

Sensor Network Playground Monitoring System

WSN Lab (WS 13/14)

Group Number 1

Group Members :

Arjun Maryankandy Rajendran

Konstantin März

Mohammed Refat Chowdhury

Pratyush Agnihotri

Rick Nitsche

Project Guide: Eugen Berlin



TECHNISCHE
UNIVERSITÄT
DARMSTADT



1. Introduction

Plants are the backbone of all life on Earth and an essential resource for human well-being. We depend on them not only for food but also most importantly for primary life-support: „Oxygen“. Since 1750, concentration of greenhouse gases has quickly increased. The coined global climate change have had enormous impacts on plant diversity patterns in the past and are seen as having significant current impacts. **Wireless sensor networks** provide a good platform to keep track of enviromental behavior. Using a system of sensor nodes one can capture, store, analyze, manage, and presents data which can be used in turns, for monitoring the system.

1.1. Problem Definition

The Goal of the project is to study the living conditions of two known plants, *Peperomia (Radiator Plant)* and *Kanlanchoe* and build a sensing system to monitor their requirements, as of, to give them a sustainable environment. The monitoring requirements of the plants includes air humidity, temperature, CO2 concentration, soil moisture and light exposure. In order to provide the plants with a nourished environment we took help of actuators such as a fan, a lamp and a heater, to supply them with fresh air, light at correct times and adjust the ambient temperature, respectively.

1.2. Plant Living Condition

We indulged ourselves into a lot of research to study growing conditions^[1] of the plants *Peperomia* and *Kalanchoe* defined as follows:-

Peperomia(Radiator Plant)	Kalanchoe
<ul style="list-style-type: none">• Air humidity: Love warm humid conditions.• CO2 concentration: Between 40 - 950 ppm.• Temperature: Optimal temperature should be 24-28°C (75-82°F) and minimal temperature should be 13-15°C (55-59°F).• Soil moisture: Water range should be normal and avoid wetting the crown of the plant.• Light exposure: There should be bright and mid-shade. Best cultivated in light.	<ul style="list-style-type: none">• Air humidity: Plenty of air flow around plant material.• CO2 concentration: Between 40 - 950 ppm.• Temperature: Minimal is 55°F (12.8 °C). It should be 10- 21°C (50-70 °F) in day and 7.2-18.3°C(45-65 °F) in night.• Soil moisture: It can be damaged by over watering. Allow the soil to dry slightly between watering.• Light exposure: There should be full sun light.

Table 1 : Plant living condition

1.3. Challenges

While building the monitoring system for plants, we faced several challenges which are quite common in every wireless sensor networks environment. We would like to state some of them, and also state our resolution we opted for in order to overcome these issues:-

Challenges faced	Description	Resolution opted
Energy-requirements	Lifetime of sensor nodes, energy consumption and battery usage	Since the monitoring system had to be of a longer lifetime, we decided to keep our sensor nodes active throughout, by connecting them to our centralized university test bed TUDμNet.
Robustness and reliability	Node and communication failures	For the node failures, we decided to simply replace the sensor node (eg. we replaced a soldered soil moisture sensor when it wasn't working fine). To handle

		communication failures, such as packet loss, we opted for <i>unicast communication protocol</i> that retransmit packets in case of communication failures.
Limited Resources	Memory and computation (handling floating point operations)	We used <i>unsigned integer</i> (data type) of smallest possible sizes, for most of our operations, wherever required. Rather than transmitting the values(<i>float</i> or <i>double</i>), we preferred to perform computation locally. (For instance we compared the measured sensor values to the threshold values - ideal conditions of plants, locally at the sensor node itself).
Calibration	Handling sensors actual output (raw values) w.r.t. the living conditions of the plants.	We converted the raw values received from sensors using the formulas in the datasheets ^[7,8,9,10] , to adhere them to the living conditions of the plants.

Table 2 : Challenges faced during project, their description and resolution opted.

2. Soldering of Sensors

2.1. Hardware used

We used following sensors and sensor nodes ^[2] to develop the monitoring system :

Basic Node	On-Board Sensors	CO2 Sensor	Soil Moisture Sensor	Base Station
XM1000	Inbuilt light sensors (Hamamatsu), Temperature and humidity sensors (SHT11)	DS1000	VH400	Raspberry Pi

Table 3 : Device/Driver information

To utilize the above sensors (like CO2 and soil moisture), we made use of the off the shelf available sensor nodes (XM1000), so that we could successfully be able to utilize the drivers and the sensors, as explained below ^[2,2].

2.2. Soldering

The XM1000 includes an expansion connector that allows the access to a number of pins in the microcontroller ^[3].

The *Soil moisture sensor (VH400)* has three wires which have specific identities as Bare : Ground , Red : Power, Black output. We soldered PIN 1(DVCC) : Red wire, PIN 3(ADC0) : Black wire and PIN 9(GND) : Bare wire ^[3].

For *soldering Co2 sensor*, we required xm1000 nodes, CO2 sensor (DS1000), PCB board and SH-300-DG. We soldered all these component together and used a DC 9V adapter jack to connect the DS1000 Board and the SH-300-DC for power supply ^[3].



Figure 1 : moisture sensor



Figure 2 : CO2 sensor

3. Test Bed Setup

3.1. Design Architecture

To create a better monitoring system, it is necessary to manage and place the sensor nodes and sensors, in such way that monitoring can be done efficiently. Therefore we proposed our test bed setup ^[4]. However, we used the common test bed setup that was also somewhat same (instead of

two sinks we recommended to use one sink) as defined in Figure 3. In this setup, there are total 6 nodes. Two nodes are placed at left and right side of the plants which are used as sinks. One CO₂ sensor is placed between the two plants and one on-board sensor node is placed at the left side of the plant. Two dedicated soil moisture sensors are used for one plant each. On-board sensor node (light, humidity and temperature), the CO₂ sensor and the soil moisture sensors collect sensor values, convert them, compare them to the threshold, check the local state of the sensor and send commands to the sink for handling actuators. For minimizing the number of packets

(commands) sent to the sink, we have saved and maintained the local state of each sensor as described in the following Section^[5,2]. If the sensor value have crossed the threshold since the last measurement, then command is sent to the sinks. After that, the commands are forwarded to the Raspberry Pi base station by the sink nodes, correspondly to handle the state of the actuators.

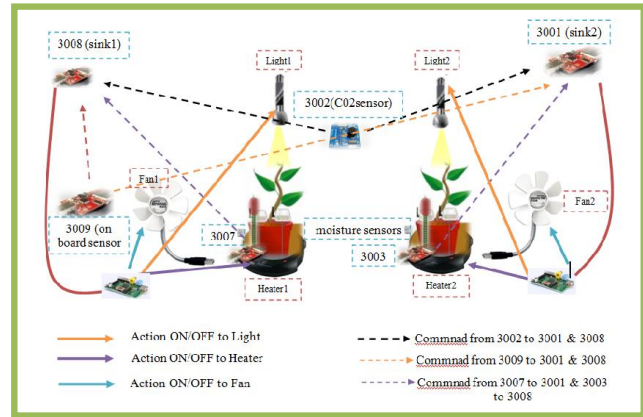


Figure 3 : Test Bed Setup

3.2. Usage of Actuators

In order to make use of the actuators, we referred to the API provided by our course supervisors^[5]. The sinks send “printf”-commands to the Raspberry Pi which in turn, controls the actuators. The API permits the control over three types of actuators via ON/OFF commands, *the fan*, *the lamp* and *the heating plate* placed under the plant. When the monitored value differs from the ideal values, a specific control command is sent to the base station (Raspberry Pi), e.g. PG:LIGHT ON to turn the light on [detailed information can be found in Actuators condition]^[6].

4. Formulas for Calculation

Monitoring requires converting raw data into human readable forms which will result in efficient sampling and visualization. We used various formulas to convert raw data into sensor values as raw data of CO₂ into ppm^[7], humidity into percentage^[8], temperature into °C^[8], light into LUX^[9], and soil moisture into VWC (Volumetric Water Content)^[10]. We converted these values for monitoring and visualization purpose while we used raw data for comparison with threshold value and sending commands.

5. Principal of Application

5.1. Introduction to Solution

To support readability, each sensor type has its own source file (c-file). That leaves the possibility to add additional sensor types in the future. A separate header file (settings.h) is used to define constant values such as node ID's, thresholds, measuring periods etc. After waiting 180 seconds for initialization of the sinks and Raspberry Pi boards, the nodes begin to operate. Every 5 seconds, a measurement procedure is invoked depending upon the respective node-ID (sensor types). Each procedure reads the sensor value and compares it with the corresponding thresholds. The respective value is then printed in readable units (e.g. LUX for the light). These print statements (“printf”s) are later used for the visualization of the measured values. To reduce energy consumption (minimize the number of packets sent), calculations are done on the sensor nodes, rather than sending all measured values to the sink node.

5.2. Local Sensor State

The local state of the sensor changes when the sensor value crosses the threshold. Every time the state change, a command is sent to the sink. This state variable prevents sending unnecessary actuator commands when the measured value stays constant. Each sensor node can maintain one of the three states: “value LOW”, “value OK” and “value HIGH”. For instance, if the CO₂ sensor value is higher than the upper threshold, we turn on the fan and save the state of the CO₂ sensor as “CO₂ HIGH”. Afterwards, if the sensor value remains constant for a certain period of time (i.e. high), we check for the state of the sensor, if it is already “CO₂ HIGH”, then the command is not sent to the sink again. However, if the state is default “CO₂ OK” and the sensor value obtained is higher than the threshold, the command (FAN ON) is sent to the sink and the state is saved as “CO₂ HIGH”. Also, if the sensor value is found to be low, we set the state as “CO₂ LOW” and print the statement “CO₂ LOW, plant name”, so that CO₂ can be provided to the respective plant. Maintaining the *local state* provides high efficiency in our code by significantly minimizing the number of packets sent to the sink. Also the property of using three states and checking the low value is unique in our approach, since we provide a check on the specific requirements of the respective plant (like low CO₂ and low soil moisture).

5.3. Routing Protocol

The *Reliable single-hop unicast (runicast)* primitive is used to prevent packet losses. The *runicast* primitive uses acknowledgments and retransmissions to ensure that packets are successfully received. The maximum number of retransmissions used in the project is 10. Different and short delays before sending were implemented for each sensor type in order to prevent occasional message losses due to simultaneous transmissions on the carrier.

6. Sampling and Visualization

6.1. Sampling



In the table below, the threshold values, based on the observations of the ideal living conditions for the plants^[Table 1] and adapted after running the code in the mid-term evaluation and checking system responsiveness are shown:-

	Air humidity(%)	CO ₂ (ppm)	Temperature(°C)	Soil Moisture (VWC)	Light Exposure (lux)
Peperomia	20 to 40	40 to 950	24 to 28	14.75 to 38.08	220
Kalanchoe	20 to 40	40 to 950	12.7 to 29	10 to 19.8	250

Table 4 : Threshold values

Finally, after collecting raw sensor values, the above thresholds are compared to these values, and send across the command (sample) to the sink. In order to *reduce the sampling rate*, we used the local sensor state mechanism as discussed above in Section ^[5.2].

6.2. Visualization

From the final evaluation of the code, following are the extracted visualization graphs of sampling. The sample values obtained from each of the sensor type is visualized by the Y-axis while the X-axis determines the system ticks(time in HH:MM:SS). The peak values  in the below graphs , range across the threshold determined by ^[Table 4]. After proper handling of the actuators, it is visible that values becomes constant over the time as per the needs of the respective plant. For instance, in Figure 8, when the sensor value determined is higher than our threshold 38.08 units(VWC), we turned on the heater, then the value significantly dropped upto 5 units  which is again less than our threshold

14.75 units, where we turned off the heater. (The sensor was put in pot of water and picked out of water). Followed by the other peaks, where our system responded well in terms of responsiveness and efficiency, which is significant from the below graphs:-

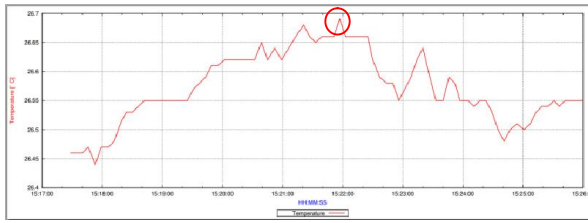


Figure 4 : Temperature

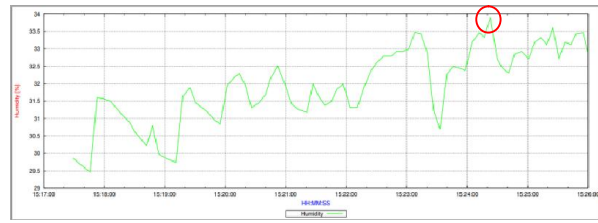


Figure 5 : Humidity

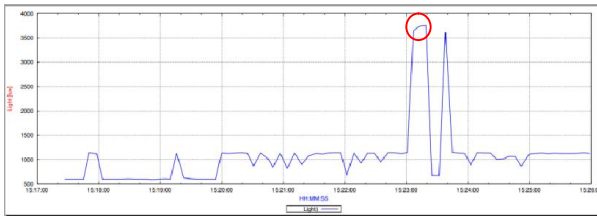


Figure 6 : Light

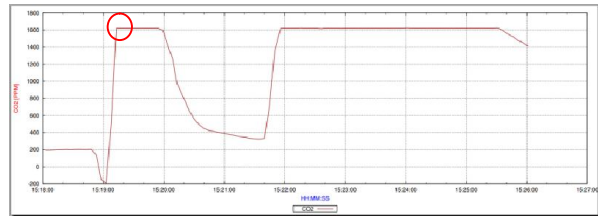


Figure 7 : CO2

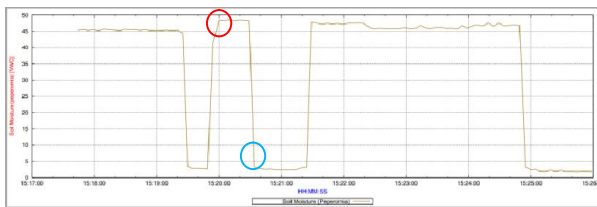


Figure 8 : Soil Moisture Peperomia

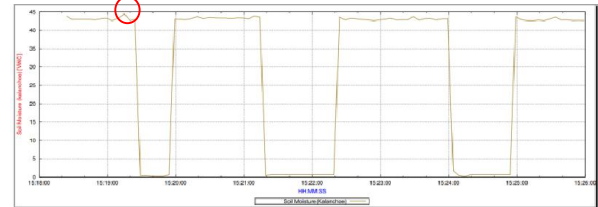


Figure 9 : Soil Moisture Kalanchoe

7. Testing of Application

To make the system more reliable and efficient, it is required to test the application. Testing does not only help in finding bugs but also in making the code more optimized and in providing efficiency to the application. We tested our application in various scenarios by applying different test cases^[11] to check how the actuators behave in these scenarios. Testing included checking the actuator functionality, run-time monitoring and system responsiveness by increasing/decreasing the soil moisture, lowering/highering the light exposure, increasing/decreasing the CO2 level w.r.t the threshold values given^[Table 4].

References

1. Plant Living Condition - ..\Project_Documents\Plant_condition\
2. Nodes and Sensors - ..\Project_Documents\Plant_condition\
3. Soldering of Sensors - ..\Project_Documents\Plant_condition\
4. TEST_BED_SET_UP - ..\Project_Documents\Test bed Step\
5. Actuators_API - ..\Project_Documents\Actuator_API\
6. Actuators condition - ..\Project_Documents\Actuators_condition\
7. DS1000 data sheet - ..\Project_Documents\Sensors_Datasheet\
8. SHT11 data sheet - ..\Project_Documents\Sensors_Datasheet\
9. http://www.advanticsys.com/wiki/index.php?title=Hamamatsu%C2%AE_S1087_Series
10. <http://www.vegetronix.com/Products/VH400/VH400-Piecewise-Curve.phtml>
11. Test_Cases - ..\Project_Documents\Test_cases\