International Journal of Signal System Control and Engineering Application 1 (3): 180-187, 2008
ISSN: 1997-5422

Design and Implementation of Platform for Embedded Ethernet Control System

[1,2]Zhu Lingbo, [1]Dai Guanzhong, [2]Lin-Shi Xuefang, [1]Zhang Huixiang and [2]Retif Jean-marie
[1]School of Automation, Northwestern Polytechnical University, Xi'an, 710072, China
[2]Laboratory Ampere, INSA de Lyon, Villeurbanne, 69621, France

Abstract: A platform named EMECS based on embedded Ethernet control system is implemented. A PC running Fedora 6 (Linux 2.6) works as a central controller. A kit running Linux 2.4 based on Samsung2410A (ARM920 t core) works as a remote controller and a DC motor (ESCAP 28HSL18-219/204) made by Portescap Company works as a plant. Firstly, system modeling is presented by analyzing the characteristics of Ethernet and the plant, based on a method of delay measurement proposed. Secondly, implementation of the system including program and hardware is described in detail, respectively. And then delay is measured and control results of the system are tested in three cases with different network load as well. The platform is proved to have flexibility to run different control algorithms and extensibility to add nodes. The results of experiment demonstrate the validity of the system.

Key words: Ethernet control system, delays, embedded linux, linux device driver

INTRODUCTION

Traditional Control Systems are aiming at accounting for local systems. The distance between the plant and its controller is in close quarters. In these systems Disturbance Control System (DCS) and Long-Distance Control (LDC) are impossible. In order to fit the increasing complication and diversification of control objects and with the rapid development of computer technique, communication technique and sensor technique, Networked Control Systems (NCS) becomes one of the hot research fields in academia and industry (Shi-Ming *et al*., 2005). The advantages of network integration into control system displacing traditional computer control system's point to point lining are obvious, such as reducing cabling cost, increasing modularity and agility. However, some disadvantages are also brought, such as network-induced delays, disturbance and packet loss. These things can decrease the stability and robust of the system, even cause instability and emanation (Xiang-Hua *et al*., 2005; You-Ping *et al*., 2004).

For revealing the stability, robust and dynamical performance of NCS, researchers built various experimental platform based on different software and hardware. For example, Chow and Tipsuwan (2003) used a PC running RTLinux working as a central controller and a Siemens C-515C (8-bit) microcontroller board working as a remote controller with RS-232 connection. Xue-yuan and Guo-ping (2005) used ARM7TDMI (S3C4510B (2002) with running uCLinux to work as a remote controller and a PC with Matlab Simulink (RTW) to work as central controller. Xiaoya *et al*., (2005) proposed a kind of NCS simulation platform based on switched Ethernet though using two computers to simulate actual controller and plant by Matlab, which were connected through real network. Zhifei and Shuqing (2005) designed a simulation platform of NCS by using Matlab to build models of real systems. Communication between the controller and the plant is fulfilled with Winsock. However, the lacks of these practices are obvious. In Mo-Yuen Chow's case, for the lack of RS-232 protocol, the distance control based on this system is impossible. And there isn't OS (Operation System) under the microcontroller board, the flexibility and expansibility are restricted. In Nie Xue-yuan's case, the delays of the system are simulated delays. In Hu Xiaoya's case, the plant is a simulated plant by Matlab and VB, as well as Liu Zhifei's. All in all, these platforms are lack of flexibility and extensibility(Mo-Yuen Chow's) or lack of actually(the others) because of using simulated data.

Internet based on countless Ethernets with long-distance data communication is the biggest network of the world, so research on Ethernet Control System (ECS) is the significant research of NCS's study. Ethernet is a multi-purpose communication protocol that has become the data standard for home and office network (Felser and Sauter, 2004). Industrial Ethernet products begin to serve

Corresponding Author: Zhu Lingbo, School of Automation, Northwestern Polytechnical University, Xi'an, 710072, China

the communications requirements of industrial customers, replacing or supplementing legacy fieldbus such as Modbus or Profibus. Whereas, these buses are mutually exclusive and are held by different companies with expensive prices. Recently, using civil Ethernet to industrial field without exorbitant requirements is being a hot field. A typical way to create an ECS is using a central controller with powerful capability of processing and several remote controllers to output orders to the plant and sample the feedback values from the plant and then send them to the central controller with a protocol, such as TCP/IP (the most popular). However, the multiple access nature of Ethernet makes it impossible to guarantee a deterministic medium access time to individual stations (Kweon *et al*., 1999; Thomas and Gopinath, 1992).

In order to solve the lacks described above the second paragraph and considering the popularity of the platform, an actual platform based on Ethernet, S3C2410A (2004) (ARM920t core) kit, DC motor is designed in physical environment in this study. And this system is named as EMECS (Embedded Motor Ethernet Control System).

SYSTEM MODELING

By way of non-losing universality, the system modeling is created upon a typical ECS. As shown in Fig. 1, an ECS can be dived into three parts: The central controller, the remote system (including remote controller), Ethernet(Network). Where, $\tau^{ca} = [\tau_1^{ca}, \tau_2^{ca}, ..., \tau_k^{ca}]^T$ denote the delays between the central controller and the remote controller in k steps. And $\tau^{sc} = [\tau_1^{sc}, \tau_2^{sc}, ..., \tau_k^{sc}]^T$ denote the delays between remote controller and the central controller in feedback loop.

Central controller: The central controller can be assumed to have enough computing power and memory to satisfy the distributed remote controllers' requests and guarantee the advanced real-time control laws to all remote units, such as running a certain control algorithm rapidly and creating connection from one remote controller quickly. Each remote system processing has its own system dynamics that can be described by the state-space shown in Fig. 1.

Where, the state vector $x^p = [x_1^p, \cdots x_n^p]^T \in X^n$ are the state space,; $r^p = [r_1^p, \cdots r_m^p]^T \in R^m$ are the system parameters; $v^p = [v_1^p, \cdots v_k^p]^T \in V^k$ are the input space; $t \in R^+$ is the time parameter and $f_p \in R^n$ is the state transfer function.

$$\dot{x}^p = f_p(x^p, r^p, v^p, t) \tag{1}$$

However, in ECS, τ^{SC} and τ^{ca} can't be neglected, so the central controller receives the vectors $y^p = [y_1^p, \cdots y_n^p]^T \in y^n$ can be described as Eq. (2),

$$y^p = x^p(t - \tau^{sc}) \tag{2}$$

Remote controller: Remote Controller- The main job of each remote controller is maintaining the combination with the central controller. And each one can be assumed to have enough capability to receive the orders from central controller and send the sampled data to central controller in time. Because the delays between central controller and remote controller are existing, the orders vector $[v_1^p, \cdots v_n^p]^T \in V^n$ can be shown as Eq. (3),

$$v^p = u^p(t - \tau^{ca}) \tag{3}$$

Table 1: DC motor parameters

J	Inertia	10.4 kgm^{2}10^{-7}
L_a	Inductance	0.5mH
R_a	Resistance	5.8 Ω
K_c	Torque constant	22 mNm/A
K_i	Back-EMF constant	2.3V/1000t/min
η	Efficiency	0.79

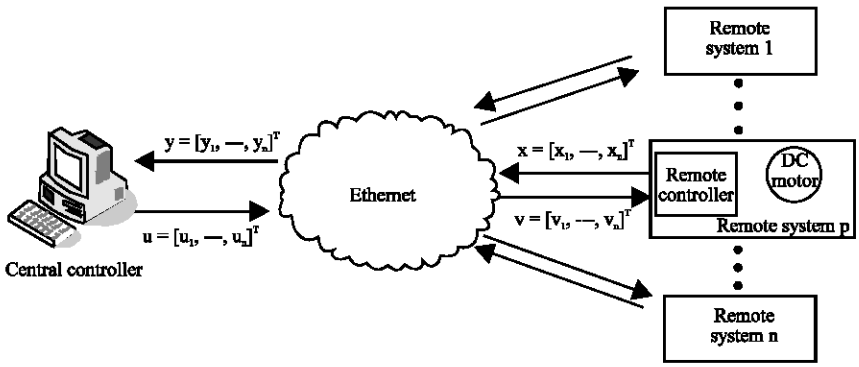

Fig. 1: Hierarchical structure of a typical ECS

Plant: The used motor is Escap (28HSL18-219/204) made by Portescap Company. A dynamo-tachymeter is connected to the motor for speed measurement. The parameters of the motor and dynamo-tachymeter are listed in Table 1.

As the electric time constant L/R = 8.62e - 5 is very small comparing to mechanical time constant J/f = 0.1337, the motor can be described by a first order model. Thus, the transfer function between the motor speed and the armature winding input voltage is:

$$\frac{\Omega(p)}{U(p)} = \frac{1/K_c}{1+\frac{RJ}{K_c^2\eta}s} \quad (4)$$

SYSTEM IMPLEMENTATION

The framework of system: In the interest of establishing EMECS, some hardwares are chosen, which are: A PC running Fedora 6 (Linux 2.6) works as a central controller; Eq. (2) a kit made by Samsung 2410A(ARM9 core, running Linux2.4) works as a remote controlle. The DC motor is ESCAP, 28HSL18-219/204. Besides, an interface card aiming to realize D/A conversion and voltage conversion is built and a server program of the central controller including control algorithm, a client program and module of Embedded Linux including AD driver, GPIO driver and interrupt functions on the kit are necessary.

Figure 2 describes the process of disposing data, the time sequence and the method to calculate the delay. Where, arrays b[k],a[k] ,Sb[k],Ca[k],Cb[k],Ss[k],Sv[k] are all the buffers, which be assumed to have enough space to storage the data. And Cs[k] (Xiang-hua *et al*., 2005) is a control array of k cycle, which can be read by module through using function 'ioctl'. The process can be depicted as,

Step 1: Build TCP/IP connection between Server and Client. Assumption, the name of the socket is sFD.

Step 2: When kT_e time arrives, Client produces a control array to tell module to interrupt for sampling data. And then module produces interrupt signal and samples voltage value from DC motor and put the value to b[k] (Assume current cycle to be No. k sampling cycle.)

Step 3: Client copies b[k] to Cb[k] from kernel layer to user layer. And then send(Cb[k], k and sFD). Get the time of successful sending and put the time into Ss[k].

Step 4: Server receives the Cb[k] by using recv (Sb[k], k and sFD). And then send Sb[k] to control algorithm. Control algorithm calculates the order by a[k] = Cal (a [0…k], p, g). Where p is the parameter of the system, g is the control target value.

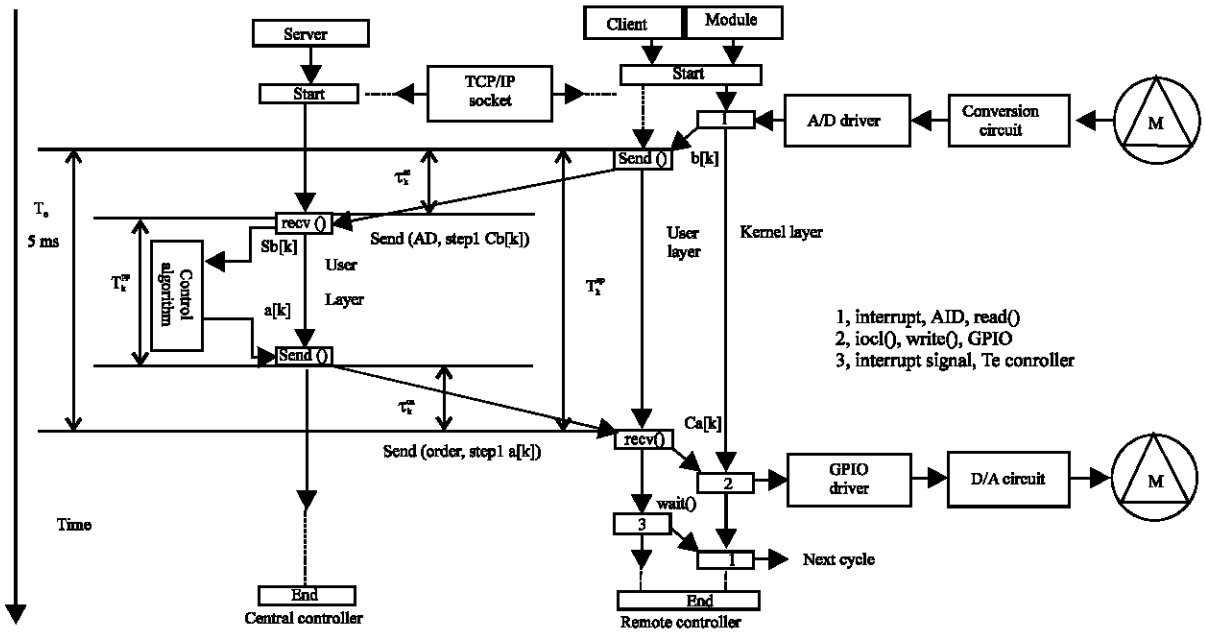

Fig. 2: Process of the system

Step 5: Server sends a[k] to Client by using send(a[k], k and sFD).

Step 6: Client receives a[k] by recv(Ca[k], k, and sFD). And then Client produces a control array to tell module to interrupt and output data.

Step 7: Client waits for $(k+1)T_e$ time arriving.

Taking kT_e time as an example, the method of calculating delay ($\tau = \tau^{sc} + \tau^{ca}$) is listed as:

Assumption, the time of ARM sending -T^{as}_k; The time of ARM receiving -T^{ar}_k; The time of PC receiving -T^{pr}_k; The time of PC sending -T^{ps}_k. From Fig. 2,

$$\begin{cases} T_k^{ap} = T_k^{ar} - T_k^{as} \\ T_k^{pp} = T_k^{ps} - T_k^{pr} \end{cases} \quad (5)$$

Then,

$$\tau_k = \tau_k^{sc} + \tau_k^{ca} = T_k^{ar} - T_k^{as} + T_k^{pr} - T_k^{ps} \quad (6)$$

Considering the OS of the PC and ARM9 are both Linux system, the function named 'gettimeofday' can be used to get the clock of the system (T^{as}_k, T^{ar}_k, T^{pr}_k, T^{ps}_k all can be gotten by this function.). The precision of the function is 1 us.

Server program

Control method: In order to be simple and to reduce the processing time in the central controller, in this study, PI control algorithm is chosen to compute the control to the remote system for steps. After letting the set-point as p, PI controller can be described as:

$$\begin{cases} u_k = r_0\varepsilon_k + r_1\varepsilon_{k-1} + u_{k-1} \\ \varepsilon_{k-1} = \varepsilon_k, u_{k-1} = u_k \\ \varepsilon_i = p - y_i \\ y_i = x_i(t - L_{sc}(i)), i = 1, 2, \cdots n \end{cases} \quad (7)$$

Where, r_0, r_1 are the parameters of the system. $x = [x_1, ..., x_k]^T$ are the state space. $L_{sc}(i) = \tau^{sc}_i$ is the delay between remote controller and central controller.

Implement with Fedora 6 (Linux2.6.18): A server program in PC working on user layer of OS should do these works: building connection between central controller and remote controller; receiving data from remote controller and sending orders to remote controller; implementing control algorithm. The program is built under Fedora 6 (Linux 2.6). The main parts and process of the program can be described in Fig. 3.

The connection is established by using socket to realize TCP/IP communication. In the program, functions including 'InitialSocket', 'Socket Process' and 'Control Process' achieve the necessary works.

Table 2: Functions descriptions

Name	Function
ConnectionMyfunction()	Building the connection
read()/write()	File I/O
promt_info()	Cycle control function

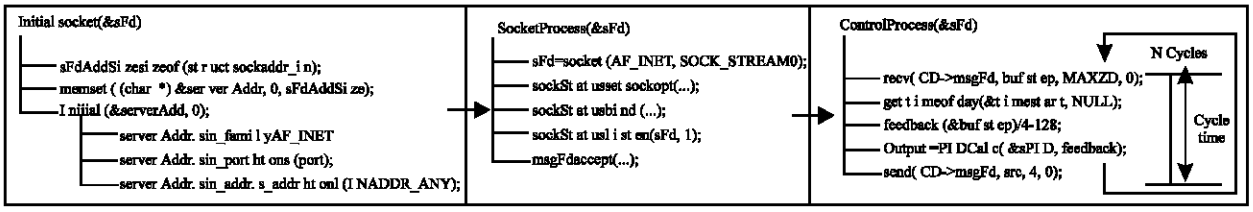

Fig. 3: Program in central controller

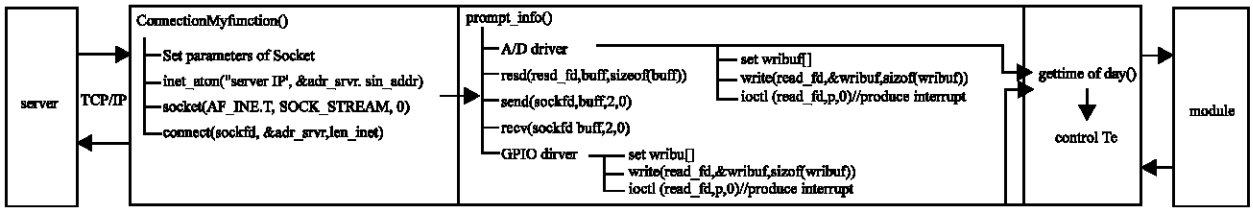

Fig. 4: The client program

Program in ARM9 (remote controller): In the ARM9 kit, a client program working on user layer of OS should do the works as: TCP/IP communication with central controller; control the sampling cycle time. And a Linux module working on kernel layer is needed to do these works: Driving A/D to sample the voltage from DC motor; Driving GPIO to output orders to DC motor; Realizing interrupts with parameters to sample and output data, respectively; Reading data from client program and copying data to client program.

Client program: The client program needs keeping communication with server program and the module. At the side of communication with server, socket with TCP/IP needs to be created. Functions' descriptions are listed in Table 2.

The process of function 'promt_info' can be written as:

Step 1: A/D sampling. This step includes setting array 'wribuf' (see next pharagh), write 'wribuf' to module and producing interrupt signal by using function 'ioctl'. Where, read_fd is the handle of module.

Step 2: Reading the A/D sampling value from module by using function 'read' function, which is realized in the module.

Step 3: Sending the value to server through socket.

Step 4: Receiving order from server through socket.

Step 5: Outputting order by GPIO. This step works like Step 1, but the function 'wribuf' are different.

The framework of the program can be depicted in Fig. 4.

Wribuf[] has 4 bytes to storage order(2 bytes) and control segment(2 bytes). The structure of Wribuf[] is shown in Fig. 5.

Embedded linux module: This module including file_operations like a typical Linux module, which defined some necessary functions, such as open, read, write, release, ioctl, etc S3C2410A (2004). However, according to the application of this implementation, GPIO Driver, A/D Driver and Interrupt functions should be made in the module. The framework of the module is shown in Fig. 6.

In AD Driver, firstly configurate the register named ADCCON though setting some bits of it, such as No.14:A/D converter prescaler enable;No.3~5:Analog input channel select;No:6~13: A/D converter prescaler value; and so on. And then start the AD and read the data from ADCDAT0. The data are the sampling data.

In GPIO Driver, GPIO_D3 and GPIO_C9~GPIO_C15 are used to output 8 bits digital signal. And GPIO_D0,GPIO_D1 and GPIO_G1 are used to control the interrupt (give interrupt signal or not). Every cycle's outputting data and sampling data are done in interrupt functions (Corbet, 2005).

Function 'ioctl' is used to produce interrupt signal. Function 'read' and 'write' are used to interact with client program.

After building the program, AD precision test is done. In testing AD, 4 groups are taken every group has 9 values. The results are shown in Fig. 7. The average error of AD sampling circuit is 1.91%.

Interface card: In the implementation, there are two circuits, which are needed. One of them is D/A circuit. When the digital signals are outputting from GPIO of the kit, the circuit changes digital signals to analog signals. And the other one is voltage conversion circuit, which changes -5V~+5V to 0~3V. This conversion can reach the A/D sampling area of the kit. In order to simple, in the implementation these two circuits are integrated in one card. The D/A chip is AD7523JN (Corbet, 2005).

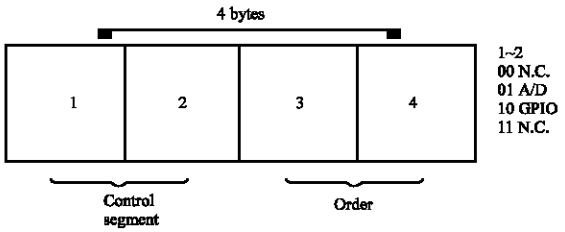

Fig. 5: Format of wribuf (Chow and Tipsuwan, 2005)

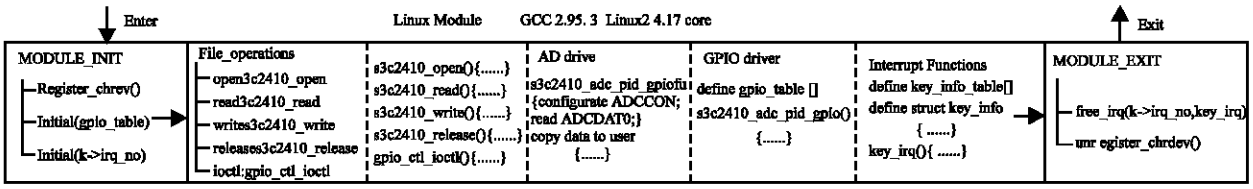

Fig. 6: Linux module of ARM9

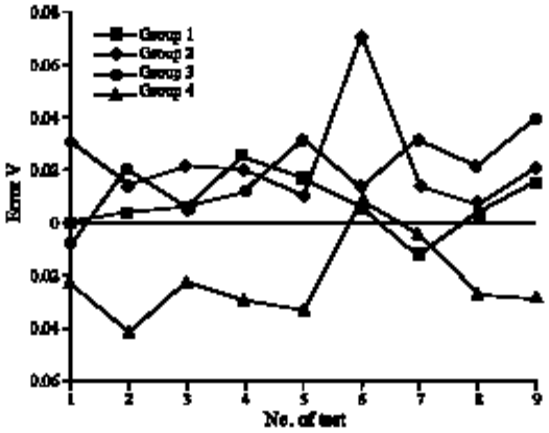

Fig. 7: Results of A/D

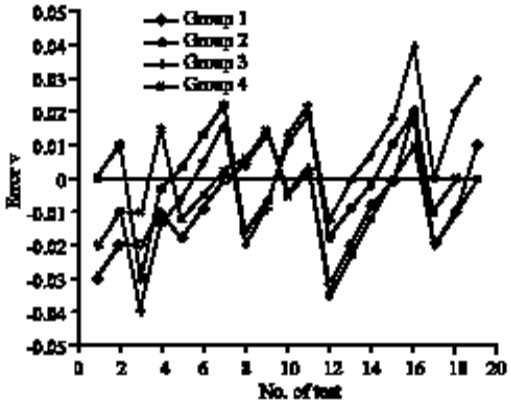

Fig. 8: Results of D/A circuit

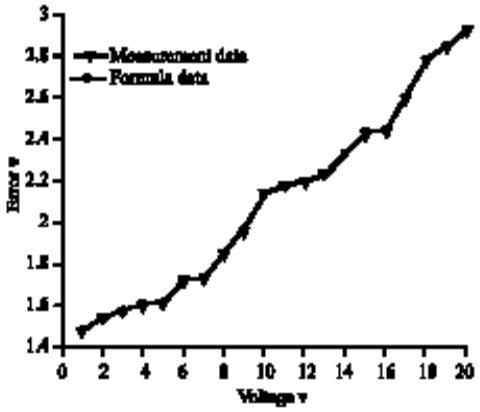

Fig. 9: Results of conversion circuit

TL081CN and TL084CN are used in the card. After designing the circuit diagram and building the card, a test for the precision of the circuit is taken. Four groups with 19 value of every group are done. The errors are shown in Fig. 8. The average error of the D/A circuit is 0.72%. In conversion circuit, 20 values are chosen to test the errors. The results are depicted in Fig. 9 and the average error is 0.41%.

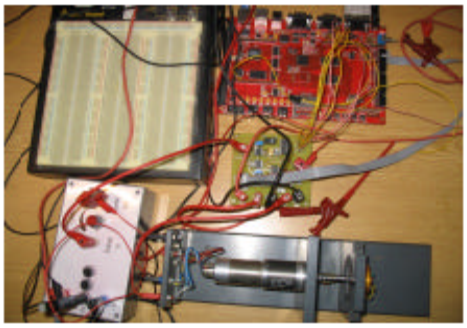

Fig. 10: Remote controller and plant

EXPERIMENT

When the programs and the interface card are prepared, the experiments are done based on a normal office Ethernet. Three ways of connections are chosen to show the capability of the system. Case 1 is to connect the central controller and the remote controller directly. In this mode, the communication just between central controller and remote controller, so in other words, which is zero load. Case 2, the central controller connects to a Switch (3COM Baseline Switch 2024) and the remote controller is also connecting to the same Switch. At the same time, the other ports of the Switch are shared by the users of the office. This type is a typical office Ethernet and the status of the network is stochastic(related to every user of the Ethernet). Case 3, based on the Case2's connection, heavy loads are added to central controller and remote controller. In every case, delays are measured by the way of described in Part 3 and the results are also shown in Fig. 9. Initial voltage is 0 v. Set-point is 0.7815v. The parameters of PI controller are: $r_0 = 0.9775$, $r_1 = -0.7183$. Sampling cycle time is 5 ms and 400 steps are done. Figure 10 shows the remote controller, the interface card and the plant of EMECS.

From Fig. 11 a, as the Load of the Ethernet increasing from zero to heavy, the delays become bigger brokenly. Packet loss and huge delays appear irregularly. And the dynamical performance of the system becomes worse. In ascending part, from case 1 to case 3, the dithering of responses is sharper and sharper Fig. 11b. In the tranquilization part, shakes are becoming bigger and bigger from case 1 to case 3. And the time restriction is broken with the delays increases, for example, the used time of 400 steps is 2 s, but in case 3, which is around 2.06 s. The experiments reveal that the network-induced delays, disturbance and packet loss can decrease the stability and robust of the system, even cause instability

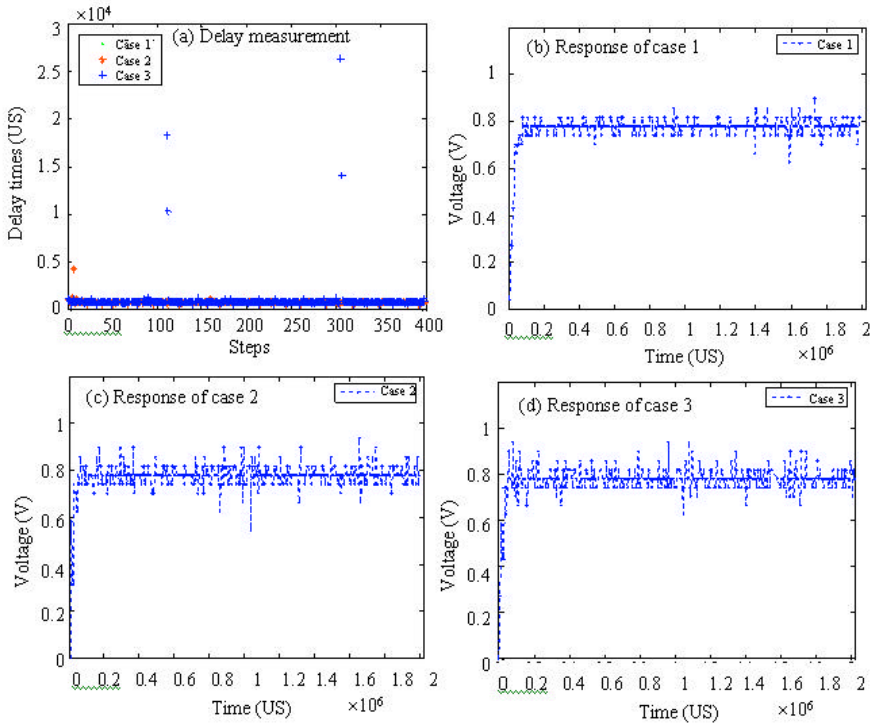

Fig. 11: Results

and emanation. There are several ways can part-solve the problem. One of them is to use new control algorithm, which can tolerant longer delay. The other way is to modify the protocol to guarantee the delay less than T_e. The third is to develop a scheme to guarantee an order can be given in every cycle by predicting orders. In this method the algorithm of estimating the order is the key problem.

CONCLUSION

In order to reveal the influence of network-induced delays, disturbance and packet loss on the stability, robust and dynamical performance, a platform named EMECS is established based on embedded Ethernet control system applying to a DC motor in this srudy. The research of building EMECS includes program's development and hardware's development. And then three cases (zero load, stochastic load, heavy load) are tested based on EMECS. The application described in this study isn't the only one of EMECS. In fact, EMECS can run different control algorithms easily. It is also easy to implement other DCS or LDS control systems. Future work can be summed up as: New control algorithm in this application to improve the capability of the system, new communication scheme between central controller and remote controller or new communication protocol to improve or stop the packet loss and huge delays. This platform is a prerequisite experiment circumstance for testing new control algorithm or new scheme to improve the capability of the system. Comparing with the platforms described in paragraph 2 of part 1, because EMECS is realized on actual hardware, EMECS expresses the circumstances of network more factually, flexibility and extensibility.

REFERENCES

Chow, M.Y. and Y. Tipsuwan, 2003. Gain Adaptation of Networked DC motor Controller Based on QoS Variations[J], IEEE, Trans. Industrial Electronics, Vol. 50.

Corbet, J., Alessandro, Rubini and G. Kroah-Hartman, 2005. Linux Device Drivers. 3rd Edn. [M], O'REILLY, pp: 42-72.

Felser, M. and T. Sauter, 2004. Standardization of Industrial Ethernet-the Nest Battlefield?,[C] 0-7803-8734-1/04/, 2004, IEEE., pp: 413-421.

Kweon, S., K.G. Shin and Q. Zheng, 1999. Statistical Real-Time Communication over Ethernet for Manufacturing Automation Systems [C], Proceedings of the Fifth IEEE Real-Time Technology and Application Symposium.

Shi-Ming, Y., Y. Ma-Ying and Y. Li, 2005. Predictive Compensation for Stochastic Time Delay in Network Control Systems, Acta Automatica Sinica J., 31 (2): 231-238

Thomas E. Bihari and Prabha Gopinath, 1992. Object-Oriented Real-Time Systems:Concepts an Examples[C], 0018-9162/92/1200-0025S03.00,1992 IEEE., pp: 25-32.

USER'S MANUAL S3C2410A-200MHz & 266MHz 32-Bit RISC Microprocessor Revision 1.0[EB/OL], 2004 Samsung Electronics.

USER'S MANUAL S3C4510B[EB/OL], 2002 Samsung Electronics.

Xiang-hua, M., X. Jian-ying and W. Zhen, 2005. Research on Networked Control Systems [J], J. Shanghai Jiaotong University (Sci.), 10 (1): 25-29.

Xiaoya, H., Z. Desen and W. Bingwen, 2005. Simulation platform for networked control system based on switched Ethernet[J], J.Huazhong Univ. Sci. and Tech. (Nature Sci. Edn.), 33 (10): 89-91.

Xue-yuan, N. and L. Guo-ping, 2005. Realization pf Embedded Networked Control Simulation Based on Simulink[J]. J. Syst. Simulation, 17 (7): 1613-1620.

You-ping, C., C. Bing and X. Jing-ming, 2004. Scientific problems and application prospects of the networked control systems. Control and Decision J., 9 (9): 961-966

Zhifei, L. and W. Shuqing, 2005. Design of a Simulation Platform of Networked Control Systems [J]. Chinese J. Scientific Instrument, pp: 597-600.