*Chi Nguyen*

Case, parts 1-3

Software Engineering

# PART 1: REQUIREMENTS ANALYSIS AND HIGH-LEVEL DESIGN

## Requirements

Functional requirements:

* The operator shall be able to track the exact time and sensor when the measurements series contain temperatures outside the limit using the sensor system.
* The operator shall be able to track whenever any of the data seems to be corrupted in the files containing the measurements using the sensor system.
* The operator shall be able to see the lowest and highest measured temperature, and the most recent measurements using the sensor system.
* The administrator shall be able to use the sensor system the same way as the operator.
* The administrator should be abke to maintain the configuration of the number of sensors and the measurements per hour interval.

Non-functional requirements:

* The software shall not require graphical user interface capabilities.
* The software shall be developed in a common programming language and other libraries available for the Rasberry Pi.

## UML use case diagram

*Insert your diagram here.*

# LOW LEVEL DESIGN AND UNIT TESTING

## UML class diagram

*Insert your diagram here.*

## Planned test cases for unit testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Description** | **Target(s)** | **Precondition(s)** | **Input(s)** | **Expected outcome(s)**  **(postcondition)** |
| TC1 | Correct limits | Function check\_limits | None | limits == [8,16] | Function returns True |
| TC2 | Incorrect limits | Function check\_limits | None | limits == [16,8] | Function returns False |
| TC3 | Sensors are read | Function read\_sensors | None | None | Results are returned |
| TC4 | Correct number of results is returned for each sensor | Function read\_sensors | None | None | 5 results are returned for each sensor |

## Implemented unit test cases and results

Test case 1 and test case 2 check the funtionability of function check\_limits.

(insert function\_limit code from sensors\_main)

This function gets an array containing the limits as a parameter and checks that the lower limit ( limit[0] ) is smaller than the higher limit ( limit[1] ). If this is the case, the function returns True. Otherwise, it returns False.

### Test case 1:

(insert code from test case 1)

In this test case test\_check\_limits1, there are correct inputs with 8 as a lower limit and 16 as a highter limit, hence it is expected that the method return True.

### Test case 2:

(insert code from test case 2)

In this test case test\_check\_limits2, there are correct inputs with 16 as a lower limit and 8 as a highter limit, hence it is expected that the method return False.

Test case 3 and test case 4 check the funtionability of function read\_sensors.

(insert code read\_sensors)

This function returns a fixed sensor readings (four sensors, five readings per sensor) for development and testing

### Test case 3:

(insert code from test case 3)

The test case test\_read\_sensors0 tests whether function read\_sensors return any sensor readings. It indeed return a result that got printed into the terminal.

(insert the print part)

### Test case 4:

(insert code from test case 4)

The test case test\_read\_sensors2 tests whether function read\_sensors return 5 result for each sensor reading.

This test failed. The reason is that in return part from the function read\_sensors in sensors\_main.py, when look into the sensors reading, we can notice that some of them only have 4 readings

(insert that part here)

# INTEGRATION AND SYSTEM TESTING & CHANGEOVER

## Planned test cases for integration and system testing

### Integration test case plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Description** | **Target(s)** | **Precondition(s)** | **Input(s)** | **Expected outcome(s)**  **(postcondition)** |
| TC5 | Test integration between main() and check\_limits() | Module  sensors\_main, function main() and  check\_limits() | Function check\_limits() has been unit tested. Function main is runnable, it is able to parse the command line arguments and the call from it to check\_limits() has been implemented. | Command line parameters (min. and max. temperature) 8 and 16. | In main, error message “Error: Incorrect command line arguments.” is printed to the console. |

### System test case plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Description** | **Inputs** | **Expected outputs(s) or results(s)** | **Actual output(s) or results(s)** | **Pass/ Fail & notes** |
| 1 | Operator starts the sensors\_main.py from command prompt | Command line  parameters 8 16 | - | - | Pass |
| 2 | Software starts and notices the correct command line  parameters | - | Console print out the sensors readings | [21.2, 18.2, 18.2, 22.2],  [-5.0, -4.2, -3.9, -4.5],  [1.2, 0.0, 0.5, -0.8, -1.0],  [25.0, -4.2, -13.9, 4.5] | Pass |
| 3 | Software exist and the command prompt is shown | - | Command prompt is shown | - | Pass |

## Implemented test case and result

*(insert source code)*

The test case passed, as the command line arguments are correct with the lower limit 8 is actually lower than the higher limit 16 and the call returned the sensors readings like in the file sensors\_main. We can also see what is in mock\_print using this line:

## Roadmap

*Insert your roadmap here.*