GREENHOUSE AUTOMATION
Design Document

Module: 307CR Interactive Pervasive Computing
Coventry University, UK

Lecturers: Dr. Elena Gaura, Dr. James Brusey

Team: Greenhouse Gang

Members: Mandave Dosanjh (dosanjhm@coventry.ac.uk)
Daniel Knott (knottd@coventry.ac.uk)
Michael Straughan (straughm@coventry.ac.uk)
Péter Török (torokp@coventry.ac.uk)

Project website: http://greenhouse.progterv.info

Abstract

Greenhouse Automation Solution which senses, processes and stores temperature values. Managing temperature by comparing the measured values with the expected ones and take action if it is necessary. The solution aims are flexibility, maintainability and usefulness.

Table of Contents

1 Project Concept (Vision)

The goal for this project is create a system which will automatically control light intensity, soil and air temperature in a greenhouse in order to aid in the growth of plants of all types. By these aims the number of staff operating the greenhouse can be reduced or the size of production can be increased with the same number of staff.

2 Motivation

Plants are very delicate and in order to grow certain types of plants very specific conditions are required. In order to create these specific conditions a greenhouse is needed. Greenhouses are buildings made from glass or plastic panels which trap electromagnetic radiation which in turn can increase temperature and humidity within the walls of the structure. By doing this one is able to control the temperature within the greenhouse. As these plants need such specific conditions a lot of care and attention is required in order maintain the conditions. The temperature is usually controlled by opening windows on the greenhouse which lets out some of the stored up heat. Soil temperature is controlled by pumping hot or cold water through a heating system under the soil. Light intensity is controlled by opening and closing blinds which cover the greenhouse on the outside. The problem is that the temperature has to be continually monitored and the actuation (closing blinds, openings windows etc) has to be performed by hand. This is not only tedious but it is also expensive. Our automated greenhouse system will be able monitor all the important variables and automatically perform the actuation without the user being present.

3 Scope

As this is a university based project and we do not have the funds or means to create and implement a fully functioning system our project will focus only on monitoring temperature. In order to accomplish this there will be 5 parts of the system which will be created. The first of these parts is the GUI or graphical user interface. This interface will let user monitor all aspects of the greenhouse (in the case of our system the user will only be able to monitor the temperature) as well as add profiles for each type of plant will contain the desired temperature, light intensity etc (in the case of our system temperature only).

The second part of the system we will implement is the base station. This will in our system will be a laptop computer with a Bluetooth connection. The base station will display the GUI and well as process all data being received by the temperature sensors and relay it to the GUI and also back to the greenhouse for actuation.

The third part is a database which will contain all of the information entered by the user in order to create profiles. This database will also be stored on the base station.

The fourth and most important part of the system is of course the sensors. Sensors will be used in order to monitor the temperature at several positions in the greenhouse. As we are only building an example system only five sensors will be used. As these sensors are wireless we will also be using gumstix which will be connected the base station which will be receiving all temperature data in the form of UDP packets over a wireless Bluetooth connection.

The fifth part is the actuation, which are the physical occurrences that will be performed by the system. As for our system the actuation will be simulated in the form of text based alerts. Therefore when the system should open a window on the greenhouse our system will instead display a text box which will state that windows are being opened.

4 Technical specifications

4.1 Hardware

- 1 base station: PC (OS: Windows XP/Vista) *OR* PDA (OS: Windows CE) with Bluetooth
- 4 Gumstix node with Bluetooth (BT)
- 20 Temperature sensors (5 per node)

4.2 Software

4.2.1 Base station

- Microsoft Windows XP/Vista OS with .NET 3.5 framework
- Bluetooth stack (by the vendor of BT device)
- Application is written in C# , technologies used:
    - Windows Communication Foundation (WCF) for remote procedure call
    - Microsoft SQL Compact Edition with LINQ for data storing

4.2.2 Gumstix node

- Linux (Emedded) Operating System
- Application is written in Python 2.5
- Sensor handling: CogentTools (http://sourceforge.net/projects/cogenttools)
- Bluetooth connection handling: CogentTools

4.2.3 User interface

- Microsoft Windows XP/Vista OS with .NET 3.5 framework
- Application is written in C# , technologies used:
    - Windows Communication Foundation (WCF) for remote procedure call
    - Windows Presentation Foundation (WPF)

4.2.4 Actuator Controller (Mock-up)

- Microsoft Windows XP/Vista OS with .NET 3.5 framework
- Application is written in C# , technologies used:
    - Windows Communication Foundation (WCF) for remote procedure call

4.3 Communication Protocols

4.3.1 Gumstix to Base station

An Ethernet stack over the Bluetooth connection, through the Ethernet network UDP packets will be sent.

4.3.2 Base station to User interface

WCF connection built up between BS and UI, where BS is the server end-point. UI will be notified by callback function through WCF.

4.3.3 Base station to Actuator Controller

WCF connection built up between BS and AC, where BS is the server end-point. AC will be notified by callback function through WCF.

5 Requirements

5.1 Functional

- User friendly interface
- User will be able to create and store profiles
- Real time temperature values available at all times
- Graphical representation of temperatures in GUI
- Feedback to user in form of alerts

5.2 Non Functional

- Temperature values should be updated every second
- Temperature data should be sent in no less that 0.00001 of a second to the base station.

6 Implementation

6.1 Wireless network

The wireless sensor network consists of 20 temperature sensors and 4 Bluetooth enabled Gumstix in total. This will be partitioned to assign 5 individual temperature sensors to each Gumstix device. This is due to the Gumstix having a maximum capacity of 5 sensors connected to it at one time. These will be wired to the Gumstix to ensure good connectivity.

Reasons for the temperature sensors being wired are that this allows a more trustworthy, cheap and simple form of connectivity. If the sensors were wirelessly connected to the Gumstix then this would throw up a great deal of problems. There would be the problem of having to power each sensor and that of battery life. This would require the user to constantly keep replacing the batteries (or recharging) each sensor and would be very costly, awkward and time consuming. Mains power would not be a solution as that would require wires, meaning that the whole point of making the sensors "wireless" would be lost. Another reason not to make the sensors wireless is the amount of airborne traffic this would create. This would cause lots of interference and wireless network problems, with each individual sensors broadcasting its findings constantly. It would also add a lot of overhead in terms of the time it takes to send a temperature reading and also synchronisation issues concerning when all of the individual temperature readings are received by a particular Gumstix node.

Having wired sensors in this case are not a problem (in fact, due to the reasons listed above, it is an advantage) as the wires can be kept neatly out of the way with cable ties. For the in lab demonstration, the wires that connect each sensor to the Gumstix are relatively short and therefore do not allow much flexibility as to where they would be placed. If we were to go into manufacturing of our system, the length of these wires would be greatly increased, allowing for a wider spread of sensor placement and therefore a greater covering of temperature knowledge.

It was decided not to do any of the processing on the Gumstix devices as this could cause a slowdown in processing the data. Also, if the data is only sent out in a raw form then the base station has greater scope of what it can do with it, whereas if it was received in a processed form, parts of information concerning it could be lost. This helps in terms of extending the system at a later date, as only the base station code would need to be changed (assuming no different types of sensors needed to be added).

6.1. Picture Gumstix

Each Gumstix uses a python configuration file to get its setup parameters. These include the address and port of the base station that it will be sending to. The id of the Gumstix is also defined as a parameter, allowing the base station knowledge of which Gumstix is attempting to communicate with it.

The key parameter is a timeframe interval (in seconds) that each Gumstix will collect and send out temperature data. This allows the temperature data to be sent at regular intervals. Due to it being a configuration parameter rather than hard coded into the system, this enables the user the ability to change the interval between which new temperature data is broadcast be the system.

For our demonstration, it was chosen to use an interval of one second to show it was working, rather than having to wait half an hour or so (as would most likely be what the user would set the time lapse to in everyday use of the system) for every update.
It was chosen to use UDP as the protocol in which to send the data as this allowed for very little overhead. It was not vital to know whether the packets had been received by the base station so the choice not to use TCP was for this reason.

If the system is to be expanded in the future to be large enough where packet loss becomes an issue then the system could easily be changed over to TCP in order to deal with this problem. However at the present time, UPD more than suites the needs of the project and requires less overhead and therefore is the better solution.

The sent UDP packet contains a string of data of the following format:

```
NODE; nodeID
TIME; timestamp
SENSOR; subID ; tempVal
SENSOR; subID ; tempVal
SENSOR; subID ; tempVal
SENSOR; subID ; tempVal
SENSOR; subID ; tempVal
```

6.1. Code Packet content

nodeID is the ID of the gumstix, defined in its configuration file.

timestamp is the current time and date

subID is the ID of the sensor relative to that particular node. Sensors are numbered from zero. nodeID and subID together gives the global ID of a sensor

tempVal is the temperature value currently read from the temperature sensor

This format has been chosen as it does not process the information in any way but instead sends it in a clear manner that both easily readable for the base station machine its self (for in use of the system) and also human readable, to help with debugging.

6.2 Base station

6.2.1 Roles

- Connects to the gumstix nodes with a Bluetooth PAN (Peronsal Area Network), utilizing an Ethernet stack over Bluetooth
- Receiving data from Gumstix nodes in UDP packets
- Process and store the values in a database
- Send the data on-the-fly to the connected user interfaces
- Evaluates the given rules and send proper instructions to the actuators
- Regularly checks the system health, is data received from each node, each sensor, if not gives an alert

6.2.2 Technical description of Base station

The Base station is a multi threaded application, there is a thread for each different task, they run parallel

- Receiving data: Listening on the configured port and process all the packets received, this thread routes the data for Database, and the User interface host.
- User interface host is role to handle the connection between the Base station and the user interface. Utilizing Windows Communication Foundation (WCF)
    - Callback called when a sensed value comes, this notifies the User interface about new senses
    - Gives the required data from the database to the user interface
    - Number of sensors
    - Layout of rooms (where the sensors are physically)
- Actuator host is the interface for the actuation devices. This is also implemented using WCF, and the actuator device controller have to implement an interface with the bool Send(int actuatorID, string command, bool status) callback function. About rule evaluation and actuators see "How it working inside" part.

6.3 Graphical User Interface

6.3.1 Role

- Monitoring the current status of the greenhouse
- Showing the sensed values, the difference from expected values and temperature maps for each room.

6.3.2 Technical description of User Interface

The Graphical User Interface coded in Windows C# using Windows Presentation Foundation (WPF) which is a framework for creating GUI's under .NET 3.5. The communication with the Base station is based on Windows Communication Foundation (WCF) of the .NET 3.5 over TCP/IP, using callback function to notify the UI if sensed values are received at the BS.

After connecting to BS, the sensors' IDs and room layouts will be given to the UI. The layout of the UI (the number of boxes) is built up dynamically in runtime according to the data received from the BS.

6.4 The Actuator Controller (Mock-up)

6.4.1 Role

Displays messages when actuation is required.

6.4.2 Technical description of Actuator Controller

Runs separately from the base station, communicates over WCF with SOAP over TCP/IP. With this standard way, a different platform also can connect to the base station to handle actuator devices. The basic concept is that Actuator Controllers connects to the BS, notify it which actuatorIDs are handled by them. When an actuator should be triggered a message will be sent to the Actuator Controller with the actuatorID, command and a Boolean value according to that command set to TRUE (On) or FALSE (Off).

6.5 Gumstix Node

6.5.1 Role

Collects the sensed values (data) from the sensor devices connected to them. Then sends the data to the Base Station for further processing and storing.

6.5.2 Technical description of Gumstix Node

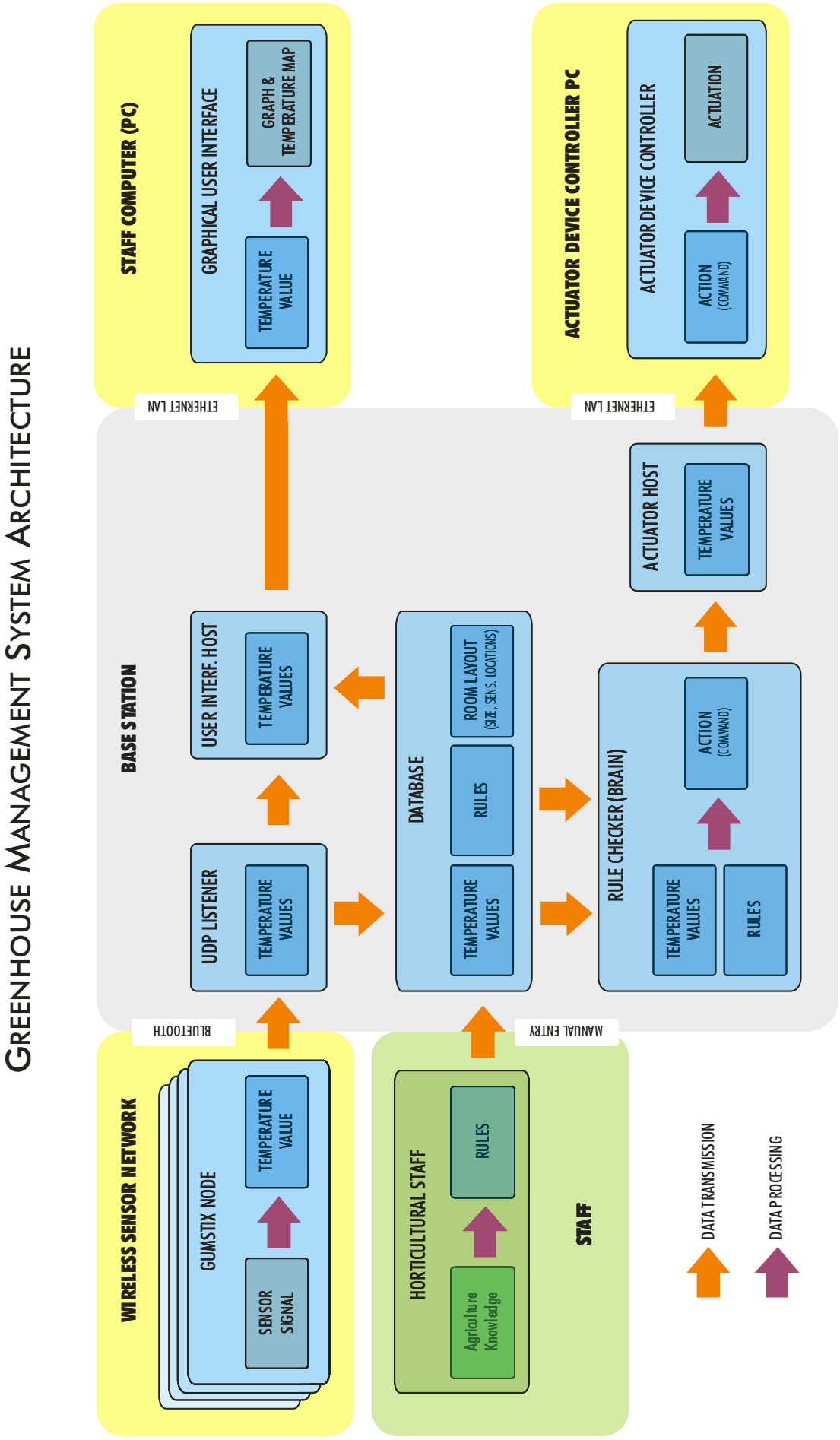
The sensors connected by wires to the gumstix board and values are read from them using CogentTools. In every 1 second (default, but stored in separate configuration file, adjustable according to the specific needs of application) the following tasks will be done:

- Read values from the sensors
- Build the message (described in the end of Chapter 6.1)
- Send the message to the BS in an UDP packet

7 The System

7.1. Picture

GREENHOUSE MANAGEMENT SYSTEM ARCHITECTURE



8 Evaluation

Although the implemented system is very basic it is still considered a success as we managed to meet the criterion by developing a pervasive computing application which can handle twenty temperature sensors on four gumstix node connected over a Bluetooth network.

We were not the only group in the class to develop a greenhouse but we approached the assignment much differently to the others by using C#.NET instead of Python to develop the application.

The different stages of the process is divided into different applications such as Gumstix Node app (Python), Base station (C#), User Interface (C#), Actuator Controller (C#) all of these are communication over some kind of network to give more flexibility to the overall system.

During the early stages of development, we decided to split the coding amongst us and focus on different aspects of the project. This method of working enabled us to progress and work as a team and we each contributed and worked well together. We discussed which sections we felt most comfortable working on and decided that:

- Mandave should develop the Graphical User Interface as he had experience developing interfaces from his previous year and was most in-tune with this aspect of the project in comparison to the others.
- Daniel and Michael should develop the wireless sensor part of the application as they felt comfortable using the library code and performing the low level operations required.
- Peter should develop Base Station as he was a bit more skilled with the technical aspects of programming

If we were given the opportunity to implement the system in a real greenhouse, the system would be updated to control other variables (the opportunity is open in the architecture) such as light intensity, air temperature as well as soil temperature and even CO2 levels. Overall, this approach worked very well, as each member of the group played to their strengths and allowed for the work to be brought together well.

8.1 Scale up considerations

The application architecture does not have actual limits to the number of sensors, nodes or rooms. However there are some bottlenecks for the whole system:

- Bluetooth communication (between the Gumstix Nodes and the Base Station) has a limited number of devices per Bluetooth PAN, which can be expanded with more BT device in the Base Station. These findings also apply for other communication channels, but according to the bandwidth and packet loss values, it first affects BT.
- The number of rules are related to the processing capacity needed for the Base Station, which is cannot be a problem in the case of a standard PC where usually there enough CPU capacity for thousands of rules. But it should be considered if we would like to run the BS software on a smaller device, such as PDA

8.2 Possible future extensions

- Actuating: Online connection the actuator hardware, such as heating system, air conditioning, watering, and feeding.

- Light sensors: Light intensity data could be collected, and comparing could be done between different rooms from different greenhouse materials, about the heat-catch of the house.
- Humidity sensors: Watering usually controlled on a time basis, according to the guess how much water is needed. With a humidity sensor the watering can be aligned to the actual needs, which can limit the resources go waste, decrease the costs.
- Statistical analysis: The data collected from the sensors are stored in the database, but not processed yet for statistical analysis, to make long term decisions. With analysis finding could be done about greenhouse materials, efficiency correlation with temperature, watering or feeding.
- Infra red pictures about the plants: Some of the plants' infections can be detected by the change in the infra red picture of the leaves. It needs infra red cameras, and algorithms for image processing, but can have fundamental outcomes, like the opportunity for immediate reactions, which can decrease the effects/spread of a disease.