Research and Implementation of PID Algorithm for Quadcopter Based on RT-Thread

Quan Huang, Lixin Li, Huisheng Zhang, Ang Gao, Wenzhong Yan
School of Electronics and Information,
Northwestern Polytechnical University, Xi'an 710129, China
1356466973@mail.nwpu.edu.cn

Abstract: In order to solve the problems in quadcopter system, such as real-time response, heavy workload and difficulty in control, this paper applies the embedded real-time operating system (RT-Thread) to the quadcopter. The PID algorithm has been considered in two structures in respect of the optional control signal applied to the quadcopter. For the better performance of quadcopter during flight the cascade control algorithm has been proposed. The practical test indicates that the quadcopter control system based on the embedded operating system can not only respond real-timely, but also can fly smoothly under the control of PID algorithm.

Keywords: PID; quadcopter; RTOS.

1. Introduction

The computation of quadcopter system is fairly complex, and most of the task is done in parallel. The key task such as attitude calculation, sensor data extraction, etc., must be executed rapidly to meet the real-time requirement, especially the quadcopter's PID control, but the others do not, such as temperature detection, voltage detection, etc. Ideally, each task is parallel, and does not interfere with each other to ensure real-time response. When one of the tasks gets into an infinite loop, the other tasks are not affected. The application of embedded real-time operation system to the quadcopter could meet the above requirements.

This paper mainly makes a research on the quadcopter's PID control and realizes it based on RT-Thread embedded operating system. The main processor of the quadcopter is STM32F407VGT6. And the RT-Thread OS runs on the processor which processes the quadcopter's data in parallel and real-timely.

2. System Development

2.1. Hardware Platform

The quadcopter's hardware is based on the STM32F407VGT6 processor which has a high-performance ARM Cortex-M4 core with a maximum system frequency of 168MHz, an FPU floating-point unit, 1Mbytes of Flash and 192Kbytes of SRAM. Additionally the processor has a wealth of peripherals, such as ADC, SPI, USART, CAN Bus, DMA and so on. Firstly, the higher operating frequencies and the high-speed memory provide powerful computational ability for the complex calculations of the quadcopter. Secondly, a lot of peripherals simplify the design of the qudacopter, reduce the amount of external IC and release the burden of CPU to some extent. Finally, the Cortex-M4 NVIC allows the quadcopter to have faster interrupt response and richer configuration capabilities, which rapidly completes the task switch.

The MPU-6050 is the world's first integrated 6-axis MotionTracking device that combines a 3-axis gyroscope,3-axis accelerometer, and a Digital Motion Processor. With its dedicated I^2C sensor bus, it directly accepts inputs from an external 3-axis compass to provide a complete 9-axis MotionFusion output. The MPU-6050 features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs and three 16-bit ADCs for digitizing the accelerometer outputs. For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ±250, ±500, ±1000 and ±2000 degree per second and a user-programmable accelerometer full-scale range of $\pm2g$, $\pm4g$, $\pm8g$,and $\pm16g$. It is widely utilized in quadcopter's attitude measurement. Figure 1 illustrates the hardware structure of the quadcopter system. It mainly includes inertial measurement unit (IMU), program download interface (ST-Link), USB interface, 4 PWM input interface, PPM input interface and USART peripheral interface.

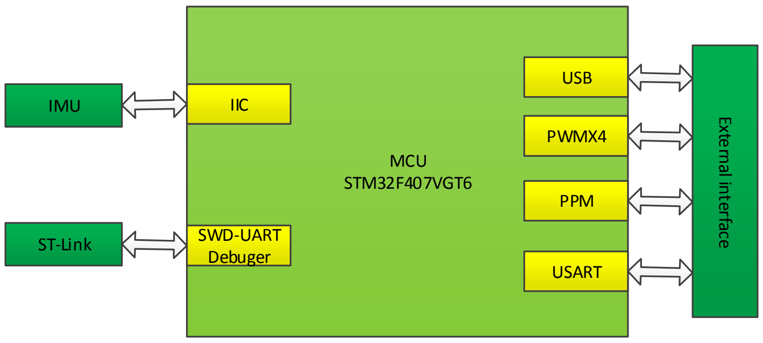


Figure 1. The quadcopter hardware structure

2.2 Real-Time OS

RT-Thread is an open source real-time operating system for embedded devices with a stable, efficient and scalable micro kernel. The core of the micro kernel includes: memory management, object manager, thread management, thread scheduling, thread communication, I/O device management and so on. RT-Thread is not only a real-time operating system, but also contains the following components: TCP / IP stack, file system, wireless network stack, device abstraction layer, storage Mechanisms, algorithms, graphics library, libc interface, POSIX interface. Except a tiny and complete kernel, it also includes the multi-tasking, semaphores, mutex lock, event collection, message queue, mailboxes and so on. And its kernel's context switching time: context switching caused by thread suspend: 2.608us; context switching caused by semaphore switching: 2.225us; mailbox-induced context switching: 4.466us; interrupt response delay: 0.450 us. It can be seen that the thread context switching time is very short. In multi-threaded applications, it can respond real-timely. RT-Thread is a complete, product level application system. This paper applies it to the quadcopter. Figure 2 is the RT-Thread system architecture.

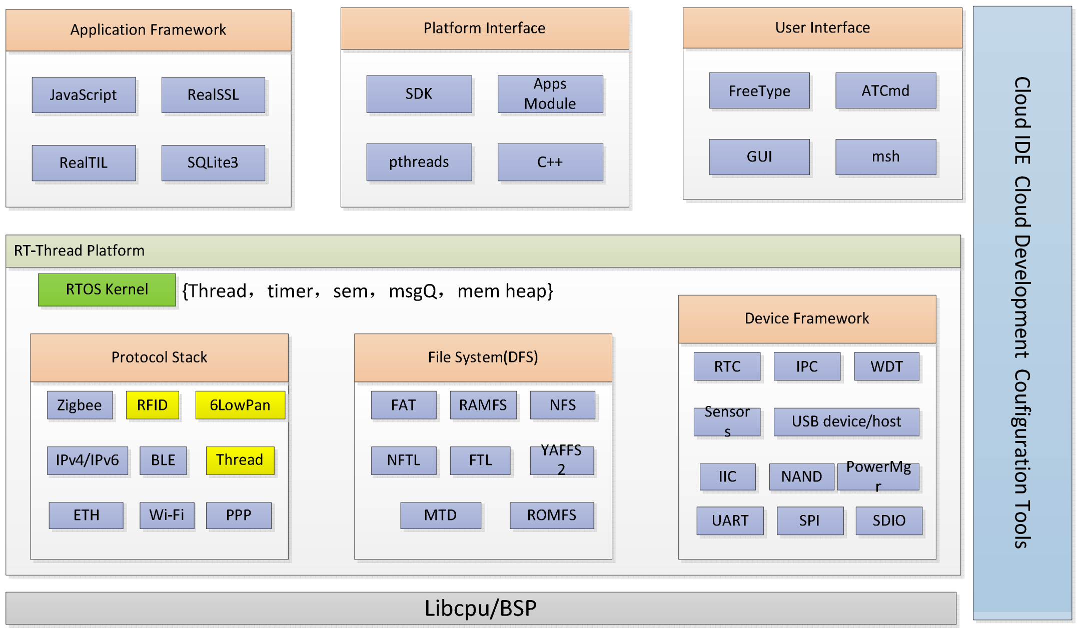


Figure 2. RT-Thread system architecture

3. PID algorithm for Quadcopter

PID control is a classical algorithm. The formula is described as follows:

$$u(t) = k_p * e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de(t)}{dt} \tag{1}$$

Where e is the difference between the target value and the current value, and t is the current time. Discrete digital structure is represented as follows:

$$u(k) = k_p * e(k) + k_i \sum_{j=0}^{k} e(j) + k_d \frac{e(k) - e(k-1)}{T} \tag{2}$$

Where $k = T, 2T, \cdots$, $e(k)$ denotes the error value at a one timestamp, and T denotes sensor sampling period. Figure 3 is the block diagram of the PID algorithm.

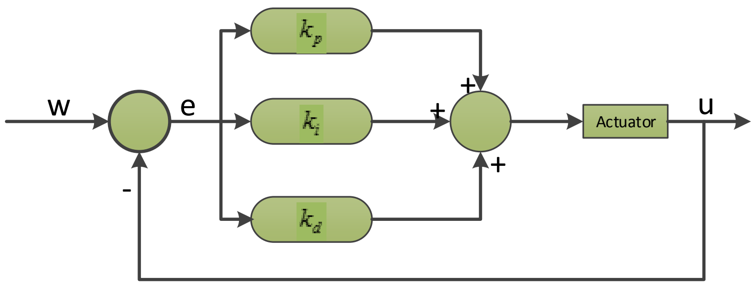


Figure 3. The PID controller principle

Applying (2) to quadcopter, its PID control structure is shown in Figure 4.

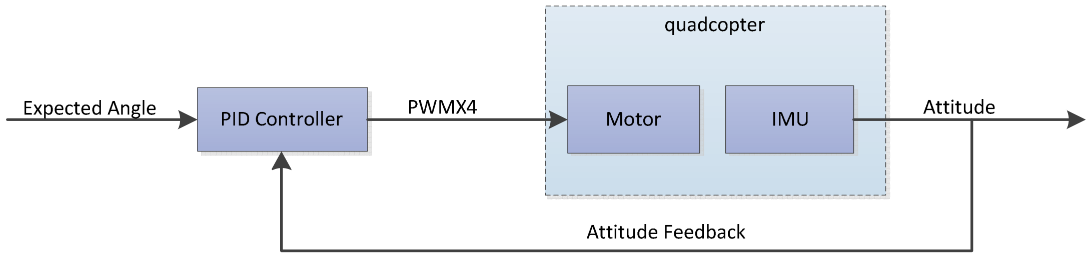


Figure 4. The quadcopter angle loop of PID control

Expected Angle is the angular value from the remote controller. The angle value is used as an input of the PID controller to fuse the data, then the outputs (*PWMX4*) are used as the inputs of the four motors. The IMU provides current attitude data of quadcopter for the PID controller.

We assume that the quadcopter read the data of gyroscope and accelerometer every 1ms and calculates the pitch and roll angles of the quadcopter, and the current attitude of the quadcopter is horizontal, i.e., pitch = 0 and roll = 0. Therefore, we can give the following PID calculation formula for the pitch angle of quadcopter.

$$PID_{pitch} = k_p A_{pitch} + k_i (A_{pitch} + A_{pitch\text{-}1} + \ldots + A_{pitch\text{-}n}) + k_d (A_{pitch} - A_{pitch\text{-}1}) \tag{3}$$

Generally, the attitude difference $(A_{pitch} - A_{pitch\text{-}1})$ can be obtained directly with the gyroscope's data multiplying 1ms. We can rewrite (3) as follows.

$$PID_{pitch} = k_p A_{pitch} + k_i (A_{pitch} + A_{pitch\text{-}1} + \ldots + A_{pitch\text{-}n}) + k_d \mathrm{Gyro}_{pitch} \tag{4}$$

For a normal flight of quadcopter, the integral portion is not necessary, because the integral will have bad impact on the quadcopter system if it is not properly used, and the stability of the quadcopter will decrease dramatically. Therefore, we can remove the integral in (4), we have

$$PID_{pitch} = k_p A_{pitch} + k_d \mathrm{Gyro}_{pitch} \tag{5}$$

From the above description, (3) is a positional PID controller which performs PID control by comparing the current position and the horizontal position (pitch = 0) as an error to realize the horizontal flight of the quadcopter. The difference of incremental PID and positional PID is that incremental PID uses the location of the change as control variable, namely,

$$\begin{aligned} &PID_{pitch} = k_p (A_{pitch} - A_{pitch\text{-}1}) + \\ &k_i ((A_{pitch} - A_{pitch\text{-}1}) + (A_{pitch-1} - A_{pitch\text{-}2}) \ldots + (A_{pitch-n=1} - A_{pitch\text{-}\mathrm{n}})) \\ &+ k_d ((A_{pitch} - A_{pitch\text{-}1}) - (A_{pitch-1} - A_{pitch\text{-}2})) \end{aligned} \tag{6}$$

We can see that the cumulative result of the integral is the current attitude subtracting the initial attitude. Assuming that the initial state is horizontal, the integral of the attitude increment is the current attitude. Proportional control is the increment of the current attitude relative to the previous attitude. The increment can be directly expressed in the gyroscope's data. Ignoring the differential control, we can have an expression as follows.

$$PID_{pitch} = k_p \mathrm{Gyro}_{pitch} + k_i A_{pitch} \tag{7}$$

Combining (5) and (7) together, we can form a cascade PID. Figure 5 illustrates a quadcopter's cascade PID block diagram.

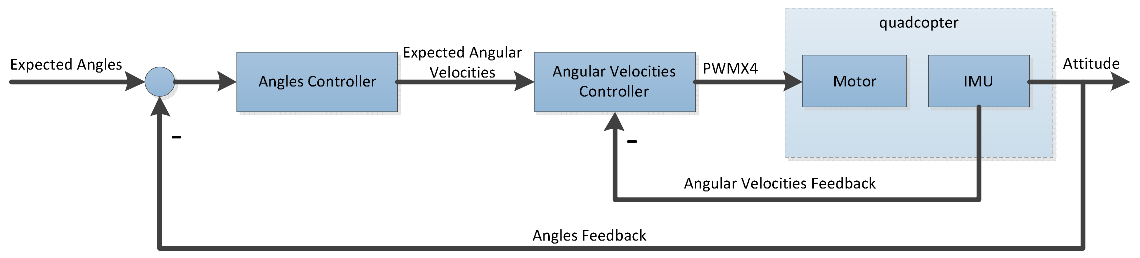

Expected Angles
Angles Controller
Expected Angular Velocities
Angular Velocities Controller
PWMX4
quadcopter
Motor
IMU
Attitude
-
-
Angular Velocities Feedback
Angles Feedback

Figure 5. Cascade PID controller for quadcopter

The inner loop is an angular velocities PID controller. It uses *Expected Angular Velocities* as its input, and outputs four channel PWM to the motor. The outer loop is an angle PID controller, it uses *Expected Angles* from RC as its input, and its output is used for the inner loop's input.

4. PID implementation for quadcopter

In RT-Thread real-time operating system, the task is implemented by using thread. The thread which describes the context and priority of a task is the basic scheduling unit in RT-Thread. RT-Thread's thread scheduler is a priority-based full preemptive scheduling, except the interrupt handling function in the system. The code locked by the thread scheduler and the interruption in RT-Thread system are not preemptible. The rest of the system can be preempted, including the thread scheduler itself. RT-Thread supports a total number of 256 priority (0 ~ 255, the lower the value the higher the priority, 0 is highest priority, 255 is allocated to the idle thread which in general is not used). In this paper, the PID control of the quadcopter is used as a thread in the RT-Thread system. The other threads that mainly involved are the attitude information acquiring thread and the attitude fusing thread.

Thread in RT-Thread generally consists of three parts: the thread entry function, the thread control block, and the thread stack. Thread entry function generally contains an infinite loop, so that the thread will not reach the end of the thread entry function, or it will be deleted by idle thread; thread block records the various properties of a thread, which is convenient for the system to use the thread block link-table to manage the thread. The thread stack is a contiguous block of memory that is used to hold the contents of the CPU registers while the thread is switching and responding to interrupts. Each thread must have its own stack.

In RT-Thread, there are two kinds of thread: dynamic thread and static thread. In a quadcopter system, the PID thread in this paper is static. Figure 6 illustrates the flow chart of creating a static thread. At first, define the thread stack, and specify the size of the stack; followed by the definition of the thread block that is used to preserve the various attributes and schedule management; and finally call the RT-Thread system API function to create thread, and start the thread by calling RT-Thread API function.

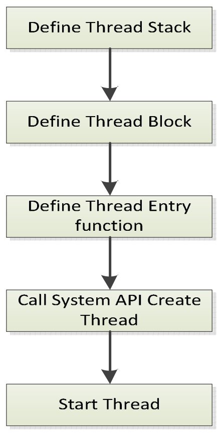
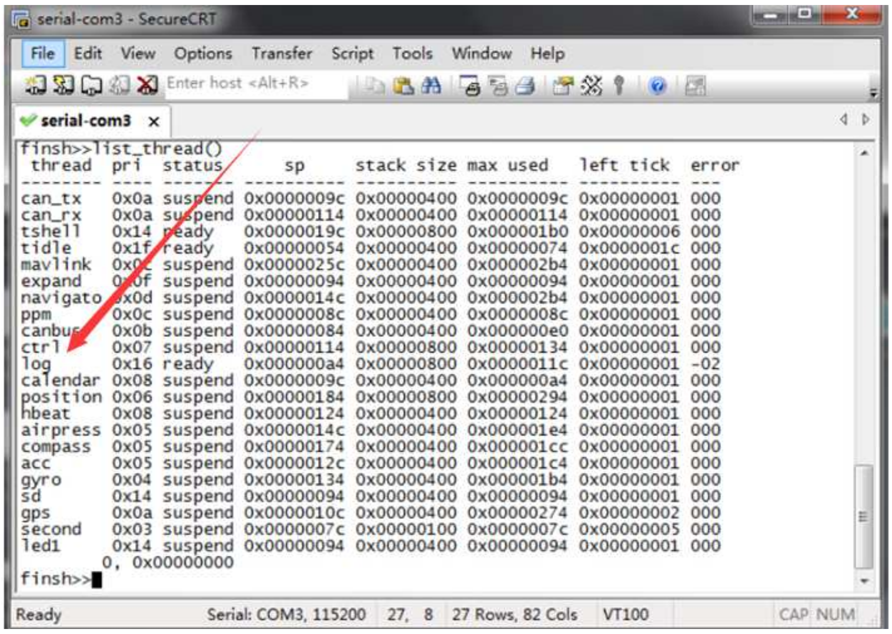


Figure 6. Create a static thread

In the quadcopter system, since the attitude of the quadcopter needs to be acquired and updated in real time, the thread priority of the attitude information is set to 4 and the time slice is set to a clock tick; the priority of the attitude fusion thread is set to 5, and the time slice is a clock tick; the PID control thread priority is set to 7, the time slice is 2 clock ticks. The frequency of acquiring the attitude information is 1 kHz.

5. Test and Flight of Quadcopter

All self-defined threads running in RT-Thread for quadcopter are shown in Figure 7. In the first column named thread, the thread called *ctrl* is the PID control thread, and its priority is 7. Because the quadcopter at this time is connected to PC for debugging, the PID control thread is suspended.



```
finsh>>list_thread()
 thread   pri  status      sp     stack size max used   left tick  error
-------- ---- ------- ---------- ---------- ---------- ---------- ---
can_tx   0x0a suspend 0x0000009c 0x00000400 0x0000009c 0x00000001 000
can_rx   0x0a suspend 0x00000114 0x00000400 0x00000114 0x00000001 000
tshell   0x14 ready   0x0000019c 0x00000800 0x000001b0 0x00000006 000
tidle    0x1f ready   0x00000054 0x00000400 0x00000074 0x0000001c 000
mavlink  0x0c suspend 0x0000025c 0x00000400 0x000002b4 0x00000001 000
expand   0x0f suspend 0x00000094 0x00000400 0x00000094 0x00000001 000
navigato 0x0d suspend 0x0000014c 0x00000400 0x000002b4 0x00000001 000
ppm      0x0c suspend 0x0000008c 0x00000400 0x0000008c 0x00000001 000
canbus   0x0b suspend 0x00000084 0x00000400 0x000000e0 0x00000001 000
ctrl     0x07 suspend 0x00000114 0x00000800 0x00000134 0x00000001 000
log      0x16 ready   0x000000a4 0x00000800 0x0000011c 0x00000001 -02
calendar 0x08 suspend 0x0000009c 0x00000400 0x000000a4 0x00000001 000
position 0x06 suspend 0x00000184 0x00000800 0x00000294 0x00000001 000
hbeat    0x08 suspend 0x00000124 0x00000400 0x00000124 0x00000001 000
airpress 0x05 suspend 0x0000014c 0x00000400 0x000001e4 0x00000001 000
compass  0x05 suspend 0x00000174 0x00000400 0x000001cc 0x00000001 000
acc      0x05 suspend 0x0000012c 0x00000400 0x000001c4 0x00000001 000
gyro     0x04 suspend 0x00000134 0x00000400 0x000001b4 0x00000001 000
sd       0x14 suspend 0x00000094 0x00000400 0x00000094 0x00000001 000
gps      0x0a suspend 0x0000010c 0x00000400 0x00000274 0x00000002 000
second   0x03 suspend 0x0000007c 0x00000100 0x0000007c 0x00000005 000
led1     0x14 suspend 0x00000094 0x00000400 0x00000094 0x00000001 000
        0, 0x00000000
finsh>>
```

Figure 7. All user self-defined thread running in RT-Thread for quadcopter

Figure 8. The actual flight of quadcopter

Figure 8 is the practical flight test of the quadcopter. We can see the quadcopter can fly steadily.

6. Conclusions

This paper describes the PID algorithm for quadcopter based on RT-Thread embedded operation system. We provide an intuitive derivation of PID algorithm for quadcopter and apply the algorithm using thread mechanism in RT-Thread system to quadcopter. The quadcopter can not only respond real-timely, but also can fly smoothly under the control of PID algorithm.

Acknowledgement

This work is supported by Shaanxi Province natural science basic research plan surface project (No. 2016JM6062), by China Aerospace Science and Technology Corporation Aerospace Science and Technology Innovation Fund funded project, by the Seed Foundation of Innovation and Creation for Graduate Students in Northwestern Polytechnical University.

References

[1] Xi Li, Yang Chen, Pengzhen Chen. Fixed Point Tracking Control of Quadcopter Based on PID Algorithm. Computer Measurement & Control, 2016:109-112.
[2] Xuebing Wang. Research on Four-rotor Attitude Control System Based on PID Algorithm. Xi’an University of Science and Technology, 2015.
[3] Qi Li, Mei Li. Design of Industrial Remote Controller Based on RT-Thread. Research and Exploration in Laboratory, 2013:61-64.
[4] Huanye Liu. Research and Design of Flight Control System for Small Quadcopter. Shanghai Jiao Tong University, 2009.
[5] Zhuan Tu. Design and Implementation of Embedded CAN-Ethernet Gateway Based on RT-Thread. Shanghai Jiao Tong University, 2014.
[6] Beard Xiong. RT-Thread Programming Guide. http://www.rt-thread.org/download/manual/rtthread_manual.zh.pdf, 2016.
[7] Jie Zhi. Study RT-Thread. http://www.rt-thread.org/download/manual/study_rtt.pdf, 2013.
[8] Zhenyun Wang. Design and Realization of Two Wheel Self-balancing Robot Based on RT-Thread and STM32. North University of China, 2016.