RT-Linux based Hard Real-Time Software Architecture for Unmanned Autonomous Helicopters

Won Eui Hong,Jae Shin Lee, Laxmisha Rai and Soon Ju Kang
School of EECS, Kyungpook National University (KNU), Korea
wonny@dreamwiz.com,faithlee3@hanmail.net,{laxmisha, sjkang}@ee.knu.ac.kr

Abstract

The UAV (Unmanned Aerial Vehicle) performs various kinds of missions while performing autonomous flight navigation. In order to realize all functionalities of the UAV, the software part becomes a very complex hard real-time system because some hard real-time tasks, soft real-time tasks and non-real-time tasks are concurrently working together under a RTOS. In this paper, a hierarchical architecture for Unmanned Autonomous Helicopter System is proposed that guarantees the real-time performance of hard real-time tasks and the re-configurability of soft real-time or non-real-time tasks under the RT-Linux. This software architecture has four layers: hardware, execution, service agent and remote user interface layer according to the reactiveness level for external events. In addition, the layered separation of concurrent tasks makes different kinds of mission reconfiguration possible in the system. An Unmanned autonomous helicopter system was implemented using Kyosho RC Helicopter to test and evaluate the performance of the proposed systems.

1. Introduction

The auto-piloted helicopter as a typical UAV system is self-operated aircraft, used to collect the environmental information of the surroundings, and it decides the tasks based on its analyzed decisions. To realize the autonomous behavior, the auto-piloted helicopter system consists of complex hardware components and software components and required highly time constrained coordination between these components. Especially, the software part as the abstracted components of hardware consists of three grouped concurrent tasks (hard real-time tasks, soft real-time tasks and non-real-time tasks) and these are always run concurrently. In order to prevent any accidental situation, the deadline of the hard real-time tasks should be guaranteed explicitly and to support the run time flexibility between concurrent tasks (e.g., change flying mode from hovering to go forward), some dynamic reconfiguration for concurrent tasks are also inevitably required. Due to the important applications and technical difficulty, many researches have been tried to solve the auto-pilot helicopter problems[1,2,3,4,5] but majority of these research are considering the auto-pilot issue only and they didn't suggest more flexible software architecture for supporting dynamic reconfiguration of concurrent tasks without loss of any hard deadline of real-time tasks.We focused our research on a hierarchical concurrent task structuring issue without the loss of missing deadline problems. Therefore, this paper proposes a layered software architecture for guarantee the hard real-time constraints and for supporting dynamic reconfiguration simultaneously.

2. Related Works

The development of autonomous helicopters for real-world applications is a challenging area of research in recent years. The Carnegie Mellon Robotics Institute, developed CMU autonomous helicopter, to build high-accuracy topological maps for NASA geologists[1].The Georgia Institute of Technology (GIT), designed and developed an UAV avionics system that provides navigational and terrain information[2]. An embedded system for the flight control of an autonomous helicopter has been developed at the Institute for Computer Systems (ICS) at ETH Zurich[3]. The University of California at Berkeley developed Giotto-based control software by implementing the autopilot system of an autonomous model helicopter. The high performance, in hard real-time embedded systems is achieved using the Giotto approach [4].

3. Design Considerations and Conceptual Design

The main task of autonomous flight system is to maintain deadline to avoid any critical conditions. The system must analyze the sensor data and controls the servomotors based on the analysis. The task deadlines

Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)
1533-2306/05 $20.00 © 2005 **IEEE**

must be met within the timeframe to avoid the out-of control situations of the aircraft. The objective of the autonomous flight is not just limited to flight and its control. It has many other various missions like, obtaining pictures or movies using camera, transporting equipments from station to station, gathering weather details etc. These tasks are non-periodic as well as non-predictable. Execution of these tasks and scheduling of these tasks are also our design considerations. In case of scheduling these tasks, the real-time tasks have to be given higher priority and other tasks with lower priority.

3.1 Proposed Software Architecture

The overview of the proposed software architecture with guaranteed real-time task execution features is as shown in fig 1. The proposed software architecture is a layered architecture with four layers as hardware layer, execution layer, service agent layer (SA) and remote user interface (RUI) layer. The hardware layer composed of sensors, actuators and communication modules and the actual physical module of the autonomous helicopter. Guaranteed real time task execution of the autonomous flight is part of the execution layer.

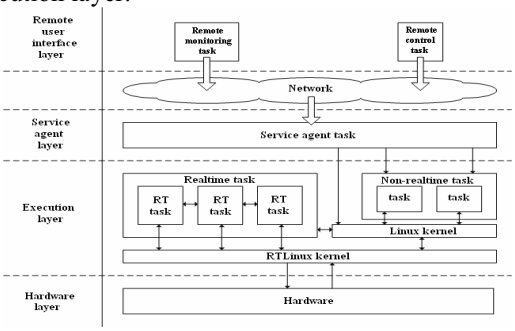

Fig. 1. Overview of proposed software architecture

The execution layer executes the autonomous flight task, which has the highest priority and also the task which controls the flight operation. The non-real time tasks in the execution layer are executed after the completion of real time tasks. The RUI layer has two functions remote monitoring (RM) and remote control (RC) task and communicates with SA layer using network. The SA task receives the required information from the RM task through the network and decides the state of the helicopter. Using this architecture, we can execute tasks parallel using multiple processors in each layer. Fig .2 shows the hardware system architecture of the proposed UAV helicopter.

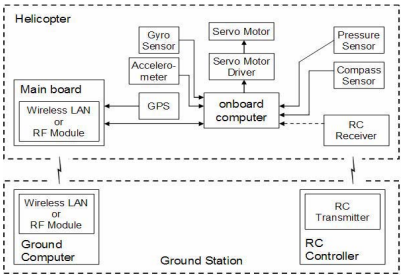

Fig 2. Hardware system of the proposed UAV helicopter.

4 Detailed Designs: Software Architecture Overview

The operator first performs the take-off operation, hovering and initial adjustments of the flight using remote controller, manually. Later, the control is switched to the automatic-mode and the helicopter operates automatically and the ground station monitors the state of the helicopter. The monitoring task is executed from the ground station and it provides the information and status about the execution of autonomous flight task of the helicopter.

4.1. Behavior Analysis

The event sequence diagram is as shown figure 3.

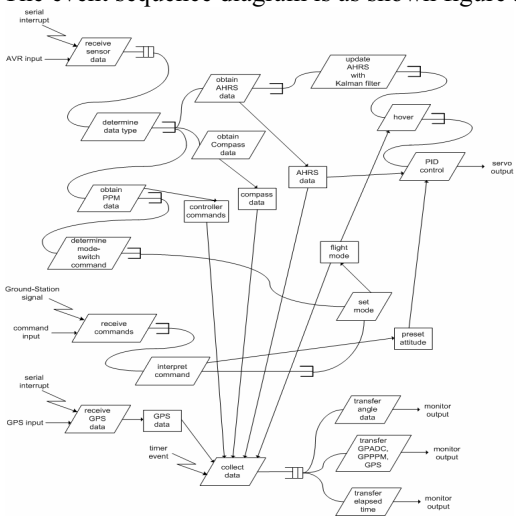

Fig 3. Event Sequence diagram

The system gets the quantized IMU (Inertial Measuring Unit) sensor's data, PPM(Pulse Position Modulation) signal compass sensor data from the remote controller using serial port. The program calculates the flight altitude using Kalman filter. Based on the information collected from IMU sensor and compass sensor, the angular velocity, slope information and attitude is calculated in a single task and it is executed periodically. This information is sent to the ground

Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)
1533-2306/05 $20.00 © 2005 **IEEE**

station for monitoring the flight. The GPS data is also collected and sent to the ground station periodically. The sequence of operations is shown in the figure. 4.

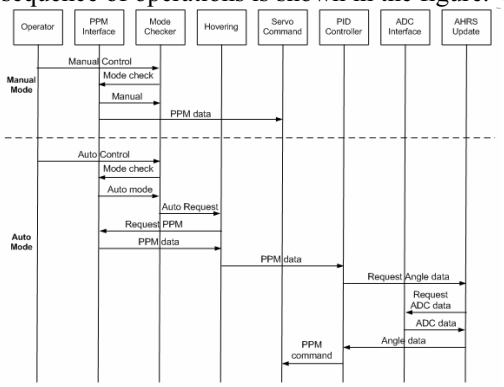

Fig 4. Sequence diagram

4.2. Scheduling analysis :Onboard and main board

According to the proposed hardware configuration as shown in the fig. 2 the scheduling the tasks are based on the tasks running on two processor boards, main board and on-board. On board tasks executed sequentially using round-robin scheduling. First, the onboard receives sensor data from the ADC converter, PPM signal from the remote controller and required data from the main board, later it sends the output to the servo motors. These tasks executed sequentially and repeatedly using fixed time periods. On the other hand, the main board processor executes tasks in multiple priority levels. These tasks are divided as higher priority, intermediate priority and lower priority tasks. The higher priority tasks include tasks to receive the sensor data from IMU, updating AHRS (Attitude and Heading Reference System) information and obtaining required information from onboard. Based on these data, the task computes the required output data using PID controller. Since, these tasks have highest priority, as they are directly related to flight operation, are not interrupted by any other tasks.

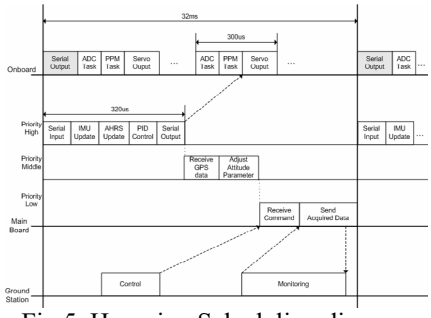

Fig 5. Hovering Scheduling diagram

These tasks have fixed running time and only after the execution of these tasks other lower priority asks will run. The intermediate tasks include task to adjust attitude parameter and task to receive GPS data. The fig.5 shows hovering scheduling diagram.

5. Implementation and Performance Evaluation

5.1 Implementation

In our research team, we selected an auto-pilot helicopter as a typical UAV system and named it as KNU Auto-pilot Helicopter. As shown in Fig. 6, the ERGO 50 model of JR PROPO and Concept 60 model of Kyosho type of RC Helicopters are used in this research. It has enough payload capacity to carry, main board, onboard, various sensor and wireless LAN equipments.

Fig 6. KNU RC Helicopter

The system is implemented in RTLinux 3.1. The related hardware includes Kyosho RC Helicopter, Transmeta Crusoe compatible x86 processor (Main board), ATM Mega 163 chip (Onboard). The communication protocols used are TCP/IP and Wireless LAN

5.2 Comparative Performance Evaluation between Linux and RT-Linux

The autonomous flight tasks are implemented as RT-Linux tasks for the successful operation of the test flight. In order to measure the performance of the proposed software architecture, relatively we tested the auto-pilot software in both a standard Linux and RT-Linux. In case of standard Linux version; autonomous flight tasks follow round-robin scheduling. The internal tasks like attitude control task, communication module of ground station executes consecutively with same priority. The delay introduced between the tasks affects the efficiency of the whole system.

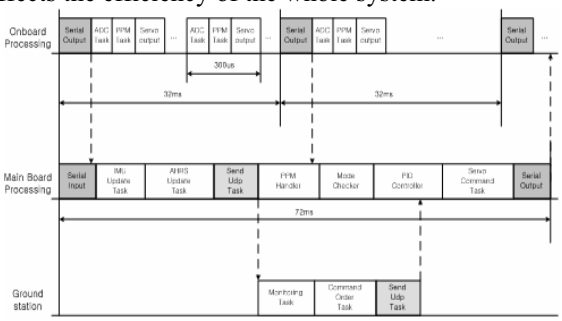

Fig. 7. Scheduling diagram of Linux based program

The fig. 7 shows the scheduling diagram of the autonomous flight program in Linux. The timing evaluation of the autonomous flight tasks include the evaluation of time taken by the sensor for input data, calculation time of the control unit, servo-motor output time. The comparative evaluation of execution time of non-real time flight task in Linux and real-time execution in RT-Linux of the autonomous flight task is shown in Table 1.

	Linux	RT-Linux
Average Value	32.242 ms	258 μs
Standard Deviation	1.669 ms	24.535 μs
Maximum Value	39.653ms	376 μs
Minimum Value	16.672ms	246 μs

Table 1. Task execution times in Linux and RT-Linux

The fig. 8 shows flight execution time in Linux and RT-Linux. In the case of RT-Linux, we can see that the execution time is range from 200μs to 400μs and it has a small deviation time of 200 μs.

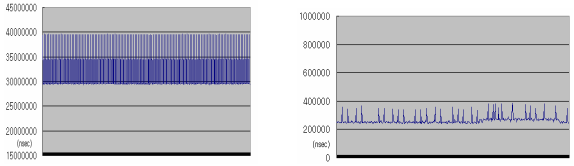

Fig 8. Flight execution time graph in(a) Linux and (b) RT-Linux

The execution time of Linux tasks are much longer than that of the RT-Linux tasks, because the TCP/IP communication of the ground station is an additional task executed, while the autonomous flight is operating. The data rate which is transmitted to the ground station is fixed. We have compared the execution time of RT-Linux task with Linux task with same resolution. It has a big difference as shown in figure 9.

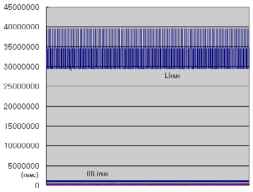

Fig 9. Comparison of execution time

6. Conclusion

In this paper, hierarchical architecture for Unmanned Aerial Vehicle is proposed that guarantees the real-time performance of autonomous flight and the re-configurability of missions of UAV. The proposed system architecture is divided into hardware layer, execution layer, service agent layer and remote user interface layer. With this architecture; we can apply this to the actual unmanned helicopter and can confirm its exact operation, based on performance tests and results. Also, this architecture can be applied to various unmanned control systems. With present results, we can apply better control algorithm for autonomous flight operation, to fly in various environments.

6. Acknowledgements

This work was supported by grant No. R01-2003-000-10252-0 from the Basic Research Program of Korea Science and Engineering Foundation

7. References

[1] Ryan Miller , Omead Amidi, and Mark Delouis, Arctic Test Flights of the CMU Autonomous Helicopter, Carnegie Mellon Robotics Institute, http://www.ri.cmu.edu/project/chopper

[2] Joerg S Dittrich and Eric N Johnson. A Multi-Sensor Navigation System for an Autonomous Helicopter, AIAA Digital Avionics Conference and No. 377, Irvine, CA and Oct. 2002.

[3] Markus Kottmann Software for Model Helicopter Flight Control and Technical Report 316, Institute of Computer Systems and ETH Z, 1999.

[4] Christoph M. Kirsch, Marco A.A. Sanvido, Thomas A. Henzinger, A Giotto-based Autonomous Flying Helicopter System, University of California at Berkeley, Berkeley, USA ,www-cad.eecs.berkeley.edu/~cm/submitted/giotto-heli.pdf

[5] Myungsoo Jun and Stergios I Roumeliotis and Gaurav S Sukhatme, State Estimation of an Autonomous Helicopter Using Kalman Filtering. Proc. 1999 IEEE/RSJ International Conference on Robots and Systems (IROS 99).

[6] Eric N Johnson and Paul A Debitetto, Modeling and Simulation for Small Autonomous Helicopter Development, American Institute of Aeronautics and Astronautics, 1997.

[7] M Morin, S Nadjm-Tehrani, P Osterling, and E Sandewall, Real- Time Hierarchical Control, IEEE Software and Vol. 9, pp51 -57, Sept. 1992.

[8] Hanssan Gomaa and Designing Concurrent Distributed and Real-Time Applications with UML and Addison Wesley, 2000.