Sentient VOC Monitoring System

Read VOC Levels from Sensor

Version <4.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <02/08/14> | <1.0> | Document Creation | Thyanna Voisine |
| <02/09/14> | <2.0> | Changed Actor’s Name | Thyanna Voisine |
| <02/09/14> | <3.0> | Added Timer Component to Scenarios | Thyanna Voisine |
| <10/03/14> | <4.0> | Adjusted to Match System | Thyanna Voisine |

Table of Contents

1. Read VOC Levels from Sensor 1

1.1 Brief Description 1

1.2 Requirements Trace 1

1.3 Involved Actors 1

1.4 Preconditions 1

1.5 Post conditions 1

1.6 Invariants 1

2. Flow of Events 1

2.1 Basic Flow 1

2.2 Alternate Course – None 1

3. Extension Points 1

4. Scenarios 2

4.1 Happy Day 2

4.2 Rainy Day 1 – VOC Level out of range 2

4.3 Rainy Day 2 –VOC Level retrieval gap 2

5. Testing

5.1 File Storage Test 3

Use Case: Read VOC Levels from Sensor

# Read VOC Levels from Sensor

## Brief Description

This use case gives the Embedded VOC system the ability to store analog VOC levels. This will allow the Embedded VOC system to eventually translate VOC level from analog to parts per millions.

## Requirements Trace

13

## Involved Actors

Arduino Clock

## Preconditions

* The Embedded VOC system is initialized.

## Post conditions

* The Embedded VOC system has VOC levels stored in persistence layer.

## Invariants

* The VOC levels retrieved are within the sensor’s accuracy range.

# Flow of Events

## Basic Flow

This use case starts when the Arduino Clock wants to retrieve VOC data

1. VOC Controller retrieves VOC levels continuously.
2. Arduino Clock keeps track of hour marker.
3. Arduino Clock notifies VOC Controller when the hour mark is met.
4. Arduino Clock sets the time and date.
5. VOC Controller sends VOC level to Arduino Persistence Layer.
6. Arduino Persistence Layer ensures analog value is with VOC Controller accuracy range.
7. Persistence layer sets accuracy flag to zero.
8. Arduino Persistence Layer stores analog value, date, time, and accuracy flag.

## Alternate Course – None

# Extension Points

None

# Scenarios

## Happy Day

Assumptions: Arduino Clock – Mr.Clock

VOC Controller- CO Sensor

Arduino Persistence Layer – Mr. Storage

Sensor Accuracy Range: 555-570

Steps:

1. CO Sensor retrieves CO levels continuously.
2. Mr. Clock keeps track of hour marker.
3. Mr. Clock notifies CO Sensor when the hour mark is met.
4. Mr. Clock sets the time and date.
5. CO Sensor sends CO level to Mr. Storage.
6. Mr. Storage ensures CO analog value is within the 555 – 570 accuracy range.
7. Mr. Storage sets accuracy flag to zero.
8. Mr. Storage stores analog value, date, time, and accuracy flag.

## Rainy Day 1 – VOC Level out of range

Assumptions: Arduino Clock – Mr.Clock

VOC Controller- CO Sensor

Arduino Persistence Layer – Mr. Storage

Sensor Accuracy Range: 555-570

Steps:

1. CO Sensor retrieves CO levels continuously.
2. Mr. Clock keeps track of hour marker.
3. Mr. Clock notifies CO Sensor when the hour mark is met.
4. Mr. Clock sets the time and date.
5. CO Sensor sends CO level to Mr. Storage.
6. Mr. Storage ensures CO analog value is within the 555 – 570 accuracy range.
7. CO analog value is below 555.
8. Mr. Storage sets accuracy flag to one to signal corrupt CO value to Sentient Management.
9. Mr. Storage stores analog value, date, time, and accuracy flag.

## Rainy Day 1 – Sensor level retrieval gap

Assumptions: Arduino Clock – Mr.Clock

VOC Controller- CO Sensor

Arduino Persistence Layer – Mr. Storage

Sensor Accuracy Range: 555-570

Steps:

1. CO Sensor retrieves CO levels continuously.

2. Mr. Clock keeps track of hour marker.

3. Mr. Clock notifies CO Sensor when the hour mark is met.

4. Mr. Clock sets the time and date.

5. CO Sensor sends CO level to Mr. Storage.

6. Mr. Storage ensures CO analog value is within the 555 – 570 accuracy range.

7. CO analog value is below 555.

8. Mr. Storage sets accuracy flag to one to signal CO gap to Sentient Management.

9. Mr. Storage stores analog value, date, time, and accuracy flag.

# Test

## File Storage Test

Precondition- Monitor is not yet implemented in network

Assumptions: Not working under rainy day scenario

Tester is in a clean air environment

Steps:

1. Run VOC Monitor for 24 hours
2. Stop the VOC Monitor
3. Remove SD Card from SD shield on Arduino
4. Insert SD card to volatile system ( Personal Computer )
5. Open SD card file on volatile system
6. Open the New File
7. The New File should appear as below

Example File (shortened version):

20,02/23/14,05,0; 20,02/23/14,06,0; 20,02/23/14,07,0; 20,02/23/14,08,0; 20,02/23/14,09,0; 20,02/23/14,10,0; 20,02/23/14,11,0; 20,02/23/14,12,0; 20,02/23/14,13,0; 20,02/23/14,14,0; 20,02/23/14,15,0; 20,02/23/14,16,0; 20,02/23/14,17,0; 20,02/23/14,18,0; 20,02/23/14,19,0;

1. VOC data comes in the following format: VOC level, date, hour, and accuracy flag. The VOC level for each VOC data segment must be 20. If all VOC level values are 20 for the New File, the test passes. If not all VOC level values are not 20 for the New File, the test fails.