Vol. 7, No. 1, pp. 1-4, June 2008

INTERNATIONAL JOURNAL OF SAFETY
www.kosos.or.kr

Designing Factory Safety Monitoring Robot Using Microsoft Robotic Studio

Byoung Gook Loh*

Department of Mechanical Systems Engineering Hansung University, Seoul 136-792, Korea

(Received January 12, 2008; Accepted June 15, 2008)

Abstract : Application of the Microsoft robotics studio (MSRS) to the design of a factory safety monitoring robot is presented. Basic structures of the MSRS and the service are introduced. The service is the key building block of the MSRS. Control of the safety monitoring robot is performed using four basic services: 1) the robot service which communicates with the embedded micro-processor and other services, 2) the sensor service that notifies the subscribing services of the change of the sensor value, 3) the motor service which controls the power levels to the motors, 4) the drive service which maneuvers the robot. With built-in capabilities of the MSRS, control of factory safety monitoring robot can be more easily performed.

Key words: microsoft robotics studio, factory safety monitoring robot, service, serial communication

1. Introduction

In a general manufacturing factory environment, there exist safety hazards such as slippery surface, leakage of toxic gases, and scattering dust, all of which require continuous monitoring to minimize health-risks to human operators. One easy way to monitor the factory environment is to dispatch multiple mobile robots equipped with sensors to gather information on the factory environment. Gathered measurements are sent to the central control unit and at the same time nearby operating workers are alerted when the levels of measurements exceed certain preset values.

As the prices of sensors and actuators decrease the availability and selection of mobile robots have greatly increased at a fast rate[1]. Control of the mobile robot has been done by writing robot-specific application peripheral interfaces (APIs) which is time-consuming and costly, because one API developed for a mobile robot can not be interchangeably used to other robots[2]. Accordingly, a universal robot control program needs to be devised in a way that most of personal computers currently run on Windows. Microsoft Robotics Studio (MSRS) addresses this issue[3-4]. The MSRS is a generic robot operating system based on the popular .NET framework, which supplies robot researchers with a standard and convenient method to control the mobile robot. In this study, application of MSRS to design of a mobile robot enhancing safety of the factory environment is investigated.

*Corresponding author: bgloh@hansung.ac.kr

2. Structure of MSRS

The MSRS runtime consists of three low-level runtimes: Concurrency and Coordination Runtime (CCR), Decentralized Software Services (DSS), and the .NET Common Language Runtime (CLR) as shown in Fig. 1.

The CCR is a key component that connects the MSRS runtime to a service and one important feature

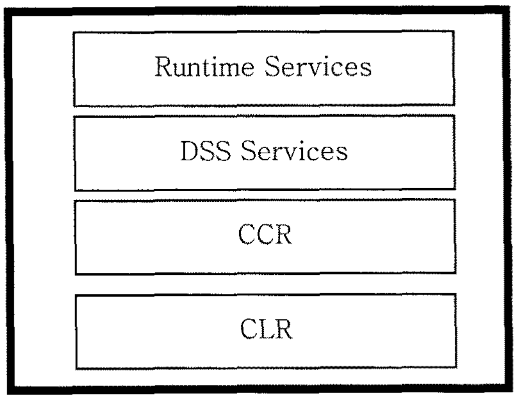

Fig. 1. Structure of MSRS

of the CCR is the ability to perform asynchronous operations in any applications using .NET library. Also, the CCR manages threading, thereby freeing developers from creating their own threading codes. Driving of the mobile robot requires asynchronous operations of actuators and sensors. Lack of asynchronous operation in the mobile robot significantly limits robot's mobility. The CCR coordinates the threads required to asynchronously operate the sensors and the actuators. Without the CCR, multiple callback routines need to be written and managed using the interrupt service routine. With DSS, which is based on the Representational State Transfer (REST), developers can monitor services interactively in real time. The REST defines the way resources are allotted and the requests are transferred via the Web. To control robots remotely, REST principles can be employed .

3. Construction of MSRS Robot Application

Services are the basic building block in controlling a mobile robot with the MSRS. Services contain the codes to read sensors and to send commands to actuators. Fig. 2 shows the construction of a service. The contract identifier identifies the service and defines the messages which can be sent to a service. Operations from other services arrive at the main port and the service handler corresponding to the operation is invoked and the state of the service can be retrieved or modified. A service can be parted with other services so that the change of the state in a service can be notified to other partnered services.

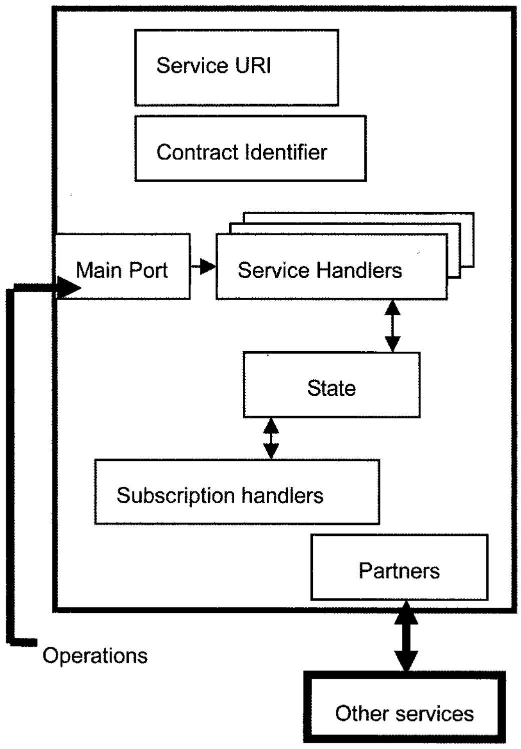

Fig. 2. Construction of service

A robot application based on the MSRS is typically comprised of multiple services to perform a pre-defined task. Fig. 3 shows a robot application service which has four basic building blocks: the sensor service, the motor service, the drive service, and the robot service. The sensor service receives the sensor change notification from the robot service and updates its sensor values and in turn, notifies other subscribed services of the change in sensor measurements. The motor service handles the actuation of the motors of the robot by supplying a certain power level to the motor. The drive service also actuates the motor but in a more controlled way such as "go forward", "turn right", "turn left", and "go backward". The robot service communicates with the robot hardware, generally a microprocessor through a serial communication. The microprocessor embedded in the robot controls the motor to drive robot and read the sensor measurements to gather information on robot's states such as position, velocity, existence of nearby obstacles, and so on. The robot is interfaced to the MSRS via the serial communication between the robot service and the robot hardware.

To explain how the robot service works, we will take X-Bot manufactured by Yujinrobot for an example. The

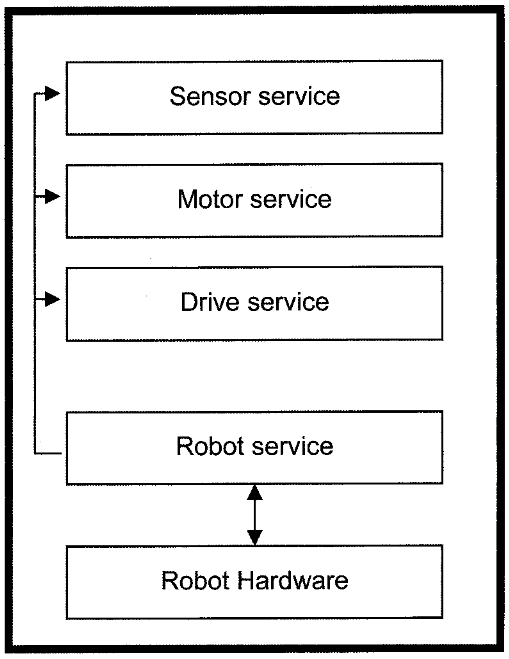

Fig. 3. Robot service with four services

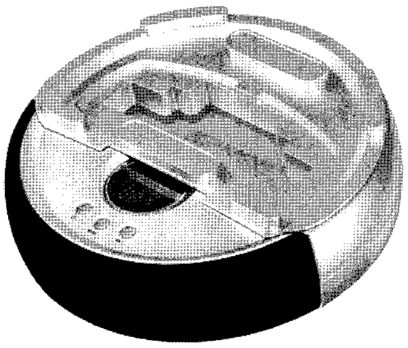

Fig. 4. Xbot (Yujin Robot)

preamble	data5	data4	data3	data2	data1	data0	end mark
0xBB	8bits	8bits	8bits	8bits	8bits	8bits	0xEE

Fig. 5. Serial communication protocol (read operation)

serial communication between the X-Bot and the MSRS is carried out with eight bytes data as shown in Fig.5 with a 10 msec interval.

The XBot is equipped with ground sensors, bumper sensors, infrared (IR) sensors, and encoders. Data arrangement for the serial communication is as follow:

Ground Sensors = data5
Bumper Sensors = data4
IR Sensor = data3
Left Encoder = data2
Right Encoder = data1
Extra Sensor Data Interface = data0

The bumper sensors (touch sensors) are located on the front and sides of the XBot, notifying contact of the robot to the near-by obstacles, while IR sensors are the proximity sensors which warn the robot when the robot comes in close proximity of the obstacles. The left and right encoders measure the rotation of the wheel. With measurements, the driving distance and speed of the robot can be estimated.

The Xbot comes with basic sensors tailed to drive the robot along a designated path in a way that minimizes or avoids contact with the neighboring objects but in order to monitor the factory environment, extra sensors such as a dust sensor, an accelerometer, a smoke sensor, etc. need to be incorporated to the robot. Thus, an additional micro-processor is required to be added to the system or the computer on which the MSRS runs should be used to interface the additional sensors.

The control of the motors can be performed via the

preamble	data1	data0	end mark
0xBB	8bits	8bits	0xEE

Fig. 6. Serial communication protocol (write operation)

serial communication as follows:

```
byte[ ] Output = new byte[4];
Output [0] = (byte)0xBB;
Output [1] = (byte)(PowerToLeftWheel);
Output [2] = (byte)(PowerToRightWheel);
Output [3] = (byte)0xEE;
serialPort.Write(Output, 0, Output.Length)
```

Fig. 7 shows inter-relations of the Xbot services which communicate each other by exchanging messages. When

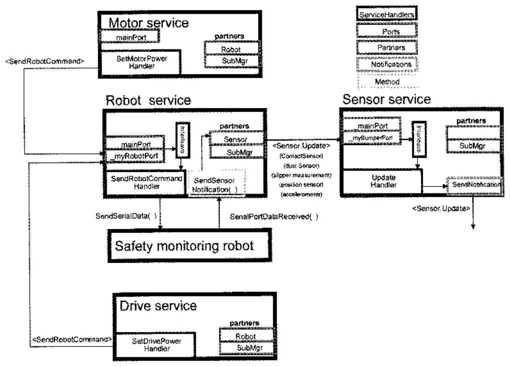

Fig. 7. Inter-relations of Xbot services

Handlers

SetMotorPowerHandler( ) :

Function ->Set motor power

SendRobotCommandHandler( ):

Function -> Send robot command to other services

SendBumperNotication ( );

Function -> Notify when the bumper is pressed

SetDrivePowerHandler ( );

Function -> Set drive power of motor

Messages:

<SendRobotCommand>

<SerialPortDataReceived>

<Bumper.update>

Fig. 8. Handlers and messages of Xbot

a message arrives at the port which is monitored in real time, the service receiving a message executes the relevant message handler as shown in Fig. 7.

One of the main advantages of building a mobile robot using the MSRS is easy interface with peripheral devices such as a webcam, wireless communications, etc. and a plethora of sub-routines arising from open architecture of the MSRS. Developers worldwide using the MSRS can submit and share their own codings, while other robot programs are intended to be used for certain robots and can not be re-used to other robots.

To interface a new hardware, which is not currently supported, to the MSRS, we need write an onboard interface that controls the robotic hardware. Also, services that communicate with the onboard interface and perform pre-defined tasks need to be written.

The onboard interfaces basically control the sensors and actuators attached to the microprocessor and communicate with the MSRS via the serial communication. And robot-specific motor service, robot service, drive service, and sensor service need to be authored in a similar manner shown in Fig. 7.

The MSRS also contains the simulation module. In the simulation, multiple robots and its environment including obstacles and terrain, on which the robots ride, can be easily created and simulated using the physics engine which handles collision, gravity, and complicated calculations so that the way the robots interacts with its environments can closely replicates the way they would do in real world. Therefore before building a robot which is costly and time-consuming, the working and performance of the robot to be built can be tested as realistically as possible.

In summary, the MSRS compatible mobile robots can be a good platform to build a factory safety monitoring mobile robot.

4. Conclusions

The control of a mobile robot can be more easily accomplished by the MSRS than robot-specific controllers because 1) the MSRS runs on Window, which gives advantages of easy accommodation of windows-based peripherals such as webcam, voice recognition, etc. and 2) basic functions of the mobile robot such as sensing and driving can be more efficiently handled without using callback routines, 3) portability of a program developed using the MSRS can be greatly enhanced. Main purpose of building a mobile robot for monitoring factory safety is to check the safety of the factory. Therefore, if the MSRS is used for controlling the robot, such peripheral tasks as navigation, obstacle avoidance, and communications can be easily accomplished and the developer can concentrate on the main task, i.e. how to perform effective gathering of information on the factory environment which might be harmful to the operators.

Acknowledgements

This research was financially supported by Hansung University in the year of 2007.

References

[1] B. Gates, "A Robot in Every Home", Scientific AmericanMagazine, 2008, http://www.sciam.cpom/article.cfmid=a-robot-in-every-home

[2] K.Johns and T. Talyor, "Professional Microsoft Robotics Developer Studio", Wiley Publishing inc.,2008.

[3] S. Morgan, "Programming Microsoft Robotics Studio", Microsoft Press,2008.

[4] J. Jackson, "Microsoft robotics studio: A technical introduction", IEEE Robotics & Automation Magazine, no.4, vol.14, pp.82~87, 2007.