Sensor Network Playground Monitoring System

WSN Lab (WS 13/14) - Group 2
Johny G. Malayil
Manisha Luthra
Vaibhav Singh
Boni Tom
Ravi K. Sharma

TECHNISCHE UNIVERSITÄT DARMSTADT

TECHNISCHE UNIVERSITÄT DARMSTADT

Abstract

Trees and plants play a vital role in conditioning of the atmosphere but there is a reverse effect on the plants themselves by the atmosphere. As plants and trees require their own preferred environment for survival and growth we can take help from technology to maintain the favorable conditions for the plants. Wireless sensor networks are good means to detect the state of the atmospheric parameters and based on these detected samples we can decide to introduce changes in the environment.

In this project, we attempted to make use of WSN nodes to learn about the environmental conditions around a small setup of two plants and change them as per the favorable conditions of the plants. Since, the test setup is housed indoor, so the changes to the environment were introduced manually in a controlled way. We also performed sample analysis on aggregated values and exploited actuators to modify the environment.

1.1 Problem Description

The goal of the project is to detect the environmental parameters around the plants and change the environment as per their favorable environment. In the experiment, there are two plants namely *Peperomia (Radiator Plant)* and *Kalanchoe*. The requirement is to deploy sensor nodes around the plants to take samples of various physical parameters such as temperature, humidity, CO_2 concentration, soil moisture and light exposure to the plant. Based on these sample values, we build a system which analyses sample values and compares them against the thresholds of favorable conditions and in case of a mismatch activates the corresponding actuators to modify the environment. The ideal conditions for the plants are as follows:

Peperomia

- Light : Place that receives moderate light.
- Temperature: The room temp should be between 65° to 75° F (18°C to 25°C).
- Humidity: It is best not to seal the room completely, as the plants are semi-succulent in nature.
- Watering: The soil needs to dry out between watering, but not become very dry.

Kalanchoe

- Light : Plants tend to get spindly in low light conditions.
- Temperatures : Between 55°F and 80°F (12.7 to 29°C) are ideal.
- Humidity: These plants do not require high humidity levels.
- Watering: Moderately throughout the summer and reduce watering in the winter.

1.2 Setup

The setup we used is shown in Fig. 1.1. It consists of six nodes - two sink nodes which can control the actuators and four other nodes responsible for sensing different phenomena as explained below.

- **Light, Humidity, Temperature:** Solar light, humidity in air and room temperature are environmental values which remain common for both plants. An explicit assumption here is that there are no microclimates and the application does not need to take care of it. Hence, we decided to keep a single sensor node to monitor all three of them.
- **Soil Moisture:** Soil moisture can vary for each of the plants according to moisture levels. Therefore, we chose to keep two separate sensor nodes to monitor moisture of each of the plants.
- CO_2 **Levels:** Since, the level of CO_2 in the air is supposed to be same in the vicinity of the playground, so we chose to keep a single sensor node for both of the plants. As before we assumed that there are no microclimates

1.2.1 Sensor Calibration for Physical Values

Sensor Calibration is a method of improving sensor performance by removing structural errors in the sensor outputs. Structural errors are differences between a sensor's expected output (raw voltage) and its measured output (unit of measurement), which shows up consistently every time a new measurement is taken. In our project we are calibrating the information acquired from sensors (Light, Temperature, Humidity, Carbon dioxide and Soil Moisture) in order to monitor the living conditions of plants. Since, even for the seemingly constant environment the sensors' raw values change significantly, sensors' calibration helps us to actually compare the calibrated values to the physical values.

1.3 Solution Design

During the design phase of our application, the team discussed about the architecture of the application. We discussed mainly two architectures considering the efficiency of the application in terms of energy consumption and intensity of communication needed, since communication is the most energy consuming operation. Mainly, the nodes have two roles 1. Sensor Nodes and 2. Sink Nodes. The two architectures are discussed in next two subsections.

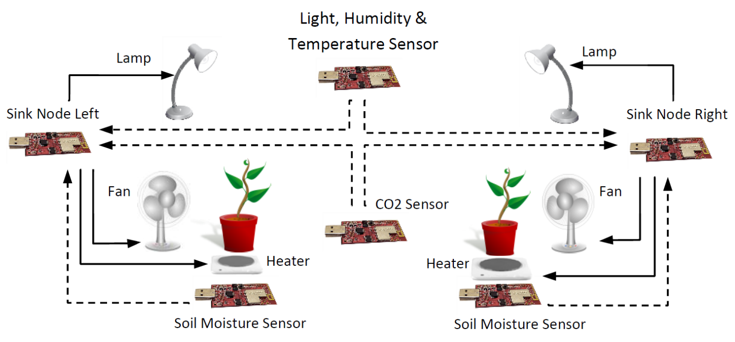

Figure 1.1: The diagram shows two sink nodes, two soil moisture sensor nodes, one CO_2 node and one node for light, temperature and humidity. The dashed arrows show the flow of command messages from sensor nodes to sink nodes and solid arrows show the control of actuators by the sink nodes.

1.3.1 Centralized Architecture

In the centralized architecture the sensor nodes act as data sample collecting nodes and only sense the environmental parameters and send these sensed values to the sink nodes. The nodes may or may not convert the raw values to real values of the parameters before they send them to sink nodes. Upon receiving the sample values, the sink node applies the aggregation or some specific statistical methods to obtain the current overall value for the specific parameter taking into account a set of values over time and determine if one of the actuators need to be activated. In case the current overall value of samples for a specific parameter crosses the threshold the sink node activates the corresponding actuator. The basic idea of this design is that, the intelligence for the decision making lies with the sink node as it receives the data from all other sensor nodes.

Advantage

The advantage of such an approach is that, the global state of actuators can be easily maintained, since the sink is always aware of the current state of the actuators(ON or OFF).

Disadvantage

The problem with this approach is that, all the sample values have to be transferred to the sink node so that sink node can analyze them, which requires a huge amount of packets to be sent from the sensor nodes to the sink node. As communication is the most energy consuming operation this architecture is not very energy efficient. Next we discuss the distributed architecture and compare it's efficiency to the centralized architecture.

1.3.2 Distributed Architecture

In the distributed architecture the nodes sense and analyze the samples over time in order to determine if the overall current value of a specific parameter crosses the threshold. So, the sensor nodes not only sense the environment but also apply statistical method for sample analysis. In simple terms, it can be said that every sensor node is made *locally intelligent* enough to decide if the actuator needs to be activated or not for the corresponding environmental parameter. If the sensor node locally decides that the actuator's state needs to be changed, it sends a command for state change to the sink node and sink node executes the command. In this architecture, the sink node acts as a dummy node which only sends the commands to actuators and no intelligence is provided to it.

Advantage

The advantage of such an approach is that, the samples are collected and analyzed locally at the sensor nodes and do not need to be sent to the sink node for the decision making. Rather, only the commands, in case of state a change, are sent to the sink node which saves a huge amount of communication in comparison to the centralized architecture.

Disadvantage

The main disadvantage of such an approach is that, it is not possible to maintain the global state of the actuators and hence there is a possibility of encountering a race condition among the sensor nodes which are using the same actuators. This race-condition can be resolved by having a priority based actuation of the sensor nodes.

1.3.3 Criteria for selecting distributed architecture

While analyzing the advantages and disadvantages of the two architectures, we can clearly see that centralized architecture is less efficient in terms of network traffic and this is the main criteria for selecting distributed architecture. We are aware of the problem of eventually running into race-condition due to the lack of global state but we try to recover from it using the prioritization of the sensors.

1.4 Communication Protocol Selection

In our setup the sensor nodes need to only communicate with the sink nodes, which can be achieved easily by having point to point communication. We analyzed the possibilities for the unicast, collect primitive, broadcast and runicast for their best suitability. Since the requirement is only to have a point to point communication, broadcast is not the best choice as it is a one to many primitive. The collect primitive seems to be a logical choice for data collection purposes in a WSN. The collect primitive makes use of the two channels, one for the neighbor discovery and other for the data sending. Since, the playground is a small scale setup and the possibility of not having a direct connection between the sensor nodes and the sink nodes is void, it does not make much sense to use the collect primitive. Other reason for not choosing the collect primitive is that it is much more energy consuming than unicast or runicast as neighbor discovery is a repetitive process.

Unicast and runicast are the main choices to make in our scenario where runicast provides a reliable delivery of packet with the help of ACKs. Although, the probability of a packet getting lost in case of runicast is also not zero, runicast provides mechanism for acknowledging the sent packets which makes it more reliable than unicast. Practically, we have to specify a maximum number of retransmissions in case of a lost packet in runicast, so lost transmissions beyond this number still brings the application back to an unstable state. Having such a limitation with the communication protocol we have to have a mechanism to deal with a lost command packets otherwise the natural choice would have been runicast in our scenario.

We conducted several experiments over time and we did not feel the need to have runicast as we did not notice a single packet loss in the small vicinity of playground. Therefore, we kept our communication protocol as unicast and went ahead with having a recovery method to deal with the lost packets. The other main reason for not selecting runicast is that the number of messages including the ACKs (in case of a packet loss) and retransmissions will be higher than the number of sent messages in the case of unicast protocol.

In case of a lost command packet while using the unicast method, a possible solution can be a recovery method based on timeout which monitors the sample values over a predefined time interval after sending the command to the sink (which is assumed to be lost). In case if the sample values do not converge towards the desired values (higher or lower than the threshold), the sensor node can decide to resend the command to the sink. The time interval can be determined for individual environmental parameters based on experience over time and criticality of the parameter. But in our current solution, we have not implemented this method as we did not notice any packet loss during the experiments.

Actuators' Local State Management

In our solution, since we use a distributed method of decision making and independently sending the commands to the sink node, we do not maintain the global state of actuators, but we maintain the local state of actuators in every node which sends commands to the sink node. We compiled a contention matrix shown in table 1.1 which shows that as per the priority of sensors the only race condition is between the CO_2 sensor and temperature sensor. In such a scenario we give priority to the CO_2 sensor and let the CO_2 sensor decide the state of actuator (The FAN).

	Light	Temperature	CO_2	Soil Moisture
Lamp	1	0	0	0
Heater	0	0	0	1
Fan	0	1	1	0

Table 1.1: Contention Matrix: The CO_2 sensor and temperature sensor both access the FAN

Prioritization of Parameters

Prioritization of contentious sensors is an important step in building such a solution. Since various nodes can send commands for changing the same actuator's state, we need to decide which parameter enjoys high priority. In our case, as shown in the contention matrix the temperature and CO_2 sensors might run into a race condition. In such a scenario we let the CO_2 Sensor decide the state of the actuator.

1.5 Data Sample Collection

The data collection has two main parts a. **Frequency of data collection** and b. **Analyses of samples** . The frequency of data collection determines how responsive the system is and analysis of samples determines the events when the state of the actuators needs to be changed. We briefly describe both parts in the following sections.

1.5.1 Frequency of Sample Collection

In our scenario, we introduced sudden changes in controlled environment, but in real world, these changes occur very slowly and the system is allowed to actually take a considerable amount of time to respond. An actual system will be much slower in response time and would ideally not respond to transients in the environmental conditions which is also necessary for energy efficiency. The calculations for the response times for our system are as follows:

CO_2 sensor takes around a 45 to 60 seconds to respond to CO_2 level changes, and the frequency of data collection is set to 3 seconds, which means after maximum 30 seconds (10 samples of values higher than the threshold value) the command to turn the FAN ON is sent to the sink. Also as soon as maximum 10 samples which have lower values than the threshold values are collected, the command to turn OFF the fan is sent, which also takes 30 seconds.

Light Sensor is more responsive and has a window size of 5 samples. The set frequency of sample collection is 2 seconds which means, within 10 seconds (5 samples of value higher than the threshold) the command to turn ON or OFF the lamp is sent.

Soil Moisture Sensor is also very responsive sensor and has a sample frequency of 2 seconds, which means after 20 seconds (i.e. 10 samples of values higher than the threshold), the command to turn on the heater is sent. Similarly, after maximum 20 seconds the command to turn OFF the heater is sent.

Temperature sensor samples are also collected at a frequency of 2 seconds which means after 20 seconds (window size 10) the state of corresponding actuator is changed.

1.5.2 Sample Analysis

The samples are stored in an array for the analysis and the size of the array is referred as the window size. The basic algorithm to analyze the samples is shown in list 1. As shown in the algorithm, the average of the samples is calculated each time a new sample is submitted. When the value of the average crosses a threshold the last five samples are checked for consistency with the condition. The check for the consistency of the most recent 5 samples helps in avoiding situations when there are sudden outliers due to some uncontrolled factors in the sensors. We could notice such outliers in the CO_2 sensor and then we filtered out them.

Listing 1.1: Pseudo Code for sample analysis algorithm

```
for each sample s in window w
  sum = sum + s;
average = sum / window_size;
if(average > threshold)
  if(last_five_samples > threshold)
    if(state_of_actuator == OFF)
      send_command_to_sink(ACTUATOR_ON);
else if(average < threshold)
  if(last_five_sample < threshold)
    if(state_of_actuator == ON)
      send_command_to_sink(ACTUATOR_OFF);
```

Clustering of Samples and Its Effect

Clustering of samples can occur when the samples have values very near to the threshold. For example, for a threshold of 40 VWC in case of soil moisture sensor, the samples can occur between 39 and 41. In such a case a frequent state change order for heater can follow. To avoid such a case, first the averaging of samples helps and secondly the monitoring of continuous 5 samples is helpful. These two methods bring down the possibility of triggering a state change due to clustered samples near the threshold.

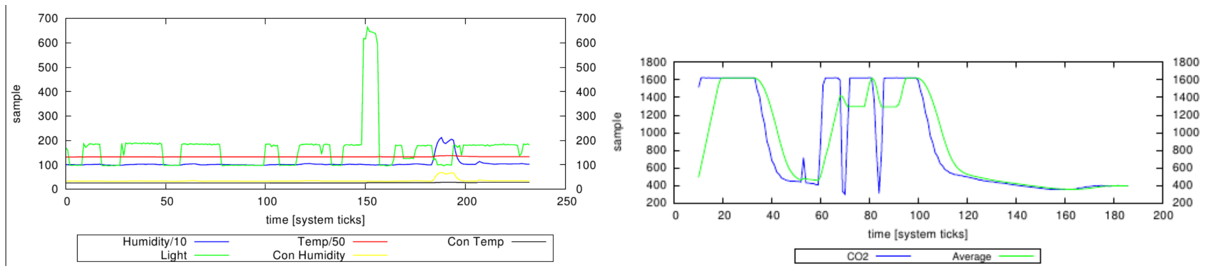

Figure 1.2: Left: the sample collection of temperature, humidity and light
Right: the sample collection for CO_2

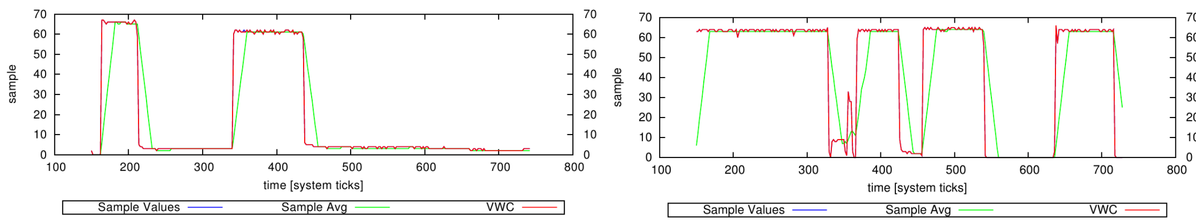

Figure 1.3: Left: Sample collection for left soil moisture sensor
Right: Sample collection for right soil moisture sensor

1.6 Final Evaluation Data Visualization

In the figures 1.2 and 1.3, the visualization of the final evaluation cycle is shown. Figure 1.2 on left shows the light, humidity and temperature graphs over time and on right it shows the CO_2 graph. Similarly, in figure 1.3 left, the soil moisture sensor data for left sensor is shown and on the right the data for the right moisture sensor is shown.

1.7 Team Segregation

In the table 1.2 the work allocation of group members is shown.

Manisha L.	Vaibhav S.	Boni T.	Johny M.	Ravi KS
Light, Temperature and Humidity Sensor	CO_2 Sensor	CO_2 Sensor	Soil Moisture Sensor	Soil Moisture Sensor (Coordination)

Table 1.2: Team Work Segregation

1.8 References

VH400 Data conversion: http://www.vegetronix.com/Products/VH400/VH400-Piecewise-Curve.phtml
VH400 Graph for VWC values : http://www.vegetronix.com/Curves/VH400-RevA/VG400-RevA-Curves.phtml
CO_2 sensor ref: http://www.epssilon.cl/files/DS1000.pdf, http://www.advanticsys.com/shop/mtsds1000-p-13.html
SHT11 Sensors: http://www.micromegacorp.com/downloads/documentation/AN033-V3%20Sensirion%20SHT11.pdf