Linux-based Real Time Monitoring System of Mobile Robots

SeungHo Cho
Dept. of Electronics Eng.
Pusan Nat. Univ.
Sungho123@pusan.ac.kr

HyoSik Choi
Dept. of Electronics Eng.
Pusan Nat. Univ.
chs3040 @pusan.ac.kr

JangMyung Lee
Dept. of Electronics Eng.
Pusan Nat. Univ.
jmlee@pusan.ac.kr

Abstract

Real time monitoring is necessary for the dynamic obstacle avoidance and trajectory tracking of mobile robots. However there are several problems in implementing a real time operating system: 1. It is expensive to develop since a high license fee is required and 2. Standards for the real-time systems are not well-established yet. For an educational system under these difficulties, a Linux-based real time system is possible to receive the display data with the minimum time delay by the kernel compiling to fit to the user system. Notice that Windows OS is suffering from code bloating phenomena, which may result in a long operation time. An i-BOT is developed for a demonstration system to show the localization schemes using the RFID and ultrasonic sensors, named as iGS (indoor GPS). An RFID is designated to synchronize the transmitter and receiver of the ultrasonic signal, and the traveling time of the ultrasonic signal has been used to calculate the distance from the iGS to a mobile robot.

Keywords: real-time, Linux, localization, RFID

1. introduction

In the near future, various new services come to our daily lives through the ubiquitous computing and network environment. Especially for a mobile service robot, it is very important to estimate the position and to recognize human beings and obstacles at any instance. The localization and recognition techniques are very important to provide various services to human beings in various environments [1].

Since the mobile robot is able to move in the working area, it can be used for various tasks unless it causes collisions to the obstacles in the environment. On account of its mobility, the mobile robot can replace human beings for the hard and dangerous tasks. To execute the given tasks successfully, the mobile robot needs to identify its own position for a certain task including sensing the environment and controlling the motors. Unexpected disasters may happen through the malfunction of the mobile robot in synchronizing to the environment. Therefore the mobile robot is a real time system to move to the goal as well as to avoid obstacles concurrently,

Most of the robot control systems are being developed using Windows O/S. Windows has many advantages of easy developing environment, such as, supporting abundant device drivers and multitasking. However it cannot support hard real time capabilities since the Windows is not designed for the real-time system originally, which emphasizes the graphic display. Moreover it is difficult and takes a lot of time to develop the device drivers [2] in Windows O/S.

Basically a real time O/S is suitable to implement a real time control system. The commercial QNX and VxWorks have the hard real time capabilities. However, the price is too high to be used for education and research purposes. Their support of device drivers for custom designed controllers is so weak that the expendability is very low [3].

In this paper, for the autonomous navigation of the intelligent robot, i-BOT, a real time control software is developed and evaluated. To keep the hard real time capabilities, Linux with RTAI has been used.

In section 2, the development environments of real time Linux and Windows O/S are analyzed to show the differences. Section 3 illustrates the hardware platform of i-BOT and the constitution of iGS (indoor GPS), and section 4 describes the operating platform for the autonomous navigation and the position estimation algorithm of iGS, and the simulation and experimental results follow in section 5. The ideas and contributions of this paper are summarized in section 6.

2. Comparison of Real Time Linux and Windows

2.1 Stability of Linux control system

Linux is a next generation O/S which can replace the Windows O/S which has code bloat caused by the continuous addition of functions when they are required. That is, the unnecessary code increase (Actually Windows has about 35 to 40 million unnecessary lines.) degrades the operating performance in the real time control system. On the contrary, the real time Linux can be ported on the small memory such as a floppy disk since it can be reconfigured by the complier to fit to the user system. On this reason, the Linux has been widely utilized for embedded systems.

2.2 Flexibility of development environment

There are several reasons that the Windows platform has been dominant so far. It provides a user friendly interface and various services, and the educational cost is very low. Now the Linux also has almost all the advantages of Windows by the continuous development. Figure 1 is a screen shot of Visual Studio which is the Windows development environment, and Fig. 2 is a screen shot of QT which is the Linux development environment. Since both of these two environments support C and C^{++} as a basic programming language, the developing environments are almost the same.

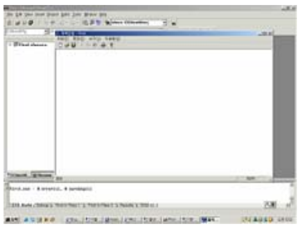


Fig. 1. Visual Studio Screen shot.

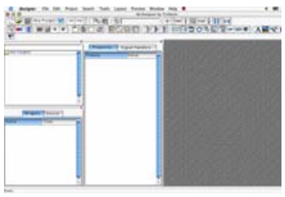


Fig 2. QT screen shot.

Linux also enables to form the tele-debugging environment by a terminal, which is very rare in Windows. Moreover the source codes of Linux are open to public. Therefore it is possible to modify the source codes according to developing objects and application environments promptly and easily. However the performance of the real time system in the Windows degrades on account of the additional codes for the various functions.

Therefore in this paper a mobile robot controller is developed based on a real time Linux, and the performance is compared to the Window based system. The modularity and real time performance of the Linux based system are emphasized to show the superiority of Linux to Windows.

3. i-BOT Platform and iGS

3.1 i-BOT Platform

The mobile robot used in this research, i-BOT, is made by Ninety Corp., which is shown in Fig. 3. For the power supply, 12 V rechargeable batteries are used. Two active wheels and one auxiliary wheel are driving the mobile robot. For the simplicity of driving mechanism, two stepping motors are used. The mobile robot can be controlled wirelessly from a PC through Bluetooth.

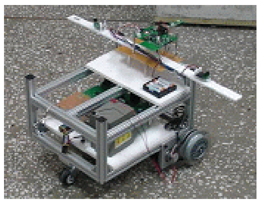


Fig. 3. i-BOT system.

3.2 Structure of the iGS system

The proposed localization scheme can be applied for any of moving objects, such as, home robot, service robot, humanoid robot, *etc.* The indoor environment consists of columns, corners, and two dimensional flat walls which can include desks, tables, and computers depending on the size of the objects.

There are four beacons in the workspace of i-BOT. Each beacon is located at a specific corner with a specific coordinates and it transmits the ultrasonic signal when it is requested. Notice that most of cases, it is convenient to install the beacon at the corner of ceiling. On the mobile robot, there is a receiver which detects the arrival time of the ultrasonic signal from the beacon. The receiver has a RF transmitter in the same body, which sends out an ID for a specific beacon assigned to the ID to request the ultrasonic transmission. To identify the orientation of the mobile robot, there are two receivers on the robot. The beacon receives an RFID from the receiver, and checks whether the ID matches to itself or not. When the matching succeeds, it sends out the ultrasonic signal to the receiver on the mobile robot.

4. O/S platform for autonomous navigation

4.1 Real-Time System

To incorporate various functions to i-BOT, not only improvement of the mechanical structure but also design and implementation of real-time software structures are important [4]. The real-time software governs the efficient flow of resources and defines data-flows among the elements, for the real-time control system. The real-time position estimation and object recognition are necessary for higher level functions, such as, the autonomous navigation and recharge. For the precise estimation and recognition, the data which have different physical dimensions from various sensors need to be processed and fused efficiently. The autonomous mobile robot may confront with unwanted collisions to obstacles when one of these elements does not keep the time constraint.

For the autonomous navigation of i-BOT, many tasks, such as, motor control, sensing, position estimation commands, need to be processed in real time. Each task has its own control period, priority, execution time, and computational complexity. Since the hard real-time properties are required in the control of motors using sensor data in these tasks, a real-time operating system which manages the resources efficiently is essential to implement the real-time control system.

The most important feature for the controller is real-time capability. Notice that the real time operating system is a higher level program to make the tasks being executed in real time. The operating system does not solely aim at the high speed operation. Within the allowed time slot, it provides the desired output utilizing the predictable system functions. Both of Windows and Linux are not possible to execute hard real time tasks, by themselves. Therefore, each operating system suggests real time kernels with the basic operating system to be used for the necessary real time system. For Windows, RTX is the most popularly used kernel. There are several on-going projects to make the Linux a real time operating system, such as, RTLinux and RTAI. The real time operating system supports the programs of multi-tasking structure with time tags, and also provides communication synchronization among tasks and scheduling mechanisms to implement the real time systems.

Linux is a performance-oriented O/S, which has the round-robin scheduler. That is, each task holds the token in a fixed period. Therefore the real time processing cannot be guaranteed. Even though a preemption function is added in the recent kernel version 2.6x, it is still not suitable for the hard-real time system, such as, a mobile robot system. There is a Linux based real-time operating system, RTLinux which is commercially available but applicable to very limited types of processors.

As a sort of interface program, RTAI is developed and it is free of charge. The RTAI provides most of the hard real-time properties, such as, task pre-emption and priority setup/inheritance. Therefore it is suitable for the real-time systems under Linux operation system.

Figure 4 shows the basic structure of RTAI. The Linux kernel is treated as a task. RTAI and Linux kernel can be interfaced to the hardware by HAL (Hardware Abstract Layer). HAL manages to execute the real time tasks preferentially, while it leaves Linux kernel and the processes working in the kernel as the lower priority tasks in RTAI [5].

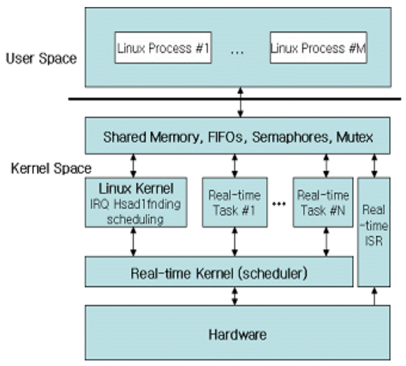


Fig. 4. Structure of RTAI.

4.2 Hierarchical control architecture

There are three levels in the architecture depending on the execution period and characteristics: 1. The highest level is the task goal which sets up the goal of the navigation, 2. The second level is the path planner which generates a path to the goal, and 3. The lowest level is the trajectory generator which specifies location and time pairs of the mobile robot on the path.

When the final goal for the autonomous navigation is determined, the highest level monitoring system plans the trajectory to the goal by selecting appropriate via and through points for the mobile robot to follow.

The path planner decides the existence of obstacles and changes the path to avoid the obstacles. For the obstacle detection, the ultrasonic transmitter sends out the signal every 100 msec, and the corresponding receiver computes the TOF (time of flight) of the ultrasonic signal and detects the obstacles within the dangerous range. After the collision avoidance, the current and goal positions are newly set in the monitoring system to execute the trajectory planning again.

In the lowest level, the stepping motors are controlled to follow the trajectory by the timer interrupt service routine in every control cycle, which provides a frequency output to control the motor.

The obstacle avoidance routine is activated when there are some static or dynamic obstacles in the path of the mobile robot, i-BOT, which senses and avoids the possible collisions using the ultrasonic sensors and the two stepping motors, respectively. When it is not possible to avoid the collision, the routine may send out STOP command to the motor controllers. After the successful avoidance, the current mobile robot follows the desired trajectory to the goal.

4.3 iGS position estimation algorithm

The localizer starts to count when it sends out an RFID signal assuming that the transmission time to a beacon is negligible. When a beacon receives its RFID, it sends out the ultrasonic signal immediately. The counting at the localizer continues until it receives the ultrasonic signal. After the distance computation using the counter value which corresponds to the TOF of the ultrasonic signal, the localizer sends out another RFID to the next beacon. When the distances to all the expected-nearby beacons are measured, the coordinates of the mobile robot is going to be calculated.

The TOF which can be calculated by the counter is basis for the computation of the distance between the ultrasonic transmitter (beacon) and receiver (localizer) as follows:.

$$v = 331.5 + 0.6 \times T \quad \text{[m/sec]} \tag{1}$$

$$s = n \times C - t_d \quad \text{[sec]} \tag{2}$$

$$r = v \cdot s \quad \text{[m]} \tag{3}$$

where T is the room temperature, C is the period of counter clock, n is the number of counter, t_d is the time delay for ultrasonic signal detection, s is the total freight time of the ultrasonic signal, and r is the distance between the beacon and the mobile robot.

The synchronization between the beacon and the receiver on the mobile robot is very important in measuring the TOF of the ultrasonic signal without interferences. To select a beacon at a time, an RFID is transmitted to the beacon from the mobile robot. At the moment of the RFID receipt, the corresponding beacon transmits the ultrasonic signal to the receiver. Therefore there is an RF receiver with a specific ID and an ultrasonic transmitter at each beacon; there is an RF transmitter and an ultrasonic receiver on the mobile robot.

For the localization of the mobile robot using the triangulation technique, at least three beacons are necessary for a mobile robot in the 3D space..

5. Simulations and experiments

To implement and to analyze the performance of a mobile robot control system under Linux O/S, a test-bed is implemented using iGS and i-BOT as shown in Fig. 5. The height, length, and width of the iGS space are 3.0 m, 3.0 m, and 2.0 m, respectively. The speed of i-BOT is kept below 35cm/sec considering the localization accuracy and safety of the mobile robot.

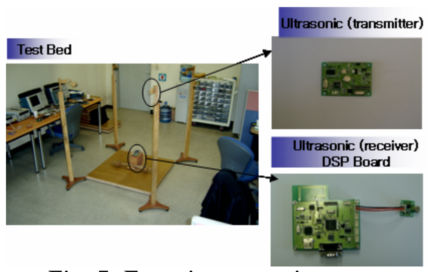
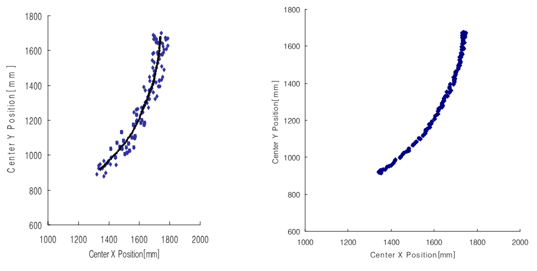


Fig. 5. Experiment environment.

Monitoring program manages the display of the position data measured by iGS and the counter values of the stepping motors sent through the Bluetooth, and it also sends out the motor control commands. In this paper, the monitoring program is implemented on a PC with the Linux using QT program, which gathers the data through Serial-COM and displays on the monitor.

To make the ultrasonic receiver free from reflected interference signals. An RFID is sent out every 20 ms for each beacon. The total data processing time of iGS to obtain the location coordinates is 90 ms.



(a) Trajectory estimation. (b) EKF results.
Fig. 6. Trajectory of a moving object.

Figure 6(a) shows experimental trajectories of the mobile robot: the black solid line represents the actual trajectory while the rectangular marks represent the estimated trajectory by the triangulation method. In the middle of the path, there are filled rectangular marks which represent the positions estimated by only two sensor data. Fig. 13 represents the trajectory obtained by the EKF (extended Kalman filter) using the data set from the triangulation method. By applying the extended Kalman filter, the error between the real and estimated trajectories has been reduced to 20 % as it is clearly recognized by comparing Fig. 6(a) and Fig. 6(b).

6. Conclusion

A flexible controller is designed to satisfy the various requirements of customers working on the Linux environment. RTAI has been utilized to form a real time control system based on Linux, since its source codes are open to public and it has higher flexibility than Windows to satisfy the controller specifications and environment. Through the various experiments, RTAI is proved to be efficient for modular and real-time programming. A monitoring system is specifically designed for the improvement of i-BOT control precision by taking advantage of this Linux-based real time O/S. It is concluded that the i-BOT working in the Linux environment developed in this research, is useful and flexible for education and researches.

Acknowledgements

This Work was supported in part by MIC&IITA through IT Leading R&D Support Project

REFERENCES

[1] S. Singh, “Obstacle detection for high speed autonomous navigation,” Proc. Of IEEE Int. Conf. on Robotics and Automation, pp. 2798-2805, 1991.

[2] C. H. Lee and C. Mavroidis, ”PC based control of and mechatronic systems under MS-windows NT workstation,” IEEE/ASME Trans. on Mechatronics, vol.6, no. 3, pp. 311-321, 2001.

[3] W. F. Lages and E. M. Hemerly, “Linux as a software platform for mobile robots,” submitted to IEEE Transactions on Software Engineering, 2000.

[4] I. S. Song and F. Karray, “Software architecture for realtime autonomous agents : a case study for digital train system,” Proc. of IEEE International Symp. on Intelligent Control, pp. 403-408, 2002.

[5] L. Dozio and P. Mantegazza, “Linux real time application interface (RTAI) in low cost high performance motion control,” Motion Control 2003, a Conference of ANIPLA, Italy, March, 2003.

[6] SeungBu Kim, Gi-Gun Nam and JangMyung Lee, “Indoor Localization Scheme of a Mobile Robot Using RFID” international robot Week, Deajeon, Korea, 2005.