|  |  |
| --- | --- |
|  |  |
|  | Sensor Network Playground Monitoring System |
|  |  |
|  |  |
|  | WSN Lab (WS 13/14)  Group number 1 Project Guide: Eugen Berlin  Group Members :  Arjun Maryankandy Rajendran  Konstantin März  Mohammed Refat Chowdhury  Pratyush Agnihotri  Rick Nitsche |
|  |  |





# 

# Introduction

We live in world surrounded by living and non-living things. However there are things which depends on us for their needs in same way we relay on them for our needs. Plants are one of them. We depend on them not only for food but also for our most important living requirement „oxygen“. A **Wireless Sensor Networks,** can provide best solution and monitoring system to save our friend plants. It is a system of sensor nodes that captures, stores, analyzes, manages, and presents data that are linked to location. It provides user with better and more efficient means of managing, monitoring, utilization of resources and keeping a record to maintain the ideal condition for plants.

## Problem Definition

The project is basically an implementation to degrade the greenhouse effect on plant using Wireless Sensors Networks. Concentration of greenhouse gases increased in last 10 years.. In result, gases like CO2 and CO increase with higher rate that can cause adverse effect on plants such as requires more water, some plants reduce their ability to do photosynthesis, plants become more vulnerable to insects, pests, and diseases.

## Goal

We built a monitoring system which can save the plants by monitoring plant’s surrounding enviornement and maintaining the ideal conditions for plant. To accomplish this task, we learnt about living conditions for to known plants : Peperpmia and Kalachoe. Furthermore, we built a sensing system to monitor air humidity, temperature, CO2 concentration, soil moisture and light exposure and expolit actuators (fan, heater and lamp) to emulate those condition.

## Challenges

Wireless sensor network is an unreliable communication which can make an efficient system to cause failure. Hence, we needed to consider all the points before and during building our monitoring system that can maintain the reliability of our system such as :

|  |  |  |
| --- | --- | --- |
| **Challenges** |  | **Description** |
| Energy-efficiency | : | Life Time, energy consumption, battery usage |
| Robustness and reliability | : | Node and communication failures, incorrect sensor readings |
| Limited Resources | : | Memory and computation (handling floating point operations) |
| Calibration | : | Handling sensors (actual output) raw values as per the living conditions of plants. |
| Cost | : | Cost per node, cost per sensors (CO2, Soil moisture), cost per actuator (fan,light and heater) |

# Device/ driver information and plant living condtion

## Plant living Condition

To create a better monitoring, it was required to know the ideal living conditions[[2]](#_References) for given plants. We indulged ourselves into a lot of research before we started the actual work on the “Save the Plant !”. We studied about the different kinds of necessary living requirements for plants as air humidity, CO2 concentration, temperature, soil moisture and light exposure that were needed to build our application.

|  |  |  |
| --- | --- | --- |
| **Peperomia(Radiator Plant)** | | **Kalanchoe** |
| * **Air humidity:** Love warm humid conditions. * **CO2 concentration:** It should be between 40 ppm to 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:** Soil range should be sandy Loam to Loam. Water range should be normal and avoiding wetting the crown of the plant. * **Light exposure:** There should be bright and mid-shade. Best cultivated in a light. | * **Air humidity:** Plenty of air flow around plant material. * **CO2 concentration:** It should be between 40 ppm to 950 ppm. * **Temperature:** It should not be fall below 55ºF (12.8 ºC). It should be 10- 21°C (50-70 °F ) in day and 7.2-18.3ºC(45-65 °F) in night. * **Soil moisture:** It can be damaged by over watering. Allow the soil to dry slightly between watering. * **Light exposure:** It should be full sun light. | |

## Device/driver Information

To use sensor in better way, it was required to study them both theoratically and pratically. We studied about the drivers and programmed them to know the behavior of them. We used following sensor and nodes[[1]](#_References) do develop a monitoring system :

|  |  |  |
| --- | --- | --- |
| Basic Node | : | XM1000 (aka TelosB) as basic node |
| On Board Sensors | : | Light sensors (Hamamatsu) |
|  |  | Temperature and humidity sensors (SHT11) |
| Co2 Sensor | : | DS1000 as gas sensor (CO, CO2 ) |
| Soil Moisture Sensor | : | VH400 as soil moisture sensor |
| Base Station | : | Raspberry Pi as base station |

# Soldering of Sensors

## XM1000 pin structure

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

## Soldering Soil Moisture sensor



Soil moisture sensor (VH400) has three wire which has specific identity as

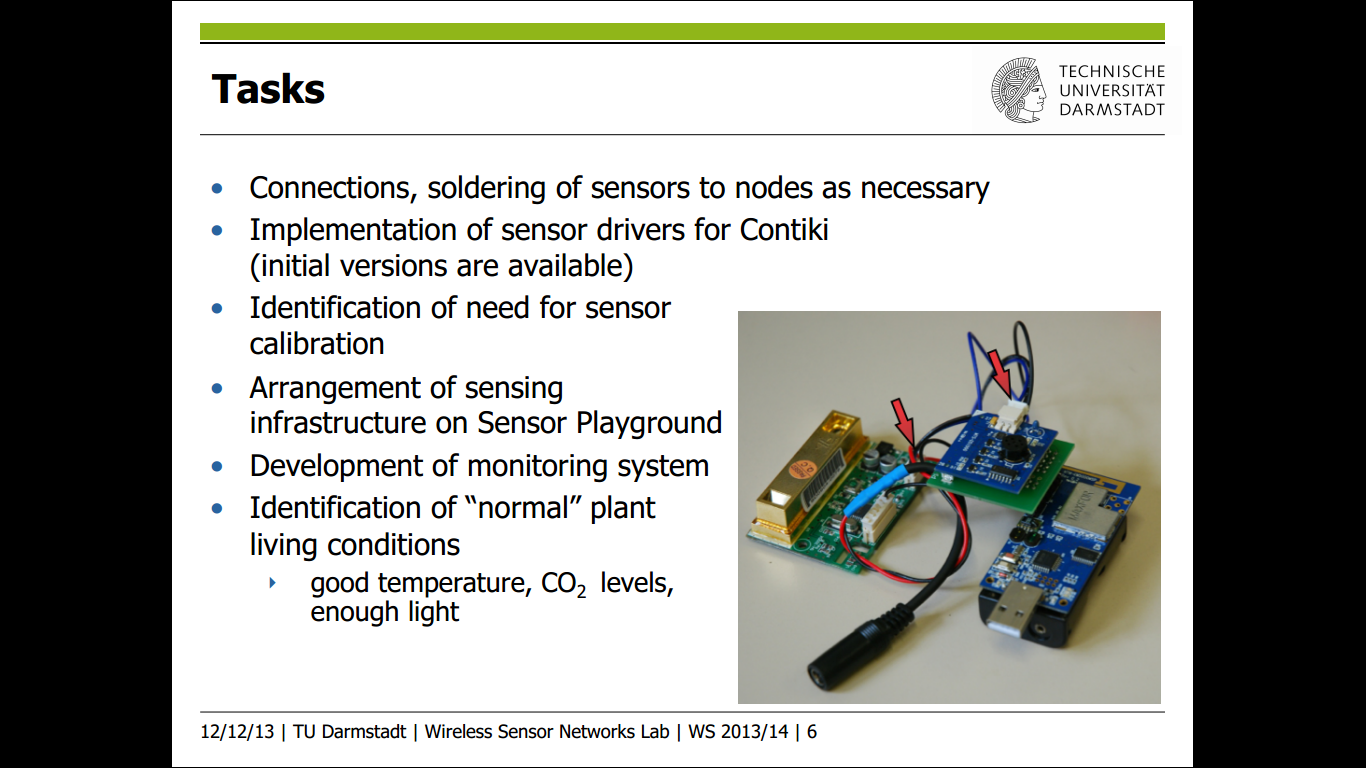
Bare : Ground , Red : Power, Black output. We used xm1000 pin structure to

solder moisture sensor. We soldered PIN 1(DVCC) : Red wire, PIN 3(ADC0) :

Black wire and PIN 9(GND) : Bare wire[[3]](#_References).

Figure 1 : Soldered moisture sensor

## Soldering CO2 sensor

****

For soldering Co2 sensor required xm1000 nodes, CO2 sensor (DS1000), PCB

board and SH-300-DG. We soldered all these compontent together and we used

DC 9V adapter jack to connect 5. DS1000 Board and SH-300-DC for power supply[[3]](#_References).

Figure 2 : Soldered CO2 sensor

# Test Bed Set up

## Design Architecture

To create better monitoring system, it is necessary to manage and place all nodes and sensor in such way so that monitoring can be done in efficient way. Therefore we proposed our test bed setup[[4]](#_References).However, a common test bed setup used by us. There were total 6 nodes. Two nodes are placed a left and right sides of the plants which were used as a sink. One CO2 sensor was placed between two plants and one on board sensor was placed at left side of plant. However, two dedicated soil moisture were used for left and right plants. On board Sensor (light, humidity and temperature), Co2 and moisture sensor (both left and right) collected a values and did

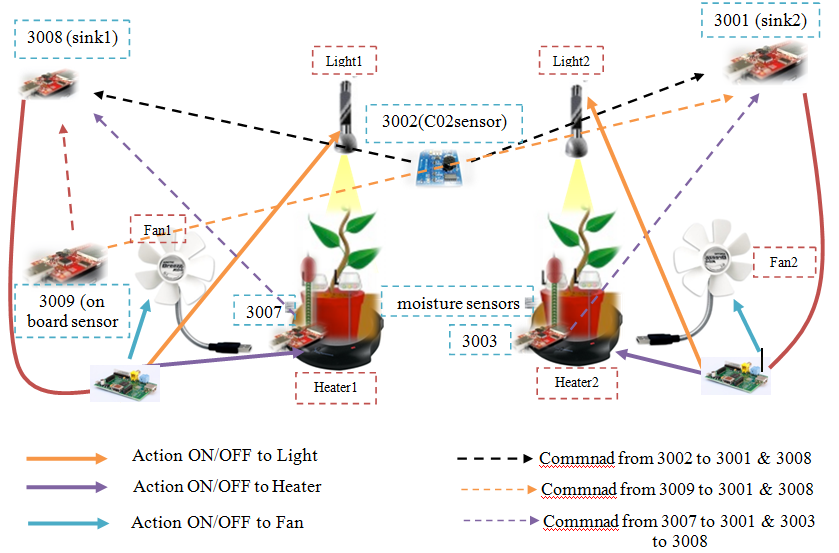


Figure 3 : Test Bed Setup

sampling. If value was below /Ok/ higher than threshold value then send command to sink 3001 and

3008. When commands was received by sink 3001 and 3008, it sent ON/OFF action to raspberry pi base station to handle corresponding actuators. Current states(actuator On/Off) stored in the sink 3001 and 3009 for respective plants. No action was sent by sink until new command was received.

## Usage of actuator

To manipulate the actuators we used the API which was provided by our course supervisors. The sinks send “printf”-commands[[6]](http://en.wikipedia.org/wiki/Device_driver#cite_note-dev1-1) to the Raspberry Pi which controls the actuators. In the current state the API permits the control over three types of actuators via ON/OFF commands, a fan, a lamp and a heating plate under the plant. If the monitored value differs from the ideal value, 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 ] [[11]](http://en.wikipedia.org/wiki/Device_driver#cite_note-dev1-1). As for now the system only respond to changes in light, CO2 and soil moisture. Temperature or humidity changes are monitored but no commands are sent to the actuators since these values don’t tend to fluctuate too much.

# Formulas for calculation

Monitoring requires converting raw data into human readable forms which we result in efficient sampling and visualization. We used various formulas to convert raw data into correspondind sensor values as raw data of Light into LUX[[8]](http://en.wikipedia.org/wiki/Device_driver#cite_note-dev1-1), Humidity into percentage[[9]](http://en.wikipedia.org/wiki/Device_driver#cite_note-dev1-1), Temperature into ºC[[10]](http://en.wikipedia.org/wiki/Device_driver#cite_note-dev1-1), Co2 into ppm[[11]](http://en.wikipedia.org/wiki/Device_driver#cite_note-dev1-1), and soil moisture into VWC (Volumetric Water Content) [[12]](http://en.wikipedia.org/wiki/Device_driver#cite_note-dev1-1). We converter these value for monitoring and visualization purpose while we used raw data for comparison with threshold value and sending command correspondingly.

# Principal of application

To support modularity each sensor type has its own .c-file. It leaves a possibility to add additional sensor types in the future. A settings file is used to define constant values such as node ID's, value 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 measurements are taken; depending on the node-ID a different measurement procedure is invoked. Each procedure reads the sensor value and compares it with the corresponding threshold. The reading is then printed in readable units (e.g. LUX for the light). These “printf”s are later used for the visualization of the measured values. To reduce energy consumption calculations are done on the sensor nodes, rather than sending measured values to the sink node every time measurements are taken. Instead the commands to manipulate the actuators are only send to the sink node if a measured value crosses a certain threshold and only on state changes. A state variable prevents sending unnecessary actuator commands if the measured values stay constant. Three state types are used: valueLOW, valueOK, valueHIGH. 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 was 10. Different but short delays before sending were implemented for each sensor type in order to prevent occasional message loses due to simultaneous transmissions on the carrier.

# Sampling and visualization

To be done

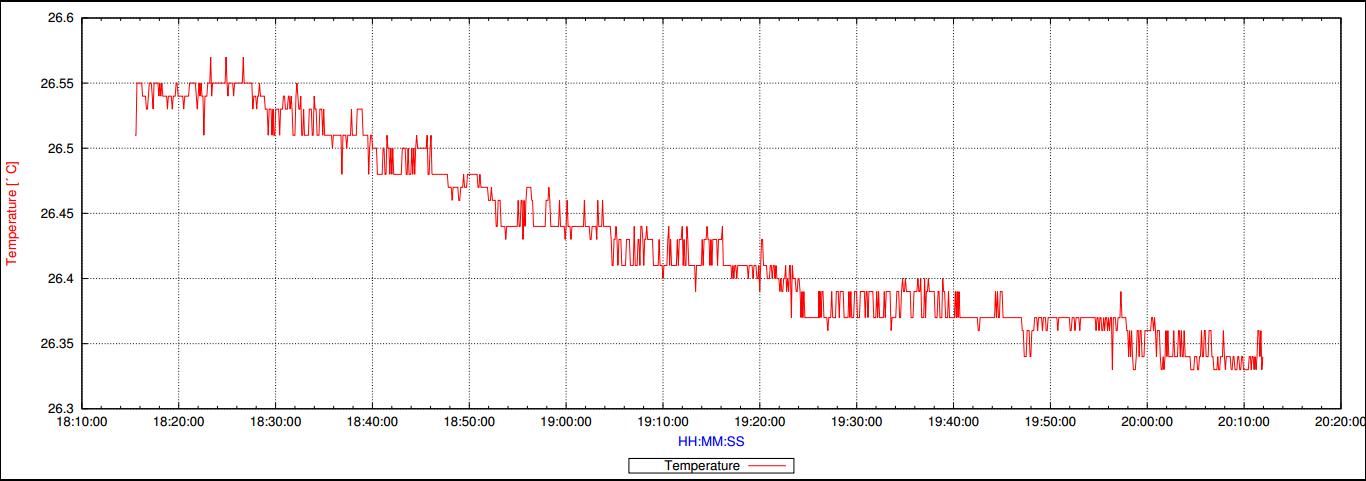


Figure 4 : Temperature

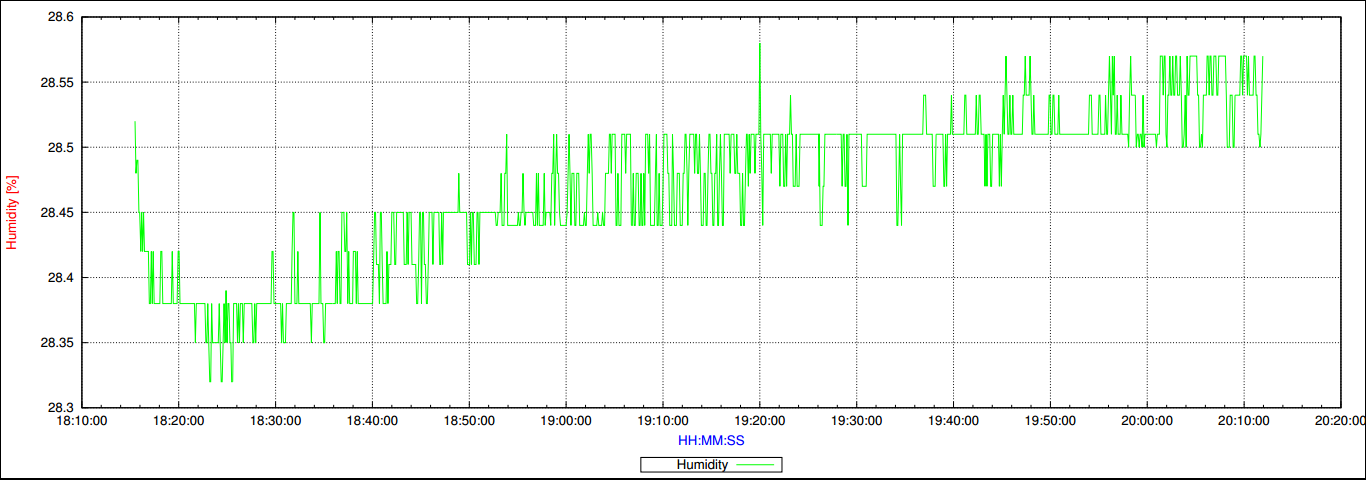


Figure 5 : Humidity

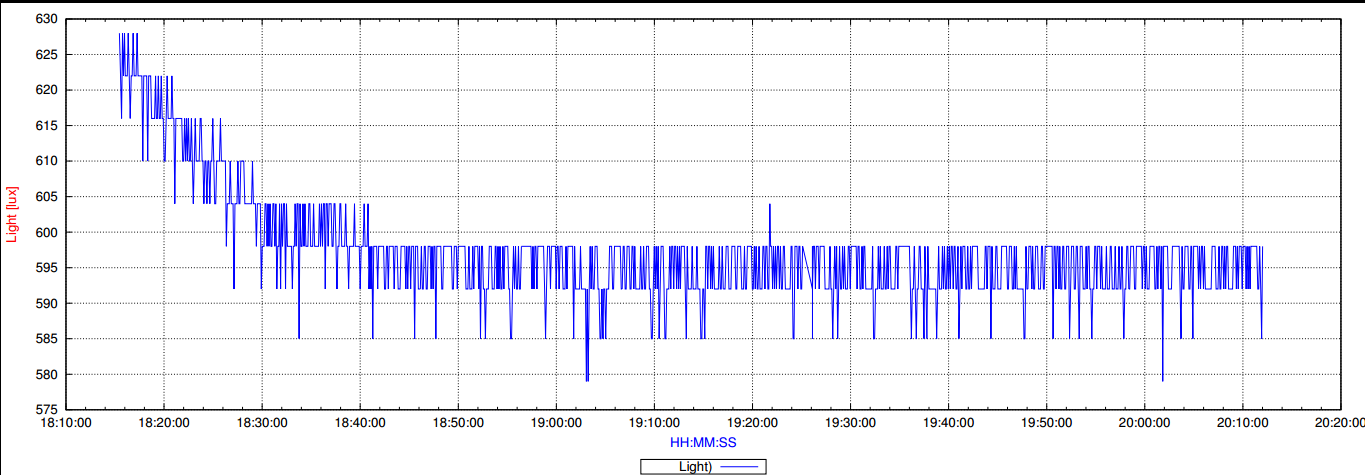


Figure 6 : Light

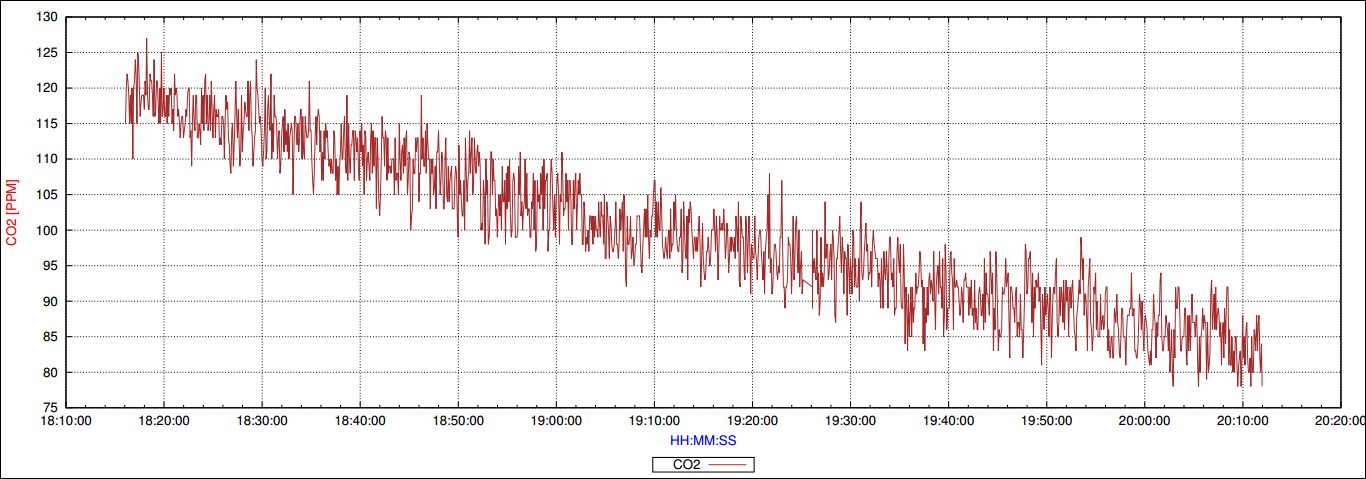
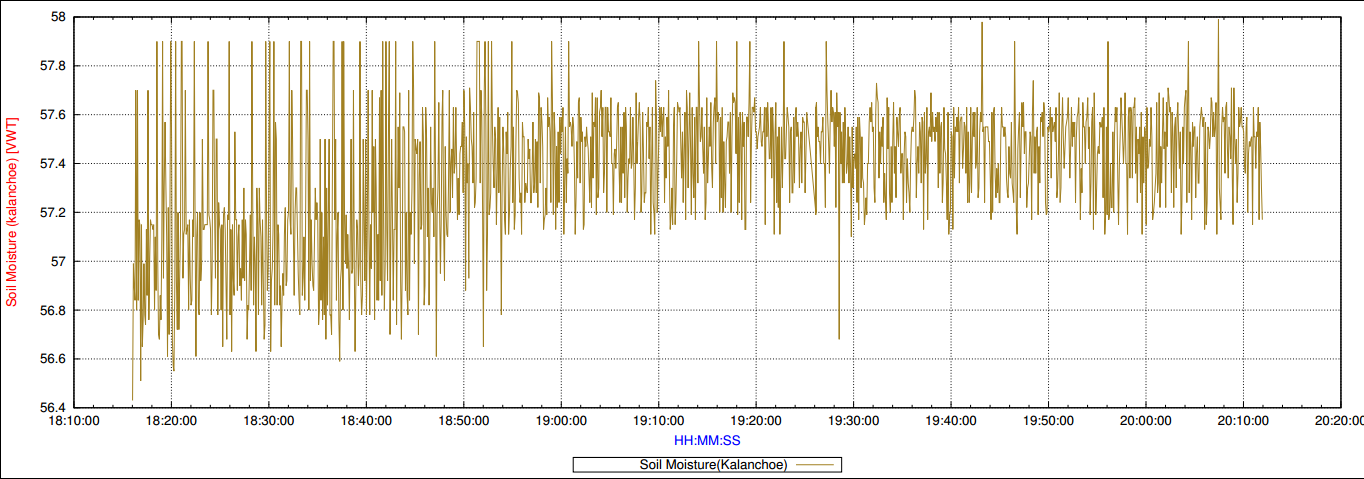


Figure 7 : CO2



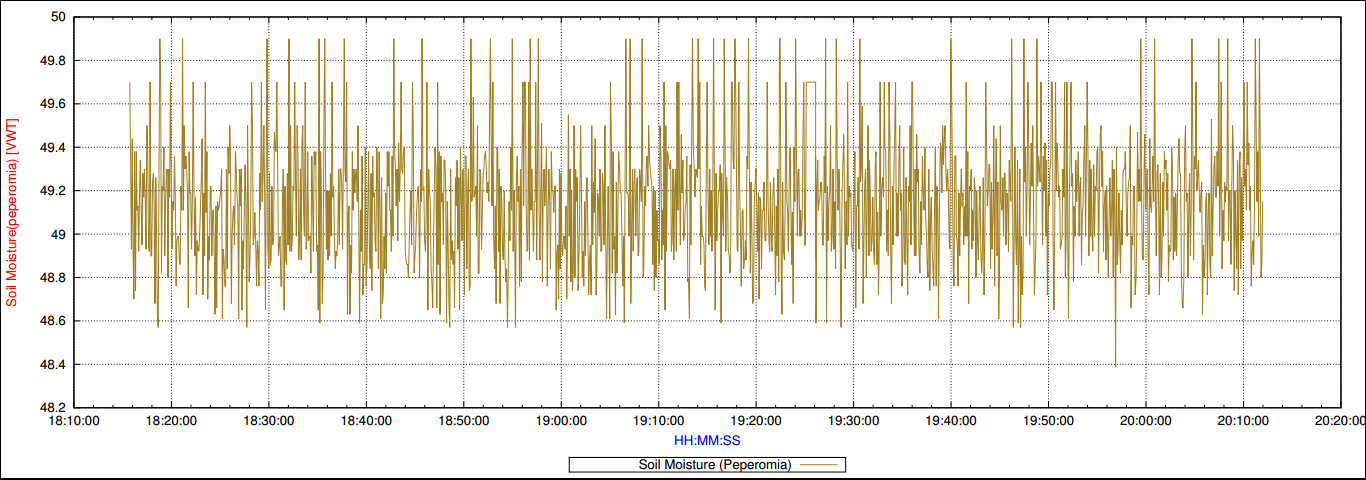


Figure 8 : Soil Moisture Kalanchoe

Figure 9 : Soil Moisture Peperomia

# Testing of application

To make system more reliable and efficient, it is required to do testing of application. Testing does not only help in solving bugs but also make code more optimizie and provide efficient application. We tested our application in various senarios by applying different test cases[[13]](#_References) to check how actuators behave in these scenarios. Testing included checking the actuator functionality, run-time monitoring and system responsiveness by increasing/decreaseing soil moisture, lowering/highering light exposure, increasing/decreasing CO2 level and rigorous testing.

# References

1. Nodes and Sensors - ..\Project\_Documents\Plant\_condition\
2. Plant Living Condition - ..\Project\_Documents\Plant\_condition\
3. Soldering of Sensors - ..\Project\_Documents\Plant\_condition\
4. TEST\_BED\_SET\_UP - .. \Project\_Documents\Test bed Step\
5. http://www.advanticsys.com/wiki/index.php?title=Hamamatsu%C2%AE\_S1087\_Series
6. Actuators\_API - ..\Project\_Documents\Actuator\_API\
7. http://www.advanticsys.com/wiki/index.php?title=Sensirion%C2%AE\_SHT11
8. DS1000 data sheet - ..\Project\_Documents\Sensors\_Datasheet\
9. SHT11 data sheet - ..\Project\_Documents\Sensors\_Datasheet\
10. http://www.vegetronix.com/Products/VH400/VH400-Piecewise-Curve.phtml
11. Actuators condition - ..\Project\_Documents\Actuators\_condition\
12. Test\_Cases - ..\Project\_Documents\Test\_cases\