Controlling of a Greenhouse by Using Intelligent Agent System

Tuncay Aydoğan and Serap Ergün

***Abstract*—An intelligent agent system for the control of greenhouse automation is presented in this paper and a prototype multi-agent-based control system has been developed for greenhouse as an example. The developed automation consist of temperature, humidity, CO_2, light, wind sensors, ventilation, irrigation, shading, lighting control systems and communicates with the computer over RS232. These sensors are modeled as intelligent agents and the greenhouse automation system architecture to implement the multi-agent-based control system is proposed and the coordination model of the reactive agents is developed. Finally, this study explains how to model these agents with JADE.NET which is one of the agent development platforms.**

***Index Terms*—Intelligent agent system, reactive agent, agent development platform, greenhouse control, JADE.NET.**

I. Introduction

Greenhouse is a vegetable production structure that allows the creation of necessary optimum growing conditions the plant's growth, development and reproduction as all physiological activities to continue at the highest level. Greenhouse performs the optimum conditions to be created under the influence of environmental conditions which feature continuously variable [17].

The main function of the greenhouses is to provide the most suitable environment for the growth of plants. For managing the greenhouse effect, it must be continuously measured several variables such as solar radiation inside and outside the greenhouse, temperature, wind speed, rain, relative humidity [23]. The changing of environmental conditions causes two major problems for the control of the parameters greenhouse. First one is to choose the most appropriate level for the plant and the second one is the control that is supplied to the selected level. Therefore, to ensure the optimum climate conditions in the greenhouse and the protection it provided continuous monitoring of climatic conditions and to intervene if necessary. To keep the optimum conditions in greenhouse, there is need for computer-microprocessor controlled information processing and automation systems [25]. The control of greenhouses by computers, embedded, distributed and remote data collection technologies are used. In addition, according to literature it is seen that such as neural networks, artificial intelligence algorithms, artificial immune systems, learning algorithms, search algorithms are used to provide automatic control for greenhouses [10], [20], [26], [29], [30].

In this study; on a greenhouse prototype a multi-agent architecture has reactive properties designed and intended to be applied. Control parameters in the greenhouse, are formed as agents. The behaviours of the agents are modelled by using JADE.NET one of development platform. Greenhouse prototype developed by communicating via RS232.

The four main steps of this paper are:

1. In this study; on a greenhouse prototype is intended to be designed and implemented an intelligent agent with a reactive architecture features multi-agents.
2. Establishing necessary to provide active control of the greenhouse is aimed to be tested on a prototype and communicate with each other.
3. Working prototype developed under greenhouse measuring 90 cm x 40 cm x 35 cm in (dimensions) size; indoor and outdoor temperature, humidity, wind speed, consist of CO_2 ratio sensor and the control driver circuit. The prototype computer RS232 (Serial Port) are communicating through.
4. To perform their duties of agents, behaviors are coded with JADE.NET.

The two main contributions of this paper are:

1. It shows JADE.NET development platform can be used as a factor.
2. The contribution of the operation in hardware, thanks to the intelligent agent technology creates greenhouse prototype "sense-think-act" autonomously communicating unit of the environment (automatic) is shown the system can be developed.

II. Materials

A. An Agent, Intelligent Agent and Multi-Agent Systems

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed. The agent takes sensory input from its environment, and produces as output actions that affect it [18], [19].

Intelligent Agent is an entity that is able to keep continuously balance between its internal and external environments.

Wooldridge and Jennings [27] define an intelligent agent as one that is capable of flexible autonomous action to meet its

Manuscript received August 7, 2017; revised November 14, 2017.

The authors are with Suleyman Demirel University Faculty of Technology, Software Eng. Dept., Turkey (e-mail: tuncayaydogan@sdu.edu.tr).

doi: 10.18178/ijiee.2017.7.6.683

design objectives. Flexible means that reactivity, pro-activeness, social ability, mobility, veracity, benevolence, rationality and learning/adaptation [27], [28].

Multi-agent systems are systems composed of multiple interacting computing elements, known as agents. Agents are computer systems with two important capabilities. First, they are at least to some extent capable of autonomous action- of deciding for themselves what they need to do in order to satisfy their design objectives. Second, they are capable of interacting with other agents- not simply by exchanging data, but by engaging in analogues of the kind of social activity that we all engage in every day of our lives: cooperation, coordination, negotiation, and the like [11], [18], [22], [24].

It has been shown in the literature of the multi-agent systems used in different areas [1], [8], [12]-[15], [22], [26].

The multi-agent systems field can be understood as consisting of two closely interwoven stands of work. The first is concerned with individual agents, while the second is concerned with collections of these agents [11], [27].

Reactive agents on the contrary do not have representation of their environment and act using a response type of behaviour that is they respond to the present state of the environment in which they are embedded. Thus, reactive agents follow simple patterns of behaviour which can easily be programed.

The most interesting part of using reactive agents do not lie in the way one agent behave, but in their ability to interact with other agents in a simple way from which global complex patterns of activities can emerge.

B. JADE

JADE (Java Agent DEvelopment Framework) [3], [4] is a well-known MAS (multi-agent system) software development framework implemented in Java language by FIPA (Foundation for Intelligent Physical Agents). It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of tools that supports the debugging and deployment phase. It provides many merits for us to develop MAS programmes. The platform can be distributed across different platforms (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another one on demand. The only system requirement is the Java Run Time version 1.4 or later [16]. The communication architecture of JADE offers flexible and efficient messaging, where JADE creates and manages a queue of incoming ACL messages, private to each agent. Agents can access their queue via a combination of several modes: blocking, polling, timeout and pattern matching [3]. The full FIPA communication model has been implemented and its components have been fully integrated: interaction protocols, envelope, ACL, content languages, encoding schemes, ontologies and, finally, transport protocols. The transport mechanism, in particular, is like a chameleon because it adapts to each situation, by transparently choosing the best available protocol [5]. Java RMI, event-notification, and IIOP are currently used, and more protocols can be easily added and integration of HTTP has been already achieved. Most of the interaction protocols defined by FIPA are already available and can be instantiated after defining the application-dependent behaviour of each state of the protocol. SL (a formal language developed by FIPA) and agent management ontology have been implemented already, as well as the support for user-defined content languages and ontologies that can be implemented, registered with agents, and automatically used by the framework. Since the newer version of JADE, an in-process interface has been implemented that allows external Java applications to use JADE as a kind of library and to launch the JADE Runtime from within the application itself. In particular, the application can control the life-cycle of the Agent. This improves the agility and enlarges the applicability of JADE [4].

JADE.NET: The JADE object model includes a class hierarchy that is similar to that of .NET. This common object-oriented approach means there is a natural fit between JADE and .NET, enabling .NET objects to be modelled from JADE database.

A .NET runtime application connects to a JADE system as a standard JADE client using the jomdotnet.dll library file on Windows (Fig. 1).

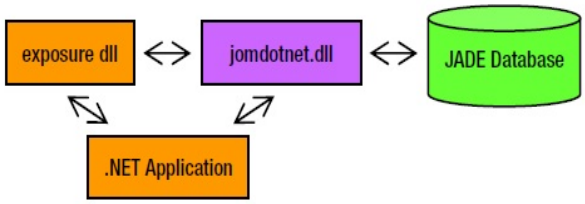

Fig. 1. How a .NET application connects to JADE [3].

JADE.NET class library; provides the access method to .NET Users to classes in the database of JADE .NET Class library of classes that jade is concerned (with the characteristics of the method) was formed by expanding. The using wizard generates C # class files to reveal its features JADE. So it can be rebuilt the .NET class library. Agents operate independently and execute in parallel with other agents. To support efficiently parallel activities within an agent, Jade has introduced a concept called [7], [9].

There is a class hierarchy similar to the object model between JADE and .NET. This common object-oriented approach means that there is a natural fit between JADE and .NET. This allows .NET objects to be modeled from the JADE database. A .NET runtime application connects to a JADE system as a standard JADE client using the jomdotnet.dll library file on Windows.

Behaviour is basically an Event Handler – it describes how an agent reacts to an event. An event is a relevant change of state e.g. a reception of a message or a Timer interrupts (after the specified time has elapsed). A fundamental characteristic of multi-agent systems is that individual agents communicate and interact. According to the FIPA specification, agents communicate via asynchronous messages. Each agent has a sort of mailbox (the agent message queue) where the JADE runtime posts messages sent by other agents [2], [21].

The work that an agent must perform in the framework of JADE.NET is done with behaviours. A behaviour is

represented as a task that an agent can perform and is implemented as an object of the class that "extends" jade.core.behaviours.Behaviour. To ensure that an agent fulfils the task performed by a behavioural object, It is enough to add the addBehaviour () method of the Agent class as a behaviour. Behaviour can be added when an agent starts (in the setup () method) or they can be added at any time from other behaviours.

JADE includes also some ready to use behaviours for the most common tasks in agent programming, such as sending and receiving messages and structuring complex tasks as aggregations of simpler ones. Behaviour is an abstract class that provides the skeleton of the elementary task to be performed. It exposes two methods: the action() method, representing the "true" task to be accomplished by the specific behaviour classes; and the done() method, used by the agent scheduler, that must return true when the behaviour is finished and can be removed from the queue, false when the behaviour has not yet finished and the action() method must be executed again.

The agents' behaviours in our greenhouse model are CyclicBehavior and TickerBehavior.

CyclicBehaviour is an abstract class that models atomic behaviours that never end and must be executed forever. CyclicBehaviours are designed to never complete; their action() method executes the same operations each time it is called. The jade.core.behaviours.CyclicBehaviour class already implements the done() method by returning false and can be conveniently extended to implement new cyclic behaviours [4].

The TickerBehaviour has action() and done() methods pre-implemented to execute the onTick() abstract method repetitively, waiting a given period (specified in the constructor) after each execution. A TickerBehaviour never completes unless it is explicitly removed or its stop() method is called [4].

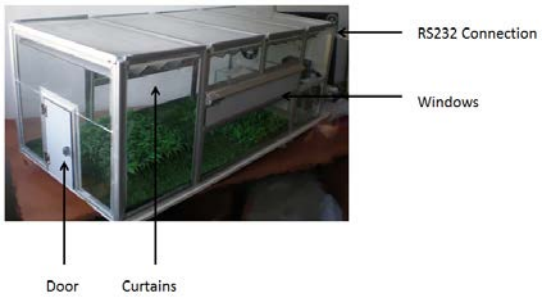

Fig. 2. The greenhouse prototype.

III. Methods

A. The Greenhouse Control System

The multi agent architecture is developed on greenhouse prototype which can be seen from Fig. 2. Working prototype developed under greenhouse measuring 90 cm × 40 cm × 35 cm in (dimensions) size; indoor and outdoor temperature, humidity, wind speed, consist of CO_2 ratio sensor and the control driver circuit. The prototype computer RS232 (Serial Port) are communicating through. The dashboard of the system is shown in Fig. 3.

Fig. 3. The dashboard of the greenhouse prototype.

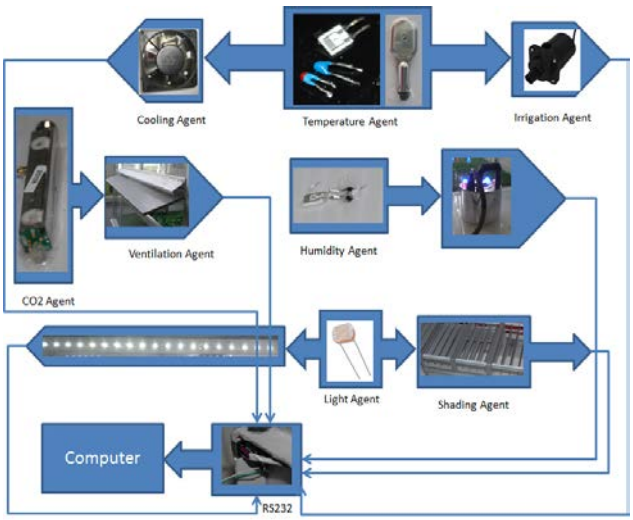

Fig. 4. The relations of agents.

Agents created and developed in this study are temperature, humidity, CO_2, ventilation, light, lighting, shading, cooling, irrigation and wind agents. The relations of created agents can be seen from Fig. 4.

For example; the temperature agent with TickerBehaviour assesses the situation by reading the temperature inside the greenhouse temperature sensor. With CyclicBehaviour, it sends the messages to the relevant agent(s). The relevant agents taken the message activate the control elements within that mission.

For measuring the temperature inside the greenhouse SHT11 temperature sensor is used, to display the measured values hygro-thermometer is used. NTC 100K temperature sensor with steel head is used for greenhouse outdoor temperature. The humidity sensor HC201 is used as humidity sensor. For CO_2 measurement the ZG106 sensor is selected, which measures over the number of particles in one million. For greenhouse lighting LED strips, for irrigation 32-04 brushless DC mini motor, to balance the humidity the AD304 humidity pump motor are used in the greenhouse. Curtains made of aluminium profiles on the greenhouse ceiling are used for shading system. Fans and windows for greenhouse ventilation are used. The fans work by increasing the temperature and windows are opened to outgas excess CO_2.

The study is created from temperature, humidity, CO_2, ventilation, light, lighting, shading, cooling, irrigation and wind factors. The task of the temperature agent; keeping

constant internal temperature set point, heating, cooling, heating if necessary communicates with irrigation factor is to ensure that the introduction of irrigation and cooling factors. Humidity agent, the internal humidity of the greenhouse is trying to keep in balance. When the CO_2 factor in domestic greenhouse CO_2 concentration increased ventilation agent calling task is to ensure the opening and closing of windows. Light agent is looking at whether the light inside the greenhouse lighting to ensure lighting and shading agents communicating performs open and close the curtains agents. An irrigation agent to make the greenhouse temperature in irrigation work fulfils its duties in accordance with the message received from the agent. Wind agent tells the wind speed outside the greenhouse system.

```
namespace Greenhouse
{
    public class HumidityAgent
    {

        override public void setup()
        {
            addBehaviour (new CollectMsgsBeh());
            addBehaviour (new WaitBeh());

        }

        public class CollectMsgsBeh:CyclicBehaviour
        {
            override public void action()
            {
                int humidity;
                int deshumvalue;
                int count=0;
                if (humidity<deshumvalue)
      {
                     ACLMessage msg=new
jade.domain.introspection.ACLMessage(ACLMessage.PROPOSE);
                msg.addReceiver(new AID ("Humidity",AID.ISLOCALNAME));
                msg.setLanguage("Agent Language");
                msg.setOntology("Turn on humidity pump motor");
                SenderBehaviour();
      }
                if (humidity> deshumvalue)
      {
                    ACLMessage msg=new
jade.domain.introspection.ACLMessage(ACLMessage.PROPOSE);
                msg.addReceiver(new AID ("HeaterAgent",AID.ISLOCALNAME));
                msg.setLanguage("Agent Language");
                msg.setOntology("Turn off Heater");
                SenderBehaviour();
      }
            }

            }
        public class WaitBeh:TickerBehaviour
        {
            protected void onTick()
            {
                incoming=ReceivedMessage();
                if(incoming!=null)
                {
                    longIncoming(incoming);
                        if(incoming.getAclRepresentation()!=ACLMessage.Equals)
                        {incoming=null;}
                }
                block(500);

            } } } }
```

Fig. 5. JADE.NET codes for humidity agent.

Fig. 5 shows the JADE.NET codes for humidity agent. Humidity factor provides a measure of the humidity of the greenhouse interior control parameters. As the temperature agent, it has CyclicBehaviour and TickerBehaviour, too. It performs a humidification operation running the pump motor to moisture in greenhouse. The classes are set before, and then the parameters are defined for the purposes of each behaviour. ACL Message determined to send messages to related agents.

IV. Results and Conclusion

The paper describes the development and implementation of control system. Establishing necessary to provide active control of the greenhouse is aimed to be tested on a prototype and communicate with each other. The system is able to interact in both directions with the greenhouse, to read the parameter values of sensors and to control the actuators. This study has demonstrated good quality of JADE.NET agent development platform applied to agent technology. Furthermore, the integration of multi agent technology with greenhouse is made possible thanks to definition of the Agent Behaviours. The contribution of the operation in hardware, thanks to the intelligent agent technology creates greenhouse prototype "sense-think-act" autonomously communicating unit of the environment (automatic) is shown the system can be developed.

Acknowledgment

The authors would like to thank to Süleyman Demirel University Research Project Coordination Unit.

References

[1] G. J. Anderson, *Multifactor Intelligent Agent Control: US Patent*, 2016
[2] F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi, "JADE—A java agent development framework multi-agent programming," *Springer*, pp. 125-147, 2005.
[3] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, "JADE: A software framework for developing multi-agent applications. Lessons learned," *Information and Software Technology*, vol. 50, no. 1, pp. 10-21, 2008.
[4] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE–A FIPA-compliant agent framework," presented at the Proceedings of PAAM, 1999.
[5] F. Bellifemine, A. Poggi, and G. Rimassa, "Developing multi-agent systems with a FIPA-compliant agent framework," *Software-Practice and Experience*, vol. 31, no. 2, pp. 103-128, 2001.
[6] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE: a FIPA2000 compliant agent development environment," presented at the Proceedings of the Fifth International Conference on Autonomous Agents, 2001
[7] F. L. Bellifemine, G. Caire, and D. Greenwood, "Developing multi-agent systems with JADE," *John Wiley & Sons*, vol. 7, 2007.
[8] A. Bielskis, V. Denisovas, D. Drungilas, G. Gricius, and O. Ramašauskas, "Modelling of intelligent multi-agent based E-Health care system for people with movement disabilities. Elektronika ir elektrotechnika," vol. 86, no. 6, pp. 37-42, 2015.
[9] G. Caire, "JADE tutorial: JADE programming for beginners," *Documentación de JADE*, vol. 3, 2003
[10] S. H. Chen, A. J. Jakeman, and J. P. Norton, "Artificial intelligence techniques: An introduction to their use for modelling environmental systems," *Mathematics and Computers in Simulation*, vol. 78, no. 2, pp. 379-400, 2008.
[11] J. Ferber, "Multi-agent systems: an introduction to distributed artificial intelligence," *Addison-Wesley Reading*, vol. 1, 1999.
[12] R. Hafezi, J. Shahrabi, and E. Hadavandi, "A bat-neural network multi-agent system (BNNMAS) for stock price prediction: Case study of DAX stock price," *Applied Soft Computing*, vol. 29, pp. 196-210, 2015.
[13] J. M. Harley, F. Bouchet, M. S. Hussain, R. Azevedo, and R. Calvo, "A multi-componential analysis of emotions during complex learning with an intelligent multi-agent system," *Computers in Human Behavior*, vol. 48, pp. 615-625, 2015.
[14] E. M. Kan, K. Yadanar, N. H. Ling, Y. Soh, and N. Lin, "Multi-agent control system with intelligent optimization for building energy management," presented at the Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems, vol. 2, 2015.
[15] W. Li, T. Logenthiran, and W. Woo, "Intelligent multi-agent system for smart home energy management," presented at the Smart Grid Technologies-Asia (ISGT ASIA), 2015 IEEE InnHata! Başvuru kaynağı bulunamadı.ovative.
[16] S. D. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci, and T. Funabashi, "Multi-agent systems for power engineering applications—Part II: Technologies, standards, and

tools for building multi-agent systems," *Power Systems*, IEEE Transactions on, vol. 22, no. 4, pp. 1753-1759, 2007.
[17] T. Morimoto and Y. Hashimoto, *An Intelligent Control Technique Based on Fuzzy Controls, Neural Networks and Genetic Algorithms for Greenhouse Automation*, 1998.
[18] W. Shen and D. H. Norrie, "Agent-based systems for intelligent manufacturing: a state-of-the-art survey," *Knowledge and Information Systems*, vol. 1, no. 2, pp. 129-156, 1999.
[19] B. Slotznick, *Intelligent Agent for Executing Delegated Tasks: Google Patents*, 1999.
[20] H. Tantau, "Optimal control for plant production in greenhouse. Mathematical and control," *Applications in Agriculture and Horticulture*, pp. 1-6, 1991.
[21] J. Tutorial, *Jade Programming for Beginners: TILAB*, 2003.
[22] E. Udoh, V. Getov, and A. Bolotov, "Sensor intelligence for tackling energy-drain attacks on wireless sensor networks," *Automated Reasoning Workshop*, May, 2016.
[23] S. Üstün and A. N. Baytorun, "Sera projelerinin hazirlanmasina yönelik bir uzman sistemin oluşturulmasi," *KSÜ Fen ve Mühendislik Dergisi*, vol. 6, no. 1, pp. 168-176, 2003.
[24] W. Van der Hoek and M. Wooldridge, "Multi-agent systems," *Handbook of Knowledge Representation*, pp. 887-928, 2008.
[25] G. Villarrubia, J. F. D. Paz, D. H. Iglesia, and J. Bajo, "Combining multi-agent systems and wireless sensor networks for monitoring crop irrigation," *Sensors*, vol. 17, no. 8, pp. 1775, 2017.
[26] T. Wanyama and B. Far, "Multi-agent system for irrigation using fuzzy logic algorithm and open platform communication data access.world academy of science, engineering and technology," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 11, no. 6, pp. 626-631, 2017.
[27] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowledge Engineering Review*, vol. 10, no. 2, pp. 115-152, 1995.
[28] M. Wooldridge, N. R. Jennings, and D. Kinny, "The Gaia methodology for agent-oriented analysis and design," *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 3, pp. 285-312, 2000.
[29] L. A. Zanlorensi, V. M. Araújo, and A. M. Guimarães, "Automatic control and robotics for greenhouses: a review on heating technologies," *Iberoamerican Journal of Applied Computing*, vol. 4, no. 3, 2016.
[30] B. Zhang, L. Yang, J. Zhu, Y. Zhao, and N. Liu, "Intelligent monitoring system of light intensity and CO_2 concentration in strawberries greenhouse," in *Proc. 2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, IEEE, August 2017, pp. 101-106.

Tuncay Aydoğan received the PhD degree in 2005 from Sakarya University. He is currently an associated professor of Suleyman Demirel University. His research interests include smart systems, fuzzy logic, automatic control and ındustrial networks.

Serap Ergün is a research assistant at the Süleyman Demirel University. Her research interests include intelligent agents, artificial intelligence, game theory, network routing.