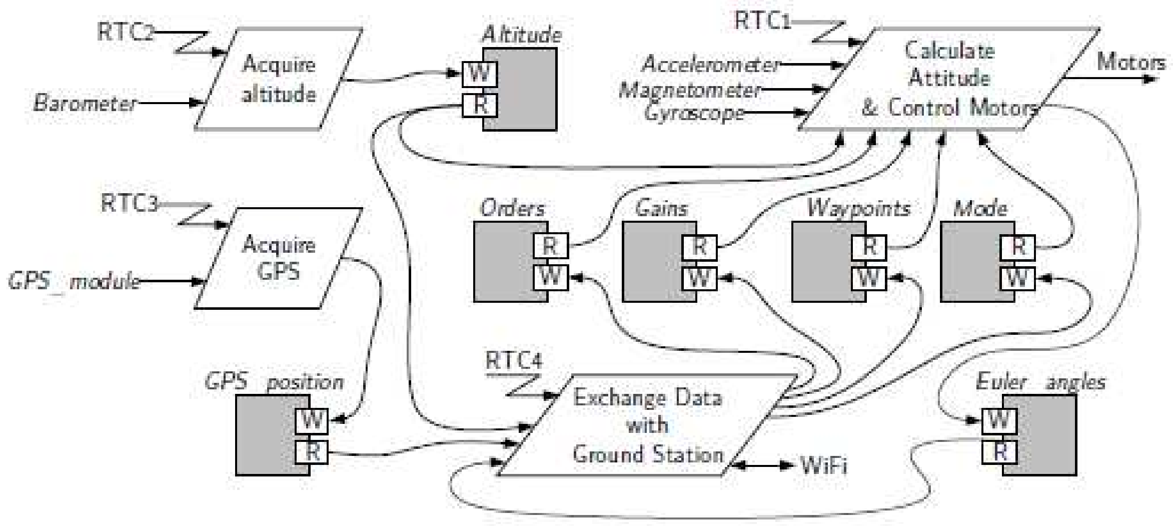

Figure 5. *Quadricopter multi-tasks in DARTS diagram.*

2.4. Multithread in ArduPilot for FlyMaple

FlyMaple is one of the boards supported by ArduPilot. However, because of the differences in architecture between this board and the wide-used board such as APM and PIXHAWK, ArduPilot does not fully support FlyMaple. One of the missing facility is multithread. In fact, the scheduler of ArduPilot for FlyMaple does not differ from that of APM older boards [10]. The only difference between FlyMaple and old version APM is that the main loop of FlyMaple can be run at 400Hz, meanwhile, this value for old boards is just 100Hz.

Even though there are many modifications have to be done to apply multithread for FlyMaple in this framework, there are still some benefits comparing to the first one. One of the most important benefits that ArduPilot is a ready-to-used framework, with all of the drivers for sensors, calculating process, communication protocol, etc. The second benefits lies on its supporting tools. ArduPilot offers developers with two simulations, Hardware In The Loop (HITL) and Software In The Loop (SITL). Depending on the requirements, users can choose the type of simulation and test the modified code before experimenting with the real system.

Fig. 6 suggests the architecture for the multithread for FlyMaple in ArduPilot. This structure is based on the multithreading structure of PIXHAWK and the proposed architecture of FlyMaple Project.

Since FlyMaple does not have the memory additional hardware like PIXHAWK, there are only two thread as shown above. The most important thread is timer, which is in charge of calculating the outputs from the inputs of sensors and the desired Euler angles. The second one, which has a lower priority, will be responsible for communication between control board, GPS system and GCS/RC. However, this is just the basic multithreading system, in case of other requirements, a more threads system can be created based on this principle. As ArduPilot is a multi-target and multi-application framework, the modification suggested here can be used in various type of multi-rotors. That is also one benefits of the second solution proposed in this article.

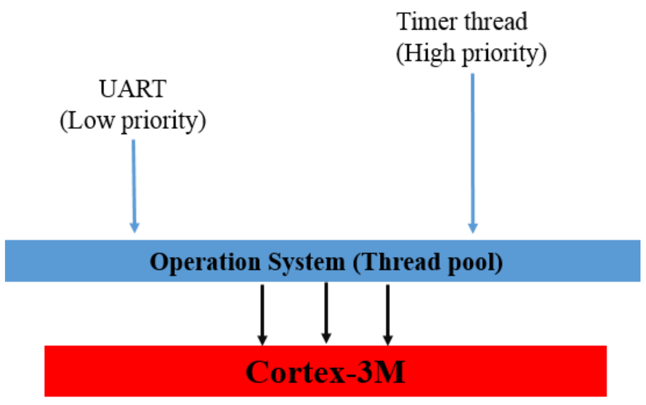

Figure 6. *Suggested Multithread for FlyMaple with ArduPilot.*

3. Conclusions

As discussed above, multithread is the rising trend of the control architecture for multi-rotors. Because of its reasonable price, FlyMaple is a promising control board to implement and experiment the multithreading control system. The two methods discussed in this article, each of them has their own pros and cons, however still can be done. The limits of our work at this moment is obviously the lacking of the successful implementation. That is one of the future work in this project.

Another future work is testing the behavior of the ready-to-used-multithreading-system like PIXHAWK, from this result and the result of FlyMaple, more comparisons can be done to increase our understanding with the multithread and its applications for multi-rotors.

Abbreviations

MCU	Microcontroller unit
RTOS	Real Time Operation System
I/O	Inputs/Outputs
UAV	Unmanned Aerial Vehicle
FIFO	First In First Out
OS	Operating System
PWM	Pulse Width Modulation
RC	Remote Controller

References

[1] Emery D. Berger, Ting Yang, Tongping Liu, Gene Novark, Grace: Safe Multithreading Programming for C/C++, Object - Oriented Programming, Systems, Languages, and Applications, Orlando, 2009.

[2] Intel, Intel Hyper-Threading Technology - Technical User's Guide, 2003.

[3] F. Ruini, Distributed Contrl for collective behaviour in micro-unmanned aerial vehicles, PhD Thesis, 2011.

[4] Patrick Fabiani et al, A multi-thread decisional architechture for real-time planning under uncertainty, The International Conference on Automated Planning and Scheduling (ICAPS 2007), 2007.

[5] G. Angeletti, J. R. Pereira Valente, L.Iocchi, D. Nardi, Autonomous Indoor Hovering with a Quadrotor, SIMPAR 2008, 2008.

[6] S. H. Yoo et al, Autonomous Aerial Robotics Team - 2013, International Aerial Roboics Competition, Oregon State University, 2013.

[7] F. D. Azevedo, Complete system for quadcopter control, Graduation Thesis Porto Alegre, 2014.

[8] O. Berthold, An Approach to UAV Controller Prototyping with Linux, Berlin, 2011.

[9] N. D. BÙI, Embedded System for Quadricopter, Internship report, Poitiers, France, 2014.

[10] A. D. Group, "Learning Ardupilot – Threading". 3Drobotics, <dev.ardupilot.com/wiki/learning-the-ardupilot-codebase/learning-ardupilot-threading/> (access Sep 2015).

Biography

Nguyen Anh Quang is a senior student from Department of Aerospace Engineering, Ho Chi Minh City University of Technology. As a student of P. F. I. E. V program, he has spent 5 months at Laboratory of Computer Science and Automatic Control for Systems (LIAS), ISAE-ENSMA, France for his graduation project.

Ngo Khanh Hieu (1978, HCM City) received Bachelor degree in Aeronautical Engineering (2001) – Ho Chi Minh City University of Technology, M. S. degree in Mechanics (2002) and PhD. degree in Computer Science (2008) from ENSMA, France. Work experience: Control-Command Systems, Flight Mechanics, R & D, Educator. Head of the Aerospace Engineering Lab., Ho Chi Minh City University of Technology.