Proceedings of the 2008 IEEE
International Conference on Robotics and Biomimetics
Bangkok, Thailand, February 21 - 26, 2009

Impact of LEGO Sensors in Remote Controlled Robot

Kenneth Chiang-Yu Chin, Seyed Mohamed Buhari and Wee-Hong Ong
Department of Computer Science
Universiti Brunei Darussalam
Jalan Tungku Link, Gadong, BE1410, Negara Brunei Darussalam
kendalfbatman@hellokitty.com, mibuhari@gmail.com and owh@ieee.org

Abstract – The functionality of a robot greatly depends on its sensory capability. Adding sensors to a robot is one major strategy to extend the robot's function and intelligence. However, this strategy can only be effective if the performance of the sensors meet the requirements of the robot application. In particular, in many systems, it is crucial for the sensors to provide reliable updates on its value to the robot system. Accessing the sensors represents an overhead to the system and shall be taken into consideration when designing such robotic system. For a remote controlled robot, one significant overhead of accessing a sensor will be the communication delay. This paper investigates the impact of sensors in a remote controlled robot created from LEGO Mindstorms NXT kit. Communication activities and overhead when using LEGO sensors are reported. The results can serve a reference when designing a remote controlled robot using LEGO Mindstorms NXT kit.

Index Terms – Remote Controlled Robot, Network Controlled Robot, LEGO Mindstorms, NXT, Sensors

I. INTRODUCTION

Taking the advantage of fast advancing Information and Communication Technology (ICT) and the rapidly growing ICT infrastructure, network remote controlled or teleoperated robots have been designed to extend robotic applications in automation and terrestrial exploration where human intervention or supervision is required while human presence on site is not feasible. However, in these applications, the insertion of communication network in the system has complicated the design of such system, in particular when dealing with time delay and uncertainty in communication medium. This is particularly crucial when designing real-time system. For reliable performance, the communication infrastructure should consider end-to-end system-wide characteristics of sensing and actuation, constraints of power and computing resources, and interoperability [15].

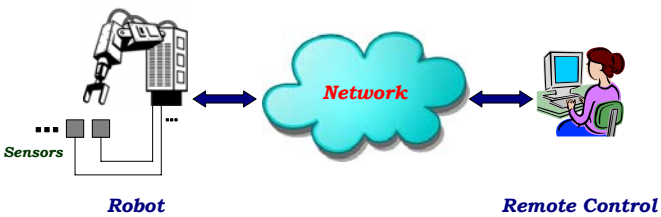

Figure 1. Network Remote Control Robotic System

In the field of education, LEGO Mindstorms, being versatile and yet inexpensive Robotic Invention System (RIS), has proven to be a useful kit for study of robotics as well as other science and technology subject matters. Previous works have recorded success in the use of LEGO Mindstorms in education [4], [11], [14], [18], [19]. Many interesting LEGO Mindstorms NXT (NXT is the current version of LEGO Mindstorms) projects have also been showcased [21] ever since the introduction of NXT in 2006. However, it has been found difficult to come across teleoperated NXT robot [23], not to mention the availability of documentation in this area.

This paper presents a remote controlled NXT robot equipped with sentry mode. In sentry mode, the performance of the robot greatly relies on the promptness of the sensing and control signals. Four sensors (touch, sound, light and ultrasonic) have been deployed in the robot. The impact of these sensors on the real-time performance is analysed by looking at the time delay the sensors, individually and in combination, are contributing to the system.

II. RELATED WORKS

The authors of [1] recorded the first teleoperated system to permit World Wide Web (WWW) users to remotely view and alter the real world. Few earlier demonstrations of teleoperated robots were reported in this paper. The paper focused on interface design, robot hardware, and system architecture. Due to the low bandwidth of WWW, the experiment in [1] did not allow real-time control and the remote operator did not receive real-time feedback from the robot.

Further experiments were reported in [2] whereby a robot located at Manchester, UK was remotely controlled and observed at Bremen, Germany. The experiments showed that low bandwidth communication links can be used for basic surveying, exploration, navigation and object manipulation tasks. However, significant delay of up to 12 seconds was found to made the control difficult. The authors of [2] have also reviewed on few earlier demonstrations of remote control robot via dedicated communication links.

In 1997, Tarn [3] demonstrated another Internet controlled robots at the International Conference on Robotics and Automation, where he pulled a joystick in Albuquerque that controlled the motions of a Puma robot in his laboratory more than 1,500 kilometres away. The experiment was conducted at a time when the traffic on the Internet was expected to be minimal to minimise the impact of transmission delay. Despite that, noticeable delay in robot movements was observed.

The impacts of the properties of the communication links had been clearly observed in the above experiments. Further works on remote control robots have seen emphasis on studying and analysing the impact of the properties of the communication link, in particular the transmission delay [7],[8],[10],[15]. NASA White Paper [15] highlighted the desired properties of communication architecture for robotic communication systems. In [15], it was recommended that the communication infrastructure should consider end-to-end system-wide characteristics of sensing and actuation, constraints of power and computing resource, and interoperability. Methods to overcome the impact of transmission delay have been proposed in various works [7],[9]. Works [5],[6],[13],[20] carried out in the area of Networked Control Systems (NCS) are highly relevant to the study of the impact of insertion of network in the control systems, which robot can be considered one.

All above works were carried out on industrial robots or custom made robots. Works carried out on LEGO Mindstorm NXT have primarily been focused on their educational values [4][11][14],[18],[19] and functionality [21]. There has been lack of study on the network performance of a LEGO Mindstorm NXT and its parts, in particular the different sensors. To date, the authors have only came across [17] that made a thorough analysis of the remote communication performance of NXT robots. The article provides useful analysis and pointers for the effective use of NXT Bluetooth communication.

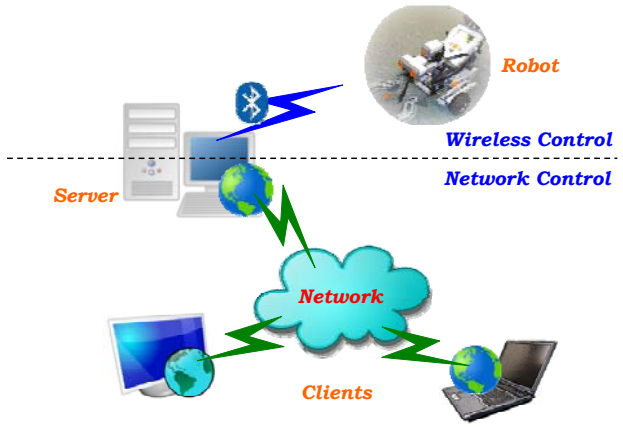

Figure 2. Experiment Setup

III. EXPERIMENTAL SETUP

A. *Overview*

The main aim of the experiment was to analyze the impact of the LEGO sensors on the real-time performance of the robot.

The experiment reported in this paper is a network controlled robot constructed using LEGO Mindstorms NXT kit. The robot was equipped with four typical LEGO sensors: Light, Sound, Touch and Ultrasonic. The robot was remotely controlled over Bluetooth communication link by a computer (Server) connected to a network. Another computer (Client) connected to the network (wired or wireless) could control the robot through the network. The Client may be located at a site geographically distant (different buildings, different countries) from the robot. The middleware (client-server application) developed for this system made use of techniques and protocols that would work over the Internet. However, the experiment was conducted over a Local Area Network.

B. *Communication Links*

Up to three types of communication links could be involved in the setup: Bluetooth wireless, wired LAN and/or wireless LAN. However, to focus on the impact of the sensors, the experiment was carried out on a LAN where network traffic can be minimized.

Bluetooth device at version 1.2, which has a maximum speed of 723kbps, was used at the Server. However, the LEGO Mindstorms NXT limits Bluetooth communication to 464kbps [17].

Figure 3. Communication Links for the Setup

C. *Hardware Specifications*

1. Two HP Compaq dc7100 running on Windows XP Service Pack 2 as Client and Server
2. USB Bluetooth dongle was installed on the Server
3. LEGO Mindstorms NXT Kit #8527 was used to construct the Robot

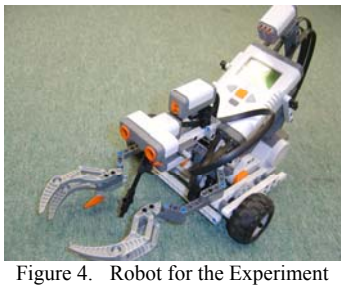

Figure 4. Robot for the Experiment

Figure 4 shows the Robot constructed for the

experiment. The processing unit of the Robot is an NXT brick, which reads four sensors and drives three servo motors. The four sensors are touch, light, sound and ultrasonic. The three servo motors moves two wheels and the gripper in front.

D. Software Specifications

1. The software (Client and Server) was developed in C# on .NET Framework 3 (minimal version required is 2.0.50727)
2. SharpDevelop 2.2 IDE was used to develop the software
3. NKH.MindSqualls [22] Library was used to implement Bluetooth communication between Server and the Robot
4. 1.04 NXT brick firmware
5. Wireshark was used to study the transmission time

E. Client and Server Applications

The Server has been designed to perform three main tasks. The first is to listen to any client trying to establish a connection. This is achieved by specifying the Port number to listen in the Server Interface shown in Figure 5, and pressing the "Start Server" button. The second is, once a client has established a connection with the Server, the Client can connect to the Robot. The Robot is connected to the Server through a *COM* port, using Bluetooth communication. The third task is to act as a middle man for the Client to remotely control/observe the Robot. The Server will receive instructions from the Client to control the Robot. In addition, the Server polls or receives messages from the Robot regarding its status information, such as the current battery level and sensors' values, and updates the information on the Client's interface. In addition, the Client and Server can communicate or chat with each other through the message panes.

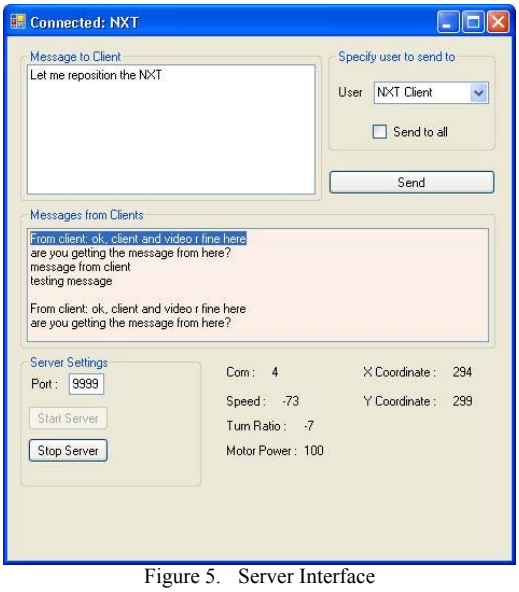

Figure 5. Server Interface

The Client application allows user to perform two main tasks. The first one is to allow the user to connect and communicate to the Server from a remote location. Referring to the Client Interface shown in Figure 6, to connect to the Server, user will enter the Server IP and the Port number that the Server is expected to be listening, in the "Connect To Server" pane and press the Connect button. Once the Client is connected to the Server, the user will enter the *COM* port number in the "Connect To NXT" pane, and pressing the Connect button.

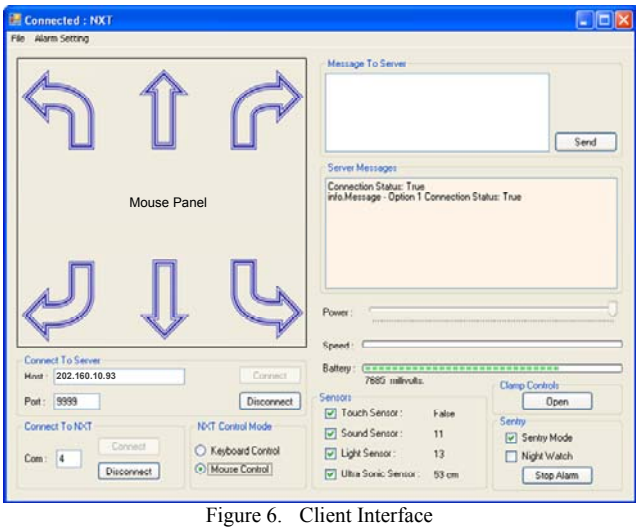

Figure 6. Client Interface

The *COM* port number must correspond to that used at the Server. The second task of the Client application is to provide a graphical user interface allowing the user to control the Robot from Client's current location. The user can choose to operate the Robot using mouse or keyboard. In Mouse Control mode, the user hovers the mouse over the Mouse Panel to move the Robot accordingly. Arrow keys are used to control the movement when operating in Mouse Control mode. The Client application also retrieves and displays feedback information from the Server and from the Robot. The Client Interface displays values of the sensors when they are enabled. The Battery level is being constantly updated by the Server.

The underlying communications between the Client, Server and Robot will be investigated in next section.

IV. RESULT AND ANALYSIS

A. The Experiment

Figure 7 shows the communications being investigated in this paper.

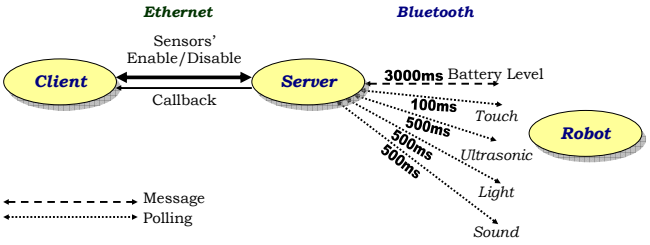

Figure 7. Robot Status Information Requests

Once connected, the Server will update the Client with current Battery Level every 3s via callback. The Server has been programmed to poll (using the Polling class in Mindsquall library) for the values of the Touch, Ultrasonic, Light and Sound sensors every 100ms, 500ms, 500ms and 500ms respectively. The polling will begin when the Client has enabled the respective sensor, and will stop when the Client disabled the sensor.

Note that all readings reported are the average of 15 measurements.

B. *Connection Setup*

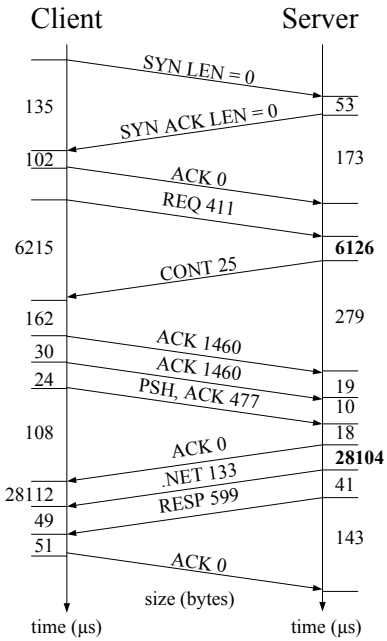

Figure 8. Connection Setup (measured using Wireshark)

Figure 8 shows the communications between Client-Server (and Robot) during the connection setup. The first three messages were the three-way handshake when the Client was connecting to the Server. The forth message was initiated when the Client requested for connection to the Robot. Upon this request, the Server took considerable time (6,126µs) to access the Mindsquall [22] library where its methods were being called. The Server then requested the Client to continue with the communication. The Client continued by replying with acknowledge message piggybacking with further data to establish connection with the Robot. The Server acknowledged immediately, while Mindsquall library methods were establishing the connection with the Robot through Bluetooth protocol. This activity was observed to take the 28,104µs, which account to about 81% of total time (34,797µs) to connect to the Robot. According to [24], Page 22, the NXT Brick will take around 30ms to reply a message as there is a time penalty of around 30ms within the Bluecore chip when switching from receive-mode to transmit-mode.

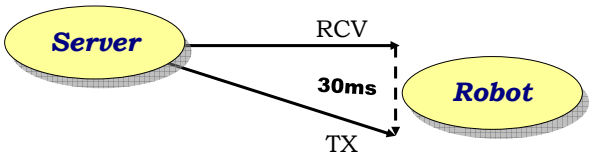

Figure 9. Message Exchange with NXT Brick

Theoretically speaking, communication of the data of (1460 + 1460 + 477 bytes) in the limited Bluetooth capacity of 464kbps would take 57ms. This value well exceeded the time recorded in Figure 8 indicating that only little portion of the data being transacted between the Client and Server are required to be sent to the Robot (via Bluetooth). The major load of the data is for the .NET framework communication, including the use of Mindsquall library.

C. *Battery Level Callback and Sensor Polling*

Figure 10 shows the communication activities when the Server was updating the Client with the latest battery level, every 3s. It is the duration for the .NET remoting call back initiated by the Server and does not include the Bluetooth communication between the Server and the Robot. The call back took less than a millisecond (898µs).

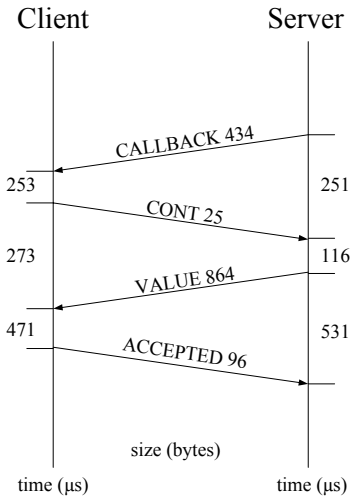

Figure 10. Battery Level Update (measured using Wireshark)

Figure 11 shows the communication activities when the Client was initiating the Ultrasonic sensor. It was

observed that Ultrasonic, Light, Sound and Touch sensors have similar performance. The initialization of the Ultrasonic sensor took approximately 7ms, with majority of the time (6,310μs) contributed from the processing at Server side (including execution of Mindsquall methods).

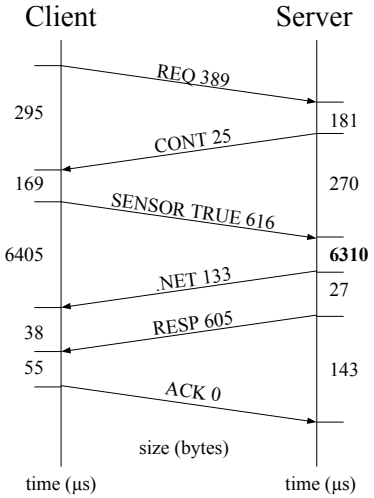

Figure 11. Ultrasonic Sensor Value (measured using Wireshark)

The above does not include the time taken for the Server to retrieve sensors' values from the NXT. Once enabled, the Server will poll the respective sensor for its value at the specified intervals mentioned in previous section. It was observed that the Server will send a data package of size 862 bytes at every polling interval. To measure the overall impact of accessing the Sensors, inline time printing was used to record the overall time starting from the Client enabling the Sensors until the time the Client has the updated value. Table 1 shows the times measured.

Time τ is expected to include processing times at Client τ_{pc} and Server τ_{ps} and communication time (Client-Server τ_{cs} and Server-Robot τ_{sr}).

$$\text{Time } \tau = \tau_{pc} + \tau_{ps} + \tau_{cs} + \tau_{sr}$$

Table 1. Time to Update Sensor Values

Sensor	Time τ (ms)
Touch	56.3
Sound	62.5
Light	46.9
Ultrasonic	52.0

From the results discussed in earlier paragraphs, τ_{cs} is expected around 7ms while τ_{sr} is expected to be around 30ms. There was significant contribution from the processing time within the .NET framework.

V. CONCLUSION

Looking at above result, the Bluetooth communication (30ms) is one major overhead for retrieving sensor values. The .NET framework communication was observed to transact larger data than that directly required by the Robot. Finally, the processing time, at both Server and Client, could have impact as much as the Bluetooth communication. However, this should depend on the specification the computer being used. Further works to investigate the hardware at low level (without third party library) would be necessary to provide clearer understanding of the sensors.

ACKNOWLEDGMENTS

The authors would like to acknowledge Ms Daphne Lai and Mr Liew Mun Kiat, from Universiti Brunei Darussalam, for their generous support and kind assistance in this experiment.

REFERENCES

[1] Ken Goldberg, Michael Mascha, Steve Gentner, Nick Rothenberg, Carl Sutter and Jeff Wiegley, "Desktop Teleoperation via the World Wide Web," Proc. of IEEE Int. Conf. on Robotics and Automation, pp.654-659, 1995

[2] Ulrich Nehmzow, Andreas Buhlmeier, Holger Durer and Manfred Nolte, "Remote Control of Mobile Robot via Internet," University of Manchester Technical Report Series UMCS-96-2-3, Feb 1996

[3] Tony Fitzpatrick, "Live Remote of a Robot via the Internet," IEEE Robot & Automation Magazine, pp.7-8, Sep 1999

[4] Philip Lau, Scott McNamara, Chris Rogers and Merredith Portsmore, "LEGO Robotics in Engineering," Proc. of American Society for Engineering Education Annual Conf. and Exhibition, Session 2620, 2001

[5] Wei Zhang, Michael S. Branicky and Stephen M. Phillips, "Stability of Networked Control Systems Stability of Networked Control Systems," IEEE Control Systems Magazine, pp.84-99, Feb 2001

[6] Q.P.Wang, D.L.Tan, Ning Xi and Y.C.Wang, "The Control Oriented QoS: Analysis and Prediction," Proc. of IEEE Int. Conf. on Robotics & Automation, Seoul, pp.1897-1902, May 2001

[7] C. D. Cheng, P. Vadakkepat, C. C. Ko, Ben M. Chen and X. Xiang, "Robot Motion Control and Image Reconstruction over Internet," International Conference on Computational Intelligence, Robotics and Autonomous Systems, Singapore, Nov 2001

[8] Lung Ngai, W.S. Newman and V. Liberatore, "An

Experiment in Internet-Based, Human-Assisted Robotics," Proc. of IEEE Int. Conf. on Robotics and Automation, vol.2, pp.2190-2195, 2002

[9] Per Cederberg, Magnus Olsson, Gunnar Bolmsjö, "Remote control of a standard ABB robot system in real time using the Robot Application Protocol (RAP)," Proc. of 33rd Int. Symp. on Robotics, Stockholm, paper no.113, Oct 2002

[10] Johan Potgieter, Glen Bright, Olaf Diegel and Sylvester Tlale, "Internet Control Of A Domestic Robot Using A Wireless Lan," Proc. Australasian Conf. on Robotics and Automation, Auckland, pp.212-215, Nov. 2002

[11] Mario A. Garcia1 and Holly Patterson-McNeill, "Learn How to Develop Software using the Toy LEGO Mindstorms," 32nd ASEE/IEEE Frontiers in Education Conf., Boston, S4D-7-10, Nov 2002

[12] Y. Zhang, K. Roufas, C. Eldershaw, M. Yim and D. Duff, "Sensor Computations in Modular Self Reconfigurable Robots," 8th Int. Symp. on Experimental Robotics (ISER 2002), Heidelberg, 2003

[13] M.S. Branicky, V. Liberatore and S.M. Phillips, "Networked Control System Co-Simulation for Co-Design," Proc. of American Control Conference, vol.4, pp.3341-3346, Jun 2003

[14] Frank Klassner and Scott D. Anderson, "LEGO MindStorms: Not Just for K-12 Anymore," IEEE Robotics & Automation Magazine, vol.10, issue.2, pp.12-18, Jun 2003

[15] Vincenzo Liberatore, Wyatt S. Newman, Kul Bhasin, Irene Bibyk and Brian P Robinson, "Robotic Communication Systems for Flexible, Sustainable, Affordable, and Autonomous Space Operations," a White Paper prepared for NASA, May 2004

[16] Masanori Sugisaka1 and Desa Hazry1, "User Interface in Web Based Communication for Internet Robot Control," ICCAS 2005, Gyeonggi-Do, Jun 2005

[17] Sivan Toledo, "Analysis of the NXT Bluetooth Communication Protocol," Sep 2006, http://www.tau.ac.il/~stoledo/lego/btperformance.html

[18] Ann-Marie Vollstedt, Michael Robinson and Eric Wang, "Using Robotics to Enhance Science, Technology, Engineering, and Mathematics Curricula," Proc. of American Society for Engineering Education Pacific Southwest Annual Conf., 2007

[19] Martha Isela Garduno Mota, "Work In Progress - Using Lego Mindstorms and Robolab as A Mean To Lowering Dropout and Failure Rate In Programming Course," 37th ASEE/IEEE Frontiers in Education Conf., Milwaukee, F4A-1-2, Oct 2007

[20] Network Control Systems and Internet Robotics, http://vorlon.case.edu/~vxl11/NetBots

[21] NXTLOG (NXT Projects), http://mindstorms.lego.com/NXTLOG

[22] MindSqualls, http://www.mindsqualls.net

[23] Remote Control of Lego Mindstorms Over the Internet, http://popbubble.com/Lego/WebBrick

[24] LEGO Mindstorms NXT Communication Protocol Document, Version 1.00, 2006