Real-Time Reactive Control Layer Design for Intelligent Silver-Mate Robot on RTAI

Hyung Sun Lee, Sang Woo Choi, and Byung Kook Kim
Real-Time Control Laboratory
Department of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology
373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
{hslee@rtcl, swchoi@rtcl, bkkim@ee}.kaist.ac.kr

Abstract

Intelligent robots are capable of handling complex tasks that includes recognition of vocal commands, logical inference, autonomous navigation and manipulation, etc. To accomplish intelligent behaviors, researchers have proposed a number of control software architectures such as tripodal schematic control architecture (TSCA).

To achieve real-time performance for robot's navigation, we have implemented software components in the reactive layer of TSCA on RTAI (Real-Time Application Interface) for our intelligent robot. In this article, we present our structures of the reactive layer components. Real-time performance of the designed reactive control layer is demonstrated via experimental results.

1 Introduction

As medical technology advances with reducing birthrate, many countries are becoming aging or aged societies. To help solve problems of aging society, more and more researches are conducted on development of helper robots [1][2][3], namely *Silver Robots*. These *Silver Robots* should be capable of handling complex tasks including human-robot interaction via voice and gesture, sensing environments using various sensors, logical inference, as well as performing jobs using manipulation and navigation.

In order to perform such complex tasks within given time limits, a solid software architecture should be used [4][5]. *Tripodal Schematic Control Architecture (TSCA)* developed at Korea Institute of Science and Technology (KIST) has been successfully implemented on three versions of their Public Service Robot (PSR) series. TSCA uses a hybrid approach with three layers for its control: deliberate layer, sequencing layer, and reactive layer. The deliberate layer includes software components for human-robot interaction and task planning. The sequencing layer has low-level configuration and process supervisor that manages reactive layer components. The reactive layer consists of real-time components such as resource components for sensor management, controller components for actuator management, and sets of behavior components for basic real-time control. TSCA of PSR series robots were implemented on a Windows-based PC, where the reactive layer is not explicit.

In efforts to develop a next generation silver robot, *Silver-Mate* Robot project has been initiated by the Center for Intelligent Robotics (CIR) at KIST. This project is composed of over twenty research institutes for development of various functionality of the *Silver-Mate* Robot, including our research on real-time implementation. Through years of research, we have proposed a modified TSCA which uses a modified version of TSCA with real-time control capability as shown in Figure 1 [6].

To achieve real-time performance for robot's sensing, navigation and manipulation, we designed and implemented software components in the reactive layer using Real-Time Application Interface (RTAI)[1] on Linux. Software components in the reactive layer require real-time performance, since sensing and movement of mobile-base or manipulator have direct effect on the safety issue of *Silver-Mate*

[1] http://www.rtai.org

Robots.

Throughout this paper, we explain how we designed and implemented the reactive layer using RTAI and evaluate its real-time performance via experimental results. Chapter 2 describes hardware and software architecture of *Silver-Mate* Robot platform. Chatper 3 describes detailed design method of real-time reactive control layer components. Experimental results are shown in Chapter 4. Finally, concluding remarks are made in Chapter 5.

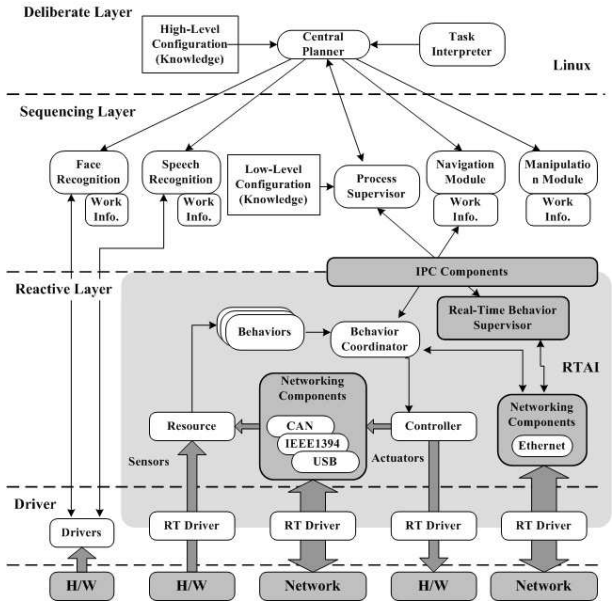

FIGURE 1: *Modified TSCA with Real-Time Capabilities in Reactive Layer*

2 Robot Platform

2.1 Hardware Architecture

Research on real-time reactive control architecture was targeted to a versatile mobile platform from Dasa Technology Inc.[2] shown in Figure 2. Platform consists of three single board computers (SBC) using P-4M 2.2GHz processor and 1GB SDRAM. Head part consists of two stereo vision cameras, a pan-tilt module, and sixteen microphones for voice processing. Vision and voice hardwares are each controlled by separate SBC for application with algorithms requiring heavy computations.

All other hardware components which are largely related to real-time software components are connected to *Main SBC* as shown in Figure 3. A gyro is connected through data acquisition (DAQ) board. Two laser range finders (LRF), two infra-red (IR) scanners and two sonar controllers, each in charge of six sonar sensors are connected to *Main SBC* through an 8-port serial board. Two BLDC motors, the only actuators currently on the platform, are controlled by a DSP-based BLDC control board which is also connected through the serial board. BLDC control board also manages encoders attached to each BLDC motors, two front bumper sensors, and also battery voltage.

FIGURE 2: *Silver-Mate Robot Platform*

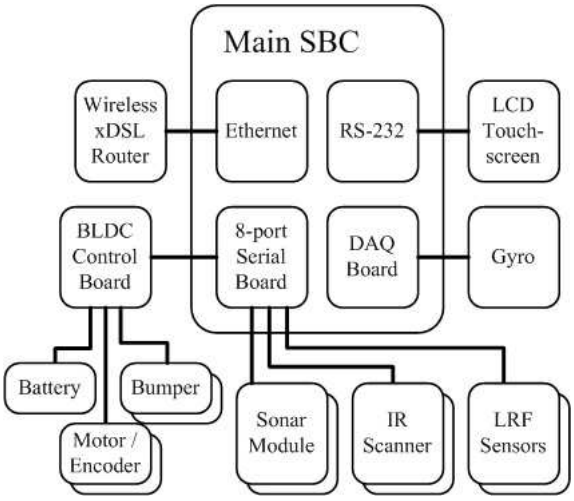

FIGURE 3: *Main SBC*

2.2 Software Architecture

Software components comprising the proposed real-time reactive control layer on *Main SBC* are structured as shown in Figure 4. The reactive layer is

[2]http://www.dasatech.co.kr

a behavior-based type with Resource, Actuator, Behavior, and Behavior Coordinator components as described by Jeon [7]. Each Resource components configures a given sensor hardware for periodic data acquisition, manages hardware operation, and also stores the sensor data in shared memory for other software components. An Actuator component configures and manages actuator hardware and sends generated control outputs. A Behavior component is a basis of reactive action which uses the sensory data to compute the control output. Finally, a Behavior Coordinator (BC) component collects outputs from a set of Behavior components and fuses them for Actuator component.

There are ten Resource components, one for each sensor hardware attached, one Actuator component for the actuator on the platform, one BC component, and two Behavior components. To communicate with components in sequencing layer, RT Task Supervisor (RTTS) component is used to manage all real-time tasks and handle service requests issued to reactive layer.

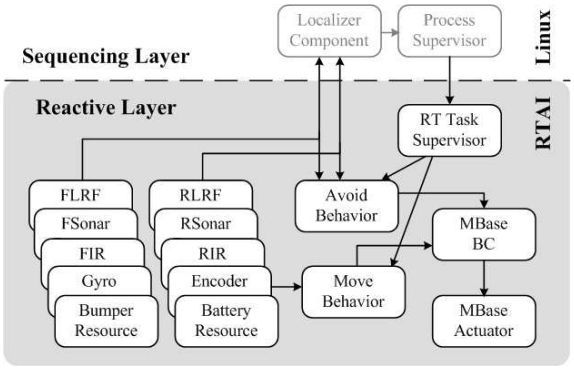

FIGURE 4: *Reactive Layer Software Components*

2.3 Real-Time OS and Drivers

Designed reactive control layer was implemented using RTAI. Version 3.1 of RTAI distribution was used, along with 2.4.27 version of Linux kernel and 0.7.69 version of COMEDI[3] drivers for the DAQ board.

Since the *Main SBC* uses an 8-port serial board, real-time serial driver included in RTAI distribution was modified accordingly. COMEDI driver was also modified to support hardware timer functionality of the DAQ board.

[3]http://www.comedi.org

3 Reactive Control Layer Design

3.1 Resource Modules

3.1.1 Gyro Resource

A gyro sensor is connected to the system through DAQ board. Hence Gyro Resource module uses modified COMEDI driver to convert analog sensor output to digital value periodically, using hardware timer functionality of the DAQ board. Registered callback function is called when a sensor data is available, hence the callback function stores sensor data in shared memory for other software components.

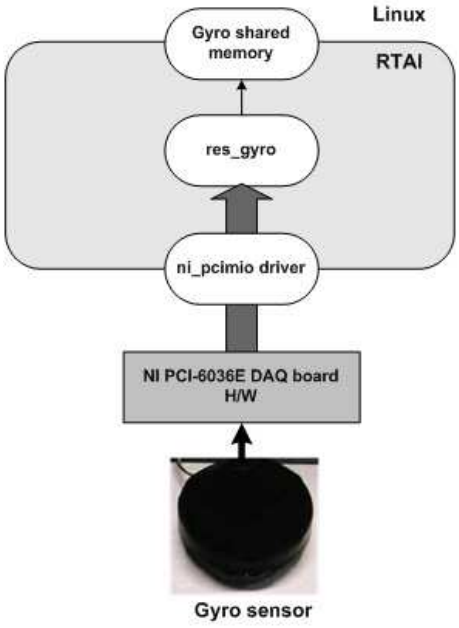

FIGURE 5: *Gyro Resource Structure*

3.1.2 LRF Resources

Two LRF sensors, LMS200 model from Sick, are connected through 8-port serial board. LMS200 can operate in continuous scan mode, so once LRF Resource module configures LMS200 upon initialization, sensor data is received periodically. LRF Resource module largely consists of two parts. One is callback function registered to RTAI serial driver, which collects low level serial bytes from LRF to form a serial packet. Second part is the RT task, which is woke up by the callback function upon reception of complete packets and parses them to extract sensor data. Extracted data is stored in shared memory for other software components.

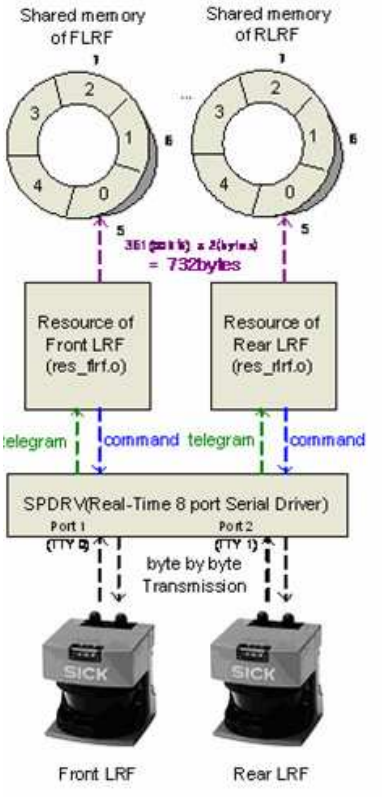

FIGURE 6: *LRF Resource Structure*

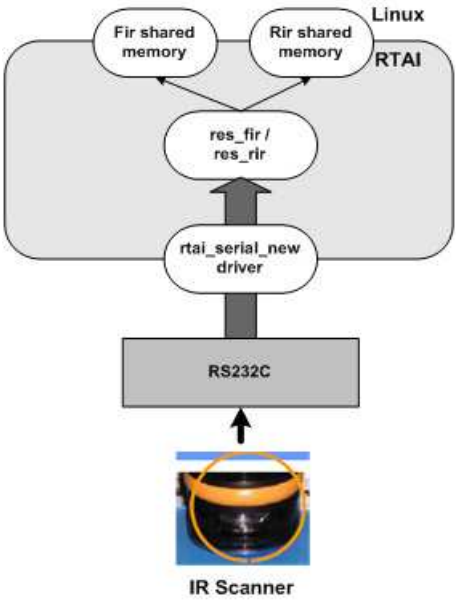

FIGURE 7: *IR Resource Structure*

3.1.3 IR Resources

Two IR scanners, PBS-03JN model from Hokuyo, are also connected through 8-port serial board. PBS-03JN does not support continuous scan mode and requires virtual link management. Data receiving part of IR Resource module uses a callback function and an RT task as in LRF Resources, but it has additional RT task with 200ms period to generate commands to request sensor readings and manage virtual link. Data extracted by reception RT task is stored in shared memory.

3.1.4 Sonar Resources

There are two sonar controller boards on the system, one in charge of six sonar sensors around the front half of the robot platform and the other in charge of six sonar sensors around the rear half. To obtain a sonar data periodically, an RT task is used. At every given time interval, RT task issues fire command, waits for some time and issues read data command for each sonar sensor. Reception of serial data is handled using a callback function and an RT task, similar to LRF or IR Resource modules. Extracted sonar data is stored in shared memory.

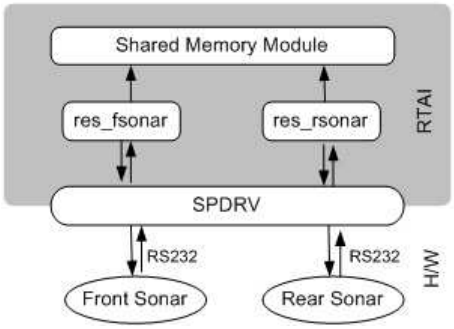

FIGURE 8: *Sonar Resource Structure*

3.1.5 Encoder, Battery, and Bumper Resources

Two encoders of BLDC motors, two front bumper sensors and battery voltage sensor are connected via BLDC control board, as shown in Figure 3. Hence they can only be accessed through serial protocol provided by the BLDC control board. Software component that manages BLDC control board hardware is MBase Actuator module, which is explained in Chapter 3.2, hence Encoder, Battery, and Bumper Resource modules depend largely on functionalities provided by MBase Actuator module.

MBase Actuator provides functionalities to register or unregister a callback function for each of encoder, battery, and bumper hardware. These three Resource modules register their callback functions upon initialization. When a new data is generated, callback function is called and it stores the sensor

data in designated shared memory. Periodic generation of sensor data is a job of MBase Actuator and no other functions are needed for these three Resource modules.

Encoder and battery data are generated periodically. However, unlike other sensor hardwares, bumper event is a sporadic event. Therefore, a new bumper sensor data is only generated when there is a change in bumper status. Reading "current bumper data" can cause blocking of the calling task.

3.2 Actuator Module

Only one set of actuators exists on the platform as shown in Figure 3, thus only one Actuator module exists. We call it MBase Actuator in short for Mobile-Base Actuator. MBase Actuator manages a DSP-based BLDC control board. BLDC control board is connected via serial port and it uses a layer of serial protocol for exchanging command and status data. MBase Actuator module provides a layer of abstraction for the underlying serial protocol by providing a set of API functions. A data structure for command buffer, a serial callback function, and a packet-parsing RT task are used.

As mentioned earlier, MBase Actuator module is in charge of generating encoder, battery, and bumper data. An RT task is used to periodically generate data request commands and a set of callback registration API functions are provided. Registered callback functions are called within the packet-parsing RT task.

In case of emergency, hardware reset button in the BLDC control board may be pressed. To detect such event and re-initialize MBase Actuator module, a watchdog RT task is used to monitor periodic exchange of serial packets.

3.3 Behavior Modules

A *Behaivor* is a basis of action used to determine actual movement of the robot platform. Many different kind of Behavior can co-exist, such as *Move* Behavior that tries to move in a certain direction, *GoTo* Behavior that tries to move to a certain position or *Obstacle Avoid* Behavior that tries to avoid static or dynamic obstacles in its path.

All Behavior modules have similar execution flow. When it wakes up, it reads sensor data needed, computes control output, passes it on to BC component, and suspends itself as shown in Figure 9.

Current implementation of *Move* Behavior receives translational and rotational velocity commands from the sequencing layer and outputs them for BC. *Obstacle Avoid* Behavior reads LRF data to find location of the nearest obstacle and generates an output to avoid it [7].

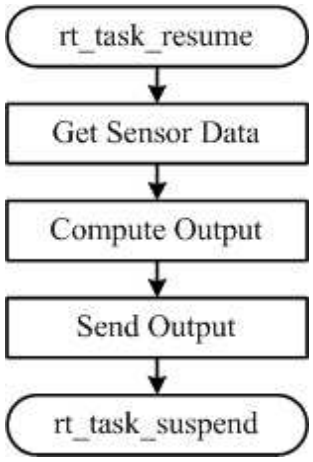

FIGURE 9: *A Behavior Execution Flow*

3.4 Behavior Coordinator Module

A Behavior Coordinator (BC) manages all Behaviors and fuse their outputs to generate actual control output for Actuators. We implemented *MBase* BC for MBase Actuator, which has a few operational modes. For each operational mode, different sets of Behaviors are used. For instance, when MBase BC operates in *Move with Obstacle Avoidance Mode*, both Move and Obstacle Avoid Behaviors are active. For simple *Move Mode*, only Move Behaivor is active.

3.5 Real-Time Task Supervisor

The reactive control layer components were implemented as kernel modules. In order to exchange messages with sequencing layer components, IPC mechanisms provided by RTAI must be used. Real-Time Task Supervisor (RTTS) component's job is to interface these two layers.

RTTS component is made up of a kernel module and a C++ class. RTTS module uses RT-FIFOs to receive service requests issued to the reactive layer and to report status of the reactive layer back to the sequencing layer. RTTS C++ class is used to abstract complex RT-FIFO communication protocol into simple class methods that can be used by other sequencing layer compnents. Current implementation provides methods shown in Table 1

Method	Description
SetOpMode	Set New Opertion Mode
GetOpMode	Get Current Opertion Mode
Move	Issue a Move Commmand
SetPosition	Set Current Position
GetPosition	Get Estimated Position
Stop	Stop Gracefully
EmgStop	Emergency Stop
Activate	Turn on/off BLDC Motors

TABLE 1: *RTTS C++ Class Methods*

4 Experimental Results

To analyze the real-time performance of implemented reactive control layer, two experiments were conducted: one to demonstrate performance of Resource components and the other to demonstrate performance of Actuator-related components.

Resource	Min	Max	Period	ΔP_{max}
Gyro	12.63	29.0	20.81	8.18
Battery	2007.9	2031.6	2019.8	11.9
Encoder	34.375	44.125	39.250	4.875
IR	194.06	206.88	200.47	6.41
LRF	207.56	217.69	212.62	5.06
Sonar	100.81	121.25	111.03	10.22

TABLE 2: *Measured Sample Interval*

4.1 Sensor Data Acquisition

One of the important job of the reactive layer is to manage sensor hardwares and gather sensor data as periodically as possible. There are six kinds of Resource components in our reactive layer. While running all Resource modules at the same time, over a thousand consecutive data samples were collected every three seconds from each Resource modules. Table 2 shows the minimum and maximum values of measured time interval between two consecutive samples. From the relationship among actual sample period, P, difference between maximum and minimum sample delays, ΔP_{max}, minimum sample interval, I_{min} and maximum sample interval, I_{max}, the values of P and ΔP_{max} can be calculated as shown in Table 2.

Calcuated actual sample period, P, and worst-case delay in sample data, ΔP_{max}, shows that generation of sample data using implemented reactive layer is periodic and deterministic.

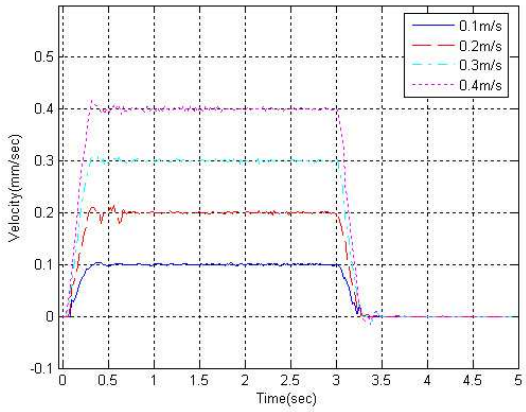

FIGURE 10: *Measured Step Responses*

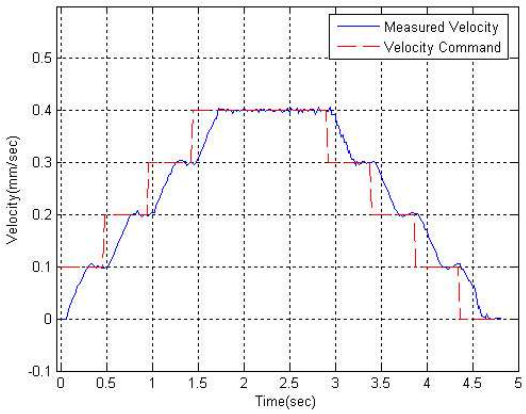

FIGURE 11: *Response to Velocity Commands*

4.2 Response to Velocity Commands

To prevent robot platform from damaging itself, BLDC control board uses accepted velocity commands to create movdified velocity profiles with increasing or decreasing slopes. Four step translational velocity commands from 0.1m/s to 0.4m/s were issued and resulting velocity were measured by reading encoder values every twenty miliseconds. Resulting step velocity responses are shown in Figure 10, where velocity command of 0m/s were issued three seconds later to stop the platform, and acceleration/deceleration time is set to 0.25s. Figure 11 shows velocity response to a series of velocity commands. These results show that implemented reactive control layer can accomplish navigation commands from sequencing layer within predictable amount of transition time.

5 Conclusion

In this paper, we explained how we implemented the real-time reactive control layer structure of modified TSCA architecture for *Silver-Mate* robot using RTAI. Experimental results demonstrates that our implementation is able to obtain sensor data periodically in a predictable manner and execute navigation commands in real-time.

As a further work, exhaustive testing will be performed on our implementation. Furthermore, our implementation will be integrated with resulting software components of other researchs for final integration and test. Our research will focus on use of Controller Area Network (CAN) and Ethernet for distributed computing environment for intelligent mobile robots.

6 Acknowledgements

This research was performed for the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Commerce, Industry and Energy of Republic of Korea.

References

[1] Hans, M., Graf, B. and Schraft, R. D., 2002, *Robotic Home Assistant Care-O-bot: Past–Present–Future*, in proceedings of IEEE RO-MAN '02, pp. 380-385.

[2] Hans, M. and Graf, B., 2004, *Care-O-bot II – Development of a Next Generation Robotic Home Assistant*, Autonomous Robot Journal, March 2004, vol. 16, issue 2, pp. 193-205.

[3] Kim, C. H., Kim, S. J. and Kim B. K., 2004, *RTAI Based Real-Time Control of Robotic Wheelchair*, 6th Real-Time Linux Workshop, Singapore.

[4] Hans, M., 2004, *The Control Architecture of Care-O-bot II*, Advances in Human-Robot Interaction, vol. 14, pp. 321-330.

[5] Kim, G., Chung, W., Kim, M. and Lee, C., 2003, *Tripodal Schematic Design of the Control Architecture for the Service Robot PSR*, in proceedings of the IEEE Conference on Robotics and Automation, Taipei, Taiwan, pp. 2792-2797.

[6] Lee, H. S. and Kim, B. K, 2004, *Research on Real-Time Implementation of Control Architecture for Silver-Mate Robots*, 3rd Technological Workshop of Center for Intelligent Robotics, Cheongpung, Republic of Korea.

[7] Jeon, S. Y., Kim, H. J., Hong, K. S. and Kim, B. K., 2005, *Reactive Layer Control Architecture for Autonomous Mobile Robots*, in proceedings of the 3rd International Conference on Mechatrinics and information Techonology (ICMIT 2005), Chongqing, China.