Design and Implementation of an Operational Flight Program for an Unmanned Helicopter FCC Based on the TMO Scheme

Se-Gi Kim[1], Seung-Hwa Song[2], Chun-Hyon Chang[2],
Doo-Hyun Kim[2], Shin Heu[3] and JungGuk Kim[1]

[1] Hankuk University of Foreign Studies
{undeadrage, jgkim}@ hufs.ac.kr
[2] Konkuk University
sshtel@gmail.com, {chchang, doohyun}@konkuk.ac.kr,
[3] Hanyang University
shinhue@cse.hanyang.ac.kr

Abstract. HELISCOPE is the name of a project support by MKE (Ministry of Knowledge & Economy) of Korea to develop flying-camera services that transmits the scene of a fire by an unmanned helicopter. In this paper, we introduce the design and implementation of the OFP (Operational Flight Program) for the unmanned helicopter's navigation based on the well-known TMO scheme. Navigation of the unmanned helicopter is done by the commands on flight mode from our GCS (Ground Control System). As the RTOS on the FCC (Flight Control Computer), RT-eCos3.0 that has been developed based on the eCos3.0 to support the basic task model of the TMO scheme is being used. To verify this navigation system, a HILS (Hardware-in-the-loop Simulation) system using the FlightGear simulator also has been developed. The structure and functions of the RT-eCos3.0 and the HILS is also introduced briefly.

Keywords: unmanned helicopter, on-flight software, TMO

1 Introduction

The HELISCOPE [1] project is to develop an unmanned helicopter and its on-flight embedded computing system for navigation and real-time transmission of the motion video of the scene of a fire using the HSDPA or Wibro communication scheme. The unmanned helicopter shall be used especially in the phase of disaster response and recovery intensively.

In this paper, we introduce the design and implementation of the OFP (Operational Flight Program) subpart of the HELISCOPE project based on the well-known TMO scheme [3]. Navigation of the unmanned helicopter is done by the commands on flight mode from our GCS (Ground Control System). As the RTOS on the FCC (Flight Control Computer), RT-eCos3.0 [2], [5] that has been developed based on the eCos3.0 to support the basic task model of the TMO scheme is being used.

The reason why we used the TMO model and RT-eCso3.0 is because an OFP must provide a well-structured real-time control mechanism with various sensors, actuators and communication devices connected to the FCC. The OFP must process real-time sensor inputs and commands coming from GPS/INS (GPS/Inertial Navigation System), AHRS (Attitude and Heading Reference System) and the GCS in a deadline based manner. A main FC (Flight Control) task of the OFP calculates real-time control signals with the inputs and must send them to actuators of the helicopter in the pre-given deadline. Also, the FCC must report the status of the aircraft to the GCS periodically or upon requests from the GCS.

Our OFP consists of one time-triggered task and several message-triggered tasks. Collecting of sensor data and commands is done by several message-triggered tasks. Upon receiving data, these tasks store them in to the ODS (Object Data Store) of the OFP-TMO. The ODS also contains some parameters on the capability of the aircraft. Periodic calculation and sending of control outputs with the data in ODS is performed by the main time-triggered task. All these tasks are scheduled by the RT-eCos3.0 based on their timing constraints.

To verify the navigation system, a HILS (Hardware-in-the-loop Simulation) system using the open-source FlightGear simulator also has been developed.

In section 2, the TMO model and the RT-eCos3.0 kernel is introduced briefly as related works and in section 3, design and implementation of the OFP are described. In section 4, the HILS system for verification will be discussed and in section 5, we will conclude.

2 The TMO model and the RT-eCos3.0 kernel

In this section, a distributed real-time object model, TMO, and the RTOS that has been used in implementing the OFP are introduced briefly as related works.

2.1 TMO model [2], [3]

The TMO model is a real-time distributed object model for timeliness-guaranteed computing at design time. A TMO instance consists of three types of object member: an ODS (Object Data Store), time-triggered methods (SpM: Spontaneous Method) and message-triggered methods (SvM: Service Method). An SpM is actually a member-thread that is activated by a pre-given timing constraint and must finish its periodic executions within the given deadline. An SvM is also a member-thread that is activated by an event-message from a source outside a TMO. Main differences between the TMO model and conventional objects can be summarized as follows.

- TMO is a distributed computing component and thus TMOs distributed over multiple nodes may interact via distributed IPC.
- The two types of method are active member threads in the TMO model (SpMs and SvMs).

- SvMs cannot disturb the executions of SpMs. This rule is called the BCC (Basic Concurrency Constraint). Basically, activation of an SvM is allowed only when potentially conflicting SpM executions are not in place.

2.2 RT-eCos3.0 scheduling and IPC [2], [5]

RT-eCos3.0 which is a real-time extension of the eCos3.0 supports multiple per-thread real-time scheduling policies. The policies are the EDF(Earliest Deadline First)/BCC, LLF(Least Laxity First)/BCC and the non-preemptive FTFS (First Triggered First Scheduled) scheduler.

The non-preemptive FTFS scheduler is used when an off-line scheduling scenario is given by our task serializer [7] that determines the initial offsets of time-triggered tasks so as that all task instances can be executed without overlap and preemption. On the other hand, the EDF/BCC and the LLF/BCC are normally used when there is no pre-analysis tool or when the task serialization is not possible. This means that the EDF and LLF schedulers are ones for the systems with dynamic execution behaviors.

With these real-time schedulers, two basic types of real-time task; time-triggered and message-triggered tasks; are supported by the kernel. SpMs and SvMs of the TMO model are mapped to these tasks by the TMOSL (TMO Support Library) for TMO programmers.

The timing precision that is used for representing timing constraints such as start/stop time, period and deadline has been enhanced into micro second unit. Although the scheduling is performed every milli-second, the kernel computes current time in micro-second unit by checking and compensating the raw PIC clock ticks from the last clock interrupt. With these this schedulers, the kernel shows task-switch overhead of 1.51 micro-sec and scheduling overhead of 2.73 micro-sec in a 206MHz ARM9 processor environment.

Management of message-triggered tasks is always done in conjunction with the logical multicast distributed IPC of the RT-eCos3.0. Once a message is arrived via the network transparent IPC at a channel associated with a message-triggered task, the task is activated and scheduled to finish its service within the pre-given deadline.

The IPC subsystem consists of two layers. The lower layer is the intra-node channel IPC layer and the upper is the network transparent distributed IPC layer. This layering is for flexible configurations of the RT-eCos3.0 IPC based on various protocols. Besides supporting this basic channel IPC, the TMOSL has been enhanced to support the Gate and RMMC (Real-time Multicast Memory replication Channel) that is a highly abstracted distributed IPC model of the TMOSM of U.C. Irvine [4]. Since the role of an SvM is to handle external asynchronous events, the channel IPC has been extended so that external devices such as an IO devices or a sensor-alarm device can be associated to a channel. In this case, a message-triggered task can be activated when an asynchronous input occurs and is scheduled to be finished within the predefined deadline.

3 Design and implementation of OFP for unmanned helicopter FCC based on the TMO scheme

In this section, control points and OFP of a helicopter is mainly described.

3.1 Helicopter mechanics

Being different from the fixed wing aircraft, a helicopter makes a stable flight by using the thrust and upward force generated by the fixed speed rotation of the engine and the angles of the main and tail rotor blades. For change of heading, vertical flight and forward movement, it uses the tilt rotor disk and tail rotor.

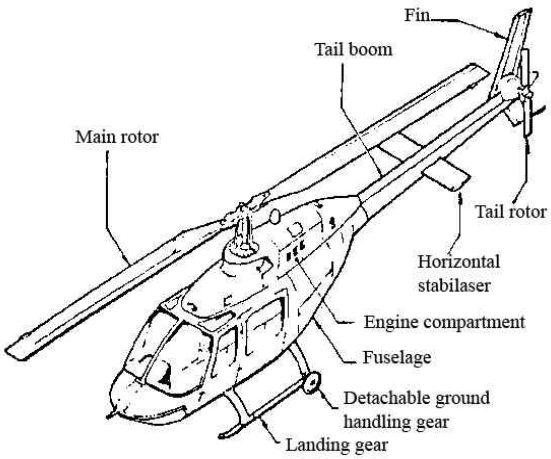

Fig 1. Main components of a helicopter [1]

A helicopter can maintain the powerful hovering state that is not possible in fixed wing aircraft but it has a difficult problem in maintain a stable attitude caused by its complicated lifting mechanism. Fig. 1 shows the main components of a helicopter and table 1 shows the major control points and their motion effects.

3.2 Flight modes of the unmanned helicopter

The unmanned helicopter receives commands from the GCS and the OFP on FCC calculates the values of control signals to be sent to control points with current sensor values from GPS/INS, AHRS and SWM (Helicopter Servo Actuator Switching Module). Theses control signals are sent to control points via the SWM. The OFP also sends information on location and attitude to the GCS for periodical monitoring. In case of losing control, the SWM is set to a manual remote-control mode. Fig. 2

describes the control structure. Actually, Fig. 2 describes the whole structure of the HELISCOPE project including the MCC (Multimedia Communication Computer) board and its communication mechanism however, the description of this part has been excluded in this paper.

Auto-flight modes of our unmanned helicopter are as follows (Fig. 3).

- Hovering
- Auto-landing
- Point navigation
- Multi-point navigation

Table 1. Helicopter controls and effects [1]

Control Points	Directly controls	Primary effect	Secondary effect	Used in forward flight	Used in hover flight
Cyclic lateral	Varies main rotor blade pitch left/right	Tilts main rotor disk left and right through the swashplate	Induces roll in direction moved	To turn the aircraft	To move sideways
Cyclic longitudinal	Varies main rotor blade pitch fore/aft	Tilts main rotor disk forward and back via the swashplate	Induces pitch nose down or up	Control attitude	To move forwards/ backwards
Collective	Collective angle of attack for the rotor main blades via the swashplate	Inc./dec. pitch angle of rotor blades causing the aircraft to rise/descend	Inc./dec. torque and engine RPM	To adjust power through rotor blade pitch setting	To adjust skid height/ vertical speed
Tail rotor: Rudder	Collective pitch supplied to tail rotor blades	Yaw rate	Inc./dec. torque and engine RPM (less than collective)	Adjust sideslip angle	Control yaw rate/ heading

In hovering mode, the helicopter tries to maintain current position and attitude even if there is a wind. And hovering is always the end of all flight modes except for auto-landing and is the most difficult mode when auto-flight mode is used. In auto-landing mode, landing velocity is controlled according to the current altitude.

In point-navigation mode, the helicopter moves to a target position given by the GCS. In this mode, a compensation algorithm is used when there is a breakaway from the original track to the target caused by a wind. When the craft arrives at the target, it turns its mode to hovering. Multi-point navigation is a sequential repetition of point-navigations.

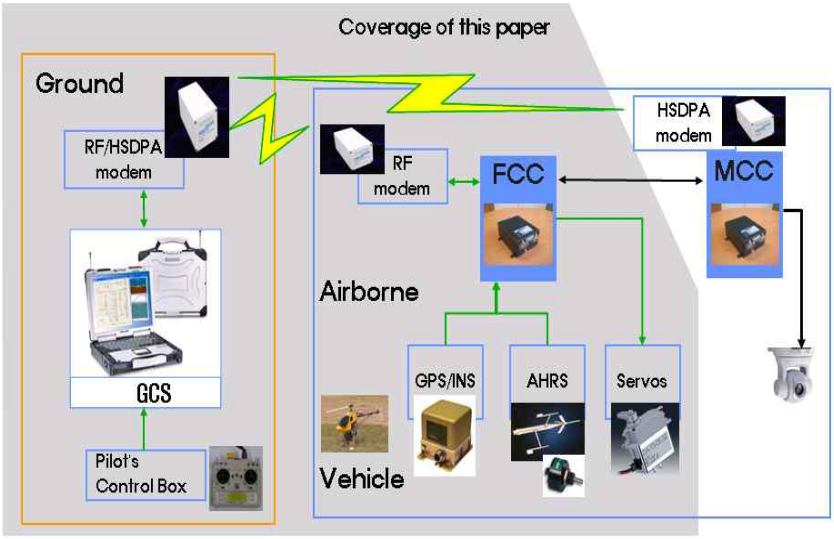

Fig 2. Structure of the HELICOPE

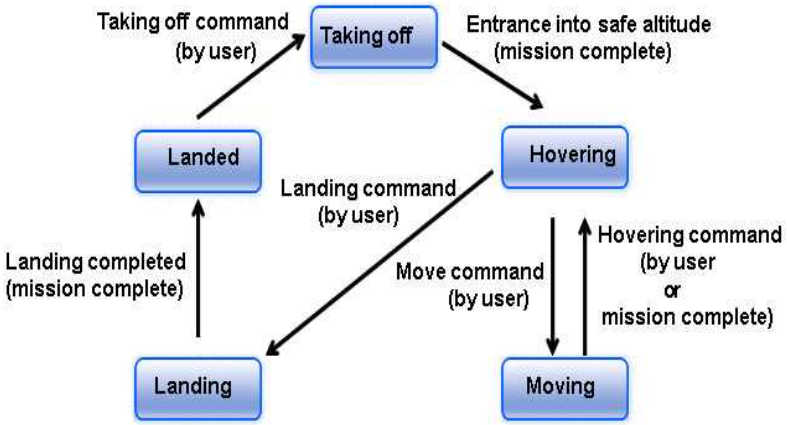

Fig 3. Flight Mode Transition

3.3 Design of the OFP based on the TMO scheme

The OFP basically consists of a TMO instance containing with one time-triggered task, four message-triggered tasks and ODS. The ODS contains data that are periodically read from GPS/INS, AHRS and SWM. Followings are descriptions on the rolls of these five tasks and data contained in the ODS.

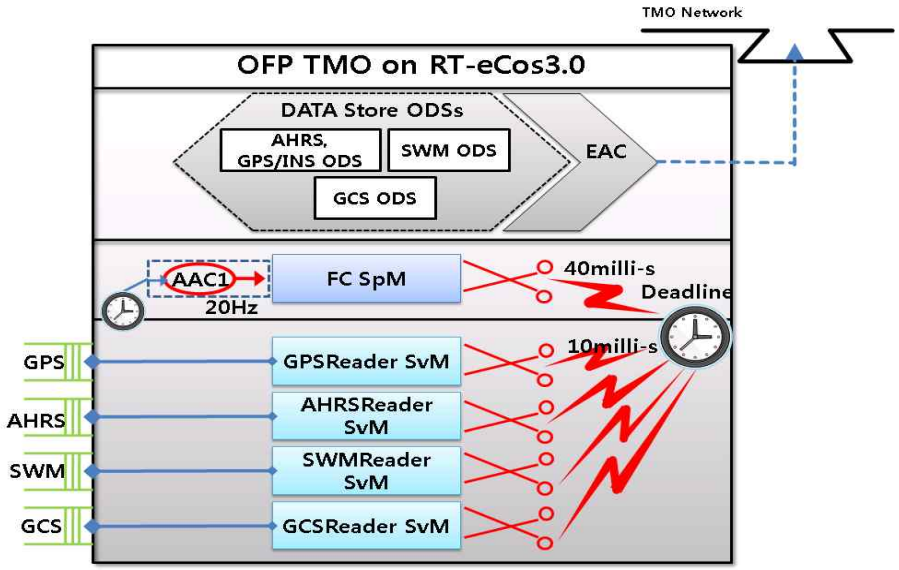

Fig 4. Structure of the OFP-TMO

- GPS/INSReader (IO-SvM): This task collects data sent from the GPS/INS device that periodically sends temporal and spatial data in 10 Hz. The data mainly consists of information on altitude, position and velocity for each direction.
- AHRSReader (IO-SvM): This task collects data sent from the AHRS device that periodically send attitude information of the aircraft in 20 Hz.
- SWMReader(IO-SvM): This task collects data sent from the SWM in 10 Hz. The data consists of current values of four control points (Cyclic-lateral, Cyclic-longitudinal, Collective, Rudder).
- GCSReader(SvM): This task receives various flight commands from the GCS sporadically. Commands from the GCS include information on flight mode, target positions, switch-to-manual-mode, etc..
- FC(SpM) : FC means Flight Control and this task runs with 20Hz period and 40 milli-second deadline. This task calculates the next values for four control points with the ODS and send control values back to the SWM for controlling the helicopter.
- ODS: Besides the data collected from the read tasks, the ODS data also contains information on the current flight mode, the next values for control points and capability parameters of the aircraft.

The frequencies of the SvMs and the FC-SpM can be changed according to the actual frequencies of real devices and the capability of the CPU used. The frequencies of the IO-SvMs above are set to the ones of the physical devices that will be used in the field test.

The calculations being performed by the FC for the four flight modes consist of four basic auto-flight rule-operations. They are SetFowardSpeed, SetSideSpeed, SetAltitude and SetHeading. SetFowardSpeed and SetSideSpeed generate values of two-axis-Cyclic for forward and side speeds. SetAltitude generates a Collective value to rise, descend or preserve the current altitude. Finally SetHeading generates a value to change heading.

For each flight mode, an appropriate combination of these four basic operations is used. For example, in the point navigation mode, the SetHeading operation generates a value for heading (rudder) to the target position and the SetForwadSpeed operation generates values for two Cyclic control points for the maximum speed to the target.

To avoid too rapid changing of attitude of an aircraft, some upper and lower limits are imposed to the values generated by the four basic operations because the maximum and minimum values for Cyclic Lateral/Longitudial, Collective, Rudder are dependent on the kinds of aircraft.[6]

3.4 GCS (Ground Control System)

A GCS is an application tool with which a user can monitor the UAV status and send control messages to FCC. Our GCS provides a graphical interface for easy control.

A protocol for the wireless communication devices (RF modem) between the UAV and GCS has been designed. There are two types of packet, to-GCS-packet for monitoring and from-GCS-packet for control. Both packets consist of header, length, status or control data, and checksum in the tail.

- to-GCS-packet

The FCC system transmits to-GCS-packets containing the UAV status data that are described in table 2 to the GCS at 20Hz.

Table 2. UAV status data

Attitude	Roll/Pitch/ Yaw angle
Location	North, East, Altitude
Velocity	Forward/Sideward/Upward velocity
Control signal	Lateral cyclic/Longitudinal cyclic/ Main rotor collective pitch/Main rotor collective pitch/Tail rotor collective pitch angle
FCC system	Battery voltage

- from-GCS-packet

A from-GCS-packet contains a command message to control the UAV flight mode described in section 3.2.

- User Interface

The user interface has been designed for prompt monitoring and convenient control. Fig. 5 and 6 shows the user interface of our GCS. Various status data including the attitude of an UAV contained in downlink packets are displayed in 2D/3D/bar graphs and gauges of the GCS for easy analysis and debugging the FCC control logic.

The GCS has six widgets for control operations such as landing, taking off, hovering, moving, self-return and update status.

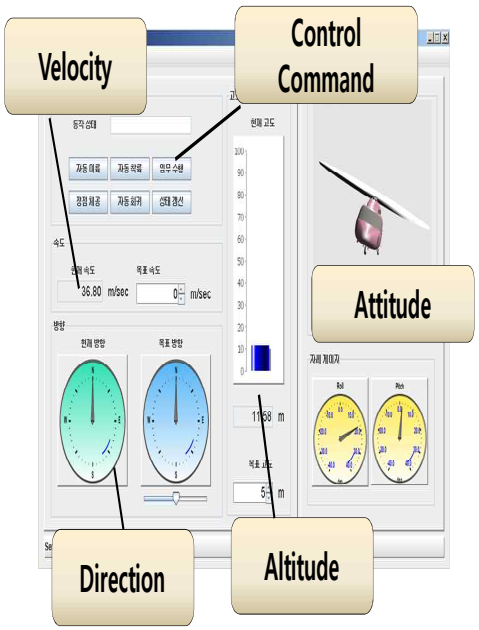

Fig 5. Main interface window

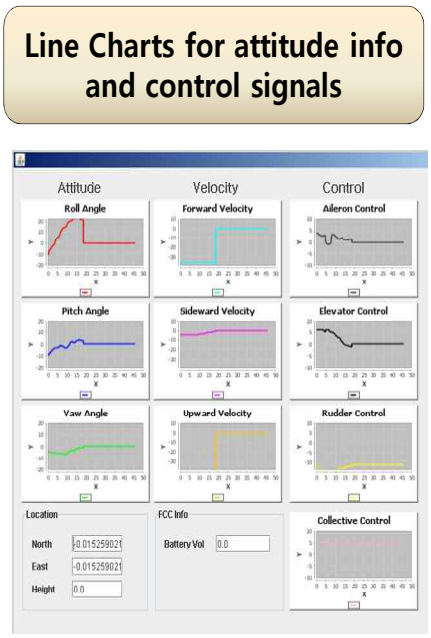

Fig 6. Control signal window

4 Experiments with the hardware-in-the-loop simulation system

To test and verify the OFP in the TMO scheme, we used a HILS system with the open-source based FlightGear-v0.9.10 simulator. The FlightGear simulator supports various flying object models, 3-D regional environments and model-dependent algorithms.

In the HILS environment, the OFP receives GPS/INS, AHRS and SWM information from the FlightGear simulator. Fig. 7 shows the HILS architecture. For the FCC, a board with the ST Thomson DX-66 STPC Client chip has been used to enhance floating point operations.

Among various testing of stability of flying that have been performed, test results of hovering and heading in the point navigation mode are shown in Fig. 8 and 9. In the hovering test, the helicopter takeoffs at altitude 7m, maintains hovering at altitude

17m for 10 seconds and does auto-landing. Fig. 8 shows the desired references and actual responses of the HILS system in this scenario and we can see that there is a deflection of almost 0.5 meter in maximum when hovering. This result is tolerable in our application.

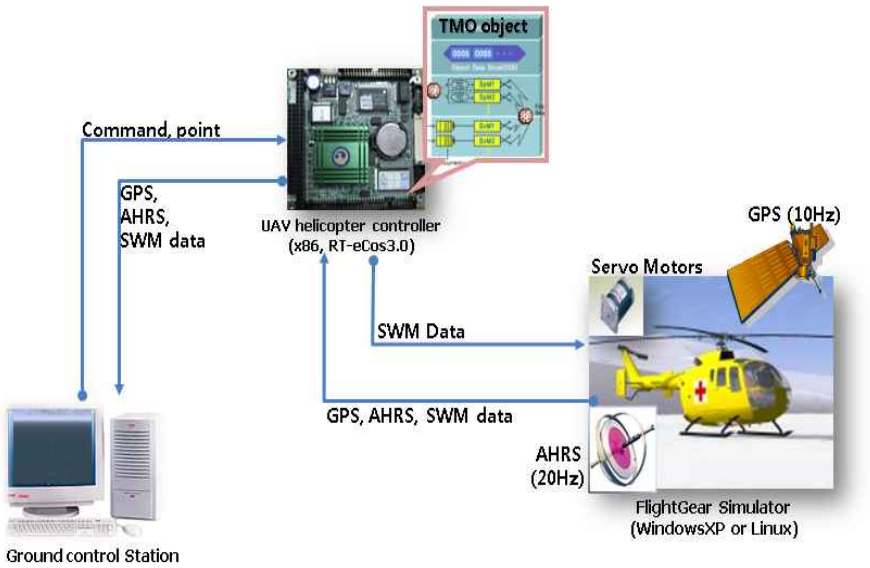

Fig 7. Structure of the HILS

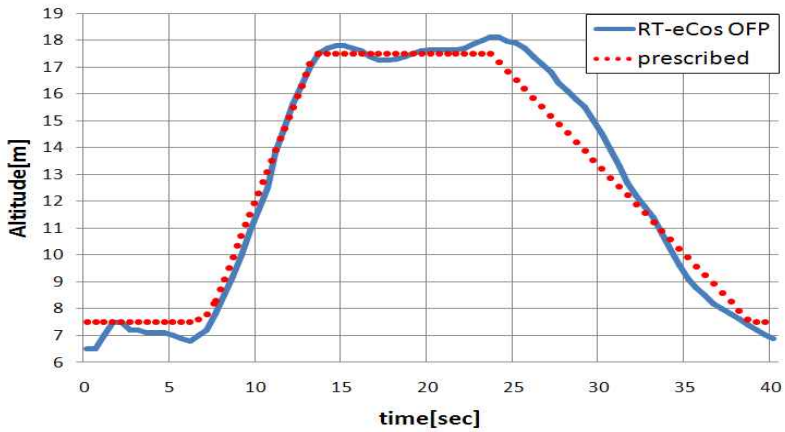

Fig 8. Stability of hovering control

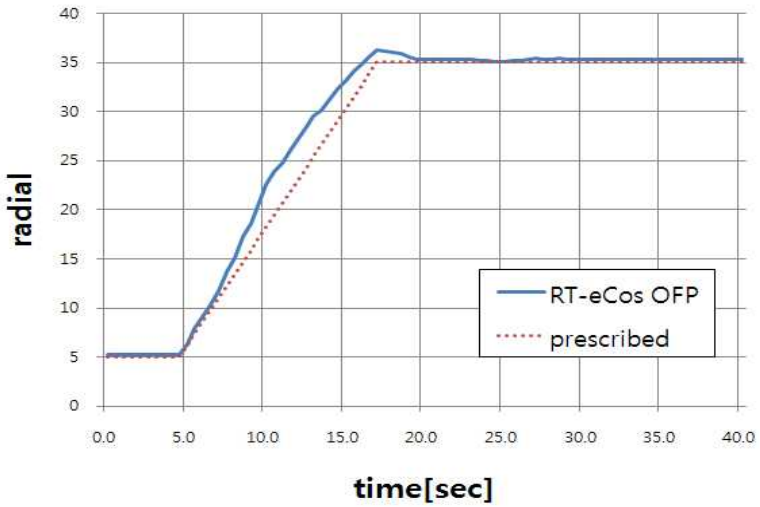

Fig 9. **Heading-control in the point navigation mode**

5 Conclusions and future works

In this paper, we introduced the design and implementation of the OFP (Operational Flight Program) for Unmanned Helicopter's navigation based on the well-known TMO scheme and the RT-eCos3.0 and verified the system using HILS.

Since the OFP can naturally be composed of time-triggered and message-triggered tasks, using of the TMO mode that supports object-oriented real-time concurrent programming is a very well-structured and easily extendable scheme in designing and implementing the OFP. Moreover, we could also find out that the RTOS, RT-eCos3.0, is a very suitable for this kind of application because of its accurate timing behavior and small size.

By finishing the testing of HILS with the flying object supported by the FlightGear, the job to be done further is to do some minor corrections on parameters and detail algorithms that are dependent on a real flying object model. Our plan is to start field testing with a real aircraft in this year.

Acknowledgement. This work was supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency" (NIPA-2009-C1090-0902-0026) and also partially supported by the MKE and the KETI, Korea, under the HVI(Human-Vehicle Interface) project(100333312) .

References

1. Kim, D. H., Nodir, K., Chang C. H., Kim J. G.: HELISCOPE Project: Research Goal and Survey on Related Technologies. In: 12th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, pp.112-118. IEEE Computer Society Press, Tokyo (2009)
2. Kim, J. H., Kim H. J., Park, J. H., Ju, H. T., Lee, B. E., Kim, S. G., Heu, S.: TMO-eCos2.0 and its Development Environment for timeliness Guaranteed Computing. In: 1st Software Technologies for Dependable Distributed Systems, pp.164-168. IEEE Computer Society Press, Tokyo (2009)
3. Kim, K.H. and Kopetz, H.: A Real-Time Object Model RTO.k and an Experimental Investigation of Its Potentials. In: 18th IEEE Computer Software & Applications Conference, pp.392-402. IEEE Computer Society Press (1994)
4. Jenks, S.F., Kim, K.H., et al.: A Middleware Model Supporting Time-triggered Message-triggered Objects for Standard Linux Systems. Real-Time Systems – J. of Time-Critical Computing systems, Vol. 36, pp.75-99. (2007)
5. Kim, J. G., et al.: TMO-eCos: An eCos-based Real-time Micro Operating system Supporting Execution of a TMO Structured Program. In: 8th IEEE International Symposium on Object/ Component/ Service-Oriented Real-Time Distributed Computing, pp. 182-189. IEEE Computer Society Press, Seattle (2005)
6. Kim, S. P.: Guide and Control Rules for an Unmanned Helicopter. In: 2nd Workshop on HELISCOPE, pp. 1-12. ITRC, Konkuk University, Seoul (2008)
7. Kim, H. J., Kim, J.G., et al.: An Efficient Task Serializer for Hard Real-time TMO Systems. In: 11th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, pp. 405-413. IEEE Computer Society Press, Orlando (2008)