Int. Agrophysics, 2007, 21, 17-27

INTERNATIONAL
Agrophysics
www.ipan.lublin.pl/int-agrophysics

Embedded real-time system for climate control in a complex greenhouse**

*A. Candido, F. Cicirelli, A. Furfaro, and L. Nigro**

Software Engineering Laboratory, Department of Electronics Informatics and Systems Science, University of Calabria, 87036 Rende, Italy

Received September 25, 2006; accepted December 1, 2006

A b s t r a c t. This paper describes the development of an embedded real-time system devoted to microclimate control in a complex greenhouse. The control system is capable of managing multiple, independent poly-tunnel units (PTUs). Both the internal temperature and humidity of PTUs as well as the external temperature, rainfall and wind conditions are monitored and regulate decisions of the control system. The control system is directed by parameters entered at configuration time through a user-friendly graphical interface. The realization depends on the use of Java technologies and on a specific methodology suited to the development of real-time systems. The approach is based on hierarchical state machines extended with timing constraints, and a supporting toolbox which enables graphical modelling, automatic code generation, simulation and real-time execution of a system. The paper discusses design and implementation aspects of the control system and reports information collected from real operation.

K e y w o r d s: greenhouse climate control, embedded real-time systems, statecharts, integrated development, Java

INTRODUCTION

Greenhouse climate control (Critten and Bailey, 2002) is a well-known challenging problem. It affects both the quality of produced horticultures and the economic expectations of the growers. Several models have been proposed in the literature, *eg* (Albright *et al*., 2001; Cunha, 2003; Caponetto *et al*., 2000; Miranda *et al*., 2006) which can predict the values of environment variables (air temperature, relative humidity, CO_2, light radiation, *etc*.) with the help of AI based techniques (genetic algorithms, artificial neural networks, fuzzy logic and so forth). These models are intrinsically difficult due to the complex interrelations which exist between greenhouse internal and external environment variables. For instance, model time-variant parameters are often required which make value prediction imprecise and with a limited time horizon validity. In addition, models can be highly demanding from the computational point of view, which makes them not always adequate for use in real-time operation.

In the work described in this paper a purely software-based approach is experimented. The approach is event-driven, time-driven and domain expert-driven. The behaviour of the control system mainly consists in the periodic reading of data from sensors and in the elaboration of corresponding response actions to be executed on actuators. Responses are sensitive to the configuration data entered by the greenhouse domain expert at system start-up time.

The development technology is centred on Java and in particular on the Hierarchical Communicating Real-Time State Machines modelling language and the supporting toolbox VIOLIN (Furfaro *et al*., 2006). VIOLIN permits visual modelling, simulation with RTL-like assertions, and Java code generation of an H-CRSM system. The generated code of a final system is automatically weaved with a custom runtime executive which supports cooperative concurrency and favours time predictability.

The achieved embedded real-time system is actually in charge of climate control in a complex greenhouse consisting of multiple and independent poly-tunnel units (PTUs), used for possibly different horticultures at a given time. The system is in real use and has significantly improved the management of the before manually operated physical system. Two basic goals were pursued: 1) protecting the greenhouse structure from damage by reacting in

*Corresponding author's e-mail: l.nigro@unical.it

**This work was partially supported by Progetto di Ricerca M.ENTE: Management of Integrated Enterprise, PON n° 12970-Mis.1.3 – M.I.U.R, Settore Agro-Industria Tema 2.

due time to external wind events and rainfalls, which obviously can also occur in combination; 2) maintaining the air temperature and relative humidity in the various PTUs within required ranges of values through PTU windows opening/closing and possibly by activating the local heater or the cooling sub-system.

The paper describes design and implementation aspects of the greenhouse climate control system together with its supporting H-CRSM methodology. The real operation of the system is then demonstrated through execution scenarios based on sampled history of logged events and reactions.

GREENHOUSE SYSTEM

The greenhouse physical system, located in the southern of Italy, occupies a surface of 12000 m^2, with 28 tunnels partitioned into 6 poly-tunnel units (PTUs). Figure 1 shows a typical PTU.

Fig. 1. A screenshot of a polytunnel unit.

The greenhouse produces both flowers *eg* chrysanthemum and horticultures *eg* string bean. Tunnels are covered by polyethylene sheets. PTUs are units for data sampling and control. They are provided with sensors for capturing the internal air temperature and relative humidity. Tunnels in a PTU have roof windows (Fig. 2a) extending over the whole length of the tunnel, which can be opened/closed through asynchronous three phase motors (Fig. 2b). Each PTU is also equipped with front/rear and lateral windows but at current time only the roof windows are under the automatic control of the software system. Depending on window placement, a tunnel can have one (near the highest position of the cover) or two (in low and opposite positions of the cover) windows.

The greenhouse is powered with a total wattage of 25KW. Particular attention was devoted to provisions capable of reducing the electrical power requirements. For instance, some PTUs can have their roof windows partitioned into two groups of alternating windows. When the need arises, windows of the two groups are then operated in an interleaved way. A similar provision is exploited for actuating the electrical fans of the PTU cooling system (Fig. 2c, d). In this case, up to three groups of fans can be established, with the first group comprising fan motors which are actuated through an inverter and the remaining group motors which are on/off actuated. Finally, a PTU can have a heater (boiler) which can generate warm air during winter cold conditions.

The whole greenhouse is equipped with a meteorological box with sensors for reading the external air temperature, velocity and direction of wind and the presence/absence of a rain/snow fall. Such data are shared and can influence the operation of all the PTUs.

DESIGN OF THE CLIMATE CONTROL SYSTEM

The following outlines the design and implementation of the proposed greenhouse climate control system -CCS-, along with its supporting methodology. The software system runs on a single Win2k platform.

Supporting methodology

CCS development is based on the H–CRSM modelling language (Furfaro *et al*., 2006) which is an extension with statecharts (Harel, 1987) of the Communicating Real-Time State Machines (Shaw, 1992; Raju and Shaw, 1994; Fortino and Nigro, 2000). H-CRSM represents a software architecture (Shaw and Garlan, 1996). Machines are the basic building blocks. They hide a behaviour modelled as a distilled statechart where only the or-decomposition of states is allowed. Machines follow a distributed model of concurrency: they share no data and concurrently execute except when they need to interact. The communication model is patterned *à la CSP ie* it is based on synchronous (rendez vous) unidirectional 1-to-1 channels with typed messages, linking matching input/output ports. State transitions in a machine are annotated by a guarded command with a timing constraint *ie* a time interval [a,b], $0 \le a \le b$, b can be ∞. When a=b, the time interval is abbreviated as [a]. Basic commands are: *input* (?), *output* (!), *timer* ?, or an *internal command*. The timing constraint is relative to the instant in time when the current state was entered and serves to express the temporal readiness of an enabled *ie* guard is true command. Internal commands have a hidden and implicit time constraint of $[0,\infty]$, whereas the explicit time interval is instead a duration interval $[d_1,d_2]$ meaning that the algorithm of the internal command is expected to require at least d_1 time units and at most d_2 time units to complete its execution. When the duration interval of an internal command is not specified, it defaults to 0 *ie* the command is supposed to consume a negligible time to complete. A timer command permits to schedule a timeout to occur after a given amount of time measured from the moment the current time is elapsed. In particular, a command like *timer(v)?[*Δt*]*

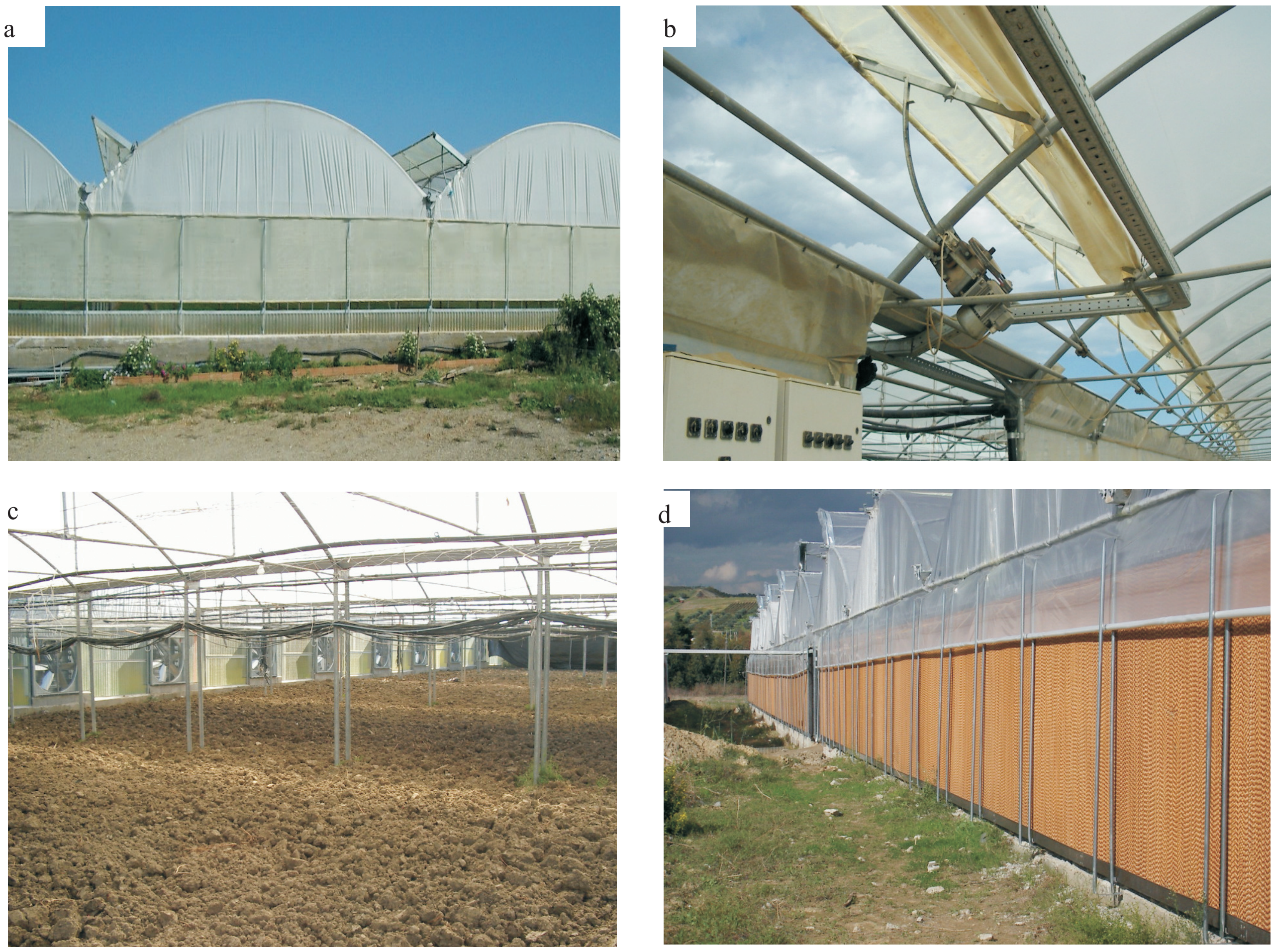


Fig. 2. Particular of: a – roof windows, b – a window motor, c – a PTU's cooling fans, and d – opposite side of PTU's cooling panels.

implicitly saves in the variable *v* the system real time (returned by the function *rt()*) when the timer expires. All of this can be exploited, for instance, to achieve a periodic behaviour.

Time constraints determine, among the enabled commands in a given machine, which command is eligible for execution. Conflicts among transitions outgoing current state of a machine are resolved according to the *earliest time first* strategy which forces firing the (or *a*) command having earliest occurrence time. The other commands are then discarded. In the case of multiple candidate events, one is chosen non-deterministically.

Full life-cycle development of H–CRSM systems is enabled by the VIOLIN toolbox (Furfaro *et al*., 2006) which permits graphical modelling, simulation with RTL-like assertions (Shaw, 1997; Fortino and Nigro, 2000) for functional/temporal property checking, and Java code generation of a system. The code generator weaves the final system implementation with a library of Java classes (framework) handling statechart hierarchies and dynamic evolution, interface classes for accessing the native driver code of sensors/actuators (Fig. 3) and a custom real-time executive which ensures cooperative concurrency instead of

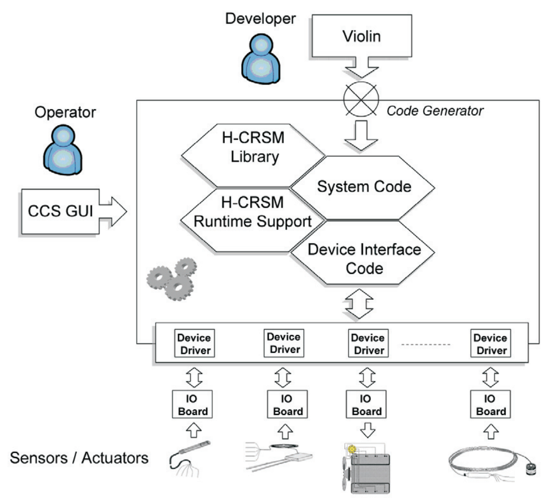


Fig. 3. CCS organization based on H–CRSM/VIOLIN.

over-killing concurrency and pre-emption. The runtime system makes it possible to statically allocate the memory of all the objects (states, transitions, channels, data messages and so forth) needed by an implemented system. The provision avoids interventions of the Java garbage collector.

CCS Architecture

Figure 4 depicts the H-CRSM architecture of CCS composed of a collection of interconnected statechart machines. The architecture is closed: particular machines *eg* sensor and motor machines explicitly model the external controlled environment (greenhouse). For instance, WindVelDirSensor machine reads wind velocity and direction; RainSensor communicates to the Controller the presence/absence of a rainfall; IntTempHumSensor machine samples the internal air temperature and relative humidity on a per PTU basis, and so forth. The Configurer takes data entered by the greenhouse manager at the CCS GUI and uses this information to initialize the various machines. The Configurer is also capable of starting/ stopping the Controller. Configuration data belong to the following categories:

a) physical boards information *ie* base address, port bit assignments for commanding a reading from a sensor or an actuation to a motor;

b) motor times *eg* the maximum time (≈70 s) required by a window motor for a full open/close operation starting respectively from a closed/opened state. Other attributes relate to the scanning times (motor scanners in Fig. 4) when actuating a group of motors of a PTU through interleaving;

c) control information *ie* the data upon which the control system bases its operation. The first kind of control information concerns the indication of the active PTUs *ie* those effectively included for monitoring and control. The second type of control information relates to the specification of the optional equipments of PTUs. For example, an enabled and active PTU can have or not the heater or the cooling system installed and/or enabled. The third kind of control information refers to thresholds *ie* the admissible ranges of values for the various controlled environmental variables, *eg* the internal temperature/humidity of a PTU, the wind velocity/direction with respect to which a PTU must react, the times at which, for a PTU, the day starts, the evening starts, the night starts, and so forth.

The control strategy of CCS was designed to react, in general, not to instantaneous changes in the environment variables, but to trend of variation. Toward this, values of a controlled variable *eg* wind velocity are buffered in a push-out queue so as to support trend variation reasoning. The size of a buffer depends on the reading period of the variable and on the observation period *ie* a time parameter entered by GUI which allows to memorize multiple variable samples. In addition, a settlement time can be used to separate consecutive control actions of CCS. Some events, though *eg* rain can require an immediate response. The control strategy is compatible with the temporal dynamics of the greenhouse system and purposely avoids intermittent window actuation, with obvious motor consumption problems *eg* during circumstances of rapid and irregular wind events.

Figure 5 shows a screenshot of CCS GUI for entering sensor parameters for the external meteorological box. As one can see, for each external variable, the reading period and linearization information necessary for translating electrical samples in the corresponding measurement units *eg* °C for the air temperature can be furnished. At configuration time, a functionality check can be performed by asking a read operation to a selected sensor. Set-up parameters can be verified for correctness and can be saved/ restored to/from a disc file.

The GUI panel in Fig. 6a permits thresholds for a given PTU to be entered. Both the meaning of low or high wind velocity is specified, as well as the orientation (against north) of the greenhouse, useful to sense wind direction. For the internal temperature, a critical minimum (tc_{min}), minimum (t_{min}), mean (t_{mean}), maximum (t_{max}), critical maximum (t_{cmax}) set points can be entered. Also the value (h_{max}) which specifies a high internal humidity can be specified. Finally, the operator can insert the minutes from midnight after which respectively the day starts, the evening starts or the night starts for the given PTU.

Panel in Fig. 6b allows one to enter, on a PTU basis, the response which CCS should actuate on the occurrence of external events. For example, in the event of a low wind, the PTU windows directly against wind should be opened at 30% of their maximum opening, whereas the PTU windows on the opposite side should be opened at 80%. The same behaviour is specified when the combination low wind/rain occurs. Figure 6b also shows the open percentage of PTU windows which should be adopted by CCS at the various time intervals of the day. The same GUI allows to enter the motor scanning time and the settlement time.

Figure 7 shows the internal structure of the top state of the Controller machine. Idle, Normal and EnergyMiss sub states are macro states *ie* they admit decomposition. Remaining states are leaf or elementary states. In the Idle state the Controller receives configuration parameters and sets up buffers for supporting trend variation analysis, both for external and internal sensors. EnergyMiss is reached when an energy loss event is sensed, in which case, if a proprietary power generator is available, the power generator can be used by CCS to continue execution or CCS can be reset and made idle. Dual operations are carried out on the coming back of external supplied electric energy. In the Normal state, Controller is capable of engaging communications with sensor machines in order to update its control status. Following each sensor interaction, the Controller resumes its previous behaviour (see the deep-history H* connector in Fig. 7).

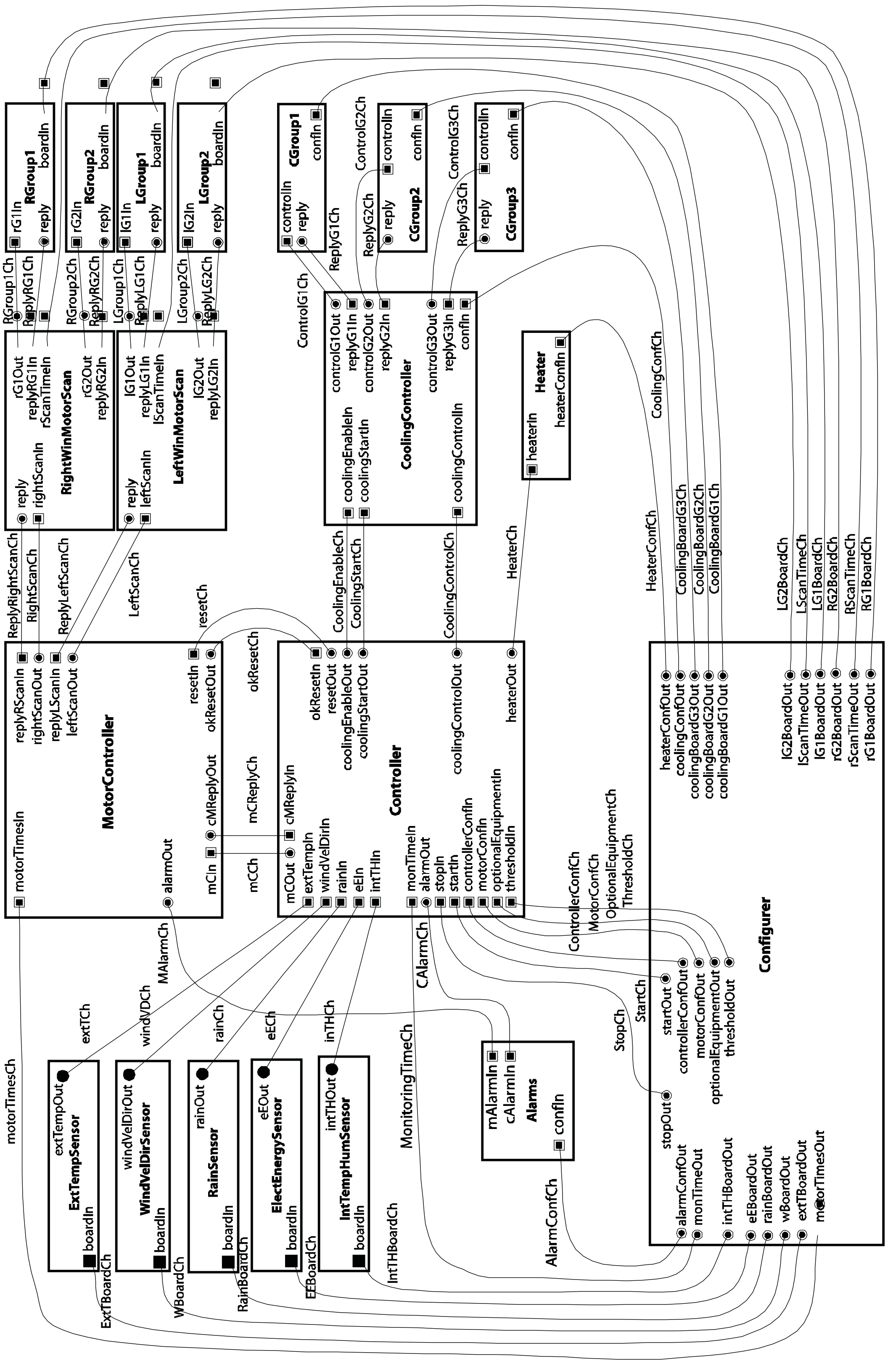
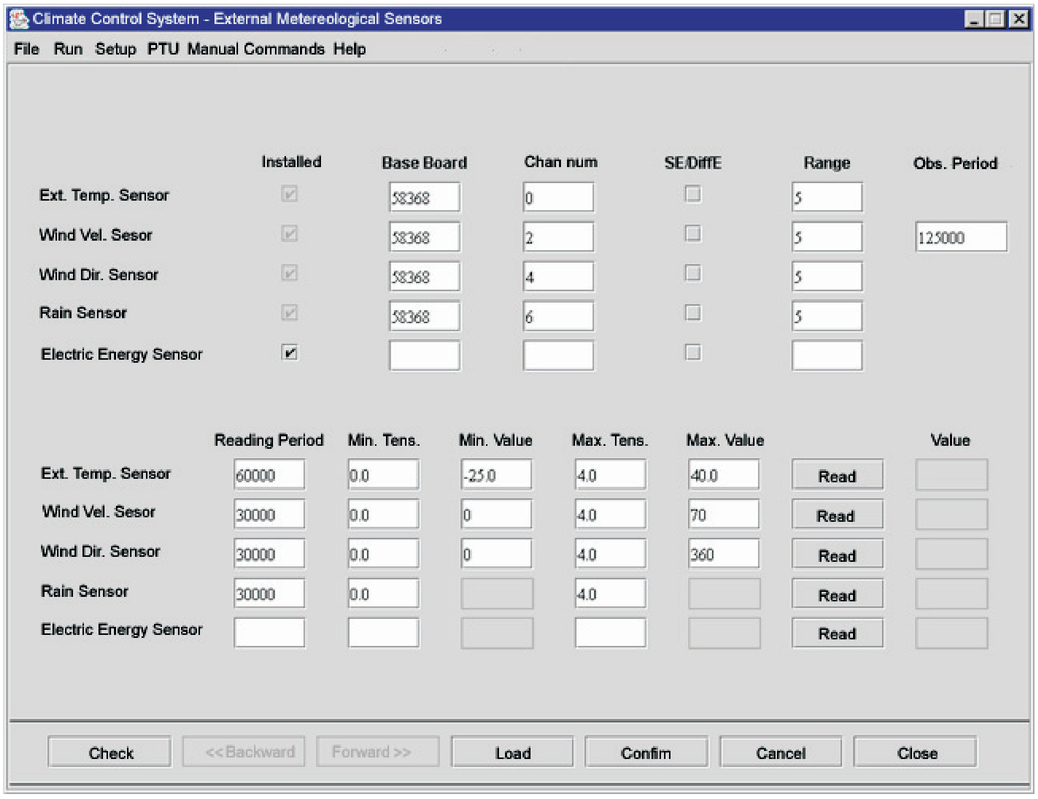


Fig. 4. H–CRSM architecture of CCS.


Climate Control System - External Metereological Sensors
File Run Setup PTU Manual Commands Help
Check
<<Backward
Forward >>
Load
Confirm
Cancel
Close

	Installed	Base Board	Chan num	SE/DiffE	Range	Obs. Period
Ext. Temp. Sensor	✓	58368	0	☐	5	
Wind Vel. Sesor	✓	58368	2	☐	5	125000
Wind Dir. Sensor	✓	58368	4	☐	5	
Rain Sensor	✓	58368	6	☐	5	
Electric Energy Sensor	✓			☐		

	Reading Period	Min. Tens.	Min. Value	Max. Tens.	Max. Value		Value
Ext. Temp. Sensor	60000	0.0	-25.0	4.0	40.0	Read	
Wind Vel. Sesor	30000	0.0	0	4.0	70	Read	
Wind Dir. Sensor	30000	0.0	0	4.0	360	Read	
Rain Sensor	30000	0.0		4.0		Read	
Electric Energy Sensor						Read	

Fig. 5. CCS GUI for setting up parameters for the meteorological box sensors.

Control logic

The mission of CCS is to preserve the integrity of the greenhouse system against wind/rain external events, then to maintain the internal temperature of PTUs within the corresponding required interval [$t_{min}..t_{max}$] and the relative humidity below the associated h_{max} value. Since temperature is assumed to be a more critical factor for growing plants in the greenhouse than humidity, CCS reacts to a high relative humidity but its ultimate goal is to keep the air temperature under control.

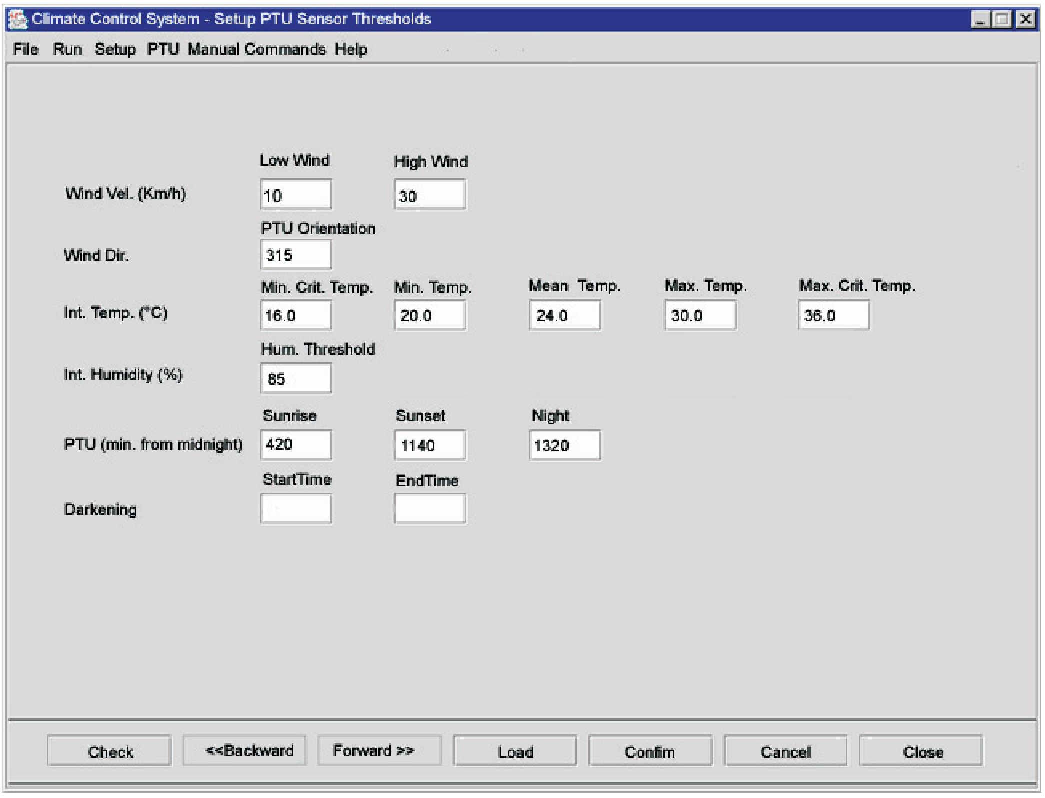
From time to time the external wind velocity/direction, rain and temperature conditions are analysed and the corresponding responses identified, namely the opening percentage of PTU windows. The required closing/opening actions, whose exact amount obviously depends on the actual opening percentage of windows, are compared against those prescribed by the greenhouse expert for the current time of day. The minimal request (worst case) is determined also considering the response requirements arising from the internal air temperature control. In order to summarize temperature control, it is useful to observe that two intervals exist for the relative humidity:

normal=[<h_{max}], exceptional=[>=h_{max}].

Normal humidity

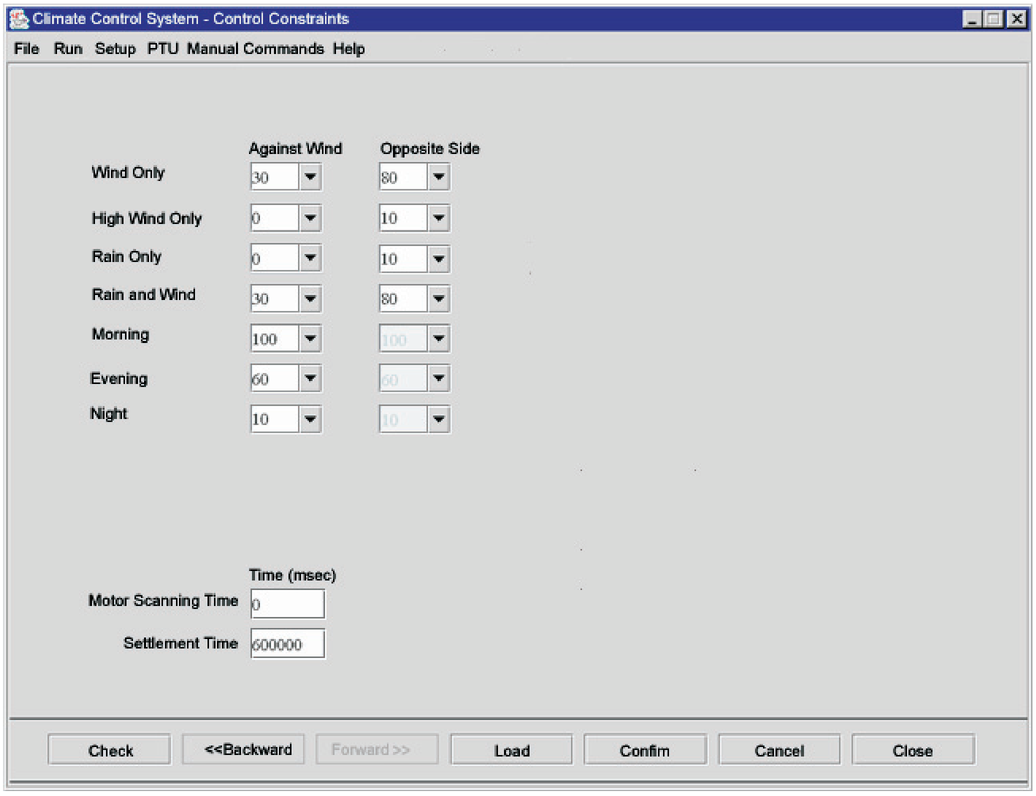
In the case the humidity is in the normal range, six intervals are considered for the internal air temperature: A=[<tc_{min}], B=[$t_{cmin}..t_{min}$], C=[$t_{min}..t_{mean}$], D=[$t_{mean}..t_{max}$], E=[$t_{max}..tc_{max}$], F=[>tc_{max}]. First the CCS classifies the temperature in its belonging interval, then its trend of variation is detected. It is the diminishing/increasing character of the air temperature which dictates the reaction of the CCS. In the case the interval is A or B and the PTU has an installed and enabled heater, the latter is turned on (with PTU windows totally closed). After two consecutive steps in which the internal temperature is found in this too cold condition and also the external temperature is low, an alarm is raised to the human operator in order to check heater functionality and/or the behaviour of temperature sensors. Another alarm is raised in the situation the heater should be turned on and it is disabled at the moment *eg* for maintenance problems. Similar considerations hold for the cases when the internal air temperature is found belonging to E or F intervals and the PTU has an installed and enabled cooling system (whose operation requires the PTU windows to be kept fully closed, too).

a


Climate Control System - Setup PTU Sensor Thresholds
File Run Setup PTU Manual Commands Help
Check
<<Backward
Forward >>
Load
Confirm
Cancel
Close

	Low Wind	High Wind			
Wind Vel. (Km/h)	10	30			
	PTU Orientation				
Wind Dir.	315				
	Min. Crit. Temp.	Min. Temp.	Mean Temp.	Max. Temp.	Max. Crit. Temp.
Int. Temp. (°C)	16.0	20.0	24.0	30.0	36.0
	Hum. Threshold				
Int. Humidity (%)	85				
	Sunrise	Sunset	Night		
PTU (min. from midnight)	420	1140	1320		
	StartTime	EndTime			
Darkening					

b


Climate Control System - Control Constraints
File Run Setup PTU Manual Commands Help
Check
<<Backward
Forward >>
Load
Confirm
Cancel
Close

	Against Wind	Opposite Side
Wind Only	30	80
High Wind Only	0	10
Rain Only	0	10
Rain and Wind	30	80
Morning	100	100
Evening	60	60
Night	10	10

	Time (msec)
Motor Scanning Time	0
Settlement Time	600000

Fig. 6. CCS GUI for: a – entering threshold parameters of a selected PTU, b – entering response parameters of a selected PTU.

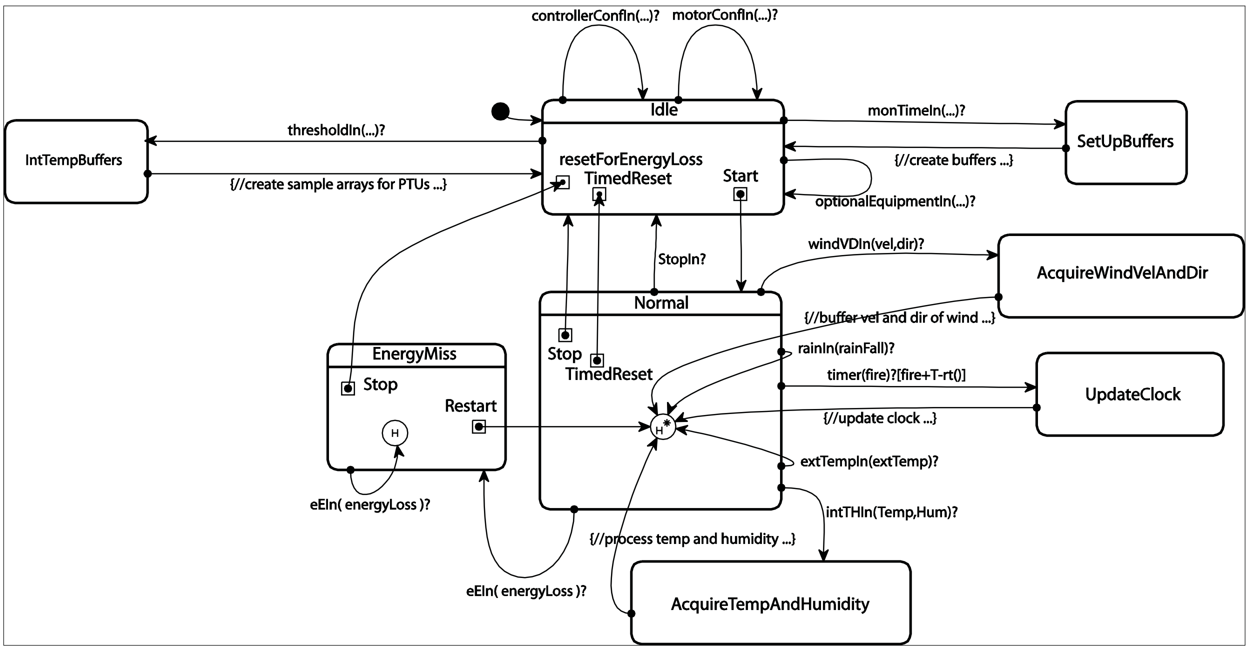


Fig. 7. Top state of Controller.

Exceptional humidity

In this case three intervals for the temperature are recognized: low=[<t_{min}], normal = [$t_{min}..t_{max}$], high=[>t_{max}]. If the internal temperature is low *eg* during winter nights or too cold days, and also the external temperature is low, provided the heater is installed and enabled, it is turned on with windows totally closed. When the external temperature is not low, an alarm is sent to the operator to check the temperature sensors. If the internal temperature and external temperature are low and the heater turned on, in order to control the humidity the following procedure is attempted. The heater is temporarily turned off. Then the PTU windows are forced to a 10% of opening percentage. The settlement time is then awaited. After that, if the system detects the same situation, the heater is turned on again and the windows fully closed. When the internal temperature is found not to be low, and the windows are required to be totally closed, the windows are temporarily forced to a 10% of opening and the settlement time waited. Then, if the situation persists, the windows are fully closed again: if it is true that a high humidity can cause some disease *eg* fungi to growing plants, it is also true that a too cold temperature can destroy the horticulture. Otherwise, if the humidity diminishes, the control behaviour reduces to that of normal humidity.

In the case the temperature is found in the normal range, nothing has to be done except to verify that the windows are in a partially opened state. Whenever, after two consecutive control steps, the situation repeats unchanged, an alarm is sent to the operator in order to check the relative humidity sensor.

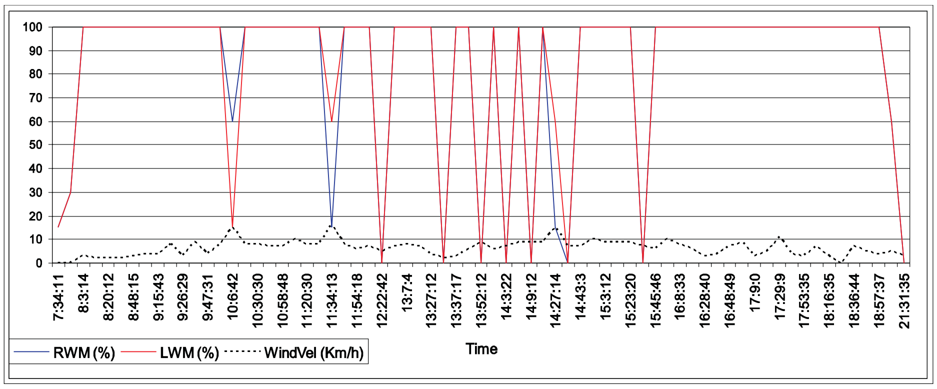
Finally, when the temperature is in the high interval, the PTU windows must be fully opened. In any case an alarm is signalled to the operator asking for a check to the humidity sensor.

OPERATIONAL CONCERNS

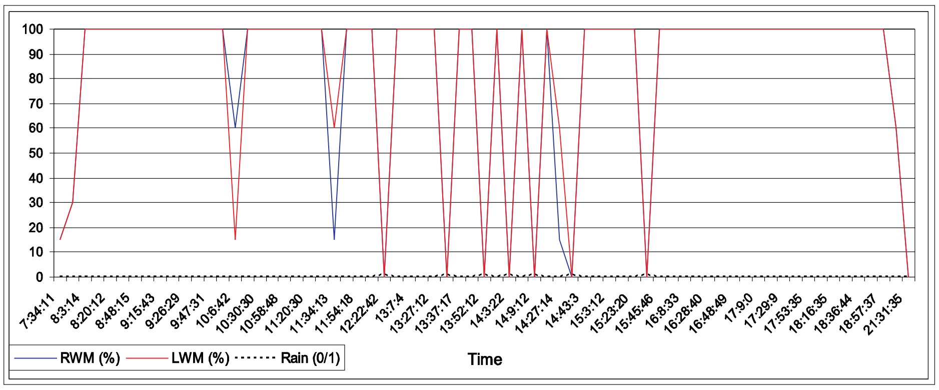
The CCS system was tested first in simulation, in the context of the Violin toolbox, then, incrementally, on the real physical system. During simulation, virtual devices (for sensors/actuators) were used and assertions (Raju and Shaw, 1994; Fortino and Nigro, 2000; Furfaro *et al.*, 2006) were introduced for checking the functional/temporal behaviour of the system. The actual shape of assertion programming, is shown below. A simple assertion is reported which checks that the Controller is always able to receive the latest sampled data from the WindVelDirSensor machine. The assertion is triggered when the control engine is up to dispatch a rendezvous on the WindVDCh channel between Wind VelDirSensor and the Controller.

```
when WindVDCh{
  if( time(WindVDCh, -2 )!= -1 )
    assert( time(WindVDCh, -1)-time(WindVDCh, -2)
    <=readingPeriod );
}//when
```

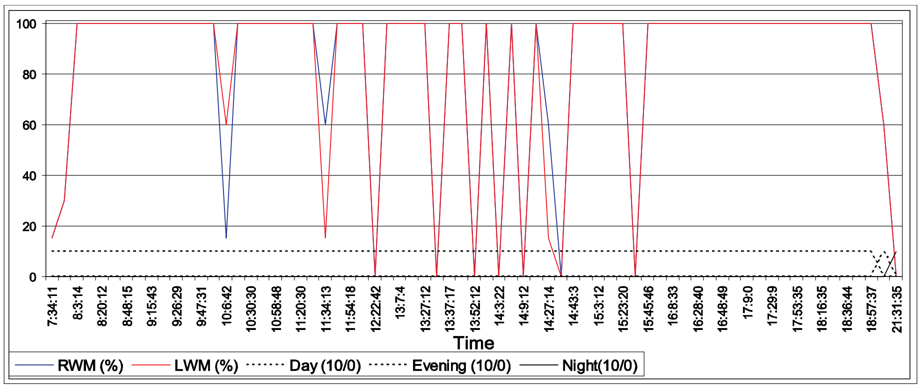
The VIOLIN control engine collects on a Timestamped Event History -TEH - (Shaw, 1997) the occurrence time and data value of each channel communication. The assertion a



b



c



d

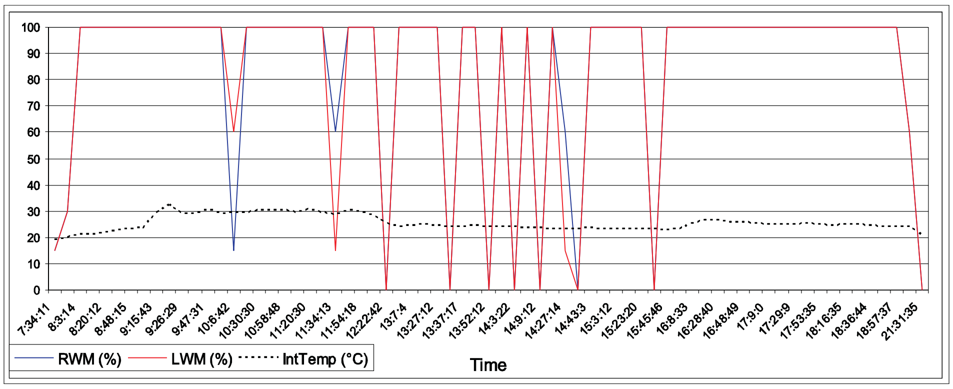


Fig. 8. Right/left window opening %: a – wind velocity vs. time, b – rain vs. time, c – vs. time of the day, and d – internal temperature vs. time of day.

first checks that a previous communication on the WindVDCh really exists on the relevant TEH, then that the time difference between current and immediately preceding communication is less than or equal to the sensor reading period (an input parameter for CCS). The assertion writes information about the check in a log file. The assertion was found always verified during the simulation. Other similar assertions were prepared for the other machines.

Snapshots from real execution

Assertions are normally excluded from the production build of an H-CRSM system. A special assertion, though, which simply stores the occurrence (time and value) of all events and reactions of CCS in a system history log file, was kept in the real-time operation in order to take snapshots from the real operation of the system.

Figures 8 depict the climatic behaviour of a particular PTU on a given day. On the chosen day some interesting external events (wind vs. rain) are sensed. The operation of the two window motor groups (RWM – right window motor, LWM - left window motor) is represented against time of the day and respectively: wind velocity, rain, opening thresholds required by times of the day and the internal air temperature.

In Fig. 8a the day starts for the PTU at about 7:00 a.m., when both RWM/LWM are opened for less than 20%, and ends at 19:00, when RWM/LWM begin to close (they are fully closed at 21:30). At about 8:00 both the windows are fully opened (100%). Figure 8 shows that at 10:08, due to a wind peak, RWM, which is in front of wind, is opened less than 20%, whereas LWM (on the opposite side with respect to wind) is opened at 60%. The situation is exactly reversed at about 11:34. After that, there are moments where both windows are completely closed *eg* at 12:22, although this action is not necessarily implied by wind velocity. As shown in Fig. 8b these instants effectively correspond to the presence of rain (rain bit true).

Figure 8c portrays the same information of RWM/LWM vs. time of day thresholds as entered through the CCS GUI. As one can see, at 21:00 the PTU switches to night.

Finally, phenomena can be watched from the viewpoint of variations of the internal air temperature (Fig. 8d). In reality, the considered PTU registers, on the chosen day, internal temperature which is always beyond the admitted maximal value, already at the beginning of the day. Accordingly, the worst-case climatic conditions command the operation of the window motors. For instance, the fact that from the beginning of the day the windows get completely opened is a direct consequence of the sensed high internal temperature. However, rain and wind can temporarily become the most critical events to which the CCS, for safety, has to react.

CONCLUSIONS

1. The greenhouse Climate Control System (CCS) described in this paper is characterized by the use of a Java centred custom and time-sensitive component-based software architecture (H-CRSM), and by its character of being application-expert configurable. The user-friendly graphical interface permits, for example, to configure the poly-tunnel units which are to be actively controlled and to enter threshold information which directs the CCS in the process of responding to climatic events.

2. Parameters which drive microclimate control are actually derived through experimental work and rely also on domain expert knowledge. In alternative, parameter values could be suggested by using specific models of greenhouse microclimates as described, for example, in Miranda *et al.*, 2006; Van Henten, 2003; Lees *et al.*, 2005.

3. CCS realization is cost-effective with respect to the cost of the controlled greenhouse physical system. Its practical use does not require computer engineering competence. Configuration data, which tend naturally to be reusable, can be saved and restored instead of being re-entered at each start-up.

4. Current efforts are directed at:
– improving data configuration,
– optimising the control strategy of the CCS,
– completing remote monitoring and control of the CCS through a smart phone.

5. Future directions which deserve further work are geared at:
– extending control to lateral and front/rear window motors in poly-tunnel units,
– enabling CO_2 and light radiation control,
– adding the automatic control of a darkening subsystem in selected poly-tunnel units,
– controlling the irrigation subsystem,
– adapting the system to hydroponics/aeroponics horticultures.

REFERENCES

Albright L.D., Gates R.S., Arvanitis K.G., and DrysdaleA.E., 2001. Environmental control for plants on earth and in space. IEEE Control Systems Magazine, October, 28-47.

Caponetto R., Fortuna L., Nunnari G., Occhipinti L., and Xibilia M.G., 2000. Soft computing for greenhouse climate control. IEEE Trans. On Fuzzy Systems, 8, 6, December, 753-760.

Critten D.L. and Bailey B.J., 2002. A review of greenhouse engineering developments during the 1990s. Agric. Forest Meteor., 112/1, 1-22.

Cunha J.B., 2003. Greenhouse Climate Models: An Overview. Proc. of EFITA 2003, 823-829.

Fortino G. and Nigro L., 2000. A toolset in Java2 for modelling, prototyping and implementing communicating real-time state machines. Microprocessors and Microsystems, 23, 3, 573-586.

Furfaro A., Nigro L., and Pupo F., 2006. Modular design of real-time systems using hierarchical communicating real-time state machines. Real-Time Systems, 32/1-2, 105-123.

Harel D., 1987. Statecharts: A visual formalism for complex systems. Sci. Computer Programming, 8, 231-274.

Lees M.J., Taylor J., Chotai A., Young P.C., and Chalabi Z.S., 2005. Design and implementation of a proportional-integral plus (PIP) control system for temperature, humidity and carbon dioxide in a glasshouse. Acta Horticulturae, 406, 115-224.

Miranda R.C., Ventura-Ramos E., Peniche-Vera R.R., and Herrera-Riuz G., 2006. Fuzzy greenhouse climate control system based on a field programmable gate array. Biosystems Eng., 94, 2, 165-177.

Raju S.C.V. and Shaw A.C., 1994. A prototyping environment for specifying and checking communicating real- time state machines. Software-Practice and Experience, 24, 2, 175-195.

Shaw A.C., 1992. Communicating real-time state machines. IEEE Transactions on Software Eng., 18, 9, 805-816.

Shaw A.C., 1997. Time-stamped event histories: a real-time programming object. Proc. 22nd IFIP/IFAC Workshop Real Time Programming (WRTP'97), 97-100.

Shaw M. and Garlan D., 1996. Software Architecture: Perspective on an Emerging Discipline. Prentice-Hall, Upper Saddle River, NJ, USA.

Van Henten E.J., 2003. Sensitivity analysis of an optimal control problem in greenhouse climate management. Biosystems Eng., 85, 3, 335-364.