Multiagent Based Greenhouse Telecontrol System as a Tool for Distance Experimentation

D.Stipanicev*, M.Stula* and Lj.Bodrozic*
* Department for Modeling and Intelligent Computer Systems
Faculty of Electrical Engineering, Machine Engineering and Naval Architecture
UNIVERSITY OF SPLIT
darko.stipanicev@fesb.hr , maja.stula@fesb.hr , ljiljana.bodrozic@fesb.hr

***Abstract*— A telecontrol system controls and monitors equipment in remote locations. The paper describes the development of multiagent system for telecontrol of dislocated laboratory greenhouse model, which also includes video telepresence and teleoperation of the camera's pan and tilt unit. The system was developed as a part of distance control engineering education system and its aim was remote experimentation.**

The system was conceived as a multiagent system, because agent technology is the emerging technology for software cooperation, especially interesting for the net applications. The paper describes the developed multiagent system in details, including the formalization of used agents. Common ontology which defines vocabulary for exchanging queries and propositions among agents was discussed, too. Ontology is the explicit specification of knowledge conceptualization and very important in any agent based system.

The experimental system was realized using JATLITE (Java Agent Template) and KAPI (KQML Application Programmer's Interface). Five different kind of agents was used and described – the user interface agent, the greenhouse agent, the greenhouse control executer agent, the video agent and the video control executer agent. These agents were responsible for all tasks, from the user – system communication to the telecontrol, video telepresence and teleoperation with camera's pan & tilt unit.

I. INTRODUCTION

In control engineering education concepts taught in classrooms has to be complemented in laboratory by experimentation. Teaching dynamic phenomena is often much more easily using experimentation then by written material. Today's information and communication technologies, particularly Internet based technologies, have opened a new opportunities in control teaching by experimentation. One of these possibilities is distant or remote experimentation.

Technology Enhanced Learning (TeL) is trying for more than decade to provide tools and infrastructure to education and training disciplines. A distant experimentation using IC technologies is one of tasks where a lot of Research and Technology Development (R&TD) projects has been done all around the world. For example some of successful general EU R&TD projects in this area were:

- **Lab@Future** [1] – learning platform that uses novel information and communication technologies to support and expand laboratory teaching practice,
- **CoLab** [2] - a new type of learning environment which was created to help learners to develop flexible knowledge in science domains, skills to collect and synthesize information and to collaborate with others,
- **MOBIlearn** [3] - a worldwide European-led research and development project exploring context-sensitive approaches to informal, problem-based and workplace learning by using key advances in mobile technologies.

There were also a lot of projects dedicated particularly to remote experimentation in control theory. One quite successful project was "**Virtual Laboratory Project"** done at FernUniversität Hagen with University of Bochum and University of Dortmund as partners [4, 5, 6]. The project has started in 1988 and until today, a lot of control experiments were realized. The other one is on-line from 1995. It is a University of Tennessee at Chattanooga project called **"Control laboratory on – line"** [7]. The main task was the same as previously, control engineering experimentation from remote location using standard Web browser on the user side. On the server side the German system was based on commercial Matlab/Symulink® WinCon® Server, and the American system was based on commercial LabView® server components.

This paper describes the different approach to the realization of distant control experimentation system. Our main task was to realize the temperature and humidity monitoring and control experiment with the possibility of the video telepresence (video monitoring) of the experimental area. One of our requests was to use existing laboratory greenhouse model and existing video system. The laboratory greenhouse and camera pan/tilt unit was previously used in teaching control principles, so we had a lot of developed software for control algorithms, sensors monitoring, power control and pan/tilt unit operations. The idea was to integrate the existing software in our distant experimentation system. Because of that we have decided to use the software agents architecture as a platform for system realization. A multiagent system was designed conceived of five specific software agents. Agents were responsible for all operations, from user – system communication to telecontrol and video monitoring operations. As the agent communication languages (ACL), the standard Knowledge Query and Manipulation Language (KQML) was used. In the next

sections the system overview and more details about agent structure and agent ontologies are described.

II. THE SYSTEM OVERVIEW

Three main tasks were defined:

- to monitor and collect greenhouse process parameters – the temperature and the humidity,
- to control the temperature and humidity in greenhouse using the electric heater, vetting system and ventilators mounted on both sides of greenhouse model,
- to have the video telepresence to experimental area with possibility of camera control (pan and tilt movements).

Fig. 1 shows the experimental setup.

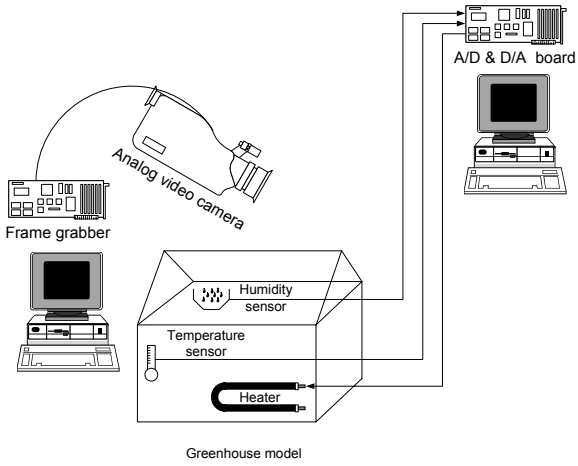

Figure 1. The experimental setup: laboratory greenhouse, PC with ADDA boards, PC with frame grabber and analog video camera

Fig. 2 shows the experimental setup photo.

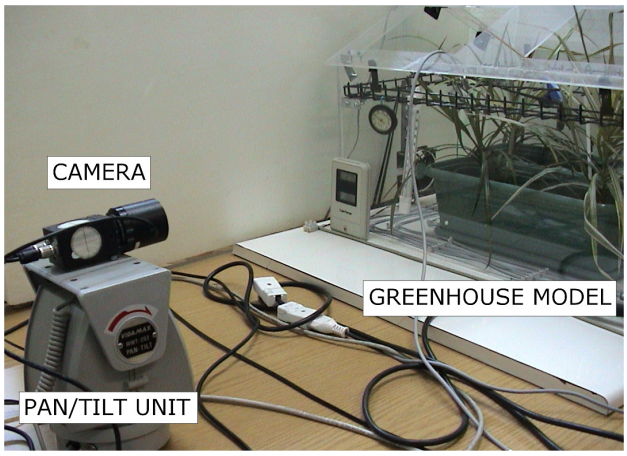

Figure 2. Laboratory greenhouse model and analog video camera mounted on pan & tilt unit

The schematic drawing of the greenhouse is shown on the Figure 3. The laboratory greenhouse was equipped with few temperature sensors for measuring temperature in air (T_0), on the ground level and inside the ground (T_{11} – T_{23}) and with one humidity sensor (H_0). In our experiments only the temperature in the air T_0 and humidity H_0 was used.

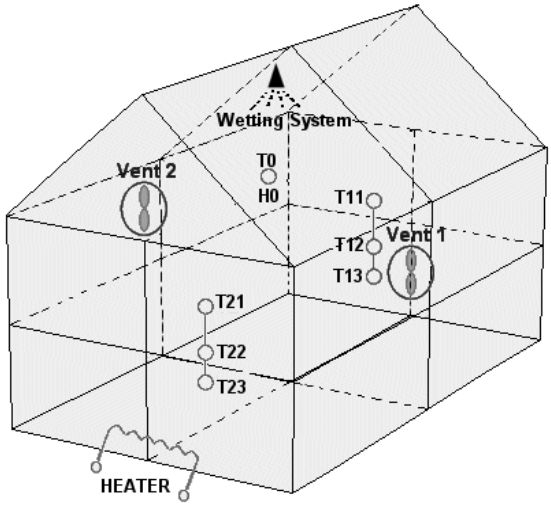

Figure 3. Experimental laboratory greenhouse model

The laboratory greenhouse model was also equipped with electric heater, controlled by triac power regulator and controlled wetting system. On both sides of laboratory greenhouse two ventilators were mounted, one for taking the air from the outside space to the greenhouse, and the other one, for taking the air from the greenhouse to the outside space. The ventilators could be also controlled, and they have been used as a disturbance to temperature control system, but also as a overheat protection system. The different initial conditions could be realized by four openings on the greenhouse roof. For temperature and humidity variables digitalization and heater, wetting system and ventilators control, two CIO-DAS08/Jr-Ao ADDA cards were used. The ADDA cards were installed in Pentium PC. Fig. 4 shows the connection block diagram.

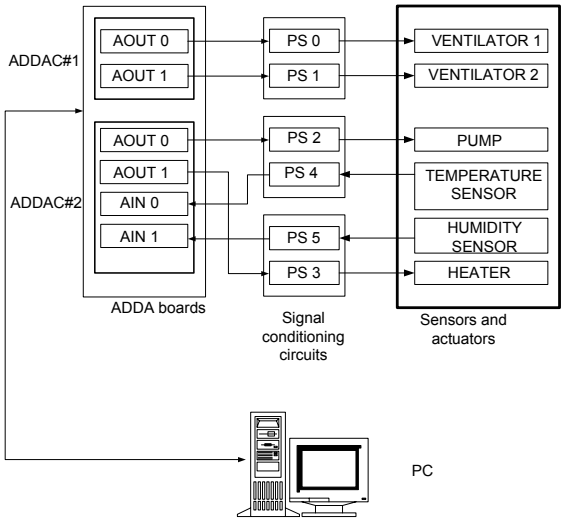

Figure 4. Block diagram of laboratory greenhouse model monitoring and control part

The video camera system, used in our experiments, was not the real time video monitoring system like network camera system. Our intention was to emphasize the

possibilities of multiagent architecture, so our video monitoring was based on analog PAL video camera, frame grabber for image digitalization and analog AC drive pan/tilt unit controlled using dedicated hardware through PC parallel port (see Figure 2). The frame grabber and pan/tilt control unit were physically located in the second PC (also "ancient" Pentium PC). The image was stored to the computer disk in jpg form every x seconds, and the user could see only the last stored image, but he could also change the x value (x was typically set to 10 seconds) and to control pan and tilt unit movements.

III. REALISATION OF THE MULTIAGENT ARCHITECTURE

For the realization of the multiagent architecture the **JATLITE** (*Java Agent Template Lite*) and **KAPI** (*KQML Application Programmer's Interface*) have been used [8]. **KQML** *(Knowledge Query and Manipulation Language)* was the Agent Communication Language (ACL) [9]. Multiagent system was based on five agent types designed to perform different tasks [8]. They were:

- user interface agent,
- greenhouse agent,
- greenhouse control executer agent
- video agent, and
- video control executer agent.

The multiagent structure is shown on the Figure 5.

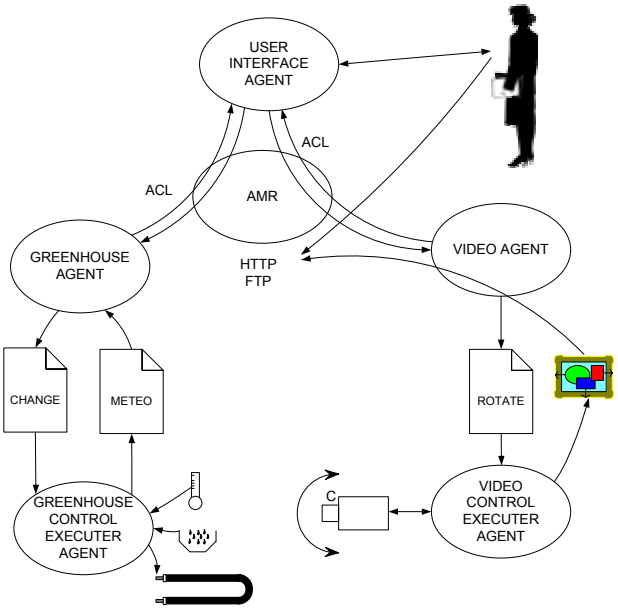

Figure 5. The multiagent structure

The **user interface agent** was used for translation of user demands to the agent communication language. This agent was made as a Java applet, so the user interface could be the standard web browser.

The task of the **greenhouse agent** was to get the user demands related to the monitoring and control of the greenhouse temperature and humidity. Agent was made as a C process running on the computer that was attached to the greenhouse sensors and actuators through ADDA card. The agent had self protecting features. If the user wanted to increase the temperature in the greenhouse higher that normal values (for example more than 50^0C), the agent would discard that demand. Also if the temperature in the greenhouse exceeds the defined upper limit, the greenhouse agent switches the ventilators 1 and /or 2 to protect the plants in greenhouse.

The third agent was the **greenhouse control executor agent**. This agent was the program previously developed for reading sensors and sending control values to actuators through ADDA card. It was running on the same computer as the greenhouse agent. These two agents did not communicate via ACL. They exchanged data trough the two data files: meteo.dat and change.dat. The first one was used to store the actual process parameters values, but also their maximal and minimal recorded values, and the second one for communication with greenhouse actuators.

Although control executor agent can't "*speak"* ACL, it could be considered as an agent according to the widely accepted definition by Franklin and Graesser [11]:

"An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and as to effect what it senses in the future."

On the video side, we had the **video agent**. Its task was to get the user demands related to the video monitoring. Agent was also made as a C process running on the second computer equipped with frame grabber.

The fifth agent, called the **video control executor agent** was also the existing program responsible for storing images from camera to the disk and sending control signals to pan/tilt unit. It was running on the same computer where video agent was active. The communication between these two agents was also not based on ACL. They exchanged data through two data files (rotate.dat and image.jpg). The first file contained data for pan / tilt camera movement, and the second one was a jpg image.

All messages exchange among agents was done through **Agent Message Router** (AMR) (see Figure 5). Using this principle the agents didn't need to maintain the exact knowledge about other agents position in the network (their IP address). They could communicate with other agents using their names only.

The user interface was the standard web page, containing user interface agent as an applet. Through the user interface agent, the user sent its requests to the multiagent system and got the response form the system. For example, if the user has requested the temperature value, the user interface agent sent the following message to the greenhouse agent:

```
(ask   :sender user1
       :receiver greenhouse_agent
       :language KQML
       :ontology temperature :content  (value))
```

The greenhouse agent read the current temperature from the file meteo.dat. This value has been written to meteo.dat file by the greenhouse control executer agent. After that the greenhouse agent sent the response to the user interface agent:

```
(tell  :sender greenhouse_agent
       :receiver user1
       :language KQML
       :ontology temperature :content (21.3))
```

The user interface agent, programmed as an applet, is shown on Figure 6.

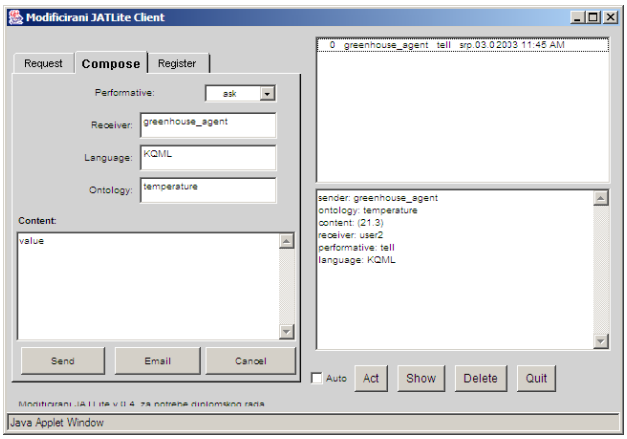

Figure 6. The user interface agent with dialogue example between the user and the greenhouse agent

On the left side there are user input forms. Figure 8 shows them enlarged. On the right side there is a communication dialogue shown in details on the Figure 9.

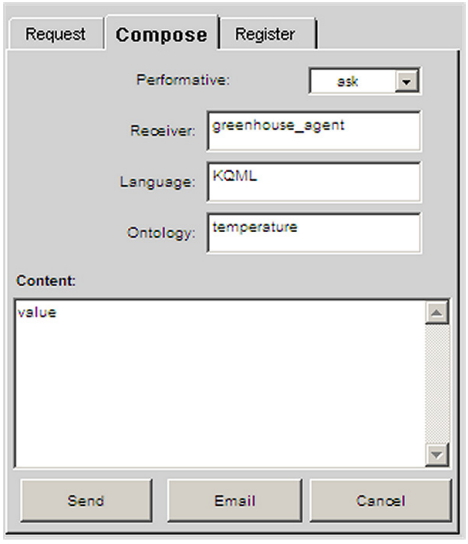

Figure 7. The left side of user interface agent with user input forms (the example is ask – temperature case)

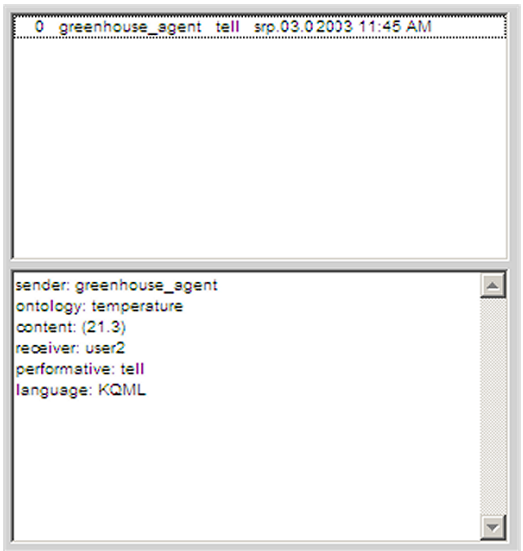

Figure 8. The right side of user interface agent with greenhouse agent response and complete dialogue

On the left side there is a button **"E-mail"** for sending ask message using Simple Mail Transfer Protocol (SMTP), and a button **"Send"** for sending message using HyperText Transfer protocol (HTTP).

If the user wanted the video image of the experimentation site, the user interface agent sent the following message to the video agent:

(ask *:sender user1*
:receiver video_ agent
:language KQML
:ontology image :content (take))

The video agent sent the current image file image.jpg from the disk to the web server using FTP protocol and sent the following response to the user interface agent:

(tell *:sender video_agent*
:receiver user1
:language KQML
:ontology image :content (Image is sent))

This image was captured and stored to disk by the video control executer agent. By clicking on the button **"Show"**, located on the right side of the user interface agent, the user could open this image in Web browser. Figure 9 shows an example.

Figure 9. An example of the image captured by the video control executor agent and transferred to the Web server by video agent

If the user wanted to rotate the camera to the left (right, up or down) the user interface agent sent the following message to the video agent:

(ask *:sender user1*
:receiver video_agent
:language KQML
:ontology image :content (left))

The video agent wrote this information to the file rotate.dat and sent the response to the user interface agent:

(tell *:sender video_agent*
:receiver user1
:language KQML
:ontology image :content (Camera in movement))

Video control executer agent periodically read the file and when the new request has been made, it sent the appropriate command to video camera pan/tilt unit.

Figure 10 shows the sequential diagram with messages exchanged among agents when the user requests the temperature value, image from the camera and demand for camera rotation.

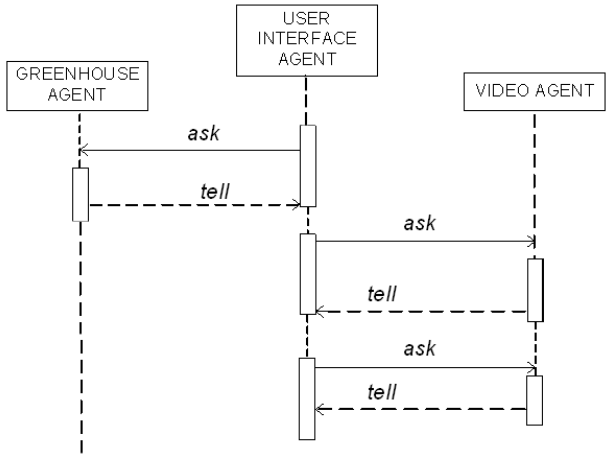

Figure 10. Sequential diagram of agents communication

Similar messages are exchanged among agents in other cases too.

IV. DESIGNING ONTOLOGIES

The agent definition is not the only one task in designing multiagent system. The other quite important one is to define **ontologies** and message content that agents would use. The ontology could be defined as a description of the entities, concepts and relationships existing in the real world we are dealing with [12]. Common ontology defines vocabulary for exchanging queries and propositions among agents. In our example the agents could speak seven main ontologies:

temperature, humidity, heater, vetting, vent1, vent2 and ***image***

Each ontology has its own content. The content of ontologies ***temperature*** and ***humidity*** is:

value, min, max and ***number***.

Content elements *'value'*, *'min'* and *'max'* are connected with ask messages and used by the user agent to get the actual value, the minimum recorded value and the maximum recorded value of the process variable. The content element *'number'* is connected with tell message and represents the returned numerical value. An example is given previously, but let us repeat only the ontology part of ask and corresponding tell message. The ask message was:

:ontology temperature :content (value)

and the corresponding tell message was:

:ontology temperature :content (21.3)

This means that the actual temperature value in greenhouse is 21.3^{o}C .

The content of ontologies ***heater, vetting, vent1*** and ***vent2*** is:

value, number and ***change(number)***

The content element *'value'* is similar to previously explain ontologies. It is used in ask message to get the actual condition connected with heater, vetting system or ventilators. The tell message returns the *'number'*, for example the percentage of applied power.

The element *'change(number)'* is the new one. It is connected with ask message and expresses the demand for changing the status of any actuator. For example the ontology part of ask message could be:

:ontology heater :content (change(80))

This expresses the user demand to set the heater level on 80% of maximal power.

The content of the ontology ***image*** is:

left, right, up, down, center, increment(number), increment_set(number), increment_value, 'Camera in movement', take, 'Image is send', delay(number), delay_set(number), delay_value, number

Content elements *'left', 'right', 'up', 'down', 'center', 'increment(number)',* and *'Camera movement in progress'* are connected with camera pan and tilt movements. The video control executer agent had its predefined increment (5^{o}), but it could be changed by content element *'increment(number)'*. The corresponding tell message is *'increment_set(number)'* where *(number)* is actual increment value. The user was also able to get this value using the content element *'increment_value'*. The meaning of elements *'left', 'right', 'up', 'down', 'center'* are self explaining and their corresponding tell message is *'Camera movement in progress'*.

Content elements *'take'* and *'Image is send'* are appropriate ask and tell messages for getting jpg image taken and stored to disk by the video control executer agent every x seconds. The value x could be defined by the element '*delay(number)'* or get by '*delay_value'*.

The multiagent architecture was quite simple, but powerful enough for experimental realization of distant monitoring and control of laboratory greenhouse and video monitoring system.

V. PROCESS CONTROL EXPERIMENTATION

The developed system was used in experimental laboratory exercise for course "Process modeling and control". It was used for practicing and learning process model identification. The student task was to identify the model of the greenhouse, having the air temperature in the greenhouse as the output variable and heater power as the input variable. Different initial condition could be realized by opening and closing the openings on the greenhouse model roof. Also the disturbances could be introduced switching-on and off ventilators on both sides of the greenhouse or switching-on the vetting system for a certain period of time.

The student communication with system was done through user interface agent. The procedure was as follows:

a) First the power level of the heater had to be set to the certain percentage of the maximal heater power.

b) After that the temperature value was read every x seconds. The student task was to makes notes about these values but also to take care about exact sampling interval.

c) The final task was to make process model identification using different identification methods.

Heating and cooling examples are shown on Figures 11 and 12. Figure 11 shows the situation where all openings on greenhouse roof were closed and the heater was set to 100% power. Figure 12 shows the situation when all openings were closed, too, heater was set to 0%, and Vent1 was set to 100%. The sampling interval in both cases was 60 seconds.

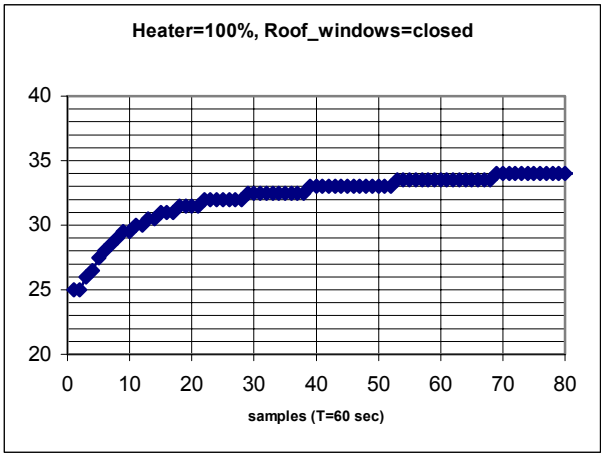

Figure 11. Data for heating example with closed roof windows

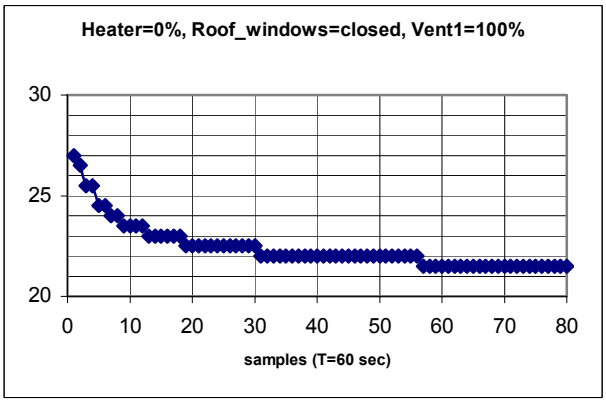

Figure 12. Data for cooling example with ventilator 1 switched on

The greenhouse was modeled as a first order thermal system, so the student task was to calculate the process time constant and the process transfer function gain. Also, she or he had to compare these values with values calculated using process impulse transfer function, estimated by Least Square Estimation (LSE) algorithm. The LSE theory was written as a chapter of e-textbook about digital control [13]. As an example of more advanced identification methods let us mention that the system was also used in one diploma work about process identification based on linear regression [14].

VI. CONCLUSION

The paper describes the development and implementation of monitoring and control system suitable for distant experimentation with laboratory greenhouse model using Internet infrastructure. The system was able to interact in both directions with the greenhouse, to read the process parameter values (temperature and humidity), but also to control the process actuators (heater, vetting, ventilators). Also the video telepresence to the experimental site was realized using video camera mounted on pan & tilt unit. Our approach was specific, because it was based on software agents and maximal use of existing software for data acquisition, process control, video image capturing and control of video camera pan and tilt unit. For agent communication, standard KQML *(Knowledge Query and Manipulation Language)* was used. The future work is to incorporate more sophisticated net cameras and to use the embedded Web servers instead of PC as a unit which will gather and route data from local sensors and actuators to the Internet network.

ACKNOWLEDGMENT

This work was supported by the Ministry of Science and Technology of Republic Croatia under Grant 0023008 "Intelligent Agents in Modeling and Control of Complex Systems".

REFERENCES

[1] Lab@future – EU project about learning platform that uses novel IC Technologies - http://www.labfuture.net

[2] CoLab – Colaborative Laboratory http://colab.edte.utwente.nl

[3] MobiLearn - problem-based and workplace learning by using key advances in mobile technologies http://www.mobilearn.org

[4] Reale Systeme im virtuellen Labor, FernUniversität Hagen, http://prt.fernuni-hagen.de/rsvl/

[5] Chr. Schmid, Virtual Control Laboratories and Remote Experimentation in Control Engineering. *Proc. 11th EAEEIE Annual Conference on Innovations in Education for Electrical and Information Engineering*, University of Ulm, 2000, S.213-218

[6] Chr. Schmid, Remote Experimentation in Control Engineering. *Proc. 11th Mediterranean Conference on Control and Automation MED 2003*, Rhodos, Paper IV12-01.

[7] Resource Center for Engineering Laboratories on the Web, University of Tennessee at Chattanooga, http://chem.engr.utc.edu

[8] M. Štula, Software agents in monitoring and control of dislocated systems, Master thesis (in Croatian), *Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture*, University of Split, Split, Croatia, 2001.

[9] T. Finin, R. Fritzson, D. McKay, R. McEntire, KQML as an Agent Communication Language, *The Proceedings of the Third International Conference on Information and Knowledge Management (CIKM '94),* ACM Press, 1994

[10] M. Štula., D. Stipaničev, Monitoring and control of complex, distributed systems using agent technology, *Proc. of SOFTCOM 99, Int. Conf. on Software in Telecommunications and Computer Networks,* Split-Rijeka -Venice , pp. 347-354, 13-16.10. 1999.

[11] S. Franklin, A. Graesser., Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents, *Proceeding of the Third International Workshop on Agent Theories, Architectures, and Languages,* Springer-Verlag, 1996.

[12] M. Štula., D. Stipaničev, M. Bonković., Questions and possible answers about ontologies concerning intelligent agents, *Proc. of SOFTCOM 2001 - Int. Conf. on Software in Telecommunications and Computer Networks,* Split-Dubrovnik-Ancona-Bari, pp. 775-781, 9-12.10. 2001

[13] D.Stipaničev, J.Marasović, Parameter estimation of impulse transfer function, *E-textbook Digital control on-line* (in Croatian), http://laris.fesb.hr/digitalno_vodjenje/text_3-13.htm

[14] J. Marasović, M. Čić , M. Žuvela, Process identification based on linear regression of data measured through the Internet, *Proc. Of SOFTCOM 2003. Int. Conf. on Software in Telecommunications and Computer Networks* , Split – Dubrovnik – Ancona - Venezia, pp.454 – 457, 7–10.10.2003.